aws

Unable to locate subtitle

AWS Glue DataBrew개발자 안내서

AWS Glue DataBrew개발자 안내서: ***Unable to locate subtitle***

Table of Contents

AWS Glue DataBrew란 무엇인가요?

AWS Glue DataBrew는 사용자가 코드를 작성하지 않고도 데이터를 정리하고 정규화할 수 있는 시각적 데이터 준비 도구입니다. DataBrew를 사용하면 분석 및 기계 학습(ML)을 위한 데이터를 준비하는 데 걸리는 시간을 사용자 지정 개발 데이터 준비에 비해 최대 80% 줄일 수 있습니다. 250개 이상의 미리 만들어진 변환 중에서 선택하여 이상 필터링, 데이터를 표준 형식으로 변환, 잘못된 값 수정과 같은 데이터 준비 작업을 자동화할 수 있습니다.

DataBrew를 사용하면 비즈니스 분석가, 데이터 과학자 및 데이터 엔지니어가 더 쉽게 협업하여 원시 데이터에서 인사이트를 얻을 수 있습니다. DataBrew는 서버리스이므로 기술 수준에 관계없이 클러스터를 생성하거나 인프라를 관리할 필요 없이 테라바이트의 원시 데이터를 탐색하고 변환할 수 있습니다.

직관적인 DataBrew 인터페이스를 사용하면 원시 데이터를 대화식으로 검색, 시각화, 정리 및 변환할 수 있습니다. DataBrew는 찾기 어렵고 수정하는 데 시간이 많이 걸릴 수 있는 데이터 품질 문제를 식별하는 데 도움이 되는 스마트 제안을 제공합니다. DataBrew에서 데이터를 준비하면 시간을 사용하여 결과에 따라 작업하고 더 빠르게 반복할 수 있습니다. 변환을 레시피의 단계로 저장할 수 있으며, 나중에 다른 데이터 세트와 함께 업데이트하거나 재사용하고 지속적으로 배포할 수 있습니다.

다음 이미지는 DataBrew가 높은 수준에서 작동하는 방식을 보여줍니다.

DataBrew를 사용하려면 프로젝트를 생성하고 데이터에 연결합니다. 프로젝트 워크스페이스에는 그리드와 같은 시각적 인터페이스에 데이터가 표시됩니다. 여기에서 데이터를 탐색하고 값 분포 및 차트를 확인하여 프로파일을 이해할 수 있습니다.

데이터를 준비하려면 250개 이상의 point-and-click 변환 중에서 선택할 수 있습니다. 여기에는 null 제거, 누락된 값 대체, 스키마 불일치 수정, 함수를 기반으로 열 생성 등이 포함됩니다. 변환을 사용하여 자연어 처리(NLP) 기술을 적용하여 문장을 문구로 분할할 수도 있습니다. 즉시 미리 보기는 변환 전후의 데이터 일부를 표시하므로 전체 데이터 세트에 적용하기 전에 레시피를 수정할 수 있습니다.

DataBrew가 데이터 세트에서 레시피를 실행한 후 출력은 Amazon Simple Storage Service(Amazon S3)에 저장됩니다. 정리되고 준비된 데이터 세트가 Amazon S3에 있으면 다른 데이터 스토리지 또는 데이터 관리 시스템에서 데이터를 수집할 수 있습니다.

의 핵심 개념 및 용어AWS Glue DataBrew

아래에서 핵심 개념 및 용어에 대한 개요를 확인할 수 있습니다AWS Glue DataBrew. 이 섹션을 읽은 후 프로젝트 생성, 데이터 세트 연결 및 작업 실행 프로세스를 안내시작하기AWS Glue DataBrew하는 섹션을 참조하세요.

주제

- Project
- 데이터세트
- 방법
- 작업
- 데이터 계보
- 데이터 프로필

Project

DataBrew의 대화형 데이터 준비 워크스페이스를 프로젝트라고 합니다. 데이터 프로젝트를 사용하여 데이터, 변환 및 예약된 프로세스와 같은 관련 항목 모음을 관리합니다. 프로젝트를 생성하는 과정에서 작업할 데이터 세트를 선택하거나 생성합니다. 다음으로 DataBrew가 실행할 일련의 지침 또는 단계인 레시피를 생성합니다. 이러한 작업은 원시 데이터를 데이터 파이프라인에서 사용할 준비가 된 형식으로 변환합니다.

데이터세트

데이터 세트는 단순히 열 또는 필드로 구분된 행 또는 레코드인 데이터 세트를 의미합니다. DataBrew 프로젝트를 생성할 때 변환하거나 준비하려는 데이터에 연결하거나 업로드합니다. DataBrew는 형식이 지정된 파일에서 가져온 모든 소스의 데이터로 작업할 수 있으며 증가하는 데이터 스토어 목록에 직접 연결됩니다.

DataBrew의 경우 데이터 세트는 데이터에 대한 읽기 전용 연결입니다. DataBrew는 데이터를 참조하기 위해 설명이 포함된 메타데이터 세트를 수집합니다. DataBrew는 실제 데이터를 변경하거나 저장할 수 없습니다. 간소화를 위해 데이터 세트를 사용하여 실제 데이터 세트와 DataBrew가 사용하는 메타데이터를 모두 참조합니다.

방법

DataBrew에서 레시피는 DataBrew가 작업할 데이터에 대한 일련의 지침 또는 단계입니다. 레시피에는 여러 단계가 포함될 수 있으며 각 단계에는 여러 작업이 포함될 수 있습니다. 도구 모음의 변환 도구를 사용하여 데이터에 적용하려는 모든 변경 사항을 설정합니다. 나중에 레시피의 완성 제품을 볼 준비가 되면이 작업을 DataBrew에 할당하고 예약합니다. DataBrew는 데이터 변환에 대한 지침을 저장하지만 실제 데이터는 저장하지 않습니다. 다른 프로젝트에서 레시피를 다운로드하여 재사용할 수 있습니다. 레시피의 여러 버전을 게시할 수도 있습니다.

작업

DataBrew는 레시피를 만들 때 설정한 지침을 실행하여 데이터를 변환하는 작업을 수행합니다. 이러한 지침을 실행하는 프로세스를 작업이라고 합니다. 작업은 사전 설정된 일정에 따라 데이터 레시피를 실행할 수 있습니다. 하지만 일정에 국한되지는 않습니다. 온디맨드로 작업을 실행할 수도 있습니다. 일부 데이터를 프로파일링하려는 경우 레시피가 필요하지 않습니다. 이 경우 프로필 작업을 설정하여 데이터 프로필을 생성할 수 있습니다.

데이터 계보

DataBrew는 시각적 인터페이스에서 데이터를 추적하여 데이터 계보라는 오리진을 결정합니다. 이 보기는 데이터가 원래 위치로부터 다른 엔터티를 통해 흐르는 방식을 보여줍니다. 오리진, 영향을 받은 다른 엔터티, 시간 경과에 따라 발생한 일, 저장된 위치를 확인할 수 있습니다.

데이터 프로필

데이터를 프로파일링하면 DataBrew는 데이터 프로파일이라는 보고서를 생성합니다. 이 요약은 콘텐츠의 컨텍스트, 데이터의 구조 및 관계를 포함하여 데이터의 기존 형태에 대해 설명합니다. 데이터 프로필 작업을 실행하여 모든 데이터 세트에 대한 데이터 프로필을 만들 수 있습니다.

제품 및 서비스 통합

이 섹션을 사용하여 DataBrew와 통합되는 제품 및 서비스를 확인할 수 있습니다.

DataBrew는 네트워킹, 관리 및 거버넌스를 위해 다음AWS서비스와 함께 작동합니다.

- Amazon CloudFront
- AWS CloudFormation
- AWS CloudTrail
- Amazon CloudWatch
- AWS Step Functions

DataBrew는 다음AWS데이터 레이크 및 데이터 스토어에서 작동합니다.

- AWS Lake Formation
- Amazon S3

DataBrew는 데이터 업로드를 위해 다음과 같은 파일 형식과 확장자를 지원합니다.

형식	파일 확장명(선택 사항)	압축 파일의 확장(필수)

쉼표로 구분된 값	`.csv`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Microsoft Excel 통합 문서	`.xlsx`	압축 지원 없음

형식	파일 확장명(선택 사항)	압축 파일의 확장(필수)

JSON(JSON 문서 및 JSON 줄)	`.json, .jsonl`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Apache ORC	`.orc`	`.zlib` `.snappy`

Apache Parquet	`.parquet`	`.gz`
		`.snappy`
		`.lz4`

DataBrew는 출력 파일을 Amazon S3에 쓰고 다음 파일 형식 및 확장자를 지원합니다.

형식	파일 확장명(압축되지 않음)	파일 확장명(압축)
쉼표로 구분된 값	`.csv`	`.csv.snappy` , `.csv.gz`, `.csv.lz4`, `csv.bz2`, `.csv.deflate` , `csv.br`
탭으로 구분된 값	`.csv`	`.tsv.snappy` , `.tsv.gz`, `.tsv.lz4`, `tsv.bz2`, `.tsv.deflate` , `tsv.br`
Apache Parquet	`.parquet`	`.parquet.snappy` , `.parquet.gz` , `.parquet.lz4` , `.parquet.lzo` , `.parquet.br`

형식	파일 확장명(압축되지 않음)	파일 확장명(압축)
AWS Glue Parquet	지원되지 않음	.glue.parquet.snappy
Apache Avro	.avro	.avro.snappy , .avro.gz, .avro.lz4 , .avro.bz2 , .avro.deflate , .avro.br
Apache ORC	.orc	.orc.snappy , .orc.lzo, .orc.zlib
XML	.xml	.xml.snappy , .xml.gz, .xml.lz4, .xml.bz2, .xml.deflate , .xml.br
JSON(JSON Lines 형식만 해당)	.json	.json.snappy , .json.gz, .json.lz4 , json.bz2, .json.deflate , .json.br
Tableau 하이퍼	지원되지 않음	해당 사항 없음

설AWS Glue DataBrew정

시작하기 전에 몇 가지 권한AWS Glue DataBrew, 사용자 및 역할을 설정해야 합니다. 먼저 다음 단계를 수행합니다.

1. 필요에 따라AWS계정에 가입하고 사용자가 DataBrew를 실행할 수 있도록AWS Identity and Access Management(IAM) 정책을 생성합니다.
   - 새AWS계정에 가입하고 사용자를 추가합니다. 자세한 내용은 새AWS계정 설정 단원을 참조하십시오.
   - 콘솔 사용자에 대한 IAM 정책 추가. 이러한 권한이 있는 사용자는에서 DataBrew에 액세스할 수 있습니다AWS Management Console.
   - IAM 역할에 대한 데이터 리소스에 대한 권한 추가. 이러한 권한이 있는 IAM 역할은 사용자를 대신하여 데이터에 액세스할 수 있습니다.

   사용자, 역할 및 정책을 생성하려면 IAM 관리자여야 합니다.
2. DataBrew에 대한 사용자 또는 그룹 추가. 올바른 권한이 연결된 사용자 또는 그룹은 콘솔에서 DataBrew에 액세스할 수 있습니다.
3. DataBrew에 대한 데이터에 액세스할 수 있는 권한이 있는 역할 추가. 올바른 권한이 있는 역할은 사용자를 대신하여 데이터에 액세스할 수 있습니다.

새AWS계정 설정

AWS계정이 없는 경우AWS계정에 가입하고 IAM 관리자 사용자를 생성합니다.

이 없는 경우 다음 단계를AWS 계정완료하여 생성합니다.

에 가입하려면AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든AWS 서비스및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

다음 옵션 중 하나를 선택하여 관리 사용자를 생성합니다.

관리자를 관리하는 방법 한 가지 선택	목적	By	다른 방법
IAM Identity Center에서 (권장)	단기 보안 인증 정보를 사용하여AWS에 액세스합니다. 이는 보안 모범 사례와 일치합니다. 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.	AWS IAM Identity Center사용 설명서의 시작하기 지침을 따릅니다.	AWS Command Line Interface 사용 설명서에서 사용하도록 AWS CLI를 구성AWS IAM Identity Center하여 프로그래밍 방식 액세스를 구성합니다.
IAM에서 (권장되지 않음)	장기 보안 인증 정보를 사용하여AWS에 액세스합니다.	IAM 사용 설명서의 비상 액세스를 위한 IAM 사용자 생성에 나와 있는 지침을 따르세요.	IAM 사용 설명서에 나온 IAM 사용자의 액세스 키 관리를 수행하여 프로그래밍 방식의 액세스를 구성합니다.

자세한 설명은 IAM 사용자 가이드에서 다음 주제를 참조하세요:

- IAM이란 무엇입니까?
- IAM 설정
- 관리 사용자 및 그룹 생성(콘솔)

설정AWS CLI

JupyterLab 또는 DataBrew API를 사용하려면 ()를AWS Command Line Interface설치해야 합니다 AWS CLI. DataBrew 콘솔을 사용하거나 시작하기 연습의 단계를 수행하는 데 필요하지 않습니다.

를 설정하려면AWS CLI

1. 다음 단계에AWS CLI따라를 다운로드하고 구성합니다.

   - AWS CLI설치
   - 구성 기본 사항

2. 명령 프롬프트에 다음 DataBrew 명령을 입력하여 설정을 확인합니다.

   ```
   aws databrew help
   ```

   이 문이 "aws: error: argument command: Invalid choice" 오류를 반환한 후 서비스 목록이 긴 경우를 제거한AWS CLI다음 다시 설치합니다. 이 작업은 기존 구성을 덮어쓰지 않습니다.

AWS CLI명령은 파라미터 또는 프로파일로 설정하지 않는 한 구성의 기본AWS리전을 사용합니다. 각 명령에 --region 파라미터를 추가할 수 있습니다.

원하는 경우 ~/.aws/config 또는 %UserProfile%/.aws/config (Microsoft Windows)에서 명명된 프로필을 추가할 수 있습니다. 명명된 프로필은 다음 예제와 같이 다른 설정도 보존할 수 있습니다.

```
[profile databrew]
aws_access_key_id = ACCESS-KEY-ID-OF-IAM-USER
aws_secret_access_key = SECRET-ACCESS-KEY-ID-OF-IAM-USER
region = us-east-1
output = text
```

AWS Identity and Access Management(IAM) 권한 설정

시작하기 전에 IAM에서 몇 가지 사항을 설정해야 합니다. 관리자이거나 관리자로부터 도움을 받아야 합니다. 그러나 관리자 액세스 권한이 있는 계정이 있는 경우 이러한 작업을 직접 수행할 수 있습니다. 이 단원에서는 각 작업에 대한 간단한 지침을 확인할 수 있습니다.

다음은 수행해야 할 작업에 대한 개요입니다.

- 이 프로세스의 일부로 사용자를 추가합니다. 새 사용자를 추가할 필요가 없으며 기존 사용자를 사용할 수 있습니다. 사용자가 DataBrew 콘솔을 열 수 있도록 DataBrew 권한을 연결합니다.
- IAM 역할을 생성합니다. 역할은 특정 작업을 허용하고 사용 시 제한 내에서 권한을 부여합니다. 예를 들어 계정의 사용자에게만 작동합니다AWS. 나중에 제한 사항을 더 추가할 수 있습니다.

- 필요한 IAM 정책을 생성합니다. 정책은 사용자가 수행할 수 있는 사물 목록입니다. 정책을 생성하려면 다른 콘솔 페이지를 열고 다운로드한 파일의 텍스트를 붙여 넣습니다.

> **Note**
>
> 여기서 제공하는 내용은 기본 설정 정보입니다. 보안 및 규정 준수 요구 사항을 충족하도록 권한을 사용자 지정하는 것이 좋습니다. 도움이 필요한 경우 관리자 또는AWS Support에 문의하세요.

필요한 권한을 추가하려면

1. 다음을 수행하여 사용자가 DataBrew를 실행할 수 있도록 IAM 정책을 생성합니다.

   - 콘솔 사용자에 대한 사용자 지정 IAM 정책을 추가합니다. 사용자 지정 정책이 필요하지 않은 경우 대신AWS관리형 정책을 선택할 수 있습니다. 2단계에서 사용자에게 추가하기만 하면 됩니다. 이러한 권한이 있는 사용자는 DataBrew 서비스 콘솔에 액세스할 수 있습니다.
   - 데이터 리소스에 대한 권한을 추가합니다. 이러한 권한이 있는 IAM 역할은 사용자를 대신하여 데이터에 액세스할 수 있습니다.

   사용자, 역할 및 정책을 생성하려면 관리자여야 합니다.
2. DataBrew에 대한 사용자 또는 그룹을 추가합니다. 올바른 권한이 연결된 사용자 또는 그룹은 DataBrew 콘솔에 액세스할 수 있습니다.
3. DataBrew에 대한 데이터에 액세스할 수 있는 권한이 있는 역할을 추가합니다. 올바른 권한이 있는 역할은 사용자를 대신하여 데이터에 액세스할 수 있습니다.

DataBrew에 대한 IAM 정책 설정

IAM 정책을 사용하여 권한을 관리합니다. 정책을 사용하면 한 번에 하나씩이 아니라 한 번에 모든 관련 권한을 더 쉽게 추가할 수 있습니다.

제공하는 것과 동일한 이름을 사용하여 정책을 생성하는 것이 좋습니다. 설명서 전체에서 이러한 정책에는 다음과 같은 이름을 사용합니다. 또한 이러한 이름을 사용하면AWS Support에 문의해야 하는 경우 더 쉬워집니다. 그러나 정책 이름과 그 내용을 모두 변경하도록 선택할 수 있습니다. IAM 정책에 대한 자세한 내용은 IAM 사용 설명서의 고객 관리형 정책 생성을 참조하세요.

DataBrew를 사용하는 데 필요한 정책을 생성한 후 사용자 및 역할에 연결합니다. 이를 수행하는 방법은이 섹션의 뒷부분에서 다룹니다.

주제

- 콘솔 사용자에 대한 IAM 정책 추가
- IAM 역할에 대한 데이터 리소스에 대한 권한 추가
- DataBrew에 대한 IAM 정책 구성

콘솔 사용자에 대한 IAM 정책 추가

에 대한 사용자 권한 설정은 선택 사항이지만 콘솔 액세스AWS Management Console가 필요한 경우 먼저이 단계를 수행합니다.

콘솔에서 DataBrew에 도달할 수 있는 권한을 설정하려면 다음 중 하나를 선택합니다.

- 에서 관리하는 정책을 사용합니다AWS`AwsGlueDataBrewFullAccessPolicy`. 이 옵션을 선택하는 경우 다음 정책인 로 건너뜁니다IAM 역할에 대한 데이터 리소스에 대한 권한 추가.
- 이 단원인에 설명된 정책을 생성합니다`AwsGlueDataBrewCustomUserPolicy`. 이 옵션을 사용하면 추가 사용자 지정 보안 요구 사항으로 정책을 사용자 지정할 수 있습니다.

다음 정책은 DataBrew 콘솔을 실행하는 데 필요한 권한을 부여합니다. IAM을 사용하여 이러한 권한을 제공합니다.

DataBrew에 대한 AwsGlueDataBrewCustomUserPolicy IAM 정책을 정의하려면(콘솔)

1. `AwsGlueDataBrewCustomUserPolicy` IAM 정책의 JSON을 다운로드합니다.
2. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
3. 탐색 창에서 Policies를 선택합니다.
4. 각 정책에 대해 정책 생성을 선택합니다.
5. 정책 생성 화면에서 JSON 탭으로 이동합니다.
6. 다운로드한 정책 JSON 문을 복사합니다. 편집기의 샘플 문 위에 붙여 넣습니다.
7. 정책이 계정, 보안 요구 사항 및 필수AWS리소스에 맞게 사용자 지정되었는지 확인합니다. 변경해야 하는 경우 편집기에서 변경할 수 있습니다.
8. 정책 검토를 선택합니다.

DataBrew에 대한 AwsGlueDataBrewCustomUserPolicy IAM 정책을 정의하려면(AWS CLI)

1. AwsGlueDataBrewCustomUserPolicy IAM 정책의 JSON을 다운로드합니다.
2. 이전 절차의 첫 번째 단계에 설명된 대로 정책을 사용자 지정합니다.
3. 다음 명령을 실행하여 정책을 생성합니다.

```
aws iam create-policy --policy-name AwsGlueDataBrewCustomUserPolicy --policy-
document file://iam-policy-AwsGlueDataBrewCustomUserPolicy.json
```

IAM 역할에 대한 데이터 리소스에 대한 권한 추가

데이터에 연결하려면 사용자를 대신하여 전달할 수 있는 IAM 역할AWS Glue DataBrew이 있어야 합니다. 아래에서 나중에 IAM 역할에 연결하는 정책을 생성하는 방법을 확인할 수 있습니다.

이 AwsGlueDataBrewDataResourcePolicy 정책은 DataBrew를 사용하여 데이터에 연결하는 데 필요한 권한을 부여합니다. Amazon S3의 객체에 액세스하는 등 다른AWS리소스의 데이터에 액세스하는 작업의 경우 DataBrew는 사용자를 대신하여 리소스에 액세스할 수 있는 권한이 필요합니다.

DataBrew에 대한 AwsGlueDataBrewDataResourcePolicy IAM 정책을 정의하려면(콘솔)

1. 용 JSON을 다운로드합니다AwsGlueDataBrewDataResourcePolicy.
2. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
3. 탐색 창에서 Policies를 선택합니다.
4. 각 정책에 대해 정책 생성을 선택합니다.
5. 정책 생성 화면에서 JSON 탭으로 이동합니다.
6. 다운로드한 정책 JSON 문을 복사합니다. 편집기의 샘플 문 위에 붙여 넣습니다.
7. 정책이 계정, 보안 요구 사항 및 필수AWS리소스에 맞게 사용자 지정되었는지 확인합니다. 변경해야 하는 경우 편집기에서 변경할 수 있습니다.
8. 정책 검토를 선택합니다.

DataBrew에 대한 AwsGlueDataBrewDataResourcePolicy IAM 정책을 정의하려면(AWS CLI)

1. 용 JSON을 다운로드합니다AwsGlueDataBrewDataResourcePolicy.

2. 이전 절차의 첫 번째 단계에 설명된 대로 정책을 사용자 지정합니다.
3. 다음 명령을 실행하여 정책을 생성합니다.

```
aws iam create-policy --policy-name AwsGlueDataBrewDataResourcePolicy --policy-
document file://iam-policy-AwsGlueDataBrewDataResourcePolicy.json
```

DataBrew에 대한 IAM 정책 구성

아래에서 DataBrew와 함께 사용할 수 있는 IAM 정책에 대한 세부 정보와 예제를 찾을 수 있습니다. 기본 정책에 대한 세부 정보는 여기에 나와 있습니다. 또한 DataBrew를 사용하는 데 필요하지 않은 더 많은 예제가 있습니다. 특정 상황에서 사용할 수 있는 추가 구성입니다.

주제

- AwsGlueDataBrewCustomUserPolicy
- AwsGlueDataBrewDataResourcePolicy
- DataBrew에서 Amazon S3 객체를 사용하기 위한 IAM 정책
- DataBrew에서 암호화를 사용하기 위한 IAM 정책

AwsGlueDataBrewCustomUserPolicy

이 `AwsGlueDataBrewCustomUserPolicy` 정책은 DataBrew 콘솔을 사용하는 데 필요한 대부분의 권한을 부여합니다. 이 정책에 지정된 일부 리소스는 DataBrew에서 사용하는 서비스를 참조합니다. 여기에는의 이름AWS Glue Data Catalog, Amazon S3 버킷, Amazon CloudWatch Logs 및AWS KMS리소스가 포함됩니다. 라는AWS관리형 정책과 유사합니다`AwsGlueDataBrewFullAccessPolicy`.

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
`"databrew:*"`	`"*"`	모든 DataBrew API 작업을 실행할 수 있는 권한을 부여합니다.

`"glue:GetDatabases"`
 	
`"glue:GetPartitions"`
	`"*"`	AWS Glue데이터베이스 및 테이블 목록을 허용합니다.

작업	리소스	설명
"glue:GetTable" "glue:GetTables" "glue:GetDataCatalogEncryptionSettings"		
"dataexchange:ListDataSets" "dataexchange:ListDataSetRevisions" "dataexchange:ListRevisionAssets" "dataexchange:CreateJob" "dataexchange:StartJob" "dataexchange:GetJob"	"*"	데이터 세트의AWS Data Exchange 리소스 목록을 허용합니다.
"kms:DescribeKey" "kms:ListKeys" "kms:ListAliases"	"*"	작업 출력 암호화에 사용할 AWS KMS키 목록을 허용합니다.
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	작업 출력을 암호화할 수 있습니다.

작업	리소스	설명
"s3:ListAllMyBuckets" "s3:GetBucketCORS" "s3:GetBucketLocation" "s3:GetEncryptionConfiguration"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	프로젝트, 데이터 세트 및 작업에 대한 Amazon S3 버킷 목록을 허용합니다. 출력 파일을 S3로 전송할 수 있습니다.
"sts:GetCallerIdentity"	"*"	현재 호출자에 대한 정보를 가져옵니다.
"cloudtrail:LookupEvents",	"*"	데이터 세트(데이터 계보)에 대한AWS CloudTrail이벤트 나열을 허용합니다.
"iam:ListRoles" "iam:GetRole"	"*"	프로젝트 및 작업에 사용할 IAM 역할을 나열할 수 있습니다.

AwsGlueDataBrewDataResourcePolicy

이 AwsGlueDataBrewDataResourcePolicy 정책은 데이터에 연결하고 DataBrew를 구성하는 데 필요한 권한을 부여합니다.

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	파일을 미리 볼 수 있습니다.
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*",	출력 파일을 S3로 전송할 수 있습니다.

작업	리소스	설명
	"arn:aws:s3:::bucket_name"	
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	DataBrew에서 생성한 객체를 삭제할 수 있습니다.
"s3:ListBucket"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	프로젝트, 데이터 세트 및 작업의 Amazon S3 버킷 목록을 허용합니다.
"kms:Decrypt"	"arn:aws:kms:::key/key_ids"	암호화된 데이터 세트에 대한 복호화를 허용합니다.
"kms:GenerateDataKey"	"arn:aws:kms:::key/key_ids"	작업 출력을 암호화할 수 있습니다.

작업	리소스	설명
"ec2:DescribeVpcEndpoints" "ec2:DescribeRouteTables" "ec2:DeleteNetworkInterface" "ec2:DescribeNetworkInterfaces" "ec2:DescribeSecurityGroups" "ec2:DescribeSubnets" "ec2:DescribeVpcAttribute" "ec2:CreateNetworkInterface"	"*"	작업 및 프로젝트를 실행할 때 Virtual Private Cloud(VPCs)와 같은 Amazon EC2 네트워크 항목을 설정할 수 있습니다.
"ec2:DeleteNetworkInterface"	"*"	VPC에서 네트워크 인터페이스를 삭제할 수 있습니다.

작업	리소스	설명

`"ec2:CreateTags"`
 	
`"ec2:DeleteTags"`
	`"arn:aws:ec2:::network-interface/*", "arn:aws:ec2:::security-group/*"`	태그를 생성하고 삭제할 수 있습니다.
		
		VPC가 활성화된AWS Glue Data Catalog를 사용하는 경우 이러한 권한이 필요합니다. DataBrew는에 데이터를 전달AWS Glue하여 작업과 프로젝트를 실행합니다. 이러한 권한을 통해 개발 엔드포인트용으로 생성된 Amazon EC2 리소스에 태그를 지정할 수 있습니다.를 사용하여 Amazon EC2 네트워크 인터페이스, 보안 그룹 및 인스턴스에AWS Glue태그를 지정합니다`aws-glue-service-resource` .

`"logs:CreateLogGroup"`
 	
`"logs:CreateLogStream"`
 	
`"logs:PutLogEvents"`
	`"arn:aws:logs:::log-group:/aws-glue-databrew/*"`	Amazon CloudWatch Logs에 로그 쓰기 허용
		
		DataBrew는 이름이 로 시작하는 로그 그룹에 로그를 씁니다`aws-glue-databrew` .

`"lakeformation:GetDataAccess"`	`"*"`	에 대한 액세스 허용 AWS Lake Formation, `"Glue":"GetTable"` 도 허용됨 Lake Formation을 사용하려면 Lake Formation 콘솔에서 추가 구성이 필요합니다.

DataBrew에서 Amazon S3 객체를 사용하기 위한 IAM 정책

이 AwsGlueDataBrewSpecificS3BucketPolicy 정책은 관리자가 아닌 사용자를 대신하여 S3에 액세스하는 데 필요한 권한을 부여합니다.

다음과 같이 정책을 사용자 지정합니다.

1. 사용하려는 경로를 가리키도록 정책의 Amazon S3 경로를 바꿉니다. 샘플 텍스트에서 *BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*는 특정 객체 또는 파일을 나타냅니다.는 경로 이름이 로 시작하는 모든 객체(*)를 *BUCKET-NAME-2/* 나타냅니다BUCKET-NAME-2/. 사용 중인 버킷의 이름을 지정하도록 업데이트합니다.
2. (선택 사항) Amazon S3 경로에서 와일드카드를 사용하여 권한을 추가로 제한합니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그 섹션을 참조하십시오.

   보안 모범 사례: 다른AWS계정의 이름이 유사한 Amazon S3 버킷에 대한 무단 액세스를 방지하려면 정책에 aws:ResourceAccount 조건 키를 포함하세요. 이렇게 하면 와일드카드 리소스 ARN을 사용하는 경우에도 DataBrew가 자체AWS계정 내의 버킷에만 액세스할 수 있습니다. ARNs 정책 문에 다음 조건을 추가합니다.

   ```
   "Condition": {
   "StringEquals": {
   "aws:ResourceAccount": "123456789012"
   }
   }
   ```

   를 실제AWS계정 ID123456789012로 바꿉니다.

   이를 위해 작업 s3:PutObject 및에 대한 권한을 제한할 수 있습니다s3:PutBucketCORS. 이러한 작업은 DataBrew 프로젝트를 생성하는 사용자에게만 필요합니다. 해당 사용자는 S3로 출력 파일을 전송할 수 있어야 하기 때문입니다.

자세한 내용과 Amazon S3의 IAM 정책에 추가할 수 있는 항목의 몇 가지 예를 보려면 Amazon S3 개발자 안내서의 버킷 정책 예제를 참조하세요. Amazon S3

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
"s3:GetObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	파일을 미리 볼 수 있습니다.
"s3:PutObject" "s3:PutBucketCORS"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	출력 파일을 S3로 전송할 수 있습니다.
"s3:DeleteObject"	"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"	객체 삭제를 허용합니다.

DataBrew에 대한 AwsGlueDataBrewSpecificS3BucketPolicy IAM 정책을 정의하려면(콘솔)

1. AwsGlueDataBrewSpecificS3BucketPolicy IAM 정책의 JSON을 다운로드합니다.
2. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
3. 탐색 창에서 Policies를 선택합니다.
4. 각 정책에 대해 정책 생성을 선택합니다.
5. 정책 생성 화면에서 JSON 탭으로 이동합니다.
6. 정책 JSON 문을 편집기의 샘플 문 위에 붙여 넣습니다.
7. 정책이 계정, 보안 요구 사항 및 필요한AWS리소스에 맞게 사용자 지정되었는지 확인합니다. 변경해야 하는 경우 편집기에서 변경할 수 있습니다.
8. 정책 검토를 선택합니다.

DataBrew에 대한 AwsGlueDataBrewSpecificS3BucketPolicy IAM 정책을 정의하려면(AWS CLI)

1. 용 JSON을 다운로드합니다AwsGlueDataBrewSpecificS3BucketPolicy.

2. 이전 절차의 첫 번째 단계에 설명된 대로 정책을 사용자 지정합니다.
3. 다음 명령을 실행하여 정책을 생성합니다.

```
aws iam create-policy --policy-name AwsGlueDataBrewSpecificS3BucketPolicy --policy-
document file://iam-policy-AwsGlueDataBrewSpecificS3BucketPolicy.json
```

DataBrew에서 암호화를 사용하기 위한 IAM 정책

이 AwsGlueDataBrewS3EncryptedPolicy 정책은 관리자가 아닌 사용자를 대신하여AWS Key Management Service(AWS KMS)로 암호화된 S3 객체에 액세스하는 데 필요한 권한을 부여합니다.

다음과 같이 정책을 사용자 지정합니다.

1. 사용하려는 경로를 가리키도록 정책의 Amazon S3 경로를 바꿉니다. 샘플 텍스트에서 *BUCKET-NAME-1/SPECIFIC-OBJECT-NAME*는 특정 객체 또는 파일을 나타냅니다.는 경로 이름이 로 시작하는 모든 객체(*)를 *BUCKET-NAME-2/* 나타냅니다BUCKET-NAME-2/. 사용 중인 버킷의 이름을 지정하도록 업데이트합니다.
2. (선택 사항) Amazon S3 경로에서 와일드카드를 사용하여 권한을 추가로 제한합니다. 자세한 내용은 IAM 정책 요소: 변수 및 태그 단원을 참조하세요.

   이를 위해 작업 s3:PutObject 및에 대한 권한을 제한할 수 있습니다s3:PutBucketCORS. 이러한 작업은 DataBrew 프로젝트를 생성하는 사용자에게만 필요합니다. 해당 사용자는 S3로 출력 파일을 전송할 수 있어야 하기 때문입니다.

   자세한 내용과 Amazon S3의 IAM 정책에 추가할 수 있는 항목의 몇 가지 예를 보려면 버킷 정책 예제를 참조하세요.
3. ToUseKms 파일에서 다음 리소스 ARNs을 찾습니다.

```
  "arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS",
"arn:aws:kms:AWS-REGION-NAME:AWS-ACCOUNT-ID-WITHOUT-DASHES:key/KEY-IDS"
```

4. 예제AWS계정을AWS계정 번호(하이픈 제외)로 변경합니다.
5. 대신 사용하려는 IAM 역할을 나열하도록 샘플 목록을 변경합니다. IAM 정책을 가능한 가장 작은 권한 세트로 조정하는 것이 좋습니다. 그러나 예를 들어 샘플 데이터와 함께 개인 학습 계정을 사용하는 경우 사용자가 모든 IAM 역할에 액세스하도록 허용할 수 있습니다. 목록이 모든 IAM 역할에 액세스하도록 허용하려면 샘플 목록을 하나의 항목인 로 변경합니다"arn:aws:iam::*111122223333*:role/*".

다음 테이블은 이 정책이 보장하는 권한을 설명합니다.

작업	리소스	설명
`"s3:GetObject"`	`"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"`	파일을 미리 볼 수 있습니다.
`"s3:ListBucket"`	`"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"`	프로젝트, 데이터 세트 및 작업의 Amazon S3 버킷 목록을 허용합니다.
`"s3:PutObject"`	`"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"`	출력 파일을 S3로 전송할 수 있습니다.
`"s3:DeleteObject"`	`"arn:aws:s3:::bucket_name/*", "arn:aws:s3:::bucket_name"`	DataBrew에서 생성한 객체를 삭제할 수 있습니다.
`"kms:Decrypt"`	`"arn:aws:kms:::key/key_ids"`	암호화된 데이터 세트에 대한 복호화를 허용합니다.
`"kms:GenerateDataKey*"`	`"arn:aws:kms:::key/key_ids"`	작업 출력을 암호화할 수 있습니다.

DataBrew에 대한 AwsGlueDataBrewS3EncryptedPolicy IAM 정책을 정의하려면(콘솔)

1. AwsGlueDataBrewS3EncryptedPolicy IAM 정책의 JSON을 다운로드합니다.
2. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/iam/ IAM 콘솔을 엽니다.
3. 탐색 창에서 Policies를 선택합니다.

4. 각 정책에 대해 정책 생성을 선택합니다.
5. 정책 생성 화면에서 JSON 탭으로 이동합니다.
6. 정책 JSON 문을 편집기의 샘플 문 위에 붙여 넣습니다.
7. 정책이 계정, 보안 요구 사항 및 필요한AWS리소스에 맞게 사용자 지정되었는지 확인합니다. 변경해야 하는 경우 편집기에서 변경할 수 있습니다.
8. 정책 검토를 선택합니다.

DataBrew에 대한 AwsGlueDataBrewS3EncryptedPolicy IAM 정책을 정의하려면(AWS CLI)

1. 용 JSON을 다운로드합니다AwsGlueDataBrewS3EncryptedPolicy.
2. 이전 절차의 첫 번째 단계에 설명된 대로 정책을 사용자 지정합니다.
3. 다음 명령을 실행하여 정책을 생성합니다.

```
aws iam create-policy --policy-name AwsGlueDataBrewS3EncryptedPolicy --policy-
document file://iam-policy-AwsGlueDataBrewS3EncryptedPolicy.json
```

DataBrew 권한이 있는 사용자 또는 그룹 추가

권한을 관리하기 위해 역할에 정책을 할당하고 사용자 및 그룹에 역할을 할당합니다. 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명(사용자, 그룹 및 역할)을 참조하세요.

시작하기 전에 권한을 할당할 사용자가 한 명 이상 있어야 합니다.

다음 절차에 따라 DataBrew 콘솔에서 작업하거나 CLI에서 DataBrew 명령을 실행해야 하는 사용자에 대해 DataBrew 권한을 설정합니다.

DataBrew 권한을 설정하려면

1. 사용자가 DataBrew AWS CLI용 및 기타 개발 도구를 사용할 수 있는 액세스 키를 생성합니다.
2. 사용자가AWS콘솔을 사용할 수 있도록 AWS Management Console액세스를 활성화합니다.
3. DataBrew 사용자 또는 그룹에 대한 역할을 생성합니다.
4. 사용 중인 정책을 선택합니다. 다음 중 하나를 수행하세요.
   - 를 생성한 경우 `AwsGlueDataBrewCustomUserPolicy`목록에서 선택합니다.

- AWS 관리형 정책을 사용하려면 목록에서 `AwsGlueDataBrewFullAccessPolicy`를 선택합니다.

5. 역할에 해당 정책을 할당합니다.
6. 사용자 또는 그룹이 관련 역할을 수임할 수 있도록 역할에 대한 신뢰 관계를 설정합니다.

   - 그룹을 사용하지 않는 경우 사용자를 역할로 신뢰합니다.
   - 그룹을 사용하는 경우 역할과 함께 그룹을 신뢰하고 사용자를 그룹에 추가합니다.

데이터 리소스 권한이 있는 IAM 역할 추가

IAM 역할을 사용하여 함께 할당된 정책을 관리합니다. IAM 역할은 DataBrew 사용자 또는 DataBrew 자체와 같은 특정 역할을 수행하는 사람이 사용할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 역할을 참조하세요.

다음 절차에 따라 DataBrew 프로젝트가 데이터에 액세스하는 데 필요한 IAM 역할을 생성합니다.

DataBrew의 새 IAM 역할에 필요한 IAM 정책을 연결하려면

1. 탐색 창에서 역할, 역할 생성을 선택합니다.
2. 신뢰할 수 있는 엔터티 유형 선택에서 카드 레이블이 지정된 AWS서비스를 선택합니다.
3. 목록에서 DataBrew를 선택한 후 다음: 권한을 선택합니다.
4. 검색 상자(이전 단계에서 생성한 IAM 정책)에 **`AwsGlueDataBrewDataResourcePolicy`**를 입력합니다. 정책을 선택하고 다음: 태그를 선택합니다.
5. 다음: 검토를 선택합니다.
6. 역할 이름에 **`AwsGlueDataBrewDataAccessRole`**을(를) 입력한 다음 역할 생성을 선택합니다.

설정AWS IAM Identity Center(IAM Identity Center)

AWS IAM Identity Center(IAM Identity Center)를 사용하면 사용자는AWS계정에 로그인하지 않고도 간단한 URL로 DataBrew에 로그인할 수AWS Management Console있습니다.

IAM Identity Center를 설정하려면

1. AWS Organizations콘솔을 열고 아직 없는 경우 조직을 생성합니다. 이 조직에서는 모든 기능이 기본적으로 활성화됩니다.

자세한 내용은 AWS IAM Identity Center사전 조건 및 조직 생성 및 관리를 참조하세요.

2. AWS IAM Identity Center콘솔을 엽니다
3. ID 소스를 선택합니다.

   빠르고 쉬운 사용자 관리를 위한 IAM Identity Center 스토어가 기본적으로 제공됩니다. 선택적으로 외부 자격 증명 공급자를 대신 연결하거나AWS Managed Microsoft AD디렉터리를 온프레미스 Active Directory에 연결할 수 있습니다. 이 가이드에서는 기본 IAM Identity Center 스토어를 사용합니다.

   자세한 내용은 AWS IAM Identity Center사용 설명서의 자격 증명 소스 선택을 참조하세요.
4. DataBrew 액세스를 위한 권한 세트를 생성합니다.

   a. IAM Identity Center 탐색 창에서 AWS계정을 선택한 다음 권한 세트를 선택합니다.
   b. 권한 세트 생성 페이지에서 사용자 지정 권한 세트 생성을 선택합니다.
   c. 릴레이 상태에를 입력합니다`https://console.aws.amazon.com/databrew/home?region=us-east-1#landing`.

      이를 입력하면 사용자가 DataBrew로 직접 이동할 수 있습니다.
   d. AWS관리형 정책 연결을 선택하고 DataBrew를 검색한 다음 AwsGlueDataBrewFullAccessPolicy를 선택합니다. 이를 선택하면 사용자에게 DataBrew에 필요한 모든 권한이 부여됩니다. 자세한 내용은에서 확인할 수 있습니다콘솔 사용자에 대한 IAM 정책 추가.
   e. (선택 사항) 사용자 지정 권한 정책 생성을 선택하고 사용자에 대한 권한을 사용자 지정합니다.
5. IAM Identity Center 탐색 창에서 그룹을 선택하고 그룹 생성을 선택합니다. 그룹 이름을 입력하고 생성을 선택합니다.
6. IAM Identity Center 스토어에 사용자를 추가합니다.

   a. IAM Identity Center 탐색 창에서 사용자를 선택합니다.
   b. 사용자 추가 화면에서 필수 정보를 입력하고 사용자에게 암호 설정 지침이 포함된 이메일 전송을 선택합니다. 사용자에게 다음 설정 단계에 대한 이메일이 전송됩니다.
   c. 다음: 그룹을 선택하고 사용자를 추가할 그룹을 선택한 다음, 사용자 추가를 선택합니다.

      사용자는 SSO를 사용하도록 초대하는 이메일을 받게 됩니다. 이 이메일에서는 초대 수락을 선택하고 암호를 설정해야 합니다. 또한 이메일에서 포털 URL을 찾을 수 있습니다. 이 URL을 사용하여 DataBrew에 액세스할 수 있습니다.

7. 계정에 각 사용자를 할당합니다.

   a. IAM Identity Center 콘솔을 열고 탐색 창에서 AWS계정을 선택합니다.

   b. AWS조직을 선택하고AWS계정을 선택합니다.

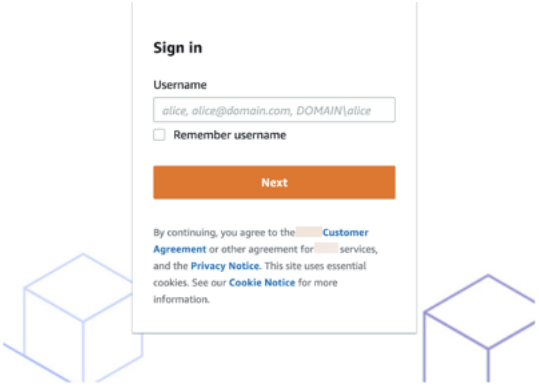
   c. 사용자 할당 화면에서 그룹 탭을 선택하고 원하는 그룹을 선택합니다.

   d. Next: Permission sets(다음: 권한 세트)를 선택합니다.

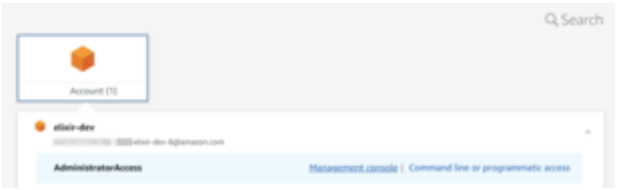
   e. DataBrew에 대한 권한 세트를 선택하고 완료를 선택합니다.

IAM Identity Center 지원 사용자의 로그인 단계

1. IAM Identity Center 지원 계정을AWS사용하여에 로그인합니다.



2. AWS계정 자격 증명을 클릭합니다.



3. DataBrew 콘솔로 원클릭 리디렉션하려면 관리 콘솔을 클릭합니다.

JupyterLab에서 DataBrew를 확장으로 사용

> **Warning**
>
> AWS Glue DataBrew JupyterLab 3는 지원이 종료되므로 JupyterLab 확장 지원은 2024년 12월 31일에 종료됩니다. 자세한 내용은 JupyterLab 3 유지 관리 종료를 참조하세요.

Jupyter Notebook 환경에서 데이터를 준비하려면 JupyterLab AWS Glue DataBrew에서의 모든 기능을 사용할 수 있습니다.

JupyterLab은 Jupyter Notebook을 위한 웹 기반 대화형 개발 환경입니다. 로컬 JupyterLab 웹 페이지에서 터미널, SQL 세션, Python 등에 대한 섹션을 추가할 수 있습니다.AWS Glue DataBrew확장을 설치한 후 DataBrew 콘솔에 대한 섹션을 추가할 수 있습니다. JupyterLab 환경에서 직접 이미 보유한 기존 노트북 또는 기타 확장으로 실행됩니다.

주제

사전 조건

시작하기 전에 다음 항목을 설정합니다.

- AWS계정 - 아직 계정이 없는 경우 로 시작합니다새AWS계정 설정.
- DataBrew에 필요한 권한에 액세스할 수 있는AWS Identity and Access Management(IAM) 사용자 - 자세한 내용은 섹션을 참조하세요DataBrew 권한이 있는 사용자 또는 그룹 추가.
- DataBrew 작업에 사용할 IAM 역할 -가 구성된 경우 기본값을 사용할 수 `AwsGlueDataBrewDataAccessRole` 있습니다. 추가 IAM 역할을 설정하려면 섹션을 참조하세요데이터 리소스 권한이 있는 IAM 역할 추가.
- JupyterLab 설치(버전 2.2.6 이상) - 자세한 내용은 JupyterLab 설명서의 다음 주제를 참조하세요.
  - JupyterLab 사전 조건
  - JupyterLab 설치 -를 사용하는 것이 좋습니다`pip install jupyterlab`.
- Node.js 설치(버전 12.0 이상).
- AWS Command Line Interface(AWS CLI) 설치 - 자세한 내용은 섹션을 참조하세요설정AWS CLI.
- AWS Jupyter 프록시 설치(`pip install aws-jupyter-proxy`) -이 확장은AWS서비스 엔드포인트와 함께 사용되어 자격 증명을 안전하게 전달합니다AWS. 자세한 내용은 GitHub의 aws-jupyter-proxy를 참조하세요.

사전 조건이 설치되어 있는지 확인하려면 다음 예제와 같이 명령줄에서 다음과 유사한 테스트를 실행할 수 있습니다.

```
echo "
AWS CLI:"
which aws
aws --version
aws configure list
aws sts get-caller-identity

echo "
Python (current environment):"
which python
python --version

echo "
Node.JS:"
which node
node --version

echo "
Jupyter:"
where jupyter
jupyter --version
jupyter serverextension list
pip3 freeze | grep jupyter
```

출력은 다음과 같아야 합니다. 디렉터리는 운영 체제 및 구성에 따라 다릅니다.

```
AWS CLI:
/usr/local/bin/aws
aws-cli/2.1.2 Python/3.7.4 Darwin/19.6.0 exe/x86_64
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************VXW4 shared-credentials-file
secret_key     ****************MRJN shared-credentials-file
    region                us-east-1      config-file    ~/.aws/config
{
    "UserId": "",
    "Account": "111122223333",
    "Arn": "arn:aws:iam::111122223333:user/user2"
}

Python (current environment):
/usr/local/opt/python /libexec/bin/python
```

```
Python 3.8.5

Node.JS:
/usr/local/bin/node
v15.0.1

Jupyter:
/usr/local/bin/jupyter
jupyter core     : 4.6.3
jupyter-notebook : 6.0.3
qtconsole        : 4.7.5
ipython          : 7.16.1
ipykernel        : 5.3.2
jupyter client   : 6.1.6
jupyter lab      : 2.2.9
nbconvert        : 5.6.1
ipywidgets       : 7.5.1
nbformat         : 5.0.7
traitlets        : 4.3.3

config dir: /usr/local/etc/jupyter
    aws_jupyter_proxy  enabled
    - Validating...
      aws_jupyter_proxy  OK
    jupyterlab  enabled
    - Validating...
      jupyterlab 2.2.9 OK

aws-jupyter-proxy==0.1.0
jupyter-client==6.1.7
jupyter-core==4.7.0
jupyterlab==2.2.9
jupyterlab-pygments==0.1.2
jupyterlab-server==1.2.0
```

확장을 사용하도록 JupyterLab 구성

JupyterLab을 설치한 후에는 데이터 액세스를 보호하고 서버 확장을 활성화하도록 구성해야 합니다.

암호 및 암호화를 구성하려면

1. 암호를 설정하여 확장에 추가하려는 데이터를 보호합니다. Jupyter는 암호 유틸리티를 제공합니다. 다음 명령을 실행하고 프롬프트에 원하는 암호를 입력하십시오.

```
jupyter notebook password
```

출력은 다음과 같습니다.

```
Enter password:
Verify password:
[NotebookPasswordApp] Wrote hashed password to /home/ubuntu/.jupyter/
jupyter_notebook_config.json
```

2. Jupyter 서버에서 암호화를 활성화합니다. 로컬 시스템에 Jupyter를 설치하고 네트워크를 통해 액세스할 수 있는 사람이 없는 경우이 단계를 건너뛸 수 있습니다.

   TLS(전송 계층 보안)를 사용하여 암호화를 설정하려면 환경에 맞게 사용자 지정된 인증서를 생성합니다. 자세한 내용은 Jupyter 설명서의 서버 보안에서 Let's Encrypt 사용을 참조하세요.

3. JupyterLab을 시작하려면 명령 프롬프트에서 다음 명령을 실행합니다.

   ```
   jupyter lab
   ```

   자세한 내용은 JupyterLab 설명서의 JupyterLab 시작을 참조하세요. JupyterLab

4. JupyterLab이 실행되는 동안와 유사한 URL에서 액세스할 수 있습니다http://localhost:8888/lab. 암호화를 설정하는 경우 `https` 대신를 사용합니다`http`. 포트를 사용자 지정한 경우 대신 포트 번호를 대체합니다`8888`.

다음 절차에 따라 타사 확장을 활성화합니다.

JupyterLab에서 타사 확장을 활성화하려면

1. JupyterLab 웹 페이지의 왼쪽 메뉴에서 확장 관리자 아이콘을 선택합니다.
2. 타사 확장 프로그램 실행의 위험에 대한 경고를 읽습니다. 신뢰하는 개발자의 확장만 설치합니다.
3. JupyterLab에서 타사 확장을 활성화하려면 활성화를 선택합니다.
4. 프롬프트에 따라 JupyterLab을 다시 빌드하고 다시 로드합니다.

JupyterLab용 DataBrew 확장 활성화

확장이 활성화된 JupyterLab을 안전하게 설치한 후 노트북에서 DataBrew를 실행할 수 있도록 DataBrew 확장을 설치합니다.

DataBrew용 확장 프로그램을 설치하려면(콘솔)

1. JupyterLab을 시작하려면 명령 프롬프트에서 다음 명령을 실행합니다.

```
jupyter lab
```

2. JupyterLab 웹 페이지의 왼쪽 메뉴에서 확장 관리자 아이콘을 선택합니다.
3. 왼쪽 상단의 검색에 "**brew**"를 입력하여 DataBrew 확장을 검색합니다.
4. 목록에서 aws_glue_databrew_jupyter를 찾되 클릭하지 마세요. 강조 표시된 확장 이름을 클릭하면 GitHub의 aws_glue_databrew_jupyter 페이지와 함께 새 브라우저 창이 열립니다.
5. DataBrew 확장을 설치하려면 다음 중 하나를 선택합니다.

   - 명령줄에서를 실행합니다`jupyter labextension install aws_glue_databrew_jupyter`.
   - 회색 문자의 "aws_glue_databrew_jupyter" 아래에 있는 확장 카드 하단에서 설치를 선택합니다.

   DataBrew 확장은 JupyterLab 버전 1.2 및 2.x와 호환됩니다.
6. 설치되었는지 확인하려면를 실행합니다`jupyter labextension list`. 출력은 다음과 같아야 합니다.

```
JupyterLab v2.2.9
Known labextensions:
   app dir: /usr/local/share/jupyter/lab  # varies by OS
        aws_glue_databrew_jupyter v1.0.1  enabled  OK
```

7. 다음 중 하나를 사용하여 JupyterLab을 다시 빌드합니다.

   - 명령 프롬프트에서를 실행합니다`jupyter lab build`.
   - 웹 페이지에서 왼쪽 상단의 재구축을 선택합니다.
8. 빌드가 완료되면 다음 중 하나를 수행합니다.

   - 명령 프롬프트에서를 실행합니다`jupyter lab`.
   - 웹 페이지의 빌드 완료 메시지에서 다시 로드를 선택합니다.
9. JupyterLab 웹 페이지에서 왼쪽 메뉴에서 아이콘을 선택하여 Extension Manager를 닫습니다.

   확장을 열려면 시작AWS Glue DataBrew 관리자 탭의 기타 섹션에서 시작을 선택합니다. 확장은 액세스 키 및AWS리전 설정에 현재AWS CLI구성을 사용합니다.

설정을 완료한 후 AWS Glue DataBrew 탭을 사용하여 JupyterLab 내에서 DataBrew와 상호 작용할 수 있습니다.

시작하기AWS Glue DataBrew

다음 자습서를 사용하여 첫 번째 DataBrew 프로젝트를 생성하는 방법을 안내할 수 있습니다. 샘플 데이터 세트를 로드하고, 해당 데이터 세트에서 변환을 실행하고, 이러한 변환을 캡처하는 레시피를 빌드하고, 변환된 데이터를 Amazon S3에 쓰는 작업을 실행합니다.

주제

사전 조건

계속하기 전에의 해당 지침을 따르세요설AWS Glue DataBrew정. 그런 다음 로 계속 진행합니다1단계: 프로젝트 생성.

1단계: 프로젝트 생성

이 단계에서는 DataBrew 콘솔을 사용하여 샘플 프로젝트를 빠르게 시작합니다.

프로젝트를 생성하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 엽니다.
2. DataBrew 콘솔에서AWS리전이 오른쪽 상단에 선택되어 있는지 확인합니다. DataBrew에서 지원하는AWS리전 목록은의 DataBrew 엔드포인트 및 할당량을 참조하세요AWS 일반 참조.
3. 탐색 창에서 프로젝트를 선택한 다음 프로젝트 생성을 선택합니다.
4. 프로젝트 세부 정보 창에서 다음을 수행합니다.

- 프로젝트 이름에를 입력합니다chess-project.
- 연결된 레시피에서 새 레시피를 생성합니다. 레시피에 대해 제안된 이름이 제공됩니다(chess-project-recipe).

5. 데이터 세트 선택 창에서 샘플 파일을 선택합니다.
6. 샘플 파일 창에서 유명한 체스 게임 이동을 선택합니다. 이 데이터 세트에는 20,000개 이상의 체스 게임에 대한 자세한 정보가 포함되어 있습니다.

   데이터 세트 이름에는 데이터 세트의 제안된 이름이 제공됩니다(chess-games).
7. 액세스 권한 창에서를 선택합니다AwsGlueDataBrewDataAccessRole. 이는 DataBrew가 사용자를 대신하여 Amazon S3 버킷에 액세스할 수 있도록 하는 서비스 연결 역할입니다.
8. 프로젝트 생성을 선택하고 DataBrew가 프로젝트 준비를 완료할 때까지 기다립니다. 창은 다음과 비슷합니다.

   표시되는 데이터는 chess-games 데이터 세트의 샘플을 나타냅니다. 기본적으로 샘플은 데이터 세트의 처음 500개 행으로 구성됩니다. 나중에이 프로젝트 설정을 변경할 수 있습니다.

   도구 모음은 데이터에 적용할 수 있는 수백 개의 데이터 변환에 대한 액세스를 제공합니다.

   DataBrew 콘솔의 오른쪽에 있는 레시피 창은 지금까지 적용한 변환을 추적합니다.

2단계: 데이터 요약

이 단계에서는이 데이터세트 및 이와 유사한 다른 데이터세트에 적용할 수 있는 변환 세트인 DataBrew 레시피를 빌드합니다. 레시피가 완료되면 사용할 수 있도록 게시합니다.

체스 게임에서 플레이어는 다른 플레이어와 비교하여 얼마나 잘 작동하는지에 따라 평가될 수 있습니다. (자세한 내용은 https://en.wikipedia.org/wiki/Chess_rating_system 단원을 참조하십시오.) 이 자습서에서는 두 플레이어가 모두 클래스 A인 게임에만 초점을 맞춥니다. 즉, 평점이 1800 이상이었습니다.

데이터를 요약하려면

1. 변환 도구 모음에서 필터, 조건별, 이상 또는 같음을 선택합니다.
2. 다음과 같이 이러한 옵션을 설정합니다.

   - 소스 열 - white_rating
   - 필터 조건 - 1800 이상

변환의 작동 방식을 확인하려면 변경 사항 미리 보기를 선택합니다. 그런 다음, 적용을 선택합니다.

3. 이전 단계를 반복하지만 이번에는 소스 열을 로 설정합니다`black_rating`. 변경 사항을 적용한 후 샘플 데이터에는 각 측의 플레이어(흑백)가 클래스 A 이상인 게임만 포함됩니다.
4. 데이터를 요약하여 각 측에서 받은 게임 수를 결정합니다. 이렇게 하려면 변환 도구 모음에서 그룹을 선택합니다.
5. 그룹 속성의 경우 다음을 수행합니다.

   a. 첫 번째 행에서 열 이름으로 `winner`를 선택합니다. 집계를 그룹화 기준으로 설정된 상태로 둡니다.

   b. 두 번째 행에서 열 이름으로 `victory_status`를 선택합니다. 집계를 그룹화 기준으로 설정된 상태로 둡니다.

   c. 다른 열 추가를 선택합니다.

   d. 세 번째 행에서 열 이름으로 `winner`를 선택합니다. 집계를 개수로 설정합니다.

   e. 그룹 유형에서 새 테이블로 그룹을 선택합니다. 미리 보기 창에는 결과가 어떻게 보일지 표시됩니다.

   f. 마침을 클릭합니다.
6. 게시를 선택하여 레시피 창의 오른쪽에 있는 작업을 저장합니다.
7. 버전 설명에 레시피의 첫 번째 버전을 입력합니다. 그런 다음 게시를 선택합니다.

3단계: 변환 추가

이 단계에서는 레시피에 변환을 더 추가하고 다른 버전의 변환을 게시합니다. 이 예제를 구체화하기 위해 모든 체스 게임이 명확한 승자가 되는 것은 아니라는 정보를 사용합니다. 일부 게임은 드로우로 재생됩니다.

레시피 변환을 추가하고 다시 게시하려면

1. 변환 도구 모음에서 Filter, By Condition, Is not를 선택하여 그리기 위해 재생된 게임을 제거합니다.
2. 다음과 같이 이러한 옵션을 설정합니다.

   - 소스 열 - `victory_status`
   - 필터 조건 -이 아님 `draw`

레시피에이 변환을 추가하려면 적용을 선택합니다.

3. 더 의미victory_status가 있도록에서 데이터를 변경합니다. 이렇게 하려면 변환 도구 모음에서 값 또는 패턴 정리, 바꾸기, 바꾸기를 선택합니다.
4. 다음과 같이 이러한 옵션을 설정합니다.

   - 소스 열 - victory_status
   - 바꿀 값 지정 - 값 또는 패턴
   - 대체할 값 - mate
   - 값으로 바꾸기 - checkmate

   레시피에이 변환을 추가하려면 적용을 선택합니다.
5. 이전 단계를 반복하되를 resign로 변경합니다other player resigned.
6. 이전 단계를 반복하되를 outoftime로 변경합니다time ran out.
7. 게시를 선택하여 레시피 창의 오른쪽에 있는 작업을 저장합니다.

4단계: DataBrew 리소스 검토

이제 샘플 프로젝트로 작업했으므로 지금까지 생성한 DataBrew 리소스를 검토합니다.

DataBrew 리소스를 검토하려면

1. 탐색 창에서 데이터 세트를 선택합니다.

   샘플 프로젝트를 생성할 때 DataBrew에서 데이터 세트를 생성했습니다(chess-games). 소스 데이터 파일은 Amazon S3에 저장되며 Microsoft Excel 형식(chess-games.xlsx)입니다. 파일에는 20,000개가 넘는 체스 게임의 메타데이터가 포함되어 있습니다. chess-games 데이터 세트는 DataBrew가 해당 파일의 데이터를 읽는 데 필요한 정보를 제공합니다.
2. 탐색 창에서 프로젝트를 선택합니다.

   이전 단계()에서 작업한 프로젝트가 표시됩니다chess-project. 이 경우 모든 프로젝트에는 데이터 세트가 필요합니다chess-games. 또한 모든 프로젝트에는 레시피가 필요하므로 진행하면서 데이터 변환 단계를 추가할 수 있습니다. 이 샘플 프로젝트를 생성할 때 DataBrew는 새 (비어 있는) 레시피를 생성하여 프로젝트에 연결했습니다.

3. 탐색 창에서 레시피를 선택하고 레시피 이름 열에서 chess-project-recipe를 선택합니다. 이는 DataBrew가 프로젝트에 대해 생성한 레시피와 변환 단계를 추가하여 개선한 레시피를 보여줍니다.
4. 왼쪽에서 게시된 레시피 버전을 확인합니다. 다음 중 하나를 선택하면 해당 버전의 레시피 세부 정보와 단계를 보여주는 레시피 단계 탭을 볼 수 있습니다.
5. 데이터 출처와 사용 방법을 보여주는 데이터 계보 탭을 봅니다. 자세한 내용은 다이어그램에서 아이콘을 선택합니다.

5단계: 데이터 프로필 생성

프로젝트에서 로 작업하면 DataBrew는 샘플의 행 수 및 각 열의 고유 값 분포와 같은 통계를 표시합니다. 이러한 통계 등은 샘플의 프로파일을 나타냅니다.

데이터 프로파일을 요청하려면 프로파일 작업을 생성하고 실행합니다.

데이터 세트를 프로파일링하려면

1. 탐색 창에서 작업을 선택합니다.
2. 프로필 작업 탭에서 작업 생성을 선택합니다.
3. 작업 이름에를 입력합니다`chess-data-profile`.
4. 작업 유형에서 프로필 작업 생성을 선택합니다.
5. 작업 입력 창에서 다음을 수행합니다.
   - 실행 시 데이터 세트를 선택합니다.
   - 사용 가능한 데이터 세트 목록을 보려면 데이터 세트 선택을 선택하고를 선택합니다`chess-games`.
6. 작업 출력 설정 창에서 다음을 수행합니다.
   - 파일 유형에서 JSON(JavaScript Object Notation)을 선택합니다.
   - 사용 가능한 Amazon S3 버킷 목록을 보려면 S3 위치를 선택하고 사용할 버킷을 선택합니다. Amazon S3 그런 다음 찾아보기를 선택합니다. 폴더 목록에서 `databrew-output`를 선택하고 선택을 선택합니다.
7. 액세스 권한 창에서를 선택합니다`AwsGlueDataBrewDataAccessRole`. 이는 DataBrew가 사용자를 대신하여 Amazon S3 버킷에 액세스할 수 있도록 하는 서비스 연결 역할입니다.
8. 작업 생성 및 실행을 선택합니다. DataBrew는 설정으로 작업을 생성한 다음 실행합니다.

9. 작업 실행 기록 창에서 작업 상태가에서 Running로 변경될 때까지 기다립니다Succeeded.
10. 프로필을 보려면 프로필 보기를 선택합니다.

DATASETS 창이 표시됩니다. 잠시 시간을 내어 다음 탭을 살펴보세요.

- 데이터 세트 미리 보기
- 프로필 개요
- 열 통계값
- 데이터 계보 통계

6단계: 데이터 세트 변환

지금까지는 데이터 세트의 샘플에서만 레시피를 테스트했습니다. 이제 DataBrew 레시피 작업을 생성하여 전체 데이터 세트를 변환할 차례입니다.

작업이 실행되면 DataBrew는 데이터 세트의 모든 데이터에 레시피를 적용하고 변환된 데이터를 Amazon S3 버킷에 기록합니다. 변환된 데이터는 원래 데이터 세트와 별개입니다. DataBrew는 소스 데이터를 변경하지 않습니다.

계속하기 전에 계정에 쓸 수 있는 Amazon S3 버킷이 있는지 확인합니다. 해당 버킷에서 폴더를 생성하여 DataBrew의 작업 출력을 캡처합니다. 이 단계를 수행하려면 다음 절차를 사용합니다.

작업 출력을 캡처할 S3 버킷 및 폴더를 생성하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ Amazon S3 콘솔을 엽니다.

   Amazon S3 버킷을 이미 사용할 수 있고 해당 버킷에 대한 쓰기 권한이 있는 경우 다음 단계를 건너뜁니다.
2. Amazon S3 버킷이 없는 경우 버킷 생성을 선택합니다. 버킷 이름에 새 버킷의 고유한 이름을 입력합니다. 버킷 생성을 선택합니다.
3. 버킷 목록에서 사용할 버킷을 선택합니다.
4. 폴더 생성을 선택합니다.
5. 폴더 이름에 databrew-output를 입력하고 폴더 생성을 선택합니다.

작업을 포함할 Amazon S3 버킷과 폴더를 생성한 후 다음 절차에 따라 작업을 실행합니다.

레시피 작업을 생성하고 실행하려면

1. 탐색 창에서 작업을 선택합니다.
2. 레시피 작업 탭에서 작업 생성을 선택합니다.
3. 작업 이름에를 입력합니다`chess-winner-summary`.
4. 작업 유형에서 레시피 작업 생성을 선택합니다.
5. 작업 입력 창에서 다음을 수행합니다.
   - 실행 시 데이터 세트를 선택합니다.
   - 사용 가능한 데이터 세트 목록을 보려면 데이터 세트 선택을 선택하고를 선택합니다`chess-games`.
   - 레시피 선택을 선택하여 사용 가능한 레시피 목록을 확인하고를 선택합니다`chess-project-recipe`.
6. 작업 출력 설정 창에서 다음을 수행합니다.
   - 파일 유형 - CSV(쉼표로 구분된 값)를 선택합니다.
   - S3 위치 - 사용 가능한 Amazon S3 버킷 목록을 보려면이 필드를 선택하고 사용할 버킷을 선택합니다. 그런 다음 찾아보기를 선택합니다. 폴더 목록에서 `databrew-output`를 선택하고 선택을 선택합니다.
7. 액세스 권한 창에서를 선택합니다`AwsGlueDataBrewDataAccessRole`. 이 서비스 연결 역할을 통해 DataBrew는 사용자를 대신하여 Amazon S3 버킷에 액세스할 수 있습니다.
8. 작업 생성 및 실행을 선택합니다. DataBrew는 설정으로 작업을 생성한 다음 실행합니다.
9. 작업 실행 기록 창에서 작업 상태가에서 `Running`로 변경될 때까지 기다립니다`Succeeded`.
10. 출력을 선택하여 Amazon S3 콘솔에 액세스합니다. S3 버킷을 선택한 다음 작업 출력에 `databrew-output` 액세스할 폴더를 선택합니다.
11. (선택 사항) 다운로드를 선택하여 파일을 다운로드하고 내용을 봅니다.

7단계: (선택 사항) 정리

연습이 완료되었습니다. 생성한 DataBrew 및 Amazon S3 리소스를 계속 사용하거나 삭제할 수 있습니다.

리소스를 정리하려면

1. https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 열고 탐색 창에서 프로젝트를 선택합니다.
2. 프로젝트(샘플 프로젝트)를 선택합니다. 작업에 대해 삭제를 선택합니다.
3. 샘플 프로젝트 삭제 창에서 연결된 레시피 삭제를 선택합니다. 그런 다음 삭제를 선택합니다. 프로젝트는 레시피 및 작업과 함께 삭제됩니다.
4. 탐색 창에서 데이터 세트를 선택합니다.
5. 데이터 세트(`chess-games`)를 선택하고 작업에서 삭제를 선택합니다.
6. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다. `databrew-output` 폴더와 그 내용을 삭제합니다.

   (선택 사항) Amazon S3 버킷이 더 이상 필요하지 않은 경우 삭제할 수 있습니다.

를 사용하여 데이터에 연결AWS Glue DataBrew

에서 데이터 세트AWS Glue DataBrew는 파일에서 업로드되거나 다른 곳에 저장된 데이터를 나타냅니다. 예를 들어 데이터는 Amazon S3, 지원되는 JDBC 데이터 소스 또는AWS Glue데이터 카탈로그에 저장할 수 있습니다. 파일을 DataBrew에 직접 업로드하지 않는 경우 데이터 세트에는 DataBrew가 데이터에 연결하는 방법에 대한 세부 정보도 포함됩니다.

데이터 세트(예: `inventory-dataset`)를 생성할 때 연결 세부 정보를 한 번만 입력합니다. 이 시점부터 DataBrew는 기본 데이터에 액세스할 수 있습니다. 이 접근 방식을 사용하면 연결 세부 정보나 파일 형식에 대해 걱정할 필요 없이 프로젝트를 생성하고 데이터에 대한 변환을 개발할 수 있습니다.

주제

데이터 소스에 지원되는 파일 유형

다음 파일 요구 사항은 Amazon S3에 저장된 파일과 로컬 드라이브에서 업로드한 파일에 적용됩니다. DataBrew는 쉼표로 구분된 값(CSV), Microsoft Excel, JSON, ORC 및 Parquet 파일 형식을 지원합니다. 파일이 지원되는 유형 중 하나인 경우 비표준 확장자가 있거나 확장자가 없는 파일을 사용할 수 있습니다.

DataBrew가 파일 유형을 추론할 수 없는 경우 올바른 파일 유형(CSV, Excel, JSON, ORC 또는 Parquet)을 직접 선택해야 합니다. 압축된 CSV, JSON, ORC 및 Parquet 파일이 지원되지만 CSV 및 JSON 파일에는 압축 코덱이 파일 확장명으로 포함되어야 합니다. 폴더를 가져오는 경우 폴더의 모든 파일은 파일 유형이 동일해야 합니다.

파일 형식과 지원되는 압축 알고리즘은 다음 표에 나와 있습니다.

> **Note**
>
> CSV, Excel 및 JSON 파일은 유니코드(UTF-8)로 인코딩되어야 합니다.

형식	파일 확장명(선택 사항)	압축 파일의 확장(필수)

쉼표로 구분된 값	`.csv`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Microsoft Excel 통합 문서	`.xlsx`	압축 지원 없음

JSON(JSON 문서 및 JSON 줄)	`.json, .jsonl`	`.gz`
		`.snappy`
		`.lz4`
		`.bz2`
		`.deflate`

Apache ORC	`.orc`	`.zlib` `.snappy`

Apache Parquet	`.parquet`	`.gz`
		`.snappy`
		`.lz4`

데이터 소스 및 출력에 지원되는 연결

DataBrew 레시피 작업에 대해 다음 데이터 소스에 연결할 수 있습니다. 여기에는 DataBrew에 직접 업로드하는 파일이 아닌 모든 데이터 소스가 포함됩니다. 사용 중인 데이터 소스를 데이터베이스, 데이터 웨어하우스 또는 기타 데이터 소스라고 할 수 있습니다. 모든 데이터 공급자를 데이터 소스 또는 연결이라고 합니다.

다음 중 하나를 데이터 소스로 사용하여 데이터 세트를 생성할 수 있습니다.

Amazon RDS를 통해 지원되는 Amazon S3 AWS Glue Data Catalog또는 JDBC 데이터베이스를 사용하여 DataBrew 레시피 작업을 출력할 수도 있습니다. Amazon AppFlow 및AWS Data Exchange는 DataBrew 레시피 작업의 출력에 대해 지원되지 않습니다.

- Amazon S3

  S3를 사용하여 원하는 양의 데이터를 저장하고 보호할 수 있습니다. 데이터 세트를 생성하려면 DataBrew가 데이터 파일에 액세스할 수 있는 S3 URL을 지정합니다. 예를 들면 다음과 같습니다. `s3://your-bucket-name/inventory-data.csv`

  DataBrew는 S3 폴더의 모든 파일을 읽을 수도 있습니다. 즉, 여러 파일에 걸쳐 있는 데이터 세트를 생성할 수 있습니다. 이렇게 하려면 형식으로 S3 URL을 지정합니다`s3://your-bucket-name/your-folder-name/`.

  DataBrew는 Standard, Reduced Redundancy, Standard-IA 및 Amazon S3 S3 스토리지 클래스만 지원합니다. DataBrew는 다른 스토리지 클래스가 있는 파일을 무시합니다. 또한 DataBrew는 빈 파일(0바이트를 포함하는 파일)을 무시합니다. Amazon S3 스토리지 클래스에 대한 자세한 내용은 Amazon S3 콘솔 사용 설명서의 Amazon S3 스토리지 클래스 사용을 참조하세요. Amazon S3

- AWS Glue Data Catalog

  데이터 카탈로그를 사용하여AWS클라우드에 저장된 데이터에 대한 참조를 정의할 수 있습니다. 데이터 카탈로그를 사용하면 다음 서비스의 개별 테이블에 대한 연결을 구축할 수 있습니다.

  - 데이터 카탈로그 Amazon S3
  - 데이터 카탈로그 Amazon Redshift
  - 데이터 카탈로그 Amazon RDS
  - AWS Glue

DataBrew는 Amazon S3 폴더의 모든 파일을 읽을 수도 있습니다. 즉, 여러 파일에 걸쳐 있는 데이터 세트를 생성할 수 있습니다. 이렇게 하려면 다음 형식으로 Amazon S3 URL을 지정합니다. `s3://your-bucket-name/your-folder-name/`

DataBrew와 함께 사용하려면에 정의된 Amazon S3 테이블에AWS Glue Data Catalog라는 테이블 속성이 추가되어 있어야 합니다. `classification`이 속성은 데이터 형식을 `csv`, `json`또는 로`parquet`,를 `typeOfData`로 식별합니다`file`. 테이블이 생성될 때 테이블 속성이 추가되지 않은 경우AWS Glue콘솔을 사용하여 추가할 수 있습니다.

DataBrew는 Amazon S3 스토리지 클래스 Standard, Reduced Redundancy, Standard-IA 및 S3 One Zone-IA만 지원합니다. DataBrew는 다른 스토리지 클래스가 있는 파일을 무시합니다. 또한 DataBrew는 빈 파일(0바이트를 포함하는 파일)을 무시합니다. Amazon S3 스토리지 클래스에 대한 자세한 내용은 Amazon S3 콘솔 사용 설명서의 Amazon S3 스토리지 클래스 사용을 참조하세요. Amazon S3

적절한 리소스 정책이 생성된 경우 DataBrew는 다른 계정의AWS Glue Data Catalog S3 테이블에 액세스할 수도 있습니다. 콘솔의AWS Glue설정 탭에 있는 데이터 카탈로그에서 정책을 생성할 수 있습니다. 다음은 단일에 대한 정책의 예입니다AWS 리전.

> **Warning**
>
> 이는의 데이터 카탈로그에 `*$ACCOUNT_TO*` 대한 무제한 액세스 권한을 부여하는 매우 허용적인 리소스 정책입니다`*$ACCOUNT_FROM*`. 대부분의 경우 리소스 정책을 특정 카탈로그 또는 테이블로 잠그는 것이 좋습니다. 자세한 내용은 AWS Glue개발자 안내서의 AWS Glue 액세스 제어를 위한 리소스 정책을 참조하세요.

경우에 따라 프로젝트를 생성하거나AWS Glue DataBrew에 있는 S3 위치를 `*$ACCOUNT_FROM*` 가리키는의AWS Glue Data Catalog S3 테이블`*$ACCOUNT_TO*`을 사용하여에서 작업을 실행할 수 있습니다`*$ACCOUNT_FROM*`. 이러한 경우에서 프로젝트 및 작업을 생성할 때 사용되는 IAM 역할에는에서 해당 S3 위치에 객체를 나열하고 가져올 수 있는 권한이 `*$ACCOUNT_TO*` 있어야 합니다`*$ACCOUNT_FROM*`. 자세한 내용은 AWS Glue개발자 안내서의 교차 계정 액세스 권한 부여를 참조하세요.

- JDBC 드라이버를 사용하여 연결된 데이터

  지원되는 JDBC 드라이버를 사용하여 데이터에 연결하여 데이터 세트를 생성할 수 있습니다. 자세한 내용은 에서 드라이버 사용AWS Glue DataBrew 단원을 참조하십시오.

DataBrew는 Java Database Connectivity(JDBC)를 사용하여 다음 데이터 소스를 공식적으로 지원합니다.

- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Amazon Redshift
- Spark용 Snowflake 커넥터

데이터 소스는 DataBrew에서 연결할 수 있는 위치에 위치할 수 있습니다. 이 목록에는 테스트한 JDBC 연결만 포함되어 있으므로 지원할 수 있습니다.

Spark 데이터 소스용 Amazon Redshift 및 Snowflake 커넥터는 다음 방법 중 하나로 연결할 수 있습니다.

- 테이블 이름을 사용합니다.
- 여러 테이블 및 작업에 걸쳐 있는 SQL 쿼리를 사용합니다.

SQL 쿼리는 프로젝트 또는 작업 실행을 시작할 때 실행됩니다.

목록에 없는 JDBC 드라이버가 필요한 데이터에 연결하려면 드라이버가 JDK 8과 호환되는지 확인합니다. 드라이버를 사용하려면 DataBrew에 대한 IAM 역할을 사용하여 드라이버에 액세스할 수 있는 버킷의 S3에 저장합니다. 그런 다음 데이터세트가 드라이버 파일을 가리키도록 합니다. 자세한 내용은 에서 드라이버 사용AWS Glue DataBrew 단원을 참조하십시오.

SQL 기반 데이터 세트에 대한 쿼리 예제:

```
SELECT
    *
FROM
    public.customer as c
JOIN
    public.customer_address as ca on c.current_address=ca.current_address
WHERE
    ca.address_id>0 AND ca.address_id<10001 ORDER BY ca.address_id
```

사용자 지정 SQL의 제한 사항

JDBC 연결을 사용하여 DataBrew 데이터 세트에 대한 데이터에 액세스하는 경우 다음 사항에 유의하세요.

- AWS Glue DataBrew는 데이터 세트 생성의 일부로 제공하는 사용자 지정 SQL을 검증하지 않습니다. SQL 쿼리는 프로젝트 또는 작업 실행을 시작할 때 실행됩니다. DataBrew는 사용자가 제공한 쿼리를 가져와 기본 또는 제공된 JDBC 드라이버를 사용하여 데이터베이스 엔진에 전달합니다.
- 잘못된 쿼리로 생성된 데이터 세트는 프로젝트 또는 작업에 사용될 때 실패합니다. 데이터 세트를 생성하기 전에 쿼리를 검증합니다.
- SQL 검증 기능은 Amazon Redshift 기반 데이터 소스에서만 사용할 수 있습니다.
- 프로젝트에서 데이터 세트를 사용하려면 프로젝트 로드 중에 제한 시간이 발생하지 않도록 SQL 쿼리 런타임을 3분 미만으로 제한합니다. 프로젝트를 생성하기 전에 쿼리 런타임을 확인합니다.

- Amazon AppFlow

  Amazon AppFlow를 사용하면 Salesforce, Zendesk, Slack, ServiceNow와 같은 타사 Software-as-a-Service(SaaS) 애플리케이션에서 Amazon S3로 데이터를 전송할 수 있습니다. 그런 다음 데이터를 사용하여 DataBrew 데이터 세트를 생성할 수 있습니다.

  Amazon AppFlow에서는 타사 애플리케이션과 대상 애플리케이션 간에 데이터를 전송하는 연결 및 흐름을 생성합니다. DataBrew와 함께 Amazon AppFlow를 사용하는 경우 Amazon AppFlow 대상 애플리케이션이 Amazon S3인지 확인합니다. Amazon S3 이외의 Amazon AppFlow 대상 애플리케이션은 DataBrew 콘솔에 표시되지 않습니다. 타사 애플리케이션에서 데이터를 전송하고 Amazon AppFlow 연결 및 흐름을 생성하는 방법에 대한 자세한 내용은 Amazon AppFlow 설명서를 참조하세요.

  DataBrew의 데이터 세트 탭에서 새 데이터 세트 연결을 선택하고 Amazon AppFlow를 클릭하면 Amazon S3로 구성된 Amazon AppFlow의 모든 흐름이 대상 애플리케이션으로 표시됩니다. 데이터 세트에 흐름의 데이터를 사용하려면 해당 흐름을 선택합니다.

  DataBrew 콘솔에서 Amazon AppFlow에 대한 흐름 생성, 흐름 관리 및 세부 정보 보기를 선택하면 해당 작업을 수행할 수 있도록 Amazon AppFlow 콘솔이 열립니다.

  Amazon AppFlow에서 데이터 세트를 생성한 후 데이터 세트 세부 정보 또는 작업 세부 정보를 볼 때 흐름을 실행하고 가장 최근의 흐름 실행 세부 정보를 볼 수 있습니다. DataBrew에서 흐름을 실행하면 데이터 세트가 S3에서 업데이트되고 DataBrew에서 사용할 준비가 됩니다.

  DataBrew 콘솔에서 Amazon AppFlow 흐름을 선택하여 데이터 세트를 생성할 때 다음과 같은 상황이 발생할 수 있습니다.

- 데이터가 집계되지 않음 - 흐름 트리거가 온디맨드 실행 또는 전체 데이터 전송으로 일정에 따라 실행인 경우 DataBrew 데이터 세트를 생성하기 전에 흐름에 대한 데이터를 집계해야 합니다. 흐름을 집계하면 흐름의 모든 레코드가 단일 파일로 결합됩니다. 트리거 유형이 증분 데이터 전송으로 일정에 따라 실행 또는 이벤트에서 실행인 흐름은 집계가 필요하지 않습니다. Amazon AppFlow에서 데이터를 집계하려면 흐름 구성 편집 > 대상 세부 정보 > 추가 설정 > 데이터 전송 기본 설정을 선택합니다.
- 흐름이 실행되지 않음 - 흐름의 실행 상태가 비어 있는 경우 다음 중 하나를 의미합니다.
  - 흐름 실행 트리거가 온디맨드 실행인 경우 흐름이 아직 실행되지 않은 것입니다.
  - 흐름 실행을 위한 트리거가 이벤트에서 실행이면 트리거 이벤트가 아직 발생하지 않은 것입니다.
  - 흐름 실행 트리거가 일정에 따라 실행인 경우 예약된 실행이 아직 발생하지 않은 것입니다.

  흐름이 있는 데이터 세트를 생성하기 전에 해당 흐름에 대해 흐름 실행을 선택합니다.

  자세한 내용은 Amazon AppFlow 사용 설명서의 Amazon AppFlow 흐름을 참조하세요. AppFlow
- AWS Data Exchange

  에서 사용할 수 있는 수백 개의 타사 데이터 소스 중에서 선택할 수 있습니다AWS Data Exchange. 이러한 데이터 소스를 구독하면 up-to-date 버전의 데이터를 얻을 수 있습니다.

  데이터 세트를 생성하려면 구독하고 사용할 권한이 있는AWS Data Exchange데이터 제품의 이름을 지정합니다.

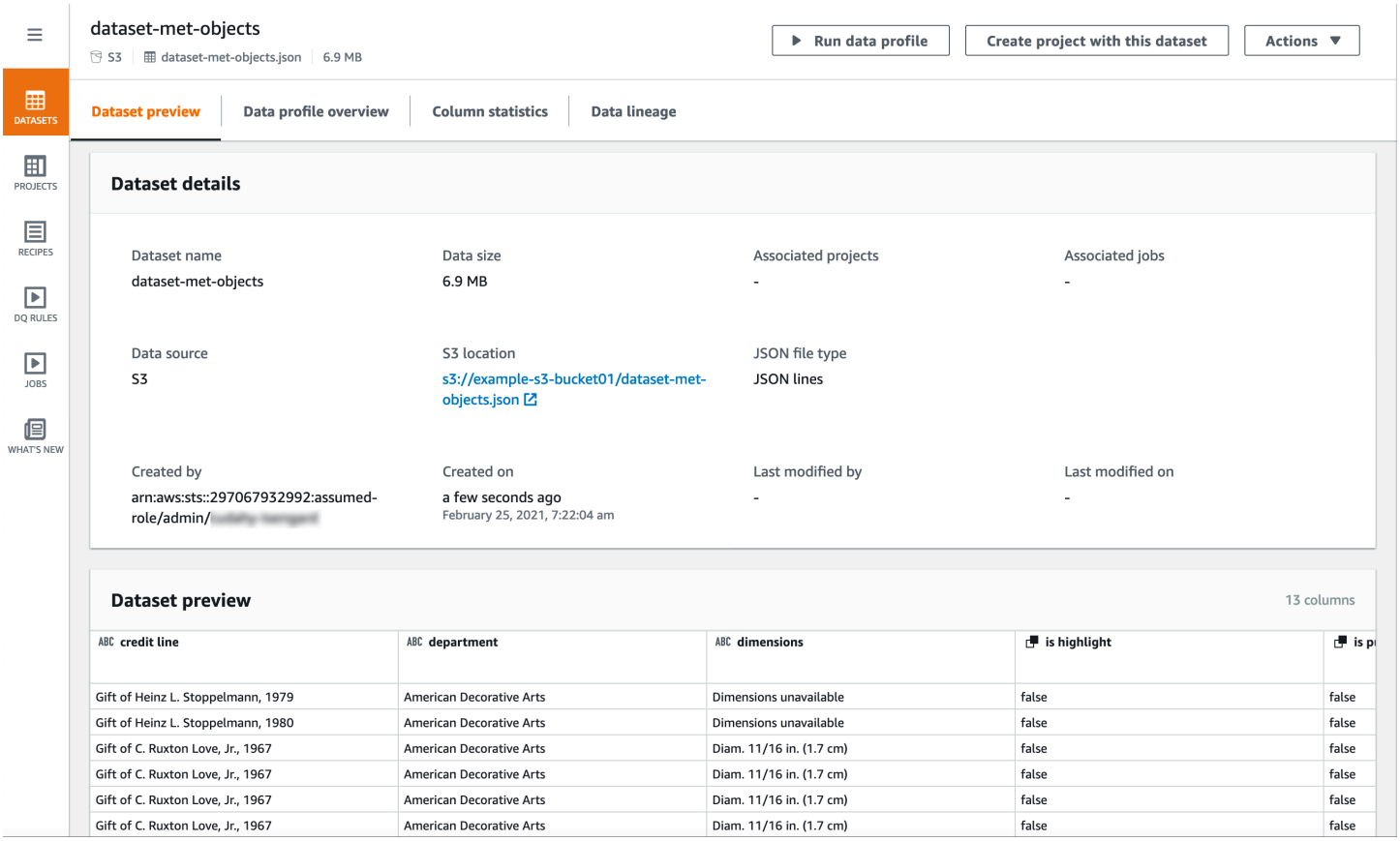
에서 데이터 세트 사용AWS Glue DataBrew

DataBrew 콘솔에서 데이터 세트 목록을 보려면 왼쪽에서 데이터 세트를 선택합니다. 데이터 세트 페이지에서 이름을 클릭하거나 컨텍스트 메뉴에서 작업, 편집을 선택하여 각 데이터 세트에 대한 세부 정보를 볼 수 있습니다.

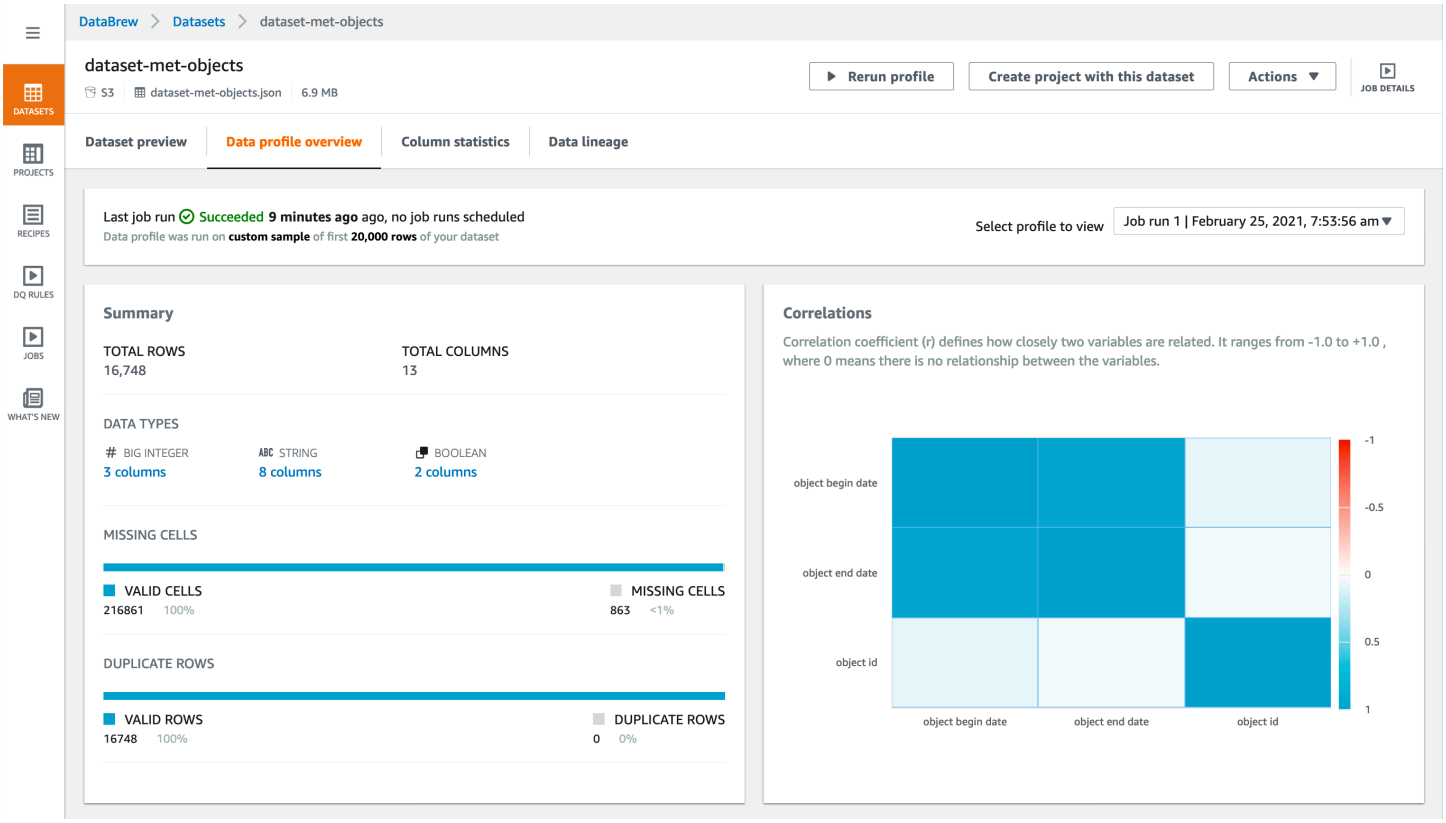
새 데이터 세트를 생성하려면 데이터 세트, 새 데이터 세트 연결을 선택합니다. 데이터 소스마다 연결 파라미터가 다르며 DataBrew가 연결할 수 있도록 이러한 파라미터를 입력합니다. 연결을 저장하고 데이터 세트 생성을 선택하면 DataBrew가 데이터에 연결하고 데이터 로드를 시작합니다. 자세한 내용은 데이터에 연결 단원을 참조하십시오.

데이터 세트 페이지에는 데이터를 탐색하는 데 도움이 되는 다음과 같은 요소가 있습니다.

데이터 세트 미리 보기 -이 탭에서는 다음과 같이 데이터 세트에 대한 연결 정보와 데이터 세트의 전체 구조에 대한 개요를 찾을 수 있습니다.

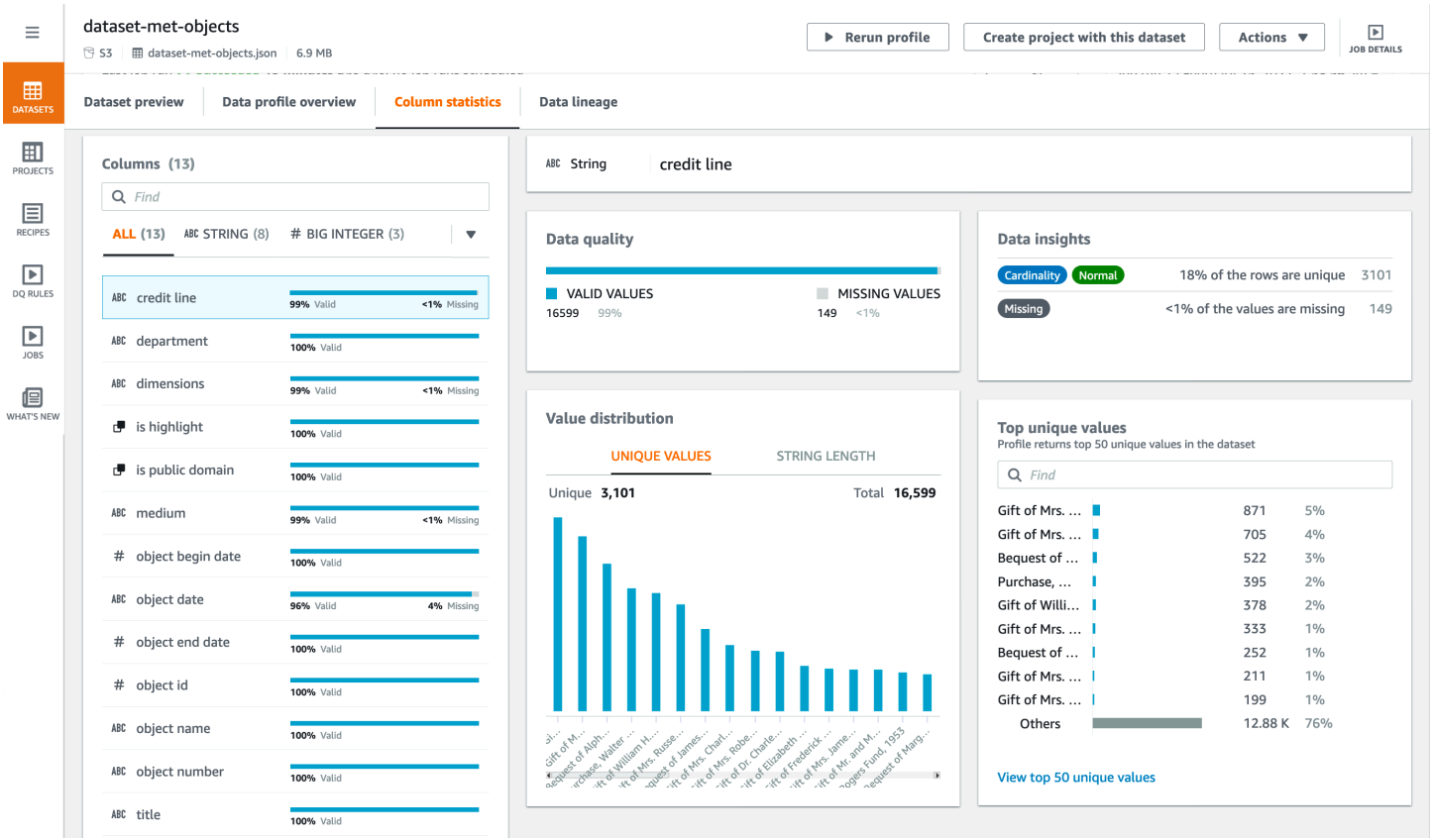


데이터 프로필 개요 -이 탭에서는 다음과 같이 데이터 세트에 대한 통계 및 볼륨 측정의 그래픽 데이터 프로필을 찾을 수 있습니다.

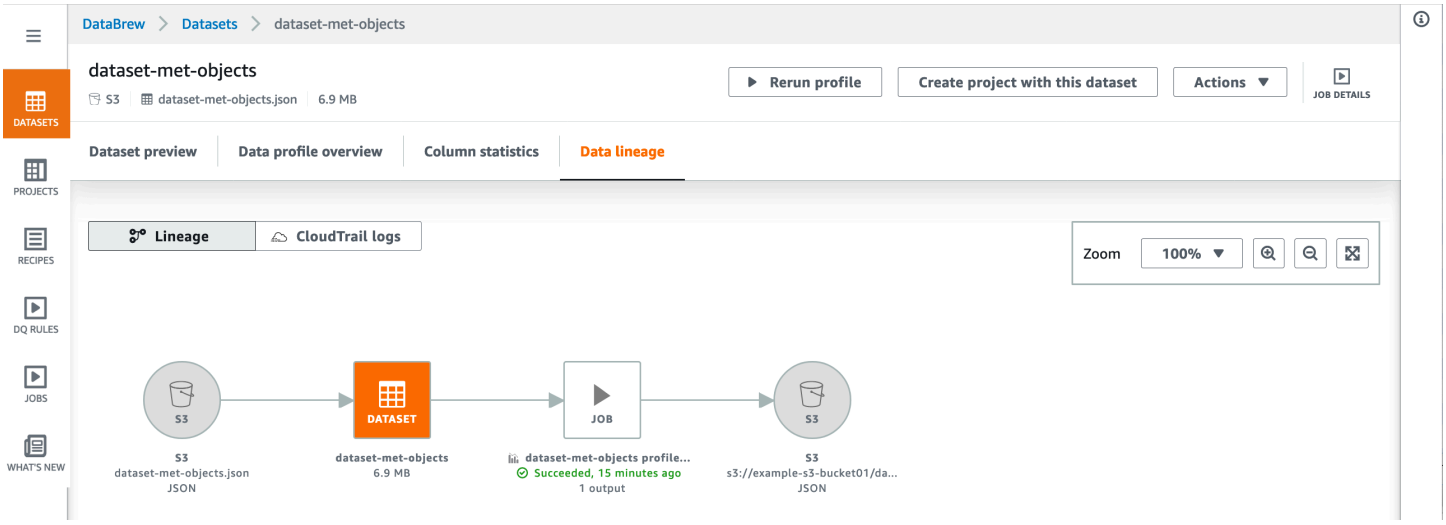


> **Note**
>
> 데이터 프로파일을 생성하려면 데이터 세트에서 DataBrew 프로파일 작업을 실행합니다. 이를 위한 자세한 방법은 5단계: 데이터 프로필 생성 섹션을 참조하세요.

열 통계 -이 탭에서는 다음과 같이 데이터 세트의 각 열에 대한 자세한 통계를 찾을 수 있습니다.



데이터 계보 -이 탭은 다음과 같이 데이터 세트가 생성된 방식과 DataBrew에서 사용되는 방식을 그래픽으로 보여줍니다.



주제

- 데이터 세트 삭제

데이터 세트 삭제

데이터 세트가 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 데이터 세트를 삭제해도 기본 데이터 소스에는 어떤 식으로든 영향을 주지 않습니다. DataBrew가 데이터 소스에 액세스하는 데 사용한 정보만 제거합니다.

다른 DataBrew 리소스가 데이터 세트를 사용하는 경우 데이터 세트를 삭제할 수 없습니다. 예를 들어 현재 데이터 세트를 사용하는 DataBrew 프로젝트가 있는 경우 데이터 세트를 삭제하기 전에 먼저 프로젝트를 삭제합니다.

데이터 세트를 삭제하려면 탐색 창에서 데이터 세트를 선택합니다. 삭제할 데이터 세트를 선택한 다음 작업에서 삭제를 선택합니다.

데이터에 연결

다음 데이터 소스에 연결하는 방법에 대한 자세한 내용은 해당하는 섹션을 선택합니다.

- AWS Glue Data Catalog - Data Catalog를 사용하여 다음 서비스를 포함하여AWS클라우드에 저장된 데이터 객체에 대한 참조를 정의할 수 있습니다.
  - Amazon Redshift
  - Aurora MySQL
  - Aurora PostgreSQL
  - Amazon RDS for MySQL
  - Amazon RDS for PostgreSQL

  DataBrew는 데이터 카탈로그 리소스에 적용된 모든 Lake Formation 권한을 인식하므로 DataBrew 사용자는 권한이 있는 경우에만 이러한 리소스에 액세스할 수 있습니다.

  데이터 세트를 생성하려면 데이터 카탈로그 데이터베이스 이름과 테이블 이름을 지정합니다. DataBrew는 다른 연결 세부 정보를 처리합니다.
- AWS데이터 교환 -AWS Data Exchange에서 사용할 수 있는 수백 개의 타사 데이터 소스 중에서 선택할 수 있습니다. 이러한 데이터 소스를 구독하면 항상 up-to-date 버전의 데이터가 제공됩니다.

  데이터 세트를 생성하려면 구독하거나 사용할 권한이 있는 Data Exchange 데이터 제품의 이름을 지정합니다.
- JDBC 드라이버 연결 - DataBrew를 JDBC 호환 데이터 소스에 연결하여 데이터 세트를 생성할 수 있습니다. DataBrew는 JDBC를 통해 다음 소스에 연결할 수 있도록 지원합니다.

- Amazon Redshift
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Snowflake

주제

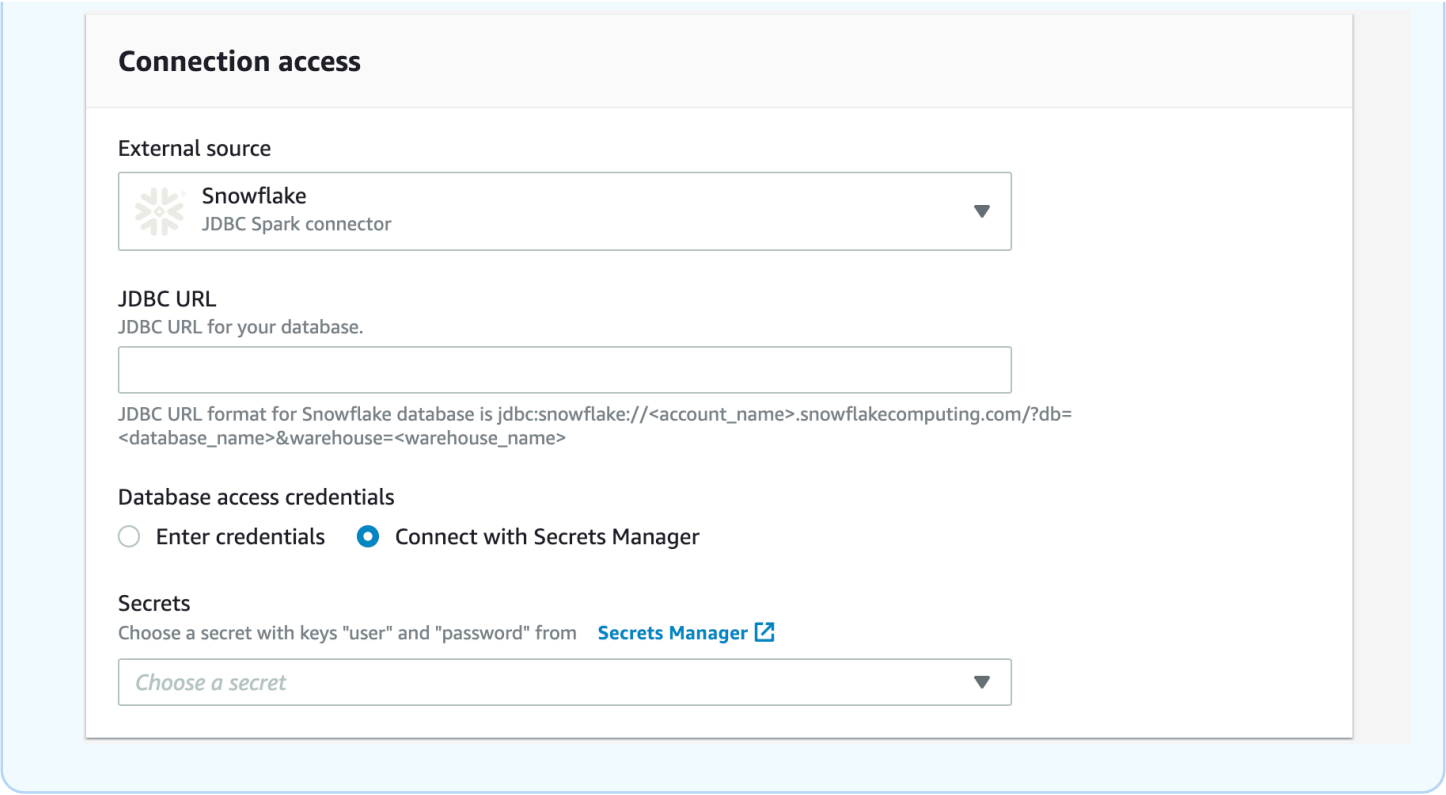
에서 드라이버 사용AWS Glue DataBrew

데이터베이스 드라이버는 Java Database Connectivity(JDBC)와 같은 데이터베이스 연결 프로토콜을 구현하는 파일 또는 URL입니다. 드라이버는 특정 데이터베이스 관리 시스템(DBMS)과 다른 시스템 간의 어댑터 또는 변환기 역할을 합니다.

이 경우AWS Glue DataBrew가 데이터에 연결할 수 있습니다. 그런 다음 지원되는 데이터 소스에서 테이블 또는 뷰와 같은 데이터베이스 객체에 액세스할 수 있습니다. 사용 중인 데이터 소스를 데이터베이스, 데이터 웨어하우스 또는 기타 데이터 소스라고 할 수 있습니다. 그러나이 설명서에서는 모든 데이터 공급자를 데이터 소스 또는 연결이라고 합니다.

JDBC 드라이버 또는 jar 파일을 사용하려면 필요한 파일을 다운로드하여 S3 버킷에 넣습니다. 데이터에 액세스하는 데 사용하는 IAM 역할에는 두 드라이버 파일에 대한 읽기 권한이 있어야 합니다.

> **Note**
>
> With AWS Glue 4.0, Snowflake에 데이터 소스로 연결하는 것은 기본적으로 지원됩니다. 사용자 지정 `jar` 파일을 제공할 필요가 없습니다. 에서AWS Glue DataBrew Snowflake를 외부 소스 연결로 선택하고 Snowflake 인스턴스의 URL을 제공합니다. URL은 `https://account_identifier.snowflakecomputing.com` 양식의 호스트 이름을 사용합니다. 데이터 액세스 자격 증명, Snowflake 데이터베이스 이름 및 Snowflake 스키마 이름을 제공합니다. 또한 Snowflake 사용자에게 기본 웨어하우스 세트가 없는 경우 웨어하우스 이름을 제공해야 합니다.
>
> Snowflake 연결은AWS Secrets Manager보안 암호를 사용하여 자격 증명 정보를 제공합니다. 의 프로젝트 및 작업 역할에는이 보안 암호를 읽을 수 있는 권한이 있어야 합니다.



DataBrew에서 드라이버를 사용하려면

1. 제품에서 제공하는 방법을 사용하여 현재 사용 중인 데이터 소스의 버전을 확인합니다.
2. 필요한 커넥터 및 드라이버의 최신 버전을 찾습니다. 데이터 공급자 웹 사이트에서이 정보를 찾을 수 있습니다.
3. JDBC 파일의 필수 버전을 다운로드합니다. 이러한 파일은 일반적으로 Java ARchives(.JAR) 파일로 저장됩니다.
4. 콘솔에서 S3 버킷으로 드라이버를 업로드하거나 .JAR 파일에 대한 S3 경로를 제공합니다.
5. 클래스, 인스턴스 등과 같은 기본 연결 세부 정보를 입력합니다.
6. Virtual Private Cloud(VPC) 정보와 같이 데이터 소스에 필요한 추가 구성 정보를 입력합니다.

지원되는 JDBC 드라이버

제품	지원되는 버전	드라이버 지침 및 다운로드	지원되는 SQL 쿼리
Microsoft SQL Server	v6.x 이상	SQL Server용 Microsoft JDBC 드라이버	지원되지 않음
MySQL	v5.1 이상	MySQL 커넥터	지원되지 않음
Oracle	v11.2 이상	Oracle JDBC 다운로드	지원되지 않음
PostgreSQL	v4.2.x 이상	PostgreSQL JDBC 드라이버	지원되지 않음
Amazon Redshift	v4.1 이상	JDBC를 사용하여 Amazon Redshift에 연결	지원됨
Snowflake	Snowflake 버전을 보려면 Snowflake 설명서에 설명된 대로 CURRENT ERSION을	Snowflake에 연결하려면 다음 두 가지가 모두 필요합니다. • Snowflake JDBC 드라이버 • Spark용 Snowflake 커넥터	지원됨

제품	지원되는 버전	드라이버 지침 및 다운로드	지원되는 SQL 쿼리
	사용합니다.		

DataBrew가 기본적으로 지원하는 것과 다른 버전의 드라이버가 필요한 데이터베이스 또는 데이터 웨어하우스에 연결하려면 원하는 JDBC 드라이버를 제공할 수 있습니다. 드라이버는 JDK 8 또는 Java 8과 호환되어야 합니다. 데이터베이스의 최신 드라이버 버전을 찾는 방법에 대한 지침은 섹션을 참조하세요에서 드라이버 사용AWS Glue DataBrew.

DataBrew를 사용하여 텍스트 파일의 데이터에 연결

DataBrew가 지원하는 입력 파일에 대해 다음 형식 옵션을 구성할 수 있습니다.

- 쉼표로 구분된 값(CSV) 파일
  - 구분 기호

    기본 구분 기호는 .csv 파일의 쉼표입니다. 파일에서 다른 구분 기호를 사용하는 경우 데이터 세트를 생성할 때 추가 구성 섹션에서 CSV 구분 기호의 구분 기호를 선택합니다. .csv 파일에는 다음 구분 기호가 지원됩니다.

    - 쉼표(,)
    - 콜론(:)
    - 세미콜론(;)
    - 파이프(|)
    - 탭(\t)
    - 캐럿(^)
    - 백슬래시(\)
    - 공간
  - 열 헤더 값

    CSV 파일에는 헤더 행이 파일의 첫 번째 행으로 포함될 수 있습니다. 그렇지 않으면 DataBrew가 헤더 행을 생성합니다.

    - CSV 파일에 헤더 행이 포함된 경우 첫 번째 행을 헤더로 처리를 선택합니다. 이렇게 하면 CSV 파일의 첫 번째 행이 열 헤더 값을 포함하는 것으로 처리됩니다.
    - CSV 파일에 헤더 행이 포함되지 않은 경우 기본 헤더 추가를 선택합니다. 이렇게 하면 DataBrew는 파일에 대한 헤더 행을 생성하고 데이터의 첫 번째 행을 헤더 값을 포함하는 것으로 취급하지 않습니다. DataBrew가 생성하는 헤더는 밑줄과 파일의 각 열에 대한 숫자로 구성되며 형식은 `Column_1`, `Column_2Column_3`, 등입니다.
- JSON 파일

  DataBrew는 JSON 파일과 JSON 문서에 대해 두 가지 형식을 지원합니다. JSON Lines 파일에는 행당 하나의 행이 포함됩니다. JSON 문서 파일에서 모든 행은 단일 JSON 구조 또는 배열에 포함됩니다. JSON 데이터 세트를 생성할 때 추가 구성 섹션에서 JSON 파일 유형을 지정할 수 있습니다. 기본 형식은 JSON Lines입니다.
- Excel 파일

  DataBrew의 Excel 시트에는 다음이 적용됩니다.

  - Excel 시트 로드

    기본적으로 DataBrew는 Excel 파일의 첫 번째 시트를 로드합니다. 그러나 Excel 데이터 세트를 생성할 때 추가 구성 섹션에서 다른 시트 번호 또는 시트 이름을 지정할 수 있습니다.
  - 열 헤더 값

    Excel 시트에는 파일의 첫 번째 행으로 헤더 행이 포함될 수 있지만, 그렇지 않으면 DataBrew가 헤더 행을 생성합니다.
    - Excel 시트에 헤더 행이 포함된 경우 첫 번째 행을 헤더로 처리를 선택합니다. 이렇게 하면 Excel 시트의 첫 번째 행이 열 헤더 값을 포함하는 것으로 처리됩니다.
    - Excel 파일에 헤더 행이 포함되지 않은 경우 기본 헤더 추가를 선택합니다. 이렇게 하면 DataBrew가 파일에 대한 헤더 행을 생성하고 데이터의 첫 번째 행을 헤더 값을 포함하는 것으로 취급하지 않도록 지정할 수 있습니다. DataBrew가 생성하는 헤더는 밑줄과 파일의 각 열에 대한 숫자로 구성되며 형식은 `Column_1`, `Column_2Column_3`, 등입니다.

Amazon S3의 여러 파일에 데이터 연결

DataBrew 콘솔을 사용하면 Amazon S3 버킷 및 폴더를 탐색하고 데이터 세트에 대한 파일을 선택할 수 있습니다. 하지만 데이터 세트를 하나의 파일로 제한할 필요는 없습니다.

이름이 인 폴더my-databrew-bucket가 포함된 이름이 인 S3 버킷이 있다고 가정해 보겠습니다databrew-input. 해당 폴더에 파일 형식과 파일 .json 확장명이 동일한 여러 개의 JSON 파일이 있다고 가정합니다. 콘솔에서의 소스 URL을 지정할 수 있습니다s3://my-databrew-bucket/databrew-input/. DataBrew 콘솔에서이 폴더를 선택할 수 있습니다. 데이터 세트는 해당 폴더의 모든 JSON 파일로 구성됩니다.

DataBrew는 다음 조건이 참인 경우에만 S3 폴더의 모든 파일을 처리할 수 있습니다.

- 폴더의 모든 파일 형식은 동일합니다.
- 폴더에 있는 모든 파일의 파일 확장명은 동일합니다.

지원되는 파일 형식 및 확장명에 대한 자세한 내용은 섹션을 참조하세요DataBrew input formats.

여러 파일을 데이터 세트로 사용할 때의 스키마

여러 파일을 DataBrew 데이터 세트로 사용하는 경우 스키마는 모든 파일에서 동일해야 합니다. 그렇지 않으면 프로젝트 Workspace는 여러 파일에서 스키마 중 하나를 자동으로 선택하고 나머지 데이터 세트 파일을 해당 스키마에 맞추려고 시도합니다. 이 동작으로 인해 프로젝트 Workspace 중에 표시되는 보기가 불규칙해지고 결과적으로 작업 출력도 불규칙해집니다.

파일에 다른 스키마가 있어야 하는 경우 여러 데이터 세트를 생성하고 별도로 프로파일링해야 합니다.

Amazon S3에 파라미터화된 경로 사용

경우에 따라 특정 명명 규칙을 따르는 파일이 포함된 데이터 세트 또는 여러 Amazon S3 폴더에 걸쳐 있을 수 있는 데이터 세트를 생성할 수 있습니다. 또는 특정 파라미터에 따라 경로가 있는 S3 위치에서 주기적으로 생성되는 동일한 구조화된 데이터에 동일한 데이터 세트를 재사용할 수 있습니다. 예를 들어 데이터 프로덕션 날짜의 라는 경로가 있습니다.

DataBrew는 파라미터화된 S3 경로로이 접근 방식을 지원합니다. 파라미터화된 경로는 정규식 또는 사용자 지정 경로 파라미터 또는 둘 다를 포함하는 Amazon S3 URL입니다.

정규식을 사용하여 S3 경로로 데이터 세트 정의

경로의 정규식은 하나 이상의 폴더에서 여러 파일을 일치시키는 동시에 해당 폴더에서 관련 없는 파일을 필터링하는 데 유용할 수 있습니다.

다음은 몇 가지 예입니다.

- 이름이 로 시작하는 폴더의 모든 JSON 파일을 포함하는 데이터 세트를 정의합니다invoice.
- 2020 이름에가 있는 폴더의 모든 파일을 포함하는 데이터 세트를 정의합니다.

데이터 세트 S3 경로에서 정규식을 사용하여 이러한 유형의 접근 방식을 구현할 수 있습니다. 이러한 정규식은 S3 URL의 키에 있는 모든 하위 문자열을 대체할 수 있습니다(버킷 이름은 대체하지 않음).

S3 URL의 키 예제는 다음을 참조하세요. 여기서 my-bucket는 버킷 이름이고, 미국 동부(오하이오)는 AWS리전이며,는 키 이름puppy.png입니다.

```
https://my-bucket.s3.us-west-2.amazonaws.com/puppy.png
```

파라미터화된 S3 경로에서는 두 개의 각도 대괄호(< 및 >) 사이의 모든 문자가 정규식으로 처리됩니다. 두 가지 예는 다음과 같습니다.

- s3://my-databrew-bucket/databrew-input/invoice<.*>/data.json는 이름이 로 시작하는의 모든 하위 폴더 data.json내에서 이름이 databrew-input인 모든 파일과 일치합니다invoice.
- s3://my-databrew-bucket/databrew-input/<.*>2020<.*>/는 폴더의 모든 파일과 이름2020의를 일치시킵니다.

이 예제에서는 0개 이상의 문자와 .* 일치합니다.

> **Note**
>
> 버킷 이름 뒤에 오는 부분인 S3 경로의 키 부분에서만 정규식을 사용할 수 있습니다. 따라서 s3://my-databrew-bucket/<.*>-input/는 유효하지만 s3://my-<.*>-bucket/<.*>-input/ 유효하지 않습니다.

정규식을 테스트하여 원하는 S3 URLs만 일치하고 원하지 않는 URL은 일치하지 않는지 확인하는 것이 좋습니다.

다음은 정규식의 몇 가지 다른 예입니다.

- <\d{2}>는 07 또는와 같이 정확히 두 개의 연속된 숫자로 구성된 문자열과 일치하지03만는 일치하지 않습니다1a2.
- <[a-z]+.*>는 하나 이상의 소문자 라틴 문자로 시작하고 그 뒤에 0개 이상의 다른 문자가 있는 문자열과 일치합니다. 예제는 a3, abc/def또는 a-z이지만는 아닙니다A2.

- <[^/]+>는 슬래시(/)를 제외한 모든 문자가 포함된 문자열과 일치합니다. S3 URL에서 슬래시는 경로의 폴더를 분리하는 데 사용됩니다.
- <.*=.*>는 등호(=)가 포함된 문자열month=02과 일치합니다. 예: abc/day=2, 또는 =10, 그러나는 포함되지 않습니다test.
- <\d.*\d>는 숫자로 시작하고 끝나는 문자열과 일치하며 1abc2, 01-02-03또는와 같이 숫자 사이에 다른 문자를 포함할 수 2020/Jul/21있지만는 포함할 수 없습니다123a.

사용자 지정 파라미터를 사용하여 S3 경로로 데이터 세트 정의

사용자 지정 파라미터를 사용하여 파라미터화된 데이터 세트를 정의하면 S3 위치에 대한 파라미터를 제공하려는 경우 정규식을 사용하는 것보다 이점이 있습니다.

- 정규식의 구문을 알 필요 없이 정규식과 동일한 결과를 얻을 수 있습니다. "starts with" 및 "contains"와 같은 친숙한 용어를 사용하여 파라미터를 정의할 수 있습니다.
- 경로의 파라미터를 사용하여 동적 데이터 세트를 정의할 때 "지난 달" 또는 "지난 24시간"과 같은 시간 범위를 정의에 포함할 수 있습니다. 이렇게 하면 데이터세트 정의가 나중에 새 수신 데이터와 함께 사용됩니다.

다음은 동적 데이터 세트를 사용할 수 있는 몇 가지 예입니다.

- 마지막으로 업데이트된 날짜 또는 기타 의미 있는 속성으로 분할된 여러 파일을 단일 데이터 세트에 연결합니다. 그런 다음 이러한 파티션 속성을 데이터 세트의 추가 열로 캡처할 수 있습니다.
- 데이터 세트의 파일을 특정 조건을 충족하는 S3 위치로 제한합니다. 예를 들어 S3 경로에와 같은 날짜 기반 폴더가 포함되어 있다고 가정합니다folder/2021/04/01/. 이 경우 날짜를 파라미터화하고 "2021년 3월 1일부터 2021년 4월 1일까지" 또는 "지난 주"와 같은 특정 범위로 제한할 수 있습니다.

파라미터를 사용하여 경로를 정의하려면 파라미터를 정의하고 다음 형식을 사용하여 경로에 추가합니다.

```
s3://my-databrew-bucket/some-folder/{parameter1}/file-{parameter2}.json
```

> **Note**
>
> S3 경로의 정규식과 마찬가지로 버킷 이름 뒤에 오는 부분인 경로의 키 부분에만 파라미터를 사용할 수 있습니다.

파라미터 정의에는 이름 및 유형이라는 두 개의 필드가 필요합니다. 유형은 문자열, 숫자 또는 날짜일 수 있습니다. DataBrew가 날짜 값을 올바르게 해석하고 비교할 수 있도록 날짜 유형의 파라미터에는 날짜 형식의 정의가 있어야 합니다. 선택적으로 파라미터에 대한 일치 조건을 정의할 수 있습니다. DataBrew 작업 또는 대화형 세션에 의해 로드될 때 파라미터의 일치하는 값을 데이터 세트에 열로 추가하도록 선택할 수도 있습니다.

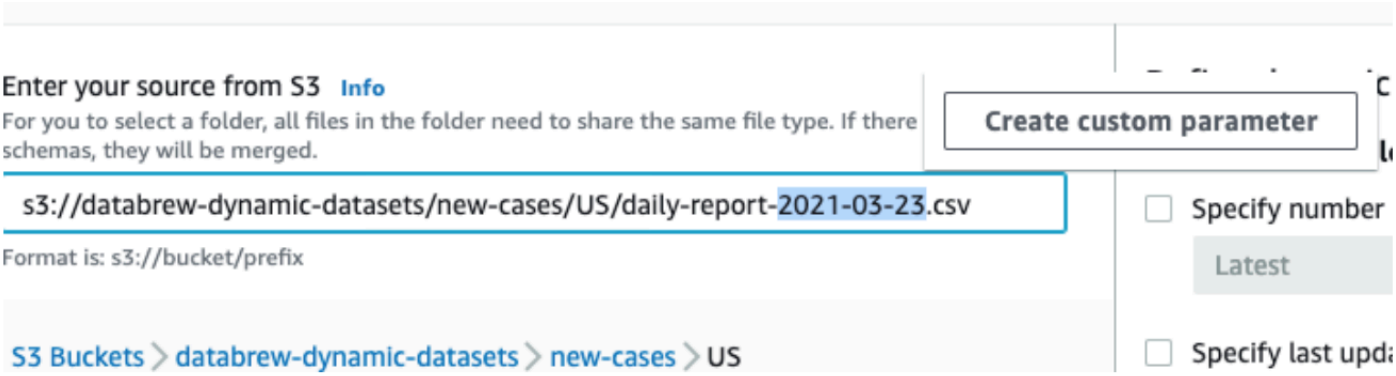
예제

DataBrew 콘솔에서 파라미터를 사용하여 동적 데이터 세트를 정의하는 예를 살펴보겠습니다. 이 예제에서는 입력 데이터가 다음과 같은 위치를 사용하여 S3 버킷에 정기적으로 기록된다고 가정합니다.

- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/UR/daily-report-2021-03-31.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-30.csv`
- `s3://databrew-dynamic-datasets/new-cases/US/daily-report-2021-03-31.csv`

여기에는 미국과 같은 국가 코드와 2021-03-30과 같은 파일 이름의 날짜라는 두 가지 동적 부분이 있습니다. 여기서는 모든 파일에 동일한 정리 레시피를 적용할 수 있습니다. 매일 정리 작업을 수행하려고 한다고 가정해 보겠습니다. 다음은이 시나리오에 대해 파라미터화된 경로를 정의하는 방법입니다.

1. 특정 파일로 이동합니다.
2. 그런 다음 날짜와 같은 다양한 부분을 선택하고 파라미터로 바꿉니다. 이 경우 날짜를 바꿉니다.



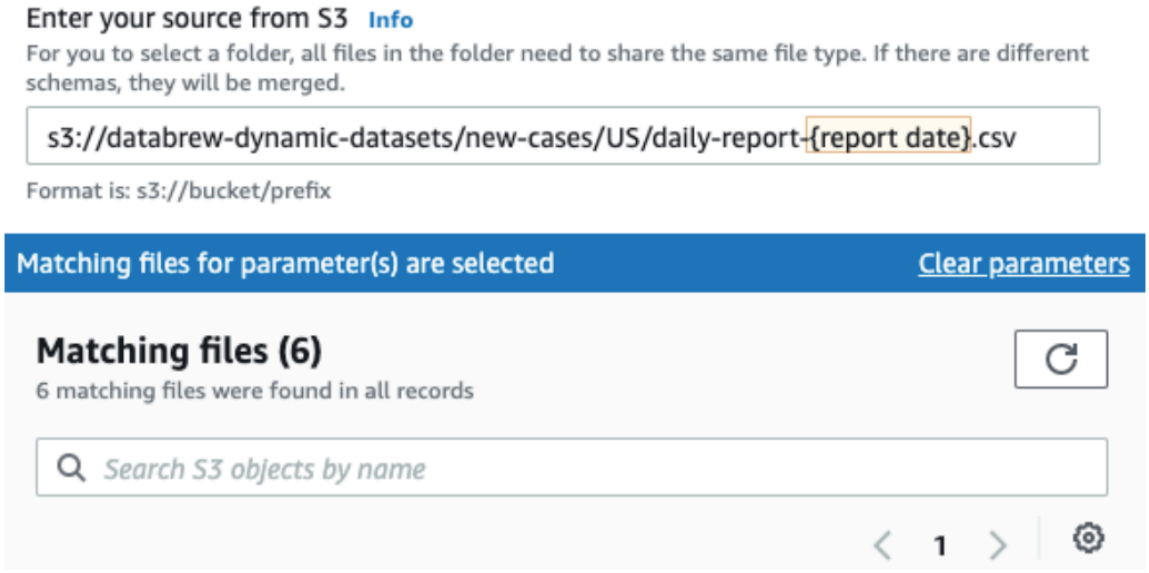
3. 사용자 지정 파라미터 생성 및 해당 파라미터에 대한 속성 설정의 컨텍스트 메뉴(마우스 오른쪽 버튼 클릭)를 엽니다.
   - 이름: 보고서 날짜
   - 유형: 날짜
   - 날짜 형식: yyyy-MM-dd(미리 정의된 형식에서 선택됨)

- 조건(시간 범위): 지난 24시간
- 열로 추가: true(확인됨)

다른 필드는 기본값으로 유지합니다.

4. 생성(Create)을 선택합니다.

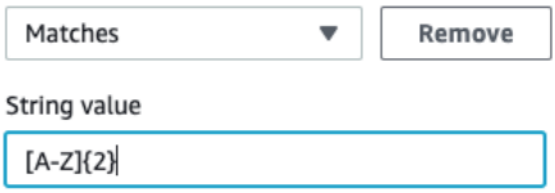
그러면 다음 스크린샷과 같이 업데이트된 경로가 표시됩니다.



이제 국가 코드에 대해 동일한 작업을 수행하고 다음과 같이 파라미터를 지정할 수 있습니다.

- 이름: 국가 코드
- 유형: 문자열
- 열로 추가: true(확인됨)

모든 값이 관련된 경우 조건을 지정할 필요가 없습니다. 예를 들어 new-cases 폴더에는 국가 코드가 있는 하위 폴더만 있으므로 조건이 필요하지 않습니다. 제외할 다른 폴더가 있는 경우 다음 조건을 사용할 수 있습니다.



이 접근 방식은 새 사례의 하위 폴더가 두 개의 라틴어 대문자를 포함하도록 제한합니다.

이 파라미터화 후에는 데이터 세트에 일치하는 파일만 있으며 데이터 세트 생성을 선택할 수 있습니다.

> **Note**
>
> 조건에서 상대 시간 범위를 사용하면 데이터 세트가 로드될 때 시간 범위가 평가됩니다. 이는 "지난 24시간"과 같은 사전 정의된 시간 범위든 "5일 전"과 같은 사용자 지정 시간 범위든 마찬가지입니다. 이 평가 접근 방식은 대화형 세션 초기화 중에 또는 작업 시작 중에 데이터 세트가 로드되는지 여부를 적용합니다.

데이터 세트 생성을 선택하면 동적 데이터 세트를 사용할 준비가 된 것입니다. 예를 들어 먼저 프로젝트를 생성하고 대화형 DataBrew 세션을 사용하여 정리 레시피를 정의할 수 있습니다. 그런 다음 매일 실행되도록 예약된 작업을 생성할 수 있습니다. 이 작업은 작업이 시작될 때 파라미터 조건을 충족하는 데이터 세트 파일에 정리 레시피를 적용할 수 있습니다.

동적 데이터 세트에 지원되는 조건

조건을 사용하여 파라미터 또는 마지막으로 수정된 날짜 속성을 사용하여 일치하는 S3 파일을 필터링할 수 있습니다.

아래에서 각 파라미터 유형에 대해 지원되는 조건 목록을 찾을 수 있습니다.

문자열 파라미터와 함께 사용되는 조건

DataBrew SDK의 이름	SDK 동의어	DataBrew 콘솔의 이름	설명
인가	당량, ==	정확히 입니다.	파라미터의 값은 조건에 제공된 값과 동일합니다.
는가 아닙니다.	not eq, !=	Is not	파라미터의 값은 조건에 제공된 값과 동일하지 않습니다.
contains		포함	파라미터의 문자열 값에는 조건에 제공된 값이 포함됩니다.

DataBrew SDK의 이름	SDK 동의어	DataBrew 콘솔의 이름	설명
에 포함되지 않음		를 포함하지 않음	파라미터의 문자열 값에 조건에 제공된 값이 포함되지 않습니다.
로 시작		다음으로 시작	파라미터의 문자열 값은 조건에 제공된 값으로 시작합니다.
를 사용하여 시작하지 않음		로 시작하지 않음	파라미터의 문자열 값은 조건에 제공된 값으로 시작되지 않습니다.
ends_with		다음으로 끝남	파라미터의 문자열 값은 조건에 제공된 값으로 끝납니다.
not ends_with		로 끝나지 않음	파라미터의 문자열 값은 조건에 제공된 값으로 끝나지 않습니다.
matches		일치 항목	파라미터의 값은 조건에 제공된 정규식과 일치합니다.
일치하지 않음		일치하지 않음	파라미터 값이 조건에 제공된 정규식과 일치하지 않습니다.

Note

문자열 파라미터의 모든 조건은 대/소문자 구분 비교를 사용합니다. S3 경로에 사용된 사례에 대해 잘 모르는 경우 로 시작하는 정규식 값과 함께 "일치" 조건을 사용할 수 있습니다`(?i)`. 이렇게 하면 대/소문자를 구분하지 않는 비교가 수행됩니다.

예를 들어 문자열 파라미터가 로 시작되길 원abc하지만 Abc 또는 도 가능하다ABC고 가정해 보겠습니다. 이 경우에서 "일치" 조건을 조건 값으로 사용할 수 (?i)^abc 있습니다.

숫자 파라미터와 함께 사용되는 조건

DataBrew SDK의 이름	SDK 동의어	DataBrew 콘솔의 이름	설명
인가	당량, ==	정확히 입니다.	파라미터의 값은 조건에 제공된 값과 동일합니다.
는가 아닙니다.	not eq, !=	Is not	파라미터의 값은 조건에 제공된 값과 동일하지 않습니다.
보다_작음	lt, <	보다 작음	파라미터의 숫자 값이 조건에 제공된 값보다 작습니다.
less_than_equal	lte, <=	작거나 같음	파라미터의 숫자 값이 조건에 제공된 값보다 작거나 같습니다.
보다 큼	gt, >	보다 큼	파라미터의 숫자 값이 조건에 제공된 값보다 큽니다.
더 큼_작음_같음	gte, >=	크거나 같음	파라미터의 숫자 값이 조건에 제공된 값보다 크거나 같습니다.

날짜 파라미터와 함께 사용되는 조건

DataBrew SDK의 이름	DataBrew 콘솔의 이름	조건 값 형식(SDK)	설명
after	시작	`2021-03-30T01:00:00Z` 또는와 같은 ISO 8601 날짜 형식 `2021-03-30T01:00-07:00`	날짜 파라미터의 값은 조건에 제공된 날짜 이후입니다.
before	종료	`2021-03-30T01:00:00Z` 또는와 같은 ISO 8601 날짜 형식 `2021-03-30T01:00-07:00`	날짜 파라미터의 값은 조건에 제공된 날짜보다 이전입니다.
상대_후	시작(상대적)	`-48h` 또는와 같은 시간 단위의 양수 또는 음수입니다`+7d`.	날짜 파라미터의 값은 조건에 제공된 상대 날짜 이후입니다. 대화형 세션이 초기화되거나 연결된 작업이 시작될 때 데이터 세트가 로드될 때 상대 날짜가 평가됩니다. 이 순간이 예제에서 "지금"이라고 합니다.
relative_before	종료(상대적)	`-48h` 또는와 같은 시간 단위의 양수 또는 음수입니다`+7d`.	날짜 파라미터의 값은 조건에 제공된 상대 날짜보다 이전입니다. 대화형 세션이 초기화되거나 연결된 작업이 시작될 때 데이터 세트가 로드될 때 상대 날짜가 평가됩니다. 이

DataBrew SDK의 이름	DataBrew 콘솔의 이름	조건 값 형식(SDK)	설명
			순간이 예제에서 "지금"이라고 합니다.

SDK를 사용하는 경우 상대 날짜를 형식으로 입력합니다`±{number_of_time_units}{time_unit}`. 다음 시간 단위를 사용할 수 있습니다.

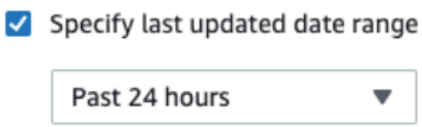
- -1h(1시간 전)
- +2d(지금부터 2일 후)
- -120m(120분 전)
- 5,000초(지금부터 5,000초)
- -3w(3주 전)
- +4M(지금부터 4개월 후)
- -1년(1년 전)

대화형 세션이 초기화되거나 연결된 작업이 시작될 때 데이터 세트가 로드될 때 상대 날짜가 평가됩니다. 이는 앞의 예제에서 "지금"이라고 하는 순간입니다.

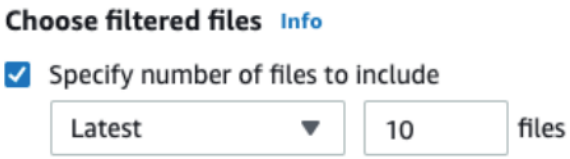
동적 데이터 세트에 대한 설정 구성

파라미터화된 S3 경로를 제공하는 것 외에도 여러 파일이 있는 데이터 세트에 대한 다른 설정을 구성할 수 있습니다. 이러한 설정은 마지막 수정 날짜별로 S3 파일을 필터링하고 파일 수를 제한합니다.

경로에서 날짜 파라미터를 설정하는 것과 마찬가지로 일치하는 파일이 업데이트된 시간 범위를 정의하고 해당 파일만 데이터 세트에 포함할 수 있습니다. "2021년 3월 30일"과 같은 절대 날짜 또는 "지난 24시간"과 같은 상대 범위를 사용하여 이러한 범위를 정의할 수 있습니다.



일치하는 파일 수를 제한하려면 0보다 큰 파일 수와 일치하는 최신 파일을 원하는지 아니면 가장 오래된 파일을 원하는지 선택합니다.



데이터 타입

데이터 세트의 각 열에 대한 데이터는 다음 데이터 유형 중 하나로 변환됩니다.

- 바이트 - 1바이트 부호 있는 정수입니다. 숫자 범위는 -128~127입니다.
- short - 2바이트 부호 있는 정수입니다. 숫자 범위는 -32768~32767입니다.
- 정수 - 부호 있는 4바이트 정수입니다. 숫자 범위는 -2147483648~2147483647입니다.
- long – 8바이트 부호 있는 정수입니다. 숫자 범위는 -9223372036854775808~9223372036854775807입니다.
- 부동 소수점 - 4바이트 단일 정밀도 부동 소수점 숫자입니다.
- double - 8바이트 배정밀도 부동 소수점 숫자입니다.
- 10진수 - 최대 총 38자리 및 소수점 뒤 18자리의 부호 있는 10진수입니다.
- string - 문자열 값입니다.
- 부울 - 부울 유형에는 `true` 및 `false` 또는 `yes` 및 `no`라는 두 가지 값 중 하나가 있습니다.
- timestamp - 연도, 월, 일, 시간, 분 및 초 필드를 구성하는 값입니다.
- 날짜 - 연도, 월, 일 필드로 구성된 값입니다.

고급 데이터 형식

고급 데이터 형식은 DataBrew가 프로젝트의 문자열 열 내에서 감지하는 데이터 형식이므로 데이터 세트의 일부가 아닙니다. 고급 데이터 형식에 대한 자세한 내용은 고급 데이터 형식을 참조하세요.

고급 데이터 형식

고급 데이터 형식은 DataBrew가 패턴 일치를 통해 프로젝트의 문자열 열 내에서 감지하는 데이터 형식입니다. 문자열 열을 클릭하면 열에 있는 값의 50% 이상이 해당 데이터 형식의 기준을 충족하는 경우 열에 해당 고급 데이터 형식으로 플래그가 지정됩니다.

DataBrew가 감지할 수 있는 데이터 형식은 다음과 같습니다.

- 날짜/타임스탬프
- SSN
- 전화번호
- 이메일
- 신용 카드
- Gender
- IP 주소
- URL
- 우편번호
- 국가
- 통화
- State
- 구/군/시

다음 변환을 사용하여 고급 데이터 유형으로 작업할 수 있습니다.

- GET_ADVANCED_DATATYPE: 문자열 열이 주어지면는 열의 고급 데이터 형식을 식별합니다.
- EXTRACT_ADVANCED_DATATYPE_DETAILS: 고급 데이터 유형에 대한 세부 정보를 추출합니다.
- AdvancedD_DATATYPE_FILTER: 고급 데이터 유형 감지를 기반으로 현재 소스 열을 필터링합니다.
- ADVANCED_DATATYPE_FLAG: 현재 소스 열의 값을 기반으로 새 플래그 열을 생성합니다.

에서 데이터 품질 검증AWS Glue DataBrew

데이터 세트의 품질을 보장하기 위해 규칙 세트에서 데이터 품질 규칙 목록을 정의할 수 있습니다. 규칙 세트는 다양한 데이터 지표를 예상 값과 비교하는 규칙 세트입니다. 규칙의 기준 중 하나라도 충족되지 않으면 규칙 세트는 전체적으로 검증에 실패합니다. 그런 다음 각 규칙에 대한 개별 결과를 검사할 수 있습니다. 검증 실패를 유발하는 모든 규칙의 경우 필요한 수정을 수행하고 다시 검증할 수 있습니다.

규칙의 예는 다음과 같습니다.

- 열의 값은 0에서 100 사이"APY"입니다.
- 열의 누락된 값 수가 5%를 초과하지 group_name 않음

개별 열에 대해 각 규칙을 정의하거나 선택한 여러 열에 독립적으로 적용할 수 있습니다. 예를 들면 다음과 같습니다.

- 열 , "rate",의 최대값은 100을 초과하지 않습니다"pay""increase".

규칙은 여러 개의 간단한 검사로 구성될 수 있습니다. 예를 들어 모두 true인지 아니면 모두 true인지 정의할 수 있습니다.

- 열의 값은 로 시작해야 "ProductId" 하며 열의 값 "asin-" 길이는 32"ProductId"입니다.

비교되는 값이 하나뿐number of duplicate values인 max, 또는 min와 같은 집계 값 또는 열의 각 행에 집계되지 않은 값에 대한 규칙을 확인할 수 있습니다. 후자의 경우와 같은 "통과" 임계값을 정의할 수도 있습니다value in columnA > value in columnB for at least 95% of rows.

프로파일 정보와 마찬가지로 문자열 및 숫자와 같은 단순 유형의 열에 대해서만 열 수준 데이터 품질 규칙을 정의할 수 있습니다. 배열 또는 구조와 같은 복잡한 유형의 열에는 데이터 품질 규칙을 정의할 수 없습니다. 프로필 정보 작업에 대한 자세한 내용은 섹션을 참조하세요AWS Glue DataBrew프로필 작업 생성 및 작업.

데이터 품질 규칙 검증

규칙 세트가 정의되면 검증을 위해 프로파일 작업에 추가할 수 있습니다. 데이터 세트에 대해 둘 이상의 규칙 세트를 정의할 수 있습니다.

예를 들어 규칙 세트 하나에 최소 허용 기준의 규칙이 포함될 수 있습니다. 해당 규칙 세트에 대한 검증 실패는 데이터가 더 이상 사용할 수 없음을 의미할 수 있습니다. 기계 학습 훈련에 사용되는 데이터 세트의 키 열에 값이 누락된 경우를 예로 들 수 있습니다. 더 엄격한 규칙이 포함된 두 번째 규칙 세트를 사용하여 데이터 세트의 품질이 양호하여 정리가 필요하지 않은지 확인할 수 있습니다.

프로파일 작업 구성에서 지정된 데이터 세트에 정의된 규칙 세트를 하나 이상 적용할 수 있습니다. 프로파일 작업이 실행되면 데이터 프로파일 외에도 검증 보고서가 생성됩니다. 검증 보고서는 프로필 데이터와 동일한 위치에서 사용할 수 있습니다. 프로필 정보와 마찬가지로 DataBrew 콘솔에서 결과를 탐색할 수 있습니다. 데이터 세트 세부 정보 보기에서 데이터 품질 탭을 선택하여 결과를 봅니다. 프로필 정보 작업에 대한 자세한 내용은 섹션을 참조하세요AWS Glue DataBrew프로필 작업 생성 및 작업.

검증 결과에 대한 조치

DataBrew 프로파일 작업이 완료되면 DataBrew는 해당 작업 실행의 세부 정보가 포함된 Amazon CloudWatch 이벤트를 보냅니다. 또한 데이터 품질 규칙을 검증하도록 작업을 구성한 경우 DataBrew는 검증된 각 규칙 세트에 대한 이벤트를 전송합니다. 이벤트에는 결과(`SUCCEEDED`, `FAILED`또는 `ERROR`)와 세부 데이터 품질 검증 보고서에 대한 링크가 포함됩니다. 그런 다음 검증 상태에 따라 다음 작업을 호출하여 추가 작업을 자동화할 수 있습니다. Amazon SNS 알림,AWS Lambda함수 호출 등과 같은 대상 작업에 이벤트를 연결하는 방법에 대한 자세한 내용은 Amazon EventBridge 시작하기를 참조하세요.

다음은 DataBrew 검증 결과 이벤트의 예입니다.

```
{
  "version": "0",
  "id": "fb27348b-112d-e7c2-560d-85e7c2c09964",
  "detail-type": "DataBrew Ruleset Validation Result",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2021-11-18T13:15:46Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "datasetName": "MyDataset",
    "jobName": "MyProfileJob",
    "jobRunId": "db_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e",
    "rulesetName": "MyRuleset",
    "validationState": "FAILED",
    "validationReportLocation": "s3://MyBucket/MyKey/
MyDataset_f07954d20d083de0c1fc1eee11498d8635ee5be4ca416af27d33933e91ff4e6e_dq-
validation-report.json"
```

```
  }
}
```

와 같은 이벤트의 속성`detail-type`source과 `detail` 속성의 중첩 속성을 사용하여 Amazon Eventbridge에서 이벤트 패턴을 생성할 수 있습니다. 예를 들어 DataBrew 작업에서 실패한 모든 검증과 일치하는 이벤트 패턴은 다음과 같습니다.

```
{
  "source": ["aws.databrew"],
  "detail-type": ["DataBrew Ruleset Validation Result"],
  "detail": {
   "validationState": ["FAILED"]
  }
}
```

규칙 세트를 생성하고 규칙을 검증하는 예제는 섹션을 참조하세요데이터 품질 규칙을 사용하여 규칙 세트 생성. DataBrew에서 CloudWatch 이벤트 작업에 대한 자세한 내용은 섹션을 참조하세요. CloudWatch Events를 사용하여 DataBrew 자동화

데이터 품질 규칙을 사용하여 규칙 세트 생성

다음 절차에서는 규칙 세트를 생성하고 데이터 세트에 적용하는 예를 찾을 수 있습니다. 규칙 세트는 다양한 데이터 지표를 예상 값과 비교하는 규칙 세트입니다. 그런 다음 프로파일 작업에서이 규칙 세트를 사용하여 포함된 데이터 품질 규칙을 검증할 수 있습니다.

데이터 품질 규칙을 사용하여 예제 규칙 세트를 생성하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 엽니다.
2. 탐색 창에서 DQ 규칙을 선택한 다음 데이터 품질 규칙 세트 생성을 선택합니다.
3. 규칙 세트의 이름을 입력합니다. 선택적으로 규칙 세트에 대한 설명을 입력합니다.
4. 연결된 데이터 세트에서 규칙 세트와 연결할 데이터 세트를 선택합니다.

   데이터 세트를 선택한 후 오른쪽에서 데이터 세트 미리 보기 창을 볼 수 있습니다.
5. 생성할 데이터 품질 규칙을 결정할 때 데이터 세트 미리 보기 창의 미리 보기를 사용하여 데이터 세트의 값과 스키마를 탐색합니다. 미리 보기는 데이터에 발생할 수 있는 잠재적 문제에 대한 인사이트를 제공합니다.

데이터베이스와 같은 일부 데이터 소스는 데이터 미리 보기를 지원하지 않습니다. 이 경우 먼저 데이터 품질 규칙을 검증하지 않고 프로파일 작업을 실행할 수 있습니다. 그런 다음 데이터 프로파일을 사용하여 데이터 스키마 및 값 분포에 대한 정보를 가져올 수 있습니다.

6. 규칙 세트를 생성할 때 사용할 수 있는 몇 가지 규칙 제안이 나열된 권장 사항 탭을 확인합니다. 권장 사항의 전체, 일부 또는 없음을 선택할 수 있습니다.

   관련 권장 사항을 선택한 후 규칙 세트에 추가를 선택합니다.

   그러면 규칙 세트에 규칙이 추가됩니다. 필요한 경우 파라미터를 검사하고 수정합니다. 문자열, 숫자 및 부울과 같은 간단한 유형의 열만 데이터 품질 규칙에 사용할 수 있습니다.

7. 다른 규칙 추가를 선택하여 권장 사항이 적용되지 않는 규칙을 추가합니다. 나중에 검증 결과를 더 쉽게 해석할 수 있도록 규칙 이름을 변경할 수 있습니다.

8. 데이터 품질 검사 범위를 사용하여이 규칙의 각 검사에 따라 개별 열을 선택할지 또는 선택한 열 그룹에 적용해야 하는지 여부를 선택합니다. 예를 들어 데이터 세트에 0에서 100 사이의 값을 가져야 하는 숫자 열이 여러 개 있는 경우 규칙을 한 번 정의하고이 규칙에서 확인할 모든 열을 선택할 수 있습니다.

9. 규칙에 검사가 두 개 이상 있는 경우 규칙 성공 기준 드롭다운에서 모든 검사를 충족해야 하는지 또는 기준을 충족하는 검사를 선택합니다.

10. 데이터 품질 검사 드롭다운에서이 규칙을 확인하기 위해 수행할 검사를 선택합니다. 사용 가능한 검사에 대한 자세한 내용은 섹션을 참조하세요사용 가능한 검사.

11. 데이터 품질 검사 범위의 각 열에 대해 개별 검사를 선택한 경우 열을 선택합니다. 이 검사의 열 이름을 선택하거나 입력합니다.

12. 검사에 따라 파라미터를 선택합니다. 일부 조건은 제공된 사용자 지정 값만 허용하고 일부는 다른 열에 대한 참조도 지원합니다.

13. 문자열 값에 대한 조건 포함과 같은 열 값 검사를 선택한 경우 “통과” 임계값을 지정할 수 있습니다. 예를 들어 값의 95% 이상이 조건을 충족하도록 하려면 같음 이상을 임계값의 조건으로 선택하고 임계값으로 95를 입력한 다음 임계값 섹션의 다음 드롭다운에 "%(%) 행"을 그대로 두어야 합니다. 또는 값이 누락된 조건이 true인 행이 10개를 넘지 않도록 하려면 조건으로 같음 미만을 선택하고 임계값에 10을 입력하고 다음 드롭다운에서 행을 선택합니다. 검증 중에 크기가 다른 샘플을 사용하는 경우 결과가 다를 수 있습니다.

14. 필요한 경우 규칙을 더 추가합니다.

15. 규칙 세트 생성을 선택합니다.

규칙 세트를 사용하여 프로파일 작업 생성

앞서 설명한 대로 규칙 세트를 생성하면 계정의 모든 규칙 세트를 표시하는 데이터 품질 규칙 페이지로 이동합니다.

규칙 세트를 포함한 프로파일 작업을 생성하려면

1. 세부 정보를 보려면 이전에 생성한 규칙 세트의 이름을 선택합니다.
2. 규칙 세트를 사용하여 프로필 작업 생성을 선택합니다.

   작업 이름은 자동으로 채워지지만 필요에 따라 변경할 수 있습니다.
3. 작업 실행 샘플의 경우 전체 데이터 세트 또는 제한된 수의 행을 실행하도록 선택할 수 있습니다.

   제한된 샘플 크기를 실행하기로 선택한 경우 특정 규칙의 경우 전체 데이터 세트와 결과가 다를 수 있습니다.
4. 작업 출력 설정에서 작업 출력의 S3 위치를 선택합니다. 명명된 Amazon S3 버킷에서 액세스할 수 있는 폴더를 선택합니다. 존재하지 않는이 버킷의 폴더 이름을 입력하면이 폴더가 생성됩니다.

   프로필 작업이 성공적으로 완료되면이 폴더에는 JSON 형식의 데이터 및 데이터 품질 규칙 검증 보고서의 프로필이 포함됩니다.
5. 데이터 품질 규칙에서 규칙 세트는 데이터 품질 규칙 세트 이름 아래에 나열됩니다.
6. 권한에서 역할을 선택하거나 생성하여 DataBrew에 입력 Amazon S3 위치에서 읽고 작업 출력 위치에 쓸 수 있는 액세스 권한을 부여합니다. 역할이 준비되지 않은 경우 새 IAM 역할 생성을 선택합니다.
7. 필요한 AWS Glue DataBrew프로필 작업 생성 및 작업경우에 설명된 대로 기타 선택적 설정을 수정합니다.
8. 작업 생성 및 실행을 선택합니다.

검증 결과 검사 및 데이터 품질 규칙 업데이트

프로필 작업이 완료되면 데이터 품질 규칙에 대한 검증 결과를 보고 필요에 따라 규칙을 업데이트할 수 있습니다.

데이터 품질 규칙에 대한 검증 데이터를 보려면

1. DataBrew 콘솔에서 데이터 프로필 보기를 선택합니다. 이렇게 하면 데이터 세트의 데이터 프로파일 개요 탭이 표시됩니다.

2. 데이터 품질 규칙 탭을 선택합니다. 이 탭에서 모든 데이터 품질 규칙에 대한 결과를 볼 수 있습니다.
3. 해당 규칙에 대한 자세한 내용을 보려면 개별 규칙을 선택합니다.

검증에 실패한 규칙의 경우 필요한 수정을 수행할 수 있습니다.

데이터 품질 규칙을 업데이트하려면

1. 탐색 창에서 DQ 규칙을 선택합니다.
2. 데이터 품질 규칙 세트 이름에서 편집하려는 규칙이 포함된 데이터 세트를 선택합니다.
3. 변경할 규칙을 선택한 다음 편집을 선택합니다.
4. 필요한 수정을 수행한 다음 규칙 세트 업데이트를 선택합니다.
5. 작업을 다시 실행합니다. 모든 검증이 통과할 때까지이 프로세스를 반복합니다.

사용 가능한 검사

다음 표에는 규칙에 사용할 수 있는 모든 사용 가능한 조건에 대한 참조가 나열되어 있습니다. 집계된 조건은 동일한 규칙에서 집계되지 않은 조건과 결합할 수 없습니다.

> **Note**
>
> SDK 사용자의 경우 여러 열에 동일한 규칙을 적용하려면 규칙의 ColumnSelectors 속성을 사용하고 이름 또는 정규식을 사용하여 검증된 열을 지정합니다. 이 경우 암시적 CheckExpression을 사용해야 합니다. 예를 들어, 선택한 각 열의 값을 제공된 값과 “`> :val`” 비교합니다. DataBrew는 동적 데이터 세트에서 FilterExpression을 정의하는 데 암시적 구문을 사용합니다. 각 검사에 대해 개별적으로 열(들)을 지정하려면 ColumnSelectors 속성을 설정하지 마십시오. 대신 명시적 표현식을 제공합니다. 규칙의 “`:col > :val`” CheckExpression을 예로 들 수 있습니다.

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
집계 데이터 세트 조건	행 수		사용자 지정 값과 숫자 비교	`"CheckExpression": "AGG(ROWS`

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
				_COUNT) > :val", "SubstitutionMap": {":val", "10000"}
	열 수		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(COLUMNS_COUNT) == :val", "SubstitutionMap": {":val", "20"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	중복된 행		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(DUPLICATE_ROWS_COUNT) < :val", "SubstitutionMap": {":val", "100"} 또는 "CheckExpression": "AGG(DUPLICATE_ROWS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
열 통계 조건 집계	누락된 값		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(MISSING_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"} 또는 "CheckExpression": "AGG(MISSING_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	중복 값		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(DUPLICATE_VALUES_COUNT) < :val", "SubstitutionMap": {":val", "100"} 또는 "CheckExpression": "AGG(DUPLICATE_VALUES_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	유효값		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(VALID_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "10000"} 또는 "CheckExpression": "AGG(VALID_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "95"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	고유 값		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(DISTINCT_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "1000"} 또는 "CheckExpression": "AGG(DISTINCT_VALUES_PERCENTAGE) >= :val", "SubstitutionMap": {":val", "50"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	고유 값		사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(UNIQUE_VALUES_COUNT) > :val", "SubstitutionMap": {":val", "100"} 또는 "CheckExpression": "AGG(UNIQUE_VALUES_PERCENTAGE) > :val", "SubstitutionMap": {":val", "20"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	이상치	Z-점수 임계값	사용자 지정 값과 숫자 비교	"CheckExpression": "AGG(Z_SCORE_OUTLIERS_COUNT, :zscore_dev) < :val", "SubstitutionMap": {":zscore_dev": "4", ":val", "100"} 또는 "CheckExpression": "AGG(Z_SCORE_OUTLIERS_PERCENTAGE) < :val", "SubstitutionMap": {":val", "5"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제

	값 분포 통계	통계 이름(다음 표 참조)	사용자 지정 값과 숫자 비교	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				또는
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				가능한 STAT_NAME 값은 다음 표를 참조하세요.

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제

	숫자 통계	통계 이름(다음 표 참조)	사용자 지정 값과 숫자 비교	`"CheckExpression": "AGG(<STAT_NAME>) < :val", "SubstitutionMap": {":val", "100"}`
				
				또는
				
				`"CheckExpression": "AGG(<STAT_NAME>, :param) < :val", "SubstitutionMap": {":param": "0.25", :val", "5"}`
				
				Note
				가능한 STAT_NAME 값은 다음 표를 참조하세요.

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
집계되지 않음(임계값 허용)	값은 정확히 입니다.		값 목록과의 정확한 비교	`"CheckExpression": ":col IN :list", "SubstitutionMap": {":col": "`size`", ":list": "[\"S\",\"M\",\"L\",\"XL\"]"}`
	값이 정확하지 않음		값이 목록의 값과 정확히 일치하지 않아야 합니다.	`"CheckExpression": ":col NOT IN :list", "SubstitutionMap": {":col": "`domain`", ":list": "[\"GOV\",\"ORG\"]"}`

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제

	문자열 값		사용자 지정 값 또는 기타 문자열 열과 문자열 비교	`"CheckExpression":":col STARTS_WITH :val", "SubstitutionMap":{":col":"`url`", ":val":"http"}`
				
				또는
				
				`"CheckExpression":":col1 contains :col2", "SubstitutionMap":{":col1":"`url`", ":col2":"`company_name`"}`

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제
	숫자 값		사용자 지정 값 또는 기타 숫자 열과 숫자 비교	"CheckExpression": ":col IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`APY`", ":val1": "0", ":val2": "10"} 또는 "CheckExpression": ":col1 <= :col2", "SubstitutionMap": {":col1": "`bank_rate`", ":col2": "`fed_rate`"}

조건 유형	데이터 품질 검사	추가 파라미터	비교 유형	SDK 구문 예제

	값 문자열 길이		사용자 지정 값 또는 기타 숫자 열과 숫자 비교	`"CheckExpression": "length(:col) IS_BETWEEN :val1 and :val2", "SubstitutionMap": {":col": "`identifier`", ":val1": "8", ":val2": "12"}`
				
				또는
				
				`"CheckExpression": "length(:col1) <= :col2", "SubstitutionMap": {":col1": "`name`", ":col2": "`max_name_len`"}`

숫자 비교

DataBrew는 숫자 비교를 위해 다음 작업을 지원합니다. Is equals (==), Is not equals (!=), Less than (<), Less than equals (<=), Greater than (>), Greater than equals (>=) 및 Is between (is_between :val1 and :val2).

문자열 비교

다음 문자열 비교가 지원됩니다. 로 시작, 로 시작하지 않음, 로 종료, 로 종료하지 않음, 포함, 포함하지 않음, 같음, 같지 않음, 일치, 일치하지 않음.

다음 표에는 값 분포 통계 및 숫자 통계에 사용할 수 있는 사용 가능한 통계가 표시됩니다.

데이터 품질 검사	통계 이름	추가 파라미터	SDK 구문
값 분포 통계	최소		`"CheckExpression": "AGG(MAX) < :val", "SubstitutionMap": {":val", "100"}`
	최대		`"CheckExpression": "AGG(MIN) > :val", "SubstitutionMap": {":val", "0"}`
	Median		`"CheckExpression": "AGG(MEDIAN) >= :val", "SubstitutionMap": {":val", "50"}`
	Mean		`"CheckExpression":`

데이터 품질 검사	통계 이름	추가 파라미터	SDK 구문
			"AGG(MEAN) <= :val", "SubstitutionMap": {":val", "10"}
	Mode		"CheckExpression": "AGG(MODE) > :val", "SubstitutionMap": {":val", "0"}
	표준 편차		"CheckExpression": "AGG(STANDARD_DEVIATION) > :val", "SubstitutionMap": {":val", "0"}
	Entropy		"CheckExpression": "AGG(ENTROPY) > :val", "SubstitutionMap": {":val", "0"}

데이터 품질 검사	통계 이름	추가 파라미터	SDK 구문
숫자 통계	Sum		`"CheckExpression": "AGG(SUM) > :val", "SubstitutionMap": {":val", "0"}`
	쿠르트증		`"CheckExpression": "AGG(KURTOSIS) > :val", "SubstitutionMap": {":val", "0"}`
	왜도		`"CheckExpression": "AGG(SKEWNESS) > :val", "SubstitutionMap": {":val", "0"}`
	분산		`"CheckExpression": "AGG(VARIANCE) > :val", "SubstitutionMap": {":val", "0"}`

데이터 품질 검사	통계 이름	추가 파라미터	SDK 구문
	절대 편차		`"CheckExpression": "AGG(MEDIAN_ABSOLUTE_DEVIATION) > :val", "SubstitutionMap": {":val", "0"}`
	분위수	분위수: '0.25', '0.5', '0.75' 중 하나	`"CheckExpression": "AGG(QUANTILE, :pct) > :val", "SubstitutionMap": {":pct": "0.25", ":val", "0"}`

AWS Glue DataBrew프로젝트 생성 및 사용

에서AWS Glue DataBrew프로젝트는 데이터 분석 및 변환 작업의 핵심입니다.

프로젝트를 생성할 때 두 가지 기본 구성 요소를 결합합니다.

- 소스 데이터에 대한 읽기 전용 액세스를 제공하는 데이터 세트입니다. 자세한 내용은 를 사용하여 데이터에 연결AWS Glue DataBrew 단원을 참조하십시오.
- 데이터 세트에 DataBrew 데이터 변환을 적용하는 레시피입니다. 자세한 내용은 AWS Glue DataBrew레시피 생성 및 사용 단원을 참조하십시오.

DataBrew 콘솔은 대화형의 직관적인 사용자 인터페이스로 프로젝트를 제공합니다. 수백 개의 데이터 변환을 실험하는 것이 좋습니다. 따라서 데이터 변환의 작동 방식과 데이터에 미치는 영향을 알아볼 수 있습니다.

프로젝트 보기에 표시되는 데이터는 데이터 세트의 샘플입니다. 데이터 세트는 수천 또는 수백만 개의 행으로 매우 클 수 있으므로 샘플을 사용하면 다양한 방식으로 샘플 데이터를 변환하는 동안 DataBrew 콘솔이 응답성을 유지하는 데 도움이 됩니다. 기본적으로 샘플은 데이터 세트의 처음 500개 데이터 행으로 구성됩니다. 샘플 크기와 선택한 행에 대해 다른 설정을 선택할 수 있습니다.

샘플 데이터를 변환할 때 DataBrew는 지금까지 적용한 step-by-step 변환 시리즈인 프로젝트 레시피를 구축하고 구체화하는 데 도움이 됩니다. work-in-progress 레시피는 자동으로 저장되므로 언제든지 프로젝트 보기를 벗어나 나중에 돌아와서 중단한 부분을 픽업할 수 있습니다.

레시피를 사용할 준비가 되면 게시할 수 있습니다. 레시피를 게시하면 전체 데이터 세트에 레시피를 적용하거나 데이터의 구조, 콘텐츠 및 통계 특성을 이해할 수 있는 광범위한 데이터 프로파일을 생성할 수 있는 DataBrew 작업 하위 시스템에서 사용할 수 있습니다.

주제

- 프로젝트 생성
- DataBrew 프로젝트 세션 개요
- 프로젝트 삭제

프로젝트 생성

다음 절차에 따라 프로젝트를 생성합니다.

프로젝트를 생성하려면

1. 에 로그인AWS Management Console하고 DataBrew 콘솔을 엽니다.
2. 탐색 창에서 프로젝트를 선택합니다. 그런 다음 프로젝트 생성을 선택합니다.
3. 프로젝트의 이름을 입력합니다. 그런 다음 프로젝트에 연결할 레시피를 선택합니다.

   - 처음부터 시작하는 경우 새 레시피 생성을 선택합니다. 이렇게 하면 빈 새 레시피가 생성되어 프로젝트에 연결됩니다.
   - 이 프로젝트에 사용할 이전에 게시된 레시피가 있는 경우 기존 레시피 편집을 선택합니다. 레시피가 현재 다른 프로젝트에 연결되어 있거나 레시피에 대해 정의된 작업이 있는 경우 새 프로젝트에서 사용할 수 없습니다. 레시피 찾아보기를 선택하여 사용 가능한 레시피를 확인합니다.
   - 이전에 게시된 기존 레시피가 있고 해당 단계를 가져오려는 경우 레시피에서 단계 가져오기를 선택한 다음 다음을 수행합니다.
     1. 레시피 찾아보기를 선택하여 사용 가능한 레시피를 확인합니다.
     2. 사용할 레시피의 게시된 버전을 선택합니다. 레시피에는 프로젝트 보기에서 작업하는 동안 레시피를 게시한 빈도에 따라 여러 버전이 있을 수 있습니다.
     3. 레시피 단계 보기를 선택하여 레시피의 데이터 변환을 검사합니다.
4. 레시피가 있으면 데이터 세트 선택 창에서 작업할 데이터 세트를 선택합니다.

   - 내 데이터 세트 - 이전에 생성한 데이터 세트를 선택합니다. 자세한 내용은 프로젝트 생성 단원을 참조하십시오.
   - 샘플 파일 -에서 유지 관리하는 샘플 데이터를 기반으로 새 데이터 세트를 생성합니다AWS. 이 샘플 데이터는 자체 데이터를 제공하지 않고도 DataBrew가 수행할 수 있는 작업을 탐색할 수 있는 좋은 방법입니다. 데이터 세트의 이름을 입력해야 합니다.
   - 새 데이터 세트 - 새 데이터 세트를 생성합니다. 자세한 내용은 프로젝트 생성 단원을 참조하십시오.
5. 액세스 권한에서 DataBrew가 Amazon S3 입력 위치에서 읽을 수 있도록 허용하는AWS Identity and Access Management(IAM) 역할을 선택합니다.AWS계정이 소유한 S3 위치의 경우 `AwsGlueDataBrewDataAccessRole` 서비스 관리형 역할을 선택할 수 있습니다. 이렇게 하면 DataBrew가 사용자가 소유한 S3 리소스에 액세스할 수 있습니다.
6. 샘플링 창에서 DataBrew가 데이터 세트에서 데이터 샘플을 빌드하는 옵션을 찾을 수 있습니다.

   유형에서 DataBrew가 데이터 세트에서 행을 가져오는 방법을 선택합니다.

   - 첫 번째 n개의 행을 사용하여 데이터 세트의 첫 번째 행을 기반으로 샘플을 생성합니다.
   - 임의 행을 사용하여 데이터 세트의 임의 행 선택을 기반으로 샘플을 생성합니다.

- 샘플에 표시할 행 수를 선택합니다. 500, 1,000, 2,500 또는 최대 5,000개의 행까지 사용자 지정 샘플 크기를 선택합니다. 샘플 크기가 작을수록 DataBrew가 더 빠르게 변환을 수행할 수 있으므로 레시피를 개발할 때 시간을 절약할 수 있습니다. 샘플 크기가 클수록 기본 소스 데이터의 구성이 더 정확하게 반영됩니다. 그러나 프로젝트 세션 초기화 및 대화형 변환은 느립니다.

7. (선택 사항) 태그를 선택하여 데이터 세트에 태그를 연결합니다.

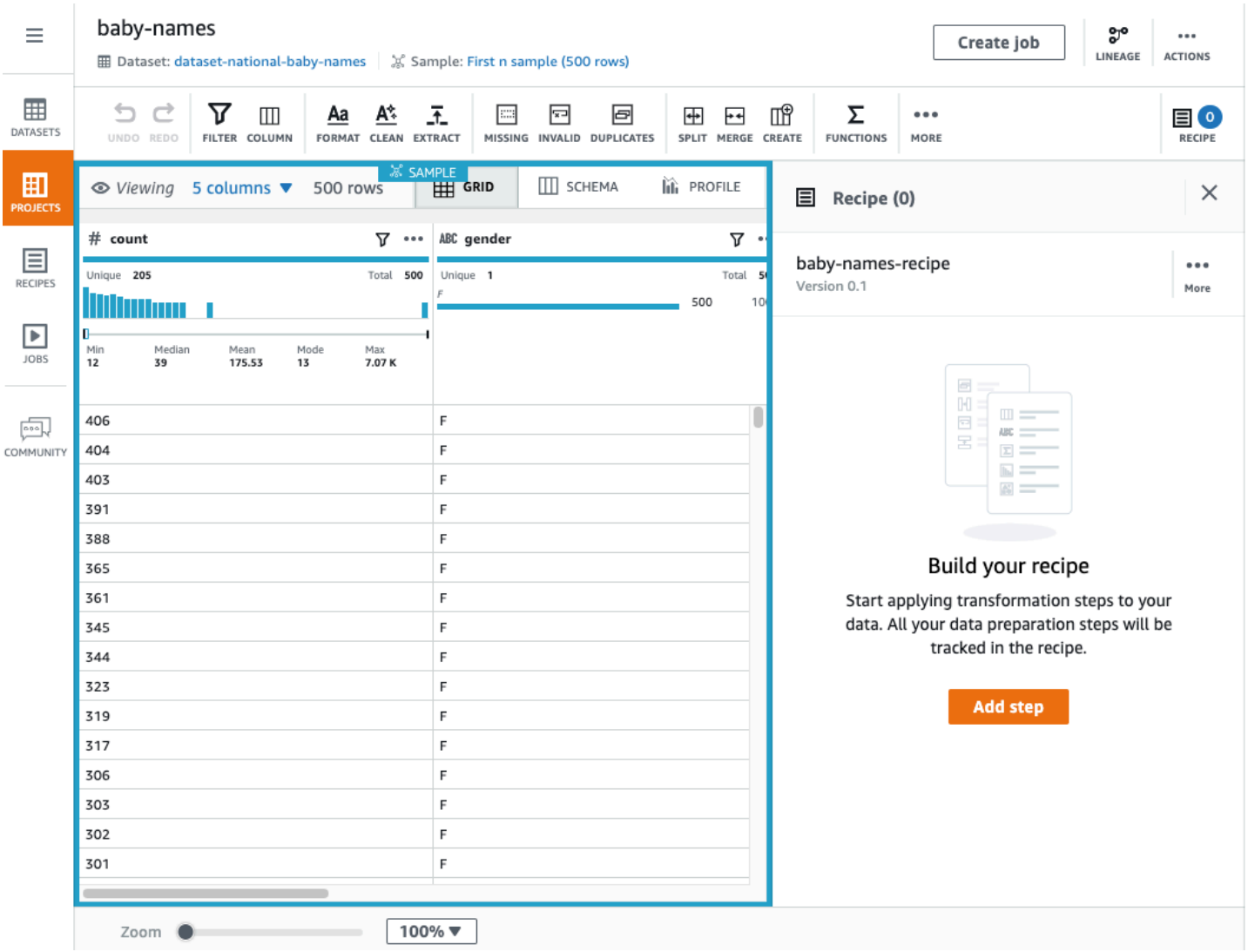
   태그는 사용자 정의 키와 선택적 값으로 구성된 간단한 레이블로, 용도, 소유자, 환경 또는 기타 기준으로 DataBrew 프로젝트를 더 쉽게 관리, 검색 및 필터링할 수 있습니다.

8. 설정이 원하는 대로 되면 작업 생성을 선택합니다.

DataBrew는 필요한 경우 새 데이터 세트를 생성하고, 필요한 경우 새 레시피를 생성하고, 데이터 샘플을 빌드하고, 대화형 프로젝트 세션을 생성합니다. 이 프로세스를 완료하는 데 몇 분 정도 걸릴 수 있습니다. 프로젝트를 사용할 준비가 되면 데이터 샘플 작업을 시작할 수 있습니다.

DataBrew 프로젝트 세션 개요

DataBrew 프로젝트 세션에서는 대화형 워크스페이스 내에서 작업합니다.



왼쪽 창에는 데이터의 현재 보기가 표시됩니다. 오른쪽 창에는 현재 비어 있는 프로젝트의 변환 레시피가 표시됩니다.

데이터 그리드의 오른쪽 상단 모서리에는 , 및 GRIDSCHEMA의 세 가지 탭이 있습니다PROFILE. 이러한 탭 중 하나를 선택하면 워크스페이스에 해당 보기가 표시됩니다. 보기는 다음에 설명됩니다.

그리드 보기

그리드 보기는 샘플이 표 형식으로 표시되는 기본 보기입니다. 그리드 보기를 간략하게 살펴보려면 다음 절차를 따르세요.

그리드 보기를 살펴보려면

1. 먼저 전체 공간을 확인합니다.

a. 모든 열을 보려면 왼쪽과 오른쪽으로 스크롤합니다.

b. 위아래로 스크롤하여 모든 데이터 값을 확인합니다.

c. 워크스페이스 하단의 확대/축소 컨트롤을 사용하여 그리드의 배율 수준을 조정합니다.

2. 오른쪽 상단에서 표시되는 샘플의 열 수와 샘플의 현재 행 수를 확인합니다.

   표시되는 열을 변경하려면 N 열 링크(여기서 N은 현재 표시된 열 수)를 선택합니다. 원하는 열을 선택하고 선택한 열 표시를 선택합니다.

3. 이제 DataBrew 변환 실험을 시작할 수 있습니다. 다음을 시도해 보세요.

   a. 변환 도구 모음에서 형식 선택, 대문자로 변경을 선택합니다.

   b. 소스 열에서 문자 데이터가 포함된 열을 선택합니다.

   c. 기타 설정은 기본값을 유지합니다.

   d. 변환된 데이터의 모양을 보려면 변경 사항 미리 보기를 선택합니다. 그런 다음이 변환을 레시피에 추가하려면 적용을 선택합니다.

   데이터 변환을 적용할 때마다 DataBrew는 이를 레시피의 작업 사본에 추가합니다. 워크스페이스 오른쪽에 표시됩니다.

4. 다음을 시도해 보세요.

   a. 변환 도구 모음에서 함수를 기반으로 생성을 선택합니다.

   b. 함수 선택에서를 선택합니다SQUARE ROOT.

   c. 소스 열에서 숫자 데이터가 포함된 열을 선택합니다.

   d. 다른 설정은 기본값으로 둡니다.

   e. 변경 사항 미리 보기를 선택하여 변환된 데이터가 어떻게 보이는지 확인합니다. 그런 다음이 변환을 레시피에 추가하려면 적용을 선택합니다.

5. RECIPE를 선택하여 오른쪽 상단의 레시피 창을 축소합니다. 레시피 창을 확장하려면 RECIPE를 다시 선택합니다.

레시피의 새 버전 게시

변환을 계속 적용하면 레시피의 단계 수가 증가합니다. 언제든지 레시피의 새 버전을 게시할 수 있습니다. 레시피를 게시하면 DataBrew의 다른 곳에서 사용할 수 있습니다. 이렇게 하면 프로젝트 데이터 샘플만 변환하는 대신 레시피 작업을 실행하여 전체 데이터 세트를 변환할 수 있습니다.

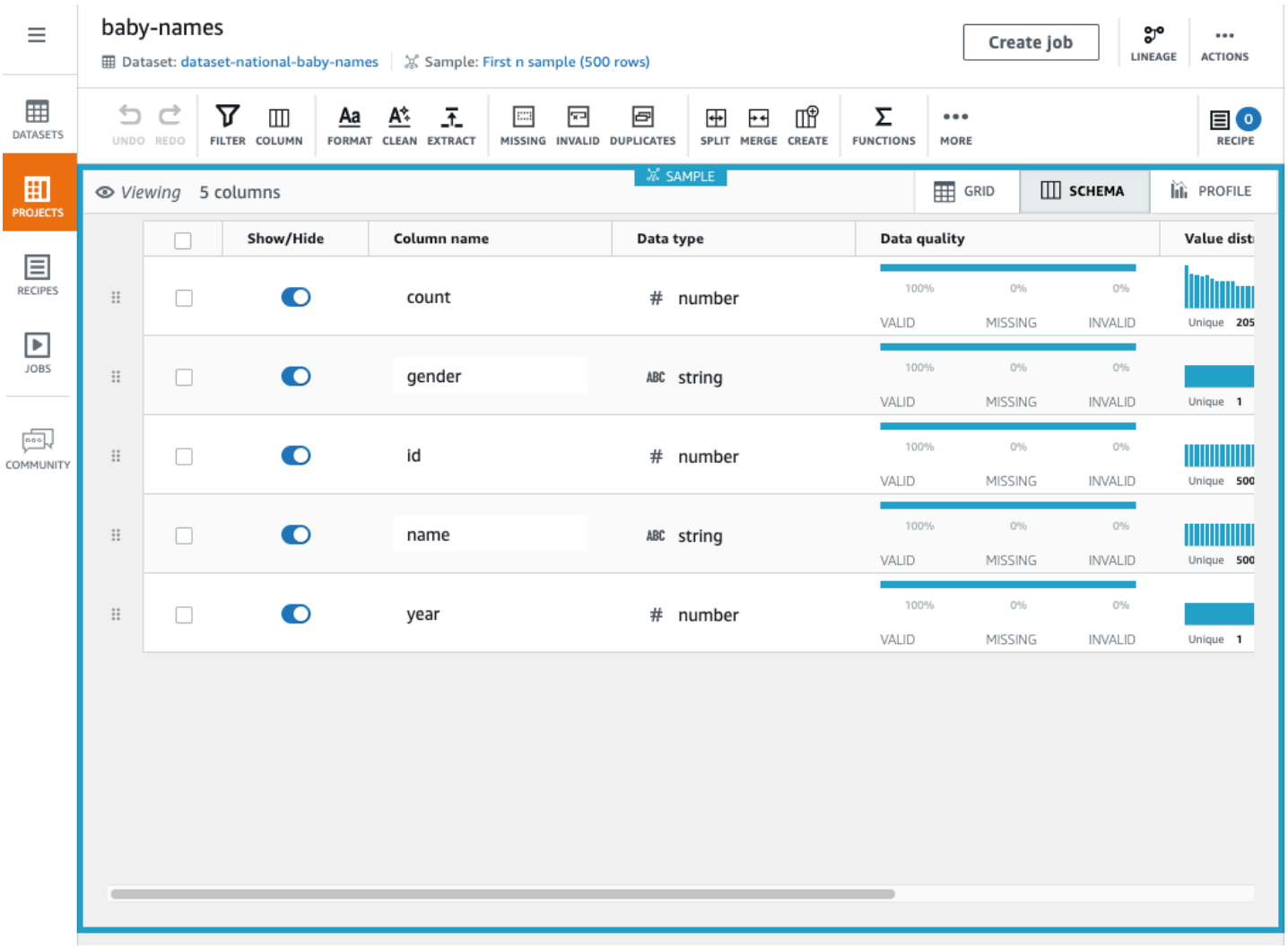
레시피를 게시하면 레시피 개발에 대한 증분적이고 반복적인 접근 방식도 장려됩니다. 즉, 레시피의 새 버전을 원하는 대로 게시할 수 있으므로 필요한 경우 "마지막으로 알려진 좋은" 레시피 버전으로 돌아갈 수 있습니다.

레시피의 새 버전을 게시하려면

- 레시피 창에서 게시를 선택합니다. 이 버전의 레시피에 대한 설명을 입력하고 게시를 선택합니다.

스키마 보기

SCHEMA 탭을 선택하면 다음 스크린샷과 같이 보기가 변경됩니다.



스키마 보기에서 각 열의 데이터 값에 대한 통계를 볼 수 있습니다.

맨 왼쪽 열의 표시/숨기기 옆에 있는 데이터 열을 선택합니다. 열 세부 정보 창이 오른쪽에 나타납니다. 이 창에는 열 값에 대한 통계 요약이 표시됩니다.

열 이름에 새 이름을 입력하여 열 이름을 바꿀 수 있습니다.

열을 끌어서 놓아 열 순서를 재정렬할 수 있습니다.

프로필 보기

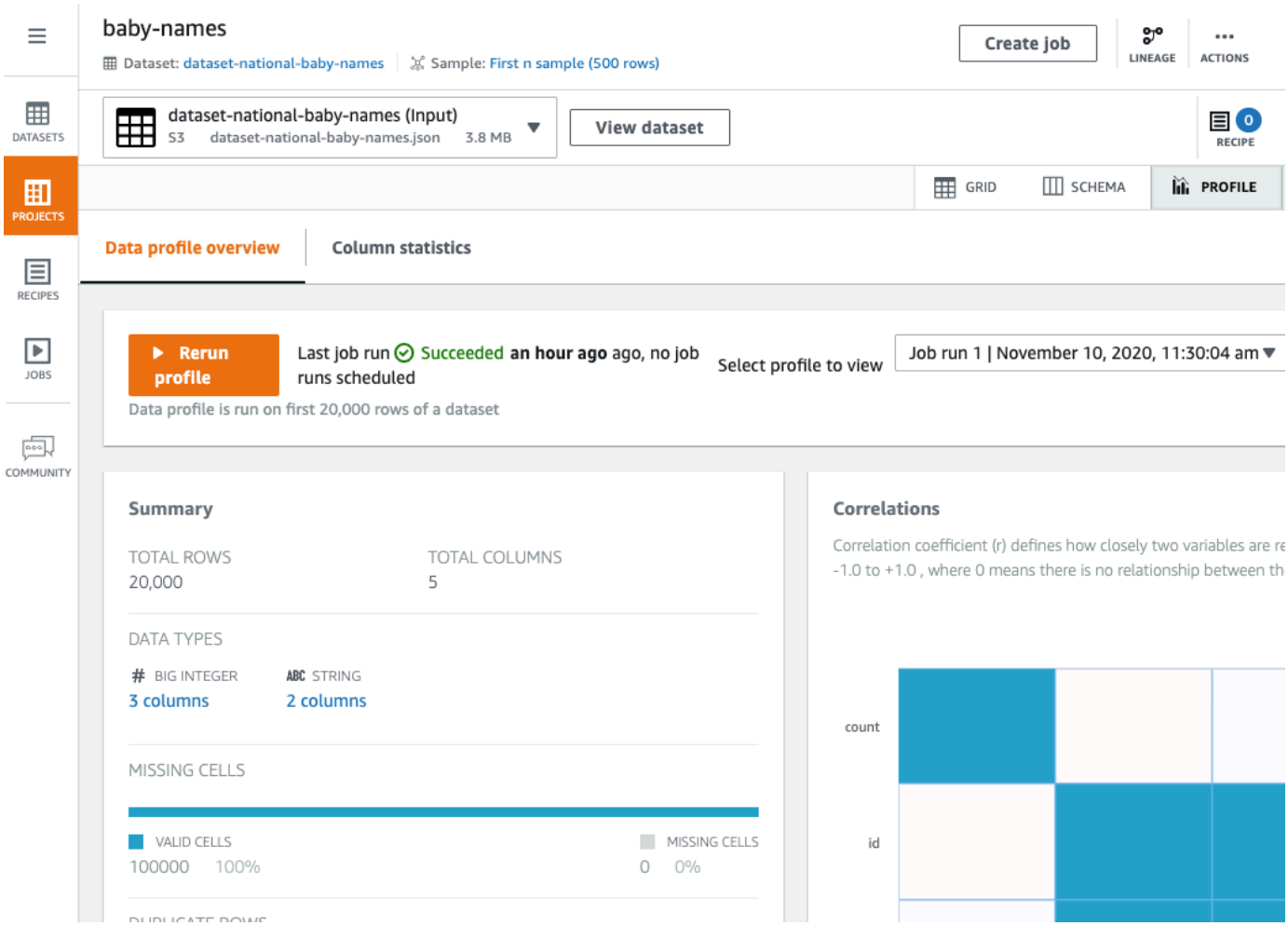
프로필 탭을 선택하면 프로젝트에 대한 자세한 볼륨 측정 정보를 볼 수 있습니다. 이렇게 하기 전에 DataBrew 작업을 실행하여 프로파일을 생성합니다.

프로필 보기 연습

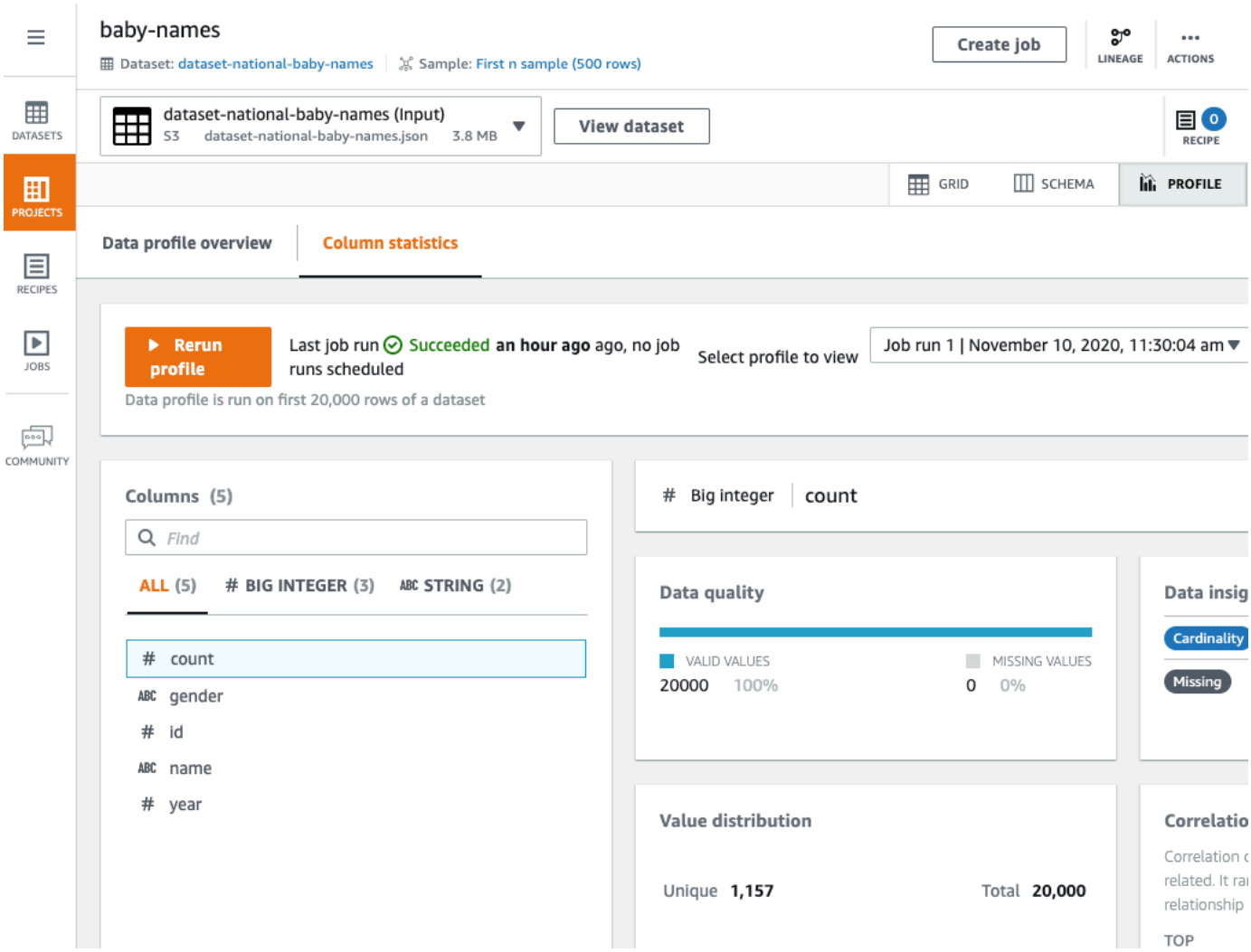
1. 작업 생성을 선택하고 작업 이름을 입력합니다.
2. 작업 출력에서 파일 유형으로 CSV를 선택합니다.
3. AWS계정에서 DataBrew의 작업 출력을 작성할 Amazon S3 버킷 및 폴더를 찾거나 생성합니다.
   - 이 Amazon S3 버킷과 폴더가 이미 있는 경우 찾아보기를 선택하고 해당 버킷과 폴더를 찾습니다. 둘 다에 대한 쓰기 권한이 있는지 확인합니다.
   - 이 Amazon S3 버킷과 폴더가 없는 경우 다음을 생성합니다.
     1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
     2. Amazon S3 버킷이 없는 경우 버킷 생성을 선택합니다. 버킷 이름에 새 버킷의 고유한 이름을 입력합니다. 버킷 생성을 선택합니다.
     3. 버킷 목록에서 사용할 버킷을 선택합니다.
     4. 폴더 생성을 선택합니다. 폴더 이름에 `databrew-output`를 입력하고 폴더 생성을 선택합니다.
4. 액세스 권한에서 DataBrew가 Amazon S3 출력 위치에 쓸 수 있도록 허용하는 IAM 역할을 선택합니다.

   AWS계정이 소유한 S3 위치의 경우 `AwsGlueDataBrewDataAccessRole` 서비스 관리형 역할을 선택할 수 있습니다. 이렇게 하면 DataBrew가 사용자가 소유한 S3 리소스에 액세스할 수 있습니다.
5. 다른 설정은 기본값으로 두고 작업 생성 및 실행을 선택합니다.
6. 작업이 완료될 때까지 실행되면 워크스페이스에 데이터 프로필의 그래픽 요약이 표시됩니다.

   데이터 프로필 개요 탭에는 다음 스크린샷과 같이 데이터 특성에 대한 상위 수준 요약이 표시됩니다.



열 통계 탭에는 데이터 값의 column-by-column 분류가 표시됩니다.



프로젝트 삭제

프로젝트가 더 이상 필요하지 않은 경우 삭제할 수 있습니다.

프로젝트 삭제

1. 탐색 창에서 프로젝트를 선택합니다.
2. 삭제할 프로젝트를 선택한 다음 작업에서 삭제를 선택합니다.

AWS Glue DataBrew레시피 생성 및 사용

DataBrew에서 레시피는 데이터 변환 단계 세트입니다. 이러한 단계를 데이터 샘플에 적용하거나 데이터 세트에 동일한 레시피를 적용할 수 있습니다.

레시피를 개발하는 가장 쉬운 방법은 데이터 샘플로 대화식으로 작업할 수 있는 DataBrew 프로젝트를 생성하는 것입니다. 자세한 내용은 섹션을 참조하세요AWS Glue DataBrew프로젝트 생성 및 사용. 프로젝트 생성 워크플로의 일부로 새 (비어 있는) 레시피가 생성되어 프로젝트에 연결됩니다. 그런 다음 데이터 변환을 추가하여 레시피 빌드를 시작할 수 있습니다.

> **Note**
>
> 단일 DataBrew 레시피에 최대 100개의 데이터 변환을 포함할 수 있습니다.

레시피 개발을 진행하면서 레시피를 게시하여 작업을 저장할 수 있습니다. DataBrew는 레시피에 대해 게시된 버전 목록을 유지합니다. 레시피 작업에서 게시된 버전을 사용하여 레시피(레시피 작업)를 실행하여 데이터 세트를 변환할 수 있습니다. 레시피 단계의 사본을 다운로드하여 다른 프로젝트 또는 다른 데이터 세트 변환에서 레시피를 재사용할 수도 있습니다.

AWS Command Line Interface(AWS CLI) 또는 SDK 중 하나를 사용하여 프로그래밍 방식으로 DataBrew 레시피를 개발할 수도 있습니다.AWS SDKs DataBrew API에서 변환을 레시피 작업이라고 합니다.

> **Note**
>
> 대화형 DataBrew 프로젝트 세션에서 적용하는 각 데이터 변환은 DataBrew API를 호출합니다. 이러한 API 호출은 behind-the-scenes 세부 정보를 알 필요 없이 자동으로 수행됩니다.

프로그래머가 아니더라도 레시피의 구조와 DataBrew가 레시피 작업을 구성하는 방법을 이해하는 것이 좋습니다.

주제

- 새 레시피 버전 게시
- 레시피 구조 정의

새 레시피 버전 게시

대화형 DataBrew 프로젝트 세션에 새 버전의 레시피를 게시합니다.

새 레시피 버전을 게시하려면

1. 레시피 창에서 게시를 선택합니다.
2. 레시피의이 버전에 대한 설명을 입력하고 게시를 선택합니다.

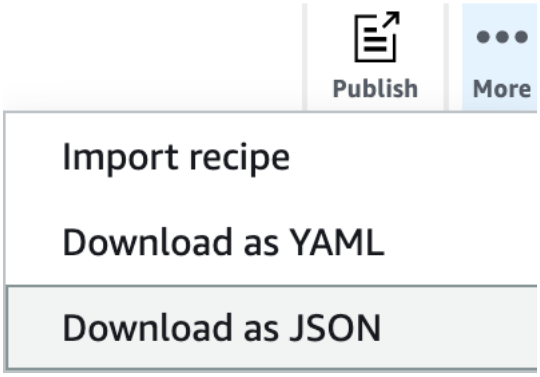
탐색 창에서 PROJECTS를 선택하여 게시된 모든 레시피와 해당 버전을 볼 수 있습니다.

레시피 구조 정의

DataBrew 콘솔을 사용하여 프로젝트를 처음 생성할 때 해당 프로젝트와 연결할 레시피를 정의합니다. 기존 레시피가 없는 경우 콘솔에서 레시피를 생성합니다.

콘솔에서 프로젝트를 작업할 때 변환 도구 모음을 사용하여 데이터 세트의 샘플 데이터에 작업을 적용합니다. 레시피를 계속 빌드하면 콘솔에 레시피 단계와 해당 단계의 순서가 표시됩니다. 단계에 만족할 때까지 레시피를 반복하고 구체화할 수 있습니다.

에서는 레시피를 시작하기AWS Glue DataBrew빌드하여 유명한 체스 게임 데이터 세트를 변환합니다. 다음 스크린샷과 같이 JSON으로 다운로드 또는 YAML로 다운로드를 선택하여 레시피 단계의 사본을 다운로드할 수 있습니다.



다운로드한 JSON 파일에는 레시피에 추가한 변환에 해당하는 레시피 작업이 포함되어 있습니다.

새 레시피에는 단계가 없습니다. 다음과 같이 새 레시피를 빈 JSON 목록으로 나타낼 수 있습니다.

```
[ ]
```

다음은에 대한 이러한 파일의 예입니다chess-project-recipe. JSON 목록에는 레시피 단계를 설명하는 여러 객체가 포함되어 있습니다. JSON 목록의 각 객체는 중괄호()로 묶입니다{ }. JSON 줄은 쉼표로 구분됩니다.

```
[
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "black_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "black_rating"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "white_rating"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "LESS_THAN",
                "Value": "1800",
                "TargetColumn": "white_rating"
            }
        ]
    },
    {
        "Action": {
            "Operation": "GROUP_BY",
            "Parameters": {
                "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"winner\",
\"targetColumnName\":\"winner_count\",\"targetColumnDataType\":\"int\",\"functionName
\":\"COUNT\"}]",
                "sourceColumns": "[\"winner\",\"victory_status\"]",
```

```
                "useNewDataFrame": "true"
            }
        }
    },
    {
        "Action": {
            "Operation": "REMOVE_VALUES",
            "Parameters": {
                "sourceColumn": "winner"
            }
        },
        "ConditionExpressions": [
            {
                "Condition": "IS",
                "Value": "[\"draw\"]",
                "TargetColumn": "winner"
            }
        ]
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "mate",
                "sourceColumn": "victory_status",
                "value": "checkmate"
            }
        }
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "resign",
                "sourceColumn": "victory_status",
                "value": "other player resigned"
            }
        }
    },
    {
        "Action": {
            "Operation": "REPLACE_TEXT",
            "Parameters": {
                "pattern": "outoftime",
```

```
                "sourceColumn": "victory_status",
                "value": "ran out of time"
            }
        }
    }
]
```

다음과 같이 새 작업에 대한 새 줄만 추가하는 경우 각 작업이 개별 줄인지 확인하는 것이 더 쉽습니다.

```
[
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "black_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "black_rating" } ] },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "white_rating" } }, "ConditionExpressions": [ { "Condition": "LESS_THAN", "Value":
 "1800", "TargetColumn": "white_rating" } ] },
 { "Action": { "Operation": "GROUP_BY", "Parameters": { "groupByAggFunctionOptions":
 "[{\"sourceColumnName\":\"winner\",\"targetColumnName\":\"winner_count\",
\"targetColumnDataType\":\"int\",\"functionName\":\"COUNT\"}]", "sourceColumns":
 "[\"winner\",\"victory_status\"]", "useNewDataFrame": "true" } } },
 { "Action": { "Operation": "REMOVE_VALUES", "Parameters": { "sourceColumn":
 "winner" } }, "ConditionExpressions": [ { "Condition": "IS", "Value": "[\"draw\"]",
 "TargetColumn": "winner" } ] },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "mate",
 "sourceColumn": "victory_status", "value": "checkmate" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "resign",
 "sourceColumn": "victory_status", "value": "other player resigned" } } },
 { "Action": { "Operation": "REPLACE_TEXT", "Parameters": { "pattern": "outoftime",
 "sourceColumn": "victory_status", "value": "ran out of time" } } }
]
```

작업은 파일에서와 동일한 순서로 순차적으로 수행됩니다.

- REMOVE_VALUES - 플레이어의 평점이 1,800 미만인 모든 게임을 필터링하려면 클래스 A 체스 플레이어에 필요한 최소 평점이 필요합니다. 이 작업에는 두 가지가 있습니다. 하나는 적어도 클래스 A 플레이어가 아닌 검은색 측의 플레이어를 제거하는 것이고 다른 하나는이 레벨에 있지 않은 흰색 측의 플레이어를 제거하는 것입니다.
- GROUP_BY - 데이터를 요약합니다. 이 경우 GROUP_BY는 winner (black 및) 값을 기준으로 행을 그룹으로 정렬합니다white. 그런 다음 각 그룹을 더 세분화하여 victory_status (, mate, resign outoftime및) 값을 기반으로 행을 하위 그룹으로 정렬합니다draw. 마지막으로 각 하위 그룹에 대한 발생 횟수가 계산됩니다. 그러면 결과 요약이 원본 데이터 샘플을 대체합니다.

- REMOVE_VALUES - 로 종료된 게임의 결과를 삭제합니다draw.
- REPLACE_TEXT -의 값을 수정합니다victory_status. 이 작업에는 mate, resign및에 대해 각각 하나씩 세 가지 발생이 있습니다oufoftime.

대화형 DataBrew 프로젝트 세션에서 각는 데이터 샘플에 적용하는 데이터 변환에 RecipeAction 해당합니다.

DataBrew는 200개 이상의 레시피 작업을 제공합니다. 자세한 내용은 레시피 단계 및 함수 참조 단원을 참조하십시오.

조건 사용

조건을 사용하여 레시피 작업의 범위를 좁힐 수 있습니다. 조건은 특정 열 값을 기반으로 원치 않는 행을 제거하는 등 데이터를 필터링하는 변환에 사용됩니다.

의 레시피 작업을 자세히 살펴보겠습니다chess-project-recipe.

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "black_rating"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "LESS_THAN",
      "Value": "1800",
      "TargetColumn": "black_rating"
    }
  ]
}
```

이 변환은 black_rating 열의 값을 읽습니다. ConditionExpressions 목록에 따라 필터링 기준이 결정됩니다. black_rating 값이 1,800 미만인 행은 데이터 세트에서 제거됩니다.

레시피의 후속 변환은에 대해 동일한 작업을 수행합니다white_rating. 이러한 방식으로 데이터는 각 플레이어(검은색 또는 흰색)가 클래스 A 이상으로 평가되는 게임으로 제한됩니다.

다음은 문자 데이터 열에 적용되는 조건의 또 다른 예입니다.

```
{
  "Action": {
    "Operation": "REMOVE_VALUES",
    "Parameters": {
      "sourceColumn": "winner"
    }
  },
  "ConditionExpressions": [
    {
      "Condition": "IS",
      "Value": "[\"draw\"]",
      "TargetColumn": "winner"
    }
  ]
}
```

이 변환은 `winner` 열의 값을 읽고 값을 찾아 해당 행을 `draw` 제거합니다. 이러한 방식으로 데이터는 명백한 우승자가 있는 게임으로만 제한됩니다.

DataBrew는 다음 조건을 지원합니다.

- `IS` - 열의 값은 조건에 제공된 값과 동일합니다.
- `IS_NOT` - 열의 값이 조건에 제공된 값과 동일하지 않습니다.
- `IS_BETWEEN` - 열의 값은 `GREATER_THAN_EQUAL`와 `LESS_THAN_EQUAL` 파라미터 사이입니다.
- `CONTAINS` - 열의 문자열 값에는 조건에 제공된 값이 포함됩니다.
- `NOT_CONTAINS` - 열의 값에 조건에 제공된 문자열이 포함되지 않습니다.
- `STARTS_WITH` - 열의 값은 조건에 제공된 문자열로 시작합니다.
- `NOT_STARTS_WITH` - 열의 값은 조건에 제공된 문자열로 시작하지 않습니다.
- `ENDS_WITH` - 열의 값은 조건에 제공된 문자열로 끝납니다.
- `NOT_ENDS_WITH` - 열의 값은 조건에 제공된 문자열로 끝나지 않습니다.
- `LESS_THAN` - 열의 값이 조건에 제공된 값보다 작습니다.
- `LESS_THAN_EQUAL` - 열의 값이 조건에 제공된 값보다 작거나 같습니다.
- `GREATER_THAN` - 열의 값이 조건에 제공된 값보다 큽니다.
- `GREATER_THAN_EQUAL` - 열의 값이 조건에 제공된 값보다 크거나 같습니다.
- `IS_INVALID` - 열의 값에 잘못된 데이터 형식이 있습니다.
- `IS_MISSING` - 열에 값이 없습니다.

AWS Glue DataBrew작업 생성, 실행 및 예약

AWS Glue DataBrew에는 두 가지 목적으로 사용되는 작업 하위 시스템이 있습니다.

1. DataBrew 데이터 세트에 데이터 변환 레시피 적용. DataBrew 레시피 작업을 사용하여이 작업을 수행합니다.
2. 데이터 세트를 분석하여 데이터의 포괄적인 프로파일을 생성합니다. DataBrew 프로필 작업을 사용하여이 작업을 수행합니다.

주제

AWS Glue DataBrew레시피 작업 생성 및 작업

DataBrew 레시피 작업을 사용하여 DataBrew 데이터 세트의 데이터를 정리 및 정규화하고 원하는 출력 위치에 결과를 작성합니다. 레시피 작업을 실행해도 데이터 세트 또는 기본 소스 데이터에는 영향을 주지 않습니다. 작업이 실행되면 읽기 전용 방식으로 소스 데이터에 연결됩니다. 작업 출력은 Amazon S3 AWS Glue Data Catalog, 또는 지원되는 JDBC 데이터베이스에서 정의한 출력 위치에 기록됩니다.

다음 절차에 따라 DataBrew 레시피 작업을 생성합니다.

레시피 작업을 생성하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 엽니다.
2. 탐색 창에서 JOBS를 선택하고 레시피 작업 탭을 선택한 다음 작업 생성을 선택합니다.
3. 작업 이름을 입력한 다음 레시피 작업 생성을 선택합니다.
4. 작업 입력에 생성하려는 작업에 대한 세부 정보, 즉 처리할 데이터 세트의 이름 및 사용할 레시피를 입력합니다.

   레시피 작업은 DataBrew 레시피를 사용하여 데이터 세트를 변환합니다. 레시피를 사용하려면 먼저 레시피를 게시해야 합니다.
5. 작업 출력 설정을 구성합니다.

작업 출력의 대상을 제공합니다. 출력 대상에 대해 DataBrew 연결이 구성되어 있지 않은 경우에 설명된 대로 데이터 세트 탭에서 먼저 구성합니다데이터 소스 및 출력에 지원되는 연결. 다음 출력 대상 중 하나를 선택합니다.

- Amazon S3,AWS Glue Data Catalog지원 여부에 관계없이
- Amazon Redshift,AWS Glue Data Catalog지원 여부에 관계없이
- JDBC
- Snowflake 테이블
- 가AWS Glue Data Catalog지원되는 Amazon RDS 데이터베이스 테이블. Amazon RDS 데이터베이스 테이블은 다음 데이터베이스 엔진을 지원합니다.
  - Amazon Aurora
  - MySQL
  - Oracle
  - PostgreSQL
  - Microsoft SQL Server
- AWS Glue Data Catalog Amazon S3 지원.

기반AWS Glue Data Catalog출력의 경우AWS Lake Formation DataBrew는 기존 파일 교체만 지원합니다. 이 접근 방식에서는 파일이 대체되어 데이터 액세스 역할에 대한 기존 Lake Formation 권한을 그대로 유지합니다. 또한 DataBrew는 테이블의 Amazon S3 위치에AWS Glue Data Catalog우선 순위를 부여합니다. 따라서 레시피 작업을 생성할 때 Amazon S3 위치를 재정의할 수 없습니다.

경우에 따라 작업 출력의 Amazon S3 위치가 데이터 카탈로그 테이블의 Amazon S3 위치와 다릅니다. 이러한 경우 DataBrew는 카탈로그 테이블의 Amazon S3 위치로 작업 정의를 자동으로 업데이트합니다. 기존 작업을 업데이트하거나 시작할 때이 작업을 수행합니다.

6. Amazon S3 출력 대상의 경우에만 추가 선택 사항이 있습니다.

   a. Amazon S3에 사용할 수 있는 데이터 출력 형식 중 하나, 선택적 압축 및 선택적 사용자 지정 구분 기호를 선택합니다. 출력 파일에 지원되는 구분 기호는 쉼표, 콜론, 세미콜론, 파이프, 탭, 캐럿, 백슬래시, 스페이스 등 입력용 구분 기호와 동일합니다. 서식 세부 정보는 다음 표를 참조하세요.

형식	파일 확장명(압축되지 않음)	파일 확장명(압축)
쉼표로 구분된 값	.csv	.csv.snappy , .csv.gz, .csv.lz4, csv.bz2, .csv.deflate , csv.br
탭으로 구분된 값	.csv	.tsv.snappy , .tsv.gz, .tsv.lz4, tsv.bz2, .tsv.deflate , tsv.br
Apache Parquet	.parquet	.parquet.snappy , .parquet.gz , .parquet.lz4 , .parquet.lzo , .parquet.br
AWS Glue Parquet	지원되지 않음	.glue.parquet.snappy
Apache Avro	.avro	.avro.snappy , .avro.gz, .avro.lz4 , .avro.bz2 , .avro.deflate , .avro.br
Apache ORC	.orc	.orc.snappy , .orc.lzo, .orc.zlib
XML	.xml	.xml.snappy , .xml.gz, .xml.lz4, .xml.bz2, .xml.deflate , .xml.br
JSON(JSON Lines 형식만 해당)	.json	.json.snappy , .json.gz, .json.lz4 , json.bz2, .json.deflate , .json.br

형식	파일 확장명(압축되지 않음)	파일 확장명(압축)
Tableau 하이퍼	지원되지 않음	해당 사항 없음

b. 단일 파일을 출력할지 아니면 여러 파일을 출력할지 선택합니다. Amazon S3를 사용한 파일 출력에는 세 가지 옵션이 있습니다.

- 파일 자동 생성(권장) - DataBrew에서 최적의 출력 파일 수를 결정합니다.
- 단일 파일 출력 - 단일 출력 파일이 생성됩니다. 사후 처리가 필요하므로이 옵션을 사용하면 추가 작업 실행 시간이 발생할 수 있습니다.
- 다중 파일 출력 - 작업 출력의 파일 수를 지정했는지 여부입니다. 유효한 값은 2~999입니다. 열 파티셔닝을 사용하거나 출력의 행 수가 지정한 파일 수보다 적은 경우 지정한 것보다 적은 파일이 출력될 수 있습니다.

c. (선택 사항) 레시피 작업 출력에 대한 열 파티셔닝을 선택합니다.

열 파티셔닝은 레시피 작업 출력을 여러 파일로 파티셔닝하는 또 다른 방법을 제공합니다. 열 파티셔닝은 새 또는 기존 Amazon S3 출력 또는 새 Data Catalog Amazon S3 출력과 함께 사용할 수 있습니다. 기존 Data Catalog Amazon S3 테이블에서는 사용할 수 없습니다. 출력 파일은 지정한 열 이름의 값을 기반으로 합니다. 지정한 열 이름이 고유한 경우 결과 Amazon S3 폴더 경로는 열 이름의 순서를 기반으로 합니다.

열 파티셔닝의 예는 다음 단원열 파티셔닝의 예을 참조하십시오.

7. (선택 사항) DataBrew가 출력 위치에 쓰는 작업 출력을 암호화하려면 작업 출력에 암호화 활성화를 선택한 다음 암호화 방법을 선택합니다.

- SSE-S3 암호화 사용 - 출력은 Amazon S3 관리형 암호화 키를 사용한 서버 측 암호화를 사용하여 암호화됩니다.
- Use AWS Key Management Service(AWS KMS) - 출력은를 사용하여 암호화됩니다AWS KMS. 이 옵션을 사용하려면 사용하려는AWS KMS키의 Amazon 리소스 이름(ARN)을 선택합니다.AWS KMS키가 없는 경우 키 생성을 선택하여 AWS KMS키를 생성할 수 있습니다.

8. 액세스 권한에서 DataBrew가 출력 위치에 쓸 수 있도록 허용하는AWS Identity and Access Management(IAM) 역할을 선택합니다. 계정이 소유한 위치의 경우 `AwsGlueDataBrewDataAccessRole` 서비스 관리형 역할을 선택할 수 있습니다AWS. 이렇게 하면 DataBrew가 사용자가 소유한AWS리소스에 액세스할 수 있습니다.

9. 고급 작업 설정 창에서 작업 실행 방법에 대한 추가 옵션을 선택할 수 있습니다.

- 최대 단위 수 - DataBrew는 병렬로 실행되는 여러 컴퓨팅 노드를 사용하여 작업을 처리합니다. 기본 노드 수는 5개입니다. 최대 노드 수는 149개입니다.
- 작업 제한 시간 - 작업 실행에 설정한 분 이상 걸리는 경우 제한 시간 오류와 함께 작업이 실패합니다. 기본값은 2,880분 또는 48시간입니다.
- 재시도 횟수 - 실행 중에 작업이 실패하면 DataBrew는 다시 실행을 시도할 수 있습니다. 기본적으로 작업은 재시도되지 않습니다.
- 작업에 Amazon CloudWatch Logs 활성화 - DataBrew가 진단 정보를 CloudWatch Logs에 게시하도록 허용합니다. 이러한 로그는 문제 해결이나 작업 처리 방법에 대한 자세한 정보를 제공하는 데 유용할 수 있습니다.

10. 일정 작업의 경우 작업이 특정 시간에 또는 반복적으로 실행되도록 DataBrew 작업 일정을 적용할 수 있습니다. 자세한 내용은 일정에 따라 작업 실행 자동화 단원을 참조하십시오.
11. 설정이 원하는 대로 되면 작업 생성을 선택합니다. 또는 작업을 즉시 실행하려면 작업 생성 및 실행을 선택합니다.

작업이 실행되는 동안 상태를 확인하여 작업 진행 상황을 모니터링할 수 있습니다. 작업 실행이 완료되면 상태가 성공으로 변경됩니다. 이제 선택한 출력 위치에서 작업 출력을 사용할 수 있습니다.

DataBrew는 나중에 동일한 작업을 실행할 수 있도록 작업 정의를 저장합니다. 작업을 다시 실행하려면 탐색 창에서 작업을 선택합니다. 작업할 작업을 선택한 다음 작업 실행을 선택합니다.

열 파티셔닝의 예

열 분할의 예로 3개의 열을 지정한다고 가정합니다. 각 열의 행에는 두 개의 가능한 값 중 하나가 포함됩니다. `Dept` 열에는 `Admin` 또는 값이 있을 수 있습니다`Eng`. `Staff-type` 열에는 `Part-time` 또는 값이 있을 수 있습니다`Full-time`. `Location` 열에는 `Office1` 또는 값이 있을 수 있습니다`Office2`. 작업 출력의 Amazon S3 버킷은 다음과 같습니다.

```
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Area=Office1/
jobId_timestamp_part0001.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0002.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0003.csv
s3://bucket/output-folder/Dept=Admin/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0004.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office1/
jobId_timestamp_part0005.csv
```

```
s3://bucket/output-folder/Dept=Eng/Staff-type=Part-time/Location=Office2/
jobId_timestamp_part0006.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office1/
jobId_timestamp_part0007.csv
s3://bucket/output-folder/Dept=Eng/Staff-type=Full-time/Location=Office2/
jobId_timestamp_part0008.csv
```

일정에 따라 작업 실행 자동화

언제든지 DataBrew 작업을 다시 실행하고 일정에 따라 DataBrew 작업 실행을 자동화할 수도 있습니다.

DataBrew 작업을 다시 실행하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 엽니다.
2. 탐색 창에서 작업을 선택합니다. 실행할 작업을 선택한 다음 작업 실행을 선택합니다.

특정 시간에 또는 반복적으로 DataBrew 작업을 실행하려면 DataBrew 작업 일정을 생성합니다. 그런 다음 일정에 따라 실행되도록 작업을 설정할 수 있습니다.

DataBrew 작업 일정을 생성하려면

1. DataBrew 콘솔의 탐색 창에서 작업을 선택합니다. 일정 탭을 선택하고 일정 추가를 선택합니다.
2. 일정 이름을 입력한 다음 실행 빈도 값을 선택합니다.

   - 반복 - 작업을 실행할 빈도를 선택합니다(예: 12시간마다). 그런 다음 작업을 실행할 요일을 선택합니다. 선택적으로 작업이 실행되는 시간을 입력할 수 있습니다.
   - 특정 시간 - 작업을 실행할 시간을 입력합니다. 그런 다음 작업을 실행할 요일을 선택합니다.
   - CRON 입력 - 유효한 cron 표현식을 입력하여 작업 일정을 정의합니다. 자세한 내용은 레시피 작업에 대한 cron 표현식 작업 단원을 참조하십시오.
3. 원하는 대로 설정되었으면 [Save]를 선택합니다.

작업을 일정과 연결하려면

1. 탐색 창에서 작업을 선택합니다.
2. 작업할 작업을 선택한 다음 작업에서 편집을 선택합니다.
3. 일정 작업 창에서 일정 연결을 선택합니다. 사용할 일정의 이름을 선택합니다.

4. 원하는 대로 설정되었으면 [Save]를 선택합니다.

레시피 작업에 대한 cron 표현식 작업

cron 표현식에는 각각 공백으로 구분되는 필수 필드 6개가 있습니다. 구문은 다음과 같습니다.

```
Minutes Hours Day-of-month Month Day-of-week Year
```

앞의 구문에서는 표시된 필드에 다음 값과 와일드카드가 사용됩니다.

필드	값	와일드카드
Minutes	0~59	, - * /
Hours	0~23	, - * /
Day-of-month	1~31	, - * ? / L W
월	1~12 또는 JAN-DEC	, - * /
요일	1~7 또는 SUN~SAT	, - * ? / L
연도	1970~2199	, - * /

다음과 같이 이러한 와일드카드를 사용합니다.

- ,(쉼표) 와일드카드는 추가 값을 포함합니다. `Month` 필드에는 1월, 2월, 3월을 `JAN,FEB,MAR` 포함합니다.
- -(en dash) 와일드카드는 범위를 지정합니다. `Day` 필드에서 1~15에는 지정된 달의 1~15일이 포함됩니다.
- *(별표) 와일드카드는 필드의 모든 값을 포함합니다. `Hours` 필드에서 *는 매시간을 포함합니다.
- /(슬래시) 와일드카드로 증분을 지정합니다. `Minutes` 필드에 **`1/10`**를 입력하여 매 10분마다 지정할 수 있습니다. 첫 1분부터 시작합니다(예: 11분, 21분, 31분).
- ?(물음표) 와일드카드는 어떤 한 가지나 다른 것을 지정합니다. 예를 들어 `Day-of-month` 필드에 7을 입력한다고 가정해 보겠습니다. 일곱 번째 주중의 요일을 신경쓰지 않았다면를 입력할 수 있습니까? `Day-of-week` 필드에를 입력합니다.

- Day-of-month 또는 Day-of-week 필드의 L 와일드카드는 월 또는 주의 마지막 날짜를 지정합니다.
-] 필드에서는 W 와일드카드로 어떤 한 평일을 지정할 수 있습니다. Day-of-month Day-of-month 필드에서 3W를 해당 월의 세 번째 평일에 가장 가까운 날을 지정할 수 있습니다.

이러한 필드 및 값에는 다음과 같은 제한 사항이 있습니다.

- 동일한 cron 표현식에 Day-of-month와 Day-of-week 필드를 지정할 수 없습니다. 이 필드 중 하나에 값을 지정하는 경우에는 다른 필드에서 반드시 ?(물음표)를 사용해야 합니다.
- 5분보다 빠른 속도로 이어지는 Cron 표현식은 지원되지 않습니다.

일정을 생성할 때는 다음과 같은 Cron 문자열을 사용할 수 있습니다.

분	시간	일	월	요일	연도	의미
0	10	*	*	?	*	매일 오전 10시(UTC)에 실행
15	12	*	*	?	*	매일 오후 12시 15분(UTC)에 실행
0	18	?	*	월-금	*	매주 월요일부터 금요일까지 오후 6시(UTC)에 실행
0	8	1	*	?	*	매월 1일 오전 8시(UTC)에 실행

분	시간	일	월	요일	연도	의미
0/15	*	*	*	?	*	15분마다 실행
0/10	*	?	*	월-금	*	월요일부터 금요일까지 10분마다 실행
0/5	8~17	?	*	월-금	*	월요일부터 금요일까지 오전 8시부터 오후 5시 55분(UTC) 사이에 5분마다 실행

예를 들어 다음 cron 표현식을 사용하여 매일 12:15 UTC에 작업을 실행할 수 있습니다.

```
15 12 * * ? *
```

작업 및 작업 일정 삭제

작업 또는 작업 일정이 더 이상 필요하지 않은 경우 삭제할 수 있습니다.

작업 삭제

1. 탐색 창에서 작업을 선택합니다.
2. 삭제할 작업을 선택한 다음 작업에서 삭제를 선택합니다.

작업 일정을 삭제하려면

1. 탐색 창에서 작업을 선택한 다음 일정 탭을 선택합니다.
2. 삭제할 일정을 선택한 다음 작업에서 삭제를 선택합니다.

AWS Glue DataBrew프로필 작업 생성 및 작업

프로파일 작업은 데이터 세트에서 일련의 평가를 실행하고 결과를 Amazon S3에 출력합니다. 데이터 프로파일링이 수집하는 정보는 데이터 세트를 이해하고 레시피 작업에서 실행할 데이터 준비 단계를 결정하는 데 도움이 됩니다.

프로필 작업을 실행하는 가장 간단한 방법은 기본 DataBrew 설정을 사용하는 것입니다. 원하는 정보만 반환하도록 프로파일 작업을 실행하기 전에 구성할 수 있습니다.

다음 절차에 따라 DataBrew 프로필 작업을 생성합니다.

프로필 작업을 생성하려면

1. 에 로그인AWS Management Console하고 https://console.aws.amazon.com/databrew/ DataBrew 콘솔을 엽니다.
2. 탐색 창에서 JOBS를 선택하고 프로필 작업 탭을 선택한 다음 작업 생성을 선택합니다.
3. 작업 이름을 입력한 다음 프로필 작업 생성을 선택합니다.
4. 작업 입력에 프로파일링할 데이터 세트의 이름을 입력합니다.
5. (선택 사항) 데이터 프로필 구성 창에서 다음을 구성합니다.

   - 데이터 세트 수준 구성 - 데이터 세트의 모든 열에 대한 프로파일 작업의 세부 정보를 구성합니다.

     선택적으로 데이터 세트에서 중복 행을 감지하고 계산하는 기능을 켤 수 있습니다. 상관 행렬 활성화를 선택하고 열을 선택하여 여러 열의 값이 얼마나 밀접한 관련이 있는지 확인할 수도 있습니다. 데이터 세트 수준에서 구성할 수 있는 통계에 대한 자세한 내용은 섹션을 참조하세요데이터 세트 수준에서 구성 가능한 통계. DataBrew 콘솔에서 또는 DataBrew API 또는AWS SDKs.
   - 열 수준 구성 - 기본 프로필 구성 설정을 사용하여 프로필 작업에 포함할 열을 선택할 수 있습니다. 구성 재정의 추가를 사용하여 수집된 통계 수를 제한하거나 특정 통계의 기본 구성을 재정의할 열을 선택합니다. 열 수준에서 구성할 수 있는 통계에 대한 자세한 내용은 섹션을 참조하세요열 수준에서 구성 가능한 통계. DataBrew 콘솔에서 또는 DataBrew API 또는AWS SDKs.

     지정한 구성 재정의를 프로필 작업에 포함시킨 열에 적용해야 합니다. 열에 대해 구성한 서로 다른 재정의 간에 충돌이 있는 경우 마지막 충돌 재정의가 우선합니다.
6. (선택 사항) 데이터 품질 규칙을 생성하고이 데이터 세트와 연결된 추가 규칙 세트를 적용하거나 이미 적용된 규칙 세트를 제거할 수 있습니다. 데이터 품질 검증에 대한 자세한 내용은 섹션을 참조하세요에서 데이터 품질 검증AWS Glue DataBrew.
7. 고급 작업 설정 창에서 작업 실행 방법에 대한 추가 옵션을 선택할 수 있습니다.

- 최대 단위 수 - DataBrew는 병렬로 실행되는 여러 컴퓨팅 노드를 사용하여 작업을 처리합니다. 기본 노드 수는 5개입니다. 최대 노드 수는 149개입니다.
- 작업 제한 시간 - 작업 실행에 설정한 분 이상 걸리는 경우 제한 시간 오류와 함께 작업이 실패합니다. 기본값은 2,880분 또는 48시간입니다.
- 재시도 횟수 - 실행 중에 작업이 실패하면 DataBrew는 다시 실행을 시도할 수 있습니다. 기본적으로 작업은 재시도되지 않습니다.
- 작업에 Amazon CloudWatch Logs 활성화 - DataBrew가 진단 정보를 CloudWatch Logs에 게시하도록 허용합니다. 이러한 로그는 문제 해결이나 작업 처리 방법에 대한 자세한 정보를 제공하는 데 유용할 수 있습니다.

8. 연결된 일정의 경우 특정 시간에 또는 반복적으로 작업이 실행되도록 DataBrew 작업 일정을 적용할 수 있습니다. 자세한 내용은 일정에 따라 작업 실행 자동화 단원을 참조하십시오.
9. 설정이 원하는 대로 되면 작업 생성을 선택합니다. 또는 작업을 즉시 실행하려면 작업 생성 및 실행을 선택합니다.

에서 프로그래밍 방식으로 프로파일 작업 구성 빌드AWS Glue DataBrew

이 섹션에서는 프로그래밍 방식으로 사용할 수 있는 프로필 작업 단계 및 함수에 대한 설명을 찾을 수 있습니다.AWS Command Line Interface(AWS CLI)에서 또는AWS SDKs.

프로필 작업에서 구성을 사용자 지정하여 DataBrew가 데이터 세트를 평가하는 방법을 제어할 수 있습니다. 구성을 데이터 세트에 적용하거나 특정 열에 적용할 수 있습니다. 프로필 작업을 생성할 때 구성을 빌드한 다음 언제든지 업데이트할 수 있습니다.

프로필 구성 구조는 네 부분으로 구성됩니다.

- ProfileColumns 섹션
- DatasetStatisticsConfiguration 섹션
- ColumnStatisticsConfigurations 섹션
- PII 구성을 위한 EntityDetectorConfiguration 섹션

다음은 한 예입니다.

```
{
    "ProfileColumns": [
        {
            "Name": "example"
```

```
        },
        {
            "Regex": "example.*"
        }
    ],
    "DatasetStatisticsConfiguration": {
        "IncludedStatistics": [
            "CORRELATION"
        ],
        "Overrides": [
            {
                "Statistic": "CORRELATION",
                "Parameters": {
                    "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
                }
            }
        ]
    },
    "ColumnStatisticsConfigurations": [
        {
            "Selectors": [
                {
                    "Name": "example"
                }
            ],
            "Statistics": {
                "IncludedStatistics": [
                    "CORRELATION",
                    "DUPLICATE_ROWS_COUNT"
                ],
                "Overrides": [
                    {
                        "Statistic": "VALUE_DISTRIBUTION",
                        "Parameters": {
                            "binNumber": "10"
                        }
                    }
                ]
            }
        }
    ]
}
```

ProfileColumns 섹션

구조의 ProfileColumns 섹션에서 프로필 작업에서 평가하려는 데이터 세트의 열을 설정합니다. ProfileColumns는 열 선택기 목록(Selectors)입니다. 열 선택기에서 열 이름 또는 정규식을 지정할 수 있습니다. 예를 들면 다음과 같습니다.

```
"ProfileColumns": [{"Name": "example"}, {"Regex": "example.*"}]
```

ProfileColumns이 지정되면 이름이의 이름 또는 정규식과 일치하는 열만 프로파일 작업에 ProfileColumns 포함됩니다. 프로파일 작업이 선택한 열의 데이터 유형을 지원하지 않는 경우 DataBrew는 작업 실행 중에 선택한 열을 건너뜁니다.

ProfileColumns가 정의되지 않은 경우 프로파일 작업은 지원되는 모든 열을 평가합니다. 지원되는 열은 지원되는 데이터 형식인 ByteType, , , ShortType, IntegerType, LongTypeFloatType, DoubleType String또는의 데이터가 포함된 열입니다Boolean.

DatasetStatisticsConfiguration 섹션

구조의 DatasetStatisticsConfiguration 섹션에서 열 간 평가를 위한 구성을 구축할 수 있습니다. 구성에는 IncludedStatistics 및가 포함됩니다Overrides. 예를 들면 다음과 같습니다.

```
"DatasetStatisticsConfiguration": {
    "IncludedStatistics": ["CORRELATION"],
    "Overrides": [
        {
            "Statistic": "CORRELATION",
            "Parameters": {
                "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*
\"}]"
            }
        }
    ]
}
```

에 평가 이름을 추가하여 원하는 평가를 선택할 수 있습니다IncludedStatistics. 예를 들면 다음과 같습니다.

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

를 지정하면 목록의 평가IncludedStatistics만 프로파일 작업에 포함됩니다. IncludedStatistics가 정의되지 않은 경우 프로파일 작업은 기본 설정으로 지원되는 모든 평가를 실행합니다. 에 NONE을 추가하여 모든 평가를 제외할 수 있습니다IncludedStatistics. 예를 들면 다음과 같습니다.

```
"IncludedStatistics": ["NONE"]
```

데이터 세트 수준에서 구성 가능한 통계

구조의 DatasetStatisticsConfiguration 섹션에서 프로필 작업은 다음 표에 표시된 평가를 지원합니다.

통계 이름	설명	지원되는 데이터 유형	기본 상태	프로필 결과의 속성	프로필 결과 유형
DUPLICATE_ROWS_COUNT	데이터 세트의 중복 행 수	모두	Enable	duplicateRowsCoun	정수
상관 관계	두 열 간의 Pearson 상관 계수	number	Enable	상관 관계(선택한 각 열에서)	객체

에서 재정의를 추가하여 각 평가의 기본 설정을 재정의할 IncludedStatistics수 있습니다. 각 재정의에는 특정 평가의 이름과 파라미터 맵이 포함됩니다.

에서 DatasetStatisticsConfiguration프로필 작업은 CORRELATION 재정의를 지원합니다. 이 재정의는 선택한 열 목록에서 두 열 간의 Pearson 상관관계 계수를 계산합니다. 기본 설정은 처음 10개의 숫자 열을 선택하는 것입니다. 여러 열 또는 열 선택기 목록을 지정하여 기본 설정을 재정의할 수 있습니다.

CORRELATION는 다음 파라미터를 사용합니다.

- columnNumber - 숫자 열 수입니다. 프로파일 작업은 데이터 세트에서 첫 번째 n개의 열을 선택합니다. 이 값은 1보다 커야 합니다. "ALL"를 사용하여 모든 숫자 열을 선택합니다.
- columnSelectors: - 열 선택기 목록입니다. 각 선택기에는 열 이름 또는 정규식이 있을 수 있습니다.

예를 들면 다음과 같습니다.

```
{
    "Statistic": "CORRELATION",
    "Parameters": {
        "columnSelectors": "[{\"name\":\"example\"}, {\"regex\":\"example.*\"}]"
    }
}
```

ColumnStatisticsConfigurations 섹션

구조의 ColumnStatisticsConfigurations 섹션에서 특정 열에 대한 구성을 빌드할 수 있습니다. ColumnStatisticsConfigurations는 ColumnStatisticsConfiguration 설정 목록입니다. 에는 통계 구성ColumnStatisticsConfigurationSelectors에 Statistics 대한 , 열 선택기 목록 및가 있습니다. 예를 들면 다음과 같습니다.

```
{
    "Selectors": [{"Name": "example"}
    ],
    "Statistics": {
       "IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
        "Overrides": [
            {
                "Statistic": "VALUE_DISTRIBUTION",
                "Parameters": {
                    "binNumber": "10"
                }
            }
        ]
    }
}
```

Selectors는 열 선택기 목록입니다. 와 마찬가지로 각 열 선택기에서 열 이름 또는 정규식을 지정할 ProfileColumns수 있습니다. 를 지정하면 Selectors의 열 선택기와 일치하는 열에 열 구성이 적용됩니다Selectors. 그렇지 않으면 지원되는 모든 열에 구성이 적용됩니다.

에서 선택한 열의 설정을 재정의Statistics할 수 있습니다. 와 마찬가지로 에는 IncludedStatistics 및 DatasetStatisticsConfigurationStatistics가 있습니다Overrides.

원하는 평가를 선택하려면에 평가 이름을 추가합니다IncludedStatistics.

```
"IncludedStatistics": ["CORRELATION", "DUPLICATE_ROWS_COUNT"]
```

를 지정하면 목록의 평가IncludedStatistics만 프로파일 작업에 포함됩니다. 그렇지 않으면 프로필 작업은 기본 설정으로 지원되는 모든 평가를 실행합니다.

에를 추가하여 모든 평가를 제외NONE할 수 있습니다IncludedStatistics.

```
"IncludedStatistics": ["NONE"]
```

경우에 따라에서 동일한 열에 적용할 수 ColumnStatisticsConfigurations 있는 구성이 다를 IncludedStatistics 수 있습니다. 이러한 경우 프로파일 작업은에서 마지막 구성을 선택하고 선택한 열IncludedStatistics에 ColumnStatisticsConfigurations 적용합니다. 새 구성은 이전 구성을 재정의합니다.

열 수준에서 구성 가능한 통계

에서 ColumnStatisticsConfigurations프로필 작업은 다음 표에 표시된 평가를 지원합니다.

이 표number에서 지원되는 데이터 형식은 속성의 데이터 형식이 ByteType, , ShortType, IntegerType, LongType FloatType또는 중 하나임을 의미합니다DoubleType.

통계 이름	설명	지원되는 데이터 유형	기본 상태	프로필 결과의 속성	프로필 결과 유형
–	열의 이름입니다.	모두	–	이름	문자열
–	열의 데이터 유형입니다.	모두	–	type	문자열

통계 이름	설명	지원되는 데이터 유형	기본 상태	프로필 결과의 속성	프로필 결과 유형
DISTINCT_VALUES_COUNT	고유 값 수입니다. 고유한 값은 한 번 이상 나타나는 값입니다.	number/boolean/string	활성화됨	distinctValuesCount	정수
엔트로피	엔트로피(정보 이론).	number/boolean/string	활성화됨	엔트로피	배정밀도 실수
INTER_QUARTILE_RANGE	숫자의 25%에서 75% 사이의 범위입니다.	number	활성화됨	interquartileRange	배정밀도 실수
첨도	열의 Kurtosis입니다.	number	활성화됨	첨도	배정밀도 실수
MAX	열의 최대값입니다.	숫자/문자열 길이	활성화됨	최대	정수/더블
MAXIMUM_VALUES	열의 최대값 및 해당 개수 목록입니다.	number	활성화됨	maximumValues	List
MEAN	열에 있는 값의 평균값입니다.	숫자/문자열 길이	활성화됨	mean	배정밀도 실수
MEDIAN	열의 값 중앙값입니다.	숫자/문자열 길이	활성화됨	median	배정밀도 실수
MEDIAN_ABSOLUTE_DEVIATION	각 데이터 포인트 간 절대 차이의 중앙값과 숫자 열의 중앙값입니다.	number	활성화됨	medianAbsoluteDeviation	배정밀도 실수
MIN	열의 최소값입니다.	숫자/문자열 길이	활성화됨	min	정수/더블

통계 이름	설명	지원되는 데이터 유형	기본 상태	프로필 결과의 속성	프로필 결과 유형
최소_값	열의 최소값 및 해당 개수 목록입니다.	number	활성화됨	minimumVa lues	List
누락된_값_COUNT	열의 누락된 값 수입니다. Null 및 빈 문자열은 누락된 것으로 간주됩니다.	모두	활성화됨	missingVa luesCount	정수
MODE	열에서 가장 자주 발생하는 값입니다. 여러 값이 자주 나타나는 경우 모드는 해당 값 중 하나입니다.	숫자/문자열 길이	활성화됨	모드	정수/더블
MOST_COMM ON_VALUES	열에서 가장 일반적인 값의 목록입니다.	number/bo olean/string	활성화됨	mostComm nValues	List
OUTLIER_D ETECTION	Z_score 알고리즘으로 열의 이상값을 감지합니다. 특이값 수를 계산하고 감지된 특이값에서 샘플 목록을 추출합니다.	숫자/문자열 길이	활성화됨	zScoreOut liersCoun t, zScoreOut liersSamp le	정수/목록
백분위수	숫자 열의 백분위수 값(5%, 25%, 75%, 95%).	number	활성화됨	백분위수 5, 백분위수25, 백분위수 75, 백분위수95	배정밀도 실수
RANGE	열의 값 범위입니다.	number	활성화됨	range	정수/더블

통계 이름	설명	지원되는 데이터 유형	기본 상태	프로필 결과의 속성	프로필 결과 유형
왜도	열에 있는 값의 왜곡입니다.	number	활성화됨	왜도	배정밀도 실수
STANDARD_DEVIATION	열 값의 편향되지 않은 샘플 표준 편차입니다.	숫자/문자열 길이	활성화됨	standardDeviation	배정밀도 실수
SUM	열의 값 합계입니다.	number	활성화됨	sum	정수/더블
UNIQUE_VALUES_COUNT	고유 값 수입니다. 고유한 값은 값이 한 번만 표시됨을 의미합니다.	number/boolean/string	활성화됨	uniqueValuesCount	정수
VALUE_DISTRIBUTION	범위별로 열의 값 분포를 측정합니다.	숫자/문자열 길이	활성화됨	valueDistribution	List
분산	열의 값 차이입니다.	number	활성화됨	variance	배정밀도 실수
Z_SCORE_DISTRIBUTION	범위별 데이터 포인트의 z-점수 값 분포를 측정합니다.	number	활성화됨	zScoreDistribution	List
ZEROS_COUNT	열의 0(0) 수입니다.	number	활성화됨	zerosCount	정수

에서 재정의를 추가하여 각 평가의 기본 파라미터를 재정의할 `IncludedStatistics`수 있습니다. 각 재정의에는 특정 평가의 이름과 파라미터 맵이 포함됩니다.

ColumnStatisticsConfigurations 열에 대한 파라미터

에서 `ColumnStatisticsConfigurations`프로파일 작업은 다음 파라미터를 지원합니다.

경우에 따라에서 동일한 열에 적용할 수 ColumnStatisticsConfigurations 있는 구성이 다를 IncludedStatistics 수 있습니다. 이러한 경우 프로파일 작업은에서 마지막 구성을 선택하고 선택한 열IncludedStatistics에 ColumnStatisticsConfigurations 적용합니다. 새 구성은 이전 구성을 재정의합니다.

MAXIMUM_VALUES

숫자 열의 최대값과 해당 개수를 나열합니다. 기본 목록 크기는 5입니다. 에 값을 지정하여 목록 크기를 재정의할 수 있습니다sampleSize.

설정

sampleSize - 숫자 열의 최대 값 수와 개수를 포함하는 목록 크기입니다. 이 값은 0보다 커야 합니다. "ALL"를 사용하여 모든 값을 나열합니다.

예제

```
{
    "Statistic": "MAXIMUM_VALUES",
    "Parameters": {
        "sampleSize": "5"
    }
}
```

최소_값

숫자 열의 최소값과 해당 개수를 나열합니다. 기본 목록 크기는 5입니다. 에 값을 지정하여 목록 크기를 재정의할 수 있습니다sampleSize.

설정

sampleSize - 숫자 열의 최대 값 수와 개수를 포함하는 목록 크기입니다. 이 값은 0보다 커야 합니다. "ALL"를 사용하여 모든 값을 나열합니다.

예제

```
{
    "Statistic": "MINIMUM_VALUES",
```

```
    "Parameters": {
        "sampleSize": "5"
    }
}
```

MOST_COMMON_VALUES

열에서 가장 일반적인 값과 해당 수를 나열합니다. 기본 목록 크기는 50입니다. 에 값을 지정하여 목록 크기를 재정의할 수 있습니다`sampleSize`.

설정

`sampleSize` - 숫자 열의 최대 값 수와 개수를 포함하는 목록 크기입니다. 이 값은 0보다 커야 합니다. `"ALL"`를 사용하여 모든 값을 나열합니다.

예제

```
{
    "Statistic": "MOST_COMMON_VALUES",
    "Parameters": {
        "sampleSize": "50"
    }
}
```

OUTLIER_DETECTION

Z_score 알고리즘을 사용하여 숫자 열 또는 문자열 열(문자열 길이 기준)의 이상값을 감지합니다.

프로필 작업은 특이값 수를 계산하고 특이값 및 해당 z-점수의 샘플 목록을 생성합니다. 샘플 목록은 z 점수의 절대값을 기준으로 정렬됩니다. 기본 목록 크기는 50입니다.

Z_Score 알고리즘은 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다. 기본 이상값 임계값은 3입니다.

임계값, 즉 임계값을 하나 더 제공하여 자세한 정보를 얻을 수 있습니다. 최소 임계값은 임계값보다 작아야 합니다. 이 기능은 기본적으로 꺼져 있습니다. 약한 임계값이 지정되면 프로파일 작업은 개수를 하나 더 반환합니다`zScoreMildOutliersCount`. 또한이 경우는 약한 임계값 이상치 샘플을 포함할 `zScoreOutliersSample` 수 있습니다.

설정

- threshold - 이상값을 감지할 때 사용할 임계값입니다. 이 값은 0보다 크거나 같아야 합니다.
- mildThreshold - 이상값을 감지할 때 사용할 최소 임계값입니다. 이 값은 0보다 크거나 같아야 하며 보다 작아야 합니다threshold.
- sampleSize - 열에 이상치를 포함하는 목록의 크기입니다. "ALL"를 사용하여 모든 값을 나열합니다.

예제

```
{
    "Statistic": "OUTLIER_DETECTION",
    "Parameters": {
        "threshold": "5",
        "mildThreshold": "3.5",
        "sampleSize": "20"
    }
}
```

VALUE_DISTRIBUTION

값의 범위를 기준으로 열의 값 분포를 측정합니다. 프로필 작업은 숫자 열 또는 문자열 열(문자열 길이 기준)의 값을 숫자 범위별로 빈으로 그룹화하고 빈 목록을 생성합니다. 빈은 연속적이며 버킷의 상한은 다음 버킷의 하한입니다.

설정

binNumber - 빈 수입니다. 이 값은 0보다 커야 합니다.

예제

```
{
    "Statistic": "VALUE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

Z_SCORE_DISTRIBUTION

숫자 열에서 값의 z-점수 분포를 측정합니다. 프로파일 작업은 값의 z-점수를 숫자 범위별로 빈으로 그룹화하고 빈 목록을 생성합니다. 빈은 연속적이며 버킷의 상한은 다음 버킷의 하한입니다.

설정

`binNumber` - 빈 수입니다. 이 값은 0보다 커야 합니다.

예제

```
{
    "Statistic": "Z_SCORE_DISTRIBUTION",
    "Parameters": {
        "binNumber": "5"
    }
}
```

PII 구성을 위한 EntityDetectorConfiguration 섹션

구조의 `EntityDetectorConfiguration` 섹션에서 DataBrew가 프로필 작업에 대한 개인 식별 정보(PII)로 감지할 데이터 세트의 엔터티 유형을 구성할 수 있습니다.

EntityTypes

DataBrew가 프로필 작업에 대한 PII로 감지할 엔터티 유형을 구성합니다.
`EntityDetectorConfiguration`가 정의되지 않으면 개체 감지가 비활성화됩니다. 데이터 세트에서 다음 엔터티 유형을 감지할 수 있습니다.

- USA_SSN
- EMAIL
- USA_ITIN
- USA_PASSPORT_NUMBER
- PHONE_NUMBER
- USA_DRIVING_LICENSE

- BANK_ACCOUNT
- CREDIT_CARD
- IP_ADDRESS
- MAC_ADDRESS
- USA_DEA_NUMBER
- USA_HCPCS_CODE
- USA_NATIONAL_PROVIDER_IDENTIFIER
- USA_NATIONAL_DRUG_CODE
- USA_HEALTH_INSURANCE_CLAIM_NUMBER
- USA_MEDICARE_BENEFICIARY_IDENTIFIER
- USA_CPT_CODE
- PERSON_NAME
- DATE

개체 유형 그룹USA_ALL도 지원되며 및 PERSON_NAME를 제외한 위의 모든 개체 유형을 포함합니다DATE.

의 유형은 문자열 EntityTypes 배열입니다.

AllowedStatistics

감지된 개체가 포함된 열에서 실행할 수 있는 통계를 구성합니다. AllowedStatistics가 정의되지 않은 경우 감지된 엔터티가 포함된 열에 대한 통계는 계산되지 않습니다. AllowedStatistics 파라미터의 유효한 값 목록은 열 수준에서 구성 가능한 통계 섹션을 참조하세요.

의 유형은 AllowedStatistics 객체의 배열AllowedStatistics입니다.

의 보안AWS Glue DataBrew

의 클라우드 보안AWS이 최우선 순위입니다.AWS고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은AWS와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 -AWS는 클라우드에서AWSAWS서비스를 실행하는 인프라를 보호할 책임이 있습니다.AWS또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. 에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 AWS프로그램 제공 범위 내 서비스규정 준수AWS Glue DataBrew 참조하세요.
- 클라우드의 보안 - 사용자의 책임은 사용하는AWS서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다AWS Glue DataBrew. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 DataBrew를 구성하는 방법을 보여줍니다. 또한 DataBrew 리소스를 모니터링하고 보호하는 데 도움이 되는 다른AWS서비스를 사용하는 방법을 알아봅니다.

주제

- 의 데이터 보호AWS Glue DataBrew
- 에 대한 자격 증명 및 액세스 관리AWS Glue DataBrew
- DataBrew의 로깅 및 모니터링
- 에 대한 규정 준수 검증AWS Glue DataBrew
- 의 복원력AWS Glue DataBrew
- 의 인프라 보안AWS Glue DataBrew
- 의 구성 및 취약성 분석AWS Glue DataBrew

의 데이터 보호AWS Glue DataBrew

DataBrew는 데이터를 보호하도록 설계된 여러 기능을 제공합니다.

주제

- 저장된 데이터 암호화
- 전송 중 암호화
- 키 관리
- 개인 식별 정보(PII) 식별 및 처리
- 다른AWS서비스에 대한 DataBrew 종속성

AWS공동 책임 모델 은AWS Glue DataBrew의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로AWS는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 일반 데이터 보호 규정(GDPR) 센터를 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고AWS 계정AWS IAM Identity Center또는AWS Identity and Access Management(IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여AWS리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다AWS CloudTrail. CloudTrail 추적을 사용하여AWS활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께AWS암호화 솔루션을 사용합니다AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를AWS통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 DataBrew 또는 기타AWS 서비스에서 콘솔AWS CLI, API 또는AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

저장된 데이터 암호화

DataBrew는 DataBrew 프로젝트 및 작업에 대해 저장 데이터 암호화를 지원합니다. 프로젝트와 작업은 암호화된 데이터를 읽을 수 있으며, 작업은 AWS Key Management Service(AWS KMS)를 호출하여 암호화된 데이터를 작성하여 키를 생성하고 데이터를 복호화할 수 있습니다. KMS 키를 사용하여 DataBrew 작업에서 생성된 작업 로그를 암호화할 수도 있습니다. DataBrew 콘솔 또는 DataBrew API를 사용하여 암호화 키를 지정할 수 있습니다.

> **Important**
>
> AWS Glue DataBrew는 대칭AWS KMS 키만 지원합니다. 자세한 내용은 AWS Key Management Service개발자 안내서의 AWS KMS 키를 참조하세요.

암호화가 활성화된 DataBrew에서 작업을 생성할 때 DataBrew 콘솔을 사용하여 S3-managed 서버 측 암호화 키(SSE-S3) 또는 (SSE-KMS)에AWS KMS저장된 KMS 키를 지정하여 저장 데이터를 암호화할 수 있습니다.

> **Important**
>
> Amazon Redshift 데이터 세트를 사용하는 경우 제공된 임시 디렉터리로 언로드된 객체는 SSE-S3로 암호화됩니다.

DataBrew 작업에서 작성한 데이터 암호화

DataBrew 작업은 암호화된 Amazon S3 대상 및 암호화된 Amazon CloudWatch Logs에 쓸 수 있습니다.

주제

- 암호화를 사용하도록 DataBrew 설정
- VPC 작업을 위한에AWS KMS대한 라우팅 생성
- AWS KMS 키를 사용한 암호화 설정

암호화를 사용하도록 DataBrew 설정

다음 절차에 따라 암호화를 사용하도록 DataBrew 환경을 설정합니다.

암호화를 사용하도록 DataBrew 환경을 설정하려면

1. AWS KMS 키를 생성하거나 업데이트하여 DataBrew 작업에 전달되는AWS Identity and Access Management(IAM) 역할에AWS KMS권한을 부여합니다. 이러한 IAM 역할은 CloudWatch Logs 및 Amazon S3 대상을 암호화하는 데 사용됩니다. 자세한 내용은 Amazon CloudWatch Logs 사용자 가이드의 Encrypt Log Data in CloudWatch Logs Using AWS KMS를 참조하세요.

   다음 예제에서 , *"role2"*및 *"role1""role3"*는 DataBrew 작업에 전달되는 IAM 역할입니다. 이 정책 문은 나열된 IAM 역할에이 KMS 키로 암호화 및 복호화할 수 있는 권한을 부여하는 KMS 키 정책을 설명합니다.

   ```
   {
       "Effect": "Allow",
       "Principal": {
           "Service": "logs.region.amazonaws.com",
           "AWS": [
               "role1",
               "role2",
               "role3"
           ]
       },
       "Action": [
           "kms:Encrypt*",
           "kms:Decrypt*",
           "kms:ReEncrypt*",
           "kms:GenerateDataKey*",
           "kms:Describe*"
       ],
       "Resource": "*"
   }
   ```

   키를 사용하여 CloudWatch Logs를 암호화하는 경우 `"Service": "logs.region.amazonaws.com"`과 같이 `Service` 문이 필요합니다.

2. 키를 사용하기 ENABLED 전에AWS KMS키가 로 설정되어 있는지 확인합니다.

AWS KMS키 정책을 사용하여 권한을 지정하는 방법에 대한 자세한 내용은 에서 키 정책 사용을AWS KMS 참조하세요.

VPC 작업을 위한에AWS KMS대한 라우팅 생성

인터넷을 통해 연결하지 않고 Virtual Private Cloud(VPC)의 프라이빗 엔드포인트를 통해AWS KMS에 직접 연결할 수 있습니다. VPC 엔드포인트를 사용하면 VPC와 간의 통신AWS KMS이 전적으로AWS 네트워크 내에서 수행됩니다.

AWS KMS VPC 내에서 VPC 엔드포인트를 생성할 수 있습니다. 이 단계가 없으면 DataBrew 작업이로 실패할 수 있습니다`kms timeout`. 자세한 지침은 AWS Key Management Service개발자 안내서의 VPC 엔드포인트를AWS KMS통해에 연결을 참조하세요.

다음 지침을 따를 때 VPC 콘솔에서 다음을 수행해야 합니다.

- 프라이빗 DNS 이름 활성화를 선택합니다.
- 보안 그룹에서 Java Database Connectivity(JDBC)에 액세스하는 DataBrew 작업에 사용할 보안 그룹(자체 참조 규칙 포함)을 선택합니다.

JDBC 데이터 스토어에 액세스하는 DataBrew 작업을 실행할 때 DataBrew에는AWS KMS엔드포인트에 대한 경로가 있어야 합니다. 경로에 NAT(네트워크 주소 변환) 게이트웨이 또는AWS KMS VPC 엔드포인트를 제공할 수 있습니다. NAT 게이트웨이를 생성하려면 Amazon VPC 사용 설명서의 NAT 게이트웨이를 참조하세요.

AWS KMS 키를 사용한 암호화 설정

작업에서 암호화를 활성화하면 Amazon S3와 CloudWatch 모두에 적용됩니다. 전달된 IAM 역할에는 다음AWS KMS권한이 있어야 합니다.

자세한 내용은 Amazon Simple Storage Service 사용 설명서의 다음 주제를 참조하세요.

- `SSE-S3`에 대한 자세한 내용은 Amazon S3가 관리하는 암호화 키(SSE-S3)를 사용하는 서버 측 암호화로 데이터 보호를 참조하십시오.
- 에 대한 자세한 내용은 KMS 관리형 키를 사용한 서버 측 암호화(SSE-KMS)를 사용하여 데이터 보호를 `SSE-KMS`참조하세요. AWS

전송 중 암호화

AWS는 이동 중인 데이터에 대해 SSL(Secure Sockets Layer) 암호화를 제공합니다.

JDBC 데이터 소스에 대한 DataBrew 지원이 이루어집니다AWS Glue. JDBC 데이터 소스에 연결할 때 DataBrew는 SSL AWS Glue연결 필요 옵션을 포함하여 연결에 대한 설정을 사용합니다. 자세한 내용은 AWS Glue개발자 안내서의 AWS Glue연결 속성 -AWS Glue 섹션을 참조하세요.

AWS KMS는 DataBrew 추출, 변환, 로드(ETL) 처리 및에 대해 '자체 키 가져오기' 암호화와 서버 측 암호화를 모두 제공합니다AWS Glue Data Catalog.

키 관리

DataBrew와 함께 IAM을 사용하여 액세스, 거부 등과 관련된 사용자,AWS리소스, 그룹, 역할 및 세분화된 정책을 정의할 수 있습니다.

조직의 요구 사항에 따라 리소스 기반 정책과 자격 증명 기반 정책을 모두 사용하여 메타데이터에 대한 액세스를 정의할 수 있습니다. 리소스 기반 정책은 교차 계정 액세스와 같은 정책을 설정할 수 있도록 리소스에 대한 액세스가 허용되거나 거부되는 원칙을 나열합니다. 자격 증명 기반 정책은 특히 IAM 내의 사용자, 그룹 및 역할에 연결됩니다.

DataBrew는 자체AWS KMS key"자체 키 가져오기" 암호화 생성을 지원합니다. 또한 DataBrew는 DataBrew 작업에AWS KMS대해의 KMS 키를 사용하여 서버 측 암호화를 제공합니다.

개인 식별 정보(PII) 식별 및 처리

분석 함수 또는 기계 학습 모델을 구축할 때는 개인 식별 정보(PII) 데이터의 노출을 방지하기 위한 보호 장치가 필요합니다. PII는 주소, 은행 계좌 번호 또는 전화번호와 같이 개인을 식별하는 데 사용할 수 있는 개인 데이터입니다. 예를 들어 데이터 분석가와 데이터 과학자가 데이터 세트를 사용하여 일반적인 인구 통계 정보를 검색하는 경우 특정 개인의 PII에 액세스할 수 없어야 합니다.

DataBrew는 데이터 준비 프로세스 중에 PII 데이터를 난독화하는 데이터 마스킹 메커니즘을 제공합니다. 조직의 요구 사항에 따라 다양한 PII 데이터 수정 메커니즘을 사용할 수 있습니다. 사용자가 되돌릴 수 없도록 PII 데이터를 난독화하거나 난독화를 되돌릴 수 있습니다.

DataBrew에서 PII 데이터를 식별하고 마스킹하려면 고객이 PII 데이터를 수정하는 데 사용할 수 있는 변환 세트를 구축해야 합니다. 이 프로세스의 일부로 DataBrew 콘솔의 Data Profile 개요 대시보드에서 PII 데이터 감지 및 통계를 제공합니다.

다음과 같은 데이터 마스킹 기법을 사용할 수 있습니다.

- 대체 - PII 데이터를 다른 실제처럼 보이는 값으로 바꿉니다.
- 셔플링 - 동일한 열의 값을 서로 다른 행으로 셔플링합니다.

- 결정적 암호화 - 열 값에 결정적 암호화 알고리즘을 적용합니다. 결정적 암호화는 항상 값에 대해 동일한 사이퍼텍스트를 생성합니다.
- 확률적 암호화 - 열 값에 확률적 암호화 알고리즘을 적용합니다. 확률적 암호화는 적용될 때마다 다른 사이퍼텍스트를 생성합니다.
- 복호화 - 암호화 키를 기반으로 열을 복호화합니다.
- Nulling out 또는 delete - 특정 필드를 null 값으로 바꾸거나 열을 삭제합니다.
- 마스킹 아웃 - 캐릭터 스크램블링을 사용하거나 열의 특정 부분을 마스킹합니다.
- 해싱 - 열 값에 해시 함수를 적용합니다.

변환 사용에 대한 자세한 내용은 개인 식별 정보(PII) 레시피 단계를 참조하세요. 탐지할 수 있는 엔터티 유형 목록을 포함하여 프로필 작업을 사용하여 PII를 탐지하는 방법에 대한 자세한 내용은 프로그래밍 방식으로 프로필 작업 구성 구축의 PII 구성에 대한 EntityDetectorConfiguration 섹션을 참조하세요.

다른AWS서비스에 대한 DataBrew 종속성

DataBrew 콘솔로 작업하려면AWS계정의 DataBrew 리소스로 작업하려면 최소 권한 집합이 필요합니다. 이러한 DataBrew 권한 외에도 콘솔에는 다음 서비스의 권한이 필요합니다.

- 로그를 표시할 CloudWatch Logs 권한입니다.
- 역할을 나열하고 전달할 수 있는 IAM 권한입니다.
- VPCs, 서브넷, 보안 그룹, 인스턴스 및 기타 객체를 나열할 수 있는 Amazon EC2 권한. DataBrew는 이러한 권한을 사용하여 DataBrew 작업을 실행할 때 VPCs와 같은 Amazon EC2 항목을 설정합니다.
- 버킷 및 객체를 나열할 수 있는 Amazon S3 권한.
- AWS Glue데이터베이스, 파티션, 테이블 및 연결과 같은AWS Glue스키마 객체를 읽을 수 있는 권한.
- AWS Lake Formation Lake Formation 데이터 레이크로 작업할 수 있는 권한.

에 대한 자격 증명 및 액세스 관리AWS Glue DataBrew

AWS Identity and Access Management(IAM)는 관리자가AWS리소스에 대한 액세스를 안전하게 제어할 수AWS 서비스있도록 도와주는 입니다. IAM 관리자는 DataBrew 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수AWS 서비스있는 입니다.

주제

ID를 통한 인증

인증은 자격 증명 자격 증명을AWS사용하여에 로그인하는 방법입니다.AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center(IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를AWS제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정루트 사용자

를 생성할 때 모든AWS 서비스및 리소스에 대한 완전한 액세스 권한이 있는AWS 계정theroot 사용자라는 하나의 로그인 자격 증명으로AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을AWS사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다.AWS CLIAWS자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을AWS생성하고 자격 증명 또는 리소스에 연결하여AWS에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을AWS평가합니다. 대부분의 정책은에 JSON 문서AWS로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한

정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의AWS관리형 정책을 사용할 수 없습니다.

DataBrew는 리소스 기반 정책을 지원하지 않습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

DataBrew는 ACLs 지원하지 않습니다.

기타 정책 유형

AWS는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) -AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를AWS결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS Glue DataBrew및AWS Lake Formation

AWS Glue DataBrew는AWS Glue Data Catalog테이블에 대한AWS Lake Formation권한을 지원합니다. 데이터 세트가 Lake Formation에 등록된AWS Glue Data Catalog테이블을 사용하는 경우 프로젝트 또는 작업에 제공되는 IAM 역할에는 테이블에 대한 DESCRIBE 및 SELECT Lake Formation 권한이 있어야 합니다.

AWS Glue DataBrew는 기반AWS Glue Data Catalog테이블에 쓰기를 지원합니다AWS Lake Formation. DataBrew 작업이 Lake Formation에 등록된 데이터 카탈로그를 사용하는 경우 작업에 제공된 IAM 역할에는 관련 테이블에 대한 Lake Formation의 INSERT, ALTER 및 DELETE 권한이 있어야 합니다. IAM 역할에는 `glue:UpdateTable` 권한이 있어야 하며 데이터 카탈로그 테이블과 연결된 데이터 위치에 대한 권한도 있어야 합니다.

AWS Glue DataBrew에서 IAM을 사용하는 방법

IAM을 사용하여 DataBrew에 대한 액세스를 관리하기 전에 DataBrew에서 사용할 수 있는 IAM 기능을 이해해야 합니다. DataBrew 및 기타AWS서비스에서 IAM을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

주제

- DataBrew 자격 증명 기반 정책
- DataBrew의 리소스 기반 정책
- DataBrew IAM 역할

DataBrew 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스를 지정할 수 있으며 작업이 허용되거나 거부되는 조건도 지정할 수 있습니다. DataBrew는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알고 싶다면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

작업

관리자는AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉,AWS JSON 정책은 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

JSON 정책의 작업 요소는 정책에서 액세스를 허용하거나 거부할 수 있는 작업을 설명합니다. 일반적으로 정책 작업의 이름은 연결된AWS API 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한

전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

DataBrew의 정책 작업은 작업 앞에 접두사를 사용합니다`databrew:`. 예를 들어 누군가에게 Amazon EC2 `RunInstances` API 작업을 통해 Amazon EC2 인스턴스를 실행할 권한을 부여하려면 해당 정책에 `ec2:RunInstances` 작업을 포함하세요. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다. DataBrew는 사용자가 수행할 수 있는 작업을 설명하는 고유한 작업 세트를 정의합니다.

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "databrew:CreateRecipeJob",
      "databrew:UpdateSchedule"
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `Describe`라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

```
"Action": "databrew:Describe*"
```

DataBrew 작업 목록을 보려면 IAM 사용 설명서의 에서 정의한 작업을AWS Glue DataBrew 참조하세요.

리소스

관리자는AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

다음은 리소스 수준 권한을 지원하지 않는 DataBrew APIs

- ListDatasets
- ListJobs
- ListProjects

- ListRecipes
- ListRulesets
- ListSchedules

DataBrew 데이터 세트 리소스에는 다음과 같은 Amazon 리소스 이름(ARN)이 있습니다.

```
arn:${Partition}:databrew:${Region}:${Account}:dataset/${Name}
```

ARNs 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARNs) 및AWS서비스 네임스페이스를 참조하세요.

예를 들어, 문에서 `i-1234567890abcdef0` 인스턴스를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/my-chess-dataset"
```

특정 계정에 속하는 모든 인스턴스를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:databrew:us-east-1:123456789012:dataset/*"
```

특정 리소스에서 리소스를 생성하기 위한 작업과 같은 일부 DataBrew 작업은 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

```
"Resource": "*"
```

DataBrew 리소스 유형 및 해당 ARNs 목록을 보려면 IAM 사용 설명서의 에서 정의한 리소스를AWS Glue DataBrew 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Glue DataBrew가 정의한 작업을 참조하세요.

조건 키

DataBrew는 서비스별 조건 키를 제공하지 않지만 일부 전역 조건 키 사용을 지원합니다. 모든AWS전역 조건 키를 보려면 IAM 사용 설명서의 AWS전역 조건 컨텍스트 키를 참조하세요.

예제

DataBrew 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS Glue DataBrew.

DataBrew의 리소스 기반 정책

DataBrew는 리소스 기반 정책을 지원하지 않습니다.

DataBrew IAM 역할

IAM 역할은 특정 권한이 있는AWS계정 내 엔터티입니다.

DataBrew에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 페더레이션을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken과 같은AWS STS API 작업을 호출하여 임시 보안 자격 증명을 얻을 수 있습니다.

DataBrew는 임시 자격 증명 사용을 지원합니다.

서비스 연결 역할

서비스 연결 역할을 사용하면AWS서비스가 다른 서비스의 리소스에 액세스하여 사용자를 대신하여 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

DataBrew에서 IAM 역할 선택

DataBrew에서 데이터 세트 리소스를 생성할 때 사용자를 대신하여 DataBrew 액세스를 허용하는 IAM 역할을 선택합니다. 이전에 서비스 역할 또는 서비스 연결 역할을 생성한 경우 DataBrew는 선택할 수 있는 역할 목록을 제공합니다. Amazon S3 버킷 또는AWS Glue Data Catalog리소스에 대한 읽기 액세스를 허용하는 역할을 적절하게 선택해야 합니다.

에 대한 자격 증명 기반 정책 예제AWS Glue DataBrew

기본적으로 사용자 및 역할에는 DataBrew 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management ConsoleAWS CLI또는AWS APIs를 사용하여 작업을 수행할 수 없습니다. 관리자는 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서의 JSON 탭에서 정책 생성을 참조하세요.

주제

- 정책 모범 사례
- DataBrew 콘솔 사용
- 사용자가 자체 권한을 볼 수 있도록 허용
- 태그를 기반으로 DataBrew 리소스 관리

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 DataBrew 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS관리형 정책을 사용합니다. 에서 사용할 수 있습니다AWS 계정. 사용 사례에 맞는AWS고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS관리형 정책 또는 AWS직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다.AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

DataBrew 콘솔 사용

AWS Glue DataBrew콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해AWS 계정의 DataBrew 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다. 최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 생성하는 경우 콘솔은 해당 정책이 있는 사용자 또는 역할에 대해 의도한 대로 작동하지 않습니다.

사용자와 역할이 DataBrew 콘솔을 사용할 수 있도록 하려면 다음AWS관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

```
AWSDataBrewConsoleAccess
```

AWS CLI또는 DataBrew API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자가 자체 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여 줍니다. 이 정책에는 콘솔에서 또는AWS CLI또는AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
```

```
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

태그를 기반으로 DataBrew 리소스 관리

자격 증명 기반 정책의 조건을 사용하여 태그를 기반으로 DataBrew 리소스를 관리할 수 있습니다. 예를 들어 리소스를 삭제, 업데이트 또는 설명할 수 있습니다. 다음 예제에서는 프로젝트 삭제를 거부하는 정책을 보여줍니다. 그러나 프로젝트 태그 소유자의 값이 관리자인 경우에만 삭제가 거부됩니다. 또한이 정책은 콘솔에서이 작업을 거부하는 데 필요한 권한을 부여합니다.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DeleteResourceInConsole",
         "Effect": "Allow",
         "Action": "databrew:DeleteProject",
         "Resource": "*"
       },
       {
         "Sid": "DenyDeleteProjectIfAdminTag",
         "Effect": "Deny",
         "Action": "databrew:DeleteProject",
         "Resource": "arn:aws:databrew:*:*:project/*",
         "Condition": {
            "StringEquals": {"aws:ResourceTag/Owner": "admin"}
         }
      }
   ]
}
```

이 정책을 계정의 사용자에게 연결할 수 있습니다. richard-roe라는 사용자가 DataBrew 프로젝트를 삭제하려고 하면 리소스에 Owner=admin 또는 owner=admin 태그가 지정되지 않아야 합니다. 그렇지 않으면 사용자가 프로젝트를 삭제할 수 있는 권한이 거부됩니다. 조건 키 이름은 대소문자를 구분하지 않으므로 조건 태그 키 소유자는 소유자와 소유자 모두와 일치합니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.

> **Note**
>
> ListDatasets, ListJobs, ListProjects, ListRecipes, ListRulesets 및 ListSchedules는 태그 기반 액세스 제어를 지원하지 않습니다.

AWS에 대한 관리형 정책AWS Glue DataBrew

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다AWS관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하기 위해서는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면AWS관리형 정책을 사용할 수 있습니다. 이러한 정책은 일반적인 사용 사례를 다루며AWS계정에서 사용할 수 있습니다.AWS관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS관리형 정책을 참조하세요.

AWS서비스는AWS관리형 정책을 유지 관리하고 업데이트합니다.AWS관리형 정책에서는 권한을 변경할 수 없습니다. 서비스는 경우에 따라AWS관리형 정책에 추가 권한을 추가하여 새 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새 기능이 시작되거나 새 작업을 사용할 수 있게 될 때AWS관리형 정책을 업데이트할 가능성이 높습니다. 서비스는AWS관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을AWS지원합니다. 예를 들어 ReadOnlyAccessAWS관리형 정책은 모든AWS서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을AWS추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 직무에 관한AWS관리형 정책을 참조하세요.

AWS관리형 정책에 대한 DataBrew 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 DataBrew의AWS관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 DataBrew 문서 기록 페이지에서 RSS 피드를 구독하세요. 관리형 정책은의AWS IAM 콘솔에서 찾을 수 있습니다AwsGlueDataBrewFullAccessPolicy.

변경	설명	Date
AWSGlueDataBrewServiceRole -AWS Glue에 대한 읽기 권한이 추가되었습니다.	이 업데이트는를 추가합니다glue:GetCustomEntityType . 이 권한은 PII 식별이 활성화된 상태에서AWS Glue DataBrew프로필 작업을 실행하는 데 필요합니다.	2024년 3월 20일
AWSGlueDataBrewServiceRole -에 대한 읽기 권한이 AWS Glue추가되었습니다.	이 업데이트는를 추가합니다glue:BatchGetCustomEntityTypes . 이 권한은 PII 식별이 활성화된AWS Glue DataBrew프로필 작업을 실행하는 데 필요합니다.	2022년 5월 9일
AwsGlueDataBrewFullAccessPolicy - Amazon Redshift-Data DescribeStatements 및 Amazon S3 GetLifecycleConfiguration에 대한 읽기 권한이 추가되었습니다.	이 업데이트는 Amazon Redshift 기반 데이터 세트를 생성할 때 SQL 검증을 지원하기 redshift-data:DescribeStatement 위해를 추가합니다. 또한 임시 디렉터리로 제공하는 Amazon S3 버킷 접두사에 수명 주기가 구성되어 있는지 여부를 평가하기 s3:GetLifecycleConfiguration 위해를 추가합니다. 또한이 변경 사항은 "databrew:*" 권한을 모든 DataBrew APIs.	2022년 2월 4일
AwsGlueDataBrewFullAccessPolicy -AWS Secrets Manager에 대한 읽기/쓰기 권한이 추가되었습니다.	이 업데이트는 DataBrew 변환secretsmanager:CreateSecret secretsmanager:GetSecretValue 에 사용할 기본 보안 암호databrew!default 인	2021년 11월 18일

변경	설명	Date
	라는 보안 암호에 및를 추가합니다. 또한 DataBrew 콘솔에서 보안 암호를 생성하기 위해 접두사가 인 보안 암호`AwsGlueDataBrew-` 에 대한 권한을 CreateSecret 에 추가합니다. AWS Key Management Service API 참조에 설명된 GenerateRandom은 암호화 방식으로 안전한 임의 바이트 문자열을 생성하는 데 사용됩니다.	
AWSGlueDataBrewServiceRole -AWS Secrets Manager에 대한 읽기/쓰기 권한이 추가되었습니다.	이 업데이트는 DataBrew 변환`secretsmanager:GetSecretValue` 에 사용할 기본 보안 암호`databrew!default` 인 라는 보안 암호에를 추가합니다.	2021년 11월 18일

변경	설명	Date
AwsGlueDataBrewFullAccessPolicy -AWS Secrets Manager에 대한 읽기/쓰기 권한이 추가되었습니다.	이 업데이트는 DataBrew 변환`secretsmanager:CreateSecret` `secretsmanager:GetSecretValue` 에 사용할 기본 보안 암호`databrew!default` 인 라는 보안 암호에 및를 추가합니다. 또한 DataBrew 콘솔에서 보안 암호를 생성하기 위해 접두사가 인 보안 암호`AwsGlueDataBrew-` 에 대한 권한을 CreateSecret 에 추가합니다. `kms:GenerateRandom` (`https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateRandom.html`)는 암호화 방식으로 안전한 임의 바이트 문자열을 생성하는 데 사용됩니다.	2021년 11월 18일
AWSGlueDataBrewServiceRole -AWS Secrets Manager에 대한 읽기/쓰기 권한이 추가되었습니다.	이 업데이트는 DataBrew 변환`secretsmanager:GetSecretValue` 에 사용할 기본 보안 암호`databrew!default` 인 라는 보안 암호에 를 추가합니다.	2021년 11월 18일

변경	설명	Date
AwsGlueDataBrewFullAccessPolicy -AWS Glue카탈로그 데이터베이스에 대한 읽기 권한과AWS Glue카탈로그 테이블에 대한 생성 권한이 추가되었습니다.	이 업데이트는 DataBrew 작업에 대한 출력 구성의 일부로 AWS Glue카탈로그 데이터베이스를 나열하고 기존 데이터베이스에서 새 카탈로그 테이블을 생성할 수 있는 권한을 추가합니다.	2021년 6월 30일
AwsGlueDataBrewFullAccessPolicy - Amazon AppFlow 데이터 세트 기능에 대한 읽기/쓰기 권한이 추가되었습니다.	이 업데이트는 기존 Amazon AppFlow 흐름 및 흐름 실행을 읽고 흐름 실행을 생성할 수 있는 권한을 추가합니다.	2021년 4월 28일
AwsGlueDataBrewFullAccessPolicy - 데이터베이스 데이터 세트에 대한 읽기 권한이 추가되었습니다.	이 업데이트는 기존AWS Glue연결을 읽고 DataBrew에 사용할 새AWS Glue연결을 생성할 수 있는 권한을 추가합니다. 또한 콘솔에서 새 연결을 더 쉽게 생성할 수 있도록 Amazon VPC 리소스 및 Amazon Redshift 클러스터를 나열할 수 있습니다. 또한 읽기 보안AWS Secrets Manager암호를 나열할 수 있지만 나열할 수 있는 권한도 부여합니다.	2021년 3월 30일
DataBrew에서 변경 사항 추적 시작	DataBrew는AWS관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2021년 3월 30일

에서 자격 증명 및 액세스 문제 해결AWS Glue DataBrew

다음 정보를 사용하여 DataBrew 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- DataBrew에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내AWS계정 외부의 사람이 내 DataBrew 리소스에 액세스하도록 허용하고 싶습니다.

DataBrew에서 작업을 수행할 권한이 없음

에서 작업을 수행할 권한이 없다는AWS Management Console메시지가 표시되면 관리자에게 문의하여 도움을 받으세요. 관리자는 로그인 보안 인증 정보를 제공한 사람입니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 프로젝트에 대한 세부 정보를 보려고 하지만 `databrew:DescribeProject` 권한이 없는 경우에 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 databrew:DescribeProject on resource: my-example-project
```

이 경우, Mateo는 *`my-example-project`* 작업을 사용하여 `databrew:`*`GetProject`* 리소스에 액세스하도록 허용하는 정책을 업데이트하라고 관리자에게 요청합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 권한이 없다는 오류가 수신되면 DataBrew에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 콘솔을 사용하여 DataBrew에서 작업을 수행하려고 `marymajor` 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내AWS계정 외부의 사람이 내 DataBrew 리소스에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- DataBrew가 이러한 기능을 지원하는지 여부를 알아보려면 섹션을 참조하세요AWS Glue DataBrew에서 IAM을 사용하는 방법.
- 소유AWS 계정한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조AWS 계정하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가AWS 계정소유한에 대한 액세스 권한 제공을AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

DataBrew의 로깅 및 모니터링

모니터링은 DataBrew 및AWS솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. 다중 지점 장애가 발생할 경우 보다 쉽게 디버깅할 수 있도록AWS솔루션의 모든 부분으로부터 모니터링 데이터를 수집해야 합니다.는 DataBrew 리소스를 모니터링하고 잠재적 인시던트에 대응하기 위한 몇 가지 도구를AWS제공합니다.

Amazon CloudWatch 경보

> Amazon CloudWatch 경보를 사용하면 지정한 기간 동안 단일 지표를 감시합니다. 지표가 지정된 임계값을 초과하면 Amazon SNS 주제 또는AWS Auto Scaling정책으로 알림이 전송됩니다. CloudWatch 경보는 특정 상태에 있다고 해서 작업을 호출하지 않습니다. 대신, 상태가 변경되어 지정된 기간 동안 유지되어야 합니다.

AWS CloudTrail로그

CloudTrail은 DataBrew에서 사용자, 역할 또는AWS서비스가 수행한 작업에 대한 레코드를 제공합니다. CloudTrail에서 수집한 정보를 사용하여 DataBrew에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

에 대한 규정 준수 검증AWS Glue DataBrew

타사 감사자는 여러 규정 준수 프로그램의AWS Glue DataBrew일환으로의 보안 및AWS규정 준수를 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다.

AWS 서비스가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스규정 준수 프로그램 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라AWS 서비스결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS보안 설명서를AWS 서비스참조하세요.

의 복원력AWS Glue DataBrew

AWS글로벌 인프라는AWS리전 및 가용 영역을 중심으로 구축됩니다.AWS리전은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

의 경우 하나 이상의 재시도를 사용하도록 작업을 구성하는AWS Glue DataBrew것이 좋습니다. 작업에 대한 재시도 횟수는 DataBrew 콘솔의 고급 작업 설정에서 구성됩니다.

AWS리전 및 가용 영역에 대한 자세한 내용은 AWS글로벌 인프라를 참조하세요.

의 인프라 보안AWS Glue DataBrew

관리형 서비스의 일부로AWS Glue DataBrew는 Amazon Web Services: 보안 프로세스 개요 백서에 설명된AWS글로벌 네트워크 보안 절차로 보호됩니다.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 DataBrew에 액세스합니다. 클라이언트가 전송 계층 보안(TLS) 1.0 이상을 지원해야 합니다. TLS 1.2 이상을 권장합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

주제

- VPC AWS Glue DataBrew에서 사용
- VPC 엔드포인트AWS Glue DataBrew와 함께 사용

VPC AWS Glue DataBrew에서 사용

Amazon VPC를 사용하여AWS리소스를 호스팅하는 경우 Amazon VPC 서비스를 기반으로 Virtual Private Cloud(VPC)를 통해 트래픽을 라우팅AWS Glue DataBrew하도록를 구성할 수 있습니다. DataBrew는 먼저 지정한 서브넷에 탄력적 네트워크 인터페이스를 프로비저닝하여이 작업을 수행합니다. 그런 다음 DataBrew는 사용자가 지정한 보안 그룹을 해당 네트워크 인터페이스에 연결하여 액세스를 제어합니다. 지정된 보안 그룹에는 모든 트래픽에 대한 자체 참조 인바운드 및 아웃바운드 규칙이 있어야 합니다. 또한 VPC에는 DNS 호스트 이름과 확인 기능이 켜져 있어야 합니다. 자세한 내용은 AWS Glue개발자 안내서의 VPC를 JDBC 데이터 스토어에 연결하도록 설정을 참조하세요.

AWS Glue Data Catalog데이터 세트의 경우 데이터 카탈로그에서 연결을 생성할AWS Glue때 VPC 정보가 구성됩니다. 이 연결을 위한 데이터 카탈로그 테이블을 생성하려면AWS Glue콘솔에서 크롤러를 실행합니다. 자세한 내용은 AWS Glue개발자 안내서의 채우기를AWS Glue Data Catalog 참조하세요.

데이터베이스 데이터 세트의 경우 DataBrew 콘솔에서 연결을 생성할 때 VPC 정보를 지정합니다.

NAT가 없는 VPC 서브넷AWS Glue DataBrew에서를 사용하려면 Amazon S3에 대한 게이트웨이 VPC 엔드포인트와AWS Glue인터페이스에 대한 VPC 엔드포인트가 있어야 합니다. 자세한 내용은 Amazon VPC 설명서의 게이트웨이 엔드포인트 생성 및 인터페이스 VPC 엔드포인트()를 참조하세요. AWS

PrivateLink DataBrew에서 프로비저닝한 탄력적 인터페이스에는 퍼블릭 IPv4 주소가 없으므로 VPC 인터넷 게이트웨이 사용을 지원하지 않습니다.

Amazon S3 인터페이스 엔드포인트는 현재 지원되지 않습니다.AWS Secrets Manager를 사용하여 보안 암호를 저장하는 경우 Secrets Manager로 가는 경로가 필요합니다. 암호화를 사용하는 경우AWS Key Management Service()에 대한 경로가 필요합니다AWS KMS.

VPC 엔드포인트AWS Glue DataBrew와 함께 사용

Amazon VPC를 사용하여AWS리소스를 호스팅하는 경우 VPC 엔드포인트를 프로비저닝하여 VPC와 DataBrew 간에 프라이빗 연결을 설정할 수 있습니다. 이 VPC 엔드포인트를 사용하여 DataBrew API를 호출할 수 있습니다.

DataBrew VPC 엔드포인트는 VPC에서 DataBrew를 사용하는 데 필요하지 않습니다. 자세한 내용은 VPC AWS Glue DataBrew에서 사용 단원을 참조하십시오.

및 VPC 엔드포인트를 모두 지원하는 모든AWS리전에서 VPC 엔드포인트AWS Glue와AWS Glue함께를 사용할 수 있습니다.

자세한 내용은 Amazon VPC 사용 설명서의 다음 주제를 참조하세요.

- Amazon VPC란 무엇인가?
- 인터페이스 엔드포인트 생성

의 구성 및 취약성 분석AWS Glue DataBrew

구성 및 IT 제어는AWS와 고객 간의 공동 책임입니다. 자세한 내용은AWS공동 책임 모델을 참조하세요.

모니터링AWS Glue DataBrew

모니터링은AWS Glue DataBrew및 다른AWS솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 DataBrew를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를AWS제공합니다.

- Amazon CloudWatch는AWS리소스와AWS에서 실행하는 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용자 안내서를 참조하세요.
- Amazon CloudWatch Events를 사용하면 DataBrew에서 특정 이벤트에 대한 자동 알림을 설정할 수 있습니다. DataBrew의 이벤트는 거의 실시간으로 CloudWatch Events로 전달됩니다. 리소스 공유의 변경을 나타내는 이벤트에 대한 응답으로 이벤트를 모니터링하고 대상을 호출하도록 CloudWatch Events를 구성할 수 있습니다. 리소스 공유를 변경하면 리소스 공유 소유자와 리소스 공유에 대한 액세스 권한이 부여된 보안 주체 모두에게 이벤트가 트리거됩니다. 자세한 내용은 Amazon CloudWatch Events 사용자 안내서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임계값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구성이 뛰어난 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용자 안내서를 참조하세요.
- AWS CloudTrail는AWS계정에서 또는 계정을 대신하여 수행한 API 호출 및 관련 이벤트를 캡처합니다. 그리고 나서 사용자가 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail사용 설명서를 참조하십시오.

주제

Amazon CloudWatch를 사용하여 DataBrew 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 CloudWatch를 사용하여 DataBrew를 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS Glue DataBrew는 `AWS/DataBrew` 네임스페이스에 다음 지표를 보고합니다.

지표	설명
`SessionCount`	고객 계정 전체의 총 DataBrew 세션 수 유효한 차원: LogGroupName 유효한 통계: Sum 단위: 개

CloudWatch Events를 사용하여 DataBrew 자동화

Amazon CloudWatch Events를 사용하면AWS서비스를 자동화하고 애플리케이션 가용성 문제 또는 리소스 변경과 같은 시스템 이벤트에 자동으로 대응할 수 있습니다.AWS서비스의 이벤트는 거의 실시간으로 CloudWatch Events로 전달됩니다. 원하는 이벤트만 표시하도록 간단한 규칙을 작성한 후 규칙과 일치하는 이벤트 발생 시 실행할 자동화 작업을 지정할 수 있습니다. 자동으로 트리거할 수 있는 태스크는 다음과 같습니다.

- Amazon EC2 실행 명령 간접 호출
- Amazon Kinesis Data Streams로 이벤트 릴레이
- AWS Step Functions상태 시스템 활성화
- SNS 주제 또는 Amazon SQS 대기열 알림

DataBrew는AWS계정의 리소스 상태가 변경될 때마다 CloudWatch Events에 이벤트를 보고합니다. 이벤트는 최선의 작업을 기반으로 발생됩니다.

다음은 DataBrew 작업의 다양한 상태를 보여주는 몇 가지 이벤트의 예입니다STOPPED. SUCCEEDED, FAILED, 및 TIMEOUT.

```
{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T18:57:21Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "SUCCEEDED",
    "jobRunId": "db_abcdef0123456789abcdef0123456789abcdef0123456789abcdef0123456789",
    "message": "Job run succeeded"
  }
}

{
  "version": "0",
  "id": "abcdef01-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-09-07T06:02:03Z",
  "region": "us-west-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "ERROR",
    "state": "FAILED",
    "jobRunId": "db_0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef",
    "message": "AnalysisException: 'Path does not exist: s3://MyBucket/MyFile;'"
  }
}

{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
```

```
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "WARN",
    "state": "TIMEOUT",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run timed out"
  }
}

{
  "version": "0",
  "id": "abcdef00-1234-5678-9abc-def012345678",
  "detail-type": "DataBrew Job State Change",
  "source": "aws.databrew",
  "account": "123456789012",
  "time": "2017-11-20T20:22:06Z",
  "region": "us-east-2",
  "resources": [],
  "detail": {
    "jobName": "MyJob",
    "severity": "INFO",
    "state": "STOPPED",
    "jobRunId": "db_abc0123456789abcdef0123456789abcdef0123456789abcdef0123456789def",
    "message": "Job run stopped"
  }
}
```


자세한 내용은 Amazon CloudWatch Events 사용자 안내서를 참조하세요.

CloudWatch Logs를 사용하여 DataBrew 모니터링

DataBrew 작업 하위 시스템에서 세부 정보를 수집하고 검토할 수 있도록 하는 CloudWatch Logs를 사용하여 DataBrew 작업을 모니터링할 수 있습니다. 이러한 로그는 프로파일 및 레시피 작업에서 사용 중인 리소스에 대한 인사이트를 얻거나 문제 해결을 위해 유용할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.

를 사용하여 DataBrew API 호출 로깅AWS CloudTrail

DataBrew는 DataBrew에서 사용자AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는AWS서비스와 통합됩니다. CloudTrail은 DataBrew에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 DataBrew 콘솔의 호출과 DataBrew API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 DataBrew 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 DataBrew에 수행된 요청을 확인할 수 있습니다. 또한 어떤 IP 주소에서 요청했는지, 누가 언제 요청했는지 등의 추가 세부 정보도 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail사용 설명서를 참조하세요.

CloudTrail의 DataBrew 정보

AWS계정을 생성할 때 계정에서 CloudTrail이 활성화됩니다. DataBrew에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른AWS서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다.AWS계정에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다. 자세한 내용은AWS CloudTrail사용 설명서에서 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

DataBrew에 대한 이벤트를 포함하여AWS계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 기본적으로 콘솔에서 추적을 생성하면 추적이 모든AWS리전에 적용됩니다. 추적은AWS파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른AWS서비스를 구성할 수 있습니다. 자세한 내용은 AWS CloudTrail User Guide의 다음 섹션을 참조하세요.

- 트레일 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에서 Amazon SNS 알림 구성
- 여러 리전으로부터 CloudTrail 로그 파일 받기 및 여러 계정으로부터 CloudTrail 로그 파일 받기

모든 DataBrew 작업은 CloudTrail에서 로깅되며 API 참조에 문서화됩니다. 예를 들어 `CreateDataset`, `UpdateRecipe`, `StartJobRun` 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 사용자 자격 증명으로 했는지 여부
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른AWS서비스에서 이루어졌는지 여부입니다.

자세한 설명은 CloudTrail userIdentity 요소를 참조하세요.

DataBrew 로그 파일 항목 이해

다시 말하지만 CloudTrail 추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예는 `CreateProfileJob`작업을 보여주는 CloudTrail 로그 항목입니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::1234567890:user/joe",
        "accountId": "1234567890",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "joe"
    },
    "eventTime": "2020-11-09T18:54:44Z",
    "eventSource": "databrew.amazonaws.com",
    "eventName": "CreateProfileJob",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "requestParameters": {
        "OutputLocation": {
            "Bucket": "bucketName",
            "Key": "keyName"
        },
        "DatasetName": "my-chess-dataset",
        "RoleArn": "arn:aws:iam::1234567890:role/custom-role",
        "Name": "my-profile-job"
    },
```

```
    "responseElements": {
        "Name": "my-profile-job"
    },
    "requestID": "993bc3b8-3980-48dd-961e-c1c8529eb248",
    "eventID": "f8128dfa-df29-458b-a2d5-34805b46eefd",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "1234567890"
}
```


AWS Glue Databrew에서AWS사용자 알림 사용

AWS사용자 알림을 사용하여AWS Glue Databrew 이벤트에 대한 알림을 받을 전송 채널을 설정할 수 있습니다. 이벤트가 지정한 규칙과 일치하면 알림을 받습니다. 이메일, 채팅 애플리케이션의 Amazon Q Developer 채팅 알림 또는 AWS Console Mobile Application 푸시 알림을 비롯한 여러 채널을 통해 이벤트에 대한 알림을 받을 수 있습니다. 콘솔 알림 센터에서도 알림을 볼 수 있습니다.AWS사용자 알림은 집계를 지원하므로 특정 이벤트 중에 수신하는 알림 수를 줄일 수 있습니다.

레시피 단계 및 함수 참조

이 참조에서는에서AWS CLI또는AWS SDKs. DataBrew에서 레시피 단계는 원시 데이터를 데이터 파이프라인에서 사용할 준비가 된 형식으로 변환하는 작업입니다. DataBrew 함수는 파라미터를 기반으로 계산을 수행하는 특수한 종류의 레시피 단계입니다.

UI의 변환 범주에는 다음이 포함됩니다.

- 기본 열 레시피 단계
  - 필터
  - 열
- 데이터 정리 레시피 단계
  - 형식
  - 친환경
  - Extract
- 데이터 품질 레시피 단계
  - 누락됨
  - 잘못됨
  - Duplicates
  - 이상치
- 개인 식별 정보(PII) 레시피 단계
  - 개인 정보 마스킹
  - 개인 정보 교체
  - 개인 정보 암호화
  - 행 섞기
- 열 구조 레시피 단계
  - 분할
  - 병합
  - 생성
- 열 형식 지정 레시피 단계
  - 십진수 정밀도
  - 수천 개의 구분자

  - 약어 숫자
- 데이터 구조 레시피 단계
  - 중첩-중첩 해제
  - Pivot(피벗)
  - Group
  - 조인
  - 결합
- 데이터 과학 레시피 단계
  - 텍스트
  - 규모 조정
  - 매핑
  - 인코딩
- 함수
  - 수학 함수
  - 집계 함수
  - 텍스트 함수
  - 날짜 및 시간 함수
  - 윈도우 함수
  - 웹 함수
  - 기타 함수

레시피에서 이러한 레시피 단계 및 함수를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요(조건 표현식 사용 포함). 레시피 구조 정의

다음 섹션에서는 레시피 단계와 함수를 작업별로 정리하여 설명합니다.

주제

- 기본 열 레시피 단계
- 데이터 정리 레시피 단계
- 데이터 품질 레시피 단계
- 개인 식별 정보(PII) 레시피 단계
- 이상치 감지 및 처리 레시피 단계

기본 열 레시피 단계

이러한 기본 열 레시피 작업을 사용하여 데이터에 대한 간단한 변환을 수행합니다.

주제

- TO_STRING_COLUMN

변경_데이터_유형

기존 열의 데이터 유형을 변경합니다.

열 값을 새 유형으로 변환할 수 없는 경우 NULL로 대체됩니다. 이는 문자열 열이 정수 열로 변환될 때 발생할 수 있습니다. 예를 들어 문자열 "123"은 정수 123이 되지만 문자열 "ABC"는 숫자가 될 수 없으므로 NULL 값으로 대체됩니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 새 유형입니다. 다음 데이터 타입이 지원됩니다.
  - 바이트: 부호 있는 1바이트 정수. 숫자 범위는 -128~127입니다.
  - short: 2바이트 부호 있는 정수. 숫자 범위는 -32768~32767입니다.
  - int: 4바이트 부호 있는 정수. 숫자 범위는 -2147483648~2147483647입니다.
  - long: 부호 있는 8바이트 정수. 숫자 범위는 -9223372036854775808~9223372036854775807입니다.
  - 부동 소수점: 4바이트 단일 정밀도 부동 소수점 숫자입니다.
  - double: 8바이트 배정밀도 부동 소수점 숫자입니다.
  - 10진수: 최대 총 38자리 및 소수점 뒤 18자리의 부호 있는 10진수입니다.
  - string: 문자열 값입니다.
  - 부울: 부울 유형에는 `true` 및 `false` 또는 `yes` 및 `no`라는 두 가지 값 중 하나가 있습니다.
  - timestamp: 연도, 월, 일, 시간, 분 및 초 필드를 구성하는 값입니다.
  - 날짜: 연도, 월, 일 필드로 구성된 값입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "CHANGE_DATA_TYPE",
        "Parameters": {
            "sourceColumn": "columnName",
            "columnDataType": "boolean"
```

```
        }
    }
}
```

DELETE

데이터 세트에서 열을 제거합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DELETE",
        "Parameters": {
            "sourceColumn": "extra_data"
        }
    }
}
```

중복

이름이 다르지만 모든 데이터가 동일한 새 열을 생성합니다. 이전 열과 새 열은 모두 데이터 세트에 유지됩니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 중복 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DUPLICATE",
```

```
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "copy_of_last_name"
        }
    }
}
```

JSON_TO_STRUCTS

JSON 문자열을 정적 형식의 구조체로 변환합니다. 변환 중에 모든 JSON 객체의 스키마를 감지하고 병합하여 전체 JSON 문자열을 나타내는 가장 일반적인 스키마를 가져옵니다. “unnestLevel” 파라미터는 구조체로 변환할 JSON 객체의 수준을 지정합니다.

파라미터

- `sourceColumns` - 소스 열 목록입니다.
- `regexColumnSelector` – 열을 선택하는 정규식입니다.
- `removeSourceColumn` - 부울 값입니다. 그런 `true` 다음 소스 열을 제거하고, 그렇지 않으면 보관합니다.
- `unnestLevel` - 중첩을 해제할 레벨 수입니다.
- `conditionExpressions` - 조건 표현식입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "JSON_TO_STRUCTS",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2"
       }
    }
}
```

MOVE_AFTER

열을 다른 열 바로 뒤에 있는 위치로 이동합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- targetColumn - 다른 열의 이름입니다. 에서 지정한 열은에서 지정한 열 바로 뒤에 sourceColumn 이동합니다targetColumn.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOVE_AFTER",
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "height_cm"
        }
    }
}
```

MOVE_BEFORE

열을 다른 열 바로 앞의 위치로 이동합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- targetColumn - 다른 열의 이름입니다. 에서 지정한 열은에서 지정한 열 바로 뒤에 sourceColumn 이동합니다targetColumn.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOVE_BEFORE",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "weight_kg"
        }
    }
```

```
}
```

MOVE_TO_END

열을 데이터 세트의 끝 위치(마지막 열)로 이동합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_END",
        "Parameters": {
            "sourceColumn": "height_cm"
        }
    }
}
```

MOVE_TO_INDEX

열을 숫자로 지정된 위치로 이동합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetIndex` - 열의 새 위치입니다. 위치는 0으로 시작합니다. 예를 들어 1는 두 번째 열을 나타내고, 2는 세 번째 열을 나타냅니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_INDEX",
        "Parameters": {
```

```
            "sourceColumn": "nationality",
            "targetIndex": "5"
        }
    }
}
```

MOVE_TO_START

열을 데이터 세트의 시작 위치(첫 번째 열)로 이동합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOVE_TO_START",
        "Parameters": {
            "sourceColumn": "first_name"
        }
    }
}
```

RENAME

이름이 다르지만 모든 데이터가 동일한 새 열을 생성합니다. 그런 다음 데이터 세트에서 이전 열이 제거됩니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 열의 새 이름입니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "RENAME",
        "Parameters": {
            "sourceColumn": "date_of_birth",
            "targetColumn": "birth_date"
        }
    }
}
```

SORT

데이터 세트의 하나 이상의 열에 있는 데이터를 오름차순, 내림차순 또는 사용자 지정 순서로 정렬합니다.

파라미터

- `expressions` - 정렬 표현식을 나타내는 하나 이상의 JSON 인코딩 문자열을 포함하는 문자열입니다.
  - `sourceColumn` - 기존 열의 이름을 포함하는 문자열입니다.
  - `ordering` - 주문은 ASCENDING 또는 DESCENDING일 수 있습니다.
  - `nullsOrdering` - Null 순서는 NULLS_TOP 또는 NULLS_BOTTOM이어서 열의 시작 또는 하단에 null 또는 누락된 값을 배치할 수 있습니다.
  - `customOrder` - 문자열 정렬에 대한 사용자 지정 순서를 정의하는 문자열 목록입니다. 기본적으로 문자열은 알파벳순으로 정렬됩니다.
  - `isCustomOrderCaseSensitive` – 부울입니다. 기본값은 `false`입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SORT",
        "Parameters": {
            "expressions": "[{\"sourceColumn\": \"A\", \"ordering\": \"ASCENDING\",
 \"nullsOrdering\": \"NULLS_TOP\"}]",
       }
    }
 }
```

Example사용자 지정 정렬 순서의 예

다음 예제에서 customOrder 표현식 문자열은 객체 목록의 형식을 갖습니다. 각 객체는 하나의 열에 대한 정렬 표현식을 설명합니다.

```
[
  {
    "sourceColumn": "A",
    "ordering": "ASCENDING",
    "nullsOrdering": "NULLS_TOP",
  },
  {
    "sourceColumn": "B",
    "ordering": "DESCENDING",
    "nullsOrdering": "NULLS_BOTTOM",
    "customOrder": ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    "isCustomOrderCaseSensitive": false,
  }
]
```

TO_BOOLEAN_COLUMN

기존 열의 데이터 형식을 BOOLEAN으로 변경합니다.

> **Note**
>
> TO_BOOLEAN_COLUMN 대신 CHANGE_DATA_TYPE 레시피 작업을 사용하는 것이 좋습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 값이 여야 합니다`boolean`.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "TO_BOOLEAN_COLUMN",
        "Parameters": {
            "columnDataType": "boolean",
            "sourceColumn": "is_present"
        }
    }
}
```

TO_DOUBLE_COLUMN

기존 열의 데이터 형식을 DOUBLE로 변경합니다.

> **Note**
>
> TO_DOUBLE_COLUMN 대신 CHANGE_DATA_TYPE 레시피 작업을 사용하는 것이 좋습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 값이 여야 합니다`number`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "TO_DOUBLE_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hourly_rate"
        }
    }
}
```

TO_NUMBER_COLUMN

기존 열의 데이터 형식을 NUMBER로 변경합니다.

> **Note**
>
> TO_NUMBER_COLUMN 대신 CHANGE_DATA_TYPE 레시피 작업을 사용하는 것이 좋습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 값이 여야 합니다`number`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "TO_NUMBER_COLUMN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "hours_worked"
        }
    }
}
```

TO_STRING_COLUMN

기존 열의 데이터 형식을 STRING으로 변경합니다.

> **Note**
>
> TO_STRING_COLUMN 대신 CHANGE_DATA_TYPE 레시피 작업을 사용하는 것이 좋습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 값이 여야 합니다`string`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "TO_STRING_COLUMN",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "age"
        }
    }
}
```

데이터 정리 레시피 단계

이러한 데이터 정리 레시피 단계를 사용하여 기존 데이터에 대해 간단한 변환을 수행합니다.

주제

- REPLACE_TEXT

대문자_대소문자

열의 각 문자열을 변경하여 각 단어를 대문자로 표시합니다. 대문자의 경우 각 단어의 첫 번째 문자는 대문자로 표시되고 나머지 단어는 소문자로 변환됩니다. 예를 들면 퀵 브라운 폭스가 울타리 위로 점프했습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "CAPITAL_CASE",
        "Parameters": {
            "sourceColumn": "last_name"
        }
    }
}
```

형식_날짜

날짜 문자열이 형식 값으로 변환되는 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetDateFormat` - 다음 날짜 형식 중 하나입니다.
  - `mm/dd/yyyy`
  - `mm-dd-yyyy`
  - `dd month yyyy`
  - `month yyyy`
  - `dd month`

Example예제

```
{
    "RecipeAction": {
        "Operation": "FORMAT_DATE",
        "Parameters": {
            "sourceColumn": "birth_date",
            "targetDateFormat": "mm-dd-yyyy"
        }
    }
}
```

LOWER_CASE

열의 각 문자열을 소문자로 변경합니다. 예를 들어 퀵 브라운 여우가 울타리 위로 점프했습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LOWER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

대문자

열의 각 문자열을 대문자로 변경합니다. 예: THE QUICK BROWN FOX JUMPED OVER THE FENCE

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UPPER_CASE",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

SENTENCE_CASE

열의 각 문자열을 문장 대/소문자로 변경합니다. 문장의 경우 각 문장의 첫 번째 문자는 대문자로 표시되고 나머지 문장은 소문자로 변환됩니다. 예는 빠른 갈색 여우입니다. 건너뜁니다. 울타리

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SENTENCE_CASE",
        "Parameters": {
            "sourceColumn": "description"
        }
    }
}
```

ADD_DOUBLE_QUOTES

열의 문자를 큰따옴표로 묶습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADD_DOUBLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_PREFIX

하나 이상의 문자를 추가하여 열의 시작 부분에 접두사로 연결합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `pattern` - 열 값의 시작 부분에 배치할 문자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADD_PREFIX",
        "Parameters": {
            "pattern": "aaa",
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_SINGLE_QUOTES

열의 문자를 작은따옴표로 묶습니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADD_SINGLE_QUOTES",
        "Parameters": {
            "sourceColumn": "info_url"
        }
    }
}
```

ADD_SUFFIX

열 끝에 접미사로 연결되는 문자를 하나 더 추가합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `pattern` - 열 끝에 배치할 문자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADD_SUFFIX",
        "Parameters": {
            "pattern": "bbb",
            "sourceColumn": "info_url"
        }
    }
}
```

EXTRACT_BETWEEN_DELIMITERS

기존 열의 값에서 구분 기호를 기반으로 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

- `targetColumn` - 생성할 새 열의 이름.
- `startPattern` - 구분된 값을 시작하는 문자를 나타내는 정규식입니다.
- `endPattern` - 구분 기호 문자 또는 구분 기호 값을 끝내는 문자를 나타내는 정규식입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": "\\/",
            "sourceColumn": "info_url",
            "startPattern": "\\/\\/",
            "targetColumn": "raw_url"
        }
    }
}
```

EXTRACT_BETWEEN_POSITIONS

기존 열의 값에서 문자 위치를 기반으로 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `startPosition` - 추출을 수행할 문자 위치입니다.
- `endPosition` - 추출을 종료할 문자 위치입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "9",
            "sourceColumn": "last_name",
```

```
            "startPosition": "3",
            "targetColumn": "characters_3_to_9"
        }
    }
}
```

EXTRACT_PATTERN

기존 열의 값에서 정규식을 기반으로 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `pattern` - 새 열을 추출하고 생성할 문자를 나타내는 정규식입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_PATTERN",
        "Parameters": {
            "pattern": "^....*...$",
            "sourceColumn": "last_name",
            "targetColumn": "first_and_last_few_characters"
        }
    }
}
```

EXTRACT_VALUE

사용자 지정 경로에서 추출된 값을 사용하여 새 열을 생성합니다. 소스 열이 Map, Array 또는 Struct 유형인 경우 경로의 각 필드는 역틱(예: `name`)을 사용하여 이스케이프 처리해야 합니다.

파라미터

- `targetColumn` - 대상 열의 이름입니다.
- `sourceColumn` - 값을 추출할 소스 열의 이름입니다.

- path - 사용자가 추출하려는 특정 키의 경로입니다. 소스 열이 Map, Array 또는 Struct 유형인 경우 경로의 각 필드는 역틱(예: `name`)을 사용하여 이스케이프 처리해야 합니다.

  사용자 정보의 다음 예를 고려하세요.

```
user {
   name: “Ammy”
   address: {
      state: "CA",
      zipcode: 12345
   },
   phoneNumber:{"home": "123123123", "work": "456456456"}
   citizenship: ["Canada", "USA", "Mexico", "India"]
}
```

  다음은 소스 열의 유형에 따라 제공할 경로의 예입니다.

- 소스 열이 유형 맵인 경우 집 전화번호를 추출하는 경로는 다음과 같습니다.

  `user`.`phoneNumber`.`home`
- 소스 열이 배열 유형인 경우 두 번째 '시민권' 값을 추출하는 경로는 다음과 같습니다.

  `user`.`citizenship`[1]
- 소스 열이 유형 구조체인 경우 우편번호를 추출하는 경로는 다음과 같습니다.

  `user`.`address`.`zipcode`

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_VALUE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "columnName",
            "path": "`age`.`name`",
        }
    }
```

```
}
```

REMOVE_COMBINED

사용자가 지정하는 내용에 따라 열에서 하나 이상의 문자를 제거합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- collapseConsecutiveWhitespace - true인 경우는 두 개 이상의 공백 문자를 정확히 하나의 공백 문자로 바꿉니다.
- removeAllPunctuation - true인 경우는 다음 문자를 모두 제거합니다. . ! , ?
- removeAllQuotes - true인 경우 작은따옴표와 큰따옴표를 모두 제거합니다.
- removeAllWhitespace - true인 경우 모든 공백 문자를 제거합니다.
- customCharacters - 조치를 취할 수 있는 하나 이상의 문자입니다.
- customValue - 조치를 취할 수 있는 값입니다.
- removeCustomCharacters - true인 경우 customCharacters 파라미터로 지정된 모든 문자를 제거합니다.
- removeCustomValue - true인 경우 customValue 파라미터로 지정된 모든 문자를 제거합니다.
- punctuationally - true인 경우 값의 시작 또는 끝에서 다음 문자가 발생하면가 제거합니다.. ! , ?
- antidisestablishmentarianism - true인 경우 값의 시작과 끝에서 작은따옴표와 큰따옴표를 제거합니다.
- removeLeadingAndTrailingWhitespace - true인 경우 값의 시작과 끝에서 모든 공백을 제거합니다.
- removeLetters - true인 경우 모든 대문자 및 소문자 알파벳 문자(A~Z, a ~z)를 제거합니다.
- removeNumbers - true인 경우 모든 숫자 문자(0~)를 제거합니다9.
- removeSpecialCharacters - true인 경우 다음 문자를 모두 제거합니다. ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "true",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "customCharacters": "¶",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "true",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "false",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
```

```
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "true",
            "customValue": "M",
            "removeAllPunctuation": "true",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "true",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "true",
            "removeLeadingAndTrailingWhitespace": "true",
            "removeLetters": "true",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "info_url"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

```
{
```

```
    "RecipeAction": {
        "Operation": "REMOVE_COMBINED",
        "Parameters": {
            "collapseConsecutiveWhitespace": "false",
            "removeAllPunctuation": "false",
            "removeAllQuotes": "false",
            "removeAllWhitespace": "false",
            "removeCustomCharacters": "false",
            "removeCustomValue": "false",
            "removeLeadingAndTrailingPunctuation": "false",
            "removeLeadingAndTrailingQuotes": "false",
            "removeLeadingAndTrailingWhitespace": "false",
            "removeLetters": "false",
            "removeNumbers": "true",
            "removeSpecialCharacters": "false",
            "sourceColumn": "first_name"
        }
    }
}
```

REPLACE_BETWEEN_DELIMITERS

두 구분 기호 사이의 문자를 사용자 지정 텍스트로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `startPattern` - 대체가 시작될 위치를 나타내는 문자 또는 문자 또는 정규 표현식입니다.
- `endPattern` - 대체가 끝날 위치를 나타내는 문자 또는 문자 또는 정규 표현식입니다.
- `value` - 대체 문자 또는 대체할 문자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_BETWEEN_DELIMITERS",
        "Parameters": {
            "endPattern": ">",
            "sourceColumn": "last_name",
            "startPattern": "&lt;",
```

```
            "value": "?"
        }
    }
}
```

REPLACE_BETWEEN_POSITIONS

두 위치 사이의 문자를 사용자 지정 텍스트로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `startPosition` - 대체가 시작될 문자열의 문자 위치를 나타내는 숫자입니다.
- `endPosition` - 대체가 종료될 문자열의 문자 위치를 나타내는 숫자입니다.
- `value` - 대체 문자 또는 대체할 문자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "20",
            "sourceColumn": "nationality",
            "startPosition": "10",
            "value": "E"
        }
    }
}
```

REPLACE_TEXT

지정된 문자 시퀀스를 다른 문자로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `pattern` - 소스 열에서 대체해야 하는 문자를 나타내는 문자 또는 문자 또는 정규 표현식입니다.

- `value` - 대체 문자 또는 대체할 문자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "x",
            "sourceColumn": "first_name",
            "value": "a"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPLACE_TEXT",
        "Parameters": {
            "pattern": "[0-9]",
            "sourceColumn": "nationality",
            "value": "!"
        }
    }
}
```

데이터 품질 레시피 단계

이러한 데이터 품질 레시피 단계를 사용하여 누락된 값을 채우거나, 잘못된 데이터를 제거하거나, 중복을 제거합니다.

주제

- AdvancedD_DATATYPE_FILTER
- ADVANCED_DATATYPE_FLAG
- DELETE_DUPLICATE_ROWS
- EXTRACT_ADVANCED_DATATYPE_DETAILS
- FILL_WITH_AVERAGE

- FILL_WITH_CUSTOM
- FILL_WITH_EMPTY
- FILL_WITH_LAST_VALID
- FILL_WITH_MEDIAN
- FILL_WITH_MODE
- FILL_WITH_MOST_FREQUENT
- FILL_WITH_NULL
- FILL_WITH_SUM
- FLAG_DUPLICATE_ROWS
- FLAG_DUPLICATES_IN_열
- GET_ADVANCED_DATATYPE
- REMOVE_DUPLICATES
- REMOVE_INVALID
- 제거_누락
- REPLACE_WITH_AVERAGE
- REPLACE_WITH_CUSTOM
- REPLACE_WITH_EMPTY
- REPLACE_WITH_LAST_VALID
- REPLACE_WITH_MEDIAN
- REPLACE_WITH_MODE
- REPLACE_WITH_MOST_FREQUENT
- REPLACE_WITH_NULL
- 평균_롤링_으로_대체
- REPLACE_WITH_ROLLING_SUM
- REPLACE_WITH_SUM

AdvancedD_DATATYPE_FILTER

고급 데이터 유형 감지를 기반으로 현재 소스 열을 필터링합니다. 예를 들어 DataBrew가 우편번호를 포함하는 것으로 식별한 열을 고려할 때이 변환은 시간대를 기준으로 열을 필터링할 수 있습니다. 추출할 수 있는 세부 정보는 아래 참고 사항에 설명된 대로 감지된 패턴에 따라 달라집니다.

파라미터

- `sourceColumn` - 문자열 소스 열의 이름입니다.
- `pattern` - 추출할 패턴입니다.
- `advancedDataType` - 전화, 우편 번호, 날짜 시간, 주, 신용 카드, URL, 이메일, SSN 또는 성별 중 하나일 수 있습니다.
- `filter values` - 사용자가 열을 필터링하려는 문자열 값의 목록입니다.
- `strategy` – KEEP_ROWS 또는 DISCARD_ROWS 또는 CLEAR_FILTERS 또는 CLEAR_OTHERS.
- `clearWithEmpty` - 부울 `true` 또는 `false`- `empty` 대신를 사용하여 행을 지웁니다`null`.

참고

- advancedDataType이 Phone인 경우 패턴은 AREA_CODE, TIME_ZONE 또는 COUNTRY_CODE일 수 있습니다.
- advancedDataType이 우편번호인 경우 패턴은 TIME_ZONE, COUNTRY, STATE, CITY, TYPE 또는 REGION일 수 있습니다.
- advancedDataType이 날짜 시간인 경우 패턴은 DAY, MONTH, MONTH_NAME, WEEK, QUARTER 또는 YEAR일 수 있습니다.
- advancedDataType이 상태인 경우 패턴은 TIME_ZONE일 수 있습니다.
- advancedDataType이 신용 카드인 경우 패턴은 LENGTH 또는 NETWORK일 수 있습니다.
- advancedDataType이 URL인 경우 패턴은 PROTOCOL, TLD 또는 DOMAIN일 수 있습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FILTER",
        "Parameters": {
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "strategy": "KEEP_ROWS"
        }
    }
```

```
}
```

ADVANCED_DATATYPE_FLAG

현재 소스 열의 값을 기반으로 새 플래그 열을 생성합니다. 예를 들어, 우편번호가 포함된 소스 열의 경우이 변환을 사용하여 값을 특정 시간대에 `false` 따라 `true` 또는 로 플래그를 지정할 수 있습니다. 추출할 수 있는 세부 정보는 아래 참고 사항에 설명된 대로 감지된 패턴에 따라 달라집니다.

파라미터

- `sourceColumn` - 문자열 소스 열의 이름입니다.
- `pattern` - 추출할 패턴입니다.
- `targetColumn` - 대상 열의 이름입니다.
- `advancedDataType` - 전화, 우편 번호, 날짜 시간, 주, 신용 카드, URL, 이메일, SSN 또는 성별 중 하나일 수 있습니다.
- `filter values` - 사용자가 열을 필터링하려는 문자열 값의 목록입니다.
- `trueString` - 대상 열의 `true` 값입니다.
- `falseString` - 대상 열의 `false` 값입니다.

참고

- advancedDataType이 Phone인 경우 패턴은 AREA_CODE, TIME_ZONE 또는 COUNTRY_CODE일 수 있습니다.
- advancedDataType이 우편번호인 경우 패턴은 TIME_ZONE, COUNTRY, STATE, CITY, TYPE 또는 REGION일 수 있습니다.
- advancedDataType이 날짜 시간인 경우 패턴은 DAY, MONTH, MONTH_NAME, WEEK, QUARTER 또는 YEAR일 수 있습니다.
- advancedDataType이 상태인 경우 패턴은 TIME_ZONE일 수 있습니다.
- advancedDataType이 신용 카드인 경우 패턴은 LENGTH 또는 NETWORK일 수 있습니다.
- advancedDataType이 URL인 경우 패턴은 PROTOCOL, TLD 또는 DOMAIN일 수 있습니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "ADVANCED_DATATYPE_FLAG",
        "Parameters": {
            "pattern": "AREA_CODE",
            "sourceColumn": "phoneColumn",
            "advancedDataType": "Phone",
            "filterValues": ['Ohio'],
            "targetColumn": "targetColumnName",
            "trueString": "trueValue",
            "falseString": "falseValue"
        }
    }
}
```

DELETE_DUPLICATE_ROWS

데이터 세트의 이전 행과 정확히 일치하는 행을 삭제합니다. 초기 발생은 이전 행과 일치하지 않으므로 삭제되지 않습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DELETE_DUPLICATE_ROWS"
    }
}
```

EXTRACT_ADVANCED_DATATYPE_DETAILS

고급 데이터 유형에 대한 세부 정보를 추출합니다. 추출할 수 있는 세부 정보는 아래 참고 사항에 설명된 대로 감지된 패턴에 따라 달라집니다.

파라미터

- `sourceColumn` - 문자열 소스 열의 이름입니다.
- `pattern` - 추출할 패턴입니다.
- `targetColumn` - 대상 열의 이름입니다.
- `advancedDataType` - 전화, 우편 번호, 날짜 시간, 주, 신용 카드, URL, 이메일, SSN 또는 성별 중 하나일 수 있습니다.

참고

- advancedDataType이 Phone인 경우 패턴은 AREA_CODE, TIME_ZONE 또는 COUNTRY_CODE일 수 있습니다.
- advancedDataType이 우편번호인 경우 패턴은 TIME_ZONE, COUNTRY, STATE, CITY, TYPE 또는 REGION일 수 있습니다.
- advancedDataType이 날짜 시간인 경우 패턴은 DAY, MONTH, MONTH_NAME, WEEK, QUARTER 또는 YEAR일 수 있습니다.
- advancedDataType이 상태인 경우 패턴은 TIME_ZONE일 수 있습니다.
- advancedDataType이 신용 카드인 경우 패턴은 LENGTH 또는 NETWORK일 수 있습니다.
- advancedDataType이 URL인 경우 패턴은 PROTOCOL, TLD 또는 DOMAIN일 수 있습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXTRACT_ADVANCED_DATATYPE_DETAILS",
        "Parameters": {
            "pattern": "TIMEZONE"
            "sourceColumn": "zipCode",
            "targetColumn": "timeZoneFromZipCode",
            "advancedDataType": "ZipCode"
        }
    }
}
```

FILL_WITH_AVERAGE

누락된 데이터가 모든 값의 평균으로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "FILL_WITH_AVERAGE",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_CUSTOM

누락된 데이터가 특정 값으로 대체된 열을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- columnDataType - 열의 데이터 유형입니다. 이 유형은 date, , number, booleanunsupported, string또는 이어야 합니다timestamp.
- value - 채울 사용자 지정 값입니다. 데이터 형식은에 대해 선택한 값과 일치해야 합니다columnDataType.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_CUSTOM",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "last_name",
            "value": "No last name provided"
        }
    }
}
```

FILL_WITH_EMPTY

누락된 데이터가 빈 문자열로 대체된 열을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_EMPTY",
        "Parameters": {
            "sourceColumn": "wind_direction"
        }
    }
}
```

FILL_WITH_LAST_VALID

누락된 데이터가 해당 열의 가장 최근 유효한 값으로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 유형입니다. 이 유형은 `date`, , `number`, `booleanunsupported`, `string`또는 이어야 합니다`timestamp`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "birth_date"
        }
    }
}
```

FILL_WITH_MEDIAN

누락된 데이터가 모든 값의 중앙값으로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MEDIAN",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FILL_WITH_MODE

누락된 데이터가 모든 값의 모드로 대체된 열을 반환합니다.

일부 값이 동일한 타이 브레이커 로직을 지정할 수도 있습니다. 예를 들어 다음 값을 고려합니다.

`1 2 2 3 3 4`

modeType의는 2MINIMUMFILL_WITH_MODE를 모드 값으로 반환합니다. modeType가 인 경우 MAXIMUM모드는 3입니다. AVERAGE의 경우 모드는 2.5입니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- modeType - 데이터의 타이 값을 확인하는 방법. 이 값은 MINIMUM, NONE, AVERAGE또는 이어야 합니다MAXIMUM.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MODE",
        "Parameters": {
            "modeType": "MAXIMUM",
            "sourceColumn": "age"
        }
    }
```

```
}
```

FILL_WITH_MOST_FREQUENT

누락된 데이터가 가장 빈번한 값으로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_MOST_FREQUENT",
        "Parameters": {
            "sourceColumn": "position"
        }
    }
}
```

FILL_WITH_NULL

데이터 값이 null로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_NULL",
        "Parameters": {
            "sourceColumn": "rating"
        }
    }
}
```

FILL_WITH_SUM

누락된 데이터가 모든 값의 합계로 대체된 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FILL_WITH_SUM",
        "Parameters": {
            "sourceColumn": "age"
        }
    }
}
```

FLAG_DUPLICATE_ROWS

각 행에 지정된 값이 있는 새 열을 반환합니다.이 열은 해당 행이 데이터 세트의 이전 행과 정확히 일치하는지 여부를 나타냅니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

파라미터

- `trueString` - 행이 이전 행과 일치하는 경우 삽입할 값.
- `falseString` - 행이 고유할 경우 삽입할 값.
- `targetColumn` - 데이터세트에 삽입된 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATE_ROWS",
        "Parameters": {
            "trueString": "TRUE",
            "falseString": "FALSE",
```

```
            "targetColumn": "Flag"
        }
    }
}
```

FLAG_DUPLICATES_IN_열

행의 소스 열에 있는 값이 소스 열의 이전 행에 있는 값과 일치하는지 여부를 나타내는 각 행에 지정된 값이 있는 새 열을 반환합니다. 일치 항목이 발견되면 중복으로 플래그가 지정됩니다. 초기 발생은 이전 행과 일치하지 않으므로 플래그가 지정되지 않습니다.

파라미터

- `sourceColumn` - 소스 열의 이름.
- `targetColumn` - 대상 열의 이름.
- `trueString` - 소스 열 값이 해당 열의 이전 값을 복제할 때 대상 열에 삽입할 문자열.
- `falseString` - 소스 열 값이 해당 열의 이전 값과 다를 때 대상 열에 삽입할 문자열.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FLAG_DUPLICATES_IN_COLUMN",
        "Parameters": {
            "sourceColumn": "Name",
            "targetColumn": "Duplicate",
            "trueString": "TRUE",
            "falseString": "FALSE"
        }
    }
}
```

GET_ADVANCED_DATATYPE

문자열 열이 주어지면는 열의 고급 데이터 형식이 있는 경우 이를 식별합니다.

파라미터

- `columnName` - 문자열 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "GET_ADVANCED_DATATYPE",
        "Parameters": {
            "sourceColumn": "columnName"
        }
    }
}
```

REMOVE_DUPLICATES

선택한 소스 열에서 중복 값이 발생하는 경우 전체 행을 삭제합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REMOVE_DUPLICATES",
        "Parameters": {
            "sourceColumn": "nationality"
        }
    }
}
```

REMOVE_INVALID

해당 행의 열에서 잘못된 값이 발생하는 경우 전체 행을 삭제합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다.

- advancedDataType - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다string. DataBrew가 string 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REMOVE_INVALID",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "help_url"
        }
    }
}
```

제거_누락

지정된 열에 데이터가 누락되지 않은 행만 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REMOVE_MISSING",
        "Parameters": {
            "sourceColumn": "last_name"
        }
    }
}
```

REPLACE_WITH_AVERAGE

열의 각 잘못된 값을 다른 모든 값의 평균으로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다. 이 유형은 여야 합니다`number`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_AVERAGE",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "age"
        }
    }
}
```

REPLACE_WITH_CUSTOM

감지된 개체를 사용자 지정 값으로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `sourceColumns` - 기존 열 이름의 목록입니다.
- `columnDataType` - 열의 데이터 형식입니다.
- `value` - 잘못된 값을 대체하는 데 사용할 사용자 지정 값입니다.
- `advancedDataType` - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다`string`. DataBrew가 `string` 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

> **Note**
>
> `sourceColumn` 또는를 사용하지`sourceColumns`만 둘 다 사용하지는 않습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_CUSTOM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "",
            "sourceColumns": ["column1", "column2"],
            "value": 0
        }
    }
}
```

REPLACE_WITH_EMPTY

열의 각 잘못된 값을 빈 값으로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다.
- `advancedDataType` - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다`string`. DataBrew가 `string` 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_EMPTY",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "nationality"
        }
    }
}
```

REPLACE_WITH_LAST_VALID

열의 각 잘못된 값을 마지막으로 유효한 값으로 바꿉니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- columnDataType - 열의 데이터 형식입니다.
- advancedDataType - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다string. DataBrew가 string 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_LAST_VALID",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "rating"
        }
    }
}
```

REPLACE_WITH_MEDIAN

열의 각 잘못된 값을 다른 모든 값의 중앙값으로 바꿉니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- columnDataType - 열의 데이터 형식입니다. 이 유형은 이어야 합니다number.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MEDIAN",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
```

```
    }
}
```

REPLACE_WITH_MODE

열의 각 잘못된 값을 다른 모든 값의 모드로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다. 이 유형은 이어야 합니다`number`.
- `modeType` - 데이터의 타이 값을 확인하는 방법. 이 값은 `MINIMUM`, `NONE`, `AVERAGE`또는 이어야 합니다`MAXIMUM`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MODE",
        "Parameters": {
            "columnDataType": "number",
            "modeType": "MAXIMUM",
            "sourceColumn": "height_cm"
        }
    }
}
```

REPLACE_WITH_MOST_FREQUENT

열의 각 잘못된 값을 가장 빈번한 열 값으로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다.
- `advancedDataType` - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다`string`. DataBrew가 `string` 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_MOST_FREQUENT",
        "Parameters": {
            "columnDataType": "string",
            "sourceColumn": "wind_direction"
        }
    }
}
```

REPLACE_WITH_NULL

열의 각 잘못된 값을 null 값으로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다.
- `advancedDataType` - 데이터 형식이 인 열에서 DataBrew가 감지하는 특수 데이터 형식입니다`string`. DataBrew가 `string` 열 내에서 감지할 수 있는 유형에는 SSN, 이메일, 전화번호, 성별, 신용 카드, URL, IP 주소, DateTime, 통화, ZipCode, 국가, 리전, 주 및 도시가 포함됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_NULL",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "weight_kg"
        }
    }
}
```

평균_롤링_으로_대체

열의 각 값을 행의 이전 "창"의 롤링 평균으로 바꿉니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- columnDataType - 열의 데이터 형식입니다. 이 유형은 이어야 합니다number.
- period - - 창의 크기입니다. 예를 들어 period가 10인 경우 롤링 평균은 이전 10개 행을 사용하여 계산됩니다.

Example예제

```
{
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_AVERAGE",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
                "period": "2"
            }
        }
    }
}
```

REPLACE_WITH_ROLLING_SUM

열의 각 값을 행의 이전 "창"의 롤링 합계로 바꿉니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- columnDataType - 열의 데이터 형식입니다. 이 유형은 이어야 합니다number.
- period - - 창의 크기입니다. 예를 들어 period가 10인 경우 롤링 합계는 이전 10개의 행을 사용하여 계산됩니다.

Example예제

```
{
```

```
    "RecipeStep": {
        "Action": {
            "Operation": "REPLACE_WITH_ROLLING_SUM",
            "Parameters": {
                "sourceColumn": "created_at",
                "columnDataType": "number",
                "period": "2"
            }
        }
    }
}
```

REPLACE_WITH_SUM

열의 각 잘못된 값을 다른 모든 값의 합계로 바꿉니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `columnDataType` - 열의 데이터 형식입니다. 이 유형은 이어야 합니다`number`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_SUM",
        "Parameters": {
            "columnDataType": "number",
            "sourceColumn": "games_won"
        }
    }
}
```

개인 식별 정보(PII) 레시피 단계

다음 레시피 단계를 사용하여 데이터 세트의 개인 식별 정보(PII)에 대한 변환을 수행합니다.

> **Note**
>
> 이 섹션의 레시피 단계 외에도 PII를 처리하는 데 사용할 수 있는 PII용으로 특별히 설계되지 않은 DataBrew 레시피 단계가 있습니다. 열을 삭제하는 기본 열 레시피 단계DELETE인가 그 예입니다.

주제

- 암호화_해시
- 복호화
- DETERMINISTIC_DECRYPT
- DETERMINISTIC_ENCRYPT
- 암호화
- MASK_CUSTOM
- MASK_DATE
- MASK_DELIMITER
- MASK_RANGE
- REPLACE_WITH_RANDOM_BETWEEN
- REPLACE_WITH_RANDOM_DATE_BETWEEN
- SHUFFLE_ROWS

암호화_해시

열의 해시 값에 알고리즘을 적용합니다.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `secretId` – Secrets Manager 시크릿 키의 ARN. 소스 열을 해시하기 위해 해시 기반 메시지 인증 코드(HMAC) 접두사 알고리즘에 사용되는 키 또는 `databrew!default`는 Secrets Manager 보안 키 값에 대한 base64 디코딩된 출력입니다.
- `secretVersion` – 선택 사항입니다. 기본적으로 최신 시크릿 버전으로 설정됩니다.
- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.

- `createSecretIfMissing` - 선택적 부울. true인 경우 호출자를 대신하여 시크릿을 생성하려고 시도합니다.
- `algorithm` – 데이터를 해시하는 데 사용되는 알고리즘. 유효한 열거형 값: MD5, SHA1, SHA256, SHA512, HMAC_MD5, HMAC_SHA1, HMAC_SHA256, HMAC_SHA512

  각 옵션은 서로 다른 해싱 알고리즘을 참조합니다. "HMAC" 접두사가 있는 이러한 옵션은 키가 지정된 해싱 알고리즘을 참조하며 `secretId` 파라미터가 필요합니다. "HMAC" 접두사가 없는 옵션의 경우 `secretId` 파라미터가 필요하지 않습니다.

  해시 알고리즘을 제공하지 않으면 서비스는 기본적으로 "HMAC_SHA256"으로 설정됩니다.

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "entityTypeFilter": ["USA_ALL"]
}
```

대화형 환경에서 작업할 때는 프로젝트의 역할 외에도 콘솔 사용자에게 제공된 Secrets Manager 보안 암호에 `secretsmanager:GetSecretValue` 대한 권한이 있어야 합니다.

샘플 정책:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

secretId로 전달하고 파라미터를 true`databrew!default`로 전달하여 DataBrew에서 생성한 기본 보안 암호를 사용하도록 선택할 수도 `createSecretIfMissing` 있습니다. 프로덕션에는 권장되지 않습니다. AwsGlueDataBrewFullAccessPolicy 역할을 가진 사람은 누구나 기본 보안 암호를 사용할 수 있습니다.

복호화

DECRYPT 변환을 사용하여 DataBrew 내부를 복호화할 수 있습니다.AWS암호화 SDK를 사용하여 DataBrew 외부에서 데이터를 복호화할 수도 있습니다. 제공된 KMS 키 ARN이 열을 암호화하는 데 사용된 것과 일치하지 않으면 복호화 작업이 실패합니다.AWS암호화 SDK에 대한 자세한 내용은 AWS Encryption SDK개발자 안내서의AWS암호화 SDK란 무엇입니까?를 참조하세요.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `kmsKeyArn` - 소스 열을 복호화하는 데 사용할AWS Key Management Service 키의 키 ARN입니다. 키 ARN에 대한 자세한 내용은 AWS Key Management Service개발자 안내서의 키 ARN을 참조하세요.

```
{
   "sourceColumns": ["phonenumber"],
   "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/<kms-key-id>"
}
```

대화형 환경에서 작업할 때는 프로젝트의 역할 외에도 콘솔 사용자에게 제공된 KMS 키`kms:Decrypt`에 대한 `kms:GenerateDataKey` 및 권한이 있어야 합니다.

샘플 정책:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
```

```
            "kms:Decrypt"
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678901:key/kms-key-id"
        ]
    }
  ]
}
```

DETERMINISTIC_DECRYPT

DETERMINISTIC_ENCRYPT로 암호화된 데이터를 해독합니다.

제공된 보안 암호 ID 및 버전이 열을 암호화하는 데 사용된 것과 일치하지 않는 경우이 변환은 작동하지 않습니다.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `secretId` - 소스 열을 해독하는 데 사용할 Secrets Manager 보안 키의 ARN입니다.
- `secretVersion` – 선택 사항입니다. 기본적으로 최신 시크릿 버전으로 설정됩니다.

예제

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123"
}
```

대화형 환경에서 작업할 때 콘솔 사용자는 프로젝트의 역할 외에도 제공된 Secrets Manager 보안 암호에 대해 secretsmanager:GetSecretValue에 대한 권한이 있어야 합니다.

샘플 정책:

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

DETERMINISTIC_ENCRYPT

256비트 키가 있는 AES-GCM-SIV를 사용하여 열을 암호화합니다. DETERMINISTIC_ENCRYPT로 암호화된 데이터는 DETERMINISTIC_DECRYPT 변환을 사용하여 DataBrew 내부에서만 해독할 수 있습니다. 이 변환은AWS KMS또는AWS Encryption SDK를 사용하지 않고 대신 AWS LC github 라이브러리를 사용합니다.

셀당 최대 400KB를 암호화할 수 있습니다. 복호화 시 데이터 형식을 보존하지 않습니다.

> **Note**
>
> 참고: 보안 암호를 1년 이상 사용하지 않는 것이 좋습니다.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `secretId` - 소스 열 또는 databrew를 암호화하는 데 사용할 Secrets Manager 보안 키의 ARN입니다.기본값입니다.
- `secretVersion` – 선택 사항입니다. 기본적으로 최신 시크릴 버전으로 설정됩니다.
- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.
- `createSecretIfMissing` - 선택적 부울. true인 경우 호출자를 대신하여 시크릿을 생성하려고 시도합니다.

예제

```
{
   "sourceColumns": ["phonenumber"],
   "secretId": "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret",
   "secretVersion": "adfe-1232-7563-3123",
   "entityTypeFilter": ["USA_ALL"]
}
```

대화형 환경에서 작업할 때는 프로젝트의 역할 외에도 콘솔 사용자에게 제공된 Secrets Manager 보안 암호에 `secretsmanager:GetSecretValue` 대한 권한이 있어야 합니다.

샘플 정책

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-east-1:012345678901:secret:mysecret"
            ]
        }
    ]
}
```

암호화

AWS Encryption SDK를 사용하여 소스 열의 값을 암호화합니다. DECRYPT 변환을 사용하여 DataBrew 내부를 복호화할 수 있습니다.AWS암호화 SDK를 사용하여 DataBrew 외부의 데이터를 복호화할 수도 있습니다.

ENCRYPT 변환은 셀당 최대 128MiB를 암호화할 수 있습니다. 복호화할 때 형식을 보존하려고 시도합니다. 데이터 유형을 보존하려면 데이터 유형 메타데이터를 1KB 미만으로 직렬화해야 합니다. 그렇지 않으면 `preserveDataType` 파라미터를 false로 설정해야 합니다. 데이터 유형 메타데이터는

암호화 컨텍스트에서 일반 텍스트로 저장됩니다. 암호화 컨텍스트에 대한 자세한 내용은 AWS Key Management Service개발자 안내서의 암호화 컨텍스트를 참조하세요.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `kmsKeyArn` - 소스 열을 암호화하는 데 사용할AWS Key Management Service 키의 키 ARN입니다. 키 ARN에 대한 자세한 내용은 AWS Key Management Service개발자 안내서의 키 ARN을 참조하세요.
- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.
- `preserveDataType` - 선택적 부울. 기본값은 true입니다. false인 경우 데이터 유형이 저장되지 않습니다.

다음 예제에서 `entityTypeFilter` 및 `preserveDataType`는 선택 사항입니다.

예제

```
{
    "sourceColumns": ["phonenumber"],
    "kmsKeyArn": "arn:aws:kms:us-east-1:012345678901:key/kms-key-id",
    "entityTypeFilter": ["USA_ALL"],
    "preserveDataType": "true"
}
```

대화형 환경에서 작업할 때는 프로젝트의 역할 외에도 콘솔 사용자에게 제공된AWS KMS키에 `kms:GenerateDataKey` 대한 권한이 있어야 합니다.

샘플 정책:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey"
```

```
        ],
        "Resource": [
            "arn:aws:kms:us-east-1:012345678901:key/kms-key-id"
        ]
      }
    ]
}
```

MASK_CUSTOM

제공된 사용자 지정 값과 일치하는 문자를 마스킹합니다.

파라미터

- `sourceColumns` - 기존 열 이름의 목록입니다.
- `maskSymbol` - 지정된 문자를 대체하는 데 사용되는 기호입니다.
- `regex` - true인 경우는 일치시킬 정규식 패턴`customValue`으로 취급합니다.
- `customValue` -의 모든 발생(또는 정규식 일치)`customValue`은 문자열에서 마스킹됩니다.
- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.

Example예제

```
// Mask all occurrences of 'amazon' in the column
{
    "RecipeAction": {
        "Operation": "MASK_CUSTOM",
        "Parameters": {
            "sourceColumns": ["company"],
            "maskSymbol": "#",
            "customValue": "amazon"
        }
    }
}
```

MASK_DATE

사용자 지정 마스크 기호로 날짜의 구성 요소를 마스킹합니다.

파라미터

- `sourceColumns` - 기존 열 이름의 목록입니다.
- `maskSymbol` - 지정된 문자를 대체하는 데 사용되는 기호입니다.
- `redact` - 마스킹할 날짜 구성 요소 열거형의 배열입니다. 유효한 열거형 값: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, MILLISECOND.
- `locale` - 선택적 IETF BCP 47 언어 태그입니다. 기본값은 en입니다. 날짜 형식 지정에 사용할 로캘입니다.

Example예제

```
// Mask year
{
    "RecipeAction": {
        "Operation": "MASK_DATE",
        "Parameters": {
            "sourceColumns": ["birthday"],
            "maskSymbol": "#",
            "redact": ["YEAR"]
        }
    }
}
```

MASK_DELIMITER

사용자 지정 마스킹 기호를 사용하여 두 구분 기호 사이의 문자를 마스킹합니다.

파라미터

- `sourceColumns` - 기존 열 이름의 목록입니다.
- `maskSymbol` - 지정된 문자를 대체하는 데 사용되는 기호입니다.
- `startDelimiter` - 마스킹을 시작할 위치를 나타내는 문자입니다. 이 파라미터를 생략하면 문자열 시작부터 마스크가 적용됩니다.
- `endDelimiter` - 마스킹이 끝날 위치를 나타내는 문자입니다. 이 파라미터를 생략하면 startDelimiter에서 문자열 끝까지 마스킹이 적용됩니다.
- `preserveDelimiters` - true인 경우 구분 기호에 마스크를 적용합니다.

- `alphabet` - 마스킹 중에 보존할 문자 집합의 배열입니다. 유효한 열거형 값: SYMBOLS, WHITESPACE.
- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.

Example예제

```
// Mask string between '<' and '>', ignoring white spaces, symbols, and lowercase
 letters
{
    "RecipeAction": {
        "Operation": "MASK_DELIMITER",
        "Parameters": {
            "sourceColumns": ["name"],
            "maskSymbol": "#",
            "startDelimiter": "<",
            "endDelimiter": ">",
            "preserveDelimiters": false,
            "alphabet": ["WHITESPACE", "SYMBOLS"]
        }
    }
}
```

MASK_RANGE

사용자 지정 마스킹 기호를 사용하여 두 위치 사이의 문자를 마스킹합니다.

파라미터

- `sourceColumns` - 기존 열 이름의 목록입니다.
- `maskSymbol` - 지정된 문자를 대체하는 데 사용되는 기호입니다.
- `start` - 마스킹을 시작할 문자 위치를 나타내는 숫자입니다(0인덱스, 포함). 음수 인덱싱이 허용됩니다. 이 파라미터를 생략하면 문자열의 시작부터 '중지'까지 마스크가 적용됩니다.
- `stop` - 마스킹이 종료될 문자 위치를 나타내는 숫자입니다(0인덱스, 제외). 음수 인덱싱이 허용됩니다. 이 파라미터를 생략하면 'start'부터 문자열 끝까지 마스크가 적용됩니다.
- `alphabet` - 마스킹 중에 보존할 문자 집합 열거형 배열입니다. 유효한 열거형 값: SYMBOLS, WHITESPACE.

- `entityTypeFilter` - 엔터티 유형의 선택적 배열입니다. 자유 텍스트 열에서 탐지된 PII만 암호화하는 데 사용할 수 있습니다.

Example예제

```
// Mask entire string
{
    "RecipeAction": {
        "Operation": "MASK_RANGE",
        "Parameters": {
            "sourceColumns": ["firstName", "lastName"],
            "maskSymbol": "#"
        }
    }
}
```

REPLACE_WITH_RANDOM_BETWEEN

값을 난수로 바꿉니다.

파라미터

- `lowerBound` - 난수 범위의 하한입니다.
- `sourceColumns` - 기존 열 이름의 목록입니다.
- `upperBound` - 난수 범위의 상한입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_BETWEEN",
        "Parameters": {
            "lowerBound": "1",
            "sourceColumns": ["column1", "column2"],
            "upperBound": "100"
        }
    }
}
```

REPLACE_WITH_RANDOM_DATE_BETWEEN

값을 임의의 날짜로 바꿉니다.

파라미터

- `startDate` - 무작위 날짜를 가져올 날짜 범위의 시작입니다.
- `sourceColumns` - 기존 열 이름의 목록입니다.
- `endDate` - 무작위 날짜를 가져올 날짜 범위의 끝입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPLACE_WITH_RANDOM_DATE_BETWEEN",
        "Parameters": {
            "startDate": "2020-12-12 12:12:12",
            "sourceColumns": ["column1", "column2"],
            "endDate": "2021-12-12 12:12:12"
        }
    }
}
```

SHUFFLE_ROWS

지정된 열의 값을 셔플합니다. 셔플링은 보조 열별로 그룹화된 값으로 발생할 수 있습니다.

파라미터

- `sourceColumns` - 기존 열의 배열.
- `groupByColumns` - 셔플링 중에 소스 열을 그룹화하는 열 배열입니다.

Example예제

```
{
   "sourceColumns": ["age"],
   "*groupByColumns*": ["country"]
```

```
}
```

이상치 감지 및 처리 레시피 단계

이러한 레시피 단계를 사용하여 데이터에서 이상치로 작업하고 해당 이상치에 대해 고급 변환을 수행합니다.

주제

- FLAG_OUTLIERS
- REMOVE_OUTLIERS
- REPLACE_OUTLIERS
- RESCALE_OUTLIERS_WITH_Z_SCORE
- RESCALE_OUTLIERS_WITH_SKEW

FLAG_OUTLIERS

소스 열 값이 이상치인지 여부를 나타내는 각 행의 사용자 지정 가능한 값이 포함된 새 열을 반환합니다.

파라미터

- `sourceColumn` - 이상치를 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- `targetColumn` - 이상치 평가 전략의 결과를 삽입할 새 열의 이름을 지정합니다.
- `outlierStrategy` - 이상값을 감지하는 데 사용할 접근 방식을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - `Z_SCORE` - 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - `MODIFIED_Z_SCORE` - 중앙값을 절대 편차 임계값 이상으로 벗어날 때 값을 이상치로 식별합니다.
  - `IQR` - 열 데이터의 첫 번째 및 마지막 사분위수를 초과할 때 값을 특이값으로 식별합니다. 사분위수 범위(IQR)는 데이터 포인트의 중간 50%가 있는 위치를 측정합니다.
- `threshold` - 이상값을 감지할 때 사용할 임계값을 지정합니다. 로 계산된 점수가이 숫자를 `outlierStrategy` 초과하면 `sourceColumn` 값이 이상치로 식별됩니다. 기본값은 3입니다.
- `trueString` - 이상치가 감지되는 경우 사용할 문자열 값을 지정합니다. 기본값은 "True"입니다.
- `falseString` - 특이값이 감지되지 않는 경우 사용할 문자열 값을 지정합니다. 기본값은 "False"입니다.

다음 예제에서는 단일 RecipeAction 작업에 대한 구문을 표시합니다. 레시피에는 하나 이상의 RecipeStep 작업이 포함되고 레시피 단계에는 하나 이상의 레시피 작업이 포함됩니다. 레시피 작업은 지정한 데이터 변환을 실행합니다. 레시피 작업 그룹은 최종 데이터 세트를 생성하기 위해 순차적으로 실행됩니다.

JSON

다음은 JSON 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example JSON의 예

```
{
    "Action": {
        "Operation": "FLAG_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "IQR",
            "threshold": "1.5",
            "trueString": "Yes",
            "falseString": "No"
        }
    }
}
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

YAML

다음은 YAML 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example YAML의 예

```
- Action:
  Operation: FLAG_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
```

```
    outlierStrategy: IQR
    trueString: Outlier
    falseString: No
    threshold: '1.5'
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

REMOVE_OUTLIERS

파라미터의 설정에 따라 이상치로 분류되는 데이터 포인트를 제거합니다.

파라미터

- sourceColumn - 이상치를 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- outlierStrategy - 이상값을 감지하는 데 사용할 접근 방식을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - Z_SCORE - 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - MODIFIED_Z_SCORE - 중앙값을 절대 편차 임계값 이상으로 벗어날 때 값을 이상치로 식별합니다.
  - IQR - 열 데이터의 첫 번째 및 마지막 사분위수를 초과할 때 값을 특이값으로 식별합니다. 사분위수 범위(IQR)는 데이터 포인트의 중간 50%가 있는 위치를 측정합니다.
- threshold - 이상값을 감지할 때 사용할 임계값을 지정합니다. 로 계산된 점수가이 숫자를 outlierStrategy 초과하면 sourceColumn 값이 이상치로 식별됩니다. 기본값은 3입니다.
- removeType - 데이터를 제거하는 방법을 지정합니다. 유효한 값에는 DELETE_ROWS 및 CLEAR(이)가 있습니다.
- trimValue - 이상치의 전부 또는 일부를 제거할지 여부를 지정합니다. 이 부울 값은 기본적으로 입니다FALSE.
  - FALSE - 모든 이상값을 제거합니다.
  - TRUE - minValue 및에 지정된 백분위수 임계값을 벗어나는 이상값을 제거합니다maxValue.
- minValue - 이상치 범위의 최소 백분위수 값을 나타냅니다. 유효한 범위는 0~100입니다.
- maxValue - 이상치 범위의 최대 백분위수 값을 나타냅니다. 유효한 범위는 0~100입니다.

다음 예제에서는 단일 RecipeAction 작업에 대한 구문을 표시합니다. 레시피에는 하나 이상의 RecipeStep 작업이 포함되고 레시피 단계에는 하나 이상의 레시피 작업이 포함됩니다. 레시피 작업은

지정한 데이터 변환을 실행합니다. 레시피 작업 그룹은 최종 데이터 세트를 생성하기 위해 순차적으로 실행됩니다.

JSON

다음은 JSON 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example JSON의 예

```
{
    "Action": {
        "Operation": "REMOVE_OUTLIERS",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "removeType": "DELETE_ROWS",
            "trimValue": "TRUE",
            "minValue": "5",
            "maxValue": "95"
        }
    }
}
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

YAML

다음은 YAML 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example YAML의 예

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    outlierStrategy: Z_SCORE
```

```
    threshold: '3'
    removeType: DELETE_ROWS
    trimValue: 'TRUE'
    minValue: '5'
    maxValue: '95'
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

REPLACE_OUTLIERS

파라미터의 설정에 따라 이상치로 분류되는 데이터 포인트 값을 업데이트합니다.

파라미터

- sourceColumn - 이상치를 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- outlierStrategy - 이상값을 감지하는 데 사용할 접근 방식을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - Z_SCORE - 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - MODIFIED_Z_SCORE - 중앙값을 절대 편차 임계값 이상으로 벗어날 때 값을 이상치로 식별합니다.
  - IQR - 열 데이터의 첫 번째 및 마지막 사분위수를 초과할 때 값을 특이값으로 식별합니다. 사분위수 범위(IQR)는 데이터 포인트의 중간 50%가 있는 위치를 측정합니다.
- threshold - 이상값을 감지할 때 사용할 임계값을 지정합니다. 로 계산된 점수가이 숫자를 outlierStrategy 초과하면 sourceColumn 값이 이상치로 식별됩니다. 기본값은 3입니다.
- replaceType - 이상치를 교체할 때 사용할 방법을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - WINSORIZE_VALUES - 최소값 및 최대값 백분위수를 사용하여 값을 제한하도록 지정합니다.
  - REPLACE_WITH_CUSTOM
  - REPLACE_WITH_EMPTY
  - REPLACE_WITH_NULL
  - REPLACE_WITH_MODE
  - REPLACE_WITH_AVERAGE
  - REPLACE_WITH_MEDIAN
  - REPLACE_WITH_SUM
  - REPLACE_WITH_MAX

- modeType -가 replaceType 일 때 사용할 모달 함수의 유형을 나타냅니다REPLACE_WITH_MODE. 유효한 값에는 MIN, 및 MAX가 포함됩니다AVERAGE.
- minValue -를 trimValue 사용할 때 적용할 이상치 범위의 최소 백분위수 값을 나타냅니다. 유효한 범위는 0~100입니다.
- maxValue -를 trimValue 사용할 때 적용할 이상치 범위의 최대 백분위수 값을 나타냅니다. 유효한 범위는 0~100입니다.
- value - 사용 시 삽입할 값을 지정합니다REPLACE_WITH_CUSTOM.
- trimValue - 이상치의 전부 또는 일부를 제거할지 여부를 지정합니다. 이 부울 값은 replaceType이 REPLACE_WITH_NULL, REPLACE_WITH_MODE또는 일 TRUE 때 로 설정됩니다WINSORIZE_VALUES. 다른 모든 경우 기본값은 FALSE입니다.
  - FALSE - 모든 이상값을 제거합니다.
  - TRUE - minValue 및에 지정된 백분위수 상한 임계값을 벗어나는 이상값을 제거합니다maxValue.

다음 예제에서는 단일 RecipeAction 작업에 대한 구문을 표시합니다. 레시피에는 하나 이상의 RecipeStep 작업이 포함되고 레시피 단계에는 하나 이상의 레시피 작업이 포함됩니다. 레시피 작업은 지정한 데이터 변환을 실행합니다. 레시피 작업 그룹은 최종 데이터 세트를 생성하기 위해 순차적으로 실행됩니다.

JSON

다음은 JSON 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example JSON의 예

```
{
    "Action": {
        "Operation": "REPLACE_OUTLIERS",
        "Parameters": {
            "maxValue": "95",
            "minValue": "5",
            "modeType": "AVERAGE",
            "outlierStrategy": "Z_SCORE",
            "replaceType": "REPLACE_WITH_MODE",
            "sourceColumn": "name-of-existing-column",
            "threshold": "3",
```

```
            "trimValue": "TRUE"
        }
    }
}
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

YAML

다음은 YAML 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example YAML의 예

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    outlierStrategy: Z_SCORE
    threshold: '3'
    replaceType: REPLACE_WITH_MODE
    modeType: AVERAGE
    minValue: '5'
    maxValue: '95'
    trimValue: 'TRUE'
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

RESCALE_OUTLIERS_WITH_Z_SCORE

파라미터의 설정에 따라 각 행에서 크기가 조정된 특이값이 있는 새 열을 반환합니다. 또한이 작업은 Z 점수 정규화를 적용하여 평균(μ)이 0이고 표준 편차(™)가 1이 되도록 데이터 값을 선형으로 조정합니다. 이상값을 처리하려면이 작업을 수행하는 것이 좋습니다.

파라미터

- `sourceColumn` - 특이값을 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.

- `targetColumn` - 이상치를 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- `outlierStrategy` - 이상치를 감지하는 데 사용할 접근 방식을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - `Z_SCORE` - 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - `MODIFIED_Z_SCORE` - 중앙값을 절대 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - `IQR` - 열 데이터의 첫 번째 및 마지막 사분위수를 초과할 때 값을 특이값으로 식별합니다. 사분위수 범위(IQR)는 데이터 포인트의 중간 50%가 있는 위치를 측정합니다.
- `threshold` - 이상값을 감지할 때 사용할 임계값입니다. 로 계산된 점수가이 숫자를 `outlierStrategy` 초과하면 `sourceColumn` 값이 이상치로 식별됩니다. 기본값은 3입니다.

다음 예제에서는 단일 RecipeAction 작업에 대한 구문을 표시합니다. 레시피에는 하나 이상의 RecipeStep 작업이 포함되고 레시피 단계에는 하나 이상의 레시피 작업이 포함됩니다. 레시피 작업은 지정한 데이터 변환을 실행합니다. 레시피 작업 그룹은 최종 데이터 세트를 생성하기 위해 순차적으로 실행됩니다.

JSON

다음은 JSON 구문`RecipeAction`을 사용하여 DataBrew 레시피 작업에 `RecipeStep` 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example JSON의 예

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_Z_SCORE",
        "Parameters": {
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "outlierStrategy": "Z_SCORE",
            "threshold": "3"
        }
    }
}
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

YAML

다음은 YAML 구문RecipeAction을 사용하여 DataBrew 레시피 작업에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example YAML의 예

```
- Action:
  Operation: REMOVE_OUTLIERS
  Parameters:
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    outlierStrategy: Z_SCORE
    threshold: '3'
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

RESCALE_OUTLIERS_WITH_SKEW

파라미터의 설정에 따라 각 행에서 크기가 조정된 특이값이 있는 새 열을 반환합니다. 이 작업은 지정된 로그 또는 루트 변환을 적용하여 배포 왜도를 줄이는 데 효과적입니다. 왜곡된 데이터를 처리하려면 이 작업을 수행하는 것이 좋습니다.

파라미터

- sourceColumn - 특이값을 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- targetColumn - 이상치를 포함할 수 있는 기존 숫자 열의 이름을 지정합니다.
- outlierStrategy - 이상치를 감지하는 데 사용할 접근 방식을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - Z_SCORE - 평균에서 표준 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - MODIFIED_Z_SCORE - 중앙값을 절대 편차 임계값 이상으로 벗어날 때 값을 이상값으로 식별합니다.
  - IQR - 열 데이터의 첫 번째 및 마지막 사분위수를 초과할 때 값을 특이값으로 식별합니다. 사분위수 범위(IQR)는 데이터 포인트의 중간 50%가 있는 위치를 측정합니다.
- threshold - 이상값을 감지할 때 사용할 임계값을 지정합니다. 로 계산된 점수가이 숫자를 outlierStrategy 초과하면 sourceColumn 값이 이상치로 식별됩니다. 기본값은 3입니다.

- skewFunction - 이상치를 교체할 때 사용할 방법을 지정합니다. 유효 값에는 다음이 포함됩니다.
  - 로그 - 긍정 및 부정 왜곡을 줄이기 위해 강력한 변환을 적용합니다. 이는 자연 로그(2.718281828)입니다.
  - ROOT(포함) value = 3 - 상당히 강력한 변환을 적용하여 긍정 및 부정 왜곡을 줄입니다. (큐브 루트)
  - ROOT(사용) value = 2 - 중간 변환을 적용하여 양의 스큐만 줄입니다. (스퀘어 루트)
  - SQUARE - 중간 변환을 적용하여 음의 스큐를 줄입니다. (사각형)
  - 사용자 지정 변환 - value 파라미터에 제공된 사용자 지정 번호를 사용하여 지정된 LOG 또는 ROOT 변환을 적용합니다.
- value - 사용자 지정 변환에 사용할 값을 지정합니다. skewFunction가 LOG인 경우이 값은 로그의 기본을 나타냅니다. skewFunction가 ROOT인 경우이 값은 루트의 파워를 나타냅니다.

다음 예제에서는 단일 RecipeAction 작업에 대한 구문을 표시합니다. 레시피에는 하나 이상의 RecipeStep 작업이 포함되고 레시피 단계에는 하나 이상의 레시피 작업이 포함됩니다. 레시피 작업은 지정한 데이터 변환을 실행합니다. 레시피 작업 그룹은 최종 데이터 세트를 생성하기 위해 순차적으로 실행됩니다.

JSON

다음은 JSON 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example JSON의 예

```
{
    "Action": {
        "Operation": "RESCALE_OUTLIERS_WITH_SKEW",
        "Parameters": {
            "outlierStrategy": "Z_SCORE",
            "threshold": "3",
            "skewFunction": "ROOT",
            "sourceColumn": "name-of-existing-column",
            "targetColumn": "name-of-new-column",
            "value": "4"
        }
    }
}
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

YAML

다음은 YAML 구문RecipeAction을 사용하여 DataBrew 레시피에 RecipeStep 대한 예제의 멤버로 사용할 예제입니다. 레시피 작업 목록을 보여주는 구문 예제는 섹션을 참조하세요레시피 구조 정의.

Example YAML의 예

```
- Action:
  Operation: RESCALE_OUTLIERS_WITH_SKEW
  Parameters:
    outlierStrategy: Z_SCORE
    threshold: '3'
    skewFunction: ROOT
    sourceColumn: name-of-existing-column
    targetColumn: name-of-new-column
    value: '4'
```

API 작업에서이 레시피 작업을 사용하는 방법에 대한 자세한 내용은 CreateRecipe 또는 섹션을 참조하세요UpdateRecipe. 자체 코드에서 이러한 작업과 기타 API 작업을 사용할 수 있습니다.

열 구조 레시피 단계

이러한 열 구조 레시피 단계를 사용하여 데이터의 열 구조를 수정합니다.

주제

- BOOLEAN_OPERATION
- CASE_OPERATION
- FLAG_COLUMN_FROM_NULL
- FLAG_COLUMN_FROM_PATTERN
- MERGE
- SPLIT_COLUMN_BETWEEN_DELIMITER
- SPLIT_COLUMN_BETWEEN_POSITIONS
- SPLIT_COLUMN_FROM_END
- SPLIT_COLUMN_FROM_START

- SPLIT_COLUMN_MULTIPLE_DELIMITER
- SPLIT_COLUMN_SINGLE_DELIMITER
- SPLIT_COLUMN_WITH_INTERVALS

BOOLEAN_OPERATION

논리적 조건 IF의 결과를 기반으로 새 열을 생성합니다. 부울 표현식이 true이면 true 값을 반환하고, 부울 표현식이 false이면 false 값을 반환하거나 사용자 지정 값을 반환합니다.

파라미터

- `trueValueExpression` - 조건이 충족될 때의 결과입니다.
- `falseValueExpression` - 조건이 충족되지 않을 때의 결과입니다.
- `valueExpression` - 부울 조건입니다.
- `withExpressions` - 집계 결과에 대한 구성입니다.
- `targetColumn` - 새로 생성된 열의 이름.

trueValueExpression, falseValueExpression 및 valueExpression에서 상수 값, 열 참조 및 집계 결과를 사용할 수 있습니다.

Example예: 상수 값

숫자 또는 문장과 같이 변경되지 않은 값입니다.

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example예: 열 참조

데이터 세트의 열인 값입니다.

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`column.2`",
        "falseValueExpression": "`column.3`",
        "valueExpression": "`column.1` < `column.4`",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example예: 결과 집계

집계 함수에 의해 계산되는 값입니다. 집계 함수는 열에 대한 계산을 수행하고 단일 값을 반환합니다.

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "`:mincolumn.2`",
        "falseValueExpression": "`:maxcolumn.3`",
        "valueExpression": "`column.1` < `:avgcolumn.4`",
        "withExpressions": "[{\"name\":\"mincolumn.2\",\"value\":\"min(`column.2`)\",
\"type\":\"aggregate\"},{\"name\":\"maxcolumn.3\",\"value\":\"max(`column.3`)\",\"type
\":\"aggregate\"},{\"name\":\"avgcolumn.4\",\"value\":\"avg(`column.4`)\",\"type\":
\"aggregate\"}]",
        "targetColumn": "result.column"
      }
    }
  }
}
```

사용자는 이스케이프를 통해 JSON을 문자열로 변환해야 합니다.

trueValueExpression, falseValueExpression 및 valueExpression의 파라미터 이름은 withExpressions의 이름과 일치해야 합니다. 일부 열의 집계 결과를 사용하려면 해당 열에 대한 파라미터를 생성하고 집계 함수를 제공해야 합니다.

Example예제:

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000",
        "targetColumn": "result.column"
      }
    }
  }
}
```

Example예: 및/또는

및 및 또는를 사용하여 여러 조건을 결합할 수 있습니다.

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
        "trueValueExpression": "It is true.",
        "falseValueExpression": "It is false.",
        "valueExpression": "`column.1` < 2000 and `column.2` >= `column.3",
        "targetColumn": "result.column"
      }
    }
  }
}
{
  "RecipeStep": {
    "Action": {
      "Operation": "BOOLEAN_OPERATION",
      "Parameters": {
```

```
        "trueValueExpression": "`column.4`",
        "falseValueExpression": "`column.5`",
        "valueExpression": "startsWith(`column1`, 'value1') or endsWith(`column2`,
 'value2')",
        "targetColumn": "result.column"
      }
    }
  }
}
```

유효한 집계 함수

아래 표에는 부울 작업에 사용할 수 있는 유효한 집계 함수가 모두 나와 있습니다.

열 유형	조건	valueExpression	withExpressions	반환 값
Numeric	Sum	`:sum.column.1`	`[ { "name": "sum.column.1", "value": "sum(`column.1`)", "type": "aggregate" } ]`	의 합계를 반환합니다. column.1
	평균	`:mean.column.1`	`[ { "name": "mean.column.1", "value": "avg(`col`	의 평균을 반환합니다. column.1

열 유형	조건	valueExpression	withExpressions	반환 값
			umn.1`)", "type": "aggregat e" }]	
	평균 절대 편차	`:meanabs olutedevi ation.column.1`	[{ "name": "meanabso lutedevia tion.colu mn.1", "value": "mean_abs olute_dev iation(`c olumn.1`) ", "type": "aggregat e" }]	의 평균 절대 편차를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	중간	`:median.column.1`	`[ { "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" } ]`	의 중앙값을 반환합니다. `column.1`
	제품	`:product.column.1`	`[ { "name": "product.column.1", "value": "product(`column.1`)", "type": "aggregate" } ]`	의 제품을 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	표준 편차	`:standarddeviation.column.1`	[{ "name": "standard deviation.column.1", "value": "stddev(`column.1`)", "type": "aggregate" }]	의 표준 편차를 반환합니다. column.1
	분산	`:variance.column.1`	[{ "name": "variance.column.1", "value": "variance(`column.1`)", "type": "aggregate" }]	의 분산을 반환합니다. column.1

열 유형	조건	valueExpression	withExpressions	반환 값
	평균의 표준 오차	`:standarderrorofmean.column.1`	`[ { "name": "standarderrorofmean.column.1", "value": "standard_error_of_mean(`column.1`)", "type": "aggregate" } ]`	평균의 표준 오차를 반환합니다. `column.1`
	왜도	`:skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	의 왜도를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	쿠르트증	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	의 kurtosis를 반환합니다. `column.1`
Datetime/ Numeric/Text	개수	`:count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	의 총 행 수를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	고유한 수	`:countdistinct.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	에서 고유 행의 총 수를 반환합니다. `column.1`
	최소	`:min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	의 최소값을 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	최대	`:max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	의 최대값을 반환합니다. `column.1`

valueExpression의 유효한 조건

아래 표에는 지원되는 조건과 사용할 수 있는 값 표현식이 나와 있습니다.

열 유형	조건	valueExpression	설명
문자열	포함	contains(`column`, 'text')	열의 값에 텍스트가 포함되어 있는지 테스트하는 조건
	를 포함하지 않음	!contains(`column`, 'text')	열의 값이 텍스트를 포함하지 않는지 테스트하는 조건
	일치 항목	matches(`column`, 'pattern')	열의 값이 패턴과 일치하는지 테스트하는 조건

열 유형	조건	valueExpression	설명
	일치하지 않음	!matches(`column`, 'pattern')	열의 값이 패턴과 일치하지 않는지 테스트하는 조건
	다음으로 시작	startsWith(`column`, 'text')	열의 값이 텍스트로 시작하는지 테스트하는 조건
	로 시작하지 않음	!startsWith(`column`, 'text')	열의 값이 텍스트로 시작되지 않는지 테스트하는 조건
	다음으로 끝남	endsWith(`column`, 'text')	열의 값이 텍스트로 끝나는지 테스트하는 조건
	로 끝나지 않음	!endsWith(`column`, 'text')	열의 값이 텍스트로 끝나지 않는지 테스트하는 조건
Numeric	보다 작음	`column` < 숫자	열의 값이 숫자보다 작은지 테스트하는 조건
	작거나 같음	`column` <= 숫자	열의 값이 숫자보다 작거나 같은지 테스트하는 조건
	보다 큼	`column` > 숫자	열의 값이 숫자보다 큰지 테스트하는 조건
	크거나 같음	`column` >= 숫자	열의 값이 숫자보다 크거나 같은지 테스트하는 조건

열 유형	조건	valueExpression	설명
	사이에 있음	isBetween(`column`, minNumber, maxNumber)	열의 값이 minNumber와 maxNumber 사이에 있는지 테스트하는 조건
	사이에 있지 않음	!isBetween(`column`, minNumber, maxNumber)	열의 값이 minNumber와 maxNumber 사이에 있지 않은지 테스트하는 조건
부울	true입니다.	`column` = TRUE	열의 값이 부울 TRUE인지 테스트하는 조건
	false임	`column` = FALSE	열의 값이 부울 FALSE인지 테스트하는 조건
Date/Timestamp	이전	`column` < 'date'	열의 값이 날짜보다 이전인지 테스트하는 조건
	이하	`column` <= 'date'	열의 값이 날짜보다 빠르거나 같은지 테스트하는 조건
	보다 이후	`column` > 'date'	열의 값이 날짜 이후인지 테스트하는 조건
	보다 크거나 같음	`column` >= 'date'	열의 값이 날짜보다 크거나 같은지 테스트하는 조건
String/Numeric/Date/타임스탬프	정확히 입니다.	`column` = '값'	열의 값이 정확히 값인지 테스트하는 조건
	Is not	`column` != '값'	열의 값이 값이 아닌지 테스트하는 조건

열 유형	조건	valueExpression	설명
	누락됨	isMissing(`column`)	열의 값이 누락된 경우 테스트할 조건
	누락되지 않음	!isMissing(`column`)	열의 값이 누락되지 않았는지 테스트하는 조건
	유효함	isValid(`column`, datatype)	열의 값이 유효한지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)
	유효하지 않음	!isValid(`column`, datatype)	열의 값이 유효하지 않은지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)
중첩	누락됨	isMissing(`column`)	열의 값이 누락된 경우 테스트할 조건
	누락되지 않음	!isMissing(`column`)	열의 값이 누락되지 않았는지 테스트하는 조건
	유효함	isValid(`column`, datatype)	열의 값이 유효한지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)

열 유형	조건	valueExpression	설명
	유효하지 않음	!isValid(`column`, datatype)	열의 값이 유효하지 않은지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)

CASE_OPERATION

논리적 조건 CASE의 결과를 기반으로 새 열을 생성합니다. 사례 작업은 사례 조건을 거치며 첫 번째 조건이 충족되면 값을 반환합니다. 조건이 true이면 작업이 읽기를 중지하고 결과를 반환합니다. true인 조건이 없는 경우 기본값을 반환합니다.

파라미터

- `valueExpression` - 조건.
- `withExpressions` - 집계 결과에 대한 구성입니다.
- `targetColumn` - 새로 생성된 열의 이름입니다.

Example예제

```
{
  "RecipeStep": {
    "Action": {
      "Operation": "CASE_OPERATION",
      "Parameters": {
        "valueExpression": "case when `column11` < `column.2` then 'result1' when
 `column2` < 'value2' then 'result2' else 'high' end",
        "targetColumn": "result.column"
      }
    }
  }
}
```

유효한 집계 함수

아래 표에는 사례 작업에 사용할 수 있는 유효한 집계 함수가 모두 나와 있습니다.

열 유형	조건	valueExpression	withExpressions	반환 값
Numeric	Sum	`:sum.column.1`	`[ { "name": "sum.column.1", "value": "sum(`column.1`)", "type": "aggregate" } ]`	의 합계를 반환합니다. `column.1`
	평균	`:mean.column.1`	`[ { "name": "mean.column.1", "value": "avg(`column.1`)", "type": "aggregate" } ]`	의 평균을 반환합니다. `column.1`
	평균 절대 편차	`:meanabsolutedeviation.column.1`	`[ { "name":`	의 평균 절대 편차를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
			"meanabsolutedeviation.column.1", "value": "mean_absolute_deviation(`column.1`)", "type": "aggregate" }]	
	중간	`:median.column.1`	[{ "name": "median.column.1", "value": "median(`column.1`)", "type": "aggregate" }]	의 중앙값을 반환합니다. column.1

열 유형	조건	valueExpression	withExpressions	반환 값
	제품	`:product .column.1`	`[ { "name": "product. column.1", "value": "product( `column.1 `)", "type": "aggregat e" } ]`	의 제품을 반환합니다. `column.1`
	표준 편차	`:standar ddeviatio n.column.1`	`[ { "name": "standard deviation .column.1 ", "value": "stddev(` column.1` )", "type": "aggregat e" } ]`	의 표준 편차를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	분산	`:varianc e.column.1`	`[ { "name": "variance .column.1 ", "value": "variance (`column. 1`)", "type": "aggregat e" } ]`	의 분산을 반환합니다. `column.1`
	평균의 표준 오차	`:standar derrorofm ean.column.1`	`[ { "name": "standard errorofme an.column .1", "value": "standard _error_of _mean(`co lumn.1`)", "type": "aggregat e" } ]`	평균의 표준 오차를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	왜도	`:skewness.column.1`	`[ { "name": "skewness.column.1", "value": "skewness(`column.1`)", "type": "aggregate" } ]`	의 왜도를 반환합니다. `column.1`
	쿠르트증	`:kurtosis.column.1`	`[ { "name": "kurtosis.column.1", "value": "kurtosis(`column.1`)", "type": "aggregate" } ]`	의 kurtosis를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
Datetime/ Numeric/Text	개수	`:count.column.1`	`[ { "name": "count.column.1", "value": "count(`column.1`)", "type": "aggregate" } ]`	의 총 행 수를 반환합니다. `column.1`
	고유한 수	`:countdistinct.column.1`	`[ { "name": "count.column.1", "value": "count(distinct `column.1`)", "type": "aggregate" } ]`	에서 고유 행의 총 수를 반환합니다. `column.1`

열 유형	조건	valueExpression	withExpressions	반환 값
	최소	`:min.column.1`	`[ { "name": "min.column.1", "value": "min(`column.1`)", "type": "aggregate" } ]`	의 최소값을 반환합니다. `column.1`
	최대	`:max.column.1`	`[ { "name": "max.column.1", "value": "max(`column.1`)", "type": "aggregate" } ]`	의 최대값을 반환합니다. `column.1`

valueExpression의 유효한 조건

아래 표에는 지원되는 조건과 사용할 수 있는 값 표현식이 나와 있습니다.

열 유형	조건	valueExpression	설명
문자열	포함	contains(`column`, 'text')	열의 값에 텍스트가 포함되어 있는지 테스트하는 조건
	를 포함하지 않음	!contains(`column`, 'text')	열의 값이 텍스트를 포함하지 않는지 테스트하는 조건
	일치 항목	matches(`column`, 'pattern')	열의 값이 패턴과 일치하는지 테스트하는 조건
	일치하지 않음	!matches(`column`, 'pattern')	열의 값이 패턴과 일치하지 않는지 테스트하는 조건
	다음으로 시작	startsWith(`column`, 'text')	열의 값이 텍스트로 시작하는지 테스트하는 조건
	로 시작하지 않음	!startsWith(`column`, 'text')	열의 값이 텍스트로 시작되지 않는지 테스트하는 조건
	다음으로 끝남	endsWith(`column`, 'text')	열의 값이 텍스트로 끝나는지 테스트하는 조건
	로 끝나지 않음	!endsWith(`column`, 'text')	열의 값이 텍스트로 끝나지 않는지 테스트하는 조건
Numeric	보다 작음	`column` < 숫자	열의 값이 숫자보다 작은지 테스트하는 조건

열 유형	조건	valueExpression	설명
	작거나 같음	`column` <= 숫자	열의 값이 숫자보다 작거나 같은지 테스트하는 조건
	보다 큼	`column` > 숫자	열의 값이 숫자보다 큰지 테스트하는 조건
	크거나 같음	`column` >= 숫자	열의 값이 숫자보다 크거나 같은지 테스트하는 조건
	사이에 있음	isBetween(`column`, minNumber, maxNumber)	열의 값이 minNumber와 maxNumber 사이에 있는지 테스트하는 조건
	사이에 있지 않음	!isBetween(`column`, minNumber, maxNumber)	열의 값이 minNumber와 maxNumber 사이에 있지 않은지 테스트하는 조건
부울	true입니다.	`column` = TRUE	열의 값이 부울 TRUE인지 테스트하는 조건
	false임	`column` = FALSE	열의 값이 부울 FALSE인지 테스트하는 조건
Date/Timestamp	이전	`column` < 'date'	열의 값이 날짜보다 이전인지 테스트하는 조건
	이하	`column` <= 'date'	열의 값이 날짜보다 빠르거나 같은지 테스트하는 조건

열 유형	조건	valueExpression	설명
	보다 이후	`column` > 'date'	열의 값이 날짜 이후인지 테스트하는 조건
	보다 크거나 같음	`column` >= 'date'	열의 값이 날짜보다 크거나 같은지 테스트하는 조건
String/Numeric/Date/타임스탬프	정확히 입니다.	`column` = '값'	열의 값이 정확히 값인지 테스트하는 조건
	Is not	`column` != '값'	열의 값이 값이 아닌지 테스트하는 조건
	누락됨	isMissing(`column`)	열의 값이 누락된 경우 테스트할 조건
	누락되지 않음	!isMissing(`column`)	열의 값이 누락되지 않았는지 테스트하는 조건
	유효함	isValid(`column`, datatype)	열의 값이 유효한지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)
	유효하지 않음	!isValid(`column`, datatype)	열의 값이 유효하지 않은지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)
중첩	누락됨	isMissing(`column`)	열의 값이 누락된 경우 테스트할 조건

열 유형	조건	valueExpression	설명
	누락되지 않음	!isMissing(`column`)	열의 값이 누락되지 않았는지 테스트하는 조건
	유효함	isValid(`column`, datatype)	열의 값이 유효한지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)
	유효하지 않음	!isValid(`column`, datatype)	열의 값이 유효하지 않은지 테스트하는 조건(값이 데이터 형식이거나 데이터 형식으로 변환할 수 있음)

FLAG_COLUMN_FROM_NULL

기존 열에 null 값이 있는지 여부에 따라 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름입니다.
- `flagType` - 로 설정해야 하는 값입니다`Null values`.
- `trueString` - 소스에서 null 값이 발견된 경우 새 열의 값입니다. 값을 지정하지 않을 경우 기본값은 `True`입니다.
- `falseString` - 소스에서 null이 아닌 값이 발견된 경우 새 열의 값입니다. 값을 지정하지 않을 경우 기본값은 `False`입니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_NULL",
        "Parameters": {
            "flagType": "Null values",
            "sourceColumn": "weight_kg",
            "targetColumn": "is_weight_kg_missing"
        }
    }
}
```

FLAG_COLUMN_FROM_PATTERN

기존 열에 사용자 지정 패턴이 있는지 여부에 따라 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름입니다.
- `flagType` - 로 설정해야 하는 값입니다`Pattern`.
- `pattern` - 평가할 패턴을 나타내는 정규식입니다.
- `trueString` - 소스에서 null 값이 발견된 경우 새 열의 값입니다. 값을 지정하지 않을 경우 기본값은 `True`입니다.
- `falseString` - 소스에서 null이 아닌 값이 발견된 경우 새 열의 값입니다. 값을 지정하지 않을 경우 기본값은 `False`입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FLAG_COLUMN_FROM_PATTERN",
        "Parameters": {
            "falseString": "No",
            "flagType": "Pattern",
            "pattern": "N.*",
            "sourceColumn": "wind_direction",
            "targetColumn": "northerly",
            "trueString": "yes"
        }
```

```
    }
}
```

MERGE

두 개 이상의 열을 새 열로 병합합니다.

파라미터

- `sourceColumns` - 병합할 하나 이상의 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `delimiter` - 대상 열에 표시할 값 사이의 선택적 구분자입니다.
- `targetColumn` - 생성할 병합된 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MERGE",
        "Parameters": {
            "delimiter": " ",
            "sourceColumns": "[\"first_name\",\"last_name\"]",
            "targetColumn": "Merged Column 1"
        }
    }
}
```

SPLIT_COLUMN_BETWEEN_DELIMITER

시작 및 끝 구분 기호에 따라 열을 세 개의 새 열로 분할합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `patternOption1` - 첫 번째 구분 기호를 나타내는 하나 이상의 문자를 나타내는 JSON 인코딩 문자열입니다.
- `patternOption2` - 두 번째 구분 기호를 나타내는 하나 이상의 문자를 나타내는 JSON 인코딩 문자열입니다.

- pattern - 데이터를 분할할 때 구분자로 사용할 하나 이상의 문자입니다.
- includeInSplit - true인 경우 새 열에 패턴을 포함하고, 그렇지 않으면 패턴이 삭제됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_DELIMITER",
        "Parameters": {
            "patternOption1": "{\"pattern\":\"H\",\"includeInSplit\":true}",
            "patternOption2": "{\"pattern\":\"M\",\"includeInSplit\":true}",
            "sourceColumn": "last_name"
        }
    }
}
```

SPLIT_COLUMN_BETWEEN_POSITIONS

지정한 오프셋에 따라 열을 세 개의 새 열로 분할합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- startPosition - 분할을 시작할 문자 위치입니다.
- endPosition - 분할이 종료되는 문자 위치입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_BETWEEN_POSITIONS",
        "Parameters": {
            "endPosition": "12",
            "sourceColumn": "last_name",
            "startPosition": "2"
        }
    }
```

```
}
```

SPLIT_COLUMN_FROM_END

열을 문자열 끝에서 오프셋된 두 개의 새 열로 분할합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `position` - 문자열의 오른쪽 끝에서 분할이 발생하는 문자 위치입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_END",
        "Parameters": {
            "position": "1",
            "sourceColumn": "nationality"
        }
    }
}
```

SPLIT_COLUMN_FROM_START

열을 문자열의 시작 부분부터 오프셋된 두 개의 새 열로 분할합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `position` - 문자열의 왼쪽 끝에서 분할이 발생하는 문자 위치입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_FROM_START",
```

```
        "Parameters": {
            "position": "1",
            "sourceColumn": "first_name"
        }
    }
}
```

SPLIT_COLUMN_MULTIPLE_DELIMITER

여러 구분 기호에 따라 열을 분할합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `patternOptions` - 분할 기준을 결정하는 하나 이상의 패턴을 나타내는 JSON 인코딩 문자열입니다.
- `pattern` - 데이터를 분할할 때 구분자로 사용할 하나 이상의 문자입니다.
- `limit` - 수행할 분할 수입니다. 최소값은 1이고 최대값은 20입니다.
- `includeInSplit` - true인 경우 새 열에 패턴을 포함하고, 그렇지 않으면 패턴이 삭제됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_MULTIPLE_DELIMITER",
        "Parameters": {
            "limit": "1",
            "patternOptions": "[{\"pattern\":\",\",\"includeInSplit\":true},{\"pattern
\":\" \",\"includeInSplit\":true}]",
            "sourceColumn": "description"
        }
    }
}
```

SPLIT_COLUMN_SINGLE_DELIMITER

특정 구분 기호에 따라 열을 하나 이상의 새 열로 분할합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- pattern - 데이터를 분할할 때 구분자로 사용할 하나 이상의 문자입니다.
- limit - 수행할 분할 수입니다. 최소값은 1이고 최대값은 20입니다.
- includeInSplit - true인 경우 새 열에 패턴을 포함하고, 그렇지 않으면 패턴이 삭제됩니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_SINGLE_DELIMITER",
        "Parameters": {
            "includeInSplit": "true",
            "limit": "1",
            "pattern": "/",
            "sourceColumn": "info_url"
        }
    }
}
```

SPLIT_COLUMN_WITH_INTERVALS

열을 n자 간격으로 분할합니다. 여기서 n을 지정합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- startPosition - 분할을 시작할 문자 위치입니다.
- interval - 다음 분할 전에 건너뛸 문자 수입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SPLIT_COLUMN_WITH_INTERVALS",
```

```
        "Parameters": {
            "interval": "4",
            "sourceColumn": "nationality",
            "startPosition": "1"
        }
    }
}
```

열 형식 지정 레시피 단계

열 형식 지정 레시피 단계를 사용하여 열의 데이터 형식을 변경합니다.

주제

- NUMBER_FORMAT
- FORMAT_PHONE_NUMBER

NUMBER_FORMAT

숫자 값이 형식이 지정된 문자열로 변환되는 열을 반환합니다.

파라미터

- `sourceColumn` – 문자열. 기존 열의 이름입니다.
- `decimalPlaces` - 정수입니다. 십진수 구분자 뒤의 자릿수 값입니다.
- `numericDecimalSeparator` – 문자열. 다음 값 중 하나는 십진수 구분자를 나타냅니다.
  - "."
  - ","
- `numericThousandSeparator` – 문자열. 다음 값 중 하나는 천 개의 구분 기호를 나타냅니다.
  - null입니다. 천 개의 구분 기호가 활성화되지 않았음을 나타냅니다.
  - ","
  - " "
  - "."
  - "\\"
- `numericAbbreviatedUnit` – 문자열. 약어 단위를 나타내는 다음 값 중 하나:
  - null입니다. 약어 단위가 활성화되지 않았음을 나타냅니다.

- “천 개”
- "백만"
- "10억"
- "조"

- numericUnitAbbreviation – 문자열. 다음 값 중 하나 또는 단위 약어를 나타내는 사용자 지정 값:
  - null입니다. 단위 약어가 활성화되지 않았음을 나타냅니다.
  -

약어 단위	옵션
수천	K, k, M, 천, 사용자 지정
Million	M, m, MM, 백만, 사용자 지정
Billion	B, bn, 십억, 사용자 지정
1조	T, tn, 조, 사용자 지정

Example예제

```
{
    "RecipeAction": {
        "Operation": "NUMBER_FORMAT",
        "Parameters": {
            "sourceColumn": "income",
            "decimalPlaces": "2",
            "numericDecimalSeparator": ".",
            "numericThousandSeparator": ",",
            "numericAbbreviatedUnit": "THOUSAND",
            "numericUnitAbbreviation": "K"
        }
    }
}
```

FORMAT_PHONE_NUMBER

전화번호 문자열이 형식 값으로 변환되는 열을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- phoneNumberFormat - 전화번호를 변환할 형식. 형식을 지정하지 않은 경우 기본값은 국제적으로 인정되는 표준 전화번호 형식(E.164)입니다. 유효 값에는 다음이 포함됩니다.
  - E164 (다음 기간 생략E)
- defaultRegion - 번호 자체에 국가 코드가 없는 경우 전화번호의 리전을 지정하는 두 개 또는 세 개의 대문자로 구성된 유효한 리전 코드. 최대 defaultRegion 또는 defaultRegionColumn 중 하나를 제공할 수 있습니다.
- defaultRegionColumn - 고급 데이터 유형의 열 이름입니다Country. 지정된 열의 리전 코드는 번호 자체에 국가 코드가 없는 경우 전화번호의 국가 코드를 결정하는 데 사용됩니다. 최대 defaultRegion 또는 defaultRegionColumn 중 하나를 제공할 수 있습니다.

참고

- 유효한 전화번호로 형식을 지정할 수 없는 입력은 수정되지 않은 상태로 유지됩니다.
- 기본 리전이 제공되지 않고 전화번호가 더하기 기호(+) 및 국가 통화 코드로 시작하지 않는 경우 전화번호의 형식이 지정되지 않습니다.

Example

예: 고정 기본 리전

```
{
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegion": "US"
        }
    }
}
```

예: 기본 리전 열 옵션

```
{
```

```
    "Action": {
        "Operation": "FORMAT_PHONE_NUMBER",
        "Parameters": {
            "sourceColumn": "Phone Number",
            "defaultRegionColumn": "Country Code"
        }
    }
}
```

데이터 구조 레시피 단계

이러한 레시피 단계를 사용하여 다양한 관점에서 데이터를 표로 작성하고 요약하거나 고급 함수를 수행할 수 있습니다.

주제

- NEST_TO_ARRAY
- NEST_TO_MAP
- NEST_TO_STRUCT
- UNNEST_ARRAY
- UNNEST_MAP
- UNNEST_STRUCT
- UNNEST_STRUCT_N
- GROUP_BY
- JOIN
- PIVOT
- SCALE
- 트랜스포지토리
- UNION
- UNPIVOT

NEST_TO_ARRAY

사용자가 선택한 열을 배열 값으로 변환합니다. 선택한 열의 순서는 결과 배열을 생성하는 동안 유지됩니다. 다양한 열 데이터 형식은 모든 열의 데이터 형식을 지원하는 공통 형식으로 형식 캐스팅됩니다.

파라미터

- `sourceColumns` - 소스 열의 목록입니다.
- `targetColumn` - 대상 열의 이름입니다.
- `removeSourceColumns` - 사용자가 선택한 소스 열을 제거할지 여부를 `false` 나타내는 `true` 또는 값을 포함합니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_ARRAY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_MAP

사용자가 선택한 열을 각각 열 이름을 나타내는 키와 행 값을 나타내는 값이 있는 키-값 페어로 변환합니다. 선택한 열의 순서는 결과 맵을 생성하는 동안 유지되지 않습니다. 다양한 열 데이터 형식은 모든 열의 데이터 형식을 지원하는 공통 형식으로 형식 캐스팅됩니다.

파라미터

- `sourceColumns` - 소스 열의 목록입니다.
- `targetColumn` - 대상 열의 이름입니다.
- `removeSourceColumns` - 사용자가 선택한 소스 열을 제거할지 여부를 `false` 나타내는 `true` 또는 값을 포함합니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "NEST_TO_MAP",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

NEST_TO_STRUCT

사용자가 선택한 열을 각각 열 이름을 나타내는 키와 행 값을 나타내는 값이 있는 키-값 페어로 변환합니다. 선택한 열의 순서와 각 열의 데이터 형식은 결과 구조체에서 유지됩니다.

파라미터

- `sourceColumns` - 소스 열의 목록입니다.
- `targetColumn` - 대상 열의 이름입니다.
- `removeSourceColumns` - 사용자가 선택한 소스 열을 제거할지 여부를 `false` 나타내는 `true` 또는 값을 포함합니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "NEST_TO_STRUCT",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "columnName",
            "removeSourceColumns": "true"
        }
    }
}
```

UNNEST_ARRAY

유형의 열을 새 열`array`로 중첩 해제합니다. 배열에 둘 이상의 값이 포함된 경우 각 요소에 해당하는 행이 생성됩니다. 이 함수는 배열 열의 한 수준만 중첩 해제합니다.

파라미터

- `sourceColumn` - 기존 열의 이름입니다. 이 열은 `struct` 유형이어야 합니다.
- `targetColumn` - 생성된 대상 열의 이름입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNNEST_ARRAY",
        "Parameters": {
            "sourceColumn": "address",
            "targetColumn": "address"
        }
    }
}
```

UNNEST_MAP

유형의 열을 중첩 해제map하고 키 및 값에 대한 열을 생성합니다. 키-값 페어가 두 개 이상인 경우 각 키 값에 해당하는 행이 생성됩니다. 이 함수는 맵 열의 한 수준만 중첩 해제합니다.

파라미터

- `sourceColumn` - 기존 열의 이름입니다. 이 열은 `struct` 유형이어야 합니다.
- `removeSourceColumn` - `true`인 경우 함수가 완료된 후 소스 열이 삭제됩니다.
- `targetColumn` - 제공된 경우 생성된 각 열은 이를 접두사로 시작합니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNNEST_MAP",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false",
            "targetColumn": "address"
        }
```

```
    }
}
```

UNNEST_STRUCT

유형의 열을 중첩 해제`struct`하고 구조체에 있는 각 키에 대한 열을 생성합니다. 이 함수는 구조체 수준 1만 중첩 해제합니다.

파라미터

- `sourceColumn` - 기존 열의 이름입니다. 이 열은 구조체 유형이어야 합니다.
- `removeSourceColumn` - `true`인 경우 함수가 완료된 후 소스 열이 삭제됩니다.
- `targetColumn` - 제공된 경우 생성된 각 열은 이를 접두사로 시작합니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT",
        "Parameters": {
            "sourceColumn": "address",
            "removeSourceColumn": "false"
            "targetColumn": "add"
        }
    }
}
```

UNNEST_STRUCT_N

유형의 선택한 열의 각 필드에 대해 새 열을 생성합니다`struct`.

예를 들어 다음과 같은 구문이 있습니다.

```
            user {
               name: “Ammy”
               address: {
                  state: "CA",
                  zipcode: 12345
               }
```

```
            }
```

이 함수는 3개의 열을 생성합니다.

user.name	user.address.state	user.address.zipcode
Ammy	CA	12345

파라미터

- `sourceColumns` - 소스 열의 목록입니다.
- `regexColumnSelector` - 중첩을 해제할 열을 선택하는 정규식입니다.
- `removeSourceColumn` - 부울 값입니다. true인 경우 소스 열을 제거하고, 그렇지 않으면 유지합니다.
- `unnestLevel` - 중첩을 해제할 레벨 수입니다.
- `delimiter` - 구분 기호는 새로 생성된 열 이름에서 구조체의 다양한 수준을 구분하는 데 사용됩니다. 예를 들어 구분 기호가 “/”인 경우 열 이름은 “user/address/state” 형식입니다.
- `conditionExpressions` - 조건 표현식입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNNEST_STRUCT_N",
        "Parameters": {
            "sourceColumns": "[\"address\"]",
            "removeSourceColumn": "true",
            "unnestLevel": "2",
            "delimiter": "/"
        }
    }
}
```

GROUP_BY

행을 하나 이상의 열로 그룹화한 다음 각 그룹에 집계 함수를 적용하여 데이터를 요약합니다.

파라미터

- sourceColumns - 각 그룹의 기반을 구성하는 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- groupByAggFunctions - 적용할 집계 함수 목록을 나타내는 JSON 인코딩 문자열입니다. (집계를 원하지 않는 경우를 지정합니다UNAGGREGATED.)
- useNewDataFrame - true인 경우 GROUP_BY의 결과를 프로젝트 세션에서 사용할 수 있게 되어 현재 내용을 대체합니다.

Example예제

```
[
  {
    "Action": {
      "Operation": "GROUP_BY",
      "Parameters": {
        "groupByAggFunctionOptions": "[{\"sourceColumnName\":\"all_votes\",
\"targetColumnName\":\"all_votes_count\",\"targetColumnDataType\":\"number\",
\"functionName\":\"COUNT\"}]",
        "sourceColumns": "[\"year\",\"state_name\"]",
        "useNewDataFrame": "true"
      }
    }
  }
]
```

JOIN

두 데이터 세트에서 조인 작업을 수행합니다.

파라미터

- joinKeys - 조인 키로 사용할 각 데이터 세트의 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- joinType - 수행할 조인의 유형입니다. | INNER_JOIN | LEFT_JOIN || RIGHT_JOIN | OUTER_JOIN LEFT_EXCLUDING_JOIN | 중 하나여야 합니다RIGHT_EXCLUDING_JOIN. OUTER_EXCLUDING_JOIN
- leftColumns - 현재 활성 데이터 세트의 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- rightColumns - 현재 데이터 세트에 조인할 다른(보조) 데이터 세트의 열 목록을 나타내는 JSON 인코딩 문자열입니다.

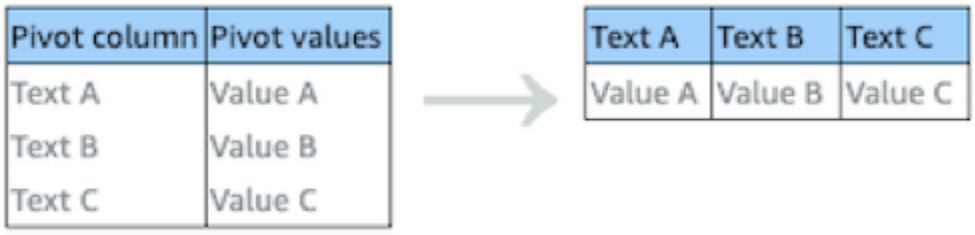
- `secondInputLocation` - 보조 데이터 세트의 데이터 파일로 확인되는 Amazon S3 URL입니다.
- `secondaryDatasetName` - 보조 데이터 세트의 이름입니다.

Example예제

```
{
    "Action": {
        "Operation": "JOIN",
        "Parameters": {
            "joinKeys": "[{\"key\":\"assembly_session\",\"value\":\"assembly_session
\"},{\"key\":\"state_code\",\"value\":\"state_code\"}]",
            "joinType": "INNER_JOIN",
            "leftColumns": "[\"year\",\"assembly_session\",\"state_code\",\"state_name
\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]",
            "rightColumns": "[\"assembly_session\",\"vote_id\",\"resolution\",
\"state_code\",\"state_name\",\"member\",\"vote\"]",
            "secondInputLocation": "s3://databrew-public-datasets-us-east-1/votes.csv",
            "secondaryDatasetName": "votes"
        }
    }
}
```

PIVOT

선택한 열의 모든 행 값을 값이 있는 개별 열로 변환합니다.



파라미터

- `sourceColumn` - 기존 열의 이름입니다. 열은 최대 10개의 고유 값을 가질 수 있습니다.
- `valueColumn` - 기존 열의 이름입니다. 열은 최대 10개의 고유 값을 가질 수 있습니다.

- aggregateFunction - 집계 함수의 이름입니다. 집계를 원하지 않는 경우 키워드를 사용합니다COLLECT_LIST.

Example예제

```
{
    "Action": {
        "Operation": "PIVOT",
        "Parameters": {
            "aggregateFunction": "SUM",
            "sourceColumn": "state_name",
            "valueColumn": "all_votes"
        }
    }
}
```

SCALE

숫자 열의 데이터 범위를 조정하거나 정규화합니다.

파라미터

- sourceColumn - 기존 열의 이름입니다.
- strategy - 열 값에 적용할 작업입니다.
  - MIN_MAX - 값을 [0,1] 범위로 다시 조정합니다.
  - SCALE_BETWEEN - 값을 지정된 두 값의 범위로 다시 조정합니다.
  - MEAN_NORMALIZATION - [-1, 1] 범위 내에서 평균(μ)이 0이고 표준 편차(™)가 1이 되도록 데이터의 크기를 조정합니다.
  - Z_SCORE - 평균(μ)이 0이고 표준 편차(™)가 1이 되도록 데이터 값을 선형으로 조정합니다. 이상 값을 처리하는 데 가장 적합합니다.
- targetColumn - 결과를 포함할 열의 이름입니다.

Example예제

```
{
    "Action": {
        "Operation": "NORMALIZATION",
```

```
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
    }
}
```

트랜스포지토리

선택한 모든 행을 열로 변환하고 열을 행으로 변환합니다.

Column 1	Column A	Column B	Column C
Row A	Value A	Value B	Value C
Row B	Value A1	Value B1	Value C1

New column	Row A	Row B
Column A	Value A	Value A1
Column B	Value B	Value B1
Column C	Value C	Value C1

파라미터

- `pivotColumns` - 행이 열 이름으로 변환될 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `valueColumns` - 행으로 변환할 하나 이상의 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `aggregateFunction` - 집계 함수의 이름입니다. 집계를 원하지 않는 경우 키워드를 사용합니다`COLLECT_LIST`.
- `newColumn` - 변환된 열을 값으로 보관할 열입니다.

Example예제

```
{
    "Action": {
        "Operation": "TRANSPOSE",
        "Parameters": {
            "pivotColumns": "[\"Teacher\"]",
            "valueColumns": "[\"Tom\",\"John\",\"Harry\"]",
```

```
            "aggregateFunction": "COLLECT_LIST",
            "newColumn": "Student"
        }
    }

}
```

UNION

둘 이상의 데이터 세트의 행을 단일 결과로 결합합니다.

파라미터

- `datasetsColumns` - 데이터 세트의 모든 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `secondaryDatasetNames` - 하나 이상의 보조 데이터 세트 목록을 나타내는 JSON 인코딩 문자열입니다.
- `secondaryInputs` - DataBrew에 보조 데이터세트(들)를 찾을 위치를 알려주는 Amazon S3 버킷 및 객체 키 이름 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumnNames` - 결과의 열 이름 목록을 나타내는 JSON 인코딩 문자열입니다.

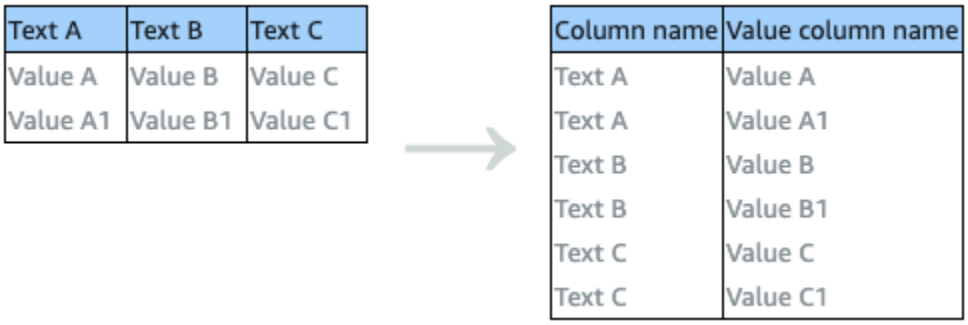
Example예제

```
{
    "Action": {
        "Operation": "UNION",
        "Parameters": {
            "datasetsColumns": "[[\"assembly_session\",\"state_code\",
\"state_name\",\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain
\",\"idealpoint_estimate\",\"affinityscore_usa\",\"affinityscore_russia\",
\"affinityscore_china\",\"affinityscore_india\",\"affinityscore_brazil\",
\"affinityscore_israel\"],[\"assembly_session\",\"state_code\",\"state_name
\",null,null,null,null,null,null,null,null,null,null,null,null]]",
            "secondaryDatasetNames": "[\"votes\"]",
            "secondaryInputs": "[{\"S3InputDefinition\":{\"Bucket\":\"databrew-public-
datasets-us-east-1\",\"Key\":\"votes.csv\"}}]",
            "targetColumnNames": "[\"assembly_session\",\"state_code\",\"state_name\",
\"year\",\"all_votes\",\"yes_votes\",\"no_votes\",\"abstain\",\"idealpoint_estimate
\",\"affinityscore_usa\",\"affinityscore_russia\",\"affinityscore_china\",
\"affinityscore_india\",\"affinityscore_brazil\",\"affinityscore_israel\"]"
        }
```

```
    }
}
```

UNPIVOT

선택한 행의 모든 열 값을 값이 있는 개별 행으로 변환합니다.



Text A	Text B	Text C
Value A	Value B	Value C
Value A1	Value B1	Value C1

Column name	Value column name
Text A	Value A
Text A	Value A1
Text B	Value B
Text B	Value B1
Text C	Value C
Text C	Value C1

파라미터

- `sourceColumns` - 피벗 해제할 하나 이상의 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `unpivotColumn` - 피벗 해제 작업의 값 열입니다.
- `valueColumn` - 피벗되지 않은 값을 저장할 열입니다.

Example예제

```
{
    "Action": {
        "Operation": "UNPIVOT",
        "Parameters": {
            "sourceColumns": "[\"idealpoint_estimate\"]",
            "unpivotColumn": "unpivoted_idealpoint_estimate",
            "valueColumn": "unpivoted_column_values"
        }
    }
}
```

데이터 과학 레시피 단계

이러한 레시피 단계를 사용하여 다양한 관점에서 데이터를 표로 작성하고 요약하거나 고급 변환을 수행할 수 있습니다.

주제

바이나리제이션

선택한 숫자 소스 열의 모든 값을 가져와 임계값과 비교하고 각 행에 대해 1 또는 0이 있는 새 열을 출력합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

  `targetColumn` - 생성할 새 열의 이름.

  `threshold` - 0 또는 1 값을 할당하기 위한 임계값을 나타내는 숫자입니다.

  `flip` - 더 낮은 값에 1이 할당되고 더 높은 값에 0이 할당되도록 이진 할당을 뒤집는 옵션입니다. 플립 파라미터가 true인 경우 임계값보다 작거나 같은 값은 1이 되고 임계값보다 큰 값은 0이 됩니다.

Example예제

```
{
    "Action": {
        "Operation": "BINARIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "threshold": "100.0",
            "flip": "false"
        }
```

```
    }
}
```

버킷화

버킷화(콘솔에서 비닝이라고 함)는 숫자 값 열의 항목을 가져와 숫자 범위로 정의된 빈으로 그룹화하고 각 행의 빈을 표시하는 새 열을 출력합니다. 버킷화는 분할 또는 백분율을 사용하여 수행할 수 있습니다. 아래 첫 번째 예제에서는 분할을 사용하고 두 번째 예제에서는 백분율을 사용합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

  `targetColumn` - 생성할 새 열의 이름.

  `bucketNames` - 버킷 이름 목록입니다.

  `splits` - 버킷 수준 목록입니다. 버킷은 연속적이며 버킷의 상한은 다음 버킷의 하한입니다.

  `percentage` - 각 버킷은 백분율로 설명됩니다.

Example분할 사용 예

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "bucketNames": "[\"Bin1\",\"Bin2\",\"Bin3\"]",
            "splits": "[\"-Infinity\",\"2\",\"20\",\"Infinity\"]"
        }
    }
}
```

Example백분율 사용 예

```
{
    "Action": {
        "Operation": "BUCKETIZATION",
```

```
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "bucketNames": "[\"Bin1\",\"Bin2\"]",
            "percentage": "50"
        }
    }
}
```

범주형_지도

하나 이상의 범주형 값을 숫자 또는 기타 값에 매핑합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

  `categoryMap` - 범주에 대한 값 맵을 나타내는 JSON 인코딩 문자열입니다.

  `deleteOtherRows` - `true`인 경우 매핑되지 않은 모든 행이 데이터 세트에서 제거됩니다.

  `other` - 제공된 경우 매핑되지 않은 모든 값이이 값으로 대체됩니다.

  `keepOthers` - true인 경우 매핑되지 않은 모든 값은 동일하게 유지됩니다.

  `mapType` - 매핑된 열의 데이터 유형입니다.

  `targetColumn` - 결과를 포함할 열의 이름입니다.

Example예제

```
{
    "Action": {
        "Operation": "CATEGORICAL_MAPPING",
        "Parameters": {
            "categoryMap": "{\"United States of America\":\"1\",\"Canada\":\"2\",\"Cuba
\":\"3\",\"Haiti\":\"4\",\"Dominican Republic\":\"5\"}",
            "deleteOtherRows": "false",
            "keepOthers": "true",
            "mapType": "NUMERIC",
            "sourceColumn": "state_name",
```

```
            "targetColumn": "state_name_mapped"
        }
    }
}
```

ONE_HOT_ENCODING

n개의 숫자 열을 생성합니다. 여기서 n은 선택한 범주형 변수의 고유 값 수입니다.

예를 들어 라는 열이 있다고 가정해 보겠습니다`shirt_size`. 의류는 스몰, 미디엄, 라지 또는 엑스트라 라지로 제공됩니다. 열 데이터는 다음과 같을 수 있습니다.

```
shirt_size
-----------
L
XL
M
S
M
M
S
XL
M
L
XL
M
```

이 시나리오에서는에 대해 네 가지 고유한 값이 있습니다`shirt_size`. 따라서는 4개의 새 열을 ONE_HOT_ENCODING 생성합니다. 각 새 열의 이름은 이며`shirt_size_`*x*, 여기서는 고유한 `shirt_size` 값을 *x* 나타냅니다.

`shirt_size` 및 4개의 생성된 열의 결과는 다음과 같습니다.

```
shirt_size    shirt_size_S    shirt_size_M    shirt_size_L    shirt_size_XL
------------    ------------    ------------    ------------    -------------
L              0               0               1               0
XL             0               0               0               1
M              0               1               0               0
S              1               0               0               0
M              0               1               0               0
M              0               1               0               0
```

```
S               1               0               0               0
XL              0               0               0               1
M               0               1               0               0
L               0               0               1               0
XL              0               0               0               1
M               0               1               0               0
```

에 지정하는 열에는 최대 열(10) 개의 고유 값이 있을 ONE_HOT_ENCODING 수 있습니다.

파라미터

- sourceColumn – 기존 열의 이름입니다. 열은 최대 10개의 고유 값을 가질 수 있습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ONE_HOT_ENCODING",
        "Parameters": {
            "sourceColumn": "shirt_size"
        }
    }
}
```

SCALE

숫자 열의 데이터 범위를 조정하거나 정규화합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- strategy - 열 값에 적용할 작업입니다.
  - MIN_MAX - 값을 [0,1] 범위로 재조정합니다.
  - SCALE_BETWEEN - 값을 2개의 지정된 값 범위로 다시 조정합니다.
  - MEAN_NORMALIZATION - [-1, 1] 범위 내에서 평균(μ)이 0이고 표준 편차(™)가 1이 되도록 데이터의 크기를 조정합니다.
  - Z_SCORE - 평균(μ)이 0이고 표준 편차(™)가 1이 되도록 데이터 값을 선형으로 조정합니다. 이상 값을 처리하는 데 가장 적합합니다.

- `targetColumn` - 결과를 포함할 열의 이름입니다.

Example예제

```
{
    "Action": {
        "Operation": "NORMALIZATION",
        "Parameters": {
            "sourceColumn": "all_votes",
            "strategy": "MIN_MAX",
            "targetColumn": "all_votes_normalized"
        }
    }
}
```

왜도

데이터 값에 변환을 적용하여 배포 셰이프와 스큐를 변경합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

  `targetColumn` - 생성할 새 열의 이름.

  `skewFunction`
  - `ROOT` - value-root를 추출합니다. `value` 파라미터에 루트를 제공할 수 있습니다.

    `LOG` - 로그 기본 값입니다. `value` 파라미터에 로그 베이스를 제공할 수 있습니다.

    `SQUARE` - 사각형 함수

  `value` - skewFunction의 인수입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SKEWNESS",
```

```
        "Parameters": {
            "sourceColumn": "level",
            "targetColumn": "bin",
            "skewFunction": "LOG",
            "value": "2.718281828"
        }
    }
}
```

토큰화

텍스트를 더 작은 단위 또는 개별 단어나 용어와 같은 토큰으로 분할합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `delimiter` - 토큰화된 단어 사이에 나타나는 사용자 지정 구분 기호입니다. (기본 동작은 각 토큰을 공백으로 구분하는 것입니다.)
- `expandContractions` - ENABLED인 경우는 계약된 단어를 확장합니다. 예: '하지 않음'은 '하지 않음'이 됩니다.
- `stemmingMode` - 텍스트를 개별 소문자 또는 용어와 같은 더 작은 단위 또는 토큰으로 분할합니다. | PORTER의 두 가지 어간 추출 모드를 사용할 수 있습니다LANCASTER.
- `stopWordRemovalMode` - a, a,와 같은 일반적인 단어를 제거합니다.
- `customStopWords` -의 경우 `StopWordRemovalMode`중지 단어의 사용자 지정 목록을 지정할 수 있습니다.
- `targetColumn` - 결과를 포함할 열의 이름입니다.

Example예제

```
{
    "Action": {
        "Operation": "TOKENIZATION",
        "Parameters": {
            "customStopWords": "[]",
            "delimiter": "- ",
            "expandContractions": "ENABLED",
            "sourceColumn": "dimensions",
```

```
            "stemmingMode": "PORTER",
            "stopWordRemovalMode": "DEFAULT",
            "targetColumn": "dimensions_tokenized"
        }
    }

}
```

수학 함수

아래에서 레시피 작업으로 작업하는 수학 함수에 대한 참조 주제를 찾습니다.

주제

- ABSOLUTE
- ADD
- CEILING
- DEGREES
- 분할
- 지수
- FLOOR
- IS_EVEN
- IS_ODD
- LN
- LOG
- MOD
- 곱하기
- 부정
- PI
- POWER
- RADIANS
- RANDOM
- RANDOM_BETWEEN

- ROUND
- SIGN
- SQUARE_ROOT
- 빼기

ABSOLUTE

새 열에 입력 번호의 절대값을 반환합니다. 절대값은 양수인지 음수인지에 관계없이 숫자가 0에서 얼마나 멀리 떨어져 있는지를 나타냅니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ABSOLUTE",
        "Parameters": {
            "sourceColumn": "freezingTemps",
            "targetColumn": "absValueOfFreezingTemps"
        }
    }
}
```

ADD

(+ `sourceColumn2`) 또는`sourceColumn1` (`sourceColumn1` +)를 사용하여 새 열의 입력 열 값을 합산합니다`value1`.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `value1` - 숫자 값입니다.

- `sourceColumn2` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ADD",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "sourceColumn2": "height_cm",
            "targetColumn": "weight_plus_height"
        }
    }
}
```

CEILING

새 열의 입력 십진수보다 크거나 같은 최소 정수 수를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value1` - 숫자 값입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "CEILING",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_CEILING"
        }
    }
}
```

DEGREES

각도의 라디안을 각도로 변환하고 결과를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DEGREES",
        "Parameters": {
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_DEGREES"
        }
    }
}
```

분할

한 입력 번호를 다른 입력 번호로 나누고 결과를 새 열에 반환합니다.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `value1` - 숫자 값입니다.
- `sourceColumn2` – 기존 열의 이름입니다.
- `value2` - 숫자 값입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "DIVIDE",
        "Parameters": {
            "sourceColumn1": "height_cm",
            "targetColumn": "divide_by_2",
            "value2": "2"
        }
    }
}
```

지수

새 열에서 Euler의 숫자를 n도로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXPONENT",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_EXPONENT"
        }
    }
}
```

FLOOR

새 열의 입력 번호보다 크거나 같은 가장 큰 정수를 반환합니다.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `value` - 숫자 값입니다.

- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "FLOOR",
        "Parameters": {
            "targetColumn": "FLOOR Column 1",
            "value": "42"
        }
    }
}
```

IS_EVEN

소스 열 또는 값이 균일한지 여부를 나타내는 부울 값을 새 열에 반환합니다. 소스 열 또는 값이 소수인 경우 결과는 false입니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `trueString` - 값이 짝수인지 여부를 나타내는 문자열.
- `falseString` - 값이 짝수가 아닌지 여부를 나타내는 문자열입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "IS_EVEN",
        "Parameters": {
            "falseString": "Value is odd",
            "sourceColumn": "height_cm",
            "targetColumn": "height_cm_IS_EVEN",
            "trueString": "Value is even"
        }
```

```
    }
}
```

IS_ODD

소스 열 또는 값이 홀수인지 여부를 나타내는 부울 값을 새 열에 반환합니다. 소스 열 또는 값이 소수인 경우 결과는 false입니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `trueString` - 값이 홀수인지 여부를 나타내는 문자열입니다.
- `falseString` - 값이 홀수가 아닌지 여부를 나타내는 문자열입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "IS_ODD",
        "Parameters": {
            "falseString": "Value is even",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_IS_ODD",
            "trueString": "Value is odd"
        }
    }
}
```

LN

새 열에 있는 값의 자연 로그(Euler 숫자)를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LN",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_LN"
        }
    }
}
```

LOG

새 열에 있는 값의 로그를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `base` - 로그의 기본입니다. 기본값은 10입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LOG",
        "Parameters": {
            "base": "10",
            "sourceColumn": "age",
            "targetColumn": "age_LOG"
        }
    }
}
```

MOD

한 숫자가 새 열에 있는 다른 숫자의 백분율을 반환합니다.

파라미터

- sourceColumn1 – 기존 열의 이름입니다.
- sourceColumn2 – 기존 열의 이름입니다.
- targetColumn - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MOD",
        "Parameters": {
            "sourceColumn1": "start_date",
            "sourceColumn2": "end_date",
            "targetColumn": "MOD Column 1"
        }
    }
}
```

곱하기

두 숫자를 곱하고 결과를 새 열로 반환합니다.

파라미터

- sourceColumn1 – 기존 열의 이름입니다.
- value1 - 숫자 값입니다.
- sourceColumn2 – 기존 열의 이름입니다.
- value2 - 숫자 값입니다.
- targetColumn - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MULTIPLY",
```

```
        "Parameters": {
            "sourceColumn1": "hourly_rate",
            "sourceColumn2": "hours",
            "targetColumn": "total_pay"
        }
    }
}
```

부정

값을 무효화하고 결과를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "NEGATE",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_NEGATE"
        }
    }
}
```

PI

새 열에 pi(3.141592653589793) 값을 반환합니다.

파라미터

- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "PI",
        "Parameters": {
            "targetColumn": "PI Column 1"
        }
    }
}
```

POWER

숫자의 값을 새 열에서 지수의 출력으로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 값을 높일 숫자입니다.
- `targetColumn` - 생성할 새 열의 이름.
- `exponent` - 값이 상승할 전력입니다.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "POWER",
        "Parameters": {
            "exponent": "3",
            "sourceColumn": "age",
            "targetColumn": "age_cubed"
        }
    }
}
```

RADIANS

각도를 라디안(180/pi로 나누기)으로 변환하고 새 열에 값을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "RADIANS",
        "Parameters": {
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_RADIANS"
        }
    }
}
```

RANDOM

새 열에 0에서 1 사이의 난수를 반환합니다.

파라미터

- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "RANDOM",
        "Parameters": {
            "targetColumn": "RANDOM Column 1"
        }
    }
}
```

RANDOM_BETWEEN

새 열에서는 지정된 하한(포함)과 지정된 상한(포함) 사이의 난수를 반환합니다.

파라미터

- `lowerBound` - 난수 범위의 하한입니다.
- `upperBound` - 난수 범위의 상한입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "RANDOM_BETWEEN",
        "Parameters": {
            "lowerBound": "1",
            "targetColumn": "RANDOM_BETWEEN Column 1",
            "upperBound": "100"
        }
    }
}
```

ROUND

숫자 값을 새 열의 가장 가까운 정수로 반올림합니다. 분율이 0.5 이상이면 반올림됩니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ROUND",
```

```
        "Parameters": {
            "sourceColumn": "rating",
            "targetColumn": "rating_ROUND"
        }
    }
}
```

SIGN

값이 0보다 작으면 -1, 값이 0이면 0, 값이 0보다 크면 +1로 새 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SIGN",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SIGN"
        }
    }
}
```

SQUARE_ROOT

새 열에 값의 제곱근을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SQUARE_ROOT",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_SQUARE_ROOT"
        }
    }
}
```

빼기

한 숫자를 다른 숫자에서 빼고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `value1` - 숫자 값입니다.
- `sourceColumn2` – 기존 열의 이름입니다.
- `value2` - 숫자 값입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SUBTRACT",
        "Parameters": {
            "sourceColumn1": "weight_kg",
            "targetColumn": "weight_minus_10_kg",
            "value2": "10"
        }
    }
}
```

집계 함수

아래에서 레시피 작업으로 작동하는 집계 함수에 대한 참조 주제를 찾습니다.

주제

ANY

새 열에서 선택한 소스 열의 모든 값을 반환합니다. 빈 값과 null 값은 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ANY",
        "Parameters": {
            "sourceColumns": "[\"age\",\"last_name\"]",
            "targetColumn": "ANY Column 1"
        }
```

```
    }
}
```

AVERAGE

소스 열에 있는 값의 평균을 계산하고 결과를 새 열에 반환합니다. 숫자가 아닌 모든는 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "AVERAGE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"weight_kg\",\"height_cm\"]",
            "targetColumn": "AVERAGE Column 1"
        }
    }
}
```

COUNT

새 열에서 선택한 소스 열의 값 수를 반환합니다. 빈 값과 null 값은 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
```

```
        "Operation": "COUNT",
        "Parameters": {
            "sourceColumns": "[\"ANY Column 1\",\"birth_date\",\"last_name\"]",
            "targetColumn": "COUNT Column 1"
        }
    }
}
```

COUNT_DISTINCT

새 열에서 선택한 소스 열의 고유 값 총 수를 반환합니다. 빈 값과 null 값은 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "COUNT_DISTINCT",
        "Parameters": {
            "sourceColumns": "[\"long_name\",\"weight_kg\"]",
            "targetColumn": "COUNT_DISTINCT Column 1"
        }
    }
}
```

KTH_LARGEST

새 열의 선택한 소스 열에서 k번째로 큰 숫자를 반환합니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.
- `value` - k를 나타내는 숫자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST",
        "Parameters": {
            "sourceColumns": "[\"height_cm\",\"weight_kg\",\"age\"]",
            "targetColumn": "KTH_LARGEST Column 1",
            "value": "2"
        }
    }
}
```

KTH_LARGEST_UNIQUE

새 열의 선택한 소스 열에서 k번째로 큰 고유 번호를 반환합니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

  `value` - k를 나타내는 숫자입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "KTH_LARGEST_UNIQUE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "KTH_LARGEST_UNIQUE Column 1",
            "value": "3"
        }
    }
}
```

MAX

새 열에서 선택한 소스 열의 최대 숫자 값을 반환합니다. 숫자가 아닌 모든는 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MAX",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MAX Column 1"
        }
    }
}
```

MEDIAN

새 열의 선택한 소스 열에서 정렬된 숫자 그룹의 중간 수인 중앙값을 반환합니다. 숫자가 아닌 모든는 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MEDIAN",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "MEDIAN Column 1"
        }
    }
}
```

MIN

새 열에서 선택한 소스 열의 최소값을 반환합니다. 숫자가 아닌 모든는 무시됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MIN",
        "Parameters": {
            "sourceColumns": "[\"age\",\"height_cm\",\"weight_kg\"]",
            "targetColumn": "MIN Column 1"
        }
    }
}
```

MODE

새 열의 선택한 소스 열에서 가장 자주 나타나는 숫자인 모드를 반환합니다. 숫자가 아닌 모든는 무시됩니다. 여러 모드의 경우 모드는 모달 함수를 사용하여 계산됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MODE",
        "Parameters": {
```

```
            "modeType": "MINIMUM",
            "sourceColumns": "[\"years_in_service\",\"age\"]",
            "targetColumn": "MODE Column 1"
        }
    }
}
```

STANDARD_DEVIATION

새 열에서 선택한 소스 열의 표준 편차를 반환합니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STANDARD_DEVIATION",
        "Parameters": {
            "sourceColumns": "[\"years_in_sservice\",\"age\"]",
            "targetColumn": "STANDARD_DEVIATION Column 1"
        }
    }
}
```

SUM

새 열에서 선택한 소스 열의 값 합계를 반환합니다. 숫자가 아닌 모든는 0으로 처리됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SUM",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "SUM Column 1"
        }
    }
}
```

분산

새 열에서 선택한 소스 열의 분산을 반환합니다. 변형은 로 정의됩니다`Var(X) = [Sum ((X - mean(X))^2)]/Count(X)`.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "VARIANCE",
        "Parameters": {
            "sourceColumns": "[\"age\",\"years_in_service\"]",
            "targetColumn": "VARIANCE Column 1"
        }
    }
}
```

텍스트 함수

아래에서 레시피 작업으로 작업하는 텍스트 함수에 대한 참조 주제를 찾습니다.

주제

- CHAR

- ENDS_WITH
- 정확
- 찾기
- LEFT
- LEN
- LOWER
- MERGE_COLUMNS_AND_VALUES
- 적절한
- REMOVE_SYMBOLS
- REMOVE_WHITESPACE
- REPEAT_STRING
- RIGHT
- 오른쪽_찾기
- STARTS_WITH
- STRING_GREATER_THAN
- STRING_GREATER_THAN_EQUAL
- STRING_LESS_THAN
- STRING_LESS_THAN_EQUAL
- SUBSTRING
- TRIM
- UNICODE
- UPPER

CHAR

새 열에서 소스 열의 각 정수 또는 사용자 지정 정수 값에 대한 유니코드 문자를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 유니코드 값을 나타내는 정수입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "sourceColumn": "age",
            "targetColumn": "age_char"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "CHAR",
        "Parameters": {
            "value": 42,
            "targetColumn": "asterisk"
        }
    }
}
```

ENDS_WITH

지정된 수의 가장 오른쪽 문자 또는 사용자 지정 문자열이 패턴과 일치하는 경우 true 새 열을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- pattern - 문자열의 끝과 일치해야 하는 정규식입니다.
- targetColumn - 생성할 새 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "ENDS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[Ss]",
            "targetColumn": "nationality_ends_with"
        }
    }
}
```

정확

다음 중 하나로 채워진 새 열을 생성합니다.

- True 열(또는 값)의 한 문자열이 다른 열(또는 값)의 다른 문자열과 정확히 일치하는 경우.
- False 일치하는 항목이 없는 경우.

파라미터

- sourceColumn1 – 기존 열의 이름입니다.
- sourceColumn2 – 기존 열의 이름입니다.
- value1 - 평가할 문자열.
- value2 - 평가할 문자열.
- targetColumn - 생성할 새 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.

- 둘 다`sourceColumn`*N*.
- 중 하나 `sourceColumn`*N* 및 중 하나`value`*N*.
- 둘 다`value`*N*.

Example예제

```
{
    "RecipeAction": {
        "Operation": "EXACT",
        "Parameters": {
            "sourceColumn1": "nationality",
            "value2": "Argentina",
            "targetColumn": "nationality_exact"
        }
    }
}
```

찾기

왼쪽에서 오른쪽으로 검색하면 소스 열 또는 사용자 지정 값에서 지정된 문자열과 일치하는 문자열을 찾고 결과를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `pattern` - 검색할 정규식입니다.
- `position` - 문자열의 왼쪽 끝에서 시작할 문자 위치입니다.
- `ignoreCase` - 인 경우 문자 간 대소문자 차이(대문자와 소문자 사이)를 `true`무시합니다. 엄격한 일치를 적용하려면 `false` 대신를 사용합니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "FIND",
        "Parameters": {
            "sourceColumn": "city",
            "pattern": "[AEIOU]",
            "position": "1",
            "ignoreCase": "false",
            "targetColumn": "begins_with_a_vowel"
        }
    }
}
```

LEFT

문자 수가 주어지면는 소스 열 또는 사용자 지정 문자열에서 문자열의 맨 왼쪽 문자 수를 가져와서 새 열에서 지정된 맨 왼쪽 문자 수를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `position` - 문자열의 왼쪽 끝에서 시작할 문자 위치입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "3",
            "sourceColumn": "city",
            "targetColumn": "city_left"
        }
```

```
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEFT",
        "Parameters": {
            "position": "5",
            "value": "How now brown cow",
            "targetColumn": "how_now_5_left_chars"
        }
    }
}
```

LEN

소스 열 또는 사용자 지정 문자열의 문자열 길이를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "last_name_len"
```

```
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LEN",
        "Parameters": {
            "value": "Hello",
            "targetColumn": "hello_len"
        }
    }
}
```

LOWER

소스 열 또는 사용자 지정 문자열의 모든 알파벳 문자를 소문자로 변환하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
            "sourceColumn": "last_name",
```

```
            "targetColumn": "last_name_lower"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "LOWER",
        "Parameters": {
            "value": "GOODBYE",
            "targetColumn": "goodbye_lower"
        }
    }
}
```

MERGE_COLUMNS_AND_VALUES

소스 열의 문자열을 연결하고 결과를 새 열에 반환합니다. 병합된 값 사이에 구분 기호를 삽입할 수 있습니다.

파라미터

- `sourceColumns` - JSON 인코딩 형식의 두 개 이상의 기존 열 이름입니다.
- `delimiter` – 선택 사항입니다. 두 소스 열 값 사이에 배치할 하나 이상의 문자입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MERGE_COLUMNS_AND_VALUES",
        "Parameters": {
            "sourceColumns": "[\"last_name\",\"birth_date\"]",
            "delimiter": " was born on: ",
            "targetColumn": "merged_column"
        }
    }
}
```

적절한

소스 열 또는 사용자 지정 값의 문자열에서 모든 알파벳 문자를 적절한 대/소문자로 변환하고 결과를 새 열로 반환합니다.

대소문자라고도 하는 적절한 경우 각 단어의 첫 번째 문자는 대문자로 표시되고 나머지 단어는 소문자로 변환됩니다. 예: 올타리 위로 점프된 퀵 브라운 폭스

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_proper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "PROPER",
        "Parameters": {
            "value": "MR. H. SMITH, ESQ.",
            "targetColumn": "formal_name_proper"
        }
```

```
    }
}
```

REMOVE_SYMBOLS

소스 열 또는 사용자 지정 문자열의 문자열에서 문자, 숫자, 강조 표시된 라틴 문자 또는 공백이 아닌 문자를 제거하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "sourceColumn": "info_url",
            "targetColumn": "info_url_remove_symbols"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_SYMBOLS",
        "Parameters": {
            "value": "$&#$&HEY!#@@",
            "targetColumn": "without_symbols"
        }
```

```
    }
}
```

REMOVE_WHITESPACE

소스 열 또는 사용자 지정 문자열의 문자열에서 공백을 제거하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "sourceColumn": "job_desc",
            "targetColumn": "job_desc_remove_whitespace"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REMOVE_WHITESPACE",
        "Parameters": {
            "value": "This string has spaces in it",
            "targetColumn": "string_without_spaces"
        }
    }
```

```
}
```

REPEAT_STRING

소스 열 또는 사용자 지정 입력 값의 문자열을 지정된 횟수만큼 반복하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `count` - 문자열을 반복할 횟수입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 3,
            "sourceColumn": "last_name",
            "targetColumn": "last_name_repeat_string"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "REPEAT_STRING",
        "Parameters": {
            "count": 80,
            "value": "*",
            "targetColumn": "80_stars"
```

```
        }
    }
}
```

RIGHT

문자 수가 주어지면는 소스 열 또는 사용자 지정 문자열에서 문자열의 가장 오른쪽 문자 수를 가져와서 새 열에서 지정된 가장 오른쪽 문자 수를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `position` - 문자열의 오른쪽에서 시작할 문자 위치입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
        "Parameters": {
            "sourceColumn": "nationality",
            "position": "3",
            "targetColumn": "nationality_right"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "RIGHT",
        "Parameters": {
```

```
            "value": "United States of America",
            "position": "7",
            "targetColumn": "usa_right"
        }
    }
}
```

오른쪽_찾기

오른쪽에서 왼쪽으로 검색하면 소스 열 또는 사용자 지정 값에서 지정된 문자열과 일치하는 문자열을 찾고 결과를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `pattern` - 검색할 정규식입니다.
- `position` - 문자열의 오른쪽 끝에서 시작할 문자 위치입니다.
- `ignoreCase` - 인 경우 문자 간 대소문자 차이(대문자와 소문자 사이)를 `true`무시합니다. 엄격한 일치를 적용하려면 `false` 대신를 사용합니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "RIGHT_FIND",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "s",
            "position": "1",
            "ignoreCase": "true",
            "targetColumn": "ends_with_an_s"
        }
    }
}
```

STARTS_WITH

지정된 수의 맨 왼쪽 문자 또는 사용자 지정 문자열이 패턴과 일치하는 경우 `true` 새 열을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `pattern` - 문자열의 시작과 일치해야 하는 정규식입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STARTS_WITH",
        "Parameters": {
            "sourceColumn": "nationality",
            "pattern": "[AEIOU]",
            "targetColumn": "nationality_starts_with"
        }
    }
}
```

STRING_GREATER_THAN

다음 중 하나로 채워진 새 열을 생성합니다.

- `True` 열의 한 문자열(또는 값)이 다른 열의 다른 문자열(또는 값)보다 큰 경우.
- `False` 일치하는 항목이 없는 경우.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `sourceColumn2` – 기존 열의 이름입니다.

- `value1` - 평가할 문자열.
- `value2` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.
>
> - 둘 다`sourceColumn`*N*.
> - 중 하나 `sourceColumn`*N* 및 중 하나`value`*N*.
> - 둘 다`value`*N*.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_greater_than"
        }
    }
}
```

STRING_GREATER_THAN_EQUAL

다음 중 하나로 채워진 새 열을 생성합니다.

- `True` 열의 한 문자열(또는 값)이 다른 열의 다른 문자열(또는 값)보다 크거나 같은 경우.
- `False` 일치하는 항목이 없는 경우.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `sourceColumn2` – 기존 열의 이름입니다.

- `value1` - 평가할 문자열.
- `value2` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.
>
> - 둘 다`sourceColumn`*N*.
> - 중 하나 `sourceColumn`*N* 및 중 하나`value`*N*.
> - 둘 다`value`*N*.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STRING_GREATER_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "nationality",
            "targetColumn": "string_greater_than_equal",
            "value2": "s"
        }
    }
}
```

STRING_LESS_THAN

다음 중 하나로 채워진 새 열을 생성합니다.

- `True` 열의 한 문자열(또는 값)이 다른 열의 다른 문자열(또는 값)보다 작은 경우.
- `False` 일치하는 항목이 없는 경우.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `sourceColumn2` – 기존 열의 이름입니다.

- `value1` - 평가할 문자열.
- `value2` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.
>
> - 둘 다`sourceColumn`*N*.
> - 중 하나 `sourceColumn`*N* 및 중 하나`value`*N*.
> - 둘 다`value`*N*.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN",
        "Parameters": {
            "sourceColumn1": "first_name",
            "sourceColumn2": "last_name",
            "targetColumn": "string_less_than"
        }
    }
}
```

STRING_LESS_THAN_EQUAL

다음 중 하나로 채워진 새 열을 생성합니다.

- `True` 열의 한 문자열(또는 값)이 다른 열의 다른 문자열(또는 값)보다 작거나 같은 경우.
- `False` 일치하는 항목이 없는 경우.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `sourceColumn2` – 기존 열의 이름입니다.

- `value1` - 평가할 문자열.
- `value2` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.
>
> - 둘 다`sourceColumn`*N*.
> - 중 하나 `sourceColumn`*N* 및 중 하나`value`*N*.
> - 둘 다`value`*N*.

Example예제

```
{
    "RecipeAction": {
        "Operation": "STRING_LESS_THAN_EQUAL",
        "Parameters": {
            "sourceColumn1": "first_name",
            "targetColumn": "string_less_than_equal",
            "value2": "s"
        }
    }
}
```

SUBSTRING

사용자 정의 시작 및 종료 인덱스 값을 기반으로 소스 열에서 지정된 문자열의 일부 또는 전부를 새 열에 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `startPosition` - 문자열의 왼쪽 끝에서 시작할 문자 위치입니다.
- `endPosition` - 문자열의 왼쪽 끝에서 끝날 문자 위치입니다.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SUBSTRING",
        "Parameters": {
            "sourceColumn": "last_name",
            "startPosition": "5",
            "endPosition": "8",
            "targetColumn": "chars_5_through_8"
        }
    }
}
```

TRIM

소스 열 또는 사용자 지정 문자열의 문자열에서 선행 및 후행 공백을 제거하고 결과를 새 열에 반환합니다. 단어 사이의 공백은 제거되지 않습니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- targetColumn - 생성할 새 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "sourceColumn": "nationality",
            "targetColumn": "nationality_trim"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "TRIM",
        "Parameters": {
            "value": "   This string should be trimmed       ",
            "targetColumn": "string_trimmed"
        }
    }
}
```

UNICODE

새 열에서 소스 열에 있는 문자열의 첫 번째 문자 또는 사용자 지정 문자열에 대한 유니코드 인덱스 값을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "UNICODE",
        "Parameters": {
            "sourceColumn": "first_name",
            "targetColumn": "first_name_unicode"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UNICODE",
        "Parameters": {
            "value": "?",
            "targetColumn": "sixty_three"
        }
    }
}
```

UPPER

소스 열 또는 사용자 지정 문자열의 모든 알파벳 문자를 대문자로 변환하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 생성할 새 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
```

```
    "RecipeAction": {
        "Operation": "UPPER",
        "Parameters": {
            "sourceColumn": "last_name",
            "targetColumn": "last_name_upper"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "UPPER",
        "Parameters": {
            "value": "a string of lowercase letters",
            "targetColumn": "string_upper"
        }
    }
}
```

날짜 및 시간 함수

아래에서 레시피 작업에 사용되는 날짜 및 시간 함수에 대한 참조 주제를 찾습니다.

주제

- CONVERT_TIMEZONE
- DATE
- DATE_ADD
- DATE_DIFF
- DATE_FORMAT
- DATE_TIME
- DAY
- 시간
- MILLISECOND
- 분
- MONTH
- MONTH_NAME

- NOW
- 분기
- SECOND
- TIME
- 오늘
- UNIX_TIME
- UNIX_TIME_FORMAT
- 주_일
- WEEK_NUMBER
- YEAR

CONVERT_TIMEZONE

지정된 시간대를 기반으로 소스 열의 시간 값을 새 열로 변환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다. 소스 열의 유형은 `string`, `date`또는 일 수 있습니다`timestamp`.
- `fromTimeZone` - 소스 값 시간대입니다. 아무것도 지정하지 않으면 기본 시간대는 UTC입니다.
- `toTimeZone` - 변환할 시간대입니다. 아무것도 지정하지 않으면 기본 시간대는 UTC입니다.
- `targetColumn` - 새로 생성된 열의 이름입니다.
- `dateTimeFormat` – 선택 사항입니다. 날짜의 형식 문자열입니다. 형식을 지정하지 않으면 기본 형식인이 사용됩니다`yyyy-mm-dd HH:MM:SS`.

Example예제

```
{
    "RecipeAction": {
        "Operation": "CONVERT_TIMEZONE",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "fromTimeZone": "UTC+08:00",
            "toTimeZone": "UTC+08:00",
```

```
            "targetColumn": "DATETIME Column CONVERT_TIMEZONE",
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS"
        }
    }
}
```

DATE

소스 열 또는 제공된 값에서 날짜 값을 포함하는 새 열을 생성합니다.

파라미터

- `dateTimeFormat` – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 이 문자열을 지정하지 않으면 기본 형식은 입니다`yyyy-mm-dd HH:MM:SS`.
- `dateTimeParameters` - 날짜 및 시간의 구성 요소를 나타내는 JSON 인코딩 문자열입니다.
  - `year`
  - `value`
  - `month`
  - `day`
  - `hour`
  - `second`

  각 구성 요소는 다음 중 하나를 지정해야 합니다.
  - `sourceColumn` – 기존 열의 이름입니다.
  - `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DATE",
        "Parameters": {
            "dateTimeFormat": "mm/dd/yy",
            "dateTimeParameters": "{\"year\":{\"value\":\"2019\"},\"month\":{\"value\":
\"12\"},\"day\":{\"value\":\"31\"},\"hour\":{},\"minute\":{},\"second\":{}}",
            "targetColumn": "DATE Column 1"
```

```
        }
    }
}
```

DATE_ADD

소스 열 또는 값의 날짜에 연도, 월 또는 일을 추가하고 결과를 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `units` - 날짜 조정을 위한 측정 단위입니다. 유효한 값은 `MONTHS`, `YEARS`, `MILLISECONDS`, `QUARTERS`, `HOURS`, `MICROSECONDS`, `WEEKS`, `SECONDS`, 및 `DAYS`입니다`MINUTES`.
- `dateAddValue` - 날짜에 추가할의 수`units`입니다.
- `dateTimeFormat` – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 지정하지 않은 경우 기본 형식은 `yyyy-mm-dd HH:MM:SS`입니다.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DATE_ADD",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "units": "DAYS",
            "dateAddValue": "14",
            "dateTimeFormat": "mm/dd/yyyy",
            "targetColumn": "DATE Column 1_DATEADD"
        }
    }
```

```
}
```

DATE_DIFF

두 날짜 간의 차이를 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn1` – 기존 열의 이름입니다.
- `sourceColumn2` – 기존 열의 이름입니다.
- `value1` - 평가할 문자열.
- `value2` - 평가할 문자열.
- `units` -의 측정 단위는 날짜 간의 차이를 설명합니다. 유효한 값은 `MONTHS`, `YEARS`, `MILLISECONDS`, `QUARTERS`, `HOURS`, `MICROSECONDS`, `WEEKS`, `SECONDS`, 및 `DAYS`입니다`MINUTES`.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> 다음 조합 중 하나만 지정할 수 있습니다.
>
> - `sourceColumn1` 및 모두`sourceColumn2`.
> - `sourceColumn1` 또는 중 하나`sourceColumn2`와 `value1` 또는 중 하나`value2`.
> - `value1` 및 모두. `value2`

Example예제

```
{
    "RecipeAction": {
        "Operation": "DATE_DIFF",
        "Parameters": {
            "value1": "2020-01-01",
            "value2": "2020-10-06",
            "units": "DAYS",
            "targetColumn": "DATEDIFF Column 1"
        }
    }
```

```
}
```

DATE_FORMAT

날짜를 나타내는 문자열에서 특정 형식의 날짜가 포함된 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열입니다.
- `dateTimeFormat` – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 지정하지 않은 경우 기본 형식은 `yyyy-mm-dd HH:MM:SS`입니다.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "sourceColumn": "DATE Column 1",
            "dateTimeFormat": "month*dd*yyyy",
            "targetColumn": "DATE Column 1_DATEFORMAT"
        }
    }
}
```

```
{
    "RecipeAction": {
        "Operation": "DATE_FORMAT",
        "Parameters": {
            "value": "22:10:47",
            "dateTimeFormat": "HH:MM:SS",
            "targetColumn": "formatted_date_value"
```

```
        }
    }
}
```

DATE_TIME

소스 열 또는 제공된 값에서 날짜 및 시간 값이 포함된 새 열을 생성합니다.

파라미터

- `dateTimeFormat` – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 이 문자열을 지정하지 않으면 기본 형식은 입니다`yyyy-mm-dd HH:MM:SS`.
- `dateTimeParameters` - 날짜 및 시간의 구성 요소를 나타내는 JSON 인코딩 문자열입니다.
  - `year`
  - `value`
  - `month`
  - `day`
  - `hour`
  - `second`

  각 구성 요소는 다음 중 하나를 지정해야 합니다.
  - `sourceColumn` – 기존 열의 이름입니다.
  - `value` - 평가할 문자열.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DATE_TIME",
        "Parameters": {
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "dateTimeParameters": "{\"year\":{\"value\":\"2010\"},\"month\":{\"value\":
\"5\"},\"day\":{\"value\":\"21\"},\"hour\":{\"value\":\"13\"},\"minute\":{\"value\":
\"34\"},\"second\":{\"value\":\"25\"}}",
            "targetColumn": "DATETIME Column 1"
        }
    }
```

```
}
```

DAY

날짜를 나타내는 문자열에서 월의 날짜를 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_DAY"
        }
    }
}
```

시간

날짜를 나타내는 문자열에서 시간 값을 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "HOUR",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_HOUR"
        }
    }
}
```

MILLISECOND

소스 열 또는 입력 값의 밀리초 값이 포함된 새 열을 생성합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다. 소스 열의 유형은 string, date또는 일 수 있습니다timestamp.
- value - 평가할 문자열.
- targetColumn - 새로 생성된 열의 이름입니다.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
```

```
        "Operation": "MILLISECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MILLISECOND"
        }
    }
}
```

분

날짜를 나타내는 문자열에서 분 값을 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MINUTE",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_MINUTE"
        }
    }
}
```

MONTH

날짜를 나타내는 문자열에서 월의 번호가 포함된 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MONTH",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTH Column 1"
        }
    }
}
```

MONTH_NAME

날짜를 나타내는 문자열에서 월의 이름이 포함된 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "MONTH_NAME",
        "Parameters": {
            "value": "2018-05-27",
            "targetColumn": "MONTHNAME Column 1"
        }
    }
}
```

NOW

형식의 현재 날짜 및 시간이 포함된 새 열을 생성합니다`yyyy-mm-dd HH:MM:SS`.

파라미터

- `timeZone` - 시간대의 이름입니다. 시간대를 지정하지 않으면 기본값은 협정 세계시(UTC)입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "NOW",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "NOW Column 1"
        }
}
```

분기

날짜를 나타내는 문자열에서 날짜 기반 분기가 포함된 새 열을 생성합니다.

> **Note**
>
> 분기는 새 열에서 1, 2, 3 또는 4로 지정됩니다.

- 1은 1월, 2월, 3월입니다.
- 2는 4월, 5월, 6월입니다.
- 3은 7월, 8월, 9월입니다.
- 4는 10월, 11월, 12월입니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다. 소스 열의 유형은 `string`, `date`또는 일 수 있습니다`timestamp`.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름입니다.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "QUARTER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_QUARTER"
        }
    }
}
```

SECOND

날짜를 나타내는 문자열에서 두 번째 값을 포함하는 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "SECOND",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_SECOND"
        }
    }
}
```

TIME

제공된 소스 열 또는 값에서 시간 값이 포함된 새 열을 생성합니다.

파라미터

- `dateTimeFormat` – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 이 문자열을 지정하지 않으면 기본 형식은 입니다`yyyy-mm-dd HH:MM:SS`.
- `dateTimeParameters` - 날짜 및 시간의 구성 요소를 나타내는 JSON 인코딩 문자열입니다.
  - `year`
  - `value`
  - `month`
  - `day`
  - `hour`
  - `second`

  각 구성 요소는 다음 중 하나를 지정해야 합니다.

  - `sourceColumn` – 기존 열의 이름입니다.
  - `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "TIME",
        "Parameters": {
            "dateTimeFormat": "HH:MM:SS",
            "dateTimeParameters": "{\"year\":{},\"month\":{},\"day\":{},\"hour\":
{\"sourceColumn\":\"rand_hour\"},\"minute\":{\"sourceColumn\":\"rand_minute\"},\"second
\":{\"sourceColumn\":\"rand_second\"}}",
            "targetColumn": "TIME Column 1"
        }
    }
}
```

오늘

형식의 현재 날짜가 포함된 새 열을 생성합니다`yyyy-mm-dd`.

파라미터

- `timeZone` - 시간대의 이름입니다. 시간대를 지정하지 않으면 기본값은 협정 세계시(UTC)입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "TODAY",
        "Parameters": {
            "timeZone": "US/Pacific",
            "targetColumn": "TODAY Column 1"
        }
    }
```

```
}
```

UNIX_TIME

소스 열 또는 입력 값을 기반으로 1970년 1월 1일 이후의 초 수인 epoch 시간(Unix 시간)을 나타내는 숫자가 포함된 새 열을 생성합니다. 시간대를 추론할 수 있는 경우 출력은 해당 시간대에 있습니다. 그렇지 않으면 출력은 협정 세계시(UTC)로 표시됩니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME",
        "Parameters": {
            "sourceColumn": "TIME Column 1",
            "targetColumn": "TIME Column 1_UNIXTIME"
        }
    }
}
```

UNIX_TIME_FORMAT

소스 열 또는 입력 값의 Unix 시간을 지정된 숫자 날짜 형식으로 변환하고 결과를 새 열로 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

- value - Unix epoch 타임스탬프를 나타내는 정수입니다.
- dateTimeFormat – 선택 사항입니다. 새 열에 표시되는 날짜의 형식 문자열입니다. 지정하지 않은 경우 기본 형식은 yyyy-mm-dd HH:MM:SS입니다.
- targetColumn - 새로 생성된 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "UNIX_TIME_FORMAT",
        "Parameters": {
            "value": "1601936554",
            "dateTimeFormat": "yyyy-mm-dd HH:MM:SS",
            "targetColumn": "UNIXTIMEFORMAT Column 1"
        }
    }
}
```

주_일

날짜를 나타내는 문자열에서 요일이 포함된 새 열을 생성합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- targetColumn - 새로 생성된 열의 이름.

> **Note**
>
> sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "WEEK_DAY",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEKDAY"
        }
    }
}
```

WEEK_NUMBER

날짜를 나타내는 문자열에서 주 수(1~52)가 포함된 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "WEEK_NUMBER",
        "Parameters": {
            "sourceColumn": "DATETIME Column 1",
            "targetColumn": "DATETIME Column 1_WEEK_NUMBER"
        }
    }
}
```

YEAR

날짜를 나타내는 문자열에서 연도가 포함된 새 열을 생성합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `value` - 평가할 문자열.
- `targetColumn` - 새로 생성된 열의 이름.

> **Note**
>
> `sourceColumn` 또는 `value`을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "YEAR",
        "Parameters": {
            "value": "2019-06-12",
            "targetColumn": "YEAR Column 1"
        }
    }
}
```

윈도우 함수

아래에서 레시피 작업으로 작동하는 창 함수에 대한 참조 주제를 찾습니다.

주제

- FILL
- next
- PREV
- 롤링_평균

- ROLLING_COUNT_A
- ROLLING_KTH_LARGEST
- 롤링_KTH_LARGEST_고유
- ROLLING_MAX
- 롤링_분
- 롤링 모드
- 롤링_표준_편차
- ROLLING_SUM
- 롤링_변수
- ROW_NUMBER
- 세션

FILL

지정된 소스 열을 기반으로 새 열을 반환합니다. 소스 열에 누락된 값 또는 null 값의 경우는 해당 소스 값 앞뒤의 행 창에서 가장 최근의 비어 있지 않은 값을 FILL 선택합니다. 그러면 선택한 값이 새 열에 배치됩니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- numRowsBefore - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- numRowsAfter - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- targetColumn - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "FILL",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "last_name",
            "targetColumn": "last_name_FILL"
```

```
        }
    }
}
```

next

새 열을 반환합니다. 여기서 각 값은 소스 열의 뒷부분에 있는 n개 행인 값을 나타냅니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- numRows - 소스 열 앞부분의 n개 행을 나타내는 값입니다. 예를 들어 numRows가 3인 경우는 세 번째 다음 `sourceColumn` 값을 새 `targetColumn` 값으로 NEXT 사용합니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "NEXT",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_NEXT"
        }
    }
}
```

PREV

새 열을 반환합니다. 여기서 각 값은 소스 열 앞의 n개 행인 값을 나타냅니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- numRows - 소스 열 앞부분의 n개 행을 나타내는 값입니다. 예를 들어 numRows가 3인 경우는 세 번째 이전 `sourceColumn` 값을 새 `targetColumn` 값으로 PREV 사용합니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "PREV",
        "Parameters": {
            "numRows": "1",
            "sourceColumn": "age",
            "targetColumn": "age_PREV"
        }
    }
}
```

롤링_평균

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수까지 값의 롤링 평균을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_AVERAGE",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_AVERAGE"
        }
    }
}
```

ROLLING_COUNT_A

새 열에서 이전에 지정된 행 수에서 지정된 열의 현재 행 뒤에 지정된 행 수로 null이 아닌 값의 롤링 수를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_COUNT_A",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_COUNT_A"
        }
    }
}
```

ROLLING_KTH_LARGEST

새 열에서 이전에 지정된 행 수에서 지정된 열의 현재 행 뒤에 지정된 행 수까지의 롤링 k번째 최대값을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `value` - k의 값입니다.

- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST",
    "Parameters": {
      "sourceColumn": "weight_kg",
      "numRowsBefore": "5",
      "numRowsAfter": "5",
      "value": "3"
      "targetColumn": "weight_kg_ROLLING_KTH_LARGEST"
    }
  }
}
```

롤링_KTH_LARGEST_고유

새 열에서 이전에 지정된 행 수에서 지정된 열의 현재 행 뒤에 지정된 행 수까지의 롤링 고유 k번째 최대 값을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `value` - k의 값입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
  "Action": {
    "Operation": "ROLLING_KTH_LARGEST_UNIQUE",
    "Parameters": {
      "sourceColumn": "games_played",
```

```
        "numRowsBefore": "3",
        "numRowsAfter": "3",
        "value": "5",
        "targetColumn": "weight_kg_ROLLING_KTH_LARGEST_UNIQUE"
      }
    }
  }
```

ROLLING_MAX

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수로 값의 롤링 최대값을 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.

  `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_MAX",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MAX"
        }
    }
}
```

롤링_분

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수로 값의 롤링 최소값을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.

  numRowsBefore - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- numRowsAfter - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- targetColumn - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_MIN",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MIN"
        }
    }
}
```

롤링 모드

새 열에서 롤링 모드(가장 일반적인 값)를 이전 지정된 행 수에서 지정된 열의 현재 행 이후 지정된 행 수로 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- numRowsBefore - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- numRowsAfter - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- modeType - 창에 적용할 모달 함수입니다. 유효한 값은 NONE, MINIMUM, MAXIMUM 및 AVERAGE입니다.
- targetColumn - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_MODE",
        "Parameters": {
            "modeType": "MINIMUM",
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_MODE"
        }
    }
}
```

롤링_표준_편차

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수까지 값의 롤링 표준 편차를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `numRowsBefore` - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- `numRowsAfter` - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_STDEV",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_STDEV"
        }
    }
}
```

ROLLING_SUM

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수까지 값의 롤링 합계를 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.

  numRowsBefore - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- numRowsAfter - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- targetColumn - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_SUM",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_SUM"
        }
    }
}
```

롤링_변수

새 열에서 앞의 지정된 행 수에서 지정된 열의 현재 행 뒤의 지정된 행 수로 값의 롤링 분산을 반환합니다.

파라미터

- sourceColumn – 기존 열의 이름입니다.
- numRowsBefore - 창의 시작을 나타내는 현재 소스 행 앞의 행 수입니다.
- numRowsAfter - 창의 끝을 나타내는 현재 소스 행 뒤의 행 수입니다.
- targetColumn - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROLLING_VAR",
        "Parameters": {
            "numRowsAfter": "10",
            "numRowsBefore": "10",
            "sourceColumn": "weight_kg",
            "targetColumn": "weight_kg_ROLLING_VAR"
        }
    }
}
```

ROW_NUMBER

새 열에 "group by" 및 "order by" 문에서 열 이름으로 생성된 창을 기반으로 세션 식별자를 반환합니다.

파라미터

- `groupByColumns` - "group by" 열을 설명하는 JSON 인코딩 문자열입니다.
- `orderByColumns` - “순서 기준” 열을 설명하는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "ROW_NUMBER",
        "Parameters": {
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "Row number"
        }
    }
}
```

세션

새 열에 "group by" 및 "order by" 문에서 열 이름으로 생성된 창을 기반으로 세션 식별자를 반환합니다.

파라미터

- `sourceColumn` – 기존 열의 이름입니다.
- `units` - 세션 길이를 설명하는 측정 단위입니다. 유효한 값은 `MONTHS`, , `YEARS`, `MILLISECONDSQUARTERS`, `HOURS`, `MICROSECONDS`, `WEEKS`, `SECONDS`, 및 `DAYS`입니다`MINUTES`.
- `value` - 기간을 `units` 정의할의 수입니다.
- `groupByColumns` - "group by" 열을 설명하는 JSON 인코딩 문자열입니다.
- `orderByColumns` - “순서 기준” 열을 설명하는 JSON 인코딩 문자열입니다.
- `targetColumn` - 새로 생성된 열의 이름.

Example예제

```
{
    "Action": {
        "Operation": "SESSION",
        "Parameters": {
            "sourceColumn": "object number",
            "units": "MINUTES",
            "value": "10",
            "groupByColumns": "[\"is public domain\"]",
            "orderByColumns": "[\"dimensions\"]",
            "targetColumn": "object number_SESSION",
        }
    }
}
```

웹 함수

아래에서 레시피 작업으로 작업하는 웹 함수에 대한 참조 주제를 찾습니다.

주제

- IP_TO_INT
- INT_TO_IP

- URL_PARAMS

IP_TO_INT

소스 열의 인터넷 프로토콜 버전 4(IPv4) 값 또는 기타 값을 대상 열의 해당 정수 값으로 변환하고 결과를 새 열로 반환합니다. 이 함수는 IPv4에서만 작동합니다.

예를 들어 다음 IP 주소를 고려합니다.

```
192.168.1.1
```

이 값을 IP_TO_INT에 대한 입력으로 사용하는 경우 출력 값은 다음과 같습니다.

```
3232235777
```

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- targetColumn - 생성할 새 열의 이름.

sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "IP_TO_INT",
        "Parameters": {
            "sourceColumn": "my_ip_address",
            "targetColumn": "IP_TO_INT Column 1"
        }
    }
}
```

INT_TO_IP

소스 열의 정수 값 또는 기타 값을 대상 열의 해당 IPv4 값으로 변환하고 결과를 새 열에 반환합니다. 이 함수는 IPv4에서만 작동합니다.

예를 들어 다음 정수를 생각해 보세요.

```
167772410
```

이 값을 INT_TO_IP에 대한 입력으로 사용하는 경우 출력 값은 다음과 같습니다.

```
10.0.0.250
```

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- targetColumn - 생성할 새 열의 이름.

sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
[  {
    "RecipeAction": {
      "Operation": "INT_TO_IP",
      "Parameters": {
        "sourceColumn": "my_integer",
        "targetColumn": "INT_TO_IP Column 1"
      }
    }
  }
]
```

URL_PARAMS

URL 문자열에서 쿼리 파라미터를 추출하여 JSON 객체로 형식을 지정한 다음 새 열에 결과를 반환합니다.

예를 들어 다음 URL을 고려합니다.

```
https://example.com/?firstParam=answer&secondParam=42
```

이 값을 URL_PARAMS에 대한 입력으로 사용하는 경우 출력 값은 다음과 같습니다.

```
{"firstParam": ["answer"], "secondParam": ["42"]}
```

파라미터

- sourceColumn – 기존 열의 이름입니다.
- value - 평가할 문자열.
- targetColumn - 생성할 새 열의 이름.

sourceColumn 또는 value을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "URL_PARAMS",
        "Parameters": {
            "sourceColumn": "my_url",
            "targetColumn": "URL_PARAMS Column 1"
        }
    }
}
```

기타 함수

아래에서 레시피 작업으로 작업하는 다른 함수에 대한 참조 주제를 찾습니다.

주제

- COALESCE
- GET_ACTION_RESULT
- GET_STEP_DATAFRAME

COALESCE

열 배열에 있는 첫 번째 null이 아닌 값을 새 열에 반환합니다. 함수에 나열된 열의 순서에 따라 검색 순서가 결정됩니다.

파라미터

- `sourceColumns` - 기존 열 목록을 나타내는 JSON 인코딩 문자열입니다.
- `targetColumn` - 생성할 새 열의 이름.

Example예제

```
{
    "RecipeAction": {
        "Operation": "COALESCE",
        "Parameters": {
            "sourceColumns": "[\"nation_position\",\"joined\"]",
            "targetColumn": "COALESCE Column 1"
        }
    }
}
```

GET_ACTION_RESULT

이전에 제출한 작업의 결과를 가져옵니다. 대화형 환경에서만 사용할 수 있습니다.

파라미터

- `actionId` - 원래 SendProjectSessionAction 응답에서 반환된 ActionId입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "GET_ACTION_RESULT",
        "Parameters": {
            "actionId": "7",
        }
    }
}
```

GET_STEP_DATAFRAME

프로젝트 레시피의 단계에서 데이터 프레임을 가져옵니다. 대화형 환경에서만 사용할 수 있습니다. ViewFrame 파라미터와 함께 사용하여 대규모 데이터 프레임에 페이지 매김합니다.

파라미터

- `stepIndex` - 프로젝트 레시피에서 데이터 프레임을 가져올 단계의 인덱스입니다.

Example예제

```
{
    "RecipeAction": {
        "Operation": "GET_STEP_DATAFRAME",
        "Parameters": {
            "stepIndex": "0"
        }
    }
}
```

에 대한 할당량AWS Glue DataBrew

Service AWS Service Quotas 콘솔에서 DataBrew 서비스 할당량을 볼 수 있습니다. 조정 가능한 모든 할당량에 대해 할당량 증가를 요청할 수도 있습니다.

AWS Glue DataBrew개발자 안내서의 문서 기록

현재 API 버전: databrew-2017-07-25

다음 표에서는이 릴리스에 대한 설명서를 설명합니다AWS Glue DataBrew. AWS Glue DataBrew개발자 안내서가 업데이트될 때 알림을 받으려면 RSS 피드를 구독하면 됩니다.

변경 사항	설명	날짜
glue:GetCustomEntityType AWS관리형 정책에 추가됨	이 권한은 PII 식별이 활성화된 AWS Glue DataBrew프로필 작업을 실행하는 데 필요합니다. 자세한 내용은 AWS Glue DataBrewAWS관리형 정책에 대한 업데이트를 참조하세요.	2024년 3월 20일
CRYPTOGRAPHIC_HASH 변환에서 여러 해싱 알고리즘 지원	이제 열에서 값을 해싱할 때 해싱 알고리즘을 지정할 수 있습니다. 자세한 내용은 CRYPTOGRAPHIC_HASH를 참조하세요.	2023년 8월 11일
glue:BatchGetCustomEntityTypes AWS관리형 정책에 추가됨	이 권한은 PII 식별이 활성화된 AWS Glue DataBrew프로필 작업을 실행하는 데 필요합니다. 자세한 내용은 AWS Glue DataBrewAWS관리형 정책에 대한 업데이트를 참조하세요.	2022년 5월 9일
Apache ORC 파일 형식 지원	이제 DataBrew는 DataBrew 데이터 소스 및 출력의 파일 형식으로 Apache ORC를 지원합니다. 자세한 내용은 데이터 소스에 지원되는 파일 유형을 참조하세요.	2022년 3월 31일

교차 계정AWS Glue Data Catalog Amazon S3 액세스 지원	이제AWS Glue콘솔에서 적절한 리소스 정책이 생성된AWS 계정경우 다른에서AWS Glue Data Catalog S3 테이블에 액세스할 수 있습니다. 정책을 생성한 후 DataBrew 데이터 세트를 생성할 때 관련 Data Catalog S3 테이블을 입력 소스로 선택할 수 있습니다. 자세한 내용은 데이터 소스 및 출력에 지원되는 연결을 참조하세요.	2022년 3월 11일
Amazon AppFlow와의 네이티브 콘솔 통합 지원	이제 DataBrew는 Amazon AppFlow와 네이티브 콘솔 통합을 제공합니다. 이 통합을 통해 Salesforce, Zendesk, Slack, ServiceNow 및 기타 software-as-a-service(SaaS) 애플리케이션의 데이터에 연결할 수 있습니다. Amazon S3 및 Amazon Redshift AWS 서비스와 같은의 데이터에 연결할 수도 있습니다. 자세한 내용은 데이터 소스 및 출력에 지원되는 연결을 참조하세요.	2021년 11월 18일
데이터 품질 규칙 지원	DataBrew는 이제 특정 데이터에 대한 비즈니스 요구 사항을 정의하는 사용자 지정 가능한 검증 검사인 데이터 품질 규칙 생성을 지원합니다. 자세한 내용은 에서 데이터 품질 검증을 참조하세요AWS Glue DataBrew.	2021년 11월 18일

사용자 지정 SQL 문 지원	DataBrew는 이제 Amazon Redshift 및 Snowflake에서 데이터를 검색하기 위한 사용자 지정 SQL 문을 지원합니다. 이 지원은 특별히 구축된 쿼리를 사용하여 큰 테이블에서 반환되는 데이터를 선택하고 제한할 수 있음을 의미합니다. 자세한 내용은 데이터 소스 및 출력에 지원되는 연결을 참조하세요.	2021년 11월 18일
PII 탐지 지원	DataBrew는 이제 개인 식별 정보(PII) 감지를 지원합니다. 이렇게 하면 데이터 준비 중에 PII를 마스킹할 수 있습니다. 자세한 내용은 개인 식별 정보(PII) 식별 및 처리를 참조하세요.	2021년 11월 18일
추가AWS리전 지원	DataBrew는 이제 추가AWS 리전을 지원합니다. 지원되는 리전 목록은 AWS Glue DataBrew엔드포인트 및 할당량을 참조하세요.	2021년 10월 5일
Lake Formation 기반 Amazon S3 테이블에 데이터 쓰기 지원	DataBrew는 이제 기반AWS Glue Data Catalog S3 테이블에 데이터 쓰기를 지원합니다AWS Lake Formation. DataBrew는 이제 Tableau Hyper 형식으로 데이터 쓰기도 지원합니다. 자세한 내용은 AWS Glue DataBrew레시피 작업 생성 및 작업을 참조하세요.	2021년 8월 13일

JDBC 대상에 데이터 쓰기 지원	DataBrew는 이제 JDBC 지원 데이터베이스 및 데이터 웨어하우스에 직접 데이터 쓰기를 지원합니다. 여기에는 Amazon Redshift, Snowflake, Microsoft SQL Server, MySQL, Oracle Database 및 PostgreSQL이 포함됩니다. 자세한 내용은 AWS Glue DataBrew레시피 작업 생성 및 작업을 참조하세요.	2021년 7월 23일
프로파일 작업에 대해 생성되는 데이터 품질 통계 지정 지원	이제 DataBrew는 프로파일 작업의 데이터 세트에 대해 자동 생성되는 데이터 품질 통계 지정을 지원합니다. 자세한 내용은 AWS Glue DataBrew레시피 작업 생성 및 작업을 참조하세요.	2021년 7월 23일
에 데이터 세트 작성 지원AWS Glue Data Catalog	DataBrew에는 이제에 직접 데이터 세트를 작성할 수 있는 지원이 포함됩니다AWS Glue Data Catalog. 데이터 카탈로그의 Amazon S3, Amazon Redshift 및 Amazon RDS 테이블에서 데이터 준비 레시피를 실행하는 작업에서 생성된 데이터 세트를 저장하도록 선택할 수 있습니다. 지원되는 RDS 테이블에는 Amazon Aurora, RDS for Oracle, RDS for Microsoft SQL Server, RDS for MySQL 및 RDS for PostgreSQL에 대한 테이블이 포함됩니다.	2021년 6월 30일

고급 데이터 형식 식별 지원	DataBrew에는 이제 열의 고급 데이터 유형을 자동으로 식별하고 표시하는 지원이 포함되어 있으므로 특정 유형의 데이터가 포함된 열을 더 쉽게 정규화할 수 있습니다. 이러한 유형의 데이터에는 사회보장번호, 이메일 주소, 전화번호, 성별, 신용 카드, URL, IP 주소, 날짜 및 시간, 통화, 우편번호, 국가, 리전, 주 및 도시가 포함됩니다.	2021년 6월 30일
Amazon AppFlow를 사용하여 SAAS 애플리케이션에서 데이터 전송 지원	DataBrew는 이제 Amazon AppFlow를 사용하여 Salesforce, Zendesk, Slack, ServiceNow와 같은 타사 software-as-a-service(SaaS) 애플리케이션에서 Amazon S3로 데이터를 전송할 수 있도록 지원합니다. 자세한 내용은 데이터 소스 및 출력에 지원되는 연결을 참조하세요.	2021년 4월 29일
JDBC 데이터베이스의 입력으로 DataBrew 데이터 세트 생성 지원	이제 DataBrew는 Amazon Redshift, Snowflake, Microsoft SQL Server, MySQL, Oracle Database 및 PostgreSQL을 포함하여 JDBC 지원 데이터베이스 및 데이터 웨어하우스의 데이터에서 데이터 세트 생성을 지원합니다. 자세한 내용은 데이터 소스 및 출력에 지원되는 연결을 참조하세요.	2021년 4월 2일

추가 지원AWS 리전	DataBrew는 이제 추가를 지원합니다AWS 리전. 지원되는 리전 목록은 AWS Glue DataBrew엔드포인트 및 할당량을 참조하세요.	2021년 1월 28일
중복 처리를 위한 새로운 변환	중복 처리를 위한 네 가지 새로운 변환이 DataBrew 콘솔 및 API에 추가되었습니다. 자세한 내용은 데이터 품질 레시피 단계에서 DELETE_DUPLICATE_ROWS, FLAG_DUPLICATE_ROWS, FLAG_DUPLICATES_IN_COLUMN 및 REMOVE_DUPLICATES를 참조하세요.	2021년 1월 28일
추가 CSV 구분 기호	이제 DataBrew는 DataBrew 데이터 세트를 생성하는 데 사용되는 쉼표로 구분된 값(CSV) 파일의 쉼표 이외의 추가 구분 기호를 지원합니다. 자세한 내용은 AWS Glue DataBrew데이터 세트 생성 및 사용을 참조하세요.	2021년 1월 28일
JupyterLab용 DataBrew 확장	이제 JupyterLab에서를 확장 AWS Glue DataBrew으로 사용할 수 있습니다. 자세한 내용은 JupyterLab에서 DataBrew를 확장으로 사용을 참조하세요.	2020년 11월 20일
새로운 데이터 준비 도구:AWS Glue DataBrew	이 문서는 첫 번째 AWS Glue DataBrew개발자 안내서 릴리스입니다.	2020년 11월 11일

AWS용어집

최신AWS용어는 AWS 용어집참조의 AWS용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.