aws

Getting Started Guide

Amazon Redshift

Amazon Redshift: Getting Started Guide

Table of Contents

Amazon Redshift will no longer support the creation of new Python UDFs starting Patch 198. Existing Python UDFs will continue to function until June 30, 2026. For more information, see the blog post .

Get started with Amazon Redshift Serverless data warehouses

If you are a first-time user of Amazon Redshift Serverless, we recommend that you read the following sections to help you get started using Amazon Redshift Serverless. The basic flow of Amazon Redshift Serverless is to create serverless resources, connect to Amazon Redshift Serverless, load sample data, and then run queries on the data. In this guide, you can choose to load sample data from Amazon Redshift Serverless or from an Amazon S3 bucket. The sample data is used throughout the Amazon Redshift documentation to demonstrate features. To get started using Amazon Redshift provisioned data warehouses, see Get started with Amazon Redshift provisioned data warehouses.

- the section called “Signing up for AWS”
- the section called “Creating a data warehouse with Amazon Redshift Serverless”
- the section called “Loading in data from Amazon S3”

Signing up for AWS

If you don't already have an AWS account, sign up for one. If you already have an account, you can skip this prerequisite and use your existing account.

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   When you sign up for an AWS account, an AWS account root user is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

Creating a data warehouse with Amazon Redshift Serverless

The first time you log in to the Amazon Redshift Serverless console, you are prompted to access the getting started experience, which you can use to create and manage serverless resources. In this guide, you'll create serverless resources using Amazon Redshift Serverless's default settings.

For more granular control of your setup, choose **Customize settings**.

> **Note**
>
> Redshift Serverless requires an Amazon VPC with three subnets in three different availability zones. Redshift Serverless also requires at least 3 available IP addresses. Make sure that the Amazon VPC that you use for Redshift Serverless has three subnets in three different availability zones, and at least 3 available IP addresses, before continuing. For more information about creating subnets in an Amazon VPC, see Create a subnet in the *Amazon Virtual Private Cloud User Guide*. For more information about IP addresses in an Amazon VPC, see IP addressing for your VPCs and subnets.

To configure with default settings:

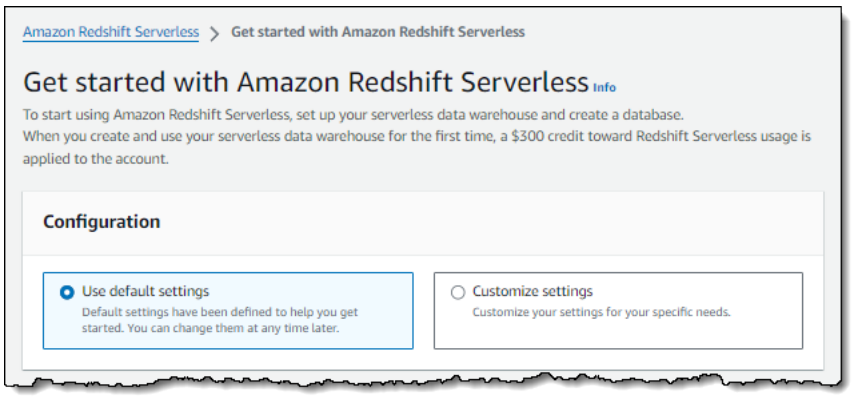
1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

   Choose **Try Redshift Serverless Free Trial**.

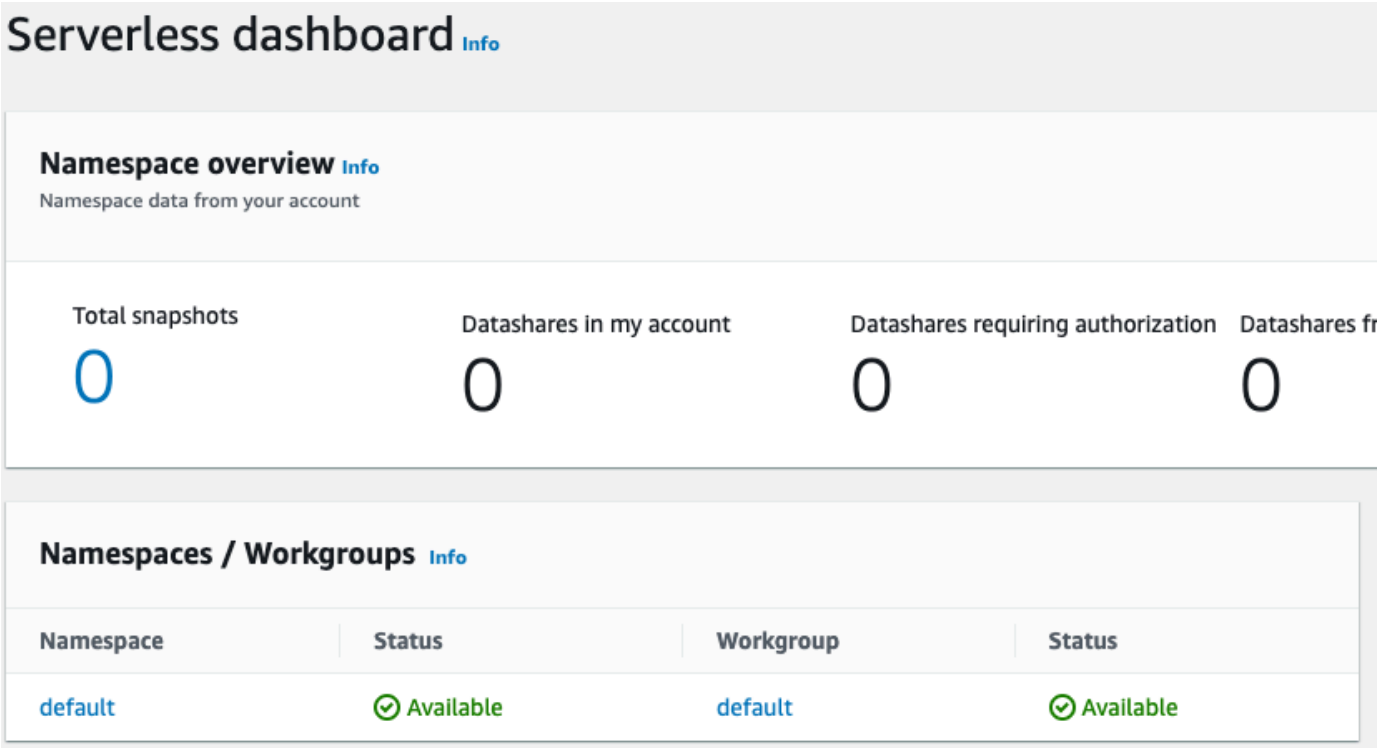
2. Under **Configuration**, choose **Use default settings**. Amazon Redshift Serverless creates a default namespace with a default workgroup associated with this namespace. Choose **Save configuration**.

   > **Note**
   >
   > A **Namespace** is a collection of database objects and users. Namespaces group together all of the resources you use in Redshift Serverless, such as schemas, tables, users, datashares, and snapshots.
   > A **Workgroup** is a collection of compute resources. Workgroups house compute resources that Redshift Serverless uses to run computational tasks.

   The following screenshot shows the default settings for Amazon Redshift Serverless.



3. After setup completes, choose **Continue** to go to your **Serverless dashboard**. You can see that the serverless workgroup and namespace are available.



> **Note**
>
> If Redshift Serverless doesn't create the workgroup successfully, you can do the following:
>
> - Address any errors that Redshift Serverless reports, such as having too few subnets in your Amazon VPC.

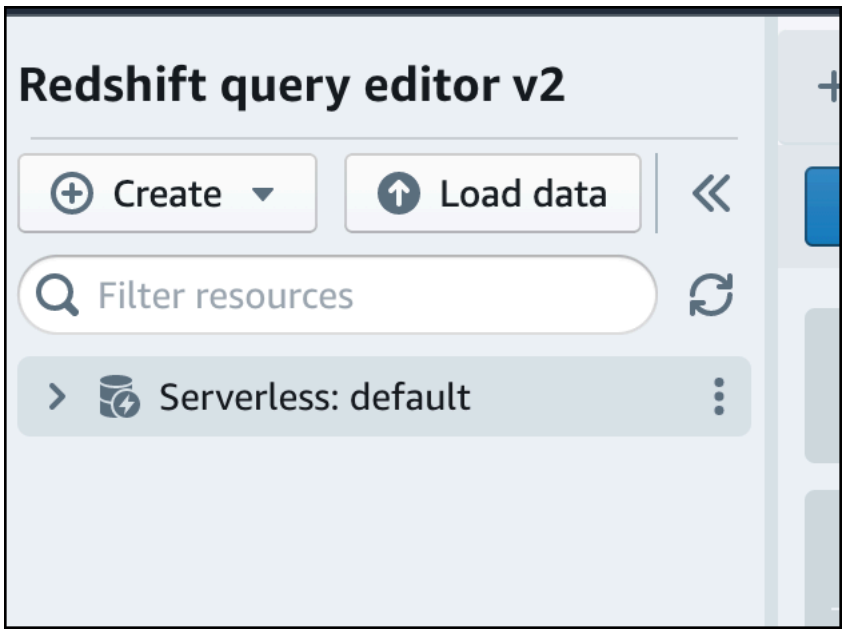
- Delete the namespace by choosing **default-namespace** in the Redshift Serverless dashboard, and then choosing **Actions**, **Delete namespace**. Deleting a namespace takes several minutes.
- When you open the Redshift Serverless console again, the welcome screen appears.

Loading sample data

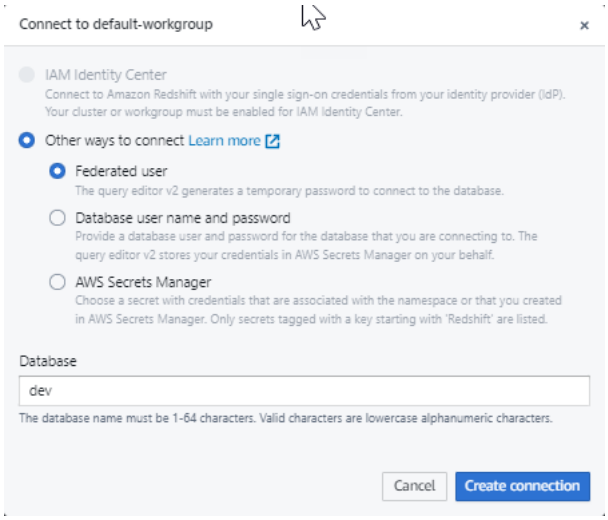
Now that you've set up your data warehouse with Amazon Redshift Serverless, you can use the Amazon Redshift query editor v2 to load sample data.

1. To launch query editor v2 from the Amazon Redshift Serverless console, choose **Query data**. When you invoke query editor v2 from the Amazon Redshift Serverless console, a new browser tab opens with the query editor. The query editor v2 connects from your client machine to the Amazon Redshift Serverless environment.

2. For this guide, you'll use your AWS administrator account and the default AWS KMS key. For information about configuring the query editor v2, including which permissions are needed, see Configuring your AWS account in the *Amazon Redshift Management Guide*. For information about configuring Amazon Redshift to use a customer managed key, or to change the KMS key that Amazon Redshift uses, see Changing the AWS KMS key for a namespace.
3. To connect to a workgroup, choose the workgroup name in the tree-view panel.

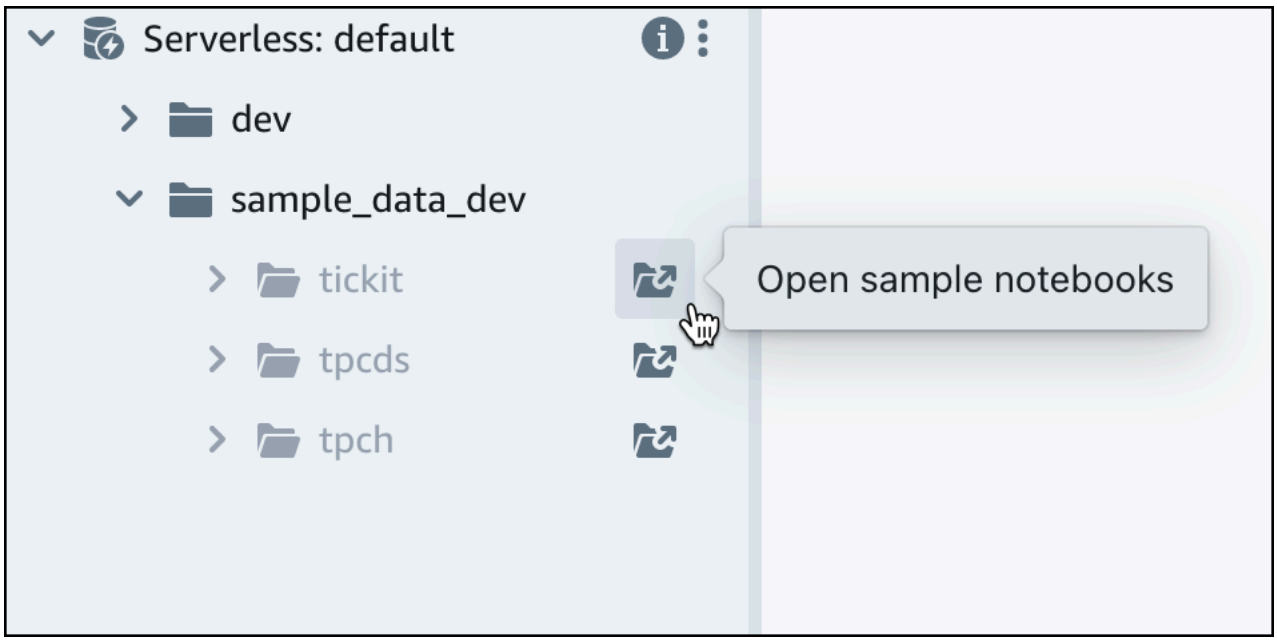


4. When connecting to a new workgroup for the first time within query editor v2, you must select the type of authentication to use to connect to the workgroup. For this guide, leave **Federated user** selected, and choose **Create connection**.



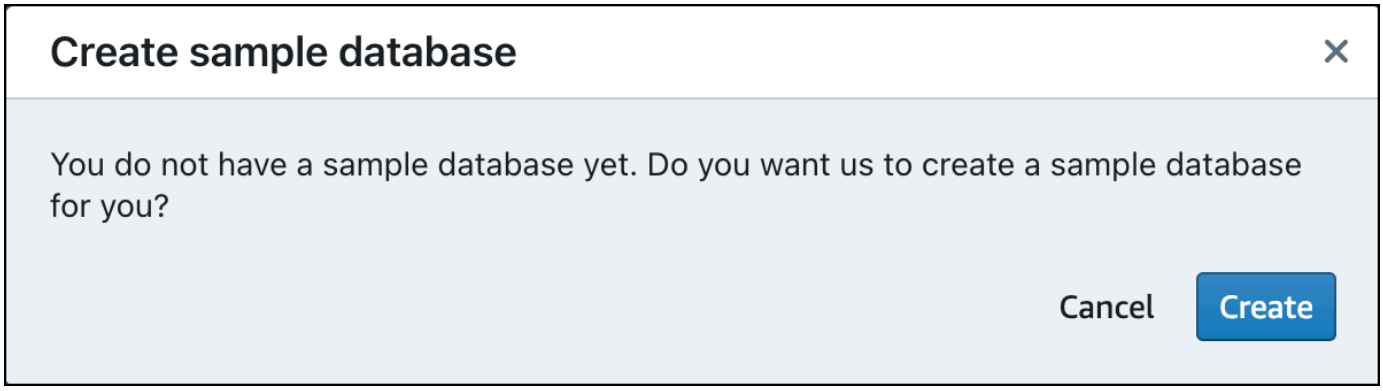
Once you are connected, you can choose to load sample data from Amazon Redshift Serverless or from an Amazon S3 bucket.

5. Under the Amazon Redshift Serverless default workgroup, expand the **sample_data_dev** database. There are three sample schemas corresponding to three sample datasets that you can load into the Amazon Redshift Serverless database. Choose the sample dataset that you want to load, and choose **Open sample notebooks**.



> **Note**
>
> A SQL notebook is a container for SQL and Markdown cells. You can use notebooks to organize, annotate, and share multiple SQL commands in a single document.

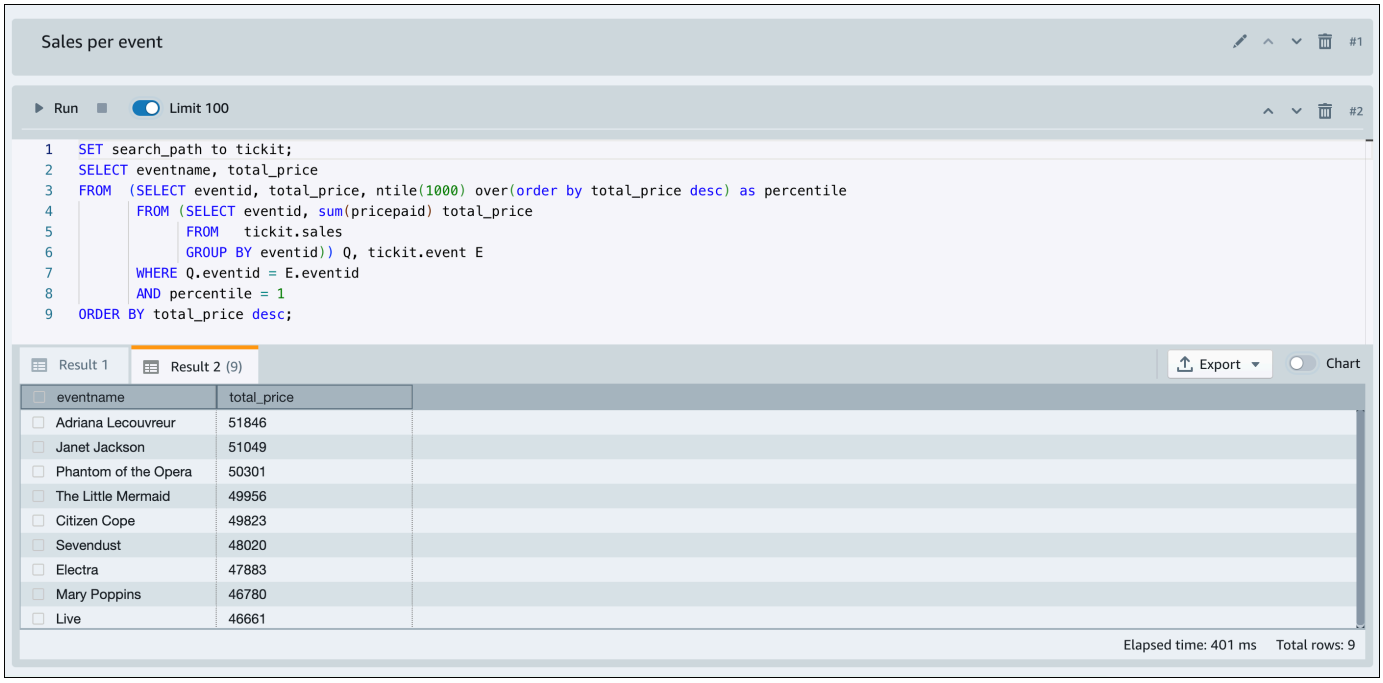
6. When loading data for the first time, query editor v2 will prompt you to create a sample database. Choose **Create**.



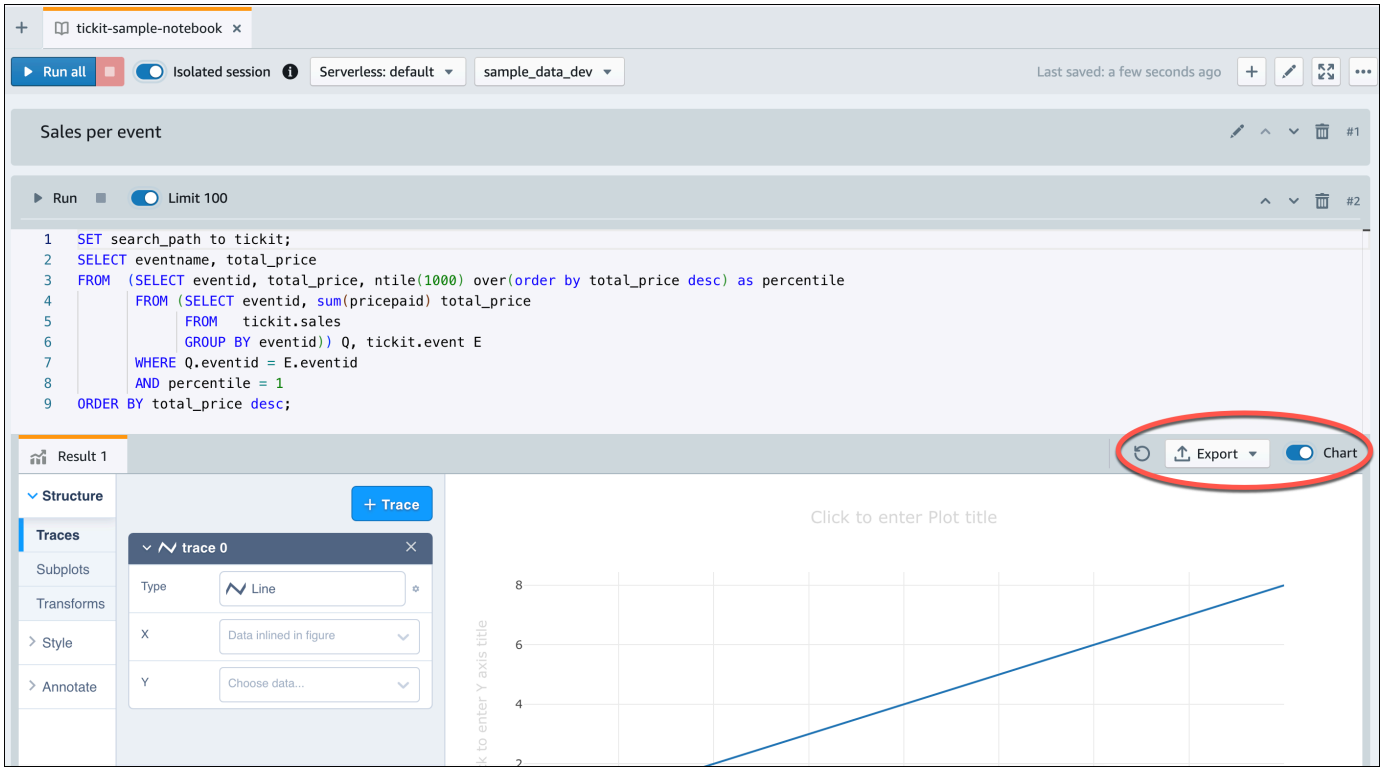
Running sample queries

After setting up Amazon Redshift Serverless, you can start using a sample dataset in Amazon Redshift Serverless. Amazon Redshift Serverless automatically loads the sample dataset, such as the tickit dataset, and you can immediately query the data.

- Once Amazon Redshift Serverless finishes loading the sample data, all of the sample queries are loaded in the editor. You can choose **Run all** to run all of the queries from the sample notebooks.



You can also export the results as a JSON or CSV file or view the results in a chart.



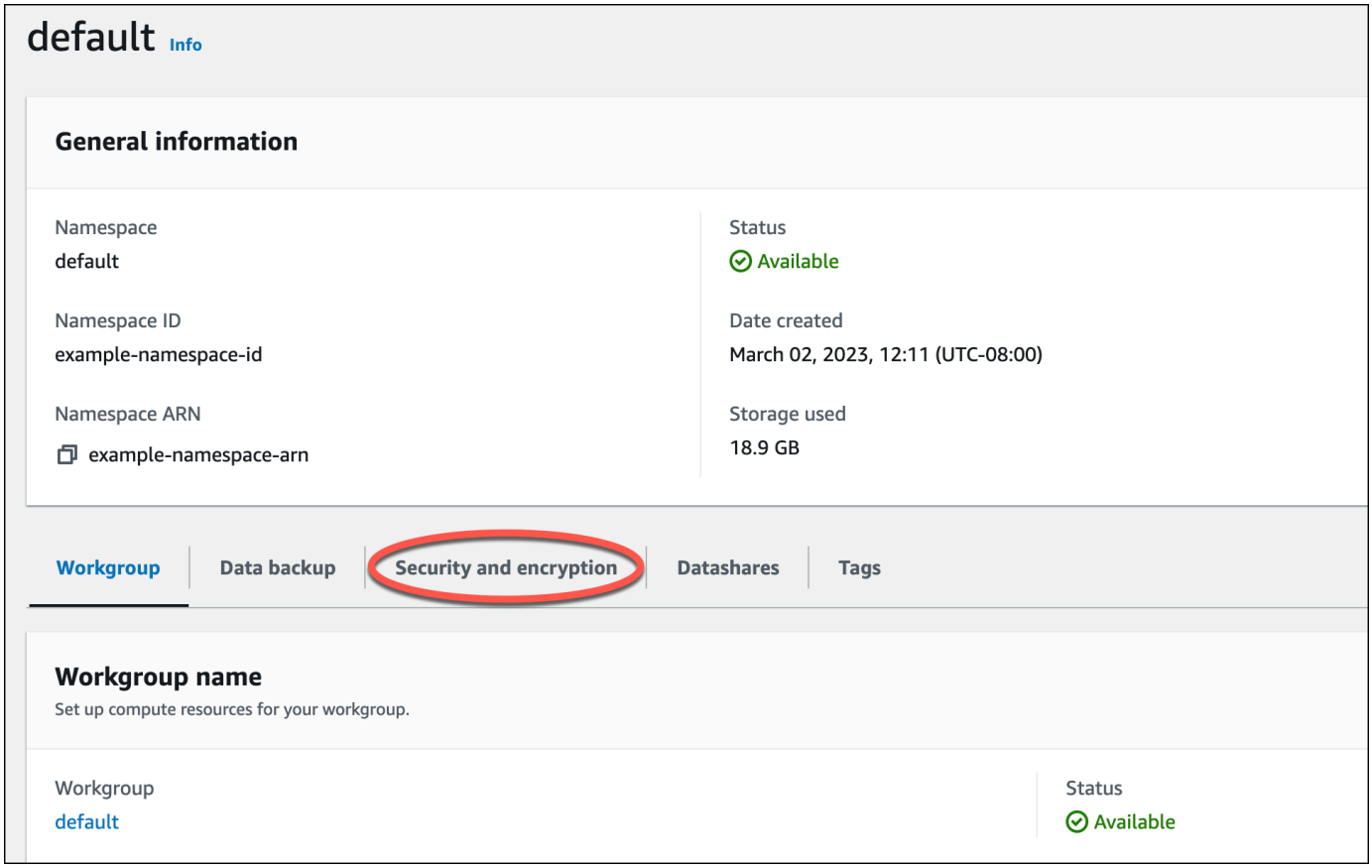
You can also load data from an Amazon S3 bucket. See the section called “Loading in data from Amazon S3” to learn more.

Loading in data from Amazon S3

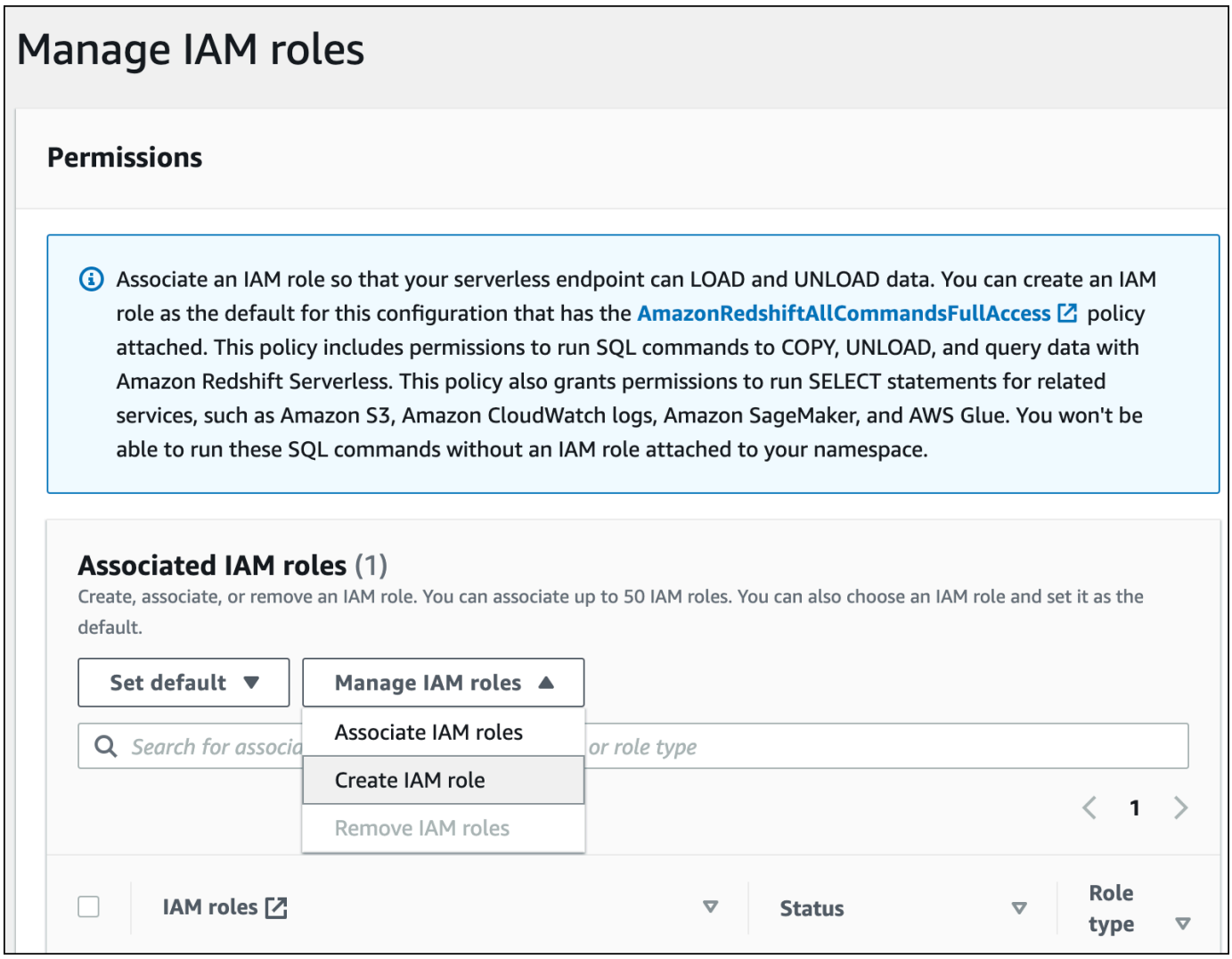
After creating your data warehouse, you can load data from Amazon S3.

At this point, you have a database named dev. Next, you will create some tables in the database, upload data to the tables, and try a query. For your convenience, the sample data that you load is available in an Amazon S3 bucket.

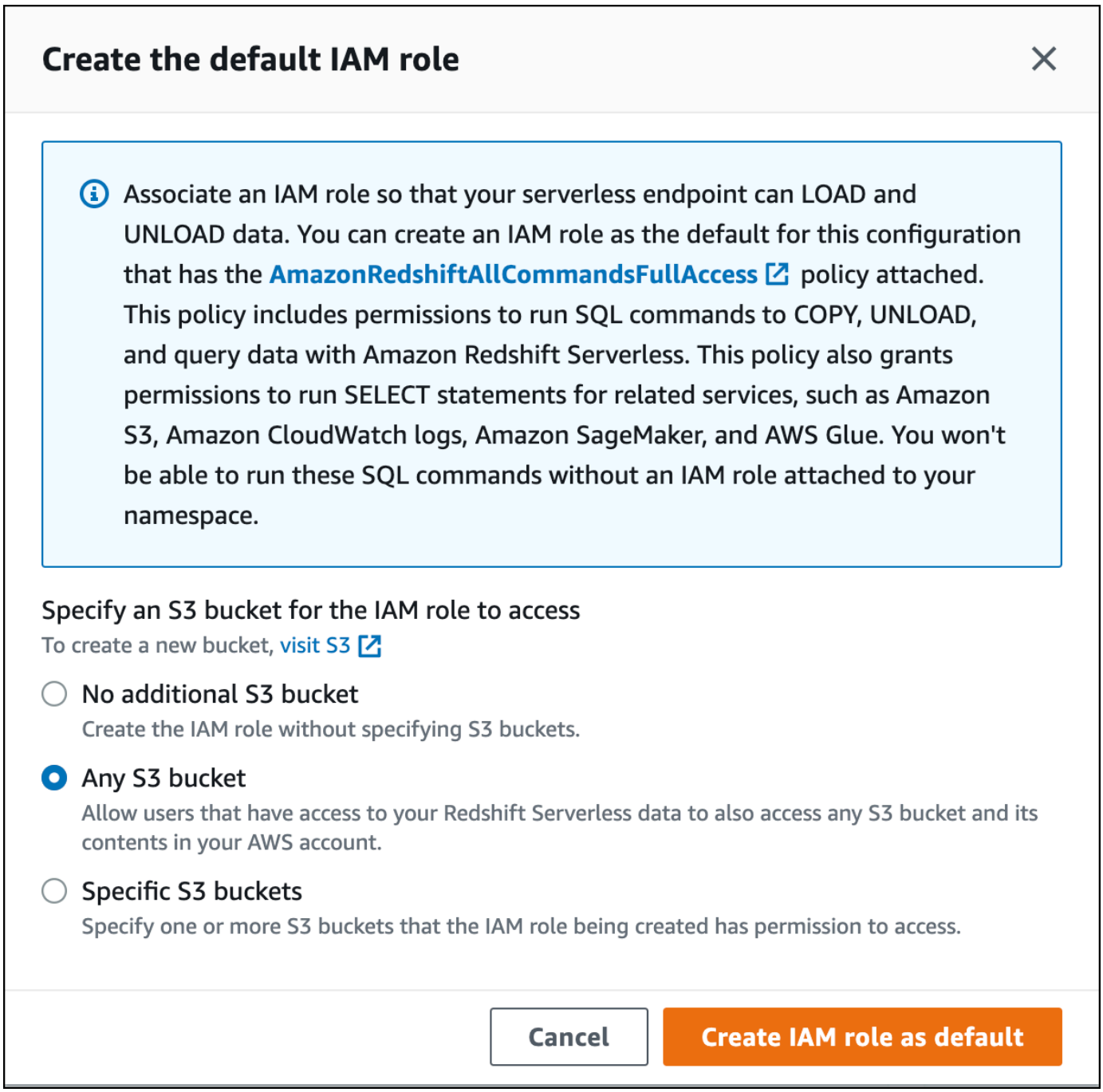
1. Before you can load data from Amazon S3, you must first create an IAM role with the necessary permissions and attach it to your serverless namespace. To do so, return to the Redshift Serverless console and choose **Namespace configuration**. From the navigation menu, choose your namespace, and then choose **Security and encryption**. Then, choose **Manage IAM roles**.



2. Expand the **Manage IAM roles** menu, and choose **Create IAM role**.



3. Choose the level of S3 bucket access that you want to grant to this role, and choose **Create IAM role as default**.



4. Choose **Save changes**. You can now load sample data from Amazon S3.

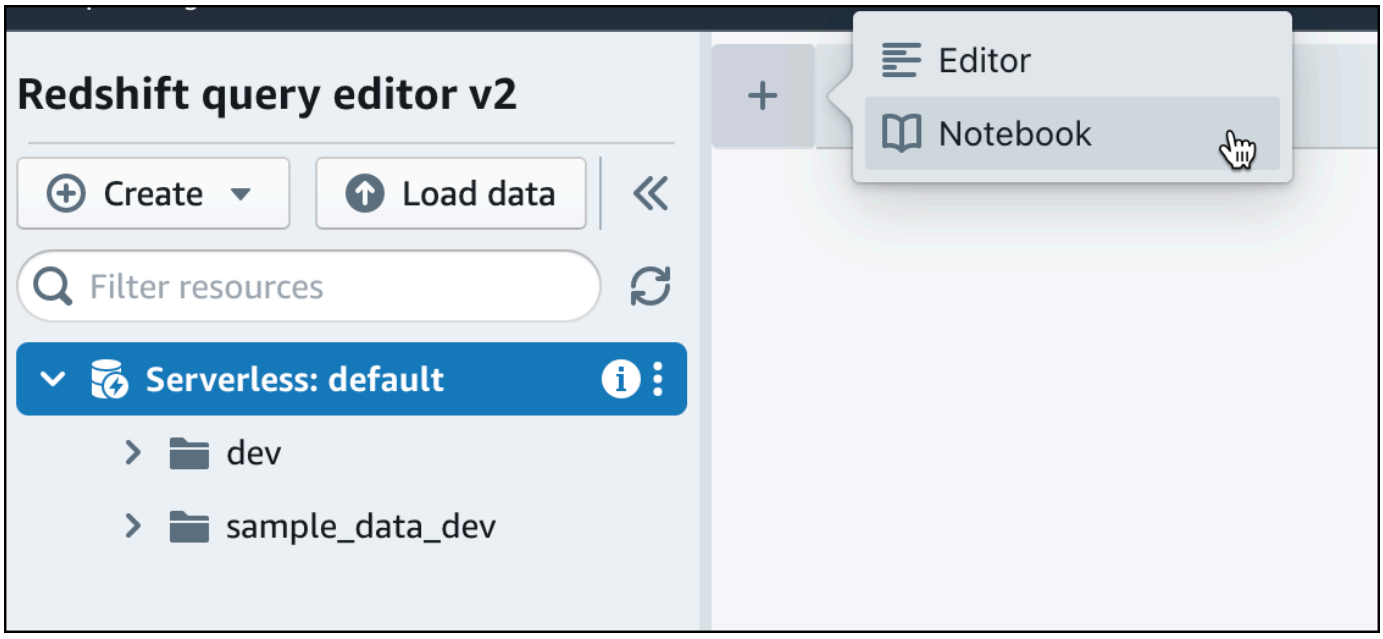
The following steps use data within a public Amazon Redshift S3 bucket, but you can replicate the same steps using your own S3 bucket and SQL commands.

Load sample data from Amazon S3

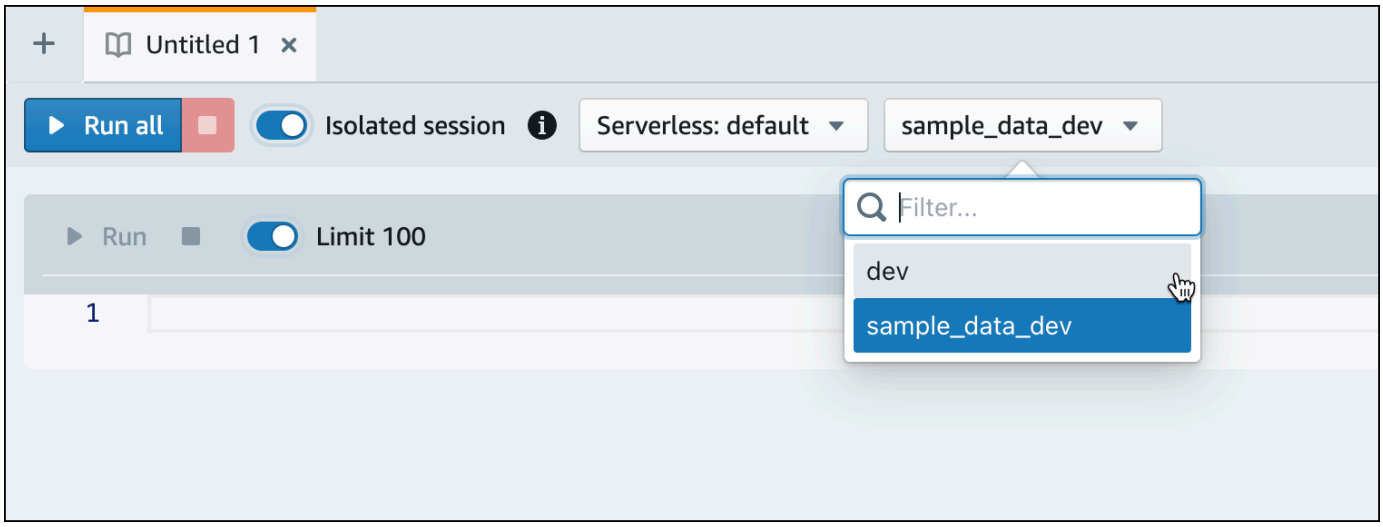
1. In query editor v2, choose

   Add, then choose **Notebook** to create a new SQL notebook.



2. Switch to the dev database.



3. Create tables.

   If you are using the query editor v2, copy and run the following create table statements to create tables in the dev database. For more information about the syntax, see CREATE TABLE in the *Amazon Redshift Database Developer Guide*.

```
create table users(
userid integer not null distkey sortkey,
username char(8),
firstname varchar(30),
lastname varchar(30),
city varchar(30),
state char(2),
email varchar(100),
phone char(14),
likesports boolean,
liketheatre boolean,
likeconcerts boolean,
likejazz boolean,
likeclassical boolean,
likeopera boolean,
likerock boolean,
likevegas boolean,
likebroadway boolean,
likemusicals boolean);

create table event(
eventid integer not null distkey,
venueid smallint not null,
catid smallint not null,
dateid smallint not null sortkey,
eventname varchar(200),
starttime timestamp);

create table sales(
salesid integer not null,
listid integer not null distkey,
sellerid integer not null,
buyerid integer not null,
eventid integer not null,
dateid smallint not null sortkey,
qtysold smallint not null,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp);
```

4. In the query editor v2, create a new SQL cell in your notebook.

5. Now use the COPY command in query editor v2 to load large datasets from Amazon S3 or Amazon DynamoDB into Amazon Redshift. For more information about COPY syntax, see COPY in the *Amazon Redshift Database Developer Guide*.

   You can run the COPY command with some sample data available in a public S3 bucket. Run the following SQL commands in the query editor v2.

```
COPY users
FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
DELIMITER '|'
TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
IGNOREHEADER 1
REGION 'us-east-1'
IAM_ROLE default;

COPY event
FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
DELIMITER '|'
TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
IGNOREHEADER 1
REGION 'us-east-1'
IAM_ROLE default;

COPY sales
FROM 's3://redshift-downloads/tickit/sales_tab.txt'
DELIMITER '\t'
TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
IGNOREHEADER 1
REGION 'us-east-1'
IAM_ROLE default;
```

6. After loading data, create another SQL cell in your notebook and try some example queries. For more information on working with the SELECT command, see SELECT in the *Amazon*

Redshift Developer Guide. To understand the sample data's structure and schemas, explore using the query editor v2.

```
-- Find top 10 buyers by quantity.
SELECT firstname, lastname, total_quantity
FROM   (SELECT buyerid, sum(qtysold) total_quantity
        FROM  sales
        GROUP BY buyerid
        ORDER BY total_quantity desc limit 10) Q, users
WHERE Q.buyerid = userid
ORDER BY Q.total_quantity desc;

-- Find events in the 99.9 percentile in terms of all time gross sales.
SELECT eventname, total_price
FROM  (SELECT eventid, total_price, ntile(1000) over(order by total_price desc) as
 percentile
       FROM (SELECT eventid, sum(pricepaid) total_price
             FROM   sales
             GROUP BY eventid)) Q, event E
       WHERE Q.eventid = E.eventid
       AND percentile = 1
ORDER BY total_price desc;
```

Now that you've loaded in data and ran some sample queries, you can explore other areas of Amazon Redshift Serverless. See the following list to learn more about how you can use Amazon Redshift Serverless.

- You can load data from an Amazon S3 bucket. See Loading data from Amazon S3 for more information.
- You can use the query editor v2 to load in data from a local character-separated file that is smaller than 5 MB. For more information, see Loading data from a local file.
- You can connect to Amazon Redshift Serverless with third-party SQL tools with the JDBC and ODBC driver. See Connecting to Amazon Redshift Serverless for more information.
- You can also use the Amazon Redshift Data API to connect to Amazon Redshift Serverless. See Using the Amazon Redshift Data API for more information.
- You can use your data in Amazon Redshift Serverless with Redshift ML to create machine learning models with the CREATE MODEL command. See Tutorial: Building customer churn models to learn how to build a Redshift ML model.

- You can query data from an Amazon S3 data lake without loading any data into Amazon Redshift Serverless. See Querying a data lake for more information.

Get started with Amazon Redshift provisioned data warehouses

If you are a first-time user of Amazon Redshift, we recommend that you read the following sections to help you get started using provisioned clusters. The basic flow of Amazon Redshift is to create provisioned resources, connect to Amazon Redshift, load sample data, and then run queries on the data. In this guide, you can choose to load sample data from Amazon Redshift or from an Amazon S3 bucket. The sample data is used throughout the Amazon Redshift documentation to demonstrate features.

This tutorial demonstrates how to use Amazon Redshift provisioned clusters, which are AWS data warehouse objects for which you manage system resources. You can also use Amazon Redshift with serverless workgroups, which are data warehouse objects that scale automatically in response to usage. To get started using Redshift Serverless, see Get started with Amazon Redshift Serverless data warehouses.

After you have created and signed in to the Amazon Redshift provisioned console, you can create and manage Amazon Redshift objects, including clusters, nodes, and databases. You can also run queries, view queries, and perform other SQL data definition language (DDL) and data manipulation language (DML) operations with a SQL client.

> **Important**
>
> The cluster that you provision for this exercise runs in a live environment. As long as it's running, it accrues charges to your AWS account. For pricing information, see the Amazon Redshift pricing page.
> To avoid unnecessary charges, delete your cluster when you are done with it. The final section of this chapter explains how to do so.

Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

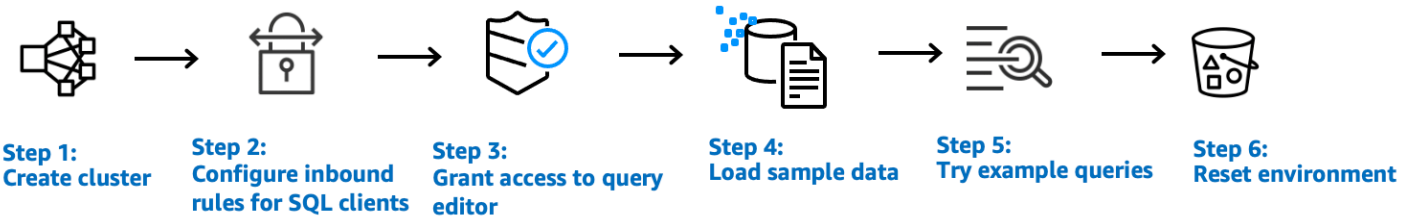
We recommend that you begin by going to the **Provisioned clusters dashboard** to get started using the Amazon Redshift console.

Depending on your configuration, the following items appear in the navigation pane of the Amazon Redshift provisioned console:

- **Redshift Serverless** – Access and analyze data without the need to set up, tune, and manage Amazon Redshift provisioned clusters.
- **Provisioned clusters dashboard** – View the list of clusters in your AWS Region, check **Cluster metrics** and **Query overview** for insights to metrics data (such as CPU utilization) and query information. Using these can help you determine if your performance data is abnormal over a specified time range.
- **Clusters** – View your list of clusters in this AWS Region, choose a cluster to start querying, or perform cluster-related actions. You can also create a new cluster from this page.
- **Query editor** – Run queries on databases hosted on your Amazon Redshift cluster. We recommend using the **Query editor v2** instead.
- **Query editor v2** – Amazon Redshift query editor v2 is a separate web-based SQL client application to author and run queries on your Amazon Redshift data warehouse. You can visualize your results in charts and collaborate by sharing your queries with others on your team.
- **Queries and loads** – Get information for reference or troubleshooting, such as a list of recent queries and the SQL text for each query.
- **Datashares** – As a producer account administrator, either authorize consumer accounts to access datashares or choose not to authorize access. To use an authorized datashare, a consumer account administrator can associate the datashare with either an entire AWS account or specific cluster namespaces in an account. An administrator can also decline a datashare.
- **Zero-ETL integrations** – Manage integrations that make transactional data available in Amazon Redshift after being written in supported sources.
- **IAM Identity Center connections** – Configure the connection between Amazon Redshift and IAM Identity Center.
- **Configurations** – Connect to Amazon Redshift clusters from SQL client tools over Java Database Connectivity (JDBC) and Open Database Connectivity (ODBC) connections. You can also set up an Amazon Redshift–managed virtual private cloud (VPC) endpoint. Doing so provides a private connection between a VPC based on the Amazon VPC service that contains a cluster and another VPC that is running a client tool.
- **AWS Partner Integration** – Create integration with a supported AWS Partner.
- **Advisor** – Get specific recommendations about changes you can make to your Amazon Redshift cluster to prioritize your optimizations.
- **AWS Marketplace** – Get information on other tools or AWS services that work with Amazon Redshift.

- **Alarms** – Create alarms on cluster metrics to view performance data and track metrics over a time period that you specify.
- **Events** – Track events and get reports on information such as the date the event occurred, a description, or the event source.
- **What's new** – View new Amazon Redshift features and product updates.

In this tutorial, you perform the following steps.



Topics

Signing up for AWS

If you don't already have an AWS account, sign up for one. If you already have an account, you can skip this prerequisite and use your existing account.

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

Determine firewall rules

> **Note**
>
> This tutorial assumes your cluster uses the default port 5439 and Amazon Redshift query editor v2 can be used to run SQL commands. It doesn't go into details about networking configurations or setting up a SQL client that might be necessary in your environment.

In some environments, you specify a port when you launch your Amazon Redshift cluster. You use this port along with the cluster's endpoint URL to access the cluster. You also create an inbound ingress rule in a security group to allow access through the port to your cluster.

If your client computer is behind a firewall, make sure that you know an open port that you can use. Using this open port, you can connect to the cluster from a SQL client tool and run queries. If you don't know an open port, work with someone who understands your network firewall rules to determine an open port in your firewall.

Though Amazon Redshift uses port 5439 by default, the connection doesn't work if that port isn't open in your firewall. You can't change the port number for your Amazon Redshift cluster after it's created. Thus, make sure that you specify an open port that works in your environment during the launch process.

Step 1: Create a sample Amazon Redshift cluster

In this tutorial, you walk through the process to create an Amazon Redshift cluster with a database. Then you load a dataset from Amazon S3 to tables in your database. You can use this sample cluster to evaluate the Amazon Redshift service.

Before you begin setting up an Amazon Redshift cluster, make sure that you complete any necessary prerequisites such as Signing up for AWS and Determine firewall rules.

For any operation that accesses data from another AWS resource, your cluster needs permission to access the resource and the data on the resource on your behalf. An example is using a SQL COPY command to load data from Amazon Simple Storage Service (Amazon S3). You provide those permissions by using AWS Identity and Access Management (IAM). You can do this through an IAM role that you create and attached to your cluster. For more information about credentials and access permissions, see Credentials and access permissions in the *Amazon Redshift Database Developer Guide*.

To create an Amazon Redshift cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

   > **Important**
   >
   > If you use IAM user credentials, ensure that you have the necessary permissions to perform the cluster operations. For more information, see Security in Amazon Redshift in the *Amazon Redshift Management Guide*.

2. On the AWS console, choose the AWS Region where you want to create the cluster.
3. On the navigation menu, choose **Clusters**, then choose **Create cluster**. The **Create cluster** page appears.
4. In the **Cluster configuration** section, specify values for **Cluster identifier**, **Node type**, and **Nodes**:

   - **Cluster identifier**: Enter **`examplecluster`** for this tutorial. This identifier must be unique. The identifier must be from 1–63 characters using as valid characters a–z (lowercase only) and - (hyphen).
   - Choose one of the following methods to size your cluster:

     > **Note**
     >
     > The following step assumes an AWS Region that supports RA3 node types. For a list of AWS Regions that support RA3 node types, see Overview of RA3 node types in the *Amazon Redshift Management Guide*. To learn more about the node specifications for each node type and size, see Node type details.

- If you don't know how large to size your cluster, choose **Help me choose**. Doing so opens a sizing calculator that asks you questions about the size and query characteristics of the data that you plan to store in your data warehouse.

  If you know the required size of your cluster (that is, the node type and number of nodes), choose **I'll choose**. Then choose the **Node type** and number of **Nodes** to size your cluster.

  For this tutorial, choose **ra3.4xlarge** for **Node type** and **2** for **Number of nodes**.

  If a choice for **AZ configuration** is available, choose **Single-AZ**.

- To use the sample dataset Amazon Redshift provides, in **Sample data**, choose **Load sample data**. Amazon Redshift loads the sample dataset Tickit to the default `dev` database and `public` schema.

5. In the **Database configuration** section, specify a value for **Admin user name**. For **Admin password**, choose from the following options:

   - **Generate a password** – Use a password generated by Amazon Redshift.
   - **Manually add an admin password** – Use your own password.
   - **Manage admin credentials in AWS Secrets Manager** – Amazon Redshift uses AWS Secrets Manager to generate and manage your admin password. Using AWS Secrets Manager to generate and manage your password's secret incurs a fee. For information on AWS Secrets Manager pricing, see AWS Secrets Manager Pricing.

   For this tutorial, use these values:

   - **Admin user name**: Enter **`awsuser`**.
   - **Admin user password**: Enter **`Changeit1`** for the password.

6. For this tutorial, create an IAM role and set it as the default for your cluster, as described following. There can only be one default IAM role set for a cluster.

   a. Under **Cluster permissions**, for **Manage IAM roles**, choose **Create IAM role**.

   b. Specify an Amazon S3 bucket for the IAM role to access by one of the following methods:

      - Choose **No additional Amazon S3 bucket** to allow the created IAM role to access only the Amazon S3 buckets that are named as `redshift`.

- Choose **Any Amazon S3 bucket** to allow the created IAM role to access all Amazon S3 buckets.
- Choose **Specific Amazon S3 buckets** to specify one or more Amazon S3 buckets for the created IAM role to access. Then choose one or more Amazon S3 buckets from the table.

c. Choose **Create IAM role as default**. Amazon Redshift automatically creates and sets the IAM role as the default for your cluster.

Because you created your IAM role from the console, it has the `AmazonRedshiftAllCommandsFullAccess` policy attached. This allows Amazon Redshift to copy, load, query, and analyze data from Amazon resources in your IAM account.

For information on how to manage the default IAM role for a cluster, see Creating an IAM role as default for Amazon Redshift in the *Amazon Redshift Management Guide*.

7. (Optional) In the **Additional configurations** section, turn off **Use defaults** to modify **Network and security**, **Database configuration**, **Maintenance**, **Monitoring**, and **Backup** settings.

   In some cases, you might create your cluster with the **Load sample data** option and want to turn on enhanced Amazon VPC routing. If so, the cluster in your virtual private cloud (VPC) requires access to the Amazon S3 endpoint for data to be loaded.

   To make the cluster publicly accessible, you can do one of two things. You can configure a network address translation (NAT) address in your VPC for the cluster to access the internet. Or you can configure an Amazon S3 VPC endpoint in your VPC. For more information about enhanced Amazon VPC routing, see Enhanced Amazon VPC routing in the *Amazon Redshift Management Guide*.

8. Choose **Create cluster**. Wait for your cluster to be created with `Available` status on the **Clusters** page.

Step 2: Configure inbound rules for SQL clients

> **Note**
>
> We recommend you skip this step and access your cluster using Amazon Redshift query editor v2.

Later in this tutorial, you access your cluster from within a virtual private cloud (VPC) based on the Amazon VPC service. However, if you use an SQL client from outside your firewall to access the cluster, make sure that you grant inbound access.

To check your firewall and grant inbound access to your cluster

1. Check your firewall rules if your cluster needs to be accessed from outside a firewall. For example, your client might be an Amazon Elastic Compute Cloud (Amazon EC2) instance or an external computer.

   For more information on firewall rules, see Security group rules in the *Amazon EC2 User Guide*.
2. To access from an Amazon EC2 external client, add an ingress rule to the security group attached to your cluster that allows inbound traffic. You add Amazon EC2 security group rules in the Amazon EC2 console. For example, a CIDR/IP of 192.0.2.0/24 allows clients in that IP address range to connect to your cluster. Find out the correct CIDR/IP for your environment.

Step 3: Grant access to a SQL client and run queries

To query databases hosted by your Amazon Redshift cluster, you have several options for SQL clients. These include:

- Connect to your cluster and run queries using Amazon Redshift query editor v2.

  If you use query editor v2, you don't have to download and set up an SQL client application. You launch Amazon Redshift query editor v2 from the Amazon Redshift console.
- Connect to your cluster using RSQL. For more information, see Connecting with Amazon Redshift RSQL in the *Amazon Redshift Management Guide*.
- Connect to your cluster through a SQL client tool, such as SQL Workbench/J. For more information, see Connect to your cluster by using SQL Workbench/J in the *Amazon Redshift Management Guide*.

This tutorial uses Amazon Redshift query editor v2 as an easy way to run queries on databases hosted by your Amazon Redshift cluster. After creating your cluster, you can immediately run queries. For details about considerations when using the Amazon Redshift query editor v2, see Considerations when working with query editor v2 in the *Amazon Redshift Management Guide*.

Granting access to the query editor v2

The first time an administrator configures query editor v2 for your AWS account, they choose the AWS KMS key that is used to encrypt query editor v2 resources. Amazon Redshift query editor v2 resources include saved queries, notebooks, and charts. By default, an AWS owned key is used to encrypt resources. Alternatively, an administrator can use a customer managed key by choosing the Amazon Resource Name (ARN) for the key in the configuration page. After you configure an account, AWS KMS encryption settings can't be changed. For more information, see Configuring your AWS account in the *Amazon Redshift Management Guide*.

To access the query editor v2, you need permission. An administrator can attach one of the AWS managed policies for Amazon Redshift query editor v2 to the IAM role or user to grant permissions. These AWS managed policies are written with different options that control how tagging resources allows sharing of queries. You can use the IAM console (https://console.aws.amazon.com/iam/) to attach IAM policies. For more information about these policies, see Accessing the query editor v2 in the *Amazon Redshift Management Guide*.

You can also create your own policy based on the permissions allowed and denied in the provided managed policies. If you use the IAM console policy editor to create your own policy, choose **SQL Workbench** as the service for which you create the policy in the visual editor. The query editor v2 uses the service name AWS SQL Workbench in the visual editor and IAM Policy Simulator.

For more information, see Working with query editor v2 in the *Amazon Redshift Management Guide*.

Step 4: Load data from Amazon S3 to Amazon Redshift

After creating your cluster, you can load data from Amazon S3 to your database tables. There are multiple ways to load data from Amazon S3.

- You can use a SQL client to run the SQL CREATE TABLE command to create a table in your database and then use the SQL COPY command to load data from Amazon S3. The Amazon Redshift query editor v2 is a SQL client.
- You can use the Amazon Redshift query editor v2 load wizard.

This tutorial demonstrates how to use the Amazon Redshift query editor v2 to run SQL commands to CREATE tables and COPY data. Launch **Query editor v2** from the Amazon Redshift console

navigation pane. Within query editor v2 create a connection to `examplecluster` cluster and database named `dev` with your admin user `awsuser`. For this tutorial choose **Temporary credentials using a database user name** when you create the connection. For details on using Amazon Redshift query editor v2, see Connecting to an Amazon Redshift database in the *Amazon Redshift Management Guide*.

Loading data from Amazon S3 using SQL commands

On the query editor v2 query editor pane, confirm you are connected to the `examplecluster` cluster and `dev` database. Next, create tables in the database and load data to the tables. For this tutorial, the data that you load is available in an Amazon S3 bucket accessible from many AWS Regions.

The following procedure creates tables and loads data from a public Amazon S3 bucket.

Use the Amazon Redshift query editor v2 to copy and run the following create table statement to create a table in the `public` schema of the `dev` database. For more information about the syntax, see CREATE TABLE in the *Amazon Redshift Database Developer Guide*.

To create and load data using a SQL client such as query editor v2

1. Run the following SQL command to CREATE the `sales` table.

```
  drop table if exists sales;
  create table sales(
salesid integer not null,
listid integer not null distkey,
sellerid integer not null,
buyerid integer not null,
eventid integer not null,
dateid smallint not null sortkey,
qtysold smallint not null,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp);
```

2. Run the following SQL command to CREATE the `date` table.

```
drop table if exists date;
```

```
create table date(
 dateid smallint not null distkey sortkey,
 caldate date not null,
 day character(3) not null,
 week smallint not null,
 month character(5) not null,
 qtr character(5) not null,
 year smallint not null,
 holiday boolean default('N'));
```

3. Load the `sales` table from Amazon S3 using the COPY command.

> **Note**
>
> We recommend using the COPY command to load large datasets into Amazon Redshift from Amazon S3. For more information about COPY syntax, see COPY in the *Amazon Redshift Database Developer Guide*.

Provide authentication for your cluster to access Amazon S3 on your behalf to load the sample data. You provide authentication by referencing the IAM role that you created and set as the `default` for your cluster when you chose **Create IAM role as default** when you created the cluster.

Load the `sales` table using the following SQL command. You can optionally download and view from Amazon S3 the source data for the `sales` table. .

```
COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    REGION 'us-east-1'
    IAM_ROLE default;
```

4. Load the `date` table using the following SQL command. You can optionally download and view from Amazon S3 the source data for the `date` table. .

```
COPY date
    FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
    DELIMITER '|'
    REGION 'us-east-1'
```

```
    IAM_ROLE default;
```

Loading data from Amazon S3 using the query editor v2

This section describes loading your own data into an Amazon Redshift cluster. The query editor v2 simplifies loading data when using the **Load data** wizard. The COPY command generated and used in the query editor v2 **Load data** wizard supports many of the parameters available to the COPY command syntax to load data from Amazon S3. For information about the COPY command and its options used to copy load from Amazon S3, see COPY from Amazon Simple Storage Service in the *Amazon Redshift Database Developer Guide*.

To load your own data from Amazon S3 to Amazon Redshift, Amazon Redshift requires an IAM role that has the required privileges to load data from the specified Amazon S3 bucket.

To load your own data from Amazon S3 to Amazon Redshift, you can use the query editor v2 load data wizard. For information on how to use the load data wizard, see Loading data from Amazon S3 in the *Amazon Redshift Management Guide*.

Create TICKIT data in your cluster

TICKIT is a sample database that you can optionally load into your Amazon Redshift cluster for the purposes of learning how to query data in Amazon Redshift. You can create the full set of TICKIT tables and load data into your cluster in the following ways:

- When you create a cluster in the Amazon Redshift console, you have the option to load sample TICKIT data at the same time. On the Amazon Redshift console, choose **Clusters**, **Create cluster**. In the **Sample data** section, select **Load sample data** Amazon Redshift loads its sample dataset to your Amazon Redshift cluster `dev` database automatically during cluster creation.
- To connect to an existing cluster, do the following:
  - In the Amazon Redshift console, choose **Clusters** from the navigation bar.
  - Choose your cluster from the **Clusters** pane.
  - Choose **Query data**, **Query in query editor v2**.
  - Expand **examplecluster** in the resources list. If this is your first time connecting to your cluster, the **Connect to examplecluster** appears. Choose **Database username and password**. Leave the database as **`dev`**. Specify **`awsuser`** for the username and **`Changeit1`** for the password.
  - Choose **Create connection**.

- With Amazon Redshift query editor v2, you can load TICKIT data into a sample database named **sample_data_dev**. Choose the **sample_data_dev** database in the resource list. Next to the **tickit** node, choose the **Open sample notebooks** icon. Confirm that you want to create the sample database.
- Amazon Redshift query editor v2 creates the sample database along with a sample notebook named **tickit-sample-notebook**. You can choose **Run all** to run this notebook to query data in the sample database.

To view details about the TICKIT data, see Sample database in the *Amazon Redshift Database Developer Guide*.

Step 5: Try example queries using the query editor

To set up and use the Amazon Redshift query editor v2 to query a database, see Working with query editor v2 in the *Amazon Redshift Management Guide*.

Now, try some example queries, as shown following. To create new queries in the query editor v2, choose the **+** icon in the upper right of the query pane, and choose **SQL**. A new query page appears where you can copy and paste the following SQL queries.

> **Note**
>
> Be sure to run the first query in the notebook first, which sets the `search_path` server configuration value to the `tickit` schema using the following SQL command:
>
> ```
> set search_path to tickit;
> ```

For more information on working with the SELECT command, see SELECT in the *Amazon Redshift Database Developer Guide*.

```
-- Get definition for the sales table.
SELECT *
FROM pg_table_def
WHERE tablename = 'sales';
```

```
-- Find total sales on a given calendar date.
SELECT sum(qtysold)
```

```
FROM   sales, date
WHERE  sales.dateid = date.dateid
AND    caldate = '2008-01-05';
```

```
-- Find top 10 buyers by quantity.
SELECT firstname, lastname, total_quantity
FROM   (SELECT buyerid, sum(qtysold) total_quantity
        FROM  sales
        GROUP BY buyerid
        ORDER BY total_quantity desc limit 10) Q, users
WHERE Q.buyerid = userid
ORDER BY Q.total_quantity desc;
```

```
-- Find events in the 99.9 percentile in terms of all time gross sales.
SELECT eventname, total_price
FROM  (SELECT eventid, total_price, ntile(1000) over(order by total_price desc) as
 percentile
       FROM (SELECT eventid, sum(pricepaid) total_price
             FROM   sales
             GROUP BY eventid)) Q, event E
       WHERE Q.eventid = E.eventid
       AND percentile = 1
ORDER BY total_price desc;
```

Step 6: Reset your environment

In the previous steps, you have successfully created an Amazon Redshift cluster, loaded data into tables, and queried data using a SQL client such as Amazon Redshift query editor v2.

When you have completed this tutorial, we suggest that you reset your environment to the previous state by deleting your sample cluster. You continue to incur charges for the Amazon Redshift service until you delete the cluster.

However, you might want to keep the sample cluster running if you intend to try tasks in other Amazon Redshift guides or tasks described in Run commands to define and use a database in your data warehouse.

To delete a cluster

1. Sign in to the AWS Management Console and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/.

2. On the navigation menu, choose **Clusters** to display your list of clusters.
3. Choose the `examplecluster` cluster. For **Actions**, choose **Delete**. The **Delete examplecluster?** page appears.
4. Confirm the cluster to be deleted, uncheck the **Create final snapshot** setting, then enter **`delete`** to confirm deletion. Choose **Delete cluster**.

On the cluster list page, the cluster status is updated as the cluster is deleted.

After you complete this tutorial, you can find more information about Amazon Redshift and next steps in Additional resources to learn about Amazon Redshift.

Run commands to define and use a database in your data warehouse

Both Redshift Serverless data warehouses and Amazon Redshift provisoned data warehouses contain databases. After you have launched your data warehouse, you can manage most database actions using SQL commands. With few exceptions, the functionality and syntax of SQL is the same for all Amazon Redshift databases. For details of SQL commands available with Amazon Redshift, see SQL commands in the *Amazon Redshift Database Developer Guide*.

When you create your data warehouse, in most scenarios, Amazon Redshift also creates the default `dev` database. After you connect to the `dev` database, you can create another database.

The following sections walk through common database tasks when working with Amazon Redshift databases. The tasks begin with creating a database and if you continue to the last task you can delete all the resources you create by dropping the database.

The examples in this section assume the following:

- You have created an Amazon Redshift data warehouse.
- You have established a connection to the data warehouse from your SQL client tool, such as the Amazon Redshift query editor v2. For more information about query editor v2, see Querying a database using the Amazon Redshift query editor v2 in the *Amazon Redshift Management Guide*.

Topics

Connecting to Amazon Redshift data warehouses

To connect to Amazon Redshift clusters, from the Amazon Redshift console **Clusters** page, expand **Connect to Amazon Redshift clusters** and do one of the following:

- Choose **Query data** to use the query editor v2 to run queries on databases hosted by your Amazon Redshift cluster. After creating your cluster, you can immediately run queries by using the query editor v2.

  For more information, see Querying a database using the Amazon Redshift query editor v2 in the *Amazon Redshift Management Guide*.
- In **Work with your client tools**, choose your cluster and connect to Amazon Redshift from your client tools using JDBC or ODBC drivers by copying the JDBC or ODBC driver URL. Use this URL from your client computer or instance. Code your applications to use JDBC or ODBC data access API operations, or use SQL client tools that support either JDBC or ODBC.

  For more information on how to find your cluster connection string, see Finding your cluster connection string.
- If your SQL client tool requires a driver, you can **Choose your JDBC or ODBC driver** to download an operating system-specific driver to connect to Amazon Redshift from your client tools.

  For more information on how to install the appropriate driver for your SQL client, see Configuring a JDBC driver version 2.2 connection.

  For more information on how to configure an ODBC connection, see Configuring an ODBC connection.

To connect to Redshift Serverless data warehouse, from the Amazon Redshift console **Serverless dashboard** page, do one of the following:

- Use the Amazon Redshift query editor v2 to run queries on databases hosted by your Redshift Serverless data warehouse. After creating your data warehouse, you can immediately run queries by using the query editor v2.

  For more information, see Querying a database using the Amazon Redshift query editor v2.
- Connect to Amazon Redshift from your client tools using JDBC or ODBC drivers by copying the JDBC or ODBC driver URL.

To work with data in your data warehouse, you need JDBC or ODBC drivers for connectivity from your client computer or instance. Code your applications to use JDBC or ODBC data access API operations, or use SQL client tools that support either JDBC or ODBC.

For more information on how to find your connection string, see Connecting to Redshift Serverless in the *Amazon Redshift Management Guide*.

Create a database

After you verify that your data warehouse is up and running, you can create a database. This database is where you actually create tables, load data, and run queries. A data warehouse can host multiple databases. For example, you can have a database for sales data named SALESDB and a database for orders data named ORDERSDB in the same data warehouse.

To create a database named **SALESDB**, run the following command in your SQL client tool.

```
CREATE DATABASE salesdb;
```

> **Note**
>
> After running the command, make sure to refresh your SQL client tool list of objects in your data warehouse to see the new `salesdb`.

For this exercise, accept the defaults. For information about more command options, see CREATE DATABASE in the *Amazon Redshift Database Developer Guide*. To delete a database and its contents, see DROP DATABASE in the *Amazon Redshift Database Developer Guide*.

After you have created the SALESDB database, you can connect to the new database from your SQL client. Use the same connection parameters as you used for your current connection, but change the database name to SALESDB.

Create a user

By default, only the admin user that you created when you launched the data warehouse has access to the default database in the data warehouse. To grant other users access, create one or

more accounts. Database user accounts are global across all the databases in a data warehouse, and not per individual database.

Use the CREATE USER command to create a new user. When you create a new user, you specify the name of the new user and a password. We recommend that you specify a password for the user. It must have 8–64 characters, and it must include at least one uppercase letter, one lowercase letter, and one numeral.

For example, to create a user named **GUEST** with password **ABCd4321**, run the following command.

```
CREATE USER GUEST PASSWORD 'ABCd4321';
```

To connect to the SALESDB database as the GUEST user, use the same password when you created the user, such as ABCd4321.

For information about other command options, see CREATE USER in the *Amazon Redshift Database Developer Guide*.

Create a schema

After you create a new database, you can create a new schema in the current database. A *schema* is a namespace that contains named database objects such as tables, views, and user-defined functions (UDFs). A database can contain one or multiple schemas, and each schema belongs to only one database. Two schemas can have different objects that share the same name.

You can create multiple schemas in the same database to organize data the way that you want or to group your data functionally. For example, you can create a schema to store all your staging data and another schema to store all the reporting tables. You can also create different schemas to store data relevant to different business groups that are in the same database. Each schema can store different database objects, such as tables, views, and user-defined functions (UDFs). In addition, you can create schemas with the AUTHORIZATION clause. This clause gives ownership to a specified user or sets a quota on the maximum amount of disk space that the specified schema can use.

Amazon Redshift automatically creates a schema called `public` for every new database. When you don't specify the schema name while creating database objects, the objects go into the `public` schema.

To access an object in a schema, qualify the object by using the `schema_name.table_name` notation. The qualified name of the schema consists of the schema name and table name

separated by a dot. For example, you might have a `sales` schema that has a `price` table and an `inventory` schema that also has a `price` table. When you refer to the `price` table, you must qualify it as `sales.price` or `inventory.price`.

The following example creates a schema named **SALES** for the user GUEST.

```
CREATE SCHEMA SALES AUTHORIZATION GUEST;
```

For information about more command options, see CREATE SCHEMA in the *Amazon Redshift Database Developer Guide*.

To view the list of schemas in your database, run the following command.

```
select * from pg_namespace;
```

The output should look similar to the following.

```
  nspname             | nspowner |         nspacl
----------------------+----------+-------------------------
  sales               |  100     |
  pg_toast            |   1      |
  pg_internal         |   1      |
  catalog_history     |   1      |
  pg_temp_1           |   1      |
  pg_catalog          |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
  public              |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
  information_schema  |   1      | {rdsdb=UC/rdsdb,=U/rdsdb}
```

For more information on how to query catalog tables, see Querying the catalog tables in the *Amazon Redshift Database Developer Guide.*

Use the GRANT statement to give permissions to users for the schemas.

The following example grants privilege to the GUEST user to select data from all tables or views in the SALES schema using a SELECT statement.

```
GRANT SELECT ON ALL TABLES IN SCHEMA SALES TO GUEST;
```

The following example grants all available privileges at one time to the GUEST user.

```
GRANT ALL ON SCHEMA SALES TO GUEST;
```

Create a table

After you create your new database, create tables to hold your data. Specify the column information when you create the table.

For example, to create a table named **DEMO**, run the following command.

```
CREATE TABLE Demo (
  PersonID int,
  City varchar (255)
);
```

By default, new database objects, such as tables, are created in the default schema named `public` created during data warehouse creation. You can use another schema to create database objects. For more information about schemas, see Managing database security in the *Amazon Redshift Database Developer Guide*.

You can also create a table using the `schema_name.object_name` notation to create the table in the SALES schema.

```
CREATE TABLE SALES.DEMO (
  PersonID int,
  City varchar (255)
);
```

To view and inspect schemas and their tables, you can use the Amazon Redshift query editor v2 . Or you can see the list of tables in schemas using system views. For more information, see Query the system tables and views.

The `encoding`, `distkey`, and `sortkey` columns are used by Amazon Redshift for parallel processing. For more information about designing tables that incorporate these elements, see Amazon Redshift best practices for designing tables.

Insert data rows into a table

After you create a table, insert rows of data into that table.

> **Note**
>
> The INSERT command inserts rows into a table. For standard bulk loads, use the COPY command. For more information, see Use a COPY command to load data.

For example, to insert values into the DEMO table, run the following command.

```
INSERT INTO DEMO VALUES (781, 'San Jose'), (990, 'Palo Alto');
```

To insert data into a table that's in a specific schema, run the following command.

```
INSERT INTO SALES.DEMO VALUES (781, 'San Jose'), (990, 'Palo Alto');
```

Select data from a table

After you create a table and populate it with data, use a SELECT statement to display the data contained in the table. The SELECT * statement returns all the column names and row values for all of the data in a table. Using SELECT is a good way to verify that recently added data was correctly inserted into the table.

To view the data that you entered in the **DEMO** table, run the following command.

```
SELECT * from DEMO;
```

The result should look like the following.

```
 personid |   city
----------+-----------
      781 | San Jose
      990 | Palo Alto
(2 rows)
```

For more information about using the SELECT statement to query tables, see SELECT.

Load data

Many of the examples in this guide use the TICKIT sample dataset. You can download the file tickitdb.zip, which contains individual sample data files. You can then upload the sample data to your own Amazon S3 bucket.

To load the sample data for your database, first create the tables. Then use the COPY command to load the tables with sample data that is stored in an Amazon S3 bucket. For steps to create tables and load sample data, see Step 4: Load data from Amazon S3 to Amazon Redshift.

Query the system tables and views

In addition to the tables that you create, your data warehouse contains a number of system tables and views. These tables and views contain information about your installation and the various queries and processes that are running on the system. You can query these system tables and views to collect information about your database. For more information, see System tables and views reference in the *Amazon Redshift Database Developer Guide*. The description for each table or view indicates whether a table is visible to all users or only to superusers. Log in as a superuser to query tables that are visible only to superusers.

View a list of table names

To view a list of all tables in a schema, you can query the PG_TABLE_DEF system catalog table. You can first examine the setting for `search_path`.

```
SHOW search_path;
```

The result should look similar to the following,

```
  search_path
---------------
 $user, public
```

The following example adds the SALES schema to the search path and shows all the tables in the SALES schema.

```
set search_path to '$user', 'public', 'sales';
```

```
SHOW search_path;

      search_path
-----------------------
 "$user", public, sales


select * from pg_table_def where schemaname = 'sales';

 schemaname | tablename |  column  |          type          | encoding | distkey |
 sortkey | notnull
------------+-----------+----------+------------------------+----------+---------
+---------+---------
 sales      | demo      | personid | integer                | az64     | f       |
  0 | f
 sales      | demo      | city     | character varying(255) | lzo      | f       |
  0 | f
```

The following example shows a list of all tables called DEMO in all schemas on the current database.

```
set search_path to '$user', 'public', 'sales';
select * from pg_table_def where tablename = 'demo';

 schemaname | tablename |  column  |          type          | encoding | distkey |
 sortkey | notnull
------------+-----------+----------+------------------------+----------+---------
+---------+---------
 public     | demo      | personid | integer                | az64     | f       |
  0 | f
 public     | demo      | city     | character varying(255) | lzo      | f       |
  0 | f
 sales      | demo      | personid | integer                | az64     | f       |
  0 | f
 sales      | demo      | city     | character varying(255) | lzo      | f       |
  0 | f
```

For more information, see PG_TABLE_DEF.

You can also use the Amazon Redshift query editor v2 to view all the tables in a specified schema by first choosing a database that you want to connect to.

View users

You can query the PG_USER catalog to view a list of all users, along with the user ID (USESYSID) and user privileges.

```
SELECT * FROM pg_user;

  usename   | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 rdsdb      |        1 | true        | true     | true      | ******** | infinity |
 awsuser    |      100 | true        | true     | false     | ******** |          |
 guest      |      104 | true        | false    | false     | ******** |          |
```

The user name `rdsdb` is used internally by Amazon Redshift to perform routine administrative and maintenance tasks. You can filter your query to show only user-defined user names by adding `where usesysid > 1` to your SELECT statement.

```
SELECT * FROM pg_user WHERE usesysid > 1;

  usename   | usesysid | usecreatedb | usesuper | usecatupd |  passwd  | valuntil |
 useconfig
------------+----------+-------------+----------+-----------+----------+----------
+-----------
 awsuser    |      100 | true        | true     | false     | ******** |          |
 guest      |      104 | true        | false    | false     | ******** |          |
```

View recent queries

In the previous example, the user ID (user_id) for `adminuser` is 100. To list the four most recent queries run by `adminuser`, you can query the SYS_QUERY_HISTORY view.

You can use this view to find the query ID (query_id) or process ID (session_id) for a recently run query. You can also use this view to check how long it took a query to complete. SYS_QUERY_HISTORY includes the first 4,000 characters of the query string (query_text) to help you locate a specific query. Use the LIMIT clause with your SELECT statement to limit the results.

```
SELECT query_id, session_id, elapsed_time, query_text
FROM sys_query_history
```

```
WHERE user_id = 100
ORDER BY start_time desc
LIMIT 4;
```

The result look something like the following.

```
 query_id |  session_id  |  elapsed_time |   query_text
----------+--------------+---------------
+-----------------------------------------------------------------
 892      |    21046     |       55868   | SELECT query, pid, elapsed, substring
 from ...
 620      |    17635     |     1296265   | SELECT query, pid, elapsed, substring
 from ...
 610      |    17607     |       82555   | SELECT * from DEMO;
 596      |    16762     |      226372   | INSERT INTO DEMO VALUES (100);
```

Determine the session ID of a running query

To retrieve system table information about a query, you might need to specify the session ID (process ID) associated with that query. Or, you might need to find the session ID for a query that is still running. For example, you need the session ID if you need to cancel a query that is taking too long to run on a provisioned cluster. You can query the STV_RECENTS system table to obtain a list of session IDs for running queries, along with the corresponding query string. If your query returns multiple session, you can look at the query text to determine which session ID you need.

To determine the session ID of a running query, run the following SELECT statement.

```
SELECT session_id, user_id, start_time, query_text
FROM sys_query_history
WHERE status='running';
```

Cancel a query

If you run a query that is taking too long or is consuming excessive resources, cancel the query. For example, create a list of ticket sellers that includes the seller's name and quantity of tickets sold. The following query selects data from the SALES table and USERS table and joins the two tables by matching SELLERID and USERID in the WHERE clause.

```
SELECT sellerid, firstname, lastname, sum(qtysold)
FROM sales, users
```

```
WHERE sales.sellerid = users.userid
GROUP BY sellerid, firstname, lastname
ORDER BY 4 desc;
```

The result looks something like the following.

```
 sellerid | firstname | lastname | sum
----------+-----------+----------+------
  48950   |   Nayda   |   Hood   | 184
  19123   |   Scott   | Simmons  | 164
  20029   |    Drew   | Mcguire  | 164
  36791   |  Emerson  | Delacruz | 160
  13567   |   Imani   |   Adams  | 156
  9697    |  Dorian   |    Ray   | 156
  41579   | Harrison  | Durham   | 156
  15591   |  Phyllis  |  Clay    | 152
  3008    |  Lucas    | Stanley  | 148
  44956   |  Rachel   |Villarreal| 148
```

Note

This is a complex query. For this tutorial, you don't need to worry about how this query is constructed.

The previous query runs in seconds and returns 2,102 rows.

Suppose that you forget to put in the WHERE clause.

```
SELECT sellerid, firstname, lastname, sum(qtysold)
FROM sales, users
GROUP BY sellerid, firstname, lastname
ORDER BY 4 desc;
```

The result set includes all of the rows in the SALES table multiplied by all the rows in the USERS table (49989*3766). This is called a Cartesian join, and it isn't recommended. The result is over 188 million rows and takes a long time to run.

To cancel a running query, use the CANCEL command with the query's session ID. With the Amazon Redshift query editor v2 you can cancel a query by choosing the cancel button while the query is running.

To find the session ID, start a new session and query the STV_RECENTS table, as shown in the previous step. The following example shows how you can make the results more readable. To do this, use the TRIM function to trim trailing spaces and show only the first 20 characters of the query string.

To determine the session ID of a running query, run the following SELECT statement.

```
SELECT user_id, session_id, start_time, query_text
FROM sys_query_history
WHERE status='running';
```

The result looks something like the following.

```
 user_id |   session_id  |   start_time               |   query_text
---------+---------------+----------------------------
+-----------------------------------------------------------------
 100     |    1073791534 | 2024-03-19 22:26:21.205739 | SELECT user_id, session_id,
 start_time, query_text FROM  ...
```

To cancel the query with session ID 1073791534, run the following command.

```
CANCEL 1073791534;
```

> **Note**
>
> The CANCEL command doesn't stop a transaction. To stop or roll back a transaction, use the ABORT or ROLLBACK command. To cancel a query associated with a transaction, first cancel the query then stop the transaction.

If the query that you canceled is associated with a transaction, use the ABORT or ROLLBACK command to cancel the transaction and discard any changes made to the data:

```
ABORT;
```

Unless you are signed on as a superuser, you can cancel only your own queries. A superuser can cancel all queries.

If your query tool doesn't support running queries concurrently, start another session to cancel the query.

For more information about canceling a query, see CANCEL in the *Amazon Redshift Database Developer Guide*.

Cancel a query using the superuser queue

If your current session has too many queries running concurrently, you might not be able to run the CANCEL command until another query finishes. In that case, run the CANCEL command using a different workload management query queue.

By using workload management, you can run queries in different query queues so that you don't need to wait for another query to complete. The workload manager creates a separate queue, called the Superuser queue, that you can use for troubleshooting. To use the Superuser queue, log on a superuser and set the query group to 'superuser' using the SET command. After running your commands, reset the query group using the RESET command.

To cancel a query using the superuser queue, run these commands.

```
SET query_group TO 'superuser';
CANCEL 1073791534;
RESET query_group;
```

Query data not in your Amazon Redshift database

Following, you can find information about how to get started querying data on remote sources, including Amazon S3 data, remote database managers, remote Amazon Redshift databases, and training machine learning (ML) models using Amazon Redshift.

Topics

Querying your data lake

You can use Amazon Redshift Spectrum to query data in Amazon S3 files without having to load the data into Amazon Redshift tables. Amazon Redshift provides SQL capability designed for fast online analytical processing (OLAP) of very large datasets that are stored in both Amazon Redshift clusters and Amazon S3 data lakes. You can query data in many formats, including Parquet, ORC, RCFile, TextFile, SequenceFile, RegexSerde, OpenCSV, and AVRO. To define the structure of the files in Amazon S3, you create external schemas and tables. Then, you use an external data catalog such as AWS Glue or your own Apache Hive metastore. Changes to either type of data catalog are immediately available to any of your Amazon Redshift clusters.

After your data is registered with an AWS Glue Data Catalog and enabled with AWS Lake Formation, you can query it by using Redshift Spectrum.

Redshift Spectrum resides on dedicated Amazon Redshift servers that are independent of your cluster. Redshift Spectrum pushes many compute-intensive tasks, such as predicate filtering and aggregation, to the Redshift Spectrum layer. Redshift Spectrum also scales intelligently to take advantage of massively parallel processing.

You can partition the external tables on one or more columns to optimize query performance through partition elimination. You can query and join the external tables with Amazon Redshift tables. You can access external tables from multiple Amazon Redshift clusters and query the Amazon S3 data from any cluster in the same AWS Region. When you update Amazon S3 data files, the data is immediately available for queries from any of your Amazon Redshift clusters.

For more information about Redshift Spectrum, including how to work with Redshift Spectrum and data lakes, see Getting started with Amazon Redshift Spectrum in *Amazon Redshift Database Developer Guide*.

Querying data on remote database managers

You can join data from an Amazon RDS database and an Amazon Aurora database with data in your Amazon Redshift database using a federated query. You can use Amazon Redshift to query operational data directly (without moving it), apply transformations, and insert data into your Redshift tables. Some of the computation for federated queries is distributed to the remote data sources.

To run federated queries, Amazon Redshift first makes a connection to the remote data source. Amazon Redshift then retrieves metadata about the tables in the remote data source, issues queries, and then retrieves the result rows. Amazon Redshift then distributes the result rows to Amazon Redshift compute nodes for further processing.

For information about setting up your environment for federated queries, see one of the following topics in the *Amazon Redshift Database Developer Guide*:

- Getting started with using federated queries to PostgreSQL
- Getting started with using federated queries to MySQL

Accessing data in other Amazon Redshift databases

Using Amazon Redshift data sharing, you can share live data with high security and greater ease across Amazon Redshift clusters or AWS accounts for read purposes. You can have instant, granular, and high-performance access to data across Amazon Redshift clusters without manually copying or moving it. Your users can see the most up-to-date and consistent information as it's updated in Amazon Redshift clusters. You can share data at different levels, such as databases, schemas, tables, views (including regular, late-binding, and materialized views), and SQL user-defined functions (UDFs).

Amazon Redshift data sharing is especially useful for these use cases:

- Centralizing business-critical workloads – Use a central extract, transform, and load (ETL) cluster that shares data with multiple business intelligence (BI) or analytic clusters. This approach provides read workload isolation and chargeback for individual workloads.

- Sharing data between environments – Share data among development, test, and production environments. You can improve team agility by sharing data at different levels of granularity.

For more information about data sharing, see Managing data sharing tasks in the *Amazon Redshift Database Developer Guide*.

Training machine learning models with Amazon Redshift data

Using Amazon Redshift machine learning (Amazon Redshift ML), you can train a model by providing the data to Amazon Redshift. Then Amazon Redshift ML creates models that capture patterns in the input data. You can then use these models to generate predictions for new input data without incurring additional costs. By using Amazon Redshift ML, you can train machine learning models using SQL statements and invoke them in SQL queries for prediction. You can continue to improve the accuracy of the predictions by iteratively changing parameters and improving your training data.

Amazon Redshift ML makes it easier for SQL users to create, train, and deploy machine learning models using familiar SQL commands. By using Amazon Redshift ML, you can use your data in Amazon Redshift clusters to train models with Amazon SageMaker AI Autopilot and automatically get the best model. You can then localize the models and make predictions from within an Amazon Redshift database.

For more information about Amazon Redshift ML, see Getting started with Amazon Redshift ML in the *Amazon Redshift Database Developer Guide.*

Learn Amazon Redshift concepts

Amazon Redshift Serverless lets you access and analyze data without all of the configurations of a provisioned data warehouse. Resources are automatically provisioned and data warehouse capacity is intelligently scaled to deliver fast performance for even the most demanding and unpredictable workloads. You don't incur charges when the data warehouse is idle, so you only pay for what you use. You can load data and start querying right away in the Amazon Redshift query editor v2 or in your favorite business intelligence (BI) tool. Enjoy the best price performance and familiar SQL features in an easy-to-use, zero administration environment.

If you are a first-time user of Amazon Redshift, we recommend that you begin by reading the following sections:

- Amazon Redshift Serverless feature overview – In this topic, you can find an overview of Amazon Redshift Serverless and its key capabilities.
- Service highlights and pricing – On this product detail page, you can find details about Amazon Redshift Serverless highlights and pricing.
- Get started with Amazon Redshift Serverless data warehouses. – In this topic, you can learn more about how to create an Amazon Redshift Serverless data warehouse, and start querying data using query editor v2.

If you prefer to manage your Amazon Redshift resources manually, you can create provisioned clusters for your data querying needs. For more information, see Amazon Redshift clusters.

If your organization is eligible and your cluster is being created in an AWS Region where Amazon Redshift Serverless is unavailable, you might be able to create a cluster under the Amazon Redshift free trial program. Choose either **Production** or **Free trial** to answer the question **What are you planning to use this cluster for?** When you choose **Free trial**, you create a configuration with the dc2.large node type. For more information about choosing a free trial, see Amazon Redshift free trial. For a list of AWS Regions where Amazon Redshift Serverless is available, see the Amazon Redshift endpoints listed for the Redshift Serverless API in the *Amazon Web Services General Reference*.

The following are some key Amazon Redshift Serverless concepts.

- **Namespace** – A collection of database objects and users. Namespaces group together all of the resources you use in Amazon Redshift Serverless, such as schemas, tables, users, datashares, and snapshots.
- **Workgroup** – A collection of compute resources. Workgroups house compute resources that Amazon Redshift Serverless use to run computational tasks. Some examples of such resources include Redshift Processing Units (RPUs), security groups, usage limits. Workgroups have network and security settings that you can configure using the Amazon Redshift Serverless console, the AWS Command Line Interface, or the Amazon Redshift Serverless APIs.

For more information about configuring namespace and workgroup resources, see Working with namespaces and Working with workgroups.

Following are some key Amazon Redshift provisioned clusters concepts:

- **Cluster** – The core infrastructure component of an Amazon Redshift data warehouse is a cluster.

  A *cluster* is composed of one or more compute nodes. The *compute nodes* run the compiled code.

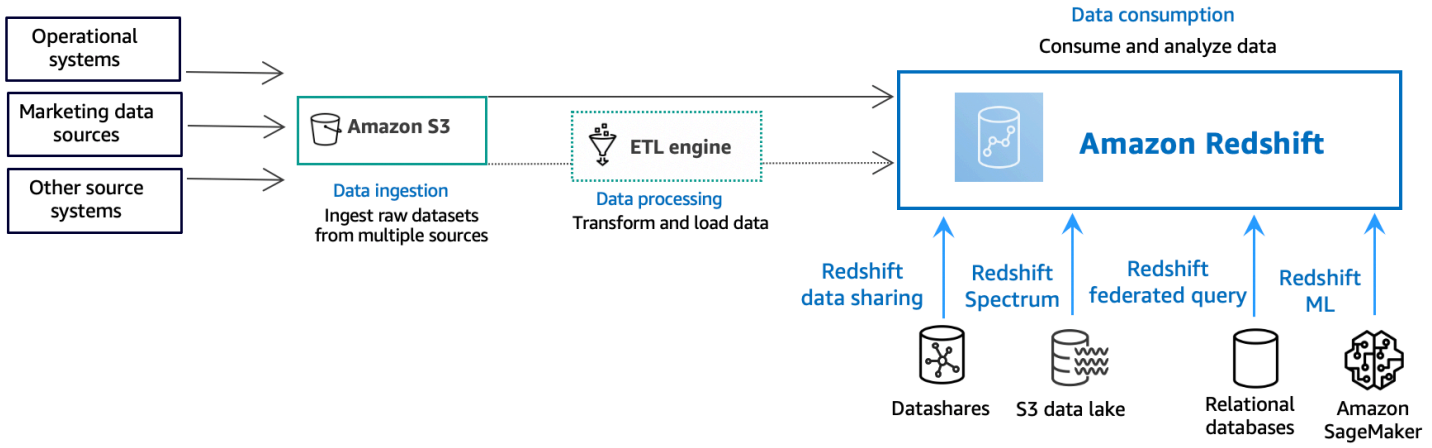
  If a cluster is provisioned with two or more compute nodes, an additional *leader node* coordinates the compute nodes. The leader node handles external communication with applications, such as business intelligence tools and query editors. Your client application interacts directly only with the leader node. The compute nodes are transparent to external applications.
- **Database** – A cluster contains one or more *databases*.

  User data is stored in one or more databases on the compute nodes. Your SQL client communicates with the leader node, which in turn coordinates running queries with the compute nodes. For details about compute nodes and leader nodes, see Data warehouse system architecture. Within a database, user data is organized into one or more schemas.

  Amazon Redshift is a relational database management system (RDBMS) and is compatible with other RDBMS applications. It provides the same functionality as a typical RDBMS, including online transaction processing (OLTP) functions such as inserting and deleting data. Amazon Redshift also is optimized for high-performance batch analysis and reporting of datasets.

Following, you can find a description of typical data processing flow in Amazon Redshift, along with descriptions of different parts of the flow. For further information about Amazon Redshift system architecture, see Data warehouse system architecture.

The following diagram illustrates a typical data processing flow in Amazon Redshift.



An Amazon Redshift *data warehouse* is an enterprise-class relational database query and management system. Amazon Redshift supports client connections with many types of applications, including business intelligence (BI), reporting, data, and analytics tools. When you run analytic queries, you are retrieving, comparing, and evaluating large amounts of data in multiple-stage operations to produce a final result.

At the *data ingestion* layer, different types of data sources continuously upload structured, semi-structured, or unstructured data to the data storage layer. This data storage area serves as a staging area that stores data in different states of consumption readiness. An example of storage might be an Amazon Simple Storage Service (Amazon S3) bucket.

At the optional *data processing* layer, the source data goes through preprocessing, validation, and transformation using extract, transform, load (ETL) or extract, load, transform (ELT) pipelines. These raw datasets are then refined by using ETL operations. An example of an ETL engine is AWS Glue.

At the *data consumption* layer, data is loaded into your Amazon Redshift cluster, where you can run analytical workloads.

For some examples of analytical workloads, see Querying outside data sources.

Additional resources to learn about Amazon Redshift

For more information about Amazon Redshift Serverless, we recommend that you continue to learn more about the concepts introduced in this guide by using the following Amazon Redshift resources:

- Feature videos: These videos help you learn about Amazon Redshift features.
  - To understand Amazon Redshift Serverless at a high-level, watch the following video. Amazon Redshift Serverless Explained in 90 Seconds .
  - To learn how to set up a serverless data warehouse and begin querying data, watch the following video. Getting Started with Amazon Redshift Serverless.'
- Amazon Redshift Management Guide: This guide builds upon this *Amazon Redshift Getting Started Guide*. It provides in-depth information about the concepts and tasks for creating, managing, and monitoring Amazon Redshift Serverless and Amazon Redshift provisioned clusters.
- Amazon Redshift Database Developer Guide: This guide also builds upon this *Amazon Redshift Getting Started Guide*. It provides in-depth information for database developers about designing, building, querying, and maintaining the databases that make up your data warehouse.
  - SQL reference: This topic describes SQL commands and function references for Amazon Redshift.
  - System tables and views reference: This topic describes system tables and views for Amazon Redshift.
- Tutorials for Amazon Redshift: This topic shows tutorials about Amazon Redshift features.
  - Loading data from Amazon S3: This tutorial describes how to load data into your Amazon Redshift database tables from data files in an Amazon S3 bucket.
  - Getting started with data sharing: This section describes how to share and access data in other Amazon Redshift clusters.
  - Using spatial SQL functions with Amazon Redshift: This tutorial demonstrates how to use some of the spatial SQL functions with Amazon Redshift.
  - Querying nested data with Amazon Redshift Spectrum: This tutorial describes how to use Redshift Spectrum to query nested data in Parquet, ORC, JSON, and Ion file formats using external tables.
  - Configuring manual workload management (WLM) queues: This tutorial describes how to configure manual workload management (WLM) in Amazon Redshift.

- Getting started with Amazon Redshift ML: This section describes how users can create, train, and deploy machine learning models using familiar SQL commands.
- What's new: This webpage lists Amazon Redshift new features and product updates.

Document history

> **Note**
>
> For a description of new features in Amazon Redshift, see What's new.

The following table describes the important documentation changes to the *Amazon Redshift Getting Started Guide*.

Change	Description	Release date
Documentation update	Updated the guide to reflect Query Editor v2 managed policy changes and improved serverless namespace and workgroup access permissions.	February 21, 2024
Documentation update	Updated screenshots and procedures to reflect latest console interface improvements and Query Editor v2 enhancements.	March 11, 2023
New feature	Updated the guide to include Amazon Redshift Serverless getting started procedures and workflows . Added comprehensive section on creating and managing serverless data warehouses.	July 12, 2022
Documentation update	Updated the guide to reflect Query Editor v2 as the primary query interface, replacing references to the legacy query editor.	February 2022
Documentation update	Updated the guide to include new sections about getting started with common database tasks, querying your data lake, querying data on remote sources, sharing data, and training machine learning models with Amazon Redshift data.	June 30, 2021
New feature	Updated the guide to describe the new sample load procedure.	June 4, 2021

Change	Description	Release date
Documentation update	Updated the guide to remove the original Amazon Redshift console and improve step flow.	August 14, 2020
New console	Updated the guide to describe the new Amazon Redshift console.	November 11, 2019
New feature	Updated the guide to describe the quick-launch cluster procedure.	August 10, 2018
New feature	Updated the guide to launch clusters from the Amazon Redshift dashboard.	July 28, 2015
New feature	Updated the guide to use new node type names.	June 9, 2015
Documentation update	Updated screenshots and procedure for configuring VPC security groups.	April 30, 2015
Documentation update	Updated screenshots and procedures to match the current console.	November 12, 2014
Documentation update	Moved loading data from Amazon S3 information into its own section and moved next steps section into the final step for better discoverability.	May 13, 2014
Documentation update	Removed the Welcome page and incorporated the content into the main Getting Started page.	March 14, 2014
Documentation update	This is a new release of the *Amazon Redshift Getting Started Guide* that addresses customer feedback and service updates.	March 14, 2014
New guide	This is the first release of the *Amazon Redshift Getting Started Guide*.	February 14, 2013