Working with Amazon Redshift

**SPL-17 Version 4.2.16**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at [*AWS Training and Certification*](https://support.aws.amazon.com/#/contacts/aws-training).

**Overview**

This lab provides an overview of Amazon Redshift. In this lab, you will launch and work with an Amazon Redshift cluster to analyze USA Domestic flight data.

Amazon Redshift is a **fast, fully managed, petabyte-scale data warehouse service** that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. It is optimized for datasets ranging **from a few hundred gigabytes to a petabyte** or more and costs less than $1,000 per terabyte per year, a tenth the cost of most traditional data warehousing solutions.

Amazon Redshift delivers fast query and I/O performance for virtually any size dataset by using **columnar storage technology** and **parallelizing and distributing queries across multiple nodes**. We’ve made Amazon Redshift easy to use by automating most of the common administrative tasks associated with provisioning, configuring, monitoring, backing up, and securing a data warehouse.

TOPICS COVERED

By the end of this lab, you will be able to:

* Launch an Amazon Redshift cluster
* Connect to Amazon Redshift by using SQL client software
* Load data from Amazon S3 into Amazon Redshift
* Query data from Amazon Redshift
* Monitor Amazon Redshift performance

LAB PRE-REQUISITES

To successfully complete this lab, you should be familiar with basic concepts of databases and SQL.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.
* **Additional information:** Where to find more information.
* **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).

**Start lab**

1. To launch the lab, at the top of the page, choose **Start lab**.

**Caution:** You must wait for the provisioned AWS services to be ready before you can continue.

1. To open the lab, choose **Open Console**.

You are automatically signed in to the AWS Management Console in a new web browser tab.

**WARNING:** **Do not change the Region unless instructed.**

COMMON SIGN-IN ERRORS

**Error: You must first sign out**



If you see the message, **You must first log out before logging into a different AWS account:**

* Choose the **click here** link.
* Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
* Choose **Open Console** again.

**Error: Choosing Start Lab has no effect**

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

* Add the lab domain name to your pop-up or script blocker’s allow list or turn it off.
* Refresh the page and try again.

**Task 1: Launch your Amazon Redshift Cluster**

You will now launch an Amazon Redshift cluster, which starts your very own database for use in this lab.

1. Wait until the status of **Provisioning lab resources** gets complete.

This indicates that required resources are available for your lab.

1. From the **AWS Management Console**, use the **AWS search bar** to search for

Amazon Redshift

 and then choose the service from the list of results in a new browser tab.

1. Choose **Create cluster** then configure:

**Note:** If you do not find a **Create cluster** button, expand the menu on the left side of the page, and choose **Clusters**.

* **Cluster identifier:**

lab

* **Node type:** **dc2.large**
* **Number of nodes:**

2

This lab uses the **dc2.large** node size, which has 160GB of storage per node. You will be using a single node for this lab, but the type and number of nodes in a Redshift cluster can be changed at any time to provide extra storage and faster data processing.

1. Scroll down to the **Database configurations** section, then configure:

* **Admin user name:**

master

* **Admin user password:**

Redshift123

1. In the **Cluster permissions** section, configure:

* Choose **Associate IAM role**
* Select  **Redshift-Role**
* Choose **Associate IAM roles**

1. Next to **Additional configurations**, un-select the slider so that it does NOT use the defaults.
2. Expand  **Network and security**, then configure:

* **Virtual private cloud (VPC)** **Lab VPC**
* **VPC security groups:**
  + Select  **Redshift Security Group**
  + Un-select  **default**

1. Expand  **Database configurations**, then configure:

* **Database name:**

lab

These settings define the network and security configurations for the Redshift cluster.

1. Choose **Create cluster**

Your cluster can take up to 5 minutes to launch.

While you are waiting, please continue reading the next section.

**Amazon Redshift Primer**

While you are waiting for your cluster to launch, here is some information that highlights the most important features of Amazon Redshift.

NODES & CLUSTERS

An Amazon Redshift data warehouse is a collection of computing resources called **nodes**. This collection of nodes is called a **cluster**. When you provision a cluster, you specify the type and the number of nodes that will make up the cluster. The node type determines the storage size, memory, CPU, and price of each node in the cluster:

| **Family** | **Node Type** | **Storage type** | **Storage per node** | **Maximum Storage** |
| --- | --- | --- | --- | --- |
| Dense Compute | dc2.large | SSD | 0.16TB NVMe SSD | 32 nodes = 5.12TB |
| Dense Compute | dc2.8xlarge | SSD | 2.56TB NVMe SSD | 128 nodes = 326TB |
| Dense Storage | ds2.xlarge | Magnetic | 2TB | 32 nodes = 64TB |
| Dense Storage | ds2.8xlarge | Magnetic | 16TB | 128 nodes = 2PB |

SCALABILITY

If your storage and performance needs change after you initially provision your cluster, you can always **scale** the cluster in or out by **adding or removing nodes**, scale the cluster up or down by **specifying a different node type**, or you can do both. Resizing the cluster in either way involves minimal downtime. Resizing replaces the old cluster at the end of the resize operation. When you submit a resize request, the source cluster remains in read-only mode until the resize operation is complete.

PARALLEL PROCESSING

Amazon Redshift **distributes workload** to each node in a cluster and **processes work in parallel**, allowing processing speed to scale in addition to storage.

COLUMNAR STORAGE

Columnar storage for database tables is an important factor in optimizing analytic query performance because it drastically **reduces the overall disk I/O requirements** and **reduces the amount of data you need to load from disk**.

Rather than storing data values together for a whole row, Amazon Redshift stores data by **column**. This means that operations on a column require less disk I/O.

COMPRESSION

Compression is a column-level operation that **reduces the size of data when it is stored**. Compression conserves storage space and reduces the size of data that is read from storage, which reduces the amount of disk I/O and therefore **improves query performance.**

SNAPSHOTS AS BACKUPS

Snapshots are **point-in-time backups** of a cluster. You can create snapshots automatically or manually. Amazon Redshift stores these snapshots internally in **Amazon S3** using an encrypted Secure Sockets Layer (SSL) connection. If you need to restore a cluster, Amazon Redshift creates a new cluster and imports data from the snapshot that you specify.

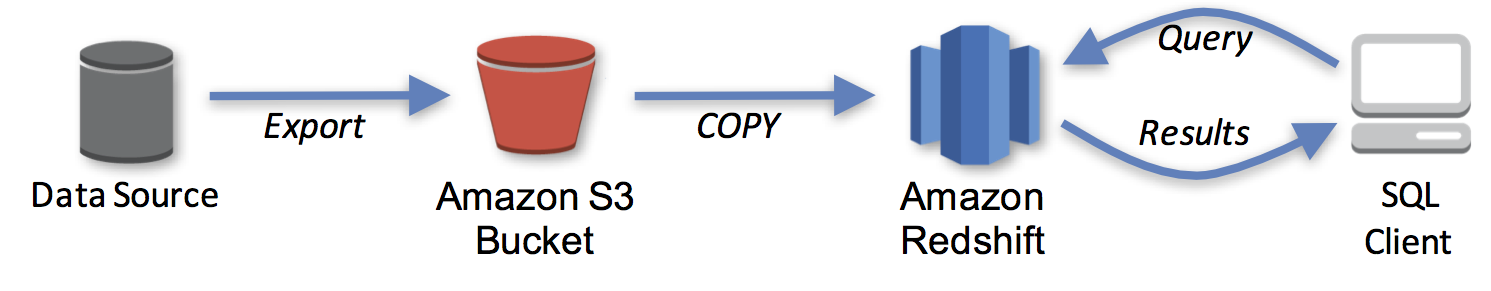
INTEGRATES WITH EXISTING BUSINESS INTELLIGENCE TOOLS

Amazon Redshift uses **industry-standard SQL** and is accessed using standard JDBC and ODBC drivers. Your existing Business Intelligence tools can easily integrate with Amazon Redshift.

DATA LOADING PROCESS

The typical process for loading data into Amazon Redshift is:

* Data is **exported** from a source system (for example, a company database).
* The data is placed into an **Amazon S3 bucket**, preferably in a compressed format to save storage space.
* The data is **copied into Amazon Redshift tables** via the COPY command.
* The **SQL client** is used to **query** Amazon Redshift.
* The **results** of the query will be returned to the SQL client.



In this lab, you will be loading **USA domestic airline data** for analysis. The data has been obtained from the United States Department of Transportation’s Bureau of Transportation Statistics.

The transport data has already been placed into an Amazon S3 bucket in a compressed format. This lab will lead you through the steps of **loading the data** into an Amazon Redshift cluster and then **running queries to analyze the data**.

**Task 2: Connect to Amazon Redshift**

In this task, you will use a web-based PostgreSQL client (“pgweb”) to connect to Redshift.

1. Verify that the **Status** of the **lab** database listed in the console is  Available . If it isn’t, wait for it to become so.
2. Choose the **lab** cluster you just created.
3. Copy the **Endpoint** (on the right side of the screen) to a text editor.

The endpoint will look similar to: *lab.czvdbh5dsk9y.us-west-2.redshift.amazonaws.com:5439/lab*

1. Remove the

:5439/lab

 ending from the Endpoint in your text editor.

1. Copy the **pgweb** IP address shown to the left of these instructions.

This is the IP address of a web server that is running the pgweb software.

1. Open a new tab in your web browser, paste the IP address and hit Enter.

You will be presented with the *pgweb* login screen.

1. Configure the following settings:

* **Host** *paste in the edited Endpoint from your text editor*
* **Username:**

master

* **Password:**

Redshift123

* **Database:**

lab

* **Port:**

5439

 (which is different to the default value)

**Note:** If you cannot change the **Port** value, set **SSL** to **disable** and try again.

1. Choose **Connect** .

You are now ready to interact with Amazon Redshift.

**Note:** If you are not able to connect, double-check your configuration.

**Task 3: Load data**

In this task, you will create a Table in Amazon Redshift. Tables are used to store a particular set of information.

The first table will be used to store flight information.

1. Copy and paste the following text into pgweb (above the *Run Query* button):

CREATE TABLE flights (

year smallint,

month smallint,

day smallint,

carrier varchar(80) DISTKEY,

origin char(3),

dest char(3),

aircraft\_code char(3),

miles int,

departures int,

minutes int,

seats int,

passengers int,

freight\_pounds int

);

1. Choose **Run Query**.

**Expected output:** A new **flights** table will appear on the left of the screen, under the *Tables* heading.

You can now **load data** into the table. The data has already been placed into an Amazon S3 bucket and can be loaded into Amazon Redshift by using the COPY command.

1. Delete the previous query text in the pgweb pane.
2. Paste the following text into pgweb **but do not run it yet**:

COPY flights

FROM 's3://us-west-2-aws-training/courses/spl-17/v4.2.16.prod-d92ec317/data/flights-usa'

IAM\_ROLE 'INSERT-YOUR-REDSHIFT-ROLE'

GZIP

DELIMITER ','

REMOVEQUOTES

REGION 'us-west-2';

1. Replace the text **INSERT-YOUR-REDSHIFT-ROLE** in the third line with the value of **RedshiftRole** shown to the left of these instructions. The result should look similar to this:

*IAM\_ROLE ‘arn:aws:iam::123456789012:role/Redshift-Role’*

1. Choose **Run Query**.

**Expected output:** It will take **approximately 3 minutes** to load the data. It will eventually return *No records found*. Please continue reading while the data is loading.

The **COPY** command is used to load data into Amazon Redshift:

* **FROM:** Indicates where the data is located
* **IAM\_ROLE:** Provides the permissions to access the data being loaded
* **GZIP:** Indicates that the data has been compressed (zipped) – Amazon Redshift will automatically decompress the data when it is loaded
* **DELIMITER:** Indicates that data items are separated by a comma
* **REMOVEQUOTES:** Tells Amazon Redshift to remove quotation marks that are included in the data
* **REGION:** Indicates which AWS region contains the S3 bucket

The data being loaded consists of:

* 23 data files in CSV format (one for each year from 1990 - 2012)
* Comprising **6 GB of data**
* Compressed with GZIP down to only 700 MB of storage

The data files are being loaded **in parallel** from Amazon S3. This is the most efficient way to load data into Amazon Redshift since the load process is distributed across multiple *slices* across all available nodes.

Each slice of a compute node is allocated a portion of the node’s memory and disk space, where it processes a portion of the workload assigned to the node. The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation.

When you create a table, you can optionally specify one column as the **distribution key**. When the table is loaded with data, the rows are distributed to the node slices according to the distribution key. Choosing a good distribution key enables Amazon Redshift to use parallel processing to load data and execute queries efficiently.

The CREATE TABLE command you ran earlier designated the carrier (airline) field as the **Distribution Key** (DISTKEY). This means the data will be split between the all available slices and nodes, but all data related to a particular carrier will always reside on the same slice. This improves processing speed when performing operations on the carrier field, such as GROUP BY and JOIN operations.

**Task 4: Run Queries**

Once the COPY command has finished running, you can examine the contents of the table.

1. Run this query (remove the existing query and paste this in to the pgweb query pane, then choose *Run Query*) to count the number of rows in the table:

SELECT COUNT(\*) FROM flights;

**Expected output:** You should see a value of **96,825,753** records. This means that almost 100 million rows have been loaded into the table.

You can also use standard SQL commands to query the data.

1. Run this query to view 10 random rows of data:

SELECT \*

FROM flights

ORDER BY random()

LIMIT 10;

This query actually assigns a random number to all 96 million rows, sorts them by the random number and then returns the first 10 results.

Here is an explanation of the columns returned, along with some sample data:

| **Column** | **Sample** | **Description** |
| --- | --- | --- |
| year | 1997 | Year |
| month | 3 | Month |
| day | 25 | Day |
| carrier | United Air Lines Inc. | Airline |
| origin | SEA | Airport where the flight started |
| dest | SFO | Airport where the flight landed |
| aircraft\_code | 616 | A code indicating the aircraft (You will decode this in a later step) |
| miles | 678 | Distance between cities |
| departures | 11 | The number of departures on that day |
| minutes | 1109 | Total minutes of flight time for all flights on that day |
| seats | 1188 | Total number of available seats for all flights on that day |
| passengers | 848 | Total number of passengers who flew for all flights on that day |
| freight\_pounds | 3335 | Total pounds of freight transported for all flights on that day |

This data is highly detailed and very *low level*, which is similar to that found in a typical corporate data warehouse.

Now that you have the data loaded, the next step is to perform **queries** to find underlying patterns in the data and to help drive business decisions.

1. Run this query that finds the top 10 carriers by number of departures:

SELECT

carrier,

SUM (departures)

FROM flights

GROUP BY carrier

ORDER BY 2 DESC

LIMIT 10;

**Note:** Who are the top 3 carriers by number of departures?

1. Change **departures** to

passengers

 and run it again to view top carriers by **passengers carried**.

**Note:** Who are the top 3 carriers by passengers carried?

1. Change **departures** to

miles

 and run it again to view top carriers by **miles flown**.

**Note:** Who are the top 3 carriers by miles flown?

1. Change **departures** to

passengers \* miles

 and run it again to view top carriers by **passenger-miles**.

**Note:** Who are the top 3 carriers by passenger-miles?

1. Change **departures** to

freight\_pounds

 and run it again to view top carriers by **freight transported**.

**Note:** Who are the top 3 carriers of freight? (You should be able to guess this one!)

Each of these queries is performing calculations against almost 100 million rows of data, but they each take only a few seconds to run. Adding additional compute nodes will make the queries run even faster.

**Task 5: Joining tables**

In this task, you will load more data and then run queries that join information between tables.

1. Run this query to create a new table for aircraft information:

CREATE TABLE aircraft (

aircraft\_code CHAR(3) SORTKEY,

aircraft VARCHAR(100)

);

**Expected output:** A new **aircraft** table will appear on the left of the screen, under the *Tables* heading.

1. Paste the following text into pgweb **but do not run it yet**:

COPY aircraft

FROM 's3://us-west-2-aws-training/courses/spl-17/v4.2.16.prod-d92ec317/data/lookup\_aircraft.csv'

IAM\_ROLE 'INSERT-YOUR-REDSHIFT-ROLE'

IGNOREHEADER 1

DELIMITER ','

REMOVEQUOTES

TRUNCATECOLUMNS

REGION 'us-west-2';

1. Replace the text **INSERT-YOUR-REDSHIFT-ROLE** in the third line with the value of **RedshiftRole** shown to the left of these instructions.
2. Run the query.

**Expected output:** Again pgweb will return *No records found*, but the underlying operation will load 383 different types of aircraft flown by the carriers.

1. Run this query to view 10 random rows of aircraft data:

SELECT \*

FROM aircraft

ORDER BY random()

LIMIT 10;

The table contains an **aircraft code** and an **aircraft description**. The two tables can be joined together to provide useful information:

1. Run this query to view the most-flown types of aircraft:

SELECT

aircraft,

SUM(departures) AS trips

FROM flights

JOIN aircraft using (aircraft\_code)

GROUP BY aircraft

ORDER BY trips DESC

LIMIT 10;

**Expected output:** The results show the names of aircraft with the most trips in the database. The **JOIN** command links the flight table with the aircraft table.

**Task 6: Analyze Performance**

You can use the EXPLAIN command to view how Amazon Redshift processes queries.

1. Run this query like the others, using the **Run Query** button:

SET enable\_result\_cache\_for\_session TO OFF;

EXPLAIN

SELECT

aircraft,

SUM(departures) AS trips

FROM flights

JOIN aircraft using (aircraft\_code)

GROUP BY aircraft

ORDER BY trips DESC

LIMIT 10;

It is the same as the previous query, but is prefixed by the EXPLAIN command.

**Expected output:** This command will return an **Explain Plan** similar to this:

XN Limit (cost=1000156830987.88..1000156830987.90 rows=10 width=29)

-> XN Merge (cost=1000156830987.88..1000156830988.84 rows=383 width=29)

Merge Key: sum(flights.departures)

-> XN Network (cost=1000156830987.88..1000156830988.84 rows=383 width=29)

Send to leader

-> XN Sort (cost=1000156830987.88..1000156830988.84 rows=383 width=29)

Sort Key: sum(flights.departures)

-> XN HashAggregate (cost=156830970.49..156830971.44 rows=383 width=29)

-> XN Hash Join DS\_BCAST\_INNER (cost=4.79..156346841.73 rows=96825752 width=29)

Hash Cond: ("outer".aircraft\_code = "inner".aircraft\_code)

-> XN Seq Scan on flights (cost=0.00..968257.52 rows=96825752 width=11)

-> XN Hash (cost=3.83..3.83 rows=383 width=32)

-> XN Seq Scan on aircraft (cost=0.00..3.83 rows=383 width=32)

The plan shows the **logical steps** that Amazon Redshift will perform when running the query. Reading the Explain Plan from the bottom up, it displays a breakdown of logical operations needed to perform the query as well as an indication of their **relative processing cost** and the amount of data that needs to be processed. By analyzing the plan, you can often identify opportunities to improve query performance.

In traditional databases, a **sequential scan** (Seq Scan) across many rows of data can be very inefficient and is normally improved by adding an index. However, Amazon Redshift **does not use indexes**, yet is able to perform extremely fast queries across huge quantities of data – in this case, scanning over 96 million rows in a few seconds.

DATA COMPRESSION & COLUMN-BASED STORAGE

Data in Amazon Redshift is stored as **columns**. This is faster than storing data as **rows**, since most queries only require a few columns of data. It also allows Amazon Redshift to compress data within each column.

When data was loaded with the COPY command earlier in this lab, Amazon Redshift performed a compression analysis to identify the optimal way to store each column. You can view the results of the analysis by using the ANALYZE COMPRESSION command.

1. Run this command to analyze the data stored in the *flights* table:

ANALYZE COMPRESSION flights;

**Expected output:** Amazon Redshift will display recommended compression settings for the data.

Compression is a column-level operation that reduces the size of data when it is stored. Possible compression methods are:

* **Byte dictionary:** A method of reference up to 256 possible values in a single byte. Ideal for fields with few, but frequently repeated, values such as Country names.
* **Delta:** Compresses data by recording the difference between values that follow each other in the column.
* **LZO:** Provides a very high compression ratio with good performance. Works well for columns that store very long character strings, especially free form text, such as product descriptions, user comments, or JSON strings.
* **Mostly:** Compresses the majority of the values in the column to a smaller standard storage size.
* **Run-length:** Replaces a value that is repeated consecutively with a token that consists of the value and a count of the number of consecutive occurrences (the length of the run). Best suited to a table in which data values are often repeated consecutively, for example, when the table is sorted by those values.
* **Text:** Compresses VARCHAR columns in which the same words recur often.
* **Zstandard: (zstd)** Provides a high compression ratio with very good performance across diverse data sets. Works especially well with CHAR and VARCHAR columns that store a wide range of long and short strings, such as product descriptions, user comments, logs, and JSON strings.
* **AZ64** AZ64 is a proprietary compression encoding algorithm designed by Amazon to achieve a high compression ratio and improved query processing.
* **Raw:** Uncompressed

When data is compressed, information can be retrieved from disk faster. Compression conserves storage space, **reduces the amount of disk I/O** and therefore **improves query performance**.

CREATING TABLES FROM OTHER TABLES

It is often necessary to manipulate data to make the information more meaningful. Amazon Redshift has the ability to **create new tables based upon data from existing tables**.

For example, if you want to closely analyze data on **passengers who fly to Las Vegas**, you can create a table with only those flights that flew to Las Vegas.

You will now load a table that converts 3-digit airport codes (eg ‘LAS’) into easily-readable city names (‘Las Vegas’).

1. **Run** this query create a new table for airport information:

CREATE TABLE airports (

airport\_code CHAR(3) SORTKEY,

airport varchar(100)

);

**Expected output:** A new **airports** table will appear on the left of the screen, under the *Tables* heading.

1. Run this command, again replacing **INSERT-YOUR-REDSHIFT-ROLE** with the **RedshiftRole** value shown to the left of these instructions:

COPY airports

FROM 's3://us-west-2-aws-training/courses/spl-17/v4.2.16.prod-d92ec317/data/lookup\_airports.csv'

IAM\_ROLE 'INSERT-YOUR-REDSHIFT-ROLE'

IGNOREHEADER 1

DELIMITER ','

REMOVEQUOTES

TRUNCATECOLUMNS

REGION 'us-west-2';

**Expected output:** Again pgweb will return *No records found*, but the underlying operation will load a list of 6,265 airports.

Next, combine the *flights* and *airports* information into a new table that only including flights that went to Las Vegas.

1. Run this query create a new table about Las Vegas flights:

CREATE TABLE vegas\_flights

DISTKEY (origin)

SORTKEY (origin)

AS

SELECT

flights.\*,

airport

FROM flights

JOIN airports ON origin = airport\_code

WHERE dest = 'LAS';

**Expected output:** A new **vegas\_flights** table will appear on the left of the screen, under the *Tables* heading.

Queries can now be run against this new *vegas\_flights* table.

1. Run this query to discover from where the most popular flights to Las Vegas originate:

SELECT

airport,

to\_char(SUM(passengers), '999,999,999') as passengers

FROM vegas\_flights

GROUP BY airport

ORDER BY SUM(passengers) desc

LIMIT 10;

**Note:** Which airport sends the most passengers to Las Vegas?

This query also demonstrates use of the PostgreSQL

to\_char

 function that formats output text in a human-friendly format.

Creating new tables in this manner can improve performance since queries only need to scan a subset of data.

EXAMINING DISK SPACE AND DATA DISTRIBUTION

Data in Amazon Redshift is distributed across **multiple nodes and hard disks**.

1. Run this query to see much disk capacity has been used:

SELECT

owner AS node,

diskno,

used,

capacity,

used/capacity::numeric \* 100 as percent\_used

FROM stv\_partitions

WHERE host = node

ORDER BY 1, 2;

**Expected output:** The output shows:

* **Node:** The node within the cluster.
* **Diskno:** The disk number. Nodes can have multiple drives, allowing data to be accessed in parallel.
* **Used:** Megabytes of disk space used.
* **Capacity:** Available disk space. There is 160GB per node, but extra is provided for database replication.
* **Percent\_used:** Percent of disk space used. The flight data is occupying less than 0.5% of available disk space.

Disk usage is also available on a per-table basis.

1. Run this query to see how much space is taken by each of the data tables:

SELECT

name,

count(\*)

FROM stv\_blocklist

JOIN (SELECT DISTINCT name, id as tbl from stv\_tbl\_perm) USING (tbl)

GROUP BY name;

**Expected output:** The amounts shown are in MB. The flights table consumes around 1600 MB. Each Amazon Redshift dc2.large node can hold 160GB, which is **one hundred times** the amount of data currently in use.

**Task 7: Explore the Amazon Redshift Console**

All your interactions with Amazon Redshift so far have been via SQL.

Amazon Redshift also has a **management console** that provides insight into operation of the system.

1. Return to your web browser tab showing the Amazon Redshift console.
2. From the collapsible left navigation pane (if it is collapsed, choose the  icon to expand it), choose **Clusters**.
3. Choose the **lab** cluster.
4. Choose the **Query monitoring** tab.

Amazon Redshift maintains information about every data load and query performed.

1. Choose the refresh  button aligned to **Query monitoring**.
2. From the **Queries and loads** section, choose one of the query job numbers for **COPY flights FROM…** in the SQL column.

Information will be displayed showing:

* The SQL used to run the query.
* The Total runtime.
* The Rows scanned.
* The Total data scanned.

1. Choose the **Query plan** tab.

This shows a chart that displays system performance during the load. This information can be used to diagnose problems and to monitor system performance during regular load processes.

1. In the left navigation pane, choose **Clusters**.
2. Choose the **lab** cluster.
3. Choose the **Cluster performance** tab to view information about the cluster.
4. Expand **Alarms**. Here you will see a section for **CloudWatch Alarms** (although this lab hasn’t created any), which can send a notification based upon cluster metrics, such as available disk space or the health of the cluster.

SNAPSHOTS

Snapshots are **point-in-time backups** of a cluster. You can create snapshots automatically or manually. Amazon Redshift stores these snapshots in Amazon S3. If you need to restore a cluster, Amazon Redshift creates a new cluster and imports data from the snapshot that you specify.

Amazon Redshift periodically takes **automated snapshots** and deletes the automated snapshot at the end of a retention period that you specify.

You can also take a **manual snapshot** whenever you wish. Manual snapshots are retained even after you delete your cluster. Manual snapshots accrue storage charges, so it is important that you manually delete them if you no longer need them.

To reduce backup times and Amazon S3 storage requirements, Amazon Redshift uses **incremental backups**. When a snapshot is taken, the backup records the cluster changes since the last snapshot.

Amazon Redshift provides **free storage for snapshots** that is equal to the storage capacity of your cluster until you delete the cluster. You can use this free storage for automated or manual snapshots. After the free backup storage limit is reached, you are charged for any additional storage at the normal rate.

1. Choose the **Maintenance** tab.

In the **Snapshots** section, you should see that a an automatic snapshot was created.

EXPORTING DATA

Data can also be **exported to Amazon S3** with the UNLOAD command. The data can then be used in other systems, such as Amazon DynamoDB, your own applications or loaded into another Amazon Redshift cluster. The command accepts an SQL query and can export the data into fixed-width or delimited text files and can also be compressed with GZIP and encrypted.

DELETE THE CLUSTER

This lab is now complete. The final step is to shut down the cluster.

1. In the **Actions** menu, choose **Delete**.
2. De-select  **Create final snapshot**.
3. Enter

delete

 into the *To confirm deletion,…* field.

1. Choose **Delete cluster**

Your cluster will now be deleted.

**Conclusion**

 Congratulations! You have now successfully learned how to:

* Launch an Amazon Redshift cluster.
* Connect to Amazon Redshift by using SQL client software.
* Load data from Amazon S3 into Amazon Redshift.
* Query data from Amazon Redshift.
* Monitor Amazon Redshift performance.

**End lab**

Follow these steps to close the console and end your lab.

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](https://aws.amazon.com/training/).

*Your feedback is welcome and appreciated.*  
If you would like to share any feedback, suggestions, or corrections, please provide the details in our [*AWS Training and Certification Contact Form*](https://support.aws.amazon.com/#/contacts/aws-training).