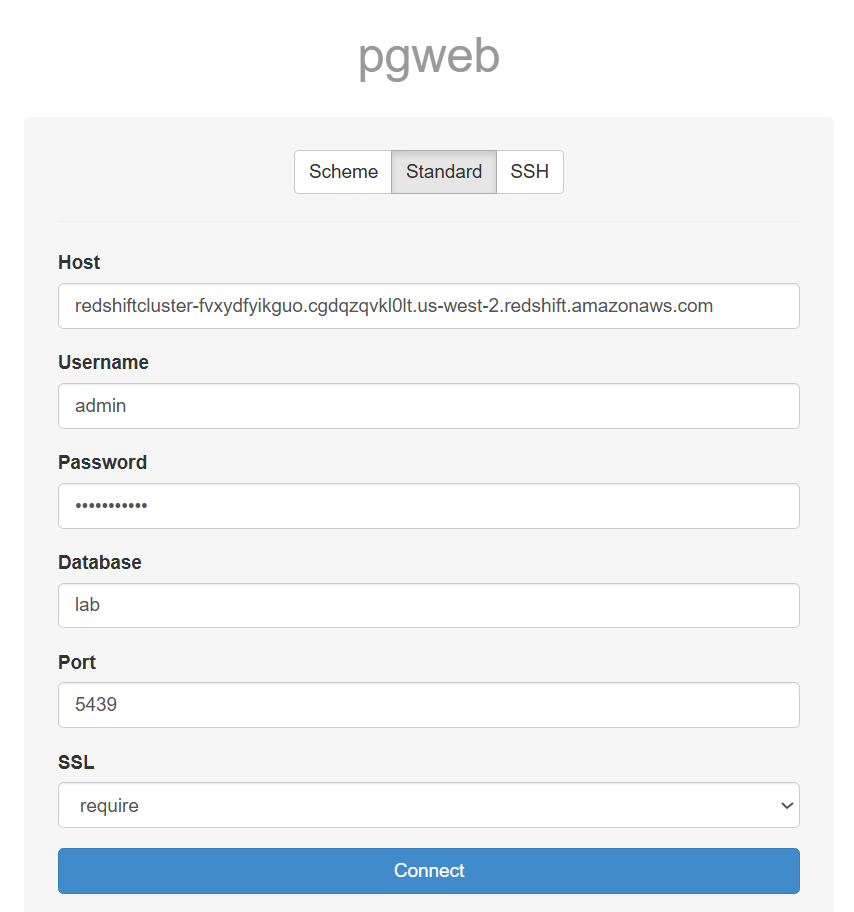
# Lab 1 - Introduction to Amazon Redshift

Objs -

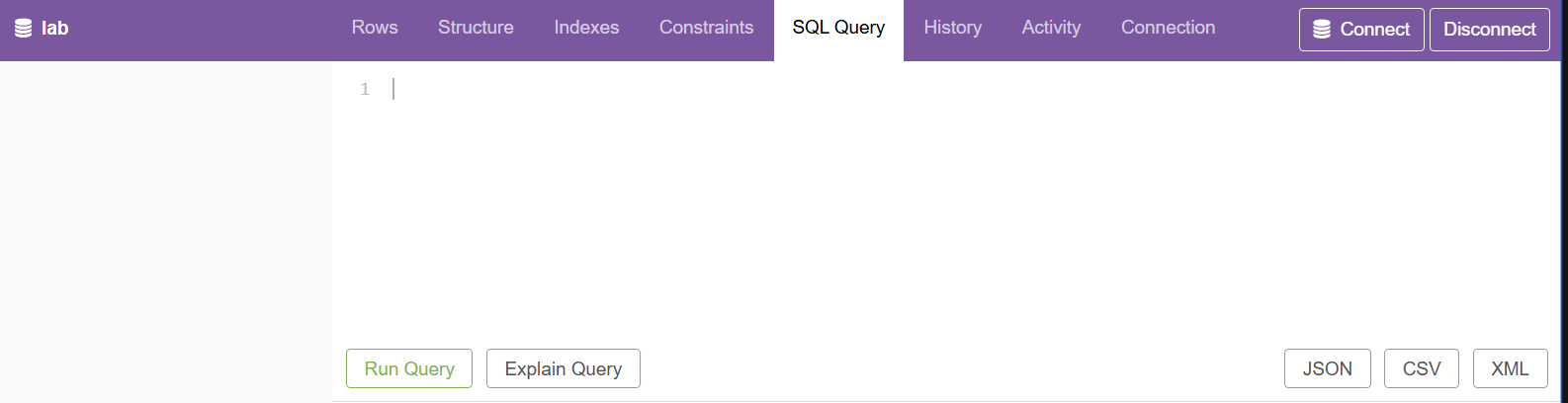
1. Connect to an Amazon Redshift cluster using SQL client software
2. Create a table in Amazon Redshift
3. Load data from Amazon Simple Storage Service (Amazon S3) into Amazon Redshift
4. Query data from Amazon Redshift
5. Analyze Amazon Redshift performance
6. Navigate the Amazon Redshift console

**Task 1: Connecting to Amazon Redshift**

1.1 Cluster has already been created, we have to connect to it

1.2 using the URL given, open the web-based PostgreSQL client (pgweb), and fill in these:  


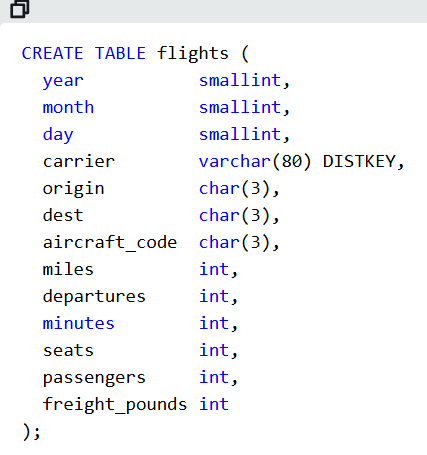
1.3 connect





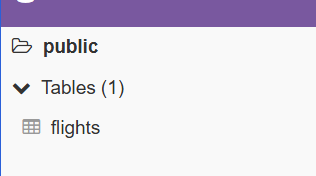
**Task 2: Loading data**

2.1 create a table



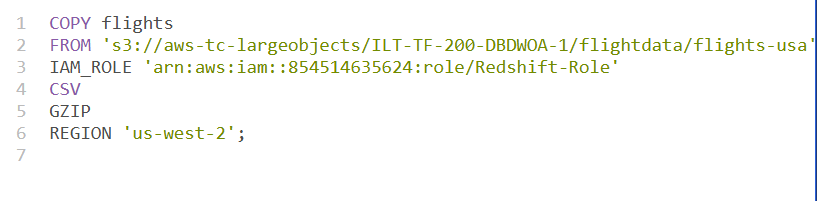
2.2 run the query

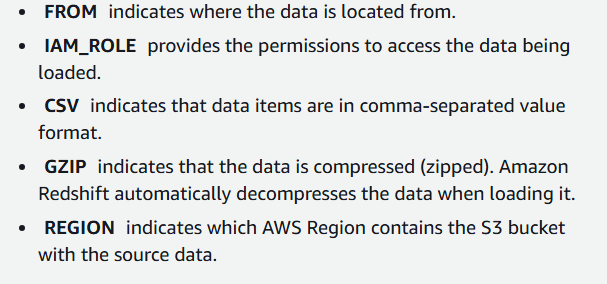
2.3 new table formed



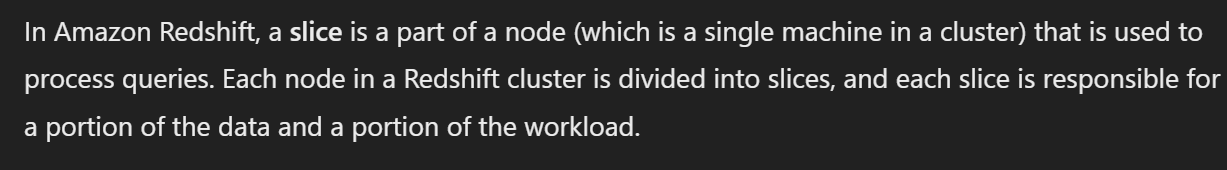
2.4 data is already in S3 bucket, load it in table created

Fill the s3 and iam values



2.5 

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

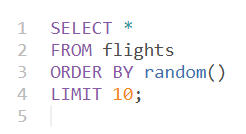


**Task 3: Running queries**

3.1



3.2



3.3 

**Task 4: Joining tables**

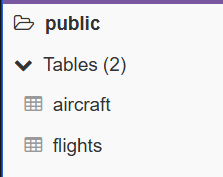
4.1 create a new table

CREATE TABLE aircraft (

aircraft\_code CHAR(3) SORTKEY,

aircraft VARCHAR(100)

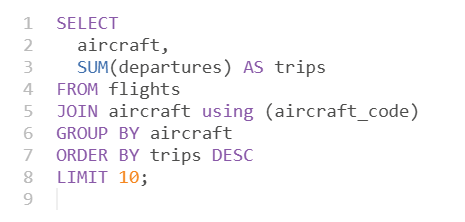
);

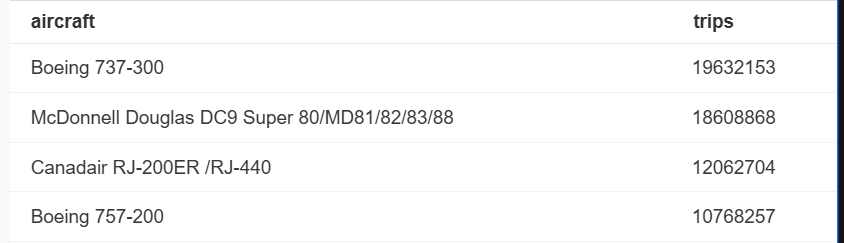


4.2 copy s3



4.3 join





**Task 5: Analyzing performance**

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

*EXPLAIN*

*SELECT*

*aircraft,*

*SUM(departures) AS trips*

*FROM flights*

*JOIN aircraft using (aircraft\_code)*

*GROUP BY aircraft*

*ORDER BY trips DESC*

*LIMIT 10;*

Same as join query, just EXPLAIN at beginning

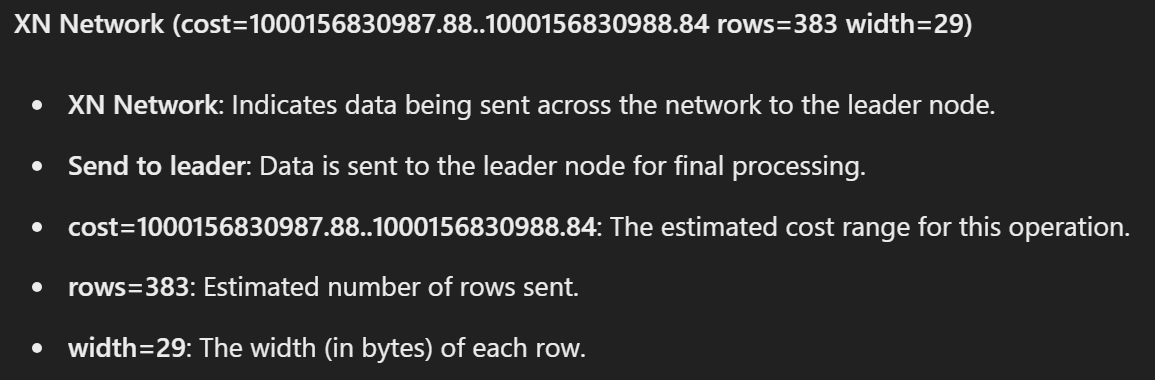


This is Execution plan

Execution plans help understand the performance and resource utilization of SQL queries.

1. XN Limit: Represents the top-level operation, which limits the output to a certain number of rows.
2. XN Merge: Combines sorted data from different sources.
3. XN Network: Sends data between different nodes.
4. XN Sort: Sorts the data.
5. XN HashAggregate: Aggregates data using a hash-based approach.
6. XN Hash Join DS\_BCAST\_INNER: Performs a hash join, with a broadcast of the inner table.
7. XN Seq Scan: Sequentially scans a table.

For example-



Cost is unitless measure, and a combination of various factors (cpu usage, network, memory usages, etc.)

Execution Plan Analysis:

1. The execution plan in Amazon Redshift outlines the logical steps the system takes to run a query, displayed from the bottom up.
2. It breaks down operations, shows relative processing costs, and indicates the amount of data processed.
3. By analyzing these plans, you can identify inefficient operations and opportunities for optimization.

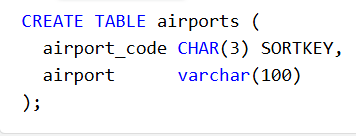
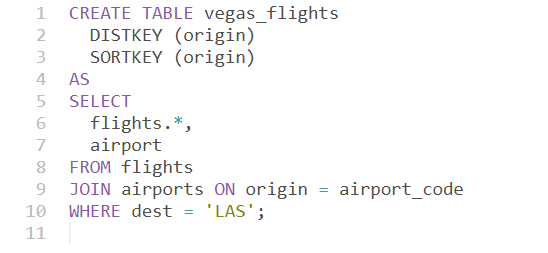
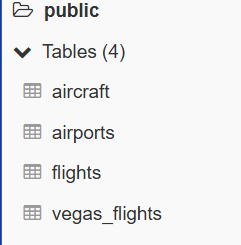
Sequential Scans and Indexes:

1. In traditional databases, sequential scans (Seq Scans) can be inefficient for large datasets, often improved by adding indexes.
2. Amazon Redshift does not use traditional indexes but relies on other mechanisms for performance optimization.
3. Despite not using indexes, Amazon Redshift can execute extremely fast queries on massive datasets.
4. Techniques such as parallel processing, data distribution across nodes, and efficient use of resources enable Redshift to scan and process millions of rows in seconds.

5.2

1. Amazon Redshift stores data in a columnar format, meaning each column's data is stored together rather than storing rows together.
2. Columnar storage enables more effective data compression. Since similar data types are stored together

5.3 CREATE TABLES FROM OTHER TABLES

1. Create a new table
   1. 
   2. 
   3. 
   4. 
   5. Selects all columns from the flights table and the airport column from the airports table.
   6. Performs an inner join between the flights table and the airports table on the origin and airport\_code columns.
   7. Filters the results to include only flights where the destination (dest) is 'LAS' (Las Vegas).

5.4 view disk space usage

*SELECT*

*owner AS node,*

*diskno,*

*used,*

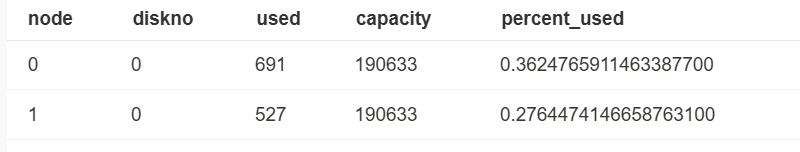
*capacity,*

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

*FROM stv\_partitions*

*WHERE host = node*

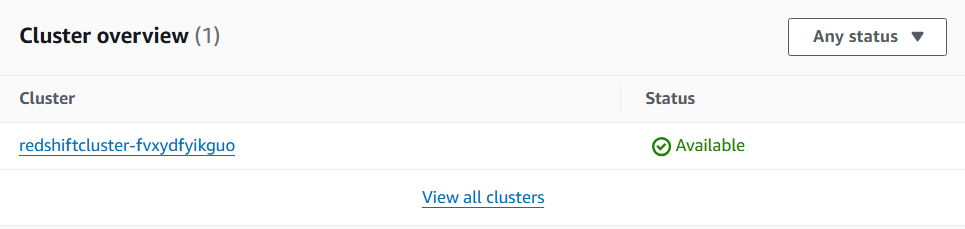
*ORDER BY 1, 2;*

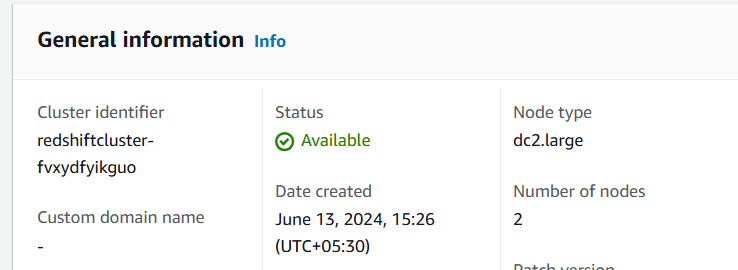


Notice we have two nodes

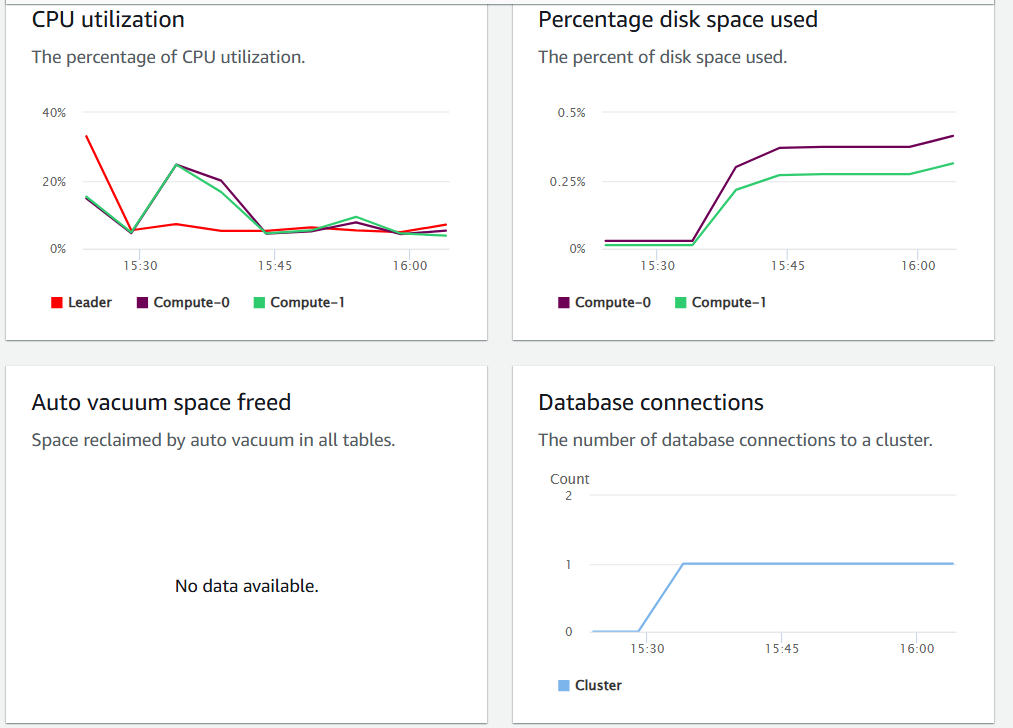
**Task 6: Exploring the Amazon Redshift console**

6.1 open redshift in console choose the cluster

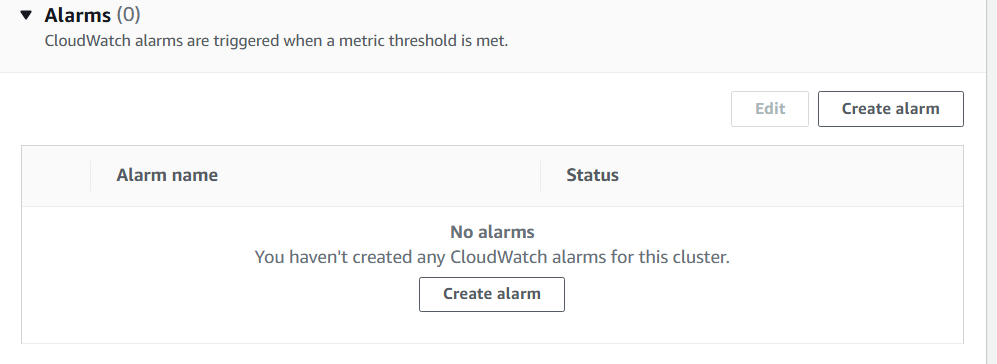


6.2 

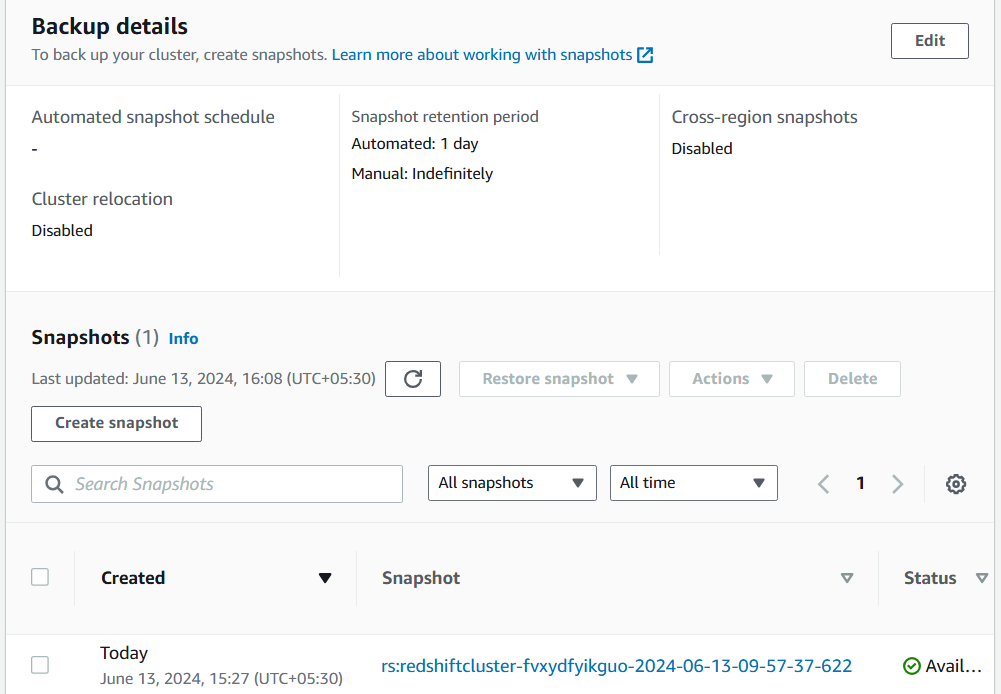
6.3



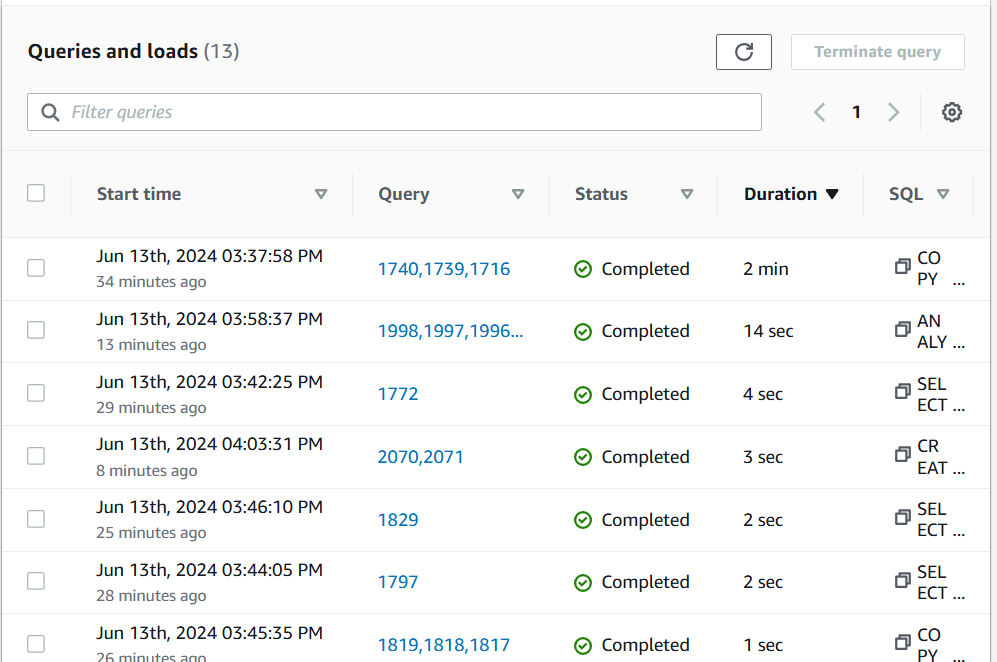
6.4 we can set alarms



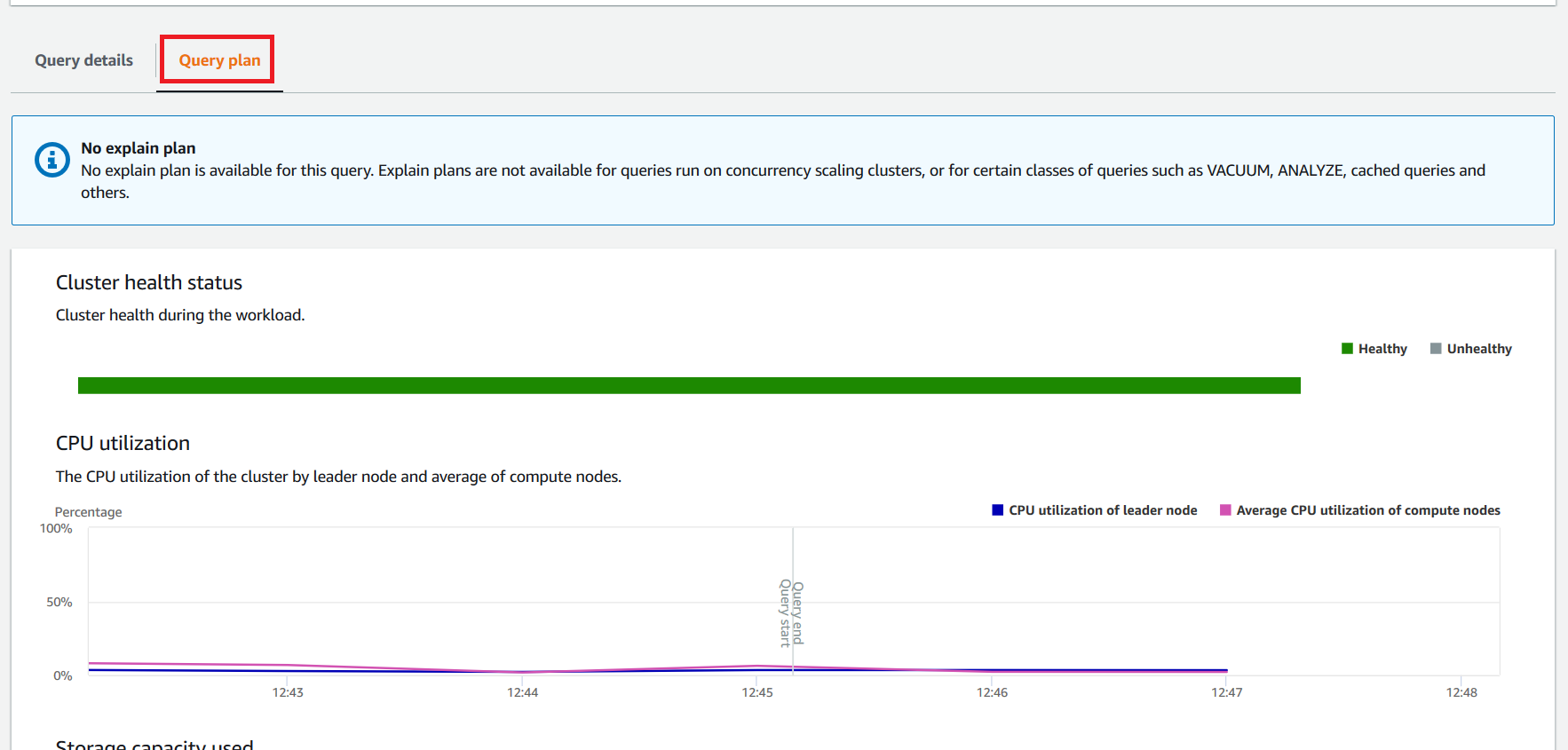
6.5 we can create manual backup



6.6 history of queries



6.7 usage per query



Also shows query start and end

6.8 delete the cluster