

Amazon Redshift

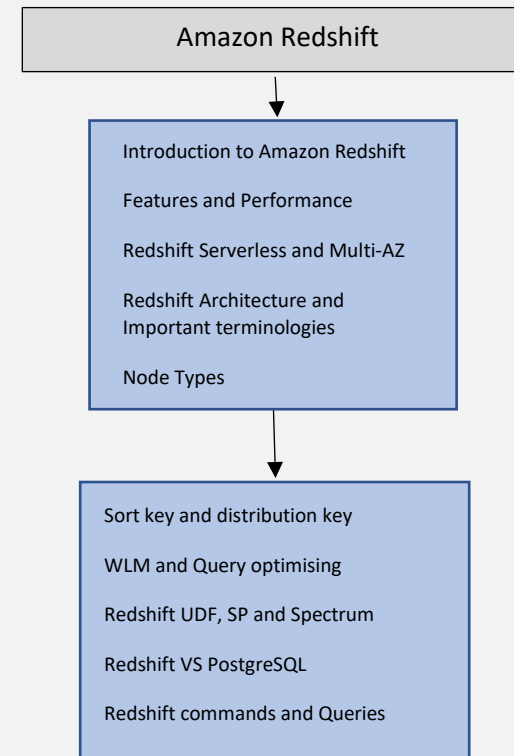
Amazon Redshift is a fully managed, fast, and powerful, petabyte-scale data warehouse service. Redshift is an OLAP data warehouse solution based on PostgreSQL.

As a part of Amazon Redshift, you covered:

- Introduction to Amazon Redshift
- Redshift Features and Performance
- Redshift Architecture
- Redshift commands and Queries

Common Interview Questions:

1. What is Redshift in AWS?
2. What are the benefits of using AWS Redshift?
3. Why use an AWS Data Pipeline to load CSV into Redshift? And How?
4. How to list tables in Amazon Redshift?
5. How are Amazon RDS, DynamoDB, and Redshift different?
6. : How far Redshift is better in performance as compared to other data warehouse technologies?
7. How do we load data into Redshift?
8. How will the price of Amazon Redshift vary?



Amazon Redshift

Amazon Redshift:

Amazon Redshift is a fully managed, fast, and powerful, petabyte-scale data warehouse service.

Redshift is an OLAP data warehouse solution based on PostgreSQL.

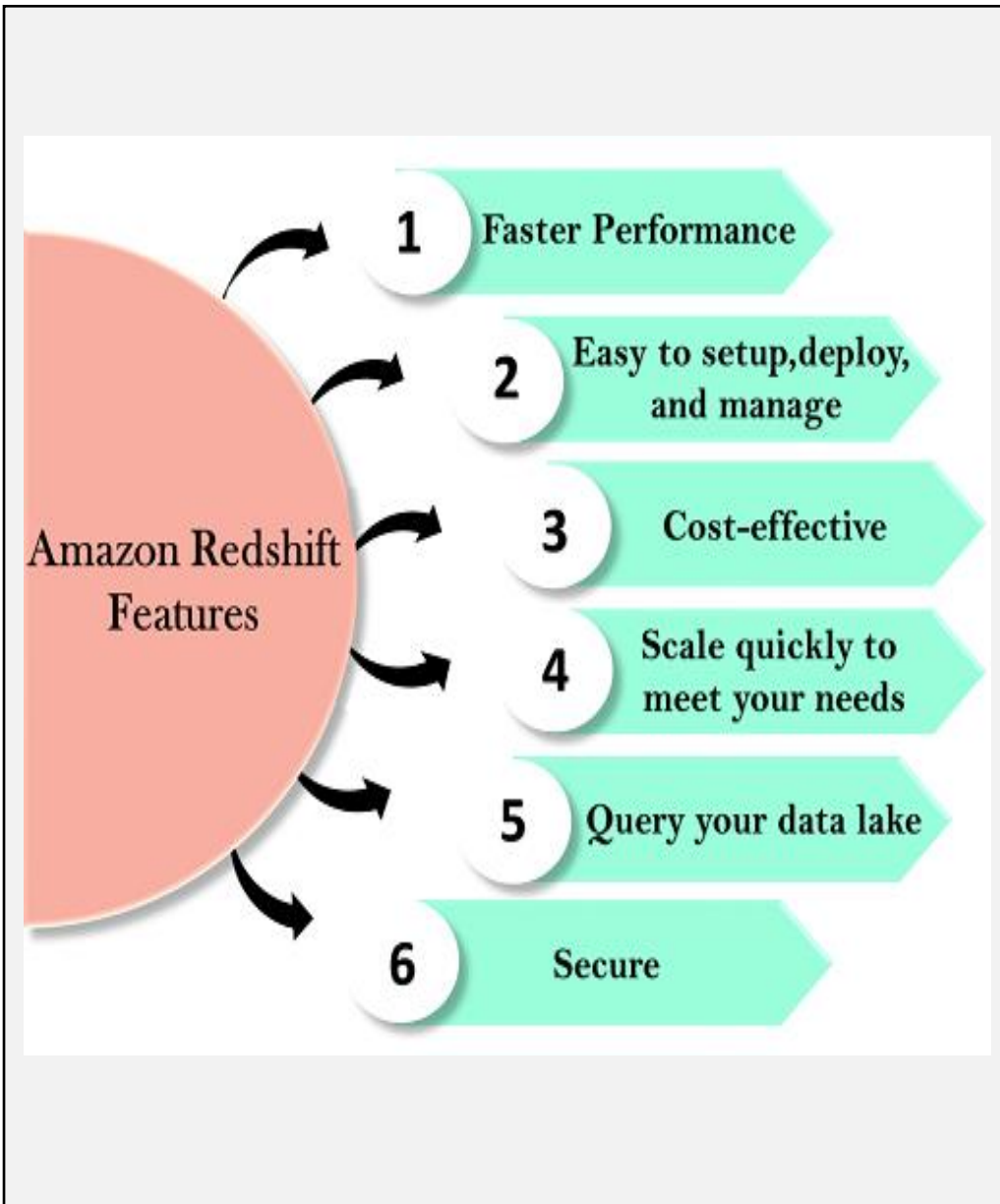
Redshift only supports Single-AZ deployments.

Redshift automatically helps set up, operate, and scale a data warehouse, from provisioning the infrastructure capacity.

- patches and backs up the data warehouse, storing the backups for a user-defined retention period.
- monitors the nodes and drives to help recovery from failures.
- significantly lowers the cost of a data warehouse, but also makes it easy to analyse large amounts of data very quickly
- provide fast querying capabilities over structured and semi-structured data using familiar SQL-based clients and business intelligence (BI) tools using standard ODBC and JDBC connections.
- uses replication and continuous backups to enhance availability and improve data durability and can automatically recover from node and component failures.
- scale up or down with a few clicks in the AWS Management Console or with a single API call
- distributes & parallelize queries across multiple physical resources
- supports VPC, SSL, AES-256 encryption, and Hardware Security Modules (HSMs) to protect the data in transit and at rest.

Amazon Redshift

Features of Amazon Redshift:



Performance of Amazon Redshift:

Redshift Performance

Massively Parallel Processing: Amazon Redshift automatically distributes the data and loads the query across various nodes.

Advanced Compression: Columnar data stores can be compressed much more than row-based data stores because similar data is stored sequentially on disk.

Columnar Data Storage: Instead of storing data as a series of rows, Amazon Redshift organizes the data by column.

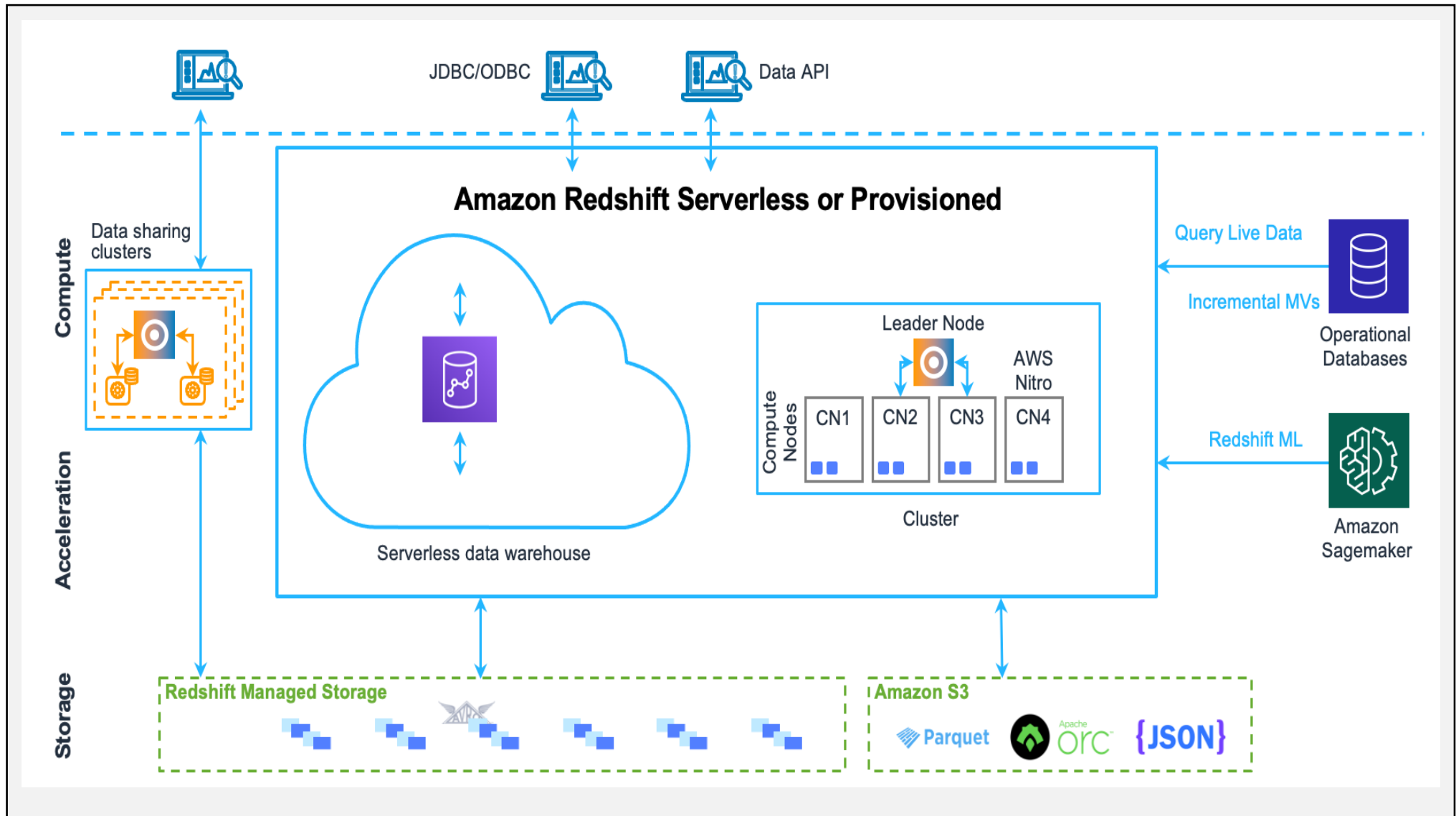
Query Optimizer: Redshift query run engine incorporates a query optimizer that is MPP-aware and also takes advantage of columnar-oriented data storage.

Result Caching: Redshift caches the results of certain types of queries in memory on the leader node.

Compiled code: Compiling the query decreases the overhead associated with an interpreter and therefore increases the runtime speed, especially for complex queries.

Amazon Redshift

Amazon Redshift Architecture:



Amazon Redshift

Important Terminologies:

Cluster: Cluster is composed of one or more compute nodes. If a cluster is provisioned with two or more compute nodes, an additional leader node coordinates the compute nodes and handles external communication.

Leader Node: It manages the client connections and receives queries. A leader node receives the queries from the client applications, parses the queries, and develops the execution plans. It coordinates with the parallel execution of these plans with the compute node and combines the intermediate results of all the nodes, and then return the final result to the client application.

Compute Node: A compute node executes the execution plans, and then intermediate results are sent to the leader node for aggregation before sending back to the client application. It can have up to 128 compute nodes.

Node slices: A compute node is partitioned into slices. Each slice is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node. 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. Number of slices per node is determined by the node size of the cluster.

Managed Storage: Data warehouse data is stored in a separate storage tier Redshift Managed Storage (RMS). RMS provides the ability to scale the storage to petabytes using S3 storage.

RMS enables scale and pay for compute and storage independently so that the cluster can be sized based only on the compute needs.

RMS automatically uses high-performance SSD-based local storage as tier-1 cache.

Amazon Redshift

Amazon Redshift Serverless:

Redshift Serverless

Redshift Serverless is a serverless option of Redshift that makes it more efficient to run and scale analytics in seconds without the need to set up and manage data warehouse infrastructure.

- Redshift Serverless automatically provisions and intelligently scales data warehouse capacity to deliver high performance for demanding and unpredictable workloads.
- Redshift Serverless helps any user to get insights from data by simply loading and querying data in the data warehouse.
- Redshift Serverless supports Concurrency Scaling feature that can support unlimited concurrent users and concurrent queries, with consistently fast query performance.
- When concurrency scaling is enabled, Redshift automatically adds cluster capacity when the cluster experiences an increase in query queueing.

Amazon Redshift Multi-AZ:

Redshift Multi-AZ

Redshift Multi-AZ deployment runs the data warehouse in multiple AWS AZs simultaneously and continues operating in unforeseen failure scenarios.

- Multi-AZ deployment is managed as a single data warehouse with one endpoint and does not require any application changes.
- Multi-AZ deployments support high availability requirements and reduce recovery time by guaranteeing capacity to automatically recover and are intended for customers with business-critical analytics applications that require the highest levels of availability and resiliency to AZ failures.
- Redshift Multi-AZ supports RPO = 0 meaning data is guaranteed to be current and up to date in the event of a failure. RTO is under a minute.

Amazon Redshift

Amazon Redshift Node Types:

RA3 Node: RA3 features high-speed caching, managed store, and high bandwidth networking. In the new RA3 generation instance type, Redshift stores permanent data to S3 and uses the local disk for caching purposes.

RA3 node types						
Node size	vCPU	RAM (GiB)	Default slices per node	Managed storage quota per node	Node range with create cluster	Total capacity
ra3.4xlarge	12	96	4	64 TB ¹	2–32 ²	4096 TB ^{2,3}
ra3.16xlarge	48	384	16	64 TB ¹	2–128	8192 TB ³

Dense Compute Node (DC2): Dense Compute nodes (DC2) are optimized for processing data and are compute-intensive data warehouses that use SSD for local storage. DC2 stores the data locally for high performance, and it allows you to add more compute nodes if you need extra space.

Dense compute node types						
Node size	vCPU	RAM (GiB)	Default slices per node	Storage per node	Node range	Total capacity
dc2.large	2	15	2	160 GB NVMe-SSD	1–32	5.12 TB
dc2.8xlarge	32	244	16	2.56 TB NVMe-SSD	2–128	326 TB
dc1.large ¹	2	15	2	160 GB SSD	1–32	5.12 TB
dc1.8xlarge ¹	32	244	32	2.56 TB SSD	2–128	326 TB

Dense Storage Node (DS2): DS2 allows you to have a storage-intensive data warehouse with vCPU and RAM included for computation. DS2 nodes use HDD(Hard Disk Drive) for storage and as a rule of thumb, if you have data more than 500 GB, then it is advisable to go for DS2 instances.

Dense storage node types						
Node size	vCPU	RAM (GiB)	Default slices per node	Storage per node	Node range	Total capacity
ds2.xlarge	4	31	2	2 TB HDD	1–32	64 TB
ds2.8xlarge	36	244	16	16 TB HDD	2–128	2 PB

Amazon Redshift

Amazon Redshift SortKeys:

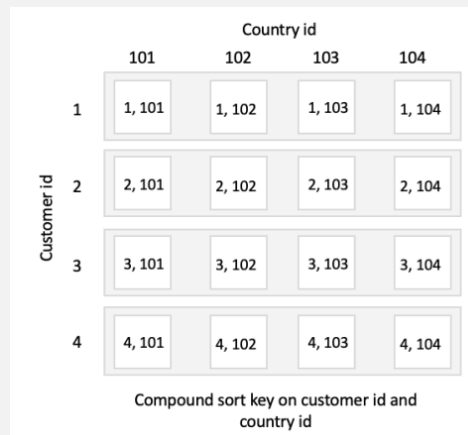
Redshift Sortkeys determines the order in which rows in a table are stored. Query performance is improved when Redshift Sortkeys are properly used as it enables the query optimizer to read fewer chunks of data filtering out the majority of it.

Compound Redshift Sortkeys

These are made up of all the columns that are listed in the Redshift Sortkeys definition during the creation of the table, in the order that they are listed. Therefore, it is advisable to put the most frequently used column at the first in the list.

For example: let us create a table with 2 Compound Redshift sortkeys.

```
CREATE TABLE customer ( c_customer_id INTEGER NOT NULL, c_country_id INTEGER NOT NULL, c_name VARCHAR(100) NOT NULL) COMPOUND SORTKEY(c_customer_id, c_country_id);
```



As we can see if you want to get all country IDs for a customer, you would require access to one block. If you need to get IDs for all customers with a specific country, you need to access all four blocks. This shows that we are unable to optimize two kinds of queries at the same time using Compound Sorting

Amazon Redshift

Interleaved Redshift Sortkeys

Interleaved Sort gives equal weight to each column in the Redshift Sortkeys. As a result, it can significantly improve query performance where the query uses restrictive predicates (equality operator in WHERE clause) on secondary sort columns.

For example, let's create a table with Interleaved Sort Keys.

```
CREATE TABLE customer (c_customer_id INTEGER NOT NULL, c_country_id INTEGER NOT NULL) INTERLEAVED  
SORTKEY (c_customer_id, c_country_id);
```



As you can see, the first block stores the first two customer IDs along with the first two country IDs. Therefore, you only scan 2 blocks to return data to a given customer or a given country.

- Use Interleaved Sort Keys when you plan to use one column as Sort Key or when WHERE clauses in your query have highly selective restrictive predicates. Or if the tables are huge. Don't use an Interleaved Sort Key on columns with monotonically increasing attributes, like an identity column, dates, or timestamps.
- Use Compound Sort Keys when you have more than one column as Sort Key, when your query includes JOINS, GROUP BY, ORDER BY, and PARTITION BY when your table size is small.

Amazon Redshift

Amazon Redshift Distribution Key:

Redshift Distribution Keys (DIST Keys) determine where data is stored in Redshift. Clusters store data fundamentally across the compute nodes. Query performance suffers when a large amount of data is stored on a single node.

The query optimizer distributes less number of rows to the compute nodes to perform joins and aggregation on query execution. This redistribution of data can include shuffling of the entire tables across all the nodes.

Redshift EVEN distribution

This is the default distribution styles of a table. In Even Distribution the Leader node of the cluster distributes the data of a table evenly across all slices, using a round-robin approach.

Example:

```
create table sample (id int, name
varchar(100), age int) DISTSTYLE
EVEN;
```

Redshift KEY distribution

The data is distributed across slices by the leader node matching the values of a designated column. So all the entries with the same value in the column end up in the same slice.

Example:

```
create table sample (id int, name
varchar(100), age int) DISTSTYLE KEY
DISTKEY(ID);
```

Redshift ALL distribution

If you specify the ALL distribution style during table creation then leader node distributes the copy of tables every node available in the cluster. If the table is small and want make colocated tables then this distribution style is optimal.

Example:

```
create table sample (id int, name
varchar(100), age int) DISTSTYLE ALL;
```

Amazon Redshift

Redshift workload management (WLM):

Redshift workload management (WLM) is used to define multiple query queues and to route queries to the appropriate queues at runtime. For example there can separate queues created for ETL, reporting and adhoc use cases .Each queue will be configured to have its own memory % ,query slot count and Query monitoring rules(QMR). Default queue will handle queries which doesn't under any of the user created queue.

Properties of each Queues

- Priority
- Concurrency scaling mode
- User groups
- Query groups
- Query monitoring rules

Types of Redshift Workload Management

Automatic WLM

Amazon Redshift manages the resources effectively to handle user queries when automatic WLM is enabled.Amazon Redshift determines how many queries run concurrently and how much memory is allocated to each dispatched query. Memory and concurrency settings are configured as auto

Manual WLM

Manual WLM provides full control to the user to manage the concurrency setting based on the query patterns and end users of the cluster. Manual WLM is suited for redshift clusters which has diverse set of business use cases and query patterns.

Amazon Redshift

Query Optimising Planning:

Some Redshift Query Optimising Planning

- Write specific, well-structured queries: Writing well-structured queries that are as specific as possible will allow you to get results as quickly as possible



```
SELECT *  
FROM table;
```



```
SELECT name, email  
FROM table  
WHERE time > '2020-10-01'
```

- Check to see if you're running out of space: The query below will give you the percentage of storage used in your cluster, which you can use to check if this is the reason for slow query performance. If you're nearing capacity (80% or greater), consider adding some more nodes or truncating some unused historical data.

```
SELECT sum(pct_used)  
FROM svv_table_info;
```

- Using Redshift is Query Plan: A query plan is what query engine follows to execute a query such as what join types to use and whether data needs to be copied from one node to another (redistribution)
- Stagger ETL processes to avoid overlap: If you have multiple ETL processes loading into your warehouse at the same time, everything will slow down. Try to schedule them at different times when your cluster is least active.

Amazon Redshift

Amazon Redshift UDF:

User Defined Functions in Redshift:

UDFs are scalar functions that can be customized and created from the Redshift data warehouse.

- Amazon recommends that all UDF names begin with f_
- UDFs can be created using a SQL select statement or as a Python function.

SQL UDF: This is the structure of a SQL UDF:

```
create function f_sql_udf ([arg1 data type], ...)
returns [return data type]
stable as $$

select ...

$$ language sql;
```

Python UDF: This is the structure of a Python UDF

```
create function f_python_udf (arg_name [arg1 data type], ...)
returns [return data type]
stable as $$

[Python code here]

$$ language plpythonu;
```

Amazon Redshift Stored Procedure:

Amazon Stored Procedure:

The stored procedure is a user-defined routine saved in the database and can execute by other external applications. Below is the Example of writing SP

```
CREATE PROCEDURE redshift_example_sp()
AS $$
BEGIN
    RAISE INFO 'This is an example of Redshift Stored Procedure';
END;
$$
LANGUAGE plpgsql
;
```







Amazon Redshift Spectrum:





Amazon Redshift Spectrum:

Amazon Redshift Spectrum is a feature within Amazon Web Services' Redshift data warehousing service that lets a data analyst conduct fast, complex analysis on objects stored on the AWS cloud. With Redshift Spectrum, an analyst can perform SQL queries on data stored in Amazon S3 buckets. This can save time and money because it eliminates the need to move data from a storage service to a database, and instead directly queries data inside an S3 bucket.

Amazon Redshift

Amazon Redshift VS PostgreSQL:

PostgreSQL	RedShift
 <p>Data is stored and managed in rows that helps in creating tables directly. This helps to build queries around the rows inserted and also we can manage the tables in the way the data got inserted into the tables.</p>	 <p>Data is inserted in the form of columns. This helps to read data faster and return the queries more efficiently than PostgreSQL. Storage efficiency is increased here as compression of data happens in column level since each column carries similar data.</p>
PostgreSQL	RedShift
 <p>Scaling is not easy in PostgreSQL as the compute nodes are not present in this database. Scaling can be done only by creating a new server for the data or copying entire data into a separate database. Vertical scaling is done with PostgreSQL which is costly.</p>	 <p>Scaling is easy in RedShift as AWS helps to manage node configuration and scale horizontally with parallel processing. This expansion of nodes helps to create more clusters. Horizontal scaling will not require more servers as it is done with the help of compute nodes so that the scaling can be done at low cost.</p>
PostgreSQL	RedShift
 <p>PostgreSQL is simple and easy to use. We can insert data and write queries in T-SQL which will give us the results. But this is suited when the data is less. If we have more data, the speed is reduced and it is not possible to scale the database easily. This is good for beginners as it is free of cost.</p>	 <p>RedShift is not freely offered and it is used along with Amazon S3 storage. But the querying is faster and the queries are similar to PostgreSQL. For any amount of data, querying can be done within minutes and this characteristic make users to select RedShift.</p>

PostgreSQL	RedShift
 <p>PostgreSQL supports arrays, bits, range, JSON, numeric and geometric types, XML, timestamp data and many other forms. If it is not big data, it is good to stick with PostgreSQL as it has different functions and triggers along with sequences.</p>	 <p>RedShift does not support all the functions as PostgreSQL and particularly no support for timestamp data. These are some disadvantages we should foresee when we go for scaling and performance of the database. For bigdata, RedShift is a good choice.</p>
PostgreSQL	RedShift
 <p>Single database is connected to one CPU and hence the data should be processed one after the other. We cannot expect the database to function faster with single CPU where scaling is not an option. We do not have clusters and only nodes are present.</p>	 <p>Massive parallel processing is done in RedShift which helps to process data simultaneously and this helps in completing the queries in a faster pace than expected. Different nodes are employed to carryon the process in the database with the cluster configuration.</p>

Amazon Redshift

Amazon Redshift Commands and Queries:

SELECT and SELECT INTO Queries:

A SELECT statement to display the data contained in the table.

```
1 select
2     custom_id,
3     count(*) as order_qty,
4     sum(amount) as total_amount
5 from
6     source.orders
7 group by
8     customer_id
9
10
11
12
13
```

In Redshift, the SELECT INTO statement retrieves data from one or more database tables and assigns the values to variables.

```
1 drop table if exists derived.aggregate_orders;
2
3 select
4     custom_id,
5     count(*) as order_qty,
6     sum(amount) as total_amount
7 into
8     derived.aggregate_orders
9 from
10    source.orders
11 group by
12     customer_id
13
```

Create table Command:

Example:

```
CREATE TABLE Employees
(
    employee_id    integer(30),
    first_name     varchar(30),
    last_name      varchar(30),
    hire_date      date,
    salary         integer
);
```

Amazon Redshift also allows users to modify their commands using parameters/keywords such as “Like”, “As”, “Default”, etc. to access or load their Amazon Redshift data with ease

Amazon Redshift

Copy command to fetch data in Redshift from S3 bucket:

The screenshot displays the Aginity Pro interface with a Redshift connection named 'dev' selected. The SQL editor contains a COPY command to load data from an S3 bucket into a Redshift table. The command is as follows:

```
1 copy sales (ID, Country, State, City, Amount)
2 from 's3://rahul-test-data/Sales'
3 iam_role 'arn:aws:iam::          role/S3_Access_Role_For_Redshift'
4 csv NOLOAD
5 IGNOREHEADER 1
```

The command was executed successfully, as indicated by the 'SUCCESS' status in the output pane. The output also shows the execution time (4.69s) and a warning message: 'WARNING: (SQLSTATE: 01000, SQLCODE: 0): Load into table 'sales' completed, 0 record(s) loaded successfully.'

The interface includes a sidebar with a 'Database Explorer' showing the 'dev' database and its schemas. The bottom status bar indicates the connection is 'Redshift - Preview' and the result is 'SUCCESS' with a duration of '252ms' and '1 row(s)'.