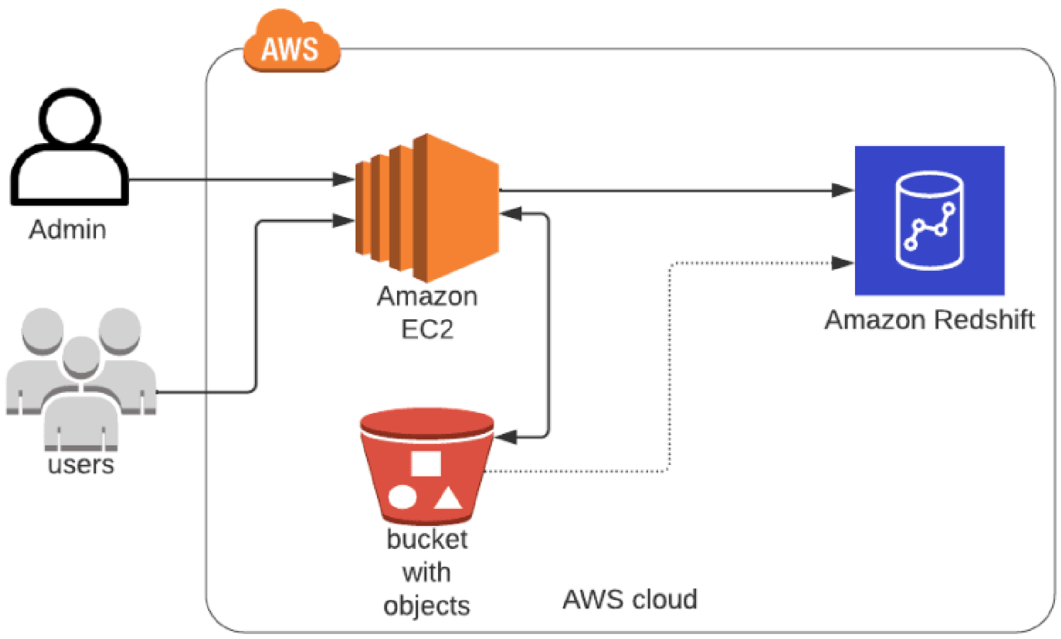
Lab 3: Working with Amazon Redshift clusters

1. Identify Amazon Redshift as a data warehouse solution.
2. Launch an Amazon Redshift cluster.
3. Create tables with Automatic Table Optimization.
4. Load data from an Amazon S3 bucket to an Amazon Redshift cluster.
5. Identify table columns to designate as sort keys.

NOT LAB GIVEN DIAGRAM-



My thinking - redshift is like EC2 for Data Analysis
Meaning it offer ways to handle very large amount of data, and get analysis from it.
Reason -
Think of a Redshift cluster as a group of super-powerful computers (servers) working together. They are specially designed to handle huge amounts of data and perform complex tasks really quickly.

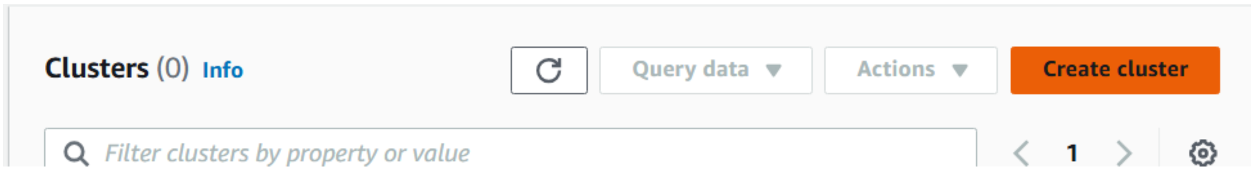
Amazon Redshift Spectrum - lets you analyze data stored in Amazon S3 without having to load it into Redshift first.

Pre-step - connect to CLI using Url given

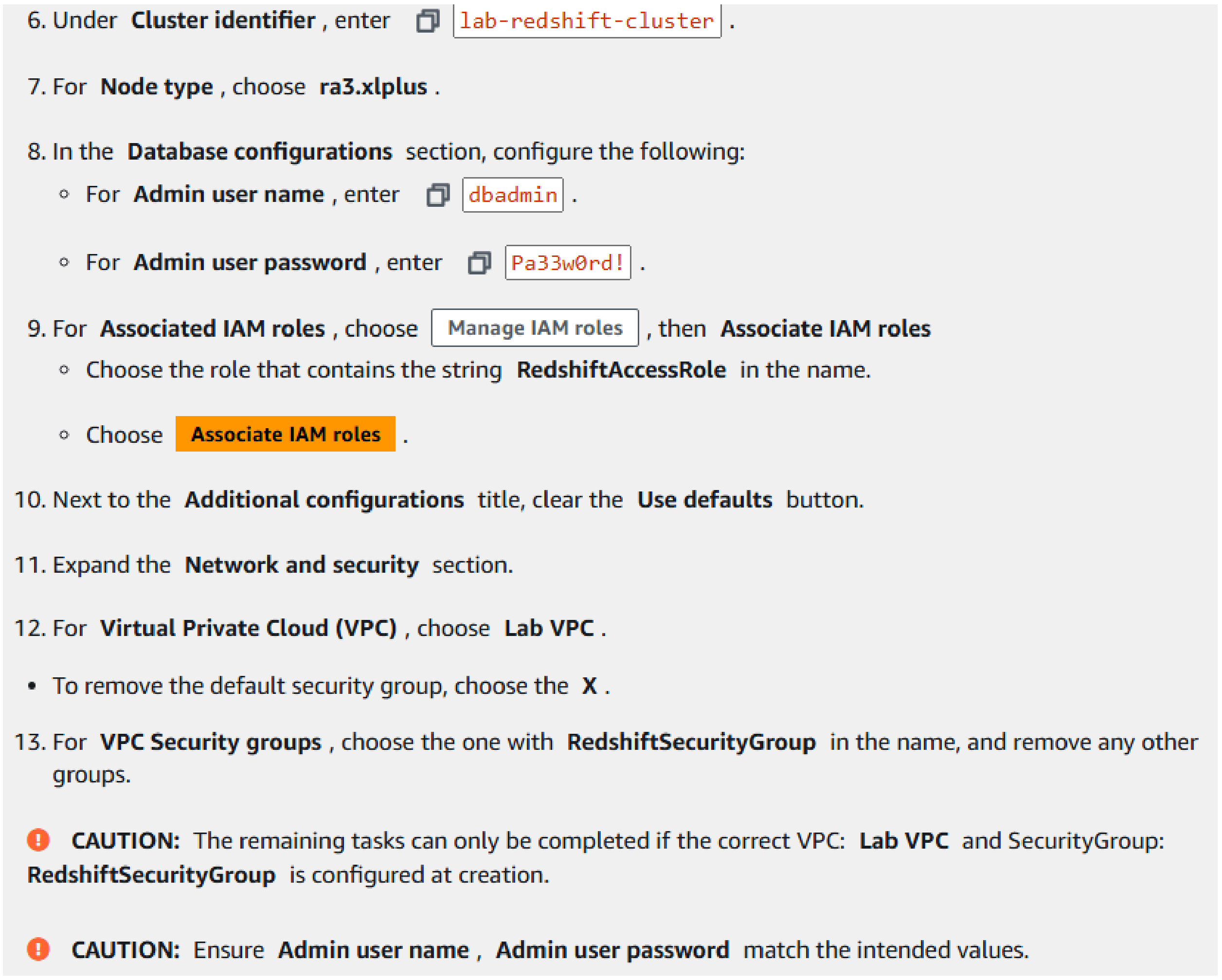
Task 1: Create a two node Amazon Redshift cluster

Two nodes mean two computer/instances will work from RedShift side

1.1 open redshift in console, click on create cluster



1.2 enter these values



6. Under **Cluster identifier** , enter `lab-redshift-cluster` .

7. For **Node type** , choose **ra3.xlplus** .

8. In the **Database configurations** section, configure the following:
   - For **Admin user name** , enter `dbadmin` .
   - For **Admin user password** , enter `Pa33w0rd!` .

9. For **Associated IAM roles** , choose Manage IAM roles , then **Associate IAM roles**
   - Choose the role that contains the string **RedshiftAccessRole** in the name.
   - Choose **Associate IAM roles** .

10. Next to the **Additional configurations** title, clear the **Use defaults** button.

11. Expand the **Network and security** section.

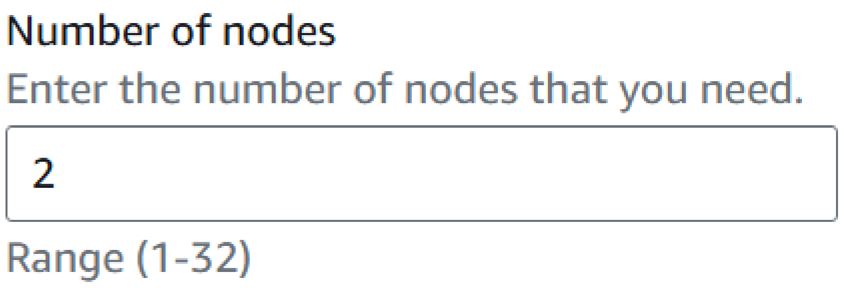
12. For **Virtual Private Cloud (VPC)** , choose **Lab VPC** .

- To remove the default security group, choose the **X** .

13. For **VPC Security groups** , choose the one with **RedshiftSecurityGroup** in the name, and remove any other groups.

CAUTION: The remaining tasks can only be completed if the correct VPC: **Lab VPC** and SecurityGroup: **RedshiftSecurityGroup** is configured at creation.

CAUTION: Ensure **Admin user name** , **Admin user password** match the intended values.



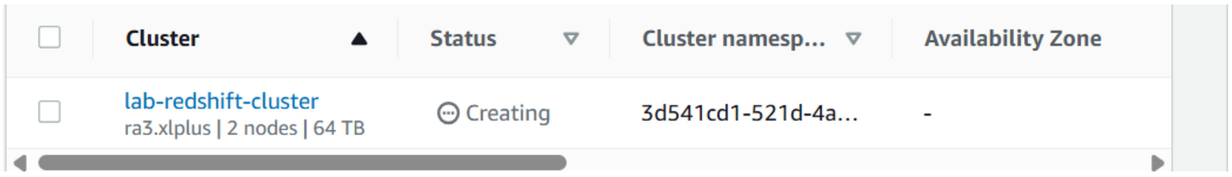
Number of nodes

Enter the number of nodes that you need.

2

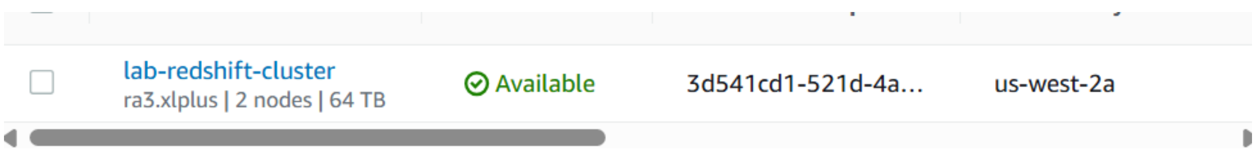
Range (1-32)

1.3 click create, wait

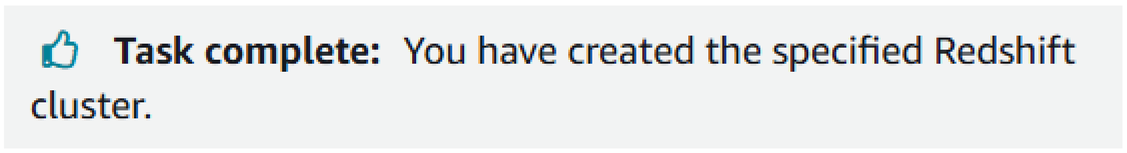


	Cluster	Status	Cluster namesp...	Availability Zone
☐	lab-redshift-cluster ra3.xlplus | 2 nodes | 64 TB	Creating	3d541cd1-521d-4a...	-

1.4 done



☐	lab-redshift-cluster ra3.xlplus | 2 nodes | 64 TB	✓ Available	3d541cd1-521d-4a...	us-west-2a



Task complete: You have created the specified Redshift cluster.

Task 2: Creating tables in Amazon Redshift

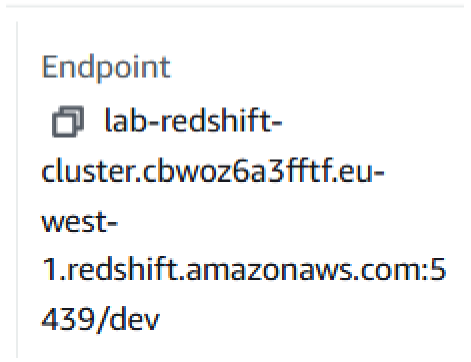
use the psql interface to create tables on the Amazon Redshift cluster that holds data.

Tables are where your data is stored within the cluster.

We use psql to create tables in redshift
We will send command to redshift cluster DB, and we will do this via psql (via CLI).

2.1 Open the created cluster by clicking on it

2.2 Copy the endpoint value somewhere

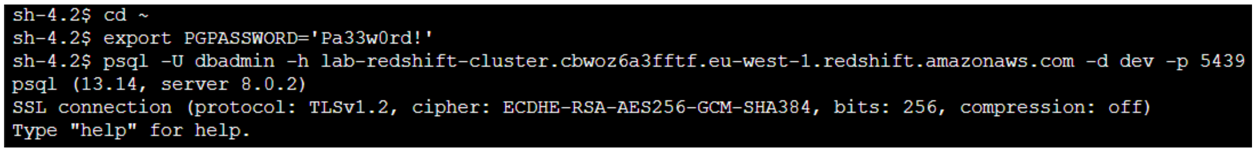


Endpoint
lab-redshift-cluster.cbwoz6a3fftf.eu-west-1.redshift.amazonaws.com:5439/dev

2.3 enter this in CLI

```
cd ~
export PGPASSWORD='Pa33w0rd!'
psql -U dbadmin -h RedshiftClusterEndpoint -d dev -p 5439
```

Replace RedshiftClusterEndpoint with 2.2 value



```
sh-4.2$ cd ~
sh-4.2$ export PGPASSWORD='Pa33w0rd!'
sh-4.2$ psql -U dbadmin -h lab-redshift-cluster.cbwoz6a3fftf.eu-west-1.redshift.amazonaws.com -d dev -p 5439
psql (13.14, server 8.0.2)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.
```

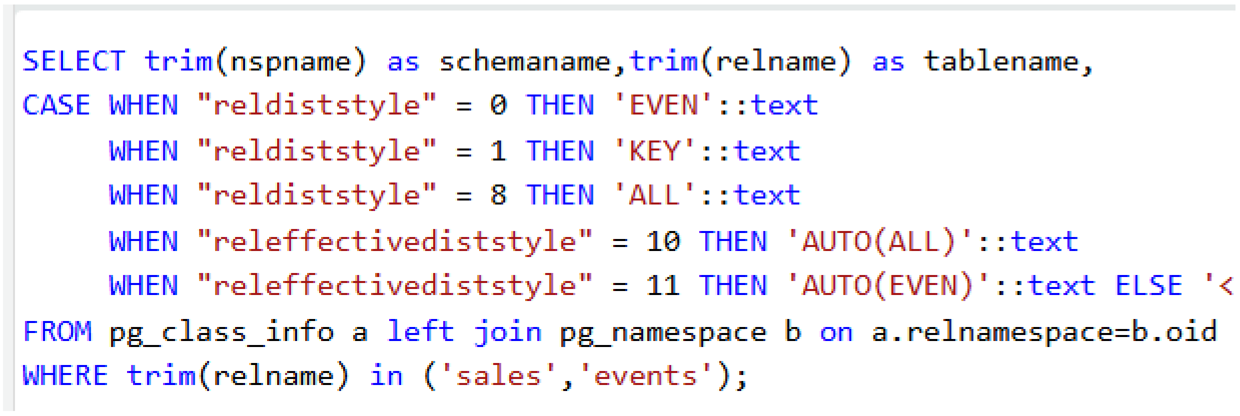
This should log you into the database and give you a prompt where you can enter **SQL** commands used in this lab.

2.4 run these sql commands:

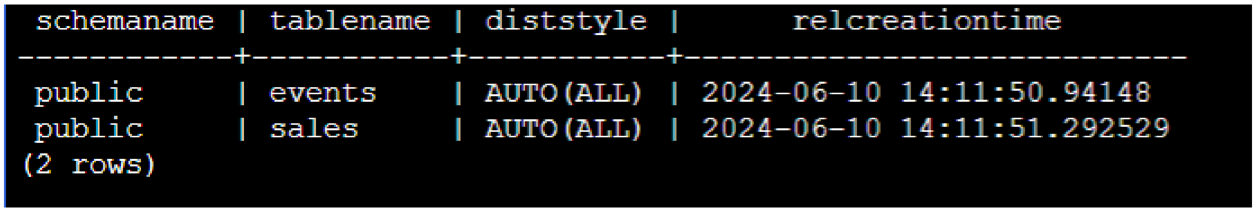
```
CREATE TABLE events (
    eventid bigint NOT NULL,
    event_category VARCHAR(30) NOT NULL
);

CREATE TABLE sales (
    salesid bigint NOT NULL,
    saletime timestamp without time zone NOT NULL,
    sale_category varchar(20) NOT NULL,
    eventid bigint NOT NULL,
    qtysold integer NOT NULL,
    pricepaid numeric(11,2) NOT NULL,
    commission numeric(11,2) NOT NULL,
    saleyear smallint NOT NULL,
    salemonth smallint NOT NULL,
    saleday smallint NOT NULL
);
```

Tables created

2.5 now run these

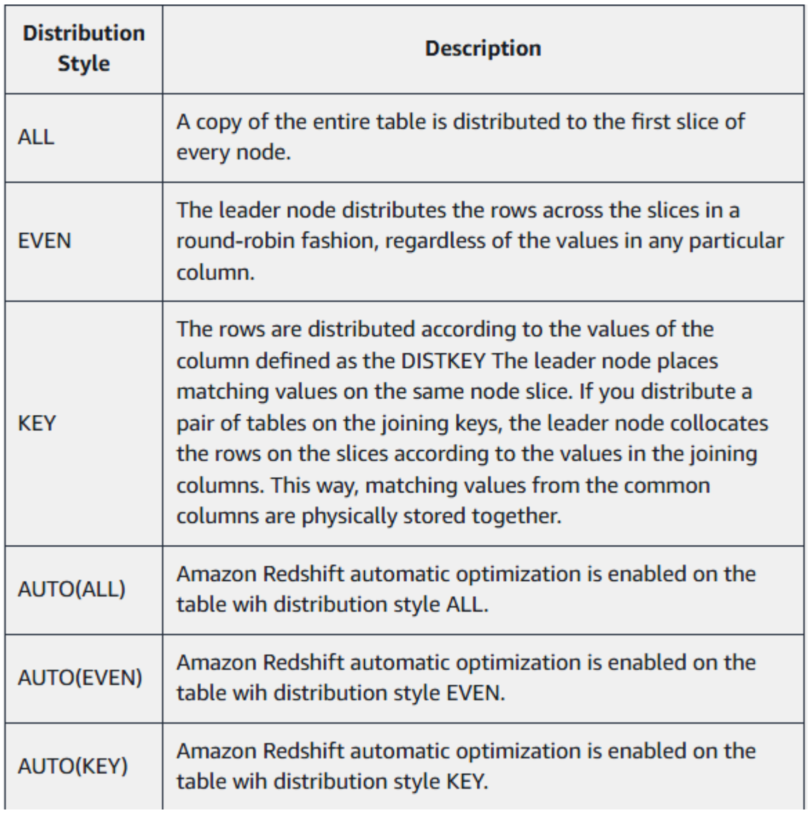


```
SELECT trim(nspname) as schemaname,trim(relname) as tablename,
CASE WHEN "reldiststyle" = 0 THEN 'EVEN'::text
     WHEN "reldiststyle" = 1 THEN 'KEY'::text
     WHEN "reldiststyle" = 8 THEN 'ALL'::text
     WHEN "releffectivediststyle" = 10 THEN 'AUTO(ALL)'::text
     WHEN "releffectivediststyle" = 11 THEN 'AUTO(EVEN)'::text ELSE '<
FROM pg_class_info a left join pg_namespace b on a.relnamespace=b.oid
WHERE trim(relname) in ('sales','events');
```



```
 schemaname | tablename | diststyle |       relcreationtime
------------+-----------+-----------+----------------------------
 public     | events    | AUTO(ALL) | 2024-06-10 14:11:50.94148
 public     | sales     | AUTO(ALL) | 2024-06-10 14:11:51.292529
(2 rows)
```

What this command does:
This SQL query retrieves information about tables named "sales" and "events" from the Amazon Redshift system catalog tables.



Distribution Style	**Description**
ALL	A copy of the entire table is distributed to the first slice of every node.
EVEN	The leader node distributes the rows across the slices in a round-robin fashion, regardless of the values in any particular column.
KEY	The rows are distributed according to the values of the column defined as the DISTKEY The leader node places matching values on the same node slice. If you distribute a pair of tables on the joining keys, the leader node collocates the rows on the slices according to the values in the joining columns. This way, matching values from the common columns are physically stored together.
AUTO(ALL)	Amazon Redshift automatic optimization is enabled on the table wih distribution style ALL.
AUTO(EVEN)	Amazon Redshift automatic optimization is enabled on the table wih distribution style EVEN.
AUTO(KEY)	Amazon Redshift automatic optimization is enabled on the table wih distribution style KEY.

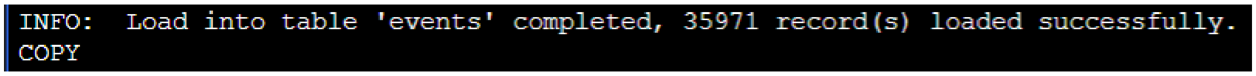
Task 3: Load tables with data from S3 objects

3.1 use this to copy data

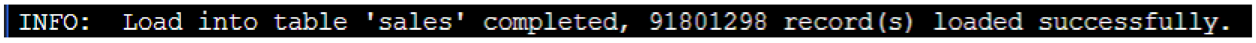
```
COPY events
FROM 's3://S3LABBUCKET/datawarehouse/events/'
IAM_ROLE 'IAMROLEARN' DELIMITER '|'  GZIP;

COPY sales
FROM 's3://S3LABBUCKET/datawarehouse/sales/'
IAM_ROLE 'IAMROLEARN' DELIMITER '|'  GZIP;
```

Replace S3 bucket path, and IAM ARN

Output-

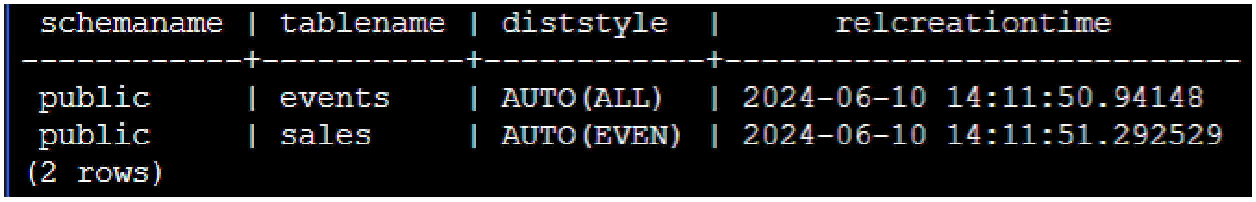


```
INFO:  Load into table 'events' completed, 35971 record(s) loaded successfully.
COPY
```



```
INFO:  Load into table 'sales' completed, 91801298 record(s) loaded successfully.
```

3.2 view the distribution style once again, it has changed



```
 schemaname | tablename | diststyle   |       relcreationtime
------------+-----------+-------------+----------------------------
 public     | events    | AUTO(ALL)   | 2024-06-10 14:11:50.94148
 public     | sales     | AUTO(EVEN)  | 2024-06-10 14:11:51.292529
(2 rows)
```

Task 4: Working with sort keys

We have to assign columns as sort keys

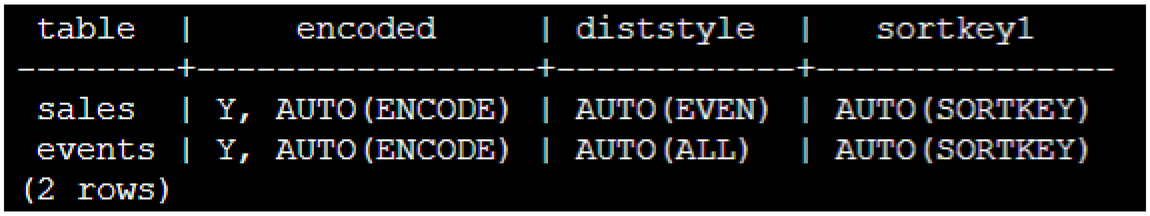
4.1 Run this to get sort keys for both tables

SELECT sti.table, encoded, diststyle, sortkey1
FROM SVV_TABLE_INFO sti
WHERE sti.database ='dev' AND sti.table in ('events','sales') order by size desc;

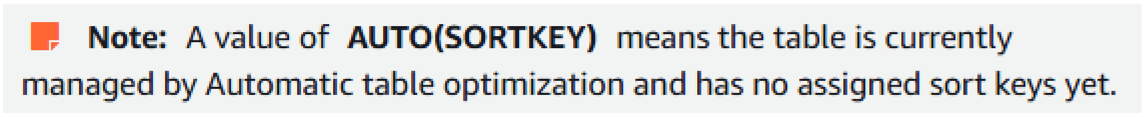
sti.table retrieves the name of the table.

SVV_TABLE_INFO is a system view in Amazon Redshift that contains metadata information about tables in the database.

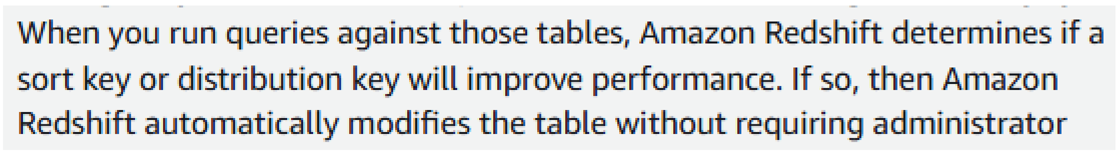
WHERE - filters the results to include only tables in the 'dev' database with names 'events' or 'sales'.



```
 table  |      encoded      | diststyle  |    sortkey1
--------+-------------------+------------+----------------
 sales  | Y, AUTO(ENCODE)  | AUTO(EVEN) | AUTO(SORTKEY)
 events | Y, AUTO(ENCODE)  | AUTO(ALL)  | AUTO(SORTKEY)
(2 rows)
```



Note: A value of **AUTO(SORTKEY)** means the table is currently managed by Automatic table optimization and has no assigned sort keys yet.



When you run queries against those tables, Amazon Redshift determines if a sort key or distribution key will improve performance. If so, then Amazon Redshift automatically modifies the table without requiring administrator

4.2 creating a new table

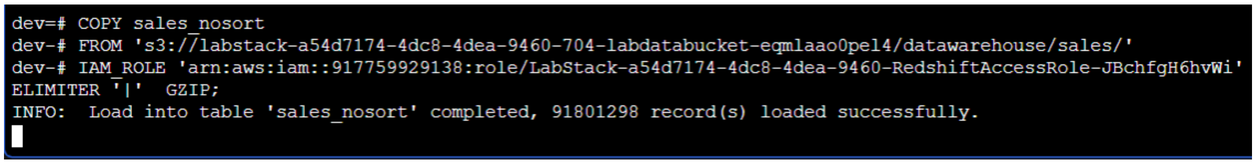
DROP TABLE IF EXISTS sales_nosort;

```
CREATE TABLE sales_nosort (
    salesid bigint NOT NULL,
    saletime timestamp without time zone NOT NULL,
    sale_category varchar(20) NOT NULL,
    eventid bigint NOT NULL,
    qtysold integer NOT NULL,
    pricepaid numeric(11,2) NOT NULL,
    commission numeric(11,2) NOT NULL,
    saleyear smallint NOT NULL,
    salemonth smallint NOT NULL,
    saleday smallint NOT NULL
);
```

4.3 load data into this table

```
COPY sales_nosort
FROM 's3://S3LABBUCKET/datawarehouse/sales/'
IAM_ROLE 'IAMROLEARN' DELIMITER '|'  GZIP;
```

Replace the values of path and ARN

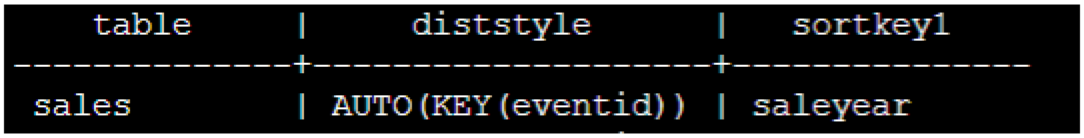


```
dev=# COPY sales_nosort
dev-# FROM 's3://labstack-a54d7174-4dc8-4dea-9460-704-labdatabucket-eqmlaao0pel4/datawarehouse/sales/'
dev-# IAM_ROLE 'arn:aws:iam::917759929138:role/LabStack-a54d7174-4dc8-4dea-9460-RedshiftAccessRole-JBchfgH6hvWi'
ELIMITER '|'  GZIP;
INFO:  Load into table 'sales_nosort' completed, 91801298 record(s) loaded successfully.
```

4.4 Run this to set the sort keys on the original sales table:

(here, we are setting saleyear, month as sort keys)
This will disable auto table optimizn

```
ALTER TABLE sales ALTER SORTKEY (saleyear, salemonth);
```



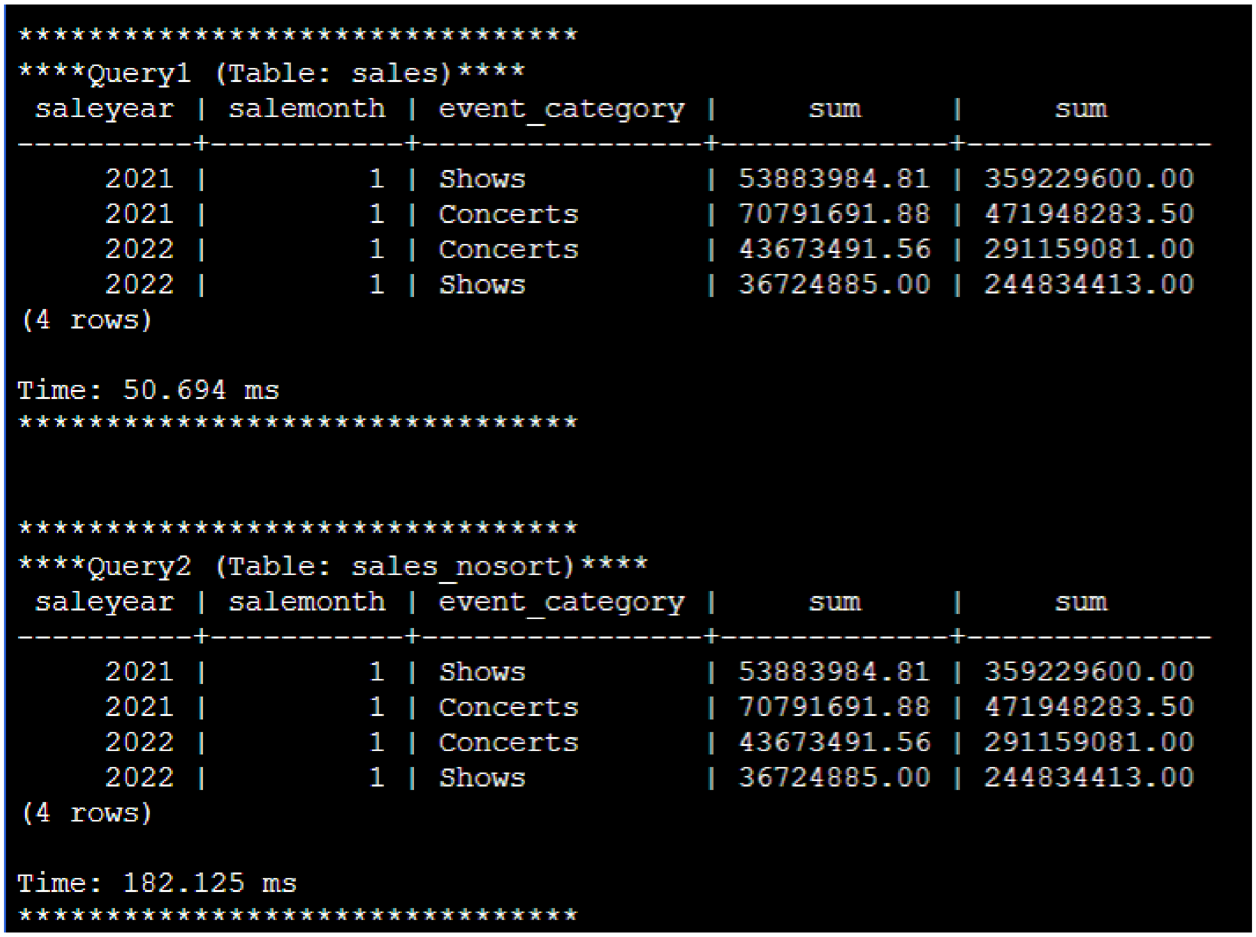
```
    table      |      diststyle       |    sortkey1
---------------+----------------------+----------------
 sales         | AUTO(KEY(eventid))   | saleyear
```

4.5 This query is performed on both the sales and sales_nosort tables to compare how much of a difference the sort key makes.

```
SELECT s.saleyear,s.salemonth,e.event_category, sum(s.commission),
sum(s.pricepaid)
FROM sales s, events e
WHERE s.saleyear BETWEEN 2021 AND 2022 AND s.salemonth=1
AND s.eventid = e.eventid
GROUP BY s.saleyear,s.salemonth,e.event_category
ORDER BY s.saleyear,s.salemonth;
```

And

```
for i in {00..05}; do psql -U dbadmin -h
lab-redshift-cluster.cbwoz6a3fftf.eu-west-1.redshift.amazonaws.com -d dev
-p 5439 -f /temp/compare-sort.sql; sleep 2; done
```



```
********************************
****Query1 (Table: sales)****
 saleyear | salemonth | event_category |     sum      |     sum
----------+-----------+----------------+--------------+--------------
     2021 |         1 | Shows          | 53883984.81 | 359229600.00
     2021 |         1 | Concerts       | 70791691.88 | 471948283.50
     2022 |         1 | Concerts       | 43673491.56 | 291159081.00
     2022 |         1 | Shows          | 36724885.00 | 244834413.00
(4 rows)

Time: 50.694 ms
********************************


********************************
****Query2 (Table: sales_nosort)****
 saleyear | salemonth | event_category |     sum      |     sum
----------+-----------+----------------+--------------+--------------
     2021 |         1 | Shows          | 53883984.81 | 359229600.00
     2021 |         1 | Concerts       | 70791691.88 | 471948283.50
     2022 |         1 | Concerts       | 43673491.56 | 291159081.00
     2022 |         1 | Shows          | 36724885.00 | 244834413.00
(4 rows)

Time: 182.125 ms
********************************
```

Notice the time difference

Task complete: You have reviewed the performance impact of using sort keys.