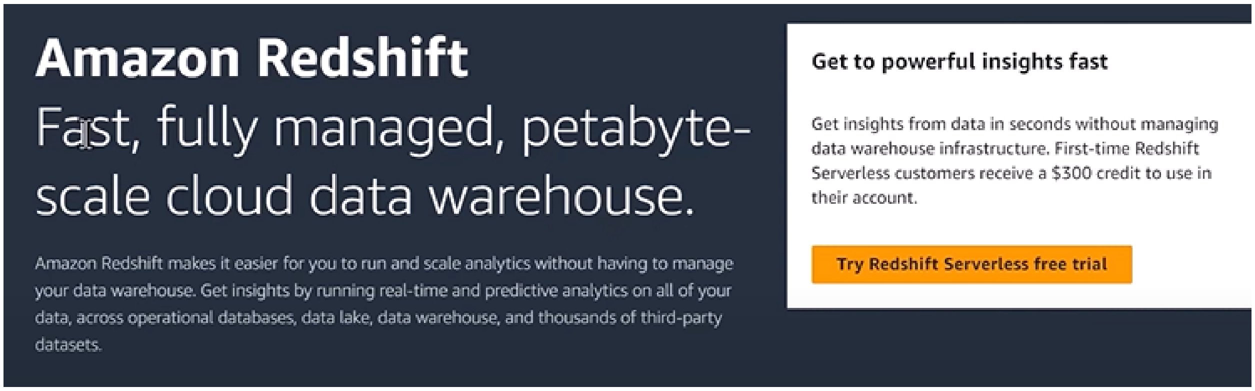
Transfer Data from s3 to redshift

1. Now we have to create redshift cluster



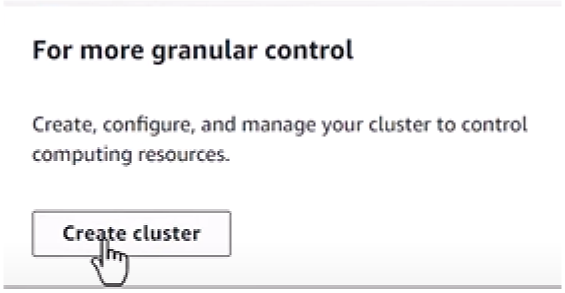
Amazon Redshift

Fast, fully managed, petabyte-scale cloud data warehouse.

Amazon Redshift makes it easier for you to run and scale analytics without having to manage your data warehouse. Get insights by running real-time and predictive analytics on all of your data, across operational databases, data lake, data warehouse, and thousands of third-party datasets.

Get to powerful insights fast

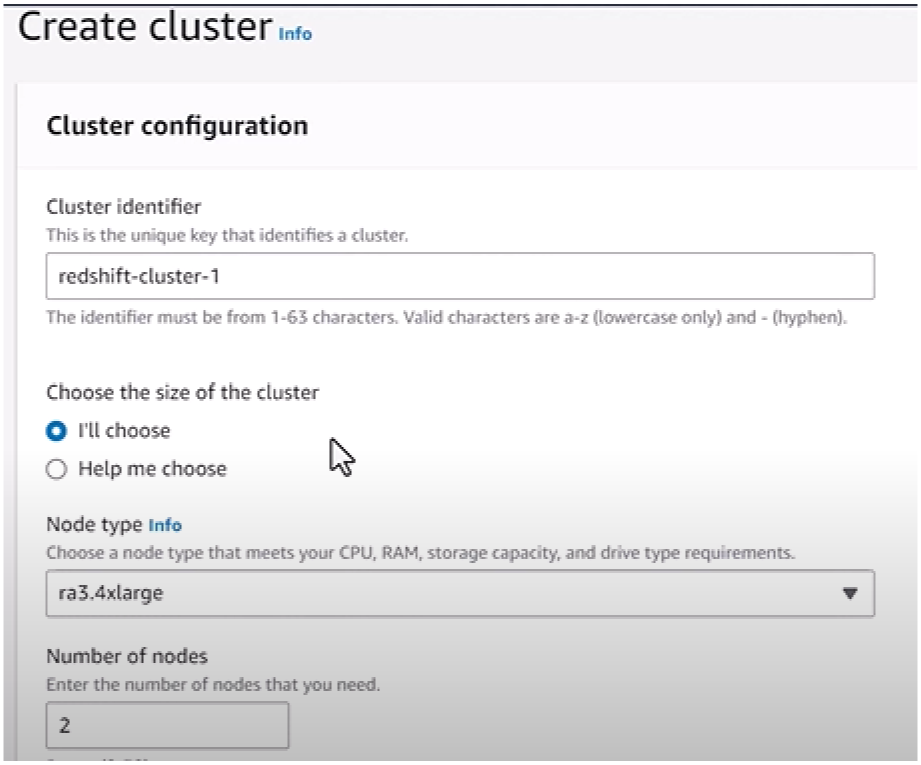
Get insights from data in seconds without managing data warehouse infrastructure. First-time Redshift Serverless customers receive a $300 credit to use in their account.



For more granular control

Create, configure, and manage your cluster to control computing resources.

2. So here we are creating the cluster



Create cluster Info

Cluster configuration

Cluster identifier

This is the unique key that identifies a cluster.

redshift-cluster-1

The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

Choose the size of the cluster

● I'll choose

○ Help me choose

Node type Info

Choose a node type that meets your CPU, RAM, storage capacity, and drive type requirements.

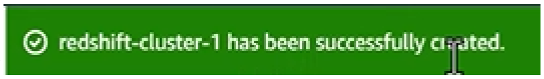
ra3.4xlarge

Number of nodes

Enter the number of nodes that you need.

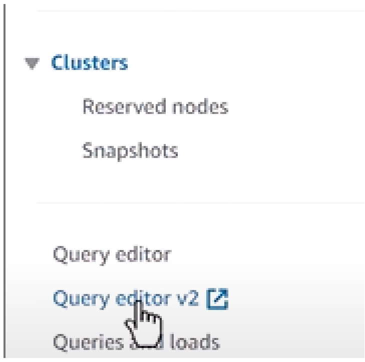
2

3. And we click on create cluster

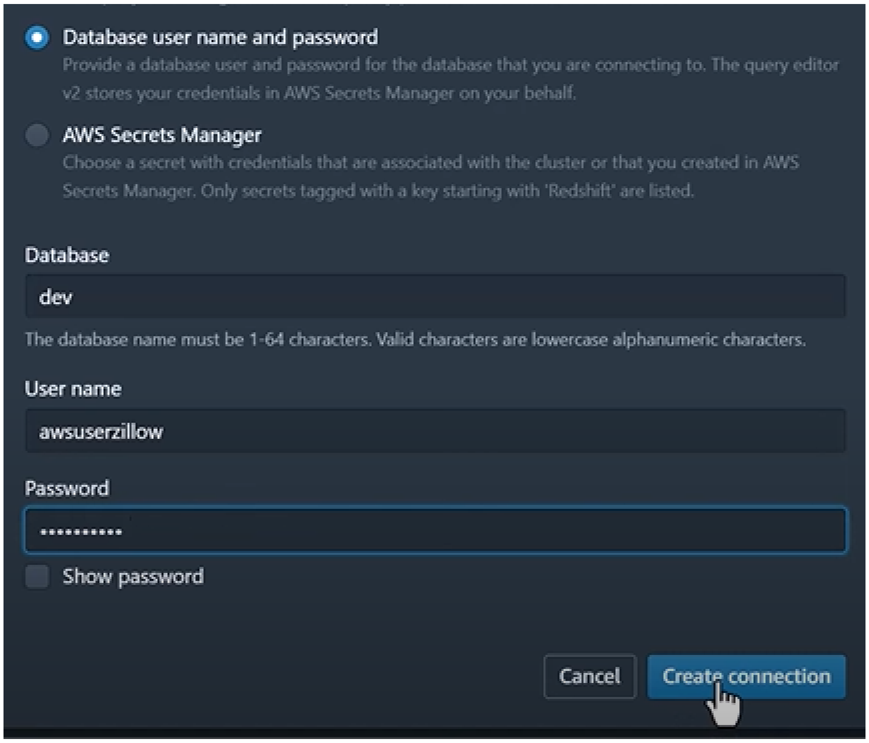


redshift-cluster-1 has been successfully created.

4. Then go to query editor



5. Next we will give username and password and will create connection



● Database user name and password

Provide a database user and password for the database that you are connecting to. The query editor v2 stores your credentials in AWS Secrets Manager on your behalf.

○ AWS Secrets Manager

Choose a secret with credentials that are associated with the cluster or that you created in AWS Secrets Manager. Only secrets tagged with a key starting with 'Redshift' are listed.

Database

dev

The database name must be 1-64 characters. Valid characters are lowercase alphanumeric characters.

User name

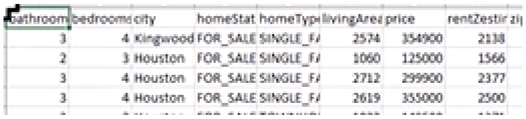
awsuserzillow

Password

••••••••••

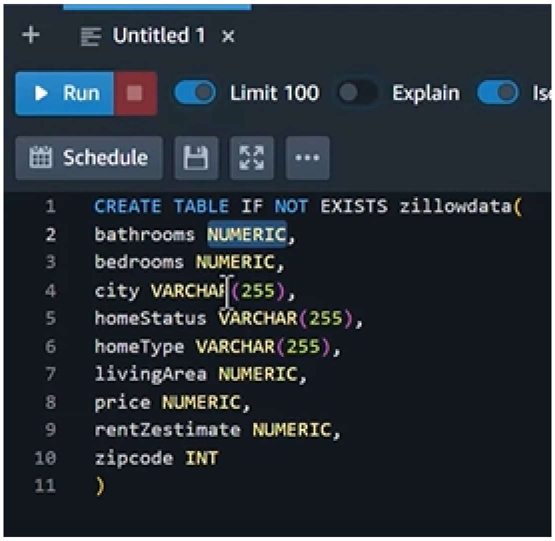
□ Show password

6. Here this is the data which is in cleaned bucket



bathroom	bedrooms	city	homeStat	homeTyp	livingArea	price	rentZestir	zi
3	4	Kingwood	FOR_SALE	SINGLE_FA	2574	354900	2138	
2	3	Houston	FOR_SALE	SINGLE_FA	1060	125000	1566	
3	4	Houston	FOR_SALE	SINGLE_FA	2712	299900	2377	
3	4	Houston	FOR_SALE	SINGLE_FA	2619	355000	2500	

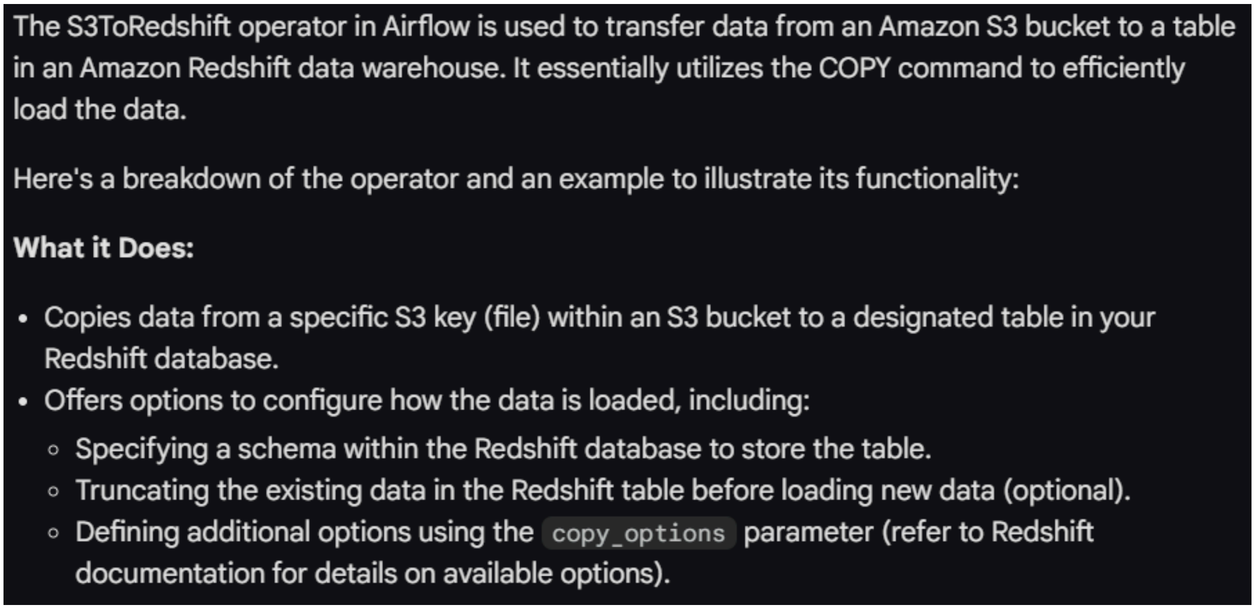
7. So we have to create a table in redshift using this data as reference



```sql
CREATE TABLE IF NOT EXISTS zillowdata(
bathrooms NUMERIC,
bedrooms NUMERIC,
city VARCHAR(255),
homeStatus VARCHAR(255),
homeType VARCHAR(255),
livingArea NUMERIC,
price NUMERIC,
rentZestimate NUMERIC,
zipcode INT
)
```

and we click on run

8. Now we will have a zillowdata table in our tables
9. Next we will be using S3toRedshift operator in airflow to tra

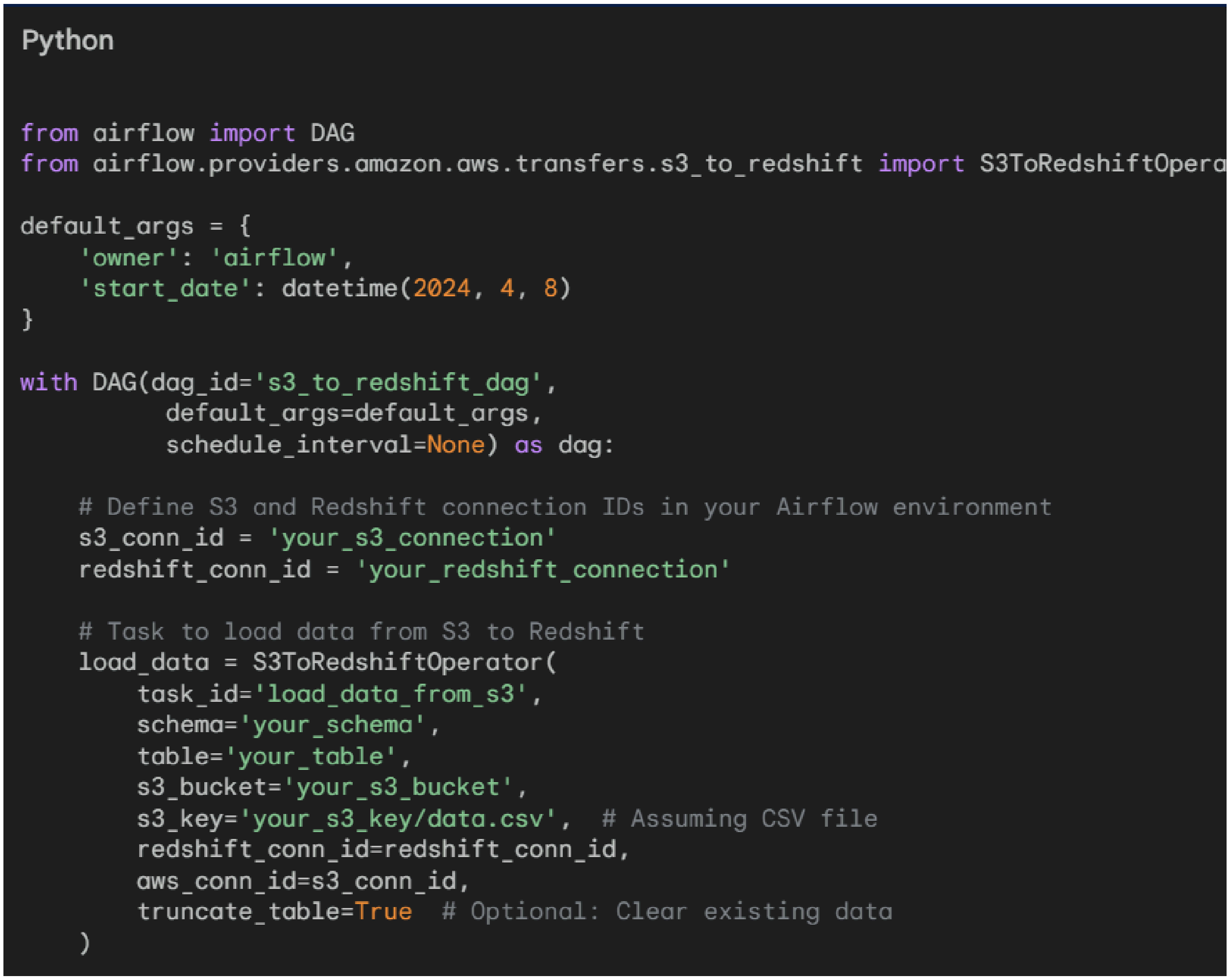


The S3ToRedshift operator in Airflow is used to transfer data from an Amazon S3 bucket to a table in an Amazon Redshift data warehouse. It essentially utilizes the COPY command to efficiently load the data.

Here's a breakdown of the operator and an example to illustrate its functionality:

What it Does:

- Copies data from a specific S3 key (file) within an S3 bucket to a designated table in your Redshift database.
- Offers options to configure how the data is loaded, including:
  - Specifying a schema within the Redshift database to store the table.
  - Truncating the existing data in the Redshift table before loading new data (optional).
  - Defining additional options using the `copy_options` parameter (refer to Redshift documentation for details on available options).



Python

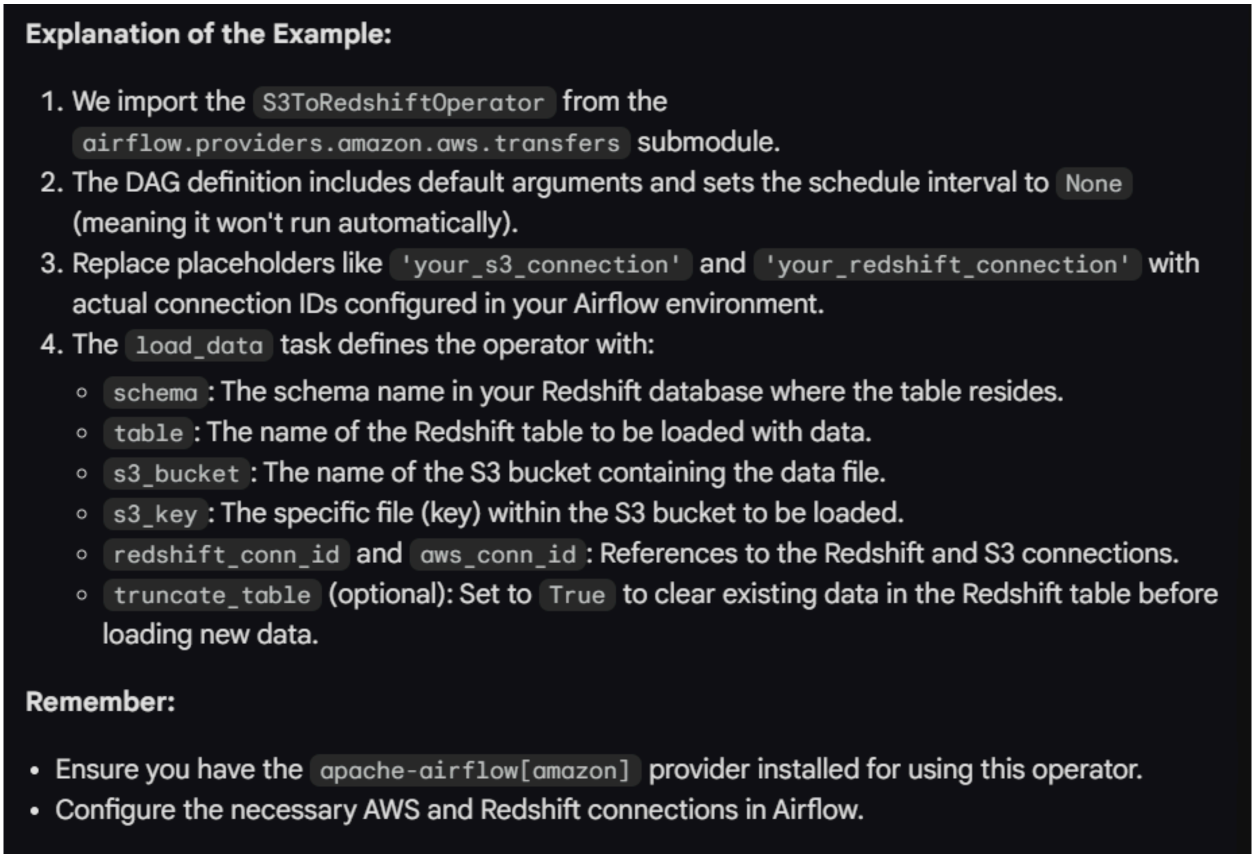
```python
from airflow import DAG
from airflow.providers.amazon.aws.transfers.s3_to_redshift import S3ToRedshiftOpera

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2024, 4, 8)
}

with DAG(dag_id='s3_to_redshift_dag',
          default_args=default_args,
          schedule_interval=None) as dag:

    # Define S3 and Redshift connection IDs in your Airflow environment
    s3_conn_id = 'your_s3_connection'
    redshift_conn_id = 'your_redshift_connection'

    # Task to load data from S3 to Redshift
    load_data = S3ToRedshiftOperator(
        task_id='load_data_from_s3',
        schema='your_schema',
        table='your_table',
        s3_bucket='your_s3_bucket',
        s3_key='your_s3_key/data.csv',  # Assuming CSV file
        redshift_conn_id=redshift_conn_id,
        aws_conn_id=s3_conn_id,
        truncate_table=True  # Optional: Clear existing data
    )
```



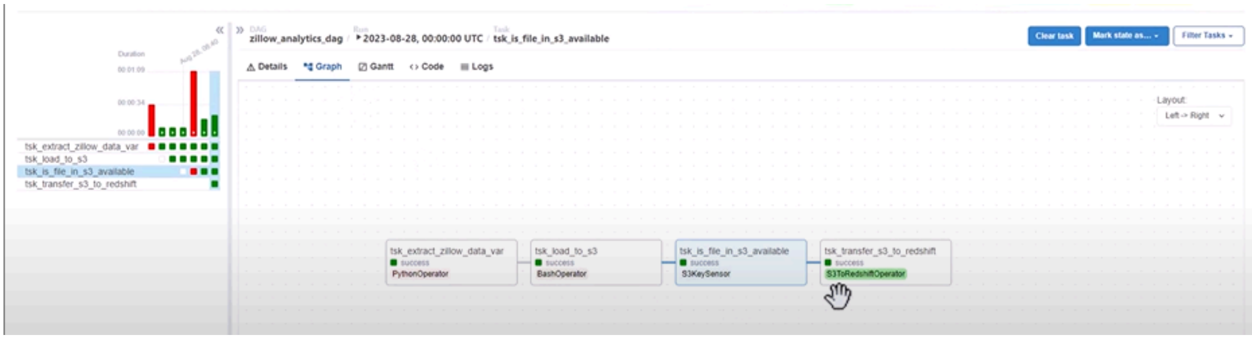
Explanation of the Example:

1. We import the `S3ToRedshiftOperator` from the `airflow.providers.amazon.aws.transfers` submodule.
2. The DAG definition includes default arguments and sets the schedule interval to `None` (meaning it won't run automatically).
3. Replace placeholders like `'your_s3_connection'` and `'your_redshift_connection'` with actual connection IDs configured in your Airflow environment.
4. The `load_data` task defines the operator with:
   - `schema`: The schema name in your Redshift database where the table resides.
   - `table`: The name of the Redshift table to be loaded with data.
   - `s3_bucket`: The name of the S3 bucket containing the data file.
   - `s3_key`: The specific file (key) within the S3 bucket to be loaded.
   - `redshift_conn_id` and `aws_conn_id`: References to the Redshift and S3 connections.
   - `truncate_table` (optional): Set to `True` to clear existing data in the Redshift table before loading new data.

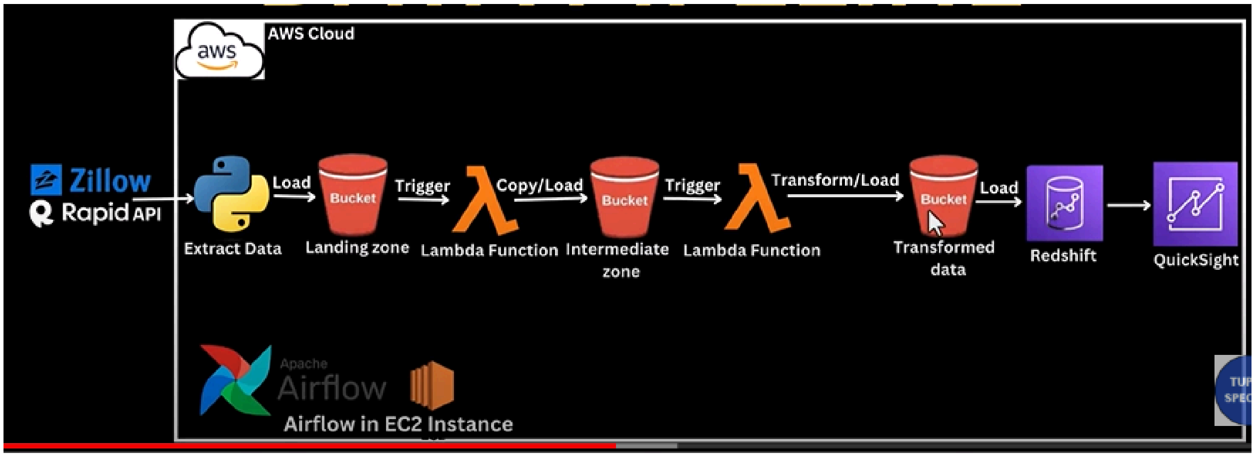
Remember:

- Ensure you have the `apache-airflow[amazon]` provider installed for using this operator.
- Configure the necessary AWS and Redshift connections in Airflow.

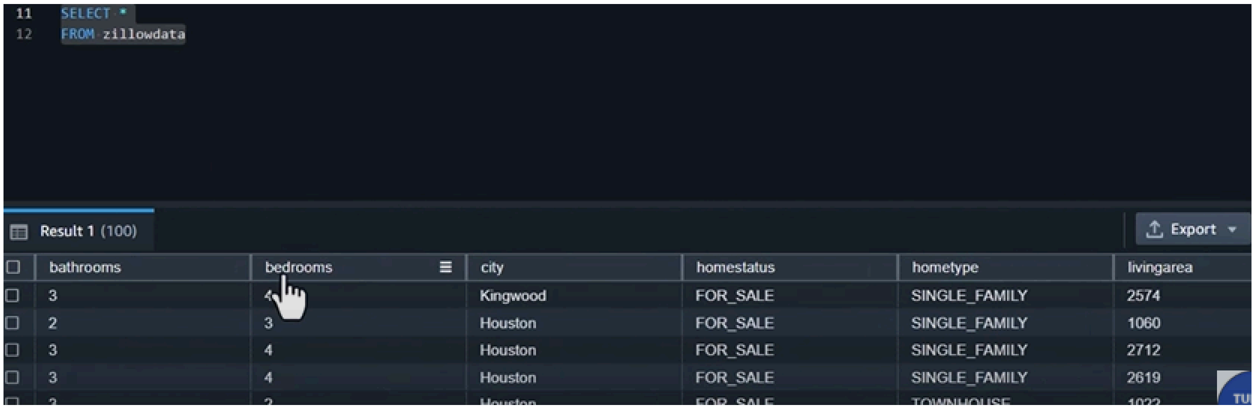
10. Now we have to connect our Airflow to redshift..via endpoints in our cluster
11. Next we have to assign policies to our ec2 instance via roles..to have full access to redshift
12. ALso we have change inbound rules for our redshift cluster
13. Then we will trigger the DAG



14. Now we have implemented our pipeline till redshift



15. Now if we run the table..we can see the data



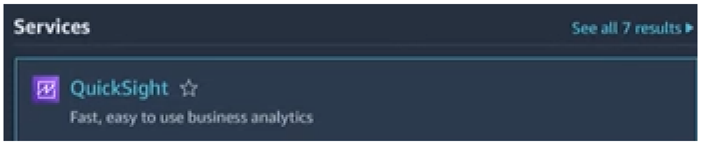
```
SELECT *
FROM zillowdata
```

Result 1 (100)

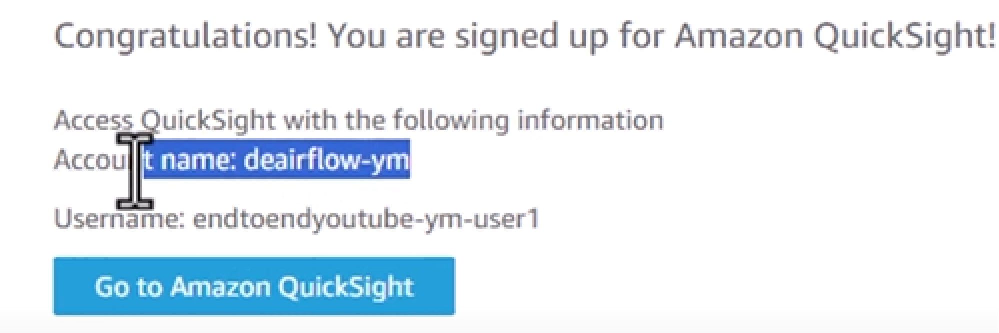
Export

□	bathrooms	bedrooms	city	homestatus	hometype	livingarea
□	3	4	Kingwood	FOR_SALE	SINGLE_FAMILY	2574
□	2	3	Houston	FOR_SALE	SINGLE_FAMILY	1060
□	3	4	Houston	FOR_SALE	SINGLE_FAMILY	2712
□	3	4	Houston	FOR_SALE	SINGLE_FAMILY	2619
□	3	2	Houston	FOR_SALE	TOWNHOUSE	1022

16. Next we are going to use quick sight to analyze our data



17. Next we need to signup for the quicksight and we choose standard version which is free



Congratulations! You are signed up for Amazon QuickSight!

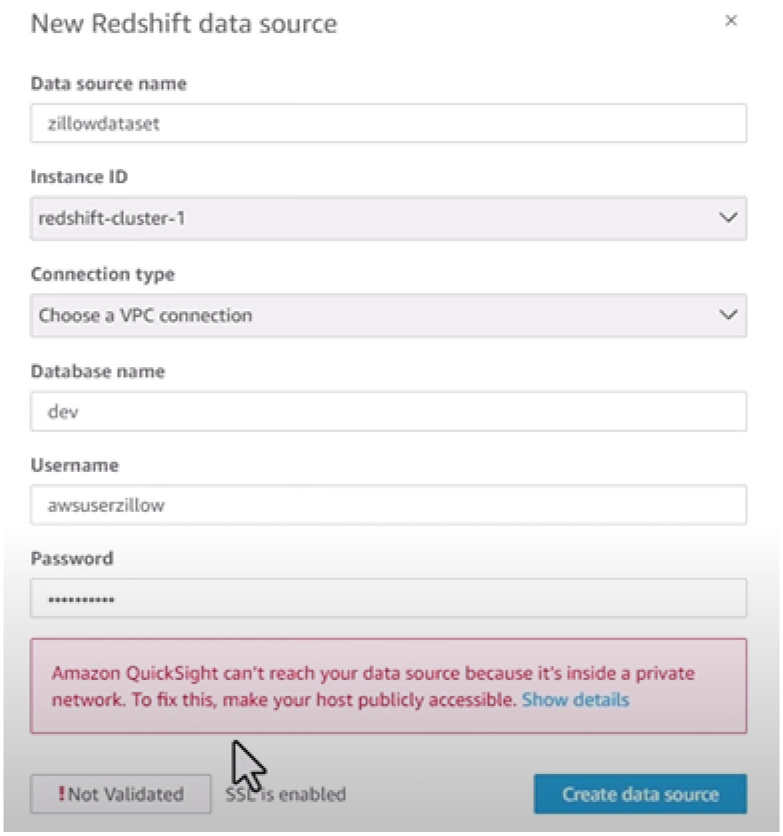
Access QuickSight with the following information

Account name: deairflow-ym

Username: endtoendyoutube-ym-user1

Go to Amazon QuickSight

18. Next we have to give redshif t data source



New Redshift data source

Data source name: zillowdataset

Instance ID: redshift-cluster-1

Connection type: Choose a VPC connection

Database name: dev

Username: awsuserzillow

Password: •••••••••••

Amazon QuickSight can't reach your data source because it's inside a private network. To fix this, make your host publicly accessible. Show details

19. And we'll make our redshift publicly accessible..and then we can access our redshift data in quicksight