

WEATHER AIRFLOW PROJECT

BY :

**TUQA HUSSIEN
ABDALLAH AMR**



Apache
Airflow

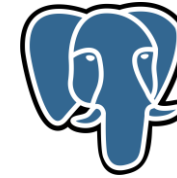
Weather Project Architecture



API



Transformation



RDS Aws Postgres



CSV in S3



RDS AWS Postgres



Joining data view

Orchestrate



Apache
Airflow



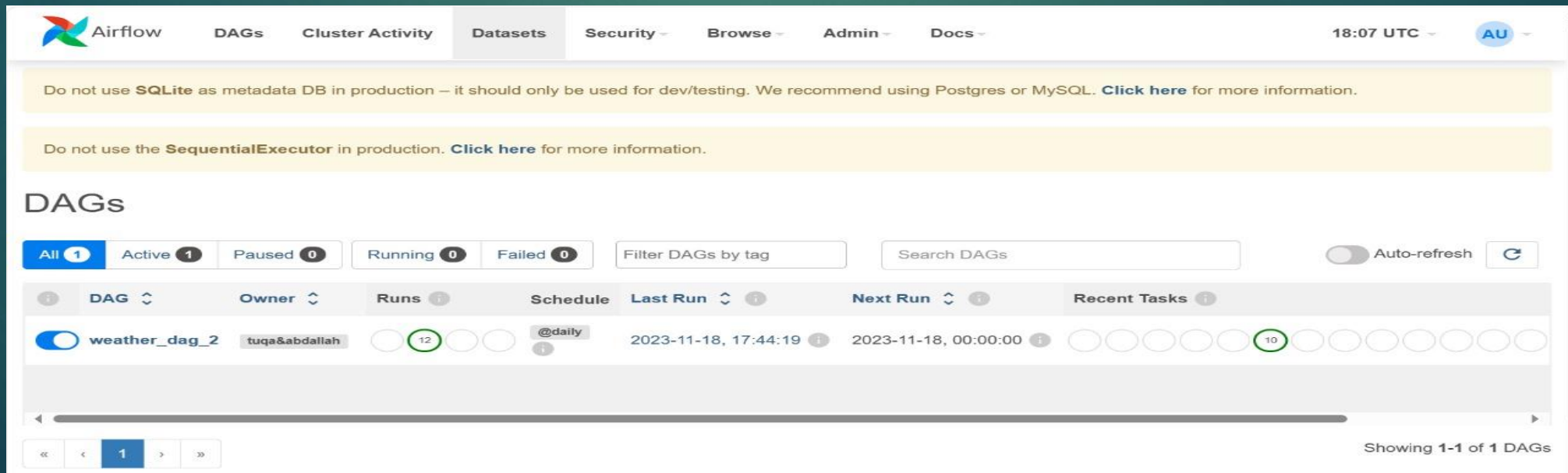
Airflow in EC2
instance

WEATHER AIRFLOW PROJECT

This project automates the retrieval, transformation, and storage of weather data from an API and csv file in s3 into a PostgreSQL database using Apache Airflow Deployed in ec2 instance .

Overview

This contains code to set up an Airflow Directed Acyclic Graph (DAG) named `weather_dag.py`. This DAG orchestrates tasks to:



The screenshot displays the Apache Airflow web interface. At the top, there's a navigation bar with links for DAGs, Cluster Activity, Datasets, Security, Browse, Admin, and Docs. The current time is 18:07 UTC. Below the navigation bar, there are two warning messages: "Do not use SQLite as metadata DB in production" and "Do not use the SequentialExecutor in production". The main section is titled "DAGs" and features a filter bar with buttons for All (1), Active (1), Paused (0), Running (0), and Failed (0). There's also a search bar and an auto-refresh toggle. The table below lists the DAGs, with columns for DAG, Owner, Runs, Schedule, Last Run, Next Run, and Recent Tasks. The only DAG listed is "weather_dag_2" owned by "tuqa&abdallah", with a daily schedule and a last run on 2023-11-18 at 17:44:19. The "Recent Tasks" column shows a sequence of task status icons, with the 10th task being green.

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
weather_dag_2	tuqa&abdallah	12	@daily	2023-11-18, 17:44:19	2023-11-18, 00:00:00	10



Airflow

DAGs

Cluster Activity

Datasets

Security

Browse

Admin

Docs

18:36 UTC

AU

Press **shift** + **/** for Shortcuts

deferred

failed

queued

removed

restarting

running

scheduled

skipped

success

up_for_reschedule

up_for_retry

upstream_failed

no_status

>> DAG

Run

Task

weather_dag_2 / 2023-11-18, 00:00:00 UTC / truncate_table_us_cities

Clear task

Mark state as... ▾

Filter Tasks ▾

△ Details

■ Graph

■ Gantt

<> Code

≡ Logs

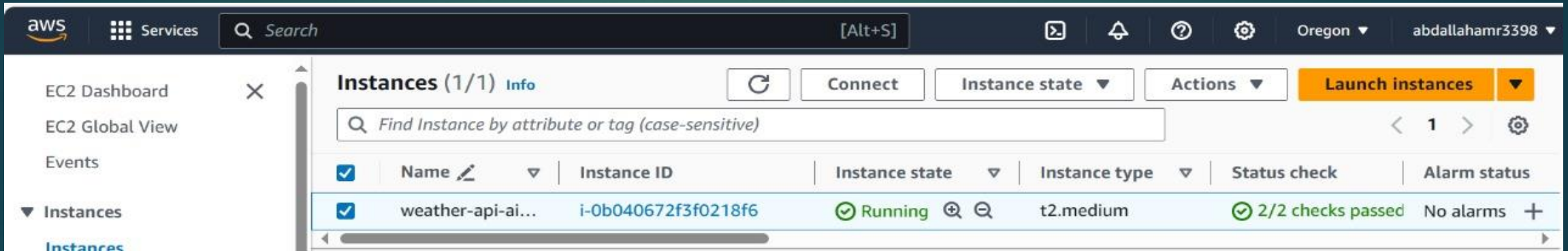
Layout:

Left -> Right ▾



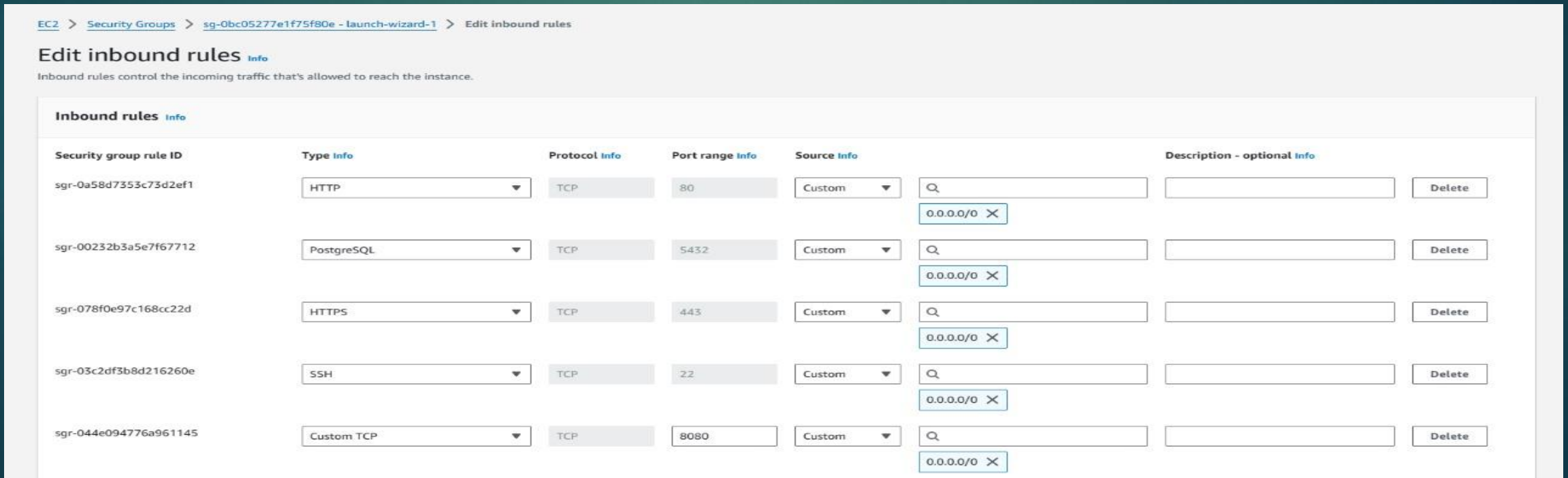
project steps :

1-Create EC2 instance with it's security Group In AWS Cloud and install (Airflow)



The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, 'Services', a search bar, and the user's account information (Oregon, abdallahamr3398). The left sidebar shows the 'Instances' menu. The main content area displays the 'Instances (1/1)' page. A table lists the instance details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
weather-api-ai...	i-0b040672f3f0218f6	Running	t2.medium	2/2 checks passed	No alarms



The screenshot shows the 'Edit inbound rules' page for a security group. The page title is 'Edit inbound rules' and it includes a sub-header 'Inbound rules control the incoming traffic that's allowed to reach the instance.' The page displays a table of inbound rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Action
sgr-0a58d7353c73d2ef1	HTTP	TCP	80	Custom	0.0.0.0/0	Delete
sgr-00232b3a5e7f67712	PostgreSQL	TCP	5432	Custom	0.0.0.0/0	Delete
sgr-078f0e97c168cc22d	HTTPS	TCP	443	Custom	0.0.0.0/0	Delete
sgr-03c2df3b8d216260e	SSH	TCP	22	Custom	0.0.0.0/0	Delete
sgr-044e094776a961145	Custom TCP	TCP	8080	Custom	0.0.0.0/0	Delete

2- Create S3 upload file CSV

Amazon S3

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

Storage Lens groups New

AWS Organizations settings

Objects

Properties

Permissions

Metrics

Management

Access Points

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Refresh

Copy S3 URI

Copy URL

Download

Open

Delete

Actions

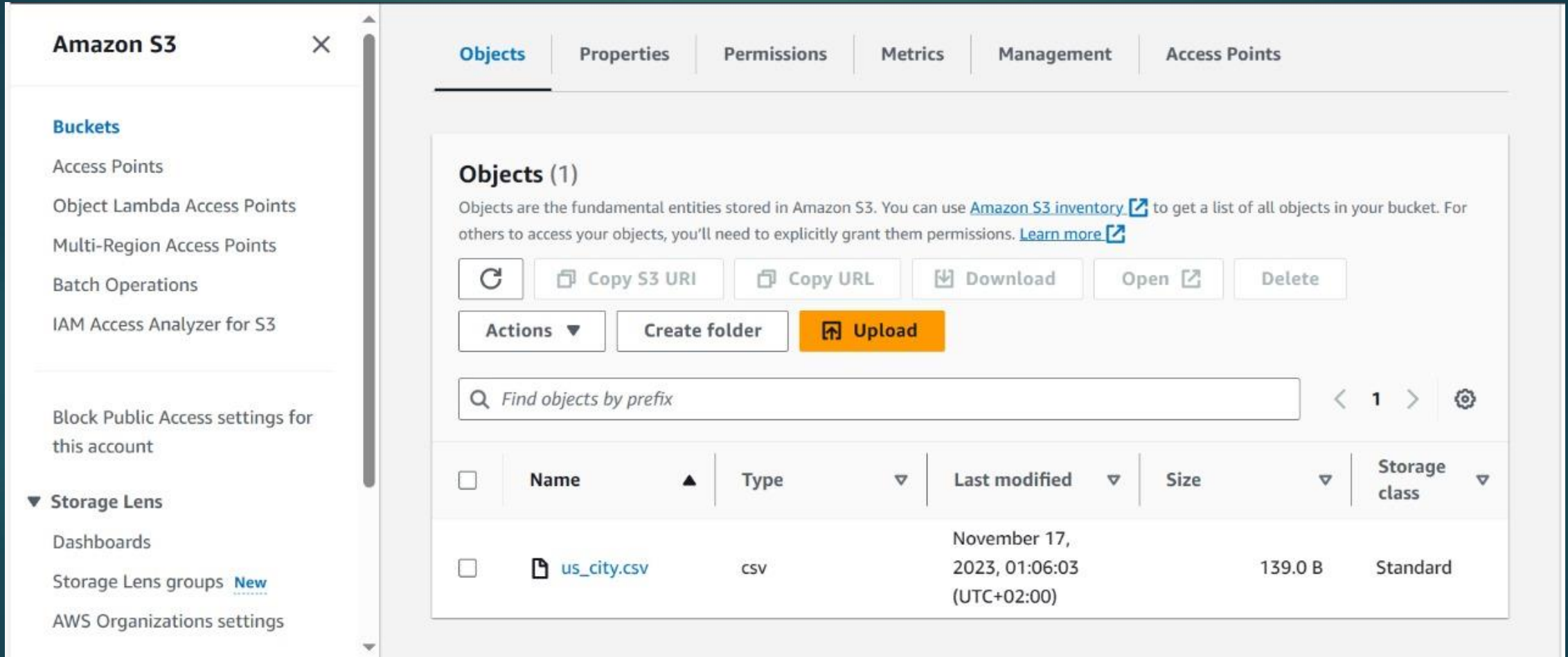
Create folder

Upload


Find objects by prefix

< 1 >

Settings

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 us_city.csv	csv	November 17, 2023, 01:06:03 (UTC+02:00)	139.0 B	Standard

3- Generate from open weather API key



weatherapi

Active



```
https://api.openweathermap.org/data/2.5/weather?q={city  
name}&appid={API key}
```



4-Create RDS Postgres Database

Amazon RDS

Dashboard

Databases

Query Editor

Performance insights

Snapshots

Exports in Amazon S3

Automated backups

Reserved instances

Proxies

Subnet groups

Parameter groups

Option groups

Custom engine versions

Zero-ETL integrations [New](#)

RDS > Databases

Consider creating a Blue/Green Deployment to minimize downtime during upgrades

You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. [RDS User Guide](#) [Aurora User Guide](#)

Databases (1)

Group resources

Modify

Actions

Restore from S3

Create database

Filter by databases

DB identifier	Status	Role	Engine	Region & AZ	Size	Actions	CPU
<div><div></div>rds-db-test</div>	<div><div></div>Available</div>	Instance	PostgreSQL	us-west-2a	db.t3.micro	<div>2 Actions</div>	<div><div></div>4.54</div>

5-Load Api Data into postgres Table

```
ubuntu@ip-172-31-29-41:~$ psql -h rds-db-test.c447ppnpbcqy.us-west-2.rds.amazonaws.com -p 5432 -U postgres -W
Password:
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.7)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.

postgres=> select * from weather_data;
```

city	description	temperature_fahrenheit	feels_like_fahrenheit	minimun_temp_fahrenheit	maximum_temp_fahrenheit	pressure	humidity	wind_speed	time_of_record	sunrise_local_time	sunset_local_time
Houston	scattered clouds	72.59	72.752	69.296	76.244	1016	68	2.68	2023-11-18 12:29:12	2023-11-18 06:48:35	2023-11-18 17:24:53

(1 row)

(END)

6-Load csv file into postgres Table

```
postgres=> select * from city_look_up;
```

city	state	census_2023	land_area_sq_mile_2023
Chicago	Illinois	2746388	227.4
Seattle	Washington	737015	83.8
Houston	Texas	2304580	640.4

(3 rows)

7-select from joining_data view

```
ubuntu@ip-172-31-29-41:~$ psql -h rds-db-test.c447ppnpbcqy.us-west-2.rds.amazonaws.com -p 5432 -U postgres -W
Password:
psql (16.1 (Ubuntu 16.1-1.pgdg22.04+1), server 14.7)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, compression: off)
Type "help" for help.
```

```
postgres=> \dt
          List of relations
 Schema |      Name      | Type  | Owner
-----+-----+-----+-----
 public | city_look_up   | table | postgres
 public | weather_data   | table | postgres
(2 rows)
```

```
postgres=> select * from joining_data;
```

city	description	temperature_fahrenheit	feels_like_fahrenheit	minimun_temp_fahrenheit	maximum_temp_fahrenheit	pressure	humidity	wind_speed	time_of_record	sunrise_local_time	sunset_local_time	state	census_2023	land_area_sq_mile_2023
Houston	scattered clouds	72.59	72.752	69.296	76.244	1016	68	2.68	2023-11-18 12:29:12	2023-11-18 06:48:35	2023-11-18 17:24:53	Texas	2304580	640.4

(1 row)

```
(END)
```

Thank You