**Documentation**

Directory structure:

code-challenge-template\answers comprises of these folders and files

a. code-challenge-template\answers\logs

b. code-challenge-template\answers\static

c. code-challenge-template\answers\table\_generation\_queries

d. code-challenge-template\answers\documentation\_of\_work

e. code-challenge-template\answers\tests

f. app.py

g. data\_ingestion\_pipeline.py

a. code-challenge-template\answers\logs

Directory captures the log file for data\_ingestion\_pipeline.py script run

b. code-challenge-template\answers\static

Directory has the swagger.yml file which captures the documentation for developed APIs

c. code-challenge-template\answers\table\_generation\_queries

Directory has the sql queries to create data tables for the given tasks.

weather\_data\_table.sql -- SQL query to create table and its attributes type.

weather\_data\_stats.sql -- SQL query to create table for weather stats and its attributes type.

aggregation\_query.sql -- SQL query to capture the aggregation mentioned in the question.

d. code-challenge-template\answers\documentation\_of\_work

Directory has the documentation doc.

e. code-challenge-template\answers\tests

Directory consists of test scripts for API testing.

f. code-challenge-template\answers\app.py

Flask backend code with two developed APIs which are documented in swagger at location code-challenge-template\answers\static\swagger.yml

g. code-challenge-template\answers\data\_ingestion\_pipeline.py

Python script which feeds on the input source files and ingests the data to database and automate the process. This script is neatly commented.

**Problem 1 - Data Modeling:**

**CREATE TABLE weather\_data(**

**id integer PRIMARY KEY,**

**time\_date date,**

**maximum\_temperature float,**

**minimum\_temperature float,**

**precipitation float,**

**location VARCHAR(100),**

**);**

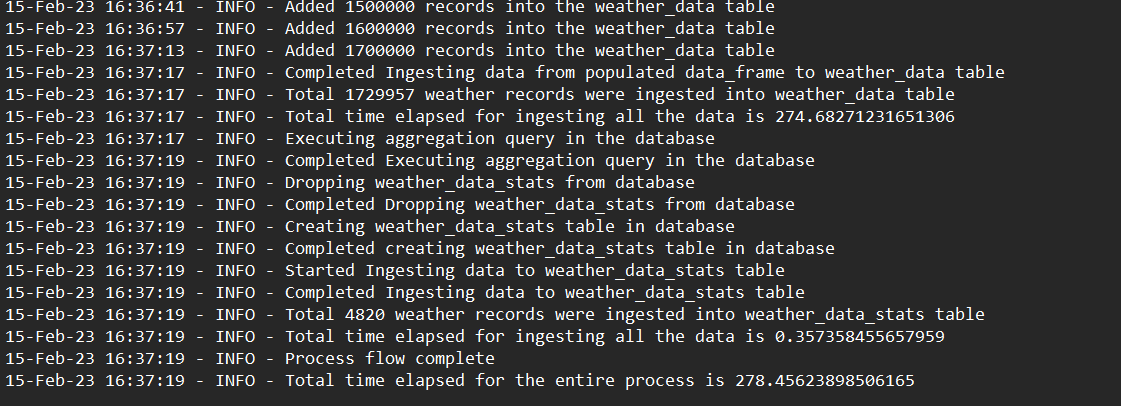
**Problem 2 – Ingestion:**

code-challenge-template\answers\data\_ingestion\_pipeline.py

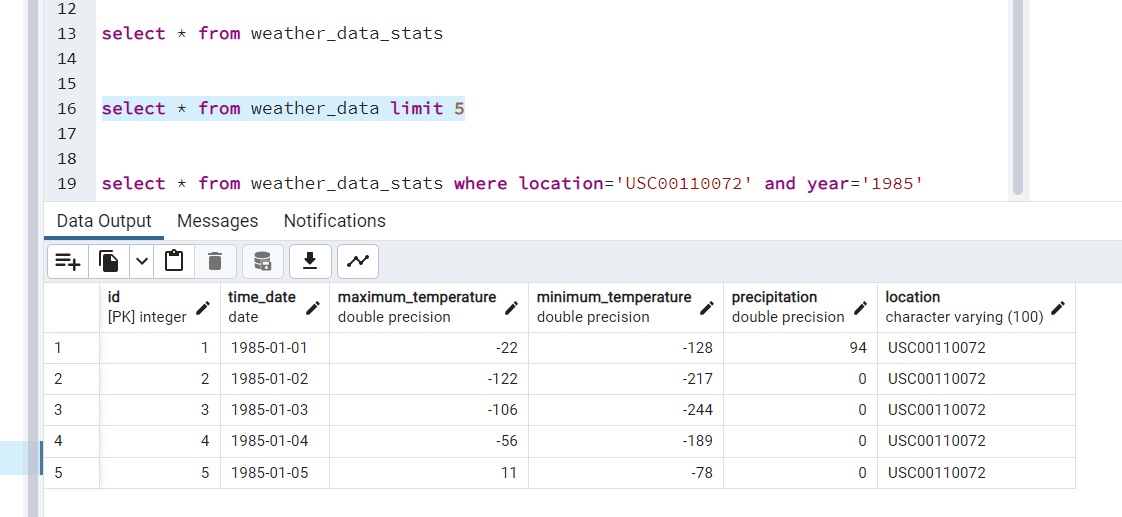
Python script which feeds on the input source files and ingests the data to database and automate the process. This script is neatly commented.

Logs are captured as file under logs folder

**code-challenge-template\answers\logs under file data\_ingestion\_pipeline.log**

****

**Data seen in postgres pgadmin:**

****

**Problem – 3 :**

**Stats Table schema:**

**CREATE TABLE weather\_data\_stats(**

**id integer PRIMARY KEY,**

**year int,**

**location VARCHAR(100),**

**avg\_max\_temp\_celsius VARCHAR(50),**

**avg\_min\_temp\_celsius VARCHAR(50),**

**total\_precipitation\_cm VARCHAR(50)**

**);**

**Aggregation query :**

**SELECT**

**EXTRACT(YEAR from time\_date) AS year,**

**location,**

**AVG(maximum\_temperature) AS avg\_max\_temp\_celsius,**

**AVG(minimum\_temperature) AS avg\_min\_temp\_celsius,**

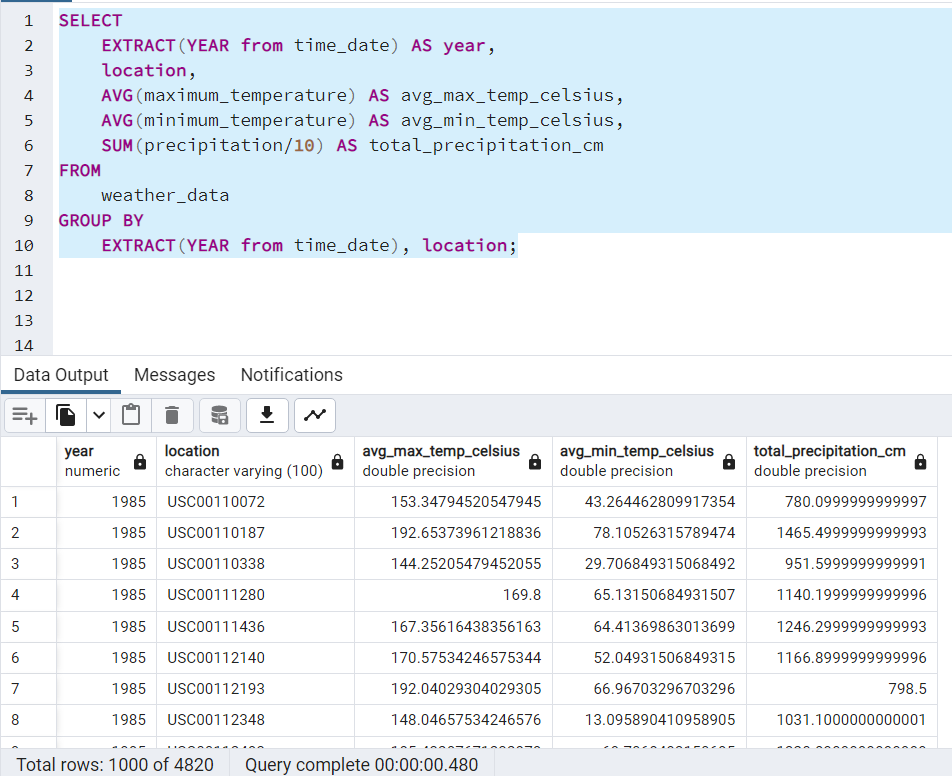
**SUM(precipitation/10) AS total\_precipitation\_cm**

**FROM**

**weather\_data**

**GROUP BY**

**EXTRACT(YEAR from time\_date), location;**

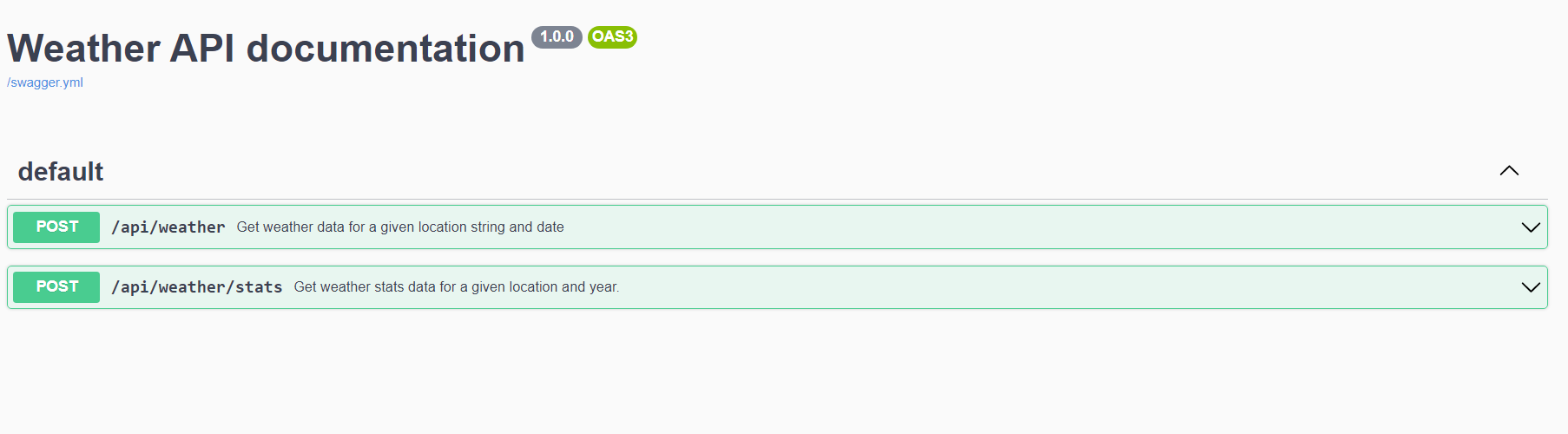
****

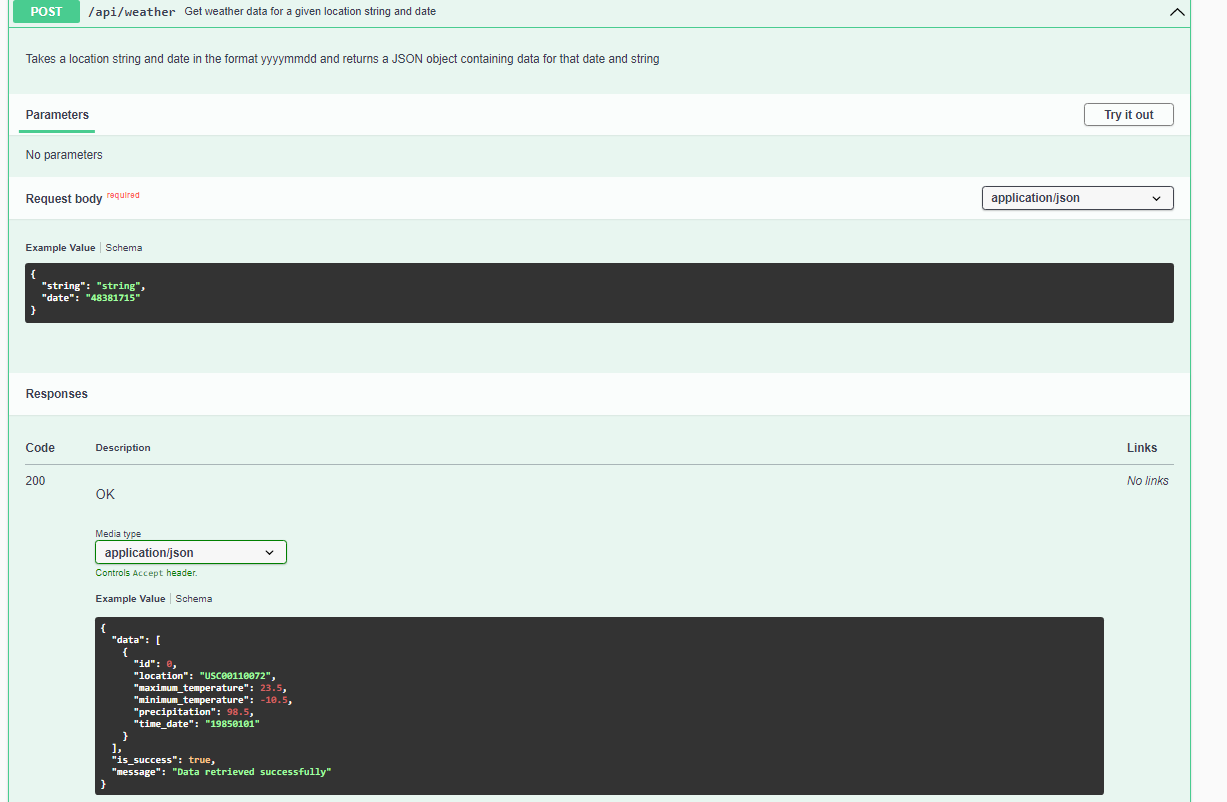
**Problem 4 - REST API:**

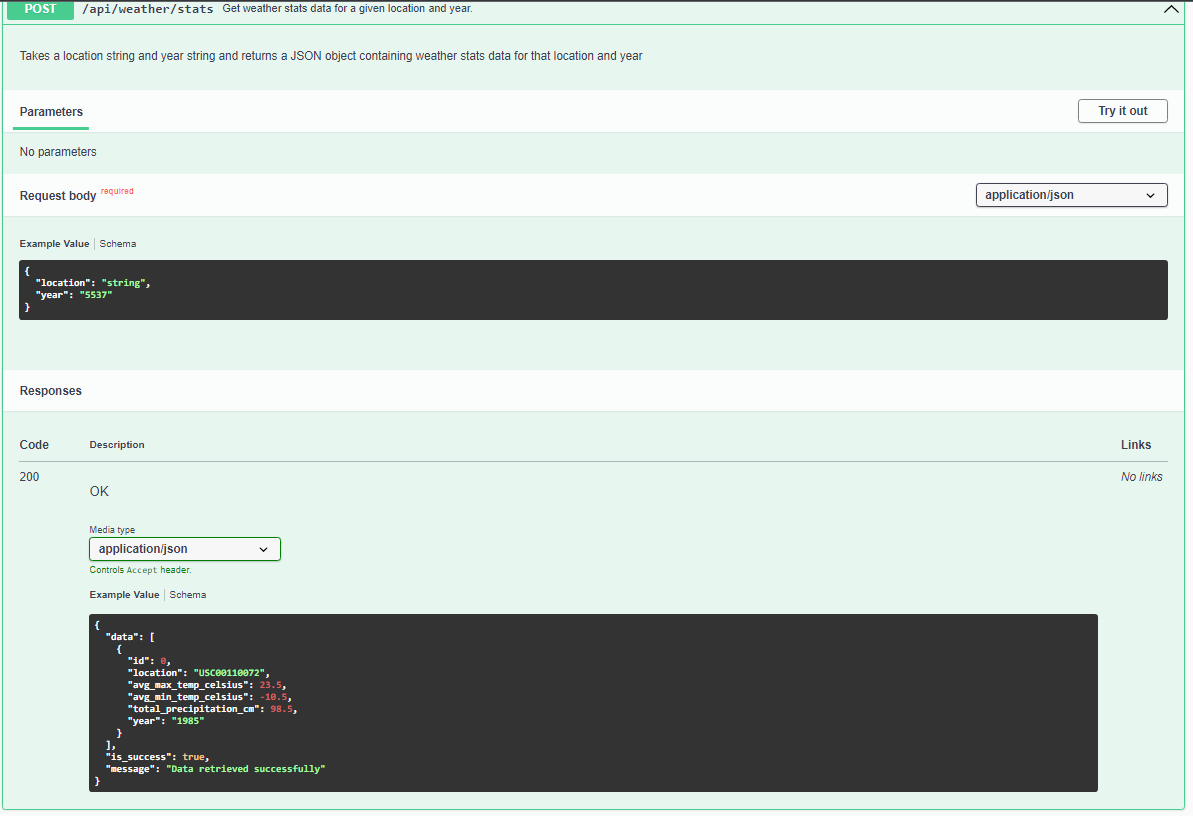
code-challenge-template\answers\app.py

Flask backend code with two developed APIs which are documented in swagger at location code-challenge-template\answers\static\swagger.yml

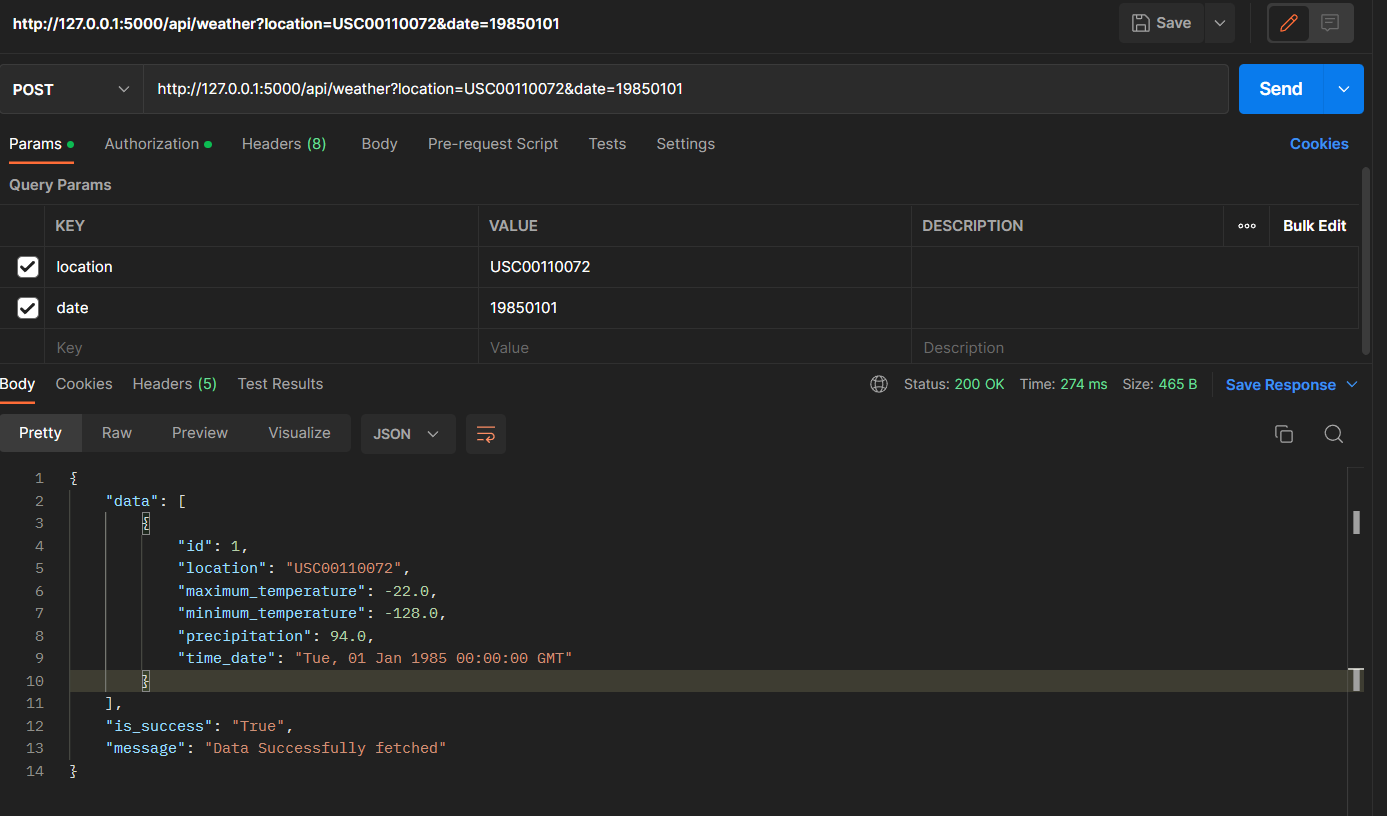
**Swagger API Documentation:**

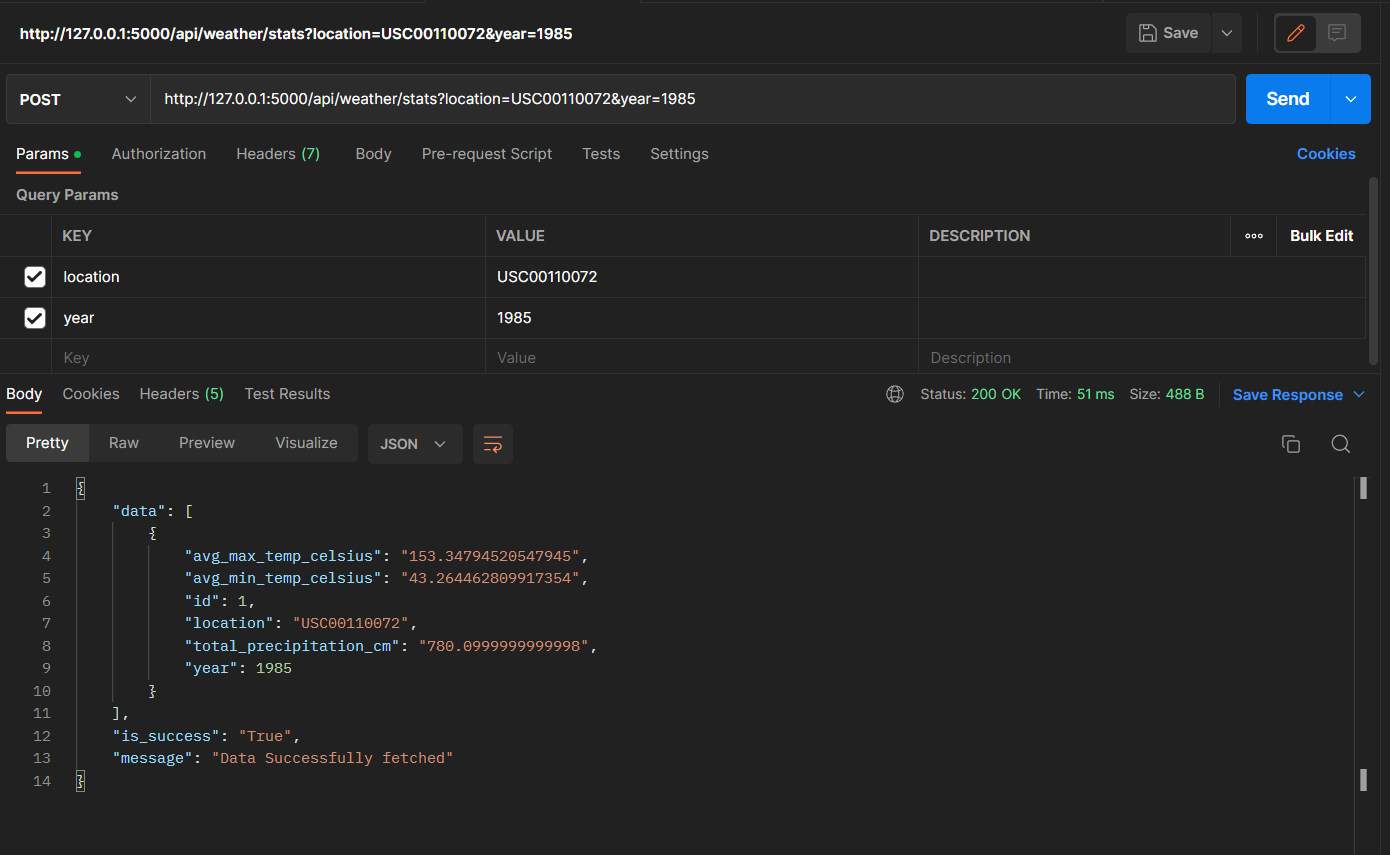
****

****

****

**Postman API Testing:**

****

****

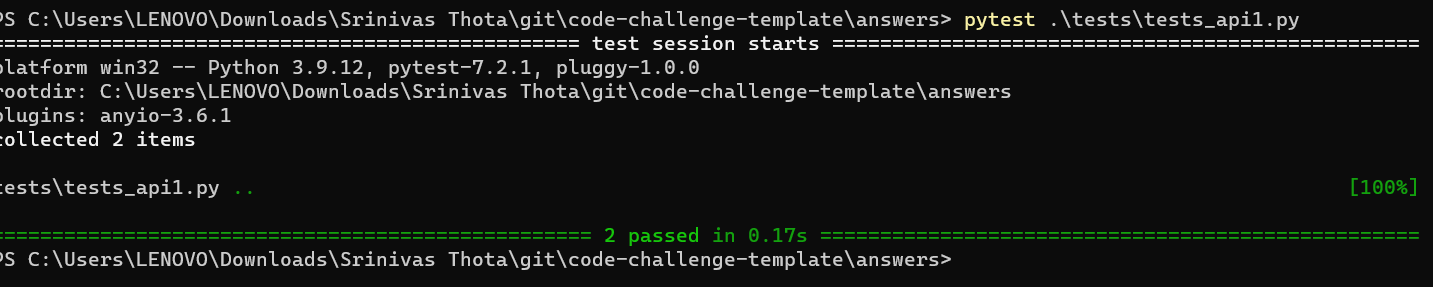
**Optional Question:**

Assume you are asked to get your code running in the cloud using AWS. What tools and AWS services would you use to deploy

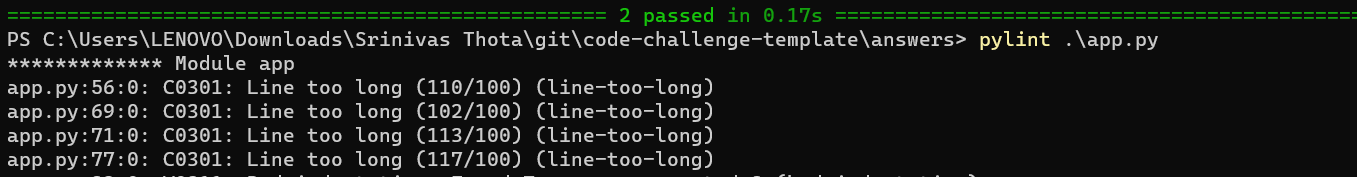
the API, database, and a scheduled version of your data ingestion code? Write up a description of your approach.

1. Firstly I will be storing the data in S3 bucket.
2. I will be using AWS lambda and AWS cloudwatch services events to schedule data ingestion code.
3. We can write the function in AWS lambda using python and trigger the flow to ingest the data.
4. I will be using Amazon RDS service to create a database, I will be creating postgresql database for this.
5. I can deploy the developed python APIs over AWS EC2 instance and fetch the data based on the input parameters. I can also use AWS API gateway for this.
6. For CI/CD I will use AWS Codepipeline for automating the deployment of the application.

**Unit Testing:**

****

**Pylint on the developed code:**

****

**Runs:**

**To ingest code:**

**python code-challenge-template/answers/data\_ingestion\_pipeline.py**

**To start Flask application and APIs:**

**python code-challenge-template/answers/app.py**