**Situación hipotética:**

Actualmente, hemos lanzado un proyecto piloto en el que debemos monitorear la temperatura de tres grandes ciudades (London, New York & Mexico City) cada hora. Dado que se busca expandir este proyecto a más ciudades en el futuro, es fundamental que la solución de gestión de datos sea escalable. Además, es necesario que los datos históricos generados estén disponibles para el equipo de análisis.

**Elección de herramientas:**

Se usará Python para la extracción, limpieza y carga de datos, dicha carga se hará en una base de datos local PostgreSQL.

Con los siguientes permisos:

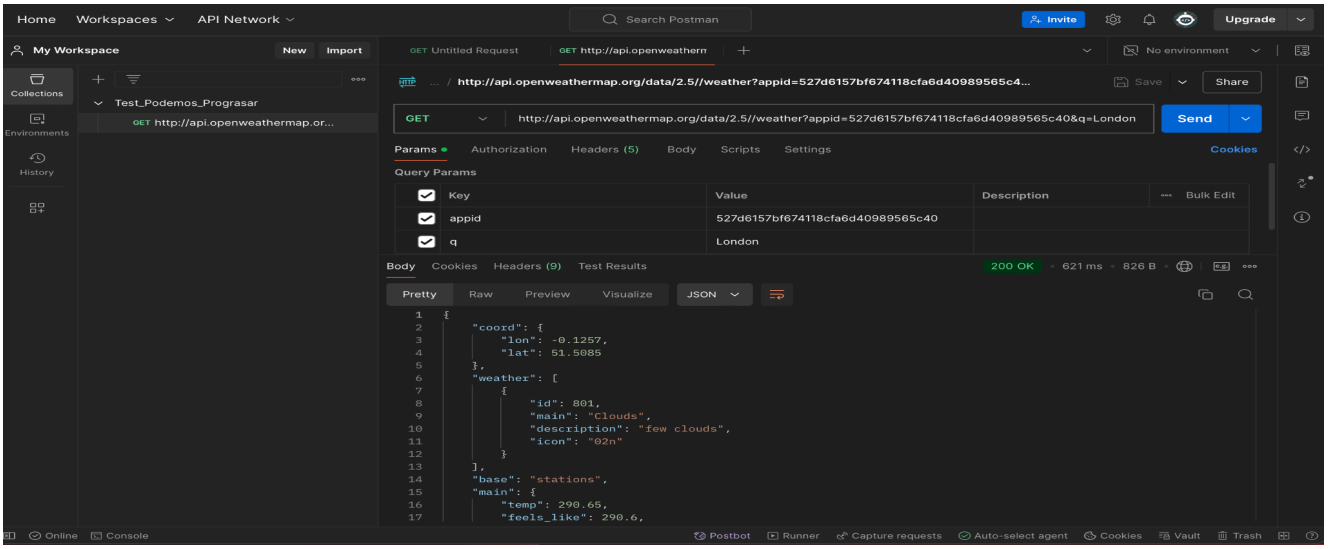
- host: localhost
- database: podemosprograsar
- user : postgres
- password: 123456

**Pruebas de Conexión a la API: OpenWeatherMap API**

Para realizar las siguientes pruebas se utilizó Postman.

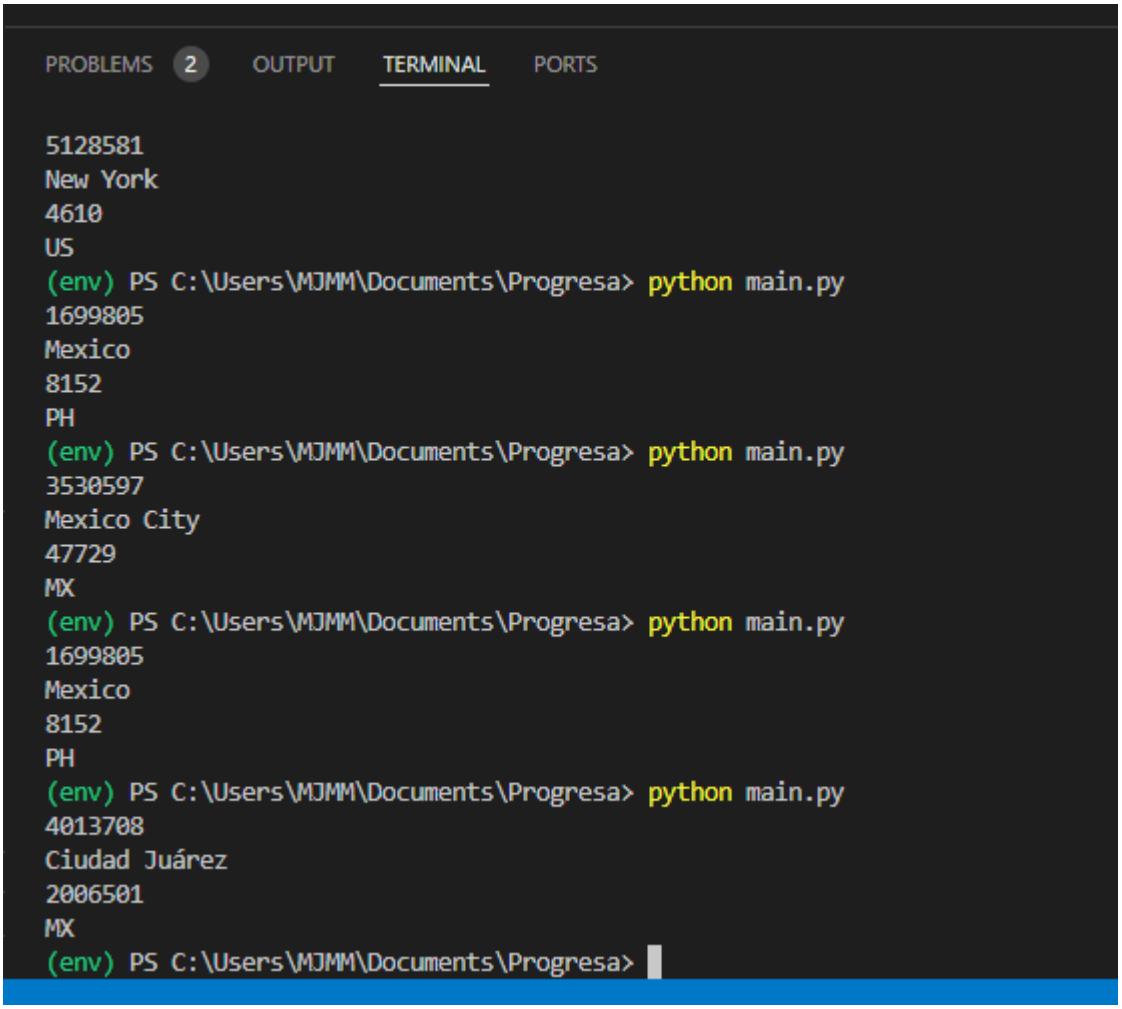
- Se creó una cuenta en la API **OpenWeatherMap API**
- Se generó el token en la página**.**


- Se realizaron las pruebas de conexión:




**Primera iteración:**

Se tomó la decisión de tener un id personal para las dimensiones, ya que se revisaron algunos ejemplo de Países y el Id no coincide.



En la siguiente iteración se recomienda tener un servicio de manejo de credenciales como AWS Secrets Manager o un archivo .env para que las credenciales no se encuentren en código directo y se pueda tener control de ellas.

En lugar de almacenar los archivos de forma local se recomienda hacerlo a un servicio de almacenamiento en la nube como S3.

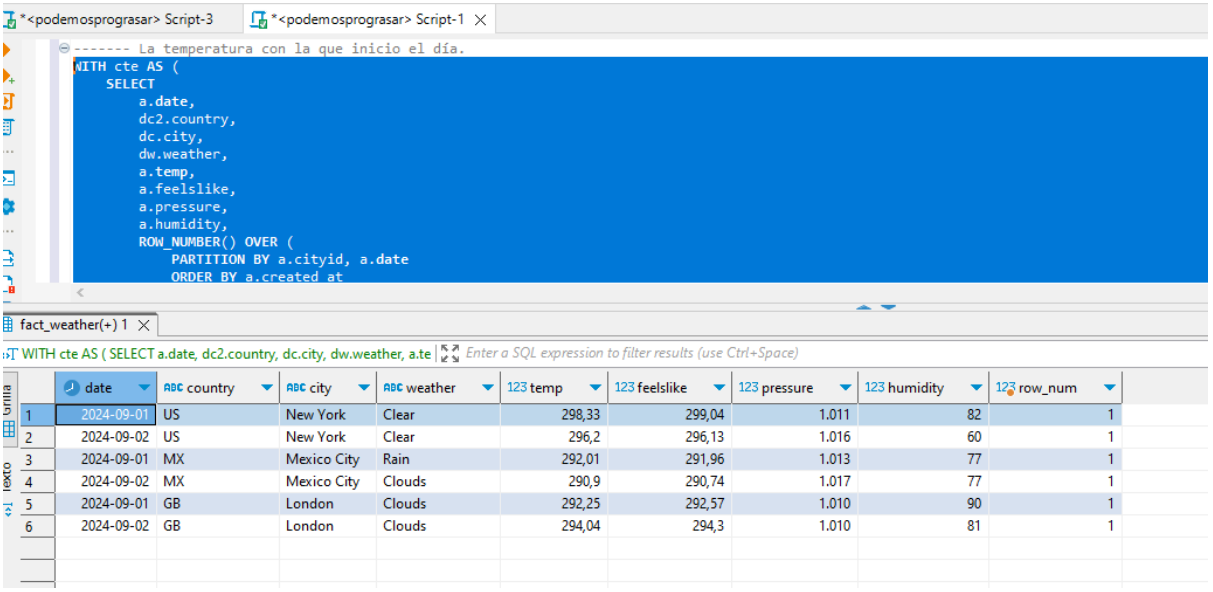
Para esta iteración y por el tiempo limitado que se tenía para realizar la prueba se utilizaron librerías ya conocidas, sin embargo en una segunda iteración que podría ser la búsqueda de más endpoints y con ello cargas históricas, se recomendaría utilizar herramientas como spark, debido a que el procesamiento que se va a requerir será mayor.

En la siguiente iteración se buscaría conectar con Airflow,y calendarizando cada hora

schedule_interval='@hourly'

**Adicionales:**

El modelo entienda relación se encuentra en la carpeta sql.
También podemos encontrar un par de consultas para análisis de datos en la carpeta.




| | date | country | city | weather | temp | feelslike | pressure | humidity | row_num |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2024-09-01 | US | New York | Clear | 298,33 | 299,04 | 1.011 | 82 | 1 |
| 2 | 2024-09-02 | US | New York | Clear | 296,2 | 296,13 | 1.016 | 60 | 1 |
| 3 | 2024-09-01 | MX | Mexico City | Rain | 292,01 | 291,96 | 1.013 | 77 | 1 |
| 4 | 2024-09-02 | MX | Mexico City | Clouds | 290,9 | 290,74 | 1.017 | 77 | 1 |
| 5 | 2024-09-01 | GB | London | Clouds | 292,25 | 292,57 | 1.010 | 90 | 1 |
| 6 | 2024-09-02 | GB | London | Clouds | 294,04 | 294,3 | 1.010 | 81 | 1 |

```sql
-------- Temperatura maxima, minima y promedio por día y ciudad

SELECT
    a.date,
    dc2.country,
    dc.city,
    AVG(a.temp) AS avg_temp,
    MIN(a.temp) AS min_temp,
    MAX(a.temp) AS max_temp
FROM fact_weather a
INNER JOIN dim_weather dw ON a.weatherid = dw.weatherid
INNER JOIN dim_city dc ON dc.cityid = a.cityid
INNER JOIN dim_country dc2 ON dc2.countryid = dc.countryid
```

_weather(+) 1

.ECT a.date, dc2.country, dc.city, AVG(a.temp) AS avg_temp, MI | Enter a SQL expression to filter results (use Ctrl+Space)

| date | country | city | avg_temp | min_temp | max_temp |
|---|---|---|---|---|---|
| 2024-09-01 | MX | Mexico City | 291,1775 | 288,68 | 292,01 |
| 2024-09-02 | MX | Mexico City | 290,9 | 290,9 | 290,9 |
| 2024-09-02 | GB | London | 294,04 | 294,04 | 294,04 |
| 2024-09-01 | GB | London | 291,98125 | 291,86 | 292,25 |
| 2024-09-02 | US | New York | 296,2 | 296,2 | 296,2 |
| 2024-09-01 | US | New York | 298,0475 | 297,83 | 298,33 |