



"El saber de mis hijos
hará mi grandeza"

**UNIVERSIDAD DE SONORA
DIVISION DE CIENCIAS EXACTAS Y
NATURALES
MAESTRIA EN CIENCIA DE DATOS**



Proyecto Final de Introducción a la Ciencia de Datos y sus Metodologías.

Creación de base de datos y sus estructuras.

Doctor Juan Pablo Soto Barrera

Alumna: Reyna Yanet Hernández Mada.

Hermosillo, Sonora 11/25/2022

Contenido

Descripción.....	3
Objetivos	3
Fuente de Datos	4
Desarrollo.....	4
Manejo de datos por pedio de PostgreSQL.	5
Creación de una Vista:	6
Creación de una Función:.....	7
Create Procedure	8
Usando PostgreSQL desde Jupyter notebook.....	9
Análisis de los datos en Jupyter Notebook.	10

Proyecto Final de Introducción a la Ciencia de Datos y sus Metodologías.

Descripción

Dentro de la ciencia de datos, actualmente es muy importante explorar manejadores de bases de datos ya que esta es una herramienta básica para acceso y manipulación de datos.

Esta herramienta es importante no solamente para facilitar la manipulación de datos en las etapas iniciales de proyectos, sino también para asegurar mantenimiento y sustentabilidad de los proyectos en mediano y largo plazo.

El presente proyecto describe el uso de PostgreSQL y PG Admin para crear la base de datos que nos ayude a analizar la información de nuestro proyecto de una manera sencilla y ordenada, importar archivos, crear tablas, así como vistas y procedimientos que nos faciliten el manejo del análisis de la información desde Python.

Objetivos

Crear una base de datos que contenga una vista, procedimiento almacenado y una función.

Exportar parte de la información para utilizarse en una libreta de Jupyter.

Efectuar una consulta desde libreta de Jupyter

Fuente de Datos

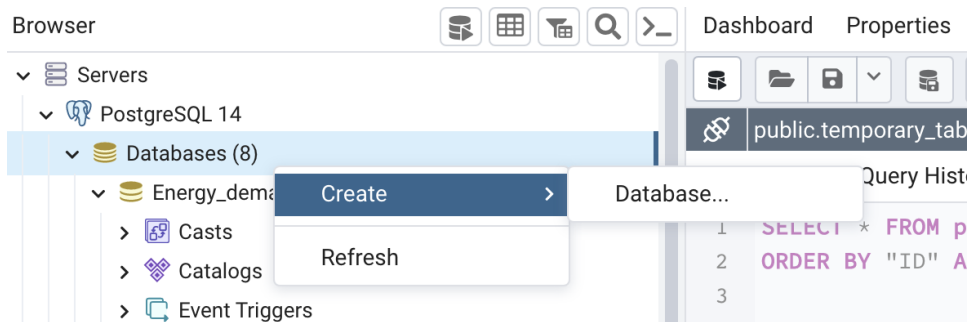
Los archivos ingresados en la base de datos se componen de dos fuentes principales:

- Archivo "energydemands2022" obtenido de Cenace, el cual contiene la demanda eléctrica de la región desde 2007 a octubre de 2022.
- Archivos generados por Meteomatics, concentrados en un archivo (complete_formated.csv) donde se incluyen variables de temperatura de diferentes ciudades de la región de la zona de carga de la Demanda de Energía.

Desarrollo

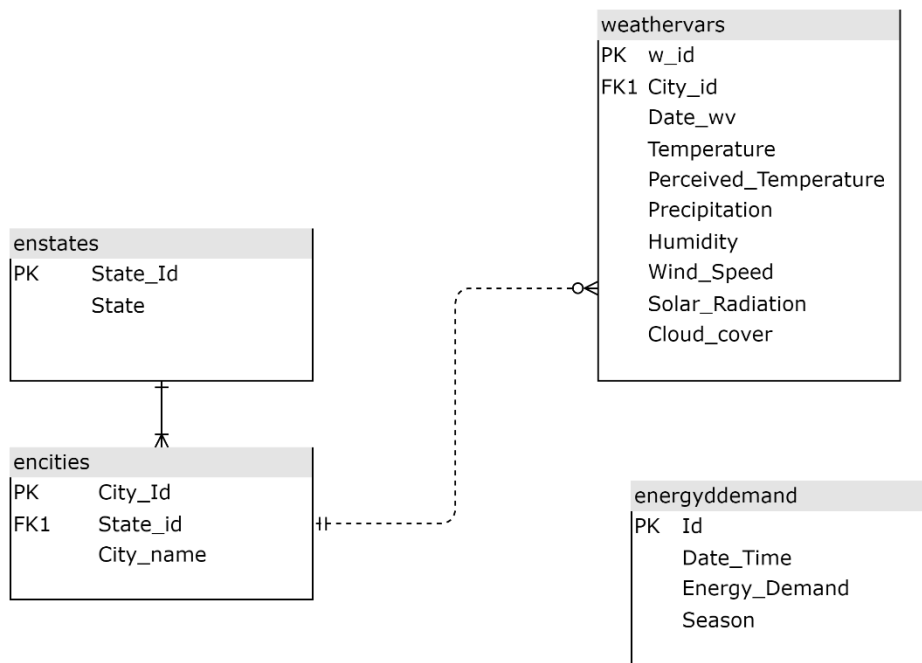
La información contenida en el presente proyecto es complementaria de la materia de Ingeniería de características. Originalmente, las bases de datos se manejaban solamente con las libretas de Jupyter (Python) Sin embargo, se encontraron algunos retos en la unión de dataframes en formato DateTime, por lo que se toma la decisión de utilizar SQL para facilitar el manejo de los archivos.

Manejo de datos por pedio de PostgreSQL



Los archivos CSV se habían ajustado previamente desde Python, por lo que se realiza el siguiente diagrama para identificar las relaciones entre tablas, así como asignar 2 tablas relacionales para ciudades y estados, de esta manera se facilitará la actualización de datos en el futuro.

Diagrama

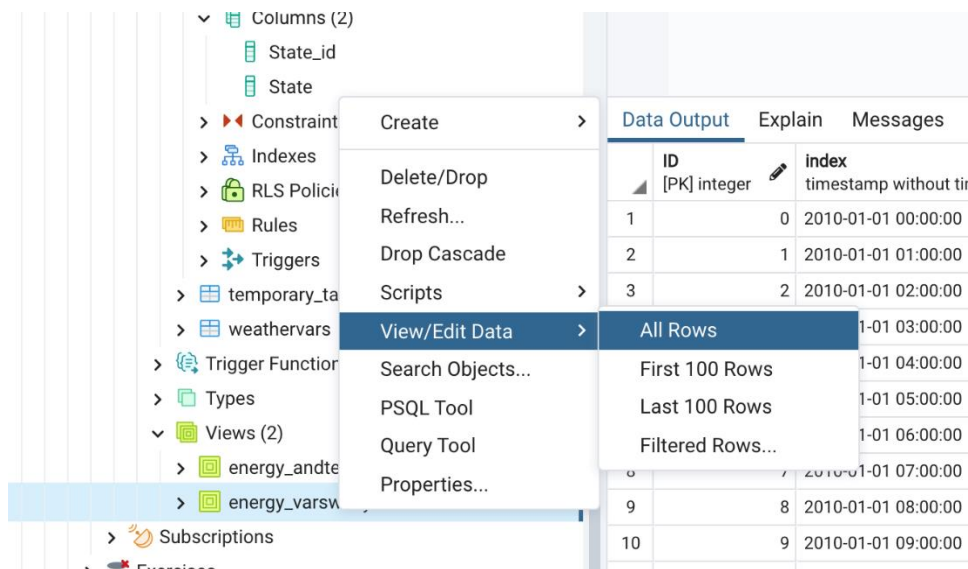


Creación de una Vista









Para crear una vista en PostgreSQL, se ejecuta el siguiente código, en la sección de Query Editor:

```
CREATE VIEW AS energy_varswcity1  
  
    SELECT Date_Time, Energy_Demand, Season, "Temperature",  
        "Perceived_Temperature", "Precipitation", "Humidity", "City_id"  
  
FROM public.energyddemand  
  
INNER JOIN public.weathervars  
  
ON (energyddemand.date_time = "Date_wv");
```

Este Código nos genera la vista necesaria, y se puede exportar fácilmente a csv, por medio del menú de pgadmin.




Al ejecutar la vista, se verifican los resultados son los que se busca exportar:

Data Output		Explain	Messages	Notifications				
	date_time timestamp without time zone	 energy_demand numeric	 season character varying (10)	 Temperature numeric	 Perceived_Temperature numeric	 Precipitation numeric	 Humidity numeric	 City_Id integer
1	2010-01-01 00:00:00	1450.0	Winter	14.7	13.7	0	58.9	6
2	2010-01-01 02:00:00	1279.0	Winter	12.5	12.1	0	64.8	6
3	2010-01-01 07:00:00	1085.0	Winter	8.9	7.9	0	79.3	6
4	2010-01-01 09:00:00	1008.0	Winter	9.1	9.4	0	72.6	6
5	2010-01-01 11:00:00	1049.0	Winter	19.9	23.4	0	45.1	6
6	2010-01-02 03:00:00	1066.0	Winter	11.4	11.5	0	81.4	6
7	2010-01-02 05:00:00	1027.0	Winter	9.6	9.6	0	82.7	6
8	2010-01-02 14:00:00	1359.0	Winter	26.1	30.8	0	37.7	6
9	2010-01-02 15:00:00	1359.0	Winter	26.7	30.3	0	33.5	6
10	2010-01-02 21:00:00	1578.0	Winter	17.4	16.9	0	62.3	6
11	2010-01-02 23:00:00	1485.0	Winter	15.5	15.4	0	68.3	6
12	2010-01-03 03:00:00	1159.0	Winter	10.6	10.7	0	86.7	6

Creación de una Función

Para crear una vista en PostgreSQL, se ejecuta el siguiente código, en la sección de Query Editor:

Energy_demand/postgres@PostgreSQL 14 ▾

Query Editor

Query History

1

2 create function f_return_lluvia (Ciudad integer)

3 RETURNS numeric as \$total\$

4 DECLARE suma_lluvia numeric;

5

6 begin

7 RETURN

8 (SELECT sum ("Precipitation") as suma_lluvia

9 FROM public.weathervars

10 WHERE "City_id"= Ciudad);

11

12

13 end;

14 \$total\$ LANGUAGE plpgsql;

15

16

17

Notifications

Data Output

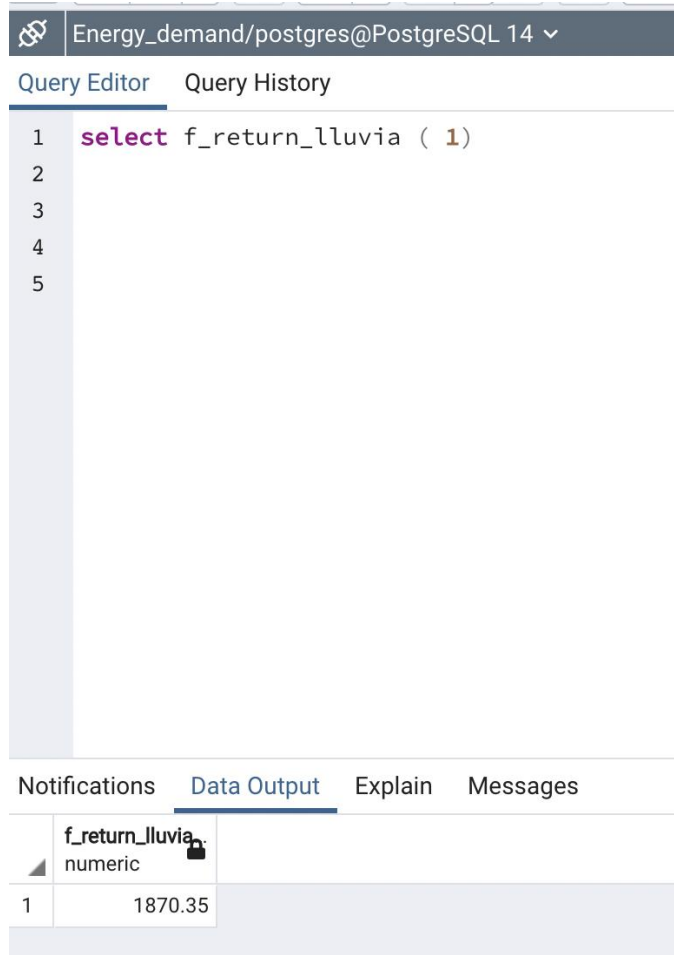
Explain

Messages

CREATE FUNCTION

Query returned successfully in 53 msec.

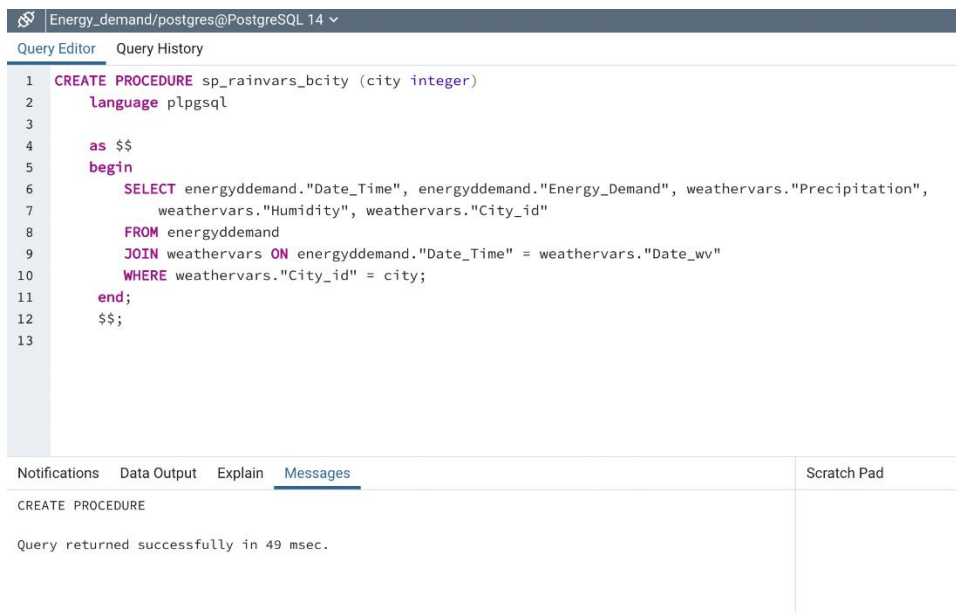
Verificamos que la función se ejecuta con la siguiente línea:



The screenshot shows a PostgreSQL query editor interface. At the top, a dark header bar contains a database icon and the text "Energy_demand/postgres@PostgreSQL 14" with a dropdown arrow. Below this, two tabs are visible: "Query Editor" (active) and "Query History". The query editor contains a single line of SQL code: `select f_return_lluvia (1)`. The code is color-coded: "select" is purple, "f_return_lluvia" is black, and "1" is brown. To the left of the code, line numbers 1 through 5 are listed. Below the query editor, there are four tabs: "Notifications", "Data Output" (active), "Explain", and "Messages". The "Data Output" tab displays a table with one column, "f_return_lluvia", and one data row with the value "1870.35". The column header also indicates the data type "numeric".

	f_return_lluvia numeric
1	1870.35

Create Procedure



```
1 CREATE PROCEDURE sp_rainvars_bcity (city integer)
2   LANGUAGE plpgsql
3
4   AS $$
5   BEGIN
6     SELECT energyddemand."Date_Time", energyddemand."Energy_Demand", weathervars."Precipitation",
7           weathervars."Humidity", weathervars."City_id"
8     FROM energyddemand
9    JOIN weathervars ON energyddemand."Date_Time" = weathervars."Date_wv"
10   WHERE weathervars."City_id" = city;
11   END;
12   $$;
13
```

Notifications Data Output Explain Messages Scratch Pad

CREATE PROCEDURE

Query returned successfully in 49 msec.

Usando PostgreSQL desde Jupyter notebook.

Como ejemplo de la conexión que se puede realizar directamente desde Python, se incluye la libreta llamada (***Connect_pgsqlrgydb.ipynb***) en el repositorio de [github](#).

Para realizar la conexión, se requiere la instalación de la librería `psycopg2`.

Por medio de las conexiones a `psycopg2`, la librería nos permite hacer consultas y modificaciones a las tablas de base de datos en PostgreSQL.

```
curr = conn.cursor()
✓ 0.1s Python

curr.execute('SELECT * FROM encities') #we test with a small reference table of cities
✓ 0.4s Python

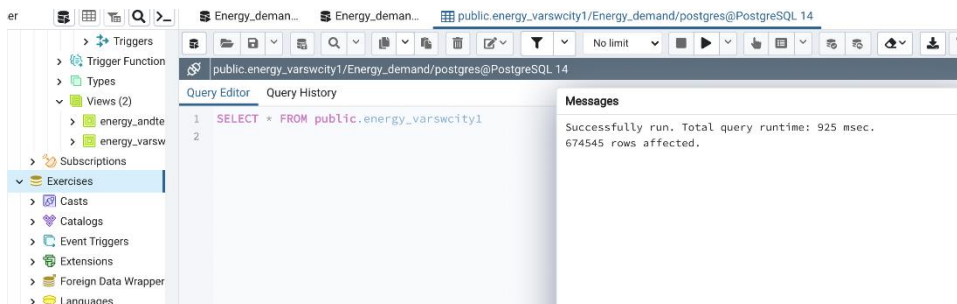
curr.fetchall() #Current data includes 6 cities from Sonora, all StateID is referenced to Sonora
✓ 0.3s Python

[(1, 1, 'Caborca'),
 (2, 1, 'Guaymas'),
 (3, 1, 'Hermosillo'),
 (4, 1, 'Navojoa'),
 (5, 1, 'Nogales'),
 (6, 1, 'Obregon')]

#to close the connection
conn.close
✓ 0.1s Python
```

Análisis de los datos en Jupyter Notebook

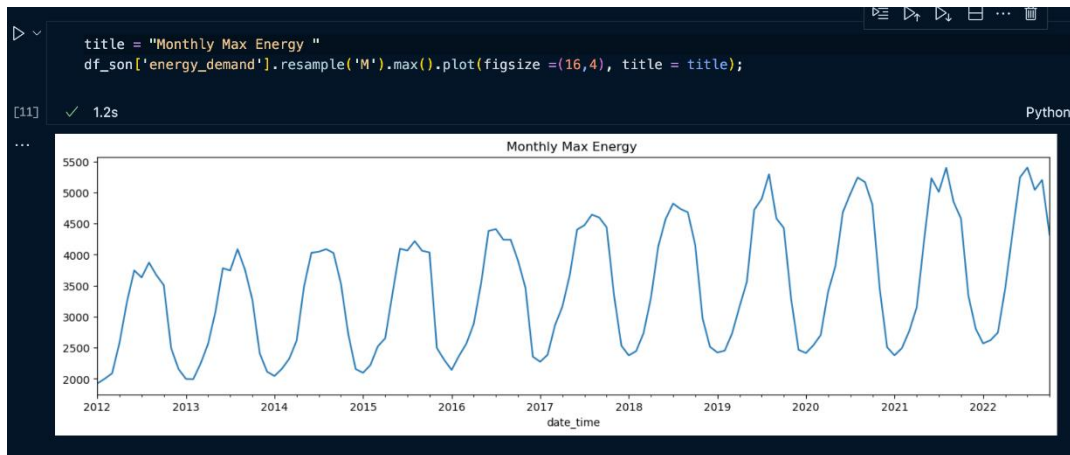
Realizamos un exporte a csv de una de las vistas generadas en PostgreSQL.



Con este exporte generamos la segunda libreta `nrgy_project.ipynb`, la cual se puede encontrar en el repositorio de [github](#).

A continuación, se presentan algunas gráficas generadas con la información de la vista.

Grafica de energía máxima, resampled por Mes.



Gráfica de cajas, para analizar el comportamiento de demanda de energía por temporada (Season).

