

Contenido

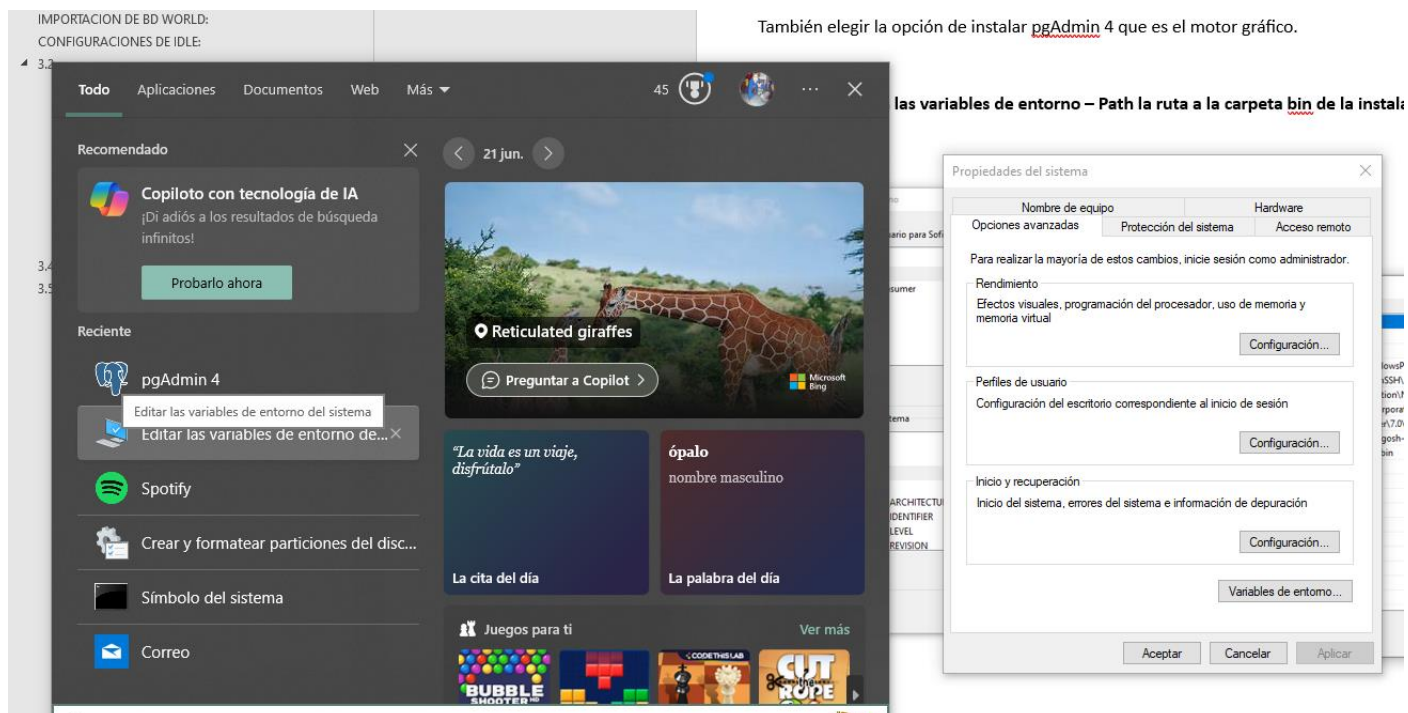
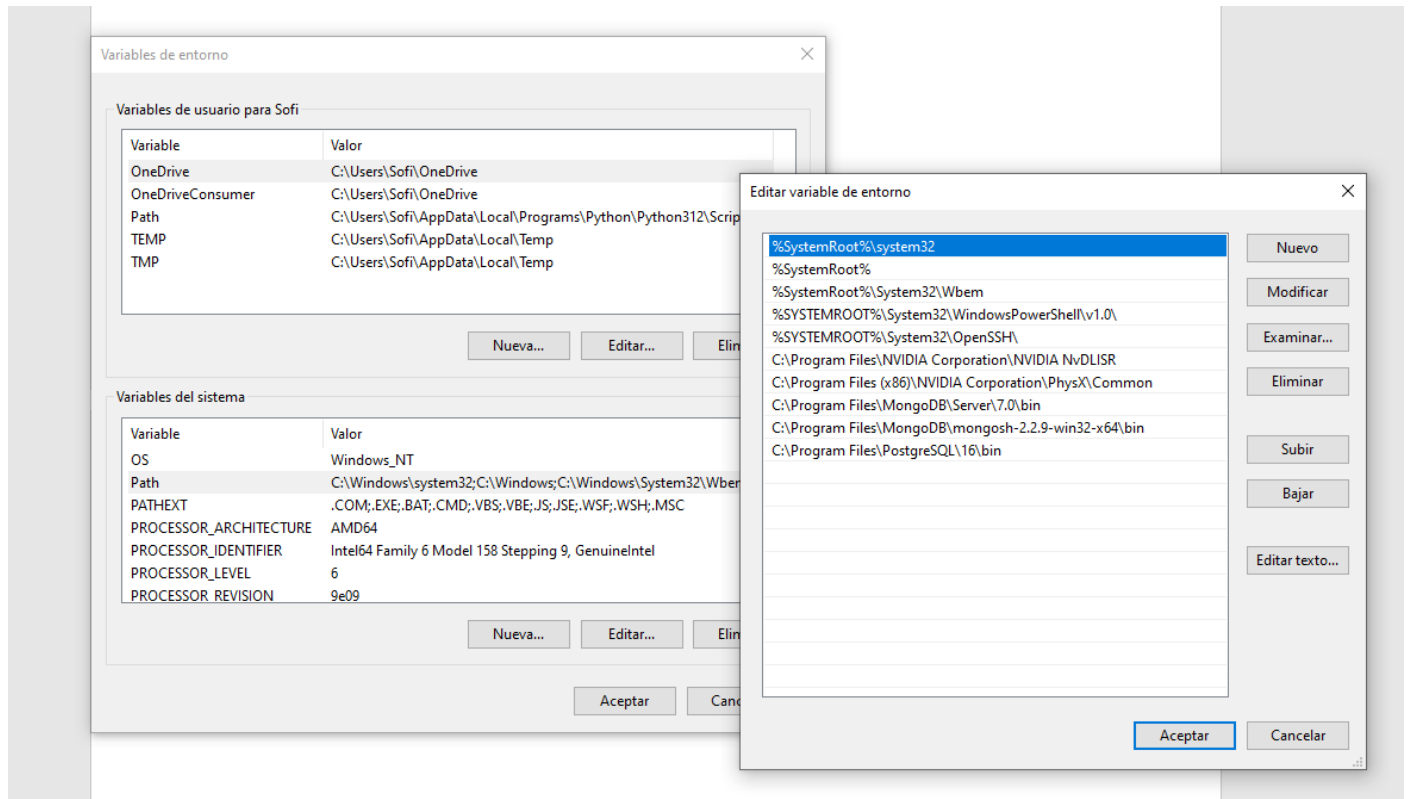
INSTALACIÓN DE POSTGRESQL:	2
IMPORTACION DE BD WORLD:.....	3
CONFIGURACIONES DE IDLE:	5
3.2.....	6
3.2.1.....	6
3.2.2.....	6
3.2.4.....	6
3.2.5.....	6
3.2.6.....	7
3.2.7.....	7
3.2.12.....	7
3.2.14.....	8
3.4.....	8
3.5.....	9
3.6.....	10

INSTALACIÓN DE POSTGRESQL:

Debemos descargarla e instalarla del sitio oficial.

También elegir la opción de instalar pgAdmin 4 que es el motor gráfico.

Luego agregar en las variables de entorno – Path la ruta a la carpeta bin de la instalación.



IMPORTACION DE BD WORLD:

Primero tengo que editar el archivo world.sql porque hay un problema con comillas que hace que gran parte de su código quede entre ellas.

Luego abro el cmd y corro el siguiente comando:

```
psql -U postgres world < "C:\Users\Sofi\OneDrive\Facu\9no cuatrimestre\Sistemas de Gestión de BD\Práctica 3\TP3 - SGBD\datos enunciado\world.sql"
```

```
C:\Users\Sofi>psql -U postgres world < "C:\Users\Sofi\OneDrive\Facu\9no cuatrimestre\Sistemas de Gestión de BD\Práctica 3\TP3 - SGBD\world.sql"
Contraseña para usuario postgres:
BEGIN
SET
CREATE TABLE
CREATE TABLE
CREATE TABLE
COPY 4079
COPY 239
COPY 984
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
ALTER TABLE
COMMIT
ANALYZE
ANALYZE
ANALYZE
```

Veo que se crean bien las tablas y se copia el contenido:

```
C:\Users\Sofi>psql -U postgres
Contraseña para usuario postgres:
psql (16.3)
ADVERTENCIA: El código de página de la consola (850) difiere del código
de página de Windows (1252).
Los caracteres de 8 bits pueden funcionar incorrectamente.
Vea la página de referencia de psql «Notes for Windows users»
para obtener más detalles.
Digite «help» para obtener ayuda.

postgres=# \c world
Ahora está conectado a la base de datos «world» con el usuario «postgres».
world=# \d

Listado de relaciones
Esquema | Nombre | Tipo | Dueño
-----+-----+-----+-----
public | city | tabla | postgres
public | country | tabla | postgres
public | countrylanguage | tabla | postgres
(3 filas)
```

```

world=# select * from city;
  id | name | countrycode | district | population
-----+-----+-----+-----+-----
  1 | Kabul | AFG | Kabol | 1780000
  2 | Qandahar | AFG | Qandahar | 237500
  3 | Herat | AFG | Herat | 186800
  4 | Mazar-e-Sharif | AFG | Balkh | 127800
  5 | Amsterdam | NLD | Noord-Holland | 731200
  6 | Rotterdam | NLD | Zuid-Holland | 593321
  7 | Haag | NLD | Zuid-Holland | 440900
  8 | Utrecht | NLD | Utrecht | 234323
  9 | Eindhoven | NLD | Noord-Brabant | 201843
 10 | Tilburg | NLD | Noord-Brabant | 193238
 11 | Groningen | NLD | Groningen | 172701
 12 | Breda | NLD | Noord-Brabant | 160398
 13 | Apeldoorn | NLD | Gelderland | 153491
 14 | Nijmegen | NLD | Gelderland | 152463
 15 | Enschede | NLD | Overijssel | 149544
 16 | Haarlem | NLD | Noord-Holland | 148772
 17 | Almere | NLD | Flevoland | 142465
 18 | Arnhem | NLD | Gelderland | 138020
 19 | Zaanstad | NLD | Noord-Holland | 135621
 20 | 's-Hertogenbosch | NLD | Noord-Brabant | 129170
 21 | Amersfoort | NLD | Utrecht | 126270
 22 | Maastricht | NLD | Limburg | 122087
 23 | Dordrecht | NLD | Zuid-Holland | 119811
 24 | Leiden | NLD | Zuid-Holland | 117196
 25 | Haarlemmermeer | NLD | Noord-Holland | 110722
 26 | Zoetermeer | NLD | Zuid-Holland | 110214
 27 | Emmen | NLD | Drenthe | 105853
-- Más --

```

```

world=# select * from country;
code | governmentform | name | continent | region | headofstate | capital | code2 | surfacearea | indepyear | population | lifeexpectancy | gnp | gnpold | localname |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
AFG | Afghanistan | Afghanistan | Asia | Southern and Central Asia | Mohammad Omar | Kabul | AF | 652000 | 1919 | 22720000 | 45.9 | 5976.00 | 5976.00 | Afganistan/Afghanistan | Islamic
NLD | Netherlands | Netherlands | Europe | Western Europe | Beatrix | Amsterdam | NL | 41526 | 1581 | 15864000 | 78.3 | 371362.00 | 360478.00 | Nederland | Constit
ANT | Netherlands Antilles | Netherlands Antilles | North America | Caribbean | Beatrix | Willemstad | AN | 800 | 217000 | 74.7 | 1941.00 | 1941.00 | Nederlandse Antillen | Nonmetr
ALB | Albania | Albania | Europe | Southern Europe | Rexhep Mejdani | Tirane | AL | 28748 | 1912 | 3401200 | 71.6 | 3205.00 | 2500.00 | Shqipëria | Republi
DZA | Algeria | Algeria | Africa | Northern Africa | Abdelaziz Bouteflika | Algiers | DZ | 2.381741e+06 | 1962 | 31471000 | 69.7 | 49982.00 | 46966.00 | Al-Jaza'ir/Algérie | Republi
ASM | American Samoa | American Samoa | Oceania | Polynesia | George W. Bush | Pago Pago | AS | 199 | 68000 | 75.1 | 334.00 | 334.00 | Amerika Samoa | US Terr
AND | Andorra | Andorra | Europe | Southern Europe | Joan Enric Vives i Sicília | Andorra la Vella | AD | 468 | 1278 | 78000 | 83.5 | 1630.00 | 1630.00 | Andorra | Parliam
AGO | Angola | Angola | Africa | Central Africa | José Eduardo dos Santos | Luanda | AO | 1.2467e+06 | 1975 | 12878000 | 38.3 | 6648.00 | 7984.00 | Angola | Republi
AIA | Anguilla | Anguilla | North America | Caribbean | Elizabeth II | The Valley | AI | 96 | 8000 | 76.1 | 63.20 | 63.20 | Anguilla | Depend
ATG | Antigua and Barbuda | Antigua and Barbuda | North America | Caribbean | Elizabeth II | St. John's | AG | 442 | 1981 | 68000 | 70.5 | 612.00 | 584.00 | Antigua and Barbuda | Constit
ARE | United Arab Emirates | United Arab Emirates | Asia | Middle East | Zayid bin Sultan al-Nahayan | Abu Dhabi | AE | 83600 | 1971 | 2441000 | 74.1 | 37966.00 | 36846.00 | Al-Imarat al-'Arabiya al-Muttahida | Emirate
ARG | Argentina | Argentina | South America | South America | Fernando de la Rúa | Buenos Aires | AR | 2.7804e+06 | 1816 | 37032000 | 75.1 | 340238.00 | 323310.00 | Argentina | Federal
ARM | Armenia | Armenia | Asia | Middle East | Robert Kochinyan | Yerevan | AM | 29800 | 1991 | 3520000 | 66.4 | 1813.00 | 1813.00 -- Más --

```

CONFIGURACIONES DE IDLE:

Obviamente primero que nada debemos tener Python instalado y configurado.

Luego debo instalar las siguientes librerías:

```
import pandas as pd
import psycpg2
import geopandas as gp
import matplotlib.pyplot as plt
```

```
python -m pip install -U pip
```

```
pip install pandas
```

```
pip install psycpg2
```

```
pip install geopandas
```

```
pip install -U matplotlib
```

3.2

3.2.1

```
SELECT * FROM Customers  
WHERE Country='Mexico';
```

https://www.w3schools.com/sql/sql_where.asp

3.2.2

SELECT DISTINCT → Sin registros duplicados en función de los campos que se desean mostrar.

SELECT DISTINCTROW → Sin registros duplicados en función de TODOS sus campos, no solo aquellos que se desean mostrar. Es decir tienen que ser filas con al menos una diferencia, si esto se cumple pueden repetirse otros datos.

<https://luciamonterorodriguez.com/sql-como-evitar-resultados-duplicados/>

3.2.4

```
SELECT *  
  
FROM sales  
  
ORDER BY sale_date DESC  
  
FETCH FIRST 10 ROWS ONLY
```

```
SELECT *  
  
FROM sales  
  
ORDER BY sale_date DESC  
  
LIMIT 10
```

<https://use-the-index-luke.com/es/sql/resultados-parciales/sentencia-top-n>

3.2.5

-- Selecting the 'working_area' column and counting the number of occurrences for each distinct value

```
SELECT working_area, COUNT(*)
```

-- From the 'agents' table

```
FROM agents
```

-- Grouping the results by the 'working_area' column

```
GROUP BY working_area
```

-- Sorting the results by the second column (COUNT(*)) in ascending order

```
ORDER BY 2;
```

<https://www.w3resource.com/sql/aggregate-functions/count-with-group-by.php>

3.2.6

```
select SUM(sueldo) as SUMA  
from empleados;
```

<https://www.migueltroiano.com/bbdd/funcion-sum-en-postgresql/>

3.2.7

```
SELECT  
    country,  
    SUM(quantity) AS total_quantity  
FROM orders  
GROUP BY country;
```

<https://learnsql.es/blog/como-usar-sum-con-group-by-una-guia-detallada-con-8-ejemplos/>

Debido a que el COUNT() se hace después, para poder usar su alias, es decir la columna cantidadPaíses, debería hacer un select por fuera, quedando:

```
SELECT * FROM(  
    SELECT continent, COUNT(name) AS cantidadPaíses FROM country  
        WHERE population > 20000000  
        GROUP BY continent  
        ORDER BY cantidadPaíses DESC  
    )  
WHERE cantidadPaíses > 15;
```

3.2.12

JOINS

Types of Joins:

INNER JOIN: Returns records that have matching values in both tables

LEFT JOIN: Returns all records from the left table, and the matched records from the right table

RIGHT JOIN: Returns all records from the right table, and the matched records from the left table

FULL JOIN: Returns all records when there is a match in either left or right table

```
SELECT product_id, product_name, category_name
FROM products
INNER JOIN categories ON products.category_id = categories.category_id;
```

https://www.w3schools.com/postgresql/postgresql_joins.php

3.2.14

Los operadores relacionales son los siguientes:

```
=      igual
<>     distinto
>      mayor
<      menor
>=     mayor o igual
<=     menor o igual
```

<https://www.tutorialesprogramacionya.com/postgresqlya/temarios/descripcion.php?inicio=0&cod=165&punto=7>

3.4

IMPORTANTE: Es necesario correr primero el 3.4.sql para tener la tabla creada y luego el 3.4.py

```


16 SELECT code, code2, name FROM country
17     WHERE continent = 'Europe'
18     ORDER BY code DESC;

```

	code [PK] character	code2 character	name text
25	IRL	IE	Ireland
26	HUN	HU	Hungary
27	HRV	HR	Croatia
28	GRC	GR	Greece
29	GIB	GI	Gibraltar
30	GBR	GB	United Kingdom
31	FRO	FO	Faroe Islands
32	FRA	FR	France
33	FIN	FI	Finland
34	EST	EE	Estonia
35	ESP	ES	Spain

3.5


Sin ningún índice sobre la tabla, el “EXPLAIN” nos devuelve lo siguiente:

	QUERY PLAN	
	text	
1	Limit (cost=0.00..3.55 rows=100 width=68) (actual time=3.934..31.191 rows=100 loops=1)	
2	-> Nested Loop (cost=0.00..55074964.85 rows=1552818247 width=68) (actual time=3.933..31.177 rows=100 loops=1)	
3	Join Filter: (s1.countrycode = s2.countrycode)	
4	Rows Removed by Join Filter: 5960	
5	-> Seq Scan on sitio s1 (cost=0.00..27421.00 rows=60564 width=34) (actual time=0.019..0.019 rows=1 loops=1)	
6	Filter: ((entidad)::text ~~ 'a%':text)	
7	-> Materialize (cost=0.00..27723.82 rows=60564 width=34) (actual time=3.911..30.461 rows=6060 loops=1)	
8	-> Seq Scan on sitio s2 (cost=0.00..27421.00 rows=60564 width=34) (actual time=3.908..29.417 rows=6060 loop...	
9	Filter: ((entidad)::text ~~ 'b%':text)	
10	Rows Removed by Filter: 108326	
11	Planning Time: 0.370 ms	
12	Execution Time: 31.336 ms	

Creamos un índice en la columna countrycode de la tabla sitio:

<https://sigdeletras.com/2019/creacion-de-indices-con-postgresql-postgis/>

<https://www.postgresql.org/docs/current/sql-createindex.html>

	QUERY PLAN	
	text	
1	Limit (cost=0.43..0.46 rows=100 width=68) (actual time=0.048..0.296 rows=100 loops=1)	
2	-> Nested Loop (cost=0.43..326380.17 rows=1552818247 width=68) (actual time=0.047..0.292 rows=100 loops=1)	
3	-> Seq Scan on sitio s1 (cost=0.00..27421.00 rows=60564 width=34) (actual time=0.015..0.015 rows=1 loops=1)	
4	Filter: ((entidad)::text ~~ 'a%':text)	
5	Rows Removed by Filter: 75	
6	-> Memoize (cost=0.43..122.70 rows=367 width=34) (actual time=0.030..0.266 rows=100 loops=1)	
7	Cache Key: s1.countrycode	
8	Cache Mode: logical	
9	Hits: 0 Misses: 1 Evictions: 0 Overflows: 0 Memory Usage: 7kB	
10	-> Index Scan using countrycodeindex on sitio s2 (cost=0.42..122.69 rows=367 width=34) (actual time=0.028..0.252 rows=100 loop...	
11	Index Cond: (countrycode = s1.countrycode)	
12	Filter: ((entidad)::text ~~ 'b%':text)	
13	Rows Removed by Filter: 1638	
14	Planning Time: 1.976 ms	
15	Execution Time: 0.320 ms	

Son distintas las cuestiones que realiza:

No necesita gastar tiempo en un Join Filter

No hace un Materialize

Realiza un Memoize buscando la información guardada en cache por el uso de índices.

En lugar de un Seq Scan, realiza un Index Scan

Es raro ver que **pareciera que, al utilizar un índice, se consumió más tiempo** (el costo es de 0.43 ..., contra 0.00...5)

3.6

IMPORTANTE: Es necesario correr primero el 3.4.sql y el 3.4.py para tener la tabla sitio con la información requerida