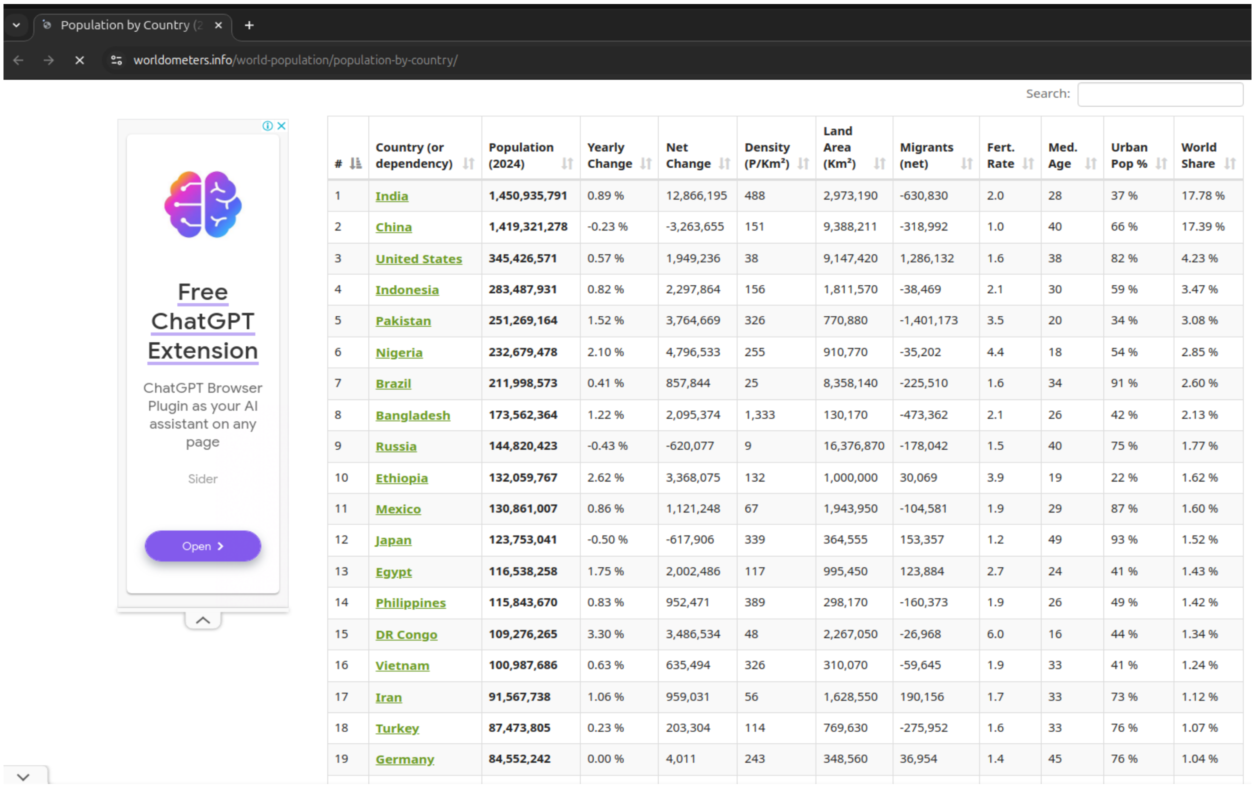
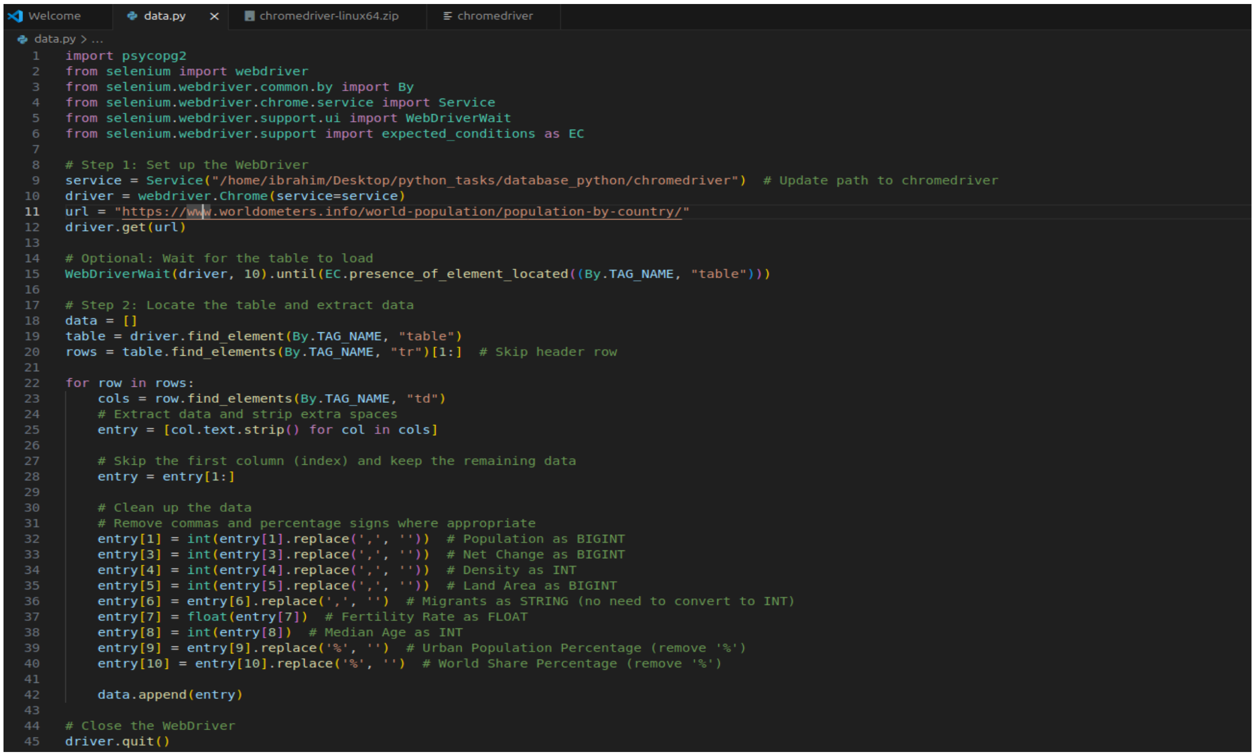
- First I searched for a webpage to scrab its content and store the data in a postgress database server(world population page).


Population by Country (2
worldometers.info/world-population/population-by-country/
Search:
Free ChatGPT Extension
ChatGPT Browser Plugin as your AI assistant on any page
Sider
Open >

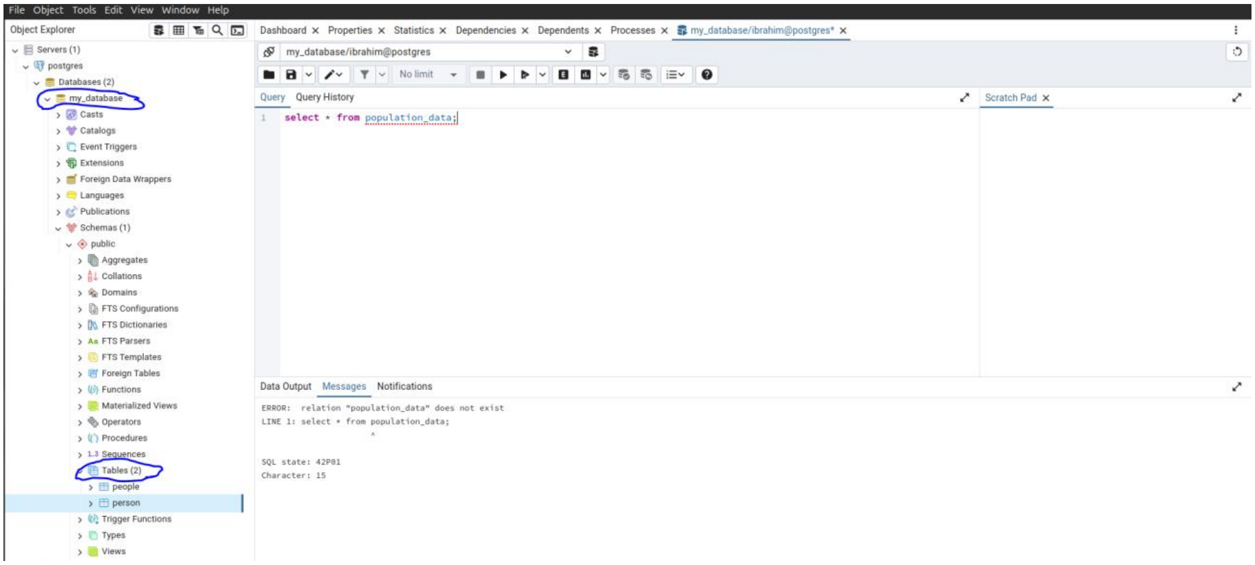
#	Country (or dependency)	Population (2024)	Yearly Change	Net Change	Density (P/Km²)	Land Area (Km²)	Migrants (net)	Fert. Rate	Med. Age	Urban Pop %	World Share
1	India	**1,450,935,791**	0.89 %	12,866,195	488	2,973,190	-630,830	2.0	28	37 %	17.78 %
2	China	**1,419,321,278**	-0.23 %	-3,263,655	151	9,388,211	-318,992	1.0	40	66 %	17.39 %
3	United States	**345,426,571**	0.57 %	1,949,236	38	9,147,420	1,286,132	1.6	38	82 %	4.23 %
4	Indonesia	**283,487,931**	0.82 %	2,297,864	156	1,811,570	-38,469	2.1	30	59 %	3.47 %
5	Pakistan	**251,269,164**	1.52 %	3,764,669	326	770,880	-1,401,173	3.5	20	34 %	3.08 %
6	Nigeria	**232,679,478**	2.10 %	4,796,533	255	910,770	-35,202	4.4	18	54 %	2.85 %
7	Brazil	**211,998,573**	0.41 %	857,844	25	8,358,140	-225,510	1.6	34	91 %	2.60 %
8	Bangladesh	**173,562,364**	1.22 %	2,095,374	1,333	130,170	-473,362	2.1	26	42 %	2.13 %
9	Russia	**144,820,423**	-0.43 %	-620,077	9	16,376,870	-178,042	1.5	40	75 %	1.77 %
10	Ethiopia	**132,059,767**	2.62 %	3,368,075	132	1,000,000	30,069	3.9	19	22 %	1.62 %
11	Mexico	**130,861,007**	0.86 %	1,121,248	67	1,943,950	-104,581	1.9	29	87 %	1.60 %
12	Japan	**123,753,041**	-0.50 %	-617,906	339	364,555	153,357	1.2	49	93 %	1.52 %
13	Egypt	**116,538,258**	1.75 %	2,002,486	117	995,450	123,884	2.7	24	41 %	1.43 %
14	Philippines	**115,843,670**	0.83 %	952,471	389	298,170	-160,373	1.9	26	49 %	1.42 %
15	DR Congo	**109,276,265**	3.30 %	3,486,534	48	2,267,050	-26,968	6.0	16	44 %	1.34 %
16	Vietnam	**100,987,686**	0.63 %	635,494	326	310,070	-59,645	1.9	33	41 %	1.24 %
17	Iran	**91,567,738**	1.06 %	959,031	56	1,628,550	190,156	1.7	33	73 %	1.12 %
18	Turkey	**87,473,805**	0.23 %	203,304	114	769,630	-275,952	1.6	33	76 %	1.07 %
19	Germany	**84,552,242**	0.00 %	4,011	243	348,560	36,954	1.4	45	76 %	1.04 %

- Then I developed the below code to take this table and store it in my postgres database server(with selenium and psycopg2)



```python
import psycopg2
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# Step 1: Set up the WebDriver
service = Service("/home/ibrahim/Desktop/python_tasks/database_python/chromedriver")  # Update path to chromedriver
driver = webdriver.Chrome(service=service)
url = "https://www.worldometers.info/world-population/population-by-country/"
driver.get(url)

# Optional: Wait for the table to load
WebDriverWait(driver, 10).until(EC.presence_of_element_located((By.TAG_NAME, "table")))

# Step 2: Locate the table and extract data
data = []
table = driver.find_element(By.TAG_NAME, "table")
rows = table.find_elements(By.TAG_NAME, "tr")[1:]  # Skip header row

for row in rows:
    cols = row.find_elements(By.TAG_NAME, "td")
    # Extract data and strip extra spaces
    entry = [col.text.strip() for col in cols]

    # Skip the first column (index) and keep the remaining data
    entry = entry[1:]

    # Clean up the data
    # Remove commas and percentage signs where appropriate
    entry[1] = int(entry[1].replace(',', ''))  # Population as BIGINT
    entry[3] = int(entry[3].replace(',', ''))  # Net Change as BIGINT
    entry[4] = int(entry[4].replace(',', ''))  # Density as INT
    entry[5] = int(entry[5].replace(',', ''))  # Land Area as BIGINT
    entry[6] = entry[6].replace(',', '')  # Migrants as STRING (no need to convert to INT)
    entry[7] = float(entry[7])  # Fertility Rate as FLOAT
    entry[8] = int(entry[8])  # Median Age as INT
    entry[9] = entry[9].replace('%', '')  # Urban Population Percentage (remove '%')
    entry[10] = entry[10].replace('%', '')  # World Share Percentage (remove '%')

    data.append(entry)

# Close the WebDriver
driver.quit()
```

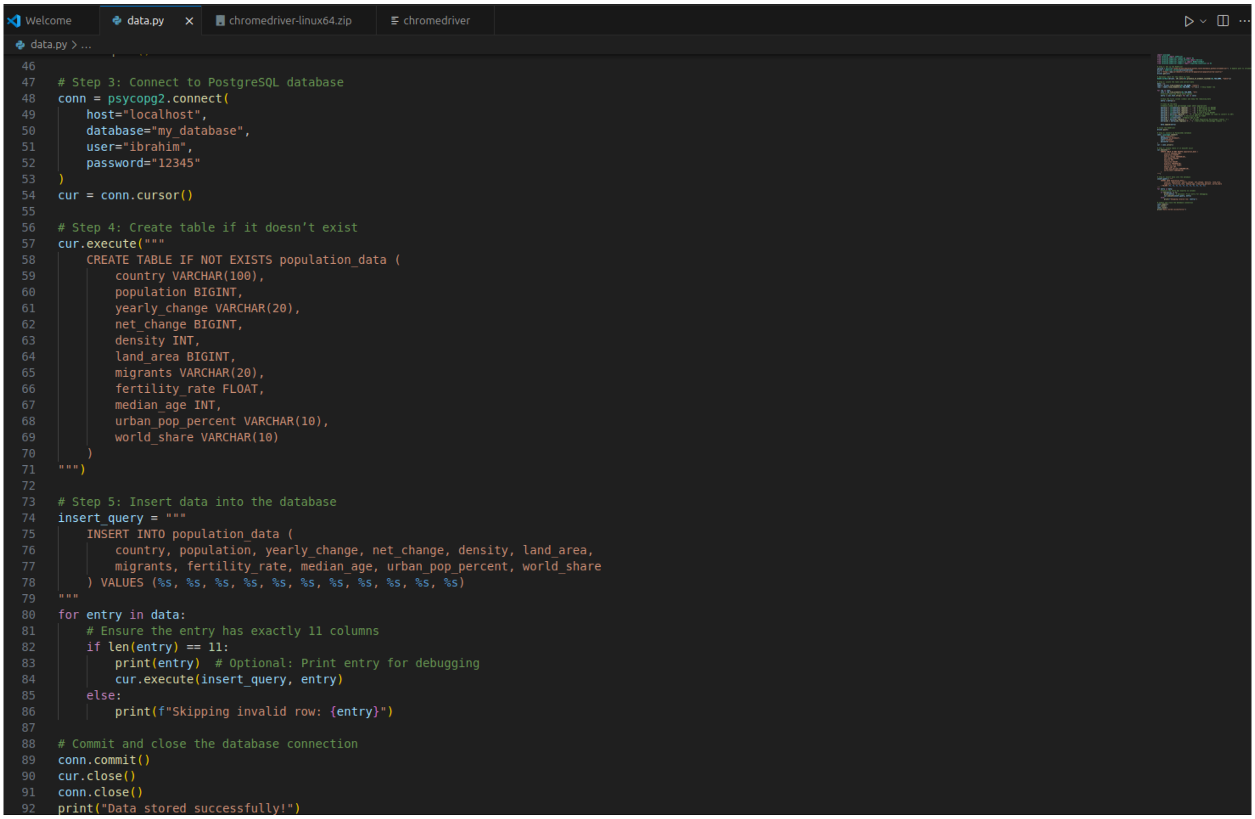
- I created a database called (my_database) on the postgres server



```sql
select * from population_data;
```

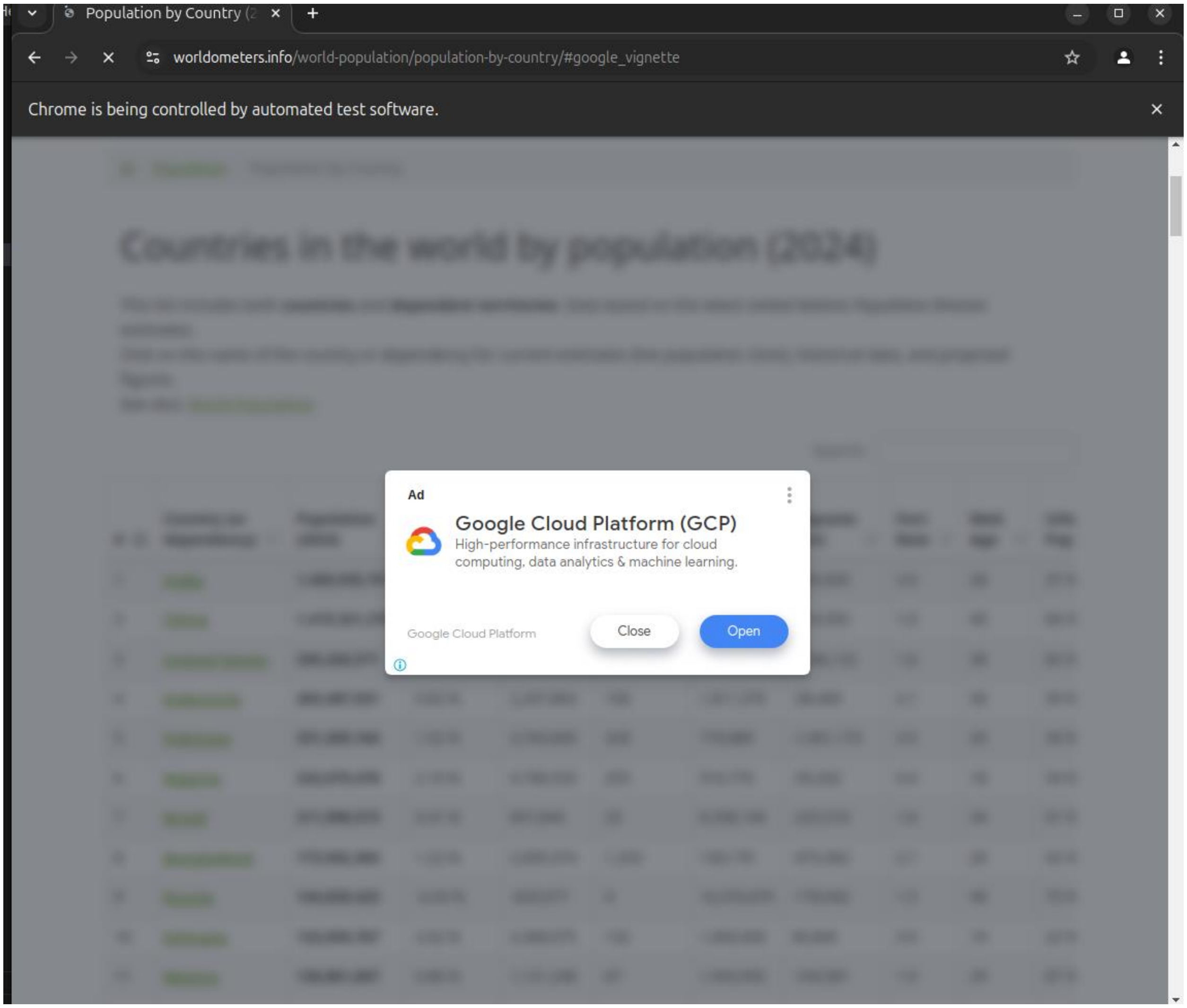
```
ERROR:  relation "population_data" does not exist
LINE 1: select * from population_data;
                      ^


SQL state: 42P01
Character: 15
```

- Now the below part of the code is to create a table called population_data and store the scraped data into it



```python

# Step 3: Connect to PostgreSQL database
conn = psycopg2.connect(
    host="localhost",
    database="my_database",
    user="ibrahim",
    password="12345"
)
cur = conn.cursor()

# Step 4: Create table if it doesn’t exist
cur.execute("""
    CREATE TABLE IF NOT EXISTS population_data (
        country VARCHAR(100),
        population BIGINT,
        yearly_change VARCHAR(20),
        net_change BIGINT,
        density INT,
        land_area BIGINT,
        migrants VARCHAR(20),
        fertility_rate FLOAT,
        median_age INT,
        urban_pop_percent VARCHAR(10),
        world_share VARCHAR(10)
    )
""")

# Step 5: Insert data into the database
insert_query = """
    INSERT INTO population_data (
        country, population, yearly_change, net_change, density, land_area,
        migrants, fertility_rate, median_age, urban_pop_percent, world_share
    ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
"""
for entry in data:
    # Ensure the entry has exactly 11 columns
    if len(entry) == 11:
        print(entry)  # Optional: Print entry for debugging
        cur.execute(insert_query, entry)
    else:
        print(f"Skipping invalid row: {entry}")

# Commit and close the database connection
conn.commit()
cur.close()
conn.close()
print("Data stored successfully!")
```

- Let’s run the script and see the result



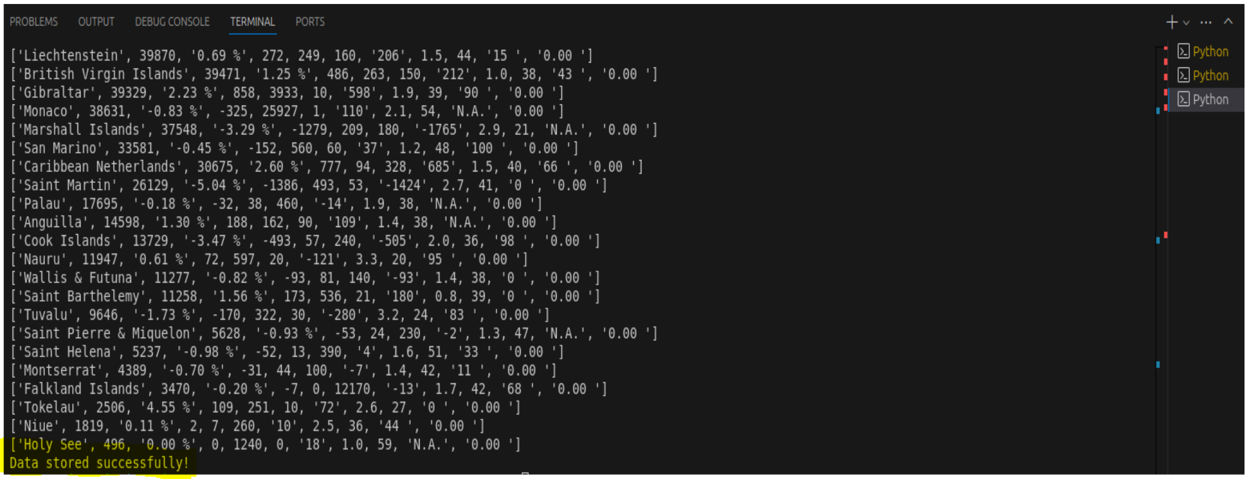
Population by Country (2
worldometers.info/world-population/population-by-country/#google_vignette

Chrome is being controlled by automated test software.

Countries in the world by population (2024)

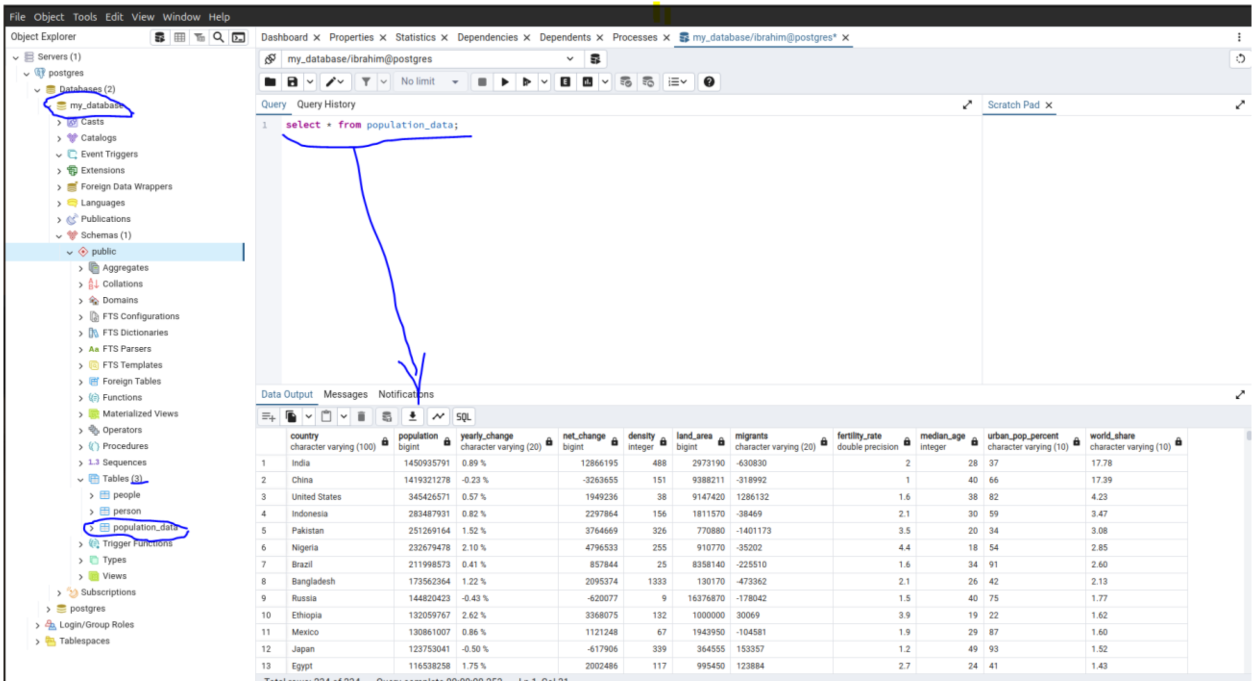
Ad
Google Cloud Platform (GCP)
High-performance infrastructure for cloud computing, data analytics & machine learning.
Google Cloud Platform
Close
Open

- Verify the results



```
['Liechtenstein', 39870, '0.69 %', 272, 249, 160, '206', 1.5, 44, '15 ', '0.00 ']
['British Virgin Islands', 39471, '1.25 %', 486, 263, 150, '212', 1.0, 38, '43 ', '0.00 ']
['Gibraltar', 39329, '2.23 %', 858, 3933, 10, '598', 1.9, 39, '90 ', '0.00 ']
['Monaco', 38631, '-0.83 %', -325, 25927, 1, '110', 2.1, 54, 'N.A.', '0.00 ']
['Marshall Islands', 37548, '-3.29 %', -1279, 209, 180, '-1765', 2.9, 21, 'N.A.', '0.00 ']
['San Marino', 33581, '-0.45 %', -152, 560, 60, '37', 1.2, 48, '100 ', '0.00 ']
['Caribbean Netherlands', 30675, '2.60 %', 777, 94, 328, '685', 1.5, 40, '66 ', '0.00 ']
['Saint Martin', 26129, '-5.04 %', -1386, 493, 53, '-1424', 2.7, 41, '0 ', '0.00 ']
['Palau', 17695, '-0.18 %', -32, 38, 460, '-14', 1.9, 38, 'N.A.', '0.00 ']
['Anguilla', 14598, '1.30 %', 188, 162, 90, '109', 1.4, 38, 'N.A.', '0.00 ']
['Cook Islands', 13729, '-3.47 %', -493, 57, 240, '-505', 2.0, 36, '98 ', '0.00 ']
['Nauru', 11947, '0.61 %', 72, 597, 20, '-121', 3.3, 20, '95 ', '0.00 ']
['Wallis & Futuna', 11277, '-0.82 %', -93, 81, 140, '-93', 1.4, 38, '0 ', '0.00 ']
['Saint Barthelemy', 11258, '1.56 %', 173, 536, 21, '180', 0.8, 39, '0 ', '0.00 ']
['Tuvalu', 9646, '-1.73 %', -170, 322, 30, '-280', 3.2, 24, '83 ', '0.00 ']
['Saint Pierre & Miquelon', 5628, '-0.93 %', -53, 24, 230, '-2', 1.3, 47, 'N.A.', '0.00 ']
['Saint Helena', 5237, '-0.98 %', -52, 13, 390, '4', 1.6, 51, '33 ', '0.00 ']
['Montserrat', 4389, '-0.70 %', -31, 44, 100, '-7', 1.4, 42, '11 ', '0.00 ']
['Falkland Islands', 3470, '-0.20 %', -7, 0, 12170, '-13', 1.7, 42, '68 ', '0.00 ']
['Tokelau', 2506, '4.55 %', 109, 251, 10, '72', 2.6, 27, '0 ', '0.00 ']
['Niue', 1819, '0.11 %', 2, 7, 260, '10', 2.5, 36, '44 ', '0.00 ']
['Holy See', 496, '0.00 %', 0, 1240, 0, '18', 1.0, 59, 'N.A.', '0.00 ']
Data stored successfully!
```

- Check the data on the postgres server



```
select * from population_data;
```

	country character varying (100)	population bigint	yearly_change character varying (20)	net_change bigint	density integer	land_area bigint	migrants character varying (20)	fertility_rate double precision	median_age integer	urban_pop_percent character varying (10)	world_share character varying (10)
1	India	1450935791	0.89 %	12866195	488	2973190	-630830	2	28	37	17.78
2	China	1419321278	-0.23 %	-3263655	151	9388211	-318992	1	40	66	17.39
3	United States	345426571	0.57 %	1949236	38	9147420	1286132	1.6	38	82	4.23
4	Indonesia	283487931	0.82 %	2297864	156	1811570	-38469	2.1	30	59	3.47
5	Pakistan	251269164	1.52 %	3764669	326	770880	-1401173	3.5	20	34	3.08
6	Nigeria	232679478	2.10 %	4796533	255	910770	-35202	4.4	18	54	2.85
7	Brazil	211998573	0.41 %	857844	25	8358140	-225510	1.6	34	91	2.60
8	Bangladesh	173562364	1.22 %	2095374	1333	130170	-473362	2.1	26	42	2.13
9	Russia	144820423	-0.43 %	-620077	9	16376870	-178042	1.5	40	75	1.77
10	Ethiopia	132059767	2.62 %	3368075	132	1000000	30069	3.9	19	22	1.62
11	Mexico	130861007	0.86 %	1121248	67	1943950	-104581	1.9	29	87	1.60
12	Japan	123753041	-0.50 %	-617906	339	364555	153357	1.2	49	93	1.52
13	Egypt	116538258	1.75 %	2002486	117	995450	123884	2.7	24	41	1.43