DELL | Data Engineering Challenge

COVID-19 analysis

Analyst: Juliana Pace Bravio

Date: 19/10/2023

Loading Data: COVID

1. Install PostgreSQL
2. Create a Database: createdb covid_data
3. Download JSON Data
4. Install Python
5. Install Required Libraries:

```
pip install --upgrade pip
pip install "psycopg[binary,pool]"
```

6. Create a Table:

```
CREATE TABLE covid_data (
  id serial PRIMARY KEY,
  country VARCHAR(255),
  country_code VARCHAR(3),
  continent VARCHAR(255),
  population INT,
  indicator VARCHAR(255),
  weekly_count INT,
  year_week VARCHAR(7),
  rate_14_day NUMERIC(9, 4),
  cumulative_count INT,
  source VARCHAR(255),
  note TEXT
);
```

7. Python script to retrieve the JSON data and insert it into the PostgreSQL database

```
        import psycopg
        import json
# Connect to the PostgreSQL database
        conn = psycopg.connect("dbname=covid_data user=postgres
        password=Senha@23")
# Open the JSON file and parse its contents
        with open('covid_data.json', 'r') as file:
          data = json.load(file)
```

```
# Create a cursor
        cur = conn.cursor()
# Loop through the JSON data and insert it into the database
        for entry in data:
          insert_query = """
          INSERT INTO covid_data (country, country_code, continent,
        population, indicator, weekly_count, year_week, rate_14_day,
        cumulative_count, source, note)
          VALUES (%(country)s, %(country_code)s, %(continent)s,
        %(population)s, %(indicator)s, %(weekly_count)s, %(year_week)s,
        %(rate_14_day)s, %(cumulative_count)s, %(source)s, %(note)s);
          """
  # Check for missing columns and replace them with None if necessary
          for column in ['country', 'country_code', 'continent', 'population',
        'indicator', 'weekly_count', 'year_week', 'rate_14_day',
        'cumulative_count', 'source', 'note']:
            if column not in entry:
              entry[column] = None

          cur.execute(insert_query, entry)
# Commit the changes and close the cursor and connection
        conn.commit()
        cur.close()
        conn.close()

        print("Data inserted successfully.")
```

8. Run the Python Script: python Load_Script.py

Loading Data: Countries

1. Download CSV Data
2. Create Table:

```
CREATE TABLE countries (
  Country VARCHAR,
  Region VARCHAR,
  Population INT,
  Area VARCHAR,
  Pop_Density VARCHAR,
  Coastline VARCHAR,
  Net_Migration VARCHAR,
  Infant_Mortality VARCHAR,
  GDP VARCHAR,
  Literacy VARCHAR,
Phones VARCHAR,
Arable VARCHAR,
Crops VARCHAR,
Other VARCHAR,
Climate VARCHAR,
Birthrate VARCHAR,
Deathrate VARCHAR,
Agriculture VARCHAR,
Industry VARCHAR,
Service VARCHAR);
```

3. Connect to PostgreSQL: psql -U postgres -d covid_data
4. Load Data:

```
\copy countries FROM
'C:\Users\Juliana\Desktop\DELL\archive\countries.csv' WITH (FORMAT
CSV, HEADER, DELIMITER ',', QUOTE '"')
```

5. Transform Data:

```
UPDATE countries
SET pop_density = REPLACE(pop_density, ',', '.'),
coastline = REPLACE(coastline, ',', '.');
Net_Migration = REPLACE(Net_Migration, ',', '.'),
Infant_Mortality = REPLACE(Infant_Mortality, ',', '.'),
GDP = REPLACE(GDP, ',', '.'),
Literacy = REPLACE(Literacy, ',', '.'),
Phones = REPLACE(Phones, ',', '.'),
Arable = REPLACE(Arable, ',', '.'),
Crops = REPLACE(Crops, ',', '.'),
Other = REPLACE(Other, ',', '.'),
Climate = REPLACE(Climate, ',', '.'),
Birthrate = REPLACE(Birthrate, ',', '.'),
Deathrate = REPLACE(Deathrate, ',', '.'),
Agriculture = REPLACE(Agriculture, ',', '.'),
Industry = REPLACE(Industry, ',', '.'),
Service = REPLACE(Service, ',', '.');

ALTER TABLE countries
ALTER COLUMN pop_density TYPE FLOAT USING (pop_density::FLOAT);
ALTER TABLE countries
ALTER COLUMN GDP TYPE FLOAT USING (GDP::FLOAT);
```

Create a Pipeline

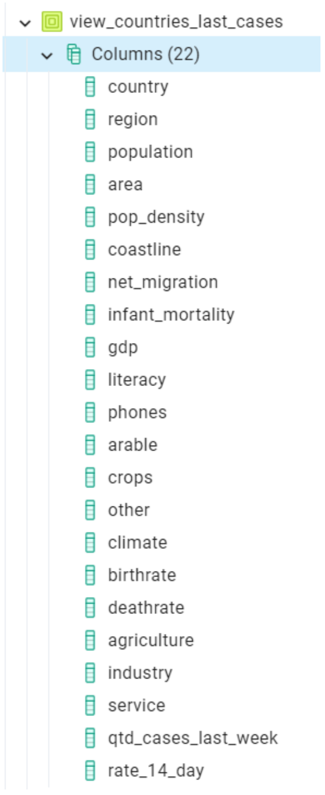
file: Update_data.py

```
import requests
import json
import psycopg

# Define the URL of the COVID-19 data source
data_source_url = 'https://opendata.ecdc.europa.eu/covid19/nationalcasedeath/json/'

# Connect to the PostgreSQL database
conn = psycopg.connect("dbname=covid_data user=postgres password=Senha@23")
cursor = conn.cursor()

try:
    # Fetch the JSON data from the source
    response = requests.get(data_source_url)
    data = response.json()

# Find the maximum 'year_week' value in the JSON data
    max_year_in_json = max(record['year_week'] for record in data)

    # Query the database to find the maximum 'year_week' value that was inserted
    cursor.execute("SELECT MAX(year_week) FROM covid_data")
    max_year_in_db = cursor.fetchone()[0]

    if max_year_in_json > max_year_in_db:
        # If a new version is available, insert the new records
      #  new_records = [record for record in data if record['year_week'] >
max_year_in_db]

        for record in data:
            # Check for missing columns and replace them with None if necessary
            for column in ['country', 'country_code', 'continent', 'population', 'indicator',
'weekly_count', 'year_week', 'rate_14_day', 'cumulative_count', 'source', 'note']:
                if column not in record:
                    record[column] = None

            # Define the insert query
            insert_query = """
            INSERT INTO covid_data (country, country_code, continent, population,
indicator, weekly_count, year_week, rate_14_day, cumulative_count, source, note)
```

```
            VALUES (%(country)s, %(country_code)s, %(continent)s, %(population)s,
%(indicator)s, %(weekly_count)s, %(year_week)s, %(rate_14_day)s,
%(cumulative_count)s, %(source)s, %(note)s);
            """

            # Insert the record into the database
            cursor.execute(insert_query, record)


        # Commit the changes to the database
        conn.commit()

    else:
        print("No new data available.")

except Exception as e:
    print(f"Error: {e}")

finally:
    # Close the cursor and database connection
    cursor.close()
    conn.close()
```

Create a View

```
CREATE VIEW View_Countries_Last_Cases AS
SELECT ps.*,
        cov.weekly_count as qtd_cases_last_week,
        cov.rate_14_day
FROM countries ps
LEFT JOIN (
        SELECT country,
                weekly_count,
                rate_14_day
        FROM covid_data
        WHERE year_week = '2023-40'
        AND indicator = 'cases') cov
ON ps.country = cov.country
```



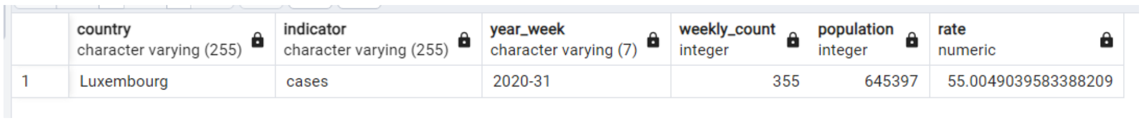
view_countries_last_cases
Columns (22)
country
region
population
area
pop_density
coastline
net_migration
infant_mortality
gdp
literacy
phones
arable
crops
other
climate
birthrate
deathrate
agriculture
industry
service
qtd_cases_last_week
rate_14_day

Queries

1. *What is the country with the highest number of Covid-19 cases per 100 000 Habitants at 31/07/2020?*

```
-- The only data information on covid_data is the year and number of week.
--Considering that 31/07/2020 is the 31 week of the year
SELECT country,
        indicator,
        year_week,
        weekly_count,
        population,
        weekly_count/(population*0.00001) as rate
FROM covid_data
WHERE year_week = '2020-31' AND indicator = 'cases'
AND country <> 'EU/EEA (total)'
ORDER BY weekly_count/(population*0.00001) DESC
LIMIT 1;
```

Answer is LUXEMBOURG

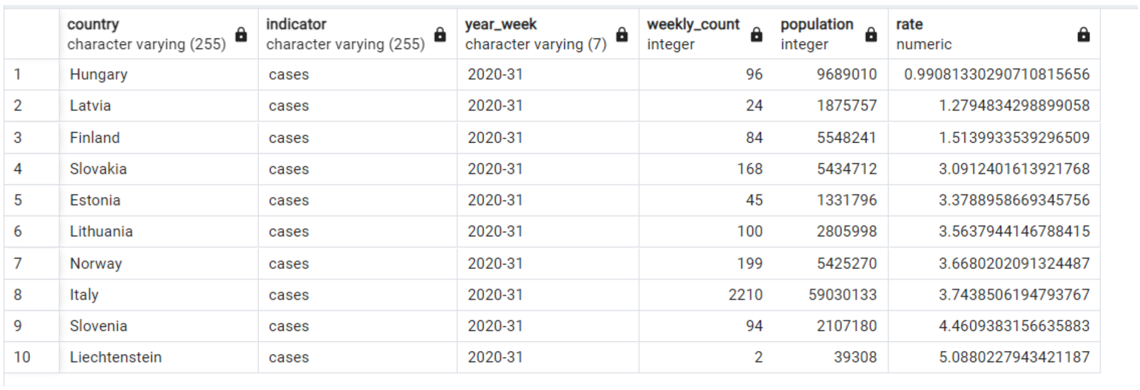


	country character varying (255)	indicator character varying (255)	year_week character varying (7)	weekly_count integer	population integer	rate numeric
1	Luxembourg	cases	2020-31	355	645397	55.0049039583388209

2. *What is the top 10 countries with the lowest number of Covid-19 cases per 100 000 Habitants at 31/07/2020?*

```
SELECT  country,
        indicator,
        year_week,
        weekly_count,
        population,
        weekly_count/(population*0.00001) as rate
FROM covid_data
WHERE year_week = '2020-31' AND indicator = 'cases'
AND country <> 'EU/EEA (total)'
ORDER BY weekly_count/(population*0.00001)
LIMIT 10;
```

Answer: Hungary, Latvia, Finland, Slvakia, Estonia, Lithuania, Norway, Italy, Slovenia, Liechtenstein.

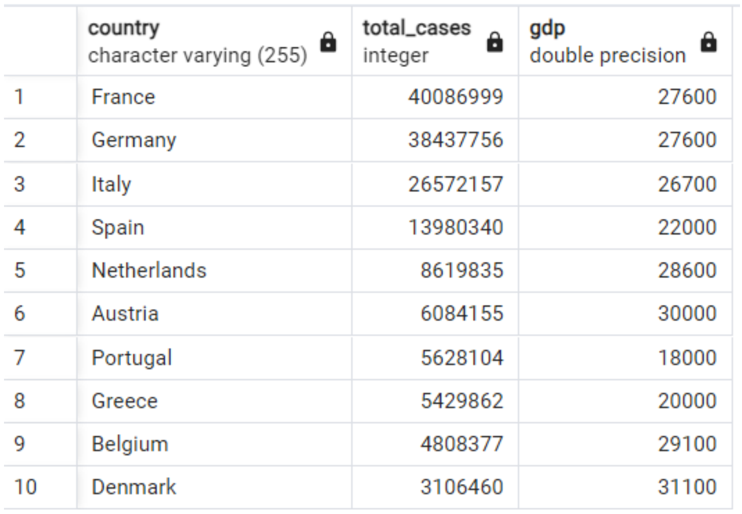


	country character varying (255)	indicator character varying (255)	year_week character varying (7)	weekly_count integer	population integer	rate numeric
1	Hungary	cases	2020-31	96	9689010	0.99081330290710815656
2	Latvia	cases	2020-31	24	1875757	1.2794834298899058
3	Finland	cases	2020-31	84	5548241	1.5139933539296509
4	Slovakia	cases	2020-31	168	5434712	3.0912401613921768
5	Estonia	cases	2020-31	45	1331796	3.3788958669345756
6	Lithuania	cases	2020-31	100	2805998	3.5637944146788415
7	Norway	cases	2020-31	199	5425270	3.6680202091324487
8	Italy	cases	2020-31	2210	59030133	3.7438506194793767
9	Slovenia	cases	2020-31	94	2107180	4.4609383156635883
10	Liechtenstein	cases	2020-31	2	39308	5.0880227943421187

3. *What is the top 10 countries with the highest number of cases among the top 20 richest countries (by GDP per capita)?*

```
--Adjust of country name
UPDATE countries
SET country = REPLACE(country, 'Czech Republic ', 'Czechia');

-- Top 20 richest countries -> 10 high number of cases
SELECT * FROM
        (SELECT cov.country,
                MAX(cov.cumulative_count) as total_cases,
                ps.gdp
        FROM covid_data cov
        LEFT JOIN countries ps
        ON RTRIM(cov.country) = RTRIM(ps.country)
        WHERE cov.indicator = 'cases'
        AND cov.country <> 'EU/EEA (total)'
        GROUP BY cov.country, ps.gdp
        ORDER BY ps.gdp DESC
        LIMIT 20) n
ORDER BY n.total_cases DESC;
```

Answer: France, Germany, Italy, Spain, Netherlands, Austria, Portugal, Greece, Belgium, Denmark.



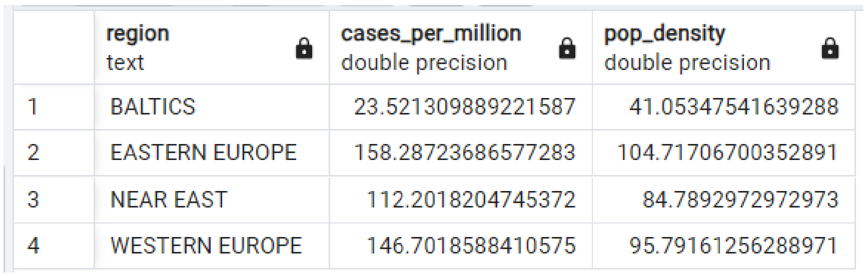
	country character varying (255)	total_cases integer	gdp double precision
1	France	40086999	27600
2	Germany	38437756	27600
3	Italy	26572157	26700
4	Spain	13980340	22000
5	Netherlands	8619835	28600
6	Austria	6084155	30000
7	Portugal	5628104	18000
8	Greece	5429862	20000
9	Belgium	4808377	29100
10	Denmark	3106460	31100

4. *List all the regions with the number of cases per million of inhabitants and display information on population density, for 31/07/2020.*

```
--Considering that 31/07/2020 is the 31 week of the year
    WITH question_4 as (
        SELECT cov.country,
            ps.country,
            RTRIM(ps.region) as region,
            ps.population,
            ps.population/1000000 as pop_per_million,
            ps.area,
            cov.weekly_count
        FROM covid_data cov
        LEFT JOIN countries ps
        ON RTRIM(cov.country) = RTRIM(ps.country)
        WHERE cov.indicator = 'cases'
        AND cov.year_week = '2020-31'
        AND cov.country <> 'EU/EEA (total)')

    SELECT region,
        SUM(weekly_count)/SUM(pop_per_million) AS cases_per_million,
        SUM(population)/SUM(area) AS pop_density
```

```
FROM question_4
GROUP BY region;
```

Answer:

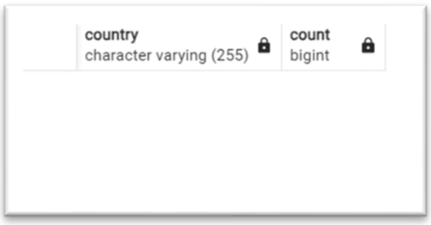


	region text	cases_per_million double precision	pop_density double precision
1	BALTICS	23.521309889221587	41.05347541639288
2	EASTERN EUROPE	158.28723686577283	104.71706700352891
3	NEAR EAST	112.2018204745372	84.7892972972973
4	WESTERN EUROPE	146.7018588410575	95.79161256288971

5. *Query the data to find duplicated records.*

--Validating table question_4

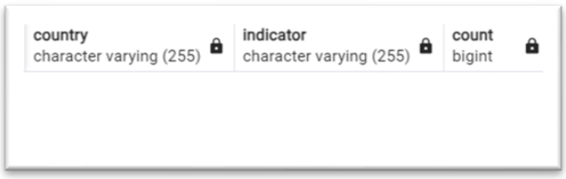
```
SELECT country, count(*) as count
FROM question_4
GROUP BY country
HAVING count(*) > 1;
```



country character varying (255)	count bigint

--Each year has 52 weeks. The database has data from 2020,2021,2022 and 2023 -> so 197 weeks analized

```
SELECT country,
       indicator,
       count(*) as count
FROM covid_data
GROUP BY country, indicator
HAVING count(*) <> 197
```



country character varying (255)	indicator character varying (255)	count bigint

Conclusion: No duplicated data

6. *Analyze the performance of all the queries and describes what you see. Get improvements suggestions.*

Query 1 and 2 are similar, the difference is the limit size. Both queries are fast with planning time of 0.107ms and execution time of 2.073 ms and 2.172 ms. There is small memory usage 25kB. Also, both queries removed by filter 12184 rows, improving the performance.

Query 3 has a planning time of 0.926 ms and execution time of 5.781 ms. Sort Method is quicksort operation with a low memory usage, which is efficient. Groping by consumes 217kB of memory. The join operation takes a reasonable amount of time.

Query 4 has a planning time of 0.181 ms and execution time of 2.256 ms. The aggregate group cost is 401.64, and the actual time is 2.220 ms. The join condition is based on trimming the 'country' column in both tables. Removed by filter 12184 rows, improving the performance.

The query's execution time is relatively low, which is a good sign. However, if this query is part of a user-facing application and users expect near-instant results, it might be considered too slow. On the other hand, if this query runs in the background as part of a batch process, it might be perfectly fine.

As improvements suggest, it is important to properly indexing the database tables. With the indexes, the usage of joins and filters, per example, can show a better performance. Other improvement is Guarente the countries names are written equals before do the correlation between table, in that way it's not necessary the usage of rtrim().

The most important activity is to understand the business rules clearly and define all the context of the outcome desired. In that way, it's possible define a strategy to select just the information necessary, removing unnecessary rows and automatically improving the query performance.

Enrich the information

Enrich the information with any other piece of data (available on the web) and justify the choice.

New Data: Testing for COVID-19
Link: https://www.ecdc.europa.eu/en/publications-data/covid-19-testing

The number of reported COVID-19 cases is related to the number of tests conducted. A higher number of tests can lead to the detection of more cases, which is an essential factor in understanding the scale of the pandemic in a region. Low testing rates may result in underreporting and a skewed perception of the actual spread of the virus.

Furthermore, by examining testing data with case data, you can gain insights into the virus's spread and the effectiveness of control measures. If the number of tests increases but the number of cases remains stable or decreases, it may suggest that containment measures are working. Conversely, a high positivity rate (the proportion of positive tests among all tests conducted) can indicate that the virus is spreading rapidly.

Analysis:

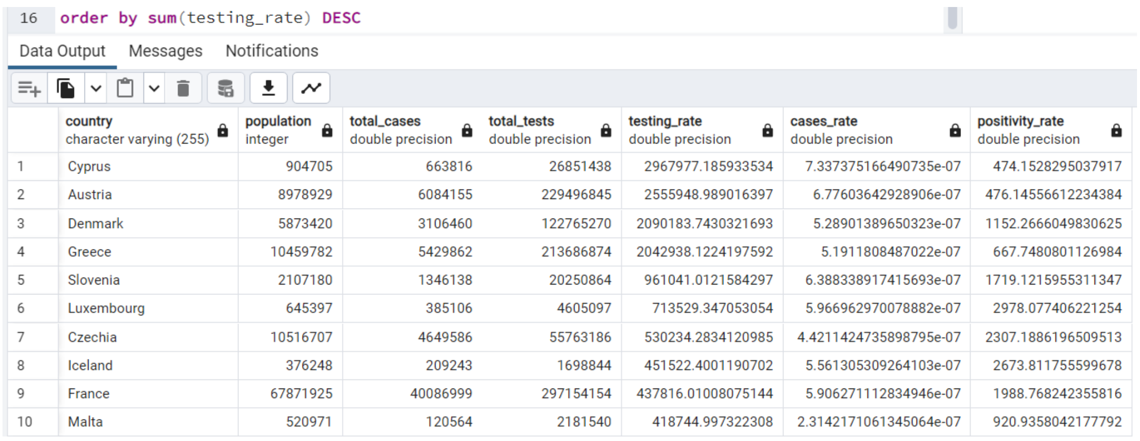
CREATE VIEW CASES AND TEST:

```
CREATE VIEW View_Cases_Testing AS
SELECT cov.country,
        cov.population,
        cov.indicator,
        cov.year_week,
        cov.rate_14_day,
        cov.cumulative_count,
        test.new_cases,
        test.tests_done,
        test.testing_rate,
        test.positivity_rate
FROM covid_data cov
LEFT JOIN test_covid test
ON cov.country = test.country
AND cov.year_week=test.year_week
WHERE indicator = 'cases';
```

THEN QUERY:

```
SELECT country,
        population,
        sum(new_cases) as total_cases,
        sum(tests_done) as total_tests,
        sum(testing_rate) as testing_rate,
        sum(new_cases)/population*0.000001 as cases_rate,
        sum(positivity_rate) as positivity_rate
FROM view_cases_testing
WHERE country <> 'EU/EEA (total)'
GROUP BY country, population
```

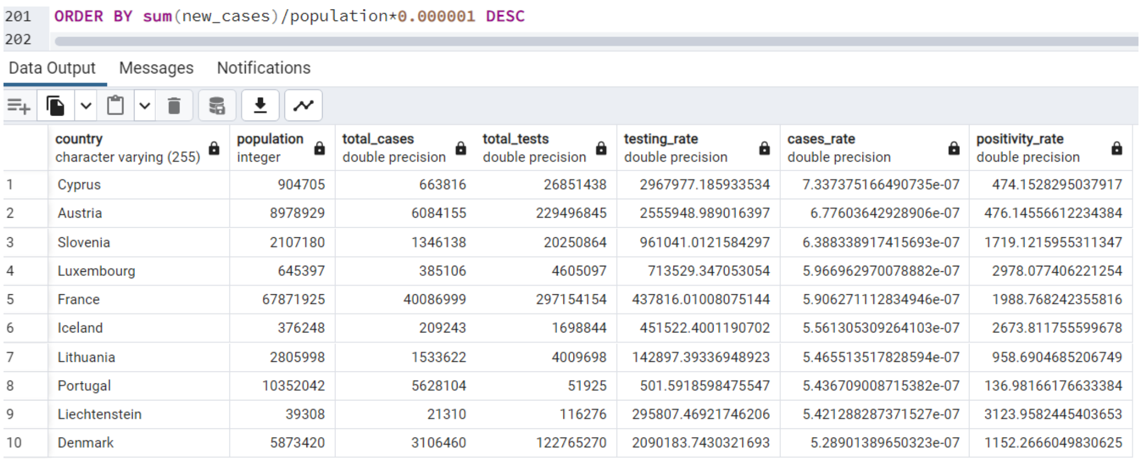
```
--ORDER BY sum(new_cases)/population*0.000001 DESC
ORDER BY sum(testing_rate) DESC
```



```
order by sum(testing_rate) DESC
```

Data Output | Messages | Notifications

	country character varying (255)	population integer	total_cases double precision	total_tests double precision	testing_rate double precision	cases_rate double precision	positivity_rate double precision
1	Cyprus	904705	663816	26851438	2967977.185933534	7.337375166490735e-07	474.1528295037917
2	Austria	8978929	6084155	229496845	2555948.989016397	6.77603642928906e-07	476.14556612234384
3	Denmark	5873420	3106460	122765270	2090183.7430321693	5.28901389650323e-07	1152.2666049830625
4	Greece	10459782	5429862	213686874	2042938.1224197592	5.1911808487022e-07	667.7480801126984
5	Slovenia	2107180	1346138	20250864	961041.0121584297	6.388338917415693e-07	1719.1215955311347
6	Luxembourg	645397	385106	4605097	713529.347053054	5.966962970078882e-07	2978.077406221254
7	Czechia	10516707	4649586	55763186	530234.2834120985	4.4211424735898795e-07	2307.1886196509513
8	Iceland	376248	209243	1698844	451522.4001190702	5.561305309264103e-07	2673.811755599678
9	France	67871925	40086999	297154154	437816.01008075144	5.906271112834946e-07	1988.768242355816
10	Malta	520971	120564	2181540	418744.997322308	2.3142171061345064e-07	920.9358042177792



```
ORDER BY sum(new_cases)/population*0.000001 DESC
```

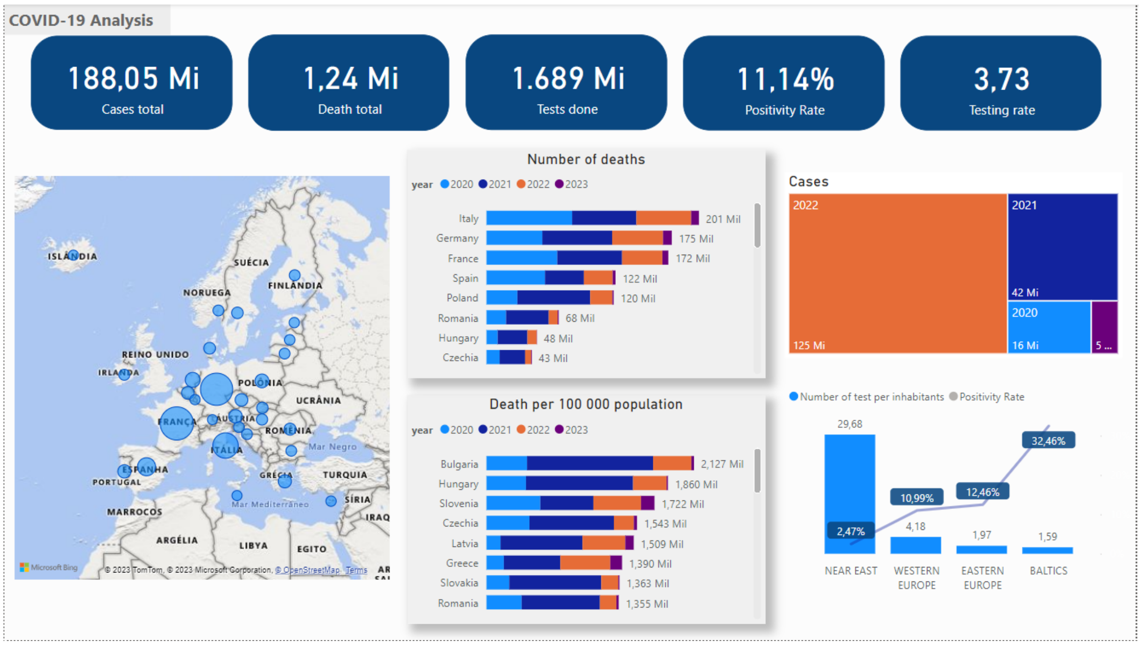
Data Output | Messages | Notifications

	country character varying (255)	population integer	total_cases double precision	total_tests double precision	testing_rate double precision	cases_rate double precision	positivity_rate double precision
1	Cyprus	904705	663816	26851438	2967977.185933534	7.337375166490735e-07	474.1528295037917
2	Austria	8978929	6084155	229496845	2555948.989016397	6.77603642928906e-07	476.14556612234384
3	Slovenia	2107180	1346138	20250864	961041.0121584297	6.388338917415693e-07	1719.1215955311347
4	Luxembourg	645397	385106	4605097	713529.347053054	5.966962970078882e-07	2978.077406221254
5	France	67871925	40086999	297154154	437816.01008075144	5.906271112834946e-07	1988.768242355816
6	Iceland	376248	209243	1698844	451522.4001190702	5.561305309264103e-07	2673.811755599678
7	Lithuania	2805998	1533622	4009698	142897.39336948923	5.465513517828594e-07	958.6904685206749
8	Portugal	10352042	5628104	51925	501.5918598475547	5.436709008715382e-07	136.98166176633384
9	Liechtenstein	39308	21310	116276	295807.46921746206	5.421288287371527e-07	3123.9582445403653
10	Denmark	5873420	3106460	122765270	2090183.7430321693	5.28901389650323e-07	1152.2666049830625

Considering, ‘Testing Rate’ is the total number of COVID-19 tests conducted in a country per 100 000 population and ‘Cases Rate’ the total number of confirmed COVID-19 cases in a country per 100 000 population.

It’s possible to highlight a correlation between these two metrics, among the countries with the highest testing rates, seven of them also feature on the list of countries with the highest cases rates. This connection suggests that when a country conducts a lot of tests, it's likely to find more cases.

Report



Between 2020 and 2023, Europe conducted nearly 4 COVID-19 tests per person, with an average positivity rate of 11%. As expected, the countries with the highest number of cases—France, Italy, and Germany—also reported the highest number of deaths.

2020 stands out as the year with the most deaths but the fewest cases (with an exception in 2023). This suggests that early identification of COVID-19 cases likely contributes to better recovery and reduced mortality rates. Also, 2020 was the year that the world was still learning with the virus and how to treat it, higher mortality rates were expected.

Some countries effectively managed the pandemic initially but faced significant challenges in 2021, leading to increased mortality rates. For instance, in 2020, Bulgaria and Hungary reported nearly identical death proportions, but in 2021, they struggled to treat patients effectively, resulting in higher mortality rates.

When examining COVID-19 testing, the Near East region notably stood out by conducting approximately 30 tests per person, surpassing the testing rates in the remainder of Europe. This region also reported the lowest positivity rates. This data may imply that in other European regions, people may primarily get tested when they exhibit symptoms, resulting in higher positivity rates.