Data Engineering Challenge

Ricardo González-Cienfuegos Mingo

Attached files:

DellTechnologiesDE.sql -> answers to questions 3 and 4

scriptCovidDataset.py -> answers to questions 1 and 2

countriesoftheworld.csv -> Dataset question 1 must be stored in the same folder as scriptCovidDataset.py

scriptCovidEnrichData.py-> answer to question 5

report.py -> answer to question 6

Excercise 1 & Exercise 2 -

Python script -> scriptCovidDataset.py

Excercise 3 -

SQL Scripts -> DellTechnologiesDE.sql

Excercise 4 -

SQL Scripts -> DellTechnologiesDE.sql

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

- **Query 1:** The first query involves sorting and grouping data based on COVID-19 cumulative counts and population, providing insights into the relationship between cases per million people. The query is well-optimized, with a reasonable execution time. To further improve it, indexing on relevant columns could be considered for larger datasets.
- **Query 2:** Query 2 also deals with COVID-19 data and population, aiming to find the top N countries with the highest cases per million. It performs sorting and joining operations efficiently. Improvements could be made by optimizing index usage and considering caching for frequently accessed data.
- **Query 3:** This query focuses on finding the countries with the highest GDP per capita. It utilizes sorting and joins to achieve its goal. While the query performs well, it might benefit from indexing the relevant columns in the dataset for faster retrieval.
- **Query 4:** Query 4 combines COVID-19 data with GDP per capita information to find the top N countries with the highest cases per million and the highest GDP per capita. It uses sorting, joins, and aggregation. Like the previous queries, indexing and caching strategies could enhance its performance, especially for large datasets.

Overall, the queries are well-structured and efficiently written, considering the specific data analysis tasks they aim to accomplish. However, depending on the size of the datasets, performance improvements can be achieved through index optimization, caching, and potentially partitioning or summarizing data to reduce query complexity. Regular monitoring and profiling of these queries can help identify specific optimization opportunities.

Q1

```
QUERY PLAN
Limit  (cost=300.47..300.47 rows=1 width=40) (actual time=7.666..7.666 rows=1 loops=1)
  ->  Sort  (cost=300.47..300.47 rows=1 width=40) (actual time=7.665..7.665 rows=1 loops=1)
        Sort Key: ((((covid_data.cumulative_count)::numeric / (countries_data.population)::numeric) *
'100000'::numeric)) DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  Nested Loop  (cost=0.00..300.46 rows=1 width=40) (actual time=0.077..7.658 rows=29
loops=1)
              Join Filter: (TRIM(BOTH FROM covid_data.country) = TRIM(BOTH FROM
countries_data.country))
              Rows Removed by Join Filter: 7008
              ->  Seq Scan on covid_data  (cost=0.00..288.21 rows=1 width=12) (actual time=0.068..6.430
rows=31 loops=1)
                    Filter: ((EXTRACT(year FROM to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(year
FROM to_date('31-07-2020'::text, 'DD-MM-YYYY'::text))) AND (EXTRACT(week FROM
to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(week FROM to_date('31-07-2020'::text, 'DD-
MM-YYYY'::text))))
                    Rows Removed by Filter: 5983
              ->  Seq Scan on countries_data  (cost=0.00..8.27 rows=227 width=18) (actual
time=0.000..0.006 rows=227 loops=31)
Planning Time: 0.142 ms
Execution Time: 7.682 ms
```

Indexing: Check if there are indexes on the columns used for joining (covid.country and cd.country) and the columns used in filtering to optimize data retrieval.

Date Manipulation: The query involves date manipulation using the EXTRACT function. Ensure that the date-related columns are of the correct data type, and consider indexing these columns if they are used frequently in filtering.

Q2

```
QUERY PLAN
Limit  (cost=300.47..300.47 rows=1 width=40) (actual time=9.240..9.242 rows=10 loops=1)
  ->  Sort  (cost=300.47..300.47 rows=1 width=40) (actual time=9.240..9.240 rows=10 loops=1)
      Sort Key: ((((covid_data.cumulative_count)::numeric / (countries_data.population)::numeric) *
'100000'::numeric))
      Sort Method: top-N heapsort  Memory: 26kB
      ->  Nested Loop  (cost=0.00..300.46 rows=1 width=40) (actual time=0.097..9.216 rows=29
loops=1)
            Join Filter: (TRIM(BOTH FROM covid_data.country) = TRIM(BOTH FROM
countries_data.country))
            Rows Removed by Join Filter: 7008
            ->  Seq Scan on covid_data  (cost=0.00..288.21 rows=1 width=12) (actual time=0.083..7.688
rows=31 loops=1)
                  Filter: ((EXTRACT(year FROM to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(year
FROM to_date('31-07-2020'::text, 'DD-MM-YYYY'::text))) AND (EXTRACT(week FROM
to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(week FROM to_date('31-07-2020'::text, 'DD-
MM-YYYY'::text))))
                  Rows Removed by Filter: 5983
            ->  Seq Scan on countries_data  (cost=0.00..8.27 rows=227 width=18) (actual
time=0.001..0.008 rows=227 loops=31)
Planning Time: 0.108 ms
Execution Time: 9.256 ms
```

Indexing: Ensure that appropriate indexes are in place on the columns used for joining and filtering (country columns and year_week column). Indexes can significantly improve query performance, especially for large datasets.

Q3

```
QUERY PLAN
Limit  (cost=1053.09..1053.10 rows=3 width=12) (actual time=6.389..6.392 rows=10 loops=1)
 ->  Sort  (cost=1053.09..1053.10 rows=3 width=12) (actual time=6.388..6.390 rows=10 loops=1)
      Sort Key: covid.cumulative_count DESC
      Sort Method: quicksort  Memory: 25kB
      ->  Hash Join  (cost=884.37..1053.07 rows=3 width=12) (actual time=5.774..6.373 rows=20
loops=1)
          Hash Cond: (TRIM(BOTH FROM covid.country) = TRIM(BOTH FROM richest.country))
          ->  Hash Join  (cost=166.53..335.00 rows=31 width=12) (actual time=3.074..3.658 rows=31
loops=1)
              Hash Cond: ((covid.country = covid_data.country) AND ((covid.year_week)::text =
(max((covid_data.year_week)::text))))
              ->  Seq Scan on covid_data covid  (cost=0.00..136.63 rows=6063 width=20) (actual
time=0.006..0.170 rows=6014 loops=1)
              ->  Hash  (cost=166.06..166.06 rows=31 width=40) (actual time=3.045..3.045 rows=31
loops=1)
                  Buckets: 1024  Batches: 1  Memory Usage: 10kB
                  ->  HashAggregate  (cost=165.75..166.06 rows=31 width=40) (actual
time=3.014..3.016 rows=31 loops=1)
                      Group Key: covid_data.country
                      Batches: 1  Memory Usage: 24kB
                      ->  Seq Scan on covid_data  (cost=0.00..136.63 rows=5824 width=16) (actual
time=0.005..0.690 rows=5772 loops=1)
                          Filter: (cumulative_count IS NOT NULL)
                          Rows Removed by Filter: 242
          ->  Hash  (cost=717.60..717.60 rows=20 width=10) (actual time=2.693..2.694 rows=20
loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 9kB
              ->  Subquery Scan on richest  (cost=717.35..717.60 rows=20 width=10) (actual
time=2.689..2.691 rows=20 loops=1)
                  ->  Limit  (cost=717.35..717.40 rows=20 width=15) (actual time=2.688..2.690 rows=20
loops=1)
                      ->  Sort  (cost=717.35..717.91 rows=227 width=15) (actual time=2.688..2.689
rows=20 loops=1)
                          Sort Key: cd.gdp_per_capita DESC
                          Sort Method: quicksort  Memory: 26kB
                          ->  HashAggregate  (cost=709.04..711.31 rows=227 width=15) (actual
time=2.676..2.678 rows=29 loops=1)
                              Group Key: cd.gdp_per_capita, cd.country
                              Batches: 1  Memory Usage: 40kB
```

```
                        ->  Merge Join  (cost=534.72..674.63 rows=6882 width=15) (actual
time=1.560..1.998 rows=5626 loops=1)
                              Merge Cond: ((TRIM(BOTH FROM cd.country)) = (TRIM(BOTH FROM
covid_1.country)))
                              ->  Sort  (cost=17.15..17.72 rows=227 width=15) (actual
time=0.218..0.221 rows=197 loops=1)
                                    Sort Key: (TRIM(BOTH FROM cd.country))
                                    Sort Method: quicksort  Memory: 36kB
                                    ->  Seq Scan on countries_data cd  (cost=0.00..8.27 rows=227
width=15) (actual time=0.008..0.041 rows=227 loops=1)
                              ->  Sort  (cost=517.56..532.72 rows=6063 width=8) (actual
time=1.339..1.436 rows=6014 loops=1)
                                    Sort Key: (TRIM(BOTH FROM covid_1.country))
                                    Sort Method: quicksort  Memory: 444kB
                                    ->  Seq Scan on covid_data covid_1  (cost=0.00..136.63 rows=6063
width=8) (actual time=0.002..0.487 rows=6014 loops=1)
Planning Time: 0.316 ms
Execution Time: 6.485 ms
```

Indexing: Ensure that appropriate indexes are in place on the columns used for joining and filtering (country columns). This can significantly improve query performance.

Q4

```
QUERY PLAN
Sort  (cost=300.50..300.51 rows=1 width=96) (actual time=11.764..11.764 rows=4 loops=1)
  Sort Key: (((sum(covid.cumulative_count))::numeric / (sum(cd.population) / '1000000'::numeric))) DESC
  Sort Method: quicksort  Memory: 25kB
  ->  GroupAggregate  (cost=300.46..300.49 rows=1 width=96) (actual time=11.753..11.760 rows=4 loops=1)
        Group Key: cd.region
        ->  Sort  (cost=300.46..300.46 rows=1 width=50) (actual time=11.745..11.746 rows=29 loops=1)
              Sort Key: cd.region
              Sort Method: quicksort  Memory: 27kB
              ->  Nested Loop  (cost=0.00..300.45 rows=1 width=50) (actual time=0.103..11.724 rows=29 loops=1)
                    Join Filter: (TRIM(BOTH FROM covid.country) = TRIM(BOTH FROM cd.country))
                    Rows Removed by Join Filter: 7008
                    ->  Seq Scan on covid_data covid  (cost=0.00..288.21 rows=1 width=12) (actual time=0.091..9.872 rows=31 loops=1)
                          Filter: ((EXTRACT(year FROM to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(year FROM to_date('31-07-2020'::text, 'DD-MM-YYYY'::text))) AND (EXTRACT(week FROM to_date((year_week)::text, 'IYYY-IW'::text)) = EXTRACT(week FROM to_date('31-07-2020'::text, 'DD-MM-YYYY'::text))))
                          Rows Removed by Filter: 5983
                    ->  Seq Scan on countries_data cd  (cost=0.00..8.27 rows=227 width=56) (actual time=0.001..0.010 rows=227 loops=31)
Planning Time: 0.112 ms
Execution Time: 11.787 ms
```

Indexing: Ensure that appropriate indexes are in place on the columns used for joining and filtering (country columns and year_week column). Indexes can significantly improve query performance, especially for large datasets.

Excercise 5 -

Python script -> scriptCovidEnrichData.py

https://opendata.ecdc.europa.eu/covid19/hospitalicuadmissionrates/json/

The inclusion of hospital capacity data is essential for a comprehensive analysis of the impact of COVID-19 on different countries. This data category offers valuable insights into the healthcare infrastructure and the ability of a country's healthcare system to respond to a crisis, such as a pandemic. Several key reasons support the decision to incorporate hospital capacity data into our dataset:

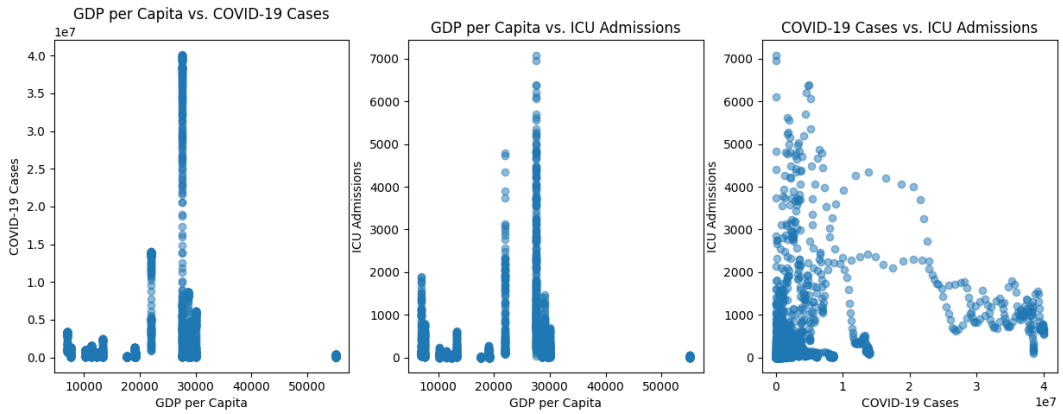
- Healthcare System Resilience: Hospital capacity information allows us to assess a country's preparedness and resilience in dealing with health emergencies. By comparing hospital bed availability, intensive care unit (ICU) capacity, and other critical resources, we can gain a deeper understanding of a nation's ability to cope with surges in COVID-19 cases.
- Economic Disparities: The economic capacity of a country plays a significant role in determining the adequacy of its healthcare system. Wealthier nations often have more robust healthcare infrastructure, including a higher number of hospital beds, advanced medical equipment, and a larger healthcare workforce. By correlating hospital capacity with a country's economic indicators, we can explore the relationship between wealth and healthcare resilience.

In summary, the addition of hospital capacity data is crucial for a holistic analysis of COVID-19's impact on different countries. It allows us to investigate the correlation between a nation's economic strength and the resilience of its healthcare system, providing valuable insights for public health research, policy formulation, and crisis management. By considering this dimension, we can gain a more nuanced understanding of the global response to the pandemic and draw meaningful conclusions about healthcare disparities among nations.

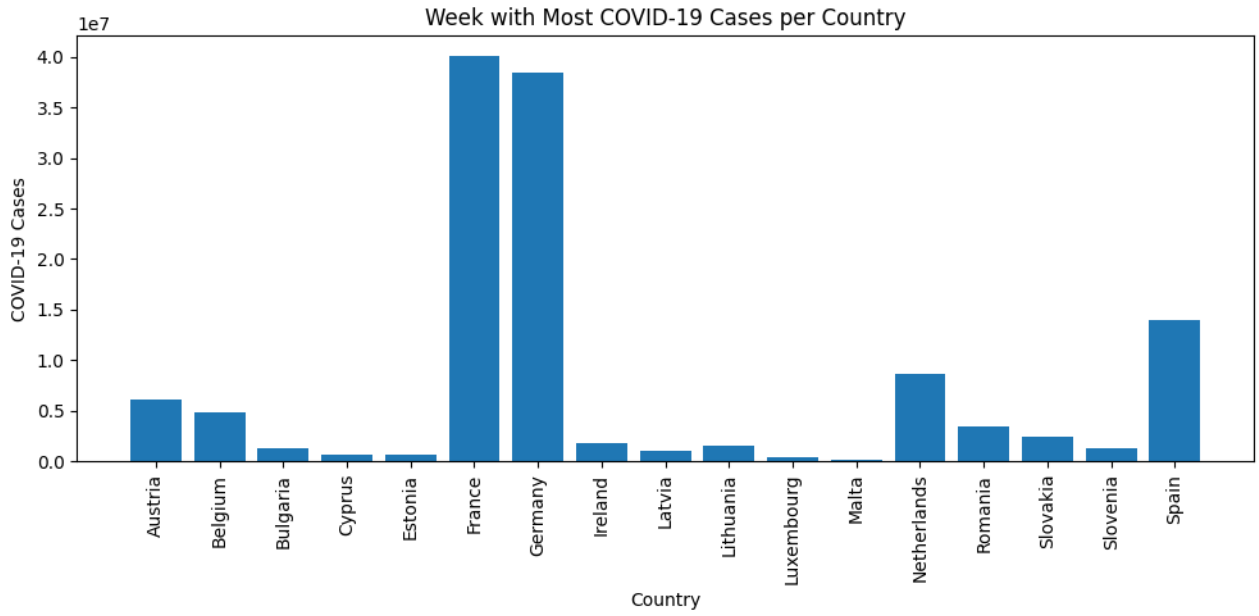
Excercise 6 -

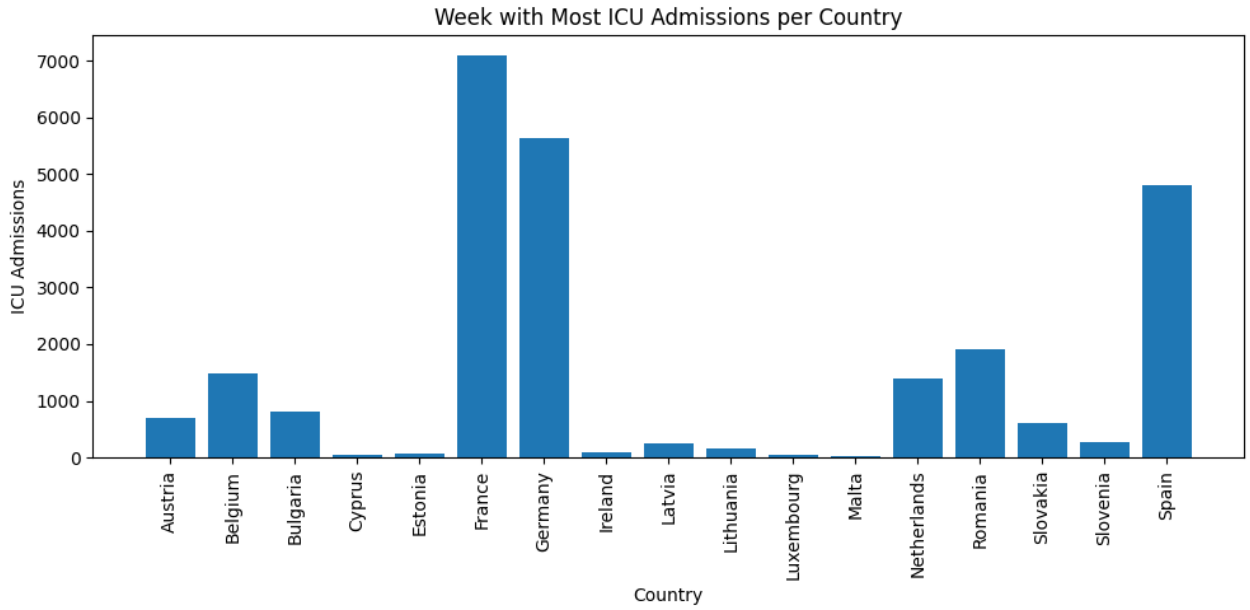
Python script -> report.py

Firstly, we have three figures that depict the relationships between GDP per capita vs. COVID-19 cases, GDP per capita vs. ICU admissions, and COVID-19 cases vs. ICU admissions.
From these graphs, we can draw the following conclusions:



Countries with higher GDP per capita were more severely affected by COVID-19, but they also exhibited greater capacity to handle patients in the ICU. In general, there appears to be an inverse relationship between the number of COVID-19 cases and the capacity to admit patients to the ICU. In other words, the more cases there were, the less capacity there seemed to be for ICU admissions.





Indeed, based on the analysis of the graphs, we can make the following observations:

- Spain, France, and Germany: These countries were among the most affected by COVID-19 in terms of absolute cases. However, they also demonstrated a high capacity to admit patients to the ICU. This suggests that despite the large number of cases, they had relatively robust healthcare infrastructure for handling critical cases.

- Romania, Belgium, and Bulgaria: These countries displayed a remarkable response. Despite having a significant number of COVID-19 cases, they provided a substantial number of ICU beds. This indicates that they were well-prepared to handle critical cases and had the infrastructure in place to support patients in need of intensive care.

These observations highlight the varying responses of different countries to the COVID-19 pandemic, with some nations effectively managing both a high caseload and ICU capacity, while others excelled in providing ICU care relative to their caseload. It underscores the importance of healthcare system resilience and preparedness during health crises.