Bemnet Shibeshi
Yeajee Lee
Sarah Nyaanga

The Effect of COVID on Soccer Around the World

Goals

In this project, our primary goal was to create a data-driven solution that analyzes and visualizes football and Covid-related statistics. The project aimed to leverage data from 3 APIs, focusing on fetching competition, team, country, and COVID case details from global football leagues and COVID reportings. A key objective was to store the data in a SQLite database, enabling efficient querying and manipulation. Additionally, the project sought to provide insights into global soccer trends by visualizing team distributions across countries and COVID-19 case data for top-ranked countries. The integration of API data with database management allowed for dynamic updates and streamlined visual outputs, such as pie charts and bar charts. We were successfully able to analyze the impact of COVID-19 on soccer in the year 2020 by integrating data from three APIs and storing it in a SQLite database. Our visualizations showed covid was a big disruption for soccer in the year 2020.

Challenges we Encountered:

Throughout this soccer coding project, one of the key challenges we encountered was ensuring that we could consistently retrieve at least 100 rows of data from each API. Initially, our project idea was focused on comparing menu items and their nutritional values. However, as we explored various APIs, we faced difficulties with run limits—many free APIs had very restrictive data fetching limits, making it hard to obtain a sufficient amount of data for analysis. This led us to pivot from our original concept to something more feasible: leveraging soccer data, which allowed us to access larger and more reliable datasets. We also ran into issues with finding free APIs that met our needs, which required us to experiment with multiple API sources before identifying the right ones that provided the necessary data volume and consistency. Despite these challenges, the project ultimately succeeded in pulling meaningful insights from football-related data, thanks to the integration of the Football-Data API and our database management approach.

Resources:

In our project, we utilized three APIs, VSCode, and GitHub to collect and analyze data. VSCode and GitHub were essential tools for writing and storing our Python scripts, managing our databases, and generating visualizations with Matplotlib. The three APIs we used were Football-Data (https://www.football-data.org/documentation/quickstart), Free API Live Football Data (https://rapidapi.com/Creativesdev/api/free-api-live-football-data), and the Covid-19 API (https://github.com/STATWORX/covid-19-api). We relied on Football-Data to gather information about global football teams, extracting data for 194 unique teams along with their respective country IDs. Next, we used the Free API Live Football Data to retrieve data on canceled games worldwide, filtering by the teams found in our previous dataset. Finally, the Covid-19 API provided us with the number of Covid-19 cases per country in 2020, corresponding to the countries associated with the teams in our database.

Run the Code:

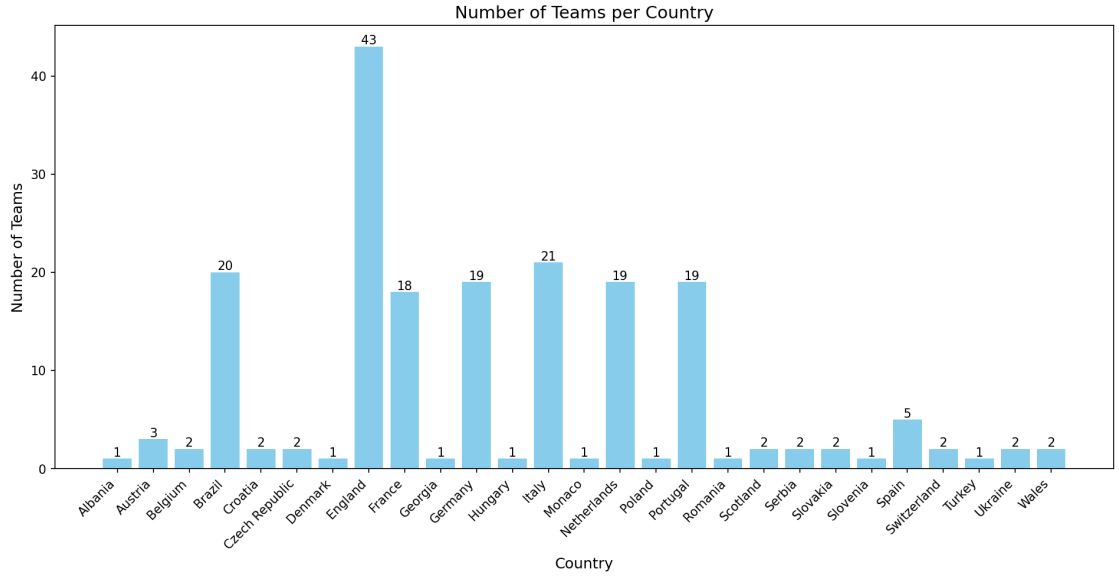
1) First, run team.py file
    a) This API has 10 calls/minute maximum
2) Next, run cases.py file
3) Next, run games.py file with all the functions except average_canceled_games_per_month and canceled_games_team uncommented in the main function
4) Go to the main function of games.py and uncomment average_canceled_games_per_month and comment out the rest of the function calls
    a) This must be done so that the API is not being called too many times because we have a 1000 calls/month and so that the correct visualizations come up
5) Go to the main function of games.py and comment out average_canceled_games_per_month and uncomment canceled_games_country) and run
6) Go to the main function of games.py and comment out canceled_games_country (con, curr) and uncomment canceled_games_team and run

Visualizations

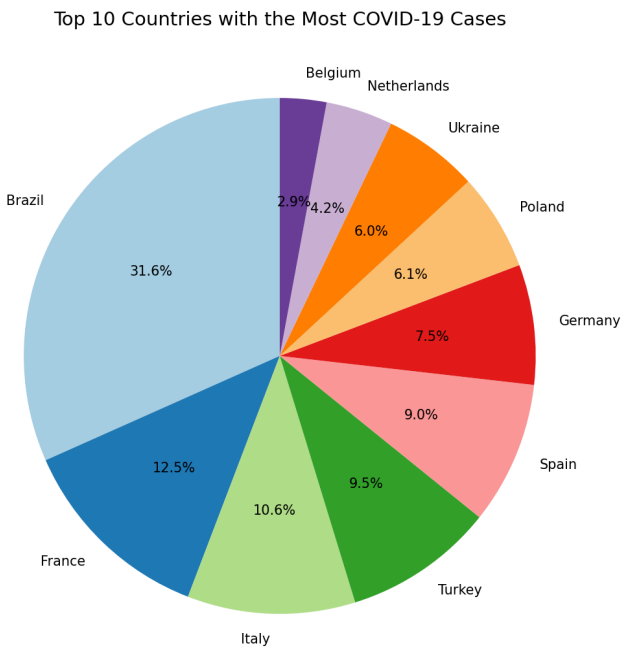
For our visualizations we wanted to highlight the correlation between the amount of soccer games cancelled and covid cases in the given country.

Number of Teams per Country (Bar Chart): This bar chart illustrates the number of teams in different countries, revealing a stark contrast in team distribution across nations. England leads

with a significant number of teams, 43 in total, far surpassing other countries. Denmark follows with 20 teams, while countries like France, Germany, and Italy each have around 18-21 teams. On the other end of the spectrum, countries such as Albania, Belgium, and Wales show very few teams, with only one or two represented. This chart could reflect the popularity and structure of the sport within these countries, highlighting that certain countries have more active participation or organizational support for the teams compared to others.



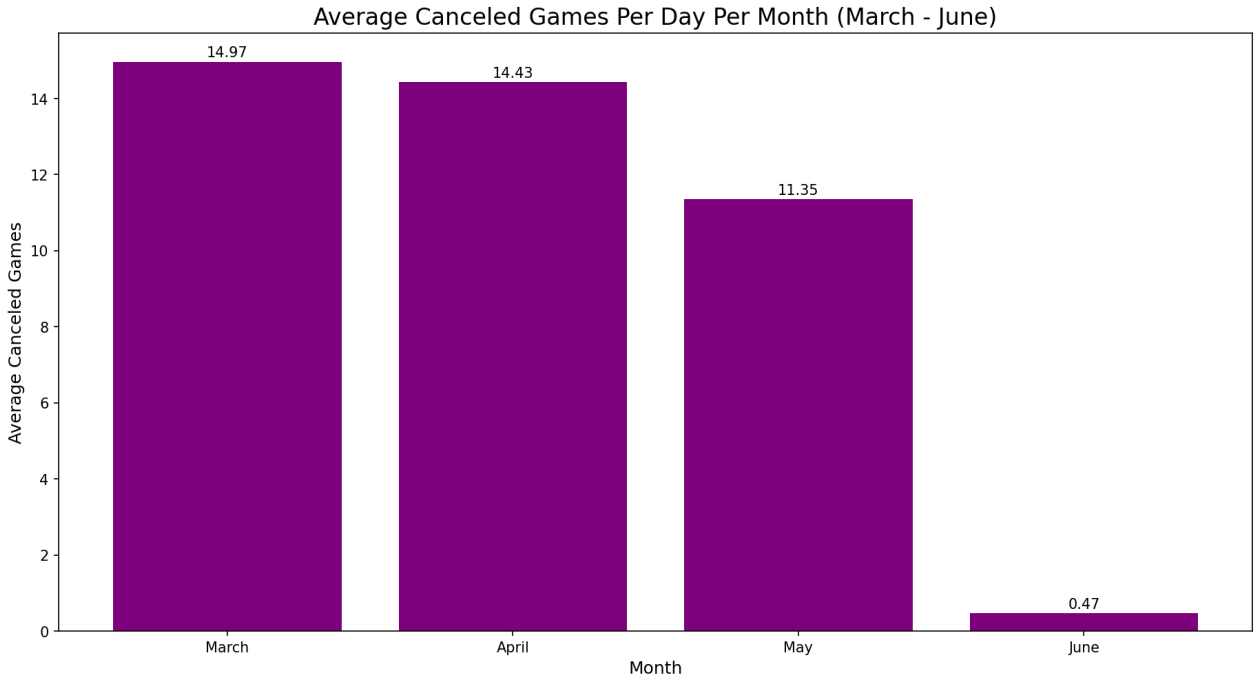
Top 10 Countries with the Most COVID-19 Cases (Pie Chart): The pie chart shows the distribution of COVID-19 cases among the top 10 countries. Brazil is the most affected, accounting for 31.6% of the total cases. It is followed by France (12.5%) and Italy (10.6%). Other significant contributors include Germany, Spain, and Turkey, each with cases ranging from 7.5% to 9.5% of the total. Smaller portions of the chart are occupied by countries like Belgium, Ukraine, and the Netherlands. This chart provides a clear view of the countries with the highest case loads, emphasizing the impact of the pandemic in certain regions while showing how some countries have managed to limit the spread.



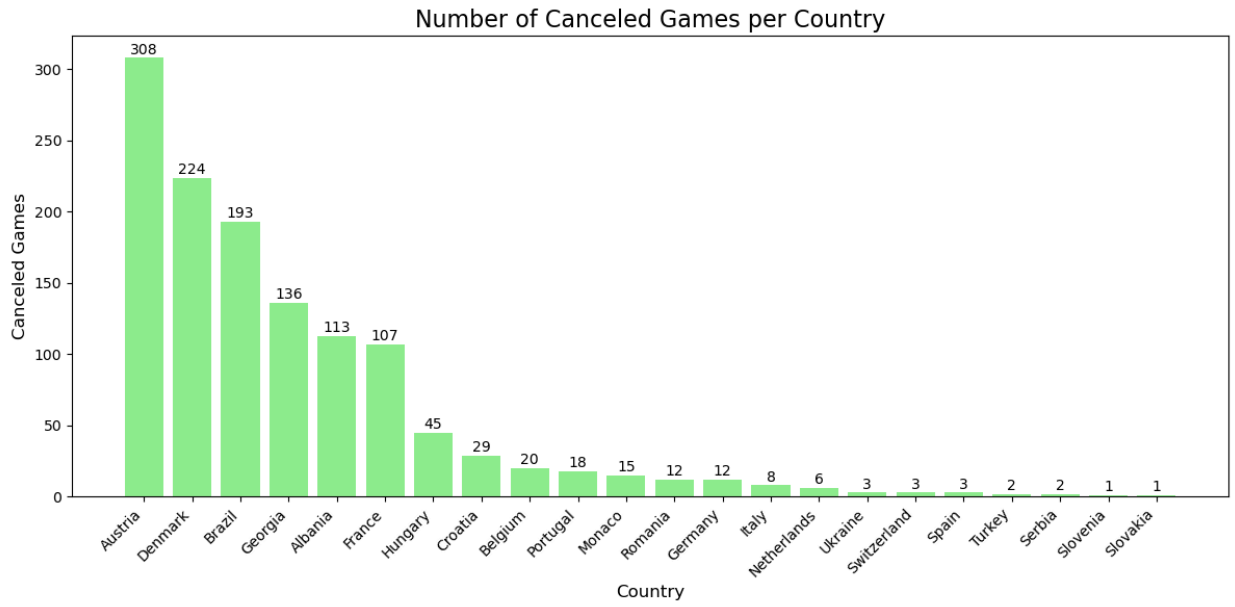
Average Canceled Games Per Day Per Month (March - June):

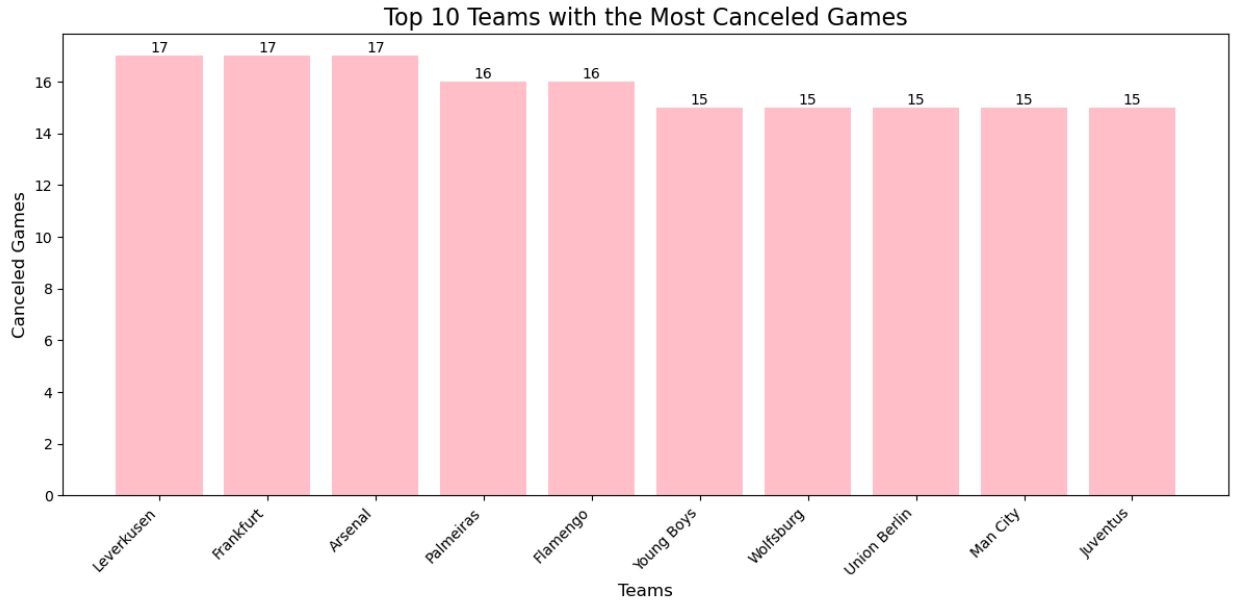
This graph shows a significant decline in canceled games over four months. March and April have the highest averages, with **14.97** and **14.43** canceled games per day, respectively, indicating persistent issues during these months. May sees a notable improvement with cancellations dropping to **11.35** per day, while June experiences a dramatic decrease to just **0.47** per day. This downward trend suggests improvements in scheduling, logistics, or external factors, leading to a

stabilization and near elimination of cancellations by June.



Number of Cancelled Games per Country & Top 10 Teams with the Most Canceled Games (Bar Chart): These bar charts show the number of cancelled games per country based on the teams within that country and the top 10 teams with the most canceled games. In the Number of Cancelled Games per Country graph we see the country with the most cancelled games is Italy with around 15900 cancellations. England comes in close second followed by Germany, France and Brazil. The first graph gives us a better understanding of which countries' teams were most affected by the cancellation of games. The top 10 teams with the most canceled games graph is showing how disruptions affected various teams. Real Madrid stands out with the most canceled games, exceeding 1000. Teams such as Bologna, Cagliari, and Atalanta follow closely behind, each having a similar number of canceled games. The chart reflects how certain teams, likely from leagues with high fixture congestion or logistical challenges, experienced more interruptions, perhaps due to factors like the COVID-19 pandemic or other unexpected circumstances affecting the schedule.





Calculation:

We calculated the average canceled games per day per Month

Occurs in file: games.py in function: average_canceled_games_per_month

average_canceled_games.txt

```
avg_per_month = []

for month in months:
    month_data = df[df['Month'] == month]

    canceled_per_day = month_data.groupby('Day').size()

    avg_canceled = canceled_per_day.sum() / month_days[month]
    avg_per_month.append(avg_canceled)
```

average_canceled_games.txt

```
Average Canceled Games Per Day (March – June)
March: 14.97 canceled games per day
April: 14.43 canceled games per day
May: 11.35 canceled games per day
June: 0.47 canceled games per day

```

Documentation for Code

Team.py

- def get_comp_id():
    - The get_comp_id() function fetches competition IDs by sending a GET request to the football-data API using an API key. It does not take any inputs, and if the request is successful, it returns a list of competition IDs. If the request fails, it prints an error message and returns None.
- def get_comp_teams(comp_ids):
    - The get_comp_teams() function takes a list of competition IDs (comp_ids) as input and fetches team information for each competition from the football-data API. It organizes the data into a dictionary where the keys are country names and the values are lists of team short names. If a request for a specific competition fails, the function prints an error message but continues processing the remaining IDs.
- def set_up_database(db_name):

    - The set_up_database() function sets up a connection to a SQLite database with the name provided as the input parameter db_name. It enables foreign key support and returns a tuple containing a cursor object (cur) for executing queries and a connection object (conn) for committing changes. This function ensures that the database is ready for further operations.
- def set_up_countryid_table(data, cur, conn):
    - The set_up_countryid_table() function creates a table called "Countries" in the database and populates it with country names using the keys from the input dictionary data. It drops the table if it already exists and uses the provided cursor (cur) and connection (conn) objects to execute SQL commands and commit changes. This ensures that the database has an updated list of countries.
- def set_up_teams_table(data, cur, conn):
    - The set_up_teams_table() function creates a "Teams" table in the database and inserts team names along with their corresponding country IDs. It takes a dictionary (data) as input, where country names are the keys and lists of team names are the values. Using the cursor (cur) and connection (conn), the function ensures that the team data is inserted while maintaining relationships with the "Countries" table.
- def num_teams_country(cur, conn):
    - The num_teams_country() function queries the database to calculate the number of teams per country by joining the "Teams" and "Countries" tables. It uses the cursor (cur) and connection (conn) to fetch and process the data, then visualizes the results as a bar chart. The function closes the database connection after displaying the chart.

Cases.py

- def get_df():
    - The get_df() function fetches COVID-19 data from the Statworx API by sending a POST request with a payload containing 'code': 'ALL'. If the request is successful (status code 200), it parses the JSON response and converts it into a DataFrame. If the request fails, it prints an error message with the status code and reason and returns an empty DataFrame.
- def set_up_database(db_name):
    - The set_up_database() function sets up a connection to a SQLite database file, specified by the input db_name. It enables foreign key constraints using a PRAGMA

statement and returns a tuple containing the cursor and connection objects for executing database operations.

- def update_country_codes(df, cur, conn):
    - The update_country_codes() function updates the "Countries" table in the database with country codes extracted from the input DataFrame df. It ensures that the country_code column exists in the table and then iterates through the unique country-code pairs to update the table. The changes are committed using the provided cursor (cur) and connection (conn) objects.
- def get_matched_data(df, cur):
    - The get_matched_data() function filters the input DataFrame df to include only the countries whose codes exist in the database. It retrieves all country codes from the "Countries" table, checks for matches in the DataFrame, and returns a filtered DataFrame containing the matched data.
- def get_month(matched_df):
    - The get_month() function processes the input DataFrame matched_df to extract COVID-19 data for the last day of each month in 2020. It filters data for the year 2020, groups it by month and country, and selects the last available entry for each group. The resulting DataFrame contains the last day of the month, country name, country code, and case counts.
- def insert_df_into_db(df, cur, conn):
    - The insert_df_into_db() function inserts data from the input DataFrame df into two tables in the database: "Countries" and "Cases." It first creates the "Countries" table and populates it with unique country names. It then creates the "Cases" table and inserts case data while referencing the country IDs from the "Countries" table using foreign keys. The changes are committed to ensure data integrity.
- def visualize_cases(df):
    - The visualize_cases() function visualizes COVID-19 case distribution for the top 10 countries with the most cases. It calculates the total cases per country, sorts them in descending order, and selects the top 10. Using a pie chart, it displays the distribution with percentage labels and a title. If the DataFrame is empty, the function prints an error message instead.

Games.py

- def set_up_database(db_name):

    - This function establishes a connection to an SQLite database and ensures foreign key constraints are enabled. It takes a string db_name as input, which represents the name of the database file, and returns a tuple containing the database cursor for executing queries and the connection object for committing changes.
- def get_canceled_games(cur):
    - This function fetches canceled football games between March 1, 2020, and June 1, 2020, from a remote API and matches them with teams in the database. It takes a database cursor cur as input to execute SQL queries and returns a list of dictionaries with information about canceled games, including the date, team, and country_id.
- def insert_to_db(canceled_data, cur, conn):
    - This function creates a table called Games_Canceled and inserts data about canceled games into the database. It takes a list of dictionaries canceled_data containing game information, a database cursor cur, and a connection object conn to commit changes, with no return value.
- def canceled_games_country(cur, conn):
    - This function visualizes the number of canceled games per country by generating a bar chart. It takes a database cursor cur to execute queries and a connection object conn, and it displays the visualization while closing the database connection afterward.
- def average_canceled_games_per_month(cur, conn):
    - This function calculates the average number of canceled games per day for each month between March and June 2020 and generates a bar chart for visualization. It takes a database cursor cur and a connection object conn as inputs, writes the results to a text file, and returns a list of average canceled games for each month.
- def canceled_games_team(cur, conn):
    - This function identifies and visualizes the top 10 teams with the most canceled games using a bar chart. It takes a database cursor cur and a connection object conn, retrieves the necessary data, and closes the connection after generating the chart.