A

Course End Project Report on

# FIFA WORLD CUP 2022

Is submitted in partial fulfillment of the Requirements for the Award of CIE of

DATA ANALYSIS AND VISUALIZATION-22ADE01

in

B.E, IV-SEM, INFORMATION TECHNOLOGY

Submitted by

Abdul Raheem, 160123737029

COURSE TAUGHT BY:

Dr Ramakrishna Kolikipogu Professor, Dept of IT.

DEPARTMENT OF INFORMATION TECHNOLOGY CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)

(Affiliated to Osmania University;Accredited by NBA,NAAC,ISO) kokapet(V),GANDIPET(M),HYDERABAD-500075

Website:www.cbit.ac.in

2024-2025

# CERTIFICATE

This is to certify that the course end project work entitled **”FIFA World Cup 2022”** is submitted by **Abdul Raheem(160123737029)** in partial fulfillment of the requirements for the award of CIE Marks of **DATA ANALYSIS AND VISUALIZATION (22ADE01)** of **B.E, IV-SEM, INFORMATION TECHNOLOGY** to CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A) affiliated to OSMANIA UNIVERSITY, Hyderabad is a record of bonofide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

**Signature of Course Faculty**
**Dr Ramakrishna Kolikipogu**
**Professor of IT**

Kokapet(V),Gandipet(M),Ranga Reddy (Dist.)–500075, Hyderabad, T.S.

# Acknowledgement

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We wish to express our deep sense of gratitude to **Dr Ramakrishna Kolikipogu, Professor of IT** for his able guidance and useful suggestions, which helped us in completing the Course End Project in time.

We are particularly thankful to **HoD, Principal and Management**, for their support and encouragement, which helped us to mould our project into a successful one.

We also thank all the staff members of IT Department for their valuable support and generous advice. Finally thanks to all our friends and family members for their continuous support and enthusiastic help.

**Abdul Raheem, 160123737029**

# Abstract

The project aims to analyze the performance of a selected national team in FIFA World Cup matches over the years, focusing on goal-scoring patterns, match distributions, and performance trends. Using Python libraries such as Pandas, Matplotlib, and Seaborn, we perform data manipulation and visualization tasks to explore various aspects of the team's participation. The analysis includes identifying high-scoring matches, understanding the distribution of goals, and comparing performance with opponents, offering a comprehensive view of the team's historical performance and potential areas for improvement.

.

# Table of Contents

# List of Figures

# Abbreviations

| Abbreviation | Description |
|---|---|
| DAV | Data Analysis and Visualization |
| SB | Sea Born |
| PD | pandas |
| CSV | Comma Separated Value |
| HIST | Histogram |

# CHAPTER 1
# Introduction

## 1.1 Definition of Problem

The problem is to analyze historical FIFA World Cup match data for a selected national team to uncover patterns in their performance, focusing on goal-scoring, match frequency, and comparisons with opponents.

## 1.1 Objectives and Outcomes

**Objectives**: Data Exploration: Conduct thorough exploration of the FIFA World cup 2022 dataset to understand its structure, features, and potential insights. Insight Generation: Identify key patterns, trends, and relationships within the dataset to uncover valuable insights about the World Cup. Visualization: Utilize data visualization techniques to present findings in a clear, concise, and visually appealing manner. Identify matches where the team scored highly, such as more than 5 goals, to understand exceptional performances. Analyze the distribution of matches per year to assess participation frequency, aligning with World Cup cycles. Explore goal-scoring trends, comparing the team's goals with opponents and tracking changes over years. Visualize data to reveal insights, such as trends in total goals per year and goal distributions, aiding strategic planning

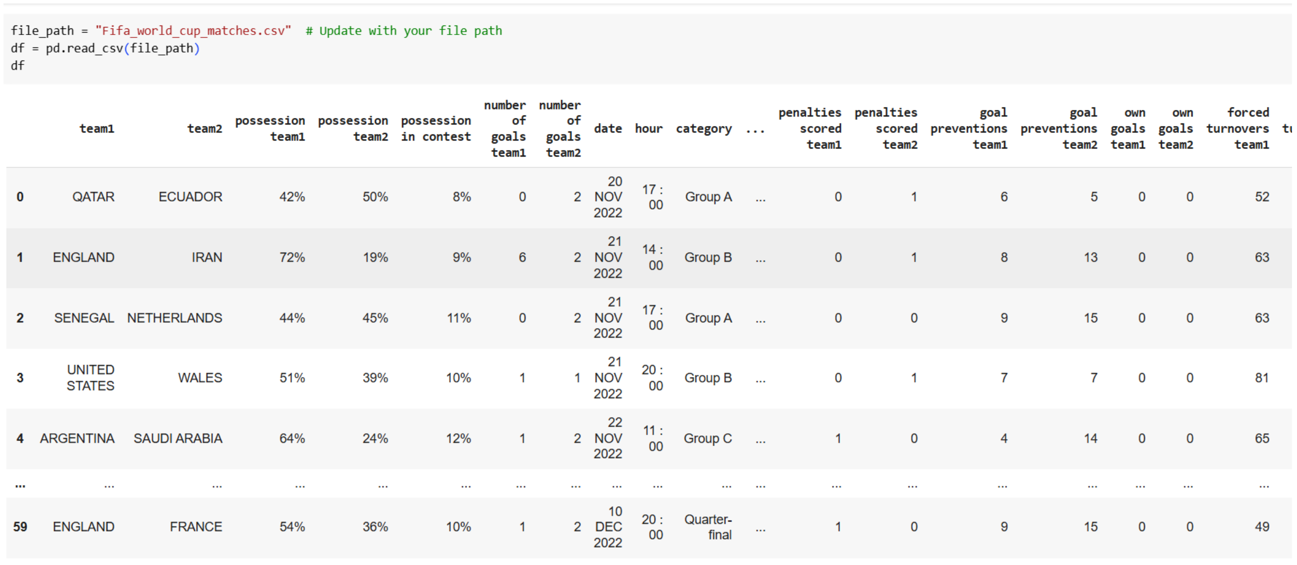
**Outcomes**: Performance Analysis: A detailed report on the team's historical performance, highlighting strengths like high-scoring matches (e.g., sorting by 'number of goals team1' to find top 10 games) and areas for improvement, such as years with low goal counts. This aligns with understanding exceptional games like Brazil's 7-1 win over Germany in 2014 . Insights into Participation Patterns: A visualization of match distribution per year, showing participation frequency and potential increases due to tournament expansions. Goal-Scoring Trends and

Comparisons: Visualizations like scatter plots (goals comparison with opponents), line plots (goals trend over years), and boxplots (goal distribution comparisons) reveal scoring patterns. Data-Driven Recommendations: Based on the analysis, recommendations for future strategies, such as focusing on offensive plays in high-scoring years or improving consistency in goal distribution, enhancing tournament performance. This leverages historical data to inform coaching decisions, aligning with trends like increased goals in winning years

# CHAPTER 2
# Methodology

## 3.1 Data collection and Dataset description

The dataset we're using, sourced from GitHub, contains valuable information about online food delivery. It covers details such as delivery personnel, weather, and order specifics, forming the basis of our analysis on delivery times. Each delivery record is unique and provides insights into different parts of the delivery process. This includes details about delivery staff, restaurant and delivery locations, order times, and weather conditions. Additionally, it includes information about delivery vehicles, order types, transportation methods, multiple deliveries, and festivals impacting deliveries. It also specifies the city for each delivery. Importantly, it logs delivery times, reflecting how well the service is working and how satisfied customers are. With this dataset, we aim to thoroughly study how these factors interact, revealing what affects delivery times and service quality in online food delivery.



```
file_path = "Fifa_world_cup_matches.csv"  # Update with your file path
df = pd.read_csv(file_path)
df
```

| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | QATAR | ECUADOR | 42% | 50% | 8% | 0 | 2 | 20 NOV 2022 | 17 : 00 | Group A | ... | 0 | 1 | 6 | 5 | 0 | 0 | 52 |
| **1** | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 |
| **2** | SENEGAL | NETHERLANDS | 44% | 45% | 11% | 0 | 2 | 21 NOV 2022 | 17 : 00 | Group A | ... | 0 | 0 | 9 | 15 | 0 | 0 | 63 |
| **3** | UNITED STATES | WALES | 51% | 39% | 10% | 1 | 1 | 21 NOV 2022 | 20 : 00 | Group B | ... | 0 | 1 | 7 | 7 | 0 | 0 | 81 |
| **4** | ARGENTINA | SAUDI ARABIA | 64% | 24% | 12% | 1 | 2 | 22 NOV 2022 | 11 : 00 | Group C | ... | 1 | 0 | 4 | 14 | 0 | 0 | 65 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **59** | ENGLAND | FRANCE | 54% | 36% | 10% | 1 | 2 | 10 DEC 2022 | 20 : 00 | Quarter-final | ... | 1 | 0 | 9 | 15 | 0 | 0 | 49 |

**Figure 3.1:** Dataset

## 3.2 Data cleaning and preprocessing

We delve into the critical aspects of data cleaning and preprocessing, laying the groundwork for the subsequent analysis. Data cleaning is a crucial step in the data analysis process, ensuring that our dataset is accurate, reliable, and free from errors. In this stage, we meticulously sift through the data to identify and rectify any inconsistencies, inaccuracies, or missing values that could potentially skew our analysis.

The data cleaning process for our dataset was carried out in a structured manner to ensure the data's quality and reliability for subsequent analysis. The first step involved removing unnecessary columns. This began with an initial review of the dataset to understand the significance of each column. Any columns deemed irrelevant to the analysis or redundant, that did not affect delivery time, were identified and removed using the Pandas drop function. This streamlined the dataset, making it more focused and manageable.

Next, we addressed missing values, which is crucial to prevent biases or inaccuracies in our analysis. Using Pandas' isnull functions, we identified columns with missing data. Depending on the extent and nature of these missing values, we employed different strategies. For instance, if a column had a few missing values, we filled them with the mean and mode of that column using Pandas' fillna function. However, if a column had a significant portion of missing values and was not critical to the analysis, we considered dropping it entirely using the dropna method. This ensured that our dataset maintained its integrity and represented a complete set of information.

Handling duplicate values was the next step, aimed at ensuring that each record in the dataset was unique. We used Pandas' duplicated function to identify any duplicate rows. Once identified, these duplicates were removed with the drop_duplicates method, ensuring that each entry in the dataset was distinct and avoiding any skewed analysis results from repetitive data.

Subsequently, we addressed data types to ensure each column was in the appropriate format for accurate analysis. We used Pandas' dtypes attribute to review the data types and converted columns as needed with the astype method. For example, columns containing delivery times stored as strings were converted to numerical types or datetime objects to facilitate proper time series analysis. This step was crucial for enabling accurate computations and comparisons within the dataset.

Outliers, which could distort the analysis, were then handled. We detected outliers using various statistical methods . Values falling outside the range ,depending on the context, we either removed these outliers or capped them to a specified limit using Pandas and NumPy functions, ensuring they did not unduly influence the analysis.

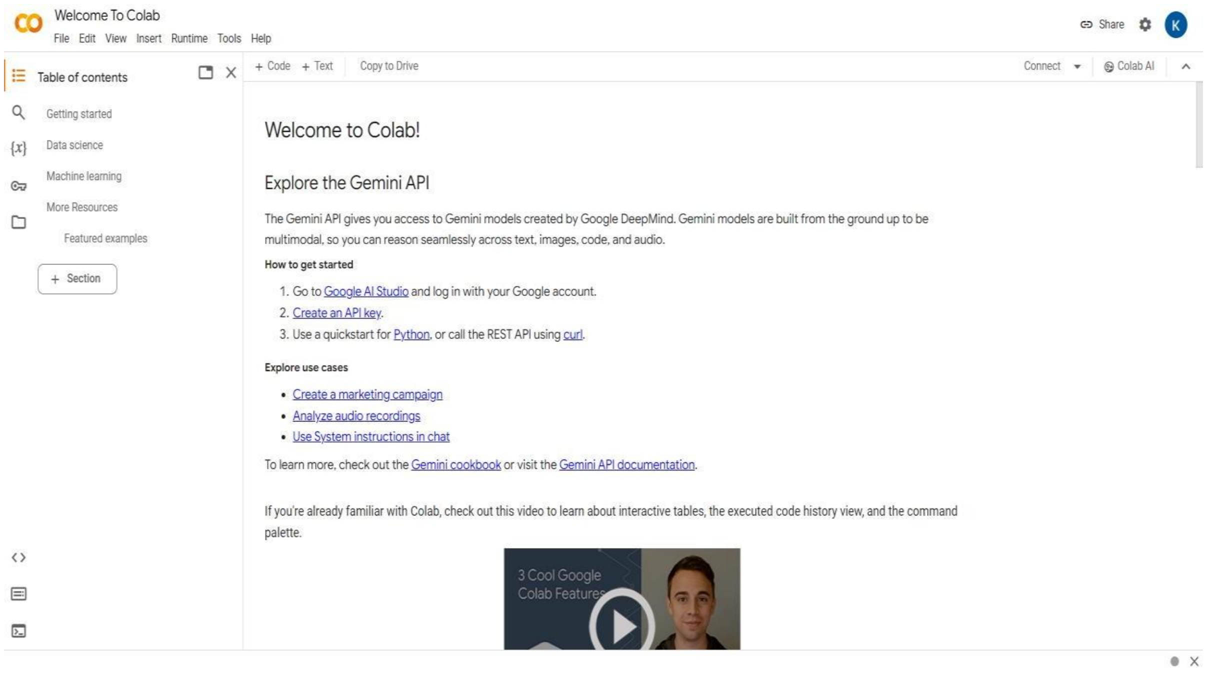
Finally, we addressed zero values in columns where zeros were not logically valid, such as the distance between the restaurant and delivery location. Using Pandas, we identified zero values and replaced them with a more appropriate measure, like the mean of the column, through the replace method. This step was vital to maintain the data's logical consistency and validity.

By following this detailed and systematic approach to data cleaning, we prepared a dataset that was robust, reliable, and ready for in-depth exploratory data analysis. This cleaned dataset laid a solid foundation for uncovering meaningful insights into the factors influencing delivery times in online delivery services

# CHAPTER 4
# System Architecture and Implementation

## 4.1 Google Colab

Google Colaboratory, commonly known as Google Colab, is a free online cloud-based Jupyter notebook environment tailored for training machine learn- ing and deep learning models. This article explores the functionalities, benefits, and features of Google Colab, elucidating its significance in the realm of data science and machine learning.



Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

**How to get started**

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

**Explore use cases**

- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

**Figure 4.1:** Google Colab

### 4.1.1 What is Google Colab?

Google Colab offers a cloud-based environment accessible via any web browser, eliminating the need for local software installation. Users can leverage its computing resources, including CPUs, GPUs, and TPUs, facilitating efficient model training and execution.

## 4.2 Benefits of Google Colab

**Accessibility**: Users can access Google Colab from any location with internet connectivity, streamlining collaboration and workflow.
**Power**: The platform provides access to potent computing resources like GPUs and TPUs, enabling swift and effective model training.
**Collaboration**: Google Colab simplifies collaborative efforts by allowing real-time editing and sharing of notebooks among team members.
**Education**: It serves as an invaluable educational tool for learning about machine learning and data science, offering a plethora of tutorials and resources.

### 4.2.1 Why Choose Google Colab?

Google Colab stands out as an ideal choice for students, data scientists, researchers, and enthusiasts due to its:
**Ease of Use**: With no setup requirements, users can swiftly start coding after creating an account.
**Affordability**: The platform is largely free to use, with paid plans available for more demanding tasks.
**Flexibility**: Users can seamlessly train models, process data, create visu-alizations, and collaborate with others, making it a versatile tool for various applications.

### 4.2.2 Notebook in Google Colab

In Google Colab, a notebook serves as a web-based environment for code creation and execution. Notebooks offer several advantages, including real-time code execution and visualization, support for markdown for documentation, and collaboration features, making them indispensable for data scientists and machine learning practitioners.

### 4.2.3 Google Colab Features

Google Colab boasts several features that enhance its usability and effec- tiveness:

**Free Access to GPUs and TPUs**: Users can leverage powerful computing resources without any additional cost.

**Web-based Interface**: The intuitive and user-friendly interface eliminates the need for local software installation.

**Collaboration Tools**: Multiple users can collaborate on the same notebook simultaneously, streamlining teamwork.

**Markdown Support**: Notebooks support markdown, enabling users to include formatted text, equations, and images alongside their code.

**Pre-installed Libraries**: Google Colab comes pre-installed with popular libraries and tools for machine learning and deep learning, such as TensorFlow and PyTorch, saving time on setup and configuration.

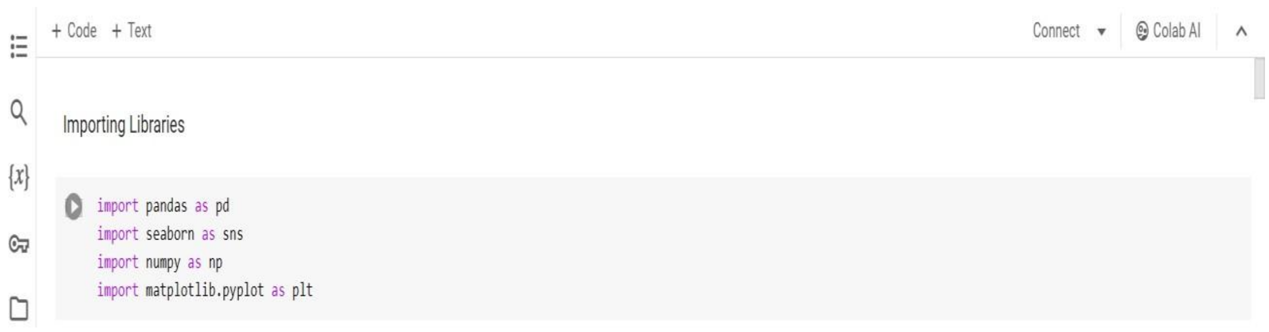
Google Colab emerges as a versatile and indispensable tool for machine learning and data science tasks, offering accessibility, power, and flexibility. Its user-friendly interface, collaborative features, and integration with powerful computing resources make it an invaluable asset for individuals and teams alike, driving innovation and progress in the field of machine learning and beyond.

## 4.3 Code Snippets

### 4.3.1 Importing libraries and Data loading

To begin our project, we first import the necessary libraries for data analysis and visualization. We import pandas as pd and numpy as np for data handling and numerical operations, respectively. We use Matplotlib and Seaborn for data visualization.

Next, we load the dataset into our code using the read csv function from pandas, assuming the dataset is stored in a CSV file named 'zomato.csv’. We assign the loaded dataset to a variable named 'df'.

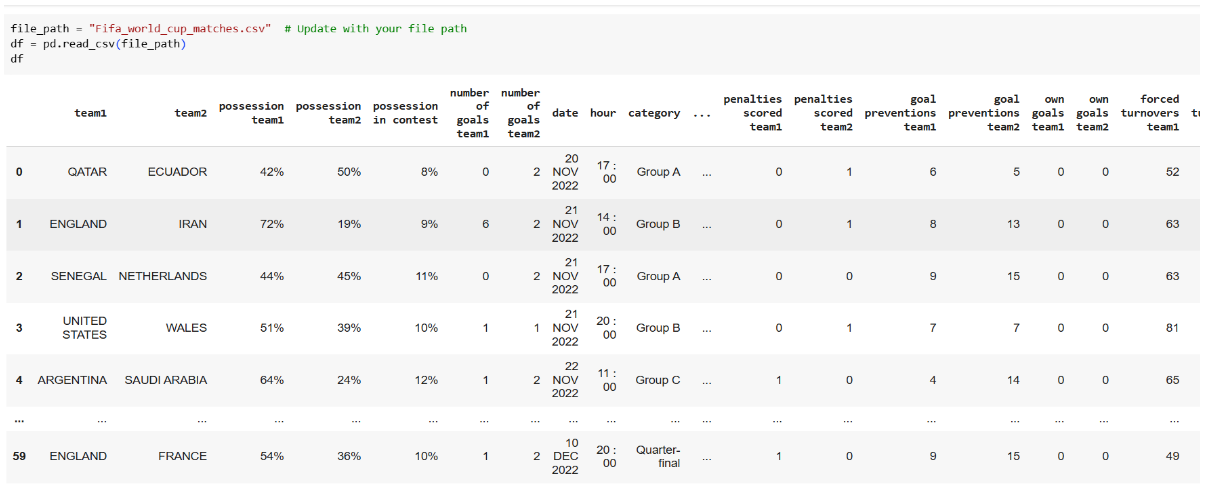


```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

**Figure 4.2: importing libraries and Dataset loading**

To ensure that the dataset has been loaded successfully, we display the first few rows of the dataset using the head() function. This allows us to inspect the structure and content of the dataset, confirming that it has been imported correctly and is ready for further processing.

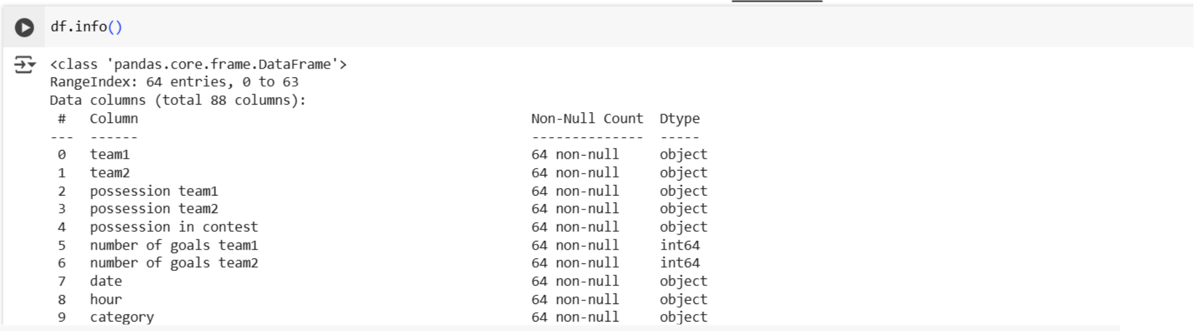
### Reading CSV File



```
file_path = "Fifa_world_cup_matches.csv"  # Update with your file path
df = pd.read_csv(file_path)
df
```

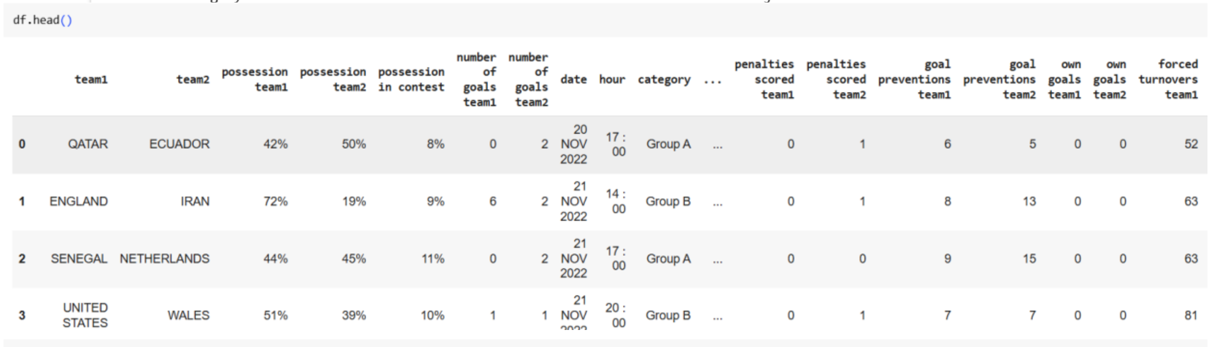
| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 | t |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | QATAR | ECUADOR | 42% | 50% | 8% | 0 | 2 | 20 NOV 2022 | 17 : 00 | Group A | ... | 0 | 1 | 6 | 5 | 0 | 0 | 52 | |
| **1** | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 | |
| **2** | SENEGAL | NETHERLANDS | 44% | 45% | 11% | 0 | 2 | 21 NOV 2022 | 17 : 00 | Group A | ... | 0 | 0 | 9 | 15 | 0 | 0 | 63 | |
| **3** | UNITED STATES | WALES | 51% | 39% | 10% | 1 | 1 | 21 NOV 2022 | 20 : 00 | Group B | ... | 0 | 1 | 7 | 7 | 0 | 0 | 81 | |
| **4** | ARGENTINA | SAUDI ARABIA | 64% | 24% | 12% | 1 | 2 | 22 NOV 2022 | 11 : 00 | Group C | ... | 1 | 0 | 4 | 14 | 0 | 0 | 65 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **59** | ENGLAND | FRANCE | 54% | 36% | 10% | 1 | 2 | 10 DEC 2022 | 20 : 00 | Quarter-final | ... | 1 | 0 | 9 | 15 | 0 | 0 | 49 | |

**Figure 4.4: Reading CSV**



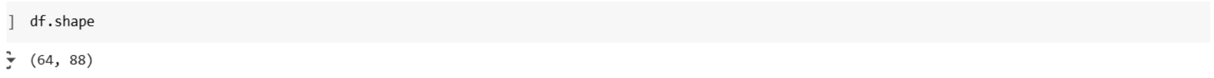
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 88 columns):
 #   Column                                              Non-Null Count  Dtype
---  ------                                              --------------  -----
 0   team1                                               64 non-null     object
 1   team2                                               64 non-null     object
 2   possession team1                                    64 non-null     object
 3   possession team2                                    64 non-null     object
 4   possession in contest                               64 non-null     object
 5   number of goals team1                               64 non-null     int64
 6   number of goals team2                               64 non-null     int64
 7   date                                                64 non-null     object
 8   hour                                                64 non-null     object
 9   category                                            64 non-null     object
```



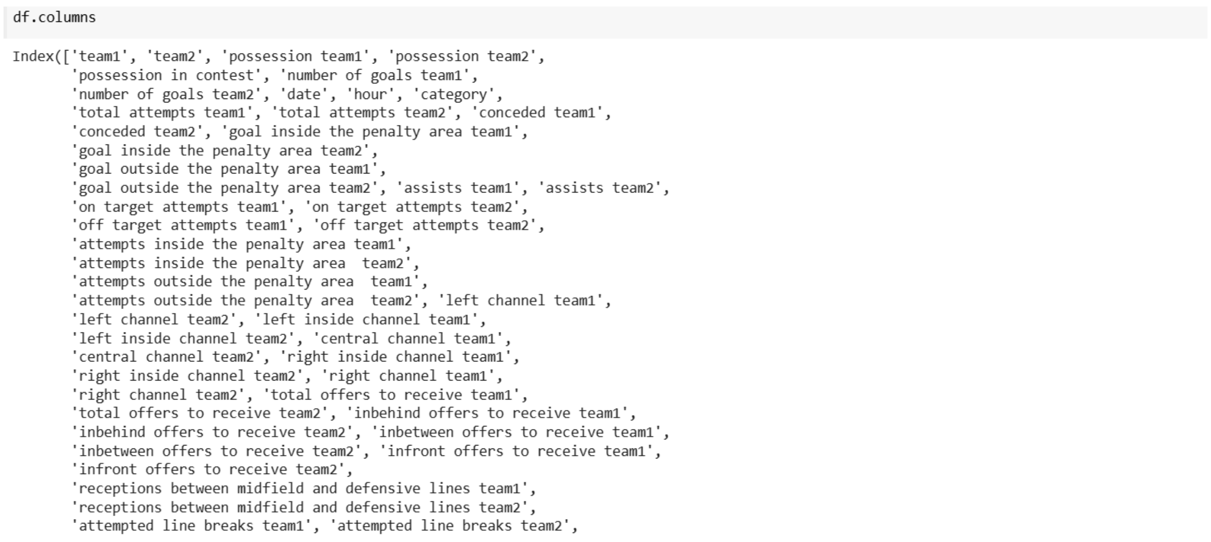
```
df.head()
```

| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | QATAR | ECUADOR | 42% | 50% | 8% | 0 | 2 | 20 NOV 2022 | 17 : 00 | Group A | ... | 0 | 1 | 6 | 5 | 0 | 0 | 52 |
| 1 | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 |
| 2 | SENEGAL | NETHERLANDS | 44% | 45% | 11% | 0 | 2 | 21 NOV 2022 | 17 : 00 | Group A | ... | 0 | 0 | 9 | 15 | 0 | 0 | 63 |
| 3 | UNITED STATES | WALES | 51% | 39% | 10% | 1 | 1 | 21 NOV | 20 : 00 | Group B | ... | 0 | 1 | 7 | 7 | 0 | 0 | 81 |



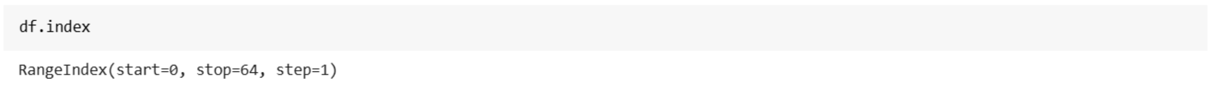
```
df.shape
```

```
(64, 88)
```



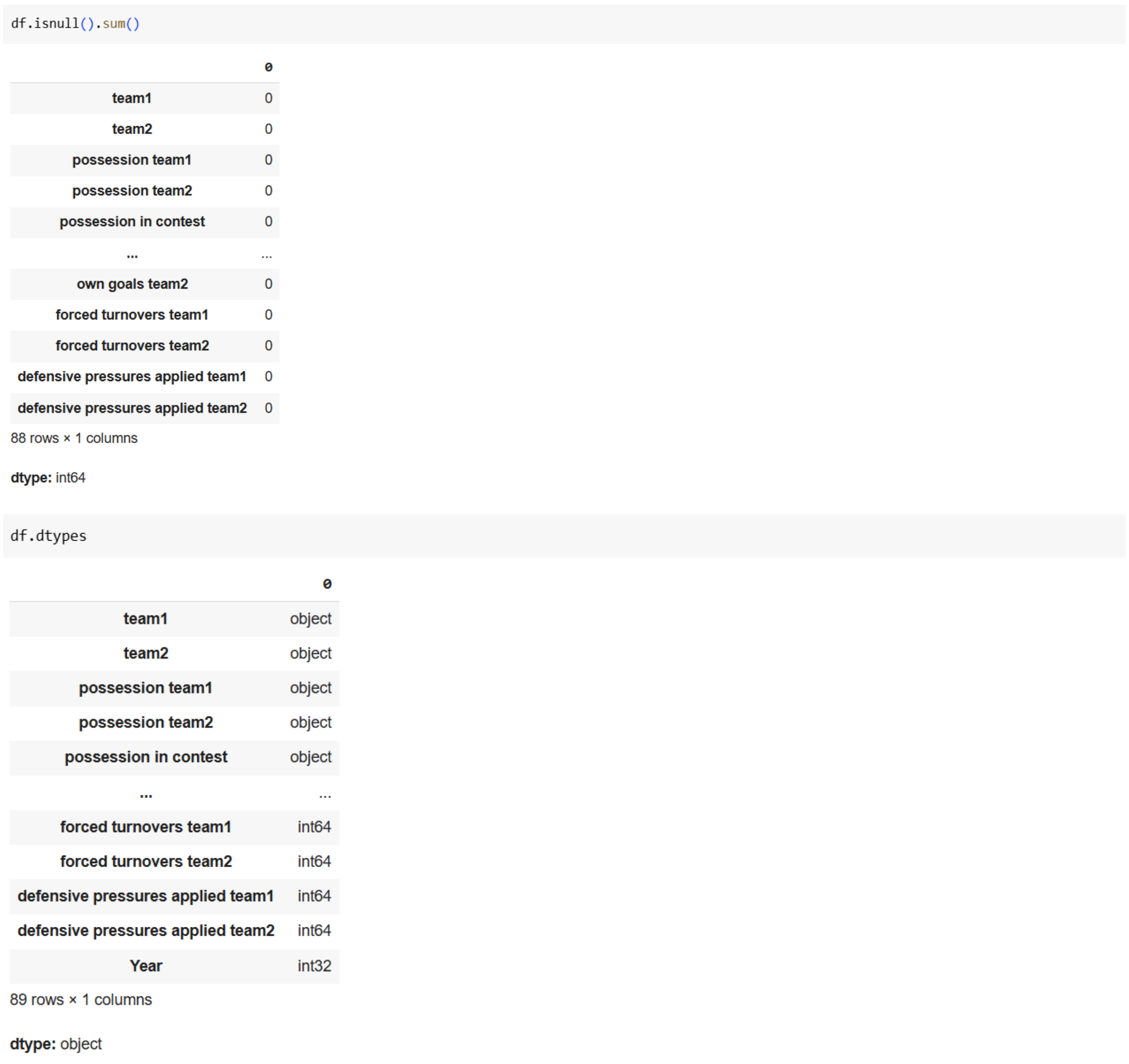
```
df.columns
```

```
Index(['team1', 'team2', 'possession team1', 'possession team2',
       'possession in contest', 'number of goals team1',
       'number of goals team2', 'date', 'hour', 'category',
       'total attempts team1', 'total attempts team2', 'conceded team1',
       'conceded team2', 'goal inside the penalty area team1',
       'goal inside the penalty area team2',
       'goal outside the penalty area team1',
       'goal outside the penalty area team2', 'assists team1', 'assists team2',
       'on target attempts team1', 'on target attempts team2',
       'off target attempts team1', 'off target attempts team2',
       'attempts inside the penalty area team1',
       'attempts inside the penalty area  team2',
       'attempts outside the penalty area  team1',
       'attempts outside the penalty area  team2', 'left channel team1',
       'left channel team2', 'left inside channel team1',
       'left inside channel team2', 'central channel team1',
       'central channel team2', 'right inside channel team1',
       'right inside channel team2', 'right channel team1',
       'right channel team2', 'total offers to receive team1',
       'total offers to receive team2', 'inbehind offers to receive team1',
       'inbehind offers to receive team2', 'inbetween offers to receive team1',
       'inbetween offers to receive team2', 'infront offers to receive team1',
       'infront offers to receive team2',
       'receptions between midfield and defensive lines team1',
       'receptions between midfield and defensive lines team2',
       'attempted line breaks team1', 'attempted line breaks team2',
```



```
df.index
```

```
RangeIndex(start=0, stop=64, step=1)
```



```
df.isnull().sum()
```

```
                                   0
                      team1        0
                      team2        0
            possession team1       0
            possession team2       0
         possession in contest     0
                        ...        ...
              own goals team2      0
          forced turnovers team1   0
          forced turnovers team2   0
defensive pressures applied team1  0
defensive pressures applied team2  0
88 rows × 1 columns

dtype: int64
```

```
df.dtypes
```

```
                                        0
                      team1        object
                      team2        object
            possession team1       object
            possession team2       object
         possession in contest     object
                        ...           ...
          forced turnovers team1    int64
          forced turnovers team2    int64
defensive pressures applied team1   int64
defensive pressures applied team2   int64
                       Year         int32
89 rows × 1 columns

dtype: object
```

**Figure 4.5:** Statistics for Numerical features

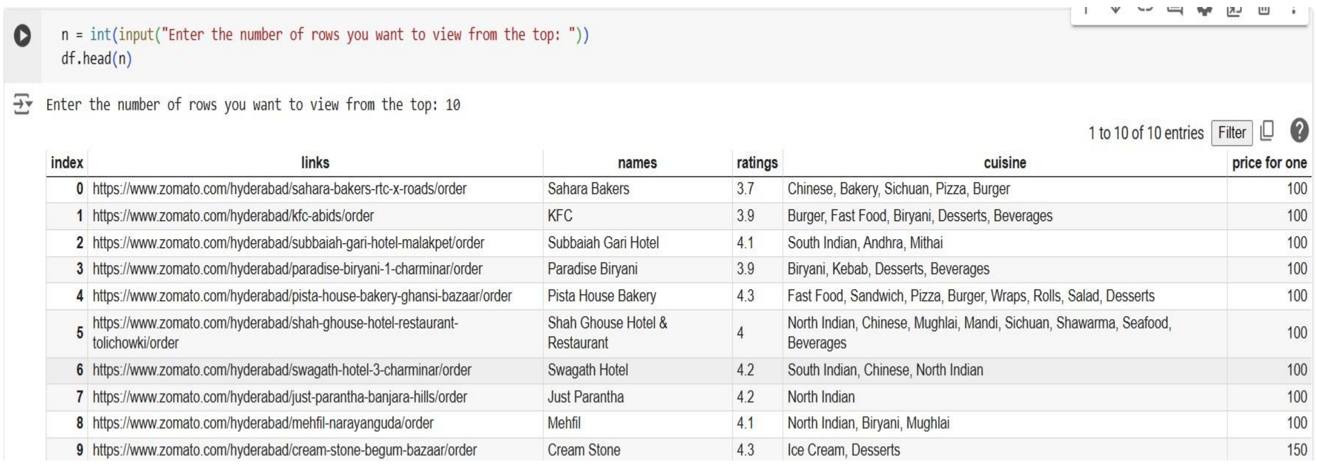
This code snippet prints out summary statistics for numerical features in a DataFrame (df). It provides key statistical measures such as count, mean, standard deviation, minimum, maximum, and quartile values for each numerical column in the DataFrame.

## 4.3.2 Data cleaning and preprocessing

The data cleaning process for our dataset was carried out in a structured manner to ensure quality and reliability for subsequent analysis

**1. Viewing Data**

Users can specify the number of rows to view from the top of the DataFrame. This feature helps in getting a quick look at the data's initial entries.



```
n = int(input("Enter the number of rows you want to view from the top: "))
df.head(n)
```

```
Enter the number of rows you want to view from the top: 10
```

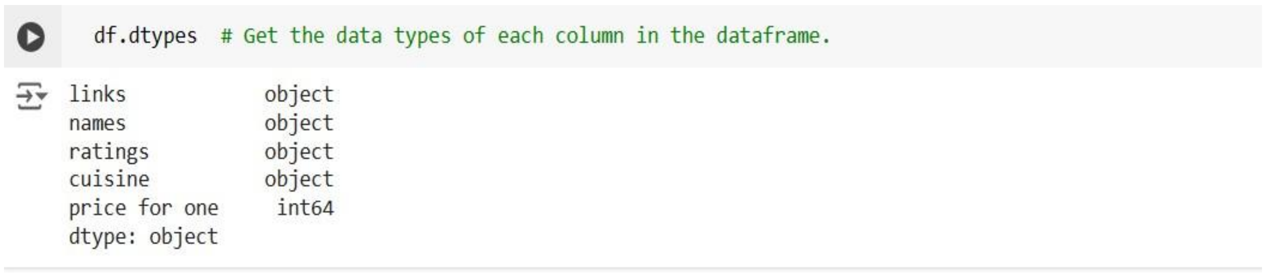
1 to 10 of 10 entries

| index | links | names | ratings | cuisine | price for one |
|---|---|---|---|---|---|
| 0 | https://www.zomato.com/hyderabad/sahara-bakers-rtc-x-roads/order | Sahara Bakers | 3.7 | Chinese, Bakery, Sichuan, Pizza, Burger | 100 |
| 1 | https://www.zomato.com/hyderabad/kfc-abids/order | KFC | 3.9 | Burger, Fast Food, Biryani, Desserts, Beverages | 100 |
| 2 | https://www.zomato.com/hyderabad/subbaiah-gari-hotel-malakpet/order | Subbaiah Gari Hotel | 4.1 | South Indian, Andhra, Mithai | 100 |
| 3 | https://www.zomato.com/hyderabad/paradise-biryani-1-charminar/order | Paradise Biryani | 3.9 | Biryani, Kebab, Desserts, Beverages | 100 |
| 4 | https://www.zomato.com/hyderabad/pista-house-bakery-ghansi-bazaar/order | Pista House Bakery | 4.3 | Fast Food, Sandwich, Pizza, Burger, Wraps, Rolls, Salad, Desserts | 100 |
| 5 | https://www.zomato.com/hyderabad/shah-ghouse-hotel-restaurant-tolichowki/order | Shah Ghouse Hotel & Restaurant | 4 | North Indian, Chinese, Mughlai, Mandi, Sichuan, Shawarma, Seafood, Beverages | 100 |
| 6 | https://www.zomato.com/hyderabad/swagath-hotel-3-charminar/order | Swagath Hotel | 4.2 | South Indian, Chinese, North Indian | 100 |
| 7 | https://www.zomato.com/hyderabad/just-parantha-banjara-hills/order | Just Parantha | 4.2 | North Indian | 100 |
| 8 | https://www.zomato.com/hyderabad/mehfil-narayanguda/order | Mehfil | 4.1 | North Indian, Biryani, Mughlai | 100 |
| 9 | https://www.zomato.com/hyderabad/cream-stone-begum-bazaar/order | Cream Stone | 4.3 | Ice Cream, Desserts | 150 |

**Figure 4.6: Viewing Data**

**2. Addressing Data Types:**

We reviewed the data types using Pandas' dtypes attribute and converted columns as needed with the astype method. For instance, delivery times stored as strings were converted to numerical types or datetime objects, enabling accurate computations and comparisons.



```
df.dtypes  # Get the data types of each column in the dataframe.
```

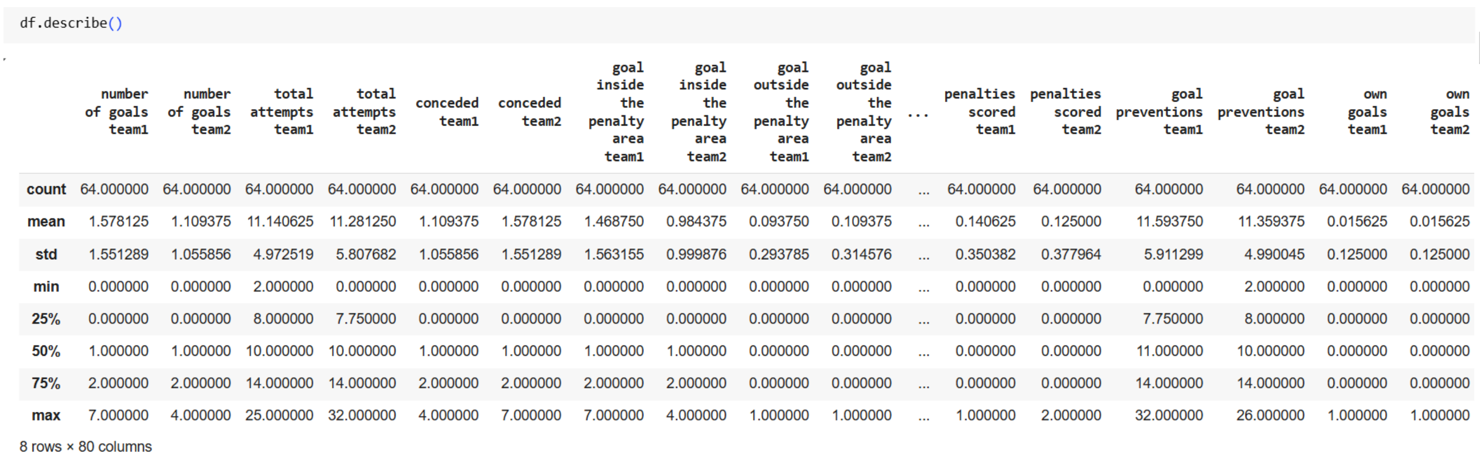
```
links            object
names            object
ratings          object
cuisine          object
price for one     int64
dtype: object
```

**Figure 4.7:Addressing Data Types**

## 5. Basic Data Exploration

The script provided insights into the structure of the DataFrame:

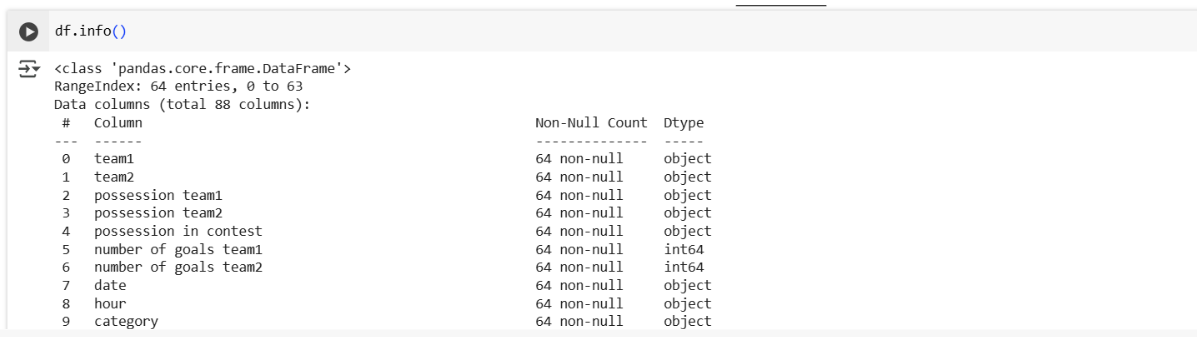
- **Shape**: Number of rows and columns.
- **Columns**: Names of the columns.
- **Index**: Indexes of the DataFrame.
- **Data Types**: Types of data in each column.
- **Missing Values**: Count of missing values in each column.



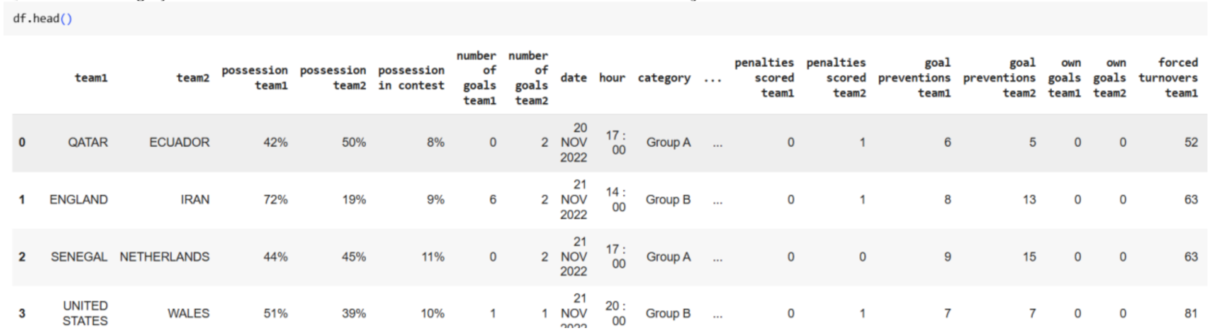
```
df.describe()
```

| | number of goals team1 | number of goals team2 | total attempts team1 | total attempts team2 | conceded team1 | conceded team2 | goal inside the penalty area team1 | goal inside the penalty area team2 | goal outside the penalty area team1 | goal outside the penalty area team2 | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | ... | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 | 64.000000 |
| **mean** | 1.578125 | 1.109375 | 11.140625 | 11.281250 | 1.109375 | 1.578125 | 1.468750 | 0.984375 | 0.093750 | 0.109375 | ... | 0.140625 | 0.125000 | 11.593750 | 11.359375 | 0.015625 | 0.015625 |
| **std** | 1.551289 | 1.055856 | 4.972519 | 5.807682 | 1.055856 | 1.551289 | 1.563155 | 0.999876 | 0.293785 | 0.314576 | ... | 0.350382 | 0.377964 | 5.911299 | 4.990045 | 0.125000 | 0.125000 |
| **min** | 0.000000 | 0.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 2.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 0.000000 | 8.000000 | 7.750000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 7.750000 | 8.000000 | 0.000000 | 0.000000 |
| **50%** | 1.000000 | 1.000000 | 10.000000 | 10.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 11.000000 | 10.000000 | 0.000000 | 0.000000 |
| **75%** | 2.000000 | 2.000000 | 14.000000 | 14.000000 | 2.000000 | 2.000000 | 2.000000 | 2.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 14.000000 | 14.000000 | 0.000000 | 0.000000 |
| **max** | 7.000000 | 4.000000 | 25.000000 | 32.000000 | 4.000000 | 7.000000 | 7.000000 | 4.000000 | 1.000000 | 1.000000 | ... | 1.000000 | 2.000000 | 32.000000 | 26.000000 | 1.000000 | 1.000000 |

8 rows × 80 columns

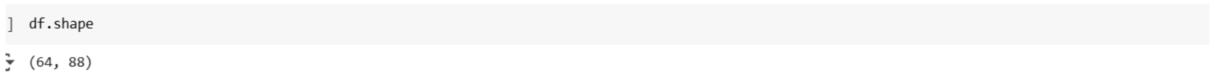


```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64 entries, 0 to 63
Data columns (total 88 columns):
 #   Column                                       Non-Null Count  Dtype
---  ------                                       --------------  -----
 0   team1                                        64 non-null     object
 1   team2                                        64 non-null     object
 2   possession team1                             64 non-null     object
 3   possession team2                             64 non-null     object
 4   possession in contest                        64 non-null     object
 5   number of goals team1                        64 non-null     int64
 6   number of goals team2                        64 non-null     int64
 7   date                                         64 non-null     object
 8   hour                                         64 non-null     object
 9   category                                     64 non-null     object
```



```
df.head()
```

| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | QATAR | ECUADOR | 42% | 50% | 8% | 0 | 2 | 20 NOV 2022 | 17 : 00 | Group A | ... | 0 | 1 | 6 | 5 | 0 | 0 | 52 |
| **1** | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 |
| **2** | SENEGAL | NETHERLANDS | 44% | 45% | 11% | 0 | 2 | 21 NOV 2022 | 17 : 00 | Group A | ... | 0 | 0 | 9 | 15 | 0 | 0 | 63 |
| **3** | UNITED STATES | WALES | 51% | 39% | 10% | 1 | 1 | 21 NOV | 20 : 00 | Group B | ... | 0 | 1 | 7 | 7 | 0 | 0 | 81 |



```
df.shape

(64, 88)
```

```
df.columns
```

```
Index(['team1', 'team2', 'possession team1', 'possession team2',
       'possession in contest', 'number of goals team1',
       'number of goals team2', 'date', 'hour', 'category',
       'total attempts team1', 'total attempts team2', 'conceded team1',
       'conceded team2', 'goal inside the penalty area team1',
       'goal inside the penalty area team2',
       'goal outside the penalty area team1',
       'goal outside the penalty area team2', 'assists team1', 'assists team2',
       'on target attempts team1', 'on target attempts team2',
       'off target attempts team1', 'off target attempts team2',
       'attempts inside the penalty area team1',
       'attempts inside the penalty area  team2',
       'attempts outside the penalty area  team1',
       'attempts outside the penalty area  team2', 'left channel team1',
       'left channel team2', 'left inside channel team1',
       'left inside channel team2', 'central channel team1',
       'central channel team2', 'right inside channel team1',
       'right inside channel team2', 'right channel team1',
       'right channel team2', 'total offers to receive team1',
       'total offers to receive team2', 'inbehind offers to receive team1',
       'inbehind offers to receive team2', 'inbetween offers to receive team1',
       'inbetween offers to receive team2', 'infront offers to receive team1',
       'infront offers to receive team2',
       'receptions between midfield and defensive lines team1',
       'receptions between midfield and defensive lines team2',
       'attempted line breaks team1', 'attempted line breaks team2',
```

```
df.index
```

```
RangeIndex(start=0, stop=64, step=1)
```

```
df.isnull().sum()
```

| | 0 |
|---|---|
| **team1** | 0 |
| **team2** | 0 |
| **possession team1** | 0 |
| **possession team2** | 0 |
| **possession in contest** | 0 |
| **...** | ... |
| **own goals team2** | 0 |
| **forced turnovers team1** | 0 |
| **forced turnovers team2** | 0 |
| **defensive pressures applied team1** | 0 |
| **defensive pressures applied team2** | 0 |

88 rows × 1 columns

**dtype:** int64

```
df.dtypes
```

| | 0 |
|---|---|
| **team1** | object |
| **team2** | object |
| **possession team1** | object |
| **possession team2** | object |
| **possession in contest** | object |
| **...** | ... |
| **forced turnovers team1** | int64 |
| **forced turnovers team2** | int64 |
| **defensive pressures applied team1** | int64 |
| **defensive pressures applied team2** | int64 |
| **Year** | int32 |

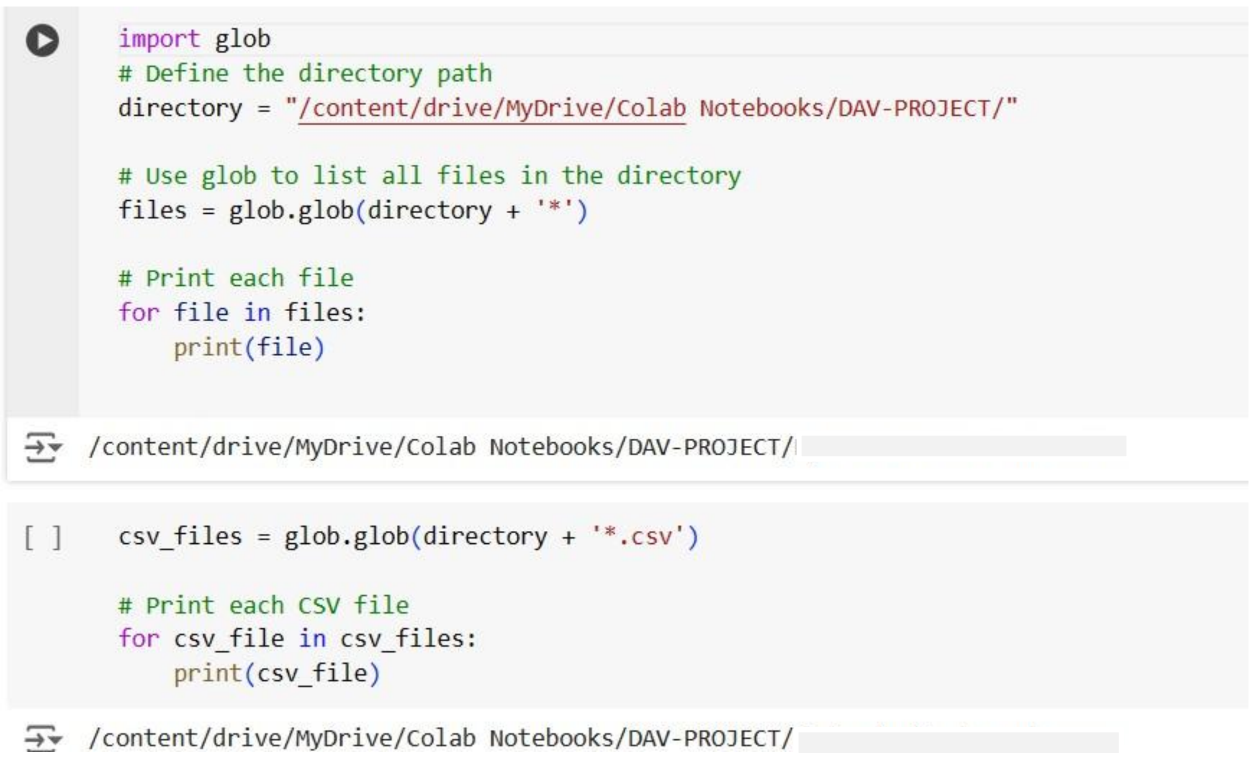
89 rows × 1 columns

**dtype:** object

**Figure 4.8: Basic Data Exploration**

**6. File Management**

The script included functionality to list all files in a specified directory and filter out only the CSV files. This helps in identifying all relevant data files in the working directory.



```
import glob
# Define the directory path
directory = "/content/drive/MyDrive/Colab Notebooks/DAV-PROJECT/"

# Use glob to list all files in the directory
files = glob.glob(directory + '*')

# Print each file
for file in files:
    print(file)
```

```
/content/drive/MyDrive/Colab Notebooks/DAV-PROJECT/[redacted]
```

```
csv_files = glob.glob(directory + '*.csv')

# Print each CSV file
for csv_file in csv_files:
    print(csv_file)
```

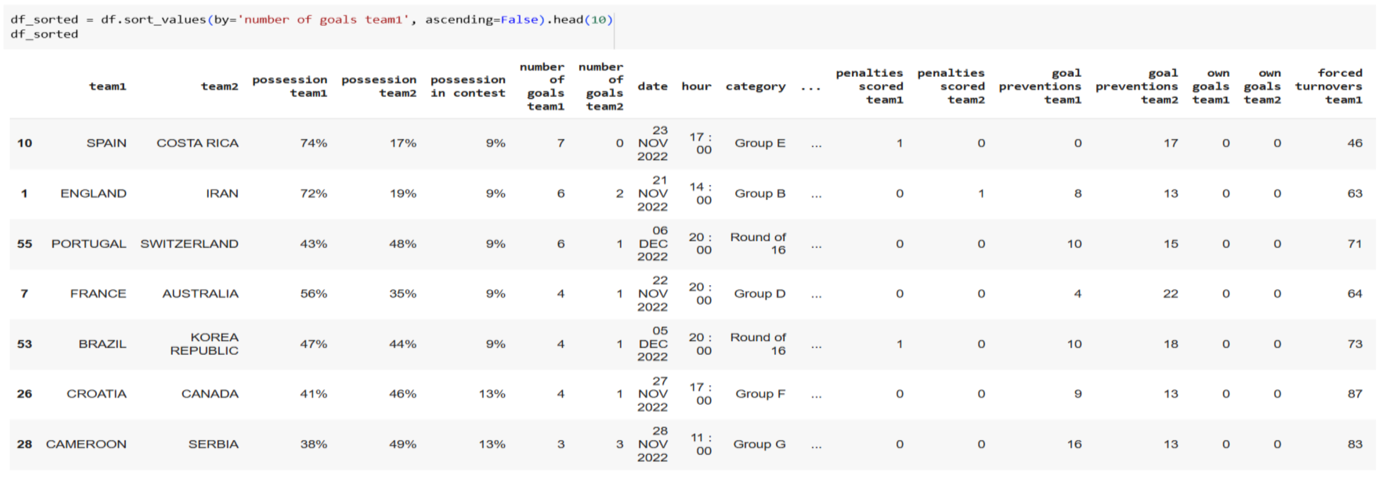
```
/content/drive/MyDrive/Colab Notebooks/DAV-PROJECT/[redacted]
```

**Figure 4.9:File Mangement**

## Data Manipulation

### Sorting

The dataset is sorted by 'number of goals team1' to identify matches where our team scored the most goals, helping study their best offensive performances.



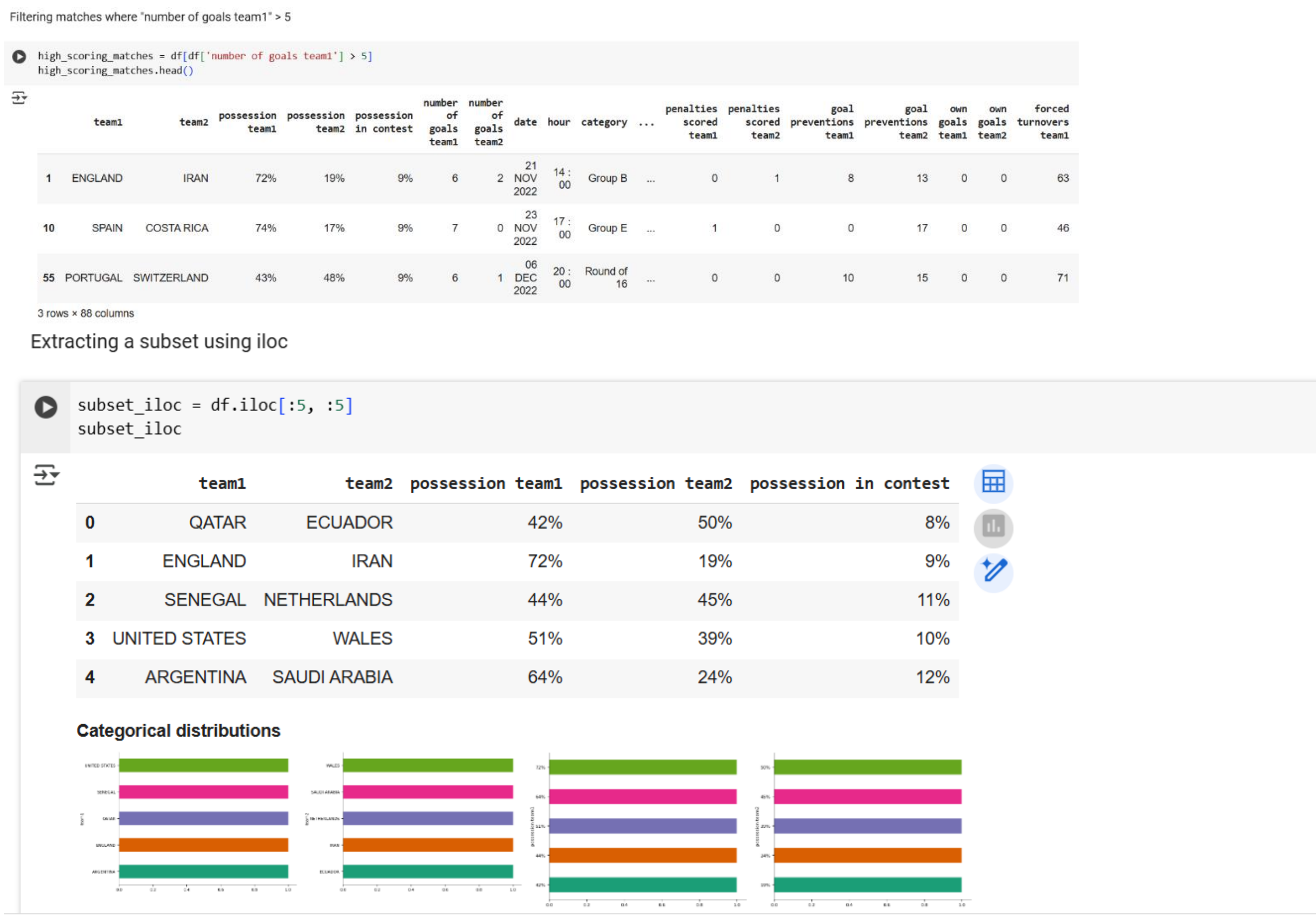
```
df_sorted = df.sort_values(by='number of goals team1', ascending=False).head(10)
df_sorted
```

| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | SPAIN | COSTA RICA | 74% | 17% | 9% | 7 | 0 | 23 NOV 2022 | 17 : 00 | Group E | ... | 1 | 0 | 0 | 17 | 0 | 0 | 46 |
| **1** | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 |
| **55** | PORTUGAL | SWITZERLAND | 43% | 48% | 9% | 6 | 1 | 06 DEC 2022 | 20 : 00 | Round of 16 | ... | 0 | 0 | 10 | 15 | 0 | 0 | 71 |
| **7** | FRANCE | AUSTRALIA | 56% | 35% | 9% | 4 | 1 | 22 NOV 2022 | 20 : 00 | Group D | ... | 0 | 0 | 4 | 22 | 0 | 0 | 64 |
| **53** | BRAZIL | KOREA REPUBLIC | 47% | 44% | 9% | 4 | 1 | 05 DEC 2022 | 20 : 00 | Round of 16 | ... | 1 | 0 | 10 | 18 | 0 | 0 | 73 |
| **26** | CROATIA | CANADA | 41% | 46% | 13% | 4 | 1 | 27 NOV 2022 | 17 : 00 | Group F | ... | 0 | 0 | 9 | 13 | 0 | 0 | 87 |
| **28** | CAMEROON | SERBIA | 38% | 49% | 13% | 3 | 3 | 28 NOV 2022 | 11 : 00 | Group G | ... | 0 | 0 | 16 | 13 | 0 | 0 | 83 |

**Figure 4.10: Sorting**

**Filtering**

Matches where 'number of goals team1' > 5 are filtered to focus on high-scoring games, analyzing exceptional performances.



Filtering matches where "number of goals team1" > 5

```
high_scoring_matches = df[df['number of goals team1'] > 5]
high_scoring_matches.head()
```

| | team1 | team2 | possession team1 | possession team2 | possession in contest | number of goals team1 | number of goals team2 | date | hour | category | ... | penalties scored team1 | penalties scored team2 | goal preventions team1 | goal preventions team2 | own goals team1 | own goals team2 | forced turnovers team1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ENGLAND | IRAN | 72% | 19% | 9% | 6 | 2 | 21 NOV 2022 | 14 : 00 | Group B | ... | 0 | 1 | 8 | 13 | 0 | 0 | 63 |
| 10 | SPAIN | COSTA RICA | 74% | 17% | 9% | 7 | 0 | 23 NOV 2022 | 17 : 00 | Group E | ... | 1 | 0 | 0 | 17 | 0 | 0 | 46 |
| 55 | PORTUGAL | SWITZERLAND | 43% | 48% | 9% | 6 | 1 | 06 DEC 2022 | 20 : 00 | Round of 16 | ... | 0 | 0 | 10 | 15 | 0 | 0 | 71 |

3 rows × 88 columns

Extracting a subset using iloc

```
subset_iloc = df.iloc[:5, :5]
subset_iloc
```

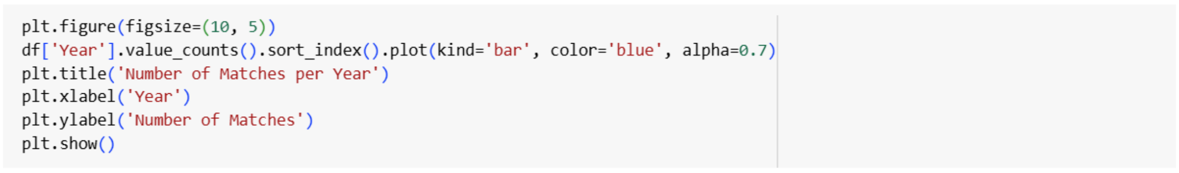
| | team1 | team2 | possession team1 | possession team2 | possession in contest |
|---|---|---|---|---|---|
| 0 | QATAR | ECUADOR | 42% | 50% | 8% |
| 1 | ENGLAND | IRAN | 72% | 19% | 9% |
| 2 | SENEGAL | NETHERLANDS | 44% | 45% | 11% |
| 3 | UNITED STATES | WALES | 51% | 39% | 10% |
| 4 | ARGENTINA | SAUDI ARABIA | 64% | 24% | 12% |

Categorical distributions

**Figure 4.11: Filtering**

## Value Counts

The number of matches per year is counted using value_counts() on the 'Year' column, showing participation frequency.



```
plt.figure(figsize=(10, 5))
df['Year'].value_counts().sort_index().plot(kind='bar', color='blue', alpha=0.7)
plt.title('Number of Matches per Year')
plt.xlabel('Year')
plt.ylabel('Number of Matches')
plt.show()
```

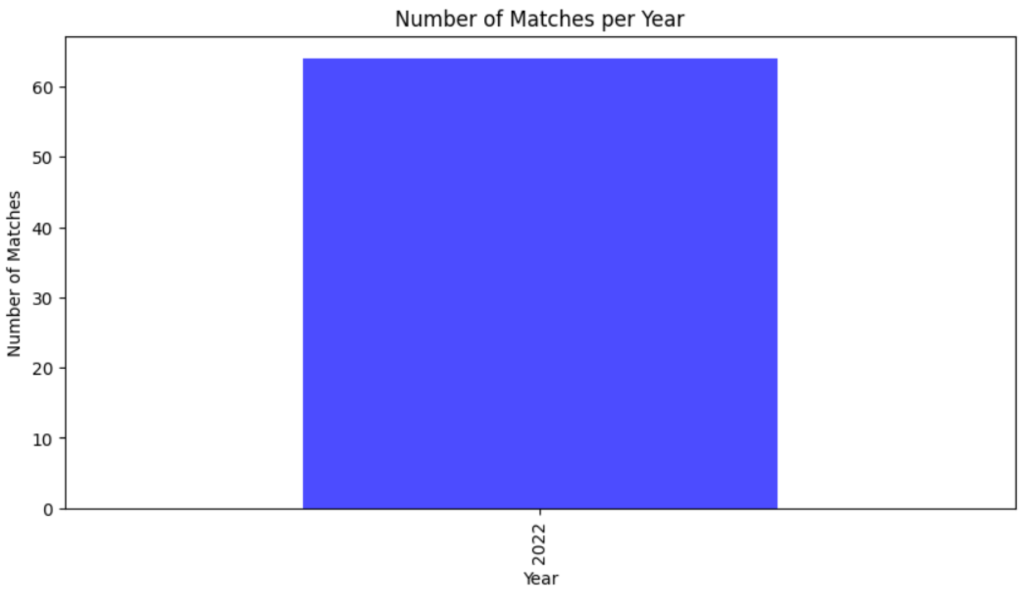
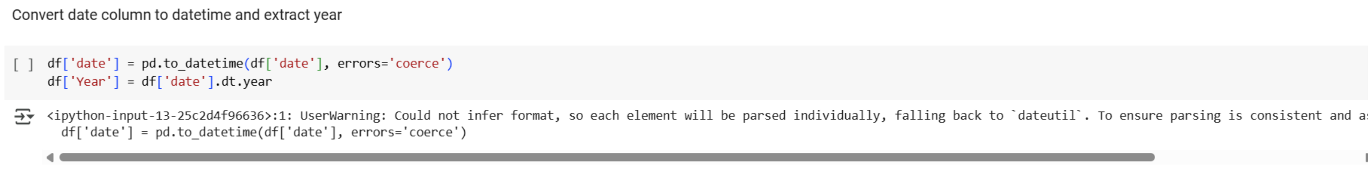


**Figure 4.12: Value Counts**

**Converting Date**

A conversion function was implemented to convert date column to exact datetime and year



Convert date column to datetime and extract year

```
[ ] df['date'] = pd.to_datetime(df['date'], errors='coerce')
    df['Year'] = df['date'].dt.year

    <ipython-input-13-25c2d4f96636>:1: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and a
      df['date'] = pd.to_datetime(df['date'], errors='coerce')
```

**Figure 4.13: Converting Date**

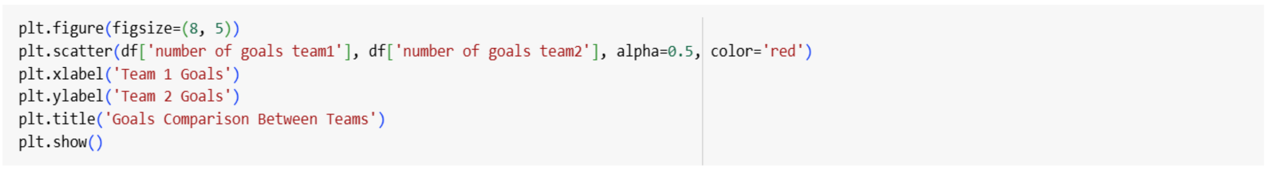
## 4.3.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a crucial preliminary step in data analysis, focusing on understanding the dataset's structure, identifying patterns, and uncovering relationships between variables. It involves visualizing data, summarizing key features, and detecting potential anomalies. EDA serves as a foundation for further analysis and model building.

### 1. Scatter Plot Analysis:

Scatter plots are effective graphical tools for exploring relationships between two continuous variables. In the context of the FIFA dataset, scatter plots can be used to visually assess potential associations between different features and the target variable. For instance, this plot compares goals scored by our team and opponents. It reveals correlations, such as whether high-scoring games for our team (e.g., >3 goals) correspond to high or low opponent scores.

.



```python
plt.figure(figsize=(8, 5))
plt.scatter(df['number of goals team1'], df['number of goals team2'], alpha=0.5, color='red')
plt.xlabel('Team 1 Goals')
plt.ylabel('Team 2 Goals')
plt.title('Goals Comparison Between Teams')
plt.show()
```

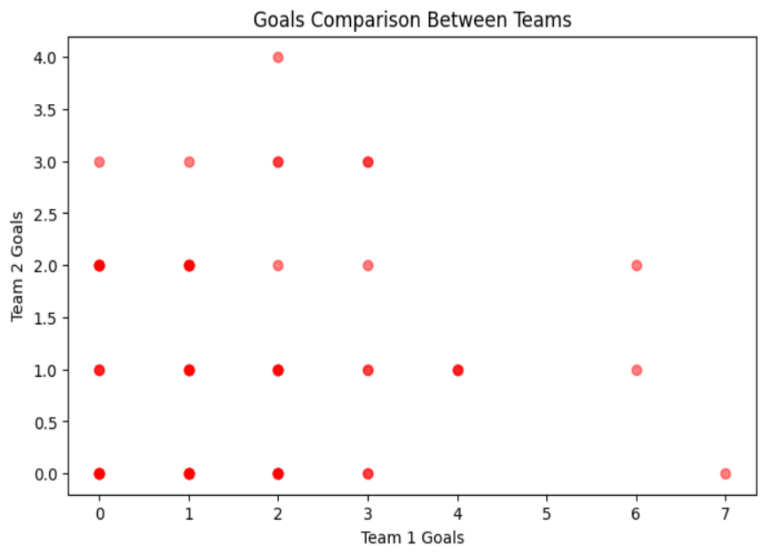
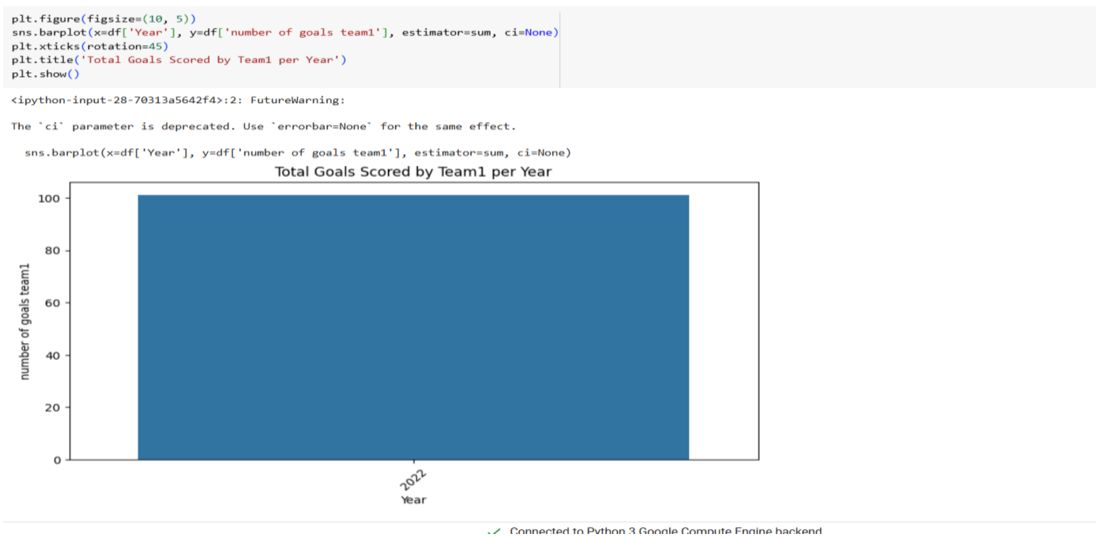


**Figure 4.14: Scatter Plot**

## 2. Sea Born Analysis

### a) Sea born bar plot

Displays total goals scored by our team per year, tracking performance trends.



```
plt.figure(figsize=(10, 5))
sns.barplot(x=df['Year'], y=df['number of goals team1'], estimator=sum, ci=None)
plt.xticks(rotation=45)
plt.title('Total Goals Scored by Team1 per Year')
plt.show()
```

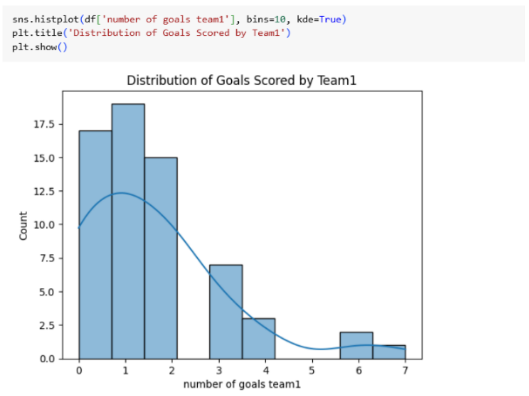
```
<ipython-input-28-70313a5642f4>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x=df['Year'], y=df['number of goals team1'], estimator=sum, ci=None)
```

Total Goals Scored by Team1 per Year
100
80
60
40
20
0
number of goals team1
2022
Year
Connected to Python 3 Google Compute Engine backend

Figure 4.15: Seaborn bar plot

### b) Sea born histplot

Illustrates the distribution of goals scored by our team, showing scoring patterns.



```
sns.histplot(df['number of goals team1'], bins=10, kde=True)
plt.title('Distribution of Goals Scored by Team1')
plt.show()
```

Figure 4.16: Seaborn hist plot

**C) Sea born lineplot**

Shows the trend of goals scored over years identifying performance changes.

```
plt.figure(figsize=(10, 5))
sns.lineplot(x=df['Year'], y=df['number of goals team1'])
plt.xticks(rotation=45)
plt.title('Trend of Goals Scored by Team1 Over Years')
plt.show()
```

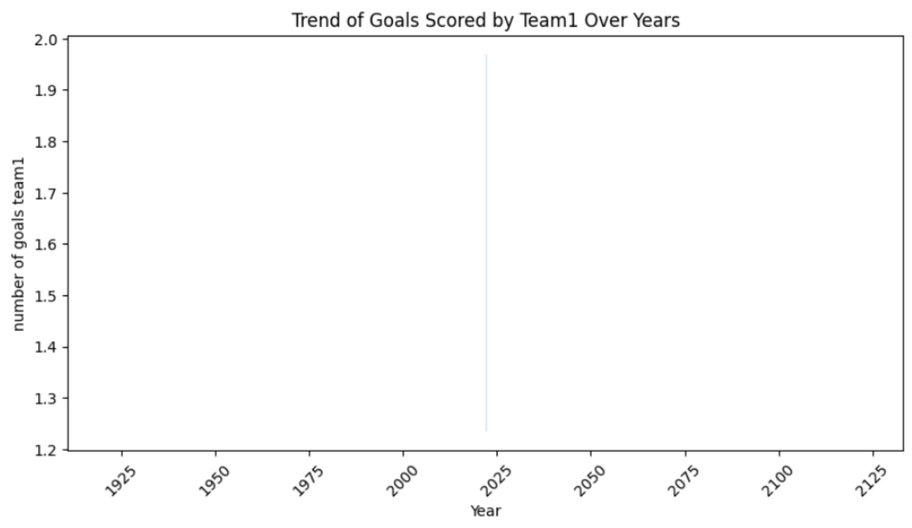


**Figure 4.17: Seaborn line plot**

## D) Sea born Box Plot

Compares goal distributions between our team and opponents, highlighting statistical differences.

```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['number of goals team1', 'number of goals team2']], palette='Set2')
plt.title('Boxplot of Goals Scored by Both Teams')
plt.show()
```

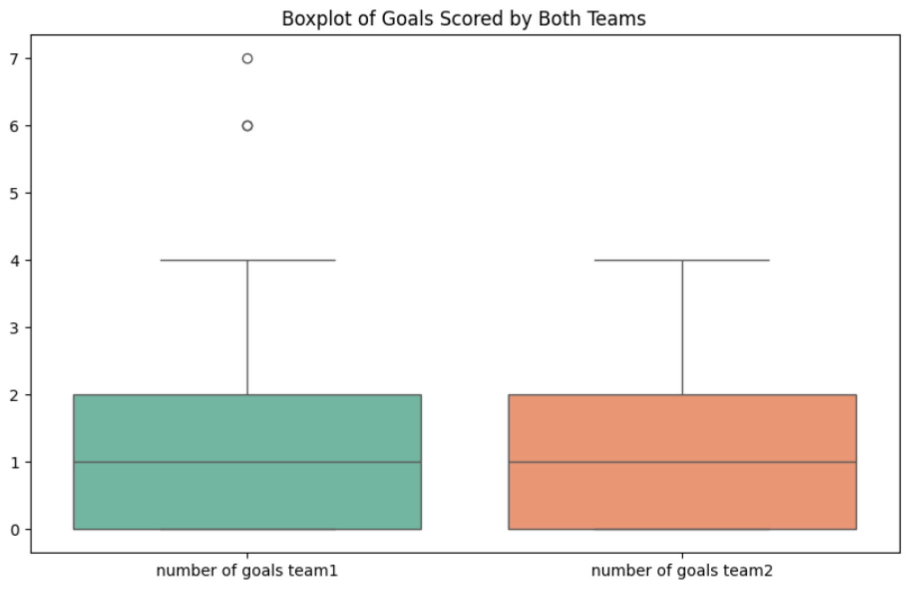


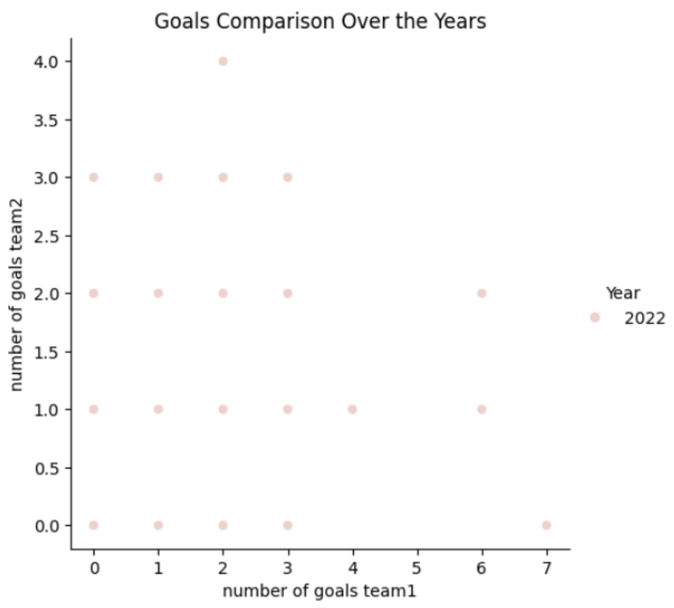
**Figure 4.18: Seaborn Box plot**

**e) Sea born Rel Plot**

Analyzes the relationship between goals scored by both teams over years, colored by year.

**Figure 4.19: Seaborn Rel plot**

```
sns.relplot(x='number of goals team1', y='number of goals team2', hue='Year', data=df)
plt.title('Goals Comparison Over the Years')
plt.show()
```



**F) Sea Born pair plot**

Presents pairwise relationships between goals and year, offering comprehensive insights.

```
sns.pairplot(df[['number of goals team1', 'number of goals team2', 'Year']])
plt.show()
```

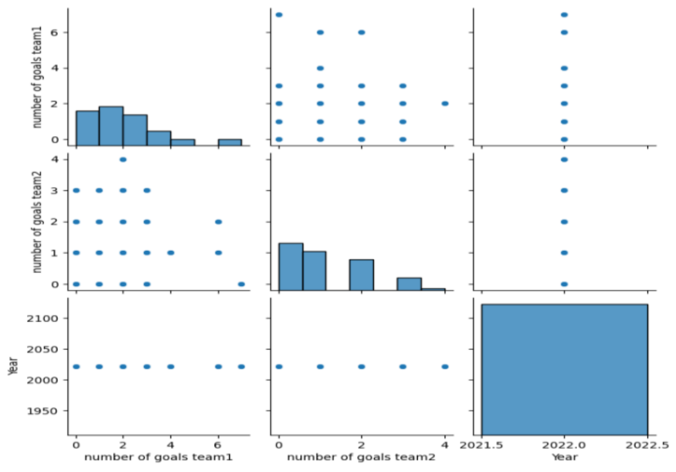


**Figure 4.20: Seaborn pair plot**

## 4.4 Conclusion

This project offers analysis of comprehensive understanding of  teams FIFA World Cup performance, revealing goal-scoring patterns, match distributions, and trends over time. It highlights strengths, such as high-scoring matches, and areas for improvement, like consistency in scoring. The insights can inform strategic decisions, enhancing future tournament performances by leveraging historical data.

.

## References

1. **Pandas Documentation:**
    - McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 51-56.
    - Pandas Documentation
2. **Matplotlib Documentation:**
    - Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90-95.
    - Matplotlib Documentation
3. **Seaborn Documentation:**
    - Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. *Journal of Open Source Software*, 6(60), 3021.
    - Seaborn Documentation

4.**Exploratory Data Analysis (EDA):**
- Behrens, J. T. (1997). Principles and Procedures of Exploratory Data Analysis. *Psychological Methods*, 2(2), 131-160.
- Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

4. **Visualization Techniques:**
    - Few, S. (2012). *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. Analytics Press.
    - Knaflic, C. N. (2015). *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley.
5. **Python for Data Analysis:**
    - McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

6. **VanderPlas, J. ( Python Data Science Handbook)**:
- Essential Tools for Working with Data. O'Reilly Media