In the premier\_league\_squad\_web\_scraping file, I web scraped information about all players in the football squad for each of the same 7 premier league squad 2023/24 I will be conducting an EDA on using FIFA 2023 player data. Information I web scraped for each club from the official Premier League site included players first name, last name, position, squad number and nationality. This was achieved in the following steps:

* I first imported the necessary python libraries including “request” and “BeautifulSoup” for web scraping, “pandas” for data manipulation and “itertool.zip\_longest”
* I then defined the URL of the Premier League squad webpage for each of the 7 clubs in the official Premier League website (Arsenal, Manchester city. Liverpool, Manchester United, Chelsea, Newcastle United and Tottenham Hotspur)
* I used the “request.get()” function to send a HTTP GET request to the specifief URL and checked the request was successful (status code 200). It was successful and I

received a valid response.

* I used “BeautifulSoup” to parse the HTML content of the webpage for each club (e.g. ‘arsenal\_soup’ or ‘man\_city\_soup’) to ensure it is structured and specific elements are extracted from it. The BeautifulSoup’s “find\_all()” method was used to extract players first name, last name, position, squad number and nationality.
* To ensure data consistency I ensured all the lists containing player information have the same length by finding the maximum length amongst them (e.g. “arsenal\_max\_length” and “man\_ city\_max\_lenght”).
* I then extracted and processed data from HTML elements into separate lists (e.g. “arsenal\_first\_names” and “man\_city\_last\_names”), using list comprehension. These lists contained player information.
* The data was then cleaned and manipulated for preparation. This included inserting empty strings in “first\_name” where players only went by their last name to align the data and used ‘zip\_longest’ function to ensure all lists have the same length, filling in missing values with “None”.
* The data was then transformed to update player position from “Goalkeeper” to “Goalkeeper (GK)” for consistency.
* The file path was specified where the DataFrame will be saved as a CSV file (e.g. ‘arsenal\_csv\_file\_path’ or ‘man\_city\_csv\_file\_path’) and the ‘to\_csv()’ is used to save the DataFrame to the file. Finally, a print message indicting the CSV file has been created and the steps above where repeated for 7 selected premier league teams.

**Data Processing, Pandas Library Storage and SQLITE Database:**

This part of the code is used to perform data cleaning and processing tasks to prepare the FIFA 2023 player data for storage in an SQLite database. The data is cleaned, by removing accents, splitting and capitalised names, making the data more suitable for analysis. Once cleaned the data is saved into an SQLite database.

1. **Importing Libraries:**

First the necessary python libraries that will be used are imported, including ‘pandas’ for data manipulation, ‘numpy’ for numerical operation, ‘unidecode’ for removing accents and special characters for text, ‘sqlite3’ for interacting with the SQLite databases, and several data visualisation libraries (‘matplotlib’, ‘plotly’, ‘seaborn’, ‘mplcursors’ for creating plots and graphs.

1. **Creating and Connecting to SQLite Database:**

The ‘sqlite3.connect()’ function was used to create a new SQLite database. The ‘conn’ variable holds the connect to the database.

1. **Reading the FIFA 2023 Player Data:**

Using the pandas DataFrame using ‘pd.read\_csv()’ function, the “fifa\_player\_data\_df” was read from a csv file. The csv file path was specified and the “utf-8” encoding was used for proper character encoding.

1. **Data Cleaning with Unicode:**

A custom function ‘clean\_name()’ that used the ‘unidecode’ library was defined to remove accents and convert special characteristics in player names to their closest ASCII represenatation. This was applied to the “Player\_Name” column of the DataFrame, cleaning up player names.

Another custom function was defined ‘split\_name()’ that splits full name column into first name and last name. it splits the names based on the spaces and returns them separately. The “Full\_Name” Column is split into “First\_Name\_Initial” and “Last\_Name” columns to store the results. The first letter of the first and last name was also capitalised.

1. **Database Table Creation:**

The ‘to\_sql()’ method of the pandas DataFrame is then used to write the cleaned data to SQLite table named “fifa\_player\_data” within the SQLite database specified by the ‘conn’ connection. If the table already exists, it replaces it (‘if\_exists=”replace”’). The data is stored without an index column (‘index=False’).

**Loading and Processing Selected 7 PREMIER LEAGUE (PL) CLUBS DATA:**

In summary, for each of the 7 Premier league (PL) clubs the CSV file produced through web scraping for each team is read, cleaned and pre-processed, it is then stored in an SQLite database table, where SQL queries are executed to retrieve player information from both the 7 PL teams datasets and the FIFA player dataset. And finally display player information for both the retrieved and unretrieved players in each of the 7 PL datasets.

1. **Reading PL Squad CSV File:**

Initially the Premier League (PL) squad CSV file which was created by web scraping (read notes above). The file path is specified in the pandas DataFrame and read using the ‘pd.read\_csv()’ function. The data is located into squad-name\_df (e.g. “arsenal\_df”, “newcastle\_df” or “man\_united\_df”).

1. **Cleaning PL Squad Data:**

The function ‘clean\_name()’ that uses ‘unidecode’ library was defined to remove accents and convert special characters in the “Last\_Name” column of the DataFrame to ensure consistent and clean data.

The custom function ‘create\_first\_name\_initial()’ was defined to extract the first letter from the “First\_Name” column and adds a period (e.g. “John” becomes “J”).

This function is applied to create the “First\_Name\_Initial” column. All leading or trailing spaces in the “First\_Name\_Initial” column were stripped.

A new column was created called “Club\_Name” to the DataFrame and set it as the names of the 7 PL clubs selected for EDA (“Arsenal”, “Manchester City”, “Liverpool”, “Manchester United”, “Chelsea”, “Newcastle United” and “Tottenham Hotspur”) respectively for all rows in each squad dataframe. The purpose behind this was to ensure the right player was retrieved when executing SQLite queries.

1. **Storing Data in SQLite Databases:**

The ‘to\_sql()’ method was used for the 7 PL dataframe (e.g. ‘arsenal\_df’, ‘man\_city\_df’, ‘newcastle\_df’) to write the DataFrame to an SQLite table named (e.g. ‘arsenal\_squad’, ‘man\_city\_squad’, ‘newcastle\_squad’) with the SQLite database specified by the ‘conn’ connection. If the table already exists it replaces (if\_exists=”replace”’) to ensure the latest data is stored in the table. The data is stored without an index column (‘index=False’).

1. **Executing SQL Queries:**

A cursor was created using the ‘conn.cursor()’ to interact with the SQLite database. An SQL query was then executed that selects distinct player information from both the “fifa\_player\_data” and the 7 PL SQLite DataFrames created for each team (e.g. ‘arsenal\_squad’, ‘man\_city\_squad’, ‘newcastle\_squad’) using an ‘INNER JOIN’ clause. The query retrieves database on matching conditions like “First\_Name\_Initial”, Last\_Name, Player\_Nationality and Club\_Name. the retrieved data is stored in a variable as a lists of tuples named (e.g. ‘arsenal\_results, ‘tottenham\_results’, ‘man\_city\_results’)

1. **Displaying Retrieved Data**

a loop is created through the results of the SQL query e.g. ‘arsenal\_results, ‘tottenham\_results’, ‘man\_city\_results’) and the player information is printed in the formatted manner, including details like first name, last name, club name, player nationality, player position, player rating and other attributes. If there are no results (empty), “No results found” is printed.

1. **Filtering Unretrieved Players:**

A set was created for all the 7 PL teams called (e.g. ‘arsenal\_retrieved\_players’, ‘man\_city\_retrieved\_players’, ‘chelsea\_retrieved\_players’) to store the tuples of the first name initials, and last names for players retrieved in the SQLite query.

I filtered through the 7 PL clubs dataframes (e.g. “arsenal\_df”, “newcastle\_df” or “man\_united\_df”) to obtain rows where the players first name initial and last name do not match the retrieved players. This results in an unretrieved players data frame for each club (e.g. ‘arsenal\_unretrieved\_players\_df’, ‘man\_city\_unretrieved\_players\_df’, ‘chelsea\_unretrieved\_players\_df’).

The unretrieved players details including first name initials, last name and player nationality are printed so they can further investigated. If all players from the 7 PL clubs dataframes (e.g. “arsenal\_df”, “newcastle\_df” or “man\_united\_df”) are retrieved the message “All team players retrieved” is printed.

1. **Closing Cursor and Connection:**

The cursor for each of the 7 PL clubs is closed (e.g. ‘arsenal\_cursor\_close’, ‘man\_united\_cursor\_close’, ‘tottenham\_cursor\_close’) and subsequently, closing the connection to the SQLite database.

Finally the database connection is closed using the ‘conn.close()’ to protect the SQLite databases after all the necessary SQL queries have been executed.

**EDA and Data Visualisation for SQL Query Results:**

In this part of the code of the EDA, a comprehensive analysis of 7 Premier League teams is carried out against FIFA 2023 player data. This includes player ratings, abilities, positions and other attributes and uses various data visualisation to present findings.

1. Team Names and Average Ratings Lists:

A list of football team names is created.

1. Calculating Average Ratings:

For each team, the script calculates three types of average ratings: overall average team rating, potential average team rating, and average performance team rating.

These ratings are calculated based on specific data for each team, such as 'arsenal\_results,' 'man\_city\_results,' etc.

1. Creating a DataFrame:

The calculated ratings are stored in dictionaries.

These dictionaries are then used to create a Pandas DataFrame (team\_ratings\_df) for better data organization.

1. Plotting a Grouped Bar Chart:

The Pandas DataFrame is used to plot a grouped bar chart using Matplotlib.

The chart displays the average team ratings for each club.

1. Overall Goalkeeper Abilities:

A dictionary named overall\_goalkeeping\_abilities is created to store various goalkeeping abilities (Diving, Hand-Game, Foot-Game, Positioning, Reflexes) for each team.

Data for goalkeeping abilities is collected from different sets of team results (e.g., arsenal\_results) based on player positions.

1. Calculating Average Goalkeeping Abilities:

The script calculates the average goalkeeping abilities for each team by taking the mean of individual abilities.

1. Creating a Bar Chart for Goalkeeping Abilities:

A grouped bar chart is created to display the average goalkeeping abilities for each team.

1. Comparing Average Ratings by Position:

The script calculates average overall ratings for different player positions (Goalkeeper, Defender, Midfielder, Forward) for each team.

1. Creating a Grouped Bar Chart for Average Player Ratings:

A grouped bar chart is created to compare the average overall player ratings for different positions within each team.

1. Player Overall Rating Distribution:

The script collects data about player overall ratings from different teams and stores it in a DataFrame. It then creates a histogram to visualize the distribution of player overall ratings across all teams.

1. Distribution Curve for Defensive Overall Rating:

Similar to the player overall rating distribution, this section focuses on defenders. It collects data on defensive overall ratings and creates a histogram.

1. Distribution Curve for Midfielders Overall Rating:

Similar to the previous section, this part focuses on midfielders' overall ratings and creates a histogram.

1. Distribution Curve for Strikers Overall Rating:

This section, like the previous ones, focuses on strikers' overall ratings and creates a histogram.

1. Comparing Scoring Abilities for Forwards (Strikers):

The script calculates and compares the average scoring abilities (Volleying, Finishing, Heading) of forward players (strikers) for each team.

1. Scatter Graph Comparing Player Effectiveness Rating and Overall

Rating:

The script creates a scatter plot to compare player effectiveness ratings and overall ratings for different teams.

1. Scatter Graph Comparing Team Defensive Consciousness and Offensive Position:

Another scatter plot is created to compare player attributes related to offensive positioning and defensive consciousness for different teams.

1. Correlation Chart Comparing Player Value and Overall Rating:

This section collects data on player values and creates a scatter plot to explore the correlation between player values and overall ratings for each club.

1. Correlation Chart Comparing Player Salary and Overall Rating:

Similar to the previous section, this part focuses on player salaries and explores the correlation between salaries and overall ratings for each club.

1. Treemap and Histogram for Player Nationalities:

This part creates a treemap visualization to show the distribution of player nationalities within each team.

Additionally, it creates a histogram to visualize the distribution of nationalities within each team.