Aim: Introduction to Data science and Data preparation using Pandas steps.

- Load data in Pandas.
- Description of the dataset.
- Drop columns that aren't useful.
- Drop rows with maximum missing values.
- Take care of missing data.
- Create dummy variables.
- Find out outliers (manually)
- Standardization and normalization of columns

Steps:

1. **Loading the Dataset**

```
# Import required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Increase default figure size for better readability
plt.rcParams["figure.figsize"] = (12, 6)

# Load the dataset (make sure to upload your CSV file to Colab)
df = pd.read_csv("/content/fifa_eda_stats.csv")  # Adjust path if needed

# Preview the first few rows of the dataset
print("First few rows of the dataset:")
print(df.head())
```

```
First few rows of the dataset:
       ID              Name  Age Nationality  Overall  Potential  \
0  158023          L. Messi   31   Argentina       94         94
1   20801  Cristiano Ronaldo   33    Portugal       94         94
2  190871          Neymar Jr   26      Brazil       92         93
3  193080             De Gea   27       Spain       91         93
4  192985       K. De Bruyne   27     Belgium       91         92

                 Club    Value   Wage Preferred Foot  ...  Composure  \
0         FC Barcelona  €110.5M  €565K           Left  ...       96.0
1             Juventus    €77M  €405K          Right  ...       95.0
2  Paris Saint-Germain  €118.5M  €290K          Right  ...       94.0
3    Manchester United    €72M  €260K          Right  ...       68.0
4      Manchester City   €102M  €355K          Right  ...       88.0

   Marking  StandingTackle SlidingTackle GKDiving GKHandling  GKKicking  \
0     33.0            28.0          26.0      6.0       11.0       15.0
1     28.0            31.0          23.0      7.0       11.0       15.0
2     27.0            24.0          33.0      9.0        9.0       15.0
3     15.0            21.0          13.0     90.0       85.0       87.0
4     68.0            58.0          51.0     15.0       13.0        5.0

  GKPositioning GKReflexes Release Clause
0          14.0        8.0        €226.5M
1          14.0       11.0        €127.1M
2          15.0       11.0        €228.1M
3          88.0       94.0        €138.6M
4          10.0       13.0        €196.4M

[5 rows x 57 columns]
```

This dataset includes various attributes of FIFA players, such as **Name**, **Age**, **Nationality**, **Overall**, **Potential**, **Club**, **Value**, **Wage**, and many other football-related stats.

2. Creating a Bar Graph and Contingency Table

Objective

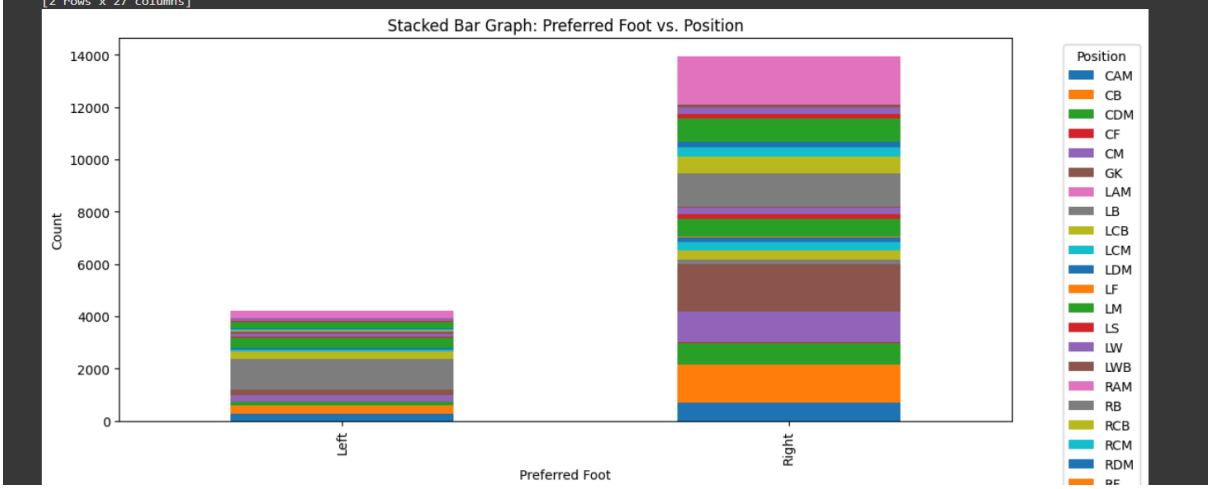
- Demonstrate how to visualize and compare two categorical variables.
- Understand the frequency distribution of one category across different subcategories.

Here, we examine **Preferred Foot** vs **Position**.

```
# Create a contingency table for Preferred Foot vs Position
contingency = pd.crosstab(df['Preferred Foot'], df['Position'])
print("Contingency Table (Preferred Foot vs Position):")
print(contingency)

# Plot a stacked bar graph using the contingency table
contingency.plot(kind='bar', stacked=True)
plt.title("Stacked Bar Graph: Preferred Foot vs. Position")
plt.xlabel("Preferred Foot")
plt.ylabel("Count")
plt.legend(title="Position", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
```

plt.show()

```
Contingency Table (Preferred Foot vs Position):
Position        CAM    CB  CDM  CF    CM    GK  LAM    LB  LCB  LCM  ... \
Preferred Foot                                                      ...
Left            256   341  126  18   249   206    9  1163  264   97  ...
Right           702  1437  822  56  1145  1819   12   159  384  298  ...

Position          RB  RCB  RCM  RDM  RF   RM   RS   RW  RWB   ST
Preferred Foot
Left              14   32   41   23   6  252   37   97    4  293
Right           1277  630  350  225  10  872  166  273   83  1859

[2 rows x 27 columns]
```



What this indicates:

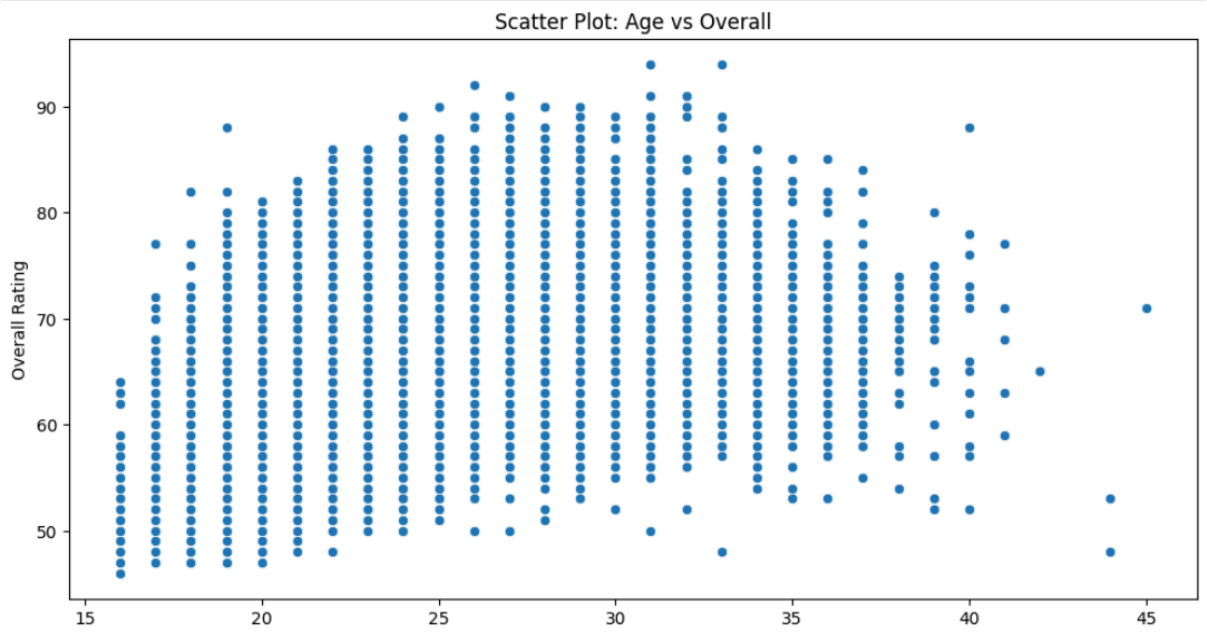
- The contingency table shows how many players prefer the left foot vs. the right foot, broken down by their playing positions.
- The stacked bar chart visually compares these counts. We can see, for instance, that **Right**-footed players dominate most positions, while there is still a notable count of **Left**-footed players in some roles (e.g., LB, LCB, etc.).
- This helps us identify whether certain positions (like defenders or midfielders) have a higher proportion of left-footed or right-footed players.

3. Scatter Plot, Box Plot, and Heatmap

Scatter Plot

We plot **Age** vs **Overall** to see if there's any relationship or trend.

```
# Scatter Plot: Age vs Overall rating
sns.scatterplot(x='Age', y='Overall', data=df)
plt.title("Scatter Plot: Age vs Overall")
plt.xlabel("Age")
plt.ylabel("Overall Rating")
plt.show()
```

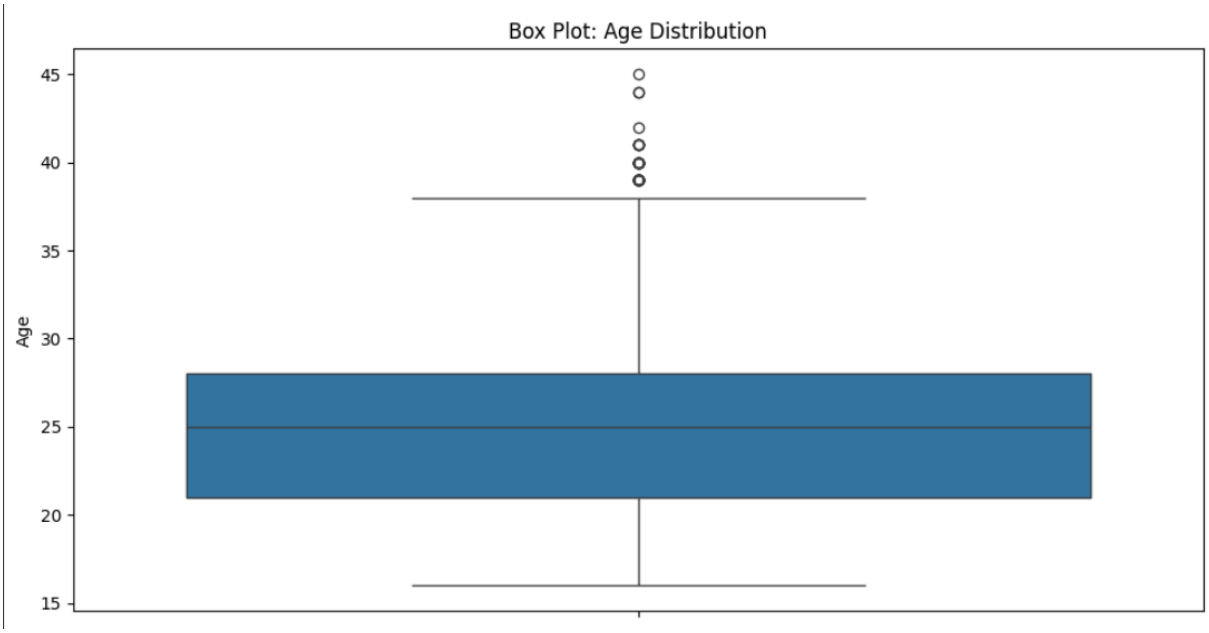


Interpretation:
The scatter plot shows player **Overall** ratings at different **Ages**. Generally, there is a concentration of higher overall ratings in the mid-20s to early 30s, though the plot suggests that many younger players still have decent ratings, and some older players also maintain high ratings.

Box Plot

We plot a box plot for **Age** to understand its distribution and check for outliers.

```
# Box Plot: Distribution of Age
sns.boxplot(y='Age', data=df)
plt.title("Box Plot: Age Distribution")
plt.ylabel("Age")
plt.show()
```



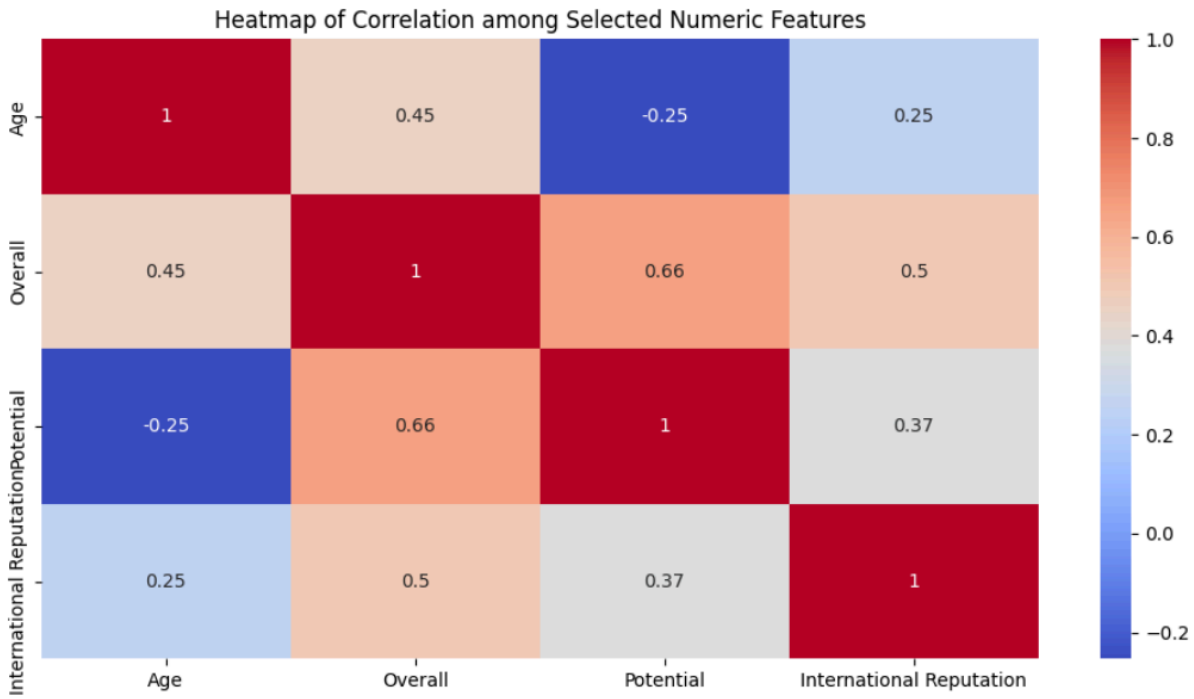
Interpretation:

- The box plot indicates that the **median age** is around the mid-20s.
- The majority of players fall between early 20s (Q1) and late 20s (Q3).
- There are a few outliers above age 35, indicating older players are relatively rare.

Heatmap

We create a heatmap of correlations among a few numeric features: **Age**, **Overall**, **Potential**, and **International Reputation**.

```
# Heatmap: Correlation among selected numeric features
numeric_cols = ["Age", "Overall", "Potential", "International Reputation"]
corr = df[numeric_cols].corr()

sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title("Heatmap of Correlation among Selected Numeric Features")
plt.show()
```

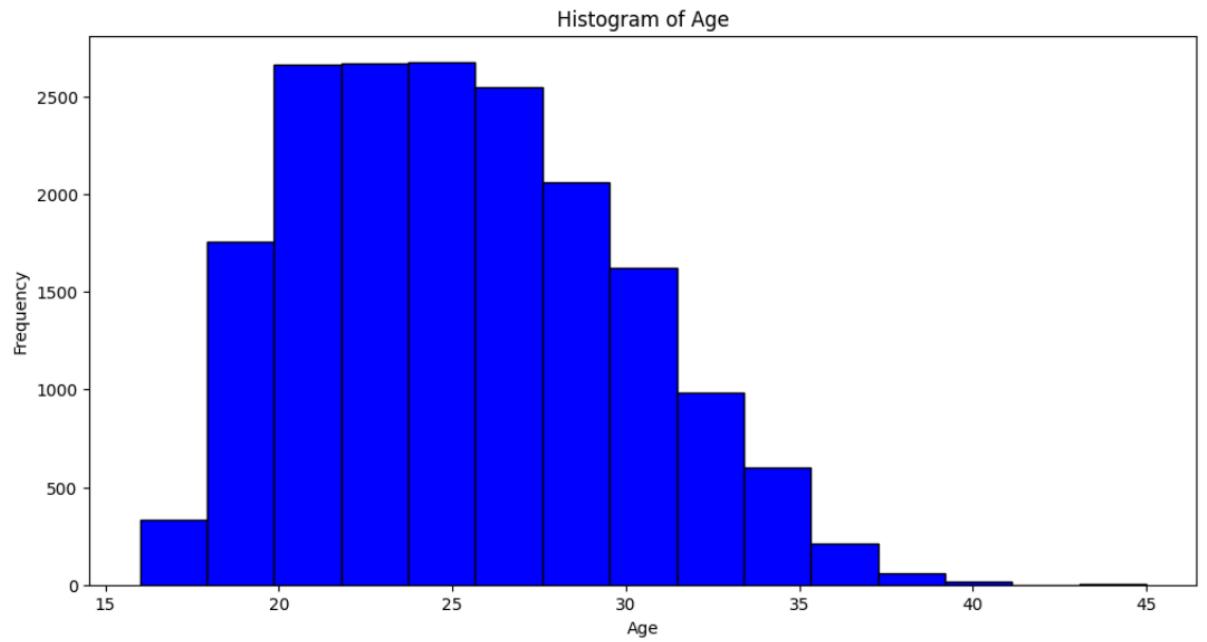


- **Interpretation:**
  - **Overall** and **Potential** are moderately positively correlated (e.g., ~0.66), suggesting that players with a higher overall rating often have high potential as well.
  - **Age** shows a slight negative correlation with **Potential** (-0.25), which makes sense: younger players often have higher potential.
  - **International Reputation** has some positive correlation with **Overall** and **Potential**, indicating that more reputable players tend to have higher overall and potential ratings.

4. Histogram and Normalized Histogram

Histogram of Age

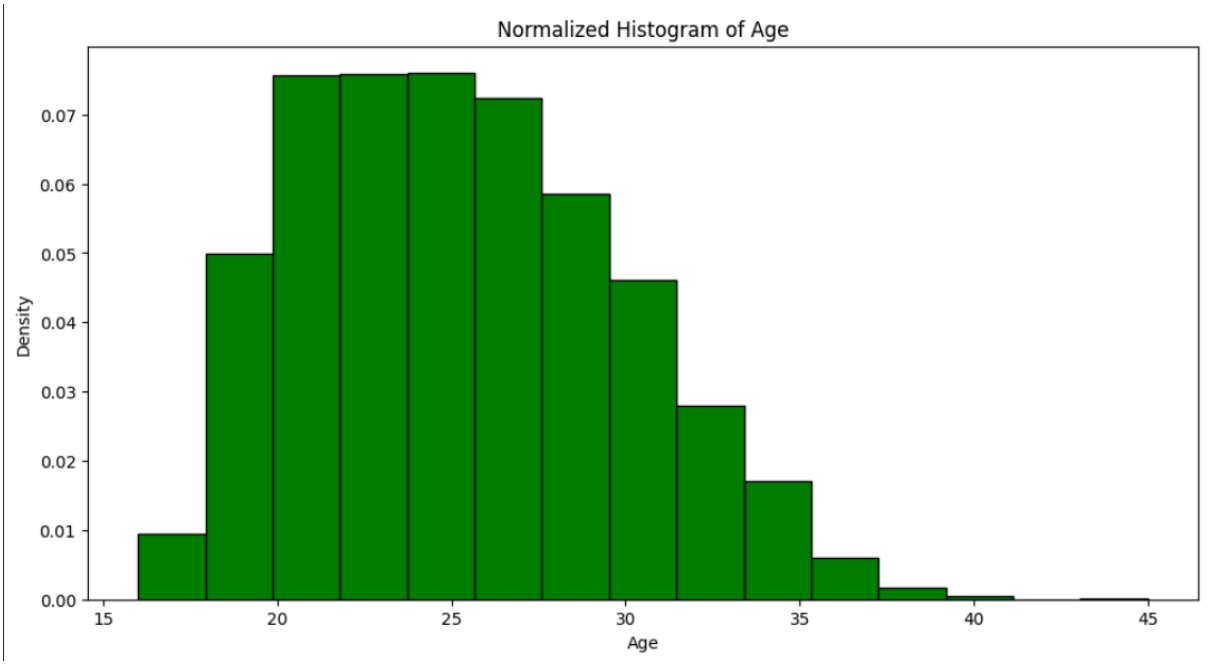
```
# Histogram of Age
plt.hist(df["Age"].dropna(), bins=15, color='blue', edgecolor='black')
plt.title("Histogram of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```



Interpretation:
This histogram shows the frequency of players at each age bracket. We can see a peak in the early-to-mid 20s, indicating that's where most players fall.

Normalized Histogram of Age

```
# Normalized (density) Histogram of Age
plt.hist(df["Age"].dropna(), bins=15, density=True, color='green', edgecolor='black')
plt.title("Normalized Histogram of Age")
plt.xlabel("Age")
plt.ylabel("Density")
plt.show()
```



- **Interpretation:**
  This is the same distribution as above but scaled to **density** instead of raw counts. It allows us to understand the **proportion** of players across different ages, irrespective of the total sample size.

5. Handling Outliers Using Box Plot and Interquartile Range (IQR)

Wage Data Preparation

The `Wage` column originally contains strings with currency symbols and letters (e.g., “€405K”). We convert them into numeric values.

```
def convert_wage(wage):
    # Remove currency symbols and whitespace
    wage = wage.replace('â,¬', '').replace('€', '').strip()
    if 'M' in wage:
        return float(wage.replace('M', '')) * 1e6
    elif 'K' in wage:
        return float(wage.replace('K', '')) * 1e3
    else:
        return float(wage)

df["Wage_numeric"] = df["Wage"].apply(convert_wage)
```

Identifying and Removing Outliers

Using the IQR method:

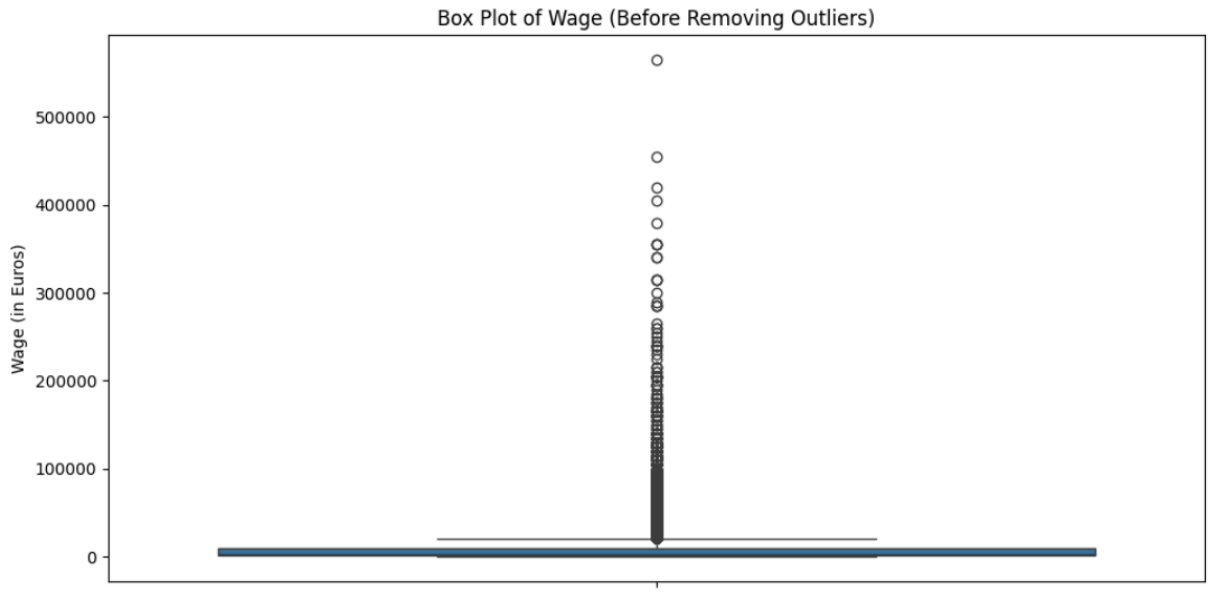
1. Compute Q1 (25th percentile) and Q3 (75th percentile).
2. Calculate IQR = Q3 - Q1.
3. Define **Lower Bound** = Q1 - 1.5 × IQR and **Upper Bound** = Q3 + 1.5 × IQR.
4. Filter out data points that fall outside these bounds.

```
Q1 = df["Wage_numeric"].quantile(0.25)
Q3 = df["Wage_numeric"].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print("Wage Outlier Bounds:")
```

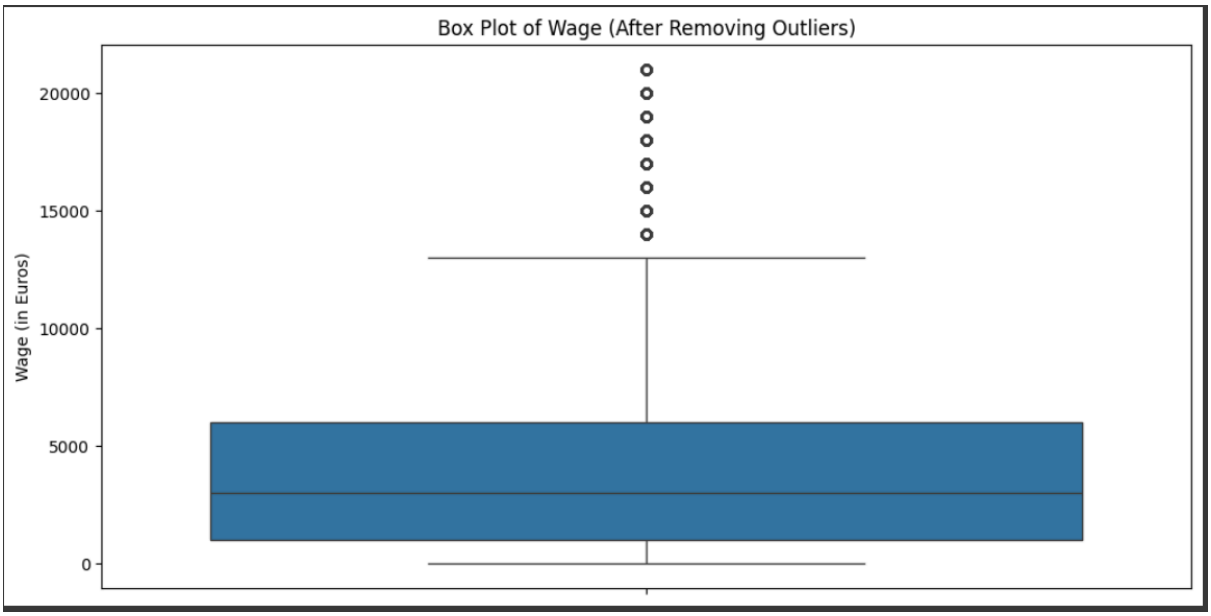
```
print("Lower bound:", lower_bound)
print("Upper bound:", upper_bound)

# Filter out the outliers
df_no_outliers = df[(df["Wage_numeric"] >= lower_bound) & (df["Wage_numeric"] <=
upper_bound)]
```

Box Plots (Before and After Removing Outliers)

```
# Box Plot Before Removing Outliers
sns.boxplot(y=df["Wage_numeric"])
plt.title("Box Plot of Wage (Before Removing Outliers)")
plt.ylabel("Wage (in Euros)")
plt.show()

# Box Plot After Removing Outliers
sns.boxplot(y=df_no_outliers["Wage_numeric"])
plt.title("Box Plot of Wage (After Removing Outliers)")
plt.ylabel("Wage (in Euros)")
plt.show()
```




```
Wage Outlier Bounds:
Lower bound: -11000.0
Upper bound: 21000.0
```



Interpretation:

- **Before Outlier Removal:** We see extremely high wage values (hundreds of thousands of Euros per week) skewing the distribution.
- **After Outlier Removal:** The distribution is more compact, and the whiskers show that most wages fall below around 20,000–21,000 Euros. Outlier removal helps us analyze the majority of players without the extreme effects of a few exceptionally high wages.

Conclusion :

In this experiment, we explored the FIFA dataset using various EDA techniques. A bar graph and contingency table showed that right-footed players dominate most positions, though certain defensive roles feature notable counts of left-footed players. Scatter plots, box plots, and heatmaps revealed that player age peaks in the early-to-mid 20s, with a moderate correlation between overall rating and potential. Histograms confirmed the age distribution skewing heavily toward the early 20s. Finally, identifying and removing wage outliers via the IQR method highlighted how a few extreme values can distort the data, allowing a clearer focus on general trends.