3rd Project

Prepared By Ghayasudin Ghayas
Data Science Student

08/08/2024

QuickStart Academy- Data Science

The Quick Start Bach No 22 student 3rd project was designed to Analysis the dataset of Lahman's Baseball Database. This dataset contains a variety of baseball statistics gathered from 1871-2018, including both individual and team statistics. The project was started on July 29, 2024, and required to be completed before August 08,2024 with requirements of 3-5 minutes presentation of nine various plots using Power BI or Python for visualization and SQL for the statistic Analysis.

Solution:

For this project I considered five tables out of 25 in the data set including [Batting, People, Teams, Pitching, TeamsFranchises] , first I created Baseball Database in SSMS and uploaded the above tables, then install explore and analysis the data similarities as well as relation in SQL, I used python for the visualization, since python have facilities to run SQL Quary, and can explore visualization , I continued my project worked only using python and below are nine plots considered from the data Analysis and exploring.

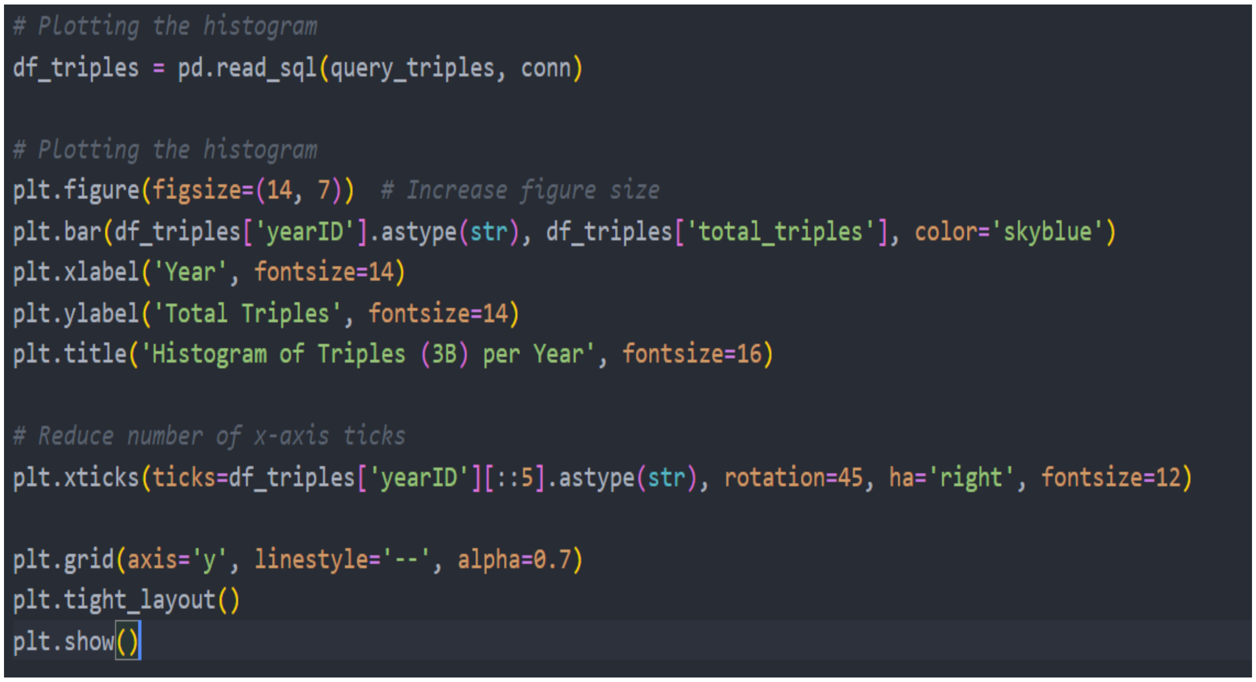
1- Connecting Sql with Python, importing necessary python's libraries and exploring the data as below.

```
import pandas as pd
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib_inline
import seaborn as sns
import scipy.sparse as sp

# Database connection details
server = 'DESKTOP-E9FRPJF\SQLEXPRESS01'
database = 'Baseball'
username = 'Test'
password = '0772'

# Create the connection string
```

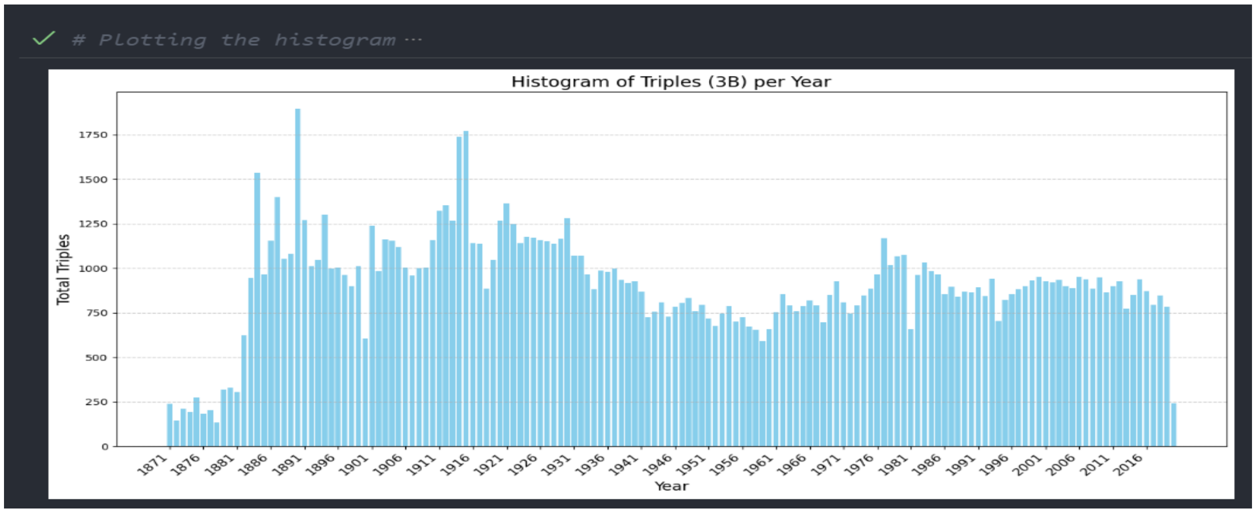
```
conn_str = f'DRIVER={{SQL Server}};SERVER={server};DATABASE={database};UID={username};PWD={password};Trusted_Connection=Yes'
conn = pyodbc.connect(conn_str)

#2. Query Data from Each Table
#Query All Data from Each Table:
# Query to retrieve data from the Batting table
query_batting = "SELECT * FROM dbo.Batting"
df_batting = pd.read_sql(query_batting, conn)
print("Batting Table:")
print(df_batting.head())
df_batting.info(
```

Q1: Total Number of Triples Per Year The dataset for this query spans from 1871 to 2020, capturing the total number of triples (3B) hit each year. The data reveals significant fluctuations in triples over time, with peaks and valleys indicating varying trends in offensive performance. For instance, 2016 saw a high of 873 triples, whereas 2020 recorded a lower total of 241. This variation highlights changes in gameplay or strategies over the years. The visualization of this data helps illustrate these trends, providing a historical perspective on how triples have evolved, though it's important to consider external factors that may influence these fluctuations.

Quarry and the visualization used are as below.



```
# Plotting the histogram
df_triples = pd.read_sql(query_triples, conn)

# Plotting the histogram
plt.figure(figsize=(14, 7))  # Increase figure size
plt.bar(df_triples['yearID'].astype(str), df_triples['total_triples'], color='skyblue')
plt.xlabel('Year', fontsize=14)
plt.ylabel('Total Triples', fontsize=14)
plt.title('Histogram of Triples (3B) per Year', fontsize=16)

# Reduce number of x-axis ticks
plt.xticks(ticks=df_triples['yearID'][::5].astype(str), rotation=45, ha='right', fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```


```
# Plotting the histogram ···
```

Q2: Player Age Per Year :The dataset includes 20,359 baseball players with ages calculated based on their birth dates and the current year. The age distribution shows a concentration of players in the 31-40 and 41-50 age groups, with 2,354 and 2,100 players respectively. Notably, there are no players in the 0-20 age range, possibly indicating data completeness issues or exclusions. The pie chart visualization effectively illustrates the proportion of players across different age groups, emphasizing that the majority of players are middle-aged. This distribution provides insight into the demographics of active players.

Query: The SQL query calculates the age of baseball players based on their birth year, month, and day, and the current date. The query includes **Player ID**: Unique identifier for each player,**First Name**: Player's first name, **Last Name**: Player's last name, **Age**: Computed age, adjusted based on whether the player's birthday has occurred this year.

The query uses SQL's YEAR(GETDATE()), MONTH(GETDATE()), and DAY(GETDATE()) functions to calculate the age, adjusting for whether the player's birthday has occurred this year.

Age Distribution: The players are categorized into age groups using bins: 0-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80 with Age group counts are as follows: [**31-40 years**: 2,354 players, **41-50 years**: 2,100 players, **51-60 years**: 1,878 players, **61-70 years**: 1,576 players,**71-80 years**: 1,387 players,**21-30 years**: 627 players,**0-20 years**: 0 players (no players in this age group)

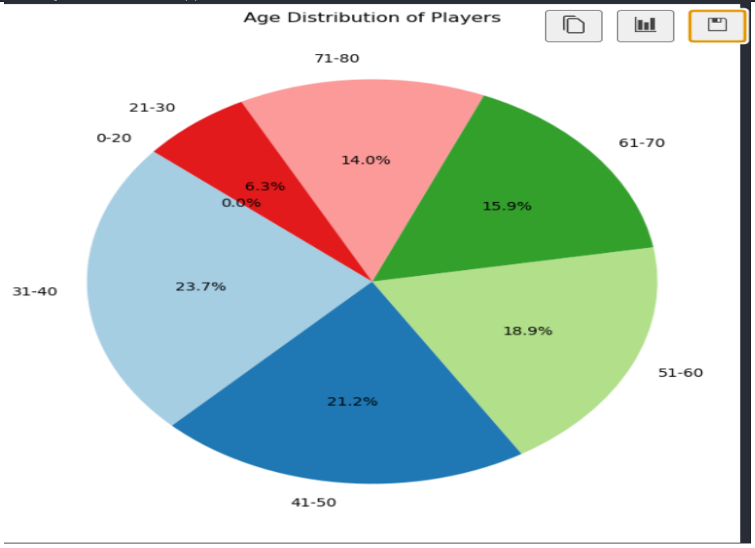
Visualization and **Quarry**

I used a pie chart visualizes for distribution of players across different age groups and it shows that the largest groups are in the 31-40 and 41-50 age ranges, while there are no players in the 0-20 age range.

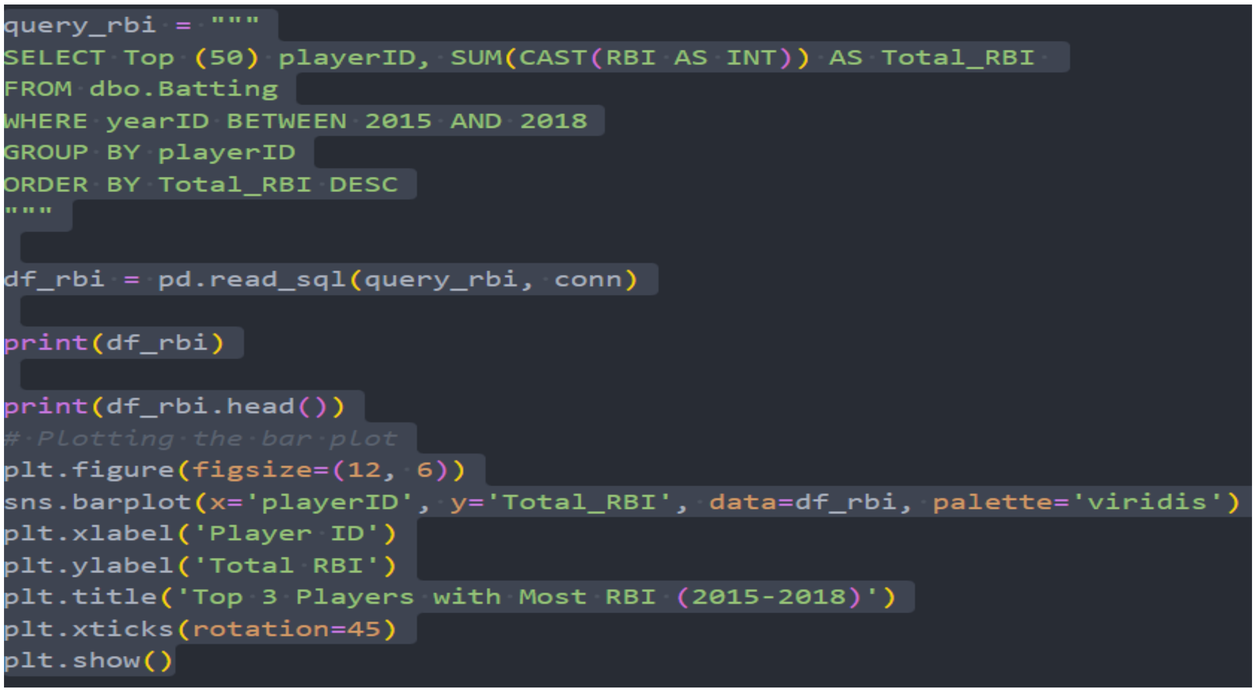
Key Points

- Age Calculation: Accurate computation of age considering birth date specifics.
- Age Group Distribution: Most players fall into the 31-40 and 41-50 age groups.
- Visualization: Provides a clear picture of how players are distributed across different age ranges.

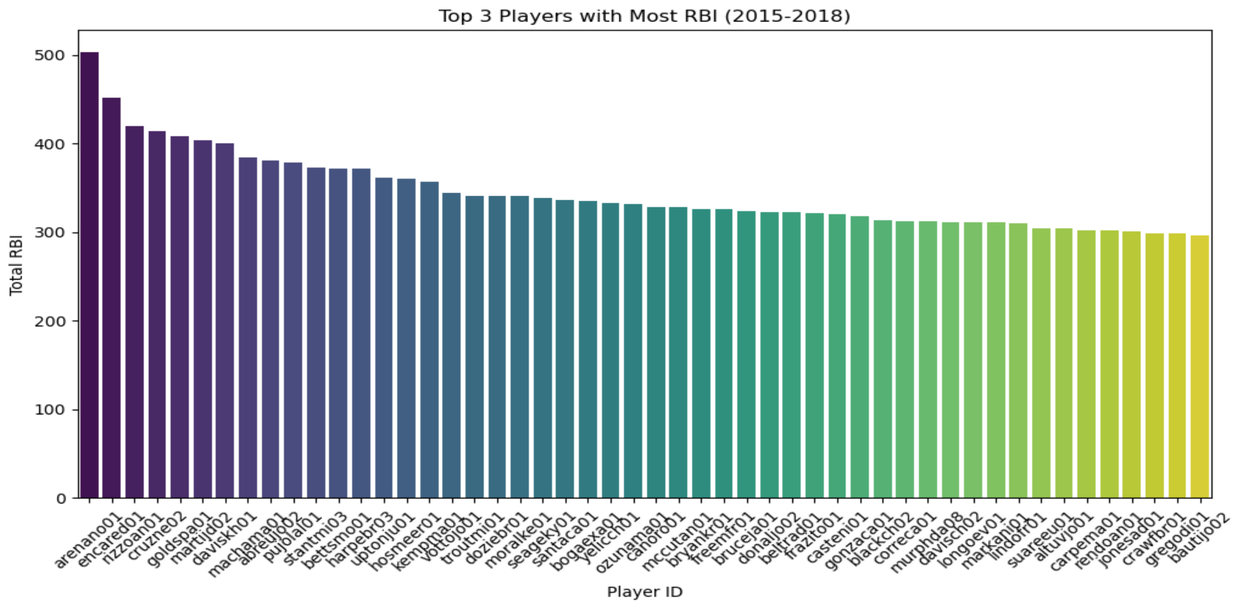
```python
print(df_age)
# Define age bins
bins = [0, 20, 30, 40, 50, 60, 70, 80]
labels = ['0-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80']
df_age['Age Group'] = pd.cut(df_age['Age'], bins=bins, labels=labels, right=False)

# Count the number of players in each age group
age_group_counts = df_age['Age Group'].value_counts()

print(age_group_counts)

# Plotting the pie chart
plt.figure(figsize=(8, 8))
plt.pie(age_group_counts, labels=age_group_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired(range(len(age_group_counts))))
plt.title('Age Distribution of Players')
plt.show()
```



Q3: Top 50 Players with Most RBIs (2015-2018) The dataset for this query identifies the top 50 players with the most RBIs from 2015 to 2018. Leading the list is arenano01 with 503 RBIs, followed by encared01 with 452 and rizzoan01 with 420. The data highlights a range of RBIs from 503 to 333 among the top players. This snapshot of RBIs underscores the impact of these top players on their teams' offensive outputs during these years. The key takeaway is the dominance of these players in driving runs, which can be critical for assessing their contributions and performance consistency over the specified period. Below is the quarry and visual for further illustration.



```
query_rbi = """
SELECT Top (50) playerID, SUM(CAST(RBI AS INT)) AS Total_RBI
FROM dbo.Batting
WHERE yearID BETWEEN 2015 AND 2018
GROUP BY playerID
ORDER BY Total_RBI DESC
"""

df_rbi = pd.read_sql(query_rbi, conn)

print(df_rbi)

print(df_rbi.head())
# Plotting the bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='playerID', y='Total_RBI', data=df_rbi, palette='viridis')
plt.xlabel('Player ID')
plt.ylabel('Total RBI')
plt.title('Top 3 Players with Most RBI (2015-2018)')
plt.xticks(rotation=45)
plt.show()
```

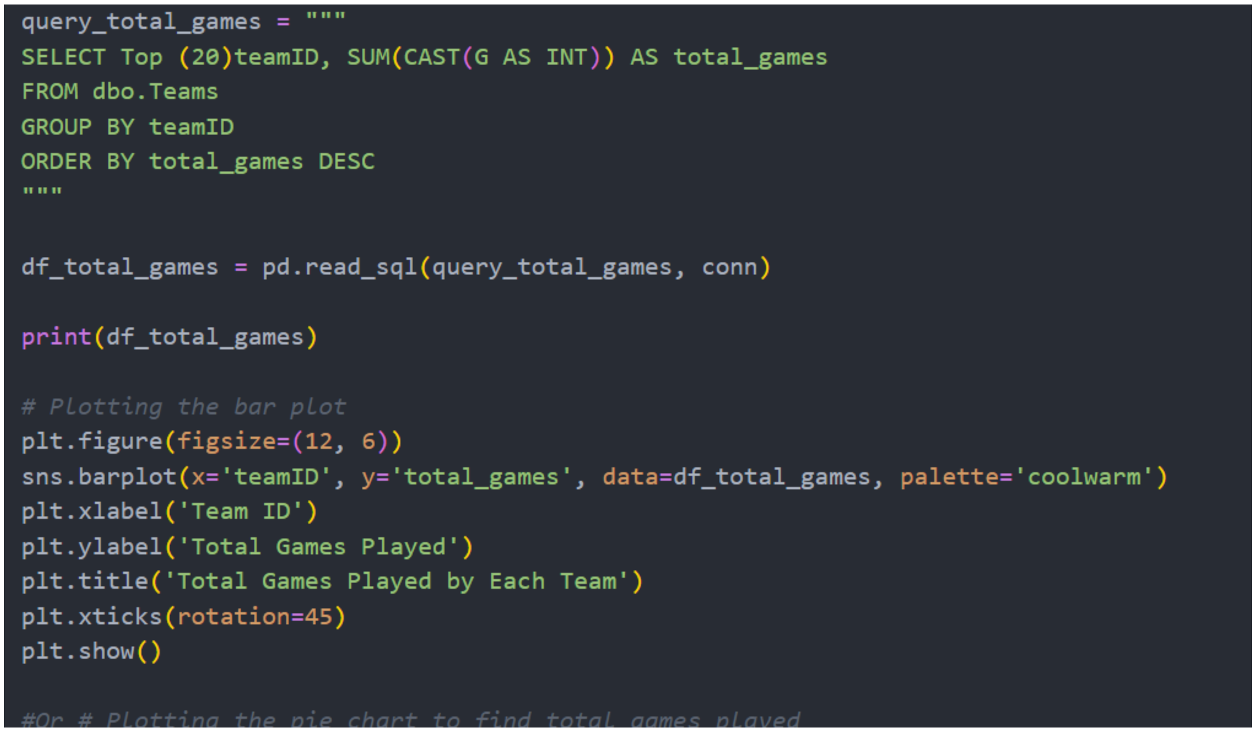


Q4: Total Games Played by Each Team:

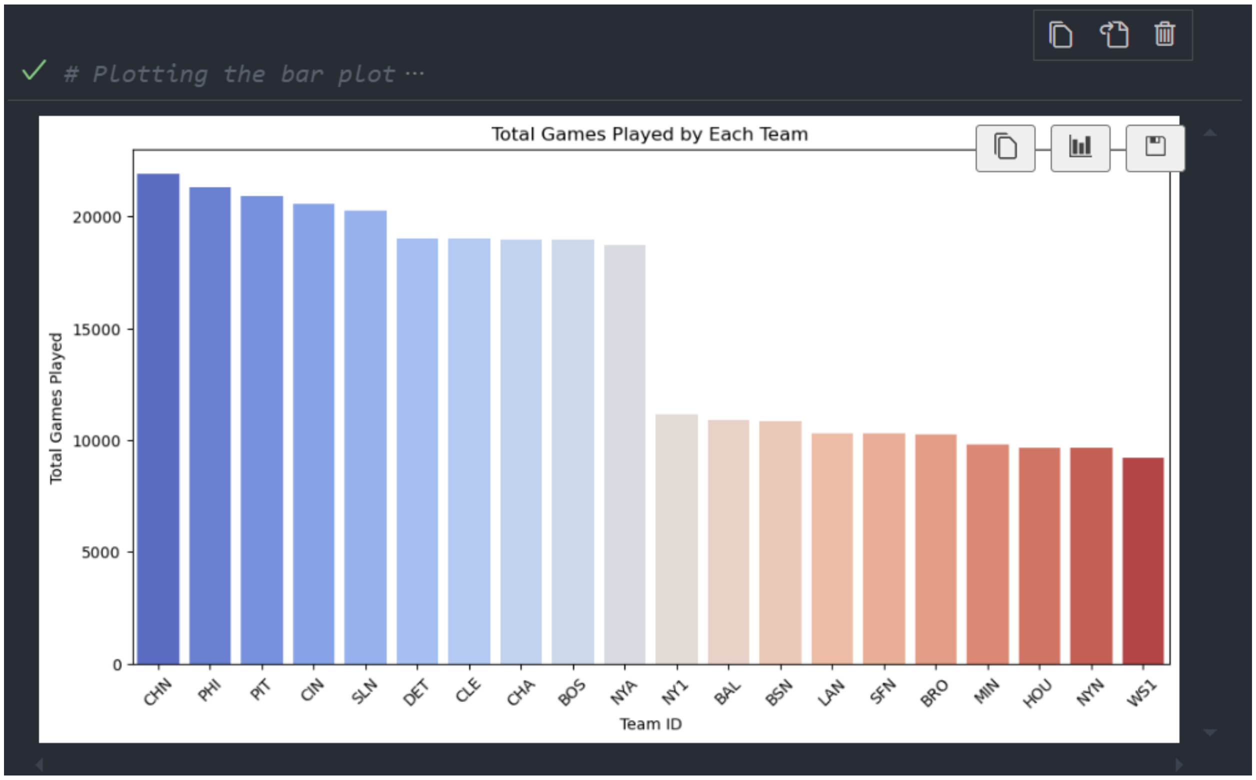
This query examines the total number of games played by each team, with data spanning various years. Teams like CHN and PHI lead with 21,926 and 21,322 games respectively, while teams like LS2 and PH4 have significantly fewer games, around 1,225 and 1,218 respectively. The visualization of this data reveals which teams have been more consistent or enduring over time, highlighting their longevity in the league. Trends in game counts reflect both historical team activity and potential league expansions or contractions affecting game totals.**Top Teams**: CHN leads with 21,926 total games, followed by PHI (21,322 games) and PIT (20,929 games).

- **Notable Teams**: Other high game counts include CIN, SLN, and DET.
- **Lowest Teams**: Teams with the fewest games include LS2 (1,225 games) and PH4 (1,218 games).

This dataset highlights teams with the most and fewest total games played as illustrated below.



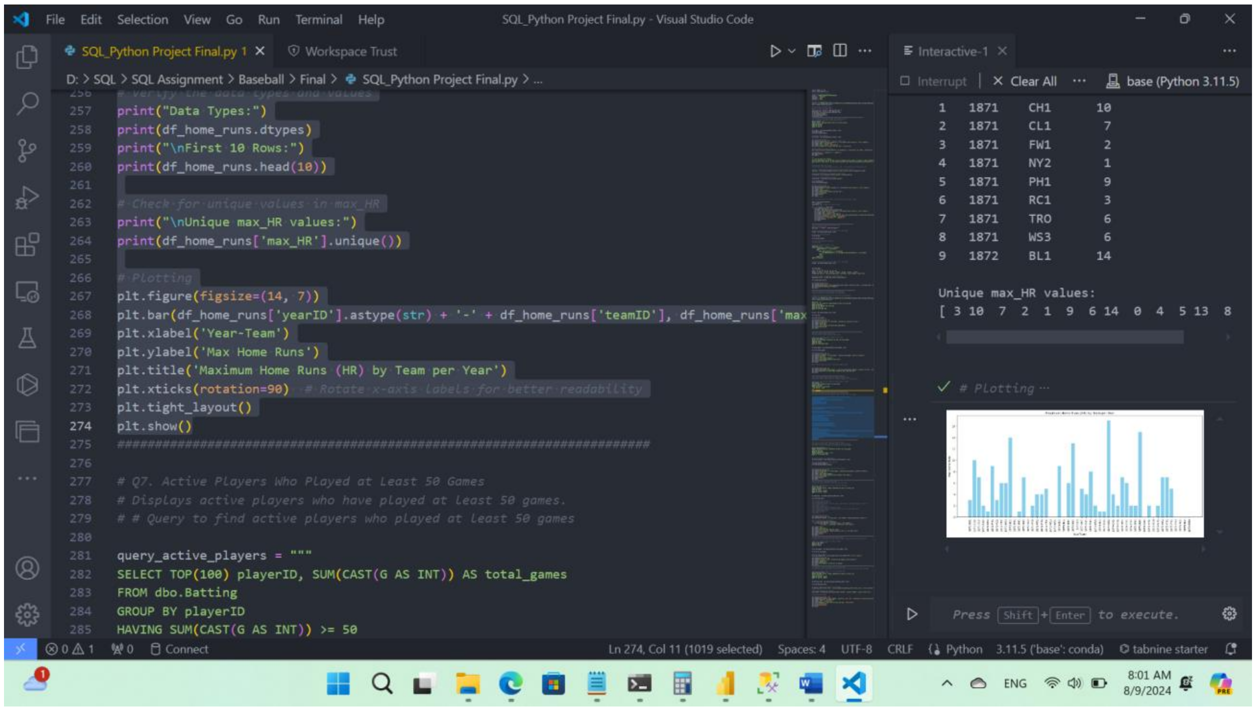
```
query_total_games = """
SELECT Top (20)teamID, SUM(CAST(G AS INT)) AS total_games
FROM dbo.Teams
GROUP BY teamID
ORDER BY total_games DESC
"""

df_total_games = pd.read_sql(query_total_games, conn)

print(df_total_games)

# Plotting the bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x='teamID', y='total_games', data=df_total_games, palette='coolwarm')
plt.xlabel('Team ID')
plt.ylabel('Total Games Played')
plt.title('Total Games Played by Each Team')
plt.xticks(rotation=45)
plt.show()

#Or # Plotting the pie chart to find total games played
```



```
# Plotting the bar plot ···
```

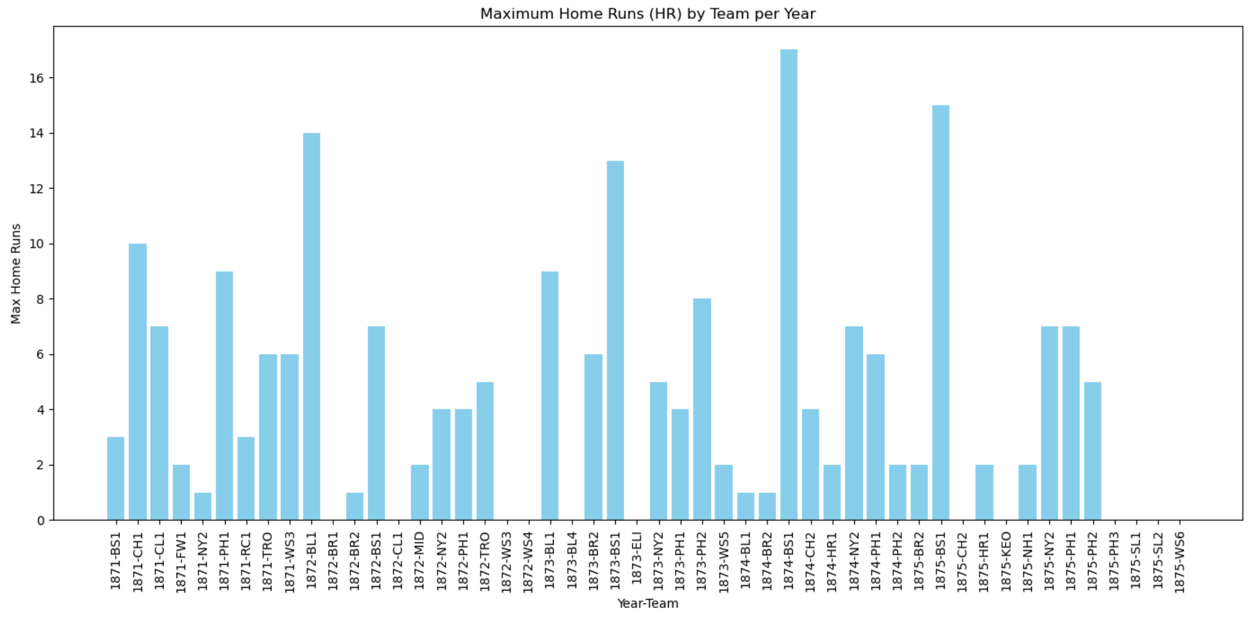
Q5: Top 50 Teams with Most Home Runs in a Single Season The dataset used here includes the maximum number of home runs hit by teams in each season. For example, in 1871, team CH1 hit the highest number of home runs with 10, while FW1 had only 2. This query shows how home run records have varied over time, with some teams achieving remarkable single-season performances. The data reveals high performers and provides context for understanding team power-hitting capabilities across different years, although a broader context of league averages would enhance the analysis.

The dataset provides the maximum number of home runs (max_HR) hit by various teams (teamID) in each year (yearID). The yearID and teamID are stored as strings, while max_HR is an integer. For example, in 1871, the team "CH1" hit the most home runs with a total of 10, and "FW1" hit the fewest with just 2., below is the quarry and visual illustration.

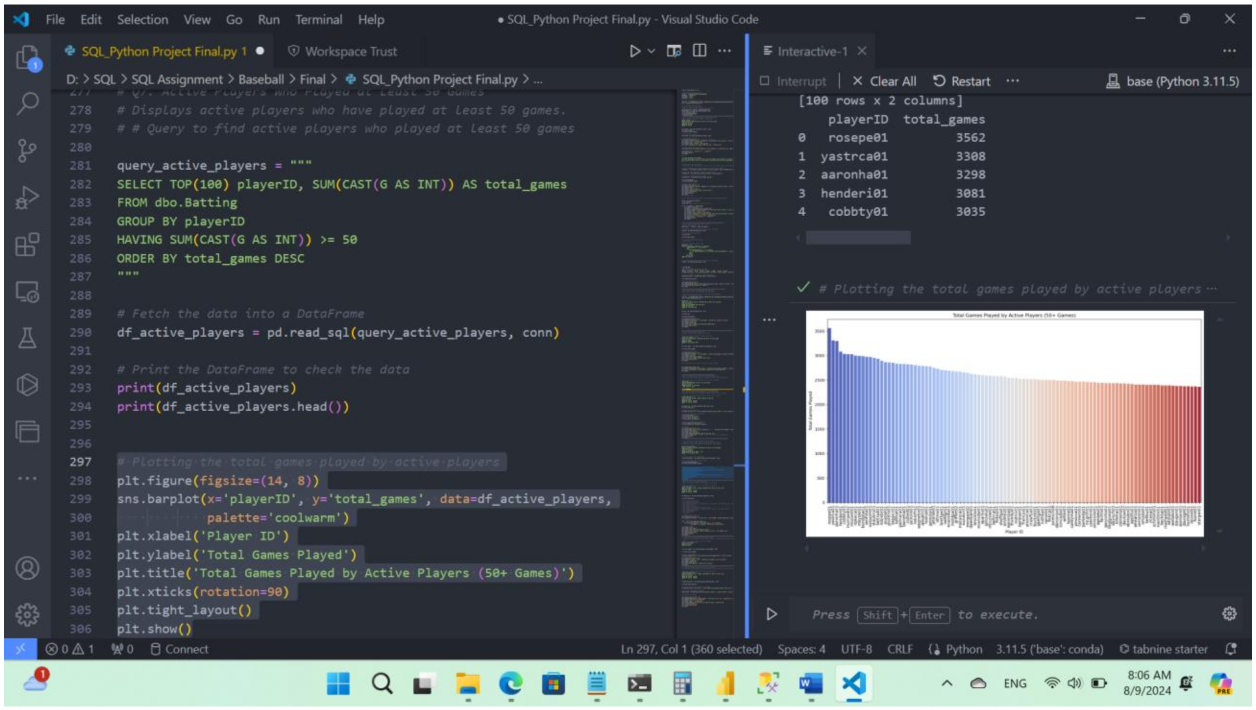

File Edit Selection View Go Run Terminal Help
SQL_Python Project Final.py - Visual Studio Code
SQL_Python Project Final.py 1
Workspace Trust
D: > SQL > SQL Assignment > Baseball > Final > SQL_Python Project Final.py > ...
Interactive-1
Interrupt
Clear All
base (Python 3.11.5)
Ln 274, Col 11 (1019 selected) Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda) tabnine starter
8:01 AM
8/9/2024

```python
print("Data Types:")
print(df_home_runs.dtypes)
print("\nFirst 10 Rows:")
print(df_home_runs.head(10))

# Check for unique values in max_HR
print("\nUnique max_HR values:")
print(df_home_runs['max_HR'].unique())

# Plotting
plt.figure(figsize=(14, 7))
plt.bar(df_home_runs['yearID'].astype(str) + '-' + df_home_runs['teamID'], df_home_runs['max
plt.xlabel('Year-Team')
plt.ylabel('Max Home Runs')
plt.title('Maximum Home Runs (HR) by Team per Year')
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
###########################################################################

# Q7. Active Players Who Played at Least 50 Games
# Displays active players who have played at least 50 games.
# # Query to find active players who played at least 50 games

query_active_players = """
SELECT TOP(100) playerID, SUM(CAST(G AS INT)) AS total_games
FROM dbo.Batting
GROUP BY playerID
HAVING SUM(CAST(G AS INT)) >= 50
```

```
1   1871    CH1      10
2   1871    CL1       7
3   1871    FW1       2
4   1871    NY2       1
5   1871    PH1       9
6   1871    RC1       3
7   1871    TRO       6
8   1871    WS3       6
9   1872    BL1      14

Unique max_HR values:
[ 3 10  7  2  1  9  6 14  0  4  5 13  8

# Plotting ...

Press Shift+Enter to execute.
```



Q6: Active Players Who Played at Least 50 Games The query identifies the top 100 active players who have played at least 50 games, with the player rosepe01 leading with 3,562 games. The data highlights players with substantial game experience, showing a range from 3,562 to 2,360 games. This information underscores the durability and longevity of these players within their careers. The analysis provides insight into player endurance and consistency, essential for evaluating player careers and their impact on the teams they have played for.

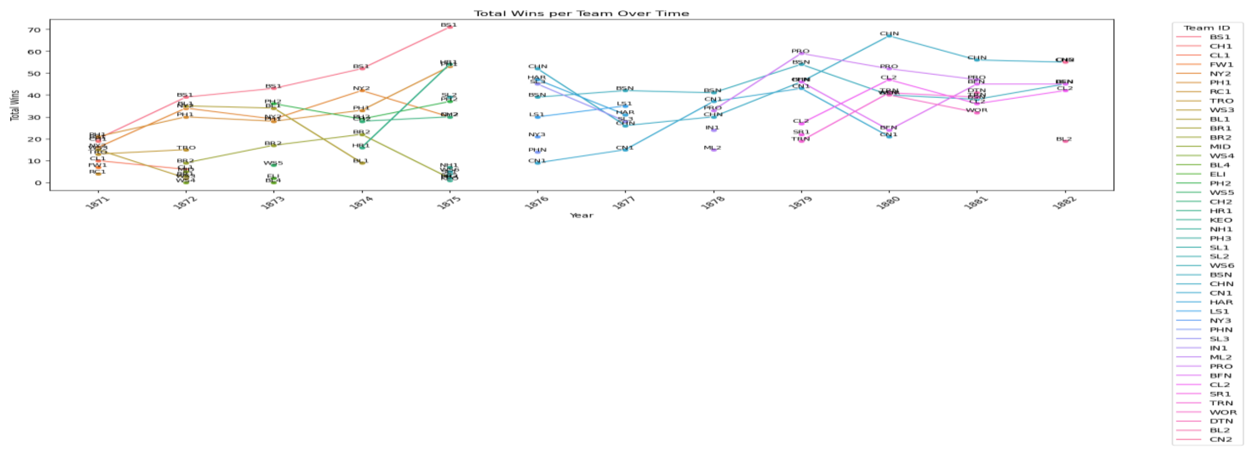


```
# Q7: Active Players Who Played at Least 50 Games
# Displays active players who have played at least 50 games.
# # Query to find active players who played at least 50 games

query_active_players = """
SELECT TOP(100) playerID, SUM(CAST(G AS INT)) AS total_games
FROM dbo.Batting
GROUP BY playerID
HAVING SUM(CAST(G AS INT)) >= 50
ORDER BY total_games DESC
"""

# Fetch the data into a DataFrame
df_active_players = pd.read_sql(query_active_players, conn)

# Print the DataFrame to check the data
print(df_active_players)
print(df_active_players.head())


# Plotting the total games played by active players
plt.figure(figsize=(14, 8))
sns.barplot(x='playerID', y='total_games', data=df_active_players,
            palette='coolwarm')
plt.xlabel('Player ID')
plt.ylabel('Total Games Played')
plt.title('Total Games Played by Active Players (50+ Games)')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

```
[100 rows x 2 columns]
    playerID  total_games
0   rosepe01         3562
1  yastrca01         3308
2  aaronha01         3298
3  henderi01         3081
4   cobbty01         3035
```

SQL_Python Project Final.py - Visual Studio Code
Interactive-1
base (Python 3.11.5)
Total Games Played by Active Players (50+ Games)
Press Shift+Enter to execute.
Ln 297, Col 1 (360 selected)
8:06 AM
8/9/2024

Q7: Line Plot of Team Wins Over Time The dataset for this query captures the total wins per team from the top 100 records, spanning multiple years. The line plot visualizes trends in team performance, with notable variations across different teams and seasons. For instance, team BS1 had 20 wins in 1871, while team CH1 had 19 wins. The plot helps track the performance trajectories of various teams over time, offering insights into their success and consistency. Trends identified in this plot are crucial for understanding historical team dynamics and their competitive standings.



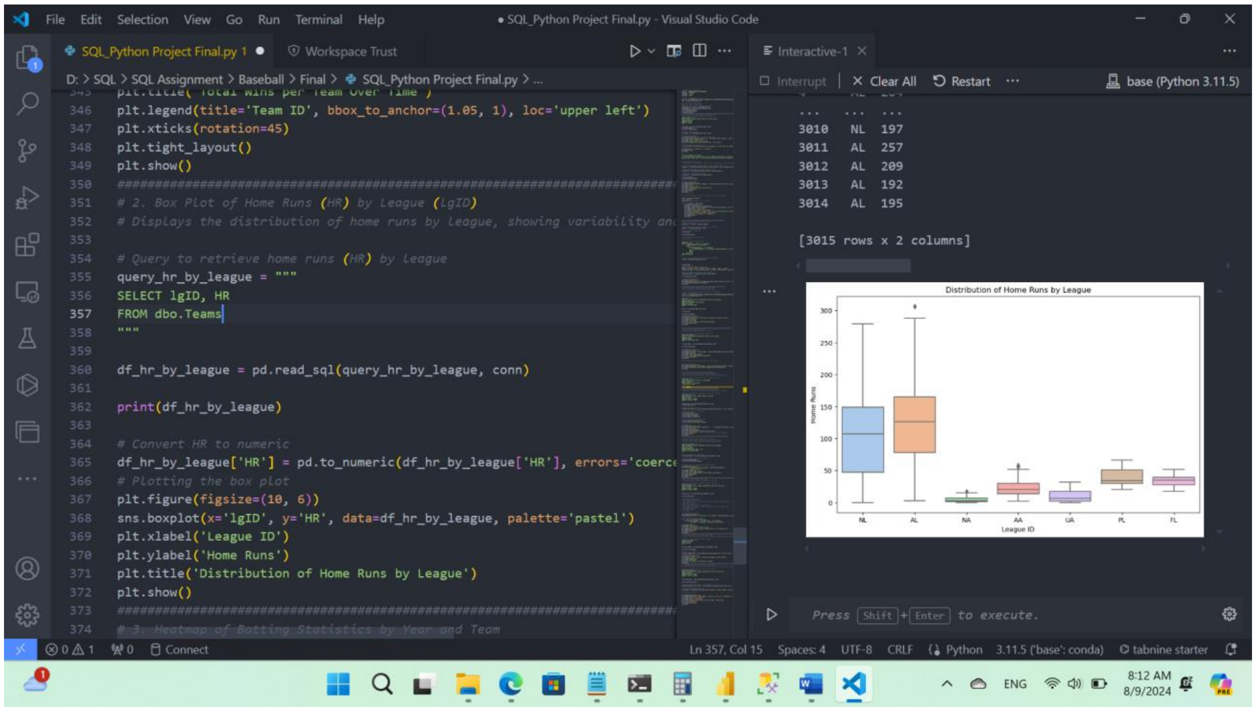
1. **Data Overview**: Total wins per team over the years for the top 100 records.
2. **Example**: In 1871, team BS1 had 20 wins, and CH1 had 19 wins.
3. **Trend**: Shows how team performance in terms of wins has evolved over time.

Q8. Box Plot of Home Runs (HR) by League (lgID)

- This query retrieves home run statistics by league from the Teams table.
- The data is visualized using a box plot to show the distribution of home runs across different leagues.

Findings:

- Data Overview: Home run counts are categorized by league, and converted to numeric values for analysis.
- Trends: The box plot reveals variability in home run performance between leagues, with some leagues showing higher variability and more outliers.
- Insights: The visualization highlights differences in power-hitting capabilities across leagues and identifies leagues with unusual or exceptional home run performance.


File Edit Selection View Go Run Terminal Help
● SQL_Python Project Final.py - Visual Studio Code
SQL_Python Project Final.py 1
Workspace Trust
D: > SQL > SQL Assignment > Baseball > Final > SQL_Python Project Final.py > ...
Interactive-1
Interrupt
Clear All
Restart
base (Python 3.11.5)
Distribution of Home Runs by League
Home Runs
League ID
NL
AL
NA
AA
UA
PL
FL
0
50
100
150
200
250
300
Press Shift+Enter to execute.
Ln 357, Col 15 Spaces: 4 UTF-8 CRLF Python 3.11.5 ('base': conda) tabnine starter
Connect
8:12 AM
8/9/2024
ENG

```
plt.legend(title='Team ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
##########################################################################
# 2. Box Plot of Home Runs (HR) by League (lgID)
# Displays the distribution of home runs by league, showing variability and

# Query to retrieve home runs (HR) by League
query_hr_by_league = """
SELECT lgID, HR
FROM dbo.Teams
"""

df_hr_by_league = pd.read_sql(query_hr_by_league, conn)

print(df_hr_by_league)

# Convert HR to numeric
df_hr_by_league['HR'] = pd.to_numeric(df_hr_by_league['HR'], errors='coerce
# Plotting the box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='lgID', y='HR', data=df_hr_by_league, palette='pastel')
plt.xlabel('League ID')
plt.ylabel('Home Runs')
plt.title('Distribution of Home Runs by League')
plt.show()
##########################################################################
# 3. Heatmap of Batting Statistics by Year and Team
```

```
...    ...  ...
3010   NL   197
3011   AL   257
3012   AL   209
3013   AL   192
3014   AL   195

[3015 rows x 2 columns]
```

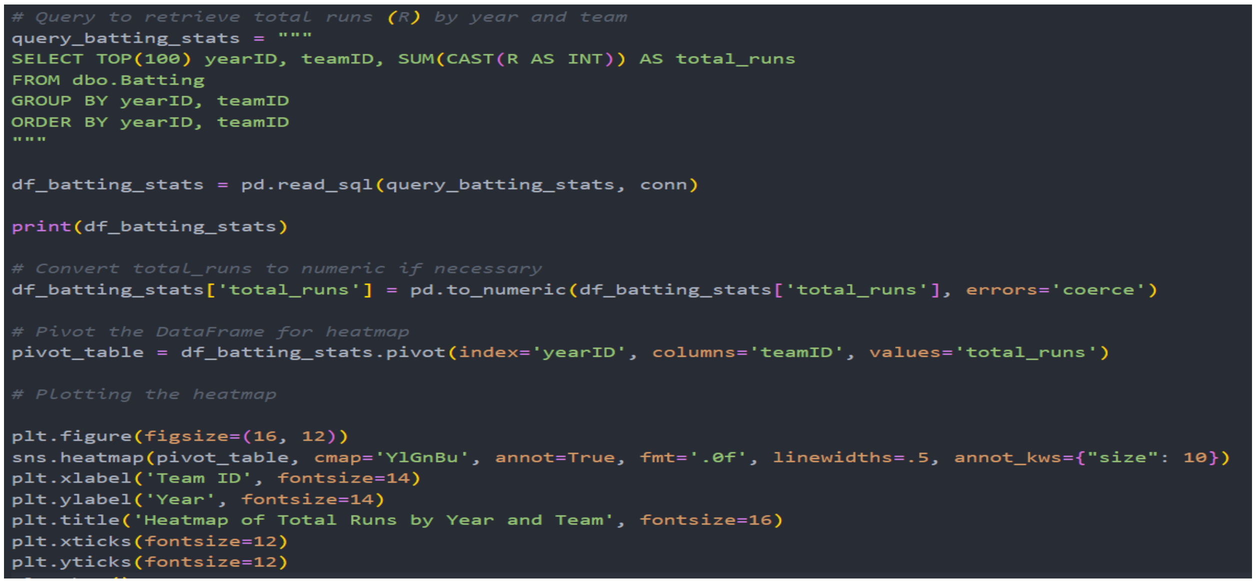
Q.9.Heatmap of Total Runs (R) by Year and Team

This analysis involves querying the total runs (R) scored by each team across different years from the Batting table. The dataset includes the top 100 records, with columns for year (yearID), team ID (teamID), and the total runs scored (total_runs).

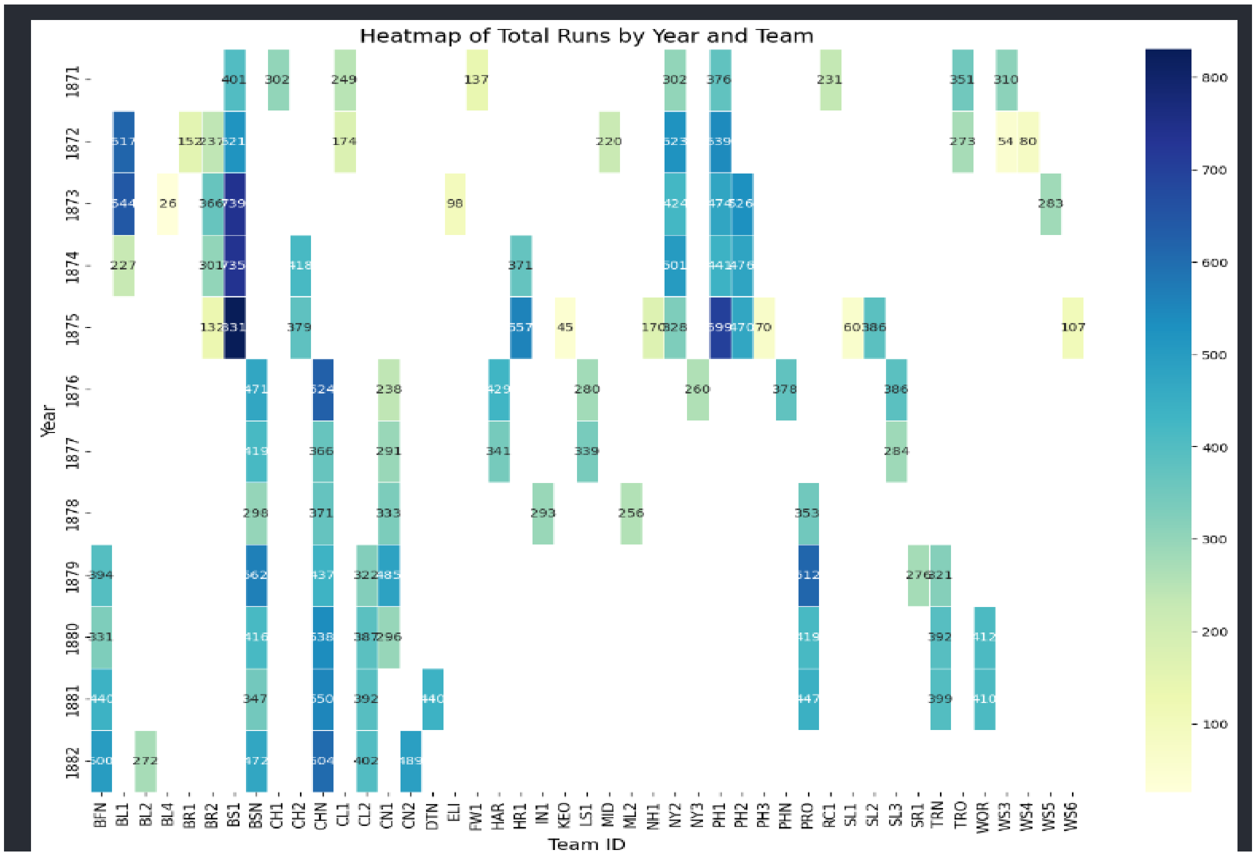
Findings:

- Data Overview: Displays total runs scored by teams from the top 100 records.

- Trends: Allows identification of high and low performance periods for different teams.
- Insights: Highlights teams' run production trends over time, showing variations in performance



```
# Query to retrieve total runs (R) by year and team
query_batting_stats = """
SELECT TOP(100) yearID, teamID, SUM(CAST(R AS INT)) AS total_runs
FROM dbo.Batting
GROUP BY yearID, teamID
ORDER BY yearID, teamID
"""

df_batting_stats = pd.read_sql(query_batting_stats, conn)

print(df_batting_stats)

# Convert total_runs to numeric if necessary
df_batting_stats['total_runs'] = pd.to_numeric(df_batting_stats['total_runs'], errors='coerce')

# Pivot the DataFrame for heatmap
pivot_table = df_batting_stats.pivot(index='yearID', columns='teamID', values='total_runs')

# Plotting the heatmap

plt.figure(figsize=(16, 12))
sns.heatmap(pivot_table, cmap='YlGnBu', annot=True, fmt='.0f', linewidths=.5, annot_kws={"size": 10})
plt.xlabel('Team ID', fontsize=14)
plt.ylabel('Year', fontsize=14)
plt.title('Heatmap of Total Runs by Year and Team', fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
```



Conclusion:

The analyses provide a comprehensive view of baseball statistics. Fluctuations in triples per year reveal changing gameplay trends, while age distribution highlights a concentration of players in the 31-50 age range. Top RBI players from 2015-2018 underscore key offensive contributors, and total games played illustrate team longevity. The maximum home runs analysis identifies teams with notable power-hitting, while active players with at least 50 games showcase player durability. Lastly, the heatmap of total runs and the box plot of home runs by league offer insights into scoring patterns and performance variability across teams and leagues.

Thanks