Data Science Project 2 Baseball Statistics

Imama Mansoor

QuickStart Data Science Master Bootcamp

OBJECTIVE

The objective of this project is to analyze baseball statistics and to answer some given questions including with some additional questions as well. The main purpose of this study is also to find some interesting insights into this huge baseball dataset.

INTRODUCTION

Baseball is a bat-and-ball game played between two opposing teams who take turns batting and fielding. The teams comprises of nine players. The game proceeds when a player on the fielding team, called the pitcher, throws a ball which a player on the batting team tries to hit with a bat. The objective of the offensive team (batting team) is to hit the ball into the field of play, allowing its players to run the bases, having them advance counter-clockwise around four bases to score what are called "runs". The objective of the defensive team (fielding team) is to prevent batters from becoming runners, and to prevent runners' advance around the bases. A run is scored when a runner legally advances around the bases in order and touches home plate (the place where the player started as a batter). The team that scores the most runs by the end of the game is the winner.

Baseball Statistics

The Official Baseball Rules administered by MLB require the official scorer to categorize each baseball play unambiguously. The rules provide detailed criteria to promote consistency. The score report is the official basis for both the box score of the game and the relevant statistical records.[136] General managers, managers, and baseball scouts use statistics to evaluate players and make strategic decisions. Our dataset consists of all these statistics throughout the life of baseball from 1871 to 2021.

DATASET

The Lahman Baseball Database is given for analyzing. This database contains pitching, hitting, and fielding statistics for Major League Baseball from 1871 through 2014. It includes data from the two current leagues (American and National), the four other "major" leagues (American Association, Union Association, Players League, and Federal League), and the National Association of 1871-1875.

This database was created by Sean Lahman, who pioneered the effort to make baseball statistics freely available to the general public. What started as a one man effort in 1994 has grown tremendously, and now a team of researchers have collected their efforts to make this the largest and most accurate source for baseball statistics available anywhere.

Data Tables

The design follows these general principles. Each player is assigned a unique number (playerID). All of the information relating to that player is tagged with his playerID. The playerIDs are linked to names and birthdates in the PEOPLE table. There are total 24 tables in this database.

The database is comprised of the following main tables:

- People - Player names, DOB, and biographical info
- Batting - batting statistics
- Pitching - pitching statistic

It is supplemented by some other tables, but we have only selected the above three tables and the below tables:

- Appearances - details on the positions a player appeared at
- AwardsPlayers - awards won by players

Attributes in Data Tables

The attributes in the data tables are defined below to create complete understanding to the tables.

1. **People table**

playerID -- A unique code assigned to each player. The playerID links the data in this file with records in the other files.
birthyear -- Year player was born
birthMonth -- Month player was born
birthDay -- Day player was born
birthCountry -- Country where player was born
birthState -- State where player was born
birthCity -- City where player was born
deathYear -- Year player died
deathMonth -- Month player died
deathDay -- Day player died
deathCountry -- Country where player died
deathState -- State where player died
deathCity -- City where player died
nameFirst -- Player's first name
nameLast -- Player's last name
nameGiven -- Player's given name (typically first and middle)
weight -- Player's weight in pounds
height -- Player's height in inches
bats -- Player's batting hand (left, right, or both)
throws -- Player's throwing hand (left or right)
debut -- Date that player made first major league appearance
finalGame -- Date that player made first major league appearance (blank if still active)
retroID -- ID used by retrosheet
bbrefID -- ID used by Baseball Reference website

2. **Batting Table**

playerID -- Player ID code
yearID -- Year
stint -- player's stint (order of appearances within a season)
teamID -- Team

lgID -- League
G -- Games
AB -- At Bats
R -- Runs
H -- Hits
2B -- Doubles
3B -- Triples
HR -- Homeruns
RBI -- Runs Batted In
SB -- Stolen Bases
CS -- Caught Stealing
BB -- Base on Balls
SO -- Strikeouts
IBB -- Intentional walks
HBP -- Hit by pitch
SH -- Sacrifice hits
SF -- Sacrifice flies
GIDP -- Grounded into double plays

3. Pitching Table

playerID -- Player ID code
yearID -- Year
stint -- player's stint (order of appearances within a season)
teamID -- Team
lgID -- League
W -- Wins
L -- Losses
G -- Games
GS -- Games Started
CG -- Complete Games
SHO -- Shutouts
SV -- Saves
IPOuts -- Outs Pitched (innings pitched x 3)
H -- Hits
ER -- Earned Runs
HR -- Homeruns
BB -- Walks
SO -- Strikeouts
BAOpp -- Opponent's Batting Average
ERA -- Earned Run Average
IBB -- Intentional Walks
WP -- Wild Pitches
HBP -- Batters Hit By Pitch
BK -- Balks
BFP -- Batters faced by Pitcher
GF -- Games Finished
R -- Runs Allowed
SH -- Sacrifices by opposing batters
SF -- Sacrifice flies by opposing batters
GIDP -- Grounded into double plays by opposing batter

4. AwardsPlayers Table

playerID -- Player ID code
awardID -- Name of award won
yearID -- Year
lgID -- League
tie -- Award was a tie (Y or N)
notes -- Notes about the award

5. **Appearances Table**

yearID -- Year
teamID -- Team
lgID -- League
playerID -- Player ID code
G_all -- Total games played
GS -- Games started
G_batting -- Games in which player batted
G_defense -- Games in which player appeared on defense
G_p -- Games as pitcher
G_c -- Games as catcher
G_1b -- Games as firstbaseman
G_2b -- Games as secondbaseman
G_3b -- Games as thirdbaseman
G_ss -- Games as shortstop
G_lf -- Games as leftfielder
G_cf -- Games as centerfielder
G_rf -- Games as right fielder
G_of -- Games as outfielder
G_dh -- Games as designated hitter
G_ph -- Games as pinch hitter
G_pr -- Games as pinch runner

TOOLS FOR VISUALIZATION

I have used following tools for data wrangling and analyzing:

- ✓ Python 3.9
- ✓ Microsoft Power BI

DETAILS FOR VISUALIZATION

Python 3.9

Jupyter Notebook is used for python scripting. As given in requirements of the project, i have imported all the necessary packages of python for the project and import the database of Baseball too.

```
# Importing all the necessary packages of Python

import sqlite3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Making connection to the Baseball database for its exploration

connection = sqlite3.connect("Baseball-SQL.db")
cursor = connection.cursor()
```

Now it is the time to create the data frame based on the given requirements. People, Batting and Appearances table are joined in such a way that all the attributes are retrieved from Batting table while only selected attributes are taken from People. Appearances table is only joined to check the given condition that all players should have played atleast 50 games. I have assumed that active players are those whose age are less than and equal to 45 and their death year is null.

```
# Making the Dataframe based on the data passed as a result of the SQL query

dfActive_Play50g = pd.read_sql_query("""SELECT Batting.*, People.nameFirst, People.nameLast, People.nameGiven, People.birthYear,
People.birthMonth ,People.birthDay, People.birthCountry, People.birthState, People.birthCity, People.weight, People.bats,
People.throws
FROM Appearances JOIN People
ON (Appearances.playerID = People.playerID)
JOIN Batting
ON (Batting.playerID = People.playerID)
WHERE ((2021 - People.birthYear) <= 45 ) and (People.deathYear is NULL) and (Appearances.G_all >= 50)""", connection)
```

After some data cleaning steps, I have made first the first calculated column “Age” by the help of birthyear.

```
# Calucalted Column Age

dfActive_Play50g['Age'] = 2021 - dfActive_Play50g['birthYear']
```

```
dfActive_Play50g['Age']
```

Now the second calculated column of Player Name is added to the data frame. Player Name is calculated by concatenating First name and the Last name.

```
# Calculated Column Player Name, got by the concatenation of First Name and Last Name

dfActive_Play50g['playerName'] = dfActive_Play50g['nameFirst'] + str(' ') + dfActive_Play50g['nameLast']
```

```
dfActive_Play50g['playerName']
```

Now drop all the columns related to Birth date and name. After that, remove duplicates rows, if any to avoid redundant rows. Also drop those rows which have missing values.

```
# Dropping all the columns related to Birth date and Name

dfActive_Play50g.drop(['birthYear', 'birthMonth', 'birthDay', 'birthCountry', 'birthState', 'birthCity',
                       'nameFirst', 'nameLast', 'nameGiven'], axis='columns', inplace=True)
```

```
dfActive_Play50g.columns
```

```
Index(['playerID', 'yearID', 'stint', 'teamID', 'lgID', 'G', 'AB', 'R', 'H',
       '2B', '3B', 'HR', 'RBI', 'SB', 'CS', 'BB', 'SO', 'IBB', 'HBP', 'SH',
       'SF', 'GIDP', 'weight', 'bats', 'throws', 'Age', 'playerName'],
      dtype='object')
```

```
# Removing duplicates from the Dataframe as to avoid redundant rows

dfActive_Play50g.drop_duplicates(inplace=True)
print(dfActive_Play50g)
```

```
# Dropping rows with all missing values

dfActive_Play50g.dropna(axis=0, how='all', inplace=True)
```

As required in Question 1, I have to find out the name of active player who had the most runs batted in, from 2015-2018. For this, I have first checked out the unique values in year id and then make a data frame only for those value which lie between 2015 to 2018. Then I have created the max condition for RBI which gives me the desired result that, Nolan Arenado is an active player who had the most runs batted in, from 2015-2018.

```
# Checking the years from YearID

dfActive_Play50g['yearID'].unique()
```

```
array([2004, 2006, 2007, 2008, 2009, 2010, 2012, 2013, 2015, 2011, 2014,
       2016, 2017, 2019, 2001, 2002, 2003, 2005, 2018, 2020, 1998, 1999,
       2000, 1997, 1996], dtype=int64)
```

```
# Which active player had the most runs batted in 2015-2018?
# Making a Dataframe which include years between 2015 and 2018

dfActive_2015_2018 = dfActive_Play50g[np.logical_and(dfActive_Play50g['yearID'] >= 2015, dfActive_Play50g['yearID'] <= 2018)]
```

```
dfActive_2015_2018['yearID'].unique()
```

```
array([2015, 2016, 2017, 2018], dtype=int64)
```

```
# Condition to get the Player Name with the maximum runs (RBI)

PlayerWithMostRuns = dfActive_2015_2018.loc[dfActive_2015_2018['RBI'] == dfActive_2015_2018['RBI'].max()]
```

```
# Fetching the particular player name

print(PlayerWithMostRuns)
```

```
      playerID  yearID  stint teamID lgID    G   AB    R    H  2B  ...  IBB  \
1699  arenano01    2016      1    COL   NL  160  618  116  182  35  ...   10

      HBP  SH  SF  GIDP  weight  bats throws Age     playerName
1699    2   0   8    17     215     R      R  30  Nolan Arenado

[1 rows x 27 columns]
```

```
# Showing the Player name who scored the most runs batted in -- RBI during 2015-2018

print(PlayerWithMostRuns['playerName'])
```

```
1699    Nolan Arenado
Name: playerName, dtype: object
```

Question 2 is that how many double plays did Albert Pujols ground into, in 2016? To answer this, I have made data frame for values exist in 2016. After creating this data frame, Albert Pujols name is equated in the condition to get his data, which gives the answer of GIDP that is 24.

```
# Making the Dataframe which contains only the data records for 2016

dfActive_2016 = dfActive_Play50g[dfActive_Play50g['yearID'] == 2016]
```

```
print(dfActive_2016)
```

```
# Fetching Albert Pujols data

Albert_Pujols_GIDP2016 = dfActive_2016.loc[dfActive_2016['playerName'] == 'Albert Pujols']
```

```
# Showing the GIDP of Albert Pujols

print(Albert_Pujols_GIDP2016['GIDP'])
```

```
56618    24
Name: GIDP, dtype: int64
```

It's time for creating histogram for Triples (3B) per year. First I made a data frame in which only those rows are taken whose 3B is greater than zero. Then I have limited the data set to only two attributes which are 'yearID' and '3B'. Apply 'groupby' operation on this data frame by yearID and taken the aggregated sum of 3B by yearID, whose value is saved to a new data frame named dfTriples_groupedYear. In this data frame, the index of each row is represented by yearID so we can count the number of occurrences of 3B through yearID.

Finally I passed all those index values to plt.hist() function, and setting the bins values for proper result. I have also set color, title, grid, x-label and y label of the histogram to show the graph explicitly.

```
# Creating a histogram for Triples(3B) per year
# Only those rows are extracted whose 3B value is greater than 0

dfTriples = dfActive_Play50g.loc[ dfActive_Play50g['3B'] > 0 ]
```

```
# Taking only necessary columns

dfTriples = dfTriples[['yearID', '3B']]
```

```
print(dfTriples)
```

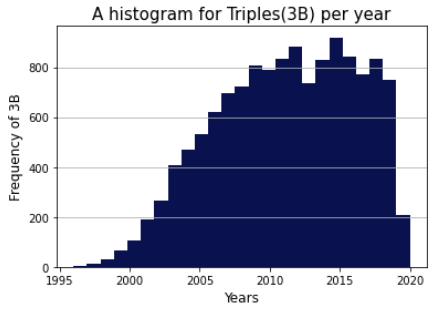
```
# Grouping by year, it will generate sum of 3B for each year

dfTriples_groupedYear = dfTriples.groupby(['yearID']).sum()
```

```
print(dfTriples_groupedYear)
print(dfTriples_groupedYear.shape)
```

```
# Dataframe for which histogram of 3B per year can be calculated

dfHistogram = dfTriples_groupedYear.loc[dfTriples_groupedYear.index.repeat(dfTriples_groupedYear['3B'])]
print(dfHistogram)
```

```
         3B
yearID
1996      4
1996      4
1996      4
1996      4
1997     12
...     ...
2020    206
2020    206
2020    206
2020    206
2020    206

[13325 rows x 1 columns]
```

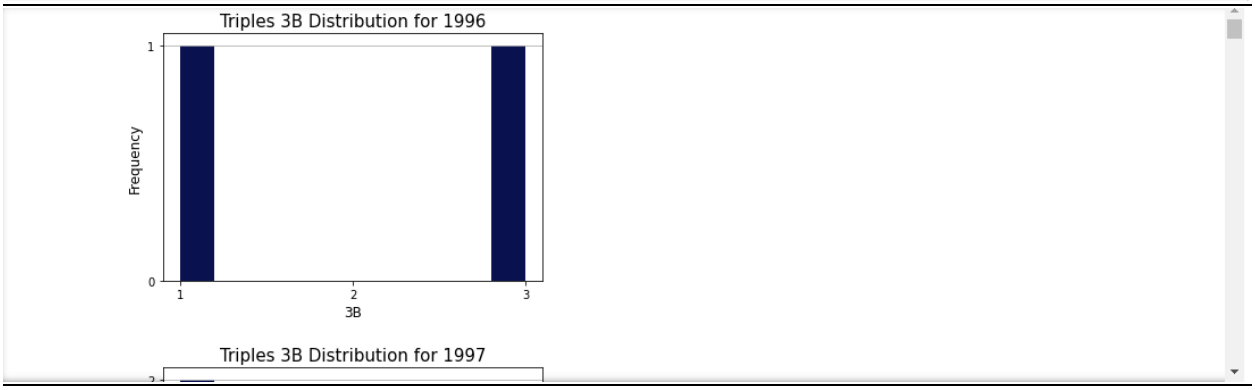
```
# Generating a histogram for Triples(3B) per year

plt.hist(dfHistogram.index.values, bins=25, color='#09124F')
plt.grid(axis='y')
plt.title('A histogram for Triples(3B) per year',fontsize=15)
plt.xlabel('Years',fontsize=12)
plt.ylabel('Frequency of 3B',fontsize=12)
plt.show()
```



For readers, I have created histograms for 3B separately for each year by a “for” loop. Additional features of plotting are also applied to make the histogram more readable and easily understandable.

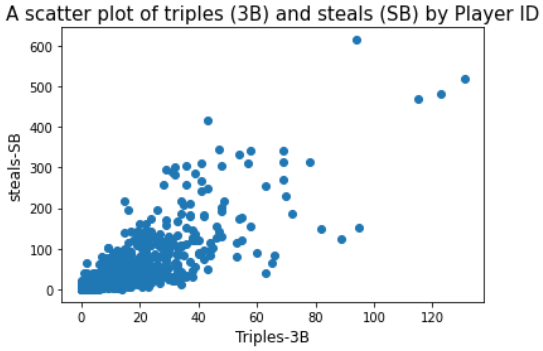
```
# Loop for creating histogram for Triples (3B) per year
# Just to show the distribution of triples(3B) in each year sepearately
# Creating a Dataframes based on YearIDs

dfYears = dfActive_Play50g['yearID'].sort_values().unique()
```

```
for year in dfYears:
    dfHisto_3B = dfActive_Play50g.loc[ dfActive_Play50g['3B'] > 0 ]
    dfHisto_3B.loc[dfHisto_3B['yearID'] == year, '3B'].plot(kind='hist', color='#09124F')
    plt.locator_params(axis="both", integer=True, tight=True)
    plt.grid(axis='y')
    plt.title('Triples 3B Distribution for' + str(' ') + str(year),fontsize=15)
    plt.xlabel('3B',fontsize=12)
    plt.ylabel('Frequency',fontsize=12)
    plt.show()
```



Secondly a scatter plot is generated to show the relation between triples (3B) and steals (SB). To make it more understandable, I have drawn it after applying groupby on the player ID attribute, and then plotting scatter on 3B and SB. There is a linear relationship between 3B and SB for each year by player ID.

```
# Creating a scatter plot relating triples (3B) and steals (SB)

dfScatter = dfActive_Play50g.groupby(['playerID']).sum()
plt.scatter( dfScatter['3B'], dfScatter['SB'])
plt.title('A scatter plot of triples (3B) and steals (SB) by Player ID',fontsize=15)
plt.xlabel('Triples-3B',fontsize=12)
plt.ylabel('steals-SB',fontsize=12)
plt.show()

# As showing in the scatter plot, there is a linear relationship between 3B and SB
```



Some additional questions are also added in this project. To answer that which player, get how many awards? A data frame is created by joining People table and AwardsPlayers table. In this query I have added an

aggregated field of CountOfAwards by using groupby on awardID. The result is shown in a fancy tabular form in which CountOfAwards is sorted in descending order.

```
# Creating a dataframe for awards to get the answer that which player get how many awards?

dfAwards = pd.read_sql_query("""SELECT P.playerID, P.nameGiven, AP.awardID, count(AP.awardID) AS CountOfAwards
FROM AwardsPlayers AS AP JOIN People AS P
ON AP.playerID = P.playerID
GROUP by AP.awardID
ORDER By CountOfAwards DESC""", connection)
```

```
# Shwoing the result in a tabular form about the Number of Awards each player got and player with maximum awards

from tabulate import tabulate
print(tabulate(dfAwards, headers=['Index', 'PlayerID', 'PlayerName', 'AwardID', 'CountOfAwards'], tablefmt="grid"))
```

```
+---------+------------+-----------------+-----------------------------------+-----------------+
|   Index | PlayerID   | PlayerName      | AwardID                           |   CountOfAwards |
+=========+============+=================+===================================+=================+
|       0 | chaseha01  | Harold Homer    | Baseball Magazine All-Star        |            1520 |
+---------+------------+-----------------+-----------------------------------+-----------------+
|       1 | bottoji01  | James Leroy     | TSN All-Star                      |            1391 |
+---------+------------+-----------------+-----------------------------------+-----------------+
|       2 | hodgegi01  | Gilbert Raymond | Gold Glove                        |            1091 |
+---------+------------+-----------------+-----------------------------------+-----------------+
|       3 | coopece01  | Cecil Celester  | Silver Slugger                    |             685 |
+---------+------------+-----------------+-----------------------------------+-----------------+
|       4 | cobbty01   | Tyrus Raymond   | Most Valuable Player              |             196 |
+---------+------------+-----------------+-----------------------------------+-----------------+
|       5 | robinja02  | Jack Roosevelt  | Rookie of the Year                |             142 |
```

Next question is about the shortest player in the baseball history. A data frame is created from People table in which only required columns for height and names are selected. Shortest player is found out by sorting the table height wise and selecting the player on the zero index of the data frame with the help of pd.iloc().

```
# Creating a dataframe to answer about the shortest baseball player

dfPeople = pd.read_sql_query("""SELECT playerID, nameFirst, nameLast, nameGiven, height
FROM People
WHERE height is not NULL
ORDER by height""", connection)
```

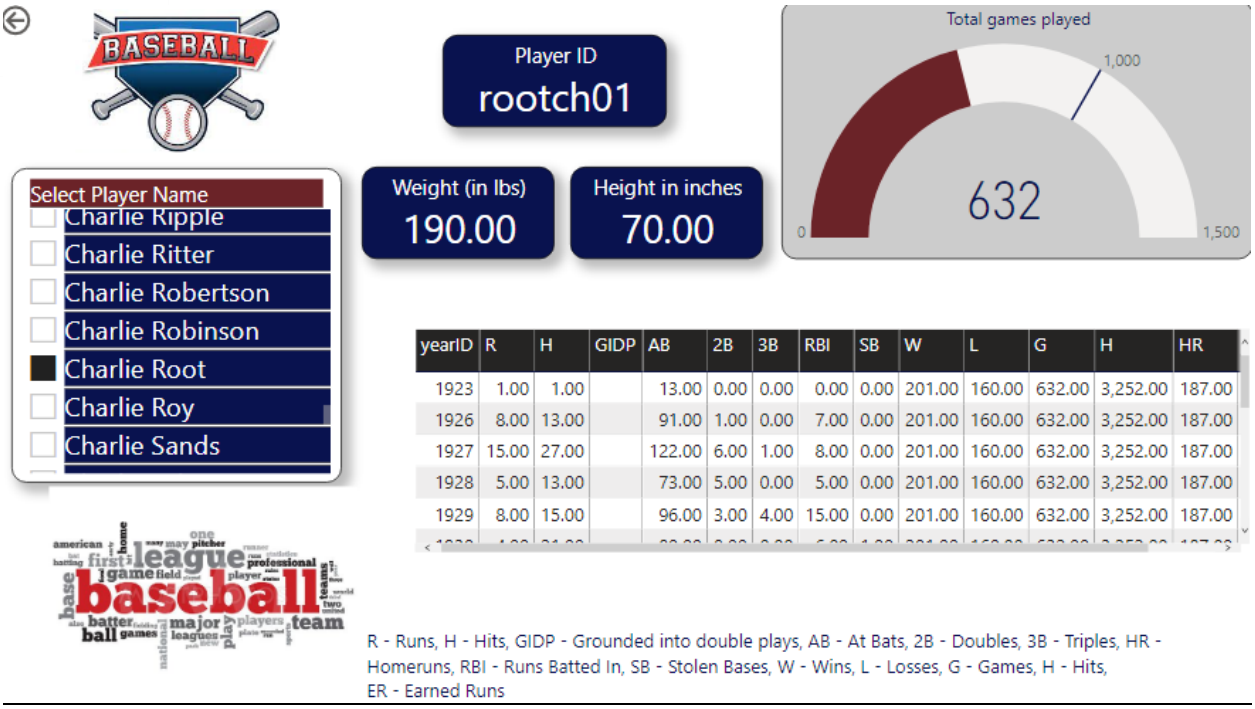
```
ShortestPlayer = dfPeople.iloc[0]
```

```
# Showing the name of the shortest player

print(ShortestPlayer)
```

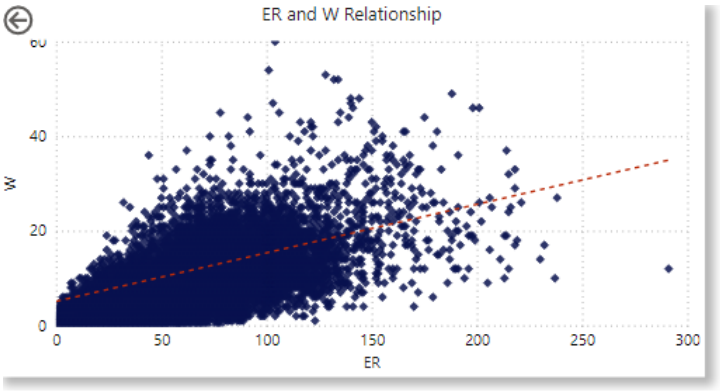
```
playerID       gaedeed01
nameFirst          Eddie
nameLast          Gaedel
nameGiven    Edward Carl
height                43
Name: 0, dtype: object
```

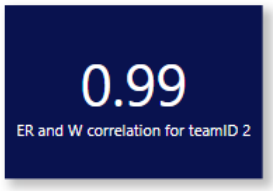
Microsoft Power BI

Power Bi is used to provide interactive visualizations for the given baseball dataset. I have imported only those tables in Power BI on which I have performed the analysis. The remaining tables have been removed as to avoid lots of irrelevant data.



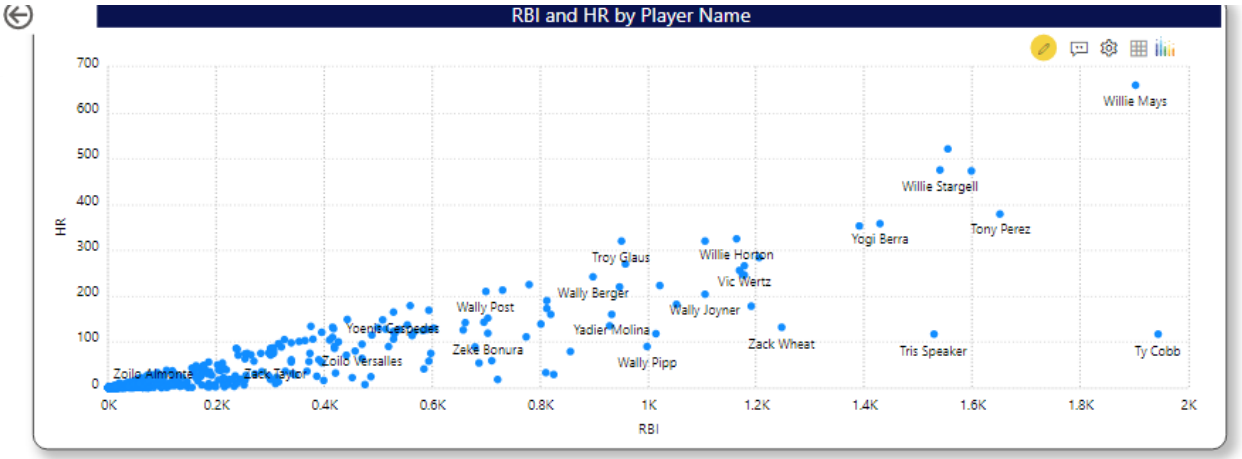
I have created this baseball dashboard as to give impression to the audience that they have full grip on the data by few clicks. Just click on the Player name and get all his baseball statistics like Runs, Hits, Grounded into Double Plays, At Bats, Doubles, Triples, Runs batted in, total games played, weight, height etc in a second!!! I have used Visualizations like Slicer, Card, Table, Text Box, Gauge and some images in this dashboard.



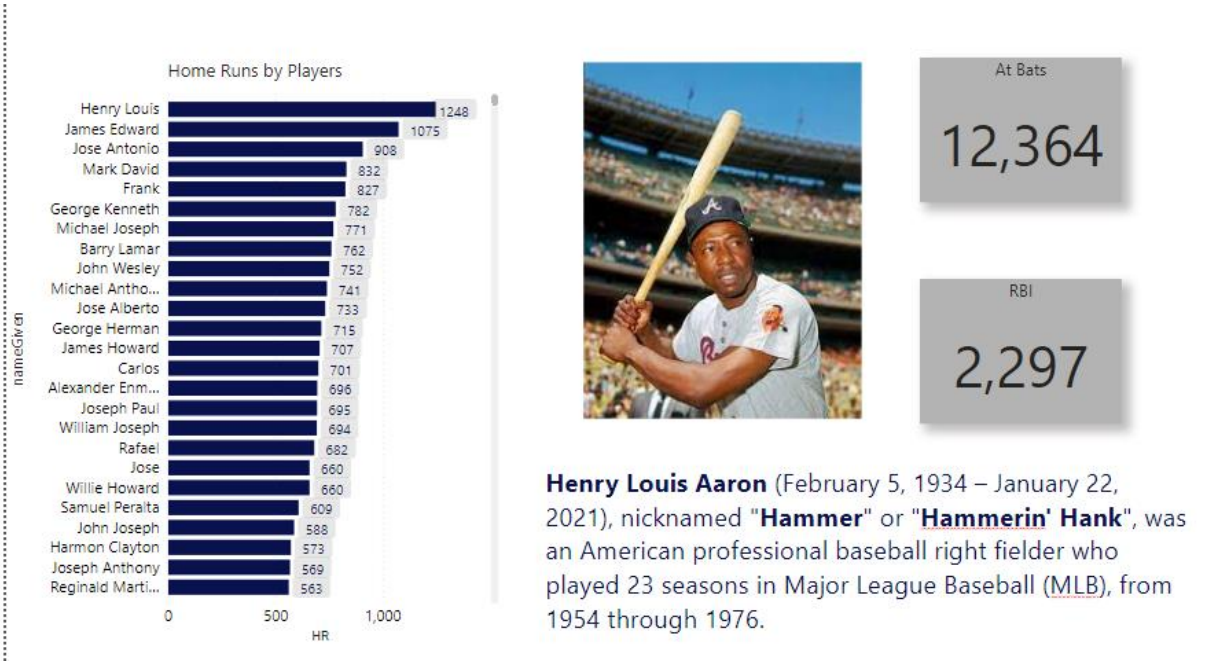


How to Win?

Examining the relationship between how many runs a team score and the number of wins they have. A scatter plot can be created between the two variables and it found out that they have an almost positive linear relationship, which can also be proven by calculating their correlation coefficient i.e 0.99. A trend line is also drawn for reference.
Hence It is intuitive that the more runs a team scores, the more likely it is that they will win games.



Sometimes the audience just want the quick analysis of some baseball variables but in a way that is not very complex to understand. I tried to do such analysis of RBI (Runs batted in) and HR (Home Runs) of a player. I have used Bubble / Scatter Chart – xViz on RBI and HR and a text box for this dashboard. Just hover on the dot of a desired player and the details of his RBI and HR will appear in a dropdown. If you want to further drill down for the same player, Click 'Drill through' and then 'Baseball Dashboard'. It will navigate you to the 'Baseball Dashboard' mentioned earlier. You will get more statistics of that particular player from there, but make sure to select the correct player's name again, if required, because sometimes data for previously selected player get join with the new selected one. Don't forget to clear the filter after getting the baseball stats of a particular player, by clicking on the eraser besides the player's name in the right bottom side pane. It will clear the selection and the dashboard would become ready for the next data analysis.



I really want to know the player's name who has made the highest number of Home Runs (HR) in the Baseball history. On creating the Clustered Bar Chart, I got Henry Louis as the answer. **Henry Louis Aaron** (February 5, 1934 – January 22, 2021), nicknamed "**Hammer**" or "**Hammerin' Hank**", was an American professional baseball right fielder who played 23 seasons in Major League Baseball (MLB), from 1954 through 1976. He made 1248 Home Runs, the highest number of home runs in baseball history. I have used visualizations like Clustered Bar Chart, Card, Text box and an image for this dashboard.

The **Griffeys (Ken Sr. and Ken Jr.)** became one of the first father-and-son tandem to play on the same Major League Baseball team. The exceptionally talented family pair played together for two seasons, in 1990 and 1991, as Senior's career was winding down and Junior's path to the Hall of Fame was beginning.

Table is showing Griffey (Ken Sr. and Jr.) RBI(Runs Batted IN), Strikeouts(SO) and Home Runs(HR)

playerID	nameFirst	nameLast	birthYear	weight	height	RBI	SO	HR
griffke01	Ken	Griffey	1,950.00	190.00	71.00	859.00	898.00	152.00
griffke02	Ken	Griffey	1,969.00	195.00	75.00	1,836.00	1,779.00	630.00

While working on this dataset, I came across with this interesting fact, that the first father and son duo in baseball players was Ken Griffey Sr. and Ken Griffey Jr. Both of them are also the members of Hall of Fame which is rare. On Sept. 14, 1990, Ken Griffey Jr. and Ken Griffey Sr. made history, as they have been known to do, when they became the first father-son duo to hit back-to-back home runs in a game against the California Angels. When Griffey Sr. signed with the Seattle Mariners (Griffey Jr.'s team at the time), no father and son had ever even played together on the same team.

The exceptionally talented family pair played together for two seasons, in 1990 and 1991, as Senior's career was winding down and Junior's path to the Hall of Fame was beginning.

IDEAS FOR FUTURE IMPROVEMENT

- There are number of records having null values in dataset, which eventually affects the analysis.
- Irrelevant data is of no use, so there is no point of using it.
- Practice on more datasets and explore Power BI for hands on experience of newer visualization techniques. Further work on my Python scripting techniques for more critical thinking and logic understanding.