Introduction:-

Investigate a Dataset of Baseball

Dat 101 Statistical Analysis Final Project

Group 7 member :- Hariyalee Patel

Section 1:-

Questions to study :-

Q 1. Study the historical baseball data(File name - 'Baseball_teams.csv')

Q 2. Teams who have strong defenses, strong offenses, or a combination of the two for an extended period of time are more successful in baseball based on correlation analysis.

Q 3. Identifying the dominant playing style for each period of time (offensive, defensive, or balanced).

Q 4. For the years 1990 to 2010, create a linear regression prediction model that can forecast how many games a team will win. And what are the factors which can help a team to win.

Q 5. Evaluate the statistical significance of the final regression model.

Q 6. Using a 1990 to 2010 regression model, predict the Yankees' and Blue Jays' number of games won using the actual values of the independent variables for 2012 and 2015.

Objective:-

Business Objective - Baseball Association wants to study historical baseball data to analyze diffrent games styles from 1871 to 2015 on based of different diffensive and offensive measures to find the answers of above questions. So they can explore what is the best stratagy to improve the chance to win in the games and on what factors they have to focus the most.

Analytical Objective:- Analytical goals include developing precise predictive models to support the decision-making process in baseball strategy. We are going to use technique of Correlation analysis to assess different periods of history of baseball to

explore different stratagies used by different teams. With the help of different data visualization techniques we are going to analyze the different deffensive and offensive variables under consideration. To make our predictive model we will use multiple linear regression modeling technique. In our prediction, we are expectating our results with 5% of error.

Import primary libraries

In [64]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.formula.api as smf
import statsmodels.api as sm2
import warnings
```

In [65]:

```
df = pd.read_csv('baseball_teams.csv')
```

Section 2 :-

Analyze the data

In [66]:

```
df.shape
```

Out[66]:

```
(2805, 43)
```

In [67]:

```
df.describe()
```

Out[67]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Score
count	2805.000000	2805.000000	2805.00000	2406.000000	2805.000000	2805.000000	2805.00000
mean	1955.036720	4.107308	150.34795	78.465919	74.749020	74.749020	681.94581
std	41.519083	2.323414	23.22725	4.698684	17.640402	17.378079	135.73824
min	1871.000000	1.000000	6.00000	44.000000	0.000000	4.000000	24.00000
25%	1919.000000	2.000000	153.00000	77.000000	66.000000	65.000000	613.00000
50%	1963.000000	4.000000	157.00000	81.000000	77.000000	76.000000	690.00000
75%	1992.000000	6.000000	162.00000	81.000000	87.000000	87.000000	763.00000
max	2015.000000	13.000000	165.00000	84.000000	116.000000	134.000000	1220.00000

8 rows × 32 columns

In [68]:
```
df.columns
```

Out[68]:
```
Index(['Year', 'League', 'Team', 'Franchise ', 'Division', 'Final_Standing',
       'Games_Played', 'Unnamed: 7', 'Games_Won', 'Games_Lost', 'Unnamed: 10',
       'Unnamed: 11', 'League_Win', 'World_Series', 'Runs_Scored', 'At_Bats',
       'Hits', 'Doubles', 'Triples', 'Home_Runs', 'Walks', 'Strike_Outs',
       'Stolen_Bases', 'Caught_Stealing', 'Hit_By_Pitch', 'Sacrifice_Fly',
       'Runs_Against', 'Earned_Runs', 'Earned_Run_Average', 'Complete_Games',
       'Shutout', 'Saves', 'Infield_Put_Outs', 'Hits_Allowed',
       'Home_Run_Allowed', 'Walks_Allowed', 'Strikeouts_Allowed', 'Errors',
       'Double_Plays', 'Fielding_Percentage', 'Team_Name', 'Home_Ball_Park',
       'Attendance'],
      dtype='object')
```

In [69]:
```
df.dtypes
```

Out[69]:

```
Year                     int64
League                  object
Team                    object
Franchise               object
Division                object
Final_Standing           int64
Games_Played             int64
Unnamed: 7             float64
Games_Won                int64
Games_Lost               int64
Unnamed: 10             object
Unnamed: 11             object
League_Win              object
World_Series            object
Runs_Scored              int64
At_Bats                  int64
Hits                     int64
Doubles                  int64
Triples                  int64
Home_Runs                int64
Walks                    int64
Strike_Outs            float64
Stolen_Bases           float64
Caught_Stealing        float64
Hit_By_Pitch           float64
Sacrifice_Fly          float64
Runs_Against             int64
Earned_Runs              int64
Earned_Run_Average     float64
Complete_Games           int64
Shutout                  int64
Saves                    int64
Infield_Put_Outs         int64
Hits_Allowed             int64
Home_Run_Allowed         int64
Walks_Allowed            int64
Strikeouts_Allowed       int64
Errors                   int64
Double_Plays           float64
Fielding_Percentage    float64
Team_Name               object
Home_Ball_Park          object
Attendance              object
dtype: object
```

In [70]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2805 entries, 0 to 2804
Data columns (total 43 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Year                 2805 non-null   int64  
 1   League               2755 non-null   object 
 2   Team                 2805 non-null   object 
 3   Franchise            2805 non-null   object 
 4   Division             1288 non-null   object 
 5   Final_Standing       2805 non-null   int64  
 6   Games_Played         2805 non-null   int64  
 7   Unnamed: 7           2406 non-null   float64
 8   Games_Won            2805 non-null   int64  
 9   Games_Lost           2805 non-null   int64  
 10  Unnamed: 10          1260 non-null   object 
 11  Unnamed: 11          624 non-null    object 
 12  League_Win           2777 non-null   object 
 13  World_Series         2448 non-null   object 
 14  Runs_Scored          2805 non-null   int64  
 15  At_Bats              2805 non-null   int64  
 16  Hits                 2805 non-null   int64  
 17  Doubles              2805 non-null   int64  
 18  Triples              2805 non-null   int64  
 19  Home_Runs            2805 non-null   int64  
 20  Walks                2805 non-null   int64  
 21  Strike_Outs          2685 non-null   float64
 22  Stolen_Bases         2661 non-null   float64
 23  Caught_Stealing      1946 non-null   float64
 24  Hit_By_Pitch         480 non-null    float64
 25  Sacrifice_Fly        480 non-null    float64
 26  Runs_Against         2805 non-null   int64  
 27  Earned_Runs          2805 non-null   int64  
 28  Earned_Run_Average   2805 non-null   float64
 29  Complete_Games       2805 non-null   int64  
 30  Shutout              2805 non-null   int64  
 31  Saves                2805 non-null   int64  
 32  Infield_Put_Outs     2805 non-null   int64  
 33  Hits_Allowed         2805 non-null   int64  
 34  Home_Run_Allowed     2805 non-null   int64  
 35  Walks_Allowed        2805 non-null   int64  
 36  Strikeouts_Allowed   2805 non-null   int64  
 37  Errors               2805 non-null   int64  
 38  Double_Plays         2488 non-null   float64
 39  Fielding_Percentage  2805 non-null   float64
 40  Team_Name            2805 non-null   object 
 41  Home_Ball_Park       2771 non-null   object 
 42  Attendance           2527 non-null   object 
dtypes: float64(9), int64(23), object(11)
memory usage: 942.4+ KB
```

```
In [71]: print(df.isnull().sum())
```

```
Year                      0
League                   50
Team                      0
Franchise                 0
Division               1517
Final_Standing            0
Games_Played              0
Unnamed: 7              399
Games_Won                 0
Games_Lost                0
Unnamed: 10            1545
Unnamed: 11            2181
League_Win               28
World_Series            357
Runs_Scored               0
At_Bats                   0
Hits                      0
Doubles                   0
Triples                   0
Home_Runs                 0
Walks                     0
Strike_Outs             120
Stolen_Bases            144
Caught_Stealing         859
Hit_By_Pitch           2325
Sacrifice_Fly          2325
Runs_Against              0
Earned_Runs               0
Earned_Run_Average        0
Complete_Games            0
Shutout                   0
Saves                     0
Infield_Put_Outs          0
Hits_Allowed              0
Home_Run_Allowed          0
Walks_Allowed             0
Strikeouts_Allowed        0
Errors                    0
Double_Plays            317
Fielding_Percentage       0
Team_Name                 0
Home_Ball_Park           34
Attendance              278
dtype: int64
```

In [74]:

```
mean_value = df[['Strike_Outs', 'Stolen_Bases', 'Caught_Stealing', 'Hit_By_Pitch', 'Sa
```

In [75]:

```
df.loc[:, ['Strike_Outs', 'Stolen_Bases', 'Caught_Stealing', 'Hit_By_Pitch', 'Sacrific
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2805 entries, 0 to 2804
Data columns (total 43 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Year                2805 non-null   int64  
 1   League              2755 non-null   object 
 2   Team                2805 non-null   object 
 3   Franchise           2805 non-null   object 
 4   Division            1288 non-null   object 
 5   Final_Standing      2805 non-null   int64  
 6   Games_Played        2805 non-null   int64  
 7   Unnamed: 7          2406 non-null   float64
 8   Games_Won           2805 non-null   int64  
 9   Games_Lost          2805 non-null   int64  
 10  Unnamed: 10         1260 non-null   object 
 11  Unnamed: 11         624 non-null    object 
 12  League_Win          2777 non-null   object 
 13  World_Series        2448 non-null   object 
 14  Runs_Scored         2805 non-null   int64  
 15  At_Bats             2805 non-null   int64  
 16  Hits                2805 non-null   int64  
 17  Doubles             2805 non-null   int64  
 18  Triples             2805 non-null   int64  
 19  Home_Runs           2805 non-null   int64  
 20  Walks               2805 non-null   int64  
 21  Strike_Outs         2805 non-null   float64
 22  Stolen_Bases        2805 non-null   float64
 23  Caught_Stealing     2805 non-null   float64
 24  Hit_By_Pitch        2805 non-null   float64
 25  Sacrifice_Fly       2805 non-null   float64
 26  Runs_Against        2805 non-null   int64  
 27  Earned_Runs         2805 non-null   int64  
 28  Earned_Run_Average  2805 non-null   float64
 29  Complete_Games      2805 non-null   int64  
 30  Shutout             2805 non-null   int64  
 31  Saves               2805 non-null   int64  
 32  Infield_Put_Outs    2805 non-null   int64  
 33  Hits_Allowed        2805 non-null   int64  
 34  Home_Run_Allowed    2805 non-null   int64  
 35  Walks_Allowed       2805 non-null   int64  
 36  Strikeouts_Allowed  2805 non-null   int64  
 37  Errors              2805 non-null   int64  
 38  Double_Plays        2805 non-null   float64
 39  Fielding_Percentage  2805 non-null   float64
 40  Team_Name           2805 non-null   object 
 41  Home_Ball_Park      2771 non-null   object 
 42  Attendance          2527 non-null   object 
dtypes: float64(9), int64(23), object(11)
memory usage: 942.4+ KB
```

Data Exploration :-

Our journey commences with the exploration of raw data. Scrutinizing data types, completeness, and cleansing strategies, we ensure our dataset's suitability for analysis. Understanding the distribution shape of each field is pivotal, guiding our feature selection process for subsequent modeling.

Assumptions regarding the normal distribution, independence, and linear relationships within variables are crucial for building robust models.

Section 3 :-

Correlation Analysis:-

Correlation analysis, a keystone in model development, examines linear relationships between independent variables and the dependent variable, "Games Won." Our focus on offensive and defensive measurements provides insights into baseball styles for different time periods. By selecting the top eight variables, we classify periods into offensive, defensive, or balanced styles of play, summarizing our findings in a comprehensive table.

In [11]:
```
df.corr(method='pearson',numeric_only='False')
```

Out[11]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost
Year	1.000000	-0.293981	0.601304	0.341956	0.418790	0.425112
Final_Standing	-0.293981	1.000000	-0.171295	-0.085398	-0.732510	0.502550
Games_Played	0.601304	-0.171295	1.000000	0.963764	0.676371	0.661355
Unnamed: 7	0.341956	-0.085398	0.963764	1.000000	0.358841	0.295313
Games_Won	0.418790	-0.732510	0.676371	0.358841	1.000000	-0.102111
Games_Lost	0.425112	0.502550	0.661355	0.295313	-0.102111	1.000000
Runs_Scored	0.273367	-0.447966	0.532620	0.353552	0.677243	0.028576
At_Bats	0.605436	-0.199481	0.986522	0.904329	0.689233	0.632149
Hits	0.506059	-0.306493	0.867794	0.547261	0.725160	0.436382
Doubles	0.683101	-0.364275	0.679297	0.379173	0.586949	0.339586
Triples	-0.610569	0.065342	0.002071	-0.169251	0.086865	-0.122844
Home_Runs	0.837680	-0.358621	0.522866	0.380709	0.484075	0.248887
Walks	0.572747	-0.283531	0.774200	0.381918	0.655929	0.385736
Strike_Outs	0.892200	-0.225856	0.650705	0.485755	0.419930	0.495577
Stolen_Bases	-0.392816	-0.044271	-0.097893	0.017897	0.043120	-0.186490
Caught_Stealing	-0.230845	0.024827	0.172786	0.040722	0.078767	0.092447
Hit_By_Pitch	-0.204524	-0.121507	-0.020682	-0.009773	0.161396	-0.163465
Sacrifice_Fly	-0.262328	-0.253320	0.086856	0.060635	0.298843	-0.297095
Runs_Against	0.275035	0.305040	0.513703	0.314490	0.006238	0.686534
Earned_Runs	0.642611	0.100081	0.703112	0.374492	0.235721	0.726280
Earned_Run_Average	0.371608	0.318717	0.174785	0.086441	-0.220270	0.480528
Complete_Games	-0.876366	0.196974	-0.246025	-0.284432	-0.134360	-0.243658
Shutout	0.095318	-0.380173	0.351139	0.121658	0.545568	-0.090116
Saves	0.897888	-0.403732	0.525381	0.408410	0.505623	0.232377
Infield_Put_Outs	0.617324	-0.202563	0.996585	0.951415	0.697276	0.637079
Hits_Allowed	0.507877	0.046543	0.858471	0.519463	0.413214	0.741285
Home_Run_Allowed	0.884038	-0.173527	0.549978	0.408671	0.320302	0.453769
Walks_Allowed	0.572311	0.013765	0.776006	0.370675	0.380921	0.667285
Strikeouts_Allowed	0.879181	-0.346624	0.641185	0.490309	0.518630	0.364530
Errors	-0.833145	0.308794	-0.494665	-0.208608	-0.430352	-0.267573
Double_Plays	0.517770	-0.030141	0.612708	0.406277	0.302562	0.354396
Fielding_Percentage	0.777934	-0.298955	0.859709	0.319166	0.644387	0.521540

32 rows × 32 columns

Period 1 (>1920)

In [12]:
```
df1=df[(df['Year']<1920)]
```

In [13]:
```
df1.describe()
```

Out[13]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	703.000000	703.000000	703.000000	304.000000	703.000000	703.000000	703.000000
mean	1897.237553	4.913229	127.769559	75.167763	62.758179	62.758179	612.795164
std	13.698282	2.722102	35.340280	5.009708	23.223088	22.548362	190.125631
min	1871.000000	1.000000	6.000000	52.000000	0.000000	4.000000	24.000000
25%	1886.000000	3.000000	116.000000	73.000000	49.000000	48.000000	500.500000
50%	1898.000000	5.000000	139.000000	77.000000	66.000000	64.000000	614.000000
75%	1909.500000	7.000000	154.000000	78.000000	80.000000	78.000000	739.000000
max	1919.000000	13.000000	162.000000	84.000000	116.000000	134.000000	1220.000000

8 rows × 32 columns

In [14]:
```
df1.corr(method='pearson',numeric_only='False')
```

Out[14]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost
Year	1.000000	-0.082193	0.802923	0.039266	0.602368	0.620393
Final_Standing	-0.082193	1.000000	-0.108053	-0.091165	-0.635882	0.489376
Games_Played	0.802923	-0.108053	1.000000	0.889232	0.765999	0.745224
Unnamed: 7	0.039266	-0.091165	0.889232	1.000000	0.303142	0.137983
Games_Won	0.602368	-0.635882	0.765999	0.303142	1.000000	0.144745
Games_Lost	0.620393	0.489376	0.745224	0.137983	0.144745	1.000000
Runs_Scored	0.170307	-0.335152	0.564020	0.123811	0.681313	0.160974
At_Bats	0.735271	-0.131851	0.987764	0.800200	0.775265	0.716733
Hits	0.655431	-0.225380	0.905877	0.307485	0.808361	0.558298
Doubles	0.562873	-0.311365	0.760180	0.120461	0.749308	0.397839
Triples	0.424524	-0.178688	0.666072	0.165061	0.644800	0.362214
Home_Runs	0.207973	-0.151843	0.431225	0.037063	0.456323	0.192673
Walks	0.684370	-0.118003	0.844984	0.348215	0.712184	0.563769
Strike_Outs	0.665538	-0.014480	0.803176	0.654670	0.549502	0.702733
Stolen_Bases	-0.133942	-0.213909	0.352508	0.325600	0.430729	0.023014
Caught_Stealing	0.942084	-0.252966	0.943815	0.327101	0.883660	0.854059
Hit_By_Pitch	NaN	NaN	NaN	NaN	NaN	NaN
Sacrifice_Fly	NaN	NaN	NaN	NaN	NaN	NaN
Runs_Against	0.173375	0.391338	0.535903	-0.004590	0.132917	0.685972
Earned_Runs	0.465079	0.315659	0.714682	-0.008117	0.316495	0.772942
Earned_Run_Average	-0.207545	0.597110	-0.122262	-0.297767	-0.408200	0.235102
Complete_Games	0.232261	-0.071234	0.701214	0.010033	0.553754	0.498868
Shutout	0.627117	-0.462850	0.553754	0.311831	0.673471	0.156623
Saves	0.676590	-0.277237	0.481929	0.248635	0.526227	0.202158
Infield_Put_Outs	0.821519	-0.136902	0.997586	0.882112	0.779830	0.727625
Hits_Allowed	0.664930	0.121449	0.900834	0.244269	0.555890	0.811717
Home_Run_Allowed	0.230242	0.159822	0.464565	0.015318	0.243336	0.464053
Walks_Allowed	0.679998	0.095441	0.834464	0.301987	0.532710	0.732441
Strikeouts_Allowed	0.676153	-0.242680	0.798978	0.577207	0.695112	0.506540
Errors	-0.549536	0.186997	-0.087397	0.087324	-0.199159	0.065033
Double_Plays	0.785223	0.087461	0.764035	0.211687	0.497300	0.544475
Fielding_Percentage	0.878291	-0.234411	0.880804	0.140322	0.752685	0.580469

32 rows × 32 columns

In [15]:
```
off_def=['Runs_Scored','At_Bats','Hits','Doubles','Triples','Home_Runs','Walks','Strik
'Runs_Against','Earned_Runs','Earned_Run_Average','Complete_Games','Shutout','Saves','
```

In [16]:
```
df1.corr(method='pearson',numeric_only=True).loc['Games_Won'][off_def].sort_values(asc
```

Out[16]:
```
Caught_Stealing         0.883660
Hits                    0.808361
Infield_Put_Outs        0.779830
At_Bats                 0.775265
Fielding_Percentage     0.752685
Doubles                 0.749308
Walks                   0.712184
Strikeouts_Allowed      0.695112
Name: Games_Won, dtype: float64
```

Top 8 Features have 3 Defensive measures (Infield Put outs, Fielding Percentage , Strikeouts Allowed) and 5 Offensive measures are (Caught Stealing , Hits , At Bats , Doubles , Walks) . So the game stratgy for time period 1 (Before 1920) was balanced as it have both type of variables <=6

Period 2 (1920 - 1959)

In [17]:
```
df2=df[(df['Year']>=1920)&(df['Year']<1959)]
```

In [18]:
```
df2.describe()
```

Out[18]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	624.000000	624.000000	624.000000	624.000000	624.000000	624.000000	624.000000
mean	1939.000000	4.479167	154.314103	77.157051	76.657051	76.657051	714.424679
std	11.263658	2.285318	1.505707	1.484326	14.447703	14.322171	108.047285
min	1920.000000	1.000000	147.000000	70.000000	38.000000	43.000000	394.000000
25%	1929.000000	2.000000	154.000000	77.000000	66.000000	66.000000	643.000000
50%	1939.000000	4.000000	154.000000	77.000000	78.000000	76.000000	707.500000
75%	1949.000000	6.000000	155.000000	78.000000	87.000000	87.000000	778.250000
max	1958.000000	8.000000	158.000000	82.000000	111.000000	115.000000	1067.000000

8 rows × 32 columns

In [19]:
```
df2.corr(method='pearson',numeric_only='False')
```

Out[19]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost
Year	1.000000	-0.000561	0.187773	0.095239	0.009203	0.009283
Final_Standing	-0.000561	1.000000	-0.094655	-0.194462	-0.944458	0.943079
Games_Played	0.187773	-0.094655	1.000000	0.576867	0.122353	-0.060200
Unnamed: 7	0.095239	-0.194462	0.576867	1.000000	0.212541	-0.177692
Games_Won	0.009203	-0.944458	0.122353	0.212541	1.000000	-0.996321
Games_Lost	0.009283	0.943079	-0.060200	-0.177692	-0.996321	1.000000
Runs_Scored	-0.255401	-0.607823	0.056384	0.107275	0.633883	-0.634312
At_Bats	-0.201823	-0.227050	0.392102	0.260312	0.237114	-0.218212
Hits	-0.579199	-0.404228	0.034058	0.086832	0.413347	-0.414882
Doubles	-0.459580	-0.264623	0.003377	0.070700	0.264535	-0.269011
Triples	-0.634954	-0.243095	-0.015786	0.084814	0.259337	-0.261680
Home_Runs	0.540612	-0.329044	0.120390	0.069412	0.347905	-0.334864
Walks	0.403945	-0.341929	0.173674	0.117755	0.362145	-0.354284
Strike_Outs	0.745435	0.071954	0.181739	0.035459	-0.060956	0.079134
Stolen_Bases	-0.534625	-0.147107	-0.021219	0.026234	0.162620	-0.164449
Caught_Stealing	-0.713872	-0.006681	-0.086535	-0.045484	0.019406	-0.025564
Hit_By_Pitch	NaN	NaN	NaN	NaN	NaN	NaN
Sacrifice_Fly	NaN	NaN	NaN	NaN	NaN	NaN
Runs_Against	-0.259871	0.590099	-0.041705	-0.137689	-0.632395	0.633330
Earned_Runs	-0.101179	0.571413	-0.021436	-0.127255	-0.609230	0.612229
Earned_Run_Average	-0.111466	0.581665	-0.085319	-0.165424	-0.622357	0.621170
Complete_Games	-0.588220	-0.333619	-0.037072	0.064088	0.365759	-0.375751
Shutout	0.197733	-0.504508	0.116415	0.199273	0.530373	-0.526597
Saves	0.526857	-0.384499	0.116224	0.067296	0.412007	-0.402038
Infield_Put_Outs	0.148140	-0.359083	0.683573	0.448297	0.397806	-0.355533
Hits_Allowed	-0.548918	0.467292	-0.030358	-0.101921	-0.498431	0.497424
Home_Run_Allowed	0.675809	0.120423	0.117846	0.015153	-0.123852	0.142282
Walks_Allowed	0.426025	0.242202	0.119133	0.008963	-0.255384	0.263750
Strikeouts_Allowed	0.708818	-0.313724	0.230651	0.175938	0.346984	-0.327303
Errors	-0.693122	0.320916	-0.107817	-0.130983	-0.348327	0.340441
Double_Plays	0.361087	-0.058163	0.116125	0.126884	0.061420	-0.046757
Fielding_Percentage	0.563599	-0.306013	0.148567	0.153690	0.324973	-0.315295

32 rows × 32 columns

In [20]:

```
df2.corr(method='pearson',numeric_only=True).loc['Games_Won'][off_def].sort_values(asc
```

Out[20]:

```
Runs_Scored            0.633883
Runs_Against          -0.632395
Earned_Run_Average    -0.622357
Earned_Runs           -0.609230
Shutout                0.530373
Hits_Allowed          -0.498431
Hits                   0.413347
Saves                  0.412007
Name: Games_Won, dtype: float64
```

Top 8 Correlated Variables from the Offensive group are - Runs Scored and Hits and Deffensive measures are - Runs Againgst ,Earned Runs, Shutouts , saves ,Earned Runs Average , Hits allowed. So the game style was deffensive in time period 2 (1920 to 1959) as deffensive measures are >=6.

Period 3(1920-1959)

In [21]:

```
df3=df[(df['Year']>=1960)&(df['Year']<1989)]
```

In [22]:

```
df3.describe()
```

Out[22]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	678.000000	678.000000	678.000000	678.000000	678.000000	678.000000	678.000000
mean	1974.952802	4.060472	159.286136	79.643068	79.541298	79.541298	665.259587
std	8.167603	2.239210	10.603537	5.369868	12.819756	12.747887	94.704128
min	1960.000000	1.000000	103.000000	47.000000	37.000000	42.000000	329.000000
25%	1968.000000	2.000000	161.000000	81.000000	71.000000	71.000000	612.000000
50%	1975.000000	4.000000	162.000000	81.000000	81.000000	79.000000	673.000000
75%	1982.000000	6.000000	162.000000	81.000000	89.000000	88.000000	729.000000
max	1988.000000	10.000000	165.000000	84.000000	109.000000	120.000000	896.000000

8 rows × 32 columns

In [23]:

```
df3.corr(method='pearson',numeric_only='False')
```

Out[23]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost
Year	1.000000	-0.225985	-0.103337	-0.102026	-0.038832	-0.039051
Final_Standing	-0.225985	1.000000	0.022724	0.026489	-0.752349	0.771939
Games_Played	-0.103337	0.022724	1.000000	0.986877	0.419862	0.411640
Unnamed: 7	-0.102026	0.026489	0.986877	1.000000	0.408798	0.411491
Games_Won	-0.038832	-0.752349	0.419862	0.408798	1.000000	-0.653611
Games_Lost	-0.039051	0.771939	0.411640	0.411491	-0.653611	1.000000
Runs_Scored	0.123500	-0.438513	0.517861	0.508217	0.677972	-0.248408
At_Bats	-0.065018	-0.030466	0.974049	0.960355	0.464663	0.344977
Hits	0.119865	-0.276893	0.728832	0.715028	0.582773	0.022938
Doubles	0.398739	-0.278873	0.430477	0.420387	0.422006	-0.060621
Triples	-0.127428	-0.091210	0.223524	0.223195	0.230426	-0.047491
Home_Runs	-0.026274	-0.216235	0.327481	0.321226	0.438812	-0.167957
Walks	0.002980	-0.215874	0.466453	0.459443	0.404958	-0.015958
Strike_Outs	-0.191727	0.193289	0.558960	0.555709	0.133940	0.330392
Stolen_Bases	0.493340	-0.269634	0.113247	0.110507	0.194718	-0.097932
Caught_Stealing	0.438374	-0.117312	0.134281	0.129969	0.036846	0.079660
Hit_By_Pitch	NaN	NaN	NaN	NaN	NaN	NaN
Sacrifice_Fly	NaN	NaN	NaN	NaN	NaN	NaN
Runs_Against	0.124761	0.403543	0.516278	0.517365	-0.232727	0.665209
Earned_Runs	0.167076	0.376344	0.494815	0.496429	-0.212740	0.627682
Earned_Run_Average	0.236416	0.433860	0.059468	0.069643	-0.469733	0.522994
Complete_Games	-0.462488	-0.170827	0.206329	0.203890	0.331273	-0.163729
Shutout	-0.236727	-0.327229	0.134420	0.130979	0.450653	-0.342018
Saves	0.247808	-0.401180	0.217303	0.210268	0.507155	-0.325266
Infield_Put_Outs	-0.111400	-0.028216	0.988740	0.975905	0.469873	0.351925
Hits_Allowed	0.121694	0.223014	0.731456	0.727754	0.049474	0.561029
Home_Run_Allowed	-0.032160	0.220013	0.397062	0.398502	0.000030	0.330626
Walks_Allowed	0.002900	0.267633	0.452790	0.453638	-0.156217	0.536507
Strikeouts_Allowed	-0.183314	-0.040948	0.536654	0.530567	0.354762	0.089233
Errors	-0.237284	0.303890	0.439297	0.434045	-0.134957	0.498784
Double_Plays	-0.039167	0.015028	0.456284	0.450226	0.170943	0.209105
Fielding_Percentage	0.208630	-0.265697	-0.058552	-0.060396	0.249344	-0.295882

32 rows × 32 columns

In [24]:
```
df3.corr(method='pearson',numeric_only=True).loc['Games_Won'][off_def].sort_values(asc
```

Out[24]:
```
Runs_Scored            0.677972
Hits                   0.582773
Saves                  0.507155
Infield_Put_Outs       0.469873
Earned_Run_Average    -0.469733
At_Bats                0.464663
Shutout                0.450653
Home_Runs              0.438812
Name: Games_Won, dtype: float64
```

Top 8 Variables correlated to Games Won are - Offensive - Runs Scored , At Bats , Hits , Home Runs.¶ Defensive - Earned Run Average , Shutouts , Saves , Infield Put Outs. So the game style of period 3 was balanced as none of varables are more the 6 for offensive or deffensive measures.

Period 4 (1990-2010)

In [25]:
```
df4=df[(df['Year']>=1990)&(df['Year']<2010)]
```

In [26]:
```
df4.describe()
```

Out[26]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	578.000000	578.000000	578.000000	578.000000	578.000000	578.000000	578.000000
mean	1999.754325	3.124567	158.771626	79.361592	79.365052	79.365052	749.415225
std	5.710387	1.567118	10.763557	5.492618	12.348912	12.324608	94.213964
min	1990.000000	1.000000	112.000000	44.000000	43.000000	40.000000	466.000000
25%	1995.000000	2.000000	162.000000	81.000000	71.000000	71.000000	688.250000
50%	2000.000000	3.000000	162.000000	81.000000	79.000000	79.000000	747.000000
75%	2005.000000	4.000000	162.000000	81.000000	88.000000	88.000000	809.750000
max	2009.000000	7.000000	163.000000	84.000000	116.000000	119.000000	1009.000000

8 rows × 32 columns

In [27]:
```
df4.corr(method='pearson',numeric_only='False')
```

Out[27]:

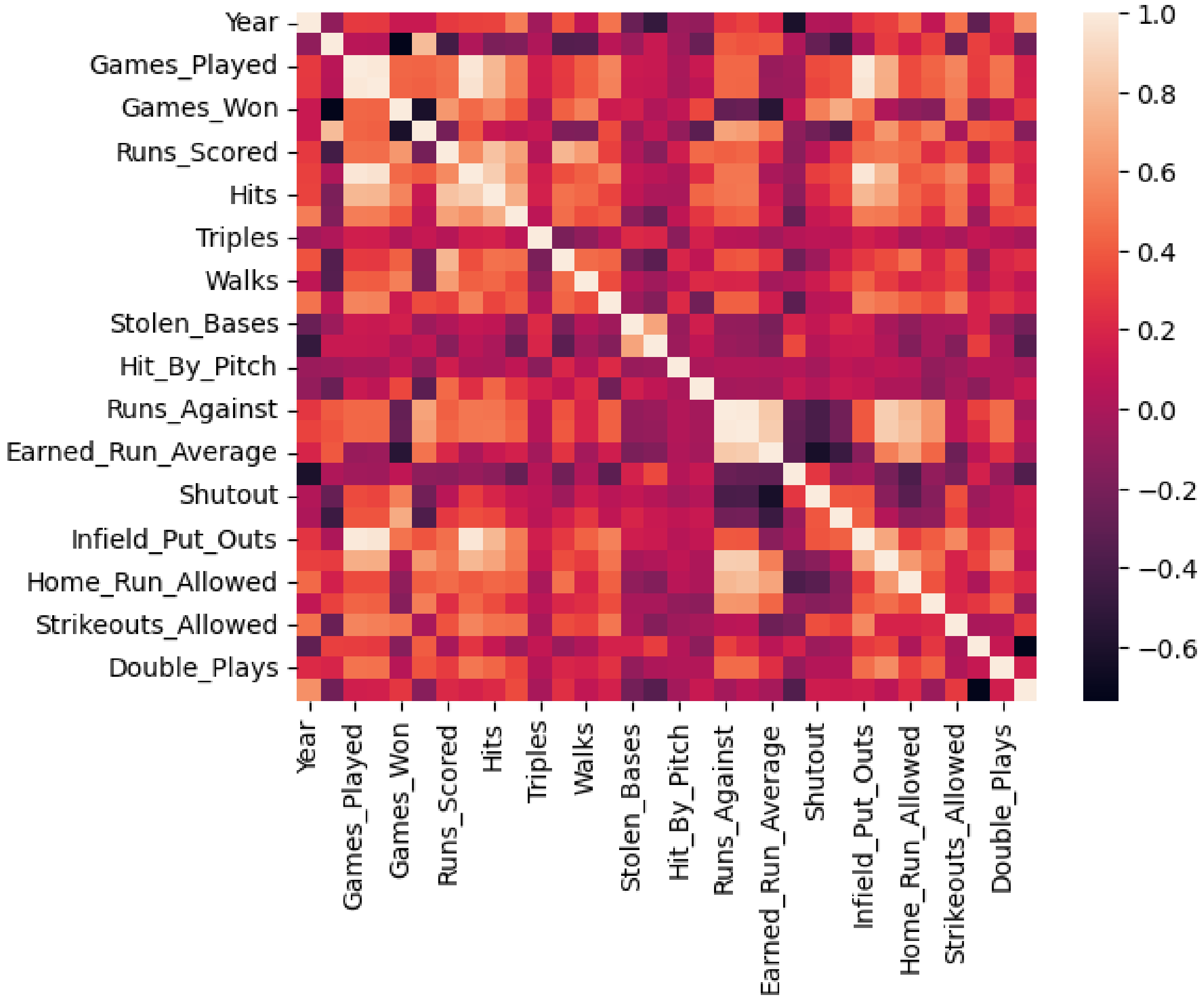
	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost
Year	1.000000	-0.107352	0.283988	0.277515	0.124086	0.124330
Final_Standing	-0.107352	1.000000	0.054707	0.045699	-0.734652	0.784864
Games_Played	0.283988	0.054707	1.000000	0.980078	0.439008	0.433981
Unnamed: 7	0.277515	0.045699	0.980078	1.000000	0.434521	0.421092
Games_Won	0.124086	-0.734652	0.439008	0.434521	1.000000	-0.618811
Games_Lost	0.124330	0.784864	0.433981	0.421092	-0.618811	1.000000
Runs_Scored	0.286646	-0.427075	0.473063	0.467861	0.627709	-0.216235
At_Bats	0.313859	0.019478	0.978408	0.958905	0.456900	0.397169
Hits	0.319450	-0.189215	0.769363	0.756511	0.554099	0.116928
Doubles	0.527540	-0.165760	0.528244	0.519716	0.393862	0.066509
Triples	-0.043209	0.014883	0.158189	0.152510	0.030072	0.108665
Home_Runs	0.373986	-0.353521	0.283593	0.276738	0.423865	-0.177911
Walks	0.081930	-0.343058	0.410851	0.407903	0.537761	-0.180357
Strike_Outs	0.488025	0.063241	0.555172	0.543812	0.137599	0.347899
Stolen_Bases	-0.261464	-0.062566	0.129099	0.118182	0.165352	-0.053715
Caught_Stealing	-0.493359	0.118041	0.118987	0.110791	0.019056	0.085206
Hit_By_Pitch	-0.064956	-0.046295	-0.012764	-0.020509	0.091551	-0.093613
Sacrifice_Fly	-0.096238	-0.261052	0.124981	0.077768	0.331581	-0.327902
Runs_Against	0.274440	0.402194	0.449139	0.445515	-0.282918	0.675790
Earned_Runs	0.318483	0.374340	0.444810	0.442029	-0.261973	0.651068
Earned_Run_Average	0.191278	0.396296	-0.073155	-0.063100	-0.550992	0.488004
Complete_Games	-0.607178	0.012466	-0.040015	-0.045660	0.082703	-0.117801
Shutout	0.030275	-0.276035	0.345723	0.325513	0.533061	-0.231001
Saves	0.006018	-0.458565	0.380938	0.378986	0.709256	-0.378716
Infield_Put_Outs	0.263700	0.010630	0.991931	0.972855	0.487229	0.378650
Hits_Allowed	0.302516	0.290372	0.726720	0.719960	0.012052	0.622535
Home_Run_Allowed	0.457301	0.159982	0.344681	0.346374	-0.111259	0.412008
Walks_Allowed	0.087109	0.309262	0.434202	0.418510	-0.146569	0.526540
Strikeouts_Allowed	0.484901	-0.266267	0.552917	0.538039	0.495095	-0.012486
Errors	-0.304833	0.303055	0.301507	0.284148	-0.144582	0.407927
Double_Plays	0.216581	0.187352	0.489225	0.479078	0.051243	0.376302
Fielding_Percentage	0.602149	-0.234164	0.153049	0.163810	0.272719	-0.138527

32 rows × 32 columns

Correlation cofficients for period 4

In [28]:
```
sns.heatmap(df4.corr(method='pearson',numeric_only='True'))
```

Out[28]:
```
<Axes: >
```



In [29]:
```
df4.corr(method='pearson',numeric_only=True).loc['Games_Won'][off_def].sort_values(asc
```

Out[29]:
```
Saves                  0.709256
Runs_Scored            0.627709
Hits                   0.554099
Earned_Run_Average    -0.550992
Walks                  0.537761
Shutout                0.533061
Strikeouts_Allowed     0.495095
Infield_Put_Outs       0.487229
Name: Games_Won, dtype: float64
```

Top 8 variables for time period 4 - Offensive - Runs Scored , Hits , Walks, Defensive - Earned Run Average , Shutout , Saves, Infield Put Out , Strike out allowed.¶ So the period 4 is also balanced in game style .

In [30]:
```
df4.describe()
```

Out[30]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	578.000000	578.000000	578.000000	578.000000	578.000000	578.000000	578.000000
mean	1999.754325	3.124567	158.771626	79.361592	79.365052	79.365052	749.415225
std	5.710387	1.567118	10.763557	5.492618	12.348912	12.324608	94.213964
min	1990.000000	1.000000	112.000000	44.000000	43.000000	40.000000	466.000000
25%	1995.000000	2.000000	162.000000	81.000000	71.000000	71.000000	688.250000
50%	2000.000000	3.000000	162.000000	81.000000	79.000000	79.000000	747.000000
75%	2005.000000	4.000000	162.000000	81.000000	88.000000	88.000000	809.750000
max	2009.000000	7.000000	163.000000	84.000000	116.000000	119.000000	1009.000000

8 rows × 32 columns

Descriptive Statistics of Period 4 :-

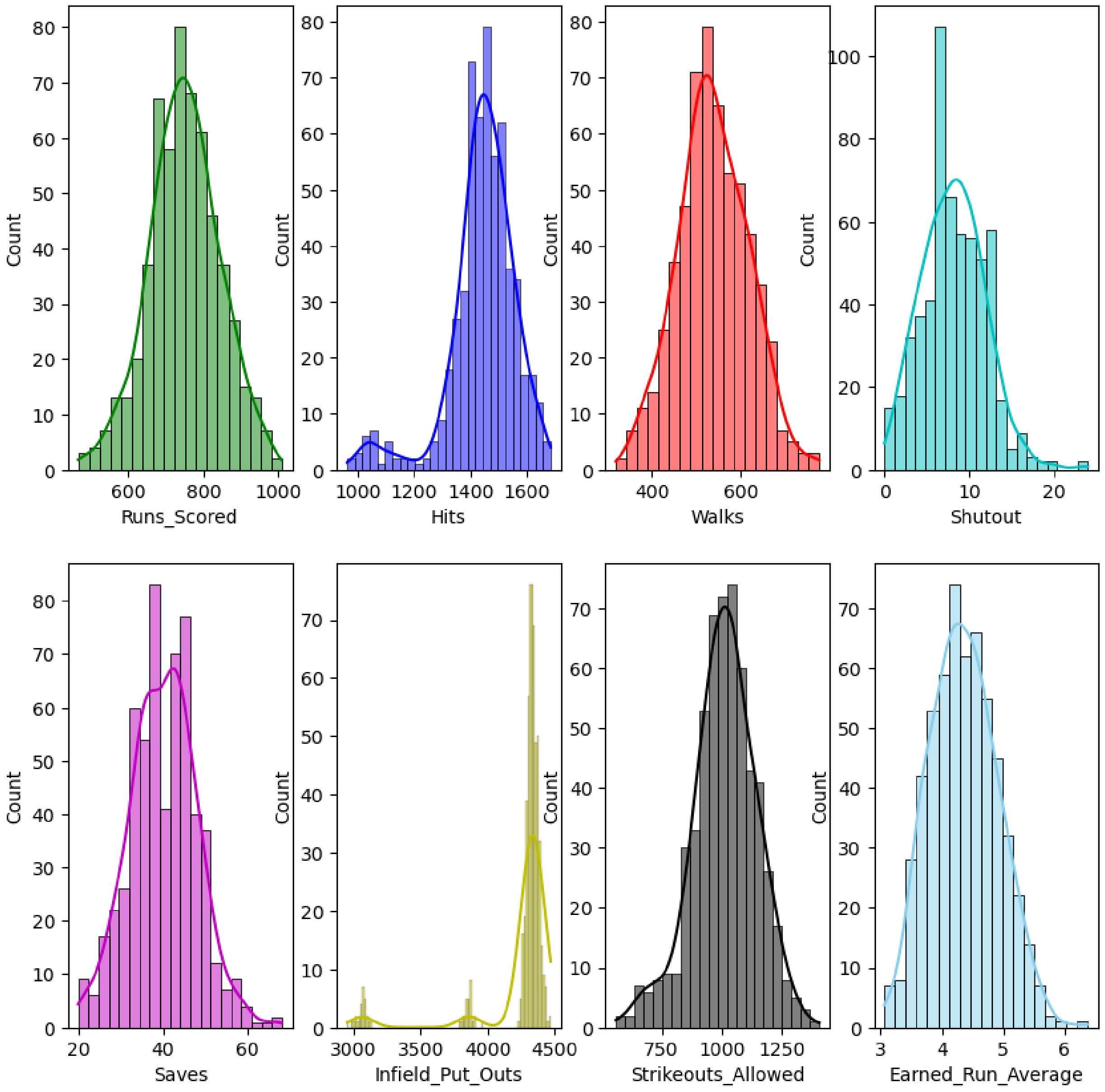
Histogram of each 8 variables of Period 4:-

In [31]:

```
fig, axs=plt.subplots(2,4,figsize=(10,10))

sns.histplot(data=df4,x='Runs_Scored',kde=True,ax=axs[0,0],color='g')

sns.histplot(data=df4,x='Hits',kde=True,ax=axs[0,1],color='b')

sns.histplot(data=df4,x='Walks',kde=True,ax=axs[0,2],color='r')

sns.histplot(data=df4,x='Shutout',kde=True,ax=axs[0,3],color='c')

sns.histplot(data=df4,x='Saves',kde=True,ax=axs[1,0],color='m')

sns.histplot(data=df4,x='Infield_Put_Outs',kde=True,ax=axs[1,1],color='y')

sns.histplot(data=df4,x='Strikeouts_Allowed',kde=True,ax=axs[1,2],color='k')

sns.histplot(data=df4,x='Earned_Run_Average',kde=True,ax=axs[1,3],color='skyblue')
```

Out[31]:

```
<Axes: xlabel='Earned_Run_Average', ylabel='Count'>
```

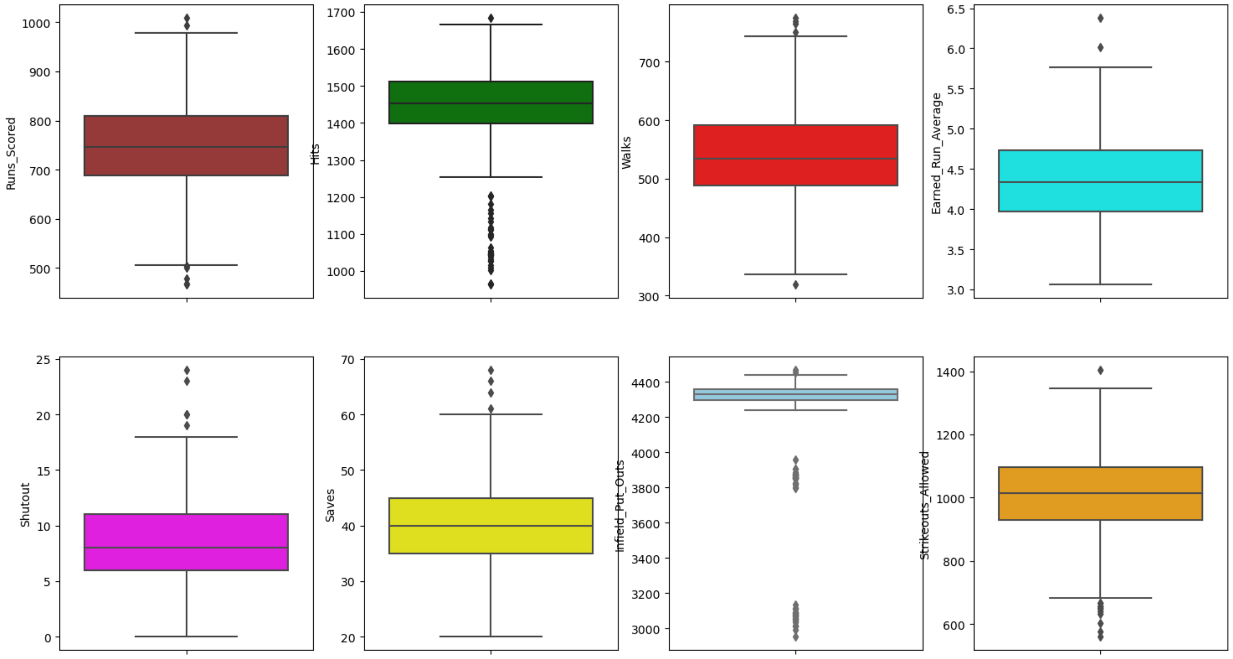


BoxPlot for each top 8 variables of Period 4:-

multiple box subplot

```
In [32]: fig, axes = plt.subplots(2, 4, figsize=(18, 10))
         sns.boxplot(ax=axes[0, 0], data=df4, y='Runs_Scored', color='brown')
         sns.boxplot(ax=axes[0, 1], data=df4,  y='Hits',color='green')
         sns.boxplot(ax=axes[0, 2], data=df4,  y='Walks',color='red')
         sns.boxplot(ax=axes[0, 3], data=df4, y='Earned_Run_Average',color='cyan')
         sns.boxplot(ax=axes[1, 0], data=df4, y='Shutout',color='magenta')
         sns.boxplot(ax=axes[1, 1], data=df4,  y='Saves',color='yellow')
         sns.boxplot(ax=axes[1, 2], data=df4,  y='Infield_Put_Outs',color='skyblue')
         sns.boxplot(ax=axes[1, 3], data=df4,  y='Strikeouts_Allowed',color='orange')
```

```
Out[32]: <Axes: ylabel='Strikeouts_Allowed'>
```

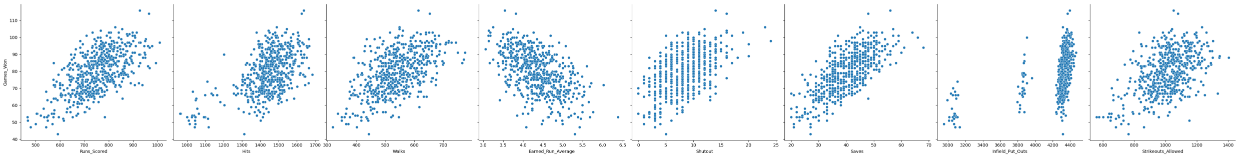


Paired scattered plots

```
In [33]: warnings.filterwarnings("ignore", category=UserWarning)
```

```
In [34]: sns.pairplot(df4,x_vars=['Runs_Scored','Hits','Walks','Earned_Run_Average','Shutout','
```

```
Out[34]: <seaborn.axisgrid.PairGrid at 0x1c7e1b07410>
```

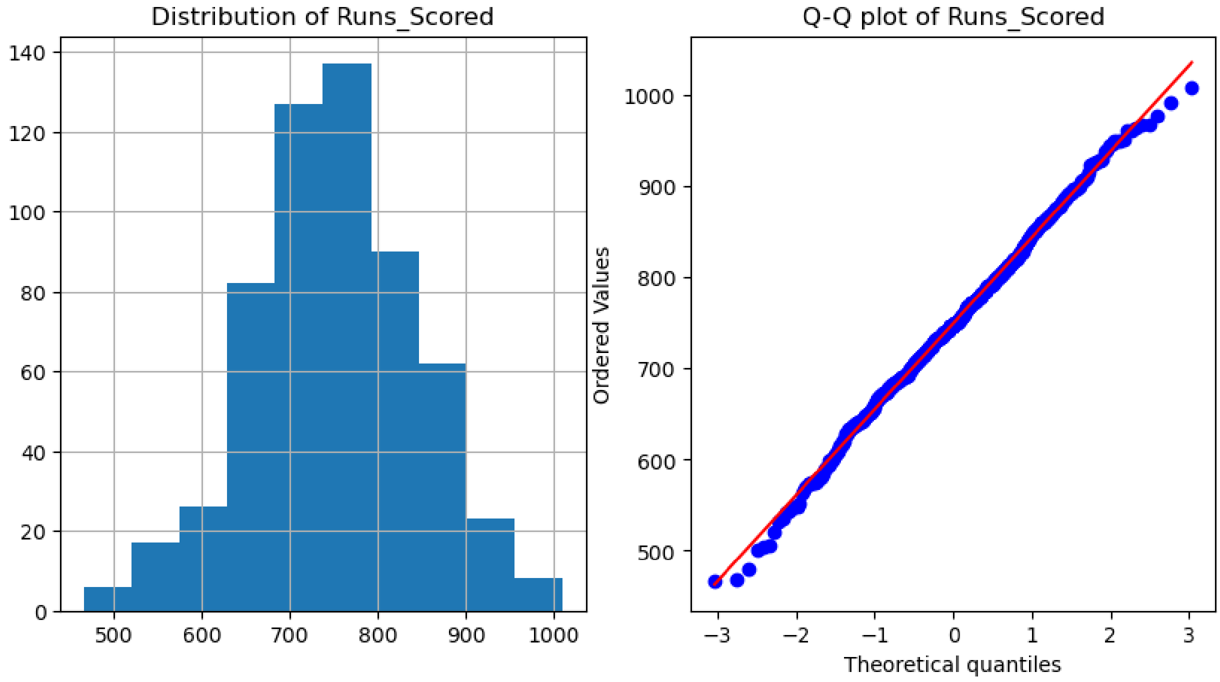


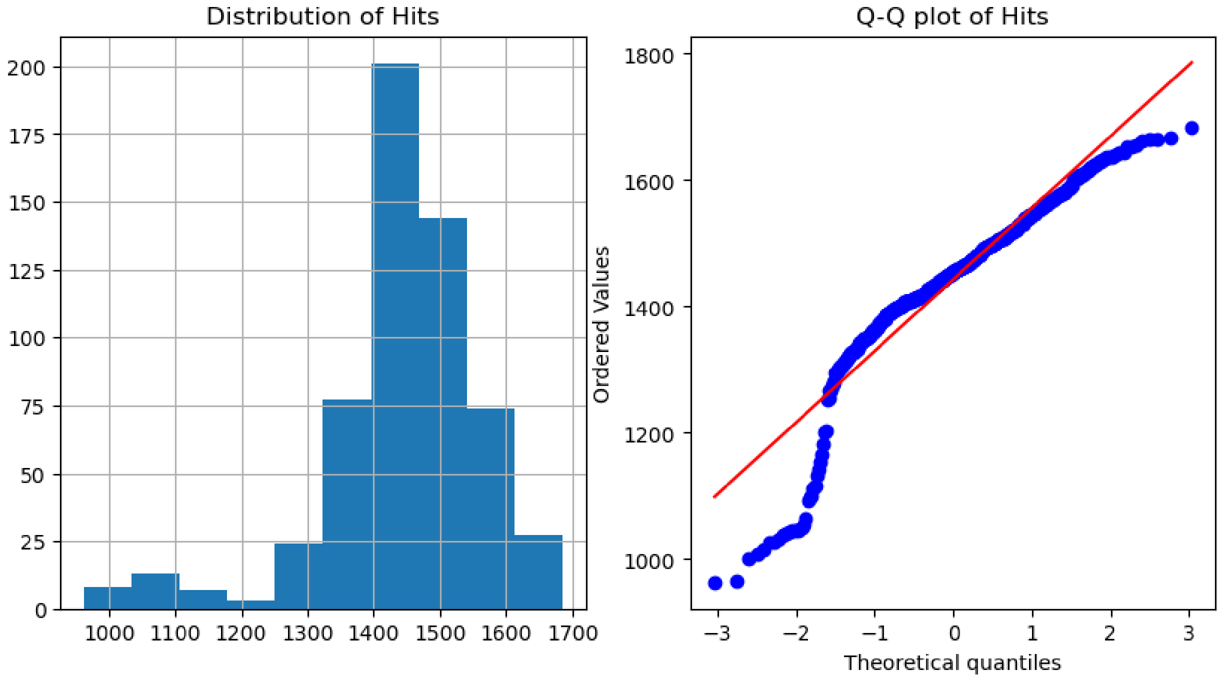
```
In [35]: warnings.resetwarnings()
```

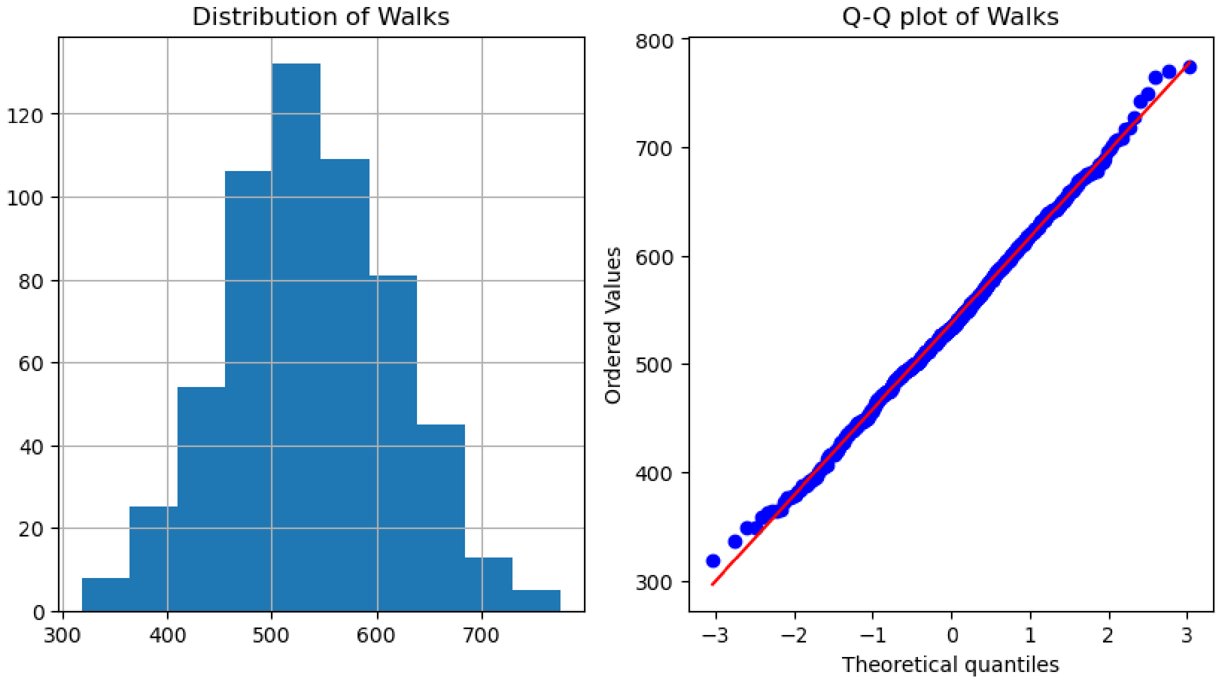
QQ Plot for Period 4:-

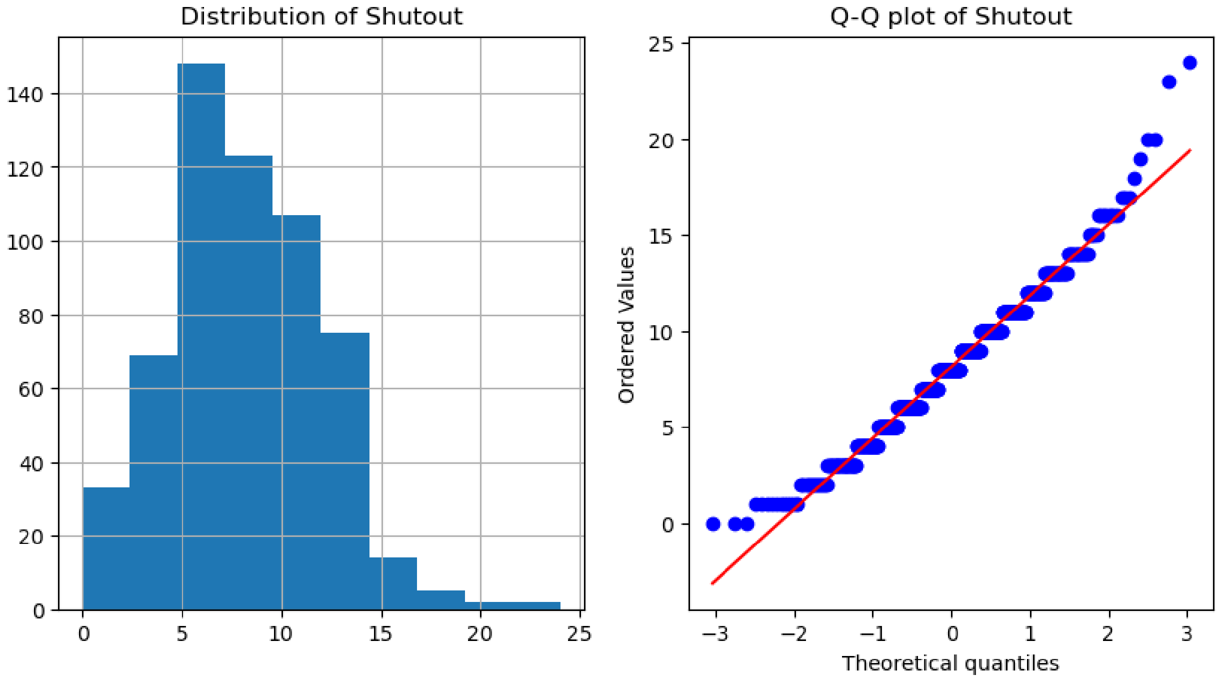
```
In [36]: import pylab
         import scipy.stats as stats
```

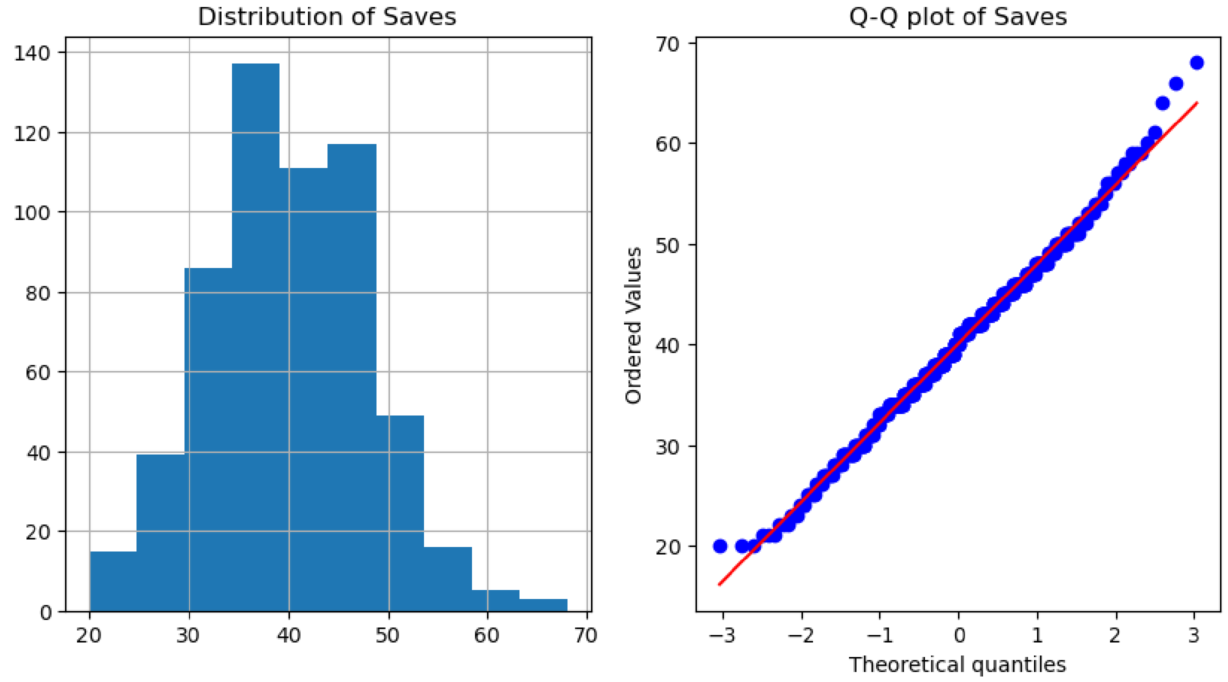
```
In [37]: for var in ['Runs_Scored', 'Hits', 'Walks', 'Shutout', 'Saves', 'Infield_Put_Outs','St
             plt.figure(figsize=(10,5))

             plt.subplot(1, 2, 1)
             df4[var].hist()
             plt.title('Distribution of '+ var)

             plt.subplot(1, 2, 2)
             stats.probplot(df4[var], dist="norm", plot=pylab)
             plt.title('Q-Q plot of '+ var)

             plt.show()
```

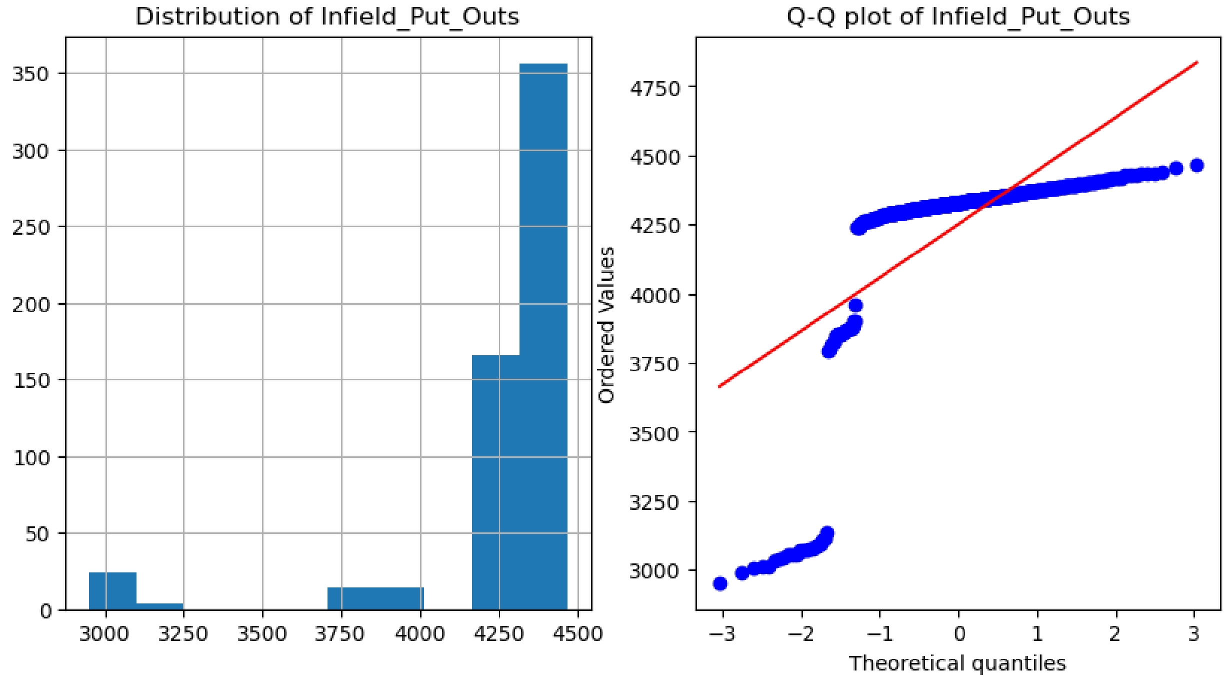

Distribution of Runs_Scored
0
20
40
60
80
100
120
140
500
600
700
800
900
1000
Q-Q plot of Runs_Scored
Ordered Values
500
600
700
800
900
1000
Theoretical quantiles
−3
−2
−1
0
1
2
3

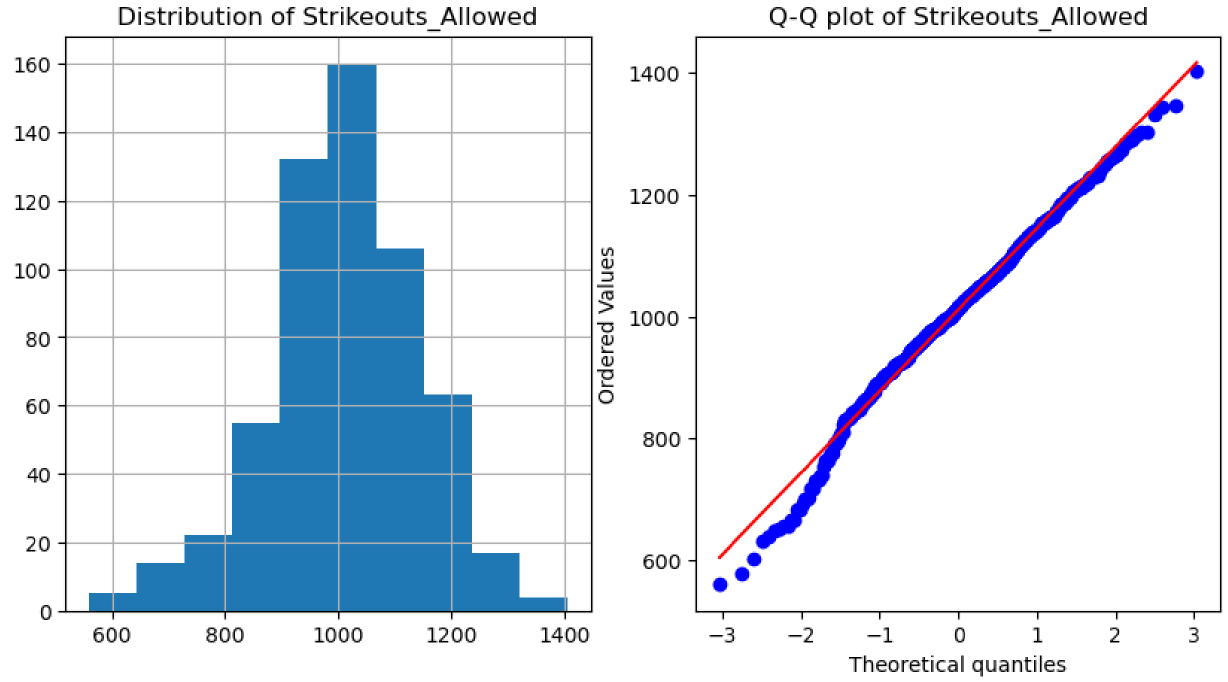

Distribution of Hits
0
25
50
75
100
125
150
175
200
1000
1100
1200
1300
1400
1500
1600
1700
Q-Q plot of Hits
Ordered Values
1000
1200
1400
1600
1800
Theoretical quantiles
−3
−2
−1
0
1
2
3

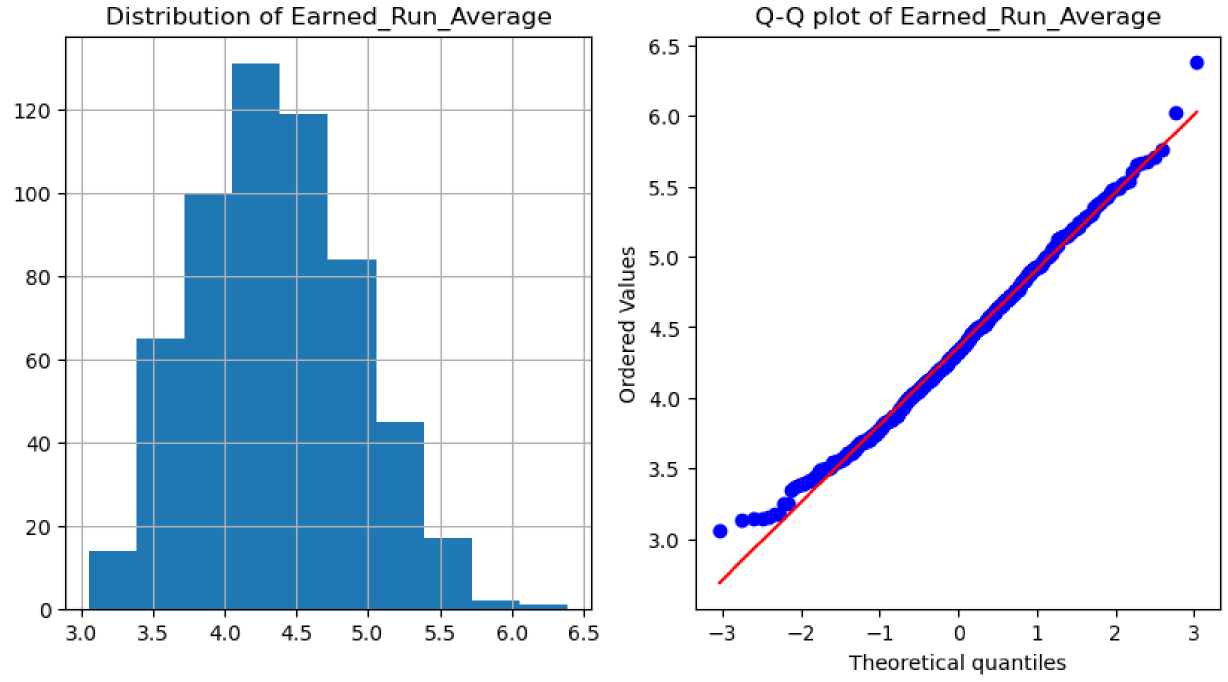

Distribution of Walks
0
20
40
60
80
100
120
300
400
500
600
700
Q-Q plot of Walks
Ordered Values
300
400
500
600
700
800
Theoretical quantiles
−3
−2
−1
0
1
2
3


Distribution of Shutout
0
20
40
60
80
100
120
140
0
5
10
15
20
25
Q-Q plot of Shutout
Ordered Values
0
5
10
15
20
25
Theoretical quantiles
−3
−2
−1
0
1
2
3


Distribution of Saves
140
120
100
80
60
40
20
0
20
30
40
50
60
70
Q-Q plot of Saves
Ordered Values
70
60
50
40
30
20
−3
−2
−1
0
1
2
3
Theoretical quantiles


Distribution of Infield_Put_Outs
350
300
250
200
150
100
50
0
3000
3250
3500
3750
4000
4250
4500
Q-Q plot of Infield_Put_Outs
Ordered Values
4750
4500
4250
4000
3750
3500
3250
3000
−3
−2
−1
0
1
2
3
Theoretical quantiles





Comment on Visual Shape of each Distribution Variable:-

1. Runs_Scored:- In my guess, It is Symmetric (Bell Shaped)

2. Hits :- Skewed Left (negatively skewed)

3. Walks :- Symmetric (Bell Shaped)

4. Earned_Run_Average :- Symmetric (Bell Shaped)

5. Shutout :- Symmetric (Bell Shaped) or Skewed Right

6. Saves :- Symmetric (Bell Shaped)

7. Infield_put_outs:- Skewed Left (negatively skewed)

8. Strikeouts_Allowed :- Symmetric (Bell Shaped)

Section 4:-

Linear Regression :-

Entering the realm of model development, we iteratively construct linear regression models. Balancing significance and integrity through R2, R2 Adjusted, and F-Statistics, we meticulously ensure that selected independent variables exhibit statistical significance based on t-statistics and p-values. Feature engineering decisions are documented to achieve the desired model performance.

In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(df4,df4['Games_Won'],train_size=0.7,rar
```

In [39]:
```
x_train.describe()
```

Out[39]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000
mean	1999.539604	3.175743	158.898515	79.418317	79.163366	79.690594	751.759901
std	5.602721	1.574078	10.393838	5.328025	12.240073	12.393900	95.483252
min	1990.000000	1.000000	112.000000	44.000000	43.000000	40.000000	466.000000
25%	1995.000000	2.000000	162.000000	81.000000	70.000000	71.000000	687.500000
50%	2000.000000	3.000000	162.000000	81.000000	79.000000	79.000000	749.500000
75%	2004.000000	4.000000	162.000000	81.000000	88.000000	88.000000	810.250000
max	2009.000000	7.000000	163.000000	84.000000	116.000000	119.000000	1009.000000

8 rows × 32 columns

In [40]:
```
x_test.describe()
```

Out[40]:

	Year	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games_Lost	Runs_Scored
count	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000	174.000000
mean	2000.252874	3.005747	158.477011	79.229885	79.833333	78.609195	743.971264
std	5.939421	1.548809	11.602623	5.870867	12.621242	12.163963	91.236175
min	1990.000000	1.000000	112.000000	53.000000	47.000000	46.000000	479.000000
25%	1995.000000	2.000000	162.000000	81.000000	71.000000	70.000000	690.000000
50%	2001.000000	3.000000	162.000000	81.000000	79.000000	78.500000	743.000000
75%	2005.000000	4.000000	162.000000	81.000000	88.750000	88.000000	807.250000
max	2009.000000	7.000000	163.000000	84.000000	114.000000	106.000000	965.000000

8 rows × 32 columns

Final Model:-

In [41]:

```
lm1=smf.ols(formula='Games_Won ~ Runs_Scored+Earned_Run_Average+Shutout+Saves', data=c
lm1.summary()
```

Out[41]:

OLS Regression Results

Dep. Variable:	Games_Won	**R-squared:**	0.923
Model:	OLS	**Adj. R-squared:**	0.923
Method:	Least Squares	**F-statistic:**	1722.
Date:	Tue, 19 Dec 2023	**Prob (F-statistic):**	1.24e-317
Time:	14:42:59	**Log-Likelihood:**	-1530.8
No. Observations:	578	**AIC:**	3072.
Df Residuals:	573	**BIC:**	3093.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	47.6309	2.142	22.234	0.000	43.423	51.838
Runs_Scored	0.0867	0.002	49.455	0.000	0.083	0.090
Earned_Run_Average	-11.7491	0.388	-30.249	0.000	-12.512	-10.986
Shutout	0.2443	0.050	4.838	0.000	0.145	0.344
Saves	0.3989	0.023	17.536	0.000	0.354	0.444

Omnibus:	0.497	**Durbin-Watson:**	2.073
Prob(Omnibus):	0.780	**Jarque-Bera (JB):**	0.365
Skew:	0.048	**Prob(JB):**	0.833
Kurtosis:	3.077	**Cond. No.**	1.15e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.15e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Runs_Scored, Earned_Run_Average, Shutout, Saves are our main feature variables for our multiple linear regression model.

Section 5:-

Model Validation :-

Validation becomes a critical checkpoint, evaluating the statistical significance of our model. Error metrics like Root Mean Squared Error (RMSE) based on residuals from test data runs are employed. The segmentation of the original

dataset into training and test data ensures the model's robustness against unseen data. Our report provides detailed insights into error metrics and their significance through trial runs.

Model Testing :-

In [76]:

```
lm1_predict=lm1.predict(x_test)
predict_values=pd.concat([x_test['Games_Won'],lm1_predict],axis=1)
predict_values.columns=['actual_Games_Won','predicted_Games_Won']
predict_values['residual']=predict_values['actual_Games_Won']-predict_values['predicte
predict_values.head()
```

Out[76]:

	actual_Games_Won	predicted_Games_Won	residual
2239	98	98.077729	-0.077729
2466	86	86.556862	-0.556862
2397	84	85.419498	-1.419498
2457	92	91.598573	0.401427
2527	88	88.178492	-0.178492

In [77]:

```
mae=metrics.mean_absolute_error(predict_values['actual_Games_Won'], predict_values['pr
mse=metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_values['pre
rmse=np.sqrt(metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_va
print('Mean Absolute Error', mae)
print('Mean Suare Error ', mse)
print('Root Mean Squared Error', rmse)
```

```
Mean Absolute Error 2.624246108829727
Mean Suare Error  10.819095853248406
Root Mean Squared Error 3.289239403456125
```

In [78]:

```
accuracy = 100 - np.mean(rmse)
print('Accuracy is :' , round(accuracy,2), '%')
```

```
Accuracy is : 96.71 %
```

p<0.05, R-squared: 0.923, adjusted R-squared: 0.923, F-statistic: 1722. , Prob (F-statistic): 1.24e-317

These values for our model showing a good indication that our model is a good predictive.

RMSE value 3.289239403456125 shows that our errors are under 5%.

Accuracy percentage is 96.71%

Section 6:-

Predictions :

Transitioning to the predictions section, our model steps into the realm of practical application. Predicted values, akin to deploying the model in production, are generated to

address the original business questions. Model error metrics, derived from historical data, supplement our evaluation, contributing to the final assessment of meeting both business and analytical objectives.

In [44]:

```
df5=df[(df['Year']==2012)&(df['Team_Name'] == 'New York Yankees')]
```

In [45]:

```
df5.head()
```

Out[45]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Wo
2702	2012	AL	NYA	NYY	E	1	162	81.0	9

1 rows × 43 columns

In [46]:

```
lm1_predict=lm1.predict(df5)
predict_values=pd.concat([df5['Games_Won'],lm1_predict],axis=1)
predict_values.columns=['actual_Games_Won','predicted_Games_Won']
predict_values['residual']=predict_values['actual_Games_Won']-predict_values['predicte
predict_values.head()
```

Out[46]:

	actual_Games_Won	predicted_Games_Won	residual
2702	95	94.770908	0.229092

In [47]:

```
mae=metrics.mean_absolute_error(predict_values['actual_Games_Won'], predict_values['pr
mse=metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_values['pre
rmse=np.sqrt(metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_va
print('Mean Absolute Error', mae)
print('Mean Suare Error ', mse)
print('Root Mean Squared Error', rmse)
```

```
Mean Absolute Error 0.22909190363127152
Mean Suare Error  0.052483100309399795
Root Mean Squared Error 0.22909190363127152
```

In [48]:

```
df6=df[(df['Year']==2012)&(df['Team_Name'] == 'Toronto Blue Jays')]
```

In [49]:

```
df6.head()
```

Out[49]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Wo
2713	2012	AL	TOR	TOR	E	4	162	81.0	7

1 rows × 43 columns

In [50]:

```
lm1_predict=lm1.predict(df6)
predict_values=pd.concat([df6['Games_Won'],lm1_predict],axis=1)
predict_values.columns=['actual_Games_Won','predicted_Games_Won']
predict_values['residual']=predict_values['actual_Games_Won']-predict_values['predicte
predict_values.head()
```

Out[50]:

	actual_Games_Won	predicted_Games_Won	residual
2713	73	69.453986	3.546014

In [51]:

```
mae=metrics.mean_absolute_error(predict_values['actual_Games_Won'], predict_values['pr
mse=metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_values['pre
rmse=np.sqrt(metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_va
print('Mean Absolute Error', mae)
print('Mean Suare Error ', mse)
print('Root Mean Squared Error', rmse)
```

```
Mean Absolute Error 3.546013580817828
Mean Suare Error  12.574212315344473
Root Mean Squared Error 3.546013580817828
```

In [52]:

```
df7=df[(df['Year']==2015)&(df['Team_Name'] == 'New York Yankees')]
```

In [53]:

```
df7.head()
```

Out[53]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Wo
2781	2015	AL	NYA	NYY	E	2	162	81.0	8

1 rows × 43 columns

In [54]:

```
lm1_predict=lm1.predict(df7)
predict_values=pd.concat([df7['Games_Won'],lm1_predict],axis=1)
predict_values.columns=['actual_Games_Won','predicted_Games_Won']
predict_values['residual']=predict_values['actual_Games_Won']-predict_values['predicte
predict_values.head()
```

Out[54]:

	actual_Games_Won	predicted_Games_Won	residual
2781	87	86.651881	0.348119

In [55]:

```
mae=metrics.mean_absolute_error(predict_values['actual_Games_Won'], predict_values['pr
mse=metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_values['pre
rmse=np.sqrt(metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_va
print('Mean Absolute Error', mae)
print('Mean Suare Error ', mse)
print('Root Mean Squared Error', rmse)
```

```
Mean Absolute Error 0.34811948991549
Mean Suare Error  0.12118717925902096
Root Mean Squared Error 0.34811948991549
```

In [56]:

```
df8=df[(df['Year']==2015)&(df['Team_Name'] == 'Toronto Blue Jays')]
```

In [57]:

```
df8.head()
```

Out[57]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Wo
2780	2015	AL	TOR	TOR	E	1	162	81.0	9

1 rows × 43 columns

In [58]:

```
lm1_predict=lm1.predict(df8)
predict_values=pd.concat([df8['Games_Won'],lm1_predict],axis=1)
predict_values.columns=['actual_Games_Won','predicted_Games_Won']
predict_values['residual']=predict_values['actual_Games_Won']-predict_values['predicte
predict_values.head()
```

Out[58]:

	actual_Games_Won	predicted_Games_Won	residual
2780	93	96.247373	-3.247373

In [59]:

```
mae=metrics.mean_absolute_error(predict_values['actual_Games_Won'], predict_values['pr
mse=metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_values['pre
rmse=np.sqrt(metrics.mean_squared_error(predict_values['actual_Games_Won'], predict_va
print('Mean Absolute Error', mae)
print('Mean Suare Error ', mse)
print('Root Mean Squared Error', rmse)
```

```
Mean Absolute Error 3.247372624478146
Mean Suare Error  10.545428962210082
Root Mean Squared Error 3.247372624478146
```

Find Accuracy with percentage :-

In [60]:

```
accuracy = 100 - np.mean(rmse)
print('Accuracy is :' , round(accuracy,2), '%')
```

```
Accuracy is : 96.75 %
```

Conclusion :-

In summary, our recommendation to the Baseball Association's aims to enhance game strategy by analyzing historical data. Through precise predictive models, correlation analysis, and data visualization, the study explores defensive and offensive measures from 1871 to 2015. Please check inline comments for the answer of each part. The focus is on identifying optimal strategies for victory, with a commitment to predictive accuracy within a 5% margin. This project provides a strategic roadmap for teams with accuracy 95% and more emphasizing key factors for success in the dynamic landscape of baseball.

Appendices of Individual Work:-

Appendix 1(Hariyalee) :- file:///C:/Users/Hariyalee/Downloads/Project%20with%20Coding%20-%20Hariyalee%20Patel.html

Each appendix provides a detailed account of the individual's work, including calculations, analyses, and reflections on the statistical concepts applied throughout the project.

In [ ]: