Project by Faraz

Table of Content

1. Introduction and objectives

While the true origins of baseball are not confirmed, it's believed to be played first in the mid-1700s in England. Statistics have always been a part of baseball, with advancements in computer and data science we now have advanced sabermetrics and visualizations to compare different players, teams, and variables to determine the outcome of the game.

1.1 Business objectives

1. Determine if teams are more successful with high offenses, tight defenses or a balance of the two.

2. Determine if this conclusion is valid across multiple time periods.

```
Period 1 - before 1920
Period 2 - 1920 to 1960
Period 3 - 1960 to 1990
Period 4 - 1990 to 2010
```

1.2 Analytical objectives

We are working with a data set that has data for 2805 matches from 1871 to 2015. In sprint 1 of the project, we will be performing an exploratory data analysis to understand the statistical properties and distribution shapes of 8 key variables and do a correlation analysis to study if win or loss has a correlation to run scores, run against, and run difference, from 1960 to 2010.
In sprint 2 of the project we will be doing correlation analysis to games won, creating a test and train data set and build linear regression model for all of our 4 time periods, and test our model by predicting games won by NYY and TORR from 2012 - 2015

CSDA1100 Data Analytics for Business - Sprint 1

2. Exploatory Data Analysis

Exploratory data analysis is important to help understand, and determine the completeness of the data set. This stage lets us know if our data set is complete and appropriate for the objectives we require and if there are null values or entry errors that need to be sorted out. In our Exploratory Data analysis, we will be cleaning our data for our project need and we will be having a closer look at 8 variables by getting their Descriptive statistics and making histograms & boxplots.

In [1]:
```
#Importing all the libraries needed.
%matplotlib inline
import pandas as pd
import os as os
import matplotlib.pyplot as plt # Matplot library for visulization
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.formula.api as smf
import statsmodels.api as sm2
```

In [2]:
```
#Defining path for the CSV file and impoting csv file in Jupyter using read_csv function
working_directory = os.getcwd() # gets the directory project file is in
print(working_directory)  #Printing to check
path = working_directory + '/baseball_teams.csv'
print(path) #Printing to check
file = pd.read_csv(path)
```

```
/Users/farazahmed/Downloads/Big data Analytics/Project
/Users/farazahmed/Downloads/Big data Analytics/Project/baseball_teams.csv
```

2.1 Data Cleaning

In this step we will have a closer look at our data set, remove columns that are missing information and make a dataframe storing data from 1960 to 2010.

In [3]:
```
file # Displaying the CSV file
```

Out[3]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games
0	1871	NaN	BS1	BNA	NaN	3	31	NaN	20	
1	1871	NaN	CH1	CNA	NaN	2	28	NaN	19	
2	1871	NaN	CL1	CFC	NaN	8	29	NaN	10	
3	1871	NaN	FW1	KEK	NaN	7	19	NaN	7	
4	1871	NaN	NY2	NNA	NaN	5	33	NaN	16	
...	...	...	...	...	...	...	...	...	...	
2800	2015	NL	LAN	LAD	W	1	162	81.0	92	
2801	2015	NL	SFN	SFG	W	2	162	81.0	84	
2802	2015	NL	ARI	ARI	W	3	162	81.0	79	
2803	2015	NL	SDN	SDP	W	4	162	81.0	74	
2804	2015	NL	COL	COL	W	5	162	81.0	68	

2805 rows × 43 columns

In [4]:
```
file.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           2526 non-null   float64
dtypes: float64(10), int64(23), object(10)
memory usage: 942.4+ KB
```

In [5]:
```
# Creating an dataframe with our required data set
```

In [6]:
```
df #displaying the data
```

Out[6]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Unnamed: 7	Games_Won	Games
1343	1960	AL	BAL	BAL	NaN	2	154	77.0	89	
1344	1960	AL	BOS	BOS	NaN	7	154	77.0	65	
1345	1960	AL	CHA	CHW	NaN	3	154	77.0	87	
1346	1960	NL	CHN	CHC	NaN	7	156	79.0	60	
1347	1960	NL	CIN	CIN	NaN	6	154	77.0	67	
...	...	...	...	...	...	...	...	...	...	...
2650	2010	NL	SLN	STL	C	2	162	81.0	86	
2651	2010	AL	TBA	TBD	E	1	162	81.0	96	
2652	2010	AL	TEX	TEX	W	1	162	81.0	90	
2653	2010	AL	TOR	TOR	E	4	162	78.0	85	
2654	2010	NL	WAS	WSN	E	5	162	81.0	69	

1312 rows × 43 columns

In [7]:
```
df.info() # Overview data set info
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1312 entries, 1343 to 2654
Data columns (total 43 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Year                 1312 non-null   int64  
 1   League               1312 non-null   object 
 2   Team                 1312 non-null   object 
 3   Franchise            1312 non-null   object 
 4   Division             1138 non-null   object 
 5   Final_Standing       1312 non-null   int64  
 6   Games_Played         1312 non-null   int64  
 7   Unnamed: 7           1312 non-null   float64
 8   Games_Won            1312 non-null   int64  
 9   Games_Lost           1312 non-null   int64  
 10  Unnamed: 10          1110 non-null   object 
 11  Unnamed: 11          474 non-null    object 
 12  League_Win           1284 non-null   object 
 13  World_Series         1284 non-null   object 
 14  Runs_Scored          1312 non-null   int64  
 15  At_Bats              1312 non-null   int64  
 16  Hits                 1312 non-null   int64  
 17  Doubles              1312 non-null   int64  
 18  Triples              1312 non-null   int64  
 19  Home_Runs            1312 non-null   int64  
 20  Walks                1312 non-null   int64  
 21  Strike_Outs          1312 non-null   float64
 22  Stolen_Bases         1312 non-null   float64
 23  Caught_Stealing      1312 non-null   float64
 24  Hit_By_Pitch         330 non-null    float64
 25  Sacrifice_Fly        330 non-null    float64
 26  Runs_Against         1312 non-null   int64  
 27  Earned_Runs          1312 non-null   int64  
 28  Earned_Run_Average   1312 non-null   float64
 29  Complete_Games       1312 non-null   int64  
 30  Shutout              1312 non-null   int64  
 31  Saves                1312 non-null   int64  
 32  Infield_Put_Outs     1312 non-null   int64  
 33  Hits_Allowed         1312 non-null   int64  
 34  Home_Run_Allowed     1312 non-null   int64  
 35  Walks_Allowed        1312 non-null   int64  
 36  Strikeouts_Allowed   1312 non-null   int64  
 37  Errors               1312 non-null   int64  
 38  Double_Plays         1312 non-null   float64
 39  Fielding_Percentage  1312 non-null   float64
 40  Team_Name            1312 non-null   object 
 41  Home_Ball_Park       1312 non-null   object 
 42  Attendance           1312 non-null   float64
dtypes: float64(10), int64(23), object(10)
memory usage: 451.0+ KB
```

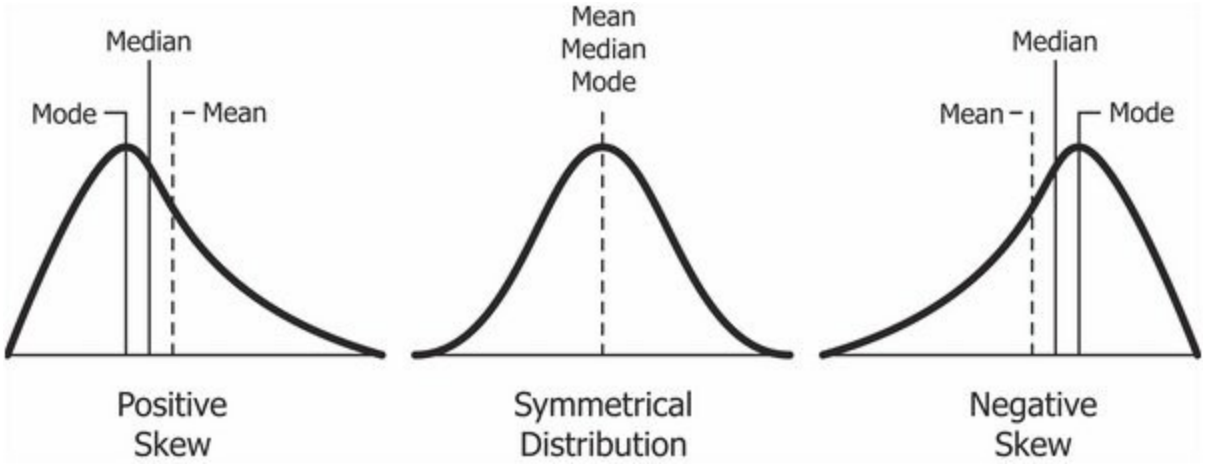
In [8]:

```
#Using for loop to drop Unnamed columns from our data frame
for nabra in df.columns:
   if 'Unnamed:' in nabra:
        del df[nabra]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1312 entries, 1343 to 2654
Data columns (total 40 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Year                 1312 non-null   int64  
 1   League               1312 non-null   object 
 2   Team                 1312 non-null   object 
 3   Franchise            1312 non-null   object 
 4   Division             1138 non-null   object 
 5   Final_Standing       1312 non-null   int64  
 6   Games_Played         1312 non-null   int64  
 7   Games_Won            1312 non-null   int64  
 8   Games_Lost           1312 non-null   int64  
 9   League_Win           1284 non-null   object 
 10  World_Series         1284 non-null   object 
 11  Runs_Scored          1312 non-null   int64  
 12  At_Bats              1312 non-null   int64  
 13  Hits                 1312 non-null   int64  
 14  Doubles              1312 non-null   int64  
 15  Triples              1312 non-null   int64  
 16  Home_Runs            1312 non-null   int64  
 17  Walks                1312 non-null   int64  
 18  Strike_Outs          1312 non-null   float64
 19  Stolen_Bases         1312 non-null   float64
 20  Caught_Stealing      1312 non-null   float64
 21  Hit_By_Pitch         330 non-null    float64
 22  Sacrifice_Fly        330 non-null    float64
 23  Runs_Against         1312 non-null   int64  
 24  Earned_Runs          1312 non-null   int64  
 25  Earned_Run_Average   1312 non-null   float64
 26  Complete_Games       1312 non-null   int64  
 27  Shutout              1312 non-null   int64  
 28  Saves                1312 non-null   int64  
 29  Infield_Put_Outs     1312 non-null   int64  
 30  Hits_Allowed         1312 non-null   int64  
 31  Home_Run_Allowed     1312 non-null   int64  
 32  Walks_Allowed        1312 non-null   int64  
 33  Strikeouts_Allowed   1312 non-null   int64  
 34  Errors               1312 non-null   int64  
 35  Double_Plays         1312 non-null   float64
 36  Fielding_Percentage  1312 non-null   float64
 37  Team_Name            1312 non-null   object 
 38  Home_Ball_Park       1312 non-null   object 
 39  Attendance           1312 non-null   float64
dtypes: float64(9), int64(23), object(8)
memory usage: 420.2+ KB
```

2.2 Closer look on 8 variables

The 8 variables we choose are Hits ,Runs_Scored ,Home_Runs ,Stolen_Bases ,Earned_Runs ,Walks_Allowed ,Hits_Allowed and Saves.



We will have a closer look at descriptive statistics for all the variables, we will look at their means, median, and modes. Also comment on their distribution, symmetry and width.

2.2.1 Descriptive statistics, Histogram & boxplots of 'Hits'

In [9]:
```
df['Hits'].describe()
```

Out[9]:
```
count    1312.000000
mean     1408.798780
std       125.978371
min       797.000000
25%      1357.000000
50%      1418.000000
75%      1489.000000
max      1684.000000
Name: Hits, dtype: float64
```

In [10]:
```
df['Hits'].median()
```

Out[10]:
```
1418.0
```

In [11]:
```
df['Hits'].skew()
```

Out[11]:
```
-1.4278154346382324
```

In [12]:
```
df['Hits'].kurtosis()
```

Out[12]:
```
3.799260421704792
```

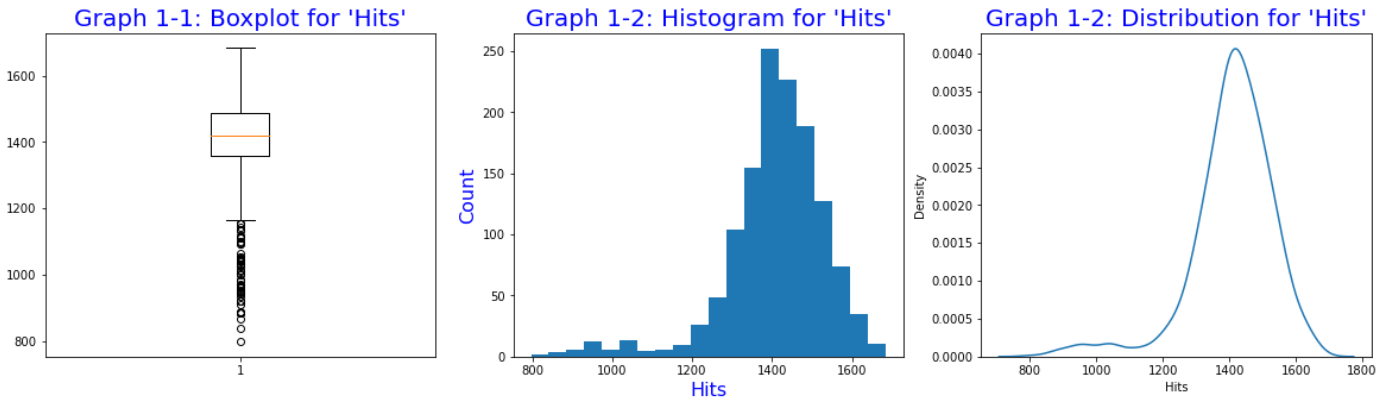
In [13]:
```
df['Hits'].mode()
```

Out[13]:
```
0    1415
Name: Hits, dtype: int64
```

In [14]:
```
#Setting figure size
plt.figure(figsize = (20,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Hits'])
plt.title("Graph 1-1: Boxplot for 'Hits'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Hits'], bins = 20)
plt.title("Graph 1-2: Histogram for 'Hits'", fontsize=20, color = 'Blue', )
plt.xlabel('Hits', fontsize=16, color = 'Blue')
```

```
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Hits'])
plt.title("Graph 1-2: Distribution for 'Hits'", fontsize=20, color = 'Blue', )
```

Out[14]:
```
Text(0.5, 1.0, "Graph 1-2: Distribution for 'Hits'")
```



- We only have **one** mode in the game at 1415
- We have data of **1312** hits with min hits at **797** and max at **1684**. As we have the mean at **1408** it means more number of 'Hits' in the game have been on the higher end.
- We can see from the graphs that the variable 'Hits' is not symmetrical and is negatively skewed so assumption that it normally distributed is negative. Also we have a many outliers on the lower end of the box graph. As we have a high kurtosis we can say our distribution is more narrow.

2.2.2 Descriptive statistics, Histogram & boxplots of 'Runs_Scored'

In [15]:
```
df['Runs_Scored'].describe()
```

Out[15]:
```
count    1312.000000
mean      703.445884
std       102.107114
min       329.000000
25%       641.000000
50%       704.000000
75%       771.000000
max      1009.000000
Name: Runs_Scored, dtype: float64
```

In [16]:
```
df['Runs_Scored'].mode()
```

Out[16]:
```
0    691
Name: Runs_Scored, dtype: int64
```

In [17]:
```
df['Runs_Scored'].skew()
```

Out[17]:
```
-0.20482883607847235
```

In [18]:
```
df['Runs_Scored'].kurtosis()
```

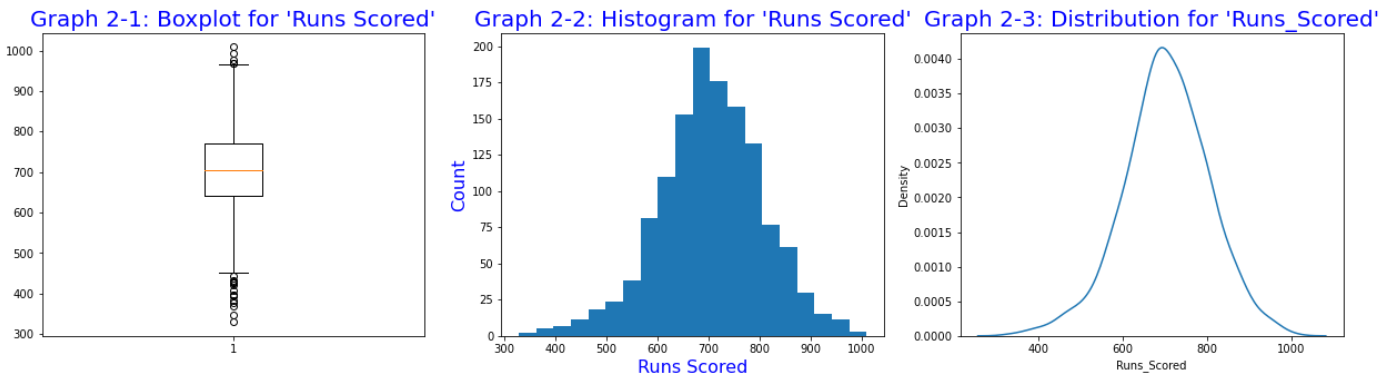
Out[18]:
```
0.43788467382393836
```

In [19]:
```
df['Runs_Scored'].median()
```

Out[19]:
```
704.0
```

In [20]:
```
#Setting figure size
plt.figure(figsize = (21,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Runs_Scored'],)
plt.title("Graph 2-1: Boxplot for 'Runs Scored'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Runs_Scored'], bins = 20)
plt.title("Graph 2-2: Histogram for 'Runs Scored'", fontsize=20, color = 'Blue')
plt.xlabel('Runs Scored', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Runs_Scored'])
plt.title("Graph 2-3: Distribution for 'Runs_Scored'", fontsize=20, color = 'Blue', )
```

Out[20]:
```
Text(0.5, 1.0, "Graph 2-3: Distribution for 'Runs_Scored'")
```



- Our mean **703.445884** is very close to our median **704.0**
- We have one mode at **691**
- We have low skew and kurtosis on this variable we can say that the graph is very close to a normal distribution and very close to being perfectly symmetrical. Graph 2-1 indicates we have more outliers in the lower end. As our kurtosis is near zero we can say we have Mesokurtic distribution.

2.2.3 Descriptive statistics, Histogram & boxplots of 'Home_Runs'

In [21]:
```
df['Home_Runs'].describe()
```

Out[21]:
```
count    1312.000000
mean      142.350610
std        40.518801
min        32.000000
25%       113.000000
50%       139.000000
75%       169.000000
max       264.000000
Name: Home_Runs, dtype: float64
```

In [22]:
```
df['Home_Runs'].mode()
```

Out[22]:
```
0    125
Name: Home_Runs, dtype: int64
```

In [23]:
```
df['Home_Runs'].median()
```

Out[23]:
```
139.0
```

In [24]:
```
df['Home_Runs'].skew()
```

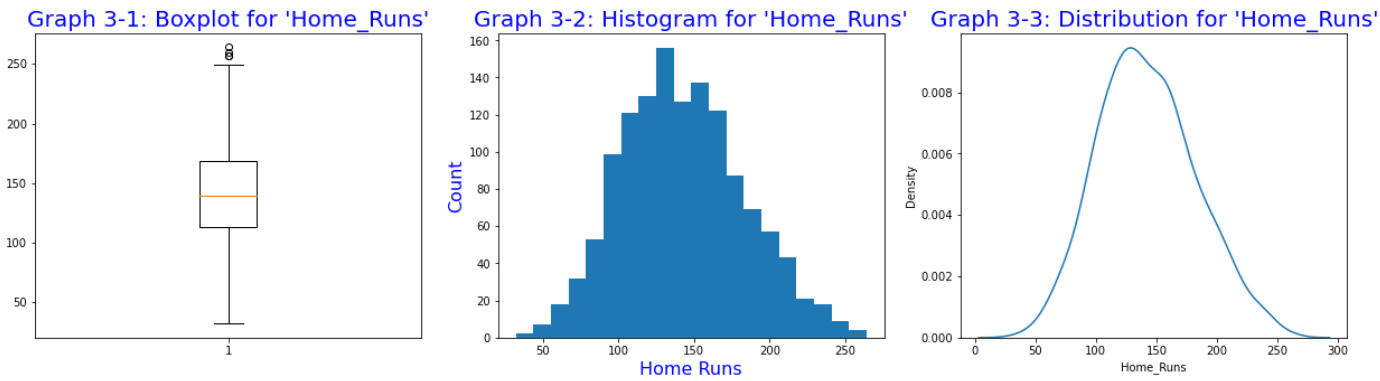
Out[24]:
```
0.2571148356699839
```

In [25]:
```
df['Home_Runs'].kurtosis()
```

Out[25]:
```
-0.25793442878669737
```

In [26]:
```
#Setting figure size
plt.figure(figsize = (21,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Home_Runs'],)
plt.title("Graph 3-1: Boxplot for 'Home_Runs'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Home_Runs'], bins = 20)
plt.title("Graph 3-2: Histogram for 'Home_Runs'", fontsize=20, color = 'Blue')
plt.xlabel('Home Runs', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Home_Runs'])
plt.title("Graph 3-3: Distribution for 'Home_Runs'", fontsize=20, color = 'Blue', )
```

Out[26]:
```
Text(0.5, 1.0, "Graph 3-3: Distribution for 'Home_Runs'")
```



- Our mean for the variable is **142.350610** and our median is **139.0**
- We only have one mode at **125**
- The variable is not symmetrical, as we can see from the Graph 3-2, we do outliers at the both end. It is close but not normally distributed. As our kurtosis is negative we can say our distribution is wider.

2.2.4 Descriptive statistics, Histogram & boxplots of 'Stolen_Bases'

In [27]:
```
df['Stolen_Bases'].describe()
```

Out[27]:
```
count    1312.000000
mean      100.237043
std        42.009582
min        16.000000
25%        69.000000
50%        95.500000
75%       126.000000
max       341.000000
Name: Stolen_Bases, dtype: float64
```

In [28]:
```
df['Stolen_Bases'].mode()
```

Out[28]:
```
0    100.0
Name: Stolen_Bases, dtype: float64
```

In [29]:
```
df['Stolen_Bases'].median()
```

Out[29]:
```
95.5
```

In [30]:
```
df['Stolen_Bases'].skew()
```

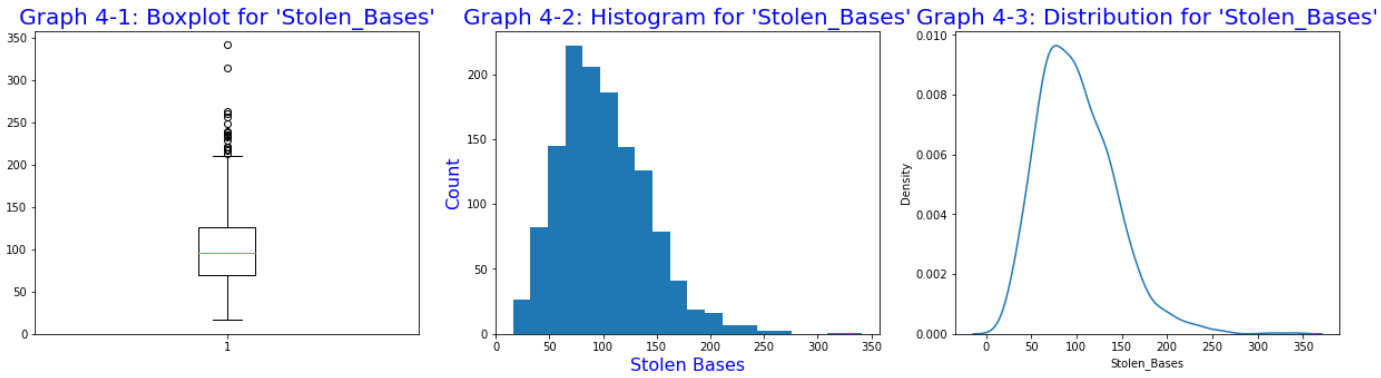
Out[30]:
```
0.8774065018139938
```

In [31]:
```
df['Stolen_Bases'].kurtosis()
```

Out[31]:
```
1.592514517074306
```

In [32]:
```
#Setting figure size
plt.figure(figsize = (21,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Stolen_Bases'],)
plt.title("Graph 4-1: Boxplot for 'Stolen_Bases'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Stolen_Bases'], bins = 20)
plt.title("Graph 4-2: Histogram for 'Stolen_Bases'", fontsize=20, color = 'Blue')
plt.xlabel('Stolen Bases', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Stolen_Bases'])
plt.title("Graph 4-3: Distribution for 'Stolen_Bases'", fontsize=20, color = 'Blue', )
```

Out[32]:
```
Text(0.5, 1.0, "Graph 4-3: Distribution for 'Stolen_Bases'")
```



- Our mean for the variable is **100.237043** close our median which is **95.5**
- Our mean and mode are same **100**
- Our graphs indicate that our variable data is positively skewed. We have few outliers on the higher end as in Graph 4-1. It is not symmetrical and normally distributed. As Kurtosis is high we can say it's a narrow graph.

2.2.5 Descriptive statistics, Histogram & boxplots of 'Earned_Runs'

In [33]:
```
df['Earned_Runs'].describe()
```

Out[33]:
```
count    1312.000000
mean      633.310976
std        99.746433
min       293.000000
25%       569.000000
50%       629.000000
75%       698.000000
max      1015.000000
Name: Earned_Runs, dtype: float64
```

In [34]:
```
df['Earned_Runs'].mode()
```

Out[34]:
```
0    598
1    615
2    621
3    675
Name: Earned_Runs, dtype: int64
```

In [35]:
```
df['Earned_Runs'].median()
```

Out[35]:
```
629.0
```

In [36]:
```
df['Earned_Runs'].skew()
```

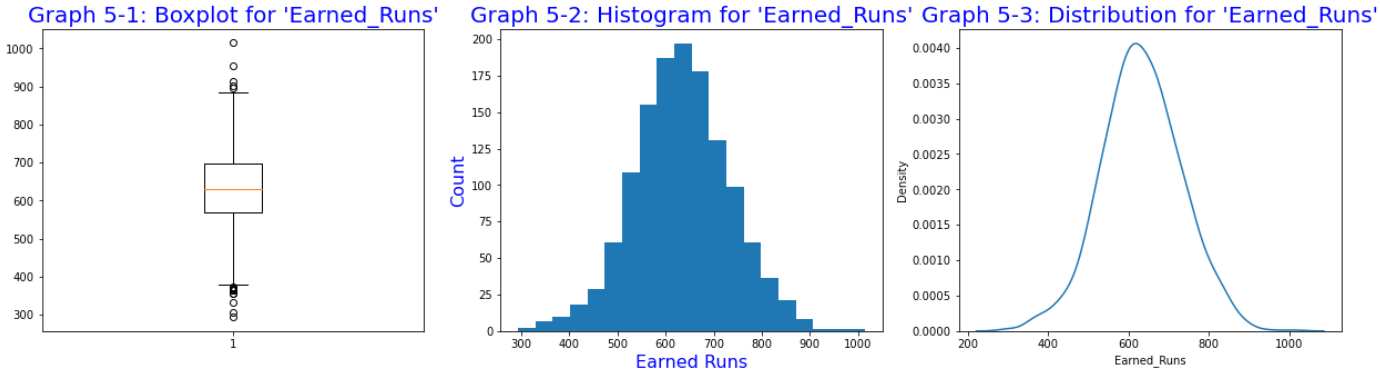
Out[36]:
```
-0.00387615925958595
```

In [37]:
```
df['Earned_Runs'].kurtosis()
```

Out[37]:
```
0.22593894570623174
```

In [38]:
```
#Setting figure size
plt.figure(figsize = (21,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Earned_Runs'],)
plt.title("Graph 5-1: Boxplot for 'Earned_Runs'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Earned_Runs'], bins = 20)
plt.title("Graph 5-2: Histogram for 'Earned_Runs'", fontsize=20, color = 'Blue')
plt.xlabel('Earned Runs', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Earned_Runs'])
plt.title("Graph 5-3: Distribution for 'Earned_Runs'", fontsize=20, color = 'Blue', )
```

Out[38]:
```
Text(0.5, 1.0, "Graph 5-3: Distribution for 'Earned_Runs'")
```



We have four modes at **598, 615, 621, and 675**

- Our mean at **633.31** is close to our **629.0**
- We have low skew on this variable we can say that the graph is very close to a normal distribution and very close to being perfectly symmetrical. Graph 5-1 indicates we have outiers on both lower and higher end. Our graph is narrow as we have positive kurtosis.

2.2.6 Descriptive statistics, Histogram & boxplots of 'Walks_Allowed'

In [39]:
```
df['Walks_Allowed'].describe()
```

Out[39]:
```
count    1312.000000
mean      524.651677
std        75.589894
min       268.000000
25%       478.000000
50%       525.000000
75%       573.250000
max       784.000000
Name: Walks_Allowed, dtype: float64
```

In [40]:
```
df['Walks_Allowed'].mode()
```

Out[40]:
```
0    504
1    518
Name: Walks_Allowed, dtype: int64
```

In [41]:
```
df['Walks_Allowed'].median()
```

Out[41]:
```
525.0
```

In [42]:
```
df['Walks_Allowed'].skew()
```

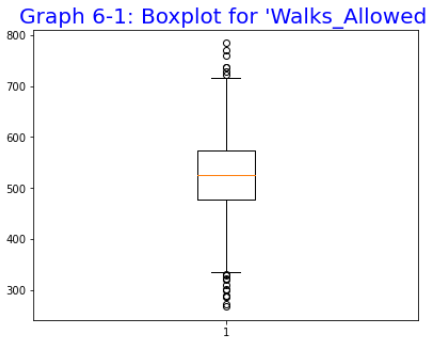
Out[42]:
```
-0.07152723544972032
```

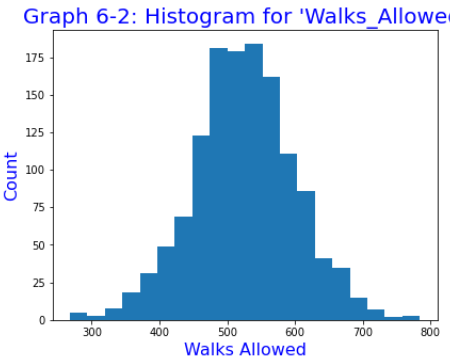
In [43]:
```
df['Walks_Allowed'].kurtosis()
```

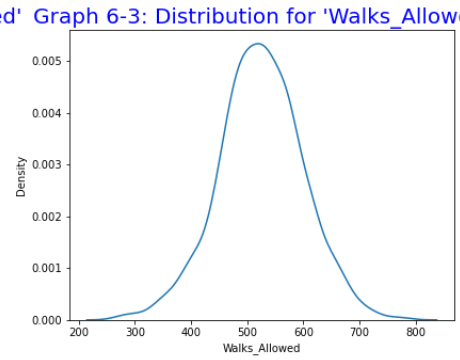
Out[43]:
```
0.3670240188296381
```

In [44]:
```
#Setting figure size
plt.figure(figsize = (22,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Walks_Allowed'],)
plt.title("Graph 6-1: Boxplot for 'Walks_Allowed'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Walks_Allowed'], bins = 20)
plt.title("Graph 6-2: Histogram for 'Walks_Allowed'", fontsize=20, color = 'Blue')
plt.xlabel('Walks Allowed', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Walks_Allowed'])
plt.title("Graph 6-3: Distribution for 'Walks_Allowed'", fontsize=20, color = 'Blue', )
```

Out[44]:
```
Text(0.5, 1.0, "Graph 6-3: Distribution for 'Walks_Allowed'")
```







- Our mean for the variable is 524.651677 and our median is **525.0**
- We have two modes at **504 and 518**
- We have low skew on this variable we can say that the graph is very close to a normal distribution and very close to being perfectly symmetrical. Graph 6-1 indicates we have outiers on both lower and higher end. Our graph is narrow as we have positive kurtosis(Leptokutric).

2.2.7 Descriptive statistics, Histogram & boxplots of 'Hits_Allowed'

In [45]:
```
df['Hits_Allowed'].describe()
```

Out[45]:
```
count    1312.000000
mean     1408.827744
std       128.065442
min       827.000000
25%      1350.000000
50%      1421.500000
75%      1488.250000
max      1734.000000
Name: Hits_Allowed, dtype: float64
```

In [46]:
```
df['Hits_Allowed'].mode()
```

Out[46]:
```
0    1402
1    1443
Name: Hits_Allowed, dtype: int64
```

In [47]:
```
df['Hits_Allowed'].median()
```

Out[47]:
```
1421.5
```

In [48]:
```
df['Hits_Allowed'].skew()
```

Out[48]:
```
-1.3469235822796282
```

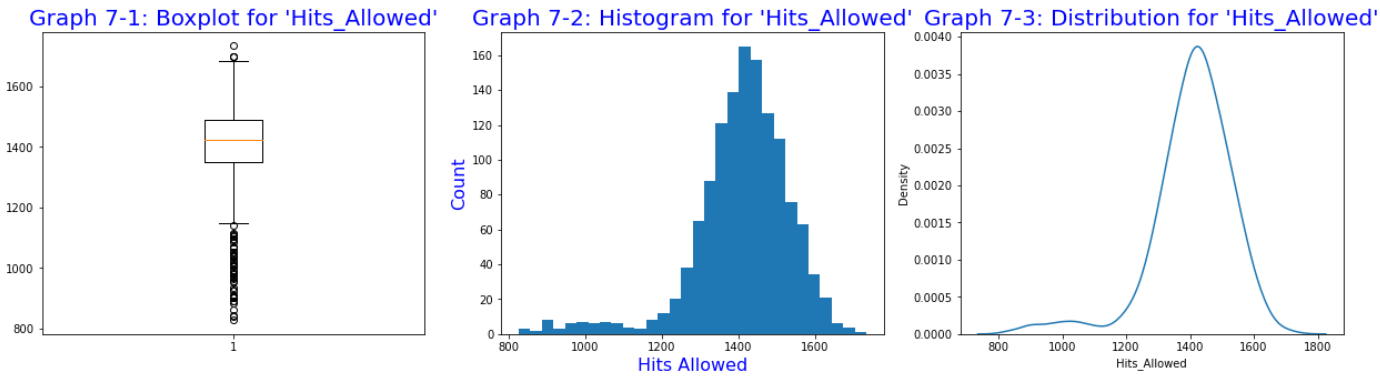
In [49]:
```
df['Hits_Allowed'].kurtosis()
```

Out[49]:
```
3.5746120336337746
```

In [50]:
```
#Setting figure size
plt.figure(figsize = (21,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Hits_Allowed'],)
plt.title("Graph 7-1: Boxplot for 'Hits_Allowed'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Hits_Allowed'], bins = 30)
```

```
plt.title("Graph 7-2: Histogram for 'Hits_Allowed'", fontsize=20, color = 'Blue')
plt.xlabel('Hits Allowed', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Hits_Allowed'])
plt.title("Graph 7-3: Distribution for 'Hits_Allowed'", fontsize=20, color = 'Blue' )
```

Out[50]:
```
Text(0.5, 1.0, "Graph 7-3: Distribution for 'Hits_Allowed'")
```



- Our mean for the variable is 1408.827744 and our median is **1421.5**
- We have two modes at **1402 and** 1443__
- We can see from the graphs that the variable 'Hits' is not symmetrical and is negatively skewed so assumption that it normally distributed is negative. Also graph 7-1 indicates that we have high number of outliers in the lower end. High kurtosis indicate that our distribution is narrow.

2.2.8 Descriptive statistics, Histogram & boxplots of 'Saves'

In [51]:
```
df['Saves'].describe()
```

Out[51]:
```
count    1312.000000
mean       36.162348
std         9.363350
min        10.000000
25%        30.000000
50%        36.000000
75%        43.000000
max        68.000000
Name: Saves, dtype: float64
```

In [52]:
```
df['Saves'].mode()
```

Out[52]:
```
0    39
Name: Saves, dtype: int64
```

In [53]:
```
df['Saves'].median()
```

Out[53]:
```
36.0
```

In [54]:
```
df['Saves'].skew()
```

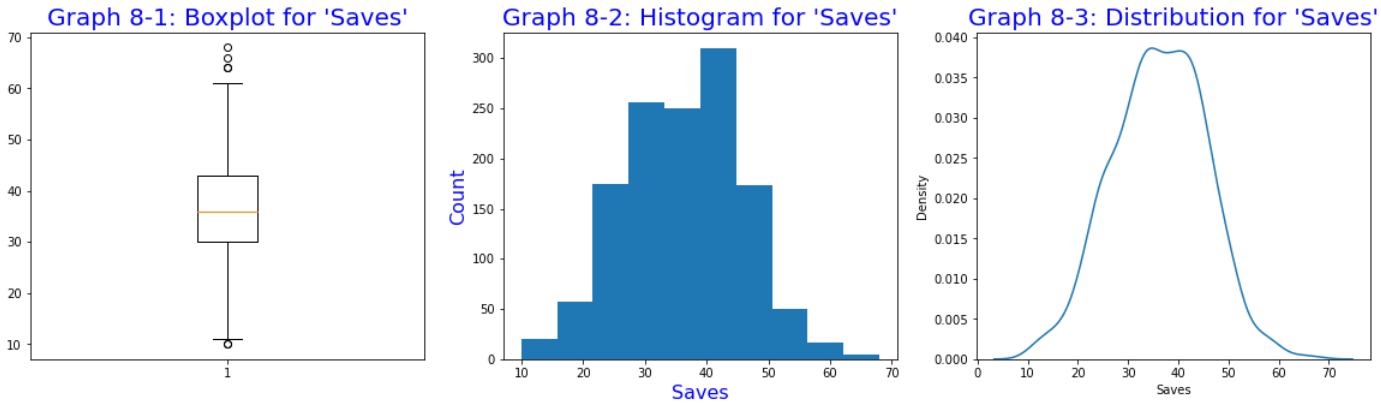
Out[54]:
```
-0.05172918289277304
```

In [55]:
```
df['Saves'].kurtosis()
```

Out[55]:
```
-0.14450579780805262
```

In [56]:
```
#Setting figure size
plt.figure(figsize = (20,5))
#Code for boxplot
plt.subplot(1,3,1)
p1 = plt.boxplot(df['Saves'],)
plt.title("Graph 8-1: Boxplot for 'Saves'", fontsize=20, color = 'Blue')
#Code for histogram
plt.subplot(1,3,2)
p2 = plt.hist(df['Saves'])
plt.title("Graph 8-2: Histogram for 'Saves'", fontsize=20, color = 'Blue')
plt.xlabel('Saves', fontsize=16, color = 'Blue')
plt.ylabel('Count', fontsize=16, color = 'Blue')
#code for distribution graph
plt.subplot(1,3,3)
sns.kdeplot(df['Saves'])
plt.title("Graph 8-3: Distribution for 'Saves'", fontsize=20, color = 'Blue' )
```

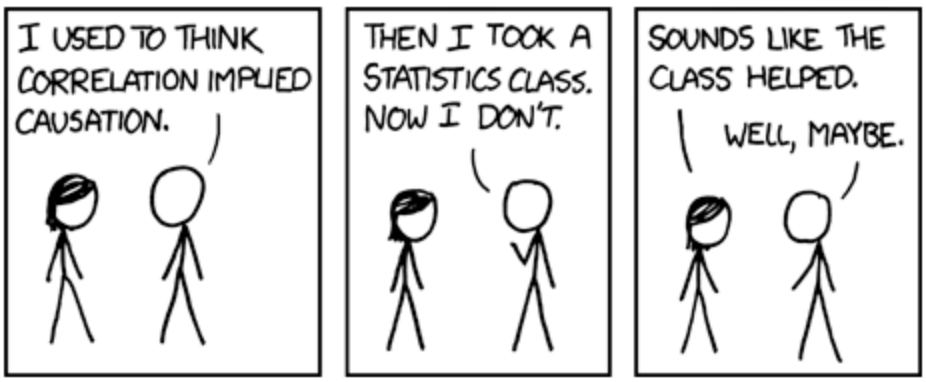
Out[56]:
```
Text(0.5, 1.0, "Graph 8-3: Distribution for 'Saves'")
```



- Our mean for the variable is 36.162348 and our median is **36.0**
- We have two modes at **39**
- Our graph is not symmetrical, negative kurtosis indicates that the distribution is wider there are more outliers on the higher end as graph 8-1 indicates, our assumption that distribution is normal is also negative.

3. Correlation Analysis

Correlation analysis is used to test relationships between different variables.



3.1 Creating a column 'Run difference'

In [57]:
```
# creating a column Run difference
df['Run_difference'] = df['Runs_Scored'] - df['Runs_Against']
df # checking for updated data frame.
```

```
/var/folders/nf/m1b0vpyx24bgc7qljpvffr8r0000gn/T/ipykernel_2246/789179352.py:2: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  df['Run_difference'] = df['Runs_Scored'] - df['Runs_Against']
```

Out[57]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Games_Won	Games_Lost	Lea
1343	1960	AL	BAL	BAL	NaN	2	154	89	65	
1344	1960	AL	BOS	BOS	NaN	7	154	65	89	
1345	1960	AL	CHA	CHW	NaN	3	154	87	67	
1346	1960	NL	CHN	CHC	NaN	7	156	60	94	
1347	1960	NL	CIN	CIN	NaN	6	154	67	87	
...	...	...	...	...	...	...	...	...	...	...
2650	2010	NL	SLN	STL	C	2	162	86	76	
2651	2010	AL	TBA	TBD	E	1	162	96	66	
2652	2010	AL	TEX	TEX	W	1	162	90	72	
2653	2010	AL	TOR	TOR	E	4	162	85	77	
2654	2010	NL	WAS	WSN	E	5	162	69	93	

1312 rows × 41 columns

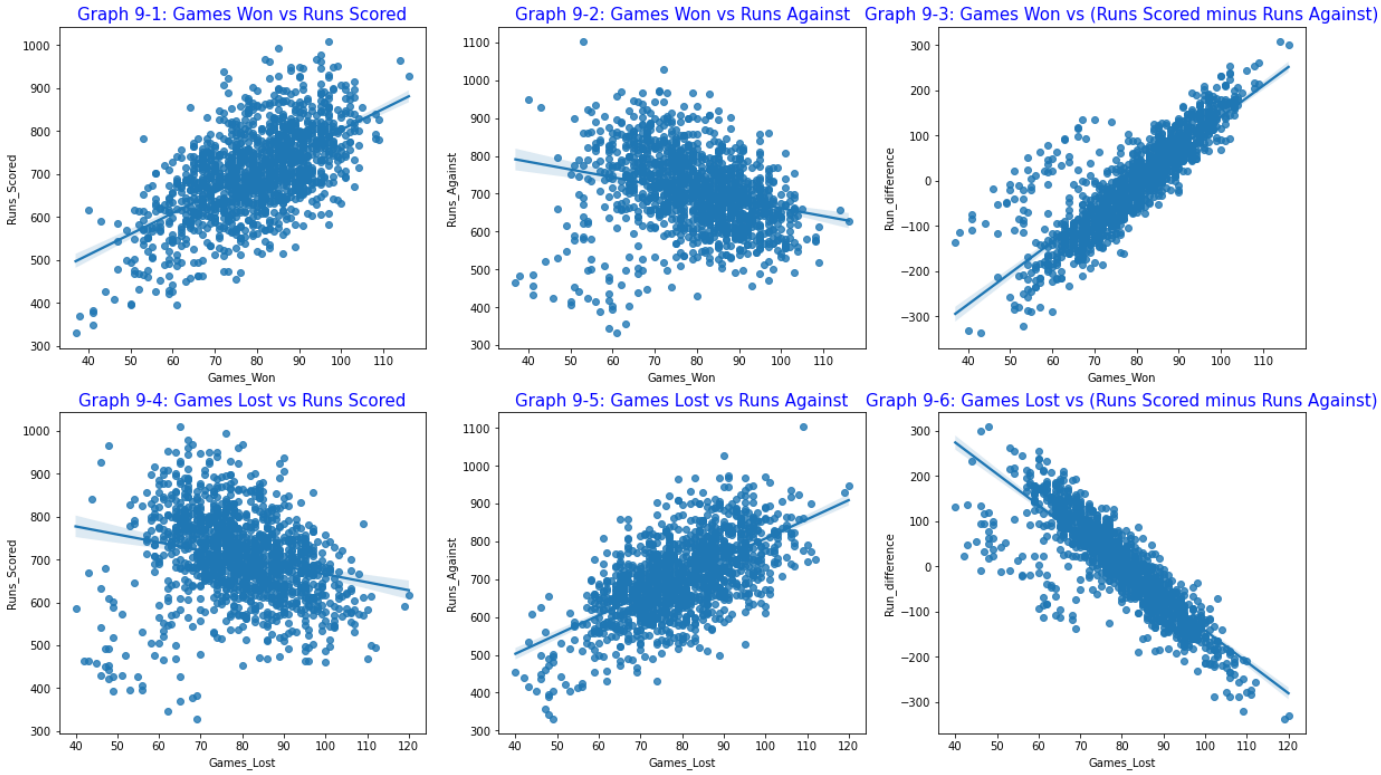
3.2 Linear plots and heatmap

In [58]:
```
plt.figure(figsize = (20,18))

plt.subplot(3,3,1)
plt.title("Graph 9-1: Games Won vs Runs Scored", fontsize=15, color = 'Blue')
sns.regplot(x=df["Games_Won"],y=df["Runs_Scored"])

plt.subplot(3,3,2)
plt.title("Graph 9-2: Games Won vs Runs Against", fontsize=15, color = 'Blue')
sns.regplot(x=df["Games_Won"],y=df["Runs_Against"])

plt.subplot(3,3,3)
plt.title("Graph 9-3: Games Won vs (Runs Scored minus Runs Against)", fontsize=15, color
x=df["Games_Won"],y=df["Run_difference"])
```

```
plt.subplot(3,3,4)
plt.title("Graph 9-4: Games Lost vs Runs Scored", fontsize=15, color = 'Blue')
sns.regplot(x=df["Games_Lost"],y=df["Runs_Scored"])

plt.subplot(3,3,5)
plt.title("Graph 9-5: Games Lost vs Runs Against", fontsize=15, color = 'Blue')
sns.regplot(x=df["Games_Lost"],y=df["Runs_Against"])

plt.subplot(3,3,6)
plt.title("Graph 9-6: Games Lost vs (Runs Scored minus Runs Against)", fontsize=15, colo
sns.regplot(x=df["Games_Lost"],y=df["Run_difference"])
```

Out[58]:

```
<AxesSubplot:title={'center':'Graph 9-6: Games Lost vs (Runs Scored minus Runs Agains
t)'}, xlabel='Games_Lost', ylabel='Run_difference'>
```



In [59]:

```
df[['Games_Won','Games_Lost','Runs_Scored','Runs_Against','Run_difference']].corr()
```

Out[59]:

	Games_Won	Games_Lost	Runs_Scored	Runs_Against	Run_difference
Games_Won	1.000000	-0.647686	0.595828	-0.248158	0.850411
Games_Lost	-0.647686	1.000000	-0.226772	0.611177	-0.849358
Runs_Scored	0.595828	-0.226772	1.000000	0.510362	0.484345
Runs_Against	-0.248158	0.611177	0.510362	1.000000	-0.505167
Run_difference	0.850411	-0.849358	0.484345	-0.505167	1.000000

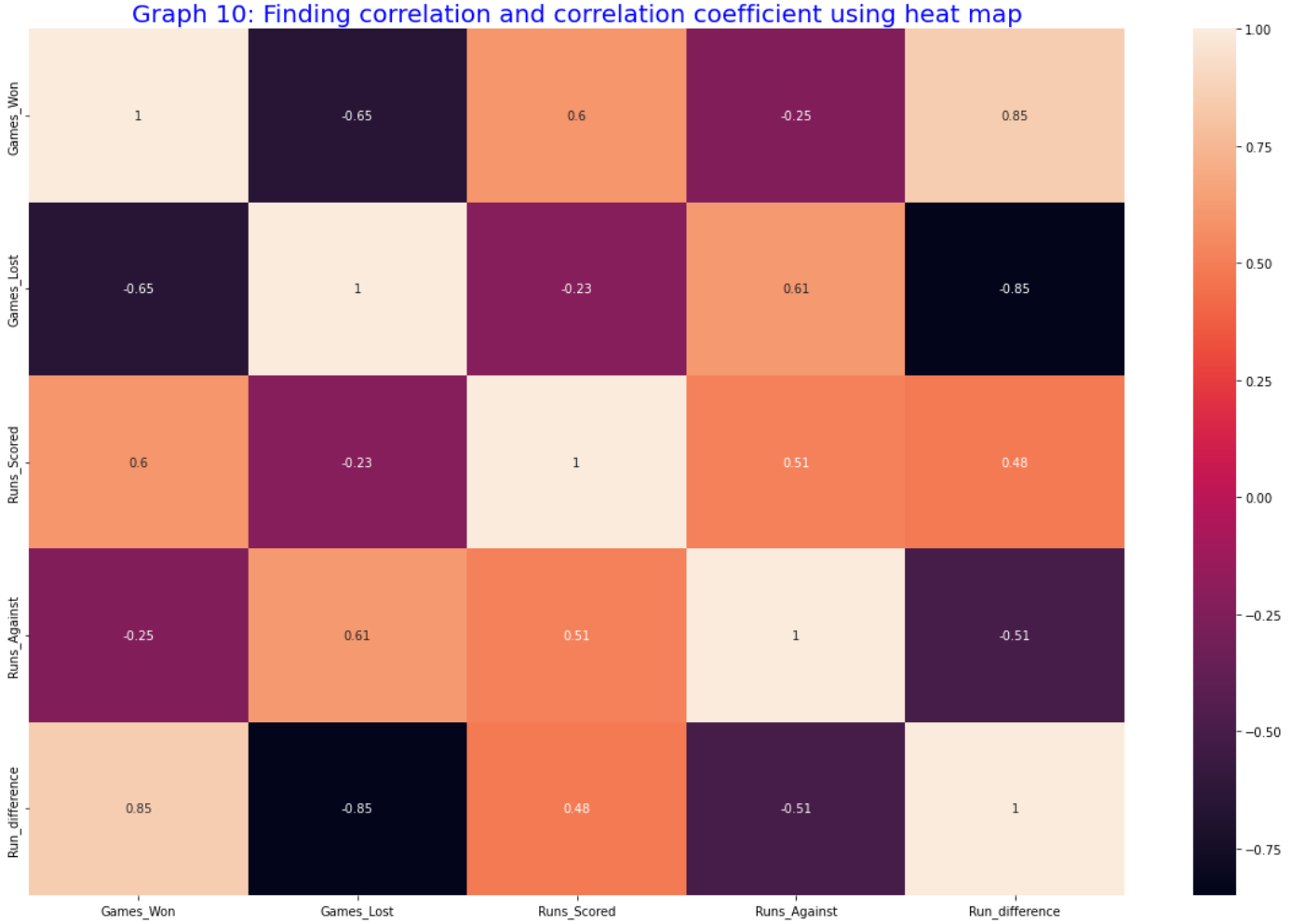
In [60]:

```
import seaborn as sns
plt.figure(figsize = (20,13))
plt.title("Graph 10: Finding correlation and correlation coefficient using heat map", fo
sns.heatmap(df[['Games_Won','Games_Lost','Runs_Scored','Runs_Against','Run_difference']]
```

Out[60]:

```
<AxesSubplot:title={'center':'Graph 10: Finding correlation and correlation coefficient
using heat map'}>
```

Graph 10: Finding correlation and correlation coefficient using heat map



3.3 Analysis conclusion

- By looking at the different graphs of 'Games_Won','Games_Lost','Runs_Scored','Runs_Against', and 'Run_difference' we can conclude that of we have a positive correlation between Games won and runs scored but when we account for the run Against and make a variable run difference(Runs Scored minus Runs Against) the correlation gets very high.
- Similarly we can conclude that of we have a negative correlation between Games lost and runs scored, and it gets more prominent when we correlate games lost to run difference.

Creating a multiple linear regression model for each of the 4 time periods

Period 1 - before 1920 Period 2 - 1920 to 1960 Period 3 - 1960 to 1990 Period 4 - 1990 to 2010

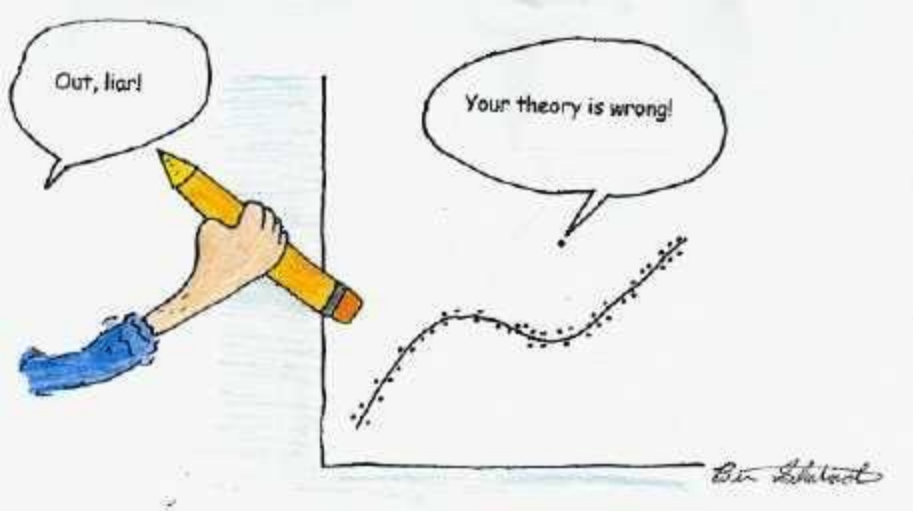
In [110…
```
#Defining path for the CSV file and impoting csv file in Jupyter using read_csv function
working_directory = os.getcwd() # gets the directory project file is in
path = working_directory + '/baseball_teams.csv'
print(path) # to check
file2 = pd.read_csv(path)
```

```
/Users/farazahmed/Downloads/Big data Analytics/Project/baseball_teams.csv
```

In [113…
```
#Using for loop to drop Unnamed columns from our data frame
for nabra in file2.columns:
   if 'Unnamed:' in nabra:
        del file2[nabra]
```

4. Selecting good independent variables for our model

To find out best variables we will be creating a heat map correlation of "Games_won" with all offensive and defensive variables



In [115…

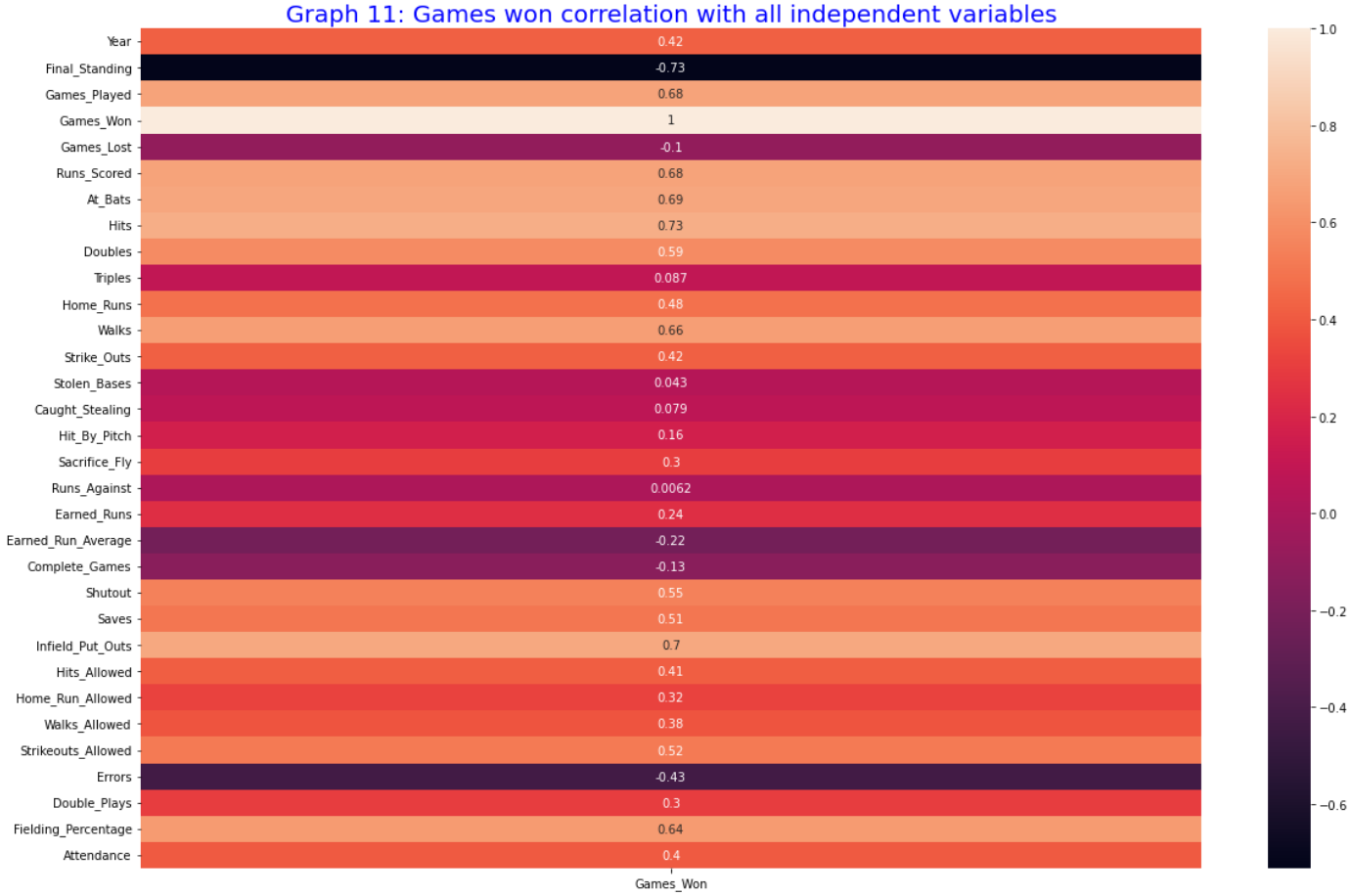
```
file2.info() # to check for variables with complete dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2805 entries, 0 to 2804
Data columns (total 40 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   Games_Won            2805 non-null   int64  
 8   Games_Lost           2805 non-null   int64  
 9   League_Win           2777 non-null   object 
 10  World_Series         2448 non-null   object 
 11  Runs_Scored          2805 non-null   int64  
 12  At_Bats              2805 non-null   int64  
 13  Hits                 2805 non-null   int64  
 14  Doubles              2805 non-null   int64  
 15  Triples              2805 non-null   int64  
 16  Home_Runs            2805 non-null   int64  
 17  Walks                2805 non-null   int64  
 18  Strike_Outs          2685 non-null   float64
 19  Stolen_Bases         2661 non-null   float64
 20  Caught_Stealing      1946 non-null   float64
 21  Hit_By_Pitch         480 non-null    float64
 22  Sacrifice_Fly        480 non-null    float64
 23  Runs_Against         2805 non-null   int64  
 24  Earned_Runs          2805 non-null   int64  
 25  Earned_Run_Average   2805 non-null   float64
 26  Complete_Games       2805 non-null   int64  
 27  Shutout              2805 non-null   int64  
 28  Saves                2805 non-null   int64  
 29  Infield_Put_Outs     2805 non-null   int64  
 30  Hits_Allowed         2805 non-null   int64  
 31  Home_Run_Allowed     2805 non-null   int64  
 32  Walks_Allowed        2805 non-null   int64  
 33  Strikeouts_Allowed   2805 non-null   int64  
 34  Errors               2805 non-null   int64  
 35  Double_Plays         2488 non-null   float64
 36  Fielding_Percentage  2805 non-null   float64
 37  Team_Name            2805 non-null   object 
 38  Home_Ball_Park       2771 non-null   object 
 39  Attendance           2526 non-null   float64
dtypes: float64(9), int64(23), object(8)
memory usage: 876.7+ KB
```

In [63]:
```
filecor = file2.corr()
plt.figure(figsize = (20,13))
plt.title("Graph 11: Games won correlation with all independent variables", fontsize=20,
filecor.drop(['Year','Final_Standing','Games_Played','Games_Lost','Runs_Scored','At_Bats
sns.heatmap(filecor, annot = True)
```

Out[63]:
```
<AxesSubplot:title={'center':'Graph 11: Games won correlation with all independent varia
bles'}>
```

Graph 11: Games won correlation with all independent variables



Based on on our correlation chart we will use 3 high correlation offensive and 3 high correlation defensive for our multiple linear regression model

```
Offensive: 'Runs_Scored', 'At_Bats', 'Hits'
Defensive: 'Infield_Put_Outs', 'Shutout','Saves'
```

To answer our business question "Determine if teams are more successful with high offenses, tight defenses or a balance of the two" we will also use data set of combined variables in all periods

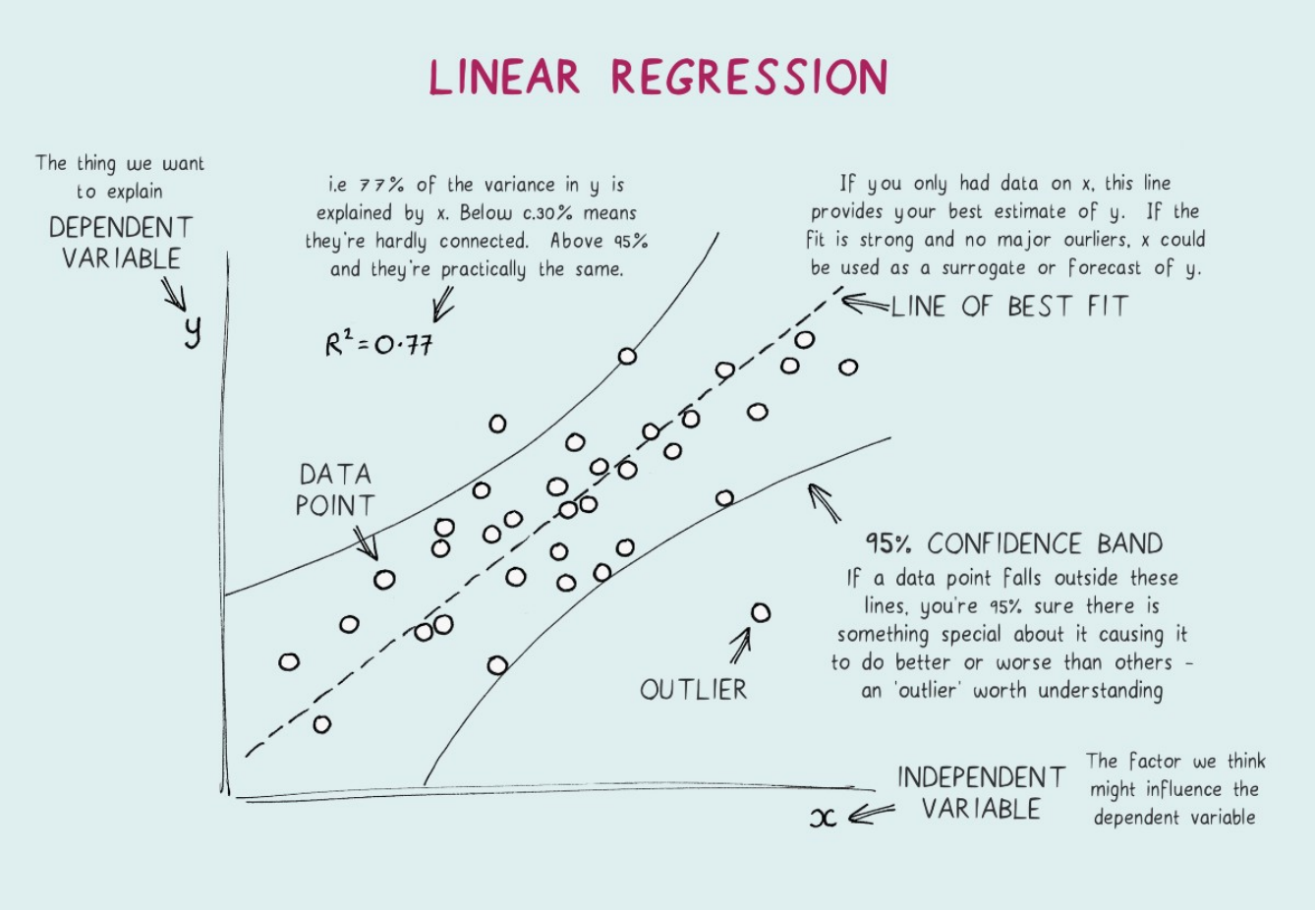
```
Combined: 'Runs_Scored', 'At_Bats', 'Hits','Infield_Put_Outs',
'Shutout','Saves'
```

5. Linear regression model and it's evaluation for each of the 4 time periods

What is linear regression?

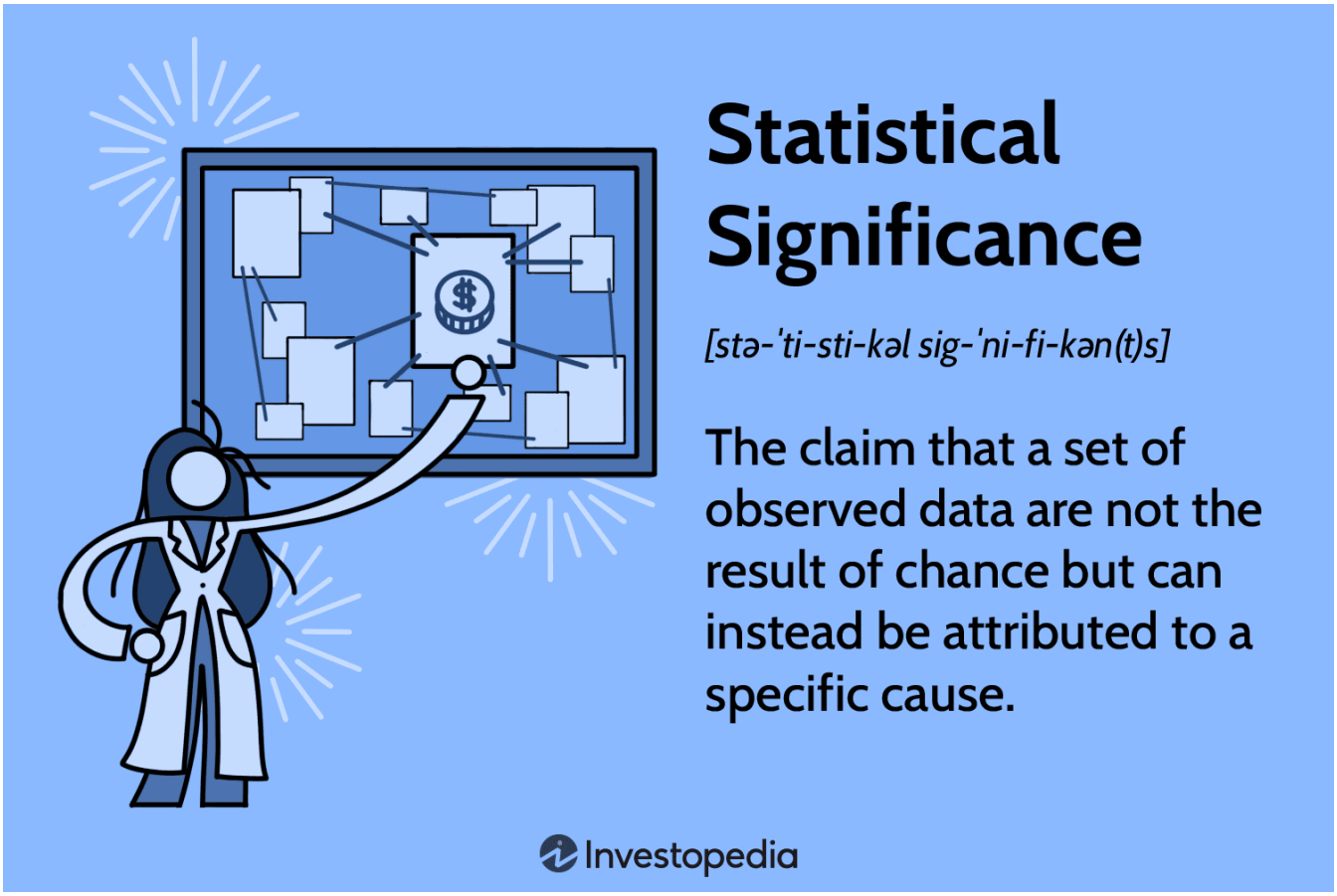
Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

Source:-IBM



What is statistical significance?

“Statistical significance helps quantify whether a result is likely due to chance or to some factor of interest,” says Redman. When a finding is significant, it simply means you can feel confident that’s it real, not that you just got lucky (or unlucky) in choosing the sample.
Source: Havard business review



5.1 Period 1 - before 1920

In [109…
```
# creating data fram for before 1920 period
dfp1 = file2[(file2.Year <= 1920)]
```

Defining x and y variables

In [65]:
```
# defining dependent variable y
y_combined = dfp1['Games_Won'].values
y_offensive = dfp1['Games_Won'].values
y_defensive = dfp1['Games_Won'].values
# defining independent variables x
x_combined = dfp1[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']]
x_offensive = dfp1[['Runs_Scored','At_Bats','Hits']].values
x_defensive = dfp1[['Infield_Put_Outs','Shutout','Saves']].values
```

Spliting training set and test set from the data set

In [66]:
```
# Combined varaibles data set
xp1_combined_train,xp1_combined_test,yp1_combined_train,yp1_combined_test = train_test_s
# Offensive varaibles data set
xp1_offensive_train,xp1_offensive_test,yp1_offensive_train,yp1_offensive_test = train_te
# Defensive varaibles data set
xp1_defensive_train,xp1_defensive_test,yp1_defensive_train,yp1_defensive_test = train_te
```

Training the model on the training set

In [67]:
```
# Training our Combined data set model
mlp1_combined = LinearRegression()
mlp1_combined.fit(xp1_combined_train,yp1_combined_train)
# Training our Offensive data set model
mlp1_offensive = LinearRegression()
mlp1_offensive.fit(xp1_offensive_train,yp1_offensive_train)
# Training our Defensive data set model
mlp1_defensive = LinearRegression()
mlp1_defensive.fit(xp1_defensive_train,yp1_defensive_train)
```

Out[67]:
```
LinearRegression()
```

Predicting the test set results

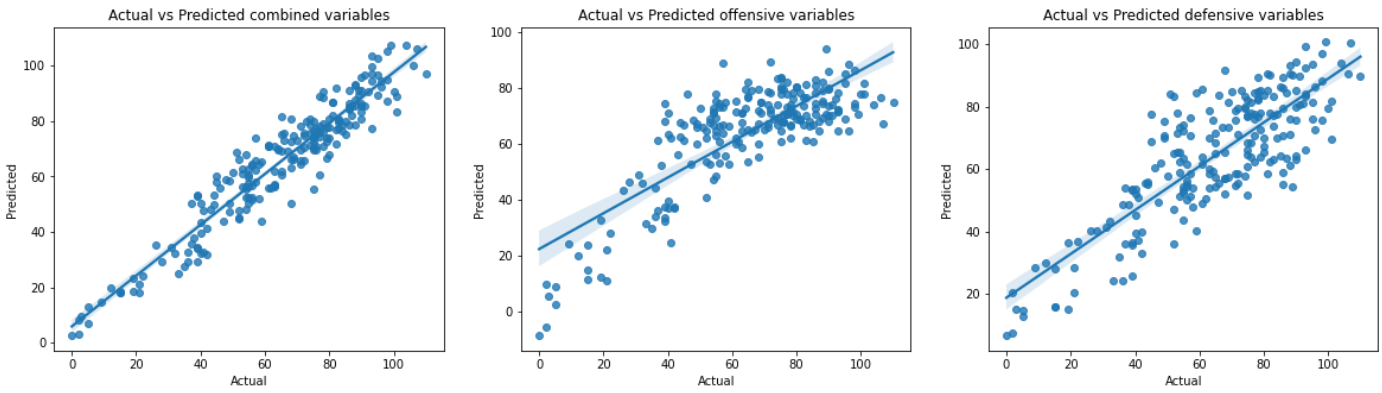
In [68]:
```
yp1_combined_pred = mlp1_combined.predict(xp1_combined_test) # combined data set perdict
yp1_offensive_pred = mlp1_offensive.predict(xp1_offensive_test) # offensive data set per
yp1_defensive_pred = mlp1_defensive.predict(xp1_defensive_test) # defensive data set per
```

Plotting our Multiple linear regression models

In [69]:
```
plt.figure(figsize = (20,5))

plt.subplot(1,3,1)
lp1_combined = sns.regplot(yp1_combined_test,yp1_combined_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted combined variables')

plt.subplot(1,3,2)
lp1_offensive = sns.regplot(yp1_offensive_test,yp1_offensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted offensive variables')

plt.subplot(1,3,3)
lp1_defensive = sns.regplot(yp1_defensive_test,yp1_defensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted defensive variables')
```

```
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[69]:
```
Text(0.5, 1.0, 'Actual vs Predicted defensive variables')
```



Evaluating each model

In [70]:
```
we=r2_score(yp1_combined_test,yp1_combined_pred)
wee=r2_score(yp1_offensive_test,yp1_offensive_pred)
weee=r2_score(yp1_defensive_test,yp1_defensive_pred)
print('Combined',we,' Offensive',wee,' Defensive',weee)
```

```
Combined 0.9070888601292032  Offensive 0.6425574790713666  Defensive 0.6830983308215997
```

Based on our regplot and r2 values we can see that model with Combined offensive and defensive independent variable is better.

Statistical summary of our most successful model

In [71]:
```
lm1 = smf.ols(formula="Games_Won ~ Runs_Scored + At_Bats + Hits + Infield_Put_Outs + Shu
lm1.summary()
```

Out[71]:

OLS Regression Results

Dep. Variable:	Games_Won	**R-squared:**	0.902
Model:	OLS	**Adj. R-squared:**	0.901
Method:	Least Squares	**F-statistic:**	1092.
Date:	Wed, 23 Nov 2022	**Prob (F-statistic):**	0.00
Time:	11:25:47	**Log-Likelihood:**	-2443.9
No. Observations:	719	**AIC:**	4902.
Df Residuals:	712	**BIC:**	4934.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.9937	1.336	0.744	0.457	-1.630	3.618
Runs_Scored	0.0921	0.004	24.413	0.000	0.085	0.099
At_Bats	-0.0408	0.003	-12.594	0.000	-0.047	-0.034
Hits	0.0171	0.004	3.892	0.000	0.008	0.026
Infield_Put_Outs	0.0445	0.003	13.072	0.000	0.038	0.051
Shutout	1.4586	0.061	23.829	0.000	1.338	1.579
Saves	0.0596	0.088	0.674	0.500	-0.114	0.233

Omnibus:	0.038	**Durbin-Watson:**	1.618
Prob(Omnibus):	0.981	**Jarque-Bera (JB):**	0.004
Skew:	-0.001	**Prob(JB):**	0.998
Kurtosis:	3.011	**Cond. No.**	2.90e+04

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 2.9e+04. This might indicate that there are strong multicollinearity or other numerical problems.

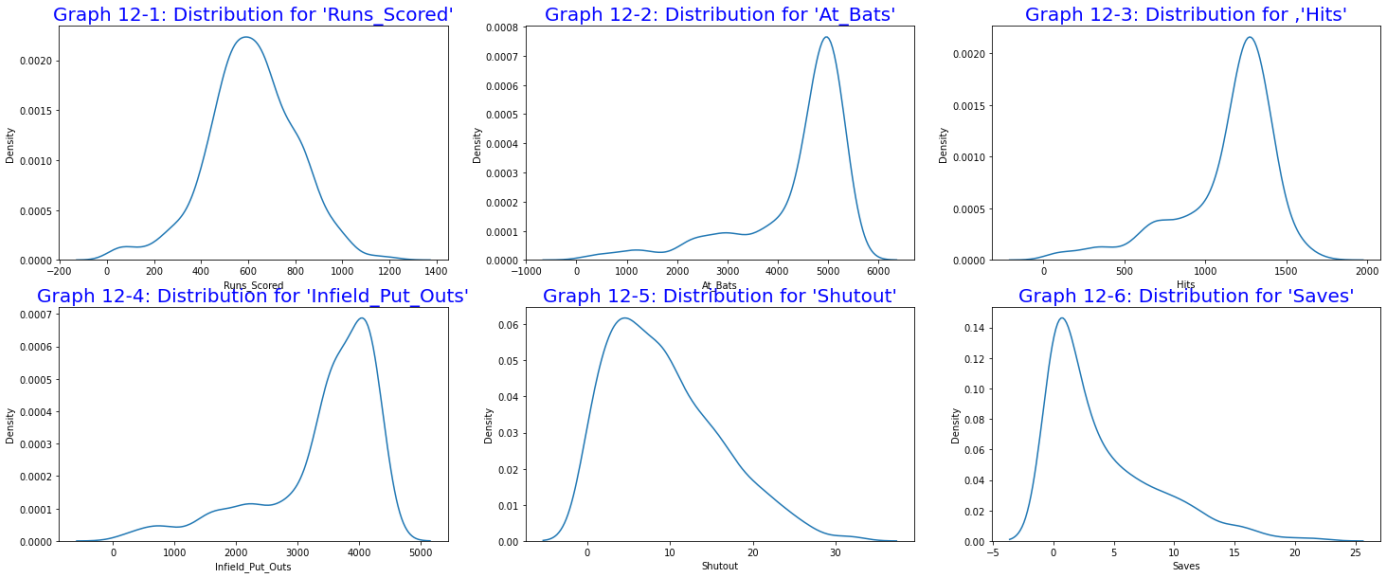
Distribution of independent variables used in our model for Period 1 - before 1920

In [72]:

```
#code for distribution graph

plt.figure(figsize = (25,10)) # figure size
plt.subplot(2,3,1)            # subplotting
sns.kdeplot(dfp1['Runs_Scored'])
plt.title("Graph 12-1: Distribution for 'Runs_Scored'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,2)
sns.kdeplot(dfp1['At_Bats'])
plt.title("Graph 12-2: Distribution for 'At_Bats'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,3)
sns.kdeplot(dfp1['Hits'])
plt.title("Graph 12-3: Distribution for ,'Hits'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,4)
```

```
sns.kdeplot(dfp1['Infield_Put_Outs'])
plt.title("Graph 12-4: Distribution for 'Infield_Put_Outs'", fontsize=20, color = 'Blue'
plt.subplot(2,3,5)
sns.kdeplot(dfp1['Shutout'])
plt.title("Graph 12-5: Distribution for 'Shutout'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,6)
sns.kdeplot(dfp1['Saves'])
plt.title("Graph 12-6: Distribution for 'Saves'", fontsize=20, color = 'blue' )
```

Out[72]:
```
Text(0.5, 1.0, "Graph 12-6: Distribution for 'Saves'")
```



Period 2 - 1920 to 1960

In [73]:
```
dfp2 = file2[(file2.Year >= 1920) & (file2.Year <= 1960)]
```

Defining x and y variables

In [74]:
```
# defining dependent variable y
y_combined = dfp2['Games_Won'].values
y_offensive = dfp2['Games_Won'].values
y_defensive = dfp2['Games_Won'].values
# defining independent variables x
x_combined = dfp2[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']]
x_offensive = dfp2[['Runs_Scored','At_Bats','Hits']].values
x_defensive = dfp2[['Infield_Put_Outs','Shutout','Saves']].values
```

Spliting training set and test set from the data set

In [75]:
```
# Combined varaibles data set
xp2_combined_train,xp2_combined_test,yp2_combined_train,yp2_combined_test = train_test_s
# Offensive varaibles data set
xp2_offensive_train,xp2_offensive_test,yp2_offensive_train,yp2_offensive_test = train_te
# Defensive varaibles data set
xp2_defensive_train,xp2_defensive_test,yp2_defensive_train,yp2_defensive_test = train_te
```

Training the model on the training set

In [76]:
```
# Training our Combined data set model
mlp2_combined = LinearRegression()
```

```
mlp2_combined.fit(xp2_combined_train,yp2_combined_train)
# Training our Offensive data set model
mlp2_offensive = LinearRegression()
mlp2_offensive.fit(xp2_offensive_train,yp2_offensive_train)
# Training our Defensive data set model
mlp2_defensive = LinearRegression()
mlp2_defensive.fit(xp2_defensive_train,yp2_defensive_train)
```

Out[76]:
```
LinearRegression()
```

Predicting the test set results

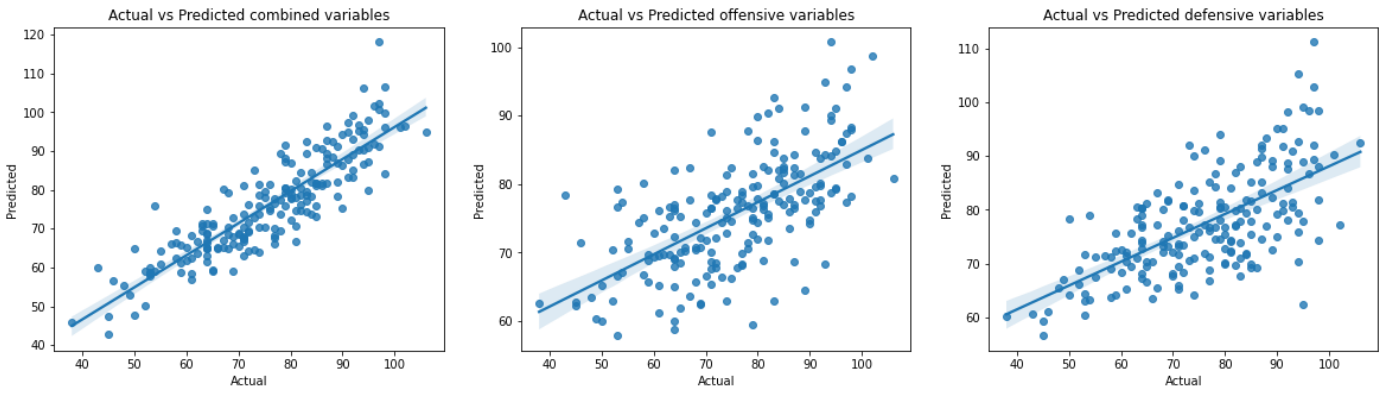
In [77]:
```
yp2_combined_pred = mlp2_combined.predict(xp2_combined_test) # combined data set perdict
yp2_offensive_pred = mlp2_offensive.predict(xp2_offensive_test) # offensive data set per
yp2_defensive_pred = mlp2_defensive.predict(xp2_defensive_test) # defensive data set per
```

Plotting our Multiple linear regression models

In [78]:
```
plt.figure(figsize = (20,5))

plt.subplot(1,3,1)
lp2_combined = sns.regplot(yp2_combined_test,yp2_combined_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted combined variables')

plt.subplot(1,3,2)
lp2_offensive = sns.regplot(yp2_offensive_test,yp2_offensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted offensive variables')

plt.subplot(1,3,3)
lp2_defensive = sns.regplot(yp2_defensive_test,yp2_defensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted defensive variables')
```

```
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[78]:
```
Text(0.5, 1.0, 'Actual vs Predicted defensive variables')
```



Evaluating each model

In [79]:
```
we=r2_score(yp2_combined_test,yp2_combined_pred)
wee=r2_score(yp2_offensive_test,yp2_offensive_pred)
weee=r2_score(yp2_defensive_test,yp2_defensive_pred)
print('Combined',we,' Offensive',wee,' Defensive',weee)
```

```
Combined 0.7738651245315898  Offensive 0.4092372047845724  Defensive 0.41131280820428084
```

Based on our regplot and r2 values we can see that model with Combined offensive and defensive independent variable is better.

Statistical summary of our most successful model

In [80]:
```
lm2 = smf.ols(formula="Games_Won ~ Runs_Scored + At_Bats + Hits + Infield_Put_Outs + Shu
lm2.summary()
```

Out[80]:

OLS Regression Results

Dep. Variable:	Games_Won	**R-squared:**	0.784
Model:	OLS	**Adj. R-squared:**	0.782
Method:	Least Squares	**F-statistic:**	393.6
Date:	Wed, 23 Nov 2022	**Prob (F-statistic):**	1.90e-212
Time:	11:25:48	**Log-Likelihood:**	-2171.7
No. Observations:	656	**AIC:**	4357.
Df Residuals:	649	**BIC:**	4389.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-124.8611	19.057	-6.552	0.000	-162.282	-87.440
Runs_Scored	0.0808	0.005	17.471	0.000	0.072	0.090
At_Bats	-0.0433	0.004	-10.163	0.000	-0.052	-0.035
Hits	0.0277	0.006	4.990	0.000	0.017	0.039
Infield_Put_Outs	0.0768	0.006	13.530	0.000	0.066	0.088
Shutout	1.2773	0.068	18.714	0.000	1.143	1.411
Saves	0.2891	0.042	6.965	0.000	0.208	0.371

Omnibus:	1.351	**Durbin-Watson:**	2.046
Prob(Omnibus):	0.509	**Jarque-Bera (JB):**	1.176
Skew:	-0.073	**Prob(JB):**	0.556
Kurtosis:	3.147	**Cond. No.**	5.07e+05

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 5.07e+05. This might indicate that there are strong multicollinearity or other numerical problems.

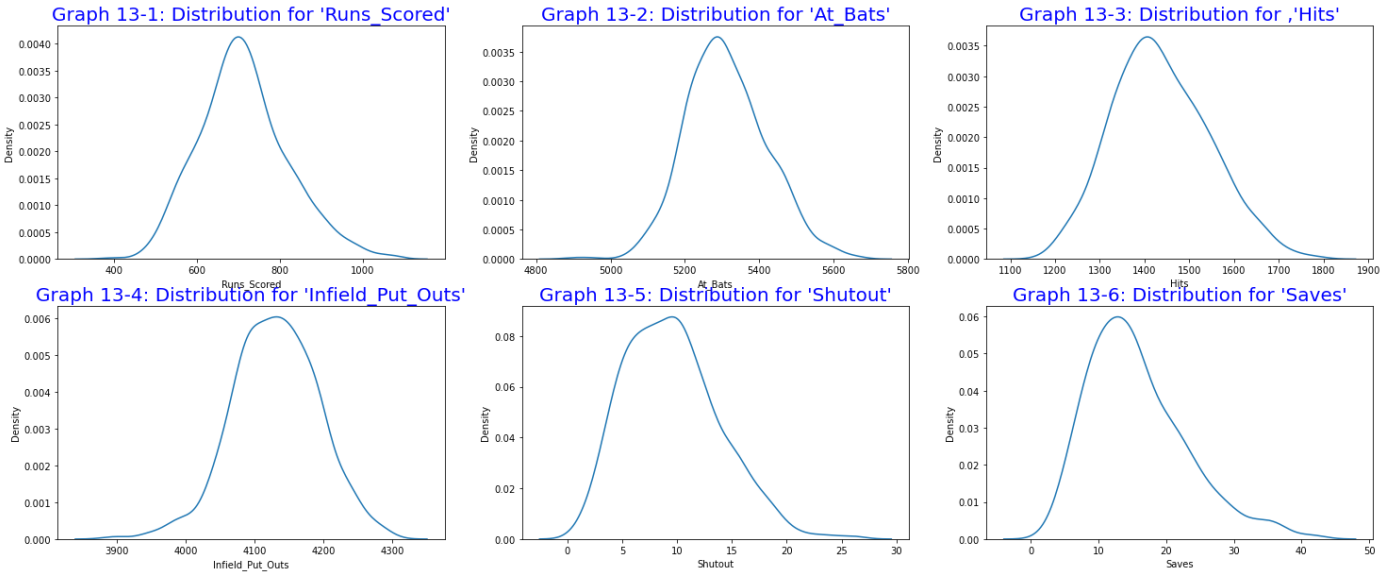
Distribution of independent variables used in our model for Period 2 - 1920 to 1960

In [81]:

```
#code for distribution graph

plt.figure(figsize = (25,10)) # figure size
plt.subplot(2,3,1)            # subplotting
sns.kdeplot(dfp2['Runs_Scored'])
plt.title("Graph 13-1: Distribution for 'Runs_Scored'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,2)
sns.kdeplot(dfp2['At_Bats'])
plt.title("Graph 13-2: Distribution for 'At_Bats'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,3)
sns.kdeplot(dfp2['Hits'])
plt.title("Graph 13-3: Distribution for ,'Hits'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,4)
```

```
sns.kdeplot(dfp2['Infield_Put_Outs'])
plt.title("Graph 13-4: Distribution for 'Infield_Put_Outs'", fontsize=20, color = 'Blue'
plt.subplot(2,3,5)
sns.kdeplot(dfp2['Shutout'])
plt.title("Graph 13-5: Distribution for 'Shutout'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,6)
sns.kdeplot(dfp2['Saves'])
plt.title("Graph 13-6: Distribution for 'Saves'", fontsize=20, color = 'blue' )
```

Out[81]:
```
Text(0.5, 1.0, "Graph 13-6: Distribution for 'Saves'")
```



Period 3 - 1960 to 1990

In [82]:
```
dfp3 = file2[(file2.Year >= 1960) & (file2.Year <= 1990)]
```

Defining x and y variables

In [83]:
```
# defining dependent variable y
y_combined = dfp3['Games_Won'].values
y_offensive = dfp3['Games_Won'].values
y_defensive = dfp3['Games_Won'].values
# defining independent variables x
x_combined = dfp3[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']]
x_offensive = dfp3[['Runs_Scored','At_Bats','Hits']].values
x_defensive = dfp3[['Infield_Put_Outs','Shutout','Saves']].values
```

Spliting training set and test set from the data set

In [84]:
```
# Combined varaibles data set
xp3_combined_train,xp3_combined_test,yp3_combined_train,yp3_combined_test = train_test_s
# Offensive varaibles data set
xp3_offensive_train,xp3_offensive_test,yp3_offensive_train,yp3_offensive_test = train_te
# Defensive varaibles data set
xp3_defensive_train,xp3_defensive_test,yp3_defensive_train,yp3_defensive_test = train_te
```

Training the model on the training set

In [85]:
```
# Training our Combined data set model
mlp3_combined = LinearRegression()
```

```
mlp3_combined.fit(xp3_combined_train,yp3_combined_train)
# Training our Offensive data set model
mlp3_offensive = LinearRegression()
mlp3_offensive.fit(xp3_offensive_train,yp3_offensive_train)
# Training our Defensive data set model
mlp3_defensive = LinearRegression()
mlp3_defensive.fit(xp3_defensive_train,yp3_defensive_train)
```

Out[85]:
```
LinearRegression()
```

Predicting the test set results

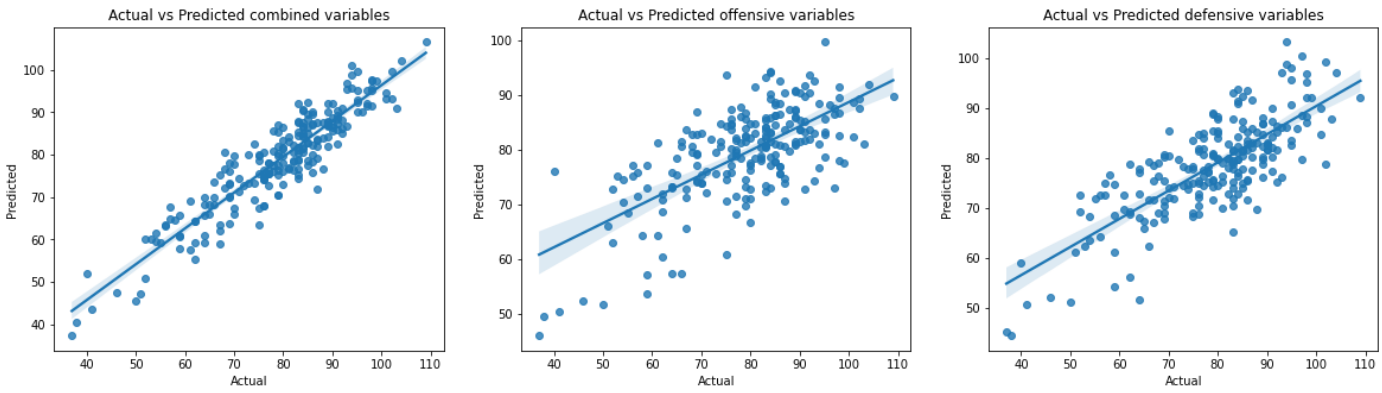
In [86]:
```
yp3_combined_pred = mlp3_combined.predict(xp3_combined_test) # combined data set perdict
yp3_offensive_pred = mlp3_offensive.predict(xp3_offensive_test) # offensive data set per
yp3_defensive_pred = mlp3_defensive.predict(xp3_defensive_test) # defensive data set per
```

Plotting our Multiple linear regression models

In [87]:
```
plt.figure(figsize = (20,5))

plt.subplot(1,3,1)
lp1_combined = sns.regplot(yp3_combined_test,yp3_combined_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted combined variables')

plt.subplot(1,3,2)
lp1_offensive = sns.regplot(yp3_offensive_test,yp3_offensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted offensive variables')

plt.subplot(1,3,3)
lp1_defensive = sns.regplot(yp3_defensive_test,yp3_defensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted defensive variables')
```

```
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[87]:
```
Text(0.5, 1.0, 'Actual vs Predicted defensive variables')
```



Evaluating each model

In [88]:
```
we=r2_score(yp3_combined_test,yp3_combined_pred)
wee=r2_score(yp3_offensive_test,yp3_offensive_pred)
weee=r2_score(yp3_defensive_test,yp3_defensive_pred)
print('Combined',we,' Offensive',wee,' Defensive',weee)
```

```
Combined 0.8556333923079179  Offensive 0.44825917332689313  Defensive 0.5970764733189431
```

Based on our regplot and r2 values we can see that model with Combined offensive and defensive independent variable is better.

Statistical summary of our most successful model

In [89]:
```
lm3 = smf.ols(formula="Games_Won ~ Runs_Scored + At_Bats + Hits + Infield_Put_Outs + Shu
lm3.summary()
```

Out[89]:

OLS Regression Results

Dep. Variable:	Games_Won	**R-squared:**	0.826
Model:	OLS	**Adj. R-squared:**	0.825
Method:	Least Squares	**F-statistic:**	572.6
Date:	Wed, 23 Nov 2022	**Prob (F-statistic):**	9.66e-271
Time:	11:25:50	**Log-Likelihood:**	-2246.9
No. Observations:	730	**AIC:**	4508.
Df Residuals:	723	**BIC:**	4540.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.6892	3.055	-1.535	0.125	-10.688	1.309
Runs_Scored	0.0841	0.004	19.388	0.000	0.076	0.093
At_Bats	-0.0839	0.005	-15.803	0.000	-0.094	-0.073
Hits	0.0590	0.006	9.665	0.000	0.047	0.071
Infield_Put_Outs	0.0888	0.006	16.044	0.000	0.078	0.100
Shutout	1.0253	0.049	20.802	0.000	0.929	1.122
Saves	0.2607	0.023	11.361	0.000	0.216	0.306

Omnibus:	0.144	**Durbin-Watson:**	1.890
Prob(Omnibus):	0.931	**Jarque-Bera (JB):**	0.180
Skew:	-0.033	**Prob(JB):**	0.914
Kurtosis:	2.961	**Cond. No.**	1.11e+05

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.11e+05. This might indicate that there are strong multicollinearity or other numerical problems.

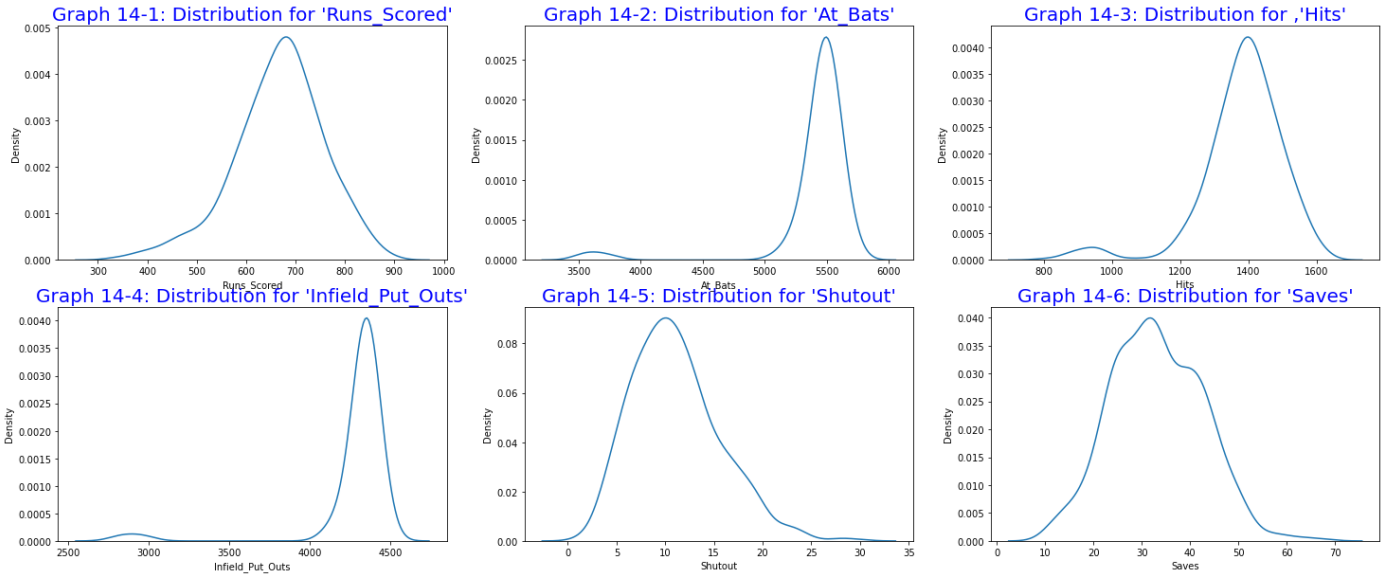
Distribution of independent variables used in our model for Period 3 - 1960 to 1990

In [90]:

```
#code for distribution graph

plt.figure(figsize = (25,10)) # figure size
plt.subplot(2,3,1)            # subplotting
sns.kdeplot(dfp3['Runs_Scored'])
plt.title("Graph 14-1: Distribution for 'Runs_Scored'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,2)
sns.kdeplot(dfp3['At_Bats'])
plt.title("Graph 14-2: Distribution for 'At_Bats'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,3)
sns.kdeplot(dfp3['Hits'])
plt.title("Graph 14-3: Distribution for ,'Hits'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,4)
```

```
sns.kdeplot(dfp3['Infield_Put_Outs'])
plt.title("Graph 14-4: Distribution for 'Infield_Put_Outs'", fontsize=20, color = 'Blue'
plt.subplot(2,3,5)
sns.kdeplot(dfp3['Shutout'])
plt.title("Graph 14-5: Distribution for 'Shutout'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,6)
sns.kdeplot(dfp3['Saves'])
plt.title("Graph 14-6: Distribution for 'Saves'", fontsize=20, color = 'blue' )
```

Out[90]:
```
Text(0.5, 1.0, "Graph 14-6: Distribution for 'Saves'")
```



Period 4 - 1990 to 2010

In [91]:
```
dfp4 = file2[(file2.Year >= 1990) & (file2.Year <= 2010 )]
```

Defining x and y variables

In [92]:
```
# defining dependent variable y
y_combined = dfp4['Games_Won'].values
y_offensive = dfp4['Games_Won'].values
y_defensive = dfp4['Games_Won'].values
# defining independent variables x
x_combined = dfp4[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']]
x_offensive = dfp4[['Runs_Scored','At_Bats','Hits']].values
x_defensive = dfp4[['Infield_Put_Outs','Shutout','Saves']].values
```

Spliting training set and test set from the data set

In [93]:
```
# Combined varaibles data set
xp4_combined_train,xp4_combined_test,yp4_combined_train,yp4_combined_test = train_test_s
# Offensive varaibles data set
xp4_offensive_train,xp4_offensive_test,yp4_offensive_train,yp4_offensive_test = train_te
# Defensive varaibles data set
xp4_defensive_train,xp4_defensive_test,yp4_defensive_train,yp4_defensive_test = train_te
```

Training the model on the training set

In [94]:
```
# Training our Combined data set model
mlp4_combined = LinearRegression()
```

```
mlp4_combined.fit(xp4_combined_train,yp4_combined_train)
# Training our Offensive data set model
mlp4_offensive = LinearRegression()
mlp4_offensive.fit(xp4_offensive_train,yp4_offensive_train)
# Training our Defensive data set model
mlp4_defensive = LinearRegression()
mlp4_defensive.fit(xp4_defensive_train,yp4_defensive_train)
```

Out[94]:
```
LinearRegression()
```

Predicting the test set results

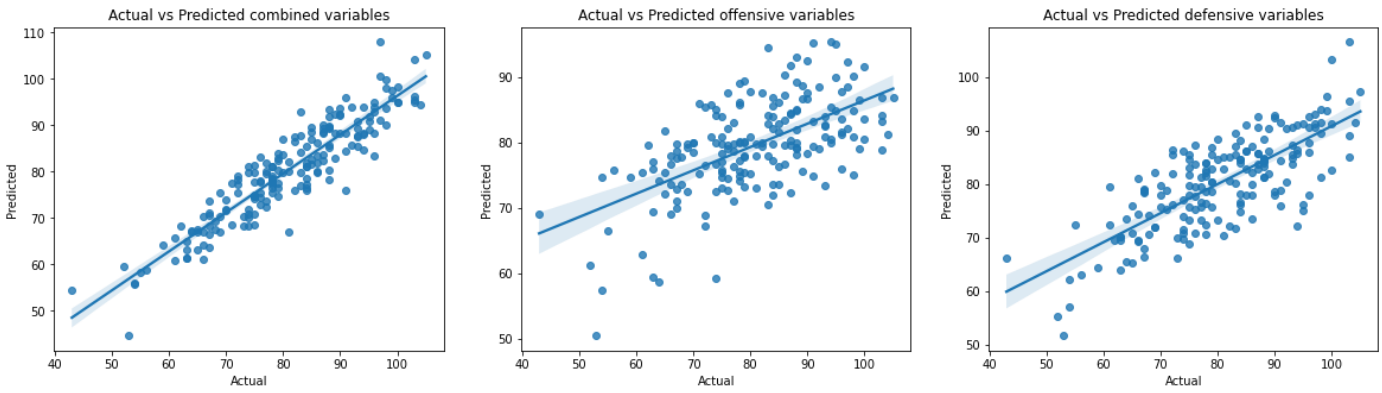
In [95]:
```
yp4_combined_pred = mlp4_combined.predict(xp4_combined_test) # combined data set perdict
yp4_offensive_pred = mlp4_offensive.predict(xp4_offensive_test) # offensive data set per
yp4_defensive_pred = mlp4_defensive.predict(xp4_defensive_test) # defensive data set per
```

Plotting our Multiple linear regression models

In [96]:
```
plt.figure(figsize = (20,5))

plt.subplot(1,3,1)
lp1_combined = sns.regplot(yp4_combined_test,yp4_combined_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted combined variables')

plt.subplot(1,3,2)
lp1_offensive = sns.regplot(yp4_offensive_test,yp4_offensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted offensive variables')

plt.subplot(1,3,3)
lp1_defensive = sns.regplot(yp4_defensive_test,yp4_defensive_pred)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted defensive variables')
```

```
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/Users/farazahmed/opt/anaconda3/lib/python3.9/site-packages/seaborn/_decorators.py:36: F
utureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[96]:
```
Text(0.5, 1.0, 'Actual vs Predicted defensive variables')
```



Evaluating each model

In [97]:
```
we=r2_score(yp3_combined_test,yp3_combined_pred)
wee=r2_score(yp3_offensive_test,yp3_offensive_pred)
weee=r2_score(yp3_defensive_test,yp3_defensive_pred)
print('Combined',we,' Offensive',wee,' Defensive',weee)
```

```
Combined 0.8556333923079179  Offensive 0.44825917332689313  Defensive 0.5970764733189431
```

Based on our regplot and r2 values we can see that model with Combined offensive and defensive independent variable is better.

Statistical summary of our most successful model

In [98]:
```
lm4 = smf.ols(formula="Games_Won ~ Runs_Scored + At_Bats + Hits + Infield_Put_Outs + Shu
lm4.summary()
```

Out[98]:

OLS Regression Results

Dep. Variable:	Games_Won	**R-squared:**	0.870
Model:	OLS	**Adj. R-squared:**	0.869
Method:	Least Squares	**F-statistic:**	670.8
Date:	Wed, 23 Nov 2022	**Prob (F-statistic):**	1.57e-262
Time:	11:25:52	**Log-Likelihood:**	-1766.8
No. Observations:	608	**AIC:**	3548.
Df Residuals:	601	**BIC:**	3578.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.3056	2.806	0.822	0.412	-3.205	7.816
Runs_Scored	0.0730	0.004	19.737	0.000	0.066	0.080
At_Bats	-0.0871	0.005	-16.853	0.000	-0.097	-0.077
Hits	0.0628	0.006	10.235	0.000	0.051	0.075
Infield_Put_Outs	0.0892	0.005	16.423	0.000	0.079	0.100
Shutout	0.8698	0.056	15.528	0.000	0.760	0.980
Saves	0.4881	0.030	16.345	0.000	0.429	0.547

Omnibus:	2.304	**Durbin-Watson:**	1.855
Prob(Omnibus):	0.316	**Jarque-Bera (JB):**	2.295
Skew:	0.150	**Prob(JB):**	0.317
Kurtosis:	2.971	**Cond. No.**	1.11e+05

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The condition number is large, 1.11e+05. This might indicate that there are strong multicollinearity or other numerical problems.

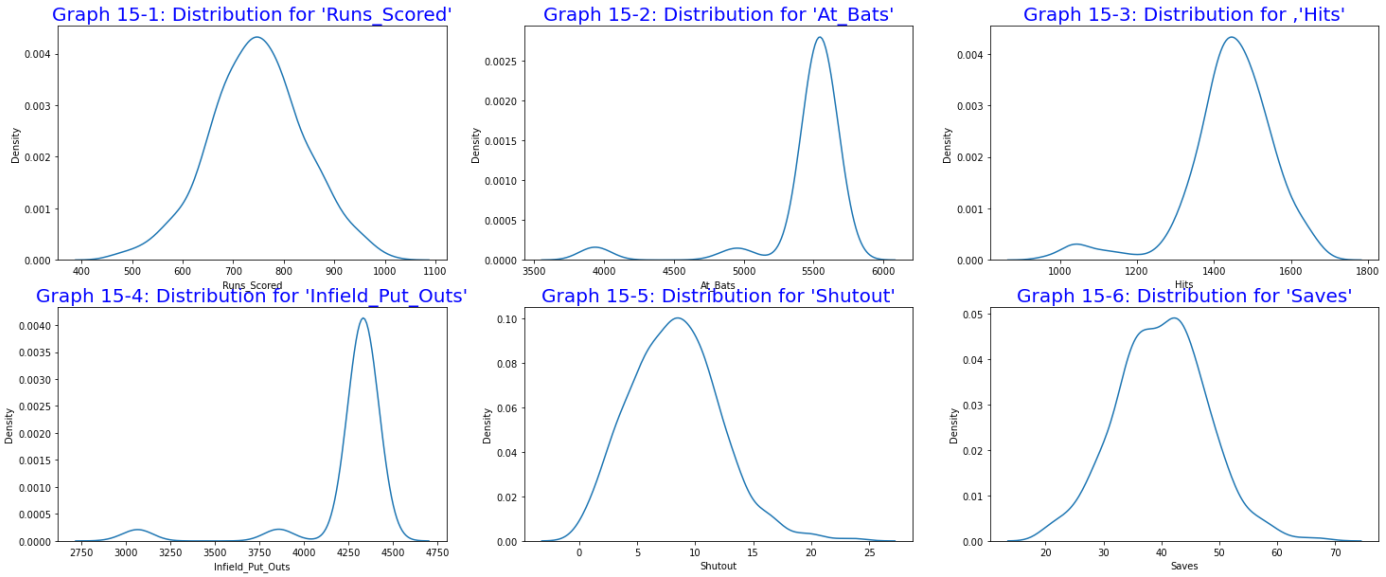
Distribution of independent variables used in our model for Period 4 - 1990 to 2010

In [99]:

```
#code for distribution graph

plt.figure(figsize = (25,10)) # figure size
plt.subplot(2,3,1)            # subplotting
sns.kdeplot(dfp4['Runs_Scored'])
plt.title("Graph 15-1: Distribution for 'Runs_Scored'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,2)
sns.kdeplot(dfp4['At_Bats'])
plt.title("Graph 15-2: Distribution for 'At_Bats'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,3)
sns.kdeplot(dfp4['Hits'])
plt.title("Graph 15-3: Distribution for ,'Hits'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,4)
```

```
sns.kdeplot(dfp4['Infield_Put_Outs'])
plt.title("Graph 15-4: Distribution for 'Infield_Put_Outs'", fontsize=20, color = 'Blue'
plt.subplot(2,3,5)
sns.kdeplot(dfp4['Shutout'])
plt.title("Graph 15-5: Distribution for 'Shutout'", fontsize=20, color = 'Blue' )
plt.subplot(2,3,6)
sns.kdeplot(dfp4['Saves'])
plt.title("Graph 15-6: Distribution for 'Saves'", fontsize=20, color = 'blue' )
```

Out[99]:
```
Text(0.5, 1.0, "Graph 15-6: Distribution for 'Saves'")
```



6. Forecasting games won by New York Yankees and the Toronto Blue Jays using values for the independent variables for 2012 and 2015.

6.1 New York Yankees

In [100…
```
df_NYY = file2[((file2.Year >= 2012) & (file2.Year <= 2015)) & ((file2.Team_Name == 'New
df_NYY
```

Out[100]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Games_Won	Games_Lost	Lea
2702	2012	AL	NYA	NYY	E	1	162	95	67	
2732	2013	AL	NYA	NYY	E	4	162	85	77	
2762	2014	AL	NYA	NYY	E	2	162	84	78	
2781	2015	AL	NYA	NYY	E	2	162	87	75	

4 rows × 40 columns

In [101…

```
# defining dependent variable y
y_NYY = df_NYY['Games_Won'].values
# defining independent variables x
x_NYY = df_NYY[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']].va
```

In [102…

```
NYY_predict= mlp4_combined.predict(x_NYY)
NYY_predict
```

Out[102]:

```
array([91.30385457, 78.28455986, 75.58680612, 78.17888923])
```

In [103…

```
plt.figure(figsize = (10,10))
plt.plot(df_NYY.Year, y_NYY, linestyle="-", marker="o", label="Actual")
plt.plot(df_NYY.Year, NYY_predict, linestyle="-", marker="o", label="Model Prediction")
plt.legend()
plt.show()
```

6.2 Toronto Blue Jays

In [104…
```
df_TOR = file2[((file2.Year >= 2012) & (file2.Year <= 2015)) & ((file2.Team_Name == 'Tor
df_TOR
```

Out[104]:

	Year	League	Team	Franchise	Division	Final_Standing	Games_Played	Games_Won	Games_Lost	Lea
2713	2012	AL	TOR	TOR	E	4	162	73	89	
2743	2013	AL	TOR	TOR	E	5	162	74	88	
2773	2014	AL	TOR	TOR	E	3	162	83	79	
2780	2015	AL	TOR	TOR	E	1	162	93	69	

4 rows × 40 columns

In [105…
```
# defining dependent variable y
y_TOR = df_TOR['Games_Won'].values
# defining independent variables x
x_TOR = df_TOR[['Runs_Scored','At_Bats','Hits','Infield_Put_Outs','Shutout','Saves']].va
```

In [106…
```
TOR_predict= mlp4_combined.predict(x_TOR)
TOR_predict
```

Out[106]:
```
array([71.29813481, 77.02795277, 83.90514479, 91.35929361])
```

In [107…
```
plt.figure(figsize = (10,10))
plt.plot(df_TOR.Year, y_TOR, linestyle="-", marker="o", label="Actual")
plt.plot(df_TOR.Year, TOR_predict, linestyle="-", marker="o", label="Model Prediction")
plt.legend()
```