School of Information Technology & Engineering (SITE)
Programme: MCA
Course: Foundations of Data Science (MAT5010)
Digital Assignment 2
Winter Semester 2022-23

Assignment Title: Univariate Statistics

Submitted to: Dr. Shashikiran V

Submitted by:

Kamran Ansari (22MCA0223)

```
# Kaggle Dataset Link
# https://www.kaggle.com/datasets/vora1011/ipl-2008-to-2021-all-match-dataset


!pip install stemgraphic
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import stemgraphic
import seaborn as sns
import statsmodels.api as sm
!gdown 1j-d55eVcPA7p9_OfRIcjQhdRJSh5NFjW

df = pd.read_csv('IPL_Matches_2008_2022.csv')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting stemgraphic
  Downloading stemgraphic-0.9.1-py3-none-any.whl (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.9/61.9 kB 2.0 MB/s eta 0:00:00
Collecting docopt
  Downloading docopt-0.6.2.tar.gz (25 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (frc
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py) ... done
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13721 sha
  Stored in directory: /root/.cache/pip/wheels/fc/ab/d4/5da2067ac95b36618c629a5f93f80
Successfully built docopt
Installing collected packages: docopt, stemgraphic
Successfully installed docopt-0.6.2 stemgraphic-0.9.1
Downloading...
From: https://drive.google.com/uc?id=1j-d55eVcPA7p9_OfRIcjQhdRJSh5NFjW
To: /content/IPL_Matches_2008_2022.csv
100% 471k/471k [00:00<00:00, 125MB/s]
```

```
print("Data has", df.shape[0], "rows and", df.shape[1], "columns" )
print(df.columns)
```

```
Data has 950 rows and 20 columns
```

✓ 1s completed at 11:34 PM

```
       'Venue', 'TossWinner', 'TossDecision', 'SuperOver', 'WinningTeam',
       'WonBy', 'Margin', 'method', 'Player_of_Match', 'Team1Players',
       'Team2Players', 'Umpire1', 'Umpire2'],
      dtype='object')
```

```
print("First five rows of the dataset are - ")
df.head()
```

```
First five rows of the dataset are -
```

	ID	City	Date	Season	MatchNumber	Team1	Team2	Ver
0	1312200	Ahmedabad	2022-05-29	2022	Final	Rajasthan Royals	Gujarat Titans	Naren M Stadiu Ahmedat
1	1312199	Ahmedabad	2022-05-27	2022	Qualifier 2	Royal Challengers Bangalore	Rajasthan Royals	Naren M Stadiu Ahmedat
2	1312198	Kolkata	2022-05-25	2022	Eliminator	Royal Challengers Bangalore	Lucknow Super Giants	Ec Garde Kolk
3	1312197	Kolkata	2022-05-24	2022	Qualifier 1	Rajasthan Royals	Gujarat Titans	Ec Garde Kolk
4	1304116	Mumbai	2022-05-22	2022	70	Sunrisers Hyderabad	Punjab Kings	Wankhe Stadiu Mum

Categorical Variables

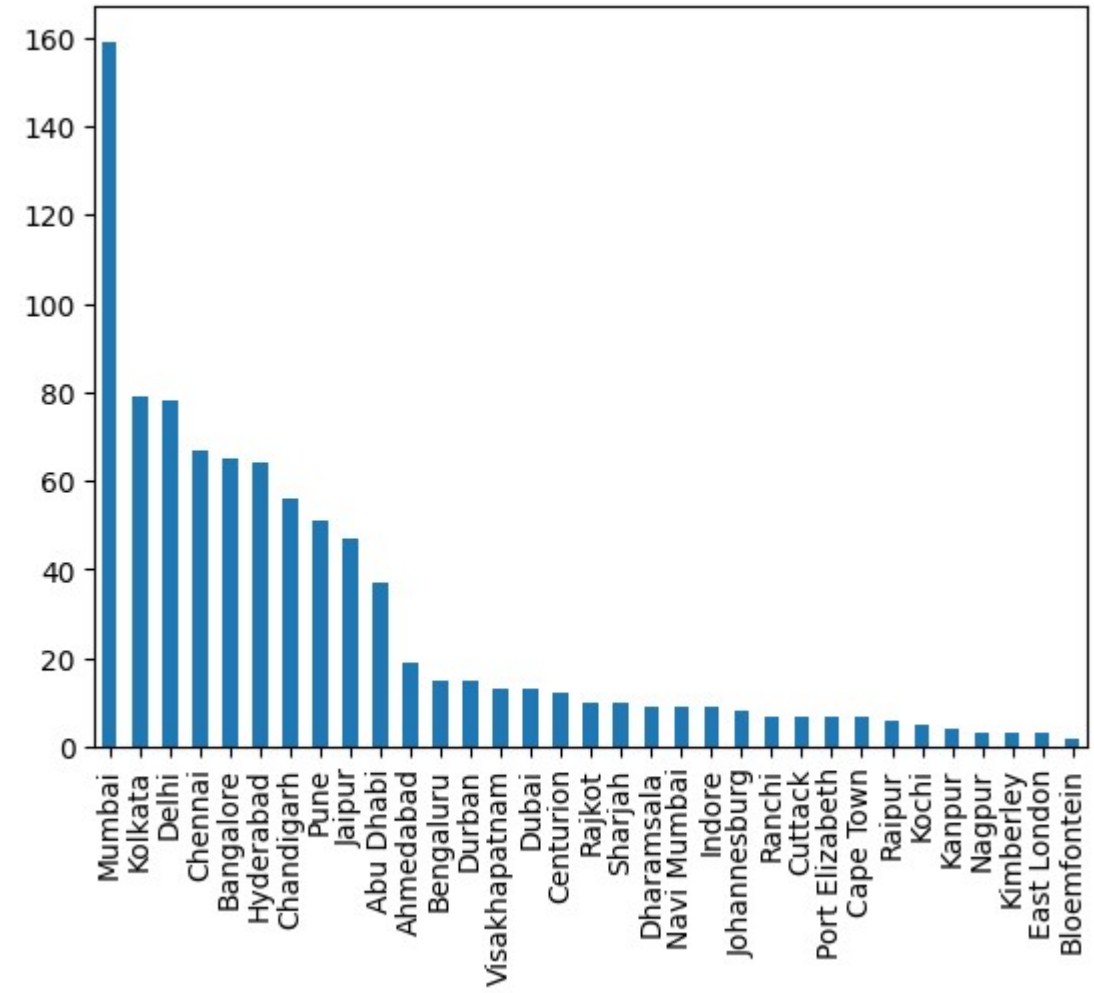
```
params = ['City', 'Season', 'Venue', 'TossWinner', 'TossDecision', 'WinningTeam', 'WonBy']

for param in params:
  print("\n\nStatistics for", param)
  freq = df[param].value_counts()
```

```
freq_df = pd.DataFrame({param:freq.index, 'Frequency':freq.values})

print("Frequency Table")
print(freq_df)

print("\nMode")
print(df[param].mode())

print("\nBar Chart")
df[param].value_counts().plot(kind='bar')
plt.xticks(rotation=90)
plt.show()

print("\nPie Chart")
top = freq_df[:8].copy()
others = pd.DataFrame(data = {
    param : ['Others'],
    'Frequency' : [freq_df[8:]['Frequency'].sum()]
})

piedf = pd.concat([top, others])
piedf.groupby([param]).sum().plot(kind='pie', y='Frequency', figsize=(8,8),  autopct='%1.
plt.show()

print("\nBox Plot")
freq_df.boxplot(column=['Frequency'], grid=False, color='black')
plt.show()
```

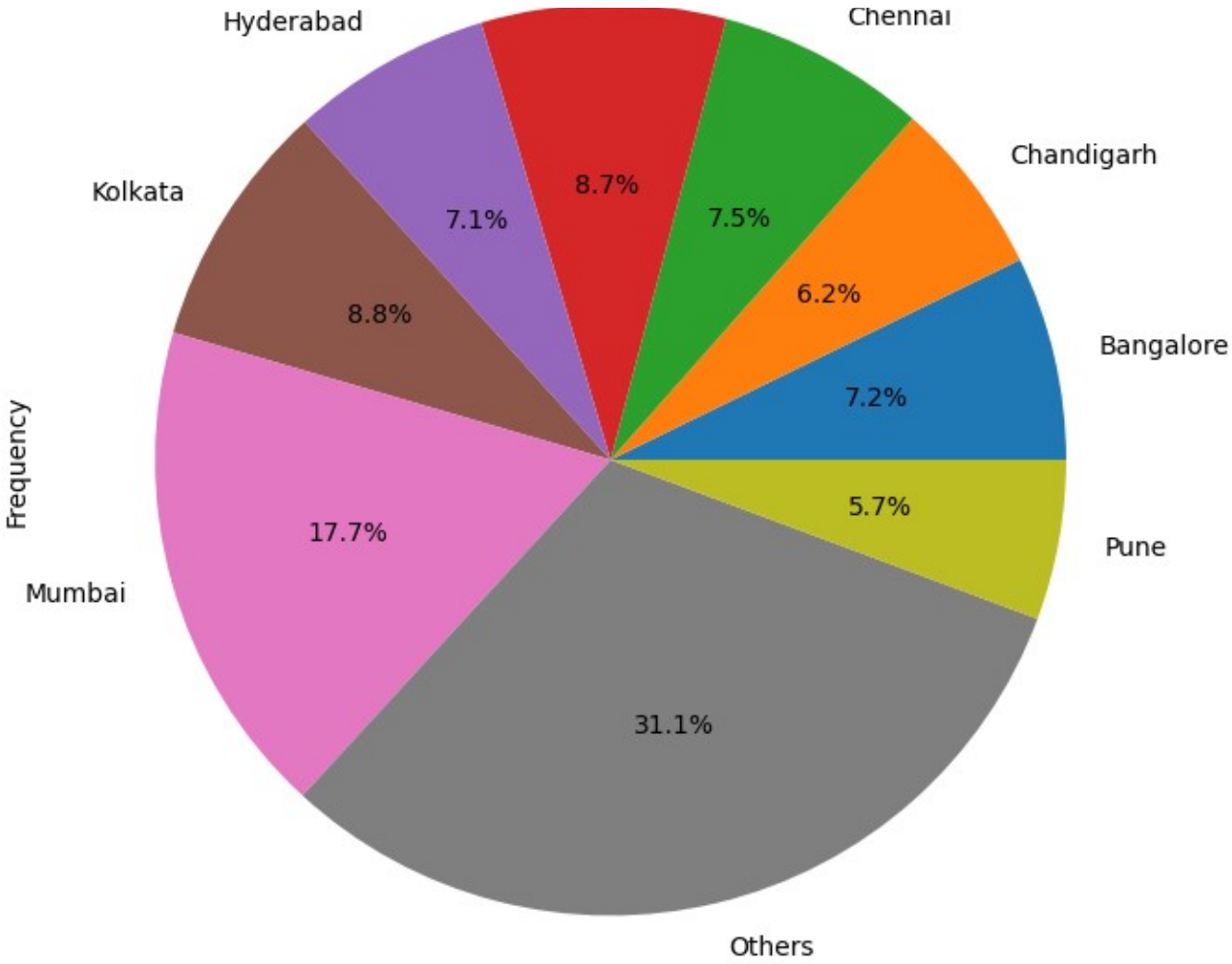
```
Statistics for City
Frequency Table
              City  Frequency
0           Mumbai        159
1          Kolkata         79
2            Delhi         78
3          Chennai         67
4        Bangalore         65
5        Hyderabad         64
6       Chandigarh         56
7             Pune         51
8           Jaipur         47
9        Abu Dhabi         37
10       Ahmedabad         19
11       Bengaluru         15
12          Durban         15
13   Visakhapatnam         13
14           Dubai         13
15       Centurion         12
16          Rajkot         10
17         Sharjah         10
18      Dharamsala          9
19     Navi Mumbai          9
```

```
19     Navi Mumbai          9
20          Indore          9
21    Johannesburg          8
22          Ranchi          7
23         Cuttack          7
24  Port Elizabeth          7
25      Cape Town           7
26          Raipur          6
27           Kochi          5
28          Kanpur          4
29          Nagpur          3
30       Kimberley          3
31     East London          3
32    Bloemfontein          2

Mode
0    Mumbai
Name: City, dtype: object
```

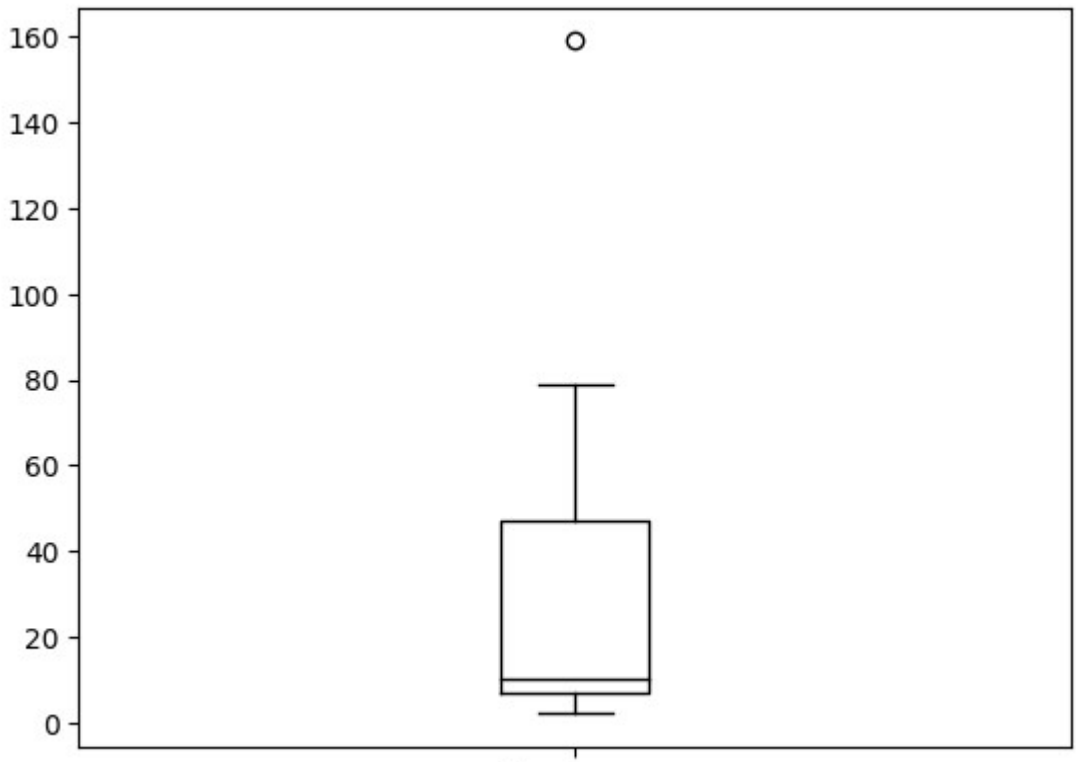
Bar Chart



Pie Chart

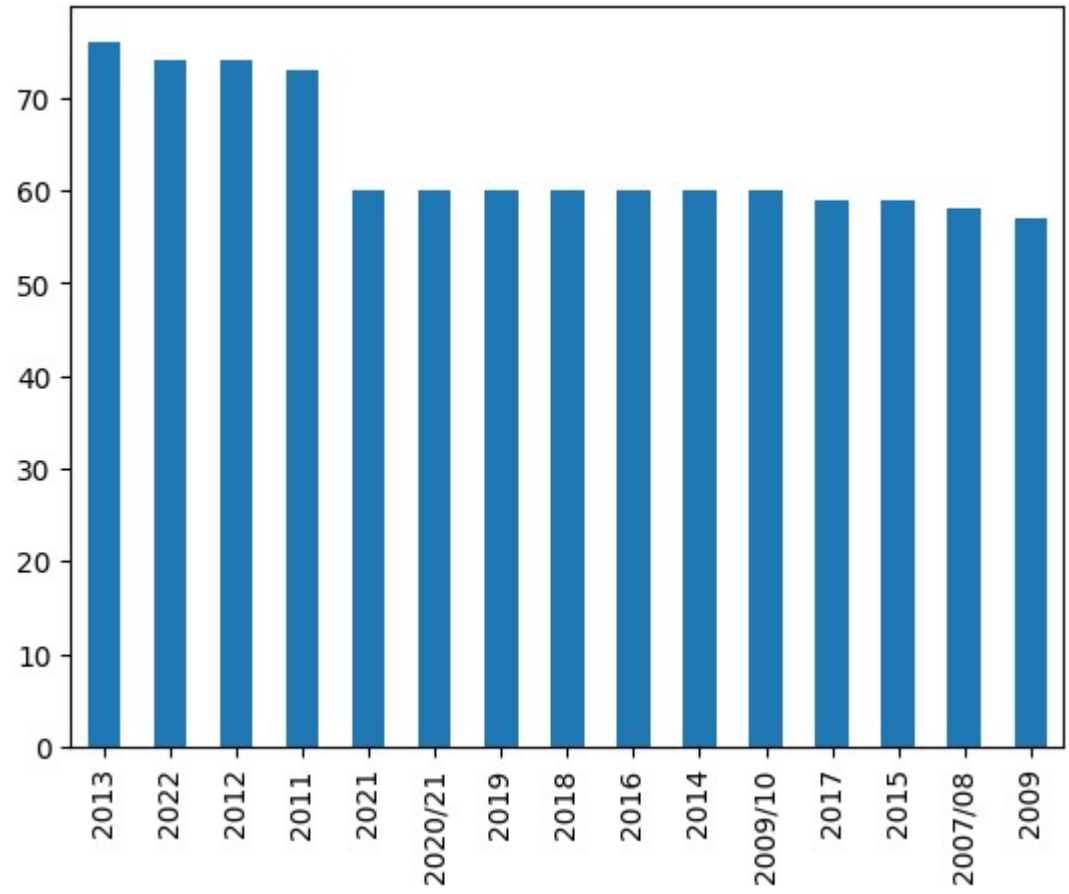

Hyderabad
Chennai
Chandigarh
Bangalore
Pune
Others
Mumbai
Kolkata
Frequency
8.7%
7.5%
6.2%
7.2%
5.7%
31.1%
17.7%
8.8%
7.1%

Box Plot

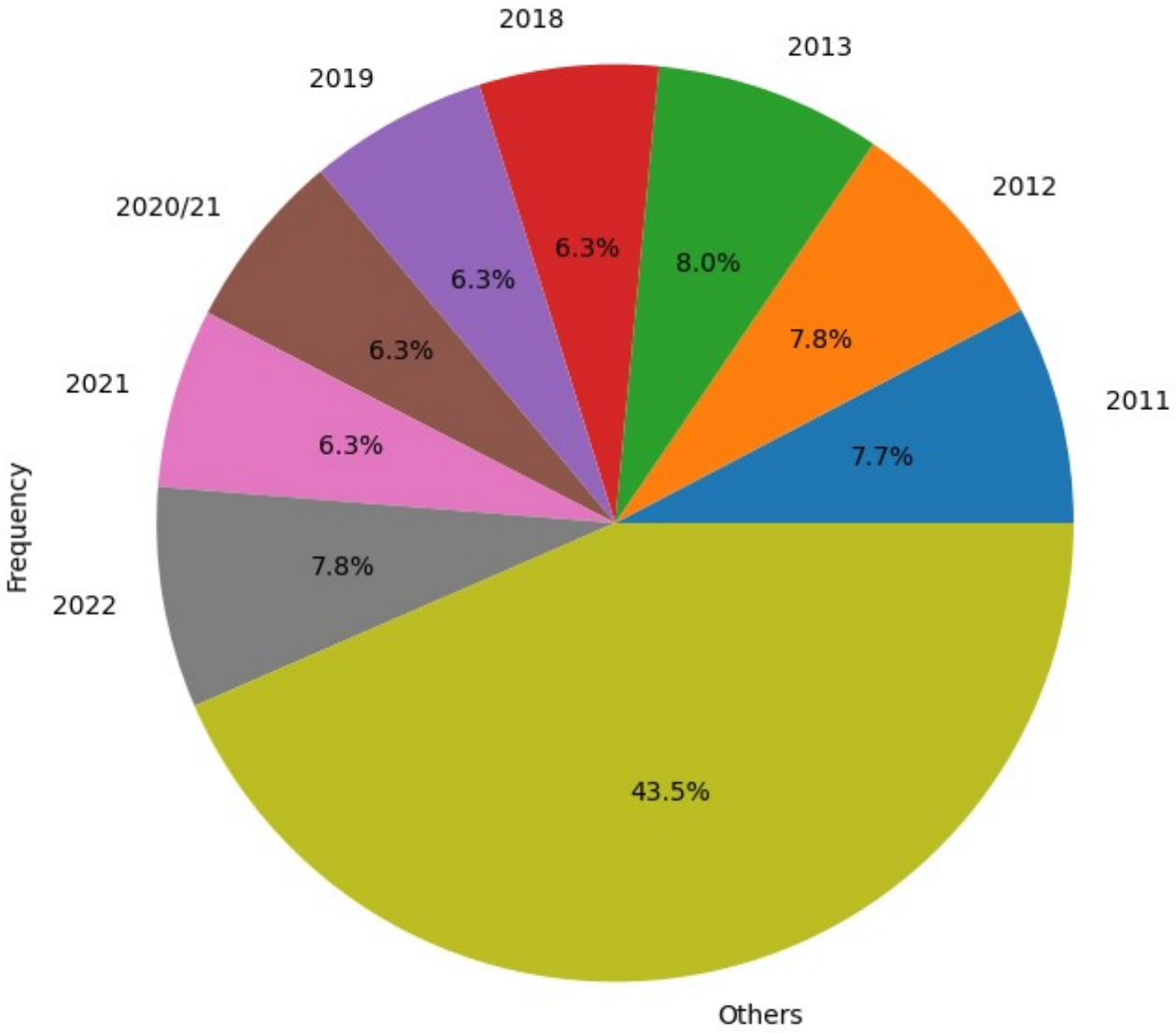

160
140
120
100
80
60
40
20
0

Frequency

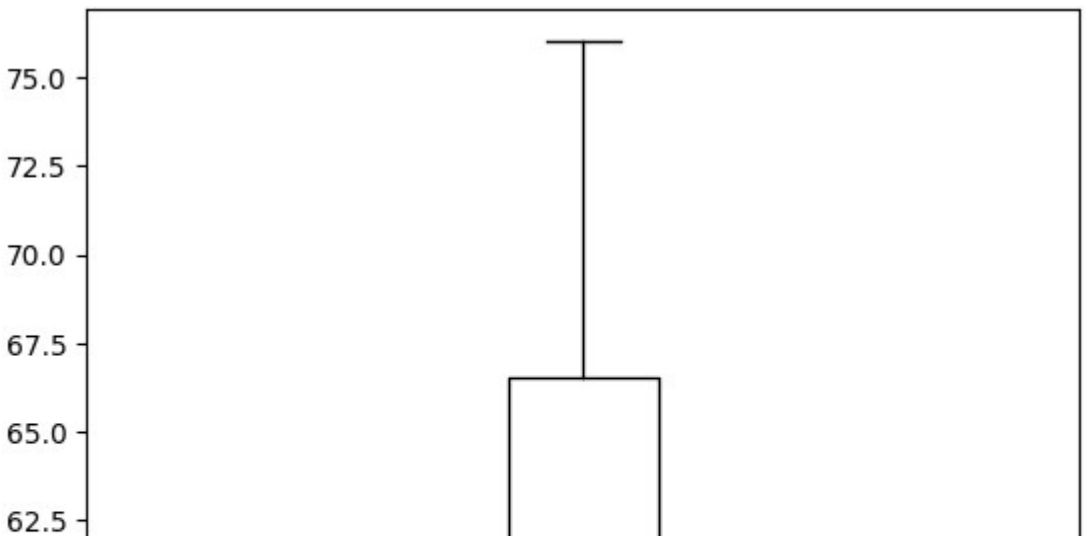
```
Statistics for Season
Frequency Table
     Season  Frequency
0      2013         76
1      2022         74
2      2012         74
3      2011         73
4      2021         60
5   2020/21         60
6      2019         60
7      2018         60
8      2016         60
9      2014         60
10  2009/10         60
11     2017         59
12     2015         59
13  2007/08         58
14     2009         57

Mode
0    2013
Name: Season, dtype: object

Bar Chart
```

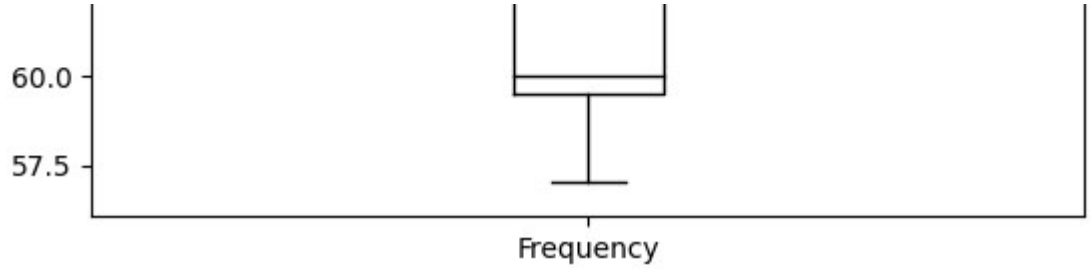


Pie Chart



Box Plot



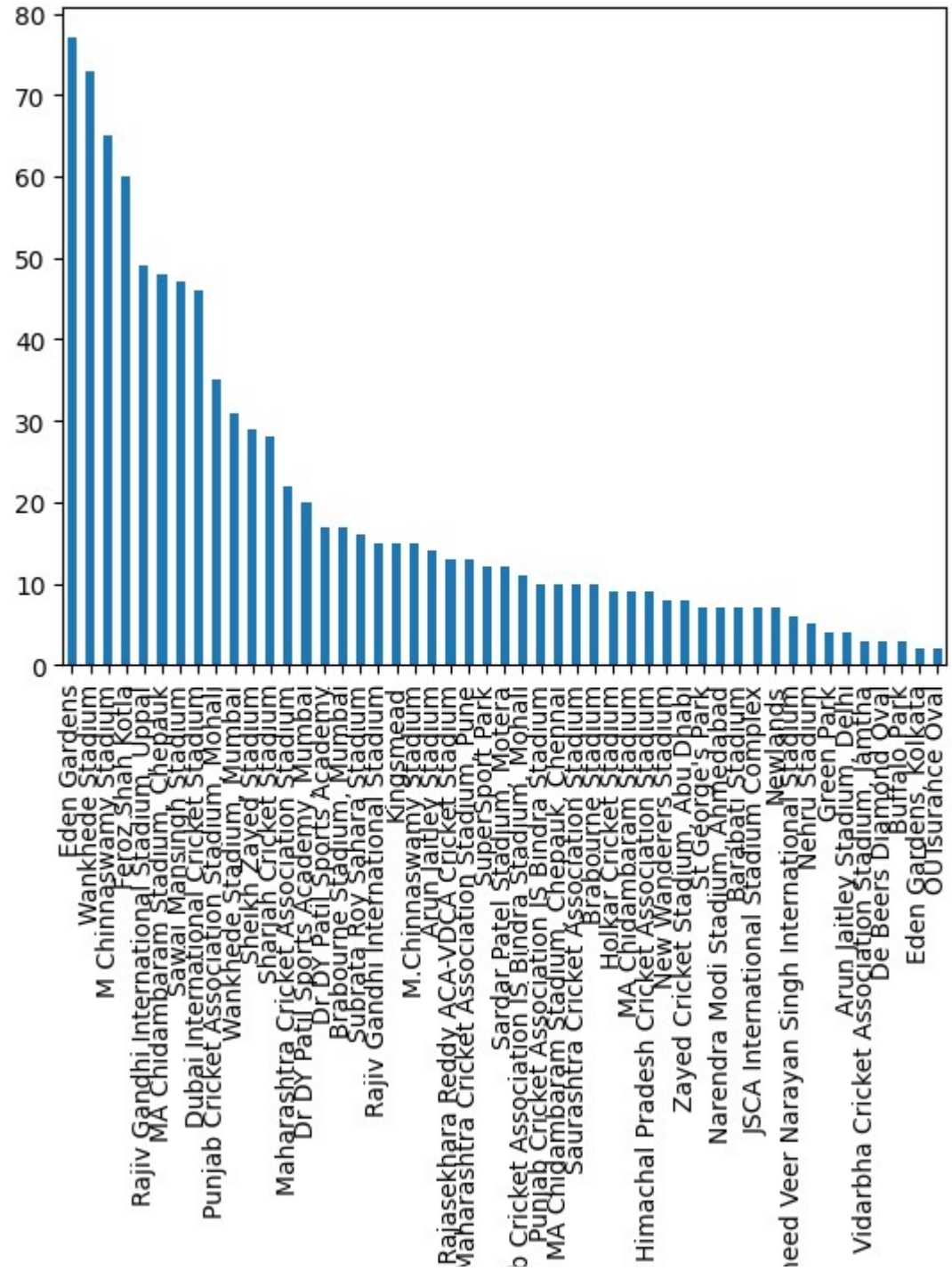


Statistics for Venue
Frequency Table

	Venue	Frequency
0	Eden Gardens	77
1	Wankhede Stadium	73
2	M Chinnaswamy Stadium	65
3	Feroz Shah Kotla	60
4	Rajiv Gandhi International Stadium, Uppal	49
5	MA Chidambaram Stadium, Chepauk	48
6	Sawai Mansingh Stadium	47
7	Dubai International Cricket Stadium	46
8	Punjab Cricket Association Stadium, Mohali	35
9	Wankhede Stadium, Mumbai	31
10	Sheikh Zayed Stadium	29
11	Sharjah Cricket Stadium	28
12	Maharashtra Cricket Association Stadium	22
13	Dr DY Patil Sports Academy, Mumbai	20
14	Dr DY Patil Sports Academy	17
15	Brabourne Stadium, Mumbai	17
16	Subrata Roy Sahara Stadium	16
17	Rajiv Gandhi International Stadium	15
18	Kingsmead	15
19	M.Chinnaswamy Stadium	15
20	Arun Jaitley Stadium	14
21	Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket St...	13
22	Maharashtra Cricket Association Stadium, Pune	13
23	SuperSport Park	12
24	Sardar Patel Stadium, Motera	12
25	Punjab Cricket Association IS Bindra Stadium, ...	11
26	Punjab Cricket Association IS Bindra Stadium	10
27	MA Chidambaram Stadium, Chepauk, Chennai	10
28	Saurashtra Cricket Association Stadium	10
29	Brabourne Stadium	10
30	Holkar Cricket Stadium	9
31	MA Chidambaram Stadium	9
32	Himachal Pradesh Cricket Association Stadium	9
33	New Wanderers Stadium	8
34	Zayed Cricket Stadium, Abu Dhabi	8
35	St George's Park	7
36	Narendra Modi Stadium, Ahmedabad	7
37	Barabati Stadium	7
38	JSCA International Stadium Complex	7
39	Newlands	7
40	Shaheed Veer Narayan Singh International Stadium	6

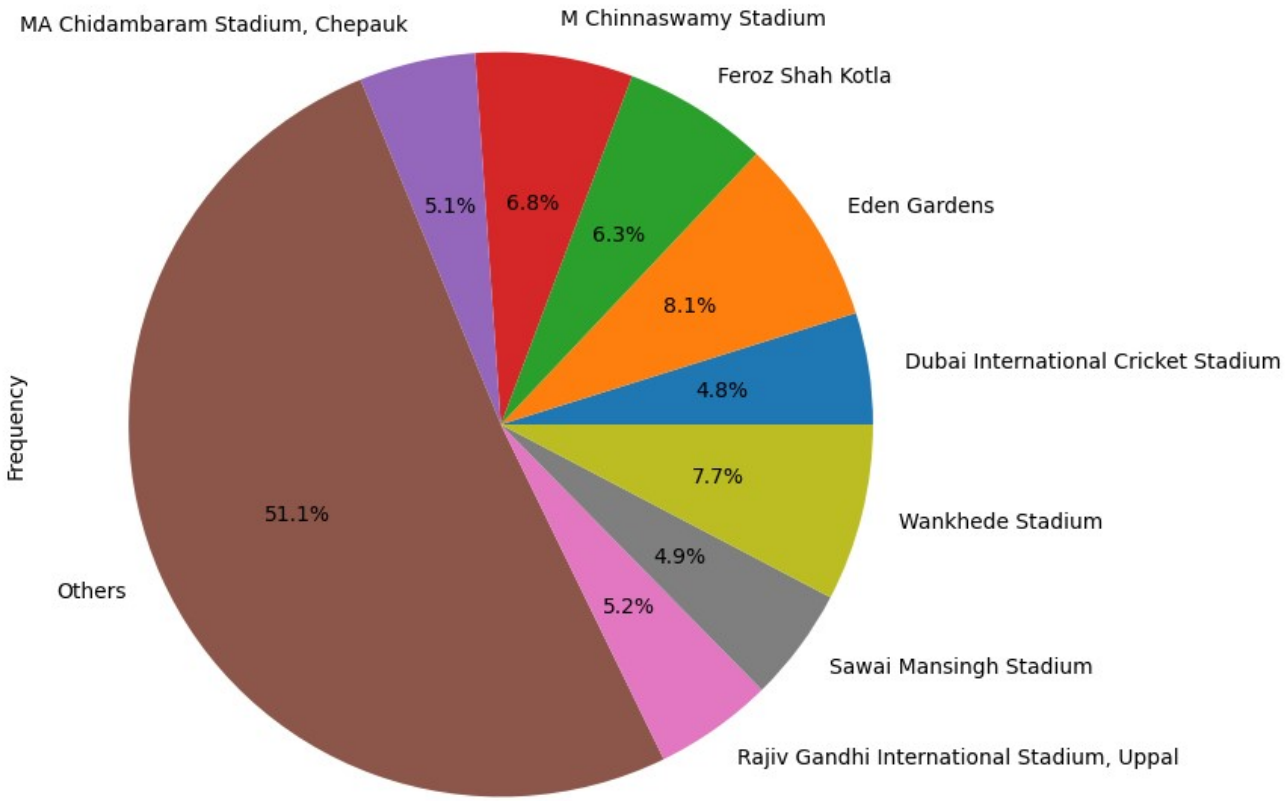
```
41                                    Nehru Stadium          5
42                                      Green Park           4
43                          Arun Jaitley Stadium, Delhi      4
44       Vidarbha Cricket Association Stadium, Jamtha        3
45                               De Beers Diamond Oval       3
46                                      Buffalo Park         3
47                               Eden Gardens, Kolkata       2
48                                     OUTsurance Oval       2

Mode
0    Eden Gardens
Name: Venue, dtype: object
```

Bar Chart

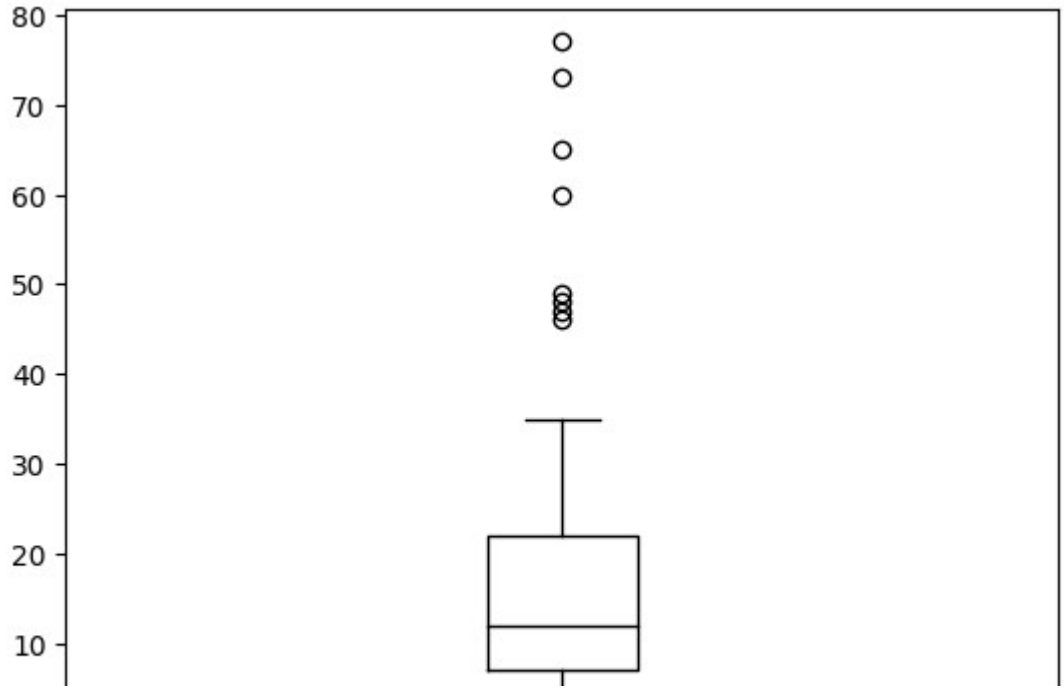


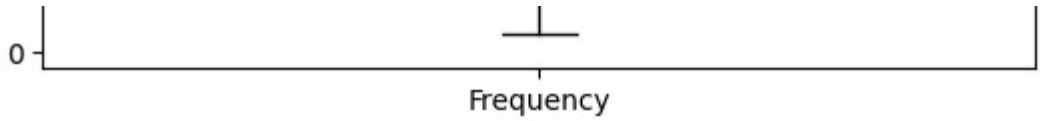
Dr. Y.S. Punjal Shah

Pie Chart



Box Plot



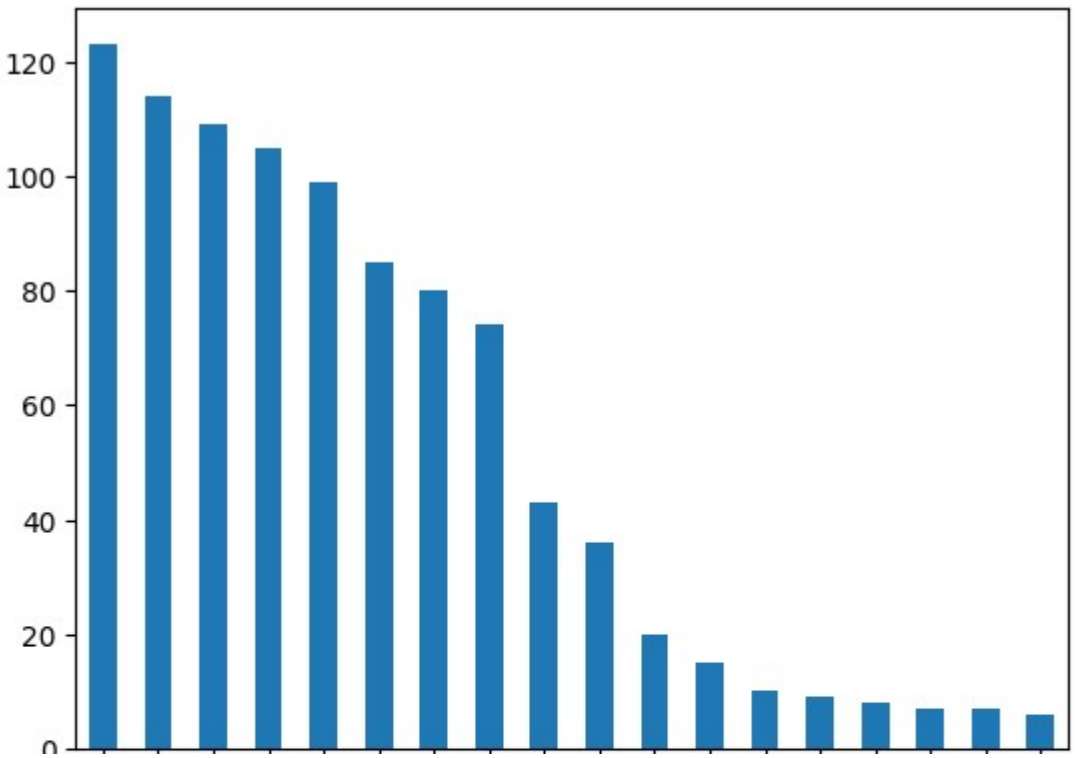


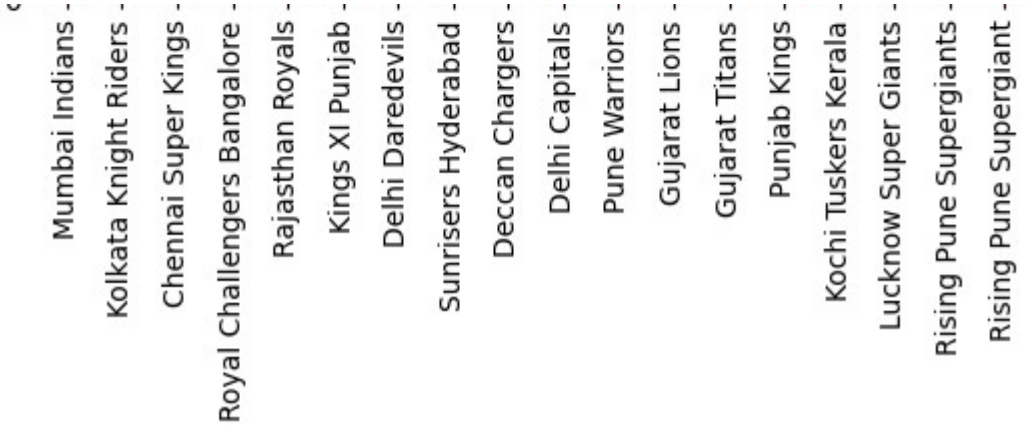
Statistics for TossWinner
Frequency Table

	TossWinner	Frequency
0	Mumbai Indians	123
1	Kolkata Knight Riders	114
2	Chennai Super Kings	109
3	Royal Challengers Bangalore	105
4	Rajasthan Royals	99
5	Kings XI Punjab	85
6	Delhi Daredevils	80
7	Sunrisers Hyderabad	74
8	Deccan Chargers	43
9	Delhi Capitals	36
10	Pune Warriors	20
11	Gujarat Lions	15
12	Gujarat Titans	10
13	Punjab Kings	9
14	Kochi Tuskers Kerala	8
15	Lucknow Super Giants	7
16	Rising Pune Supergiants	7
17	Rising Pune Supergiant	6

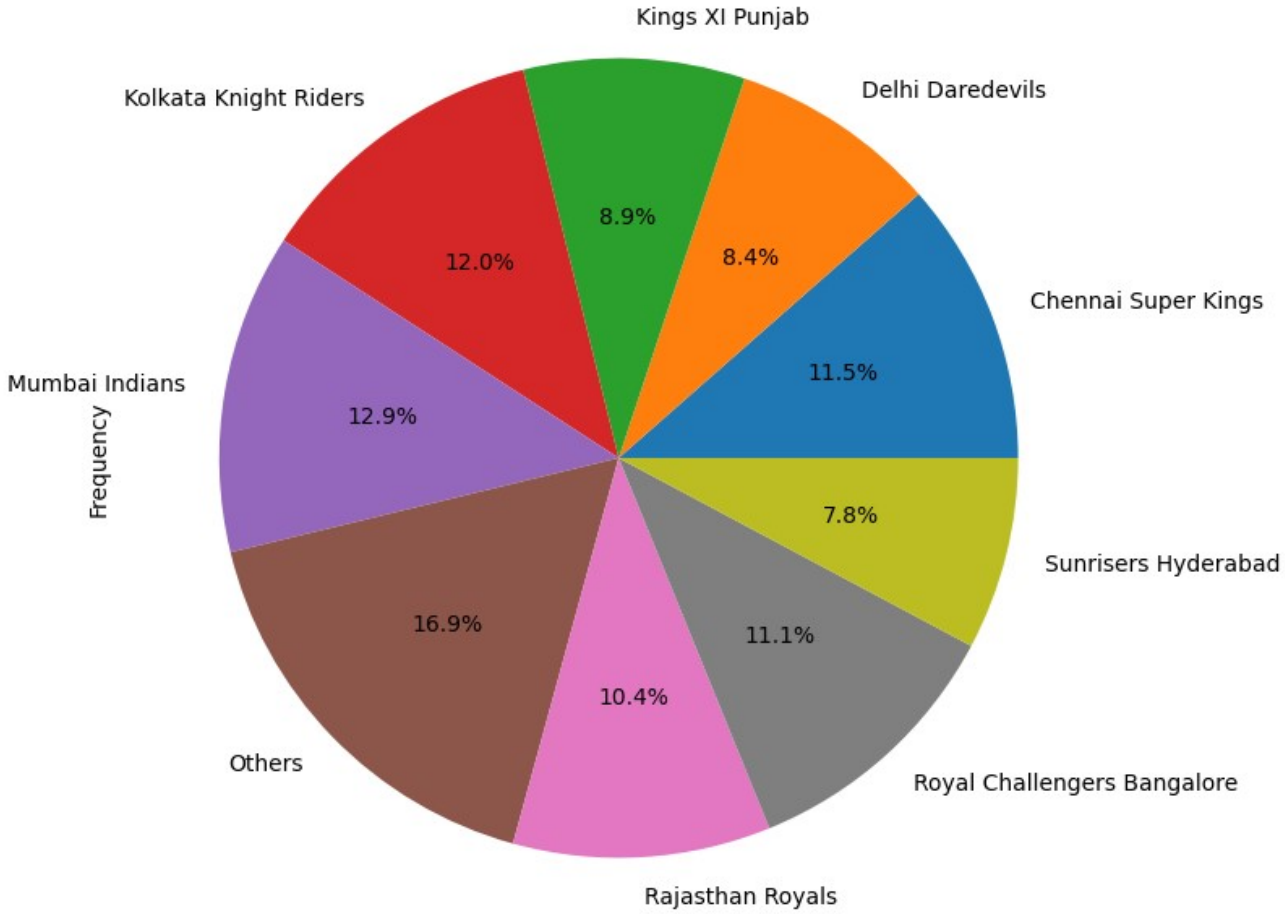
Mode
0 Mumbai Indians
Name: TossWinner, dtype: object

Bar Chart

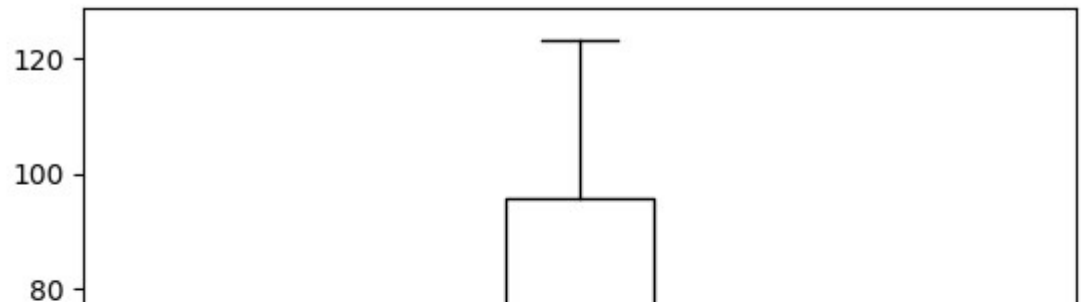


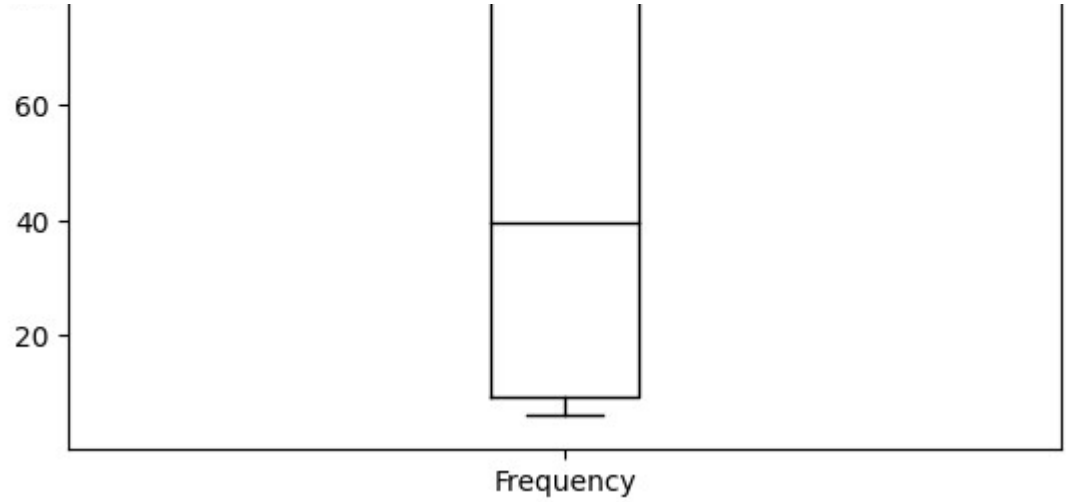


Pie Chart

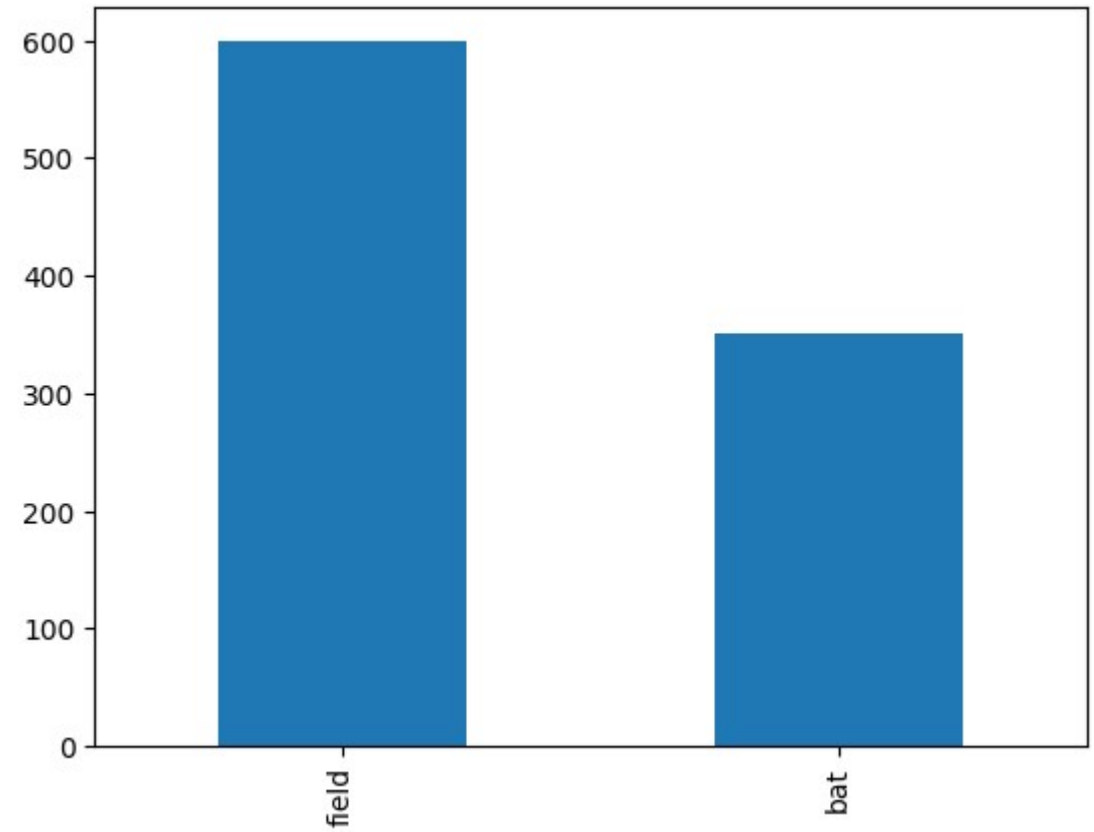


Box Plot

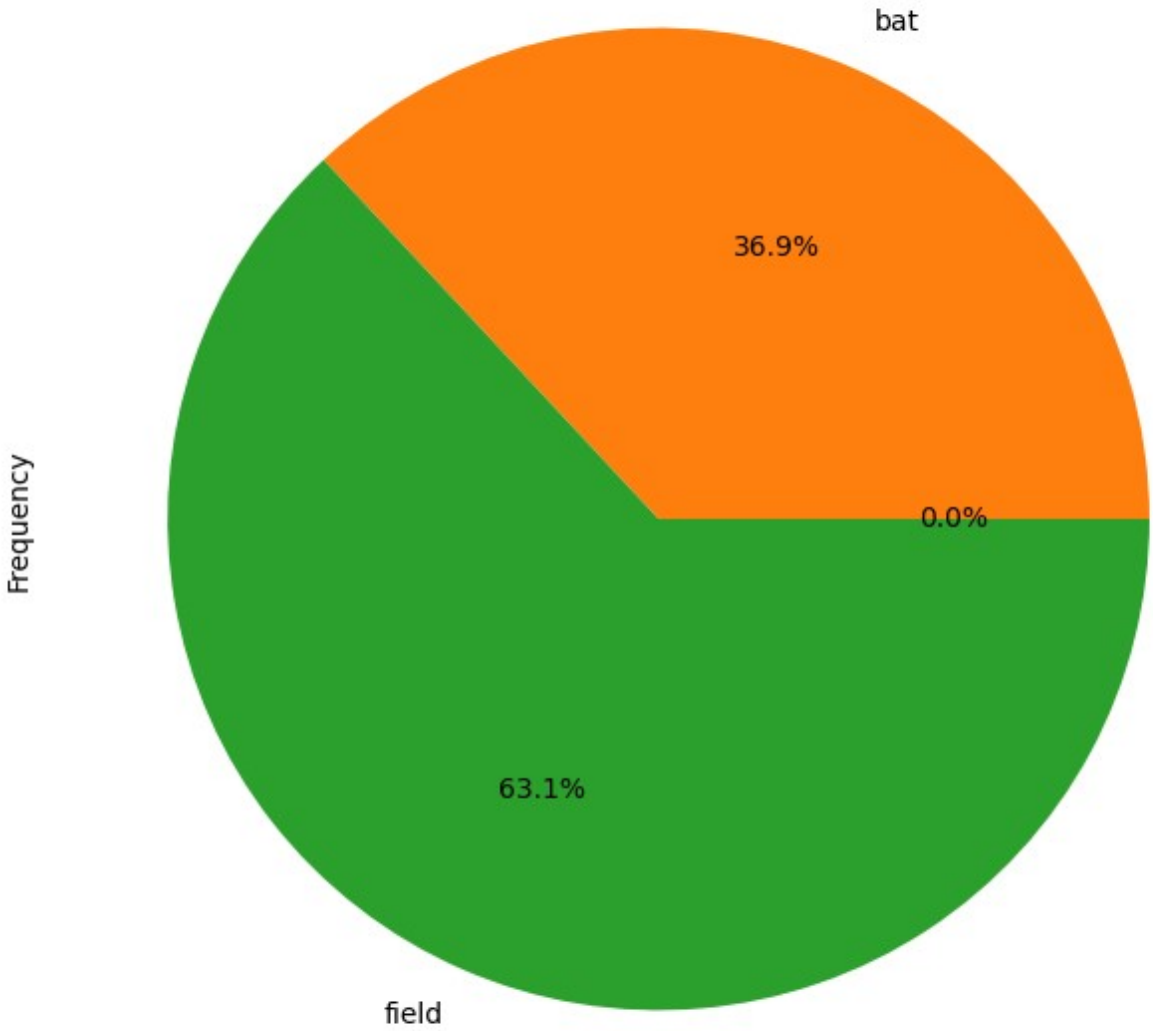




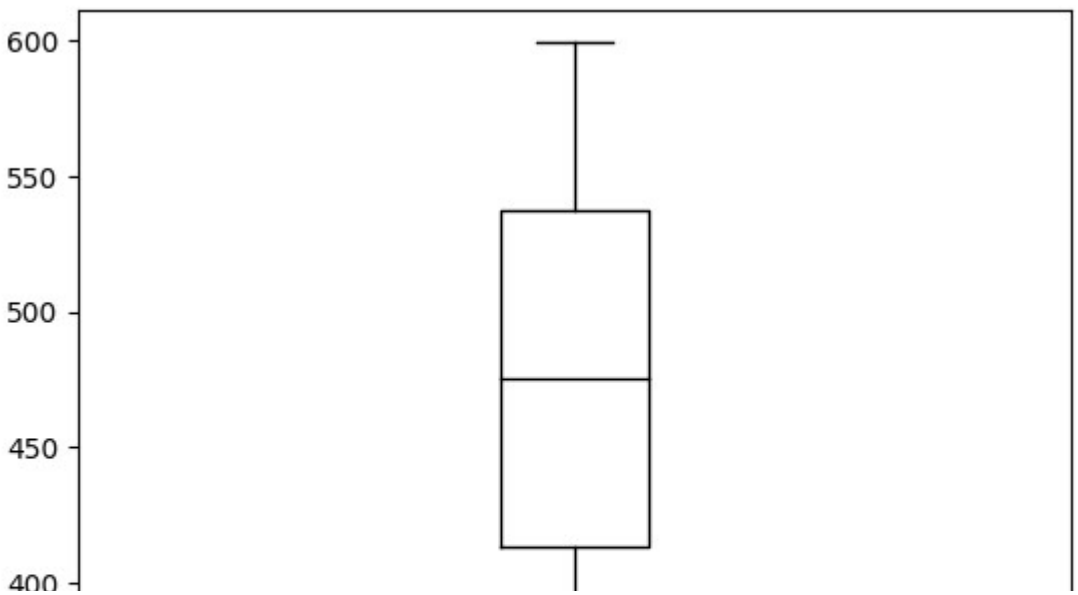

```
Statistics for TossDecision
Frequency Table
  TossDecision  Frequency
0        field        599
1          bat        351

Mode
0    field
Name: TossDecision, dtype: object

Bar Chart
```

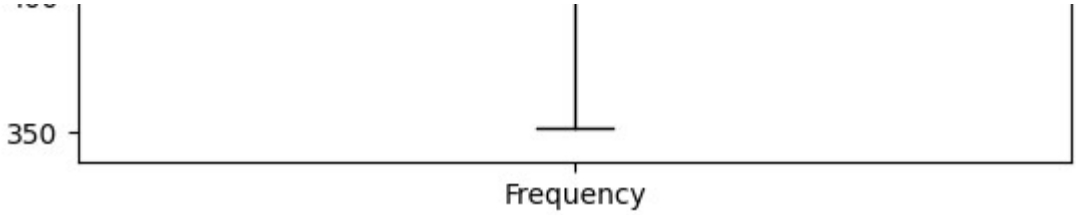



```
Pie Chart
```



Box Plot



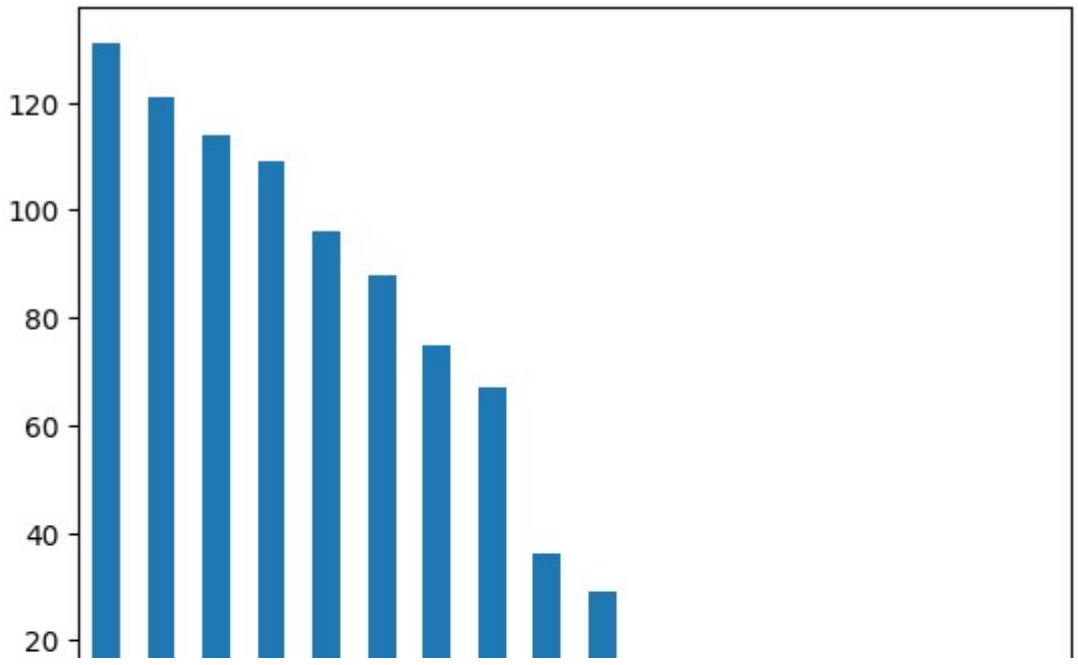


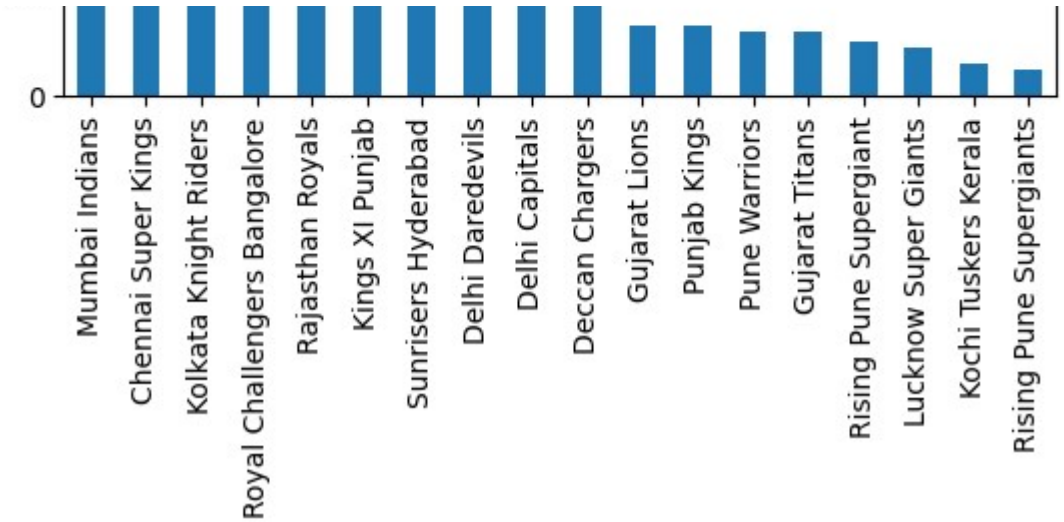
Statistics for WinningTeam
Frequency Table

	WinningTeam	Frequency
0	Mumbai Indians	131
1	Chennai Super Kings	121
2	Kolkata Knight Riders	114
3	Royal Challengers Bangalore	109
4	Rajasthan Royals	96
5	Kings XI Punjab	88
6	Sunrisers Hyderabad	75
7	Delhi Daredevils	67
8	Delhi Capitals	36
9	Deccan Chargers	29
10	Gujarat Lions	13
11	Punjab Kings	13
12	Pune Warriors	12
13	Gujarat Titans	12
14	Rising Pune Supergiant	10
15	Lucknow Super Giants	9
16	Kochi Tuskers Kerala	6
17	Rising Pune Supergiants	5

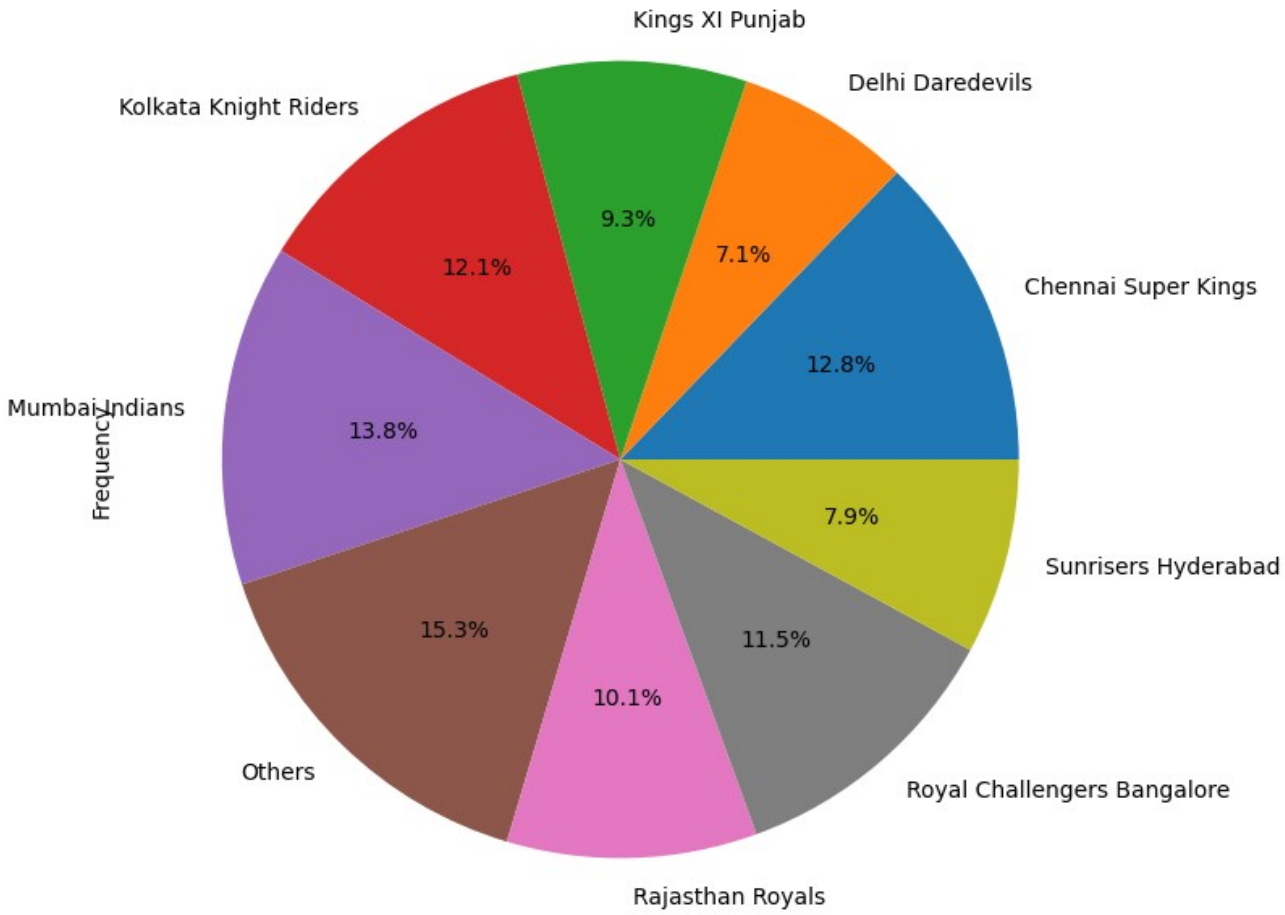
Mode
0 Mumbai Indians
Name: WinningTeam, dtype: object

Bar Chart

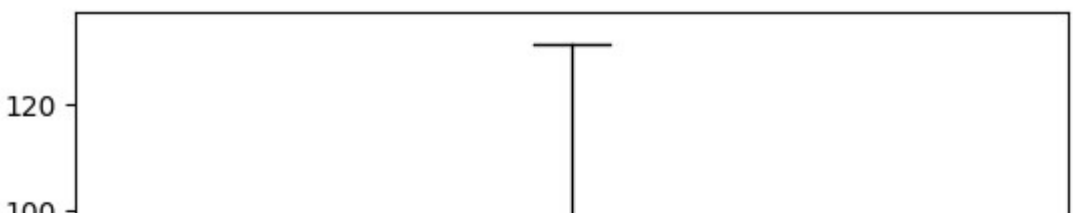


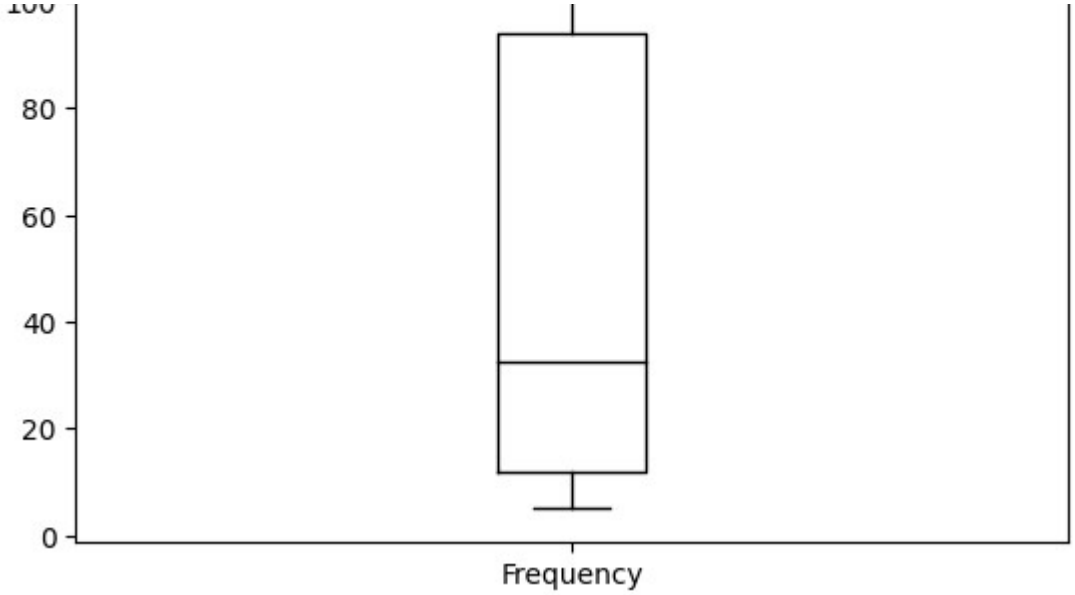


Pie Chart

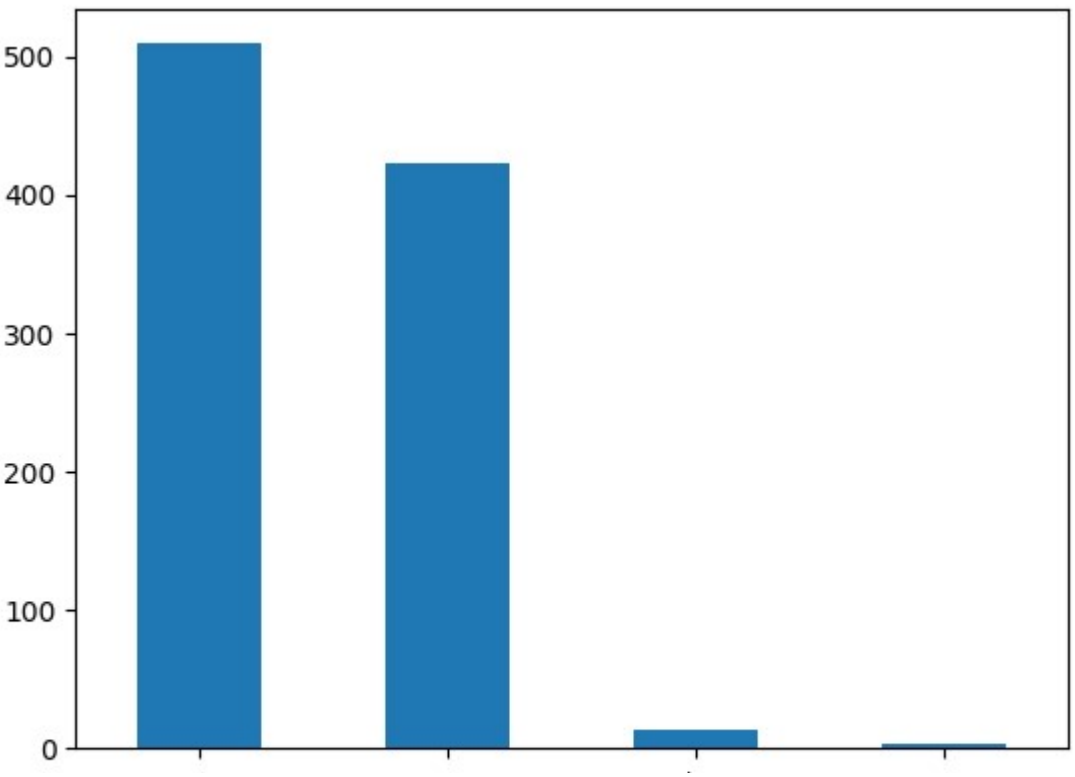


Box Plot

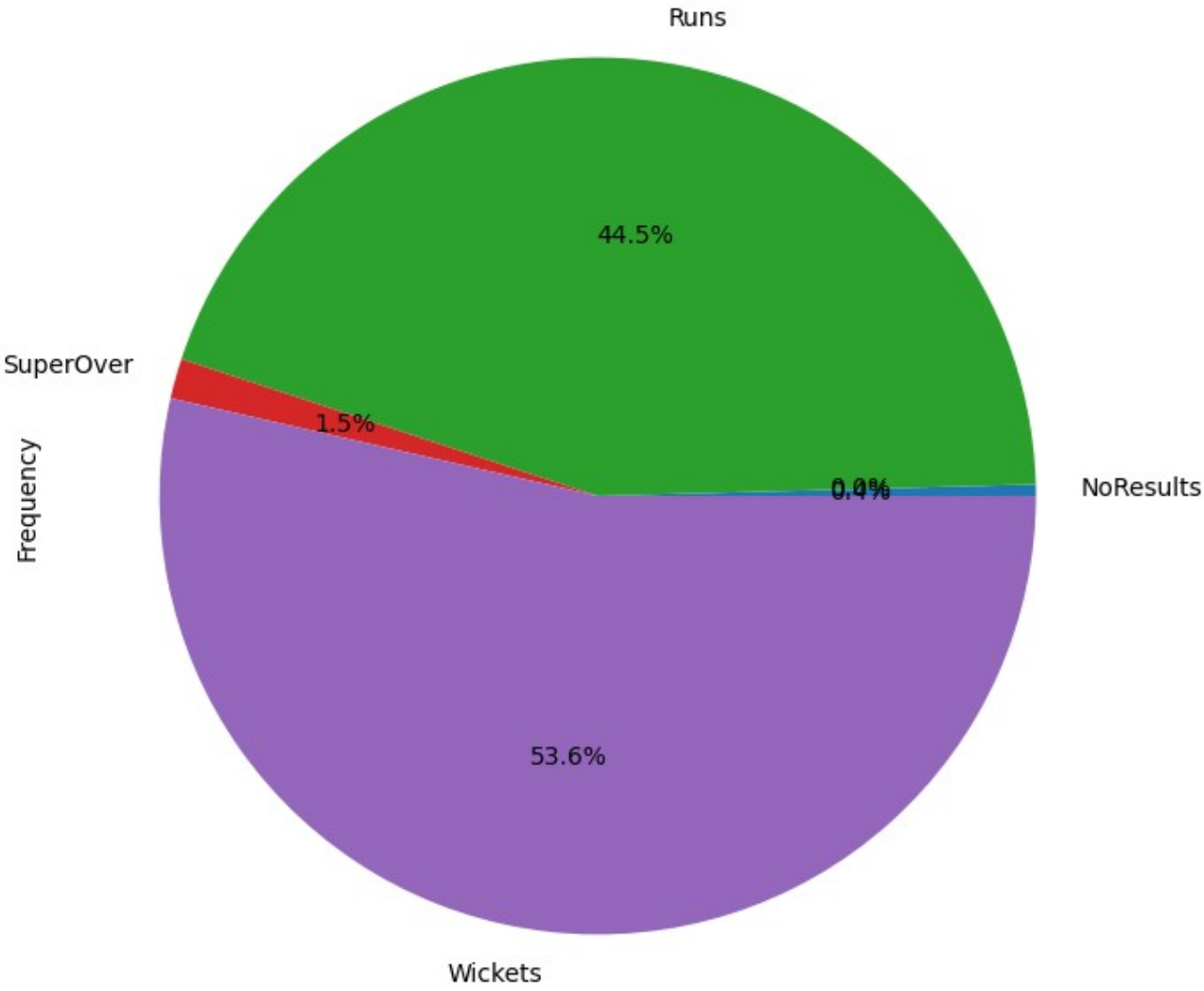




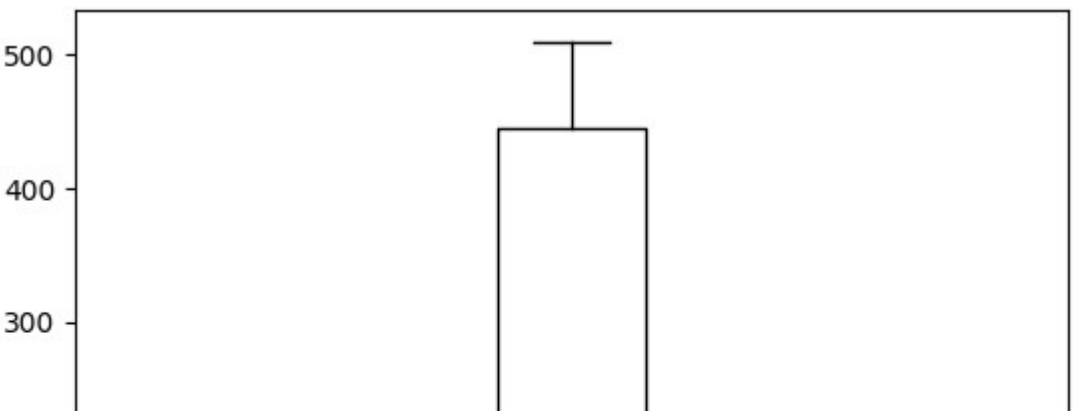

```
Statistics for WonBy
Frequency Table
       WonBy  Frequency
0    Wickets        509
1       Runs        423
2  SuperOver         14
3  NoResults          4

Mode
0    Wickets
Name: WonBy, dtype: object

Bar Chart
```

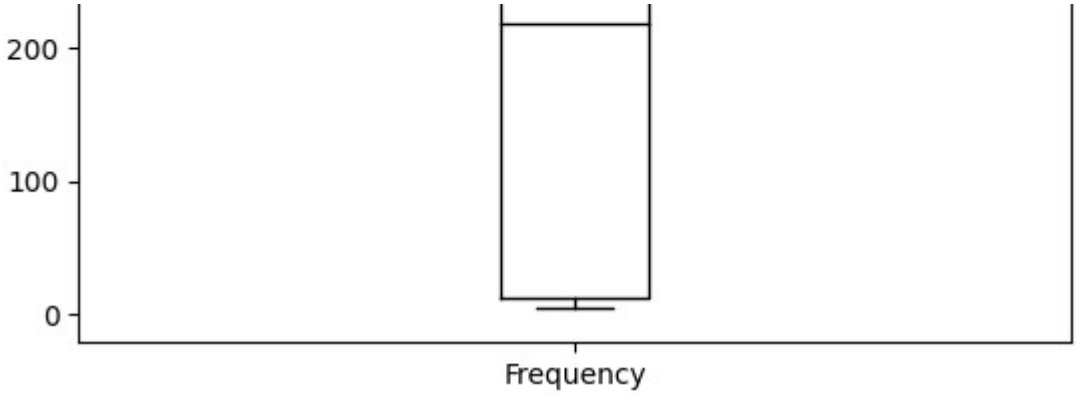


Pie Chart



Box Plot





Quantitative Variables

```
# filter out win margins when match was won defending
# conver dataset into single variable "Margin"

rdf = df.filter(['WonBy', 'Margin'])

# WonBy (NoResult, Wickets, Runs, SuperOver)
rdf = rdf.query('WonBy == "Runs"')

margins = rdf.filter(['Margin'])

print("The size of the cleaned dataset is:", margins.size, "records")
print("")
print("And it has the following data:\n", margins)
```

```
The size of the cleaned dataset is: 423 records

And it has the following data:
      Margin
2      14.0
8       2.0
9       3.0
10     17.0
11     24.0
..      ...
935    13.0
940    66.0
942     6.0
948    33.0
949   140.0

[423 rows x 1 columns]
```

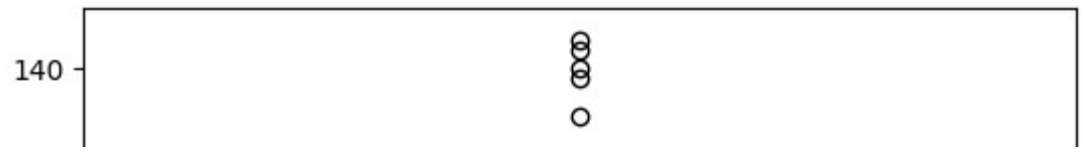
```
# Measures of central tendency
print("Mean win run margin is:", margins['Margin'].mean(), "\n")

# Measures of dispersion
print("Standard Deviation of win run margin is:", margins['Margin'].std(), "\n")

print("Variance of win run margin is:", margins['Margin'].var(), "\n")

min = margins['Margin'].min()
max = margins['Margin'].max()
range = max - min
print("Min: ", min, "Max: ", max, "Range: ", range)
print("")

q3, q1 = np.percentile(margins['Margin'], [75 ,25])
iqr = q3 - q1
print("Q1: ", q1, "Q3: ", q3, "IQR: ", iqr)
print("")

skewness = margins['Margin'].skew()
print("Skewness:", skewness)

if skewness < -1:
    print("The distribution is highly negatively skewed.")
elif skewness >= -1 and skewness < -0.5:
    print("The distribution is moderately negatively skewed.")
elif skewness >= -0.5 and skewness < 0.5:
    print("The distribution is approximately symmetric.")
elif skewness >= 0.5 and skewness < 1:
    print("The distribution is moderately positively skewed.")
else:
    print("The distribution is highly positively skewed.")
print("")
```

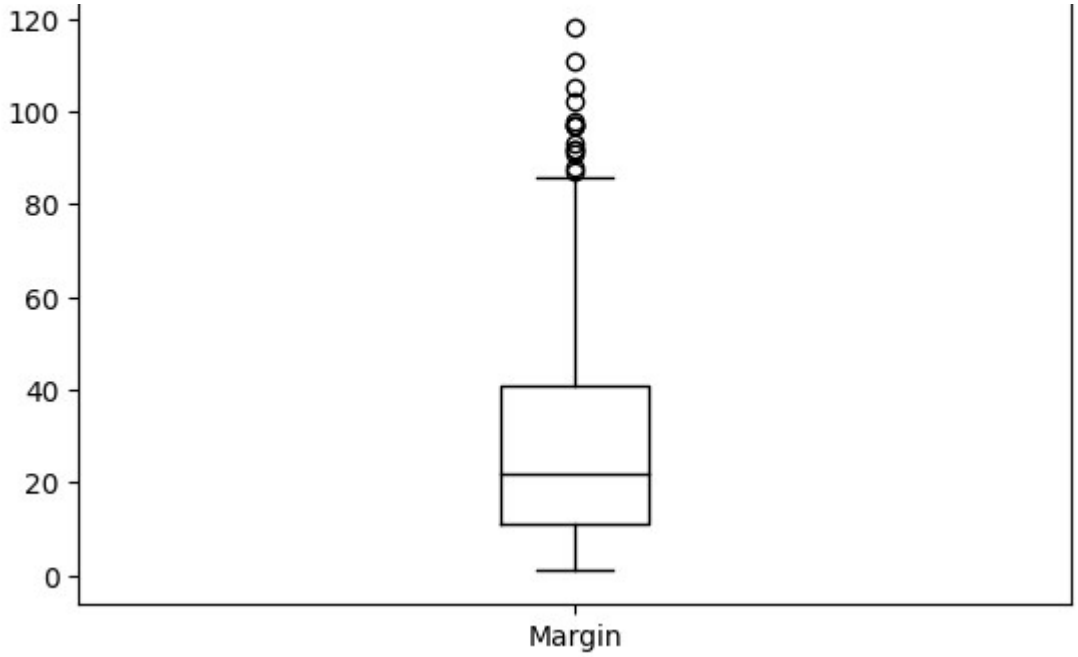
```
plt.show()

kurtosis = margins['Margin'].kurt()
print("Kurtosis:", kurtosis)

if kurtosis < 0:
    print("The distribution is platykurtic (lighter tails than normal).")
elif kurtosis > 0:
    print("The distribution is leptokurtic (heavier tails than normal).")
else:
    print("The distribution is mesokurtic (normal kurtosis).")
print("")
```

```
Mean win run margin is: 30.073286052009458

Standard Deviation of win run margin is: 26.78573611236935

Variance of win run margin is: 717.4756590814874

Min:  1.0 Max:  146.0 Range:  145.0

Q1:  11.0 Q3:  41.0 IQR:  30.0

Skewness: 1.5983821952655934
The distribution is highly positively skewed.

Kurtosis: 2.865078044947995
The distribution is leptokurtic (heavier tails than normal).
```

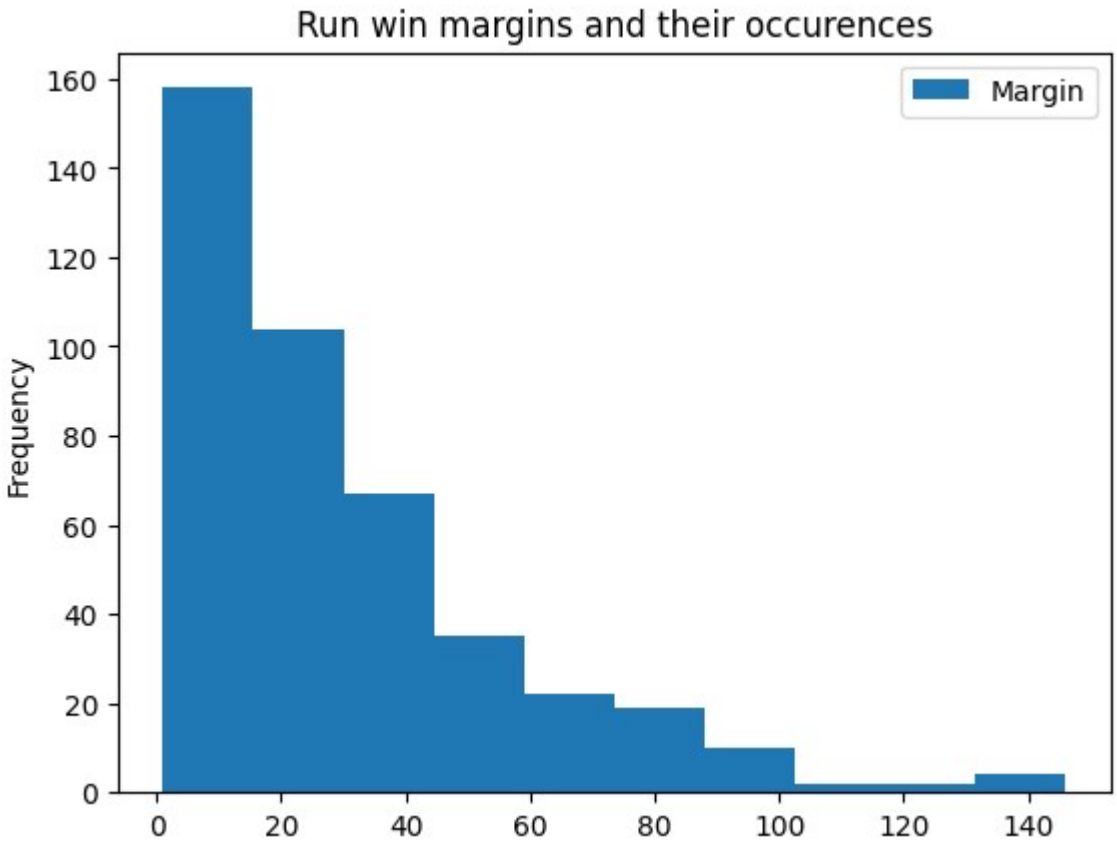
```
print("\nBoxplot")
margins.boxplot(column=['Margin'], grid=False, color='black')
plt.show()

print("\nHistogram")
margins.plot(kind='hist', title='Run win margins and their occurences', xlabel='Run win mar
plt.show()

print("\nKDE Plot")
sns.displot(data=margins, x="Margin", kde=True)
plt.show()

print("\nQQ Plot")
sm.qqplot(margins['Margin'], line='s')
plt.show()
```

```
Boxplot
```

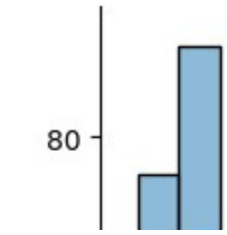


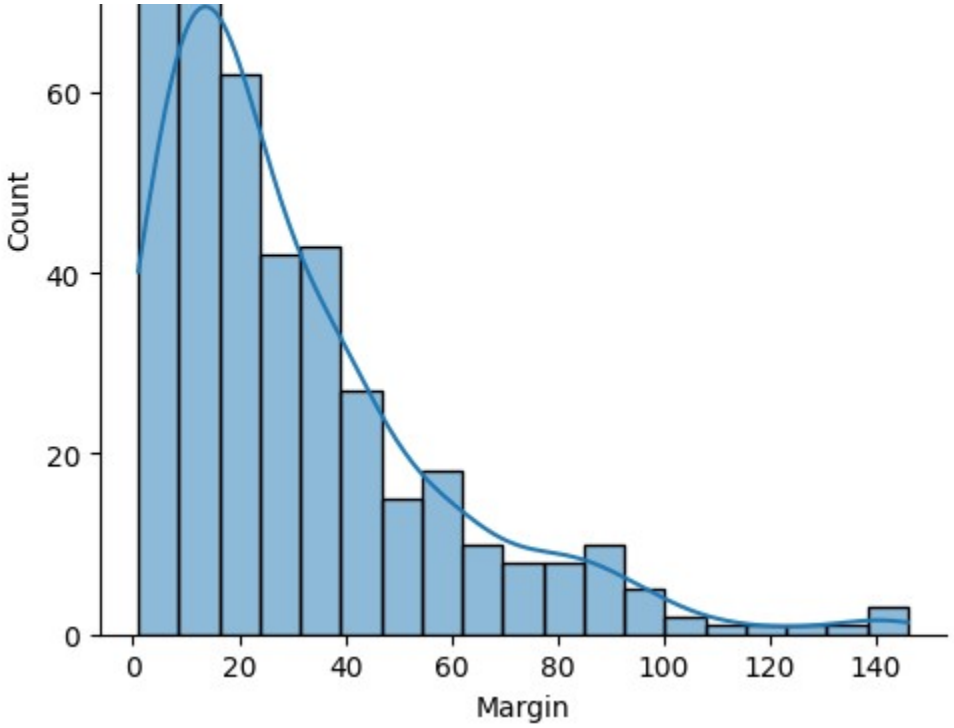


Histogram

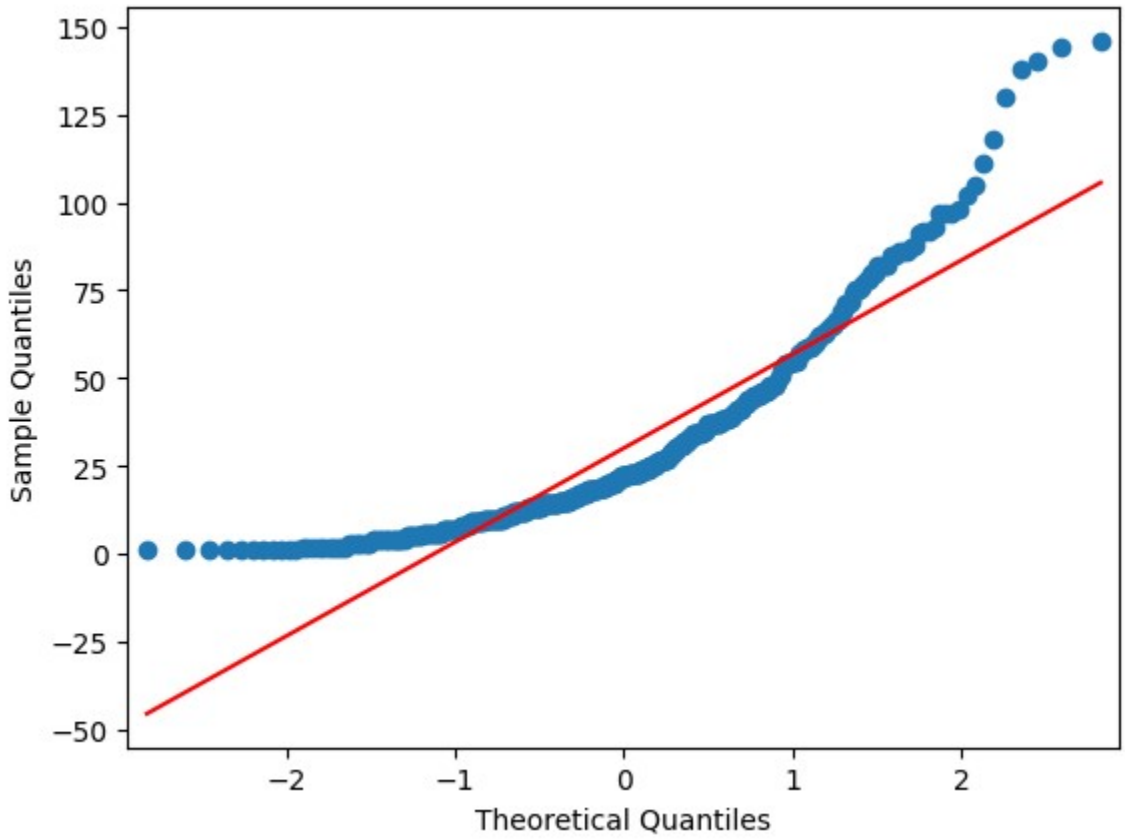


KDE Plot





QQ Plot




```
# filter out win margins when match was won chasing
# convert dataset into single variable "Margin"

rdf = df.filter(['WonBy', 'Margin'])
```

```
# WonBy (NoResult, Wickets, Runs, SuperOver)
rdf = rdf.query('WonBy == "Wickets"')

margins = rdf.filter(['Margin'])

print("The size of the cleaned dataset is:", margins.size, "records")
print("")
print("And it has the following data:\n", margins)
```

```
The size of the cleaned dataset is: 509 records

And it has the following data:
      Margin
0       7.0
1       7.0
3       7.0
4       5.0
5       5.0
..      ...
943     9.0
944     6.0
945     5.0
946     5.0
947     9.0

[509 rows x 1 columns]
```

```
# Measures of central tendency
print("Median win wicket margin is:", margins['Margin'].median())
print("")

print("Frequency table")
print(margins['Margin'].value_counts())

print("")
print("Mode win wicket margin is:")
print(margins['Margin'].mode())
print("")
```

```
Median win wicket margin is: 6.0

Frequency table
6.0     104
7.0     102
5.0      90
8.0      69
4.0      49
9.0      44
3.0      25
10.0     15
2.0       8
1.0       3
Name: Margin, dtype: int64
```

```
Name: Margin, dtype: int64

Mode win wicket margin is:
0    6.0
Name: Margin, dtype: float64
```

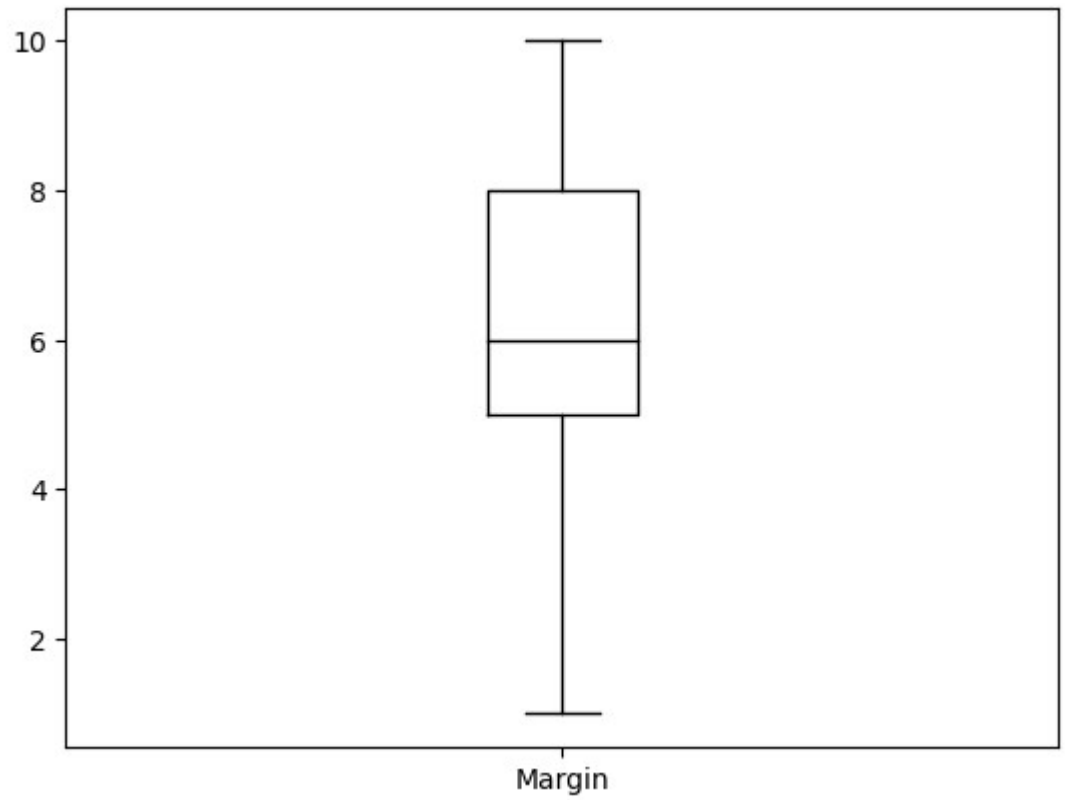
```
# Measures of dispersion
min = margins['Margin'].min()
max = margins['Margin'].max()
range = max - min

print("Min: ", min, "Max: ", max, "Range: ", range)
print("")

q3, q1 = np.percentile(margins['Margin'], [75 ,25])
iqr = q3 - q1
print("Q1: ", q1, "Q3: ", q3, "IQR: ", iqr)
print("")

skewness = margins['Margin'].skew()
print("Skewness:", skewness)

if skewness < -1:
    print("The distribution is highly negatively skewed.")
elif skewness >= -1 and skewness < -0.5:
    print("The distribution is moderately negatively skewed.")
elif skewness >= -0.5 and skewness < 0.5:
    print("The distribution is approximately symmetric.")
elif skewness >= 0.5 and skewness < 1:
    print("The distribution is moderately positively skewed.")
else:
    print("The distribution is highly positively skewed.")
print("")

kurtosis = margins['Margin'].kurt()
print("Kurtosis:", kurtosis)

if kurtosis < 0:
    print("The distribution is platykurtic (lighter tails than normal).")
elif kurtosis > 0:
    print("The distribution is leptokurtic (heavier tails than normal).")
else:
    print("The distribution is mesokurtic (normal kurtosis).")
print("")
```

```
Min:  1.0 Max:  10.0 Range:  9.0

Q1:  5.0 Q3:  8.0 IQR:  3.0

Skewness: -0.15206026931294128
The distribution is approximately symmetric.
```

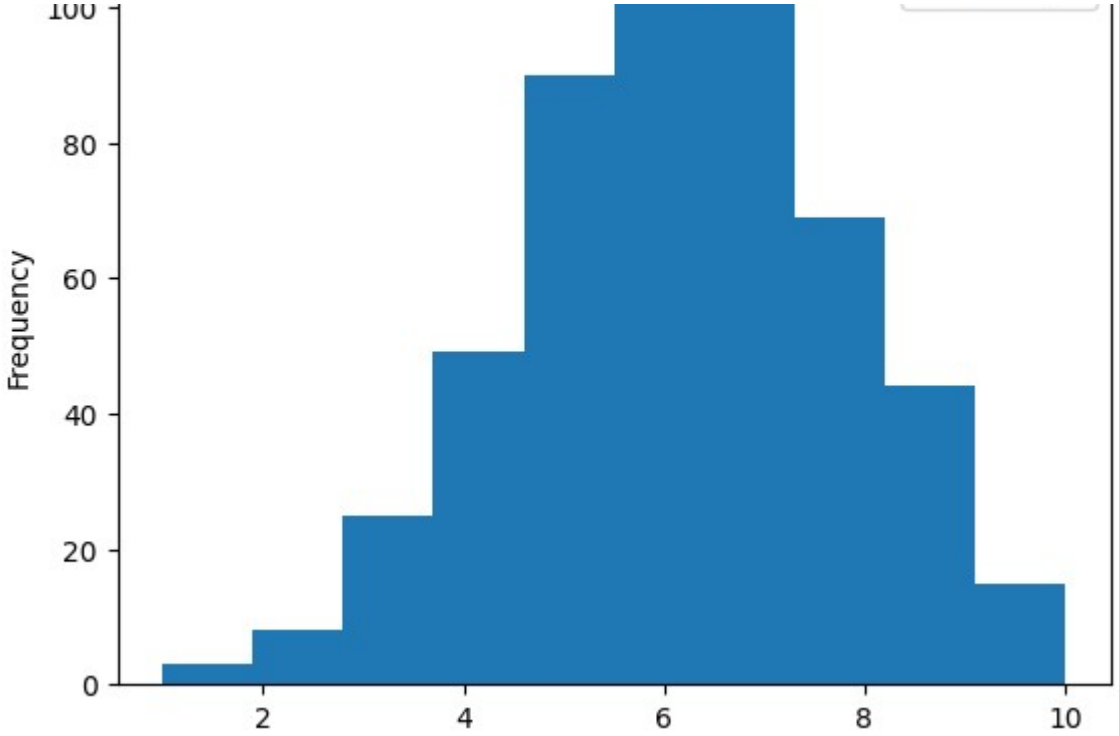
```
The distribution is approximately symmetric.

Kurtosis: -0.2931899308857573
The distribution is platykurtic (lighter tails than normal).
```

```
print("\nBoxplot")
margins.boxplot(column=['Margin'], grid=False, color='black')
plt.show()

print("\nHistogram")
margins.plot(kind='hist', title='Wicket win margins and their occurences', xlabel='Run win
plt.show()

print("\nKDE Plot")
sns.displot(data=margins, x="Margin", kde=True)
plt.show()

print("\nQQ Plot")
sm.qqplot(margins['Margin'], line='s')
plt.show()
```

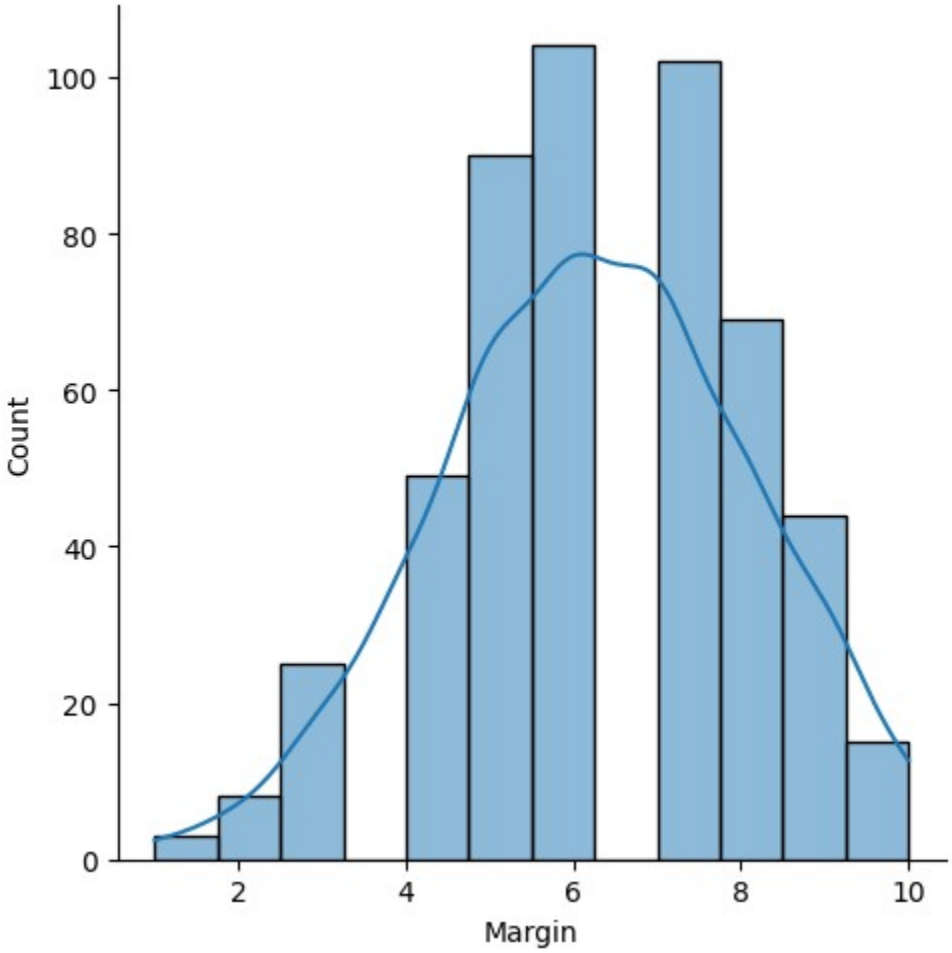
Boxplot



Histogram



KDE Plot



QQ Plot