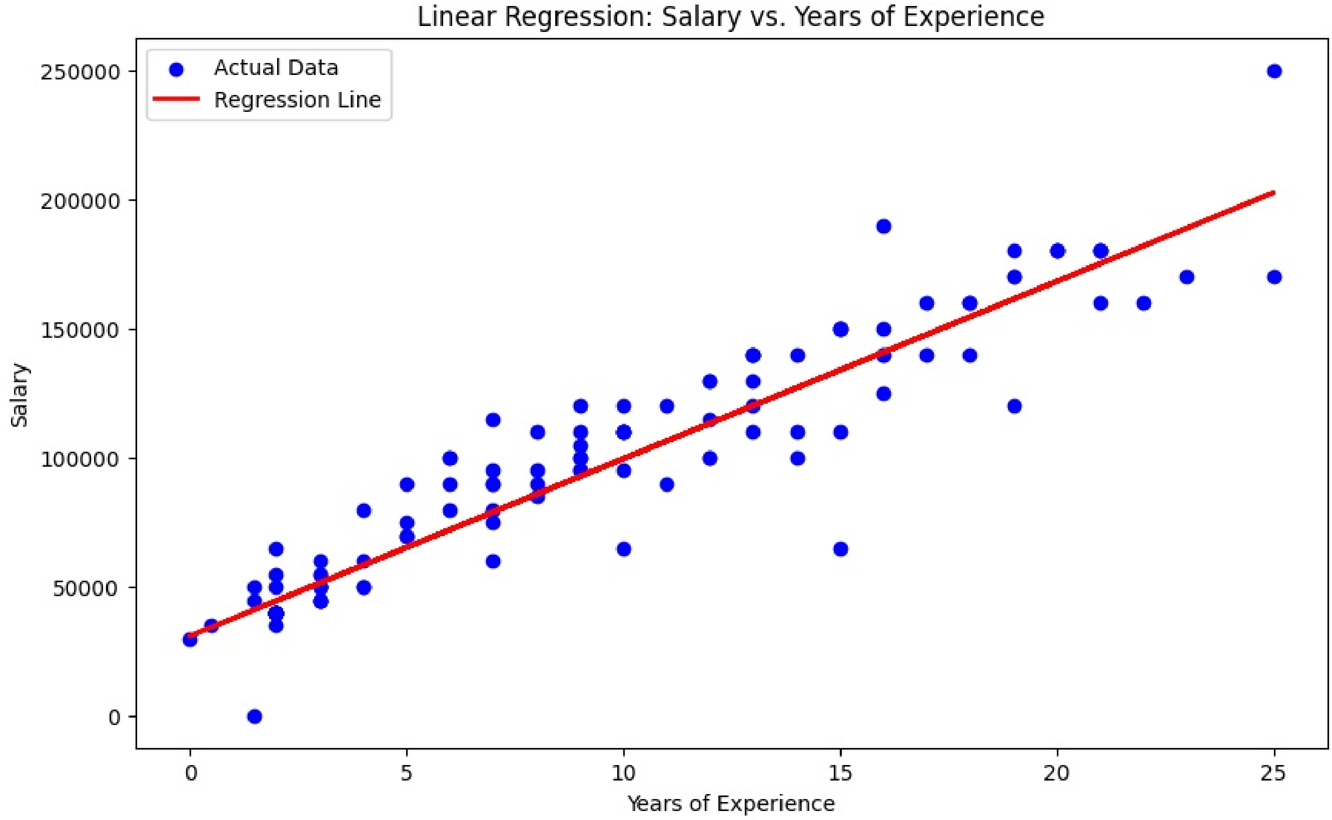
week 6

44. Salary Prediction according to Experience.

In [7]:

```
import pandas as p
import matplotlib.pyplot as m
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
da=p.read_csv("Salary_Experience.csv")
da.info()
da.dropna(axis=0,inplace=True)
x=da['Years of Experience'].values.reshape(-1,1)
y=da['Salary']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=42)
model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
m.figure(figsize=(10, 6))
m.scatter(x_test, y_test, color='blue', label='Actual Data')
m.plot(x_test, y_pred, color='red', linewidth=2, label='Regression Line')
m.title('Linear Regression: Salary vs. Years of Experience')
m.xlabel('Years of Experience');m.ylabel('Salary')
m.legend();m.show()
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("\nMean Squared Error:", mse)
print("\nR-squared:", r2)
Expe= [[3]] # Replace with the desired experience value
Sal = model.predict(Expe)
print(f"Predicted Salary for {Expe[0][0]} Year Experience : {Sal[0]}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 2 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Years of Experience  373 non-null    float64
 1   Salary               373 non-null    float64
dtypes: float64(2)
memory usage: 6.0 KB
```



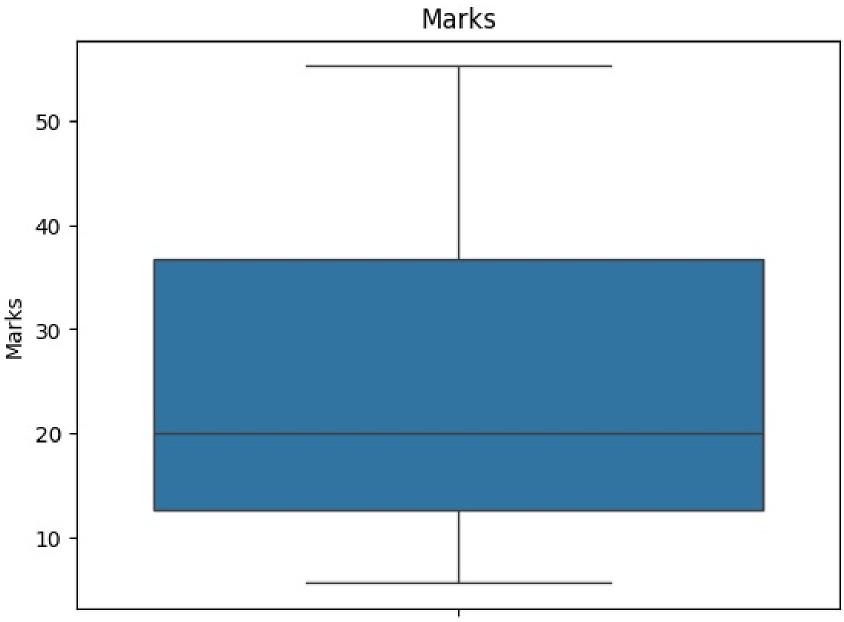

```
Mean Squared Error: 292476161.7847894

R-squared: 0.8705110527402834
Predicted Salary for 3 Year Experience : 51539.68015642409
```

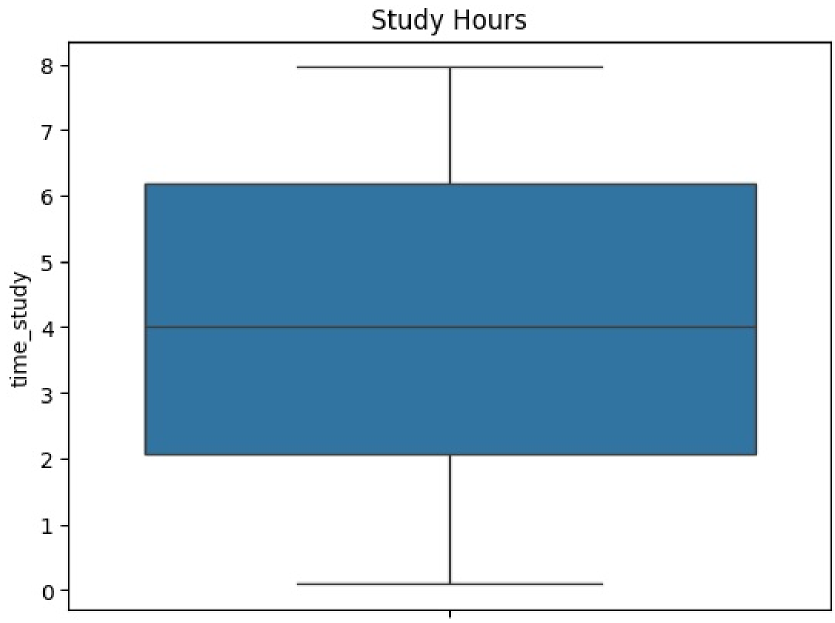
45. Marks Prediction according to Study Hours.

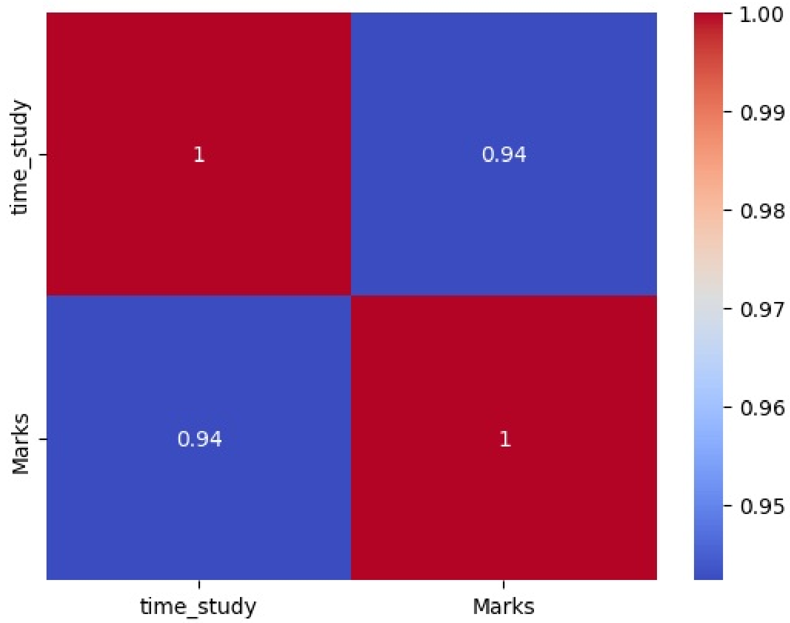
In [13]:

```
import pandas as p
import matplotlib.pyplot as m
import seaborn as s
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
da=p.read_csv("Student_marks.csv")
print(da.isna().sum())
s.boxplot(da['Marks']);m.title('Marks')
m.show()
s.boxplot(da['time_study']);m.title('Study Hours')
m.show()
s.heatmap(data=da.corr(), annot=True, cmap='coolwarm')
m.show()
X = da['time_study'].values.reshape(-1,1)
y = da['Marks']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)
m.figure(figsize=(10, 6))
m.scatter(X_test, y_test, color='blue', label='Actual Data')
m.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
m.title('Linear Regression: Salary vs. Years of Experience')
m.xlabel('Years of Experience');m.ylabel('Salary')
m.legend()
m.show()
Hours= [[20]]# Replace with the desired experience value
predicted_marks = model.predict(Hours)
print(f"Predicted Marks for {Hours[0][0]} Hours : {predicted_marks[0]}")
```

```
time_study    0
Marks         0
dtype: int64
```








```
Mean Squared Error: 25.23674562363223
R-squared: 0.9040228286990537
```


```
Predicted Marks for 20 Hours : 109.62199613197404
```

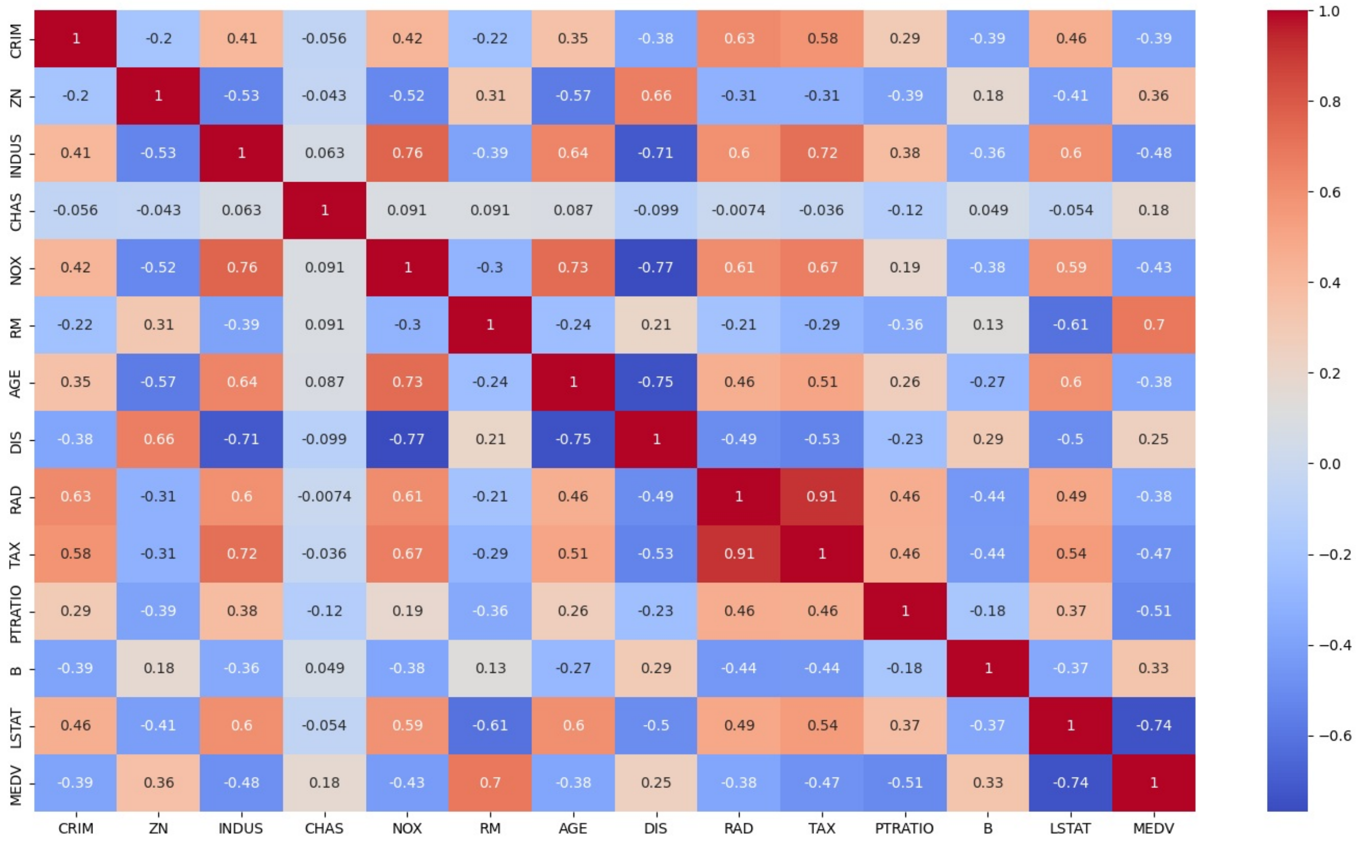
46. Boston housing price

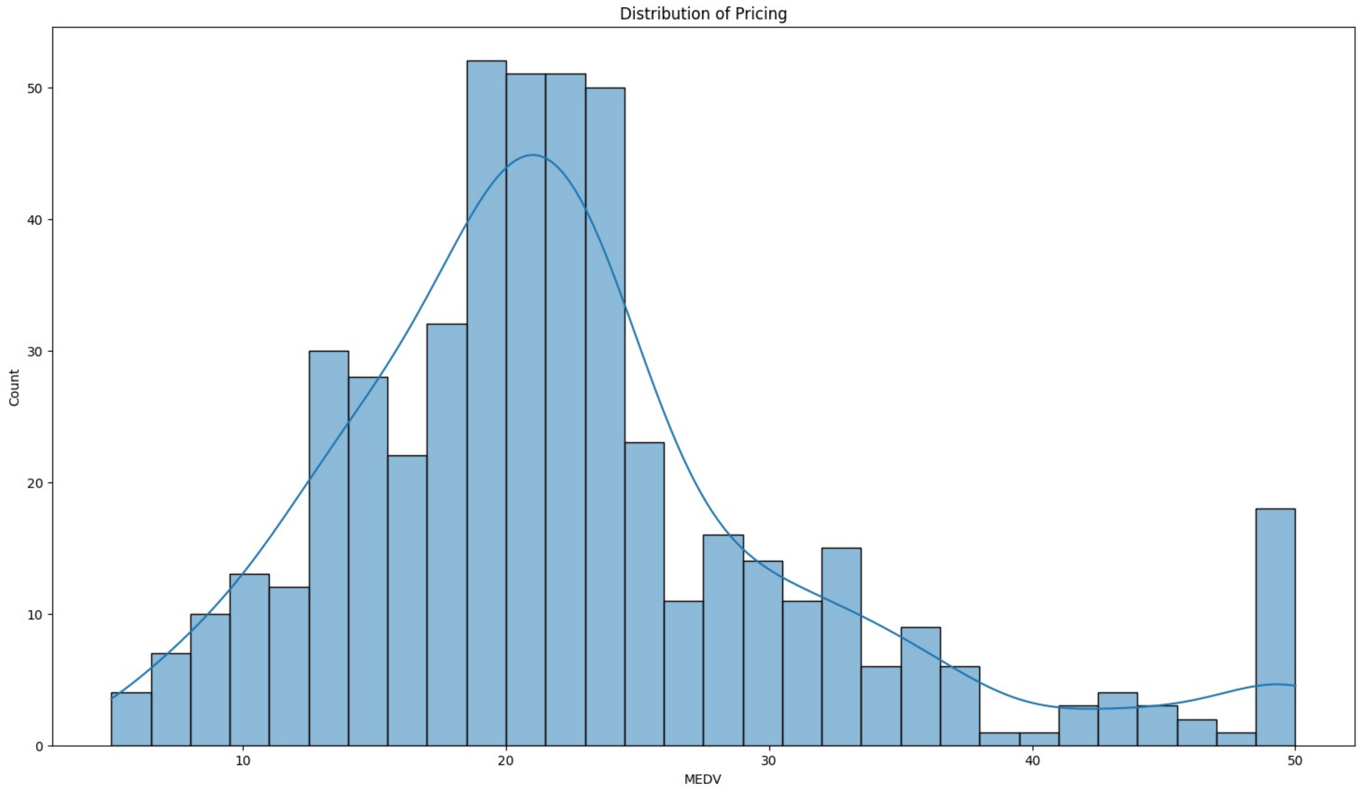
a) Preprocessing & Exploration

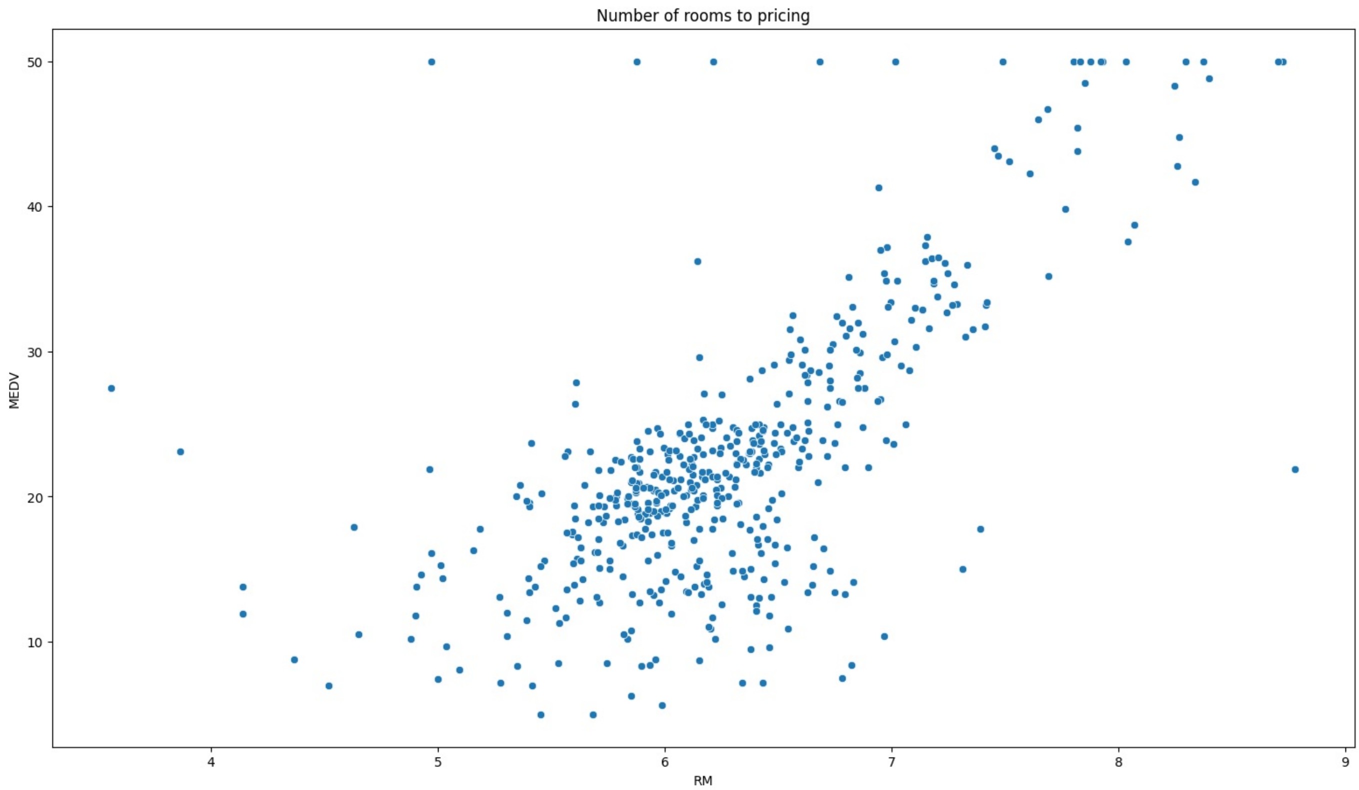
In [17]:

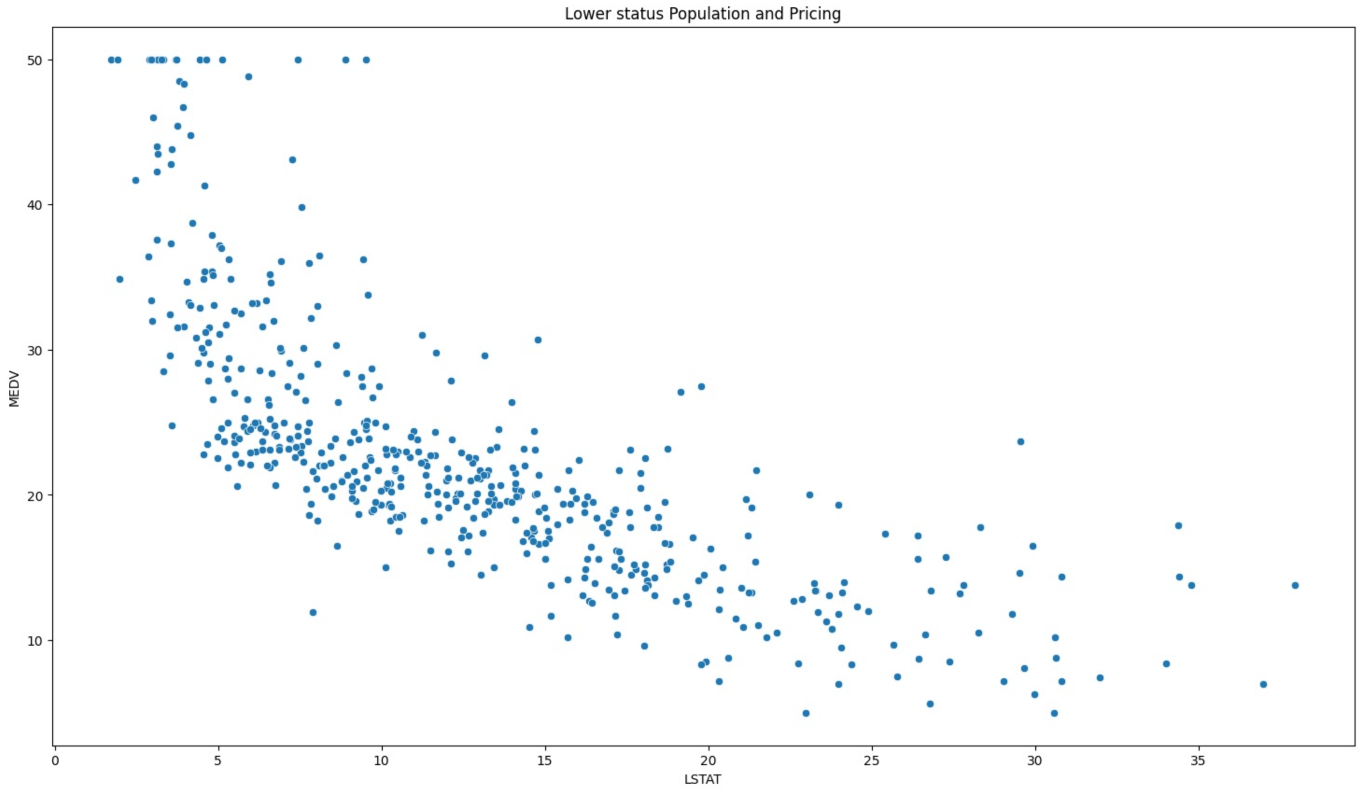
```
import pandas as p
import seaborn as s
import matplotlib.pyplot as m
da=p.read_csv("boston.csv")
da.info()
print(da.describe())
m.figure(figsize=(18,10))
s.heatmap(data=da.corr(),annot=True,cmap='coolwarm')
m.show()
m.figure(figsize=(18,10))
s.histplot(data=da,x='MEDV',bins=30,kde=True)
m.title('Distribution of Pricing')
m.show()
m.figure(figsize=(18,10))
s.scatterplot(data=da,x='RM',y='MEDV')
m.title('Number of rooms to pricing')
m.show()
m.figure(figsize=(18,10))
s.scatterplot(data=da,x='LSTAT',y='MEDV')
m.title('Lower status Population and Pricing')
m.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    int64  
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
             CRIM          ZN       INDUS        CHAS         NOX          RM  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000   
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634   
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617   
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000   
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500   
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500   
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500   
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000   

              AGE         DIS         RAD         TAX     PTRATIO           B  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000   
mean    68.574901    3.795043    9.549407  408.237154   18.455534  356.674032   
std     28.148861    2.105710    8.707259  168.537116    2.164946   91.294864   
min      2.900000    1.129600    1.000000  187.000000   12.600000    0.320000   
25%     45.025000    2.100175    4.000000  279.000000   17.400000  375.377500   
50%     77.500000    3.207450    5.000000  330.000000   19.050000  391.440000   
75%     94.075000    5.188425   24.000000  666.000000   20.200000  396.225000   
max    100.000000   12.126500   24.000000  711.000000   22.000000  396.900000   

            LSTAT        MEDV  
count  506.000000  506.000000  
mean    12.653063   22.532806  
std      7.141062    9.197104  
min      1.730000    5.000000  
25%      6.950000   17.025000  
50%     11.360000   21.200000  
75%     16.955000   25.000000  
max     37.970000   50.000000  
```




Distribution of Pricing
Count
MEDV
0
10
20
30
40
50


Number of rooms to pricing
MEDV
RM
4
5
6
7
8
9
10
20
30
40
50



b. Splitting

In [20]:
```
import numpy as n
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
X = da.drop('MEDV', axis=1)
y = da['MEDV']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("Train set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
rmse = n.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Train set: (354, 13) (354,)
Test set: (152, 13) (152,)
Root Mean Squared Error: 4.638689926172827
R-squared: 0.7112260057484925
```

47. Cricket match result

a. Data Preprocessing

In [23]:
```
import pandas as p
da=p.read_csv("matches.csv")
da.info()
#PRE_PROCESSING
miss=da.isna().sum()
print("\nMissing Values:\n",miss)
da.drop('umpire3', axis=1, inplace=True)
da.dropna(axis=0,inplace=True)
da.drop('date',axis=1,inplace=True)
print("\nMissing Values:\n",da.isna().sum())
```

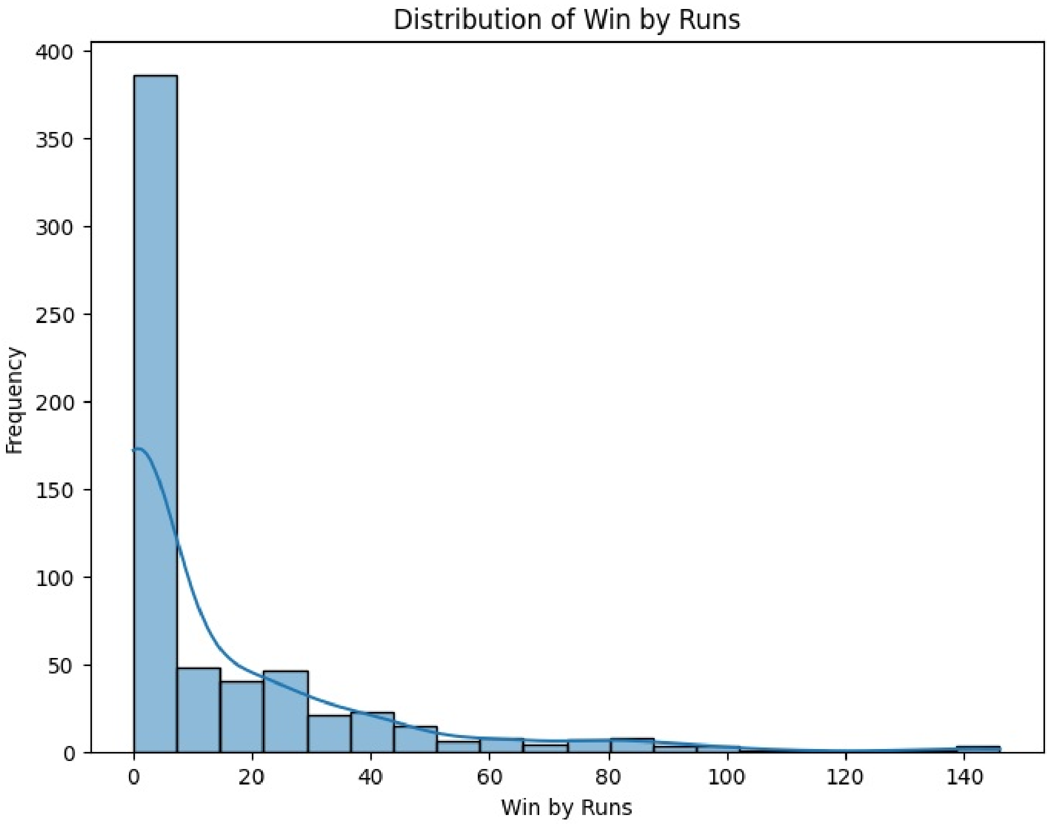
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB

Missing Values:
 id                   0
season               0
city                 7
date                 0
team1                0
team2                0
toss_winner          0
toss_decision        0
result               0
dl_applied           0
winner               3
win_by_runs          0
win_by_wickets       0
player_of_match      3
venue                0
umpire1              1
umpire2              1
umpire3            636
dtype: int64

Missing Values:
 id                 0
season             0
city               0
team1              0
team2              0
toss_winner        0
toss_decision      0
result             0
dl_applied         0
winner             0
win_by_runs        0
win_by_wickets     0
player_of_match    0
venue              0
umpire1            0
umpire2            0
dtype: int64
```

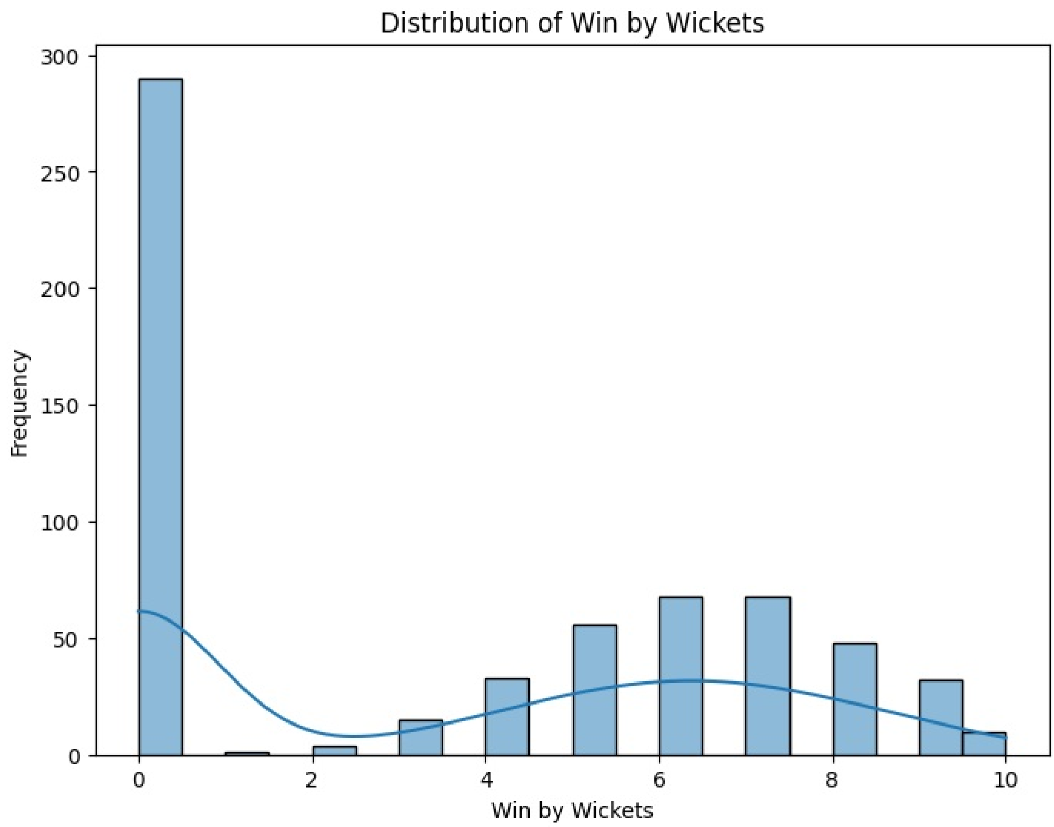
b. Data Exploration

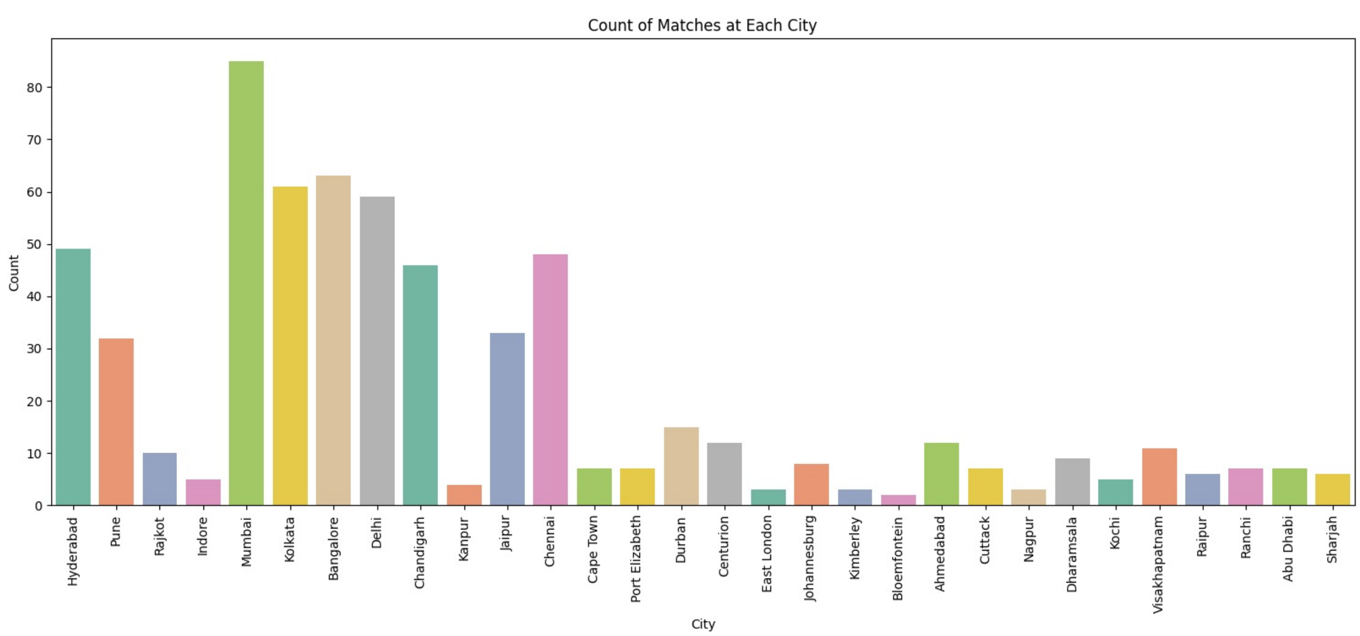
In [58]:

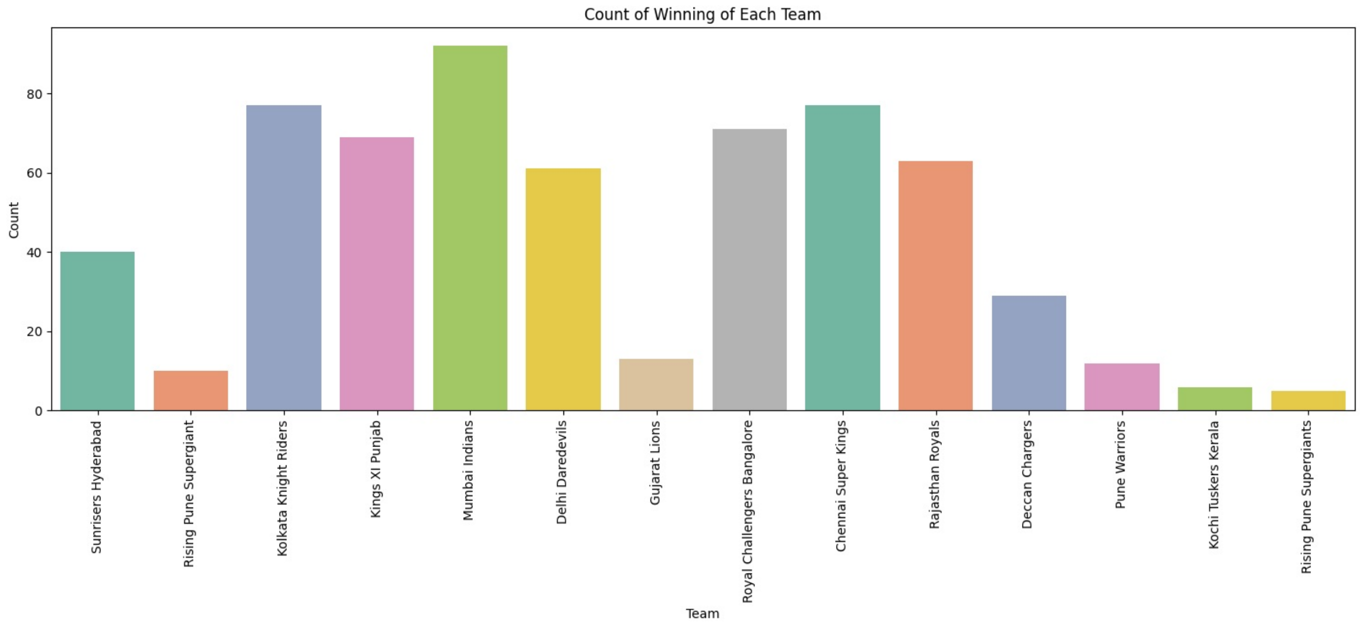
```python
import matplotlib.pyplot as m
import seaborn as s
# Function to create a histogram
def hist(da, col, title, xlabel):
    m.figure(figsize=(8, 6))
    s.histplot(da[col], bins=20, kde=True)
    m.title(title)
    m.xlabel(xlabel)
    m.ylabel('Frequency')
    m.show()
# Function to create a count plot
def countplot(da, x, title, xlabel):
    m.figure(figsize=(15, 7))
```

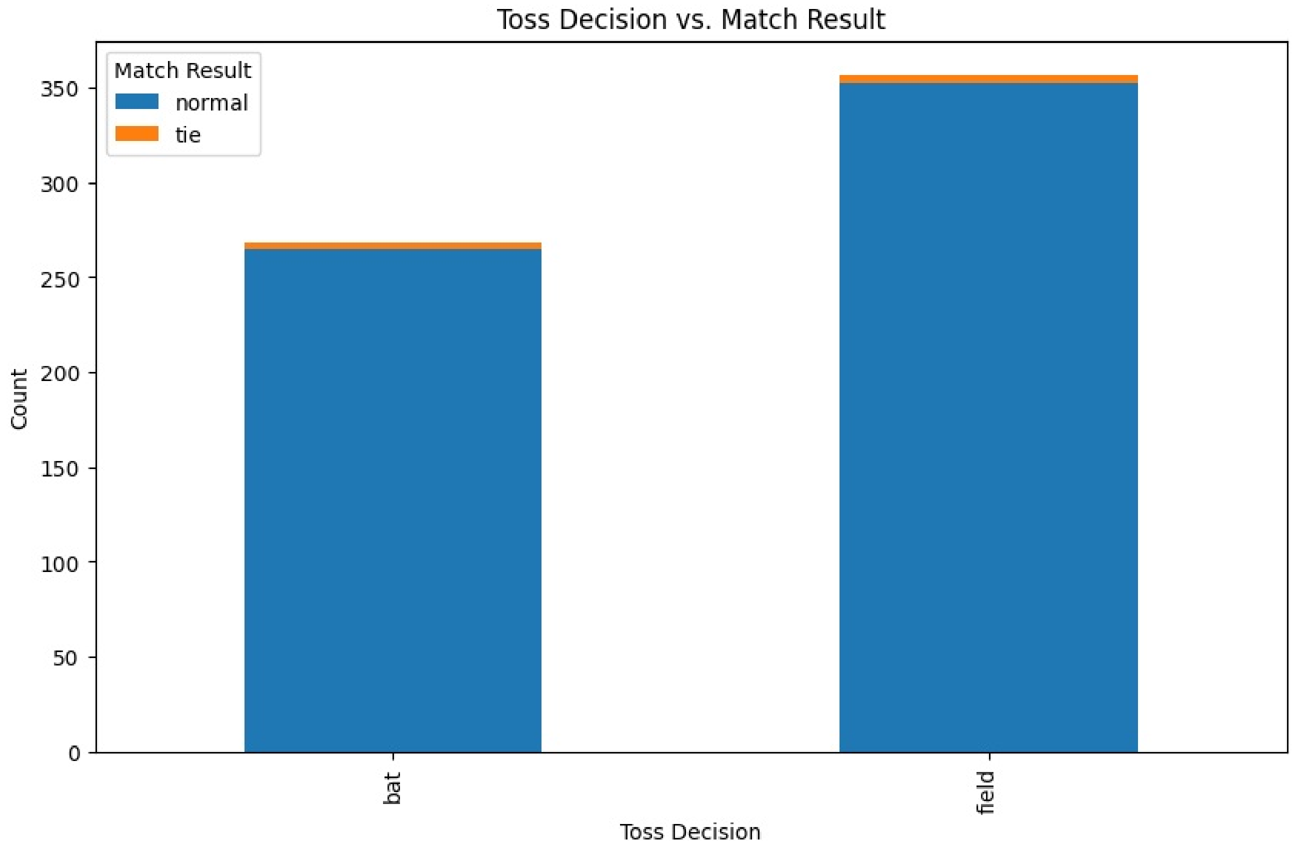
```
    s.countplot(x=x, data=da, hue=x, palette='Set2', legend=False)
    m.title(title)
    m.xlabel(xlabel)
    m.ylabel('Count')
    m.xticks(rotation=90)
    m.tight_layout()
    m.show()
# Function to create a stacked bar chart
def sta_bar(da, x, y, title, xlabel, ylabel, legend_title=None):
    dp = da.groupby([x, y]).size().unstack()
    dp.plot(kind='bar', stacked=True, figsize=(10, 6))
    m.title(title)
    m.xlabel(xlabel)
    m.ylabel(ylabel)
    if legend_title:
        m.legend(title=legend_title)
    m.show()
#DATA_EXPLORATION
hist(da, 'win_by_runs', 'Distribution of Win by Runs', 'Win by Runs')
hist(da, 'win_by_wickets', 'Distribution of Win by Wickets', 'Win by Wickets')
countplot(da, 'city', 'Count of Matches at Each City', 'City')
countplot(da, 'winner', 'Count of Winning of Each Team', 'Team')
sta_bar(da, 'toss_decision', 'result', 'Toss Decision vs. Match Result', 'Toss Decision', 'Count', legend_title=
sta_bar(da, 'season', 'toss_decision', 'Season-wise Toss Decision', 'Season', 'Count', legend_title='Toss Decis
```

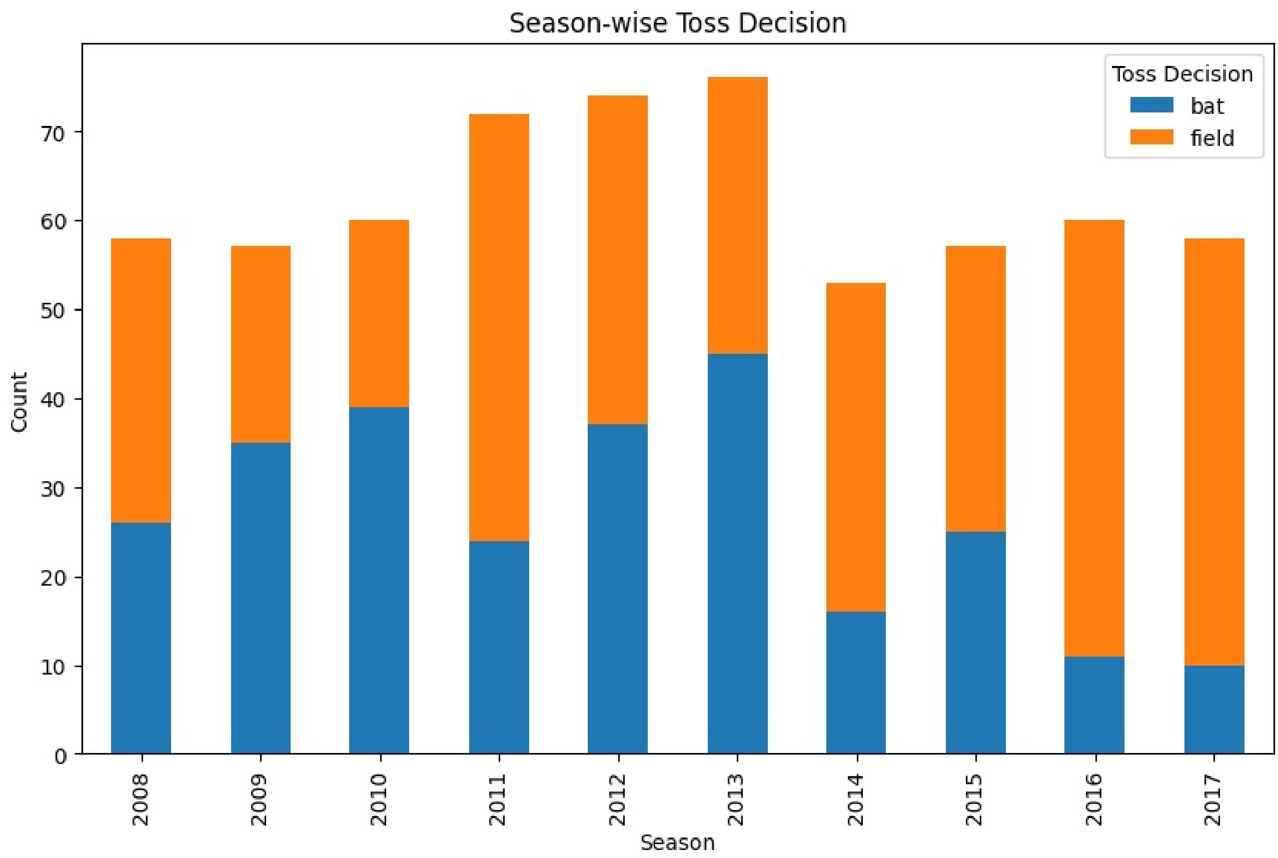



Distribution of Win by Wickets
Frequency
Win by Wickets
0
50
100
150
200
250
300
0
2
4
6
8
10


Count of Matches at Each City
Count
City
0
10
20
30
40
50
60
70
80
Hyderabad
Pune
Rajkot
Indore
Mumbai
Kolkata
Bangalore
Delhi
Chandigarh
Kanpur
Jaipur
Chennai
Cape Town
Port Elizabeth
Durban
Centurion
East London
Johannesburg
Kimberley
Bloemfontein
Ahmedabad
Cuttack
Nagpur
Dharamsala
Kochi
Visakhapatnam
Raipur
Ranchi
Abu Dhabi
Sharjah


Count of Winning of Each Team
Count
Team
Sunrisers Hyderabad
Rising Pune Supergiant
Kolkata Knight Riders
Kings XI Punjab
Mumbai Indians
Delhi Daredevils
Gujarat Lions
Royal Challengers Bangalore
Chennai Super Kings
Rajasthan Royals
Deccan Chargers
Pune Warriors
Kochi Tuskers Kerala
Rising Pune Supergiants


Toss Decision vs. Match Result
Match Result
normal
tie
Count
Toss Decision
bat
field



c. Splitting

In [13]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import pandas as p
da=p.read_csv("matches.csv")
#Transformation
categorical_columns = ['city', 'team1', 'team2', 'toss_winner',
'toss_decision', 'result', 'player_of_match', 'venue', 'umpire1',
'umpire2','winner','date']
label_encoders = {}
for column in categorical_columns:
    label_encoders[column] = LabelEncoder()
    da[column] = label_encoders[column].fit_transform(da[column])
X = da.drop('winner', axis=1)
y = da['winner']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
print("Train set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
Train set: (445, 17) (445,)
Test set: (191, 17) (191,)
```

48. Performance of a cricket player

a. Preprocessing

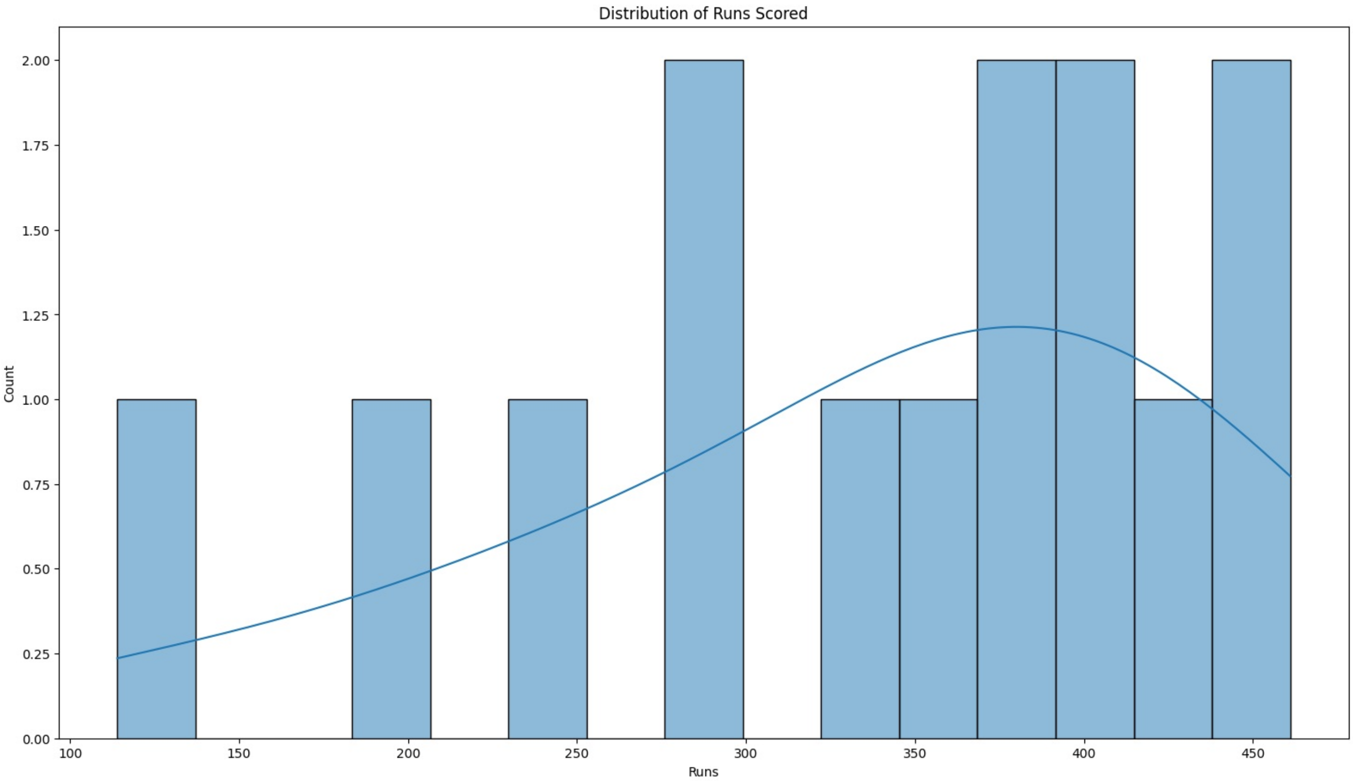
In [17]:
```
import pandas as p
da=p.read_csv("Dhoni1.csv")
da.info()
#PRE_PROCESSING
miss=da.isna().sum()
print("\nMissing Values:\n",miss)
da.dropna(inplace=True)
print("\nMissing Values:\n",da.isna().sum())
```

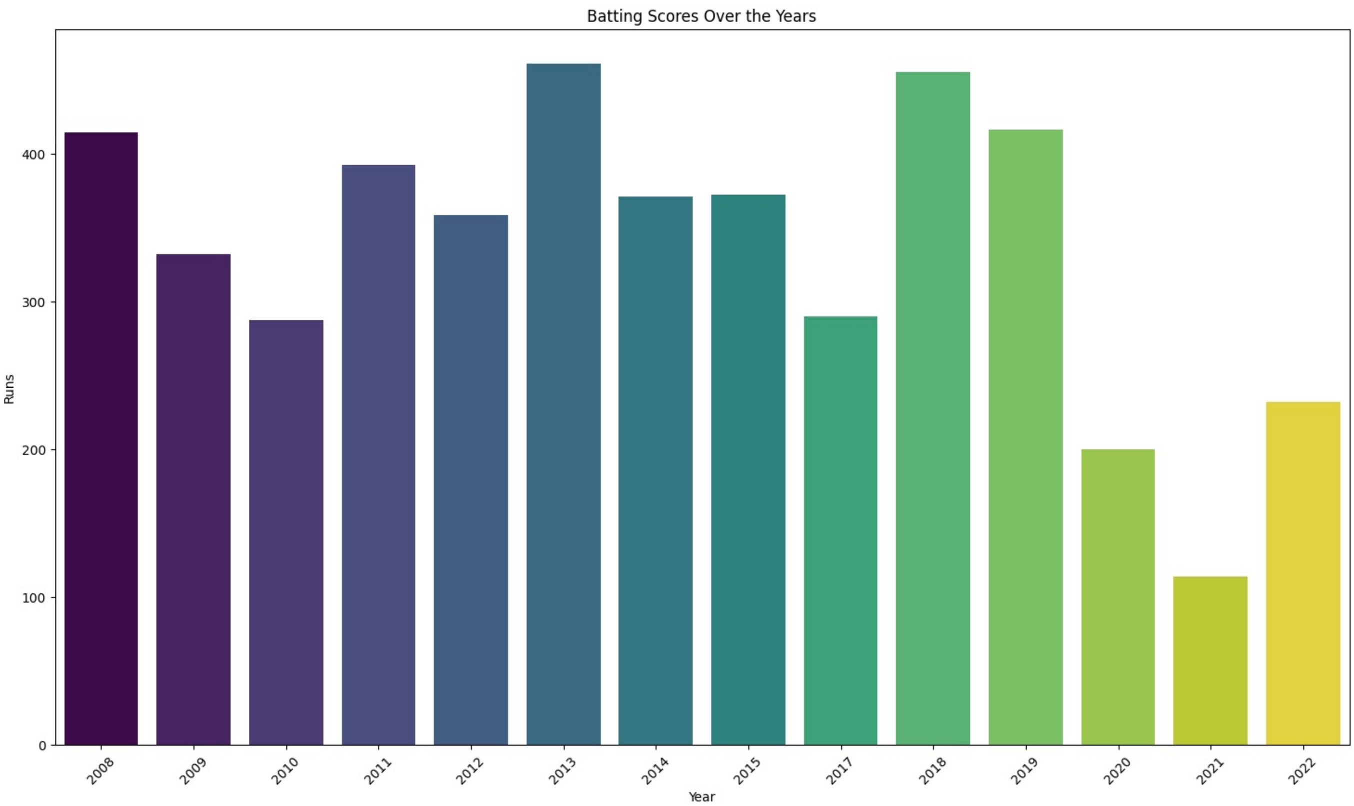
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Year            15 non-null     int64
 1   Matches         14 non-null     float64
 2   Innings         14 non-null     float64
 3   N.O.            14 non-null     float64
 4   Runs            14 non-null     float64
 5   Highest Score   14 non-null     float64
 6   Average         14 non-null     float64
 7   Strike Rate     14 non-null     float64
 8   100             14 non-null     float64
 9   50              14 non-null     float64
 10  Fours           14 non-null     float64
 11  Sixes           14 non-null     float64
 12  Catches Taken   14 non-null     float64
 13  Stumpings       14 non-null     float64
dtypes: float64(13), int64(1)
memory usage: 1.8 KB

Missing Values:
 Year             0
Matches          1
Innings          1
N.O.             1
Runs             1
Highest Score    1
Average          1
Strike Rate      1
100              1
50               1
Fours            1
Sixes            1
Catches Taken    1
Stumpings        1
dtype: int64

Missing Values:
 Year             0
Matches          0
Innings          0
N.O.             0
Runs             0
Highest Score    0
Average          0
Strike Rate      0
100              0
50               0
Fours            0
Sixes            0
Catches Taken    0
Stumpings        0
dtype: int64
```

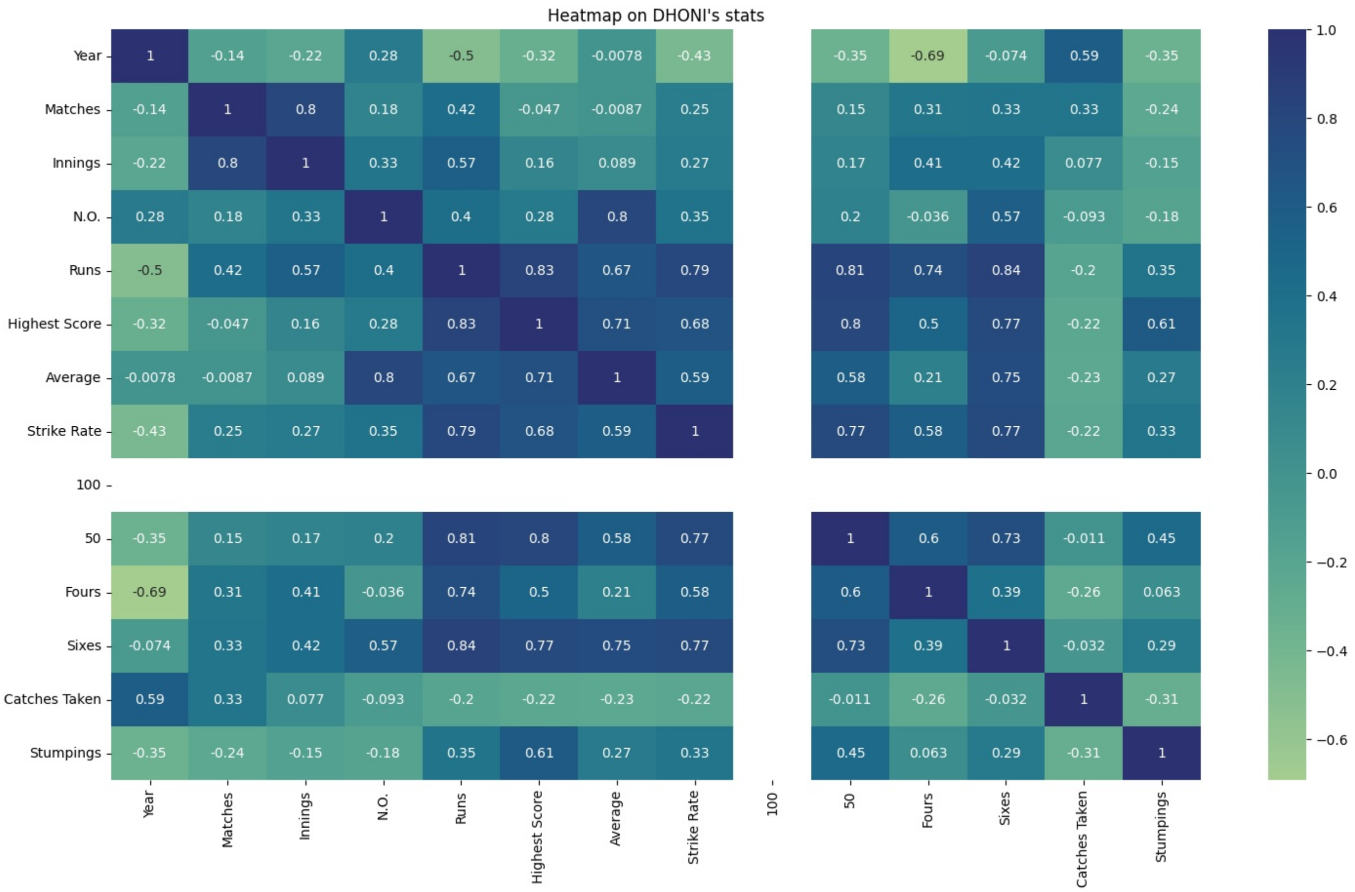
b. Exploration

In [21]:
```
import matplotlib.pyplot as m
import seaborn as s

# First plot: Histogram
```

```python
m.figure(figsize=(18,10))
s.histplot(da['Runs'], bins=15, kde=True)
m.title('Distribution of Runs Scored')
m.show()

# Second plot: Barplot
m.figure(figsize=(18,10))
s.barplot(data=da, x='Year', y='Runs', hue='Year', palette='viridis', legend=False)
m.title('Batting Scores Over the Years')
m.xlabel('Year')
m.ylabel('Runs')
m.xticks(rotation=45)
m.show()

# Third plot: Heatmap
m.figure(figsize=(18,10))
s.heatmap(data=da.corr(), annot=True, cmap='crest')
m.title("Heatmap on DHONI's stats")
m.show()
```







c. splitting

```
In [111…
from sklearn.model_selection import train_test_split
X = da.drop('Average', axis=1)
y = da['Average']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("Train set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
Train set: (10, 13) (10,)
Test set: (5, 13) (5,)
```

49. Crop yield

a. Preprocessing

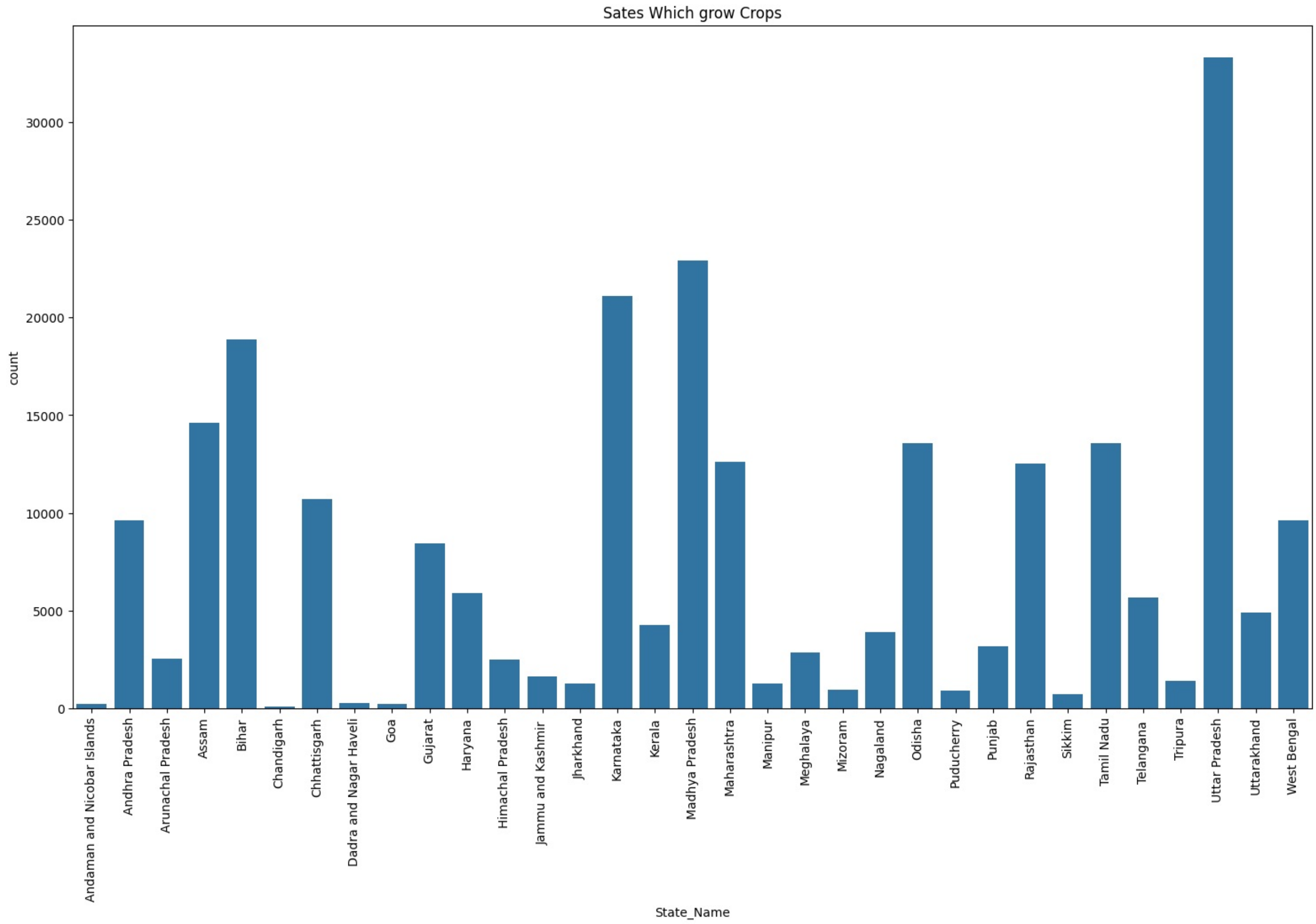
```
In [114…
import pandas as p
da=p.read_csv("crop_pre.csv")
da.info()
#PRE_PROCESSING
miss=da.isna().sum()
print("\nMissing Values:\n",miss)
da.dropna(inplace=True)
print("\nMissing Values:\n",da.isna().sum())
```

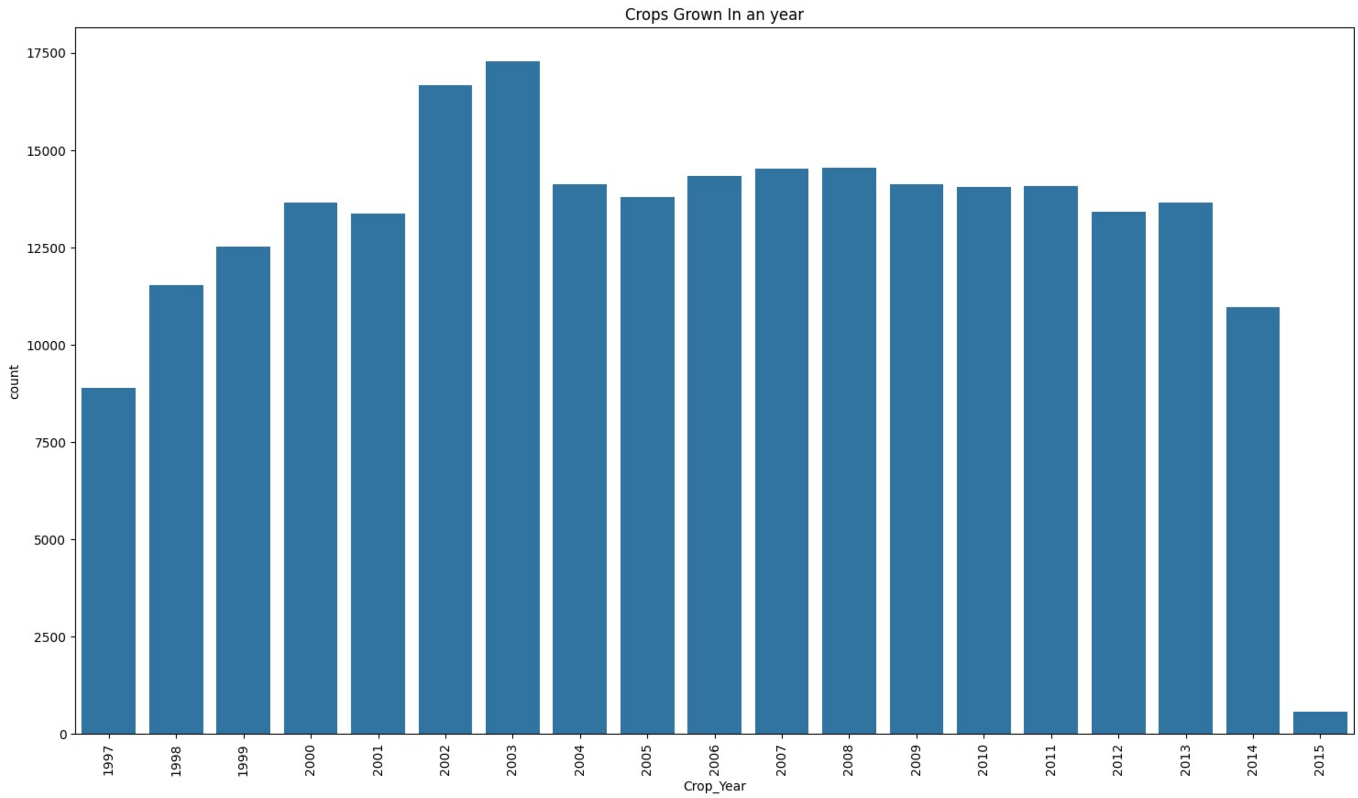
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 246091 entries, 0 to 246090
Data columns (total 7 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   State_Name     246091 non-null  object 
 1   District_Name  246091 non-null  object 
 2   Crop_Year      246091 non-null  int64  
 3   Season         246091 non-null  object 
 4   Crop           246091 non-null  object 
 5   Area           246091 non-null  float64
 6   Production     242361 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.1+ MB

Missing Values:
 State_Name          0
District_Name       0
Crop_Year           0
Season              0
Crop                0
Area                0
Production       3730
dtype: int64

Missing Values:
 State_Name       0
District_Name    0
Crop_Year        0
Season           0
Crop             0
Area             0
Production       0
dtype: int64
```

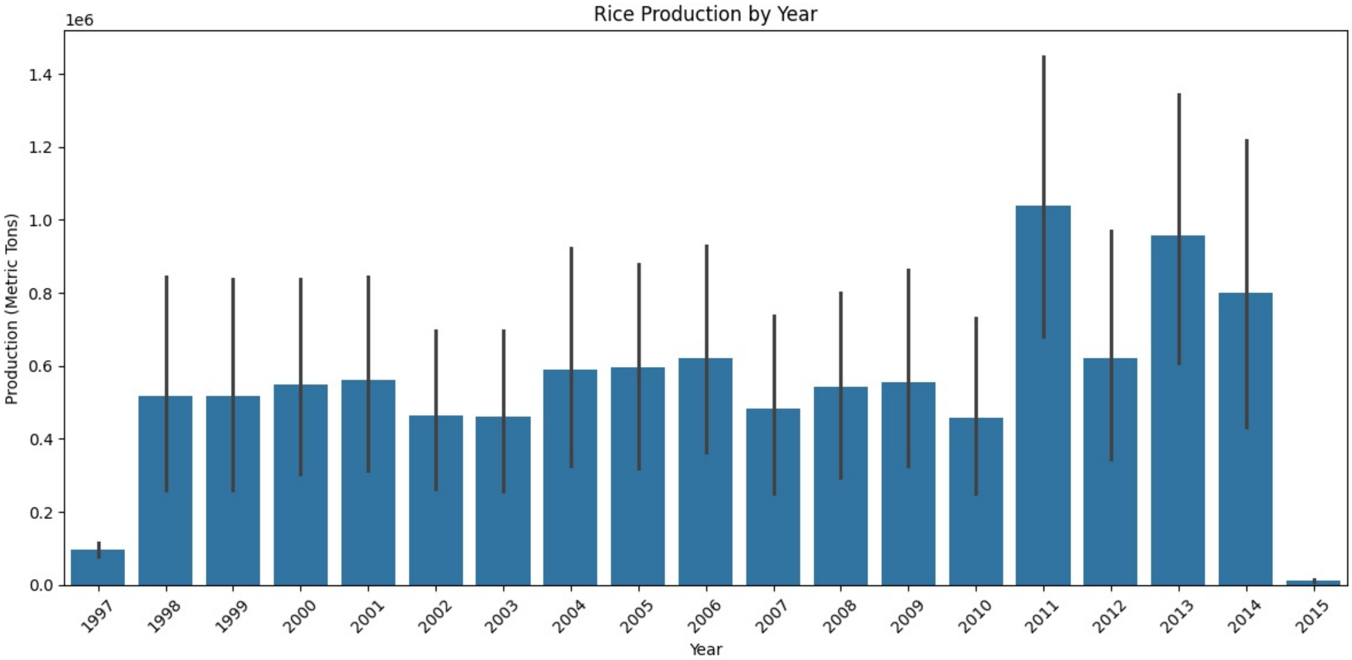
Data Exploration

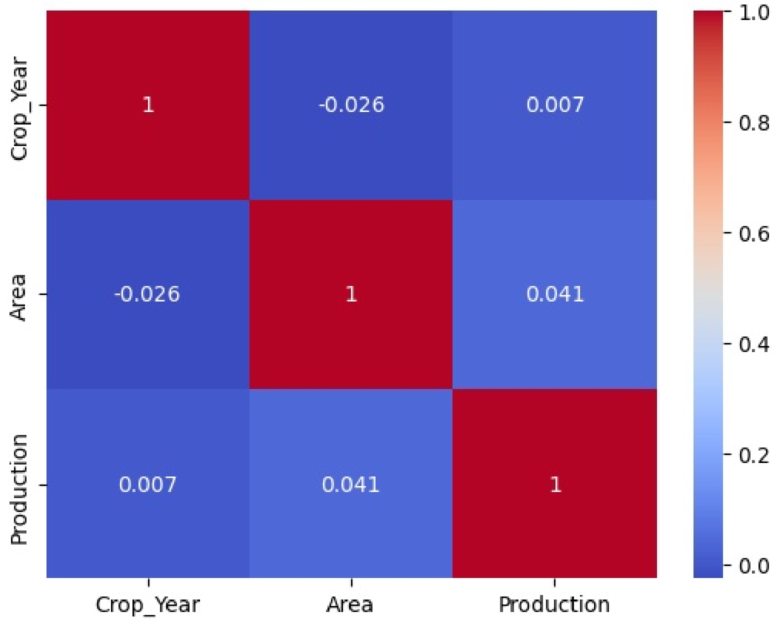
In [35]:

```python
import matplotlib.pyplot as m
import seaborn as s
import pandas as p
da=p.read_csv("crop_pre.csv")
m.figure(figsize=(18,10))
s.countplot(data=da, x='State_Name')
m.title("Sates Which grow Crops")
m.xticks(rotation=90)
m.show()

m.figure(figsize=(18,10))
s.countplot(data=da,x='Crop_Year')
m.title('Crops Grown In an year')
m.xticks(rotation=90)
m.show()

m.figure(figsize=(12, 6))
s.barplot(x=da['Crop_Year'], y=da['Production'],data=da)
m.title('Rice Production by Year')
m.xlabel('Year')
m.ylabel('Production (Metric Tons)')
m.xticks(rotation=45)
m.tight_layout()
m.show()

sn=da.select_dtypes(exclude=['object'])
s.heatmap(data=sn.corr(),annot=True,cmap='coolwarm')
m.show()
```


Sates Which grow Crops
count
30000
25000
20000
15000
10000
5000
0
Andaman and Nicobar Islands
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chandigarh
Chhattisgarh
Dadra and Nagar Haveli
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Puducherry
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
State_Name


Crops Grown In an year
count
17500
15000
12500
10000
7500
5000
2500
0
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
Crop_Year





c. Splitting

In [122…

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
#Transformation
categorical_columns = ['State_Name','District_Name','Season','Crop']
label_encoders = {}
for column in categorical_columns:
    label_encoders[column] = LabelEncoder()
    da[column] = label_encoders[column].fit_transform(da[column])
X = da.drop('Production', axis=1)
y = da['Production']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("Train set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
Train set: (169652, 6) (169652,)
Test set: (72709, 6) (72709,)
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js