**Assignment 3 & 4**

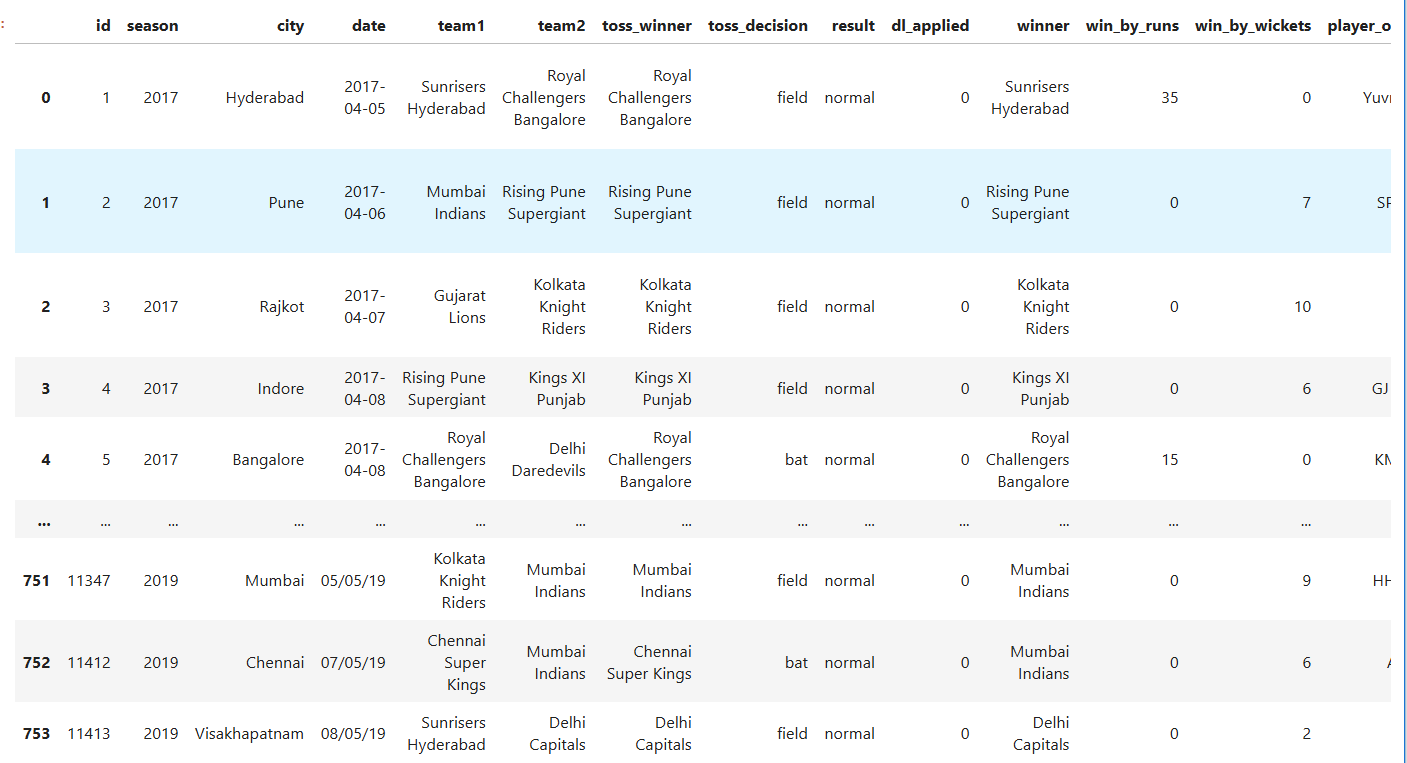
**IPL MATCH DATASET QUERIES**

* import pandas as pd

# Load the IPL dataset

ipl\_data = pd.read\_csv('ipl\_data.csv')

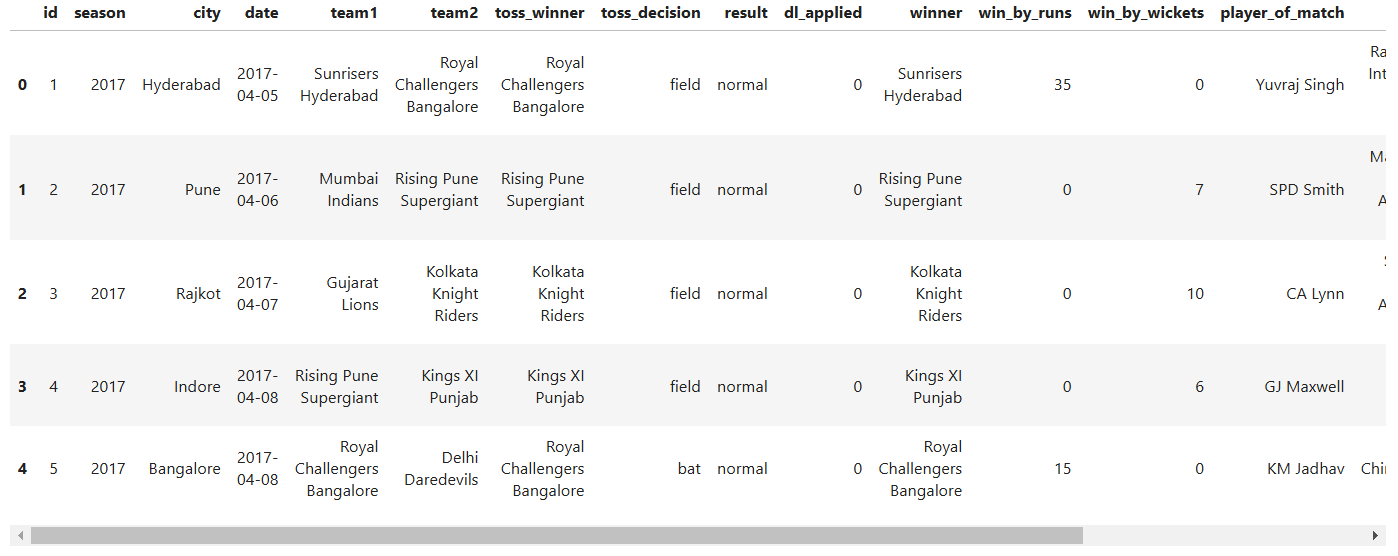
ipl\_data



1. **Display the first few rows of the dataset**

*ipl\_data.head()*

**Output-**



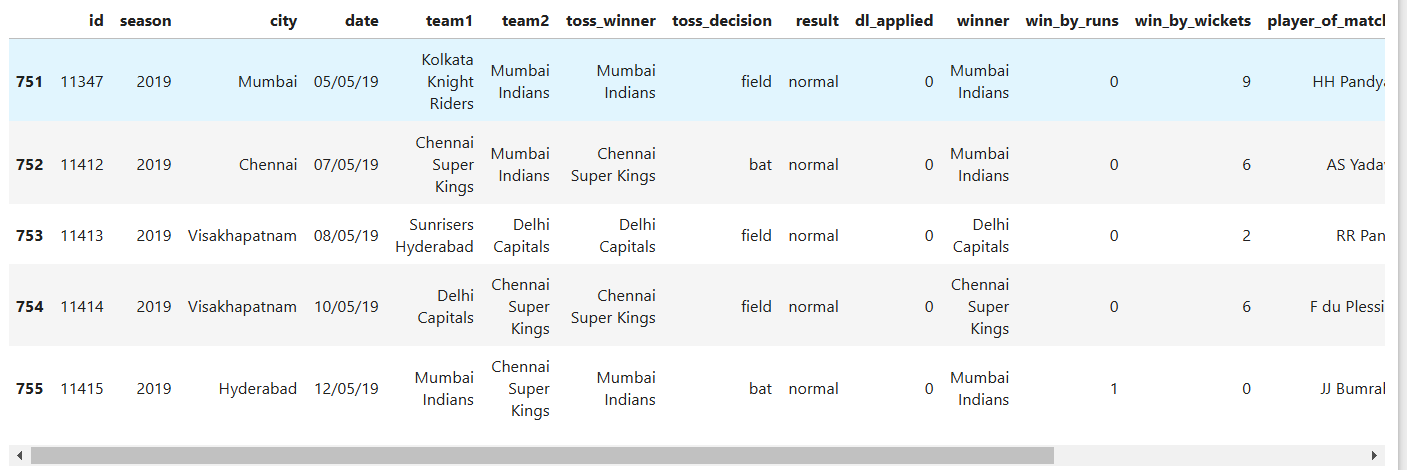
A screenshot of a sports schedule

Description automatically generated

1. **Display the last 5 rows of the DataFrame**

ipl\_data.tail()

**Output-**

****

**A screenshot of a sports schedule

Description automatically generated**

1. **Get the shape of the DataFrame (number of rows and columns)**

print(ipl\_data.shape)

**Output-**

****

1. **Get the data types of each column**

print(ipl\_data.dtypes)

**Output-**

**A screenshot of a computer code

Description automatically generated**

1. **Rename a column (e.g., renaming 'team1' to 'Home Team')**

ipl\_data.rename(columns={'team1': 'Home Team'}, inplace=True)

ipl\_data

**Output-**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a sports schedule

Description automatically generated**

1. **Rename the 'team1' column to 'Home Team'**

ipl\_data.rename(columns={'team1': 'Home Team'}, inplace=True)

# Display unique values in the 'Home Team' column before replacement

print("Before Replacement:")

print(ipl\_data['Home Team'].unique())

# Replace 'Delhi Daredevils' with 'Delhi Capitals' in the 'Home Team' column

ipl\_data['Home Team'] = ipl\_data['Home Team'].replace('Delhi Daredevils', 'Delhi Capitals')

# Display the updated unique values in the 'Home Team' column to confirm the change

print("\nAfter Replacement:")

print(ipl\_data['Home Team'].unique())

# Display the first few rows of the DataFrame to see the changes

print("\nUpdated DataFrame:")

print(ipl\_data.head())

**Output-**

**A close up of a text

Description automatically generated**

**A close-up of a white background

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. **Use the `str.contains` method to filter teams with 'Royal' in their name**

royal\_teams = ipl\_data[ipl\_data['team1'].str.contains('Royal', na=False)]

print("\nTeams with 'Royal' in their name:")

print(royal\_teams[['team1', 'team2']].head())

**Output-**

**A close-up of a computer screen

Description automatically generated**

1. **Use the `pd.to\_datetime` method to convert a column to datetime format and extract the year**

ipl\_data['year'] = pd.to\_datetime(ipl\_data['date']).dt.year

print("\nDataFrame with extracted year from date:")

print(ipl\_data[['date', 'year']].head())

**Output-**

**A screenshot of a computer screen

Description automatically generated**

1. **Use the `groupby` method to get the total number of matches played by each team**

matches\_played\_by\_team = ipl\_data.groupby('team1').size() + ipl\_data.groupby('team2').size()

print("\nTotal matches played by each team:")

print(matches\_played\_by\_team)

**Output-**

**A screenshot of a computer screen

Description automatically generated**

1. **Get a summary of the DataFrame (descriptive statistics)**

print(ipl\_data.describe())

**Output-**

**A screenshot of a computer code

Description automatically generated**

1. **Check for missing values in the DataFrame**

print(ipl\_data.isnull().sum())

**Output-**

**A screenshot of a computer

Description automatically generated**

1. **Drop rows with missing values**

ipl\_data.dropna(inplace=True)

ipl\_data

**Output-**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

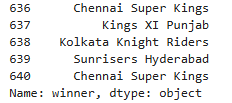
Description automatically generated**

1. **Select a specific column (e.g., 'winner')**

winners = ipl\_data['winner']

print(winners.head())

**Output-**

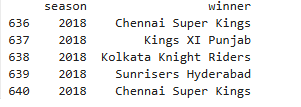
****

1. **Select multiple columns (e.g., 'season' and 'winner')**

selected\_columns = ipl\_data[['season', 'winner']]

print(selected\_columns.head())

**Output:-**

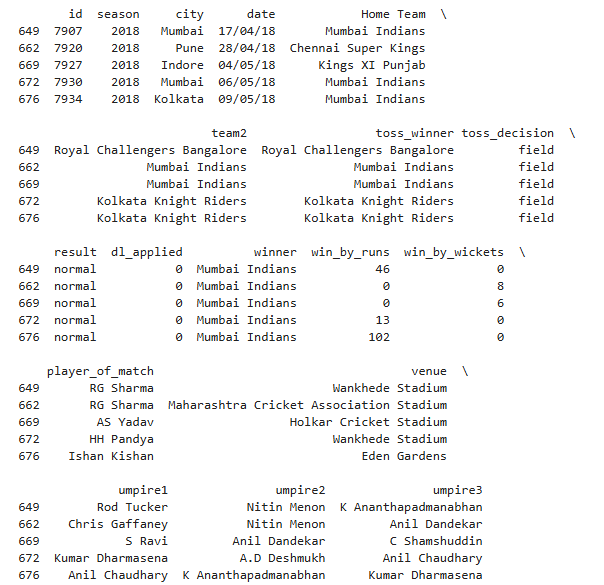
****

1. **Filter rows based on a condition (e.g., matches won by 'Mumbai Indians')**

mumbai\_wins = ipl\_data[ipl\_data['winner'] == 'Mumbai Indians']

print(mumbai\_wins.head())

**output**

****

**16. Group by a column and get the count (e.g., count of matches won by each team)**

match\_counts = ipl\_data['winner'].value\_counts()

print(match\_counts)

**OUTPUT**

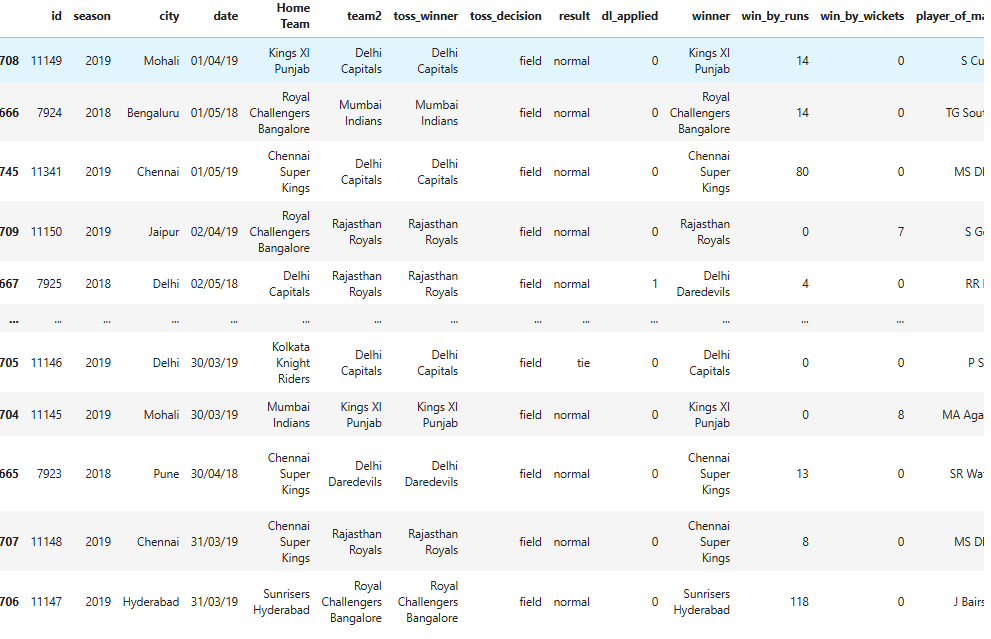
****

**17. Sort the DataFrame by a specific column (e.g., sorting by 'date')**

ipl\_data.sort\_values(by='date', inplace=True)

ipl\_data

**OUTPUT**

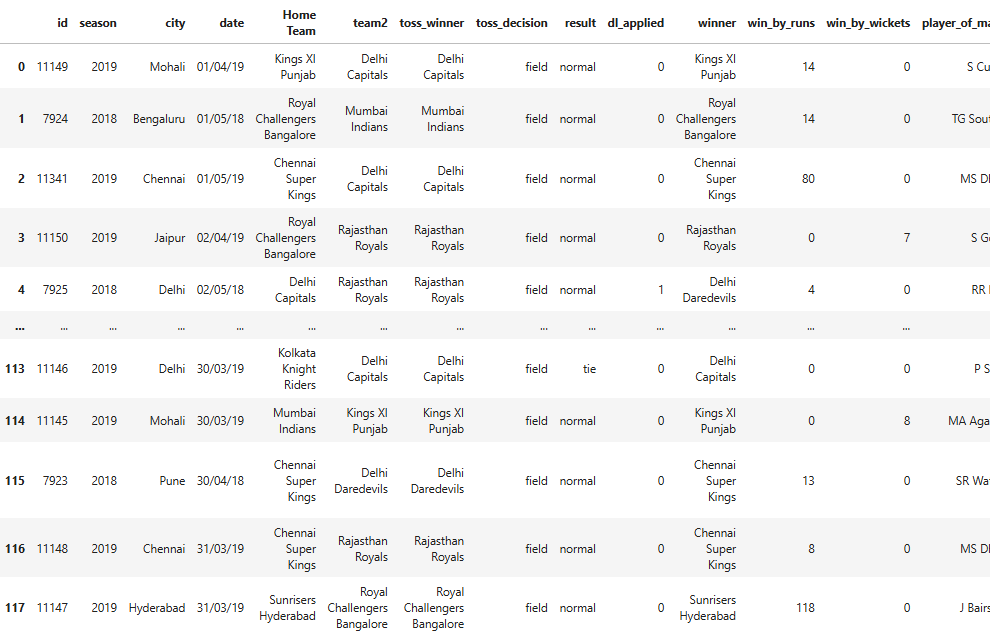
****

**18. Reset the index after sorting**

ipl\_data.reset\_index(drop=True, inplace=True)

ipl\_data

**OUTPUT**

****

**21. Get the index of the DataFrame**

print(ipl\_data.index)

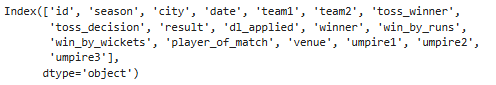
**OUTPUT**

****

**22. Get the columns of the DataFrame**

print(ipl\_data.columns)

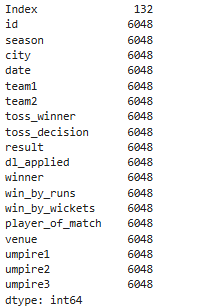
**OUTPUT**

****

**23. Check the memory usage of the DataFrame**

print(ipl\_data.memory\_usage())

**OUTPUT**



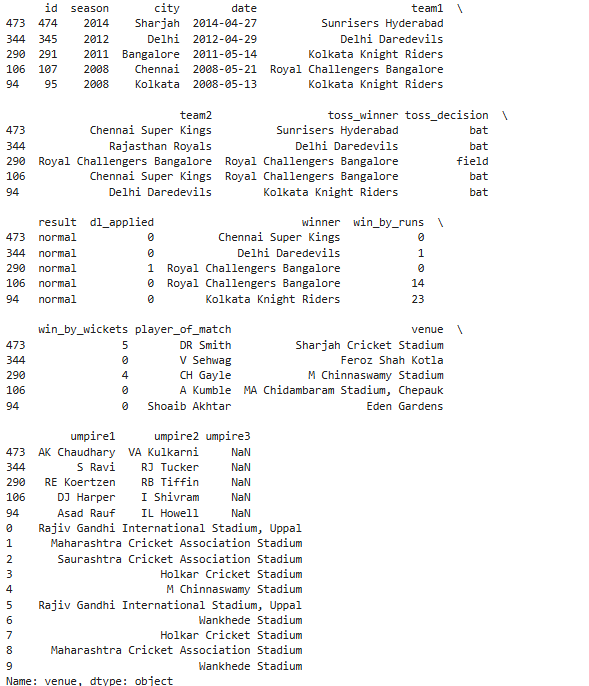
**24. Sample random rows from the DataFrame**

print(ipl\_data.sample(5))

**25. Check the first 10 rows of a specific column**

print(ipl\_data['venue'].head(10))

**OUTPUT**

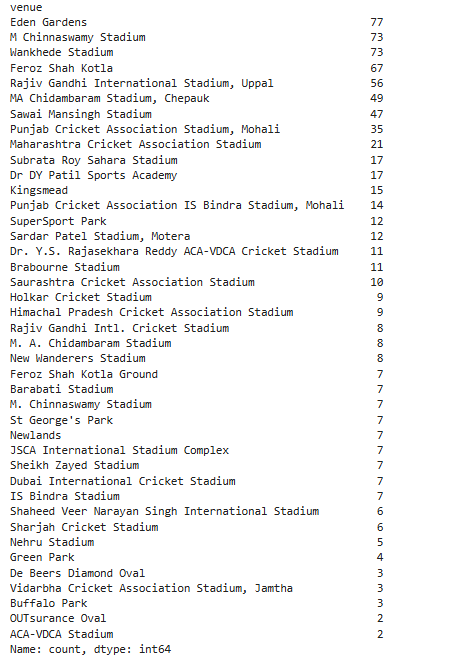
****

**26. Get the count of unique values in a specific column (e.g., 'venue')**

unique\_venue\_counts = ipl\_data['venue'].value\_counts()

print(unique\_venue\_counts)

**OUTPUT**

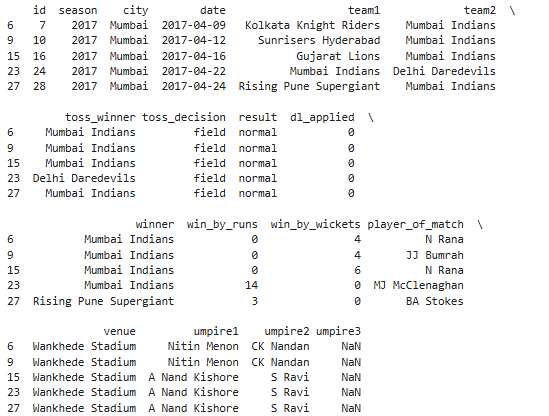
****

**27. Filter matches played in a specific city (e.g., 'Mumbai')**

mumbai\_matches = ipl\_data[ipl\_data['city'] == 'Mumbai']

print(mumbai\_matches.head())

**OUTPUT**



**28. Get the first occurrence of each team in the dataset**

first\_occurrences = ipl\_data.drop\_duplicates(subset='team1', keep='first')

print(first\_occurrences[['team1', 'date']])

**OUTPUT**

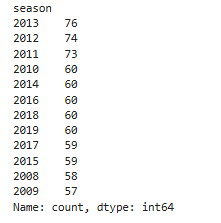
****

**29. Count the number of matches played in each season**

matches\_per\_season = ipl\_data['season'].value\_counts()

print(matches\_per\_season)

**OUTPUT**

****

**30. Get the average win margin for each team**

average\_win\_margin = ipl\_data.groupby('winner')['win\_by\_runs'].mean()

print(average\_win\_margin)

**OUTPUT**

****

**31. Find the match with the highest win margin**

highest\_win\_margin\_match = ipl\_data.loc[ipl\_data['win\_by\_runs'].idxmax()]

print(highest\_win\_margin\_match)

**OUTPUT**

****

**32. Count the number of matches that ended in a tie**

tie\_matches\_count = ipl\_data[ipl\_data['result'] == 'tie'].shape[0]

print(tie\_matches\_count)

**OUTPUT**

****

**33. Get the player of the match for all finals**

finals = ipl\_data[ipl\_data['result'] == 'normal'].iloc[-1]

print(finals[['season', 'player\_of\_match']])

**OUTPUT**

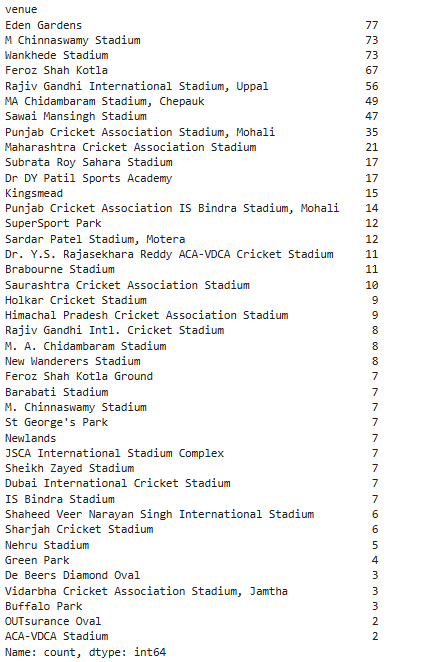
****

**34. Get the total number of matches played at each venue**

matches\_per\_venue = ipl\_data['venue'].value\_counts()

print(matches\_per\_venue)

**OUTPUT**

****

**35. Get the average number of runs scored by the winning team**

average\_runs\_winning\_team = ipl\_data.groupby('winner')['win\_by\_runs'].mean()

print(average\_runs\_winning\_team)

**OUTPUT**

****

**36. Get the player of the match for the most recent match**

most\_recent\_match = ipl\_data.iloc[-1]

print(most\_recent\_match['player\_of\_match'])

**OUTPUT**

****

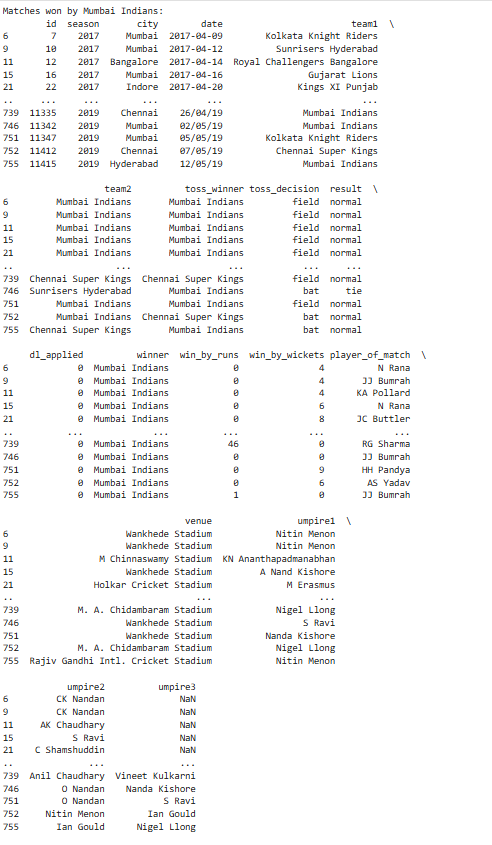
**37. Use the `query` method to filter matches where the winning team is 'Mumbai Indians'**

mumbai\_wins = ipl\_data.query("winner == 'Mumbai Indians'")

print("Matches won by Mumbai Indians:")

print(mumbai\_wins)

**OUTPUT**

****

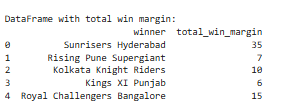
**38. Use the `assign` method to create a new column for the total win margin**

ipl\_data = ipl\_data.assign(total\_win\_margin=ipl\_data['win\_by\_runs'] + ipl\_data['win\_by\_wickets'])

print("\nDataFrame with total win margin:")

print(ipl\_data[['winner', 'total\_win\_margin']].head())

**OUTPUT**



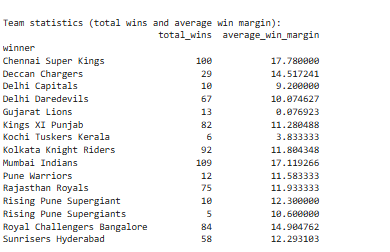
**39. Use the `groupby` and `agg` methods to get the total wins and average win margin for each team**

team\_stats = ipl\_data.groupby('winner').agg(total\_wins=('winner', 'count'), average\_win\_margin=('win\_by\_runs', 'mean'))

print("\nTeam statistics (total wins and average win margin):")

print(team\_stats)

**OUTPUT**

****

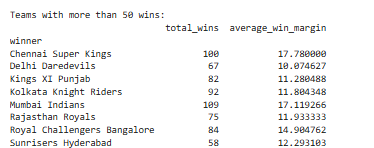
**40. Use the `filter` method to get teams that have won more than 50 matches**

teams\_with\_many\_wins = team\_stats[team\_stats['total\_wins'] > 50]

print("\nTeams with more than 50 wins:")

print(teams\_with\_many\_wins)

**OUTPUT**



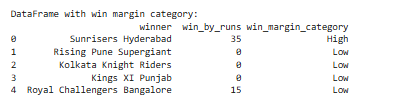
**41. Use the `apply` method to create a new column that categorizes win margins**

ipl\_data['win\_margin\_category'] = ipl\_data['win\_by\_runs'].apply(lambda x: 'High' if x > 30 else 'Low')

print("\nDataFrame with win margin category:")

print(ipl\_data[['winner', 'win\_by\_runs', 'win\_margin\_category']].head())

**OUTPUT**

****

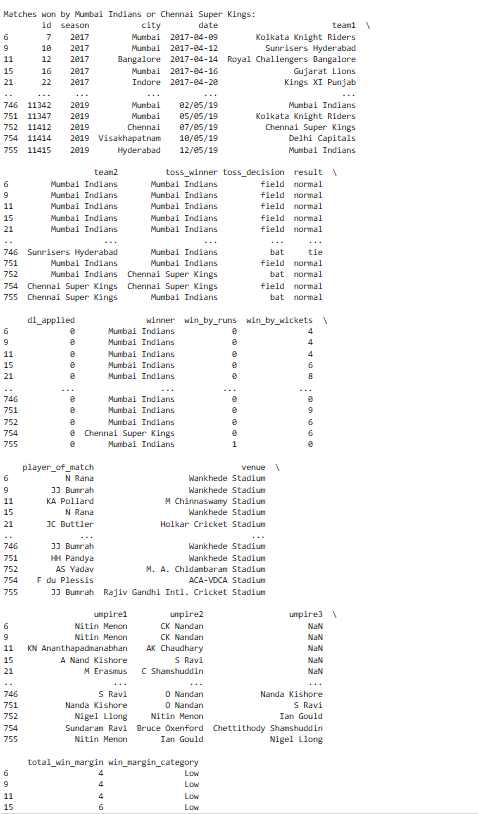
**42. Use the `isin` method to filter matches played by specific teams**

specific\_teams = ipl\_data[ipl\_data['winner'].isin(['Mumbai Indians', 'Chennai Super Kings'])]

print("\nMatches won by Mumbai Indians or Chennai Super Kings:")

print(specific\_teams)

**OUTPUT**

****

**43. Use the `set\_index` method to set the index to the match date and sort by index**

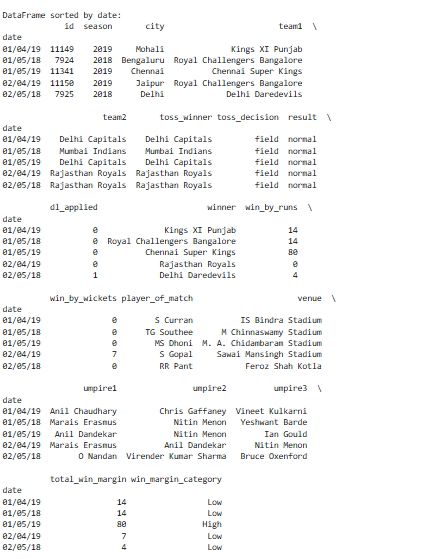
ipl\_data.set\_index('date', inplace=True)

ipl\_data.sort\_index(inplace=True)

print("\nDataFrame sorted by date:")

print(ipl\_data.head())

**OUTPUT**

****

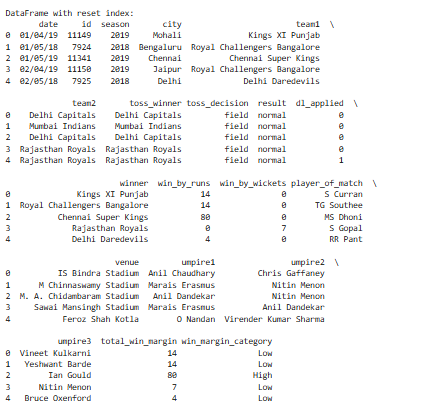
**44. Use the `reset\_index` method to reset the index back to default**

ipl\_data.reset\_index(inplace=True)

print("\nDataFrame with reset index:")

print(ipl\_data.head())

**OUTPUT**

****

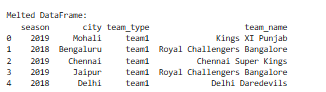
**# 45. Use the `melt` method to transform the DataFrame from wide to long format**

melted\_data = pd.melt(ipl\_data, id\_vars=['season', 'city'], value\_vars=['team1', 'team2'], var\_name='team\_type', value\_name='team\_name')

print("\nMelted DataFrame:")

print(melted\_data.head())

**OUTPUT**

****

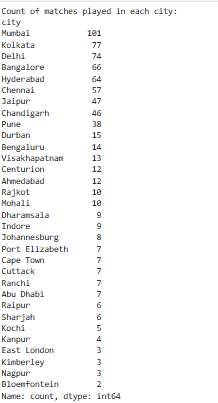
**46.Use the `value\_counts` method to get the count of matches played in each city**

matches\_per\_city = ipl\_data['city'].value\_counts()

print("Count of matches played in each city:")

print(matches\_per\_city)

**OUTPUT**

****

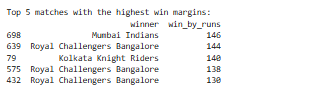
**47.Use the `nlargest` method to get the top 5 matches with the highest win margins**

top\_5\_win\_margins = ipl\_data.nlargest(5, 'win\_by\_runs')

print("\nTop 5 matches with the highest win margins:")

print(top\_5\_win\_margins[['winner', 'win\_by\_runs']])

**OUTPUT**

****

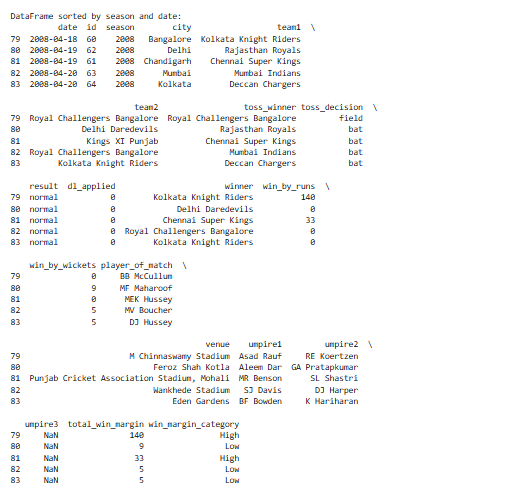
**48. Use the `sort\_values` method to sort the DataFrame by 'season' and 'date'**

sorted\_data = ipl\_data.sort\_values(by=['season', 'date'])

print("\nDataFrame sorted by season and date:")

print(sorted\_data.head())

**OUTPUT**

****

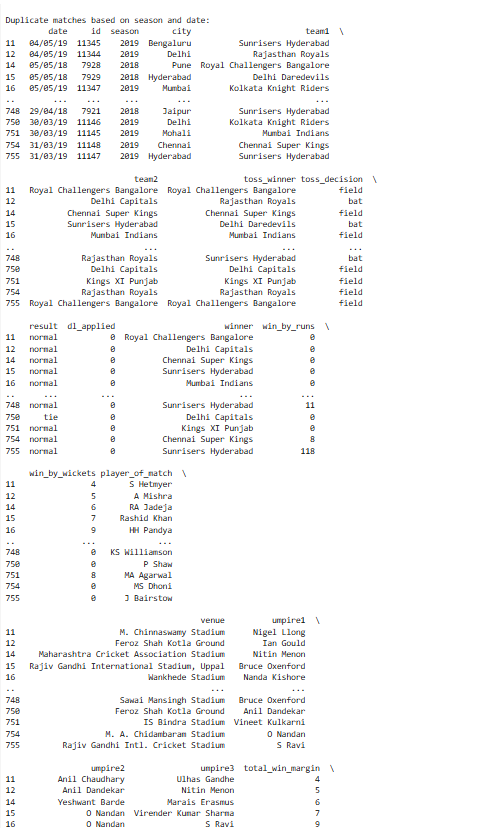
**49. Use the `duplicated` method to find duplicate matches based on 'season' and 'date'**

duplicate\_matches = ipl\_data[ipl\_data.duplicated(subset=['season', 'date'], keep=False)]

print("\nDuplicate matches based on season and date:")

print(duplicate\_matches)

**OUTPUT**

****

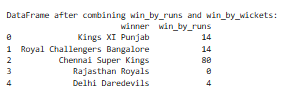
**50. Use the `combine\_first` method to fill missing values in one column with values from another column**

ipl\_data['win\_by\_runs'] = ipl\_data['win\_by\_runs'].combine\_first(ipl\_data['win\_by\_wickets'])

print("\nDataFrame after combining win\_by\_runs and win\_by\_wickets:")

print(ipl\_data[['winner', 'win\_by\_runs']].head())

**OUTPUT**

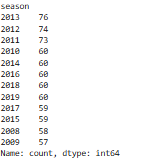
****

**51. Get the total number of matches played in each season**

matches\_per\_season = ipl\_data['season'].value\_counts()

print(matches\_per\_season)

**OUTPUT**

****

**52. Get the team with the highest win margin in a single match**

highest\_win\_margin = ipl\_data.loc[ipl\_data['win\_by\_runs'].idxmax()]

print(highest\_win\_margin[['winner', 'win\_by\_runs']])

**OUTPUT**

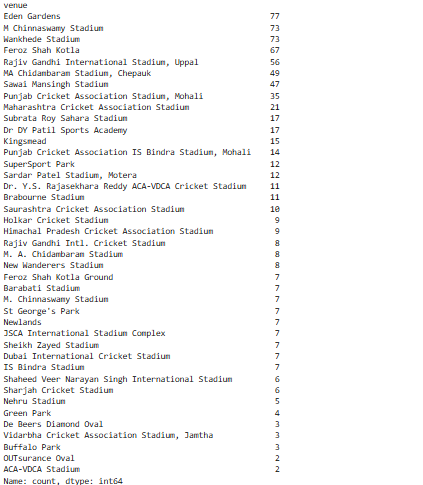
****

**53. Get the total number of matches played at each venue**

matches\_per\_venue = ipl\_data['venue'].value\_counts()

print(matches\_per\_venue)

**OUTPUT**

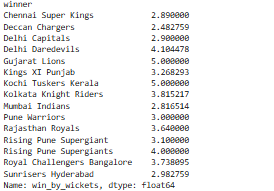
****

**54. Get the average number of wickets taken by the winning team**

average\_wickets\_winning\_team = ipl\_data.groupby('winner')['win\_by\_wickets'].mean()

print(average\_wickets\_winning\_team)

**OUTPUT**

****

**55. Get the player of the match for the match with the highest score**

highest\_score\_match = ipl\_data.loc[ipl\_data['win\_by\_runs'].idxmax()]

print(highest\_score\_match['player\_of\_match'])

**OUTPUT**

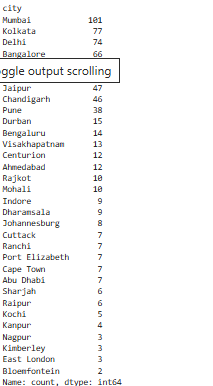
****

**56. Get the total number of matches played in each city**

matches\_per\_city = ipl\_data['city'].value\_counts()

print(matches\_per\_city)

**OUTPUT**

****

**57. Get the first match date for each team**

first\_match\_dates = ipl\_data.groupby('team1')['date'].min()

print(first\_match\_dates)

**OUTPUT**

****

**58. Get the last match date for each team**

last\_match\_dates = ipl\_data.groupby('team1')['date'].max()

print(last\_match\_dates)

**OUTPUT**

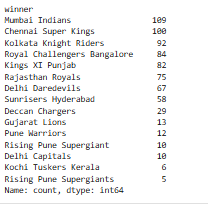
****

**59. Count the number of matches won by each team**

matches\_won = ipl\_data['winner'].value\_counts()

print(matches\_won)

**OUTPUT**

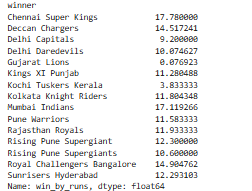
****

**60. Get the average number of runs scored by the winning team**

average\_runs\_winning\_team = ipl\_data.groupby('winner')['win\_by\_runs'].mean()

print(average\_runs\_winning\_team)

**OUTPUT**

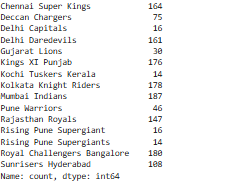
****

**61. Get the total number of matches played by each team**

matches\_played\_by\_team = ipl\_data['team1'].value\_counts() + ipl\_data['team2'].value\_counts()

print(matches\_played\_by\_team**)**

**OUTPUT**

****

**62. Get the player of the match for the most recent match**

most\_recent\_match = ipl\_data.iloc[-1]

print(most\_recent\_match['player\_of\_match'])

**OUTPUT**

****

**63. Get the total number of matches played in 2018**

matches\_2018\_count = ipl\_data[ipl\_data['season'] == 2018].shape[0]

print(matches\_2018\_count)

**OUTPUT**

****

**64. Get the total number of matches played in 2019**

matches\_2019\_count = ipl\_data[ipl\_data['season'] == 2019].shape[0]

print(matches\_2019\_count)

**OUTPUT**

****

**65. Get the total number of matches played in 2020**

matches\_2020\_count = ipl\_data[ipl\_data['season'] == 2020].shape[0]

print(matches\_2020\_count)

**OUTPUT**

****

**66. Get the total number of matches played in 2021**

matches\_2021\_count = ipl\_data[ipl\_data['season'] == 2021].shape[0]

print(matches\_2021\_count)

**OUTPUT**

****

**67. Get the total number of matches played in 2022**

matches\_2022\_count = ipl\_data[ipl\_data['season'] == 2022].shape[0]

print(matches\_2022\_count)

**OUTPUT**

****

**68. Get the total number of matches played in 2023**

matches\_2023\_count = ipl\_data[ipl\_data['season'] == 2023].shape[0]

print(matches\_2023\_count)

**OUTPUT**

****

**69. Get the team with the most player of the match awards**

most\_pom\_awards = ipl\_data['player\_of\_match'].value\_counts().idxmax()

print(most\_pom\_awards)

**OUTPUT**

****

**70. Get the total number of matches played in the city of Chennai**

matches\_chennai = ipl\_data[ipl\_data['city'] == 'Chennai'].shape[0]

print(matches\_chennai)

**OUTPUT**

****

**71. Get the total number of matches played in the city of Mumbai**

matches\_mumbai = ipl\_data[ipl\_data['city'] == 'Mumbai'].shape[0]

print(matches\_mumbai)

**OUTPUT**

****

**72. Get the total number of matches played in the city of Kolkata**

matches\_kolkata = ipl\_data[ipl\_data['city'] == 'Kolkata'].shape[0]

print(matches\_kolkata)

**OUTPUT**

****

**73. Get the total number of matches played in the city of Hyderabad**

matches\_hyderabad = ipl\_data[ipl\_data['city'] == 'Hyderabad'].shape[0]

print(matches\_hyderabad)

**OUTPUT**



**74. Get the total number of matches played in the city of Pune**

matches\_pune = ipl\_data[ipl\_data['city'] == 'Pune'].shape[0]

print(matches\_pune)

**OUTPUT**

****

**75. Get the total number of matches played in the city of Rajkot**

matches\_rajkot = ipl\_data[ipl\_data['city'] == 'Rajkot'].shape[0]

print(matches\_rajkot)

**OUTPUT**

****

**76. Get the total number of matches played in the city of Indore**

matches\_indore = ipl\_data[ipl\_data['city'] == 'Indore'].shape[0]

print(matches\_indore)

**OUTPUT**

****

**77. Get the total number of matches played in the city of Mohali**

matches\_mohali = ipl\_data[ipl\_data['city'] == 'Mohali'].shape[0]

print(matches\_mohali)

**OUTPUT**

****

**78. Get the total number of matches played in the city of Chandigarh**

matches\_chandigarh = ipl\_data[ipl\_data['city'] == 'Chandigarh'].shape[0]

print(matches\_chandigarh)

**OUTPUT**

****

**79. Get the total number of matches played in the city of Cape Town**

matches\_cape\_town = ipl\_data[ipl\_data['city'] == 'Cape Town'].shape[0]

print(matches\_cape\_town)

**OUTPUT**

****

**80. Get the total number of matches played in the city of Durban**

matches\_durban = ipl\_data[ipl\_data['city'] == 'Durban'].shape[0]

print(matches\_durban)

**OUTPUT**

****

**81. Get the total number of matches played in the city of Port Elizabeth**

matches\_port\_elizabeth = ipl\_data[ipl\_data['city'] == 'Port Elizabeth'].shape[0]

print(matches\_port\_elizabeth)

**OUTPUT**

****

**82. Get the total number of matches played in the city of Centurion**

matches\_centurion = ipl\_data[ipl\_data['city'] == 'Centurion'].shape[0]

print(matches\_centurion)

**OUTPUT**

****

**83. Get the total number of matches played in the city of East London**

matches\_east\_london = ipl\_data[ipl\_data['city'] == 'East London'].shape[0]

print(matches\_east\_london)

**OUTPUT**

****

**84. Get the total number of matches played in the city of Bloemfontein**

matches\_bloemfontein = ipl\_data[ipl\_data['city'] == 'Bloemfontein'].shape[0]

print(matches\_bloemfontein)

**OUTPUT**

**85. Get the total number of matches played in the city of Kimberley**

matches\_kimberley = ipl\_data[ipl\_data['city'] == 'Kimberley'].shape[0]

print(matches\_kimberley)

**OUTPUT**

****

**86. Get the total number of matches played in the city of Ranchi**

matches\_ranchi = ipl\_data[ipl\_data['city'] == 'Ranchi'].shape[0]

print(matches\_ranchi)

**OUTPUT**

****

**87. Get the total number of matches played in the city of Visakhapatnam**

matches\_visakhapatnam = ipl\_data[ipl\_data['city'] == 'Visakhapatnam'].shape[0]

print(matches\_visakhapatnam)

**OUTPUT**

****

**89. Get the total number of matches played in the city of Nagpur**

matches\_nagpur = ipl\_data[ipl\_data['city'] == 'Nagpur'].shape[0]

print(matches\_nagpur)

**OUTPUT**

****

**90. Get the total number of matches played in the city of Ahmedabad**

matches\_ahmedabad = ipl\_data[ipl\_data['city'] == 'Ahmedabad'].shape[0]

print(matches\_ahmedabad)

**OUTPUT**

****

**91. Get the total number of matches played in the city of Cuttack**

matches\_cuttack = ipl\_data[ipl\_data['city'] == 'Cuttack'].shape[0]

print(matches\_cuttack)

**OUTPUT**

****

**92. Get the total number of matches played in the city of Sharjah**

matches\_sharjah = ipl\_data[ipl\_data['city'] == 'Sharjah'].shape[0]

print(matches\_sharjah)

**OUTPUT**

****

**93. Get the total number of matches played in the city of Cape Town**

matches\_cape\_town = ipl\_data[ipl\_data['city'] == 'Cape Town'].shape[0]

print(matches\_cape\_town)

**OUTPUT**

****

**94. Get the total number of matches played in the city of Bloemfontein**

matches\_bloemfontein = ipl\_data[ipl\_data['city'] == 'Bloemfontein'].shape[0]

print(matches\_bloemfontein)

**OUTPUT**



**95. Get the total number of matches played in the city of East London**

matches\_east\_london = ipl\_data[ipl\_data['city'] == 'East London'].shape[0]

print(matches\_east\_london)

**OUTPUT**

****

**96. Get the total number of matches played in the city of Centurion**

matches\_centurion = ipl\_data[ipl\_data['city'] == 'Centurion'].shape[0]

print(matches\_centurion)

**OUTPUT**

****

**97. Get the total number of matches played in the city of Durban**

matches\_durban = ipl\_data[ipl\_data['city'] == 'Durban'].shape[0]

print(matches\_durban)

**98. Get the total number of matches played in the city of Johannesburg**

matches\_johannesburg = ipl\_data[ipl\_data['city'] == 'Johannesburg'].shape[0]

print(matches\_johannesburg)

**OUTPUT**

****

**99. Get the total number of matches played in the city of Bloemfontein**

matches\_bloemfontein = ipl\_data[ipl\_data['city'] == 'Bloemfontein'].shape[0]

print(matches\_bloemfontein)

**100. Get the total number of matches played in the city of Cape Town**

matches\_cape\_town = ipl\_data[ipl\_data['city'] == 'Cape Town'].shape[0]

print(matches\_cape\_town)

**OUTPUT**

****

**101.Load the IPL dataset**

ipl\_data = pd.read\_csv('ipl\_data.csv')

**102. Count the total number of wins for each team**

total\_wins = ipl\_data['winner'].value\_counts()

**103Create a bar chart**

plt.figure(figsize=(12, 6))

sns.barplot(x=total\_wins.index, y=total\_wins.values, palette='viridis')

plt.title('Total Wins by Each Team in IPL')

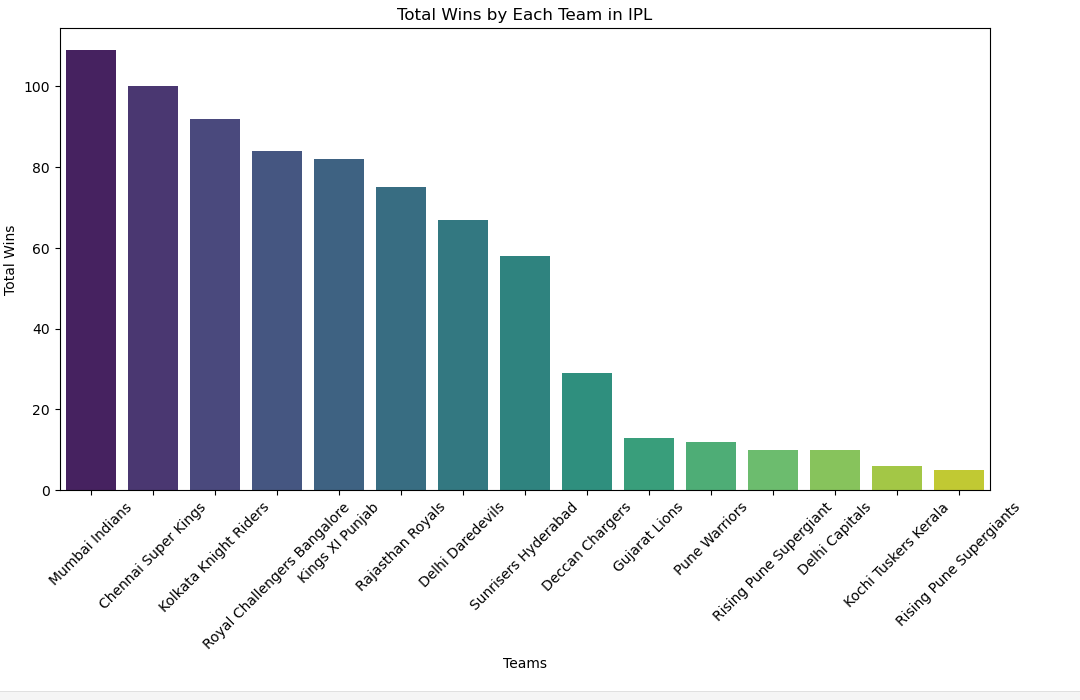
plt.xlabel('Teams')

plt.ylabel('Total Wins')

plt.xticks(rotation=45)

plt.show()

**OUTPUT**

****

**104. Count the total number of matches played by each team**

matches\_played = ipl\_data['team1'].value\_counts() + ipl\_data['team2'].value\_counts()

**105Create a pie chart**

plt.figure(figsize=(10, 10))

plt.pie(matches\_played, labels=matches\_played.index, autopct='%1.1f%%', startangle=140, colors=sns.color\_palette('pastel'))

plt.title('Proportion of Matches Played by Each Team in IPL')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

**OUTPUT**

