IPL Data Analysis and Visualization Project using Python

Introduction

Data science is the study of data to extract knowledge and insights from the data and apply knowledge and actionable insights. I work on IPL Data Analysis and Visualization Project using Python where we will explore interesting insights from the data of IPL matches like most run by a player, most wicket taken by a player, and much more from IPL season 2008-2019.

The steps demonstrated in this notebook are:

1. Loading the data
2. Familiarizing with data
3. Visualizing the data
4. Data Analysis
5. Conclusion

Importing Libraries

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt
```

IPL Dataset

Our IPL dataset contains ball by ball records from the first match played in the 2008 season till the complete 2019 season.

1.Importing IPL Dataset

We have imported the CSV dataset below with the help of pandas read_csv functions We can see the content of the dataset by using head() function.

```
matche=pd.read_csv("matches.csv")
```

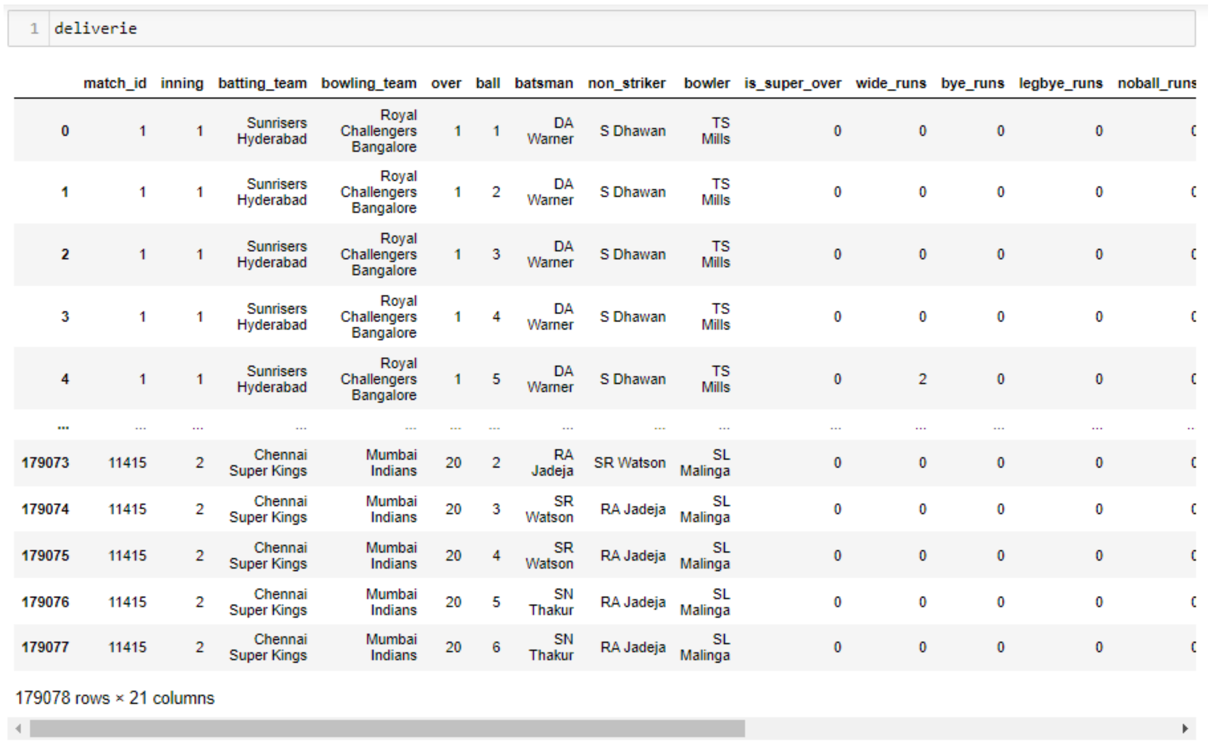
```
deliverie=pd.read_csv("deliveries.csv")
```

2.Familiarizing with Data:

Before we proceed with our Python data analysis of IPL data, we should know what columns are present in the dataset, their count, and data type. For this, we use Pandas info() function

Analysing Deliverie Dataset:-

1. Shape of dataframe
2. Listing the features of the dataset
3. Information about the dataset
4. checking for null value
5. describtion of dataset



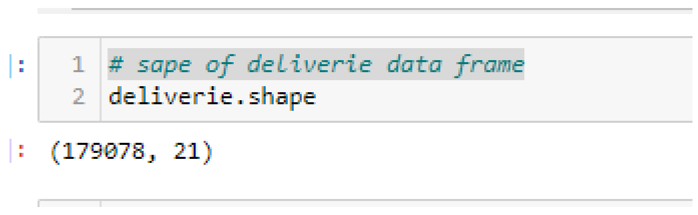
```
deliverie
```

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	0	0	0	(
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	0	0	0	(
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	0	0	0	(
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	0	0	0	(
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	2	0	0	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga	0	0	0	0	(
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga	0	0	0	0	(
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga	0	0	0	0	(
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga	0	0	0	0	(
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga	0	0	0	0	(

179078 rows × 21 columns

1. **Shape of dataframe**

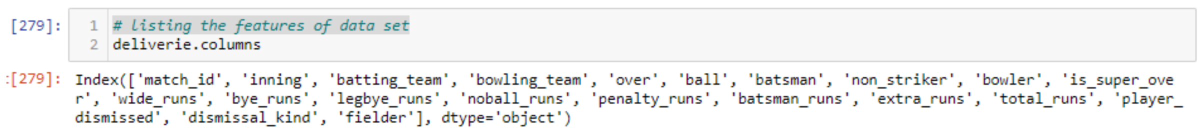
   Pandas **`shape`** are used to return shape and dimensions of data frames



```
# sape of deliverie data frame
deliverie.shape
```

```
(179078, 21)
```

2. **Listing the features of the dataset**

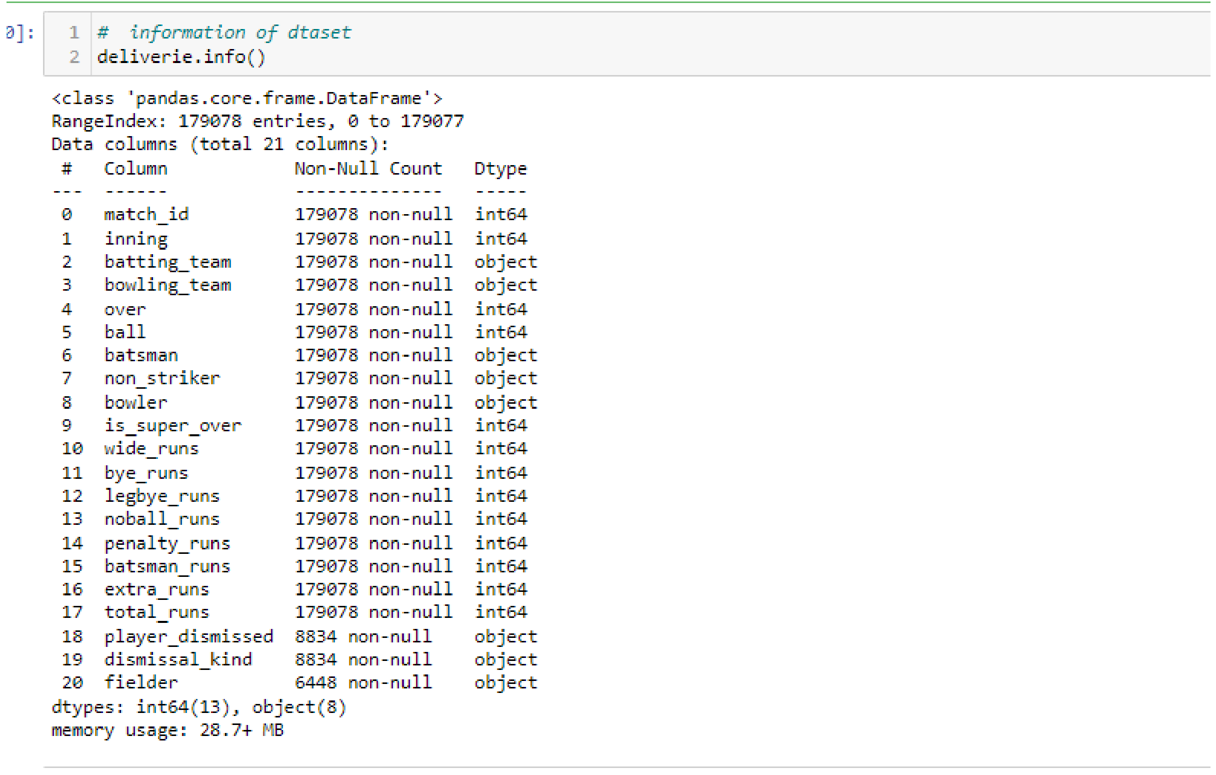


```
[279]:  # listing the features of data set
        deliverie.columns
```

```
:[279]: Index(['match_id', 'inning', 'batting_team', 'bowling_team', 'over', 'ball', 'batsman', 'non_striker', 'bowler', 'is_super_ove
        r', 'wide_runs', 'bye_runs', 'legbye_runs', 'noball_runs', 'penalty_runs', 'batsman_runs', 'extra_runs', 'total_runs', 'player_
        dismissed', 'dismissal_kind', 'fielder'], dtype='object')
```

3. **Information about the dataset**

   Pandas **dataframe.info()** function is used to get a summary of the dataframe. It comes really handy when doing exploratory analysis of the data.

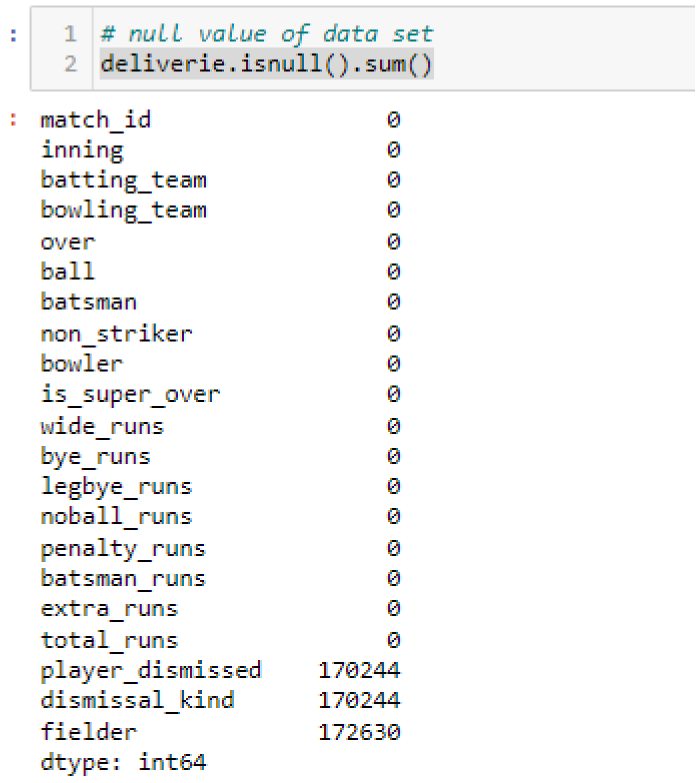


```
#  information of dtaset
deliverie.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179078 entries, 0 to 179077
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          179078 non-null  int64 
 1   inning            179078 non-null  int64 
 2   batting_team      179078 non-null  object
 3   bowling_team      179078 non-null  object
 4   over              179078 non-null  int64 
 5   ball              179078 non-null  int64 
 6   batsman           179078 non-null  object
 7   non_striker       179078 non-null  object
 8   bowler            179078 non-null  object
 9   is_super_over     179078 non-null  int64 
 10  wide_runs         179078 non-null  int64 
 11  bye_runs          179078 non-null  int64 
 12  legbye_runs       179078 non-null  int64 
 13  noball_runs       179078 non-null  int64 
 14  penalty_runs      179078 non-null  int64 
 15  batsman_runs      179078 non-null  int64 
 16  extra_runs        179078 non-null  int64 
 17  total_runs        179078 non-null  int64 
 18  player_dismissed  8834 non-null    object
 19  dismissal_kind    8834 non-null    object
 20  fielder           6448 non-null    object
dtypes: int64(13), object(8)
memory usage: 28.7+ MB
```

4. checking for null value

These function can also be used in Pandas Series in order to find null values in a series. In order to check null values in Pandas DataFrame, we use isnull () function this function return dataframe of Boolean values which are True for NaN values

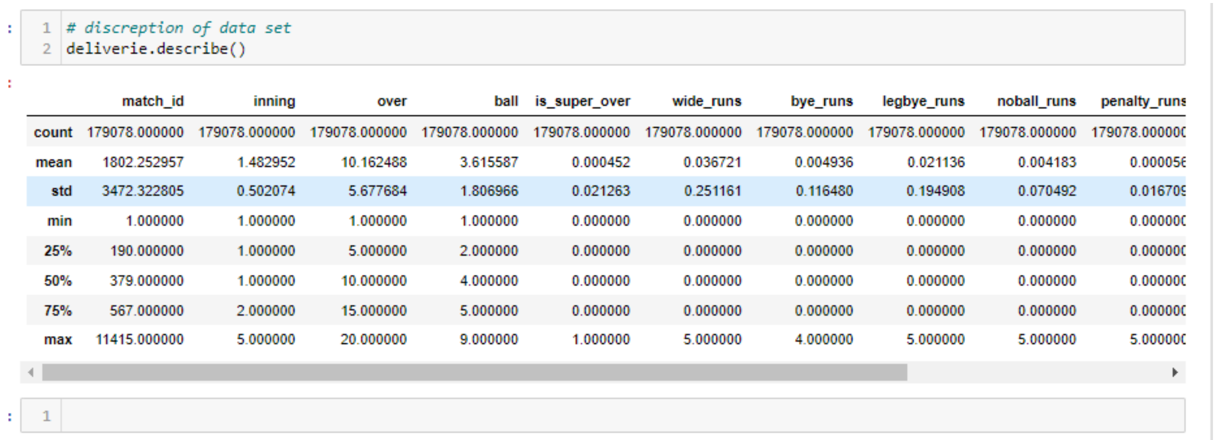


```
# null value of data set
deliverie.isnull().sum()
```

```
match_id                 0
inning                   0
batting_team             0
bowling_team             0
over                     0
ball                     0
batsman                  0
non_striker              0
bowler                   0
is_super_over            0
wide_runs                0
bye_runs                 0
legbye_runs              0
noball_runs              0
penalty_runs             0
batsman_runs             0
extra_runs               0
total_runs               0
player_dismissed    170244
dismissal_kind      170244
fielder             172630
dtype: int64
```

5. describtion of dataset

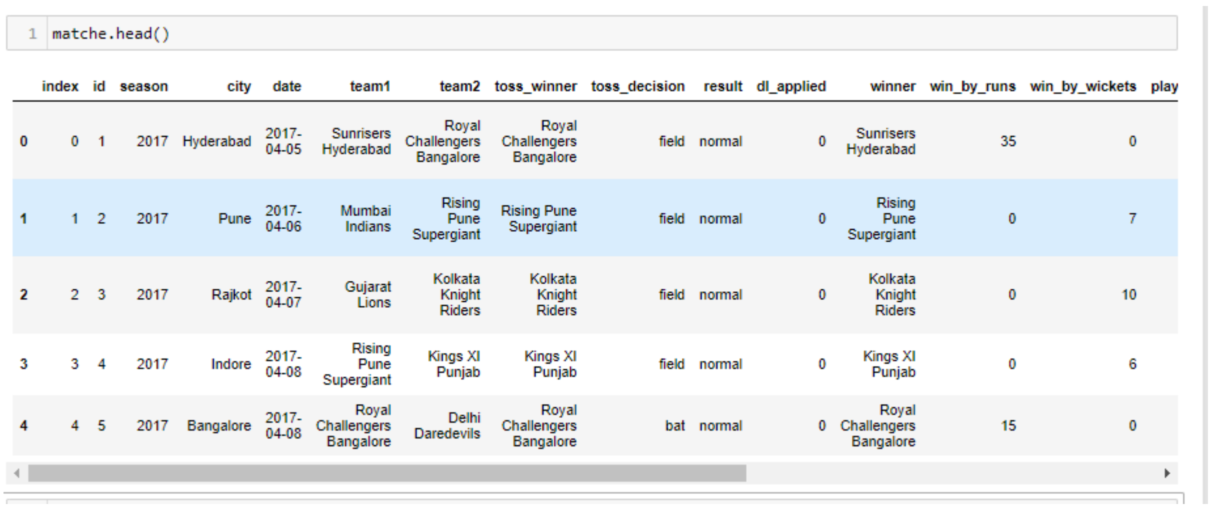
Pandas `describe()` is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. When this method is applied to a series of string



```
# discreption of data set
deliverie.describe()
```

	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs	penalty_runs
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.00000
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	0.036721	0.004936	0.021136	0.004183	0.00005
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161	0.116480	0.194908	0.070492	0.01670
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	5.000000	5.00000

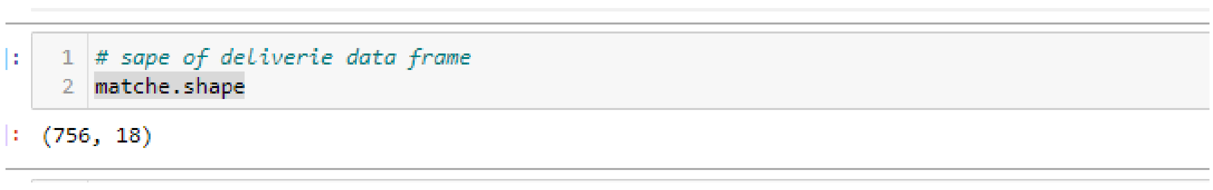
Analysing Matches Dataset:-



```
matche.head()
```

	index	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	play
0	0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	
1	1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	
2	2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	
3	3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	
4	4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	

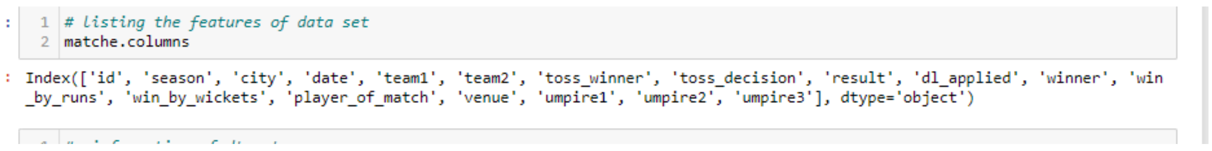
1. Shape of dataframe



```
# sape of deliverie data frame
matche.shape
```

```
(756, 18)
```

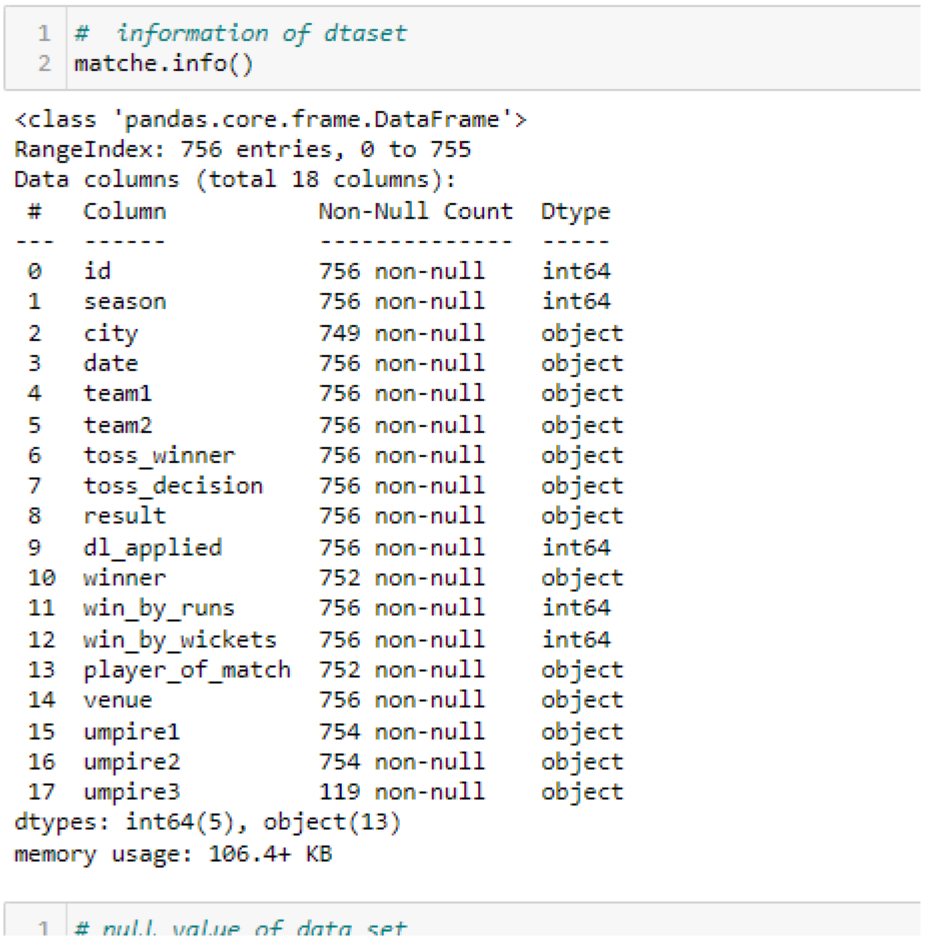
2. Listing the features of the dataset



```
# listing the features of data set
matche.columns
```

```
Index(['id', 'season', 'city', 'date', 'team1', 'team2', 'toss_winner', 'toss_decision', 'result', 'dl_applied', 'winner', 'win
_by_runs', 'win_by_wickets', 'player_of_match', 'venue', 'umpire1', 'umpire2', 'umpire3'], dtype='object')
```

3. Information about the dataset

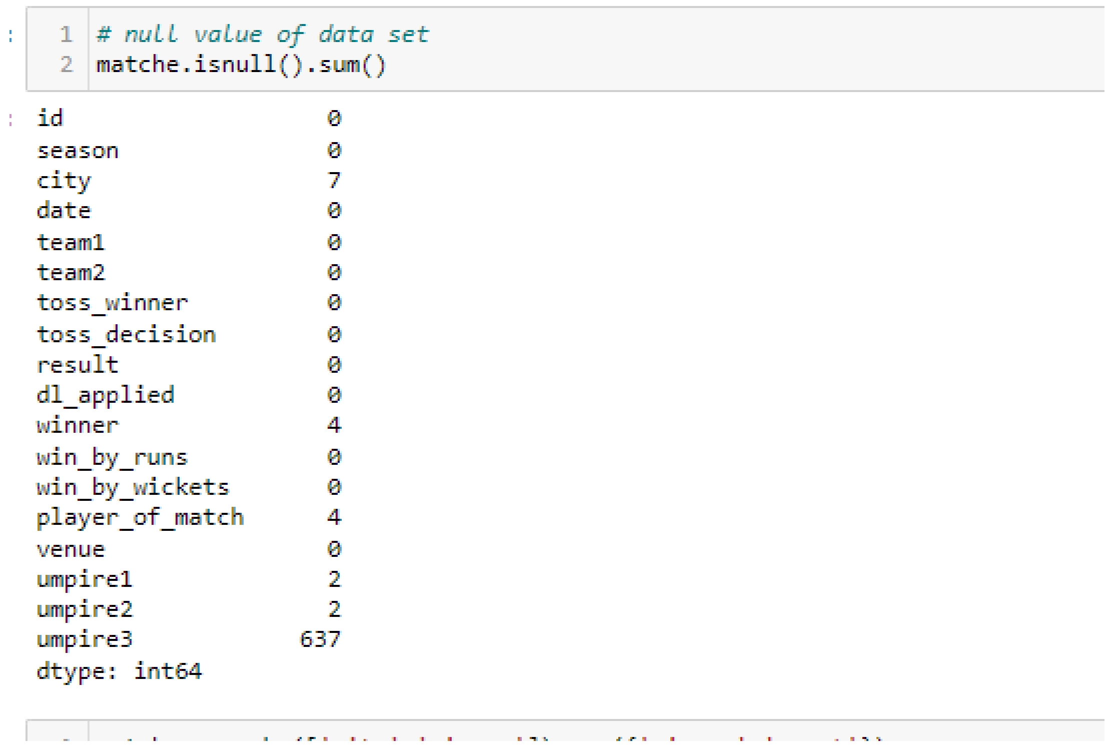


```python
#  information of dtaset
matche.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               756 non-null    int64 
 1   season           756 non-null    int64 
 2   city             749 non-null    object
 3   date             756 non-null    object
 4   team1            756 non-null    object
 5   team2            756 non-null    object
 6   toss_winner      756 non-null    object
 7   toss_decision    756 non-null    object
 8   result           756 non-null    object
 9   dl_applied       756 non-null    int64 
 10  winner           752 non-null    object
 11  win_by_runs      756 non-null    int64 
 12  win_by_wickets   756 non-null    int64 
 13  player_of_match  752 non-null    object
 14  venue            756 non-null    object
 15  umpire1          754 non-null    object
 16  umpire2          754 non-null    object
 17  umpire3          119 non-null    object
dtypes: int64(5), object(13)
memory usage: 106.4+ KB
```

```python
# null value of data set
```

4. checking for null value

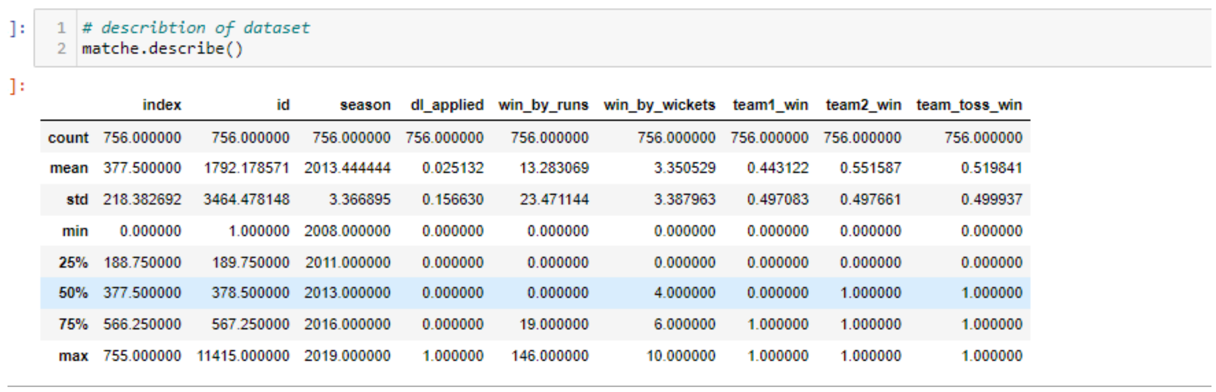


```python
# null value of data set
matche.isnull().sum()
```

```
id                   0
season               0
city                 7
date                 0
team1                0
team2                0
toss_winner          0
toss_decision        0
result               0
dl_applied           0
winner               4
win_by_runs          0
win_by_wickets       0
player_of_match      4
venue                0
umpire1              2
umpire2              2
umpire3            637
dtype: int64
```

[illegible]

5. describtion of dataset



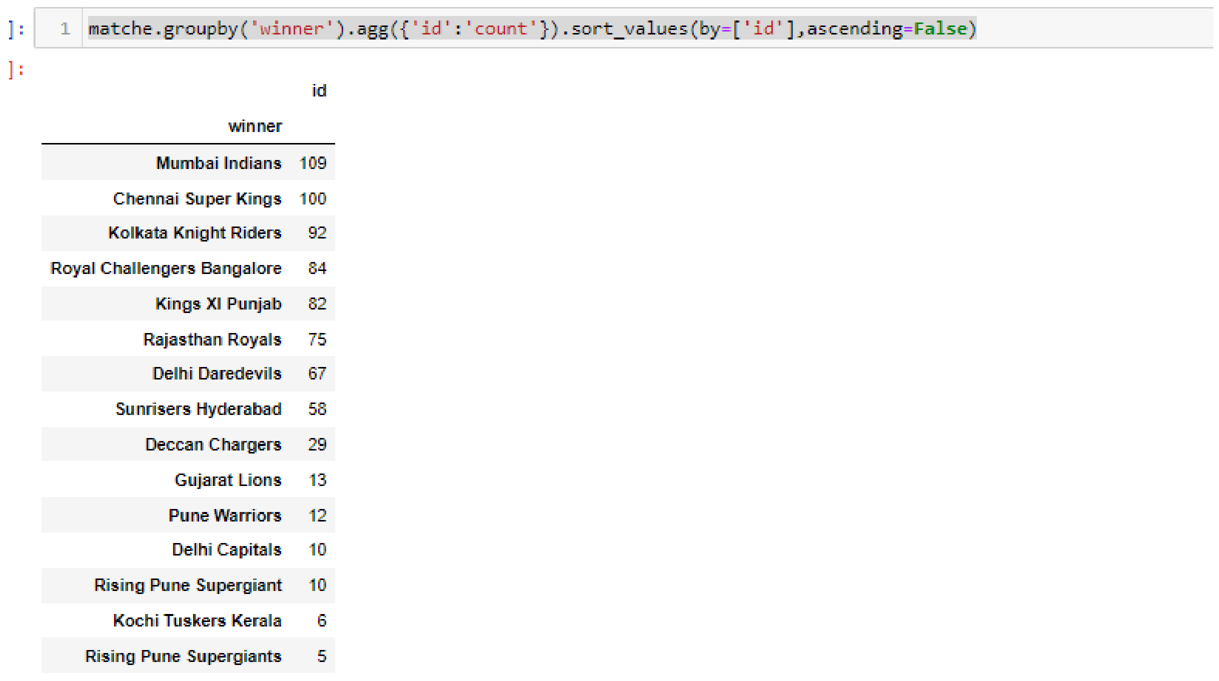
```
]: # describtion of dataset
   matche.describe()
]:
```

	index	id	season	dl_applied	win_by_runs	win_by_wickets	team1_win	team2_win	team_toss_win
count	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000	756.000000
mean	377.500000	1792.178571	2013.444444	0.025132	13.283069	3.350529	0.443122	0.551587	0.519841
std	218.382692	3464.478148	3.366895	0.156630	23.471144	3.387963	0.497083	0.497661	0.499937
min	0.000000	1.000000	2008.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	188.750000	189.750000	2011.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	377.500000	378.500000	2013.000000	0.000000	0.000000	4.000000	0.000000	1.000000	1.000000
75%	566.250000	567.250000	2016.000000	0.000000	19.000000	6.000000	1.000000	1.000000	1.000000
max	755.000000	11415.000000	2019.000000	1.000000	146.000000	10.000000	1.000000	1.000000	1.000000

FIND THE VALUE :

City in which most matches have been won	:	Mumbai
Team that has won most matches	:	Mumbai Indians
Player who has been man of the match most times	:	CH Gayle
Most frequent Umpire 1	:	HDPK Dharmasena
Most frequent Umpire 2	:	C Shamshuddin

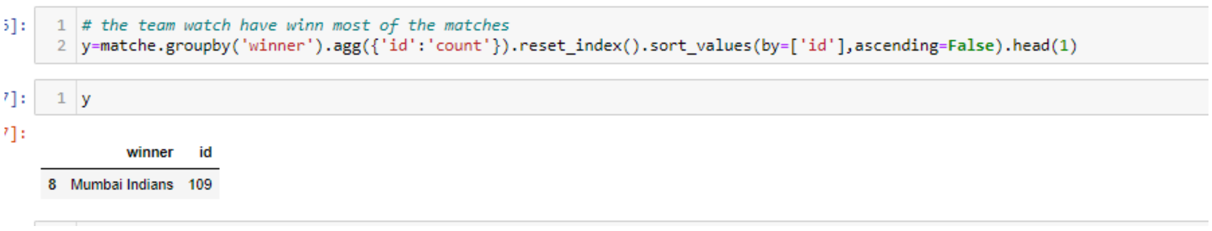
City in which most matches have been won



```
]: matche.groupby('winner').agg({'id':'count'}).sort_values(by=['id'],ascending=False)
]:
```

	id
winner	
Mumbai Indians	109
Chennai Super Kings	100
Kolkata Knight Riders	92
Royal Challengers Bangalore	84
Kings XI Punjab	82
Rajasthan Royals	75
Delhi Daredevils	67
Sunrisers Hyderabad	58
Deccan Chargers	29
Gujarat Lions	13
Pune Warriors	12
Delhi Capitals	10
Rising Pune Supergiant	10
Kochi Tuskers Kerala	6
Rising Pune Supergiants	5

Team that has won most matches

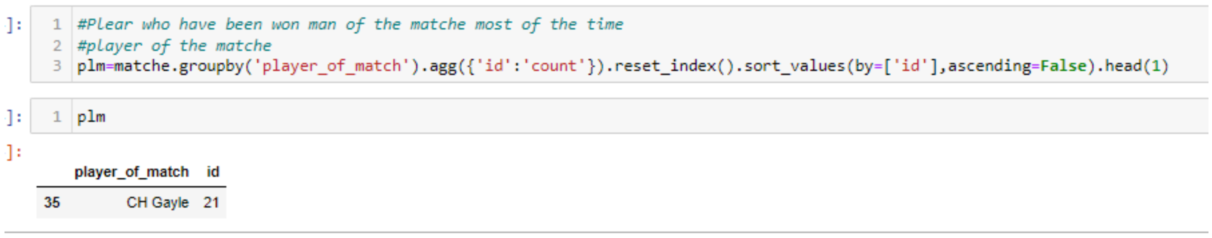


```
# the team watch have winn most of the matches
y=matche.groupby('winner').agg({'id':'count'}).reset_index().sort_values(by=['id'],ascending=False).head(1)
```

```
y
```

	winner	id
8	Mumbai Indians	109

Player who has been man of the match most times:

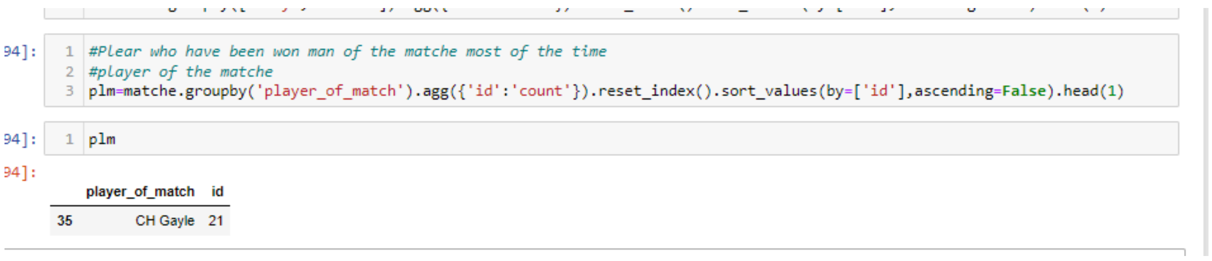


```
#Plear who have been won man of the matche most of the time
#player of the matche
plm=matche.groupby('player_of_match').agg({'id':'count'}).reset_index().sort_values(by=['id'],ascending=False).head(1)
```

```
plm
```

	player_of_match	id
35	CH Gayle	21

Player who has been man of the match most

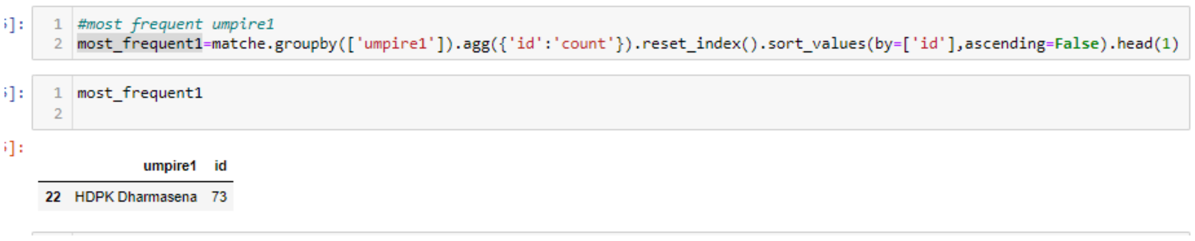


```
#Plear who have been won man of the matche most of the time
#player of the matche
plm=matche.groupby('player_of_match').agg({'id':'count'}).reset_index().sort_values(by=['id'],ascending=False).head(1)
```

```
plm
```

	player_of_match	id
35	CH Gayle	21

Most frequent Umpire 1

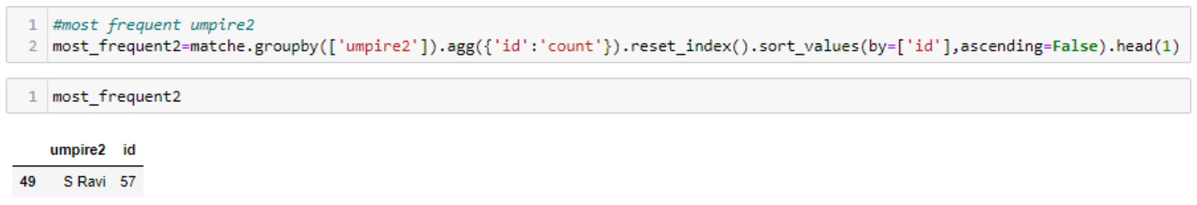


```
#most frequent umpire1
most_frequent1=matche.groupby(['umpire1']).agg({'id':'count'}).reset_index().sort_values(by=['id'],ascending=False).head(1)
```

```
most_frequent1
```

	umpire1	id
22	HDPK Dharmasena	73

Most frequent Umpire 2

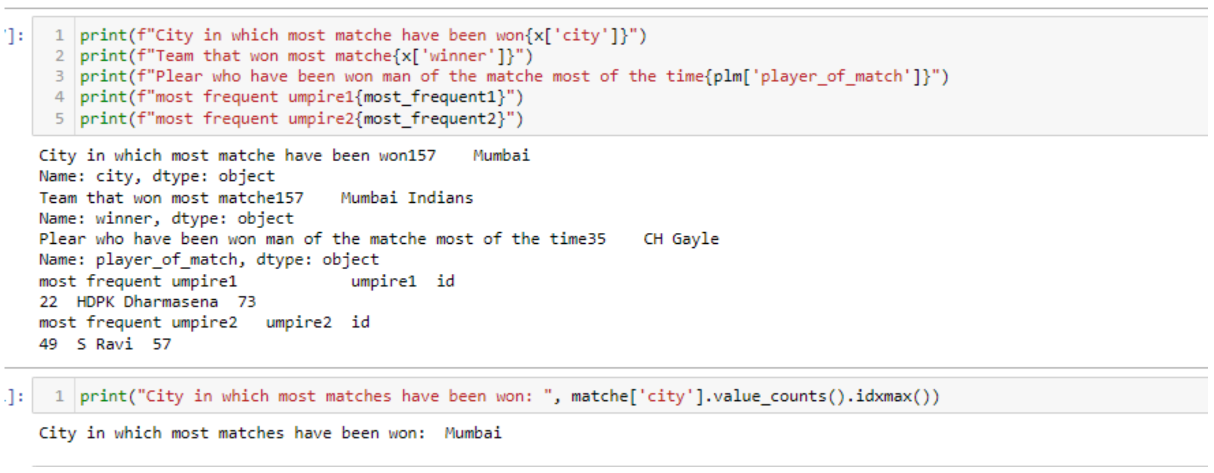


```
#most frequent umpire2
most_frequent2=matche.groupby(['umpire2']).agg({'id':'count'}).reset_index().sort_values(by=['id'],ascending=False).head(1)
```

```
most_frequent2
```

	umpire2	id
49	S Ravi	57

over all o/p



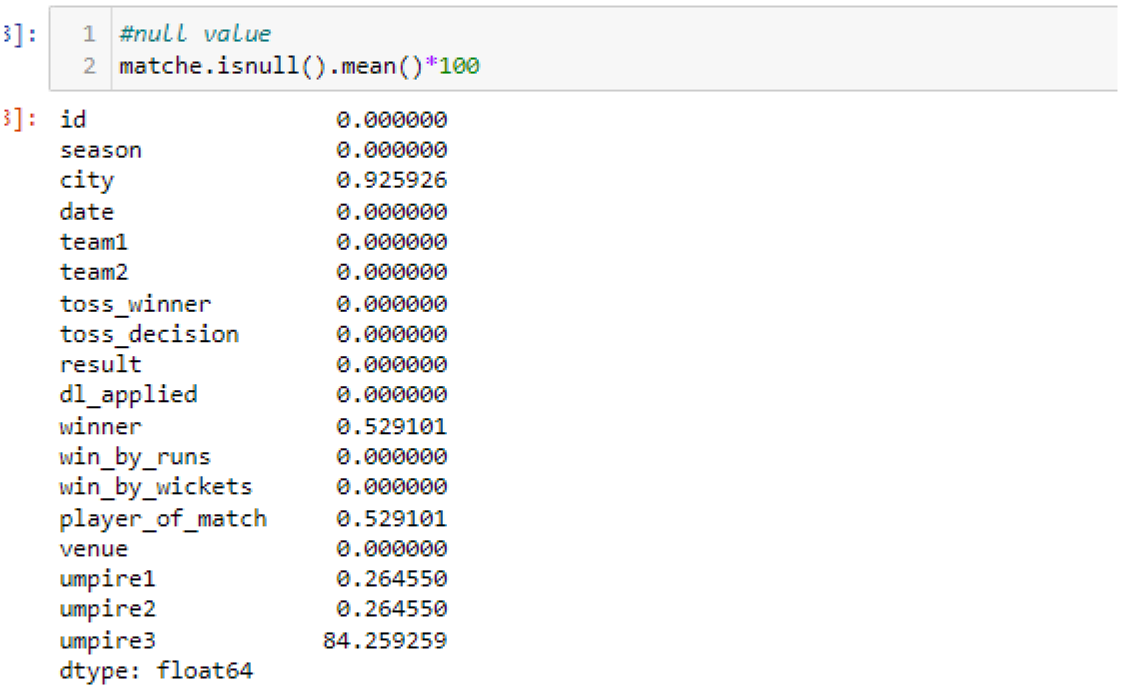
```
print(f"City in which most matche have been won{x['city']}")
print(f"Team that won most matche{x['winner']}")
print(f"Plear who have been won man of the matche most of the time{plm['player_of_match']}")
print(f"most frequent umpire1{most_frequent1}")
print(f"most frequent umpire2{most_frequent2}")
```

```
City in which most matche have been won157    Mumbai
Name: city, dtype: object
Team that won most matche157    Mumbai Indians
Name: winner, dtype: object
Plear who have been won man of the matche most of the time35    CH Gayle
Name: player_of_match, dtype: object
most frequent umpire1            umpire1  id
22  HDPK Dharmasena  73
most frequent umpire2   umpire2  id
49  S Ravi  57
```

```
print("City in which most matches have been won: ", matche['city'].value_counts().idxmax())
```

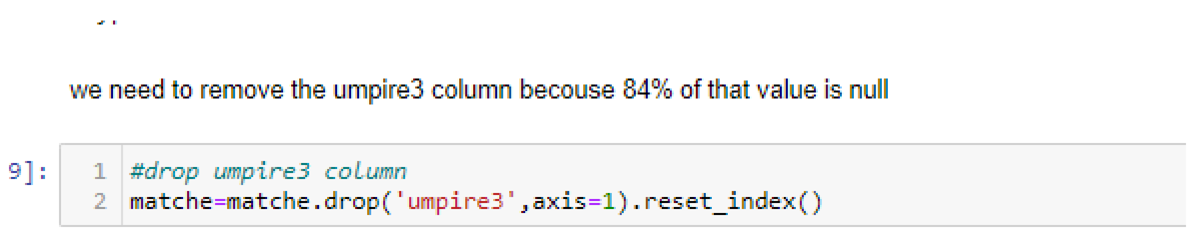
```
City in which most matches have been won:  Mumbai
```

fill null values



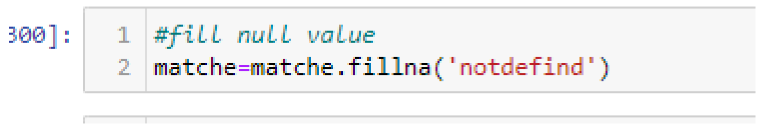
```
#null value
matche.isnull().mean()*100
```

```
id                  0.000000
season              0.000000
city                0.925926
date                0.000000
team1               0.000000
team2               0.000000
toss_winner         0.000000
toss_decision       0.000000
result              0.000000
dl_applied          0.000000
winner              0.529101
win_by_runs         0.000000
win_by_wickets      0.000000
player_of_match     0.529101
venue               0.000000
umpire1             0.264550
umpire2             0.264550
umpire3            84.259259
dtype: float64
```



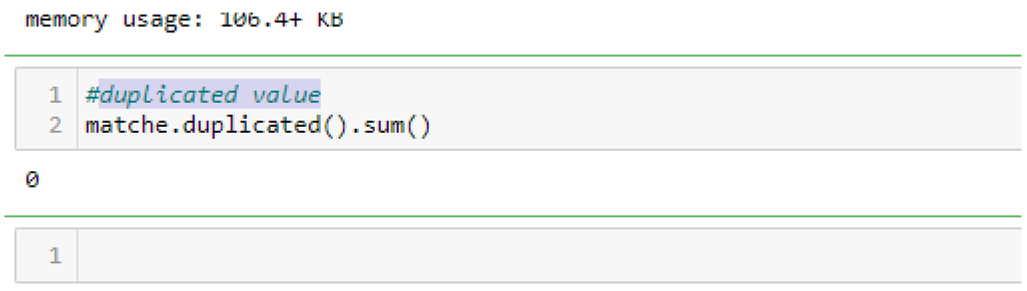
we need to remove the umpire3 column becouse 84% of that value is null

```
#drop umpire3 column
matche=matche.drop('umpire3',axis=1).reset_index()
```



```
#fill null value
matche=matche.fillna('notdefind')
```

duplicated value



```
memory usage: 106.4+ KB
```

```
#duplicated value
matche.duplicated().sum()
```

```
0
```

3.Visualizing the data:

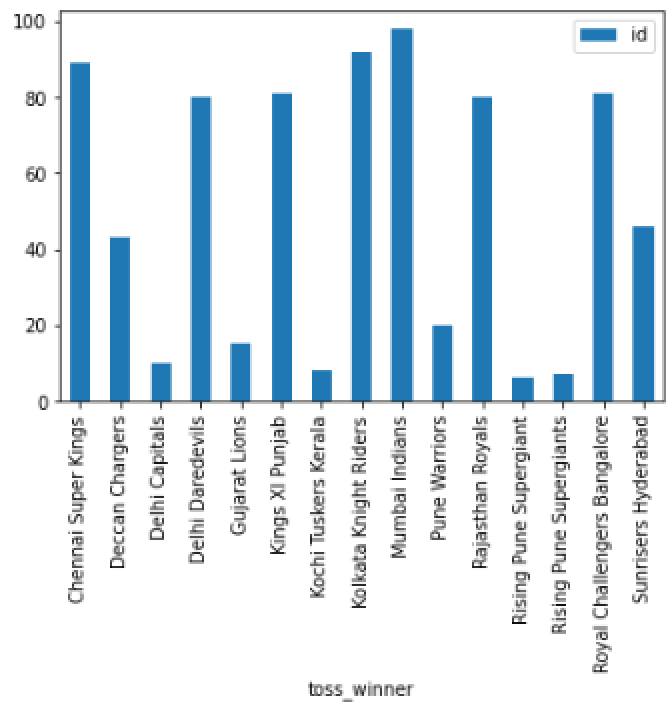
Few plots and graphs are displayed to find how the data is distributed and the how features are realated to each other

Finding the top team of the players

```python
tos_team=matche.groupby('toss_winner').agg({'id':'count'})
```

```python
tos_team.plot(kind='bar')
```

```
<AxesSubplot:xlabel='toss_winner'>
```

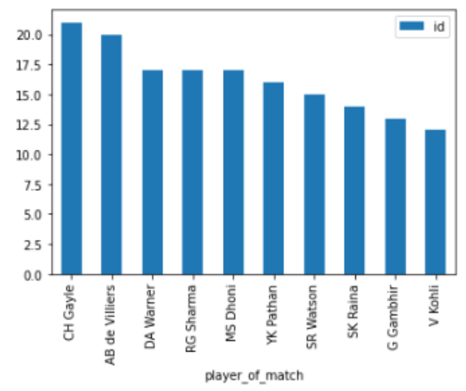


Player of the matche

```python
playyerofmatche=matche.groupby('player_of_match').agg({'id':'count'}).sort_values(by=['id'],ascending=False).head(10)
```

```python
playyerofmatche.plot(kind='bar')
```

```
<AxesSubplot:xlabel='player_of_match'>
```



Facotors affecting the Victory

```
matche.corr()
```

	index	id	season	dl_applied	win_by_runs	win_by_wickets	team1_win	team2_win
index	1.000000	0.668512	0.690898	0.012101	-0.032690	-0.019528	-0.023140	0.015203
id	0.668512	1.000000	0.668304	-0.011658	-0.039403	-0.012239	-0.022899	0.018579
season	0.690898	0.668304	1.000000	-0.001116	-0.037529	-0.009379	-0.027611	0.022660
dl_applied	0.012101	-0.011658	-0.001116	1.000000	-0.016349	-0.011631	-0.058168	0.059809
win_by_runs	-0.032690	-0.039403	-0.037529	-0.016349	1.000000	-0.560420	0.625426	-0.618675
win_by_wickets	-0.019528	-0.012239	-0.009379	-0.011631	-0.560420	1.000000	-0.882762	0.892265
team1_win	-0.023140	-0.022899	-0.027611	-0.058168	0.625426	-0.882762	1.000000	-0.989349
team2_win	0.015203	0.018579	0.022660	0.059809	-0.618675	0.892265	-0.989349	1.000000

how many season will be according to this data set

```
matche['season'].unique()
```

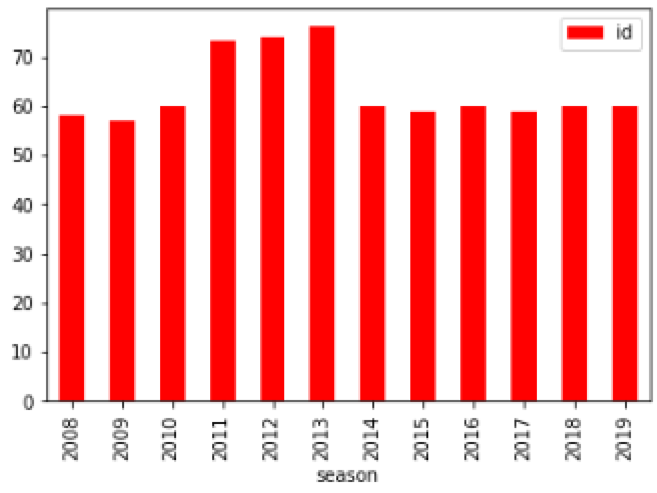
```
array([2017, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2018,
       2019], dtype=int64)
```

How meny match was done in b/w 2008 to 2019 and also plot the groph b/w

```
kk=matche.groupby('season').agg({'id':'count'})
```

```
kk.plot(kind='bar',color='red')
```

```
<AxesSubplot:xlabel='season'>
```



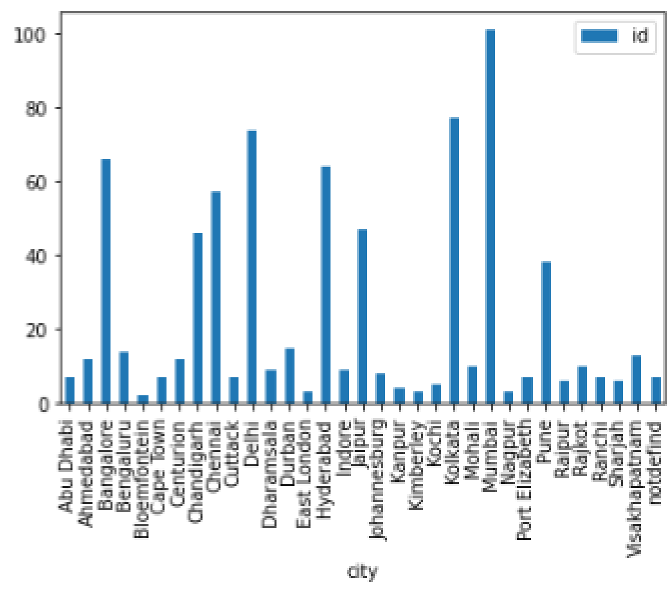
Also find the which city have done most of the matches

Also find the which city have done most of the matche

```python
k=matche.groupby('city').agg({'id':'count',})
```

```python
k.plot(kind='bar')
```

```
<AxesSubplot:xlabel='city'>
```



How meny match was done in b/w 2008 to 2019

```python
matche['id'].count()
```

```
756
```

Matches played with each team and how much much win and lose

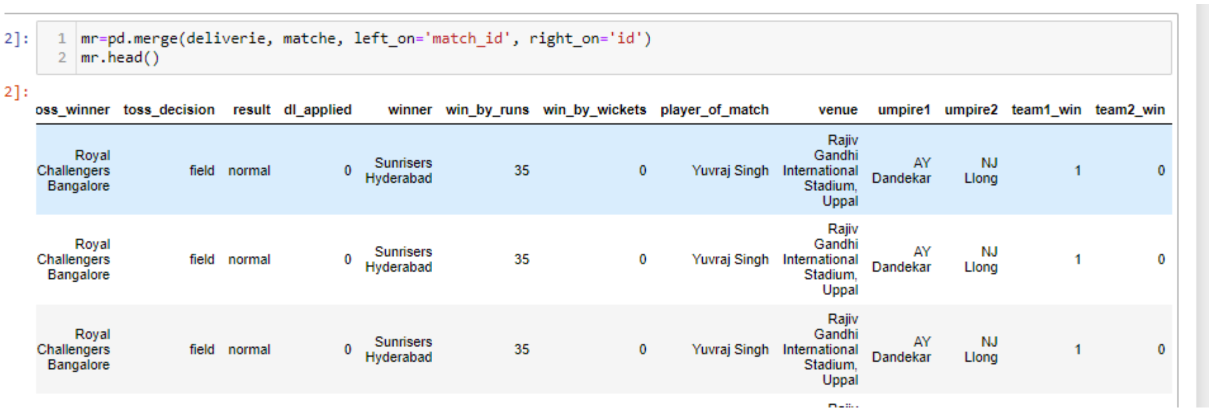
```python
matche['team1_win']=np.where(matche['team1']==matche['winner'],1,0)
matche['team2_win']=np.where(matche['team2']==matche['winner'],1,0)
```

```python
matche.groupby(['team1','team2']).agg({'id':'count','team1_win':'sum','team2_win':'sum'})
```

team1	team2	id	team1_win	team2_win
Chennai Super Kings	Deccan Chargers	8	5	3
	Delhi Capitals	1	1	0
	Delhi Daredevils	12	8	4
	Kings XI Punjab	11	7	4
	Kochi Tuskers Kerala	2	1	1

4. Data analysis

Marging the two datasets into a new dataset and Reading it(join on match-id)

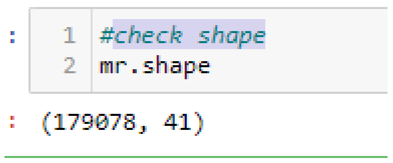


```
2]: mr=pd.merge(deliverie, matche, left_on='match_id', right_on='id')
    mr.head()
```

2]:

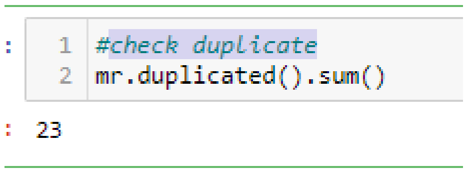
oss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	team1_win	team2_win
Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	1	0
Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	1	0
Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	1	0

check shape:



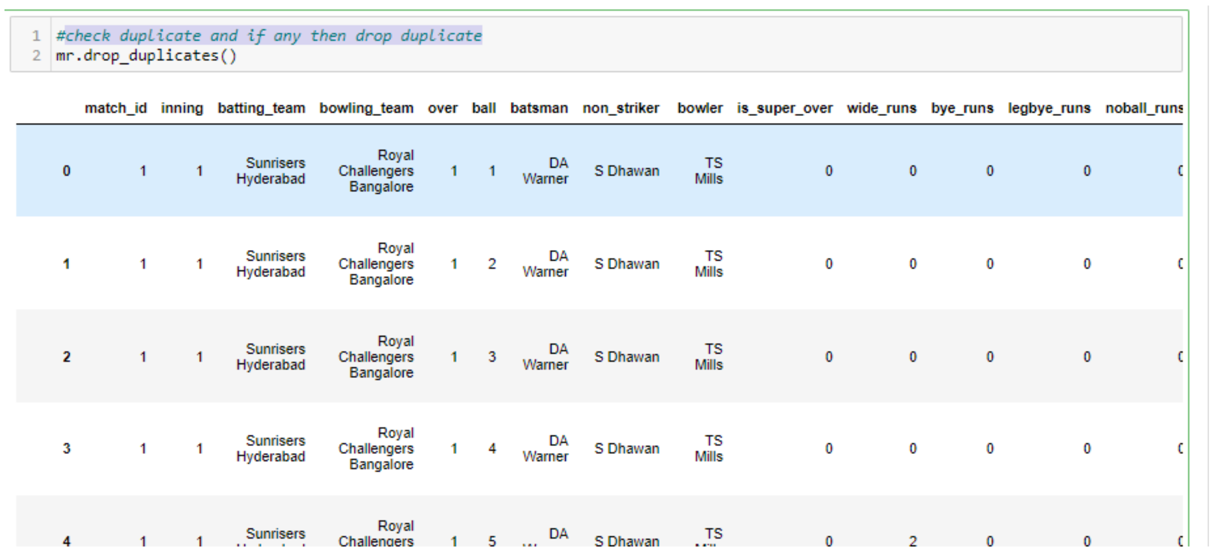
```
: #check shape
  mr.shape

: (179078, 41)
```

Check duplicate:



```
: #check duplicate
  mr.duplicated().sum()

: 23
```

check duplicate and if any then drop duplicate:

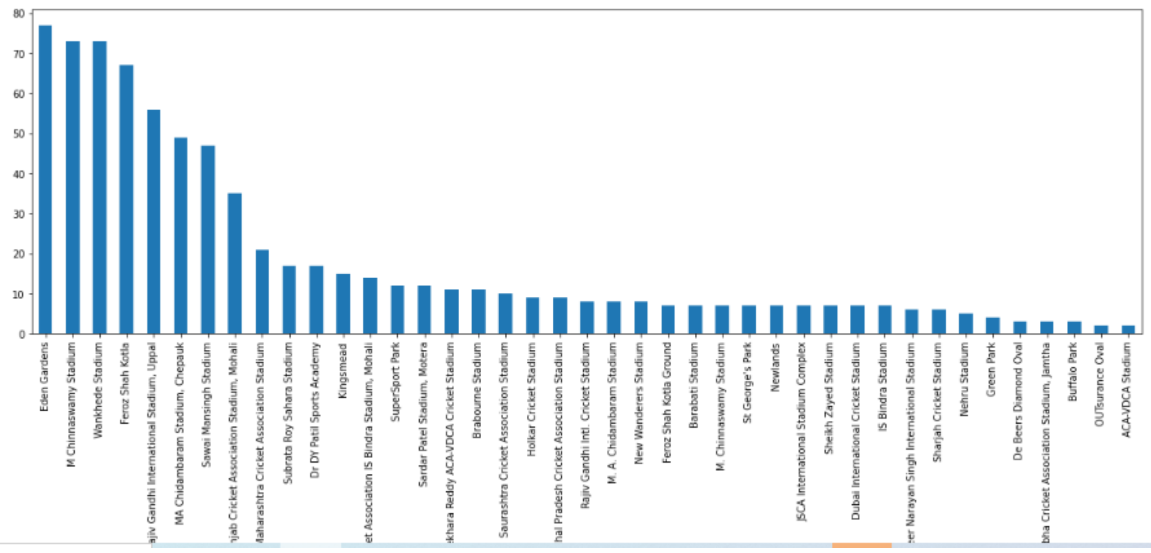


```
#check duplicate and if any then drop duplicate
mr.drop_duplicates()
```

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	wide_runs	bye_runs	legbye_runs	noball_runs
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	0	0	0	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	0	0	0	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	0	0	0	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	0	0	0	
4	1	1	Sunrisers	Royal Challengers	1	5	DA	S Dhawan	TS	0	2	0	0	

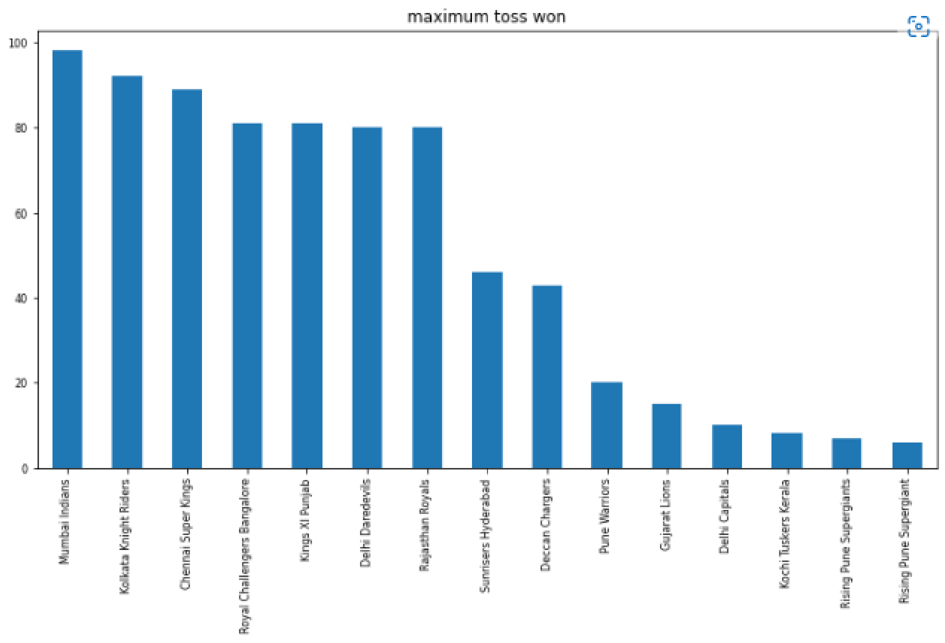
Number of matches played in each stadium:

```
]:  machese_played_in_each_stadium=matche['venue'].value_counts()


]:  machese_played_in_each_stadium.plot(kind='bar',figsize=(20,6))

]:  <AxesSubplot:>
```



Max toss won:

```
:  matche['toss_winner'].value_counts().plot(kind='bar',figsize=(12,6),title='maximum toss won',fontsize=(8))

:  <AxesSubplot:title={'center':'maximum toss won'}>
```

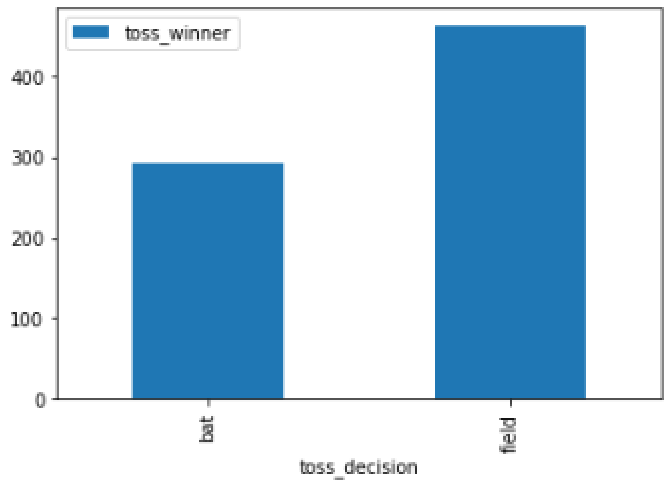


Deciding Whether to Bat or Field After Winning the Toss

```
t=matche.groupby('toss_decision').agg({'toss_winner':'count'})
```

```
t.plot(kind='bar')
```

```
<AxesSubplot:xlabel='toss_decision'>
```

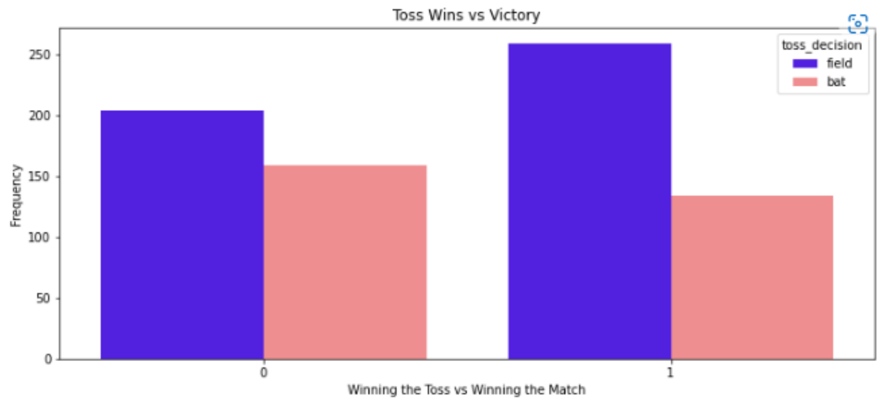


Relation between Winning toss and victory

```
matche['team_toss_win']=np.where((matche.toss_winner==matche.winner),1,0)
plt.figure(figsize=(12,5))
sns.countplot('team_toss_win', data=matche, hue='toss_decision', palette='gnuplot2')
plt.xlabel("Winning the Toss vs Winning the Match")
plt.ylabel("Frequency")
plt.title("Toss Wins vs Victory")
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

```
Text(0.5, 1.0, 'Toss Wins vs Victory')
```



Batsmen overview

```
: str_rate=mr.groupby('batsman').agg({'ball':'count','total_runs':'sum'}).sort_values(by='total_runs',ascending=False)
```

```
: str_rate['strikerate']=(str_rate['total_runs']/str_rate['ball'])*100
```

```
: str_rate
```

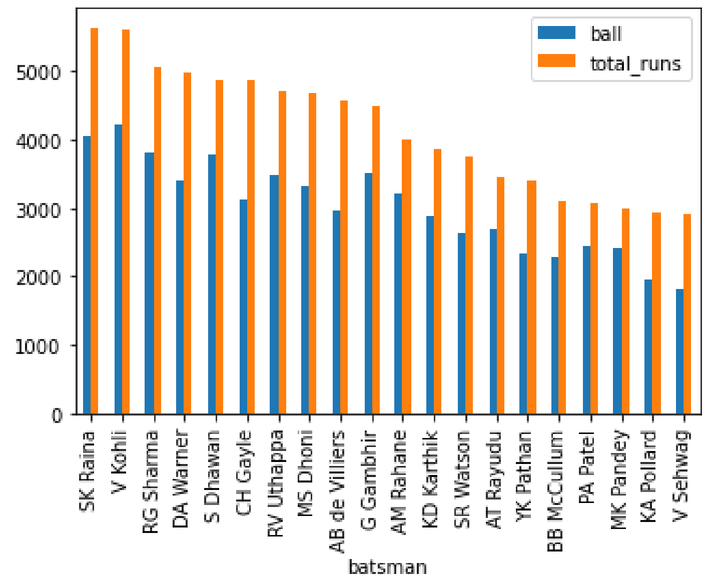
:

	ball	total_runs	strikerate
batsman			
SK Raina	4044	5651	139.737883
V Kohli	4211	5616	133.364996
RG Sharma	3816	5057	132.520964
DA Warner	3398	4975	146.409653
S Dhawan	3776	4876	129.131356
...	...	...	...
J Denly	1	0	0.000000
V Pratap Singh	1	0	0.000000
Abdur Razzak	2	0	0.000000
Sunny Gupta	1	0	0.000000
ND Doshi	13	0	0.000000

516 rows × 3 columns

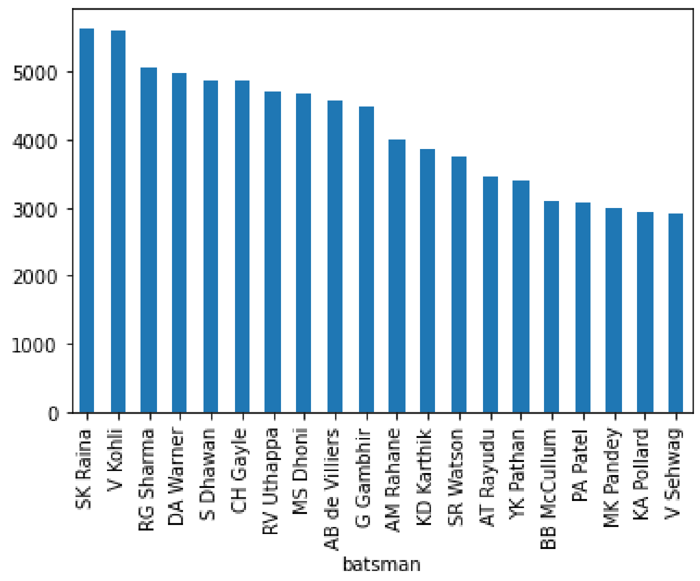
```
[83]: ku.plot(kind='bar')
```

```
[83]: <AxesSubplot:xlabel='batsman'>
```

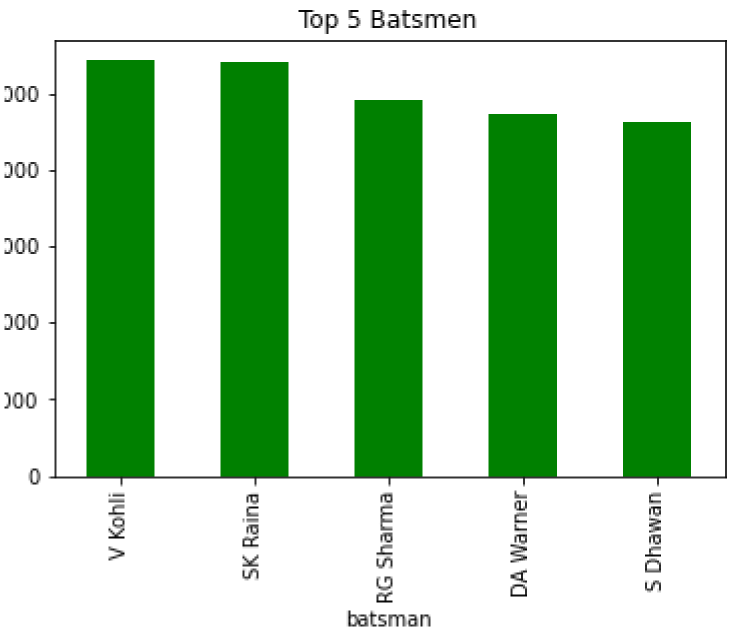


Total runs by each batsmen

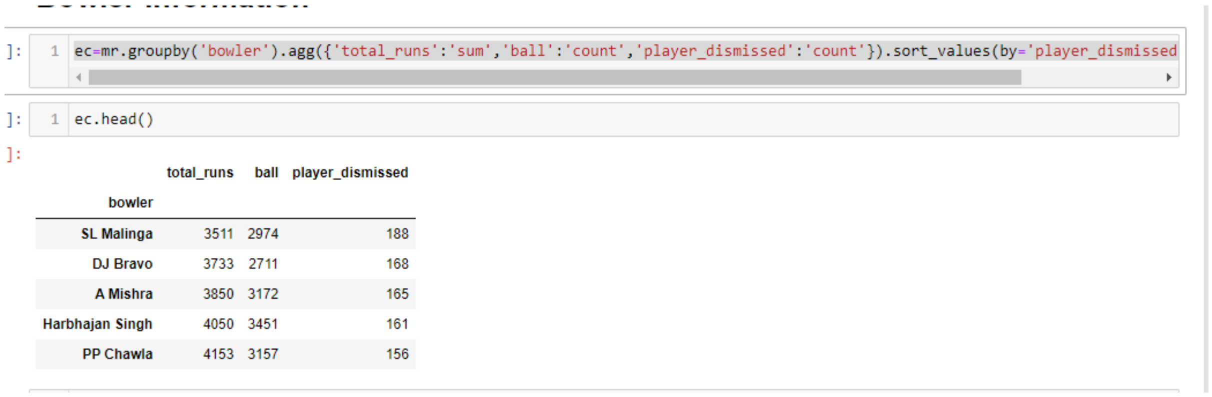
```
2]: <AxesSubplot:xlabel='batsman'>
```



Top 5 Batsmen



Bowler information

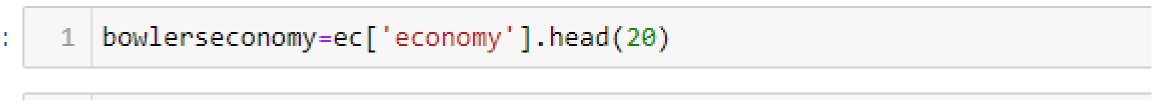


```python
ec=mr.groupby('bowler').agg({'total_runs':'sum','ball':'count','player_dismissed':'count'}).sort_values(by='player_dismissed
```

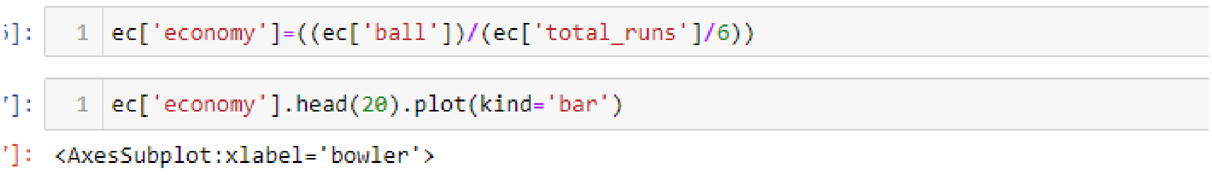
```python
ec.head()
```

bowler	total_runs	ball	player_dismissed
SL Malinga	3511	2974	188
DJ Bravo	3733	2711	168
A Mishra	3850	3172	165
Harbhajan Singh	4050	3451	161
PP Chawla	4153	3157	156

top 20 boler with economy:



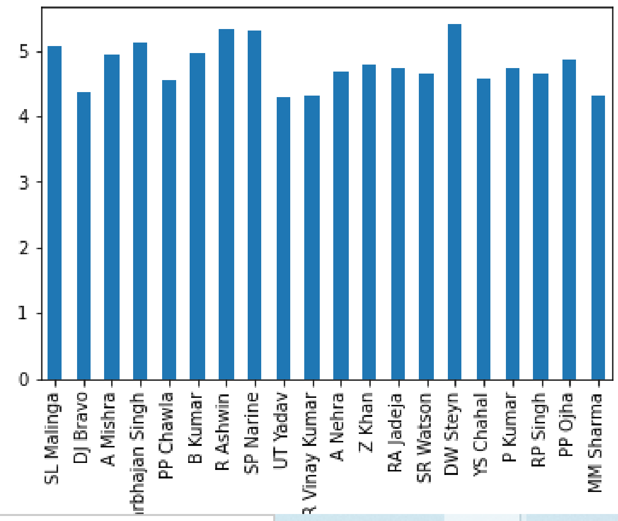
```python
bowlerseconomy=ec['economy'].head(20)
```



```python
ec['economy']=((ec['ball'])/(ec['total_runs']/6))
```

```python
ec['economy'].head(20).plot(kind='bar')
```

```
<AxesSubplot:xlabel='bowler'>
```

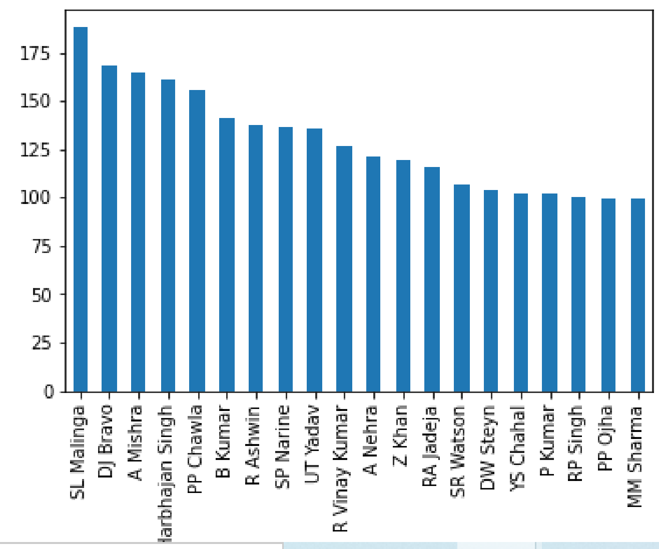


Wickets taken by a bowle

```
Wickets_taken_by_a_bowle=ec['player_dismissed'].head(20)
```

```
Wickets_taken_by_a_bowle.plot(kind='bar')
```

```
<AxesSubplot:xlabel='bowler'>
```

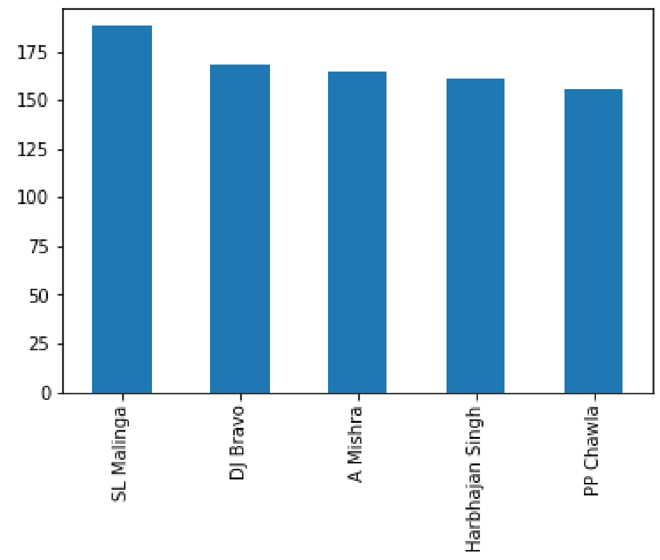


Top 5 Bowlers

```
Top_5_Bowlers=ec['player_dismissed'].head(5)
```

```
Top_5_Bowlers.plot(kind='bar')
```

```
<AxesSubplot:xlabel='bowler'>
```



5.Conclusion

Let’s summarize the important observations we made during Exploratory Data Analysis:

Mumbai Indians is the most successful team in IPL.

Mumbai Indians has won the most number of toss.

The Mumbai city has hosted the most number of IPL matches.

Chris Gayle has won the maximum number of player of the match title.

Winning toss gives a slight edge(52% probability of winning) against the opponents.

Five Indian players have figured in the top ten IPL players list.

etc.