

Exploratory Data Analysis on IPL Dataset



In [2]:

```
1  ## Importing Library
2
3  import numpy as np
4  import pandas as pd
5  import matplotlib.pyplot as plt
6  %matplotlib inline
```

In [2]:

```
1  pd.set_option('display.max_rows', 700)
2  pd.set_option('display.max_columns', 500)
3  pd.set_option('display.width', 1000)
```

In [7]:

```
1 ## Importing Dataset
2 ipl = pd.read_csv('matches.csv')
```

In [8]:

```
1 ## Checking ALL Information Related with Dataset
2 ipl.info()
```

```
<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
```

In [5]:

```
1 ## Checking Top 5 Rows
2 ipl.head(5)
```

Out[5]:

am2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs
oyal gers alore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35
sing Pune giant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0
kata ight ders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0
s XI njab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0
delhi avils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15

In [9]:

```
1 ## Checking Rows and Columns
2 ipl.shape
```

Out[9]:

(636, 18)

In [7]:

```
1 ## Changing Datatype from Integer to Category
2 ipl['season']=ipl['season'].astype('category')
```

In [8]:

```
1 ## Checking ALL Information Related with Dataset
2 ipl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
id                636 non-null int64
season            636 non-null category
city              629 non-null object
date              636 non-null object
team1             636 non-null object
team2             636 non-null object
toss_winner       636 non-null object
toss_decision     636 non-null object
result           636 non-null object
dl_applied        636 non-null int64
winner            633 non-null object
win_by_runs       636 non-null int64
win_by_wickets    636 non-null int64
player_of_match   633 non-null object
venue             636 non-null object
umpire1           635 non-null object
umpire2           635 non-null object
umpire3           0 non-null float64
dtypes: category(1), float64(1), int64(4), object(12)
memory usage: 85.6+ KB
```

Q) find the venue in which highest number of matches were held

In [9]:

```
1 venue=ipl.groupby('venue')
2 venue.size().sort_values().reset_index().tail(1)
```

Out[9]:

	venue	0
34	M Chinnaswamy Stadium	66

Q) Find the team winning most number of matches since 2008

In [10]:

```
1 ipl['winner'].value_counts().sort_values().tail(1)
```

Out[10]:

```
Mumbai Indians    92  
Name: winner, dtype: int64
```

Q) Find the team winning least number of matches

In [11]:

```
1 ipl['winner'].value_counts().sort_values().head(1).index[0]
```

Out[11]:

```
'Rising Pune Supergiants'
```

Q) Find the team who played most number of matches. note down winning doesnt matter

In [12]:

```
1 (ipl['team2'].value_counts() + ipl['team1'].value_counts()).sort_values
```

Out[12]:

```
Mumbai Indians    157  
dtype: int64
```

Q) List out all team names since 2008

In [13]:

```
1 (ipl['team2'].value_counts() + ipl['team1'].value_counts()).drop_duplic
```

Out[13]:

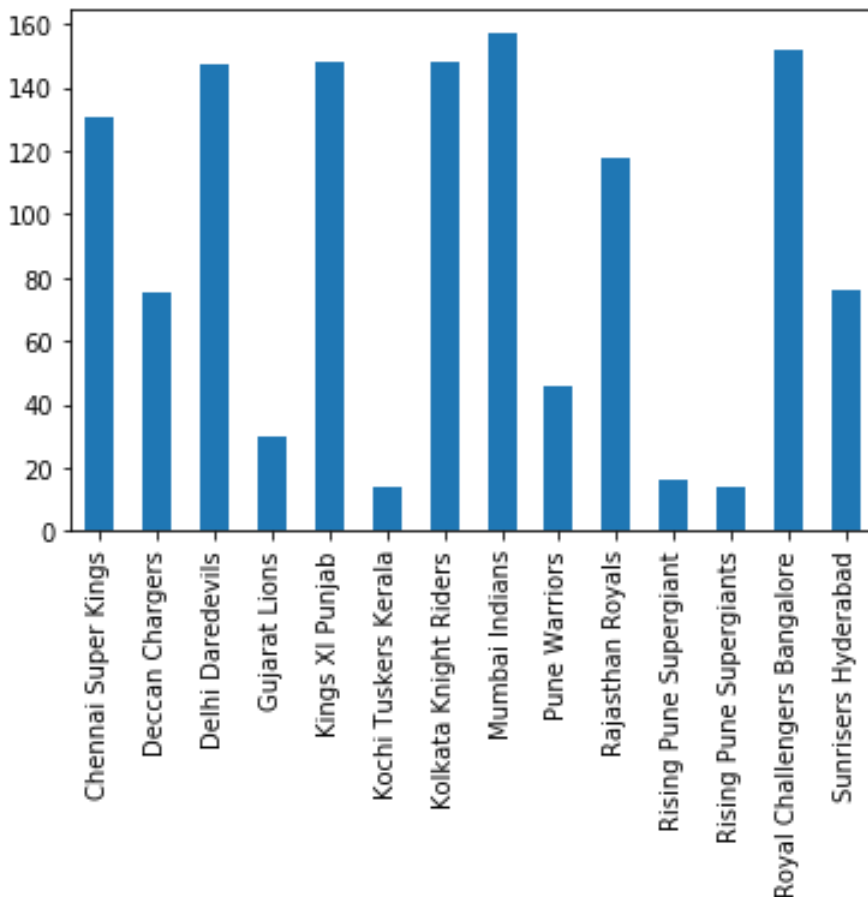
```
Index(['Chennai Super Kings', 'Deccan Chargers', 'Delhi Dared  
evils', 'Gujarat Lions', 'Kings XI Punjab', 'Kochi Tuskers Ke  
rala', 'Mumbai Indians', 'Pune Warriors', 'Rajasthan Royals',  
'Rising Pune Supergiant', 'Royal Challengers Bangalore', 'Sun  
risers Hyderabad'], dtype='object')
```

In [14]:

```
1 #plot bar graph for the teams and their number of matches played  
2 (ipl['team2'].value_counts() + ipl['team1'].value_counts()).plot.bar()
```

Out[14]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fc2d58585f8>



Q) List out each seasons winners.

logic is that each seasons last match is the final match and its winner is the seasons winner

In [15]:

```
1 x=ipl.drop_duplicates('season',keep='last')
```

In [16]:

```
1 x[['season','winner']].sort_values('season')
```

Out[16]:

	season	winner
116	2008	Rajasthan Royals
173	2009	Deccan Chargers
233	2010	Chennai Super Kings
306	2011	Chennai Super Kings
380	2012	Kolkata Knight Riders
456	2013	Mumbai Indians
516	2014	Kolkata Knight Riders
575	2015	Mumbai Indians
635	2016	Sunrisers Hyderabad
58	2017	Mumbai Indians

Q) how many times each team has won the finals

In [17]:

```
1 x['winner'].value_counts().reset_index()
```

Out[17]:

	index	winner
0	Mumbai Indians	3
1	Kolkata Knight Riders	2
2	Chennai Super Kings	2
3	Rajasthan Royals	1
4	Deccan Chargers	1
5	Sunrisers Hyderabad	1

In [18]:

```
1 ipl.head(10)
```

Out[18]:

	id	season	city	date	team1	team2	toss_winner	toss_
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	
5	6	2017	Hyderabad	2017-04-09	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	
6	7	2017	Mumbai	2017-04-09	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	
7	8	2017	Indore	2017-04-10	Royal Challengers Bangalore	Kings XI Punjab	Royal Challengers Bangalore	
8	9	2017	Pune	2017-04-11	Delhi Daredevils	Rising Pune Supergiant	Rising Pune Supergiant	
9	10	2017	Mumbai	2017-04-12	Sunrisers Hyderabad	Mumbai Indians	Mumbai Indians	

Q)Find out whether the toss winner of the game has actually won

the game by his decision to bat or field first?

""" [Toss Decision](#) """

In [19]:

```
1 toss_data = ipl[['season', 'city', 'venue', 'toss_winner', 'toss_decision',
```

In [20]:

```
1 toss_data.head(10)
```

Out[20]:

	season	city	venue	toss_winner	toss_decision	winner
0	2017	Hyderabad	Rajiv Gandhi International Stadium, Uppal	Royal Challengers Bangalore	field	Sunrisers Hyderabad
1	2017	Pune	Maharashtra Cricket Association Stadium	Rising Pune Supergiant	field	Rising Pune Supergiant
2	2017	Rajkot	Saurashtra Cricket Association Stadium	Kolkata Knight Riders	field	Kolkata Knight Riders
3	2017	Indore	Holkar Cricket Stadium	Kings XI Punjab	field	Kings XI Punjab
4	2017	Bangalore	M Chinnaswamy Stadium	Royal Challengers Bangalore	bat	Royal Challengers Bangalore
5	2017	Hyderabad	Rajiv Gandhi International Stadium, Uppal	Sunrisers Hyderabad	field	Sunrisers Hyderabad
6	2017	Mumbai	Wankhede Stadium	Mumbai Indians	field	Mumbai Indians
7	2017	Indore	Holkar Cricket Stadium	Royal Challengers Bangalore	bat	Kings XI Punjab
8	2017	Pune	Maharashtra Cricket Association Stadium	Rising Pune Supergiant	field	Delhi Daredevils
9	2017	Mumbai	Wankhede Stadium	Mumbai Indians	field	Mumbai Indians



In [21]:

```
1 toss_data['toss_with_bat']=""  
2 toss_data['toss_with_field']=""
```

In [22]:

```
1 d1 = toss_data['toss_winner']==toss_data['winner']
```

In [23]:

```
1 toss_data = toss_data[d1].sort_values('toss_winner')
```

In [24]:

```
1 def findValues_for_bat(toss_decision):  
2     if toss_decision=='bat':  
3         return 'yes'  
4     else:  
5         return 'no'  
6 def findValues_for_field(toss_decision):  
7     if toss_decision=='field':  
8         return 'yes'  
9     else:  
10        return 'no'
```

In [25]:

```
1 toss_data['toss_with_bat']=toss_data['toss_decision'].apply(findValues_  
2 toss_data['toss_with_field']=toss_data['toss_decision'].apply(findValue
```

In [26]:

```
1 toss_data.head(10)
```

Out[26]:

	season	city	venue	toss_winner	toss_decision	winner
520	2015	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
181	2010	Kolkata	Eden Gardens	Chennai Super Kings	bat	Chennai Super Kings
445	2013	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
60	2008	Chandigarh	Punjab Cricket Association Stadium, Mohali	Chennai Super Kings	bat	Chennai Super Kings
303	2011	Mumbai	Wankhede Stadium	Chennai Super Kings	field	Chennai Super Kings
424	2013	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
306	2011	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
421	2013	Pune	Subrata Roy Sahara Stadium	Chennai Super Kings	bat	Chennai Super Kings
86	2008	Delhi	Feroz Shah Kotla	Chennai Super Kings	field	Chennai Super Kings
541	2015	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings

In [27]:

```
1 toss_data=toss_data.rename(columns={'toss_with_bat':'bat first','toss_w  
2 toss_data.head(10)
```

Out[27]:

	season	city	venue	toss_winner	toss_decision	winner
520	2015	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
181	2010	Kolkata	Eden Gardens	Chennai Super Kings	bat	Chennai Super Kings
445	2013	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
60	2008	Chandigarh	Punjab Cricket Association Stadium, Mohali	Chennai Super Kings	bat	Chennai Super Kings
303	2011	Mumbai	Wankhede Stadium	Chennai Super Kings	field	Chennai Super Kings
424	2013	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
306	2011	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings
421	2013	Pune	Subrata Roy Sahara Stadium	Chennai Super Kings	bat	Chennai Super Kings
86	2008	Delhi	Feroz Shah Kotla	Chennai Super Kings	field	Chennai Super Kings
541	2015	Chennai	MA Chidambaram Stadium, Chepauk	Chennai Super Kings	bat	Chennai Super Kings

In [28]:

```
1 #toss_data = toss_data.drop(['venue', 'city', 'winner'],axis=1)
```

In [29]:

```
1 t = toss_data.groupby(['season', 'toss_winner', 'bat first'],sort=True)
2 #t = toss_data.groupby(['season', 'toss_winner', 'toss_decision'],sort=Tr
```

In [30]:

```
1 t.size().head(15)
```

Out[30]:

season	toss_winner	bat first	
2008	Chennai Super Kings	no	1
		yes	2
	Deccan Chargers	no	2
	Delhi Daredevils	no	2
	Kings XI Punjab	no	3
		yes	1
	Kolkata Knight Riders	yes	3
	Mumbai Indians	no	4
	Rajasthan Royals	no	7
		yes	2
2009	Royal Challengers Bangalore	yes	1
	Chennai Super Kings	yes	4
	Deccan Chargers	no	3
		yes	4
	Delhi Daredevils	no	4

dtype: int64

Q)Does the below thing have any real meaning? like is the below result useful?

In [31]:

```
1 t.describe()
```

Out[31]:

		count		unique	city		count		unique	to
season	toss_winner	bat first			top	freq				
	Chennai Super Kings	no	1	1	Delhi	1	1	1	Feroz Sha Kot	
		yes	2	2	Bangalore	1	2	2	Punjab Crick Associatic Stadiur Moha	
	Deccan Chargers	no	2	2	Mumbai	1	2	2	Dr DY Pa Spor Academ	

In []:

```
1
```

In [11]:

```
1 ## Importing Dataset
2 delivery = pd.read_csv('deliveries.csv')
```

In [12]:

```
1 ## Checking ALL Information Related with Dataset
2 delivery.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
#   Column                Non-Null Count  Dtype
---  -
0   match_id              150460 non-null  int64
1   inning                150460 non-null  int64
2   batting_team          150460 non-null  object
3   bowling_team          150460 non-null  object
4   over                  150460 non-null  int64
5   ball                  150460 non-null  int64
6   batsman                150460 non-null  object
7   non_striker            150460 non-null  object
8   bowler                 150460 non-null  object
9   is_super_over          150460 non-null  int64
10  wide_runs              150460 non-null  int64
11  bye_runs                150460 non-null  int64
12  legbye_runs             150460 non-null  int64
13  noball_runs             150460 non-null  int64
14  penalty_runs            150460 non-null  int64
15  batsman_runs            150460 non-null  int64
16  extra_runs              150460 non-null  int64
17  total_runs              150460 non-null  int64
18  player_dismissed        7438 non-null    object
19  dismissal_kind           7438 non-null    object
20  fielder                 5369 non-null    object
dtypes: int64(13), object(8)
memory usage: 24.1+ MB
```

In [13]:

```
1 ## Checking Rows and Column
2 delivery.shape
```

Out[13]:

(150460, 21)

In [14]:

```
1 ## Checking all Columns Available  
2 delivery.columns
```

Out[14]:

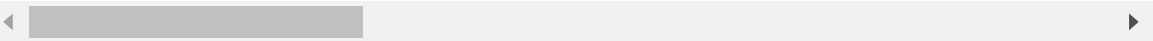
```
Index(['match_id', 'inning', 'batting_team', 'bowling_team',  
      'over', 'ball',  
      'batsman', 'non_striker', 'bowler', 'is_super_over',  
      'wide_runs',  
      'bye_runs', 'legbye_runs', 'noball_runs', 'penalty_run  
s',  
      'batsman_runs', 'extra_runs', 'total_runs', 'player_di  
smitted',  
      'dismissal_kind', 'fielder'],  
      dtype='object')
```

In [36]:

```
1 delivery.head(10)
```

Out[36]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_s
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S DI
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S DI
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S DI
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S DI
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S DI
5	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	6	S Dhawan	DA V
6	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	7	S Dhawan	DA V
7	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	1	S Dhawan	DA V
8	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	2	DA Warner	S DI
9	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	3	DA Warner	S DI



Q) top 5 batsman in ipl according to most number of runs?

In [37]:

```
1 t=delivery.groupby(['batsman'])['batsman_runs'].sum()
```

In [38]:

```
1 t.sort_values(ascending=False).head(5)
```

Out[38]:

```
batsman
SK Raina      4548
V Kohli       4423
RG Sharma     4207
G Gambhir     4132
DA Warner     4014
Name: batsman_runs, dtype: int64
```

Q) which batsman has played most number of balls

In [39]:

```
1 t=delivery.groupby(['batsman'])
```

In [40]:

```
1 t['batsman_runs'].count().sort_values(ascending=False).head(5)
2 # logic is each row is one ball
```

Out[40]:

```
batsman
V Kohli      3494
G Gambhir    3433
SK Raina     3369
RG Sharma    3274
S Dhawan     3005
Name: batsman_runs, dtype: int64
```

Q) which batsman has hit highest number of 4s

In [41]:

```
1 mask=delivery['batsman_runs']==4
```

In [42]:

```
1 delivery_fours = delivery[mask].groupby('batsman')['batsman_runs'].count()
2 # count() instead of sum() because we need to count the number of 4s in
```

In [43]:

```
1 delivery_fours.head(5)
```

Out[43]:

```
batsman
G Gambhir    484
SK Raina     402
DA Warner    401
S Dhawan     401
V Kohli      384
Name: batsman_runs, dtype: int64
```

Q) how many total fours in ipl till now

In [44]:

```
1 len(delivery[mask])
```

Out[44]:

```
17033
```

Q) which batsman has hit highest number of 6's

In [45]:

```
1 mask=delivery['batsman_runs']==6
2 delivery_fours = delivery[mask].groupby('batsman')['batsman_runs'].count()
3 delivery_fours.head(5)
```

Out[45]:

```
batsman
CH Gayle      266
SK Raina      174
RG Sharma     173
DA Warner     160
V Kohli       160
Name: batsman_runs, dtype: int64
```

Q) which bowler has given most no. of dot balls

In [46]:

```
1 mask = delivery['total_runs']==0
2 delivery_dot = delivery[mask]
3 delivery_dot.groupby('bowler')['total_runs'].count().sort_values(ascending=False)
```

Out[46]:

```
bowler
P Kumar      1075
Harbhajan Singh  1062
SL Malinga    1060
DW Steyn      978
A Mishra      953
Name: total_runs, dtype: int64
```

Q) batsman which when on non-striker end, there has been most dismissals

Case 1: Only the striker is dismissed everytime

Case 2: The striker and non-striker both can be dismissed

Sir's version runs for case 1 and 2 both and my version runs only for case 1.

In [47]:

```
1 #My version - only works for case 1
2 delivery_dismissal1 = delivery.dropna(subset=['player_dismissed'])
3 mask=delivery_dismissal1['batsman']==delivery_dismissal1['player_dismissed']
4 delivery_dismissal1 = delivery_dismissal1[mask]
5 ans=delivery_dismissal1.groupby('non_striker')['player_dismissed'].count()
6 #delivery_dismissal1.shape
7 ans
```

Out[47]:

	non_striker	player_dismissed
0	RG Sharma	170
1	V Kohli	150
2	SK Raina	138
3	G Gambhir	135
4	RV Uthappa	134

In [48]:

```
1 #Sirs version - works for both the cases but actually the question aske
2 delivery_dismissal2 = delivery.fillna({'player_dismissed':'Not'})
3 mask=delivery_dismissal2['player_dismissed']!='Not'
4 #mask1=delivery_dismissal2['player_dismissed']!='delivery_dismissal2['no
5 #delivery_dismissal2 = delivery_dismissal2[mask & mask1]
6 delivery_dismissal2 = delivery_dismissal2[mask]
7 ans1=delivery_dismissal2.groupby('non_striker')['player_dismissed'].cou
8 #delivery_dismissal2.shape
9 ans1
```

Out[48]:

	non_striker	player_dismissed
0	RG Sharma	171
1	V Kohli	151
2	SK Raina	145
3	G Gambhir	142
4	RV Uthappa	138
5	KD Karthik	125
6	MS Dhoni	120
7	AB de Villiers	120
8	S Dhawan	116
9	DA Warner	112

Below cell is the proof by showing that RG Sharma has been out being on non_strikers end just one time

Comparing sir's and my version and according to the question that is:- count the number of times batsman on strikers end got out while a particular batsman was on the non-strikers end.

My version seems to be correct

In [49]:

```
1 #Finding the cases when the non-striker was the player_dismissed
2 dell = delivery.dropna(subset=['player_dismissed'])
3 mask = dell['non_striker']==dell['player_dismissed']
4 mask1 = dell['non_striker']=='RG Sharma'
5 dell = dell[mask & mask1]
6 dell.groupby('non_striker')['player_dismissed'].count().sort_values(asc
```

Out[49]:

	non_striker	player_dismissed
0	RG Sharma	1

In [50]:

```
1 7084+354 # means sir's version did take into account that non-striker ca
```

Out[50]:

7438

Q) make a function with one argument:- batsman name and return the name of the team that batsman has hit most runs against

In [51]:

```
1 # Finding the batsmans and the team they have hit most number of runs against
2 hit_runs = delivery.groupby(['bowling_team', 'batsman'])['batsman_runs'].sum()
3 hit_runs
```

Out[51]:

	bowling_team	batsman	batsman_runs
0	Kings XI Punjab	CH Gayle	797
1	Kolkata Knight Riders	RG Sharma	710
2	Mumbai Indians	SK Raina	708
3	Chennai Super Kings	V Kohli	706
8	Delhi Daredevils	RG Sharma	670
11	Royal Challengers Bangalore	G Gambhir	644
31	Rajasthan Royals	AB de Villiers	485
50	Sunrisers Hyderabad	V Kohli	439
76	Pune Warriors	CH Gayle	383
99	Deccan Chargers	R Dravid	339
103	Gujarat Lions	DA Warner	336
288	Rising Pune Supergiants	V Kohli	188
503	Rising Pune Supergiant	PA Patel	108
540	Kochi Tuskers Kerala	SR Tendulkar	100

In [52]:

```
1 # Finding virat has hit most runs against which team
2 mask=delivery['batsman']=='V Kohli'
3 bats = delivery[mask]
4 bats.groupby('bowling_team')['batsman_runs'].sum().sort_values(ascending=True)
```

Out[52]:

'Chennai Super Kings'

In [53]:

```
1 def opp_team_most_runs(batsman_name):
2     mask=delivery['batsman']==batsman_name
3     bats = delivery[mask]
4     ans=bats.groupby('bowling_team')['batsman_runs'].sum().sort_values(
5     return ans
```

In [54]:

```
1 opp_team_most_runs('RG Sharma')
```

Out[54]:

'Kolkata Knight Riders'

Q) make a function which takes one argument:- batsman and return which bowler has givn highest number of runs to that batsman

In [55]:

```
1 mask=delivery['batsman']=='V Kohli'
2 bats = delivery[mask]
3 bats.groupby('bowler')['batsman_runs'].sum().sort_values(ascending=False)
```

Out[55]:

```
bowler
A Mishra      149
UT Yadav      141
DJ Bravo      130
R Ashwin      127
RA Jadeja     104
Name: batsman_runs, dtype: int64
```

In [56]:

```
1 def most_runs_taken_from_bowler(batsman_name):
2     mask=delivery['batsman']==batsman_name
3     bats = delivery[mask]
4     return bats.groupby('bowler')['batsman_runs'].sum().sort_values(asc
```

In [57]:

```
1 most_runs_taken_from_bowler('V Kohli')
```

Out[57]:

```
'A Mishra'
```

Q)in each over which team has hit how many sixes

In [58]:

```
1 # Pivot_table method
2 sixes = delivery[delivery['batsman_runs']==6]
3 y=sixes.pivot_table(index=['batting_team'],columns=['over'],values='bat
4 print(y)
```

over	1	2	3	4	5	
6 7 8 9 10 11 12 13 14 15 1						
6 17 18 19 20						
batting_team						
Chennai Super Kings	5.0	17.0	37.0	34.0	41.0	4
3.0 22.0 25.0 23.0 23.0	36.0	36.0	35.0	45.0	43.0	4
6.0 51.0 58.0 54.0 68.0						
Deccan Chargers	3.0	21.0	11.0	17.0	27.0	2
2.0 11.0 13.0 17.0 14.0	15.0	26.0	23.0	22.0	20.0	2
1.0 38.0 37.0 23.0 19.0						
Delhi Daredevils	14.0	19.0	30.0	41.0	29.0	2
7.0 20.0 26.0 32.0 24.0	34.0	38.0	37.0	35.0	28.0	4
3.0 56.0 44.0 59.0 50.0						
Gujarat Lions	4.0	6.0	14.0	9.0	11.0	1
2.0 10.0 6.0 11.0 8.0	5.0	7.0	5.0	8.0	5.0	
5.0 7.0 9.0 9.0 4.0						
Kings XI Punjab	10.0	19.0	27.0	36.0	30.0	3
9.0 19.0 35.0 36.0 29.0	37.0	38.0	47.0	41.0	51.0	5
9.0 39.0 57.0 53.0 60.0						
Kochi Tuskers Kerala	2.0	4.0	3.0	3.0	3.0	
3.0 2.0 2.0 1.0 3.0	1.0	3.0	2.0	3.0	4.0	N
aN 3.0 4.0 5.0 2.0						
Kolkata Knight Riders	10.0	13.0	28.0	35.0	25.0	3
4.0 26.0 25.0 27.0 21.0	34.0	32.0	42.0	31.0	40.0	5
0.0 55.0 52.0 45.0 34.0						
Mumbai Indians	9.0	17.0	22.0	28.0	43.0	5
0.0 18.0 23.0 35.0 22.0	33.0	49.0	51.0	50.0	53.0	6
6.0 60.0 86.0 72.0 89.0						
Pune Warriors	5.0	6.0	6.0	7.0	7.0	
3.0 5.0 10.0 9.0 6.0	13.0	5.0	10.0	13.0	13.0	1
6.0 13.0 12.0 18.0 19.0						
Rajasthan Royals	12.0	7.0	13.0	21.0	23.0	2
4.0 15.0 24.0 20.0 26.0	30.0	33.0	39.0	38.0	34.0	3
7.0 45.0 34.0 37.0 26.0						
Rising Pune Supergiant	1.0	NaN	2.0	5.0	8.0	
6.0 1.0 NaN 3.0 8.0	3.0	2.0	7.0	4.0	5.0	
5.0 2.0 2.0 14.0 11.0						
Rising Pune Supergiants	NaN	2.0	1.0	1.0	5.0	
5.0 4.0 3.0 1.0 1.0	2.0	2.0	1.0	1.0	7.0	
6.0 6.0 9.0 2.0 9.0						
Royal Challengers Bangalore	20.0	28.0	40.0	43.0	40.0	2
9.0 18.0 34.0 47.0 36.0	47.0	36.0	51.0	53.0	54.0	7
1.0 61.0 82.0 74.0 71.0						
Sunrisers Hyderabad	3.0	12.0	16.0	8.0	17.0	1
0.0 13.0 19.0 15.0 12.0	11.0	11.0	19.0	18.0	33.0	2
2.0 18.0 37.0 42.0 28.0						

In [59]:

```
1 # Groupby method
2 sixes.groupby(['batting_team', 'over'])['batsman_runs'].count().head(10)
```

Out[59]:

```
batting_team    over
Chennai Super Kings  1      5
                   2     17
                   3     37
                   4     34
                   5     41
                   6     43
                   7     22
                   8     25
                   9     23
                  10     23
Name: batsman_runs, dtype: int64
```

In [60]:

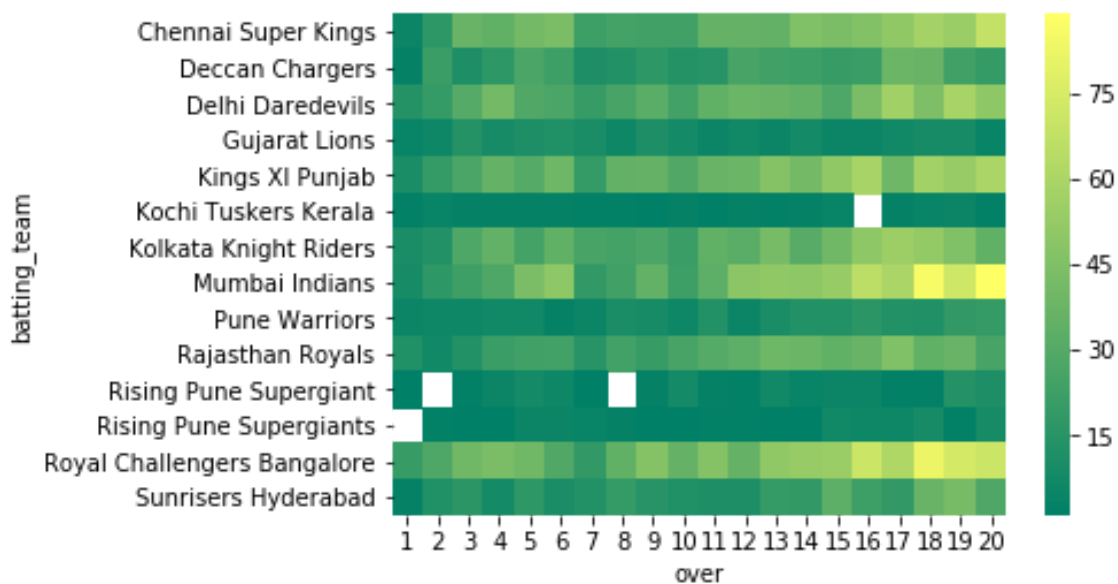
```
1 import seaborn as sns
```

In [61]:

```
1 sns.heatmap(y,cmap='summer')
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc2c8be1128>
```



Q) find the orange cap holders of each season

In [62]:

```
1 merged_data = ipl.merge(delivery,left_on='id',right_on='match_id')
```

In [63]:

```
1 merged_data.head()
```

Out[63]:

	id	season	city	date	team1	team2	toss_winner	toss_d
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
1	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
2	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
3	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
4	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	

In [64]:

```
1 df = merged_data.groupby(['season','batsman'])['batsman_runs'].sum().re
```

In [65]:

```
1 df.sort_values('batsman_runs').drop_duplicates(subset=['season'],keep=''
```

Out[65]:

	season	batsman	batsman_runs
115	2008	SE Marsh	616
229	2009	ML Hayden	572
446	2010	SR Tendulkar	618
502	2011	CH Gayle	608
684	2012	CH Gayle	733
910	2013	MEK Hussey	733
1088	2014	RV Uthappa	660
1148	2015	DA Warner	562
1383	2016	V Kohli	973
1422	2017	DA Warner	641

Q1) last 5 overs (death overs) mein sabse dangerous batsman, strike rate (no. of runs divided by number of balls)*100.

base criteria is player has played 200 balls, (between 16 and 20 overs he has completed 200 balls)

In [66]:

```
1 mask = merged_data['over']>15
```

In [67]:

```
1 # Only for death overs that is from 16-20 both inclusive
2 danger = merged_data[mask]
3 danger.head(10)
```

Out[67]:

	id	season	city	date	team1	team2	toss_winner	toss
93	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
94	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
95	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
96	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
97	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
98	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
99	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
100	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	
101	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	

	id	season	city	date	team1	team2	toss_winner	toss
102	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	

In [68]:

```

1 # Finding each batsmans total runs in history in death overs
2 runs = danger.groupby('batsman')['batsman_runs'].sum().sort_values(asc=
3 runs.head(10)

```

Out[68]:

	batsman	batsman_runs
0	MS Dhoni	2076
1	KA Pollard	1352
2	RG Sharma	1314
3	AB de Villiers	1203
4	V Kohli	993
5	YK Pathan	930
6	Yuvraj Singh	883
7	JP Duminy	869
8	SK Raina	767
9	RA Jadeja	753

In [69]:

```
1 # Finding number of balls played by each batsman in death overs
2 balls = danger.groupby('batsman')['ball'].count().sort_values(ascending
3 balls.head(10)
```

Out[69]:

	batsman	ball
0	MS Dhoni	1224
1	KA Pollard	838
2	RG Sharma	748
3	YK Pathan	584
4	RA Jadeja	576
5	AB de Villiers	570
6	V Kohli	546
7	JP Duminy	518
8	Yuvraj Singh	516
9	IK Pathan	465

In [70]:

```
1 # Merging runs and ball values in a single dataframe for simplicity
2 balls_and_runs = balls.merge(runs,left_on='batsman',right_on='batsman')
```

In [71]:

```
1 balls_and_runs = balls_and_runs[balls_and_runs['ball']>=200]
2 len(balls_and_runs)
```

Out[71]:

44

In [72]:

```
1 balls_and_runs['strike']=0
```

In [73]:

```
1 balls_and_runs['strike']=(balls_and_runs['batsman_runs']/balls_and_runs
2 balls_and_runs.sort_values('strike',ascending=False).head(2)
```

Out[73]:

	batsman	ball	batsman_runs	strike
5	AB de Villiers	570	1203	211.052632
38	DA Warner	228	432	189.473684

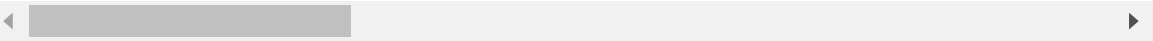
Q2) top 10 batsman, and top 10 bowlers with max number of wickets and combine them and make a heatmap which tells which top batsman has hitten most runs against a top bowler

In [74]:

```
1 top_bowlers=delivery.dropna(subset=['player_dismissed']).copy()
2 top_bowlers[top_bowlers['bowler']==top_bowlers['fielder']].reset_index()
```

Out[74]:

	index	match_id	inning	batting_team	bowling_team	over	ball	batsma
0	557	3	1	Gujarat Lions	Kolkata Knight Riders	11	2	AJ Finc
1	2700	12	1	Royal Challengers Bangalore	Mumbai Indians	18	7	P Neq
2	5632	24	1	Mumbai Indians	Delhi Daredevils	19	7	HI Pandya
3	5966	25	2	Rising Pune Supergiant	Sunrisers Hyderabad	14	1	MS Dhoi
4	6129	26	1	Kings XI Punjab	Gujarat Lions	20	6	WP Sah
5	6680	28	2	Mumbai Indians	Rising Pune Supergiant	20	5	Harbhaja Sing
6	7766	33	2	Royal Challengers Bangalore	Rising Pune Supergiant	8	4	KI Jadha
7	7963	34	1	Gujarat Lions	Mumbai Indians	19	7	J Faulkne
8	8386	36	1	Sunrisers Hyderabad	Kolkata Knight Riders	20	6	Yuvra Sing
9	12427	53	1	Mumbai Indians	Kolkata Knight Riders	16	6	A Rayud



In [75]:

```
1 l=list()
2 for i,row in top_bowlers.iterrows():
3     if row['dismissal_kind']=='run out' or row['dismissal_kind']=='retu
4         l.append(i)
5 for i in l:
6     top_bowlers.drop(i,inplace=True)
```

In [76]:

```
1 top_bowlers.shape
```

Out[76]:

(6673, 21)

In [77]:

```
1 top_bowlers=top_bowlers.groupby('bowler')['player_dismissed'].count().s
2 #top_bowlers.groupby('bowler')['dismissal_kind'].count().sort_values(as
```

In [78]:

```
1 top_batsman=delivery.groupby('batsman')['batsman_runs'].sum().sort_valu
```

In [79]:

```
1 #mask1=top_bowlers['bowler']==delivery['bowler']
2 #mask2=top_batsman['batsman']==delivery['batsman']
3 #bat_and_bowl = delivery.mask(top_bowlers['bowler']==delivery['bowler']
4 top_batsman.head()
```

Out[79]:

	batsman	batsman_runs
0	SK Raina	4548
1	V Kohli	4423
2	RG Sharma	4207
3	G Gambhir	4132
4	DA Warner	4014

In [80]:

```
1 top_bowlers
```

Out[80]:

	bowler	player_dismissed
0	SL Malinga	154
1	A Mishra	134
2	Harbhajan Singh	127
3	PP Chawla	126
4	DJ Bravo	122
5	B Kumar	111
6	A Nehra	106
7	R Vinay Kumar	103
8	Z Khan	102
9	R Ashwin	100

In [81]:

```
1 ex=delivery.copy()
2 ex['top_batsmans']=0.0
3 ex['top_bowlers']=0.0
4 ex.head(10)
```

Out[81]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_s
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S DI
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S DI
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S DI
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S DI
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S DI
5	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	6	S Dhawan	DA V
6	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	7	S Dhawan	DA V
7	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	1	S Dhawan	DA V
8	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	2	DA Warner	S DI
9	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	2	3	DA Warner	S DI

In [82]:

```
1 for i,row in top_batsman.iterrows():
2     e=str(row['batsman'])
3     for j,r in ex.iterrows():
4         if str(r['batsman'])==e:
5             result=1
6             ex.at[j,'top_batsmans']=result
7     print('done with ',e)
8 print('done with batsman')
```

done with SK Raina
done with V Kohli
done with RG Sharma
done with G Gambhir
done with DA Warner
done with RV Uthappa
done with CH Gayle
done with S Dhawan
done with MS Dhoni
done with AB de Villiers
done with batsman

In [83]:

```
1 for i,row in top_bowlers.iterrows():
2     e=row['bowler']
3     for j,r in ex.iterrows():
4         if r['bowler']==e:
5             ex.at[j,'top_bowlers']=1
6     print('done with ',e)
```

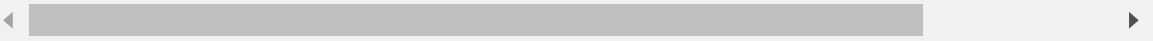
done with SL Malinga
done with A Mishra
done with Harbhajan Singh
done with PP Chawla
done with DJ Bravo
done with B Kumar
done with A Nehra
done with R Vinay Kumar
done with Z Khan
done with R Ashwin

In [84]:

```
1 mask1=ex['top_batsmans']==1
2 mask2=ex['top_bowlers']==1
3 y=ex[mask1 & mask2].pivot_table(index='batsman',columns='bowler',values
4 y
```

Out[84]:

bowler	A Mishra	A Nehra	B Kumar	DJ Bravo	Harbhajan Singh	PP Chawla	R Ashwin	R Vinay Kumar
batsman								
AB de Villiers	9.0	61.0	48.0	69.0	80.0	43.0	42.0	16.0
CH Gayle	45.0	45.0	104.0	53.0	78.0	103.0	49.0	0.0
DA Warner	37.0	25.0	2.0	44.0	97.0	72.0	48.0	5.0
G Gambhir	54.0	27.0	84.0	32.0	84.0	53.0	43.0	62.0
MS Dhoni	24.0	76.0	81.0	55.0	47.0	72.0	NaN	33.0
RG Sharma	78.0	62.0	31.0	66.0	8.0	136.0	70.0	22.0
RV Uthappa	84.0	55.0	65.0	32.0	75.0	41.0	72.0	65.0
S Dhawan	16.0	42.0	14.0	48.0	103.0	46.0	51.0	24.0
SK Raina	83.0	15.0	62.0	55.0	132.0	152.0	12.0	98.0
V Kohli	149.0	60.0	53.0	130.0	78.0	95.0	127.0	26.0

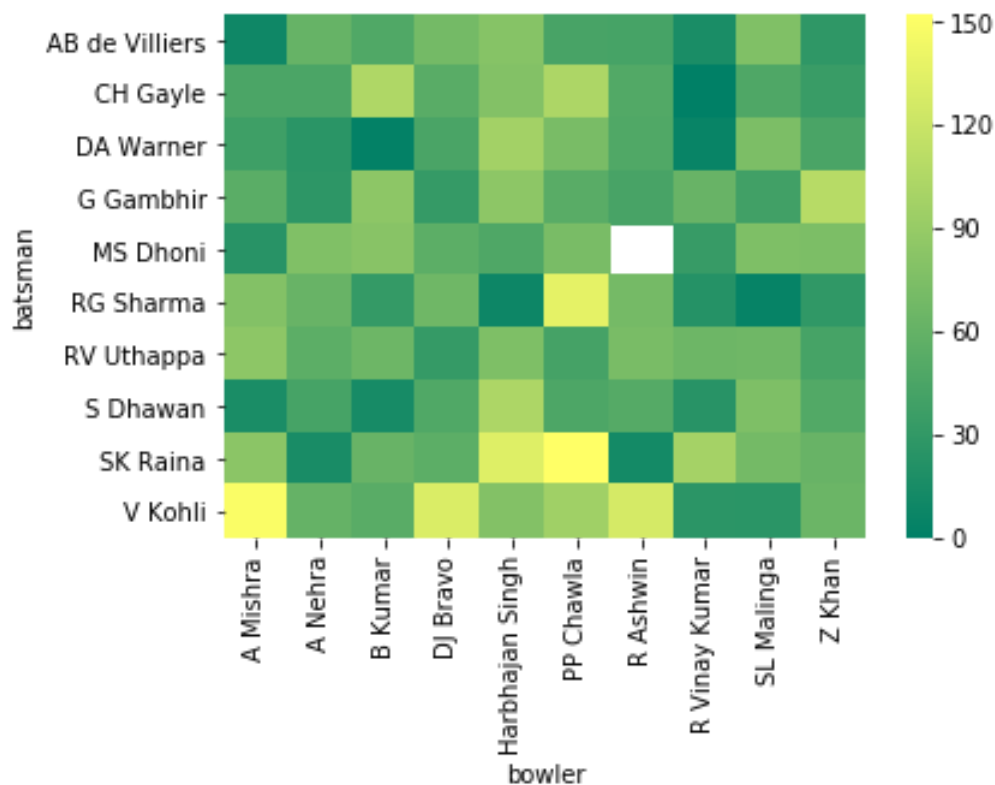


In [85]:

```
1 import seaborn as sns
2 sns.heatmap(y,cmap='summer')
```

Out[85]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fc2c8be10f0>



In [86]:

```
1 mask1=ex['batsman']=='CH Gayle'
2 mask2=ex['bowler']=='A Mishra'
3 ex[mask1 & mask2].groupby('batsman')['batsman_runs'].sum()
```

Out[86]:

```
batsman
CH Gayle    45
Name: batsman_runs, dtype: int64
```

Q3) most economical bowler in death overs

Economy rate = runs conceded / overs bowled

Example if a bowler has given 35 runs in 3.1 overs

so his overs bowled will be calculated as $3 + 1/6 = 3.1666$

and Economy rate would be $35/3.166 = 11.054$

In [87]:

```
1 danger=delivery[delivery['over']>15].copy()
```

In [88]:

```
1 def over_conversion(over,ball):
2     o=str(over-15)
3     b=str(ball)
4     return float(o+"."+b)
5 def econ_rate(runs,ball):
6     return float(float(runs)/float(ball/6))
```

In [89]:

```
1 over_conversion(16,1)
```

Out[89]:

1.1

In [90]:

```
1 danger['o']=0.0
```

In [91]:

```
1 # useful knowledge to update values
2 for i,row in danger.iterrows():
3     result = over_conversion(row['over'],row['ball'])
4     danger.at[i,'o']=result #way to update data in row
```

Make sure u run either of the below two cells

In [92]:

```
1 #Finding out each team's most economical bowler in death overs
2 danger=danger[['bowling_team','over','ball','o','bowler','total_runs']]
3 total_runs_given=danger.groupby(['bowling_team','bowler'])['total_runs']
4 total_balls_bowled=danger.groupby(['bowling_team','bowler'])['ball'].co
5 runs_and_balls=total_runs_given.merge(total_balls_bowled,left_on=['bowl
```

In []:

```
1 # Just finding the most economical bowler without consideration which t
2 danger=danger[['bowling_team','over','ball','o','bowler','total_runs']]
3 total_runs_given=danger.groupby('bowler')['total_runs'].sum().sort_valu
4 total_balls_bowled=danger.groupby('bowler')['ball'].count().sort_values
5 runs_and_balls=total_runs_given.merge(total_balls_bowled,left_on=['bowl
```

In [93]:

```
1 runs_and_balls['econ_rate']=0.0
2 #runs_and_balls['econ_rate']=runs_and_balls['total_runs']/(runs_and_bal
```

In [94]:

```
1 for i,row in runs_and_balls.iterrows():
2     result = econ_rate(row['total_runs'],row['ball'])
3     runs_and_balls.at[i,'econ_rate']=result
```


In [95]:

```
1 runs_and_balls[runs_and_balls['ball']>100].sort_values('econ_rate').hea
```

Out[95]:

	bowling_team	bowler	total_runs	ball	econ_rate
74	Chennai Super Kings	M Muralitharan	208	192	6.500000
132	Rajasthan Royals	Sohail Tanvir	122	107	6.841121
81	Royal Challengers Bangalore	DW Steyn	189	165	6.872727
3	Kolkata Knight Riders	SP Narine	790	664	7.138554
46	Chennai Super Kings	DE Bollinger	297	242	7.363636
0	Mumbai Indians	SL Malinga	1300	1050	7.428571
67	Royal Challengers Bangalore	A Kumble	217	174	7.482759
64	Delhi Daredevils	CH Morris	222	177	7.525424
69	Sunrisers Hyderabad	Mustafizur Rahman	212	168	7.571429
49	Royal Challengers Bangalore	MA Starc	265	210	7.571429

In [96]:

```
1 runs_and_balls[runs_and_balls['ball']>=100].sort_values('econ_rate',asc
```

Out[96]:

	index	bowling_team	bowler	total_runs	ball	econ_rate
0	74	Chennai Super Kings	M Muralitharan	208	192	6.500000
1	132	Rajasthan Royals	Sohail Tanvir	122	107	6.841121
2	81	Royal Challengers Bangalore	DW Steyn	189	165	6.872727
3	3	Kolkata Knight Riders	SP Narine	790	664	7.138554
4	0	Mumbai Indians	SL Malinga	1300	1050	7.428571
5	64	Delhi Daredevils	CH Morris	222	177	7.525424
6	69	Sunrisers Hyderabad	Mustafizur Rahman	212	168	7.571429
7	73	Pune Warriors	B Kumar	209	159	7.886792
8	51	Deccan Chargers	DW Steyn	255	188	8.138298
9	112	Kings XI Punjab	B Lee	143	105	8.171429
10	99	Rising Pune Supergiant	JD Unadkat	158	115	8.243478
11	82	Gujarat Lions	Basil Thampi	188	108	10.444444

In []:

```
1
```

Sirs methods for the 3 above questions

Q1) last 5 overs (death overs) mein sabse dangerous batsman, strike rate (no. of runs divided by number of balls)*100.

base criteria is player has played 200 balls, (between 16 and 20 overs he has completed 200 balls)

In [97]:

```
1 df = delivery[delivery['over']>15] #only need to use data from 16 to 20
```

In [98]:

```
1 a=df.groupby('batsman')['batsman_runs'].count() #counting number of bal
```

In [99]:

```
1 b=df.groupby('batsman')['batsman_runs'].count()>200 # only need to take
```

In [100]:

```
1 c=a[b].index.tolist()
```

In [101]:

1	c
---	---

Out[101]:

```
['A Mishra',  
 'AB de Villiers',  
 'AD Mathews',  
 'AM Rahane',  
 'AR Patel',  
 'AT Rayudu',  
 'BJ Hodge',  
 'DA Miller',  
 'DA Warner',  
 'DJ Bravo',  
 'DJ Hussey',  
 'DPMD Jayawardene',  
 'Harbhajan Singh',  
 'IK Pathan',  
 'JA Morkel',  
 'JH Kallis',  
 'JP Duminy',  
 'JP Faulkner',  
 'KA Pollard',  
 'KD Karthik',  
 'KM Jadhav',  
 'LRPL Taylor',  
 'MK Pandey',  
 'MK Tiwary',  
 'MS Dhoni',  
 'NV Ojha',  
 'P Kumar',  
 'PP Chawla',  
 'R Vinay Kumar',  
 'RA Jadeja',  
 'RG Sharma',  
 'RV Uthappa',  
 'S Badrinath',  
 'S Dhawan',  
 'SK Raina',  
 'SPD Smith',  
 'SS Tiwary',  
 'STR Binny',  
 'V Kohli',  
 'WP Saha',  
 'Y Venugopal Rao',  
 'YK Pathan',  
 'Yuvraj Singh']
```

In [102]:

```
1 df.head(10)
```

Out[102]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	no
93	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	1	MC Henriques	
94	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	2	MC Henriques	
95	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	3	Yuvraj Singh	
96	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	4	DJ Hooda	
97	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	5	Yuvraj Singh	
98	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	16	6	DJ Hooda	
99	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	17	1	DJ Hooda	
100	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	17	2	Yuvraj Singh	
101	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	17	3	Yuvraj Singh	
102	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	17	4	Yuvraj Singh	



In [103]:

```
1 d=df[df['batsman'].isin(c)] #isin() function is used to find only those  
2 # stored in a list in c, this
```

In [104]:

```
1 runs=d.groupby('batsman')['batsman_runs'].sum()
```

In [105]:

```
1 balls=d.groupby('batsman')['batsman_runs'].count()
```

In [106]:

```
1 (runs/balls*100).sort_values(ascending=False)
```


Out[106]:

batsman	
AB de Villiers	211.052632
DA Warner	189.473684
DA Miller	186.666667
V Kohli	181.868132
RG Sharma	175.668449
DJ Hussey	175.213675
RV Uthappa	173.454545
Yuvraj Singh	171.124031
JH Kallis	170.562771
MS Dhoni	169.607843
SPD Smith	169.303797
JP Duminy	167.760618
DJ Bravo	167.726161
SK Raina	167.467249
AT Rayudu	165.411765
WP Saha	163.389831
KA Pollard	161.336516
YK Pathan	159.246575
S Dhawan	158.847737
BJ Hodge	157.402597
AM Rahane	152.985075
LRPL Taylor	152.941176
KD Karthik	152.051836
DPMD Jayawardene	152.032520
MK Pandey	151.785714
JA Morkel	149.882353
JP Faulkner	149.319728
S Badrinath	149.116608
Y Venugopal Rao	148.846154
Harbhajan Singh	147.607656
AD Mathews	147.058824
KM Jadhav	144.378698
STR Binny	144.036697
AR Patel	142.794760
IK Pathan	142.580645
MK Tiwary	140.189125
SS Tiwary	136.666667
NV Ojha	134.868421
RA Jadeja	130.729167
PP Chawla	120.257235
P Kumar	109.701493
R Vinay Kumar	108.936170
A Mishra	100.888889

Name: batsman_runs, dtype: float64

Q2) top 10 batsman, and top 10 bowlers with max number of wickets and combine them and make a heatmap which tells which top batsman has hitten most runs against a top bowler

In [107]:

```
1 top_batsman=delivery.groupby('batsman')['batsman_runs'].sum().sort_valu
2 #Finding the top 10 batsman and storing the names in a list using index
```

In [108]:

```
1 top_batsman
```

Out[108]:

```
['SK Raina',
 'V Kohli',
 'RG Sharma',
 'G Gambhir',
 'DA Warner',
 'RV Uthappa',
 'CH Gayle',
 'S Dhawan',
 'MS Dhoni',
 'AB de Villiers']
```

In [109]:

```
1 delivery['dismissal_kind'].value_counts()
```

Out[109]:

```
caught          4373
bowled          1382
run out         755
lbw             455
stumped        243
caught and bowled 211
retired hurt      9
hit wicket        9
obstructing the field 1
Name: dismissal_kind, dtype: int64
```

In [110]:

```
1 dismissal=['caught','bowled','lbw','stumped','caught and bowled','hit w
2 #only in these cases the wicket credit is given to bowler
```

In [111]:

```
1 out=delivery[delivery['dismissal_kind'].isin(dismissal)]
2 #Filtering out batsman who got out by the above ways
```

In [112]:

```
1 bowler=out.groupby('bowler')['dismissal_kind'].count().sort_values(asce
2 #Taking out list of top 10 bowlers having most wickets
```

In [113]:

```
1 len(out)
```

Out[113]:

6673

In [114]:

```
1 batsmandf=delivery[delivery['batsman'].isin(top_batsman)]
2 # filtering out top10 batsman and storing in new variable
```

In [115]:

```
1 finaldf=batsmandf[batsmandf['bowler'].isin(bowler)]
2 # filtering out batsmans who faced the top10 batsmans
```

In [116]:

```
1 finaldf.shape
```

Out[116]:

(4625, 21)

In [117]:

```
1 y=finaldf.pivot_table(index='bowler',columns='batsman',values='batsman_
```

In [118]:

```
1 sns.heatmap(y,cmap='summer')
```

Out[118]:

<matplotlib.axes._subplots.AxesSubplot at 0x7fc2c8aba748>



Q3) most economical bowler in death overs

Economy rate = runs conceded / overs bowled

Example if a bowler has given 35 runs in 3.1 overs

so his overs bowled will be calculated as $3 + 1/6 = 3.1666$

and Economy rate would be $35 / 3.166 = 11.054$

In [119]:

```
1 do=delivery[delivery['over']>15]
2 #we only need overs between 16 to 20 overs
```

In [120]:

```
1 len(do)
```

Out[120]:

33737

In [121]:

```
1 a=do.groupby('bowler')['total_runs'].count() #Finding number of balls b
2 a.sort_values(ascending=False)
```

Out[121]:

bowler	
SL Malinga	1050
DJ Bravo	885
B Kumar	715
R Vinay Kumar	673
SP Narine	664
UT Yadav	631
P Kumar	624
DW Steyn	624
Z Khan	620
SR Watson	611
RP Singh	556
A Nehra	546
L Balaji	523
IK Pathan	505
JP Faulkner	493
A Mishra	461
MM Sharma	454

In [122]:

```
1 b=do.groupby('bowler')['total_runs'].count(>100 #Only need bowlers who
```

In [123]:

```
1 bowler=a[b].index.tolist() # storing bowler names in a list
```

In [124]:

```
1 newdf=delivery[delivery['bowler'].isin(bowler)]
```

In [125]:

```
1 newdf.shape
```

Out[125]:

```
(110789, 21)
```

In [126]:

```
1 run=newdf.groupby('bowler')['total_runs'].sum()
```

In [127]:

```
1 balls=newdf.groupby('bowler')['total_runs'].count()
```

In [128]:

```
1 balls=balls/6
```

In [129]:

```
1 (run/balls).sort_values().head
```

Out[129]:

<bound method NDFrame.head of bowler

Sohail Tanvir	6.226415
SP Narine	6.395706
R Ashwin	6.490886
DW Steyn	6.600278
A Kumble	6.646999
M Muralitharan	6.698292
SL Malinga	6.757238
DL Vettori	6.833121
J Botha	6.922426
Harbhajan Singh	6.931415
S Nadeem	7.029024
Mustafizur Rahman	7.038168
B Kumar	7.039922
R Sharma	7.058824
DP Nannes	7.097242
MA Starc	7.107843
Shakib Al Hasan	7.115100
Iqbal Abdulla	7.158006
DE Bollinger	7.160000
SK Warne	7.187244
M Kartik	7.197970
MF Maharooof	7.238095
WD Parnell	7.297837
A Mishra	7.336293
SK Raina	7.357143
Yuvraj Singh	7.373993
B Lee	7.375546
PP Ojha	7.400514
AR Patel	7.425000
R Bhatia	7.437688
SK Trivedi	7.491329
SW Tait	7.505618
MM Patel	7.523878
CH Morris	7.525912
RJ Harris	7.526012
Z Khan	7.539543
M Morkel	7.543261
NM Coulter-Nile	7.566553
P Kumar	7.604096
SR Watson	7.618562
AC Thomas	7.633028
BW Hilfenhaus	7.646154
Sandeep Sharma	7.649647
PP Chawla	7.667695
DT Christian	7.678715
Azhar Mahmood	7.684783

IK Pathan	7.698060
A Nehra	7.711246
S Sreesanth	7.736008
RP Singh	7.738527
DS Kulkarni	7.749495
RA Jadeja	7.767584
CH Gayle	7.772021
Harmeet Singh	7.792350
MG Johnson	7.818493
JH Kallis	7.831017
YS Chahal	7.875308
A Singh	7.888889
KK Cooper	7.890000
JJ Bumrah	7.931734
L Balaji	7.940280
J Theron	7.940426
AB Dinda	7.940843

Thank You

