```
In [1]: import pandas as pd
        import copy
```

```
In [2]: xls = pd.ExcelFile('Cricket Data Set.xlsx')
        xls

Out[2]: <pandas.io.excel._base.ExcelFile at 0x29082be1ac0>
```

```
In [3]: print(len(xls.sheet_names))
        xls.sheet_names

        12

Out[3]: ['Player Details',
         'Parameters',
         'Control Sheet',
         'IND Vs SA',
         'IND Vs PAK',
         'IND Vs AUS',
         'IND Vs AFG',
         'IND Vs WI',
         'IND Vs ENG',
         'IND Vs BAN',
         'IND Vs SL',
         'Analysis parameters']
```

df_names=[] for sheet_names in xls.sheet_names: df_name = '' for name in sheet_names.split(): df_name += f'{name.lower()}_' df_name += 'df' df_names.append(df_name)

print(len(df_names))
df_names

```
In [4]: df_names = ['_'.join(sheet_names.lower().split()) + '_df' for sheet_names in xls.sheet_names]
        print(len(df_names))
        df_names

        12

Out[4]: ['player_details_df',
         'parameters_df',
         'control_sheet_df',
         'ind_vs_sa_df',
         'ind_vs_pak_df',
         'ind_vs_aus_df',
         'ind_vs_afg_df',
         'ind_vs_wi_df',
         'ind_vs_eng_df',
         'ind_vs_ban_df',
         'ind_vs_sl_df',
         'analysis_parameters_df']
```

sheet_index = 0

for df_name in df_names: exec(f'{df_name} = pd.read_excel(xls,sheet_index)') sheet_index +=1

```
In [5]: df_dict = {df_names[sheet_index] : pd.read_excel(xls,sheet_index) for sheet_index in range(len(df_names))}
        df_dict.keys()

Out[5]: dict_keys(['player_details_df', 'parameters_df', 'control_sheet_df', 'ind_vs_sa_df', 'ind_vs_pak_df', 'ind_vs_
        aus_df', 'ind_vs_afg_df', 'ind_vs_wi_df', 'ind_vs_eng_df', 'ind_vs_ban_df', 'ind_vs_sl_df', 'analysis_paramete
        rs_df'])
```

```
In [6]: pd.set_option('max_columns', None)
```

In [7]:
```python
for df_name in df_names:
    print(df_name)
    display(df_dict[df_name].head(5))
```

```
player_details_df
```

	S0No	Player ID	Player Name	Type of Player	Jersey Number	Country
0	1.0	INDO175	Virat Kohli (c)	Batsmen	18.0	India
1	2.0	INDO168	Rohit Sharma (vc)	Batsmen	45.0	India
2	3.0	INDO230	Mayank Agarwal	Batsmen	16.0	India
3	4.0	INDO210	Jasprit Bumrah	Bowler	93.0	India
4	5.0	INDO211	Yuzvendra Chahal	Bowler	3.0	India

```
parameters_df
```

	Ball Type	ball symbol	Shot Type	shot symbol	Dismissal Kind	dismisal symbol	Pitch type
0	Full Toss	FT	Defend	D	Catch	Ct	Green tops

In [8]:
```python
df_dict['control_sheet_df'].head(30)
```

Out[8]:

	Unnamed: 0	Unnamed: 1	F	Correct execution of shot according to the ball	Effectiveness of the shot execution	Y	Correct execution of shot according to the ball.1	Effectiveness of the shot execution.1	FT	Correct execution of shot according to the ball.2	Effectiveness of the shot execution.2	G	Correct execution of shot according to the ball.3
0	D	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0
1	NaN	1.0	1.0	1.0	0.8	1.0	1.0	0.8	1.0	1.0	0.8	1.0	1.0
2	NaN	1.0	2.0	1.0	0.6	2.0	1.0	0.6	2.0	1.0	0.6	2.0	1.0
3	NaN	1.0	3.0	1.0	0.4	3.0	1.0	0.4	3.0	1.0	0.4	3.0	1.0
4	NaN	1.0	4.0	1.0	0.2	4.0	1.0	0.2	4.0	1.0	0.2	4.0	1.0
5	NaN	1.0	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0
6	NaN	1.0	6.0	1.0	0.2	6.0	1.0	0.2	6.0	1.0	0.2	6.0	1.0
7	NaN	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Dr	1.0	0.0	1.0	0.2	0.0	1.0	0.2	0.0	1.0	0.2	0.0	1.0
10	NaN	1.0	1.0	1.0	0.4	1.0	1.0	0.4	1.0	1.0	0.4	1.0	1.0
11	NaN	1.0	2.0	1.0	0.5	2.0	1.0	0.5	2.0	1.0	0.5	2.0	1.0
12	NaN	1.0	3.0	1.0	0.6	3.0	1.0	0.6	3.0	1.0	0.6	3.0	1.0
13	NaN	1.0	4.0	1.0	1.0	4.0	1.0	1.0	4.0	1.0	1.0	4.0	1.0
14	NaN	1.0	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0
15	NaN	1.0	6.0	1.0	1.0	6.0	1.0	1.0	6.0	1.0	1.0	6.0	1.0
16	NaN	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	C	1.0	0.0	0.0	0.2	0.0	0.0	0.2	0.0	1.0	0.2	0.0	1.0
19	NaN	1.0	1.0	0.0	0.4	1.0	0.0	0.2	1.0	1.0	0.4	1.0	1.0
20	NaN	1.0	2.0	0.0	0.4	2.0	0.0	0.2	2.0	1.0	0.4	2.0	1.0
21	NaN	1.0	3.0	0.0	0.4	3.0	0.0	0.2	3.0	1.0	0.4	3.0	1.0
22	NaN	1.0	4.0	0.0	0.5	4.0	0.0	0.2	4.0	1.0	0.5	4.0	1.0
23	NaN	1.0	5.0	0.0	0.2	5.0	0.0	0.2	5.0	1.0	0.2	5.0	1.0
24	NaN	1.0	6.0	0.0	0.5	6.0	0.0	0.2	6.0	1.0	0.5	6.0	1.0
25	NaN	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0
26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	P	1.0	0.0	0.0	0.2	0.0	0.0	0.2	0.0	0.0	0.2	0.0	1.0
28	NaN	1.0	1.0	0.0	0.2	1.0	0.0	0.2	1.0	0.0	0.2	1.0	1.0
29	NaN	1.0	2.0	0.0	0.2	2.0	0.0	0.2	2.0	0.0	0.2	2.0	1.0

In [9]:
```
df_dict['control_sheet_df'].rename({'Unnamed: 0': 'Shot Type'}, axis=1, inplace=True)
df_dict['control_sheet_df'].drop('Unnamed: 1',axis = 1, inplace = True)
```

In [10]:
```
df_dict['control_sheet_df'].head(30)
```

Out[10]:

	Shot Type	F	Correct execution of shot according to the ball	Effectiveness of the shot execution	Y	Correct execution of shot according to the ball.1	Effectiveness of the shot execution.1	FT	Correct execution of shot according to the ball.2	Effectiveness of the shot execution.2	G	Correct execution of shot according to the ball.3	Effectiveness of the shot execution.3
0	D	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0
1	NaN	1.0	1.0	0.8	1.0	1.0	0.8	1.0	1.0	0.8	1.0	1.0	0.8
2	NaN	2.0	1.0	0.6	2.0	1.0	0.6	2.0	1.0	0.6	2.0	1.0	0.6
3	NaN	3.0	1.0	0.4	3.0	1.0	0.4	3.0	1.0	0.4	3.0	1.0	0.4
4	NaN	4.0	1.0	0.2	4.0	1.0	0.2	4.0	1.0	0.2	4.0	1.0	0.2
5	NaN	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2
6	NaN	6.0	1.0	0.2	6.0	1.0	0.2	6.0	1.0	0.2	6.0	1.0	0.2
7	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	Dr	0.0	1.0	0.2	0.0	1.0	0.2	0.0	1.0	0.2	0.0	1.0	0.2
10	NaN	1.0	1.0	0.4	1.0	1.0	0.4	1.0	1.0	0.4	1.0	1.0	0.4
11	NaN	2.0	1.0	0.5	2.0	1.0	0.5	2.0	1.0	0.5	2.0	1.0	0.5
12	NaN	3.0	1.0	0.6	3.0	1.0	0.6	3.0	1.0	0.6	3.0	1.0	0.6
13	NaN	4.0	1.0	1.0	4.0	1.0	1.0	4.0	1.0	1.0	4.0	1.0	1.0
14	NaN	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2
15	NaN	6.0	1.0	1.0	6.0	1.0	1.0	6.0	1.0	1.0	6.0	1.0	1.0
16	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	C	0.0	0.0	0.2	0.0	0.0	0.2	0.0	1.0	0.2	0.0	1.0	0.2
19	NaN	1.0	0.0	0.4	1.0	0.0	0.2	1.0	1.0	0.4	1.0	1.0	0.4
20	NaN	2.0	0.0	0.4	2.0	0.0	0.2	2.0	1.0	0.4	2.0	1.0	0.5
21	NaN	3.0	0.0	0.4	3.0	0.0	0.2	3.0	1.0	0.4	3.0	1.0	0.6
22	NaN	4.0	0.0	0.5	4.0	0.0	0.2	4.0	1.0	0.5	4.0	1.0	1.0
23	NaN	5.0	0.0	0.2	5.0	0.0	0.2	5.0	1.0	0.2	5.0	1.0	0.2
24	NaN	6.0	0.0	0.5	6.0	0.0	0.2	6.0	1.0	0.5	6.0	1.0	1.0
25	NaN	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0	Out	0.0	-1.0
26	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	P	0.0	0.0	0.2	0.0	0.0	0.2	0.0	0.0	0.2	0.0	1.0	0.2
28	NaN	1.0	0.0	0.2	1.0	0.0	0.2	1.0	0.0	0.2	1.0	1.0	0.4
29	NaN	2.0	0.0	0.2	2.0	0.0	0.2	2.0	0.0	0.2	2.0	1.0	0.5

In [11]:
```
df_dict['control_sheet_df'].shape
```

Out[11]:
```
(107, 28)
```

In [12]:
```
for row_index in range(8,df_dict['control_sheet_df'].shape[0],9):
    print(row_index)
```

```
8
17
26
35
44
53
62
71
80
89
98
```

In [13]:
```
df_dict['control_sheet_df'].isna().sum()
```

Out[13]:
```
Shot Type                                           95
F                                                   11
Correct execution of shot according to the ball     11
Effectiveness of the shot execution                 11
Y                                                   11
Correct execution of shot according to the ball.1   11
Effectiveness of the shot execution.1               11
FT                                                  11
Correct execution of shot according to the ball.2   11
Effectiveness of the shot execution.2               11
G                                                   11
Correct execution of shot according to the ball.3   11
Effectiveness of the shot execution.3               11
S                                                   11
Correct execution of shot according to the ball.4   11
Effectiveness of the shot execution.4               11
Sl                                                  11
Correct execution of shot according to the ball.5   11
Effectiveness of the shot execution.5               11
L                                                   11
Correct execution of shot according to the ball.6   11
Effectiveness of the shot execution.6               11
GY                                                  11
Correct execution of shot according to the ball.7   11
Effectiveness of the shot execution.7               11
LB                                                  11
Correct execution of shot according to the ball.8   11
Effectiveness of the shot execution.8               11
dtype: int64
```

In [14]:
```
df_dict['control_sheet_df'] = df_dict['control_sheet_df'].dropna(how='all')
df_dict['control_sheet_df'].isna().sum()
```

Out[14]:
```
Shot Type                                           84
F                                                    0
Correct execution of shot according to the ball      0
Effectiveness of the shot execution                  0
Y                                                    0
Correct execution of shot according to the ball.1    0
Effectiveness of the shot execution.1                0
FT                                                   0
Correct execution of shot according to the ball.2    0
Effectiveness of the shot execution.2                0
G                                                    0
Correct execution of shot according to the ball.3    0
Effectiveness of the shot execution.3                0
S                                                    0
Correct execution of shot according to the ball.4    0
Effectiveness of the shot execution.4                0
Sl                                                   0
Correct execution of shot according to the ball.5    0
Effectiveness of the shot execution.5                0
L                                                    0
Correct execution of shot according to the ball.6    0
Effectiveness of the shot execution.6                0
GY                                                   0
Correct execution of shot according to the ball.7   0
Effectiveness of the shot execution.7                0
LB                                                   0
Correct execution of shot according to the ball.8    0
Effectiveness of the shot execution.8                0
dtype: int64
```

```
In [15]: df_dict['parameters_df']
```

	Ball Type	ball symbol	Shot Type	shot symbol	Dismissal Kind	dismisal symbol	Pitch type
0	Full Toss	FT	Defend	D	Catch	Ct	Green tops
1	Yorker	Y	OFF DRIVE	Dr	Bowled	B	Hard and bouncy
2	Full	F	Cut	C	Run Out	RO	Slow and dusty
3	Good	G	Pull	P	Leg Before Wicket	LBW	Dead and flat pitch
4	Short	S	Sweep	Sw	Stumped	St	NaN
5	slower	SL	Reverse sweep	RS	Hit Wicket	H	NaN
6	length	L	Leave	L	NaN	NaN	NaN
7	googly	GY	ONDRIVE	ODR	NaN	NaN	NaN
8	leg break	LB	FLICK	F	NaN	NaN	NaN
9	NaN	NaN	RAMP	R	NaN	NaN	NaN
10	NaN	NaN	Leg Glance	LG	NaN	NaN	NaN
11	NaN	NaN	scoop	SC	NaN	NaN	NaN

```
In [16]: shot_type = df_dict['control_sheet_df'].pop('Shot Type')
```

```
In [17]: shot_type
```

```
Out[17]: 0        D
         1      NaN
         2      NaN
         3      NaN
         4      NaN
               ...
         102    NaN
         103    NaN
         104    NaN
         105    NaN
         106    NaN
         Name: Shot Type, Length: 96, dtype: object
```

```
In [18]: shot_type.value_counts()
```

```
Out[18]: D      1
         Dr     1
         C      1
         P      1
         Sw     1
         RS     1
         ODR    1
         F      1
         R      1
         LG     1
         SC     1
         L      1
         Name: Shot Type, dtype: int64
```

```
In [19]: shot_type.isna().sum()
```

```
Out[19]: 84
```

```
In [20]: shot_type_temp = copy.deepcopy(shot_type)
         shot_type_temp
```

```
Out[20]: 0        D
         1      NaN
         2      NaN
         3      NaN
         4      NaN
               ...
         102    NaN
         103    NaN
         104    NaN
         105    NaN
         106    NaN
         Name: Shot Type, Length: 96, dtype: object
```

In [21]:
```
for index,value in shot_type_temp.items():
    if type(value) != str:
        shot_type_temp[index] = shot_type_temp[index-1]
shot_type_temp
```

Out[21]:
```
0      D
1      D
2      D
3      D
4      D
      ..
102    L
103    L
104    L
105    L
106    L
Name: Shot Type, Length: 96, dtype: object
```

In [22]:
```
shot_type_temp.value_counts()
```

Out[22]:
```
D      8
Dr     8
C      8
P      8
Sw     8
RS     8
ODR    8
F      8
R      8
LG     8
SC     8
L      8
Name: Shot Type, dtype: int64
```

In [23]:
```
shot_type = shot_type_temp
shot_type.value_counts()
```

Out[23]:
```
D      8
Dr     8
C      8
P      8
Sw     8
RS     8
ODR    8
F      8
R      8
LG     8
SC     8
L      8
Name: Shot Type, dtype: int64
```

In [24]:
```
df_dict['control_sheet_df'].shape
```

Out[24]:
```
(96, 27)
```

In [25]:
```
ball_type_df = df_dict['parameters_df'].loc[:,['Ball Type','ball symbol']].dropna()
ball_type_dict = dict(zip(ball_type_df['ball symbol'],ball_type_df['Ball Type']))
ball_type_dict
```

Out[25]:
```
{'FT': 'Full Toss',
 'Y': 'Yorker',
 'F': 'Full',
 'G': 'Good',
 'S': 'Short',
 'SL': 'slower',
 'L': 'length',
 'GY': 'googly',
 'LB': 'leg break'}
```

In [26]:
```
print(df_dict['control_sheet_df'].columns)
df_dict['control_sheet_df'].rename(columns = {'Sl':'SL'}, inplace = True)
df_dict['control_sheet_df'].columns
```

```
Index(['F', 'Correct execution of shot according to the ball',
       'Effectiveness of the shot execution', 'Y',
       'Correct execution of shot according to the ball.1',
       'Effectiveness of the shot execution.1', 'FT',
       'Correct execution of shot according to the ball.2',
       'Effectiveness of the shot execution.2', 'G',
       'Correct execution of shot according to the ball.3',
       'Effectiveness of the shot execution.3', 'S',
       'Correct execution of shot according to the ball.4',
       'Effectiveness of the shot execution.4', 'Sl',
       'Correct execution of shot according to the ball.5',
       'Effectiveness of the shot execution.5', 'L',
       'Correct execution of shot according to the ball.6',
       'Effectiveness of the shot execution.6', 'GY',
       'Correct execution of shot according to the ball.7',
       'Effectiveness of the shot execution.7', 'LB',
       'Correct execution of shot according to the ball.8',
       'Effectiveness of the shot execution.8'],
      dtype='object')
```

Out[26]:
```
Index(['F', 'Correct execution of shot according to the ball',
       'Effectiveness of the shot execution', 'Y',
       'Correct execution of shot according to the ball.1',
       'Effectiveness of the shot execution.1', 'FT',
       'Correct execution of shot according to the ball.2',
       'Effectiveness of the shot execution.2', 'G',
       'Correct execution of shot according to the ball.3',
       'Effectiveness of the shot execution.3', 'S',
       'Correct execution of shot according to the ball.4',
       'Effectiveness of the shot execution.4', 'SL',
       'Correct execution of shot according to the ball.5',
       'Effectiveness of the shot execution.5', 'L',
       'Correct execution of shot according to the ball.6',
       'Effectiveness of the shot execution.6', 'GY',
       'Correct execution of shot according to the ball.7',
       'Effectiveness of the shot execution.7', 'LB',
       'Correct execution of shot according to the ball.8',
       'Effectiveness of the shot execution.8'],
      dtype='object')
```

In [27]:
```
def generate_ballwise_df(df = df_dict['control_sheet_df']):
    for col_index in range(0,df.shape[1],3):
        ball_symbol = df.iloc[:,col_index:col_index+3].columns[0]
        ball_type = ball_type_dict[ball_symbol]

        df_dict[f'{ball_type}_shots_df'] = df.iloc[:,col_index:col_index+3]
        df_dict[f'{ball_type}_shots_df'].columns = ['Runs Scored', 'Correct execution of shot according to the
        df_dict[f'{ball_type}_shots_df'].insert(0, 'Ball Type', ball_symbol)
        df_dict[f'{ball_type}_shots_df'].insert(1,'Shot Type', shot_type)
```

In [28]:
```
generate_ballwise_df()
```

```
In [29]: for df_name in df_dict.keys():
             print(df_name)
             display(df_dict[df_name].head(10))
```

	Ball Type	ball symbol	Shot Type	shot symbol	Dismissal Kind	dismisal symbol	Pitch type
0	Full Toss	FT	Defend	D	Catch	Ct	Green tops
1	Yorker	Y	OFF DRIVE	Dr	Bowled	B	Hard and bouncy
2	Full	F	Cut	C	Run Out	RO	Slow and dusty
3	Good	G	Pull	P	Leg Before Wicket	LBW	Dead and flat pitch
4	Short	S	Sweep	Sw	Stumped	St	NaN
5	slower	SL	Reverse sweep	RS	Hit Wicket	H	NaN
6	length	L	Leave	L	NaN	NaN	NaN
7	googly	GY	ONDRIVE	ODR	NaN	NaN	NaN
8	leg break	LB	FLICK	F	NaN	NaN	NaN
9	NaN	NaN	RAMP	R	NaN	NaN	NaN

```
In [30]: df_dict['analysis_parameters_df'] = df_dict['analysis_parameters_df'].dropna(how = 'all')
         df_dict['analysis_parameters_df'].reset_index(drop = True, inplace = True)
```

```
In [31]: pd.set_option('display.max_colwidth', None)
         df_dict['analysis_parameters_df']
```

Out[31]:

	Analysis Parameters	Formulae	Parameters	Unnamed: 3	Unnamed: 4
0	Best off side player	Highest average of effectiveness of shot of all off side shots	Dr,C and RS	The highest percentage of shots in Dr, C and RS	O= ((no of Dr+C+RS)/ total number of balls faced) * ((sum of effectiveness of Dr +C+RS)/no of Dr+C+RS)
1	Best Leg side player	Highest average of effectiveness of shot of all Leg side shots	P, ODR, LG, F and S	The highest percentage of shots in P, ODR, LG, F and S	O= ((no of P+ODR+LG+F+S)/ total number of balls faced) * ((sum of effectiveness of P+ODR+LG+F+S)/no of P+ODR+LG+F+S)
2	Best player for bouncy tracks	Highest average of effectiveness of shot of all Bouncy pitch shots	D, P, C and L	The highest percentage of shots in D, P, C and L	O= ((no of D+P+C+L)/ total number of balls faced) * ((sum of effectiveness of D+P+C+L)/no of D+P+C+L)
3	Best player for high scoring matches	Highest average of effectiveness of shot of all agressive shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)
4	Best defencive player	NaN	D and L	The highest percentage of shots in D and L	O= ((no of D+L)/ total number of balls faced) * ((sum of effectiveness of D+L)/no of D+L)
5	Best spin ball player	Highest average of effectiveness of shot of all spin ball shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)
6	The player with the best control	Highest control of shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)

```
In [32]: def unique_values(df):
             unique_vals = [(col,list(df[col].unique())) for col in df.columns if col!='Sno']
             return unique_vals
```

In [33]:
```python
for df_name in df_dict.keys():
    if 'vs' not in df_name:
        continue
    print(df_name)
    for col,vals in unique_values(df_dict[df_name]):
        print(f'{col} : {vals}')
    print()
```

```
ind_vs_sa_df
Player Name : ['Hardik Pandya', 'Rohit Sharma', 'MS Dhoni', 'KL Rahul', 'Virat Kohli', 'Shikhar Dhawan']
Player ID : ['IND0215', 'IND0168', 'IND0157', 'IND0213', 'IND0175', 'IND0188']
Match ID : ['WC19M08']
Pitch Type : [nan]
Bowler : ['Andile Phehlukwayo', 'Chris Morris', 'Kagiso Rabada', 'Imran Tahir', 'Tabraiz Shamsi']
Player ID.1 : ['SA0118', 'SA0110', 'SA0114', 'SA0102', 'SA0116']
Over No : [47.0, 46.0, 45.0, 44.0, 43.0, 42.0, 41.0, 40.0, 39.0, 38.0, 37.0, 36.0, 35.0, 34.0, 33.0, 32.0,
31.0, 30.0, 29.0, 28.0, 27.0, 26.0, 25.0, 24.0, 23.0, 22.0, 21.0, 20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 14.
0, 13.0, 12.0, 11.0, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0, 0.0]
Ball No : [3.0, 2.0, 1.0, 6.0, 5.0, 4.0]
Ball type : ['F', 'S', 'L', 'FT', 'Y', 'FL']
Type of Shot : ['C', 'L', 'DR', 'P', 'SW', 'ODR', 'D', 'S', 'LG', 'RS']
Wide : [nan, 'Yes', 'Wide']
No ball : [nan]
Hit the Bat : [1.0, nan, 0.0]
Control : [0.0, 1.0, -1.0]
X : [0.5, 0.0, 1.0, 0.4, 0.2, -1.0, 0.6, 0.8]
Runs : [4.0, '1w', 0.0, 2.0, 1.0, 'W', 3.0, '1lb', 6.0]
```

In [34]:
```python
for df_name in df_dict.keys():
    if 'vs' in df_name:
        continue
    print(df_name)
    for col,vals in unique_values(df_dict[df_name]):
        print(f'{col} : {vals}')
    print()
```

```
n', 'Liam Dawson', 'Liam Plunkett', 'Adil Rashid', 'Joe Root', 'Jason Roy', 'Ben Stokes', 'James Vince',
'Chris Woakes', 'Mark Wood', 'Joe Denly', 'Alex Hales', 'David Willey', 'Mashrafe Mortaza (c)', 'Abu Jaye
d', 'Litton Das †', 'Mahmudullah', 'Mehidy Hasan Miraz', 'Mohammad Mithun', 'Mohammad Saifuddin', 'Mosadde
k Hossain', 'Mushfiqur Rahim †', 'Mustafizur Rahman', 'Rubel Hossain', 'Sabbir Rahman', 'Shakib Al Hasan
(vc)', 'Soumya Sarkar', 'Tamim Iqbal', 'Dimuth Karunaratne (c)', 'Dhananjaya de Silva', 'Avishka Fernand
o', 'Suranga Lakmal', 'Lasith Malinga', 'Angelo Mathews', 'Kusal Mendis †', 'Jeevan Mendis', 'Kusal Perera
†', 'Thisara Perera', 'Milinda Siriwardana', 'Lahiru Thirimanne', 'Isuru Udana', 'Jeffrey Vandersay', 'Nuw
an Pradeep', 'Kasun Rajitha']
Type of Player : ['Batsmen', 'Bowler', 'Wicket Keeper', 'All Rounder', 'Batting All Rounder', 'Bowling All
Rounder', 'Batter']
Jersey Number : [18.0, 45.0, 16.0, 93.0, 3.0, 7.0, 8.0, 21.0, 79.0, 23.0, 15.0, 11.0, 17.0, 33.0, 1.0, 42.
0, 59.0, 90.0, 12.0, 14.0, 99.0, 4.0, 10.0, 2.0, 22.0, 29.0, 25.0, 26.0, 72.0, 20.0, 5.0, 65.0, 6.0, 30.0,
54.0, 67.0, 32.0, 47.0, 49.0, 56.0, 13.0, 31.0, 63.0, 9.0, 60.0, 39.0, 89.0, 87.0, 40.0, 41.0, 83.0, 44.0,
66.0, 50.0, nan, 88.0, 19.0, 55.0, 77.0, 97.0, 46.0, 85.0, 24.0, 51.0, 95.0, 53.0, 74.0, 34.0, 75.0, 28.0,
82.0, 69.0, 86.0, 57.0]
Country : ['India', 'South Africa', 'Australia', 'Pakistan', 'Afghanistan', 'West Indies', 'England', 'Ban
gladesh', 'Srilanka']

parameters_df
Ball Type : ['Full Toss', 'Yorker', 'Full', 'Good', 'Short', 'slower', 'length', 'googly', 'leg break', na
```

In [35]:
```
for df_name in df_dict.keys():
    if 'vs' not in df_name:
        continue
    print(df_name)
    print(df_dict[df_name][df_dict[df_name]['Dismissal kind'].notnull()])
```

```
ind_vs_sa_df
       Sno      Player Name Player ID Match ID  Pitch Type              Bowler  \
9     10.0         MS Dhoni   IND0157  WC19M08         NaN        Chris Morris
98    99.0         KL Rahul   IND0213  WC19M08         NaN       Kagiso Rabada
195  196.0      Virat Kohli   IND0175  WC19M08         NaN  Andile Phehlukwayo
258  259.0   Shikhar Dhawan   IND0188  WC19M08         NaN       Kagiso Rabada

    Player ID.1  Over No  Ball No Ball type Type of Shot Wide  No ball  \
9        SA0110     46.0      1.0         S           DR  NaN      NaN
98       SA0114     31.0      3.0         L           DR  NaN      NaN
195      SA0118     15.0      3.0         S           DR  NaN      NaN
258      SA0114      5.0      1.0         S            C  NaN      NaN

     Hit the Bat  Control    X Runs  Wicket Dismissal kind Caught/Bowled By
9            NaN      0.0 -1.0    W  Wicket              C           SA0110
98           NaN      1.0 -1.0    W  Wicket              C           SA0101
195          NaN      1.0 -1.0    W  Wicket              C           SA0101
258          NaN      0.0 -1.0    W  Wicket              C           SA0105
ind_vs_pak_df
```

In [36]:
```
df_dict['ind_vs_afg_df']['Dismissal kind'] = df_dict['ind_vs_afg_df']['Dismissal kind'].replace('Leg Stump', 'B
df_dict['ind_vs_afg_df']['Dismissal kind'] = df_dict['ind_vs_afg_df']['Dismissal kind'].replace('Stump Out', 'S
```

In [37]:
```
df_dict['ind_vs_afg_df'].loc[df_dict['ind_vs_afg_df']['Dismissal kind'].notnull() & df_dict['ind_vs_afg_df']['W
```

In [38]:
```
def clean_missing_values(df):
    cols_nan = [col_name for col_name, nan_count in df.isna().sum().items() if nan_count!=0 ]
    unary_cols = ['Pitch Type', 'Match ID']
    binary_cols = ['Wide', 'No ball', 'Wicket', 'Hit the Bat']

    for col in cols_nan:
        if col in unary_cols:
            df.drop(col, axis = 1, inplace = True)
            continue
        elif col in binary_cols:
            df[col] = df[col].notnull().astype('float')
            continue

        df[col].fillna(value = 0, inplace = True)

    return df
```

In [39]:
```
def drop_incorrect_data(df,column_name,child_values,parent_values):
    if all(value in parent_values for x in child_values):
        return

    incorrect_values = [val for val in child_values if val not in parent_values]
    df.drop(df[df[column_name].isin(incorrect_values)].index, inplace=True)
    return
```

In [40]:
```
def validate_column_values(df):
    ball_symbol_list = df_dict['parameters_df']['ball symbol'].dropna().tolist()
    ball_symbol_list = list(map(lambda x: x.upper(),ball_symbol_list))

    shot_symbol_list = df_dict['parameters_df']['shot symbol'].dropna().tolist()
    shot_symbol_list = list(map(lambda x: x.upper(),shot_symbol_list))

    dismissal_symbol_list = df_dict['parameters_df']['dismisal symbol'].dropna().tolist()
    dismissal_symbol_list = list(map(lambda x: x.upper(),dismissal_symbol_list))
    dismissal_dict = dict(zip(df_dict['parameters_df']['Dismissal Kind'].dropna(),df_dict['parameters_df']['dis

    control_values_list = [0.0,1.0]
    effectiveness_values_list = [x/10.0 for x in range(0,11)] + [-1.0]

    for col_name, col_values in unique_values(df):
        if 'shot' in col_name.lower():
            col_values = list(map(lambda x: x.upper(), col_values))
            drop_incorrect_data(df,col_name,col_values,shot_symbol_list)
        elif col_name.lower() == 'ball type':
            col_values = list(map(lambda x: x.upper(), col_values))
            drop_incorrect_data(df,col_name,col_values,ball_symbol_list)
        elif 'control' in col_name.lower():
            df[col_name] = df[col_name].replace(-1.0,0.0) if -1.0 in col_values else df[col_name]
            col_values.remove(-1.0) if -1.0 in col_values else col_values
            drop_incorrect_data(df,col_name,col_values,control_values_list)
        elif 'effectiveness' in col_name.lower():
            drop_incorrect_data(df,col_name,col_values,effectiveness_values_list)
        elif 'dismissal' in col_name.lower():
            col_values.remove(0)
            df[col_name] = df[col_name].replace('C','Ct') if 'C' in col_values else df[col_name]
            col_values = ['Ct' if col_value == 'C' else col_value for col_value in col_values]

            for col_value in col_values:
                df[col_name] = df[col_name].replace(col_value,dismissal_dict[col_value]) if len(col_value) > 3

            col_values = [dismissal_dict[col_value] if len(col_value) > 3 else col_value for col_value in col_v
            col_values = [col_value.upper() for col_value in col_values]
            drop_incorrect_data(df,col_name,col_values,dismissal_symbol_list)

    return df
```

In [41]:
```
def clean_df(df):
    df.drop('Sno', axis = 1, inplace = True)
    df = df.dropna(how = 'all')
    df.reset_index(drop = True, inplace = True)
    df.rename(columns = {'Player Name':'Batsman Name', 'Player ID':'Batsman Player ID', 'Bowler':'Bowler Name',

    non_nan_df = clean_missing_values(df)
    validated_df = validate_column_values(non_nan_df)

    return validated_df
```

In [42]:
```
length = 0

for df_name in df_dict.keys():
    if 'vs' not in df_name:
        continue
    length += df_dict[df_name].shape[0]

print(length)
```

```
2397
```

In [43]:
```
matches_df = pd.DataFrame()

for df_name in df_dict.keys():
    if 'vs' not in df_name:
        continue
    cleaned_df = clean_df(df_dict[df_name])
    matches_df = matches_df.append(cleaned_df, ignore_index = True)

matches_df.shape
```

Out[43]:
```
(2387, 18)
```

In [44]:
```python
for df_name in df_dict.keys():
    if 'vs' not in df_name:
        continue
    print(df_name)
    for col_name, col_values in unique_values(df_dict[df_name]):
        print(col_name, col_values)
    print()
```

```
Runs ['w', 1.0, 0.0, 4.0, 2.0, '4lb', 3.0, '1w', '1lb', 6.0, '5nb']
Wicket ['Wicket', nan]
Dismissal kind ['Bowled', 'Run Out', nan, 'Catch']
Caught/Bowled By ['BANO118', 'BANO81', nan, 'BANO82', 'BANO119', 'BANO115', 'BANO95', 'BANO117']

ind_vs_sl_df
Player Name ['Hardik Pandya', 'Virat Kohli', 'Rishabh Pant', 'KL Rahul', 'Rohit Sharma']
Player ID ['INDO215', 'INDO175', 'INDO224', 'INDO213', 'INDO168']
Match ID ['WC19M44']
Pitch Type [nan]
Bowler ['Isuru Udana', 'Lasith Malinga', 'Kusal Perera', 'Dhananjaya de Silva', 'Kasun Rajitha']
Player ID.1 ['SRIO152', 'SRIO123', 'SRIO155', 'SRIO169', 'SRIO146']
Over No [43.0, 42.0, 41.0, 40.0, 39.0, 38.0, 37.0, 36.0, 35.0, 34.0, 33.0, 32.0, 31.0, 30.0, 29.0, 28.0, 2
7.0, 26.0, 25.0, 24.0, 23.0, 22.0, 21.0, 20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 14.0, 13.0, 12.0, 11.0, 10.0,
9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0, 0.0]
Ball No [3.0, 2.0, 1.0, 6.0, 5.0, 4.0]
Ball type ['SL', 'G', 'L', 'FT', 'S', 'Y', 'F']
Type of Shot ['DR', 'P', 'ODR', 'LG', 'L', 'SW', 'C', 'F', 'D', '`F']
Wide [nan, 'Yes']
No ball [nan]
```

In [45]:
```python
matches_df
```

Out[45]:

	Batsman Name	Batsman Player ID	Match ID	Bowler Name	Bowler Player ID	Over No	Ball No	Ball type	Type of Shot	Wide	No ball	Hit the Bat	Control	Effectiveness	Runs	Wicke
0	Hardik Pandya	INDO215	WC19M08	Andile Phehlukwayo	SAO118	47.0	3.0	F	C	0.0	0.0	1.0	0.0	0.5	4.0	0.
1	Hardik Pandya	INDO215	WC19M08	Andile Phehlukwayo	SAO118	47.0	3.0	S	L	1.0	0.0	0.0	1.0	0.0	1w	0.
2	Hardik Pandya	INDO215	WC19M08	Andile Phehlukwayo	SAO118	47.0	2.0	S	L	0.0	0.0	1.0	1.0	1.0	0.0	0.
3	Hardik Pandya	INDO215	WC19M08	Andile Phehlukwayo	SAO118	47.0	1.0	S	C	0.0	0.0	1.0	1.0	0.5	2.0	0.
4	Hardik Pandya	INDO215	WC19M08	Chris Morris	SAO110	46.0	6.0	S	DR	0.0	0.0	1.0	1.0	0.4	1.0	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
2382	KL Rahul	INDO213	WC19M44	Lasith Malinga	SRIO123	0.0	5.0	L	L	0.0	0.0	1.0	1.0	1.0	0.0	0.
2383	KL Rahul	INDO213	WC19M44	Lasith Malinga	SRIO123	0.0	4.0	L	DR	0.0	0.0	1.0	1.0	0.2	0.0	0.
2384	KL Rahul	INDO213	WC19M44	Lasith Malinga	SRIO123	0.0	3.0	F	DR	0.0	0.0	1.0	1.0	0.2	0.0	0.
2385	Rohit Sharma	INDO168	WC19M44	Lasith Malinga	SRIO123	0.0	2.0	G	ODR	0.0	0.0	1.0	1.0	0.4	1.0	0.
2386	KL Rahul	INDO213	WC19M44	Lasith Malinga	SRIO123	0.0	1.0	L	ODR	0.0	0.0	1.0	1.0	0.4	1.0	0.

2387 rows × 18 columns

In [46]:
```
for col_name, col_values in unique_values(matches_df):
    print(col_name,col_values)
```

```
Batsman Name ['Hardik Pandya', 'Rohit Sharma', 'MS Dhoni', 'KL Rahul', 'Virat Kohli', 'Shikhar Dhawan', 'Kedar
Jadhav', 'Vijay Shankar', 'Shikar Dhawan', 'Jasprit Bumrah', 'Kuldeep Yadav', 'Mohammed Shami', 'Kedhar Jadha
v', 'Rishabh Pant', 'Bhuvneshwar Kumar', 'Dinesh Karthik']
Batsman Player ID ['INDO215', 'INDO168', 'INDO157', 'INDO213', 'INDO175', 'IND0188', 'INDO205', 'INDO226', 'IN
DO210', 'INDO217', 'INDO195', 'INDO224', 'INDO194', 'INDO156']
Match ID ['WC19M08', 'WC19M22', 'WC19M14', 'WC19M28', 'WC19M34', 'WC19M38', 'WC19M40', 'WC19M44']
Bowler Name ['Andile Phehlukwayo', 'Chris Morris', 'Kagiso Rabada', 'Imran Tahir', 'Tabraiz Shamsi', 'Mohammad
Amir', 'Wahab Riaz', 'Hasan Ali', 'Shadab Khan', 'Imad Wasim', 'Mohammad Hafeez', 'Shoaib Malik', 'Marcus Stoi
nis', 'Mitchell Starc', 'Pat Cummins', 'Adam Zampa', 'Nathan Coulter-Nile', 'Glenn Maxwell', 'Gulbadin Naib',
'Aftab Alam', 'Rashid Khan', 'Mujeeb Ur Rahman', 'Mohammad Nabi', 'Rahmat Shah', 'Oshane Thomas', 'Sheldon Cot
trell', 'Carlos Brathwaite', 'Jason Holder', 'Fabian Allen', 'Kemar Roach', 'Chris Woakes', 'Jofra Archer', 'M
ark Wood', 'Liam Plunkett', 'Adil Rashid', 'Ben Stokes', 'Mustafizur Rahman', 'Mohammad Saifuddin', 'Shakib Al
Hasan', 'Rubel Hossain', 'Soumya Sarkar', 'Mosaddek Hossain', 'Mashrafe Mortaza', 'Isuru Udana', 'Lasith Malin
ga', 'Kusal Perera', 'Dhananjaya de Silva', 'Kasun Rajitha']
Bowler Player ID ['SAO118', 'SAO110', 'SAO114', 'SAO102', 'SAO116', 'PAKO173', 'PAKO168', 'PAKO209', 'PAKO21
1', 'PAKO204', 'PAKO144', 'PAKO128', 'AUS0209', 'AUS0185', 'AUS0189', 'AUS0212', 'AUSO204', 'AUS0196', 'AFGO2
4', 'AFGO17', 'AFGO36', 'AFGO43', 'AFGO7', 'AFGO29', 'WIO186', 'WIO169', 'WIO161', 'WIO166', 'WIO188', 'WIO14
4', 'ENGO217', 'ENGO252', 'ENGO241', 'ENGO190', 'ENGO210', 'ENGO221', 'BANO118', 'BANO125', 'BANO82', 'BANO9
5', 'BANO115', 'BANO119', 'BANO54', 'SRIO152', 'SRIO123', 'SRIO155', 'SRIO169', 'SRIO146']
Over No [47.0, 46.0, 45.0, 44.0, 43.0, 42.0, 41.0, 40.0, 39.0, 38.0, 37.0, 36.0, 35.0, 34.0, 33.0, 32.0, 31.0,
30.0, 29.0, 28.0, 27.0, 26.0, 25.0, 24.0, 23.0, 22.0, 21.0, 20.0, 19.0, 18.0, 17.0, 16.0, 15.0, 14.0, 13.0, 1
2.0, 11.0, 10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0, 0.0, 49.0, 48.0]
Ball No [3.0, 2.0, 1.0, 6.0, 5.0, 4.0]
Ball type ['F', 'S', 'L', 'FT', 'Y', 'GY', 'LB', 'G', 'SL']
Type of Shot ['C', 'L', 'DR', 'P', 'SW', 'ODR', 'D', 'LG', 'RS', 'F', 'SC', 'R']
Wide [0.0, 1.0]
No ball [0.0, 1.0]
Hit the Bat [1.0, 0.0]
Control [0.0, 1.0]
Effectiveness [0.5, 0.0, 1.0, 0.4, 0.2, -1.0, 0.6, 0.8, 0.3]
Runs [4.0, '1w', 0.0, 2.0, 1.0, 'W', 3.0, '1lb', 6.0, '1b', '4lb', '5nb']
Wicket [0.0, 1.0]
Dismissal kind [0, 'Ct', 'B', 'St', 'LBW', 'RO']
Caught/Bowled By [0, 'SAO110', 'SAO101', 'SAO105', 'PAKO159', 'PAKO203', 'PAKO168', 'AUS0189', 'AUS0209', 'AUS
O197', 'AUSO194', 'AUSO223', 'AFGO9', 'AFGO24', 'AFGO45', 'AFGO29', 'AFGO34', 'AFGO43', 'WIO175', 'WIO188', 'W
IO146', 'WIO166', 'ENGO239', 'ENGO217', 'ENGO226', 'BANO118', 'BANO81', 'BANO82', 'BANO119', 'BANO115', 'BANO9
5', 'BANO117', 'SRIO152', 'SRIO155', 'SRIO137']
```

In [47]:
```
matches_df.nunique()
```

Out[47]:
```
Batsman Name         16
Batsman Player ID    14
Match ID              8
Bowler Name          48
Bowler Player ID     48
Over No              50
Ball No               6
Ball type             9
Type of Shot         12
Wide                  2
No ball               2
Hit the Bat           2
Control               2
Effectiveness         9
Runs                 12
Wicket                2
Dismissal kind        6
Caught/Bowled By     35
dtype: int64
```

In [48]:
```
player_info = dict(zip(matches_df['Batsman Name'],matches_df['Batsman Player ID']))
player_info
```

Out[48]:
```
{'Hardik Pandya': 'IND0215',
 'Rohit Sharma': 'IND0168',
 'MS Dhoni': 'IND0157',
 'KL Rahul': 'IND0213',
 'Virat Kohli': 'IND0175',
 'Shikhar Dhawan': 'IND0188',
 'Kedar Jadhav': 'IND0205',
 'Vijay Shankar': 'IND0226',
 'Shikar Dhawan': 'IND0188',
 'Jasprit Bumrah': 'IND0210',
 'Kuldeep Yadav': 'IND0217',
 'Mohammed Shami': 'IND0195',
 'Kedhar Jadhav': 'IND0205',
 'Rishabh Pant': 'IND0224',
 'Bhuvneshwar Kumar': 'IND0194',
 'Dinesh Karthik': 'IND0156'}
```

In [49]:
```
len(player_info)
```

Out[49]:
```
16
```

In [50]:
```
matches_df['Batsman_name'] = matches_df['Batsman Name'].replace('Kedar Jadhav','Kedhar Jadhav', inplace = True)
matches_df['Batsman_name'] = matches_df['Batsman Name'].replace('Shikar Dhawan','Shikhar Dhawan', inplace = Tru
```

In [51]:
```
matches_df.nunique()
```

Out[51]:
```
Batsman Name         14
Batsman Player ID    14
Match ID              8
Bowler Name          48
Bowler Player ID     48
Over No              50
Ball No               6
Ball type             9
Type of Shot         12
Wide                  2
No ball               2
Hit the Bat           2
Control               2
Effectiveness         9
Runs                 12
Wicket                2
Dismissal kind        6
Caught/Bowled By     35
Batsman_name          0
dtype: int64
```

In [52]:
```
df_dict['analysis_parameters_df']
```

Out[52]:

	Analysis Parameters	Formulae	Parameters	Unnamed: 3	Unnamed: 4
0	Best off side player	Highest average of effectiveness of shot of all off side shots	Dr,C and RS	The highest percentage of shots in Dr, C and RS	O= ((no of Dr+C+RS)/ total number of balls faced) * ((sum of effectiveness of Dr +C+RS)/no of Dr+C+RS)
1	Best Leg side player	Highest average of effectiveness of shot of all Leg side shots	P, ODR, LG, F and S	The highest percentage of shots in P, ODR, LG, F and S	O= ((no of P+ODR+LG+F+S)/ total number of balls faced) * ((sum of effectiveness of P+ODR+LG+F+S)/no of P+ODR+LG+F+S)
2	Best player for bouncy tracks	Highest average of effectiveness of shot of all Bouncy pitch shots	D, P, C and L	The highest percentage of shots in D, P, C and L	O= ((no of D+P+C+L)/ total number of balls faced) * ((sum of effectiveness of D+P+C+L)/no of D+P+C+L)
3	Best player for high scoring matches	Highest average of effectiveness of shot of all agressive shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)
4	Best defencive player	NaN	D and L	The highest percentage of shots in D and L	O= ((no of D+L)/ total number of balls faced) * ((sum of effectiveness of D+L)/no of D+L)
5	Best spin ball player	Highest average of effectiveness of shot of all spin ball shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)
6	The player with the best control	Highest control of shots	D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	The highest percentage of shots in D, Dr, C, P, Sw, RS, L, ODR, F, R, LG and SC	O= ((no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/ total number of balls faced) * ((sum of effectiveness of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)/no of D+Dr+ C+P+Sw+RS+L+ODR+F+ R,+LG +SC)

In [53]:
```
def off_side_score(df = matches_df):
    off_side_shots = ['DR','C','RS']

    off_side_shots_count = df[df['Type of Shot'].isin(off_side_shots)].groupby('Batsman Name').agg(off_side_sho
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    off_side_score_df = off_side_shots_count.merge(grouped_df,on = 'Batsman Name')
    off_side_score_df = off_side_score_df.reset_index()

    off_side_score_df['Highest percentage of off side shots'] = (off_side_score_df['off_side_shots_count']/off_
    off_side_score_df['Highest effectiveness average of off side shots'] = off_side_score_df['sum_effectiveness
    off_side_score_df['off_side_score'] = (off_side_score_df['Highest percentage of off side shots']/100) * off_

    off_side_score_df = off_side_score_df.sort_values(by=['off_side_score'], ascending=False)

    return off_side_score_df.loc[:,['Batsman Name','Highest percentage of off side shots','Highest effectivenes
```

In [54]:
```
off_side_score()
```

Out[54]:

	Batsman Name	Highest percentage of off side shots	Highest effectiveness average of off side shots	off_side_score
0	Bhuvneshwar Kumar	50.000000	0.400000	0.200000
2	Hardik Pandya	42.465753	0.422581	0.179452
8	Shikhar Dhawan	43.089431	0.383019	0.165041
7	Rohit Sharma	42.057489	0.378417	0.159153
10	Virat Kohli	41.880342	0.320408	0.134188
6	Rishabh Pant	31.168831	0.404167	0.125974
3	KL Rahul	40.086207	0.280108	0.112284
5	MS Dhoni	35.245902	0.306977	0.108197
1	Dinesh Karthik	44.444444	0.200000	0.088889
4	Kedhar Jadhav	26.923077	0.321429	0.086538
9	Vijay Shankar	30.769231	0.266667	0.082051

In [55]:
```
def leg_side_score(df = matches_df):
    leg_side_shots = ['P','ODR','LG','F','S']

    leg_side_shots_count = df[df['Type of Shot'].isin(leg_side_shots)].groupby('Batsman Name').agg(leg_side_sho
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    leg_side_score_df = leg_side_shots_count.merge(grouped_df,on = 'Batsman Name')
    leg_side_score_df = leg_side_score_df.reset_index()

    leg_side_score_df['Highest percentage of leg side shots'] = (leg_side_score_df['leg_side_shots_count']/leg_
    leg_side_score_df['Highest effectiveness average of leg side shots'] = leg_side_score_df['sum_effectiveness
    leg_side_score_df['leg_side_score'] = (leg_side_score_df['Highest percentage of leg side shots']/100) * leg

    leg_side_score_df = leg_side_score_df.sort_values(by=['leg_side_score'], ascending=False)

    return leg_side_score_df.loc[:,['Batsman Name','Highest percentage of leg side shots','Highest effectivenes
```

In [56]:
```
leg_side_score()
```

Out[56]:

	Batsman Name	Highest percentage of leg side shots	Highest effectiveness average of leg side shots	leg_side_score
3	Jasprit Bumrah	100.000000	0.400000	0.400000
6	Kuldeep Yadav	100.000000	0.300000	0.300000
9	Rishabh Pant	55.844156	0.400000	0.223377
7	MS Dhoni	43.852459	0.412150	0.180738
2	Hardik Pandya	44.520548	0.372308	0.165753
5	Kedhar Jadhav	49.038462	0.323529	0.158654
12	Vijay Shankar	35.897436	0.428571	0.153846
13	Virat Kohli	39.743590	0.371505	0.147650
4	KL Rahul	35.775862	0.398193	0.142457
10	Rohit Sharma	32.829047	0.393548	0.129198
1	Dinesh Karthik	55.555556	0.200000	0.111111
11	Shikhar Dhawan	25.203252	0.332258	0.083740
8	Mohammed Shami	83.333333	-0.200000	-0.166667
0	Bhuvneshwar Kumar	25.000000	-1.000000	-0.250000

In [57]:
```
def bouncy_track_score(df = matches_df):
    bouncy_track_shots = ['D','P','C','L']

    bouncy_track_shots_count = df[df['Type of Shot'].isin(bouncy_track_shots)].groupby('Batsman Name').agg(boun
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    bouncy_track_score_df = bouncy_track_shots_count.merge(grouped_df,on = 'Batsman Name')
    bouncy_track_score_df = bouncy_track_score_df.reset_index()

    bouncy_track_score_df['Highest percentage of bouncy track shots'] = (bouncy_track_score_df['bouncy_track_sh
    bouncy_track_score_df['Highest effectiveness average of bouncy track shots'] = bouncy_track_score_df['sum_e
    bouncy_track_score_df['bouncy_track_score'] = (bouncy_track_score_df['Highest percentage of bouncy track sh

    bouncy_track_score_df = bouncy_track_score_df.sort_values(by=['bouncy_track_score'], ascending=False)

    return bouncy_track_score_df.loc[:,['Batsman Name','Highest percentage of bouncy track shots','Highest effe
```

In [58]:
```
bouncy_track_score()
```

Out[58]:

	Batsman Name	Highest percentage of bouncy track shots	Highest effectiveness average of bouncy track shots	bouncy_track_score
9	Shikhar Dhawan	50.406504	0.622581	0.313821
8	Rohit Sharma	46.898638	0.658065	0.308623
10	Vijay Shankar	42.307692	0.715152	0.302564
3	KL Rahul	40.732759	0.669841	0.272845
5	MS Dhoni	41.803279	0.592157	0.247541
4	Kedhar Jadhav	39.423077	0.600000	0.236538
11	Virat Kohli	37.820513	0.546893	0.206838
2	Hardik Pandya	36.986301	0.490741	0.181507
7	Rishabh Pant	25.974026	0.350000	0.090909
0	Bhuvneshwar Kumar	50.000000	0.000000	0.000000
6	Mohammed Shami	50.000000	-0.066667	-0.033333
1	Dinesh Karthik	33.333333	-0.200000	-0.066667

In [59]:
```
def aggressive_shots_score(df = matches_df):
    aggressive_shots = ['Dr', 'C', 'P', 'Sw', 'RS', 'ODR', 'F', 'R', 'LG', 'SC']

    aggressive_shots_count = df[df['Type of Shot'].isin(aggressive_shots)].groupby('Batsman Name').agg(aggressi
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    aggressive_score_df = aggressive_shots_count.merge(grouped_df,on = 'Batsman Name')
    aggressive_score_df = aggressive_score_df.reset_index()

    aggressive_score_df['Highest percentage of aggressive shots'] = (aggressive_score_df['aggressive_shots_coun
    aggressive_score_df['Highest effectiveness average of aggressive shots'] = aggressive_score_df['sum_effecti
    aggressive_score_df['aggressive_score'] = (aggressive_score_df['Highest percentage of aggressive shots']/10

    aggressive_score_df = aggressive_score_df.sort_values(by=['aggressive_score'], ascending=False)

    return aggressive_score_df.loc[:,['Batsman Name','Highest percentage of aggressive shots','Highest effectiv
```

In [60]:
```
aggressive_shots_score()
```

Out[60]:

	Batsman Name	Highest percentage of aggressive shots	Highest effectiveness average of aggressive shots	aggressive_score
3	Jasprit Bumrah	100.000000	0.400000	0.400000
6	Kuldeep Yadav	100.000000	0.300000	0.300000
9	Rishabh Pant	59.740260	0.395652	0.236364
2	Hardik Pandya	57.534247	0.388095	0.223288
7	MS Dhoni	51.229508	0.385600	0.197541
10	Rohit Sharma	47.049924	0.398071	0.187292
5	Kedhar Jadhav	54.807692	0.328070	0.179808
13	Virat Kohli	50.427350	0.346610	0.174786
4	KL Rahul	44.612069	0.363768	0.162284
12	Vijay Shankar	43.589744	0.358824	0.156410
1	Dinesh Karthik	77.777778	0.200000	0.155556
11	Shikhar Dhawan	46.341463	0.315789	0.146341
0	Bhuvneshwar Kumar	25.000000	-1.000000	-0.250000
8	Mohammed Shami	100.000000	-0.333333	-0.333333

In [61]:
```
def defense_score(df = matches_df):
    defense_shots = ['D','L']

    defense_shots_count = df[df['Type of Shot'].isin(defense_shots)].groupby('Batsman Name').agg(defense_shots_
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    defense_score_df = defense_shots_count.merge(grouped_df,on = 'Batsman Name')
    defense_score_df = defense_score_df.reset_index()

    defense_score_df['Highest percentage of defense shots'] = (defense_score_df['defense_shots_count']/defense_
    defense_score_df['Highest effectiveness average of defense shots'] = defense_score_df['sum_effectiveness']/
    defense_score_df['defense_score'] = (defense_score_df['Highest percentage of defense shots']/100) * defense

    defense_score_df = defense_score_df.sort_values(by=['defense_score'], ascending=False)

    return defense_score_df.loc[:,['Batsman Name','Highest percentage of defense shots','Highest effectiveness
```

In [62]:
```
defense_score()
```

Out[62]:

	Batsman Name	Highest percentage of defense shots	Highest effectiveness average of defense shots	defense_score
8	Vijay Shankar	30.769231	0.891667	0.274359
0	Bhuvneshwar Kumar	25.000000	1.000000	0.250000
7	Shikhar Dhawan	23.577236	0.924138	0.217886
2	KL Rahul	21.982759	0.935294	0.205603
6	Rohit Sharma	20.877458	0.931884	0.194554
3	Kedhar Jadhav	20.192308	0.952381	0.192308
4	MS Dhoni	18.442623	0.782222	0.144262
9	Virat Kohli	16.239316	0.886842	0.144017
1	Hardik Pandya	8.904110	0.707692	0.063014
5	Rishabh Pant	2.597403	0.000000	0.000000

In [63]:
```
def spin_ball_score(df = matches_df):
    spin_ball_shots = ['D', 'Dr', 'C', 'P', 'Sw', 'RS', 'ODR', 'F', 'R', 'LG', 'SC']
    spin_ball = ['GY', 'LB']

    spin_ball_shots_count = df[df['Type of Shot'].isin(spin_ball_shots) & df['Ball type'].isin(spin_ball)].grou
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    spin_ball_score_df = spin_ball_shots_count.merge(grouped_df,on = 'Batsman Name')
    spin_ball_score_df = spin_ball_score_df.reset_index()

    spin_ball_score_df['Highest percentage of spin ball shots'] = (spin_ball_score_df['spin_ball_shots_count']/
    spin_ball_score_df['Highest effectiveness average of spin ball shots'] = spin_ball_score_df['sum_effectiven
    spin_ball_score_df['spin_ball_score'] = (spin_ball_score_df['Highest percentage of spin ball shots']/100) *

    spin_ball_score_df = spin_ball_score_df.sort_values(by=['spin_ball_score'], ascending=False)

    return spin_ball_score_df.loc[:,['Batsman Name','Highest percentage of spin ball shots','Highest effectiven
```

In [64]:
```
spin_ball_score()
```

Out[64]:

	Batsman Name	Highest percentage of spin ball shots	Highest effectiveness average of spin ball shots	spin_ball_score
4	Vijay Shankar	10.256410	0.762500	0.078205
1	Kedhar Jadhav	14.423077	0.520000	0.075000
2	MS Dhoni	4.508197	0.345455	0.015574
5	Virat Kohli	1.068376	0.840000	0.008974
0	Hardik Pandya	0.684932	1.000000	0.006849
3	Rohit Sharma	0.151286	0.400000	0.000605

In [65]:
```
def control_score(df = matches_df):
    shots = ['D', 'Dr', 'C', 'P', 'Sw', 'RS', 'ODR', 'F', 'R', 'LG', 'SC', 'L']

    shot_count_df = df[df['Control'] == 1.0].groupby('Batsman Name').agg(controlled_shots_count = ('Type of Sho
    grouped_df = df.groupby('Batsman Name').agg(total_no_of_balls_faced = ('Type of Shot', 'count'))

    control_score_df = shot_count_df.merge(grouped_df, on ='Batsman Name')
    control_score_df = control_score_df.reset_index()

    control_score_df['Highest percentage of controlled shots'] = (control_score_df['controlled_shots_count']/co
    control_score_df['Highest effectiveness average of controlled shots'] = control_score_df['sum_effectiveness
    control_score_df['control_score'] = (control_score_df['Highest percentage of controlled shots']/100) * cont

    control_score_df = control_score_df.sort_values(by=['control_score'], ascending=False)

    return control_score_df.loc[:,['Batsman Name','Highest percentage of controlled shots','Highest effectivene
```

In [66]:
```
control_score()
```

Out[66]:

	Batsman Name	Highest percentage of controlled shots	Highest effectiveness average of controlled shots	control_score
10	Rohit Sharma	94.553707	0.506560	0.478971
11	Shikhar Dhawan	86.991870	0.539252	0.469106
4	KL Rahul	90.086207	0.464115	0.418103
13	Virat Kohli	92.307692	0.445602	0.411325
3	Jasprit Bumrah	100.000000	0.400000	0.400000
2	Hardik Pandya	89.726027	0.438168	0.393151
7	MS Dhoni	81.967213	0.460500	0.377459
12	Vijay Shankar	71.794872	0.485714	0.348718
9	Rishabh Pant	90.909091	0.370000	0.336364
6	Kuldeep Yadav	100.000000	0.300000	0.300000
5	Kedhar Jadhav	65.384615	0.398529	0.260577