EDA with python on Cricket dataset

This is the final project submission for program 'Data analysis with Python : Zero to Pandas' provided by Jovian.ai.

Whole project is divided in following steps :

--> Dataset selection and loading it into Jupyter notebook.

--> Data cleaning with various measures

--> Exploratory analysis and visualization

--> Asking and answering questions about dataset

--> Inferences and conclusions after analysis

About Dataset

This dataset is chosen from kaggle with URL : https://www.kaggle.com/datasets/notkrishna/cricket-statistics-for-all-formats

The data is sourced from ESPNCricinfo which is leading and trusted site for sports domain.

The dataset contains info. about the the game of cricket across different countries, formats with stats of players.

It contains 9 csv files. Out of which, file named 'twbo.csv' is chosen for EDA.

In chosen file, some of the stats mentioned for bowlers in T20 games starting from 2006 to 2021.

```
project_name = "EDA with python on Cricket dataset"
```

```
!pip install jovian --upgrade -q
```

```
import jovian
```

```
jovian.commit(project=project_name)
```

```
[jovian] Updating notebook "hmehta1789/eda-with-python-on-cricket-dataset" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset
```

```
'https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset'
```

(1) Lets import the CSV file with Pandas

```
import pandas as pd
```

```
df= pd.read_csv('twbo.csv')
```

```
df
```

	Unnamed: 0	Player	Span	Mat	Inns	Overs	Mdns	Runs	Wkts	BBI	Ave	Econ	SR	4	5
0	0	Shakib Al Hasan (BAN)	2006-2021	94	93	347.3	3	2316	117	5/20	19.79	6.66	17.8	5	1
1	1	TG Southee (NZ)	2008-2021	92	90	332.5	2	2729	111	5/18	24.58	8.19	17.9	1	1
2	2	SL Malinga (SL)	2006-2020	84	83	299.5	1	2225	107	5/6	20.79	7.42	16.8	1	2
3	3	Rashid Khan (AFG/ICC)	2015-2021	56	56	211.2	1	1312	103	5/3	12.73	6.20	12.3	4	2
4	4	Shahid Afridi (ICC/PAK)	2006-2018	99	97	361.2	4	2396	98	4/11	24.44	6.63	22.1	3	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
176	176	LE Plunkett (ENG)	2006-2019	22	22	79.2	1	627	25	3/21	25.08	7.90	19.0	0	0
177	177	Harbhajan Singh (INDIA)	2006-2016	28	27	102.0	5	633	25	4/12	25.32	6.20	24.4	1	0
178	178	B Muzarabani (ZIM)	2018-2021	21	21	81.4	0	653	25	3/21	26.12	7.99	19.6	0	0
179	179	Mohammad Nawaz (PAK)	2016-2021	30	30	97.0	1	694	25	2/11	27.76	7.15	23.2	0	0
180	180	Washington Sundar (INDIA)	2017-2021	31	30	103.3	1	750	25	3/22	30.00	7.24	24.8	0	0

181 rows × 15 columns

This file contains rows/records for 181 players starting from index 0 to 180. It contains total 15 columns, out of which 1st contains index and other 14 contains data. The indexing is look like done on the basis of number of wickets.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181 entries, 0 to 180
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  181 non-null    int64
 1   Player      181 non-null    object
 2   Span        181 non-null    object
 3   Mat         181 non-null    int64
 4   Inns        181 non-null    int64
```

```
 5   Overs       181 non-null    float64
 6   Mdns        181 non-null    int64
 7   Runs        181 non-null    int64
 8   Wkts        181 non-null    int64
 9   BBI         181 non-null    object
 10  Ave         181 non-null    float64
 11  Econ        181 non-null    float64
 12  SR          181 non-null    float64
 13  4           181 non-null    int64
 14  5           181 non-null    int64
dtypes: float64(4), int64(8), object(3)
memory usage: 21.3+ KB
```

There are 8 columns which are of integer type, 4 columns are of floating number type and rest 3 are of object datatype.

(2) Data Cleaning

Lets load the list of columns which are there in file.

```
df.columns
```

```
Index(['Unnamed: 0', 'Player', 'Span', 'Mat', 'Inns', 'Overs', 'Mdns', 'Runs',
       'Wkts', 'BBI', 'Ave', 'Econ', 'SR', '4', '5'],
      dtype='object')
```

1) The title for first column here is 'unnamed : 0', so changing it from 'unnamed : 0' to 'Rank'. Also at last, 4 and 5 suggesting 4 wicket haul and 5 wicket haul respectively. The term BBI refers to Best Bowling Figures.

```
df.rename(columns = {'Unnamed: 0' : 'Rank', '4' : '4WH', '5' : '5WH', 'BBI': 'BBF'}, ir
```

```
df.head()
```

	Rank	Player	Span	Mat	Inns	Overs	Mdns	Runs	Wkts	BBF	Ave	Econ	SR	4WH	5WH
0	0	Shakib Al Hasan (BAN)	2006-2021	94	93	347.3	3	2316	117	5/20	19.79	6.66	17.8	5	1
1	1	TG Southee (NZ)	2008-2021	92	90	332.5	2	2729	111	5/18	24.58	8.19	17.9	1	1
2	2	SL Malinga (SL)	2006-2020	84	83	299.5	1	2225	107	5/6	20.79	7.42	16.8	1	2
3	3	Rashid Khan (AFG/ICC)	2015-2021	56	56	211.2	1	1312	103	5/3	12.73	6.20	12.3	4	2
4	4	Shahid Afridi (ICC/PAK)	2006-2018	99	97	361.2	4	2396	98	4/11	24.44	6.63	22.1	3	0

2) 'Player' column also include countries. Doing separation as new two columns will have only players name and countries .

```
df[['Players_1','Country']] = df['Player'].str.split('(', expand = True)
```

```
df['Country'] = df['Country'].str.strip(")")
df['Country']
```

```
0          BAN
1           NZ
2           SL
3      AFG/ICC
4      ICC/PAK
        ...
176        ENG
177      INDIA
178        ZIM
179        PAK
180      INDIA
Name: Country, Length: 181, dtype: object
```

```
df = df.drop(columns = 'Player')
```

```
df
```

	Rank	Span	Mat	Inns	Overs	Mdns	Runs	Wkts	BBF	Ave	Econ	SR	4WH	5WH	Players_1	Cou
0	0	2006-2021	94	93	347.3	3	2316	117	5/20	19.79	6.66	17.8	5	1	Shakib Al Hasan	B
1	1	2008-2021	92	90	332.5	2	2729	111	5/18	24.58	8.19	17.9	1	1	TG Southee	
2	2	2006-2020	84	83	299.5	1	2225	107	5/6	20.79	7.42	16.8	1	2	SL Malinga	
3	3	2015-2021	56	56	211.2	1	1312	103	5/3	12.73	6.20	12.3	4	2	Rashid Khan	AFG/
4	4	2006-2018	99	97	361.2	4	2396	98	4/11	24.44	6.63	22.1	3	0	Shahid Afridi	ICC/P
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
176	176	2006-2019	22	22	79.2	1	627	25	3/21	25.08	7.90	19.0	0	0	LE Plunkett	E
177	177	2006-2016	28	27	102.0	5	633	25	4/12	25.32	6.20	24.4	1	0	Harbhajan Singh	IN
178	178	2018-2021	21	21	81.4	0	653	25	3/21	26.12	7.99	19.6	0	0	B Muzarabani	Z
179	179	2016-2021	30	30	97.0	1	694	25	2/11	27.76	7.15	23.2	0	0	Mohammad Nawaz	P
180	180	2017-2021	31	30	103.3	1	750	25	3/22	30.00	7.24	24.8	0	0	Washington Sundar	IN

181 rows × 16 columns

3) Lets Convert 'Span' column into two new columns namely, 'Start_year' and 'last_record_year' then subtracting both for better analysis.

```
df['Span']
```

```
0      2006-2021
1      2008-2021
2      2006-2020
3      2015-2021
4      2006-2018
         ...
176    2006-2019
177    2006-2016
178    2018-2021
179    2016-2021
180    2017-2021
Name: Span, Length: 181, dtype: object
```

```
df[['start_year','last_recorded_year']] = df['Span'].str.split('-', expand = True)
```

```
df[['start_year','last_recorded_year']]
```

	start_year	last_recorded_year
0	2006	2021
1	2008	2021
2	2006	2020
3	2015	2021
4	2006	2018
...	...	...
176	2006	2019
177	2006	2016
178	2018	2021
179	2016	2021
180	2017	2021

181 rows × 2 columns

```
df['last_recorded_year'] =  df['last_recorded_year'].astype(int)
df[ 'start_year'] = df[ 'start_year'].astype(int)
```

```
df['total_years'] = df['last_recorded_year'] - df[ 'start_year']
```

```
df['total_years']
```

```
0      15
1      13
2      14
3       6
4      12
       ..
176    13
177    10
178     3
179     5
180     4
Name: total_years, Length: 181, dtype: int64
```

```
print (df.head(10))

print()

print(df.info())
```

```
   Rank       Span  Mat  Inns  Overs  Mdns  Runs  Wkts   BBF    Ave  Econ  \
0     0  2006-2021   94    93  347.3     3  2316   117  5/20  19.79  6.66
1     1  2008-2021   92    90  332.5     2  2729   111  5/18  24.58  8.19
2     2  2006-2020   84    83  299.5     1  2225   107   5/6  20.79  7.42
3     3  2015-2021   56    56  211.2     1  1312   103   5/3  12.73  6.20
4     4  2006-2018   99    97  361.2     4  2396    98  4/11  24.44  6.63
5     5  2015-2021   61    61  219.4     6  1678    86  5/22  19.51  7.63
6     6  2007-2016   60    60  200.3     2  1443    85   5/6  16.97  7.19
7     7  2009-2015   64    63  238.2     2  1516    85  4/19  17.83  6.36
8     8  2014-2021   66    64  226.3     0  1824    83  4/28  21.97  8.05
9     9  2014-2021   71    70  246.5     2  2114    79   4/6  26.75  8.56

     SR  4WH  5WH           Players_1  Country  start_year  \
0  17.8    5    1    Shakib Al Hasan       BAN        2006
1  17.9    1    1         TG Southee        NZ        2008
2  16.8    1    2         SL Malinga        SL        2006
3  12.3    4    2        Rashid Khan   AFG/ICC        2015
4  22.1    3    0      Shahid Afridi   ICC/PAK        2006
5  15.3    3    1  Mustafizur Rahman       BAN        2015
6  14.1    4    2           Umar Gul       PAK        2007
7  16.8    4    0        Saeed Ajmal       PAK        2009
8  16.3    2    0           IS Sodhi        NZ        2014
9  18.7    2    0          CJ Jordan       ENG        2014

   last_recorded_year  total_years
0                2021           15
1                2021           13
```

```
2                2020           14
3                2021            6
4                2018           12
5                2021            6
6                2016            9
7                2015            6
8                2021            7
9                2021            7
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181 entries, 0 to 180
Data columns (total 19 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Rank                181 non-null    int64  
 1   Span                181 non-null    object 
 2   Mat                 181 non-null    int64  
 3   Inns                181 non-null    int64  
 4   Overs               181 non-null    float64
 5   Mdns                181 non-null    int64  
 6   Runs                181 non-null    int64  
 7   Wkts                181 non-null    int64  
 8   BBF                 181 non-null    object 
 9   Ave                 181 non-null    float64
 10  Econ                181 non-null    float64
 11  SR                  181 non-null    float64
 12  4WH                 181 non-null    int64  
 13  5WH                 181 non-null    int64  
 14  Players_1           181 non-null    object 
 15  Country             181 non-null    object 
 16  start_year          181 non-null    int64  
 17  last_recorded_year  181 non-null    int64  
 18  total_years         181 non-null    int64  
dtypes: float64(4), int64(11), object(4)
memory usage: 27.0+ KB
None
```

Now data is ready for analysis and visualization.

(3) Exploratory Analysis and Visualization

After cleaning the data the next step would be to do some analysis and visualization with different types of graphs.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "hmehta1789/eda-with-python-on-cricket-dataset" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset
```

'https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset'

Lets look at the data with numeric view.

```
df.describe()
```

	Rank	Mat	Inns	Overs	Mdns	Runs	Wkts	Ave	
count	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	18
mean	90.000000	41.745856	38.281768	128.826519	1.524862	950.154696	42.955801	22.409613	
std	52.394338	21.473020	16.525507	56.972751	1.529593	421.642105	18.355388	5.105992	
min	0.000000	13.000000	13.000000	45.000000	0.000000	236.000000	25.000000	9.440000	
25%	45.000000	26.000000	26.000000	88.300000	0.000000	658.000000	29.000000	19.040000	
50%	90.000000	35.000000	33.000000	111.000000	1.000000	854.000000	37.000000	22.160000	
75%	135.000000	51.000000	48.000000	155.000000	2.000000	1175.000000	51.000000	25.360000	
max	180.000000	124.000000	97.000000	361.200000	8.000000	2729.000000	117.000000	36.350000	

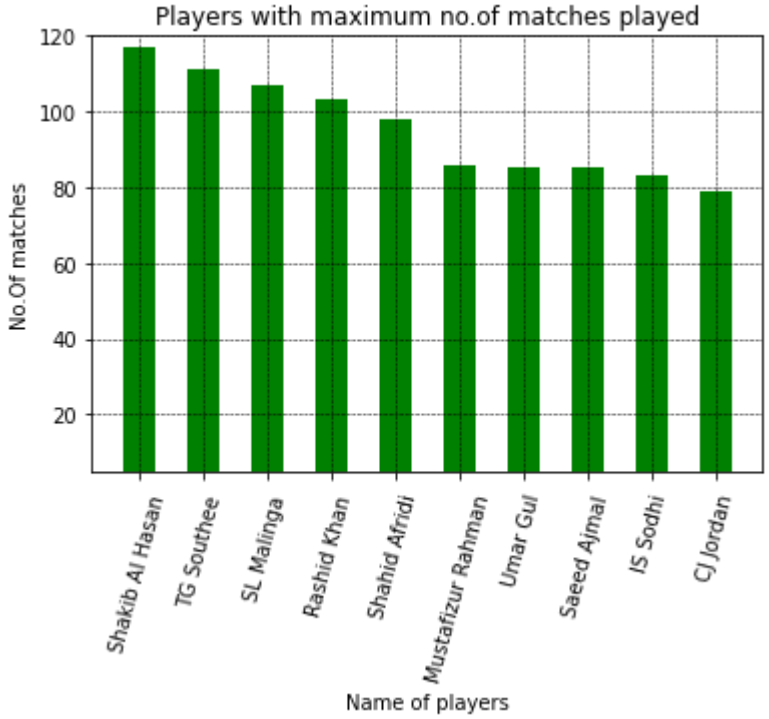
Let's begin by importing`matplotlib.pyplot` and `seaborn`.

```
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

1. Lets see the players which played highest matches

```
plt.bar(df.Players_1.head(10),df.Wkts.head(10),align = 'center', color = 'Green', width
plt.xticks(rotation = 75)
plt.grid(color = 'Black',linestyle = '--', linewidth = 0.5)
plt.title('Players with maximum no.of matches played')
plt.xlabel('Name of players')
plt.ylabel('No.Of matches')
plt.ylim(5,120)
```

```
(5.0, 120.0)
```



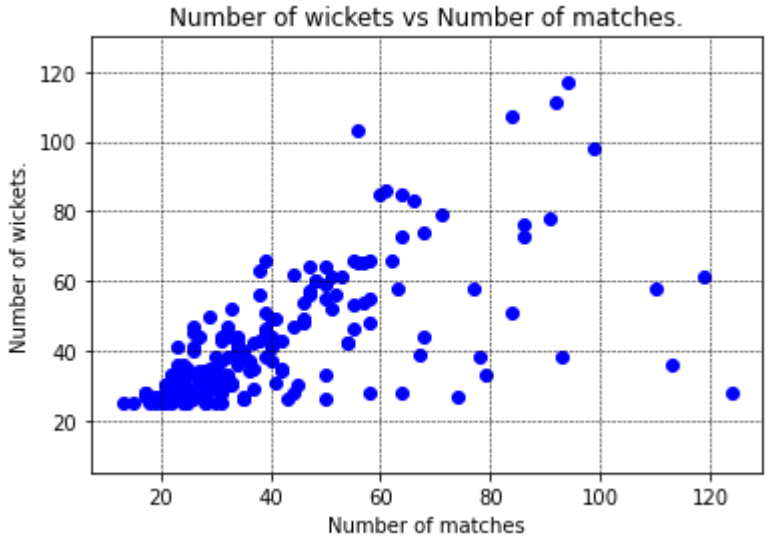
---- > Shakib Al Hasan tops the list of 'Number of matches played' with close to 120 matches, while TG Southee and SL Malinga comes 2nd and 3rd respectively.

----- > Mustafizur Rahman, Umar Gul and Saeed Ajmal seems to have almost equal experience of above 80 matches.

2. Lets see the relation between number of matches and number of wickets.

```
plt.scatter(df.Mat,df.Wkts, color = 'BLue')
plt.grid(color = 'Black',linestyle = '--', linewidth = 0.5)
plt.title('Number of wickets vs Number of matches.')
plt.xlabel('Number of matches')
plt.ylabel('Number of wickets.')
plt.ylim(5,130)
```

```
(5.0, 130.0)
```



---- >The highly dense region shows that - players in starting of their career gets wicket evenly as they play matches. (The region around 20 to 40 on both axis is almost linear)

-——→ However as the number of matches increases this relation is not linear now. Maybe due to the form or returning after injury are the reason behind that.

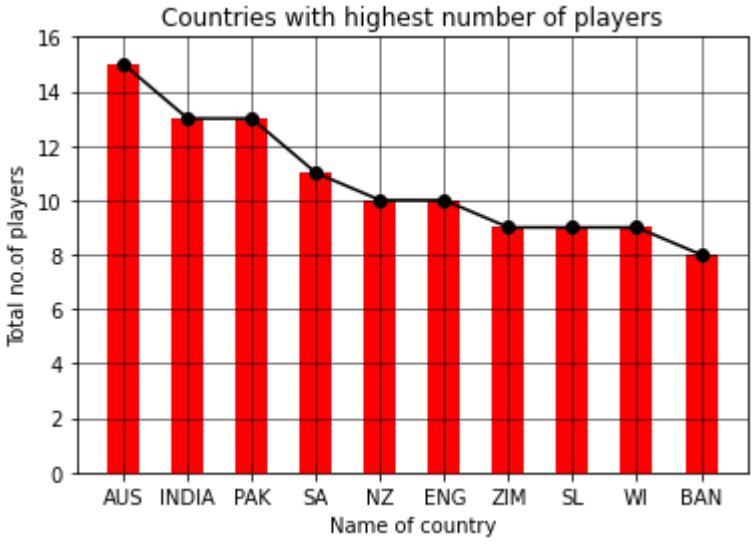
3. Lets see the number of players for country wise data.

```
Country_wise_data = df['Country'].value_counts().head(10)
```

```
plt.plot(Country_wise_data.index,Country_wise_data, color = 'BLack', marker = 'o')

plt.bar(Country_wise_data.index,Country_wise_data,align = 'center', width = 0.5, color
plt.grid(color = 'Black', linewidth = 0.5)
plt.title('Countries with highest number of players')
plt.xlabel('Name of country')
plt.ylabel('Total no.of players')
plt.ylim(0,16)
```

```
(0.0, 16.0)
```



-——→ The Bar-chart along with line suggest that the, Australia is the country from which,highest number of players(15) played T20s.

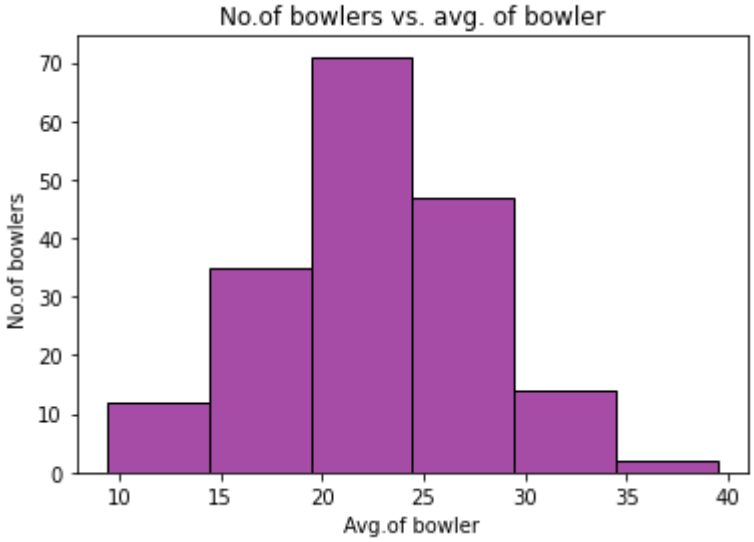
-——→ India and Pak(13); Nz and Eng(10); Zim, Sl and Wi(9) are the pairs of countries which have similar no.of players played T20s.

4. Lets see the distribution of 'average' for all players.

```
Low_avg = df['Ave']
```

```
sns.histplot(data = Low_avg,color = 'purple',binwidth = 5, alpha = 0.7)
plt.xlabel('Avg.of bowler')
plt.ylabel('No.of bowlers ')
plt.title('No.of bowlers vs. avg. of bowler')
```

```
Text(0.5, 1.0, 'No.of bowlers vs. avg. of bowler')
```



-——→ In cricket, a player's bowling average is the number of runs they have conceded per wicket taken. The lower the bowling average is, the better the bowler is performing.

-——→ This Histogram shows the distribution of no.of bolwers for bowling avg. It shows that the average of nearly 70 bowlers lies within 20 to 25. nearly half of this lies within 15 and 20.

——→ It also suggest that few bowlers have average less than 10, and one or maybe two bowlers have average greater than 35.

5. Lets see the how many players are playing till 2021.

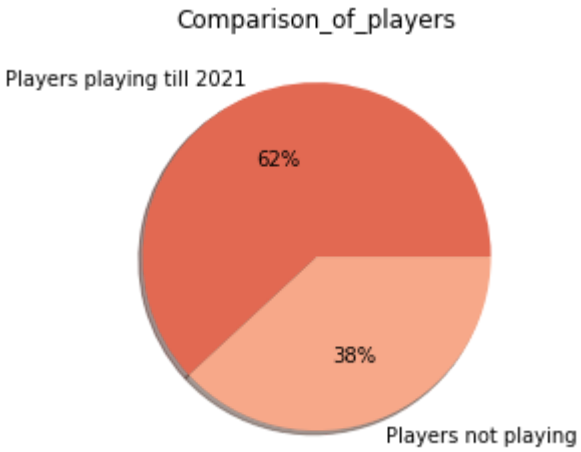
```
currently_playing = (df['last_recorded_year'] == 2021).value_counts()
```

```
currently_playing
```

```
True     112
False     69
Name: last_recorded_year, dtype: int64
```

```
labels = ['Players playing till 2021','Players not playing']
colors = sns.color_palette('coolwarm_r')
plt.pie(currently_playing, labels =labels,autopct = '%0.0f%%', colors = colors, shadow=
plt.title('Comparison_of_players')
```

```
Text(0.5, 1.0, 'Comparison_of_players')
```



Let us save and upload our work to Jovian before continuing

-——→ The Piechart shows the comparison between the players who are still playing as of recorded date on 2021 and players which are not playing.

--——→ The 62% players are still playing in 2021, among all T20 players played, means majority of total is playing as T20 format is newly introduced as compared to Test matches and One-day Matches.

-——→ The 38%maybe shows the combination of players who are not played due to injury or bad form and players who are retired from T20 Matches. Focusing on particular format of play such as Test or ODI ,can also be the reason for not playing T20s.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "hmehta1789/eda-with-python-on-cricket-dataset" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/hmehta1789/eda-with-python-on-
cricket-dataset
```

```
'https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset'
```

(4) Asking and Answering Questions

Lets ask and answer some question about data.

Q1: Which are the bowlers who have taken a five-wicket haul more than one time?

```
more_5WH = df[df['5WH'] > 1]
```

```
more_5WH[['Players_1','Country', '5WH']]
```

	Players_1	Country	5WH

	Players_1	Country	5WH
2	SL Malinga	SL	2
3	Rashid Khan	AFG/ICC	2
6	Umar Gul	PAK	2
15	BAW Mendis	SL	2
23	Imran Tahir	SA/World	2
68	AC Agar	AUS	2
79	DM Nakrani	UGA	2

--- > Five-wicket haul is great achievement for bowlers, these are the bowlers whom done it for more than one time.

Making a new variable 'More_5WH' and assigning it to a dataframe, in which the value of column '5WH' is more than one is filtered out.

Printing name of player, Country and filtered column '5WH' in final result.

Q2: Which five bowlers have lowest economy amongst bolwers who have played 50 or more than 50 matches?

```
Economic_bolwers = df[df['Mat']>= 50].head(5).sort_values(by ='Econ' )
```

```
Economic_bolwers[['Players_1','Country', 'Wkts', 'Econ']]
```

	Players_1	Country	Wkts	Econ
3	Rashid Khan	AFG/ICC	103	6.20
4	Shahid Afridi	ICC/PAK	98	6.63
0	Shakib Al Hasan	BAN	117	6.66
2	SL Malinga	SL	107	7.42
1	TG Southee	NZ	111	8.19

--- > In cricket, a bowler's economy rate is the average number of runs they have conceded per over bowled. Considering 50+ matches of experience and lowest economy, these are the go-to bowlers for any captain.

Making a new variable 'Economic_bowlers' and assigning it to a dataframe, in which the value of column 'Mat' is kept more than or equal to fifty, with sorting it in ascending order according to economy. Using 'head' for first five results.

Printing name of player,country filtered column 'Mat' and sorted 'Econ'.

Q3: List down top five bowlers with lowest Strike-rate.

```
Wicket_taking_bowlers =df[[ 'Players_1', 'Country', 'SR' ]].head().sort_values(by = 'SR
```

```
Wicket_taking_bowlers
```

	Players_1	Country	SR
3	Rashid Khan	AFG/ICC	12.3
2	SL Malinga	SL	16.8
0	Shakib Al Hasan	BAN	17.8
1	TG Southee	NZ	17.9
4	Shahid Afridi	ICC/PAK	22.1

--- > Here strike-rate refers to the total no.of balls a bowler take, to get a wicket. Lowest strike-rate refers to the ability of bowlers to take wickets frequently.

Making a new variable 'Wicket_taking_bowlers' with assigning it to a list of columns of 'players_1' and 'SR', as sorting it with 'SR' in ascending order and using head to print first five results.

printing Players name,country and filtered 'SR'.

Q4: List down three bowlers, which are having less than five years of experience and more than 50 wickets.

```
Good_bowlers = df[(df.total_years < 5) & (df.Wkts > 50) ].head(3)
```

```
Good_bowlers[['Players_1','Country', 'total_years', 'Wkts']]
```

	Players_1	Country	total_years	Wkts
13	Shadab Khan	PAK	4	73
34	T Shamsi	SA	4	57
43	PWH de Silva	SL	2	52

--- > Only few bowlers can reach the milestone of taking 50 wickets in t20s with less than 5 years of experience.These are the star-performers for their team.

Making a new variable 'Good_bowlers' and assigning it to a dataframe of two conditions required with using AND(&) operator for 'total_years' and 'Wkts'. Using head to print out three results.

printing Players name, country with filtered 'total_years' and 'Wkts'.

Q5: Describe the BBF of the bolwers for more than 5 maiden overs delivered.

```
Good_BBF = df[df['Mdns'] > 5].sort_values(ascending = False, by= 'Mdns')
```

```
Good_BBF[['Players_1','Country', 'Mdns', 'BBF']]
```

	Players_1	Country	Mdns	BBF
16	JJ Bumrah	INDIA	8	3/11
5	Mustafizur Rahman	BAN	6	5/22
18	KMDN Kulasekara	SL	6	4/31

--- > The maiden over is one, when bowler didn't conceded single run in entire over. These bowlers are always captain's favorite when runs are leaking at other end.

Making a new variable 'Good_BBF' and assigning it to a dataframe where 'Mdns' is more than five and sorting it in descending order to get high maiden overs.

printing players name,country BBFs and sorted Maidens.

Let us save and upload our work to Jovian before continuing.

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "hmehta1789/eda-with-python-on-cricket-dataset" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset
```

```
'https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset'
```

(5) Inferences and Conclusion

——→ Players like SL Malinga, Rashid Khan appears all three times while answering the questions of first three questions. They can be considered as legendry bowlers as far as International T20s are concerned.(Ref- Q.1,Q.2,Q.3)

--- > Bowlers like Shakib-al-Hasan and Shaheed Afridi has good experience, good no. of wickets as well as good strike rate.(Ref- Q.2,Q.3)

--- > J J Boomrah has 8 maiden overs, which is highest for any bowler till date, shows his great ability of bowling.(Ref - Q.5)

---- > Shadab Khan has highest wickets among players who has less than 50 matches experience. As considering gap of wickets of 1st and 2nd bowler in this list, shows his good performance.(Ref - Q.4)

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Updating notebook "hmehta1789/eda-with-python-on-cricket-dataset" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset
```

```
'https://jovian.ai/hmehta1789/eda-with-python-on-cricket-dataset'
```

References and Future Work

As more data is added in this dataset in future, variety of questions can be asked and with the help of visualization more insights can be gathered.

> Submission Instructions (delete this cell)
>
> - Upload your notebook to your Jovian.ml profile using `jovian.commit`.
> - **Make a submission here**: https://jovian.ml/learn/data-analysis-with-python-zero-to-pandas/assignment/course-project
> - Share your work on the forum: https://jovian.ml/forum/t/course-project-on-exploratory-data-analysis-discuss-and-share-your-work/11684
> - Share your work on social media (Twitter, LinkedIn, Telegram etc.) and tag @JovianML
>
> (Optional) Write a blog post
>
> - A blog post is a great way to present and showcase your work.
> - Sign up on Medium.com to write a blog post for your project.
> - Copy over the explanations from your Jupyter notebook into your blog post, and embed code cells & outputs
> - Check out the Jovian.ml Medium publication for inspiration: https://medium.com/jovianml

```
import jovian
```

```
jovian.commit()
```

```
[jovian] Attempting to save notebook..
[jovian] Updating notebook "aakashns/zerotopandas-course-project-starter" on https://jovian.ml/
[jovian] Uploading notebook..
[jovian] Capturing environment..
[jovian] Committed successfully! https://jovian.ml/aakashns/zerotopandas-course-project-starter
```

```
'https://jovian.ml/aakashns/zerotopandas-course-project-starter'
```