Basic steps to start an EDA

- Reading / importing / download the dataset into Python environment
- Size of data - w.r.t. numer of rows and columns, or disk space
- Columns / fields - important to know what the dataset is about
- Datatypes - what are different datatypes in the columns
- Unique values - how many differnt values are there in each columns
- Value counts - what are the counts of each element in a particular column
- Missing values - check and handle
- Check the statistical properties
- Correlation - within columns
- Visualization - scatter plots, bar charts, histograms, pie charts
- Deriving basic conclusions

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

Reading / importing / download the dataset into Python environment

In [2]:
```
df_iris = pd.read_csv("./iris_csv (1).csv")
```

Size of data - w.r.t. numer of rows and columns, or disk space

In [3]:
```
df_iris.shape
```

Out[3]:
```
(150, 5)
```

In [4]:
```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  150 non-null    float64
 1   sepalwidth   130 non-null    float64
 2   petallength  150 non-null    float64
 3   petalwidth   150 non-null    float64
 4   class        150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
df_iris
```

Out[5]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

Columns / fields

In [6]:
```
df_iris.columns
```

Out[6]:
```
Index(['sepallength', 'sepalwidth', 'petallength', 'petalwidth', 'class'], dtype
='object')
```

Datatypes

In [7]:
```
df_iris.dtypes
```

Out[7]:
```
sepallength    float64
sepalwidth     float64
petallength    float64
petalwidth     float64
class           object
dtype: object
```

Number of unique values

In [8]:
```
dfs_nunique = df_iris.nunique()
dfs_nunique
```

Out[8]:
```
sepallength    35
sepalwidth     19
petallength    43
petalwidth     22
class           3
dtype: int64
```

In [9]:
```
type(df_iris.nunique())
```

Out[9]:
```
pandas.core.series.Series
```

In [10]:
```
dfs_nunique[dfs_nunique > 30].index
```

Out[10]:
```
Index(['sepallength', 'petallength'], dtype='object')
```

In [11]:
```
df_iris[['sepallength', 'petallength']].describe()
```

Out[11]:

	sepallength	petallength
count	150.000000	150.000000
mean	5.843333	3.758667
std	0.828066	1.764420
min	4.300000	1.000000
25%	5.100000	1.600000
50%	5.800000	4.350000
75%	6.400000	5.100000
max	7.900000	6.900000

In [12]:
```
df_iris.describe()
```

Out[12]:

	sepallength	sepalwidth	petallength	petalwidth
count	150.000000	130.000000	150.000000	150.000000
mean	5.843333	2.987692	3.758667	1.198667
std	0.828066	0.396304	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.725000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.200000	5.100000	1.800000
max	7.900000	3.900000	6.900000	2.500000

In [13]:
```
df_nunique = pd.DataFrame(df_iris.nunique(), columns = ["nunique"])
df_nunique
```

Out[13]:

	nunique
sepallength	35
sepalwidth	19
petallength	43
petalwidth	22
class	3

In [14]:
```
# Filter rows having numbers greater than 30
# Preparing the condition / mask : numbers are greater than 30
df_nunique["nunique"] > 30
```

Out[14]:
```
sepallength     True
sepalwidth     False
petallength     True
petalwidth     False
class          False
Name: nunique, dtype: bool
```

In [15]:
```
# Use the True / False mask to filter the rows
df_nunique[df_nunique["nunique"] > 30]
```

Out[15]:

	nunique
sepallength	35
petallength	43

In [16]:
```
df_nunique[df_nunique["nunique"] > 30].index
```

Out[16]:
```
Index(['sepallength', 'petallength'], dtype='object')
```

In [17]:
```
df_iris["sepallength"].nunique()
```

Out[17]:
```
35
```

Unique values

In [18]:
```
df_iris["sepallength"].unique()
```

Out[18]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [19]:
```
len(df_iris["sepallength"].unique())
```

Out[19]:
```
35
```

Value counts

In [20]:
```
df_iris["sepallength"].value_counts()
```

Out[20]:
```
sepallength
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
4.5     1
7.9     1
Name: count, dtype: int64
```

Missing values - checking

In [21]:
```
df_iris.isna()
```

Out[21]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [22]:
```
df_iris.isna().sum()
```

Out[22]:
```
sepallength     0
sepalwidth     20
petallength     0
petalwidth      0
class           0
dtype: int64
```

Missing values - handling

In [23]:
```
# Dropping the rows where missing values is present
df_iris.dropna()
```

Out[23]:

	sepallength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

130 rows × 5 columns

Check the statistical properties

In [24]:
```
import numpy as np
np.array([3, 2.5]).mean()
```

Out[24]:
```
2.75
```

In [25]:
```
df_iris.describe()
```

Out[25]:

	sepallength	sepalwidth	petallength	petalwidth
count	150.000000	130.000000	150.000000	150.000000
mean	5.843333	2.987692	3.758667	1.198667
std	0.828066	0.396304	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.725000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.200000	5.100000	1.800000
max	7.900000	3.900000	6.900000	2.500000

In [26]:
```
# Check how many non-numeric entries are there in sepallength
for i in df_iris["sepallength"]:
#     print(i)
    print(type(i))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

```
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

In [27]:

```
df_iris.iloc[18]
```

Out[27]:

```
sepallength            5.7
sepalwidth             3.8
petallength            1.7
petalwidth             0.3
class          Iris-setosa
Name: 18, dtype: object
```

In [28]:

```
# Drop those rows having non-numeric values in sepallength column
df_iris = df_iris.drop(index = [2, 18])
df_iris.describe()
```

Out[28]:

	sepallength	sepalwidth	petallength	petalwidth
count	148.000000	128.000000	148.000000	148.000000
mean	5.852027	2.979687	3.789189	1.211486
std	0.828199	0.392320	1.756316	0.760194
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.700000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.200000	5.100000	1.800000
max	7.900000	3.900000	6.900000	2.500000

In [29]:
```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 148 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  148 non-null    float64
 1   sepalwidth   128 non-null    float64
 2   petallength  148 non-null    float64
 3   petalwidth   148 non-null    float64
 4   class        148 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.9+ KB
```

In [30]:
```
# Converting the datatype of sepallength column
df_iris["sepallength"] = df_iris["sepallength"].astype(float)
```

In [31]:
```
df_iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 148 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepallength  148 non-null    float64
 1   sepalwidth   128 non-null    float64
 2   petallength  148 non-null    float64
 3   petalwidth   148 non-null    float64
 4   class        148 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.9+ KB
```

In [32]:
```
df_iris.describe()
```

Out[32]:

	sepallength	sepalwidth	petallength	petalwidth
count	148.000000	128.000000	148.000000	148.000000
mean	5.852027	2.979687	3.789189	1.211486
std	0.828199	0.392320	1.756316	0.760194
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.700000	1.600000	0.300000
50%	5.800000	3.000000	4.400000	1.300000
75%	6.400000	3.200000	5.100000	1.800000
max	7.900000	3.900000	6.900000	2.500000

Correlations

In [33]:
```
# Slicing the last column, as correlation function cannot work on string columns
df_iris.iloc[:, :-1]
```

Out[33]:

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

148 rows × 4 columns

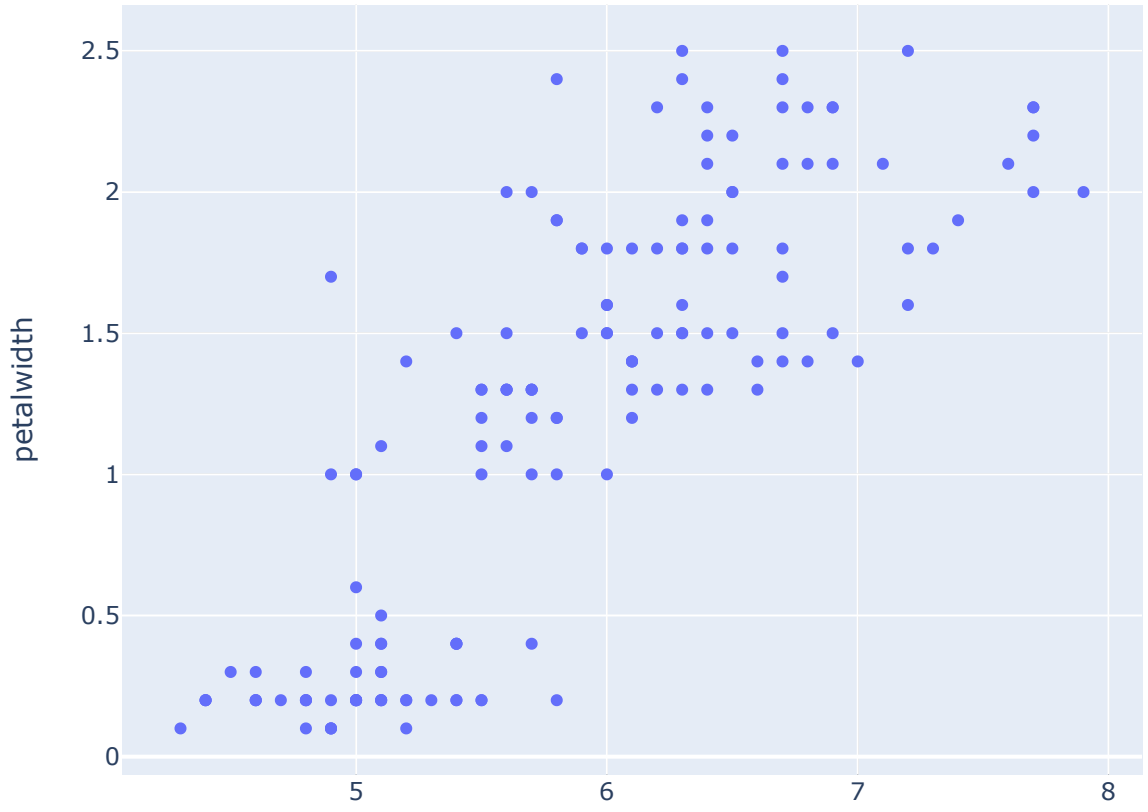
In [34]:
```
df_iris.iloc[:, :-1].corr() # Correlation values ie between -1 to +1
```

Out[34]:

	sepallength	sepalwidth	petallength	petalwidth
sepallength	1.000000	-0.015810	0.872839	0.818277
sepalwidth	-0.015810	1.000000	-0.278102	-0.202211
petallength	0.872839	-0.278102	1.000000	0.961954
petalwidth	0.818277	-0.202211	0.961954	1.000000

Visualization

In [35]:
```
import plotly.express as px
px.scatter(data_frame = df_iris, x = "sepallength", y = "petalwidth")
```



In [36]:
```
df_marks = pd.DataFrame({"Marks": [70, 75, 55, 80, 83, 40, 41, 95, 98], "Sleep hour
df_marks
```

Out[36]:

	Marks	Sleep hours
0	70	5
1	75	5
2	55	7
3	80	8
4	83	7
5	40	5
6	41	6
7	95	7
8	98	6

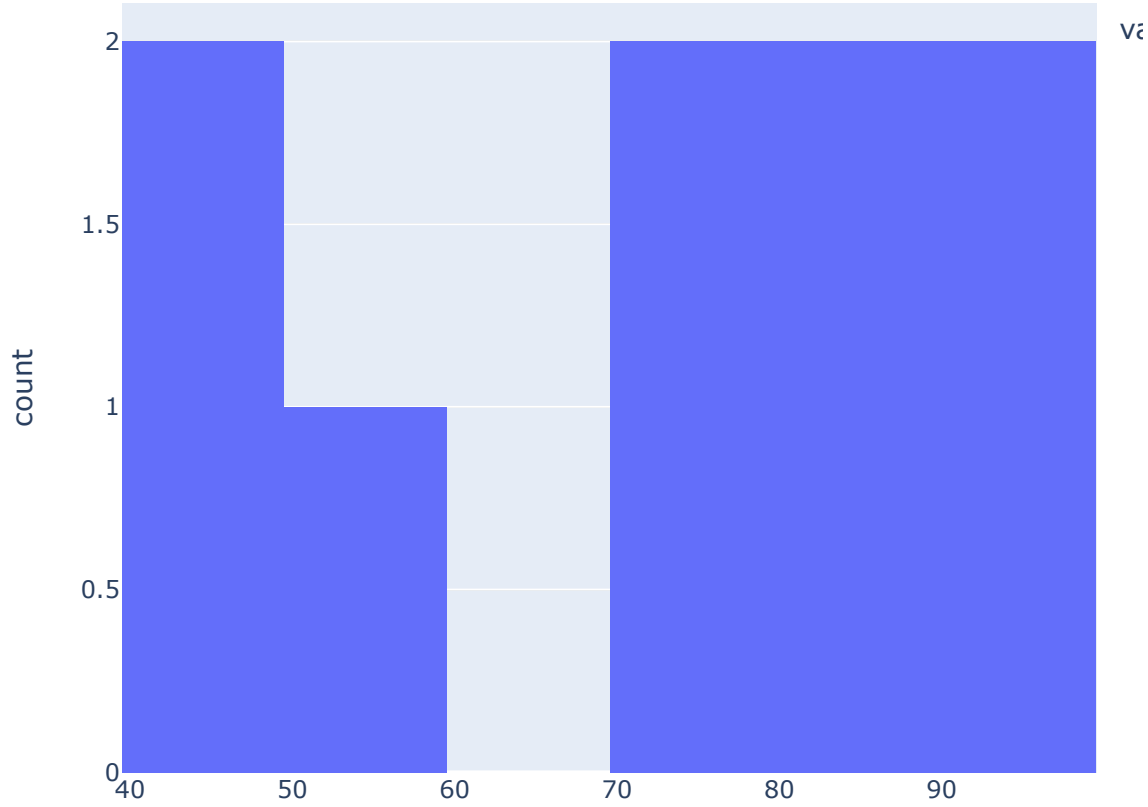
Univariate analysis

- Histogram

In [37]:
```
df_marks["Marks"]
```

Out[37]:
```
0    70
1    75
2    55
3    80
4    83
5    40
6    41
7    95
8    98
Name: Marks, dtype: int64
```

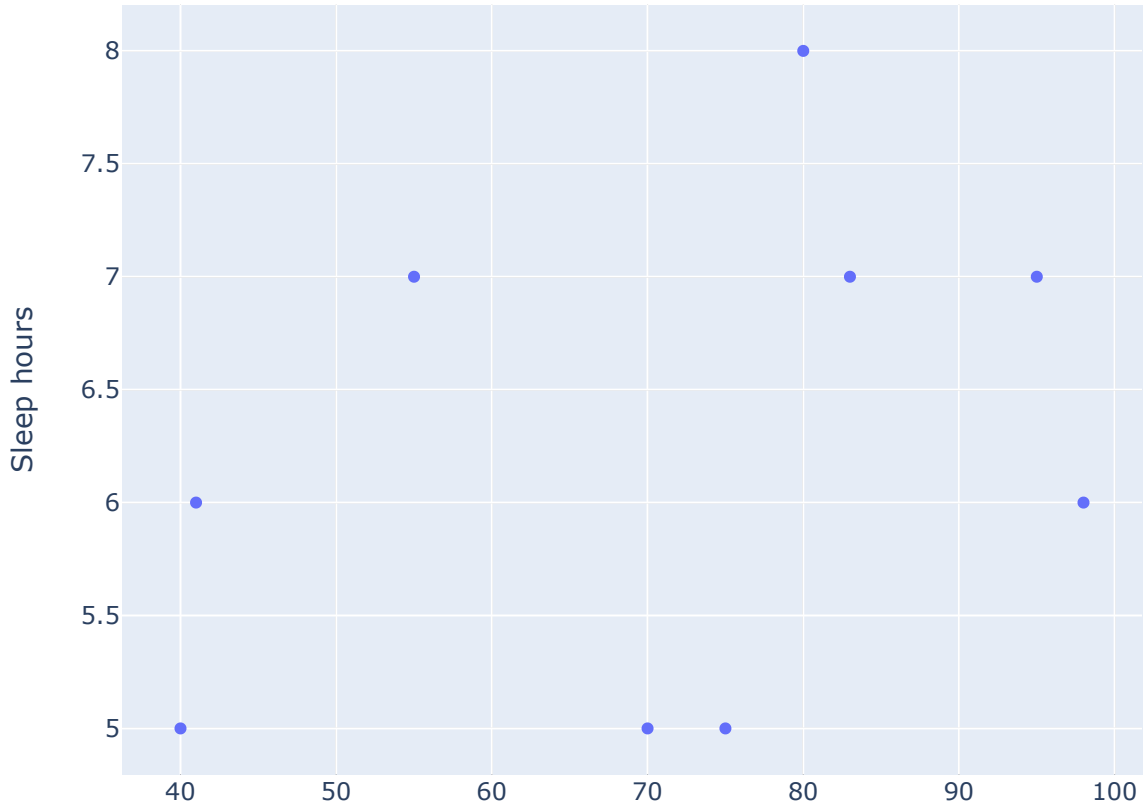
In [38]:
```
px.histogram(df_marks["Marks"], nbins = 10)
```



Bi-variate analysis

- Scatter plot

In [39]:
```
px.scatter(df_marks, x = "Marks", y = "Sleep hours")
```



In [40]:
```
df_marks.corr()
```

Out[40]:

	Marks	Sleep hours
Marks	1.000000	0.355123
Sleep hours	0.355123	1.000000