Understanding dataset handling using Pandas

In [60]:

```
# import CSV data

import pandas as pd

df = pd.read_csv("./dataset_1.csv")

df.head()
```

Out[60]:

	x	y	label
0	1.1	39343	0
1	1.3	46205	0
2	1.5	37731	0
3	2.0	43525	0
4	2.2	39891	0

In [61]:

```
# Count values

df['x'].value_counts()
```

Out[61]:

```
4.0     2
3.2     2
11.1    1
10.9    1
2.0     1
3.0     1
4.5     1
9.0     1
9.5     1
10.5    1
11.5    1
7.9     1
10.7    1
3.7     1
4.1     1
7.1     1
10.3    1
8.2     1
8.7     1
11.7    1
2.2     1
2.9     1
1.3     1
11.3    1
3.9     1
1.1     1
9.6     1
1.5     1
Name: x, dtype: int64
```

In [62]:

```
# Find number of unique values
df.nunique()
```

Out[62]:

```
x        28
y        27
```

```
label     2
dtype: int64
```

In [63]:
```
# Statistics of the data

df.describe()
```

Out[63]:

	x	y	label
count	30.00000	30.000000	30.000000
mean	6.42000	83168.200000	0.500000
std	3.72405	31965.248116	0.508548
min	1.10000	37731.000000	0.000000
25%	3.20000	56720.750000	0.000000
50%	5.80000	81359.000000	0.500000
75%	10.12500	113517.750000	1.000000
max	11.70000	122391.000000	1.000000

In [64]:
```
# Dataset memory information

df.memory_usage()
```

Out[64]:
```
Index     128
x         240
y         240
label     240
dtype: int64
```

In [108…
```
# Find data type of all columns

df.dtypes
```

Out[108…
```
x          float64
y            int64
label     category
dtype: object
```

In [110…
```
# Change OR alter data type of a column

df['label']= df.label.astype('int64')
df.dtypes
```

Out[110…
```
x         float64
y           int64
label       int64
dtype: object
```

In [111…
```
df['label']= df.label.astype('category')
df.dtypes
```

Out[111…
```
x          float64
y            int64
label     category
dtype: object
```

In [66]:
```
# Split OR partition a dataframe

# Get first five rows and all the columns
df.loc[0:4]
```

Out[66]:

	x	y	label
0	1.1	39343	0
1	1.3	46205	0
2	1.5	37731	0
3	2.0	43525	0
4	2.2	39891	0

In [67]:
```

# Get first five rows and some columns
display(df.loc[0:4, ['x', 'label']])
```

	x	label
0	1.1	0
1	1.3	0
2	1.5	0
3	2.0	0
4	2.2	0

In [68]:
```

# Condition implementations

display(df.loc[(df.label == 0)])
```

	x	y	label
0	1.1	39343	0
1	1.3	46205	0
2	1.5	37731	0
3	2.0	43525	0
4	2.2	39891	0
5	2.9	56642	0
6	3.0	60150	0
7	3.2	54445	0
8	3.2	64445	0
9	3.7	57189	0
10	3.9	63218	0
11	4.0	55794	0
12	4.0	56957	0
13	4.1	57081	0

	x	y	label
14	4.5	61111	0

In [69]:
```
# selecting rows from 1 to 4 and columns from 2 to 2
display(df.iloc[1: 5, 2: 3])
```

	label
1	0
2	0
3	0
4	0

In [70]:
```
# selecting 0th, 2th index rows
display(df.iloc[[0, 2]])
```

	x	y	label
0	1.1	39343	0
2	1.5	37731	0

In [71]:
```
# Remove duplicates

df.drop_duplicates(inplace=False)
```

Out[71]:

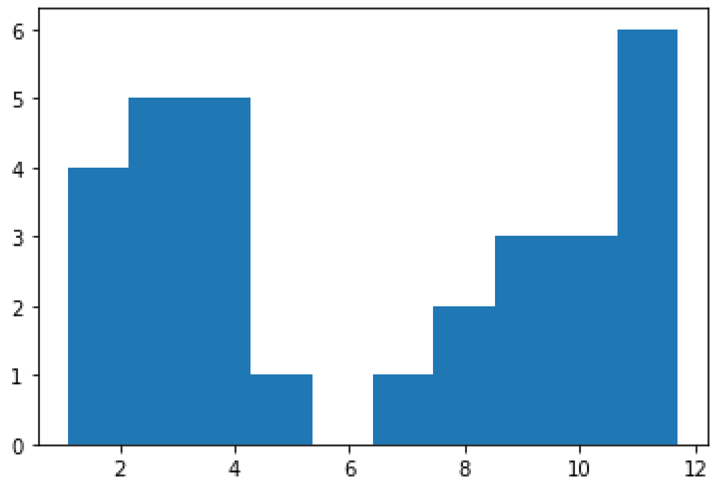
	x	y	label
0	1.1	39343	0
1	1.3	46205	0
2	1.5	37731	0
3	2.0	43525	0
4	2.2	39891	0
5	2.9	56642	0
6	3.0	60150	0
7	3.2	54445	0
8	3.2	64445	0
9	3.7	57189	0
10	3.9	63218	0
11	4.0	55794	0
12	4.0	56957	0
13	4.1	57081	0
14	4.5	61111	0
15	7.1	98273	1

	x	y	label
16	7.9	101302	1
17	8.2	113812	1
18	8.7	109431	1
19	9.0	105582	1
20	9.5	116969	1
21	9.6	112635	1
22	10.3	122391	1
23	10.5	121872	1
24	10.7	121772	1
25	10.9	121872	1
26	11.1	105582	1
27	11.3	105982	1
28	11.5	121872	1
29	11.7	121972	1

Histogram representations of the dataset

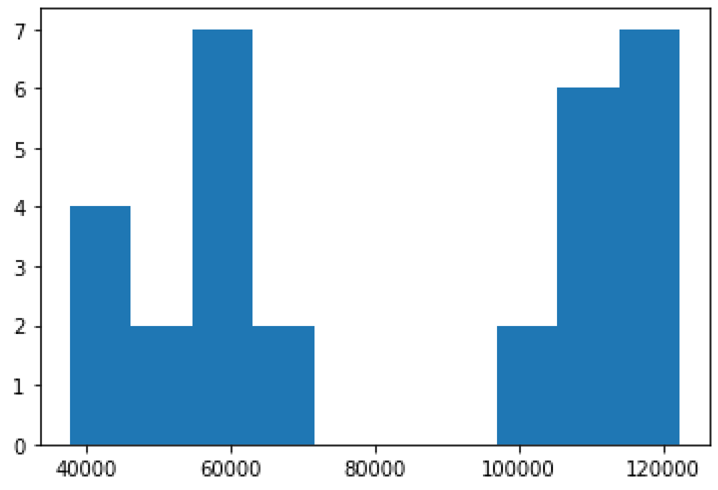
In [77]:

```
plt.hist(df.x)
plt.show()
```

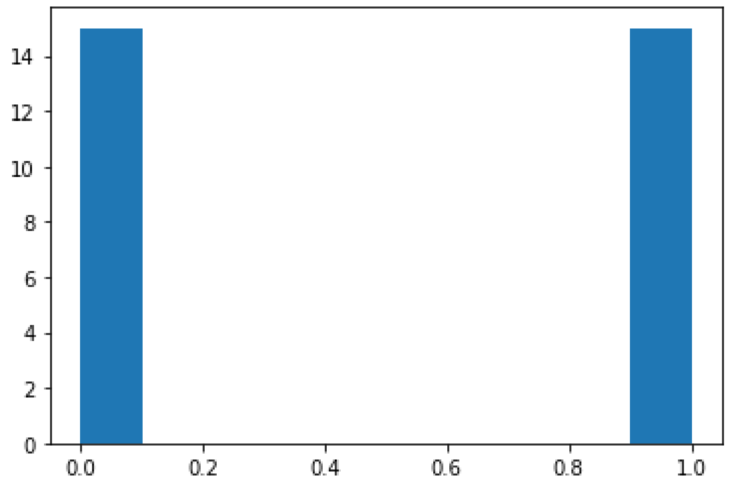


In [78]:

```
plt.hist(df.y)
plt.show()
```



```
In [79]:
plt.hist(df.label)
plt.show()
```



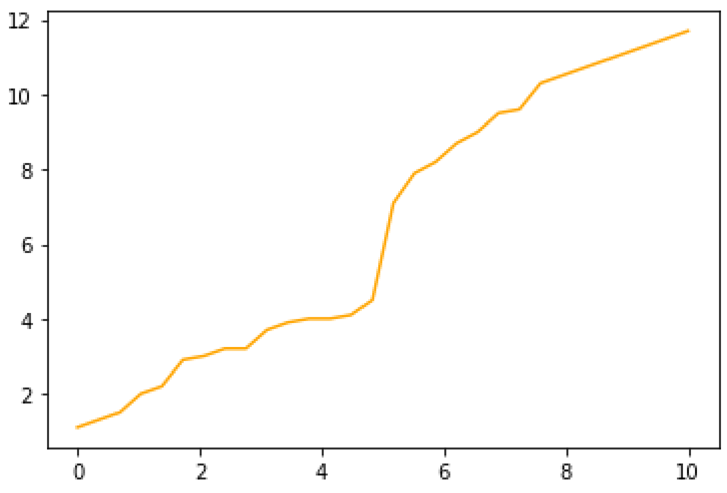
Drawing line charts

```
In [98]:
import numpy as np
x1 = np.linspace(0, 10, 30)
print(x1)
```

```
[ 0.          0.34482759  0.68965517  1.03448276  1.37931034  1.72413793
  2.06896552  2.4137931   2.75862069  3.10344828  3.44827586  3.79310345
  4.13793103  4.48275862  4.82758621  5.17241379  5.51724138  5.86206897
  6.20689655  6.55172414  6.89655172  7.24137931  7.5862069   7.93103448
  8.27586207  8.62068966  8.96551724  9.31034483  9.65517241 10.        ]
```

```
In [102…
plt.plot(x1, df.x.to_numpy(), '-',color='orange')
```

```
Out[102…  [<matplotlib.lines.Line2D at 0x1db3102f850>]
```

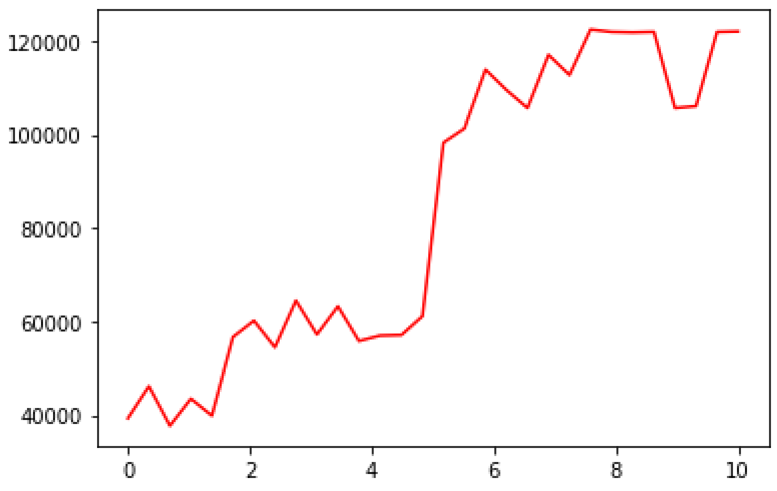


In [103...

```
plt.plot(x1, df.y.to_numpy(), '-',color='red')
```

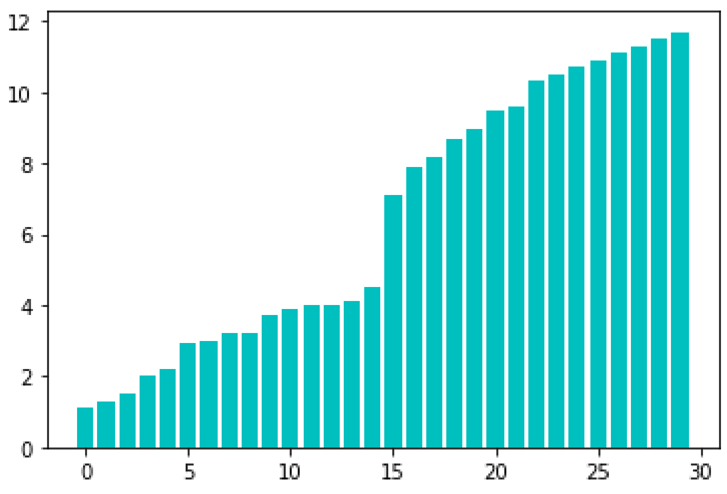
Out[103...

```
[<matplotlib.lines.Line2D at 0x1db30d03af0>]
```



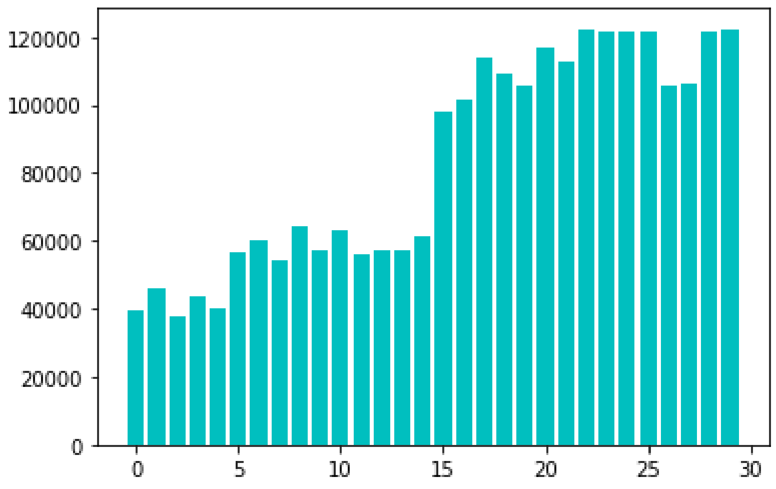
In [104...

```
plt.bar(range(len(df.x)), df.x,color='c')
plt.show()
```



In [105...

```
plt.bar(range(len(df.y)), df.y,color='c')
plt.show()
```



Scatter plots

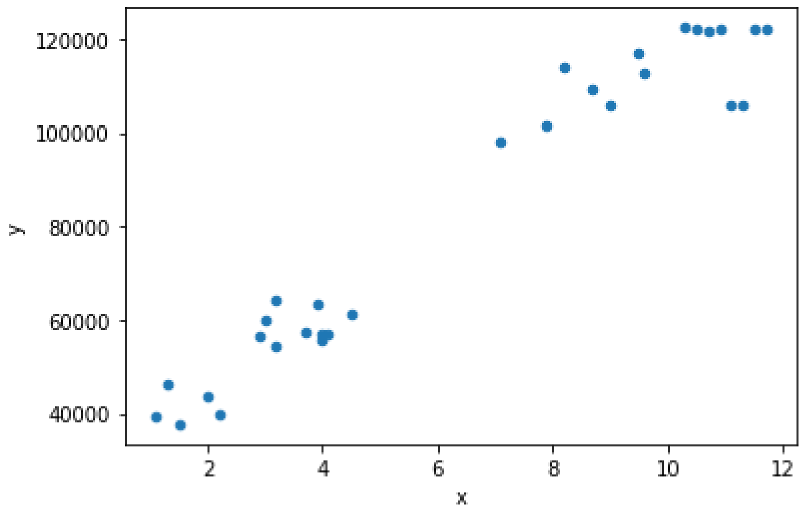
In [106…

```
import matplotlib.pyplot as plt

df.plot.scatter(x='x', y='y')
```

Out[106…

```
<AxesSubplot:xlabel='x', ylabel='y'>
```



Solve the following

In [ ]:

```
# Colour the scatter plot based on the different classes
```

Creating a dataframe object

In [ ]:

```
data = pd.DataFrame({'Brand': ['Maruti', 'Hyundai', 'Tata',
                               'Mahindra', 'Maruti', 'Hyundai',
                               'Renault', 'Tata', 'Maruti'],
                     'Year': [2012, 2014, 2011, 2015, 2012, 2016, 2014, 2018, 2019],
                     'Kms Driven': [50000, 30000, 60000, 25000, 10000, 46000, 31000,
                     'City': ['Gurgaon', 'Delhi', 'Mumbai',
                              'Delhi', 'Mumbai', 'Delhi',
                              'Mumbai', 'Chennai',  'Ghaziabad'],
                     'Mileage':  [28, 27, 25, 26, 28, 29, 24, 21, 24]})
```

```
# displaying the DataFrame
display(data)
```

Storing and retreiving nested data

In [114…

```
# Understanding JSON

import json

# some JSON:
x =  '{ "name":"John", "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["age"])
```

```
30
```

In [115…

```
# some JSON:
x =  '{ "name":{"First":"John", "Surname":"Cena"}, "age":30, "city":"New York"}'

# parse x:
y = json.loads(x)

# the result is a Python dictionary:
print(y["name"]["First"])
```

```
John
```