1. Import all the required Python Libraries

In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:

```

```

1. Load the Dataset into pandas data frame

In [12]:

```
data = {
    "Roll_No" : [1,2,3],
    "Marks" : [50,40,45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

df
```

Out[12]:

	Roll_No	Marks
0	**1**	**50**
1	**2**	**40**
2	**3**	**45**

In [13]:

```
print(df)
```

```
   Roll_No  Marks
0        1     50
1        2     40
2        3     45
```

In [24]:

```
print(df.to_string)
```

```
<bound method DataFrame.to_string of      Roll_No  Marks
ABC         1     50
PQR         2     40
XYZ         3     45>
```

In [23]:

```
print(df.to_string())
```

```
     Roll_No  Marks
ABC         1     50
PQR         2     40
XYZ         3     45
```

In [18]:

```
print(df.loc[2])
```

```
Roll_No     3
Marks      45
Name: 2, dtype: int64
```

In [19]:

```
print(df.loc[[0,2]])
```

```
   Roll_No  Marks
0        1     50
2        3     45
```

In [20]:

```
#load data into a DataFrame object:
df = pd.DataFrame(data,index = ["ABC","PQR","XYZ"])
print(df)
```

```
     Roll_No  Marks
ABC        1     50
PQR        2     40
XYZ        3     45
```

In [21]:

```
print(df.loc["XYZ"])
```

```
Roll_No     3
Marks      45
Name: XYZ, dtype: int64
```

In [ ]:

```

```

1. Data Preprocessing

In [25]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df
```

Out[25]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3

	Duration	Date	Pulse	Maxpulse	Calories
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [26]:

```
df.head()
```

Out[26]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0

In [27]:

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

Out[27]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0

In [28]:

```
df.tail()
```

Out[28]:

	Duration	Date	Pulse	Maxpulse	Calories
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [29]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  32 non-null     int64  
 1   Date      31 non-null     object 
 2   Pulse     32 non-null     int64  
 3   Maxpulse  32 non-null     int64  
 4   Calories  30 non-null     float64
dtypes: float64(1), int64(3), object(1)
memory usage: 1.4+ KB
```

In [30]:

```
new_df = df.dropna()
print(new_df.to_string())
```

```
    Duration           Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [31]:

```
df
```

Out[31]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [33]:

```
df.dropna(inplace = True)
print(df.to_string())
```

```
    Duration           Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
```

```
10         60   '2020/12/11'     103        147      329.3
11         60   '2020/12/12'     100        120      250.7
12         60   '2020/12/12'     100        120      250.7
13         60   '2020/12/13'     106        128      345.3
14         60   '2020/12/14'     104        132      379.3
15         60   '2020/12/15'      98        123      275.0
16         60   '2020/12/16'      98        120      215.2
17         60   '2020/12/17'     100        120      300.0
19         60   '2020/12/19'     103        123      323.0
20         45   '2020/12/20'      97        125      243.0
21         60   '2020/12/21'     108        131      364.2
23         60   '2020/12/23'     130        101      300.0
24         45   '2020/12/24'     105        132      246.0
25         60   '2020/12/25'     102        126      334.5
26         60       20201226     100        120      250.0
27         60   '2020/12/27'      92        118      241.0
29         60   '2020/12/29'     100        132      280.0
30         60   '2020/12/30'     102        129      380.3
31         60   '2020/12/31'      92        115      243.0
```

In [34]:

```
df
```

Out[34]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
29	60	'2020/12/29'	100	132	280.0

	Duration	Date	Pulse	Maxpulse	Calories
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [36]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df.fillna(130, inplace = True)
df
```

Out[36]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	130.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	130	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	130.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [37]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df["Calories"].fillna(130, inplace = True)
```

```
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112     130.0
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132     130.0
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [38]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
x = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True)
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130    409.10
1         60  '2020/12/02'    117       145    479.00
2         60  '2020/12/03'    103       135    340.00
3         45  '2020/12/04'    109       175    282.40
4         45  '2020/12/05'    117       148    406.00
5         60  '2020/12/06'    102       127    300.00
6         60  '2020/12/07'    110       136    374.00
7        450  '2020/12/08'    104       134    253.30
8         30  '2020/12/09'    109       133    195.10
9         60  '2020/12/10'     98       124    269.00
10        60  '2020/12/11'    103       147    329.30
11        60  '2020/12/12'    100       120    250.70
12        60  '2020/12/12'    100       120    250.70
13        60  '2020/12/13'    106       128    345.30
14        60  '2020/12/14'    104       132    379.30
15        60  '2020/12/15'     98       123    275.00
16        60  '2020/12/16'     98       120    215.20
17        60  '2020/12/17'    100       120    300.00
18        45  '2020/12/18'     90       112    304.68
19        60  '2020/12/19'    103       123    323.00
20        45  '2020/12/20'     97       125    243.00
21        60  '2020/12/21'    108       131    364.20
22        45           NaN    100       119    282.00
23        60  '2020/12/23'    130       101    300.00
24        45  '2020/12/24'    105       132    246.00
25        60  '2020/12/25'    102       126    334.50
26        60      20201226    100       120    250.00
```

```
27         60   '2020/12/27'      92        118     241.00
28         60   '2020/12/28'     103        132     304.68
29         60   '2020/12/29'     100        132     280.00
30         60   '2020/12/30'     102        129     380.30
31         60   '2020/12/31'      92        115     243.00
```

In [39]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
x = df["Calories"].median()
df["Calories"].fillna(x, inplace = True)
df
```

Out[39]:

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	291.2
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	291.2
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

In [43]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df["Date"] = pd.to_datetime(df["Date"])
print(df)
```

```
    Duration       Date  Pulse  Maxpulse  Calories
0         60 2020-12-01    110       130     409.1
1         60 2020-12-02    117       145     479.0
2         60 2020-12-03    103       135     340.0
3         45 2020-12-04    109       175     282.4
4         45 2020-12-05    117       148     406.0
5         60 2020-12-06    102       127     300.0
6         60 2020-12-07    110       136     374.0
7        450 2020-12-08    104       134     253.3
8         30 2020-12-09    109       133     195.1
9         60 2020-12-10     98       124     269.0
10        60 2020-12-11    103       147     329.3
11        60 2020-12-12    100       120     250.7
12        60 2020-12-12    100       120     250.7
13        60 2020-12-13    106       128     345.3
14        60 2020-12-14    104       132     379.3
15        60 2020-12-15     98       123     275.0
16        60 2020-12-16     98       120     215.2
17        60 2020-12-17    100       120     300.0
18        45 2020-12-18     90       112       NaN
19        60 2020-12-19    103       123     323.0
20        45 2020-12-20     97       125     243.0
21        60 2020-12-21    108       131     364.2
22        45        NaT    100       119     282.0
23        60 2020-12-23    130       101     300.0
24        45 2020-12-24    105       132     246.0
25        60 2020-12-25    102       126     334.5
26        60 2020-12-26    100       120     250.0
27        60 2020-12-27     92       118     241.0
28        60 2020-12-28    103       132       NaN
29        60 2020-12-29    100       132     280.0
30        60 2020-12-30    102       129     380.3
31        60 2020-12-31     92       115     243.0
```

In [46]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df.dropna(subset = ["Date"] , inplace = True)
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
```

```
26         60      20201226    100        120      250.0
27         60  '2020/12/27'     92        118      241.0
28         60  '2020/12/28'    103        132        NaN
29         60  '2020/12/29'    100        132      280.0
30         60  '2020/12/30'    102        129      380.3
31         60  '2020/12/31'     92        115      243.0
```

In [49]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df.loc[7, "Duration"] = 45
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7         45  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132       NaN
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [50]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
for x in df.index :
    if df.loc[x, "Duration"] > 120 :
        df.loc[x, "Duration"] = 120
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        120  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
```

```
14         60  '2020/12/14'    104        132      379.3
15         60  '2020/12/15'     98        123      275.0
16         60  '2020/12/16'     98        120      215.2
17         60  '2020/12/17'    100        120      300.0
18         45  '2020/12/18'     90        112        NaN
19         60  '2020/12/19'    103        123      323.0
20         45  '2020/12/20'     97        125      243.0
21         60  '2020/12/21'    108        131      364.2
22         45           NaN    100        119      282.0
23         60  '2020/12/23'    130        101      300.0
24         45  '2020/12/24'    105        132      246.0
25         60  '2020/12/25'    102        126      334.5
26         60      20201226    100        120      250.0
27         60  '2020/12/27'     92        118      241.0
28         60  '2020/12/28'    103        132        NaN
29         60  '2020/12/29'    100        132      280.0
30         60  '2020/12/30'    102        129      380.3
31         60  '2020/12/31'     92        115      243.0
```

In [51]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
for x in df.index :
    if df.loc[x, "Duration"] > 120 :
        df.drop(x, inplace = True)
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
12        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132       NaN
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [52]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
print(df.duplicated())
```

```
0     False
1     False
2     False
3     False
4     False
```

```
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13    False
14    False
15    False
16    False
17    False
18    False
19    False
20    False
21    False
22    False
23    False
24    False
25    False
26    False
27    False
28    False
29    False
30    False
31    False
dtype: bool
```

In [53]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\dirtydata.csv"
df = pd.read_csv(path)
df.drop_duplicates(inplace = True)
print(df)
```

```
    Duration          Date  Pulse  Maxpulse  Calories
0         60  '2020/12/01'    110       130     409.1
1         60  '2020/12/02'    117       145     479.0
2         60  '2020/12/03'    103       135     340.0
3         45  '2020/12/04'    109       175     282.4
4         45  '2020/12/05'    117       148     406.0
5         60  '2020/12/06'    102       127     300.0
6         60  '2020/12/07'    110       136     374.0
7        450  '2020/12/08'    104       134     253.3
8         30  '2020/12/09'    109       133     195.1
9         60  '2020/12/10'     98       124     269.0
10        60  '2020/12/11'    103       147     329.3
11        60  '2020/12/12'    100       120     250.7
13        60  '2020/12/13'    106       128     345.3
14        60  '2020/12/14'    104       132     379.3
15        60  '2020/12/15'     98       123     275.0
16        60  '2020/12/16'     98       120     215.2
17        60  '2020/12/17'    100       120     300.0
18        45  '2020/12/18'     90       112       NaN
19        60  '2020/12/19'    103       123     323.0
20        45  '2020/12/20'     97       125     243.0
21        60  '2020/12/21'    108       131     364.2
22        45           NaN    100       119     282.0
23        60  '2020/12/23'    130       101     300.0
24        45  '2020/12/24'    105       132     246.0
25        60  '2020/12/25'    102       126     334.5
26        60      20201226    100       120     250.0
27        60  '2020/12/27'     92       118     241.0
28        60  '2020/12/28'    103       132       NaN
29        60  '2020/12/29'    100       132     280.0
30        60  '2020/12/30'    102       129     380.3
31        60  '2020/12/31'     92       115     243.0
```

In [10]:

```
data = pd.Series({'1st':1, '2nd':2, '3rd':3, '4th':4})
```

```
print(data,'\n')
print('Size = ',data.size)
```

```
1st    1
2nd    2
3rd    3
4th    4
dtype: int64

Size =  4
```

In [11]:

```
df = pd.DataFrame({'1st':[1,2], '2nd':[3,4], '3rd':[5,6], '4th':[7,8]})
print(df,'\n')
print('Size = ',df.size)
```

```
   1st  2nd  3rd  4th
0    1    3    5    7
1    2    4    6    8

Size =  8
```

In [12]:

```
df = pd.DataFrame({'1st':[1,2], '2nd':[3,4], '3rd':[5,6], '4th':[7,8]})
print(df,'\n')
print('Size = ',df.size)
print('Dimension = ',df.ndim)
print('Shape = ',df.shape)
```

```
   1st  2nd  3rd  4th
0    1    3    5    7
1    2    4    6    8

Size =  8
Dimension =  2
Shape =  (2, 4)
```

In [ ]:

```

```

1. Data Formatting and Data Normalization

In [13]:

```
df = pd.DataFrame({'Name':['Rohit', 'Raj', 'Shubh', 'Shivam'], 'Marks':[95,74,84,26], 'S
ubject':['Maths', 'Science', 'English', 'Social Science']})
column_names=df.columns
print(column_names)
```

```
Index(['Name', 'Marks', 'Subject'], dtype='object')
```

In [14]:

```
data = {'Name':['Rohit', 'Raj', 'Shubh', 'Shivam'], 'Marks':[95,74,84,26]}
df = pd.DataFrame(data)
column_names=df.columns
print(column_names)
```

```
Index(['Name', 'Marks'], dtype='object')
```

In [16]:

```
df = pd.DataFrame({'A':[21, 11, 19, None, 1],
                   'B':[7, 19, 57, 3, None],
                   'C':[10, 16, 11, 3, 8],
                   'D':[14 , 3, None, 2, 6]})
index_row = ['Row_1', 'Row_2', 'Row_3', 'Row_4', 'Row_5']
```

```
df.index = index_row
print(df)
print(df.columns)
```

```
          A     B   C     D
Row_1  21.0   7.0  10  14.0
Row_2  11.0  19.0  16   3.0
Row_3  19.0  57.0  11   NaN
Row_4   NaN   3.0   3   2.0
Row_5   1.0   NaN   8   6.0
Index(['A', 'B', 'C', 'D'], dtype='object')
```

In [17]:

```
dict = {'Phone':['Samsung S20', 'iPhone 11', 'Reliance Jio'], 'Price':[1000, 1100, 100]}
df = pd.DataFrame(dict)
print('The DataType of DataFrame is: ')
print(df.dtypes)
```

```
The DataType of DataFrame is:
Phone    object
Price     int64
dtype: object
```

In [18]:

```
dict = {'Phone':['Samsung S20', 'iPhone 11', 'Reliance Jio'], 'Price':[1000, 1100, 100],
'Discount':[np.nan, np.nan, np.nan]}
df = pd.DataFrame(dict)
print('The DataType of DataFrame is: ')
print(df.dtypes)
```

```
The DataType of DataFrame is:
Phone        object
Price         int64
Discount    float64
dtype: object
```

In [19]:

```
dict = {'Phone':['Samsung S20', 'iPhone 11', 'Reliance Jio'], 'Price':[1000, 1100, 100],
'Discount':[np.nan, np.nan, np.nan], 'ArrivalDate':[pd.Timestamp('20180310'), pd.Timesta
mp('20190310'), pd.Timestamp('20140310')]}
df = pd.DataFrame(dict)
print('The DataType of DataFrame is: ')
print(df.dtypes)
```

```
The DataType of DataFrame is:
Phone                   object
Price                    int64
Discount               float64
ArrivalDate    datetime64[ns]
dtype: object
```

In [21]:

```
dict = {'Phone':['Samsung S20', 'iPhone 11', 'Reliance Jio'], 'Price':[1000, 1100, 100],
'Discount':[np.nan, np.nan, np.nan], 'ArrivalDate':[pd.Timestamp('20180310'), pd.Timesta
mp('20190310'), pd.Timestamp('20140310')]}
df = pd.DataFrame(dict)
print('The Info of DataFrame is: ')
print(df.info())
```

```
The Info of DataFrame is:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Phone        3 non-null      object
 1   Price        3 non-null      int64
 2   Discount     0 non-null      float64
```

```
 3   ArrivalDate  3 non-null      datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(1), object(1)
memory usage: 224.0+ bytes
None
```

In [25]:

```
dataset = {'Name':['Rohit', 'Raj', 'Shubh', 'Shivam', 'Arun'],
           'Roll_No':['01', '02', '03', '04', np.nan],
           'Maths':['93', '63', np.nan, '94', '83'],
           'Science':['88', np.nan, '66', '94', np.nan],
           'English':['93', '74', '84', '92', '87']}
df = pd.DataFrame(dataset)

print('DataFrame: \n\n',df)

print('\nCount: \n')
df2 = df.count()
print(df2)
```

```
DataFrame:

      Name Roll_No Maths Science English
0   Rohit      01    93      88      93
1     Raj      02    63     NaN      74
2   Shubh      03   NaN      66      84
3  Shivam      04    94      94      92
4    Arun     NaN    83     NaN      87

Count:

Name       5
Roll_No    4
Maths      4
Science    3
English    5
dtype: int64
```

In [26]:

```
dataset = {'Name':['Rohit', 'Raj', 'Shubh', 'Shivam', 'Arun'],
           'Roll_No':['01', '02', '03', '04', np.nan],
           'Maths':['93', '63', np.nan, '94', '83'],
           'Science':['88', np.nan, '66', '94', np.nan],
           'English':['93', '74', '84', '92', '87']}
df = pd.DataFrame(dataset)

print('DataFrame: \n\n',df)

print('\nCount: \n')
df2 = df.count(axis='columns')
print(df2)
```

```
DataFrame:

      Name Roll_No Maths Science English
0   Rohit      01    93      88      93
1     Raj      02    63     NaN      74
2   Shubh      03   NaN      66      84
3  Shivam      04    94      94      92
4    Arun     NaN    83     NaN      87

Count:

0    5
1    4
2    4
3    5
4    3
dtype: int64
```

In [27]:

```
dataset = {'Name':['Rohit', 'Raj', 'Shubh', 'Shivam', 'Arun'],
           'Roll_No':['01', '02', '03', '04', np.nan],
           'Maths':['93', '63', np.nan, '94', '83'],
           'Science':['88', np.nan, '66', '94', np.nan],
           'English':['93', '74', '84', '92', '87']}
df = pd.DataFrame(dataset)

print('DataFrame: \n\n',df)

print('\nCount: \n')
df2 = df.set_index(['Maths', 'English']).count(level='Maths')
print(df2)
```

```
DataFrame:

      Name Roll_No Maths Science English
0    Rohit      01    93      88      93
1      Raj      02    63     NaN      74
2    Shubh      03   NaN      66      84
3   Shivam      04    94      94      92
4     Arun     NaN    83     NaN      87

Count:

        Name  Roll_No  Science
Maths
63         1        1        0
83         1        0        0
93         1        1        1
94         1        1        1
```

In [5]:

```
df = pd.DataFrame([
    [180000, 110, 18.9, 1400],
    [360000, 905, 23.4, 1800],
    [230000, 230, 14.0, 1300],
    [60000, 450, 13.5, 1500]],

    columns=['Col A', 'Col B', 'Col C', 'Col D'])

display(df)
```

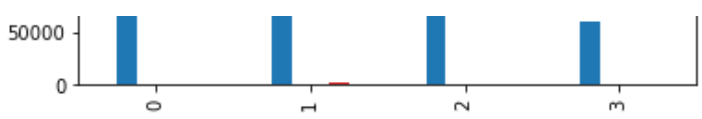
	Col A	Col B	Col C	Col D
0	**180000**	**110**	**18.9**	**1400**
1	**360000**	**905**	**23.4**	**1800**
2	**230000**	**230**	**14.0**	**1300**
3	**60000**	**450**	**13.5**	**1500**

In [31]:

```
df.plot(kind = 'bar')
```

Out[31]:

<AxesSubplot:>



In [32]:

```
df_max_scaled = df.copy()

for column in df_max_scaled.columns:
    df_max_scaled[column] = df_max_scaled[column]/df_max_scaled[column].abs().max()

display(df_max_scaled)
```

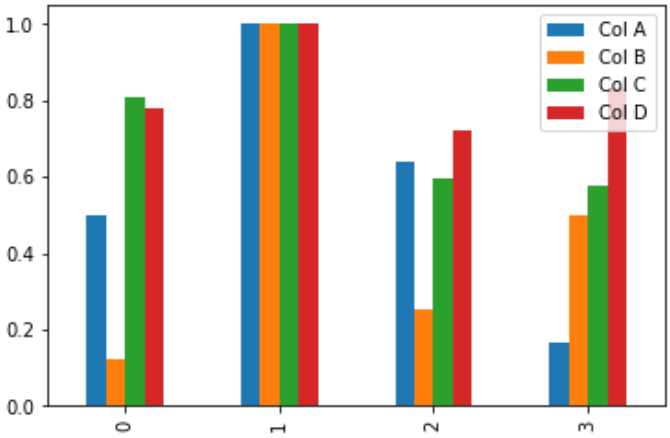
	Col A	Col B	Col C	Col D
0	**0.500000**	**0.121547**	**0.807692**	**0.777778**
1	**1.000000**	**1.000000**	**1.000000**	**1.000000**
2	**0.638889**	**0.254144**	**0.598291**	**0.722222**
3	**0.166667**	**0.497238**	**0.576923**	**0.833333**

In [33]:

```
df_max_scaled.plot(kind = 'bar')
```

Out[33]:

<AxesSubplot:>



In [6]:

```
df_min_max_scaled = df.copy()

for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column]-df_min_max_scaled[column].min
())/(df_min_max_scaled[column].max()-df_min_max_scaled[column].min())

display(df_min_max_scaled)
```

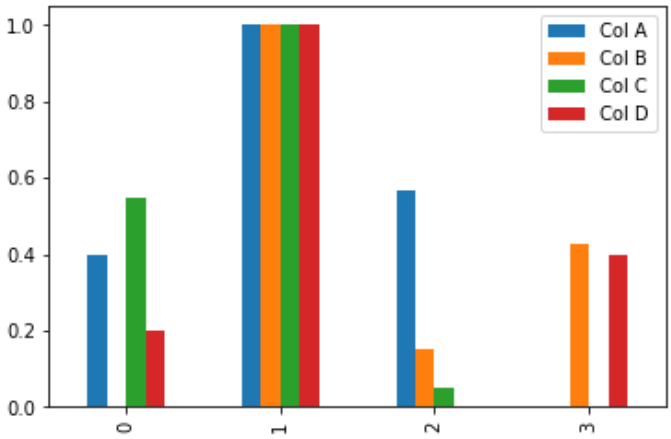
	Col A	Col B	Col C	Col D
0	**0.400000**	**0.000000**	**0.545455**	**0.2**
1	**1.000000**	**1.000000**	**1.000000**	**1.0**
2	**0.566667**	**0.150943**	**0.050505**	**0.0**
3	**0.000000**	**0.427673**	**0.000000**	**0.4**

In [7]:

```
df_min_max_scaled.plot(kind = 'bar')
```

Out[7]:

<AxesSubplot:>



In [8]:

```
df_z_scaled = df.copy()

for column in df_z_scaled.columns:
    df_z_scaled[column] = (df_z_scaled[column]-df_z_scaled[column].mean())/df_z_scaled[c
olumn].std()

display(df_z_scaled)
```

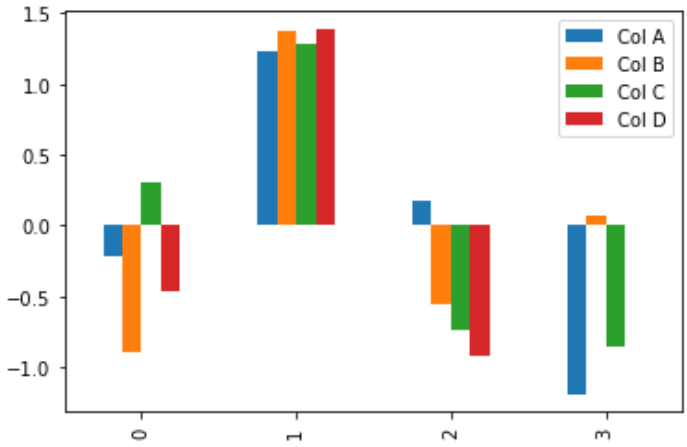
	Col A	Col B	Col C	Col D
0	-0.221422	-0.895492	0.311486	-0.46291
1	1.227884	1.373564	1.278167	1.38873
2	0.181163	-0.552993	-0.741122	-0.92582
3	-1.187625	0.074922	-0.848531	0.00000

In [12]:

```
df_z_scaled.plot(kind = 'bar')
```

Out[12]:

<AxesSubplot:>



In [ ]:

```

```

1. **Turn categorical variables into quantitative variables in Python**

In [13]:

```
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
df.dtypes
```

Out[13]:

```
col1    int64
col2    int64
dtype: object
```

In [14]:

```
df.astype('int32').dtypes
```

Out[14]:

```
col1    int32
col2    int32
dtype: object
```

In [15]:

```
df.astype({'col1': 'int32'}).dtypes
```

Out[15]:

```
col1    int32
col2    int64
dtype: object
```

In [18]:

```
ser = pd.Series([1, 2], dtype='int32')
ser
```

Out[18]:

```
0    1
1    2
dtype: int32
```

In [19]:

```
ser.astype('int64')
```

Out[19]:

```
0    1
1    2
dtype: int64
```

In [20]:

```
ser.astype('category')
```

Out[20]:

```
0    1
1    2
dtype: category
Categories (2, int64): [1, 2]
```

In [21]:

```
from pandas.api.types import CategoricalDtype
cat_dtype = CategoricalDtype(categories=[2, 1], ordered=True)
ser.astype(cat_dtype)
```

Out[21]:

```
0    1
1    2
dtype: category
Categories (2, int64): [2 < 1]
```

In [22]:

```
ser = pd.Series([1, 2])
```

```
s1 = pd.Series([1, 2])
s2 = s1.astype('int64', copy=False)
s2[0] = 10
s1
```

Out[22]:

```
0    10
1     2
dtype: int64
```

In [23]:

```
ser_date = pd.Series(pd.date_range('20200101', periods=3))
ser_date
```

Out[23]:

```
0   2020-01-01
1   2020-01-02
2   2020-01-03
dtype: datetime64[ns]
```

In [40]:

```
path = "C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\sales_data_types.csv"
df = pd.read_csv(path)
df
```

Out[40]:

	Customer Number	Customer Name	2016	2017	Percent Growth	Jan Units	Month	Day	Year	Active
0	10002.0	Quest Industries	$125,000.00	$162500.00	30.00%	500	1	10	2015	Y
1	552278.0	Smith Plumbing	$920,000.00	$101,2000.00	10.00%	700	6	15	2014	Y
2	23477.0	ACME Industrial	$50,000.00	$62500.00	25.00%	125	3	29	2016	Y
3	24900.0	Brekke LTD	$350,000.00	$490000.00	4.00%	75	10	27	2015	Y
4	651029.0	Harbor Co	$15,000.00	$12750.00	-15.00%	Closed	2	2	2014	N

In [4]:

```
df['2016']+df['2017']
```

Out[4]:

```
0      $125,000.00$162500.00
1    $920,000.00$101,2000.00
2        $50,000.00$62500.00
3      $350,000.00$490000.00
4        $15,000.00$12750.00
dtype: object
```

In [6]:

```
df.dtypes
```

Out[6]:

```
Customer Number    float64
Customer Name       object
2016                object
2017                object
Percent Growth      object
Jan Units           object
Month                int64
Day                  int64
Year                 int64
Active              object
dtype: object
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Customer Number  5 non-null      float64
 1   Customer Name    5 non-null      object 
 2   2016             5 non-null      object 
 3   2017             5 non-null      object 
 4   Percent Growth   5 non-null      object 
 5   Jan Units        5 non-null      object 
 6   Month            5 non-null      int64  
 7   Day              5 non-null      int64  
 8   Year             5 non-null      int64  
 9   Active           5 non-null      object 
dtypes: float64(1), int64(3), object(6)
memory usage: 528.0+ bytes
```

In [8]:

```
df['Customer Number'].astype('int')
```

Out[8]:

```
0     10002
1    552278
2     23477
3     24900
4    651029
Name: Customer Number, dtype: int32
```

In [29]:

```
df['Customer Number'] = df['Customer Number'].astype('int')
df.dtypes
```

Out[29]:

```
Customer Number      int32
Customer Name       object
2016               float64
2017               float64
Percent Growth      object
Jan Units           object
Month                int64
Day                  int64
Year                 int64
Active                bool
dtype: object
```

In [13]:

```
df['2016'].astype('float')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-13-999869d577b0> in <module>
----> 1 df['2016'].astype('float')

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\generic.py in astype(self, dtype
, copy, errors)
   5875         else:
   5876             # else, only a single dtype is given
-> 5877             new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5878             return self._constructor(new_data).__finalize__(self, method="astype
")
   5879

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in astype(
self, dtype, copy, errors)
```

```
    629             self, dtype, copy: bool = False, errors: str = "raise"
    630         ) -> "BlockManager":
--> 631             return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    632
    633         def convert(

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply(s
elf, f, align_keys, ignore_failures, **kwargs)
    425                         applied = b.apply(f, **kwargs)
    426                     else:
--> 427                         applied = getattr(b, f)(**kwargs)
    428                 except (TypeError, NotImplementedError):
    429                     if not ignore_failures:

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py in astype(se
lf, dtype, copy, errors)
    671             vals1d = values.ravel()
    672             try:
--> 673                 values = astype_nansafe(vals1d, dtype, copy=True)
    674             except (ValueError, TypeError):
    675                 # e.g. astype_nansafe can fail on object-dtype of strings

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\dtypes\cast.py in astype_nansafe
(arr, dtype, copy, skipna)
   1095     if copy or is_object_dtype(arr) or is_object_dtype(dtype):
   1096         # Explicit copy, or required since NumPy can't view from / to object.
-> 1097         return arr.astype(dtype, copy=True)
   1098
   1099     return arr.view(dtype)

ValueError: could not convert string to float: '$125,000.00'
```

In [14]:

```
df['Jan Units'].astype('int')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-14-31333711e4a4> in <module>
----> 1 df['Jan Units'].astype('int')

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\generic.py in astype(self, dtype
, copy, errors)
   5875         else:
   5876             # else, only a single dtype is given
-> 5877             new_data = self._mgr.astype(dtype=dtype, copy=copy, errors=errors)
   5878             return self._constructor(new_data).__finalize__(self, method="astype
")
   5879

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in astype(
self, dtype, copy, errors)
    629             self, dtype, copy: bool = False, errors: str = "raise"
    630         ) -> "BlockManager":
--> 631             return self.apply("astype", dtype=dtype, copy=copy, errors=errors)
    632
    633         def convert(

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in apply(s
elf, f, align_keys, ignore_failures, **kwargs)
    425                         applied = b.apply(f, **kwargs)
    426                     else:
--> 427                         applied = getattr(b, f)(**kwargs)
    428                 except (TypeError, NotImplementedError):
    429                     if not ignore_failures:

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\internals\blocks.py in astype(se
lf, dtype, copy, errors)
    671             vals1d = values.ravel()
    672             try:
--> 673                 values = astype_nansafe(vals1d, dtype, copy=True)
    674             except (ValueError, TypeError):
```

```
    675                 # e.g. astype_nansafe can fail on object-dtype of strings

D:\Program Files\Anaconda3\lib\site-packages\pandas\core\dtypes\cast.py in astype_nansafe
(arr, dtype, copy, skipna)
   1072         # work around NumPy brokenness, #1987
   1073         if np.issubdtype(dtype.type, np.integer):
-> 1074             return lib.astype_intsafe(arr.ravel(), dtype).reshape(arr.shape)
   1075
   1076         # if we have a datetime/timedelta array of objects

pandas\_libs\lib.pyx in pandas._libs.lib.astype_intsafe()

ValueError: invalid literal for int() with base 10: 'Closed'
```

In [15]:

```
df['Active'].astype('bool')
```

Out[15]:

```
0    True
1    True
2    True
3    True
4    True
Name: Active, dtype: bool
```

In [16]:

```
df
```

Out[16]:

	Customer Number	Customer Name	2016	2017	Percent Growth	Jan Units	Month	Day	Year	Active
0	10002	Quest Industries	$125,000.00	$162500.00	30.00%	500	1	10	2015	Y
1	552278	Smith Plumbing	$920,000.00	$101,2000.00	10.00%	700	6	15	2014	Y
2	23477	ACME Industrial	$50,000.00	$62500.00	25.00%	125	3	29	2016	Y
3	24900	Brekke LTD	$350,000.00	$490000.00	4.00%	75	10	27	2015	Y
4	651029	Harbor Co	$15,000.00	$12750.00	-15.00%	Closed	2	2	2014	N

In [45]:

```
def convert_currency(val):
    new_val = val.replace(',','').replace('$','')
    return float(new_val)
```

In [18]:

```
df['2016'].apply(convert_currency)
```

Out[18]:

```
0    125000.0
1    920000.0
2     50000.0
3    350000.0
4     15000.0
Name: 2016, dtype: float64
```

In [19]:

```
df['2016'].apply(lambda x: x.replace(',','').replace('$','')).astype('float')
```

Out[19]:

```
0    125000.0
1    920000.0
2     50000.0
3    350000.0
```

```
4     15000.0
Name: 2016, dtype: float64
```

In [24]:

```
df['2016'] = df['2016'].apply(convert_currency)
df['2017'] = df['2017'].apply(convert_currency)

df.dtypes
```

Out[24]:

```
Customer Number     float64
Customer Name        object
2016                float64
2017                float64
Percent Growth       object
Jan Units            object
Month                 int64
Day                   int64
Year                  int64
Active               object
dtype: object
```

In [25]:

```
df['Percent Growth'].apply(lambda x: x.replace('%','')).astype('float')/100
```

Out[25]:

```
0    0.30
1    0.10
2    0.25
3    0.04
4   -0.15
Name: Percent Growth, dtype: float64
```

In [46]:

```
def convert_percent(val):
    new_val = val.replace('%','')
    return float(new_val)/100

df['Percent Growth'].apply(convert_percent)
```

Out[46]:

```
0    0.30
1    0.10
2    0.25
3    0.04
4   -0.15
Name: Percent Growth, dtype: float64
```

In [27]:

```
df['Active'] = np.where(df['Active'] == 'Y',True,False)
```

In [28]:

```
df
```

Out[28]:

	Customer Number	Customer Name	2016	2017	Percent Growth	Jan Units	Month	Day	Year	Active
0	10002.0	Quest Industries	125000.0	162500.0	30.00%	500	1	10	2015	True
1	552278.0	Smith Plumbing	920000.0	1012000.0	10.00%	700	6	15	2014	True
2	23477.0	ACME Industrial	50000.0	62500.0	25.00%	125	3	29	2016	True
3	24900.0	Brekke LTD	350000.0	490000.0	4.00%	75	10	27	2015	True

4	Customer Number	Customer Name	2016	2017	Percent Growth	Jan Units	Month	Day	Year	Active

In [30]:

```
df.dtypes
```

Out[30]:

```
Customer Number      int32
Customer Name       object
2016               float64
2017               float64
Percent Growth      object
Jan Units           object
Month                int64
Day                  int64
Year                 int64
Active                bool
dtype: object
```

In [31]:

```
pd.to_numeric(df['Jan Units'], errors='coerce')
```

Out[31]:

```
0    500.0
1    700.0
2    125.0
3     75.0
4      NaN
Name: Jan Units, dtype: float64
```

In [32]:

```
pd.to_numeric(df['Jan Units'], errors='coerce').fillna(0)
```

Out[32]:

```
0    500.0
1    700.0
2    125.0
3     75.0
4      0.0
Name: Jan Units, dtype: float64
```

In [33]:

```
pd.to_datetime(df[['Month','Day','Year']])
```

Out[33]:

```
0   2015-01-10
1   2014-06-15
2   2016-03-29
3   2015-10-27
4   2014-02-02
dtype: datetime64[ns]
```

In [34]:

```
df['Jan Units'] = pd.to_numeric(df['Jan Units'], errors='coerce')
df['Start Date'] = pd.to_datetime(df[['Month','Day','Year']])

df.dtypes
```

Out[34]:

```
Customer Number             int32
Customer Name              object
2016                      float64
2017                      float64
Percent Growth             object
```

```
Jan Units                 float64
Month                       int64
Day                         int64
Year                        int64
Active                       bool
Start Date         datetime64[ns]
dtype: object
```

In [64]:

```
df_2 = pd.read_csv("C:\\Users\\OJUS\\OneDrive\\Desktop\\ \\DBDA\\Data Set\\sales_data_types.csv",
                   dtype={'Customer Number' : 'int'},
                   converters={'2016' : convert_currency,
                               '2017' : convert_currency,
                               'Percent Growth' : convert_percent,
                               'Jan Units' : lambda x: pd.to_numeric(df['Jan Units'], errors='coerce').fillna(0),
                               'Active' : lambda x: np.where(df['Active'] == 'Y',True,False)
                              }
                  )

df_2.dtypes
```

Out[64]:

```
Customer Number      int32
Customer Name       object
2016               float64
2017               float64
Percent Growth     float64
Jan Units           object
Month                int64
Day                  int64
Year                 int64
Active              object
dtype: object
```

In [5]:

```
dictionary = {'OUTLOOK' : ['Rainy', 'Rainy',
                           'Overcast', 'Sunny',
                           'Sunny', 'Sunny',
                           'Overcast', 'Rainy',
                           'Rainy', 'Sunny',
                           'Rainy', 'Overcast',
                           'Overcast', 'Sunny'],
              'TEMPERATURE' : ['Hot', 'Hot',
                               'Hot', 'Mild',
                               'Cool', 'Cool',
                               'Cool', 'Mild',
                               'Cool', 'Mild',
                               'Mild', 'Mild',
                               'Hot', 'Mild'],
              'HUMIDITY': ['High', 'High', 'High',
                           'High', 'Normal', 'Normal',
                           'Normal', 'High', 'Normal',
                           'Normal', 'Normal', 'High',
                           'Normal', 'High'],
              'WINDY': ['No', 'Yes', 'No', 'No',
                        'No','Yes', 'Yes', 'No',
                        'No', 'No', 'Yes', 'Yes',
                        'No','Yes']
             }

df = pd.DataFrame(dictionary)

df
```

Out[5]:

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY
0	Rainy	Hot	High	No
1	Rainy	Hot	High	Yes
2	Overcast	Hot	High	No
3	Sunny	Mild	High	No
4	Sunny	Cool	Normal	No
5	Sunny	Cool	Normal	Yes
6	Overcast	Cool	Normal	Yes
7	Rainy	Mild	High	No
8	Rainy	Cool	Normal	No
9	Sunny	Mild	Normal	No
10	Rainy	Mild	Normal	Yes
11	Overcast	Mild	High	Yes
12	Overcast	Hot	Normal	No
13	Sunny	Mild	High	Yes

In [66]:

```
df2 = df.copy()
df2 = pd.get_dummies(df2, columns = ['WINDY', 'OUTLOOK'])
df2
```

Out[66]:

	TEMPERATURE	HUMIDITY	WINDY_No	WINDY_Yes	OUTLOOK_Overcast	OUTLOOK_Rainy	OUTLOOK_Sunny
0	Hot	High	1	0	0	1	0
1	Hot	High	0	1	0	1	0
2	Hot	High	1	0	1	0	0
3	Mild	High	1	0	0	0	1
4	Cool	Normal	1	0	0	0	1
5	Cool	Normal	0	1	0	0	1
6	Cool	Normal	0	1	1	0	0
7	Mild	High	1	0	0	1	0
8	Cool	Normal	1	0	0	1	0
9	Mild	Normal	1	0	0	0	1
10	Mild	Normal	0	1	0	1	0
11	Mild	High	0	1	1	0	0
12	Hot	Normal	1	0	1	0	0
13	Mild	High	0	1	0	0	1

In [68]:

```
from sklearn.preprocessing import LabelBinarizer

df3 = df.copy()
label_binarizer = LabelBinarizer()
label_binarizer_output = label_binarizer.fit_transform(df3['TEMPERATURE'])
result_df = pd.DataFrame (label_binarizer_output, columns = label_binarizer.classes_)

display(result_df)
```

	Cool	Hot	Mild
0	0	1	0

	Cool	Hot	Mild
1	0	1	0
2	0	1	0
3	0	0	1
4	1	0	0
5	1	0	0
6	1	0	0
7	0	0	1
8	1	0	0
9	0	0	1
10	0	0	1
11	0	0	1
12	0	1	0
13	0	0	1

In [1]:

```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.3.0-py2.py3-none-any.whl (82 kB)
Requirement already satisfied: scikit-learn>=0.20.0 in d:\program files\anaconda3\lib\sit
e-packages (from category_encoders) (0.24.1)
Requirement already satisfied: patsy>=0.5.1 in d:\program files\anaconda3\lib\site-packag
es (from category_encoders) (0.5.1)
Requirement already satisfied: statsmodels>=0.9.0 in d:\program files\anaconda3\lib\site-
packages (from category_encoders) (0.12.2)
Requirement already satisfied: scipy>=1.0.0 in d:\program files\anaconda3\lib\site-packag
es (from category_encoders) (1.6.2)
Requirement already satisfied: pandas>=0.21.1 in d:\program files\anaconda3\lib\site-pack
ages (from category_encoders) (1.2.4)
Requirement already satisfied: numpy>=1.14.0 in d:\program files\anaconda3\lib\site-packa
ges (from category_encoders) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in d:\program files\anaconda3\lib\site-packag
es (from pandas>=0.21.1->category_encoders) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in d:\program files\anaconda3\lib\s
ite-packages (from pandas>=0.21.1->category_encoders) (2.8.1)
Requirement already satisfied: six in d:\program files\anaconda3\lib\site-packages (from
patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\program files\anaconda3\lib\sit
e-packages (from scikit-learn>=0.20.0->category_encoders) (2.1.0)
Requirement already satisfied: joblib>=0.11 in d:\program files\anaconda3\lib\site-packag
es (from scikit-learn>=0.20.0->category_encoders) (1.0.1)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.3.0
```

In [6]:

```
import category_encoders as cat_encoder

df4 = df.copy()

encoder = cat_encoder.BinaryEncoder (cols = df4.columns)
df_category_encoder = encoder.fit_transform(df4)

display(df_category_encoder)
```

	OUTLOOK_0	OUTLOOK_1	TEMPERATURE_0	TEMPERATURE_1	HUMIDITY_0	HUMIDITY_1	WINDY_0	WINDY_1
0	0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	1	0
2	1	0	0	1	0	1	0	1

	OUTLOOK_0	OUTLOOK_1	TEMPERATURE_0	TEMPERATURE_1	HUMIDITY_0	HUMIDITY_1	WINDY_0	WINDY_1
3	1	0	1	0	0	1	0	1
4	1	0	1	0	0	1	1	0
5	1	0	1	0	0	1	1	0
6	1	0	1	0	0	1	1	0
7	0	1	1	0	0	1	0	1
8	0	1	1	0	0	1	0	1
9	1	0	1	0	0	1	0	1
10	0	1	1	0	0	1	0	1
11	1	0	1	0	0	1	1	0
12	1	0	0	1	0	1	0	1
13	1	0	1	0	0	1	1	0