Numpy

Filtering

In [2]:
```
l = [10,20,30,40,50,60]
for i in l:
    if i>30:
        print(i)
```

```
40
50
60
```

In [3]:
```
[i for i in l if i>30]
```

Out[3]:
```
[40, 50, 60]
```

In [4]:
```
import numpy as np
```

In [5]:
```
arr = np.array([10,20,30,40,50,60])
arr
```

Out[5]:
```
array([10, 20, 30, 40, 50, 60])
```

In [6]:
```
arr>30
```

Out[6]:
```
array([False, False, False,  True,  True,  True])
```

In [7]:
```
arr[arr>30]
```

Out[7]:
```
array([40, 50, 60])
```

In [8]:
```
x = np.arange(50,100)
x
```

Out[8]:
```
array([50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,
       67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83,
       84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [9]:
```
x%2==0
```

Out[9]:
```
array([ True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False])
```

In [12]:
```
1 x[x%2==0]
```

Out[12]:
```
array([50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82,
       84, 86, 88, 90, 92, 94, 96, 98])
```

In [17]:
```
1 x[(x>60)&(x<90)]
```

Out[17]:
```
array([61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77,
       78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89])
```

In [20]:
```
1 x[(x>60) & (x<90) &(x%2==0)]
2
```

Out[20]:
```
array([62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88])
```

In [21]:
```
1 x[((x>60)&(x<90))&(x%2==0)]
2
```

Out[21]:
```
array([62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88])
```

```
1 |---- or
2 !---- not
```

Pandas

- Data Analysis and Data Manipulation
- It is in the format of table(rows,columns)
- Import / Export the data

2 types:

- Series - 1d array - list,tuple,array
  - data types - float,int,string/object
- DataFrame - 2d array - list,tuple,dict,array
  - data types - float,int,string/object

In [24]:
```
1 import pandas as pd
```

In [25]:
```
1 # pip install pandas
```

In [28]:
```
a = pd.Series([1,2,3,4,7.8,"apssdc"])
a
```

Out[28]:
```
0         1
1         2
2         3
3         4
4       7.8
5    apssdc
dtype: object
```

In [31]:
```
b = pd.Series((1,2,3,4,7.8))
b
```

Out[31]:
```
0    1.0
1    2.0
2    3.0
3    4.0
4    7.8
dtype: float64
```

In [34]:
```
b = pd.Series((1,2,3,4,7.8),index=["a","b","c","d","e"])
b
```

Out[34]:
```
a    1.0
b    2.0
c    3.0
d    4.0
e    7.8
dtype: float64
```

In [35]:
```
b.index
```

Out[35]:
```
Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

In [36]:
```
a.index
```

Out[36]:
```
RangeIndex(start=0, stop=6, step=1)
```

In [37]:
```
b
```

Out[37]:
```
a    1.0
b    2.0
c    3.0
d    4.0
e    7.8
dtype: float64
```

In [38]:
```
b["e"]
```

Out[38]:
```
7.8
```

In [39]:
```
1 pd.DataFrame([1,2,3,4])
```

Out[39]:

	0
0	1
1	2
2	3
3	4

In [40]:
```
1 d = pd.DataFrame([["apssdc",14,7],["ml",15,7]])
2 d
```

Out[40]:

	0	1	2
0	apssdc	14	7
1	ml	15	7

In [41]:
```
1 d = pd.Series([["apssdc",14,7],["ml",15,7]])
2 d
```

Out[41]:
```
0    [apssdc, 14, 7]
1       [ml, 15, 7]
dtype: object
```

In [43]:
```
1 d = pd.DataFrame([["apssdc",14,7],["ml",15,7]],columns =("name","date","mont
2                  ,index = ["a","b"])
3 d
```

Out[43]:

	name	date	month
a	apssdc	14	7
b	ml	15	7

In [ ]:
```
1 {key:values}
```

In [48]:
```
1 d1 = pd.DataFrame({"a":[1,2,3],"b":[4,5,6]},index = [1,2,3])
2 d1
```

Out[48]:

	a	b
1	1	4
2	2	5
3	3	6

In [49]:
```
1 d1.index
```

Out[49]:
```
Int64Index([1, 2, 3], dtype='int64')
```

In [50]:
```
1 d1.columns
```

Out[50]:
```
Index(['a', 'b'], dtype='object')
```

In [51]:
```
1 d1.shape
```

Out[51]:
```
(3, 2)
```

In [52]:
```
1 # Accessing the elements
```

In [53]:
```
1 d
```

Out[53]:

	name	date	month
a	apssdc	14	7
b	ml	15	7

In [54]:
```
1 d.shape
```

Out[54]:
```
(2, 3)
```

In [56]:
```
1 d["name"]
```

Out[56]:
```
a    apssdc
b        ml
Name: name, dtype: object
```

In [62]:
```
1 d.name
```

Out[62]:
```
a    apssdc
b        ml
Name: name, dtype: object
```

In [57]:
```
1 d["month"]
```

Out[57]:
```
a    7
b    7
Name: month, dtype: int64
```

In [61]:
```
d[["month","name","date"]]
```

Out[61]:

	month	name	date
a	7	apssdc	14
b	7	ml	15

In [65]:
```
d1[["a","b"]]
```

Out[65]:

	a	b
1	1	4
2	2	5
3	3	6

In [66]:
```
d
```

Out[66]:

	name	date	month
a	apssdc	14	7
b	ml	15	7

In [70]:
```
d["date"]["b"]
```

Out[70]: 15

In [71]:
```
d1
```

Out[71]:

	a	b
1	1	4
2	2	5
3	3	6

In [72]:
```
d1["a"][2]
```

Out[72]: 2

In [80]:
```
d1[-1:]
```

Out[80]:

	a	b
3	3	6

In [81]:
```
1 d1[0:1]
```

Out[81]:

	a	b
1	1	4

- Load datasets

In [82]:
```
1 data = pd.read_csv("market.csv")
2
```

In [83]:
```
1 data
2
```

Out[83]:

Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51	
Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56	
Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90	
Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34	
Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87	
Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64	
Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32	
Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.91	
Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84	
Ord_4725	Prod_6	SHP_6593	Cust_1641	57.2200	0.07	8	-27.72	
Ord_4743	Prod_2	SHP_6615	Cust_1641	4072.0100	0.01	43	1675.98	
Ord_1925	Prod_6	SHP_2637	Cust_708	465.9000	0.05	38	79.34	
Ord_2978	Prod_16	SHP_4112	Cust_1088	305.0500	0.04	27	23.12	
Ord_2207	Prod_11	SHP_3093	Cust_839	3364.2480	0.10	15	-693.23	
Ord_2207	Prod_10	SHP_3006	Cust_839	1410.9300	0.08	10	-317.48	
Ord_2280	Prod_5	SHP_3114	Cust_839	460.6900	0.06	48	-103.48	
Ord_2282	Prod_9	SHP_3122	Cust_839	443.4600	0.06	30	193.12	
Ord_4471	Prod_15	SHP_6228	Cust_1521	13255.9300	0.02	25	4089.27	
Ord_4427	Prod_6	SHP_6171	Cust_1521	283.1300	0.08	45	-141.26	
Ord_996	Prod_13	SHP_1378	Cust_371	41.9700	0.05	12	-37.03	
Ord_996	Prod_13	SHP_1378	Cust_371	57.1700	0.08	18	-24.03	
Ord_996	Prod_6	SHP_1378	Cust_371	81.2500	0.01	11	-44.54	
Ord_996	Prod_5	SHP_1377	Cust_371	3202.2500	0.09	44	991.26	
Ord_996	Prod_7	SHP_1378	Cust_371	35.6400	0.05	10	-0.71	
Ord_2573	Prod_3	SHP_3525	Cust_931	197.6100	0.08	13	3.46	
Ord_2335	Prod_13	SHP_3204	Cust_931	38.2600	0.03	22	-2.34	
Ord_2456	Prod_5	SHP_3367	Cust_931	109.5800	0.00	13	31.32	
Ord_2405	Prod_9	SHP_3300	Cust_931	1062.6900	0.01	28	401.80	
Ord_2573	Prod_4	SHP_3527	Cust_931	3594.7435	0.05	38	1016.97	
Ord_2478	Prod_12	SHP_3395	Cust_931	139.9800	0.07	33	-140.54	
...	...	...	...	...	...	...	...	
Ord_3633	Prod_3	SHP_5031	Cust_1274	1169.2600	0.02	41	515.62	
Ord_2696	Prod_13	SHP_3690	Cust_1006	62.7800	0.04	20	-17.75	

Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
Ord_2624	Prod_4	SHP_3591	Cust_1006	4924.1350	0.07	28	1049.54	
Ord_2772	Prod_9	SHP_3806	Cust_1006	56.9000	0.03	7	12.64	
Ord_2600	Prod_16	SHP_3560	Cust_1006	106.6400	0.10	30	-31.95	
Ord_2658	Prod_5	SHP_3637	Cust_1006	1082.6600	0.08	14	-256.93	
Ord_2772	Prod_3	SHP_3806	Cust_1006	1413.8200	0.10	47	226.53	
Ord_2624	Prod_8	SHP_3590	Cust_1006	1211.0000	0.00	36	-27.99	
Ord_2722	Prod_12	SHP_3729	Cust_1006	34.0100	0.00	12	10.58	
Ord_2706	Prod_2	SHP_3705	Cust_1006	1361.9100	0.05	20	312.52	
Ord_2722	Prod_5	SHP_3730	Cust_1006	1008.9500	0.04	41	69.31	
Ord_2772	Prod_6	SHP_3807	Cust_1006	308.9200	0.04	45	-143.58	
Ord_2696	Prod_4	SHP_3691	Cust_1006	2836.0505	0.01	25	561.13	
Ord_2658	Prod_3	SHP_3636	Cust_1006	120.9800	0.00	28	-92.85	
Ord_2722	Prod_1	SHP_3731	Cust_1006	3508.3300	0.04	21	-546.98	
Ord_4620	Prod_3	SHP_6435	Cust_1577	59.6200	0.04	10	-56.30	
Ord_1833	Prod_3	SHP_2527	Cust_637	611.1600	0.04	46	100.22	
Ord_2324	Prod_7	SHP_3189	Cust_851	121.8700	0.07	39	11.32	
Ord_2220	Prod_3	SHP_3019	Cust_851	41.0600	0.04	4	-16.39	
Ord_4424	Prod_1	SHP_6165	Cust_1519	994.0400	0.03	10	-335.06	
Ord_4444	Prod_13	SHP_6192	Cust_1519	159.4100	0.00	44	34.68	
Ord_5435	Prod_16	SHP_7594	Cust_1798	316.9900	0.04	47	-276.54	
Ord_5435	Prod_4	SHP_7594	Cust_1798	1991.8985	0.07	20	88.36	
Ord_5384	Prod_9	SHP_7519	Cust_1798	181.5000	0.08	43	-6.24	
Ord_5348	Prod_8	SHP_7470	Cust_1798	356.7200	0.07	9	12.61	
Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	
Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	
Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	
Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	
Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	

rows × 10 columns

In [84]:
```
1 data.columns
```

Out[84]:
```
Index(['Ord_id', 'Prod_id', 'Ship_id', 'Cust_id', 'Sales', 'Discount',
       'Order_Quantity', 'Profit', 'Shipping_Cost', 'Product_Base_Margin'],
      dtype='object')
```

In [85]:
```
1 data.shape
```

Out[85]: (8399, 10)

In [86]:
```
1 data.head()
```

Out[86]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [87]:
```
1 data.index
```

Out[87]: RangeIndex(start=0, stop=8399, step=1)

In [88]:
```
1 data.head(10)
```

Out[88]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shippi
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.8100	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.2700	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.6900	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.8900	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.1500	0.08	35	1219.87	
5	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.0200	0.03	23	-47.64	
6	Ord_31	Prod_12	SHP_41	Cust_26	14.7600	0.01	5	1.32	
7	Ord_4725	Prod_4	SHP_6593	Cust_1641	3410.1575	0.10	48	1137.91	
8	Ord_4725	Prod_13	SHP_6593	Cust_1641	162.0000	0.01	33	45.84	
9	Ord_4725	Prod_6	SHP_6593	Cust_1641	57.2200	0.07	8	-27.72	

In [89]:
```
1 data.tail()
```

Out[89]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Ship
8394	Ord_5353	Prod_4	SHP_7479	Cust_1798	2841.4395	0.08	28	374.63	
8395	Ord_5411	Prod_6	SHP_7555	Cust_1798	127.1600	0.10	20	-74.03	
8396	Ord_5388	Prod_6	SHP_7524	Cust_1798	243.0500	0.02	39	-70.85	
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.8700	0.03	23	565.34	
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.6900	0.00	47	131.39	

In [90]:
```
1 data.tail(2)
```

Out[90]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shippi
8397	Ord_5348	Prod_15	SHP_7469	Cust_1798	3872.87	0.03	23	565.34	
8398	Ord_5459	Prod_6	SHP_7628	Cust_1798	603.69	0.00	47	131.39	

In [93]:
```
1 data.sample(3)
```

Out[93]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipp
3351	Ord_2613	Prod_2	SHP_3575	Cust_1012	1757.43	0.02	40	490.77	
6594	Ord_2083	Prod_5	SHP_2847	Cust_793	3991.99	0.00	42	-1014.11	
3335	Ord_477	Prod_1	SHP_643	Cust_164	783.96	0.05	38	-1195.29	

In [94]:
```
1 data.columns
```

Out[94]:
```
Index(['Ord_id', 'Prod_id', 'Ship_id', 'Cust_id', 'Sales', 'Discount',
       'Order_Quantity', 'Profit', 'Shipping_Cost', 'Product_Base_Margin'],
      dtype='object')
```

In [95]:
```
1 data.index
```

Out[95]:
```
RangeIndex(start=0, stop=8399, step=1)
```

In [96]:
```
1 data.shape
```

Out[96]:
```
(8399, 10)
```

In [97]:
```
1 # statistical analysis
```

In [98]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8399 entries, 0 to 8398
Data columns (total 10 columns):
Ord_id                  8399 non-null object
Prod_id                 8399 non-null object
Ship_id                 8399 non-null object
Cust_id                 8399 non-null object
Sales                   8399 non-null float64
Discount                8399 non-null float64
Order_Quantity          8399 non-null int64
Profit                  8399 non-null float64
Shipping_Cost           8399 non-null float64
Product_Base_Margin     8336 non-null float64
dtypes: float64(5), int64(1), object(4)
memory usage: 656.2+ KB
```

In [99]:
```
1 8399-8336
```

Out[99]: 63

In [100]:
```
1 data.describe()
```

Out[100]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Mar
count	8399.000000	8399.000000	8399.000000	8399.000000	8399.000000	8336.000
mean	1775.878179	0.049671	25.571735	181.184424	12.838557	0.512
std	3585.050525	0.031823	14.481071	1196.653371	17.264052	0.135
min	2.240000	0.000000	1.000000	-14140.700000	0.490000	0.350
25%	143.195000	0.020000	13.000000	-83.315000	3.300000	0.380
50%	449.420000	0.050000	26.000000	-1.500000	6.070000	0.520
75%	1709.320000	0.080000	38.000000	162.750000	13.990000	0.590
max	89061.050000	0.250000	50.000000	27220.690000	164.730000	0.850

In [102]:
```
1 data.isnull().sum()
```

```
Ord_id                  0
Prod_id                 0
Ship_id                 0
Cust_id                 0
Sales                   0
Discount                0
Order_Quantity          0
Profit                  0
Shipping_Cost           0
Product_Base_Margin    63
dtype: int64
```

In [103]:
```
1 data.isnull().sum().sum()
```

Out[103]: 63

In [104]:
```
1 # selecting rows
```

In [105]:
```
1 data[10:11]
```

Out[105]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shippin
10	Ord_4743	Prod_2	SHP_6615	Cust_1641	4072.01	0.01	43	1675.98	

In [106]:
```
1 data[100:110]
```

Out[106]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shippi
100	Ord_1638	Prod_8	SHP_5892	Cust_1433	220.79	0.00	7	-91.51	
101	Ord_1638	Prod_7	SHP_2261	Cust_565	201.14	0.08	37	-51.25	
102	Ord_1557	Prod_2	SHP_2152	Cust_565	1736.53	0.10	46	457.03	
103	Ord_209	Prod_13	SHP_285	Cust_45	30.51	0.09	11	-17.68	
104	Ord_246	Prod_8	SHP_341	Cust_45	862.20	0.10	29	-45.10	
105	Ord_122	Prod_17	SHP_165	Cust_45	42.31	0.06	2	-53.08	
106	Ord_121	Prod_8	SHP_164	Cust_45	1143.45	0.06	31	304.42	
107	Ord_250	Prod_15	SHP_346	Cust_45	8901.78	0.04	31	2795.36	
108	Ord_122	Prod_3	SHP_166	Cust_45	152.67	0.04	11	12.76	
109	Ord_139	Prod_17	SHP_186	Cust_45	2671.21	0.06	14	636.18	

In [107]:
```
1 data[100:120:3]
```

Out[107]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipp
100	Ord_1638	Prod_8	SHP_5892	Cust_1433	220.7900	0.00	7	-91.51	
103	Ord_209	Prod_13	SHP_285	Cust_45	30.5100	0.09	11	-17.68	
106	Ord_121	Prod_8	SHP_164	Cust_45	1143.4500	0.06	31	304.42	
109	Ord_139	Prod_17	SHP_186	Cust_45	2671.2100	0.06	14	636.18	
112	Ord_209	Prod_1	SHP_286	Cust_45	127.8000	0.07	8	-30.48	
115	Ord_23	Prod_12	SHP_31	Cust_18	282.0700	0.03	39	140.01	
118	Ord_5351	Prod_4	SHP_7474	Cust_1800	1463.0965	0.00	14	162.67	

In [108]:
```
1 # selecting the columns
```

In [110]:
```
1 data["Ship_id"].head()
```

Out[110]:
```
0    SHP_7609
1    SHP_7549
2    SHP_7610
3    SHP_7625
4    SHP_7664
Name: Ship_id, dtype: object
```

In [112]:
```
1 data[["Ship_id","Cust_id","Sales"]].head()
```

Out[112]:

	Ship_id	Cust_id	Sales
0	SHP_7609	Cust_1818	136.81
1	SHP_7549	Cust_1818	42.27
2	SHP_7610	Cust_1818	4701.69
3	SHP_7625	Cust_1818	2337.89
4	SHP_7664	Cust_1818	4233.15

In [113]:
```
1 # iloc and loc
2
3 # iloc --> position based index
4 # loc ---> label based index
```

In [114]:
```
1 data.iloc[100]
```

Out[114]:
```
Ord_id                Ord_1638
Prod_id                 Prod_8
Ship_id               SHP_5892
Cust_id              Cust_1433
Sales                   220.79
Discount                     0
Order_Quantity               7
Profit                   -91.51
Shipping_Cost                4
Product_Base_Margin       0.74
Name: 100, dtype: object
```

In [116]:
```
1 data[100:101]
```

Out[116]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_
100	Ord_1638	Prod_8	SHP_5892	Cust_1433	220.79	0.0	7	-91.51	

In [117]:
```
1 data.iloc[100:200]
```

Out[117]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Sh
100	Ord_1638	Prod_8	SHP_5892	Cust_1433	220.7900	0.00	7	-91.51	
101	Ord_1638	Prod_7	SHP_2261	Cust_565	201.1400	0.08	37	-51.25	
102	Ord_1557	Prod_2	SHP_2152	Cust_565	1736.5300	0.10	46	457.03	
103	Ord_209	Prod_13	SHP_285	Cust_45	30.5100	0.09	11	-17.68	
104	Ord_246	Prod_8	SHP_341	Cust_45	862.2000	0.10	29	-45.10	
105	Ord_122	Prod_17	SHP_165	Cust_45	42.3100	0.06	2	-53.08	
106	Ord_121	Prod_8	SHP_164	Cust_45	1143.4500	0.06	31	304.42	
107	Ord_250	Prod_15	SHP_346	Cust_45	8901.7800	0.04	31	2795.36	
108	Ord_122	Prod_3	SHP_166	Cust_45	152.6700	0.04	11	12.76	
109	Ord_139	Prod_17	SHP_186	Cust_45	2671.2100	0.06	14	636.18	
110	Ord_239	Prod_4	SHP_332	Cust_45	2157.3085	0.00	38	519.25	
111	Ord_250	Prod_2	SHP_345	Cust_45	324.5500	0.08	5	-12.82	
112	Ord_209	Prod_1	SHP_286	Cust_45	127.8000	0.07	8	-30.48	
113	Ord_135	Prod_9	SHP_182	Cust_45	30.8300	0.04	2	7.27	
114	Ord_60	Prod_12	SHP_81	Cust_45	176.2600	0.07	44	75.13	
115	Ord_23	Prod_12	SHP_31	Cust_18	282.0700	0.03	39	140.01	
116	Ord_23	Prod_4	SHP_31	Cust_18	426.0370	0.01	24	-78.96	
117	Ord_5374	Prod_8	SHP_7504	Cust_1800	1222.5900	0.02	29	501.99	
118	Ord_5351	Prod_4	SHP_7474	Cust_1800	1463.0965	0.00	14	162.67	
119	Ord_5417	Prod_12	SHP_7565	Cust_1800	128.1300	0.09	42	32.53	
120	Ord_5354	Prod_5	SHP_7480	Cust_1800	152.4400	0.08	20	-16.64	
121	Ord_5423	Prod_17	SHP_7577	Cust_1800	13070.2000	0.07	4	-6923.60	
122	Ord_5351	Prod_6	SHP_7475	Cust_1800	246.3000	0.02	38	-67.49	
123	Ord_2340	Prod_5	SHP_3210	Cust_933	136.6100	0.01	8	80.43	
124	Ord_2432	Prod_4	SHP_3339	Cust_933	3883.4715	0.10	42	707.17	
125	Ord_2480	Prod_3	SHP_3397	Cust_933	169.6100	0.10	29	-211.06	
126	Ord_2394	Prod_13	SHP_3284	Cust_933	64.3600	0.06	37	-1.53	
127	Ord_2372	Prod_10	SHP_3256	Cust_933	1764.9700	0.07	13	-383.50	
128	Ord_2566	Prod_13	SHP_3515	Cust_933	422.2500	0.06	35	-116.02	
129	Ord_2340	Prod_11	SHP_3211	Cust_933	2907.6300	0.04	42	54.60	
...	...	...	...	...	...	...	...	...	
170	Ord_2973	Prod_4	SHP_4102	Cust_1480	6264.1855	0.01	34	1312.04	
171	Ord_2973	Prod_9	SHP_6073	Cust_1480	5410.9500	0.09	36	2077.91	
172	Ord_4572	Prod_6	SHP_6360	Cust_1480	377.8300	0.06	41	122.06	

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Sh
173	Ord_1995	Prod_6	SHP_2724	Cust_702	240.1400	0.06	36	57.78	
174	Ord_1915	Prod_6	SHP_2623	Cust_702	980.9500	0.02	41	234.68	
175	Ord_2273	Prod_5	SHP_3101	Cust_1104	180.4300	0.02	8	-44.86	
176	Ord_2273	Prod_8	SHP_3101	Cust_878	2899.9800	0.10	19	666.01	
177	Ord_2273	Prod_6	SHP_3102	Cust_878	233.3800	0.10	36	-180.17	
178	Ord_2246	Prod_2	SHP_3054	Cust_878	294.5200	0.09	14	15.34	
179	Ord_2273	Prod_4	SHP_3100	Cust_878	1077.8085	0.06	20	17.25	
180	Ord_5153	Prod_13	SHP_7201	Cust_1753	176.5000	0.01	46	-119.35	
181	Ord_5153	Prod_1	SHP_7201	Cust_1753	1546.8000	0.07	42	25.51	
182	Ord_5239	Prod_13	SHP_7318	Cust_1753	452.2800	0.09	48	-21.00	
183	Ord_5157	Prod_15	SHP_7208	Cust_1753	5149.0600	0.03	27	605.44	
184	Ord_5230	Prod_4	SHP_7304	Cust_1753	2197.4115	0.00	20	305.96	
185	Ord_5153	Prod_11	SHP_7202	Cust_1753	3457.5600	0.02	35	-365.44	
186	Ord_5159	Prod_5	SHP_7210	Cust_1753	2213.9200	0.03	20	768.34	
187	Ord_5230	Prod_1	SHP_7304	Cust_1753	423.0400	0.05	2	-270.70	
188	Ord_5239	Prod_13	SHP_7318	Cust_1753	55.8200	0.01	18	14.19	
189	Ord_2831	Prod_3	SHP_3893	Cust_1036	18092.6600	0.09	36	7917.76	
190	Ord_2791	Prod_12	SHP_3837	Cust_1036	68.5400	0.10	25	22.20	
191	Ord_2791	Prod_11	SHP_3838	Cust_1036	10351.0100	0.08	19	-1331.55	
192	Ord_2739	Prod_5	SHP_3758	Cust_1036	363.9200	0.00	44	-144.56	
193	Ord_2585	Prod_11	SHP_3540	Cust_997	7287.5500	0.04	24	-715.78	
194	Ord_2653	Prod_6	SHP_3630	Cust_997	306.3000	0.07	34	18.73	
195	Ord_2691	Prod_5	SHP_3685	Cust_997	311.1900	0.03	33	-9.18	
196	Ord_2753	Prod_15	SHP_3779	Cust_997	2549.5800	0.02	10	41.53	
197	Ord_2753	Prod_3	SHP_3778	Cust_997	23106.4600	0.08	28	9527.47	
198	Ord_2653	Prod_13	SHP_3630	Cust_997	163.9800	0.01	42	55.73	
199	Ord_2799	Prod_15	SHP_3850	Cust_997	11002.6600	0.07	46	2349.29	

100 rows × 10 columns

In [119]:
```
1 data.iloc[[2,10,45,1000,3000,4]]
```

Out[119]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipp
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
10	Ord_4743	Prod_2	SHP_6615	Cust_1641	4072.01	0.01	43	1675.98	
45	Ord_4768	Prod_12	SHP_6650	Cust_1579	3.85	0.08	1	-1.36	
1000	Ord_1895	Prod_4	SHP_2601	Cust_686	2433.55	0.05	23	355.93	
3000	Ord_5342	Prod_3	SHP_7459	Cust_1771	242.46	0.02	44	9.30	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [120]:
```
1 data.iloc[3:10,4:]
```

Out[120]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
3	2337.8900	0.09	43	729.34	14.30	0.37
4	4233.1500	0.08	35	1219.87	26.30	0.38
5	164.0200	0.03	23	-47.64	6.15	0.37
6	14.7600	0.01	5	1.32	0.50	0.36
7	3410.1575	0.10	48	1137.91	0.99	0.55
8	162.0000	0.01	33	45.84	0.71	0.52
9	57.2200	0.07	8	-27.72	6.60	0.37

In [122]:
```
1 data.loc[4,"Sales"]
```

Out[122]:
```
4233.15
```

In [123]:
```
1 data.loc[9,"Order_Quantity"]
```

Out[123]:
```
8
```

In [125]:
```
1 data["Order_Quantity"].head()
```

Out[125]:
```
0    23
1    13
2    26
3    43
4    35
Name: Order_Quantity, dtype: int64
```

In [126]:
```
1 # Filtering
```

In [128]:
```
1 data.head()
```

Out[128]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [131]:
```
1 data[(data["Sales"]>4000) & (data["Profit"]>4000)]
```

Out[131]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shi
17	Ord_4471	Prod_15	SHP_6228	Cust_1521	13255.93	0.02	25	4089.27	
133	Ord_2432	Prod_3	SHP_3338	Cust_933	15337.58	0.10	30	6670.41	
165	Ord_4702	Prod_3	SHP_6560	Cust_1603	11823.52	0.10	34	4592.74	
189	Ord_2831	Prod_3	SHP_3893	Cust_1036	18092.66	0.09	36	7917.76	
197	Ord_2753	Prod_3	SHP_3778	Cust_997	23106.46	0.08	28	9527.47	
385	Ord_3707	Prod_17	SHP_5136	Cust_1307	28359.40	0.05	49	14440.39	
535	Ord_4857	Prod_17	SHP_6777	Cust_227	19100.45	0.00	40	6839.95	
699	Ord_2839	Prod_17	SHP_3905	Cust_1048	14383.83	0.01	37	5050.10	
734	Ord_4705	Prod_17	SHP_6566	Cust_1592	21320.58	0.09	27	5381.02	
755	Ord_1026	Prod_17	SHP_1417	Cust_363	15602.93	0.00	48	4875.89	
808	Ord_4322	Prod_17	SHP_6027	Cust_1440	15152.55	0.05	34	7719.21	
954	Ord_1989	Prod_10	SHP_2718	Cust_750	19269.05	0.00	38	4127.79	
1184	Ord_5181	Prod_17	SHP_7238	Cust_1750	23775.56	0.03	44	7080.99	
1220	Ord_283	Prod_17	SHP_383	Cust_85	10051.52	0.05	48	4938.78	
1252	Ord_1536	Prod_14	SHP_2125	Cust_561	14475.74	0.06	28	4963.89	
1265	Ord_5250	Prod_17	SHP_7331	Cust_1769	24105.87	0.07	14	4073.25	
1324	Ord_214	Prod_17	SHP_297	Cust_82	15251.50	0.06	31	5353.19	
1448	Ord_2673	Prod_1	SHP_3661	Cust_1027	21337.27	0.02	49	7606.00	
1471	Ord_893	Prod_17	SHP_1228	Cust_315	18028.07	0.02	36	8157.70	
1494	Ord_188	Prod_14	SHP_253	Cust_97	17387.65	0.08	35	6907.61	
1576	Ord_5416	Prod_3	SHP_7562	Cust_1812	14410.78	0.02	37	6365.58	
1681	Ord_809	Prod_17	SHP_1113	Cust_266	18775.76	0.03	45	8504.47	
1740	Ord_3667	Prod_17	SHP_5082	Cust_1291	21532.26	0.09	44	8323.39	
1787	Ord_2872	Prod_14	SHP_7219	Cust_1748	21555.60	0.02	47	8965.83	
1792	Ord_5148	Prod_14	SHP_7194	Cust_1748	19224.92	0.10	47	6635.15	
1848	Ord_5315	Prod_14	SHP_7423	Cust_1780	21141.07	0.06	49	6225.36	
2013	Ord_4305	Prod_14	SHP_6008	Cust_1436	17599.39	0.03	28	5513.86	
2216	Ord_4216	Prod_14	SHP_5881	Cust_1432	26126.92	0.04	42	9498.60	
2253	Ord_3143	Prod_14	SHP_4362	Cust_1170	28664.52	0.09	50	13340.26	
2259	Ord_1978	Prod_17	SHP_2703	Cust_725	25312.00	0.01	48	8788.81	
...	...	...	...	...	...	...	...	...	
6218	Ord_1764	Prod_3	SHP_2445	Cust_578	18697.24	0.02	44	8918.74	
6252	Ord_4757	Prod_15	SHP_6633	Cust_1604	16451.33	0.01	31	5325.14	
6323	Ord_3581	Prod_1	SHP_4958	Cust_1264	17853.64	0.02	41	6227.33	

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shi
6384	Ord_4473	Prod_17	SHP_6232	Cust_1535	20265.22	0.08	47	6168.64	
6484	Ord_3627	Prod_17	SHP_5022	Cust_1281	24559.91	0.01	47	7358.66	
6494	Ord_1293	Prod_17	SHP_1784	Cust_483	17717.34	0.00	37	8291.08	
6653	Ord_5391	Prod_14	SHP_7527	Cust_1799	24233.54	0.07	43	6492.67	
6660	Ord_5425	Prod_14	SHP_7580	Cust_1799	27720.98	0.07	46	11984.40	
6705	Ord_706	Prod_2	SHP_966	Cust_242	16002.29	0.09	47	4604.79	
6758	Ord_128	Prod_17	SHP_174	Cust_87	16066.85	0.07	31	7416.43	
6759	Ord_2163	Prod_15	SHP_2947	Cust_815	15963.09	0.03	45	4276.73	
6765	Ord_5186	Prod_17	SHP_7247	Cust_1763	26095.13	0.03	35	12606.81	
6797	Ord_1265	Prod_14	SHP_1749	Cust_466	24051.49	0.07	41	9791.04	
6855	Ord_4336	Prod_17	SHP_6043	Cust_1461	15260.78	0.07	39	8734.88	
6926	Ord_2171	Prod_15	SHP_2958	Cust_800	23239.96	0.06	47	6888.36	
6979	Ord_3236	Prod_10	SHP_4488	Cust_1194	21062.91	0.01	23	5713.53	
7006	Ord_5361	Prod_3	SHP_7489	Cust_1793	21752.01	0.03	25	9296.35	
7015	Ord_1151	Prod_3	SHP_1585	Cust_442	15383.70	0.03	50	6523.26	
7091	Ord_911	Prod_10	SHP_1255	Cust_302	28180.08	0.02	32	7513.88	
7286	Ord_1767	Prod_14	SHP_2450	Cust_583	16743.76	0.00	23	6079.63	
7547	Ord_3170	Prod_10	SHP_4400	Cust_1162	29345.27	0.03	34	7497.55	
7592	Ord_3229	Prod_17	SHP_4479	Cust_1195	16172.44	0.03	32	7176.12	
7638	Ord_4953	Prod_3	SHP_6911	Cust_1692	10094.43	0.03	24	4451.01	
7706	Ord_79	Prod_17	SHP_105	Cust_42	22079.47	0.06	12	5322.14	
7964	Ord_1686	Prod_15	SHP_2331	Cust_581	19342.84	0.01	39	5603.95	
8042	Ord_5357	Prod_11	SHP_7483	Cust_1803	14451.75	0.01	40	4503.63	
8078	Ord_5368	Prod_14	SHP_7497	Cust_1795	17279.62	0.04	40	4176.25	
8217	Ord_3359	Prod_10	SHP_7245	Cust_1762	28389.14	0.07	33	7132.18	
8292	Ord_1765	Prod_14	SHP_2446	Cust_595	14647.26	0.07	25	5485.15	
8366	Ord_3593	Prod_3	SHP_4974	Cust_1274	12073.06	0.03	39	5081.87	

128 rows × 10 columns

Data Preprocessing using scikit-Learn

- Problems:
  - insufficient of data
  - outliers
  - missing data

    - too much of data
    - duplicate data
- Scaling techniques
    - Standarad scalar
    - Robust scalar
    - Mix-Max scalar/range scalar

In [134]:
```
1 from sklearn import preprocessing
```

In [135]:
```
1 print(dir(preprocessing))
```

```
['Binarizer', 'CategoricalEncoder', 'FunctionTransformer', 'Imputer', 'KBinsDis
cretizer', 'KernelCenterer', 'LabelBinarizer', 'LabelEncoder', 'MaxAbsScaler',
'MinMaxScaler', 'MultiLabelBinarizer', 'Normalizer', 'OneHotEncoder', 'OrdinalE
ncoder', 'PolynomialFeatures', 'PowerTransformer', 'QuantileTransformer', 'Robu
stScaler', 'StandardScaler', '__all__', '__builtins__', '__cached__', '__doc_
_', '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec_
_', '_discretization', '_encoders', '_function_transformer', 'add_dummy_featur
e', 'base', 'binarize', 'data', 'imputation', 'label', 'label_binarize', 'maxab
s_scale', 'minmax_scale', 'normalize', 'power_transform', 'quantile_transform',
'robust_scale', 'scale']
```

In [136]:
```
1 print(help(preprocessing.minmax_scale))
```

```
Help on function minmax_scale in module sklearn.preprocessing.data:

minmax_scale(X, feature_range=(0, 1), axis=0, copy=True)
    Transforms features by scaling each feature to a given range.

    This estimator scales and translates each feature individually such
    that it is in the given range on the training set, i.e. between
    zero and one.

    The transformation is given by (when ``axis=0``)::

        X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
        X_scaled = X_std * (max - min) + min

    where min, max = feature_range.

    The transformation is calculated as (when ``axis=0``)::

       X_scaled = scale * X + min - X.min(axis=0) * scale
       where scale = (max - min) / (X.max(axis=0) - X.min(axis=0))

    This transformation is often used as an alternative to zero mean,
    unit variance scaling.

    Read more in the :ref:`User Guide <preprocessing_scaler>`.

    .. versionadded:: 0.17
       *minmax_scale* function interface
       to :class:`sklearn.preprocessing.MinMaxScaler`.

    Parameters
    ----------
    X : array-like, shape (n_samples, n_features)
        The data.

    feature_range : tuple (min, max), default=(0, 1)
        Desired range of transformed data.

    axis : int (0 by default)
        axis used to scale along. If 0, independently scale each feature,
        otherwise (if 1) scale each sample.

    copy : boolean, optional, default is True
        Set to False to perform inplace scaling and avoid a copy (if the inpu
t
        is already a numpy array).

    See also
    --------
    MinMaxScaler: Performs scaling to a given range using the``Transformer``
API
        (e.g. as part of a preprocessing :class:`sklearn.pipeline.Pipeline`).

    Notes
```

```
    -----
    For a comparison of the different scalers, transformers, and normalizers,
    see :ref:`examples/preprocessing/plot_all_scaling.py
    <sphx_glr_auto_examples_preprocessing_plot_all_scaling.py>`.
```

None

In [137]:
```
1 data.head()
```

Out[137]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [138]:
```
1 data1 = data[["Sales","Discount","Order_Quantity","Profit","Shipping_Cost","
```

In [139]:
```
1 data1.head()
```

Out[139]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	136.81	0.01	23	-30.51	3.60	0.56
1	42.27	0.01	13	4.56	0.93	0.54
2	4701.69	0.00	26	1148.90	2.50	0.59
3	2337.89	0.09	43	729.34	14.30	0.37
4	4233.15	0.08	35	1219.87	26.30	0.38

In [140]:
```
1 sc = preprocessing.StandardScaler()
```

In [141]:
```
1 sc_t = sc.fit_transform(data1)
```

```
C:\Users\Alekhya\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:645:
DataConversionWarning: Data with input dtype int64, float64 were all converted
to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\Alekhya\Anaconda3\lib\site-packages\sklearn\base.py:464: DataConversio
nWarning: Data with input dtype int64, float64 were all converted to float64 by
StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

In [142]:
```
1 sc_t
```

Out[142]:
```
array([[-0.4572225 , -1.24669977, -0.17760343, -0.17691592, -0.53516446,
         0.35024724],
       [-0.48359469, -1.24669977, -0.86820119, -0.14760744, -0.68983029,
         0.20273374],
       [ 0.81616312, -1.56095642,  0.0295759 ,  0.8087331 , -0.59888446,
         0.5715175 ],
       ...,
       [-0.42758656, -0.93244313,  0.927353  , -0.2106286 , -0.43379172,
        -0.82986078],
       [ 0.58496154, -0.61818648, -0.17760343,  0.32104405,  0.99411568,
         0.79278775],
       [-0.32698506, -1.56095642,  1.47983121, -0.04161388, -0.46217609,
        -0.97737429]])
```

In [143]:
```
1 data1.columns
```

Out[143]:
```
Index(['Sales', 'Discount', 'Order_Quantity', 'Profit', 'Shipping_Cost',
       'Product_Base_Margin'],
      dtype='object')
```

In [145]:
```
1 sc1_data = pd.DataFrame(sc_t,columns = data1.columns)
2 sc1_data.head()
```

Out[145]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	-0.457222	-1.246700	-0.177603	-0.176916	-0.535164	0.350247
1	-0.483595	-1.246700	-0.868201	-0.147607	-0.689830	0.202734
2	0.816163	-1.560956	0.029576	0.808733	-0.598884	0.571517
3	0.156775	1.267353	1.203592	0.458101	0.084657	-1.051131
4	0.685463	0.953097	0.651114	0.868044	0.779785	-0.977374

Robust scaler

In [147]:
```
1 rs = preprocessing.RobustScaler()
```

In [148]:
```
1 rs_t = rs.fit_transform(data1)
```

In [149]:
```
1 rs_t
```

Out[149]:
```
array([[-0.19960731, -0.66666667, -0.12      , -0.11789568, -0.23105706,
         0.19047619],
       [-0.25997286, -0.66666667, -0.52      ,  0.02462764, -0.4808232 ,
         0.0952381 ],
       [ 2.71515364, -0.83333333,  0.        ,  4.67518745, -0.33395697,
         0.33333333],
       ...,
       [-0.13177109, -0.5       ,  0.52      , -0.2818361 , -0.06735267,
        -0.57142857],
       [ 2.18593663, -0.33333333, -0.12      ,  2.30361896,  2.23854069,
         0.47619048],
       [ 0.09850427, -0.83333333,  0.84      ,  0.54006055, -0.1131899 ,
        -0.66666667]])
```

In [151]:
```
1 pd.DataFrame(rs_t,columns=data1.columns).head()
```

Out[151]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Margin
0	-0.199607	-0.666667	-0.12	-0.117896	-0.231057	0.190476
1	-0.259973	-0.666667	-0.52	0.024628	-0.480823	0.095238
2	2.715154	-0.833333	0.00	4.675187	-0.333957	0.333333
3	1.205823	0.666667	0.68	2.970110	0.769878	-0.714286
4	2.415982	0.500000	0.36	4.963607	1.892423	-0.666667

In [ ]:
```
1
```