E-Commerce Shipping Analysis using python

Import the libraries of phthon

In [1]:

```
import pandas as pd
import numpy as np

import seaborn as sb
import matplotlib.pyplot as plt
```

read the csv file

In [2]:

```
d=pd.read_csv("C:/Users/Ajay/Downloads/Train.csv")
```

In [3]:

```
#print the csv file
d
```

Out[3]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

10999 rows × 12 columns

read the top five records

In [4]:

```
d.head()
```

Out[4]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	

read the top seven records

In [39]:

```
d.head(7)
```

Out[39]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
5	6	F	Flight	3	1	
6	7	D	Flight	3	4	

read the last five records

In [5]:

```
d.tail()
```

Out[5]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

to check the column names

In [4]:

```
d.columns
```

Out[4]:

```
Index(['ID', 'Warehouse_block', 'Mode_of_Shipment', 'Customer_care_calls',
       'Customer_rating', 'Cost_of_the_Product', 'Prior_purchases',
       'Product_importance', 'Gender', 'Discount_offered', 'Weight_in_gms',
       'Reached.on.Time_Y.N'],
      dtype='object')
```

basic information of dataset

In [6]:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   10999 non-null  int64 
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64 
 4   Customer_rating      10999 non-null  int64 
 5   Cost_of_the_Product  10999 non-null  int64 
 6   Prior_purchases      10999 non-null  int64 
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64 
 10  Weight_in_gms        10999 non-null  int64 
 11  Reached.on.Time_Y.N  10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

cleaning the data set

In [ ]:

```
# check if the data containing null values
```

In [8]:

```
d.isna()
```

Out[8]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	False	False	False	
4	False	False	False	False	False	
...	...	...	...	...	...	
10994	False	False	False	False	False	
10995	False	False	False	False	False	
10996	False	False	False	False	False	
10997	False	False	False	False	False	
10998	False	False	False	False	False	

10999 rows × 12 columns

In [7]:

```
d.isna().sum()
```

Out[7]:

```
ID                   0
Warehouse_block      0
Mode_of_Shipment     0
Customer_care_calls  0
Customer_rating      0
Cost_of_the_Product  0
Prior_purchases      0
Product_importance   0
Gender               0
Discount_offered     0
Weight_in_gms        0
Reached.on.Time_Y.N  0
dtype: int64
```

statistical information about the dataset

In [8]:

```
d.describe()
```

Out[8]:

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchase
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.00000
mean	5500.00000	4.054459	2.990545	210.196836	3.56759
std	3175.28214	1.141490	1.413603	48.063272	1.52286
min	1.00000	2.000000	1.000000	96.000000	2.00000
25%	2750.50000	3.000000	2.000000	169.000000	3.00000
50%	5500.00000	4.000000	3.000000	214.000000	3.00000
75%	8249.50000	5.000000	4.000000	251.000000	4.00000
max	10999.00000	7.000000	5.000000	310.000000	10.00000

number of Rows and Columns of data is available

In [41]:

```
d.shape
```

Out[41]:

```
(10999, 12)
```

In [7]:

```
shipment_d=pd.DataFrame(d)
d
```

Out[7]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

10999 rows × 12 columns

calculating the detials of flight shipment

In [8]:

```
d_1=shipment_d.loc[shipment_d['Mode_of_Shipment']=="Flight"]
d_1
```

Out[8]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10972	10973	C	Flight	4	1	
10973	10974	F	Flight	5	2	
10974	10975	D	Flight	5	2	
10975	10976	F	Flight	5	2	
10976	10977	A	Flight	3	2	

1777 rows × 12 columns

calculating the product impotance[medium]

In [33]:

```
d_7=shipment_d.loc[shipment_d['Product_importance']=="medium"]
d_7
```

Out[33]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
5	6	F	Flight	3	1	
9	10	B	Flight	3	2	
10	11	C	Flight	3	4	
...	...	...	...	...	...	
10991	10992	F	Ship	5	2	
10992	10993	D	Ship	5	1	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10997	10998	F	Ship	5	2	

4754 rows × 12 columns

In [21]:

```
d_2=d_1.loc[:,'Prior_purchases']
s1=np.sum(d_2)
s1
```

Out[21]:

```
6338
```

calculating the product impotance[low]

In [16]:

```
d_3=d_1.loc[d_1['Product_importance']=='low']
d_3
```

Out[16]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
6	7	D	Flight	3	4	
7	8	F	Flight	4	1	
...	...	...	...	...	...	
10971	10972	B	Flight	5	5	
10972	10973	C	Flight	4	1	
10973	10974	F	Flight	5	2	
10974	10975	D	Flight	5	2	
10975	10976	F	Flight	5	2	

838 rows × 12 columns

calculating the gender[male] in flight

In [23]:

```
d_4=d_1.loc[d_1['Gender']=='M']
d_4
```

Out[23]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
10	11	C	Flight	3	4	
13	14	F	Flight	4	4	
...	...	...	...	...	...	
10969	10970	F	Flight	5	2	
10970	10971	A	Flight	4	4	
10971	10972	B	Flight	5	5	
10975	10976	F	Flight	5	2	
10976	10977	A	Flight	3	2	

915 rows × 12 columns

calculating the gender[Female] in flight

In [25]:

```
d_5=d_1.loc[d_1['Gender']=='F']
d_5
```

Out[25]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
0	1	D	Flight	4	2	
4	5	C	Flight	2	2	
5	6	F	Flight	3	1	
6	7	D	Flight	3	4	
7	8	F	Flight	4	1	
...	...	...	...	...	...	
10966	10967	C	Flight	5	5	
10968	10969	D	Flight	4	4	
10972	10973	C	Flight	4	1	
10973	10974	F	Flight	5	2	
10974	10975	D	Flight	5	2	

862 rows × 12 columns

calculating the product importance according to the mode of shipment

In [24]:

```
d.groupby('Mode_of_Shipment')['Product_importance'].value_counts()
```

Out[24]:

```
Mode_of_Shipment  Product_importance
Flight            low                  838
                  medium               776
                  high                 163
Road              low                  857
                  medium               745
                  high                 158
Ship              low                 3602
                  medium              3233
                  high                 627
Name: Product_importance, dtype: int64
```

calculating the gender according to the warehouse_block

In [34]:

```
d.groupby('Warehouse_block')['Gender'].value_counts()
```

Out[34]:

```
Warehouse_block  Gender
A                F          928
                 M          905
B                M          925
                 F          908
C                F          921
                 M          912
D                F          933
                 M          901
F                F         1855
                 M         1811
Name: Gender, dtype: int64
```

calculating the detials of Warehouse_block[C]

In [20]:

```
d_6=shipment_d.loc[shipment_d['Warehouse_block']=='C']
d_6
```

Out[20]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cos
4	5	C	Flight	2	2	
10	11	C	Flight	3	4	
16	17	C	Flight	3	4	
22	23	C	Ship	2	5	
28	29	C	Ship	2	3	
...	...	...	...	...	...	
10972	10973	C	Flight	4	1	
10978	10979	C	Ship	4	1	
10984	10985	C	Ship	5	1	
10990	10991	C	Ship	5	4	
10996	10997	C	Ship	5	4	

1833 rows × 12 columns

In [ ]:

```
# Query- warehouse_block
```

In [35]:

```
c_1=list(d_3.loc[:,'Warehouse_block'].unique())
c_1.sort()
print(c_1)
```

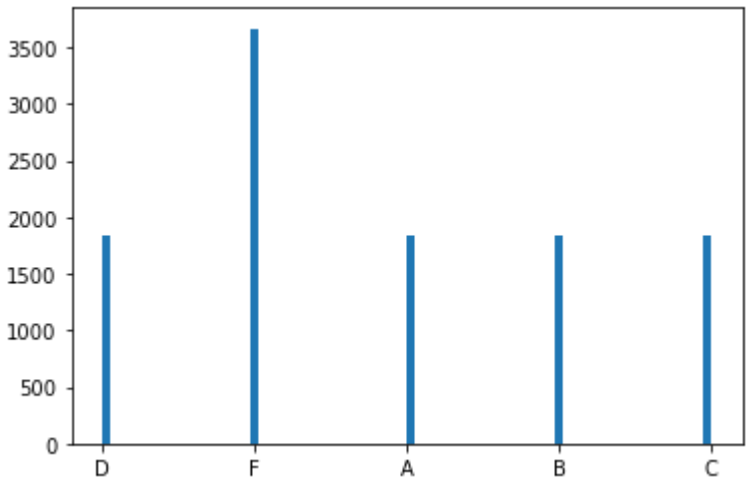
```
['A', 'B', 'C', 'D', 'F']
```

In [29]:

```
plt.hist(d['Warehouse_block'],bins=70)
```

Out[29]:

```
(array([1834.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0., 3666.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0., 1833.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0., 1833.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0., 1833.]),
 array([0.        , 0.05714286, 0.11428571, 0.17142857, 0.22857143,
        0.28571429, 0.34285714, 0.4       , 0.45714286, 0.51428571,
        0.57142857, 0.62857143, 0.68571429, 0.74285714, 0.8       ,
        0.85714286, 0.91428571, 0.97142857, 1.02857143, 1.08571429,
        1.14285714, 1.2       , 1.25714286, 1.31428571, 1.37142857,
        1.42857143, 1.48571429, 1.54285714, 1.6       , 1.65714286,
        1.71428571, 1.77142857, 1.82857143, 1.88571429, 1.94285714,
        2.        , 2.05714286, 2.11428571, 2.17142857, 2.22857143,
        2.28571429, 2.34285714, 2.4       , 2.45714286, 2.51428571,
        2.57142857, 2.62857143, 2.68571429, 2.74285714, 2.8       ,
        2.85714286, 2.91428571, 2.97142857, 3.02857143, 3.08571429,
        3.14285714, 3.2       , 3.25714286, 3.31428571, 3.37142857,
        3.42857143, 3.48571429, 3.54285714, 3.6       , 3.65714286,
        3.71428571, 3.77142857, 3.82857143, 3.88571429, 3.94285714,
        4.        ]),
 <BarContainer object of 70 artists>)
```

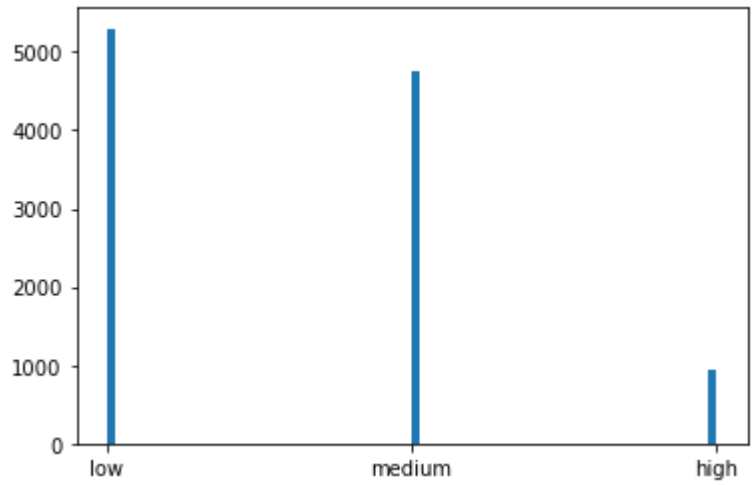


In [30]:

```
plt.hist(d['Product_importance'],bins=70)
```

Out[30]:

```
(array([5297.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0., 4754.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
           0.,    0.,    0.,    0.,    0.,    0.,  948.]),
 array([0.        , 0.02857143, 0.05714286, 0.08571429, 0.11428571,
        0.14285714, 0.17142857, 0.2       , 0.22857143, 0.25714286,
        0.28571429, 0.31428571, 0.34285714, 0.37142857, 0.4       ,
        0.42857143, 0.45714286, 0.48571429, 0.51428571, 0.54285714,
        0.57142857, 0.6       , 0.62857143, 0.65714286, 0.68571429,
        0.71428571, 0.74285714, 0.77142857, 0.8       , 0.82857143,
        0.85714286, 0.88571429, 0.91428571, 0.94285714, 0.97142857,
        1.        , 1.02857143, 1.05714286, 1.08571429, 1.11428571,
        1.14285714, 1.17142857, 1.2       , 1.22857143, 1.25714286,
        1.28571429, 1.31428571, 1.34285714, 1.37142857, 1.4       ,
        1.42857143, 1.45714286, 1.48571429, 1.51428571, 1.54285714,
        1.57142857, 1.6       , 1.62857143, 1.65714286, 1.68571429,
        1.71428571, 1.74285714, 1.77142857, 1.8       , 1.82857143,
        1.85714286, 1.88571429, 1.91428571, 1.94285714, 1.97142857,
        2.        ]),
 <BarContainer object of 70 artists>)
```

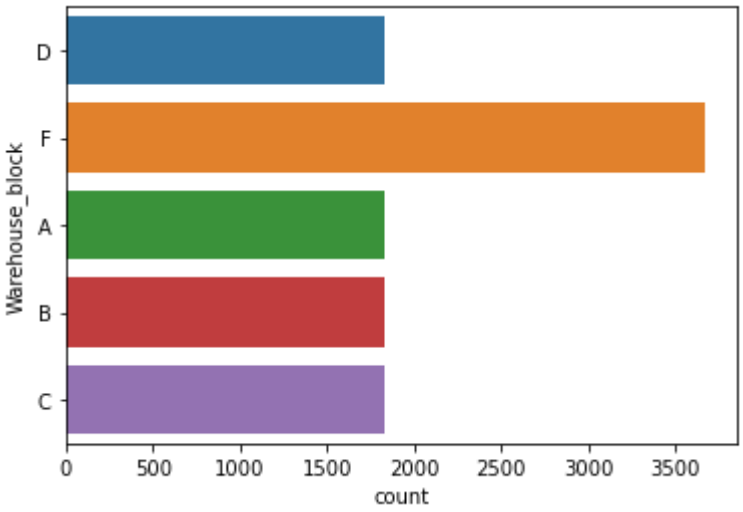


In [31]:

```
sb.countplot(y='Warehouse_block',data=d)
```

Out[31]:

```
<AxesSubplot:xlabel='count', ylabel='Warehouse_block'>
```

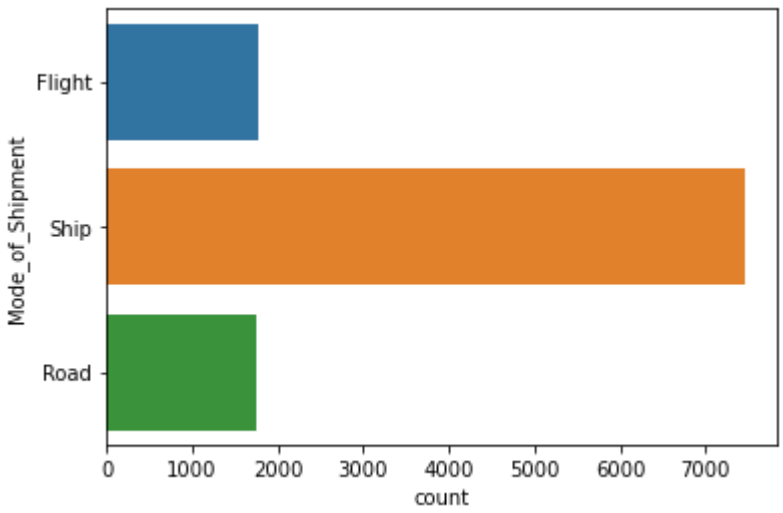


In [32]:

```
sb.countplot(y='Mode_of_Shipment',data=d)
```

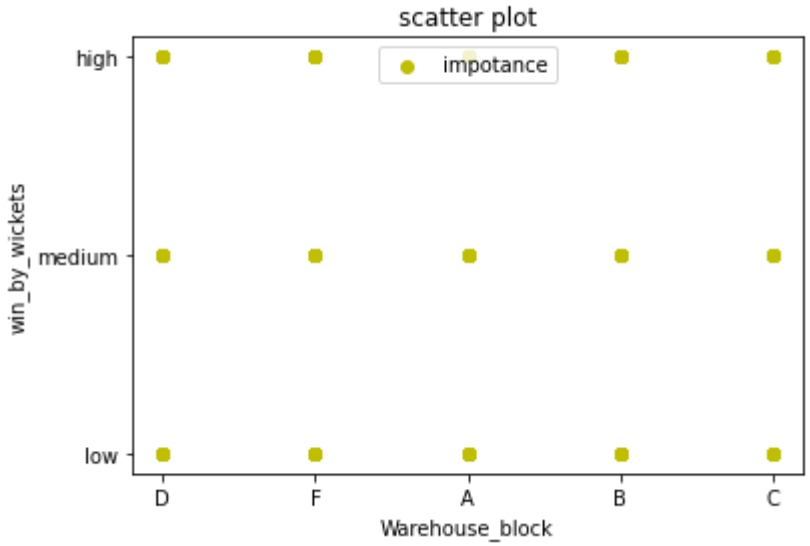
Out[32]:

```
<AxesSubplot:xlabel='count', ylabel='Mode_of_Shipment'>
```



In [38]:

```
x=d['Warehouse_block']
y=d['Product_importance']
plt.scatter(x,y,label="impotance",color="y",linewidth=1)
plt.title("scatter plot")
plt.ylabel('win_by_wickets')
plt.xlabel('Warehouse_block')
plt.legend()
plt.show()
```

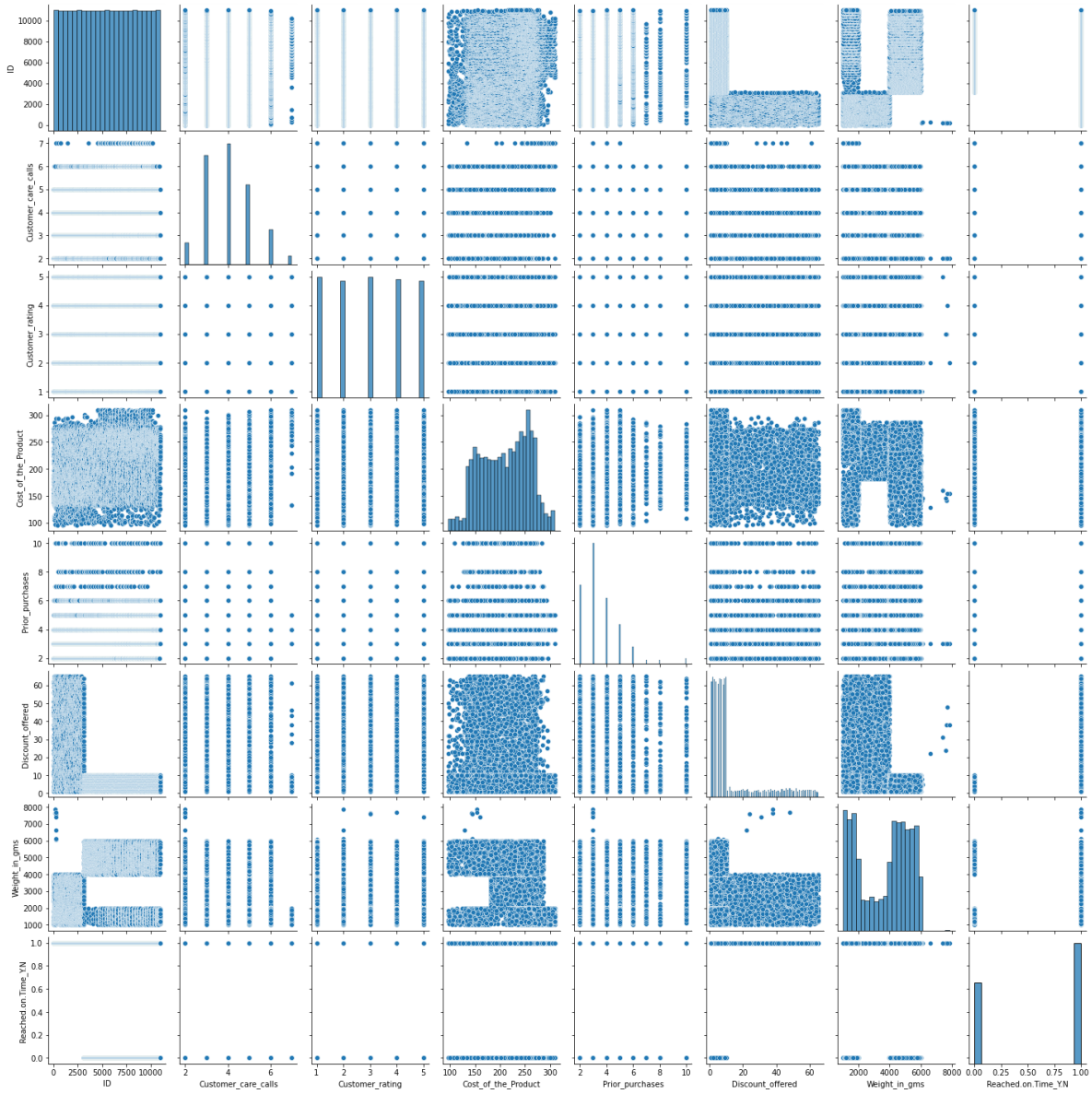


In [36]:

```
sb.pairplot(d)
```

Out[36]:

```
<seaborn.axisgrid.PairGrid at 0x2030ec57b50>
```



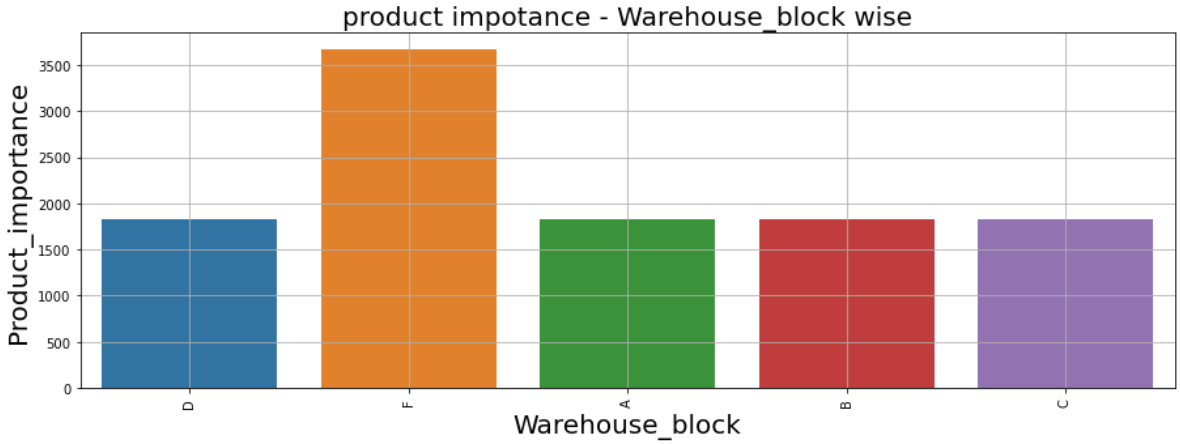
In [5]:

```
from matplotlib import style
```

In [8]:

```
plt.figure(figsize=(15,5))
sb.countplot(d['Warehouse_block'])
plt.xticks(rotation='vertical')
plt.xlabel('Warehouse_block', size=20)
plt.ylabel('Product_importance', size=20)
plt.title('product impotance - Warehouse_block wise', size=20)
plt.grid()
plt.show()
```

```
C:\New folder\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pa
ss the following variable as a keyword arg: x. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]: