Use e-commerce data to analyze and classify customer behavior and implement precision marketing

edit by David Yang 02/15/2023

1. Data preparation

import module

In [4]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

import data， path is './工作/data.csv'， Encoding format is 'utf-8'

In [91]:

```
missing_values = ['-','na','none','null','']
test_data = pd.read_csv('E:/风变/数据分析实训营/all_data.csv',na_values = missing_values,
test_data.head()
```

Out[91]:

	订单号	顾客ID	订单时间	付款金额	
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	18.12	87285b34884572
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	2.00	87285b34884572
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	18.59	87285b34884572
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29	37.77	87285b34884572
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24	37.77	87285b34884572

In [92]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115878 entries, 0 to 115877
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   订单号     115878 non-null  object 
 1   顾客ID    115878 non-null  object 
 2   订单时间    115878 non-null  object 
 3   付款金额    115878 non-null  float64
 4   商品ID    115878 non-null  object 
 5   商品描述    115878 non-null  object 
dtypes: float64(1), object(5)
memory usage: 5.3+ MB
```

Data Cleansing

In [93]:

```
missing_value = ['-','na','none','null','inf']
```

In [94]:

```
test_data.isnull().sum()
```

Out[94]:

```
订单号     0
顾客ID    0
订单时间    0
付款金额    0
商品ID    0
商品描述    0
dtype: int64
```

In [95]:

```
test_data[test_data.duplicated()].tail()
```

Out[95]:

	订单号	顾客ID	订单时间	付款金额	
115714	5020a3db49225f967490d76021c7d13a	5a8b3e70cb6bfdbc353bcb5ae2b4d4eb	2018-01-28 23:36	188.45	3fdb534d
115715	5020a3db49225f967490d76021c7d13a	5a8b3e70cb6bfdbc353bcb5ae2b4d4eb	2018-01-28 23:36	188.45	3fdb534d
115716	5020a3db49225f967490d76021c7d13a	5a8b3e70cb6bfdbc353bcb5ae2b4d4eb	2018-01-28 23:36	188.45	3fdb534d
115737	b144e2ac9863ed27bc59dbe4dd2f8773	49bc0bacf1f213a2d30e240c648ccb01	2017-12-06 14:04	99.70	f83fd2b5
115781	161f105f25baba98c7604aad9b99d9a6	b9dd6c551bfe1ea46e2ca722708df61d	2018-03-14 12:26	170.60	7515ab3

In [96]:

```
test_data1 = test_data.drop_duplicates().reset_index(drop=True)
test_data1[test_data1.duplicated()]
```

Out[96]:

订单号	顾客ID	订单时间	付款金额	商品ID	商品描述

In [97]:

```
test_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104839 entries, 0 to 104838
Data columns (total 6 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   订单号     104839 non-null  object 
 1   顾客ID    104839 non-null  object 
 2   订单时间    104839 non-null  object 
 3   付款金额    104839 non-null  float64
 4   商品ID    104839 non-null  object 
 5   商品描述    104839 non-null  object 
dtypes: float64(1), object(5)
memory usage: 4.8+ MB
```

In [98]:

```
test_data1.tail()
```

Out[98]:

	订单号	顾客ID	订单时间	付款金额	
104834	0b82d0616f1ad8da15cf967b984b4004	986632b40c38f4240caf8608cb01d40d	2018-08-03 21:35	33.69	4a24717
104835	2ef4a11b6e24fdfbb43b92cb5f95edff	ee1cfdc92e449920e25d3ca4ab4da4f6	2018-07-23	84.63	9c313ad

	订单号	顾客ID	订单时间	付款金额	
			18:35		
104836	2ef4a11b6e24fdfbb43b92cb5f95edff	ee1cfdc92e449920e25d3ca4ab4da4f6	2018-07-23 18:35	84.63	eacb104
104837	2c4ada2e75c2ad41dd93cebb5df5f023	363d3a9b2ec5c5426608688ca033292d	2017-01-26 11:09	209.06	6c7a0a34
104838	bede3503afed051733eeb4a84d1adcc5	919570a26efbd068d6a0f66d5c5072a3	2017-09-17 16:51	115.45	8db75af9

In [99]:

```
test_data1['付款金额'].describe()
```

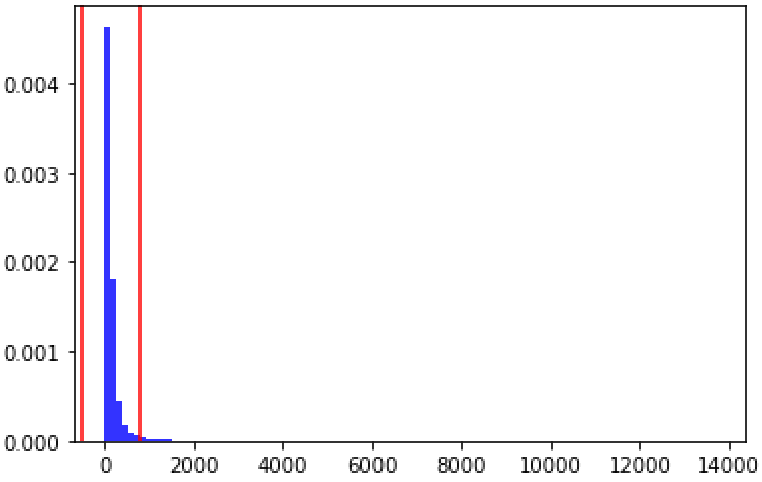
Out[99]:

```
count    104839.000000
mean        158.264636
std         218.993424
min           0.000000
25%          58.370000
50%         102.850000
75%         177.320000
max       13664.080000
Name: 付款金额, dtype: float64
```

use 3*6 method to remove outlier

In [100…

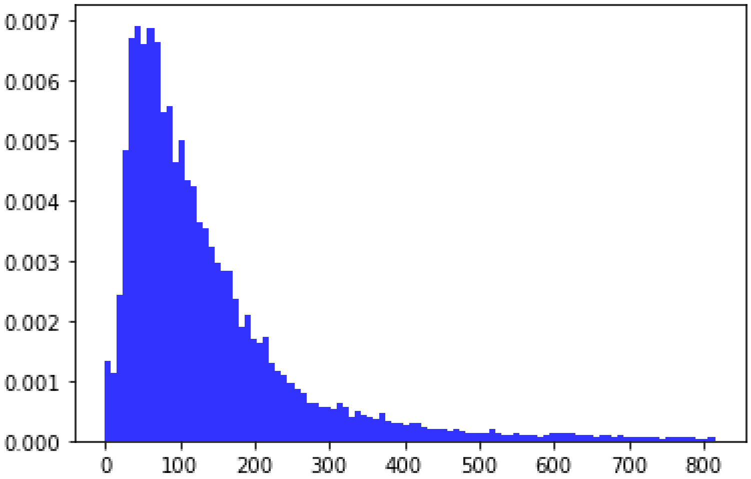
```
plt.hist(test_data1['付款金额'],100,density=True,facecolor='b',alpha=0.8)
m = test_data1['付款金额'].mean()
std = test_data1['付款金额'].std()
plt.axvline(x=m+3*std,color='r')
plt.axvline(x=m-3*std,color='r')
plt.show()
```



In [101…

```
price_sorted = sorted(test_data1["付款金额"])
threshhold = m+3*std
price_normal = []
```

```
price_outlier = []

for price in price_sorted:
    if price<threshhold:
        price_normal.append(price)
    else:
        price_outlier.append(price)

plt.hist(price_normal,100,density=True,facecolor='b',alpha=0.8)
plt.show()
```



In [102…

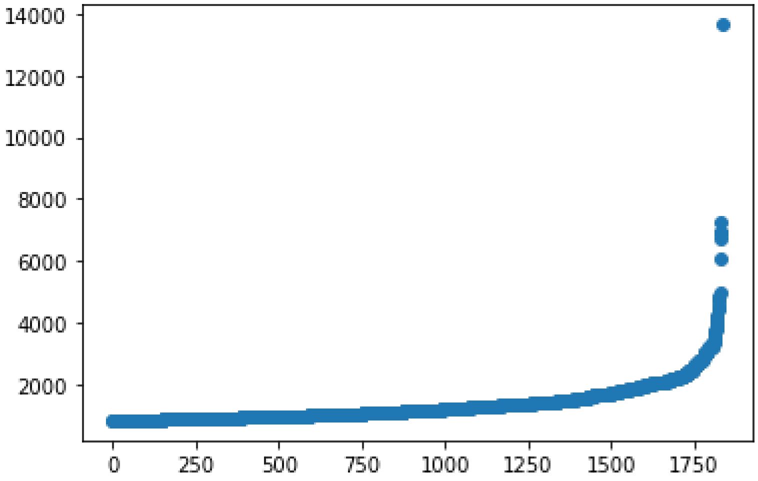
```
len(price_outlier)
```

Out[102…

```
1833
```

In [103…

```
plt.scatter(range(len(price_outlier)),price_outlier)
plt.show()
```



In [104…

```
test_data2 = test_data1[test_data1["付款金额"]<m+3*std].reset_index(drop=True)
```

In [105…

```
test_data2.describe()
```

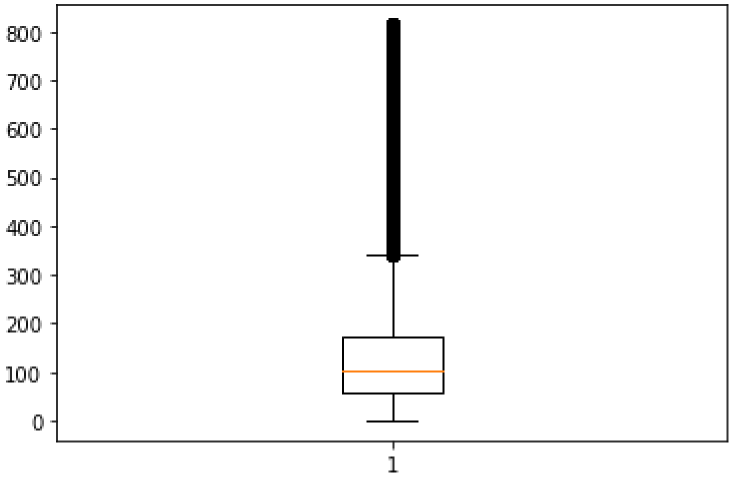
Out[105…

	付款金额
count	103006.000000
mean	137.298398
std	122.271094
min	0.000000
25%	57.770000
50%	100.940000
75%	171.780000
max	814.960000

use 1.5*IQR to analyze based on the 3*6 method to remove outlier

In [106…

```
plt.boxplot(test_data2["付款金额"])
plt.show()
```



In [107…

```
Q1 = test_data2["付款金额"].quantile(0.25)
Q3 = test_data2["付款金额"].quantile(0.75)
IQR = Q3-Q1
IQR
```

Out[107…

```
114.00999999999999
```

In [108…

```
test_data2_normal = test_data2[(test_data2["付款金额"]>Q1-1.5*IQR) & (test_data2["付款金
test_data2_normal.describe()
```

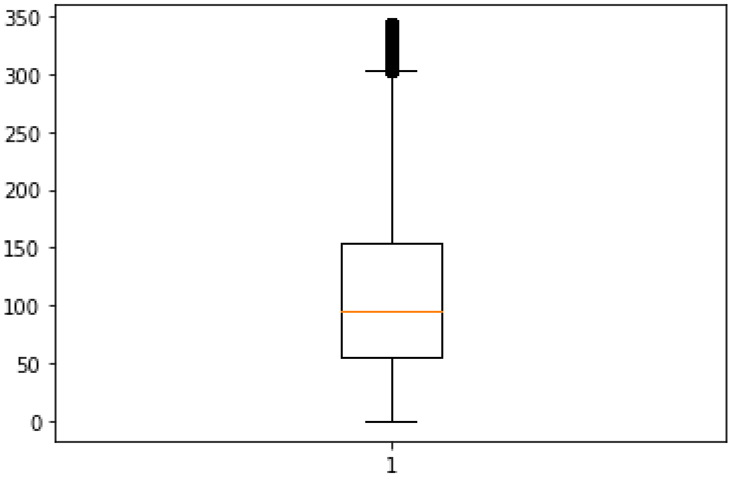
Out[108…

	付款金额
count	96204.000000
mean	112.228934
std	73.282026
min	0.000000

	付款金额
25%	55.240000
50%	94.520000
75%	154.200000
max	342.690000

In [109…

```
plt.boxplot(test_data2_normal["付款金额"])
plt.show()
```



In [110…

```
test_data2_normal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 96204 entries, 0 to 103005
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   订单号     96204 non-null  object
 1   顾客ID    96204 non-null  object
 2   订单时间    96204 non-null  object
 3   付款金额    96204 non-null  float64
 4   商品ID    96204 non-null  object
 5   商品描述    96204 non-null  object
dtypes: float64(1), object(5)
memory usage: 5.1+ MB
```

In [111…

```
test_data2_normal.head()
```

Out[111…

	订单号	顾客ID	订单时间	付款金额	
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	18.12	87285b34884572
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	2.00	87285b34884572

	订单号	顾客ID	订单时间	付款金额	
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56	18.59	87285b34884572
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29	37.77	87285b34884572
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24	37.77	87285b34884572

data wrangling

In [112…

```
test_data2_normal['订单时间'] = test_data2_normal['订单时间'].astype('datetime64')
test_data2_normal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 96204 entries, 0 to 103005
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   订单号     96204 non-null  object
 1   顾客ID    96204 non-null  object
 2   订单时间    96204 non-null  datetime64[ns]
 3   付款金额    96204 non-null  float64
 4   商品ID    96204 non-null  object
 5   商品描述    96204 non-null  object
dtypes: datetime64[ns](1), float64(1), object(4)
memory usage: 5.1+ MB
```

In [113…

```
test_data2_normal['year'] = test_data2_normal['订单时间'].dt.year
test_data2_normal['month'] = test_data2_normal['订单时间'].dt.month
test_data2_normal['day'] = test_data2_normal['订单时间'].dt.day
```

In [114…

```
test_data2_normal.head()
```

Out[114…

	订单号	顾客ID	订单时间	付款金额	
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.12	87285b348845
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	2.00	87285b348845
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.59	87285b348845

	订单号	顾客ID	订单时间	付款金额	
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:00	37.77	87285b348845
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:00	37.77	87285b348845

In [115…

```
test_data2_normal.to_csv('E:/风变/数据分析实训营/cleansing_data.csv',encoding = 'utf-8-si
```

2. Data analyze

In [5]:

```
df = pd.read_csv('E:/风变/数据分析实训营/cleansing_data.csv')
df.columns = ['order_id','cust_id','order_time','order_payment','pro_id','pro_describe'
df.head()
```

Out[5]:

	order_id	cust_id	order_time	order_payment	
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.12	8
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	2.00	8
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.59	8
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:00	37.77	8
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:00	37.77	8

increase weekday

In [6]:

```
df['weekday'] = pd.to_datetime(df['order_time']).dt.weekday
df.head()
```

Out[6]:

	order_id	cust_id	order_time	order_payment	
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.12	8
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	2.00	8
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.59	8

	order_id	cust_id	order_time	order_payment
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:00	37.77
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:00	37.77

remove the data in 2016

In [7]:

```
df.groupby('year')['year'].value_counts()
```

Out[7]:

```
year  year
2016  2016      304
2017  2017    43687
2018  2018    52213
Name: year, dtype: int64
```

In [8]:

```
df = df[(df['year']==2017) | (df['year']==2018)]
df.groupby('year')['year'].value_counts()
```

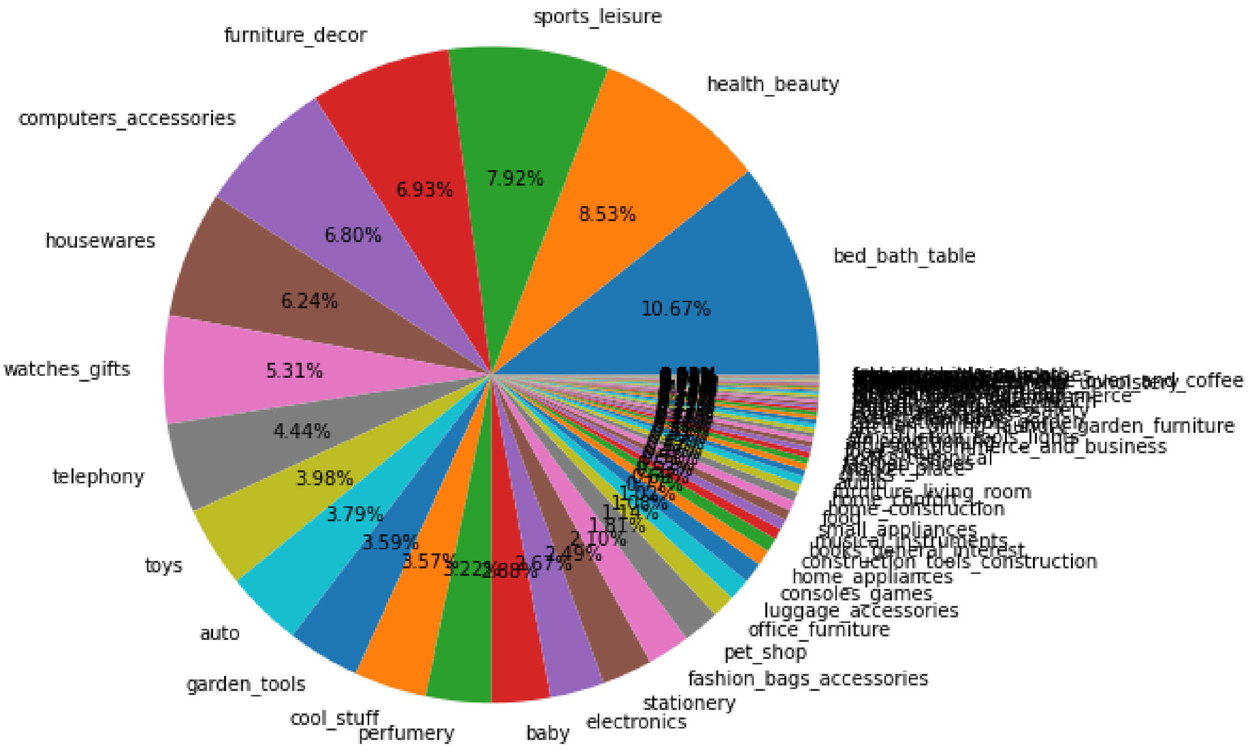
Out[8]:

```
year  year
2017  2017    43687
2018  2018    52213
Name: year, dtype: int64
```

draw a Pie of product deacribe

In [9]:

```
ratio_describe = df['pro_describe'].value_counts() / df['pro_describe'].value_counts().
ratio_describe.plot(kind='pie', autopct='%.2f%%',figsize=(8,8),label='')
plt.show()
```



analyze the sales in different year

In [10]:

```
plt.rcParams['figure.figsize'] = 10,6
df.groupby(['year','month'])['order_payment'].sum()
```

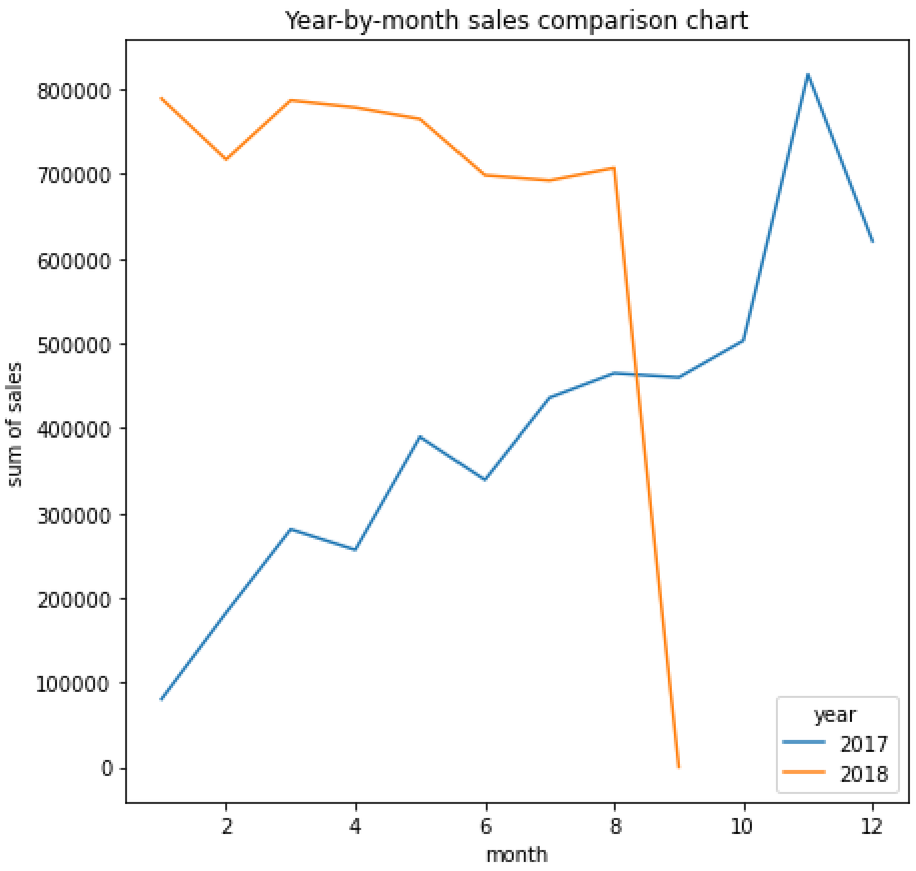
Out[10]:

```
year  month
2017  1         80203.60
      2        182304.65
      3        280771.10
      4        256491.77
      5        389725.59
      6        339110.83
      7        435607.42
      8        464427.76
      9        459472.04
      10       502989.91
      11       817452.04
      12       620276.42
2018  1        788700.86
      2        716850.69
      3        786641.68
      4        778140.09
      5        764595.04
      6        698249.14
      7        691996.42
      8        706881.42
      9           166.46
Name: order_payment, dtype: float64
```

In [11]:

```
year_month_sales_sum = df.groupby(['year','month'])['order_payment'].sum().unstack(leve
Out[11]:  year        2017        2018
          month
              1     80203.60   788700.86
              2    182304.65   716850.69
              3    280771.10   786641.68
              4    256491.77   778140.09
              5    389725.59   764595.04
              6    339110.83   698249.14
              7    435607.42   691996.42
              8    464427.76   706881.42
              9    459472.04      166.46
             10    502989.91         NaN
             11    817452.04         NaN
             12    620276.42         NaN
```

```
In [12]:
year_month_sales_sum.plot(kind='line',figsize=(7,7))
plt.xlabel('month')
plt.ylabel('sum of sales')
plt.title('Year-by-month sales comparison chart')
plt.show()
```



use seaborn to draw the means of weekday

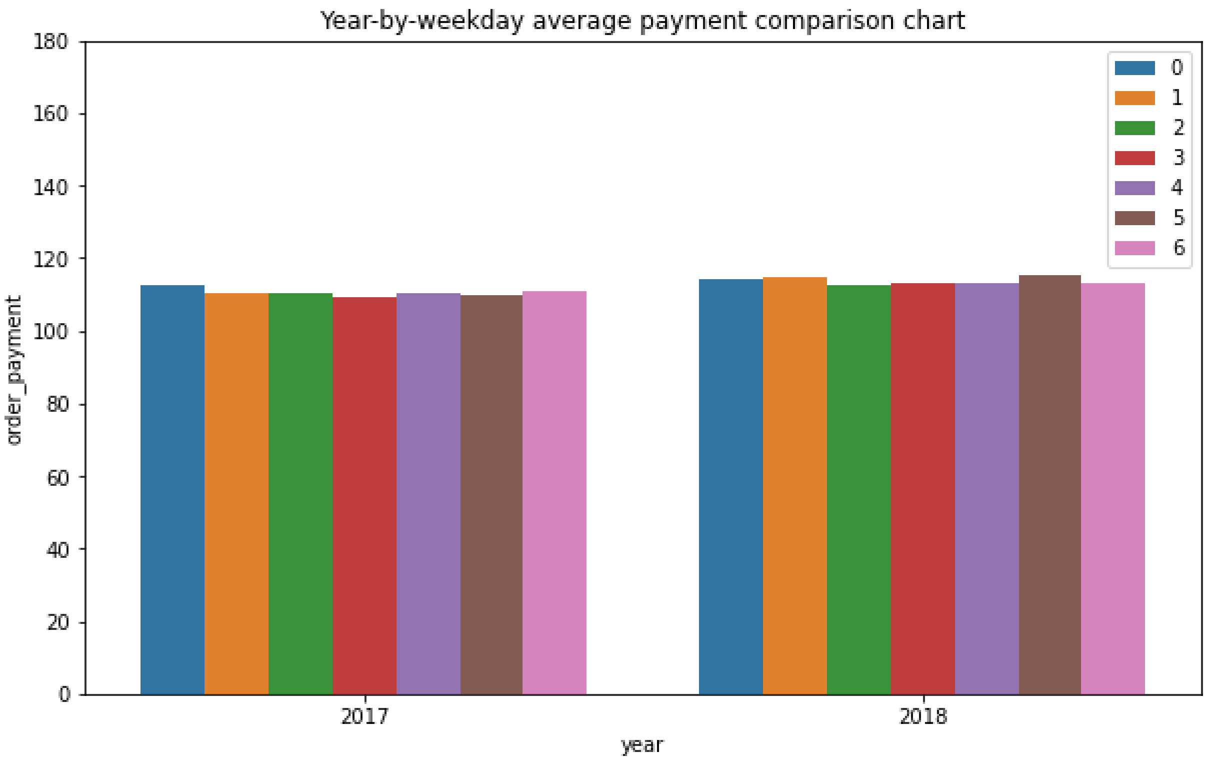
```
In [13]:
```

```python
import seaborn as sns
```

```
In [14]:
```

```python
# to compare the average payment of every weekday in different year
# use hue to class and ci to remove the error interval
sns.barplot(x='year',y='order_payment', data=df, hue='weekday', ci=None)
# extend axis Y
plt.ylim(0,180)
# set the legend
plt.legend(loc='upper right')
plt.title('Year-by-weekday average payment comparison chart')
```

```
Out[14]: Text(0.5, 1.0, 'Year-by-weekday average payment comparison chart')
```



use seaborn to analyze total customers and total payment in every month of 2017

In [18]:

```
# To compare total of customers in every month, need to combine the same cust_id in the
# after group the data in 2017 by month and cust_id, use agg to get the sum of order_pa
# mean of year just for the chart
df_customer = df.groupby(['year','month','cust_id']).agg({'order_payment':'sum'})
df_month_customer = df_customer.reset_index()
df_month_customer.tail()
```

Out[18]:

	year	month	cust_id	order_payment
89222	2018	8	ffb3857a7f2f2945434d57e00d0a97a7	131.38
89223	2018	8	ffb5eaca500a57b7dd52256fcfc82e12	93.63
89224	2018	8	ffe1eab23bff108bf37c973b05d4e9ba	98.65
89225	2018	8	fff212062d600f2e1d53f3c5d4a25138	65.44
89226	2018	9	4b7decb9b58e2569548b8b4c8e20e8d7	166.46

In [19]:

```
# continue to group the data by month and count the total of cumtomer every month
customer_payment = df_month_customer.groupby(['year','month']).agg({'cust_id':'count','
customer_payment
```

Out[19]:

	year	month	cust_id	order_payment
0	2017	1	684	80203.60
1	2017	2	1541	182304.65
2	2017	3	2375	280771.10

	year	month	cust_id	order_payment
3	2017	4	2123	256491.77
4	2017	5	3288	389725.59
5	2017	6	2929	339110.83
6	2017	7	3642	435607.42
7	2017	8	3902	464427.76
8	2017	9	3813	459472.04
9	2017	10	4126	502989.91
10	2017	11	6757	817452.04
11	2017	12	5093	620276.42
12	2018	1	6531	788700.86
13	2018	2	6136	716850.69
14	2018	3	6488	786641.68
15	2018	4	6287	778140.09
16	2018	5	6246	764595.04
17	2018	6	5615	698249.14
18	2018	7	5692	691996.42
19	2018	8	5958	706881.42
20	2018	9	1	166.46

In [20]:

```
customer_payment = customer_payment.rename(columns={'cust_id':'total_cust','order_payme
customer_payment
```

Out[20]:

	year	month	total_cust	total_payment
0	2017	1	684	80203.60
1	2017	2	1541	182304.65
2	2017	3	2375	280771.10
3	2017	4	2123	256491.77
4	2017	5	3288	389725.59
5	2017	6	2929	339110.83
6	2017	7	3642	435607.42
7	2017	8	3902	464427.76
8	2017	9	3813	459472.04
9	2017	10	4126	502989.91
10	2017	11	6757	817452.04
11	2017	12	5093	620276.42

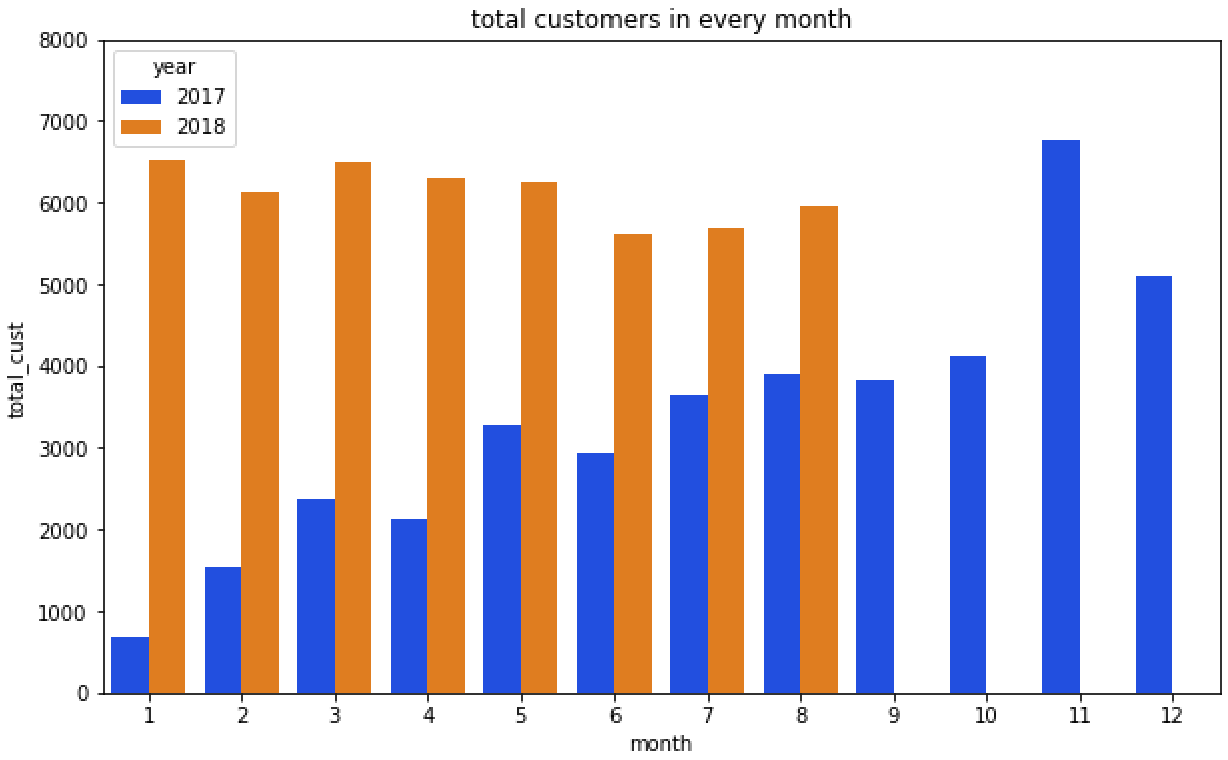
	year	month	total_cust	total_payment
12	2018	1	6531	788700.86
13	2018	2	6136	716850.69
14	2018	3	6488	786641.68
15	2018	4	6287	778140.09
16	2018	5	6246	764595.04
17	2018	6	5615	698249.14
18	2018	7	5692	691996.42
19	2018	8	5958	706881.42
20	2018	9	1	166.46

In [22]:

```
sns.barplot(x='month',y='total_cust', data=customer_payment, hue='year',palette='bright
# extend axis Y
plt.ylim(0,8000)
plt.title('total customers in every month')
```

Out[22]:

```
Text(0.5, 1.0, 'total customers in every month')
```

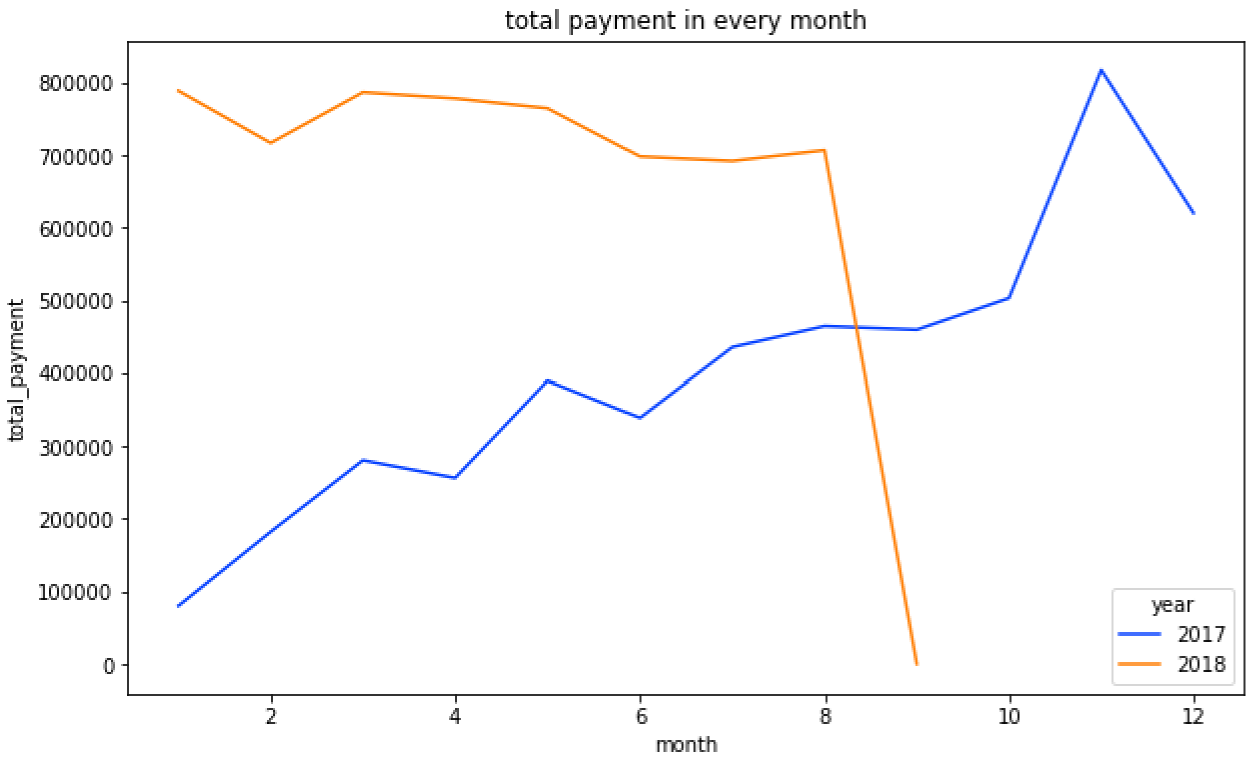


In [25]:

```
sns.lineplot(x='month',y='total_payment', data=customer_payment, hue='year',palette='br
plt.title('total payment in every month')
```

Out[25]:

```
Text(0.5, 1.0, 'total payment in every month')
```



In [132…

```
df.to_csv('E:/风变/数据分析实训营/analyze_data_1.csv',encoding = 'utf-8-sig',index = Fals
```

In [133…

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [134…

```
df = pd.read_csv('E:/风变/数据分析实训营/analyze_data_1.csv',encoding = 'utf-8')
```

In [135…

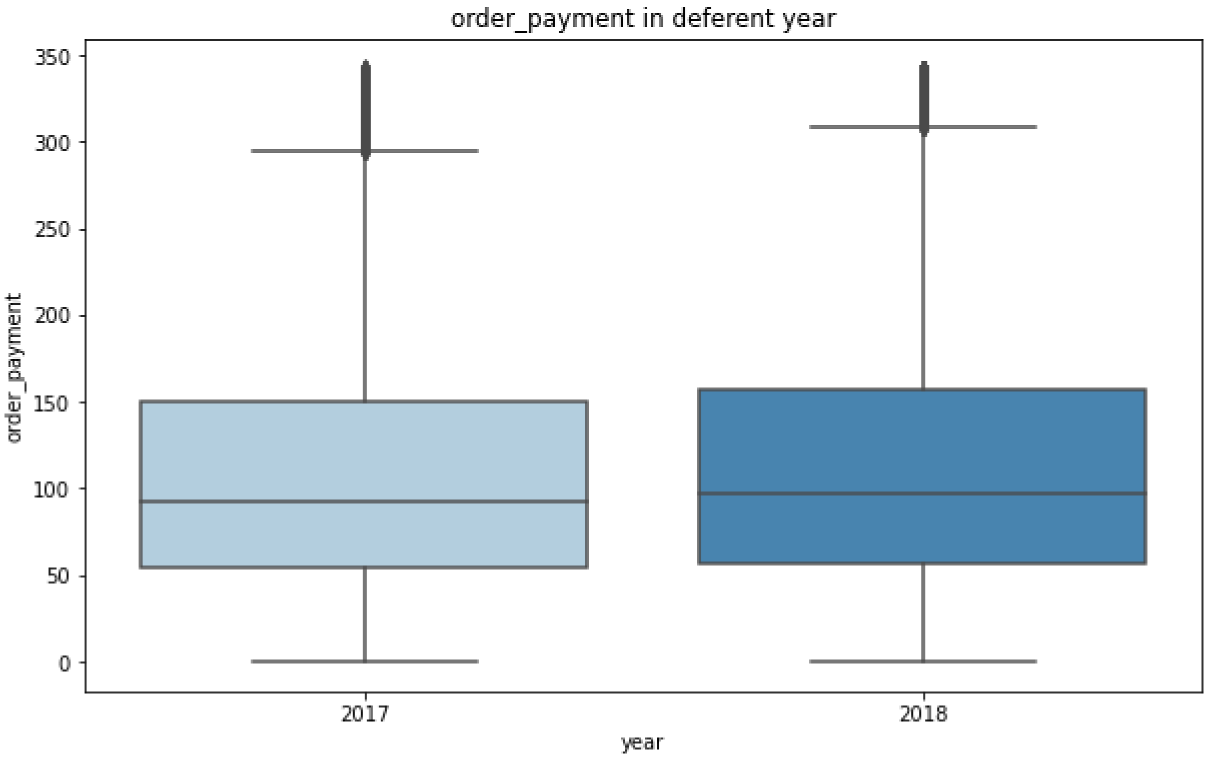
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95900 entries, 0 to 95899
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   order_id       95900 non-null  object 
 1   cust_id        95900 non-null  object 
 2   order_time     95900 non-null  object 
 3   order_payment  95900 non-null  float64
 4   pro_id         95900 non-null  object 
 5   pro_describe   95900 non-null  object 
 6   year           95900 non-null  int64  
 7   month          95900 non-null  int64  
 8   day            95900 non-null  int64  
 9   weekday        95900 non-null  int64  
dtypes: float64(1), int64(4), object(5)
memory usage: 7.3+ MB
```

use seaborn to analyze order_payment in deferent year

In [136…

```
sns.boxplot(x='year',y='order_payment',data=df,palette='Blues')
plt.title('order_payment in deferent year')
plt.show()
```



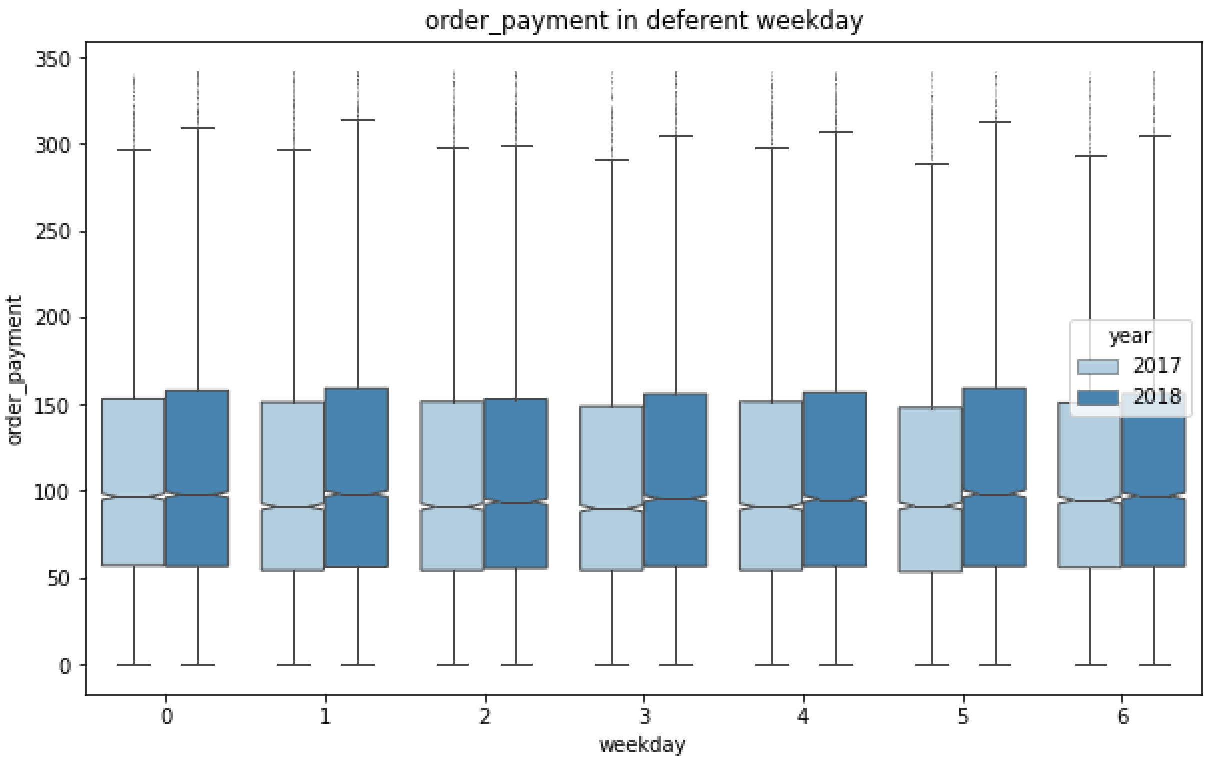
use seaborn to analyze order_payment in diferent weekday

In [137…

```
sns.boxplot(x='weekday',y='order_payment', hue='year',linewidth=1,fliersize=0.05,whis=1
plt.title('order_payment in deferent weekday')
plt.show
```

Out[137…

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



use seaborn to analyze year-by-month total of the customer

In [138…

```
# get the different customer of each month
df_month_customerid = df.groupby(['year','month','cust_id'])['order_payment'].sum().res
df_month_customerid
```

Out[138…

	year	month	cust_id	order_payment
0	2017	1	0040b00970e2139e8c43b647c0da5305	41.93
1	2017	1	0051337a96842850e1ec728dd158f4b3	237.99
2	2017	1	007b7f04a35e02745c23ea706492ca20	77.06
3	2017	1	00f3b3a7cd0b6566435090c7fbda03a2	57.51
4	2017	1	01a0d45a369a4356ac4652584652109a	45.86
...	...	...	...	...
89222	2018	8	ffb3857a7f2f2945434d57e00d0a97a7	131.38
89223	2018	8	ffb5eaca500a57b7dd52256fcfc82e12	93.63
89224	2018	8	ffe1eab23bff108bf37c973b05d4e9ba	98.65
89225	2018	8	fff212062d600f2e1d53f3c5d4a25138	65.44
89226	2018	9	4b7decb9b58e2569548b8b4c8e20e8d7	166.46

89227 rows × 4 columns

In [139…

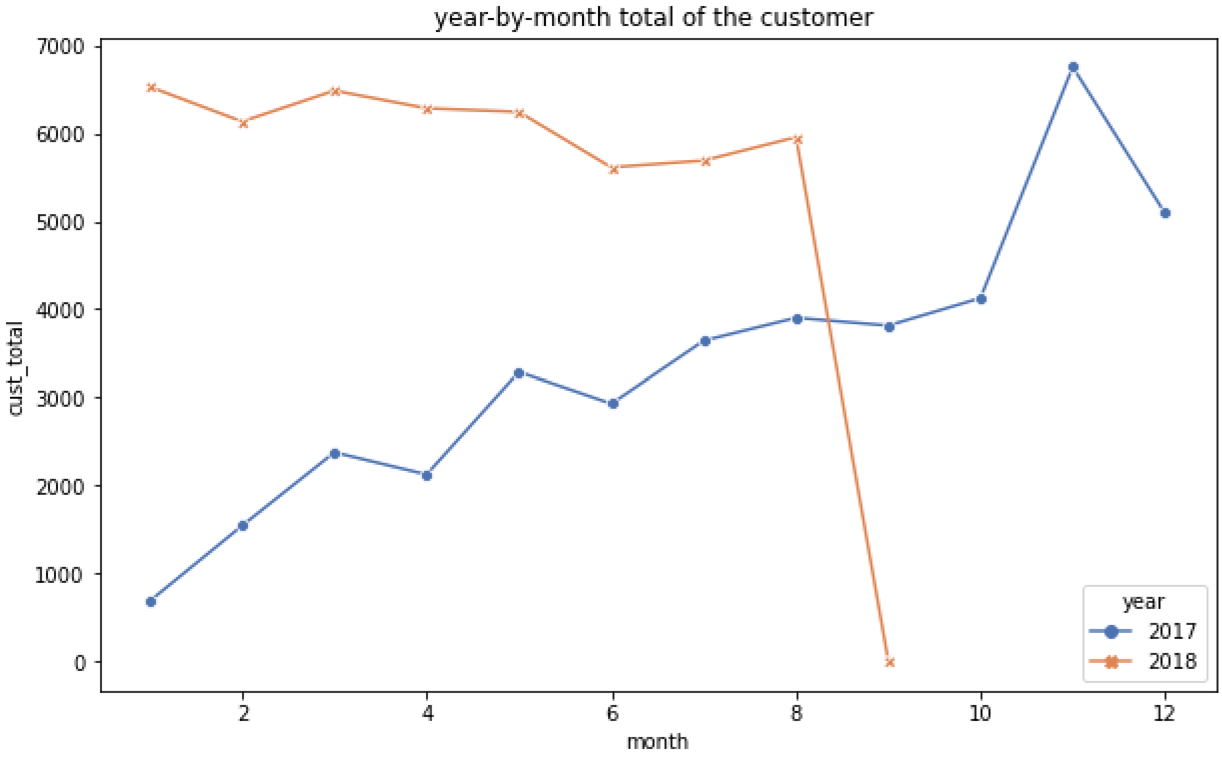
```
df_month_customer_all = df_month_customerid.groupby(['year','month'])['cust_id'].count(
df_month_customer_all
```

Out[139…

	year	month	cust_total
0	2017	1	684
1	2017	2	1541
2	2017	3	2375
3	2017	4	2123
4	2017	5	3288
5	2017	6	2929
6	2017	7	3642
7	2017	8	3902
8	2017	9	3813
9	2017	10	4126
10	2017	11	6757
11	2017	12	5093
12	2018	1	6531
13	2018	2	6136
14	2018	3	6488
15	2018	4	6287
16	2018	5	6246
17	2018	6	5615
18	2018	7	5692
19	2018	8	5958
20	2018	9	1

In [140…

```
sns.lineplot(x='month',y='cust_total',data=df_month_customer_all,hue='year',style='year
plt.title('year-by-month total of the customer ')
plt.show()
```



use seaborn to analyze the distribution of customer according to the order\'s number

In [141…

```
# sum the order number of different customer
df_diff_customer_order = df.groupby('cust_id')['order_id'].count().reset_index().rename
df_diff_customer_order
```

Out[141…

	cust_id	order_total
0	00012a2ce6f8dcda20d059ce98491703	1
1	000161a058600d5901f007fab4c27140	1
2	0001fd6190edaaf884bcaf3d49edf079	1
3	0002414f95344307404f0ace7a26f1d5	1
4	000379cdec625522490c315e70c7a9fb	1
...	...	...
89222	fffcb937e9dd47a13f05ecb8290f4d3e	1
89223	fffecc9f79fd8c764f843e9951b11341	3
89224	fffeda5b6d849fbd39689bb92087f431	1
89225	ffff42319e9b2d713724ae527742af25	1
89226	ffffa3172527f765de70084a7e53aae8	1

89227 rows × 2 columns

In [142…

```
# sum the total of customer
customer_total = len(df_diff_customer_order)
```

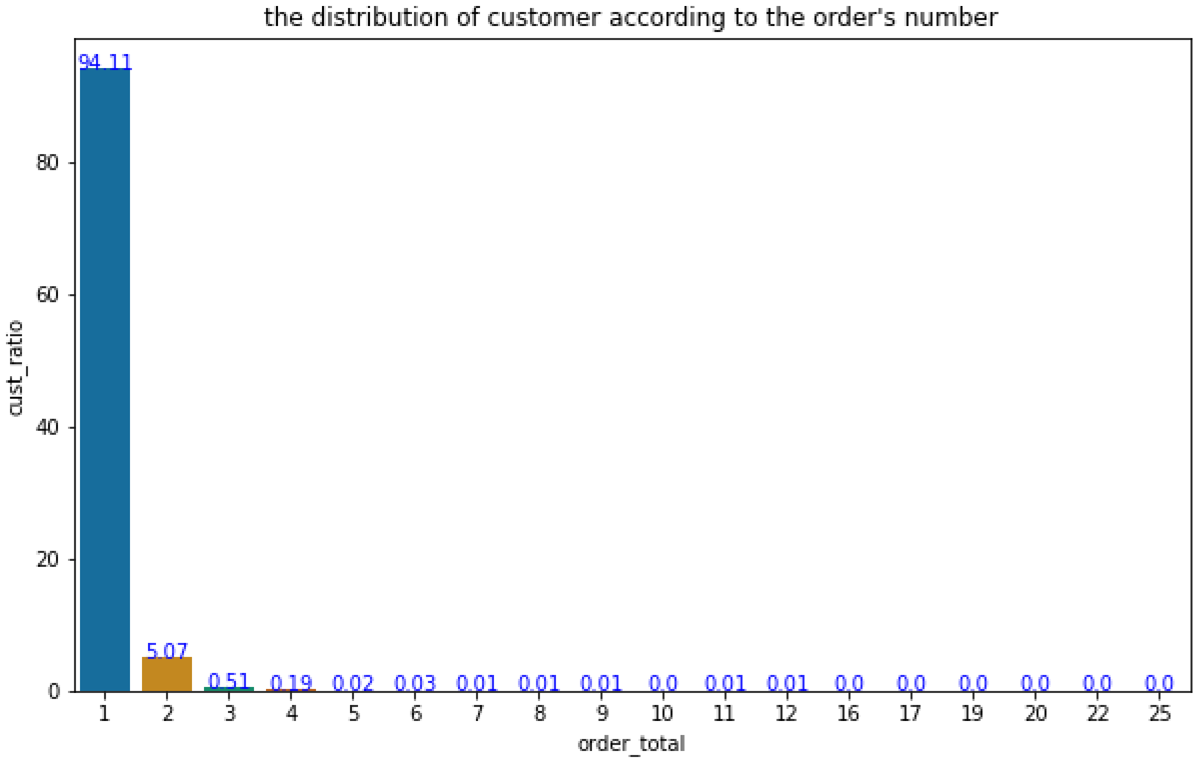
```
# get the total of customers with different order number
df_customer_diff_order = df_diff_customer_order.groupby('order_total')['cust_id'].count
# get the retio of customers with different order number
ratio_cust__diff_order = (df_customer_diff_order / customer_total*100).round(2).to_fram
ratio_cust__diff_order
```

Out[142...

```
    order_total  cust_ratio
 0            1       94.11
 1            2        5.07
 2            3        0.51
 3            4        0.19
 4            5        0.02
 5            6        0.03
 6            7        0.01
 7            8        0.01
 8            9        0.01
 9           10        0.00
10           11        0.01
11           12        0.01
12           16        0.00
13           17        0.00
14           19        0.00
15           20        0.00
16           22        0.00
17           25        0.00
```

In [143...

```
# draw the bar chart
g = sns.barplot(x='order_total',y='cust_ratio',data=ratio_cust__diff_order,palette='col
# add the tags for every bar
for index,row in ratio_cust__diff_order.iterrows():
    g.text(row.name,row['cust_ratio'],round(row['cust_ratio'],2),color='blue',ha='cente

plt.xlabel('order_total')
plt.ylabel('cust_ratio')
plt.title('the distribution of customer according to the order\'s number')
plt.show()
```



3. RFM analyze

In [65]:

```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [66]:

```
df = pd.read_csv('E:/风变/数据分析实训营/analyze_data_1.csv',encoding = 'utf-8')
```

In [67]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95900 entries, 0 to 95899
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   order_id       95900 non-null  object 
 1   cust_id        95900 non-null  object 
 2   order_time     95900 non-null  object 
 3   order_payment  95900 non-null  float64
 4   pro_id         95900 non-null  object 
 5   pro_describe   95900 non-null  object 
 6   year           95900 non-null  int64  
 7   month          95900 non-null  int64  
 8   day            95900 non-null  int64  
 9   weekday        95900 non-null  int64  
dtypes: float64(1), int64(4), object(5)
memory usage: 7.3+ MB
```

In [68]:

```python
df['order_time'] = df['order_time'].astype('datetime64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 95900 entries, 0 to 95899
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   order_id       95900 non-null  object        
 1   cust_id        95900 non-null  object        
 2   order_time     95900 non-null  datetime64[ns]
 3   order_payment  95900 non-null  float64       
 4   pro_id         95900 non-null  object        
 5   pro_describe   95900 non-null  object        
 6   year           95900 non-null  int64         
 7   month          95900 non-null  int64         
 8   day            95900 non-null  int64         
 9   weekday        95900 non-null  int64         
dtypes: datetime64[ns](1), float64(1), int64(4), object(4)
memory usage: 7.3+ MB
```

get value of Recency, Frequency, and Monetary

In [69]:

```python
max(df['order_time'] )
```

Out[69]:

```
Timestamp('2018-09-03 09:06:00')
```

In [70]:

```python
today = '2019-01-01 00:00:00'
df['interval'] = (pd.to_datetime(today)-pd.to_datetime(df['order_time'])).dt.days
df.head()
```

Out[70]:

	order_id	cust_id	order_time	order_payment
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.12
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	2.00

	order_id	cust_id	order_time	order_payment
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:00	18.59
3	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:00	37.77
4	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:00	37.77

In [71]:

```
# get values of R, F, and M
rfm_data = df.groupby('cust_id').agg({'interval':'min','order_payment':'sum','order_id'

rfm_data = rfm_data.rename(columns={'interval':'min_interval','order_payment':'total_pa

rfm_data
```

Out[71]:

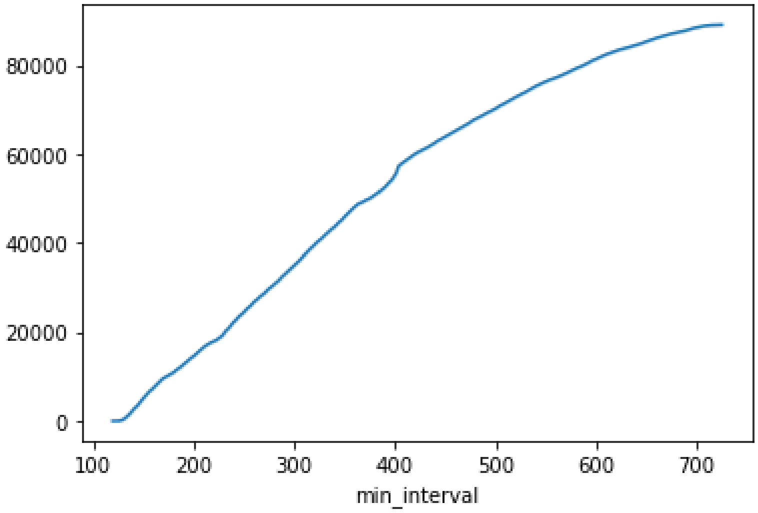
	cust_id	min_interval	total_pay	times
0	00012a2ce6f8dcda20d059ce98491703	412	114.74	1
1	000161a058600d5901f007fab4c27140	533	67.41	1
2	0001fd6190edaaf884bcaf3d49edf079	671	195.42	1
3	0002414f95344307404f0ace7a26f1d5	502	179.35	1
4	000379cdec625522490c315e70c7a9fb	273	107.01	1
...	...	...	...	...
89222	fffcb937e9dd47a13f05ecb8290f4d3e	289	91.91	1
89223	fffecc9f79fd8c764f843e9951b11341	277	81.36	3
89224	fffeda5b6d849fbd39689bb92087f431	223	63.13	1
89225	ffff42319e9b2d713724ae527742af25	201	214.13	1
89226	ffffa3172527f765de70084a7e53aae8	485	45.50	1

89227 rows × 4 columns

get threshold of Recency, Frequency, and Monetary

In [72]:

```
x_r = rfm_data['min_interval'].sort_values()
y_r = rfm_data.index
sns.lineplot(x=x_r,y=y_r,data=rfm_data)
plt.show()
```

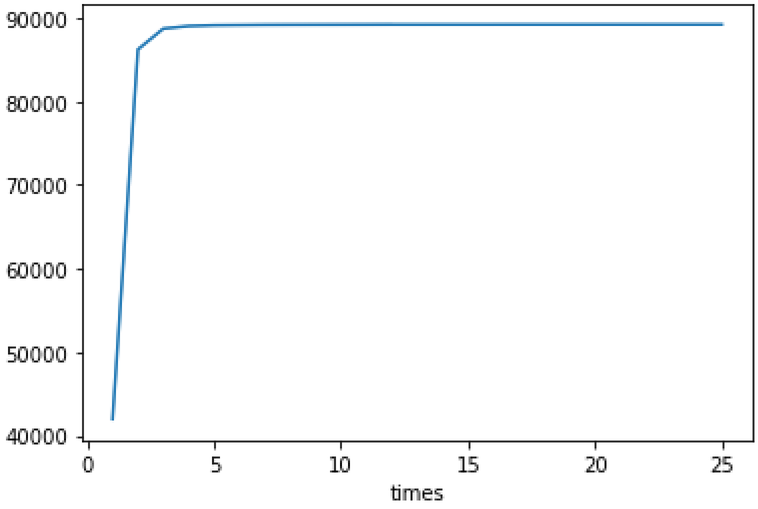


In [73]:

```
def calculate_r(s):
    if s <= 250:
        return 5
    elif s <=370:
        return 4
    elif s <=490:
        return 3
    elif s <=610:
        return 2
    else:
        return 1

rfm_data['R'] = rfm_data['min_interval'].agg(calculate_r)
```

In [74]:

```
x_f = rfm_data['times'].sort_values()
y_f = rfm_data.index
sns.lineplot(x=x_f,y=y_f,data=rfm_data)
plt.show()
```



In [75]:

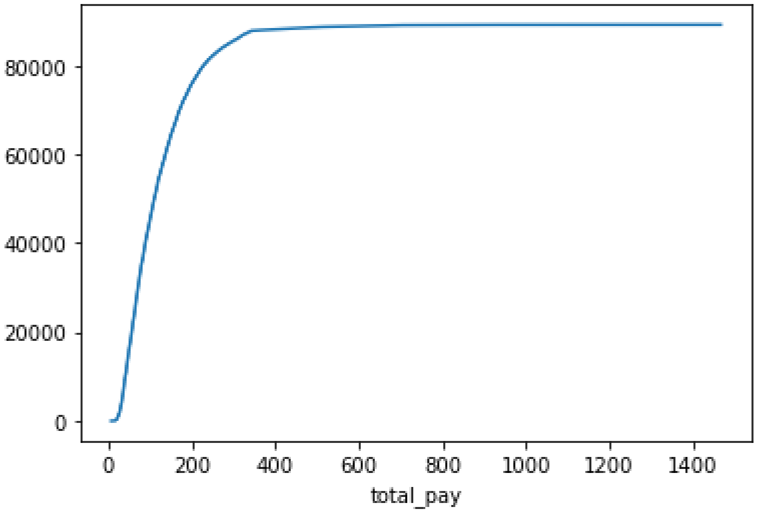
```
def calculate_f(s):
    if s == 0:
        return 1
    elif s == 1:
        return 2
    elif s == 2 :
        return 3
    elif s == 3:
        return 4
    else:
        return 5

rfm_data['F'] = rfm_data['times'].agg(calculate_f)
rfm_data.head()
```

Out[75]:

	cust_id	min_interval	total_pay	times	R	F
0	00012a2ce6f8dcda20d059ce98491703	412	114.74	1	3	2
1	000161a058600d5901f007fab4c27140	533	67.41	1	2	2
2	0001fd6190edaaf884bcaf3d49edf079	671	195.42	1	1	2
3	0002414f95344307404f0ace7a26f1d5	502	179.35	1	2	2
4	000379cdec625522490c315e70c7a9fb	273	107.01	1	4	2

In [76]:

```
x_m = rfm_data['total_pay'].sort_values()
y_m = rfm_data.index
sns.lineplot(x=x_m,y=y_m,data=rfm_data)
plt.show()
```



In [77]:

```
def calculate_m(s):
    if s >= 300:
        return 5
    elif s >= 225:
        return 4
    elif s >= 150 :
        return 3
    elif s >= 75:
        return 2
    else:
        return 1
```

```
rfm_data['M'] = rfm_data['total_pay'].agg(calculate_m)
```

Out[77]:

	cust_id	min_interval	total_pay	times	R	F	M
0	00012a2ce6f8dcda20d059ce98491703	412	114.74	1	3	2	2
1	000161a058600d5901f007fab4c27140	533	67.41	1	2	2	1
2	0001fd6190edaaf884bcaf3d49edf079	671	195.42	1	1	2	3
3	0002414f95344307404f0ace7a26f1d5	502	179.35	1	2	2	3
4	000379cdec625522490c315e70c7a9fb	273	107.01	1	4	2	2
...	...	...	...	...	...	...	...
89222	fffcb937e9dd47a13f05ecb8290f4d3e	289	91.91	1	4	2	2
89223	fffecc9f79fd8c764f843e9951b11341	277	81.36	3	4	4	2
89224	fffeda5b6d849fbd39689bb92087f431	223	63.13	1	5	2	1
89225	ffff42319e9b2d713724ae527742af25	201	214.13	1	5	2	3
89226	ffffa3172527f765de70084a7e53aae8	485	45.50	1	3	2	1

89227 rows × 7 columns

In [78]:

```
r_avg = round(rfm_data['R'].mean(),2)
f_avg = round(rfm_data['F'].mean(),2)
m_avg = round(rfm_data['M'].mean(),2)
print('the average of R is {},the average of R is {},the average of R is {}.'.format(r_
```

```
the average of R is 3.54,the average of R is 2.07,the average of R is 2.06.
```

Get customer's evaluation value of R, F, and M

In [79]:

```
rfm_data['R'] = (rfm_data['R'] > r_avg) * 1
rfm_data['F'] = (rfm_data['F'] > f_avg) * 1
rfm_data['M'] = (rfm_data['M'] > m_avg) * 1

rfm_data.head()
```

Out[79]:

	cust_id	min_interval	total_pay	times	R	F	M
0	00012a2ce6f8dcda20d059ce98491703	412	114.74	1	0	0	0
1	000161a058600d5901f007fab4c27140	533	67.41	1	0	0	0
2	0001fd6190edaaf884bcaf3d49edf079	671	195.42	1	0	0	1
3	0002414f95344307404f0ace7a26f1d5	502	179.35	1	0	0	1
4	000379cdec625522490c315e70c7a9fb	273	107.01	1	1	0	0

In [80]:

```
rfm_data_score = rfm_data['R'].astype(str) + rfm_data['F'].astype(str) + rfm_data['M'].

trans_label = {
                '111':'Important value customers','101':'Important development customer
                '011':'Important to keep customers','001':'Important to retain customer
                '110':'General value customers','100':'General development customers',
                '010':'General to keep customers','000':'General to retain customers',
              }

rfm_data['Customer Style'] = rfm_data_score.replace(trans_label)

rfm_data
```

Out[80]:

	cust_id	min_interval	total_pay	times	R	F	M	Customer Style
0	00012a2ce6f8dcda20d059ce98491703	412	114.74	1	0	0	0	General to retain customers
1	000161a058600d5901f007fab4c27140	533	67.41	1	0	0	0	General to retain customers
2	0001fd6190edaaf884bcaf3d49edf079	671	195.42	1	0	0	1	Important to retain customers
3	0002414f95344307404f0ace7a26f1d5	502	179.35	1	0	0	1	Important to retain customers
4	000379cdec625522490c315e70c7a9fb	273	107.01	1	1	0	0	General development customers
...	...	...	...	...	...	...	...	...
89222	fffcb937e9dd47a13f05ecb8290f4d3e	289	91.91	1	1	0	0	General development customers
89223	fffecc9f79fd8c764f843e9951b11341	277	81.36	3	1	1	0	General value customers
89224	fffeda5b6d849fbd39689bb92087f431	223	63.13	1	1	0	0	General development customers
89225	ffff42319e9b2d713724ae527742af25	201	214.13	1	1	0	1	Important development customers
89226	ffffa3172527f765de70084a7e53aae8	485	45.50	1	0	0	0	General to retain customers

89227 rows × 8 columns

Comparison of various customers

In [82]:

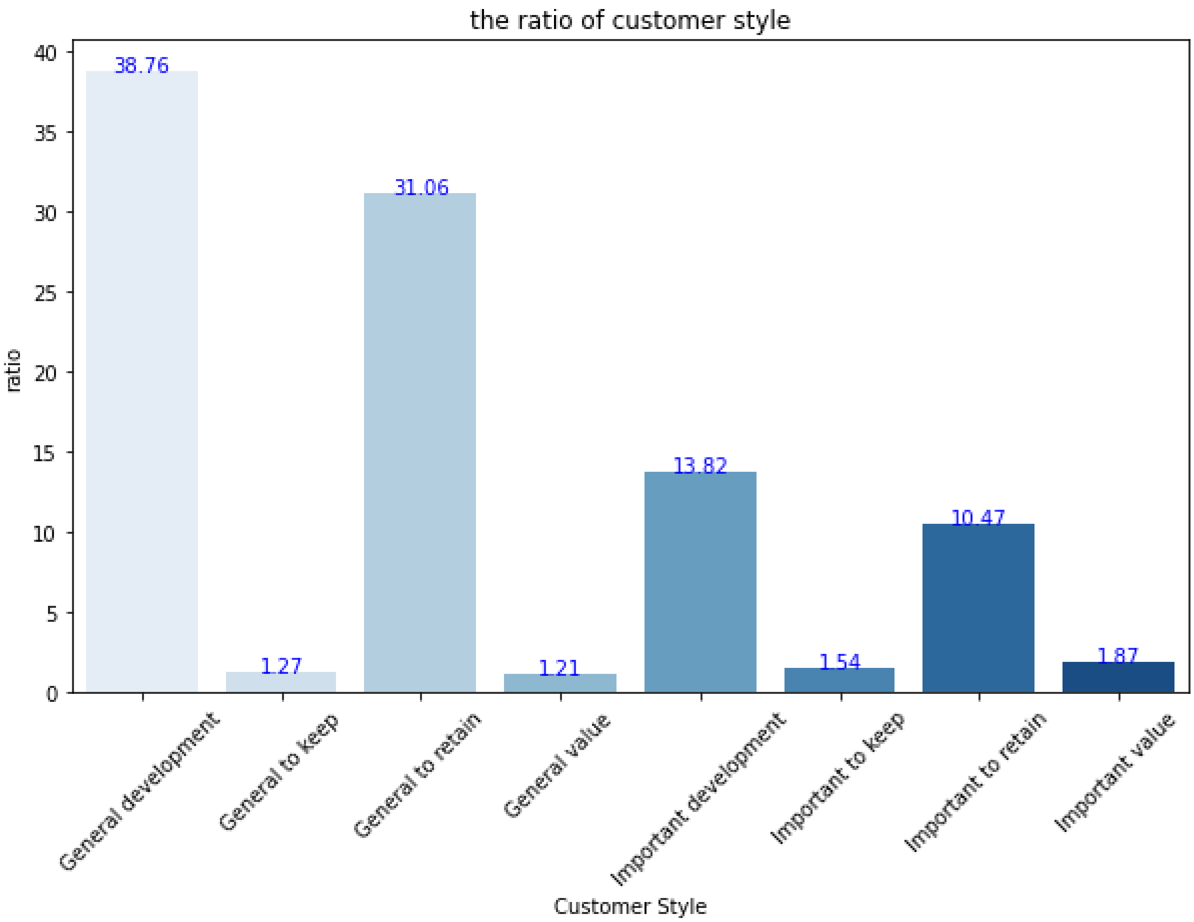
```
rfm_result = rfm_data.groupby('Customer Style')['cust_id'].count().reset_index().rename

rfm_result['ratio'] = (rfm_result['cust_num']/len(rfm_data)*100).round(2)

rfm_result
```

Out[82]:

	Customer Style	cust_num	ratio
0	General development customers	34580	38.76
1	General to keep customers	1137	1.27
2	General to retain customers	27717	31.06
3	General value customers	1077	1.21
4	Important development customers	12330	13.82
5	Important to keep customers	1370	1.54
6	Important to retain customers	9346	10.47
7	Important value customers	1670	1.87

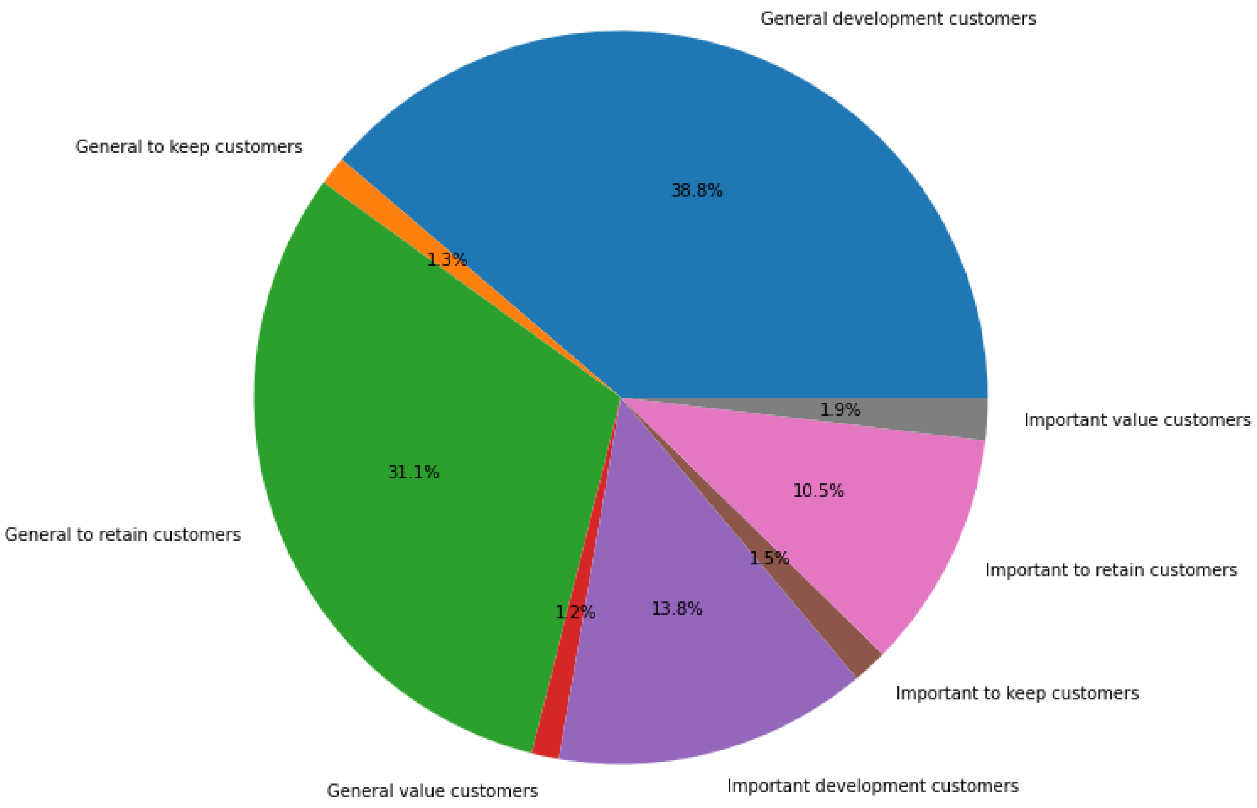
In [160…

```
# draw the bar chart
g = sns.barplot(x='Customer Style',y='ratio',data=rfm_result,palette='Blues')

g.set_xticklabels(labels=['General development','General to keep','General to retain','
# add the tags for every bar
for index,row in rfm_result.iterrows():
    g.text(row.name,row['ratio'],round(row['ratio'],2),color='blue',ha='center')

plt.xlabel('Customer Style')
plt.ylabel('ratio')

plt.title('the ratio of customer style')
plt.show()
```



In [161…

```
plt.figure(figsize=(15,10))

#rfm_result_pie = rfm_result.groupby('Customer Style')

plt.pie(rfm_result['cust_num'],labels=rfm_result['Customer Style'],autopct='%0.1f%%')
plt.show()
```



4. Conclusions and Suggestions

Conclusions

1. Categories that account for more than 5% of sales are are ranked from high to low： |category | ratio of sales |------------------|--------| |bed_bath_table| 10.67%| |health_beauty| 8.53%| |sports_leisure| 7.92%| |furniture_decor| 6.93%| |computers_acc3ssories| 6.80%| |housewares| 6.24%| |watches_gifts| 5.31%|

2. Generally speaking, sales rose rapidly in 2017 and declined slowly in 2018. Sales in the second half of 2017 rose steadily

3. In both 2017 and 2018, the sales in both 2017 and 2018 were compared with the previous month, rising in March, falling in April, rising in May, and falling in June. Sales in November 2017 showed significant positive fluctuations

4. From January to March in 2017, monthly new customers increased by 800 month-on-month, but from January to March in 2018, the number of new customers dropped significantly month-on-month.

5. In January 2018, the total number of customers increased by 8 times compared with that in January 2017, but this growth rate has not been maintained since then

6. There is not much difference in the order payment distribution between in 2017 and in 2018, and there is not much difference in the daily average order payment by week.

7. According to the results of RFM analysis: |Customer Category | Customer Percentage |--------------------------------|----------| |General development customers|38.76%| |General to retain customers|31.06%| |Important development customers| 13.82%| |Important to retain customers|10.47%| |Important value customers|1.87%| |Important to keep customers|1.54%| |General to keep customers|1.27%| |General value customers|1.21%| The proportion of important value customers is 1.87%. It is slightly lower than that of peers.

Cause Analysis and Suggestions

1. The company has obvious brand advantages in household products.
2. In 2017, new customer attracting marketing played an obvious role, especially the promotional activities on November 11 greatly boosted the annual sales.
3. According to the repurchase rate of the same customer, the promotional effect on repeated purchases by new customers in 2018 is not obvious. Of course, it is also related to the fact that the main products sold in 2017 are durable household products, and it is necessary to increase the promotion of other fast-moving consumer goods.
4. Continue to improve the VIP service for important value customers. The operation department strengthens the launch of regular discount activities for important development customers. The customer service department strengthens return visits and increases the repurchase rate for important customers. Retain customers, understand the main reasons through user surveys, and find ways to retain them.
5. Increase the repurchase rate of ordinary value customers and ordinary development customers through coupons, SMS, etc.The technical department enhances the competitiveness of the platform, enhances the competition of products, presents coupons to general retaining customers and collects customer evaluations on the platform.
6. Purchasing department and warehouse management should continue to improve customer response speed.