In [2]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
```

In [3]:
```
user = pd.read_csv("/Users/fennyw/Desktop/tianchi_fresh_comp_train_user.csv",dtype=str)
#把数据转换成字符串类型
```

In [4]:
```
user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15463110 entries, 0 to 15463109
Data columns (total 6 columns):
 #   Column         Dtype 
---  ------         ----- 
 0   user_id        object
 1   item_id        object
 2   behavior_type  object
 3   user_geohash   object
 4   item_category  object
 5   time           object
dtypes: object(6)
memory usage: 707.8+ MB
```

In [6]:
```
user.tail()
```

Out[6]:

	user_id	item_id	behavior_type	user_geohash	item_category	time
15463105	65341491	259008790	1	NaN	13164	2014-12-03 12
15463106	65341491	336404938	1	NaN	13164	2014-12-03 12
15463107	65341491	52142024	1	95qhbsu	5201	2014-12-10 22
15463108	65341491	250557965	1	NaN	13164	2014-12-03 12
15463109	65341491	300315408	1	NaN	1838	2014-11-29 08

In [160]:
```
user.isnull().sum()
```

Out[160]:
```
user_id               0
item_id               0
behavior_type         0
user_geohash    8207386
item_category         0
time                  0
dtype: int64
```

In [161]:
```
user["date"] = user["time"].str[:10]#日期
user["hour"] = user["time"].str[11:]#时间
```

In [162]:
```
user.dtypes
```

Out[162]:
```
user_id          object
item_id          object
behavior_type    object
user_geohash     object
item_category    object
time             object
date             object
hour             object
dtype: object
```

In [163]:
```
#转换日期类型
user["time"] = pd.to_datetime(user["time"])
user["date"] = pd.to_datetime(user["date"])
user["hour"] = user["hour"].astype(int)
```

In [164]:
```
user.dtypes
```

Out[164]:
```
user_id                  object
item_id                  object
behavior_type            object
user_geohash             object
item_category            object
time             datetime64[ns]
date             datetime64[ns]
hour                      int64
dtype: object
```

In [165]:
```
#查看异常值情况
user.sort_values(by="time",ascending=True,inplace=True)
```

In [166]:
```
user.head()
```

Out[166]:

	user_id	item_id	behavior_type	user_geohash	item_category	time	dat
6001596	111623089	19437313	1	NaN	5232	2014-11-18	2014-11-1
15041208	58156776	389968914	1	NaN	5399	2014-11-18	2014-11-1
15041205	58156776	269275438	1	NaN	5399	2014-11-18	2014-11-1
13178633	24236959	140503813	4	NaN	1810	2014-11-18	2014-11-1
13178631	24236959	227677489	1	NaN	2551	2014-11-18	2014-11-1

In [167]:
```
#重置索引，并删除原来索引
user.reset_index(drop=True,inplace=True)
```

构建模型

In [168]:
```
#日访问量
pv_daily = user.groupby("date").count()["user_id"]
#日访客量
uv_daily = user.groupby("date")["user_id"].apply(lambda x:x.drop_du
plicates().count())
```

In [169]:
```
#按列合并日访问量和日访客量
pv_uv_daily = pd.concat([pv_daily,uv_daily],axis=1)
pv_uv_daily.columns = ["pv","uv"]
```

In [170]:
```
pv_uv_daily.head()
```

Out[170]:

	pv	uv
date		
2014-11-18	434005	12534
2014-11-19	442531	12597
2014-11-20	429380	12525
2014-11-21	408187	12298
2014-11-22	426698	12175

In [171]:
```
#查看日访问量和日访客量的相关系数
pv_uv_daily.corr(method="pearson")
```

Out[171]:

	pv	uv
pv	1.000000	0.961899
uv	0.961899	1.000000

In [172]:
```
pv_uv_daily.corr(method="spearman")
```

Out[172]:

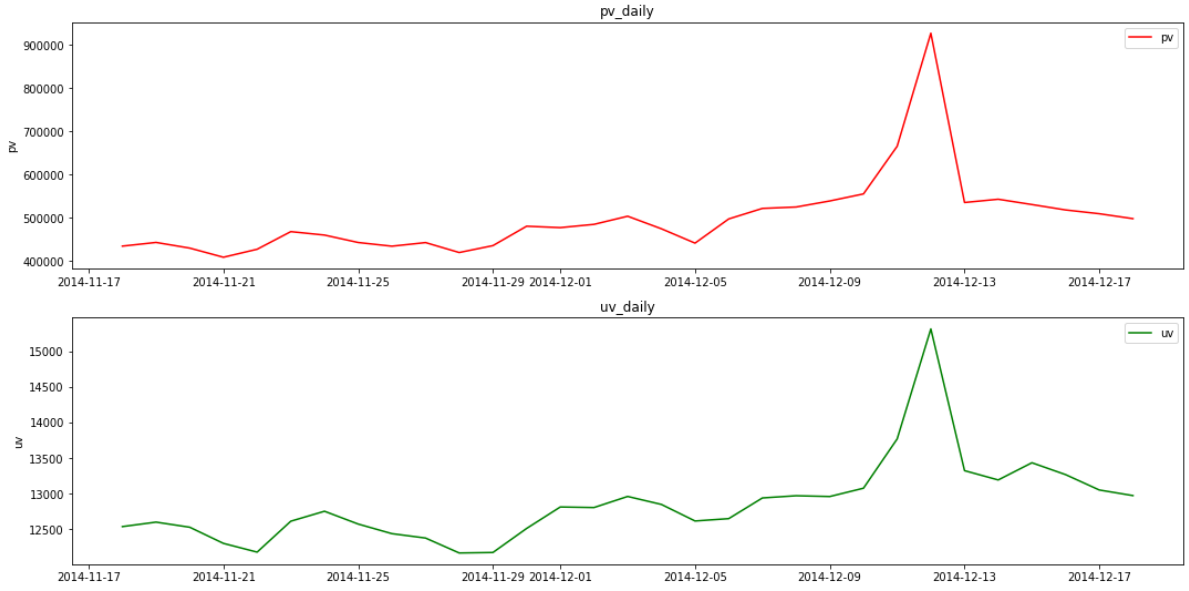
	pv	uv
pv	1.000000	0.921371
uv	0.921371	1.000000

In [173]:
```
plt.figure(figsize=(18,9))

plt.subplot(211)
plt.plot(pv_daily,color="red",label="pv")
plt.ylabel("pv")
plt.title("pv_daily")
plt.legend()

plt.subplot(212)
plt.plot(uv_daily,color="green",label="uv")
plt.ylabel("uv")
plt.title("uv_daily")
plt.legend()

plt.show()
```



由图可知，在双十二期间访问量和访客量达到峰值

In [174]:
```
#每小时访问量、每小时访客量
pv_hour = user.groupby("hour").count()["user_id"]
uv_hour = user.groupby("hour")["user_id"].apply(lambda x:x.drop_dup
licates().count())

pv_uv_hour = pd.concat([pv_hour,uv_hour],axis=1)
pv_uv_hour.columns = ["pv","uv"]
```

In [175]:
```
pv_uv_hour.head()
```

Out[175]:

hour	pv	uv
0	482658	11317
1	244723	7272
2	141102	4828
3	91814	3725
4	72813	3314

In [176]:
```
#查看每小时访问量和每小时访客量的相关系数
pv_uv_hour.corr("pearson")
```

Out[176]:

	pv	uv
pv	1.00000	0.85726
uv	0.85726	1.00000

In [177]:
```
pv_uv_hour.corr("spearman")
```

Out[177]:

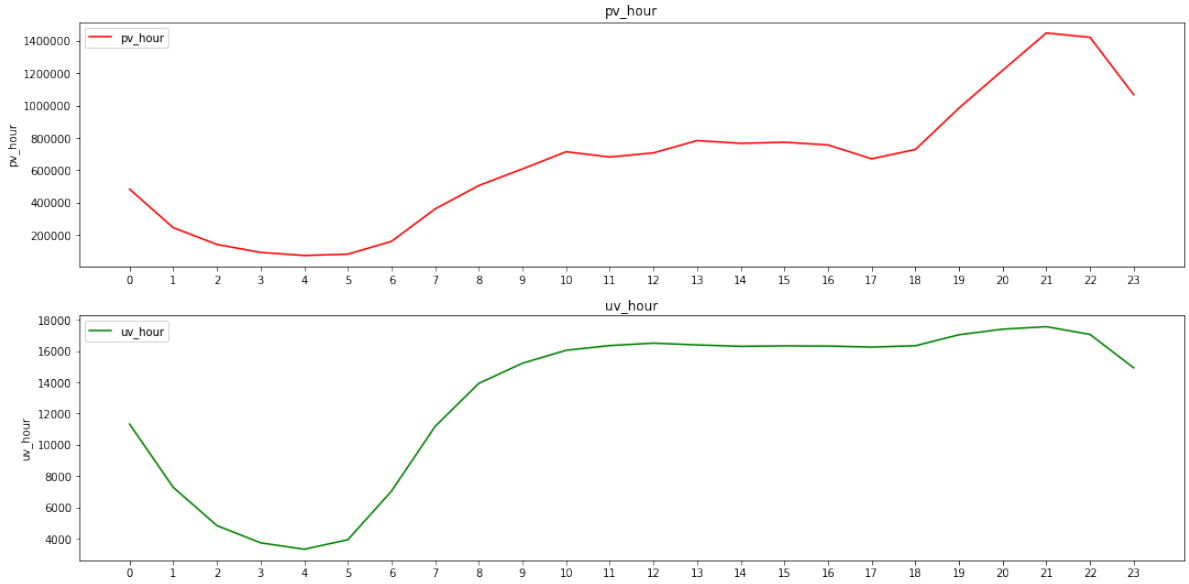
	pv	uv
pv	1.000000	0.896522
uv	0.896522	1.000000

In [178]:
```
plt.figure(figsize = (18,9))

plt.subplot(211)
plt.plot(pv_hour,color="red",label="pv_hour")
plt.ylabel("pv_hour")
plt.title("pv_hour")
plt.xticks(range(0,24),pv_uv_hour.index)
plt.legend()

plt.subplot(212)
plt.plot(uv_hour,color="g",label="uv_hour")
plt.ylabel("uv_hour")
plt.title("uv_hour")
plt.xticks(range(0,24),pv_uv_hour.index)
plt.legend()


plt.show()
```



由图可以看出，

0:00-5:00:pv与uv的波动情况比较相似，呈下降趋势，访问量比较小；

5:00-10:00:pv与uv呈上升趋势；

18:00-:pv与uv呈上升趋势，用户高度活跃；

In [179]:
```
#用户在一天不同时间段的访问量及不同行为类型统计
#beheviour_type:1,2,3,4----点击，收藏，加购物车，支付

pv_detail = pd.pivot_table(columns = "behavior_type",index="hour",d
ata = user,aggfunc = np.size , values = "user_id")
```

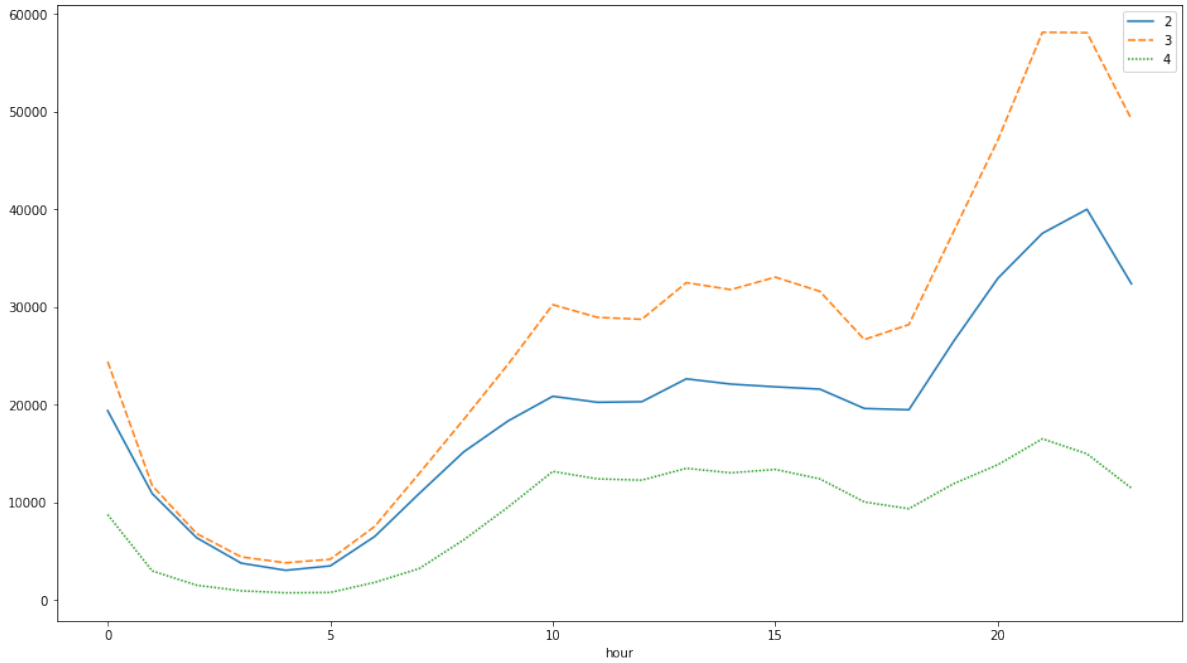
In [180]:
```
print(pv_detail)
```

```
behavior_type        1      2      3      4
hour
0                430156  19372  24400   8730
1                219200  10876  11695   2952
2                126490   6346   6778   1488
3                 82767   3747   4386    914
4                 65308   3014   3783    708
5                 72672   3464   4144    744
6                144520   6481   7511   1779
7                333666  10883  12943   3194
8                465465  15152  18476   6140
9                555753  18331  24150   9505
10               649628  20843  30228  13147
11               619783  20229  28930  12383
12               645664  20287  28729  12253
13               714528  22631  32487  13462
14               699006  22090  31770  12999
15               704532  21813  33049  13350
16               690304  21577  31593  12382
17               613898  19596  26655  10014
18               670934  19462  28190   9325
19               908145  26444  37670  11877
20              1123477  32935  47110  13838
21              1335962  37541  58135  16486
22              1308043  40000  58094  14950
23               973540  32365  49229  11435
```

In [181]:
```
#剔除了点击量的可视化显示

plt.figure(figsize=(16,9))
sns.lineplot(data = pv_detail.iloc[:,1:])

plt.show()
```



用户行为转化漏斗计算

In [182]:
```
#查看不同用户行为类型的次数

user_type_count = user.groupby("behavior_type").size()
print(user_type_count)
```

```
behavior_type
1    14153441
2      455479
3      640135
4      214055
dtype: int64
```

In [183]:
```
#1.点击量到收藏的流失率
(user_type_count[0] - user_type_count[1])/user_type_count[0]
```

Out[183]:
```
0.9678184972827456
```

In [184]:
```
#2.点击量到加购的流失率
(user_type_count[0] - user_type_count[2])/user_type_count[0]
```

Out[184]:
```
0.9547717759942618
```

从以上分析可以看出，浏览到确定购买意向的流失率太大，说明在点击-收藏-收购的这一行为过程中，用户可能需要花大量时间才能寻找到适合的产品。 可以针对性的优化平台的筛选功能，让用户能够更容易地寻找到适合的产品，并将流程指标再细化后进行分析，找到影响用户流失的关键问题点。

In [185]:
```
#3.点击量到支付的流失率
(user_type_count[0] - user_type_count[3])/user_type_count[0]
```

Out[185]:
```
0.9848761159918638
```

In [186]:
```
#4.加购到支付的流失率
(user_type_count[2]-user_type_count[3])/user_type_count[2]
```

Out[186]:
```
0.6656095979754271
```

In [187]:
```
#统计每位用户的消费次数
per_buy = user[user.behavior_type =="4"].groupby("user_id").size()
per_buy.head()
```

Out[187]:
```
user_id
100002985    3
10001025     9
10001082     4
100027096    1
100029775    5
dtype: int64
```

In [188]:
```
per_buy.describe()
```

Out[188]:
```
count    17654.000000
mean        12.125014
std         14.460703
min          1.000000
25%          4.000000
50%          8.000000
75%         15.000000
max        342.000000
dtype: float64
```

发现用户消费次数普遍在4次以下，需要重点关注消费在4次以内的群体

In [189]:
```
#每位付费用户日消费次数

day_buy = user[user["behavior_type"] == "4"].groupby(["date","user_id"]).count()["behavior_type"]

day_buy = day_buy.reset_index().rename(columns={"behavior_type":"total"})

day_buy.head()
```

Out[189]:

	date	user_id	total
0	2014-11-18	100051753	5
1	2014-11-18	100121281	1
2	2014-11-18	100212013	2
3	2014-11-18	100226515	1
4	2014-11-18	100234519	3

In [190]:
```
#每日消费次数
tt=day_buy.groupby("date")["total"].sum()
#每日消费人数
pp=day_buy.groupby("date").count()["user_id"]


#每日人均消费次数
per_buy = tt/pp


print(per_buy)
```

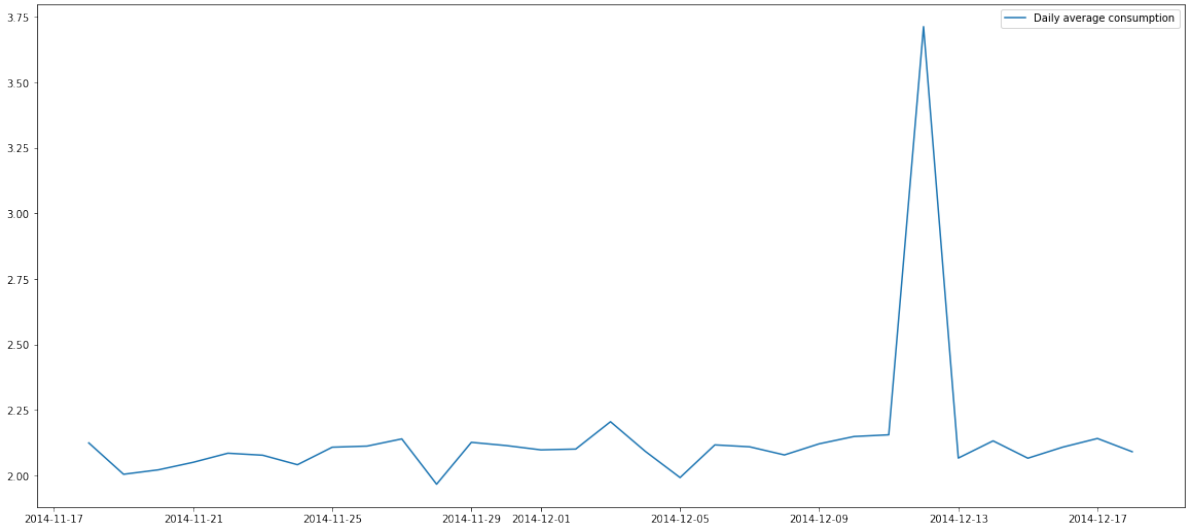
```
date
2014-11-18    2.124739
2014-11-19    2.005133
2014-11-20    2.022013
2014-11-21    2.050719
2014-11-22    2.085414
2014-11-23    2.077774
2014-11-24    2.041559
2014-11-25    2.108361
2014-11-26    2.112566
2014-11-27    2.140062
2014-11-28    1.967115
2014-11-29    2.126937
2014-11-30    2.114645
2014-12-01    2.098128
2014-12-02    2.101234
2014-12-03    2.205610
2014-12-04    2.092599
2014-12-05    1.992511
2014-12-06    2.117193
2014-12-07    2.109871
2014-12-08    2.078814
2014-12-09    2.121331
2014-12-10    2.149181
2014-12-11    2.155670
2014-12-12    3.713046
2014-12-13    2.066881
2014-12-14    2.132567
2014-12-15    2.066382
2014-12-16    2.108527
2014-12-17    2.141686
2014-12-18    2.091058
dtype: float64
```

In [191]:
```
per_buy.describe()
```

Out[191]:
```
count    31.000000
mean      2.145785
std       0.295204
min       1.967115
25%       2.072327
50%       2.108361
75%       2.125838
max       3.713046
dtype: float64
```

可以看出，每日人均消费次数为2.14

In [192]:
```
plt.figure(figsize=(20,9))
sns.lineplot(data=per_buy,label="Daily average consumption")
plt.show()
```



In [193]:
```
#用户的活跃统计

user_count = user.groupby(["date","user_id","behavior_type"]).count
()["item_id"].reset_index().rename(columns={"item_id":"total"})

user_count.head()
```

Out[193]:

	date	user_id	behavior_type	total
0	2014-11-18	10001025	1	65
1	2014-11-18	10001025	2	1
2	2014-11-18	100029775	1	3
3	2014-11-18	100045402	1	11
4	2014-11-18	100051753	1	37

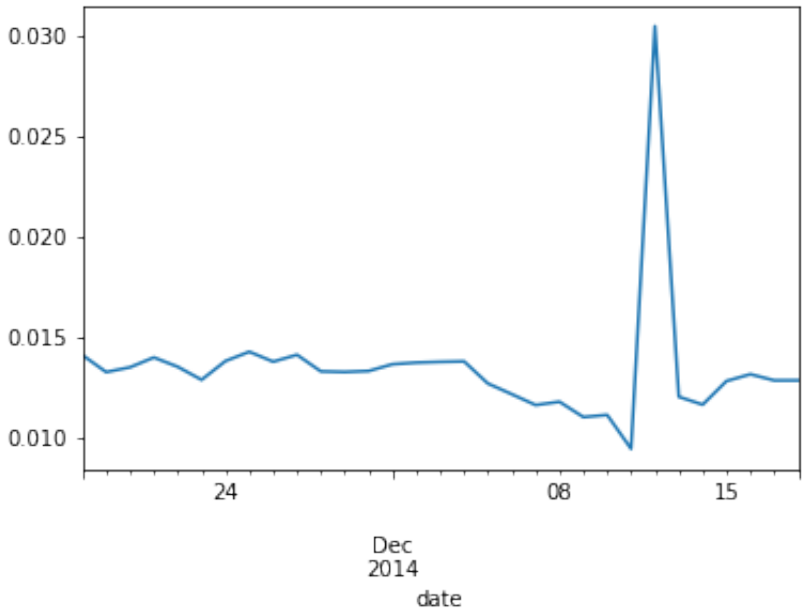
In [200]:
```
#每日用户活跃总次数
hh=user_count.groupby("date")["total"].sum()

#每日用户付费次数

bb = user_count[user_count["behavior_type"] == "4"].groupby("date")
["total"].sum()

#一日内的付费用户占比
qq=bb/hh

qq.plot()
plt.show()
```



In [201]:
```
#回购率

rebuy=user[user["behavior_type"]=="4"].groupby("user_id")["date"].c
ount()
rebuy[rebuy>=2].count()/rebuy.count()
```

Out[201]: 0.9156565084400136

用户价值分层(RFM模型)

由于数据缺少M（消费金额）列，暂且通过R（最近一次购买时间）和F（消费频率）的数据对客户价值进行打分。

In [207]:
```
recent_buy_time = user[user["behavior_type"]=="4"].groupby("user_id
")["date"].apply(lambda x:datetime(2014,12,20)-x.sort_values().iloc
[-1])
recent_buy_time
```

Out[207]:
```
user_id
100002985     3 days
10001025      8 days
10001082      6 days
100027096    12 days
100029775     7 days
               ...
65332965      6 days
65341491      4 days
6534626       3 days
65353086      3 days
6536165      24 days
Name: date, Length: 17654, dtype: timedelta64[ns]
```

In [209]:
```
recent_buy_time = recent_buy_time.reset_index().rename(columns={"da
te":"recent"})
recent_buy_time['recent'] = recent_buy_time['recent'].map(lambda x:
x.days)
recent_buy_time.head()
```

Out[209]:

	index	user_id	recent
0	0	100002985	3
1	1	10001025	8
2	2	10001082	6
3	3	100027096	12
4	4	100029775	7

In [211]:
```
buy_freq = user[user["behavior_type"]=="4"].groupby("user_id")["dat
e"].count().reset_index().rename(columns={"date":"freq"})
buy_freq.head()
```

Out[211]:

	user_id	freq
0	100002985	3
1	10001025	9
2	10001082	4
3	100027096	1
4	100029775	5

In [212]:
```
RFM = pd.merge(recent_buy_time,buy_freq,left_on="user_id",right_on="user_id")
RFM.head()
```

Out[212]:

	index	user_id	recent	freq
0	0	100002985	3	3
1	1	10001025	8	9
2	2	10001082	6	4
3	3	100027096	12	1
4	4	100029775	7	5

In [213]:
```
RFM['recent_value'] = pd.qcut(RFM.recent,2,labels=['1','0'])
RFM['freq_value'] = pd.qcut(RFM.freq,2,labels=['0','1'])
RFM.head()
```

Out[213]:

	index	user_id	recent	freq	recent_value	freq_value
0	0	100002985	3	3	1	0
1	1	10001025	8	9	0	1
2	2	10001082	6	4	1	0
3	3	100027096	12	1	0	0
4	4	100029775	7	5	0	0

In [214]:
```
RFM['RFM'] = RFM['recent_value'].str.cat(RFM['freq_value'])
RFM.head()
```

Out[214]:

	index	user_id	recent	freq	recent_value	freq_value	RFM
0	0	100002985	3	3	1	0	10
1	1	10001025	8	9	0	1	01
2	2	10001082	6	4	1	0	10
3	3	100027096	12	1	0	0	00
4	4	100029775	7	5	0	0	00

In [215]:
```
#查看用户等级数量的分布
RFM["RFM"].value_counts()
```

Out[215]:
```
11    5981
00    5918
10    3572
01    2183
Name: RFM, dtype: int64
```

In [220]:
```
def trans_value(x):
    if x=="11":
        return "Key customers"
    elif x=="10":
        return "Developing customers"
    elif x=="01":
        return "Retention customers"
    elif x=="00":
        return "Common customers"
```

In [221]:
```
RFM["classification"] = RFM["RFM"].map(trans_value)
```

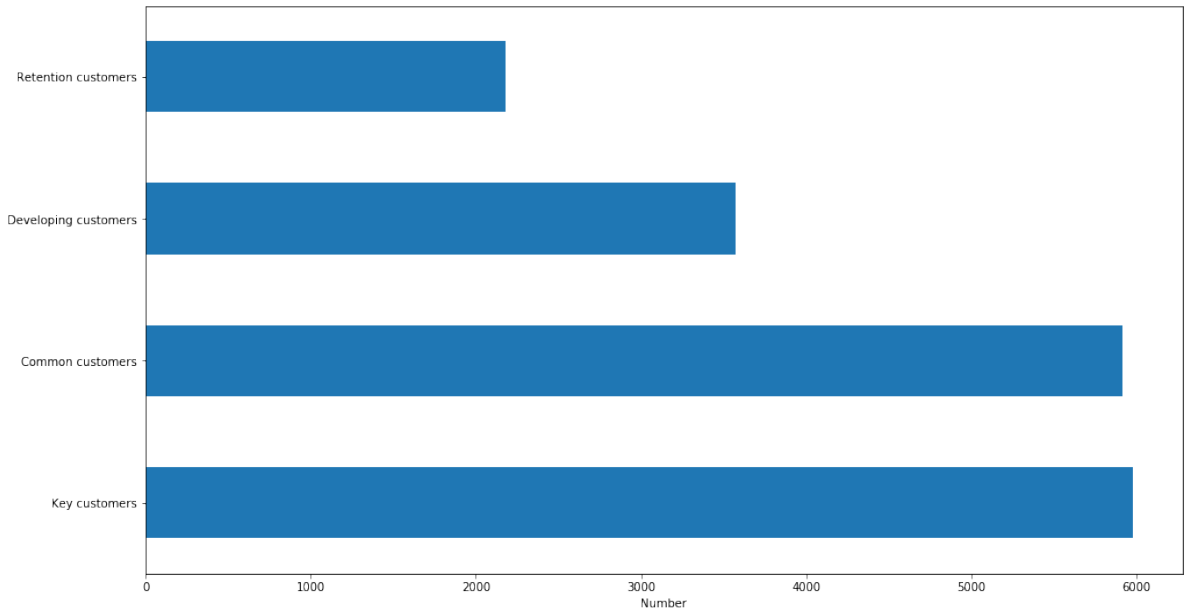
In [222]:
```
RFM.head()
```

Out[222]:

	index	user_id	recent	freq	recent_value	freq_value	RFM	classification
0	0	100002985	3	3	1	0	10	Developing customers
1	1	10001025	8	9	0	1	01	Retention customers
2	2	10001082	6	4	1	0	10	Developing customers
3	3	100027096	12	1	0	0	00	Common customers
4	4	100029775	7	5	0	0	00	Common customers

In [225]:
```
#对客户等级做可视化处理
plt.figure(figsize=(16,9))
RFM["classification"].value_counts().plot(kind="barh")
plt.xlabel("Number")

plt.show()
```



对于重要价值用户，他们是最优质的用户，需要重点关注并保持， 应该提高满意度，增加留存；

对于重要保持用户，他们最近有购买，但购买频率不高，可以通过活动等提高其购买频率；

对于重要发展用户，他们虽然最近没有购买，但以往购买频率高，可以做触达，以防止流失；

对于一般价值用户，他们最近没有购买，以往购买频率也不高，特别容易流失，所以应该赠送优惠券或推送活动信息，唤醒购买意愿.

结论与建议

1.激活用户：

用户流失率高，针对这个问题：

(1)优化电商平台的搜索匹配度和推荐策略，根据用户喜好推荐相关的商品；

(2)给客户提供同类产品比较的功能；

(3)随机筛选用户进行调查问卷，采纳客户建议。

2.提高收入:

(1)一天时间内，18点以后客户浏览量处于上升阶段，此时可以多推出一些营销活动，提高店铺收入。

(2)回购率在69%左右，应将经营重点转化为培养用户的忠诚度上，鼓励用户更高频次的消费。

3.通过RFM模型找出最具价值的核心付费用户群，对这部分用户的行为进行分析:

(1)等级是11的用户是体系中的最有价值用户，需要重点关注。并且活动投放时需谨慎对待，不要引起用户反感。

(2)对于价值评分是10的用户，消费时间间隔较短，运营活动可以重点针对这部分用户，提高用户的产品使用频率。

In [ ]: