Data Analysis

Summarize:

1. This part shows the work of data analysis.

2. The basic part has been finished.

3. Difficulity: How to find the two product which are always bought together. We only want **one** pair of product.

We have calculated the times of every pair of product are bought together, all we need now is a good standard to decide.

Details:

1. I need to get some statistic data from the original dataset and write the data to a file. The front end use the data to draw the charts, such as histogram and pie chart.

Here are the data we need and the related code.

(1) Top 10

(a) Top 10 sales.

```
blackFriday = pd.read_csv(’BlackFriday.csv’)
a1=blackFriday.groupby([’Product_ID’],as_index=False).agg({’Purchase’:sum})
a1.sort_values([’Purchase’], ascending=False, inplace=True)
```

```
      Product_ID Purchase
249   P00025442  27532426
1014  P00110742  26382569
2441  P00255842  24652442
1743  P00184942  24060871
581   P00059442  23948299
1028  P00112142  23882624
1016  P00110942  23232538
2261  P00237542  23096487
565   P00057642  22493690
104   P00010742  21865042
```

(b) Top 10 product category

```
a1=blackFriday.groupby([’Product_Category_1’],as_index=False).agg({’Purchase’:sum})
a1.sort_values([’Purchase’], ascending=False, inplace=True)
```

```
    Product_Category_1    Purchase
0                    1  1882666325
4                    5   926917497
7                    8   840693394
5                    6   319355286
1                    2   264497242
2                    3   200412211
15                  16   143168035
10                  11   112203088
9                   10    99029631
14                  15    91658147
```

(2) Top 10 buyers.

```
a2=blackFriday[[’Product_Category_1’,’Purchase’]].groupby([’Product_Category_1’
],as_index=False).agg({’Purchase’:sum})
a2.sort_values([’Purchase’], ascending=False, inplace=True)
```

```
      User_ID  Purchase
4166  1004277  10536783
1634  1001680   8699232
2831  1002909   7577505
1885  1001941   6817493
416   1000424   6573609
4335  1004448   6565878
981   1001015   6511302
3297  1003391   6476786
1142  1001181   6387899
534   1000549   6310604
```

(3) The purchase of man and woman.

```
a3=blackFriday[[’Gender’,’Purchase’]].groupby([’Gender’],as_index=False).agg({’
Purchase’:sum})
a3.sort_values([’Purchase’],ascending=False,inplace=True)
```

```
     Gender     Purchase
1      M        3853044357
0      F        1164624021
```

(a)Top 10 product that man like most.

```
woman=blackFriday[[’Gender’,’Product_ID’,’Purchase’]].groupby(’Gender’).get_group(’F’).groupby(’Product_ID’).agg({’Purchase’:sum})
woman.sort_values([’Purchase’],ascending=False,inplace=True)
```

```
Product_ID  Purchase
P00255842    6690088
P00059442    6007826
P00110842    5933348
P00025442    5763524
P00110742    5632357
P00110942    5066142
P00148642    5049905
P00112142    4901047
P00028842    4867128
P00184942    4723224
```

(b)Top 10 product that woman like most.

```
man=blackFriday[[’Gender’,’Product_ID’,’Purchase’]].groupby(’Gender’).get_group(’M’)
man2 = man.groupby(’Product_ID’).agg({’Purchase’:sum})
man2.sort_values([’Purchase’],ascending=False,inplace=True)
```

```
Product_ID  Purchase
P00025442   21768902
P00110742   20750212
P00184942   19337647
P00112142   18981577
P00057642   18720360
P00237542   18562039
P00110942   18166396
P00255842   17962354
P00059442   17940473
P00010742   17517618
```

(4) The purchase of different age.

```
a4=blackFriday[[’Age’,’Purchase’]].groupby([’Age’],as_index=False).agg({’Purchase’:sum})
a4.sort_values([’Purchase’],ascending=False,inplace=True)
```

```
     Age    Purchase
2  26-35  1999749106
3  36-45  1010649565
1  18-25   901669280
4  46-50   413418223
5  51-55   361908356
6    55+   197614842
0   0-17   132659006
```

(a)Top 10 product that people whose age of 0-17 like most.

```
a0_17=blackFriday.groupby(’Age’).get_group(’0-17’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
Product_ID    Purchase
P00255842    1096484
P00237542     946872
P00145042     935033
P00112142     931216
P00025442     852540
P00242742     787132
P00184942     728494
P00110742     724021
P00355142     643958
P00110942     634797
```

(b)Top 10 product that people whose age of 18-25 like most.

```
a18_25=blackFriday.groupby(’Age’).get_group(’18-25’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
    Product_ID    Purchase
    P00110742    5532933
    P00112142    5479058
```

```
P00237542    5029687
P00255842    4954222
P00010742    4944820
P00025442    4884642
P00110842    4678954
P00184942    4587243
P00028842    4566353
P00057642    4446409
```

(c)Top 10 product that people whose age of 25-35 like most.

```
a26_35=blackFriday.groupby(’Age’).get_group(’26-35’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
Product_ID  Purchase
P00110742   10605442
P00025442   10594786
P00255842    9860878
P00237542    9697110
P00184942    9493975
P00028842    9286868
P00112142    9258356
P00110942    9218356
P00059442    9211235
P00057642    9110947
```

(d)Top 10 product that people whose age of 36-45 like most.

```
a36_45 = blackFriday.groupby(’Age’).get_group(’36-45’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
Product_ID    Purchase
P00025442     5917938
P00110742     5105081
P00255842     4774004
P00059442     4769210
P00110942     4666976
```

```
P00057642     4645954
P00184942     4564488
P00052842     4493193
P00080342     4414988
P00112142     4368457
```

(e)Top 10 product that people whose age of 46-50 like most.

```
a46_50 = blackFriday.groupby(’Age’).get_group(’46-50’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

```
Product_ID    Purchase
P00025442     2098048
P00184942     2024545
P00059442     1998392
P00046742     1947669
P00110942     1875581
P00148642     1865060
P00080342     1803851
P00255842     1778397
P00116142     1696717
P00112142     1689463
```

(f)Top 10 product that people whose age of 51-55 like most.

```
a51_55 = blackFriday.groupby(’Age’).get_group(’51-55’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

```
Product_ID    Purchase
P00025442     2041357
P00059442     1985347
P00080342     1853997
P00110742     1812670
P00010742     1738741
P00052842     1656476
P00121342     1589537
P00116142     1587614
```

```
P00057642     1576856
P00148642     1525972
```

(g)Top 10 product that people whose age of 55+ like most.

```
a55Up=blackFriday.groupby(’Age’).get_group(’55+’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
Product_ID    Purchase
P00080342     1341782
P00059442     1262662
P00184942     1222830
P00085342     1150944
P00025442     1143115
P00110942      983101
P00116142      976456
P00010742      951037
P00110742      917425
P00121342      901195
```

(5) The purchase of different occupation.

```
a5=blackFriday[[’Occupation’,’Purchase’]].groupby([’Occupation’],as_index=False).agg({’Purchase’:sum})
a5.sort_values([’Purchase’],ascending=False,inplace=True)
```

```
Occupation       Purchase
4             4  657530393
0             0  625814811
7             7  549282744
1             1  414552829
17           17  387240355
12           12  300672105
20           20  292276985
14           14  255594745
16           16  234442330
2             2  233275393
6             6  185065697
3             3  160428450
```

```
15            15  116540026
10            10  114273954
5              5  112525355
11            11  105437359
19            19   73115489
13            13   71135744
18            18   60249706
9              9   53619309
8              8   14594599
```

(a)Top 10 product that occupation 0 like most.

```
o0=blackFriday.groupby(’Occupation’).get_group(0).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(b)Top 10 product that occupation 1 like most.

```
o1=blackFriday.groupby(’Occupation’).get_group(1).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(c)Top 10 product that occupation 2 like most.

```
o2=blackFriday.groupby(’Occupation’).get_group(2).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(d)Top 10 product that occupation 3 like most.

```
o3=blackFriday.groupby(’Occupation’).get_group(3).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(e)Top 10 product that occupation 4 like most.

```
o4=blackFriday.groupby(’Occupation’).get_group(4).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(f)Top 10 product that occupation 5 like most.

```
o5=blackFriday.groupby(’Occupation’).get_group(5).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(g)Top 10 product that occupation 6 like most.

```
o6=blackFriday.groupby(’Occupation’).get_group(6).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(h)Top 10 product that occupation 7 like most.

```
o7=blackFriday.groupby(’Occupation’).get_group(7).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False, ascending=False)
```

(i)Top 10 product that occupation 8 like most.

```
o8=blackFriday.groupby(’Occupation’).get_group(8).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

(j)Top 10 product that occupation 9 like most.

```
o9=blackFriday.groupby(’Occupation’).get_group(9).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

(k)Top 10 product that occupation 10 like most.

(l)Top 10 product that occupation 11 like most.

(m)Top 10 product that occupation 12 like most.

(n)Top 10 product that occupation 13 like most.

(o)Top 10 product that occupation 14 like most.

(p)Top 10 product that occupation 15 like most.

(q)Top 10 product that occupation 16 like most.

(r)Top 10 product that occupation 17 like most.

(s)Top 10 product that occupation 18 like most.

(t)Top 10 product that occupation 19 like most.

(u)Top 10 product that occupation 20 like most.

```
o10 = blackFriday.groupby(’Occupation’).get_group(10).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o11 = blackFriday.groupby(’Occupation’).get_group(11).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o12 = blackFriday.groupby(’Occupation’).get_group(12).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o13 = blackFriday.groupby(’Occupation’).get_group(13).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o14 = blackFriday.groupby(’Occupation’).get_group(14).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o15 =
blackFriday.groupby(’Occupation’).get_group(15).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o16 =
blackFriday.groupby(’Occupation’).get_group(16).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o17 =
blackFriday.groupby(’Occupation’).get_group(17).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o18 =
blackFriday.groupby(’Occupation’).get_group(18).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o19 =
blackFriday.groupby(’Occupation’).get_group(19).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
o20 =
blackFriday.groupby(’Occupation’).get_group(20).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
0
            Purchase
Product_ID
P00025442    3082080
P00110742    3046312
P00059442    2803380
P00184942    2770200
P00057642    2624741
P00255842    2618002
P00237542    2613143
P00112142    2532049
P00110842    2500692
P00110942    2343433

1
            Purchase
```

```
Product_ID
P00110742    2060684
P00025442    2039871
P00059442    2027632
P00255842    1920305
P00184942    1907123
P00080342    1843757
P00110842    1816385
P00110942    1811047
P00028842    1791146
P00112142    1755209

2
            Purchase
Product_ID
P00025442    1314065
P00059442    1203452
P00110842    1128831
P00110742    1048300
P00052842    1028403
P00237542     953969
P00112142     939542
P00057642     909862
P00110942     907640
P00080342     877452

3
            Purchase
Product_ID
P00255842     792072
P00025442     749383
P00110842     740812
P00059442     740285
```

```
P00110742     663237
P00110942     654321
P00237542     647903
P00057642     624675
P00148642     611491
P00114942     602232

4
            Purchase
Product_ID
P00110742    4048063
P00025442    3821259
P00112142    3783361
P00237542    3745206
P00028842    3546970
P00255842    3515094
P00184942    3496923
P00010742    3462799
P00110942    3320694
P00110842    3249265

5
            Purchase
Product_ID
P00114942     579614
P00110742     569372
P00025442     557516
P00057642     526306
P00112542     520450
P00255842     514009
P00080342     503367
P00128942     485208
P00237542     480876
```

```
P00010742     470592

6
            Purchase
Product_ID
P00255842    1080045
P00110742    1011923
P00025442     967611
P00184942     943994
P00010742     942453
P00080342     929441
P00148642     863011
P00112142     850808
P00057642     841797
P00059442     837661

7
            Purchase
Product_ID
P00025442    3155471
P00110742    3097921
P00110942    3013049
P00184942    2998813
P00255842    2792269
P00010742    2723861
P00059442    2693158
P00112142    2692172
P00080342    2626270
P00046742    2530922

8
            Purchase
Product_ID
```

```
P00052842      118467
P00112142      113742
P00114942       99863
P00242742       96941
P00127642       84689
P00127842       78672
P00270942       74379
P00046742       73501
P00016042       70584
P00110842       69555

9
            Purchase
Product_ID
P00059442      342833
P00145042      306907
P00255842      305520
P00110842      259411
P00184942      253922
P00110942      252359
P00221442      240836
P00110742      236724
P00085942      231542
P00000142      222531

10
            Purchase
Product_ID
P00255842      958951
P00145042      957464
P00025442      886768
P00112142      871275
P00237542      871101
```

```
P00242742     794823
P00184942     674123
P00110742     633429
P00334242     609386
P00355142     592296

11
            Purchase
Product_ID
P00025442     641609
P00059442     629457
P00148642     611638
P00110942     600271
P00080342     592248
P00052842     589440
P00112142     562794
P00184942     548051
P00113242     526591
P00112442     524780


12
            Purchase
Product_ID
P00025442    2076718
P00057642    2011741
P00112142    1929774
P00052842    1911324
P00237542    1893729
P00110742    1836883
P00255842    1777972
P00110942    1739226
P00114942    1679938
```

```
P00059442    1560621

13
            Purchase
Product_ID
P00010742     498547
P00080342     480635
P00184942     417389
P00025442     395832
P00177442     373318
P00110742     365628
P00114342     359853
P00057642     353588
P00116142     352670
P00110942     336613

14
            Purchase
Product_ID
P00184942    1556438
P00025442    1415003
P00148642    1349608
P00237542    1334397
P00005042    1315141
P00110742    1313562
P00028842    1266773
P00255842    1212831
P00010742    1201460
P00110942    1156946

15
            Purchase
Product_ID
```

```
P00025442     902703
P00110742     793929
P00110942     733301
P00059442     714088
P00110842     690887
P00112142     690020
P00255842     684399
P00057642     679579
P00080342     624583
P00046742     588535

16
            Purchase
Product_ID
P00255842    1063543
P00025442    1038864
P00110942    1014872
P00052842    1001701
P00110742     974492
P00046742     968884
P00148642     950646
P00184942     909690
P00059442     884547
P00005042     865338

17
            Purchase
Product_ID
P00025442    2426017
P00110742    2329323
P00057642    2315821
P00237542    2298225
P00112142    2234829
```

```
P00255842    2234351
P00184942    2200776
P00110942    2139736
P00114942    1961365
P00046742    1938491

18
            Purchase
Product_ID
P00010742     416243
P00080342     390289
P00184942     315384
P00028842     287513
P00046742     283143
P00057642     272759
P00112142     260348
P00059442     259517
P00112542     259474
P00110842     251661

19
            Purchase
Product_ID
P00237542     419498
P00059442     377675
P00111742     360504
P00028842     358315
P00112142     346839
P00025442     337630
P00071442     334536
P00145042     330857
P00010742     314734
P00112442     313479
```

```
20
            Purchase
Product_ID
P00059442    1543162
P00110742    1323818
P00025442    1251344
P00052842    1242651
P00148642    1186607
P00110842    1183362
P00080342    1148077
P00028842    1108659
P00255842    1070224
P00184942    1045758
```

(6) The purchase of people in different city.

(a) Top 10 product that city A like most.

(b) Top 10 product that city B like most.

(c) Top 10 product that city C like most.

```
a6 =
blackFriday[[’City_Category’,’Purchase’]].groupby([’City_Category’],as_index=False).agg({’Purchase’:sum})
a6.sort_values([’Purchase’],ascending=False,inplace=True)
A =
blackFriday.groupby(’City_Category’).get_group(’A’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
B =
blackFriday.groupby(’City_Category’).get_group(’B’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
C =
blackFriday.groupby(’City_Category’).get_group(’C’).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
  City_Category    Purchase
```

```
1             B  2083431612
2             C  1638567969
0             A  1295668797

A
           Purchase
Product_ID
P00025442   5386234
P00059442   5346393
P00255842   5332815
P00110742   5311364
P00110842   4977844
P00110942   4954446
P00052842   4911429
P00237542   4772290
P00057642   4700520
P00028842   4638774

B
           Purchase
Product_ID
P00110742   9844481
P00025442   9667058
P00059442   8815852
P00184942   8707024
P00237542   8521897
P00028842   8361856
P00110942   8341816
P00255842   8319809
P00010742   8282542
P00112142   8241958

C
```

```
            Purchase
Product_ID
P00025442   12479134
P00110742   11226724
P00112142   11052443
P00255842   10999818
P00184942   10978970
P00110942    9936276
P00057642    9891283
P00010742    9834335
P00237542    9802300
P00059442    9786054
```

(7) The purchase of people in different living time.

(a) Top 10 product that people who live in the city for 1 year like most.

(b) Top 10 product that people who live in the city for 2 year like most.

(c) Top 10 product that people who live in the city for 3 year like most.

(d) Top 10 product that people who live in the city for 4 and more than 4 year like most.

```
a7 = blackFriday[['Stay_In_Current_City_Years','Purchase']].groupby(['Stay_In_Current_City_Years'],as_index=False).agg({'Purchase':sum})
a7.sort_values(['Purchase'],ascending=False,inplace=True)
s0 = blackFriday.groupby('Stay_In_Current_City_Years').get_group('0').groupby('Product_ID').agg({'Purchase':sum}).sort_values("Purchase",inplace=False, ascending=False)
s1 = blackFriday.groupby('Stay_In_Current_City_Years').get_group('1').groupby('Product_ID').agg({'Purchase':sum}).sort_values("Purchase",inplace=False, ascending=False)
s2 = blackFriday.groupby('Stay_In_Current_City_Years').get_group('2').groupby('Product_ID').agg({'Purchase':sum}).sort_values("Purchase",inplace=False,
ascending=False)
s3 =
blackFriday.groupby(’Stay_In_Current_City_Years’).get_group(’3’).groupby(’Produ
ct_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False,
ascending=False)
s4Up =
blackFriday.groupby(’Stay_In_Current_City_Years’).get_group(’4+’).groupby(’Prod
uct_ID’).agg({’Purchase’:sum}).sort_values(″Purchase″,inplace=False,
ascending=False)
```

```
1                            1  1763243917
2                            2   934676626
3                            3   872531130
4                           4+   774711276
0                            0   672505429

0
             Purchase
Product_ID
P00025442     4027605
P00255842     3333440
P00110742     3319509
P00112142     3301452
P00110942     3230136
P00057642     3150441
P00184942     3032851
P00110842     3027622
P00059442     2958983
P00237542     2902598

1
             Purchase
Product_ID
P00025442     9360495
```

```
P00110742    9309594
P00255842    8903024
P00184942    8621415
P00112142    8406403
P00110942    8306778
P00057642    8138730
P00010742    8136005
P00059442    8120583
P00237542    8103709

2
           Purchase
Product_ID
P00025442    5514347
P00110742    5103819
P00112142    4792310
P00184942    4651101
P00059442    4620803
P00255842    4604033
P00110842    4404119
P00237542    4352629
P00010742    4349317
P00057642    4322813
3
           Purchase
Product_ID
P00025442    4805818
P00110742    4513712
P00059442    4261683
P00237542    4072551
P00110942    3959104
P00255842    3942298
P00184942    3780153
```

```
P00052842     3743709
P00112142     3583927
P00110842     3544879


4+
             Purchase
Product_ID
P00110742     4135935
P00059442     3986247
P00184942     3975351
P00255842     3869647
P00025442     3824161
P00112142     3798532
P00237542     3665000
P00080342     3607250
P00028842     3524975
P00110942     3506142
```

(8) The purchase of people who are married or not.

(a) Top 10 product that people who has been married like most.

(b) Top 10 product that people who has not been married like most.

```
a8 = blackFriday[[’Marital_Status’,’Purchase’]].groupby([’Marital_Status’],as_index=False).agg({’Purchase’:sum})
a8.sort_values([’Purchase’],ascending=False,inplace=True)
m0 = blackFriday.groupby(’Marital_Status’).get_group(0).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
m1 = blackFriday.groupby(’Marital_Status’).get_group(1).groupby(’Product_ID’).agg({’Purchase’:sum}).sort_values(”Purchase”,inplace=False, ascending=False)
```

```
   Marital_Status    Purchase
0               0  2966289500
1               1  2051378878
```

```
0
           Purchase
Product_ID
P00025442  16529903
P00110742  15887215
P00255842  15080130
P00112142  14458721
P00237542  14271524
P00059442  14042723
P00184942  13988935
P00110942  13978740
P00057642  13545888
P00028842  12993842


1
           Purchase
Product_ID
P00025442  11002523
P00110742  10495354
P00184942  10071936
P00059442   9905576
P00255842   9572312
P00112142   9423903
P00110942   9253798
P00010742   9227900
P00057642   8947802
P00080342   8894650
```

(9) Top 10 product that costumer who are age of 26-35 like most.

```
a9 =
blackFriday.groupby('Marital_Status').get_group(1).groupby('Gender').get_group(
'M').groupby('Product_ID').agg({'Purchase':sum})
a9.sort_values(['Purchase'],ascending=False,inplace=True)
```

```
a9 =
blackFriday.groupby('Marital_Status').get_group(1).groupby('Gender').get_group(
'F').groupby('Product_ID').agg({'Purchase':sum})
a9.sort_values(['Purchase'],ascending=False,inplace=True)
```

26-35

```
            Purchase
Product_ID
P00110742   10605442
P00025442   10594786
P00255842    9860878
P00237542    9697110
P00184942    9493975
P00028842    9286868
P00112142    9258356
P00110942    9218356
P00059442    9211235
P00057642    9110947
```

2

man

```
            Purchase
Product_ID
P00025442    8499099
P00110742    8213098
P00184942    7937618
P00059442    7469419
P00057642    7392952
P00112142    7364620
P00010742    7331324
P00110942    7131772
P00237542    7109247
P00080342    6936382
```

woman

```
            Purchase
Product_ID
P00255842    2794105
P00025442    2503424
P00110842    2481040
P00059442    2436157
P00148642    2360330
P00110742    2282256
P00184942    2134318
P00110942    2122026
P00112142    2059283
P00080342    1958268
```

2. We want to find the two products which are always bought together. Now we have the number of every two product are bought together, we haven’t decide the standard of the ‘most’.

The code are as follows.

```
import csv
import numpy as np
#只读打开
csvFile = open("BlackFriday.csv", "r")
reader = csv.reader(csvFile)
Uid_PidDic = {}
productSet = set()

for item in reader:
    # 忽略第一行
    if reader.line_num == 1:
        continue
    if item[0] in Uid_PidDic:
        Uid_PidDic[item[0]].append(item[1])
        productSet.add(item[1])
    else:
        Uid_PidDic[item[0]] = [item[1]]
        productSet.add(item[1])
#print(Uid_PidDic[’1000001’])
#print(len(Uid_PidDic))
csvFile.close()



#给商品重新编号
product_num = {}
num = 0
for item in productSet:
    product_num[item] = num
    num+=1



#商品对出现次数数组，以及商品单独出现次数
prodCor = np.zeros((3623,3623),dtype=np.int)
prodNum = np.zeros(3623,dtype=np.int)



values=Uid_PidDic.values()
location = 0

#取出某一个顾客的商品购买列表
for value in values:
    #print(’location: ’ + str(location), file=f)
    i = 1
    index = 0
#取出该顾客商品购买列表中的一个商品
    for product in value:
        prodNum[product_num[product]]+=1
        i = index + 1
        #print(’index: ’+ str(index), file=f)
        while i<len(value):
            prodCor[product_num[product]][product_num[value[i]]]+=1
            #print(prodCor[product_num[product]][product_num[value[i]]])
            i+=1
        index+=1
    location += 1

#f = open(”output.rtf”, ’w+’)
#f.close()

for i in range(len(prodCor)):
    for j in range(len(prodCor[i])):
        if i < j:
            temp = prodCor[i][j]
            prodCor[i][j] = prodCor[j][i] = temp + prodCor[j][i]




max = prodCor[0][0]
x=y=0
a=b=0
for i in range(len(prodCor)):
    for j in range(len(prodCor[i])):
        if i < j and (prodCor[i][j] > max):
            max = prodCor[i][j]
            x = i
            y = j
```

Addition

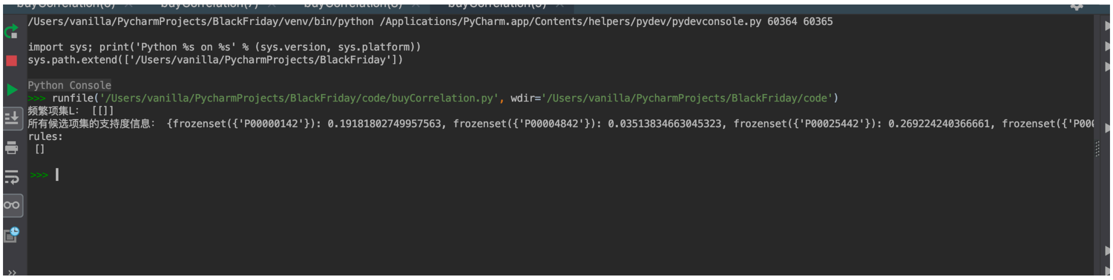
We finally solve the problem of finding out the correlation by using Apriori. Here is the code.

```

def loadDataSet():
    '''创建一个用于测试的简单的数据集'''
    return Uid_PidDic.values()



def createC1(dataSet):
    '''
        构建初始候选项集的列表，即所有候选项集只包含一个元素，
        C1 是大小为 1 的所有候选项集的集合
    '''
    C1 = []
    for transaction in dataSet:
        for item in transaction:
            if [item] not in C1:
                C1.append([item])
    C1.sort()
    # return map( frozenset, C1 )
    # return [var for var in map(frozenset,C1)]
    return [frozenset(var) for var in C1]



def scanD(D, Ck, minSupport):
    '''
        计算 Ck 中的项集在数据集合 D(记录或者 transactions)中的支持度,
        返回满足最小支持度的项集的集合，和所有项集支持度信息的字典。
    '''
    ssCnt = {}
    for tid in D:  # 对于每一条 transaction
        for can in Ck:  # 对于每一个候选项集 can，检查是否是 transaction 的一部分 # 即该候选 can 是否得到 transaction 的支持
            if can.issubset(tid):
                ssCnt[can] = ssCnt.get(can, 0) + 1
    numItems = float(len(D))
    retList = []
    supportData = {}
    for key in ssCnt:
        support = ssCnt[key] / numItems  # 每个项集的支持度
        if support >= minSupport:  # 将满足最小支持度的项集，加入 retList
            retList.insert(0, key)
        supportData[key] = support  # 汇总支持度数据
    return retList, supportData


def aprioriGen(Lk, k):  # Aprior 算法
    '''
        由初始候选项集的集合 Lk 生成新的生成候选项集，
        k 表示生成的新项集中所含有的元素个数
    '''
    retList = []
    lenLk = len(Lk)
    for i in range(lenLk):
        for j in range(i + 1, lenLk):
            L1 = list(Lk[i])[: k - 2];
            L2 = list(Lk[j])[: k - 2];
            L1.sort();
            L2.sort()
            if L1 == L2:
                retList.append(Lk[i] | Lk[j])
    return retList


def apriori(dataSet, minSupport=0.5):
    C1 = createC1(dataSet)  # 构建初始候选项集 C1
    # D = map( set, dataSet )                               # 将 dataSet 集合
化，以满足 scanD 的格式要求
    # D=[var for var in map(set,dataSet)]
    D = [set(var) for var in dataSet]
    L1, suppData = scanD(D, C1, minSupport)  # 构建初始的频繁项集，即所有项集只有一个元素
    L = [L1]  # 最初的 L1 中的每个项集含有一个元素，新生成的
    k = 2  # 项集应该含有 2 个元素，所以 k=2

    while (len(L[k - 2]) > 0):
        Ck = aprioriGen(L[k - 2], k)
        Lk, supK = scanD(D, Ck, minSupport)
        suppData.update(supK)  # 将新的项集的支持度数据加入原来的总支持度字典中
        L.append(Lk)  # 将符合最小支持度要求的项集加入 L
        k += 1  # 新生成的项集中的元素个数应不断增加
    return L, suppData  # 返回所有满足条件的频繁项集的列表，和所有候选项集的支持度信息



def calcConf(freqSet, H, supportData, brl, minConf=0.7):  # 规则生成与评价
    '''
        计算规则的可信度，返回满足最小可信度的规则。
        freqSet(frozenset):频繁项集
        H(frozenset):频繁项集中所有的元素
        supportData(dic):频繁项集中所有元素的支持度
        brl(tuple):满足可信度条件的关联规则
        minConf(float):最小可信度
    '''
    prunedH = []
    for conseq in H:
        conf = supportData[freqSet] / supportData[freqSet - conseq]
        if conf >= minConf:
            print(freqSet - conseq, '-->', conseq, 'conf:', conf)
            brl.append((freqSet - conseq, conseq, conf))
            prunedH.append(conseq)
    return prunedH
```

```
def rulesFromConseq(freqSet, H, supportData, brl, minConf=0.7):
    '''
        对频繁项集中元素超过 2 的项集进行合并。
        freqSet(frozenset):频繁项集
        H(frozenset):频繁项集中的所有元素，即可以出现在规则右部的元素
        supportData(dict):所有项集的支持度信息
        brl(tuple):生成的规则
    '''
    m = len(H[0])
    if len(freqSet) > m + 1:  # 查看频繁项集是否足够大，以到于移除大小为 m 的子集，否则继续生成 m+1 大小的频繁项集
        Hmp1 = aprioriGen(H, m + 1)
        Hmp1 = calcConf(freqSet, Hmp1, supportData, brl, minConf)  # 对于新生成的 m+1 大小的频繁项集，计算新生成的关联规则的右则的集合
        if len(Hmp1) > 1:  # 如果不止一条规则满足要求（新生成的关联规则的右则的集合的大小大于 1），进一步递归合并，
            # 这样做的结果就是会有“[1|多]->多”(右边只会是“多”，因为合并的本质是频繁子项集变大，
            # 而 calcConf 函数的关联结果的右侧就是频繁子项集）的关联结果
            rulesFromConseq(freqSet, Hmp1, supportData, brl, minConf)


def generateRules(L, supportData, minConf=0.7):
    '''
        根据频繁项集和最小可信度生成规则。
        L(list):存储频繁项集
        supportData(dict):存储着所有项集（不仅仅是频繁项集）的支持度
        minConf(float):最小可信度
    '''
    bigRuleList = []
    for i in range(1, len(L)):
        for freqSet in L[i]:  # 对于每一个频繁项集的集合 freqSet
```

```
            H1 = [frozenset([item]) for item in freqSet]
            if i > 1:  # 如果频繁项集中的元素个数大于 2，需要进一步合并，这样做的结果就是会有“[1|多]->多”(右边只会是“多”，
                # 因为合并的本质是频繁子项集变大，而 calcConf 函数的关联结果的右侧就是频繁子项集），的关联结果
                rulesFromConseq(freqSet, H1, supportData, bigRuleList, minConf)
            else:
                calcConf(freqSet, H1, supportData, bigRuleList, minConf)
    return bigRuleList



if __name__ == '__main__':
    myDat = loadDataSet()  # 导入数据集
    # C1 = createC1( myDat )                                   # 构建第一个候选项集列表 C1
    # D = map( set, myDat )                                    # 构建集合表示的数据集 D，python3 中的写法，或者下面那种
    # D=[var for var in map(set,myDat)]
    # D=[set(var) for var in myDat] #D: [{1, 3, 4}, {2, 3, 5}, {1, 2, 3, 5}, {2, 5}]
    # L, suppData = scanD( D, C1, 0.5 )                        # 选择出支持度不小于 0.5 的项集作为频繁项集
    # print(u"频繁项集 L：", L)
    # print(u"所有候选项集的支持度信息：", suppData)
    # print("myDat",myDat)
    L, suppData = apriori(myDat, 0.5)  # 选择频繁项集
    print(u"频繁项集 L：", L)
    print(u"所有候选项集的支持度信息：", suppData)
    rules = generateRules(L, suppData, minConf=0.7)
    print('rules:\n', rules)
```

And the result



```
/Users/vanilla/PycharmProjects/BlackFriday/venv/bin/python /Applications/PyCharm.app/Contents/helpers/pydev/pydevconsole.py 60364 60365

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/vanilla/PycharmProjects/BlackFriday'])

Python Console
>>> runfile('/Users/vanilla/PycharmProjects/BlackFriday/code/buyCorrelation.py', wdir='/Users/vanilla/PycharmProjects/BlackFriday/code')
频繁项集L:  [[]]
所有候选项集的支持度信息:  {frozenset({'P00000142'}): 0.19181802749957563, frozenset({'P00004842'}): 0.03513834663045323, frozenset({'P00025442'}): 0.269224240366661, frozenset({'P00
rules:
 []

>>> 
```