#导入数据分析相关库  
import numpy as np  
import pandas as pd  
from pandas import Series,DataFrame  
import matplotlib.pyplot as plt  
import seaborn as sns  
#设置幕布大小  
plt.rcParams['figure.figsize'] = (20.0,10.0)

# 数据处理相关函数

#函数说明  
#输入：df\_init(导入文件DataFrame) 输入：df（出除无关列数据DataFrame）  
#用途：生成df，以便后续函数处理  
def df\_init(df\_init):  
 df = df\_init.drop(df\_init.columns[[1,2,3,4,5,7,8]],axis = 1)  
 return df

#函数说明  
#输入：df的出入时间 输出：df的出入日期（相当于去掉秒分时）  
#用途：用于Data\_init\_day(df)函数，使df['出入时间'].map()可调用  
def ConvertToDay(time):  
 return int(time/1000000)

#函数说明  
#输入：df的出入时间 输出：df的出入小时（相当于去掉秒分）  
#用途：用于Data\_init\_Hour(df)函数，使df['出入时间'].map()可调用  
def ConvertToHour(time):  
 return int(time/10000)

#函数说明  
#输入：原始df 输出：数据初始化后的dfn  
#用途：由于原始数据含有少数冗余数据，需要剔除冗余数据以便于数据分析  
def Data\_init\_day(df):  
 s = df['出入时间'].map(ConvertToDay).tolist()#通过map，修改出入时间为出入日期，并将Series转换为list  
 df.insert(0,'出入日期',s)#通过insert，插入一列新数据，数据column名为‘出入日期’  
 df = df.drop(df.columns[3],axis = 1)#删除原来的‘出入日期’，注意如果反复使用该函数，需要初始化df  
 df = df.rename(columns = {'出入日期':'Date','工具类型':'ToolType','出入类型':'AccessType','验放标志':'label'})#传入字典，修改列名  
 df.drop(index = df[df['ToolType']>43].index,inplace=True)#剔除ToolType的冗余项  
 df.drop(index = df[df['label']>6].index,inplace=True)#剔除label的冗余项  
 df.drop(index = df[df['label']<5].index,inplace=True)#剔除label的冗余项  
 s = df.value\_counts(sort=False)#生成多级Series，统计每种类型搭配的数量  
 dfn = DataFrame(s)#根据多级Series生成dfn，dfn为透视表  
 dfn = dfn.rename(columns = {0:'Quantity'})#将统计数量column命名为Quantity  
 dfn = dfn.unstack().unstack().unstack()#多级DataFrame降维  
 return dfn

#函数说明  
#输入：原始df 输出：数据初始化后的dft  
#用途：由于原始数据含有少数冗余数据，需要剔除冗余数据以便于数据分析  
#与Data\_init\_day的区别：仅将ConvertToDay替换为ConvertToHour  
def Data\_init\_Hour(df):  
 s = df['出入时间'].map(ConvertToHour).tolist()  
 df.insert(0,'出入日期',s)  
 df = df.drop(df.columns[3],axis = 1)  
 df = df.rename(columns = {'出入日期':'Date','工具类型':'ToolType','出入类型':'AccessType','验放标志':'label'})#传入字典  
 df.drop(index = df[df['ToolType']>43].index,inplace=True)  
 df.drop(index = df[df['label']>6].index,inplace=True)  
 df.drop(index = df[df['label']<5].index,inplace=True)  
 s = df.value\_counts(sort=False)  
 dft = DataFrame(s)  
 dft = dft.rename(columns = {0:'Quantity'})  
 dft = dft.unstack().unstack().unstack()  
 return dft

# 文件导入

#我将一个文件的多个sheet拆分为多个excel，然后读入  
#如果直接读入，会使得程序读入较慢，直接读入代码df\_init = pd.read\_excel('A题附件2：出行车辆数据明细',sheet\_name = 2018..)  
#注意ipyon文件必须和待读入文件在同一目录下才能直接以文件名作为路径，否则需要填写相对路径或绝对路径  
df201801\_init = pd.read\_excel('201801.xlsx')  
df201802\_init = pd.read\_excel('201802.xlsx')  
df201803\_init = pd.read\_excel('201803.xlsx')  
df201804\_init = pd.read\_excel('201804.xlsx')  
df201805\_init = pd.read\_excel('201805.xlsx')  
df201806\_init = pd.read\_excel('201806.xlsx')  
df201807\_init = pd.read\_excel('201807.xlsx')  
df201808\_init = pd.read\_excel('201808.xlsx')  
df201809\_init = pd.read\_excel('201809.xlsx')  
df201810\_init = pd.read\_excel('201810.xlsx')  
df201811\_init = pd.read\_excel('201811.xlsx')  
df201812\_init = pd.read\_excel('201812.xlsx')

df201801\_init.head()#查看文件导入是否成功

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

工具类型
航次
境外标识
境内标志
出入口岸
通道号
出入类型
验放部门
检查员
出入时间
验放标志
0
41
UW3018
UW3018
粤Z.\*\*\*\*港
492
Z517
1
492108
44969
20180101063226
6
1
43
RR4055
RR4055
粤B6\*\*\*\*G
492
Z434
2
492105
46083
20180101063231
6
2
43
FV851
FV851
粤Z.\*\*\*\*港
492
Z433
2
492105
44956
20180101063246
6
3
41
UD3418
UD3418
粤B8\*\*\*\*S
492
Z518
1
492108
44969
20180101063310
6
4
43
FV1646
FV1646
粤K3\*\*\*\*8
492
Z422
2
492105
42792
20180101063342
5

# 调用函数处理df\_init,并导出透视表excel

#初始化  
df201801 = df\_init(df201801\_init)  
df201802 = df\_init(df201802\_init)  
df201803 = df\_init(df201803\_init)  
df201804 = df\_init(df201804\_init)  
df201805 = df\_init(df201805\_init)  
df201806 = df\_init(df201806\_init)  
df201807 = df\_init(df201807\_init)  
df201808 = df\_init(df201808\_init)  
df201809 = df\_init(df201809\_init)  
df201810 = df\_init(df201810\_init)  
df201811 = df\_init(df201811\_init)  
df201812 = df\_init(df201812\_init)

df201801#查看函数使用成功与否

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

工具类型
出入类型
出入时间
验放标志
0
41
1
20180101063226
6
1
43
2
20180101063231
6
2
43
2
20180101063246
6
3
41
1
20180101063310
6
4
43
2
20180101063342
5
...
...
...
...
...
417830
43
2
20180131235000
5
417831
43
1
20180131235012
5
417832
41
2
20180131235244
6
417833
43
2
20180131235445
5
417834
41
2
20180131235713
6

#生成透视表（如果不小心多点了一次，就得重新初始化，再调用这个函数了哦）  
df201801n = Data\_init\_day(df201801)  
df201802n = Data\_init\_day(df201802)  
df201803n = Data\_init\_day(df201803)  
df201804n = Data\_init\_day(df201804)  
df201805n = Data\_init\_day(df201805)  
df201806n = Data\_init\_day(df201806)  
df201807n = Data\_init\_day(df201807)  
df201808n = Data\_init\_day(df201808)  
df201809n = Data\_init\_day(df201809)  
df201810n = Data\_init\_day(df201810)  
df201811n = Data\_init\_day(df201811)  
df201812n = Data\_init\_day(df201812)

df201801n#查看函数使用成功与否

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead tr th {  
 text-align: left;  
}  
  
.dataframe thead tr:last-of-type th {  
 text-align: right;  
}

Quantity
label
5
6
AccessType
1
2
1
2
ToolType
41
42
43
41
42
43
41
42
43
41
42
43
Date
20180101
1.0
303.0
3071.0
2.0
323.0
1158.0
430.0
NaN
136.0
397.0
NaN
2247.0
20180102
3.0
281.0
4764.0
5.0
290.0
1799.0
1688.0
NaN
109.0
1545.0
2.0
2354.0
20180103
3.0
264.0
4463.0
2.0
265.0
1883.0
1945.0
1.0
97.0
1846.0
3.0
2370.0
20180104
1.0
472.0
4351.0
5.0
474.0
1939.0
2102.0
1.0
146.0
1991.0
3.0
2590.0
20180105
1.0
498.0
4532.0
3.0
471.0
2157.0
2205.0
NaN
197.0
2090.0
5.0
2842.0
20180106
1.0
373.0
3708.0
2.0
355.0
1559.0
1490.0
NaN
124.0
1640.0
4.0
2619.0
20180107
3.0
349.0
3653.0
2.0
348.0
1262.0
615.0
1.0
67.0
601.0
2.0
2837.0
20180108
2.0
414.0
4927.0
4.0
440.0
1870.0
1973.0
1.0
171.0
1757.0
4.0
2518.0
20180109
3.0
452.0
4518.0
2.0
452.0
1913.0
2057.0
2.0
123.0
1806.0
3.0
2389.0
20180110
4.0
366.0
4337.0
3.0
367.0
1905.0
2009.0
1.0
125.0
1891.0
4.0
2487.0
20180111
7.0
462.0
4546.0
4.0
460.0
2002.0
2172.0
1.0
106.0
1932.0
4.0
2647.0
20180112
4.0
415.0
4703.0
2.0
397.0
2132.0
2253.0
7.0
182.0
2086.0
3.0
2974.0
20180113
4.0
456.0
3768.0
6.0
430.0
1624.0
1631.0
5.0
127.0
1672.0
4.0
2546.0
20180114
4.0
351.0
3649.0
NaN
364.0
1384.0
650.0
NaN
91.0
639.0
2.0
2768.0
20180115
4.0
418.0
5172.0
2.0
442.0
1949.0
2148.0
NaN
248.0
1914.0
2.0
2543.0
20180116
5.0
465.0
4729.0
2.0
470.0
1956.0
2253.0
NaN
132.0
2021.0
3.0
2584.0
20180117
1.0
449.0
4675.0
2.0
441.0
1928.0
2240.0
1.0
114.0
1935.0
4.0
2804.0
20180118
2.0
417.0
4603.0
7.0
406.0
1966.0
2204.0
1.0
132.0
2020.0
4.0
2826.0
20180119
3.0
439.0
4749.0
2.0
425.0
2077.0
2207.0
2.0
154.0
2035.0
5.0
3069.0
20180120
4.0
331.0
4029.0
3.0
322.0
1660.0
1621.0
NaN
134.0
1700.0
3.0
2785.0
20180121
2.0
406.0
3715.0
1.0
441.0
1421.0
608.0
NaN
87.0
606.0
4.0
2696.0
20180122
5.0
465.0
5169.0
3.0
466.0
1936.0
2054.0
1.0
130.0
1799.0
3.0
2539.0
20180123
2.0
498.0
4761.0
5.0
496.0
1924.0
2090.0
1.0
122.0
1911.0
3.0
2713.0
20180124
6.0
437.0
4760.0
5.0
446.0
1988.0
2157.0
1.0
135.0
1873.0
4.0
2828.0
20180125
1.0
407.0
4755.0
4.0
396.0
1991.0
2172.0
1.0
125.0
1921.0
4.0
3012.0
20180126
2.0
497.0
5084.0
2.0
479.0
1993.0
2233.0
NaN
180.0
2051.0
4.0
3493.0
20180127
NaN
336.0
4316.0
NaN
319.0
1610.0
1638.0
2.0
132.0
1681.0
4.0
3141.0
20180128
2.0
317.0
4015.0
NaN
315.0
1386.0
630.0
1.0
113.0
575.0
3.0
3180.0
20180129
3.0
377.0
5301.0
2.0
381.0
1851.0
2061.0
1.0
184.0
1742.0
8.0
2894.0
20180130
3.0
398.0
5035.0
4.0
406.0
1931.0
2230.0
1.0
139.0
1848.0
3.0
2870.0
20180131
NaN
464.0
4806.0
5.0
456.0
1965.0
2171.0
NaN
136.0
1724.0
3.0
3002.0

#导出透视表excel，将数据交由合作同学，计算数据的相关性系数  
#对于python，亦可以处理相关性系数，参照python的math库  
df201801n.to\_excel('df201801n\_all.xlsx')  
df201802n.to\_excel('df201802n\_all.xlsx')  
df201803n.to\_excel('df201803n\_all.xlsx')  
df201804n.to\_excel('df201804n\_all.xlsx')  
df201805n.to\_excel('df201805n\_all.xlsx')  
df201806n.to\_excel('df201806n\_all.xlsx')  
df201807n.to\_excel('df201807n\_all.xlsx')  
df201808n.to\_excel('df201808n\_all.xlsx')  
df201809n.to\_excel('df201809n\_all.xlsx')  
df201810n.to\_excel('df2018010n\_all.xlsx')  
df201811n.to\_excel('df2018011n\_all.xlsx')  
df201812n.to\_excel('df2018012n\_all.xlsx')

#以上为day，以下为hour  
#这是最终提交文件，我实则已经处理过dfn的数据分析，发现不同月存在相似性，故研究“时”时，我们选取具有代表性的1，4，7，10月  
df201801 = df\_init(df201801\_init)  
df201804 = df\_init(df201804\_init)  
df201807 = df\_init(df201807\_init)  
df201810 = df\_init(df201810\_init)  
  
df201801t = Data\_init\_Hour(df201801)  
df201804t = Data\_init\_Hour(df201804)  
df201807t = Data\_init\_Hour(df201807)  
df201810t = Data\_init\_Hour(df201810)  
  
df201801t.to\_excel('df201801t\_all.xlsx')  
df201804t.to\_excel('df201804t\_all.xlsx')  
df201807t.to\_excel('df201807t\_all.xlsx')  
df201810t.to\_excel('df201810t\_all.xlsx')

# 全年合并

df2018 = pd.concat([df201801,df201802,df201803,df201804,df201805,df201806,df201807,df201808,df201809,df201810,df201811,df201812])

df2018#查看合并成功与否

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

工具类型
出入类型
出入时间
验放标志
0
41
1
20180101063226
6
1
43
2
20180101063231
6
2
43
2
20180101063246
6
3
41
1
20180101063310
6
4
43
2
20180101063342
5
...
...
...
...
...
387903
43
2
20181231235038
5
387904
43
2
20181231235249
5
387905
43
2
20181231235419
5
387906
43
2
20181231235702
5
387907
43
2
20181231235816
5

df2018n = Data\_init\_day(df2018)#如果报错，那就是多次点击了，需要重新初始化df

df2018n.to\_excel('df2018n\_all.xlsx')

# 生成图标，看看数据背后有什么关系

### 全年热力图，折线图和棒状图

sns.heatmap(df2018n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

df2018n.plot()

<AxesSubplot:xlabel='Date'>

df2018n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

### 1-12月数据精确到天的热力图

sns.heatmap(df201801n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201802n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201803n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201804n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201805n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201806n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201807n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201808n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201809n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201810n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201811n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201812n.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

### 1，4，7和10月精确至小时的热力图

sns.heatmap(df201801t.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201804t.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201807t.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201810t.T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

### 1-12月数据精确到天的折线图

从折线图可以看出除2，10月有较大不同，其余月份数据特征大体一致  
查阅日历，不难看出因为2月农历新年，10月国庆节导致数据出现异化特征

df201801n.plot()

<AxesSubplot:xlabel='Date'>

df201802n.plot()

<AxesSubplot:xlabel='Date'>

df201803n.plot()

<AxesSubplot:xlabel='Date'>

df201804n.plot()

<AxesSubplot:xlabel='Date'>

df201805n.plot()

<AxesSubplot:xlabel='Date'>

df201806n.plot()

<AxesSubplot:xlabel='Date'>

df201807n.plot()

<AxesSubplot:xlabel='Date'>

df201808n.plot()

<AxesSubplot:xlabel='Date'>

df201809n.plot()

<AxesSubplot:xlabel='Date'>

df201810n.plot()

<AxesSubplot:xlabel='Date'>

df201811n.plot()

<AxesSubplot:xlabel='Date'>

df201812n.plot()

<AxesSubplot:xlabel='Date'>

### 1，4，9和10月数据精确到小时的折线图

不难看出，每天的数据特征大体一致

df201801t.plot()

<AxesSubplot:xlabel='Date'>

df201804t.plot()

<AxesSubplot:xlabel='Date'>

df201807t.plot()

<AxesSubplot:xlabel='Date'>

df201810t.plot()

<AxesSubplot:xlabel='Date'>

### 1-12月数据精确到天的棒状图

折线图可以看出第一线在时间上的变化规律，那么***棒状图***可以看出内部的比列

df201801n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201802n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201803n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201804n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201805n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201806n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201807n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201808n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201809n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201810n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201811n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201812n.plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

### 1，4，7，10数据精确到小时的棒状图

df201801t.plot(kind = 'bar',stacked =True)#似乎数据太多，导致有些糊了

<AxesSubplot:xlabel='Date'>

df201804t.plot(kind = 'bar',stacked =True)#似乎数据太多，导致有些糊了

<AxesSubplot:xlabel='Date'>

df201807t.plot(kind = 'bar',stacked =True)#似乎数据太多，导致有些糊了

<AxesSubplot:xlabel='Date'>

df201810t.plot(kind = 'bar',stacked =True)#似乎数据太多，导致有些糊了

<AxesSubplot:xlabel='Date'>

### 7月1，11，21天的小时图，数据趋势大体一致

这么糊看不了啊，精确到小时的数据有30天X18小时的数据，我们切片出一天看看热力图，折线图和棒状图（只看7月的1,11,21天）

sns.heatmap(df201801t.iloc[0:18,:].T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201801t.iloc[180:198,:].T)

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

sns.heatmap(df201801t.iloc[361:379,:].T)#数据多了一小时，这可能与港口加班等情况有关

<AxesSubplot:xlabel='Date', ylabel='None-label-AccessType-ToolType'>

df201801t.iloc[0:18,:].plot()

<AxesSubplot:xlabel='Date'>

df201801t.iloc[180:198,:].plot()

<AxesSubplot:xlabel='Date'>

df201801t.iloc[361:379,:].plot()

<AxesSubplot:xlabel='Date'>

df201801t.iloc[0:18,:].plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201801t.iloc[180:198,:].plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

df201801t.iloc[361:379,:].plot(kind = 'bar',stacked =True)

<AxesSubplot:xlabel='Date'>

### 其它

将201801的数据按不同特征分成三个透视表，并生成excel

df201801 = df201801\_init.drop(df201801\_init.columns[[1,2,3,4,5,7,8]],axis = 1)  
s201801 = df201801['出入时间'].map(ConvertToDay).tolist()  
df201801.insert(0,'出入日期',s201801)  
df201801 = df201801.drop(df201801.columns[3],axis = 1)  
df201801 = df201801.rename(columns = {'出入日期':'Date','工具类型':'ToolType','出入类型':'AccessType','验放标志':'label'})#传入字典  
df201801.drop(index = df201801[df201801['ToolType']>43].index,inplace=True)  
df201801.drop(index = df201801[df201801['label']>6].index,inplace=True)  
df201801.drop(index = df201801[df201801['label']<5].index,inplace=True)  
df201801

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

Date
ToolType
AccessType
label
0
20180101
41
1
6
1
20180101
43
2
6
2
20180101
43
2
6
3
20180101
41
1
6
4
20180101
43
2
5
...
...
...
...
...
417830
20180131
43
2
5
417831
20180131
43
1
5
417832
20180131
41
2
6
417833
20180131
43
2
5
417834
20180131
41
2
6

df201801\_tool = df201801.drop(df201801.columns[[2,3]],axis = 1)  
df201801\_tool

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

Date
ToolType
0
20180101
41
1
20180101
43
2
20180101
43
3
20180101
41
4
20180101
43
...
...
...
417830
20180131
43
417831
20180131
43
417832
20180131
41
417833
20180131
43
417834
20180131
41

df201801\_access = df201801.drop(df201801.columns[[1,3]],axis = 1)  
df201801\_access

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

Date
AccessType
0
20180101
1
1
20180101
2
2
20180101
2
3
20180101
1
4
20180101
2
...
...
...
417830
20180131
2
417831
20180131
1
417832
20180131
2
417833
20180131
2
417834
20180131
2

df201801\_label = df201801.drop(df201801.columns[[1,2]],axis = 1)  
df201801\_label

.dataframe tbody tr th:only-of-type {
vertical-align: middle;
}</body>

.dataframe tbody tr th {  
 vertical-align: top;  
}  
  
.dataframe thead th {  
 text-align: right;  
}

Date
label
0
20180101
6
1
20180101
6
2
20180101
6
3
20180101
6
4
20180101
5
...
...
...
417830
20180131
5
417831
20180131
5
417832
20180131
6
417833
20180131
5
417834
20180131
6

s201801\_tool = df201801\_tool.value\_counts(sort=False)  
df201801n\_tool = DataFrame(s201801\_tool)  
df201801n\_tool = df201801n\_tool.rename(columns = {0:'Quantity'})  
df201801n\_tool = df201801n\_tool.unstack()  
df201801n\_tool.head()  
df201801n\_tool.to\_excel('df201801n\_tool.xlsx')

s201801\_access = df201801\_access.value\_counts(sort=False)  
df201801n\_access = DataFrame(s201801\_access)  
df201801n\_access = df201801n\_access.rename(columns = {0:'Quantity'})  
df201801n\_access = df201801n\_access.unstack()  
df201801n\_access.head()  
df201801n\_access.to\_excel('df201801n\_access.xlsx')

s201801\_label = df201801\_label.value\_counts(sort=False)  
df201801n\_label = DataFrame(s201801\_label)  
df201801n\_label = df201801n\_label.rename(columns = {0:'Quantity'})  
df201801n\_label = df201801n\_label.unstack()  
df201801n\_label.head()  
df201801n\_label.to\_excel('df201801n\_label.xlsx')

## 重庆邮电大学2021数模A题第一问 有关python数据可视化的分析 代码编写：江佳骏