百货商场用户画像描绘与价值分析报告

# 数据准备

使用python进行数据分析时，要先将数据导入。数据存在的形式多样，有文件（CSV、Excel、TXT）和数据库（SQLServer、MySQL）等形式。

这里以CSV文件为例，导入（读取）文件名为'cumcm2018c1的excel数据和cumcm2018c2的csv数据，并且可以查看一下导入的数据是否有误。

|  |
| --- |
| import numpy as np #导入numpy库  import pandas as pd #导入pandas库  data1=pd.read\_excel('cumcm2018c1.xlsx') #导入数据  data2=pd.read\_csv('cumcm2018c2.csv')  print(data1.head()) # 查看前5行数据  data2.head() |

运行可得如下结果：



# 数据处理

数据清洗就是处理缺失数据以及清除毫无意义的信息。通过查看数据的各个属性来发现数据中存在的缺失值、异常值等。

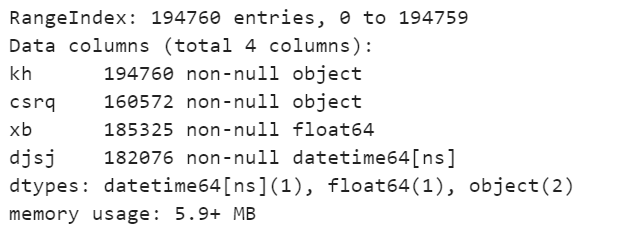
|  |
| --- |
| # 查看数据的形状  print("data1数据的形状：", data1.shape)  print("data2数据的形状：", data2.shape)  # 查看每列数据的非空值的个数和数据类型  print(" data1每列数据的非空值的个数和数据类型：\n", data1.info())  print(" data2每列数据的非空值的个数和数据类型：\n", data2.info())  # 查看每列数据的描述统计信息  print("data1数据的描述统计信息：\n", data1.describe())  print("data2数据的描述统计信息：\n", data2.describe())  # 查看每列数据的空值个数  print("data1每列数据的空值个数:\n", data1.isnull().sum())  print("data2每列数据的空值个数:\n", data2.isnull().sum()) |

运行可得如下结果：

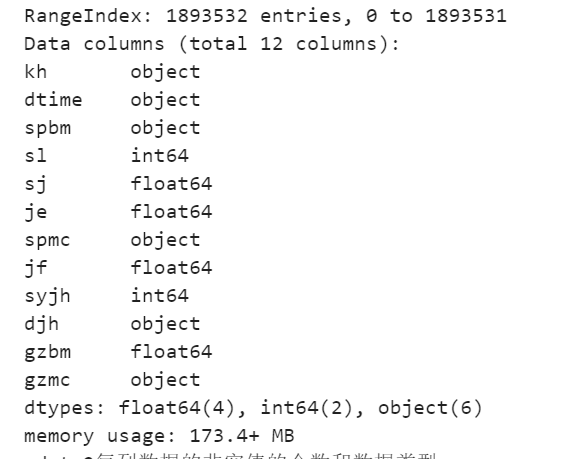
data1数据的形状：(194760, 4)

data2数据的形状：(1893532，12)

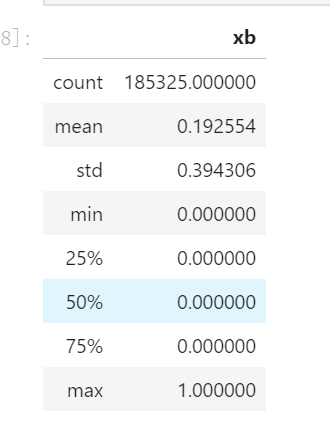
data1每列数据的非空值的个数和数据类型：



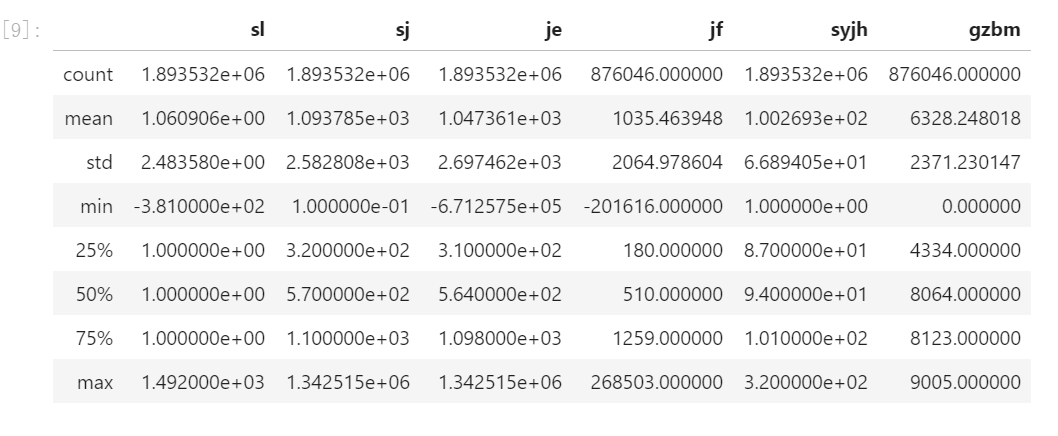
data2每列数据的非空值的个数和数据类型：



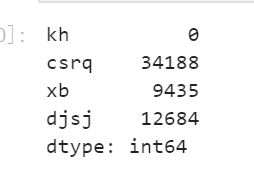
data1数据的描述统计信息：



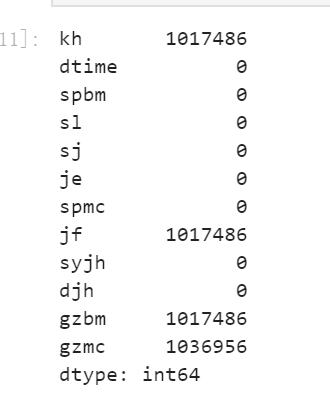
data2数据的描述统计信息：



data1每列数据的空值个数:



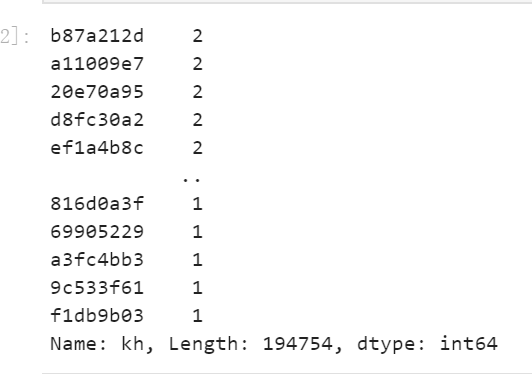
data2每列数据的空值个数:



使用value\_counts()函数可以查看数据中每个值的个数，以data1中卡号“kh”列为例：

|  |
| --- |
| # 查看卡号（kh）列数据中每个值的个数  data1.kh.value\_counts() |

运行代码可得如下结果：



通过查看data1和data2数据的各个属性可以看出，数据中存在缺失值和逻辑错误的数据，因此需要对数据进行清洗。

## 重复值处理。

drop\_duplicates()：把数据结构中行相同的数据去除（保留其中一行）

|  |
| --- |
| # 数据去重  new\_data1 = data1.drop\_duplicates() #data1  print(new\_data1.shape)  new\_data2 = data2.drop\_duplicates() #data2  print(new\_data2.shape) |

运行代码可得到：

|  |
| --- |
| (194760, 4)  (1893532，12) |

由结果可看到，data1数据形状由(194760, 4)变更为(194760, 4），data2数据形状由(1893532，12)变更为(1893532，12)，说明数据中不存在重复值。

## 缺失值处理。

对于缺失值的处理方式，有数据补齐、删除对应行、不处理等方法。

dropna()：去除数据结构中值为空的数据行。

|  |
| --- |
| # 去除数据中的空值  data1\_drop = new\_data1.dropna()  print(data1\_drop.shape)  data2\_drop = new\_data2.dropna()  print(data2\_drop.shape) |

运行代码可得到：

|  |
| --- |
| **（153492，4）**  **（856576，12）** |

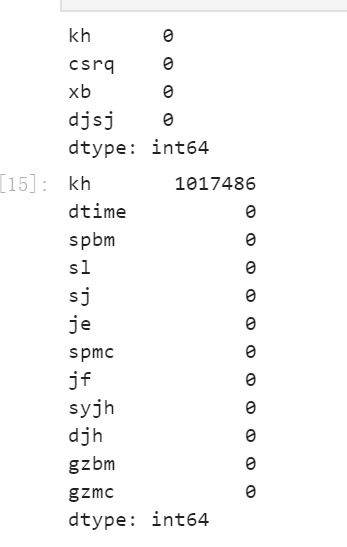
可以看到data1数据形状由(194760, 4)变更为(153492，4)，data2数据形状由(1893532，12)变更为(856576，12)，说明使用dropna()已经将数据中值为空的数据行去除了。但是缺失值的数据占总数据的总比很大，所以不能直接删除。

fillna()：用其他数值代替数据中的空值。

interpolate()：使用线性差值对数据进行填充

|  |
| --- |
| # 数值型：通常用均值，中位数和众数等代替  # 类别型：众数代替  #对于data1会员信息表，csrq、xb和djsj均是类别型数据，所以都用众数来替换缺失值  for i in ['csrq','xb','djsj']:  new\_data1[i]=new\_data1[i].fillna(new\_data1[i].value\_counts().index[0])  print(new\_data1.isnull().sum())  #将gzbm、gzmc用众数来替换缺失值，jf用平均数来替换缺失值  for j in ['gzbm','gzmc']:  new\_data2[j]=new\_data2[j].fillna(new\_data2[j].value\_counts().index[0])  new\_data2['jf']=new\_data2['jf'].fillna(new\_data2['jf'].mean())  new\_data2.isnull().sum() |

运行代码可得到：



具体使用哪种缺失值处理方法，要根据具体情况决定。就data1和data2数据而言，缺失值的数据占总数据的总比很大，直接删除会对后面进行分析产生影响。

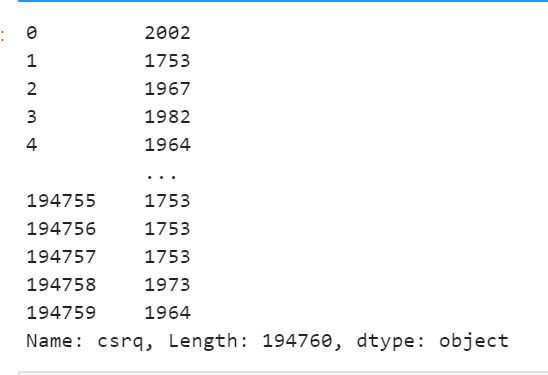
## 数据抽取

**字段抽取**是根据已知的列数据的开始和结束位置，抽取出新的列。字段抽取的函数是：slice(start, stop)，其中 start 是开始位置，stop 是结束位置。

new\_data1中的出生日期（csrq）字段有不符常理的时间，用slice()函数作用在字符型数据上的，提取一下csrq的年份，抽取1900年到2018年出生的用户，所以需要先将列数据转换一下，：

|  |
| --- |
| # 字段抽取  new\_data1["csrq"] =new\_data1["csrq"].astype(str)  new\_data1["csrq"].str.slice(0, 4) |

运行可得到：



**记录抽取**是根据一定条件对数据进行抽取。函数是 dataframe[condition]，其中 condition 是过滤的条件，返回值类型是 DataFrame。

常用的condition类型如下：

比较运算：>, <, >=, <=, !=

范围运算：between(left, right)

空值匹配：pandas.isnull(column)

字符匹配：str.contains(patten, na=False)

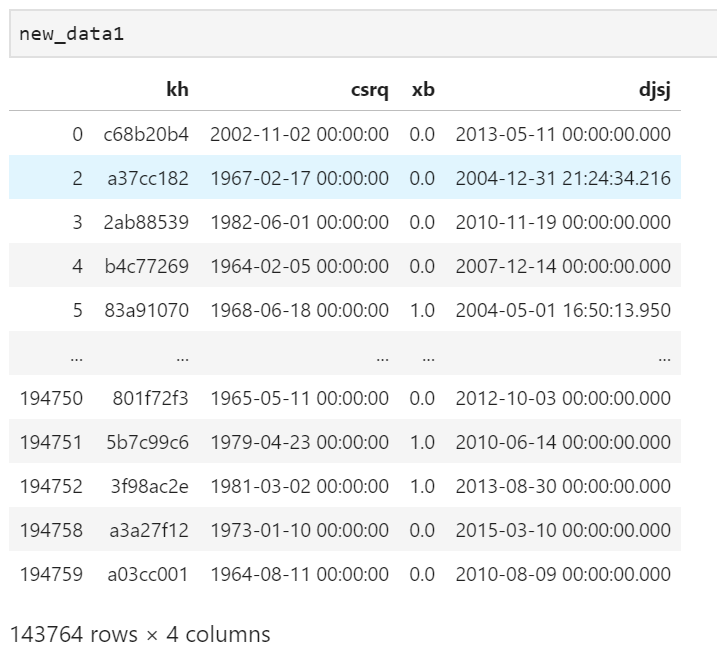
逻辑运算：与（&），或（|），取反（not）

|  |
| --- |
| # 按条件抽取数据(异常值处理)  # 抽取new\_data1出生日期1900年1月1日到2018年1月1日的数据  new\_data1 =new\_data1[(new\_data1.csrq<="20180101") & (new\_data1.csrq>="19000101")]  new\_data1= new\_data1[new\_data1['csrq']!=20150229] # new\_data1抽取出生日期不为2月29日的数据  # 抽取new\_data2销售日期2015年2月1日到2015年2月28日的数据  new\_data2= new\_data2[(new\_data2.dtime<="20181230") & (new\_data2.dtime>="20150101")]  new\_data2= new\_data2[new\_data2['sl']>0] # 抽取new\_data2销售数量大于0的数据  new\_data2= new\_data2[new\_data2['dtime']!=20150229] # new\_data2抽取销售日期不为2月29日的数据 |

## 数据合并

**记录合并**是指两个结构相同的数据框合并成一个数据框，也就是在一个数据框中追加另一个数据框的数据记录。函数是 pandas.concat([dataframe1, dataframe2,…])，其中dataframe1为数据框，返回值类型是 DataFrame。

new\_data1数据：

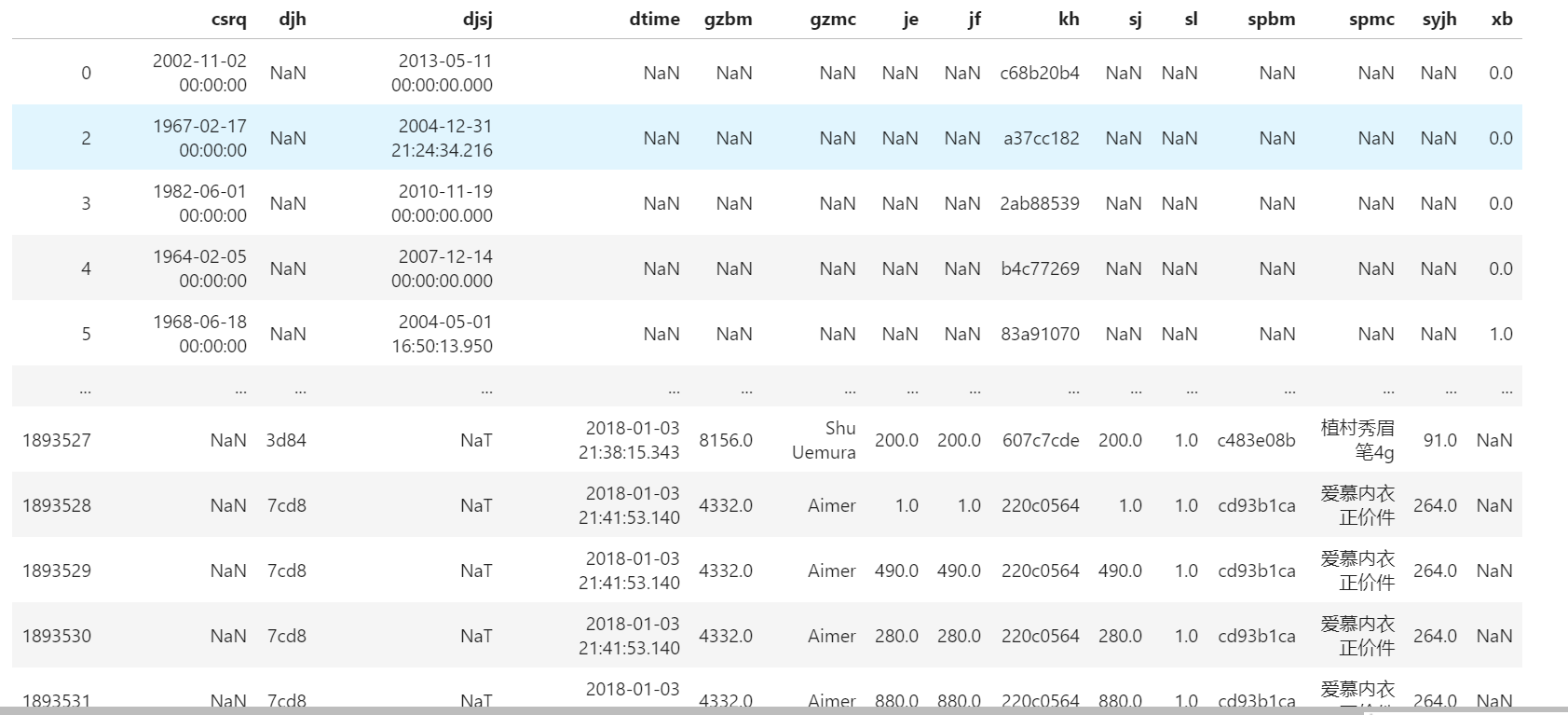


data2数据：



|  |
| --- |
| # 普通合并new\_data1、new\_data2数据  pd.concat([new\_data1, new\_data2]) |

运行代码可得到：

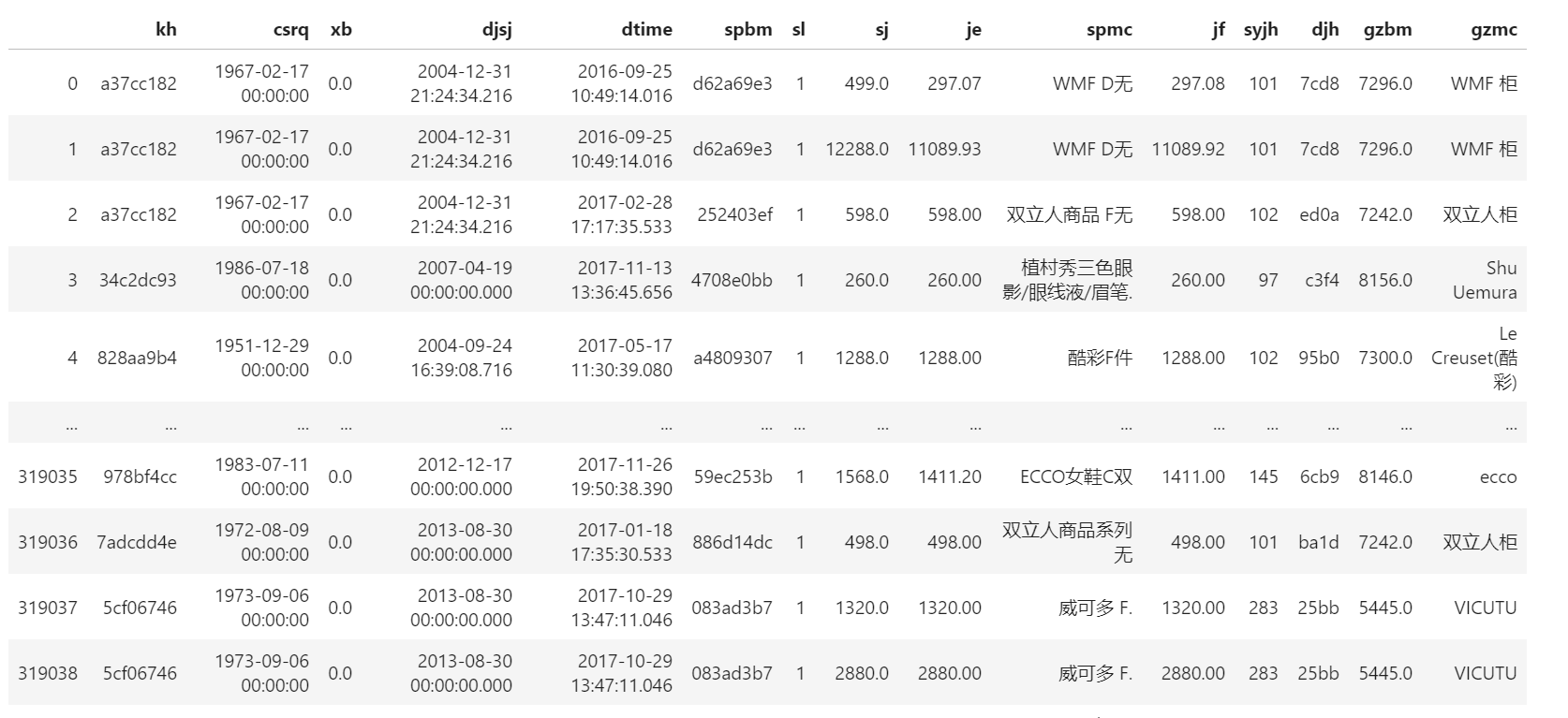


**字段匹配**是指不同结构的数据框（两个或两个以上的数据框），按照一定的条件进行合并。函数是 merge(x, y, left\_on, right\_on, how="inner")，x是第一个数据框，y是第二个数据框，left\_on是第一个数据框用于匹配的列，right\_on是第二个数据框用于匹配的列，how是数据框合并的方式，默认是“inner”，即内连接，合并的结果是两个数据的交集。常用的数据框合并方式还有左外连接（“left”）、右外连接（“right”）、全外连接（“outer”）。

本题是将new\_data1和new\_data2两张数据表进行外连接。

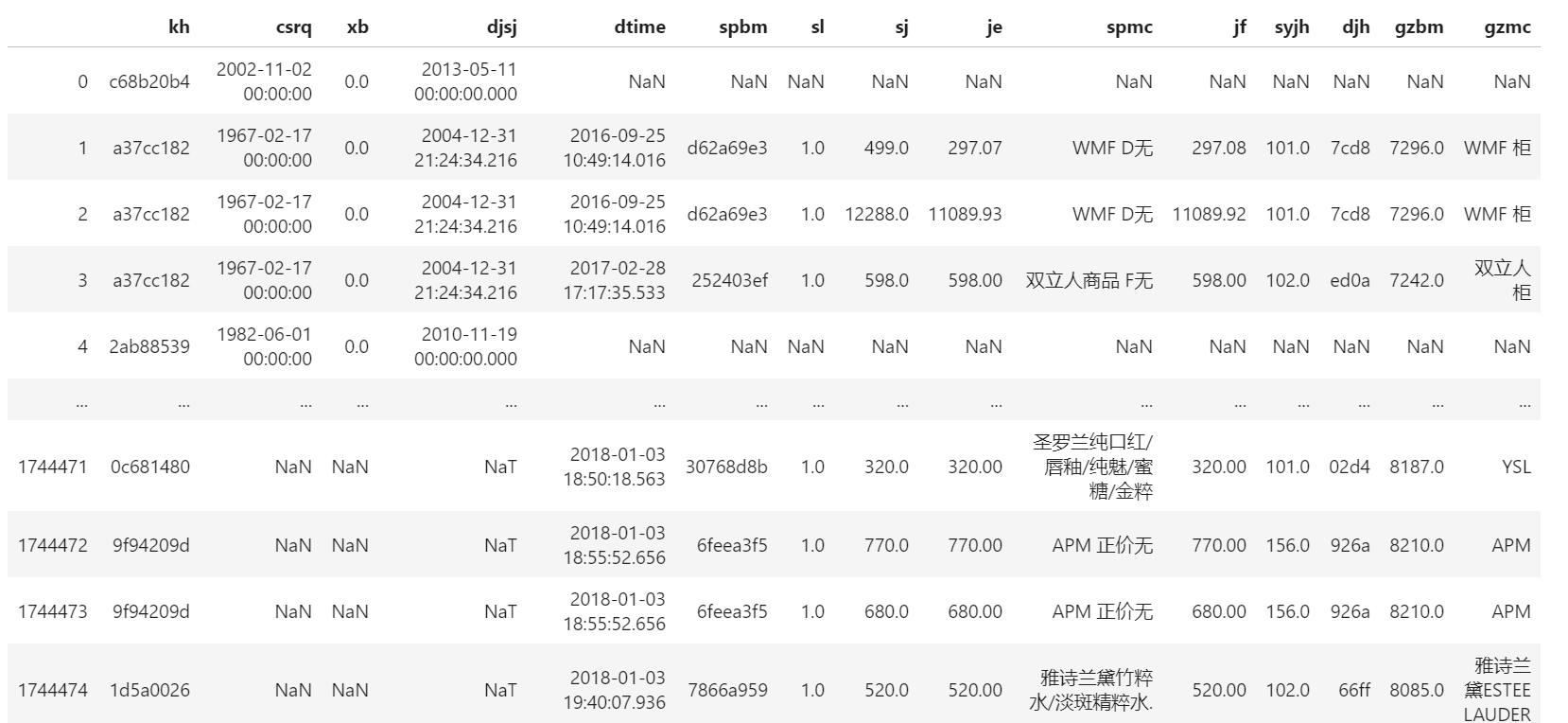
|  |
| --- |
| data\_inner = pd.merge(new\_data1, new\_data2, how="inner")  data\_inner #内连接 |

运行代码可得到：



|  |
| --- |
| data\_outer = pd.merge(new\_data1, new\_data2, how="outer")  data\_outer #外连接 |

运行代码可得到：



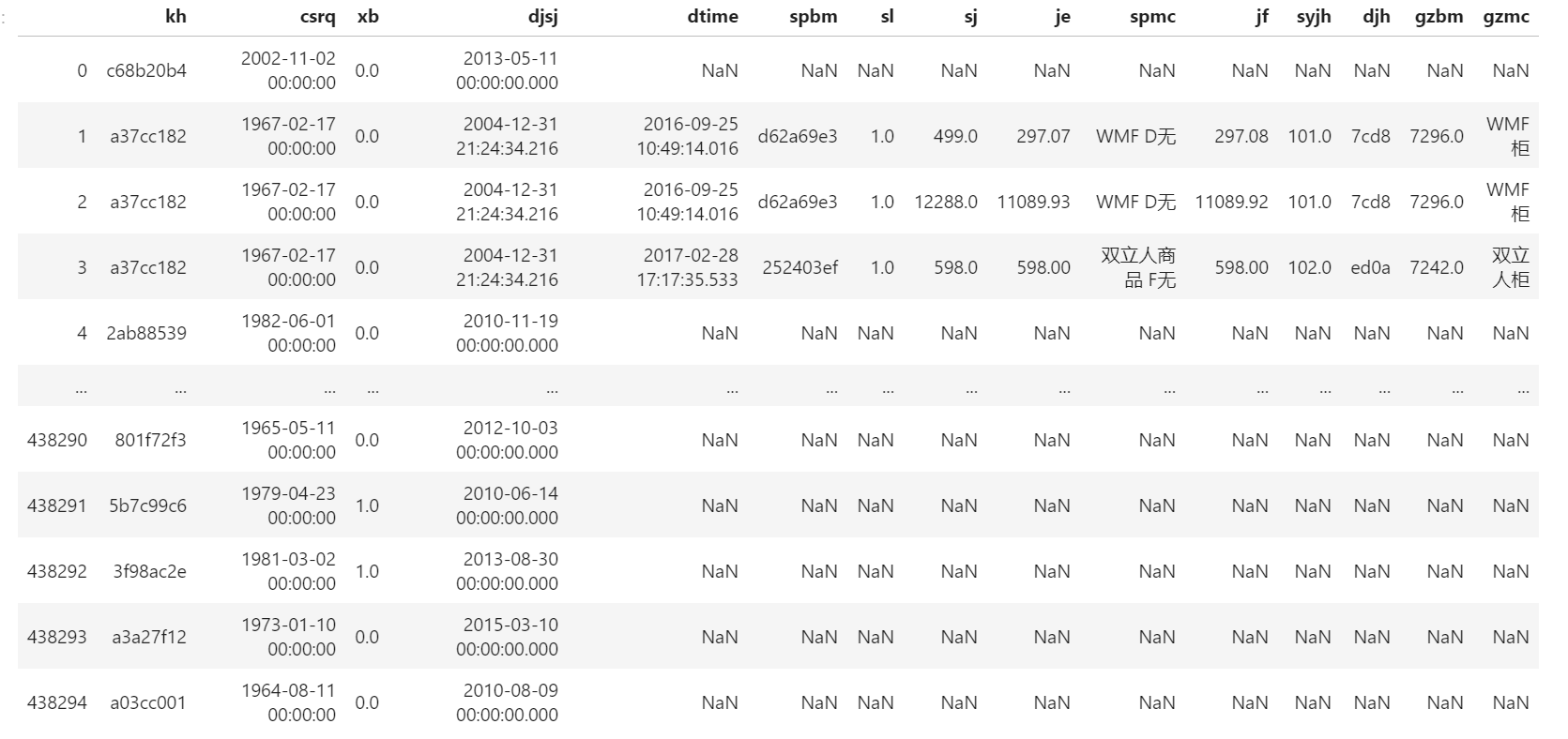
|  |
| --- |
| data\_right = pd.merge(new\_data1, new\_data2, how="right")  data\_right #右外连接 |

运行代码可得到：



|  |
| --- |
| data\_left = pd.merge(new\_data1, new\_data2, how="left")  data\_left #左外连接 |

运行代码可得到：



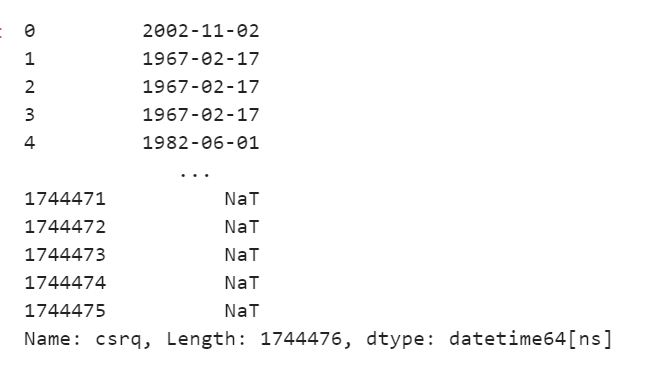
## 日期处理

to\_datetime(dateString, format)：将字符型的日期格式转换为日期格式数据。

以总表中用户出生日期（csrq），会员入会登记时间（djsj），消费产生时间（dtime）列数据为例：

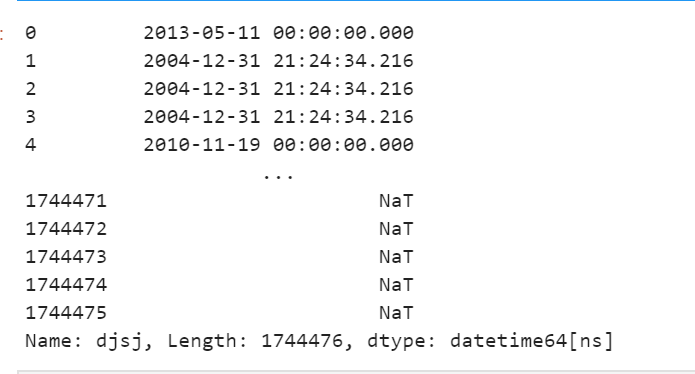
|  |
| --- |
| # 日期转换  data\_outer["csrq"] = pd.to\_datetime(data\_outer.csrq)  data\_outer.csrq |

运行代码可得到：



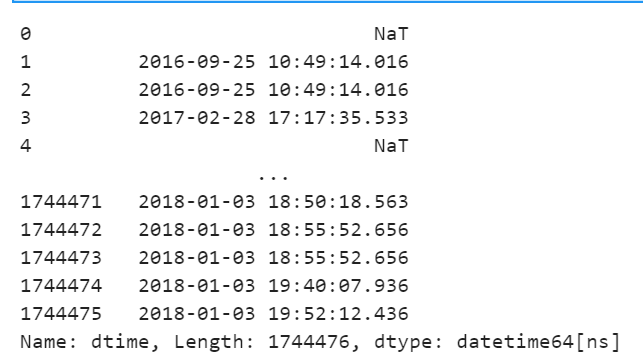
|  |
| --- |
| # 日期转换  data\_outer["djsj"] = pd.to\_datetime(data\_outer.djsj)  data\_outer.djsj |

运行代码可得到：



|  |
| --- |
| # 日期转换  data\_outer["dtime"] = pd.to\_datetime(data\_outer.dtime)  data\_outer.dtime |

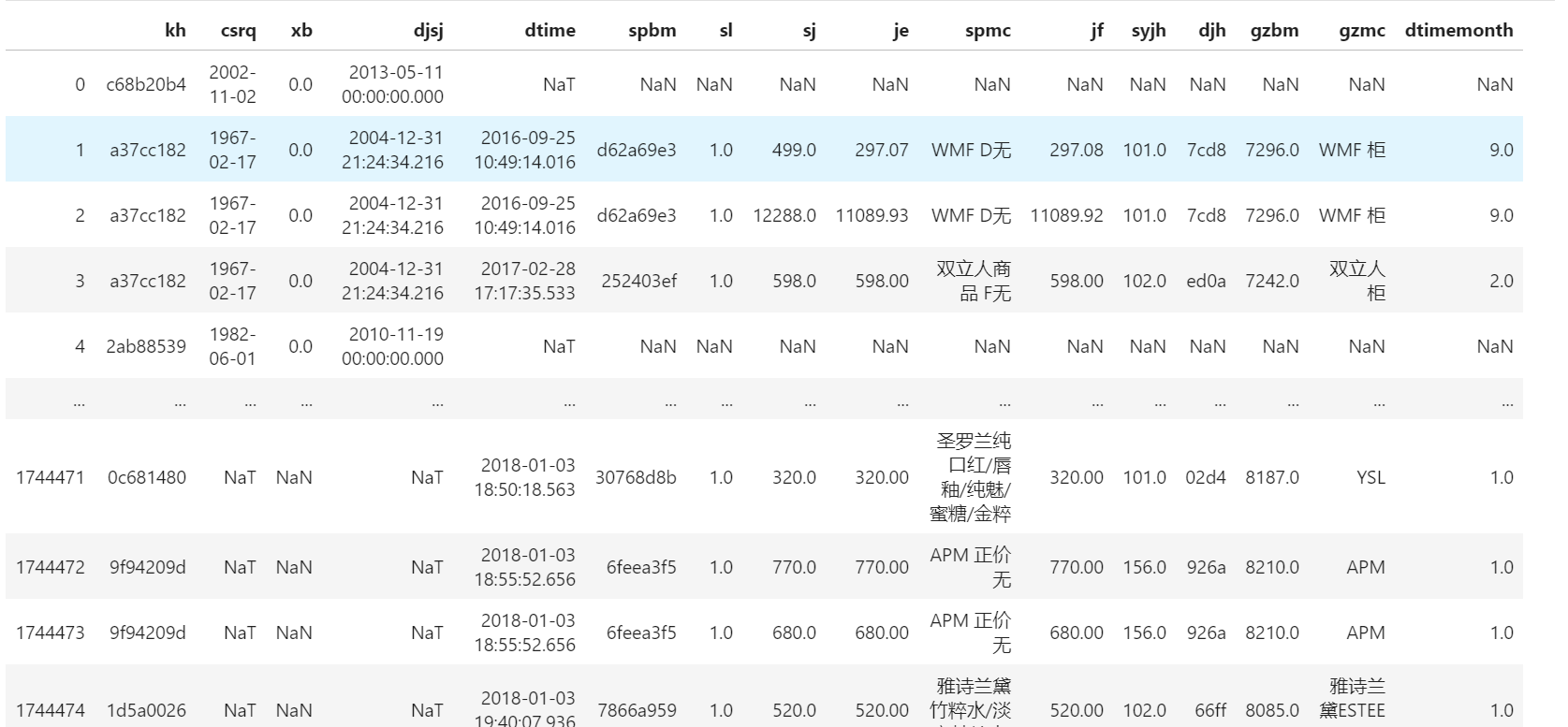
运行代码可得到：



从日期格式数据中可以抽取出需要的部分属性，比如可以抽取消费产生时间（dtime）列数据的月份信息：

|  |
| --- |
| # 日期抽取,并增加一列  data\_outer['dtimemonth']=data\_outer.dtime.dt.month  data\_outer |

运行代码可得到：



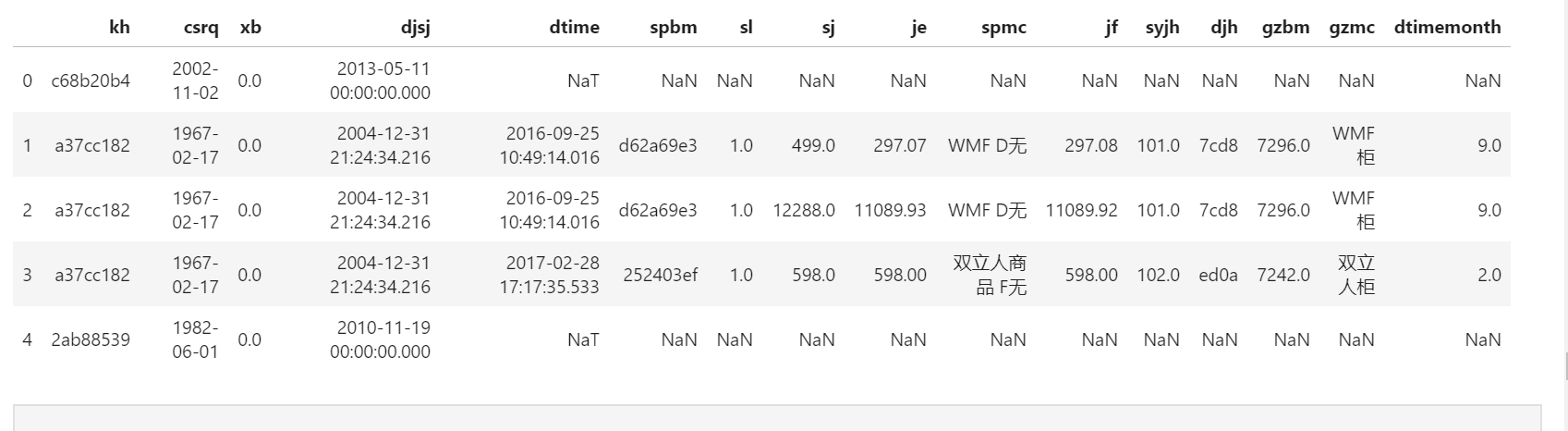
# 分析问题

## 会员年龄分段并绘制占比饼图

1、提取出会员表

|  |
| --- |
| #会员表  huiyuan=data\_outer[data\_outer['kh'].notnull()==True]  huiyuan.head() |

运行代码可得到：



2、将会员的年龄算出（利用datetime知识）

|  |
| --- |
| #将会员的年龄添加到表中  csrq=pd.DataFrame(huiyuan['csrq'])  csrq['csrq'] =pd.to\_datetime(huiyuan['csrq'])  import datetime as dt  now\_year=dt.datetime.today().year  huiyuan['age']=now\_year-csrq['csrq'].dt.year #增加一列age到huiyuan表中  huiyuan |

运行代码可得到：



3、将会员按年龄划分为老年、中年、青年，绘制会员年龄占比饼图，利用pandas.cut函数。

分布分析是根据分析的目的，将数据（定量数据）进行等距或不等距的分组，是研究各组分布规律的一种分析方法。常用的划分数据的函数：

|  |
| --- |
| cut(series, bins, right=True, labels=NULL) |

其中series为需要分组的数据，bins为分组的依据数据，right为分组的时候右边是否闭合，labels为分组的自定义标签，可以不自定义。

功能:将数据进行离散化

pandas.cut(x,bins,right=True,labels=None,retbins=False,precision=3,include\_lowest=False)

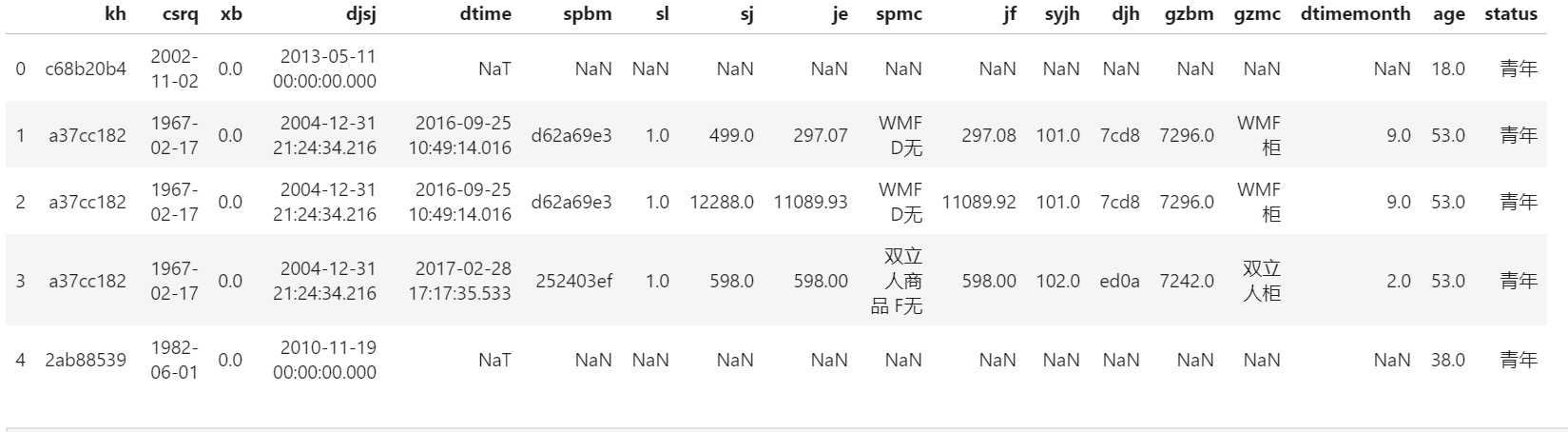
     参数说明:

     x   :进行划分的一维数组

     bins : 1,整数---将x划分为多少个等间距的区间

|  |
| --- |
| #将会员按年龄划分为老年、中年、青年，绘制会员年龄占比饼图  huiyuan['status']=pd.cut(huiyuan['age'],bins=[17,65,79,99],labels=['青年','中年','老年'],)  huiyuan.head() |

运行代码可得到：

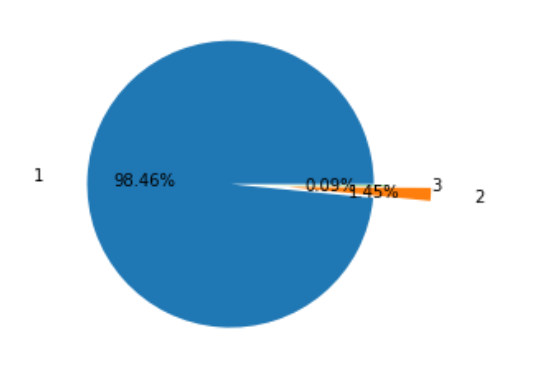


4、绘制会员年龄占比饼图

利用matplotlib中的pie函数

|  |
| --- |
| import matplotlib.pyplot as plt  #绘制会员年龄占比饼图  a=huiyuan[huiyuan['status']=='青年']['status'].value\_counts().loc['青年']  b=huiyuan[huiyuan['status']=='中年']['status'].value\_counts().loc['中年']  c=huiyuan[huiyuan['status']=='老年']['status'].value\_counts().loc['老年']  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=[a,b,c]  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.3,0], labels=['1','2','3'], labeldistance=1.3)  #数值显示情况：autopct='%.2f%%'表示显示两位小数，%%显示%，第一个%表示转译  #explode=[0.1,0,0] 每一份的尖端离圆心的距离，突出显示作用  #设置标签：labels=['1','2','3']  #标签离圆心的距离：labeldistance=1.3 圆半径默认为1  plt.show() |

运行代码可得到：



## 对会员统计不同年龄的消费金额并可视化

分组分析是根据分组字段将分析对象划分成不同的部分，以对比分析各组之间差异性的一种分析方法。常用的形式：

|  |
| --- |
| DataFrame.groupby(by=["分类1","分类2",…])["被统计的列"].agg({"列别名1":统计函数1,…}) |

by—用于分组的列。

[ ]—用于统计的列。

1、统计不同年龄的会员用户的销售金额：

|  |
| --- |
| #针对不同会员（kh），统计不同年龄的消费金额，并可视化  #以年龄分类  import numpy as np  huiyuan['kh']=huiyuan['kh'].apply(str)  jine=huiyuan.groupby(by=["age"])["je"].agg({"不同年龄会员的总销售金额":np.sum})  jine |

运行代码可得到：

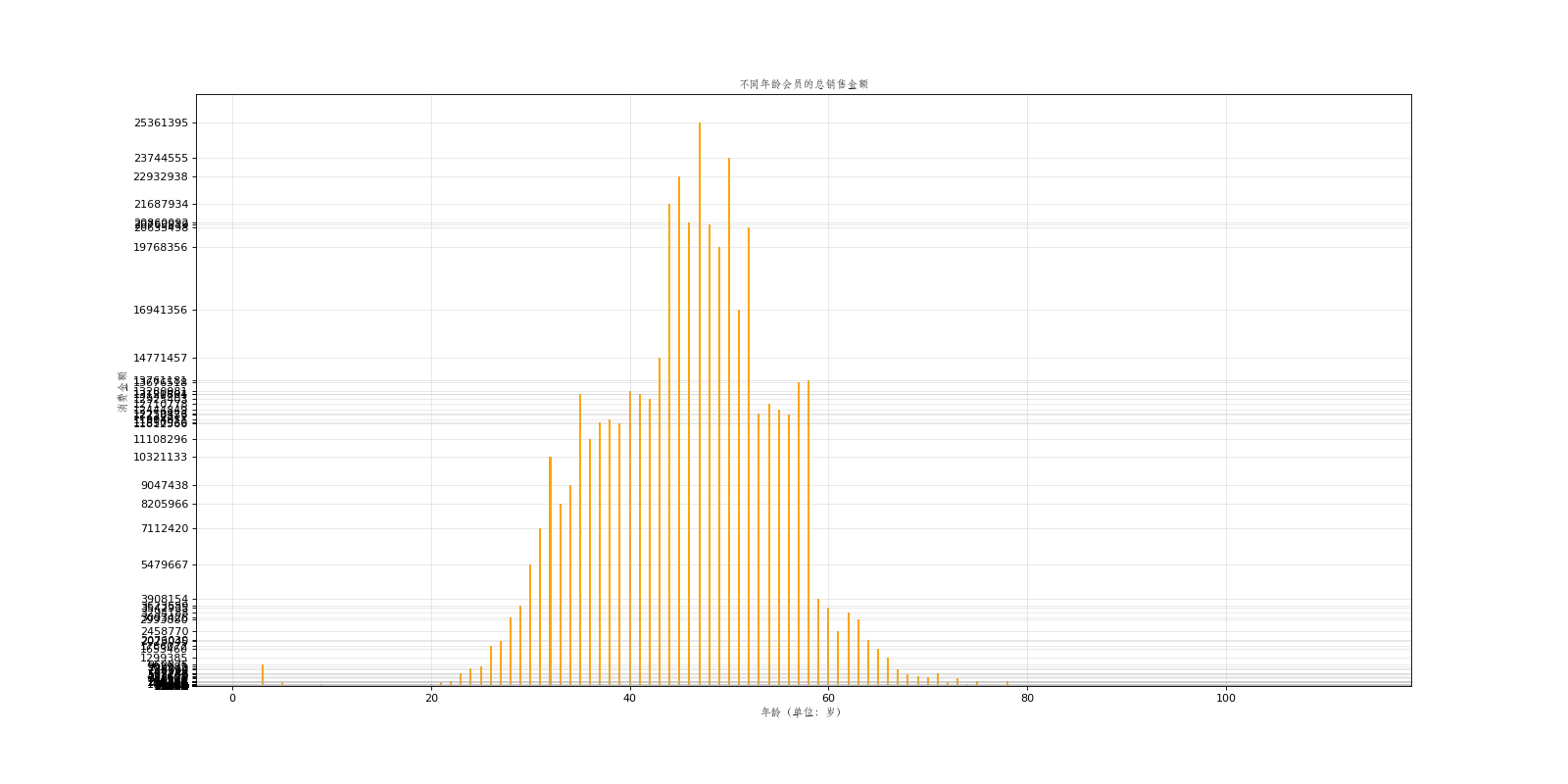


2、绘制不同年龄会员消费金额占比饼图

利用matplotlib中的pie函数

|  |
| --- |
| #绘制不同年龄会员的总销售金额条形图  from matplotlib import pyplot as plt  from matplotlib import font\_manager  #设置中文  my\_font=font\_manager.FontProperties(fname="C:/Windows/Fonts/STFANGSO.ttf")  #调节图片大小  fig=plt.figure(figsize=(20,10),dpi=80)  x=jine.index #将年龄设为x  y=jine['不同年龄会员的总销售金额'] #将消费总额设为y  plt.bar(x,y,width=0.2,color="orange")  #设置标题  plt.title('不同年龄会员的总销售金额',fontproperties=my\_font)  plt.yticks(y)  #设置网格，使其更加醒目 透明度为0.3  plt.grid(True,alpha=0.3) #也可这样写plt.grid(alpha=0.3)  #添加描述信息  plt.xlabel("年龄（单位：岁）",fontproperties=my\_font)  plt.ylabel('消费金额',fontproperties=my\_font)  #保存图片  plt.savefig("./jine.png")  #展示  plt.show() |

运行代码可得到：

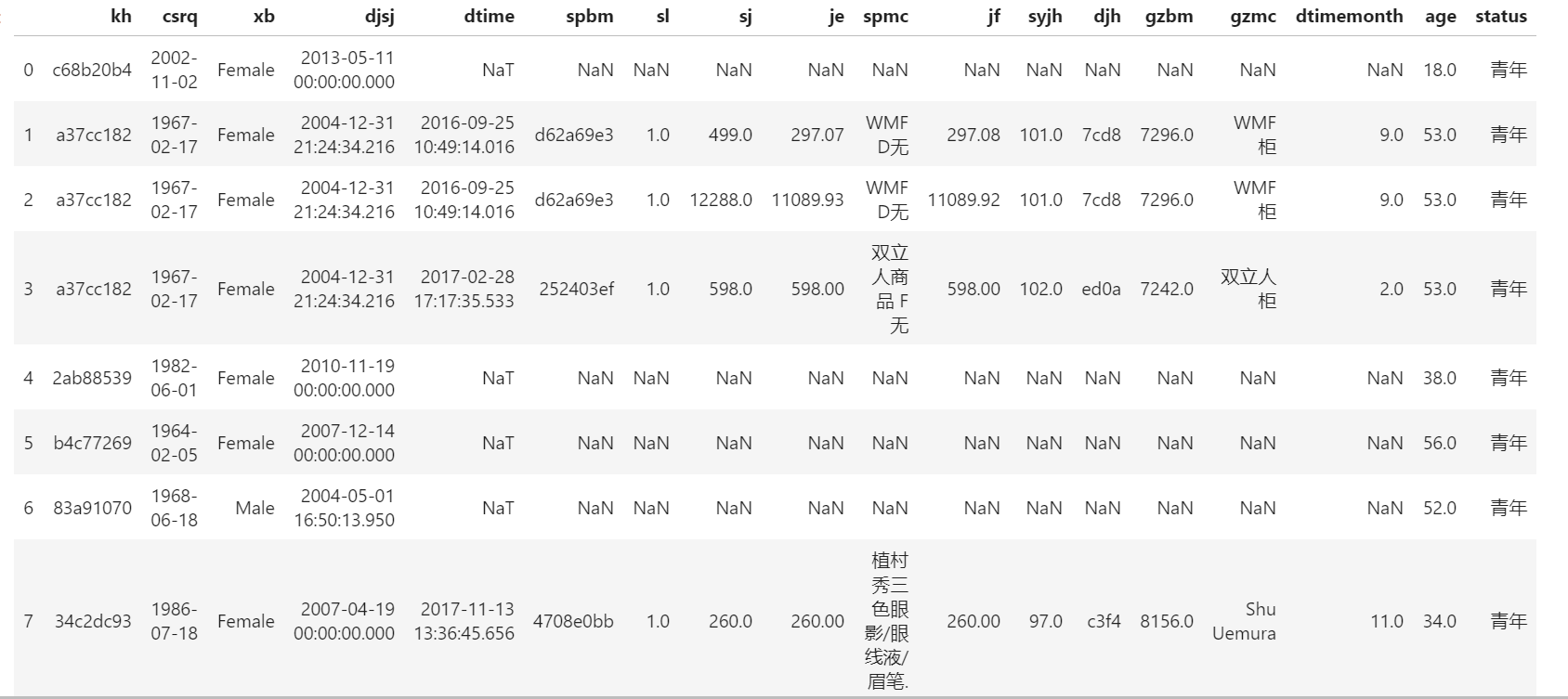


## 统计会员性别人数比例和不同性别消费金额比例并进行可视化

1. 将性别列将数值型改为类别型

|  |
| --- |
| #将性别列将数值型改为类别型  huiyuan['xb']=huiyuan['xb'].astype('category')  huiyuan['xb'].cat.categories=['Female','Male'] #0女性 1男性  huiyuan.head(10) #查看数据前十行 |

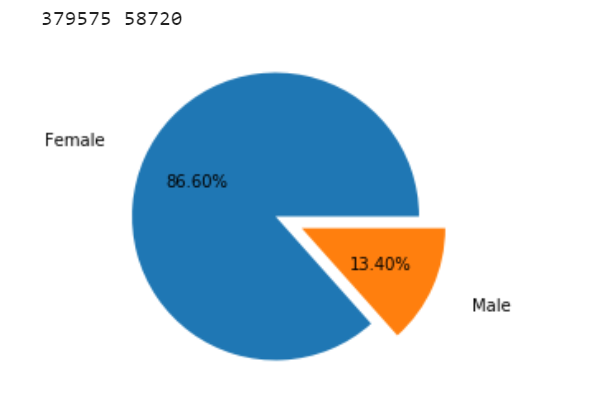
运行代码可得到：



1. 不同性别人数比例及可视化

|  |
| --- |
| female=huiyuan[huiyuan['xb']=='Female']['xb'].value\_counts().loc['Female'] #女性人数  male=huiyuan[huiyuan['xb']=='Male']['xb'].value\_counts().loc['Male'] #男性人数  print(female,male)  #绘制男女比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=[female,male]  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1], labels=['Female','Male'], labeldistance=1.3)  #数值显示情况：autopct='%.2f%%'表示显示两位小数，%%显示%，第一个%表示转译  #explode=[0.1,0,0] 每一份的尖端离圆心的距离，突出显示作用  #设置标签：labels=['1','2','3']  #标签离圆心的距离：labeldistance=1.3 圆半径默认为1  plt.show() |

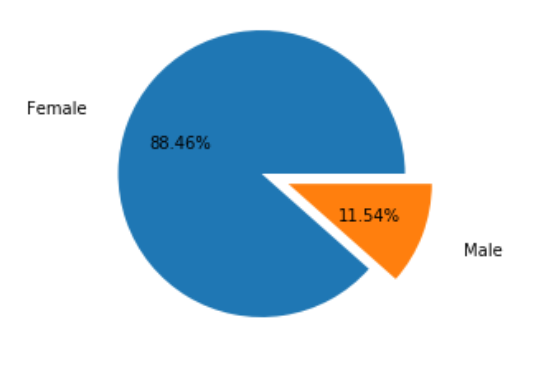
运行代码可得到：



1. 不同性别消费金额比例并进行可视化

|  |
| --- |
| #不同性别消费金额比例并进行可视化  #以性别分组  jine\_xb=huiyuan.groupby(by=["xb"])["je"].agg({"不同性别会员的总销售金额":np.sum})  #绘制男女消费总金额比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=jine\_xb['不同性别会员的总销售金额']  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1], labels=['Female','Male'], labeldistance=1.3)  #数值显示情况：autopct='%.2f%%'表示显示两位小数，%%显示%，第一个%表示转译  #explode=[0.1,0,0] 每一份的尖端离圆心的距离，突出显示作用  #设置标签：labels=['1','2','3']  #标签离圆心的距离：labeldistance=1.3 圆半径默认为1  plt.show() |

运行代码可得到：



## 根据单据号字段，分别统计会员与非会员的订单数并可视化

1、将非会员分离出来，并求出会员和非会员的总单据数

|  |
| --- |
| #非会员表  fhuiyuan=data\_outer[data\_outer['kh'].notnull()==False]  #会员的单据数  H=huiyuan['djh'].count()  #非会员的单据数  F=fhuiyuan['djh'].count()  print(H,F) |

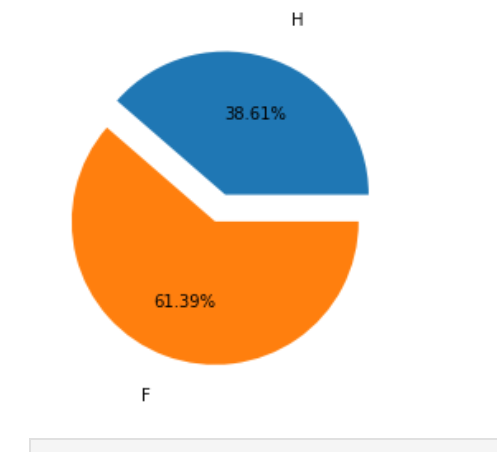
运行代码可得到：



1. 绘制会员与非会员的订单数的比例饼图

|  |
| --- |
| #绘制会员和非会员的单据数比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=[H,F]  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1], labels=['H','F'], labeldistance=1.3)  plt.show() |

运行代码可得到：



## 根据消费金额字段，分别统计会员与非会员的消费金额并可视化

1. 统计会员和非会员的消费金额

|  |
| --- |
| #统计会员与非会员的消费金额  #会员的单据数  H=huiyuan['je'].sum()  #非会员的单据数  F=fhuiyuan['je'].sum()  print(H,F) |

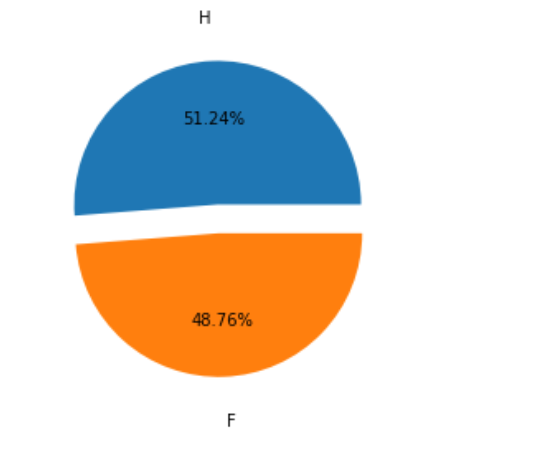
运行代码可得到：



1. 绘制会员和非会员的消费金额饼图

|  |
| --- |
| #绘制会员和非会员的消费金额比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=[H,F]  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1], labels=['H','F'], labeldistance=1.3)  plt.show() |

运行代码可得到：

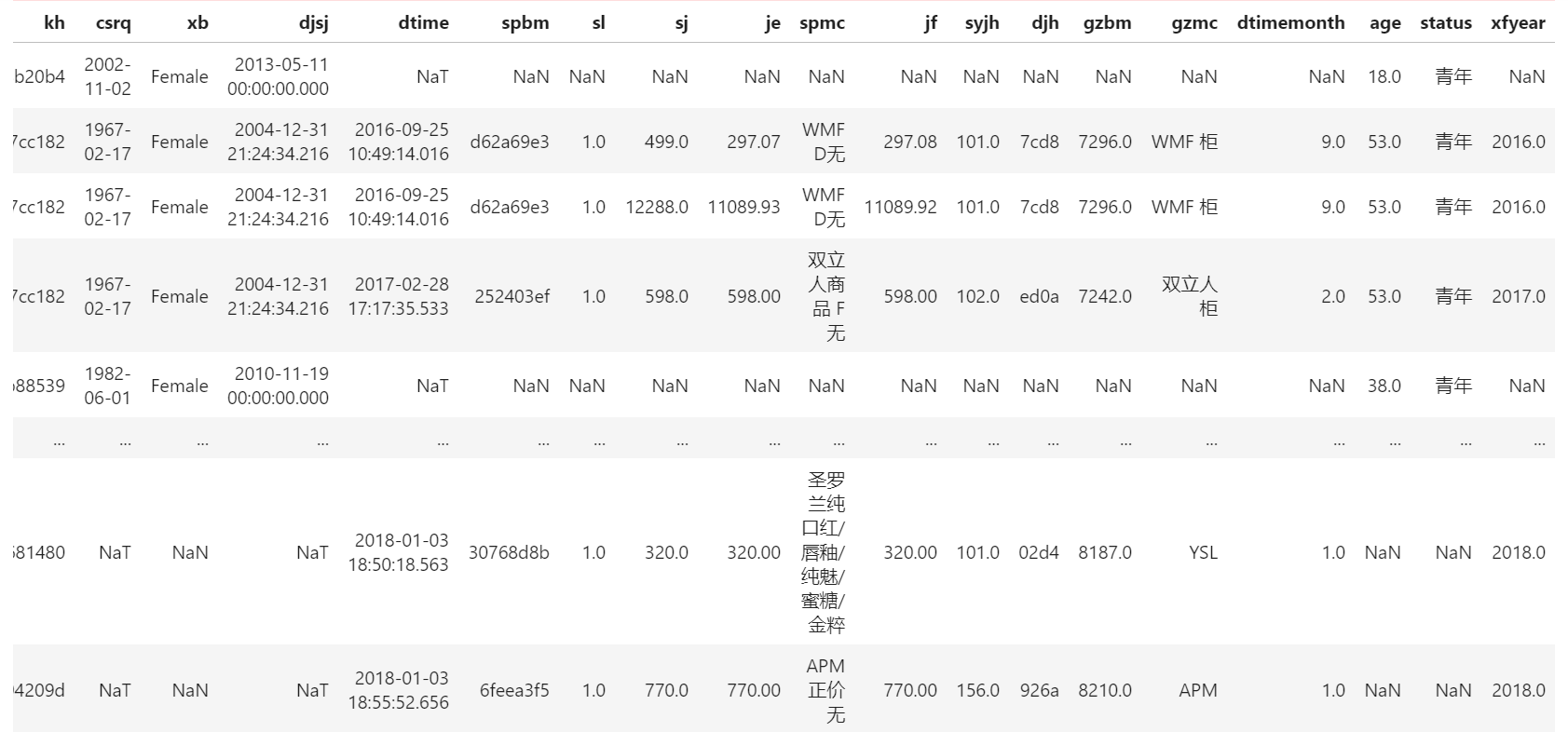


## 统计不同年份的会员消费金额并可视化

1. 先将会员的消费时间的年份分离出来

|  |
| --- |
| #将会员消费时间的年份分离出来  huiyuan['dtime'] = pd.to\_datetime(huiyuan['dtime'],format='%Y-%m-%d %H:%M:%S')#将读取的日期转为datatime格式  huiyuan['xfyear']=[i.year for i in huiyuan['dtime']]#遍历时间变量，提取年份，使用datatime模块的属性  huiyuan |

运行代码可得到：



1. 再按会员的消费时间的年份分组得到总的消费金额

|  |
| --- |
| #再按会员消费时间的年份分组  jine\_HY=huiyuan.groupby(by=["xfyear"])["je"].agg({"不同年份会员的总销售金额":np.sum})  jine\_HY |

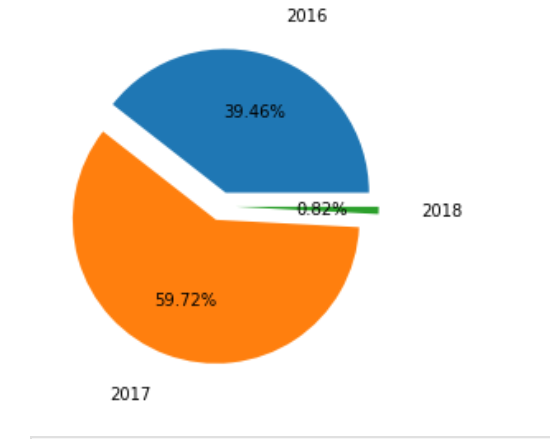
运行代码可得到：



1. 不同年份消费金额的饼图

|  |
| --- |
| #绘制不同年份会员的总销售金额比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=jine\_HY['不同年份会员的总销售金额']  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1,0.1], labels=['2016','2017','2018'], labeldistance=1.3)  plt.show() |

运行代码可得到：

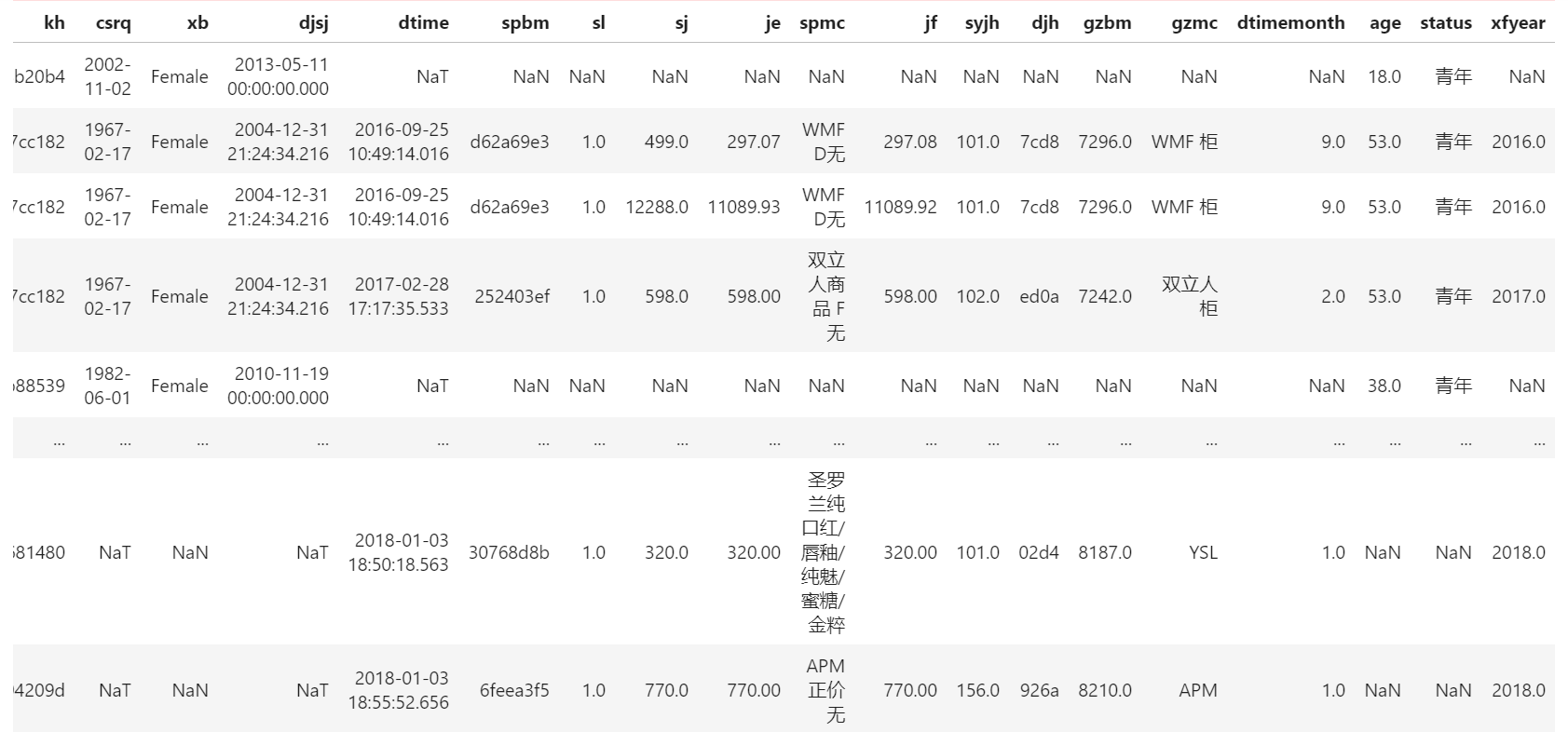


## 按不同年份统计会员各月份消费金额并可视化

1、先将会员的消费时间的年份分离出来

|  |
| --- |
| #将会员消费时间的年份分离出来  huiyuan['dtime'] = pd.to\_datetime(huiyuan['dtime'],format='%Y-%m-%d %H:%M:%S')#将读取的日期转为datatime格式  huiyuan['xfyear']=[i.year for i in huiyuan['dtime']]#遍历时间变量，提取年份，使用datatime模块的属性  huiyuan |

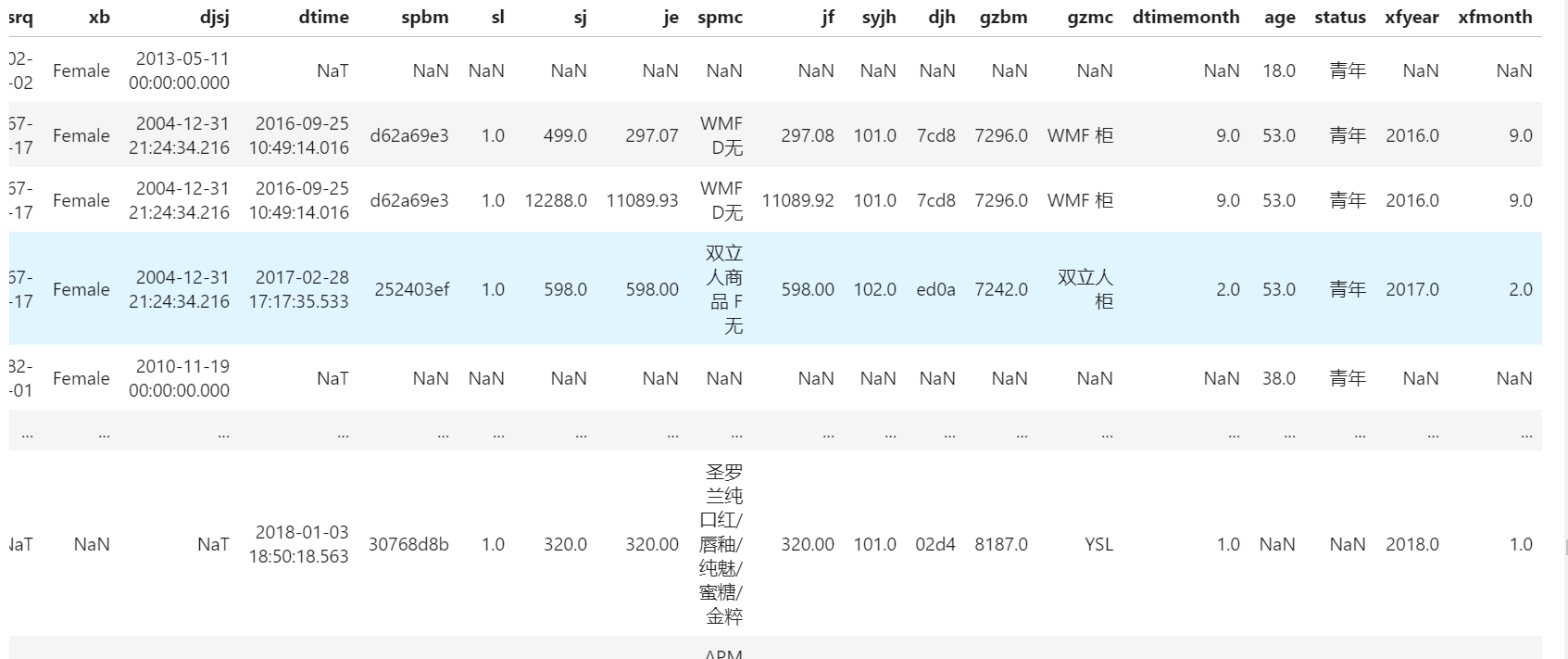
运行代码可得到：



2、再将会员的消费时间的月份分离出来

|  |
| --- |
| #将会员消费时间的月份分离出来  huiyuan['dtime'] = pd.to\_datetime(huiyuan['dtime'],format='%Y-%m-%d %H:%M:%S')#将读取的日期转为datatime格式  huiyuan['xfmonth']=[i.month for i in huiyuan['dtime']]#遍历时间变量，提取年份，使用datatime模块的属性  huiyuan |

运行代码可得到：



3、按不同年份统计会员各月份消费金额

|  |
| --- |
| #按照消费时间年份和月份分组求金额和  year\_month=huiyuan.groupby(by=["xfyear", "xfmonth"])["je"].agg({"销售金额（按年份月份）":np.sum})  year\_month |

运行代码可得到：

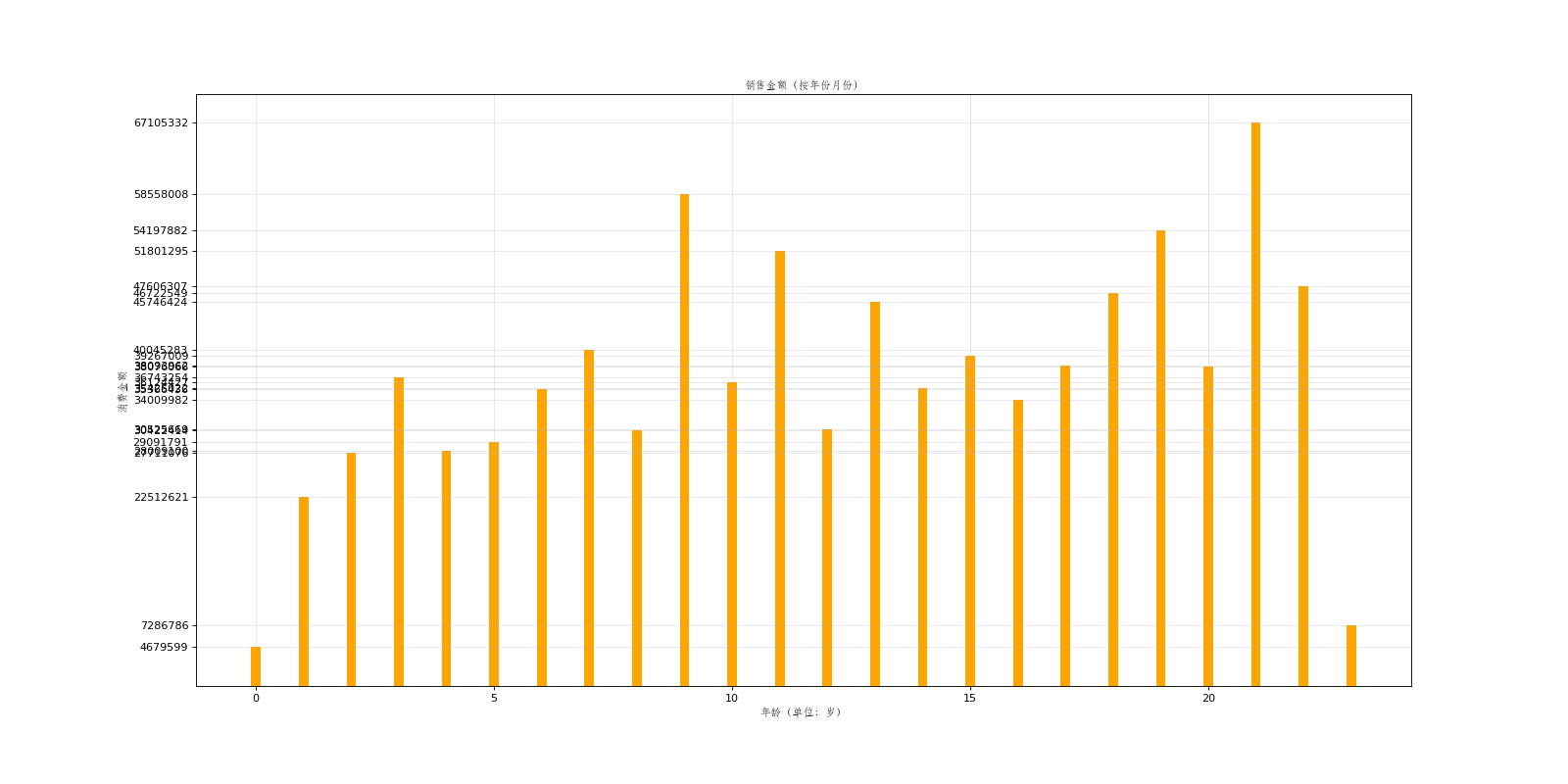




1. 对不同年份统计会员各月份消费金额进行可视化

|  |
| --- |
| #绘制按照消费时间年份和月份分组求金额和条形图  from matplotlib import pyplot as plt  from matplotlib import font\_manager  #设置中文  my\_font=font\_manager.FontProperties(fname="C:/Windows/Fonts/STFANGSO.ttf")  #调节图片大小  fig=plt.figure(figsize=(20,10),dpi=80)  x=range(len(year\_month['销售金额（按年份月份）'] ))  y=year\_month['销售金额（按年份月份）'] #将消费总额设为y  plt.bar(x,y,width=0.2,color="orange")  #设置标题  plt.title('销售金额（按年份月份）',fontproperties=my\_font)  plt.yticks(y)  #设置网格，使其更加醒目 透明度为0.3  plt.grid(True,alpha=0.3) #也可这样写plt.grid(alpha=0.3)  #添加描述信息  plt.xlabel("年龄（单位：岁）",fontproperties=my\_font)  plt.ylabel('消费金额',fontproperties=my\_font)  #保存图片  plt.savefig("./jineyear\_month.png")  #展示  plt.show() |

运行代码可得到：

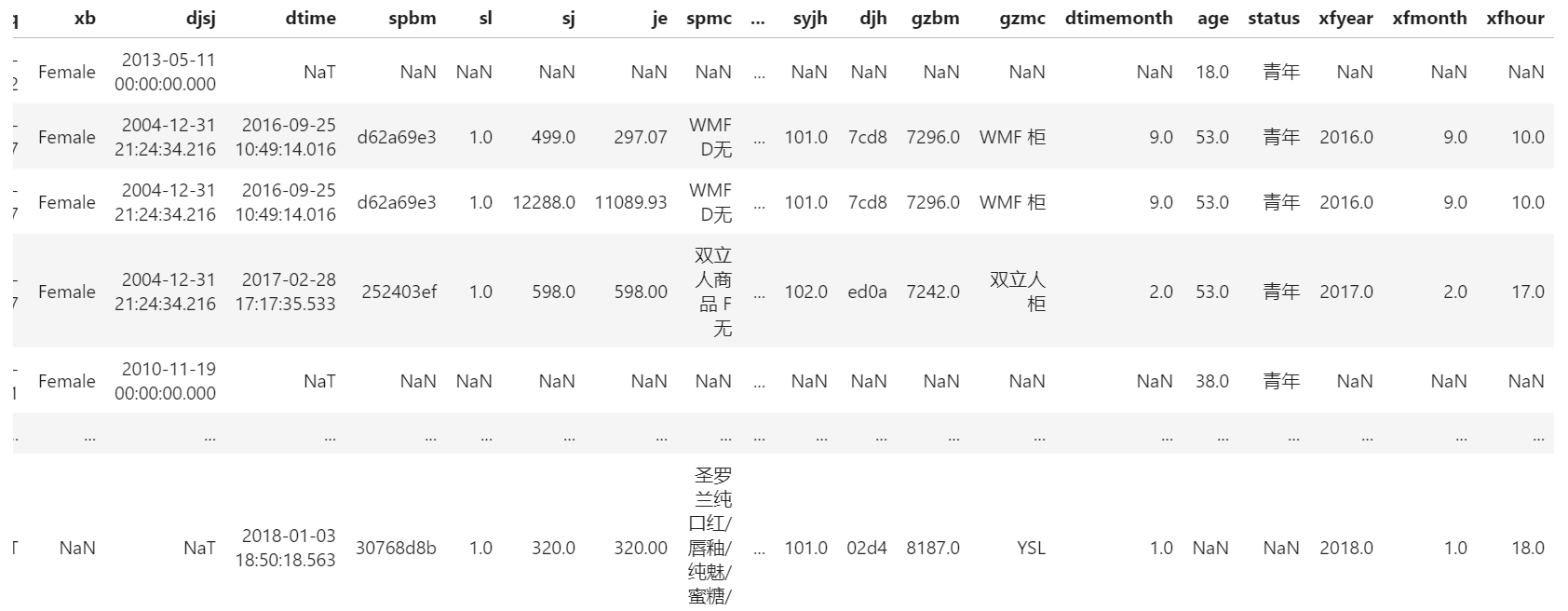


## 根据会员消费时间，将一天分为凌晨，早上，上午，中午，下午，晚上，统计每个时段的会员人数并可视化

1、先将消费时间的是分离出来

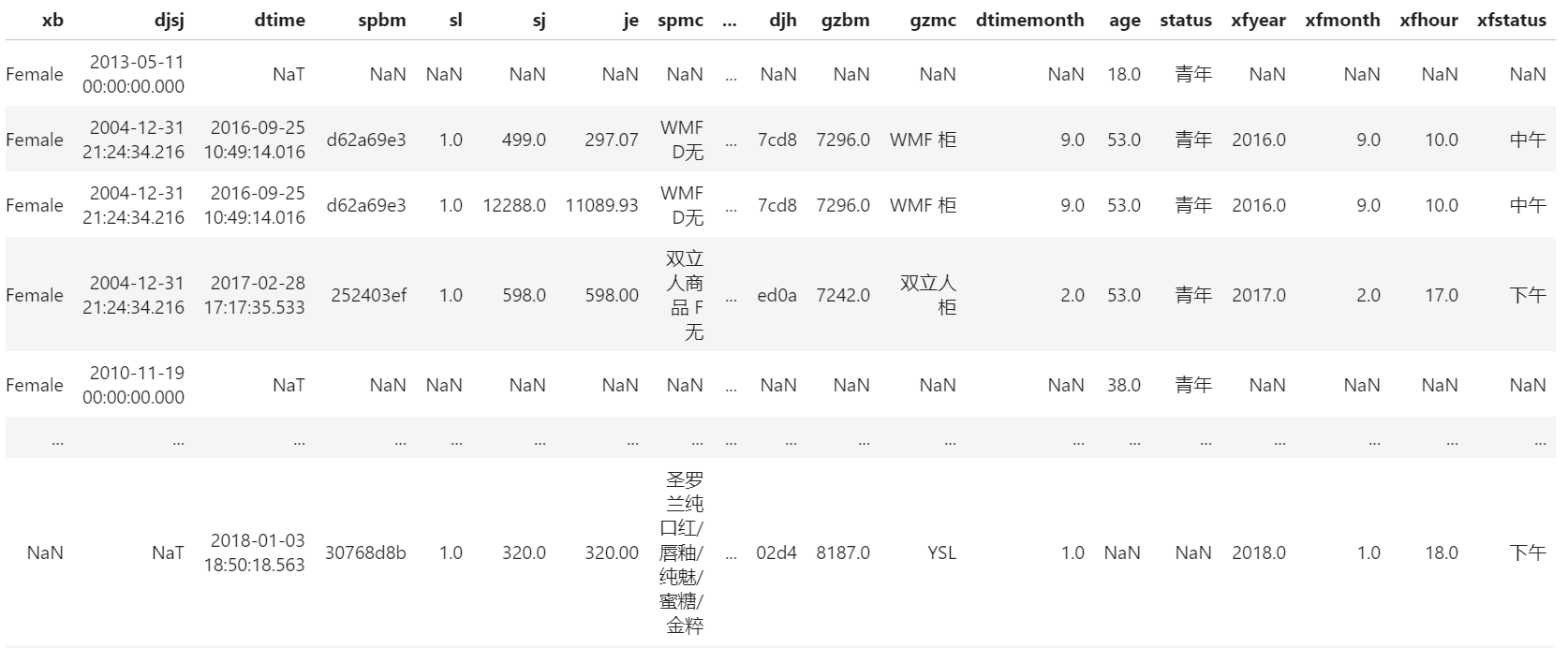
|  |
| --- |
| #将会员消费时间的时分离出来  huiyuan['dtime'] = pd.to\_datetime(huiyuan['dtime'],format='%Y-%m-%d %H:%M:%S')#将读取的日期转为datatime格式  huiyuan['xfhour']=[i.hour for i in huiyuan['dtime']]#遍历时间变量，提取年份，使用datatime模块的属性  huiyuan |

运行代码可得到：

2、再按照刚刚分离出来的时进行划分

|  |
| --- |
| #划分  huiyuan['xfstatus']=pd.cut(huiyuan['xfhour'],bins=[0,5,9,14,18,24],labels=['凌晨','早上','中午','下午','晚上'])  huiyuan |

运行代码可得到：



1. 以刚刚分出的时状态进行分组，求每个时段人数

|  |
| --- |
| #按照xfstatus分组  people1=huiyuan.groupby(by=["xfstatus"])["kh"].count()  people=pd.DataFrame(people1)  people |

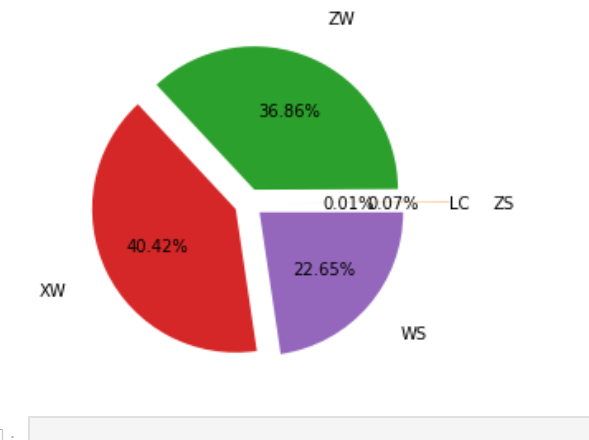
运行代码可得到：



1. 以得到的各时段的人数进行可视化

|  |
| --- |
| #绘制不同时段会员人数比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=people['kh']  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.4,0.1,0.1,0.1], labels=['LC','ZS','ZW','XW','WS'], labeldistance=1.3)  plt.show() |

运行代码可得到：

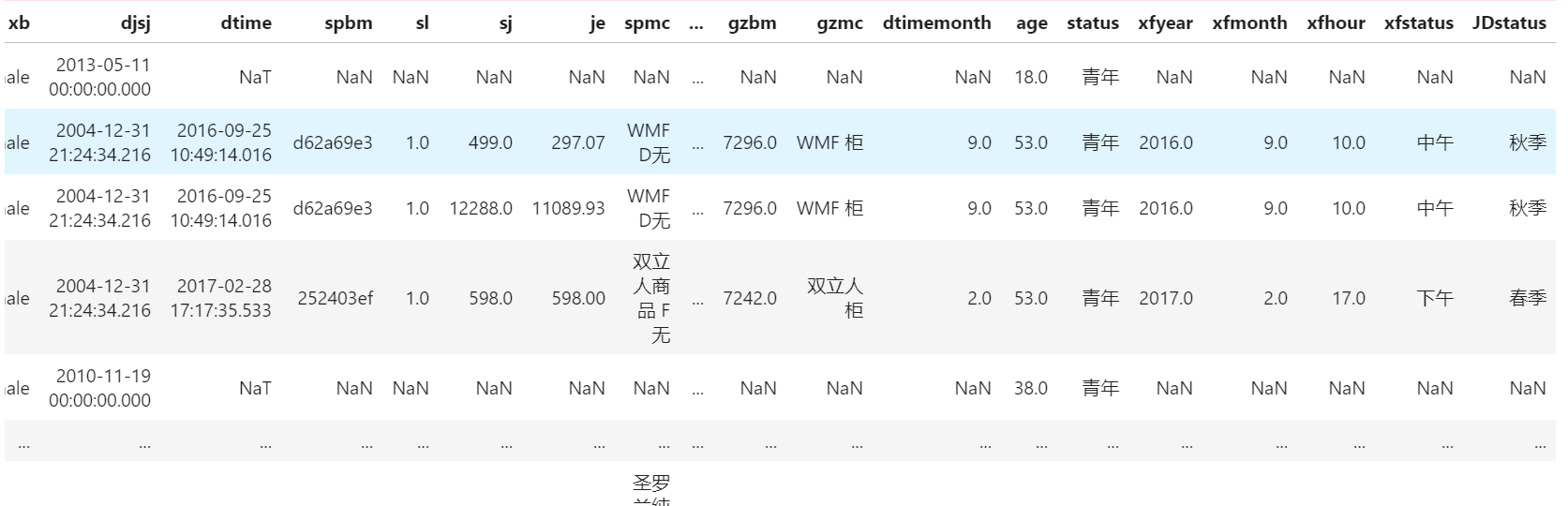


## 按照会员消费时间，按年份统计每个季度会员消费人数并可视化

1. 将上面项目中的求出的xfmonth字段进行划分

|  |
| --- |
| #划分  huiyuan['JDstatus']=pd.cut(huiyuan['xfmonth'],bins=[0,3,6,9,12],labels=['春季','夏季','秋季','冬季'])  huiyuan |

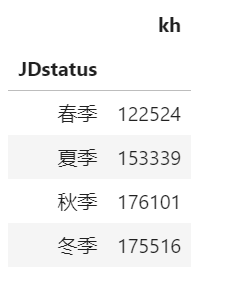
运行代码可得到：



2、以刚刚分出的时状态进行分组，求每个时段人数

|  |
| --- |
| #按照JDstatus分组  people2=huiyuan.groupby(by=["JDstatus"])["kh"].count()  people22=pd.DataFrame(people2)  people22 |

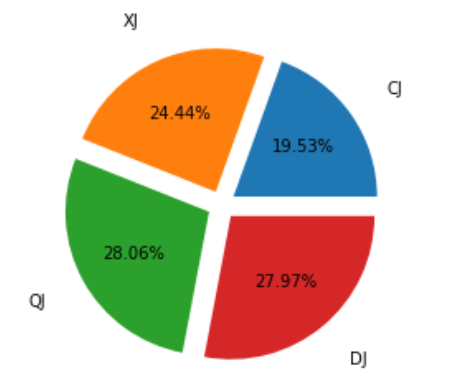
运行代码可得到：



3、以得到的各时段的人数进行可视化

|  |
| --- |
| #绘制不同时段会员人数比例饼图  plt.figure(figsize=(4,4)) #不为正方形，则饼图是椭圆的  num=people22['kh']  plt.pie(num, autopct='%.2f%%',explode=[0.1,0.1,0.1,0.1], labels=['CJ','XJ','QJ','DJ'], labeldistance=1.3)  plt.show() |

运行代码可得到：



## 针对会员用户基本特征，构建了会员卡号，年龄，性别，会员积分特征标签

会员卡号，年龄，性别均可以从huiyuan表中看出

求会员积分

|  |
| --- |
| #按照卡号分组求积分和  JF=huiyuan.groupby(by=["kh"])["jf"].agg({"会员积分":np.sum})  JF |

运行代码可得到：

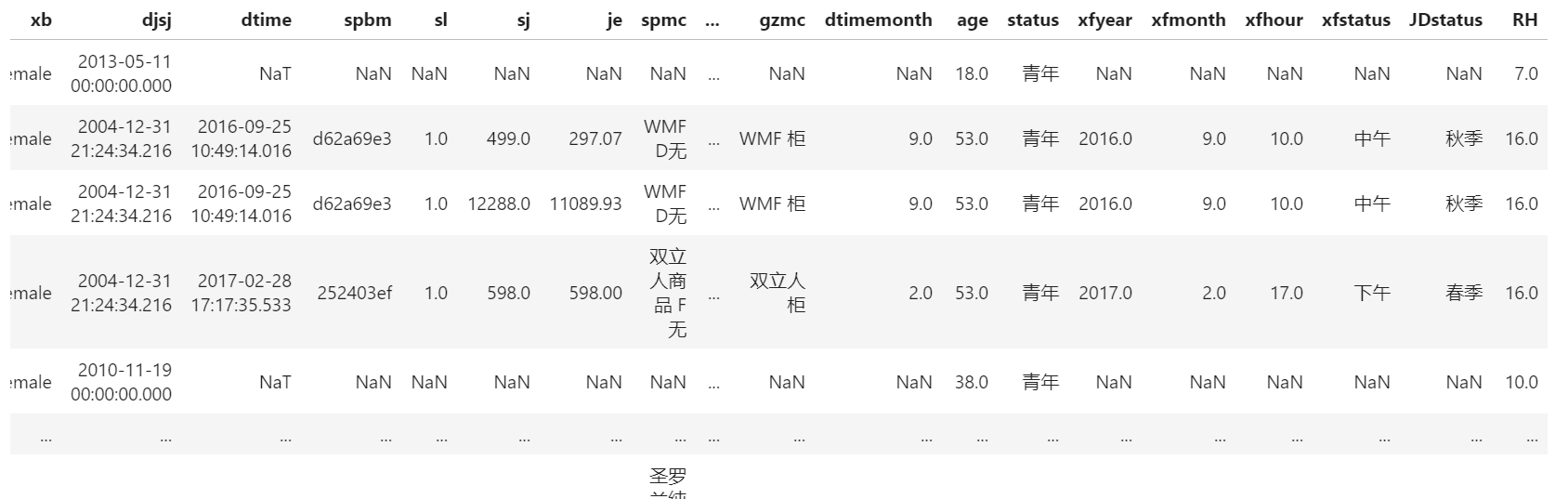


## 针对会员用户业务特征，构建了根据入会时长来确定新/老会员，消费水平特征标签

1. 计算入会时长，并添加

|  |
| --- |
| #计算入会时长，并增加  RHtime=pd.DataFrame(huiyuan['djsj'])  RHtime['djsj'] =pd.to\_datetime(huiyuan['djsj'])  import datetime as dt  now\_year=dt.datetime.today().year  huiyuan['RH']=now\_year-RHtime['djsj'].dt.year  huiyuan |

运行代码可得到：



1. 求出会员入会时长的平均值，大于这个平均值的为老会员，小于这个平均值的为新会员

|  |
| --- |
| #求出会员时长的平均值，大于这个时长就定义为老会员，小于则为新会员  min\_RH=huiyuan['RH'].min()  mean\_RH=huiyuan['RH'].mean()  max\_RH=huiyuan['RH'].max()  huiyuan['RHstatus']=pd.cut(huiyuan['RH'],bins=[min\_RH,mean\_RH,max\_RH],labels=['新会员','老会员'])  huiyuan |

运行代码可得到：



1. 消费金额高于200高消费，100-200中等消费，低于100低消费

|  |
| --- |
| #根据消费水平，来判断会员永辉是高消费（200元以上） 中等消费（100-200元） 低消费（100元以下）  min\_je=huiyuan['je'].min()  max\_je=huiyuan['je'].max()  huiyuan['jestatus']=pd.cut(huiyuan['je'],bins=[min\_je,100,200,max\_je],labels=['低消费','中等消费','高消费'])  huiyuan |

运行代码可得到：



## 针对会员用户兴趣特征，构建了购物时间段的偏好（由xfstatus可知），购物季节的偏好（JDstatus可知），购物类型的偏好（根据用户购买物品的柜组名称gzmc可知）特征标签

## 构建了RFM模型，R—最近一次消费时间，F—消费频率，M—消费金额

1. 以卡号和消费时间分组，求金额和，消费频率

|  |
| --- |
| #以会员卡号和消费时间分组，求金额和，得到每个会员的消费总金额，每个会员消费时间里最下面那个时间就是最近一次消费时间  task\_kh=huiyuan.groupby(by=['kh','dtime'])  kh\_dtime=huiyuan.groupby(by=['kh','dtime'])["je"].agg({"会员积分":np.sum})  kh\_dtime  #求得每个会员的消费频率（会员购买总次数）  #首先按照卡号和消费时间分组后得到每个会员在某个时间内的总次数  count=task\_kh['dtime'].count()  count1=pd.DataFrame(count)  kh\_dtime['count']=count  kh\_dtime |

运行代码可得到：



2、使用sklearn估计器构建聚类模型

|  |
| --- |
| from sklearn.cluster import KMeans #构建聚类模型  model=KMeans(n\_clusters=3).fit(kh\_dtime[['会员积分','count']]) #构建的是K\_Means聚类的模型  #看聚类结果  print(model.labels\_)  print(model.cluster\_centers\_) #聚类中心  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  # 绘制雷达图  #标签  labels = ['je','count']  #数据个数  k = 3  plot\_data = model.cluster\_centers\_  #指定颜色  color = ['b', 'g', 'r']  angles = np.linspace(0, 2\*np.pi, k, endpoint=False)  # 闭合  plot\_data = np.concatenate((plot\_data, plot\_data[:,[0]],plot\_data[:,[0]]), axis=1)  # 闭合  angles = np.concatenate((angles, [angles[0]]))  fig = plt.figure()  #polar参数  ax = fig.add\_subplot(111, polar=True)  for i in range(len(plot\_data)):  # 画线  ax.plot(angles, plot\_data[i], 'o-', color = color[i], label = 'client'+str(i), linewidth=2)  ax.set\_rgrids(np.arange(0.01, 3.5, 0.5), np.arange(-1, 2.5, 0.5))  ax.set\_thetagrids(angles \* 180/np.pi, labels)  plt.legend(loc = 4)  plt.show() |

运行代码可得到：



进行聚类时设置的K值分别为3，能很好的反应情况

## 

1、统计会员中购买的商品名称最多

|  |
| --- |
| #统计会员中购买的商品名称最多  import matplotlib.pyplot as plt  import wordcloud  plt.figure(figsize=(20,8))  mid\_je = huiyuan.groupby(by=["spmc"])["je"].sum()  mid\_je = mid\_je.reset\_index()  dic = {x[0]:x[1] for x in mid\_je.loc[:,:].values}  wc= wordcloud.WordCloud(scale=16, font\_path='simhei.ttf',  background\_color='white', max\_words=100,colormap="coolwarm")  X = wc.generate\_from\_frequencies(dic)  plt.axis('off')  plt.imshow(X) |

运行代码可得到：



2、统计会员购买时间偏好

|  |
| --- |
| #统计会员购买时间偏好  import matplotlib.pyplot as plt  import wordcloud  plt.figure(figsize=(20,8))  mid\_je = huiyuan.groupby(by=["xfstatus"])["kh"].count()  mid\_je = mid\_je.reset\_index()  dic = {x[0]:x[1] for x in mid\_je.loc[:,:].values}  wc= wordcloud.WordCloud(scale=16, font\_path='simhei.ttf',  background\_color='white', max\_words=100,colormap="coolwarm")  X = wc.generate\_from\_frequencies(dic)  plt.axis('off')  plt.imshow(X) |

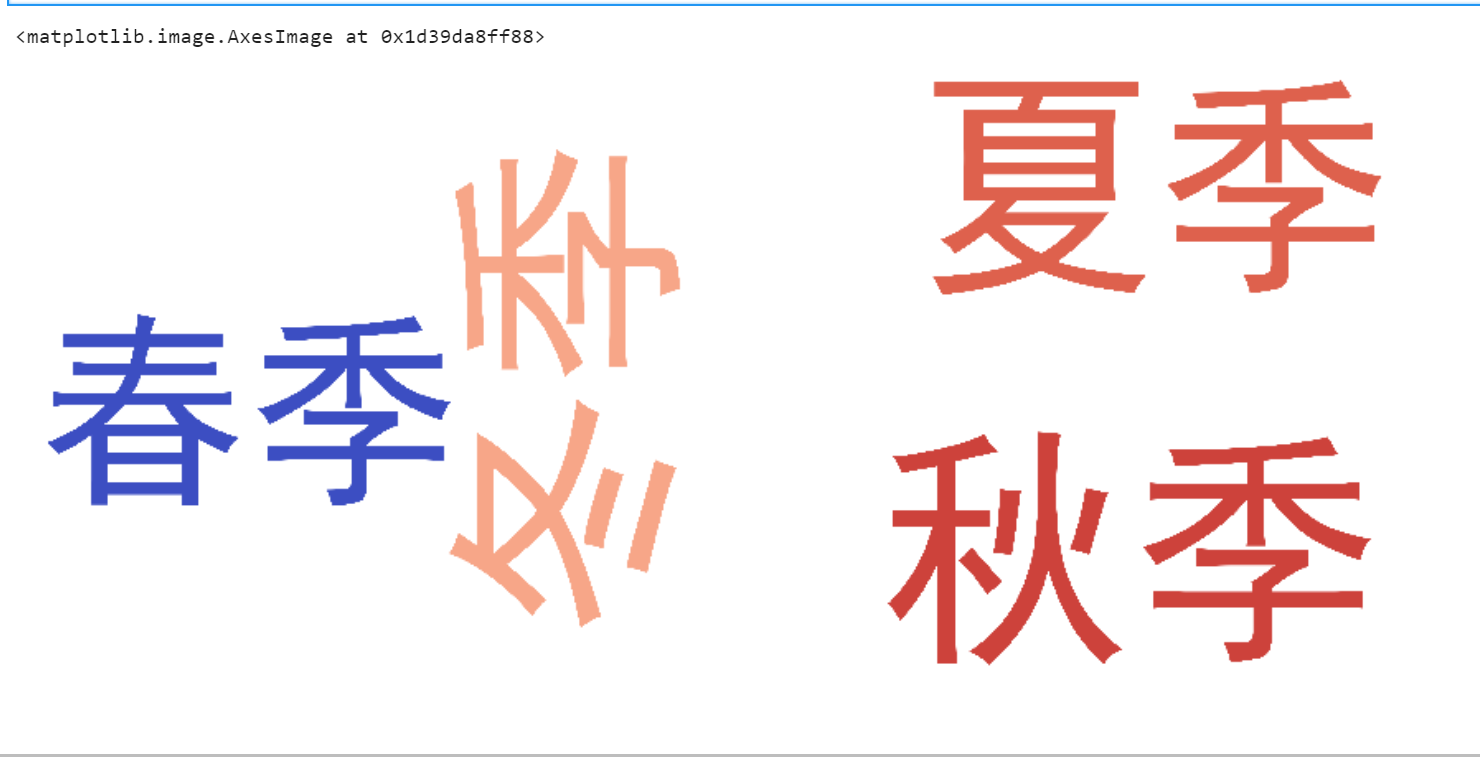
运行代码可得到：



3、统计会员购买季度偏好

|  |
| --- |
| #统计会员购买季度偏好  import matplotlib.pyplot as plt  import wordcloud  plt.figure(figsize=(20,8))  mid\_je = huiyuan.groupby(by=["JDstatus"])["kh"].count()  mid\_je = mid\_je.reset\_index()  dic = {x[0]:x[1] for x in mid\_je.loc[:,:].values}  wc= wordcloud.WordCloud(scale=16, font\_path='simhei.ttf',  background\_color='white', max\_words=100,colormap="coolwarm")  X = wc.generate\_from\_frequencies(dic)  plt.axis('off')  plt.imshow(X) |

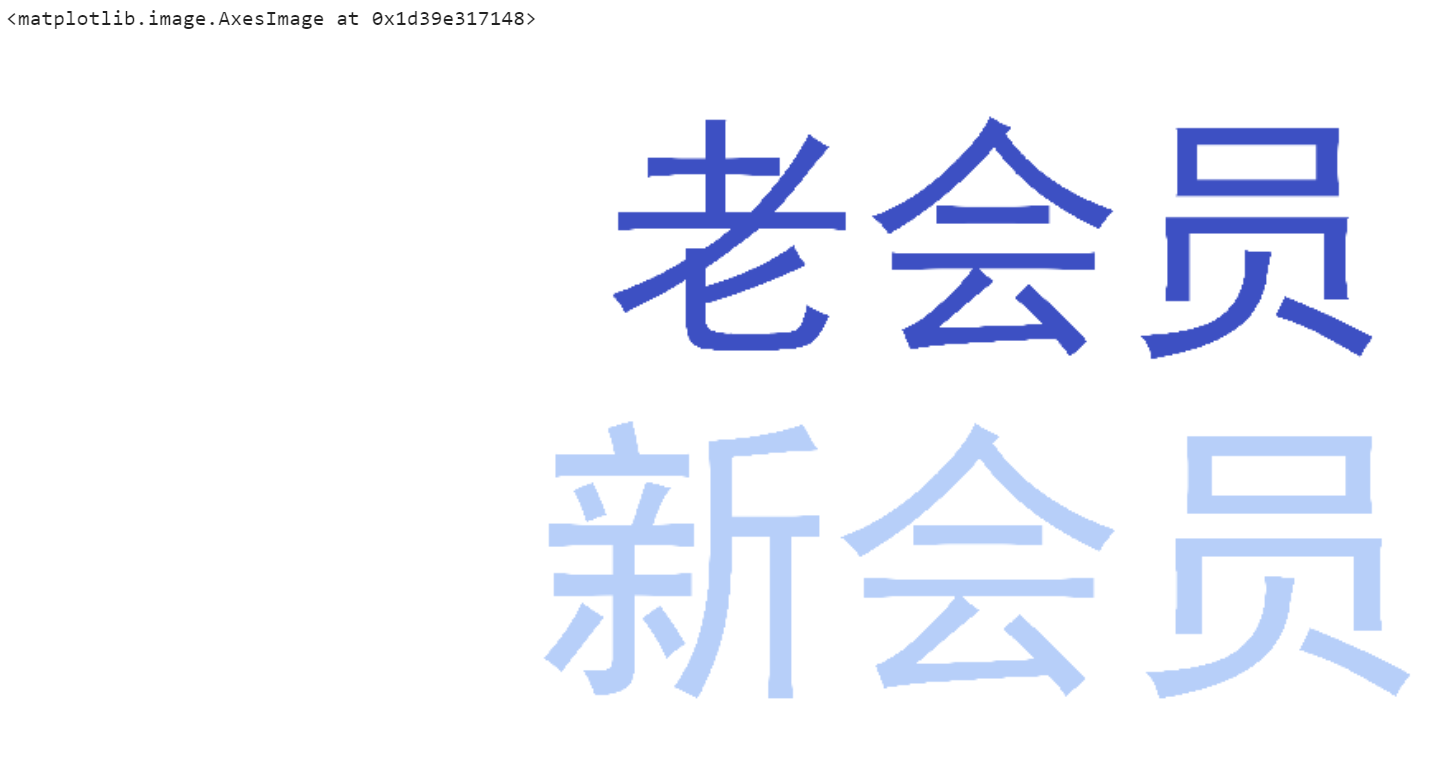
运行代码可得到：



4、统计会员分类

|  |
| --- |
| #统计会员分类  import matplotlib.pyplot as plt  import wordcloud  plt.figure(figsize=(20,8))  mid\_je = huiyuan.groupby(by=["RHstatus"])["kh"].count()  mid\_je = mid\_je.reset\_index()  dic = {x[0]:x[1] for x in mid\_je.loc[:,:].values}  wc= wordcloud.WordCloud(scale=16, font\_path='simhei.ttf',  background\_color='white', max\_words=100,colormap="coolwarm")  X = wc.generate\_from\_frequencies(dic)  plt.axis('off')  plt.imshow(X) |

运行代码可得到：



5、统计会员消费水平

|  |
| --- |
| #统计会员消费水平  import matplotlib.pyplot as plt  import wordcloud  plt.figure(figsize=(20,8))  mid\_je = huiyuan.groupby(by=["jestatus"])["kh"].count()  mid\_je = mid\_je.reset\_index()  dic = {x[0]:x[1] for x in mid\_je.loc[:,:].values}  wc= wordcloud.WordCloud(scale=16, font\_path='simhei.ttf',  background\_color='white', max\_words=100,colormap="coolwarm")  X = wc.generate\_from\_frequencies(dic)  plt.axis('off')  plt.imshow(X) |

运行代码可得到：

