**高礼课程报告：复杂SQL查询与Python分析**

——基于TPC-DS数据库的商业分析

**2020200883 经济学院 钟东彤**

**主题：**企业销售内在规律探索与未来销售策略分析

**目标：**

通过Sql Server和Python以及power BI的交叉分析与应用，基于TPC-DS数据库进行企业销售规律现状和未来商业策略的制定，对数据的进行仪表板的可视化，同时通过机器学习对用户进行更进一步的商业价值评估，实现企业效益最大化。

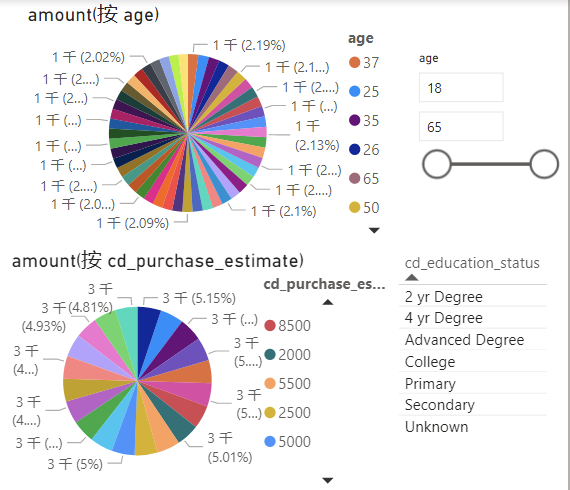
**子主题1：顾客画像**

用户画像主要就是从年龄、地区、教育程度、潜在购买力这几个方面进行分析，同时为了更加合理化地体现购买力，选取了18至65岁为主要分析年龄层。【年龄计算基于原始数据集的最晚日期2003年】

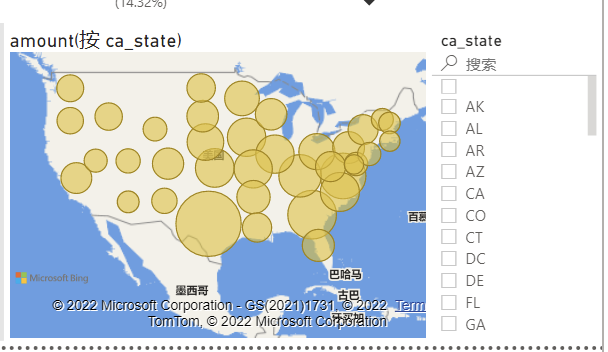
然后在三个进一步查询中通过三种分类的依据对顾客群的分布构成进行进一步的细化分析。

|  |
| --- |
| #建立视图  create view customer\_demo as  Select (2003-c\_birth\_year) age,ca\_state , cd\_education\_status, cd\_purchase\_estimate, count(\*) amount  from customer,customer\_demographics,customer\_address  where c\_current\_cdemo\_sk=cd\_demo\_sk  and c\_current\_addr\_sk=ca\_address\_sk  and c\_birth\_year is not null  and 2003-c\_birth\_year between 18 and 65  group by ca\_state,cd\_education\_status,c\_birth\_year,cd\_purchase\_estimate; |

（1）顾客的本身分布特征能够通过年龄滑块、教育程度以及预期购买力的多维度综合交叉展示，使用数量分布的饼图进行数据可视化展示，由于其组合维度过于多样，操作也比较便捷，具体可以看最终的power BI文件，此处不再赘述。



（2）地理分布特征可以通过气泡图和地图进行呈现，图中的气泡大小代表该地区的用户数量，通过切片器组件能够实现随便若干各地区之间的用户数量对比，不难看出用户大多聚集于美国东部中部，其中TX州的用户数量最多，达到了5103人，DC州的用户最少，只有16人。因此未来的门店可以多往用户多的州进行拓展，而用户少的州可以考虑进行差异化的商品销售策略。

，

**子主题2：销售渠道的差异化分析**

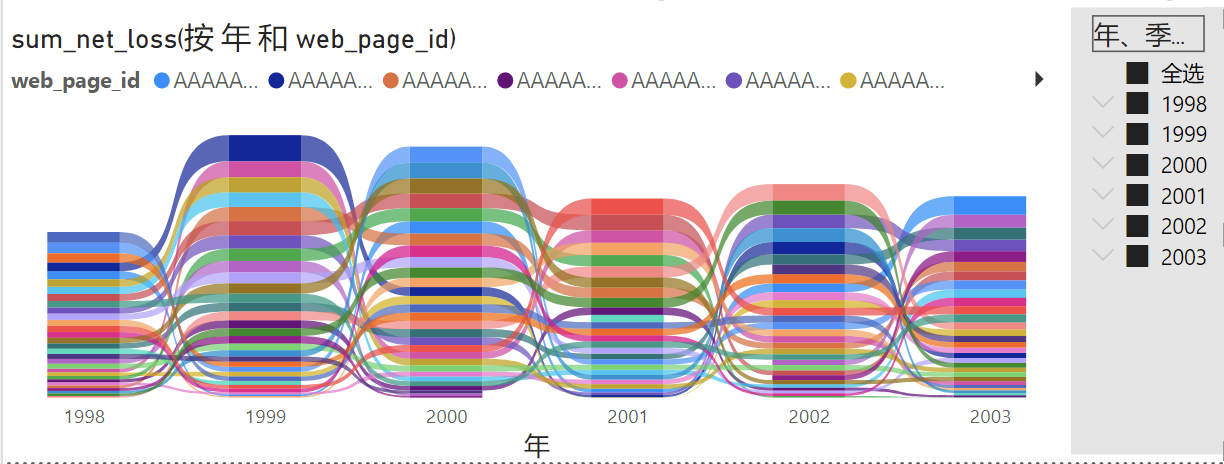
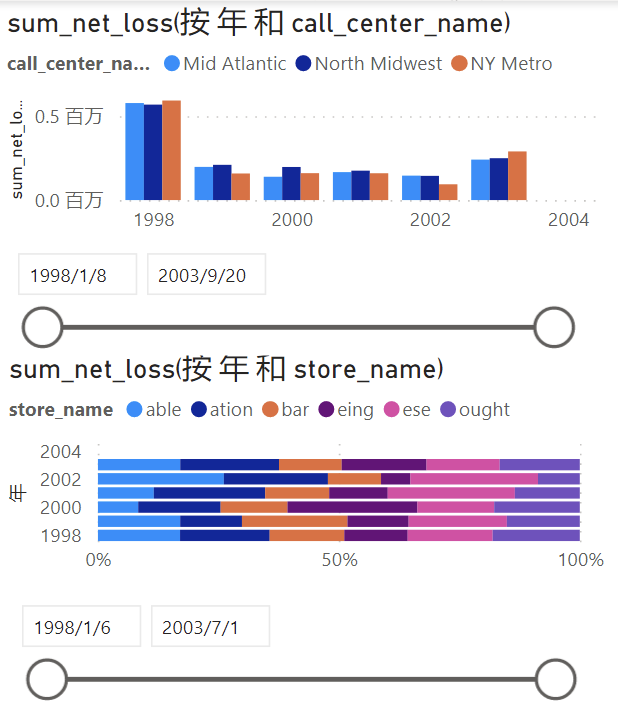
首先做不同分销渠道的销售量时间序列分析【时间跨度为100个销售日，视乎原始数据集】，通过建立视图的方式进行三个渠道的分开呈现，因为同时进行三个表格的链接查询耗时太久，所以分开来更合适。

|  |
| --- |
| #创建时间序列销量视图1（catalog）  Create view cs\_sum\_sales  select top(100) d\_date sold\_date,sum(cs\_quantity)as sum\_catalog\_sales  from date\_dim,catalog\_sales  where cs\_sold\_date\_sk = d\_date\_sk  group by d\_date  order by d\_date desc  #创建时间序列销量视图2（store）  Create view ss\_sum\_sales  select top(100) d\_date sold\_date,sum(ss\_quantity)as sum\_store\_sales  from date\_dim, store\_sales  where ss\_sold\_date\_sk = d\_date\_sk  group by d\_date  order by d\_date desc  #创建时间序列销量视图3（web）  Create view ws\_sum\_sales  select top(100) d\_date sold\_date,sum(ws\_quantity)as sum\_web\_sales  from date\_dim,web\_sales  where ws\_sold\_date\_sk = d\_date\_sk  group by d\_date  order by d\_date desc  #创建视图（catalog下细分各call center的net\_loss分析）  create view catalog\_net\_loss as  select top(1000) d\_date c\_date,cc\_name call\_center\_name, sum(cr\_net\_loss)as sum\_net\_loss  from catalog\_returns,date\_dim,call\_center  where cc\_call\_center\_sk=cr\_call\_center\_sk  and cr\_returned\_date\_sk=d\_date\_sk  group by d\_date,cc\_name  order by sum\_net\_loss  #进一步查询  select call\_center\_name,sum(sum\_net\_loss)as sum\_net\_loss  from catalog\_net\_loss  where sum\_net\_loss is not null  group by call\_center\_name  order by sum\_net\_loss  #创建视图（store下细分各store的net\_loss分析）  create view store\_net\_loss as  select top(1000) d\_date s\_date,s\_store\_name store\_name,sum(sr\_net\_loss)as sum\_net\_loss  from store\_returns,date\_dim,store  where sr\_store\_sk=s\_store\_sk  and sr\_returned\_date\_sk=d\_date\_sk  group by d\_date,s\_store\_name  order by sum\_net\_loss  #进一步查询（各store的net\_loss分别汇总）  select store\_name,sum(sum\_net\_loss)as sum\_net\_loss  from store\_net\_loss  where sum\_net\_loss is not null  group by store\_name  order by sum\_net\_loss  #创建视图（web下细分各web page的net\_loss分析）  create view web\_net\_loss as  select top(1000) d\_date w\_date, wp\_web\_page\_id web\_page\_id, sum(wr\_net\_loss)as sum\_net\_loss  from web\_returns,date\_dim, web\_page  where wr\_web\_page\_sk=wp\_web\_page\_sk  and wr\_returned\_date\_sk=d\_date\_sk  group by d\_date,wp\_web\_page\_id  order by sum\_net\_loss  #进一步查询  select web\_page\_id,sum(sum\_net\_loss)as sum\_net\_loss  from web\_net\_loss  where sum\_net\_loss is not null  group by web\_page\_id  order by sum\_net\_loss  #时间序列分析数据集  create view Sales as  with Catalogsales(The\_Date,Catalog\_Sales)as(  select d\_date The\_Date,sum(cs\_quantity)as Catalog\_Sales  from date\_dim,catalog\_sales  where cs\_sold\_date\_sk = d\_date\_sk  group by d\_date),  Storesales(The\_Date,Store\_Sales)as(  select d\_date The\_Date,sum(ws\_quantity)as Store\_Sales  from date\_dim,web\_sales  where ws\_sold\_date\_sk = d\_date\_sk  group by d\_date),  Websales(The\_Date,Web\_Sales)as(  select d\_date The\_Date,sum(ws\_quantity)as Web\_Sales  from date\_dim,web\_sales  where ws\_sold\_date\_sk = d\_date\_sk  group by d\_date)  select Catalogsales.The\_date,Catalog\_Sales,Store\_Sales,Web\_Sales  from Catalogsales,Storesales,Websales  where Catalogsales.The\_date=Storesales.The\_date  and Catalogsales.The\_date=Websales.The\_Date; |

1. Net loss分析

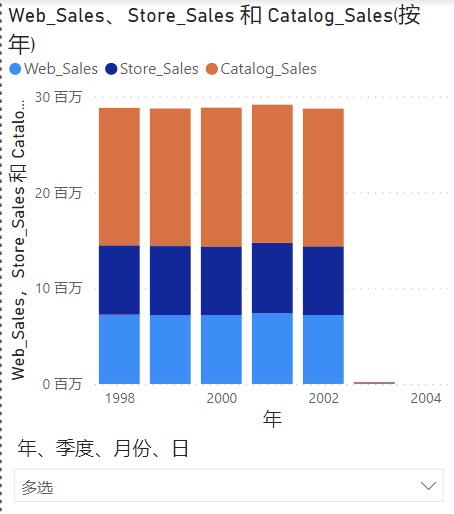
通过对各个不同的call center、store和web page的net loss的同比，环比，可以看到各个不同销售渠道在控制损失上的努力与效果差异，有以下结论：

1. 通过柱状图可以看出，在catalog销售渠道中，Mid Atlantic这个呼叫中心的表现较好，而NY Metro的表现较差，这可能与两地区的业务量有关系。
2. 通过百分比堆积条形图能够看出在store销售渠道中，bar商店的综合表现最好而able商店和ese商店表现都较差。
3. 通过丝带图可以看到在web销售渠道中，不同的网页在销售上的表现差异和发展趋势，能够识别出最有价值的网页，进而加大对该网页的投资力度，同时淘汰一些表现不好的网页。



1. Sales时间序列分析

通过对各个渠道的销售数据进行汇总，并通过查询与连接的操作给数据打上时间序列的标签，便于后面的进一步时间序列分析，从堆积柱状图可以看出，不同的渠道每年饿销售额波动极小，而且catalog渠道的销售数量远远高于其他渠道，说明水运对其运输和送货能力的提升非常明显，而线上的web和线下的store销售额相差不大，说明线上销售渠道的发展仍然任重道远，需要进一步重视。



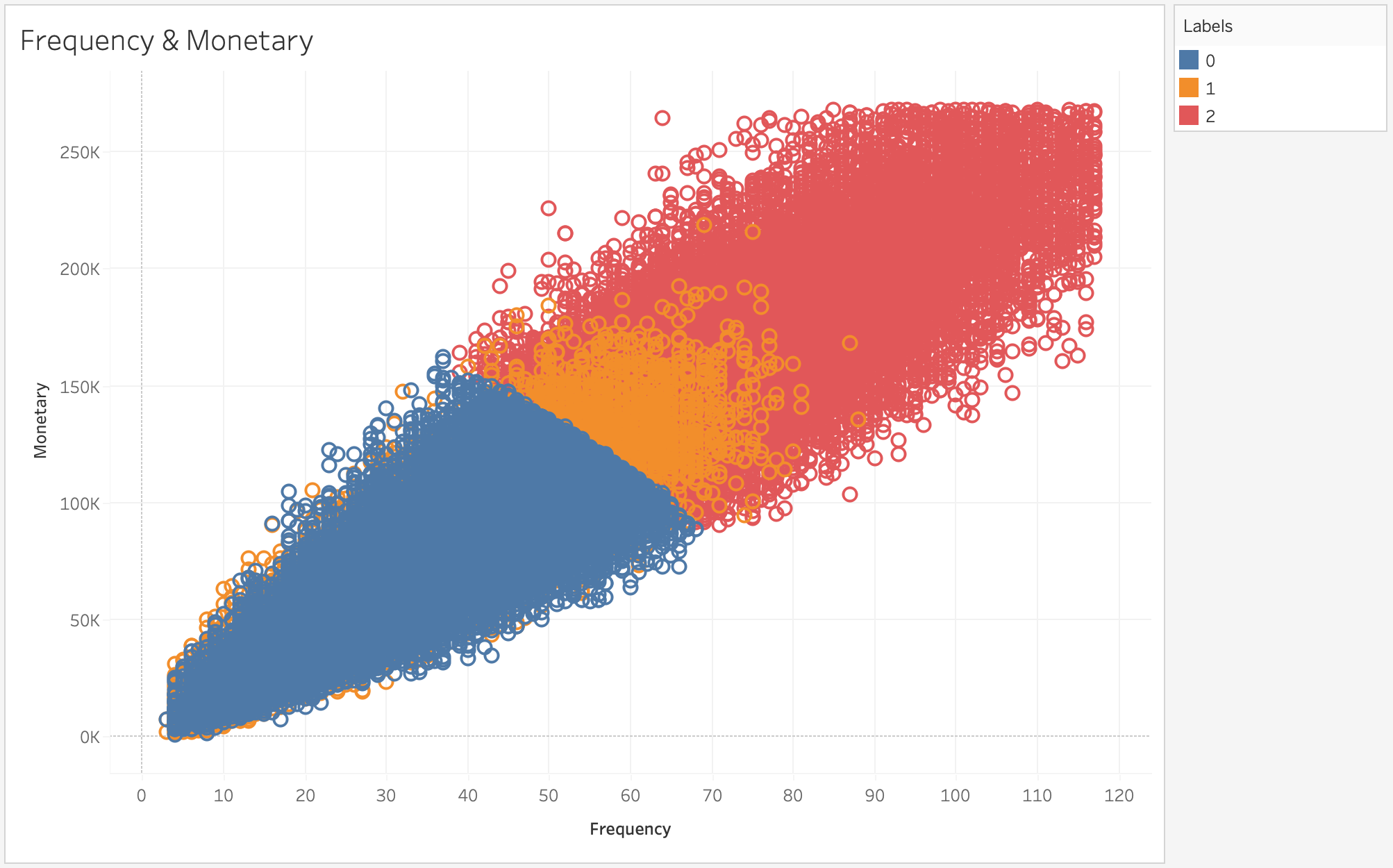
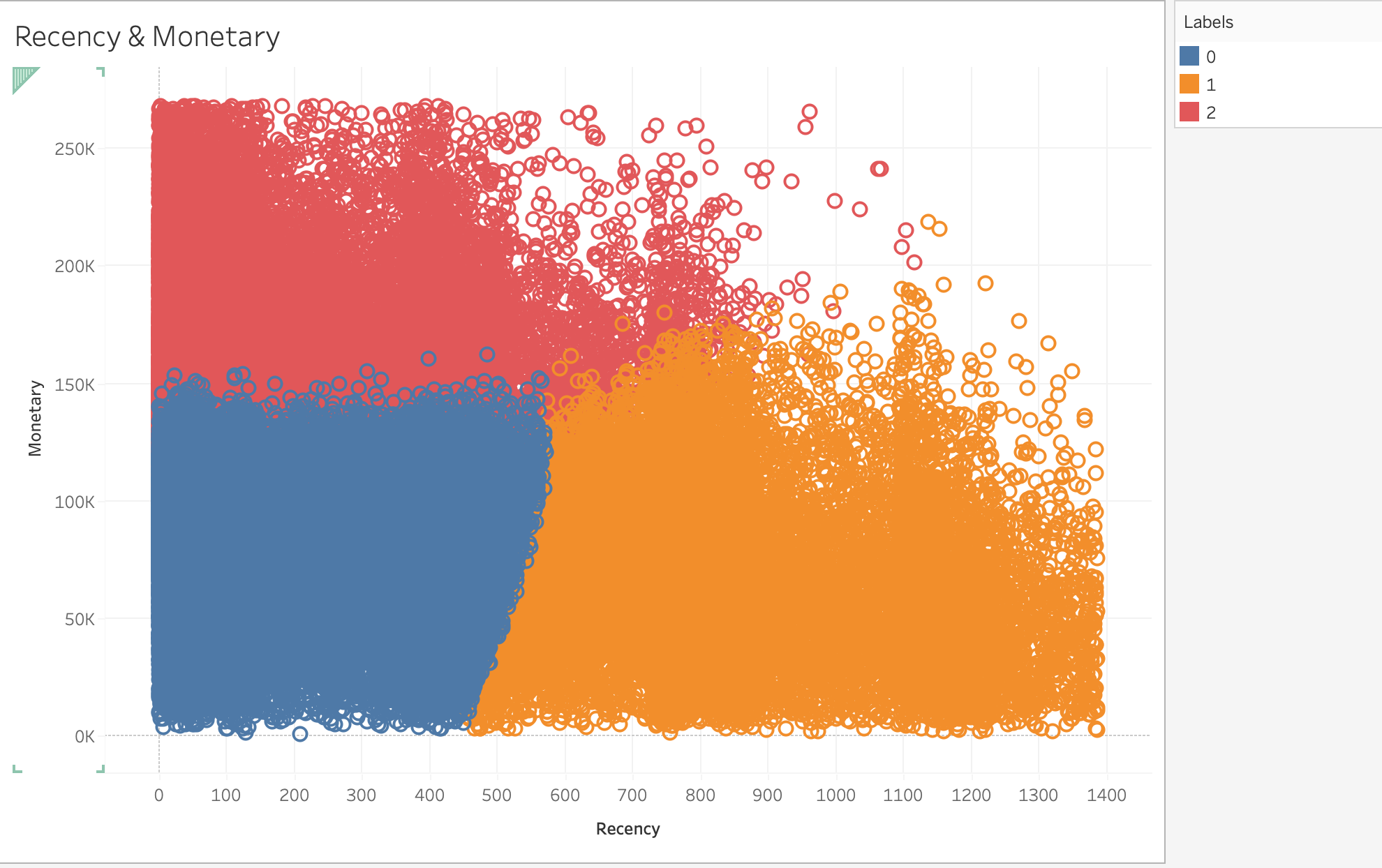
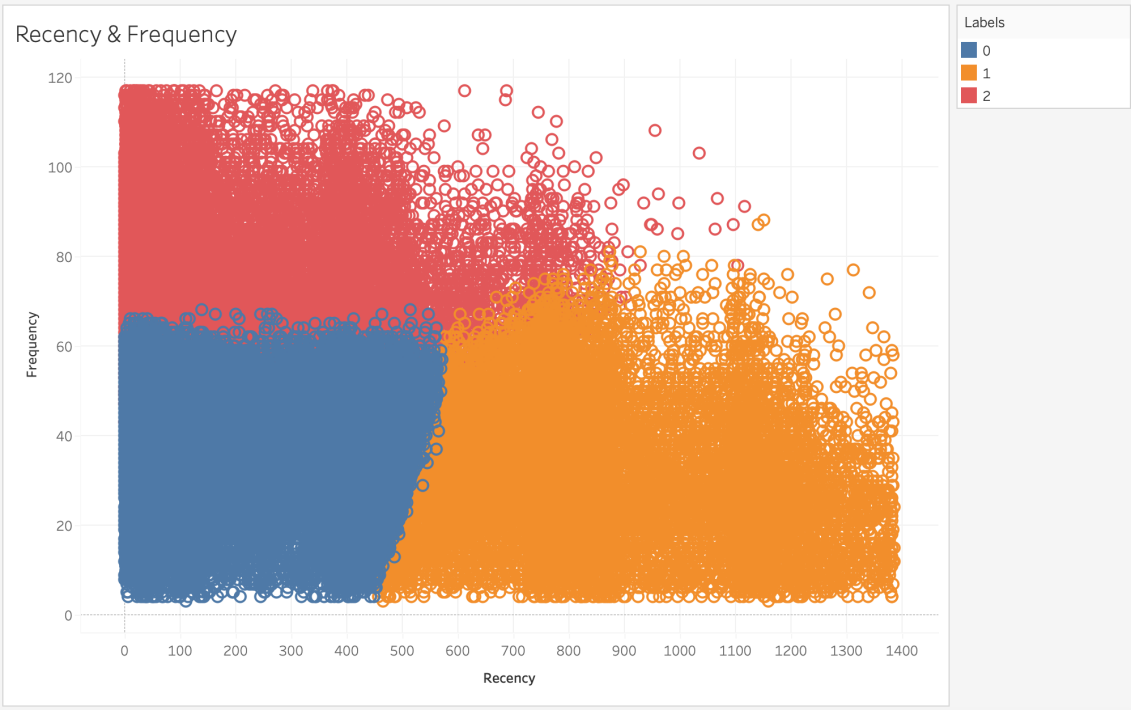
子主题3：RFM用户价值分析

|  |
| --- |
| import pymssql  import pandas as pd  import matplotlib.pyplot as plt  from pylab import\*  mpl.rcParams["font.sans-serif"]=["SimHei"]  #设置SQL Server 2019 连接参数  connect=pymssql.connect('DESKTOP-OLITH9E', 'tcxhb', '2001', 'TPCDS')  cursor=connect.cursor()  query="""  select  customer\_sk,  datediff(day,max(d\_date),'2002-12-31')as recency,  count(customer\_sk) as frequency,  sum(monetary) as monetary  from  (select  ss\_customer\_sk as customer\_sk,  ss\_sold\_date\_sk as sold\_date\_sk,  ss\_ext\_sales\_price as monetary  from  store\_sales  union all  select  ws\_bill\_customer\_sk as customer\_sk,  ws\_sold\_date\_sk as sold\_date\_sk,  ws\_ext\_sales\_price as monetary  from  web\_sales  union all  select  cs\_bill\_customer\_sk as customer\_sk,  cs\_sold\_date\_sk as sold\_date\_sk,  cs\_ext\_sales\_price as monetary  from  catalog\_sales)a  left join date\_dim d  on a.sold\_date\_sk=d.d\_date\_sk  where  d\_date between '1998-01-01'and'2002-12-31'  group by  customer\_sk  """  #获取原始数据  dt\_raw=pd.read\_sql(query,con=connect)  #删除空缺值  dt\_raw=dt\_raw.dropna()  #查看数据  dt\_raw.head()  #######################################################################  绘制recency直方图  plt.hist(dt\_raw["recency"],histtype='bar',bins=40,rwidth=0.8,color='brown')  plt.show  #######################################################################  #绘制frequency直方图  plt.hist(dt\_raw["frequency"],histtype='bar',bins=40,rwidth=0.8,color='blue')  plt.show  #######################################################################  #绘制monetary直方图  plt.hist(dt\_raw["monetary"],histtype='bar',bins=40,rwidth=0.8,color='blue')  plt.show  ##############################  #去除异常点（某用户在某个变量的取值大于或小于所有用户均值3个标准差以上，为异常值）  #保证Kmeans聚类分析的合理性  #计算recency的标准差与均值  recency\_std=np.std(dt\_raw["recency"])  recency\_mean=np.mean(dt\_raw["recency"])  #计算frequency的标准差与均值  frequency\_std=np.std(dt\_raw["frequency"])  frequency\_mean=np.mean(dt\_raw["frequency"])  #计算monetary的标准差与均值  monetary\_std=np.std(dt\_raw["monetary"])  monetary\_mean=np.mean(dt\_raw["monetary"])  #去掉任一变量的取值在3倍标准差之外的用户记录  dt\_1=dt\_raw[(dt\_raw["recency"]<=(recency\_mean+3\*recency\_std))&  (dt\_raw["frequency"]<=(frequency\_mean+3\*frequency\_std))&  (dt\_raw["monetary"]<=(monetary\_mean+3\*monetary\_std))&  (dt\_raw["recency"]>=(recency\_mean-3\*recency\_std))&  (dt\_raw["frequency"]>=(frequency\_mean-3\*frequency\_std))&  (dt\_raw["monetary"]>=(monetary\_mean-3\*monetary\_std))]  #绘制删除异常点后的recency直方图  plt.title("删除异常点后的recency直方图")  plt.hist(dt\_1["recency"],histtype='bar',bins=40,rwidth=0.6)  plt.show  删除异常点后的recency直方图  绘制删除异常点后的frequency直方图  plt.title("删除异常点后的frequency直方图")  plt.hist(dt\_1["frequency"],histtype='bar',bins=40,rwidth=0.6)  plt.show  删除异常点后的frequency直方图  #绘制删除异常点后的monetary直方图  plt.title("删除异常点后的monetary直方图")  plt.hist(dt\_1["monetary"],histtype='bar',bins=40,rwidth=0.6)  plt.show  删除异常点后的monetary直方图  #数据标准化  dt\_2=dt\_1[["recency","frequency","monetary"]]  #新建数据框  dt\_3=pd.DataFrame()  #标准化计算  dt\_3["recency\_s"]=(dt\_2["recency"]-min(dt\_2["recency"]))/(max(dt\_2["recency"])-min(dt\_2["recency"]))  dt\_3["frequency\_s"]=(dt\_2["frequency"]-min(dt\_2["frequency"]))/(max(dt\_2["frequency"])-min(dt\_2["frequency"]))  dt\_3["monetary\_s"]=(dt\_2["monetary"]-min(dt\_2["monetary"]))/(max(dt\_2["monetary"])-min(dt\_2["monetary"]))  #查看标准化后的数据  dt\_3.head()  #保存文件  dt\_3.to\_csv(r"\\Mac\Home\Desktop高礼课程报告\standardized\_data.csv",index=False,header=True)  ########################  引入Kmeans算法包  from sklearn.cluster import KMeans  #新建列表用于存储不同簇数的畸变程度之和  SSE=[]  #对于1到9个聚类簇数进行循环  for k in range(1,9):  est=KMeans(n\_clusters=k).fit(dt\_3)  SSE.append(est.inertia\_)  #绘制手肘图，确定最佳聚类簇数  X=range(1,9)  plt.title("手肘图")  plt.xlabel("k")  plt.ylabel("SSE")  plt.plot(X,SSE,"o-")  plt.show()  截屏2022-05-10 19.54.59  #由图可得，当k=3时，聚类收益最高  k=3  #执行聚类分析  est=KMeans(n\_clusters=k).fit(dt\_3)  clusters=est.labels\_  dt\_3["labels"]=clusters  dt\_4=pd.concat([dt\_1,dt\_3["labels"]],axis=1)  #查看聚类结果  dt\_4.head()  cluster\_result=pd.merge(dt\_4[["customer\_sk","labels"]].groupby("labels",as\_index=False).count(),  dt\_4[["recency","frequency","monetary","labels"]].groupby("labels",as\_index=False).mean(),  on="labels",how="inner")  cluster\_result.columns=["类别","用户数","recency 均值","frequency 均值","monetary 均值"]  cluster\_result  截屏2022-05-10 20.08.09  #保存聚类结果  dt\_4.to\_csv(r"\\Mac\Home\Desktop\高礼课程报告\cluster\_result.csv",index = False,header=True)  cluster\_result.to\_csv(r"\\Mac\Home\Desktop\高礼课程报告\cluster\_result\_overall.csv",index=False,header=True) |

1. 聚类效果可视化

为了检验分类效果，本文采用Tableau中的散点图对从四个维度对聚类的结果进行可视化。

如下3图所示：



依图可得，聚类分析的结果在recency&monetary、recency&frequency上表现较好。

1. 结果分析

通过RFM用户价值聚类分析，不难看出，用户群体大致分为3类。其中0类的用户人数最多，有44746人，他们的recency、frequency、monetary都适中，属于中等价值用户，可以采取一定的营销手段进行挽留，避免其发展为低价值用户，同时通过进一步的市场营销与市场消费观念教育将其发展为高价值用户；而1类的用户人数适中，recency最低而且frequency和monetary均值都比另外两类用户高很多，说明这类用户近期有消费，且消费频次和消费金额都很高，可以定义为高价值用户，可以对其采取更加积极主动的优惠营销策略，最大限度刺激他们消费；2类的用户人数最少，recency显著高于另外两类用户，但frequency和monetary都低于另外两类用户，说明该类用户消费频次低，消费金额少，而且近期内基本没有消费行为，可以划分为低价值用户，几乎没有运营价值。