**基于 TPC-H 数据集的可视化数据分析**

首都经济贸易大学

程义淇

2019/12/3

目录

[一、 摘要 2](#_Toc26629576)

[1. 作业要求 2](#_Toc26629577)

[2. 技术实现 2](#_Toc26629578)

[3. 主要发现 2](#_Toc26629579)

[二、 数据来源与说明 3](#_Toc26629580)

[三、 数据预处理 4](#_Toc26629581)

[1. SQL Server 中创建 TPC-H 数据库 4](#_Toc26629582)

[2. 利用SQLServer生成大宽表 4](#_Toc26629583)

[3. Tableau 6](#_Toc26629584)

[4. Python 7](#_Toc26629585)

[四、 可视化描述性分析 8](#_Toc26629586)

[1. 主题分析 8](#_Toc26629587)

[2. Tableau仪表盘设计 12](#_Toc26629588)

[五、 数据库分析 12](#_Toc26629589)

[1. 复杂查询1 13](#_Toc26629590)

[2. 复杂查询2 13](#_Toc26629591)

[3. 复杂查询3 14](#_Toc26629592)

[六、 数据挖掘 15](#_Toc26629593)

[1. 选取指标 15](#_Toc26629594)

[2. 数据导入与预处理 16](#_Toc26629595)

[3. 数据概述 17](#_Toc26629596)

[4. 客户价值分析 18](#_Toc26629597)

[七、 总结与改进方向 21](#_Toc26629598)

1. 摘要
2. 作业要求

基于 TPC-H 数据集，模拟淘宝数据分析场景，通过SQLserver、python及Tableau综合展示数据内部的规律，并最终形成数据分析报告。

1. 技术实现
2. SQLserver 复杂查询、多表连接嵌套、建库建表
3. Jupyter（python）、Tableau、PowerBI与本地SQLserver 相连以及数据传输
4. Python语言下的客户价值分析
5. 主要发现
6. 已有记录的客户可以分成三类：

重要保持客户多年来均保持了购买习惯，订单的数量多且总金额大，是需要重点维持的客户；

重要发展客户同样多年来保持在此处购买的习惯，但购买力不如第一类用户。因为这类用户一直保持着购买行为，所以应刺激这类人群的购买行为，进行一系列营销；

低价值用户在几年内并未流失，但订单数量和总金额都很少，可能是相隔若干年再次进行购买，这类人购买需求并不高。可尝试营销但未见成果应果断放弃。

1. 客户在各维度的属性上均非常均匀（猜测是数据自动生成的原因），如来源国家地区、喜好零件品牌、种类、容器、购买年份、购买月份、可接受零件成本、运输方式等等，难以根据其自身属性总结规律进行营销。
2. 数据来源与说明

使用 SQL Server 中创建 TPC-H 数据库。各表结构如下：

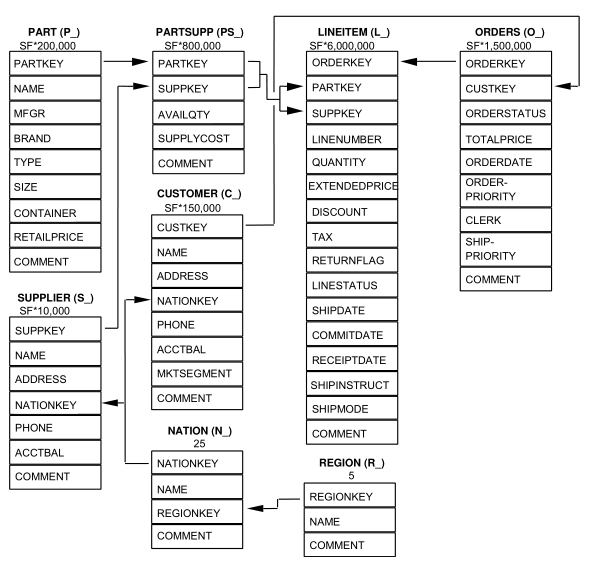


图2-1-1 数据集结构

1. 数据预处理

1. SQL Server 中创建 TPC-H 数据库

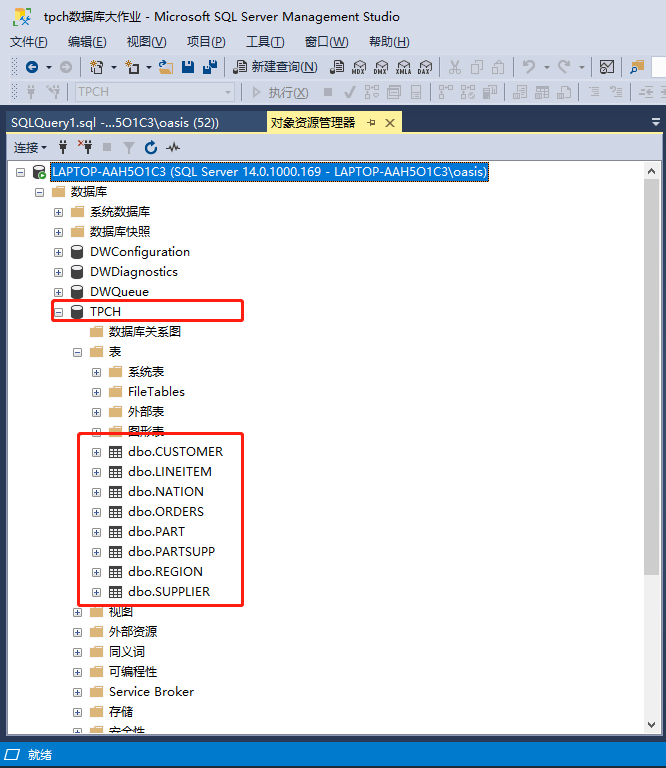
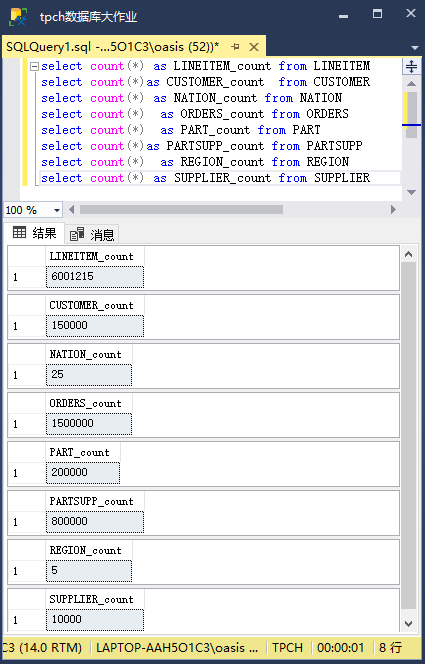


图3-1-1 数据库

1. 利用SQLServer生成大宽表

从上图的数据量可看出，各个表之间具有复杂的连接关系，且数据量较大，若直接导入绘图软件容易造成计算机死机，故先在SQLServer中进行数据处理，生成具有研究价值的宽表后进行下一步分析，将该表命名为“PARTSALE”，每一行记录代表一个订单中一个零件的购买记录。

1. **SELECT** \* **into** TEST1 **from** (**select**
2. L\_ORDERKEY,L\_PARTKEY, L\_LINENUMBER,L\_QUANTITY,L\_EXTENDEDPRICE,
3. L\_DISCOUNT,L\_TAX,L\_SHIPDATE,L\_SHIPMODE,O\_CUSTKEY,O\_TOTALPRICE,O\_ORDERPRIORITY
4. **from** LINEITEM,ORDERS **where** L\_ORDERKEY=O\_ORDERKEY)a
6. **select** \* **into** TEST2 **from** (**select** TEST1.\*,C\_CUSTKEY,C\_NATIONKEY,C\_MKTSEGMENT
7. **from** TEST1,CUSTOMER **where** TEST1.O\_CUSTKEY=C\_CUSTKEY)b
9. **select** \* **into** TEST3 **from** (**select** TEST2.\*,P\_NAME,P\_BRAND,P\_TYPE,P\_SIZE,P\_CONTAINER,P\_RETAILPRICE
10. **from** TEST2,PART **where** TEST2.L\_PARTKEY=P\_PARTKEY)c
12. **select** \* **into** TEST4 **from** (**select** TEST3.\*,N\_NAME,N\_REGIONKEY
13. **from** TEST3,NATION **where** C\_NATIONKEY = N\_NATIONKEY )d
15. **select** \* **into** PARTSALE **from** (**select** TEST4.\*,R\_NAME
16. **from** TEST4,REGION **where** N\_REGIONKEY = R\_REGIONKEY )e

|  |  |  |
| --- | --- | --- |
| **宽表数据字典** | | |
| **字段名** | **含义** | **数据类型** |
| L\_ORDERKEY | 订单key | int |
| L\_PARTKEY | 零件key | int |
| L\_LINENUMBER | 该零件在该订单中序号 | int |
| L\_QUANTITY | 购买数量 | float |
| L\_EXTENDEDPRICE | 该订单中该零件总价 | float |
| L\_DISCOUNT | 折扣率 | float |
| L\_TAX | 税率 | float |
| L\_SHIPDATE | 发货日期 | str |
| L\_SHIPMODE | 发货方式 | str |
| O\_CUSTKEY | 客户key | int |
| O\_TOTALPRICE | 订单总金额 | float |
| O\_ORDERPRIORITY | 订单优先级 | str |
| C\_CUSTKEY | 客户key | int |
| C\_NATIONKEY | 国家key | int |
| C\_MKTSEGMENT | 类别 | str |
| P\_NAME | 零件名 | str |
| P\_BRAND | 零件品牌 | str |
| P\_TYPE | 零件种类 | str |
| P\_SIZE | 零件尺寸 | int |
| P\_CONTAINER | 零件容器 | str |
| P\_RETAILPRICE | 零件单价 | float |
| N\_NAME | 客户所属国家名 | str |
| N\_REGIONKEY | 地区key | int |
| R\_NAME | 客户所属地区名 | str |

1. Tableau

实现Tableau与数据库的连接



图3-3-1 连tableau

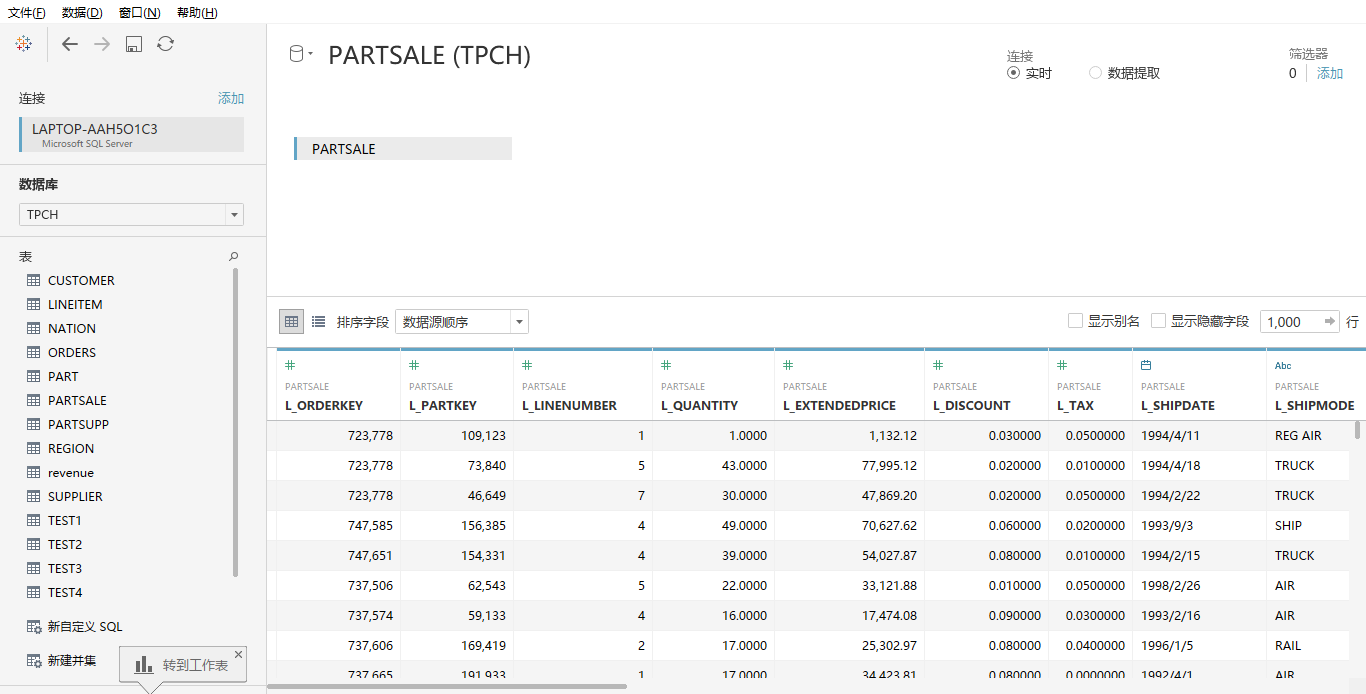


图3-3-2 tableau导数据

1. Python

实现python与数据库的连接

1. **import** pyodbc
2. **import** pandas as pd
3. **import** numpy as np
4. **from** pandas **import** DataFrame
6. #连接本地数据库
7. cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER=LAPTOP-AAH5O1C3;DATABASE=TPCH;UID=sa;PWD=000')
8. cursor = cnxn.cursor()
9. #列名
10. columnDes=cursor.execute("select top 2 \* from PARTSALE").description
11. columnNames = [columnDes[i][0] **for** i **in** range(len(columnDes))]
13. #一次导入全部数据600w+的数据量单机的python无法支持，故选取1993一年的为例，数据量约100w
14. data=DataFrame( list(i) **for** i **in** cursor.execute("select  \* from PARTSALE where L\_SHIPDATE like '1993%'"))
15. data.columns=columnNames

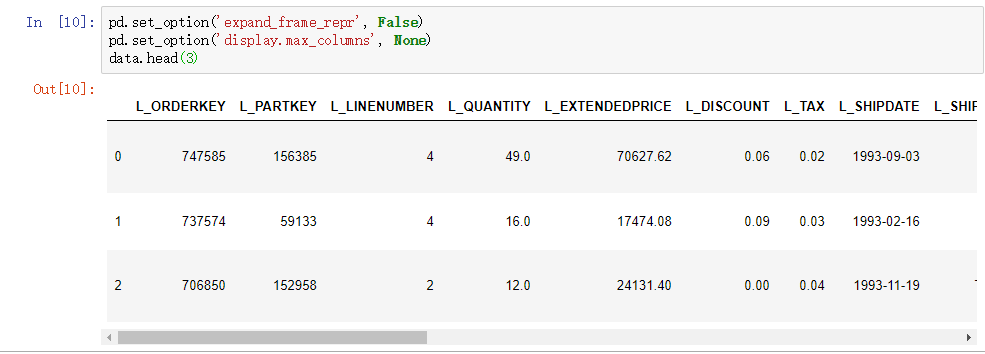


图3-4-1 连python

1. 可视化描述性分析
2. 主题分析

分析 TPC-H 数据集的结构和数据特征，从 customer、supplier、part 三个维度来分析销售数据在不同视角，不同分析粒度上的数据规律与特征，综合运用数据可视化组件展现数据特点。

1. 客户维度分析

如下图所示，购买者来源国家分布均匀，以法国最多。

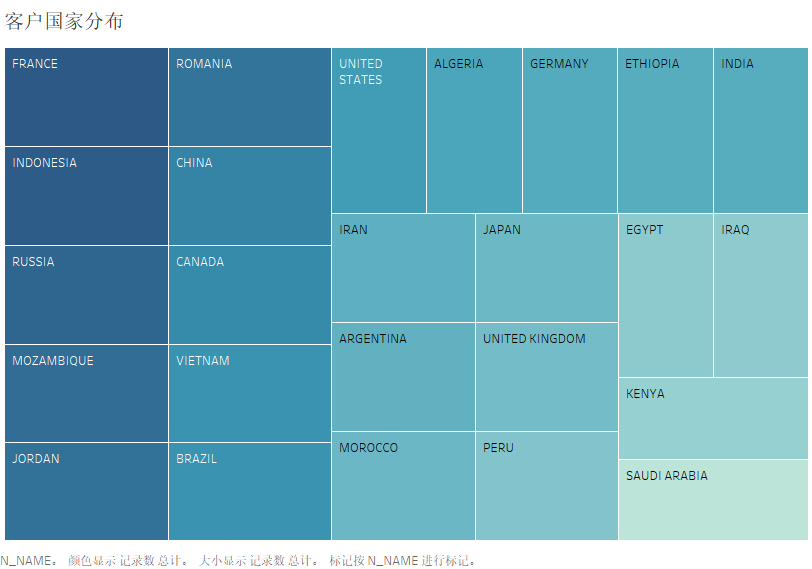


图4-1-1 客户国家

如下图所示，购买者地区来源均匀，以欧洲最多

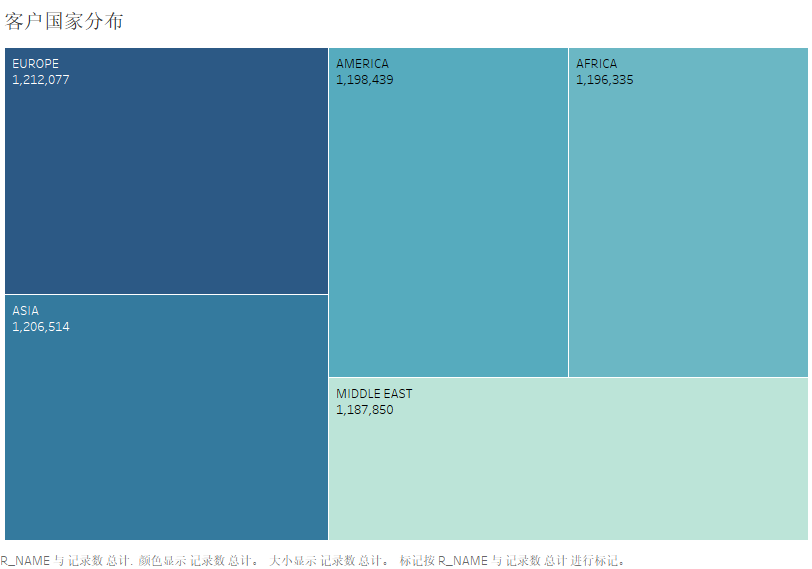


图4-1-2 客户地区

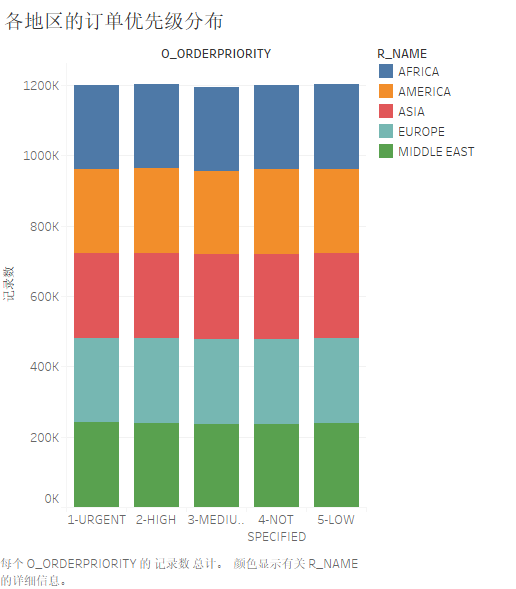


图4-1-3 各地区优先级分布

1. 供应商维度分析

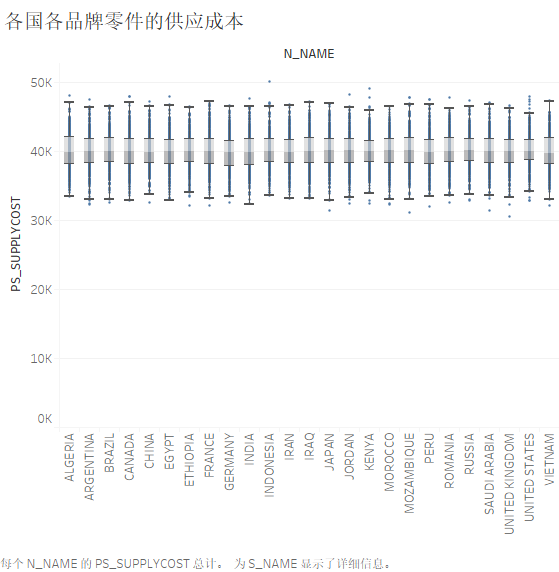
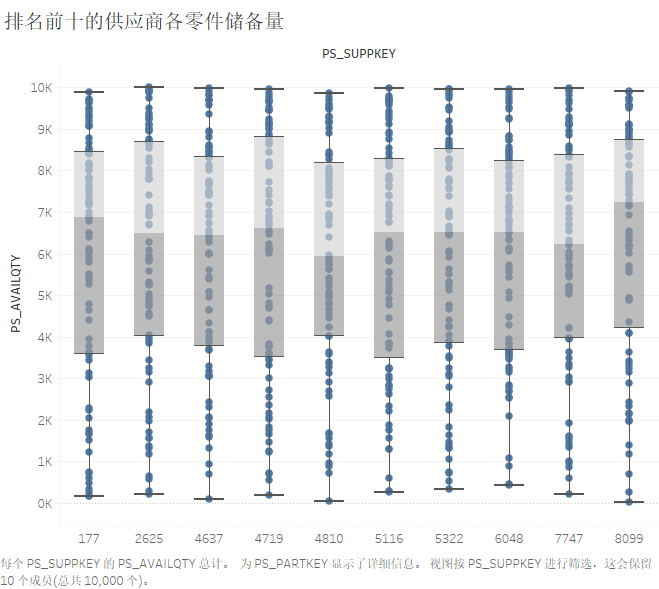


图4-1-4 供应商零件供应情况

1. 零件维度分析

不能确认初始年与末年为全年数据，故认为零件售卖各年较为均匀，且从各月份来看春夏销量较高，秋冬销量较低。

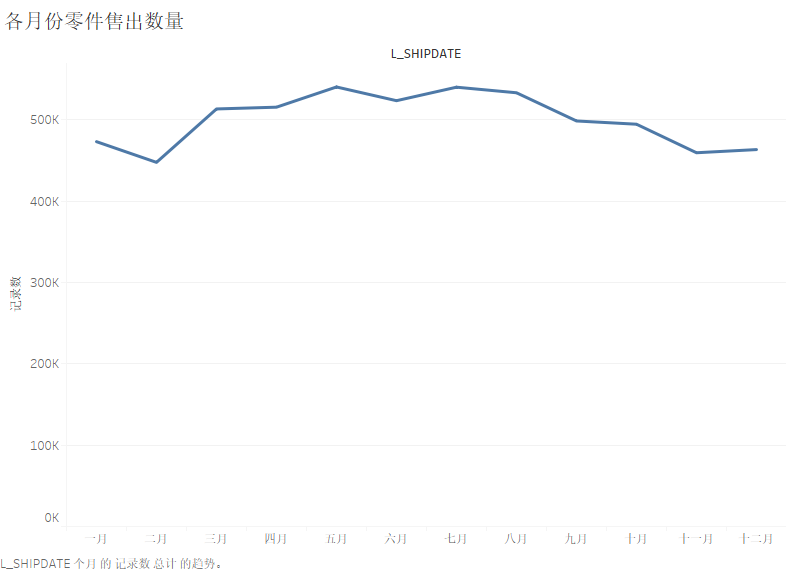
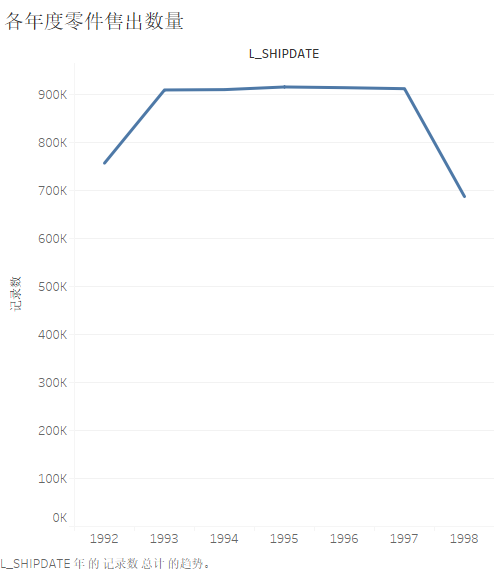


图4-1-5 时间序列因素

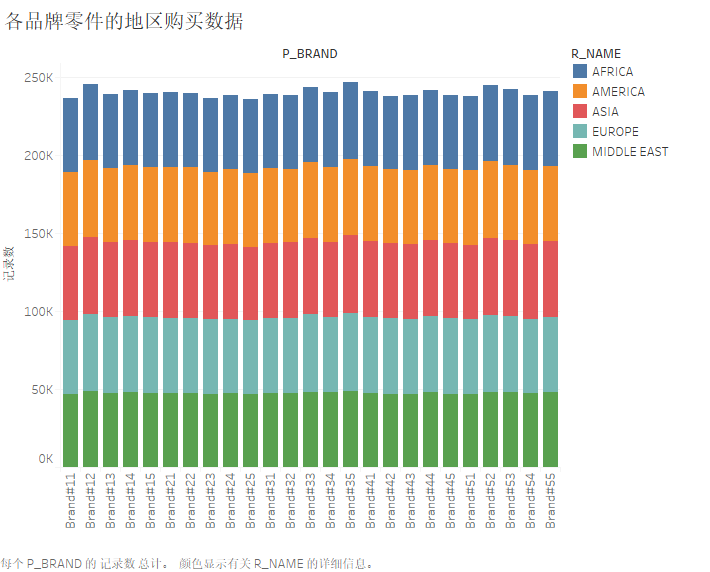


图4-1-6 各品牌零件在各地区受欢迎程度

1. Tableau仪表盘设计

这是一个全局的仪表板页，设计面向 TPC-H 数据集的分析方案，通过多个图表控件设计一个能够良好展现数据宏观、微观及特定分析主题的综合数据视图，能够说明仪表板的使用及所展现的数据分析结果。

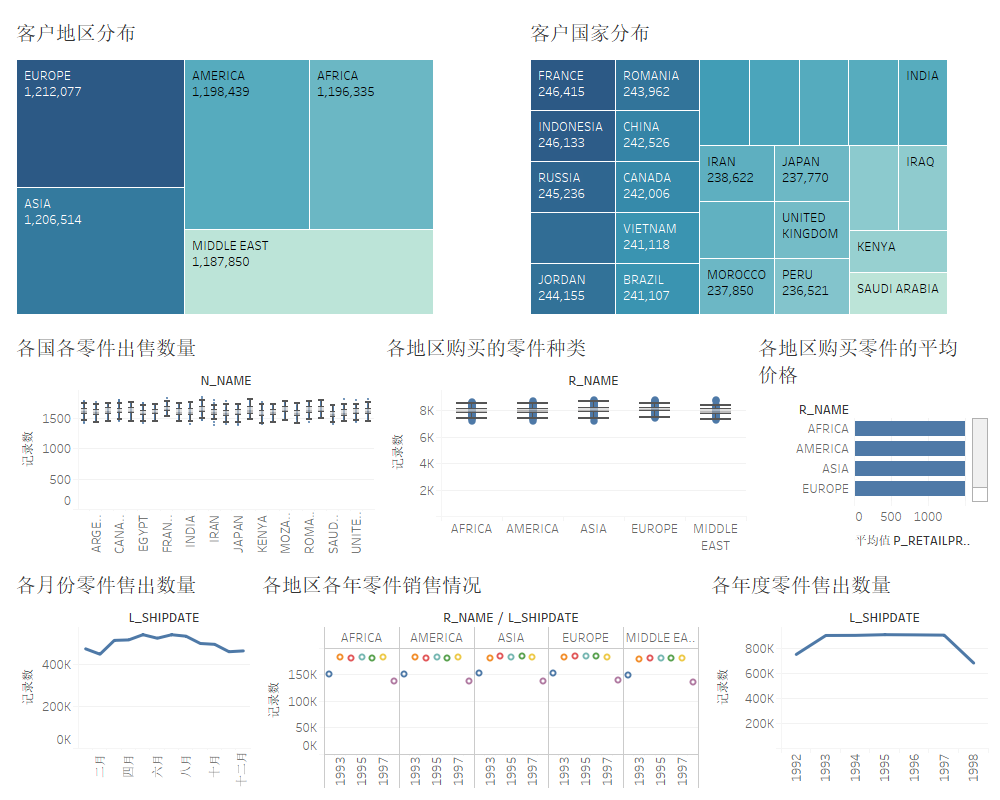


图4-2-1 Tableau仪表盘

1. 数据库分析

通过 SQL 语言构建常规多维分析无法解决的复杂分析任务，含构建复杂数据分析视图。设计 3 个基于派生查询的复杂查询任务，实现面向现实应用需求、基于原始数据再加工的分析任务，并给出查询优化方案。

1. 复杂查询1

查询获得某一个月的收入中有多大的百分比是来自促销零件。用以监视促销带来的市场反应。

由以下语句得出1995年6月的收入中有16.81%来自于促销零件

1. **select**
2. 100.00 \* sum(case
3. **when** p\_type like 'PROMO%'
4. **then** l\_extendedprice\*(1-l\_discount)
5. **else** 0
6. **end**) / sum(l\_extendedprice \* (1 - l\_discount)) **as** promo\_revenue
7. **from**
8. lineitem, part
9. **where**
10. l\_partkey = p\_partkey
11. and l\_shipdate >=  '1995-06-01'
12. and l\_shipdate <  '1995-07-01'
13. 复杂查询2

查询获得某段时间内为总收入贡献最多的供货商（排名第一）的信息。可用以决定对哪些头等供货商给予奖励、给予更多订单、给予特别认证、给予鼓舞等激励。带有分排序、聚集、聚集子查询操作并存的普通表与视图的连接操作。

由语句得出

|  |  |
| --- | --- |
| total\_revenue | 2932 |
| s\_name | Supplier#000002932 |
| s\_address | gAIrgSCdvtJltNKuZKRGYeYLRf |
| s\_phone | 14-722-108-2914 |
| s\_suppkey | 1756106.186 |

1. **WITH** revenue (supplier\_no, total\_revenue) **as** (**SELECT** l\_suppkey,SUM(l\_extendedprice \* (1-l\_discount))
2. **FROM** lineitem **WHERE** l\_shipdate >= '1995-06-07' AND l\_shipdate < '1995-09-07' **GROUP** **BY** l\_suppkey)
3. **SELECT** s\_suppkey,s\_name,s\_address,s\_phone,total\_revenue **FROM** supplier,revenue
4. **WHERE** s\_suppkey = supplier\_no AND total\_revenue = (**SELECT** **MAX** (total\_revenue) **FROM** revenue)
5. **ORDER** **BY** s\_suppkey;
6. 复杂查询3

  查询获得能够以指定的贡献条件供应零件的供货商数量。可用于决定在订单量大，任务紧急时，是否有充足的供货商。带有分组、排序、聚集、去重、NOT IN子查询操作并存的两表连接操作

得出结果如下：

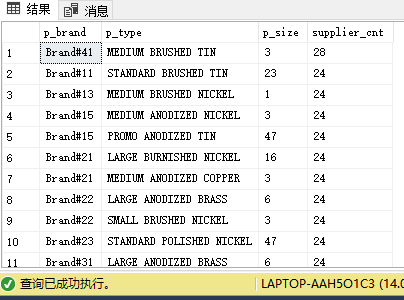


图5-3-1

1. **select**
2. p\_brand,
3. p\_type,
4. p\_size,
5. count(**distinct** ps\_suppkey) **as** supplier\_cnt --聚集、去重操作
6. **from**
7. partsupp,
8. part
9. **where**
10. p\_partkey = ps\_partkey
11. and p\_brand <> 'Brand#34'
13. and p\_size in (1, 3, 4, 43, 23, 16, 47, 6)
14. --TYPEX是在1到50之间任意选择的一组八个不同的值
15. and ps\_suppkey not in ( --NOT IN子查询，消费者排除某些供货商
16. **select**
17. s\_suppkey
18. **from**
19. supplier
20. **where**
21. s\_comment like '%Customer%Complaints%'
22. )
23. **group** **by** --分组操作
24. p\_brand,
25. p\_type,
26. p\_size
27. **order** **by** --排序操作
28. supplier\_cnt **desc**, --按数量降序排列，按品牌、种类、尺寸升序排列
29. p\_brand,
30. p\_type,
31. p\_size;
32. 数据挖掘

基于 Python实现对 TPC-H 数据的深度分析，构建客户价值分析模型。通过客户分类，对客户群体进行细分，区别出低价值客户、高价值客户，对不同的客户群体开展不同的个性化服务，将有限的资源合理地分配给不同价值的客户，实现效益最大化。

分析中使用Kmeans聚类，且将数据分析的结果可视化。

1. 选取指标

在客户分类中，RFM模型是一个经典的分类模型，模型利用通用交易环节中最核心的三个维度——最近消费(Recency)、消费频率(Frequency)、消费金额(Monetary)细分客户群体，从而分析不同群体的客户价值。根据此例数据，选取同一客户的总计订单数量、总计消费金额、所有购买商品的平均折扣率以及第一次订单和最后一次订单的时间间隔作为四个指标。计算逻辑如下：

|  |  |
| --- | --- |
| ordercount | 对ORDER表，以CUSTKEY为key，计数 |
| totalprice | 对ORDER表，以CUSTKEY为key，ORDERPRICE的和 |
| dateinter | 对ORDER表，以CUSTKEY为key，求最远时间与最近时间的差 |
| meandiscount | 对LINEITEM和ORDER的外联表，以CUSTKEY为key，求L\_DISCOUNT的和 |

1. 数据导入与预处理
2. **import** pyodbc
3. **import** pandas as pd
4. **import** numpy as np
5. **from** pandas **import** DataFrame
7. ###  数据预处理
8. #连接本地数据库
9. cnxn = pyodbc.connect('DRIVER={SQL Server};SERVER=LAPTOP-AAH5O1C3;DATABASE=TPCH;UID=sa;PWD=000')
10. cursor = cnxn.cursor()
11. #列名
12. columnDes=cursor.execute("select top 2 \* from PARTSALE").description
13. columnNames = [columnDes[i][0] **for** i **in** range(len(columnDes))]
14. **print**(columnNames)
15. #一次导入全部数据600w+的数据量单机的python无法支持，故选取 C\_CUSTKEY>13000 and C\_CUSTKEY<15000
16. data=DataFrame( list(i) **for** i **in** cursor.execute("select \* from PARTSALE where C\_CUSTKEY>13000 and C\_CUSTKEY<15000"))
17. data.columns=columnNames
18. pd.set\_option('expand\_frame\_repr', False)
19. pd.set\_option('display.max\_columns', None)
21. 数据概述

### 数据的概况

1. explore=data.describe(percentiles=[],include='all').T
2. explore['null']=len(data)-explore['count']
3. #大宽表
4. explore

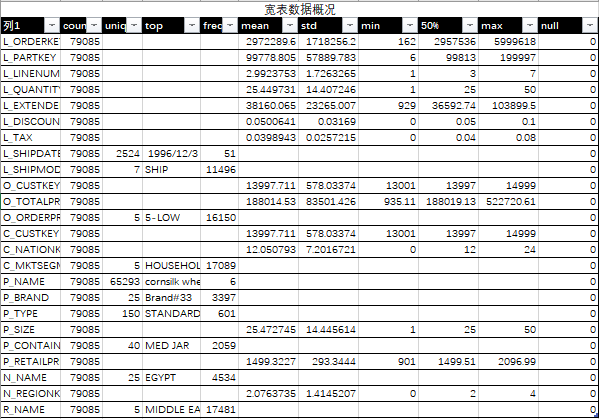


图6-3-1 宽表数据概况

1. #生成客户价值模型需要的数据集
2. df1\_0=data[['L\_ORDERKEY','C\_CUSTKEY','L\_LINENUMBER','O\_TOTALPRICE','L\_SHIPDATE','L\_DISCOUNT','O\_ORDERPRIORITY']]
3. #用订单表和宽表的折扣率字段做客户价值分析
4. order=DataFrame( list(i) **for** i **in** cursor.execute("select \* from ORDERS where O\_CUSTKEY <15000 and O\_CUSTKEY>13000"))
6. #订单表列名
7. cd\_order=cursor.execute("select top 2 \* from ORDERS").description
8. cn\_order = [cd\_order[i][0] **for** i **in** range(len(cd\_order))]
9. order.columns=cn\_order
10. explore\_or=order.describe(percentiles=[],include='all').T
11. explore\_or['null']=len(order)-explore\_or['count']
13. #订单表
14. explore\_or



图6-3-2 订单表数据概况

1. 客户价值分析
2. order=order.drop(['O\_CLERK','O\_COMMENT'],axis=1)
3. #一个客户的订单数量、总金额、持续时间、平均折扣
4. test = order.groupby('O\_CUSTKEY').agg({'O\_ORDERKEY':'count','O\_TOTALPRICE':'sum','O\_ORDERDATE':'max'})
5. test2=order.groupby('O\_CUSTKEY').agg({'O\_ORDERDATE':'min'})
6. test3 = pd.concat((test,test2),axis=1)
7. test3.columns=['ordercount','totalprice','lastdate','firstdate']
8. test3['dateinter']=pd.to\_datetime(test3['lastdate'])-pd.to\_datetime(test3['firstdate'])
9. #增加平均折扣率
10. #导进来lineitem 按custkey算discount平均值
11. df1\_0.head()
12. test4 = df1\_0.groupby('C\_CUSTKEY').agg({'L\_DISCOUNT':'mean'})
13. test5 = pd.concat((test3,test4),axis=1)
14. use0=test5.drop(['lastdate','firstdate'],axis=1)
15. use0=use0.rename(columns={'L\_DISCOUNT':'meandiscount'})
16. use0['dateinter']=(use0['dateinter'] / np.timedelta64(1, 'D')).astype(int)
17. #标准化
18. use1=use0.apply(**lambda** x: (x - np.min(x)) / (np.max(x) - np.min(x)))
19. #数据建模
20. **from** sklearn.cluster **import** KMeans
21. kmodel = KMeans(n\_clusters=3)
22. kmodel.fit(use1)
23. tezheng=pd.DataFrame(kmodel.cluster\_centers\_)
24. lei=pd.DataFrame(kmodel.labels\_)
25. lei.columns=['lei']
26. use00=use0.reset\_index()
27. final=pd.concat((use00,lei),axis=1)
28. final

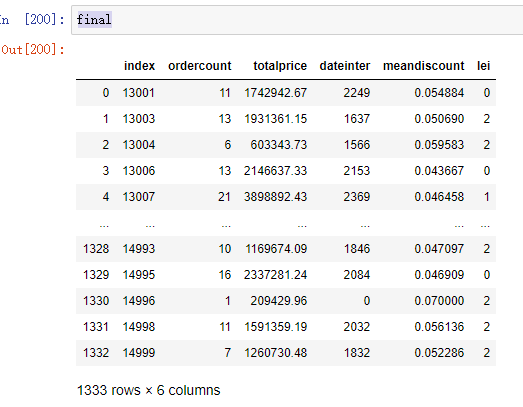


图6-4-1 客户价值分类打标签后结果



图6-4-2 客户价值分析结果

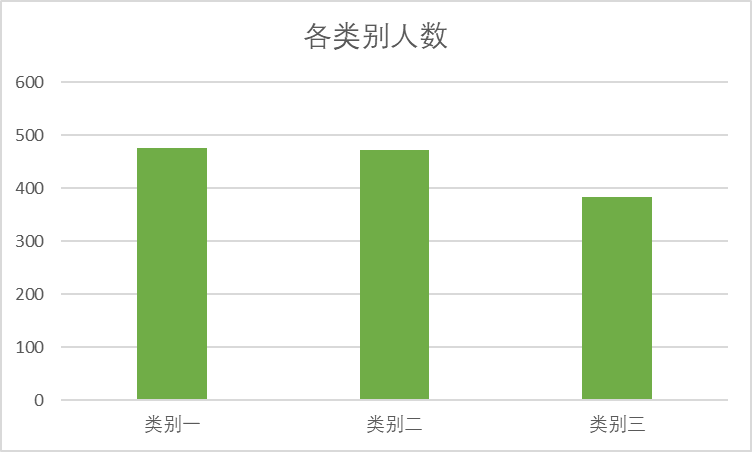
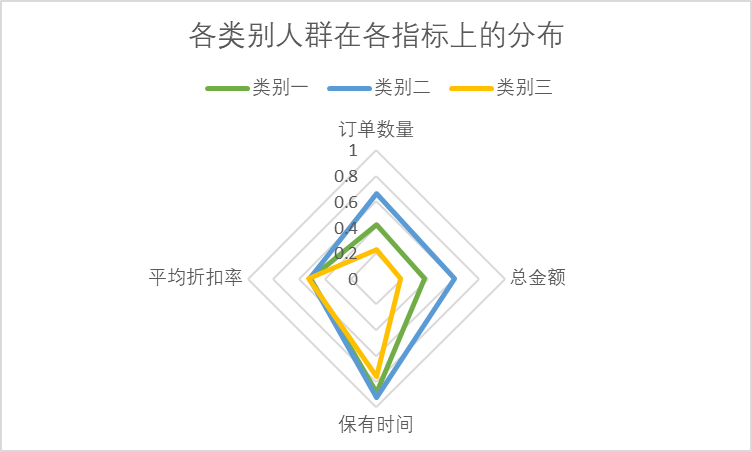


图6-4-3 各类别情况

分析：

1. 各类别的人数分布较为平均
2. 类别二在各属性上数值都最大
3. 类别三在各属性上的数值都最小
4. 三个类别在保有时间和平均折扣率两项上的指标数量相差不大

针对业务需要，及参考模型对客户类别的分类，定义三个类别的人群特征：

1. 重要保持客户

这类客户多年来均保持了购买习惯，订单的数量多且总金额大，是需要重点维持的客户。

1. 重要发展客户

这类客户同样多年来保持在此处购买的习惯，但购买力不如第一类用户。因为这类用户一直保持着购买行为，所以应刺激这类人群的购买行为，进行一系列营销。

1. 低价值用户

这类客户在几年内并未流失，但订单数量和总金额都很少，可能是相隔若干年再次进行购买，可见这类人购买需求并不高。

1. 总结与改进方向
2. 本文档实现了从数据到结论的全流程框架。其中主要涉及到的技术环节包含：
3. SQLserver建立数据库数据表以及生成数据
4. SQLserver上的多表连接多条件复杂查询任务
5. SQLserver将同一分析任务下的多张表的数据连接起来生成容易调用的一张大表
6. 将SQLserver生成的表上传至hadoop平台，尝试全数据量的数据探索
7. SQLserver与python的IDE Jupyter成功连接，可在Jupyter中通过写sql语句的方式导入数据到本地，再进行后续一系列python的操作
8. SQLserver与Tableau、powerBI的连接（因功能有重复，只用了Tableau做分析）
9. 使用python做基于k均值聚类的客户价值模型，并能实现给每个客户打上标签
10. 本文档的不足之处有：
11. 因为数据集是自动生成的，里面没有缺失值，没有异常值，导致数据清洗工作中的很大一部分被忽略掉了，本文档没有体现出来。
12. 同样因为数据集自动生成，各维度上的数据分布没有显著差异，导致此例中的描述性分析步骤无太多有价值的发现，可视化方面较为欠缺。