公司实体店销售数据分析

课　　程：　　数据库课程

实验名称：　　数据库个人分析报告

指导老师：　　张延松

专业班级：　　中财大数据班

实验日期：　　2017.12.7

报告成绩：

目录

一、选题依据

1.研究意义

销售数据是公司日常经营、进出货、维护管理客户等所必要的数据，暗含了一定的商业规律，对公司来说是提高经营水平、管理效率所不可或缺的。本文采用描述统计方法和数据挖掘方法，分析某公司实体店的销售数据（TPCDS 数据库中只涉及实体店的表格），帮助管理者更好地掌握经营情况，同时发现对商品销售的有利因素，这对实体店经营管理有一定的意义。

2.研究的具体问题与研究方法

（1）实体店销售额影响因素分析。

主要分析实体店的平均销售价格，销售商品种类，退货，顾客数等对销售额的影响，时间单位为月份。前期需要使用 SQL 把所需数据集中在一张表格，后期的数据挖掘使用 R 语言做纵向数据分析。

（2）实体店销售情况分析。

统计 12 家实体店的年销售额，年利润，发现销售发展状况优秀的门店及销售额和利润的关系。需要使用 services analysis 进行数据分析，使用 excel 和 powerBI 展示数据分析的结果，不涉及数据挖掘。

（3）实体店销售的最佳时间分析。

分别统计一天中，一周中，一年中销售最好的时间段，给门店销售提供建议。需要使用 services analysis 进行数据分析，使用 excel 和 powerBI 展示数据分析的结果，不涉及数据挖掘。

（4）商品畅销影响因素分析。

主要分析商品的品牌，类别，目录，颜色，尺码，平均价格等对销售额的影响。需要使用 analysis services 进行数据分析，使用 excel 和 powerBI 展示数据分析的结果，再使用 R 和 analysis services 对商品进行聚类完成数据挖掘任务。

（5）客户群地域特点分析。

分析不同地域的客户，购买额、购买商品种类、客户数量的差别，前期需要使用 SQL 把所需数据集中在一张表格，之后使用 powerBI 静态展示分析结果，power map 动态展示分析结果，不涉及数据挖掘。

二、设计分析数据库

1.使用表格及依据

使用表格及使用依据都是根据数据分析子任务确定的。

表 2- 1 使用表格、变量说明及依据

表名	中文名	主要使用变量	依据：涉及该表格的子任务
Store_sale	店铺销售	交易日期、时间、销量、销售价、净利润	（1）（2）（3）（4）（5）
Store	店铺	所在州	（1）（2）（3）
Store_return	店铺退货	退货日期、时间、退货净损失	（1）
Item	货物	品牌、类别、目录、颜色、尺码	（4）（5）
Customer	顾客	顾客 ID	（5）
Customer_address	用户地址	所在州	（5）
Date_dim	日期	年、月、季度	（1）（2）（3）（5）
Time_dim	时间	上下午、时	（3）

2.表与表之前的关系

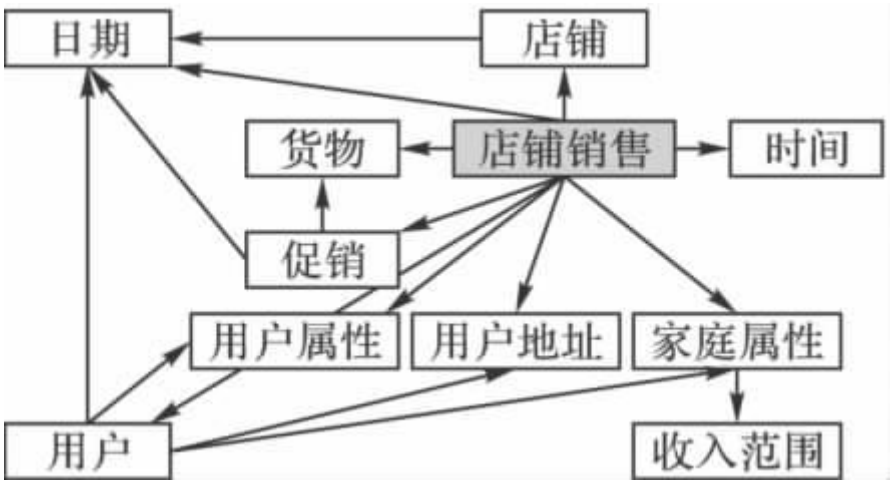

图 2-2 各个表格之间的关系

3.建表命令、约束条件命令

由于代码较长，放报告末尾的附录 1 和附录 2。

三、数据分析任务

为了从多维度探究实体店销售的情况，涉及的数据结构有多维数据集和普通二维表，主要实现的技术有 SQL 查询、analysis service 的多维数据集的设计，excel 数据透视表，数据透视图，power BI 报表设计，power map 地图数据展示。

1.SQL Server 初步统计

首先使用 SQL 查询语言对五个数据分析子任务进行初步的数据统计，使用的技术为分类汇总。有的子任务可通过查询结果直接发现其中规律（如子任务（2）（3）（5）），有的子任务需要查询生成的表格进行下一步分析或挖掘工作（如子任务（1）（4））。

（1）实体店销售额影响因素分析。使用 SQL 查询语言生成数据挖掘所需要的表格，包括日期、可能影响销售额的变量。

```
select s_store_sk,d_year,d_moy,s_city,s_state,
sum(cast(ss_sales_price*ss_quantity as bigint))/sum(cast(ss_quantity as bigint)) as
```

```
avg_price,
count(ss_item_sk) as diversity,count(ss_customer_sk) as customer_flow,
sum(cast(sr_net_loss as bigint)) as return_loss,
sum(cast(ss_sales_price*ss_quantity as bigint)) as sales
into store_sales_analysis
from store,store_sales,store_returns,date_dim,promotion
where s_store_sk=ss_store_sk
and ss_sold_date_sk=d_date_sk
and s_store_sk=sr_store_sk
and d_date_sk=sr_returned_date_sk
group by s_store_sk,d_year,d_moy,s_city,s_state
order by  s_store_sk,d_year,d_moy,s_city,s_state
```

（2）实体店销售情况分析。使用 SQL 查询语言统计 12 家实体店的年销售额，年利润，发现销售额越多的实体店，净利润（负值）则越小，即亏损越大。

```
select s_store_sk,d_year,d_date,
sum(ss_sales_price*ss_quantity) as sales,sum(ss_net_profit*ss_quantity) as profits
into store_sales_analysis2
from store,store_sales,date_dim
where s_store_sk=ss_store_sk
and ss_sold_date_sk=d_date_sk
group by s_store_sk,d_year,d_date
order by  d_year,s_store_sk desc --第 7 第 1 家店销售最多，亏损最多
```

（3）实体店销售的最佳时间分析。使用 SQL 查询语言统计一天中，一周中，一年中销售最好的时间段，发现第四季度，11、12 月，周末销售额明显偏高。

```
select top 100 d_qoy,d_moy,d_day_name,t_hour, --第四季度12月
sum(ss_sales_price*ss_quantity) as sales
into store_time_analysis
from date_dim,time_dim,store_sales
where ss_sold_date_sk=d_date_sk
and ss_sold_time_sk=t_time_sk
group by d_qoy,d_moy,d_day_name,t_hour
order by sales desc
```

（4）商品畅销影响因素分析。使用 SQL 查询语言统计商品的品牌，类别，目录，颜色，尺码，平均价格，销售额，从查询结果可以看出商品的品牌，类别，目录，颜色有非常多的分类，并不适合做回归等数据挖掘，查询结果保存成表格以供后续数据挖掘使用。

```
select i_item_sk,i_brand,i_class,i_category,i_size,i_color,i_current_price,
sum(ss_sales_price*ss_quantity) as sales
into item_sales_analysis
from item,store_sales
```

```
where i_item_sk=ss_item_sk
group by i_item_sk,i_brand,i_class,i_category,i_size,i_color,i_current_price
order by i_item_sk,i_brand,i_class,i_category,i_size,i_color,i_current_price
```

（5）客户群地域特点分析。使用 SQL 查询语言统计不同州的客户的购买额、购买商品种类、客户数量的差别，从查询结果可以看出 TX 州顾客最多，因为该公司的实体店都在 TX 州，所以 TX 州顾客贡献销售额、购买的商品种类、人均贡献销售额最多，但 GA 州的人均贡献销售额与 TX 州并列第一。查询结果保存成表格以供后续制作地图使用。

```
select ca_state ,count(c_customer_sk) as customer_count,
avg(ss_sales_price*ss_quantity) as avg_comtri_buy,
sum(ss_sales_price*ss_quantity) as comtri_buy,
count(ss_item_sk) as item_kinds
into customer_analysis
from customer,customer_address,store_sales
where c_current_addr_sk=ca_address_sk
and c_customer_sk=ss_customer_sk
group by ca_state
order by count(c_customer_sk) desc
```

2. Analysis Service 设计多维数据集

（1）度量

store sales 表格，这是所有子任务都涉及的表格，主要是对 sales（=quantity*sales_price）进行分组聚合。

图 3-2-1 度量值设置

(2) 维度和层次结构

多维数据集的维度和层次设计如下表 3-2-2 所示.

表 3-2-2 多维数据集的维度和层次设计

维度	层次结构	层数	设计思想
日期维	年、季度、月份、星期几	四层	用于实体店销售的一年中最佳时间分析
顾客地理位置维	国家、州、城市	三层	用于客户群地域特点分析中展示地图数据
日时间维	上午或下午、时	二层	用于实体店销售的一天中最佳时间分析
商品属性维	品牌，类别，目录，颜色，尺码，平均价格	无	用于商品畅销影响因素分析中分类统计商品销售额

图 3-2-2 维度与层次设置

3.Excel 数据透视表展示分析结果

（1）实体店销售的最佳时间分析。分别统计一天中，一周中，一年中销售最好的时间段。

图 3-3-1-1 统计了数据集中所有门店所有天数内按小时分组的销售额（quantity*sale_price）加总，可以看出，实体店营业时间为早上 8-11 点，下午 12-20 点，其中 10 点、14 点、17 点为销售额最高的时候，实体店经历可在这个三个小时在店内多安排一些人手。

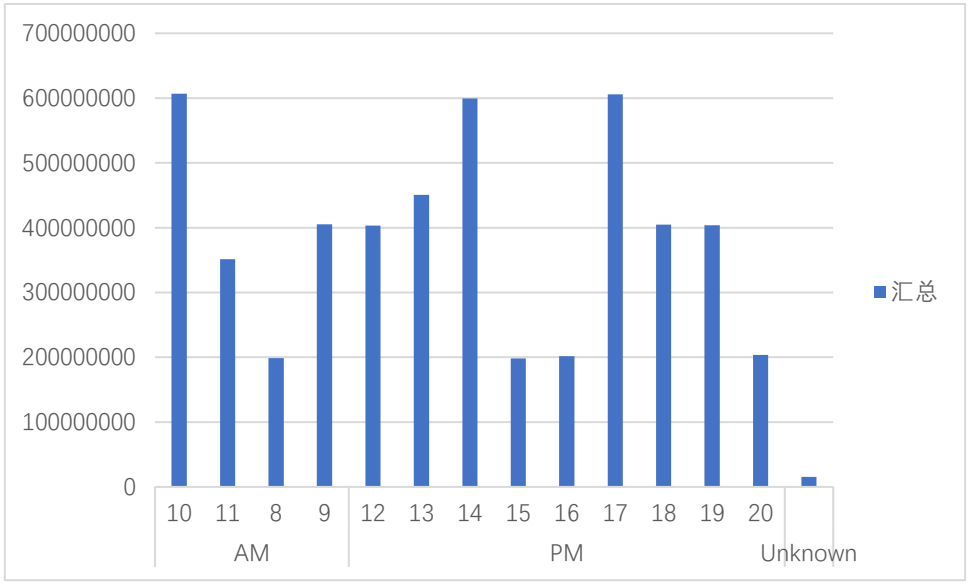

图 3-3- 1-1 一天中开业时间的销售额情况

图 3-3-1-2 统计了数据集中所有门店所有天数内按季度分组的销售额加总，可以看出，该公司的实体店每年第四季度是旺季，第四季度的销售额可达到第一、二季度销售额的 3 倍。 而图 3-3-1-3 反映了实体店在一年中，从 1 月到 7 月销售额几乎持平，8 月-12 月逐步上升，12 月销售额达到顶峰。

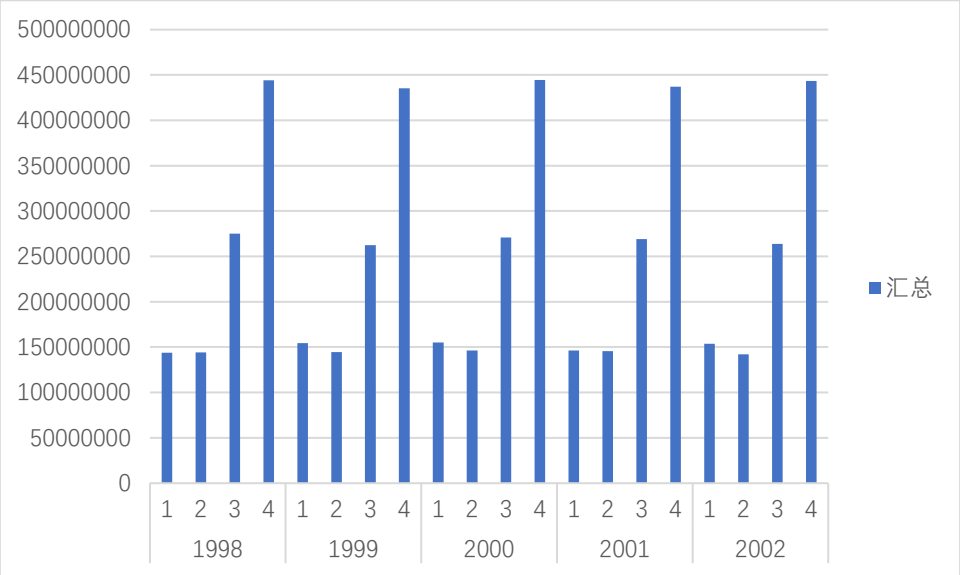

图 3-3- 1-2 1998-2002 销售额情况

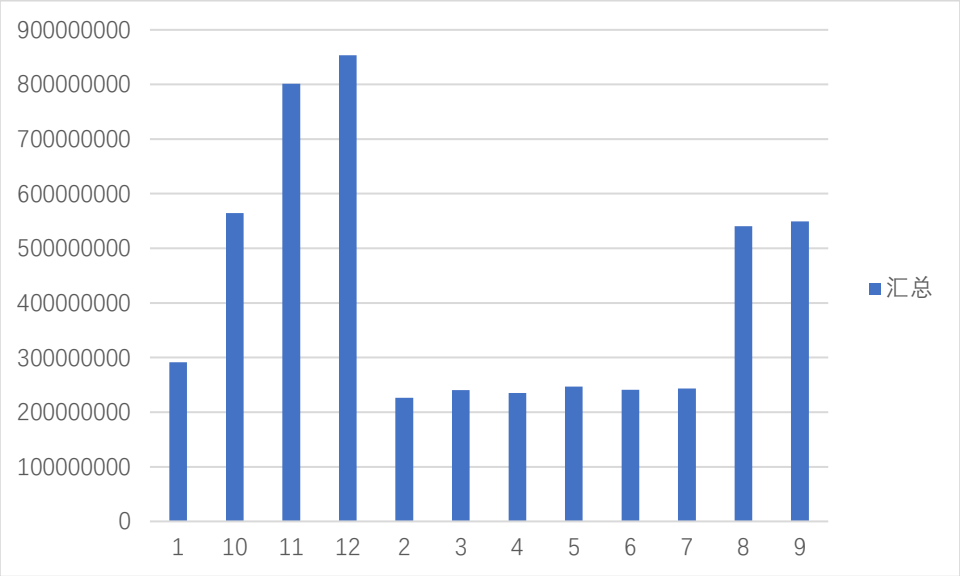

图 3-3- 1-2 一年中每个月的销售情况

（2）实体店销售情况分析。统计 6 家实体店的年销售额，年利润，发现销售发展状况优秀的门店。

从图 3-3-2 可以看出，6 家实体店销售额越大，亏损越多。

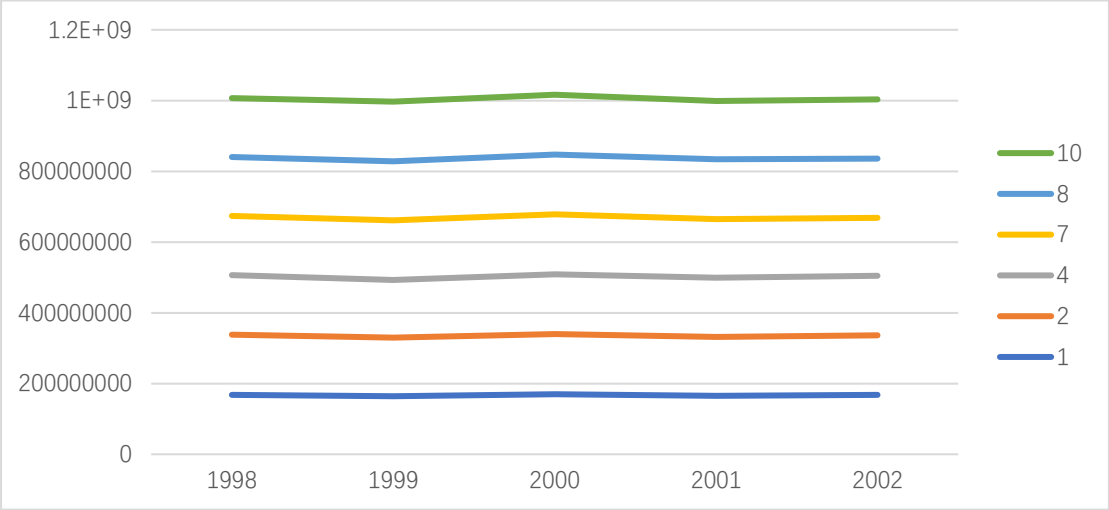

图 3-3- 2-1 **各实体店** 1998-2002 **年销售额情况**

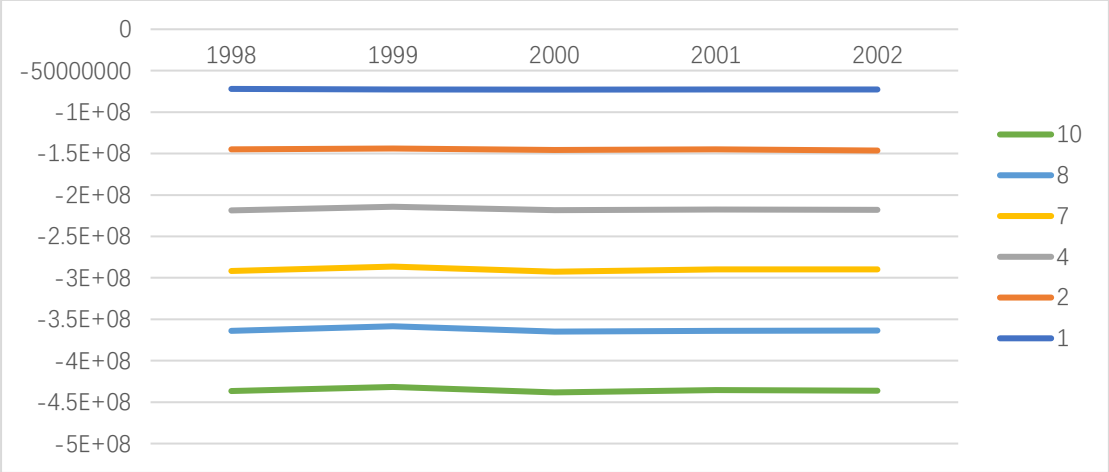

图 3-3- 2-2 **各实体店** 1998-2002 **年净利润情况**

（3）商品畅销影响因素分析。主要分析畅销商品的品牌，类别，目录，颜色，尺码等。

从商品类别看，music 类别总销售额最高，shoes 次之。从商品等级和销售额的等级上看，虽然等级很多，很明显等级可以根据销售额划分为两类，一类畅销等级，一类普通等级。

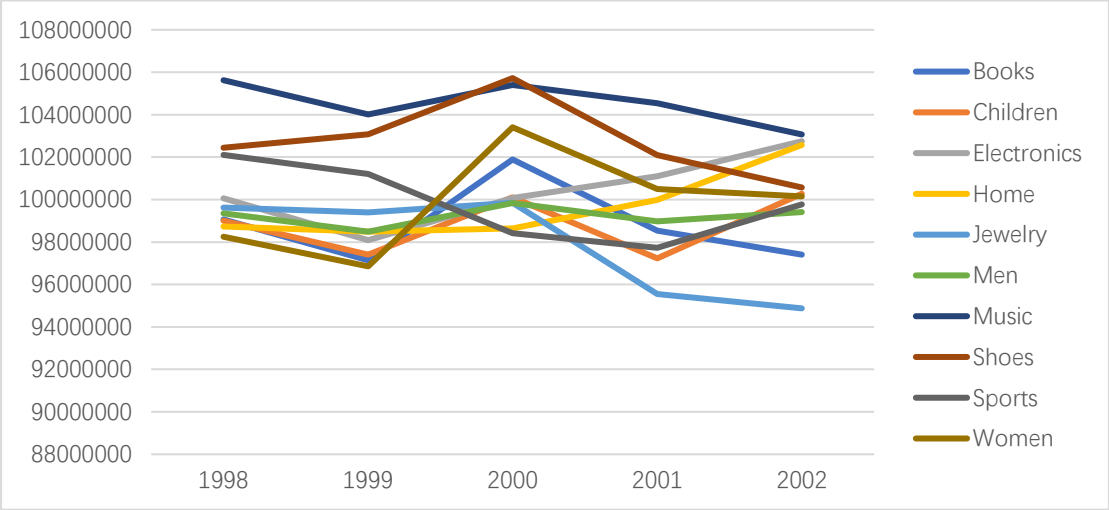

图 3-3- 3-1 不同商品类别 1998-2002 的年销售额情况

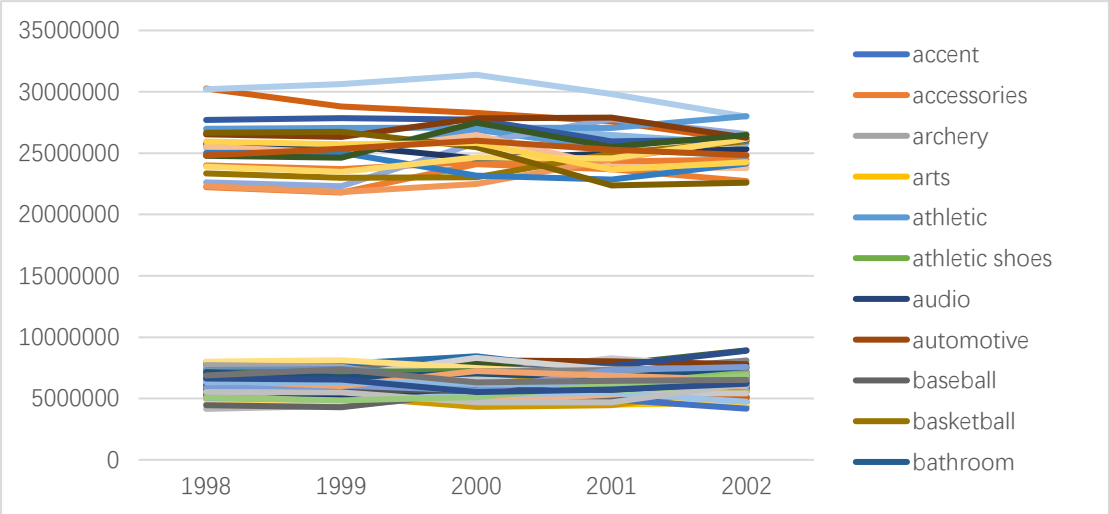

图 3-3- 3-2 不同商品等级 1998-2002 年销售额情况

从商品尺寸上看，中码最畅销。从商品颜色上看，虽然颜色种类繁多，但很明显颜色可以根据销售额划分为两类，一类畅销颜色，一类普通颜色。

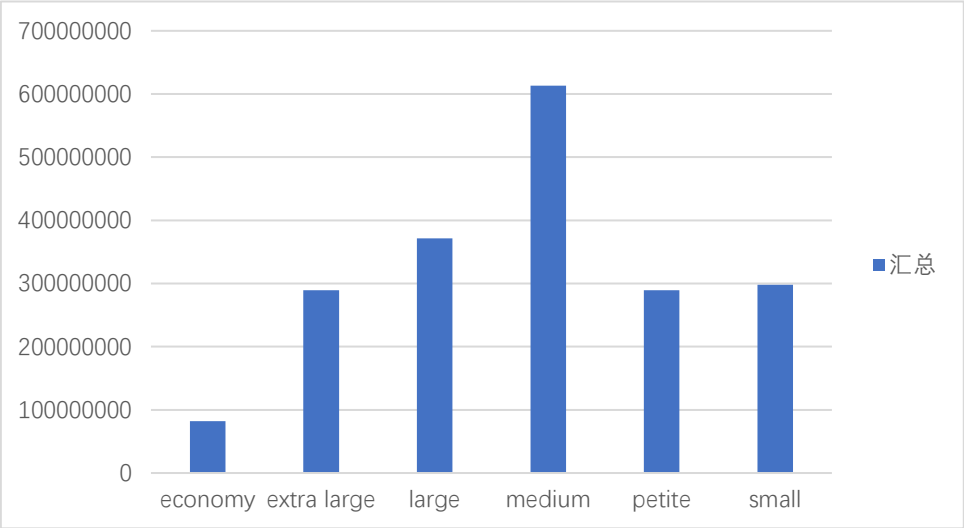

图 3-3- 3-3 不同商品尺寸 1998-2002 总销售额情况

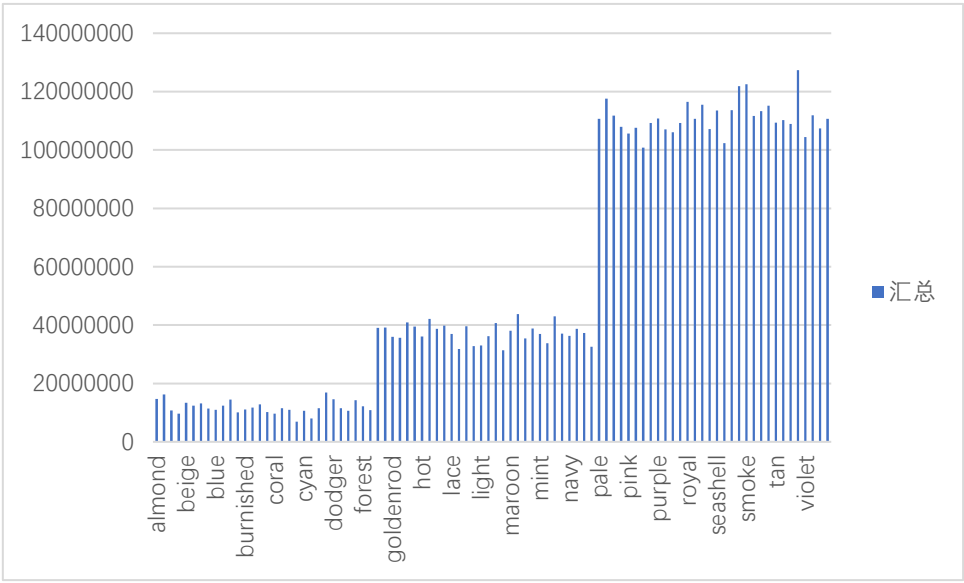

图 3-3- 3-4 不同商品颜色 1998-2002 总销售额情况

4.Power BI 报表展示数据分析成果

如图 3-4-1 所示，该页报表展示实体店 1998-2002 年的销售情况。

左边为销售额按时间加总、按实体店堆叠的堆叠条形图，左上的时间轴为按季度展开，左下的时间轴按月份展开。可以看出实体店每年第四季度是旺季，第四季度的销售额可达到第一、二季度销售额的 3 倍，实体店在一年中，从 1 月到 7 月销售额几乎持平，8 月-12 月逐步上升，12 月销售额达到顶峰。

右边为销售额和净利润的时间序列对比图，右上为销售额按年展开的时间序列图，右下为净利润按年展开的时间序列图。可以看出实体店的情况为销售额约

多，亏损越多。

4-4- 1 **店铺销售额利润报表**

如图 3-4-2 所示，该页报表从不同维度展示商品的销售额与它自身特点的关系。

左边上下两幅图分别为按商品的等级、品牌分类的销售额时间序列图，可以看出商品的等级可根据销售额大小可以明显分为两类，畅销类和普通类，商品的品牌可以根据销售额划分为三类，一类到 1999 年以前销售良好，之后急转直下，2000 年停产，一类普通类，一类畅销类，从 1999 年开始销售额上升。

右边上下两幅图为按商品颜色分类的销售额漏斗图、按尺寸分类的条形图，可以明显看出颜色可以根据销售额划分为三类，而尺寸中，中码最畅销。

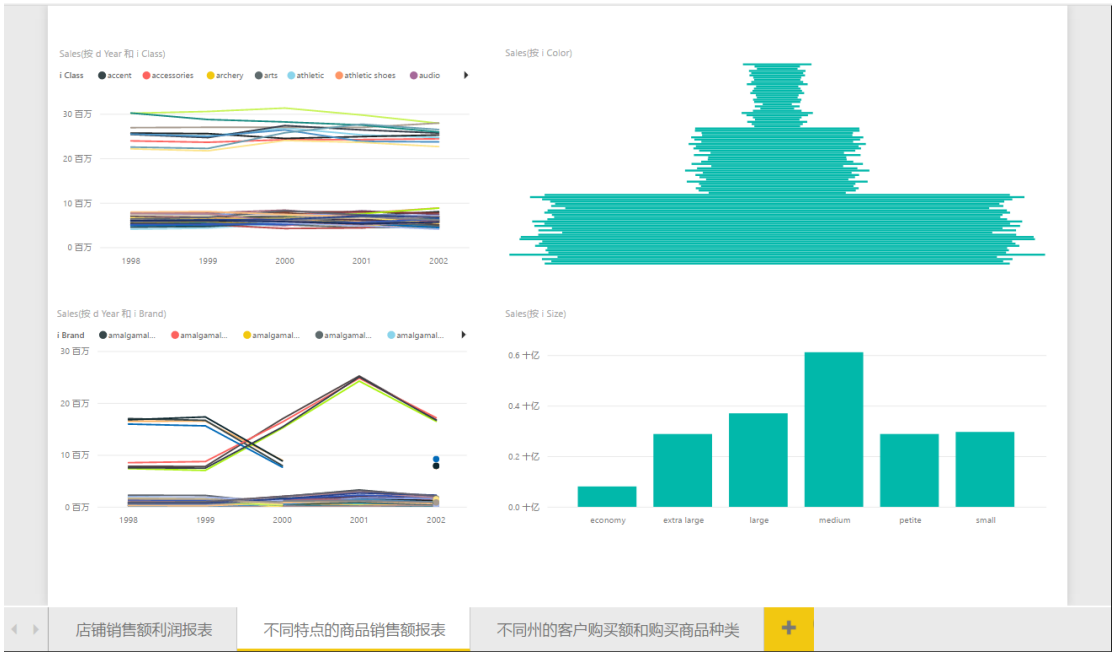

图 3-4- 2 不同特点的商品的销售额报表

如图 3-4-3 所示，该页报表展示顾客购买情况的地理信息。

左右两幅图分别是不同州的顾客贡献的商品购买额和商品丰富度的情况。两图虽变量不同，但深浅几乎一致，说明购买商品种类丰富的客户往往购买数额大，尤其是南部的德克萨斯州，这是因为该公司的 6 家实体店都在德克萨斯州，所以客户多，购买的商品种类也多。

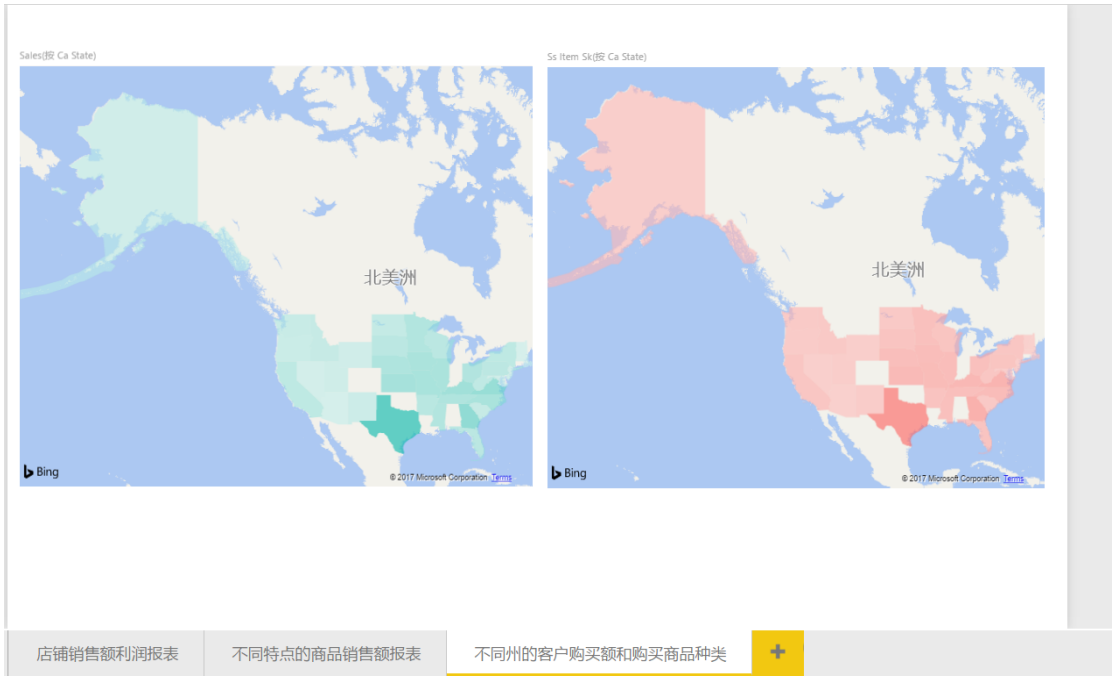

图 3-4- 3 不同州的客户购买额和后买商品种类

5.Power Map 展示客户地域特点[1]

数据来自之前 SQL 查询生成的表格，包括以日为单位的顾客购买商品种类数、购买额、顾客的地理位置。视频展示了从 1998-2002 年 4 年之间不同地域的顾客购买商品种类数、购买额的变化。从视频可以看出，从空间上看，顾客在西南部比较稠密，订单数多且大；从时间上看，每年第一季度是淡季，第四季度是旺季。

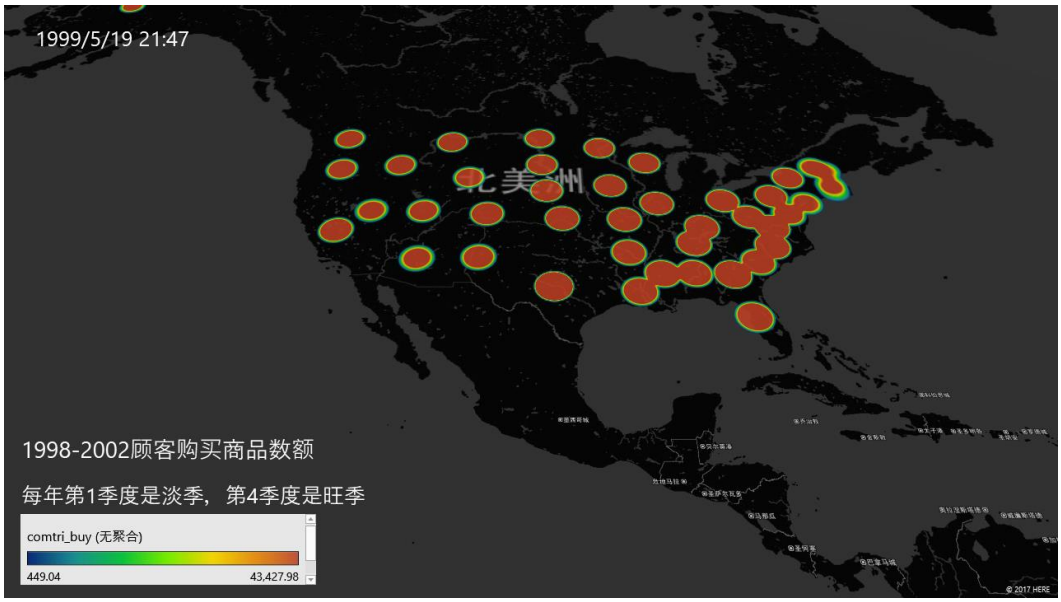

图 3-5-1 1998-2001 顾客购买商品数额视频截图

图 3-5-2 1998-2002 顾客购买商品种类视频截图

[1] 视频附在文件夹中

四、数据挖掘任务

由于笔记本电脑中的 SQL Server 内置 R 服务无法安装成功，因此只用了 RStudio 和 analysis service 对数据进行数据挖掘任务，最后比较两者的优缺点及各自的适用场景。

1.R：聚类分析和纵向数据分析

R 语言中所使用的数据均来自最开始的 SQL 查询命令设计数据分析子任务时保存在 TPCDS 数据库的表格，解决（1）实体店销售额影响因素分析 和（4）商品销售额影响因素分析 这两个子任务。

（1）聚类分析：

由此前的商品销售额报表图 3-4-2 可以看出，不同的商品价格（i_current_price）、销售额（sales=quantity*sale_price）相差可能很大，因此根据商品价格和销售额进行 knn 聚类分析。

```
    #R 版本改成 SQL 的 R 版本
connStr <- paste("Driver=SQL Server; Server=", "DESKTOP-
R5O5NHN",";Database=", "TPCDS", ";Trusted_Connection=true;", sep =
"");

    #影响商品销售的因素
item_returns <- RxSqlServerData(table =
"dbo.item_sales_analysis",connectionString = connStr, returnDataFrame
= TRUE);
i_data <- rxImport(item_returns);
head(i_data)
summary(i_data)

    #销售额聚类,划分商品等级
kc <- kmeans(na.omit(i_data[,c("i_current_price","sales")]), 4);

    #聚类结果可视化
   library(ggplot2)
plot_data = na.omit(i_data[,c("i_current_price","sales")])
plot_data$col = kc$cluster
head(plot_data)
```

```
ggplot(plot_data,aes(i_current_price,sales,colour=col))+geom_point()
ggplot(plot_data,aes(x=sales,fill =
factor(col)))+geom_density(alpha=.35,colour="grey")

    #畅销划分
a1 = aggregate(sales~col,max,data = plot_data)
a2 = aggregate(sales~col,min,data = plot_data)
(a2$sales[order(a1$sales)][-1]+a1$sales[order(a1$sales)][-4])/2
```

根据聚类结果把商品分为四类，并为各个商品样本打上类别标签，绘制的商品的销售额和价格的关系如下图 4-1-1-1 所示，价格几乎对聚类没有影响，主要是根据销售额聚类只不过多数商品的价格在 10 美元以内。图 4-1-1-2 为单独画四类商品的销售额密度图，存在四个明显的波峰，不同类别之间有明显的销售额区间段。

使用 R 语言求得四类商品的销售额区间段为(0,173943),(173944,286867),(286867,458057), (458057,714640)。

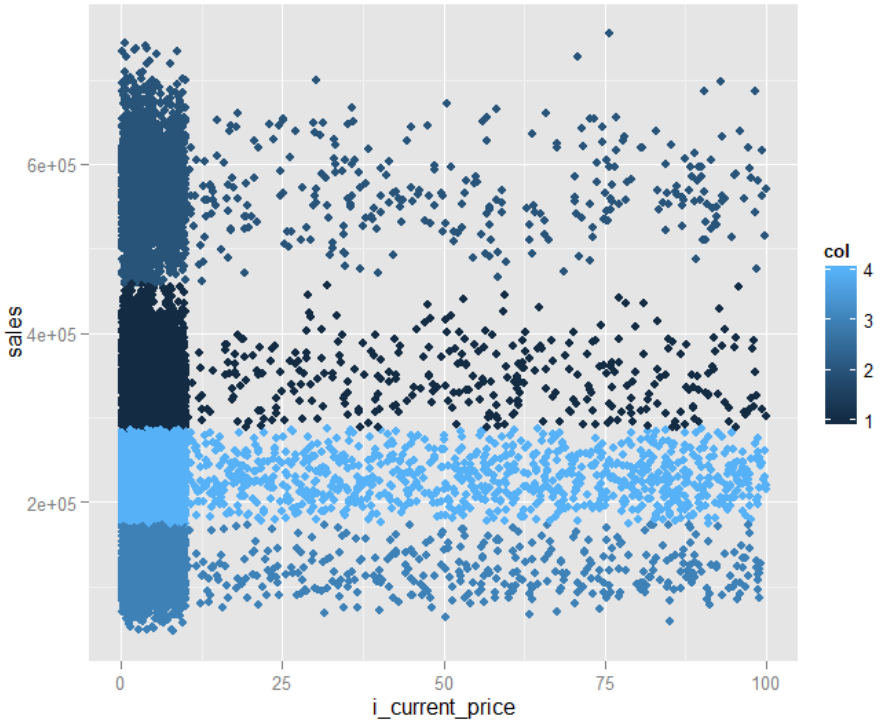

图 4-1-1- 1 商品价格与销售额的散点图

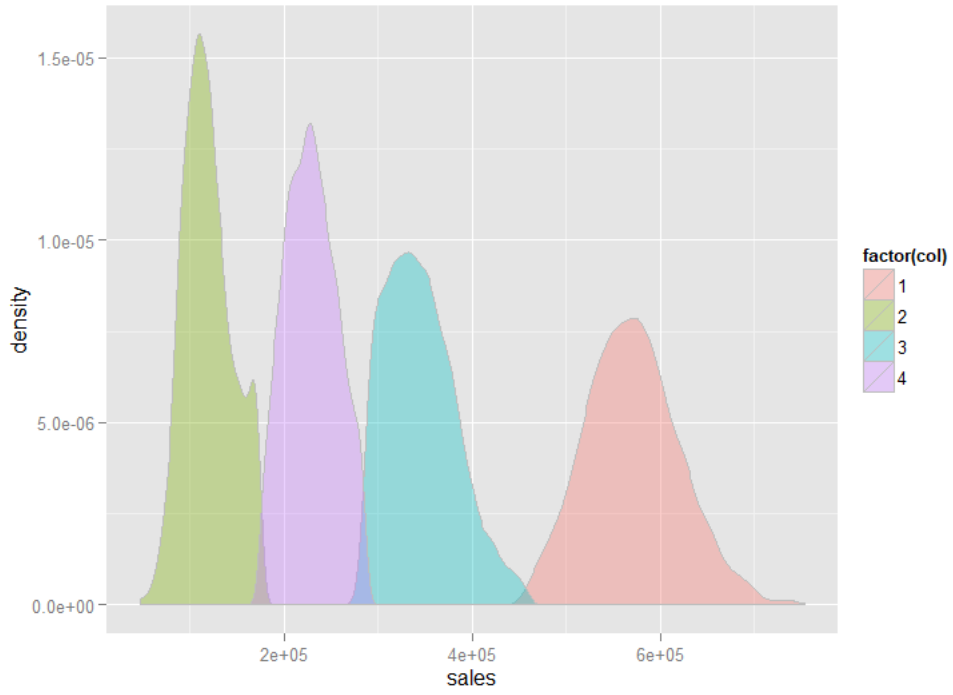

图 4-1-1- 2 商品销售额的直方图

（2）纵向数据分析

为探究实体店月销售额的影响因素，数据为纵向数据，因此不能使用回归分析，而应使用纵向数据分析。本次数据分析选择了四个变量，分别是商品在销售期间的平均价格（avg_price)，商品种类数(diversity)，光临顾客数（customer_flow），由于退货造成的损失（return_loss)。

```
store_returns <- RxSqlServerData(table =
"dbo.store_sales_analysis",connectionString = connStr,
returnDataFrame = TRUE);
ss_data <- rxImport(store_returns);
head(ss_data)
pairs(ss_data[,c("avg_price","diversity","customer_flow","return_loss
","sales")])#散点图矩阵
library(nlme)
fm <- lme( sales ~ avg_price + diversity+customer_flow+return_loss,
data = ss_data, random = ~ 1|s_store_sk)
summary(fm)
```

由散点图矩阵图 4-1-2-1 可以看出，商品种类数，购买顾客数,由于退货造成的损失几乎成完全正相关，因此剔除光临顾客数,由于退货造成的损失两个变

量。在模型的固定效应中放入商品平均价格、购买顾客数，代表所有实体店收影响的共同因素；随机效应中放入门店的 id，代表不同的实体店的个体差异。

从 R 语言的输出结果可以看出，固定效应变量均显著，说明商品平均价格、购买顾客数对月销售额有明显影响；随机效应中，各个门店有明显不同的个体效应。

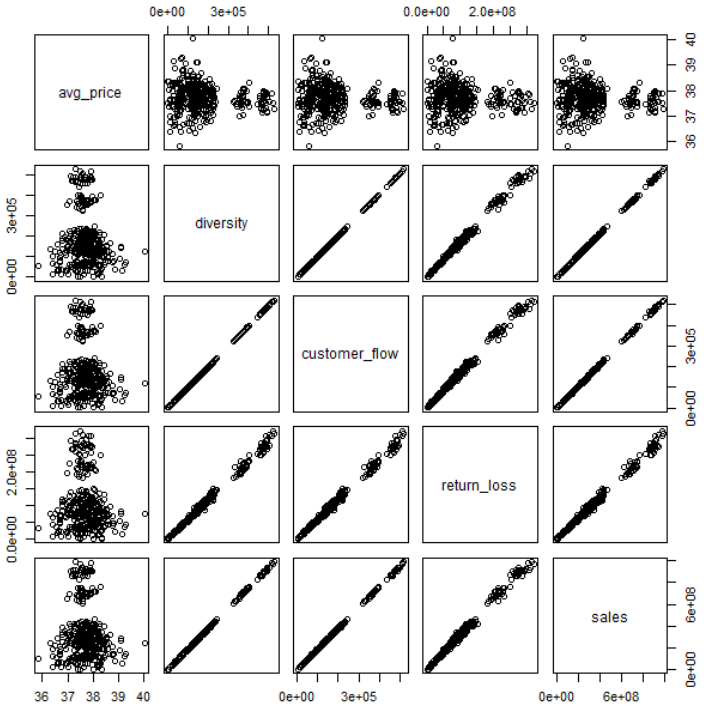

图 4-1-2- 1 影响实体店销售额变量的散点图矩阵

```
Random effects:
 Formula: ~1 | s_store_sk
        (Intercept)
StdDev:    92357.25

 Formula: ~1 | is_q4 %in% s_store_sk
        (Intercept) Residual
StdDev:    521.5015  3502679

Fixed effects: sales ~ avg_price + customer_flow
                    Value Std.Error  DF   t-value p-value
(Intercept)   -231776767  12983106 352  -17.8522       0
avg_price        6164420    343759 352   17.9324       0
customer_flow       1899         1 352 1296.5492       0
 Correlation:
              (Intr) avg_pr
avg_price     -1.000
customer_flow -0.089  0.070

> fm$coefficients
$fixed
  (Intercept)     avg_price customer_flow
-4.825428e+10  1.302602e+09  1.899077e+03

$random
$random$s_store_sk
   (Intercept)
1      3392833
2     87591341
4    -29016806
7      5858506
8    -35965682
10   -31860192
```

图 4-1-2 R 语言输出结果

2. Analysis Service：聚类分析

用 analysis service 做与之前 R 做的一样的商品销售额聚类分析，结果如图 4-2-1 所示，划分区间与 R 的结果相差不大。

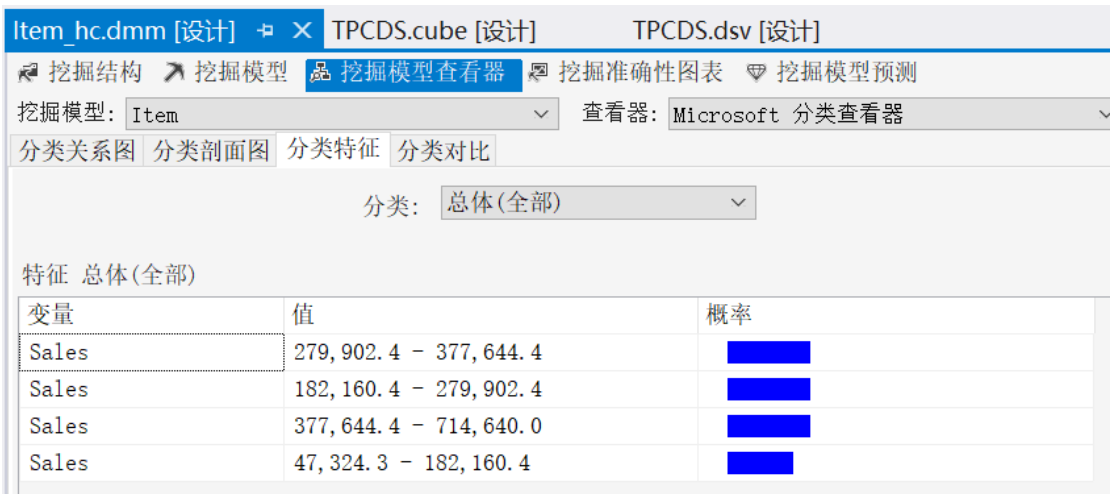

图 4-2- 1 analysis service 聚类结果

3.不同数据挖掘软件的比较

对于 analysis service,特点是可以根据之前的设定的维度进行数据挖掘，对多维数据集的挖掘较为方便；优点是无需编程，简单易操作，数据无需导入，运行速度快； 缺点是数据挖掘的方法非常少，而且方法内部的处理方法只有 MS 公司提供的，缺乏灵活性，数据挖掘的结果展示与传统的统计软件输出的结果展示非常不一样，分为“挖掘结构”，“挖掘模型”，“挖掘结构查看器”，“挖掘准确性图标”，“挖掘模型预测”这五部分，一开始会使人费解。适合的应用场景为：设定的维度对数据挖掘很重要，数据太大，只需要做主流的简单的数据挖掘时，使用 analysis service 较合适。

对于 R 语言，特点是可以把数据库的表格变为 R 数据框，从而可以应用多种 R 函数处理；优点是非常灵活，方法选择非常多样，甚至可以自己设计函数做数据挖掘；缺点是需要编程，对于没有 R 语言编程基础的人来说上手太难，需要把数据导入 R，运行速度较慢。适合的应用场景为：非多维数据集，所需处理的数据均在同一张表格，数据不大（如不超过 1G），需要应用多种方法，使用 R 语言较合适。

附录 1 Create table 命令

```
create table customer_address
(
    ca_address_sk             integer               not null,
    ca_address_id             char(16)              not null,
    ca_street_number          char(10)                      ,
    ca_street_name            varchar(60)                   ,
    ca_street_type            char(15)                      ,
    ca_suite_number           char(10)                      ,
    ca_city                   varchar(60)                   ,
    ca_county                 varchar(30)                   ,
    ca_state                  char(2)                       ,
    ca_zip                    char(10)                      ,
    ca_country                varchar(20)                   ,
    ca_gmt_offset             decimal(5,2)                  ,
    ca_location_type          char(20)                      ,
```

```
    primary key (ca_address_sk)
);
create table date_dim
(
    d_date_sk                 integer               not null,
    d_date_id                 char(16)              not null,
    d_date                    date                          ,
    d_month_seq               integer                       ,
    d_week_seq                integer                       ,
    d_quarter_seq             integer                       ,
    d_year                    integer                       ,
    d_dow                     integer                       ,
    d_moy                     integer                       ,
    d_dom                     integer                       ,
    d_qoy                     integer                       ,
    d_fy_year                 integer                       ,
    d_fy_quarter_seq          integer                       ,
    d_fy_week_seq             integer                       ,
    d_day_name                char(9)                       ,
    d_quarter_name            char(6)                       ,
    d_holiday                 char(1)                       ,
    d_weekend                 char(1)                       ,
    d_following_holiday       char(1)                       ,
    d_first_dom               integer                       ,
    d_last_dom                integer                       ,
    d_same_day_ly             integer                       ,
    d_same_day_lq             integer                       ,
    d_current_day             char(1)                       ,
    d_current_week            char(1)                       ,
    d_current_month           char(1)                       ,
    d_current_quarter         char(1)                       ,
    d_current_year            char(1)                       ,
    primary key (d_date_sk)
);
create table time_dim
(
    t_time_sk                 integer               not null,
    t_time_id                 char(16)              not null,
    t_time                    integer                       ,
    t_hour                    integer                       ,
    t_minute                  integer                       ,
    t_second                  integer                       ,
    t_am_pm                   char(2)                       ,
    t_shift                   char(20)                      ,
```

```
    t_sub_shift               char(20)                    ,
    t_meal_time               char(20)                    ,
    primary key (t_time_sk)
);
create table item
(
    i_item_sk                 integer               not null,
    i_item_id                 char(16)              not null,
    i_rec_start_date          date                        ,
    i_rec_end_date            date                        ,
    i_item_desc               varchar(200)                ,
    i_current_price           decimal(7,2)                ,
    i_wholesale_cost          decimal(7,2)                ,
    i_brand_id                integer                     ,
    i_brand                   char(50)                    ,
    i_class_id                integer                     ,
    i_class                   char(50)                    ,
    i_category_id             integer                     ,
    i_category                char(50)                    ,
    i_manufact_id             integer                     ,
    i_manufact                char(50)                    ,
    i_size                    char(20)                    ,
    i_formulation             char(20)                    ,
    i_color                   char(20)                    ,
    i_units                   char(10)                    ,
    i_container               char(10)                    ,
    i_manager_id              integer                     ,
    i_product_name            char(50)                    ,
    primary key (i_item_sk)
);
create table store
(
    s_store_sk                integer               not null,
    s_store_id                char(16)              not null,
    s_rec_start_date          date                        ,
    s_rec_end_date            date                        ,
    s_closed_date_sk          integer                     ,
    s_store_name              varchar(50)                 ,
    s_number_employees        integer                     ,
    s_floor_space             integer                     ,
    s_hours                   char(20)                    ,
    s_manager                 varchar(40)                 ,
    s_market_id               integer                     ,
    s_geography_class         varchar(100)                ,
```

```
    s_market_desc             varchar(100)                ,
    s_market_manager          varchar(40)                 ,
    s_division_id             integer                     ,
    s_division_name           varchar(50)                 ,
    s_company_id              integer                     ,
    s_company_name            varchar(50)                 ,
    s_street_number           varchar(10)                 ,
    s_street_name             varchar(60)                 ,
    s_street_type             char(15)                    ,
    s_suite_number            char(10)                    ,
    s_city                    varchar(60)                 ,
    s_county                  varchar(30)                 ,
    s_state                   char(2)                     ,
    s_zip                     char(10)                    ,
    s_country                 varchar(20)                 ,
    s_gmt_offset              decimal(5,2)                ,
    s_tax_precentage          decimal(5,2)                ,
    primary key (s_store_sk)
);
create table customer
(
    c_customer_sk             integer               not null,
    c_customer_id             char(16)              not null,
    c_current_cdemo_sk        integer                       ,
    c_current_hdemo_sk        integer                       ,
    c_current_addr_sk         integer                       ,
    c_first_shipto_date_sk    integer                       ,
    c_first_sales_date_sk     integer                       ,
    c_salutation              char(10)                      ,
    c_first_name              char(20)                      ,
    c_last_name               char(30)                      ,
    c_preferred_cust_flag     char(1)                       ,
    c_birth_day               integer                       ,
    c_birth_month             integer                       ,
    c_birth_year              integer                       ,
    c_birth_country           varchar(20)                   ,
    c_login                   char(13)                      ,
    c_email_address           char(50)                      ,
    c_last_review_date        char(10)                      ,
    primary key (c_customer_sk)
);
create table store_returns
(
    sr_returned_date_sk       integer                       ,
```

```
    sr_return_time_sk         integer                       ,
    sr_item_sk                integer               not null,
    sr_customer_sk            integer                       ,
    sr_cdemo_sk               integer                       ,
    sr_hdemo_sk               integer                       ,
    sr_addr_sk                integer                       ,
    sr_store_sk               integer                       ,
    sr_reason_sk              integer                       ,
    sr_ticket_number          integer               not null,
    sr_return_quantity        integer                       ,
    sr_return_amt             decimal(7,2)                  ,
    sr_return_tax             decimal(7,2)                  ,
    sr_return_amt_inc_tax     decimal(7,2)                  ,
    sr_fee                    decimal(7,2)                  ,
    sr_return_ship_cost       decimal(7,2)                  ,
    sr_refunded_cash          decimal(7,2)                  ,
    sr_reversed_charge        decimal(7,2)                  ,
    sr_store_credit           decimal(7,2)                  ,
    sr_net_loss               decimal(7,2)                  ,
    primary key (sr_item_sk, sr_ticket_number)
);
create table store_sales
(
    ss_sold_date_sk           integer                       ,
    ss_sold_time_sk           integer                       ,
    ss_item_sk                integer               not null,
    ss_customer_sk            integer                       ,
    ss_cdemo_sk               integer                       ,
    ss_hdemo_sk               integer                       ,
    ss_addr_sk                integer                       ,
    ss_store_sk               integer                       ,
    ss_promo_sk               integer                       ,
    ss_ticket_number          integer               not null,
    ss_quantity               integer                       ,
    ss_wholesale_cost         decimal(7,2)                  ,
    ss_list_price             decimal(7,2)                  ,
    ss_sales_price            decimal(7,2)                  ,
    ss_ext_discount_amt       decimal(7,2)                  ,
    ss_ext_sales_price        decimal(7,2)                  ,
    ss_ext_wholesale_cost     decimal(7,2)                  ,
    ss_ext_list_price         decimal(7,2)                  ,
    ss_ext_tax                decimal(7,2)                  ,
    ss_coupon_amt             decimal(7,2)                  ,
    ss_net_paid               decimal(7,2)                  ,
```

```
    ss_net_paid_inc_tax       decimal(7,2)                ,
    ss_net_profit             decimal(7,2)                ,
    primary key (ss_item_sk, ss_ticket_number)
);
```

附录 2 创建主-外键参照完整性约束条件的命令

```
alter table customer add constraint c_a foreign key  (c_current_addr_sk) references
customer_address (ca_address_sk);
alter table customer add constraint c_fsd foreign key  (c_first_sales_date_sk)
references date_dim (d_date_sk);
alter table customer add constraint c_fsd2 foreign key  (c_first_shipto_date_sk)
references date_dim (d_date_sk);

alter table store add constraint s_close_date foreign key  (s_closed_date_sk)
references date_dim (d_date_sk);
alter table store_returns add constraint sr_a foreign key  (sr_addr_sk) references
customer_address (ca_address_sk);
alter table store_returns add constraint sr_c foreign key  (sr_customer_sk) references
customer (c_customer_sk);
alter table store_returns add constraint sr_i foreign key  (sr_item_sk) references item
(i_item_sk);

alter table store_returns add constraint sr_ret_d foreign key  (sr_returned_date_sk)
references date_dim (d_date_sk);
alter table store_returns add constraint sr_t foreign key  (sr_return_time_sk)
references time_dim (t_time_sk);
alter table store_returns add constraint sr_s foreign key  (sr_store_sk) references
store (s_store_sk);
alter table store_sales add constraint ss_a foreign key  (ss_addr_sk) references
customer_address (ca_address_sk);
alter table store_sales add constraint ss_c foreign key  (ss_customer_sk) references
customer (c_customer_sk);
alter table store_sales add constraint ss_i foreign key  (ss_item_sk) references item
(i_item_sk);
alter table store_sales add constraint ss_d foreign key  (ss_sold_date_sk) references
date_dim (d_date_sk);
alter table store_sales add constraint ss_t foreign key  (ss_sold_time_sk) references
time_dim (t_time_sk);
alter table store_sales add constraint ss_s foreign key  (ss_store_sk) references store
(s_store_sk);
```