

# 超市销售数据分析项目

日期： 2025年08月11日

## 项目目标

1. 探索超市销量变化趋势
2. 分析不同大类产品的销售情况
3. 构建顾客分分层，重点维护
4. 为超市促销提供数据支持

## 1. 数据加载与初探

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 设置图表样式
sns.set_style("whitegrid")
plt.rcParams['font.sans-serif'] = ['SimHei'] # 正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 正常显示负号

# 加载数据
market_df = pd.read_csv('./data/market_sales_data.csv',encoding='gbk')
# 查看数据基本信息
print(market_df.shape)
print(market_df.head())
print(market_df.info())
```

(42816, 17)

```
顾客编号 大类编码 大类名称 中类编码 中类名称 小类编码 小类名称 销  
售日期 销售月份 \
0 0 12 蔬果 1201 蔬菜 120109 其它蔬菜 20150101 201501
1 1 20 粮油 2014 酱菜类 201401 榨菜 20150101 201501
2 2 15 日配 1505 冷藏乳品 150502 冷藏加味酸乳 20150101 201501
3 3 15 日配 1503 冷藏料理 150305 冷藏面食类 20150101 201501
4 4 15 日配 1505 冷藏乳品 150502 冷藏加味酸乳 20150101 201501
```

	商品编码	规格型号	商品类型	单位	销售数量	销售金额	商品单价	是否促销
0	DW-1201090311		生鲜	个	8.0	4.0	2.0	否
1	DW-2014010019	60g	一般商品	袋	6.0	3.0	0.5	否
2	DW-1505020011	150g	一般商品	袋	1.0	2.4	2.4	否
3	DW-1503050035	500g	一般商品	袋	1.0	6.5	8.3	否
4	DW-1505020020	100g*8	一般商品	袋	1.0	11.9	11.9	否

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42816 entries, 0 to 42815
Data columns (total 17 columns):
 #   Column   Non-Null Count  Dtype  
 --- 
 0   顾客编号    42816 non-null   int64  
 1   大类编码    42816 non-null   int64  
 2   大类名称    42816 non-null   object 
 3   中类编码    42816 non-null   int64  
 4   中类名称    42816 non-null   object 
 5   小类编码    42816 non-null   int64  
 6   小类名称    42816 non-null   object 
 7   销售日期    42816 non-null   int64  
 8   销售月份    42816 non-null   int64  
 9   商品编码    42816 non-null   object 
 10  规格型号   42816 non-null   object 
 11  商品类型    42816 non-null   object 
 12  单位        42816 non-null   object 
 13  销售数量    42814 non-null   float64 
 14  销售金额    42816 non-null   float64 
 15  商品单价    42816 non-null   float64 
 16  是否促销    42816 non-null   object 
dtypes: float64(3), int64(6), object(8)
memory usage: 5.6+ MB
None
```

In [2]: # 数值列的统计性描述

```
market_df[["销售数量", "销售金额", "商品单价"]].describe()
```

Out[2]:

	销售数量	销售金额	商品单价
<b>count</b>	42814.000000	42816.000000	42816.000000
<b>mean</b>	1.199202	10.608517	12.541321
<b>std</b>	2.519092	41.826800	16.658380
<b>min</b>	-16.000000	-145.000000	0.000000
<b>25%</b>	0.530000	2.900000	3.960000
<b>50%</b>	1.000000	5.700000	6.900000
<b>75%</b>	1.000000	10.960000	14.992500
<b>max</b>	216.000000	5340.000000	890.000000

初步观察:

- 数据集包含42816条销售数据
- 销售数量存在缺失值，需要预处理
- 重要的数值类列中，销售数量，销售金额，商品单价都存在负的错误值
- 销售日期是数值型，需要转化为日期型，方便后续分析
- 根据前5条数据发现，部分销售金额≠销售数量\*销售单价，需要找出来，修改销售金额，或删除对应数据行。

## 2. 数据清洗与预处理

In [3]:

```
# 根据分析需求选择合适的列
market_df.drop(columns=["大类编码","中类编码","规格型号","单位"],inplace=True)

# 删除空值
market_df.dropna(subset=["销售数量"],inplace=True)

# 去除销售数量等存在负值的情况
market_df = market_df[(market_df["销售数量"] > 0) & (market_df["销售金额"] > 0) & (market_df["商品单价"] > 0)]

# 在大部分促销商品中，销售金额 != 销售数量*商品单价是正常的，可以不改动，但非促销商品需要改动
mask = market_df["是否促销"] == "否"
market_df.loc[mask,"销售金额"] = round(market_df.loc[mask,"销售数量"] * market_df.loc[mask,"商品单价"])

# 将销售日期和销售月份转化为date类型
# 去除异常值20150229，2015年2月只有28天
market_df = market_df[market_df["销售日期"] != 20150229]
market_df["销售日期"] = pd.to_datetime(market_df["销售日期"].astype(str),format='%Y%m%d')
market_df["销售月份"] = pd.to_datetime(market_df["销售月份"].astype(str),format='%Y%m')
```

In [4]:

```
# 查看数据基本信息
market_df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
Index: 42719 entries, 0 to 42815
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype  
--- 
 0   顾客编号    42719 non-null   int64  
 1   大类名称    42719 non-null   object  
 2   中类名称    42719 non-null   object  
 3   小类编码    42719 non-null   int64  
 4   小类名称    42719 non-null   object  
 5   销售日期    42719 non-null   datetime64[ns]
 6   销售月份    42719 non-null   int32  
 7   商品编码    42719 non-null   object  
 8   商品类型    42719 non-null   object  
 9   销售数量    42719 non-null   float64 
 10  销售金额    42719 non-null   float64 
 11  商品单价    42719 non-null   float64 
 12  是否促销    42719 non-null   object  
dtypes: datetime64[ns](1), float64(3), int32(1), int64(2), object(6)
memory usage: 4.4+ MB

```

### 3. 探索性数据分析

```

In [5]: # 按时间分析，分析日度销售额趋势
# 按天汇总销售额
daily_sales = market_df.groupby("销售日期")["销售金额"].sum()

# 绘制日度销售额趋势图
plt.figure(figsize=(12,6))
plt.plot(daily_sales.index, daily_sales.values, label='日度销售额')
plt.title('日度销售额趋势')
plt.xlabel('日期')
plt.ylabel('销售额')
plt.legend()
plt.grid(True)
plt.show()

```



**日度分析:**

- 销售额最高的日期是2015-02-03，次高的日期是2015-02-18。
- 可以详细看一下这两天的大类销售情况。

```
In [6]: # 提取两天的数据
date_0203 = market_df[market_df["销售日期"]=="2015-02-03"]
date_0218 = market_df[market_df["销售日期"]=="2015-02-18"]

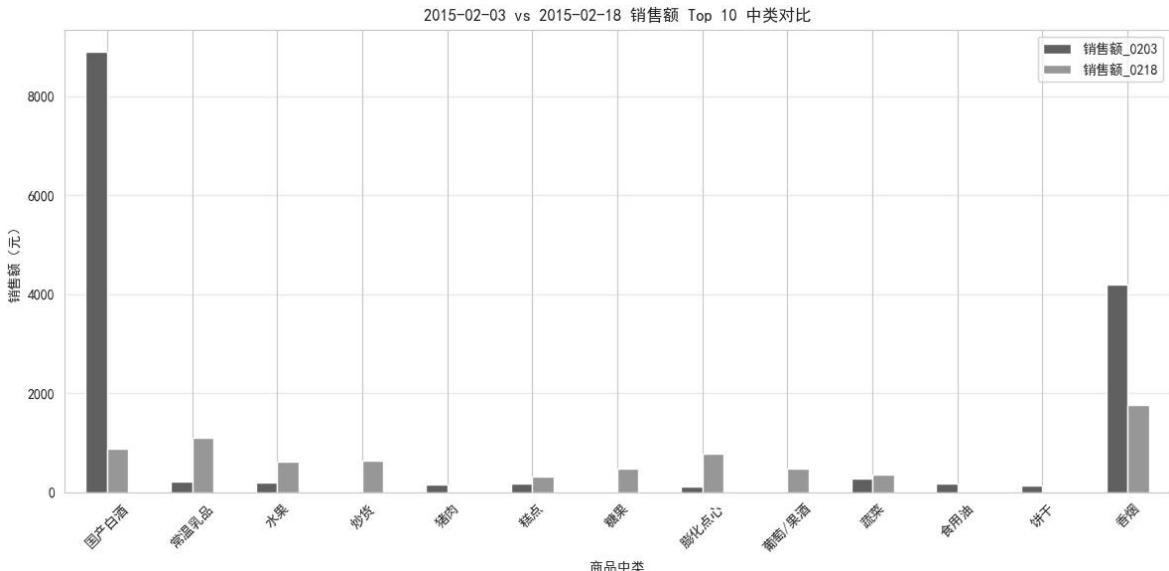
# 分析函数
def analyze_day(df):
    summary = df.groupby('中类名称').agg(
        订单数=('顾客编号', 'count'),          # 交易笔数
        销售额=('销售金额', 'sum'),            # 总销售额
        唯一顾客数=('顾客编号', 'nunique')     # 实际独立顾客数
    ).sort_values(by='销售额', ascending=False)
    return summary

# 执行分析
summary_0203 = analyze_day(date_0203)
summary_0218 = analyze_day(date_0218)

# 将结果可视化
# 获取两天的 Top 10 大类销售额
top10_0203 = summary_0203.head(10)[['销售额']]
top10_0218 = summary_0218.head(10)[['销售额']]

# 合并对比
comparison = pd.merge(top10_0203, top10_0218, left_index=True, right_index=True,
comparison = comparison.fillna(0) # 缺失值补0

# 绘图
comparison.plot(kind='bar', figsize=(12,6))
plt.title('2015-02-03 vs 2015-02-18 销售额 Top 10 中类对比')
plt.ylabel('销售额 (元)')
plt.xlabel('商品中类')
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)
plt.tight_layout()
plt.show()
```



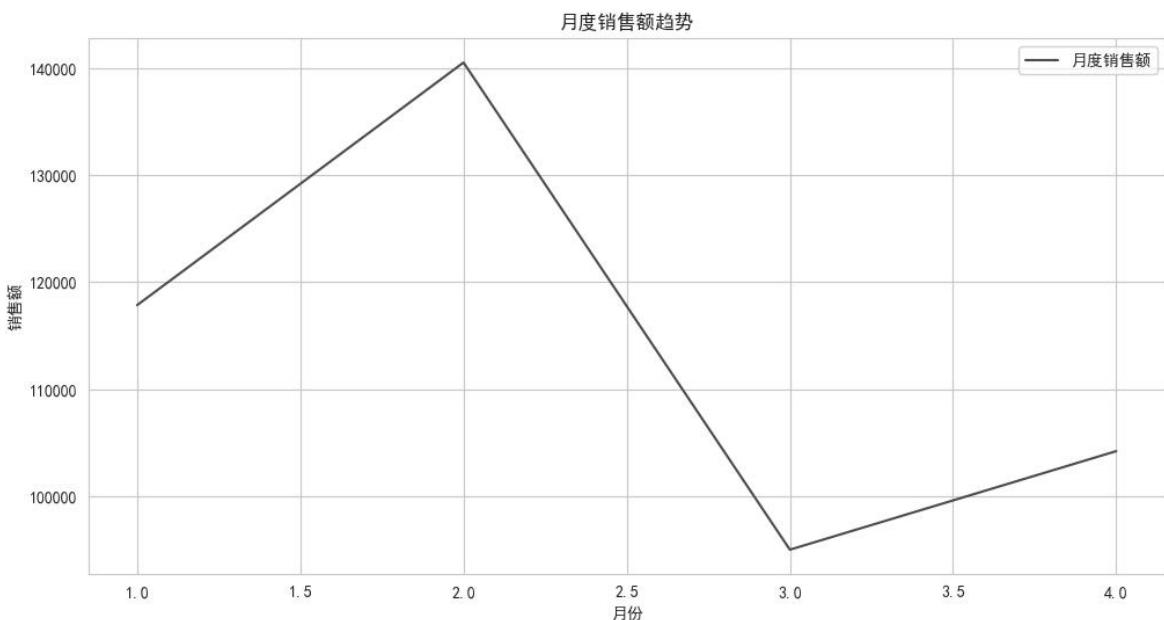
```
In [7]: # 按周汇总销售额
weekly_sales = market_df.resample('W-MON', on='销售日期')['销售金额'].sum()
```

```
# 绘制周度销售额趋势图
plt.figure(figsize=(12,6))
plt.plot(weekly_sales.index, weekly_sales.values, label='周度销售额', color='orange')
plt.title('周度销售额趋势')
plt.xlabel('日期')
plt.ylabel('销售额')
plt.legend()
plt.grid(True)
plt.show()
```



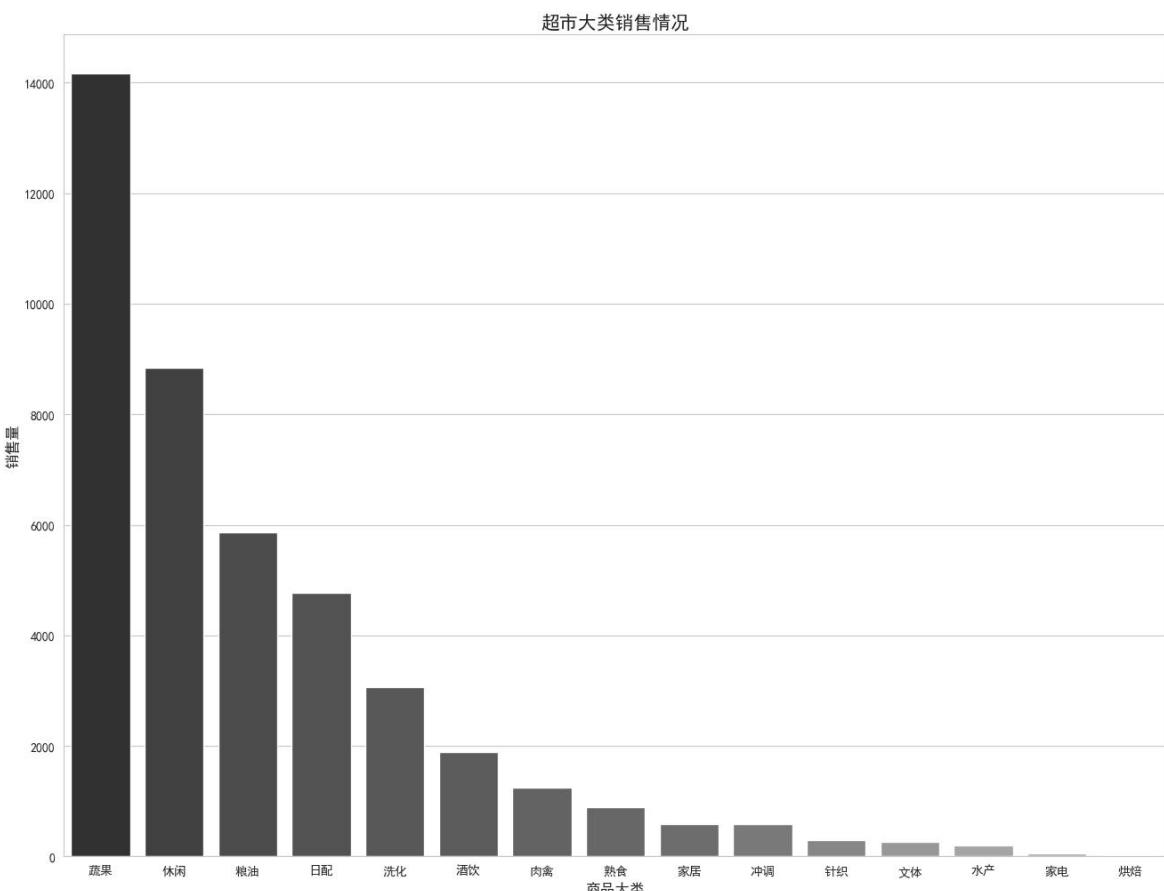
```
In [8]: # 按月汇总销售额
monthly_sales = market_df.groupby("销售月份")['销售金额'].sum()

# 绘制月度销售额趋势图
plt.figure(figsize=(12,6))
plt.plot(monthly_sales.index, monthly_sales.values, label='月度销售额', color='red')
plt.title('月度销售额趋势')
plt.xlabel('月份')
plt.ylabel('销售额')
plt.legend()
plt.grid(True)
plt.show()
```



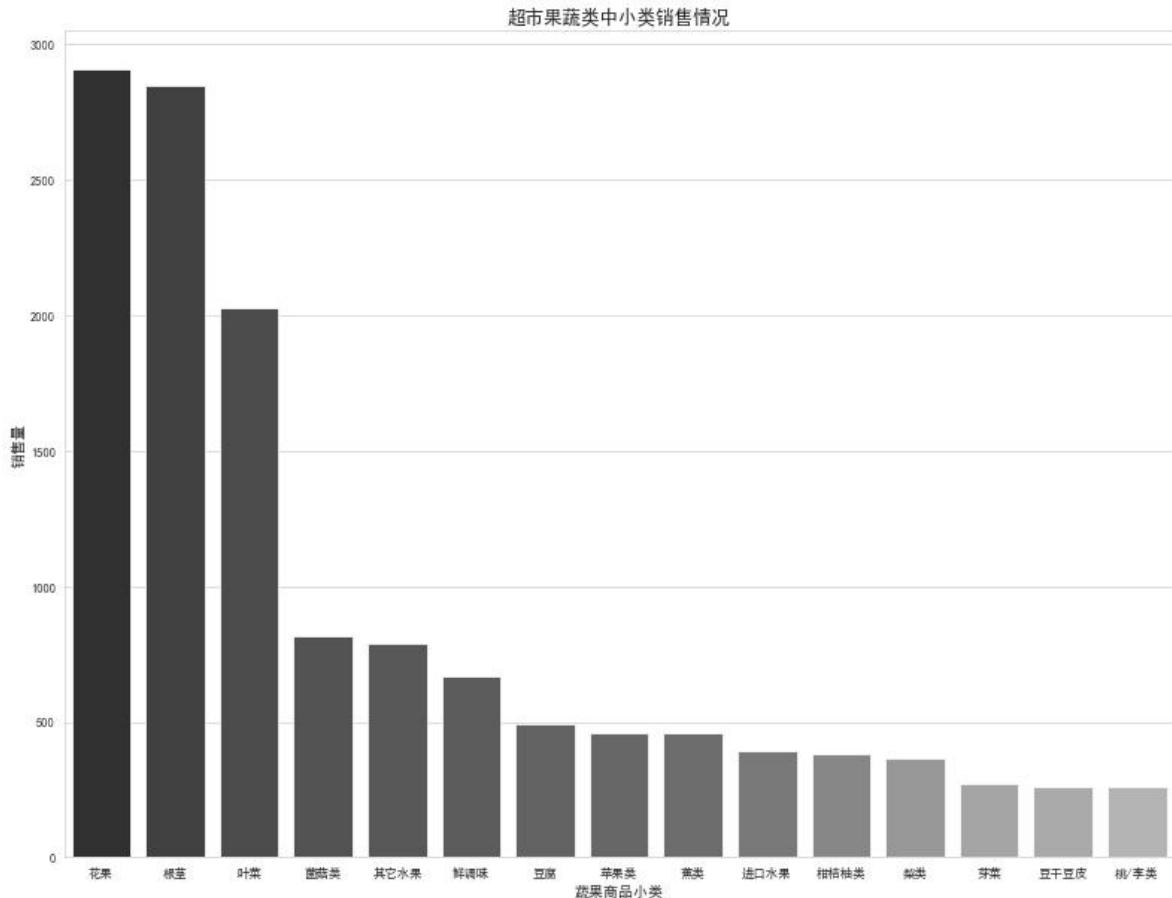
```
In [9]: # 按商品类别分析，看哪些类卖的更好

# 先看一下哪个大类卖的更好
plt.figure(figsize=(16,12))
category_cnt = market_df.groupby("大类名称")["顾客编号"].count().sort_values(ascending=False)
sns.barplot(x=category_cnt.index,y=category_cnt.values,hue=category_cnt.index,palette="dark")
plt.title("超市大类销售情况",fontsize=15)
plt.xlabel("商品大类",fontsize=12)
plt.ylabel("销售量",fontsize=12)
plt.show()
```



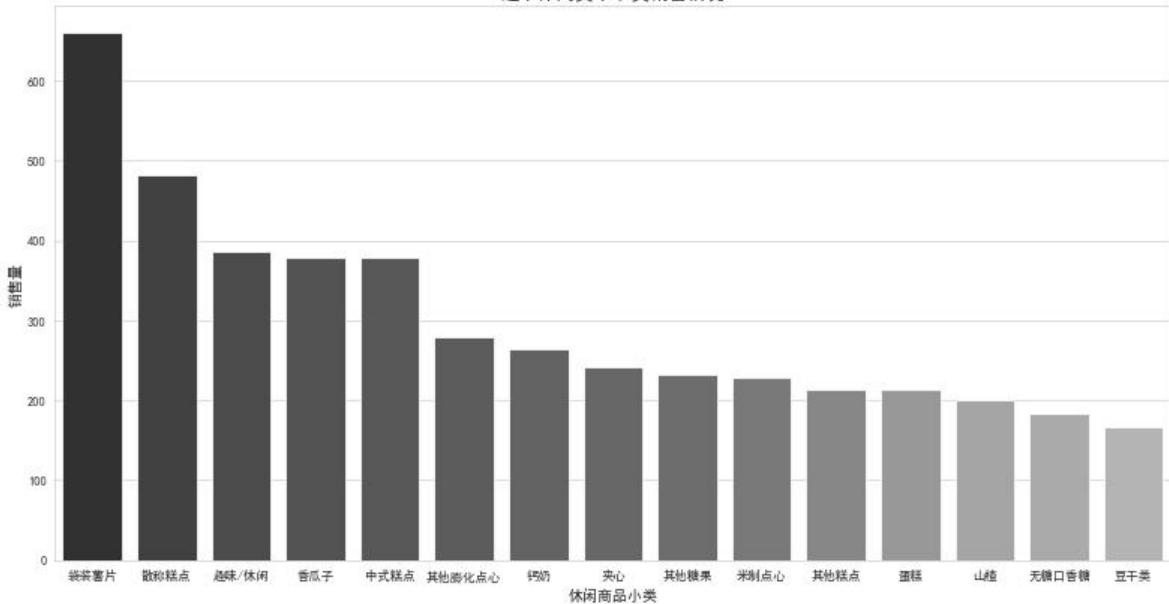
```
In [10]: # 看一下蔬果哪个小类卖的更好
plt.figure(figsize=(16,12),dpi=60)
```

```
medium_cnt = market_df[market_df["大类名称"]=="蔬果"].groupby("小类名称")["顾客编号"]
sns.barplot(x=medium_cnt.index,y=medium_cnt.values,hue=medium_cnt.index,palette="dark")
plt.title("超市果蔬类中小类销售情况",fontsize=15)
plt.xlabel("蔬果商品小类",fontsize=12)
plt.ylabel("销售量",fontsize=12)
plt.show()
```



```
In [11]: # 看一下休闲哪个小类卖的更好
plt.figure(figsize=(16,8),dpi=60)
leisure_medium_cnt = market_df[market_df["大类名称"]=="休闲"].groupby("小类名称")
sns.barplot(x=leisure_medium_cnt.index,y=leisure_medium_cnt.values,hue=leisure_medium_cnt.index,palette="dark")
plt.title("超市休闲类中小类销售情况",fontsize=15)
plt.xlabel("休闲商品小类",fontsize=12)
plt.ylabel("销售量",fontsize=12)
plt.show()
```

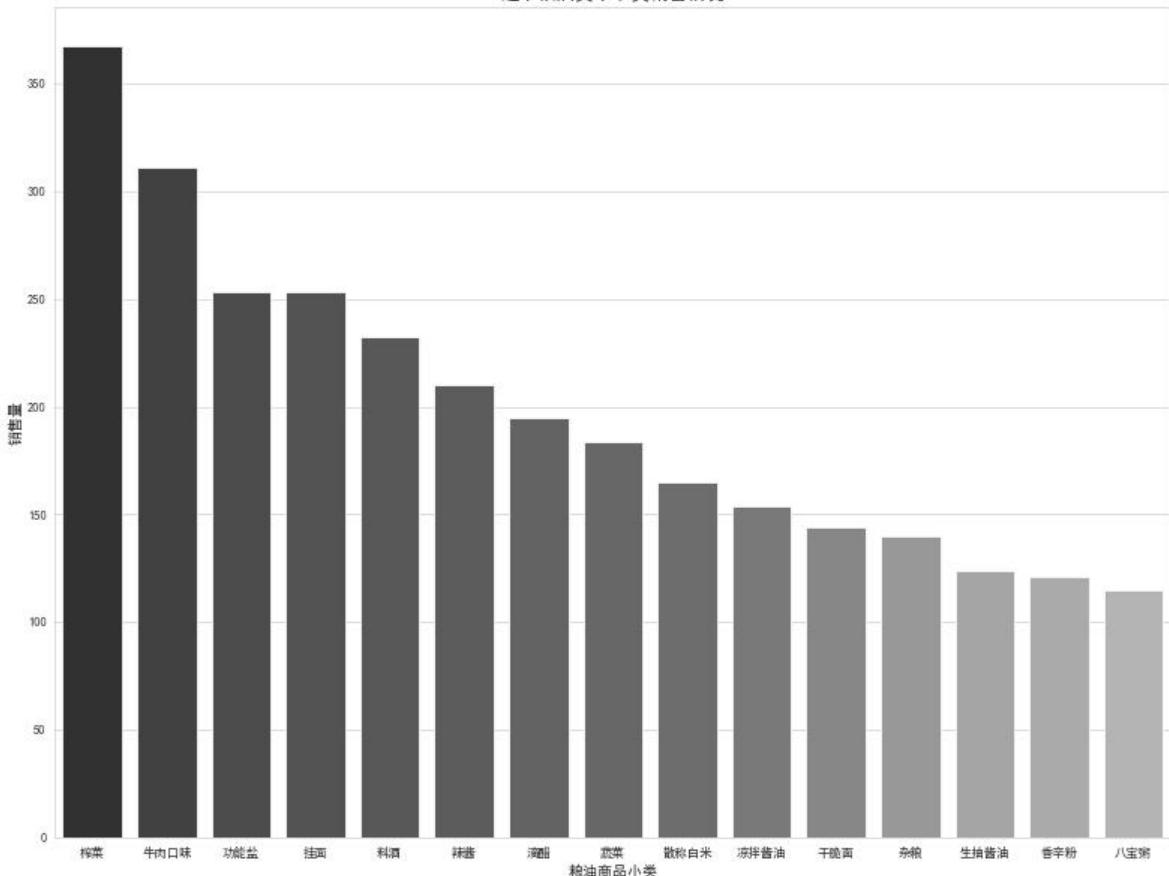
超市休闲类中小类销售情况



```
In [12]: # 看一下休闲哪个小类卖的更好
```

```
plt.figure(figsize=(16,12),dpi=60)
daily_medium_cnt = market_df[market_df["大类名称"]=="粮油"].groupby("小类名称")[
sns.barplot(x=daily_medium_cnt.index,y=daily_medium_cnt.values,hue=daily_medium_
plt.title("超市粮油类中小类销售情况",fontsize=15)
plt.xlabel("粮油商品小类",fontsize=12)
plt.ylabel("销售量",fontsize=12)
plt.show()
```

超市粮油类中小类销售情况



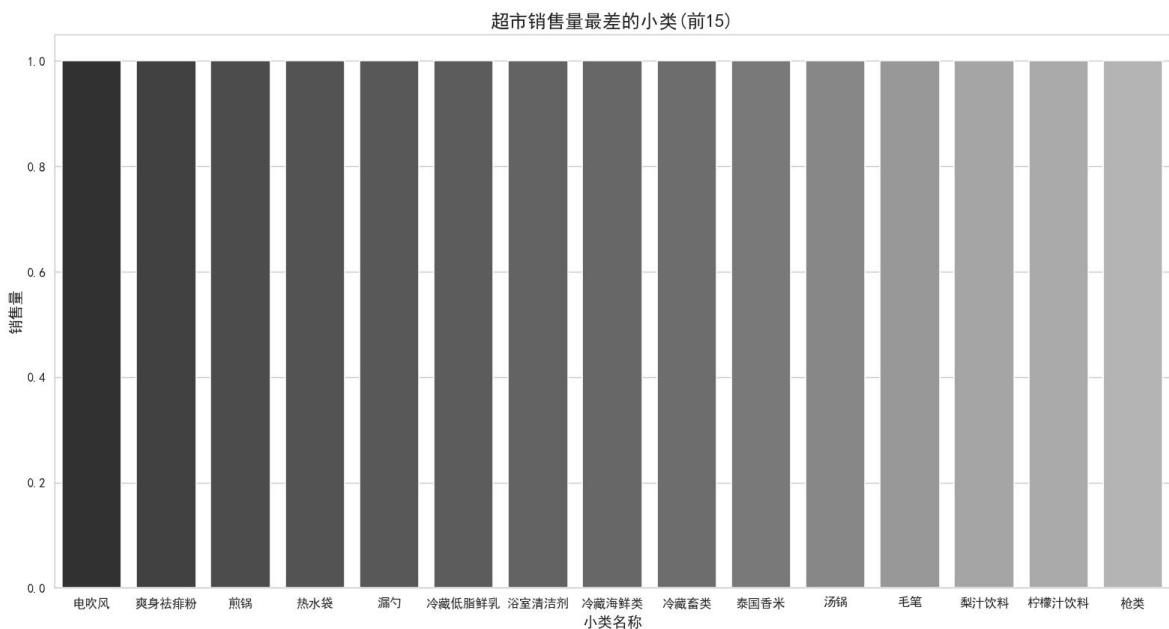
```
In [13]: # 看一下卖的最差的15个小类
```

```
plt.figure(figsize=(16,8),dpi=120)
bad_selling_product = market_df.groupby("小类名称")["顾客编号"].count().sort_val
sns.barplot(x=bad_selling_product.index,y=bad_selling_product.values,hue=bad_sel
```

```

plt.title("超市销售量最差的小类(前15)", fontsize=15)
plt.xlabel("小类名称", fontsize=12)
plt.ylabel("销售量", fontsize=12)
plt.show()

```



```

In [14]: # 看一下促销对品类销量的影响
# 将是否促销的数据分开
promotion_yes = market_df[market_df["是否促销"]=="是"]
promotion_no = market_df[market_df["是否促销"]=="否"]

# 分析函数
def analyze_promotion(df):
    summary = df.groupby('中类名称').agg(
        订单数=('顾客编号', 'count'),           # 交易笔数
        销售额= ('销售金额', 'sum'),            # 总销售额
        唯一顾客数= ('顾客编号', 'nunique')     # 实际独立顾客数
    ).sort_values(by='销售额', ascending=False)
    return summary

# 执行分析
summary_yes = analyze_promotion(promotion_yes)
summary_no = analyze_promotion(promotion_no)

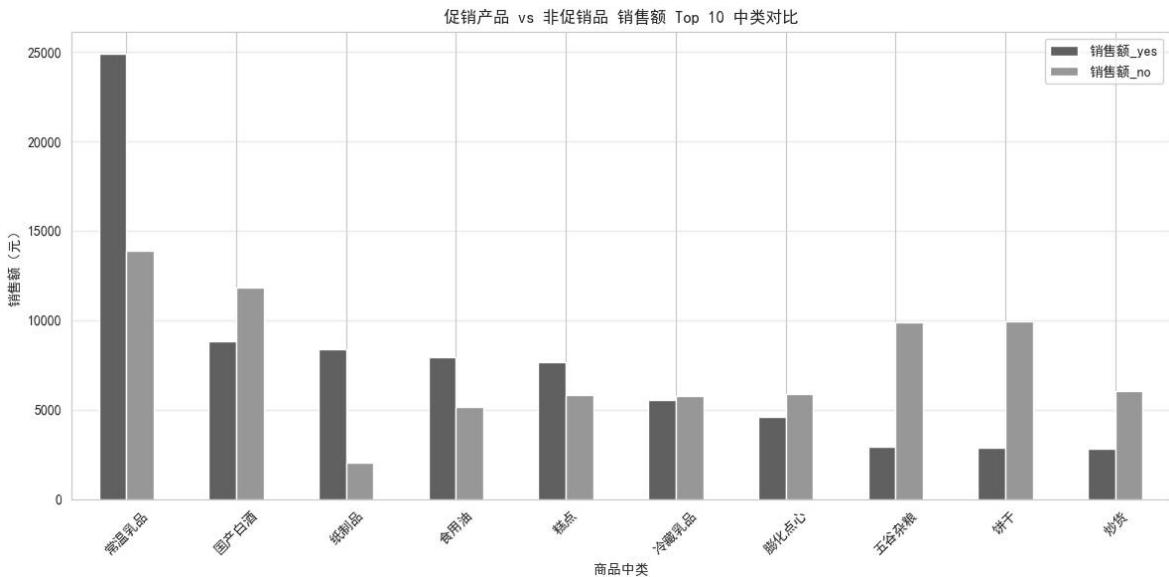
# 将结果可视化
# 获取两类的 Top 10 大类销售额
top10_yes = summary_yes.head(10)[['销售额']]
top10_no = summary_no[['销售额']]

# 合并对比
comparison = pd.merge(top10_yes, top10_no, left_index=True, right_index=True, how='left')
comparison = comparison.fillna(0)  # 缺失值补0

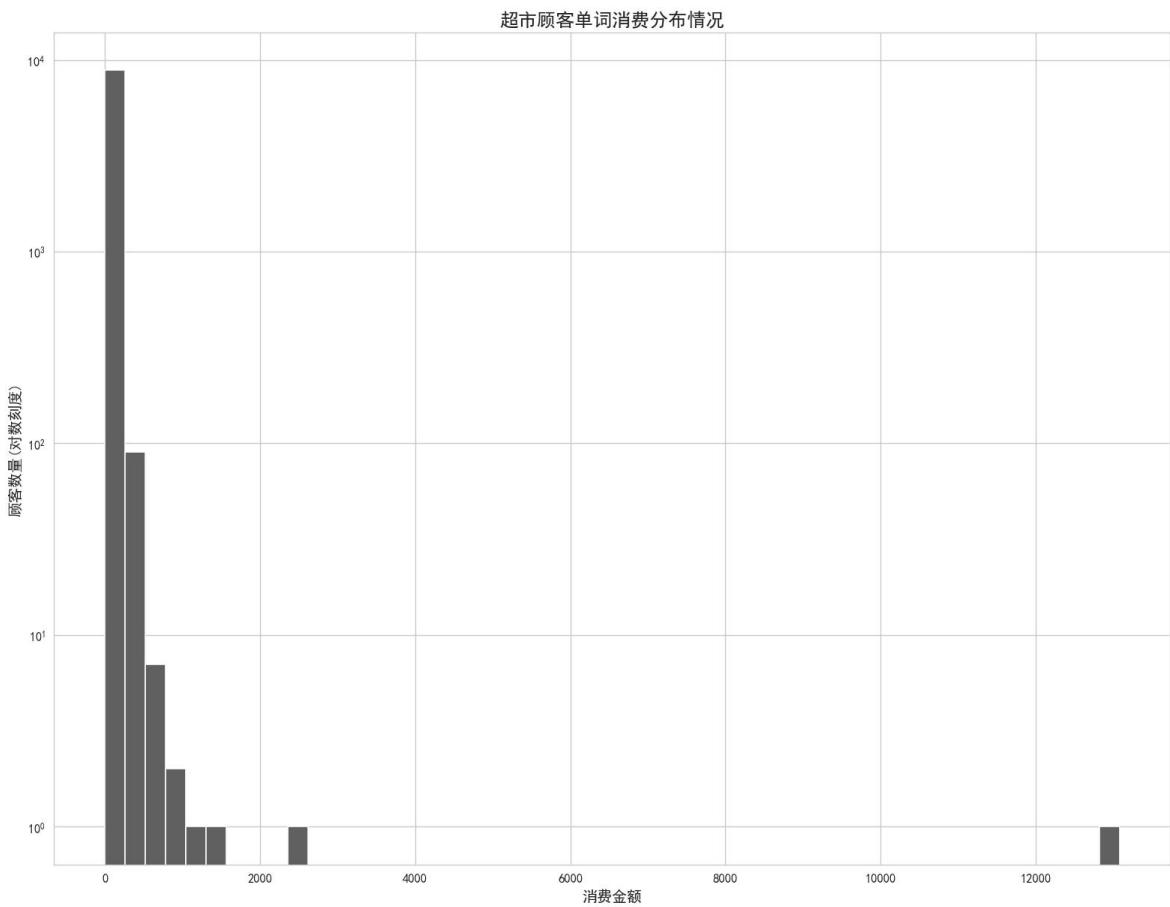
# 绘图
comparison.plot(kind='bar', figsize=(12,6))
plt.title('促销产品 vs 非促销品 销售额 Top 10 中类对比')
plt.ylabel('销售额 (元)')
plt.xlabel('商品中类')
plt.xticks(rotation=45)
plt.grid(axis='y', alpha=0.3)

```

```
plt.tight_layout()  
plt.show()
```



```
In [15]: # 对订单分析，看一下消费区间，以及高消费人群选择的商品  
# 从订单上分析，看顾客的消费水平  
plt.figure(figsize=(16,12),dpi=120)  
customer_amount = market_df.groupby(["顾客编号","销售日期"])["销售金额"].sum().reset_index()  
plt.hist(customer_amount["销售金额"],bins=50)  
plt.yscale('log')  
plt.title("超市顾客单词消费分布情况",fontsize=15)  
plt.xlabel("消费金额",fontsize=12)  
plt.ylabel("顾客数量(对数刻度)",fontsize=12)  
plt.show()  
print("超市顾客消费中位数是: ",customer_amount["销售金额"].median())  
print("超市顾客消费平均值是: ",round(customer_amount["销售金额"].mean(),2))  
print("超市顾客消费最大值是: ",customer_amount["销售金额"].max())
```



超市顾客消费中位数是: 33.39

超市顾客消费平均值是: 51.16

超市顾客消费最大值是: 13080.0

```
In [16]: # 分析单次消费最多的顾客购物情况
max_amount = customer_amount["销售金额"].max()
max_amount_id = customer_amount[customer_amount["销售金额"] == max_amount][["顾客
max_amount_date = customer_amount[customer_amount["销售金额"] == max_amount][["销
print(market_df[(market_df["顾客编号"]==max_amount_id) & (market_df["销售日期"]=

```

月份	顾客编号	大类名称	中类名称	小类编码	小类名称	销售日期	销售月份	
1月	13223	1177	酒饮	国产白酒	231101	名酒	2015-02-03	2 DW-2311010015
1月	13426	1177	酒饮	国产白酒	231101	名酒	2015-02-03	2 DW-2311010018
1月	13465	1177	酒饮	香烟	231602	国产省外香烟	2015-02-03	2 DW-2316020016

	销售数量	销售金额	商品单价	是否促销
13223	6.0	3540.0	690.0	是
13426	6.0	5340.0	890.0	否
13465	60.0	4200.0	70.0	否

```
In [17]: # 根据顾客消费中位数和平均值可以知道大部分消费还是在50一下，计算单次消费在50以下的
customer_amount_cnt = customer_amount["顾客编号"].count()
target_total = customer_amount[customer_amount["销售金额"]<=50][["顾客编号"]].count()
target_proportion = round(target_total / customer_amount_cnt *100 , 2)
print("超市单次消费低于50元的消费占比是: {}%".format(target_proportion))
```

超市单次消费低于50元的消费占比是: 67.65%

```
In [18]: # 可以分析，看一些客户留存，以及新客户量
# 计算总的客户量
```

```

unique_customers = customer_amount["顾客编号"].nunique()
print("超市顾客总数为: {}".format(unique_customers))
# 计算顾客复购率
repeat_customers = (customer_amount.groupby("顾客编号")["销售日期"].count() > 1).sum()
repeat_rate = round(repeat_customers / unique_customers * 100, 2)
print("超市顾客复购率为: {}%".format(repeat_rate))

```

超市顾客总数为: 2612  
超市顾客复购率为: 51.72%

```

In [19]: # 利用RFM分析对用户分层
# 计算每位顾客最后一次消费时间距离数据截至时间有多久
recency = (customer_amount["销售日期"].max() - customer_amount.groupby("顾客编号"))

# 计算每位顾客的购物次数
frequency = customer_amount.groupby("顾客编号")["销售日期"].count()

# 计算每位顾客的总消费金额
monetary = customer_amount.groupby("顾客编号")["销售金额"].sum()

```

```

In [20]: # 构建 RFM 数据框
rfm_df = pd.DataFrame({
    'R': recency,
    'F': frequency,
    'M': monetary
})

# 定义打分函数: 根据分位数打 1-4 分。R是越小越好, FM是越大越好
def r_score(x, x_quantiles):
    if x <= x_quantiles[0.25]:
        return 4
    elif x <= x_quantiles[0.5]:
        return 3
    elif x <= x_quantiles[0.75]:
        return 2
    else:
        return 1

def fm_score(x, x_quantiles):
    if x <= x_quantiles[0.25]:
        return 1
    elif x <= x_quantiles[0.5]:
        return 2
    elif x <= x_quantiles[0.75]:
        return 3
    else:
        return 4

# 计算分位数
r_quartiles = rfm_df['R'].quantile([0.25, 0.5, 0.75])
f_quartiles = rfm_df['F'].quantile([0.25, 0.5, 0.75])
m_quartiles = rfm_df['M'].quantile([0.25, 0.5, 0.75])

# 打分
rfm_df['R_score'] = rfm_df['R'].apply(r_score, args=(r_quartiles,))
rfm_df['F_score'] = rfm_df['F'].apply(fm_score, args=(f_quartiles,))
rfm_df['M_score'] = rfm_df['M'].apply(fm_score, args=(m_quartiles,))

# 对客户进行分级
def rfm_level(df):

```

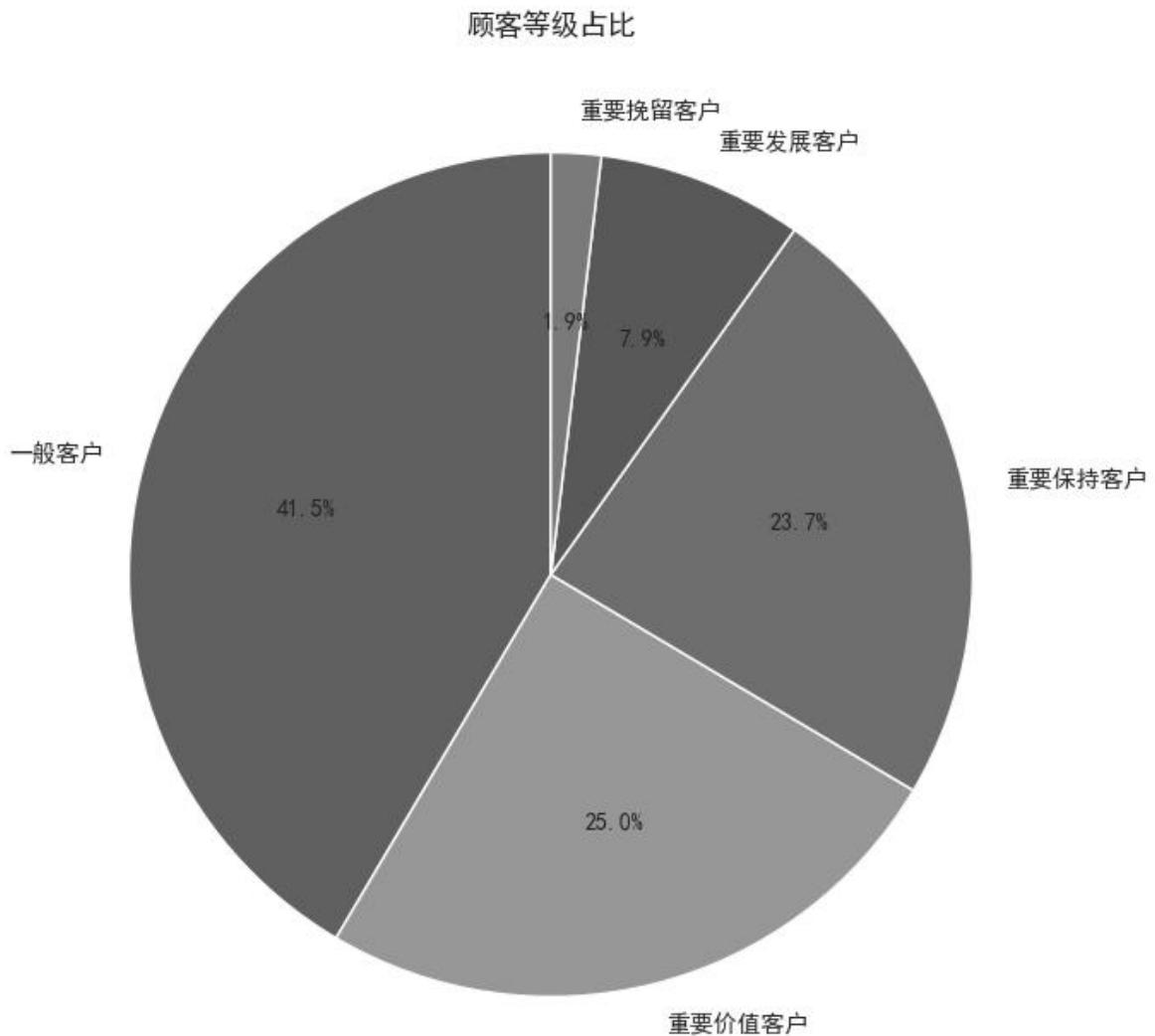
```
if df['R_score'] >= 3 and df['F_score'] >= 3 and df['M_score'] >= 3:  
    return '重要价值客户'  
elif df['R_score'] >= 3 and df['F_score'] < 3:  
    return '重要保持客户'  
elif df['F_score'] >= 3 and df['R_score'] < 3:  
    return '重要发展客户'  
elif df['R_score'] >= 3 and df['M_score'] < 3:  
    return '重要挽留客户'  
else:  
    return '一般客户'  
  
rfm_df['客户等级'] = rfm_df.apply(rfm_level, axis=1)  
rfm_cnt = rfm_df['客户等级'].value_counts()  
print("\n客户等级分布: ")  
print(rfm_cnt)  
# 绘图  
rfm_cnt.plot.pie( autopct='%.1f%%', startangle=90, figsize=(8, 8), title='顾客等  
plt.ylabel('') # 移除y轴标签  
plt.show()
```

客户等级分布:

客户等级

一般客户	1084
重要价值客户	652
重要保持客户	620
重要发展客户	206
重要挽留客户	50

Name: count, dtype: int64



## 4. 分析结论与业务建议

### 关键发现

#### 1. 时间维度

- 日度数据中，2015年春节是在2月19号，日最高销量在2月3号，次高销量在2月18号，最低销量在2月4号。
- 周度数据中，春节前后的两周销售金额最高，春节过后工作日第一周销量急剧下滑。
- 月度数据中，2月份销售总额最高，3月最低。
- 建议：**假期前对于礼品类做好促销活动

#### 2. 类别维度

- 高销量种类:** 蔬果(33.15%)、休闲(20.68%)、粮油(13.71%)、日配 (11.17%) 占据主导
- 促销的影响:** 促销活动对于常温乳类，纸制品有显著提升，对于食用油和糕点也有正面的影响。
- 小类分析:**

- 蔬果大类中，花果、根茎、叶菜类占据主导地位，其他小类销量比较均衡。
- 休闲大类中，袋装薯片销量很高，散装蛋糕、香瓜子、中式糕点、趣味·休闲类次之。
- 粮油大类中，榨菜最高、其次牛肉味、功能盐、桂圆、料酒等销量也不错。
- 在销量最差的15个小类中，电吹风、爽身祛痒粉、煎锅、热水袋等都是耐用产品。

### 3. 订单维度

- 超市的平均消费在51元，消费中位数是33，说明超市以日常小单为主。
- 单笔最高消费13080，2月3号的烟酒消费，大概率是送人或为过年准备的。
- 超市单笔消费低于50元的订单占比是67%，表示日常购物需求较大。

### 4. 顾客维度

- **复购率:** 超市共有2612名顾客，顾客复购率为**51.72%**。
- **顾客行为分析:**
  - 通过RFM分析，对顾客进行分类：一般顾客（41.5%）占比最高，也是需要转化的对象，重要价值客户和重要保持客户需要多维护。
  - 建议：针对不同等级的顾客制定个性化的营销策略，提高整体顾客忠诚度。

## 可行性建议

### 1. 优化节假日营销策略：

- **针对春节等假期提前布局促销活动:** 通过分析，春节前两周销售总额最高，应提前做好春节促销活动。
- **重点推广礼品类商品:** 通过2月3号和18号大类销售情况对比，春节前1~2周重点促销烟酒、礼盒装商品，推出“团购优惠”；除夕夜以购买新鲜蔬果、奶类等即时消费品类为主的“年夜饭主题”。
- **节后及时调整策略:** 春节后首周销量骤降，建议开展“节后返场”“家庭补货”等主题促销，刺激消费回流。

### 2. 优化产品结构陈列：

- **强化高销量品类的主导地位:**
  - 蔬果（33.15%）、休闲（20.68%）是核心品类，应保证充足供应和新鲜度。
  - 在门店黄金位置（如入口、主通道）陈列花果、根茎、叶菜类及袋装薯片、中式糕点等畅销小类。
- **提升低销量种类的曝光和转化:**
  - 热水袋、电吹风等耐用产品与其他产品进行捆绑销售，如热水袋+暖宝宝+保温杯的“冬季三件套”。
  - 设置“家庭日用品专区”，讲煎锅、汤锅等和粮油放置在一起，提升连带购买率。
  - 梨汁饮料、柠檬汁饮料等打折促销。
  - 减少库存积压，避免资金占用。

### 3. 提升客单价与订单价值：

- **将小额订单向中高订单转化:**
  - 当前67%订单低于50元，平均消费51元，说明以日常小单为主。
  - 可以在关键节点推出“满59减10”，“满99减20”等阶梯优惠，引导顾客凑单。

- 设置“爆款+高毛利”组合：
- 比如，袋装薯片+饮料、榨菜+面条等，提升连带销量。

#### 4. 实施客户分层精准营销：

- 可以运营超市社群，结合RFM的分析结果对不同客户群体进行差异化营销。
- **重要价值客户**：重点维护，可以推出专属优惠，生日优惠等，加强粘性。
- **一般客户**：占比大，可以结合社群，不定期发放优惠券，逐步转化。
- **重要挽留客户**：粘性高，消费能力一般，可以提供高性价比推荐。
- **重要保持客户**：发放“回店礼”、“满减券”，提高购买频率。
- **重要发展客户**：可以联系到的情况下，可以推送提醒最新优惠活动。

#### 5. 加强数据驱动的日常运营：

- **建立销售预警机制**：监控日/周销售额波动，有异常下滑时及时预警。
- **定期复盘品类表现**：每季度分析品类占比变化，及时淘汰滞销品，引入潜力新品。

## 项目总结

本项目基于超市销售数据，从时间、品类、订单、顾客四个不同维度展开分析，得出以下三项核心建议：

1. **“注重节假日营销”**：节前走亲访友的习俗会加大对礼品类（如烟酒）的需求，过节期间团圆饭对新鲜食材（如蔬果）的需求也会激增，建议提前布局节日促销。
2. **“实施客户分层营销”**：通过RFM模型识别高价值客户和潜在流失客户，针对不同类型的客户采用不同的营销方法，加强粘性，提升复购率。
3. **“优化商品结构”**：高销量产品应保障供应和品质，低销量的产品可以尝试捆绑销售或组合促销，减少库存。

**未来方向** 建议建立常态化数据监控机制，结合销售数据，实时调整营销策略。

## 5. 技术总结

### 技术栈：

- Python数据分析: Pandas, NumPy
- 数据可视化: Matplotlib, Seaborn
- 数据清洗: 缺失值处理、异常值检测
- 特征工程: 构建复购率、RFM 用户分层、订单统计等关键业务指标

### 关键技能：

1. 复杂数据清洗：处理缺失值、异常值(销售数量/销售金额小于0)
2. 不同时间维度分析：通过销售日期按日，周，月不同维度分析
3. 对比分析：对比2月3号和2月18号种类销量，找出差异
4. 分组聚合分析：按类别统计销量，按顾客编号统计订单
5. 业务洞察转化：将统计结果转化为可行建议

### **数据局限性:**

- 数据集仅包含超市2015年1~4月的数据
- 缺少商品成本, 利润信息