参赛编号：	SJFX000855ZCB
参赛组别：	A
选题方向：	数据统计分析

基于 Python 与 RFM 模型对电商大数据的分析

摘　要

直播带货，作为直播娱乐行业与电商结合的创新模式，已经成为现代商业活动的重要组成部分。根据国家统计局发布的《中国电子商务报告》，截至 2022 年，尽管面临复杂严峻的发展环境，我国依然有效应对各种挑战，实现了经济的平稳运行。电子商务作为推动消费、保障民生、稳定外贸的关键力量，展现出强劲的发展势头，为激发经济活力、促进灵活就业、增强发展信心等方面做出了显著贡献。

对于电商平台而言，通过系统性地分析顾客的关键信息，可以深入洞察顾客的消费模式和电商产品的销售动态，进而有效促进消费，提高销售额。本文旨在介绍一种基于 Python 的 Pandas 库和 RFM 模型对电商大数据进行分析的方法。通过这种方法，可以精准分析顾客行为，制定针对性的营销活动，激发消费者的购买欲望，达到提高销售额、增强市场竞争力的目标。

关键词：电商；RFM 模型；Python

目　录

1 研究背景 1
2 模型介绍 2
2.1 Pandas 库介绍 2
2.2 Matplotlib 与 Seaborn 库介绍 2
2.3 RFM 模型介绍 2
3 数据预处理 4
3.1 数据清洗 4
3.2 清洗数据展示 4
3.3 数据清洗代码截取 5
4. 问题一求解 6
4.1 目标分析 6
4.2 解题思路 6
4.3 结果展示 6
5 问题二的求解 9
5.1 目标分析 9
5.2 解题思路 9
5.3. 结果展示 10
6 问题三的求解 11
6.1 目标分析 11
6.2 特征构造 11
6.3 计算 RFM 打分 12
6.4 客户细分 12
6.5 代金券设计 13
6.6 模型求解 13
7 问题四的求解 15
7.1 目标分析 15
7.2 评估选择 15
7.3 评估指标 16
7.4 结果展示 17
8 问题五的求解 19
8.1 目标分析 19
8.2 优惠券投放设计 19
8.3 结果展示 20
结论与展望 22
结论 22
展望 22
文献引用 23
附录 24

1 研究背景

随着互联网的迅猛发展和智能移动设备的广泛普及，中国乃至全球社会已经全面迈入信息化和数字化的新纪元。大数据正日益渗透到每个人的日常生活中，尤其是在电子商务领域，这一现象与信息技术的突飞猛进和高效智能的数据处理技术的广泛应用密切相关。

根据国家统计局发布的最新数据[1]，截至 2022 年，中国的网民规模已经达到 10.51 亿人，互联网普及率攀升至 74.4%。如图 1-1 所示，全国电子商务交易额呈现出稳步增长的趋势。此外，跨境电商作为电子商务的重要组成部分，其进出口总额也在 2022 年实现了逐年的缓慢增长，如图 1-2 所示。 这些数据不仅彰显了中国互联网产业的巨大发展活力和韧性，同时也揭示了在同比增长放缓和增长乏力的背景下，我国面临的新挑战和建设新经济的重要难题。因此，深入研究如何激发电子商务的活力、提升销售额，对于我国应对新挑战、推动经济高质量发展具有重要的现实意义和战略价值。

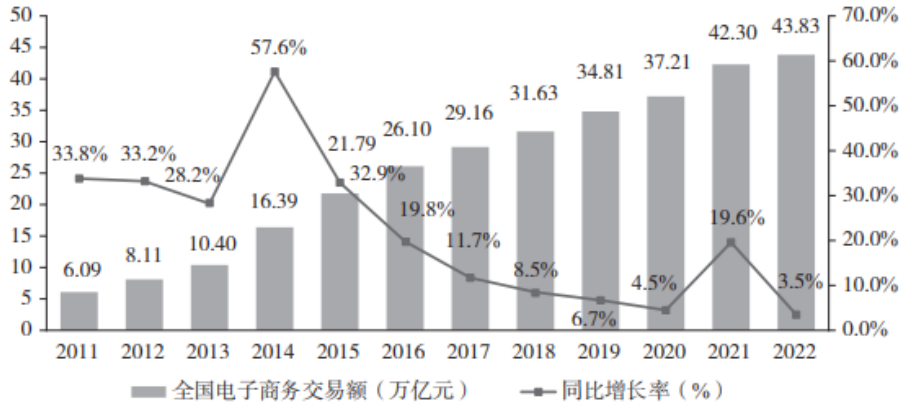

图 1-1 2011-2022 全国电子商务交易额

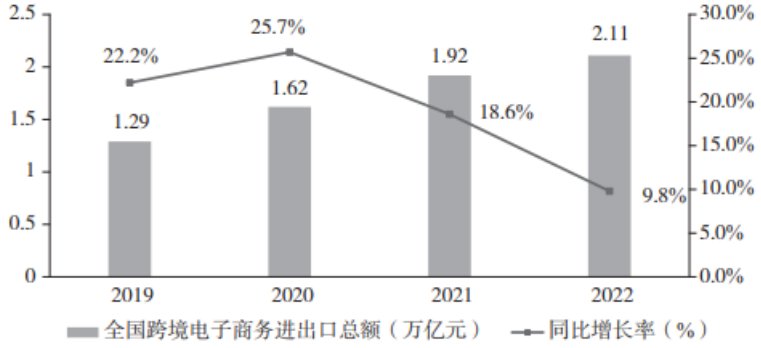

图 1-2 2019-2022 全国跨境电子商务进出口总额

2 模型介绍

2.1 Pandas 库介绍

Pandas[3]是 Python 编程语言中一个功能强大且灵活的数据分析库，它为数据处理提供了高效、直观的工具，极大地简化了数据分析的复杂性。Pandas 的核心数据结构是 DataFrame，它类似于电子表格或 SQL 数据库表，能够存储多种数据类型，并支持广泛的数据操作，如数据筛选、分组、合并和重塑等。

除了 DataFrame，Pandas 还提供了 Series 对象，这是一种带标签的一维数组，它使得一维数据的管理和分析变得简单快捷。Pandas 的另一大优势在于其数据清洗和预处理能力，包括处理缺失值、数据类型转换、数据结构重塑等，这些功能使得原始数据能够被清洗和转换为更适合分析和建模的格式。

Pandas 的强大功能和易用性使其成为了数据科学和数据分析领域不可或缺的工具。在本项目中将利用 Pandas 提供的分组聚合功能以及其他封装函数，以满足特定的数据处理和分析需求，从而有效地完成任务目标。

2.2 Matplotlib 与 Seaborn 库介绍

Matplotlib 是 Python 数据可视化领域中最受欢迎的库之一，它提供了一套全面而强大的绘图工具，能够生成包括折线图、散点图、直方图、饼图在内的多种图表类型。Matplotlib 的设计极具灵活性，不仅支持创建高质量的图形输出，还允许用户对图形的样式和布局进行细致的自定义。

Seaborn 是一个建立在 Matplotlib 基础之上的统计数据可视化库，它进一步提供了更为高级的统计图形生成功能和更为简洁的 API 接口。Seaborn 使得创建复杂的统计图形，如分布图、箱型图、热力图等，变得轻而易举，同时它还拥有一系列美观的默认样式和丰富的颜色选项，进一步提升了图形的视觉吸引力。Matplotlib 与 Seaborn 的结合为数据分析人员提供了一个强大的工具集，使得数据探索、趋势识别以及分析结果的传达变得更加直观和高效。

在本项目中，将利用这两种库的功能来解决数据可视化的需求。

2.3 RFM 模型介绍

RFM 模型[7]是一种深入分析顾客行为的有效工具，它通过三个关键指标——最近一次消费时间（Recency）、消费频率（Frequency）和消费金

额（Monetary）——来衡量和评估顾客的价值，进而为顾客细分提供依据。

R（Recency） 指的是顾客最后一次购买距离当前时间的间隔。这一指标对于判断顾客的活跃度至关重要。通常来说，顾客的最近购买时间越接近现在，表明其对品牌的忠诚度和活跃度越高，流失风险相对较低。活跃顾客对企业营销活动的响应度更高，因此，R（Recency）是衡量顾客活跃程度的关键指标。

F（Frequency） 表示顾客在特定时期内的购买频率。这一指标揭示了顾客对品牌的依赖程度。消费频率较高的顾客往往对品牌的满意度更高，忠诚度也更强。此外，消费频率与顾客在企业中的消费价值呈正相关关系，即频率越高，顾客的价值也越大。因此，F（Frequency）是衡量顾客粘性的重要指标。

M（Monetary） 反映的是顾客在一定时间内的总消费金额，它是评估顾客购买力和对企业贡献度的直接指标。顾客的总消费能力是企业收益的重要来源，尤其是那些消费能力较强的顾客，他们对企业的利润贡献尤为显著。根据帕累托原则，即“二八定律”，企业大部分的收益往往来自一小部分高价值顾客。因此，M（Monetary）不仅体现了顾客的价值，也是企业在资源分配和个性化服务提供时关注的重点。

通过 RFM 模型的应用，企业可以更精准地识别出最有价值的顾客群体，从而有针对性地设计营销策略，提升顾客满意度和忠诚度，最终实现企业收益的最大化。

3 数据预处理

3.1 数据清洗

在数据分析的整个过程中，数据预处理[2]是一个至关重要的步骤，它直接影响到后续分析的质量和可靠性。数据预处理的目的是为了确保数据的质量，使其尽可能地接近真实情况，从而为数据分析提供坚实的基础。在数据预处理阶段，常见的问题包括测量误差、数据收集错误、噪声、离群点、缺失值、不一致值和重复数据等。针对这些问题，数据清洗的主要任务包括填写缺失值、平滑噪声数据、删除离群点以及解决属性的不一致性。

在本例中，观察到原始数据表中存在多种数据缺陷，例如缺少顾客 ID、购买数量出现负数、数据重复、时间信息错误等。为了解决这些问题，我们采用了 Pandas 库中的 dropna 和 duplicates 函数进行数据清洗。通过指定规则，我们删除了那些包含缺失或不正确记录的元组。具体步骤如下：

1. 使用 dropna 函数删除包含缺失值的行，确保数据集中没有缺失关键信息的记录。
2. 通过 drop_duplicates 函数删除数据中的完全重复行，避免分析时的冗余。
3. 利用 pd.to_datetime 函数将日期列转换为 datetime 类型，并对非法日期进行处理，确保时间信息的准确性。
4. 对日期转换失败的行进行删除，保持数据的一致性。
5. 移除 Quantity 列中值为负数或零的记录，因为购买数量不可能为负或零。
6. 保存清洗后的数据记录为 data3.csv

(*注：因为题目并没有给出清洗建议，这里在删除缺失值和重复值的基础上；规则定义清洗了 Price<0，Quantity<=0 的数据；这里默认产品 ID 不空则是正确；并对 Date 做了格式转换)

3.2 清洗数据展示

在结果展示方面，如图 3-3-1 所示，原始数据集包含 541910 条记录。经过清洗过程，删除了存在缺陷的记录后，最终保留了 397925 条记录，如图 3-3-2 所示。清洗数据后提升了数据的可用性和分析的准确性，为进一步的数据分析和决策提供了可靠的数据支持。

541897	541896	22556	PLASTERS	12	2/9/2023	16.5	12680	France
541898	541897	22555	PLASTERS	12	2/9/2023	16.5	12680	France
541899	541898	22728	ALARM CL	4	2/9/2023	37.5	12680	France
541900	541899	22727	ALARM CL	4	2/9/2023	37.5	12680	France
541901	541900	22726	ALARM CL	4	2/9/2023	37.5	12680	France
541902	541901	22730	ALARM CL	4	2/9/2023	37.5	12680	France
541903	541902	22367	CHILDREN	8	2/9/2023	19.5	12680	France
541904	541903	22629	SPACEBO	12	2/9/2023	19.5	12680	France
541905	541904	23256	CHILDREN	4	2/9/2023	41.5	12680	France
541906	541905	22613	PACK OF	12	2/9/2023	8.5	12680	France
541907	541906	22899	CHILDREN	6	2/9/2023	21	12680	France
541908	541907	23254	CHILDREN	4	2/9/2023	41.5	12680	France
541909	541908	23255	CHILDREN	4	2/9/2023	41.5	12680	France
541910	541909	22138	BAKING S	3	2/9/2023	49.5	12680	France
541911	541910	22138	BAKING S	3	2/9/2023	49.5	12680	France

图 3-3-1 原始数据截取

397912	541896	22556	PLASTERS	12	2023/2/9 12:50	16.5	12680	France
397913	541897	22555	PLASTERS	12	2023/2/9 12:50	16.5	12680	France
397914	541898	22728	ALARM CL	4	2023/2/9 12:50	37.5	12680	France
397915	541899	22727	ALARM CL	4	2023/2/9 12:50	37.5	12680	France
397916	541900	22726	ALARM CL	4	2023/2/9 12:50	37.5	12680	France
397917	541901	22730	ALARM CL	4	2023/2/9 12:50	37.5	12680	France
397918	541902	22367	CHILDREN	8	2023/2/9 12:50	19.5	12680	France
397919	541903	22629	SPACEBO	12	2023/2/9 12:50	19.5	12680	France
397920	541904	23256	CHILDREN	4	2023/2/9 12:50	41.5	12680	France
397921	541905	22613	PACK OF	12	2023/2/9 12:50	8.5	12680	France
397922	541906	22899	CHILDREN	6	2023/2/9 12:50	21	12680	France
397923	541907	23254	CHILDREN	4	2023/2/9 12:50	41.5	12680	France
397924	541908	23255	CHILDREN	4	2023/2/9 12:50	41.5	12680	France
397925	541909	22138	BAKING S	3	2023/2/9 12:50	49.5	12680	France
397926	541910	22138	BAKING S	3	2023/2/9 12:51	49.5	12680	France

图 3-3-2 清洗后数据截取

3.3 数据清洗代码截取

```
# 读取数据
df = pd.read_csv('data.csv')

# 数据预处理
df.dropna(inplace=True) #删除包含缺失值的行
df.drop_duplicates(inplace=True) #删除数据中完全的重复行
df['Date']=pd.to_datetime(df['Date'],format='%m/%d/%Y %H:%M',
errors='coerce') df = df.dropna(subset=['Date']) #去除转换日期失败的行
df = df[(df['Quantity'] > 0)]  #删除 Quantity 列值为负数和 0 的行

# 保存预处理后的数据为 data3.csv
df.to_csv('data3.csv', index=False)
```

4. 问题一求解

4.1 目标分析

1. 最畅销商品：确定销量最高的前 10 种商品。
2. 购买力最强顾客：找出购买商品总数最多的前 10 名顾客。
3. 购买力最强国家：识别购买商品总数最多的前 10 个国家。
4. 购买时间段分析：分析顾客的购买行为在一天中的分布情况。
5. 商品价格分布：研究商品价格的分布情况
6. 可视化展示结果：利用 Seaborn 库实现可视化

4.2 解题思路

1. 数据准备：读取经过预处理的数据集，确保数据的完整性和准确性。

2. 数据统计分析：利用 Pandas 库中的 groupby 和 sum 函数，对数据进行分组和聚合操作，计算每个商品、顾客和国家的购买数量。然后，使用 sort_values 函数对结果进行排序，并提取排名靠前的记录。

3. 通过 Matplotlib 和 Seaborn 库来可视化图表。

4.3 结果展示

下面便是本题的解题结果，源码可以查看 question1.py

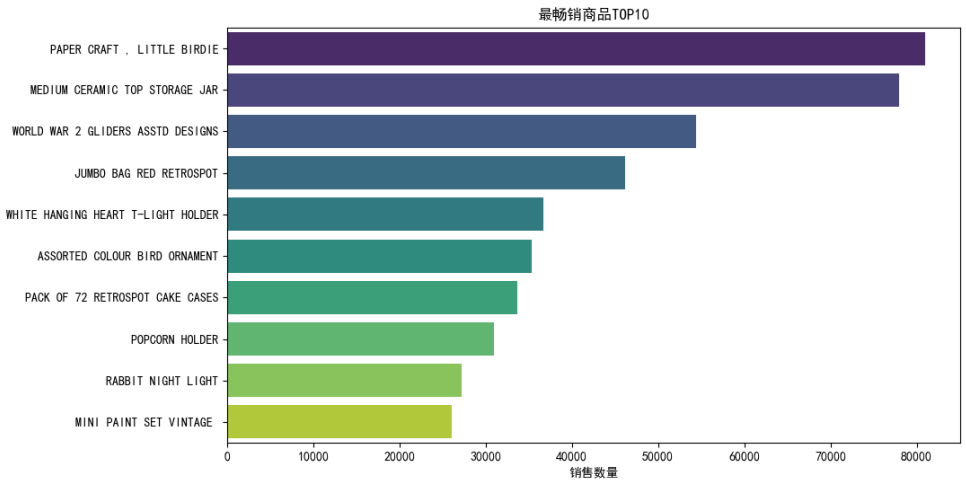

图 4-3-1 最畅销商品 TOP10

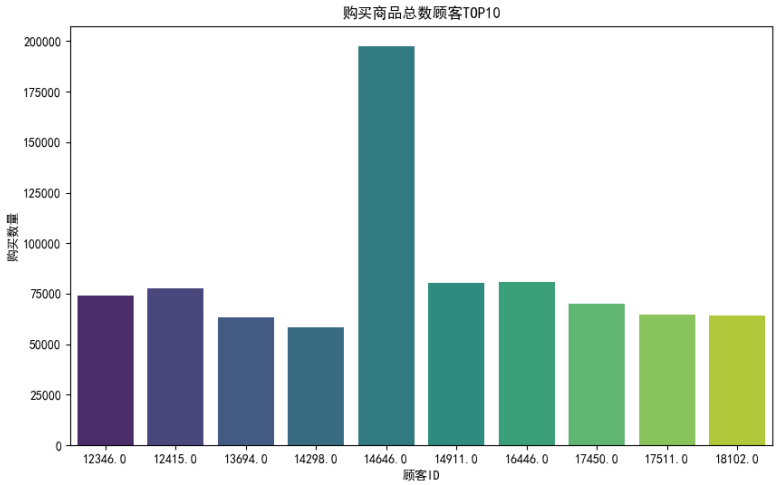

图 4-3-2 顾客购买量排行 TOP10

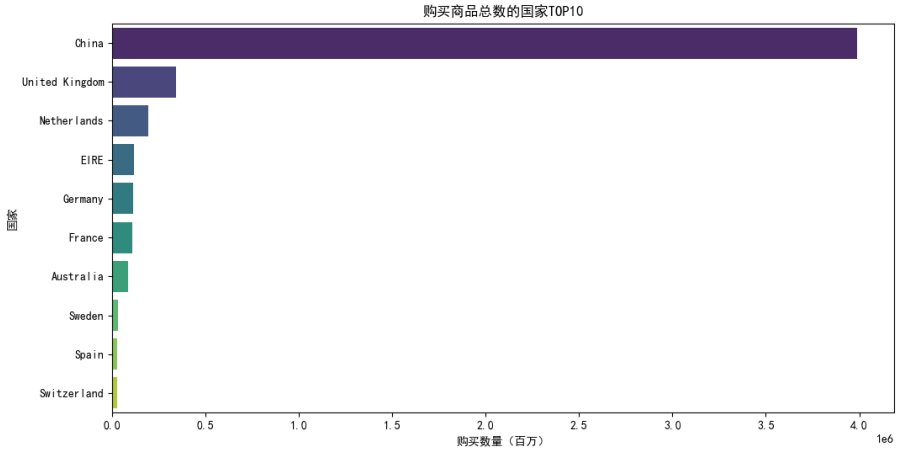

图 4-3-3 购买量国家排行 TOP10

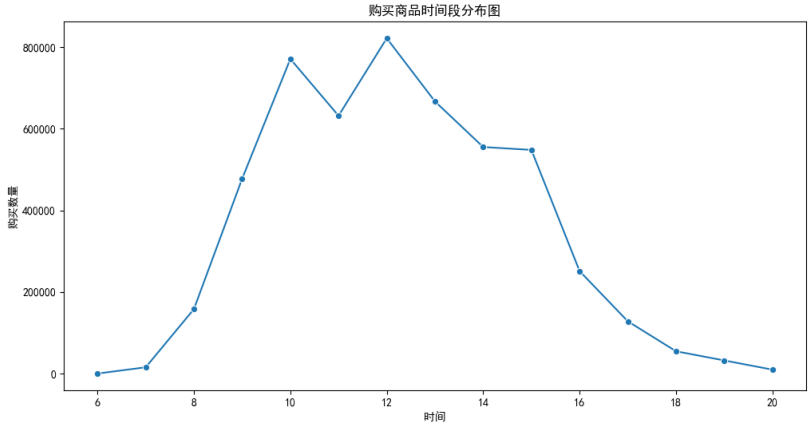

图 4-3-4 购买时间段分布

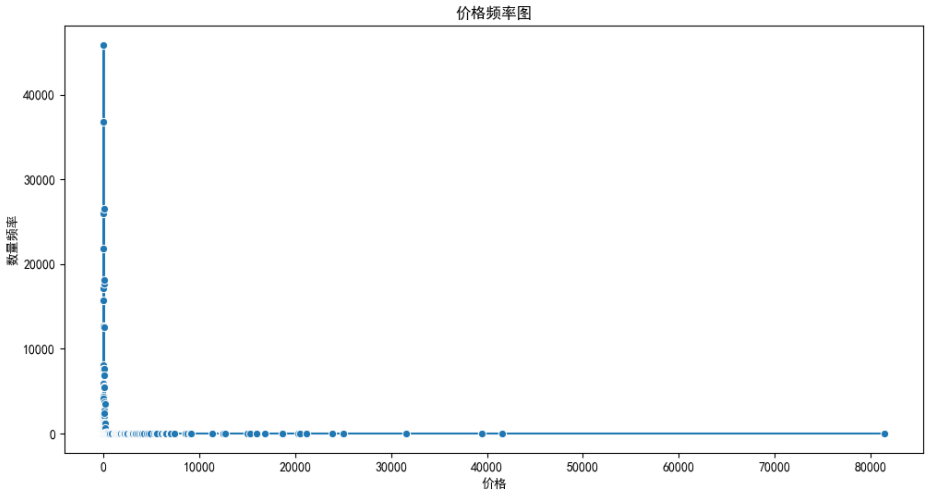

图 4-3-5 价格频率图

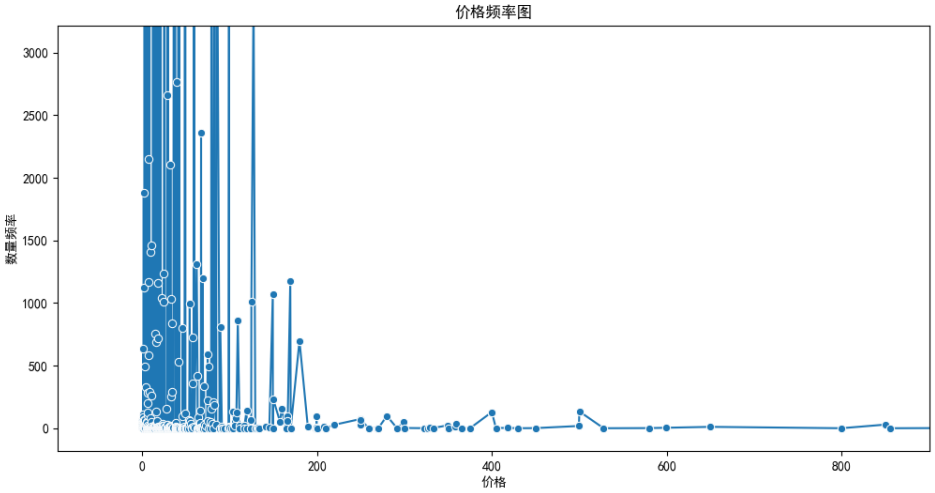

图 4-3-6 价格频率放大图

5 问题二的求解

5.1 目标分析

根据以下几个维度来构建顾客标签：
1. 顾客属性：顾客 ID
2. 消费水平：总消费金额、平均订单金额、购买次数
3. 消费偏好：购买最多的商品类别、购买频率等。

5.2 解题思路

1.首先单独构建出顾客 id 与总消费金额分布、平均订单金额分布、订单次数分布分布、经常购买物品的关系标签

2.然后将上面的 4 个标签进行合并为用户行为标签 customer_profile 来为用户打标签并绘制出可视化图形展示不同的人数

```
customer_profile = pd.DataFrame({
    'TotalAmount': customer_total_amount,
    'AvgOrderAmount': customer_avg_order_amount,
    'OrderCount': customer_order_count
}).reset_index()
customer_profile=
customer_profile.merge(customer_favorite_product[['CustomerID',
'Product']], on='CustomerID')
```

3.标签数学逻辑(由国家统计局发布的《中国电子商务报告》综合出的数据指导)

判定条件	高价值客户	中等价值客户	低价值客户
TotalAmount	≥ 80% 百分位数	20% - 80%	≤ 20% 百分位数
AvgOrderAmount	≥ 50% 百分位数	40% - 50%	≤ 50% 百分位数
OrderCount	≥ 60% 百分位数	40% - 60%	≤ 40% 百分位数

5.3. 结果展示

以下便是本题的解题结果截取，源码可以查看 question2.py。每位客户的价值标签和详细信息可以在 customer_value.csv 中查看。

CustomerId	CustomerValue
12346	Middle Value
12347	High Value
12348	Middle Value
12349	Middle Value
12350	Middle Value
12352	High Value
12353	Middle Value
12354	Middle Value
12355	Middle Value
12356	High Value
……	……

表 5-3-1 结果截取展示

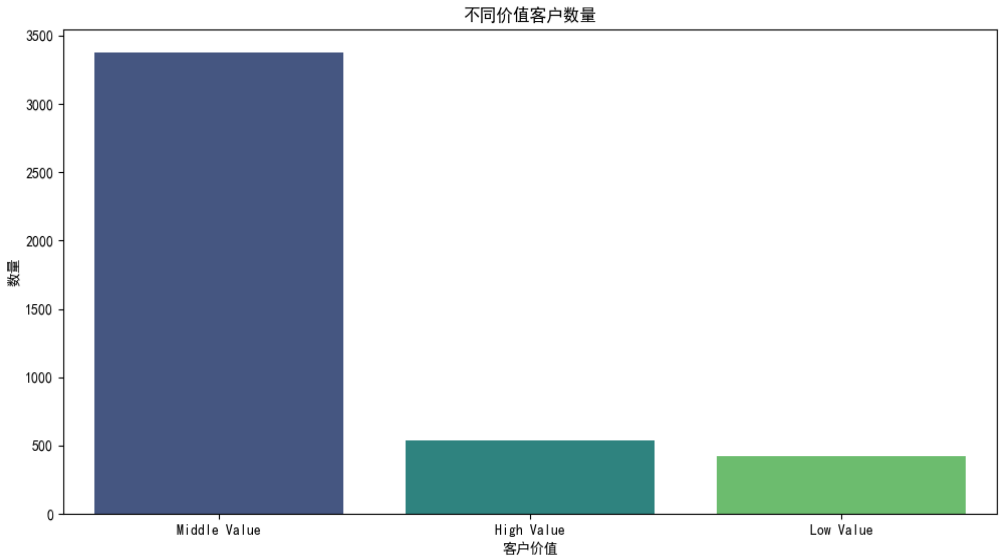

图 5-3-1 客户价值人数统计

6 问题三的求解

6.1 目标分析

在零售和电子商务领域，理解和评估顾客的价值是至关重要的。为了设计出能够吸引并保持顾客忠诚度的代金券策略，这里采用了 RFM 分析模型，这是一个基于顾客行为的细分和价值评估工具。RFM 模型通过考虑三个关键维度——最近购买时间（Recency）、购买频率（Frequency）和总消费金额（Monetary），帮助识别出不同价值的客户群体。通过这种方式，可以针对高价值客户、潜在客户以及需要特别关注的客户群体，设计出更具针对性的代金券策略。[8]

6.2 特征构造

指标	指标含义	单位
R	顾客最近一次消费时间距离统计时间的时间间隔	天
F	顾客在统计时间内的总消费频次	次
M	顾客在统计时间内的总消费金额	金额

R 表示顾客最近一次消费时间距离统计时间的时间间隔，其计算公式为：

$$R_i = \min(d_{end} - d_r)_{ij} \qquad \text{(公式 6-2-1)}$$

其中d_{end}表示统计时间内的截止时间，d_r表示顾客在电商平台最后一次消费的时间。

F 表示顾客在统计时间内的总消费频次，其计算公式为：

$$F_i = \sum_{J=1}^{M} f_{it} \qquad \text{(公式 6-2-2)}$$

其中f_{it}表示在日期 t 顾客 i 在该平台的购买次数。

M 表示顾客在统计时间内的总消费金额，计算方式为：

$$M_i = \sum_{J=1}^{M} m_{ij} \qquad \text{(公式 6-2-3)}$$

其中m_{ij}表示顾客 i 在该平台对商品 j 的购买金额。

综上所述， RFM 模型可表示为：

$$\mathrm{RFM} = \{R_i, F_i, M_i | \mathrm{i} \in 1,2..\mathrm{n}\} \qquad \text{(公式 6-2-4)}$$

6.3 计算 RFM 打分

Recency 得分：$R_{Sore} = qcut(\mathrm{Recency})$ [9] (公式 6-3-1)

其中 qcut 是基于分位数的离散化函数，将连续变量转换为 5 个等级，标签逆序。

Frequency 得分：$F_{Sore} = qcut(rank(\mathrm{Frequency}))$ (公式 6-3-2)
其中 rank 是排名函数，rank(Frequency)表示购买频率的排名。

Monetary 得分：$M_{Sore} = qcut(rank(\mathrm{Monetary}))$ (公式 6-3-3)
类似于 Frequency 得分，Monetary 得分也是基于消费金额的排名。

综合评分：$RFM_{Sore} = R_{Sore} + F_{Sore} + M_{Sore}$ (公式 6-3-4)

6.4 客户细分

利用指标分段细分法作为顾客价值的评估模型，根据 RFM 三个指标取值情况可以将顾客划分为不同的价值等级：

1. 基于原始数据计算平台所有顾客四个指标具体取值情况 Ri，Fi，Mi
2. 根据全体顾客 RFM 指标得分情况，将指标值划分为 N 段。
3. 根据每个顾客的得分情况将顾客一一纳入相应的分段区间。
4. 根据 RFM 评分，我们将顾客细分为不同的类别：高价值客户、低价值客户、最近购买的客户、频繁购买的客户等。

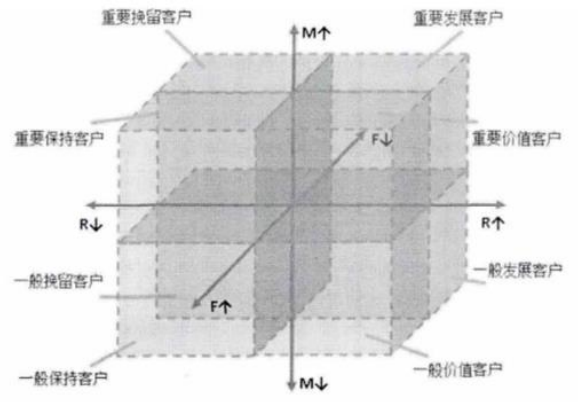

图 6-4-1 通用 RFM 模型客户细分矩阵[5]

6.5 代金券设计

通过基于客户购买能力的细分的结果和结合历来同行电商平台的优惠数据，我们为每个客户类别设计不同价格的代金券，并进行合理分配。

题一计算出的商品价格分布数据：

mean	std	min	max	25%	50%	75%
31.2	220.9	0	81427.5	12.5	19.5	37.5

代金券设计

```
voucher_design = {
    'High Value': 100,   # 高价值客户，代金券 100 元
    'Low Value': 10,     # 低价值客户，满 30 减 10 元
    'Recent': 50,        # 最近有购买行为的客户，满 100 减 50 元
    'Frequent': 70,      # 频繁购买的客户，满 200 减 70 元
    'Other': 15          # 中等价值客户，满 50 减 15 元
}
```

6.6 模型求解

最终通过一系列的分析和设计步骤后完成代码，确保代金券策略的有效性同时，还能提高顾客满意度和忠诚度。求解结果如下所示（源码可以查看 question3.py,详细评分结果可以在 customer_rfm.csv 中查看）：

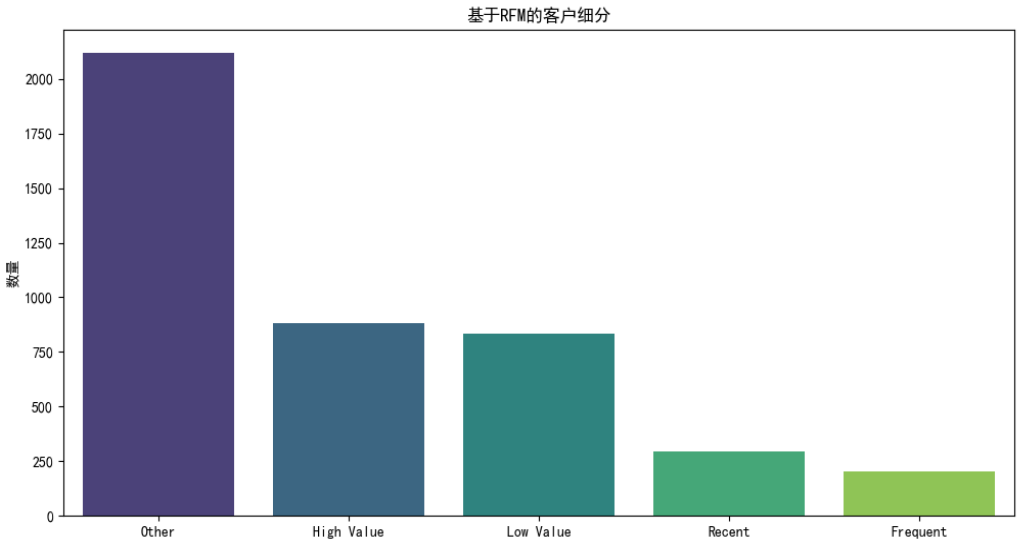

图 6-6-1 基于 RFM 的客户细分统计

CustomerId	RFM_Socre	Segment	Voucher
12346	115	Other	15
12347	555	High Value	100
12348	534	Other	15
12349	444	High Value	100
12350	122	Low Value	10
12352	345	Other	15
12353	111	Low Value	10
12354	134	Other	15
12355	112	Low Value	10
12356	445	High Value	100
12357	355	Other	15
12358	124	Other	15
12359	355	Other	15
12360	355	Other	15
12361	111	Low Value	10
12362	355	Other	15
12363	223	Other	15
12364	344	Other	15
12365	123	Other	15
……	……	……	……

表 6-6-1 客户 RFM 评分详细信息截取

7 问题四的求解

7.1 目标分析

在问题三中，已经构建了一个基于 RFM（最近一次消费时间、消费频率、消费金额）的推荐模型。现在，问题四的重点是评估这个模型的性能，特别是通过为每位用户分配合适的优惠券来实现。评估推荐模型的效果需要依赖于一些明确的指标，就推荐算法的评价而言，指标类型的选择因评价角度的不同而不同，这些指标的选择取决于评估者希望从哪个角度来评价推荐系统。

7.2 评估选择

从顾客的角度来看，推荐结果的质量是他们最关心的。为了衡量推荐结果的质量，可以考虑以下几种常见的评估指标来进行量化[6]：

1. 顾客满意度：顾客满意度指的是顾客对推荐结果的满意程度，通过衡量推荐系统预测顾客可能感兴趣的商品列表和顾客自身的真实感兴趣的商品之间的差异大小，是顾客对推荐结果最直观的感受。但是这种评价指标不是一种可测量的客观数据，因此想要获取该指标的具体数据往往需要企业通过问卷调查等一些方法来得到一些 能够衡量顾客对推荐结果的满意度的行为数据。这种指标虽然十分直观能够准确 反映出顾客的真实感受，但是其获取数据的难度较大，因此在衡量推荐结果时并不常用。

2. 预测准确率：预测准确度用来衡量推荐算法对顾客行为的预测能力，该指标与上述提到的顾客满意度不同，是一个可量化的指标，所以预测准确率是目前评估推荐算法最常用的指标。根据预测的目的不同评估的指标大致可以分为两类：一类是评分预测，目的是预测顾客的真实评分和预测评分之间的差异；另一类是分类准确性，关注的是推荐算法给出的有效推荐的百分比。

3. 描述性统计：描述性统计是对 RFM 模型各分段的基础数据进行分析，目的是了解各分段客户的基本情况和特征。观察不同 RFM 分段的客户在实际消费数据中的表现。例如，高价值客户是否确实消费最多，低价值客户是否消费最少。分析各 RFM 分段在时间上的稳定性。这是一种基于业务指标的评估：比较不同 RFM 分段客户的生命周期价值（Customer Lifetime Value, CLV）。分析不同 RFM 分段客户的回购率和留存率。在实际的营销活动中测试 RFM 模型的效果。例如，向高价值客户发送代金券，观察他们的响应率和消费金额。通过 A/B 测试评估 RFM 模型在实际促销活动中的效果。

7.3 评估指标

由于获取顾客的真实评价存在一定的难度，这里选择使用描述性统计作为机器评估的方法。在问题三中，已经介绍了 RFM 评分的计算方式。接下来将介绍如何根据 RFM 评分对客户进行分段，以及如何计算各分段的关键指标：

各分段客户数量：这是通过指示函数来确定的，指示函数用于判断客户是否属于特定的分段。

$$\text{CustomerCount}_{segment} = \sum_{i=1}^{n} I(Customer \in segment) \qquad \text{(公式 7-3-1)}$$

其中，I 为指示函数，表示客户是否属于该分段。

各分段客户的平均消费金额：这是通过将所有客户的消费金额求和，然后除以该分段的客户数量来计算的。

$$\text{AvgMonetary}_{segment} = \frac{\sum_{i=1}^{n} \text{Monetary}_i}{\text{CustomerCount}_{segment}} \qquad \text{(公式 7-3-2)}$$

回购率：这是通过计算在该分段中重复购买的客户数量与该分段客户总数的比值来确定的。

$$\text{RepeatPurchaseRate}_{segment} = \frac{\sum_{i=1}^{n} I(\text{客户重复购买})}{\text{CustomerCount}_{segment}} \qquad \text{(公式 7-3-3)}$$

然后通过公式设计出代码在问题三得出的数据基础上进行模型的评估

7.4 结果展示

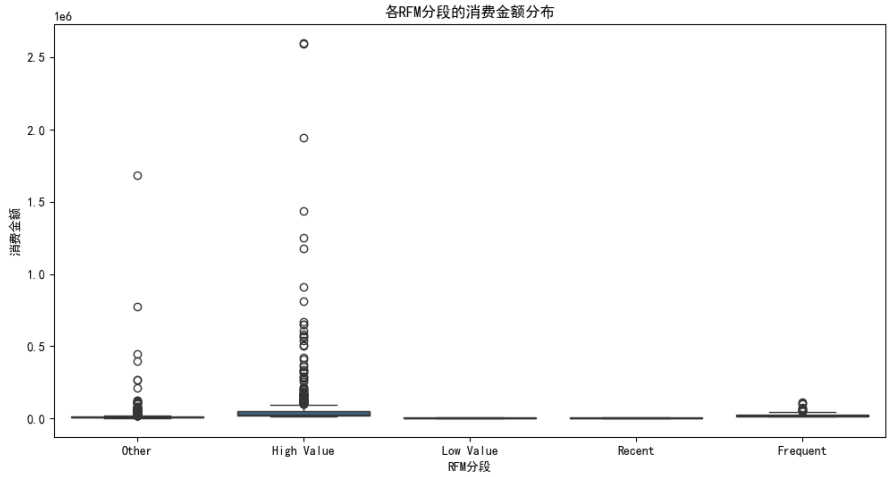

图 7-4-1 各 RFM 分段的消费金额分布

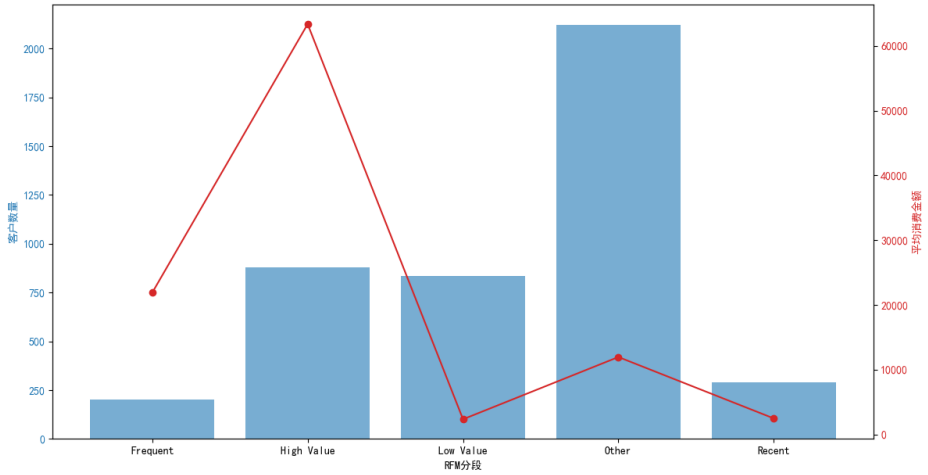

图 7-4-2 各 RFM 分段的平均消费金额分布

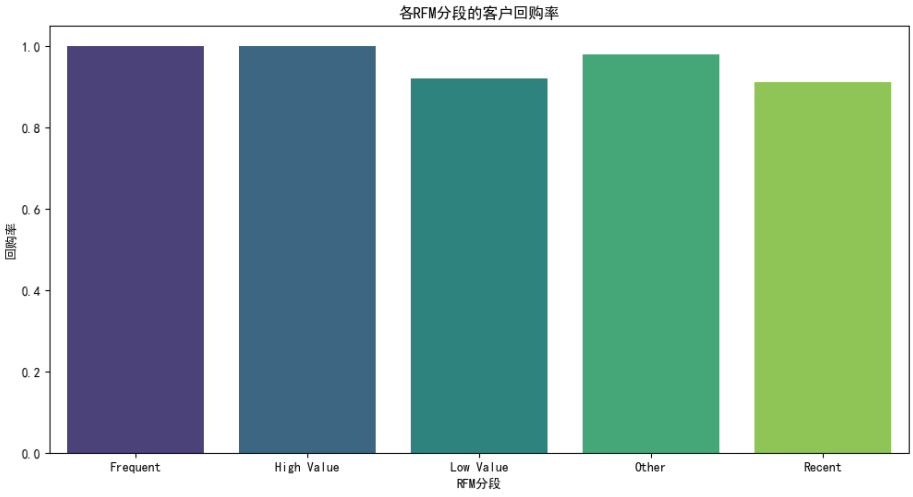

图 7-4-3 各 RFM 分段的客户回购率

从分析结果来看，可以清晰地发现 RFM 模型在区分客户价值方面发挥了显著的作用。如图 7-4-1 所示，高价值客户群体的消费能力得到了充分的体现，这表明 RFM 模型能够有效识别出那些对企业贡献最大的客户群体。

然而，在平均消费额方面，如图表 7-4-2 所展示的，情况则稍显复杂。尽管高价值客户在整体消费能力上表现突出，但在平均消费额上却相对较低。这表明，中等价值客户群体（Other）在平均消费额上扮演了主力军的角色，这可能意味着中等价值客户在消费频次上更为活跃。

进一步观察客户的回购率，我们可以发现，不同阶段客户的回购率相对均匀，如图 7-4-3 所示。这一发现表明，RFM 模型不仅能够识别出高价值客户，还能够在一定程度上促进各个价值阶段客户的回购行为。这种均匀的回购率分布，有效地增强了客户的忠诚度和粘性，从而提升了客户的活跃度和平台的整体热度。

综上所述，RFM 模型的应用在客户价值识别、消费行为分析以及客户忠诚度提升方面均显示出了其强大的功能和潜力。通过进一步的优化和应用，有理由相信 RFM 模型将为企业带来更多的客户洞察和商业价值。

8 问题五的求解

8.1 目标分析

设计一份优惠券的投放策略，需要综合考虑多个因素，以确保策略的有效性和最大化商业价值。正确的投放优惠券可以吸引顾客购买，增加销售额，尤其是对于新产品或季节性商品，扩大品牌影响力和市场份额，吸引新顾客。助力于电子消费产业的发展。同时通还能过优惠券的使用情况，收集顾客行为数据，为市场分析和产品开发提供参考。下面是一些需要考虑的策略要素：

1. **优惠券数量**：根据目标顾客群体的大小和预期的转化率来确定优惠券的发行数量。确保数量足够吸引顾客，但又不至于造成资源浪费。

2. **优惠券金额**：优惠券的金额需要平衡成本和吸引力。通常，优惠券的金额应该足够吸引顾客，但又不至于让公司亏损。可以通过市场调研来确定最佳的优惠金额。

3. **投放时间段**：选择投放优惠券的时间段，可以是节假日、特殊事件或销售淡季。例如，618、双十一等购物节是优惠券投放的绝佳时机。

4. **投放商品种类**：根据商品的销售情况和库存状况来决定哪些商品需要优惠券的支持。对于销量不佳的商品，可以提供更大的折扣来刺激销售。

5. **目标顾客群体**：明确优惠券的目标顾客群体，如新顾客、老顾客、高价值顾客等。根据不同群体的特点定制优惠券的策略。

8.2 优惠券投放设计

在第四题的策略上，进一步完善优惠券投放设计，增加投放时间段和投放数量的因素。这里物品种类可以通过机器学习获取，但是这里并不去实现。考虑到没有物品种类的数据，保持优惠券种类的通用性。具体投放时间段设置为两个消费节 618 和双 12，每个节日各投放 50 万的优惠券金额。以下是详细的投放策略设计：

策略表：

投放人群：基于 RFM 模型细分的客户群体得到的高价值客户、低价值客户、最近活跃客户、频繁购买客户、中等价值客户

优惠券金额：根据客户群体的价值设置不同金额的优惠券
高价值客户：消费券 100 元
低价值客户：消费券满 30 减 10 元
最近活跃客户：消费券满 100 减 50 元
频繁购买客户：消费券满 200 减 70 元
中等价值客户：消费券满 50 减 15 元

投放时间段：选择 618 和双 12 两个重要促销节点进行投放
618 促销节：每年 6 月 18 日
双 12 促销节：每年 12 月 12 日

投放数量：模拟每个节日总投放金额为 50 万元，根据客户群体的数量分配

投放商品种类：保持通用，适用于所有商品

通过这种策略，旨在最大化优惠券的效果，同时确保资源的合理分配。通过精准的目标定位和精心设计的优惠券策略，可以有效地提升客户的购买意愿和忠诚度，从而推动销售增长。

8.3 结果展示

（具体的代码实现可以参考 question5.py，以获得更详细的操作指导）

标签	人数	占比	优惠券数量(张)	分配金额(元)
Other	2120	0.490	16331.56	244973.42
High Value	880	0.203	1016.87	101687.08
Low Value	834	0.192	9637.16	96371.62
Recent	292	0.067	674.83	33741.62
Frequent	201	0.046	331.80	23226.25

表 8-3-1 分配结果展示

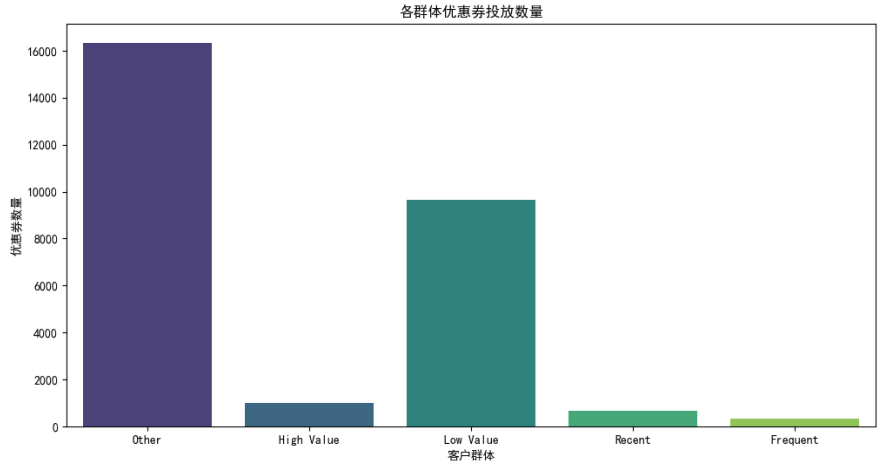

图 8-3-1 各群体优惠券分配数量

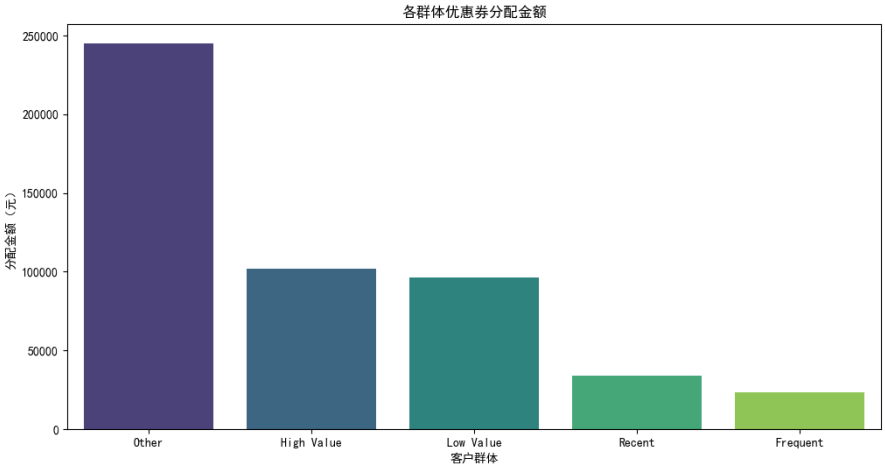

图 8-3-2 各群体优惠券分配金额

结论与展望

结论

本文通过基于 Python 的 Pandas 库和 RFM 模型对电商大数据进行分析，成功实现了对顾客行为的深入洞察和消费模式的有效识别。通过对数据的系统性分析，本文不仅识别出了最畅销的商品、购买力最强的顾客和国家，还对购买时间段和商品价格分布进行了细致的分析。此外，本文还构建了基于 RFM 模型的顾客细分策略，设计了个性化的代金券，并通过描述性统计对模型的有效性进行了评估。

结果表明，RFM 模型是一个强大的工具，能够帮助电商平台精准地识别出最有价值的顾客群体，并为这些群体设计具有针对性的营销活动。通过这种分析，企业能够提高销售额、增强市场竞争力，并最终实现收益的最大化。

展望

尽管本文的研究取得了一定的成果，但仍存在一些局限性和未来改进的空间。首先，当前的分析主要依赖于历史数据，未来的研究可以考虑引入实时数据流，以实现动态的顾客行为分析和即时的营销策略调整。其次，本文的 RFM 模型主要关注了顾客的购买行为，未来的研究可以进一步拓展模型，将顾客的社交行为、品牌互动等多维度因素纳入考量，以获得更全面的顾客画像，可以使用更优的 RFMQ 模型。

当前以人工智能为代表的新一轮科技革命在全球范围内迅速兴起,人工智能在各个行业的应用程度都呈现不断加深的趋势, 应用场景也越来越广泛.应支持电商平台企业加强能力建设,运用人工智能、云计算、虚拟现实等先进技术,创新管理模式、运营模式和营销模式,打造新的业 务增长点、提升用户体验、保持核心竞争力.随着电商平台数字技术能力的不断提升,其助力传统产业转型升级的能力也将与日俱增,在赋能制造业转型升级、 推动农业数字化转型、促进服务业线上线下融合转型及扩大内需等方面将创造更大价值.未来,随着平台经济在数字技术上不断增加投入,以及系统化、长期稳定、常态化的平台经济综合监管生态的形成和完善,平台经济将加快构建以数实融合为抓手的更高水平供求匹配新机制,在促进经济高质量发展中发挥更重要作用,更多中国电商平台企业也将成为创新引领实体经济发展的重要力量[1]。

最后，本文的研究主要集中在单一的电商平台，未来的研究可以考虑跨平台的数据整合，以获得更广泛的市场视角和更丰富的顾客洞察。通过跨平台分析，企业能够更好地理解顾客在不同平台间的行为差异，从而设计出更具吸引力的跨平台营销活动，不断探索新的分析方法和策略，实现可持续发展。

文献引用

[1] 国家统计局. 中国电子商务报告 2022[R]. 国家统计局, 2022.

[2] 曹洁. Python 数据分析[M]. 清华大学出版社, 2020.

[3].Pandas：强大的 Python 数据分析支持库[EB/OL]. [2024-5-20]. https://pypandas.cn/docs/.

[4] 康钰浩.大数据背景下企业统计分析应用研究[J].中国中小企业,2023,(12):162-164.

[5] 李宗阳.基于 RFM 模型的电商平台个性化推荐方法研究[D].河南财经政法大学,2023.DOI:10.27113/d.cnki.ghncc.2023.000737.

[6] 刘朝华,梅强,蔡淑琴.基于 RFM 的客户分类及价值评价模型[J].技术经济与管理研究,2012(5):33-36

[7] Verhoef, P. C., & Donkers, B. (2001). Predicting customer potential value: An application in the insurance industry. Decision Support Systems, 32(2), 189-199.

[8] Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). "RFM" and "CLV": Using Iso-Value Curves for Customer Base Analysis. Journal of Marketing Research, 42(4), 415-430.

[9] Bose, I., & Chen, X. (2009). Quantitative models for direct marketing: A review from systems perspective. European Journal of Operational Research, 195(1), 1-16.

[10] Liu, Y., & Shankar, V. (2011). The Dynamic Impact of Product-Harm Crises on Brand Preference and Purchase Behavior: An Empirical Analysis. Management Science, 57(5), 915–929. Link

附录

queston1.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

# 读取数据
df = pd.read_csv('data.csv')

# 数据预处理
df.dropna(inplace=True) #删除包含缺失值的行
df.drop_duplicates(inplace=True) #删除数据中完全的重复行
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y %H:%M', errors='coerce') #转换日期列，处理非法日期
# df = df.dropna(subset=['Date']) #去除转换日期失败的行
df = df[(df['Quantity'] > 0)]    #删除 Quantity 列值为负数和 0 的行

# 保存预处理后的数据为 data3.csv
# df.to_csv('data3.csv', index=False)

# 设置中文字体，解决中文显示问题
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 最畅销的商品前 10 名
top_products = df.groupby('Product')['Quantity'].sum().sort_values(ascending=False).head(10)
print("最畅销的商品前 10 名：")
print(top_products)
print("\n")

# 购买商品总数最多的顾客前 10 名
top_customers =
df.groupby('CustomerID')['Quantity'].sum().sort_values(ascending=False).head(10)
print("购买商品总数最多的顾客前 10 名：")
print(top_customers)
print("\n")
```

```
# 购买商品总数最多的国家前 10 名
top_countries = df.groupby('Country')['Quantity'].sum().sort_values(ascending=False).head(10)
print("购买商品总数最多的国家前 10 名：")
print(top_countries)
print("\n")

# 购买商品的时间段分析
df['Hour'] = df['Date'].dt.hour
time_distribution = df.groupby('Hour')['Quantity'].sum()
print("购买商品的时间段分布：")
print(time_distribution)
print("\n")

# 商品价格的分布分析
price_distribution = df['Price']
print("商品价格分布：")
print(price_distribution.describe())
print("\n")
prices = df['Price']
price_counts = prices.value_counts().sort_index()

# 数据可视化
# 最畅销的商品前 10 名图表
plt.figure(figsize=(12, 6))
sns.barplot(x=top_products.values, y=top_products.index, palette='viridis')
plt.title('最畅销商品 TOP10')
plt.xlabel('销售数量')
plt.ylabel('')
plt.subplots_adjust(left=0.22)
plt.show()

# 购买商品总数最多的顾客前 10 名图表
plt.figure(figsize=(10, 6))
sns.barplot(x=top_customers.index, y=top_customers.values, palette='viridis')
plt.title('购买商品总数顾客 TOP10')
plt.xlabel('顾客 ID')
plt.ylabel('购买数量')
plt.show()


# 购买商品总数最多的国家前 10 名图表
plt.figure(figsize=(12, 6))
sns.barplot(x=top_countries.values, y=top_countries.index, palette='viridis')
plt.title('购买商品总数的国家 TOP10')
```

```
plt.xlabel('购买数量（百万）')
plt.ylabel('国家')
plt.show()

# 购买商品的时间段分布图表
plt.figure(figsize=(12, 6))
sns.lineplot(x=time_distribution.index, y=time_distribution.values, marker='o')
plt.title('购买商品时间段分布图')
plt.xlabel('时间')
plt.ylabel('购买数量')
plt.show()

# 价格频率图
plt.figure(figsize=(12, 6))
sns.lineplot(x=price_counts.index, y=price_counts.values, marker='o')
plt.title('价格频率图')
plt.xlabel('价格')
plt.ylabel('数量频率')
plt.show()
```

question2.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

# 读取数据
df = pd.read_csv('data3.csv')

# 设置中文字体，解决中文显示问题
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 构建顾客标签

# 1. 总消费金额
df['TotalAmount'] = df['Quantity'] * df['Price']
customer_total_amount = df.groupby('CustomerID')['TotalAmount'].sum()
# 2. 平均订单金额
```

```
customer_avg_order_amount = df.groupby('CustomerID')['TotalAmount'].mean()
# 3. 购买频率（订单次数）
customer_order_count = df.groupby('CustomerID')['EventID'].count()
# 4. 最常购买的商品
customer_favorite_product = df.groupby(['CustomerID', 'Product'])['Quantity'].sum().reset_index()
customer_favorite_product = customer_favorite_product.loc[customer_favorite_product.groupby('CustomerID')['Quantity'].idxmax()]

# 合并标签数据
customer_profile = pd.DataFrame({
    'TotalAmount': customer_total_amount,
    'AvgOrderAmount': customer_avg_order_amount,
    'OrderCount': customer_order_count
}).reset_index()

customer_profile = customer_profile.merge(customer_favorite_product[['CustomerID', 'Product']], on='CustomerID')
customer_profile.rename(columns={'Product': 'FavoriteProduct'}, inplace=True)

# 创建标签
def create_labels(row):
    if (row['TotalAmount'] >= customer_profile['TotalAmount'].quantile(0.8) and
        row['AvgOrderAmount'] >= customer_profile['AvgOrderAmount'].quantile(0.5) and
        row['OrderCount'] >= customer_profile['OrderCount'].quantile(0.6)):
        return 'High Value'
    elif (row['TotalAmount'] <= customer_profile['TotalAmount'].quantile(0.2) and
          row['AvgOrderAmount'] <= customer_profile['AvgOrderAmount'].quantile(0.5) and
          row['OrderCount'] <= customer_profile['OrderCount'].quantile(0.4)):
        return 'Low Value'
    else:
        return 'Middle Value'
customer_profile['CustomerValue'] = customer_profile.apply(create_labels, axis=1)

print(customer_profile.head()) # 查看前 5 行数据

# 可视化顾客画像
```

```
plt.figure(figsize=(12, 6))
sns.countplot(data=customer_profile, x='CustomerValue', palette='viridis')
plt.title('不同价值客户数量')
plt.xlabel('客户价值')
plt.ylabel('数量')
plt.show()

# 1. 总消费金额分布
plt.figure(figsize=(12, 6))
sns.histplot(customer_profile['TotalAmount'], bins=50, kde=True,
color='blue')
plt.title('客户总消费金额频率')
plt.xlabel('总消费金额')
plt.ylabel('频率')
plt.show()

# 2. 平均订单金额分布
plt.figure(figsize=(12, 6))
sns.histplot(customer_profile['AvgOrderAmount'], bins=50, kde=True,
color='green')
plt.title('客户平均消费额频率')
plt.xlabel('平均消费额')
plt.ylabel('频率')
plt.show()

# 3. 订单次数分布
plt.figure(figsize=(12, 6))
sns.histplot(customer_profile['OrderCount'], bins=50, kde=True, color='red')
plt.title('客户订单次数频率')
plt.xlabel('订单数')
plt.ylabel('频率')
plt.show()

# 导出数据到CSV 文件
customer_profile.to_csv('customer_value.csv', index=False)
```

question3.py

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import datetime as dt
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_csv('data.csv')

# 数据预处理
df.dropna(inplace=True) #删除包含缺失值的行
df.drop_duplicates(inplace=True) #删除数据中完全的重复行
df['Date']    =    pd.to_datetime(df['Date'],    format='%m/%d/%Y    %H:%M',
errors='coerce') #转换日期列，处理非法日期
# df = df.dropna(subset=['Date']) #去除转换日期失败的行
df = df[(df['Quantity'] > 0)]  #删除 Quantity 列值为负数和 0 的行

# 删除无法解析的日期
df = df.dropna(subset=['Date'])

# 计算总消费金额
df['TotalAmount'] = df['Quantity'] * df['Price']

# 设定分析基准日期为数据集中最近的日期
latest_date = df['Date'].max() + dt.timedelta(days=1)

# 构建 RFM 特征
rfm = df.groupby('CustomerID').agg({
    'Date': lambda x: (latest_date - x.max()).days,
    'EventID': 'count',
    'TotalAmount': 'sum'
})

# 重命名列
rfm.rename(columns={
    'Date': 'Recency',
    'EventID': 'Frequency',
    'TotalAmount': 'Monetary'
}, inplace=True)

# 计算 RFM 打分
rfm['R_Score'] = pd.qcut(rfm['Recency'], 5, labels=range(5, 0, -1))
rfm['F_Score']    =    pd.qcut(rfm['Frequency'].rank(method='first'),    5,
```

```
labels=range(1, 6))
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 5, labels=range(1, 6))

# 计算 RFM 综合评分
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) +
rfm['M_Score'].astype(str)

# 查看 RFM 打分
print(rfm.head())

# 客户细分
def rfm_segment(df):
    if df['R_Score'] in [4, 5] and df['F_Score'] in [4, 5] and df['M_Score']
in [4, 5]:
        return 'High Value'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [1, 2] and df['M_Score']
in [1, 2]:
        return 'Low Value'
    elif df['R_Score'] in [4, 5] and df['F_Score'] in [1, 2] and df['M_Score']
in [1, 2]:
        return 'Recent'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [4, 5] and df['M_Score']
in [4, 5]:
        return 'Frequent'
    else:
        return 'Other'

rfm['Segment'] = rfm.apply(rfm_segment, axis=1)

# 查看客户细分
print(rfm['Segment'].value_counts())

# 代金券设计
voucher_design = {
    'High Value': 100,  # 高价值客户，代金券 100 元
    'Low Value': 10,    # 低价值客户，代金券 10 元
    'Recent': 50,       # 最近有购买行为的客户，代金券 50 元
    'Frequent': 70,     # 频繁购买的客户，代金券 70 元
    'Other': 15         # 其他客户（中等价值），代金券 15 元
}

# 为每个客户分配代金券
rfm['Voucher'] = rfm['Segment'].map(voucher_design)
```

```
# 查看分配结果
print(rfm.head())

# 可视化 RFM 细分结果
plt.figure(figsize=(12, 6))
sns.countplot(data=rfm, x='Segment', palette='viridis')
plt.title('基于 RFM 的客户细分')
plt.xlabel('')
plt.ylabel('数量')
plt.show()

# 导出结果到 CSV 文件
rfm.to_csv('customer_rfm.csv')
```

question4.py

```
import pandas as pd
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_csv('data.csv')

# 数据预处理
df.dropna(inplace=True) #删除包含缺失值的行
df.drop_duplicates(inplace=True) #删除数据中完全的重复行
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y %H:%M',
errors='coerce') #转换日期列，处理非法日期
# df = df.dropna(subset=['Date']) #去除转换日期失败的行
df = df[(df['Quantity'] > 0)] #删除 Quantity 列值为负数和 0 的行



# 删除无法解析的日期
df = df.dropna(subset=['Date'])
```

```
# 计算总消费金额
df['TotalAmount'] = df['Quantity'] * df['Price']

# 设定分析基准日期为数据集中最近的日期
latest_date = df['Date'].max() + dt.timedelta(days=1)

# 构建 RFM 特征
rfm = df.groupby('CustomerID').agg({
    'Date': lambda x: (latest_date - x.max()).days,
    'EventID': 'count',
    'TotalAmount': 'sum'
})

# 重命名列
rfm.rename(columns={
    'Date': 'Recency',
    'EventID': 'Frequency',
    'TotalAmount': 'Monetary'
}, inplace=True)

# 计算 RFM 打分
rfm['R_Score'] = pd.qcut(rfm['Recency'], 5, labels=range(5, 0, -1))
rfm['F_Score']    =    pd.qcut(rfm['Frequency'].rank(method='first'),    5,
labels=range(1, 6))
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 5, labels=range(1, 6))

# 计算 RFM 综合评分
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) +
rfm['M_Score'].astype(str)

# 客户细分
def rfm_segment(df):
    if df['R_Score'] in [4, 5] and df['F_Score'] in [4, 5] and df['M_Score']
in [4, 5]:
        return 'High Value'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [1, 2] and df['M_Score']
in [1, 2]:
        return 'Low Value'
    elif df['R_Score'] in [4, 5] and df['F_Score'] in [1, 2] and df['M_Score']
in [1, 2]:
        return 'Recent'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [4, 5] and df['M_Score']
in [4, 5]:
        return 'Frequent'
```

```
    else:
        return 'Other'

rfm['Segment'] = rfm.apply(rfm_segment, axis=1)

# 代金券设计
voucher_design = {
    'High Value': 100,  # 高价值客户，代金券 100 元
    'Low Value': 10,    # 低价值客户，代金券 10 元
    'Recent': 50,       # 最近有购买行为的客户，代金券 50 元
    'Frequent': 70,     # 频繁购买的客户，代金券 70 元
    'Other': 15         # 其他客户（中等价值），代金券 15 元
}

# 为每个客户分配代金券
rfm['Voucher'] = rfm['Segment'].map(voucher_design)


# 开始解决问题四
# 查看各 RFM 分段的描述性统计
print(rfm.groupby('Segment').agg({
    'Recency': ['mean', 'median', 'std'],
    'Frequency': ['mean', 'median', 'std'],
    'Monetary': ['mean', 'median', 'std']
}))

# 可视化各 RFM 分段的消费金额分布
plt.figure(figsize=(12, 6))
sns.boxplot(data=rfm, x='Segment', y='Monetary', palette='viridis')
plt.title('各 RFM 分段的消费金额分布')
plt.xlabel('RFM 分段')
plt.ylabel('消费金额')
plt.show()

# 分析各 RFM 分段的客户数量和平均消费金额
rfm_analysis = rfm.groupby('Segment').agg({
    'Recency': 'count',
    'Monetary': 'mean'
}).rename(columns={'Recency': 'CustomerCount', 'Monetary': 'AvgMonetary'})

print(rfm_analysis)

# 可视化客户数量和平均消费金额
fig, ax1 = plt.subplots(figsize=(12, 6))
```

```
color = 'tab:blue'
ax1.set_xlabel('RFM 分段')
ax1.set_ylabel('客户数量', color=color)
ax1.bar(rfm_analysis.index,    rfm_analysis['CustomerCount'],    color=color,
alpha=0.6)
ax1.tick_params(axis='y', labelcolor=color)
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('平均消费金额', color=color)
ax2.plot(rfm_analysis.index,    rfm_analysis['AvgMonetary'],    color=color,
marker='o')
ax2.tick_params(axis='y', labelcolor=color)
fig.tight_layout()
plt.title('各 RFM 分段的客户数量和平均消费金额')
plt.show()

# 回购率 = 回购客户数量 / 总客户数量
# 模拟回购数据
np.random.seed(42)
df['IsRepeatedPurchase'] = np.random.choice([0, 1], size=len(df), p=[0.7,
0.3])
repeat_purchase                                                          =
df.groupby('CustomerID')['IsRepeatedPurchase'].max().reset_index()

rfm = rfm.reset_index().merge(repeat_purchase, on='CustomerID', how='left')
rfm['IsRepeatedPurchase'].fillna(0, inplace=True)

repeat_purchase_rate = rfm.groupby('Segment')['IsRepeatedPurchase'].mean()
print(repeat_purchase_rate)

# 可视化回购率
plt.figure(figsize=(12, 6))
sns.barplot(x=repeat_purchase_rate.index,      y=repeat_purchase_rate.values,
palette='viridis')
plt.title('各 RFM 分段的客户回购率')
plt.xlabel('RFM 分段')
plt.ylabel('回购率')
plt.show()
```

question5.py

```
import pandas as pd
import datetime as dt
import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
# 读取数据
df = pd.read_csv('data.csv')

# 数据预处理
df.dropna(inplace=True) # 删除包含缺失值的行
df.drop_duplicates(inplace=True) # 删除数据中完全的重复行
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y %H:%M',
errors='coerce') # 转换日期列，处理非法日期
df = df[(df['Quantity'] > 0)] # 删除 Quantity 列值为负数和 0 的行

# 删除无法解析的日期
df = df.dropna(subset=['Date'])

# 计算总消费金额
df['TotalAmount'] = df['Quantity'] * df['Price']

# 设定分析基准日期为数据集中最近的日期
latest_date = df['Date'].max() + dt.timedelta(days=1)

# 构建 RFM 特征
rfm = df.groupby('CustomerID').agg({
    'Date': lambda x: (latest_date - x.max()).days,
    'EventID': 'count',
    'TotalAmount': 'sum'
})

# 重命名列
rfm.rename(columns={
    'Date': 'Recency',
    'EventID': 'Frequency',
    'TotalAmount': 'Monetary'
}, inplace=True)
```

```
# 计算 RFM 打分
rfm['R_Score'] = pd.qcut(rfm['Recency'], 5, labels=range(5, 0, -1))
rfm['F_Score'] = pd.qcut(rfm['Frequency'].rank(method='first'), 5,
labels=range(1, 6))
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 5, labels=range(1, 6))

# 计算 RFM 综合评分
rfm['RFM_Score'] = rfm['R_Score'].astype(str) +
rfm['F_Score'].astype(str) + rfm['M_Score'].astype(str)

# 客户细分
def rfm_segment(df):
    if df['R_Score'] in [4, 5] and df['F_Score'] in [4, 5] and
df['M_Score'] in [4, 5]:
        return 'High Value'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [1, 2] and
df['M_Score'] in [1, 2]:
        return 'Low Value'
    elif df['R_Score'] in [4, 5] and df['F_Score'] in [1, 2] and
df['M_Score'] in [1, 2]:
        return 'Recent'
    elif df['R_Score'] in [1, 2] and df['F_Score'] in [4, 5] and
df['M_Score'] in [4, 5]:
        return 'Frequent'
    else:
        return 'Other'

rfm['Segment'] = rfm.apply(rfm_segment, axis=1)

# 计算每个群体的客户数量
segment_counts = rfm['Segment'].value_counts().reset_index()
segment_counts.columns = ['Segment', 'Count']

# 代金券设计
voucher_design = {
    'High Value': 100,  # 高价值客户，代金券 100 元
    'Low Value': 10,    # 低价值客户，代金券 10 元
    'Recent': 50,       # 最近有购买行为的客户，代金券 50 元
    'Frequent': 70,     # 频繁购买的客户，代金券 70 元
    'Other': 15         # 其他客户（中等价值），代金券 15 元
}

# 合并客户数量数据
```

```
segment_counts['VoucherAmount'] =
segment_counts['Segment'].map(voucher_design)

# 总投放金额
total_amount = 1000000  # 100 万元

# 每个节日投放金额
amount_per_event = total_amount / 2  # 50 万元

# 计算每个群体在整体客户中的比例
segment_counts['Proportion'] = segment_counts['Count'] /
segment_counts['Count'].sum()

# 根据比例分配总投放金额
segment_counts['AllocatedAmount'] = segment_counts['Proportion'] *
amount_per_event

# 计算每种类型优惠券的投放数量
segment_counts['VoucherCount'] = segment_counts['AllocatedAmount'] /
segment_counts['VoucherAmount']

# 打印结果
print("各群体优惠券投放数量和金额分配：")
print(segment_counts[['Segment', 'Count', 'Proportion',
'AllocatedAmount', 'VoucherAmount', 'VoucherCount']])

# 可视化各群体优惠券投放数量
plt.figure(figsize=(12, 6))
sns.barplot(data=segment_counts, x='Segment', y='VoucherCount',
palette='viridis')
plt.title('各群体优惠券投放数量')
plt.xlabel('客户群体')
plt.ylabel('优惠券数量')
plt.show()
# 可视化各群体优惠券分配金额
plt.figure(figsize=(12, 6))
sns.barplot(data=segment_counts, x='Segment', y='AllocatedAmount',
palette='viridis')
plt.title('各群体优惠券分配金额')
plt.xlabel('客户群体')
plt.ylabel('分配金额（元）')
plt.show()
```