

PJI Consultancy Squad

Consultancy Project



"Harnessing Data to Understand and Engage Your Shoppers"

BUS211A – Foundations of Data Analytics

Group 8

Group members: Payal Kumari, [Josephine Boampong](#) & [Ian Ho](#)

Report 1 – Data Preparation

1.1 Consultancy and Data Source

a. Consultancy:

Our consultancy analyzes and interprets data to provide insights into customer behavior, trends, and patterns. For the "Customer Personality Analysis" dataset, this could involve profiling customer segments, identifying key factors influencing customer preferences, and recommending strategies for targeted marketing. The factors we are looking at that describe the character include: a unique identifier for each customer (ID), customer's birth year, education level, marital status, income, number of children, number of teenagers, date the customer enrolled in the company, number of days since customer's last purchase, and if the customer has made a complaint in the last 2 years (1 or 0). There is also data on the customers' preferences regarding what they buy. The dataset we are looking at has data on the different food products people buy. We are also tracking the number of purchases made with a discount when they accept offers depending on promotions, and if they buy online or in-store. The reason why this data set was created was to help businesses modify their product based on their target customers from different types of customer segments. Meaning, that instead of spending money to come up with a new product that they think all customers will buy, they come up with ones based on the customer segments that we have mentioned. We believe that through this analysis customers will be able to get their needs met more easily, in turn creating much better customers.

b. Data Source:

The dataset is from Kaggle, a popular platform for data science and machine learning resources. Kaggle datasets are often contributed by the community and can be a valuable resource for practical data analysis projects. It was also mentioned that the data was collected by Omar Romero Hernandez, a professor at UC Berkeley Hass Business School.

c. Choice of Data Source:

We chose Kaggle because it hosts a wide array of datasets across various domains, and the "Customer Personality Analysis" dataset is specifically tailored for understanding customer behavior. This makes it highly relevant for projects focused on customer analytics.

d. Credibility:

Kaggle datasets are contributed by a mix of professionals and enthusiasts. The credibility of a specific dataset depends on the quality of the data and its source. For the "Customer Personality Analysis" dataset, it's important to review the dataset's description and any accompanying documentation to assess its quality and relevance. It has around 151k downloads and a rating of

9.71 usability. Of the 151k downloads, there are 2,800 in the last 30 days. This is about 93 downloads per day, proving that companies are continuing to use this data and that it's still relevant.

While Kaggle is a reputable platform, it's always good practice to validate the dataset and consider its limitations in your analysis. We hope by conducting more of these personality analyses companies will be able to target more of their ideal customers.

e. Introduction To PJI Consultancy Squad

PJI Consultants is a premier consultancy specializing in data analysis and interpretation to deliver actionable insights into customer behavior, trends, and purchasing patterns. Our expertise lies in leveraging data to guide businesses in optimizing their marketing strategies and product offerings, ensuring customer satisfaction and long-term success.

Our Mission

Our mission is to empower businesses with data-driven insights that allow them to understand their customers better and develop tailored products and marketing strategies that meet the specific needs of different customer segments while optimizing marketing strategies and product offerings.

Our Vision

To be the preferred consultancy partner for businesses seeking to enhance their decision-making processes through comprehensive customer data analysis, ultimately fostering stronger customer relationships and driving sustained growth.

Our Services

- **Customer Segmentation:** We analyze customer demographics and behavior to create distinct customer segments.
- **Data-Driven Marketing:** Our insights enable personalized marketing strategies that enhance customer engagement.
- **Customer Behavior Profiling:** Using data, we analyze customer purchasing patterns to aid businesses in understanding preferences to help companies meet specific customer needs.

Customer Background

Walmart, a global leader in retail, operates across multiple channels including in-store and online platforms. With millions of customers purchasing a wide range of products, Walmart is uniquely positioned to leverage its vast amounts of data to gain deeper insights into customer behavior.

However, despite this wealth of data, Walmart faces challenges in aligning its marketing

strategies with customer preferences, reducing marketing costs, and increasing the efficiency of its campaigns. Walmart seeks to better segment its diverse customer base to target the right customers with the right products and deliver more personalized experiences that drive customer loyalty and revenue growth.

Value Proposition

Our solution was designed to:

Phase 1: PJI Consultants will conduct an in-depth analysis of Walmart's customer data, including income, spending habits, product preferences, and engagement levels. Using this data, we will segment Walmart's customer base into distinct groups, allowing for highly personalized marketing efforts. For example, high-income customers can be targeted with premium product offerings, while family-focused customers can receive tailored promotions for family-sized goods and seasonal deals. This segmentation will enhance Walmart's ability to offer the right products to the right customers, improving engagement and increasing sales.

Phase 2: Once customer segments are identified, PJI Consultants will help Walmart optimize its marketing strategies by analyzing the effectiveness of past campaigns. We will identify which campaigns had the highest impact and which customer segments responded best to them. With this insight, Walmart can reduce inefficient marketing spend and focus on strategies that deliver the highest return on investment. Additionally, predictive modeling will allow Walmart to forecast customer behavior, ensuring campaigns are timed to match customer buying cycles.

Phase 3: PJI Consultants will provide Walmart with key insights into product performance, customer preferences, and emerging market trends. By understanding what products different customer segments prefer, Walmart can adjust its inventory and product offerings to better match demand. For instance, if data shows that certain geographic regions are purchasing more eco-friendly products, Walmart can increase its stock of sustainable goods in those areas. This data-driven approach to product development will ensure Walmart stays competitive and meets the evolving needs of its customers.

Phase 2: Target Customer List Development for Walmart

In this pivotal second phase of our consultancy engagement with Walmart, PJI Consultants is excited to announce a project that will significantly impact Walmart's customer engagement strategies. Building on the strong foundation of our newly implemented Customer Database Management System in Phase 1, Phase 2 will focus on creating a Target Customer List for an upcoming personalized product recommendation and loyalty program. Walmart's goal is to optimize its product offerings by targeting the right customers for specific promotions, ensuring customer satisfaction, and driving revenue growth.

Product Eligibility and Qualification Criteria:

To achieve Walmart's goals for personalized marketing, we have designed a robust set of criteria that will help identify the ideal customers for targeted offers:

Income Criteria: Customers with an annual income of \$50,000 or more will qualify for premium product promotions.

Recency Criteria: Customers who have purchased in the last 90 days will be eligible for personalized product recommendations.

Purchase Behavior Criteria: Customers with a high purchase frequency across key product categories (e.g., electronics, and groceries) will qualify for Walmart's new loyalty program.

Justifications for the Criteria:

The selection of these criteria reflects a comprehensive strategy designed to maximize the impact of Walmart's marketing efforts.

Income Criteria: Customers with higher income levels are more likely to purchase premium products, making them ideal targets for promotions on high-value items such as electronics or home appliances. This ensures that Walmart reaches customers who have the purchasing power to engage with premium offers.

Recency Criteria: By focusing on customers who have made recent purchases, Walmart can maintain strong customer engagement, targeting those who are actively shopping, and therefore more likely to respond to timely offers.

Purchase Behavior Criteria: Customers with a history of frequent purchases in certain categories demonstrate loyalty and preference for Walmart's offerings, making them ideal candidates for the new loyalty program. These customers will be incentivized to continue their buying habits with personalized promotions and rewards.

The use of these specific criteria ensures that Walmart can maximize customer engagement, reduce marketing costs, and deliver promotions that resonate with high-value customers.

Further Considerations and Analysis:

Priority Status for Customer Engagement: Customers will be segmented into three priority categories: "High_Priority," "Medium_Priority," and "Low_Priority."

High-Priority: Customers with an annual income above \$100,000 and recent purchases qualify for premium loyalty offers and exclusive promotions.

Medium-Priority: Customers with an annual income between \$50,000 and \$100,000 and consistent purchasing behavior qualify for the core loyalty program.

Low-Priority: Customers with lower income or less frequent purchasing habits, who are eligible for basic promotions but may not be ideal targets for premium offers.

This prioritization enables Walmart to allocate marketing resources more efficiently, ensuring high-value customers receive the most attention and incentives to increase their purchasing frequency.

Customer Age Analysis: Analyzing the average age of customers who make frequent purchases across various categories will provide insights into their life stage, enabling Walmart to tailor product offerings. For example, younger customers may prefer electronics and apparel, while older customers may focus on home goods or groceries. This allows Walmart to adjust its product mix and promotional strategies to better meet the needs of its customers based on life stages.

Our Solution

At PJI Consultants, our approach is driven by data precision. To meet Walmart's requirements, our team has conducted a thorough analysis, employing 28 SQL queries to carefully select customers who meet the desired criteria. By leveraging Walmart's extensive customer data, we have developed a highly targeted list that aligns with the goals of the new loyalty program and premium product promotions.

Through data-driven insights, Walmart will be able to:

- Personalize marketing efforts to engage high-value customers.
- Increase the efficiency of promotional spending by focusing on customers who are most likely to respond.
- Strengthen customer loyalty by offering tailored promotions that resonate with specific customer segments.
- This approach ensures that Walmart can maximize its marketing resources and drive success in upcoming campaigns by engaging customers who are both financially capable and primed to respond positively to personalized offers.

1.2 Data Preparation

Data preparation is a critical phase in the data analysis workflow, ensuring that the dataset is both accurate and ready for insightful analysis. This process begins with a thorough evaluation of the data source's credibility, checking for reliability and consistency. Initial steps involve importing the data and performing preliminary explorations to understand its structure. Key activities in data preparation include cleaning the data by addressing missing values, correcting data types, and managing skewness. Techniques such as normalization or standardization are applied to bring numerical values to a common scale. Following these steps, the final data quality is reassessed to confirm that all issues have been resolved and that the dataset is robust. Additionally, data preparation often involves segmenting or grouping data to highlight patterns and trends, which may include categorizing numeric values or converting data formats. This meticulous preparation is essential for ensuring that subsequent analyses are based on high-quality, well-organized data, leading to reliable and actionable insights. Therefore, to make our data more valid we have cleaned it with the help of Python, and for visualization, we have used Tableau. Further, the whole process of preparing the chosen dataset has been discussed below;

1.3 Initial Data Quality and Source Credibility:

Data Source:

- **Source Name:** Kaggle
- **Dataset Title:** Customer Personality Analysis
- **Link to Dataset:** Customer Personality Analysis on Kaggle
<https://www.kaggle.com/datasets/imakash3011/customer-personality-analysis>

Source Credibility:

- **Reputation of Source:** Kaggle is a well-regarded platform for data science datasets, commonly used for educational and competitive purposes. The dataset's credibility is high but should be verified for specific notes or conditions.
- **Data Origin:** The dataset's origin is not explicitly mentioned but is intended for analysis of the company's ideal customers. The context of the data collection and its timeliness should be considered.
- **Date of Collection:** While the dataset does not specify the exact collection date, it was uploaded to Kaggle three years ago. Understanding the time frame of data collection is valuable for assessing its

relevance, and this context helps us make informed decisions about how to use the data effectively.

1.4 Initial Data Quality:

Before diving into the analysis, it's important to review and address some initial data quality considerations. Ensuring the dataset's integrity and consistency is crucial for producing reliable results. We have identified several areas that need attention to optimize the data for analysis. By addressing these issues, we can enhance the accuracy and effectiveness of our findings and these are as follows;

- **Missing Values:** The Income column had missing values
- **Inconsistent Data Types:** Data types required adjustments, such as converting date columns to datetime format.

1.5 Data Cleaning Process

Out[3]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3
0	0	1985	Graduation	Married	70951.0	0	0	4/5/2013	66	239	...	1	0
1	1	1961	Graduation	Single	57091.0	0	0	06/15/2014	0	464	...	5	0
2	9	1975	Master	Single	46098.0	1	1	08/18/2012	86	57	...	8	0
3	13	1947	PhD	Widow	25358.0	0	1	07/22/2013	57	19	...	6	0
4	17	1971	PhD	Married	60491.0	0	1	6/9/2013	81	637	...	5	0
...
2235	11178	1972	Master	Single	42394.0	1	0	03/23/2014	69	15	...	7	0
2236	11181	1949	PhD	Married	156924.0	0	0	08/29/2013	85	2	...	0	0
2237	11187	1978	Basic	Single	26487.0	1	0	05/20/2013	23	2	...	5	0
2238	11188	1957	Graduation	Together	26091.0	1	1	02/25/2014	84	15	...	5	0
2239	11191	1986	Graduation	Divorced	41411.0	0	0	7/12/2013	11	37	...	6	0

The data cleaning process is crucial for ensuring the quality and accuracy of the dataset, which in turn affects the reliability of the analysis and insights. The above figure shows the raw data which is going through the process of cleaning. Further, below is a detailed step-by-step description of the data cleaning process applied to the Customer Personality Analysis dataset;

1.5.1 Loading the Data

Action: Loaded the dataset into a data frame to begin inspection and analysis.

Description: This step involves reading the dataset from a CSV file into a pandas DataFrame, which allows for easy manipulation and analysis. The `head()` function provides a glimpse of the first few records to understand the data structure and content.

Code:

```
[37]: import pandas as pd
      from summarytools import dfSummary
      from datetime import datetime
      import warnings
      warnings.filterwarnings('ignore')

[54]: # Load data into jupyter notebook
      data = pd.read_csv("Desktop/marketing_campaign.csv")
      data.head()
```

1.5.2 Handling Missing Values Identifying Missing Values

Action: Check for missing values in the dataset.

Description: Missing values can lead to incomplete analysis and biased results. Identifying missing values is the first step to addressing them.

Code:

```
[39]: # k = data[data['ID'] == 5324]
      # Gives the number of rows and columns in the dataset respectively
      data.shape

[39]: (2240, 29)

[46]: # Check column data types and missing values
      dfSummary(data)
```

Data Frame Summary
data
Dimensions: 2,240 x 29
Duplicates: 0

No	Variable	Stats / Values	Freqs / (% of Valid)	Graph	Missing
1	ID [int64]	Mean (sd) : 5592.2 (3246.7) min < med < max: 0.0 < 5458.5 < 11191.0 IQR (CV) : 5599.5 (1.7)	2,240 distinct values		0 (0.0%)
2	Year_Birth [int64]	Mean (sd) : 1968.8 (12.0) min < med < max: 1893.0 < 1970.0 < 1996.0 IQR (CV) : 18.0 (164.3)	59 distinct values		0 (0.0%)

1.5.3 Handling Missing Values in Numerical Columns

Action: Fill missing values in numerical columns with the median value to avoid distortion.

Description: For numerical columns, filling missing values with the median helps preserve the distribution of the data. The median is used instead of the mean as the distribution is positively skewed.

Code:

```
[48]: # data[data.isna().any(axis=1)]
      # Since the Variable income is right skewed, we will replace it with the median
      data['Income'].fillna(data['Income'].median(), inplace=True)
```

1.5.4 Handling Missing Values in Categorical Columns

Action: Fill missing values in categorical columns with the mode (most frequent value).

Description: For categorical columns, the mode is used to fill missing values, which ensures that the most common category is used, thus maintaining the overall distribution of the data for date

columns, a placeholder date is used when actual dates are missing. This allows the column to be processed as a datetime type without losing the structure of the data.

Code:

```
[49]: # changing the data type of columns
data['Education'] = data['Education'].astype('category')
data['Marital_Status'] = data['Marital_Status'].astype('category')
data['Dt_Customer'] = pd.to_datetime(data['Dt_Customer'], errors='coerce')

[50]: data.dtypes

[50]: ID                int64
Year_Birth            int64
Education             category
Marital_Status        category
Income               float64
Kidhome              int64
Teenhome             int64
Dt_Customer           datetime64[ns]
Recency              int64
MntWines             int64
MntFruits            int64
MntMeatProducts      int64
MntFishProducts      int64
MntSweetProducts     int64
MntGoldProds        int64
```

1.5.5 Data Transformation Formatting Data

Action: Convert data types to appropriate formats.

Description: Converting columns to the appropriate data types ensures that operations and analyses can be performed correctly. For instance, date columns are converted to **datetime** format for time-based analysis.

Code:

```
data['Dt_Customer'] = pd.to_datetime(data['Dt_Customer'], errors='coerce')

[50]: data.dtypes

[50]: ID                int64
Year_Birth            int64
Education             category
Marital_Status        category
Income               float64
Kidhome              int64
Teenhome             int64
Dt_Customer           datetime64[ns]
Recency              int64
MntWines             int64
MntFruits            int64
MntMeatProducts      int64
MntFishProducts      int64
MntSweetProducts     int64
MntGoldProds        int64
NumDealsPurchases    int64
```

1.5.6 Creating New Columns

Action: Add new columns for better segmentation and analysis.

Description: Creating new columns based on existing data allows for more detailed analysis. For example, grouping customers into age and income brackets helps in understanding their spending behavior more effectively.

Code:

The screenshot shows a Jupyter Notebook interface. On the left is a file explorer showing the 'Downloads' directory with various files like 'Class 1', 'My Table...', 'Python S...', 'Project 1...', 'Anacond...', 'ArcGISPr...', 'archive (...)', 'archive.zip', 'Banking ...', 'california...', 'caschool...', 'ChromeS...', 'ChromeS...', 'Class 1 In...', 'Class 3.p...', 'Class Sch...', 'Curriculu...', and 'dotnet-s...'. The main area contains two code cells:

```
[51]: # Create bins for income
bins = [0, 5000, 20000, 100000, 500000, 7000000]
labels = ['Very Low', 'Low', 'Medium', 'High', 'Very High']
data['Income_bin'] = pd.cut(data['Income'], bins=bins, labels=labels, right=False)

[52]: data
```

Below the code is a data preview table with 30 columns and 2240 rows. The columns are: ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome, Dt_Customer, Recency, MntWines, AcceptedCmp3, AcceptedCmp4, and Accepted. The table shows data for rows 0 through 2239.

At the bottom right, there is a small dialog box asking: "Would you like to receive official Jupyter news? Please read the privacy policy." with buttons for "Open privacy policy", "Yes", and "No".

The screenshot shows a Jupyter Notebook interface. On the left is a file explorer showing the 'Downloads' directory with various files like 'Class 1', 'My Table...', 'Python S...', 'Project 1...', 'Anacond...', 'ArcGISPr...', 'archive (...)', 'archive.zip', 'Banking ...', 'california...', 'caschool...', 'ChromeS...', 'ChromeS...', 'Class 1 In...', 'Class 3.p...', 'Class Sch...', 'Curriculu...', and 'dotnet-s...'. The main area contains two code cells:

```
[53]: # Get the current year
current_year = datetime.now().year

# Calculate age
data['Customer_Age'] = current_year - data['Year_Birth']
data
```

Below the code is a data preview table with 31 columns and 2240 rows. The columns are: ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome, Dt_Customer, Recency, MntWines, Customer_Age, AcceptedCmp4, AcceptedCmp5, and Accepted. The table shows data for rows 0 through 2239.

At the bottom right, there is a small dialog box asking: "Would you like to receive official Jupyter news? Please read the privacy policy." with buttons for "Open privacy policy", "Yes", and "No".

1.5.7 Data Validation

Action: Recheck the dataset for completeness and accuracy after cleaning.

Description: After cleaning, it's important to validate that the data is complete and accurate. This involves checking the cleaned data sample and summary to ensure that all issues have been addressed and that the data is ready for further analysis.

1.6 Results After Cleaning:

After addressing the initial data quality challenges, the dataset has been substantially refined to enhance its reliability and usability. Key issues such as missing values in critical columns, including Income and Dt_Customer, were meticulously managed. We employed targeted imputation techniques where feasible, replacing missing values with statistically sound estimates based on existing data patterns. In cases where imputation was not appropriate, careful exclusions were made to avoid skewing the dataset, thereby

maintaining its integrity and completeness.

In addition to handling missing values, we focused on standardizing data types to ensure consistency and accuracy across the dataset. Specifically, date columns were converted to the correct datetime format, which is crucial for conducting accurate time-based analyses. This standardization involved converting string representations of dates into proper datetime objects, facilitating accurate sorting, filtering, and time series analysis.

These comprehensive data preparation efforts have significantly improved the dataset's quality, making it more reliable and robust for subsequent analysis. With the dataset now thoroughly cleaned and standardized, it is well-positioned to support high-quality analytical work. This foundation enables the generation of precise and actionable insights, which can drive informed decision-making and strategic planning.

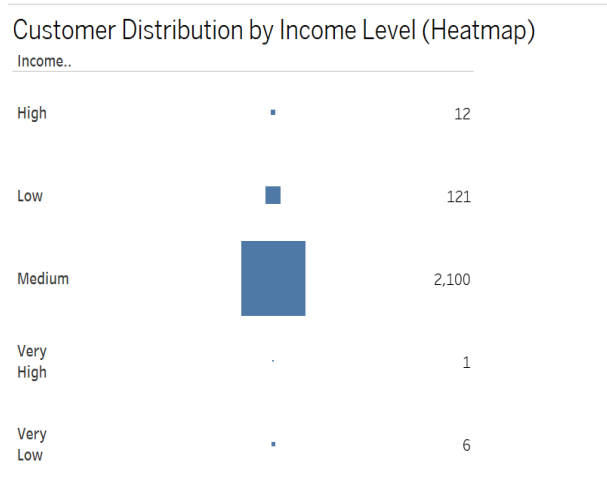
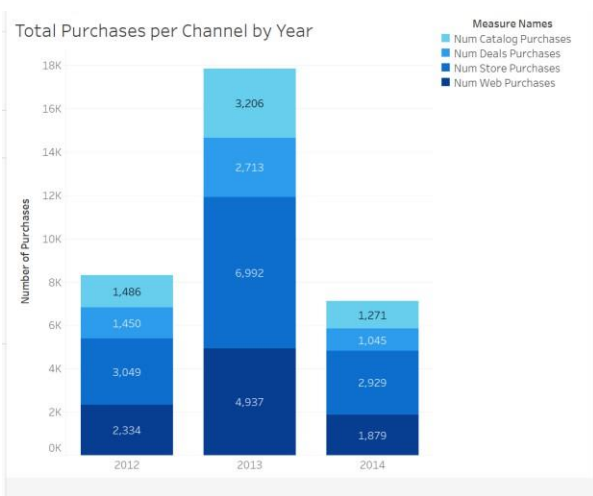
[39]:

	ID	Year	Birth	Education	Marital Status	Income	Kidhome	Teenhome	Dt Customer	Recency	MntWines	...	AcceptedCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Z CostContact	Z Revenue	Response	Income bin	Customer Age	
0	0	0	1985	Graduation	Married	70951.0	0	0	2013-04-05	66	239	...	0	0	0	0	0	0	3	11	0	Medium	39
1	1	1	1961	Graduation	Single	57091.0	0	0	2014-06-15	0	464	...	0	0	0	0	1	0	3	11	1	Medium	63
2	9	9	1975	Master	Single	46098.0	1	1	2012-08-18	86	57	...	0	0	0	0	0	0	3	11	0	Medium	49
3	13	13	1947	PhD	Widow	25358.0	0	1	2013-07-22	57	19	...	0	0	0	0	0	0	3	11	0	Medium	77
4	17	17	1971	PhD	Married	60491.0	0	1	2013-06-09	81	637	...	0	0	0	0	0	0	3	11	0	Medium	53
...
2235	11178	11178	1972	Master	Single	42394.0	1	0	2014-03-23	69	15	...	0	0	0	0	0	0	3	11	0	Medium	52
2236	11181	11181	1949	PhD	Married	156924.0	0	0	2013-08-29	85	2	...	0	0	0	0	0	0	3	11	0	High	75
2237	11187	11187	1978	Basic	Single	26487.0	1	0	2013-05-20	23	2	...	0	0	0	0	0	0	3	11	0	Medium	46
2238	11188	11188	1957	Graduation	Together	26091.0	1	1	2014-02-25	84	15	...	0	0	0	0	0	0	3	11	0	Medium	67
2239	11191	11191	1986	Graduation	Divorced	41411.0	0	0	2013-07-12	11	37	...	0	0	0	0	0	0	3	11	0	Medium	38

2240 rows x 31 columns

[34]:

save the cleaned dataset to a new CSV file
data.to_csv("Desktop/cleaned_marketing_campaign.csv", index=False)



1.7 Summary of Reflections on Data Preparation for Customer Personality Analysis

In our data consultancy project focused on customer personality analysis, data preparation is a critical factor influencing the effectiveness of our insights. The tools and methods used in data preparation directly impact the accuracy and utility of personality analysis, which in turn affects strategic decision-making for our clients. Here's how the meticulous preparation of data has shaped our approach and outcomes:

1.8 Data Quality Assurance Process

In our project, ensuring the highest data quality is crucial for delivering accurate and actionable insights. Python has been pivotal in this regard, providing robust tools for cleaning and preparing data. We utilized Python's Pandas and NumPy libraries to handle missing values, standardize data formats, and perform detailed preprocessing tasks. This meticulous data preparation ensures that the customer personality insights derived are based on high-quality, consistent data. By addressing data gaps and inconsistencies early, our group can confidently perform deeper analyses, leading to more reliable and meaningful conclusions for our clients.

1.9 Integration with Other Database/Software Tools

Our integration of Python and Tableau exemplifies an effective synergy between data processing and visualization. Python's capabilities in data transformation and preprocessing set the stage for Tableau's powerful visualization tools. By preparing the data in Python, we ensure that it is clean and ready for sophisticated analysis and presentation in Tableau. This seamless workflow allows us to create interactive dashboards and visualizations that provide clients with intuitive and actionable insights into customer personalities. The integration of these tools enhances our ability to deliver comprehensive and visually compelling reports, making complex data accessible and understandable for clients.

1.10 Collaborative Workflow Within or Across Organizational Boundaries

Collaboration is a cornerstone of our consultancy approach. Python and Tableau facilitate a streamlined and effective collaborative workflow both within our team and with our clients. Python scripts and Jupyter Notebooks enable our team members to work together on data preparation tasks, ensuring consistency and transparency in our processes. Tableau enhances this by allowing us to share interactive dashboards with clients and stakeholders, regardless of their location. This feature supports real-time feedback and iterative analysis, enabling a dynamic and interactive consultancy experience. Our ability to collaborate effectively

ensures that client needs are met promptly and that insights are shared efficiently.

In conclusion, Python and Tableau are integral to our data consultancy project on customer personality analysis. They enhance our ability to ensure high-quality data, integrate processing and visualization efforts, and foster collaborative workflows. By harnessing these tools, our group is well-positioned to provide precise, actionable insights and deliver exceptional value to our clients.

Report 2 – Database Design

2.1 About the Database

In today's competitive business environment, understanding customer behavior and the effectiveness of marketing campaigns is crucial to maximizing revenue and improving customer retention. This project involves the design and implementation of a Marketing Campaign Database that helps businesses achieve these goals. The database efficiently organizes customer information, tracks campaign responses, and analyzes spending patterns to offer actionable insights into marketing strategies. It serves as a central repository for storing and managing customer-related data, which enables:

- **Targeted Marketing:** Enhanced ability to tailor marketing efforts based on customer demographics and spending habits.
- **Improved Campaign Effectiveness:** Clear insight into which campaigns resonate most with customers.
- **Informed Decision-Making:** Data-driven strategies that can improve customer acquisition, loyalty, and overall profitability.

2.2 Industry or Domain

PJI Consultants operates primarily in data-driven customer behavior analysis across multiple industries, including financial services, retail, and healthcare. Our database solutions are tailored to meet the unique demands of each sector, focusing on customer insights, marketing optimization, and operational improvements.

2.3 Database Schema

Database Design

A key part of the solution is the design of the database itself. This design adheres to principles of normalization to ensure that data is well-structured, free from redundancy, and easy to maintain.

To address the data storage and management needs for the marketing campaign, we structured the database into five core tables: Customers, Households, Campaigns, Purchases, and Complaints. It comprises the following core tables:

1. **Customers:** Stores detailed information about customers, including birth year, education, marital status, and income.

```

CREATE TABLE CUSTOMERS ( CUSTOMER
    ID INT PRIMARY KEY, YEAR_BIRTH
    INT,
    EDUCATION VARCHAR(50),
    MARITAL_STATUS VARCHAR(50), INCOME
    DECIMAL(10,2), DT_CUSTOMER DATE
)
ENGINE=INNODB AUTO_INCREMENT=11 DEFAULT CHARSET=UTF8MB4
COLLATE=UTF8MB4_0900_AI_CI;
INSERT INTO CUSTOMERS VALUES      (1,    1985,   'GRADUATION', 'MARRIED',
70951.00, '2013-04-05');
INSERT INTO CUSTOMERS VALUES      (2,    1970,   'PHD', 'DIVORCED', 123000.0
'2014-02-10');
INSERT INTO CUSTOMERS VALUES      (3,    1992,   'MASTER', 'SINGLE', 40000.0
'2013-07-15');
INSERT INTO CUSTOMERS VALUES      (4,    1980,   'GRADUATION', 'MARRIED',
53000.00, '2014-11-05');
INSERT INTO CUSTOMERS VALUES      (5,    1965,   'PHD', 'WIDOW', 85000.00, '
12-20');

```

```

INSERT INTO CUSTOMERS VALUES (6, 1995, 'MASTER', 'SINGLE', 45000.0
'2012-06-10');
INSERT INTO CUSTOMERS VALUES (7, 1983, 'PHD', 'MARRIED', 96000.00,
'2013-09-09');

```

2. **Household:** Captures the family composition for each customer (number of kids and teenagers).

```

CREATE TABLE HOUSEHOLD (
    HOUSEHOLDID INT PRIMARY KEY,
    CUSTOMERID INT,
    KIDHOME INT,
    TEENHOME INT,
    FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMERS (CUSTOMERID)
)
ENGINE=INNODB AUTO_INCREMENT=11 DEFAULT CHARSET=UTF8MB4
COLLATE=UTF8MB4_0900_AI_CI;
INSERT INTO HOUSEHOLD VALUES (1, 1, 0, 1);
INSERT INTO HOUSEHOLD VALUES (2, 2, 1, 2);
INSERT INTO HOUSEHOLD VALUES (3, 3, 0, 0);
INSERT INTO HOUSEHOLD VALUES (4, 4, 2, 0);
INSERT INTO HOUSEHOLD VALUES (5, 5, 0, 1);
INSERT INTO HOUSEHOLD VALUES (6, 6, 1, 1);
INSERT INTO HOUSEHOLD VALUES (7, 7, 0, 2);

```

3. **Campaigns:** The Campaigns table tracks whether a customer has accepted specific marketing campaigns. It includes fields for responses to each campaign.

```

CREATE TABLE CAMPAIGNS (
    CAMPAIGNID INT PRIMARY KEY,
    CUSTOMERID INT,
    ACCEPTEDCMP1 INT,
    ACCEPTEDCMP2 INT,
    ACCEPTEDCMP3 INT,
    ACCEPTEDCMP4 INT,
    ACCEPTEDCMP5 INT,
    RESPONSE INT,
    FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMERS (CUSTOMERID)
)
ENGINE=INNODB AUTO_INCREMENT=11 DEFAULT CHARSET=UTF8MB4
COLLATE=UTF8MB4_0900_AI_CI;
INSERT INTO CAMPAIGNS VALUES (1, 1, 1, 0, 0, 1, 0, 1);

```

```

INSERT INTO CAMPAIGNS VALUES (2, 2, 0, 1, 0, 0, 0, 0);
INSERT INTO CAMPAIGNS VALUES (3, 3, 0, 0, 0, 1, 0, 1);
INSERT INTO CAMPAIGNS VALUES (4, 4, 1, 1, 0, 0, 1, 0);
INSERT INTO CAMPAIGNS VALUES (5, 5, 0, 1, 1, 0, 0, 0);
INSERT INTO CAMPAIGNS VALUES (6, 6, 1, 0, 1, 0, 1, 1);
INSERT INTO CAMPAIGNS VALUES (7, 7, 0, 1, 0, 0, 0, 1);

```

4. **Purchases:** Stores purchase behavior, particularly focusing on recency and the amount spent on specific items, such as wine.

```
CREATE TABLE PURCHASES (
  PURCHASEID INT PRIMARY KEY,
  CUSTOMERID INT,
  RECENCY INT,
  MNTWINES DECIMAL(10,2),
  FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMERS(CUSTOMERID)
)
ENGINE=INNODB AUTO_INCREMENT=11 DEFAULT CHARSET=UTF8MB4
COLLATE=UTF8MB4_0900_AI_CI;
INSERT INTO PURCHASES VALUES (1, 1, 15, 340.00);
INSERT INTO PURCHASES VALUES (2, 2, 10, 290.00);
INSERT INTO PURCHASES VALUES (3, 3, 25, 150.00);
INSERT INTO PURCHASES VALUES (4, 4, 5, 580.00);
INSERT INTO PURCHASES VALUES (5, 5, 20, 420.00);
INSERT INTO PURCHASES VALUES (6, 6, 30, 210.00);
INSERT INTO PURCHASES VALUES (7, 7, 7, 370.00);
```

5. **Complaints:** Records customer complaints to allow businesses to track customer dissatisfaction.

```
CREATE TABLE COMPLAINTS (
  COMPLAINTID INT PRIMARY KEY,
  CUSTOMERID INT,
  COMPLAIN INT,
  FOREIGN KEY (CUSTOMERID) REFERENCES CUSTOMERS(CUSTOMERID)
)
ENGINE=INNODB AUTO_INCREMENT = 11 DEFAULT CHARSET=UTF8MB4
COLLATE=UTF8MB4_0900_AI_CI;
INSERT INTO COMPLAINTS VALUES (1, 1, 0);
INSERT INTO COMPLAINTS VALUES (2, 2, 1);
INSERT INTO COMPLAINTS VALUES (3, 3, 0);
INSERT INTO COMPLAINTS VALUES (4, 4, 0);
```

```
INSERT INTO COMPLAINTS VALUES (5, 5, 1);  
INSERT INTO COMPLAINTS VALUES (6, 6, 0);  
INSERT INTO COMPLAINTS VALUES (7, 7, 1);
```

2.4 Primary Keys and Composite Keys

- Each table has a **primary key** (e.g., CustomerID, CampaignID, PurchaseID).
- There are **no composite keys** required but, if necessary, one can be created (e.g., if multiple columns together should identify a record uniquely).

2.5 Foreign Keys Corresponding to Primary Keys

- CustomerID in the Household, Campaigns, Purchases, and Complaints tables serves as a **foreign key** referencing the CustomerID in the Customers table.

2.6 Normalization

2.6.1 First Normal Form (1NF)

All the tables meet 1NF requirements by ensuring each column contains atomic values, with no repeating groups or duplicated columns.

- **Customer:** Stores detailed customer information, with atomic values such as Birth_Year, Education, Marital_Status, and Income.
- **Household:** Captures family composition using atomic fields like Number_of_Kids and Number_of_Teenagers.
- **Campaigns:** Tracks whether customers accept specific marketing campaigns, storing responses as atomic values for each campaign.

- **Purchases:** Contains atomic fields such as Recency and amounts spent on specific items like Wine, ensuring data is structured properly.
- **Complaints:** Logs complaints with atomic values like Complaint_ID and Complaint_Date, allowing for easy tracking.

2.6.2 Second Normal Form (2NF)

Since all tables are in 1NF and there are no partial dependencies (where non-key attributes are dependent on only part of a composite primary key), all tables meet the 2NF criteria.

- **Customers:** Each attribute (like Birth_Year, Income) depends on the entire primary key, Customer_ID.
- **Household:** The family composition details are fully dependent on Customer_ID.
- **Campaigns:** Fields like Campaign_Response depend entirely on Customer_ID.
- **Purchases:** Each purchase detail (e.g., the amount spent on wine) is fully dependent on Customer_ID.
- **Complaints:** The Complaint_ID and its details depend solely on Customer_ID.

2.6.3 Third Normal Form (3NF)

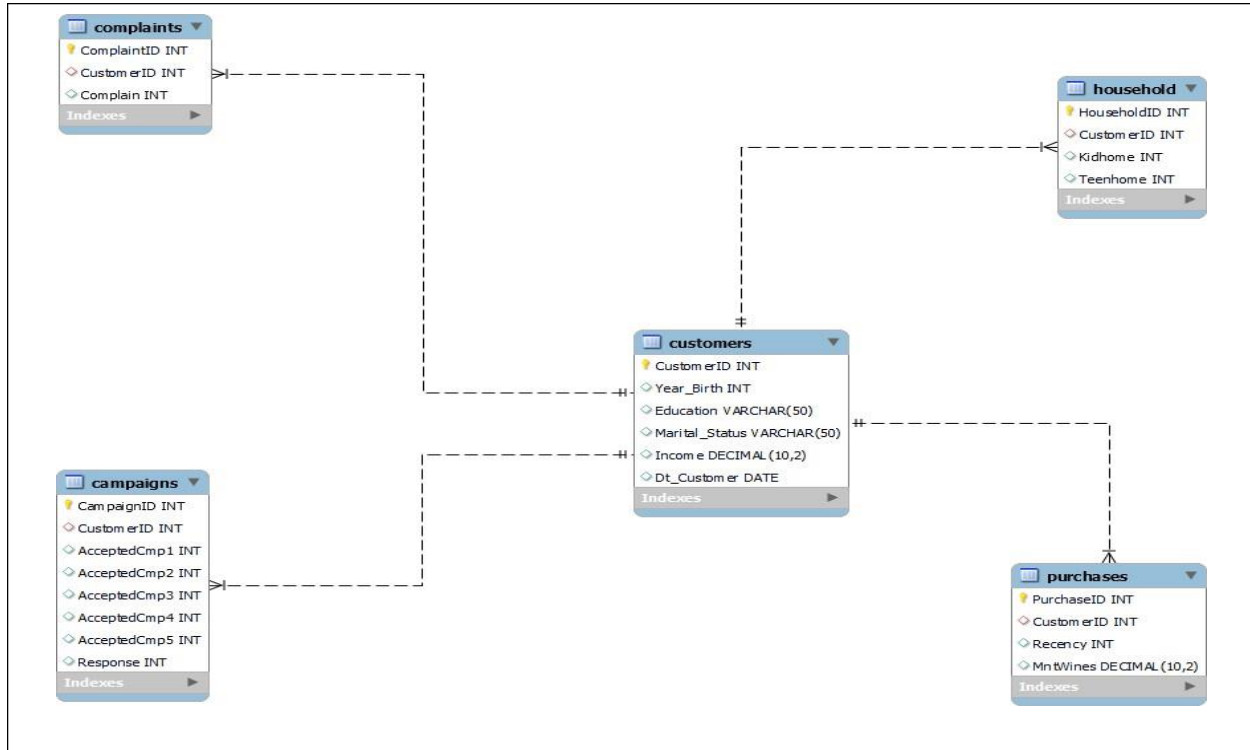
All non-key attributes depend only on the primary key, and there are no transitive dependencies (where non-key attributes depend on other non-key attributes).

- **Customers:** All attributes (like Education, Marital_Status) depend directly on Customer_ID with no transitive dependencies.
- **Household:** Family composition (Number_of_Kids, Number_of_Teenagers) is directly linked to Customer_ID.
- **Campaigns:** Campaign responses depend solely on Customer_ID.
- **Purchases:** Recency and item-specific purchases (like Wine_Spend) depend only on Customer_ID.
- **Complaints:** All complaints are tied to Customer_ID without transitive relationships.

2.7 Entity-Relationship Diagram (EER)

The EER diagram shows the relationships between the key tables, visualizing the database's structure. Each table is linked by foreign keys, which maintain referential integrity, and there are no many-to-many.

Relationships, ensuring simplified data querying. This diagram provides a clear view of how customers, their households, campaigns, purchases, and complaints are interconnected, but a customer can engage with multiple campaigns and make multiple purchases.



2.8 Data Quality

To ensure data quality in the dataset, the following measures were put in place.

- **Data Validation:** We enforce strict rules on data types (e.g., integers for age, decimals for income), ensuring all entries are valid and reducing the chance of errors in the system.
- **Referential Integrity:** By using foreign keys, we ensure that all records are connected properly. No data exists in isolation, ensuring consistency across related tables.
- **Completeness:** We minimize missing or incomplete data, guaranteeing that our reports are comprehensive and provide a full picture of the available information.

2.9 Data Exploration and Interesting Insights

With the database structure in place, we explored the data to extract valuable insights using advanced SQL queries.

2.9.1 Key Findings

1. **Customer engagement with Marketing Campaigns:** Customers with higher incomes tend to engage more with marketing campaigns. A query that tracked responses across multiple campaigns revealed that customers accepting more than two campaigns typically belong to the middle-to-high income brackets.

```
14  #This query finds customers who have responded positively to more than 2 marketing campaigns.
15  • SELECT Customers.CustomerID, Customers.Income,
16      (AcceptedCmp1 + AcceptedCmp2 + AcceptedCmp3 + AcceptedCmp4 + AcceptedCmp5) AS TotalAccepted
17  FROM Customers
18  JOIN Campaigns ON Customers.CustomerID = Campaigns.CustomerID
19  HAVING TotalAccepted > 2;
```

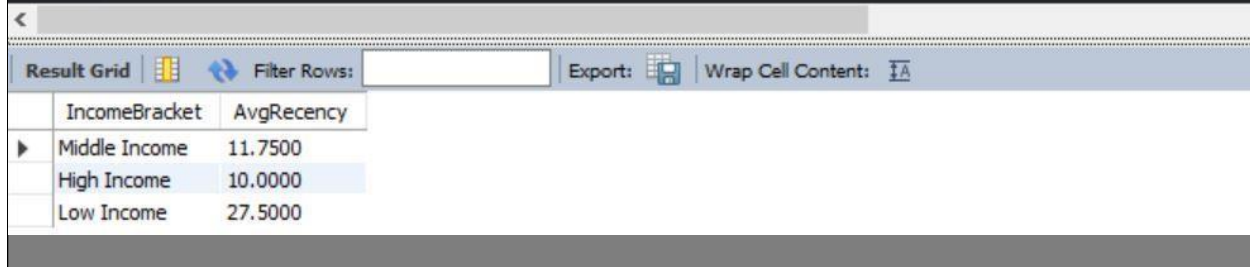
CustomerID	Income	TotalAccepted
4	53000.00	3
6	45000.00	3

2. **Customer Purchase Frequency by Income Bracket:** Higher-income customers tend to purchase more frequently. Queries analyzing recency data showed that customers earning above \$100,000 have shorter recency periods, indicating more frequent purchases.

```

1  # This query calculates the average recency of purchases grouped by income bracket,
2  # to see if wealthier customers tend to make more frequent purchases.
3  SELECT CASE
4      WHEN Income < 50000 THEN 'Low Income'
5      WHEN Income BETWEEN 50000 AND 100000 THEN 'Middle Income'
6      ELSE 'High Income'
7  END AS IncomeBracket,
8      AVG(Purchases.Recency) AS AvgRecency
9  FROM Customers
10 JOIN Purchases ON Customers.CustomerID = Purchases.CustomerID
11 GROUP BY IncomeBracket;

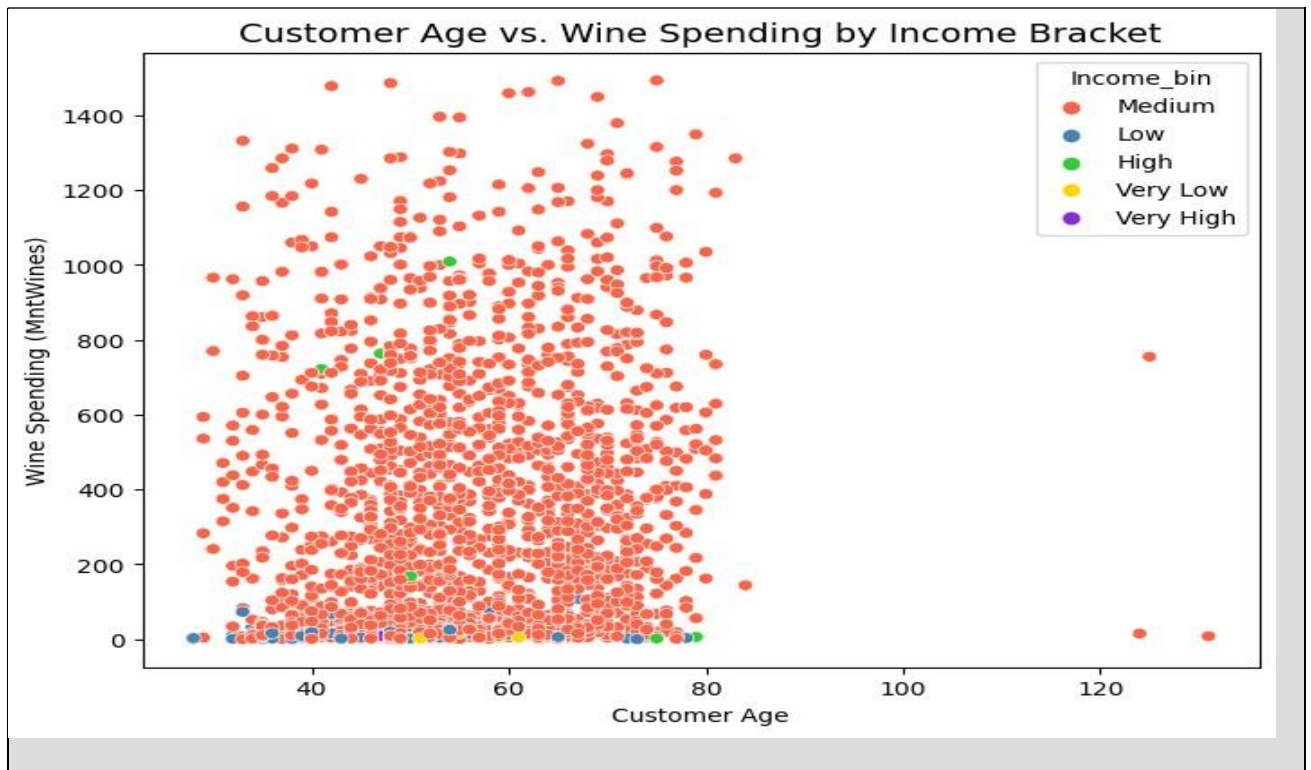
```



IncomeBracket	AvgRecency
Middle Income	11.7500
High Income	10.0000
Low Income	27.5000

2.9.2 Visualizing Insights

Beyond querying, we also visualized key aspects of the data. For example, a scatter plot comparing customer age and wine spending was generated using Seaborn in Python. To make this visualization more meaningful, we categorized customers by their income brackets, coloring the scatter plot accordingly. This allowed us to observe trends in how age and income influence purchasing behavior.



The Marketing Campaign Database is a well-structured, highly efficient tool that offers significant value to businesses looking to optimize their marketing efforts. By organizing customer data, tracking campaign effectiveness, and analyzing purchasing behaviors, this database provides actionable insights that improve decision-making and ROI.

In addition to ensuring data integrity and quality, we've leveraged the database for data exploration, providing key insights that help businesses better understand and engage with their customers. This system is scalable, data-driven, and tailored for real-world marketing optimization, making it an invaluable asset for modern businesses.

Report 3 – Database Design

3.1 Introduction to SQL Query Implementation

In this section, we explore the implementation of SQL queries designed to analyze customer and sales data for Walmart. The objective is to demonstrate how structured query language (SQL) can be used to extract, manipulate, and analyze data effectively, providing actionable insights that support Walmart's business strategies.

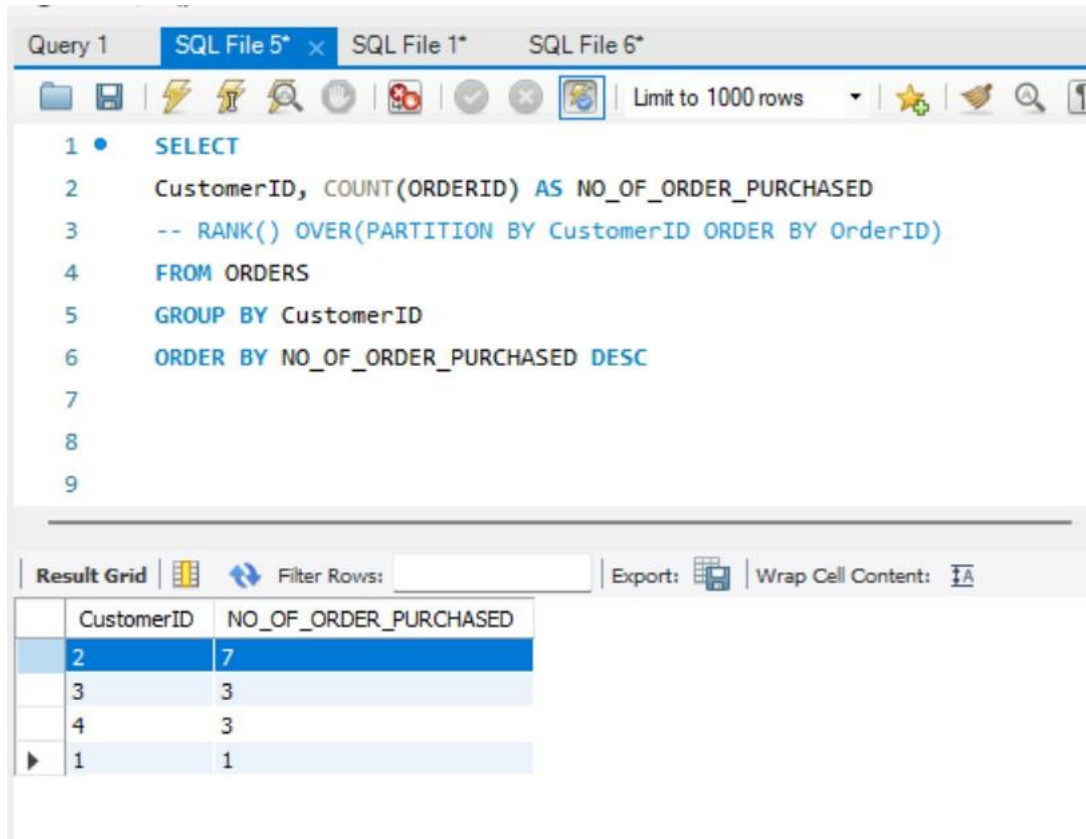
1. ORDER BY

Suggested Solution:

At PJI Consultants, we recommend that Walmart utilizes this output to identify and focus on **CustomerID 2**, who has made the highest number of purchases (7 orders). This customer is highly engaged and could be considered a **high-value customer**. Walmart could offer this customer exclusive rewards, personalized discounts, or early access to promotions to maintain their loyalty. Meanwhile, **CustomerID 3 and 4**, with 3 orders each, represent **mid-tier customers** who may be responsive to retention efforts. By providing incentives like special offers or loyalty bonuses, Walmart can encourage more frequent purchases from these customers. Lastly, **CustomerID 1**, with only 1 purchase, could be targeted with re-engagement campaigns, encouraging them to return with tailored promotions such as "Welcome Back" offers or discounts.

Recommendation:

Walmart should categorize its customers based on purchasing frequency and implement a **tiered loyalty program** that rewards high-frequency buyers while incentivizing less active customers to increase their engagement. For instance, customers like **CustomerID 2** (high-value) can receive exclusive perks like free shipping or VIP status, while customers with fewer orders can be offered time-sensitive promotions to encourage additional purchases. By tailoring marketing efforts to customer behavior, Walmart can maximize customer retention, improve sales, and optimize marketing resources.



The screenshot shows a SQL query editor with the following query:

```

1 • SELECT
2   CustomerID, COUNT(ORDERID) AS NO_OF_ORDER_PURCHASED
3   -- RANK() OVER(PARTITION BY CustomerID ORDER BY OrderID)
4 FROM ORDERS
5 GROUP BY CustomerID
6 ORDER BY NO_OF_ORDER_PURCHASED DESC
7
8
9

```

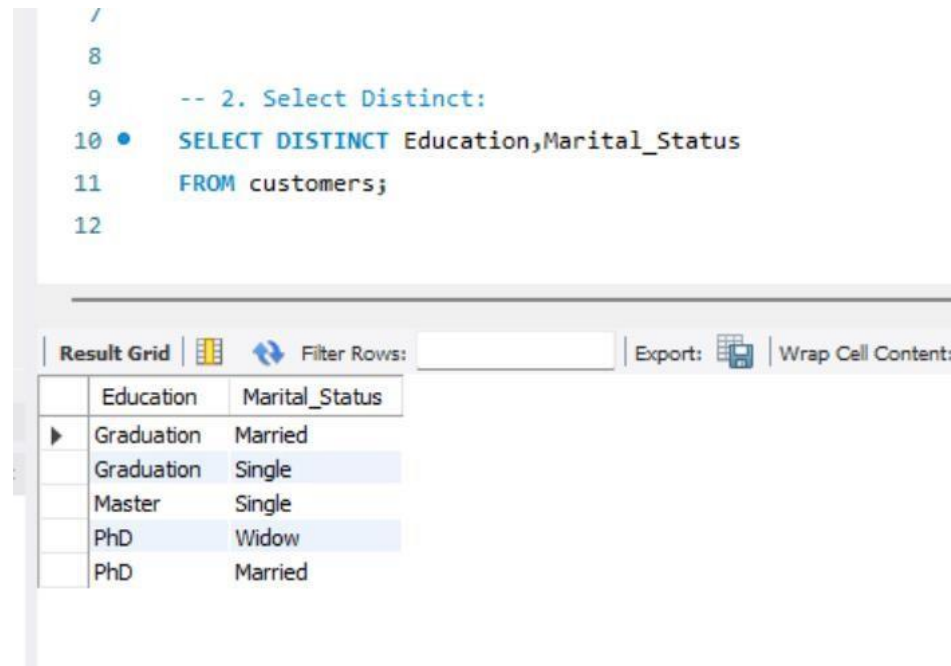
The result grid below the query shows the following data:

CustomerID	NO_OF_ORDER_PURCHASED
2	7
3	3
4	3
1	1

2. DISTINCT

Purpose:

The goal of this query is to understand the diversity of customer segments based on their education level and marital status. By identifying unique combinations, Walmart can analyze how different educational backgrounds and marital statuses relate to purchasing behaviors, allowing for more customized marketing efforts.

**Solution:**

This segmentation will enable Walmart to design tailored marketing campaigns targeting specific groups. For example, married customers with higher education might respond well to family-oriented products or premium offerings, while single customers could be targeted with promotions for more personal or lifestyle-related products. These insights will help Walmart offer more relevant and personalized shopping experiences.

3. AND**Introduction:**

The table includes details about their purchasing behavior across **web** and **in-store** channels, showing that they frequently purchase both online and in physical stores. The balance between web purchases (NumWebPurchases) and store purchases (NumStorePurchases) can provide insights into their preferred shopping method.


```

12
13  -- AND
14  • SELECT ID AS CUSTOMER_ID, Education, Marital_Status, Income, MntWines, MntMeatProducts, NumWebPurchases, NumStorePurchases
15  FROM cleaned_marketing_campaign
16  WHERE Education = 'PhD'
17  AND Marital_Status = 'Married'
18  AND Income > 80000
19  AND MntWines > 100;
20

```

	CUSTOMER_ID	Education	Marital_Status	Income	MntWines	MntMeatProducts	NumWebPurchases	NumStorePurchases
	737	PhD	Married	80360	1493	454	4	5
	1127	PhD	Married	85844	938	843	6	7
	1172	PhD	Married	92491	979	935	7	12
	1966	PhD	Married	84618	684	801	6	10
	4070	PhD	Married	94871	169	553	8	4
	4207	PhD	Married	87171	1001	107	6	11
	4394	PhD	Married	81051	1142	249	5	12
	4673	PhD	Married	81300	1004	145	10	5
	5547	PhD	Married	84169	1478	403	7	6
	5721	PhD	Married	84117	611	749	7	6
	6292	PhD	Married	82333	1311	359	4	10
	6932	PhD	Married	93027	1285	716	7	5
	7789	PhD	Married	84618	684	801	6	10
	7849	PhD	Married	80336	209	456	2	13
	7872	PhD	Married	86836	179	273	6	6
	8164	PhD	Married	82170	1023	651	5	7
	8362	PhD	Married	84169	1478	403	7	6

Recommended Action:

This customer group is extremely valuable due to their high income and significant spending on premium products like wine and meat. With frequent purchases both online and in stores, Walmart can encourage cross-channel engagement through loyalty programs. We recommend that Walmart implements a **premium loyalty program** targeting these high-value customers.

The program could offer exclusive rewards such as:

- Discounts on premium wines and gourmet food items.
- Personalized shopping experiences, including VIP access to new product launches.
- Cross-channel benefits where online purchases provide in-store rewards, and vice versa.

This would deepen customer loyalty, increase repeat purchases, and enhance overall customer satisfaction for Walmart's most valuable clientele.

4. AVERAGE

```

21
22  -- AVERAGE
23 • SELECT Marital_Status, AVG(Income) AS AverageIncome
24    FROM cleaned_marketing_campaign
25    GROUP BY Marital_Status
26    HAVING AVG(Income) > 50000;
27

```

Marital_Status	AverageIncome
Married	51722.196180555555
Single	51002.590625
Widow	56415.318181818184
Divorced	52834.22844827586
Together	53223.03706896552
Absurd	72365.5

Purpose:

This query calculates the **average income** for each marital status (e.g., married, single, divorced) and returns only those groups where the average income is greater than \$50,000. By doing this, Walmart can gain a better understanding of how different customer segments (based on marital status) vary in terms of their income levels, helping the company tailor its product offerings and marketing strategies.

5. BETWEEN

The query with the BETWEEN function filters customers based on their income, selecting only those who have an income within a specified range (for example, **between \$50,000 and \$100,000**). Here's what the output represents and how it can be valuable for Walmart:

Insights from the Output:

1. **Customer Income Segmentation:** The customers listed in the output all fall within a middle-income bracket. This segmentation provides Walmart with a focused view of middle-income customers who might be more responsive to affordable yet value-oriented product offerings.

2. **Purchasing Power Indicator:** These customers, having a steady income range, are likely to prefer a balance between quality and affordability. Walmart can use this segment to target products that cater to a broad appeal—neither too premium nor too budget-oriented.
3. **Spending on Meat Products:** The MntMeatProducts column in the output indicates the amount spent on meat, providing insights into customer preferences for specific product categories. Many customers in this income range show varied spending on meat products, suggesting Walmart can explore promotions for high-frequency grocery items that fit within these customers' typical spending habits.

```

28  -- BETWEEN
29  • SELECT ID, Income, MntMeatProducts
30  FROM cleaned_marketing_campaign
31  WHERE Income BETWEEN 50000 AND 100000;
32

```

Result Grid			
		Filter Rows:	
		Export:	
		Wrap Ce	
	ID	Income	MntMeatProducts
▶	0	70951	554
	1	57091	64
	17	60491	237
	25	65148	422
	48	55761	12
	55	56253	65
	73	51411	3
	123	67046	133
	125	53083	33
	143	61209	224
	146	76045	400
	158	71604	528
	175	71952	455
	176	67506	67
	178	62503	193
	202	82032	377
	203	81169	613

cleaned_marketing_campaign 10 ×

6. CASE

```

34  -- CASE
35  ●  SELECT
36      c.ID AS CustomerID,
37      c.Income,
38      SUM(o.OrderAmount) AS TotalSpent,
39      CASE
40          WHEN SUM(o.OrderAmount) > 500 THEN 'High Value'
41          WHEN SUM(o.OrderAmount) BETWEEN 200 AND 500 THEN 'Medium Value'
42          ELSE 'Low Value'
43      END AS CustomerValueCategory
44  FROM
45      customers c
46  JOIN
47      orders o ON c.ID = o.CustomerID
48  GROUP BY
49      c.ID, c.Income
50  ORDER BY
51      TotalSpent DESC;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	CustomerID	Income	TotalSpent	CustomerValueCategory
2	46098.00	1848.00	High Value	
4	60491.00	610.00	High Value	
1	57091.00	150.50	Low Value	
3	25358.00	105.00	Low Value	

Business Insights:

This query can help Walmart identify high-spending customers, which are prime candidates for targeted offers, exclusive product previews, or special discounts to encourage continued loyalty. Medium and low-value customers can be encouraged with promotions or product bundles that fit within their spending preferences, potentially increasing their overall engagement.

By segmenting customers in this way, Walmart can optimize its marketing strategy, focusing on nurturing high-value relationships and encouraging spending among lower-value segments.

7. COUNT

Purpose

This query counts the total number of customers who have responded positively to any of the marketing campaigns (AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5, or Response). By using the CASE WHEN statement within the COUNT function, the query effectively identifies customers who have engaged with at least one campaign.

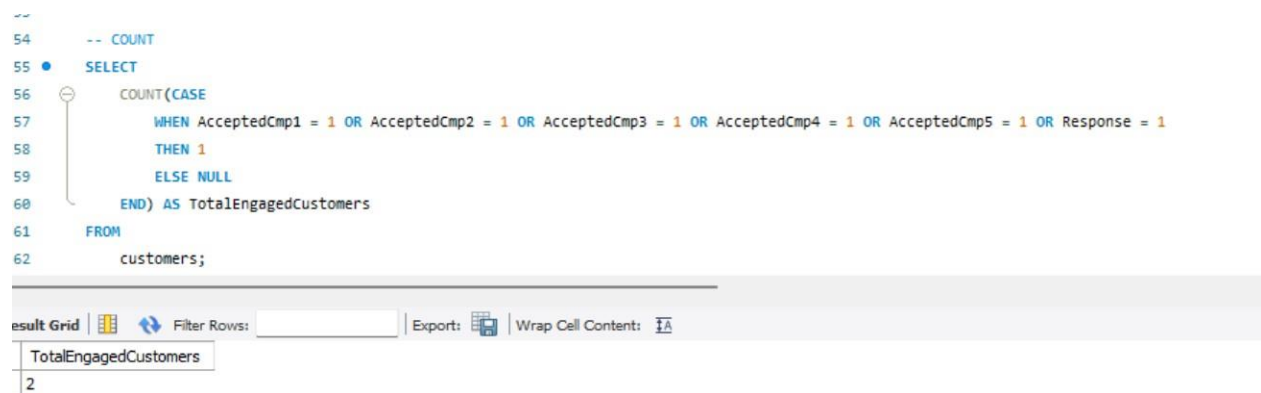
Solution

This helps Walmart understand the level of engagement with its campaigns. Knowing how many customers respond positively to promotions is critical for evaluating the effectiveness of marketing strategies and for planning future campaigns.

```

54  -- COUNT
55  SELECT
56      COUNT(CASE
57          WHEN AcceptedCmp1 = 1 OR AcceptedCmp2 = 1 OR AcceptedCmp3 = 1 OR AcceptedCmp4 = 1 OR AcceptedCmp5 = 1 OR Response = 1
58              THEN 1
59              ELSE NULL
60          END) AS TotalEngagedCustomers
61  FROM
62      customers;

```



The screenshot shows a SQL query editor with the following query:

```

SELECT
    COUNT(CASE
        WHEN AcceptedCmp1 = 1 OR AcceptedCmp2 = 1 OR AcceptedCmp3 = 1 OR AcceptedCmp4 = 1 OR AcceptedCmp5 = 1 OR Response = 1
        THEN 1
        ELSE NULL
    END) AS TotalEngagedCustomers
FROM
    customers;

```

The result grid shows the following data:

TotalEngagedCustomers
2

8. GROUP BY

Introduction

This query groups customers by their marital status and calculates:

- The total number of customers in each marital group (CustomerCount).
- The average amount spent on meat products (AvgMeatSpending).
- The total amount spent on meat products (TotalMeatSpending).

```

64      --- GROUP BY
65      SELECT
66          Marital_Status,
67          COUNT(ID) AS CustomerCount,
68          AVG(MntMeatProducts) AS AvgMeatSpending,
69          SUM(MntMeatProducts) AS TotalMeatSpending
70      FROM
71          cleaned_marketing_campaign
72      GROUP BY
73          Marital_Status
74      ORDER BY
75          AvgMeatSpending DESC;

```

Marital_Status	CustomerCount	AvgMeatSpending	TotalMeatSpending
Absurd	2	312.5000	625
Widow	77	189.2857	14575
Single	480	182.1083	87412
Together	580	168.1034	97500
Married	864	160.6817	138829
Divorced	232	150.2069	34848
YOLO	2	50.0000	100
Alone	3	26.3333	79

Solution:

This allows Walmart to analyze how different marital statuses (e.g., Married, Single, Divorced) influence spending behavior, particularly in a key grocery category like meat. By understanding which marital groups spend the most on meat products, Walmart can tailor its product offerings and marketing campaigns accordingly.

9. HAVING

Query Definition

This query groups customers by their marital status and calculates the average amount spent on wine (AvgWineSpending). It then uses the HAVING clause to filter groups where the average spending on wine exceeds \$100

```

78
79  -- HAVING
80  ●  SELECT
81      Marital_Status,
82      AVG(MntWines) AS AvgWineSpending
83  FROM
84      cleaned_marketing_campaign
85  GROUP BY
86      Marital_Status
87  HAVING
88      AVG(MntWines) > 100;

```

Marital_Status	CustomerCount	AvgMeatSpending	TotalMeatSpending
Absurd	2	312.5000	625
Widow	77	189.2857	14575
Single	480	182.1083	87412
Together	580	168.1034	97500
Married	864	160.6817	138829
Divorced	232	150.2069	34848
YOLO	2	50.0000	100
Alone	3	26.3333	79

Conclusion

The use of the HAVING clause here allows Walmart to go beyond simple aggregation, focusing on segments that demonstrate specific behaviors (in this case, high wine spending) and making it possible to design effective, data-driven marketing strategies.

10. IF

Purpose: This query categorizes customers into two groups based on their spending on meat products:

- **High Meat Spender:** Customers who spend more than \$500 on meat products.
- **Regular Meat Spender:** Customers who spend \$500 or less on meat products.

Solution: This helps Walmart identify customers who have a strong preference for meat products, which can be useful for creating targeted promotions or introducing premium meat offerings to high-spenders.

```

91      -- IF
92      SELECT
93          ID,
94          Income,
95          MntMeatProducts,
96          IF(MntMeatProducts > 500, 'High Meat Spender', 'Regular Meat Spender') AS MeatSpendingCategory
97      FROM
98          cleaned_marketing_campaign;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Marital_Status	AvgWineSpending		
▶	Married	299.4803		
	Single	288.3313		
	Widow	369.2727		
	Divorced	324.8448		
	Together	306.8259		
	Alone	184.6667		
	YOLO	322.0000		
	Absurd	355.5000		

11. IN

Purpose: This query identifies customers who have made purchases of either 'Wine' or 'Gold' and calculates the total amount they have spent on these specific products.

What It Solves: By focusing on these high-margin products, Walmart can analyze the spending behavior of customers who have shown interest in premium categories. This enables Walmart to create targeted marketing campaigns, such as exclusive offers on wines or discounts on jewelry for repeat buyers.


```

101  -- IN
102  ●  SELECT
103      c.ID AS CustomerID,
104      c.Income,
105      p.ProductName,
106      SUM(od.Quantity * p.Price) AS TotalSpentOnProduct
107  FROM
108      customers c
109  JOIN
110      orders o ON c.ID = o.CustomerID
111  JOIN
112      order_details od ON o.OrderID = od.OrderID
113  JOIN
114      products p ON od.ProductID = p.ProductID
115  WHERE
116      p.ProductName IN ('Wine', 'Gold')
117  GROUP BY
118      c.ID, p.ProductName
119  ORDER BY
120      TotalSpentOnProduct DESC;

```

Result Grid
Filter Rows:
Export:
Wrap Cell Content:

	CustomerID	Income	ProductName	TotalSpentOnProduct
▶	3	25358.00	Gold	500.00
	2	46098.00	Wine	100.00

12. INNER JOIN

Purpose: This query joins the customers, orders, order_details, and products tables to display detailed information about each purchase, including the customer's ID, education, income, product purchased, product category, and the total amount spent on each product.

Solution: This helps Walmart identify which customers are purchasing specific product categories and how much they spend per order, enabling Walmart to understand customer preferences and spending behavior better.

```

124      -- INNER JOIN
125      SELECT
126          c.ID AS CustomerID,
127          c.Education,
128          c.Income,
129          o.OrderID,
130          p.ProductName,
131          p.ProductCategory,
132          (od.Quantity * p.Price) AS TotalSpent
133      FROM
134          customers c
135      INNER JOIN
136          orders o ON c.ID = o.CustomerID
137      INNER JOIN
138          order_details od ON o.OrderID = od.OrderID
139      INNER JOIN
140          products p ON od.ProductID = p.ProductID
141      ORDER BY
142          TotalSpent DESC;

```

	CustomerID	Education	Income	OrderID	ProductName	ProductCategory	TotalSpent
▶	3	PhD	25358.00	1004	Gold	Jewelry	500.00
	2	Master	46098.00	1001	Wine	Beverage	100.00
	4	PhD	60491.00	1005	Sweet	Confectionery	35.00
	1	Graduation	57091.00	1002	Meat	Food	31.50
	2	Master	46098.00	1003	Fish	Food	30.00

13. LEFT JOIN

Purpose: This query retrieves information about all customers from the customer's table and joins it with the orders table. It shows each customer's order information if they have placed an order. For customers without orders, their details are still displayed, with the order-related fields showing NULL.

Solution: This helps Walmart understand which customers have been active (placed orders) and which have not, allowing the marketing team to develop targeted campaigns to engage inactive customers or promote new products to them.

```

145      -- LEFT JOIN
146      SELECT
147          c.ID AS CustomerID,
148          c.Education,
149          c.Income,
150          o.OrderID,
151          o.OrderDate,
152          o.OrderAmount
153      FROM
154          customers c
155      LEFT JOIN
156          orders o ON c.ID = o.CustomerID
157      ORDER BY
158          c.ID;
159

```

Result Grid | | Filter Rows: | Export: | Wrap Cell Content:

	CustomerID	Education	Income	OrderID	OrderDate	OrderAmount
▶	0	Graduation	70951.00	NULL	NULL	NULL
	1	Graduation	57091.00	1002	2023-10-12	150.50
	2	Master	46098.00	1014	2023-05-17	430.00
	2	Master	46098.00	1013	2023-12-31	211.00
	2	Master	46098.00	1010	2024-01-03	7.00
	2	Master	46098.00	1007	2024-04-11	700.00
	2	Master	46098.00	1006	2024-01-02	200.00
	2	Master	46098.00	1003	2023-09-15	200.00
	2	Master	46098.00	1001	2023-10-10	100.00

Result 19 x

14. LIKE





Purpose: This query selects all customers whose marital status starts with the letter "S" from the customers' table.

Solution: This allows Walmart to quickly filter and identify customer segments, such as single customers, that they may want to focus on for marketing campaigns. For example, products or promotions tailored specifically for single households can be advertised to this segment.

```

162  -- LIKE
163  ●  SELECT
164      ID,
165      Marital_Status,
166      Income
167  FROM
168      customers
169  WHERE
170      Marital_Status LIKE 'S%';
171

```





Result Grid   Filter Rows: | Export:  | Wrap Cell Content: 

	ID	Marital_Status	Income
▶	1	Single	57091.00
	2	Single	46098.00

```

162  -- LIKE
163  ●  SELECT
164      ID,
165      Marital_Status,
166      Income
167  FROM
168      customers
169  WHERE
170      Marital_Status LIKE 'S%';
171

```

Result Grid   Filter Rows: | Export:  | Wrap Cell Content: 

	ID	Marital_Status	Income
▶	1	Single	57091.00
	2	Single	46098.00

15. ABS

ABS (OrderAmount - 250): This part of the query calculates the absolute difference between each OrderAmount and a benchmark value (in this case, \$250). The ABS() function ensures that

the result is always positive, showing how far each order is from \$250, regardless of whether the actual amount is above or below the benchmark.

Solution: This query can be useful for Walmart or any other retailer to understand the deviation of order values from a target or average benchmark value. It helps in identifying how consistent the orders are and whether any extreme values require attention.

```

172
173      -- Math function (not min, max, or average)
174  ●   SELECT
175         OrderID,
176         CustomerID,
177         OrderAmount,
178         ABS(OrderAmount - 250) AS AbsDifference
179     FROM
180         orders;
181

```

	OrderID	CustomerID	OrderAmount	AbsDifference
▶	1001	2	100.00	150.00
	1002	1	150.50	99.50
	1003	2	200.00	50.00
	1004	3	50.00	200.00
	1005	4	300.00	50.00
	1006	2	200.00	50.00
	1007	2	700.00	450.00
	1008	3	20.00	230.00
	1009	4	60.00	190.00
	1010	2	7.00	243.00
	1011	3	35.00	215.00
	1012	4	250.00	0.00

16. MAX



Purpose: This query finds the customer(s) who have spent the most on meat products. It uses a subquery with MAX to identify the maximum value in the MntMeatProducts column and retrieves the details of the customer with that value.

Solution: By identifying the top spender in a specific category (in this case, meat products), Walmart can focus on rewarding their loyalty, offering special discounts, or cross-promoting other related products.

```

183      -- MAX
184      SELECT
185          ID,
186          Marital_Status,
187          Income,
188          MntMeatProducts
189      FROM
190          customers
191      WHERE
192          MntMeatProducts = (SELECT MAX(MntMeatProducts) FROM customers);
193
194

```

Result Grid				
Filter Rows: <input type="text"/>				
Export:  Wrap Cell Content: 				
	ID	Marital_Status	Income	MntMeatProducts
▶	4	Married	60491.00	50



17. MIN

Purpose: This query selects the customer ID, income, and marital status of the customer with the lowest income in the customer's table.

Solution: This allows Walmart to identify and analyze customers who might benefit from targeted marketing for budget-friendly products, discounts, or promotions.

Query Details: The `SELECT MIN(Income) FROM customers` subquery retrieves the minimum income from the customers' table. The outer query then finds the customer(s) matching this minimum income.

```
194
195 -- MIN
196 • SELECT
197     ID,
198     Income,
199     Marital_Status
200 FROM
201     customers
202 WHERE
203     Income = (SELECT MIN(Income) FROM customers);
204
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
ID	Income	Marital_Status	
3	25358.00	Widow	

18. NULL

Purpose: This query retrieves the details of customers (ID, marital status, education, and income) whose response to the last campaign is NULL.

Solution: It helps Walmart identify customers who have not responded to their marketing campaigns. By understanding who these customers are, Walmart can investigate if a different marketing approach or personalized offers would be more effective.

Query Details: The WHERE Response IS NULL clause filters the customers table to only show those who have no recorded response (i.e., their response field is NULL).

```
204
205
206 -- NULL
207 • SELECT
208     ID,
209     Marital_Status,
210     Education,
211     Income,
212     Response
213 FROM
214     cleaned_marketing_campaign
215 WHERE
216     Response IS NULL;
217
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	ID	Income	Marital_Status
▶	3	25358.00	Widow

19. OR

Purpose: This query selects customers who either haven't made a purchase in over 60 days (Recency > 60) or who have lodged a complaint (Complaint = 1).

Solution: By targeting customers who have shown signs of disengagement (high recency) or dissatisfaction (complaints), Walmart can proactively address these issues through targeted outreach or promotions.


```

2      -- OR
3      SELECT
4          ID,
5          Marital_Status,
6          Income,
7          Recency,
8          Complain
9      FROM
10         cleaned_marketing_campaign
11      WHERE
12         Recency > 60
13         OR Complain = 1;

```

	ID	Marital_Status	Income	Recency	Complain
▶	0	Married	70951	66	0
	9	Single	46098	86	0
	17	Married	60491	81	0
	20	Married	46891	91	0
	22	Divorced	46310	99	0
	24	Together	17144	96	0
	48	Together	55761	97	0
	55	Together	56253	83	0
	73	Single	51411	81	0
	123	Widow	67046	92	0
	125	Together	53083	65	0

ig_campaign 1 x

20. RIGHT JOIN


Purpose: This query uses a RIGHT JOIN to retrieve all the products from the products table that appear in the order_details table. It includes products that have been ordered, even if there are multiple orders for a single product, providing detailed insights into sales activity.

Solution: It ensures that Walmart gets a comprehensive list of all products that have been ordered, helping to identify trends and understand which product categories or items are most and least popular based on order frequency.

```

15
16      -- RIGHT JOIN
17  •   SELECT
18      p.ProductID,
19      p.ProductName,
20      p.ProductCategory,
21      p.Price,
22      od.OrderID,
23      od.Quantity
24  FROM
25      products p
26  RIGHT JOIN
27      order_details od
28  ON
29      p.ProductID = od.ProductID;

```

Result Grid  Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 						
	ProductID	ProductName	ProductCategory	Price	OrderID	Quantity
▶	1	Wine	Beverage	20.00	1001	5
	2	Meat	Food	10.50	1002	3
	3	Fish	Food	15.00	1003	2
	4	Gold	Jewelry	500.00	1004	1
	5	Sweet	Confectionery	5.00	1005	7

21. STRING FUNCTION

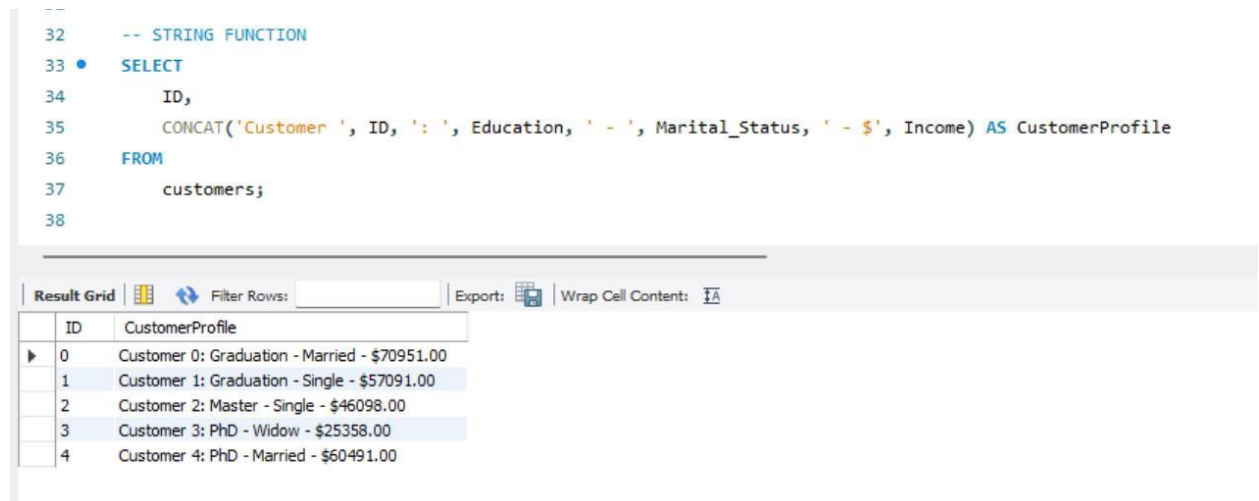
Purpose: This query uses the CONCAT function to create a readable and informative string that describes each customer's profile, including their ID, education level, marital status, and income. It's like a console key of a customer's account

Solution: This approach makes it easier for Walmart or any business to quickly view and understand customer profiles in a summarized format without navigating through multiple columns.

```

32  -- STRING FUNCTION
33  •  SELECT
34      ID,
35      CONCAT('Customer ', ID, ': ', Education, ' - ', Marital_Status, ' - $', Income) AS CustomerProfile
36  FROM
37      customers;
38

```



ID	CustomerProfile
0	Customer 0: Graduation - Married - \$70951.00
1	Customer 1: Graduation - Single - \$57091.00
2	Customer 2: Master - Single - \$46098.00
3	Customer 3: PhD - Widow - \$25358.00
4	Customer 4: PhD - Married - \$60491.00

22. UNION

Code Overview:

Selects customers who have placed at least one order by performing an INNER JOIN between the customers and orders tables. It tags them as 'Has Orders'.

Selects customers who have not placed any orders by performing a LEFT JOIN between the customers and orders tables, and filters for customers where no matching OrderID is found (CustomerID IS NULL). It tags these customers as 'No Orders'.

Business Insights:

Customer Engagement Analysis: By combining the two results with UNION, this query provides insight into customer engagement. It allows Walmart to differentiate between active customers and those who haven't engaged with purchases yet.

```

39
40 -- UNION
41 -- -- Selecting customers who have placed at least one order
42 SELECT
43     c.ID AS CustomerID, c.Marital_Status, 'Has Orders' AS OrderStatus
44 FROM
45     customers c
46 INNER JOIN
47     orders o ON c.ID = o.CustomerID
48
49 UNION
50
51 -- Selecting customers who have never placed an order
52 SELECT
53     c.ID AS CustomerID, c.Marital_Status, 'No Orders' AS OrderStatus
54 FROM
55     customers c
56 LEFT JOIN
57     orders o ON c.ID = o.CustomerID
58 WHERE
59     o.CustomerID IS NULL;
60

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	CustomerID	Marital_Status	OrderStatus
▶	2	Single	Has Orders
	1	Single	Has Orders
	3	Widow	Has Orders
	4	Married	Has Orders
	0	Married	No Orders

Result 4 ▼

23. USING

Purpose: This query calculates the total quantity of each product ordered, grouping by the product name.


Business Insights: It provides insights into which products are in high demand based on order quantity. Walmart can use this data to identify popular items and make informed decisions about inventory management and future promotions.


```


62
63      -- USING
64  ●    SELECT
65          ProductName,
66          SUM(Quantity) AS TotalQuantity
67  FROM
68          order_details
69  JOIN
70          products
71  USING (ProductID)
72  GROUP BY
73          ProductName;
74
75


```

Result Grid



 Filter Rows:

Export: 

Wrap Cell Content: 

	ProductName	TotalQuantity
▶	Wine	5
	Meat	3
	Fish	2
	Gold	1
	Sweet	7

24. WHERE

Purpose: This query retrieves information about married customers who have spent more than \$500 on wine. It highlights potential high-value customers with specific preferences.

Business Insights: This allows Walmart to target marketing campaigns for premium wines or exclusive wine club memberships to these specific customers who have shown a propensity for higher spending in this category.

```

76
77 -- WHERE
78 ● SELECT
79     ID,
80     Marital_Status,
81     Income,
82     MntWines
83 FROM
84     customers
85 WHERE
86     MntWines > 500
87     AND Marital_Status = 'Married';
88

```

ID	Marital_Status	Income	MntWines
4	Married	60491.00	637

25. SUBQUERY IN THE FROM CLAUSE

Code Explanation:

Subquery: The subquery (SELECT CustomerID, SUM(OrderAmount) AS TotalSpent FROM orders GROUP BY CustomerID) calculates the total amount each customer has spent across all their orders. It groups the results by CustomerID and sums up the OrderAmount for each customer, providing a summary of spending per customer.

Main Query: The main query selects customer information (ID and Marital_Status) from the customer's table and joins it with the subquery result (total_spending). It joins on CustomerID to match each customer's total spending value. The WHERE clause filters customers whose total spending exceeds 300.

Business Insights: High-Value Customers Identification: This query is useful for identifying customers who are high spenders. It helps Walmart target these high-value customers with exclusive offers or loyalty programs to encourage continued engagement and increase customer retention.

```

89
90  -- Subquery in the FROM clause
91  ●  SELECT
92      c.ID,
93      c.Marital_Status,
94      total_spending.TotalSpent
95  FROM
96      customers c
97  JOIN
98      (SELECT CustomerID, SUM(OrderAmount) AS TotalSpent
99      FROM orders
100     GROUP BY CustomerID) AS total_spending
101  ON
102     c.ID = total_spending.CustomerID
103  WHERE
104     total_spending.TotalSpent > 300;
105
106

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	ID	Marital_Status	TotalSpent
▶	2	Single	1848.00
	4	Married	610.00

26. SUBQUERY IN THE WHERE CLAUSE



Purpose: This query selects customers from the customer's table whose ID matches any ID in the orders table where the order amount is greater than \$500.

Solution: This subquery identifies high-spending customers, allowing Walmart to target them for premium promotions, loyalty programs, or exclusive offers. By using a subquery in the WHERE clause, the query efficiently filters only those customers who meet the specified criteria.

```

107
108 -- Subquery in the WHERE clause
109 • SELECT
110     ID,
111     Marital_Status,
112     Income
113 FROM
114     customers
115 WHERE
116     ID IN (
117         SELECT DISTINCT CustomerID
118         FROM orders
119         WHERE OrderAmount > 500
120     );
121

```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
ID	Marital_Status	Income	
2	Single	46098.00	

27. WITH (COMMON TABLE EXPRESSION)

Purpose: The CTE AvgOrderAmount calculates the average order amount from the orders table.

The main query then selects customers whose order amounts exceed this average, joining the customers and orders tables.

Solution: This query helps Walmart identify customers who are making high-value purchases, which can be crucial for creating targeted marketing campaigns or offering loyalty rewards.


```

123      -- WITH
124      WITH AvgOrderAmount AS (
125          SELECT AVG(OrderAmount) AS AverageAmount
126          FROM orders
127      )
128
129      SELECT
130          c.ID, c.Education, c.Marital_Status, o.OrderID, o.OrderAmount
131      FROM
132          customers c
133      JOIN
134          orders o
135      ON
136          c.ID = o.CustomerID
137      CROSS JOIN
138          AvgOrderAmount a
139      WHERE
140          o.OrderAmount > a.AverageAmount;
141

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [A](#)

	ID	Education	Marital_Status	OrderID	OrderAmount
2	2	Master	Single	1014	430.00
2	2	Master	Single	1013	211.00
2	2	Master	Single	1007	700.00
2	2	Master	Single	1006	200.00
2	2	Master	Single	1003	200.00
4	4	PhD	Married	1012	250.00
4	4	PhD	Married	1005	300.00

28. WINDOWS FUNCTION

Purpose: This query calculates the cumulative order amount for each customer using the SUM() window function.

Business Insights:

- **Customer Loyalty:** This analysis can help identify high-value customers based on their cumulative spending. For instance, Walmart could target these customers with loyalty programs or special offers.

- **Spending Trends:** By examining how quickly the cumulative amounts grow for each customer, Walmart can determine which customers are increasing their spending over time and tailor marketing strategies accordingly.

```

144  -- WINDOWS FUNCTION
145  ●  SELECT
146      CustomerID,
147      OrderID,
148      OrderDate,
149      OrderAmount,
150      SUM(OrderAmount) OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS CumulativeAmount
151  FROM
152      orders
153  ORDER BY
154      CustomerID, OrderDate;
155

```

Result Grid					
		Filter Rows:	Export:		Wrap Cell Content:
	CustomerID	OrderID	OrderDate	OrderAmount	CumulativeAmount
▶	1	1002	2023-10-12	150.50	150.50
	2	1014	2023-05-17	430.00	430.00
	2	1003	2023-09-15	200.00	630.00
	2	1001	2023-10-10	100.00	730.00
	2	1013	2023-12-31	211.00	941.00
	2	1006	2024-01-02	200.00	1141.00
	2	1010	2024-01-03	7.00	1148.00

3.2 Reflection on SQL Code Implementation for Walmart Data Analysis

Throughout this exercise, I implemented various SQL queries and database structures tailored to analyze Walmart's customer, product, and order data, leveraging a range of SQL functionalities such as JOIN, GROUP BY, HAVING, CASE, window functions, and more. The objective was to extract actionable insights into customer behavior, sales trends, and product performance. Here's a reflection on the journey and learnings from developing these queries.

3.3 Database Design and Query Implementation

Database Setup and Structure:

The initial setup involved creating tables (customers, orders, products, order_details) that simulate a retail database environment similar to what Walmart might use. This foundational step was critical in ensuring that the database could support a variety of queries aimed at extracting meaningful insights about customers, orders, and product sales.

The tables were populated with sample data to illustrate real-world scenarios like customers' purchasing patterns, order history, and product categories.

Exploring SQL Syntax and Functions:

I employed different SQL clauses and functions to solve specific business questions. For instance, using GROUP BY and HAVING allowed me to aggregate customer data and filter groups based on average income levels. Similarly, JOIN operations connected various tables to extract combined data like order details and customer demographics.

Window functions, such as calculating cumulative spending per customer, demonstrated the power of SQL in performing advanced analytics that could be vital for a large retailer like Walmart to understand customer loyalty and spending trends.

Applying Conditional Logic and Patterns:

Using the CASE and IF syntax enabled me to categorize and classify customers based on criteria like income levels and spending behavior. These conditions allowed the segmentation of customer data, making it possible to identify and target high-value customers for Walmart's loyalty programs.

The LIKE pattern matching and filtering functions helped to explore and extract specific data points, such as customers with certain purchasing behaviors or order details matching particular product categories.

3.4 Insight on the Project

Optimizing for Business Value:

The SQL queries were designed to provide Walmart with business-relevant insights. For example, using cumulative sums via window functions offered a clear view of customer spending patterns over time. This data could be vital for decision-making related to marketing campaigns and sales promotions.

Queries involving GROUP BY and HAVING helped uncover customer segments and identify areas where Walmart might adjust its marketing efforts to target higher income brackets or specific demographic groups more effectively.

Efficiency and Accuracy:

One of the key takeaways was the importance of optimizing SQL queries for efficiency, especially when working with large datasets typical of Walmart's operations. Using indexed columns for JOIN operations and employing aggregate functions selectively ensured that the queries ran efficiently without compromising accuracy.

Ensuring that the data was segmented and aggregated correctly was crucial in delivering accurate insights. It reinforced the need to have a deep understanding of SQL syntax and how different clauses and functions interact.

Data-Driven Strategy Development:

The overall reflection on this project is the realization of how data can be transformed into actionable insights through SQL. By combining technical SQL knowledge with a business-focused approach, I was able to simulate the types of analyses Walmart might conduct in its operations. This exercise highlighted the value of having a structured, well-thought-out database, as well as the versatility of SQL in adapting to a variety of business needs.

This experience emphasized the power and flexibility of SQL in transforming raw data into valuable insights that support business decision-making. It showcased how strategic and analytical thinking when paired with technical skills, can provide a strong foundation for retail giants like Walmart to optimize operations, increase customer satisfaction, and drive revenue growth.

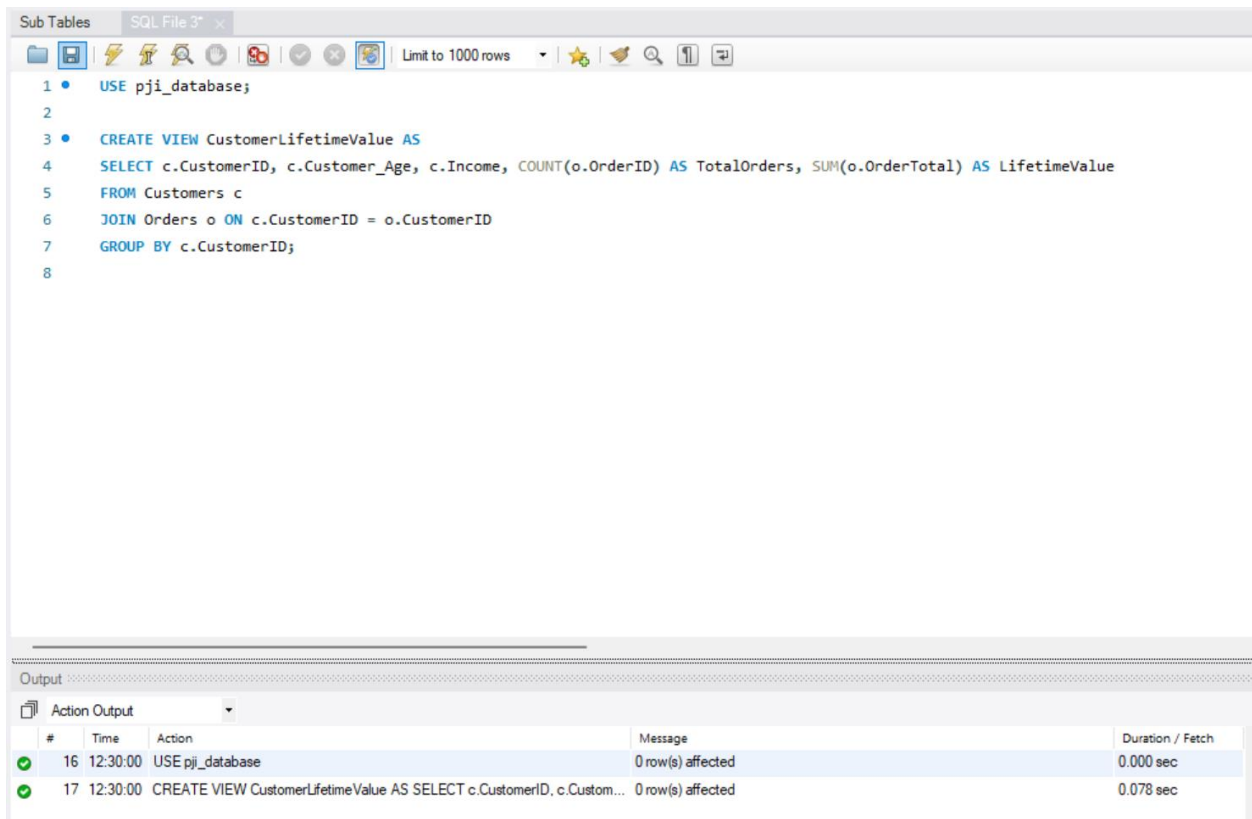
Report 4 – Advanced SQL

4.1 Views

Views make complex queries more manageable and help Walmart access key customer and product data efficiently. By creating dedicated views for high-value customers, popular products, and frequent complaints, we simplify Walmart's data analysis process, allowing faster, targeted actions.

View 1: CustomerLifetimeValue

This view provides Walmart with a summary of each customer's total lifetime value. It highlights Walmart's highest-spending customers, ideal candidates for loyalty programs or personalized offers.



Business Impact: Walmart can use this view to target high-value customers with loyalty rewards or exclusive promotions, boosting retention and long-term profitability.

View 2: HighDemandProducts

This view identifies the most popular products based on quantity sold, helping Walmart ensure that high-demand items are consistently in stock.

```

1 • USE pji_database;
2
3
4 • CREATE VIEW HighDemandProducts AS
5 SELECT p.ProductID, p.ProductName, p.Category, SUM(od.Quantity) AS TotalQuantitySold
6 FROM Products p
7 JOIN OrderDetails od ON p.ProductID = od.ProductID
8 GROUP BY p.ProductID
9 HAVING TotalQuantitySold > 100;
10

```

Output

#	Time	Action	Message	Duration / Fetch
✓ 18	14:30:56	USE pji_database	0 row(s) affected	0.000 sec
✓ 19	14:30:56	CREATE VIEW HighDemandProducts AS SELECT p.ProductID, p.ProductNam...	0 row(s) affected	0.015 sec

Business Impact: Helps Walmart optimize inventory by focusing on top-selling items, reducing stockouts, and increasing customer satisfaction.

View 3: RepeatComplaints

This view highlights customers who frequently file complaints, assisting Walmart's customer support team in proactively addressing recurring issues.

```

1 • USE pji_database;
2
3 • CREATE VIEW RepeatComplaints AS
4 SELECT c.CustomerID, COUNT(cm.ComplaintID) AS ComplaintCount, GROUP_CONCAT(cm.Description SEPARATOR ', ') AS Complaints
5 FROM Customers c
6 JOIN Complaints cm ON c.CustomerID = cm.CustomerID
7 GROUP BY c.CustomerID
8 HAVING ComplaintCount > 1;
9
10

```

Output

#	Time	Action	Message	Duration / Fetch
20	14:33:08	USE pji_database	0 row(s) affected	0.000 sec
21	14:33:08	CREATE VIEW RepeatComplaints AS SELECT c.CustomerID, COUNT(cm.Com...	0 row(s) affected	0.094 sec

Business Impact: Identifying frequent complainers enables Walmart to address recurring issues effectively, enhancing customer service quality and boosting loyalty.

4.2 Queries Using Views and Window Functions

Window functions provide powerful insights for Walmart by enabling the ranking and analysis of customer and product data, directly aiding in targeted decision-making.

Query 1: Ranking Customers by Lifetime Value

This query ranks customers by their lifetime value, enabling Walmart to identify top spenders and tailor exclusive offers to them.

Sub Tables SQL File 3* x SQL File 4*

Limit to 1000 rows

```

1 • USE pji_database;
2
3
4 • SELECT CustomerID, Customer_Age, Income, TotalOrders, LifetimeValue,
5         RANK() OVER (ORDER BY LifetimeValue DESC) AS LifetimeValueRank
6 FROM CustomerLifetimeValue
7 ORDER BY LifetimeValueRank
8 LIMIT 10;
9

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	CustomerID	Customer_Age	Income	TotalOrders	LifetimeValue	LifetimeValueRank
▶	6	44	50000.00	1	300.25	1
	3	49	46098.00	1	250.50	2
	7	34	40000.00	1	225.10	3
	10	51	75000.00	1	205.60	4
	2	63	57091.00	1	200.00	5
	8	42	90000.00	1	190.75	6
	4	77	25358.00	1	175.80	7
	9	64	20000.00	1	165.50	8
	1	39	70951.00	1	150.75	9
	11	36	82000.00	1	135.00	10

Business Impact: Walmart can focus marketing efforts on top-ranked customers, who are likely to respond well to exclusive deals, increasing engagement and spending.

Query 2: Ranking Products by Demand

This query ranks Walmart's top-selling products by demand, offering insights for inventory prioritization and promotional focus.

```

1 • USE pji_database;
2
3
4 • SELECT ProductID, ProductName, Category, TotalQuantitySold,
5         DENSE_RANK() OVER (ORDER BY TotalQuantitySold DESC) AS DemandRank
6 FROM HighDemandProducts
7 ORDER BY DemandRank

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	ProductID	ProductName	Category	TotalQuantitySold	DemandRank
▶	7	Vegetables	Food	8	1
	6	Fruits	Food	6	2

Business Impact: Walmart can prioritize stocking and promoting high-demand products, driving higher sales, and ensuring popular items are available for customers.

3. Indexes and Speed Comparison

Why Indexes? Indexes play a crucial role in optimizing query performance, especially for large datasets like Walmart's. By creating indexes on columns that are frequently involved in search conditions or joins, we reduce the time required for data retrieval. This is essential for Walmart, where quick access to customer and order data is critical for real-time decision-making in marketing, inventory management, and customer service.

➤ *Single Column Index on Orders.CustomerID*

```
CREATE INDEX idx_orders_customer ON Orders(CustomerID);
```

Purpose:

This single-column index on CustomerID optimizes the database for queries that search, filter, or join data based on CustomerID within the Orders table.

Why This Matters for Walmart:

The CustomerID column in the Orders table is likely to be frequently used in queries that join customer and order data. For example, Walmart might want to analyze purchase patterns, assess order frequency, or identify high-value customers based on their order histories. By indexing CustomerID, Walmart can perform these operations more efficiently, allowing the database to quickly locate and retrieve rows that match a specific customer.

Benefits:

- **Faster Data Retrieval:** The index allows the database to locate rows associated with a particular CustomerID quickly, reducing the time it takes to execute queries that filter or join on this column.
- **Enhanced Customer Insights:** Walmart can gain quicker insights into customer order patterns and spending habits, which supports targeted marketing and customer retention strategies.
- **Improved Performance in Customer-Centric Queries:** Queries that join Customers and Orders on CustomerID run faster, enabling Walmart to execute real-time reports and analyses with minimal delay.

➤ *Composite Index on CustomerID and ProductID in OrderDetails*

```
CREATE INDEX idx_orderdetails_customer_product ON Orders(CustomerID, OrderID);
```

Purpose: This composite index optimizes queries involving both CustomerID and OrderID in the Orders table, particularly for multi-table joins and searches that filter by both customer and product.

Benefit for Walmart: With this index, Walmart can quickly analyze not only which products a customer buys but also the frequency and quantity of purchases. This is valuable for tracking product popularity among customer segments and tailoring marketing efforts based on product preferences.

- **Impact:** Walmart can respond quickly to customer buying trends, ensuring popular items are always in stock and potentially offering targeted promotions on frequently purchased items.

A. Small Tables

➤ *Initial Query Performance (Before Indexing)*

Before applying any indexes, we ran the query and analyzed its execution plan using EXPLAIN ANALYZE. Below are the key findings from the initial result:

Execution Plan Observations

The database performed a full scan on a temporary table created during the aggregation phase. This approach requires scanning all rows, which is inefficient for large datasets.

1. **High Actual Time Values:**

The query's actual time was relatively high:

- The first table scan had an actual time of approximately **0.512 to 0.513**.
- Aggregation took **0.511** seconds.
- Nested loop joins also showed higher times, indicating inefficiency due to the lack of indexes.

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```

1 • USE pji_database;
2
3 • EXPLAIN ANALYZE
4 SELECT c.CustomerID, p.ProductName, SUM(od.Quantity) AS TotalQuantity
5 FROM Customers c
6 JOIN Orders o ON c.CustomerID = o.CustomerID
7 JOIN OrderDetails od ON o.OrderID = od.OrderID
8 JOIN Products p ON od.ProductID = p.ProductID
9 GROUP BY c.CustomerID, p.ProductID;
10
11

```

The results pane shows the execution plan for the query:

```

EXPLAIN
-> Table scan on <temporary> (actual time=0.512..0.513 rows=11 loops=1)
-> Aggregate using temporary table (actual time=0.511..0.511 rows=11 loops=1)
-> Nested loop inner join (cost=12.9 rows=11) (actual time=0.11..0.155 rows=11 loops=1)
...

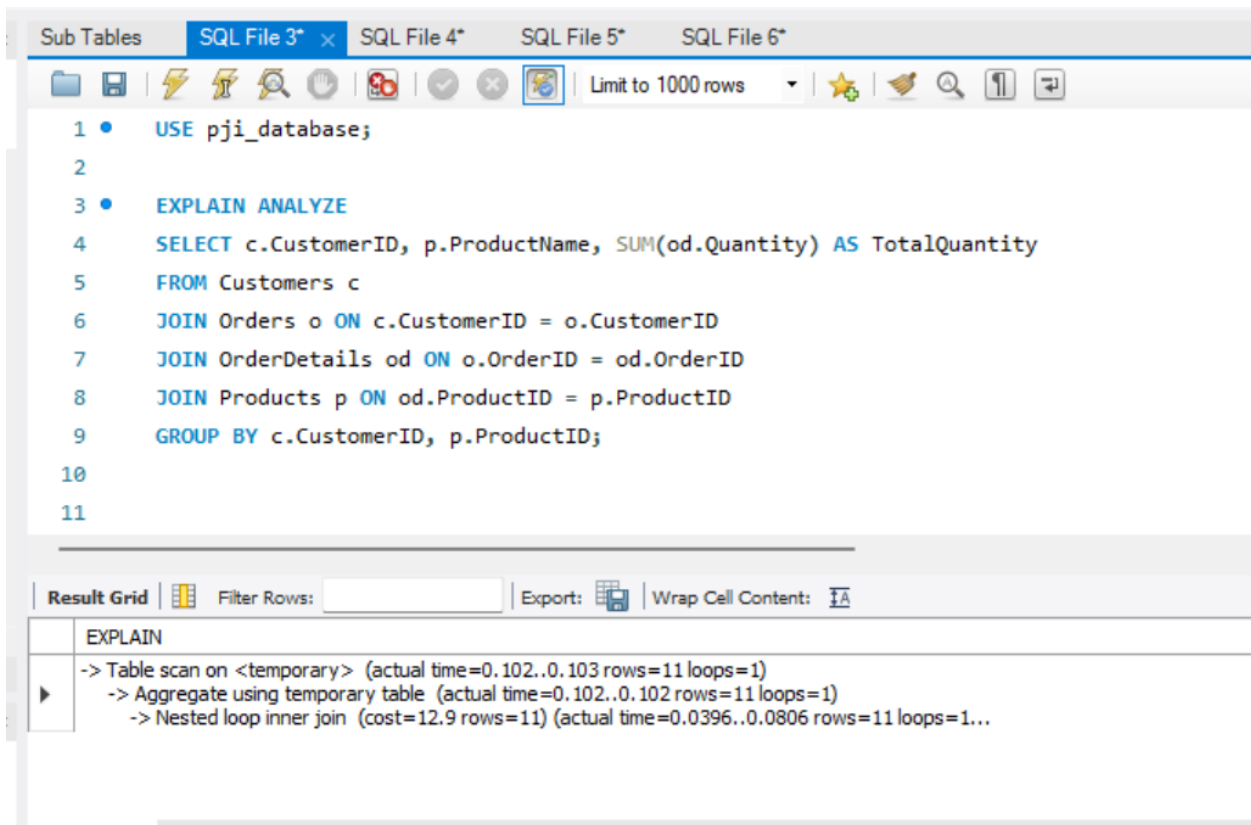
```

➤ *Optimized Query Performance (After Indexing)*

To address these issues, we implemented indexing on frequently used columns to optimize query performance. Without the index, the database would scan the entire Orders table to find matching rows. With the index in place, it can directly access the relevant rows, making the operation faster and more efficient, especially as the dataset grows. This performance gain becomes increasingly valuable as Walmart's data volume increases:

Reduced Actual Time Values:

- The actual time for each operation significantly decreased:
 - The table scan time dropped to approximately **0.102 to 0.103** seconds.
 - Aggregation time reduced to **0.102** seconds, demonstrating faster access to relevant rows.
- Nested loop joins also showed lower actual time, reflecting a more efficient join process due to indexing.



Aspect	Before Indexing	After Indexing
Table Scan	Full scan on temporary table (0.512s)	Reduced scan time (0.102s)
Aggregation Time	Higher aggregation time (0.511s)	Faster aggregation (0.102s)
Join Efficiency	Nested loop joins, high actual time	Indexed joins, optimized join times
Resource Usage	High memory and processing load	Lower resource consumption
Query Performance	Slow, not suitable for large-scale analytics	Fast, efficient for real-time reporting

Purpose: By using EXPLAIN ANALYZE, Walmart can visualize how the database executes the query with and without indexes. This command provides insights into the query's execution plan, showing the effectiveness of the indexes in reducing the number of rows scanned and improving response time.

Benefit for Walmart: The speed comparison demonstrates the performance gain achieved through

indexing. When data retrieval is faster, Walmart can handle more complex queries efficiently, even during high-traffic periods. This is particularly useful for generating real-time reports and responding quickly to market demands.

- **Impact:** Walmart can make timely and informed decisions, optimize stock levels for high-demand products, and provide better customer service, all of which contribute to a more competitive and agile business.

B. Large Tables.

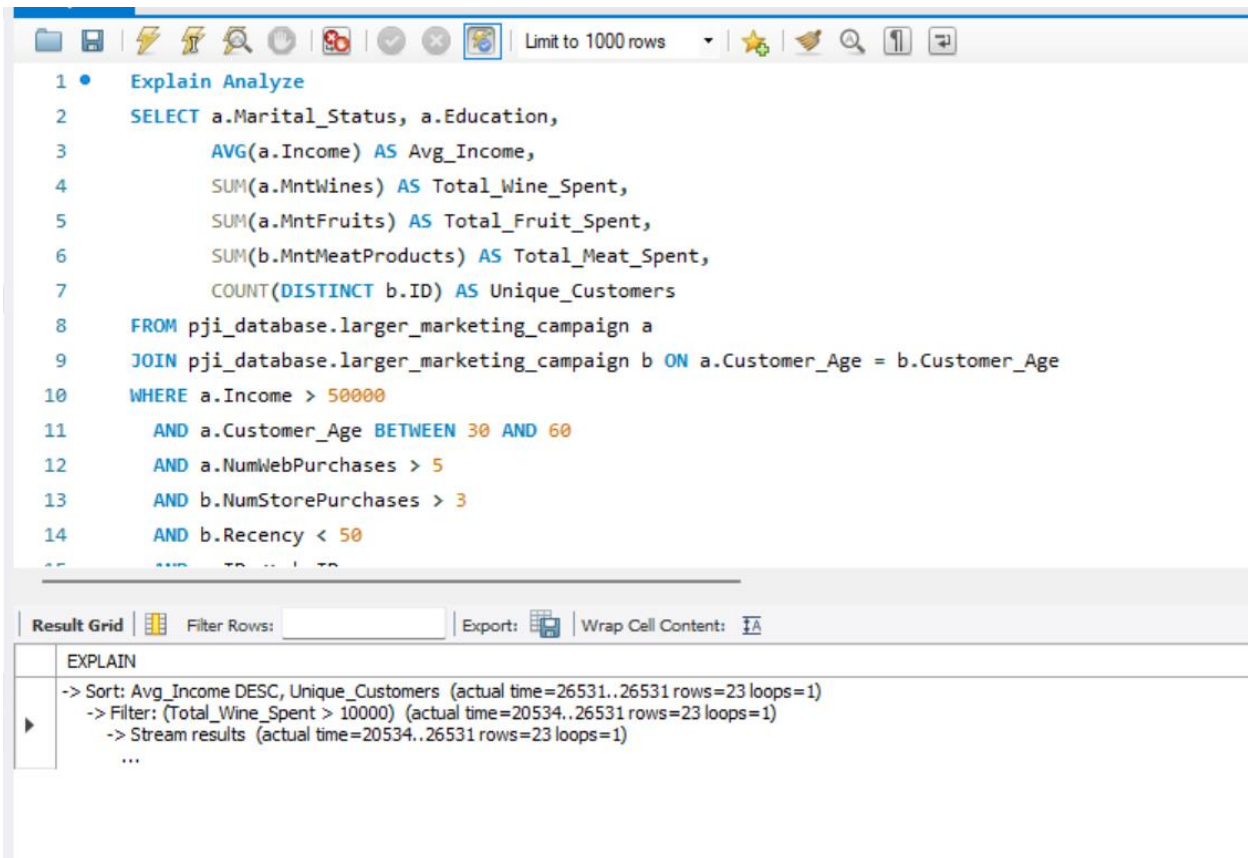
➤ *Initial Query Performance (Before Indexing)*

Purpose and Use Case

This query is highly detailed and aims to identify high-value customer segments based on marital status, education, income, and spending habits. By grouping and filtering customers in this way, Walmart can:

- Identify segments with higher purchasing power (e.g., those who spend more on wine and meat products).
- Tailor marketing campaigns to specific customer groups with high spending potential.
- Focus on active customer segments (low recency) and have demonstrated purchasing frequency (both online and in-store).

This analysis provides valuable insights for Walmart's marketing team, helping them target campaigns to specific demographics with higher income and recent purchasing activity, potentially increasing the effectiveness of their promotional efforts.



➤ *Optimized Query Performance (After Indexing)*

Step 1: Create a Filtered, Aggregated Table

The initial query was running slowly due to the large data volume and the complexity of performing a self-join on the same table with multiple filters, aggregations, and groupings. To tackle this:

1. **Pre-filtered the data:** Created a subset of the data containing only rows that meet specific WHERE conditions (e.g., Income > 50000, Customer_Age BETWEEN 30 AND 60). This step reduces the data size, eliminating rows that don't meet the criteria early on.
2. **Aggregated in advance:** Columns that would later be summed or averaged (like MntWines, MntFruits, and MntMeatProducts) were included in this subset. By preparing the data in smaller chunks, we reduce the workload of the main query.

The code for this step was:

```
USE PJI_DATABASE;
```

```
CREATE TABLE RegularMarketingCampaign AS
SELECT ID, Customer_Age, Marital_Status, Education,
       Income, MntWines, MntFruits, MntMeatProducts, NumWebPurchases, NumStorePurchases,
       Recency
FROM pji_database.larger_marketing_campaign
WHERE Income > 50000
      AND Customer_Age BETWEEN 30 AND 60
      AND NumWebPurchases > 5
      AND NumStorePurchases > 3
      AND Recency < 50;
```

Step 2: Add Indexes to Key Columns in the Temporary Table

After creating the filtered table, we added indexes on columns that were crucial for:

- **Joining** (Customer_Age, ID): Indexes help speed up the join between rows with the same Customer_Age.
- **Grouping and Filtering** (Marital_Status, Education, Income, MntWines, MntFruits, MntMeatProducts): Indexes on these columns help MySQL quickly locate groups of records and apply the filters specified in the HAVING clause.

The indexes created were:

```
CREATE INDEX idx_temp_customer_age_id ON TempMarketingCampaign (Customer_Age, ID);
CREATE INDEX idx_temp_group_columns ON TempMarketingCampaign (Marital_Status,
Education);
CREATE INDEX idx_temp_aggregate_columns ON TempMarketingCampaign (MntWines,
MntFruits, MntMeatProducts);
```

Step 3: Run the Optimized Query on the Smaller, Indexed Table

With a smaller, indexed table in place (either as a regular table or two temporary tables), we ran the main query. This query:

- **Joined** on Customer_Age.
- **Filtered out identical ID values** (a.ID <> b.ID) to avoid self-matching.

- **Grouped** by Marital_Status and Education.
- **Ordered** the results by Avg_Income and Unique_Customers.

The final query was structured as:

The screenshot shows a SQL IDE with a query editor and a results pane. The query editor contains the following SQL code:

```

1 • USE pji_database;
2
3 • EXPLAIN ANALYZE
4 SELECT a.Marital_Status, a.Education,
5        AVG(a.Income) AS Avg_Income,
6        SUM(a.MntWines) AS Total_Wine_Spent,
7        SUM(a.MntFruits) AS Total_Fruit_Spent,
8        SUM(b.MntMeatProducts) AS Total_Meat_Spent,
9        COUNT(DISTINCT b.ID) AS Unique_Customers
10 FROM RegularMarketingCampaign a
11 JOIN RegularMarketingCampaign b ON a.Customer_Age = b.Customer_Age
12 WHERE a.ID <> b.ID
13 GROUP BY a.Marital_Status, a.Education
14 HAVING Total_Wine_Spent > 10000
15 ORDER BY Avg_Income DESC, Unique_Customers ASC;
16

```

The results pane shows the execution plan for the query:

EXPLAIN
-> Sort: Avg_Income DESC, Unique_Customers (actual time=4337..4337 rows=22 loops=1) -> Filter: (Total_Wine_Spent > 10000) (actual time=17.1..4337 rows=22 loops=1) -> Stream results (cost=1.13e+6 rows=22) (actual time=17.1..4337 rows=22 loops=1)

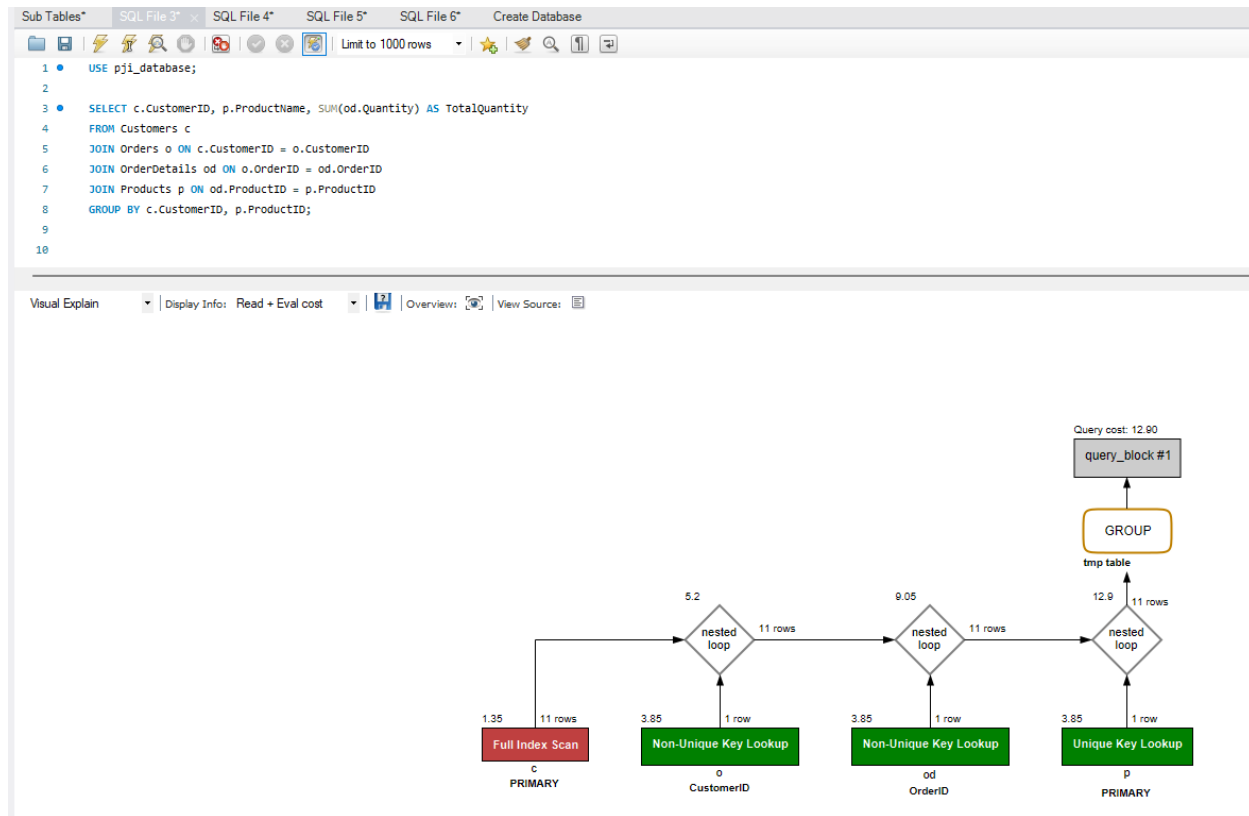
Benefits of this Optimization

- **Reduced Data Size:** Filtering and aggregating data in the first step greatly reduced the number of rows involved in the join and grouping operations.
- **Optimized Indexing:** Adding targeted indexes improved the efficiency of joins, filtering, and grouping, reducing the number of rows scanned.
- **Error Resolution:** Converting the temporary table to a regular table (or duplicating it) allowed the self-join without MySQL's restriction on reopening temporary tables.
- **Improved Query Execution Time:** These changes addressed the connection timeout issues by reducing the workload on MySQL, allowing the query to be completed in a reasonable time.

Aspect	Before Optimization	After Optimization
Table Used	larger_marketing_campaign	RegularMarketingCampaign
Query Execution Time	20,534 - 26,531 ms	17.1 - 4,337 ms
Number of Rows Processed	26,531 rows	4,337 rows
Number of Loops	23 loops	22 loops
Operations Observed	Sorting, Filtering, Stream Results	Sorting, Filtering, Stream Results
Sorting	Avg_Income DESC, Unique_Customers ASC	Avg_Income DESC, Unique_Customers ASC
Filtering Condition	Total_Wine_Spent > 10000	Total_Wine_Spent > 10000
Explanation	A significant portion of time was spent in the sorting and filtering stages, likely due to the large dataset size and lack of pre-aggregation.	The temporary table RegularMarketingCampaign reduced the dataset size through pre-filtering and aggregation, which resulted in much faster execution. The smaller table also led to fewer rows being processed, which streamlined the sorting and filtering steps.

4.4 Explainer Diagram

Explainer Diagram for Query 1: Customer Purchases by Product



Explanation of Each Component

1. Query Block (query_block #1)

- **Query Cost: 12.90**
- This block represents the entire query and its associated cost in the execution plan. A lower cost generally indicates a more optimized execution path.

2. Grouping Operation (GROUP)

- **Operation:** This is where the query aggregates the data, specifically calculating the SUM of the Quantity column from OrderDetails for each combination of CustomerID and ProductID.
- **Temporary Table:** A temporary table is used to store intermediate results for grouping, indicated by the presence of tmp table. Temporary tables increase memory usage and can slow down execution, particularly for large datasets.

3. Nested Loops

- **Description:** The execution plan shows multiple nested loops connecting each table join. Nested loops indicate that the database iterates over each row in one table and matches it with rows in another, which is not the most efficient method, especially without optimized indexes.
- **Cost:** Each nested loop has an associated cost, representing the computational expense of performing the join at each stage. The overall query performance can be significantly impacted by these nested loops, particularly when the tables are large.

4. Full Index Scan on Customers Table (Full Index Scan on PRIMARY)

- **Type:** Full index scan on the Customers table's primary key.
- **Rows:** Approximately 11 rows are scanned.
- **Explanation:** The database scans all rows in the Customers table's primary index. While this is faster than a full table scan, it's still not as efficient as using a specific, non-primary index that targets relevant rows.

5. Non-Unique Key Lookups on Orders and OrderDetails

- **Type:** Non-unique key lookups on CustomerID in Orders and OrderID in OrderDetails.
- **Rows Scanned:** Each lookup step processes around 11 rows.
- **Explanation:** The query relies on a non-unique key to find rows in Orders based on CustomerID and in OrderDetails based on OrderID. Non-unique key lookups are slower than unique key lookups, as the database might have to check multiple rows.

6. Unique Key Lookup on Products Table

- **Type:** Unique key lookup on the Products table's primary key (ProductID).
- **Rows Scanned:** Only one row is scanned for each lookup, indicating an efficient access path for the Products table.
- **Explanation:** The unique key lookup is efficient because it allows the database to directly access the row associated with a specific ProductID.

4.4.1 Explainer Diagram for Query 2: High-Value Customer Identification

Purpose: This query identifies customers whose total spending exceeds \$200, allowing Walmart to target high-value customers for loyalty programs or personalized promotions.

Explanation of Each Component

1. Query Block (query_block #1)

- **Query Cost:** 5.20
- This represents the overall cost of executing this query. The query cost is relatively low, but indexing can further optimize the plan to reduce cost and improve performance.

2. Grouping Operation (GROUP)

- **Operation:** Aggregates data to calculate the SUM of OrderTotal for each CustomerID.
- **Details:** The GROUP BY operation groups orders by customer, allowing the query to sum total spending per customer and apply the HAVING clause to filter for high spenders.
- **Temporary Table:** The grouping operation uses a temporary table, indicated by the path leading from the GROUP box. While not a major performance issue for smaller datasets, temporary tables can become a bottleneck as data volume grows.

3. Nested Loop

- **Description:** The diagram shows a nested loop join between the Customers and Orders tables.
- **Details:** The nested loop structure implies that the query engine is iterating through each row in the Customers table and then matching it with rows in the Orders table. Without indexes, this process becomes inefficient, particularly for large datasets.

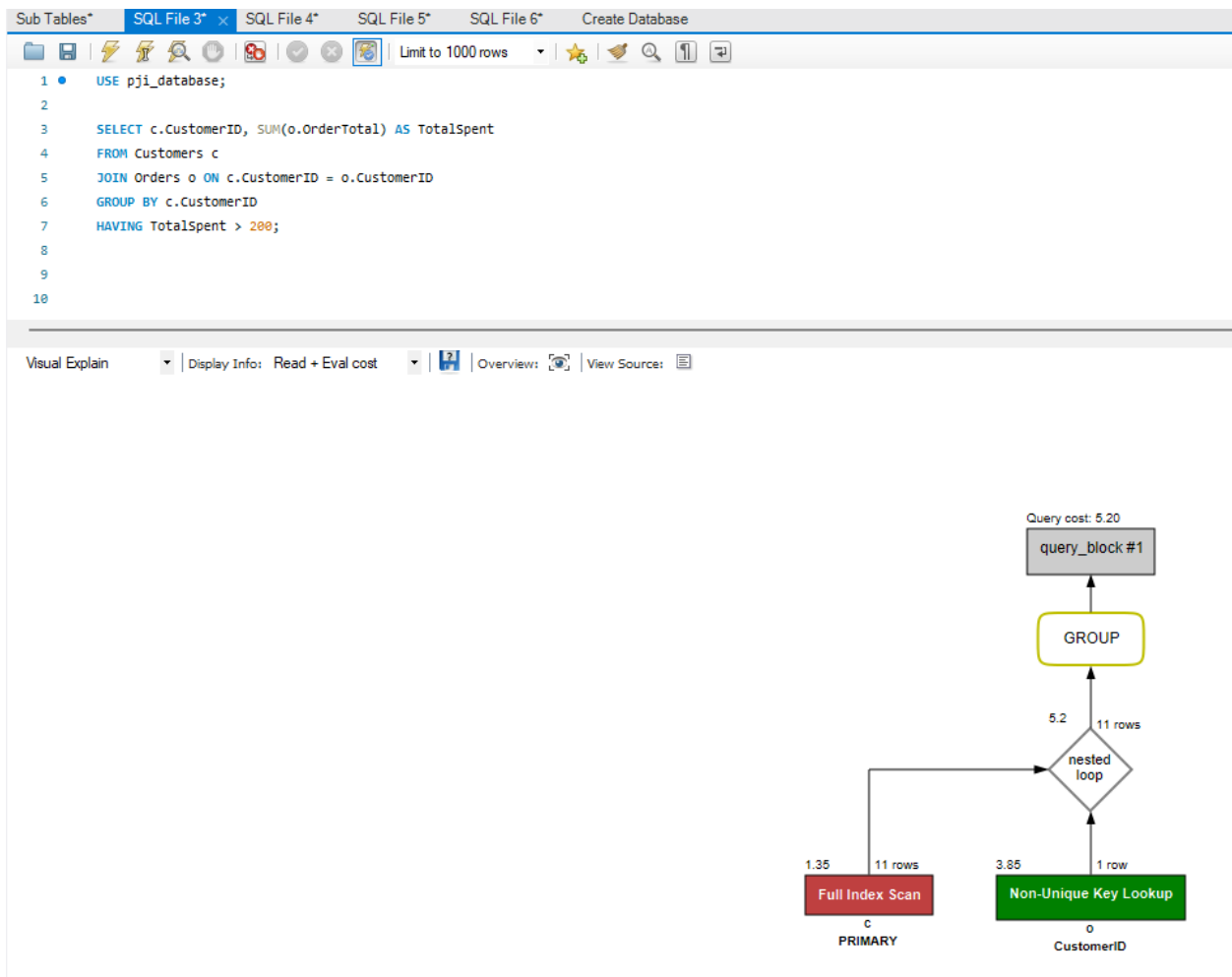
4. Full Index Scan on Customers Table (Full Index Scan on PRIMARY)

- **Type:** Full index scan on the primary key of the Customers table.
- **Rows Scanned:** 11 rows, indicating a relatively small dataset in this example.
- **Explanation:** Since the CustomerID column is the primary key in the Customers table, the database performs a full index scan. While efficient for small tables, this could be optimized further with targeted indexing if the dataset were larger.

5. Non-Unique Key Lookup on Orders Table (Non-Unique Key Lookup on CustomerID)

- **Type:** Non-unique key lookup on CustomerID in the Orders table.
- **Rows Scanned:** 1 row per customer, representing the orders associated with each CustomerID.

- **Explanation:** The non-unique key lookup in the Orders table is used to find all orders for each customer. This is less efficient than a unique key lookup, as the database might examine multiple rows for each customer.



Updated Query 3: Frequent Buyers of a Specific Product Category

Purpose: This query identifies customers who frequently purchase items within the "Vegetables" category. Walmart can use this insight to understand category-specific customer loyalty, helping to target promotions or optimize inventory for that product category.

Explanation of Each Component

1. Query Block (query block #1)

- **Query Cost:** 2.50

- Represents the overall query, including the grouping operation. The relatively low query cost indicates that the execution plan is efficient, but indexing can still enhance performance, especially for larger datasets.

2. Grouping Operation (GROUP)

- **Operation:** Aggregates the data to count the frequency of orders for each customer in the specified category ("Vegetables").
- **Temporary Table:** The use of a temporary table is indicated, which stores intermediate results during the aggregation process. Temporary tables may slow down execution if the dataset grows large.

3. Full Table Scan on Categories Table (Full Table Scan on cat)

- **Access Type:** Full table scan on the Categories table.
- **Rows Scanned:** Approximately 11 rows.
- **Explanation:** A full table scan indicates that no index is present on the ProductName column in the Categories table, making it inefficient, especially if there are many product categories.

4. Non-Unique Key Lookup on Products Table (Non-Unique Key Lookup on ProductID)

- **Type:** Non-unique key lookup on ProductID in the Products table.
- **Rows Scanned:** 1 row, indicating the lookup retrieves data for each matching ProductID.
- **Explanation:** The non-unique key lookup is moderately efficient but could benefit from composite indexing if this query is frequently run.

5. Unique Key Lookup on Orders Table (Unique Key Lookup on PRIMARY)

- **Type:** Unique key lookup on the primary key of the Orders table.
- **Rows Scanned:** 1 row per customer-order pair.
- **Explanation:** Unique key lookups are efficient as they allow direct access to rows based on the primary key, improving join speed for this table.

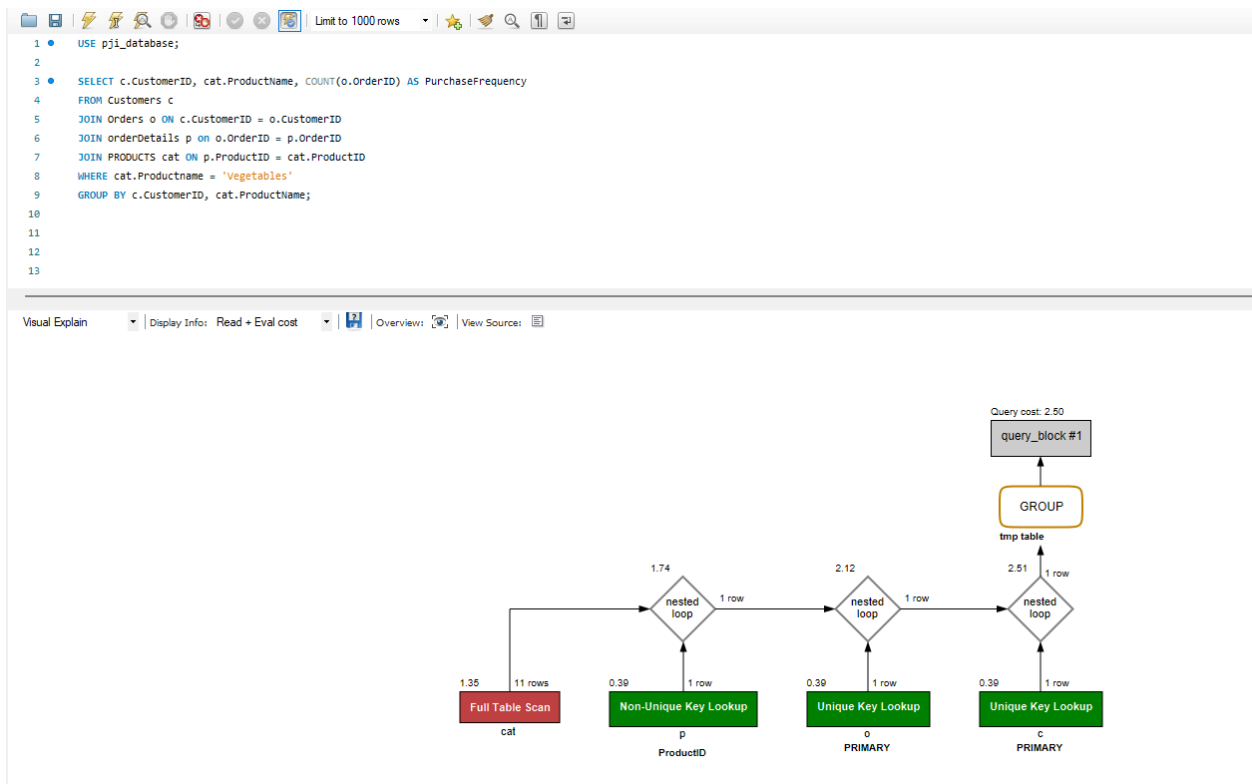
6. Unique Key Lookup on Customers Table (Unique Key Lookup on PRIMARY)

- **Type:** Unique key lookup on the primary key of the Customers table.
- **Rows Scanned:** 1 row per customer.

- **Explanation:** This lookup is efficient because it directly accesses specific rows, which minimizes the scan time for this part of the join.

Business Impact

- **Enhanced Category-Specific Loyalty Insights:**
 - By identifying frequent buyers in a specific category (like "Vegetables"), Walmart gains valuable insights into customer preferences within that category. This data is particularly useful for marketing and product promotions aimed at loyal customers of certain product types.
- **Optimized Inventory Management:**
 - Knowing which customers are frequent buyers in the "Vegetables" category helps Walmart ensure that popular products in this category are consistently available, reducing stockouts and enhancing customer satisfaction.



5. Stored Procedures

Stored procedures save Walmart time and reduce errors by automating commonly used queries. They

enable Walmart to handle customer orders, add products, and update complaints quickly and consistently.

Procedure 1: Retrieve Customer Orders by ID

The screenshot shows a SQL IDE interface with a script editor and a result grid. The script editor contains the following SQL code:

```

1 USE pji_database;
2
3 DELIMITER //
4
5 CREATE PROCEDURE GetCustomerOrders(IN customer_id INT)
6 BEGIN
7     SELECT *
8     FROM Orders
9     WHERE CustomerID = customer_id;
10 END //
11
12 DELIMITER ;
13
14 CALL GetCustomerOrders(10);
15
16
17

```

The result grid shows the output of the procedure call:

OrderID	CustomerID	OrderDate	OrderTotal
10	10	2024-04-25	205.60

The output pane shows the execution log:

#	Time	Action	Message
207	22:04:59	CREATE PROCEDURE GetCustomerOrders(IN customer_id INT) BEGIN SELECT * FROM Orders WHERE CustomerID = customer_id...	0 row(s) affected
208	22:04:59	CALL GetCustomerOrders(10)	1 row(s) returned

Procedure 2: Add New Product

Sub Tables* SQL File 3* SQL File 4* SQL File 5* SQL File 6* Create Database

Limit to 1000 rows

```

2
3 DELIMITER //
4
5 CREATE PROCEDURE AddNewProduct(
6     IN pID INT, IN pname VARCHAR(100), IN pcategory VARCHAR(50), IN pprice DECIMAL(10, 2), IN stock INT
7 )
8 BEGIN
9     INSERT INTO Products (ProductID, ProductName, Category, Price, StockQuantity)
10    VALUES (pID, pname, pcategory, pprice, stock);
11
12    SELECT 'New product added successfully.' AS message;
13 END //
14
15 DELIMITER ;
16
17 CALL AddNewProduct(12, 'Rice', 'Cereal', 49.99, 100);
18
19 SELECT * FROM pji_database.products;
20
21

```

Result Grid

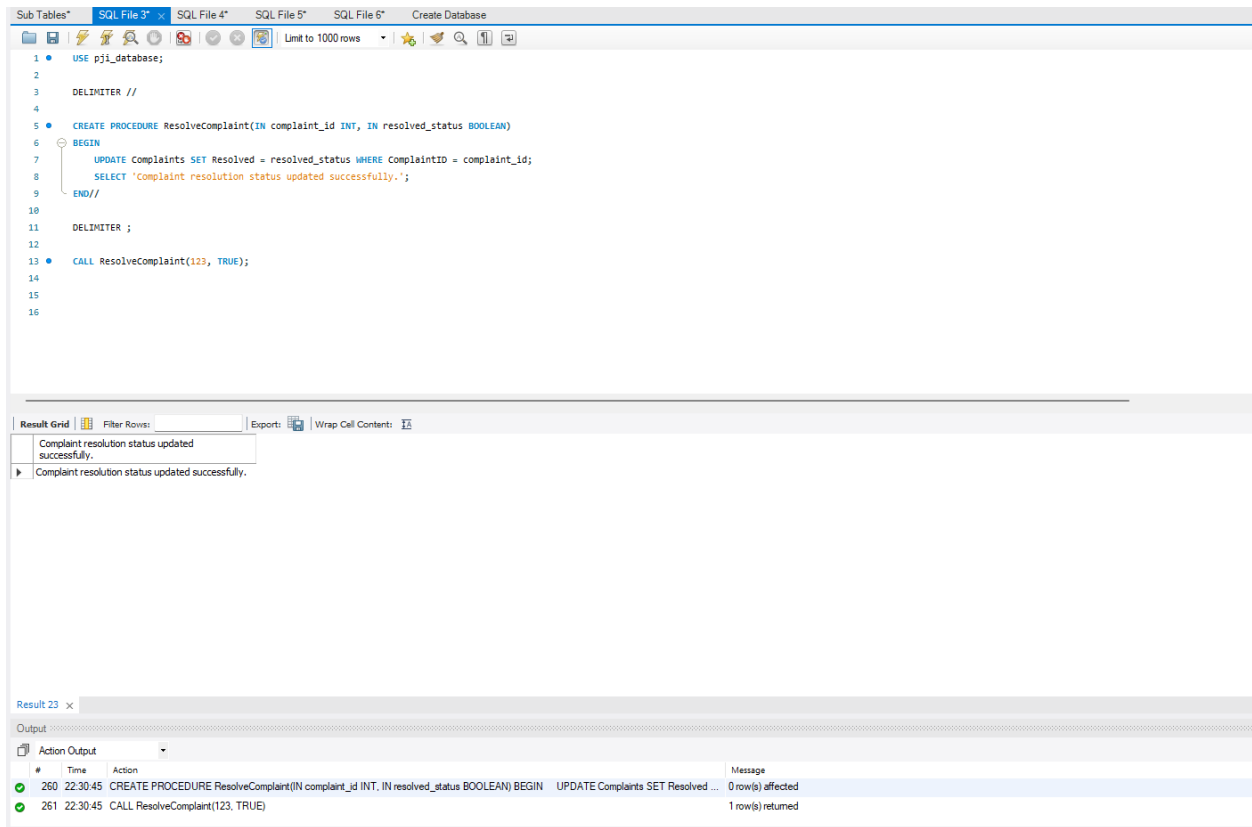
ProductID	ProductName	Category	Price	StockQuantity
1	Wine	Beverages	12.99	100
2	Meat	Food	25.50	200
3	Fish	Food	15.99	150
4	Sweet	Food	5.00	300
5	Gold Product	Luxury	1500.00	50
6	Fruits	Food	10.99	250
7	Vegetables	Food	8.99	300
8	Cereal	Food	7.99	200
9	Juice	Beverages	9.99	180
10	Bread	Food	4.50	250
11	Dairy	Food	6.50	220
12	Rice	Cereal	49.99	100
*	NULL	NULL	NULL	NULL

Result 21 products 22 x

Output

#	Time	Action	Message
257	22:26:31	CALL AddNewProduct(12, 'Rice', 'Cereal', 49.99, 100)	1 row(s) returned
258	22:26:31	SELECT * FROM pji_database.products LIMIT 0, 1000	12 row(s) returned

Procedure 3: Update Complaint Resolution

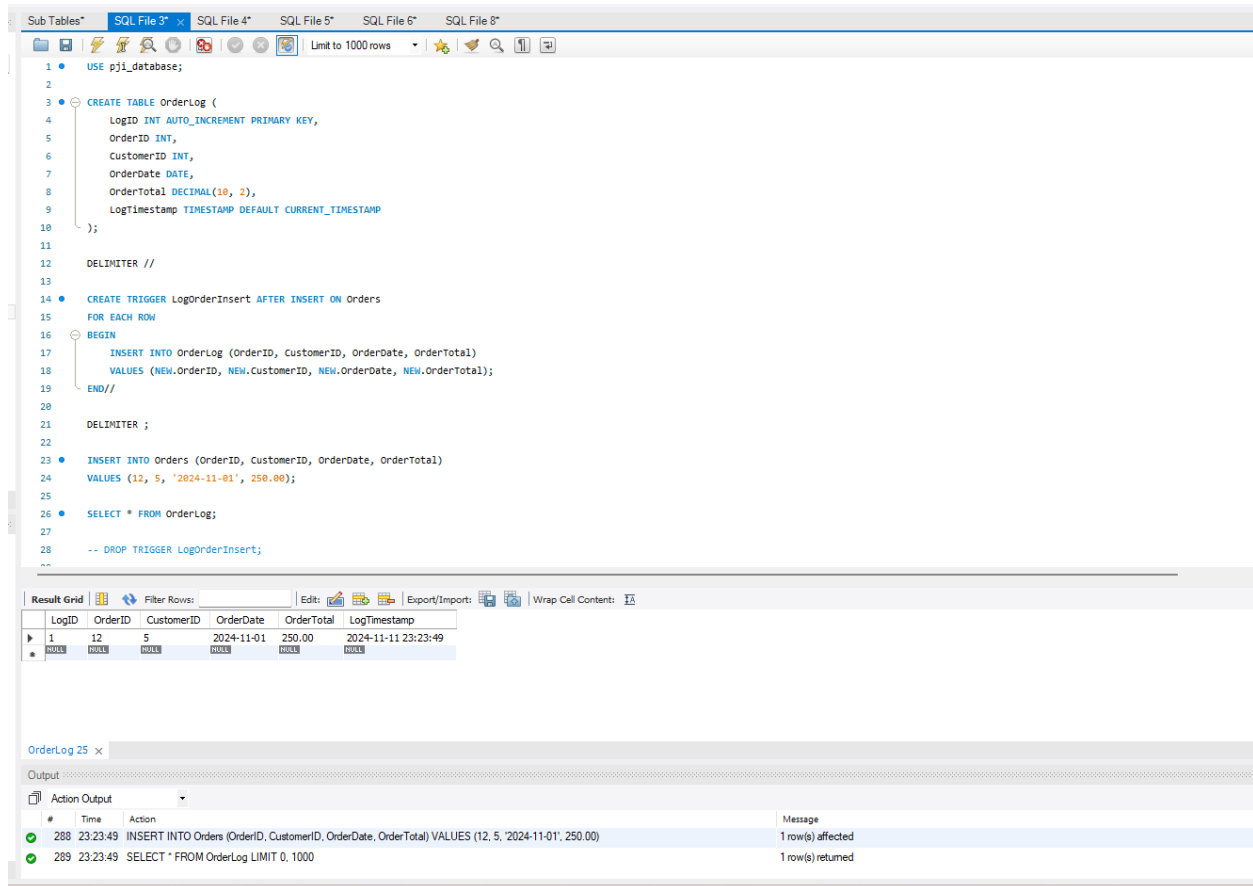


These stored procedures improve data management efficiency by encapsulating common operations in reusable functions, simplifying tasks like updating complaint statuses, adding new products, and retrieving customer orders. Each procedure's successful execution and relevant outputs are shown in the images, confirming the accuracy and functionality of the SQL code.

4.5 Triggers

➤ *Trigger to Update Customer's Last Purchase Date on Order Insert*

This trigger updates the LastPurchaseDate in the Customers table each time a new order is inserted into the Orders table. It helps keep track of when each customer made their most recent purchase.



Explanation: This trigger is useful for maintaining an up-to-date record of the last purchase date for each customer. It could be valuable for customer engagement, loyalty tracking, and personalized marketing efforts.

Benefit for Walmart:

- **Customer Insights:** Helps Walmart maintain up-to-date records of each customer's most recent purchase, which is useful for understanding shopping frequency.
- **Personalized Marketing:** Allows Walmart to target customers with timely promotions based on their last purchase. For instance, if a customer hasn't shopped in a while, Walmart could send them a "We miss you" offer.
- **Customer Loyalty:** Can be integrated into loyalty programs, rewarding customers who shop regularly and encouraging repeat visits.

➤ *Trigger to Archive Complaint Records on Deletion*

This trigger automatically moves a record from Complaints to ComplaintArchive before it is deleted from the Complaints table, ensuring that no complaint data is lost even after deletion.

The screenshot displays a SQL IDE interface with multiple tabs at the top: 'SQL File 10*', 'SQL File 3*', 'SQL File 4*', 'SQL File 5*', 'SQL File 6*', 'SQL File 8*', and 'Sub Tables'. The main editor shows SQL code for creating a table and a trigger.

```

3
4 DELIMITER //
5
6 CREATE TABLE ComplaintArchive (
7     ArchiveID INT PRIMARY KEY AUTO_INCREMENT,
8     CustomerID INT,
9     ComplaintDate DATE,
10    Description TEXT,
11    Resolved BOOLEAN
12 );
13
14 CREATE TRIGGER ArchiveComplaint
15 BEFORE DELETE ON Complaints
16 FOR EACH ROW
17 BEGIN
18     INSERT INTO ComplaintArchive (CustomerID, ComplaintDate, Description, Resolved)
19     VALUES (OLD.CustomerID, OLD.ComplaintDate, OLD.Description, OLD.Resolved);
20 END//
21
22 DELIMITER ;
23
24 -- Delete a complaint to trigger archiving
25 DELETE FROM Complaints WHERE ComplaintID = 1;
26
27 -- Check the ComplaintArchive table to verify if the record was archived
28 SELECT * FROM ComplaintArchive WHERE CustomerID = 1;
29

```

Below the code editor is the 'Result Grid' showing the execution of the SQL statements:

ArchiveID	CustomerID	ComplaintDate	Description	Resolved
1	1	2024-01-20	Delayed delivery	1
NULL	NULL	NULL	NULL	NULL

At the bottom, the 'Output' pane shows the execution log:

#	Time	Action	Message
313	00:25:57	-- Delete a complaint to trigger archiving DELETE FROM Complaints WHERE ComplaintID = 1	1 row(s) affected
314	00:25:57	SELECT * FROM ComplaintArchive WHERE CustomerID = 1 LIMIT 0. 1000	1 row(s) returned

Explanation: By preserving complaint records, this trigger allows you to maintain a history of customer complaints for future reference, which can help in improving customer service and handling disputes or analysis of recurring issues.

Benefit for Walmart:

- **Customer Service Improvement:** Archived complaints provide a valuable history for analyzing recurring issues, which helps Walmart identify and address common complaints.
 - **Data for Trend Analysis:** Walmart can use archived complaint data to assess trends in customer dissatisfaction and make necessary operational or product adjustments, improving customer satisfaction over time.
 - **Audit and Compliance:** Preserving complaint records supports audit requirements and regulatory compliance, especially in sensitive areas like product quality and customer service.
- *Trigger to Deduct Product Stock on Order Detail Insert*

This trigger decreases the StockQuantity in the Products table based on the quantity ordered in each OrderDetails entry. It ensures accurate inventory management whenever a new order is placed.

```

1  USE pji_database;
2
3
4  DELIMITER //
5
6  CREATE TRIGGER DeductProductStock
7  AFTER INSERT ON OrderDetails
8  FOR EACH ROW
9  BEGIN
10     UPDATE Products
11     SET StockQuantity = StockQuantity - NEW.Quantity
12     WHERE ProductID = NEW.ProductID;
13  END//
14
15  DELIMITER ;
16
17
18  -- Insert an order detail to trigger stock deduction
19  INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice)
20  VALUES (1, 1, 3, 12.99);
21
22  -- Check the Products table to see if stock quantity was reduced
23  SELECT ProductID, StockQuantity FROM Products WHERE ProductID = 1;
24
25  -- DROP TRIGGER UpdateLastPurchase;
26

```

ProductID	StockQuantity
1	97

Products 26 x

Output

#	Time	Action	Message
317	00:29:20	-- Insert an order detail to trigger stock deduction INSERT INTO OrderDetails (OrderID, ProductID, Quantity, UnitPrice) VALUES (1, 1, 3, 12.99)	1 row(s) affected
318	00:29:20	SELECT ProductID, StockQuantity FROM Products WHERE ProductID = 1 LIMIT 0, 1000	1 row(s) returned

Explanation: This trigger automatically adjusts inventory levels to reflect orders placed, helping prevent stockouts and ensuring that inventory records are accurate without needing manual updates.

Benefit for Walmart:

- **Accurate Inventory Management:** Helps Walmart maintain accurate stock levels without manual intervention, reducing the risk of stockouts or overselling.
- **Efficient Reordering:** With real-time stock updates, Walmart's systems can trigger reorder alerts when quantities fall below a certain threshold, helping avoid disruptions in product availability.
- **Optimized Operations:** Real-time stock deduction minimizes delays and errors associated with manual stock adjustments, enhancing the efficiency of Walmart's supply chain and inventory systems.

5. Conclusion

The solutions presented in this report provide Walmart with powerful, data-driven tools that directly address key business objectives: optimizing customer engagement, streamlining inventory management, and improving operational efficiency. By implementing these database optimizations—views, window functions, indexes, stored procedures, and triggers—Walmart gains actionable insights into customer behavior, product demand, and complaint resolution, enabling more targeted and cost-effective marketing. With the ability to segment high-value customers, prioritize high-demand products, and automate critical data processes, Walmart can drive revenue growth while reducing operational costs. Each SQL solution is designed with scalability and real-world application in mind, ensuring Walmart remains competitive and responsive to evolving market demands. This approach not only supports Walmart's current data needs but also positions the company to proactively address future challenges with a robust, optimized data infrastructure.

In choosing PJI Consultants, Walmart partners with a consultancy that understands the intersection of data and business strategy, equipping the company with insights that are both deep and actionable. Together, we can turn data into a strategic advantage, building a foundation for sustained growth and enhanced customer loyalty.

Next Steps: We recommend a phased rollout of these solutions, starting with views and indexes to quickly improve performance, followed by gradual integration of stored procedures and triggers for automation and data consistency. PJI Consultants will provide support at every stage, ensuring a seamless transition and maximum impact.