



TARGET DATA WAREHOUSE PROJECT – REPORT



Target Data Warehouse Project – Final Report

1. Introduction

This report details the process of sourcing, integrating, and implementing an ETL pipeline for three datasets that are meaningfully combined in a data warehouse. The datasets are selected based on their suitability for integration, their raw format, and their adequacy for dimensional modeling. The report also includes the complete ETL process, dimensional modeling, and analytical querying.

2. Data Warehouse Platform and Tools Used

2.1 Data Warehouse Platform:

- MySQL Database for structured storage and querying

2.2 ETL Tools Used:

- Excel: Used for preprocessing small datasets (cleaning, formatting, and deduplication)
- MySQL LOAD DATA INFILE: Used for bulk data loading

3. Data Source Selection Criteria

The following criteria were used to select the data sources:

- Raw Format: Data is available in CSV format.
- Related Information: Datasets contain interconnected details that allow for integration.
- Record Count: Each dataset contains at least 500 records to facilitate meaningful analysis.
- Dimensional Modeling Suitability: The datasets are structured using a star schema.

4. Data Source Selection Criteria

To start the project, three related datasets were chosen: Customers_TargetEcom, Products_TargetEcom, and Sales_TargetEcom. These datasets covered the most important aspects of an e-commerce business. They contained information about customer demographics, purchase history, transactional details, and product catalog data.

4.1 Customers_TargetEcom Dataset

- **Format:** CSV
- **Description:** Contains demographic and behavioral information about customers.
- **Attributes:** CustomerID, Name, Email, Phone, Address, City, State, Zip_Code, Registration_Date, Loyalty_Score, Customer_Type.

- **Integration Justification:** This dataset is essential for analyzing customer behaviors, purchase patterns, and segmentation strategies.

4.2 Products_TargetEcom Dataset

- **Format:** CSV
- **Description:** Includes details about the product catalog available in the Target e-commerce store.
- **Attributes:** ProductID, ProductName, Category, Brand, Price, Stock_Level, Review_Rating.
- **Integration Justification:** Links with transaction data to evaluate product performance and sales trends.

4.3 Sales_TargetEcom Dataset

- **Format:** CSV
- **Description:** Serves as the main fact table containing transactional details.
- **Attributes:** OrderID, Dates, StoreID, CustomerID, ProductID, Quantity, Discount, Tax, ShippingFee, TotalAmount, PaymentMethod, City, State, OrderStatus, DeliveryType.
- **Integration Justification:** Provides critical insights into revenue, customer purchase behaviors, and operational efficiency.

A **detailed data dictionary** was created to document field definitions and relationships between these datasets, ensuring consistency and clarity in data integration.

Documentation of Data Sources & Relationships

Each dataset is related through unique identifiers:

- **Customers (customer_id)** links to **Sales (customer_id)**
- **Products (product_id)** links to **Sales (product_id)**
- **Stock(stock_id)** links to **Products (product_id)**
- **Sales (transaction_id)** links to **PaymentMethods (transaction_id)**
- **Sales (transaction_id)** links to **ShippingStatus (transaction_id)**

5. Normalized Database Implementation

The raw datasets were analyzed to identify entities, attributes, and their relationships. The normalization process resulted in the creation of **seven** interrelated tables that adhere to **Third Normal Form (3NF)**, ensuring that each attribute depends solely on the primary key.

5.1 Customers Table

Stores primary customer data, ensuring the uniqueness of customer records.

- **Key Attributes:** customer_id (Primary Key), name, email, phone, address, city, state, zip_code, registration_date, loyalty_score.
- **Normalization Justification:** Customer demographic details are stored separately, ensuring data integrity and eliminating redundancy.

5.2 Products Table

Contains unique product-related details to standardize product information.

- **Key Attributes:** product_id (Primary Key), product_name, category, price, brand, supplier.
- **Normalization Justification:** Product categories are stored separately to reduce data duplication and ensure efficient updates.

5.3 Stock Table

Stores stock-related details, maintaining a reference to the Products table.

- **Key Attributes:** stock_id (Primary Key), product_id (Foreign Key), stock_quantity, discount, rating.
- **Normalization Justification:** Separating stock information prevents unnecessary duplication within the Products table.

5.4 Payment Methods Table

A reference table for payment methods to standardize transactional data.

- **Key Attributes:** payment_id (Primary Key), payment_method.
- **Normalization Justification:** Standardizing payment methods avoids inconsistencies across transaction records.

5.5 Shipping Status Table

Manages shipping details with a unique status reference.

- **Key Attributes:** shipping_id (Primary Key), shipping_status.

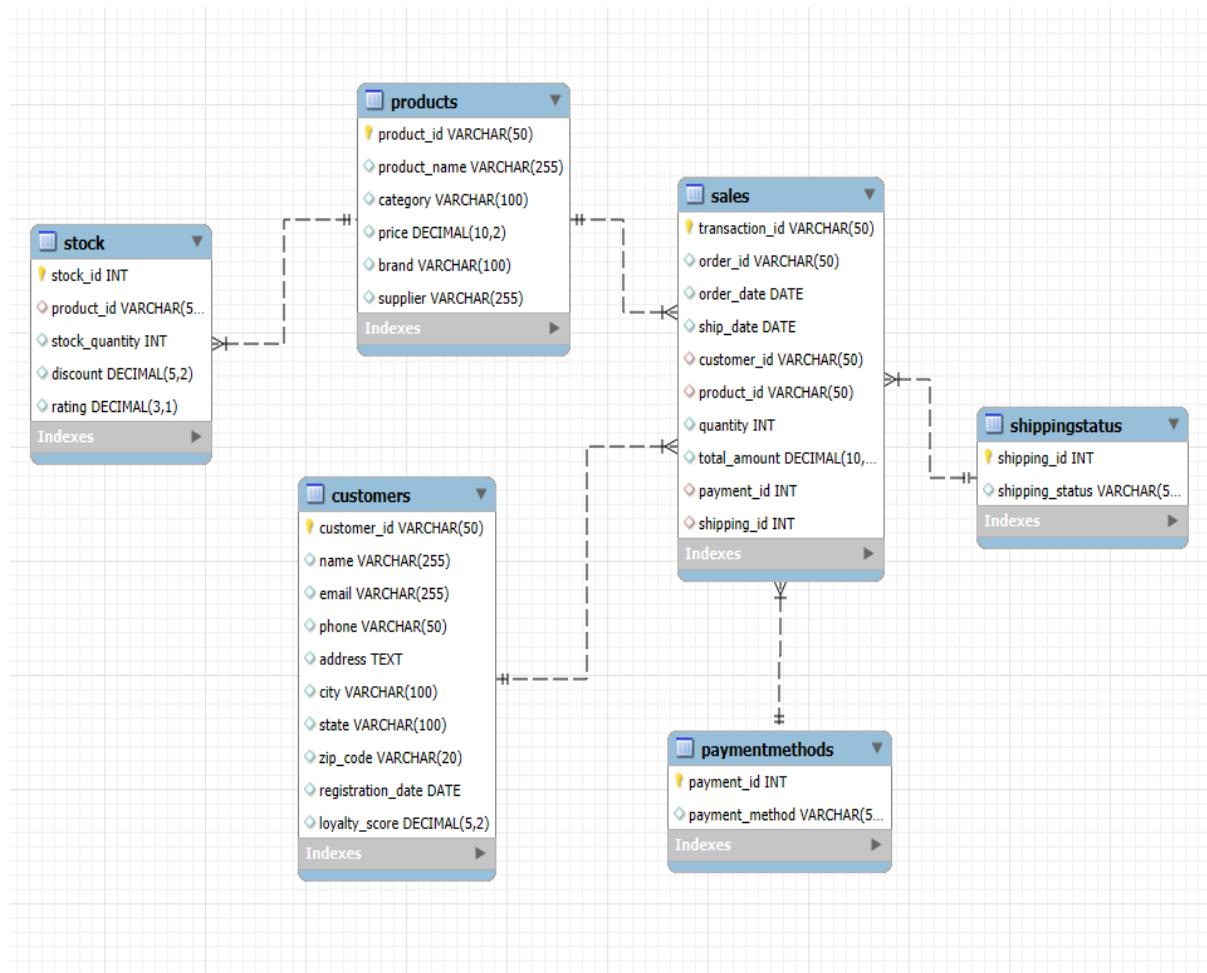
- **Normalization Justification:** Creating a lookup table ensures shipping status is not repeated in multiple transaction records.

5.6 Sales Table

Holds transactional order details linked to customers and locations.

- **Key Attributes:** transaction_id (Primary Key), customer_id (Foreign Key), payment_id(FK), product_id(FK), shipping_id(FK), order_date, ship_date.
- **Normalization Justification:** Storing order headers separately allows efficient tracking and relationship management.

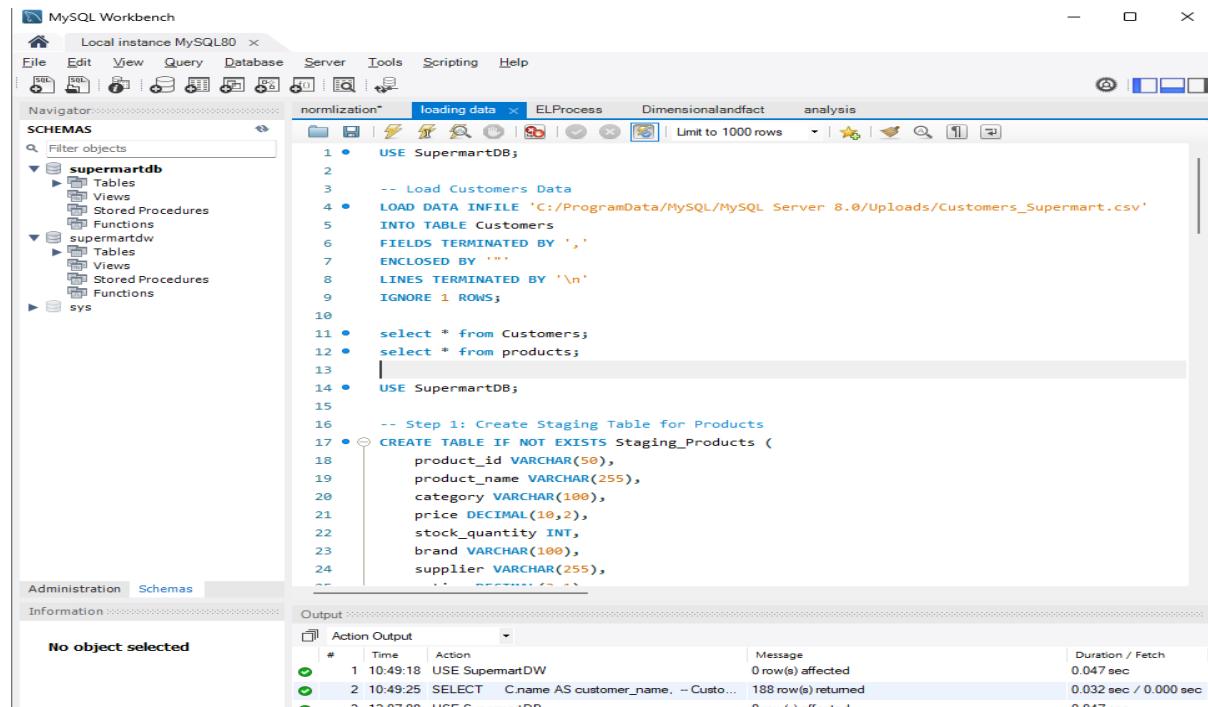
5.8 ER Diagram Representation



6. Data Loading and Staging Implementation

6.1 Data Ingestion into Normalized Tables

Raw data was imported into the database tables using bulk loading techniques, ensuring efficiency and minimal manual intervention.



The screenshot shows the MySQL Workbench interface with the 'loading data' tab selected. The central pane displays the following SQL script:

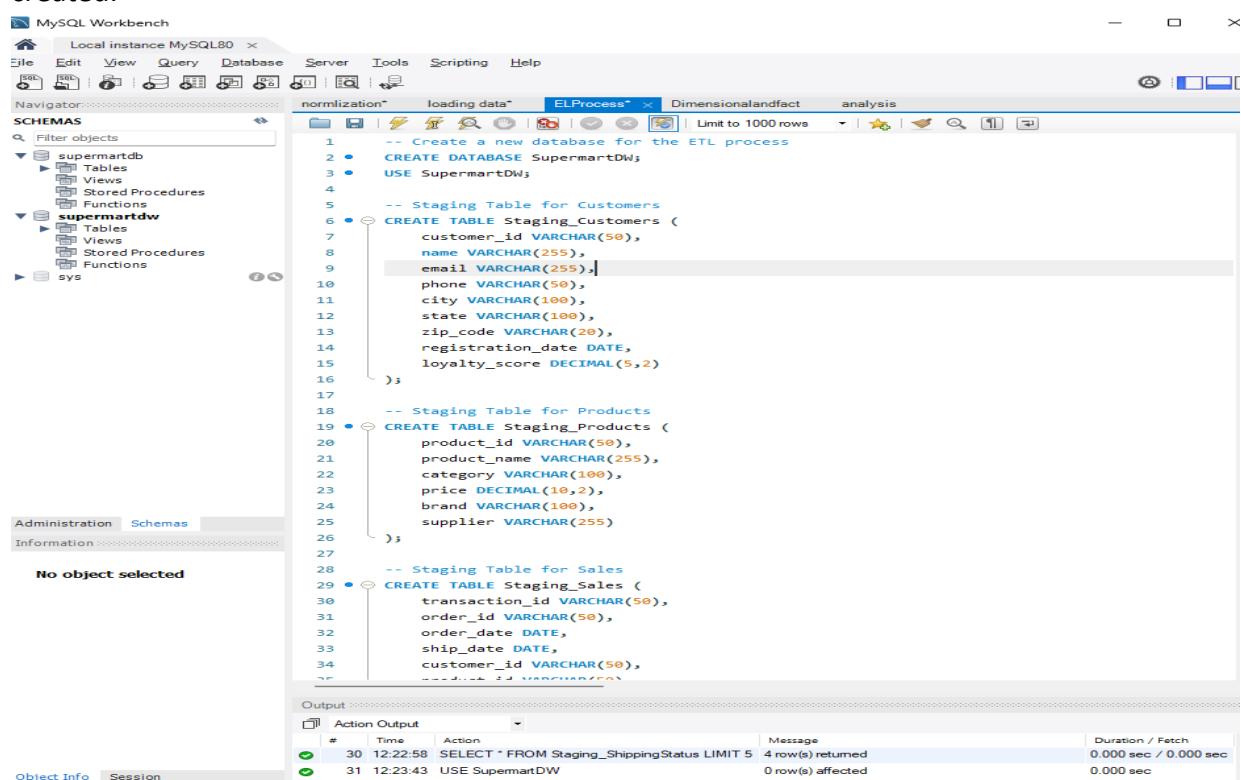
```
1 USE SupermartDB;
2
3 -- Load Customers Data
4 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/Customers_Supermart.csv'
5 INTO TABLE Customers
6 FIELDS TERMINATED BY ','
7 ENCLOSED BY ""
8 LINES TERMINATED BY '\n'
9 IGNORE 1 ROWS;
10
11 select * from Customers;
12 select * from products;
13
14 USE SupermartDB;
15
16 -- Step 1: Create Staging Table for Products
17 CREATE TABLE IF NOT EXISTS Staging_Products (
18     product_id VARCHAR(50),
19     product_name VARCHAR(255),
20     category VARCHAR(100),
21     price DECIMAL(10,2),
22     stock_quantity INT,
23     brand VARCHAR(100),
24     supplier VARCHAR(255),
25     ... );
```

The 'Output' tab at the bottom shows the execution results:

| # | Time | Action | Message | Duration / Fetch |
|---|----------|---|-------------------|-----------------------|
| 1 | 10:49:18 | USE SupermartDW | 0 row(s) affected | 0.047 sec |
| 2 | 10:49:25 | SELECT C.name AS customer_name, - Cust... 188 row(s) returned | | 0.032 sec / 0.000 sec |
| 3 | 12:07:08 | USE SupermartDB | 0 row(s) affected | 0.047 sec |

6.2 Data Staging for Transformation

To maintain data integrity before inserting records into the main tables, staging tables were created.



The screenshot shows the MySQL Workbench interface with the 'ELProcess' tab selected. The central pane displays the following SQL script:

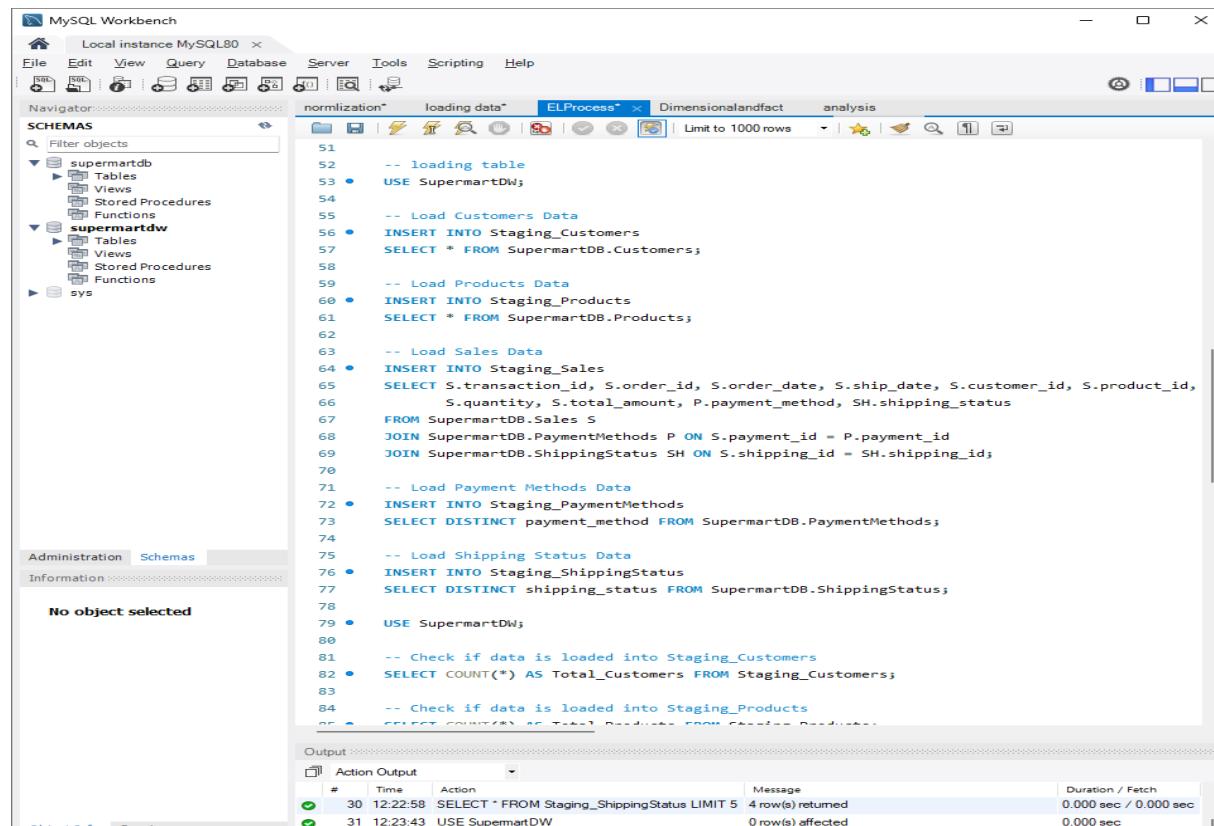
```
1 -- Create a new database for the ETL process
2 CREATE DATABASE SupermartDW;
3 USE SupermartDW;
4
5 -- Staging Table for Customers
6 CREATE TABLE Staging_Customers (
7     customer_id VARCHAR(50),
8     name VARCHAR(255),
9     email VARCHAR(255),
10    phone VARCHAR(50),
11    city VARCHAR(100),
12    state VARCHAR(100),
13    zip_code VARCHAR(20),
14    registration_date DATE,
15    loyalty_score DECIMAL(5,2)
16 );
17
18 -- Staging Table for Products
19 CREATE TABLE Staging_Products (
20     product_id VARCHAR(50),
21     product_name VARCHAR(255),
22     category VARCHAR(100),
23     price DECIMAL(10,2),
24     brand VARCHAR(100),
25     supplier VARCHAR(255)
26 );
27
28 -- Staging Table for Sales
29 CREATE TABLE Staging_Sales (
30     transaction_id VARCHAR(50),
31     order_id VARCHAR(50),
32     order_date DATE,
33     ship_date DATE,
34     customer_id VARCHAR(50),
35     product_id VARCHAR(50)
```

The 'Output' tab at the bottom shows the execution results:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 30 | 12:22:58 | SELECT * FROM Staging_ShippingStatus LIMIT 5 | 4 row(s) returned | 0.000 sec / 0.000 sec |
| 31 | 12:23:43 | USE SupermartDW | 0 row(s) affected | 0.000 sec |

6.3 Data Transformation & Loading

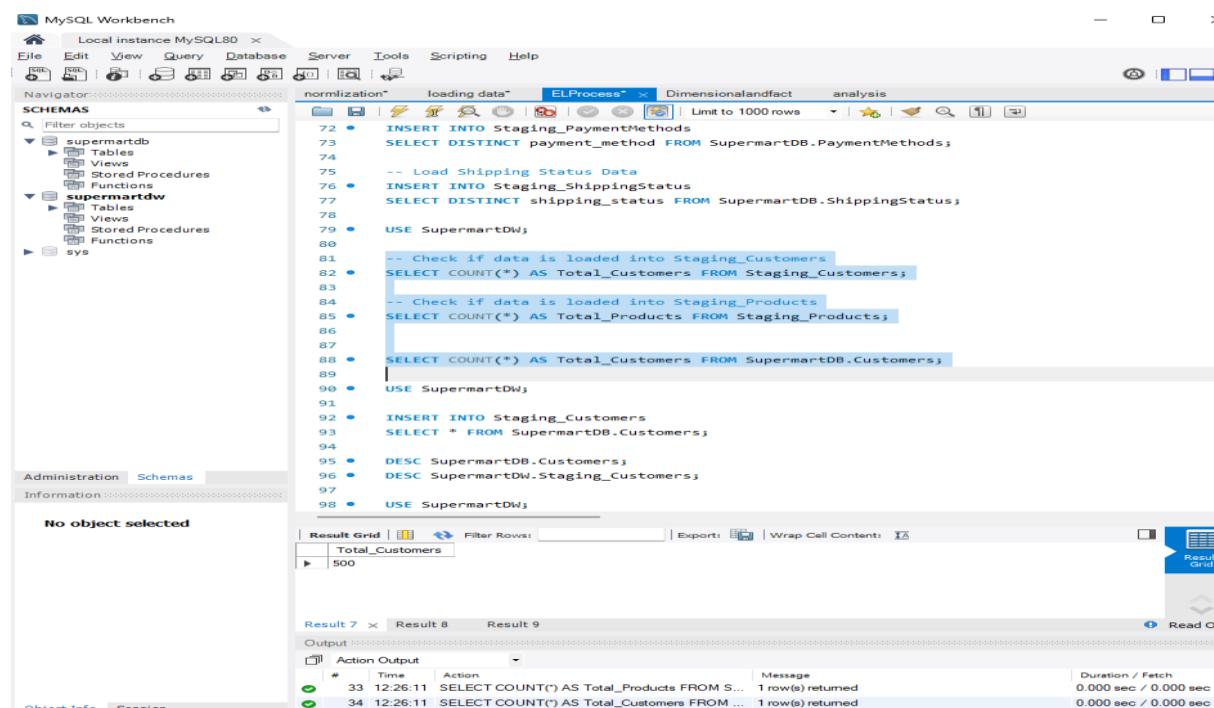
The data from the staging tables underwent transformation processes, ensuring referential integrity and correctness before being inserted into the normalized tables.



This screenshot shows the MySQL Workbench interface with the 'ELProcess' tab selected. The central area displays an ETL script for loading data into a normalized database. The script includes code to use the SupermartDW database, insert data into Staging_Customers, Staging_Products, and Staging_Sales, and to check if data is loaded into these tables. The 'Output' pane at the bottom shows the execution results, including the count of rows returned for each query.

```
51 -- loading table
52 USE SupermartDW;
53
54
55 -- Load Customers Data
56 INSERT INTO Staging_Customers
57 SELECT * FROM SupermartDB.Customers;
58
59 -- Load Products Data
60 INSERT INTO Staging_Products
61 SELECT * FROM SupermartDB.Products;
62
63 -- Load Sales Data
64 INSERT INTO Staging_Sales
65 SELECT S.transaction_id, S.order_id, S.order_date, S.ship_date, S.customer_id, S.product_id,
66 S.quantity, S.total_amount, P.payment_method, SH.shipping_status
67 FROM SupermartDB.Sales S
68 JOIN SupermartDB.PaymentMethods P ON S.payment_id = P.payment_id
69 JOIN SupermartDB.ShippingStatus SH ON S.shipping_id = SH.shipping_id;
70
71 -- Load Payment Methods Data
72 INSERT INTO Staging_PaymentMethods
73 SELECT DISTINCT payment_method FROM SupermartDB.PaymentMethods;
74
75 -- Load Shipping Status Data
76 INSERT INTO Staging_ShippingStatus
77 SELECT DISTINCT shipping_status FROM SupermartDB.ShippingStatus;
78
79 USE SupermartDW;
80
81 -- Check if data is loaded into Staging_Customers
82 SELECT COUNT(*) AS Total_Customers FROM Staging_Customers;
83
84 -- Check if data is loaded into Staging_Products
85 SELECT COUNT(*) AS Total_Products FROM Staging_Products;
```

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 30 | 12:22:58 | SELECT * FROM Staging_ShippingStatus LIMIT 5 | 4 row(s) returned | 0.000 sec / 0.000 sec |
| 31 | 12:23:43 | USE SupermartDW | 0 row(s) affected | 0.000 sec |



This screenshot shows the MySQL Workbench interface with the 'ELProcess' tab selected. The central area displays the same ETL script as the previous screenshot. The 'Result Grid' tab is active, showing the results of the 'SELECT COUNT(*) AS Total_Customers FROM Staging_Customers;' query. The results show a single row with a value of 500. The 'Output' pane at the bottom shows the execution results for the entire script.

| Total_Customers |
|-----------------|
| 500 |

| # | Time | Action | Message | Duration / Fetch |
|----|----------|---|-------------------|-----------------------|
| 33 | 12:26:11 | SELECT COUNT(*) AS Total_Products FROM Staging_Products | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 34 | 12:26:11 | SELECT COUNT(*) AS Total_Customers FROM Staging_Customers | 1 row(s) returned | 0.000 sec / 0.000 sec |

| Total_Products | 1000 |
|----------------|------|
|----------------|------|

| Total_Customers | 500 |
|-----------------|-----|
|-----------------|-----|

7. ETL Process Implementation

The ETL process was carried out in four distinct phases:

7.1 Phase 1: Extraction

- Data was extracted from SupermartDB into staging tables in SupermartDW.
- Staging tables were used to separate operational and analytical environments, ensuring transformations did not affect source data.
- Staging tables were created for each entity: Staging_Customers, Staging_Products, Staging_Sales, Staging_PaymentMethods, and Staging_ShippingStatus.
- INSERT INTO SELECT statements were used to move data from normalized tables to staging tables.

7.2 Phase 2: Dimension Creation

- Data from staging tables was transformed and loaded into four dimension tables:
 - Customer Dimension (dim_customers): Categorized customers into Premium and Standard based on loyalty scores, implemented Slowly Changing Dimension (SCD) Type 2 for tracking changes.
 - Product Dimension (dim_products): Combined product details with denormalized category information and included a derived StockStatus field.
 - Payment Method Dimension (dim_payment_methods): Contained distinct payment methods used in transactions.
 - Shipping Status Dimension (dim_shipping_status): Captured distinct shipping methods from transaction records.
- Surrogate keys were assigned to improve query performance and historical tracking.

MySQL Workbench - Local instance MySQL80

Schemas

- supermartdb
- supermartdw
- sys

Query Editor - Dimensionalandfact*

```

22     product_id VARCHAR(50) UNIQUE NOT NULL,
23     product_name VARCHAR(255),
24     category VARCHAR(100),
25     price DECIMAL(10,2),
26     brand VARCHAR(100),
27     supplier VARCHAR(255)
28   );
29
30 • CREATE TABLE SupermartDW.dim_payment_methods (
31     payment_sk INT AUTO_INCREMENT PRIMARY KEY,
32     payment_method VARCHAR(100) UNIQUE NOT NULL
33   );
34
35 • CREATE TABLE SupermartDW.dim_shipping_status (
36     shipping_sk INT AUTO_INCREMENT PRIMARY KEY,
37     shipping_status VARCHAR(100) UNIQUE NOT NULL
38   );
39
40 • show tables from SupermartDW;
41
42

```

Result Grid

| Tables_in_supermartdw |
|------------------------|
| dim_customers |
| dim_payment_methods |
| dim_products |
| dim_shipping_status |
| fact_sales |
| staging_customers |
| staging_paymentmethods |
| staging_products |
| staging_sales |
| staging_shippingstatus |

Output

| # | Time | Action | Message | Duration / Fetch |
|----|----------|---|--------------------|-----------------------|
| 41 | 13:12:32 | SELECT * FROM SupermartDW.fact_sales LIMIT 5; | 5 row(s) returned | 0.000 sec / 0.000 sec |
| 42 | 13:13:05 | show tables from SupermartDW | 10 row(s) returned | 0.031 sec / 0.000 sec |

Object Info

MySQL Workbench - Local instance MySQL80

Schemas

- supermartdb
- supermartdw
- sys

Query Editor - Dimensionalandfact*

```

89     C.customer_sk,
90     P.product_sk,
91     S.quantity,
92     S.total_amount,
93     PM.payment_sk,
94     SH.shipping_sk
95   FROM SupermartDW.Staging_Sales S
96   JOIN SupermartDW.dim_customers C ON S.customer_id = C.customer_id
97   JOIN SupermartDW.dim_products P ON S.product_id = P.product_id
98   JOIN SupermartDW.dim_payment_methods PM ON S.payment_method = PM.payment_method
99   JOIN SupermartDW.dim_shipping_status SH ON S.shipping_status = SH.shipping_status;
100
101 •  SELECT COUNT(*) FROM SupermartDW.dim_customers;
102 •  SELECT COUNT(*) FROM SupermartDW.dim_products;
103 •  SELECT COUNT(*) FROM SupermartDW.dim_payment_methods;
104 •  SELECT COUNT(*) FROM SupermartDW.dim_shipping_status;
105 •  SELECT COUNT(*) FROM SupermartDW.fact_sales;
106
107 •  SELECT * FROM SupermartDW.fact_sales LIMIT 5;
108
109

```

Result Grid

| COUNT(*) |
|----------|
| 500 |

Output

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 46 | 13:14:16 | SELECT COUNT(*) FROM SupermartDW.dim_s... | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 47 | 13:14:16 | SELECT COUNT(*) FROM SupermartDW.fact_s... | 1 row(s) returned | 0.000 sec / 0.000 sec |

Object Info

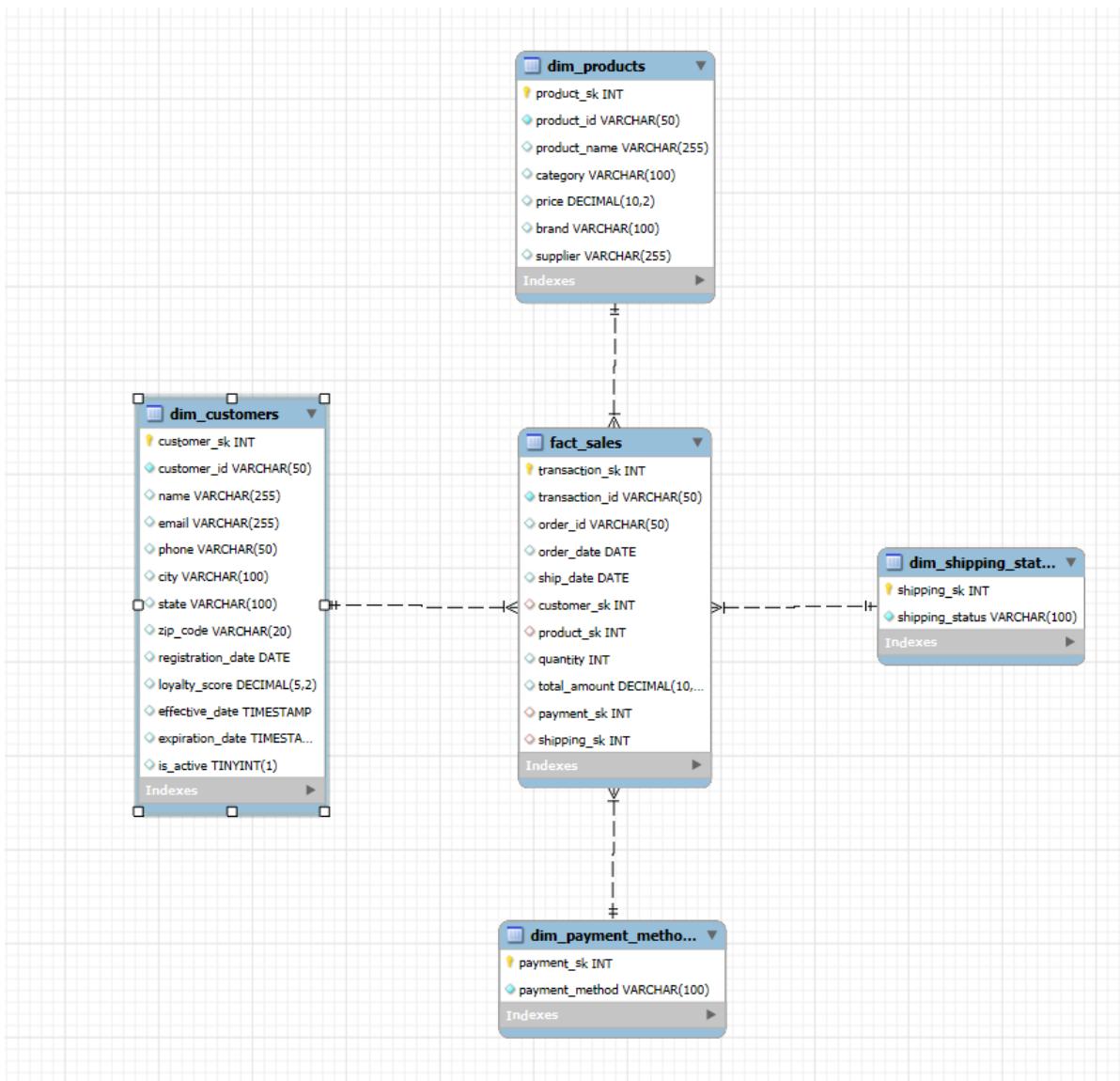
7.3 Phase 3: Fact Table Population

- The fact_sales table was created and populated:
 - Stored transactional details, including transaction_id, order_id, order_date, ship_date, quantity, total_amount, and discounts.
 - Data from staging tables was joined with dimension tables to assign surrogate keys and derive additional measures.
 - Referential integrity was enforced to maintain correct relationships between fact and dimension tables.

The screenshot shows the MySQL Workbench interface with the Dimensionalandfact tab selected. The query editor contains a complex SQL script for populating the fact_sales table, joining multiple dimension tables (dim_customers, dim_products, dim_payment_methods, dim_shipping_status) with the Staging_Sales table. The result grid displays four rows of data from the fact_sales table, and the output pane shows two log entries for the executed queries.

```
transaction_sk transaction_id order_id order_date ship_date customer_sk product_sk quantity total_amount
1 064e97f5 42c7de49 2023-09-24 2023-11-17 222 12 5 963.85
2 11087e17 8bd9959db 2023-04-16 2025-01-21 68 310 4 605.52
3 11e99ec4 d44afdfde 2023-11-14 2024-11-14 241 173 1 368.01
4 13d3743a d2bae51d 2023-05-13 2024-01-21 464 92 4 661.46
```

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 40 | 13:12:10 | SELECT COUNT(*) FROM SupermartDW.fact_s... | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 41 | 13:12:32 | SELECT * FROM SupermartDW.fact_sales LIMI... | 5 row(s) returned | 0.000 sec / 0.000 sec |



7.4 Phase 4: Verification

- **Dimension Table Checks:**
 - Verified record counts and correctness of data categorization.
 - Ensured **Slowly Changing Dimensions (SCD Type 2)** were implemented correctly.
- **Fact Table Validation:**
 - Ensured accuracy of aggregated sales data, discount calculations, and total revenue figures.
- **Data Quality Audits:**
 - Checked for **NULL values** in critical fields.

- Ensured **referential integrity** between dimensions and fact tables.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema tree with databases `supermarketdb`, `supermartdw`, and `sys`.
- Query Editor:** Displays ETL scripts for verifying data in the `SupermartDW` database. The script includes:


```

102 •   SELECT COUNT(*) FROM SupermartDW.dim_products;
103 •   SELECT COUNT(*) FROM SupermartDW.dim_payment_methods;
104 •   SELECT COUNT(*) FROM SupermartDW.dim_shipping_status;
105 •   SELECT COUNT(*) FROM SupermartDW.fact_sales;
106
107 •   SELECT * FROM SupermartDW.fact_sales LIMIT 5;
108
109
110
111    -- Verification for ETL
112
113 •   SELECT customer_id, COUNT(*)
      FROM SupermartDW.Dim_Customers
      GROUP BY customer_id
      HAVING COUNT(*) > 1;
114
115 •   SELECT *
      FROM SupermartDW.Fact_Sales
      WHERE total_amount < 0;
116
117
118
119
120
121
122
      
```
- Result Grid:** Shows the output of the verification query, displaying a single row with `customer_id` and `COUNT(*)`.
- Output:** Shows the log of actions taken, with two entries:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|-------------------|-----------------------|
| 48 | 13:14:58 | SELECT customer_id, COUNT(*) FROM SupermartDW.Dim_Customers | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 49 | 13:14:58 | SELECT * FROM SupermartDW.Fact_Sales WHERE total_amount < 0; | 0 row(s) returned | 0.000 sec / 0.000 sec |
- Result Grid:** Shows the output of the `SELECT * FROM SupermartDW.Fact_Sales` query, displaying columns: `transaction_sk`, `transaction_id`, `order_id`, `order_date`, `ship_date`, `customer_sk`, `product_sk`, `quantity`, and `total_amount`. All values are currently null.

8. Analytical Queries

1. Most Preferred Payment Methods

Overview:

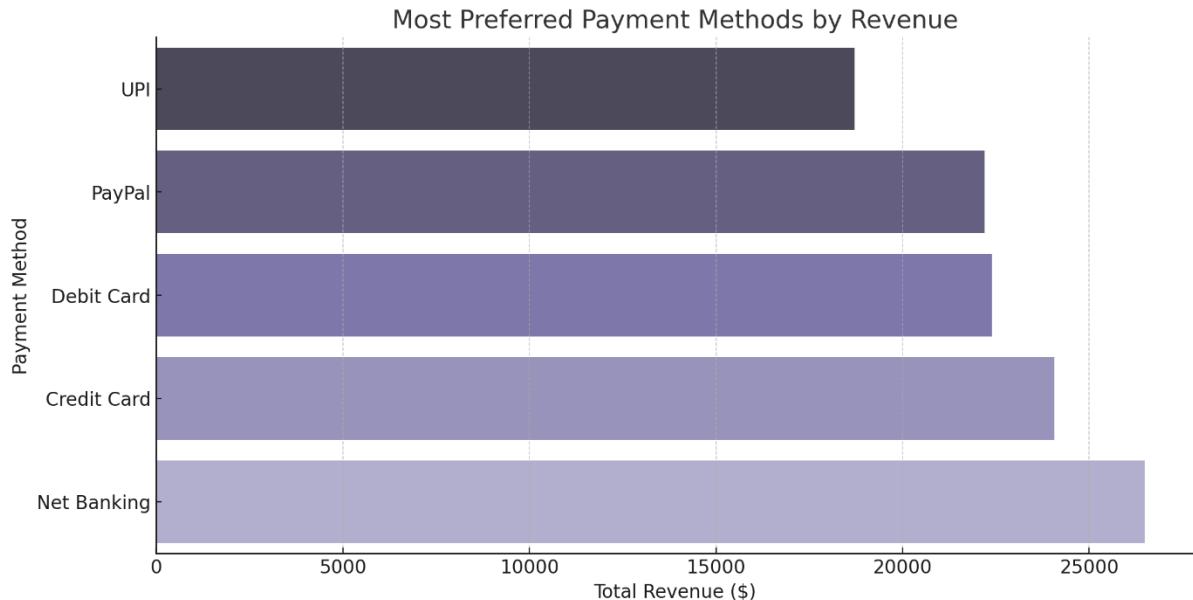
This query identified the most preferred payment methods based on total revenue generated and transaction count.

Insights:

- Net Banking was the most used method with 54 transactions, generating \$26,482.79 in revenue.
- Credit Cards and Debit Cards were also popular, collectively contributing \$46,459.72.
- Digital wallets (PayPal & UPI) accounted for a significant portion of transactions, indicating customer preference for online payment methods.

Recommendations:

- Offer discounts or cashback on high-value transactions via preferred payment methods.
- Optimize the checkout experience by prioritizing fast and secure digital payment options.



2. Monthly Sales Trends

Overview:

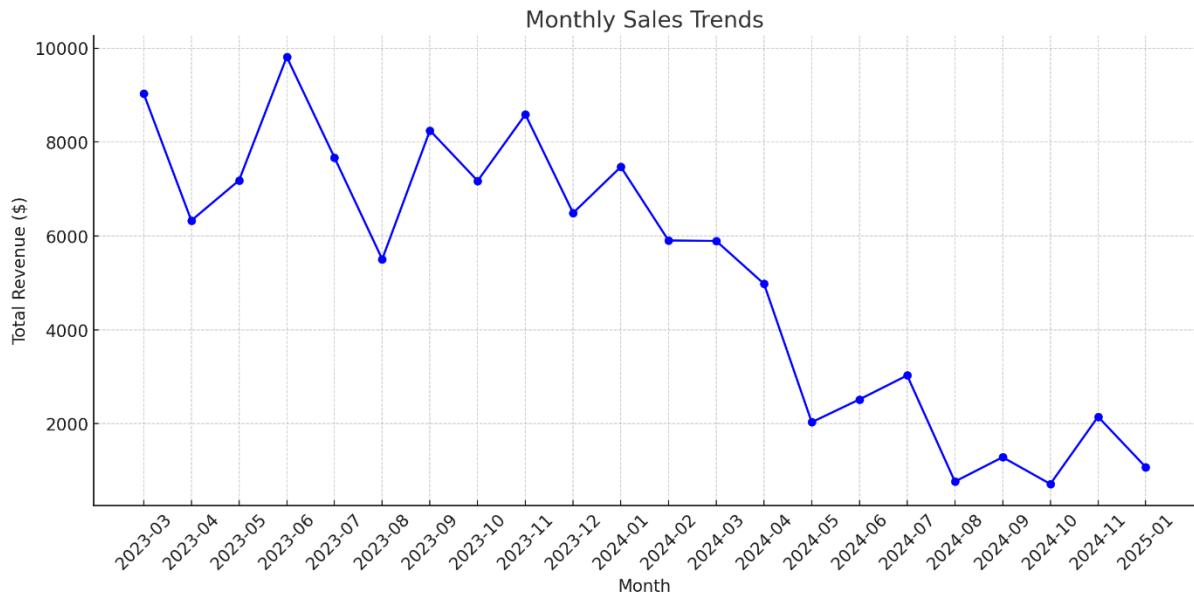
This query extracted **monthly sales revenue trends** to analyze seasonal variations and sales performance.

Insights:

- The highest revenue month was **June 2023 (\$9,812.35)**, followed by **March 2023 (\$9,035.23)**.
- Sales fluctuated over the months, with a noticeable drop in **August 2024 (\$768.54)**.
- Significant revenue declines in **late 2024** may indicate **seasonal demand changes or external factors**.

Recommendations:

- Strengthen **marketing efforts** during low-revenue months to maintain sales consistency.
- Prepare for peak sales months with **inventory adjustments and promotional strategies**.



3. Top-Selling Products

Overview:

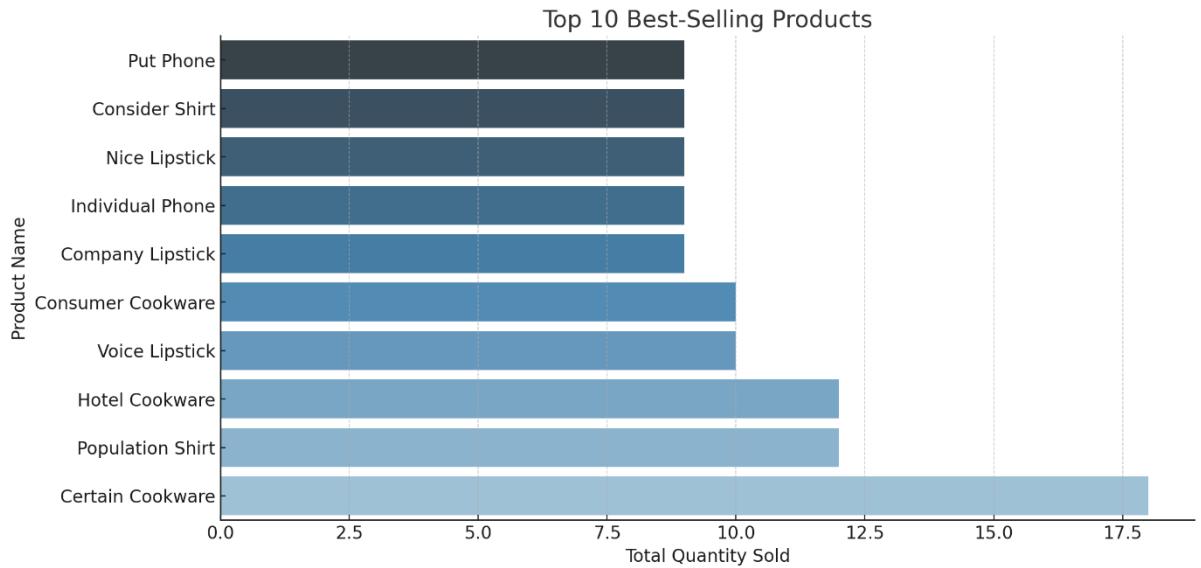
This query determined the **best-selling products** by aggregating **total quantity sold**.

Insights:

- **Certain Cookware** was the best-selling product with **18 units sold**.
- Clothing, electronics, and kitchen products were among the most frequently purchased categories.
- Some products had **low sales performance**, indicating potential for improvement via promotions.

Recommendations:

- Ensure **high-demand products are adequately stocked**.
- Use **cross-selling techniques** to bundle less popular products with high-demand items.



4. Customer Purchase Behaviour

Overview:

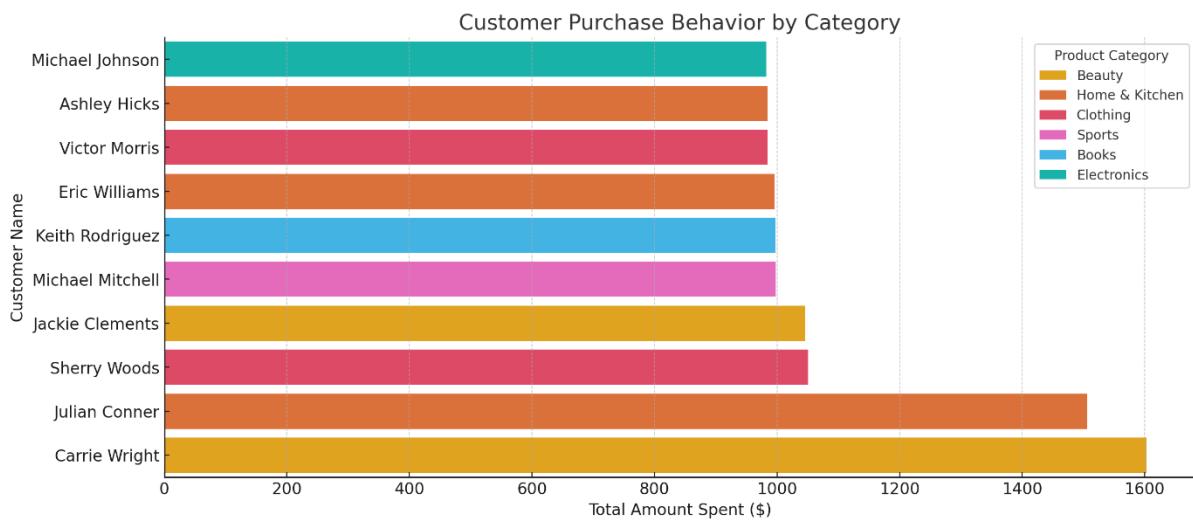
This query analysed **customer spending behaviour** across different **product categories**.

Insights:

- **Carrie Wright spent the highest amount (\$1,603.10) on beauty products**, followed by **Julian Conner (\$1,506.28) on home & kitchen items**.
- Customers showed strong purchasing patterns in categories like **beauty, electronics, and home & kitchen**.
- Some customers made multiple purchases within the same category, indicating **brand loyalty or recurring needs**.

Recommendations:

- Implement **personalized marketing** for customers based on their **preferred categories**.
- Introduce **membership programs** to reward high-spending customers and encourage repeat purchases.



5. Total Sales per Customer

Overview:

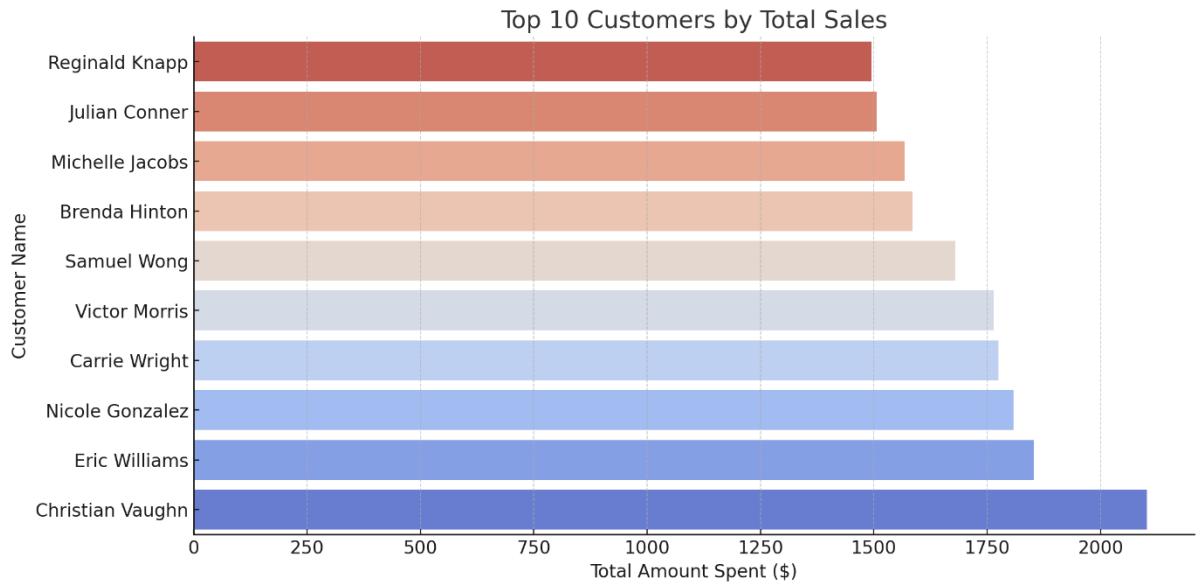
This query identified the **highest-spending customers** to understand purchasing power.

Insights:

- **Christian Vaughn** was the **top spender** with **\$2,102.08**, followed by **Eric Williams** (**\$1,853.55**).
- High-value customers **significantly impact total revenue**, making them ideal targets for **premium offers and loyalty programs**.
- There was a **wide range of customer spending**, suggesting **different consumer segments**.

Recommendations:

- Develop **customer tiers** (e.g., VIP, frequent buyers) to **offer exclusive deals and rewards**.
- Encourage **low-spending customers** to increase purchases through **personalized recommendations and targeted ads**.



9. Recommendations for Future Improvements

Based on the analysis of total sales per customer, top-selling products, preferred payment methods, monthly sales trends, and customer purchase behavior, the following improvements can be made:

9.1 Enhancing Customer Segmentation and Loyalty Programs

Current Observations:

- Some customers contribute significantly to sales, while others have lower spending.
- High-value customers tend to buy frequently from specific categories.

Recommendations:

- Implement **customer segmentation** (e.g., VIP, Regular, Occasional Buyers).
- Offer **personalized discounts, cashback, and loyalty points** for high-value customers.
- **Encourage repeat purchases** from mid-tier customers by sending **targeted promotions and recommendations**.

9.2 Optimizing Inventory and Stock Management

Current Observations:

- Certain products (like **cookware and clothing**) are **top sellers**, while others underperform.
- Stockouts may lead to missed revenue opportunities.

Recommendations:

- Automate inventory tracking to ensure **top-selling products are always available**.
- Offer **discounts or bundling** for slow-moving products to clear out excess stock.
- Use **predictive analytics** to forecast **demand trends** and optimize restocking.

9.3 Improving Payment and Checkout Experience

Current Observations:

- **Net Banking, Credit Cards, and PayPal** generate the most revenue.
- Some payment methods are underutilized.

Recommendations:

- Promote **popular payment methods** by offering incentives (e.g., cashback on credit card purchases).
- Optimize the **checkout experience** to reduce cart abandonment and improve conversions.
- Introduce **buy now, pay later (BNPL)** options to attract higher-value purchases.

9.4 Seasonal and Monthly Sales Planning

Current Observations:

- Sales fluctuate significantly, with some months showing a **major decline**.
- Peak sales months indicate **seasonal trends**.

Recommendations:

- Run **promotions during low-sales months** to encourage spending.
- Leverage **peak seasons** with targeted ad campaigns and limited-time offers.
- Introduce **seasonal products and special bundles** to capitalize on demand trends.

9.5 Data-Driven Business Strategies

Current Observations:

- Insights from customer spending and product trends can **drive better decision-making**.

Recommendations:

- Implement **machine learning models** to **predict customer preferences and demand**.
- Use **real-time analytics dashboards** for tracking sales performance.
- Continuously refine **pricing strategies** based on competitor analysis and market trends.

Conclusion

The **Dimensional Modeling Implementation for Target E-Commerce** successfully integrated **three structured datasets**—Customers, Products, and Sales—into a **star-schema-based data warehouse (SupermartDW)**. The **ETL process** ensured data consistency, integrity, and accessibility for analytical queries, supporting **business intelligence and decision-making**.

Key Achievements:

Optimized Data Warehouse: Implemented a **structured ETL process** in MySQL, ensuring seamless data integration and transformation.

Enhanced Analytical Capabilities: Developed insightful **business intelligence queries** to analyze **customer spending, product sales trends, and payment preferences**.

Improved Business Insights: Identified **high-value customers, top-selling products, preferred payment methods, and seasonal sales trends** to drive data-driven strategies.

Effective Inventory & Payment Management: Analyzed product demand and payment trends to optimize **stock management and checkout experience**.

Scalable and Future-Ready System: Designed a **scalable warehouse** that can support future enhancements like **real-time analytics, predictive modeling, and AI-driven recommendations**.

Future Scope

- ◆ Implement **Machine Learning (ML) models** for **sales forecasting and demand prediction**.
- ◆ Integrate **real-time ETL processing** to support **instant decision-making**.
- ◆ Enhance **personalized marketing strategies** through **customer segmentation and behavioural analytics**.
- ◆ Optimize **supply chain efficiency** by linking sales insights with logistics and supplier data.

By leveraging the structured **dimensional model and analytical insights**, the business can improve **customer engagement, sales performance, and operational efficiency**, ensuring a **competitive advantage in the e-commerce industry**.