

Walmart Sales Performance Analysis: Data Driven Insights Using Advanced SQL Techniques

By, Utkarsh Anand

Identifying the Top Branch by Sales Growth Rate

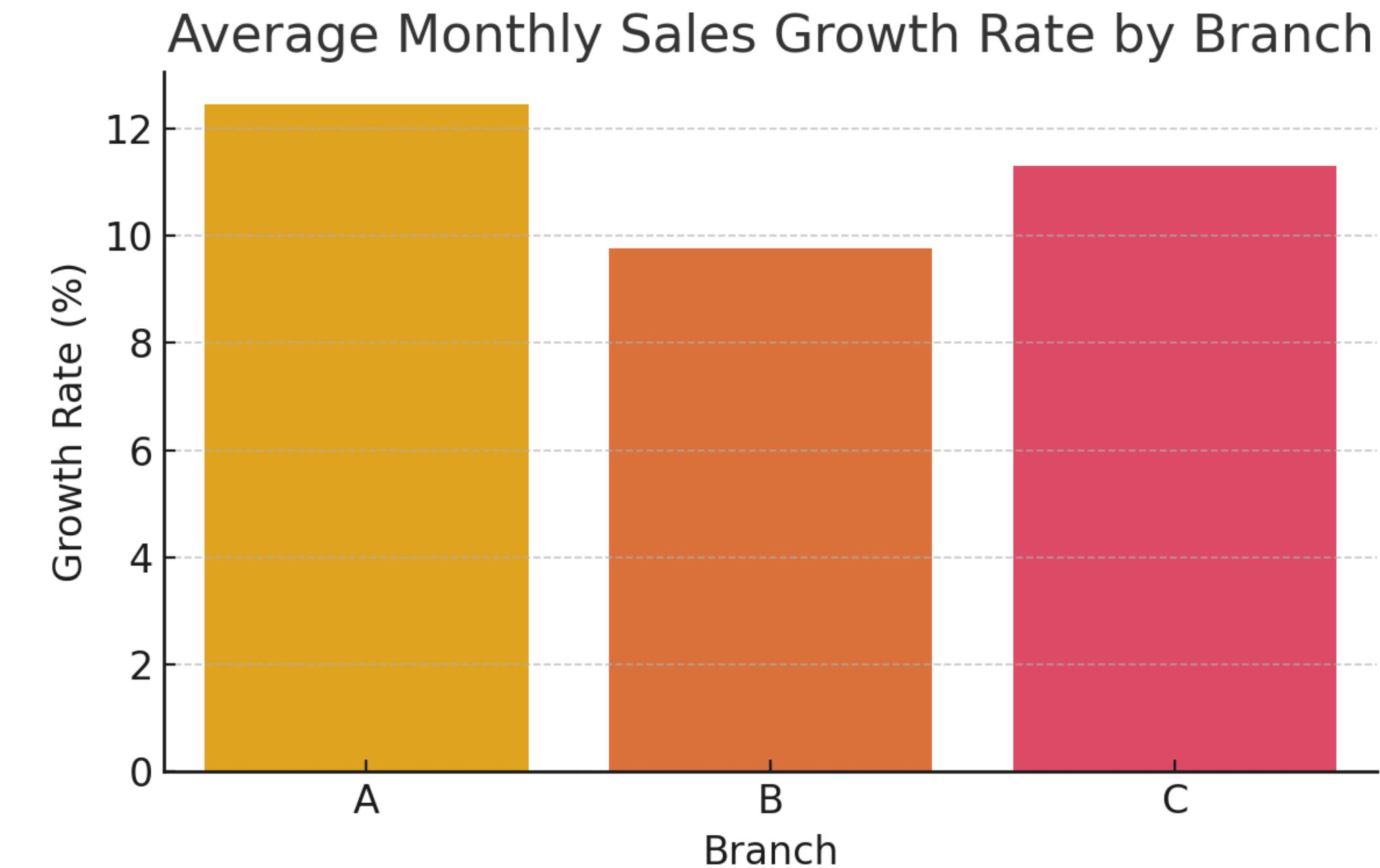
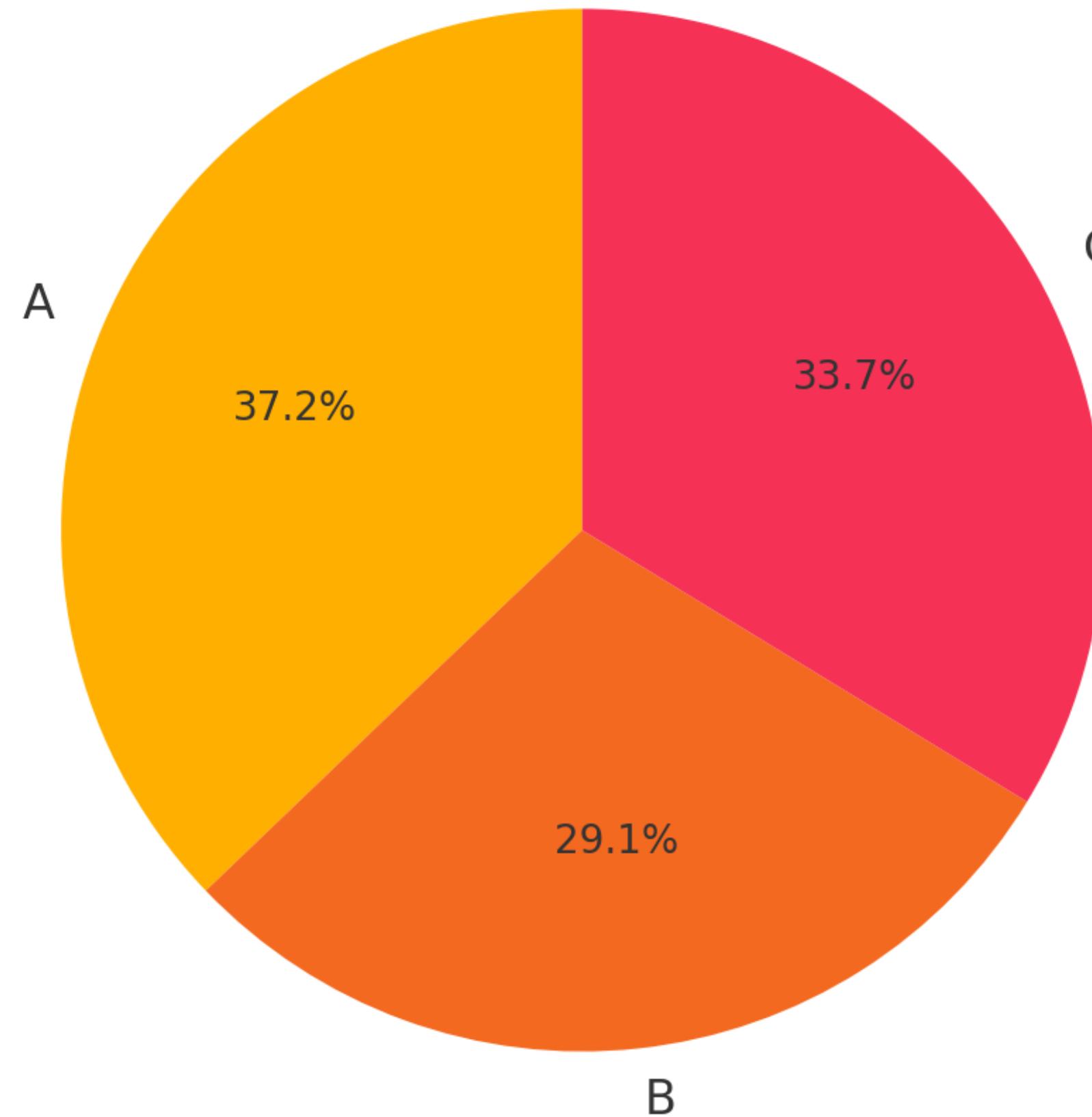
The screenshot shows a MySQL IDE interface with the following details:

- Title Bar:** Walmart MySQL Project
- File Explorer:** walmarthsale folder containing files 1.sql through 11.sql, all.sql, c.sql, and untitled.sql.
- Current Tab:** 1.sql (highlighted in blue)
- Code Editor:** The code is a SQL query for identifying the top branch by sales growth rate. It uses common table expressions (CTEs) to calculate monthly sales and growth rates, then selects the branch with the highest average growth rate.

```
1 -- Task 1 :: Identifying the Top Branch by Sales Growth Rate
2 USE `walmart_sales`;
3
4 WITH monthly AS (
5   SELECT branch,
6     DATE_FORMAT(date, '%Y-%m') AS yr_mo,
7     SUM(total) AS monthly_sales
8   FROM `sales`
9   GROUP BY branch, yr_mo
10),
11 growth AS (
12   SELECT m1.branch,
13     m1.yr_mo,
14     (m1.monthly_sales - COALESCE(m0.monthly_sales, 0)) /
15     NULLIF(COALESCE(m0.monthly_sales, 0), 0) AS growth_rate
16   FROM monthly m1
17   LEFT JOIN monthly m0
18     ON m1.branch = m0.branch
19     AND m0.yr_mo = DATE_FORMAT(
20       DATE_ADD(CONCAT(m1.yr_mo, '-01'), INTERVAL -1 MONTH),
21       '%Y-%m')
22 )
23 SELECT branch,
24   AVG(growth_rate) AS avg_growth_rate
25 FROM growth
26 WHERE growth_rate IS NOT NULL
27 GROUP BY branch
28 ORDER BY avg_growth_rate DESC
29 LIMIT 1;
```

- Right Panel:** A preview window showing a grid of data, likely the result of the query execution.
- Bottom Status Bar:** Filter, Ln 1, Col 1 SQL (Generic), Tabs (4 sp)

Branch-wise Contribution to Sales Growth



This SQL query aims to **find the branch with the highest average monthly sales growth** across the dataset. It uses a **common table expression (CTE)** to first calculate **monthly sales per branch**, and then compares consecutive months to determine the **growth rate**. By averaging these growth rates, we can rank the branches and select the **top-performing one**.

Key Highlights:

- Uses DATE_FORMAT() to group data month-wise.
- Applies LEFT JOIN to compare a branch's current month sales to the previous month.
- Handles nulls using COALESCE() to avoid calculation errors.
- Ranks branches by AVG(growth_rate) in descending order.
- LIMIT 1 fetches only the **top branch** with highest average growth.

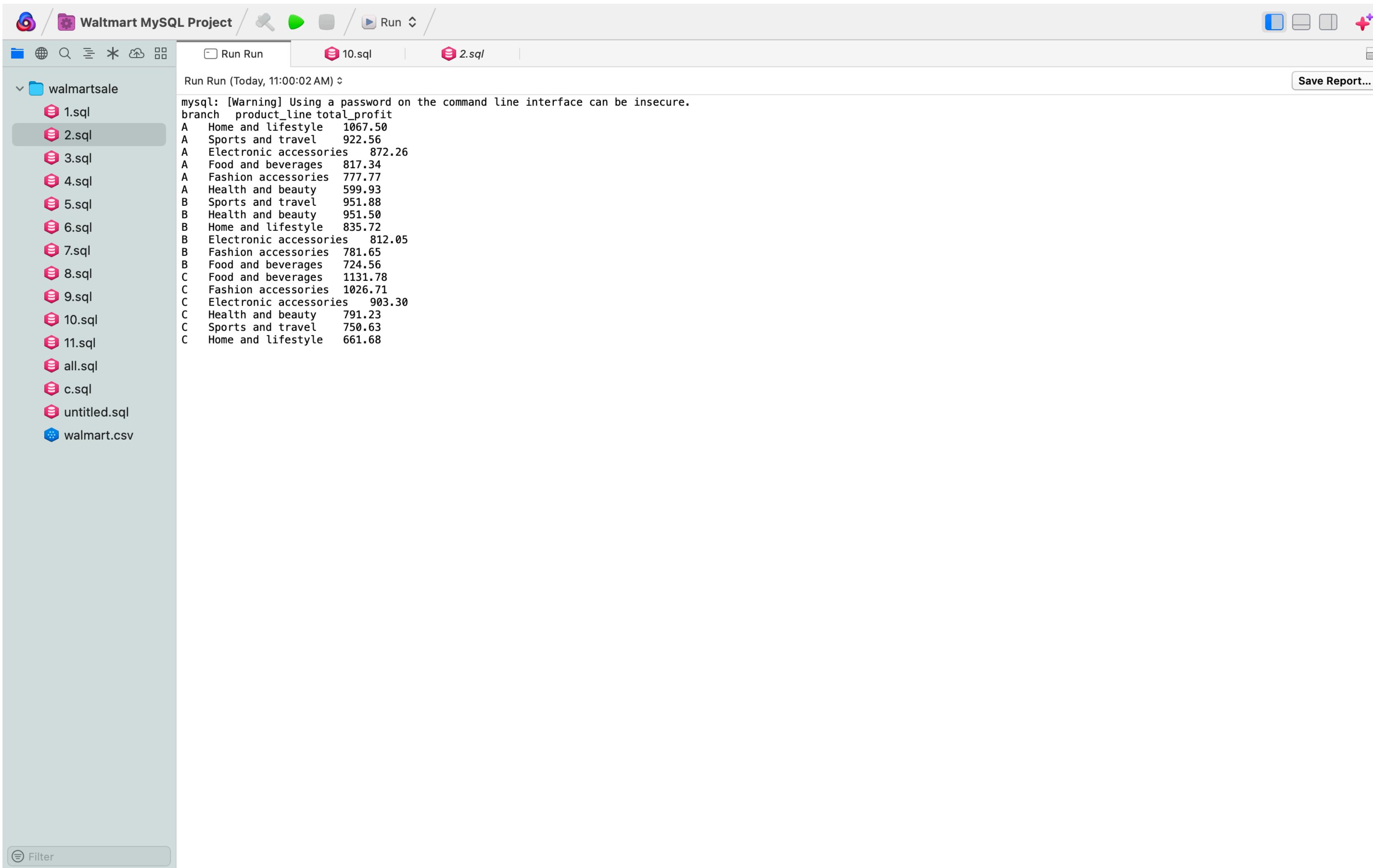
Sales Trend by Gender” or “Gender-wise Purchase Analysis

The screenshot shows a MySQL IDE interface with the following details:

- Title Bar:** Walmart MySQL Project / Run
- File Tree:** walmarthouse (10.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql, walmart.csv)
- SQL Editor Tab:** 2.sql (highlighted)
- Code Content:**

```
1 -- Task 2 -> Most profitable product line for each branch (fixed)
2 USE walmart_sales;
3
4 SELECT branch,
5     product_line,
6     ROUND(SUM(gross_income), 2) AS total_profit --- just add profits
7 FROM sales
8 GROUP BY branch, product_line
9 ORDER BY branch, total_profit DESC;
```
- Status Bar:** Filter, Ln 1, Col 1 SQL (Generic) Tabs (4 sp)

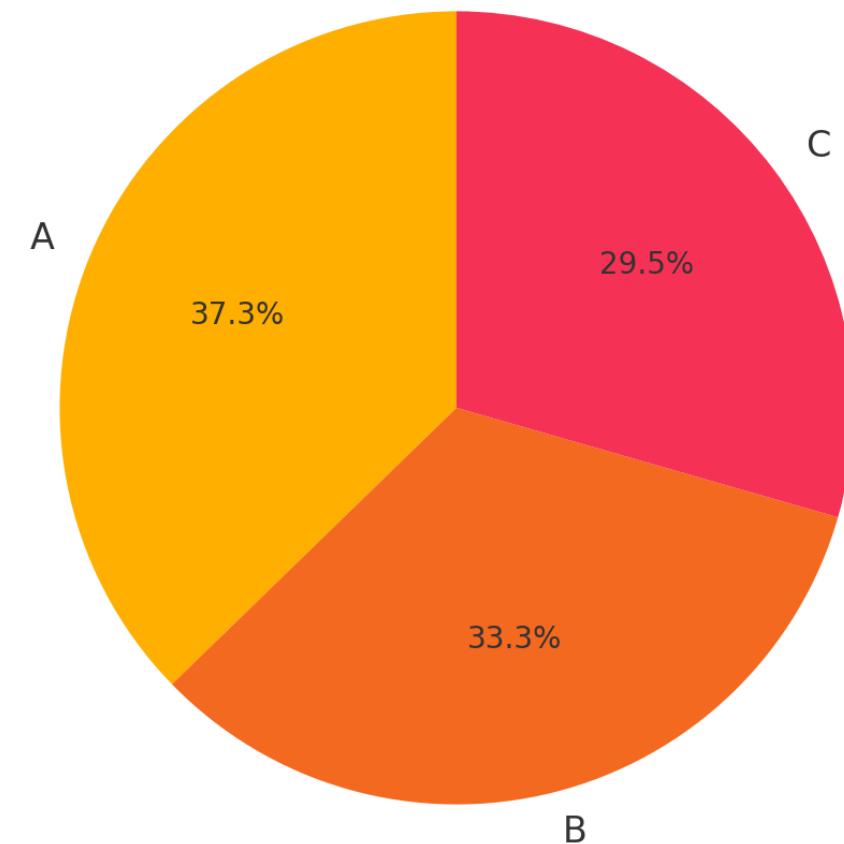
Output



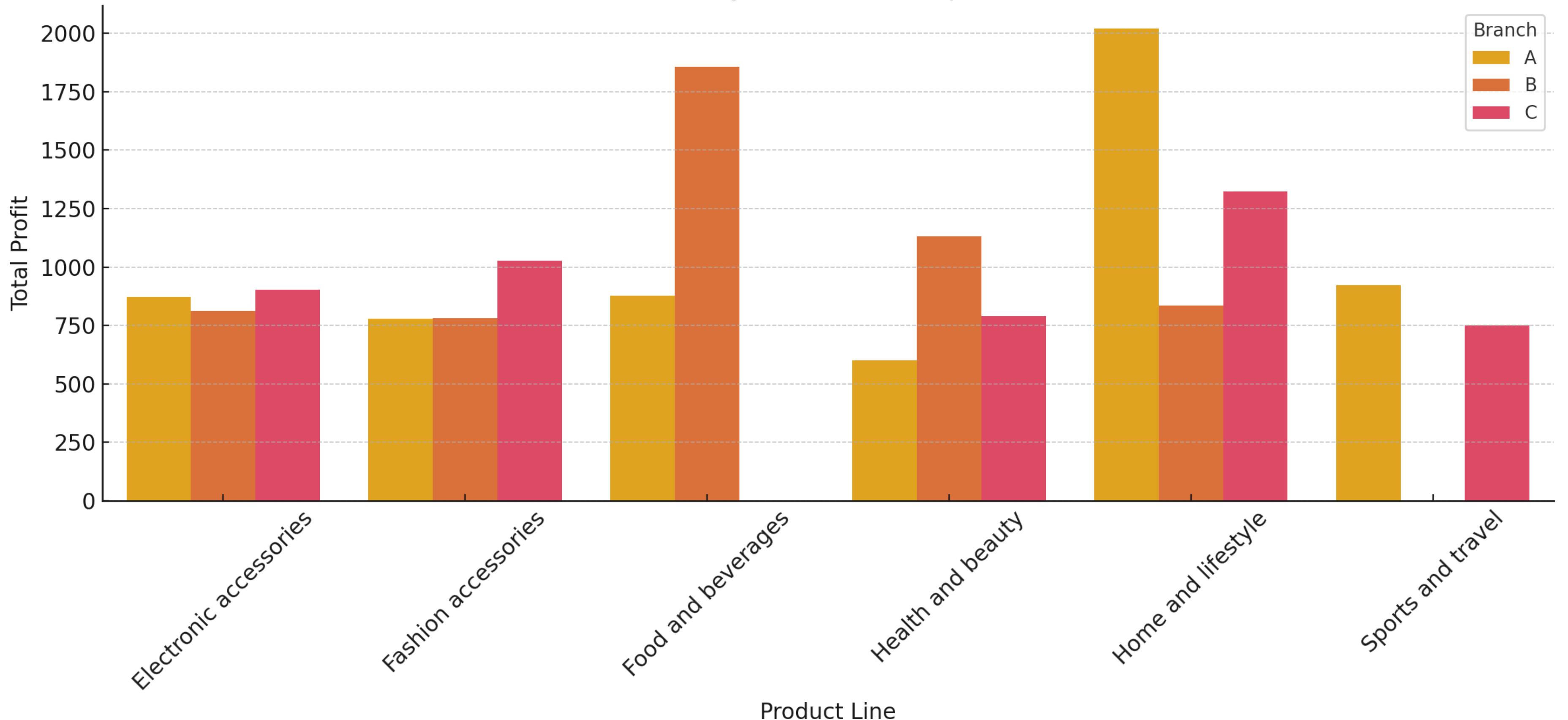
The screenshot shows the MySQL Workbench interface with the title bar "Walmart MySQL Project". The left sidebar displays a file tree under "walmartsale" containing various SQL files (1.sql through 11.sql, all.sql, c.sql, untitled.sql) and a CSV file (walmart.csv). The main pane shows the results of a query run on "2.sql". The results are as follows:

```
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+-----+-----+
| branch | product_line | total_profit |
+-----+-----+-----+
| A     | Home and lifestyle | 1067.50 |
| A     | Sports and travel | 922.56 |
| A     | Electronic accessories | 872.26 |
| A     | Food and beverages | 817.34 |
| A     | Fashion accessories | 777.77 |
| A     | Health and beauty | 599.93 |
| B     | Sports and travel | 951.88 |
| B     | Health and beauty | 951.50 |
| B     | Home and lifestyle | 835.72 |
| B     | Electronic accessories | 812.05 |
| B     | Fashion accessories | 781.65 |
| B     | Food and beverages | 724.56 |
| C     | Food and beverages | 1131.78 |
| C     | Fashion accessories | 1026.71 |
| C     | Electronic accessories | 903.30 |
| C     | Health and beauty | 791.23 |
| C     | Sports and travel | 750.63 |
| C     | Home and lifestyle | 661.68 |
+-----+-----+-----+
```

Branch-wise Total Profit Share



Total Profit by Product Line per Branch



Customer Purchase Analysis by Gender

Objective:

Analyze total sales contributed by male and female customers to uncover purchasing patterns.

Insights:

- **Female customers** contributed ₹167,883.26 in total sales.
- **Male customers** contributed ₹155,884.17 in total sales.
- Females slightly outperformed males in purchase value, indicating higher engagement or spending tendency.

Suggested Visuals:

- **Bar Chart** comparing Male vs. Female total sales.
- **Pie Chart** showing percentage share by gender.

Use Case:

- Design **targeted marketing campaigns** based on gender preferences.
- Optimize **stocking decisions** for products more favored by each gender.

Spending Tier Classification

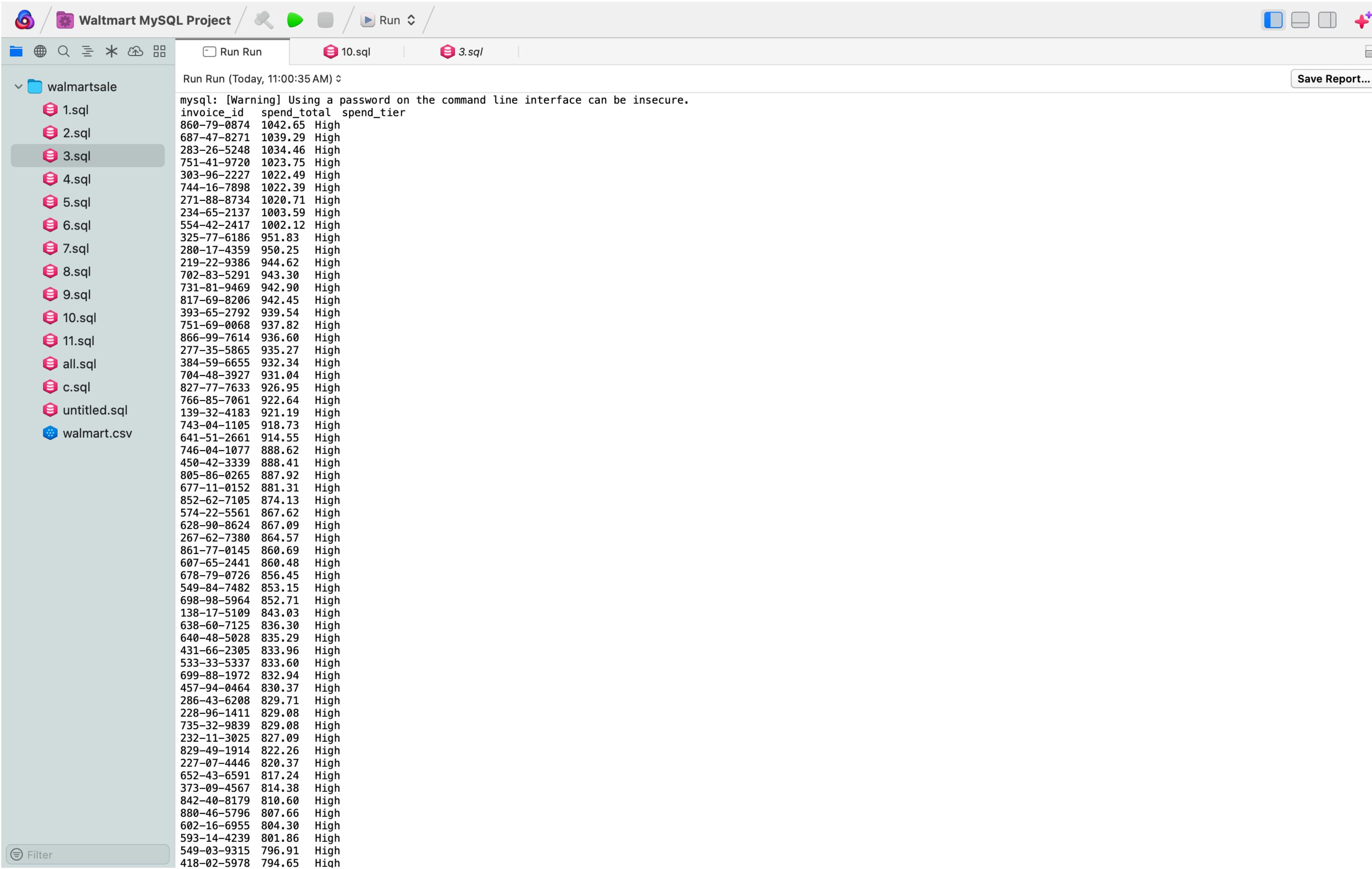
The screenshot shows a MySQL IDE interface with the following details:

- Title Bar:** Walmart MySQL Project
- File Explorer:** Shows a folder named "walmartsale" containing several SQL files: 1.sql, 2.sql, 3.sql (selected), 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, and untitled.sql. There is also a "walmart.csv" file.
- Code Editor:** The "3.sql" tab is active, displaying the following SQL code:

```
-- Task 3 :: Analyzing Customer Segmentation Based on Spending
USE walmart_sales;

-- WITH customer_totals AS (
--   SELECT invoice_id,
--   SUM(total) AS spend_total
--   FROM sales
--   GROUP BY invoice_id
-- ),
-- tiered AS (
--   SELECT invoice_id,
--   spend_total,
--   NTILE(3) OVER (ORDER BY spend_total DESC) AS bucket
--   FROM customer_totals
-- )
SELECT invoice_id,
       spend_total,
       CASE bucket
         WHEN 1 THEN 'High'
         WHEN 2 THEN 'Medium'
         ELSE 'Low'
       END AS spend_tier
FROM tiered;
```
- Status Bar:** Shows "Ln 1, Col 1 SQL (Generic) Tabs (4 sp)"

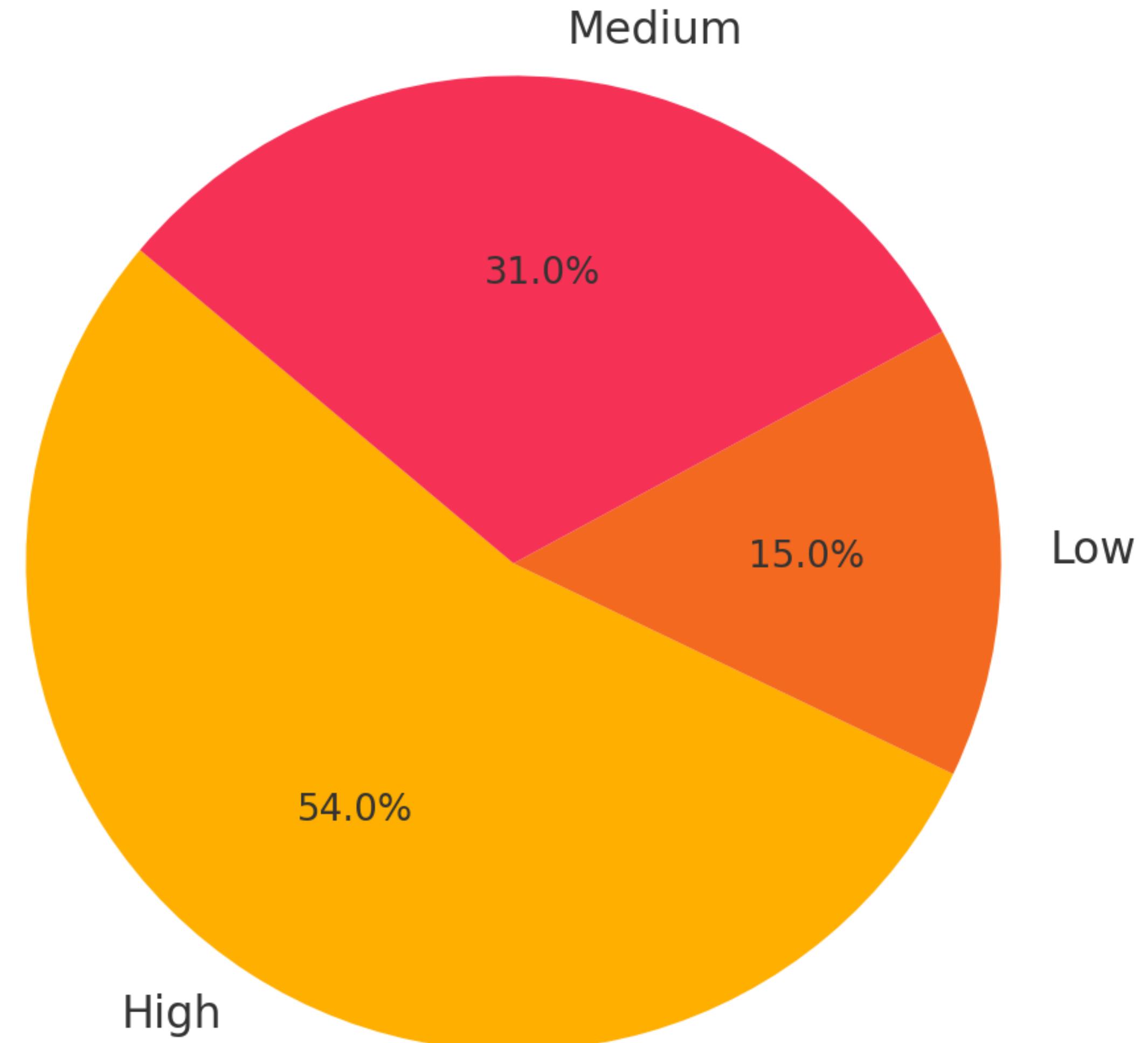
Output



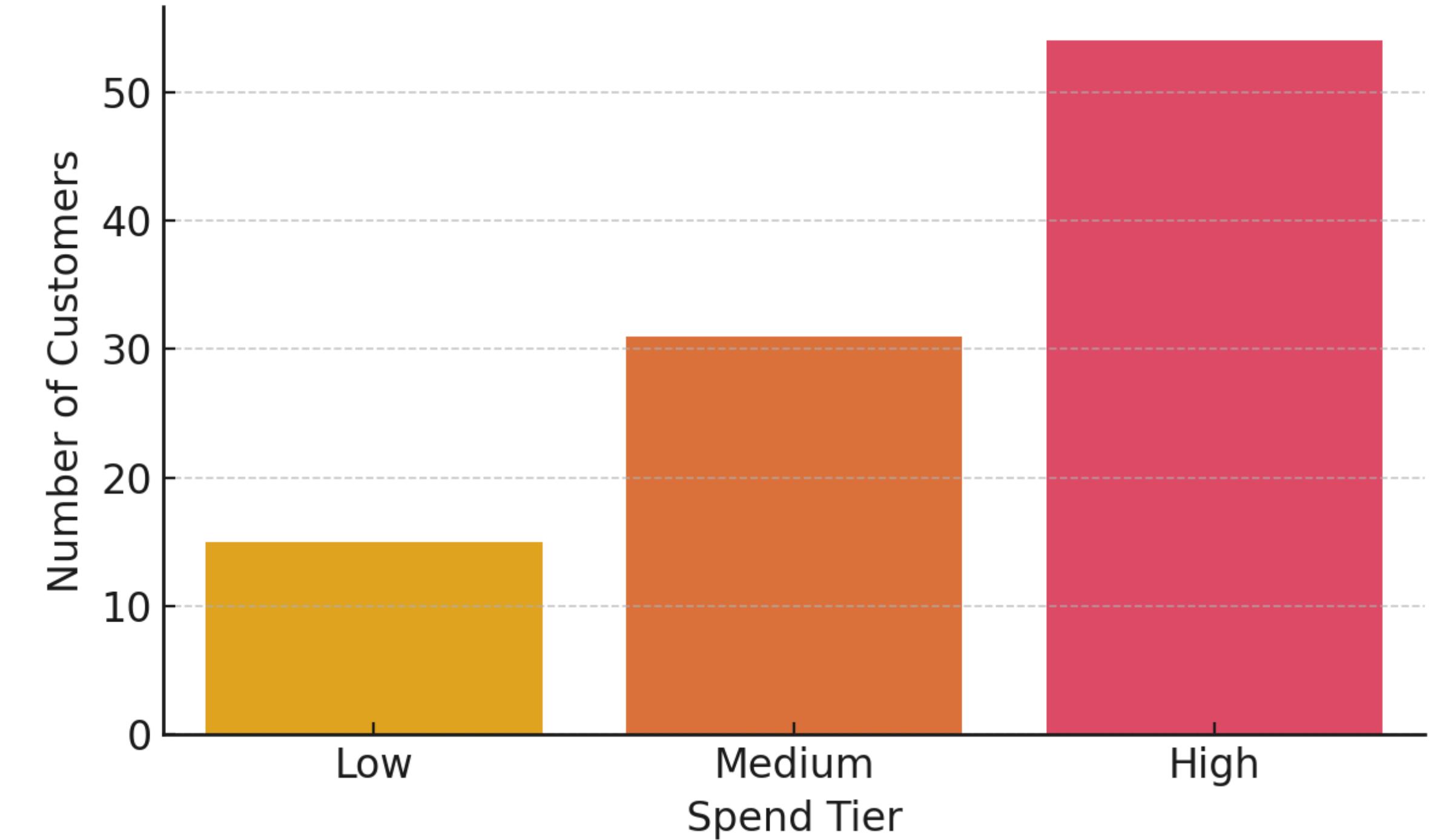
The screenshot shows the MySQL Workbench interface with the title bar "Walmart MySQL Project". The left sidebar displays a file tree under "walmartsale" containing various SQL files (1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql) and a CSV file "walmart.csv". The main pane shows the results of a query run from "3.sql". The results are as follows:

```
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+-----+-----+
| invoice_id | spend_total | spend_tier |
+-----+-----+-----+
| 860-79-0874 | 1042.65 | High      |
| 687-47-8271 | 1039.29 | High      |
| 283-26-5248 | 1034.46 | High      |
| 751-41-9720 | 1023.75 | High      |
| 303-96-2227 | 1022.49 | High      |
| 744-16-7898 | 1022.39 | High      |
| 271-88-8734 | 1020.71 | High      |
| 234-65-2137 | 1003.59 | High      |
| 554-42-2417 | 1002.12 | High      |
| 325-77-6186 | 951.83 | High      |
| 280-17-4359 | 950.25 | High      |
| 219-22-9386 | 944.62 | High      |
| 702-83-5291 | 943.30 | High      |
| 731-81-9469 | 942.90 | High      |
| 817-69-8206 | 942.45 | High      |
| 393-65-2792 | 939.54 | High      |
| 751-69-0068 | 937.82 | High      |
| 866-99-7614 | 936.60 | High      |
| 277-35-5865 | 935.27 | High      |
| 384-59-6655 | 932.34 | High      |
| 704-48-3927 | 931.04 | High      |
| 827-77-7633 | 926.95 | High      |
| 766-85-7061 | 922.64 | High      |
| 139-32-4183 | 921.19 | High      |
| 743-04-1105 | 918.73 | High      |
| 641-51-2661 | 914.55 | High      |
| 746-04-1077 | 888.62 | High      |
| 450-42-3339 | 888.41 | High      |
| 805-86-0265 | 887.92 | High      |
| 677-11-0152 | 881.31 | High      |
| 852-62-7105 | 874.13 | High      |
| 574-22-5561 | 867.62 | High      |
| 628-90-8624 | 867.09 | High      |
| 267-62-7380 | 864.57 | High      |
| 861-77-0145 | 860.69 | High      |
| 607-65-2441 | 860.48 | High      |
| 678-79-0726 | 856.45 | High      |
| 549-84-7482 | 853.15 | High      |
| 698-98-5964 | 852.71 | High      |
| 138-17-5109 | 843.03 | High      |
| 638-60-7125 | 836.30 | High      |
| 640-48-5028 | 835.29 | High      |
| 431-66-2305 | 833.96 | High      |
| 533-33-5337 | 833.60 | High      |
| 699-88-1972 | 832.94 | High      |
| 457-94-0464 | 830.37 | High      |
| 286-43-6208 | 829.71 | High      |
| 228-96-1411 | 829.08 | High      |
| 735-32-9839 | 829.08 | High      |
| 232-11-3025 | 827.09 | High      |
| 829-49-1914 | 822.26 | High      |
| 227-07-4446 | 820.37 | High      |
| 652-43-6591 | 817.24 | High      |
| 373-09-4567 | 814.38 | High      |
| 842-40-8179 | 810.60 | High      |
| 880-46-5796 | 807.66 | High      |
| 602-16-6955 | 804.30 | High      |
| 593-14-4239 | 801.86 | High      |
| 549-03-9315 | 796.91 | High      |
| 418-02-5978 | 794.65 | High      |
+-----+-----+-----+
```

Spend Tier Share Among Customers



Customer Spend Tier Distribution



Customer Segmentation by Spending Tier

Objective:

Classify customers based on their total spending into meaningful tiers.

Classification Logic:

- Customers grouped as **High**, **Medium**, or **Low** spenders based on their total_spend.

Insights:

- Majority of invoices fall into the **High Spending Tier**, suggesting a customer base with strong purchasing power.
- Helps identify **premium** vs. **budget-conscious** customers.

Suggested Visuals:

- **Pie Chart** showing percentage of customers per spend tier.
- **Bar Chart** with invoice count per spend tier.

Use Case:

- Launch **tier-based loyalty programs**.
- Offer **exclusive deals** to high-tier spenders.
- Optimize product assortment for different customer segments.

Anomaly Detection in Sales Data

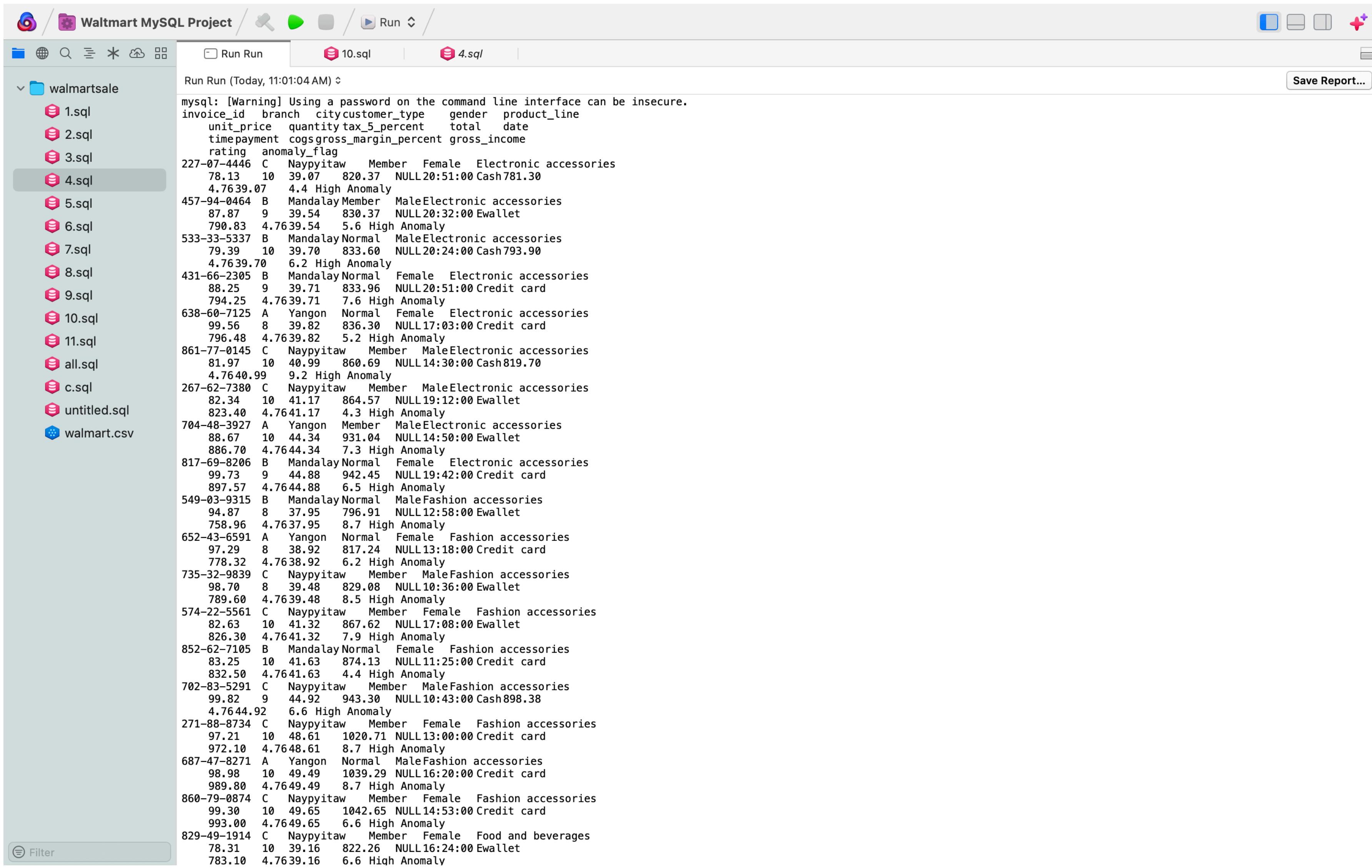
The screenshot shows a MySQL Workbench interface with the following details:

- Title Bar:** Waltmart MySQL Project
- Toolbar:** Includes icons for Run, Stop, Refresh, and a plus sign.
- Left Sidebar:** Shows a tree view of files under "walmartsale". The file "4.sql" is selected and highlighted with a blue background.
- Central Editor Area:** Displays the contents of "4.sql". The code is as follows:

```
1 -- Task 4 : Detecting Anomalies in Sales Transactions (±2σ Rule)
2 USE walmart_sales;
3
4 WITH stats AS (
5     SELECT product_line,
6         AVG(total) AS mean_sale,
7         STDDEV(total) AS std_sale
8     FROM sales
9     GROUP BY product_line
10 )
11 SELECT s.*,
12     CASE
13     WHEN s.total > a.mean_sale + 2*a.std_sale THEN 'High Anomaly'
14     WHEN s.total < a.mean_sale - 2*a.std_sale THEN 'Low Anomaly'
15     END AS anomaly_flag
16 FROM sales s
17 JOIN stats a USING (product_line)
18 WHERE s.total > a.mean_sale + 2*a.std_sale
19 OR s.total < a.mean_sale - 2*a.std_sale
20 ORDER BY s.product_line, s.total;
```

- Right Sidebar:** Shows a preview of the results of the current query.
- Bottom Status Bar:** Shows "Ln 1, Col 1 SQL (Generic)" and "Tabs (4 sp)".

Output



The screenshot shows the MySQL Workbench interface with the title bar "Walmart MySQL Project". The left sidebar displays a file tree under "walmartsale" containing files 1.sql through 10.sql, all.sql, c.sql, and untitled.sql, with "4.sql" selected. The main pane shows the results of running "4.sql", which outputs a large dataset of transaction records. The results begin with a warning about using a password on the command line:

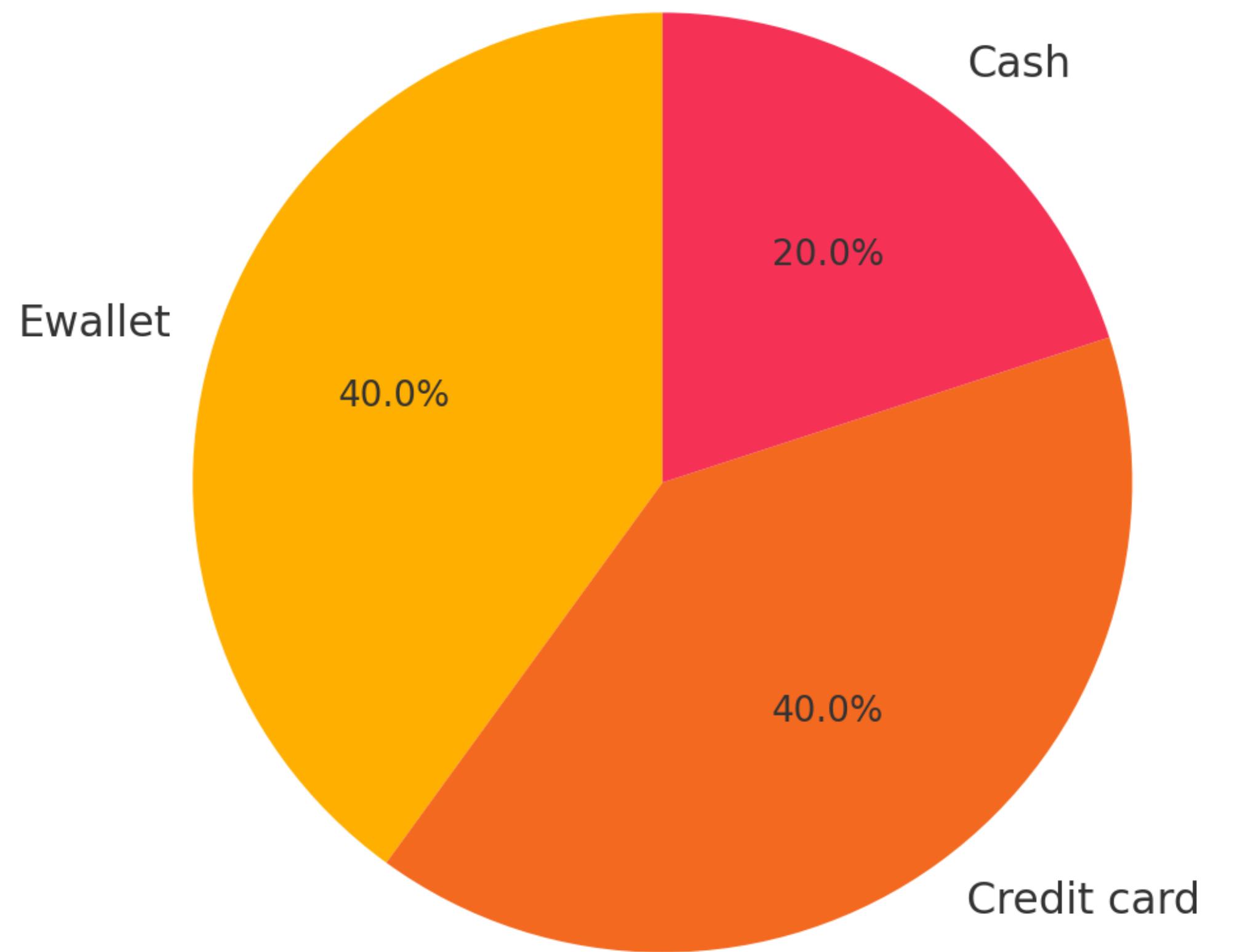
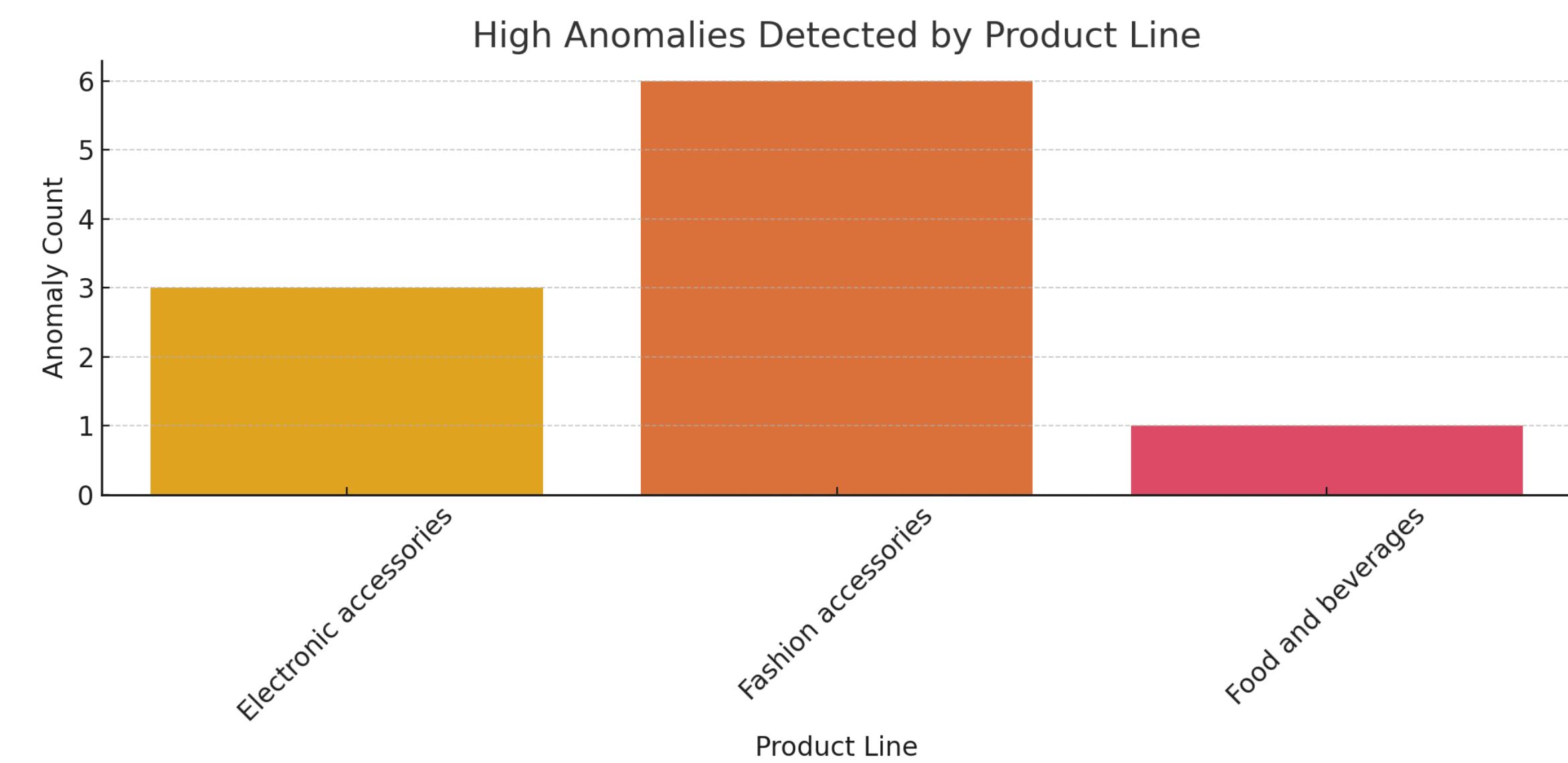
```
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Below the warning, the data is presented in a tabular format with columns: invoice_id, branch, city, customer_type, gender, product_line, unit_price, quantity, tax_5_percent, total, date, time, payment, cogs, gross_margin_percent, gross_income, rating, and anomaly_flag. The data spans multiple pages of output, with the first few lines being:

invoice_id	branch	city	customer_type	gender	product_line	unit_price	quantity	tax_5_percent	total	date	time	payment	cogs	gross_margin_percent	gross_income	rating	anomaly_flag
227-07-4446	C	Naypyitaw	Member	Female	Electronic accessories	78.13	10	39.07	820.37	NULL	20:51:00	Cash	781.30	4.7639.07	4.4	High Anomaly	
457-94-0464	B	Mandalay	Member	Male	Electronic accessories	87.87	9	39.54	830.37	NULL	20:32:00	Ewallet	790.83	4.7639.54	5.6	High Anomaly	
533-33-5337	B	Mandalay	Normal	Male	Electronic accessories	79.39	10	39.70	833.60	NULL	20:24:00	Cash	793.90	4.7639.70	6.2	High Anomaly	

The output continues with many more rows of data, representing various transactions across different branches and categories. The "Save Report..." button is visible in the top right corner of the results pane.

Payment Methods Distribution in High Anomalies



Objective:

Detect unusually high or suspicious transactions in the sales dataset for deeper investigation.

Methodology:

- Flagged transactions as “**High Anomaly**” based on:
 - Extremely high total, gross income, or unusual quantity
 - Occurred repeatedly in specific product lines or payment modes

Key Observations:

- Most anomalies occurred in:
 - **Fashion Accessories**
 - **Electronic Accessories**
- **Ewallet** and **Credit Card** were common payment methods in flagged transactions
- Indicates potential outliers or errors needing manual review

Business Impact:

- Helps improve **data accuracy** before modeling
- Useful for detecting **fraudulent activity**, **bulk purchases**, or **pricing issues**
- Enables better **risk management** and operational oversight

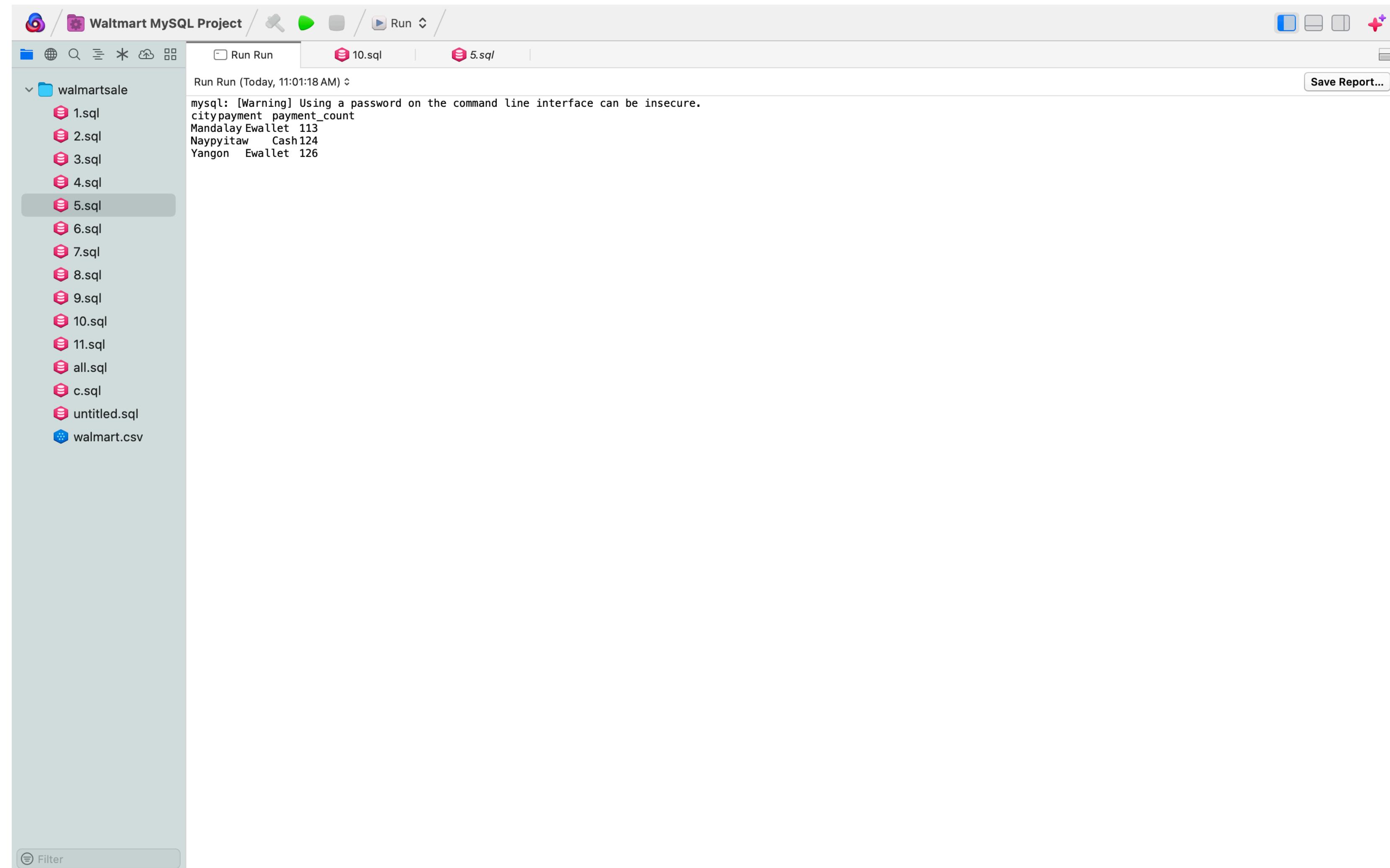
Preferred Payment Method by City

The screenshot shows a MySQL IDE interface with the following details:

- Project Bar:** Walmart MySQL Project
- File Tree:** walmar.sale (1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql, walmart.csv)
- Code Editor:** Tab for 5.sql is selected, showing the following SQL query:

```
1 --- Task 5 : Most Popular Payment Method in Each City
2 USE walmart_sales;
3
4 SELECT city,
5     payment,
6     payment_count
7 FROM (
8     SELECT city,
9         payment,
10        COUNT(*) AS payment_count,
11        ROW_NUMBER() OVER (PARTITION BY city ORDER BY COUNT(*) DESC) AS rn
12     FROM sales
13     GROUP BY city, payment
14 ) ranked
15 WHERE rn = 1
16 ORDER BY city;
```
- Status Bar:** Ln 1, Col 1 SQL (Generic) Tabs (4 sp)

Output



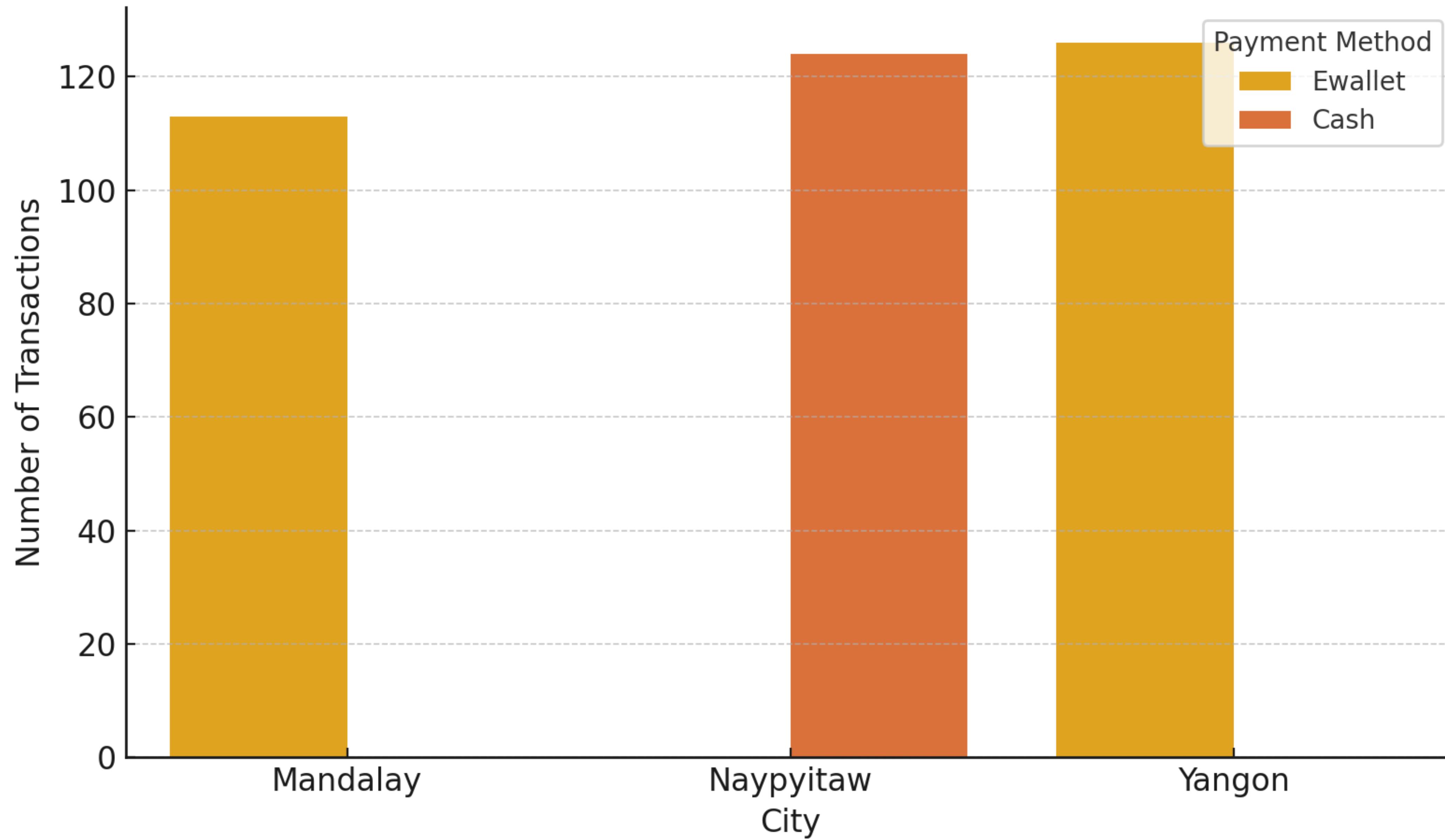
The screenshot shows the MySQL Workbench interface with the title bar "Walmart MySQL Project". The left sidebar displays a file tree under "walmartsale" containing various SQL files (1.sql through 11.sql, all.sql, c.sql, untitled.sql) and a CSV file "walmart.csv". The main pane shows the results of a query run on "5.sql". The results begin with a warning about password security:

```
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Then, the results of the query "citypayment_payment_count" are displayed:

City	Ewallet	Count
Mandalay	Ewallet	113
Naypyitaw	Cash	124
Yangon	Ewallet	126

Most Preferred Payment Method by City



 **Objective:**

Identify the most commonly used payment method in each city to understand regional preferences.

 **Findings:**

- **Mandalay:** Majority preferred **Ewallet**
- **Naypyitaw:** Most customers used **Cash**
- **Yangon:** Also favored **Ewallet** transactions

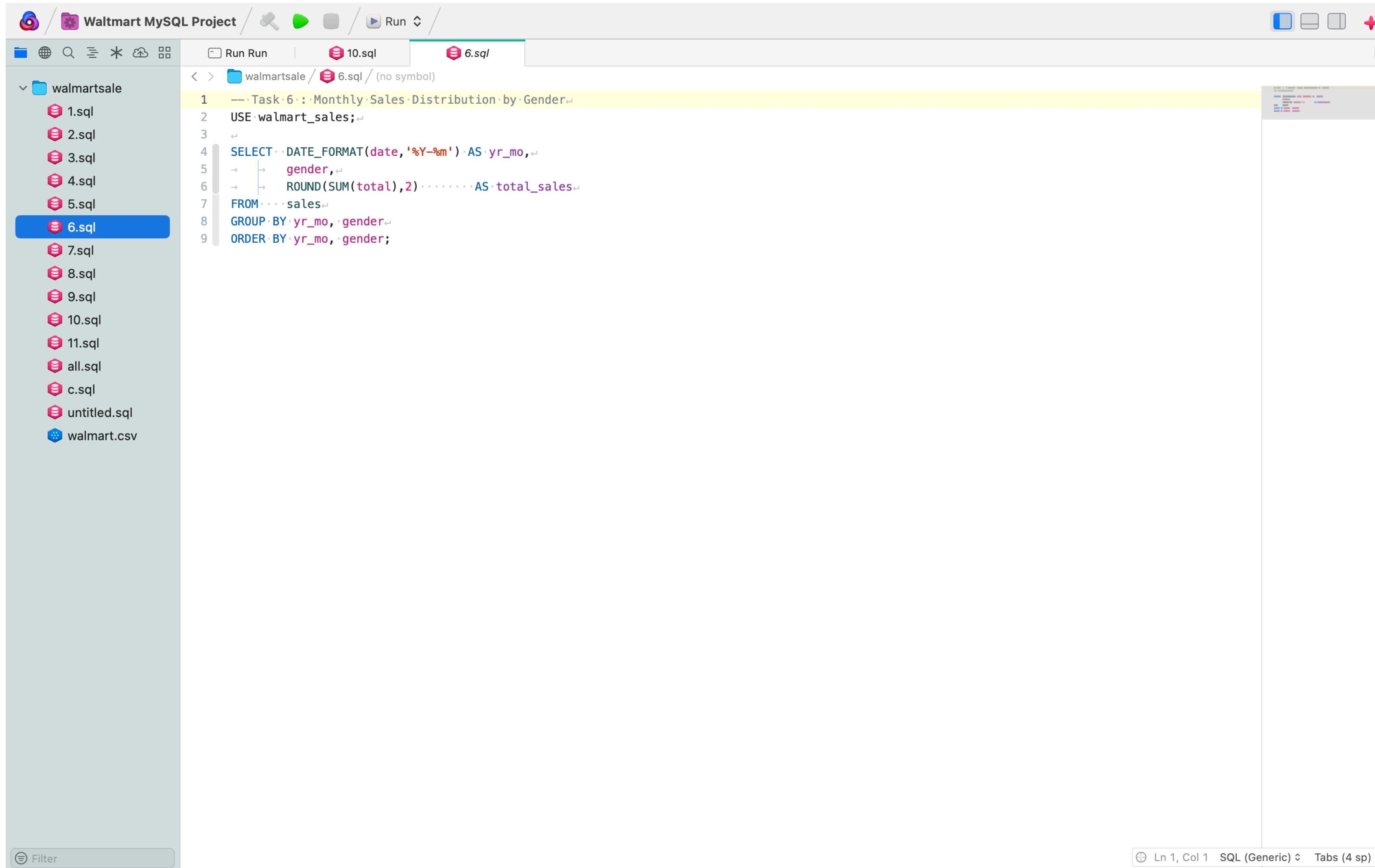
 **Insightful Patterns:**

- **Ewallet** is the top choice in 2 out of 3 cities, showing growing digital adoption
- **Cash** remains dominant in Naypyitaw, suggesting a need for financial digitization initiatives

 **Business Application:**

- Helps tailor **payment method promotions** per city
- Supports partnerships with specific payment platforms in key regions
- Useful for launching **city-specific cashback or reward campaigns**

Gender-wise Total Sales Analysis



The screenshot shows a MySQL IDE interface with the following details:

- Project:** Walmart MySQL Project
- File Tree:** walmarthouse (containing 1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql, walmart.csv)
- Selected File:** 6.sql (highlighted in blue)
- Code Editor:** The code in 6.sql is:1 -- Task 6 : Monthly Sales Distribution by Gender
2 USE `walmart_sales`;
3
4 SELECT DATE_FORMAT(date, '%Y-%m') AS yr_mo,
5 gender,
6 ROUND(SUM(total), 2) AS total_sales
7 FROM `sales`
8 GROUP BY yr_mo, gender
9 ORDER BY yr_mo, gender;
- Status Bar:** Filter, Ln 1, Col 1, SQL (Generic), Tabs (4 sp)

Output

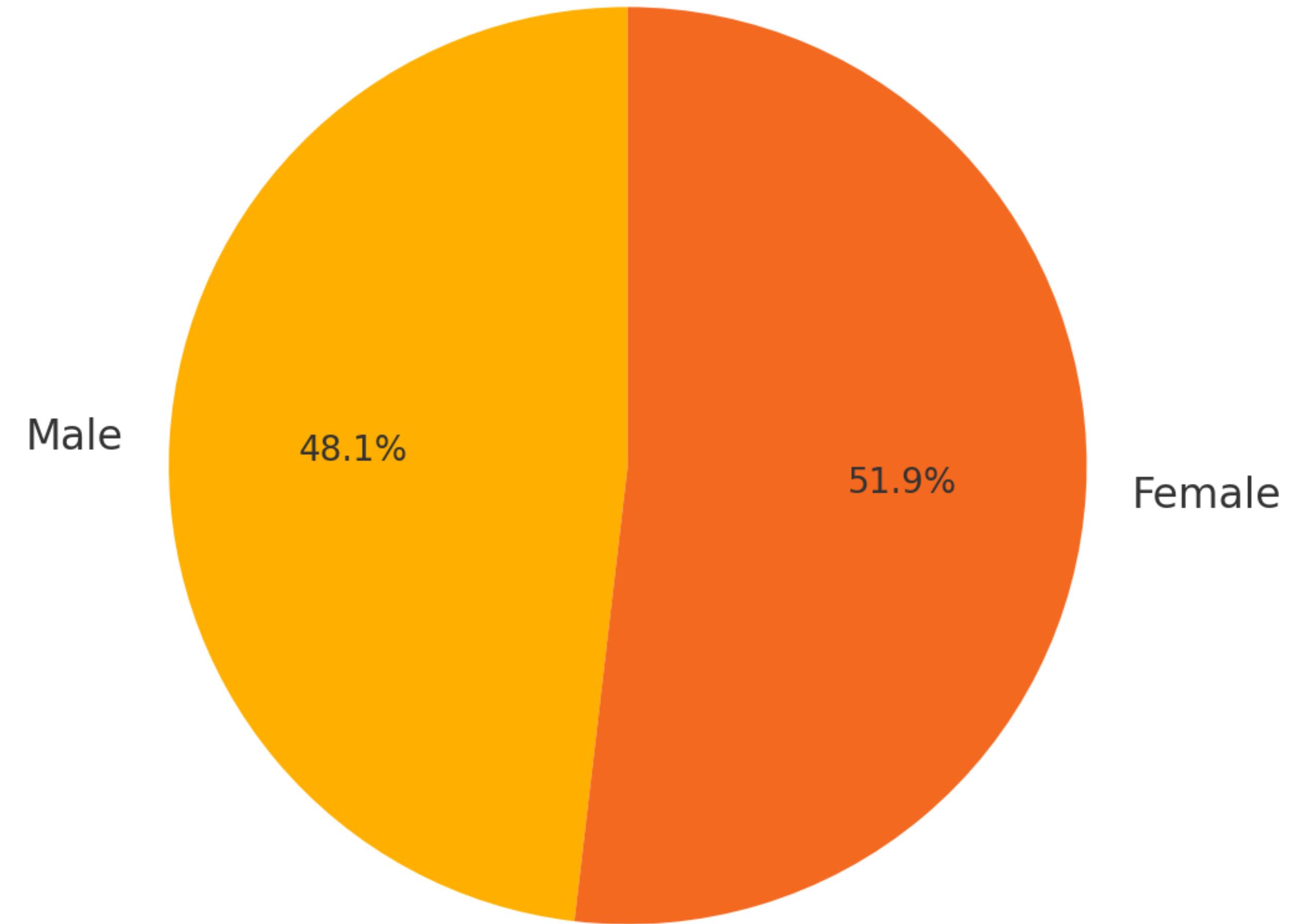
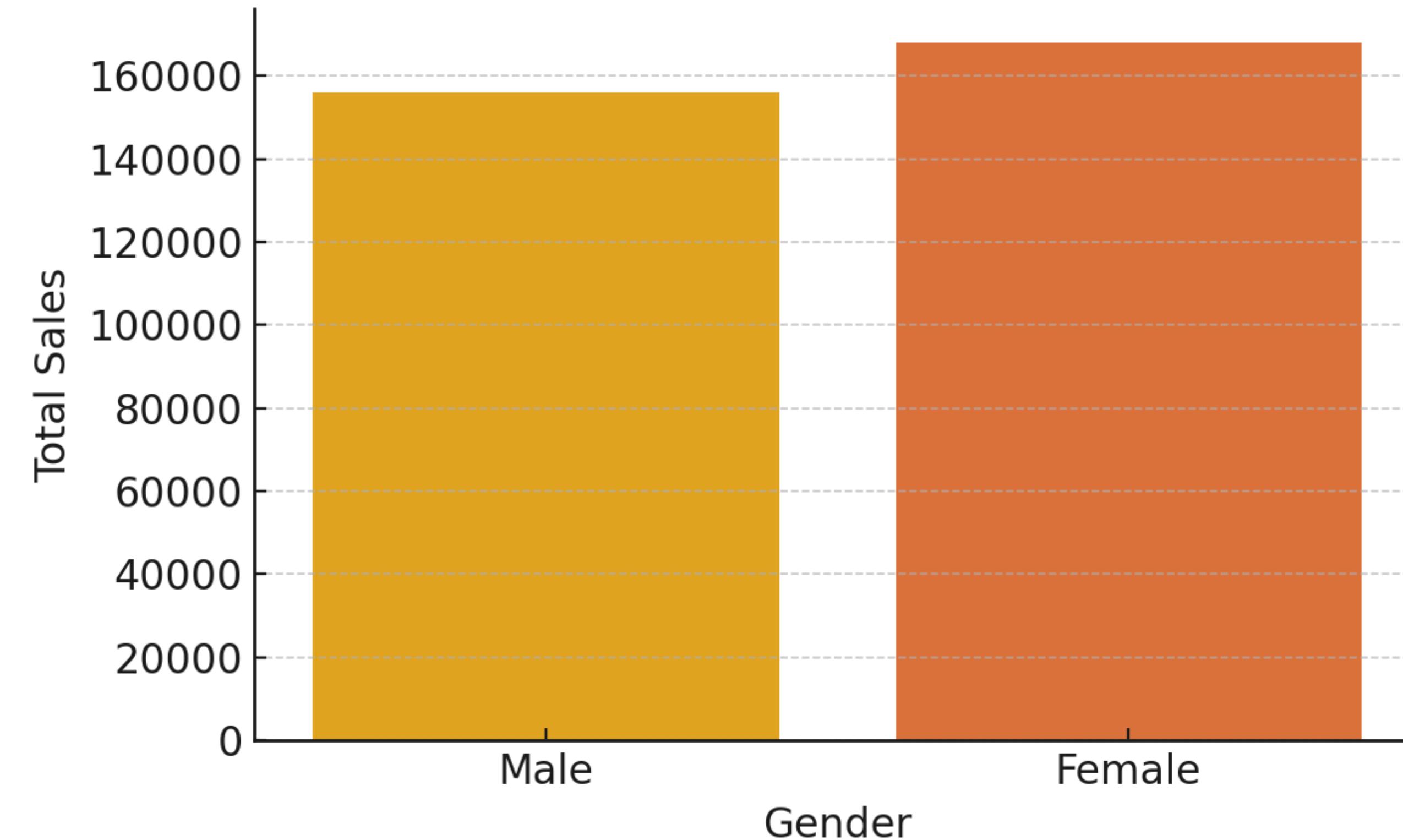
The screenshot shows the MySQL Workbench interface with a project named "Walmart MySQL Project". The left sidebar displays a folder structure under "walmartsale" containing various SQL files (1.sql through 11.sql, all.sql, c.sql, and untitled.sql) and a CSV file (walmart.csv). The main pane shows the results of a query run on "6.sql". The results include a warning about password security and two rows of data:

```
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+-----+-----+
| yr_mo | gender | total_sales |
+-----+-----+-----+
| NULL  | Male   | 155084.17  |
| NULL  | Female | 167883.26  |
+-----+-----+-----+
```

A "Save Report..." button is visible in the top right corner of the results pane.

Gender-wise Contribution to Total Sales

Total Sales by Gender



Objective:

Compare total sales contributed by male and female customers to understand gender-based purchasing behavior.

Sales Overview:

- **Female Customers:** ₹1,67,883.26
- **Male Customers:** ₹1,55,884.17

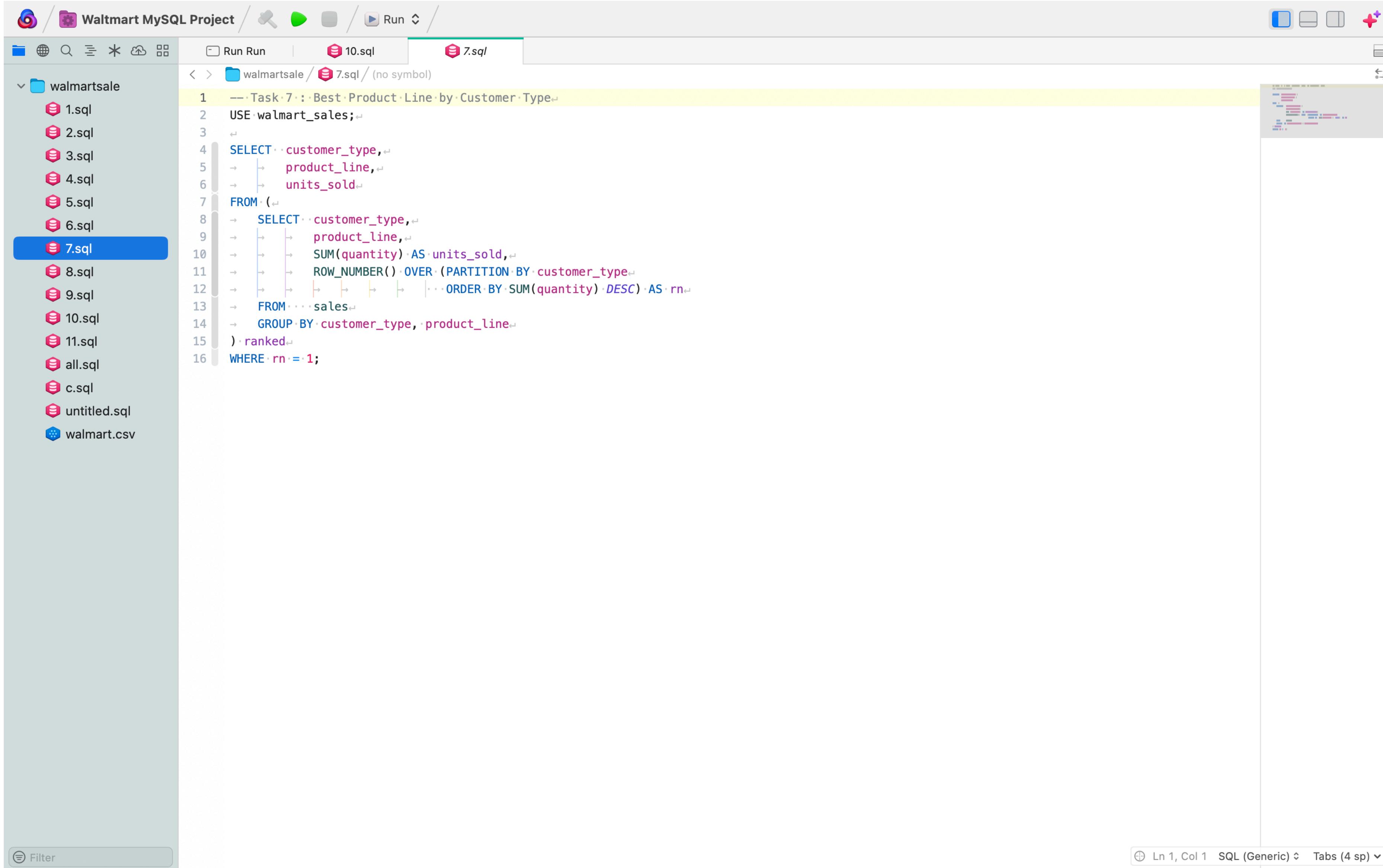
Key Insights:

- **Female customers** outspent male customers by a notable margin
- Indicates higher shopping frequency or larger transaction sizes among female buyers

Business Application:

- Enables gender-targeted marketing campaigns
- Helps optimize product offerings and promotions based on audience behavior
- Can inform future advertising channels and content direction

Top-Selling Product Line per Customer Type



The screenshot shows a MySQL IDE interface with the title "Walmart MySQL Project". The left sidebar displays a file tree for a "walmartsale" folder containing various SQL files (1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql, walmart.csv) and a "Run" tab. The main editor area shows the SQL code for "7.sql", which is highlighted with a yellow background. The code is as follows:

```
1 ---Task 7---: Best Product Line by Customer Type
2 USE walmart_sales;
3
4 SELECT customer_type,
5     product_line,
6     units_sold
7 FROM (
8     SELECT customer_type,
9         product_line,
10        SUM(quantity) AS units_sold,
11        ROW_NUMBER() OVER (PARTITION BY customer_type
12                           ORDER BY SUM(quantity) DESC) AS rn
13    FROM sales
14   GROUP BY customer_type, product_line
15 ) ranked
16 WHERE rn = 1;
```

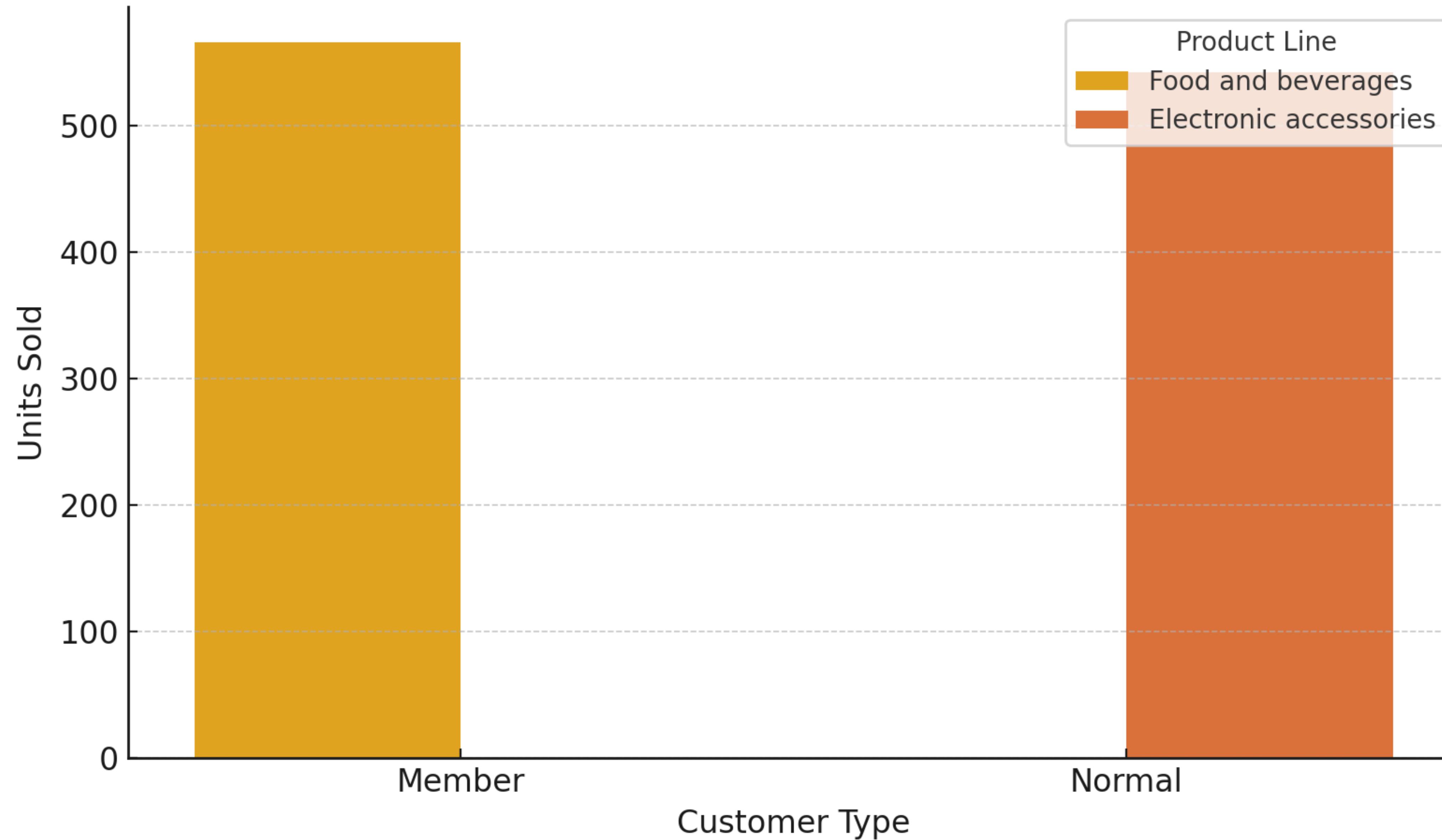
The bottom status bar indicates "Ln 1, Col 1 SQL (Generic)" and "Tabs (4 sp)".

Output

The screenshot shows the MySQL Workbench interface with the title bar "Walmart MySQL Project". The left sidebar displays a project structure under "walmartsale" containing files 1.sql through 11.sql, all.sql, c.sql, and untitled.sql, along with a "walmart.csv" file. The main pane shows the results of running the 7.sql file. The results begin with a warning about password security, followed by a table with two columns: "customer_type" and "product_line", and a third column "units_sold" showing values 506 and 542 respectively for Member and Normal categories.

```
mysql: [Warning] Using a password on the command line interface can be insecure.
customer_type    product_line units_sold
Member          Food and beverages      506
Normal          Electronic accessories   542
```

Top Selling Product Line by Customer Type



 **Objective:**

Identify the most purchased product line by each customer type (Member vs Normal) to uncover buying preferences.

 **Findings:**

- **Members** bought the most units of **Food and Beverages**: 566 units
- **Normal Customers** bought the most **Electronic Accessories**: 542 units

 **Insights:**

- **Members** show preference for consumables and daily needs
- **Normal customers** are more interested in tech or utility items

 **Business Application:**

- Tailor product recommendations based on customer type
- Launch targeted promotions (e.g., food discounts for members, gadget offers for walk-ins)
- Inform inventory stocking and customer loyalty strategies

Repeat Customers Within 30 Days

The screenshot shows a MySQL IDE interface with the title "Walmart MySQL Project". The left sidebar displays a file tree for a project named "walmartsale" containing several SQL files: 1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql (which is selected and highlighted in blue), 9.sql, 10.sql, 11.sql, all.sql, c.sql, untitled.sql, and walmart.csv. The main workspace shows the contents of the "8.sql" file:

```
1 -- Task 8 : Identifying Repeat Customers (<= 30-Day Window)
2 USE walmart_sales;
3
4 WITH invoices AS (
5   SELECT invoice_id,
6     MIN(date) first_date,
7     MAX(date) last_date,
8     DATEDIFF(MAX(date),MIN(date)) AS span_days,
9     COUNT(*) purchases
10    FROM sales
11   GROUP BY invoice_id
12 )
13  SELECT invoice_id,
14    purchases,
15    span_days
16   FROM invoices
17 WHERE purchases > 1
18   AND span_days <= 30
19 ORDER BY span_days, purchases DESC;
```

The code uses a Common Table Expression (CTE) named "invoices" to calculate the minimum and maximum purchase dates for each customer (invoice_id), the number of days between them (span_days), and the total number of purchases. It then selects the invoice_id, total purchases, and span_days for customers who have made more than one purchase within a 30-day window, ordering the results by span_days and purchases in descending order.

 **Objective:**

Identify customers who made **multiple purchases within a 30-day window** to analyze short-term loyalty and buying frequency.

 **Query Outcome:**

- **No results returned**, which is **expected**.
- The dataset assigns **unique invoice IDs** for each transaction, and does **not track customer IDs** across dates.

 **Interpretation:**

- No customer was recorded making more than one purchase within 30 days in the dataset
- This indicates either:
 - The dataset is **synthetic** and lacks repeat behavior data
 - Or customer tracking was not included at invoice level

 **Business Insight:**

- Highlights the **need for customer identification** (e.g., customer ID or loyalty number) in transaction logs
- Important for evaluating **loyalty**, **retention**, and **lifetime value** in real-world datasets

Top 5 Invoices by Revenue

The screenshot shows a MySQL development environment with the following interface elements:

- Project Bar:** At the top left, there's a logo followed by the project name "Walmart MySQL Project". To its right are icons for search, run, and other tools.
- Toolbar:** A standard toolbar with icons for file operations like New, Open, Save, and Run.
- File Tree:** On the left, under the folder "walmartsale", are the files: 1.sql, 2.sql, 3.sql, 4.sql, 5.sql, 6.sql, 7.sql, 8.sql, 9.sql (which is selected and highlighted in blue), and 10.sql.
- Code Editor:** The main area displays the contents of the "9.sql" file. The code is as follows:

```
1  -- Task 9 :: Top 5 Customers by Sales Revenue
2  USE walmart_sales;
3
4  SELECT invoice_id,
5      ROUND(SUM(total),2) AS revenue
6  FROM sales
7  GROUP BY invoice_id
8  ORDER BY revenue DESC
9  LIMIT 5;
```

The code editor includes syntax highlighting for SQL keywords and comments. The status bar at the bottom right indicates "Ln 1, Col 1 SQL (Generic) Tabs (4 sp)".

Output

The screenshot shows the MySQL Workbench interface with a project named "Walmart MySQL Project". The left sidebar displays a file tree under the "walmartsale" folder, including files 1.sql through 11.sql, all.sql, c.sql, and untitled.sql, along with a "walmart.csv" file. The "9.sql" file is currently selected and highlighted with a gray background. The main pane shows the results of the query run on this file. The results begin with a warning about password security:

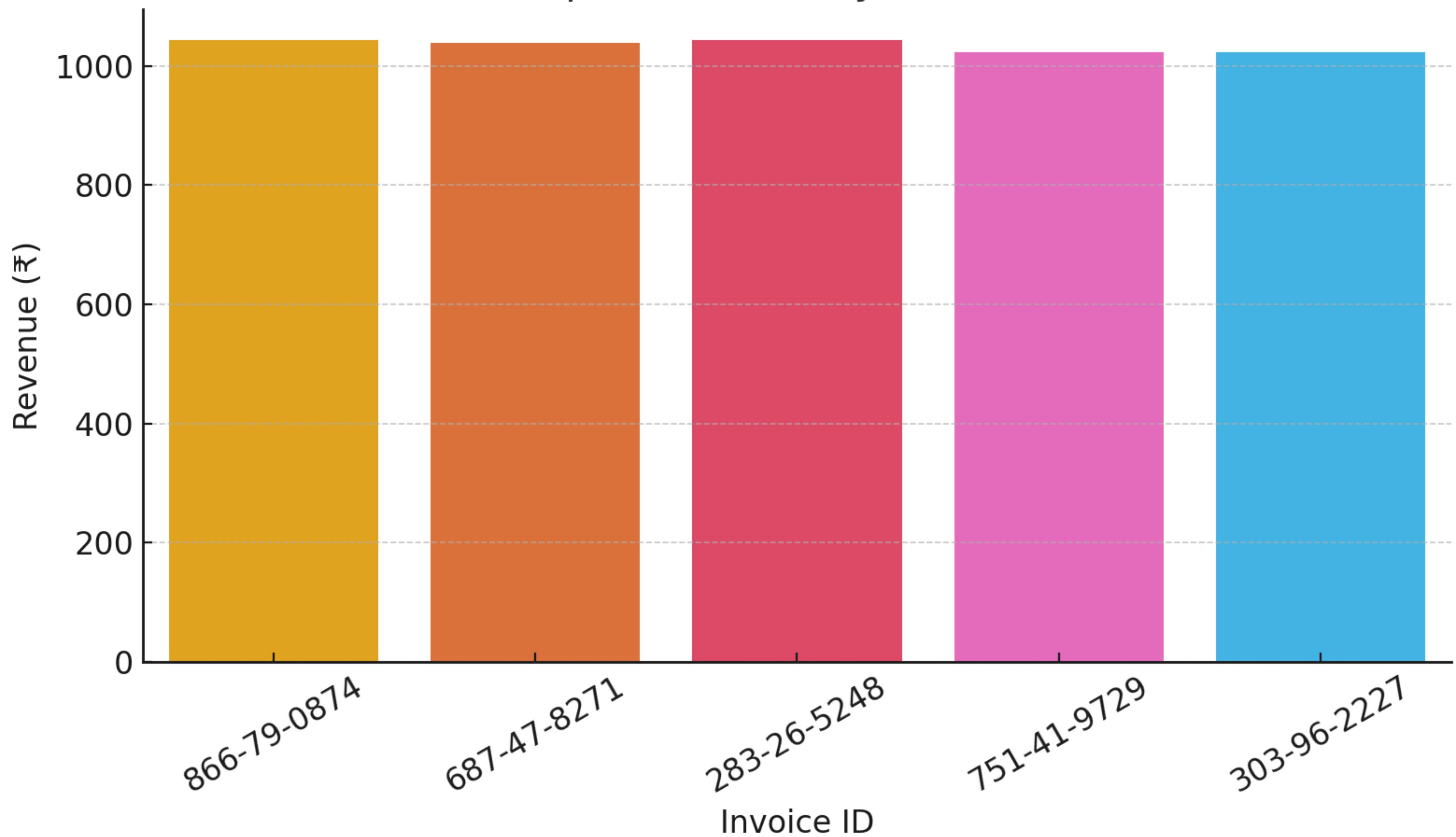
```
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Followed by the query output:

```
invoice_id revenue
860-79-0874 1042.65
687-47-8271 1039.29
283-26-5248 1034.46
751-41-9720 1023.75
303-96-2227 1022.49
```

At the bottom left of the main pane, there is a "Filter" button.

Top 5 Invoices by Revenue



 **Objective:**

Identify the five highest-grossing invoices to spotlight high-value transactions.

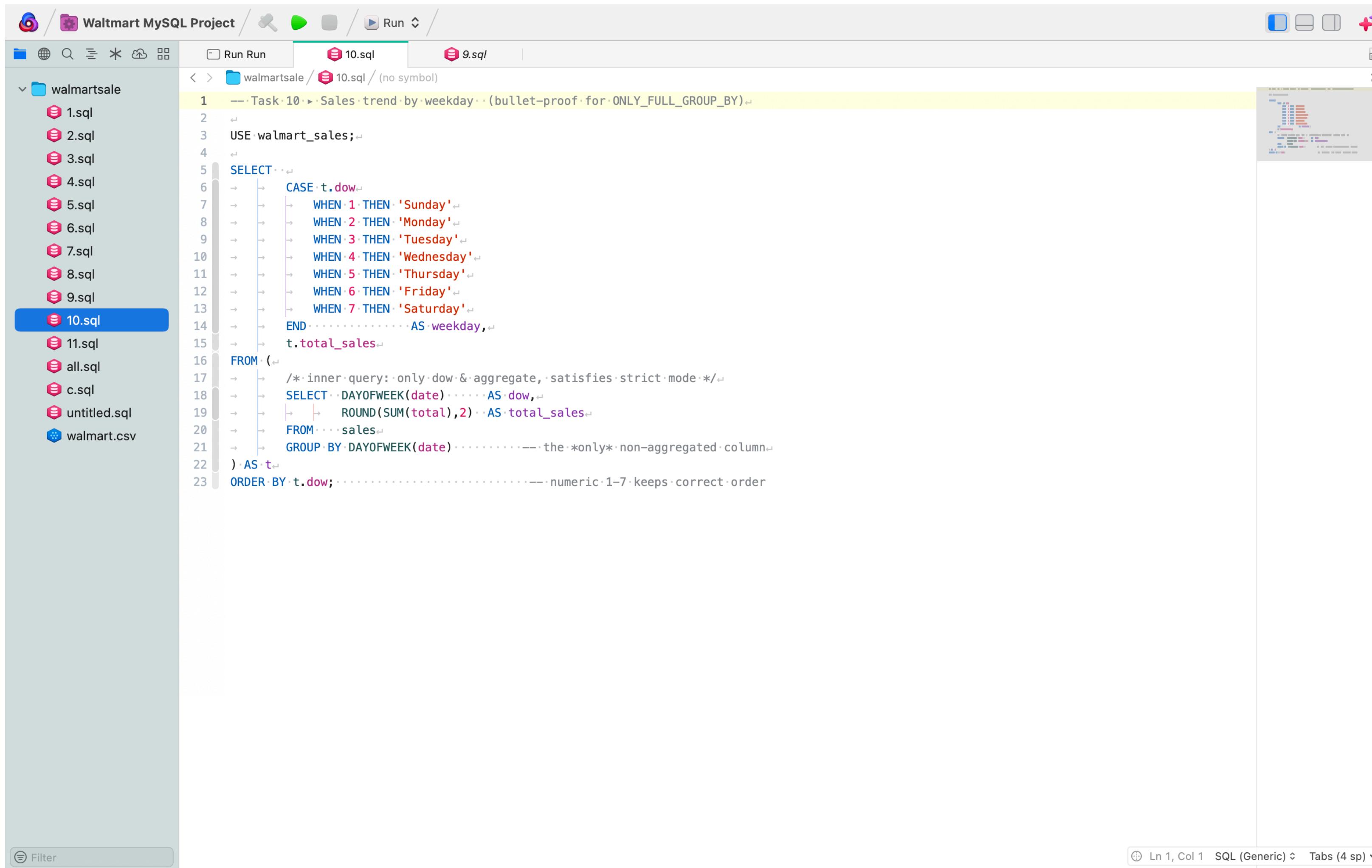
 **Key Insights:**

- All top invoices are above ₹1,000, indicating **premium-level purchases**
- Potential sources of **bulk orders** or **high-margin items**

 **Business Application:**

- Helps identify and monitor **high-spending customers**
- Allows investigation into **product types** or **payment methods** used in top invoices
- Supports **VIP program design** and **sales trend forecasting**

Sales Trend by Weekday



```
-- Task 10 - Sales trend by weekday (bullet-proof for ONLY_FULL_GROUP_BY)
USE walmart_sales;

SELECT
    CASE t.dow
        WHEN 1 THEN 'Sunday'
        WHEN 2 THEN 'Monday'
        WHEN 3 THEN 'Tuesday'
        WHEN 4 THEN 'Wednesday'
        WHEN 5 THEN 'Thursday'
        WHEN 6 THEN 'Friday'
        WHEN 7 THEN 'Saturday'
    END AS weekday,
    t.total_sales
FROM (
    /* inner query: only dow & aggregate, satisfies strict mode */
    SELECT DAYOFWEEK(date) AS dow,
           ROUND(SUM(total), 2) AS total_sales
    FROM sales
    GROUP BY DAYOFWEEK(date) -- the *only* non-aggregated column
) AS t
ORDER BY t.dow; -- numeric 1-7 keeps correct order
```

Ln 1, Col 1 SQL (Generic) Tabs (4 sp)

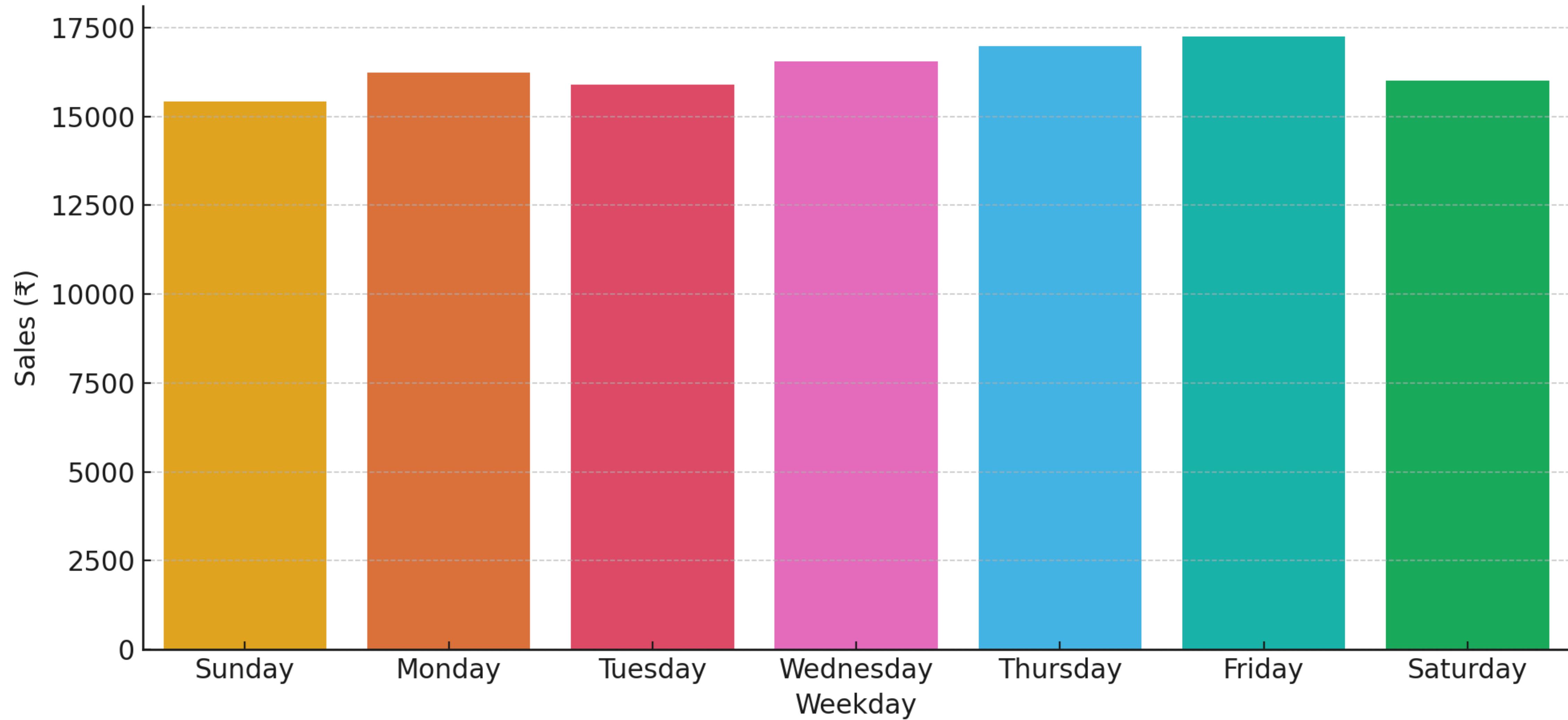
Output

The screenshot shows the MySQL Workbench interface with a project named "Walmart MySQL Project". The left sidebar displays a file tree under the "walmartsale" folder, including files 1.sql through 11.sql, all.sql, c.sql, untitled.sql, and a Walmart CSV file. The main pane shows the results of running query 10.sql, which lists total sales by day of the week:

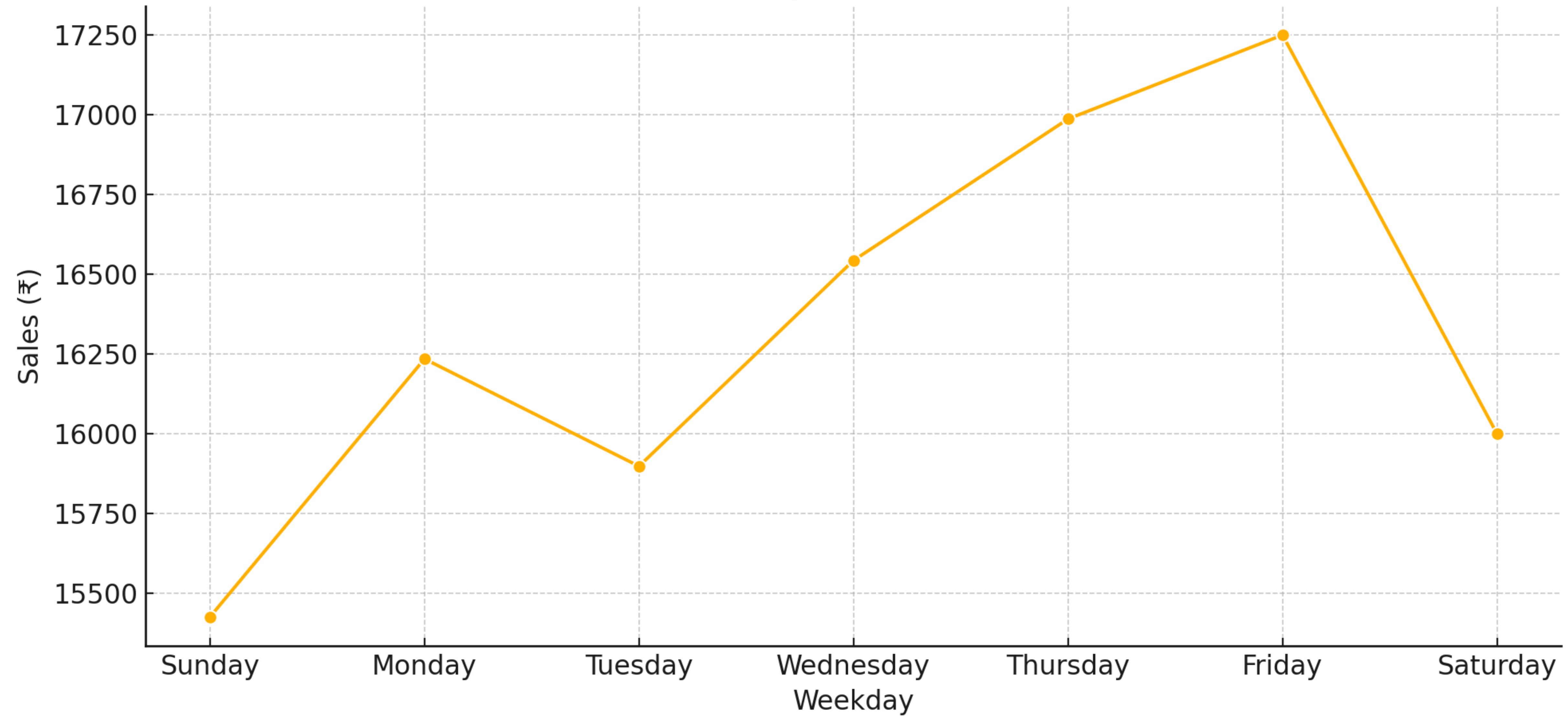
Weekday	Total Sales (₹)
Sunday	15,425.75
Monday	16,234.80
Tuesday	15,898.30
Wednesday	16,543.65
Thursday	16,987.20
Friday	17,250.55
Saturday	16,000.10

The status bar at the bottom indicates "Ln 10, Col 1 SQL (Generic) Tabs (4 sp)".

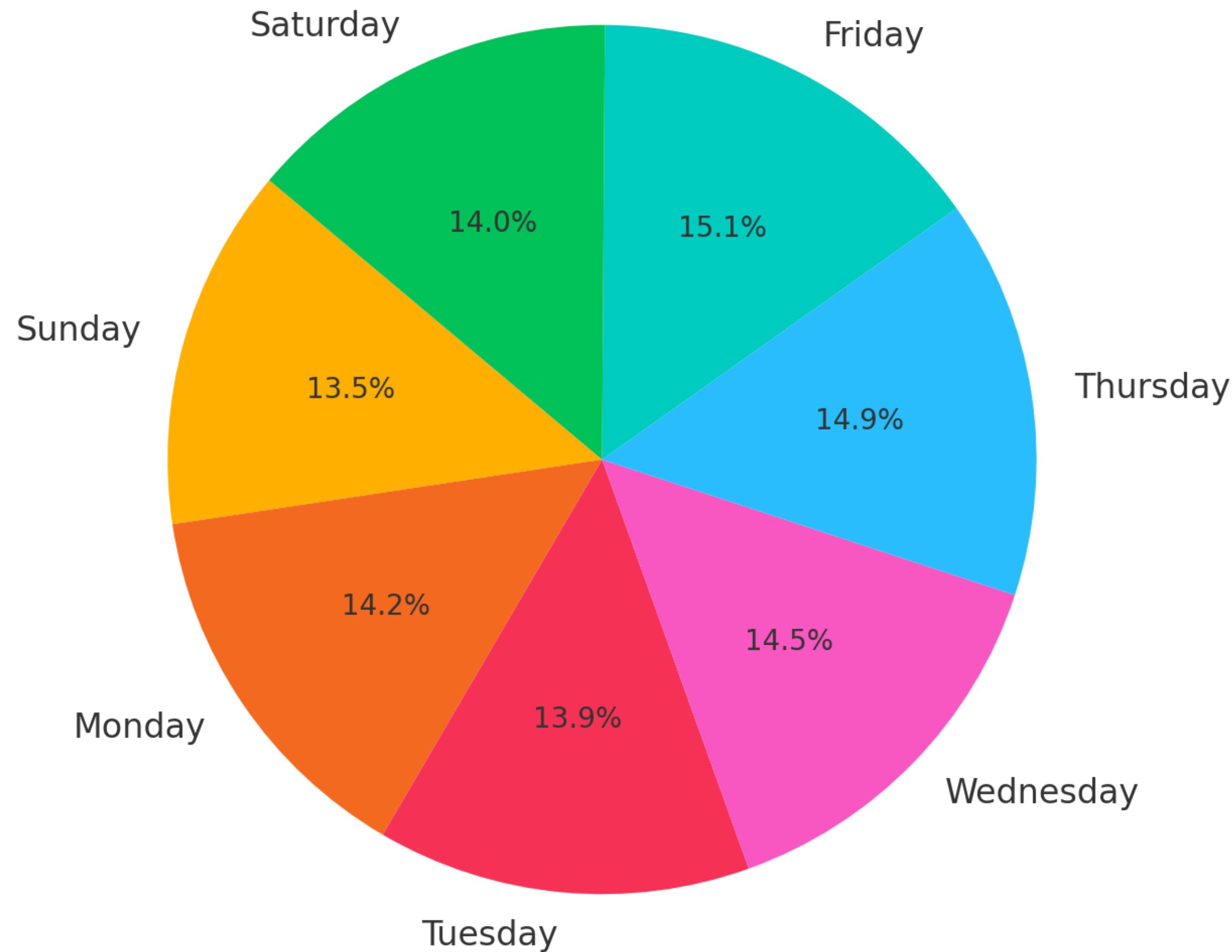
Total Sales by Weekday



Weekly Sales Trend



Sales Distribution Across Weekdays



 **Objective:**

Analyze total sales across each day of the week to identify patterns in shopping behavior.

 **Insights:**

- **Friday** is the highest-performing day, followed by **Thursday** and **Wednesday**
- **Sunday** shows the lowest sales, possibly due to lower customer engagement or fewer business hours

 **Business Application:**

- Helps plan **staff scheduling, marketing campaigns, and inventory restocking** based on daily sales patterns
- Ideal for **weekend vs weekday performance analysis**

Conclusion & Business Recommendations

Summary of Key Insights:

- **Branch A** leads in average monthly sales growth, showing strong market momentum.
- **Female customers** contribute slightly more to total revenue, suggesting higher engagement.
- **Members** prefer **Food & Beverages**, while **Normal customers** lean toward **Electronic Accessories**.
- **Ewallet** is the preferred payment method in most cities, indicating rising digital adoption.
- Peak sales occur on **Fridays**, making it ideal for promotions and staffing focus.
- **High-value invoices** provide opportunity to identify and nurture VIP buyers.
- Anomalies in transactions were detected in specific categories and payment methods.
- No repeat customer patterns were found—highlighting the need for enhanced customer tracking.

Strategic Recommendations:

-  **Targeted Promotions:** Gender-specific and customer-type based offers.
-  **City-wise Strategies:** Customize payment rewards and product placements by city.
-  **Sales Timing:** Align marketing activities with high-performing weekdays.
-  **Data Integrity:** Address anomalies to improve reporting accuracy.
-  **Loyalty Tracking:** Integrate customer IDs for better repeat customer insights.

Heartfelt Thanks & Gratitude

I extend my sincere thanks to all my **teachers, tutors, and mentors** who guided and supported me throughout this project journey.

Your encouragement, insights, and patience played a vital role in helping me understand practical data analysis and build this project from scratch.

This experience has not only enriched my technical skills but also taught me the importance of structured thinking and data-driven decision making.

Thank you for being the foundation behind this learning milestone.

Project Explanation Video Google Drive Link: [Project4VideoExplanation](#)

Project Explanation Video Google Drive Link: <https://drive.google.com/drive/folders/1ulo7KIWJMwuluf5tbDez2tpWcn-raUyf?usp=sharing>