SVKM's NMIMS
School of Technology Management & Engineering, Chandigarh
A.Y. 2023 - 24
Course: Database Management Systems

Project Report

Program	Btech CE (B)	
Semester	4th	
Name of the Project:	Walmart-Sales-Data-Analysis	
Details of Project Members		
Batch	Roll No.	Name
B1	B033	Saakshi Jain
Date of Submission:		

Contribution of each project Members:

Roll No.	Name:	Contribution
B033	Saakshi Jain	she helped in Storyline Components of Database Design Entity Relationship Diagram Relational Model Normalization SQL Queries Learning from the Project Project Demonstration Self-learning beyond classroom Learning from the project Challenges faced Conclusion

Github link of your project:

https://github.com/saakshijain2022/Walmart-Sales-Insights-SQL-Analysis

Note:

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004_DBMSProject)
3. Submit all relevant files of your work (Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

Rubrics for the Project evaluation:

First phase of evaluation: Innovative Ideas (5 Marks) Design and Partial implementation (5 Marks)	10 marks
Final phase of evaluation Implementation, presentation and viva, Self-Learning and Learning Beyond classroom	10 marks

Project Report

Selected Topic

by

Student 1, Roll number: B033

Course: DBMS

AY: 2023-24

Table of Contents

Sr no.	Topic	Page no.
1	Storyline	
2	Components of Database Design	
3	Entity Relationship Diagram	
4	Relational Model	
5	Normalization	
6	SQL Queries	
7	Learning from the Project	
8	Project Demonstration	
9	Self-learning beyond classroom	
10	Learning from the project	

8	Challenges faced	
9	Conclusion	

I. Storyline

The Walmart Sales Data Analysis project aims to delve into the sales data of Walmart's branches located in Mandalay, Yangon, and Naypyitaw. The dataset, sourced from the Kaggle Walmart Sales Forecasting Competition, contains detailed information about sales transactions, including invoice ID, branch, city, customer type, product line, unit price, quantity, VAT, total amount, date, time, payment method, cost of goods sold, gross margin percentage, gross income, and rating.

The project revolves around understanding Walmart's sales patterns, identifying high-performing branches, analyzing product line performance, and evaluating customer behavior. By exploring various factors influencing sales across different branches, the project aims to optimize sales strategies and enhance overall performance.

II. Components of Database Design

Entities and Attributes:

Branch:

Attributes: Branch (Primary Key), City

Product:

Attributes: Product Line (Primary Key), Unit Price, VAT, Product Category

Sales Transaction:

Attributes: Invoice ID (Primary Key), Branch (Foreign Key), City, Customer Type, Gender, Product Line (Foreign Key), Unit Price, Quantity, Total Amount, Date, Time, Payment Method, Cost of Goods Sold, Gross Margin Percentage, Gross Income, Rating

Relationships:

Branch - Sales Transaction:

Cardinality: One branch can have many sales transactions.

Participation: Mandatory on the branch side (each sale must be associated with a branch), optional on the sales transaction side (not all branches may have sales transactions).

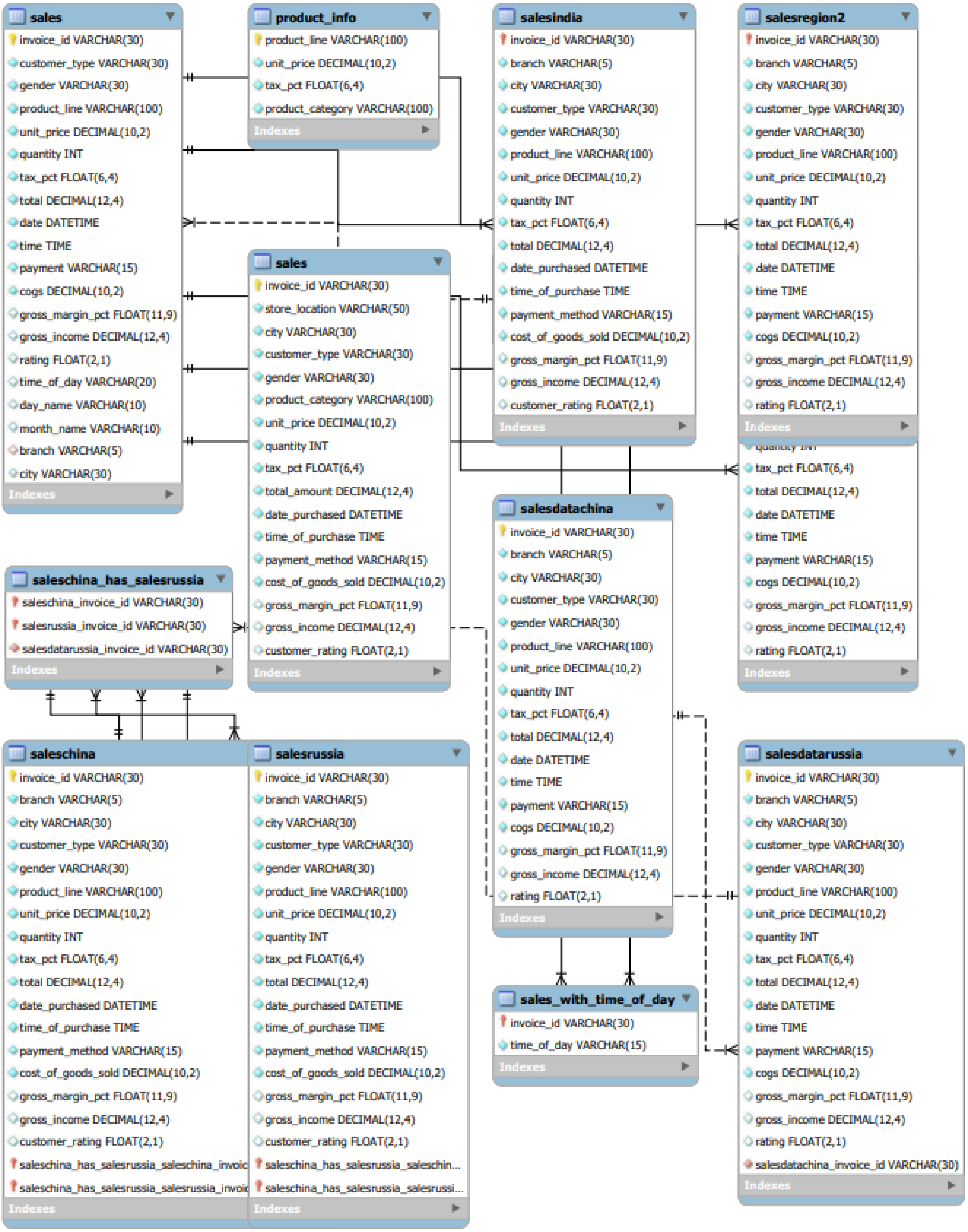
Product - Sales Transaction:

Cardinality: One product can be sold in many transactions.

Participation: Mandatory on the product side (each sale must be associated with a product), optional on the sales transaction side (not all products may be sold in transactions).

The database design ensures proper normalization by separating entities into distinct tables and establishing appropriate relationships between them. This structure facilitates efficient data management and analysis, enabling comprehensive exploration of Walmart's sales data.

III. Entity Relationship Diagram & Relational Model



```
-- Create table
CREATE TABLE IF NOT EXISTS sales(
         invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
   branch VARCHAR(5) NOT NULL,
   city VARCHAR(30) NOT NULL,
   customer_type VARCHAR(30) NOT NULL,
   gender VARCHAR(30) NOT NULL,
   product_line VARCHAR(100) NOT NULL,
   unit_price DECIMAL(10,2) NOT NULL,
   quantity INT NOT NULL,
   tax_pct FLOAT(6,4) NOT NULL,
   total DECIMAL(12, 4) NOT NULL,
   date DATETIME NOT NULL,
   time TIME NOT NULL,
   payment VARCHAR(15) NOT NULL,
   cogs DECIMAL(10,2) NOT NULL,
   gross_margin_pct FLOAT(11,9),
   gross_income DECIMAL(12, 4),
   rating FLOAT(2, 1)
);

-- Data cleaning
SELECT * FROM salesDataWalmart.sales;




-- Create table for Indian region sales
CREATE TABLE IF NOT EXISTS salesIndia (
   invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
   branch VARCHAR(5) NOT NULL,
   city VARCHAR(30) NOT NULL,
   customer_type VARCHAR(30) NOT NULL,
   gender VARCHAR(30) NOT NULL,
   product_line VARCHAR(100) NOT NULL,
   unit_price DECIMAL(10,2) NOT NULL,
   quantity INT NOT NULL,
   tax_pct FLOAT(6,4) NOT NULL,
   total DECIMAL(12, 4) NOT NULL,
   date_purchased DATETIME NOT NULL,
   time_of_purchase TIME NOT NULL,
   payment_method VARCHAR(15) NOT NULL,
   cost_of_goods_sold DECIMAL(10,2) NOT NULL,
   gross_margin_pct FLOAT(11,9),
   gross_income DECIMAL(12, 4),
   customer_rating FLOAT(2, 1)
);
```

```
-- Create a relationship between the salesIndia table and the sales table
ALTER TABLE salesIndia
ADD CONSTRAINT fk_invoice_id_salesIndia
FOREIGN KEY (invoice_id) REFERENCES sales(invoice_id);

-- Create databases
CREATE DATABASE IF NOT EXISTS salesDataAfrica;
CREATE DATABASE IF NOT EXISTS salesDataChina;
CREATE DATABASE IF NOT EXISTS salesDataRussia;
CREATE DATABASE IF NOT EXISTS salesDataCanada;
CREATE DATABASE IF NOT EXISTS salesDataAustralia;

-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesAfrica(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating FLOAT(2, 1)
);

-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesDataChina(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating DECIMAL(10, 2)
);

-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesDataRussia(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating INT
);

-- Create a relationship between the salesChina table and salesDataWalmart.sales table
ALTER TABLE salesChina
ADD CONSTRAINT fk_invoice_id_salesChina
FOREIGN KEY (invoice_id) REFERENCES salesDataWalmart.sales(invoice_id);

-- Create table for Chinese region sales
CREATE TABLE IF NOT EXISTS salesChina (
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date_purchased DATETIME NOT NULL,
    time_of_purchase TIME NOT NULL,
    payment_method VARCHAR(15) NOT NULL,
    cost_of_goods_sold DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    customer_rating FLOAT(2, 1)
);
```

V. Normalization

Perform normalization (1NF, 2NF, 3NF, BCNF) as applicable for the entire database.

Normalization Steps:
1. First Normal Form (1NF):
1NF requires that each column in a table should contain atomic (indivisible) values, and there should be no repeating groups or arrays.

To ensure 1NF:

Make sure each column contains atomic values.
Remove any repeating groups or arrays by splitting them into separate tables if needed.
2. Second Normal Form (2NF):
2NF requires that the table is in 1NF and all non-key attributes are fully functional dependent on the primary key.

To ensure 2NF:

If there are any partial dependencies (attributes depend on only part of the primary key), move them to separate tables.
3. Third Normal Form (3NF):
3NF requires that the table is in 2NF and there are no transitive dependencies.

To ensure 3NF:

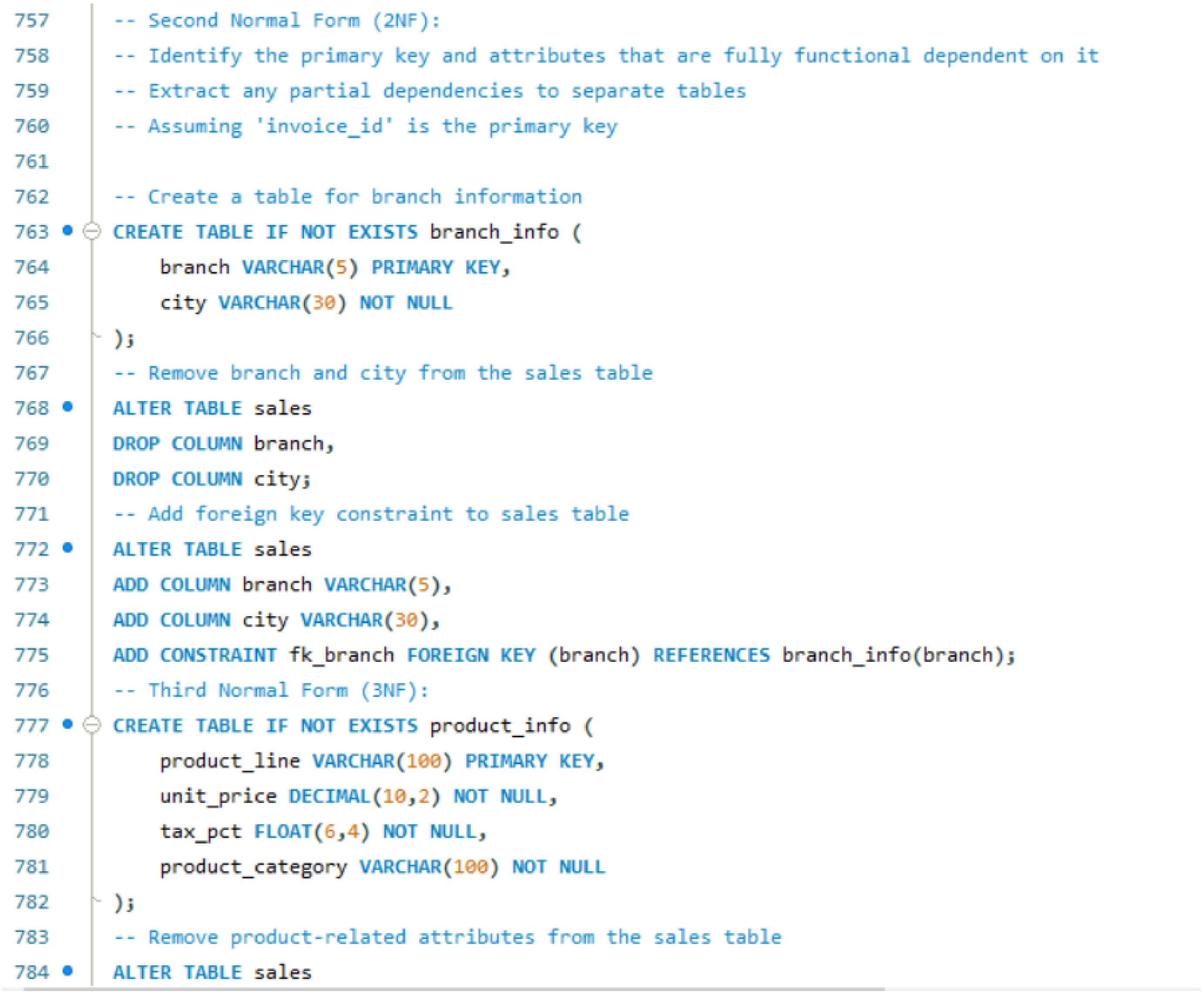
Remove any transitive dependencies by moving attributes to separate tables.
Boyce-Codd Normal Form (BCNF):
BCNF is a stricter form of 3NF, which requires that every determinant be a candidate key.

To ensure BCNF:

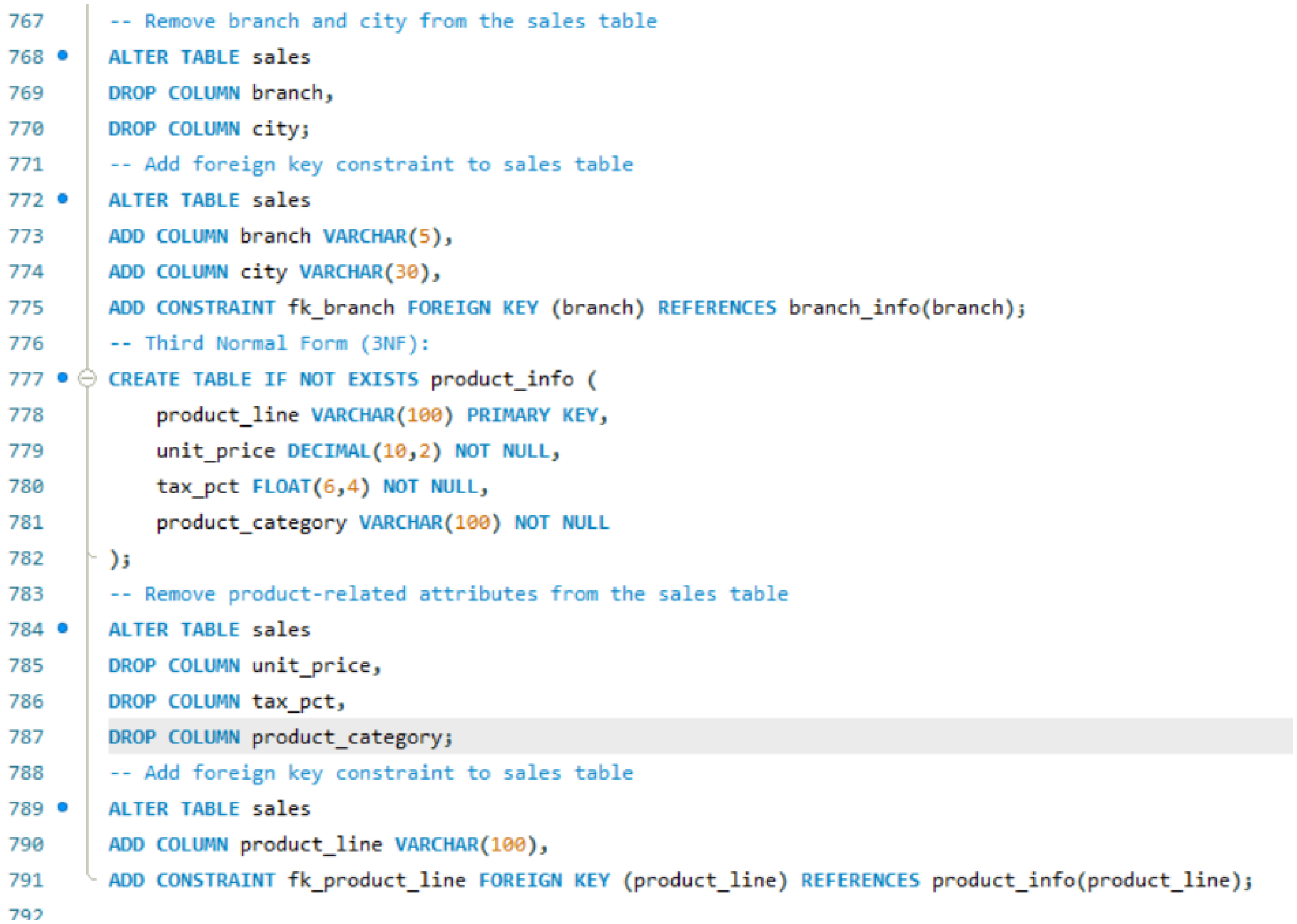
Verify that every determinant is a candidate key.

```
-- Second Normal Form (2NF):
```

```
-- Identify the primary key and attributes that are fully functional dependent on it
-- Extract any partial dependencies to separate tables
-- Assuming 'invoice_id' is the primary key

-- Create a table for branch information
CREATE TABLE IF NOT EXISTS branch_info (
    branch VARCHAR(5) PRIMARY KEY,
    city VARCHAR(30) NOT NULL
);

-- Remove branch and city from the sales table
ALTER TABLE sales
DROP COLUMN branch,
DROP COLUMN city;

-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN branch VARCHAR(5),
ADD COLUMN city VARCHAR(30),
ADD CONSTRAINT fk_branch FOREIGN KEY (branch) REFERENCES branch_info(branch);

-- Third Normal Form (3NF):
-- Check for transitive dependencies and move attributes to separate tables if necessary
-- Assuming 'product_line' determines 'unit_price', 'tax_pct', 'product_category'

-- Create a table for product information
CREATE TABLE IF NOT EXISTS product_info (
    product_line VARCHAR(100) PRIMARY KEY,
    unit_price DECIMAL(10,2) NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    product_category VARCHAR(100) NOT NULL
);

-- Remove product-related attributes from the sales table
ALTER TABLE sales
DROP COLUMN unit_price,
DROP COLUMN tax_pct,
DROP COLUMN product_category;

-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN product_line VARCHAR(100),
ADD CONSTRAINT fk_product_line FOREIGN KEY (product_line) REFERENCES product_info(product_line);
```



```sql
-- Second Normal Form (2NF):
-- Identify the primary key and attributes that are fully functional dependent on it
-- Extract any partial dependencies to separate tables
-- Assuming 'invoice_id' is the primary key

-- Create a table for branch information
CREATE TABLE IF NOT EXISTS branch_info (
    branch VARCHAR(5) PRIMARY KEY,
    city VARCHAR(30) NOT NULL
);
-- Remove branch and city from the sales table
ALTER TABLE sales
DROP COLUMN branch,
DROP COLUMN city;
-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN branch VARCHAR(5),
ADD COLUMN city VARCHAR(30),
ADD CONSTRAINT fk_branch FOREIGN KEY (branch) REFERENCES branch_info(branch);
-- Third Normal Form (3NF):
CREATE TABLE IF NOT EXISTS product_info (
    product_line VARCHAR(100) PRIMARY KEY,
    unit_price DECIMAL(10,2) NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    product_category VARCHAR(100) NOT NULL
);
-- Remove product-related attributes from the sales table
ALTER TABLE sales
```



```
-- Remove branch and city from the sales table
ALTER TABLE sales
DROP COLUMN branch,
DROP COLUMN city;
-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN branch VARCHAR(5),
ADD COLUMN city VARCHAR(30),
ADD CONSTRAINT fk_branch FOREIGN KEY (branch) REFERENCES branch_info(branch);
-- Third Normal Form (3NF):
CREATE TABLE IF NOT EXISTS product_info (
    product_line VARCHAR(100) PRIMARY KEY,
    unit_price DECIMAL(10,2) NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    product_category VARCHAR(100) NOT NULL
);
-- Remove product-related attributes from the sales table
ALTER TABLE sales
DROP COLUMN unit_price,
DROP COLUMN tax_pct,
DROP COLUMN product_category;
-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN product_line VARCHAR(100),
ADD CONSTRAINT fk_product_line FOREIGN KEY (product_line) REFERENCES product_info(product_line);
```

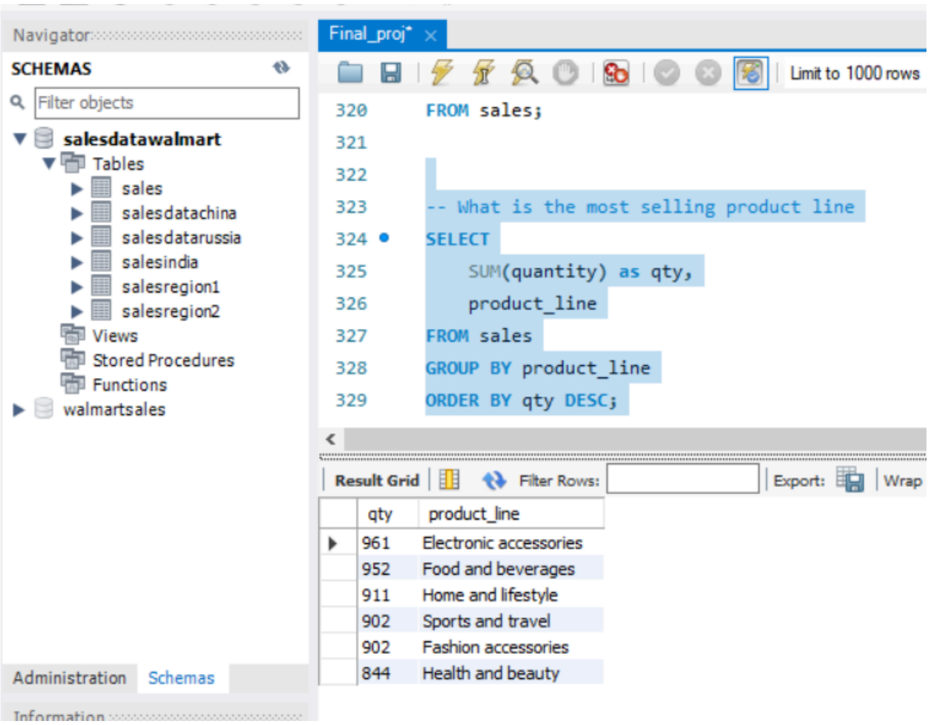
VI. SQL Queries

Using a DBMS software (SQLite3 or MySQL or any other of your choice):

- Create the tables
- Populate the tables (insert some meaningful data, at least 10 tuples for each relation)
- Run SQL queries (minimum 20) covering **all concepts** learned in the class

This section should contain the question, SQL code, and the output snapshot for each query.

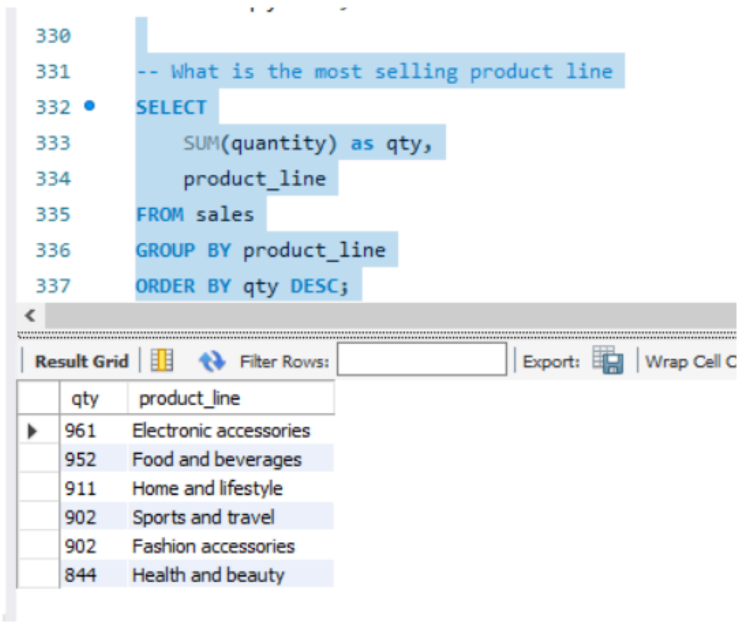
```
-- What is the most selling product line
SELECT
        SUM(quantity) as qty,
   product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;
```



```
FROM sales;


-- What is the most selling product line
SELECT
    SUM(quantity) as qty,
    product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;
```

qty	product_line
961	Electronic accessories
952	Food and beverages
911	Home and lifestyle
902	Sports and travel
902	Fashion accessories
844	Health and beauty

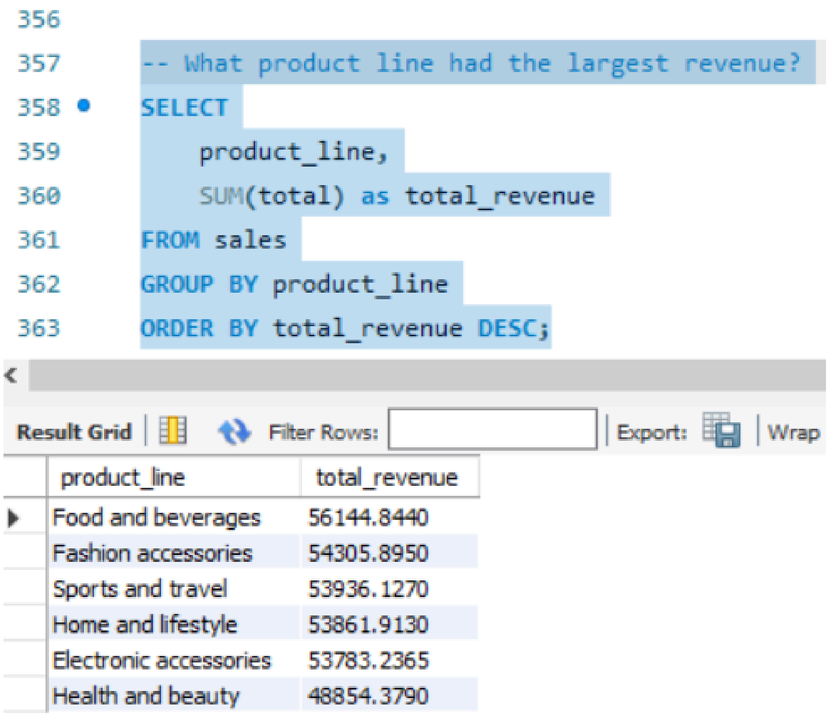
```
-- What is the most selling product line
SELECT
        SUM(quantity) as qty,
   product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;
```



```
-- What is the most selling product line
SELECT
    SUM(quantity) as qty,
    product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;
```

qty	product_line
961	Electronic accessories
952	Food and beverages
911	Home and lifestyle
902	Sports and travel
902	Fashion accessories
844	Health and beauty

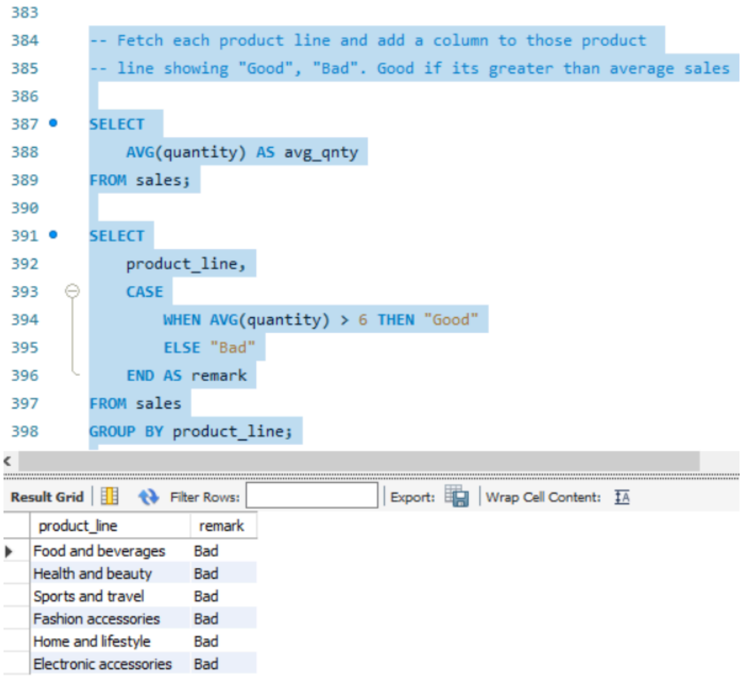
```
-- What product line had the largest revenue?
SELECT
        product_line,
        SUM(total) as total_revenue
FROM sales
GROUP BY product_line
ORDER BY total_revenue DESC;
```



```
-- What product line had the largest revenue?
SELECT
    product_line,
    SUM(total) as total_revenue
FROM sales
GROUP BY product_line
ORDER BY total_revenue DESC;
```

product_line	total_revenue
Food and beverages	56144.8440
Fashion accessories	54305.8950
Sports and travel	53936.1270
Home and lifestyle	53861.9130
Electronic accessories	53783.2365
Health and beauty	48854.3790

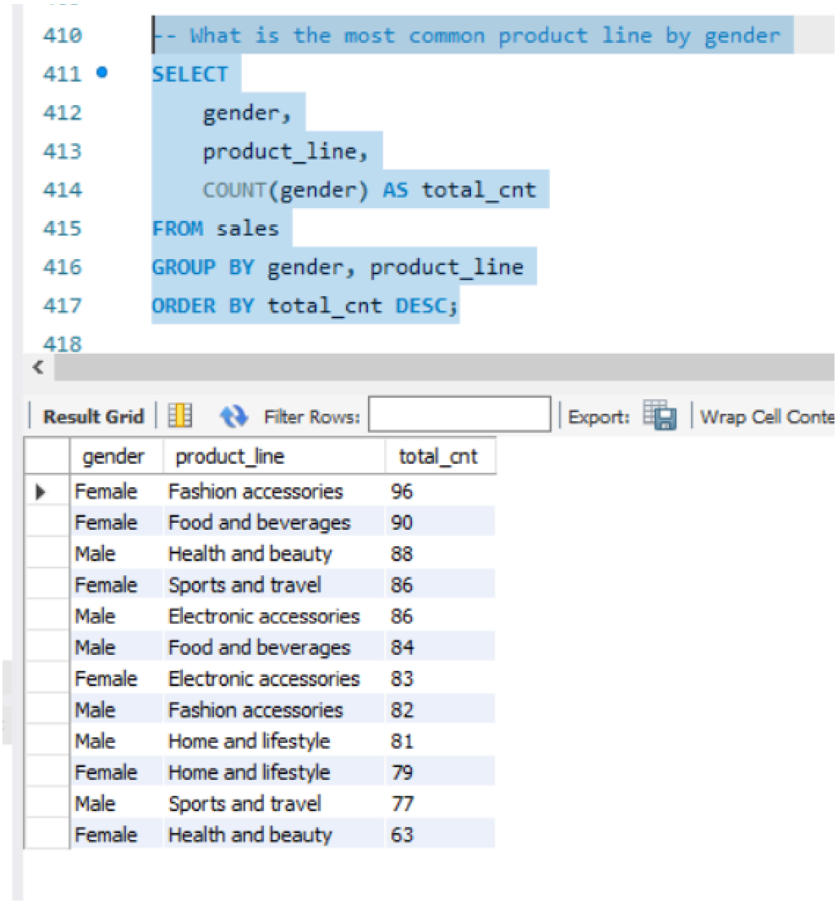
```
-- Fetch each product line and add a column to those product
-- line showing "Good", "Bad". Good if its greater than average sales

SELECT
        AVG(quantity) AS avg_qnty
FROM sales;

SELECT
        product_line,
        CASE
                WHEN AVG(quantity) > 6 THEN "Good"
      ELSE "Bad"
   END AS remark
FROM sales
GROUP BY product_line;
```



```
-- Fetch each product line and add a column to those product
-- line showing "Good", "Bad". Good if its greater than average sales

SELECT
    AVG(quantity) AS avg_qnty
FROM sales;

SELECT
    product_line,
    CASE
        WHEN AVG(quantity) > 6 THEN "Good"
        ELSE "Bad"
    END AS remark
FROM sales
GROUP BY product_line;
```

product_line	remark
Food and beverages	Bad
Health and beauty	Bad
Sports and travel	Bad
Fashion accessories	Bad
Home and lifestyle	Bad
Electronic accessories	Bad

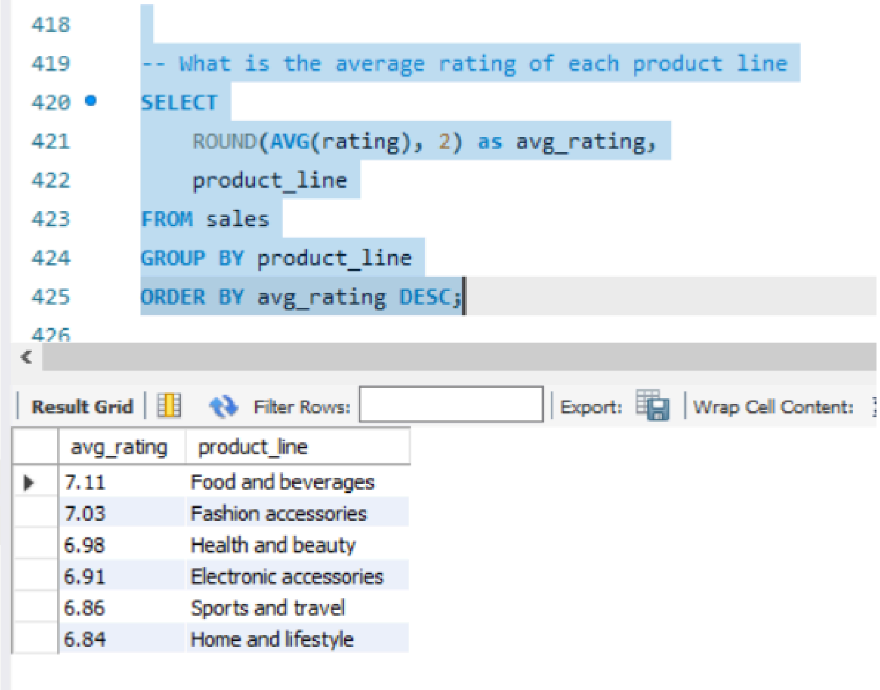
```
-- What is the most common product line by gender
SELECT
        gender,
   product_line,
   COUNT(gender) AS total_cnt
FROM sales
GROUP BY gender, product_line
ORDER BY total_cnt DESC;
```



```sql
-- What is the most common product line by gender
SELECT
    gender,
    product_line,
    COUNT(gender) AS total_cnt
FROM sales
GROUP BY gender, product_line
ORDER BY total_cnt DESC;
```

gender	product_line	total_cnt
Female	Fashion accessories	96
Female	Food and beverages	90
Male	Health and beauty	88
Female	Sports and travel	86
Male	Electronic accessories	86
Male	Food and beverages	84
Female	Electronic accessories	83
Male	Fashion accessories	82
Male	Home and lifestyle	81
Female	Home and lifestyle	79
Male	Sports and travel	77
Female	Health and beauty	63

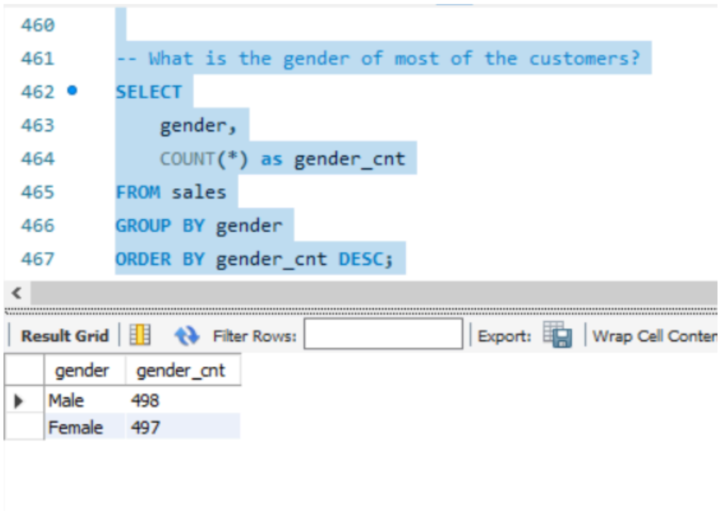
```sql
-- What is the average rating of each product line
SELECT
	ROUND(AVG(rating), 2) as avg_rating,
    product_line
FROM sales
GROUP BY product_line
ORDER BY avg_rating DESC;
```



```
-- What is the average rating of each product line
SELECT
    ROUND(AVG(rating), 2) as avg_rating,
    product_line
FROM sales
GROUP BY product_line
ORDER BY avg_rating DESC;
```

avg_rating	product_line
7.11	Food and beverages
7.03	Fashion accessories
6.98	Health and beauty
6.91	Electronic accessories
6.86	Sports and travel
6.84	Home and lifestyle

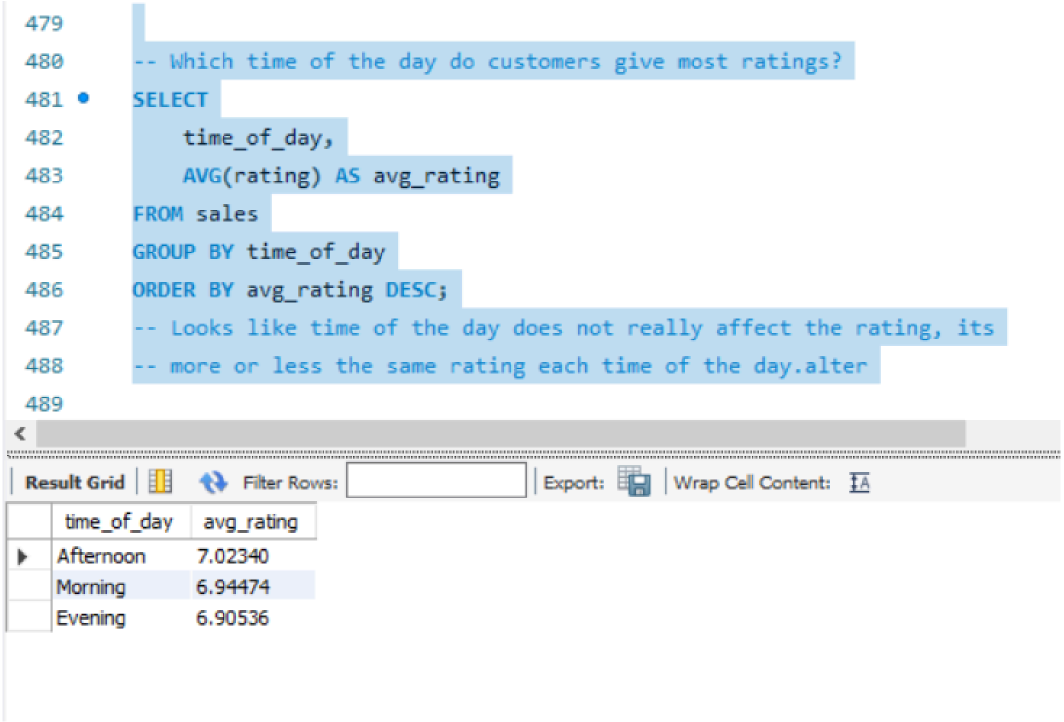
```
-- What is the gender of most of the customers?
SELECT
	gender,
	COUNT(*) as gender_cnt
FROM sales
GROUP BY gender
ORDER BY gender_cnt DESC;
```



```
-- What is the gender of most of the customers?
SELECT
    gender,
    COUNT(*) as gender_cnt
FROM sales
GROUP BY gender
ORDER BY gender_cnt DESC;
```

gender	gender_cnt
Male	498
Female	497

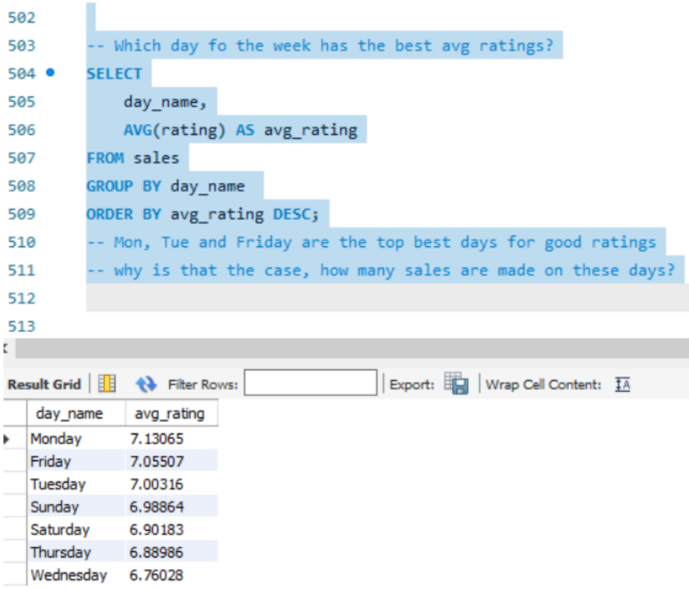
```
-- Which time of the day do customers give most ratings?
SELECT
	time_of_day,
	AVG(rating) AS avg_rating
FROM sales
GROUP BY time_of_day
ORDER BY avg_rating DESC;
-- Looks like time of the day does not really affect the rating, its
-- more or less the same rating each time of the day.alter
```



```
-- Which time of the day do customers give most ratings?
SELECT
    time_of_day,
    AVG(rating) AS avg_rating
FROM sales
GROUP BY time_of_day
ORDER BY avg_rating DESC;
-- Looks like time of the day does not really affect the rating, its
-- more or less the same rating each time of the day.alter
```

time_of_day	avg_rating
Afternoon	7.02340
Morning	6.94474
Evening	6.90536

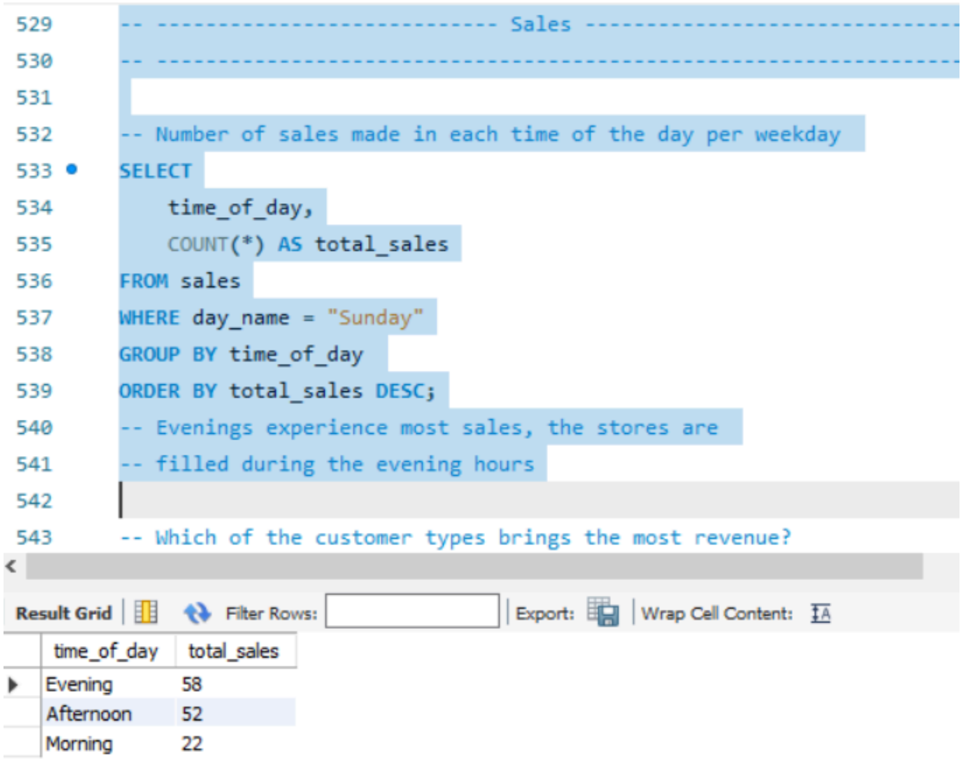
```
-- Which day fo the week has the best avg ratings?
SELECT
	day_name,
	AVG(rating) AS avg_rating
FROM sales
GROUP BY day_name
ORDER BY avg_rating DESC;
-- Mon, Tue and Friday are the top best days for good ratings
-- why is that the case, how many sales are made on these days?
```



```sql

-- Which day fo the week has the best avg ratings?
SELECT
    day_name,
    AVG(rating) AS avg_rating
FROM sales
GROUP BY day_name
ORDER BY avg_rating DESC;
-- Mon, Tue and Friday are the top best days for good ratings
-- why is that the case, how many sales are made on these days?

```

day_name	avg_rating
Monday	7.13065
Friday	7.05507
Tuesday	7.00316
Sunday	6.98864
Saturday	6.90183
Thursday	6.88986
Wednesday	6.76028

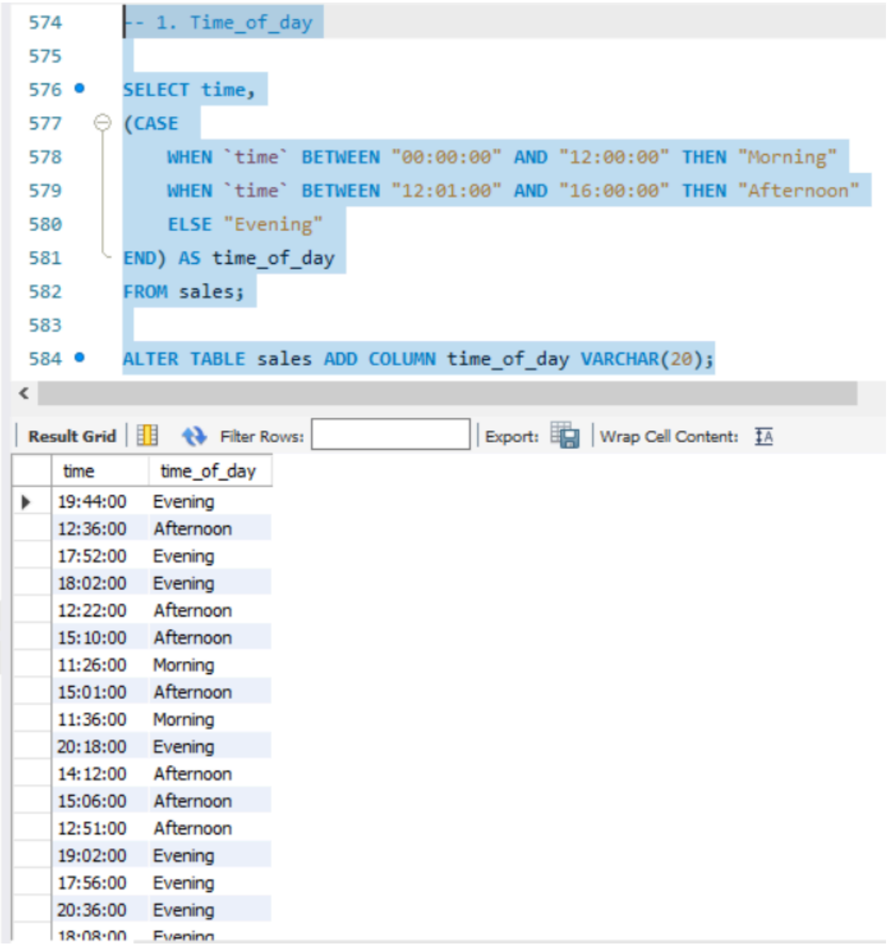
```sql
-- ---------------------------- Sales --------------------------------
-- -------------------------------------------------------------------

-- Number of sales made in each time of the day per weekday
SELECT
	time_of_day,
	COUNT(*) AS total_sales
FROM sales
WHERE day_name = "Sunday"
GROUP BY time_of_day
ORDER BY total_sales DESC;
-- Evenings experience most sales, the stores are
-- filled during the evening hours
```



```
-- ---------------------------- Sales -------------------------------
-- -----------------------------------------------------------------

-- Number of sales made in each time of the day per weekday
SELECT
    time_of_day,
    COUNT(*) AS total_sales
FROM sales
WHERE day_name = "Sunday"
GROUP BY time_of_day
ORDER BY total_sales DESC;
-- Evenings experience most sales, the stores are
-- filled during the evening hours

-- Which of the customer types brings the most revenue?
```

time_of_day	total_sales
Evening	58
Afternoon	52
Morning	22

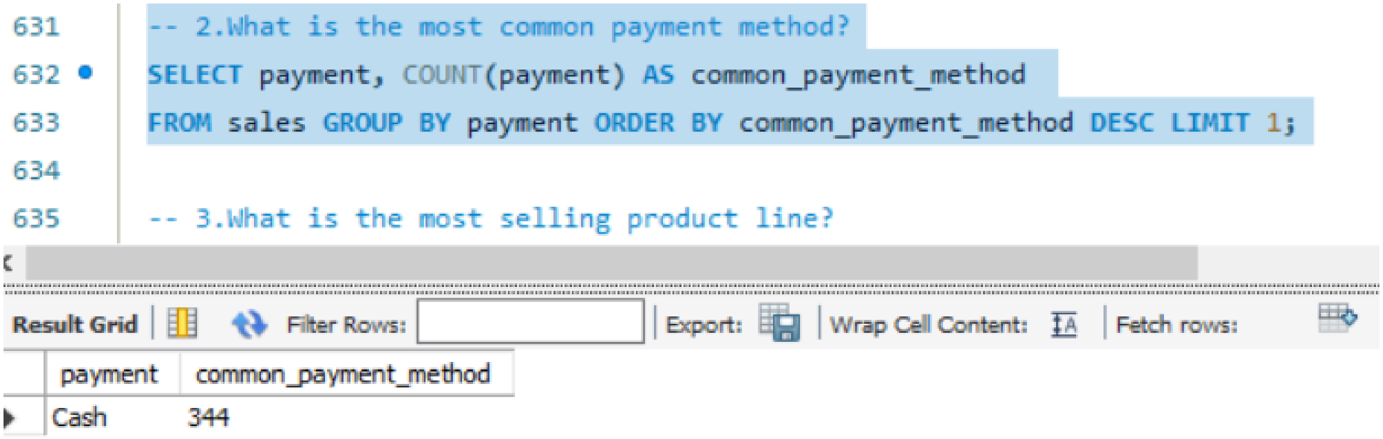
```
-- ------------------- Feature Engineering -----------------------------
-- 1. Time_of_day

SELECT time,
(CASE
        WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
END) AS time_of_day
FROM sales;

ALTER TABLE sales ADD COLUMN time_of_day VARCHAR(20);

UPDATE sales
SET time_of_day = (
        CASE
                WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
                WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
                ELSE "Evening"
        END
);
```



```sql
-- 1. Time_of_day

SELECT time,
(CASE
    WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
    WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
    ELSE "Evening"
END) AS time_of_day
FROM sales;

ALTER TABLE sales ADD COLUMN time_of_day VARCHAR(20);
```

time	time_of_day
19:44:00	Evening
12:36:00	Afternoon
17:52:00	Evening
18:02:00	Evening
12:22:00	Afternoon
15:10:00	Afternoon
11:26:00	Morning
15:01:00	Afternoon
11:36:00	Morning
20:18:00	Evening
14:12:00	Afternoon
15:06:00	Afternoon
12:51:00	Afternoon
19:02:00	Evening
17:56:00	Evening
20:36:00	Evening
18:08:00	Evening

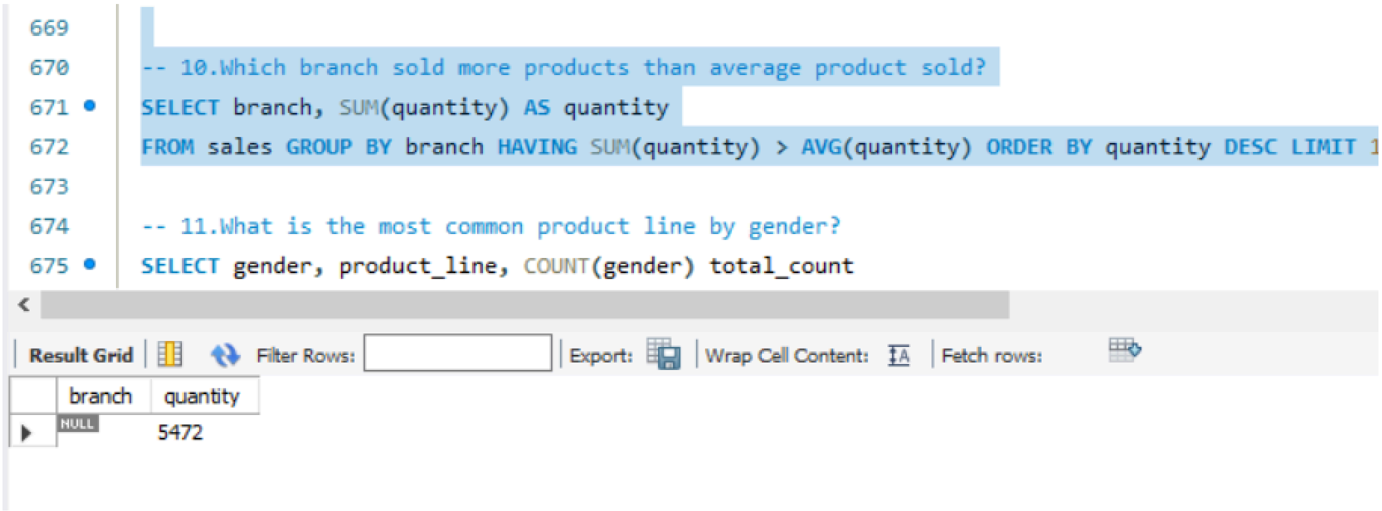
-- 2.What is the most common payment method?
SELECT payment, COUNT(payment) AS common_payment_method
FROM sales GROUP BY payment ORDER BY common_payment_method DESC LIMIT 1;



```sql
-- 2.What is the most common payment method?
SELECT payment, COUNT(payment) AS common_payment_method
FROM sales GROUP BY payment ORDER BY common_payment_method DESC LIMIT 1;

-- 3.What is the most selling product line?
```

payment	common_payment_method
Cash	344

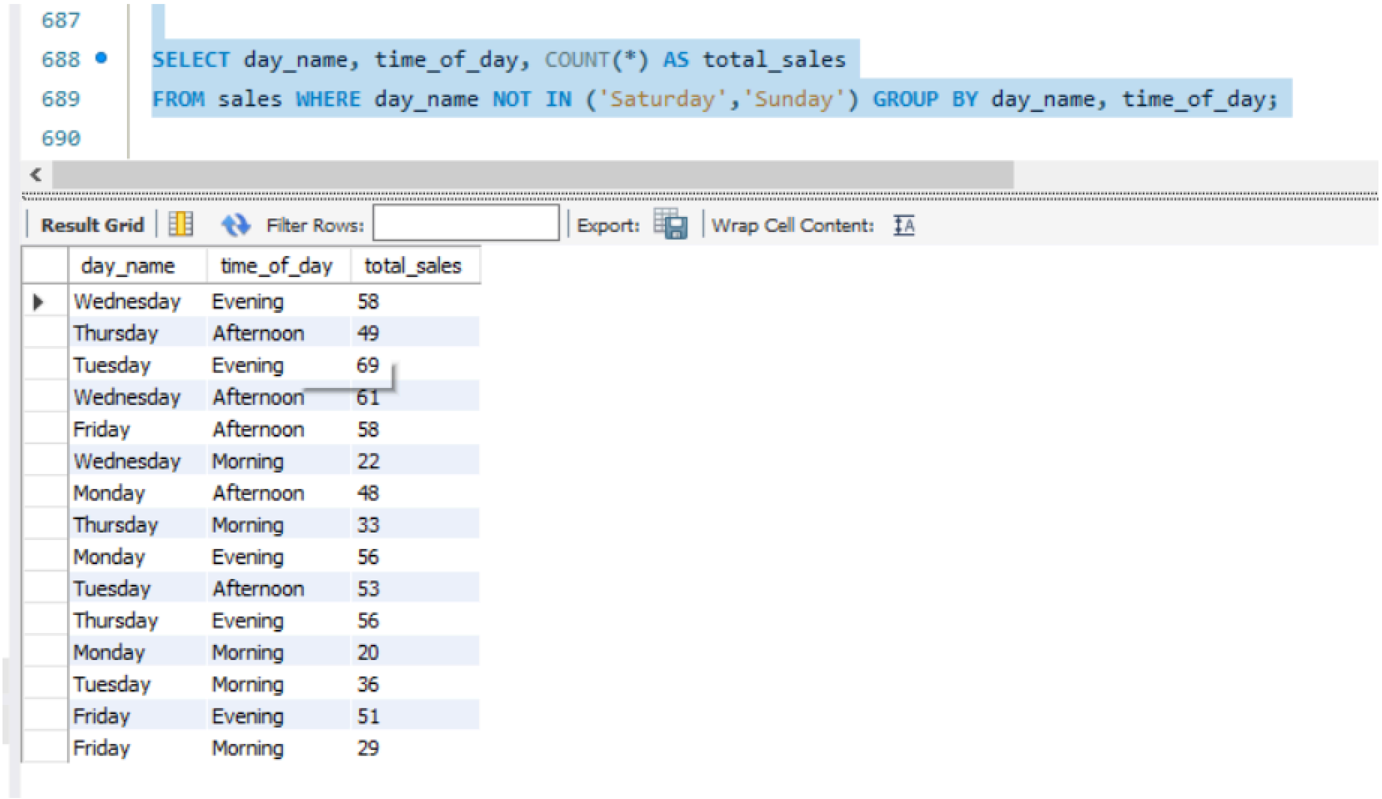
```
-- 10.Which branch sold more products than average product sold?
SELECT branch, SUM(quantity) AS quantity
FROM sales GROUP BY branch HAVING SUM(quantity) > AVG(quantity) ORDER BY quantity DESC
LIMIT 1;
```



```

-- 10.Which branch sold more products than average product sold?
SELECT branch, SUM(quantity) AS quantity
FROM sales GROUP BY branch HAVING SUM(quantity) > AVG(quantity) ORDER BY quantity DESC LIMIT 1

-- 11.What is the most common product line by gender?
SELECT gender, product_line, COUNT(gender) total_count
```

branch	quantity
NULL	5472

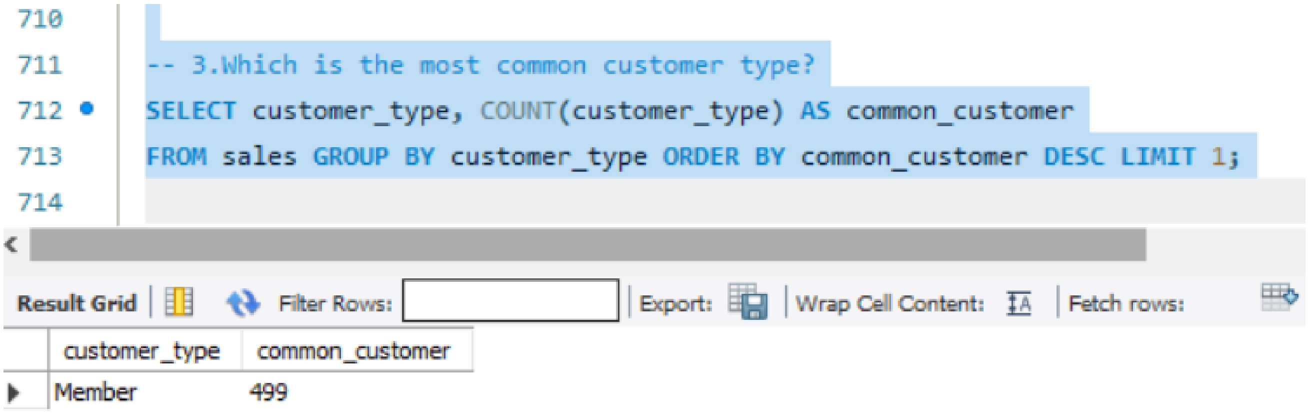
```
SELECT day_name, time_of_day, COUNT(*) AS total_sales
FROM sales WHERE day_name NOT IN ('Saturday','Sunday') GROUP BY day_name, time_of_day;
```



```
SELECT day_name, time_of_day, COUNT(*) AS total_sales
FROM sales WHERE day_name NOT IN ('Saturday','Sunday') GROUP BY day_name, time_of_day;
```

day_name	time_of_day	total_sales
Wednesday	Evening	58
Thursday	Afternoon	49
Tuesday	Evening	69
Wednesday	Afternoon	61
Friday	Afternoon	58
Wednesday	Morning	22
Monday	Afternoon	48
Thursday	Morning	33
Monday	Evening	56
Tuesday	Afternoon	53
Thursday	Evening	56
Monday	Morning	20
Tuesday	Morning	36
Friday	Evening	51
Friday	Morning	29

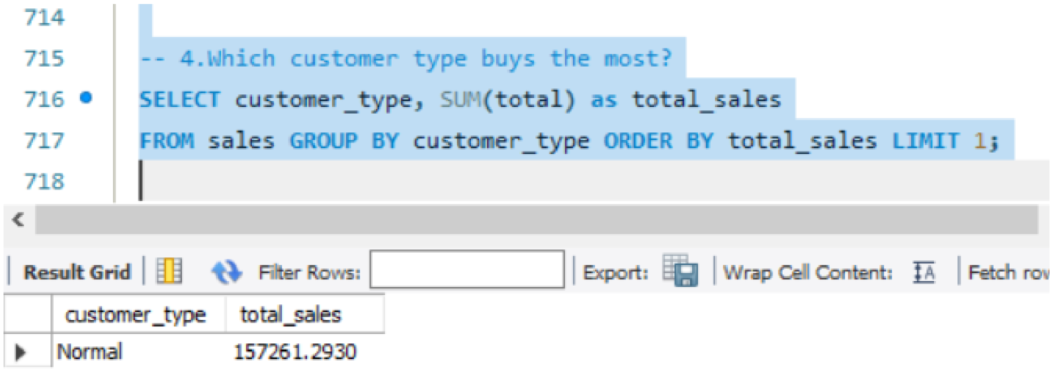
-- 3.Which is the most common customer type?
SELECT customer_type, COUNT(customer_type) AS common_customer
FROM sales GROUP BY customer_type ORDER BY common_customer DESC LIMIT 1;



```
-- 3.Which is the most common customer type?
SELECT customer_type, COUNT(customer_type) AS common_customer
FROM sales GROUP BY customer_type ORDER BY common_customer DESC LIMIT 1;
```

customer_type	common_customer
Member	499

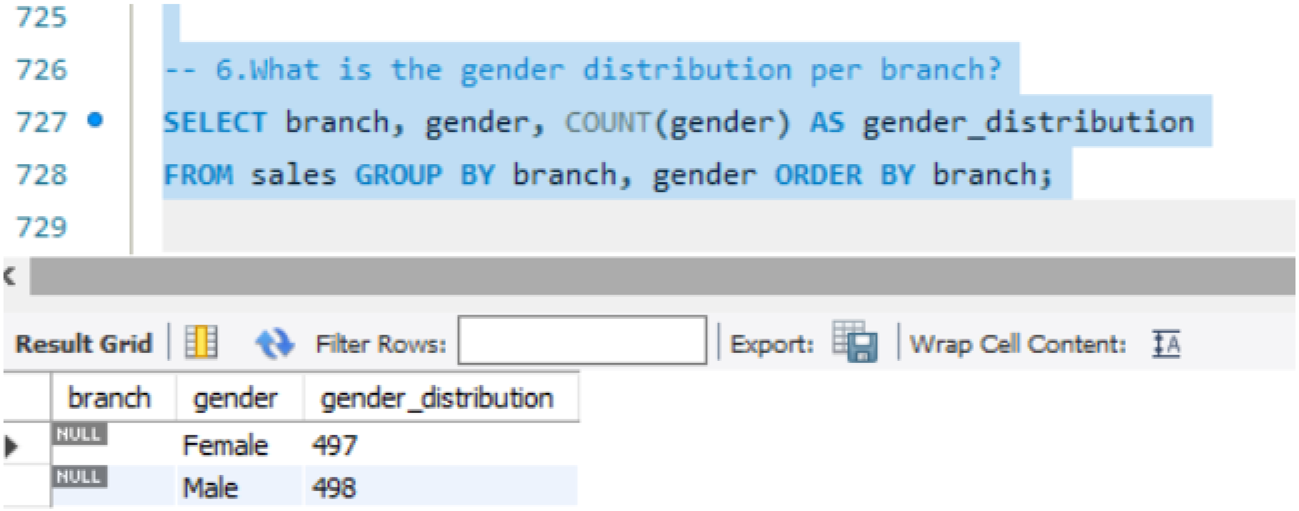
-- 4.Which customer type buys the most?
SELECT customer_type, SUM(total) as total_sales
FROM sales GROUP BY customer_type ORDER BY total_sales LIMIT 1;



```sql
-- 4.Which customer type buys the most?
SELECT customer_type, SUM(total) as total_sales
FROM sales GROUP BY customer_type ORDER BY total_sales LIMIT 1;
```

customer_type	total_sales
Normal	157261.2930

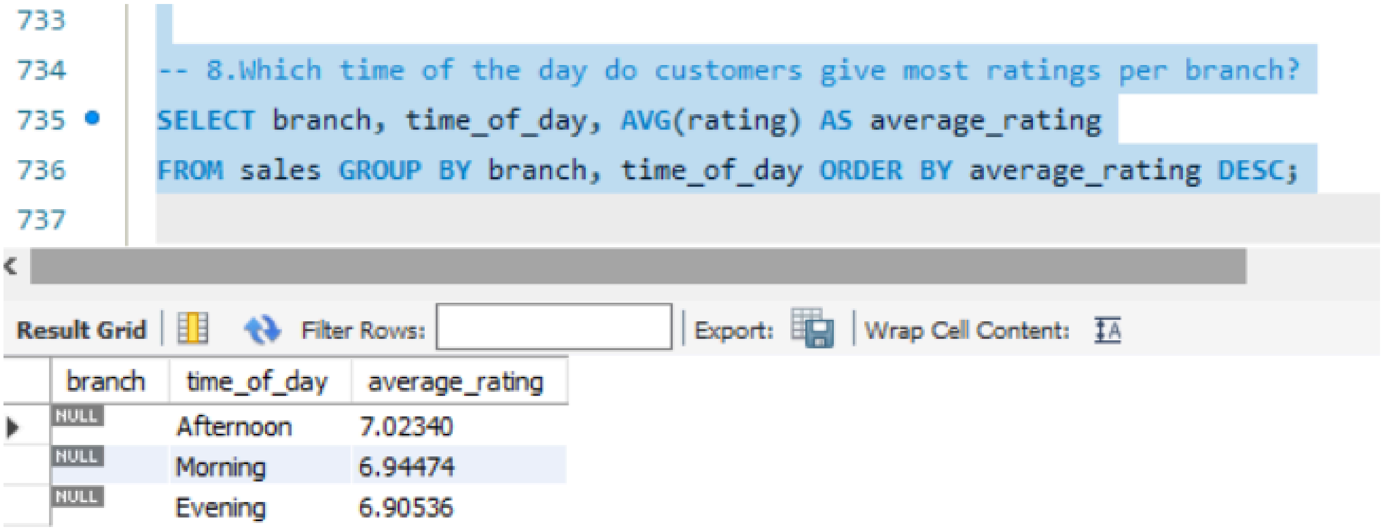
-- 6.What is the gender distribution per branch?
SELECT branch, gender, COUNT(gender) AS gender_distribution
FROM sales GROUP BY branch, gender ORDER BY branch;



```sql
-- 6.What is the gender distribution per branch?
SELECT branch, gender, COUNT(gender) AS gender_distribution
FROM sales GROUP BY branch, gender ORDER BY branch;
```

branch	gender	gender_distribution
NULL	Female	497
NULL	Male	498

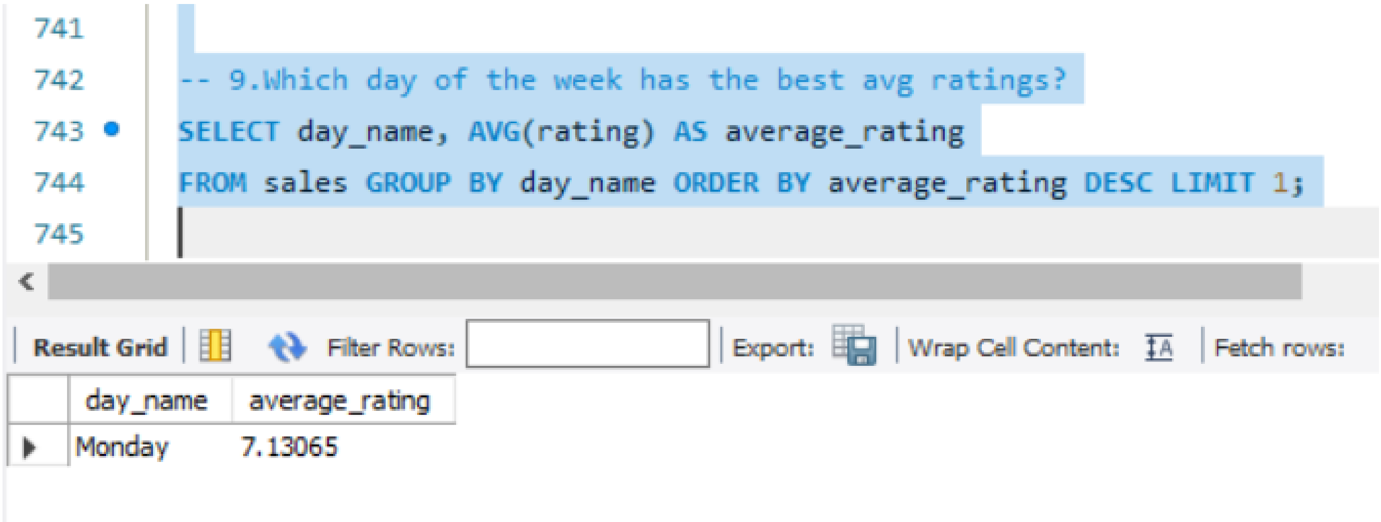
-- 8.Which time of the day do customers give most ratings per branch?
SELECT branch, time_of_day, AVG(rating) AS average_rating
FROM sales GROUP BY branch, time_of_day ORDER BY average_rating DESC;



```
-- 8.Which time of the day do customers give most ratings per branch?
SELECT branch, time_of_day, AVG(rating) AS average_rating
FROM sales GROUP BY branch, time_of_day ORDER BY average_rating DESC;
```

branch	time_of_day	average_rating
NULL	Afternoon	7.02340
NULL	Morning	6.94474
NULL	Evening	6.90536

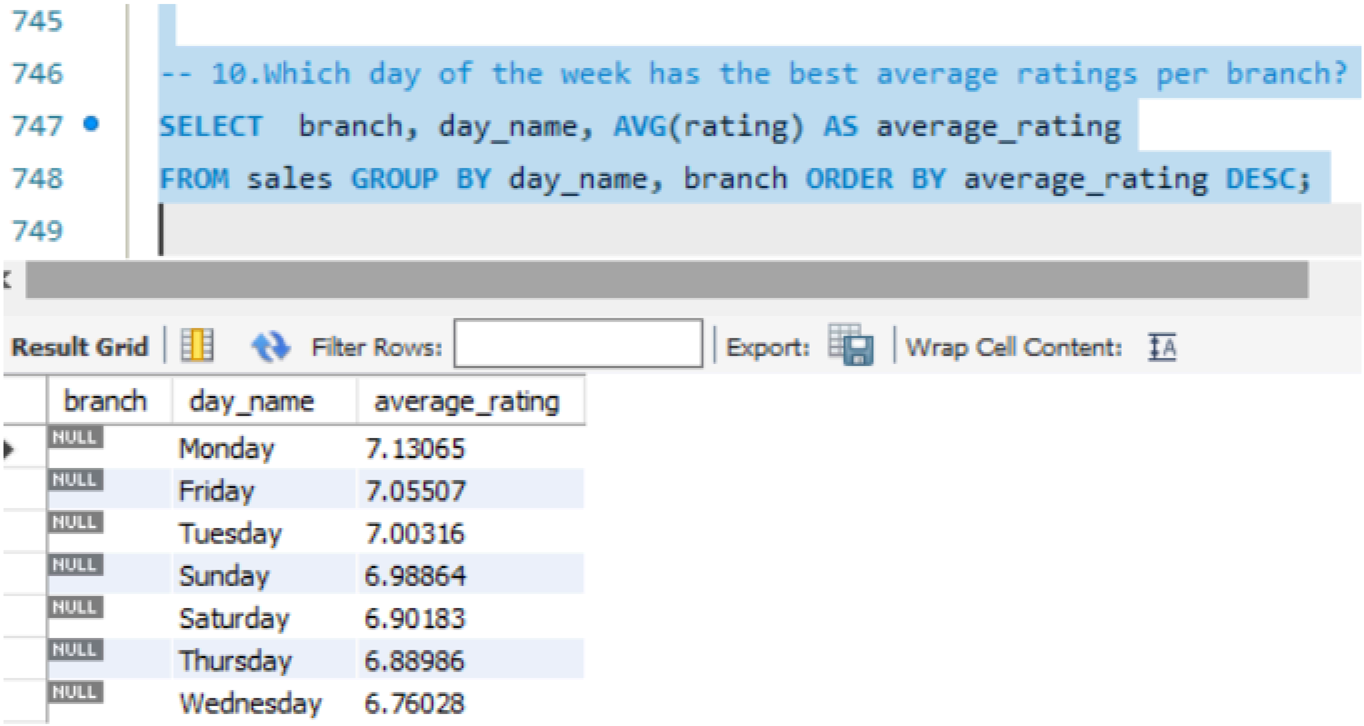
-- 9.Which day of the week has the best avg ratings?
SELECT day_name, AVG(rating) AS average_rating
FROM sales GROUP BY day_name ORDER BY average_rating DESC LIMIT 1;



```
-- 9.Which day of the week has the best avg ratings?
SELECT day_name, AVG(rating) AS average_rating
FROM sales GROUP BY day_name ORDER BY average_rating DESC LIMIT 1;
```

day_name	average_rating
Monday	7.13065

-- 10.Which day of the week has the best average ratings per branch?
SELECT branch, day_name, AVG(rating) AS average_rating
FROM sales GROUP BY day_name, branch ORDER BY average_rating DESC;



```sql
-- 10.Which day of the week has the best average ratings per branch?
SELECT  branch, day_name, AVG(rating) AS average_rating
FROM sales GROUP BY day_name, branch ORDER BY average_rating DESC;
```

branch	day_name	average_rating
NULL	Monday	7.13065
NULL	Friday	7.05507
NULL	Tuesday	7.00316
NULL	Sunday	6.98864
NULL	Saturday	6.90183
NULL	Thursday	6.88986
NULL	Wednesday	6.76028

VI. Project demonstration

- Tools/software/ libraries used
- Screenshot and Description of the Demonstration of project (If GUI is made)

```
-- Create database
CREATE DATABASE IF NOT EXISTS salesDataWalmart;

CREATE DATABASE IF NOT EXISTS walmartSales;

-- Create table
CREATE TABLE IF NOT EXISTS sales(
        invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
   branch VARCHAR(5) NOT NULL,
   city VARCHAR(30) NOT NULL,
   customer_type VARCHAR(30) NOT NULL,
   gender VARCHAR(30) NOT NULL,
   product_line VARCHAR(100) NOT NULL,
   unit_price DECIMAL(10,2) NOT NULL,
   quantity INT NOT NULL,
   tax_pct FLOAT(6,4) NOT NULL,
   total DECIMAL(12, 4) NOT NULL,
   date DATETIME NOT NULL,
   time TIME NOT NULL,
   payment VARCHAR(15) NOT NULL,
   cogs DECIMAL(10,2) NOT NULL,
   gross_margin_pct FLOAT(11,9),
   gross_income DECIMAL(12, 4),
   rating FLOAT(2, 1)
);

-- Data cleaning
SELECT * FROM salesDataWalmart.sales;




-- Create table for Indian region sales
CREATE TABLE IF NOT EXISTS salesIndia (
   invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
   branch VARCHAR(5) NOT NULL,
   city VARCHAR(30) NOT NULL,
   customer_type VARCHAR(30) NOT NULL,
   gender VARCHAR(30) NOT NULL,
   product_line VARCHAR(100) NOT NULL,
   unit_price DECIMAL(10,2) NOT NULL,
   quantity INT NOT NULL,
   tax_pct FLOAT(6,4) NOT NULL,
   total DECIMAL(12, 4) NOT NULL,
   date_purchased DATETIME NOT NULL,
   time_of_purchase TIME NOT NULL,
   payment_method VARCHAR(15) NOT NULL,
   cost_of_goods_sold DECIMAL(10,2) NOT NULL,
   gross_margin_pct FLOAT(11,9),
   gross_income DECIMAL(12, 4),
   customer_rating FLOAT(2, 1)
);

-- Create a relationship between the salesIndia table and the sales table
ALTER TABLE salesIndia
ADD CONSTRAINT fk_invoice_id_salesIndia
FOREIGN KEY (invoice_id) REFERENCES sales(invoice_id);

-- Create databases
CREATE DATABASE IF NOT EXISTS salesDataAfrica;
CREATE DATABASE IF NOT EXISTS salesDataChina;
CREATE DATABASE IF NOT EXISTS salesDataRussia;
CREATE DATABASE IF NOT EXISTS salesDataCanada;
CREATE DATABASE IF NOT EXISTS salesDataAustralia;
```

```
-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesAfrica(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating FLOAT(2, 1)
);

-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesDataChina(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating DECIMAL(10, 2)
);

-- Create tables for Africa
CREATE TABLE IF NOT EXISTS salesDataRussia(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date DATETIME NOT NULL,
    time TIME NOT NULL,
    payment VARCHAR(15) NOT NULL,
    cogs DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    rating INT
);

-- Create a relationship between the salesChina table and salesDataWalmart.sales table
ALTER TABLE salesChina
ADD CONSTRAINT fk_invoice_id_salesChina
FOREIGN KEY (invoice_id) REFERENCES salesDataWalmart.sales(invoice_id);

-- Create table for Chinese region sales
CREATE TABLE IF NOT EXISTS salesChina (
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date_purchased DATETIME NOT NULL,
    time_of_purchase TIME NOT NULL,
    payment_method VARCHAR(15) NOT NULL,
    cost_of_goods_sold DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    customer_rating FLOAT(2, 1)
);

-- Create table for Russian region sales
CREATE TABLE IF NOT EXISTS salesRussia (
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    branch VARCHAR(5) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_line VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total DECIMAL(12, 4) NOT NULL,
    date_purchased DATETIME NOT NULL,
    time_of_purchase TIME NOT NULL,
    payment_method VARCHAR(15) NOT NULL,
    cost_of_goods_sold DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    customer_rating FLOAT(2, 1)
);


-- Create relationships with the original sales table
ALTER TABLE salesRegion1
ADD CONSTRAINT fk_invoice_id_salesRegion1
FOREIGN KEY (invoice_id) REFERENCES sales(invoice_id);

ALTER TABLE salesRegion2
ADD CONSTRAINT fk_invoice_id_salesRegion2
FOREIGN KEY (invoice_id) REFERENCES sales(invoice_id);


-- Create a new table similar to 'sales' table
CREATE TABLE IF NOT EXISTS walmartSales.sales(
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    store_location VARCHAR(50) NOT NULL,
    city VARCHAR(30) NOT NULL,
    customer_type VARCHAR(30) NOT NULL,
    gender VARCHAR(30) NOT NULL,
    product_category VARCHAR(100) NOT NULL,
    unit_price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    total_amount DECIMAL(12, 4) NOT NULL,
    date_purchased DATETIME NOT NULL,
    time_of_purchase TIME NOT NULL,
    payment_method VARCHAR(15) NOT NULL,
    cost_of_goods_sold DECIMAL(10,2) NOT NULL,
    gross_margin_pct FLOAT(11,9),
    gross_income DECIMAL(12, 4),
    customer_rating FLOAT(2, 1)
);

-- Data cleaning
SELECT * FROM walmartSales.sales;

-- Joining both tables on the common primary key (invoice_id)
SELECT *
FROM sales s
JOIN walmartSales.sales ws ON s.invoice_id = ws.invoice_id;

-- Inserting sample data into walmartSales.sales table
INSERT INTO walmartSales.sales (invoice_id, store_location, city, customer_type, gender,
product_category, unit_price, quantity, tax_pct, total_amount, date_purchased, time_of_purchase,
payment_method, cost_of_goods_sold, gross_margin_pct, gross_income, customer_rating)
VALUES
('INV001', 'Store A', 'New York', 'Regular', 'Male', 'Electronics', 500.00, 2, 0.05, 1050.00, '2024-03-14
09:30:00', '09:30:00', 'Credit Card', 1000.00, 0.25, 50.00, 4.5),
('INV002', 'Store B', 'Los Angeles', 'Regular', 'Female', 'Fashion', 100.00, 3, 0.08, 324.00, '2024-03-15
14:45:00', '14:45:00', 'Cash', 240.00, 0.30, 84.00, 4.0),
('INV003', 'Store C', 'Chicago', 'VIP', 'Male', 'Home Appliances', 800.00, 1, 0.1, 880.00, '2024-03-16
17:20:00', '17:20:00', 'Debit Card', 800.00, 0.20, 80.00, 4.8);


-- Create a new table with invoice_id and time_of_day attributes
CREATE TABLE IF NOT EXISTS sales_with_time_of_day (
    invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,
    time_of_day VARCHAR(15) NOT NULL,
    FOREIGN KEY (invoice_id) REFERENCES sales(invoice_id)
);

-- Populate the new table with invoice_id and corresponding time_of_day values
INSERT INTO sales_with_time_of_day (invoice_id, time_of_day)
SELECT
    invoice_id,
    CASE
        WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
    END AS time_of_day
FROM sales;


-- Alter the sales_with_time_of_day table to add a foreign key constraint
```

```
ALTER TABLE sales_with_time_of_day
ADD CONSTRAINT fk_invoice_id
FOREIGN KEY (invoice_id)
REFERENCES sales(invoice_id);


-- Add the time_of_day column
SELECT
            time,
            (CASE
                        WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
    END) AS time_of_day
FROM sales;


ALTER TABLE sales ADD COLUMN time_of_day VARCHAR(20);

-- For this to work turn off safe mode for update
-- Edit > Preferences > SQL Edito > scroll down and toggle safe mode
-- Reconnect to MySQL: Query > Reconnect to server
UPDATE sales
SET time_of_day = (
            CASE
                        WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
    END
);


-- Add day_name column
SELECT
            date,
            DAYNAME(date) AS day_name
FROM sales;

ALTER TABLE sales ADD COLUMN day_name VARCHAR(10);

UPDATE sales
SET day_name = DAYNAME(date);


-- Add month_name column
SELECT
```

```
		date,
		MONTHNAME(date)
FROM sales;

ALTER TABLE sales ADD COLUMN month_name VARCHAR(10);

UPDATE sales
SET month_name = MONTHNAME(date);

-- --------------------------------------------------------------------
-- ---------------------------- Generic ------------------------------
-- --------------------------------------------------------------------
-- How many unique cities does the data have?
SELECT
		DISTINCT city
FROM sales;

-- In which city is each branch?
SELECT
		DISTINCT city,
   branch
FROM sales;

-- --------------------------------------------------------------------
-- ---------------------------- Product -------------------------------
-- --------------------------------------------------------------------

-- How many unique product lines does the data have?
SELECT
		DISTINCT product_line
FROM sales;


-- What is the most selling product line
SELECT
		SUM(quantity) as qty,
   product_line
FROM sales
GROUP BY product_line
ORDER BY qty DESC;

-- What is the most selling product line
SELECT
		SUM(quantity) as qty,
   product_line
FROM sales
```

```
GROUP BY product_line
ORDER BY qty DESC;

-- What is the total revenue by month
SELECT
        month_name AS month,
        SUM(total) AS total_revenue
FROM sales
GROUP BY month_name
ORDER BY total_revenue;


-- What month had the largest COGS?
SELECT
        month_name AS month,
        SUM(cogs) AS cogs
FROM sales
GROUP BY month_name
ORDER BY cogs;


-- What product line had the largest revenue?
SELECT
        product_line,
        SUM(total) as total_revenue
FROM sales
GROUP BY product_line
ORDER BY total_revenue DESC;

-- What is the city with the largest revenue?
SELECT
        branch,
        city,
        SUM(total) AS total_revenue
FROM sales
GROUP BY city, branch
ORDER BY total_revenue;


-- What product line had the largest VAT?
SELECT
        product_line,
        AVG(tax_pct) as avg_tax
FROM sales
GROUP BY product_line
ORDER BY avg_tax DESC;
```

```
-- Fetch each product line and add a column to those product
-- line showing "Good", "Bad". Good if its greater than average sales

SELECT
        AVG(quantity) AS avg_qnty
FROM sales;

SELECT
        product_line,
        CASE
                WHEN AVG(quantity) > 6 THEN "Good"
      ELSE "Bad"
   END AS remark
FROM sales
GROUP BY product_line;


-- Which branch sold more products than average product sold?
SELECT
        branch,
   SUM(quantity) AS qnty
FROM sales
GROUP BY branch
HAVING SUM(quantity) > (SELECT AVG(quantity) FROM sales);


-- What is the most common product line by gender
SELECT
        gender,
   product_line,
   COUNT(gender) AS total_cnt
FROM sales
GROUP BY gender, product_line
ORDER BY total_cnt DESC;

-- What is the average rating of each product line
SELECT
        ROUND(AVG(rating), 2) as avg_rating,
   product_line
FROM sales
GROUP BY product_line
ORDER BY avg_rating DESC;

-- --------------------------------------------------------------
-- -----------------------------------------------------------------------

-- -----------------------------------------------------------------------
-- --------------------------- Customers ---------------------------------
-- -----------------------------------------------------------------------

-- How many unique customer types does the data have?
SELECT
        DISTINCT customer_type
FROM sales;

-- How many unique payment methods does the data have?
SELECT
        DISTINCT payment
FROM sales;


-- What is the most common customer type?
SELECT
        customer_type,
        count(*) as count
FROM sales
GROUP BY customer_type
ORDER BY count DESC;

-- Which customer type buys the most?
SELECT
        customer_type,
   COUNT(*)
FROM sales
GROUP BY customer_type;


-- What is the gender of most of the customers?
SELECT
        gender,
        COUNT(*) as gender_cnt
FROM sales
GROUP BY gender
ORDER BY gender_cnt DESC;

-- What is the gender distribution per branch?
SELECT
        gender,
        COUNT(*) as gender_cnt
FROM sales
```

```
WHERE branch = "C"
GROUP BY gender
ORDER BY gender_cnt DESC;
-- Gender per branch is more or less the same hence, I don't think has
-- an effect of the sales per branch and other factors.

-- Which time of the day do customers give most ratings?
SELECT
	time_of_day,
	AVG(rating) AS avg_rating
FROM sales
GROUP BY time_of_day
ORDER BY avg_rating DESC;
-- Looks like time of the day does not really affect the rating, its
-- more or less the same rating each time of the day.alter


-- Which time of the day do customers give most ratings per branch?
SELECT
	time_of_day,
	AVG(rating) AS avg_rating
FROM sales
WHERE branch = "A"
GROUP BY time_of_day
ORDER BY avg_rating DESC;
-- Branch A and C are doing well in ratings, branch B needs to do a
-- little more to get better ratings.


-- Which day fo the week has the best avg ratings?
SELECT
	day_name,
	AVG(rating) AS avg_rating
FROM sales
GROUP BY day_name
ORDER BY avg_rating DESC;
-- Mon, Tue and Friday are the top best days for good ratings
-- why is that the case, how many sales are made on these days?



-- Which day of the week has the best average ratings per branch?
SELECT
	day_name,
	COUNT(day_name) total_sales
FROM sales
```

```
WHERE branch = "C"
GROUP BY day_name
ORDER BY total_sales DESC;


-- ------------------------------------------------------------------
-- ------------------------------------------------------------------


-- ------------------------------------------------------------------
-- --------------------------- Sales --------------------------------
-- ------------------------------------------------------------------

-- Number of sales made in each time of the day per weekday
SELECT
	time_of_day,
	COUNT(*) AS total_sales
FROM sales
WHERE day_name = "Sunday"
GROUP BY time_of_day
ORDER BY total_sales DESC;
-- Evenings experience most sales, the stores are
-- filled during the evening hours

-- Which of the customer types brings the most revenue?
SELECT
	customer_type,
	SUM(total) AS total_revenue
FROM sales
GROUP BY customer_type
ORDER BY total_revenue;

-- Which city has the largest tax/VAT percent?
SELECT
	city,
   ROUND(AVG(tax_pct), 2) AS avg_tax_pct
FROM sales
GROUP BY city
ORDER BY avg_tax_pct DESC;

-- Which customer type pays the most in VAT?
SELECT
	customer_type,
	AVG(tax_pct) AS total_tax
FROM sales
GROUP BY customer_type
ORDER BY total_tax;
```

```
-- ----------------------------------------------------------------------
-- ----------------------------------------------------------------------



-- ------------------- Feature Engineering ----------------------------
-- 1. Time_of_day

SELECT time,
(CASE
        WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
        WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
        ELSE "Evening"
END) AS time_of_day
FROM sales;

ALTER TABLE sales ADD COLUMN time_of_day VARCHAR(20);

UPDATE sales
SET time_of_day = (
        CASE
                WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
                WHEN `time` BETWEEN "12:01:00" AND "16:00:00" THEN "Afternoon"
                ELSE "Evening"
        END
);


-- 2.Day_name

SELECT date,
DAYNAME(date) AS day_name
FROM sales;

ALTER TABLE sales ADD COLUMN day_name VARCHAR(10);

UPDATE sales
SET day_name = DAYNAME(date);

-- 3.Momth_name

SELECT date,
MONTHNAME(date) AS month_name
FROM sales;
```

```
ALTER TABLE sales ADD COLUMN month_name VARCHAR(10);

UPDATE sales
SET month_name = MONTHNAME(date);


-- ----------------Exploratory Data Analysis (EDA)----------------------
-- Generic Questions
-- 1.How many distinct cities are present in the dataset?
SELECT DISTINCT city FROM sales;

-- 2.In which city is each branch situated?
SELECT DISTINCT branch, city FROM sales;

-- Product Analysis
-- 1.How many distinct product lines are there in the dataset?
SELECT COUNT(DISTINCT product_line) FROM sales;

-- 2.What is the most common payment method?
SELECT payment, COUNT(payment) AS common_payment_method
FROM sales GROUP BY payment ORDER BY common_payment_method DESC LIMIT 1;

-- 3.What is the most selling product line?
SELECT product_line, count(product_Line) AS most_selling_product
FROM sales GROUP BY product_line ORDER BY most_selling_product DESC LIMIT 1;

-- 4.What is the total revenue by month?
SELECT month_name, SUM(total) AS total_revenue
FROM SALES GROUP BY month_name ORDER BY total_revenue DESC;

-- 5.Which month recorded the highest Cost of Goods Sold (COGS)?
SELECT month_name, SUM(cogs) AS total_cogs
FROM sales GROUP BY month_name ORDER BY total_cogs DESC;

-- 6.Which product line generated the highest revenue?
SELECT product_line, SUM(total) AS total_revenue
FROM sales GROUP BY product_line ORDER BY total_revenue DESC LIMIT 1;

-- 7.Which city has the highest revenue?
SELECT city, SUM(total) AS total_revenue
FROM sales GROUP BY city ORDER BY total_revenue DESC LIMIT 1;

-- 8.Which product line incurred the highest VAT?
SELECT product_line, SUM(vat) as VAT
FROM sales GROUP BY product_line ORDER BY VAT DESC LIMIT 1;
```

```
-- 9.Retrieve each product line and add a column product_category, indicating 'Good' or 'Bad,'based on
whether its sales are above the average.

ALTER TABLE sales ADD COLUMN product_category VARCHAR(20);

UPDATE sales
SET product_category=
(CASE
        WHEN total >= (SELECT AVG(total) FROM sales) THEN "Good"
   ELSE "Bad"
END)FROM sales;

-- 10.Which branch sold more products than average product sold?
SELECT branch, SUM(quantity) AS quantity
FROM sales GROUP BY branch HAVING SUM(quantity) > AVG(quantity) ORDER BY quantity DESC
LIMIT 1;

-- 11.What is the most common product line by gender?
SELECT gender, product_line, COUNT(gender) total_count
FROM sales GROUP BY gender, product_line ORDER BY total_count DESC;

-- 12.What is the average rating of each product line?
SELECT product_line, ROUND(AVG(rating),2) average_rating
FROM sales GROUP BY product_line ORDER BY average_rating DESC;


-- Sales Analysis
1.Number of sales made in each time of the day per weekday
SELECT day_name, time_of_day, COUNT(invoice_id) AS total_sales
FROM sales GROUP BY day_name, time_of_day HAVING day_name NOT IN ('Sunday','Saturday');

SELECT day_name, time_of_day, COUNT(*) AS total_sales
FROM sales WHERE day_name NOT IN ('Saturday','Sunday') GROUP BY day_name, time_of_day;

-- 2.Identify the customer type that generates the highest revenue.
SELECT customer_type, SUM(total) AS total_sales
FROM sales GROUP BY customer_type ORDER BY total_sales DESC LIMIT 1;

-- 3.Which city has the largest tax percent/ VAT (Value Added Tax)?
SELECT city, SUM(VAT) AS total_VAT
FROM sales GROUP BY city ORDER BY total_VAT DESC LIMIT 1;

-- 4.Which customer type pays the most in VAT?
SELECT customer_type, SUM(VAT) AS total_VAT
FROM sales GROUP BY customer_type ORDER BY total_VAT DESC LIMIT 1;
```

Customer Analysis

```
-- 1.How many unique customer types does the data have?
SELECT COUNT(DISTINCT customer_type) FROM sales;

-- 2.How many unique payment methods does the data have?
SELECT COUNT(DISTINCT payment) FROM sales;

-- 3.Which is the most common customer type?
SELECT customer_type, COUNT(customer_type) AS common_customer
FROM sales GROUP BY customer_type ORDER BY common_customer DESC LIMIT 1;

-- 4.Which customer type buys the most?
SELECT customer_type, SUM(total) as total_sales
FROM sales GROUP BY customer_type ORDER BY total_sales LIMIT 1;

SELECT customer_type, COUNT(*) AS most_buyer
FROM sales GROUP BY customer_type ORDER BY most_buyer DESC LIMIT 1;

-- 5.What is the gender of most of the customers?
SELECT gender, COUNT(*) AS all_genders
FROM sales GROUP BY gender ORDER BY all_genders DESC LIMIT 1;

-- 6.What is the gender distribution per branch?
SELECT branch, gender, COUNT(gender) AS gender_distribution
FROM sales GROUP BY branch, gender ORDER BY branch;

-- 7.Which time of the day do customers give most ratings?
SELECT time_of_day, AVG(rating) AS average_rating
FROM sales GROUP BY time_of_day ORDER BY average_rating DESC LIMIT 1;

-- 8.Which time of the day do customers give most ratings per branch?
SELECT branch, time_of_day, AVG(rating) AS average_rating
FROM sales GROUP BY branch, time_of_day ORDER BY average_rating DESC;

SELECT branch, time_of_day,
AVG(rating) OVER(PARTITION BY branch) AS ratings
FROM sales GROUP BY branch;

-- 9.Which day of the week has the best avg ratings?
SELECT day_name, AVG(rating) AS average_rating
FROM sales GROUP BY day_name ORDER BY average_rating DESC LIMIT 1;

-- 10.Which day of the week has the best average ratings per branch?
SELECT  branch, day_name, AVG(rating) AS average_rating
```

```
FROM sales GROUP BY day_name, branch ORDER BY average_rating DESC;

SELECT  branch, day_name,
AVG(rating) OVER(PARTITION BY branch) AS rating
FROM sales GROUP BY branch ORDER BY rating DESC;




-- Second Normal Form (2NF):
-- Identify the primary key and attributes that are fully functional dependent on it
-- Extract any partial dependencies to separate tables
-- Assuming 'invoice_id' is the primary key

-- Create a table for branch information
CREATE TABLE IF NOT EXISTS branch_info (
    branch VARCHAR(5) PRIMARY KEY,
    city VARCHAR(30) NOT NULL
);
-- Remove branch and city from the sales table
ALTER TABLE sales
DROP COLUMN branch,
DROP COLUMN city;
-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN branch VARCHAR(5),
ADD COLUMN city VARCHAR(30),
ADD CONSTRAINT fk_branch FOREIGN KEY (branch) REFERENCES branch_info(branch);
-- Third Normal Form (3NF):
CREATE TABLE IF NOT EXISTS product_info (
    product_line VARCHAR(100) PRIMARY KEY,
    unit_price DECIMAL(10,2) NOT NULL,
    tax_pct FLOAT(6,4) NOT NULL,
    product_category VARCHAR(100) NOT NULL
);
-- Remove product-related attributes from the sales table
ALTER TABLE sales
DROP COLUMN unit_price,
DROP COLUMN tax_pct,
DROP COLUMN product_category;
-- Add foreign key constraint to sales table
ALTER TABLE sales
ADD COLUMN product_line VARCHAR(100),
ADD CONSTRAINT fk_product_line FOREIGN KEY (product_line) REFERENCES
product_info(product_line);
```

VII. Self -Learning beyond classroom

1. Learned about advanced SQL queries for data analysis and normalization.
2. Explored methods for data cleaning and feature engineering.
3. Gained insights into database design principles and normalization techniques.
4. Improved skills in exploratory data analysis and deriving actionable insights from data.

VIII. Learning from the Project

1. Enhanced understanding of database management and SQL queries.
2. Practiced applying normalization techniques to improve database efficiency and structure.
3. Developed proficiency in data analysis and visualization using SQL.

IX. Challenges Faced

1. Understanding and implementing normalization steps effectively.
2. Dealing with complex data analysis queries to derive meaningful insights.
3. Ensuring data consistency and accuracy during the cleaning process.

X. Conclusion

This project provided valuable hands-on experience in database management, normalization, and data analysis using SQL. Through exploring real-world datasets, I gained practical insights into database design principles, data cleaning techniques, and exploratory data analysis. The project enhanced my problem-solving skills and deepened my understanding of SQL queries and database optimization. Overall, it was a rewarding learning experience that contributed to my skill development in data management and analysis.