1. Products Table

The Products table contains details about products, including their names, categories, and unit

prices. It provides reference data for linking product information to sales transactions.

Query:

-- Create Products table

CREATE TABLE Products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category VARCHAR(50),

unit\_price DECIMAL(10, 2)

);

-- Insert sample data into Products table

INSERT INTO Products (product\_id, product\_name, category, unit\_price) VALUES

(101, &#39;Laptop&#39;, &#39;Electronics&#39;, 500.00),

(102, &#39;Smartphone&#39;, &#39;Electronics&#39;, 300.00),

(103, &#39;Headphones&#39;, &#39;Electronics&#39;, 30.00),

(104, &#39;Keyboard&#39;, &#39;Electronics&#39;, 20.00),

(105, &#39;Mouse&#39;, &#39;Electronics&#39;, 15.00);

1. Retrieve all columns from the product table.

2. Retrieve the product\_name and unit\_price from the Products table.

3. Filter the Products table to show only products in the &#39;Electronics&#39; category.

4. Retrieve the product\_id and product\_name from the Products table for products with a

unit\_price greater than $100.

5. Calculate the average unit\_price of products in the Products table.

6. Retrieve product\_name and unit\_price from the Products table with the Highest Unit Price

7. Retrieve the product\_name and unit\_price from the Products table, ordering the results by

unit\_price in descending order.

8. Retrieve the product\_name and unit\_price from the Products table, filtering the unit\_price to show only values between $20 and $600.

9. Retrieve the product\_name and category from the Products table, ordering the results by

category in ascending order.

**SOLUTION:**

CREATE DATABASE products;

USE products;

CREATE TABLE products(

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category VARCHAR(50),

unit\_price DECIMAL(10,2));

INSERT INTO products VALUES

(101, 'Laptop', 'Electronics', 500.00),

(102, 'Smartphone', 'Electronics', 300.00),

(103, 'Headphone', 'Electronics', 30.00),

(104, 'Keyboard', 'Electronics', 20.00),

(105, 'Mouse', 'Electronics', 15.00);

SELECT \* FROM products

SELECT product\_name, unit\_price FROM products;

SELECT product\_name,product\_id,unit\_price FROM products

WHERE unit\_price>100;

SELECT AVG(unit\_price) AS average\_price FROM products;

SELECT product\_name, unit\_price FROM products

WHERE unit\_price = (SELECT MAX(unit\_price) FROM products);

SELECT product\_name, unit\_price FROM products

ORDER BY unit\_price DESC;

SELECT product\_name, unit\_price FROM products

WHERE unit\_price BETWEEN 20 AND 600;

SELECT product\_name, category FROM products

ORDER BY category ASC;

1. Sales Table

The Sales table records information about product sales, including the quantity sold, sale date,

and total price for each sale. It serves as a transactional data source for analyzing sales trends.

Query:

-- Create Sales table

CREATE TABLE Sales (

sale\_id INT PRIMARY KEY,

product\_id INT,

quantity\_sold INT,

sale\_date DATE,

total\_price DECIMAL(10, 2)

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

-- Insert sample data into Sales table

INSERT INTO Sales (sale\_id, product\_id, quantity\_sold, sale\_date, total\_price) VALUES

(1, 101, 5, ‘2024-01-01’, 2500.00),

(2, 102, 3, ‘2024-01-02’, 900.00),

(3, 103, 2, ‘2024-01-02’, 60.00),

(4, 104, 4, ‘2024-01-03’, 80.00),

(5, 105, 6, ‘2024-01-03’, 90.00);

1. Retrieve all columns from the Sales table.

2. Retrieve the sale\_id and sale\_date from the Sales table.

3. Filter the Sales table to show only sales with a total\_price greater than $100.

4. Retrieve the sale\_id and total\_price from the Sales table for sales made on January 3, 2024.

5. Calculate the total revenue generated from all sales in the Sales table.

6. Calculate the total quantity\_sold from the Sales table.

7. Retrieve the sale\_id, product\_id, and total\_price from the Sales table for sales with a quantity\_sold greater than 4.

8.  Calculate the average total\_price of sales in the Sales table.

**SOLUTION:**

CREATE DATABASE Sales;

USE Sales;

CREATE TABLE Sales (

sale\_id INT PRIMARY KEY,

product\_id INT,

quantity\_sold INT,

sale\_date DATE,

total\_price DECIMAL(10, 2),

FOREIGN KEY (product\_id) REFERENCES Products(product\_id)

);

-- Insert sample data into Sales table

INSERT INTO Sales (sale\_id, product\_id, quantity\_sold, sale\_date, total\_price) VALUES

(1, 101, 5, '2024-01-01', 2500.00),

(2, 102, 3, '2024-01-02', 900.00),

(3, 103, 2, '2024-01-02', 60.00),

(4, 104, 4, '2024-01-03', 80.00),

(5, 105, 6, '2024-01-03', 90.00);

SELECT \* FROM Sales;

SELECT sale\_id, sale\_date FROM Sales;

SELECT \* FROM Sales WHERE total\_price > 100;

SELECT sale\_id, total\_price FROM Sales

WHERE sale\_date = '2024-01-03';

SELECT SUM(total\_price) AS total\_revenue FROM Sales;

SELECT SUM(quantity\_sold) AS total\_quantity\_sold FROM Sales;

SELECT sale\_id, product\_id, total\_price FROM Sales

WHERE quantity\_sold > 4;

SELECT AVG(total\_price) AS average\_total\_price FROM Sales;