QUESTION 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, &amp;#39;2024-01-01&amp;#39;, 2500.00),
(2, 102, 3, &amp;#39;2024-01-02&amp;#39;, 900.00),
(3, 103, 2, &amp;#39;2024-01-02&amp;#39;, 60.00),
(4, 104, 4, &amp;#39;2024-01-03&amp;#39;, 80.00),
(5, 105, 6, &amp;#39;2024-01-03&amp;#39;, 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:

```
USE products;
DROP TABLE IF EXISTS 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 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 FROM Sales;

SELECT sale_id, product_id, total_price FROM Sales
WHERE quantity_sold > 4;

SELECT AVG(total_price) AS average_price FROM Sales;
```

QUESTION 2:

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 'Electronics' 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:

```
DROP TABLE IF EXISTS 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);
INSERT INTO products VALUES (102, 'Smartphone', 'Electronics', 300.00);
INSERT INTO products VALUES (103, 'Headphone', 'Electronics', 30.00);
INSERT INTO products VALUES (104, 'Keyboard', 'Electronics', 20.00);
INSERT INTO products VALUES (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;
```