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 TABLE Products (
   product_id INT PRIMARY KEY,
   product_name VARCHAR(100),
   category VARCHAR(50),
   unit_price DECIMAL(10, 2)
);

INSERT INTO Products (product_id, product_name, category, unit_price) VALUES
(101, 'Laptop', 'Electronics', 500.00),
(102, 'Smartphone', 'Electronics', 300.00),
(103, 'Headphones', 'Electronics', 30.00),
(104, 'Keyboard', 'Electronics', 20.00),
(105, 'Mouse', 'Electronics', 15.00);
```

1. Retrieve all columns from the product table.

```
select * from pro_table;
```

2. Retrieve the product_name and unit_price from the Products table.

```
select pro_name,unit_price from pro_table;
```

3. Filter the Products table to show only products in the 'Electronics' category.

```
select pro_name,category from pro_table  where category = 'electonics';
```

4. Retrieve the product_id and product_name from the Products table for products with a unit_price greater than $100.

```
select  pro_id,pro_name from pro_table
where unit_price > 1000;
```

5. **Calculate the average unit_price of products in the Products table.**

```
select avg(unit_price) from pro_table;
```

6. **Retrieve product_name and unit_price from the Products table with the Highest Unit Price**

```
select pro_name,unit_price,max(unit_price) from pro_table
group by pro_name,unit_price limit 1;
```

7. **Retrieve the product_name and unit_price from the Products table, ordering the results by unit_price in descending order.**

```
select pro_name,unit_price from pro_table
order by unit_price desc;
```

8. **Retrieve the product_name and unit_price from the Products table, filtering the unit_price to show only values between $20 and $600.**

```
select pro_name,unit_price from pro_table
where unit_price between 20 and 600;
```

9. **Retrieve the product_name and category from the Products table, ordering the results by category in ascending order.**

```
select pro_name,category from pro_table
order by category;
```

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 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);
```

1. Retrieve all columns from the Sales table.

```
select * from Sales;
```

2. Retrieve the sale_id and sale_date from the Sales table.

```
select sale_id,sale_date from Sales;
```

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

```
select sale_id, product_id, quantity_sold, sale_date, total_price from Sales
where total_price > 100;
```

4. Retrieve the sale_id and total_price from the Sales table for sales made on January 3, 2024.

```
select sale_id,total_price from Sales where
sale_date='2024-01-03';
```

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

```
select sum(total_price) as total_revenue from Sales;
```

6. Calculate the total quantity_sold from the Sales table.

```
select sum(quantity_sold) as total_quantity_sold from Sales;
```

7. Retrieve the sale_id, product_id, and total_price from the Sales table for sales with a quantity_sold greater than 4.

```
select sale_id,product_id,total_price from Sales where quantity_sold > 4;
```

8. Calculate the average total_price of sales in the Sales table.

```
select avg(total_price) from Sales;
```