## **Product Sales Analysis using SQL**

To solve this problem, we need to analyze the product sales and identify the top-selling products based on their total sales. We need to write a query that retrieves the product name from the “Products” table and calculates the total revenue by multiplying the quantity by the sales for each order from the “Orders” table.

CREATE TABLE Products (

productID INT PRIMARY KEY,

productName VARCHAR(50),

price DECIMAL(10, 2)

);

INSERT INTO Products (productID, productName, price) VALUES

(1, 'Apple', 2.5),

(2, 'Banana', 1.5),

(3, 'Orange', 3.0),

(4, 'Mango', 2.0);

CREATE TABLE Orders (

orderID INT PRIMARY KEY,

productID INT,

quantity INT,

sales DECIMAL(10, 2)

);

INSERT INTO Orders (orderID, productID, quantity, sales) VALUES

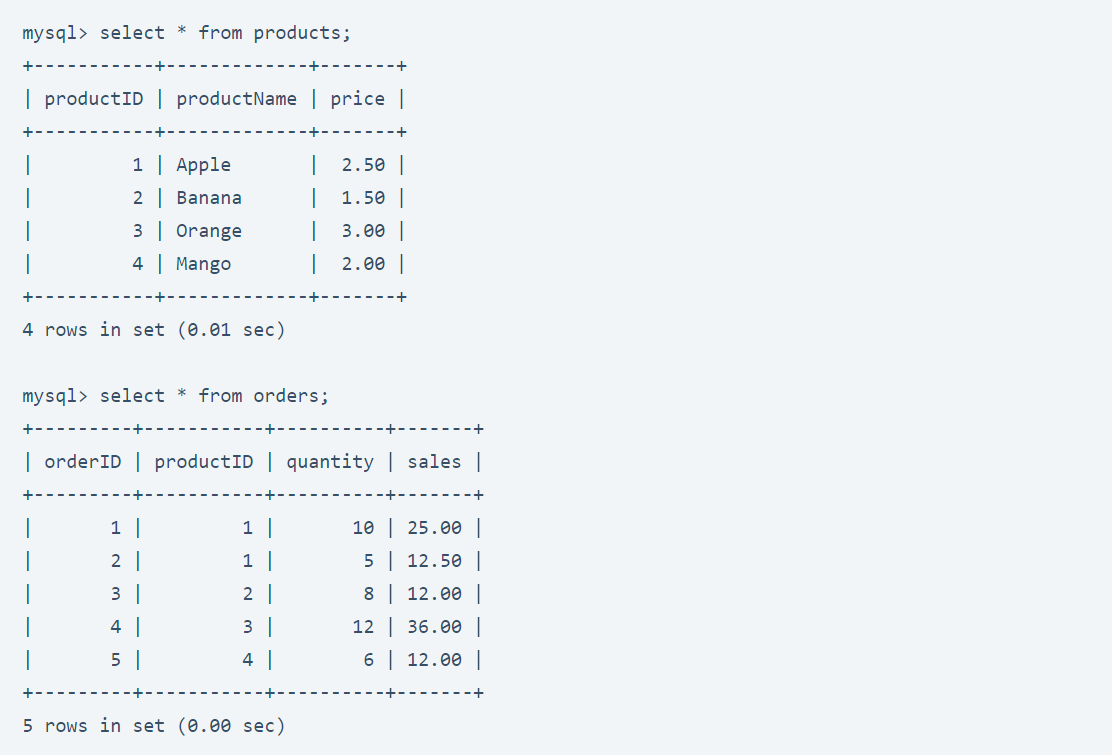
(1, 1, 10, 25.0),

(2, 1, 5, 12.5),

(3, 2, 8, 12.0),

(4, 3, 12, 36.0),

(5, 4, 6, 12.0);



Solution the Product Sales Analysis problem using SQL:

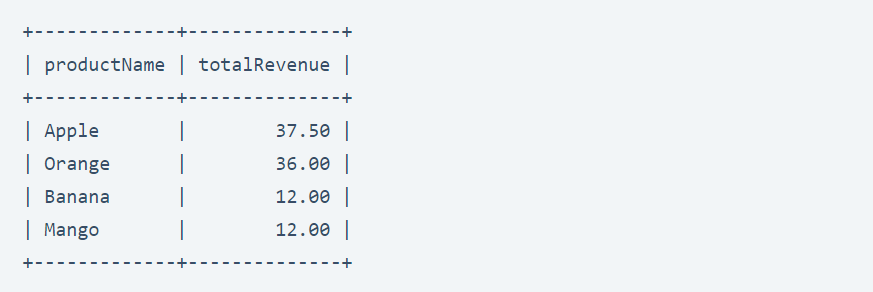
SELECT p.productName, SUM(o.quantity \* p.price) AS totalRevenue

FROM Products p

JOIN Orders o ON p.productID = o.productID

GROUP BY p.productName

ORDER BY totalRevenue DESC;



The given SQL query aims to analyze product sales and identify top-selling products based on their total revenue. Here’s how the query works:

* The SELECT statement specifies the columns to be included in the output: productName from the “Products” table and the calculated totalRevenue for each product.
* The FROM clause specifies the source tables for the query. We join the “Products” table p with the “Orders” table o using the common productID column. This allows us to retrieve relevant information from both tables.
* The JOIN condition p.productID = o.productID ensures that we only consider matching records between the “Products” and “Orders” tables, linking the corresponding products and orders based on their IDs.
* The GROUP BY clause groups the results based on the productName column. This aggregation ensures that the subsequent calculations and aggregations are performed on a per-product basis.
* The expression SUM(o.quantity \* p.price) calculates the totalRevenue for each product. It multiplies the quantity of each product sold (o.quantity) with the price per unit (p.price) from the “Products” table for each order in the “Orders” table. The SUM function then calculates the sum of these values for each product.
* The ORDER BY clause sorts the result set in descending order based on the totalRevenue. By using DESC, the products with the highest total revenue will appear at the top of the list.