

Backend SQL Task

Creation of table structure: -

Customer Table: -

```
CREATE TABLE Customers (  
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    Email VARCHAR(100),  
    DateOfBirth DATE  
);  
  
INSERT INTO Customers (FirstName, LastName, Email, DateOfBirth) VALUES  
(  
'John', 'Doe', 'john.doe@example.com', '1985-01-15'),  
(  
'Jane', 'Smith', 'jane.smith@example.com', '1990-06-20')
```

Products Table: -

```
CREATE TABLE Products (  
    ProductID INT AUTO_INCREMENT PRIMARY KEY,  
    ProductName VARCHAR(100),  
    Price DECIMAL(10, 2)  
);  
  
INSERT INTO Products (ProductName, Price) VALUES  
(  
'Laptop', 1000),  
(  
'Smartphone', 600),  
(  
'Headphones', 100);
```

Orders Table:-

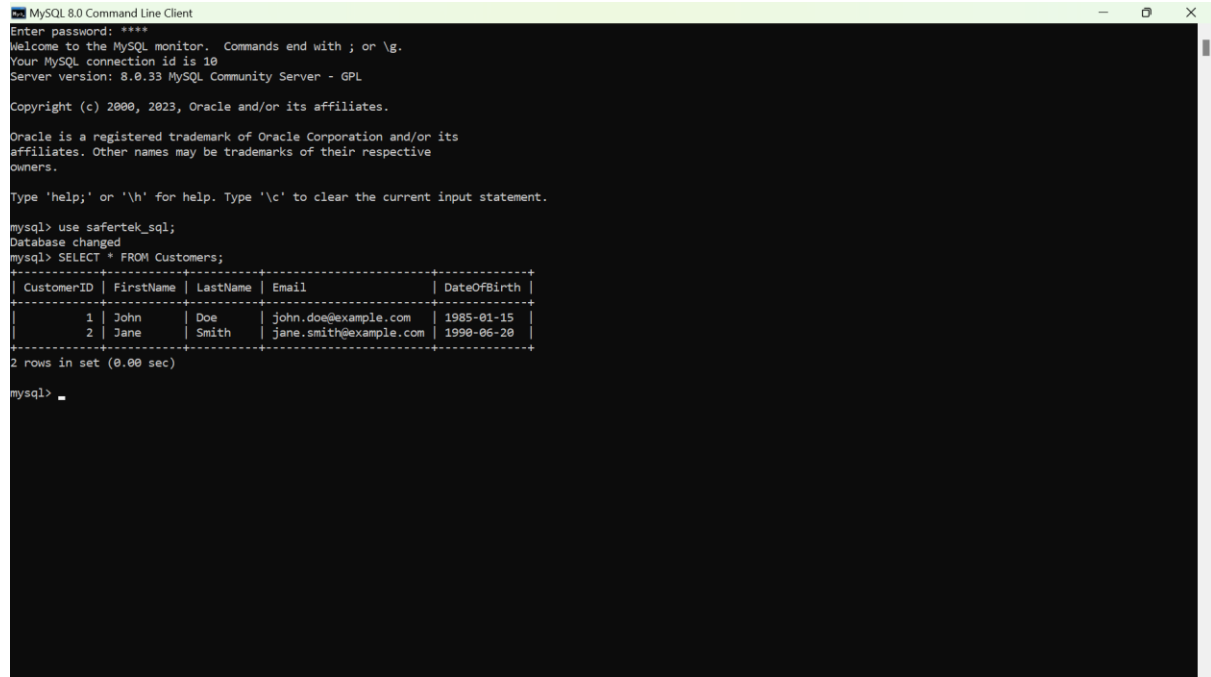
```
CREATE TABLE Orders (  
    OrderID INT AUTO_INCREMENT PRIMARY KEY,  
    CustomerID INT,  
    OrderDate DATE,  
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  
);  
  
INSERT INTO Orders (CustomerID, OrderDate) VALUES  
(1, '2023-01-10'),  
(2, '2023-01-12');
```

OrderItems Table:-

```
CREATE TABLE OrderItems (  
    OrderItemID INT AUTO_INCREMENT PRIMARY KEY,  
    OrderID INT,  
    ProductID INT,  
    Quantity INT,  
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),  
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
);  
  
INSERT INTO OrderItems (OrderID, ProductID, Quantity) VALUES  
(1, 1, 1),  
(1, 3, 2),  
(2, 2, 1),  
(2, 3, 1);
```

1. List all customers.

```
SELECT * FROM Customers;
```



```
MySQL 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

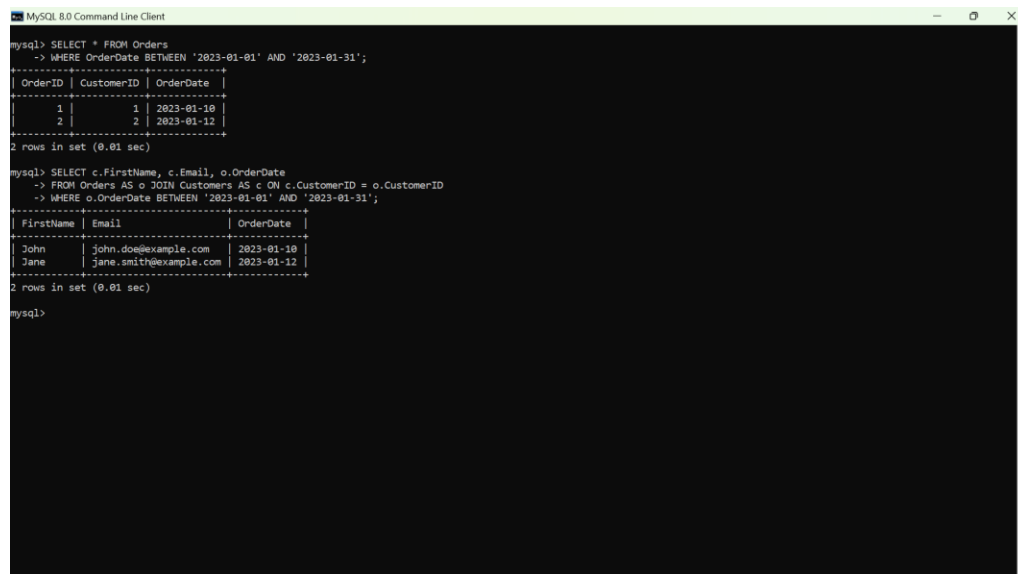
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use safertek_sql;
Database changed
mysql> SELECT * FROM Customers;
+-----+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | Email | DateOfBirth |
+-----+-----+-----+-----+-----+
| 1 | John | Doe | john.doe@example.com | 1985-01-15 |
| 2 | Jane | Smith | jane.smith@example.com | 1990-06-20 |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

2. Find all orders placed in January 2023.

```
SELECT * FROM Orders WHERE OrderDate BETWEEN '2023-01-01' AND '2023-01-31';
```



```
MySQL 8.0 Command Line Client

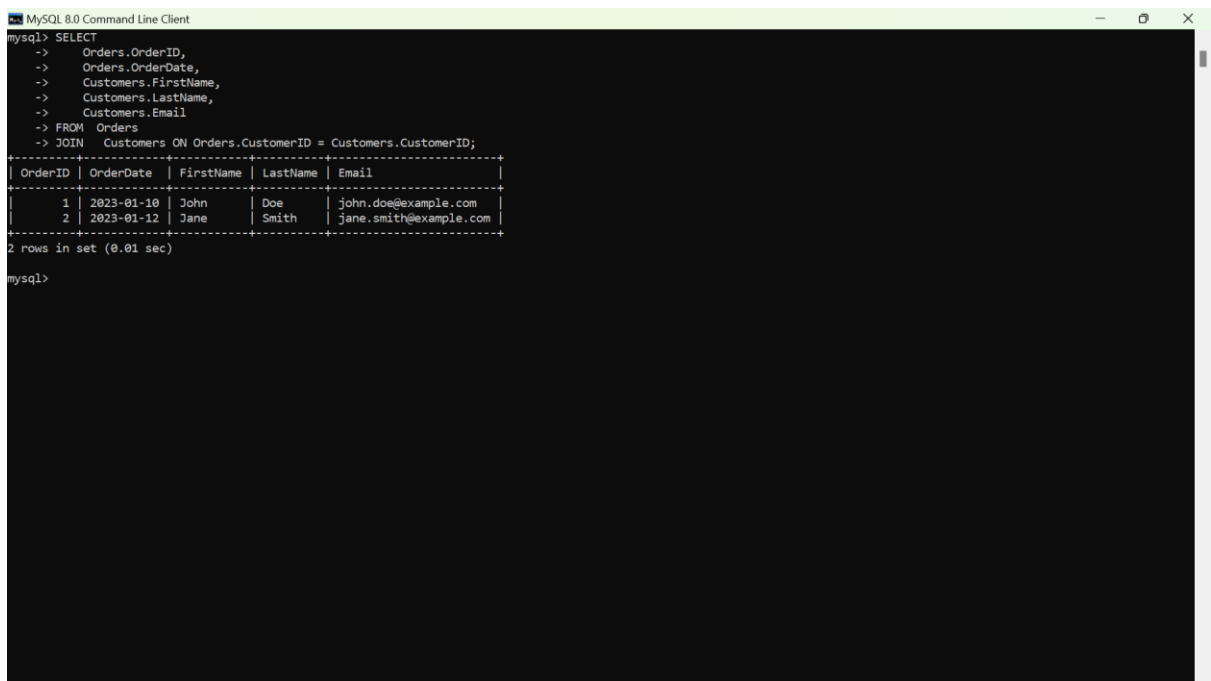
mysql> SELECT * FROM Orders
-> WHERE OrderDate BETWEEN '2023-01-01' AND '2023-01-31';
+-----+-----+-----+
| OrderID | CustomerID | OrderDate |
+-----+-----+-----+
| 1 | 1 | 2023-01-10 |
| 2 | 2 | 2023-01-12 |
+-----+-----+-----+
2 rows in set (0.01 sec)

mysql> SELECT c.FirstName, c.Email, o.OrderDate
-> FROM Orders AS o JOIN Customers AS c ON c.CustomerID = o.CustomerID
-> WHERE o.OrderDate BETWEEN '2023-01-01' AND '2023-01-31';
+-----+-----+-----+
| FirstName | Email | OrderDate |
+-----+-----+-----+
| John | john.doe@example.com | 2023-01-10 |
| Jane | jane.smith@example.com | 2023-01-12 |
+-----+-----+-----+
2 rows in set (0.01 sec)

mysql>
```

3. Get the details of each order including the customer name and email.

```
SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName,  
Customers.Email  
  
FROM Orders  
  
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```



The screenshot shows a MySQL 8.0 Command Line Client window. The user has entered a SQL query to select order details along with customer information. The query is as follows:

```
mysql> SELECT  
-> Orders.OrderID,  
-> Orders.OrderDate,  
-> Customers.FirstName,  
-> Customers.LastName,  
-> Customers.Email  
-> FROM Orders  
-> JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

The result set displays two rows of data:

OrderID	OrderDate	FirstName	LastName	Email
1	2023-01-10	John	Doe	john.doe@example.com
2	2023-01-12	Jane	Smith	jane.smith@example.com

Below the table, it indicates "2 rows in set (0.01 sec)". The prompt "mysql>" is visible at the bottom of the window.

4. List the products purchased in a specific order (e.g., OrderID = 1).

```
SELECT Products.ProductName, OrderItems.Quantity  
  
FROM OrderItems  
  
JOIN Products ON OrderItems.ProductID = Products.ProductID  
  
WHERE OrderItems.OrderID = 1;
```

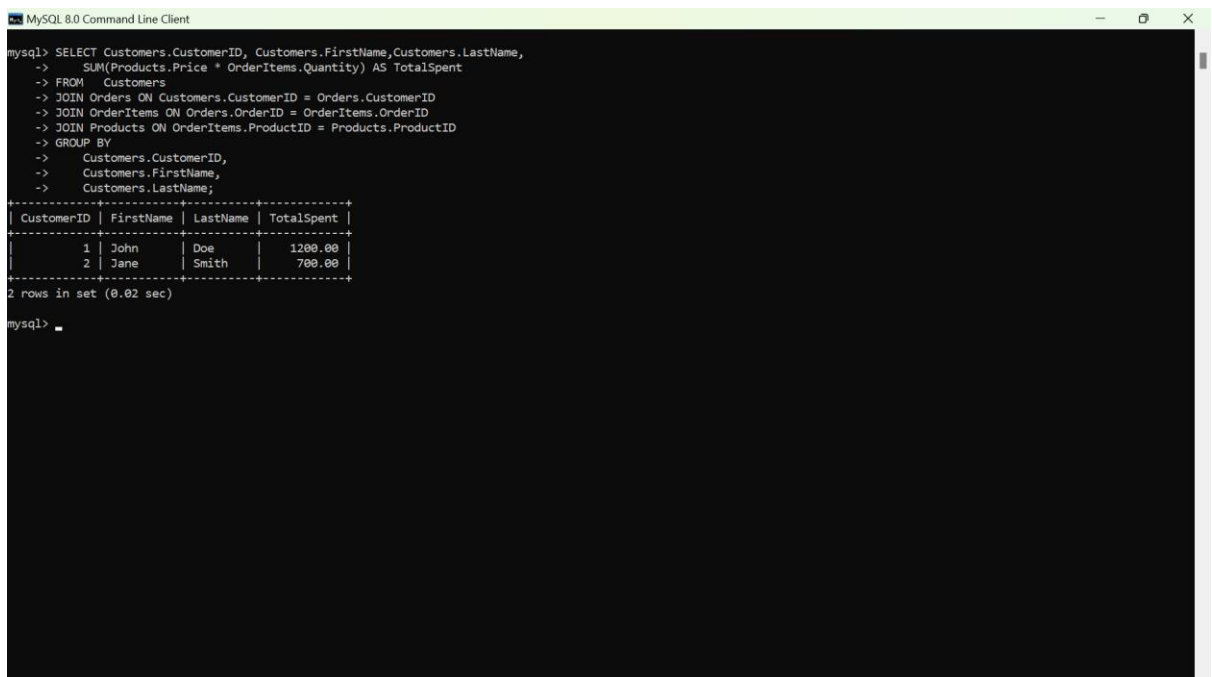
```
mysql> SELECT Products.ProductName, OrderItems.Quantity FROM OrderItems
-> JOIN
-> Products ON OrderItems.ProductID = Products.ProductID
-> WHERE
-> OrderItems.OrderID = 1;
```

ProductName	Quantity
Laptop	1
Headphones	2

2 rows in set (0.00 sec)

5. Calculate the total amount spent by each customer.

```
SELECT
Customers.CustomerID, Customers.FirstName, Customers.LastName,
SUM(Products.Price * OrderItems.Quantity) AS TotalSpent
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID
JOIN Products ON OrderItems.ProductID = Products.ProductID
GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName;
```



```
MySQL 8.0 Command Line Client
mysql> SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName,
-> SUM(Products.Price * OrderItems.Quantity) AS TotalSpent
-> FROM Customers
-> JOIN Orders ON Customers.CustomerID = Orders.CustomerID
-> JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID
-> JOIN Products ON OrderItems.ProductID = Products.ProductID
-> GROUP BY
-> Customers.CustomerID,
-> Customers.FirstName,
-> Customers.LastName;
```

CustomerID	FirstName	LastName	TotalSpent
1	John	Doe	1200.00
2	Jane	Smith	700.00

2 rows in set (0.02 sec)

```
mysql>
```

6. Find the most popular product (the one that has been ordered the most).

```
SELECT Products.ProductID, Products.ProductName,
SUM(OrderItems.Quantity) AS TotalQuantitySold
FROM Products JOIN OrderItems ON Products.ProductID = OrderItems.ProductID
GROUP BY Products.ProductID, Products.ProductName
ORDER BY TotalQuantitySold DESC LIMIT 1;
```

```
mysql> SELECT
->     Products.ProductID,
->     Products.ProductName,
->     SUM(OrderItems.Quantity) AS TotalQuantitySold
-> FROM
->     Products
-> JOIN
->     OrderItems ON Products.ProductID = OrderItems.ProductID
-> GROUP BY
->     Products.ProductID,
->     Products.ProductName
-> ORDER BY TotalQuantitySold DESC
-> LIMIT 1;
```

ProductID	ProductName	TotalQuantitySold
3	Headphones	3

1 row in set (0.00 sec)

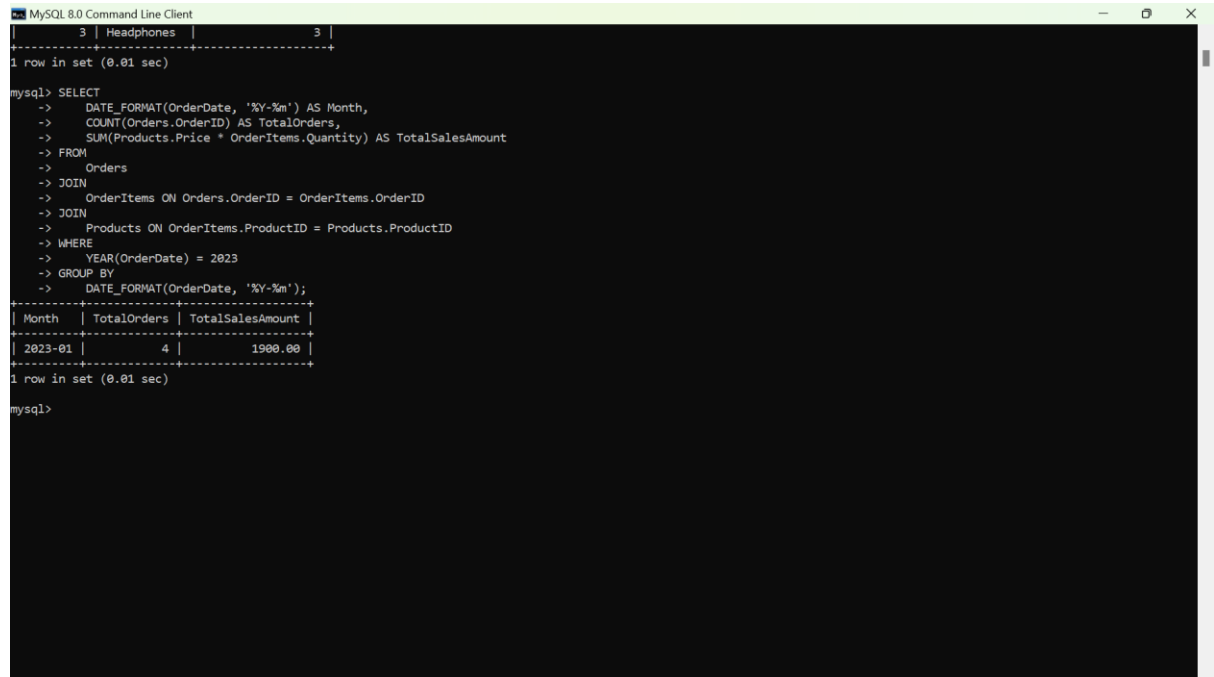
7. Get the total number of orders and the total sales amount for each month in 2023.

```
SELECT DATE_FORMAT(OrderDate, '%Y-%m') AS Month, COUNT(Orders.OrderID)
AS TotalOrders,
SUM(Products.Price * OrderItems.Quantity) AS TotalSalesAmount
FROM Orders
JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID
```

JOIN Products ON OrderItems.ProductID = Products.ProductID

WHERE YEAR(OrderDate) = 2023

GROUP BY DATE_FORMAT(OrderDate, '%Y-%m');



```

mysql> SELECT
  -> DATE_FORMAT(OrderDate, '%Y-%m') AS Month,
  -> COUNT(Orders.OrderID) AS TotalOrders,
  -> SUM(Products.Price * OrderItems.Quantity) AS TotalSalesAmount
  -> FROM
  -> Orders
  -> JOIN
  -> OrderItems ON Orders.OrderID = OrderItems.OrderID
  -> JOIN
  -> Products ON OrderItems.ProductID = Products.ProductID
  -> WHERE
  -> YEAR(OrderDate) = 2023
  -> GROUP BY
  -> DATE_FORMAT(OrderDate, '%Y-%m');
+-----+-----+-----+
| Month | TotalOrders | TotalSalesAmount |
+-----+-----+-----+
| 2023-01 | 4 | 1900.00 |
+-----+-----+-----+
1 row in set (0.01 sec)

mysql>

```

8. Find customers who have spent more than \$1000.

SELECT Customers.CustomerID, Customers.FirstName, Customers.LastName,
SUM(Products.Price * OrderItems.Quantity) AS TotalSpent

FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID

JOIN OrderItems ON Orders.OrderID = OrderItems.OrderID

JOIN Products ON OrderItems.ProductID = Products.ProductID

GROUP BY Customers.CustomerID, Customers.FirstName, Customers.LastName

HAVING TotalSpent > 1000;

```
mysql> SELECT
->     Customers.CustomerID,
->     Customers.FirstName,
->     Customers.LastName,
->     SUM(Products.Price * OrderItems.Quantity) AS TotalSpent
-> FROM
->     Customers
-> JOIN
->     Orders ON Customers.CustomerID = Orders.CustomerID
-> JOIN
->     OrderItems ON Orders.OrderID = OrderItems.OrderID
-> JOIN
->     Products ON OrderItems.ProductID = Products.ProductID
-> GROUP BY
->     Customers.CustomerID,
->     Customers.FirstName,
->     Customers.LastName
-> HAVING
->     TotalSpent > 1000;
+-----+-----+-----+-----+
| CustomerID | FirstName | LastName | TotalSpent |
+-----+-----+-----+-----+
|          1 | John     | Doe      |    1200.00 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```