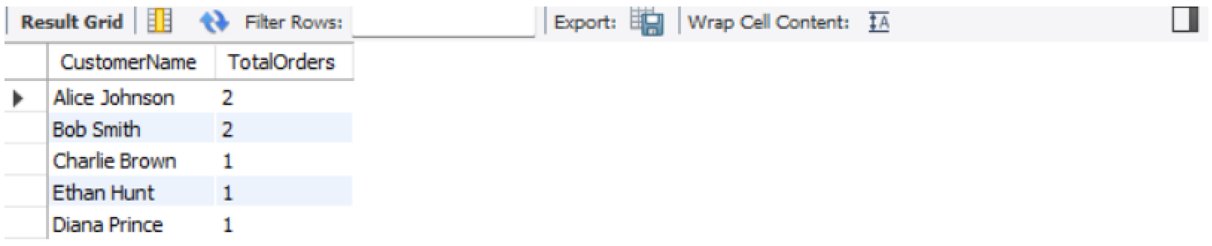
Task3:SQL QUERIES

Name: Tetali Durga Venkata Reddy

Email: tetalidurgavenkatareddy@gmail.com

TABLE CREATION AND DATA INSERTION:

```
SELECT Customers.CustomerName, COUNT(Orders.OrderID) AS TotalOrders
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
WHERE Orders.OrderDate >= '2024-01-01'
GROUP BY Customers.CustomerName
ORDER BY TotalOrders DESC;
```



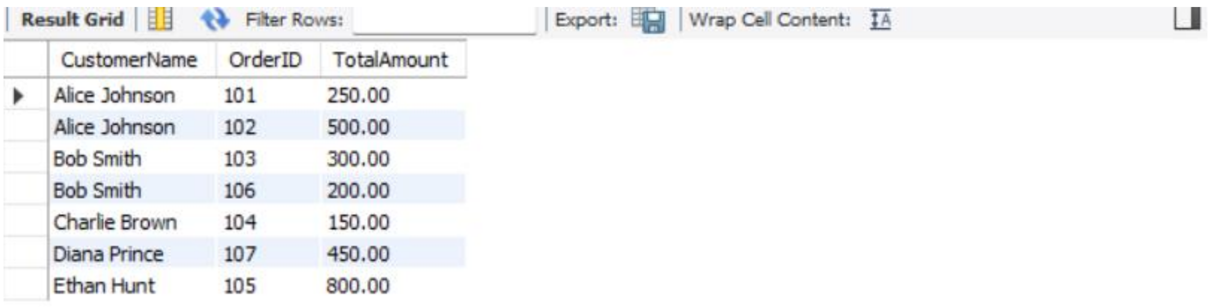
CustomerName	TotalOrders
Alice Johnson	2
Bob Smith	2
Charlie Brown	1
Ethan Hunt	1
Diana Prince	1

INNER JOIN:

SELECT, WHERE, GROUP BY, ORDER BY:

```
SELECT
    Customers.CustomerName,
    Orders.OrderID,
    Orders.TotalAmount
FROM
    Customers
INNER JOIN
```

Orders ON Customers.CustomerID = Orders.CustomerID;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerName	OrderID	TotalAmount
Alice Johnson	101	250.00
Alice Johnson	102	500.00
Bob Smith	103	300.00
Bob Smith	106	200.00
Charlie Brown	104	150.00
Diana Prince	107	450.00
Ethan Hunt	105	800.00

LEFT JOIN:

-- Show all customers, even if they have no orders

SELECT

Customers.CustomerName,

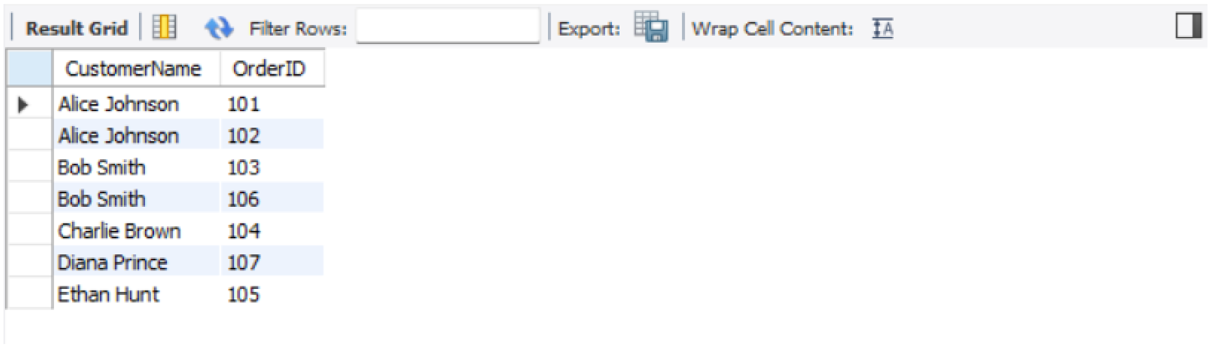
Orders.OrderID

FROM

Customers

LEFT JOIN

Orders ON Customers.CustomerID = Orders.CustomerID;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CustomerName	OrderID
Alice Johnson	101
Alice Johnson	102
Bob Smith	103
Bob Smith	106
Charlie Brown	104
Diana Prince	107
Ethan Hunt	105

RIGHT JOIN:

-- Show all orders, even if there is no matching customer

SELECT

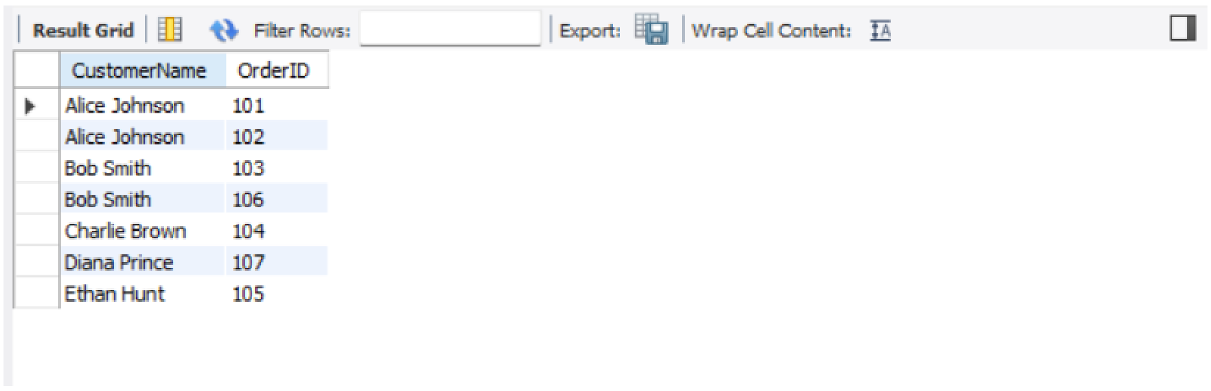
Customers.CustomerName,

Orders.OrderID

FROM

Customers

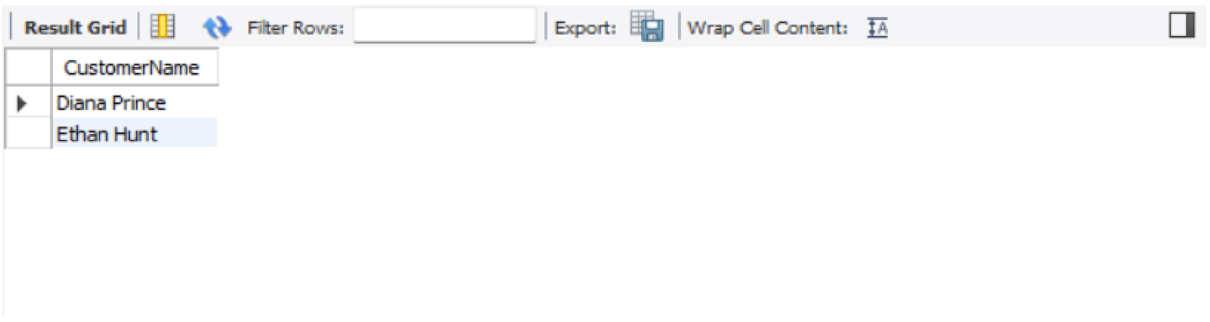
```
RIGHT JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID;
```



CustomerName	OrderID
Alice Johnson	101
Alice Johnson	102
Bob Smith	103
Bob Smith	106
Charlie Brown	104
Diana Prince	107
Ethan Hunt	105

SUBQUERY:

```
-- Customers whose average order amount is higher than the overall average
SELECT
    CustomerName
FROM
    Customers
WHERE
    CustomerID IN (
        SELECT
            CustomerID
        FROM
            Orders
        GROUP BY
            CustomerID
        HAVING
            AVG(TotalAmount) > (SELECT AVG(TotalAmount) FROM Orders)
    );
```

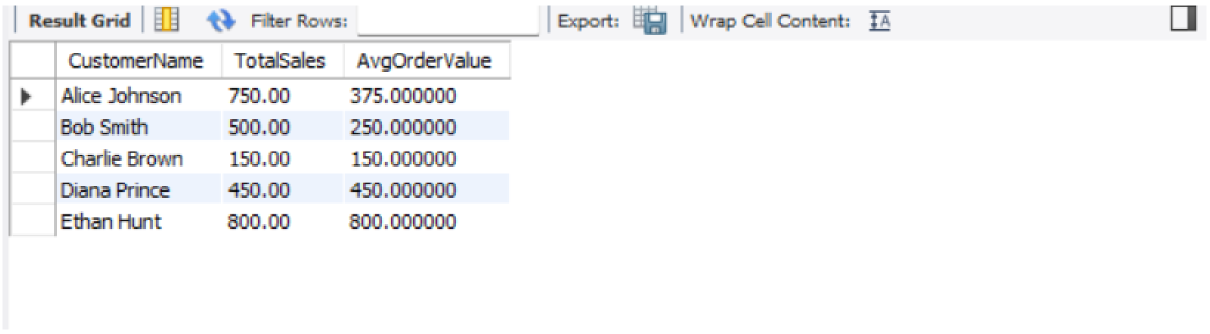


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

```
CustomerName
Diana Prince
Ethan Hunt
```

AGGREGATE FUNCTIONS (SUM, AVG):

```
-- Show total and average order amounts by each customer
SELECT
    Customers.CustomerName,
    SUM(Orders.TotalAmount) AS TotalSales,
    AVG(Orders.TotalAmount) AS AvgOrderValue
FROM
    Customers
INNER JOIN
    Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY
    Customers.CustomerName;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

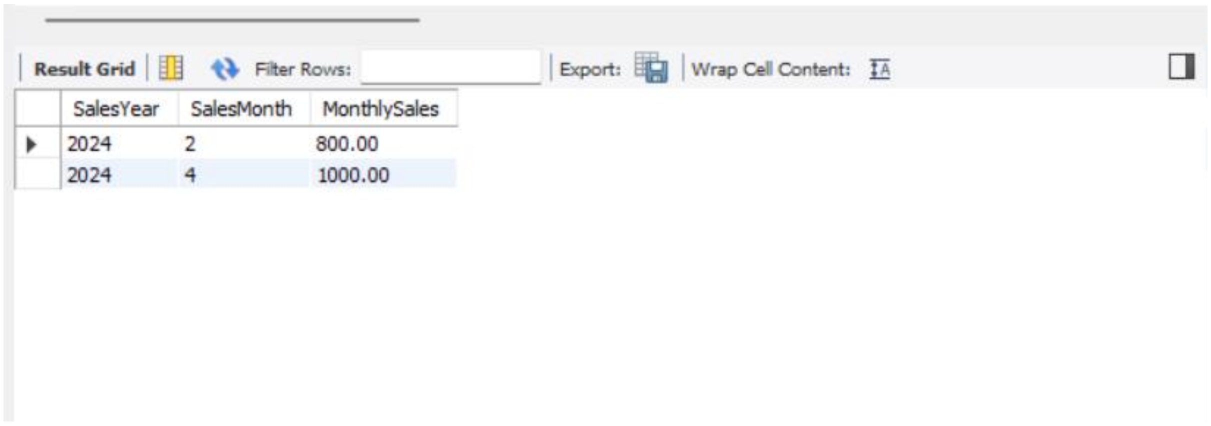
```
CustomerName   TotalSales  AvgOrderValue
Alice Johnson  750.00      375.000000
Bob Smith      500.00      250.000000
Charlie Brown  150.00      150.000000
Diana Prince   450.00      450.000000
Ethan Hunt     800.00      800.000000
```

CREATE VIEW for Analysis:

```
-- Create a view to see monthly sales
CREATE VIEW MonthlySalesView AS
SELECT
```

```
    YEAR(OrderDate) AS SalesYear,
    MONTH(OrderDate) AS SalesMonth,
    SUM(TotalAmount) AS MonthlySales
FROM
    Orders
GROUP BY
    YEAR(OrderDate), MONTH(OrderDate);

-- Query the View
SELECT * FROM MonthlySalesView WHERE MonthlySales > 500;
```



SalesYear	SalesMonth	MonthlySales
2024	2	800.00
2024	4	1000.00

CREATE INDEX for Optimization:

```
-- Create an index on CustomerID to speed up JOINs
CREATE INDEX idx_orders_customerid ON Orders(CustomerID);

-- Create another index on OrderDate + TotalAmount (optional for analysis queries)
CREATE INDEX idx_orders_orderdate_totalamount ON Orders(OrderDate, TotalAmount);

-- View all indexes created
SHOW INDEX FROM Orders;
```

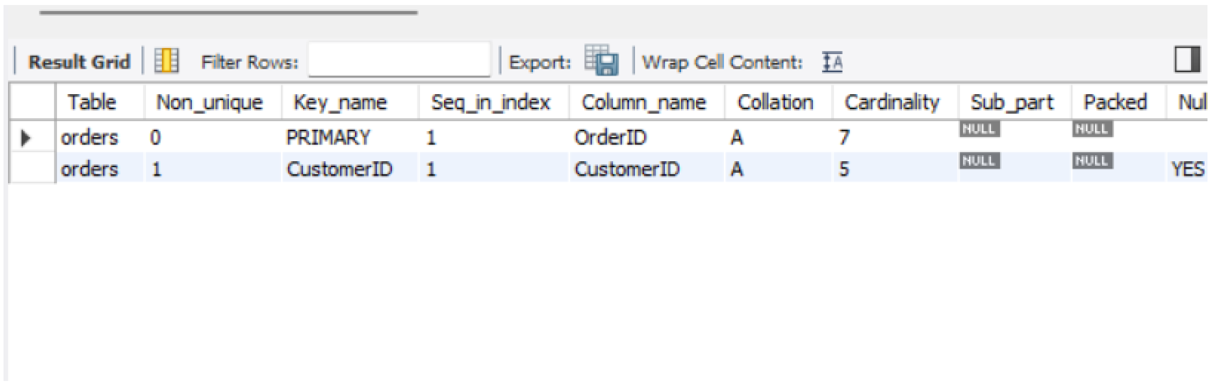


	Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Nul
▶	orders	0	PRIMARY	1	OrderID	A	7	NULL	NULL	
	orders	1	CustomerID	1	CustomerID	A	5	NULL	NULL	YES