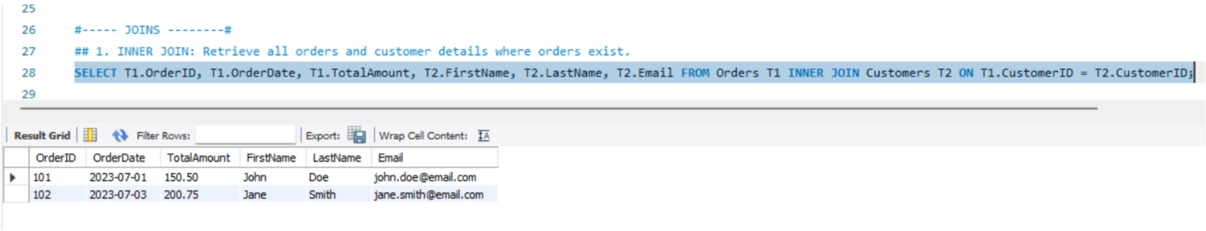
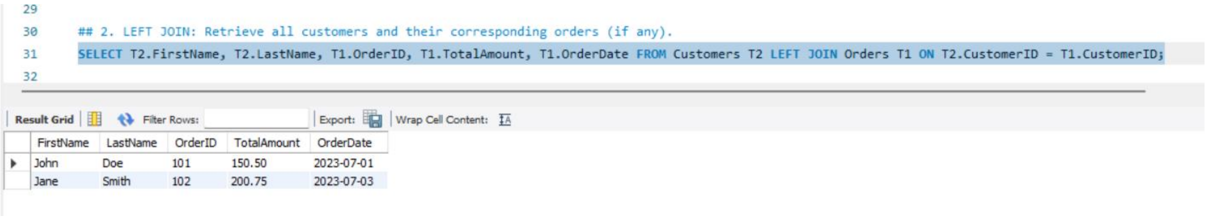
OUTPUT



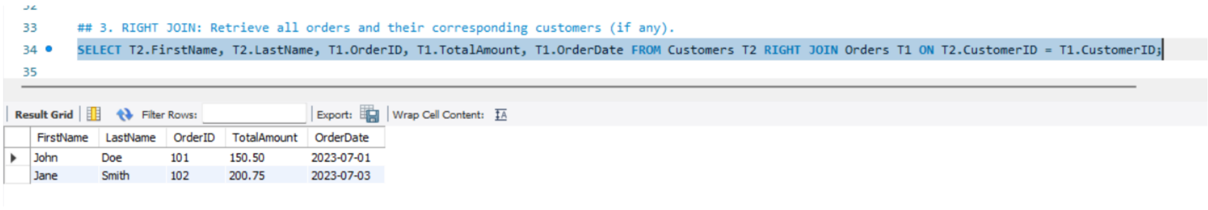
```
#----- JOINS --------#
## 1. INNER JOIN: Retrieve all orders and customer details where orders exist.
SELECT T1.OrderID, T1.OrderDate, T1.TotalAmount, T2.FirstName, T2.LastName, T2.Email FROM Orders T1 INNER JOIN Customers T2 ON T1.CustomerID = T2.CustomerID;
```

OrderID	OrderDate	TotalAmount	FirstName	LastName	Email
101	2023-07-01	150.50	John	Doe	john.doe@email.com
102	2023-07-03	200.75	Jane	Smith	jane.smith@email.com



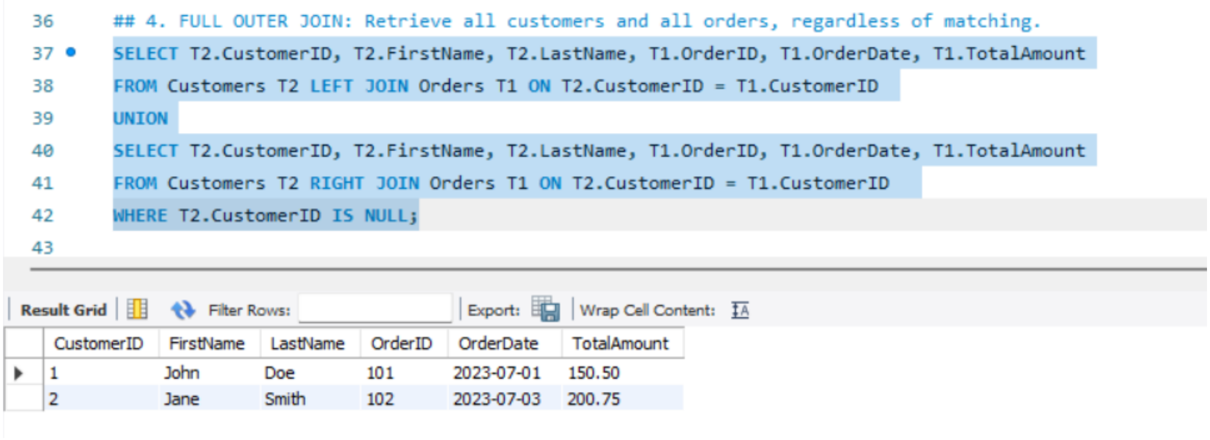
```
## 2. LEFT JOIN: Retrieve all customers and their corresponding orders (if any).
SELECT T2.FirstName, T2.LastName, T1.OrderID, T1.TotalAmount, T1.OrderDate FROM Customers T2 LEFT JOIN Orders T1 ON T2.CustomerID = T1.CustomerID;
```

FirstName	LastName	OrderID	TotalAmount	OrderDate
John	Doe	101	150.50	2023-07-01
Jane	Smith	102	200.75	2023-07-03



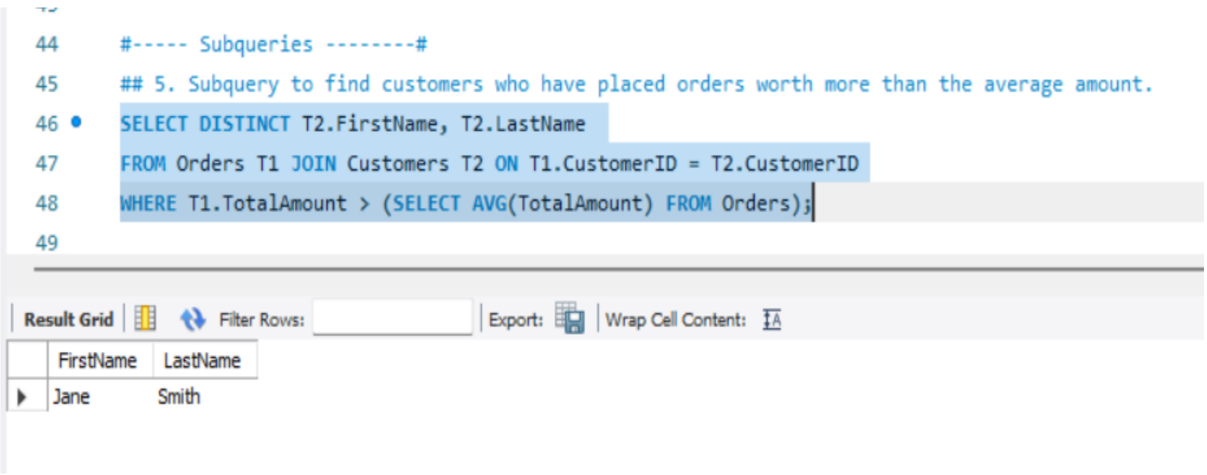
```
## 3. RIGHT JOIN: Retrieve all orders and their corresponding customers (if any).
SELECT T2.FirstName, T2.LastName, T1.OrderID, T1.TotalAmount, T1.OrderDate FROM Customers T2 RIGHT JOIN Orders T1 ON T2.CustomerID = T1.CustomerID;
```

FirstName	LastName	OrderID	TotalAmount	OrderDate
John	Doe	101	150.50	2023-07-01
Jane	Smith	102	200.75	2023-07-03



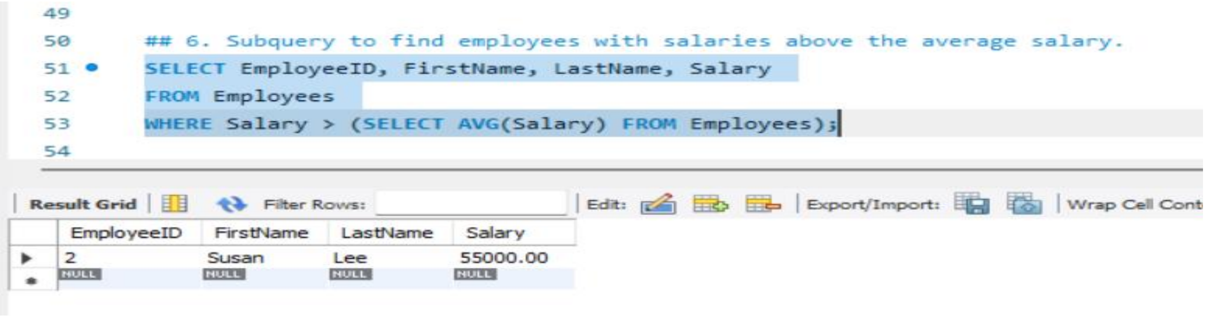
```
## 4. FULL OUTER JOIN: Retrieve all customers and all orders, regardless of matching.
SELECT T2.CustomerID, T2.FirstName, T2.LastName, T1.OrderID, T1.OrderDate, T1.TotalAmount
FROM Customers T2 LEFT JOIN Orders T1 ON T2.CustomerID = T1.CustomerID
UNION
SELECT T2.CustomerID, T2.FirstName, T2.LastName, T1.OrderID, T1.OrderDate, T1.TotalAmount
FROM Customers T2 RIGHT JOIN Orders T1 ON T2.CustomerID = T1.CustomerID
WHERE T2.CustomerID IS NULL;
```

CustomerID	FirstName	LastName	OrderID	OrderDate	TotalAmount
1	John	Doe	101	2023-07-01	150.50
2	Jane	Smith	102	2023-07-03	200.75



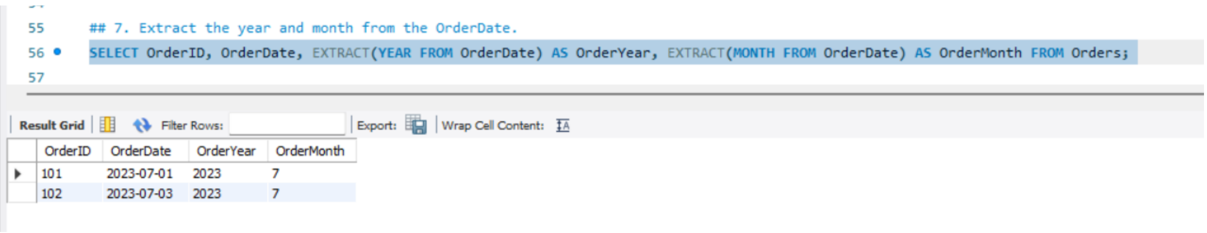
```
#----- Subqueries --------#
## 5. Subquery to find customers who have placed orders worth more than the average amount.
SELECT DISTINCT T2.FirstName, T2.LastName
FROM Orders T1 JOIN Customers T2 ON T1.CustomerID = T2.CustomerID
WHERE T1.TotalAmount > (SELECT AVG(TotalAmount) FROM Orders);
```

FirstName	LastName
Jane	Smith



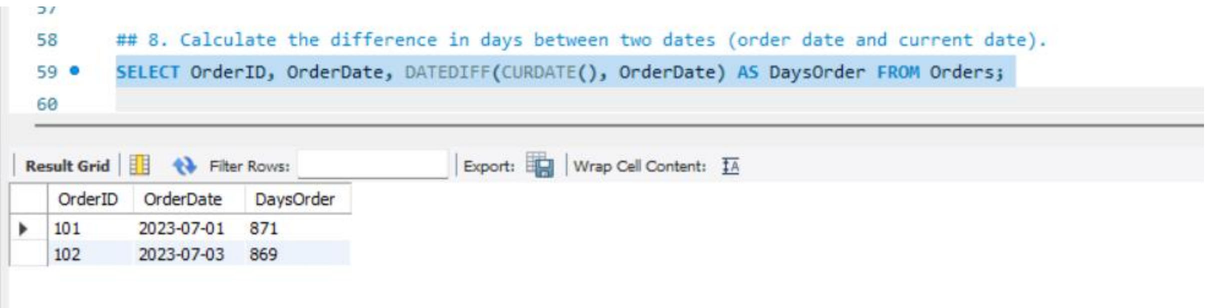
```
## 6. Subquery to find employees with salaries above the average salary.
SELECT EmployeeID, FirstName, LastName, Salary
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

EmployeeID	FirstName	LastName	Salary
2	Susan	Lee	55000.00
NULL	NULL	NULL	NULL



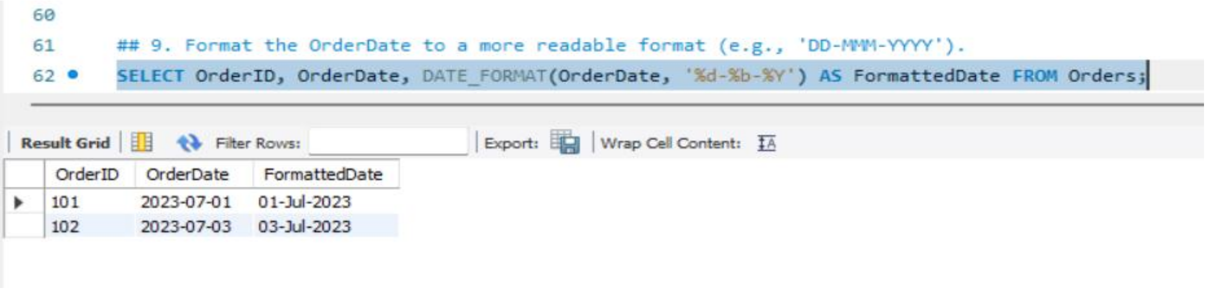
```
## 7. Extract the year and month from the OrderDate.
SELECT OrderID, OrderDate, EXTRACT(YEAR FROM OrderDate) AS OrderYear, EXTRACT(MONTH FROM OrderDate) AS OrderMonth FROM Orders;
```

OrderID	OrderDate	OrderYear	OrderMonth
101	2023-07-01	2023	7
102	2023-07-03	2023	7



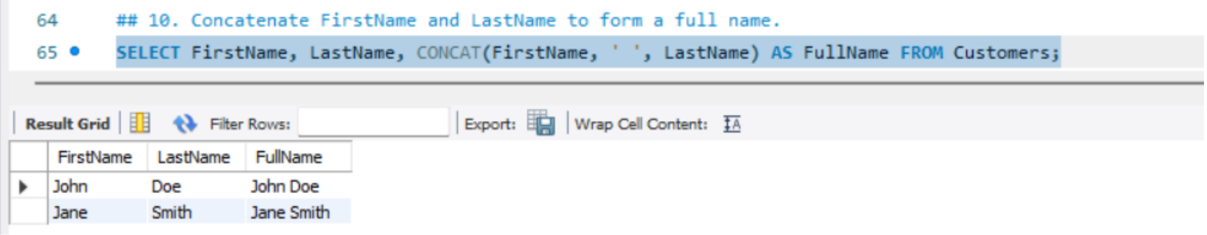
```
## 8. Calculate the difference in days between two dates (order date and current date).
SELECT OrderID, OrderDate, DATEDIFF(CURDATE(), OrderDate) AS DaysOrder FROM Orders;
```

OrderID	OrderDate	DaysOrder
101	2023-07-01	871
102	2023-07-03	869



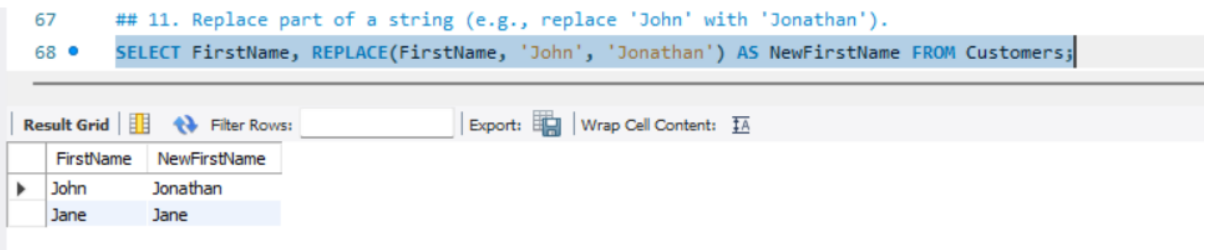
```
## 9. Format the OrderDate to a more readable format (e.g., 'DD-MMM-YYYY').
SELECT OrderID, OrderDate, DATE_FORMAT(OrderDate, '%d-%b-%Y') AS FormattedDate FROM Orders;
```

OrderID	OrderDate	FormattedDate
101	2023-07-01	01-Jul-2023
102	2023-07-03	03-Jul-2023



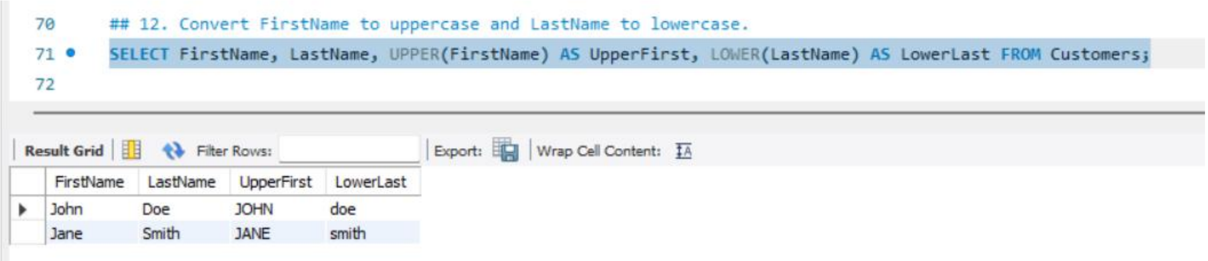
```
## 10. Concatenate FirstName and LastName to form a full name.
SELECT FirstName, LastName, CONCAT(FirstName, ' ', LastName) AS FullName FROM Customers;
```

FirstName	LastName	FullName
John	Doe	John Doe
Jane	Smith	Jane Smith



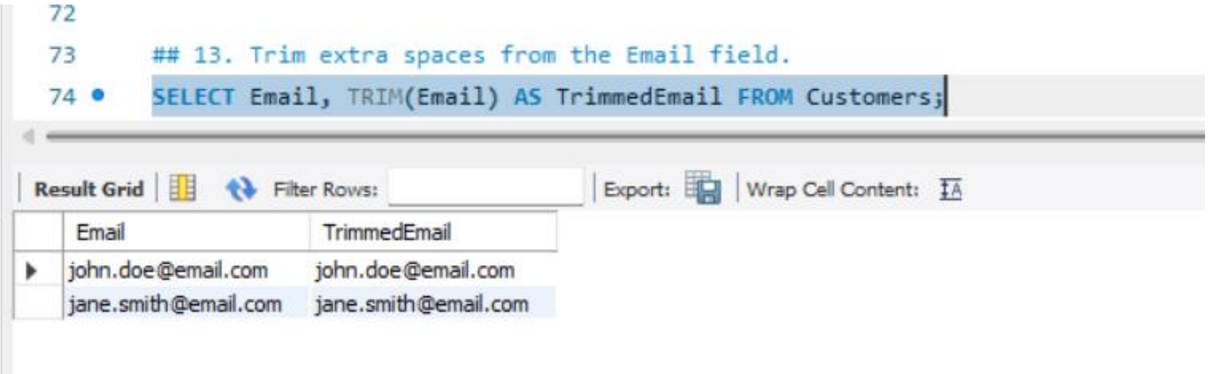
```
## 11. Replace part of a string (e.g., replace 'John' with 'Jonathan').
SELECT FirstName, REPLACE(FirstName, 'John', 'Jonathan') AS NewFirstName FROM Customers;
```

FirstName	NewFirstName
John	Jonathan
Jane	Jane



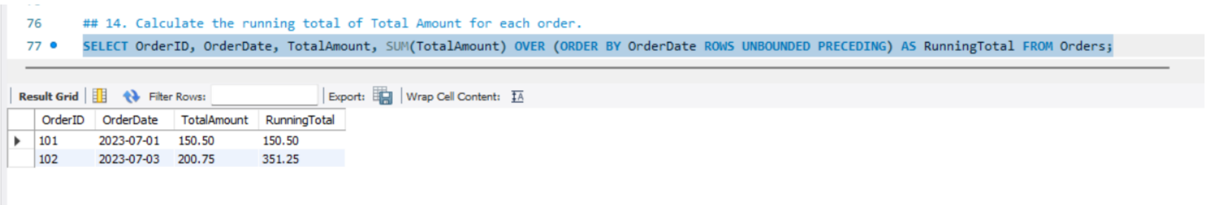
```
## 12. Convert FirstName to uppercase and LastName to lowercase.
SELECT FirstName, LastName, UPPER(FirstName) AS UpperFirst, LOWER(LastName) AS LowerLast FROM Customers;
```

FirstName	LastName	UpperFirst	LowerLast
John	Doe	JOHN	doe
Jane	Smith	JANE	smith



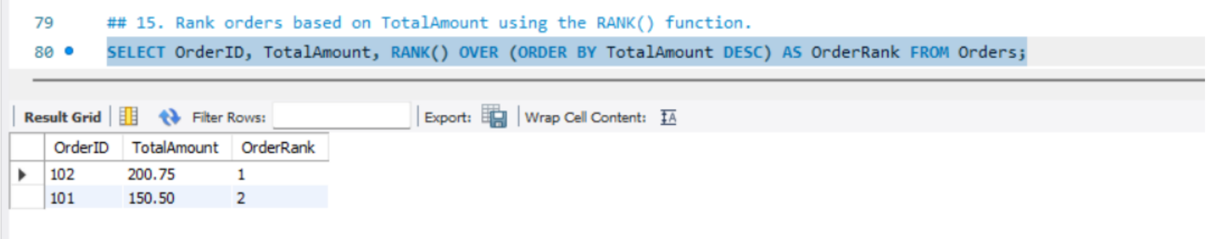
```
## 13. Trim extra spaces from the Email field.
SELECT Email, TRIM(Email) AS TrimmedEmail FROM Customers;
```

Email	TrimmedEmail
john.doe@email.com	john.doe@email.com
jane.smith@email.com	jane.smith@email.com



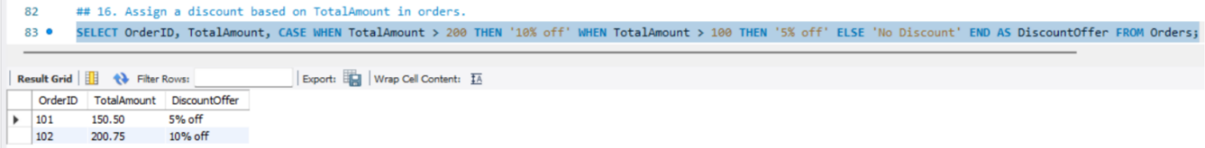
```
## 14. Calculate the running total of Total Amount for each order.
SELECT OrderID, OrderDate, TotalAmount, SUM(TotalAmount) OVER (ORDER BY OrderDate ROWS UNBOUNDED PRECEDING) AS RunningTotal FROM Orders;
```

OrderID	OrderDate	TotalAmount	RunningTotal
101	2023-07-01	150.50	150.50
102	2023-07-03	200.75	351.25



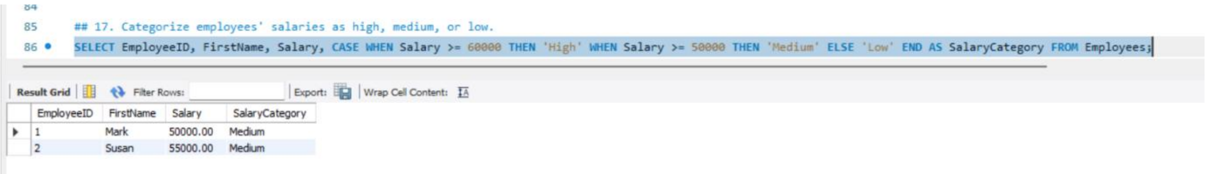
```
## 15. Rank orders based on TotalAmount using the RANK() function.
SELECT OrderID, TotalAmount, RANK() OVER (ORDER BY TotalAmount DESC) AS OrderRank FROM Orders;
```

OrderID	TotalAmount	OrderRank
102	200.75	1
101	150.50	2



```
## 16. Assign a discount based on TotalAmount in orders.
SELECT OrderID, TotalAmount, CASE WHEN TotalAmount > 200 THEN '10% off' WHEN TotalAmount > 100 THEN '5% off' ELSE 'No Discount' END AS DiscountOffer FROM Orders;
```

OrderID	TotalAmount	DiscountOffer
101	150.50	5% off
102	200.75	10% off



```
## 17. Categorize employees' salaries as high, medium, or low.
SELECT EmployeeID, FirstName, Salary, CASE WHEN Salary >= 60000 THEN 'High' WHEN Salary >= 50000 THEN 'Medium' ELSE 'Low' END AS SalaryCategory FROM Employees;
```

EmployeeID	FirstName	Salary	SalaryCategory
1	Mark	50000.00	Medium
2	Susan	55000.00	Medium