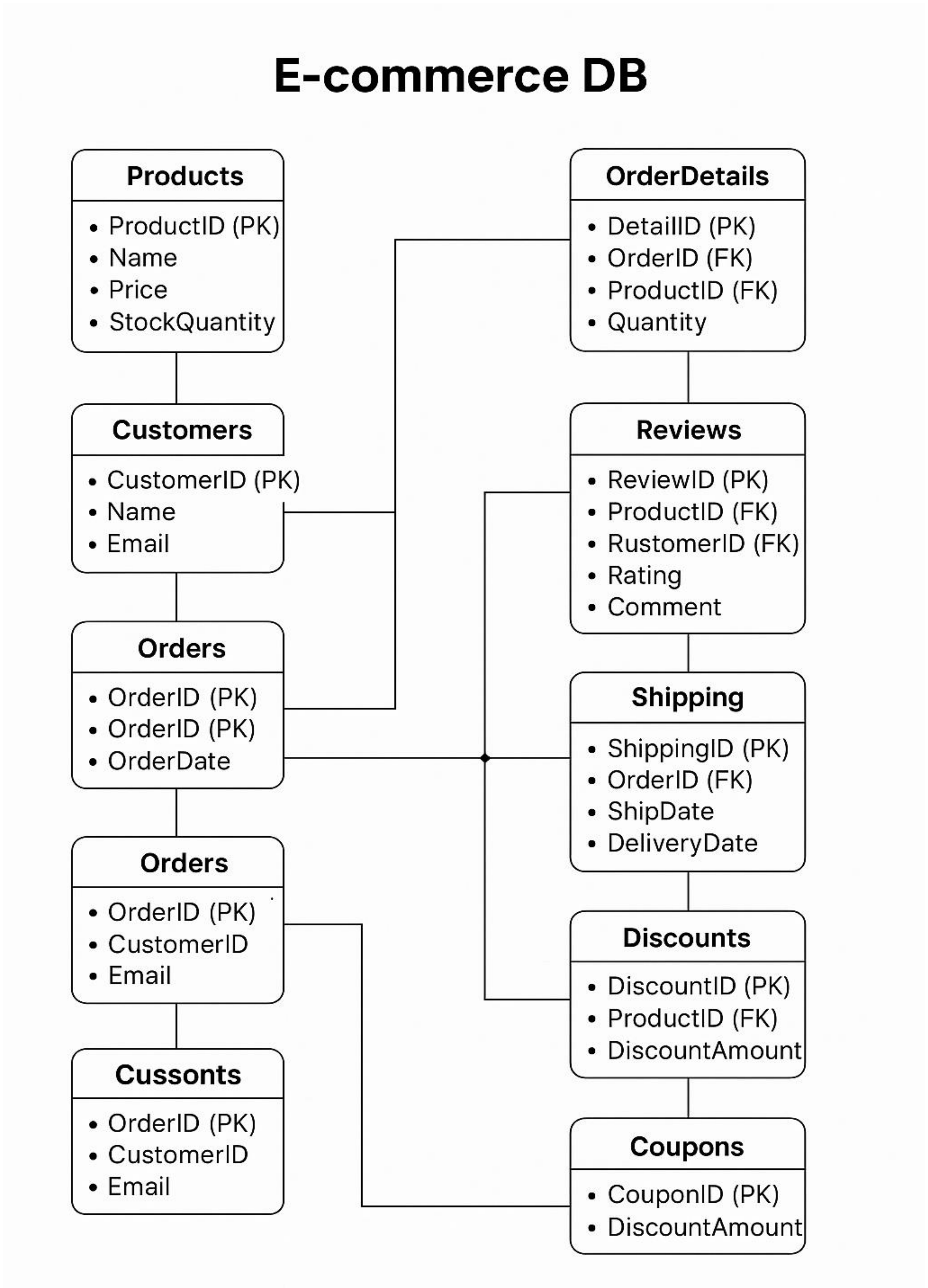
RIDDHI AGRAWAL

MYSQL

ECOMMERCE MANAGEMENT SYSTEM



Database Name: Riddhi

```
CREATE TABLE Categories (
   CategoryID INT auto_increment PRIMARY KEY,
   CategoryName VARCHAR(100) NOT NULL
);
```

```
CREATE TABLE Products (
   ProductID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(100) NOT NULL,
   Price DECIMAL(10, 2) NOT NULL,
   StockQuantity INT NOT NULL,
   CategoryID INT,
   FOREIGN KEY (CategoryID) REFERENCES
Categories(CategoryID)
);
```

```
CREATE TABLE Customers (
   CustomerID INT PRIMARY KEY
AUTO_INCREMENT,
   Name VARCHAR(255) NOT NULL,
   Email VARCHAR(255) UNIQUE NOT NULL
);
```

```
CREATE TABLE Shipping (
   ShippingID INT PRIMARY KEY AUTO_INCREMENT,
   OrderID INT,
   ShipDate DATE,
   DeliveryDate DATE,
   FOREIGN KEY (OrderID) REFERENCES
Orders(OrderID)
);
```

```
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY AUTO_INCREMENT,
   CustomerID INT,
   OrderDate DATE NOT NULL,
   FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID)
);
```

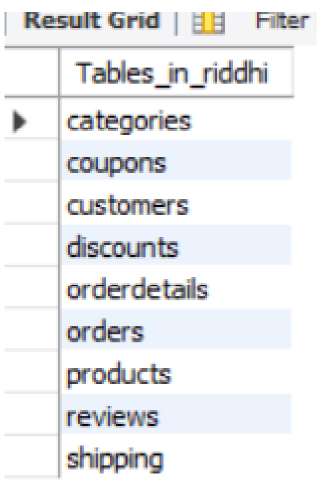
```
CREATE TABLE Discounts (
   DiscountID INT PRIMARY KEY
AUTO_INCREMENT,
   ProductID INT,
   DiscountAmount DECIMAL(10, 2),
   FOREIGN KEY (ProductID) REFERENCES
Products(ProductID)
);
```

```
CREATE TABLE OrderDetails (
   DetailID INT PRIMARY KEY AUTO_INCREMENT,
   OrderID INT,
   ProductID INT,
   Quantity INT NOT NULL,
   FOREIGN KEY (OrderID) REFERENCES
Orders(OrderID),
   FOREIGN KEY (ProductID) REFERENCES
Products(ProductID)
);
```

```
CREATE TABLE Reviews (
   ReviewID INT PRIMARY KEY AUTO_INCREMENT,
   ProductID INT,
   CustomerID INT,
   Rating INT CHECK (Rating >= 1 AND Rating <= 5),
   Comment TEXT,
   FOREIGN KEY (ProductID) REFERENCES
Products(ProductID),
   FOREIGN KEY (CustomerID) REFERENCES
Customers(CustomerID)
);
```

```
CREATE TABLE Coupons (
   CouponID INT PRIMARY KEY AUTO_INCREMENT,
   DiscountAmount DECIMAL(10, 2)
);
```

Requirement:



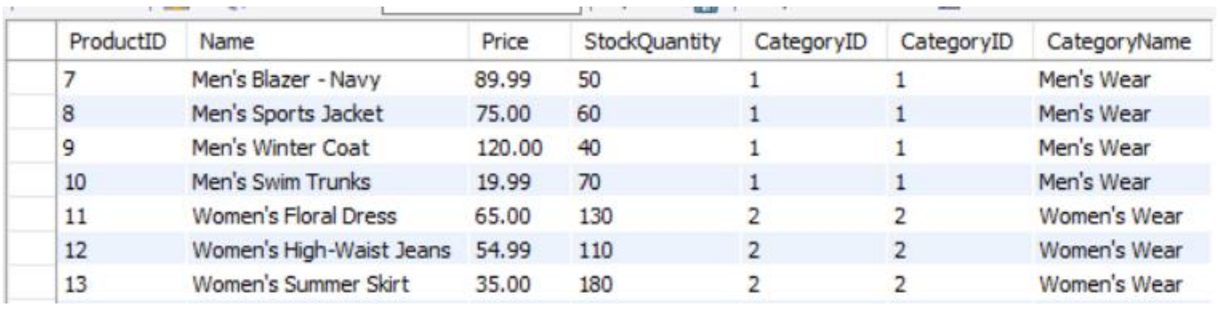
	Tables_in_riddhi
▶	categories
	coupons
	customers
	discounts
	orderdetails
	orders
	products
	reviews
	shipping

1. Perform analytics and show at-least 50 queries
2. with those 50 queries (80% of the queries should have joins, subqueries)
3. Insert atleast 70-80 records minimum and maximum of 100 records
4. insert, update ,delete will not be considered as a part of analytics.

- **JOINS (INNER, LEFT, RIGHT, FULL)**

1. **List all products along with their category names.**

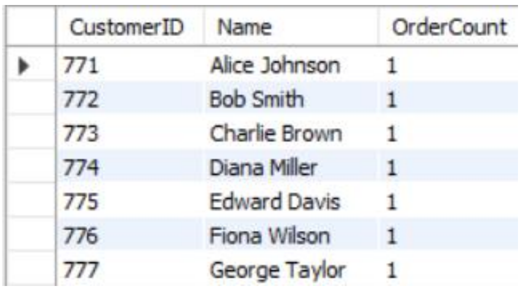
```
SELECT *
FROM Products
LEFT JOIN Categories
ON Products.CategoryID = Categories.CategoryID;
```



	ProductID	Name	Price	StockQuantity	CategoryID	CategoryID	CategoryName
	7	Men's Blazer - Navy	89.99	50	1	1	Men's Wear
	8	Men's Sports Jacket	75.00	60	1	1	Men's Wear
	9	Men's Winter Coat	120.00	40	1	1	Men's Wear
	10	Men's Swim Trunks	19.99	70	1	1	Men's Wear
	11	Women's Floral Dress	65.00	130	2	2	Women's Wear
	12	Women's High-Waist Jeans	54.99	110	2	2	Women's Wear
	13	Women's Summer Skirt	35.00	180	2	2	Women's Wear

2. **Show each customer's order count**

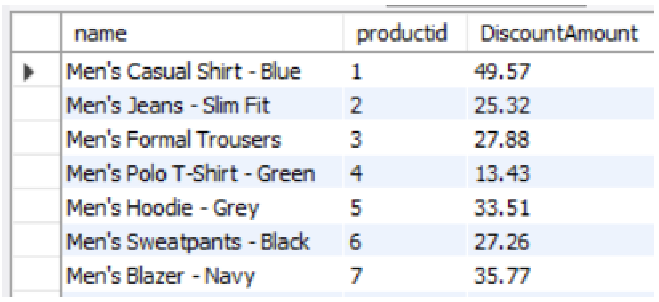
```
SELECT Customers.CustomerID, Customers.Name,
COUNT(Orders.OrderID) AS OrderCount
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.Name;
```



	CustomerID	Name	OrderCount
▶	771	Alice Johnson	1
	772	Bob Smith	1
	773	Charlie Brown	1
	774	Diana Miller	1
	775	Edward Davis	1
	776	Fiona Wilson	1
	777	George Taylor	1

3. **List product names and their discount amounts.**

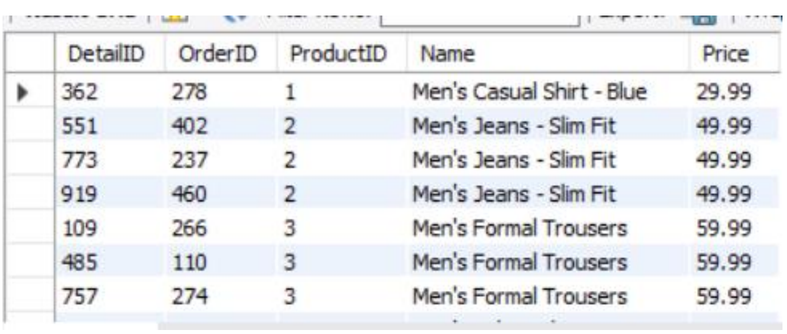
```
select products.name, products.productid,
discounts.DiscountAmount
from products right join discounts
on products.productid= discounts.productid;
```



	name	productid	DiscountAmount
▶	Men's Casual Shirt - Blue	1	49.57
	Men's Jeans - Slim Fit	2	25.32
	Men's Formal Trousers	3	27.88
	Men's Polo T-Shirt - Green	4	13.43
	Men's Hoodie - Grey	5	33.51
	Men's Sweatpants - Black	6	27.26
	Men's Blazer - Navy	7	35.77

4. **Show order details with product names and prices.**

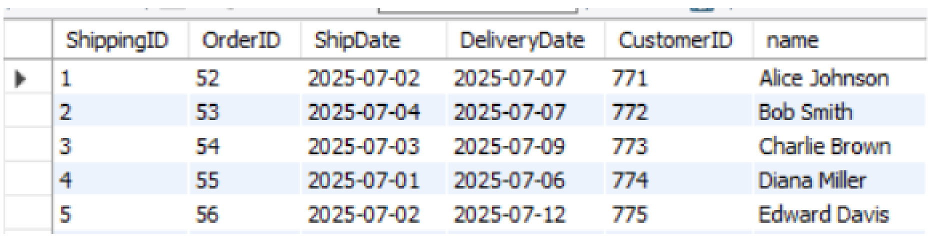
```
SELECT  OrderDetails.DetailID, OrderDetails.OrderID,  OrderDetails.ProductID,
Products.Name, Products.Price
FROM OrderDetails
JOIN Products
ON OrderDetails.ProductID = Products.ProductID;
```



DetailID	OrderID	ProductID	Name	Price
362	278	1	Men's Casual Shirt - Blue	29.99
551	402	2	Men's Jeans - Slim Fit	49.99
773	237	2	Men's Jeans - Slim Fit	49.99
919	460	2	Men's Jeans - Slim Fit	49.99
109	266	3	Men's Formal Trousers	59.99
485	110	3	Men's Formal Trousers	59.99
757	274	3	Men's Formal Trousers	59.99

5. **Find shipping info with order and customer names.**

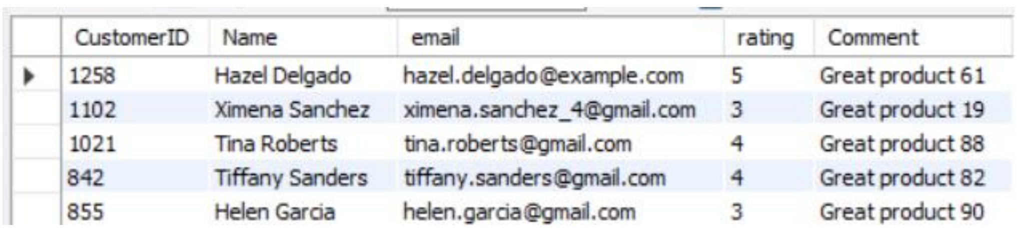
```
SELECT Shipping.ShippingID, Shipping.OrderID, Shipping.ShipDate, Shipping.DeliveryDate,
Customers.CustomerID,Customers.Name AS CustomerName
FROM Shipping
JOIN Orders ON Shipping.OrderID = Orders.OrderID
JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```



ShippingID	OrderID	ShipDate	DeliveryDate	CustomerID	name
1	52	2025-07-02	2025-07-07	771	Alice Johnson
2	53	2025-07-04	2025-07-07	772	Bob Smith
3	54	2025-07-03	2025-07-09	773	Charlie Brown
4	55	2025-07-01	2025-07-06	774	Diana Miller
5	56	2025-07-02	2025-07-12	775	Edward Davis

6. **Display all customers and any reviews they've written.**

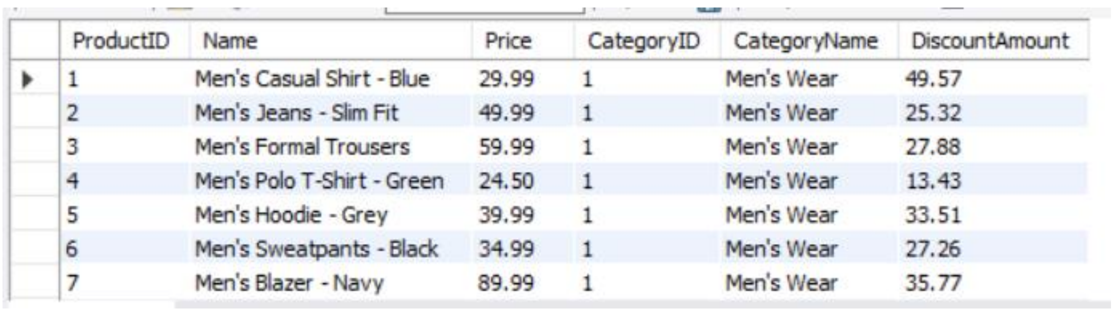
```
select customers.CustomerID, customers.Name, customers.email,
reviews.rating, reviews.Comment
from customers join reviews
on customers.customerID = reviews.customerID;
```



CustomerID	Name	email	rating	Comment
1258	Hazel Delgado	hazel.delgado@example.com	5	Great product 61
1102	Ximena Sanchez	ximena.sanchez_4@gmail.com	3	Great product 19
1021	Tina Roberts	tina.roberts@gmail.com	4	Great product 88
842	Tiffany Sanders	tiffany.sanders@gmail.com	4	Great product 82
855	Helen Garcia	helen.garcia@gmail.com	3	Great product 90

7. **List all products with their category and discount (if any).**

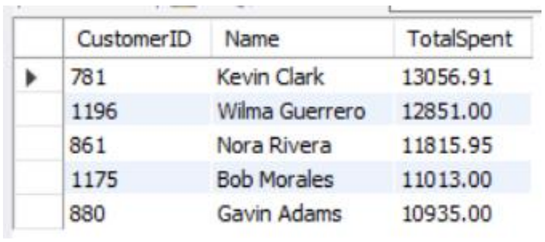
```
Select products.ProductID, products.Name, products.Price, Products.CategoryID
,categories.CategoryName,discounts.DiscountAmount
from products
join categories on products.categoryID = categories.categoryID
join discounts on Products.productID = discounts.ProductID;
```



ProductID	Name	Price	CategoryID	CategoryName	DiscountAmount
1	Men's Casual Shirt - Blue	29.99	1	Men's Wear	49.57
2	Men's Jeans - Slim Fit	49.99	1	Men's Wear	25.32
3	Men's Formal Trousers	59.99	1	Men's Wear	27.88
4	Men's Polo T-Shirt - Green	24.50	1	Men's Wear	13.43
5	Men's Hoodie - Grey	39.99	1	Men's Wear	33.51
6	Men's Sweatpants - Black	34.99	1	Men's Wear	27.26
7	Men's Blazer - Navy	89.99	1	Men's Wear	35.77

8. Find the top 5 customers who spent the most in total.

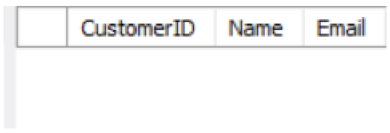
```
select customers.customerID, customers.Name,
SUM(Products.Price * OrderDetails.Quantity) AS TotalSpent
from customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Customers.CustomerID
ORDER BY TotalSpent DESC
LIMIT 5;
```



CustomerID	Name	TotalSpent
781	Kevin Clark	13056.91
1196	Wilma Guerrero	12851.00
861	Nora Rivera	11815.95
1175	Bob Morales	11013.00
880	Gavin Adams	10935.00

9. Display customers who haven't placed any orders.

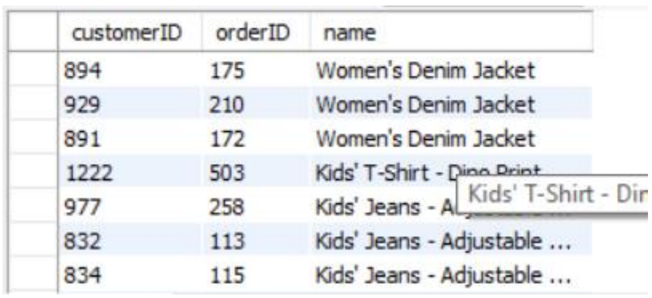
```
Select customers.customerID, customers.Name, customers.Email
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
WHERE Orders.OrderID IS NULL;
```



CustomerID	Name	Email

10. Show each customer's name, their order ID, and the product names they ordered.

```
select customers.customerID, orders.orderID, products.name from customers
join orders on customers.customerID = orders.customerID
JOIN orderDetails on orders.OrderID = orderDetails.orderID
JOIN products on orderDetails.ProductID= products.productID;
```

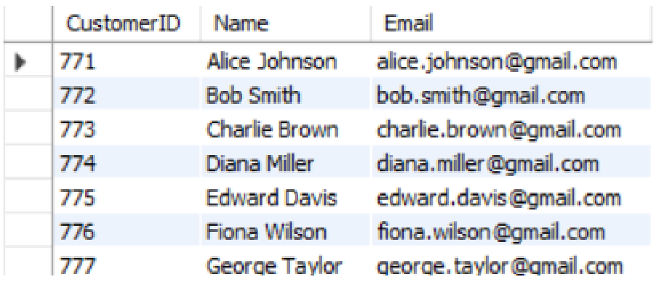


customerID	orderID	name
894	175	Women's Denim Jacket
929	210	Women's Denim Jacket
891	172	Women's Denim Jacket
1222	503	Kids' T-Shirt - Dino Print
977	258	Kids' Jeans - A
832	113	Kids' Jeans - Adjustable ...
834	115	Kids' Jeans - Adjustable ...

- **SUBQUERIES**

11. Show customers who have never written a review.

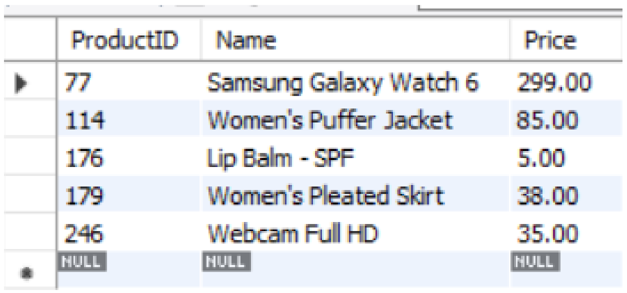
```
SELECT CustomerID, Name, Email
FROM Customers
WHERE CustomerID NOT IN (
SELECT  CustomerID
FROM Review
);
```



CustomerID	Name	Email
771	Alice Johnson	alice.johnson@gmail.com
772	Bob Smith	bob.smith@gmail.com
773	Charlie Brown	charlie.brown@gmail.com
774	Diana Miller	diana.miller@gmail.com
775	Edward Davis	edward.davis@gmail.com
776	Fiona Wilson	fiona.wilson@gmail.com
777	George Taylor	george.taylor@gmail.com

12. **List products that have never been ordered**.

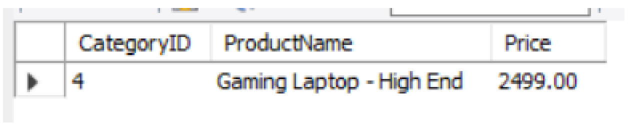
```
SELECT ProductID, Name, Price
FROM products
WHERE productID NOT  IN (
SELECT  ProductID
FROM orderdetails
);
```



ProductID	Name	Price
77	Samsung Galaxy Watch 6	299.00
114	Women's Puffer Jacket	85.00
176	Lip Balm - SPF	5.00
179	Women's Pleated Skirt	38.00
246	Webcam Full HD	35.00
NULL	NULL	NULL

13. **Find the most expensive product in all.**

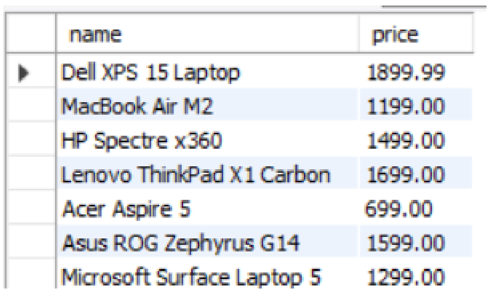
```
SELECT CategoryID, Name AS ProductName, Price
FROM Products
WHERE Price = (
SELECT MAX(Price)
FROM Products
);
```



CategoryID	ProductName	Price
4	Gaming Laptop - High End	2499.00

14. List all products with price greater than the average price of all products.

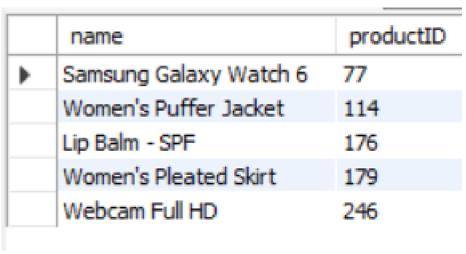
```
Select name,price from products where price >= (
select AVG(price)
from products
);
```



name	price
Dell XPS 15 Laptop	1899.99
MacBook Air M2	1199.00
HP Spectre x360	1499.00
Lenovo ThinkPad X1 Carbon	1699.00
Acer Aspire 5	699.00
Asus ROG Zephyrus G14	1599.00
Microsoft Surface Laptop 5	1299.00

15.Show products that have never been ordered.

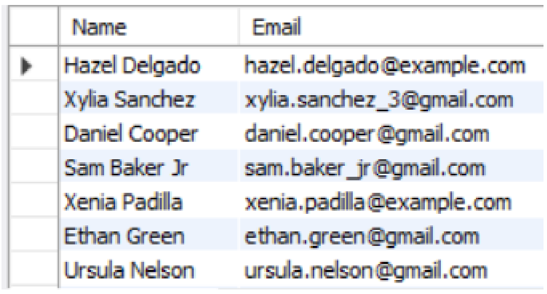
```
select name, productID from products
where ProductID NOT in (
select productID from orderDetails
);
```



name	productID
Samsung Galaxy Watch 6	77
Women's Puffer Jacket	114
Lip Balm - SPF	176
Women's Pleated Skirt	179
Webcam Full HD	246

16. Show customers who have only written reviews with 5-star ratings.

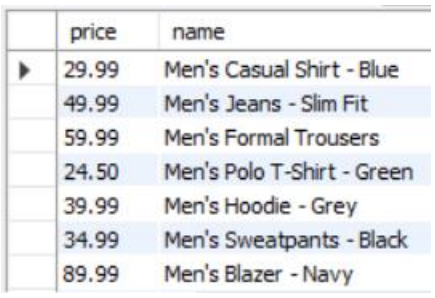
```
SELECT Name, Email FROM Customers
WHERE CustomerID IN (
SELECT CustomerID
FROM Reviews
WHERE Rating = 5
);
```



Name	Email
Hazel Delgado	hazel.delgado@example.com
Xylia Sanchez	xylia.sanchez_3@gmail.com
Daniel Cooper	daniel.cooper@gmail.com
Sam Baker Jr	sam.baker_jr@gmail.com
Xenia Padilla	xenia.padilla@example.com
Ethan Green	ethan.green@gmail.com
Ursula Nelson	ursula.nelson@gmail.com

17. Find products with a price higher than the average price.

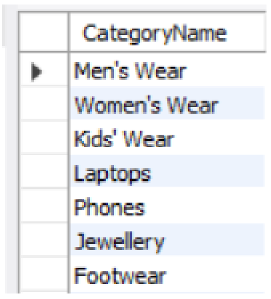
```
select price, name from products where (
select AVG(price)
from products
);
```



price	name
29.99	Men's Casual Shirt - Blue
49.99	Men's Jeans - Slim Fit
59.99	Men's Formal Trousers
24.50	Men's Polo T-Shirt - Green
39.99	Men's Hoodie - Grey
34.99	Men's Sweatpants - Black
89.99	Men's Blazer - Navy

18. Find categories that have more than 5 products.

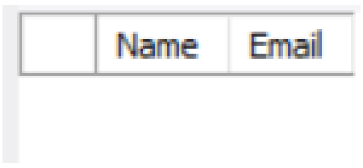
```
SELECT CategoryName FROM Categories
WHERE CategoryID IN (
SELECT CategoryID FROM Products
GROUP BY CategoryID
HAVING COUNT(*) > 5
);
```



CategoryName
Men's Wear
Women's Wear
Kids' Wear
Laptops
Phones
Jewellery
Footwear

19. Show customers who have placed more than 3 orders.

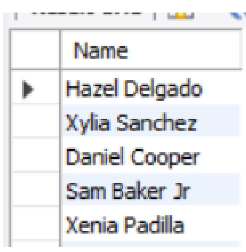
```
select Name, Email from customers
where customerID IN (
select customerID
from Orders GROUP BY CustomerID
HAVING COUNT(OrderID) > 3
);
```



Name	Email

20. List the names of customers who gave a 5-star review.

```
Select name from customers
where customerID IN (
select customerID FROM Reviews
WHERE Rating = 5
 );
```

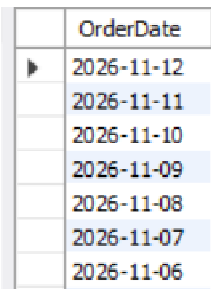


Name
Hazel Delgado
Xylia Sanchez
Daniel Cooper
Sam Baker Jr
Xenia Padilla

- **DATE FUNCTIONS**

21. Find all orders placed in the last 300 days.

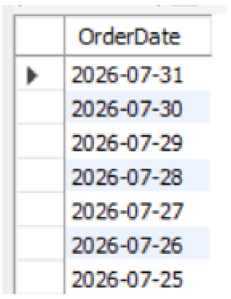
```
select OrderDate
from orders
wHERE datediff(curdate(),OrderDate) <=300;
```



OrderDate
2026-11-12
2026-11-11
2026-11-10
2026-11-09
2026-11-08
2026-11-07
2026-11-06

22. List products ordered in the July 2026.

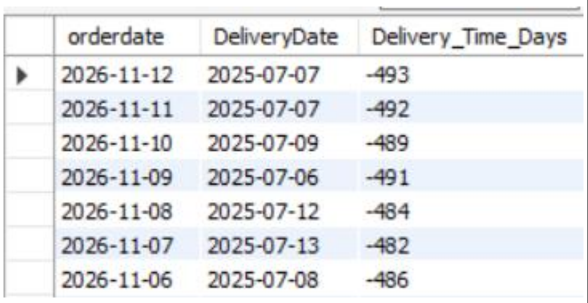
```
SELECT OrderDate FROM Orders
WHERE monthname(OrderDate) = 'July'
&& year(OrderDate) = 2026;
```



OrderDate
2026-07-31
2026-07-30
2026-07-29
2026-07-28
2026-07-27
2026-07-26
2026-07-25

23. Show the delivery time (in days) for each order.

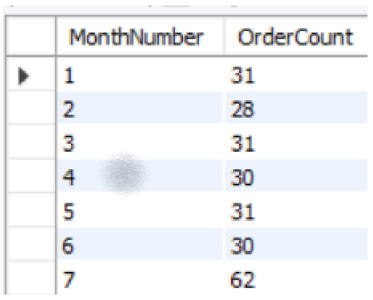
```
SELECT Orders.orderdate, shipping.DeliveryDate,
DATEDIFF(Shipping.DeliveryDate, Orders.OrderDate)
AS Delivery_Time_Days
FROM Orders
JOIN Shipping ON Orders.OrderID = Shipping.OrderID;
```



orderdate	DeliveryDate	Delivery_Time_Days
2026-11-12	2025-07-07	-493
2026-11-11	2025-07-07	-492
2026-11-10	2025-07-09	-489
2026-11-09	2025-07-06	-491
2026-11-08	2025-07-12	-484
2026-11-07	2025-07-13	-482
2026-11-06	2025-07-08	-486

24. Count orders per month.

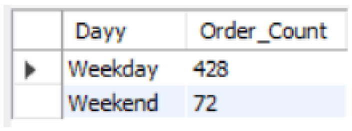
```
select month(OrderDate) AS MonthNumber,
COUNT(*) AS OrderCount
FROM Orders
GROUP BY MONTH(OrderDate)
ORDER BY MonthNumber ASC;
```



MonthNumber	OrderCount
1	31
2	28
3	31
4	30
5	31
6	30
7	62

25. Find how many orders were placed on weekends.

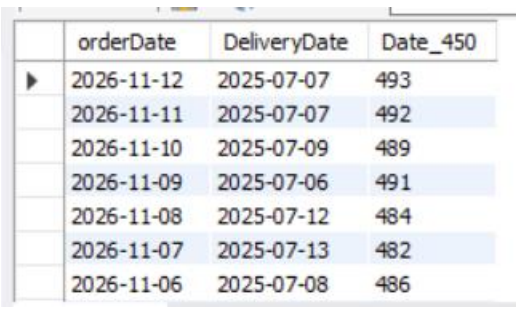
```
select case
when dayname(OrderDate) in (1,7) then 'Weekend'
else 'Weekday'
end as Dayy,
count(*) as Order_Count
from orders
group by Dayy;
```



Dayy	Order_Count
Weekday	428
Weekend	72

26. Show orders that took more than 5 days to deliver.

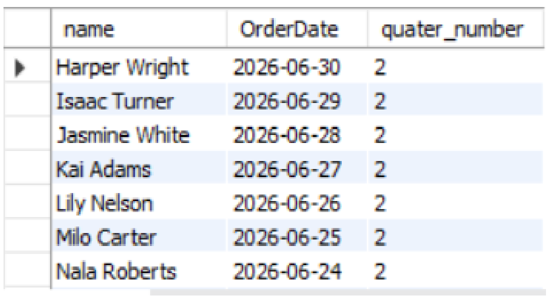
```
select Orders.orderDate, shipping.DeliveryDate,
datediff(Orders.orderDate,shipping.DeliveryDate) as Date_450
from orders
Join shipping on orders.OrderID= shipping.OrderID
where datediff(Orders.orderDate,shipping.DeliveryDate)  >=450;
```



orderDate	DeliveryDate	Date_450
2026-11-12	2025-07-07	493
2026-11-11	2025-07-07	492
2026-11-10	2025-07-09	489
2026-11-09	2025-07-06	491
2026-11-08	2025-07-12	484
2026-11-07	2025-07-13	482
2026-11-06	2025-07-08	486

27. **Find customers who placed orders only in the 2nd quater**.

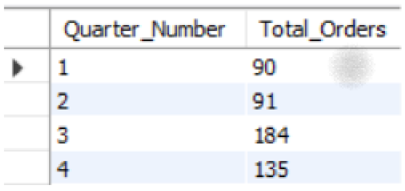
```
select customers.name, orders.OrderDate,
quarter(orders.orderDate)
as quater_number from customers
join orders on customers.CustomerID = orders.CustomerID
where quarter(orders.orderDate)= 2 ;
```



name	OrderDate	quater_number
Harper Wright	2026-06-30	2
Isaac Turner	2026-06-29	2
Jasmine White	2026-06-28	2
Kai Adams	2026-06-27	2
Lily Nelson	2026-06-26	2
Milo Carter	2026-06-25	2
Nala Roberts	2026-06-24	2

28. Find total count per Quarter of orders.

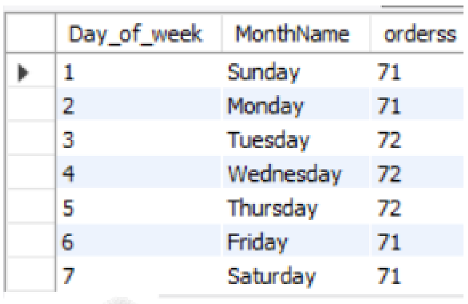
```
select quater(orderDate) AS Quarter_Number,
count(*) AS Total_Orders
from orders
group by Quarter_Number
order by Quarter_Number;
```



Quarter_Number	Total_Orders
1	90
2	91
3	184
4	135

29. Show Order per week.

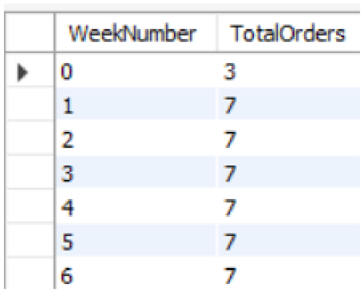
```
select dayofweek(OrderDate) as Day_of_week ,
DAYNAME(OrderDate) as MonthName,
count(*) as orders
from orders
group by Day_of_week, MonthName
order by Day_of_week ASC;
```



Day_of_week	MonthName	orderss
1	Sunday	71
2	Monday	71
3	Tuesday	72
4	Wednesday	72
5	Thursday	72
6	Friday	71
7	Saturday	71

28. Show total orders per week for the last 6 months.

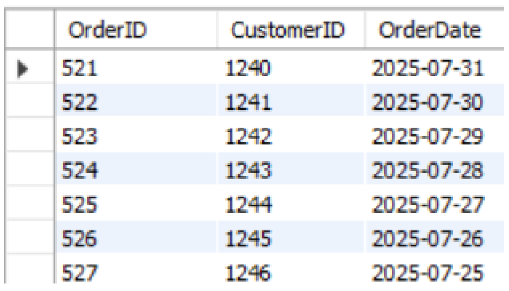
```
Select WEEK(OrderDate) AS WeekNumber,
count(*) AS TotalOrders
FROM Orders
WHERE OrderDate >= curdate() – interval 6 month
GROUP BY WeekNumber
ORDER BY WeekNumber;
```



WeekNumber	TotalOrders
0	3
1	7
2	7
3	7
4	7
5	7
6	7

29. Show all orders placed in the current month.

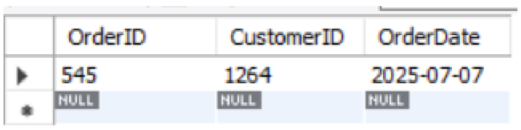
```
SELECT *
FROM Orders
WHERE MONTH(OrderDate) = MONTH(CURDATE())
AND YEAR(OrderDate) = YEAR(CURDATE());
```



OrderID	CustomerID	OrderDate
521	1240	2025-07-31
522	1241	2025-07-30
523	1242	2025-07-29
524	1243	2025-07-28
525	1244	2025-07-27
526	1245	2025-07-26
527	1246	2025-07-25

30. Show all orders placed yesterday.

```
SELECT *
FROM Orders
WHERE OrderDate = CURDATE() - INTERVAL 1 DAY;
```

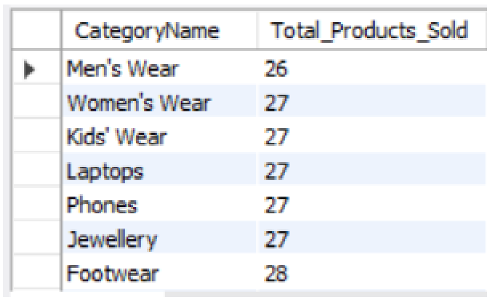


OrderID	CustomerID	OrderDate
545	1264	2025-07-07
NULL	NULL	NULL

- **AGGREGATE FUNCTIONS + GROUP BY + HAVING**

31. Count how many products are in each category.

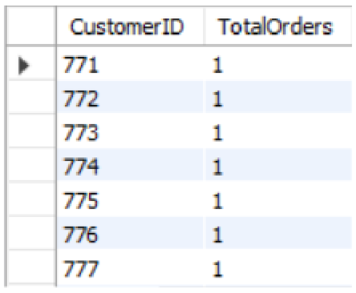
```
select categories.CategoryName,
count(products.ProductID) as Total_Products_Sold
from categories
join products on categories.CategoryID = products.CategoryID
group by categories.CategoryName;
```



CategoryName	Total_Products_Sold
Men's Wear	26
Women's Wear	27
Kids' Wear	27
Laptops	27
Phones	27
Jewellery	27
Footwear	28

32. Total orders placed by each customer

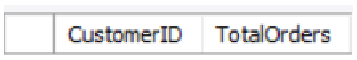
```
select customerID, count(OrderID) as TotalOrders
from orders
group by CustomerID;
```



CustomerID	TotalOrders
771	1
772	1
773	1
774	1
775	1
776	1
777	1

33. Customers who placed more than 3 orders.

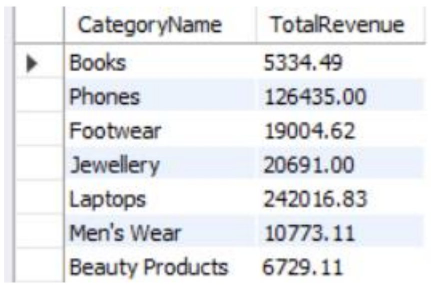
```
select customerID, count(OrderID) AS TotalOrders
from Orders
GROUP BY CustomerID
having COUNT(OrderID) > 3;
```



CustomerID	TotalOrders

34. Total revenue (price × quantity) per category.

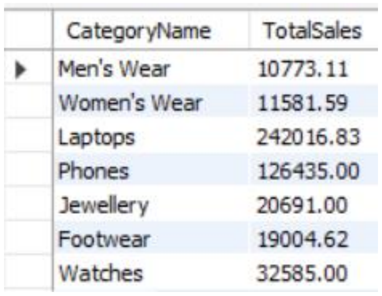
```
SELECT Categories.CategoryName,
SUM(Products.Price * OrderDetails.Quantity) AS TotalRevenue
FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
JOIN Categories ON Products.CategoryID = Categories.CategoryID
GROUP BY Categories.CategoryName;
```



CategoryName	TotalRevenue
Books	5334.49
Phones	126435.00
Footwear	19004.62
Jewellery	20691.00
Laptops	242016.83
Men's Wear	10773.11
Beauty Products	6729.11

35. Categories with total sales over ₹10,000

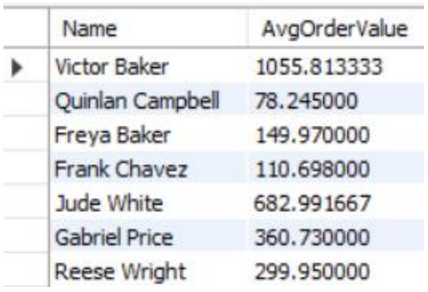
```
SELECT Categories.CategoryName,
SUM(Products.Price * OrderDetails.Quantity) AS TotalRevenue
FROM Orders
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
JOIN Categories ON Products.CategoryID = Categories.CategoryID
GROUP BY Categories.CategoryName
hAVING TotalSales > 10000;
```



CategoryName	TotalSales
Men's Wear	10773.11
Women's Wear	11581.59
Laptops	242016.83
Phones	126435.00
Jewellery	20691.00
Footwear	19004.62
Watches	32585.00

36. Find the average order value per customer.

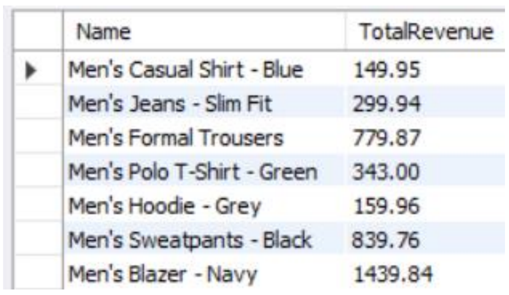
```
SELECT customers.Name,
AVG(OrderDetails.Quantity * Products.Price) AS AvgOrderValue
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Customers.Name;
```



	Name	AvgOrderValue
▶	Victor Baker	1055.813333
	Quinlan Campbell	78.245000
	Freya Baker	149.970000
	Frank Chavez	110.698000
	Jude White	682.991667
	Gabriel Price	360.730000
	Reese Wright	299.950000

37. Show total revenue generated by each product.

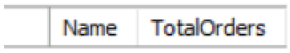
```
SELECT Products.Name,
SUM(OrderDetails.Quantity * Products.Price) AS TotalRevenue
FROM OrderDetails
JOIN Products ON OrderDetails.ProductID = Products.ProductID
GROUP BY Products.Name;
```



	Name	TotalRevenue
▶	Men's Casual Shirt - Blue	149.95
	Men's Jeans - Slim Fit	299.94
	Men's Formal Trousers	779.87
	Men's Polo T-Shirt - Green	343.00
	Men's Hoodie - Grey	159.96
	Men's Sweatpants - Black	839.76
	Men's Blazer - Navy	1439.84

38. List customers who placed more than 5 orders.

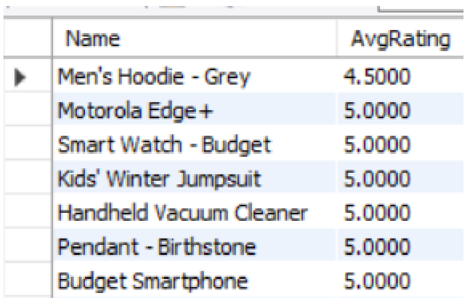
```
SELECT Customers.Name,
COUNT(Orders.OrderID) AS TotalOrders
FROM Customers
JOIN Orders ON Customers.CustomerID = Orders.CustomerID
GROUP BY Customers.CustomerID, Customers.Name
HAVING COUNT(Orders.OrderID) > 5;
```



	Name	TotalOrders

39. Show products with an average rating above 4.0.

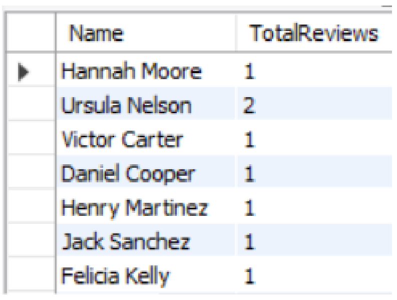
```
SELECT Products.Name, AVG(Reviews.Rating) AS AvgRating
FROM Products
JOIN Reviews ON Products.ProductID = Reviews.ProductID
GROUP BY Products.ProductID, Products.Name
HAVING AVG(Reviews.Rating) > 4;
```



	Name	AvgRating
▶	Men's Hoodie - Grey	4.5000
	Motorola Edge+	5.0000
	Smart Watch - Budget	5.0000
	Kids' Winter Jumpsuit	5.0000
	Handheld Vacuum Cleaner	5.0000
	Pendant - Birthstone	5.0000
	Budget Smartphone	5.0000

40. Count reviews given per customer.

```
SELECT Customers.Name,
COUNT(Reviews.ReviewID) AS TotalReviews
FROM Customers
JOIN Reviews ON Customers.CustomerID = Reviews.CustomerID
GROUP BY Customers.CustomerID, Customers.Name;
```



	Name	TotalReviews
▶	Hannah Moore	1
	Ursula Nelson	2
	Victor Carter	1
	Daniel Cooper	1
	Henry Martinez	1
	Jack Sanchez	1
	Felicia Kelly	1