

**Assignment  
 TechShop**

**Mayank Kumar Jain J513**

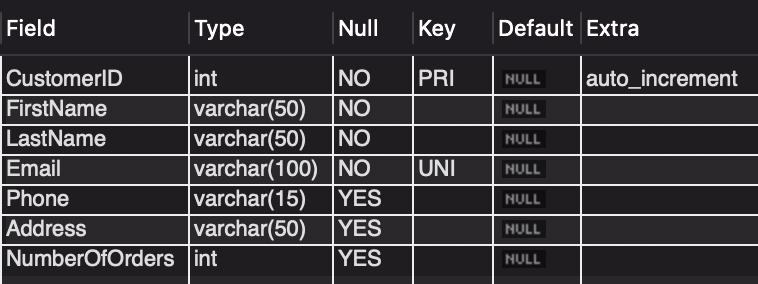
**Task:1. Database Design:**

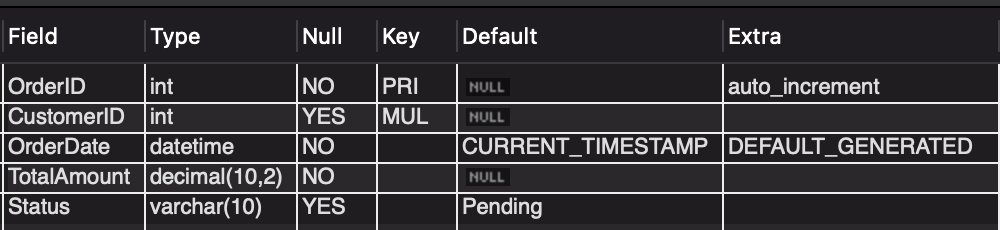
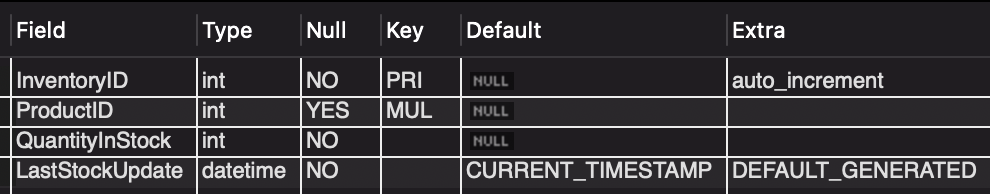
**1.Create the database named "TechShop"**

=> CREATE DATABASE TechShop;

**2. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory tables**

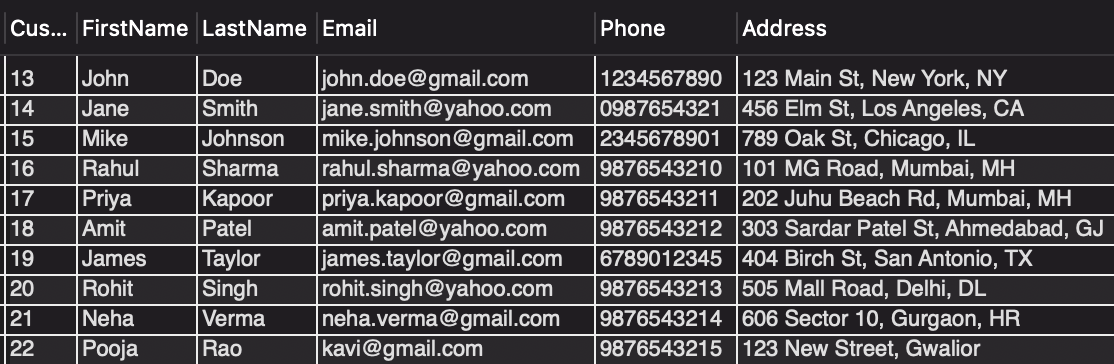
**based on the provided schema.**

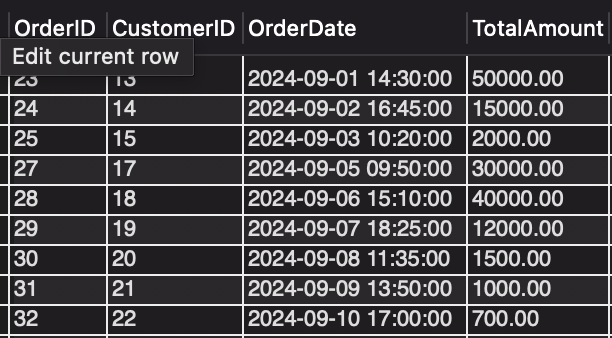
****

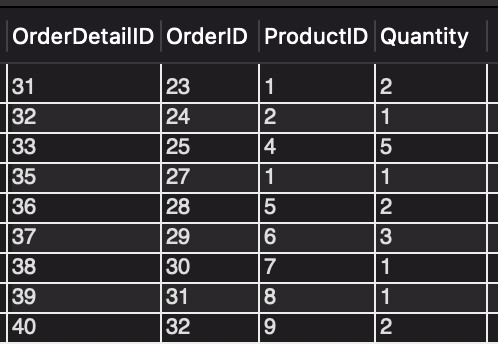


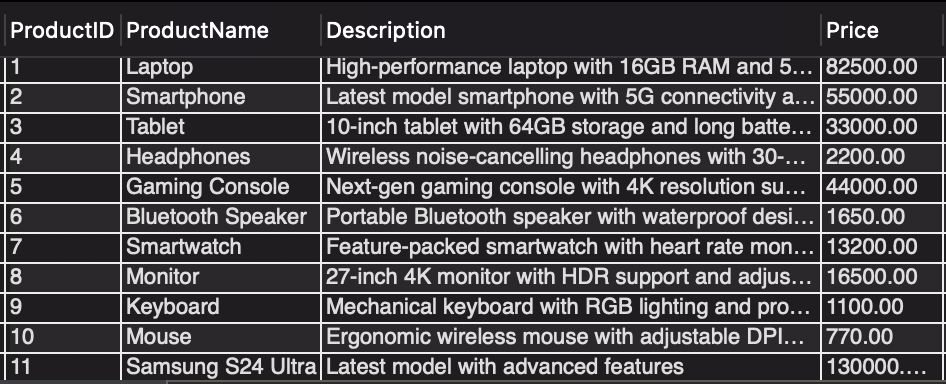
3. Create an ERD (Entity Relationship Diagram) for the database.

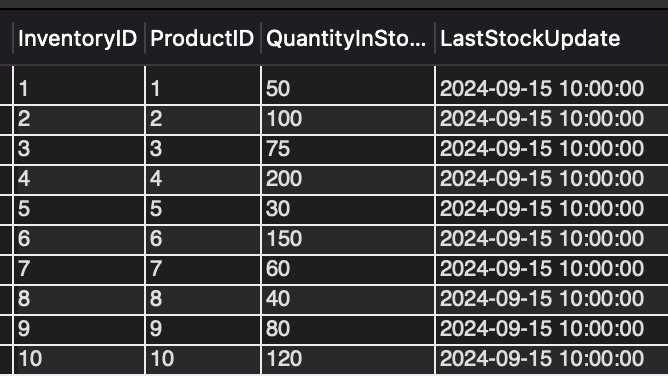
5. Insert at least 10 sample records into each of the following tables.







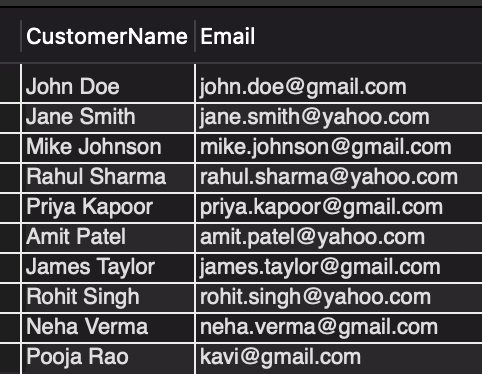




Tasks 2: Select, Where, Between, AND, LIKE:

1. Write an SQL query to retrieve the names and emails of all customers

=> SELECT CONCAT(FirstName, ' ', LastName) AS CustomerName, Email FROM Customers;



2.Write an SQL query to list all orders with their order dates and corresponding customer

Names.

=> select O.OrderID, O.OrderDate, C.FirstName from Orders O JOIN Customers C ON O.CustomerID = C.CustomerID;



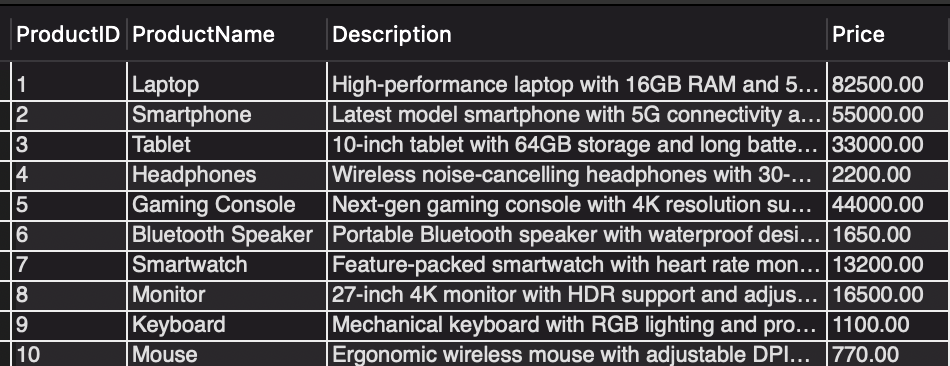
3.Write an SQL query to insert a new customer record into the "Customers" table. Include

customer information such as name, email, and address.

=> INSERT INTO Customers(FirstName, LastName, Email, Phone, Address) VALUES('Rohan', 'Arora', 'rohan@gmail.com', '9836478397, 'City Center, Gwalior’);

4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.

=> UPDATE Products SET Price = Price\*1.1;

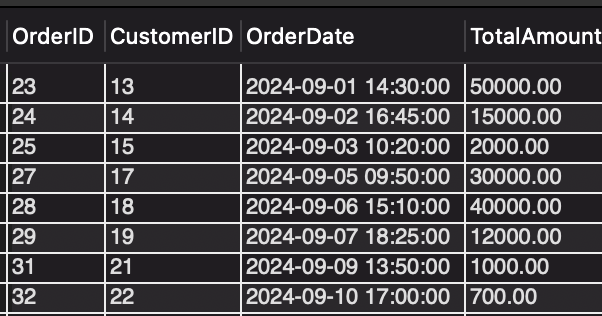


5.Write an SQL query to delete a specific order and its associated order details from the

"Orders" and "OrderDetails" tables. Allow users to input the order ID as a parameter.

=> DELETE FROM Orders

WHERE OrderID = 10;



6.Write an SQL query to insert a new order into the "Orders" table. Include the customer ID,

order date, and any other necessary information.

=> INSERT INTO Orders (CustomerID, OrderDate, TotalAmount)

VALUES (2, '2024-09-30 13:30:00', 60000);



7.Write an SQL query to update the contact information (e.g., email and address) of a specific

customer in the "Customers" table. Allow users to input the customer ID and new contact

information.

=> UPDATE Customers

SET Email = 'kavi@gmail.com', Address = '123 New Street, Gwalior'

WHERE CustomerID = 22;



8.Write an SQL query to recalculate and update the total cost of each order in the "Orders"

table based on the prices and quantities in the "OrderDetails" table.

=> UPDATE Orders o

SET TotalAmount = (

SELECT SUM(od.Quantity \* p.Price)

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE od.OrderID = o.OrderID

GROUP BY od.OrderID

);

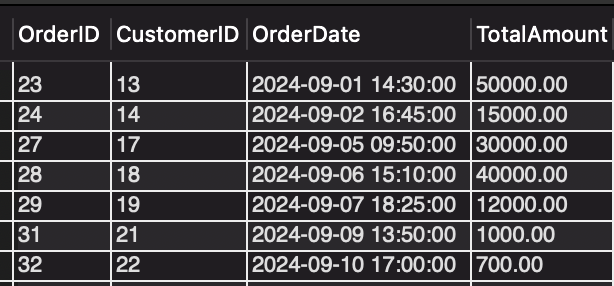
9.Write an SQL query to delete all orders and their associated order details for a specific

customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID

as a parameter.

=> DELETE FROM Orders

WHERE CustomerID = 15;

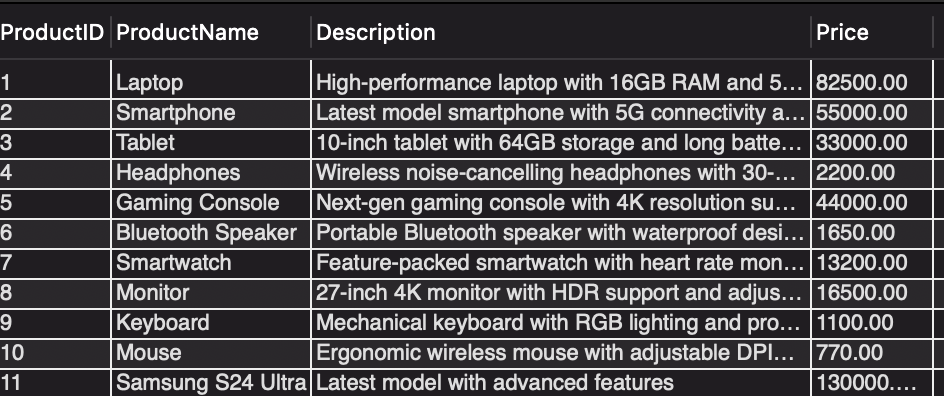


10.Write an SQL query to insert a new electronic gadget product into the "Products" table,

including product name, category, price, and any other relevant details.

=> INSERT INTO Products (ProductName, Description, Price)

VALUES ('Samsung S24 Ultra', 'Latest model with advanced features', 130000);



11.Write an SQL query to update the status of a specific order in the "Orders" table (e.g., from

"Pending" to "Shipped"). Allow users to input the order ID and the new status.

=> ALTER TABLE Orders ADD Status VARCHAR(10) default 'Pending';

UPDATE Orders SET Status = 'Shipped' WHERE OrderID = '30';



12.Write an SQL query to calculate and update the number of orders placed by each customer

in the "Customers" table based on the data in the "Orders" table.

=>

ALTER TABLE Customers

ADD NumberOfOrders INT;

UPDATE Customers C

SET C.NumberOfOrders = (

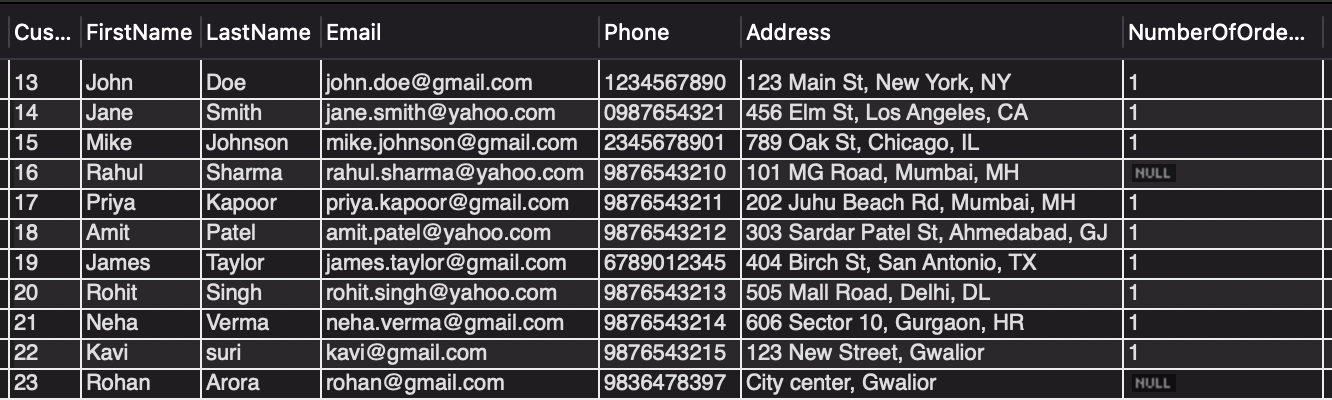
SELECT COUNT(O.OrderID)

FROM Orders O

WHERE O.CustomerID = C.CustomerID

GROUP BY O.CustomerID

);

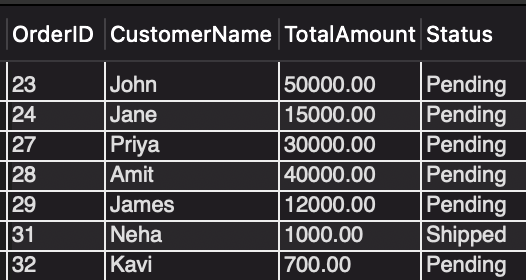


**Task 3. Aggregate functions, Having, Order By, GroupBy and Joins:**

1.Write an SQL query to retrieve a list of all orders along with customer information (e.g.,

customer name) for each order.

=> SELECT o.OrderID, c.FirstName as CustomerName, o.TotalAmount, o.Status FROM Orders o JOIN Customers c ON o.CustomerID = c.CustomerID;



2.Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.

=>

SELECT

p.ProductName,

SUM(od.Quantity \* p.Price) AS TotalRevenue

FROM

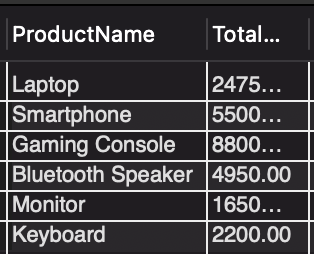
Products AS p

JOIN

OrderDetails AS od ON od.ProductID = p.ProductID

GROUP BY

p.ProductName;



3.Write an SQL query to list all customers who have made at least one purchase. Include their names and contact information.

=>

SELECT

c.FirstName,

c.LastName,

c.Email,

c.Phone

FROM

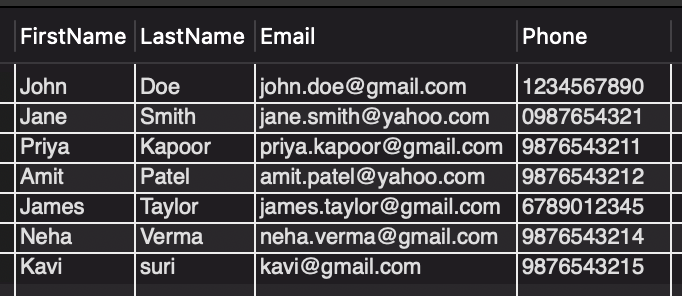
Customers AS c

JOIN

Orders AS o ON c.CustomerID = o.CustomerID

GROUP BY

c.CustomerID, c.FirstName, c.LastName, c.Email, c.Phone;



4.Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.

=>

SELECT

p.ProductName,

SUM(od.Quantity) AS TotalQuantityOrdered

FROM

Products AS p

JOIN

OrderDetails AS od ON p.ProductID = od.ProductID

JOIN

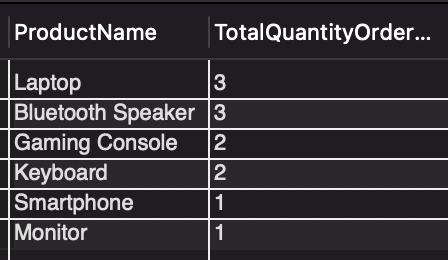
Orders AS o ON od.OrderID = o.OrderID

GROUP BY

p.ProductID, p.ProductName

ORDER BY

TotalQuantityOrdered DESC;



5. Write an SQL query to retrieve a list of electronic gadgets along with their corresponding

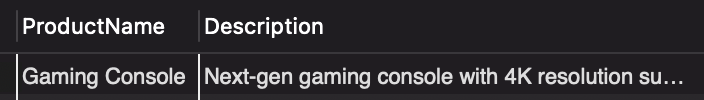
Categories.

=>

SELECT ProductName, Description

FROM Products

WHERE Description = 'Next-gen gaming console with 4K resolution support.';



6.Write an SQL query to calculate the average order value for each customer. Include the

customer's name and their average order value.

=>

SELECT

c.FirstName,

c.LastName,

AVG(o.TotalAmount) AS AverageOrderValue

FROM

Customers AS c

JOIN

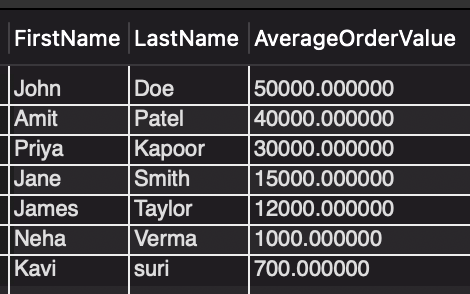
Orders AS o ON c.CustomerID = o.CustomerID

GROUP BY

c.CustomerID, c.FirstName, c.LastName

ORDER BY

AverageOrderValue DESC;



7.Write an SQL query to find the order with the highest total revenue. Include the order ID,

customer information, and the total revenue.

=>

SELECT

o.OrderID,

c.FirstName,

c.LastName,

SUM(od.Quantity \* p.Price) AS TotalRevenue

FROM

Orders AS o

JOIN

Customers AS c ON o.CustomerID = c.CustomerID

JOIN

OrderDetails AS od ON o.OrderID = od.OrderID

JOIN

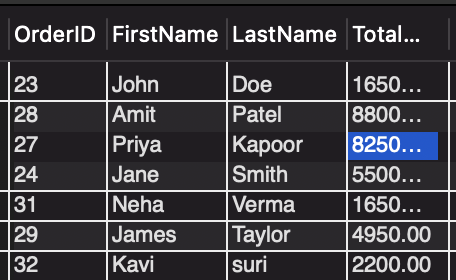
Products AS p ON od.ProductID = p.ProductID

GROUP BY

o.OrderID, c.FirstName, c.LastName

ORDER BY

TotalRevenue DESC;



8.Write an SQL query to list electronic gadgets and the number of times each product has been ordered.

=>

SELECT

p.ProductName,

COUNT(od.ProductID) as NumberOfOrders

FROM

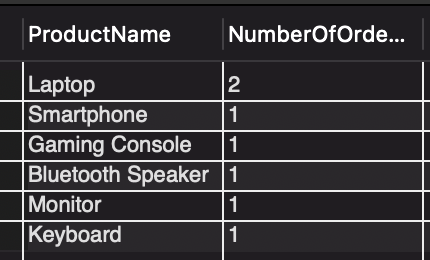
OrderDetails od

JOIN

Products p ON od.productID = p.ProductID

GROUP BY

ProductName;



9.Write an SQL query to find customers who have purchased a specific electronic gadget product. Allow users to input the product name as a parameter.

=>

SELECT

c.FirstName,

p.ProductName

FROM

Customers C

JOIN

Orders o ON c.CustomerID = o.CustomerID

JOIN

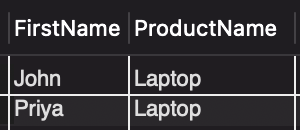
OrderDetails od ON o.OrderID = od.OrderID

JOIN

Products AS p ON od.ProductID = p.ProductID

WHERE

p.ProductName = 'Laptop';



10. Write an SQL query to calculate the total revenue generated by all orders placed within a specific time period. Allow users to input the start and end dates as parameters.

=>

SELECT

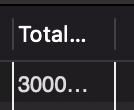
SUM(o.TotalAmount) AS TotalRevenue

FROM

Orders AS o

WHERE

o.OrderDate BETWEEN '2024-09-03' AND '2024-09-06';



**Task 4. Subquery and its type:**

1.Write an SQL query to find out which customers have not placed any orders.

=>

SELECT

c.CustomerID

FROM

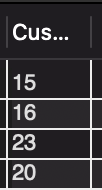
Customers c

WHERE

c.CustomerID NOT IN (

SELECT o.CustomerID

FROM Orders o );



2. Write an SQL query to find the total number of products available for sale.

=>

SELECT

i.ProductID,

(i.QuantityInStock - COALESCE(SUM(od.Quantity), 0)) AS AvailableStock

FROM

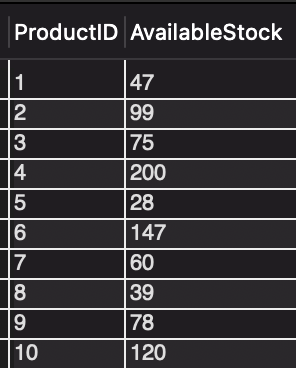
Inventory i

LEFT JOIN

OrderDetails od ON i.ProductID = od.ProductID

GROUP BY

i.ProductID, i.QuantityInStock;



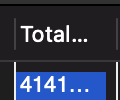
3.Write an SQL query to calculate the total revenue generated by TechShop.

=>

SELECT SUM(od.Quantity \* p.Price) AS TotalRevenue

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID;



4.Write an SQL query to calculate the average quantity ordered for products in a specific category. Allow users to input the category name as a parameter

=>

SELECT AVG(od.Quantity) AS AverageQuantityOrdered

FROM OrderDetails od

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Description = 'Next-gen gaming console with 4K resolution support.';

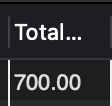
5.Write an SQL query to calculate the total revenue generated by a specific customer. Allow users to input the customer ID as a parameter.

=>

SELECT SUM(TotalAmount) AS TotalRevenue

FROM Orders

WHERE CustomerID = 22;



6.Write an SQL query to find the customers who have placed the most orders. List their names

and the number of orders they've placed.

=>

SELECT c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY OrderCount DESC;



7.Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.

8.Write an SQL query to find the customer who has spent the most money (highest total revenue)

on electronic gadgets. List their name and total spending.

=>

SELECT c.FirstName, c.LastName, SUM(od.Quantity \* p.Price) AS TotalSpending

FROM Customers c

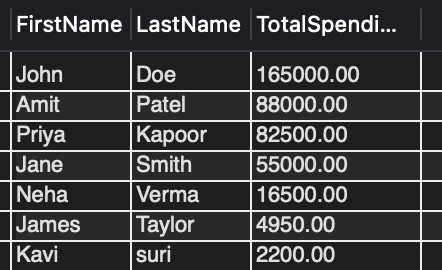
JOIN Orders o ON c.CustomerID = o.CustomerID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY TotalSpending DESC;



9.Write an SQL query to calculate the average order value (total revenue divided by the number of orders) for all customers.

=>

SELECT c.CustomerID, c.FirstName, c.LastName,

SUM(o.TotalAmount) / COUNT(o.OrderID) AS AvgOrderValue

FROM Customers c

JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName;



10.Write an SQL query to find the total number of orders placed by each customer and list their names along with the order count.

=>

SELECT c.CustomerID, c.FirstName, c.LastName, COUNT(o.OrderID) AS OrderCount

FROM Customers c

LEFT JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName;

