ASSIGNMENT-1

# TechShop, an electronic gadgets shop

## Task 1: Database Design

1. Create the database named "TechShop"

Create database techshop;

Use techshop;

1. Define the schema for the Customers, Products, Orders, OrderDetails and Inventory table based on the provided schema.
2. Customers:

Create table customers(

customerId integer(3) Primary key,

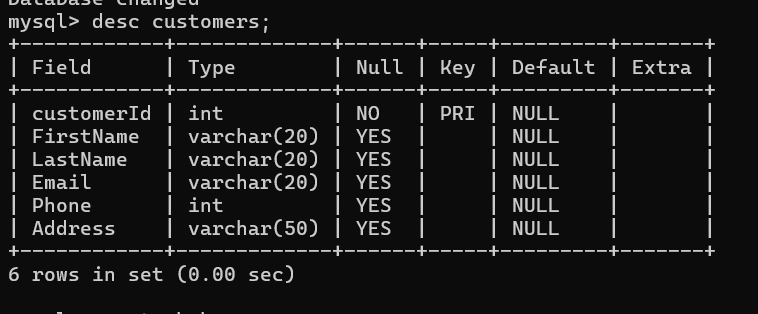
FirstName varchar(20),

LastName varchar(20),

Email varchar(20),

Phone integer(10),

Address varchar(50) );



1. Products:

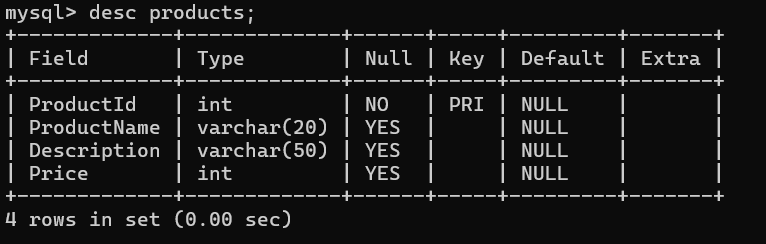
Create table products(

ProductId integer Primary key,

ProductName varchar(20),

Description varchar(50),

Price integer );



1. Orders :

Create table Orders(

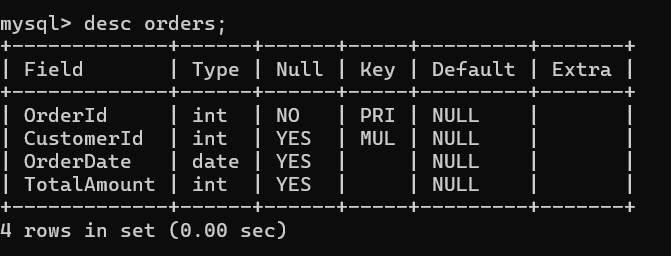
OrderId integer Primary key,

CustomerId integer,

OrderDate date,

TotalAmount integer,

Constraint Orders\_CustomerId\_fk Foreign key(CustomerId) references Customers(CustomerId));



1. OrderDetails:

Create table OrderDetails(

OrderDetailsId integer Primary Key,

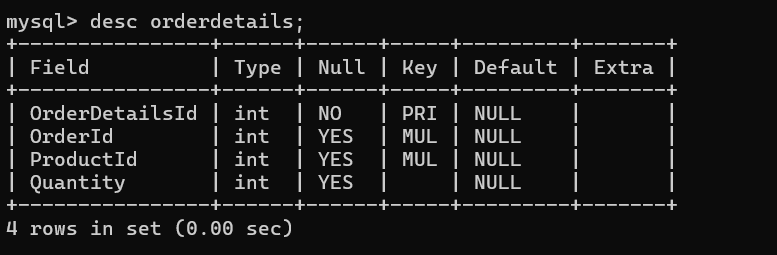
OrderId integer,

ProductId integer,

Quantity integer

Constraint OrderDetails\_OrderId\_fk Foreign key(OrderId) references Orders(OrderId),

Constraint OrderDetails\_ProductId\_fk Foreign key(ProductId) references Products(ProductId));



1. Inventory:

Create table Inventory(

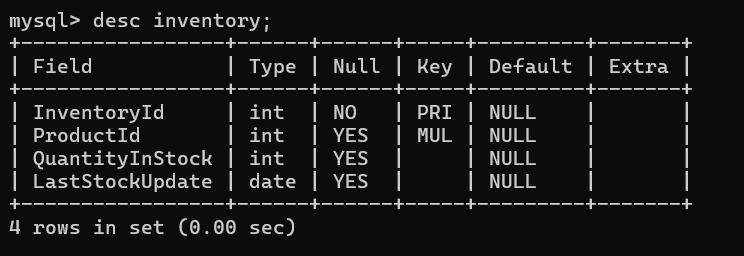
InventoryId integer Primary key,

ProductId integer,

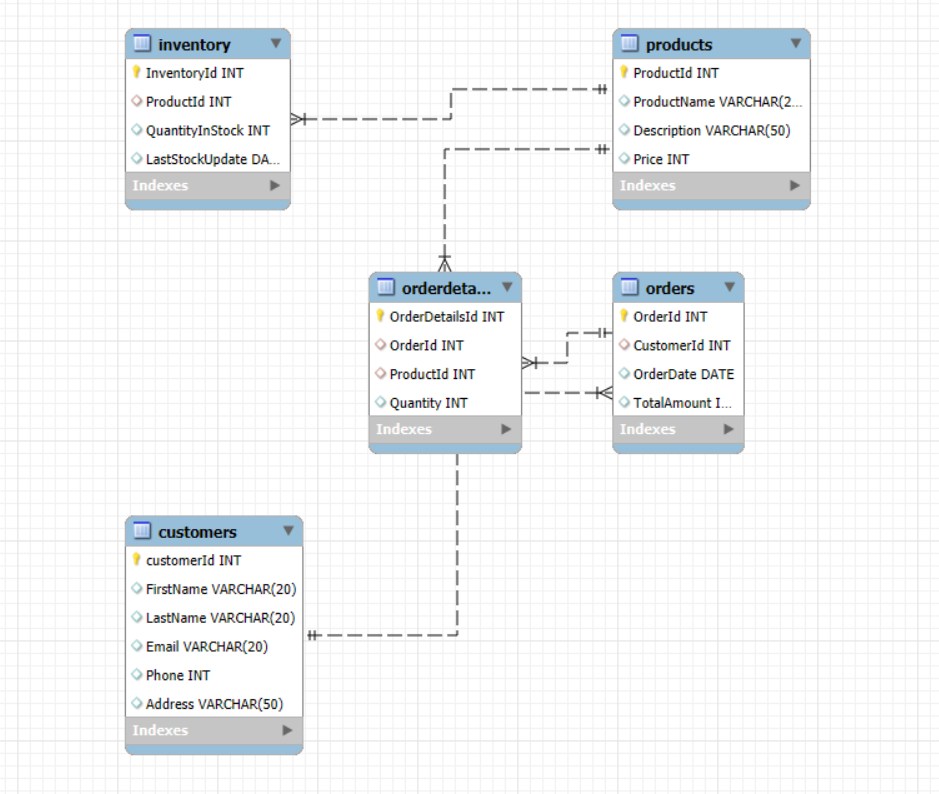
QuantityInStock integer,

LastStockUpdate date

Constraint Inventory\_ProductId\_fk Foreign key(ProductId) references Products(ProductId) );



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



1. Create appropriate Primary Key and Foreign Key constraints for referential integrity.
2. Customer:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(20) NOT NULL,

LastName VARCHAR(20) NOT NULL,

Email VARCHAR(20) NOT NULL UNIQUE,

Phone INT,

Address VARCHAR(50) );

1. Products:

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(20) ,

Price DECIMAL(10,2),

Description VARCHAR(50)NOT NULL);

1. Orders:

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID) ON DELETE CASCADE );

4.OrderDetails:

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT NOT NULL,

Contraint OrderDetails\_OrderId\_fk FOREIGN KEY (OrderID) REFERENCES Orders(OrderID) ON DELETE CASCADE,

Contraint OrderDetails\_ProductId\_fk FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE );

v. Inventory:

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY,

ProductID INT,

Contraint Inventory\_ProductId\_fk FOREIGN KEY (ProductID) REFERENCES Products(ProductID) ON DELETE CASCADE );

1. Insert at least 10 sample records into each of the following tables.
2. Customers:

insert into customers values(101,'John', 'Doe', 'johndoe@gmail.com',9876543, '123 Main St, New York');

insert into customers values(102,'Jane', 'Smith', 'janesmith@gmail.com', '9873210', '456 Oak St, Los Angeles');

insert into customers values (103,'Alice', 'Johnson', 'alicej@gmail.com', '1234567', '789 Pine St, Chicago');

insert into customers values (104,'Bob', 'Williams', 'bobwill@gmail.com', '3334444', '321 Maple st');

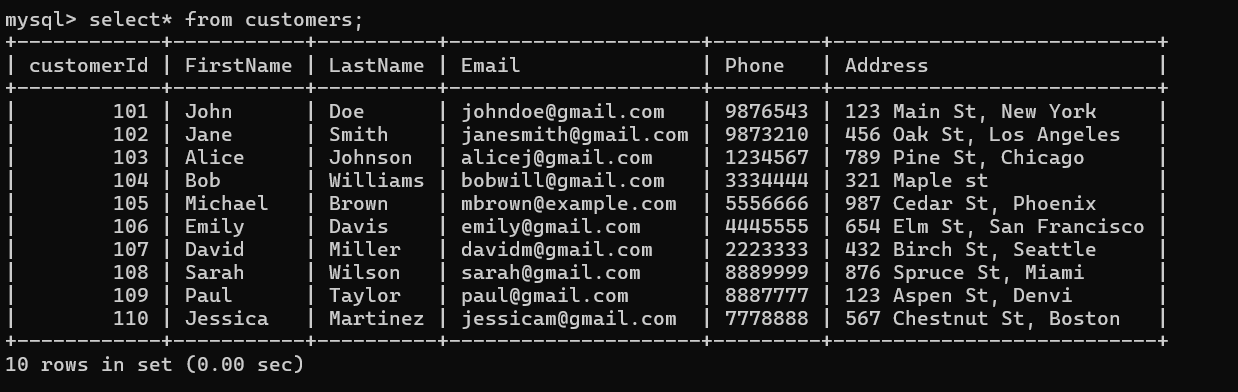
insert into customers values (105,'Michael', 'Brown', 'mbrown@example.com', '5556666', '987 Cedar St, Phoenix');

insert into customers values (106,'Emily', 'Davis', 'emily@gmail.com', '4445555', '654 Elm St, San Francisco');

insert into customers values (107,'David', 'Miller', 'davidm@gmail.com', '2223333', '432 Birch St, Seattle');

insert into customers values (108,'Sarah', 'Wilson', 'sarah@gmail.com', '8889999', '876 Spruce St, Miami');

insert into customers values (109,'Paul', 'Taylor', 'paul@gmail.com', '8887777', '123 Aspen St, Denvi');

 insert into customers values (110,'Jessica', 'Martinez', 'jessicam@gmail.com', '7778888', '567 Chestnut St, Boston');

1. Products:

insert into products values(1,'Smartphone', '5G smartphone, 128GB', 20000);

insert into products values(2,'Laptop', 'RTX 3070, 16GB RAM',50000);

insert into products values(3,'BluetoothEarbuds', 'Noise cancelling',3000);

insert into products values(4,'Smart TV', '55inch UHD TV',60000);

insert into products values(5,'PowerBank', '20,000mAh fast',2000);

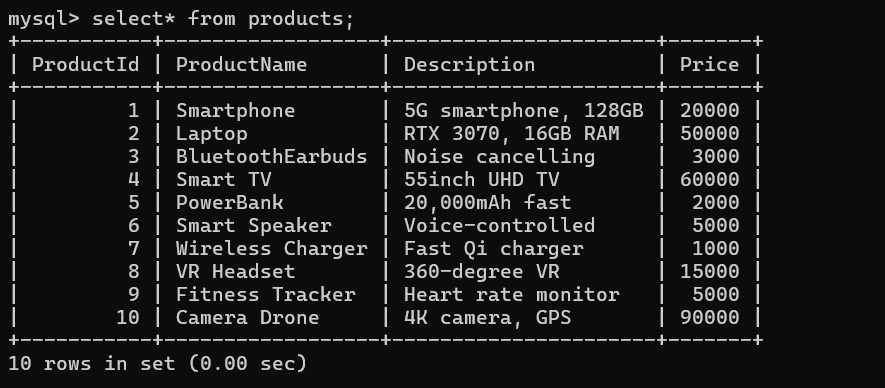
insert into products values(6,'Smart Speaker', 'Voice-controlled',5000);

insert into products values(7,'Wireless Charger', 'Fast Qi charger',1000);

insert into products values(8,'VR Headset', '360-degree VR',15000);

insert into products values(9,'Fitness Tracker', 'Heart rate monitor',5000);

insert into products values(10,'Camera Drone', '4K camera, GPS',90000);



1. Orders:

insert into orders values(1001,101,'2024-09-15',5000);

insert into orders values(1002,103,'2024-07-14',1000);

insert into orders values(1003,104,'2024-06-24',10000);

insert into orders values(1004,102,'2024-06-16',50000);

insert into orders values(1005,102,'2024-05-16',45000);

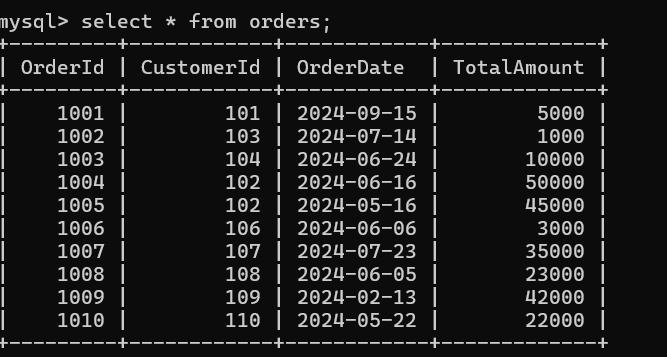
insert into orders values(1006,106,'2024-06-06',3000);

insert into orders values(1007,107,'2024-07-23',35000);

insert into orders values(1008,108,'2024-06-05',23000);

insert into orders values(1009,109,'2024-02-13',42000);

insert into orders values(1010,110,'2024-05-22',22000);



1. OrderDetails:

insert into orderdetails values(31,1001,1,500);

insert into orderdetails values(32,1005,2,200);

insert into orderdetails values(33,1003,3,100);

insert into orderdetails values(34,1002,4,50);

insert into orderdetails values(35,1003,5,600);

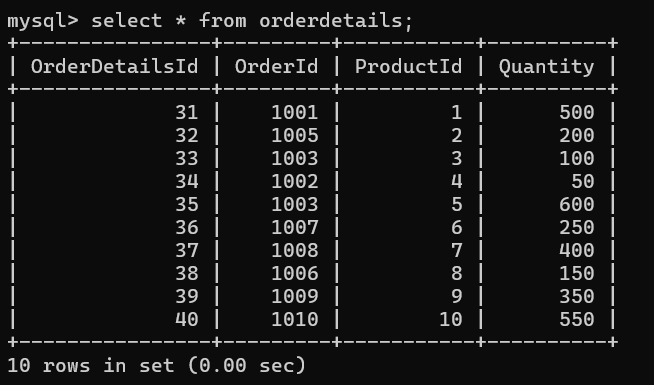
insert into orderdetails values(36,1007,6,250);

insert into orderdetails values(37,1008,7,400);

insert into orderdetails values(38,1006,8,150);

insert into orderdetails values(39,1009,9,350);

insert into orderdetails values(40,1010,10,550);



1. Inventory:

insert into inventory values (11,1,15,'2024-05-10');

insert into inventory values (12,2,25,’2024-05-15’);

insert into inventory values (13,3,10,’2024-05-20’);

insert into inventory values (14,4,05,’2024-05-13’);

insert into inventory values (15,5,25,’2024-05-17’);

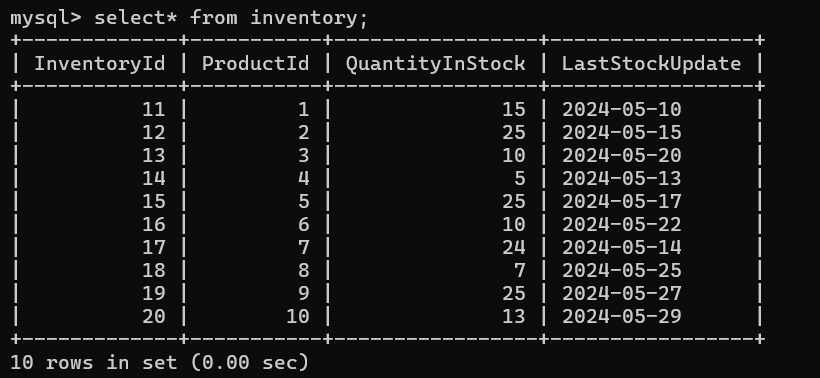
insert into inventory values (16,6,10,’2024-05-22’);

insert into inventory values (17,7,24,’2024-05-14’);

insert into inventory values (18,8,07,’2024-05-25’);

insert into inventory values (19,9,25,’2024-05-27’);

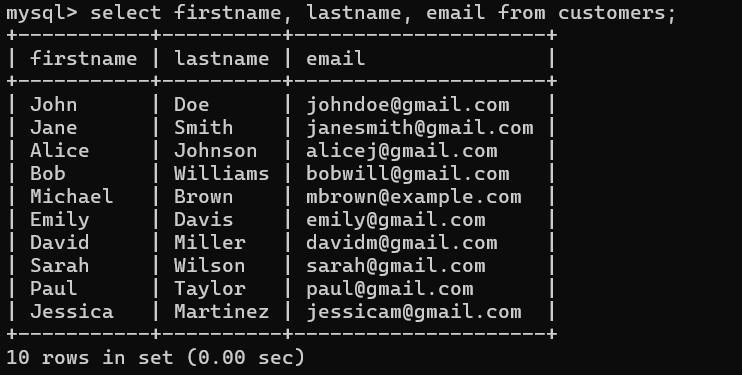
insert into inventory values (20,10,13,’2024-05-29’);



## Task 2: Select, Where, Between, AND, LIKE:

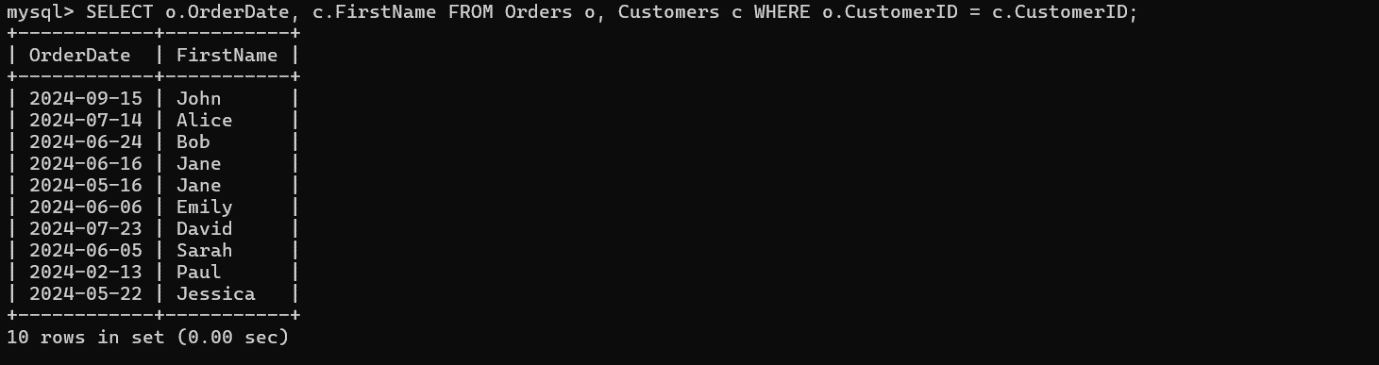
1. Write an SQL query to retrieve the names and emails of all customers.

Query: select firstname, lastname, email from customers;



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

Query: SELECT o.OrderDate, c.FirstName FROM Orders o, Customers c WHERE o.CustomerID = c.CustomerID;



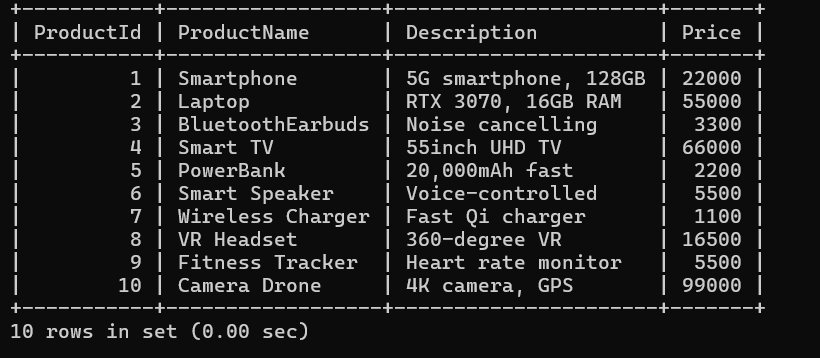
1. Write an SQL query to insert a new customer record into the "Customers" table. Include customer information such as name, email, and address.

Query: insert into customers values(111,'John','Mathew','john234@gmail.com',null,'678,Town St,USA');



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

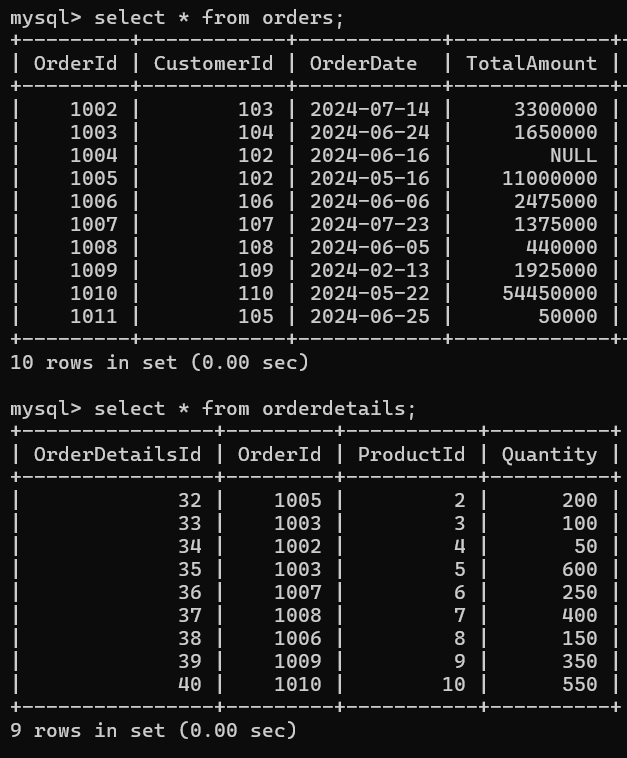
Query: update products set price = price \*1.10 ;



1. 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.

Query: DELETE FROM OrderDetails WHERE OrderID =1001;

DELETE FROM Orders WHERE OrderID = 1001;



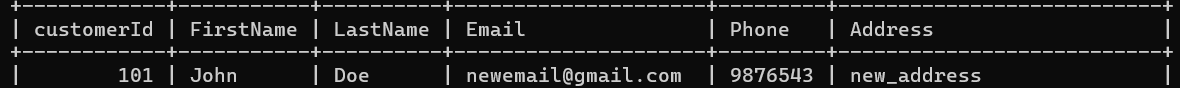
1. Write an SQL query to insert a new order into the "Orders" table. Include the customer ID, order date, and any other necessary information.

Query: insert into orders values (1011,105,'2024-06-25',50000);



1. 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.

Query: UPDATE Customers SET Email = 'newemail@gmail.com',Address = 'new\_address' WHERE CustomerID = 101;



1. 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.

Query: SELECT OrderID,

SUM(

Quantity \* (

SELECT Price FROM Products WHERE ProductID = od.ProductID))

AS TotalAmount

FROM OrderDetails od GROUP BY OrderID;

UPDATE Orders

SET TotalCost = CASE WHEN OrderID = 1001 THEN

(SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1001)

WHEN OrderID = 1002 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1002)

WHEN OrderID = 1003 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1003)

WHEN OrderID = 1004 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1004)

WHEN OrderID = 1005 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1005)

WHEN OrderID = 1006 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1006)

WHEN OrderID = 1007 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1007)

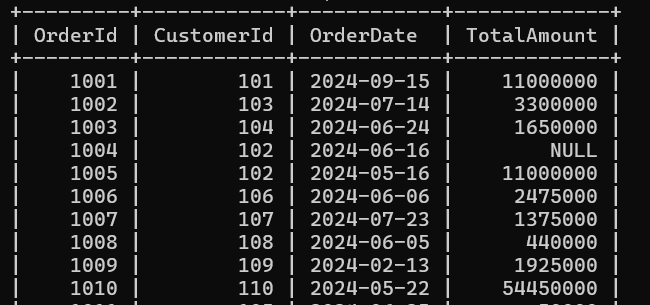
WHEN OrderID = 1008 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID =1008)

WHEN OrderID = 1009 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1009)

WHEN OrderID = 1010 THEN (SELECT SUM(Quantity \* (SELECT Price FROM Products WHERE ProductID = od.ProductID)) FROM OrderDetails od WHERE od.OrderID = 1010)

ELSE TotalCost

END WHERE OrderID IN (1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010);

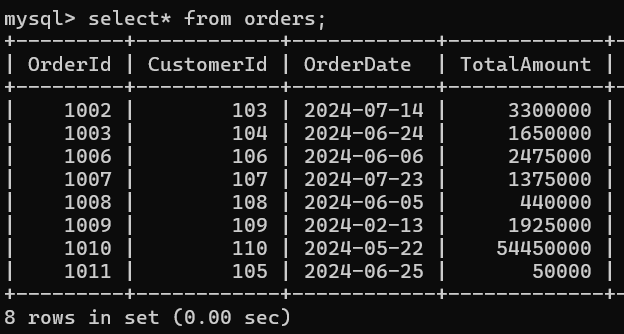


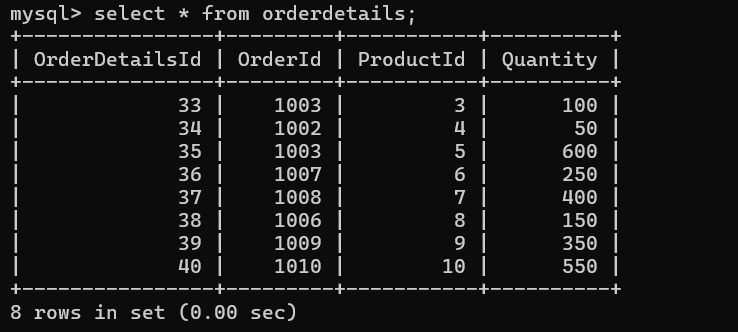
1. Write an SQL query to delete all orders and their associated order details for a specific customer from the "Orders" and "OrderDetails" tables. Allow users to input the customer ID as a parameter.

Query: DELETE FROM OrderDetails WHERE OrderID IN (

SELECT OrderID FROM Orders WHERE CustomerID = 102);

DELETE FROM Orders WHERE CustomerID = 102;





1. 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.

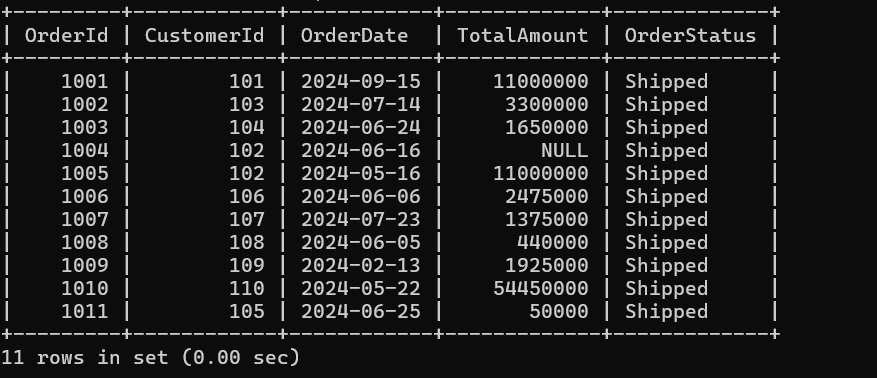
Query: insert into products values(11,'SmartWatch','Advanced Watch',5000);



1. 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.

Query: alter table orders add column OrderStatus varchar(50) ;

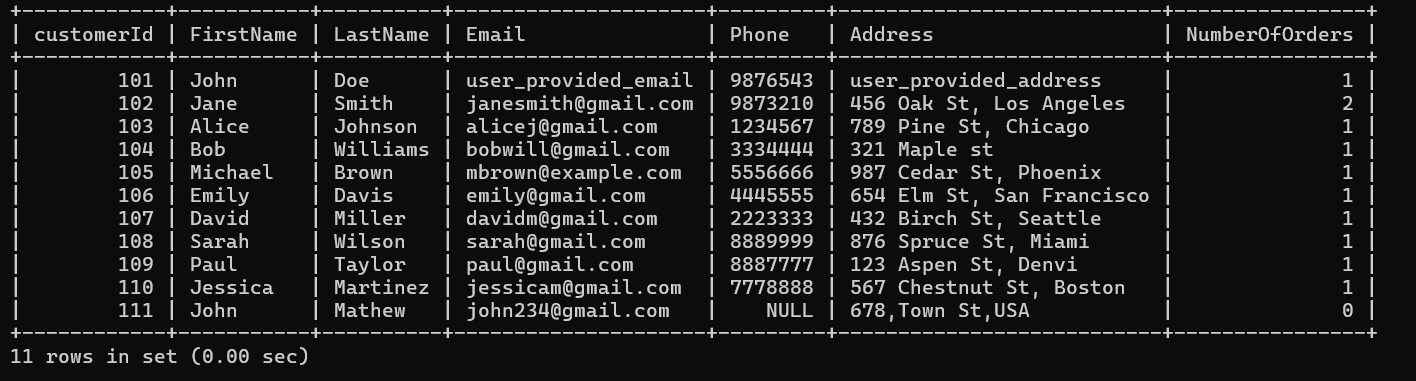
update orders set orderstatus ='Shipped' where orderId in(1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1011);



1. 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.

Query: ALTER TABLE Customers ADD COLUMN NumberOfOrders INT DEFAULT 0;

UPDATE Customers SET NumberOfOrders = ( SELECT COUNT(\*) FROM Orders WHERE Orders.CustomerID = Customers.CustomerID ) WHERE CustomerID IN (SELECT DISTINCT CustomerID FROM Orders);

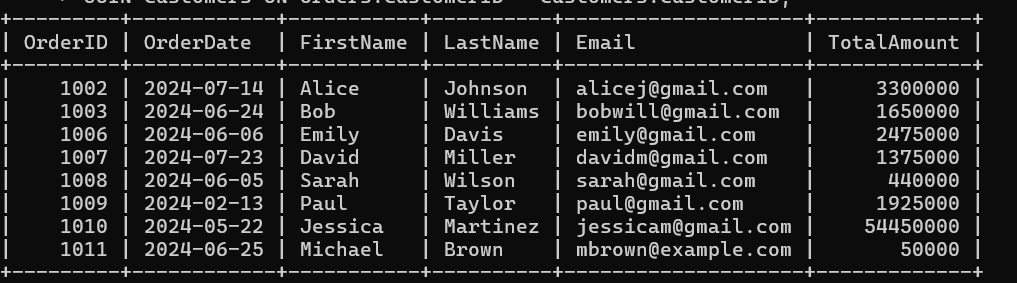


## Task 3: Aggregate functions, Having, Order By, GroupBy and Joins:

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

Query: SELECT Orders.OrderID, Orders.OrderDate, Customers.FirstName, Customers.LastName, Customers.Email, Orders.TotalAmount

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

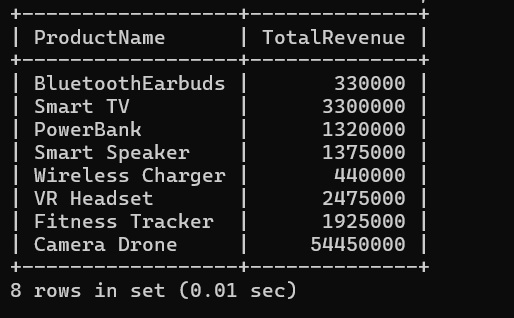


1. Write an SQL query to find the total revenue generated by each electronic gadget product. Include the product name and the total revenue.  
    Query: SELECT Products.ProductName,

SUM(OrderDetails.Quantity \* Products.Price) AS TotalRevenue

FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.ProductID;



1. Write an SQL query to list all customers who have made at least one purchase. Include their

names and contact information.

Query: SELECT DISTINCT Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email, Customers.Phone, Customers.Address

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



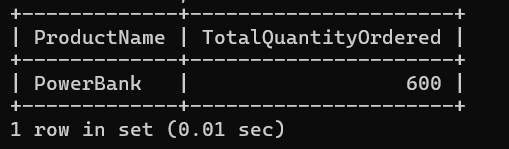
1. Write an SQL query to find the most popular electronic gadget, which is the one with the highest total quantity ordered. Include the product name and the total quantity ordered.

Query: SELECT Products.ProductName,

SUM(OrderDetails.Quantity) AS TotalQuantityOrdered

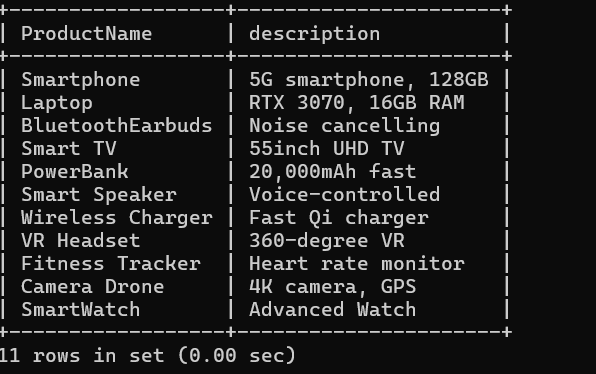
FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID

GROUP BY Products.ProductID ORDER BY TotalQuantityOrdered DESC LIMIT 1;



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

Query: SELECT ProductName, description from products;



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

customer's name and their average order value.

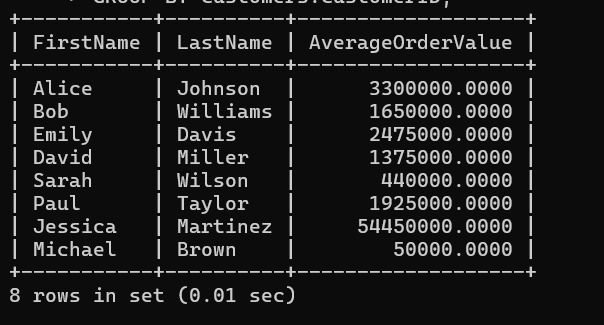
Query: SELECT Customers.FirstName, Customers.LastName,

AVG(Orders.TotalAmount) AS AverageOrderValue

FROM Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

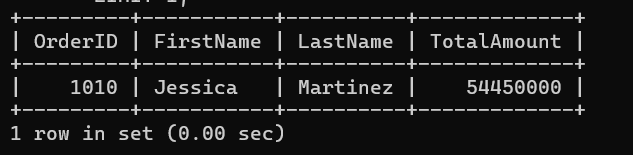
GROUP BY Customers.CustomerID;



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

customer information, and the total revenue.

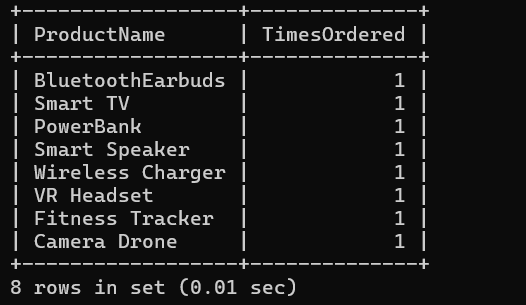
Query: SELECT Orders.OrderID, Customers.FirstName, Customers.LastName, Orders.TotalAmount FROM Orders JOIN Customers ON Orders.CustomerID = Customers.CustomerID ORDER BY Orders.TotalAmount DESC LIMIT 1;



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

ordered.

Query: SELECT Products.ProductName, COUNT(OrderDetails.OrderDetailsID) AS TimesOrdered FROM Products JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID GROUP BY Products.ProductID;

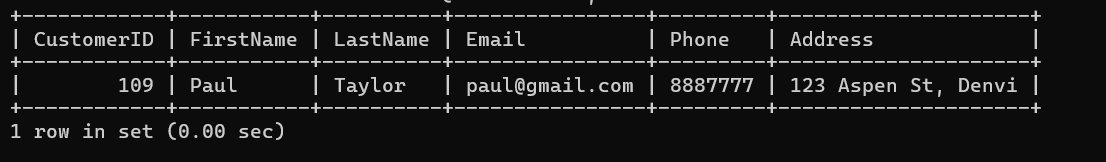


1. Write an SQL query to find customers who have purchased a specific electronic gadget product.Allow users to input the product name as a parameter.  
   Query: Set @productname=’fitness tracker’;

SELECT DISTINCT Customers.CustomerID, Customers.FirstName, Customers.LastName, Customers.Email, Customers.Phone, Customers.Address

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

JOIN Products ON OrderDetails.ProductID = Products.ProductID

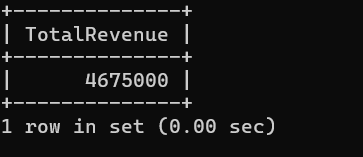
WHERE Products.ProductName = @ProductName;

1. 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.

Query: SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders

WHERE OrderDate BETWEEN '2024-07-14' AND '2024-07-23';

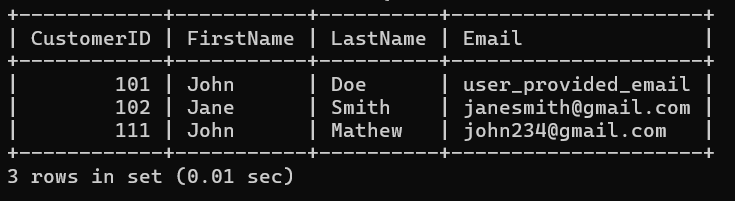


## Task4: Subquery and its type

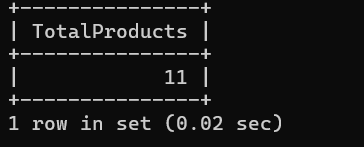
1. Write an SQL query to find out which customers have not placed any orders.

Query: SELECT CustomerID, FirstName, LastName, Email

FROM Customers WHERE CustomerID NOT IN (SELECT CustomerID FROM Orders);

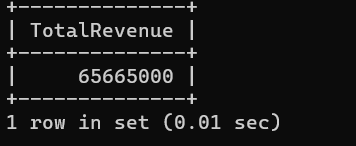


1. Write an SQL query to find the total number of products available for sale.

Query: SELECT COUNT(\*) AS TotalProducts FROM Products;

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

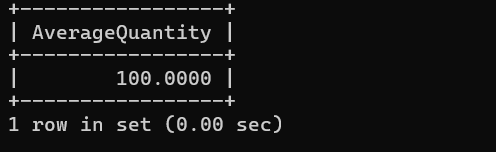
Query: SELECT SUM(TotalAmount) AS TotalRevenue FROM Orders;



1. 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.

Query: SELECT AVG(OrderDetails.Quantity) AS AverageQuantity FROM OrderDetails JOIN Products ON OrderDetails.ProductID = Products.ProductID

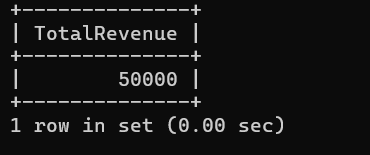
WHERE Products.ProductName = ‘BluetoothEarbuds’;



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

Query: SELECT SUM(Orders.TotalAmount) AS TotalRevenue FROM Orders

WHERE Orders.CustomerID = 105;



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

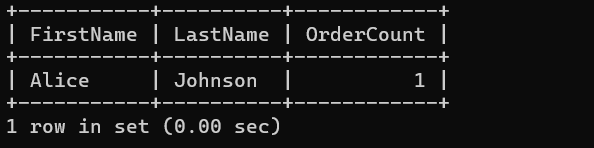
and the number of orders they've placed.

Query: SELECT Customers.FirstName, Customers.LastName, COUNT(Orders.OrderID)

AS OrderCount FROM Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

GROUP BY Customers.CustomerID ORDER BY OrderCount DESC LIMIT 1;



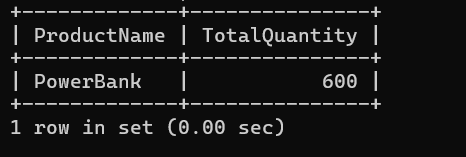
1. Write an SQL query to find the most popular product category, which is the one with the highest total quantity ordered across all orders.

Query: SELECT Products.ProductName, SUM(OrderDetails.Quantity) AS TotalQuantity FROM ProductsJOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID

GROUP BY Products.ProductName

ORDER BY TotalQuantity DESC

LIMIT 1;



1. 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.

Query: SELECT Customers.FirstName, Customers.LastName, SUM(OrderDetails.Quantity \* Products.Price) AS TotalSpending FROM Customers

JOIN Orders ON Customers.CustomerID = Orders.CustomerID

JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID

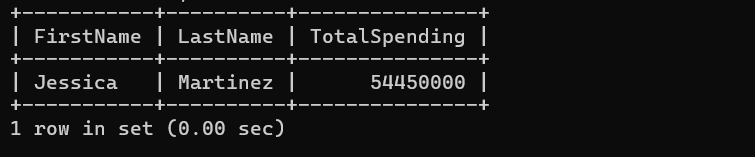
JOIN Products ON OrderDetails.ProductID = Products.ProductID

WHERE Products.ProductName ='Camera Drone'

GROUP BY Customers.CustomerID

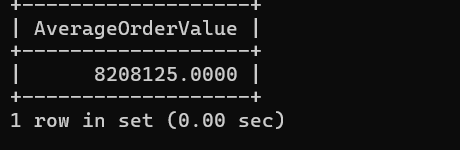
ORDER BY TotalSpending DESC

LIMIT 1;



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

Query:



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

Query: SELECT Customers.FirstName, Customers.LastName, COUNT(Orders.OrderID) AS OrderCount FROM Customers JOIN Orders ON Customers.CustomerID = Orders.CustomerID GROUP BY Customers.CustomerID;

