

# Assignment 1: TechShop, an electronic gadgets shop.

-Sarthak Niranjana Kulkarni (Maverick)

- [sarthakkul2311@gmail.com](mailto:sarthakkul2311@gmail.com)

- (+91) 93256 02791

## **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.**

CREATE TABLE Customers (

    CustomerId VARCHAR(3) NOT NULL PRIMARY KEY,

    FirstName VARCHAR(15) NOT NULL,

    LastName VARCHAR(15) NOT NULL,

    email VARCHAR(30),

    Phone VARCHAR(15),

    Address VARCHAR(15)

);

select \* from Customers;

create table Products (

    ProductId VARCHAR(3) NOT NULL PRIMARY KEY,

    ProductName varchar(30) not null,

    Description varchar(30),

    Price money

);

select \* from Products;

create table orders (

```

        OrderId int not null primary key,

        CustomerId VARCHAR(3) NOT NULL,

        OrderDate DATE,

        TotalAmount MONEY,

        CONSTRAINT FK_CustomerOrder FOREIGN KEY (CustomerId) REFERENCES
Customers(CustomerId)

);

select * from orders;

create table OrderDetails(

        OrderDetailId int not null primary key,

        OrderId int not null,

        ProductId VARCHAR(3) NOT NULL,

        Quantity bigint,

        CONSTRAINT FK_order_OrderDetails FOREIGN KEY (OrderId) REFERENCES
orders(OrderId),

        CONSTRAINT FK_Product_OrderDetails FOREIGN KEY (ProductId) REFERENCES
Products(ProductId)

);

select * from OrderDetails;

create table Inventory(

        InventoryId int not null primary key,

        ProductId varchar(3) not null,

        QuantityInStock int not null,

        LastStockUpdate Date,

        constraint FK_Product_Inventory Foreign key (ProductId) references
Products(ProductId)

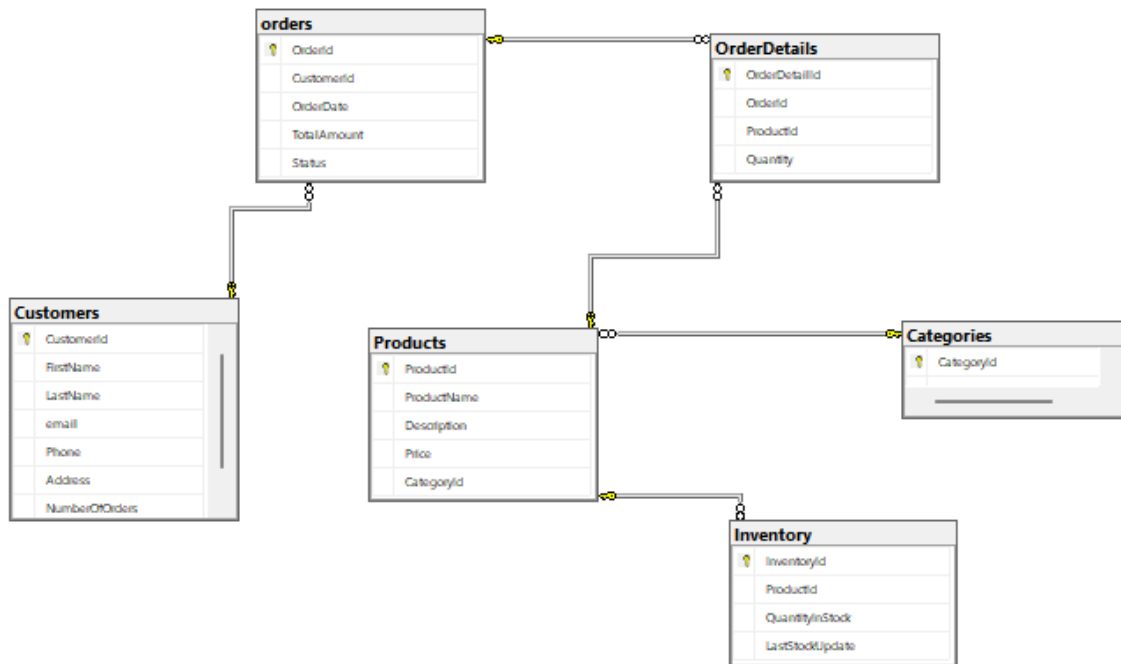
```

);

select \* from Inventory;

---

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



### 4. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

→ CREATE TABLE Customers (

CustomerId VARCHAR(3) NOT NULL PRIMARY KEY,

FirstName VARCHAR(15) NOT NULL,

LastName VARCHAR(15) NOT NULL,

email VARCHAR(30),

Phone VARCHAR(15),

Address VARCHAR(15)

);

select \* from Customers;

create table Products (

```

        ProductId VARCHAR(3) NOT NULL PRIMARY KEY,
        ProductName varchar(30) not null,
        Description varchar(30),
        Price money
    );

select * from Products;

create table orders (
    OrderId int not null primary key,
    CustomerId VARCHAR(3) NOT NULL,
    OrderDate DATE,
    TotalAmount MONEY,
    CONSTRAINT FK_CustomerOrder FOREIGN KEY (CustomerId) REFERENCES
Customers(CustomerId)
);

select * from orders;

create table OrderDetails(
    OrderDetailId int not null primary key,
    OrderId int not null,
    ProductId VARCHAR(3) NOT NULL,
    Quantity bigint,
    CONSTRAINT FK_order_OrderDetails FOREIGN KEY (OrderId) REFERENCES
orders(OrderId),
    CONSTRAINT FK_Product_OrderDetails FOREIGN KEY (ProductId) REFERENCES
Products(ProductId)
);

select * from OrderDetails;

```

```
create table Inventory(  
    InventoryId int not null primary key,  
    ProductId varchar(3) not null,  
    QuantityInStock int not null,  
    LastStockUpdate Date,  
    constraint FK_Product_Inventory Foreign key (ProductId) references  
    Products(ProductId)  
);
```

---

**5. Insert at least 10 sample records into each of the following tables. a. Customers b. Products c. Orders d. OrderDetails e. Inventory**

→ INSERT INTO Customers (CustomerId, FirstName, LastName, email, Phone, Address)  
VALUES

('C01', 'John', 'Doe', 'john.doe@example.com', '123-456-7890', '123 Elm St'),  
( 'C02', 'Jane', 'Smith', 'jane.smith@example.com', '123-555-7891', '456 Oak St'),  
( 'C03', 'Alice', 'Johnson', 'alice.johnson@example.com', '123-555-7892', '789 Pine St'),  
( 'C04', 'Bob', 'Brown', 'bob.brown@example.com', '123-555-7893', '101 Maple St'),  
( 'C05', 'Charlie', 'Davis', 'charlie.davis@example.com', '123-555-7894', '202 Birch St'),  
( 'C06', 'Diana', 'Martinez', 'diana.martinez@example.com', '123-555-7895', '303 Cedar St'),  
( 'C07', 'Eve', 'Wilson', 'eve.wilson@example.com', '123-555-7896', '404 Spruce St'),  
( 'C08', 'Frank', 'Taylor', 'frank.taylor@example.com', '123-555-7897', '505 Fir St'),  
( 'C09', 'Grace', 'Anderson', 'grace.anderson@example.com', '123-555-7898', '606 Redwood St'),  
( 'C10', 'Hannah', 'Thomas', 'hannah.thomas@example.com', '123-555-7899', '707 Willow St');

100 %						
Results Messages						
	CustomerId	FirstName	LastName	email	Phone	Address
1	C01	John	Doe	john.doe@example.com	123-456-7890	123 Elm St
2	C02	Jane	Smith	jane.smith@example.com	123-555-7891	456 Oak St
3	C03	Alice	Johnson	alice.johnson@example.com	123-555-7892	789 Pine St
4	C04	Bob	Brown	bob.brown@example.com	123-555-7893	101 Maple St
5	C05	Charlie	Davis	charlie.davis@example.com	123-555-7894	202 Birch St
6	C06	Diana	Martinez	diana.martinez@example.com	123-555-7895	303 Cedar St
7	C07	Eve	Wilson	eve.wilson@example.com	123-555-7896	404 Spruce St
8	C08	Frank	Taylor	frank.taylor@example.com	123-555-7897	505 Fir St
9	C09	Grace	Anderson	grace.anderson@example.com	123-555-7898	606 Redwood St
10	C10	Hannah	Thomas	hannah.thomas@example.com	123-555-7899	707 Willow St

INSERT INTO Products (ProductId, ProductName, Description, Price) VALUES

('P01', 'Laptop', '15-inch laptop with 16GB RAM', 999.99),

('P02', 'Smartphone', 'Latest with 128GB storage', 699.99),

('P03', 'Headphones', 'Noise-cancelling headphones', 199.99),

('P04', 'Monitor', '24-inch full HD monitor', 149.99),

('P05', 'Keyboard', 'Mechanical keyboard', 89.99),

('P06', 'Mouse', 'Wireless mouse', 49.99),

('P07', 'Printer', 'All-in-one printer', 129.99),

('P08', 'External HDD', '1TB external hard drive', 89.99),

('P09', 'Webcam', 'HD webcam with microphone', 79.99),

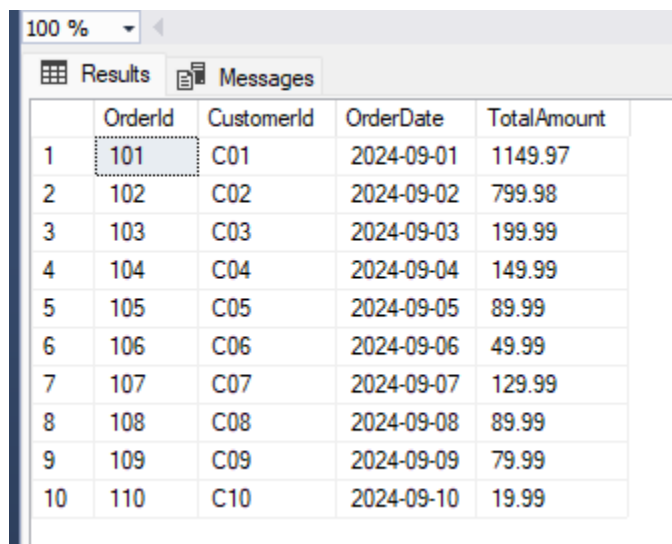
('P10', 'USB Drive', '32GB USB flash drive', 19.99);

100 %				
Results Messages				
	ProductId	ProductName	Description	Price
1	P01	Laptop	15-inch laptop with 16GB RAM	999.99
2	P02	Smartphone	Latest with 128GB storage	699.99
3	P03	Headphones	Noise-cancelling headphones	199.99
4	P04	Monitor	24-inch full HD monitor	149.99
5	P05	Keyboard	Mechanical keyboard	89.99
6	P06	Mouse	Wireless mouse	49.99
7	P07	Printer	All-in-one printer	129.99
8	P08	External HDD	1TB external hard drive	89.99
9	P09	Webcam	HD webcam with microphone	79.99
10	P10	USB Drive	32GB USB flash drive	19.99

```

INSERT INTO orders (OrderId, CustomerId, OrderDate, TotalAmount) VALUES
(101, 'C01', '2024-09-01', 1149.97),
(102, 'C02', '2024-09-02', 799.98),
(103, 'C03', '2024-09-03', 199.99),
(104, 'C04', '2024-09-04', 149.99),
(105, 'C05', '2024-09-05', 89.99),
(106, 'C06', '2024-09-06', 49.99),
(107, 'C07', '2024-09-07', 129.99),
(108, 'C08', '2024-09-08', 89.99),
(109, 'C09', '2024-09-09', 79.99),
(110, 'C10', '2024-09-10', 19.99);

```



The screenshot shows a database query results window with a zoom level of 100%. The window has two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 5 columns: an index column, OrderId, CustomerId, OrderDate, and TotalAmount. The table contains 10 rows of data, corresponding to the SQL statement above. The first row (index 1) is highlighted with a blue selection bar.

	OrderId	CustomerId	OrderDate	TotalAmount
1	101	C01	2024-09-01	1149.97
2	102	C02	2024-09-02	799.98
3	103	C03	2024-09-03	199.99
4	104	C04	2024-09-04	149.99
5	105	C05	2024-09-05	89.99
6	106	C06	2024-09-06	49.99
7	107	C07	2024-09-07	129.99
8	108	C08	2024-09-08	89.99
9	109	C09	2024-09-09	79.99
10	110	C10	2024-09-10	19.99

```

INSERT INTO OrderDetails (OrderDetailId, OrderId, ProductId, Quantity) VALUES
(201, 101, 'P01', 1),
(202, 101, 'P03', 1),
(203, 101, 'P08', 1),
(204, 102, 'P02', 1),
(205, 102, 'P07', 1),
(206, 103, 'P03', 1),

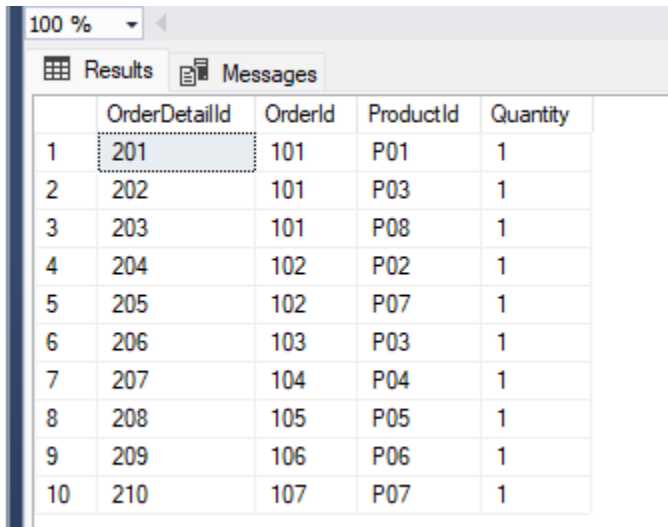
```

(207, 104, 'P04', 1),

(208, 105, 'P05', 1),

(209, 106, 'P06', 1),

(210, 107, 'P07', 1);



The screenshot shows a SQL query results window with a zoom level of 100%. The window has two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 10 rows and 5 columns. The columns are 'OrderDetailId', 'OrderId', 'ProductId', and 'Quantity'. The first row is highlighted with a dashed border.

	OrderDetailId	OrderId	ProductId	Quantity
1	201	101	P01	1
2	202	101	P03	1
3	203	101	P08	1
4	204	102	P02	1
5	205	102	P07	1
6	206	103	P03	1
7	207	104	P04	1
8	208	105	P05	1
9	209	106	P06	1
10	210	107	P07	1

INSERT INTO Inventory (InventoryId, ProductId, QuantityInStock, LastStockUpdate) VALUES

(301, 'P01', 50, '2024-09-01'),

(302, 'P02', 75, '2024-09-02'),

(303, 'P03', 100, '2024-09-03'),

(304, 'P04', 40, '2024-09-04'),

(305, 'P05', 60, '2024-09-05'),

(306, 'P06', 80, '2024-09-06'),

(307, 'P07', 30, '2024-09-07'),

(308, 'P08', 25, '2024-09-08'),

(309, 'P09', 15, '2024-09-09'),

(310, 'P10', 120, '2024-09-10');



	InventoryId	ProductId	QuantityInStock	LastStockUpdate
1	301	P01	50	2024-09-01
2	302	P02	75	2024-09-02
3	303	P03	100	2024-09-03
4	304	P04	40	2024-09-04
5	305	P05	60	2024-09-05
6	306	P06	80	2024-09-06
7	307	P07	30	2024-09-07
8	308	P08	25	2024-09-08
9	309	P09	15	2024-09-09
10	310	P10	120	2024-09-10

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

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

→ Select FirstName, LastName, email

from Customers

order by FirstName;

	FirstName	LastName	email
1	Alice	Johnson	alice.johnson@example.com
2	Bob	Brown	bob.brown@example.com
3	Charlie	Davis	charlie.davis@example.com
4	Diana	Martinez	diana.martinez@example.com
5	Eve	Wilson	eve.wilson@example.com
6	Frank	Taylor	frank.taylor@example.com
7	Grace	Anderson	grace.anderson@example.com
8	Hannah	Thomas	hannah.thomas@example.com
9	Jane	Smith	jane.smith@example.com
10	John	Doe	john.doe@example.com

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

→ SELECT o.OrderId, o.OrderDate, c.FirstName, c.LastName

FROM Orders o

JOIN Customers c ON o.CustomerId = c.CustomerId;

100 %					
		Results	Messages		
	OrderId	OrderDate	FirstName	LastName	
1	101	2024-09-01	John	Doe	
2	102	2024-09-02	Jane	Smith	
3	103	2024-09-03	Alice	Johnson	
4	104	2024-09-04	Bob	Brown	
5	105	2024-09-05	Charlie	Davis	
6	106	2024-09-06	Diana	Martinez	
7	107	2024-09-07	Eve	Wilson	
8	108	2024-09-08	Frank	Taylor	
9	109	2024-09-09	Grace	Anderson	
10	110	2024-09-10	Hannah	Thomas	

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 values('C11','Harry','Potter','harrypotter@example.com','321-456-7809','345 Palm St');

		Results	Messages			
	CustomerId	FirstName	LastName	email	Phone	Address
1	C01	John	Doe	john.doe@example.com	123-456-7890	123 Elm St
2	C02	Jane	Smith	jane.smith@example.com	123-555-7891	456 Oak St
3	C03	Alice	Johnson	alice.johnson@example.com	123-555-7892	789 Pine St
4	C04	Bob	Brown	bob.brown@example.com	123-555-7893	101 Maple St
5	C05	Charlie	Davis	charlie.davis@example.com	123-555-7894	202 Birch St
6	C06	Diana	Martinez	diana.martinez@example.com	123-555-7895	303 Cedar St
7	C07	Eve	Wilson	eve.wilson@example.com	123-555-7896	404 Spruce St
8	C08	Frank	Taylor	frank.taylor@example.com	123-555-7897	505 Fir St
9	C09	Grace	Anderson	grace.anderson@example.com	123-555-7898	606 Redwood St
10	C10	Hannah	Thomas	hannah.thomas@example.com	123-555-7899	707 Willow St
11	C11	Harry	Potter	harrypotter@example.com	321-456-7809	345 Palm St

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 \* 1.10;

Results		Messages		
	ProductId	ProductName	Description	Price
1	P01	Laptop	15-inch laptop with 16GB RAM	1099.989
2	P02	Smartphone	Latest with 128GB storage	769.989
3	P03	Headphones	Noise-cancelling headphones	219.989
4	P04	Monitor	24-inch full HD monitor	164.989
5	P05	Keyboard	Mechanical keyboard	98.989
6	P06	Mouse	Wireless mouse	54.989
7	P07	Printer	All-in-one printer	142.989
8	P08	External HDD	1TB external hard drive	98.989
9	P09	Webcam	HD webcam with microphone	87.989
10	P10	USB Drive	32GB USB flash drive	21.989

**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.**

→ DECLARE @OrderId INT = 107;

DELETE FROM OrderDetails

WHERE OrderId = @OrderId;

DELETE FROM orders

WHERE OrderId = @OrderId;

100 %

Results

Messages

	OrderId	CustomerId	OrderDate	TotalAmount
1	101	C01	2024-09-01	1149.97
2	102	C02	2024-09-02	799.98
3	103	C03	2024-09-03	199.99
4	104	C04	2024-09-04	149.99
5	105	C05	2024-09-05	89.99
6	106	C06	2024-09-06	49.99
7	108	C08	2024-09-08	89.99
8	109	C09	2024-09-09	79.99

	OrderDetailId	OrderId	ProductId	Quantity
1	201	101	P01	1
2	202	101	P03	1
3	203	101	P08	1
4	204	102	P02	1
5	205	102	P07	1
6	206	103	P03	1
7	207	104	P04	1
8	208	105	P05	1
9	209	106	P06	1

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 Customers (CustomerId, FirstName, LastName, email, Phone, Address)  
VALUES ('C12', 'John', 'wick', 'john.wick@example.com', '2135468790', '619 Raw St');

insert into orders(OrderId,CustomerId, OrderDate, TotalAmount)  
values(112, 'C12', '2024-10-12', 250.75);

100 %						
Results Messages						
	CustomerId	FirstName	LastName	email	Phone	Address
1	C01	John	Doe	john.doe@example.com	123-456-7890	123 Elm St
2	C02	Jane	Smith	jane.smith@example.com	123-555-7891	456 Oak St
3	C03	Alice	Johnson	alice.johnson@example.com	123-555-7892	789 Pine St
4	C04	Bob	Brown	bob.brown@example.com	123-555-7893	101 Maple St
5	C05	Charlie	Davis	charlie.davis@example.com	123-555-7894	202 Birch St
6	C06	Diana	Martinez	diana.martinez@example.com	123-555-7895	303 Cedar St
7	C07	Eve	Wilson	eve.wilson@example.com	123-555-7896	404 Spruce St
8	C08	Frank	Taylor	frank.taylor@example.com	123-555-7897	505 Fir St

	OrderId	CustomerId	OrderDate	TotalAmount
1	101	C01	2024-09-01	1149.97
2	102	C02	2024-09-02	799.98
3	103	C03	2024-09-03	199.99
4	104	C04	2024-09-04	149.99
5	105	C05	2024-09-05	89.99
6	106	C06	2024-09-06	49.99
7	108	C08	2024-09-08	89.99
8	109	C09	2024-09-09	79.99
9	110	C10	2024-09-10	19.99
10	112	C12	2024-10-12	250.75

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.

→ declare @custid varchar(3) = 'C01';

update Customers

set email = 'doe.john@example.com', Address = '123 Mle St'

where CustomerId = @custid;

select \* from Customers;

100 %						
Results Messages						
	CustomerId	FirstName	LastName	email	Phone	Address
1	C01	John	Doe	doe.john@example.com	123-456-7890	123 Mle St
2	C02	Jane	Smith	jane.smith@example.com	123-555-7891	456 Oak St
3	C03	Alice	Johnson	alice.johnson@example.com	123-555-7892	789 Pine St
4	C04	Bob	Brown	bob.brown@example.com	123-555-7893	101 Maple St
5	C05	Charlie	Davis	charlie.davis@example.com	123-555-7894	202 Birch St
6	C06	Diana	Martinez	diana.martinez@example.com	123-555-7895	303 Cedar St
7	C07	Eve	Wilson	eve.wilson@example.com	123-555-7896	404 Spruce St
8	C08	Frank	Taylor	frank.taylor@example.com	123-555-7897	505 Fir St
9	C09	Grace	Anderson	grace.anderson@example.c...	123-555-7898	606 Redwo...
10	C10	Hannah	Thomas	hannah.thomas@example.c...	123-555-7899	707 Willow St
11	C11	Harry	Potter	harrypotter@example.com	321-456-7809	345 Palm St
12	C12	John	wick	john.wick@example.com	2135468790	619 Raw St

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

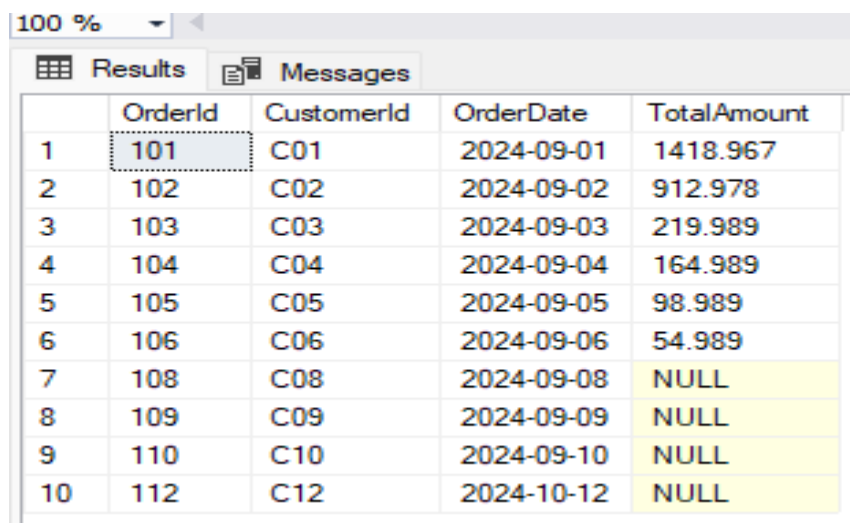
SET TotalAmount = (

SELECT SUM(Products.Price \* OrderDetails.Quantity)

FROM Products, OrderDetails

WHERE OrderDetails.ProductID = Products.ProductID

AND OrderDetails.OrderID = Orders.OrderID);



	OrderId	CustomerId	OrderDate	TotalAmount
1	101	C01	2024-09-01	1418.967
2	102	C02	2024-09-02	912.978
3	103	C03	2024-09-03	219.989
4	104	C04	2024-09-04	164.989
5	105	C05	2024-09-05	98.989
6	106	C06	2024-09-06	54.989
7	108	C08	2024-09-08	NULL
8	109	C09	2024-09-09	NULL
9	110	C10	2024-09-10	NULL
10	112	C12	2024-10-12	NULL

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.

→ DECLARE @CustomerId VARCHAR(3) = 'C08';

DELETE FROM OrderDetails

WHERE OrderId IN (

SELECT OrderId

FROM orders

WHERE CustomerId = @CustomerId

);

DELETE FROM orders

WHERE CustomerId = @CustomerId;

(1 row affected)

Completion time: 2024-09-23T12:56:48.5978996+05:30

**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 (ProductId, ProductName, Description, Price)

VALUES ('P12','Camera','Hd camera with 256GB', 899.99);

select \* from Products;

100 %

	ProductId	ProductName	Description	Price
1	P01	Laptop	15-inch laptop with 16GB RAM	1099.989
2	P02	Smartphone	Latest with 128GB storage	769.989
3	P03	Headphones	Noise-cancelling headphones	219.989
4	P04	Monitor	24-inch full HD monitor	164.989
5	P05	Keyboard	Mechanical keyboard	98.989
6	P06	Mouse	Wireless mouse	54.989
7	P07	Printer	All-in-one printer	142.989
8	P08	External HDD	1TB external hard drive	98.989
9	P09	Webcam	HD webcam with microphone	87.989
10	P10	USB Drive	32GB USB flash drive	21.989
11	P12	Camera	Hd camera with 256GB	899.99

**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(20);

DECLARE @OrderId INT;

DECLARE @NewStatus VARCHAR(20);

```
SET @OrderId = 102;
```

```
SET @NewStatus = 'Shipped';
```

```
UPDATE Orders
```

```
SET Status = @NewStatus
```

```
WHERE OrderId = @OrderId;
```

```
SELECT * FROM Orders;
```

Results		Messages			
	OrderId	CustomerId	OrderDate	TotalAmount	Status
1	101	C01	2024-09-01	1418.967	NULL
2	102	C02	2024-09-02	912.978	Shipped
3	103	C03	2024-09-03	219.989	NULL
4	104	C04	2024-09-04	164.989	NULL
5	105	C05	2024-09-05	98.989	NULL
6	106	C06	2024-09-06	54.989	NULL
7	109	C09	2024-09-09	NULL	NULL
8	110	C10	2024-09-10	NULL	NULL
9	112	C12	2024-10-12	NULL	NULL

**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 DEFAULT 0;
```

```
UPDATE Customers
```

```
SET NumberOfOrders = COALESCE(OrderCounts.NumberOfOrders, 0)
```

```
FROM Customers
```

```
LEFT JOIN (
```

```
    SELECT CustomerId, COUNT(*) AS NumberOfOrders
```

```
    FROM orders
```

```
    GROUP BY CustomerId
```



) AS OrderCounts

ON Customers.CustomerId = OrderCounts.CustomerId;

select \* from Customers;

Results

Messages

	CustomerId	FirstName	LastName	email	Phone	Address	NumberOfOrders
1	C01	John	Doe	doe.john@example.com	123-456-7890	123 Mle St	1
2	C02	Jane	Smith	jane.smith@example.com	123-555-7891	456 Oak St	1
3	C03	Alice	Johnson	alice.johnson@example.com	123-555-7892	789 Pine St	1
4	C04	Bob	Brown	bob.brown@example.com	123-555-7893	101 Maple St	1
5	C05	Charlie	Davis	charlie.davis@example.com	123-555-7894	202 Birch St	1
6	C06	Diana	Martinez	diana.martinez@example.com	123-555-7895	303 Cedar St	1
7	C07	Eve	Wilson	eve.wilson@example.com	123-555-7896	404 Spruce St	0
8	C08	Frank	Taylor	frank.taylor@example.com	123-555-7897	505 Fir St	0
9	C09	Grace	Anderson	grace.anderson@example.com	123-555-7898	606 Redwood St	1
10	C10	Hannah	Thomas	hannah.thomas@example.com	123-555-7899	707 Willow St	1
11	C11	Harry	Potter	harrypotter@example.com	321-456-7809	345 Palm St	0
12	C12	John	wick	john.wick@example.com	2135468790	619 Raw St	1

### **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

orders.OrderId,  
orders.OrderDate,  
orders.TotalAmount,  
Customers.FirstName,  
Customers.LastName,  
Customers.Email,  
Customers.Phone

FROM orders

JOIN Customers ON Orders.CustomerId = Customers.CustomerId;

	OrderId	OrderDate	TotalAmount	FirstName	LastName	Email	Phone
1	101	2024-09-01	1418.967	John	Doe	doe.john@example.com	123-456-7890
2	102	2024-09-02	912.978	Jane	Smith	jane.smith@example.com	123-555-7891
3	103	2024-09-03	219.989	Alice	Johnson	alice.johnson@example.com	123-555-7892
4	104	2024-09-04	164.989	Bob	Brown	bob.brown@example.com	123-555-7893
5	105	2024-09-05	98.989	Charlie	Davis	charlie.davis@example.com	123-555-7894
6	106	2024-09-06	54.989	Diana	Martinez	diana.martinez@example.com	123-555-7895
7	109	2024-09-09	NULL	Grace	Anderson	grace.anderson@example.com	123-555-7898
8	110	2024-09-10	NULL	Hannah	Thomas	hannah.thomas@example.com	123-555-7899
9	112	2024-10-12	NULL	John	wick	john.wick@example.com	2135468790

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

	ProductName	TotalRevenue
1	External HDD	98.989
2	Headphones	439.978
3	Keyboard	98.989
4	Laptop	1099.989
5	Monitor	164.989
6	Mouse	54.989
7	Printer	142.989
8	Smartphone	769.989

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

```
→ select distinct
      Customers.FirstName,
      Customers.LastName,
      Customers.Phone
```

from Customers

join orders on customers.CustomerId = orders.CustomerId;

Results		Messages	
	FirstName	LastName	Phone
1	Alice	Johnson	123-555-7892
2	Bob	Brown	123-555-7893
3	Charlie	Davis	123-555-7894
4	Diana	Martinez	123-555-7895
5	Grace	Anderson	123-555-7898
6	Hannah	Thomas	123-555-7899
7	Jane	Smith	123-555-7891
8	John	Doe	123-456-7890
9	John	wick	2135468790

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

Products.ProductName,

SUM(OrderDetails.Quantity) AS TotalQuantityOrdered

FROM Products

JOIN OrderDetails ON Products.ProductId = OrderDetails.ProductId

GROUP BY Products.ProductName

ORDER BY TotalQuantityOrdered DESC

OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY;

Results		Messages	
	ProductName	TotalQuantityOrdered	
1	Headphones	2	

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

→ SELECT Products.ProductName, Categories.CategoryName

FROM Products

JOIN Categories ON Products.CategoryId = Categories.CategoryId;

	ProductName	CategoryName
1	Laptop	Laptops
2	Smartphone	Smartphones
3	Headphones	Accessories
4	Monitor	Accessories
5	Keyboard	Accessories
6	Mouse	Accessories
7	Printer	Printers
8	External HDD	Storage Devices
9	Webcam	Accessories
10	USB Drive	Storage Devices

**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 Customers.FirstName,  
        Customers.LastName,  
        AVG(Orders.TotalAmount) AS AverageOrderValue  
FROM Customers
```

JOIN Orders ON Customers.CustomerId = Orders.CustomerId

```
GROUP BY Customers.FirstName,  
        Customers.LastName;
```

	FirstName	LastName	AverageOrderValue
1	Grace	Anderson	NULL
2	Bob	Brown	164.989
3	Charlie	Davis	98.989
4	John	Doe	1418.967
5	Alice	Johnson	219.989
6	Diana	Martinez	54.989
7	Jane	Smith	912.978
8	Hannah	Thomas	NULL
9	John	wick	NULL

**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 orders.OrderId,
```

```

orders.TotalAmount AS TotalRevenue,
Customers.FirstName,
Customers.LastName,
Customers.email,
Customers.Phone
FROM orders
JOIN Customers ON Orders.CustomerId = Customers.CustomerId
ORDER BY orders.TotalAmount DESC
OFFSET 0 ROWS FETCH NEXT 1 ROWS ONLY;

```

Results		Messages				
	OrderId	TotalRevenue	FirstName	LastName	email	Phone
1	101	1418.967	John	Doe	doe.john@example.com	123-456-7890

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

```

→ select Products.ProductName,
    count(OrderDetails.OrderId) as 'Times Ordered'
from Products
join OrderDetails on Products.ProductId = OrderDetails.ProductId
group by Products.ProductName
order by 'Times Ordered' desc;

```

Results		Messages	
	ProductName	Times Ordered	
1	Headphones	2	
2	Keyboard	1	
3	Laptop	1	
4	Monitor	1	
5	Mouse	1	
6	Printer	1	
7	Smartphone	1	
8	External HDD	1	

**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.**

→ declare @Productname varchar(15) = 'Headphones';

select Customers.FirstName,

Customers.LastName,

Customers.email,

Customers.Phone

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;

Results		Messages		
	FirstName	LastName	email	Phone
1	John	Doe	doe.john@example.com	123-456-7890
2	Alice	Johnson	alice.johnson@example.com	123-555-7892

**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.**

→ declare @start date = '2024-09-01';

declare @end date = '2024-09-30';

select sum(TotalAmount) as Total\_Revenue

from orders

where OrderDate between @start and @end;

Results		Messages		
	Total_Revenue			
1	2870.901			

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

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

→ SELECT Customers.CustomerId,

Customers.FirstName,

Customers.LastName,

Customers.Email

FROM Customers

Left JOIN orders ON Customers.CustomerId = orders.CustomerId

WHERE orders.OrderId IS NULL;

Results		Messages		
	CustomerId	FirstName	LastName	Email
1	C07	Eve	Wilson	eve.wilson@example.com
2	C08	Frank	Taylor	frank.taylor@example.com
3	C11	Harry	Potter	harrypotter@example.com

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

→ select count(\*) as 'Available Products' from products

Join Inventory on Products.ProductId = Inventory.ProductId

where Inventory.QuantityInStock >= 1 ;

Results		Messages		
	Available Products			
1	10			

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

→ select sum(TotalAmount) as 'Total Revenue'

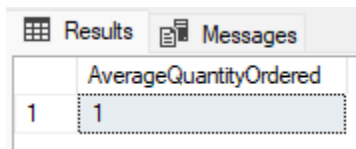
from orders;

Results		Messages		
	Total Revenue			
1	2870.901			

**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.**

→ DECLARE @CategoryName VARCHAR(15) = 'Printers';

```
SELECT AVG(od.Quantity) AS AverageQuantityOrdered
FROM OrderDetails od
JOIN Products p ON od.ProductId = p.ProductId
JOIN Categories c ON p.CategoryId = c.CategoryId
WHERE c.CategoryName = @CategoryName;
```



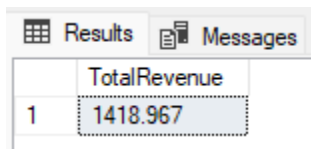
The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a single row of data. The column header is 'AverageQuantityOrdered' and the value in the row is '1'.

	AverageQuantityOrdered
1	1

**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.**

→ DECLARE @CustomerId VARCHAR(3) = 'C01';

```
SELECT SUM(o.TotalAmount) AS TotalRevenue
FROM orders o
WHERE o.CustomerId = @CustomerId;
```



The screenshot shows a SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a single row of data. The column header is 'TotalRevenue' and the value in the row is '1418.967'.

	TotalRevenue
1	1418.967

**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 'Number Of Orders'
FROM Customers c
JOIN orders o ON c.CustomerId = o.CustomerId
GROUP BY c.FirstName, c.LastName
```



ORDER BY 'Number Of Orders' DESC;

Results		Messages	
	FirstName	LastName	Number Of Orders
1	Grace	Anderson	1
2	Bob	Brown	1
3	Charlie	Davis	1
4	John	Doe	1
5	Alice	Johnson	1
6	Diana	Martinez	1
7	Jane	Smith	1
8	Hannah	Thomas	1
9	John	wick	1

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.

```
→ SELECT c.CategoryName,  
       SUM(od.Quantity) AS TotalQuantityOrdered  
FROM OrderDetails od  
JOIN Products p ON od.ProductId = p.ProductId  
JOIN Categories c ON p.CategoryId = c.CategoryId  
GROUP BY c.CategoryName  
ORDER BY TotalQuantityOrdered DESC  
OFFSET 0 ROWS FETCH NEXT 1 ROW ONLY;
```

Results		Messages	
	CategoryName	TotalQuantityOrdered	
1	Accessories	5	

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(o.TotalAmount) 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
OFFSET 0 ROWS FETCH NEXT 1 ROW ONLY;

```

Results		Messages	
	FirstName	LastName	TotalSpending
1	John	Doe	4256.901

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

→ SELECT AVG(TotalRevenue) AS AverageOrderValue

FROM (

SELECT c.CustomerId,

SUM(o.TotalAmount) AS TotalRevenue

FROM Customers c

LEFT JOIN orders o ON c.CustomerId = o.CustomerId

GROUP BY c.CustomerId

) AS CustomerTotals;

Results		Messages	
	AverageOrderValue		
1	478.4835		

**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.FirstName,

c.LastName,

COUNT(o.OrderId) AS TotalOrders

FROM Customers c

LEFT JOIN orders o ON c.CustomerId = o.CustomerId

GROUP BY c.FirstName, c.LastName;

Results Messages			
	FirstName	LastName	TotalOrders
1	Grace	Anderson	1
2	Bob	Brown	1
3	Charlie	Davis	1
4	John	Doe	1
5	Alice	Johnson	1
6	Diana	Martinez	1
7	Harry	Potter	0
8	Jane	Smith	1
9	Frank	Taylor	0
10	Hannah	Thomas	1
11	John	wick	1
12	Eve	Wilson	0