

Foundation Technical Training

Assignment No. 1 - SQL - Task 2

Name: Niranjan Kolpe, Batch: C#-Batch 2

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

1. Write an SQL query to retrieve the names and emails of all customers.
2. Write an SQL query to list all orders with their order dates and corresponding customer names.
3. Write an SQL query to insert a new customer record into the "Customers" table. Include customer information such as name, email, and address.
4. Write an SQL query to update the prices of all electronic gadgets in the "Products" table by increasing them by 10%.
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.
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.
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.
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.
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.
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.
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.
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.

Program Code in SQL:

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

USE NiranjanDB;

SELECT FirstName, LastName, Email FROM Customers;

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

SELECT OrderID, FirstName, LastName, OrderDate FROM Orders, Customers WHERE Orders.CustomerID=Customers.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 VALUES (11, 'Benedict', 'Cumberbatch',  
'benedict@gmail.com', 9999999982, 'Chennai');
```

```
SELECT * FROM Customers;
```

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+(Price*10/100);
```

```
SELECT * FROM Products;
```

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 = 10;
```

--before deletion

```
select * from Orders;
```

```
select * from OrderDetails;
```

--after deletion

```
delete from OrderDetails
```

```
where OrderID=@OrderID;
```

```
delete from Orders
```

```
where OrderID = @OrderID;
```

```
Select * from Orders;
```

```
select * from OrderDetails;
```

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 VALUES (10, 2, '2024-09-19', 1000);
```

```
SELECT * FROM Orders;
```

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 @CustomerID int = 7;
```

```
DECLARE @Email varchar(50) = 'clintbarton@gmail.com';
```

```
SELECT * FROM Customers WHERE CustomerID = @CustomerID;
```

```
UPDATE Customers SET Email = @Email WHERE CustomerID = @CustomerID;
```

```
SELECT * from Customers WHERE CustomerID = @CustomerID;
```

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.

```
SELECT * FROM Orders;
```

```
-- The "TotalAmount" column has already been set according to the "Price" and  
"Quantity" columns,  
  
-- when the order was placed and the record for it was inserted.
```

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.

```
SELECT * FROM Orders;
```

```
SELECT * FROM OrderDetails;
```

```
BEGIN TRANSACTION;
```

```
DECLARE @CustomerID INT = 6;
```

```
DELETE FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders WHERE  
CustomerID = @CustomerID);
```

```
DELETE FROM Orders WHERE CustomerID = @CustomerID;
```

```
COMMIT TRANSACTION;
```

```
SELECT * FROM Orders;
```

```
SELECT * FROM OrderDetails;
```

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.

```
BEGIN TRANSACTION;
```

```
SELECT * FROM Products;
```

```
INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES (11, 'Speaker', 'Bluetooth Speaker', 2000);
```

```
SELECT * FROM Products;
```

```
COMMIT TRANSACTION;
```

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.

```
UPDATE Orders SET Status='Pending';
```

```
SELECT * FROM Orders;
```

```
BEGIN TRANSACTION;
```

```
DECLARE @OrderID INT = 8;
```

```
UPDATE Orders SET Status='Shipped' WHERE OrderID=@OrderID;
```

```
SELECT * FROM Orders;
```

```
COMMIT TRANSACTION;
```

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.

```
BEGIN TRANSACTION;
```

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

```
SELECT * FROM Customers;
```

```
COMMIT TRANSACTION;
```

The screenshot shows a Windows desktop with the following elements:

- SQL Server Enterprise Edition:**
 - Object Explorer:** Shows the database structure for 'DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DES)'. The 'Customers' table is expanded, showing columns: CustomerID (PK, int, not null), FirstName (text, not null), LastName (text, null), Email (varchar(50), null), Phone (decimal(18,0), null), and Address (varchar(100), null).
 - SQL Query Editor:** Contains a query to retrieve customer names and emails from the 'Customers' table in the 'NirnanjanDB' database.


```
-- Write an SQL query to retrieve the names and emails of all customers.
USE NirnanjanDB;
SELECT FirstName, LastName, Email FROM Customers;
```
 - Results:** The query is executed successfully, returning 10 rows of data.

	FirstName	LastName	Email
1	Nirnanjan	Kolpe	nirnanjan@gmail.com
2	Steve	Rogers	steve@gmail.com
3	Tory	Stark	tony@gmail.com
4	Natasha	Romanoff	natasha@gmail.com
5	Bruce	Banner	bruce@gmail.com
6	Thor	Odinson	thor@gmail.com
7	Clint	Barton	clint@gmail.com
8	Wanda	Maximoff	wanda@gmail.com
9	Sam	Wilson	sam@gmail.com
10	Peter	Parker	peter@gmail.com
- Web Browser:** Shows a Google search page for 'Nirnanjan'. The search results are not fully visible, but the search bar contains the text 'Nirnanjan'.
- Taskbar:** Shows the system clock as 7:49 PM on 9/20/2024. The weather is 80°F and Mostly cloudy.

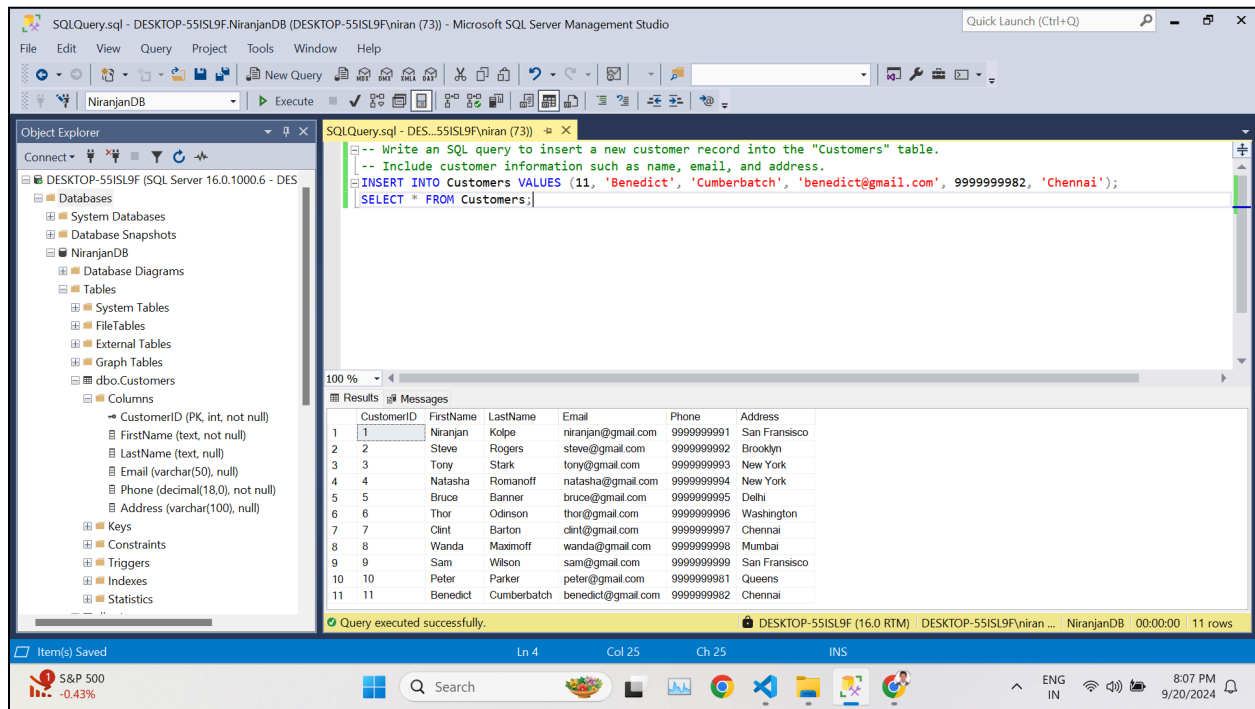
The screenshot shows a Windows desktop with the following elements:

- Taskbar:** Includes a clock showing 8:02 PM on 9/20/2024, system icons for network, volume, and battery, and a search bar.
- Background:** A blue wall with a 'Fall Colors' wallpaper.
- SQL Server Enterprise Edition (Frontground):**
 - Top Bar:** Displays 'SQLQuery.sql - DESKTOP-55ISL9F\NiranjanDB (DESKTOP-55ISL9F\Niranjan (73)) - Microsoft SQL Server Enterprise Edition'.
 - Object Explorer:** Shows the database structure for 'NiranjanDB', including tables like 'OrderDetails', 'Orders', and 'Products'.
 - Query Editor:** Contains a SQL query:


```
-- Write an SQL query to list all orders with their order dates and corresponding customer names.
SELECT OrderID, FirstName, LastName, OrderDate FROM Orders, Customers WHERE Orders.CustomerID=Customers.CustomerID;
```
 - Results Panel:** Shows a table with 10 rows of order data:

	OrderID	FirstName	LastName	OrderDate
1	1	Bruce	Banner	2024-09-01
2	2	Clint	Barton	2024-09-02
3	3	Thor	Odinson	2024-09-02
4	4	Natasha	Romanoff	2024-09-03
5	5	Sam	Wilson	2024-09-04
6	6	Steve	Rogers	2024-09-06
7	7	Niranjan	Kolpe	2024-09-15
8	8	Tony	Stark	2024-09-16
9	9	Wanda	Maximoff	2024-09-17
10	10	Peter	Parker	2024-09-18
 - Status Bar:** Indicates 'Query executed successfully.' and '10 rows'.
- Web Browser (Background):** A Google Chrome window titled 'Quick Launch (Ctrl+Q)' is visible in the top right corner.

Output 3:



SQLQuery.sql - DESKTOP-55ISL9F.NiranjanDB (DESKTOP-55ISL9F\Niran (73)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTOP-55ISL9F) | Databases | NiranjanDB | Tables | Columns

SQLQuery.sql - DESKTOP-55ISL9F\Niran (73) | Execute | Query executed successfully.

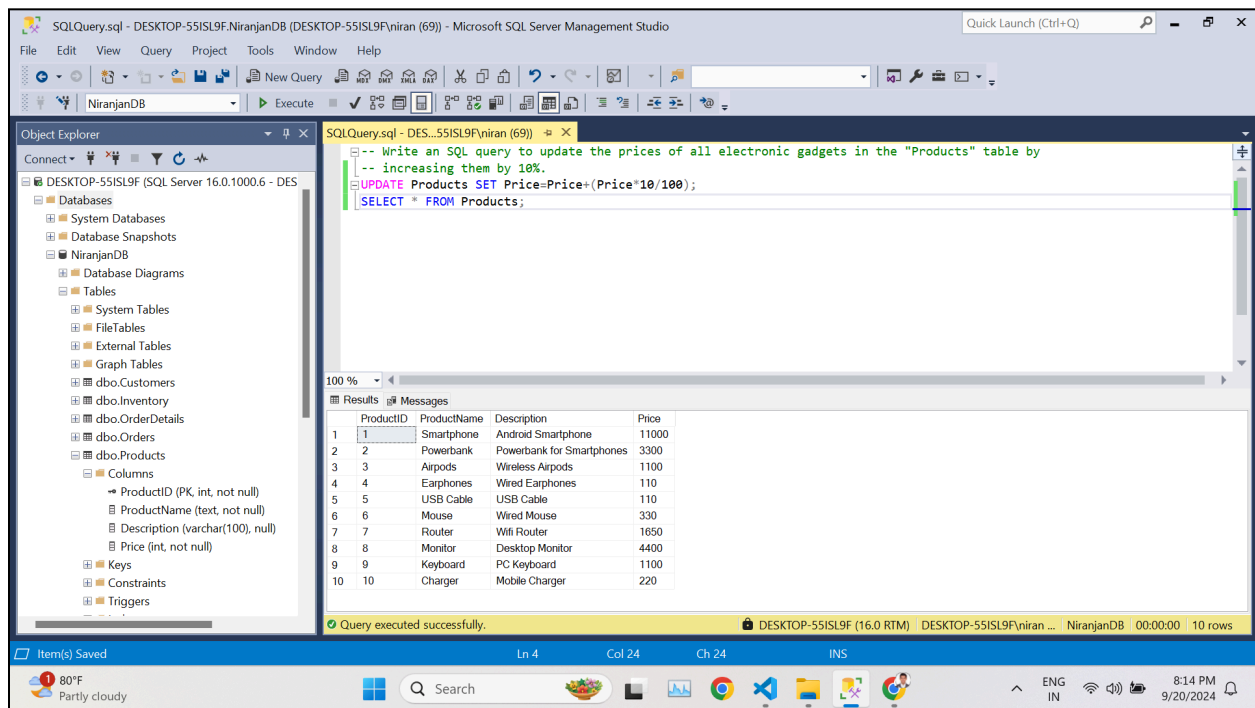
```
-- Write an SQL query to insert a new customer record into the "Customers" table.
-- Include customer information such as name, email, and address.
INSERT INTO Customers VALUES (11, 'Benedict', 'Cumberbatch', 'benedict@gmail.com', 9999999982, 'Chennai');
SELECT * FROM Customers;
```

Results: Messages

CustomerID	FirstName	LastName	Email	Phone	Address
1	Niranjan	Kolpe	niranjan@gmail.com	9999999991	San Francisco
2	Steve	Rogers	steve@gmail.com	9999999992	Brooklyn
3	Tony	Stark	tony@gmail.com	9999999993	New York
4	Natasha	Romanoff	natasha@gmail.com	9999999994	New York
5	Bruce	Banner	bruce@gmail.com	9999999995	Delhi
6	Thor	Odinson	thor@gmail.com	9999999996	Washington
7	Clint	Barton	clint@gmail.com	9999999997	Chennai
8	Wanda	Maximoff	wanda@gmail.com	9999999998	Mumbai
9	Sam	Wilson	sam@gmail.com	9999999999	San Francisco
10	Peter	Parker	peter@gmail.com	9999999981	Queens
11	Benedict	Cumberbatch	benedict@gmail.com	9999999982	Chennai

Item(s) Saved | S&P 500 -0.43% | Search | 8:07 PM 9/20/2024

Output 4:



SQLQuery.sql - DESKTOP-55ISL9F.NiranjanDB (DESKTOP-55ISL9F\Niran (69)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTOP-55ISL9F) | Databases | NiranjanDB | Tables | Columns

SQLQuery.sql - DESKTOP-55ISL9F\Niran (69) | Execute | Query executed successfully.

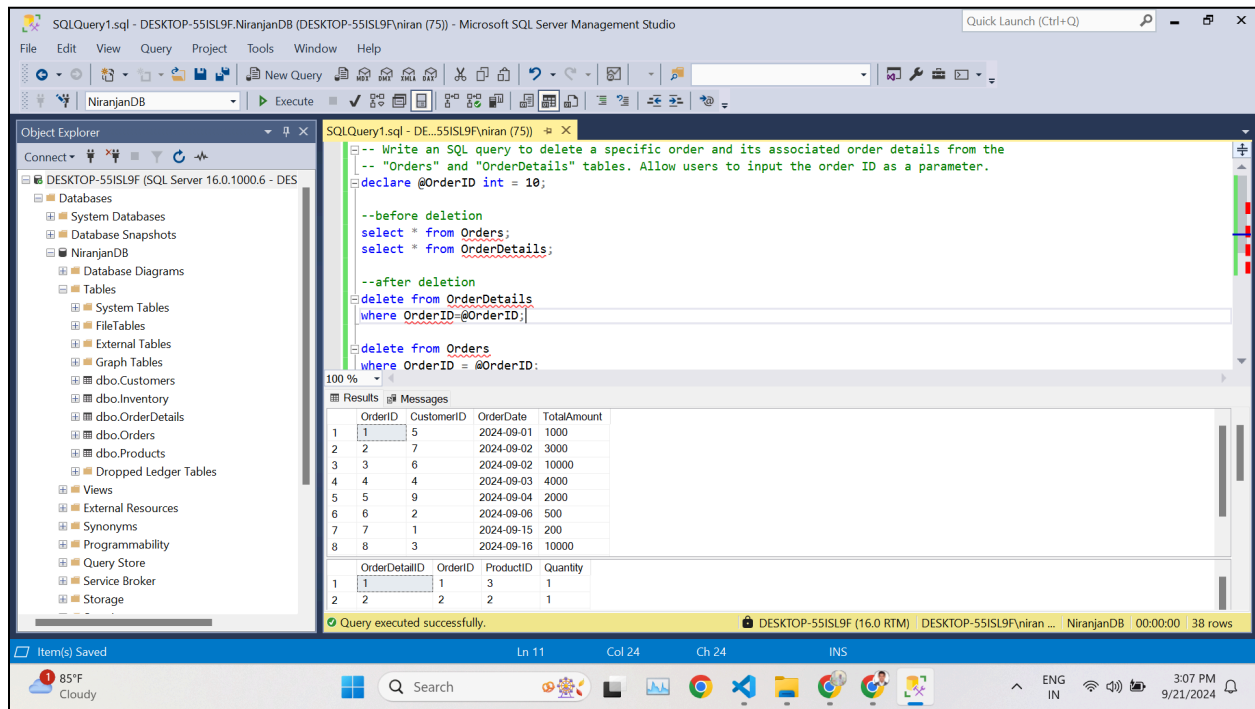
```
-- Write an SQL query to update the prices of all electronic gadgets in the "Products" table by
-- increasing them by 10%.
UPDATE Products SET Price=Price+(Price*10/100);
SELECT * FROM Products;
```

Results: Messages

ProductID	ProductName	Description	Price
1	Smartphone	Android Smartphone	11000
2	Powerbank	Powerbank for Smartphones	3300
3	Airpods	Wireless Airpods	1100
4	Earphones	Wired Earphones	110
5	USB Cable	USB Cable	110
6	Mouse	Wired Mouse	330
7	Router	WiFi Router	1650
8	Monitor	Desktop Monitor	4400
9	Keyboard	PC Keyboard	1100
10	Charger	Mobile Charger	220

Item(s) Saved | 80°F Partly cloudy | Search | 8:14 PM 9/20/2024

Output 5:



SQLQuery1.sql - DESKTOP-55ISL9F.NiranjanDB (DESKTOP-55ISL9F\Niran (75)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to NiranjanDB (SQL Server 16.0.1000.6 - DESKTOP-55ISL9F.NiranjanDB)

SQLQuery1.sql - DESKTOP-55ISL9F\Niran (75)

```
-- 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 = 10;

--before deletion
select * from Orders;
select * from OrderDetails;

--after deletion
delete from OrderDetails
where OrderID=@OrderID;

delete from Orders
where OrderID = @OrderID;
```

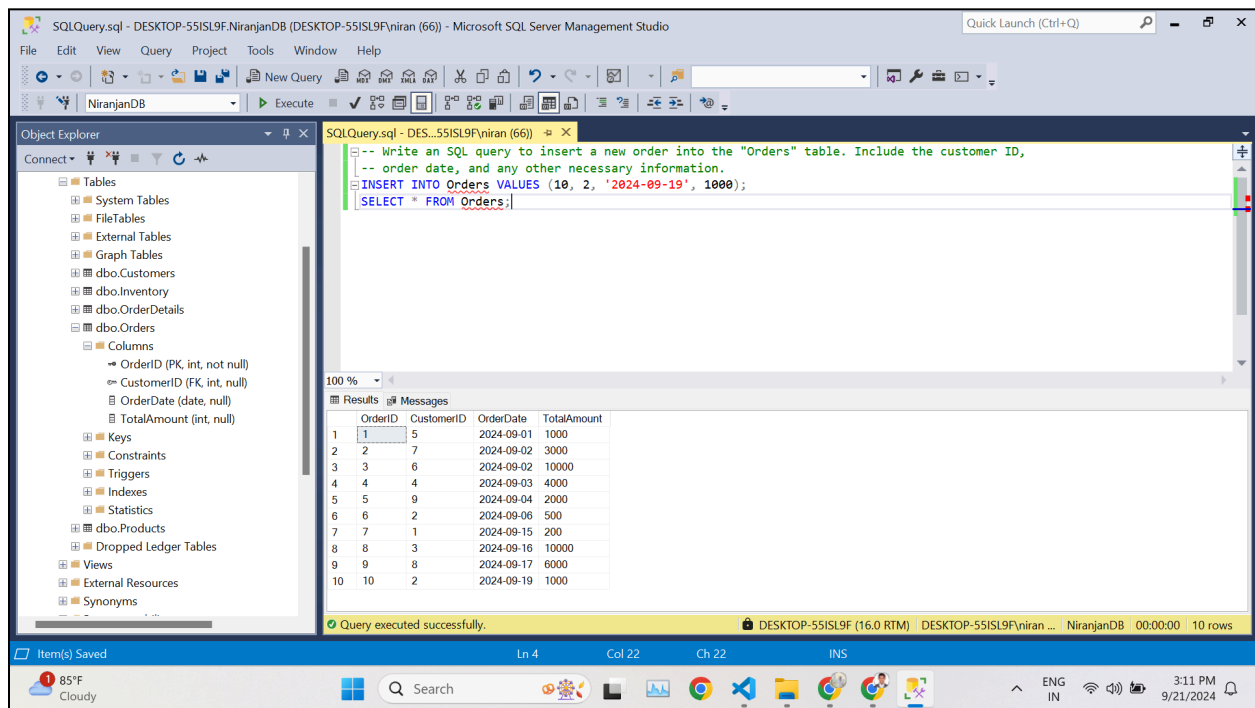
Results: Messages

OrderID	CustomerID	OrderDate	TotalAmount
1	5	2024-09-01	1000
2	7	2024-09-02	3000
3	6	2024-09-02	10000
4	4	2024-09-03	4000
5	9	2024-09-04	2000
6	2	2024-09-06	500
7	1	2024-09-15	200
8	3	2024-09-16	10000

OrderDetailID	OrderID	ProductID	Quantity
1	1	3	1
2	2	2	1

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) DESKTOP-55ISL9F\Niran ... NiranjanDB 00:00:00 38 rows

Output 6:



SQLQuery.sql - DESKTOP-55ISL9F.NiranjanDB (DESKTOP-55ISL9F\Niran (66)) - Microsoft SQL Server Management Studio

Object Explorer: Connect to NiranjanDB (SQL Server 16.0.1000.6 - DESKTOP-55ISL9F.NiranjanDB)

SQLQuery.sql - DESKTOP-55ISL9F\Niran (66)

```
-- 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 VALUES (10, 2, '2024-09-19', 1000);
SELECT * FROM Orders;
```

Results: Messages

OrderID	CustomerID	OrderDate	TotalAmount
1	5	2024-09-01	1000
2	7	2024-09-02	3000
3	6	2024-09-02	10000
4	4	2024-09-03	4000
5	9	2024-09-04	2000
6	2	2024-09-06	500
7	1	2024-09-15	200
8	3	2024-09-16	10000
9	8	2024-09-17	6000
10	2	2024-09-19	1000

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) DESKTOP-55ISL9F\Niran ... NiranjanDB 00:00:00 10 rows

Output 7:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB', including tables like 'Customers'. The main query window contains the following SQL code:

```
-- 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 @CustomerID int = 7;
DECLARE @Email varchar(50) = 'clintbarton@gmail.com';
SELECT * FROM Customers WHERE CustomerID = @CustomerID;
UPDATE Customers SET Email = @Email WHERE CustomerID = @CustomerID;
SELECT * FROM Customers WHERE CustomerID = @CustomerID;
```

The Results pane shows the data before and after the update. The status bar at the bottom indicates 'Query executed successfully.' and '2 rows'.

CustomerID	FirstName	LastName	Email	Phone	Address
1	7	Clint	Barton	clint@gmail.com	9999999997 Chennai

CustomerID	FirstName	LastName	Email	Phone	Address
1	7	Clint	Barton	clintbarton@gmail.com	9999999997 Chennai

Output 8:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanDB', including tables like 'Orders' and 'OrderDetails'. The main query window contains the following SQL code:

```
-- 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.
SELECT * FROM Orders;
-- The "TotalAmount" column has already been set according to the "Price" and "Quantity" columns,
-- when the order was placed and the record for it was inserted.
```

The Results pane shows the data after the query execution. The status bar at the bottom indicates 'Query executed successfully.' and '10 rows'.

OrderID	CustomerID	OrderDate	TotalAmount
1	5	2024-08-01	1000
2	7	2024-08-02	3000
3	6	2024-08-02	10000
4	4	2024-09-03	4000
5	9	2024-09-04	2000
6	2	2024-09-06	500
7	1	2024-09-15	200
8	3	2024-09-16	10000
9	8	2024-09-17	6000
10	2	2024-09-19	1000

Output 9:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'Niranjandb'. The main query window contains the following SQL script:

```
-- 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.

SELECT * FROM Orders;
SELECT * FROM OrderDetails;
BEGIN TRANSACTION;
DECLARE @CustomerID INT = 6;
DELETE FROM OrderDetails WHERE OrderID IN (SELECT OrderID FROM Orders WHERE CustomerID = @CustomerID);
DELETE FROM Orders WHERE CustomerID = @CustomerID;
COMMIT TRANSACTION;
SELECT * FROM Orders;
SELECT * FROM OrderDetails;
```

The Results pane shows the output of the query, displaying the 'Orders' and 'OrderDetails' tables. The 'Orders' table has 4 rows, and the 'OrderDetails' table has 7 rows.

OrderID	CustomerID	OrderDate	TotalAmount
1	5	2024-09-01	1000
2	7	2024-09-02	3000
3	4	2024-09-03	4000
4	9	2024-09-04	2000

OrderDetailID	OrderID	ProductID	Quantity
1	1	3	1
2	2	2	1
3	4	8	1
4	5	9	2
5	6	4	5
6	7	10	1

Output 10:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'Niranjandb'. The main query window contains the following SQL script:

```
-- 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.

BEGIN TRANSACTION;
SELECT * FROM Products;
INSERT INTO Products (ProductID, ProductName, Description, Price) VALUES (11, 'Speaker', 'Bluetooth Speaker', 2000);
SELECT * FROM Products;
COMMIT TRANSACTION;
```

The Results pane shows the output of the query, displaying the 'Products' table. The table has 11 rows, including the newly inserted 'Speaker' product.

ProductID	ProductName	Description	Price
4	Earphones	Wired Earphones	100
5	USB Cable	USB Cable	100
6	Mouse	Wired Mouse	300
7	Router	Wifi Router	1500
8	Monitor	Desktop Monitor	4000
9	Keyboard	PC Keyboard	1000
10	Charger	Mobile Charger	200
11	Speaker	Bluetooth Speaker	2000

Output 11:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanaDB', including tables like 'Customers', 'Inventory', 'OrderDetails', and 'Orders'. The main query window contains the following SQL code:

```
-- 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.

UPDATE Orders SET Status='Pending';
SELECT * FROM Orders;
BEGIN TRANSACTION;
DECLARE @OrderID INT = 8;
UPDATE Orders SET Status='Shipped' WHERE OrderID=@OrderID;
SELECT * FROM Orders;
COMMIT TRANSACTION;
```

The Results pane shows the output of the query, displaying a table with columns: OrderID, CustomerID, OrderDate, TotalAmount, and Status. The data is as follows:

OrderID	CustomerID	OrderDate	TotalAmount	Status
2	7	2024-09-02	3000	Pending
3	4	2024-09-03	4000	Pending
4	5	2024-09-04	2000	Pending
5	6	2024-09-06	500	Pending
6	7	2024-09-15	200	Pending
7	8	2024-09-16	10000	Shipped
8	9	2024-09-17	6000	Pending
9	10	2024-09-19	1000	Pending

The status bar at the bottom indicates 'Query executed successfully.' and '18 rows'.

Output 12:

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'NiranjanaDB', including tables like 'Customers', 'Inventory', 'OrderDetails', and 'Orders'. The main query window contains the following SQL code:

```
-- 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.

BEGIN TRANSACTION;
UPDATE Customers SET NumberOfOrders = (SELECT COUNT(*) FROM Orders WHERE Orders.CustomerID = Customers.CustomerID);
SELECT * FROM Customers;
COMMIT TRANSACTION;
```

The Results pane shows the output of the query, displaying a table with columns: CustomerID, FirstName, LastName, Email, Phone, Address, and NumberOfOrders. The data is as follows:

CustomerID	FirstName	LastName	Email	Phone	Address	NumberOfOrders
1	Nirajan	Kolpe	nirajan@gmail.com	9999999991	San Francisco	1
2	Steve	Rogers	steve@gmail.com	9999999992	Brooklyn	2
3	Tony	Stark	tony@gmail.com	9999999993	New York	1
4	Natasha	Romanoff	natasha@gmail.com	9999999994	New York	1
5	Bruce	Banner	bruce@gmail.com	9999999995	Delhi	1
6	Thor	Odinson	thor@gmail.com	9999999996	Washington	0
7	Clint	Barton	clintbarton@gmail.com	9999999997	Chennai	1
8	Wanda	Maximoff	wanda@gmail.com	9999999998	Mumbai	1
9	Sam	Wilson	sam@gmail.com	9999999999	San Francisco	1
10	Peter	Parker	peter@gmail.com	9999999981	Queens	0

The status bar at the bottom indicates 'Query executed successfully.' and '10 rows'.