# Foundation Technical Training

## Lab Exercise: Data Selection

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

**Problem Statements:**

1. Write a query to display the customer list by the first name in descending order.
2. Write a query to display the first name, last name, and city of the customers. It sorts the customer list by the city first and then by the first name.
3. Write a query to return the top three most expensive products.
4. Write a query to find the products whose list price is greater than 300 and model year is 2018.
5. Write a query to find products whose list price is greater than 3,000 or model year is 2018. Any product that meets one of these conditions is included in the result set.
6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.
7. Write a query using the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
8. Write a query to the customers where the first character in the last name is the letter in the range A through C.
9. Write a query using NOT LIKE operator to find customers where the first character in the first name is not the letter A:
10. Write a query to return the number of customers by state and city group state and city.
11. Write a query to return the number of orders placed by the customer group by customer id and year.
12. Write a query to find the maximum and minimum list group by category id. Then, it filters out the category which has the maximum list price greater than 4,000 or the minimum list price less than 500.

**Program Code in SQL:**

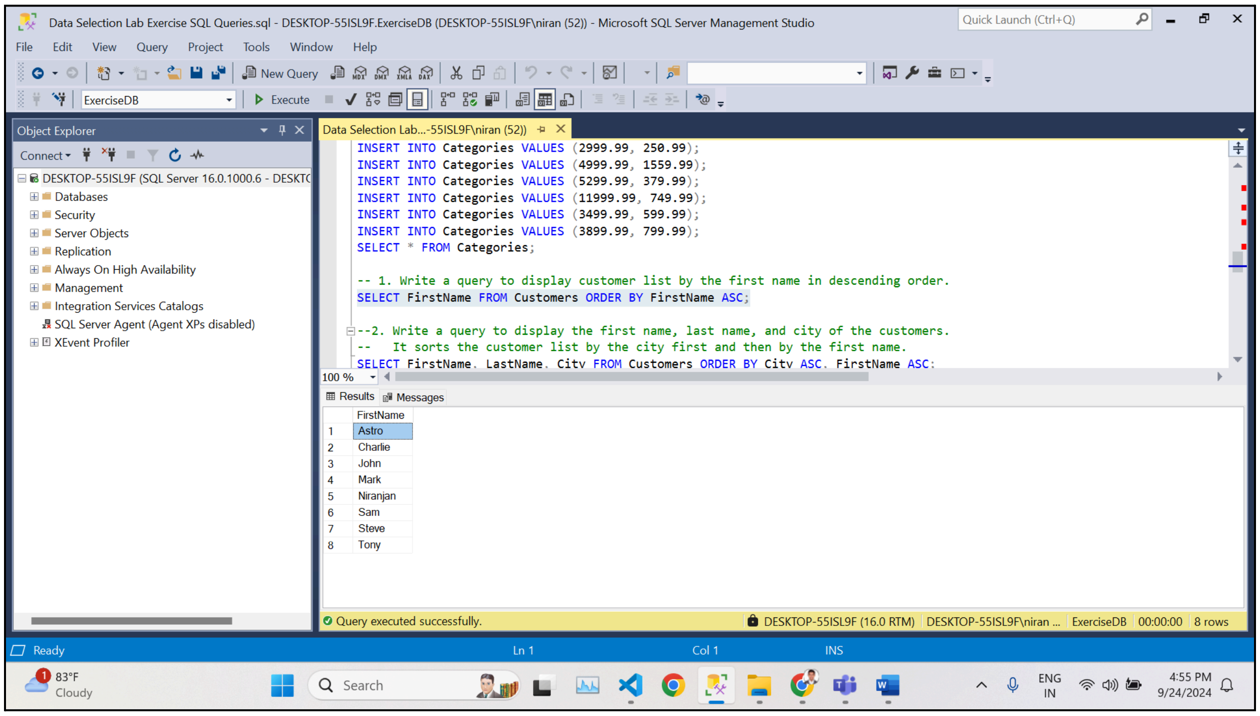
```sql
-- MS-SQL Server

-- Lab Exercise: Data Selection

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



-- Setup Queries for Lab Questions

USE master;

CREATE DATABASE ExerciseDB;

USE ExerciseDB;
```

```sql
CREATE  TABLE  Customers  (CustomerID  INT  PRIMARY  KEY  IDENTITY,  FirstName
VARCHAR(20) NOT NULL,

                                    LastName VARCHAR(20), City VARCHAR(20), State
VARCHAR(20));

INSERT INTO Customers VALUES ('Niranjan', 'Kolpe',   'Mumbai',    'Maharashtra');

INSERT INTO Customers VALUES ('Mark',     'Ruffalo', 'Bengaluru', 'Karnataka');

INSERT INTO Customers VALUES ('Sam',      'Wilson',  'Chennai',   'Tamilnadu');

INSERT INTO Customers VALUES ('John',     'Wick',    'Chennai',   'Tamilnadu');

INSERT INTO Customers VALUES ('Steve',    'Rogers',  'Mumbai',    'Maharashtra');

INSERT  INTO  Customers  VALUES  ('Tony',       'Stark',     'Indore',      'Madhya
Pradesh');

INSERT  INTO  Customers  VALUES  ('Astro',       'Austin',    'Lucknow',     'Uttar
Pradesh');

INSERT INTO Customers VALUES ('Charlie',  'Chester', 'Pune',      'Maharashtra');

SELECT * FROM Customers;


CREATE  TABLE  Products  (ProductID  INT  PRIMARY  KEY  IDENTITY,  ProductName
VARCHAR(20) NOT NULL,

                        ModelYear INT NOT NULL, Price FLOAT NOT NULL);

INSERT INTO Products VALUES ('Smartphone', 2018, 1929.99);

INSERT INTO Products VALUES ('Laptop',     2021, 39999.99);

INSERT INTO Products VALUES ('Speaker',    2022, 299.99);

INSERT INTO Products VALUES ('Charger',    2021, 489.99);

INSERT INTO Products VALUES ('Earpods',    2023, 466.99);

SELECT * FROM Products;


CREATE TABLE Orders (OrderID INT PRIMARY KEY IDENTITY,
                     CustomerID INT FOREIGN KEY REFERENCES Customers(CustomerID),
                     ProductID INT FOREIGN KEY REFERENCES Products(ProductID),
                     Quantity INT NOT NULL DEFAULT 0,
                     OrderYear INT NOT NULL);
INSERT INTO Orders VALUES (1, 1, 1, 2018);
INSERT INTO Orders VALUES (2, 2, 1, 2021);
INSERT INTO Orders VALUES (3, 3, 1, 2022);
INSERT INTO Orders VALUES (4, 4, 1, 2021);
INSERT INTO Orders VALUES (5, 5, 1, 2023);
INSERT INTO Orders VALUES (2, 4, 1, 2023);
INSERT INTO Orders VALUES (3, 1, 1, 2021);
INSERT INTO Orders VALUES (5, 2, 1, 2023);
SELECT * FROM Orders;

CREATE TABLE Categories (CategoryID INT PRIMARY KEY IDENTITY, MaxListPrice FLOAT
NOT NULL, MinListPrice FLOAT NOT NULL);
INSERT INTO Categories VALUES (489.99,  89.99);
INSERT INTO Categories VALUES (2599.99, 416.99);
INSERT INTO Categories VALUES (2999.99, 250.99);
INSERT INTO Categories VALUES (4999.99, 1559.99);
INSERT INTO Categories VALUES (5299.99, 379.99);
INSERT INTO Categories VALUES (11999.99, 749.99);
INSERT INTO Categories VALUES (3499.99, 599.99);
INSERT INTO Categories VALUES (3899.99, 799.99);
SELECT * FROM Categories;
```

```sql
-- 1. Write a query to display customer list by the first name in descending order.

SELECT FirstName FROM Customers ORDER BY FirstName ASC;


--2. Write a query to display the first name, last name, and city of the customers.

--   It sorts the customer list by the city first and then by the first name.

SELECT FirstName, LastName, City FROM Customers ORDER BY City ASC, FirstName ASC;


--3. Write a query to returns the top three most expensive products.

SELECT TOP 3 ProductName, Price FROM Products ORDER BY Price DESC;


--4. Write a query to finds the products whose list price is greater than 300 and model year is 2018.

SELECT * FROM Products WHERE Price>300.00 AND ModelYear=2018;


--5. Write a query to finds products whose list price is greater than 3,000 or model year is 2018.

--    Any product that meets one of these conditions is included in the result set.

SELECT * FROM Products WHERE Price>3000.00 OR ModelYear=2018;


--6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.

SELECT * FROM Products WHERE Price>1899.00 AND Price<1999.99;


--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
```

```sql
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);


--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:

SELECT * FROM Customers WHERE LastName LIKE ('[a-c]%');


--9. Write a query using NOT LIKE operator to find customers,

--   where the first character in the first name is not the letter A:

SELECT * FROM Customers WHERE FirstName NOT LIKE ('a%');


--10. Write a query to return the number of customers by state and city group state and city.

SELECT State, City, COUNT(*) AS CustomerCount FROM Customers GROUP BY State, City ORDER BY State ASC, City ASC;


--11. Write a query to return the number of orders placed by the customer group by customer id and year.

SELECT CustomerID, OrderYear, COUNT(*) AS OrderCount FROM Orders GROUP BY CustomerID, OrderYear ORDER BY CustomerID ASC, OrderYear ASC;


--12. Write query to finds the maximum and minimum list group by category id.

--    Then, it filters out the category which has the maximum list price,

--    greater than 4,000 or the minimum list price less than 500.

SELECT * FROM Categories WHERE MaxListPrice=(SELECT MAX(MaxListPrice) FROM Categories) OR MaxListPrice>4000.00 OR MinListPrice<500.00 GROUP BY CategoryID, MaxListPrice, MinListPrice;
```

**Output 1:**



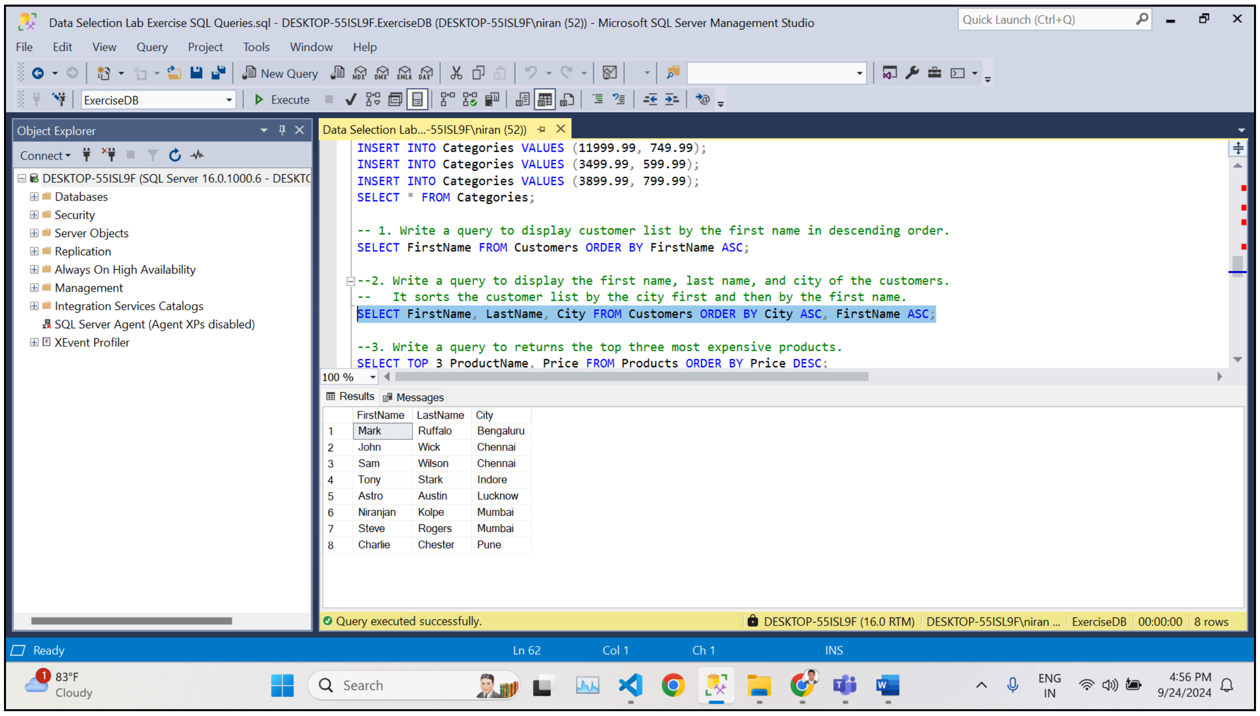
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```
INSERT INTO Categories VALUES (2999.99, 250.99);
INSERT INTO Categories VALUES (4999.99, 1559.99);
INSERT INTO Categories VALUES (5299.99, 379.99);
INSERT INTO Categories VALUES (11999.99, 749.99);
INSERT INTO Categories VALUES (3499.99, 599.99);
INSERT INTO Categories VALUES (3899.99, 799.99);
SELECT * FROM Categories;

-- 1. Write a query to display customer list by the first name in descending order.
SELECT FirstName FROM Customers ORDER BY FirstName ASC;

--2. Write a query to display the first name, last name, and city of the customers.
--   It sorts the customer list by the city first and then by the first name.
SELECT FirstName, LastName, City FROM Customers ORDER BY City ASC, FirstName ASC;
```

| | FirstName |
|---|---|
| 1 | Astro |
| 2 | Charlie |
| 3 | John |
| 4 | Mark |
| 5 | Niranjan |
| 6 | Sam |
| 7 | Steve |
| 8 | Tony |

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 8 rows

**Output 2:**



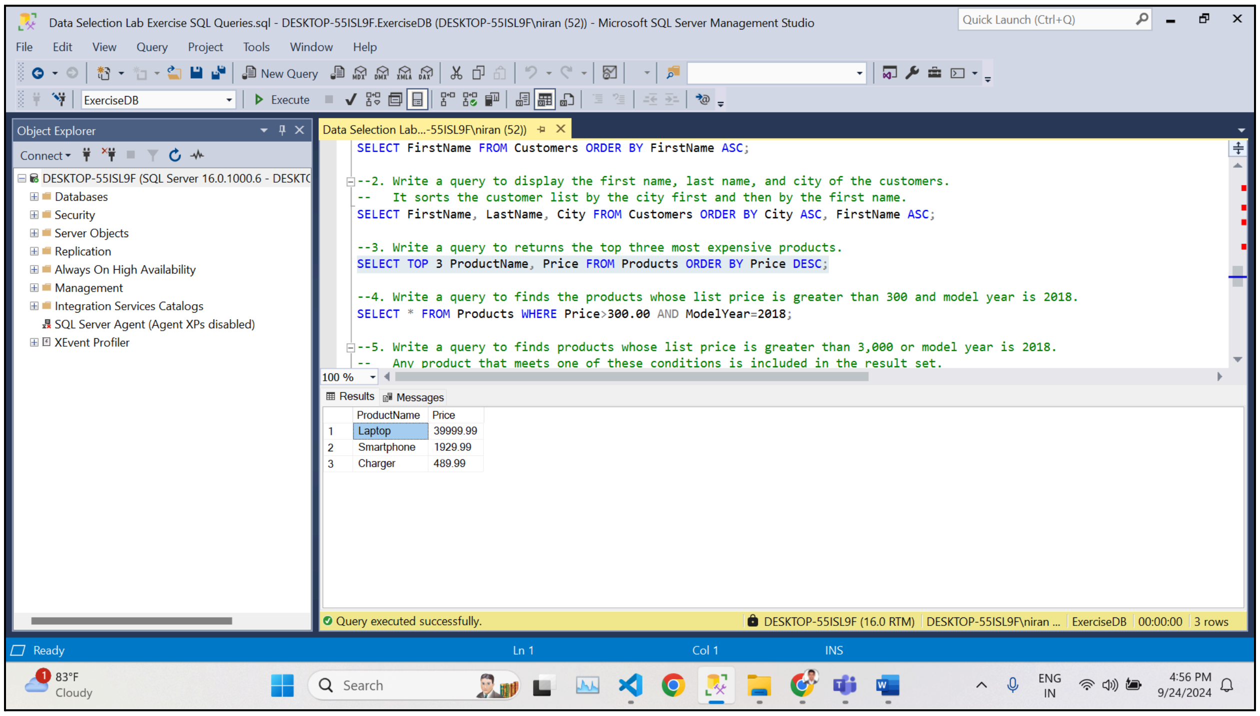
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```
INSERT INTO Categories VALUES (11999.99, 749.99);
INSERT INTO Categories VALUES (3499.99, 599.99);
INSERT INTO Categories VALUES (3899.99, 799.99);
SELECT * FROM Categories;

-- 1. Write a query to display customer list by the first name in descending order.
SELECT FirstName FROM Customers ORDER BY FirstName ASC;

--2. Write a query to display the first name, last name, and city of the customers.
--   It sorts the customer list by the city first and then by the first name.
SELECT FirstName, LastName, City FROM Customers ORDER BY City ASC, FirstName ASC;

--3. Write a query to returns the top three most expensive products.
SELECT TOP 3 ProductName, Price FROM Products ORDER BY Price DESC;
```

| | FirstName | LastName | City |
|---|---|---|---|
| 1 | Mark | Ruffalo | Bengaluru |
| 2 | John | Wick | Chennai |
| 3 | Sam | Wilson | Chennai |
| 4 | Tony | Stark | Indore |
| 5 | Astro | Austin | Lucknow |
| 6 | Niranjan | Kolpe | Mumbai |
| 7 | Steve | Rogers | Mumbai |
| 8 | Charlie | Chester | Pune |

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 8 rows

**Output 3:**



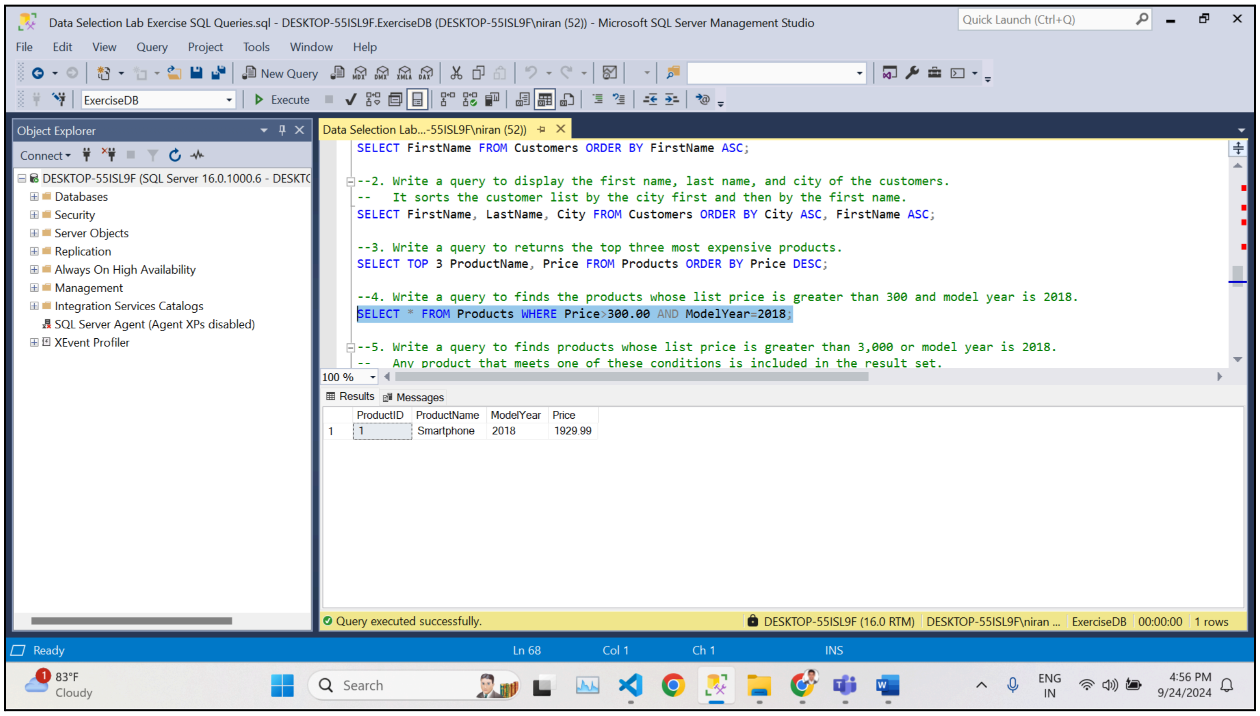
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
SELECT FirstName FROM Customers ORDER BY FirstName ASC;

--2. Write a query to display the first name, last name, and city of the customers.
--   It sorts the customer list by the city first and then by the first name.
SELECT FirstName, LastName, City FROM Customers ORDER BY City ASC, FirstName ASC;

--3. Write a query to returns the top three most expensive products.
SELECT TOP 3 ProductName, Price FROM Products ORDER BY Price DESC;

--4. Write a query to finds the products whose list price is greater than 300 and model year is 2018.
SELECT * FROM Products WHERE Price>300.00 AND ModelYear=2018;

--5. Write a query to finds products whose list price is greater than 3,000 or model year is 2018.
--   Any product that meets one of these conditions is included in the result set.
```

| | ProductName | Price |
|---|---|---|
| 1 | Laptop | 39999.99 |
| 2 | Smartphone | 1929.99 |
| 3 | Charger | 489.99 |

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 3 rows

**Output 4:**



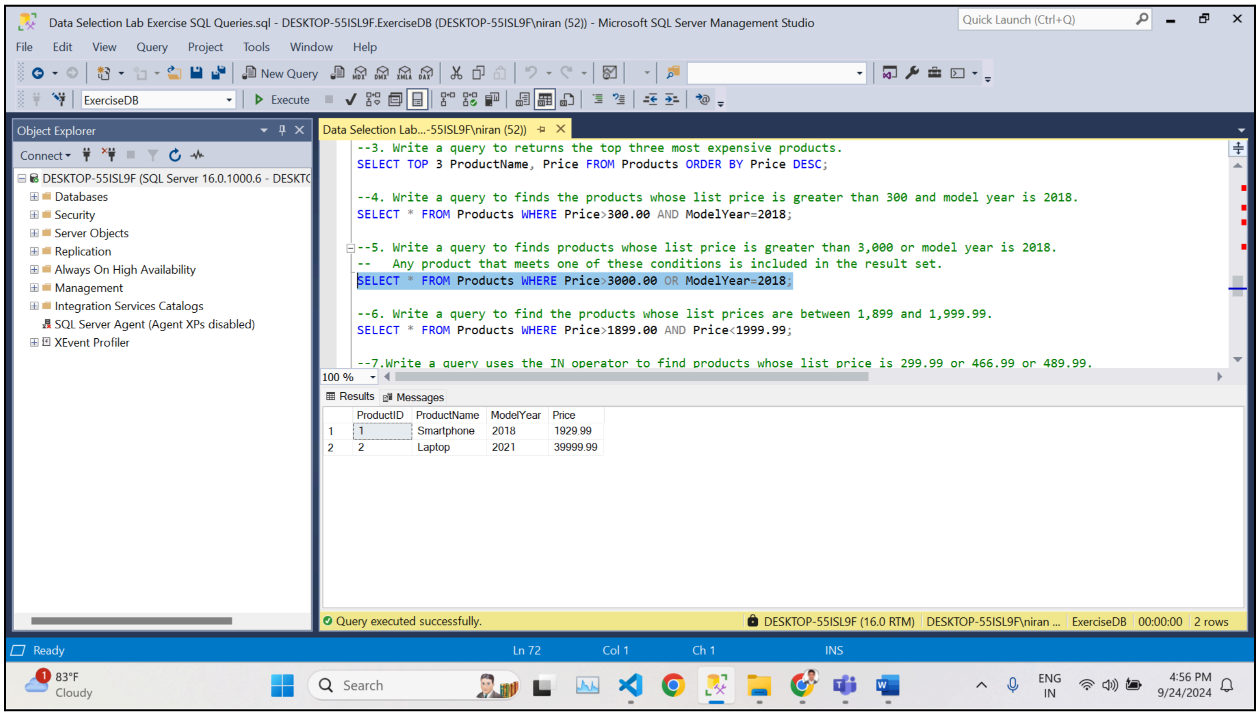
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
SELECT FirstName FROM Customers ORDER BY FirstName ASC;

--2. Write a query to display the first name, last name, and city of the customers.
--   It sorts the customer list by the city first and then by the first name.
SELECT FirstName, LastName, City FROM Customers ORDER BY City ASC, FirstName ASC;

--3. Write a query to returns the top three most expensive products.
SELECT TOP 3 ProductName, Price FROM Products ORDER BY Price DESC;

--4. Write a query to finds the products whose list price is greater than 300 and model year is 2018.
SELECT * FROM Products WHERE Price>300.00 AND ModelYear=2018;

--5. Write a query to finds products whose list price is greater than 3,000 or model year is 2018.
--   Any product that meets one of these conditions is included in the result set.
```

| | ProductID | ProductName | ModelYear | Price |
|---|---|---|---|---|
| 1 | 1 | Smartphone | 2018 | 1929.99 |

Query executed successfully. DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 1 rows

**Output 5:**

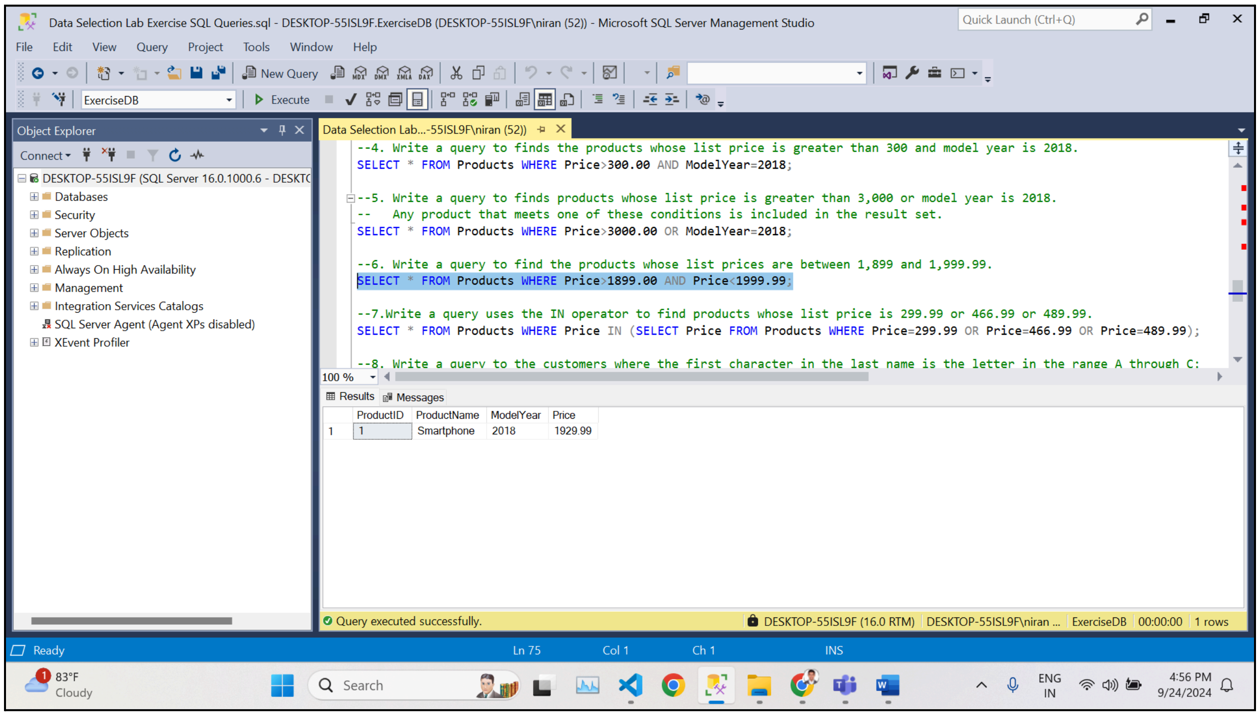

Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio
File
Edit
View
Query
Project
Tools
Window
Help
New Query
ExerciseDB
Execute
Object Explorer
DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTO
Databases
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
Data Selection Lab...-55ISL9F\niran (52))
100 %
Results
Messages
Ready
Ln 72
Col 1
Ch 1
INS

```
--3. Write a query to returns the top three most expensive products.
SELECT TOP 3 ProductName, Price FROM Products ORDER BY Price DESC;

--4. Write a query to finds the products whose list price is greater than 300 and model year is 2018.
SELECT * FROM Products WHERE Price>300.00 AND ModelYear=2018;

--5. Write a query to finds products whose list price is greater than 3,000 or model year is 2018.
--   Any product that meets one of these conditions is included in the result set.
SELECT * FROM Products WHERE Price>3000.00 OR ModelYear=2018;

--6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.
SELECT * FROM Products WHERE Price>1899.00 AND Price<1999.99;

--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
```

| | ProductID | ProductName | ModelYear | Price |
|---|---|---|---|---|
| 1 | 1 | Smartphone | 2018 | 1929.99 |
| 2 | 2 | Laptop | 2021 | 39999.99 |

```
Query executed successfully.    DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 2 rows
```

**Output 6:**

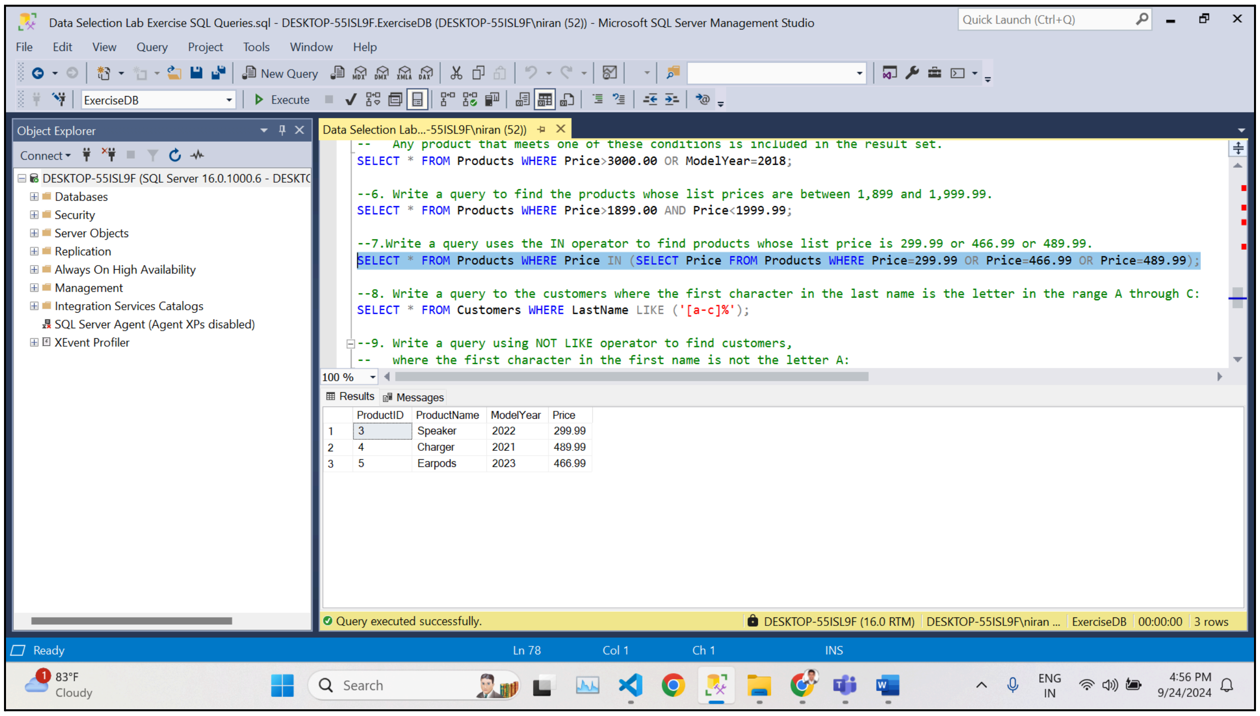

Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio
File
Edit
View
Query
Project
Tools
Window
Help
New Query
ExerciseDB
Execute
Object Explorer
DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTO
Databases
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
Data Selection Lab...-55ISL9F\niran (52))
100 %
Results
Messages
Ready
Ln 75
Col 1
Ch 1
INS

```
--4. Write a query to finds the products whose list price is greater than 300 and model year is 2018.
SELECT * FROM Products WHERE Price>300.00 AND ModelYear=2018;

--5. Write a query to finds products whose list price is greater than 3,000 or model year is 2018.
--   Any product that meets one of these conditions is included in the result set.
SELECT * FROM Products WHERE Price>3000.00 OR ModelYear=2018;

--6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.
SELECT * FROM Products WHERE Price>1899.00 AND Price<1999.99;

--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);

--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:
```

| | ProductID | ProductName | ModelYear | Price |
|---|---|---|---|---|
| 1 | 1 | Smartphone | 2018 | 1929.99 |

```
Query executed successfully.    DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 1 rows
```

**Output 7:**



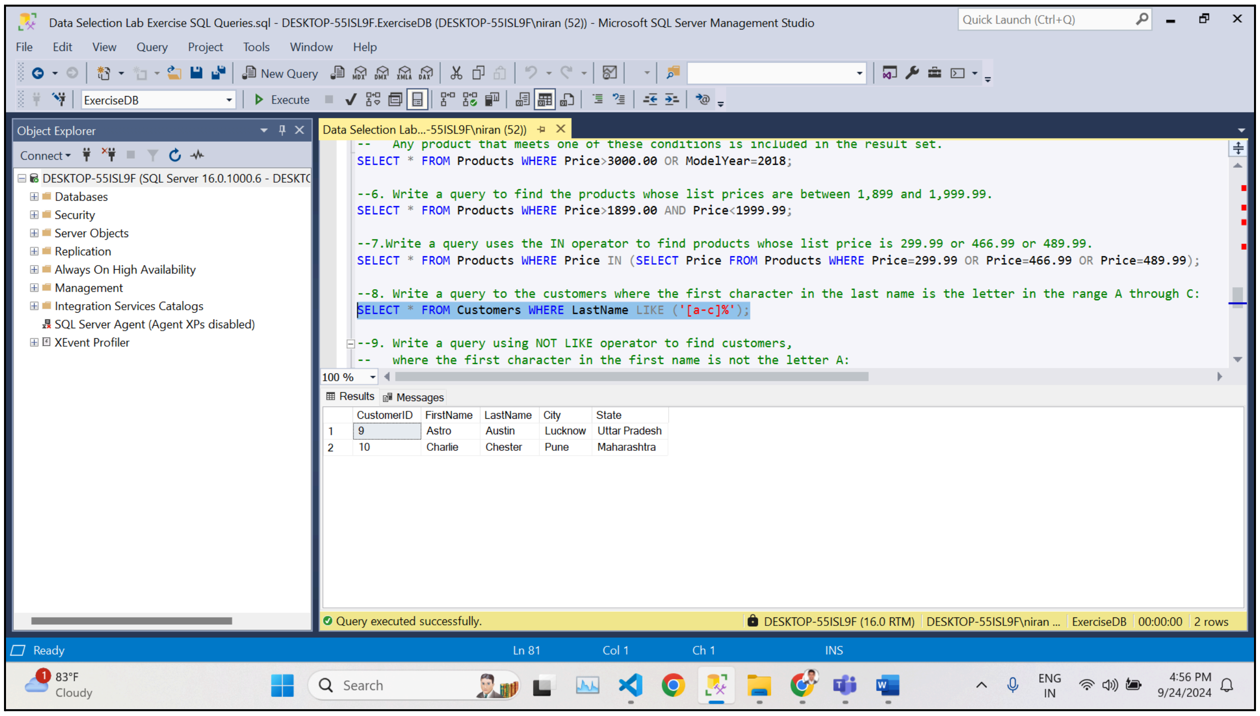
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
--   Any product that meets one of these conditions is included in the result set.
SELECT * FROM Products WHERE Price>3000.00 OR ModelYear=2018;

--6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.
SELECT * FROM Products WHERE Price>1899.00 AND Price<1999.99;

--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);

--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:
SELECT * FROM Customers WHERE LastName LIKE ('[a-c]%');

--9. Write a query using NOT LIKE operator to find customers,
--   where the first character in the first name is not the letter A:
```

| | ProductID | ProductName | ModelYear | Price |
|---|---|---|---|---|
| 1 | 3 | Speaker | 2022 | 299.99 |
| 2 | 4 | Charger | 2021 | 489.99 |
| 3 | 5 | Earpods | 2023 | 466.99 |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 3 rows

**Output 8:**



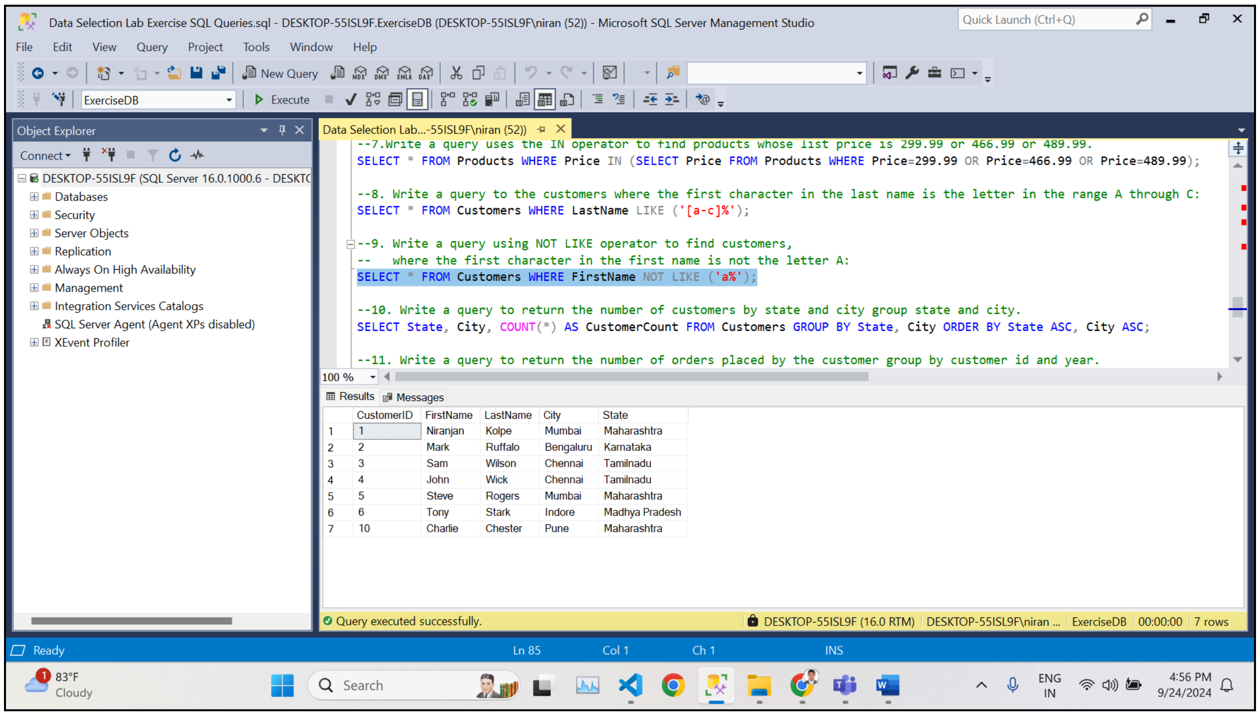
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
--   Any product that meets one of these conditions is included in the result set.
SELECT * FROM Products WHERE Price>3000.00 OR ModelYear=2018;

--6. Write a query to find the products whose list prices are between 1,899 and 1,999.99.
SELECT * FROM Products WHERE Price>1899.00 AND Price<1999.99;

--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);

--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:
SELECT * FROM Customers WHERE LastName LIKE ('[a-c]%');

--9. Write a query using NOT LIKE operator to find customers,
--   where the first character in the first name is not the letter A:
```

| | CustomerID | FirstName | LastName | City | State |
|---|---|---|---|---|---|
| 1 | 9 | Astro | Austin | Lucknow | Uttar Pradesh |
| 2 | 10 | Charlie | Chester | Pune | Maharashtra |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 2 rows

## Output 9:

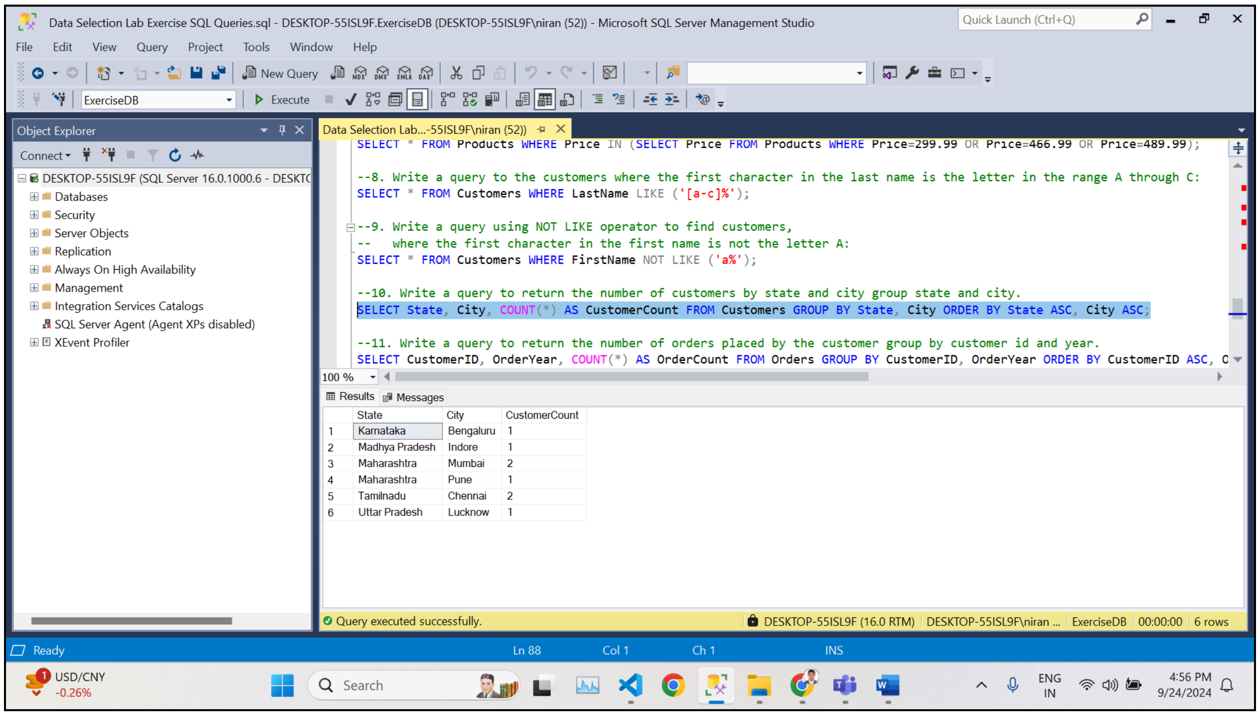

Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
ExerciseDB
Execute
Object Explorer
DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTO
Databases
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
Data Selection Lab...-55ISL9F\niran (52))
Results
Messages
Ready
Ln 85
Col 1
Ch 1
INS

```
--7.Write a query uses the IN operator to find products whose list price is 299.99 or 466.99 or 489.99.
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);

--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:
SELECT * FROM Customers WHERE LastName LIKE ('[a-c]%');

--9. Write a query using NOT LIKE operator to find customers,
--   where the first character in the first name is not the letter A:
SELECT * FROM Customers WHERE FirstName NOT LIKE ('a%');

--10. Write a query to return the number of customers by state and city group state and city.
SELECT State, City, COUNT(*) AS CustomerCount FROM Customers GROUP BY State, City ORDER BY State ASC, City ASC;

--11. Write a query to return the number of orders placed by the customer group by customer id and year.
```

| | CustomerID | FirstName | LastName | City | State |
|---|---|---|---|---|---|
| 1 | 1 | Niranjan | Kolpe | Mumbai | Maharashtra |
| 2 | 2 | Mark | Ruffalo | Bengaluru | Karnataka |
| 3 | 3 | Sam | Wilson | Chennai | Tamilnadu |
| 4 | 4 | John | Wick | Chennai | Tamilnadu |
| 5 | 5 | Steve | Rogers | Mumbai | Maharashtra |
| 6 | 6 | Tony | Stark | Indore | Madhya Pradesh |
| 7 | 10 | Charlie | Chester | Pune | Maharashtra |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 7 rows

## Output 10:

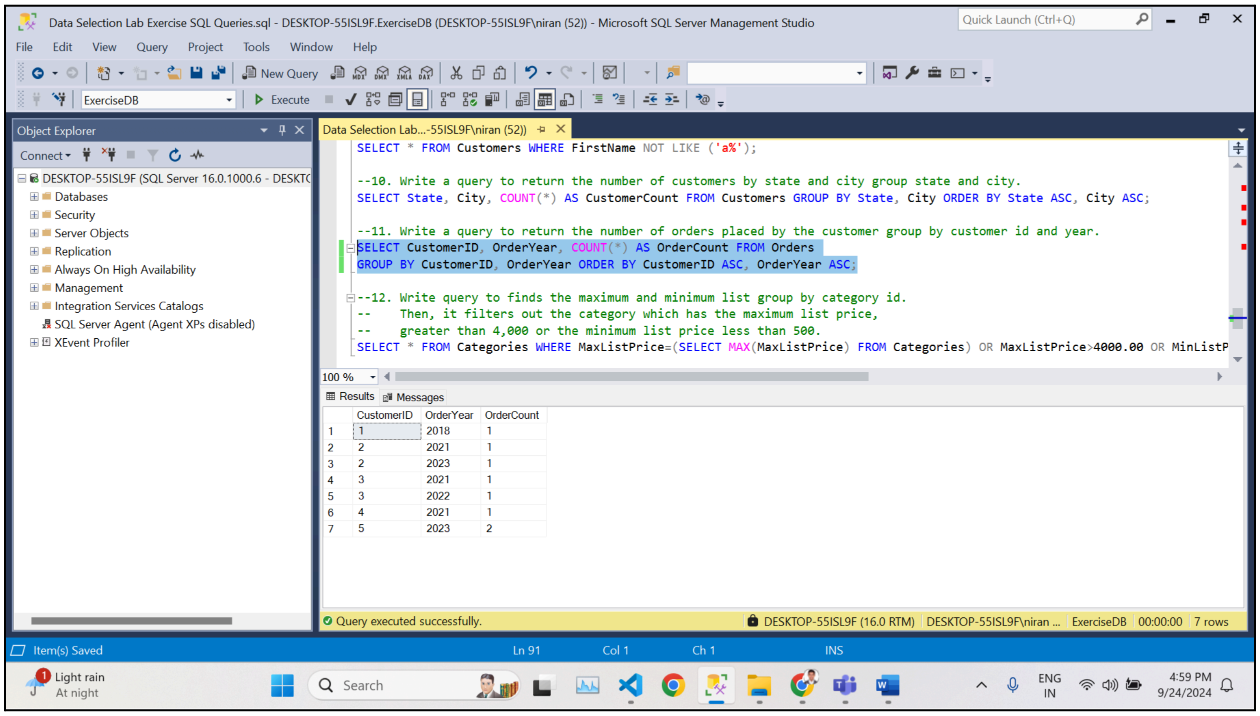

Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
ExerciseDB
Execute
Object Explorer
DESKTOP-55ISL9F (SQL Server 16.0.1000.6 - DESKTO
Databases
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler
Data Selection Lab...-55ISL9F\niran (52))
Results
Messages
Ready
Ln 88
Col 1
Ch 1
INS

```
SELECT * FROM Products WHERE Price IN (SELECT Price FROM Products WHERE Price=299.99 OR Price=466.99 OR Price=489.99);

--8. Write a query to the customers where the first character in the last name is the letter in the range A through C:
SELECT * FROM Customers WHERE LastName LIKE ('[a-c]%');

--9. Write a query using NOT LIKE operator to find customers,
--   where the first character in the first name is not the letter A:
SELECT * FROM Customers WHERE FirstName NOT LIKE ('a%');

--10. Write a query to return the number of customers by state and city group state and city.
SELECT State, City, COUNT(*) AS CustomerCount FROM Customers GROUP BY State, City ORDER BY State ASC, City ASC;

--11. Write a query to return the number of orders placed by the customer group by customer id and year.
SELECT CustomerID, OrderYear, COUNT(*) AS OrderCount FROM Orders GROUP BY CustomerID, OrderYear ORDER BY CustomerID ASC, O
```

| | State | City | CustomerCount |
|---|---|---|---|
| 1 | Karnataka | Bengaluru | 1 |
| 2 | Madhya Pradesh | Indore | 1 |
| 3 | Maharashtra | Mumbai | 2 |
| 4 | Maharashtra | Pune | 1 |
| 5 | Tamilnadu | Chennai | 2 |
| 6 | Uttar Pradesh | Lucknow | 1 |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 6 rows

## Output 11:



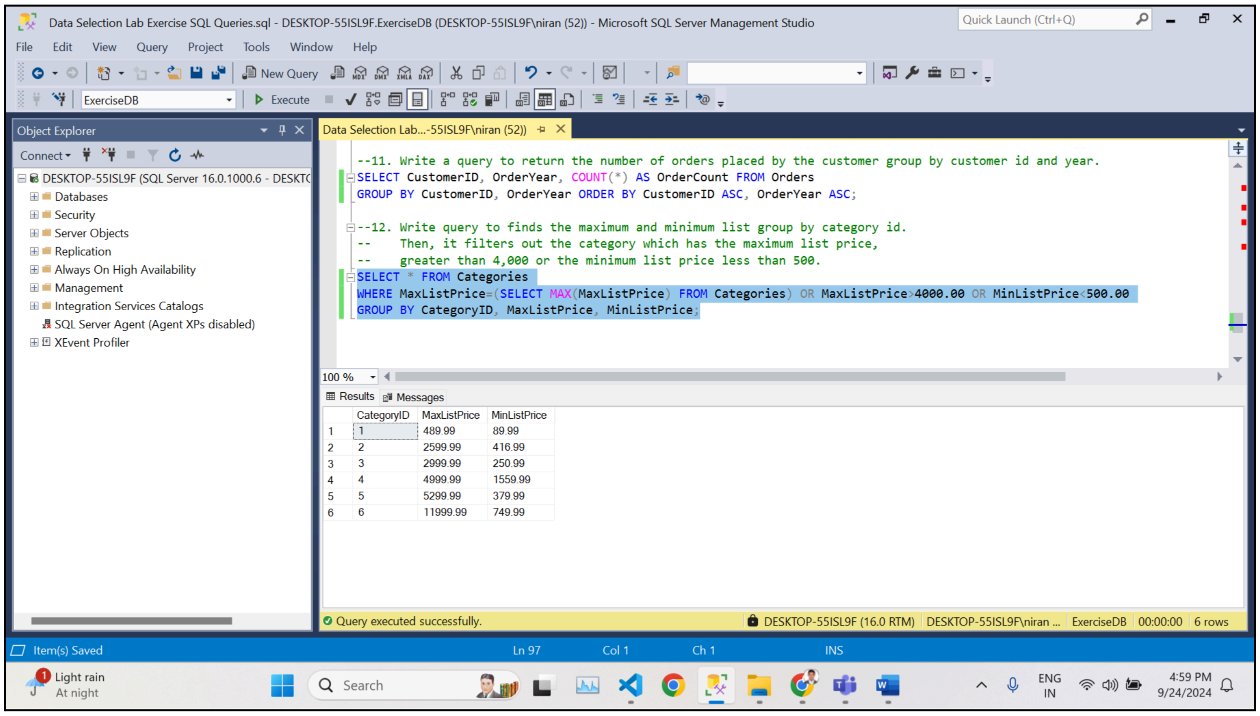
Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
SELECT * FROM Customers WHERE FirstName NOT LIKE ('a%');

--10. Write a query to return the number of customers by state and city group state and city.
SELECT State, City, COUNT(*) AS CustomerCount FROM Customers GROUP BY State, City ORDER BY State ASC, City ASC;

--11. Write a query to return the number of orders placed by the customer group by customer id and year.
SELECT CustomerID, OrderYear, COUNT(*) AS OrderCount FROM Orders
GROUP BY CustomerID, OrderYear ORDER BY CustomerID ASC, OrderYear ASC;

--12. Write query to finds the maximum and minimum list group by category id.
--    Then, it filters out the category which has the maximum list price,
--    greater than 4,000 or the minimum list price less than 500.
SELECT * FROM Categories WHERE MaxListPrice=(SELECT MAX(MaxListPrice) FROM Categories) OR MaxListPrice>4000.00 OR MinListP
```

| | CustomerID | OrderYear | OrderCount |
|---|---|---|---|
| 1 | 1 | 2018 | 1 |
| 2 | 2 | 2021 | 1 |
| 3 | 2 | 2023 | 1 |
| 4 | 3 | 2021 | 1 |
| 5 | 3 | 2022 | 1 |
| 6 | 4 | 2021 | 1 |
| 7 | 5 | 2023 | 2 |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 7 rows

## Output 12:



Data Selection Lab Exercise SQL Queries.sql - DESKTOP-55ISL9F.ExerciseDB (DESKTOP-55ISL9F\niran (52)) - Microsoft SQL Server Management Studio

```sql
--11. Write a query to return the number of orders placed by the customer group by customer id and year.
SELECT CustomerID, OrderYear, COUNT(*) AS OrderCount FROM Orders
GROUP BY CustomerID, OrderYear ORDER BY CustomerID ASC, OrderYear ASC;

--12. Write query to finds the maximum and minimum list group by category id.
--    Then, it filters out the category which has the maximum list price,
--    greater than 4,000 or the minimum list price less than 500.
SELECT * FROM Categories
WHERE MaxListPrice=(SELECT MAX(MaxListPrice) FROM Categories) OR MaxListPrice>4000.00 OR MinListPrice<500.00
GROUP BY CategoryID, MaxListPrice, MinListPrice;
```

| | CategoryID | MaxListPrice | MinListPrice |
|---|---|---|---|
| 1 | 1 | 489.99 | 89.99 |
| 2 | 2 | 2599.99 | 416.99 |
| 3 | 3 | 2999.99 | 250.99 |
| 4 | 4 | 4999.99 | 1559.99 |
| 5 | 5 | 5299.99 | 379.99 |
| 6 | 6 | 11999.99 | 749.99 |

Query executed successfully. | DESKTOP-55ISL9F (16.0 RTM) | DESKTOP-55ISL9F\niran ... | ExerciseDB | 00:00:00 | 6 rows