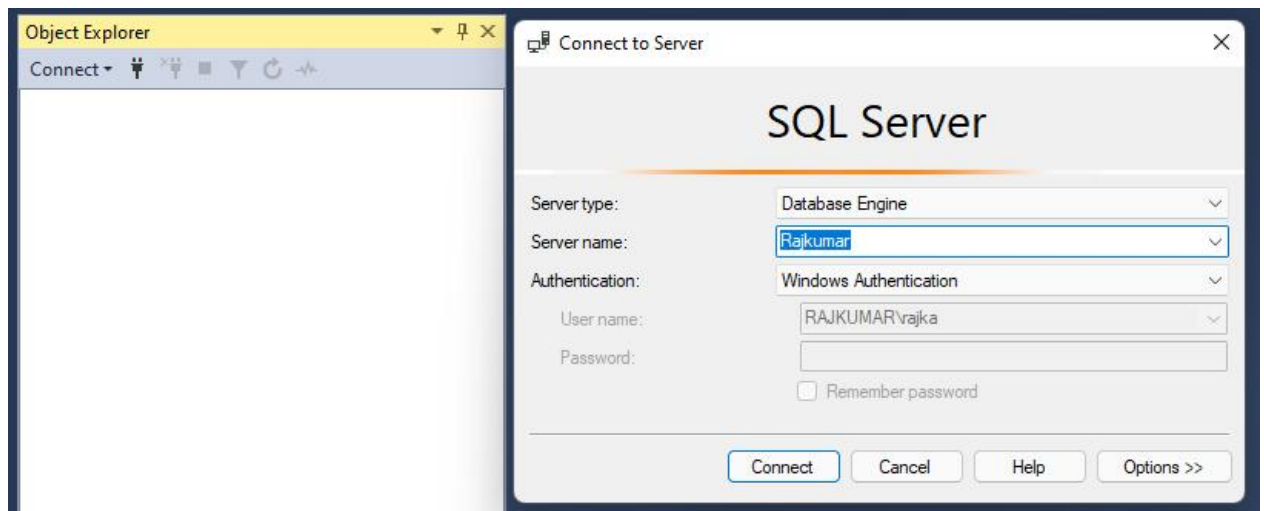
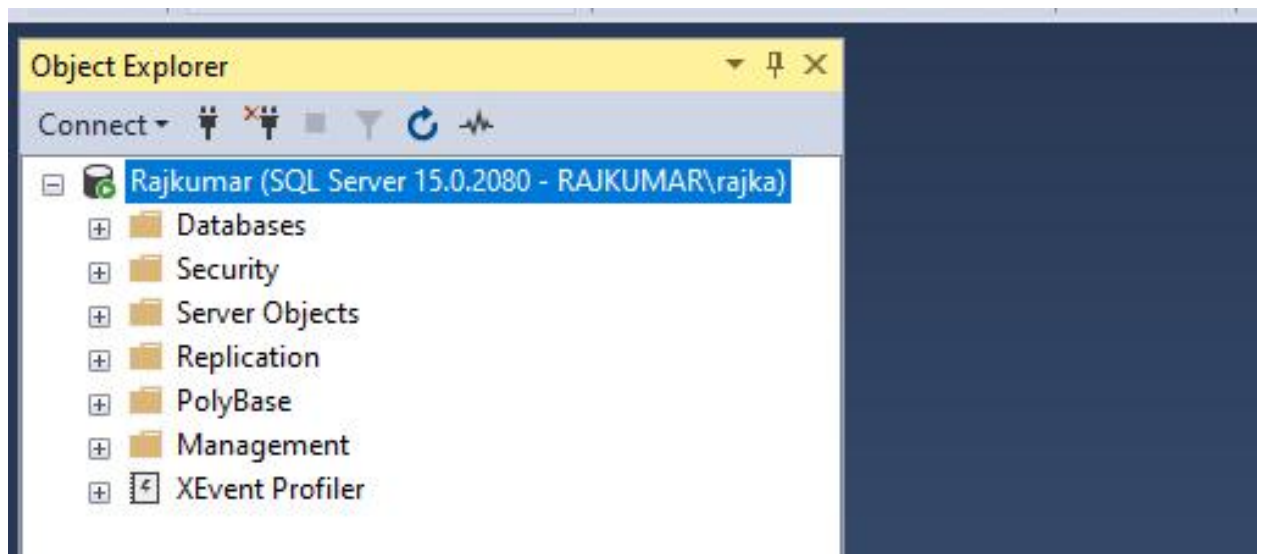


Connecting to server:



Connected to Server:

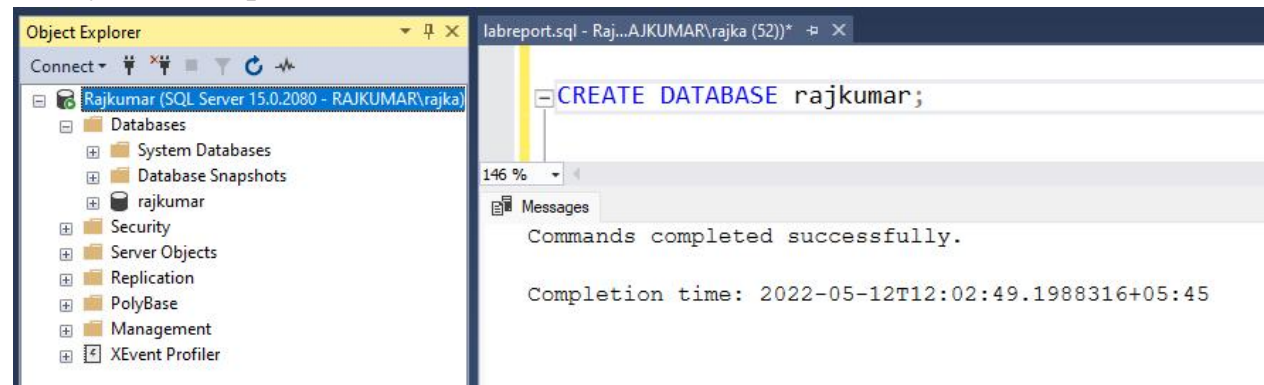


1. Creating and removing database queries

i. Syntax: (CREATE)

CREATE DATABASE db_name;

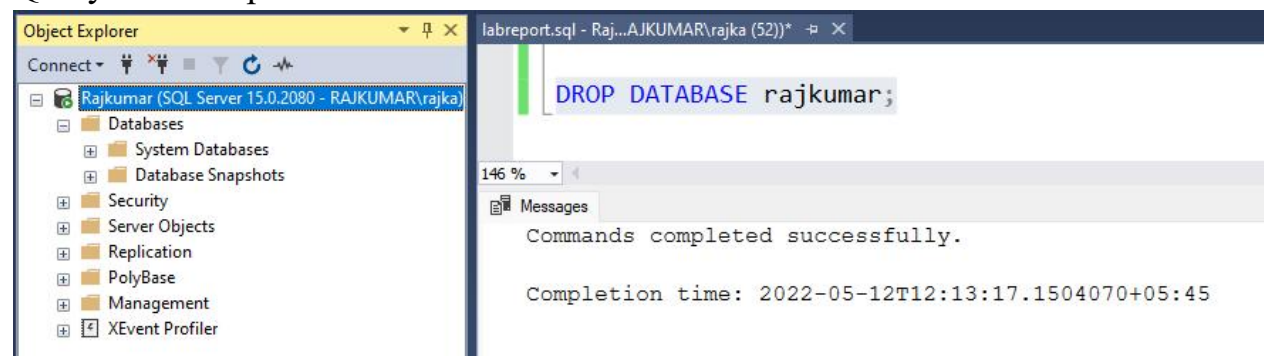
Query with Output:



ii. Syntax: (DROP)

DROP DATABASE db_name;

Query with Output:



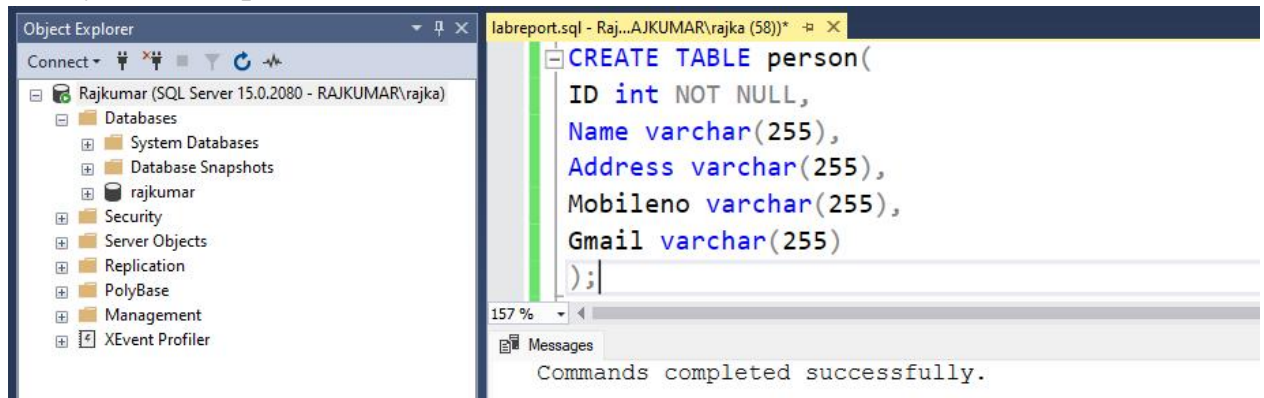
Handwritten Code:

2. Creating and Removing Table Queries

i. Syntax: (CREATE)

CREATE TABLE tb_tablename(
column_name1,
Column_name2...);

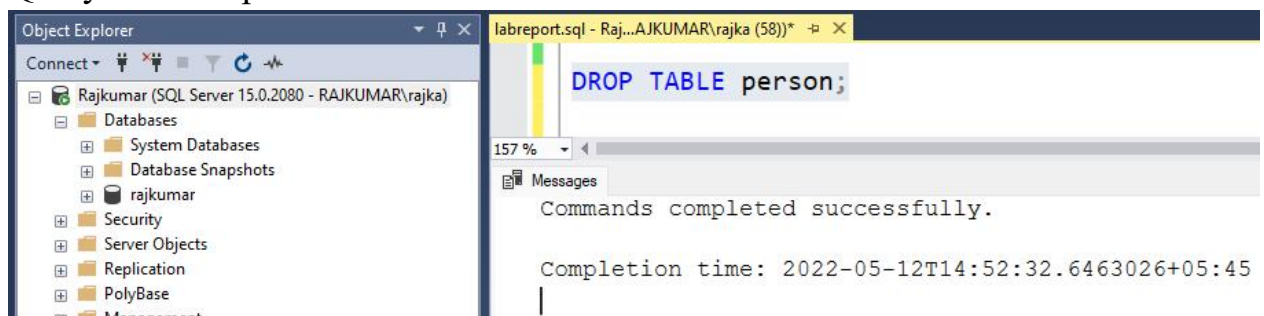
Query with Output:



ii. Syntax: (DROP)

DROP TABLE tb_tablename;

Query with Output:



Handwritten Code:

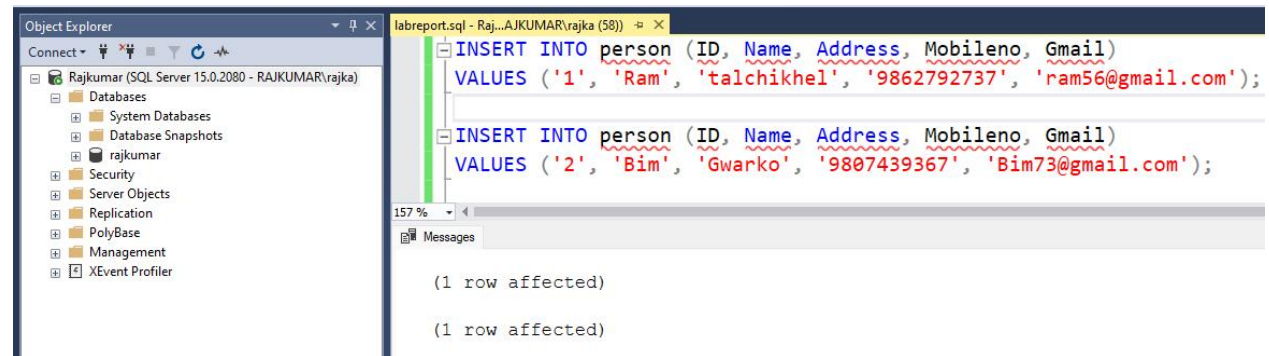
3. Insert into the table, select all, and select distinct from the table.

i. Syntax: (INSERT INTO)

INSERT INTO table_name(column1, column2, column3,...)

VALUES (value1, value2, value3,...);

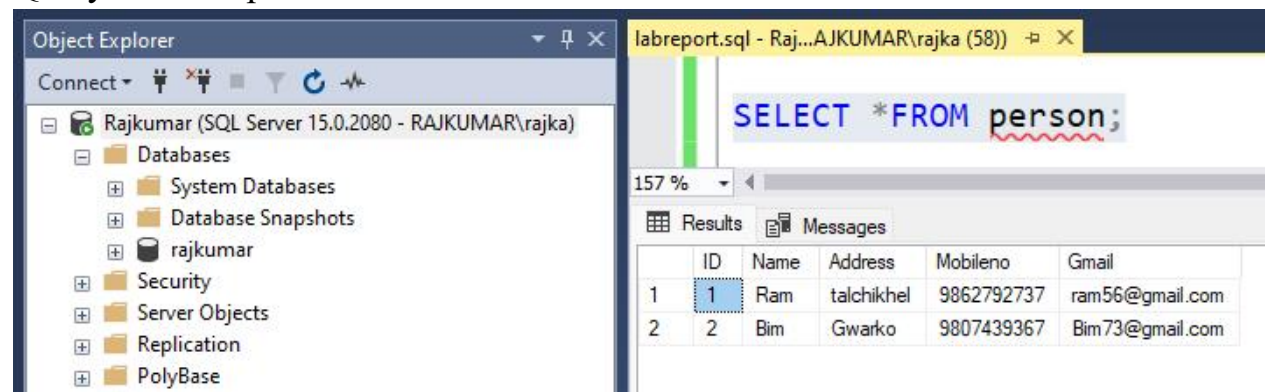
Query with Output:



ii. Syntax: (SELECT)

SELECT * FROM table_name;

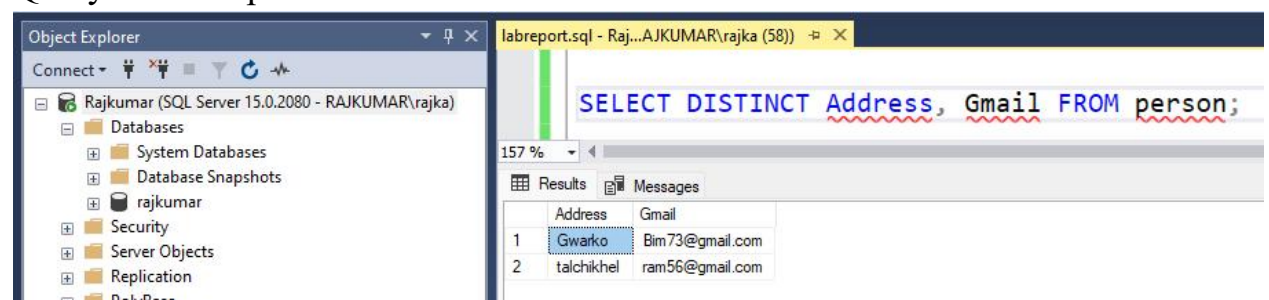
Query with Output:



iii. Syntax: (SELECT DISTINCT)

SELECT DISTINCT column1, column2, FROM table_name;

Query with Output:



Handwritten Code:

4. To show the auto increment in the primary key

i. Syntax: (AUTO INCREMENT)

CREATE TABLE tb_tablename(

Column 1 datatype IDENTITY(starting number, increment by number) PRIMARY KEY,

Column 2,

Column 3....);

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the 'Rajkumar' database. The main query window on the right contains the following SQL code:

```
CREATE TABLE person(  
    PersonID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Address varchar(255),  
    Mobileno varchar(255),  
    Gmail varchar(255)  
);
```

Below the query window, the 'Messages' pane displays the message: 'Commands completed successfully.'

The screenshot shows the SQL Server Enterprise Manager interface. The main query window contains two INSERT statements:

```
INSERT INTO person (Name, Address, Mobileno, Gmail)  
VALUES ('Ram', 'talchikhel', '9862792737', 'ram56@gmail.com');  
  
INSERT INTO person (Name, Address, Mobileno, Gmail)  
VALUES ('Bim', 'Gwarko', '9807439367', 'Bim73@gmail.com');
```

Below the query window, the 'Messages' pane displays the message: '(1 row affected)' twice.

The screenshot shows the SQL Server Enterprise Manager interface. The main query window contains the following SQL code:

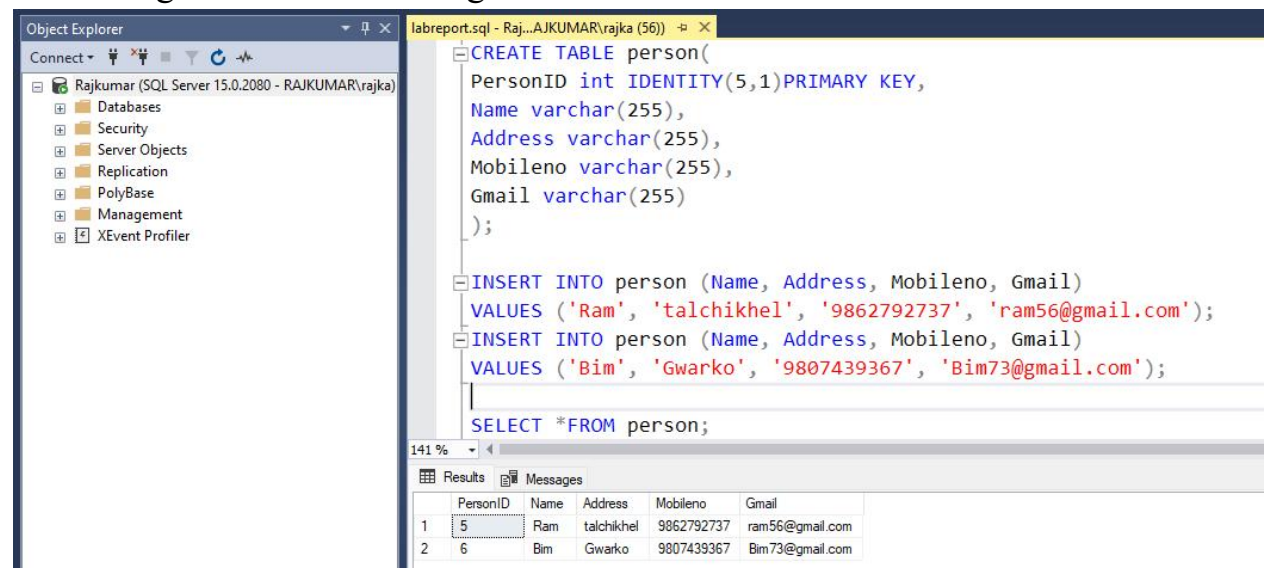
```
SELECT * FROM person;
```

Below the query window, the 'Results' pane displays the output of the query:

	PersonID	Name	Address	Mobileno	Gmail
1	5	Ram	talchikhel	9862792737	ram56@gmail.com
2	6	Bim	Gwarko	9807439367	Bim73@gmail.com

Handwritten Code:

5. Altering the table and adding a column to the table



```
CREATE TABLE person(
    PersonID int IDENTITY(5,1) PRIMARY KEY,
    Name varchar(255),
    Address varchar(255),
    Mobilenumber varchar(255),
    Gmail varchar(255)
);

INSERT INTO person (Name, Address, Mobilenumber, Gmail)
VALUES ('Ram', 'talchikhel', '9862792737', 'ram56@gmail.com');

INSERT INTO person (Name, Address, Mobilenumber, Gmail)
VALUES ('Bim', 'Gwarko', '9807439367', 'Bim73@gmail.com');

SELECT * FROM person;
```

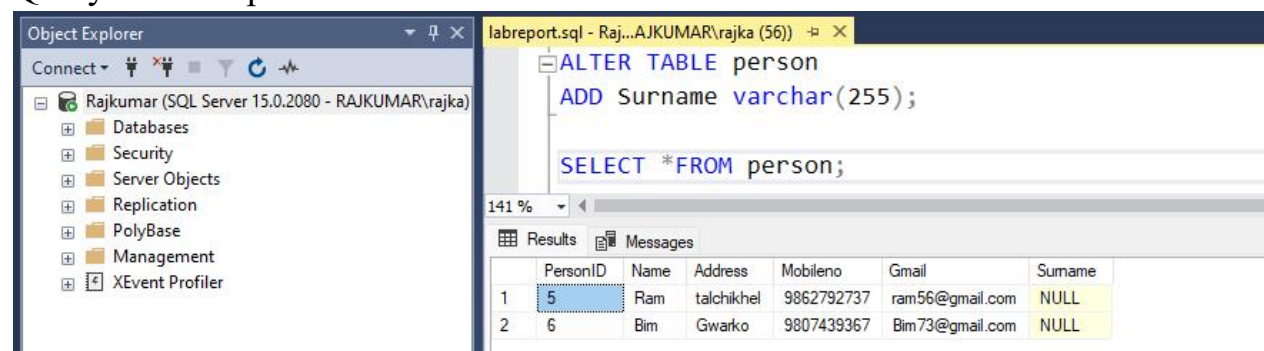
PersonID	Name	Address	Mobilenumber	Gmail
5	Ram	talchikhel	9862792737	ram56@gmail.com
6	Bim	Gwarko	9807439367	Bim73@gmail.com

Syntax:

ALTER TABLE tb_tablename

ADD column_name datatype ;

Query with Output:



```
ALTER TABLE person
ADD Surname varchar(255);

SELECT * FROM person;
```

PersonID	Name	Address	Mobilenumber	Gmail	Surname
5	Ram	talchikhel	9862792737	ram56@gmail.com	NULL
6	Bim	Gwarko	9807439367	Bim73@gmail.com	NULL

Handwritten Code:

6. Declare constraints in the table and remove one of them.

i. Syntax:(Unique)

CREATE TABLE tb_tablename(

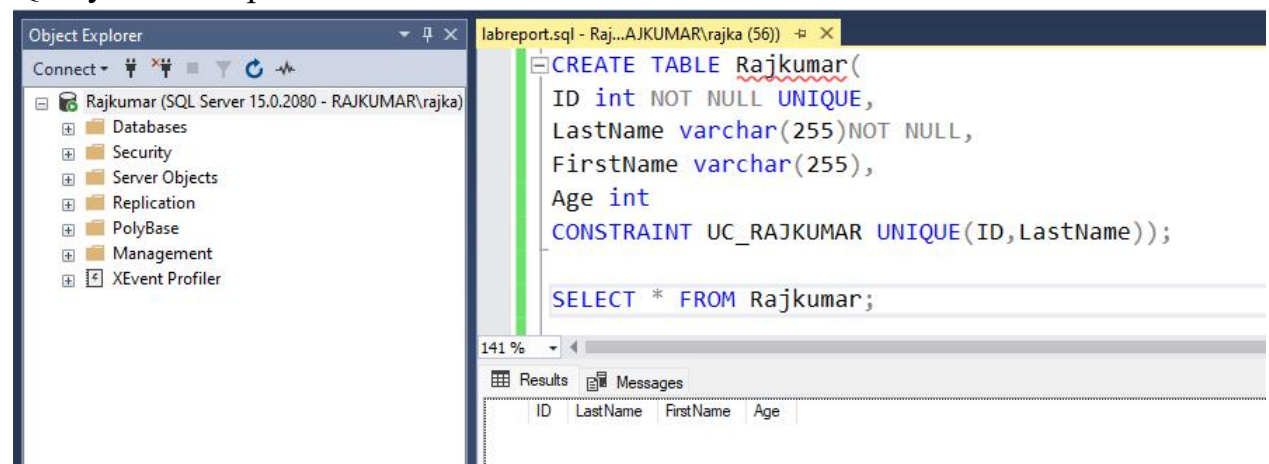
Column 1 datatype NOT NULL UNIQUE,

Column 2 NOT NULL,

Column 3....

CONSTRAINT UC_TABLENAME UNIQUE (column_name,column_name));

Query with Output:

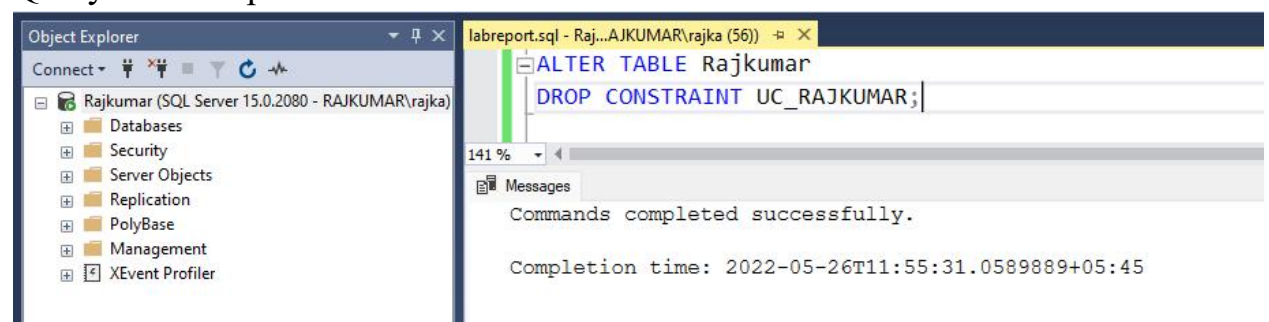


Syntax: (Dropping Unique)

ALTER TABLE table_name

DROP CONSTRAINT UC_TABLENAME;

Query with Output:



ii. Syntax:(Primary Key)

CREATE TABLE tb_tablename(

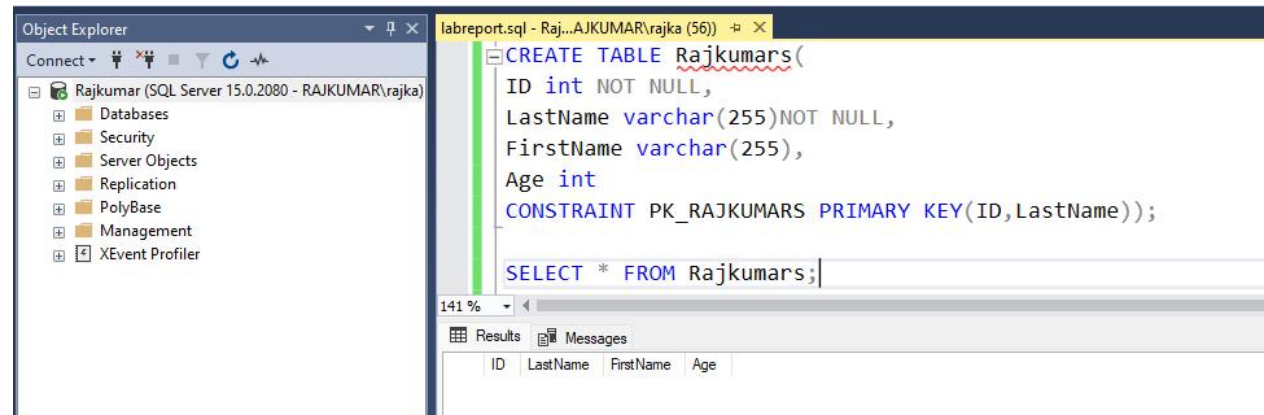
Column 1 datatype NOT NULL UNIQUE,

Column 2 NOT NULL,

Column 3....

CONSTRAINT PK_TABLENAME PRIMARY KEY (column_name,column_name));

Query with Output:



iii. Syntax:(Function Key)

CREATE TABLE tb_tablename(

Column 1 datatype NOT NULL UNIQUE,

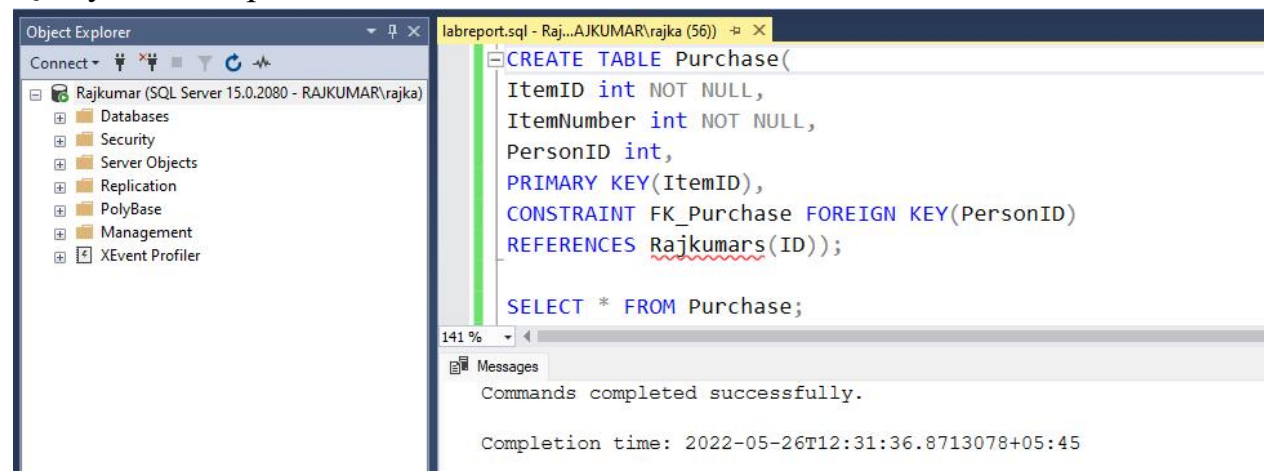
Column 2 NOT NULL,

Column 3....

CONSTRAINT FK_TABLENAME FOREIGN KEY (column_name)

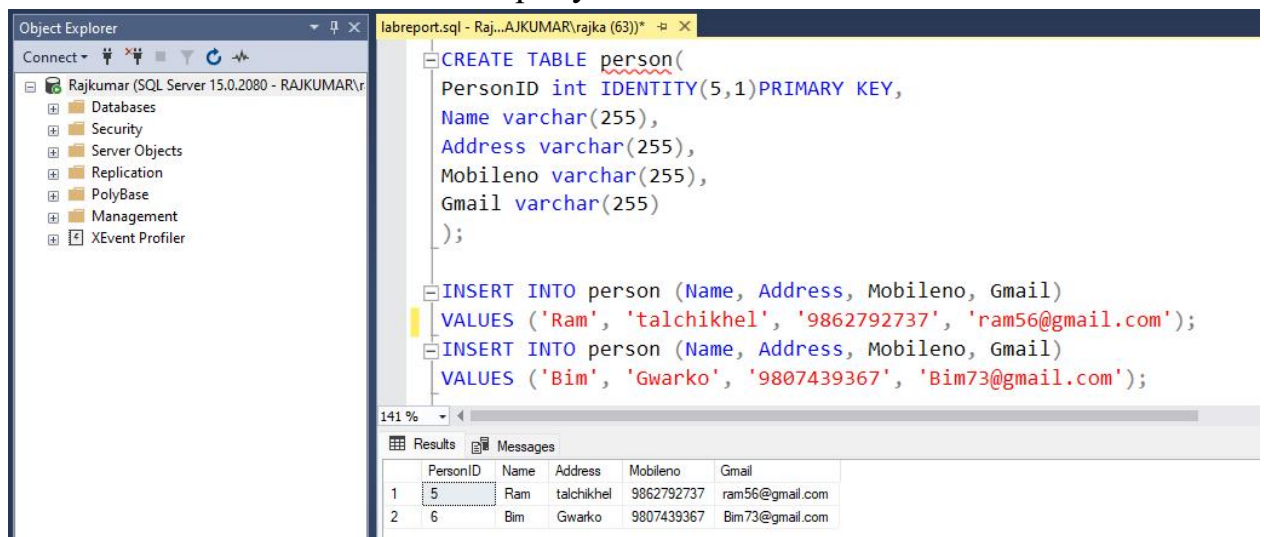
REFERENCES tablename(column_name));

Query with Output:



Handwritten Code:

7. Use the WHERE clause in the query.



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\rajka)'. The query window on the right contains the following SQL code:

```
CREATE TABLE person(  
    PersonID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Address varchar(255),  
    Mobileno varchar(255),  
    Gmail varchar(255)  
);  
  
INSERT INTO person (Name, Address, Mobileno, Gmail)  
VALUES ('Ram', 'talchikhel', '9862792737', 'ram56@gmail.com');  
  
INSERT INTO person (Name, Address, Mobileno, Gmail)  
VALUES ('Bim', 'Gwarko', '9807439367', 'Bim73@gmail.com');
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	PersonID	Name	Address	Mobileno	Gmail
1	5	Ram	talchikhel	9862792737	ram56@gmail.com
2	6	Bim	Gwarko	9807439367	Bim73@gmail.com

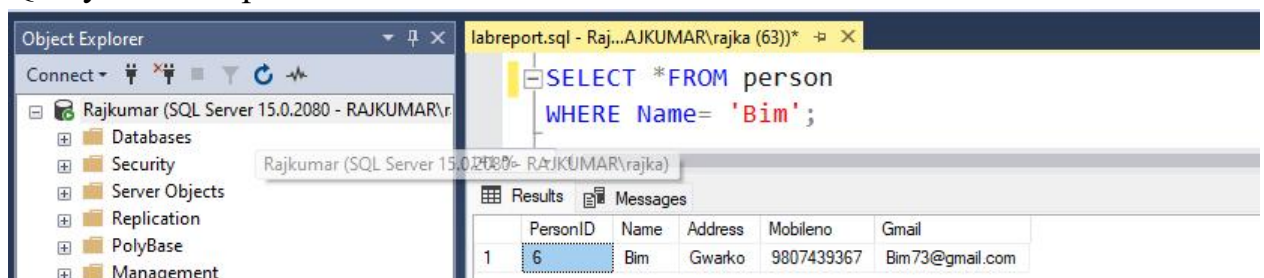
Syntax:

SELECT column1, column2,....

FROM table_name

WHERE condition;

Query with Output:



The screenshot shows the SQL Server Enterprise Manager interface. The query window on the right contains the following SQL code:

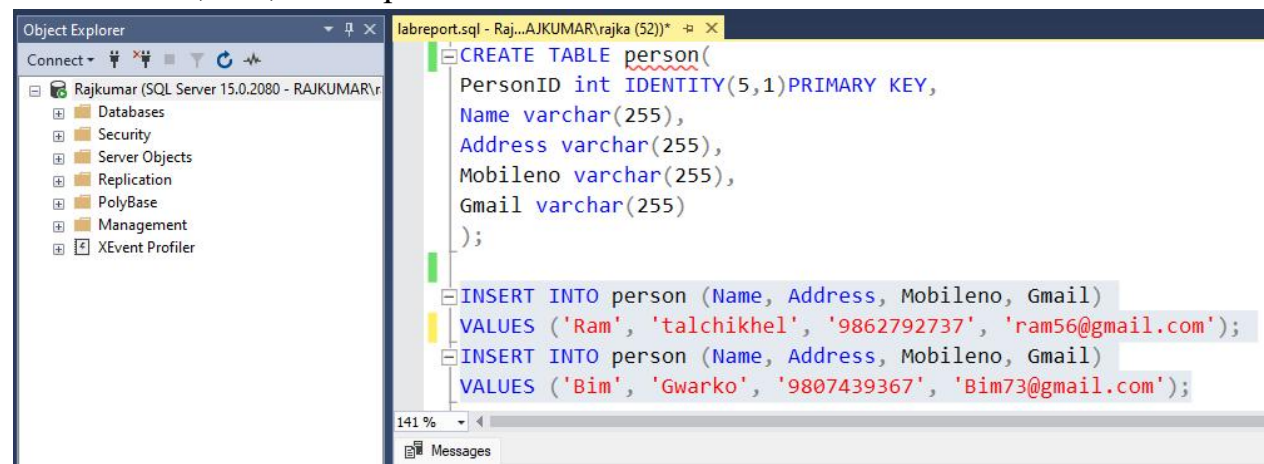
```
SELECT * FROM person  
WHERE Name = 'Bim';
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	PersonID	Name	Address	Mobileno	Gmail
1	6	Bim	Gwarko	9807439367	Bim73@gmail.com

Handwritten Code:

8. Use AND, OR, NOT queries



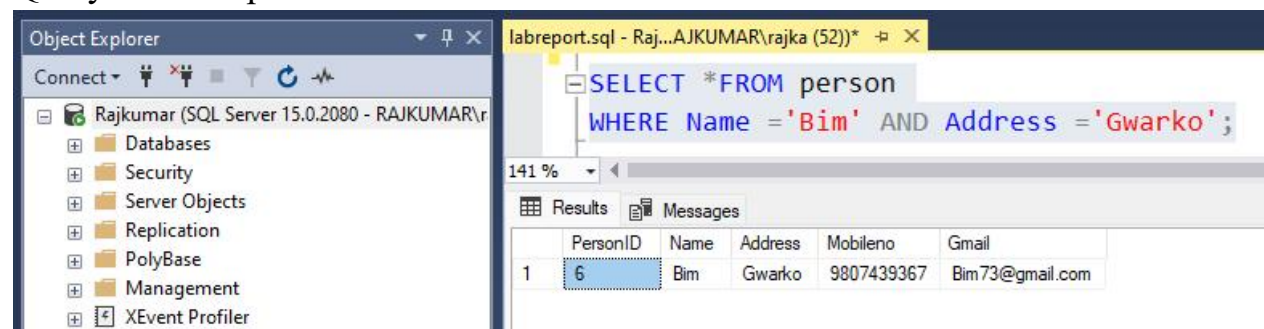
i. Syntax: (AND)

SELECT column1, column2, ...

FROM table_name

WHERE condition1 AND condition2 AND condition3....;

Query with Output:



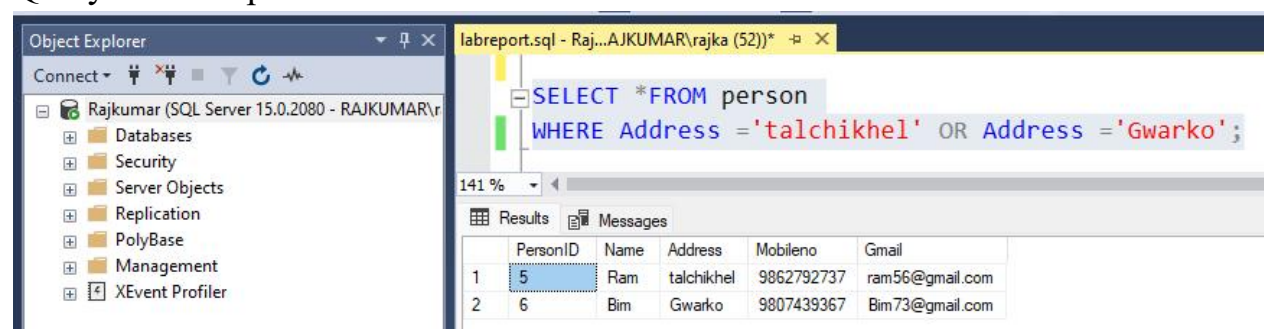
ii. Syntax: (OR)

SELECT column1, column2, ...

FROM table_name

WHERE condition1 OR condition 2 or condition3....;

Query with Output:



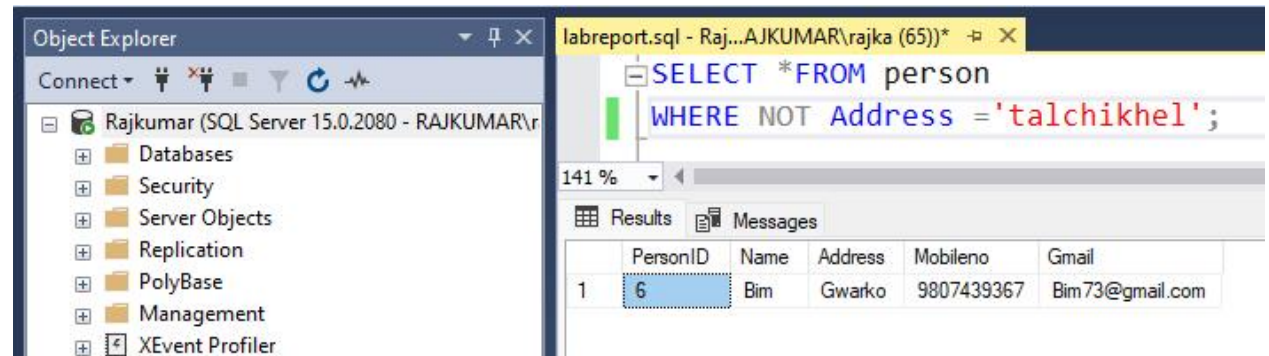
iii. Syntax: (NOT)

SELECT column1, column2, ...

FROM table_name

WHERE NOT condition;

Query with Output:



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane shows the server 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\r)' with folders for Databases, Security, Server Objects, Replication, PolyBase, Management, and XEvent Profiler. The main window shows a query in the 'labreport.sql' file:

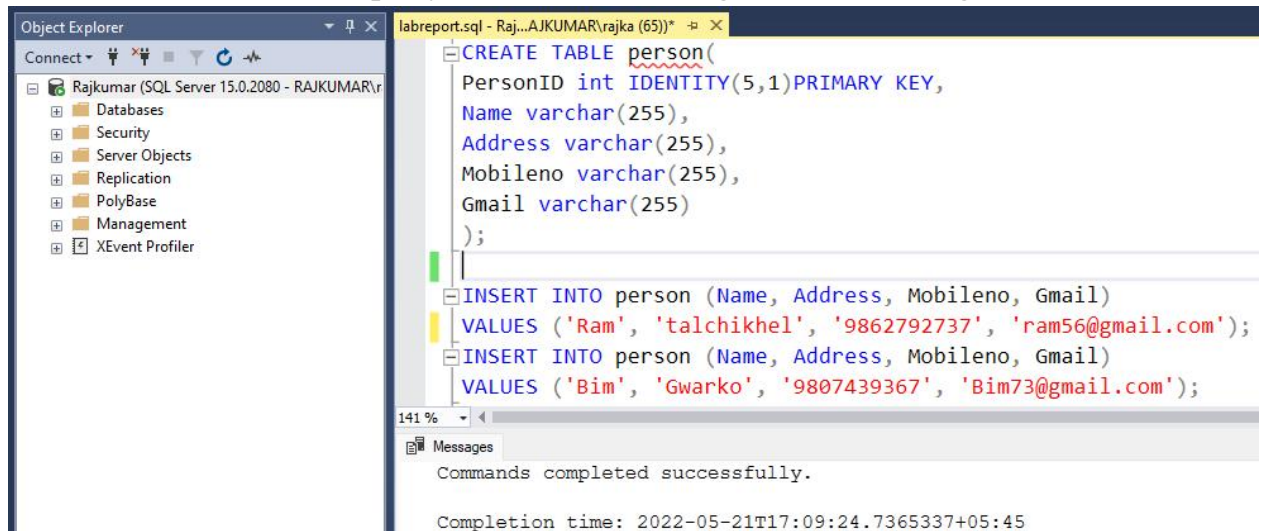
```
SELECT * FROM person  
WHERE NOT Address = 'talchikhel';
```

The query results are displayed in a table with the following columns: PersonID, Name, Address, Mobilen, and Gmail. The results show one record where PersonID is 6, Name is Bim, Address is Gwarko, Mobilen is 9807439367, and Gmail is Bim73@gmail.com.

	PersonID	Name	Address	Mobilen	Gmail
1	6	Bim	Gwarko	9807439367	Bim73@gmail.com

Handwritten Code:

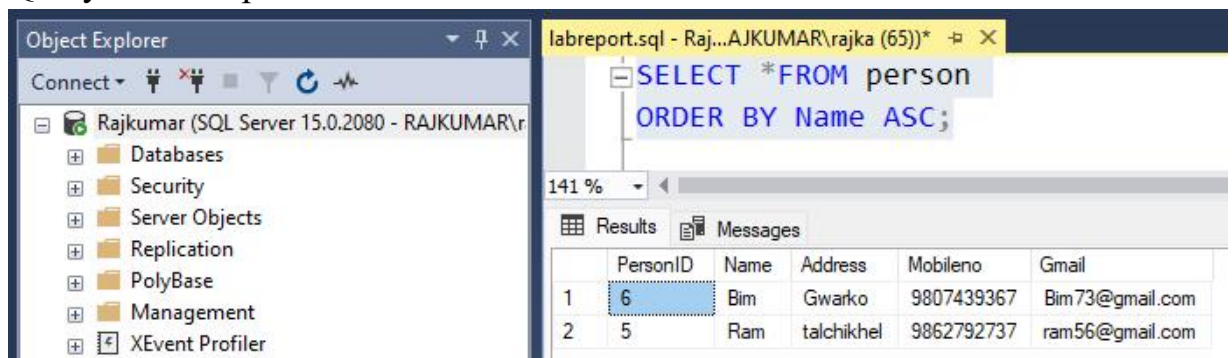
9. Use “ ORDER BY ” query for both ascending and descending orders.



i. Syntax: (Ascending)

SELECT column1, column2,... FROM table_name
ORDER BY column1, column2,...ASC

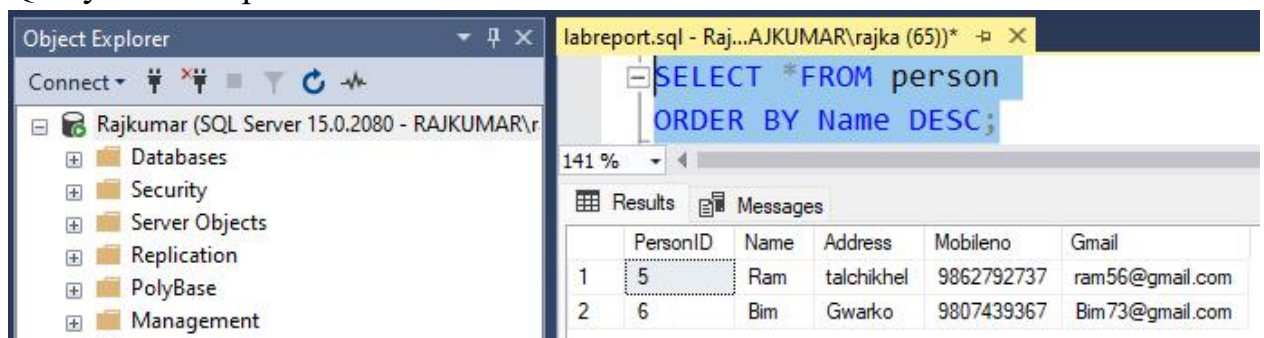
Query with Output:



ii. Syntax: (Descending)

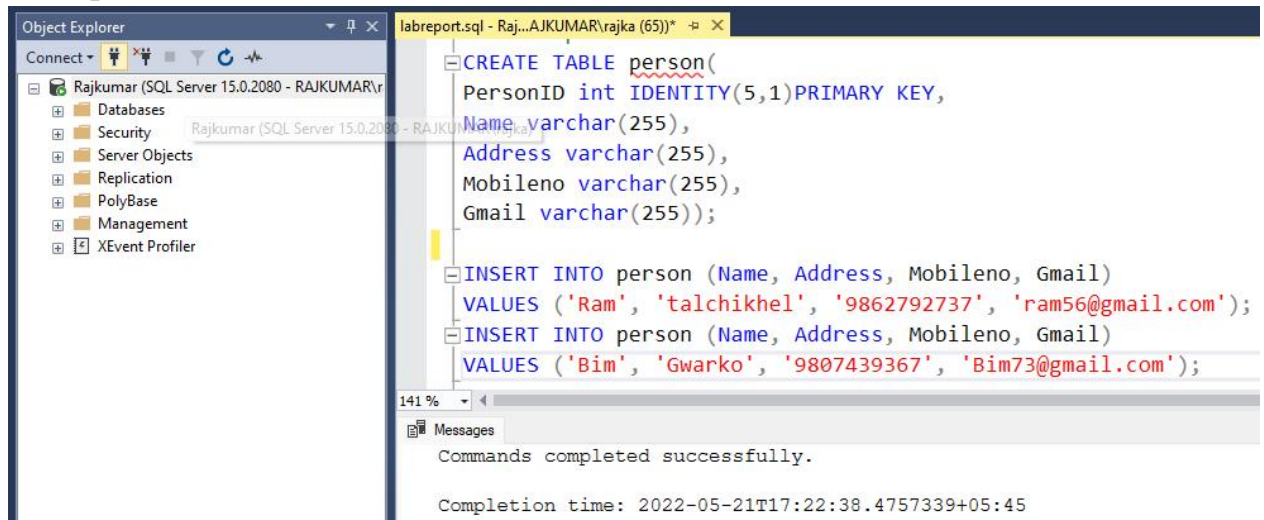
SELECT column1, column2,... FROM table_name
ORDER BY column1, column2,...DESC;

Query with Output:



Handwritten Code:

10. Update and delete table fields.



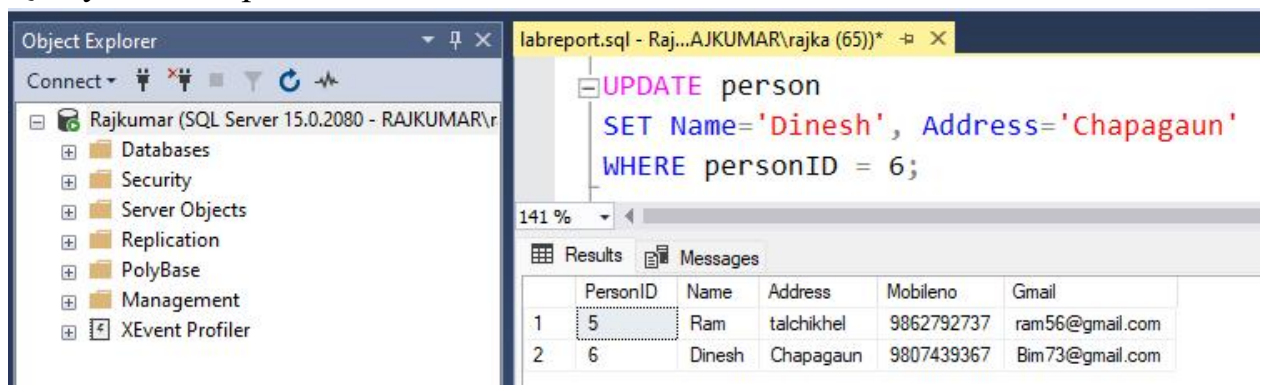
i. Syntax: (UPDATE)

UPDATE table_name

SET column1= value1, column2=value2,.....

WHERE condition;

Query with Output:

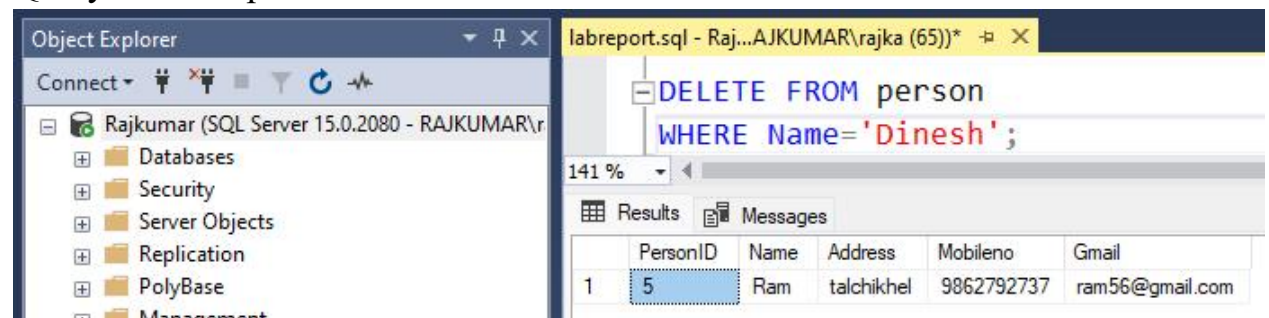


ii. Syntax: (DELETE)

DELETE FROM table_name

WHERE condition;

Query with Output:



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\r)'. The main window shows a query editor with the following SQL code:

```
DELETE FROM person  
WHERE Name='Dinesh';
```

Below the query editor, the 'Results' tab is active, displaying a table with the following data:

	PersonID	Name	Address	Mobileno	Gmail
1	5	Ram	talchikhel	9862792737	ram56@gmail.com

Handwritten Code:

11. Use Join queries

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the 'Rajkumar' database. The main pane shows a SQL script titled 'labreport.sql - Raj...AJKUMAR\rajka (63)'. The script defines a 'Country' table with 'CountryId' as the primary key and 'CountryName' as a varchar(255). It then inserts three rows: China, India, and Nepal. A 'SELECT * FROM Country;' query is executed, and the results are displayed in the 'Results' pane below the script editor.

```
---Country:
CREATE TABLE Country(
  CountryId int IDENTITY(1,1) PRIMARY KEY,
  CountryName varchar(255)
);

INSERT INTO Country(CountryName)
VALUES('China');
INSERT INTO Country(CountryName)
VALUES('India');
INSERT INTO Country(CountryName)
VALUES('Nepal');

SELECT * FROM Country;
```

CountryId	CountryName
1	China
2	India
3	Nepal

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the 'Rajkumar' database. The main pane shows a SQL script titled 'labreport.sql - Raj...AJKUMAR\rajka (63)'. The script defines a 'State' table with 'StateId' as the primary key, 'CountryId' as a foreign key, and 'StateName' as a varchar(255). It then inserts three rows: Benjing (CountryId 1), NewDelhi (CountryId 1), and Kathmand (CountryId 2). A 'SELECT * FROM State;' query is executed, and the results are displayed in the 'Results' pane below the script editor.

```
---State:
CREATE TABLE State(
  StateId int IDENTITY(1,1) PRIMARY KEY,
  CountryId int,
  StateName varchar(255)
);

INSERT INTO State(CountryId, StateName)
VALUES(1, 'Benjing');
INSERT INTO State(CountryId, StateName)
VALUES(1, 'NewDelhi');
INSERT INTO State(CountryId, StateName)
VALUES(2, 'Kathmand');

SELECT * FROM State;
```

StateId	CountryId	StateName
1	1	Benjing
2	1	NewDelhi
3	2	Kathmand

Object Explorer: Connect, Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\), Databases, Security, Server Objects, Replication, PolyBase, Management, XEvent Profiler

labreport.sql - Raj...AJKUMAR\rajka (63))

```

---People:
CREATE TABLE People(
  PeopleID int IDENTITY(1, 1) PRIMARY KEY,
  StateId int,
  PeopleName varchar(255),
  Wages int);

INSERT INTO People(StateId, PeopleName, Wages)
VALUES(1, 'Ram', 52000);
INSERT INTO People(StateId, PeopleName, Wages)
VALUES(2, 'Prabin', 35000);
INSERT INTO People(StateId, PeopleName, Wages)
VALUES(3, 'Yam', 62000);

SELECT * FROM People;

```

141 %

Results Messages

	PeopleID	StateId	PeopleName	Wages
1	1	1	Ram	52000
2	2	2	Prabin	35000
3	3	3	Yam	62000

i. Syntax: (INNER JOIN)

SELECT column_name(s) FROM table1

INNER JOIN table2

ON table1.column_name = table2.column_name;

Query with Output:

Object Explorer: Connect, Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\), Databases, Security, Server Objects, Replication, PolyBase, Management, XEvent Profiler

labreport.sql - Raj...AJKUMAR\rajka (63))

```

SELECT * FROM Country
INNER JOIN State
ON Country.CountryId=State.CountryId

```

141 %

Results Messages

	CountryId	CountryName	StateId	CountryId	StateName
1	1	China	1	1	Benjing
2	1	China	2	1	NewDelhi
3	2	India	3	2	Kathmand

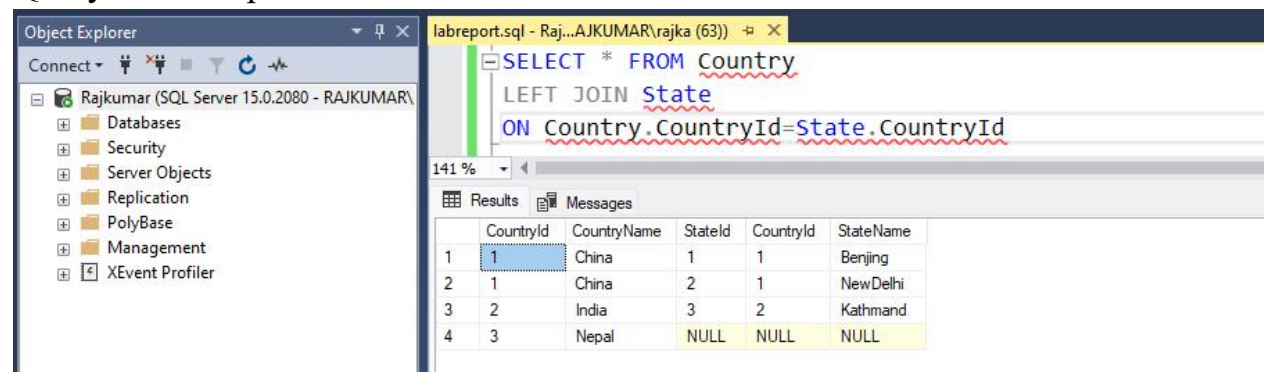
ii. Syntax: (LEFT JOIN)

SELECT column_name(s) FROM table1

LEFT JOIN table2

ON table1.column_name = table2.column_name;

Query with Output:



labreport.sql - Raj...AJKUMAR\rajka (63)

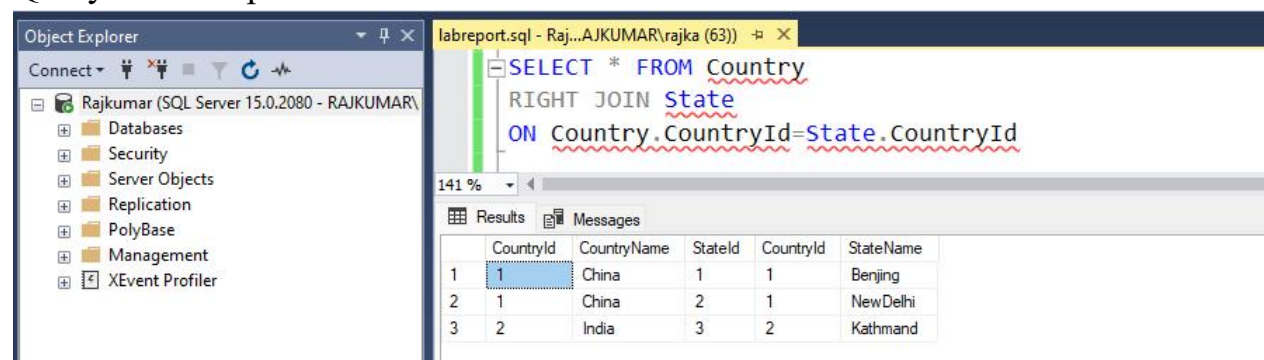
```
SELECT * FROM Country
LEFT JOIN State
ON Country.CountryId=State.CountryId
```

	CountryId	CountryName	StateId	CountryId	StateName
1	1	China	1	1	Benjing
2	1	China	2	1	NewDelhi
3	2	India	3	2	Kathmand
4	3	Nepal	NULL	NULL	NULL

iii. Syntax: (RIGHT JOIN)

SELECT column_name(s) FROM table1
 RIGHT JOIN table2
 ON table1.column_name = table2.column_name;

Query with Output:



labreport.sql - Raj...AJKUMAR\rajka (63)

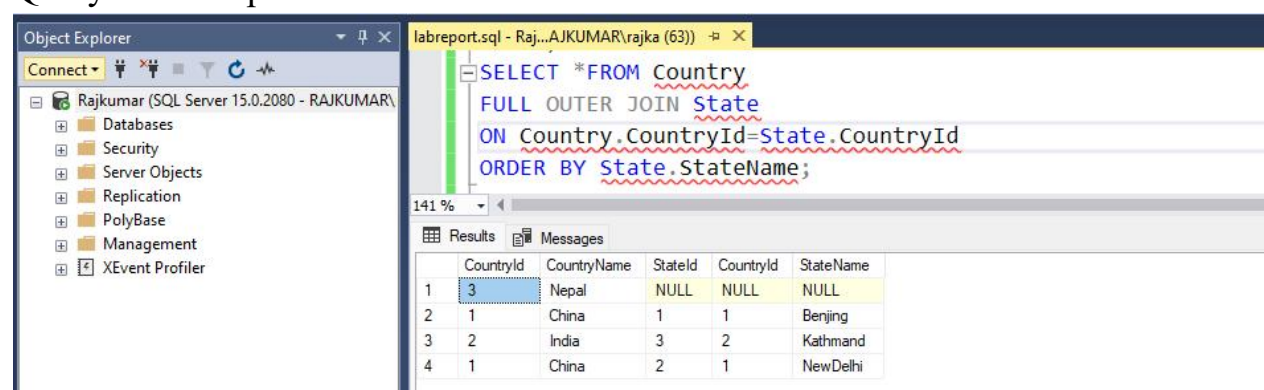
```
SELECT * FROM Country
RIGHT JOIN State
ON Country.CountryId=State.CountryId
```

	CountryId	CountryName	StateId	CountryId	StateName
1	1	China	1	1	Benjing
2	1	China	2	1	NewDelhi
3	2	India	3	2	Kathmand

iv. Syntax: (FULL OUTER JOIN)

SELECT column_name(s) FROM table1
 FULL OUTER JOIN table2
 ON table1.column_name = table2.column_name;
 WHERE condition;

Query with Output:



labreport.sql - Raj...AJKUMAR\rajka (63)

```
SELECT * FROM Country
FULL OUTER JOIN State
ON Country.CountryId=State.CountryId
ORDER BY State.StateName;
```

	CountryId	CountryName	StateId	CountryId	StateName
1	3	Nepal	NULL	NULL	NULL
2	1	China	1	1	Benjing
3	2	India	3	2	Kathmand
4	1	China	2	1	NewDelhi

Handwritten Code:

12. Use the MIN and MAX functions.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\rajka (65))'. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
CREATE TABLE trade(  
    ProductID int IDENTITY(1,1) PRIMARY KEY,  
    Name varchar(255),  
    Price int);  
  
INSERT INTO trade (Name, Price)  
VALUES ('Rice', '1500');  
  
INSERT INTO trade (Name, Price)  
VALUES ('Chocolate', '100');  
  
INSERT INTO trade (Name, Price)  
VALUES ('Biscuit', '50');
```

Below the script, the 'Results' tab shows the output of the insert statements:

	ProductID	Name	Price
1	1	Rice	1500
2	2	Chocolate	100
3	3	Biscuit	50

i. Syntax: (MIN Function)

SELECT MIN(column_name)

FROM table_name

WHERE condition;

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
SELECT MIN(Price) AS SmallestPrice  
FROM trade;
```

Below the script, the 'Results' tab shows the output of the query:

	SmallestPrice
1	50

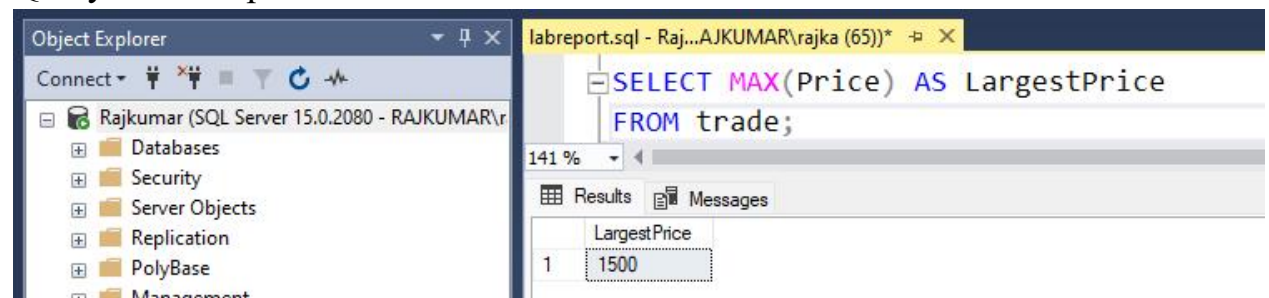
ii. Syntax: (MAX Function)

SELECT MAX(column_name)

FROM table_name

WHERE condition;

Query with Output:



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane shows the server 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\ra)' with folders for Databases, Security, Server Objects, Replication, PolyBase, and Management. The main query editor window, titled 'labreport.sql - Raj...AJKUMAR\rajka (65))', contains the following SQL query:

```
SELECT MAX(Price) AS LargestPrice  
FROM trade;
```

Below the query editor, the 'Results' pane shows the output of the query. It displays a single row with the value 1500 under the column header 'LargestPrice'.

	LargestPrice
1	1500

Handwritten Code:

13. Use the COUNT, AVERAGE, and SUM functions.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\r)'. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
CREATE TABLE trade(  
    ProductID int IDENTITY(1,1) PRIMARY KEY,  
    Name varchar(255),  
    Price int);  
  
INSERT INTO trade (Name, Price)  
VALUES ('Rice', '1500');  
INSERT INTO trade (Name, Price)  
VALUES ('Chocolate', '100');  
INSERT INTO trade (Name, Price)  
VALUES ('Biscuit', '50');
```

Below the script, the 'Results' tab shows the output of the insert statements:

	ProductID	Name	Price
1	1	Rice	1500
2	2	Chocolate	100
3	3	Biscuit	50

i. Syntax: (COUNT)

SELECT COUNT (column_name)

FROM table_name

WHERE condition;

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
SELECT COUNT(Price)  
FROM trade;
```

Below the script, the 'Results' tab shows the output of the query:

	(No column name)
1	3

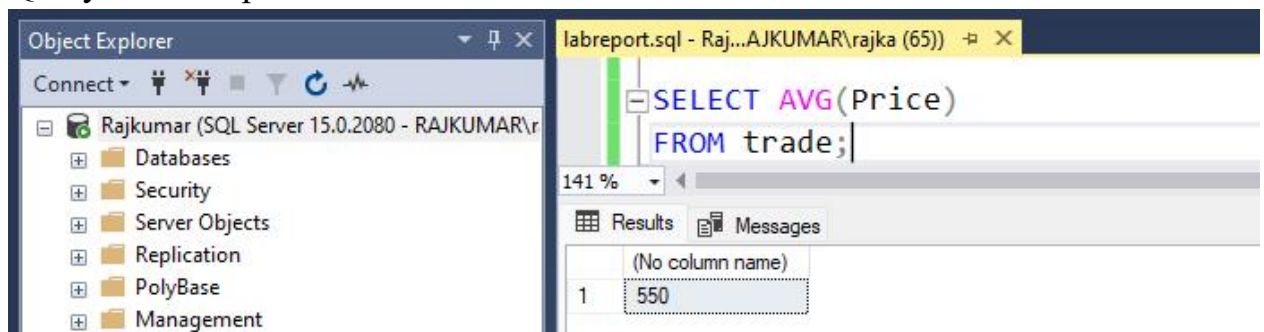
ii. Syntax: (AVERAGE)

SELECT AVG(column_name)

FROM table_name

WHERE condition;

Query with Output:



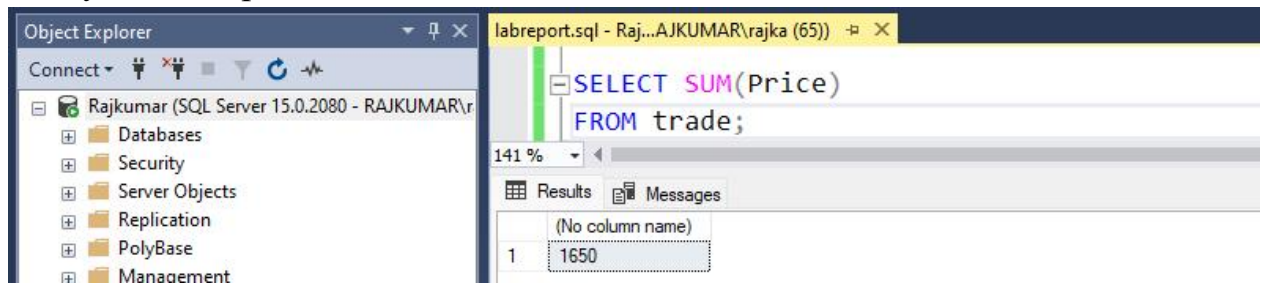
iii. Syntax: (SUM)

SELECT SUM (column_name)

FROM table_name

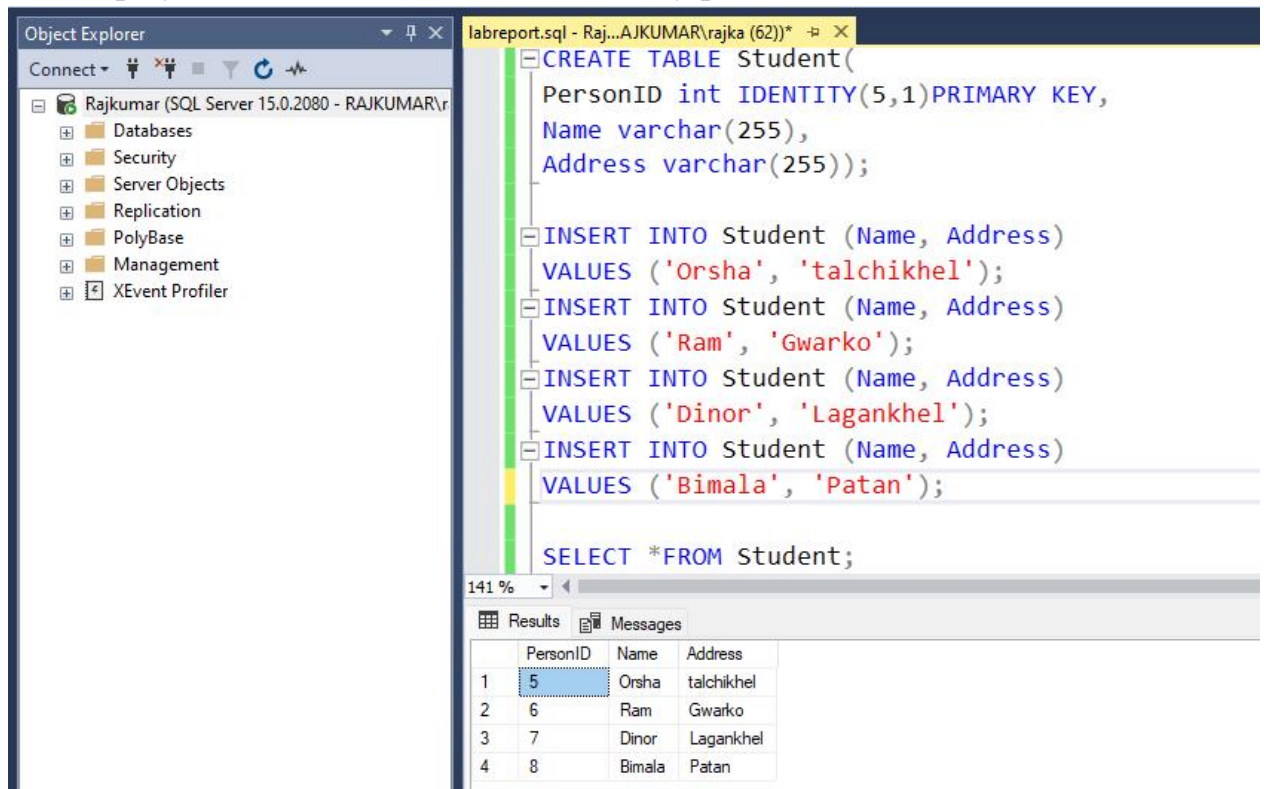
WHERE condition;

Query with Output:



Handwritten Code:

14. Display students' names with "OR" in any position.



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the 'Rajkumar' server selected. The right pane shows a query window titled 'labreport.sql - Raj...AJKUMAR\rajka (62))' containing the following SQL code:

```
CREATE TABLE Student(  
    PersonID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Address varchar(255));  
  
INSERT INTO Student (Name, Address)  
VALUES ('Orsha', 'talchikhel');  
INSERT INTO Student (Name, Address)  
VALUES ('Ram', 'Gwarko');  
INSERT INTO Student (Name, Address)  
VALUES ('Dinor', 'Lagankhel');  
INSERT INTO Student (Name, Address)  
VALUES ('Bimala', 'Patan');  
  
SELECT * FROM Student;
```

The Results pane shows the output of the query, displaying a table with 4 rows and 4 columns: PersonID, Name, and Address.

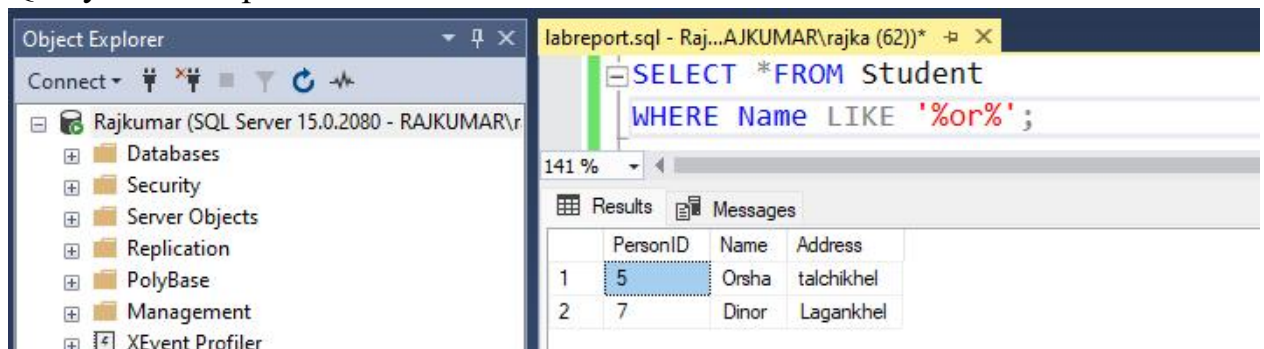
	PersonID	Name	Address
1	5	Orsha	talchikhel
2	6	Ram	Gwarko
3	7	Dinor	Lagankhel
4	8	Bimala	Patan

Syntax:

SELECT * FROM table_name

WHERE column_name LIKE '%a%';

Query with Output:



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the Object Explorer with the 'Rajkumar' server selected. The right pane shows a query window titled 'labreport.sql - Raj...AJKUMAR\rajka (62))' containing the following SQL code:

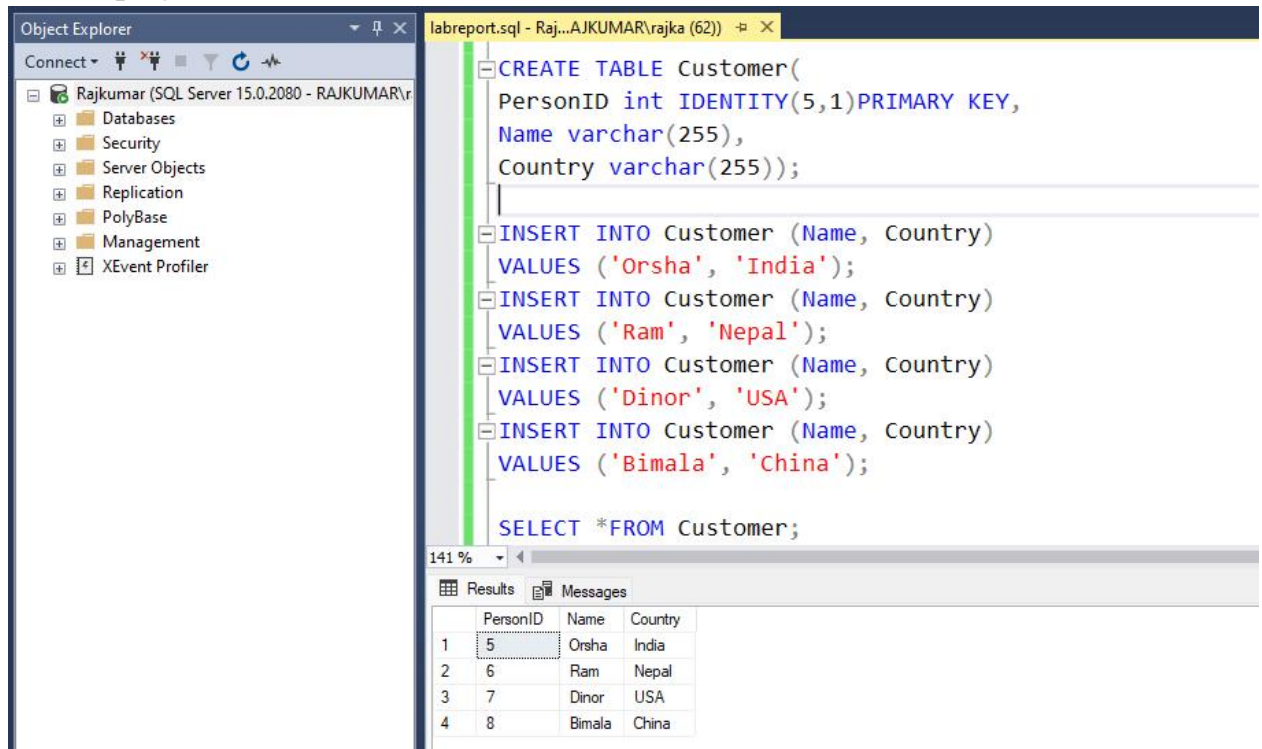
```
SELECT * FROM Student  
WHERE Name LIKE '%or%';
```

The Results pane shows the output of the query, displaying a table with 2 rows and 4 columns: PersonID, Name, and Address.

	PersonID	Name	Address
1	5	Orsha	talchikhel
2	7	Dinor	Lagankhel

Handwritten Code:

15. Display all customers that are not located in India and the USA.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\...)'. The main window shows a query script in 'labreport.sql' with the following SQL code:

```
CREATE TABLE Customer(  
    PersonID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Country varchar(255));  
  
INSERT INTO Customer (Name, Country)  
VALUES ('Orsha', 'India');  
INSERT INTO Customer (Name, Country)  
VALUES ('Ram', 'Nepal');  
INSERT INTO Customer (Name, Country)  
VALUES ('Dinor', 'USA');  
INSERT INTO Customer (Name, Country)  
VALUES ('Bimala', 'China');  
  
SELECT * FROM Customer;
```

The query results are displayed in a table with the following data:

	PersonID	Name	Country
1	5	Orsha	India
2	6	Ram	Nepal
3	7	Dinor	USA
4	8	Bimala	China

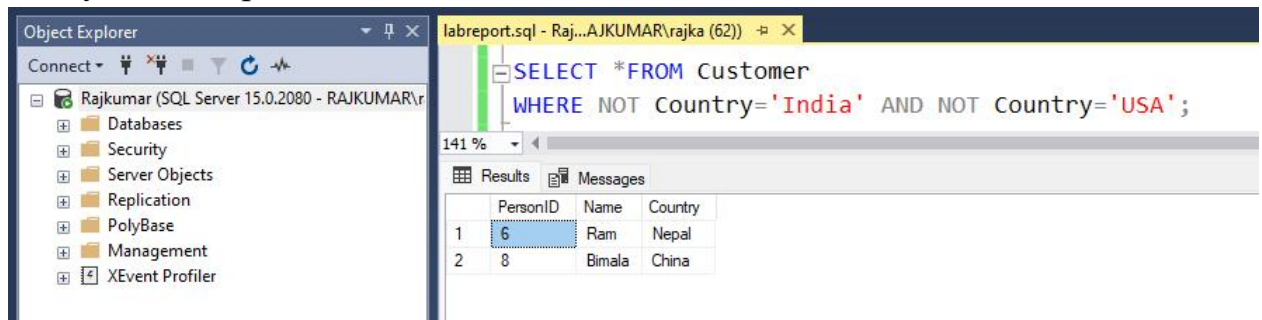
Syntax:

SELECT column1, column2,

FROM table_name

WHERE NOT condition ;

Query with Output:



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\...)'. The main window shows a query script in 'labreport.sql' with the following SQL code:

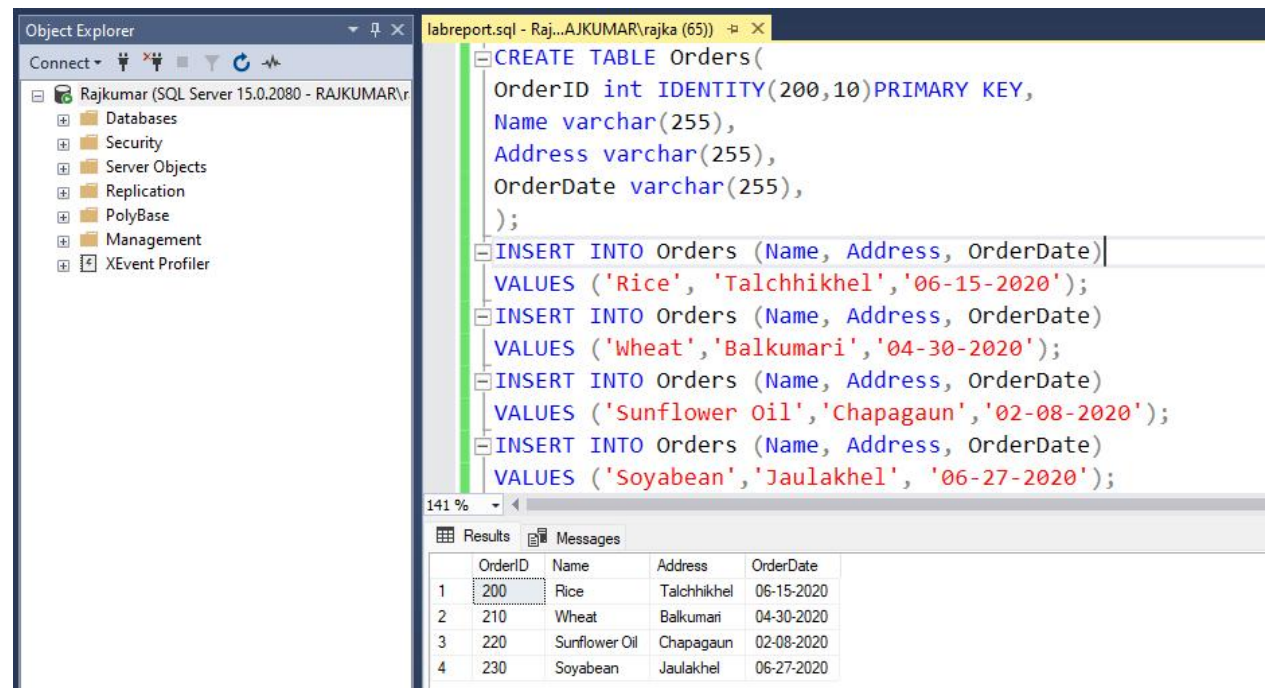
```
SELECT * FROM Customer  
WHERE NOT Country='India' AND NOT Country='USA';
```

The query results are displayed in a table with the following data:

	PersonID	Name	Country
1	6	Ram	Nepal
2	8	Bimala	China

Handwritten Code:

16. Display all orders with an OrderDate BETWEEN '06-01-2020' AND '06-31-2020'



```

CREATE TABLE Orders(
  OrderID int IDENTITY(200,10) PRIMARY KEY,
  Name varchar(255),
  Address varchar(255),
  OrderDate varchar(255),
);
INSERT INTO Orders (Name, Address, OrderDate)
VALUES ('Rice', 'Talchhikhel', '06-15-2020');
INSERT INTO Orders (Name, Address, OrderDate)
VALUES ('Wheat', 'Balkumari', '04-30-2020');
INSERT INTO Orders (Name, Address, OrderDate)
VALUES ('Sunflower Oil', 'Chapagaun', '02-08-2020');
INSERT INTO Orders (Name, Address, OrderDate)
VALUES ('Soyabean', 'Jaulakhel', '06-27-2020');

```

	OrderID	Name	Address	OrderDate
1	200	Rice	Talchhikhel	06-15-2020
2	210	Wheat	Balkumari	04-30-2020
3	220	Sunflower Oil	Chapagaun	02-08-2020
4	230	Soyabean	Jaulakhel	06-27-2020

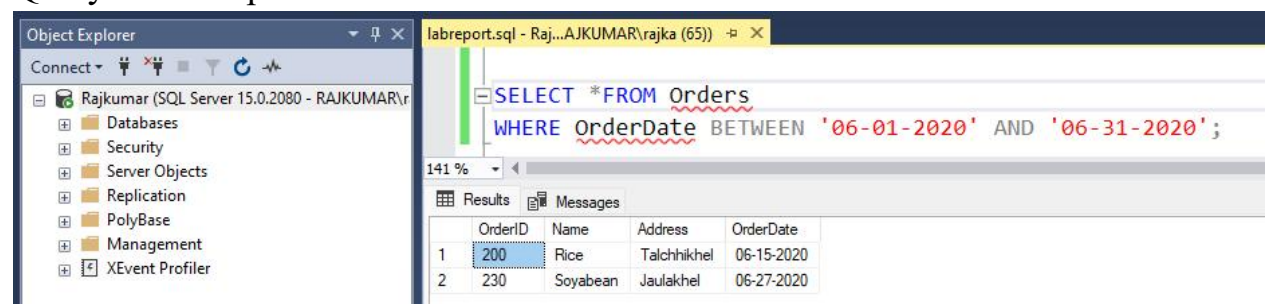
Syntax:

SELECT column_name(s)

FROM table_name

WHERE column_name BETWEEN value1 AND value2;

Query with Output:



```

SELECT * FROM Orders
WHERE OrderDate BETWEEN '06-01-2020' AND '06-31-2020';

```

	OrderID	Name	Address	OrderDate
1	200	Rice	Talchhikhel	06-15-2020
2	230	Soyabean	Jaulakhel	06-27-2020

Handwritten Code:

17. Count the number of customers in each country.

The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\rajka)'. The main window shows a SQL script in 'labreport.sql' with the following code:

```
CREATE TABLE Customer(  
    CustomerID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Country varchar(255));  
  
INSERT INTO Customer (Name, Country)  
VALUES ('Orsha', 'India');  
INSERT INTO Customer (Name, Country)  
VALUES ('Ram', 'Nepal');  
INSERT INTO Customer (Name, Country)  
VALUES ('Dinor', 'Nepal');  
INSERT INTO Customer (Name, Country)  
VALUES ('Bimala', 'India');  
INSERT INTO Customer (Name, Country)  
VALUES ('Anish', 'India');  
  
SELECT * FROM Customer;
```

The Results pane at the bottom shows the output of the SELECT query:

	CustomerID	Name	Country
1	5	Orsha	India
2	6	Ram	Nepal
3	7	Dinor	Nepal
4	8	Bimala	India
5	9	Anish	India

Syntax:

```
SELECT COUNT (column_name)  
FROM table_name  
GROUP BY column_name;
```

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. The main window shows a SQL script in 'labreport.sql' with the following code:

```
SELECT COUNT(CustomerID), Country  
FROM Customer  
GROUP BY Country;
```

The Results pane at the bottom shows the output of the query:

	(No column name)	Country
1	3	India
2	2	Nepal

Handwritten Code:

18. Create a view to display all tourists from China.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\rajka)'. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
CREATE TABLE Tourists(  
    CustomerID int IDENTITY(5,1) PRIMARY KEY,  
    Name varchar(255),  
    Country varchar(255));  
  
INSERT INTO Tourists (Name, Country)  
VALUES ('Orsha', 'China');  
  
INSERT INTO Tourists (Name, Country)  
VALUES ('Ram', 'Nepal');  
  
INSERT INTO Tourists (Name, Country)  
VALUES ('Dinor', 'China');  
  
INSERT INTO Tourists (Name, Country)  
VALUES ('Bimala', 'India');  
  
INSERT INTO Tourists (Name, Country)  
VALUES ('Anish', 'China');  
  
SELECT * FROM Tourists;
```

Below the script, the 'Results' tab shows the output of the SELECT statement:

	CustomerID	Name	Country
1	5	Orsha	China
2	6	Ram	Nepal
3	7	Dinor	China
4	8	Bim...	India
5	9	Anish	China

Syntax:

```
CREATE VIEW view_name AS  
SELECT column1, column2, .....  
FROM table_name  
WHERE condition;
```

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\rajka)'. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
CREATE VIEW [China Tourists] AS  
SELECT TouristName, Country  
FROM Tourists  
WHERE Country = 'China';  
  
SELECT * FROM [China Tourists];
```

Below the script, the 'Results' tab shows the output of the SELECT statement:

	TouristName	Country
1	Orsha	China
2	Dinor	China
3	Anish	China

Handwritten Code:

19. Create a stored procedure named "SelectAllStudents" that selects all records from the "Students" table.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\ra)'. The main pane shows a script titled 'labreport.sql - Raj...AJKUMAR\rajka (65))' with the following SQL code:

```
CREATE TABLE Students(  
    StudentID int IDENTITY(100,11) PRIMARY KEY,  
    Name varchar(255),  
    Address varchar(255),  
    RollNO varchar(255));  
  
INSERT INTO Students (Name, Address, RollNo)  
VALUES ('Ram', 'Talchhikhel', '4006');  
  
INSERT INTO Students (Name, Address, RollNo)  
VALUES ('Logan', 'Patan', '4007');  
  
INSERT INTO Students (Name, Address, RollNo)  
VALUES ('Sandeep', 'Gwarko', '4008');
```

Below the script, the 'Results' pane shows the output of the insert statements as a table:

	StudentID	Name	Address	RollNO
1	100	Ram	Talchhikhel	4006
2	111	Logan	Patan	4007
3	122	Sandeep	Gwarko	4008

Syntax:

CREATE PROCEDURE procedure_name

AS

sql_statement

GO;

Query with Output:

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\ra)'. The main pane shows a script titled 'labreport.sql - Raj...AJKUMAR\rajka (65))' with the following SQL code:

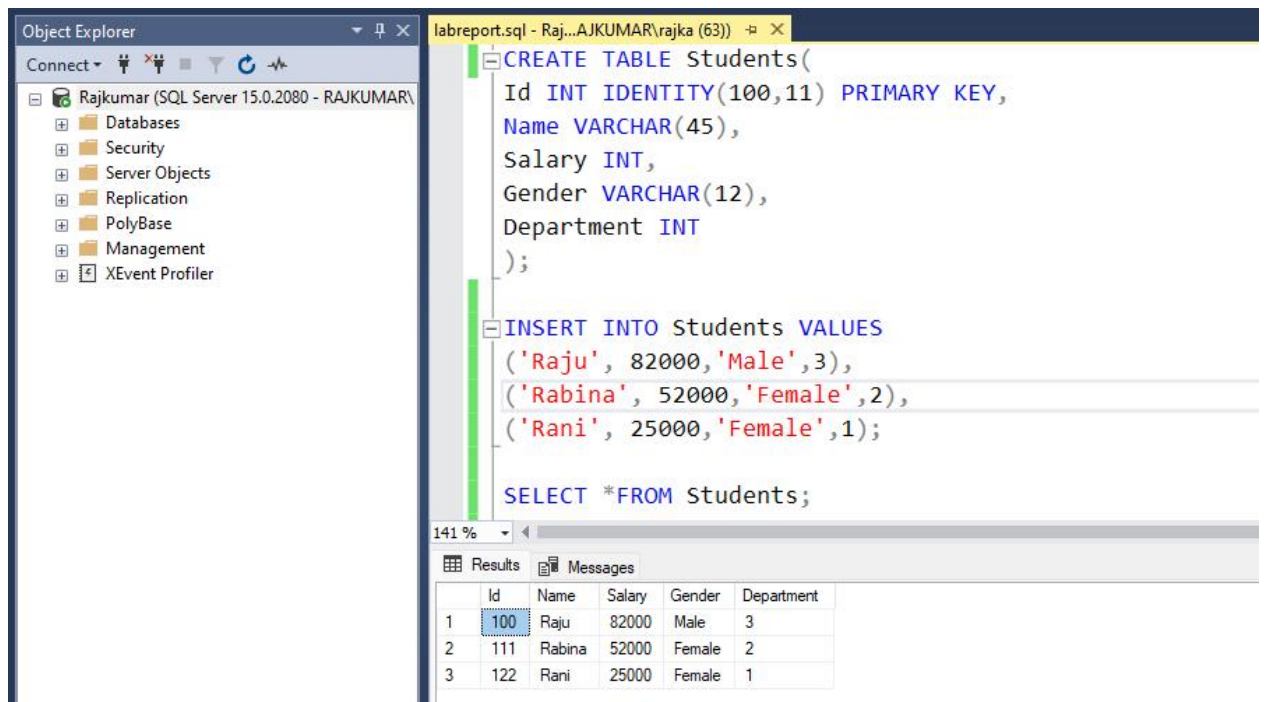
```
CREATE PROCEDURE SelectAllStudents  
AS  
SELECT * FROM Students  
GO;  
  
EXEC SelectAllStudents;
```

Below the script, the 'Results' pane shows the output of the execution as a table:

	StudentID	Name	Address	RollNO
1	100	Ram	Talchhikhel	4006
2	111	Logan	Patan	4007
3	122	Sandeep	Gwarko	4008

Handwritten Code:

20. Create a trigger that stores transaction records of each inserted operation on the Students table.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\...)'. The main pane shows a SQL script in 'labreport.sql' with the following code:

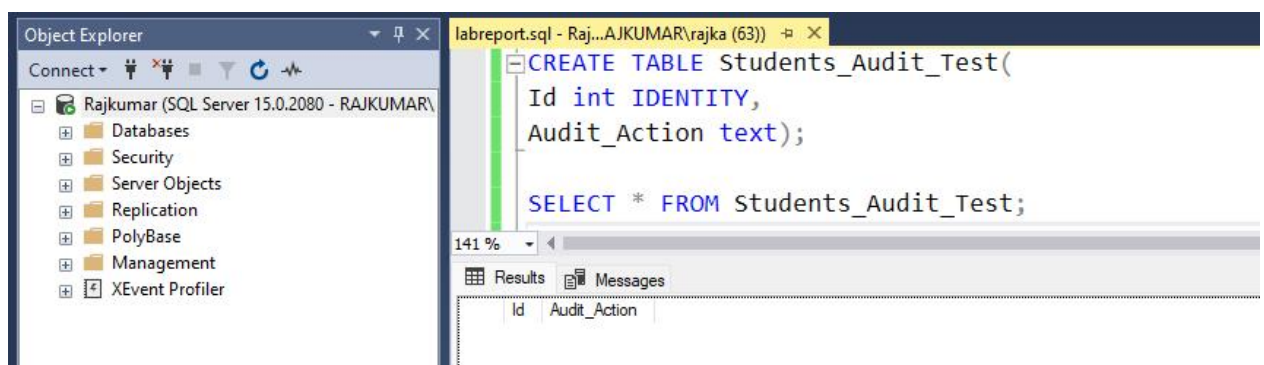
```
CREATE TABLE Students(
    Id INT IDENTITY(100,11) PRIMARY KEY,
    Name VARCHAR(45),
    Salary INT,
    Gender VARCHAR(12),
    Department INT
);

INSERT INTO Students VALUES
('Raju', 82000, 'Male', 3),
('Rabina', 52000, 'Female', 2),
('Rani', 25000, 'Female', 1);

SELECT * FROM Students;
```

Below the script, the 'Results' tab shows the output of the SELECT statement:

	Id	Name	Salary	Gender	Department
1	100	Raju	82000	Male	3
2	111	Rabina	52000	Female	2
3	122	Rani	25000	Female	1



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the server structure for 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR\...)'. The main pane shows a SQL script in 'labreport.sql' with the following code:

```
CREATE TABLE Students_Audit_Test(
    Id int IDENTITY,
    Audit_Action text);

SELECT * FROM Students_Audit_Test;
```

Below the script, the 'Results' tab shows the output of the SELECT statement:

Id	Audit_Action
----	--------------

Syntax:

```
CREATE TRIGGER schema.trigger_name
ON table_name
AFTER {INSERT}
{NOT FOR REPLICATION}
AS
{SQL Statements}
```

Query with Output:

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the server 'Rajkumar (SQL Server 15.0.2080 - RAJKUMAR)' with folders for Databases, Security, Server Objects, Replication, PolyBase, Management, and XEvent Profiler. The main window shows a SQL query in the 'labreport.sql' file. The query is as follows:

```
CREATE TRIGGER trInsertStudents
ON Students
FOR INSERT
AS
BEGIN
    Declare @Id int
    SELECT @Id = Id from inserted
    INSERT INTO Students_Audit_Test
    VALUES ('New Students with Id = ' + CAST(@Id AS VARCHAR(10)) + ' is added at '
    + CAST(Getdate() AS VARCHAR(22)))
END

INSERT INTO Students VALUES ('Prabin', 5000, 'Male', 3)
INSERT INTO Students VALUES ('Shyam', 6000, 'Male', 1)

SELECT * FROM Students_Audit_Test;
```

Below the query editor, the 'Results' tab shows the output of the query. It contains two rows of data:

Id	Audit_Action
1	New Students with Id = 133 is added at May 23 20...
2	New Students with Id = 144 is added at May 23 20...

Handwritten Code:

