



Module 15: Fundamental SQL Statements



Reference Tables Used in This Module

We will use the following sample tables to demonstrate all SQL statements in this module:

- Professors
- Department
- Register

🌟 Image Reference:



Register

DeptID	PID	Combine col(DeptID_PID)
D1	P2	D1_P2
D1	P3	D1_P3
D2	P1	D2_P1
D3	P4	D3_P4
D3	P6	D3_P6
D4	P5	D4_P5

Professors

PID	First name	Last name	DeptID
P1	Tom	Jones	D2
P2	Erica	Sharma	D1
P3	Cole	David	D1
P4	Yash	Sharma	D3
P5	Tarini	Mittal	D4
P6	Riya	Kapoor	D3

Department

DeptID	Department
D1	Computer Science
D2	English
D3	Statistics
D4	Geography

◆ How to Create a Database

Definition:

Creating a database means setting up a named space to store related tables and data objects.

Syntax:

`CREATE DATABASE database_name;`

Method 1: Using SSMS

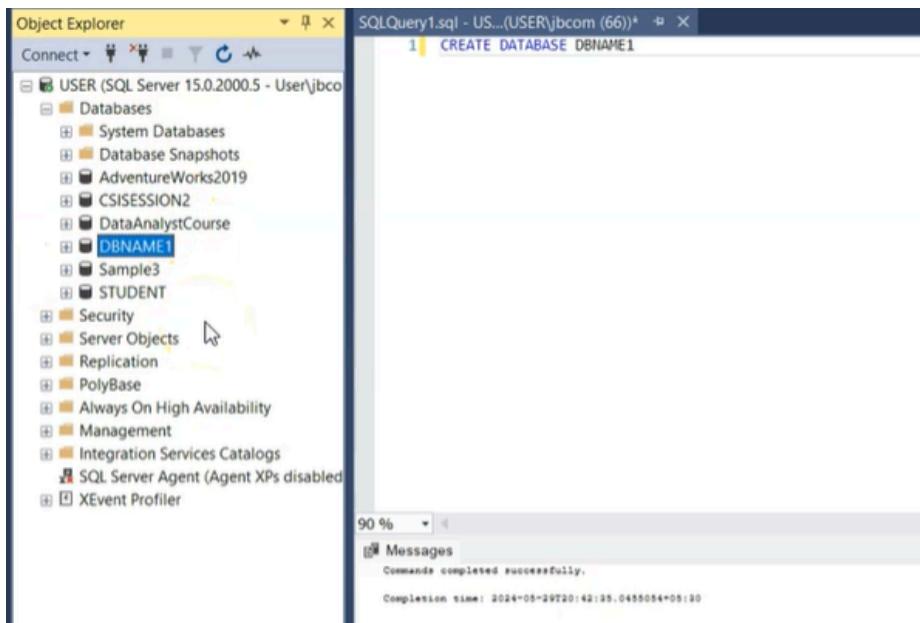
1. Open **SQL Server Management Studio**
2. In **Object Explorer**, right-click on **Databases** → **New Database**
3. Name your database (e.g., `CollegeDB`) → Click **OK**

✓ Method 2: Using Query

CREATE DATABASE database_name;

📌 Example:

CREATE DATABASE DBNAME1;



⌚ How to Select a Database

✓ Definition:

You must select the **active database** you want to work with.

✓ Method 1: Dropdown in SSMS

- Use the dropdown menu on the toolbar to select your database (e.g., `CollegeDB`)

✓ Method 2: Using Query

USE database_name;

📌 Example:

USE CollegeDB;

SQL Constraints

Definition:

Constraints are **rules** applied to table columns to enforce **data integrity**.

Constraint	Purpose
PRIMARY KEY	Uniquely identifies each row in a table
FOREIGN KEY	Links a column to another table's primary key
NOT NULL	Prevents null (empty) values
UNIQUE	Ensures all values are different
CHECK	Ensures column value meets a condition
DEFAULT	Sets a default value

SQL Data Types

Data Type	Description
INT	Whole numbers
NVARCHAR(n)	Variable-length text (Unicode)
DATE	Stores date values

1. **CREATE** Statement

Definition:

Used to **create new tables or databases** in SQL.

Syntax:

```
CREATE TABLE table_name (
    column_name datatype constraints,
    ... );
```

Examples:

1.

```
CREATE TABLE Department (
    DeptID NVARCHAR(20) PRIMARY KEY,
    DeptName NVARCHAR(20) NOT NULL );
```

2.

```
CREATE TABLE Professors (
    PID NVARCHAR(20) PRIMARY KEY,
    Fullname NVARCHAR(50) NOT NULL,
    Age INT,
    DeptID NVARCHAR(20),
    FOREIGN KEY (DeptID) REFERENCES Department(DeptID) );
```

The screenshot shows the SSMS interface. On the left, the Object Explorer pane displays a tree view of the database structure under 'USER (SQL Server 15.0.2000.5 - User\jbc0)'. The 'Tables' node under 'DataAnalystCourse' is expanded, showing 'dbo.Department' and 'dbo.Professors'. On the right, the 'SQLQuery2.sql - not connected*' window contains the following SQL code:

```
1 CREATE TABLE Department(
2     DeptID nvarchar(20) primary key,
3     DeptName nvarchar(20) NOT NULL);
4
5 CREATE TABLE Professors(
6     PID nvarchar(20) primary key,
7     Fullname nvarchar(20) NOT NULL,
8     Age int ,
9     DeptID nvarchar(20));
```

The status bar at the bottom of the window shows 'Commands completed successfully.' and a completion time of '2024-05-29T21:04:54.2116056+05:30'.

✍ 2. **INSERT** Statement

✓ Definition:

Used to **insert data into a table**.

✓ Syntax (Recommended):

```
INSERT INTO table_name (col1, col2, ...)
VALUES (val1, val2, ...);
```

✓ Syntax (Less preferred):

```
INSERT INTO table_name
VALUES (val1, val2, ...);
```

✓ Why use column names?

✓ Safer, clearer, prevents column mismatch errors.

📌 Example (from your image):

```
INSERT INTO Department VALUES
('D1', 'Computer Science'),
('D2', 'English'),
('D3', 'Statistics'),
('D4', 'Geography');
```

```
INSERT INTO Professors VALUES
('P1', 'Tom Jones', 40, 'D2'),
('P2', 'Erica Sharma', 38, 'D1'),
('P3', 'Cole David', 42, 'D1'),
('P4', 'Yash Sharma', 35, 'D3'),
('P5', 'Tarini Mittal', 33, 'D4'),
('P6', 'Riya Kapoor', 31, 'D3');
```

```
SQLQuery1.sql - G...(GUDDU\nik32 (61))*
[ ] INSERT INTO Department VALUES
[ ] ('D1', 'Computer Science'),
[ ] ('D2', 'English'),
[ ] ('D3', 'Statistics'),
[ ] ('D4', 'Geography');

[ ] INSERT INTO Professors VALUES
[ ] ('P1', 'Tom Jones', 40, 'D2'),
[ ] ('P2', 'Erica Sharma', 38, 'D1'),
[ ] ('P3', 'Cole David', 42, 'D1'),
[ ] ('P4', 'Yash Sharma', 35, 'D3'),
[ ] ('P5', 'Tarini Mittal', 33, 'D4'),
[ ] ('P6', 'Riya Kapoor', 31, 'D3');

100 % < 
Messages

(4 rows affected)
(6 rows affected)
Completion time: 2025-05-30T09:13:38.3185361+05:30
```

3. Primary Key & Foreign Key

Primary Key:

A column that **uniquely identifies** each row in a table.

Foreign Key:

A column that **links** to another table's primary key.

Syntax:

```
-- Inline with CREATE TABLE
PID NVARCHAR(20) PRIMARY KEY,
-- Foreign Key
FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
```

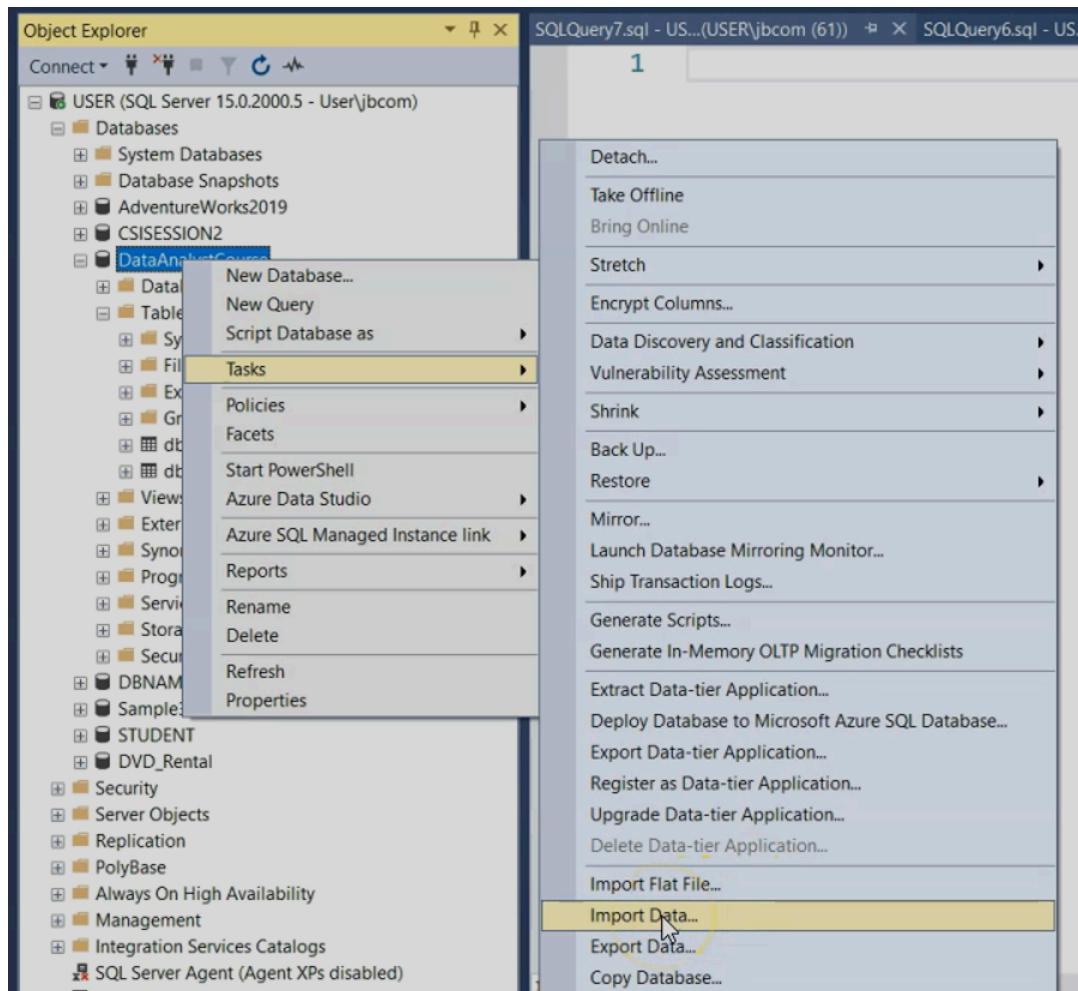
3 Relationships (from your image):

1. Each professor belongs to one department (via DeptID).
2. Each department may have multiple professors.
3. **Register** shows combinations of multiple professors in departments (many-to-many).

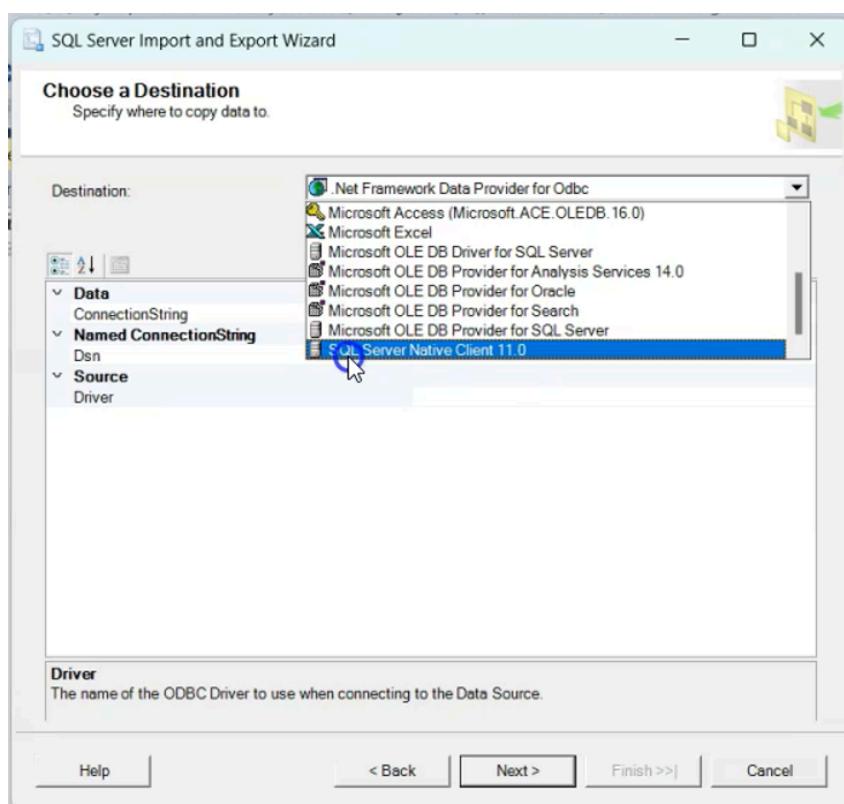
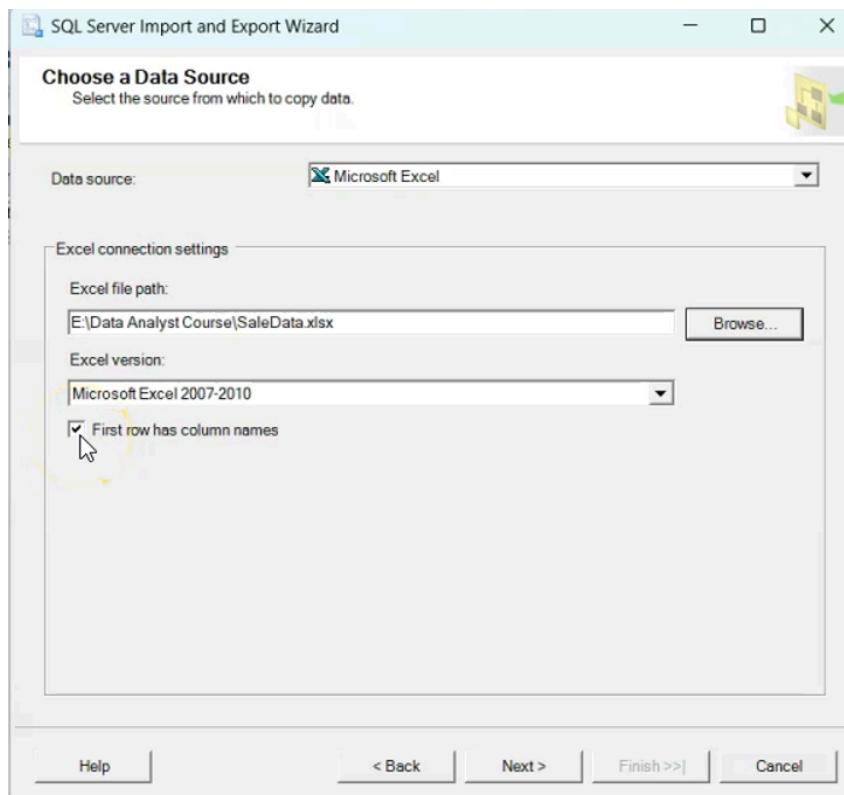
4. Import Data from File (CSV)

✓ Steps:

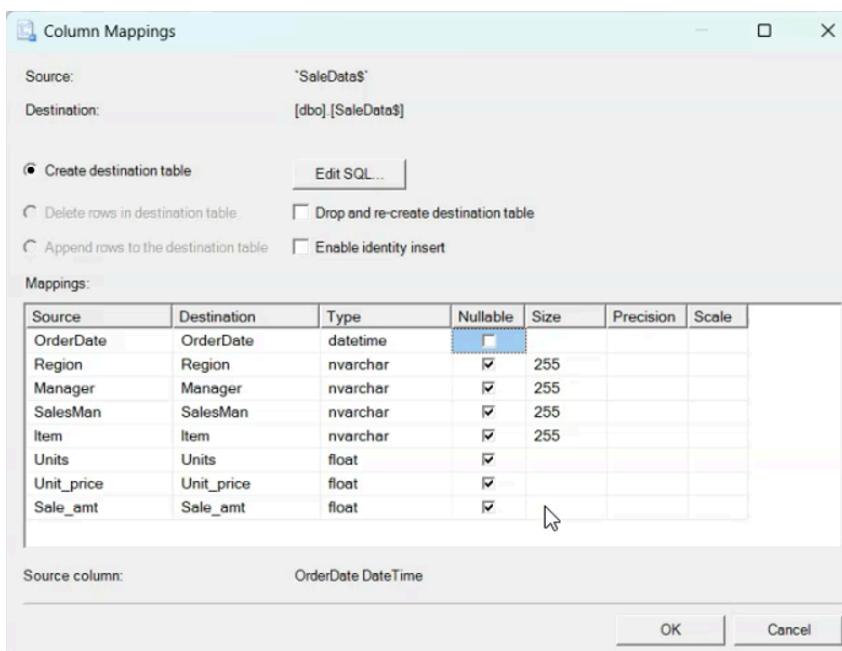
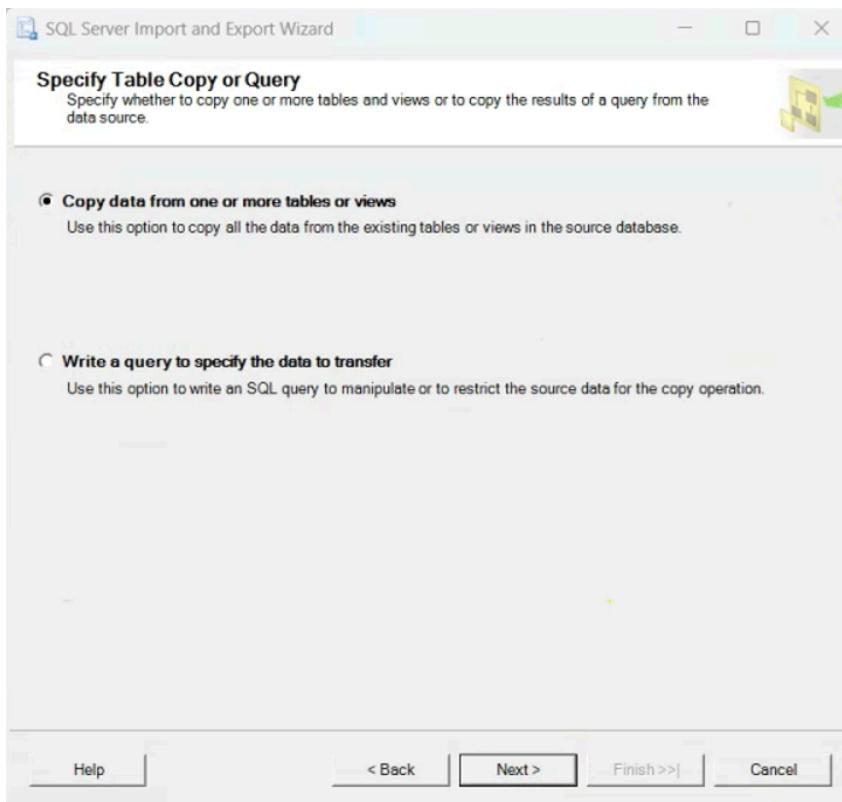
1. Right-click database → Tasks → Import Flat File

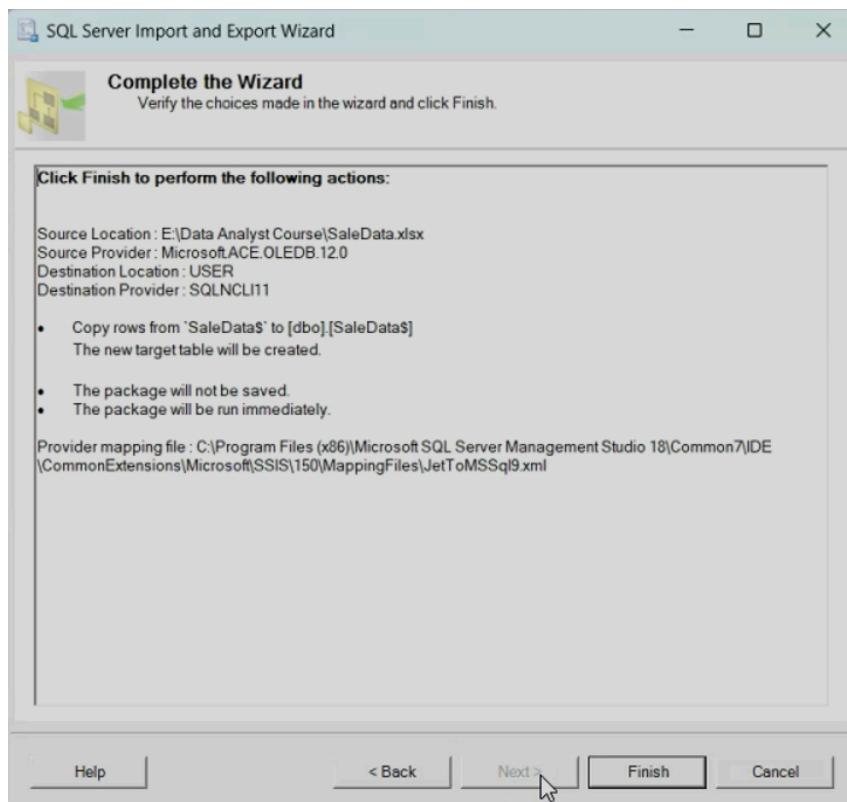
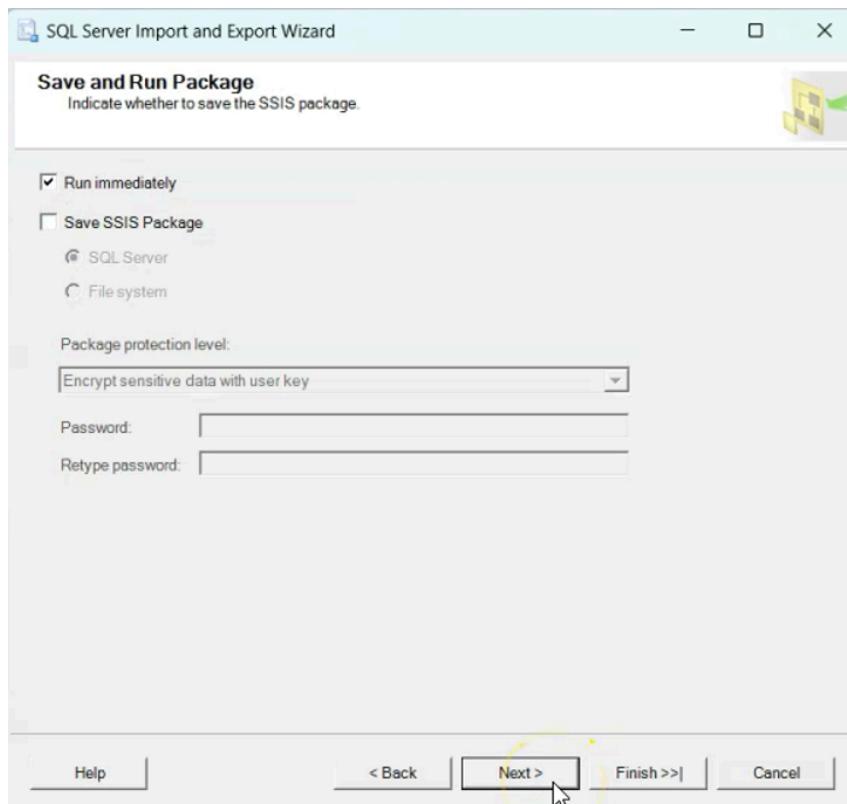


2. Select the CSV file → Click Next

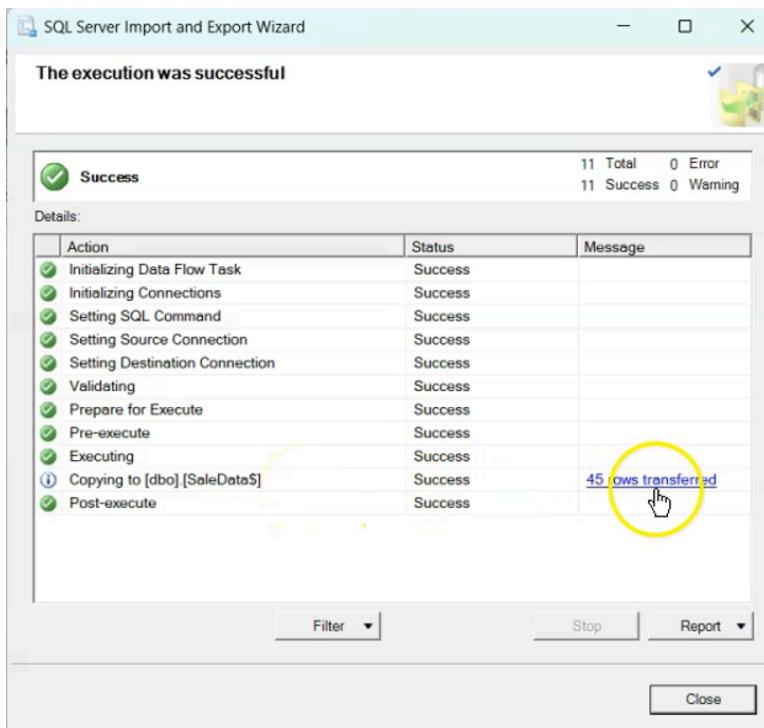


3. Define column names and types





4. Preview and finish



✓ The table is created and data is imported automatically.

A screenshot of the Microsoft SQL Server Management Studio (SSMS) interface. On the left, there is a code editor window with a yellow vertical scrollbar. The code shown is: 66, 67 select * from sales; 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80. Below the code editor is a status bar showing '145 %'. On the right, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is selected and displays a table of 12 rows of sales data. The columns are: OrderDate, Region, Manager, SalesMan, Item, Units, Unit_price, Sale_amt. The data is as follows:

	OrderDate	Region	Manager	SalesMan	Item	Units	Unit_price	Sale_amt
1	2018-01-06	East	Martha	Alexander	Television	95	1198	113810
2	2018-01-23	Central	Hermann	Shelli	Home Theater	50	500	25000
3	2018-02-09	Central	Hermann	Luis	Television	36	1198	43128
4	2018-02-26	Central	Timothy	David	Cell Phone	27	225	6075
5	2018-03-15	West	Timothy	Stephen	Television	56	1198	67088
6	2018-04-01	East	Martha	Alexander	Home Theater	60	500	30000
7	2018-04-18	Central	Martha	Steven	Television	75	1198	89850
8	2018-05-05	Central	Hermann	Luis	Television	90	1198	107820
9	2018-05-22	West	Douglas	Michael	Television	32	1198	38336
10	2018-06-08	East	Martha	Alexander	Home Theater	60	500	30000
11	2018-06-25	Central	Hermann	Sigal	Television	90	1198	107820
12	2018-07-12	East	Martha	Diana	Home Theater	29	500	14500

5. **SELECT Statement**

Definition:

Used to **retrieve data** from a table.

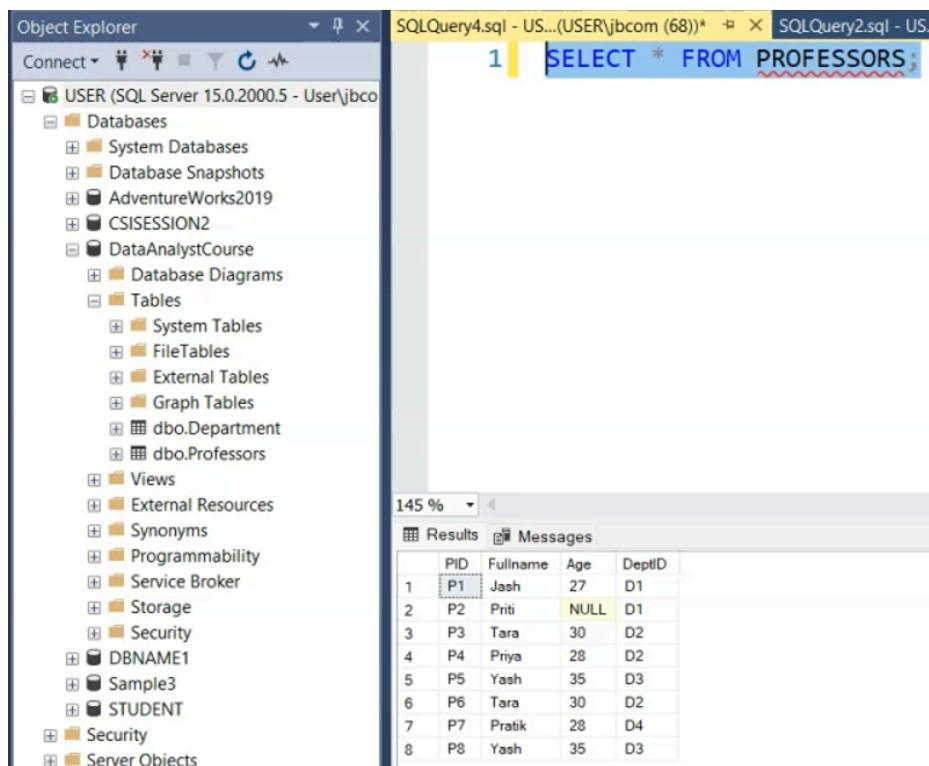
Syntax:

`SELECT column1, column2 FROM table_name;`

`SELECT * FROM table_name; -- All columns`

Example:

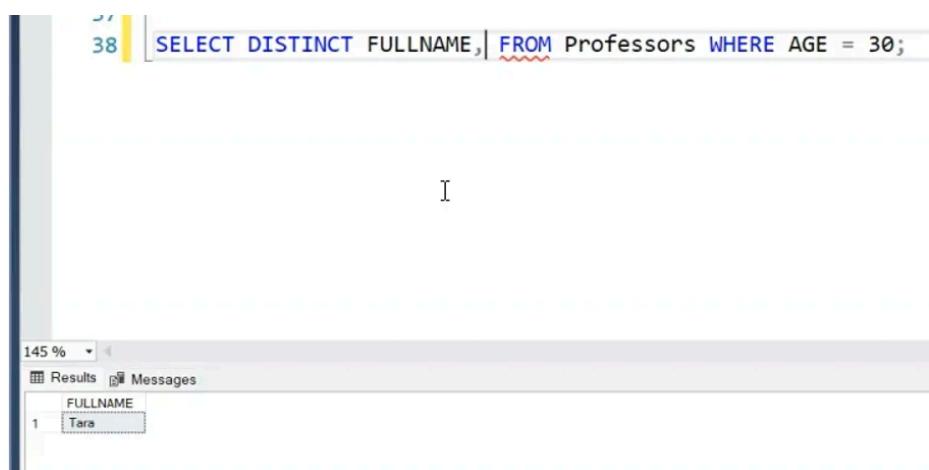
1. `SELECT * FROM Professors;`



The screenshot shows the SSMS interface. On the left is the Object Explorer pane, which lists various databases and objects within the 'USER' database, including 'AdventureWorks2019', 'CSISESSION2', 'DataAnalystCourse', 'dbo', 'Views', and 'Security'. On the right is the SQL Query window titled 'SQLQuery4.sql - US...'. It contains the query `SELECT * FROM PROFESSORS;`. Below the query is a results grid titled 'Results' showing the following data:

	PID	Fullname	Age	DeptID
1	P1	Jash	27	D1
2	P2	Priti	NULL	D1
3	P3	Tara	30	D2
4	P4	Priya	28	D2
5	P5	Yash	35	D3
6	P6	Tara	30	D2
7	P7	Pratik	28	D4
8	P8	Yash	35	D3

2. `SELECT DISTINCT Fullname FROM Professors;`



The screenshot shows the SSMS interface with the same query as above. The results grid shows the following data:

	FULLNAME
1	Tara

6. SELECT DISTINCT

✓ Definition:

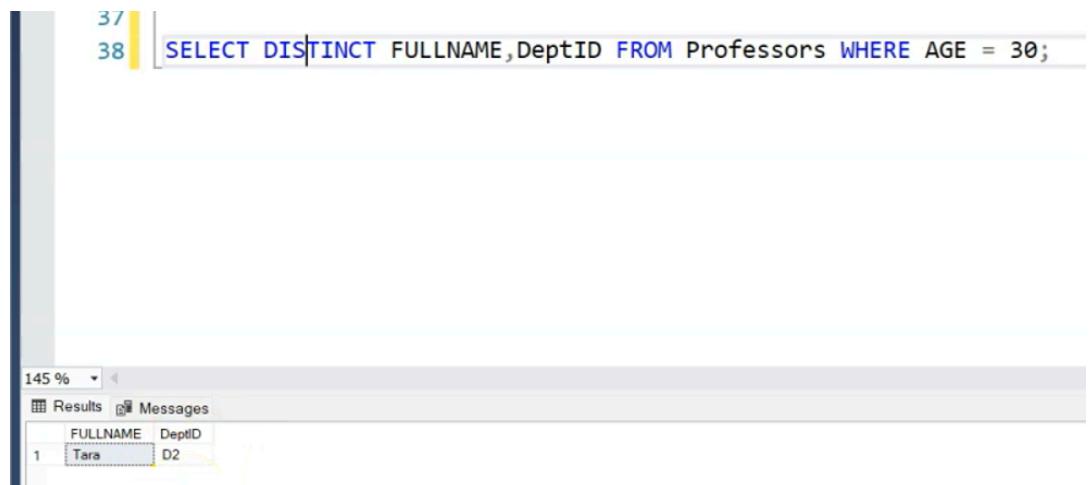
Returns only **unique values**, removing duplicates.

✓ Syntax:

```
SELECT DISTINCT column FROM table_name;
```

📌 Example:

```
1. SELECT DISTINCT Fullname, DeptID  
   FROM Professors  
   WHERE Age < 30;
```



The screenshot shows a SQL query window with two numbered lines of code:

```
37  
38 | SELECT DISTINCT FULLNAME,DeptID FROM Professors WHERE AGE = 30;
```

Below the code, the results pane displays a table with one row:

	FULLNAME	DeptID
1	Tara	D2

🔍 7. WHERE Clause

✓ Definition:

Filters rows based on a given condition.

✓ Syntax:

```
SELECT column1 FROM table_name WHERE condition;
```

📌 Example:

```
1. SELECT DISTINCT Fullname  
   FROM Professors  
   WHERE Age < 30;
```

41
42 | SELECT DISTINCT Fullname FROM Professors WHERE AGE < 30;

145 %

Results Messages

	Fullname
1	Jash
2	Pratik
3	Priya

8. Logical Operators

Definition:

Used in `WHERE` clause to combine conditions.

Operator	Meaning
AND	All conditions true
OR	At least one true
NOT	Negates a condition

Syntax:

`SELECT * FROM table_name WHERE condition1 LogicalOperator condition2;`

Examples:

1. `SELECT * FROM Professors WHERE Age = 30 AND DeptID = 'D2';`

43
44 | SELECT * FROM Professors WHERE Age = 30 AND deptid = 'D2';

145 %

Results Messages

	PID	Fullname	Age	DeptID
1	P3	Tara	30	D2
2	P6	Tara	30	D2

2. SELECT * FROM Professors WHERE Age = 30 OR DeptID = 'D2';

```
43 | 
44 | SELECT * FROM Professors WHERE Age = 30 OR deptid = 'D2';
```

145 % ▾

Results Messages

PID	Fullname	Age	DeptID	
1	P3	Tara	30	D2
2	P4	Priya	28	D2
3	P6	Tara	30	D2

3. SELECT * FROM Professors WHERE NOT DeptID = 'D2';

```
45 | 
46 | SELECT * FROM Professors WHERE NOT DeptID = 'D2';
```

145 % ▾

Results Messages

PID	Fullname	Age	DeptID	
1	P1	Jash	27	D1
2	P2	Priti	NULL	D1
3	P5	Yash	35	D3
4	P7	Pratik	28	D4
5	P8	Yash	35	D3

9. UPDATE Statement

Definition:

Used to **modify** existing records in a table.

Syntax:

```
UPDATE table_name  
SET column = value  
WHERE condition;
```

📌 Example:

1. UPDATE Professors
SET Email = 'demo@gmail.com';

```
49
50 UPDATE Professors SET email = 'demo@gmail.com';
51 SELECT * FROM Professors;
```

145 %

Results Messages

	PID	Fullname	Age	DeptID	email
1	P1	Jash	27	D1	demo@gmail.com
2	P2	Priti	NULL	D1	demo@gmail.com
3	P3	Tara	30	D2	demo@gmail.com
4	P4	Priya	28	D2	demo@gmail.com
5	P5	Yash	35	D3	demo@gmail.com
6	P6	Tara	30	D2	demo@gmail.com
7	P7	Pratik	28	D4	demo@gmail.com
8	P8	Yash	35	D3	demo@gmail.com

10. **DELETE** Statement

Definition:

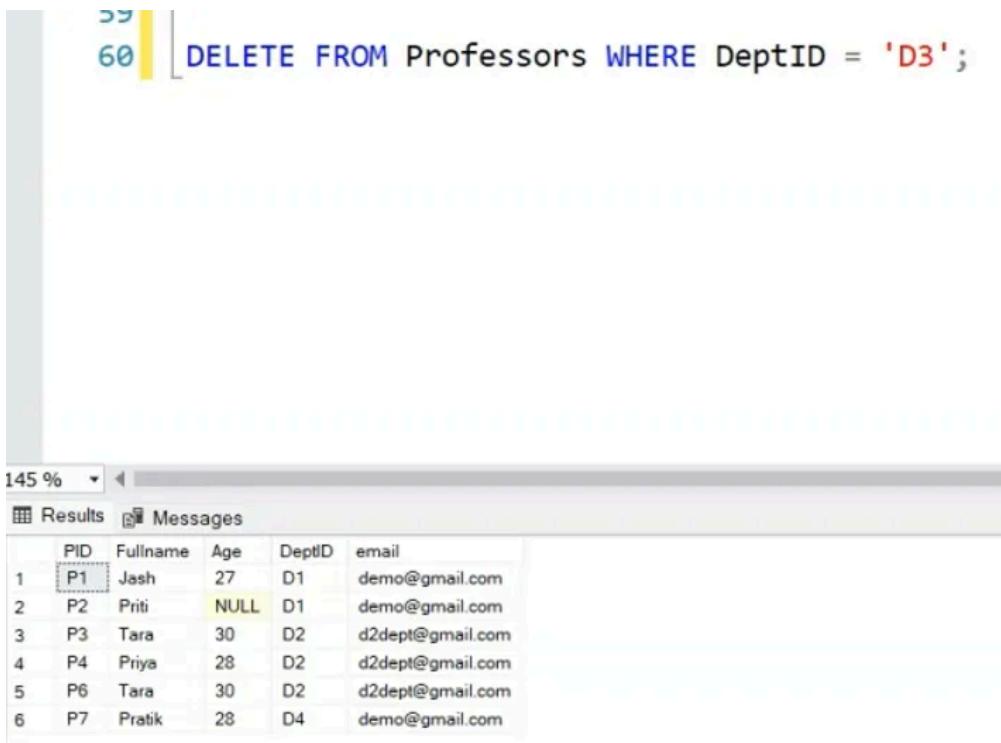
Used to **remove** records from a table.

Syntax:

DELETE FROM table_name WHERE condition;

Examples:

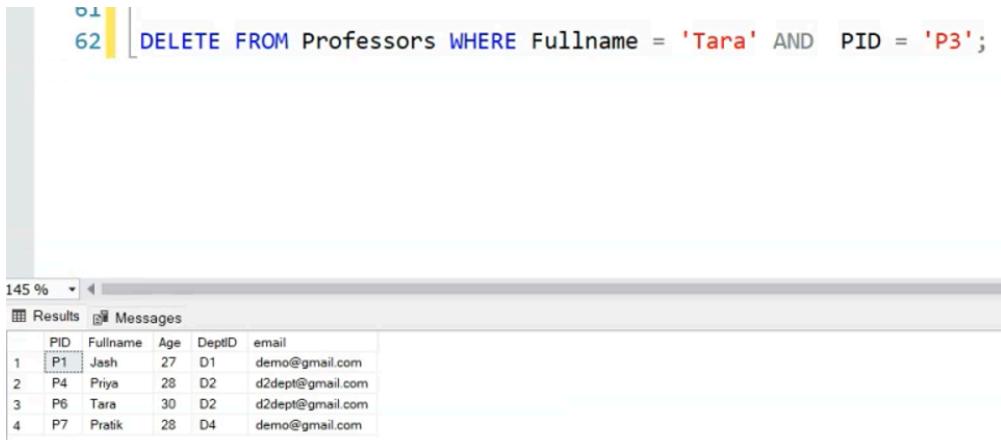
1. DELETE FROM Professors WHERE DeptID = 'D3';



The screenshot shows the results of a DELETE query on the Professors table. The table has columns: PID, Fullname, Age, DeptID, and email. The data is as follows:

	PID	Fullname	Age	DeptID	email
1	P1	Jash	27	D1	demo@gmail.com
2	P2	Priti	NULL	D1	demo@gmail.com
3	P3	Tara	30	D2	d2dept@gmail.com
4	P4	Priya	28	D2	d2dept@gmail.com
5	P6	Tara	30	D2	d2dept@gmail.com
6	P7	Pratik	28	D4	demo@gmail.com

2. DELETE FROM Professors WHERE Fullname = 'Tarini Mittal' AND DeptID = 'D4';



The screenshot shows the results of a DELETE query on the Professors table. The table has columns: PID, Fullname, Age, DeptID, and email. The data is as follows:

	PID	Fullname	Age	DeptID	email
1	P1	Jash	27	D1	demo@gmail.com
2	P4	Priya	28	D2	d2dept@gmail.com
3	P6	Tara	30	D2	d2dept@gmail.com
4	P7	Pratik	28	D4	demo@gmail.com

3. DELETE FROM Professors; -- Deletes all rows

The screenshot shows a SQL query window with two numbered lines of code:

```
53
64  DELETE FROM Professors;
```

Below the code, there are tabs for "Results" and "Messages". The "Results" tab displays a table header with columns: PID, Fullname, Age, DeptID, and email. The "Messages" tab shows the message "Commands completed successfully."

11. ALTER Statement

Definition:

Used to **modify the structure** of an existing table.

Syntax:

```
ALTER TABLE table_name ADD column_name datatype;
ALTER TABLE table_name DROP COLUMN column_name;
```

Examples:

1. ALTER TABLE Professors ADD Address NVARCHAR(30);

The screenshot shows the Object Explorer on the left and a query window on the right. The query window contains the following code:

```
1  ALTER TABLE Professors
2  ADD address nvarchar(30);
```

The results pane at the bottom shows the message "Commands completed successfully."

2. ALTER TABLE Professors DROP COLUMN Address;

The screenshot shows the Object Explorer on the left with the path: Databases > AdventureWorks2019 > Tables > dbo.Professors > Columns. The context menu for the Address column is open, showing options like 'ALTER', 'DROP', 'RENAME', etc. On the right, the Results pane displays the T-SQL command:

```
3 | ALTER TABLE Professors  
4 | DROP COLUMN Address;
```

The status bar at the bottom right shows 'Commands completed successfully.'

Difference: `DROP` vs `DELETE` vs `TRUNCATE`

Command	What It Does	Structure?	Reversible?
<code>DROP</code>	Deletes entire table	✓ Yes	✗ No
<code>DELETE</code>	Deletes rows (with condition)	✗ No	✓ Yes
<code>TRUNCATE</code>	Deletes all rows (fast)	✗ No	✗ No

Final Notes for Students

- Use `CREATE`, `INSERT`, `SELECT`, `WHERE`, `UPDATE`, `DELETE` to manage data.
- Learn `PRIMARY` and `FOREIGN KEY` to **link tables properly**.
- Always use **column names** while inserting data.
- `WHERE` filters rows; `DISTINCT` removes duplicates.
- Use `ALTER` to adjust tables without deleting them.