



**Al-Mustaqbal University**

**College of Sciences**

**Cyber Security Department**



جامعة المستقبل  
AL MUSTAQBAL UNIVERSITY

كلية العلوم  
قسم الامن السيبراني

## Lecture: (8)

**Subject: Database**

**Level: Second**

**Lecturer: Asst. Lecturer Qusai AL-Durrah**



## Lec 8: SQL Basics

### Understanding SQL and SQL Server

Structured Query Language (SQL) serves as the standard language for relational database management systems (RDBMS). SQL Server, developed by Microsoft, is one of the leading RDBMS platforms used across industries for data storage, retrieval, and management. SQL commands fall into several categories including Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL), each serving specific functions in database operations.

### Creating and Dropping Databases and Tables

The foundation of working with SQL begins with creating the structures to store your data. SQL Server allows you to create both databases (collections of related tables) and tables (structures that hold specific data) using straightforward syntax.

#### Creating a Database

Before creating tables, you need a database to contain them. The CREATE DATABASE statement establishes a new database in SQL Server:

```
CREATE DATABASE UniversityManagement;
```

```
-- Switch to the new database
```

```
USE UniversityManagement;
```

This command creates all necessary files to store database objects and data. SQL Server manages two primary files: an .mdf file for data and an .ldf file for transaction logs.

#### Creating Tables

Tables are defined by their columns, each with a specific data type and optional constraints. Here's how to create a basic Student table:

```
CREATE TABLE Students ( StudentID INT NOT NULL,
```

```
FirstName VARCHAR(50) NOT NULL, LastName VARCHAR(50) NOT NULL,
```

```
DateOfBirth DATE, Email VARCHAR(100) UNIQUE,
```

```
EnrollmentDate DATE, DepartmentID INT,
```

```
GPA FLOAT);
```



Students							
StudentID	FirstName	LastName	DateOfBirth	Email	Enrollment Date	DepartmentID	GPA
1	ahmed	aljbory	22/2/1996	a@email.com	2/11/2015	4	3.2
2	zahraa	aljanabi	18/1/1996	b@email.com	3/11/2015	3	3.8
3	yusef	aljbory	25/4/1996	c@email.com	9/11/2015	1	2.9
4	zainab	alsaffar	29/9/1996	d@email.com	2/11/2015	2	2.5

This example demonstrates several important concepts:

Column definitions with appropriate data types (INT, VARCHAR, DATE, FLOAT)

NOT NULL constraints ensure that the data is required to be filled

UNIQUE constraint preventing duplicate email addresses

### Dropping Database Objects

When you need to remove databases or tables, SQL Server provides the DROP commands:

```
-- Remove a table
```

```
DROP TABLE Students;
```

```
-- Remove a database (use with extreme caution)
```

```
DROP DATABASE UniversityManagement;
```

**Important:** DROP operations are permanent and cannot be undone unless you have a backup. Always be absolutely certain before executing DROP commands, especially in production environments.

## SELECT Statements: Retrieving Data

The SELECT statement is arguably the most frequently used SQL command, allowing you to retrieve and view data from your tables. Understanding how to craft effective queries is essential for working with databases.

### Basic SELECT Syntax

At its simplest, a SELECT statement specifies which columns you want to retrieve and from which table:



-- Retrieve all columns from Students table

```
SELECT * FROM Students;
```

-- Retrieve specific columns

```
SELECT StudentID, FirstName, LastName, GPA FROM Students;
```

While SELECT \* is convenient for exploration, specifying exact columns is more efficient in production code and creates more maintainable queries.

### Filtering Data with WHERE Clause

The WHERE clause allows you to filter rows based on specific conditions:

-- Find students with GPA above 3.5

```
SELECT FirstName, LastName, GPA FROM Students WHERE GPA > 2.5;
```

-- Find students in a specific department

```
SELECT * FROM Students WHERE DepartmentID = 4;
```

-- Find students matching multiple criteria

```
SELECT * FROM Students WHERE DepartmentID = 2 AND GPA >= 3.0;
```

-- Find students using pattern matching

```
SELECT * FROM Students WHERE Firstname LIKE 'a%'; -- Last names starting with S
```

### Sorting Results with ORDER BY

The ORDER BY clause sorts your query results:

-- Sort students by GPA in descending order

```
SELECT FirstName, LastName, GPA FROM Students ORDER BY GPA DESC;
```

```
SELECT FirstName, LastName, GPA FROM Students ORDER BY GPA ASC;
```

## INSERT Statements: Adding Data to Tables

Once you've created tables in your database, you'll need to populate them with data. The INSERT statement is the primary method for adding new records to a table in SQL Server.

### Basic INSERT Syntax

The INSERT INTO statement has two main forms: one where you specify both the columns and their values, and another where you provide only the values in the exact order of the table's columns.



-- Insert with explicit column names (recommended approach)

```
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email,
DepartmentID) VALUES (1001, 'John', 'Smith', '2000-05-15',
'john.smith@email.com', 1);
```

-- Insert without specifying columns (depends on knowing exact column order)

```
INSERT INTO Students VALUES (1002, 'Emily', 'Johnson', '2001-03-22',
'emily.j@email.com', '2021-09-01', 2, 3.75);
```

The first approach is generally preferred because it:

- Makes queries more readable and maintainable
- Allows skipping columns with default values
- Remains valid even if table structure changes (as long as specified columns still exist)

## Inserting Multiple Rows

For better performance when adding multiple records, you can insert several rows in a single statement:

-- Insert multiple student records simultaneously

```
INSERT INTO Students (StudentID, FirstName, LastName, Email,
DepartmentID, GPA) VALUES
(1003, 'Michael', 'Williams', 'michael.w@email.com', 1, 3.42),
(1004, 'Sarah', 'Brown', 'sarah.b@email.com', 3, 3.91),
(1005, 'David', 'Jones', 'david.j@email.com', 2, 3.25);
```

## Handling NULL and Default Values

When inserting data, you can explicitly specify NULL for nullable columns or omit columns to use their default values:



-- Explicitly setting NULL

```
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email,  
DepartmentID) VALUES (1006, 'Jessica', 'Davis', NULL, 'jessica.d@email.com', 3);
```

-- Using DEFAULT keyword

```
INSERT INTO Students (StudentID, FirstName, LastName, Email, EnrollmentDate,  
DepartmentID) VALUES (1007, 'Robert', 'Wilson', 'robert.w@email.com', DEFAULT,  
2);
```

## Inserting Data from Other Tables

You can insert records based on results from a SELECT query, useful for copying or transforming data:

-- Insert students transferring from another department

```
INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email,  
DepartmentID)
```

```
SELECT TransferID, FirstName, LastName, BirthDate, EmailAddress,  
DepartmentID FROM TransferStudents
```

```
WHERE ApprovalStatus = 'Approved';
```