Introduction to Databases & Entity Framework Core

Database

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system.
- A database is usually controlled by a database management system (DBMS).
- Stores and manages information efficiently
- Supports data retrieval, insertion, and updates

Database Concepts

Table

- Core structure of a relational database
- Stores data in a structured, row and column
- Example: Customers table with Name, Email, Phone

Rows (Record)

- Each row = one complete data entry in the table
- Contains values for all defined columns
- Example: A single customer's details in the Customers table

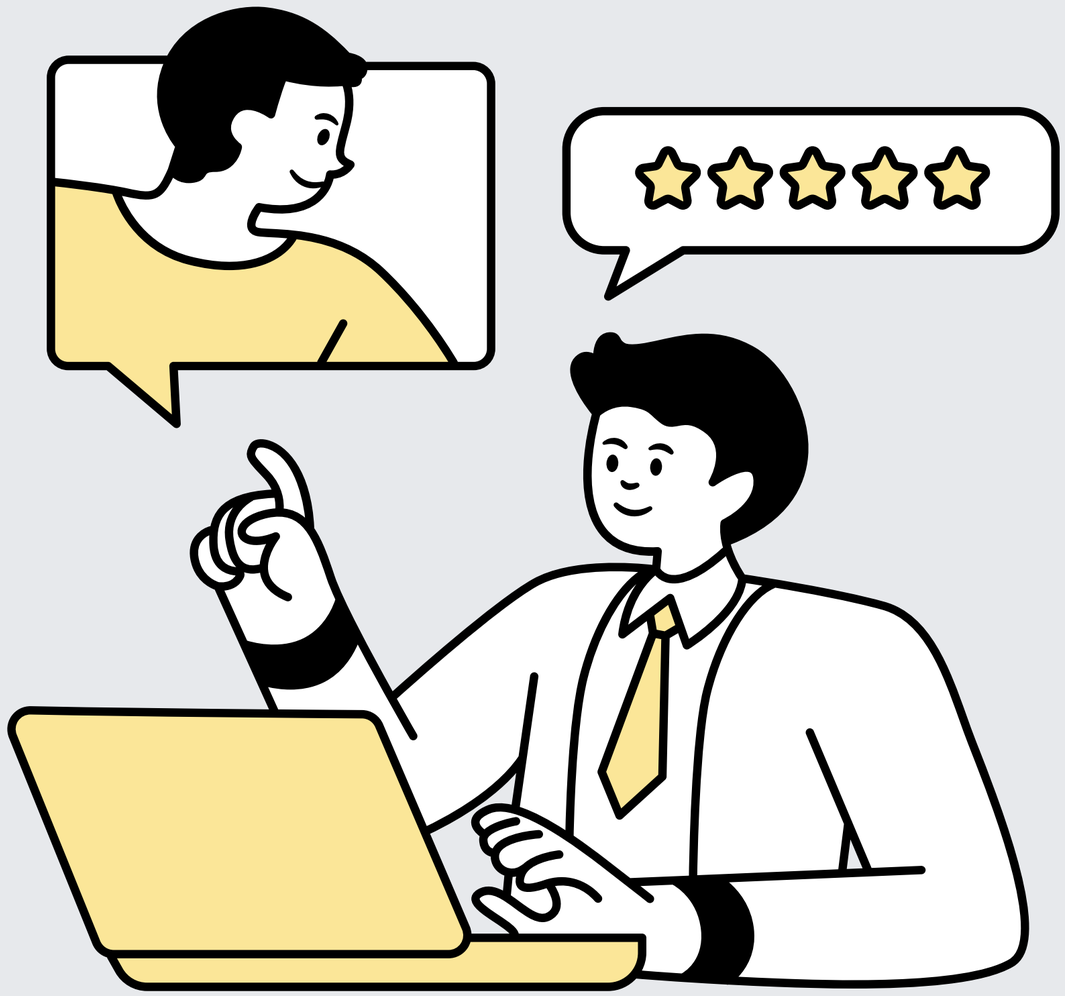
Column (Fields)

- Define the attributes or properties of data
- Each column has a specific data type (text, number, date, etc.)
- Example: Name (Text), Age (Integer), DOB (Date)

Keys

- Primary Key: Unique identifier for each record (no duplicates, not NULL).
  Example: CustomerID in Customers table

- Foreign Key: Field linking two tables by referencing another table's Primary Key.
  Example: Order.CustomerID → Customers.CustomerID



Database Design Principles

- Ensure data integrity & consistency
- Use normalization to avoid redundancy
- Maintain relationships between tables
- Plan for scalability & security

Demonstration

- To identify use database type : mysql -u root -p
- To see the databases use : Show databases;
- To create new database: create database name;
- To use the created database: Use Databasename;

```
(c) Microsoft Corporation. All rights reserved.

C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.4.22-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> Show databases;
+--------------------+
| Database           |
+--------------------+
| believe_wear       |
| c13                |
| hospital_management |
| information_schema |
| merosite           |
| movie_rental       |
| mycompany          |
| mysql              |
| mystore            |
| performance_schema |
| phpmyadmin         |
| program            |
| tech               |
| tech_academy       |
| test               |
| user               |
| week3              |
| week5c13           |
+--------------------+
18 rows in set (0.175 sec)

MariaDB [(none)]> CREATE DATABASE internship_amnil;
Query OK, 1 row affected (0.035 sec)

MariaDB [(none)]> Use internship_amnil;
Database changed
MariaDB [internship_amnil]>
```

Create Table

Creates a new table with specified columns and data types

```
MariaDB [(none)]> Use internship_amnil;
Database changed
MariaDB [internship_amnil]> CREATE TABLE Students (
    ->   Id INT PRIMARY KEY,
    ->   FirstName VARCHAR(50),
    ->   LastName VARCHAR(50),
    ->   Email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.099 sec)

MariaDB [internship_amnil]>
```

Insert

```
MariaDB [internship_amnil]> INSERT INTO Students (Id, FirstName, LastName, Email)
    -> VALUES (1, 'John', 'Doe', 'john.doe@example.com'),
    ->        (2, 'Jane', 'Smith', 'jane.smith@example.com'),
    ->        (3, 'Sam', 'Wilson', 'sam.wilson@example.com');
Query OK, 3 rows affected (0.090 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Select

```
MariaDB [internship_amnil]> Select * from Students;
+----+-----------+----------+-----------------------+
| Id | FirstName | LastName | Email                 |
+----+-----------+----------+-----------------------+
|  1 | John      | Doe      | john.doe@example.com  |
|  2 | Jane      | Smith    | jane.smith@example.com |
|  3 | Sam       | Wilson   | sam.wilson@example.com |
+----+-----------+----------+-----------------------+
3 rows in set (0.001 sec)

MariaDB [internship_amnil]>
```

Update

```
MariaDB [internship_amnil]> UPDATE Students
    -> SET Email = 'jane.smith@newdomain.com'
    -> WHERE Id = 2;
Query OK, 1 row affected (0.038 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [internship_amnil]> Select * from Students;
+----+-----------+----------+-------------------------+
| Id | FirstName | LastName | Email                   |
+----+-----------+----------+-------------------------+
|  1 | John      | Doe      | john.doe@example.com    |
|  2 | Jane      | Smith    | jane.smith@newdomain.com |
|  3 | Sam       | Wilson   | sam.wilson@example.com  |
+----+-----------+----------+-------------------------+
3 rows in set (0.000 sec)

MariaDB [internship_amnil]>
```

Delete

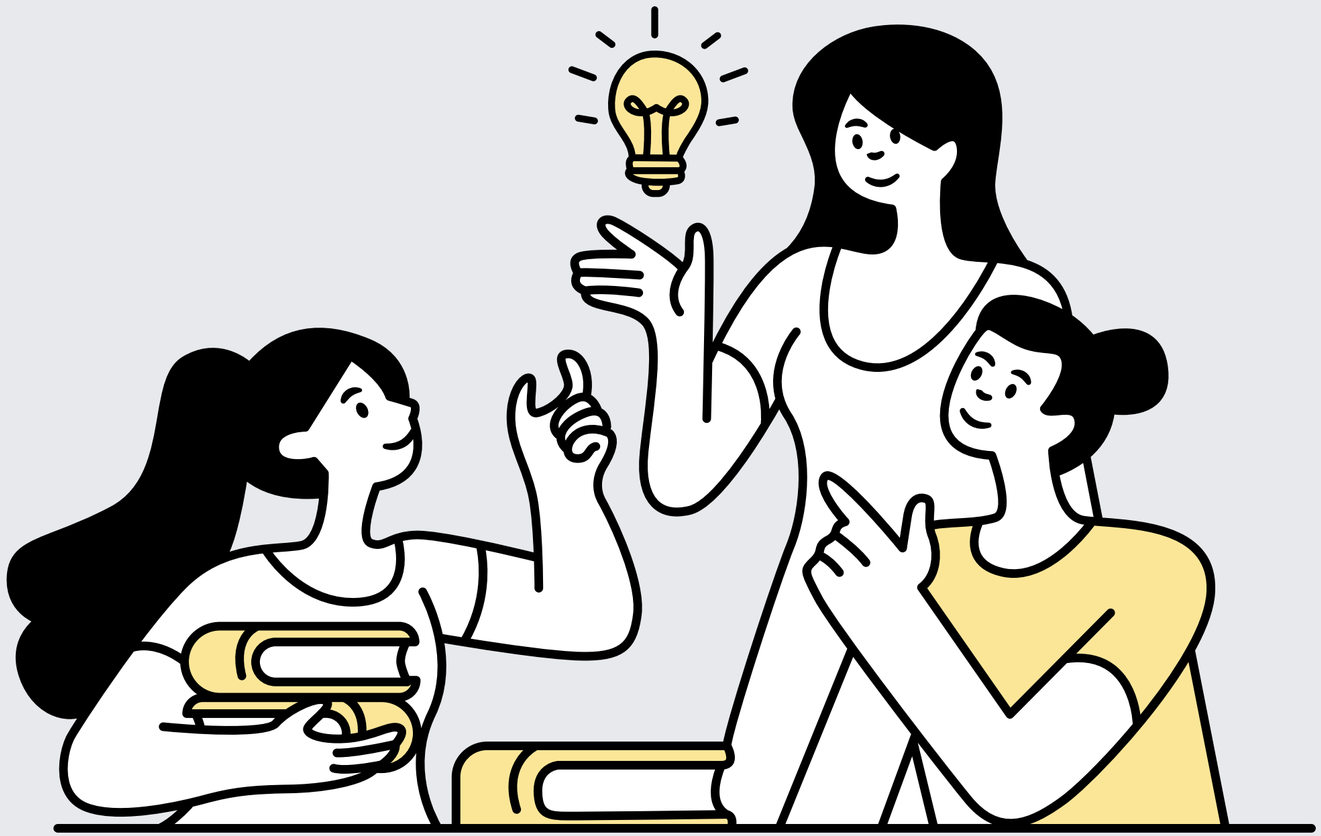
```
MariaDB [internship_amnil]> DELETE FROM Students
    -> WHERE Id = 3;
Query OK, 1 row affected (0.030 sec)

MariaDB [internship_amnil]> Select * from Students;
+----+-----------+----------+-------------------------+
| Id | FirstName | LastName | Email                   |
+----+-----------+----------+-------------------------+
|  1 | John      | Doe      | john.doe@example.com    |
|  2 | Jane      | Smith    | jane.smith@newdomain.com |
+----+-----------+----------+-------------------------+
2 rows in set (0.000 sec)

MariaDB [internship_amnil]>
```

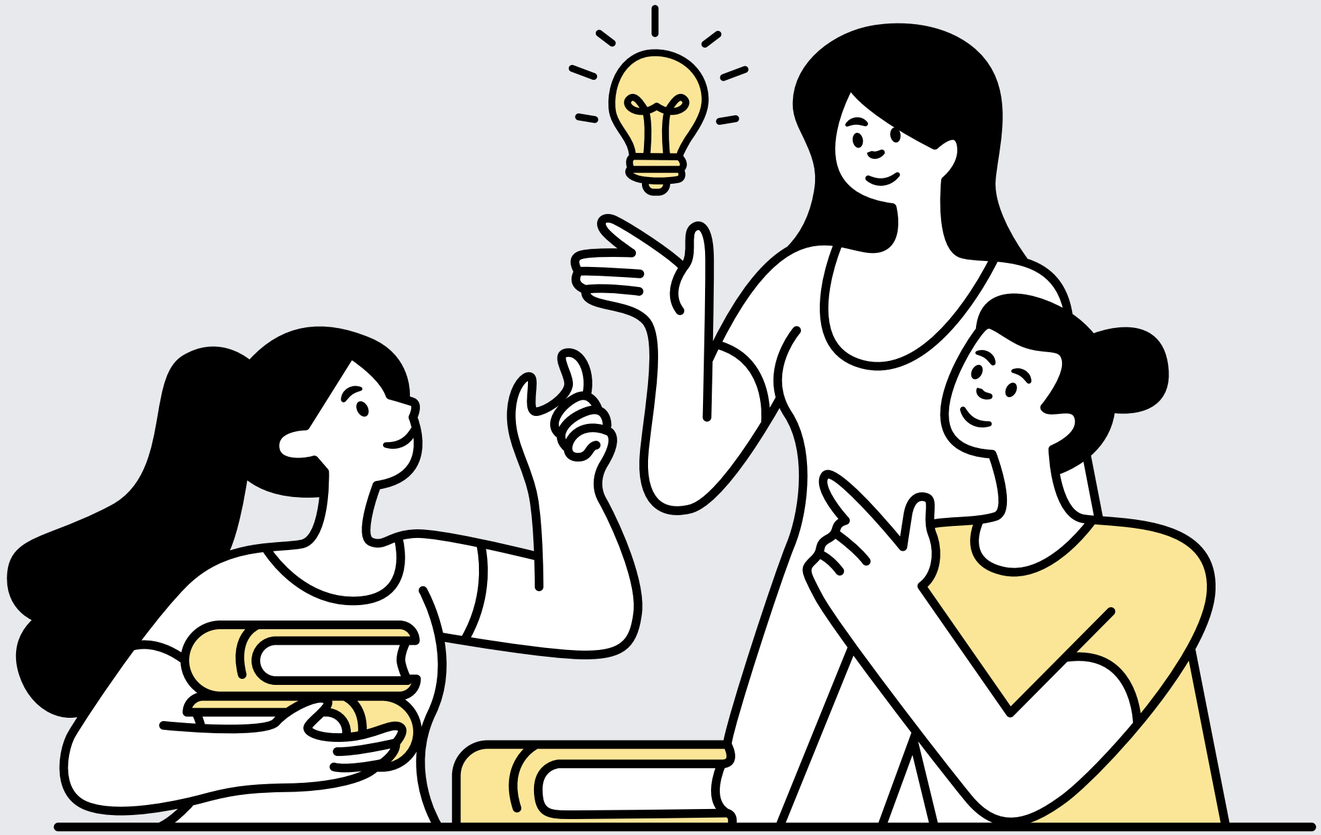
ORM

- ORM = Object-Relational Mapping
- ORM is a technique that connects your program’s objects to a database.
- In simple terms: it lets you work with databases using C# classes instead of writing SQL queries manually.
- Maps C# classes to database tables



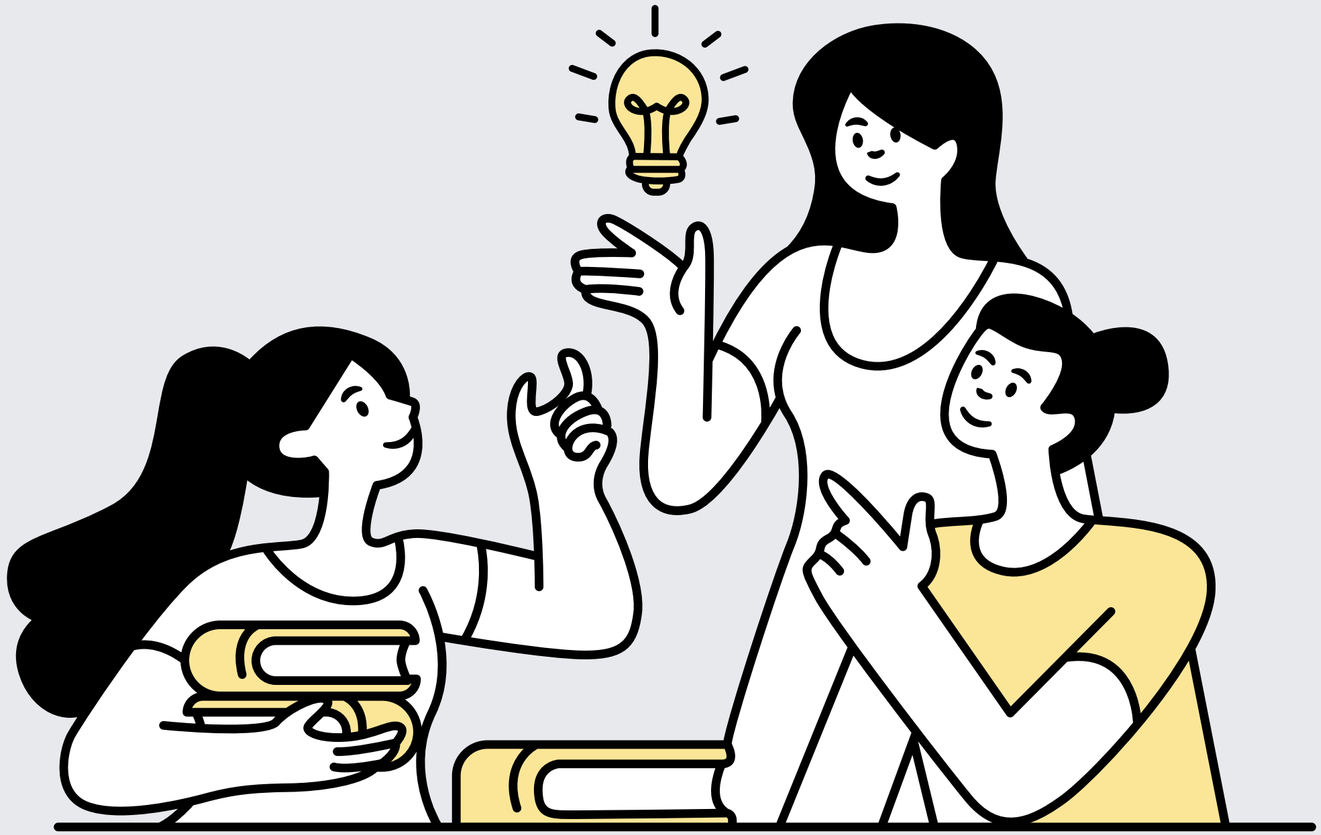
ORM Benefits

- No need to write raw SQL – you use classes and objects.
- CRUD operations become easy – create, read, update, delete using code.
- Maintains relationships automatically – like foreign keys between tables.
- Database independent – you can switch SQL Server → MySQL with minimal code changes.



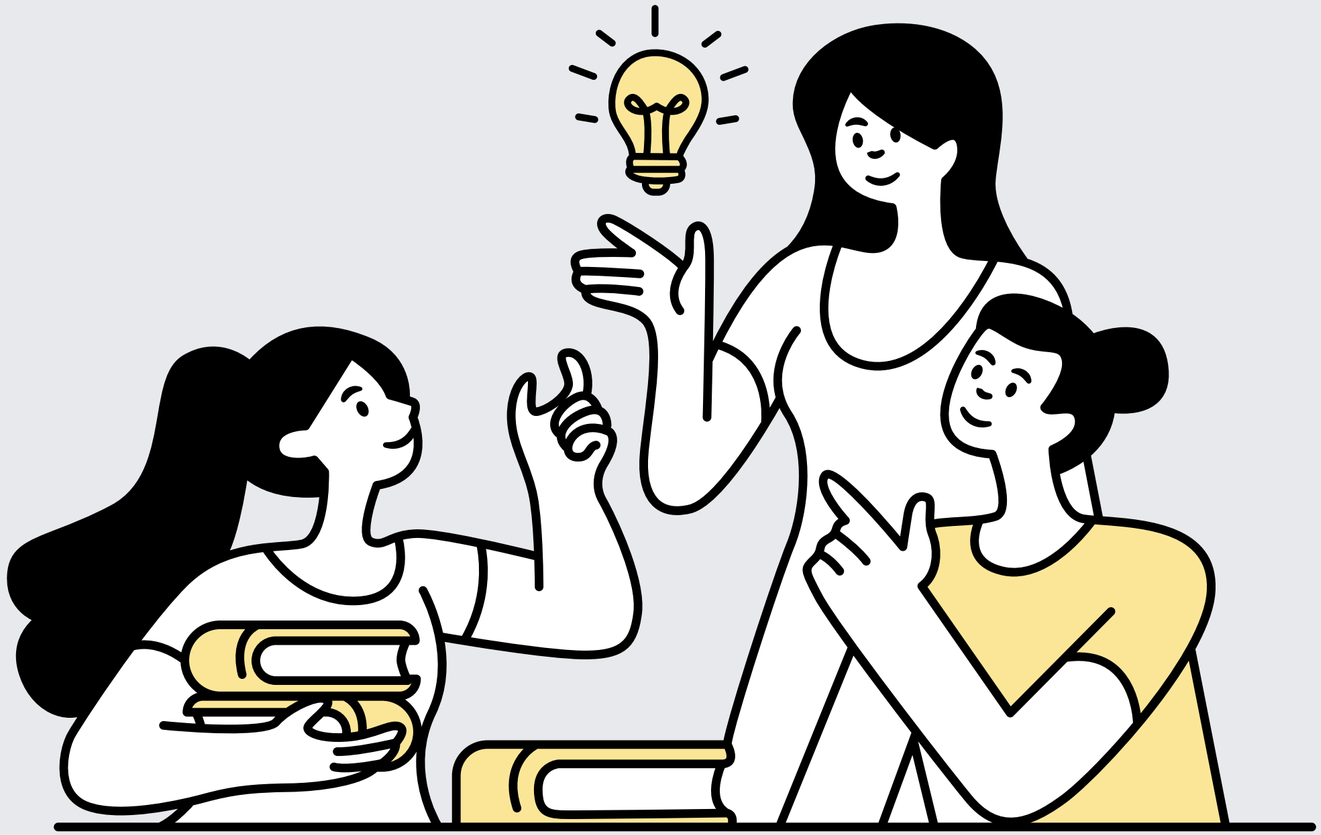
EF core

- EF Core (Entity Framework Core) is Microsoft’s ORM for .NET.
- It allows C# developers to interact with databases using objects.
- Supports Code-First, Database-First, and Migrations
- Supports Relationships – One-to-One, One-to-Many, Many-to-Many.
- Cross-platform – works on Windows, Linux, macOS.



EF core

- EF Core (Entity Framework Core) is Microsoft’s ORM for .NET.
- It allows C# developers to interact with databases using objects.
- Supports Code-First, Database-First, and Migrations
- Supports Relationships – One-to-One, One-to-Many, Many-to-Many.
- Cross-platform – works on Windows, Linux, macOS.



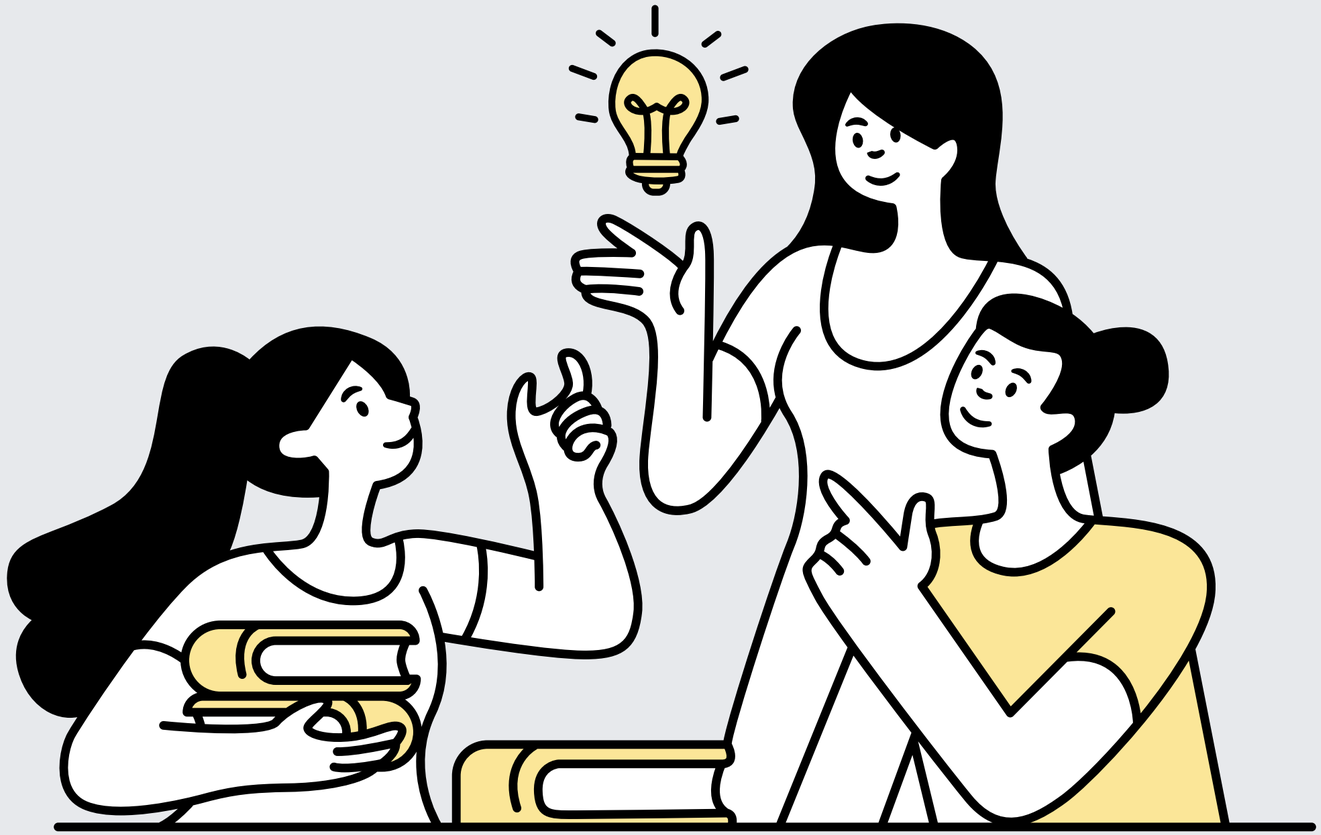
EF core setup

- Install NuGet packages:

  Microsoft.EntityFrameworkCore

  Microsoft.EntityFrameworkCore.SqlServer

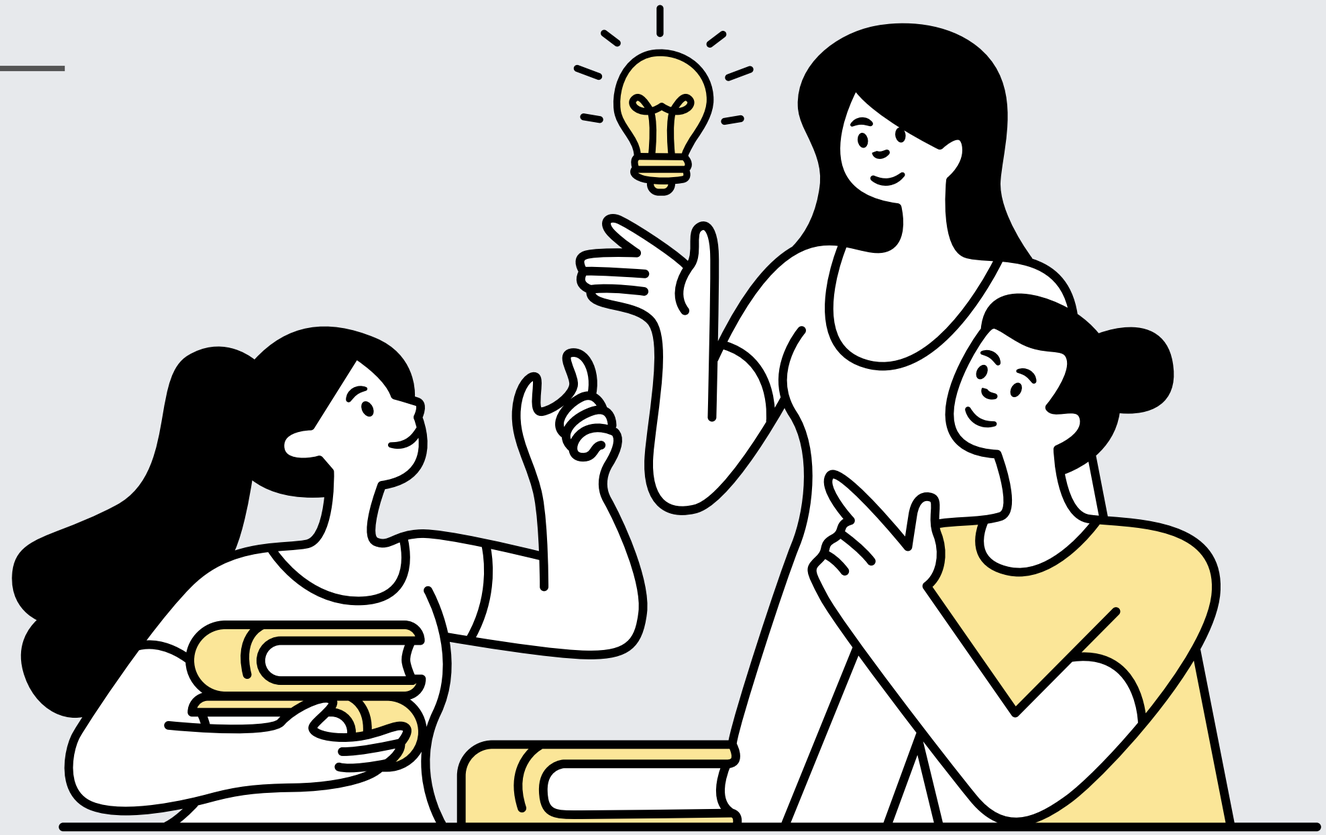
  (or other provider)
- Create a DbContext class
- Configure connection string



DbContext

What is DbContext?

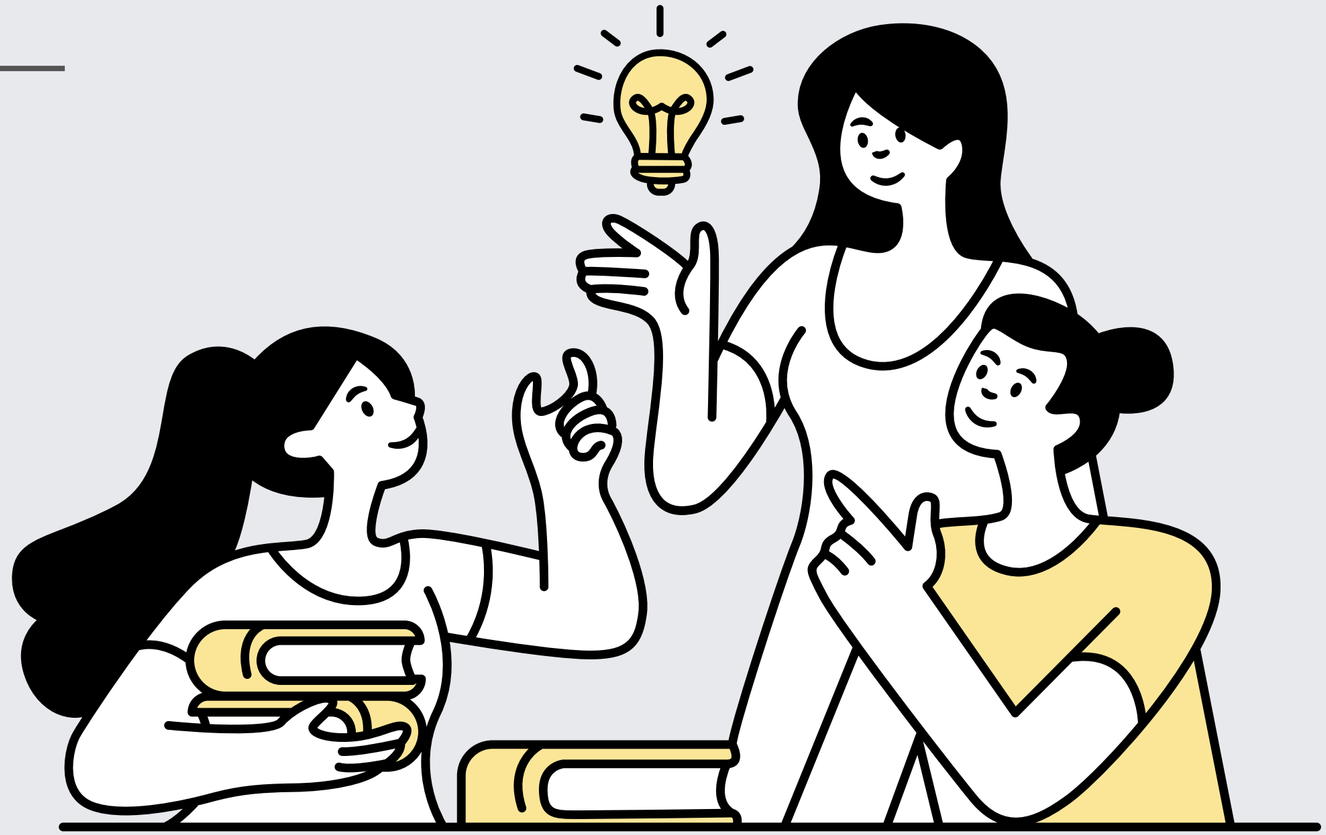
- DbContext is the main class that manages the database connection.
- Tracks objects and handles saving changes to the database.
- Think of it as a session with the database.



DbSet<T>

What is DbSet<T>?

- DbSet<T> represents a table in the database.
- Allows you to query, add, update, and delete records in that table.



Code First Approach

- Create C# classes first → generate database automatically

Advantages:

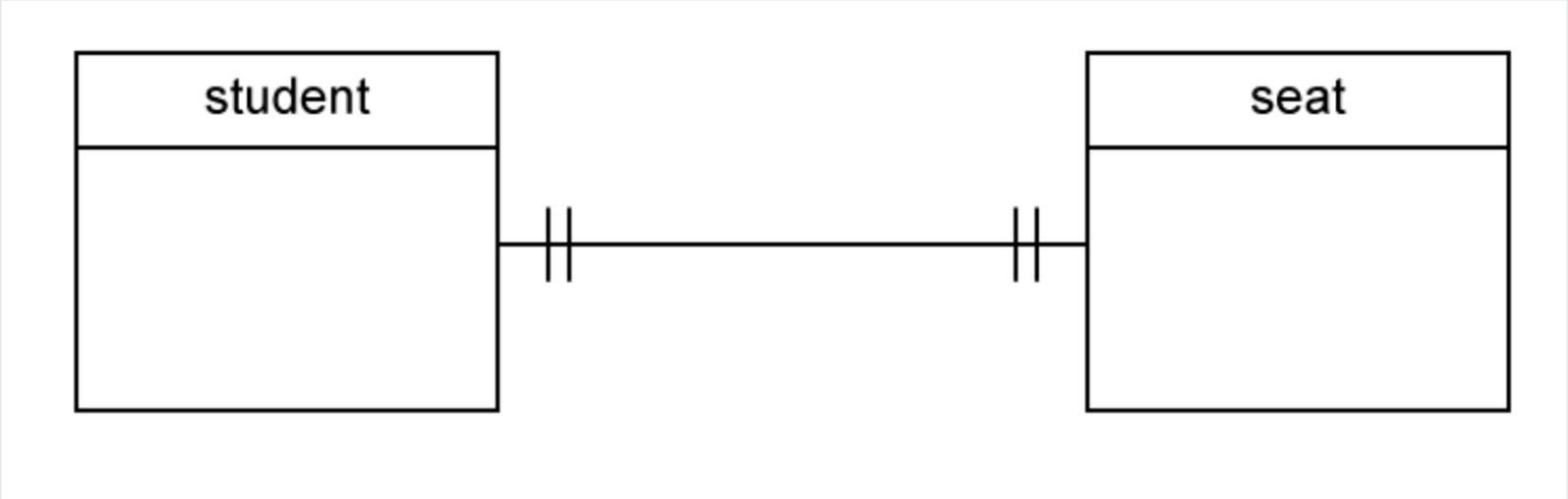
- Start with code, not database
- Easy migrations

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

One-To-One Relationship

Each record in Table A is associated with one and only one record in Table B, and vice versa.

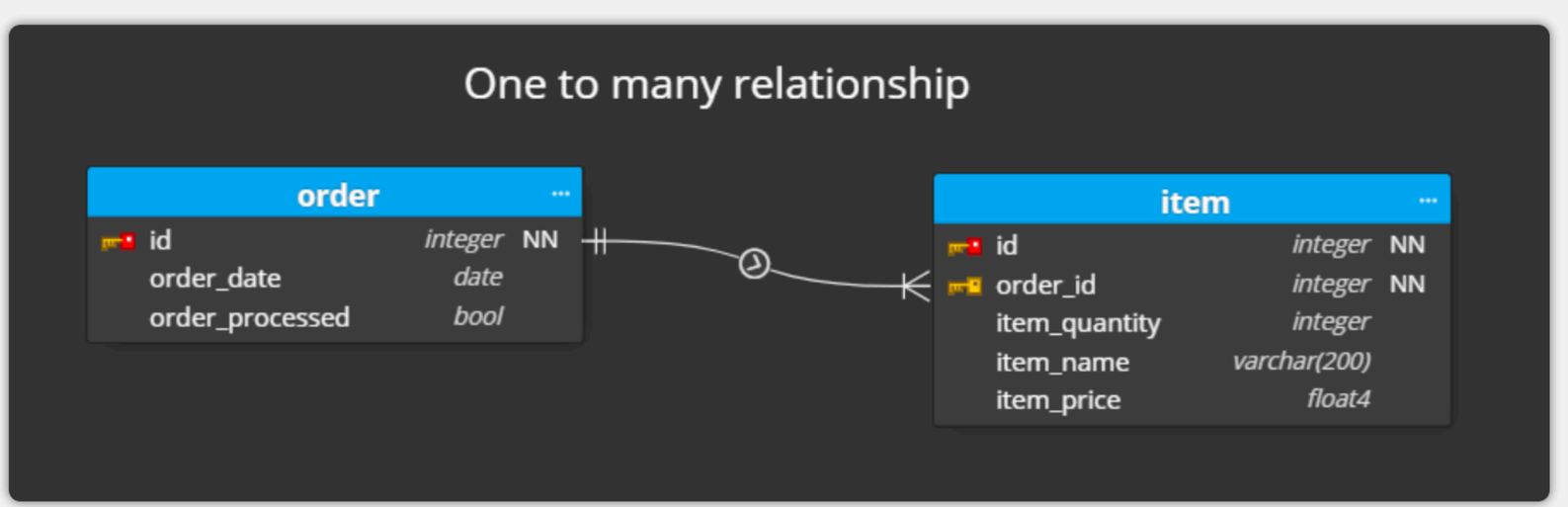
- Setup: Include a foreign key in one of the tables that references the primary key of the other table.
- For example: Tables users and user_profiles, where each user has a single corresponding profile.



One-To-Many Relationship

Each record in Table A can be associated with multiple records in Table B, but each record in Table B is associated with only one record in Table A.

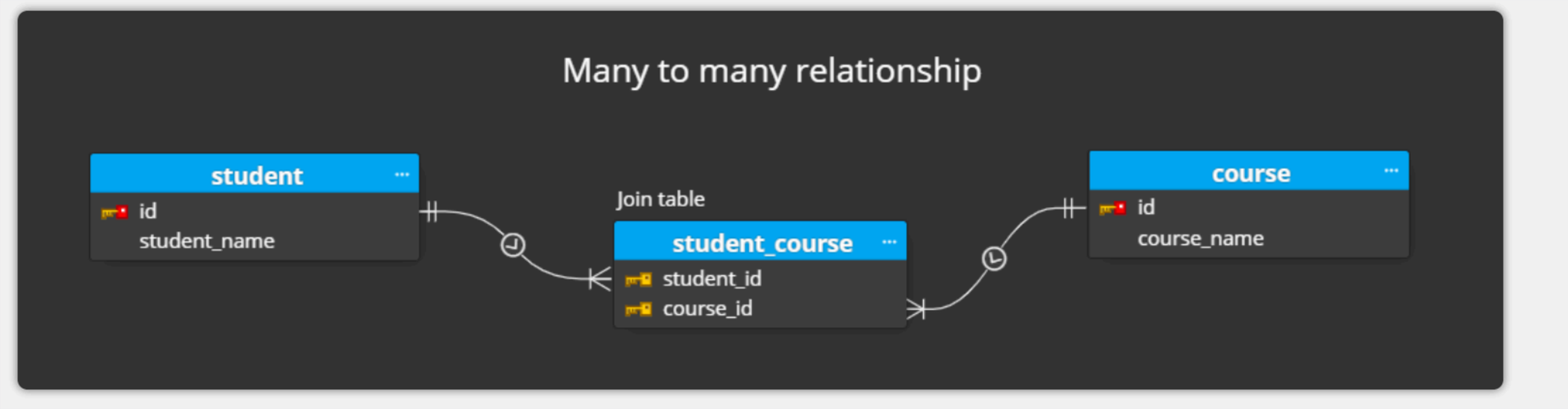
- Setup: Include a foreign key in the "many" side table (Table B) that references the primary key of the "one" side table (Table A).
- For example: Tables departments and employees, where each department can have multiple employees, but each employee belongs to one department.



Many-To-Many Relationship

Each record in Table A can be associated with multiple records in Table B, and vice versa.

- Setup: Create an intermediate table (also known as a junction or linking table) that contains foreign keys referencing both related tables.
- For example: Tables students and courses, where each student can enroll in multiple courses, and each course can have multiple students.



Navigation Property

When you have two tables in a database, they can be related to each other. For example:

- Students table
- Courses table

A student can take many courses, and a course can have many students (Many to Many relationship)

- In C# classes, a navigation property is a property that points to another class to represent a relationship.
- It allows you to navigate from one object to related objects without writing SQL.

Eager Loading and Lazy Loading

When you query a table in EF Core, sometimes you also want to load related data (via navigation properties).

EF Core provides two main ways to load related data: Eager Loading and Lazy Loading.

Eager Loading (Load immediately)

- Definition: EF Core loads the main entity and its related data in a single query.
- Use when: You know you will need the related data immediately.

Lazy Loading (Load on demand)

- Definition: EF Core loads the main entity first, and loads related data only when you access it.
- Use when: You might not always need the related data.

Thank you for listening!