Table Relations - Exercises

[1.](#_gjdgxs) Specification: 1

[2.](#_30j0zll) Identifying Entities and Relationships 1

[3.](#_1fob9te) Designing Tables with Primary and Foreign Keys 1

[4.](#_3znysh7) Implementing One-to-Many Relationships 2

[5.](#_2et92p0) Implementing Many-to-Many Relationships 2

[6.](#_tyjcwt) Cascade Delete 2

[7.](#_3dy6vkm) Cascade Update 2

[8.](#_1t3h5sf) Writing JOIN Queries 3

[9.](#_4d34og8) Database Normalization 3

[10.](#_2s8eyo1) Visualizing Relationships 3

[11.](#_17dp8vu) QUERIES 3

# Specification:

Develop a system to manage information about employees, projects, and departments.

* Each employee has a name, ID, job title, and email.
* Each project has a name, description, and deadline.
* Each department has a name, ID, and manager.

Relationships:

* An employee can work on multiple projects, and each project can have multiple employees
* Each employee belongs to only one department, but a department can have multiple employees

# Identifying Entities and Relationships

Based on the new specification, identify the entities, their attributes, and relationships.

* What are the entities in this system?
* Define the attributes for each entity.
* Identify the type of relationships between entities (e.g., one-to-many, many-to-many).

# Designing Tables with Primary and Foreign Keys

Design the SQL schema for the entities and relationships identified in Exercise 1.

* Create tables for **Employees**, **Projects**, and **Departments**.
* Define primary keys and appropriate foreign keys.
* Write SQL statements to create these tables.

# Implementing One-to-Many Relationships

**Scenario:** Each employee belongs to only one department, but a department can have multiple employees.

Update the schema to implement this one-to-many relationship using foreign keys.

Write the SQL statements for:

* Creating the updated **Employees** table.
* Adding foreign key constraints.

# Implementing Many-to-Many Relationships

**Scenario:** Employees can work on multiple projects, and each project can have multiple employees.

Design and implement a mapping table for this many-to-many relationship.

Write the SQL statements for:

* Creating a mapping table (e.g., **EmployeeProjects**).
* Establishing the necessary foreign key constraints.

# Cascade Delete

**Scenario:** If a department is deleted, all employees belonging to that department should also be deleted.

Modify the schema to include **ON DELETE CASCADE** for the relevant foreign key.

Write the SQL statement to implement this behavior in the **Employees** table.

# Cascade Update

**Scenario:** If the **DepartmentID** of a department is updated, the change should be reflected in the **Employees** table.

Modify the schema to include **ON UPDATE CASCADE** for the relevant foreign key.

Write the SQL statement to implement this behavior.

# Writing JOIN Queries

Write SQL queries to perform the following:

* Retrieve all employees working on a specific project.
* List all projects assigned to employees in a specific department.
* Display all departments along with the names of their employees.

# Database Normalization

**Scenario:** A table stores employee details and the names of projects they are working on. The data is as follows:

| **EmployeeID** | **EmployeeName** | **ProjectName** |
| --- | --- | --- |
| 1 | Alice Brown | Project Alpha |
| 2 | Bob Smith | Project Beta |
| 1 | Alice Brown | Project Beta |

1. Identify the anomalies in this design.
2. Normalize the data to 3rd Normal Form (3NF).
3. Write SQL statements to create the normalized tables.

# Visualizing Relationships

Using SSMS or any database design tool:

1. Create an Entity-Relationship (E/R) Diagram for the system.
2. Show all relationships, including one-to-many and many-to-many.

# QUERIES

**Use sirmadb.sql to create the database.**

**List All Employees**

Write a query to retrieve all employees' full names, job titles, and their salaries.

**Employees by Department**Write a query to display the names of employees and their departments.

**List Employees in a Specific Location**Retrieve the names and job titles of employees working in the "UK Branch".

**Highest Paid Employee**Find the employee with the highest salary.

**Average Salary by Department**Calculate the average salary for each department.

**Employees Without a Manager**Retrieve the names of employees who are not managers.

**Departments and Their Managers**Display each department name along with the full name of its manager.

**Employees in a Specific Country**Write a query to find all employees working in offices located in "Bulgaria".

**Total Salaries by Country**Calculate the total salaries of employees working in each country.

**Employees Earning Above Average Salary**Find employees who earn above the average salary for the company.