

CREATE DATABASE mydb;

USE mydb;

CREATE TABLE Employees (

Emp\_ID INT PRIMARY KEY,

Emp\_Name VARCHAR(50),

Dept\_ID INT,

Dept\_Name VARCHAR(50),

Manager\_ID INT,

Manager\_Name VARCHAR(50),

Salary DECIMAL(10, 2),

Project\_IDs VARCHAR(100),

Project\_Names VARCHAR(100)

);

SELECT \* FROM Employees;

INSERT INTO Employees (Emp\_ID, Emp\_Name, Dept\_ID, Dept\_Name, Manager\_ID, Manager\_Name, Salary, Project\_IDs, Project\_Names) VALUES

(101, 'John', 1, 'HR', 201, 'Alice', 50000, 'P01,P02', 'Recruit,Payroll'),

(102, 'Sarah', 1, 'HR', 201, 'Alice', 52000, 'P01', 'Recruit'),

(103, 'Mike', 2, 'IT', 202, 'Bob', 60000, 'P03', 'Infrastructure'),

(104, 'Linda', 3, 'Finance', 203, 'Charlie', 58000, 'P04,P05', 'Audit,Budgeting'),

(105, 'Tom', 2, 'IT', 202, 'Bob', 61000, 'P03', 'Infrastructure'),

(106, 'Steve', 4, 'Marketing', 204, 'Daisy', 53000, 'P06', 'Campaign'),

(107, 'Emma', 3, 'Finance', 203, 'Charlie', 59000, 'P04', 'Audit'),

(108, 'Rachel', 4, 'Marketing', 204, 'Daisy', 54000, 'P06,P07', 'Campaign,Analytics'),

(109, 'Kevin', 2, 'IT', 202, 'Bob', 62000, 'P08', 'Cloud Migration'),

(110, 'David', 1, 'HR', 201, 'Alice', 51000, 'P02', 'Payroll');

Redundancy

Dept\_Name, Manager\_Name, and Project\_Names are repeated.

Alice is mentioned repeatedly for HR employees.

Anomalies

Update anomaly:

If Manager Alice changes her name, we must update multiple rows.

Insert anomaly:

Can't add a new project without assigning an employee.

Can't add a department without at least one employee.

Delete anomaly:

If the only employee in Finance leaves, we lose data about the Finance department.

Non-atomic values:

Project\_IDs and Project\_Names contain multiple values in one column (comma-separated).

First Normal Form (1NF)

Goal: Eliminate multivalued and composite attributes. Ensure all fields are atomic.

Fix: Split multiple project entries into separate rows.

CREATE TABLE Employees (

Emp\_ID INT,

Emp\_Name VARCHAR(50),

Dept\_ID INT,

Dept\_Name VARCHAR(50),

Manager\_ID INT,

Manager\_Name VARCHAR(50),

Salary DECIMAL(10, 2),

Project\_ID VARCHAR(10),

Project\_Name VARCHAR(50),

PRIMARY KEY (Emp\_ID, Project\_ID)

);

SELECT \* FROM Employees;

INSERT INTO Employees2 (Emp\_ID, Emp\_Name, Dept\_ID, Dept\_Name, Manager\_ID, Manager\_Name, Salary, Project\_ID, Project\_Name) VALUES

(101, 'John', 1, 'HR', 201, 'Alice', 50000, 'P01', 'Recruit'),

(101, 'John', 1, 'HR', 201, 'Alice', 50000, 'P02', 'Payroll'),

(102, 'Sarah', 1, 'HR', 201, 'Alice', 52000, 'P01', 'Recruit'),

(103, 'Mike', 2, 'IT', 202, 'Bob', 60000, 'P03', 'Infrastructure'),

(104, 'Linda', 3, 'Finance', 203, 'Charlie', 58000, 'P04', 'Audit'),

(104, 'Linda', 3, 'Finance', 203, 'Charlie', 58000, 'P05', 'Budgeting'),

(105, 'Tom', 2, 'IT', 202, 'Bob', 61000, 'P03', 'Infrastructure'),

(106, 'Steve', 4, 'Marketing', 204, 'Daisy', 53000, 'P06', 'Campaign'),

(107, 'Emma', 3, 'Finance', 203, 'Charlie', 59000, 'P04', 'Audit'),

(108, 'Rachel', 4, 'Marketing', 204, 'Daisy', 54000, 'P06', 'Campaign'),

(108, 'Rachel', 4, 'Marketing', 204, 'Daisy', 54000, 'P07', 'Analytics'),

(109, 'Kevin', 2, 'IT', 202, 'Bob', 62000, 'P08', 'Cloud Migration'),

(110, 'David', 1, 'HR', 201, 'Alice', 51000, 'P02', 'Payroll');

Second Normal Form (2NF)

Goal: Remove partial dependencies. Every non-key attribute must depend on the entire composite key.

In 1NF, Emp\_ID + Project\_ID is a composite key, but:

Emp\_Name, Dept\_Name, Manager\_Name, Salary depend only on Emp\_ID, not Project\_ID.

Split into tables:

Employee(Emp\_ID, Emp\_Name, Dept\_ID, Manager\_ID, Salary)

Department(Dept\_ID, Dept\_Name)

Manager(Manager\_ID, Manager\_Name)

Project(Project\_ID, Project\_Name)

Employee\_Project(Emp\_ID, Project\_ID)

SELECT \* FROM EMPLOYEES

Uniqueness:

Each Emp\_ID corresponds to a unique set of values for Emp\_Name, Dept\_ID, Manager\_ID, and Salary. No two employees will have the same Emp\_ID.

Data Integrity: Ensures that the data is consistent and accurate. If you know the Emp\_ID, you can reliably determine the employee's name, department, manager, and salary.

Functional Dependency: This notation indicates that the value of Emp\_ID uniquely determines the values of Emp\_Name, Dept\_ID, Manager\_ID, and Salary.

Dependent Attributes: Emp\_Name, Dept\_ID, Manager\_ID, and Salary are dependent attributes, meaning their values are determined by the value of Emp\_ID.

Department Table

Schema: Department(Dept\_ID, Dept\_Name)

Primary Key: Dept\_ID

Relationship: One-to-Many with Employee

Explanation: Each department can have multiple employees, but each employee belongs to only one department.

Manager Table

Schema: Manager(Manager\_ID, Manager\_Name)

Primary Key: Manager\_ID

Relationship: One-to-Many with Employee

Explanation: Each manager can manage multiple employees, but each employee is managed by only one manager.

Project Table

Schema: Project(Project\_ID, Project\_Name)

Primary Key: Project\_ID

Relationship: Many-to-Many with Employee via Employee\_Project

Explanation: Each project can involve multiple employees, and each employee can work on multiple projects.

Employee Table

Schema: Employee(Emp\_ID, Emp\_Name, Dept\_ID, Manager\_ID, Salary)

Primary Key: Emp\_ID

Foreign Keys: Dept\_ID (references Department), Manager\_ID (references Manager)

Relationships:

One-to-Many with Department: Each employee belongs to one department.

One-to-Many with Manager: Each employee is managed by one manager.

Many-to-Many with Project via Employee\_Project: Each employee can work on multiple projects.

Employee\_Project Table

Schema: Employee\_Project(Emp\_ID, Project\_ID)

Primary Key: Composite key (Emp\_ID, Project\_ID)

Foreign Keys: Emp\_ID (references Employee), Project\_ID (references Project)

Relationship: Many-to-Many between Employee and Project

Explanation: This table establishes a many-to-many relationship, where each row represents an assignment of an employee to a project. An employee can be assigned to multiple projects, and a project can have multiple employees.

CREATE TABLE Department (

Dept\_ID INT PRIMARY KEY,

Dept\_Name VARCHAR(100) NOT NULL

);

CREATE TABLE Manager (

Manager\_ID INT PRIMARY KEY,

Manager\_Name VARCHAR(100) NOT NULL

);

CREATE TABLE Project (

Project\_ID VARCHAR(10) PRIMARY KEY,

Project\_Name VARCHAR(100) NOT NULL

);

CREATE TABLE Employee (

Emp\_ID INT PRIMARY KEY,

Emp\_Name VARCHAR(100) NOT NULL,

Dept\_ID INT NOT NULL,

Manager\_ID INT NOT NULL,

Salary DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (Dept\_ID) REFERENCES Department(Dept\_ID),

FOREIGN KEY (Manager\_ID) REFERENCES Manager(Manager\_ID)

);

CREATE TABLE Employee\_Project (

Emp\_ID INT NOT NULL,

Project\_ID VARCHAR(10) NOT NULL,

PRIMARY KEY (Emp\_ID, Project\_ID),

FOREIGN KEY (Emp\_ID) REFERENCES Employee(Emp\_ID),

FOREIGN KEY (Project\_ID) REFERENCES Project(Project\_ID)

);

Third Normal Form (3NF)

Goal: Remove transitive dependencies.

Elimination of Transitive Dependencies: Ensures that non-key attributes are not dependent on other non-key attributes.

Manager\_Name depends on Manager\_ID, which is not a primary key in Employee. Same for Dept\_Name.

Already split into separate tables — so now it's in 3NF.

INSERT INTO Department (Dept\_ID, Dept\_Name) VALUES

(1, 'HR'),

(2, 'IT'),

(3, 'Finance'),

(4, 'Marketing');

INSERT INTO Manager (Manager\_ID, Manager\_Name) VALUES

(201, 'Alice'),

(202, 'Bob'),

(203, 'Charlie'),

(204, 'Daisy');

INSERT INTO Project (Project\_ID, Project\_Name) VALUES

('P01', 'Recruit'),

('P02', 'Payroll'),

('P03', 'Infrastructure'),

('P04', 'Audit'),

('P05', 'Budgeting'),

('P06', 'Campaign'),

('P07', 'Analytics'),

('P08', 'Cloud Migration');

INSERT INTO Employee (Emp\_ID, Emp\_Name, Dept\_ID, Manager\_ID, Salary) VALUES

(101, 'John', 1, 201, 50000),

(102, 'Sarah', 1, 201, 52000),

(103, 'Mike', 2, 202, 60000),

(104, 'Linda', 3, 203, 58000),

(105, 'Tom', 2, 202, 61000),

(106, 'Steve', 4, 204, 53000),

(107, 'Emma', 3, 203, 59000),

(108, 'Rachel', 4, 204, 54000),

(109, 'Kevin', 2, 202, 62000),

(110, 'David', 1, 201, 51000);

INSERT INTO Employee\_Project (Emp\_ID, Project\_ID) VALUES

(101, 'P01'),

(101, 'P02'),

(102, 'P01'),

(103, 'P03'),

(104, 'P04'),

(104, 'P05'),

(105, 'P03'),

(106, 'P06'),

(107, 'P04'),

(108, 'P06'),

(108, 'P07'),

(109, 'P08'),

(110, 'P02');

**Assignment Operator (=)**

**Get the manager whose ID is 101.**

SELECT Manager\_ID, Manager\_Name

FROM Manager

WHERE Manager\_ID = 101;

**Find employees who belong to department ID 2.**

SELECT Emp\_ID, Emp\_Name

FROM Employee

WHERE Dept\_ID = 2;

**Arithmetic Operators (+, -, \*, /)**

**Show each employee's salary after a 10% bonus.**

SELECT Emp\_ID, Emp\_Name, Salary, Salary \* 1.10 AS Salary\_With\_Bonus

FROM Employee;

**Calculate the difference between each employee's salary and 100,000.**

SELECT Emp\_ID, Emp\_Name, Salary, 100000 - Salary AS Salary\_Difference

FROM Employee;

**Comparison Operators (>, <, >=, <=, !=)**

**Find employees earning more than 80,000.**

SELECT Emp\_ID, Emp\_Name, Salary

FROM Employee

WHERE Salary > 80000;

**List departments with IDs not equal to 1.**

SELECT Dept\_ID, Dept\_Name

FROM Department

WHERE Dept\_ID != 1;

**Logical Operators (AND, OR, NOT)**

**Find employees in department 2 with salary over 70,000.**

SELECT Emp\_ID, Emp\_Name, Salary

FROM Employee

WHERE Dept\_ID = 2 AND Salary > 70000;

**Get employees who are either in department 1 or earn less than 60,000.**

SELECT Emp\_ID, Emp\_Name, Dept\_ID, Salary

FROM Employee

WHERE Dept\_ID = 1 OR Salary < 60000;

**IN Operator**

**Find employees in departments 1, 2, or 3.**

SELECT Emp\_ID, Emp\_Name, Dept\_ID

FROM Employee

WHERE Dept\_ID IN (1, 2, 3);

**Get projects with IDs in a specific list.**

SELECT Project\_ID, Project\_Name

FROM Project

WHERE Project\_ID IN ('PRJ001', 'PRJ002', 'PRJ005');

**BETWEEN Operator**

**Find employees with salaries between 60,000 and 90,000.**

SELECT Emp\_ID, Emp\_Name, Salary

FROM Employee

WHERE Salary BETWEEN 60000 AND 90000;

**NOT IN Operator**

**Find employees not in departments 1 or 3.**

SELECT Emp\_ID, Emp\_Name, Dept\_ID

FROM Employee

WHERE Dept\_ID NOT IN (1, 3);

**Get projects not in a given list.**

SELECT Project\_ID, Project\_Name

FROM Project

WHERE Project\_ID NOT IN ('PRJ001', 'PRJ003');

**NOT BETWEEN Operator**

SELECT Emp\_ID, Emp\_Name, Salary

FROM Employee

WHERE Salary NOT BETWEEN 50000 AND 80000;

**Retrieve all employees who earn more than 70,000.**

SELECT Emp\_Name, Salary

FROM Employee

WHERE Salary > 70000;

**List all employees ordered by their salary in descending order.**

SELECT Emp\_ID, Emp\_Name, Salary

FROM Employee

ORDER BY Salary DESC;

**Display all departments sorted alphabetically by department name.**

SELECT Dept\_ID, Dept\_Name

FROM Department

ORDER BY Dept\_Name ASC;

**Show all managers sorted by their names in reverse alphabetical order.**

SELECT Manager\_ID, Manager\_Name

FROM Manager

ORDER BY Manager\_Name DESC;

**List all projects ordered by project name.**

SELECT Project\_ID, Project\_Name

FROM Project

ORDER BY Project\_Name ASC;

**Display employees sorted by department ID and then by salary (lowest to highest) within each department.**

SELECT Emp\_ID, Emp\_Name, Dept\_ID, Salary

FROM Employee

ORDER BY Dept\_ID ASC, Salary ASC;

**Get the top 5 highest-paid employees.**

SELECT TOP 5 Emp\_ID, Emp\_Name, Salary

FROM Employee

ORDER BY Salary DESC;

**Retrieve the top 3 departments based on alphabetical order.**

SELECT TOP 3 Dept\_ID, Dept\_Name

FROM Department

ORDER BY Dept\_Name ASC;

**Get the top 2 employees with the lowest salaries.**

SELECT TOP 2 Emp\_ID, Emp\_Name, Salary

FROM Employee

ORDER BY Salary ASC;

**Count the number of employees in each department.**

SELECT Dept\_ID, COUNT(\*) AS Employee\_Count

FROM Employee

GROUP BY Dept\_ID;

**Find the average salary of employees under each manager.**

SELECT Manager\_ID, AVG(Salary) AS Average\_Salary

FROM Employee

GROUP BY Manager\_ID;

**Count how many employees are assigned to each project.**

SELECT Project\_ID, COUNT(\*) AS Employee\_Count

FROM Employee\_Project

GROUP BY Project\_ID;

**Count the number of projects each employee is working on.**

SELECT Emp\_ID, COUNT(\*) AS Project\_Count

FROM Employee\_Project

GROUP BY Emp\_ID;

**Count how many departments have the same name (to check for duplicates).**

SELECT Dept\_Name, COUNT(\*) AS Dept\_Count

FROM Department

GROUP BY Dept\_Name

HAVING COUNT(\*) > 1;

**Find departments that have more than 5 employees**

SELECT Dept\_ID, COUNT(\*) AS Employee\_Count

FROM Employee

GROUP BY Dept\_ID

HAVING COUNT(\*) > 5;

**List managers who manage employees with an average salary greater than 80,000.**

SELECT Manager\_ID, AVG(Salary) AS Avg\_Salary

FROM Employee

GROUP BY Manager\_ID

HAVING AVG(Salary) > 80000;

**Show employees who are assigned to more than 3 projects.**

SELECT Emp\_ID, COUNT(\*) AS Project\_Count

FROM Employee\_Project

GROUP BY Emp\_ID

HAVING COUNT(\*) > 3;

**Find projects that have more than 2 employees assigned.**

SELECT Project\_ID, COUNT(\*) AS Employee\_Count

FROM Employee\_Project

GROUP BY Project\_ID

HAVING COUNT(\*) > 2;

**Identify departments where the total salary of employees exceeds 500,000.**

SELECT Dept\_ID, SUM(Salary) AS Total\_Salary

FROM Employee

GROUP BY Dept\_ID

HAVING SUM(Salary) > 500000;

**List all employees with their department names.**

SELECT

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Employee E

INNER JOIN Department D ON E.Dept\_ID = D.Dept\_ID;

**List all employees with their manager names.**

SELECT

    E.Emp\_Name,

    M.Manager\_Name

FROM

    Employee E

INNER JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List employees who are assigned to at least one project.**

SELECT

    E.Emp\_Name,

    EP.Project\_ID

FROM

    Employee E

INNER JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID;

**List all employees with their department and manager**

SELECT

    E.Emp\_Name,

    D.Dept\_Name,

    M.Manager\_Name

FROM

    Employee E

INNER JOIN Department D ON E.Dept\_ID = D.Dept\_ID

INNER JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all projects that have employees assigned, along with employee and department names.**

SELECT

    P.Project\_Name,

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Project P

INNER JOIN Employee\_Project EP ON P.Project\_ID = EP.Project\_ID

INNER JOIN Employee E ON EP.Emp\_ID = E.Emp\_ID

INNER JOIN Department D ON E.Dept\_ID = D.Dept\_ID;

**Find employees who are working on projects and have a salary above 60,000.**

SELECT

    E.Emp\_Name,

    P.Project\_Name,

    E.Salary

FROM

    Employee E

INNER JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

INNER JOIN Project P ON EP.Project\_ID = P.Project\_ID

WHERE

    E.Salary > 60000;

**List managers and their employees who are assigned to projects, but only if the employee has a department assigned (i.e., Dept\_ID is not NULL).**

SELECT

    M.Manager\_Name,

    E.Emp\_Name,

    P.Project\_Name

FROM

    Manager M

INNER JOIN Employee E ON M.Manager\_ID = E.Manager\_ID

INNER JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

INNER JOIN Project P ON EP.Project\_ID = P.Project\_ID

WHERE

    E.Dept\_ID IS NOT NULL;

**List all employees and their department names, including employees without a department.**

SELECT

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Employee E

LEFT JOIN Department D ON E.Dept\_ID = D.Dept\_ID;

**List all employees and their manager names, including employees without a manager.**

SELECT

    E.Emp\_Name,

    M.Manager\_Name

FROM

    Employee E

LEFT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all employees and the projects they are assigned to, including employees not assigned to any project.**

SELECT

    E.Emp\_Name,

    EP.Project\_ID

FROM

    Employee E

LEFT JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID;

**List all projects and the employees assigned to them, including projects with no employees**

SELECT

    P.Project\_Name,

    EP.Emp\_ID

FROM

    Project P

LEFT JOIN Employee\_Project EP ON P.Project\_ID = EP.Project\_ID;

**Employees and their departments (simulating INNER JOIN)**

SELECT

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Employee E

LEFT JOIN Department D ON E.Dept\_ID = D.Dept\_ID

WHERE D.Dept\_ID IS NOT NULL;

**Projects and assigned employees (simulating INNER JOIN)**

SELECT

    P.Project\_Name,

    EP.Emp\_ID

FROM

    Project P

LEFT JOIN Employee\_Project EP ON P.Project\_ID = EP.Project\_ID

WHERE EP.Emp\_ID IS NOT NULL;

**List all departments and the employees in them, including departments with no employees.**

SELECT

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Employee E

RIGHT JOIN Department D ON E.Dept\_ID = D.Dept\_ID;

**List all managers and the employees they manage, including managers with no employees.**

SELECT

    E.Emp\_Name,

    M.Manager\_Name

FROM

    Employee E

RIGHT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all employees and their assigned projects, including projects with no employees.**

SELECT

    E.Emp\_Name,

    P.Project\_Name

FROM

    Employee\_Project EP

RIGHT JOIN Project P ON EP.Project\_ID = P.Project\_ID

LEFT JOIN Employee E ON EP.Emp\_ID = E.Emp\_ID;

**List all departments and the employees in them, including departments with no employees, and show their managers if available.**

SELECT

    D.Dept\_Name,

    E.Emp\_Name,

    M.Manager\_Name

FROM

    Employee E

RIGHT JOIN Department D ON E.Dept\_ID = D.Dept\_ID

LEFT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all employees with their department and manager names, including employees even if their department or manager is missing.**

SELECT

    E.Emp\_Name,

    D.Dept\_Name,

    M.Manager\_Name

FROM

    Employee E

LEFT JOIN Department D ON E.Dept\_ID = D.Dept\_ID

LEFT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

SELECT

    E.Emp\_Name,

    D.Dept\_Name,

    M.Manager\_Name

FROM

    Employee E

LEFT JOIN Department D ON E.Dept\_ID = D.Dept\_ID

LEFT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

SELECT

    E.Emp\_Name,

    P.Project\_Name

FROM

    Employee E

LEFT JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

LEFT JOIN Project P ON EP.Project\_ID = P.Project\_ID;

**List all departments and their employees with their managers, including departments with no employees.**

SELECT

    D.Dept\_Name,

    E.Emp\_Name,

    M.Manager\_Name

FROM

    Department D

LEFT JOIN Employee E ON D.Dept\_ID = E.Dept\_ID

LEFT JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all departments and employees in them, including departments with no employees.**

SELECT

    E.Emp\_Name,

    D.Dept\_Name

FROM

    Employee E

RIGHT JOIN Department D ON E.Dept\_ID = D.Dept\_ID;

**List all projects and the employees assigned to them, including projects with no employees.**

SELECT

    E.Emp\_Name,

    P.Project\_Name

FROM

    Employee\_Project EP

RIGHT JOIN Project P ON EP.Project\_ID = P.Project\_ID

LEFT JOIN Employee E ON EP.Emp\_ID = E.Emp\_ID;

**List all employees and their assigned projects, including employees without projects and projects without employees.**

SELECT

    E.Emp\_Name,

    P.Project\_Name

FROM

    Employee E

FULL OUTER JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

FULL OUTER JOIN Project P ON EP.Project\_ID = P.Project\_ID;

**List all departments and employees, including departments with no employees and employees with invalid departments.**

SELECT

    D.Dept\_Name,

    E.Emp\_Name

FROM

    Department D

FULL OUTER JOIN Employee E ON D.Dept\_ID = E.Dept\_ID;

**List all managers and employees, including managers with no employees and employees with invalid managers.**

SELECT

    M.Manager\_Name,

    E.Emp\_Name

FROM

    Manager M

FULL OUTER JOIN Employee E ON M.Manager\_ID = E.Manager\_ID;

**List employees with their department and manager names**

SELECT E.Emp\_Name, D.Dept\_Name, M.Manager\_Name

FROM Employee E

JOIN Department D ON E.Dept\_ID = D.Dept\_ID

JOIN Manager M ON E.Manager\_ID = M.Manager\_ID;

**List all employees working on projects**

SELECT E.Emp\_Name, P.Project\_Name

FROM Employee E

JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

JOIN Project P ON EP.Project\_ID = P.Project\_ID;

**List employees working in IT department and their projects**

SELECT E.Emp\_Name, P.Project\_Name

FROM Employee E

JOIN Department D ON E.Dept\_ID = D.Dept\_ID

JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

JOIN Project P ON EP.Project\_ID = P.Project\_ID

WHERE D.Dept\_Name = 'IT';

**Find employees working on more than one project**

SELECT E.Emp\_Name, COUNT(EP.Project\_ID) AS Project\_Count

FROM Employee E

JOIN Employee\_Project EP ON E.Emp\_ID = EP.Emp\_ID

GROUP BY E.Emp\_Name

HAVING COUNT(EP.Project\_ID) > 1;

**Total Salary by Department Name**

SELECT d.Dept\_Name, SUM(e.Salary) AS Total\_Salary

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

GROUP BY d.Dept\_Name;

**Number of Employees per Project**

SELECT p.Project\_Name, COUNT(ep.Emp\_ID) AS Employee\_Count

FROM Employee\_Project ep

JOIN Project p ON ep.Project\_ID = p.Project\_ID

GROUP BY p.Project\_Name;

**Average Salary per Project (via Employee\_Project)**

SELECT p.Project\_Name, AVG(e.Salary) AS Avg\_Salary

FROM Employee\_Project ep

JOIN Employee e ON ep.Emp\_ID = e.Emp\_ID

JOIN Project p ON ep.Project\_ID = p.Project\_ID

GROUP BY p.Project\_Name;

**Total Salary by Department and Manager**

SELECT d.Dept\_Name, m.Manager\_Name, SUM(e.Salary) AS Total\_Salary

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

JOIN Manager m ON e.Manager\_ID = m.Manager\_ID

GROUP BY d.Dept\_Name, m.Manager\_Name;

**Number of Projects per Department**

SELECT d.Dept\_Name, COUNT(DISTINCT ep.Project\_ID) AS Project\_Count

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

JOIN Employee\_Project ep ON e.Emp\_ID = ep.Emp\_ID

GROUP BY d.Dept\_Name;

**Average Salary per Project and Manager**

SELECT p.Project\_Name, m.Manager\_Name, AVG(e.Salary) AS Avg\_Salary

FROM Employee\_Project ep

JOIN Employee e ON ep.Emp\_ID = e.Emp\_ID

JOIN Project p ON ep.Project\_ID = p.Project\_ID

JOIN Manager m ON e.Manager\_ID = m.Manager\_ID

GROUP BY p.Project\_Name, m.Manager\_Name;

**Departments with More Than 3 Projects**

SELECT d.Dept\_Name, COUNT(DISTINCT ep.Project\_ID) AS Project\_Count

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

JOIN Employee\_Project ep ON e.Emp\_ID = ep.Emp\_ID

GROUP BY d.Dept\_Name

HAVING COUNT(DISTINCT ep.Project\_ID) > 3;

**Employees with the Same Manager**

SELECT e1.Emp\_Name AS Employee1, e2.Emp\_Name AS Employee2, m.Manager\_Name

FROM Employee e1

JOIN Employee e2 ON e1.Manager\_ID = e2.Manager\_ID AND e1.Emp\_ID < e2.Emp\_ID

JOIN Manager m ON e1.Manager\_ID = m.Manager\_ID;

**Employees in the Same Department**

SELECT e1.Emp\_Name AS Employee1, e2.Emp\_Name AS Employee2, d.Dept\_Name

FROM Employee e1

JOIN Employee e2 ON e1.Dept\_ID = e2.Dept\_ID AND e1.Emp\_ID < e2.Emp\_ID

JOIN Department d ON e1.Dept\_ID = d.Dept\_ID;

**Employees with Higher Salary than Their Peers in the Same Department**

SELECT e1.Emp\_Name AS Higher\_Paid, e2.Emp\_Name AS Lower\_Paid, d.Dept\_Name

FROM Employee e1

JOIN Employee e2 ON e1.Dept\_ID = e2.Dept\_ID AND e1.Salary > e2.Salary

JOIN Department d ON e1.Dept\_ID = d.Dept\_ID;

**List All Employees with Their Department and Manager**

SELECT

    e.Emp\_Name,

    d.Dept\_Name,

    m.Manager\_Name

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

JOIN Manager m ON e.Manager\_ID = m.Manager\_ID;

**List All Projects with Assigned Employees**

SELECT

    p.Project\_Name,

    e.Emp\_Name

FROM Employee\_Project ep

JOIN Employee e ON ep.Emp\_ID = e.Emp\_ID

JOIN Project p ON ep.Project\_ID = p.Project\_ID

ORDER BY p.Project\_Name;

**Total Salary Paid per Project**

SELECT

    p.Project\_Name,

    SUM(e.Salary) AS Total\_Salary

FROM Employee\_Project ep

JOIN Employee e ON ep.Emp\_ID = e.Emp\_ID

JOIN Project p ON ep.Project\_ID = p.Project\_ID

GROUP BY p.Project\_Name;

**Departments with Their Managers and Number of Employees**

SELECT

    d.Dept\_Name,

    m.Manager\_Name,

    COUNT(e.Emp\_ID) AS Employee\_Count

FROM Employee e

JOIN Department d ON e.Dept\_ID = d.Dept\_ID

JOIN Manager m ON e.Manager\_ID = m.Manager\_ID

GROUP BY d.Dept\_Name, m.Manager\_Name;

**Employees Not Assigned to Any Project**

SELECT

    e.Emp\_Name

FROM Employee e

LEFT JOIN Employee\_Project ep ON e.Emp\_ID = ep.Emp\_ID

WHERE ep.Project\_ID IS NULL;

**List All Unique Names from Employees and Managers**

SELECT Emp\_Name AS Name FROM Employee

UNION

SELECT Manager\_Name AS Name FROM Manager;

**UNION ALL: List All Names from Employees and Managers (Including Duplicates)**

SELECT Emp\_Name AS Name FROM Employee

UNION ALL

SELECT Manager\_Name AS Name FROM Manager;

**EXCEPT: Employees Not Assigned to Any Project**

SELECT Emp\_ID FROM Employee

EXCEPT

SELECT Emp\_ID FROM Employee\_Project;

**INTERSECT: Employees Assigned to Projects and Still in the Employee Table**

SELECT Emp\_ID FROM Employee

INTERSECT

SELECT Emp\_ID FROM Employee\_Project;

**Projects With No Employees Assigned**

SELECT Project\_ID FROM Project

EXCEPT

SELECT Project\_ID FROM Employee\_Project;

**Employees Earning More Than the Average Salary**

SELECT Emp\_Name, Salary

FROM Employee

WHERE Salary > (SELECT AVG(Salary) FROM Employee);

SELECT Emp\_Name, Salary

FROM Employee

WHERE Salary > (SELECT AVG(Salary) FROM Employee);

**Departments with More Than 5 Employees**

SELECT Dept\_Name

FROM Department

WHERE Dept\_ID IN (

    SELECT Dept\_ID

    FROM Employee

    GROUP BY Dept\_ID

    HAVING COUNT(\*) > 5

);

**Employees Not Assigned to Any Project**

SELECT Emp\_Name

FROM Employee

WHERE Emp\_ID NOT IN (

    SELECT Emp\_ID FROM Employee\_Project

);

**Projects with No Employees Assigned**

SELECT Project\_Name

FROM Project

WHERE Project\_ID NOT IN (

    SELECT Project\_ID FROM Employee\_Project

);

**Managers Managing More Than One Employee**

SELECT Manager\_Name

FROM Manager

WHERE Manager\_ID IN (

    SELECT Manager\_ID

    FROM Employee

    GROUP BY Manager\_ID

    HAVING COUNT(\*) > 1

);

**Employees Working on the Same Projects as a Specific Employee**

SELECT DISTINCT e.Emp\_Name

FROM Employee e

WHERE e.Emp\_ID IN (

    SELECT ep2.Emp\_ID

    FROM Employee\_Project ep1

    JOIN Employee\_Project ep2 ON ep1.Project\_ID = ep2.Project\_ID

    WHERE ep1.Emp\_ID = 101 AND ep2.Emp\_ID != 101

);

**Departments with the Highest Average Salary**

SELECT Dept\_Name

FROM Department

WHERE Dept\_ID = (

    SELECT Dept\_ID

    FROM Employee

    GROUP BY Dept\_ID

    ORDER BY AVG(Salary) DESC

    LIMIT 1

);

**Employees Who Are Also Managers**

SELECT Emp\_Name

FROM Employee

WHERE Emp\_Name IN (

    SELECT Manager\_Name FROM Manager

);

**Employees Assigned to All Projects**

SELECT Emp\_Name

FROM Employee

WHERE NOT EXISTS (

    SELECT Project\_ID FROM Project

    EXCEPT

    SELECT Project\_ID FROM Employee\_Project WHERE Employee.Emp\_ID = Employee\_Project.Emp\_ID

);

**Projects with the Most Employees**

SELECT Project\_Name

FROM Project

WHERE Project\_ID = (

    SELECT Project\_ID

    FROM Employee\_Project

    GROUP BY Project\_ID

    ORDER BY COUNT(\*) DESC

    LIMIT 1

);