MYSQL PROJECT PRESENTATION

Presented by *RojaPriya.Ch*

**Project Title**: Comprehensive Employee Management System

Objective:

Students will create and manage a complex SQL database designed to handle employee records, department information, salary details, project assignments, performance reviews, and more. This project will help students learn SQL commands, create tables, manage data, perform subqueries, joins, aggregate functions, and manage transactions, all with a substantial dataset.

Database Design:

=> Database Name: `employee\_management\_system`

-=>Tables:

**1. Departments**

🡪`dept\_id` (Primary Key, INT)

🡪`dept\_name` (VARCHAR)

🡪`location` (VARCHAR)

**2. Employees**

🡪 `emp\_id` (Primary Key, INT, AUTO\_INCREMENT)

🡪 `emp\_name` (VARCHAR)

🡪`dept\_id` (INT)

🡪 `position` (VARCHAR)

🡪`salary` (INT)

🡪`hire\_date` (DATE)

🡪`email` (VARCHAR)

🡪`phone` (VARCHAR)

3. **Salaries**

🡪 `salary\_id` (Primary Key, INT, AUTO\_INCREMENT)

🡪`emp\_id` (INT)

🡪 `base\_salary` (INT)

🡪 `bonus` (INT)

🡪`total\_salary` (Computed)

🡪 `pay\_date` (DATE)

4. **Projects**

🡪`project\_id` (Primary Key, INT, AUTO\_INCREMENT)

🡪 `project\_name` (VARCHAR)

🡪start\_date` (DATE)

🡪`end\_date` (DATE)

🡪`budget` (INT)

5. **EmployeeProjects**

🡪emp\_id` (INT)

🡪project\_id` (INT)

🡪`role` (VARCHAR)

6. **PerformanceReviews**

🡪 `review\_id` (Primary Key, INT, AUTO\_INCREMENT)

🡪 emp\_id` (INT)

🡪 `review\_date` (DATE)

🡪 `review\_score` (INT)

🡪`comments` (TEXT)

# Step-by-Step Implementation with Large Sample Data

# Step 1: Create the Database

```sql

CREATE DATABASE employee\_management\_system;

USE employee\_management\_system;

```

- Explanation:

- The `CREATE DATABASE` command creates a new database.

- The `USE` command switches to the `employee\_management\_system` database.

---

# Step 2: Create the Tables

1. Departments Table:

```sql

CREATE TABLE Departments (

dept\_id INT PRIMARY KEY,

dept\_name VARCHAR(50),

location VARCHAR(50)

);

```

- Explanation:

- The `Departments` table includes a unique `dept\_id`, the department's name, and its location.

2. Employees Table:

```sql

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY AUTO\_INCREMENT,

emp\_name VARCHAR(50) NOT NULL,

dept\_id INT,

position VARCHAR(50),

salary INT,

hire\_date DATE,

email VARCHAR(100),

phone VARCHAR(15)

);

```

- Explanation:

- The `Employees` table now includes additional columns for `email` and `phone` to store contact details.

3. Salaries Table:

```sql

CREATE TABLE Salaries (

salary\_id INT PRIMARY KEY AUTO\_INCREMENT,

emp\_id INT,

base\_salary INT,

bonus INT,

total\_salary AS (base\_salary + bonus),

pay\_date DATE

);

```

- Explanation:

- The `Salaries` table now includes a `pay\_date` column to track when salaries were paid.

4. Projects Table:

```sql

CREATE TABLE Projects (

project\_id INT PRIMARY KEY AUTO\_INCREMENT,

project\_name VARCHAR(100),

start\_date DATE,

end\_date DATE,

budget INT

);

```

- Explanation:

- The `Projects` table tracks projects with their names, start and end dates, and budget.

5. EmployeeProjects Table:

```sql

CREATE TABLE EmployeeProjects (

emp\_id INT,

project\_id INT,

role VARCHAR(50)

);

```

- Explanation:

- The `EmployeeProjects` table links employees to projects and assigns them roles.

6. PerformanceReviews Table:

```sql

CREATE TABLE PerformanceReviews (

review\_id INT PRIMARY KEY AUTO\_INCREMENT,

emp\_id INT,

review\_date DATE,

review\_score INT,

comments TEXT

);

```

- Explanation:

- The `PerformanceReviews` table stores reviews for each employee, with a score and comments.

---

# Step 3: Insert Extensive Sample Data

1. **Insert Data into Departments Table**:

-- Inserting 100 departments

INSERT INTO Departments (dept\_name, location) VALUES('HR', 'New York'), ('Finance', 'San Francisco'),

('Engineering', 'Seattle'),

('Marketing', 'Chicago'),

('Sales', 'Los Angeles'),

('IT', 'San Jose'),

('Support', 'Austin'),

('Legal', 'Boston'),

('R&D', 'Denver'),

('Administration', 'Dallas'),

('Operations', 'Houston'),

('Procurement', 'Phoenix'),

('Logistics', 'Philadelphia'),

('Customer Service', 'Orlando'),

('Quality Assurance', 'Atlanta'),

('Training', 'Miami'),

('Compliance', 'Las Vegas'),

('Public Relations', 'San Diego'),

('Business Development', 'Nashville'),

('Strategy', 'Tampa'),

('Corporate Affairs', 'Detroit'),

('Investor Relations', 'Indianapolis'),

('Corporate Communications', 'Charlotte'),

('Product Development', 'San Francisco'),

('Data Analytics', 'Seattle'),

('Content Management', 'New York'),

( 'Social Media', 'Chicago'),

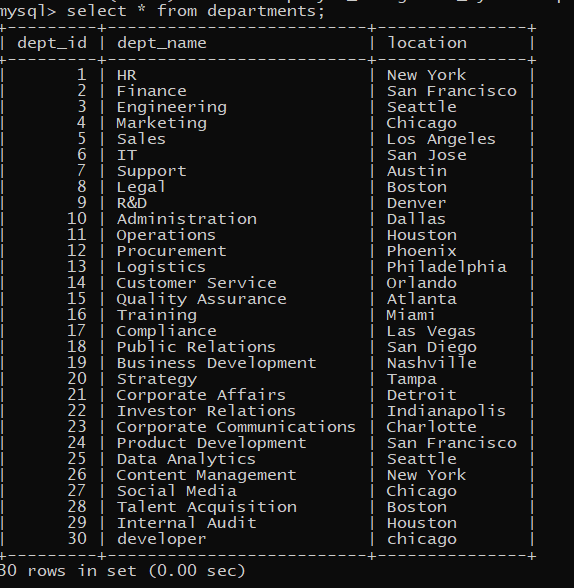
('Talent Acquisition', 'Boston'),

( 'Internal Audit', 'Houston'),

("developer",'chicago');

SELECT \* FROM Department;

OUTPUT:



-- Insert additional departments up to 100 with unique names and locations

INSERT INTO Departments (dept\_id, dept\_name, location)

SELECT

n,

CONCAT('Dept\_', n),

CASE

WHEN n % 2 = 0 THEN 'New York'

WHEN n % 3 = 0 THEN 'San Francisco'

ELSE 'Chicago'

END

FROM

(SELECT n FROM (SELECT @row := @row + 1 AS n FROM (SELECT 0 UNION ALL SELECT 1) t1 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t2 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t3 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t4 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t5 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t6, (SELECT @row := 0) r) num LIMIT 95) as nums;

```

- Explanation:

- Adds 100 departments with unique names and locations.

2. **Insert Data into Employees Table**:

-- Inserting 1000 employees with random data

CREATE TABLE Employees (

emp\_id INT PRIMARY KEY AUTO\_INCREMENT,emp\_name VARCHAR(50) NOT NULL,dept\_id INT,position VARCHAR(50),salary INT,hire\_date DATE,

email VARCHAR(100),

phone VARCHAR(15));

INSERT INTO Employees (emp\_name, dept\_id, position, salary, hire\_date, email, phone) VALUES

('Alice Johnson', 1, 'Manager', 80000, '2021-06-15', 'alice.johnson@example.com', '555-1001'),

('Bob Smith', 2, 'Developer', 75000, '2020-03-22', 'bob.smith@example.com', '555-1002'),

('Catherine Davis', 3, 'Accountant', 70000, '2019-07-30', 'catherine.davis@example.com', '555-1003'),

('David Brown', 4, 'Marketing Specialist', 65000, '2021-01-17', 'david.brown@example.com', '555-1004'),

('Eva Green', 5, 'Sales Representative', 60000, '2020-09-10', 'eva.green@example.com', '555-1005'),

('Frank White', 1, 'HR Specialist', 62000, '2018-11-05', 'frank.white@example.com', '555-1006'),

('Grace Lee', 2, 'Senior Developer', 90000, '2019-02-14', 'grace.lee@example.com', '555-1007'),

('Henry Adams', 3, 'Financial Analyst', 72000, '2021-04-23', 'henry.adams@example.com', '555-1008'),

('Isabella Moore', 4, 'Content Writer', 58000, '2019-10-19', 'isabella.moore@example.com', '555-1009'),

('Jack Wilson', 5, 'Sales Manager', 82000, '2020-07-08', 'jack.wilson@example.com', '555-1010'),

('Karen Taylor', 1, 'HR Manager', 85000, '2017-12-12', 'karen.taylor@example.com', '555-1011'),

('Liam Martin', 2, 'Lead Developer', 92000, '2021-05-03', 'liam.martin@example.com', '555-1012'),

('Mia Thomas', 3, 'Accounts Manager', 78000, '2018-06-09', 'mia.thomas@example.com', '555-1013'),

('Noah Harris', 4, 'SEO Specialist', 66000, '2021-03-21', 'noah.harris@example.com', '555-1014'),

('Olivia Clark', 5, 'Sales Executive', 62000, '2019-11-13', 'olivia.clark@example.com', '555-1015'),

('Paul Lewis', 1, 'HR Assistant', 58000, '2020-01-25', 'paul.lewis@example.com', '555-1016'),

('Quinn Walker', 2, 'Junior Developer', 68000, '2019-08-28', 'quinn.walker@example.com', '555-1017'),

('Rachel King', 3, 'Accountant', 71000, '2018-04-06', 'rachel.king@example.com', '555-1018'),

('Samuel Scott', 4, 'Marketing Manager', 85000, '2017-10-03', 'samuel.scott@example.com', '555-1019'),

('Tara Allen', 5, 'Sales Coordinator', 61000, '2020-12-09', 'tara.allen@example.com', '555-1020'),

('Uma Mitchell', 1, 'HR Consultant', 73000, '2019-09-16', 'uma.mitchell@example.com', '555-1021'),

('Victor Evans', 2, 'DevOps Engineer', 89000, '2021-07-04', 'victor.evans@example.com', '555-1022'),

('Wendy Baker', 3, 'Financial Controller', 85000, '2018-05-29', 'wendy.baker@example.com', '555-1023'),

('Xander Phillips', 4, 'Digital Marketing Specialist', 64000, '2020-08-02', 'xander.phillips@example.com', '555-1024'),

('Yvonne Turner', 5, 'Sales Analyst', 69000, '2019-02-18', 'yvonne.turner@example.com', '555-1025'),

('Zachary Brooks', 1, 'HR Generalist', 72000, '2021-06-28', 'zachary.brooks@example.com', '555-1026'),

('Amy Foster', 2, 'Web Developer', 74000, '2019-12-22', 'amy.foster@example.com', '555-1027'),

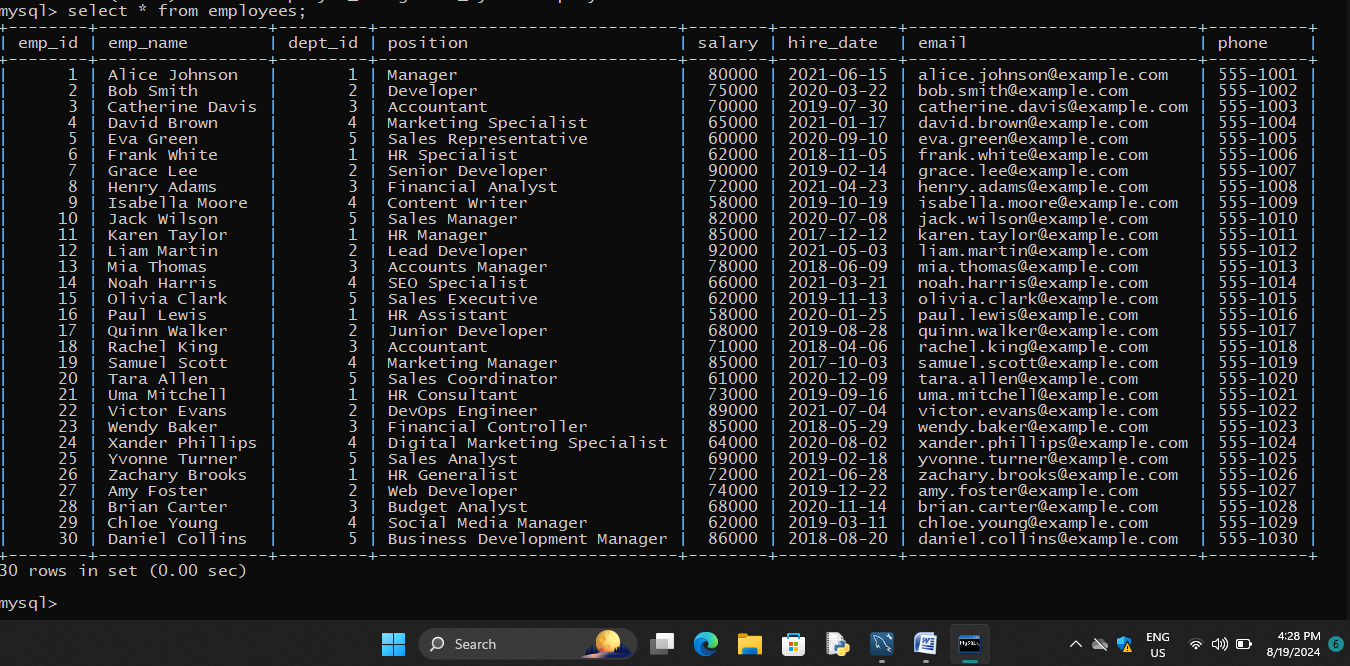
('Brian Carter', 3, 'Budget Analyst', 68000, '2020-11-14', 'brian.carter@example.com', '555-1028'),

('Chloe Young', 4, 'Social Media Manager', 62000, '2019-03-11', 'chloe.young@example.com', '555-1029'),

('Daniel Collins', 5, 'Business Development Manager', 86000, '2018-08-20', 'daniel.collins@example.com', '555-1030');

SELECT \* FROM Employee;

OUTPUT:



INSERT INTO Employees (emp\_name, dept\_id, position, salary, hire\_date, email, phone)

SELECT

CONCAT('Employee\_', n),

FLOOR(1 + (RAND() \* 100)), -- Random dept\_id between 1 and 100

CASE

WHEN n % 2 = 0 THEN 'Manager'

WHEN n % 3 = 0 THEN 'Developer'

ELSE 'Analyst'

END,

FLOOR(30000 + (RAND() \* 50000)), -- Random salary between 30,000 and 80,000

DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 3650) DAY), -- Random hire\_date in the last 10 years

CONCAT('employee\_', n, '@company.com'),

CONCAT('555-', LPAD(FLOOR(RAND() \* 10000), 4, '0'))

FROM

(SELECT n FROM (SELECT @row := @row + 1 AS n FROM (SELECT 0 UNION ALL SELECT 1) t1 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t2 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t3 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t4 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t5 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t6, (SELECT @row := 0) r) num LIMIT 1000) as nums;

```

- Explanation:

- Adds 1000 employees with random department assignments, positions, salaries, and contact details.

3. **Insert Data into Salaries Table**:

-- Inserting 1000 salary records with random data

CREATE TABLE Salaries (salary\_id INT PRIMARY KEY AUTO\_INCREMENT,emp\_id INT,base\_salary INT,bonus INT,total\_salary int AS (base\_salary + bonus),

pay\_date DATE);

INSERT INTO Salaries (emp\_id, base\_salary, bonus, pay\_date) VALUES

(1, 80000, 5000, '2024-01-31'),

(2, 75000, 4500, '2024-01-31'),

(3, 70000, 4000, '2024-01-31'),

(4, 65000, 3500, '2024-01-31'),

(5, 60000, 3000, '2024-01-31'),

(6, 62000, 3100, '2024-01-31'),

(7, 90000, 5500, '2024-01-31'),

(8, 72000, 3600, '2024-01-31'),

(9, 58000, 2900, '2024-01-31'),

(10, 82000, 4100, '2024-01-31'),

(11, 85000, 4250, '2024-01-31'),

(12, 92000, 4600, '2024-01-31'),

(13, 78000, 3900, '2024-01-31'),

(14, 66000, 3300, '2024-01-31'),

(15, 62000, 3100, '2024-01-31'),

(16, 58000, 2900, '2024-01-31'),

(17, 68000, 3400, '2024-01-31'),

(18, 71000, 3550, '2024-01-31'),

(19, 85000, 4250, '2024-01-31'),

(20, 61000, 3050, '2024-01-31'),

(21, 73000, 3650, '2024-01-31'),

(22, 89000, 4450, '2024-01-31'),

(23, 85000, 4250, '2024-01-31'),

(24, 64000, 3200, '2024-01-31'),

(25, 69000, 3450, '2024-01-31'),

(26, 72000, 3600, '2024-01-31'),

(27, 74000, 3700, '2024-01-31'),

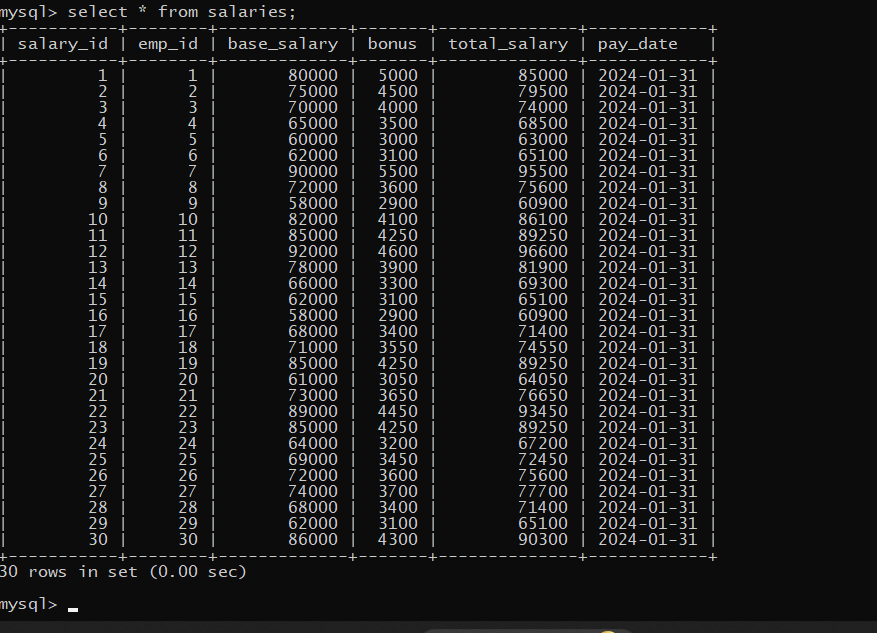
(28, 68000, 3400, '2024-01-31'),

(29, 62000, 3100, '2024-01-31'),

(30, 86000, 4300, '2024-01-31');

SELECT \* FROM Salaries;

OUTPUT:



INSERT INTO Salaries (emp\_id, base\_salary, bonus, pay\_date)

SELECT

emp\_id,

salary,

FLOOR(RAND() \* 10000), -- Random bonus up to 10,000

DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 365) DAY) -- Random pay\_date in the last year

FROM

Employees;

```

- Explanation:

- Adds 1000 salary records with random bonuses and payment dates.

**4. Insert Data into Projects Table:**

-- Inserting 200 projects with random data

INSERT INTO Projects (project\_name, start\_date, end\_date, budget)

VALUES

('Project Alpha', '2024-02-01', '2024-12-31', 150000),

('Project Beta', '2024-03-01', '2024-11-30', 100000),

('Project Alpha', '2023-01-01', '2023-06-01', 50000),

('Project Beta', '2023-02-01', '2023-07-01', 60000),

('Project Gamma', '2023-03-01', '2023-08-01', 45000),

('Project Delta', '2023-04-01', '2023-09-01', 55000),

('Project Epsilon', '2023-05-01', '2023-10-01', 70000),

('Project Zeta', '2023-06-01', '2023-11-01', 80000),

('Project Eta', '2023-07-01', '2023-12-01', 90000),

('Project Theta', '2023-08-01', '2024-01-01', 100000),

('Project Iota', '2023-09-01', '2024-02-01', 75000),

('Project Kappa', '2023-10-01', '2024-03-01', 85000),

('Project Lambda', '2023-11-01', '2024-04-01', 95000),

('Project Mu', '2023-12-01', '2024-05-01', 65000),

('Project Nu', '2024-01-01', '2024-06-01', 55000),

('Project Xi', '2024-02-01', '2024-07-01', 45000),

('Project Omicron', '2024-03-01', '2024-08-01', 70000),

('Project Pi', '2024-04-01', '2024-09-01', 60000),

('Project Rho', '2024-05-01', '2024-10-01', 50000),

('Project Sigma', '2024-06-01', '2024-11-01', 75000),

('Project Tau', '2024-07-01', '2024-12-01', 85000),

('Project Upsilon', '2024-08-01', '2025-01-01', 95000),

('Project Phi', '2024-09-01', '2025-02-01', 100000),

('Project Chi', '2024-10-01', '2025-03-01', 65000),

('Project Psi', '2024-11-01', '2025-04-01', 55000),

('Project Omega', '2024-12-01', '2025-05-01', 45000),

('Project Alpha II', '2023-01-15', '2023-07-15', 70000),

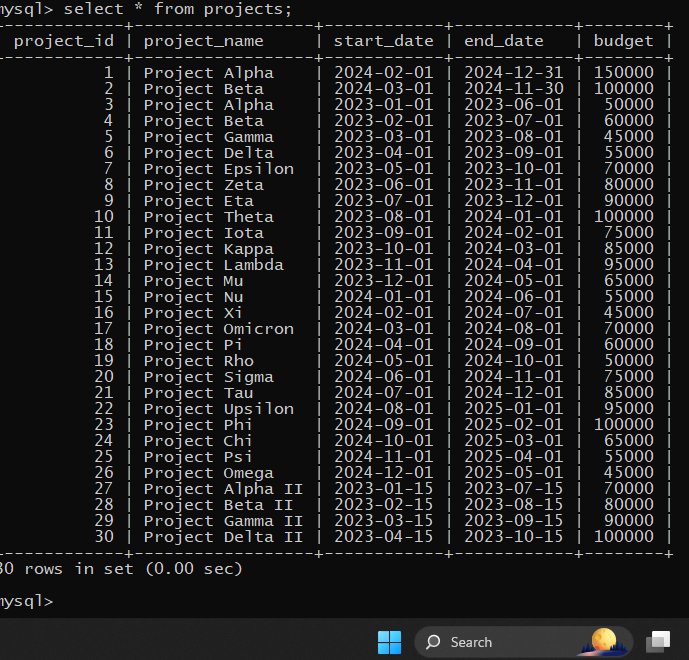
('Project Beta II', '2023-02-15', '2023-08-15', 80000),

('Project Gamma II', '2023-03-15', '2023-09-15', 90000),

('Project Delta II', '2023-04-15', '2023-10-15', 100000);

SELECT \* FROM Projects;

OUTPUT:



INSERT INTO Projects (project\_name, start\_date, end\_date, budget)

SELECT

CONCAT('Project\_', n),

DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 365 \* 2) DAY), -- Random start\_date in the last 2 years

DATE\_ADD(CURDATE(), INTERVAL FLOOR(RAND() \* 365) DAY), -- Random end\_date within the next year

FLOOR(50000 + (RAND() \* 150000)) -- Random budget between 50,000 and 200,000

FROM

(SELECT n FROM (SELECT @row := @row + 1 AS n FROM (SELECT 0 UNION ALL SELECT 1) t1 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t2 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t3 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t4 CROSS JOIN (SELECT 0 UNION ALL SELECT

1) t5 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t6, (SELECT @row := 0) r) num LIMIT 200) as nums;

```

- Explanation:

- Adds 200 projects with random start and end dates, and budgets.

**5. Insert Data into EmployeeProjects Table:**

-- Linking employees to projects, inserting 3000 records with random data

INSERT INTO EmployeeProjects (emp\_id, project\_id, role)

VALUES

(1, 1, 'Developer'),

(2, 1, 'Tester'),

(3, 2, 'Project Manager'),

(4, 2, 'Developer'),

(5, 3, 'Tester'),

(6, 3, 'Developer'),

(7, 4, 'Project Manager'),

(8, 4, 'Tester'),

(9, 5, 'Developer'),

(10, 5, 'Tester'),

(11, 6, 'Developer'),

(12, 6, 'Project Manager'),

(13, 7, 'Tester'),

(14, 7, 'Developer'),

(15, 8, 'Project Manager'),

(16, 8, 'Tester'),

(17, 9, 'Developer'),

(18, 9, 'Tester'),

(19, 10, 'Developer'),

(20, 10, 'Project Manager'),

(21, 11, 'Tester'),

(22, 11, 'Developer'),

(23, 12, 'Project Manager'),

(24, 12, 'Tester'),

(25, 13, 'Developer'),

(26, 13, 'Tester'),

(27, 14, 'Developer'),

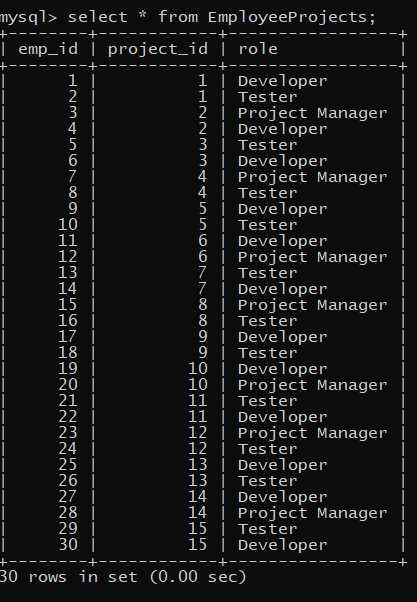
(28, 14, 'Project Manager'),

(29, 15, 'Tester'),

(30, 15, 'Developer');

SELECT \* FROM Employee to Projects;

OUTPUT:



INSERT INTO EmployeeProjects (emp\_id, project\_id, role)

SELECT

emp\_id,

FLOOR(1 + (RAND() \* 200)), -- Random project\_id between 1 and 200

CASE

WHEN n % 2 = 0 THEN 'Team Member'

WHEN n % 3 = 0 THEN 'Lead'

ELSE 'Consultant'

END

FROM

Employees

CROSS JOIN (SELECT @row := @row + 1 AS n FROM (SELECT 0 UNION ALL SELECT 1) t1 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t2 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t3 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t4 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t5 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t6, (SELECT @row := 0) r) as nums LIMIT 3000;

```

- Explanation:

- Adds 3000 records linking employees to projects with roles.

**6. Insert Data into PerformanceReviews Table:**

-- Inserting 1000 performance reviews with random data

INSERT INTO PerformanceReviews (emp\_id, review\_date, review\_score, comments) VALUES

(1, '2023-01-15', 85, 'Consistently meets expectations.'),

(2, '2023-01-20', 90, 'Exceeds expectations in most areas.'),

(3, '2023-02-05', 78, 'Shows potential but needs improvement in certain areas.'),

(4, '2023-02-10', 88, 'Strong performance overall.'),

(5, '2023-02-25', 92, 'Outstanding performer.'),

(6, '2023-03-01', 80, 'Solid performance with room for growth.'),

(7, '2023-03-15', 75, 'Needs improvement in time management.'),

(8, '2023-03-20', 89, 'Great team player.'),

(9, '2023-04-05', 84, 'Reliable and consistent work.'),

(10, '2023-04-10', 91, 'Shows leadership qualities.'),

(11, '2023-04-25', 86, 'Good communication skills.'),

(12, '2023-05-01', 82, 'Meets expectations but can be more proactive.'),

(13, '2023-05-15', 79, 'Needs to focus more on detail.'),

(14, '2023-05-20', 87, 'Very committed to the job.'),

(15, '2023-06-05', 83, 'Steady performance.'),

(16, '2023-06-10', 90, 'Always willing to help others.'),

(17, '2023-06-25', 85, 'Dependable and trustworthy.'),

(18, '2023-07-01', 88, 'Positive attitude and approach.'),

(19, '2023-07-15', 81, 'Delivers on promises.'),

(20, '2023-07-20', 77, 'Needs to improve on deadlines.'),

(21, '2023-08-05', 92, 'Excellent problem-solving skills.'),

(22, '2023-08-10', 89, 'Highly creative and innovative.'),

(23, '2023-08-25', 76, 'Needs more focus on accuracy.'),

(24, '2023-09-01', 85, 'Good at handling pressure.'),

(25, '2023-09-15', 88, 'Shows great initiative.'),

(26, '2023-09-20', 82, 'Works well under minimal supervision.'),

(27, '2023-10-05', 90, 'Always meets deadlines.'),

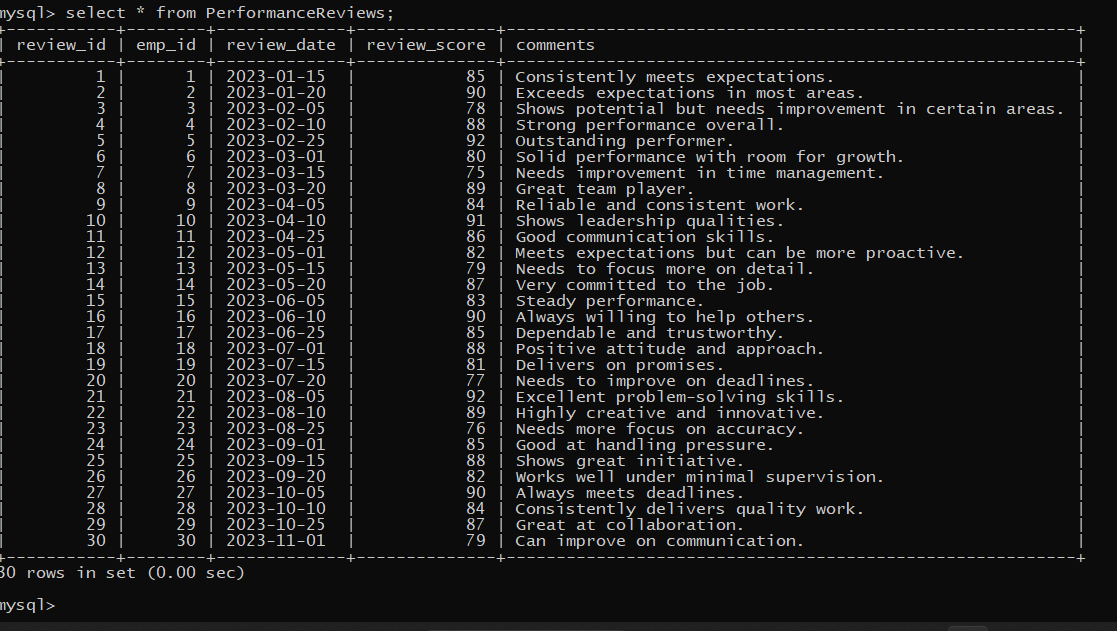
(28, '2023-10-10', 84, 'Consistently delivers quality work.'),

(29, '2023-10-25', 87, 'Great at collaboration.'),

(30, '2023-11-01', 79, 'Can improve on communication.');

SELECT \* FROM PerformanceReviews;

OUTPUT:



INSERT INTO PerformanceReviews (emp\_id, review\_date, review\_score, comments)

SELECT

emp\_id,

DATE\_SUB(CURDATE(), INTERVAL FLOOR(RAND() \* 365) DAY), -- Random review\_date in the last year

FLOOR(1 + (RAND() \* 5)), -- Random review\_score between 1 and 5

CASE

WHEN n % 2 = 0 THEN 'Excellent performance'

WHEN n % 3 = 0 THEN 'Needs improvement'

ELSE 'Satisfactory performance'

END

FROM

Employees

CROSS JOIN (SELECT @row := @row + 1 AS n FROM (SELECT 0 UNION ALL SELECT 1) t1 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t2 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t3 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t4 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t5 CROSS JOIN (SELECT 0 UNION ALL SELECT 1) t6, (SELECT @row := 0) r) as nums LIMIT 1000;

```

- Explanation:

- Adds 1000 performance reviews with random scores and comments.

---

Step 4: Complex Queries and Operations

After setting up and populating the tables with large datasets, students can perform the following complex operations:

# 1. Complex JOINs to Retrieve Data

```sql

-- Retrieve all employees along with their department names and project roles

SELECT

Employees.emp\_name,

Departments.dept\_name,

EmployeeProjects.role,

Projects.project\_name

FROM

Employees

JOIN

Departments ON Employees.dept\_id = Departments.dept\_id

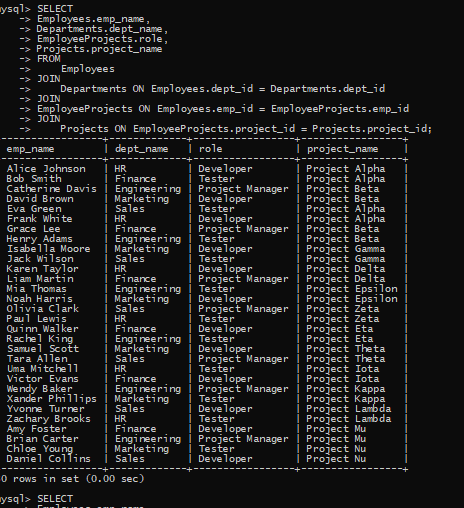
JOIN

EmployeeProjects ON Employees.emp\_id = EmployeeProjects.emp\_id

JOIN

Projects ON EmployeeProjects.project\_id = Projects.project\_id;

OUTPUT:



- Explanation:

- This query joins multiple tables to retrieve comprehensive employee information, including their department and project details.

# 2. Subquery Example to Find the Highest Salary in Each Department

```sql

-- **Find the highest salary in each department**

SELECT

dept\_name,

MAX(salary) AS highest\_salary

FROM

Employees

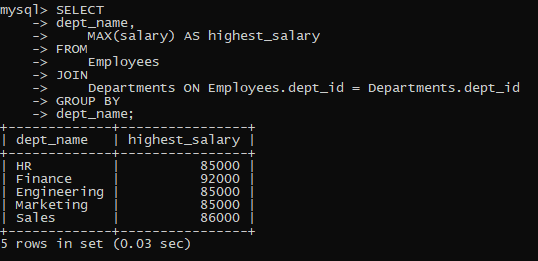
JOIN

Departments ON Employees.dept\_id = Departments.dept\_id

GROUP BY

dept\_name;

OUTPUT:



```

- Explanation:

- This query uses `GROUP BY` and `MAX` to find the highest salary for each department.

# 3. Transaction Example - Salary Update with Rollback

```sql

START TRANSACTION;

UPDATE Salaries

SET base\_salary = base\_salary \* 1.1

WHERE emp\_id = 1;

ROLLBACK;

```

- Explanation:

- This demonstrates a transaction where a salary update is performed and then rolled back, undoing the changes.