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

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:

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:

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:

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:

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:

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:

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

INSERT INTO Departments VALUES

(1, 'Cloud Services', 'Delhi'),

(2, 'Network Administration', 'Delhi'),

(3, 'Software Development', 'Bangalore'),

(4, 'Database Administration', 'Delhi'),

(5, 'System Analysis', 'Mumbai'),

(6, 'Technical Support', 'Bangalore'),

(7, 'Cybersecurity', 'Hyderabad'),

(8, 'IT Project Management', 'Delhi'),

(9, 'Quality Assurance', 'Chennai'),

(10, 'Business Intelligence', 'Delhi'),

(11, 'Dev', 'Pune'),

(12, 'Data Science', 'Delhi'),

(13, 'Software Development', 'Kolkata'),

(14, 'Cloud Services', 'Delhi'),

(15, 'IT Consultancy', 'Noida'),

(16, 'Data Science', 'Delhi'),

(17, 'Software Engineering', 'chennai'),

(18, 'IT Operations', 'Delhi'),

(19, 'IT Training', 'Chennai'),

(20, 'Technical Support', 'Hyderabad'),

(21, 'System Administration', 'Delhi'),

(22, 'Information Security', 'Mumbai'),

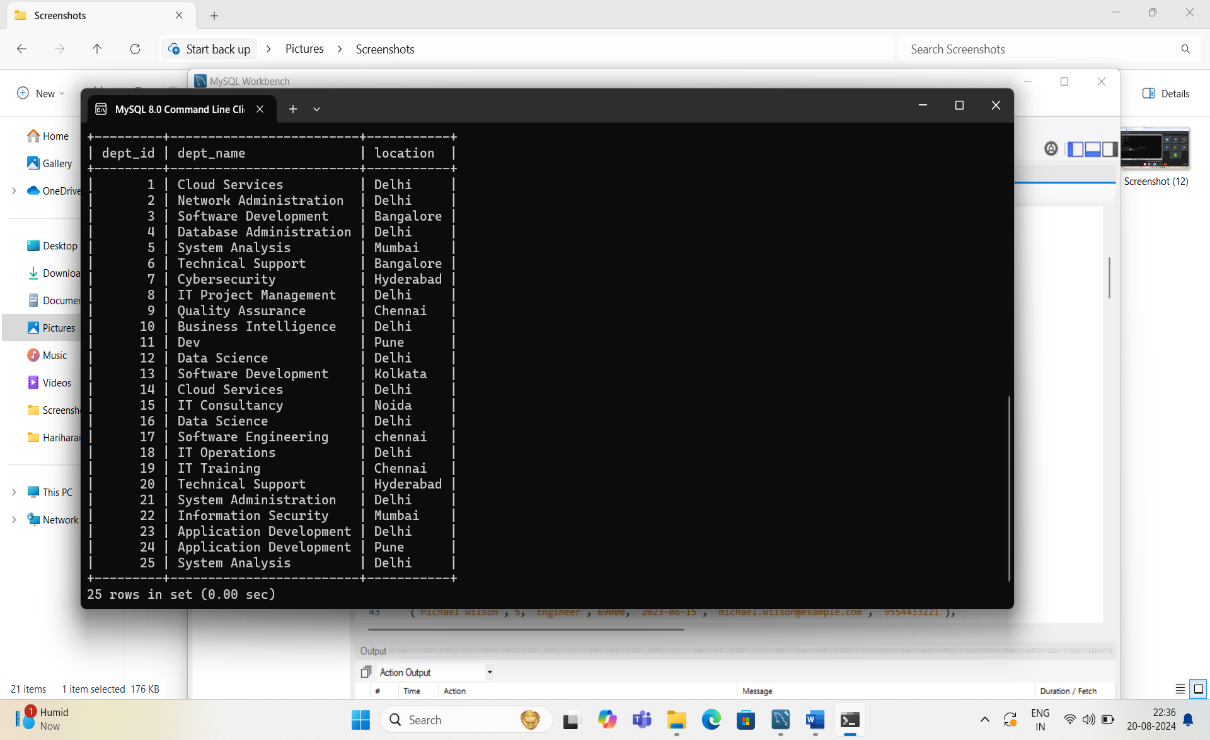
(23, 'Application Development', 'Delhi'),

(24, 'Application Development', 'Pune'),

(25, 'System Analysis', 'Delhi');

select \* from Departments ;

**OUTPUT:**

****

-- Insert additional departments up to 20 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 25 departments with unique names and locations.

2. Insert Data into Employees Table:

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

('John Doe', 1, 'Developer', 60000, '2023-01-15', 'john.doe@example.com', '9876543210'),

('Jane Smith', 2, 'Designer', 62000, '2023-02-20', 'jane.smith@example.com', '9887654321'),

('Alice Johnson', 1, 'Manager', 75000, '2022-03-25', 'alice.johnson@example.com', '9998887777'),

('Bob Brown', 3, 'Analyst', 68000, '2023-04-30', 'bob.brown@example.com', '9776655443'),

('Emily Davis', 4, 'HR Specialist', 72000, '2022-05-10', 'emily.davis@example.com', '9665544332'),

('Michael Wilson', 5, 'Engineer', 69000, '2023-06-15', 'michael.wilson@example.com', '9554433221'),

('Sarah Miller', 1, 'Developer', 61000, '2023-07-20', 'sarah.miller@example.com', '9443322110'),

('David Taylor', 2, 'Marketing Manager', 73000, '2022-08-25', 'david.taylor@example.com', '9332211009'),

('Laura Anderson', 3, 'Financial Analyst', 74000, '2023-09-30', 'laura.anderson@example.com', '9221100998'),

('James Thomas', 4, 'Recruiter', 66000, '2023-10-15', 'james.thomas@example.com', '9110998877'),

('Linda Harris', 5, 'Project Manager', 78000, '2022-11-20', 'linda.harris@example.com', '9008877665'),

('Robert Martinez', 1, 'Senior Developer', 80000, '2023-12-01', 'robert.martinez@example.com', '9888776655'),

('Karen Robinson', 2, 'Sales Coordinator', 65000, '2023-01-10', 'karen.robinson@example.com', '9777888999'),

('Steven Walker', 3, 'Accountant', 67000, '2023-02-15', 'steven.walker@example.com', '9666778899'),

('Jessica Young', 4, 'Training Specialist', 70000, '2022-03-20', 'jessica.young@example.com', '9555667788'),

('Daniel King', 5, 'IT Support', 64000, '2023-04-25', 'daniel.king@example.com', '9444556677'),

('Michelle Scott', 1, 'System Analyst', 72000, '2022-05-30', 'michelle.scott@example.com', '9333445566'),

('Paul Adams', 2, 'Product Manager', 76000, '2023-06-10', 'paul.adams@example.com', '9222334455'),

('Nancy Nelson', 3, 'Finance Manager', 78000, '2022-07-15', 'nancy.nelson@example.com', '9111223344'),

('George Carter', 4, 'Employee Relations', 69000, '2023-08-20', 'george.carter@example.com', '9000112233'),

('Helen Mitchell', 5, 'Business Analyst', 71000, '2022-09-25', 'helen.mitchell@example.com', '9888991122'),

('Gary Roberts', 1, 'Database Administrator', 73000, '2023-10-30', 'gary.roberts@example.com', '9777002233'),

('Deborah Evans', 2, 'Content Writer', 62000, '2023-11-10', 'deborah.evans@example.com', '9666113344'),

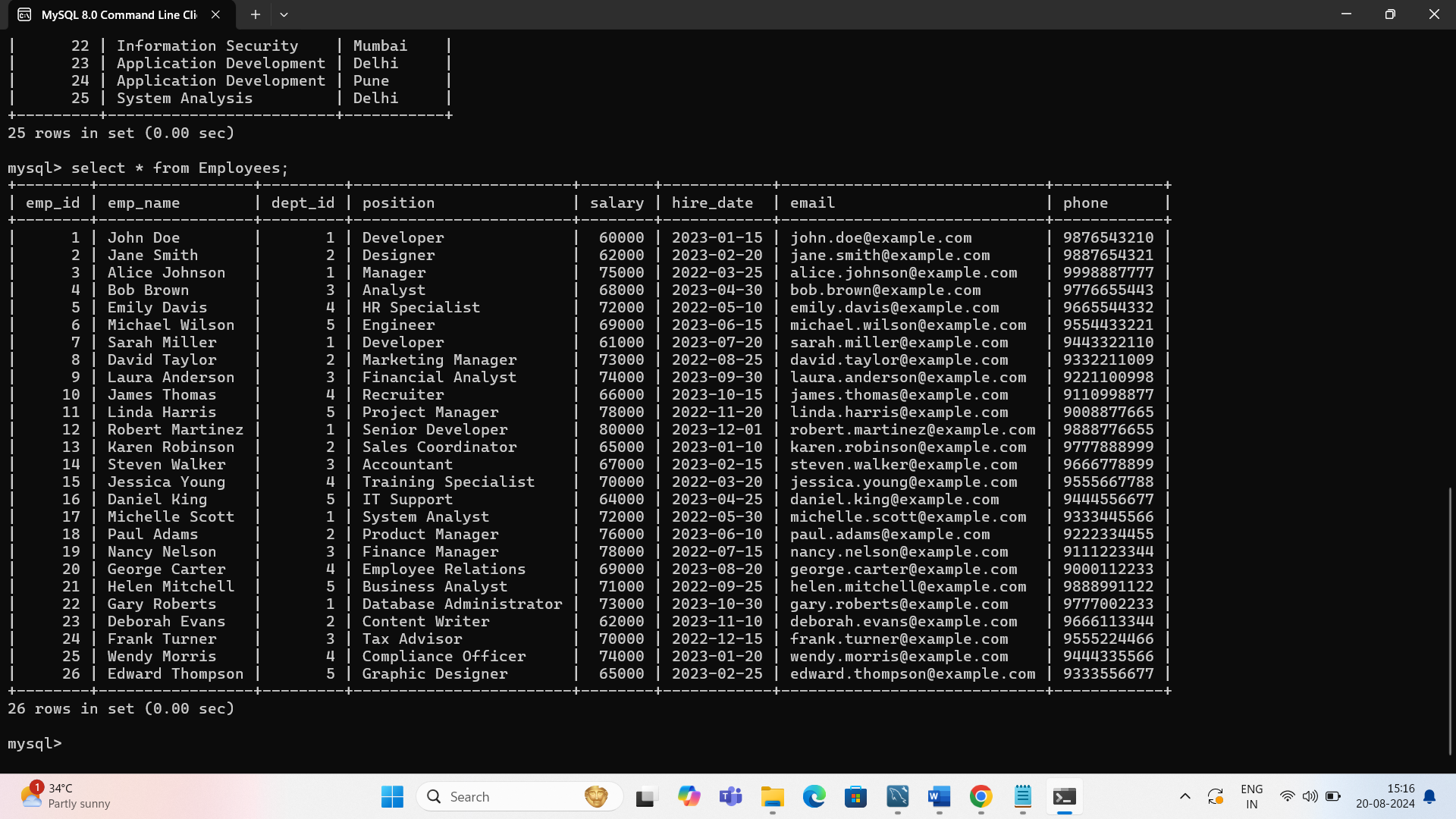
('Frank Turner', 3, 'Tax Advisor', 70000, '2022-12-15', 'frank.turner@example.com', '9555224466'),

('Wendy Morris', 4, 'Compliance Officer', 74000, '2023-01-20', 'wendy.morris@example.com', '9444335566'),

('Edward Thompson', 5, 'Graphic Designer', 65000, '2023-02-25', 'edward.thompson@example.com', '9333556677');

select \* from Employees;

**OUTPUT:**

****

-- Inserting 25 employees with random data

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 25 employees with random department assignments, positions, salaries, and contact details.

3. Insert Data into Salaries Table:

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

(1, 60000, 5000, '2024-01-15'),

(2, 62000, 4500, '2024-02-15'),

(3, 75000, 7000, '2024-03-15'),

(4, 68000, 5500, '2024-04-15'),

(5, 72000, 6000, '2024-05-15'),

(6, 69000, 6500, '2024-06-15'),

(7, 61000, 5200, '2024-07-15'),

(8, 73000, 5800, '2024-08-15'),

(9, 74000, 6000, '2024-09-15'),

(10, 66000, 5300, '2024-10-15'),

(11, 78000, 7000, '2024-11-15'),

(12, 80000, 7500, '2024-12-15'),

(13, 65000, 4800, '2024-01-30'),

(14, 62000, 4700, '2024-02-28'),

(15, 70000, 5300, '2024-03-31'),

(16, 68000, 5600, '2024-04-30'),

(17, 69000, 5800, '2024-05-31'),

(18, 71000, 5900, '2024-06-30'),

(19, 73000, 6000, '2024-07-31'),

(20, 74000, 6200, '2024-08-31'),

(21, 75000, 6400, '2024-09-30'),

(22, 76000, 6600, '2024-10-31'),

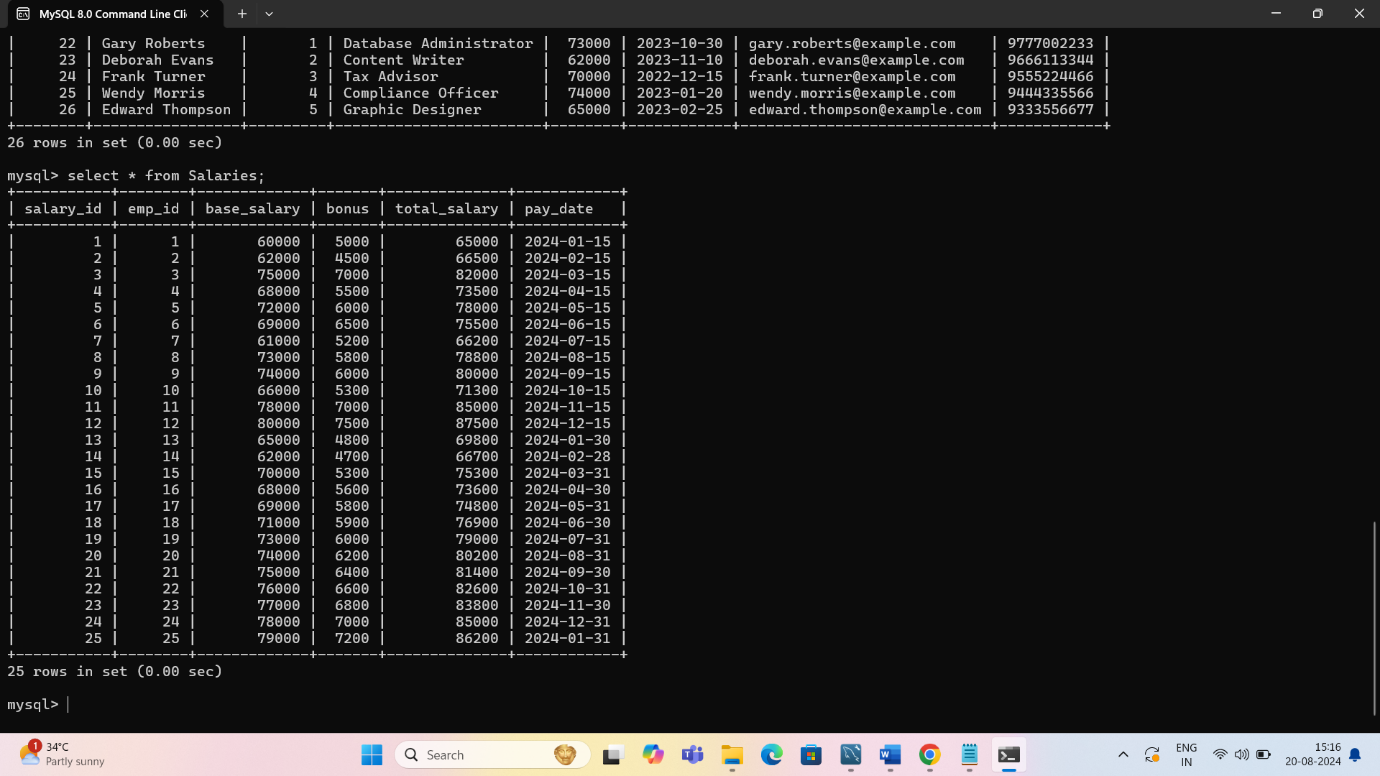
(23, 77000, 6800, '2024-11-30'),

(24, 78000, 7000, '2024-12-31'),

(25, 79000, 7200, '2024-01-31');

select \* from Salaries ;

**OUTPUT:**

****

-- Inserting 1000 salary records with random data

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 25 salary records with random bonuses and payment dates.

4. Insert Data into Projects Table:

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

('New Website Launch', '2024-01-15', '2024-06-15', 100000),

('Mobile App Development', '2024-02-01', '2024-07-01', 150000),

('Customer Portal Upgrade', '2024-03-01', '2024-08-01', 200000),

('CRM Integration', '2024-04-01', '2024-09-01', 120000),

('Cloud Migration', '2024-05-01', '2024-10-01', 180000),

('Data Analytics Platform', '2024-06-01', '2024-11-01', 160000),

('HR System Overhaul', '2024-07-01', '2024-12-01', 140000),

('E-commerce Platform', '2024-08-01', '2025-01-01', 190000),

('AI Chatbot Implementation', '2024-09-01', '2025-02-01', 210000),

('Sales Dashboard Redesign', '2024-10-01', '2025-03-01', 220000),

('ERP System Upgrade', '2024-11-01', '2025-04-01', 230000),

('Cybersecurity Enhancement', '2024-12-01', '2025-05-01', 250000),

('Infrastructure Modernization', '2025-01-01', '2025-06-01', 260000),

('Business Intelligence Tool', '2025-02-01', '2025-07-01', 270000),

('Marketing Automation System', '2025-03-01', '2025-08-01', 280000),

('Project Management Suite', '2025-04-01', '2025-09-01', 290000),

('Finance System Integration', '2025-05-01', '2025-10-01', 300000),

('Supply Chain Optimization', '2025-06-01', '2025-11-01', 310000),

('Customer Feedback System', '2025-07-01', '2025-12-01', 320000),

('Product Launch Strategy', '2025-08-01', '2026-01-01', 330000),

('Employee Training Program', '2025-09-01', '2026-02-01', 340000),

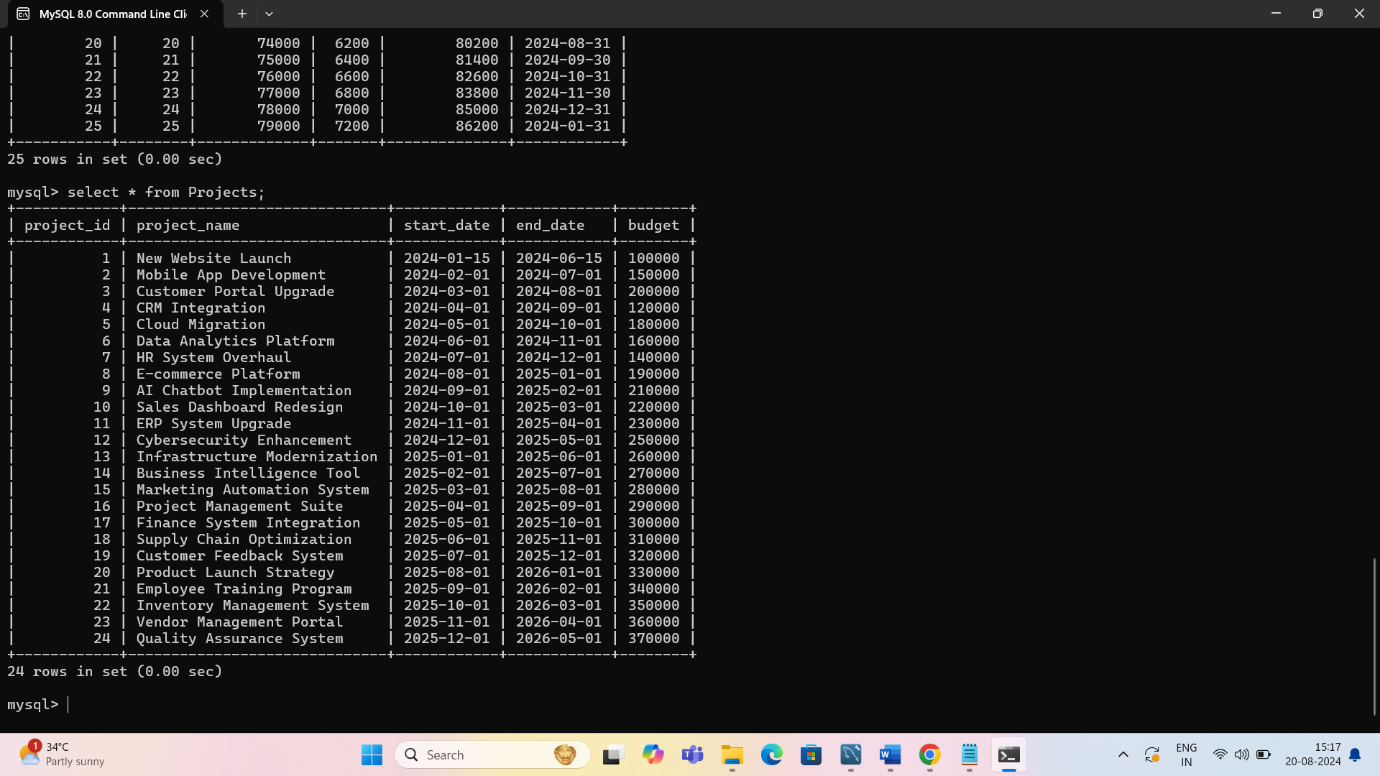
('Inventory Management System', '2025-10-01', '2026-03-01', 350000),

('Vendor Management Portal', '2025-11-01', '2026-04-01', 360000),

('Quality Assurance System', '2025-12-01', '2026-05-01', 370000);

select \* from Projects;

**OUTPUT:**

****

-- Inserting 25 projects with random data

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:

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

(1, 1, 'Project Manager'),

(1, 2, 'Lead Developer'),

(2, 1, 'Developer'),

(2, 3, 'Developer'),

(3, 2, 'Business Analyst'),

(3, 4, 'Consultant'),

(4, 3, 'Project Manager'),

(4, 5, 'Tester'),

(5, 4, 'Developer'),

(5, 6, 'Lead Developer'),

(6, 5, 'Analyst'),

(6, 7, 'Project Manager'),

(7, 6, 'Tester'),

(7, 8, 'Consultant'),

(8, 7, 'Developer'),

(8, 9, 'Project Manager'),

(9, 8, 'Lead Developer'),

(9, 10, 'Tester'),

(10, 9, 'Business Analyst'),

(10, 10, 'Consultant'),

(11, 1, 'Project Manager'),

(11, 2, 'Lead Developer'),

(12, 3, 'Developer'),

(12, 4, 'Tester'),

(13, 5, 'Business Analyst'),

(13, 6, 'Consultant'),

(14, 7, 'Project Manager'),

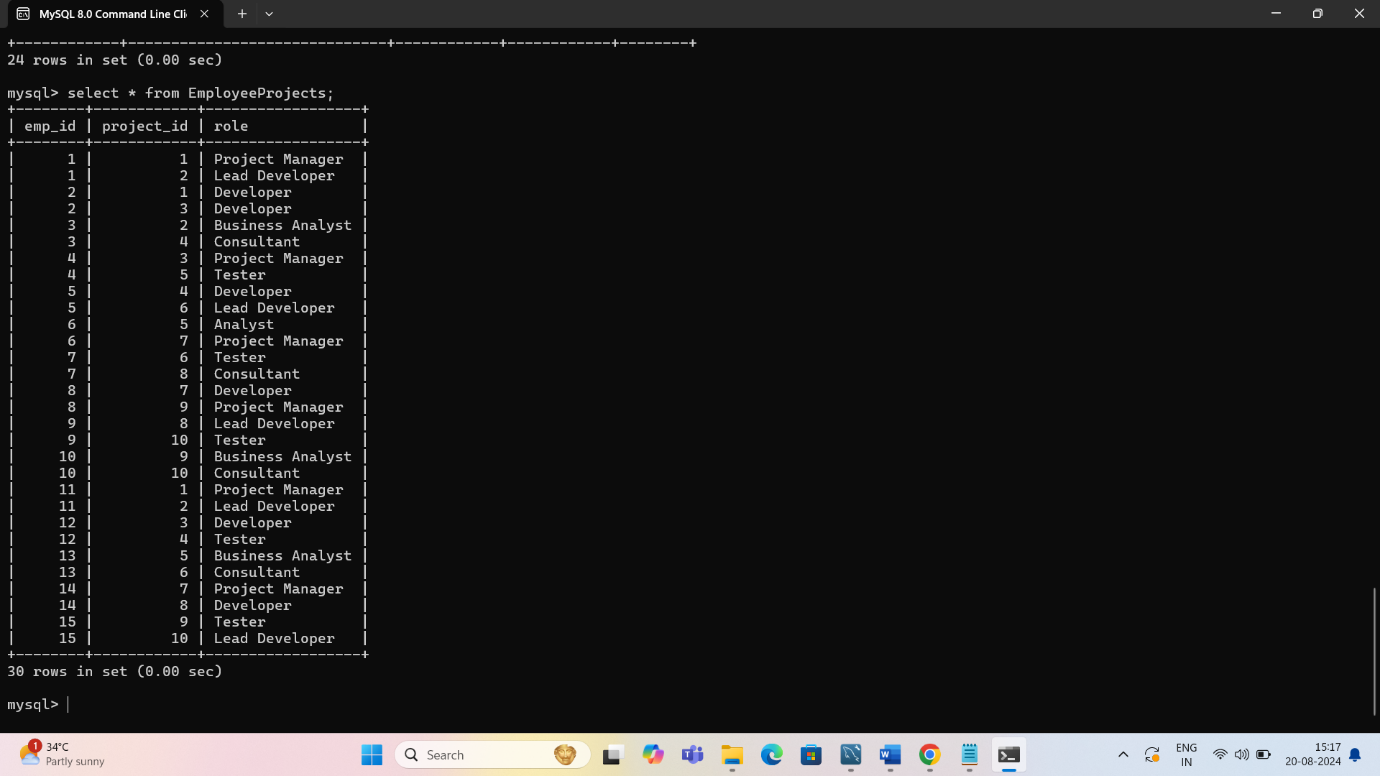
(14, 8, 'Developer'),

(15, 9, 'Tester'),

(15, 10, 'Lead Developer');

select \* from EmployeeProjects;

**OUTPUT:**

****

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

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;

6. Insert Data into PerformanceReviews Table:

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

(1, '2024-01-15', 9, 'Excellent performance throughout the year.'),

(2, '2024-02-20', 7, 'Good performance, but room for improvement in communication.'),

(3, '2024-03-25', 6, 'Satisfactory performance. Needs to meet deadlines more consistently.'),

(4, '2024-04-30', 10, 'Outstanding work, exceeded expectations on all projects.'),

(5, '2024-05-10', 8, 'Solid performance with good teamwork skills.'),

(6, '2024-06-15', 5, 'Needs to work on technical skills and problem-solving.'),

(7, '2024-07-20', 7, 'Good progress, should focus on leadership skills.'),

(8, '2024-08-25', 10, 'Exceptional performance, a key player on the team.'),

(9, '2024-09-30', 3, 'Performance below expectations, needs significant improvement.'),

(10, '2024-10-15', 8, 'Strong performance, but needs to improve punctuality.'),

(11, '2024-11-20', 6, 'Average performance, should focus on meeting goals.'),

(12, '2024-12-25', 9, 'Excellent contribution to major projects.'),

(13, '2025-01-10', 8, 'Good performance, with notable improvements in recent months.'),

(14, '2025-02-15', 6, 'Satisfactory, but requires better time management.'),

(15, '2025-03-20', 9, 'Outstanding work ethic and productivity.'),

(16, '2025-04-25', 8, 'Solid performance, demonstrates strong problem-solving skills.'),

(17, '2025-05-30', 6, 'Needs to enhance skills in project management.'),

(18, '2025-06-15', 7, 'Good overall performance, should improve client interaction.'),

(19, '2025-07-20', 10, 'Exceptional performance and initiative.'),

(20, '2025-08-25', 4, 'Performance is below the expected standard.'),

(21, '2025-09-30', 8, 'Good work, with strong collaboration skills.'),

(22, '2025-10-15', 6, 'Satisfactory performance, needs to work on quality of deliverables.'),

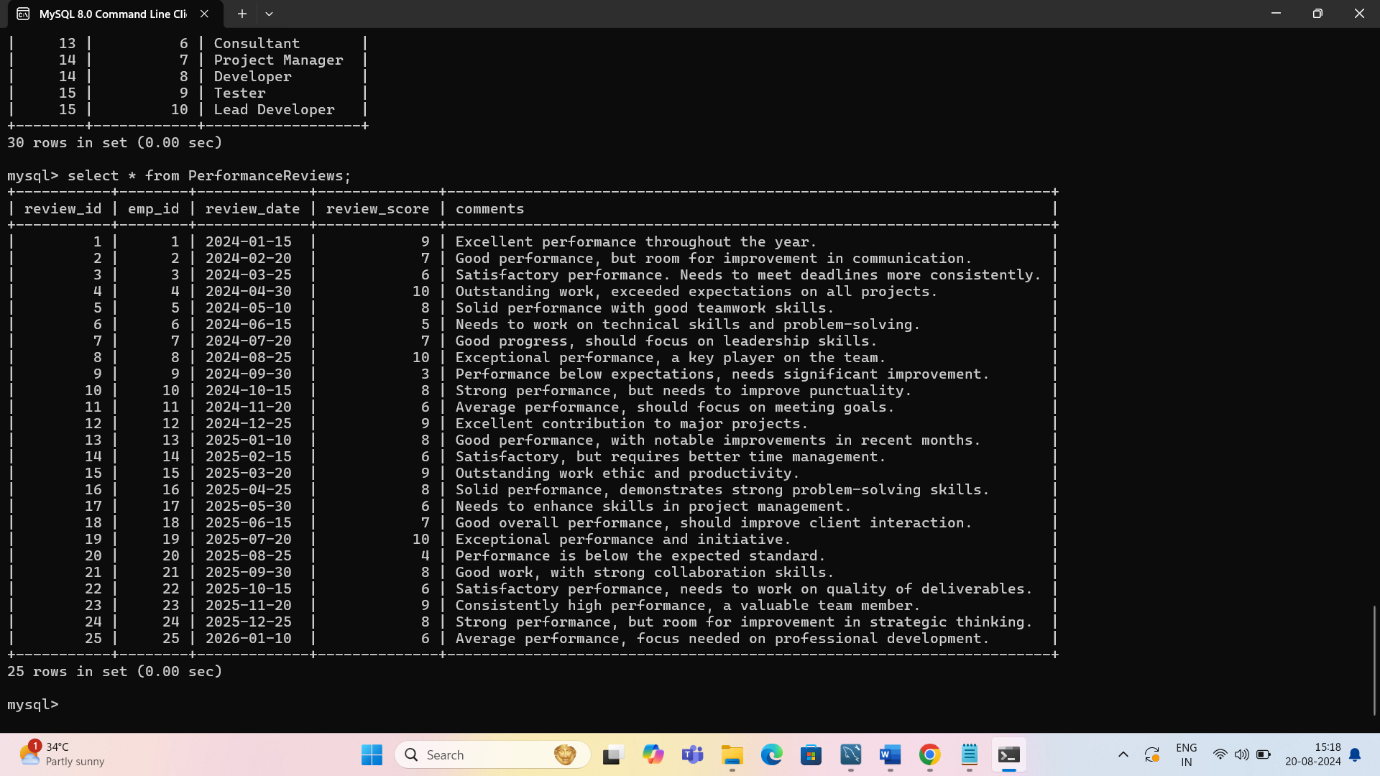
(23, '2025-11-20', 9, 'Consistently high performance, a valuable team member.'),

(24, '2025-12-25', 8, 'Strong performance, but room for improvement in strategic thinking.'),

(25, '2026-01-10', 6, 'Average performance, focus needed on professional development.');

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;

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

* + 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

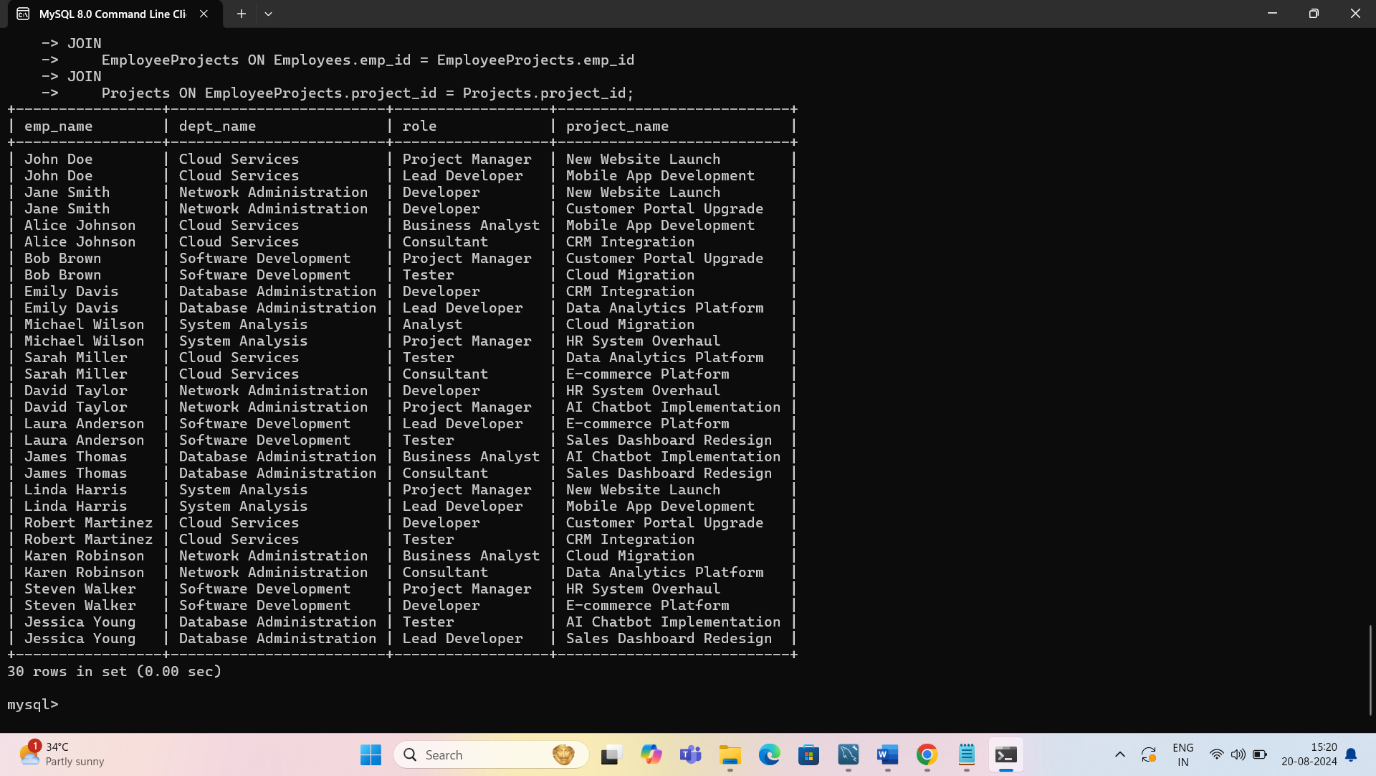
JOIN

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

Find the highest salary in each department

SELECT

dept\_name,

MAX(salary) AS highest\_salary

FROM

Employees

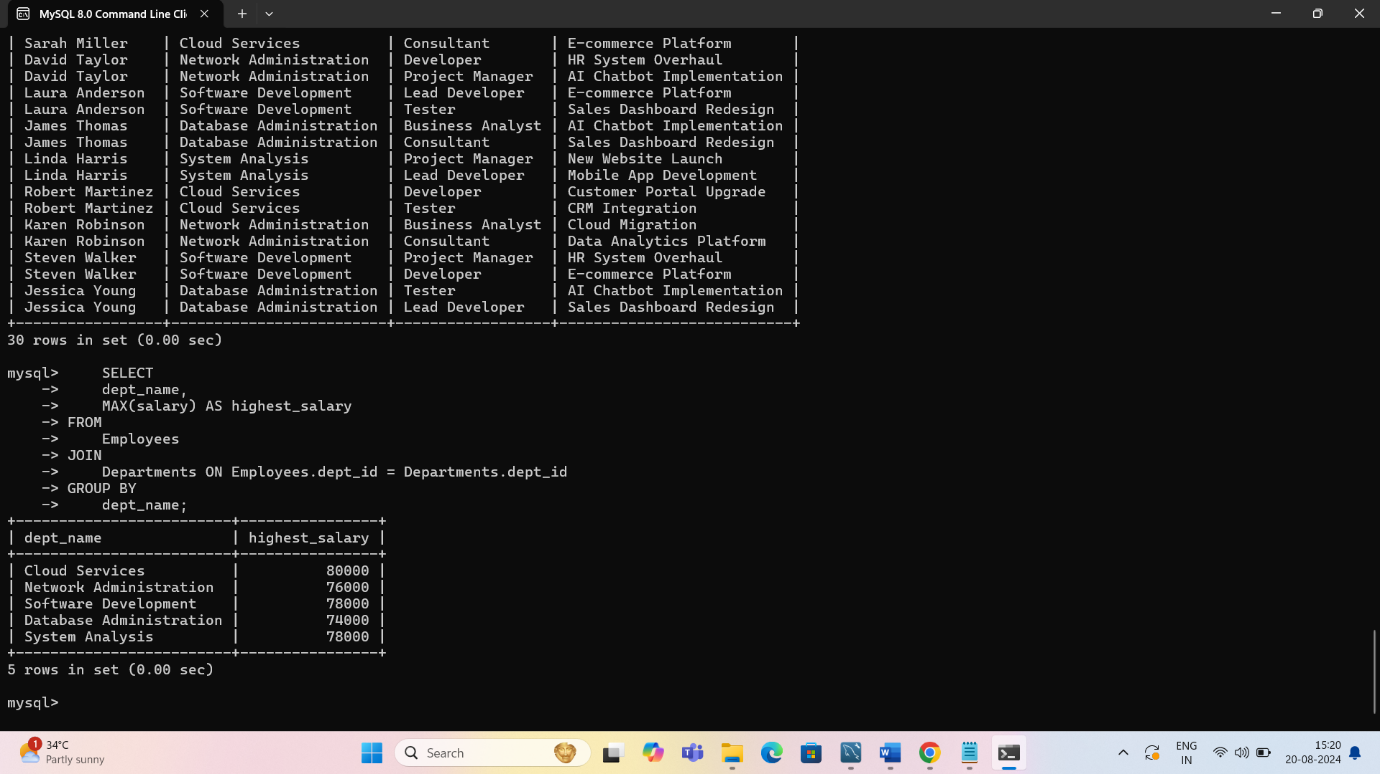
JOIN

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

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.

*Presented by bhavani.M*