**Employee Payroll Management System**

Group 9 (Roll No 41-45)

Roll no 41 – Gourav Gupta

Roll no 42 – Lingudu Devi Prasad

Roll no 43 – Supriya Mohan Goswami

Roll no 44 – Akshit Thapar

Roll no 45 – Saloni Singh

DBMS Lab - Code: CSS453

**National Institute of Technology, Durgapur**

DBMS Lab Report

# Group 9 (Roll No 41-45)

## Problem Statement:

Create a database for an Employee Payroll Management System.

## Description:

The system should manage employee salaries, deductions, and bonuses.

## Tables:

- Employees (EmployeeID, Name, Department, Designation, Salary)

- Payroll (PayrollID, EmployeeID, BasicSalary, Deductions, Bonus, NetSalary)

- Attendance (AttendanceID, EmployeeID, Date, Status)

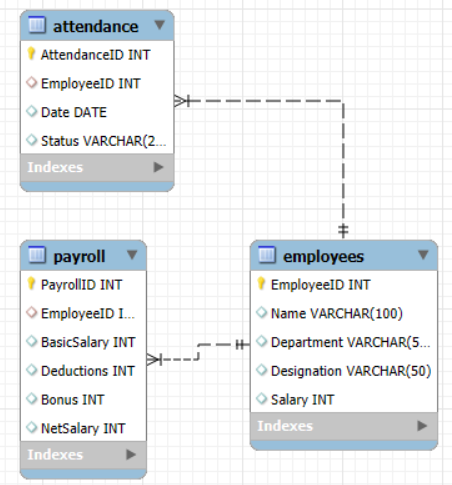
## Queries:

* - Retrieve employee salary details.
* - Calculate net salary after deductions and bonuses.
* - List employees with the highest salary.
* - Find employees with perfect attendance.
* - Update salary details after a promotion.
* - Get employees with the most deductions.
* - Retrieve payroll details for a given month.
* - Identify employees with overtime pay.
* - Delete payroll records older than a year.
* - Find employees who have been with the company the longest.

# *PRINCIPAL:*

* This database is designed to manage the employee payroll system effectively and efficiently. It consists of three main tables: **Employees**, **Payroll**, and **Attendance**.
* **Employees Table**: This stores details about each employee, such as their name, department, designation, and salary. It acts as the core entity that connects other tables.
* **Payroll Table**: This handles salary-related information, including basic salary, deductions, bonuses, and the net salary calculation. It has a foreign key relationship with the Employees table to link payroll data to individual employees.
* **Attendance Table**: This tracks daily attendance records of employees, including their status (e.g., Present, Absent, or Overtime). It also connects to the Employees table via a foreign key, ensuring attendance records are associated with specific employees.
* The database supports operations like retrieving salary details, calculating net salary, identifying employees with perfect attendance, and managing payroll and attendance data. Additionally, it facilitates updates, such as promotions, and can identify trends like employees with overtime pay or the longest tenure.
* It provides a structured solution to organize and access employee data, making payroll and attendance management smooth and systematic. Let me know if you'd like more specifics about any part of the database!

# E-R DIAGRAM:

* 
* APPLICATION :
* An Employee Payroll Management System database has numerous practical applications across organizations of all scales. Here's how it can be useful:
* Streamlined Payroll Processing: Automates the calculation of employee salaries, accounting for deductions, bonuses, and overtime. This minimizes manual errors and speeds up the payroll process.
* Attendance Tracking: Keeps precise records of employee attendance, making it easy to calculate wages based on workdays or identify trends such as overtime or absenteeism.
* Employee Data Management: Provides a centralized repository for all employee details, making it easier to manage HR tasks such as promotions, department transfers, or performance reviews.
* Financial Planning and Reporting: Facilitates the generation of detailed reports, including salary trends, deductions, and bonus allocations, aiding in better budget management.
* Regulatory Compliance: Ensures accurate record-keeping and reporting, making it easier to comply with labor laws and tax regulations.
* Decision-Making: Identifies top performers, employees with perfect attendance, or those due for promotions, enabling informed decision-making for HR and management.
* Historical Record Keeping: Maintains historical data for payroll and attendance, which can be useful for audits, resolving disputes, or tracking employee progress.
* Scalability: Adapts to the needs of growing organizations, allowing new employees, departments, and payroll complexities to be integrated seamlessly.

## SQL Queries:

Table Creation Queries:

create database Employee\_Payroll\_Management\_System;

use Employee\_Payroll\_Management\_System;

Employees Table

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Department VARCHAR(50),

Designation VARCHAR(50),

   Salary INT

);

Payroll Table

CREATE TABLE Payroll (

PayrollID INT PRIMARY KEY,

EmployeeID INT,

BasicSalary INT,

Deductions INT,

Bonus INT,

NetSalary INT,

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID));

Attendance Table

CREATE TABLE Attendance (

AttendanceID INT PRIMARY KEY,

EmployeeID INT,

Date DATE,

Status VARCHAR(20),

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)

);

Inserting Values:

Employees table

INSERT INTO Employees VALUES

(1, 'Gourav Gupta', 'HR', 'Manager', 70000),

(2, 'Devi Prasad', 'IT', 'Developer', 80000),

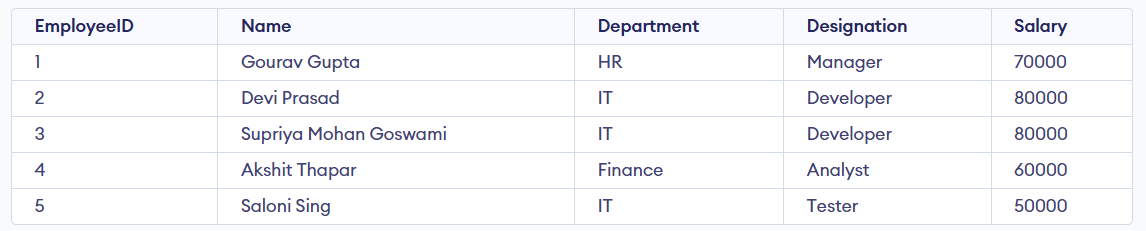
(3, 'Supriya Mohan Goswami', 'IT', 'Developer', 80000),

(4, 'Akshit Thapar', 'Finance', 'Analyst', 60000),

(5, 'Saloni Sing', 'IT', 'Tester', 50000);

Table After Inserting

Select \* from Employees;



Payroll table

INSERT INTO Payroll VALUES

(201, 1, 80000, 6000, 3000, 68000),

(202, 2, 90000, 3000, 5000, 84000),

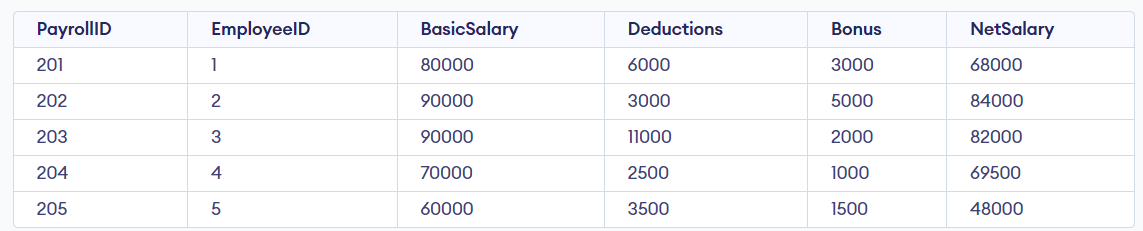
(203, 3, 90000, 11000, 2000, 82000),

(204, 4, 70000, 2500, 1000, 69500),

(205, 5, 60000, 3500, 1500, 48000);

Table After Inserting

Select \* from Payroll;



Attendance Table

INSERT INTO Attendance VALUES

(1, 1, '2025-03-01', 'Present'),

(2, 1, '2025-03-02', 'Present'),

(3, 2, '2025-03-01', 'Present'),

(4, 2, '2025-03-02', 'Overtime'),

(5, 3, '2025-03-01', 'Present'),

(6, 3, '2025-03-02', 'Absent'),

(7, 4, '2025-03-01', 'Present'),

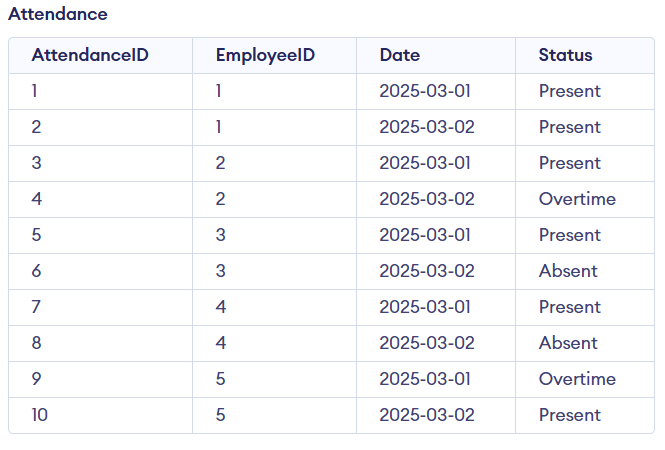
(8, 4, '2025-03-02', 'Absent'),

(9, 5, '2025-03-01', 'Overtime'),

(10, 5, '2025-03-02', 'Present');

Table After Inserting

Select \* from Attendance;



Queries:

Q) Retrieve employee salary details

SELECT Name, Department, Salary

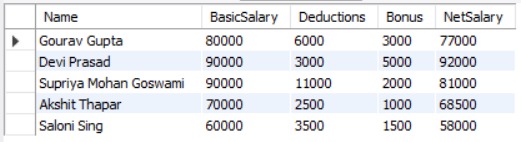
FROM Employees;

  
  
  
  
Q) Calculate net salary after deductions and bonuses.

SELECT Employees.Name, Payroll.BasicSalary, Payroll.Deductions, Payroll.Bonus, (Payroll.BasicSalary - Payroll.Deductions + Payroll.Bonus) AS NetSalary

FROM Payroll

JOIN Employees ON Employees.EmployeeID = Payroll.EmployeeID;

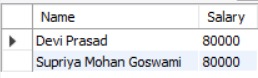
****

**Q) List employees with the highest salary**

SELECT Name, Salary

FROM Employees

WHERE Salary = (SELECT MAX(Salary) FROM Employees);



**Q) Find employees with perfect attendance**

SELECT Employees.Name

FROM Employees

WHERE NOT EXISTS (

SELECT 1 FROM Attendance

WHERE Attendance.EmployeeID = Employees.EmployeeID AND Attendance.Status = 'Absent'

);

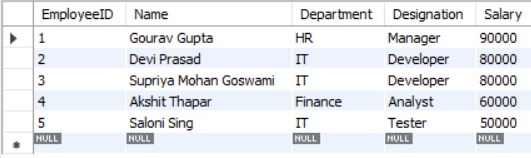


**Q)** **Update salary details after a promotion**

UPDATE Employees

SET Salary = Salary + 10000

WHERE EmployeeID = 1;



**Q) Get employees with the most deductions**

SELECT Employees.Name, Payroll.Deductions

FROM Payroll

JOIN Employees ON Employees.EmployeeID = Payroll.EmployeeID

WHERE Payroll.Deductions = (SELECT MAX(Deductions) FROM Payroll);



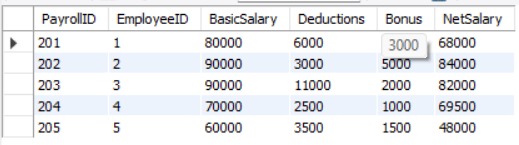
**Q)**  **Retrieve payroll details for a given month**

SELECT DISTINCT P.\*

FROM Payroll P

JOIN Attendance A ON P.EmployeeID = A.EmployeeID

WHERE MONTH(A.Date) = 3 AND YEAR(A.Date) = 2025;



**Q) Identify employees with overtime pay**

SELECT DISTINCT Employees.Name

FROM Attendance

JOIN Employees ON Employees.EmployeeID = Attendance.EmployeeID

WHERE Attendance.Status = 'Overtime'**;**



**Q) Delete payroll records older than a year**

SET SQL\_SAFE\_UPDATES = 0;

DELETE FROM Payroll

WHERE EmployeeID IN (

SELECT E.EmployeeID

FROM Employees E

LEFT JOIN Attendance A ON E.EmployeeID = A.EmployeeID

GROUP BY E.EmployeeID

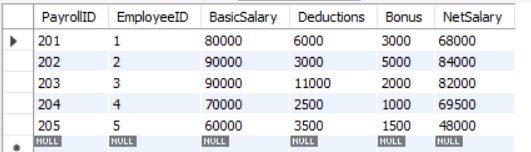
HAVING MAX(A.Date) < DATE\_SUB(CURDATE(), INTERVAL 1 YEAR)

OR MAX(A.Date) IS NULL

);

SET SQL\_SAFE\_UPDATES = 1;

select \* from Payroll;



**Q) Find employees who have been with the company the longest.**

SELECT E.EmployeeID, E.Name, MIN(A.Date) AS StartDate

FROM Employees E

JOIN Attendance A ON E.EmployeeID = A.EmployeeID

GROUP BY E.EmployeeID, E.Name

ORDER BY StartDate ASC;



GitHub Repository:

[**https://github.com/GouravGupta19/DBMS\_Group9\_MiniProject**](https://github.com/GouravGupta19/DBMS_Group9_MiniProject)

# *CONCLUSION:*

Completing this DBMS mini project offered practical insights into the core principles of database development. We began with requirement gathering and conceptual modeling through ER diagrams, which were then converted into precise relational schemas representing real-life data scenarios and rules.

A major part of our work involved applying data normalization techniques to minimize redundancy and enhance data consistency. We also implemented constraints like primary keys, foreign keys, and checks to uphold data integrity and reliability.

Using SQL, we built the database with close attention to ensuring the logical structure aligned with the physical implementation. Additionally, we emphasized query efficiency, using optimized queries and indexing strategies to improve system performance.

This project served as a valuable bridge between theoretical knowledge and real-world application, helping us develop the skills needed to design robust, efficient, and scalable database systems.