Employee Payroll Management System

A Database Solution for Efficient Payroll Management

- Disha Andre - 123B1B078
- Anishka Sah - 123B1B080
- Ankitkumar Shah - 123B1B081

Presentation Outline

1. Introduction

2. Problem Statement & Objectives

3. Entities & Attributes

4. Relationships, Cardinality & Participation

5. ER Diagram

6. Normalization Process
   - First Normal Form (1NF)
   - Second Normal Form (2NF)
   - Third Normal Form (3NF)

7. Final Optimized Tables

8. SQL Implementation
   - Table Creation Rules
   - Table Creation Queries

9. Queries & Constraints
   - DML Queries
   - Constraints & Data Integrity

10. Conclusion

INTRODUCTION

Purpose of the Project

This system is designed to manage employee payroll information efficiently, helping HR departments automate payment processing, track attendance, and manage employee benefits and leave requests.

Importance of Database in Payroll

Databases centralize and store payroll data securely, ensuring accuracy and reducing human errors. A well-structured database can automate repetitive tasks like calculating salaries and deductions.

Problem Statement & Objective

Problem Statement

Manual Payroll Systems Issues:

- Slow, error-prone, difficult to manage, and prone to human mistakes like incorrect calculations and missed deadlines.
- Employees may face delayed or incorrect payments, leading to dissatisfaction and lack of trust in the payroll system.

Objective of the Project:

Automated System:
To develop an efficient, reliable, and secure payroll management system with features for employee data, payroll calculation, attendance tracking, and leave management.

Goals:

1. Ensure accurate payroll calculation
2. Manage employee leave requests efficiently
3. Track employee attendance

Why Use a Database for Payroll?

Security & Accuracy: Payroll data is protected from unauthorized access.

Efficiency: Automates repetitive calculations like salary, deductions, and taxes.

Scalability: Can handle a growing workforce efficiently.

Entities & Attributes

Each entity stores specific payroll-related data, while attributes define the details of each entity.

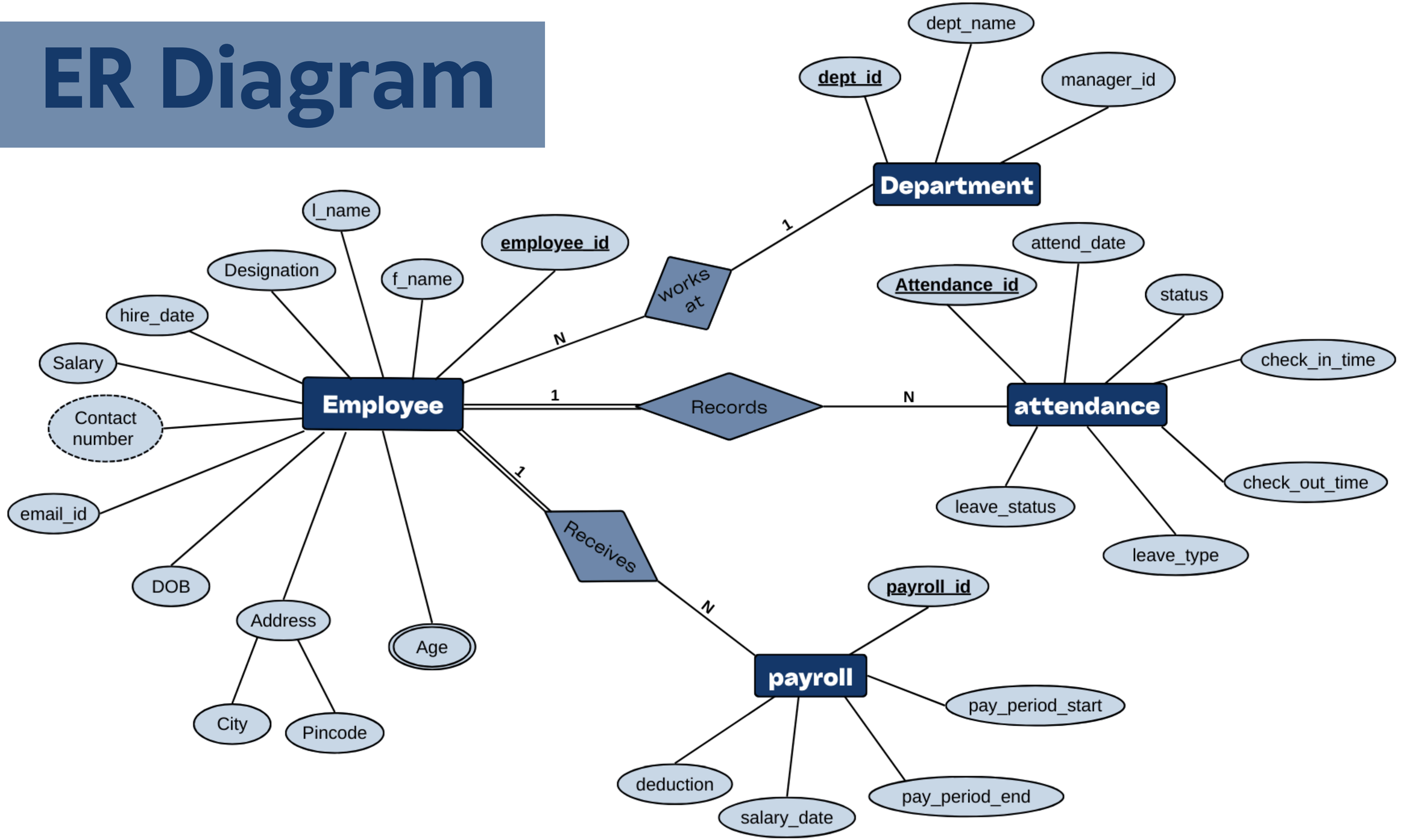
Entity	Attributes
Employee	Stores employee details employee_id (PK), f_name, l_name, designation, salary, hire_date, contact_number, email, address(city, pincode), date_of_birth, gender, dept_id (FK)
Department	Stores department details. dept_id (PK), dept_name, manager_id
Payroll	Stores payroll details. payroll_id (PK), pay_period_start, pay_period_end, salary_date, deductions, employee_id (FK).
Attendance	Tracks employee attendance and leave details. attendance_id (PK), attendance_date, status, check_in_time, check_out_time, leave_type, leave_status, employee_id (FK),

Relationships, Cardinality & Participation

The relationships define how entities are connected, cardinality specifies the number of associations, and participation determines whether relationships are mandatory or optional.

Entity 1	Employee	Employee	Employee
Entity 2	Department	Payroll	Attendance
Relationship Type	Works in	Receives	Has
Cardinality	Many-to-One (N:1)	One-to-Many (1:N)	One-to-Many (1:N)
Participation of Entity 1	Total: Every employee must belong to a department.	Total: Every employee must have at least one payroll record.	Total: Every attendance or leave record must be linked to an employee.
Participation of Entity 2	Partial: A department may exist without employees.	Total: Every payroll record must be linked to an employee.	Partial: An employee may not have attendance records if they have not started working yet.
Description	Each employee belongs to one department, but a department can have multiple employees.	Each employee has multiple payroll records (one per pay cycle).	Each employee has multiple attendance and leave records (one per workday).

ER Diagram


dept_name
dept_id
manager_id
Department
1
Works at
N
l_name
employee_id
Designation
f_name
hire_date
Salary
Contact number
email_id
DOB
Address
Age
City
Pincode
Employee
1
Records
N
attend_date
Attendance_id
status
check_in_time
attendance
check_out_time
leave_status
leave_type
1
Receives
N
payroll_id
payroll
pay_period_start
deduction
salary_date
pay_period_end

Normalization Process

Normalization is the process of organizing data in a database to eliminate redundancy and improve integrity. It involves structuring tables efficiently through normal forms such as 1NF, 2NF, and 3NF.

First Normal Form (1NF):

A relation is in First Normal Form (1NF) if:

- All attributes contain atomic values (no multivalued or composite attributes).
- Each column contains values of a single type.

Example Before 1NF (Unnormalized Table):

-> Multivalued attribute (Phone Numbers contains multiple values in a single field).

```
Employee_ID | Name  | Contact_Numbers
------------|-------|--------------------
101         | John  | 9876543210, 9123456789
```

Example After Applying 1NF:

-> Created a new row for each Phone Number to eliminate multivalued attributes.

```
Employee_ID | Name  | Contact_Number
------------|-------|---------------
101         | John  | 9876543210
101         | John  | 9123456789
```

Normalization Process

Normalization is the process of organizing data in a database to eliminate redundancy and improve integrity. It involves structuring tables efficiently through normal forms such as 1NF, 2NF, and 3NF.

Second Normal Form (2NF):

A relation is in Second Normal Form (2NF) if:

- It is already in 1NF.
- No **partial dependency** (every non-key attribute must be fully dependent on the entire prime key).

Example Before 2NF (Partial Dependency):

-> Salary depends only on Employee_ID, not on the whole composite key (Payroll_ID, Employee_ID).

-> Salary is an Employee-specific value, not a Payroll-specific value.

This is a partial dependency, violating 2NF.

Payroll_ID	Employee_ID	Pay_Period_Start	Pay_Period_End	Deduction	Salary
501	101	2024-02-01	2024-02-28	2000	48000
502	102	2024-02-01	2024-02-28	1500	53000

After 2NF (Good Design - Separate Department Table):

-> To remove partial dependency, we split into two tables:

1. Payroll Table → Stores Payroll-specific details.
2. Employee Table (No Change) → Stores Employee details including Salary.

Payroll_ID	Employee_ID	Pay_Period_Start	Pay_Period_End	Deduction
501	101	2024-02-01	2024-02-28	2000
502	102	2024-02-01	2024-02-28	1500

Employee_ID	Name	Designation	Salary
101	John	Engineer	48000
102	Alice	Analyst	53000

Normalization Process

Normalization is the process of organizing data in a database to eliminate redundancy and improve integrity. It involves structuring tables efficiently through normal forms such as 1NF, 2NF, and 3NF.

Third Normal Form (3NF):

A table is in 3NF if:

- It is already in 2NF.
- There are no **transitive dependencies** (Attributes should depend only on the prime key, not on another non-key attribute).

Example (Before 3NF - Transitive Dependency in Employee Table):
-> Dept_Name depends on Dept_ID, not directly on Employee_ID. This is a transitive dependency because Dept_Name is dependent on Dept_ID, not the primary key (Employee_ID).

After 3NF (Good Design - Separate Department Table):
-> Move Dept_Name to a separate Department table. Now, very non-key column depends only on the primary key, and no transitive dependencies exist.

Employee_ID	Name	Dept_ID	Dept_Name
101	John	D1	HR
102	Alice	D2	IT

Employee_ID	Name	Dept_ID
101	John	D1
102	Alice	D2

Dept_ID	Dept_Name
D1	HR
D2	IT

Final Optimized Tables

After applying 1NF, 2NF, and 3NF, the database schema is optimized to avoid redundancy, improve data integrity, and maintain efficiency. Below is the final structure of tables along with their attributes and relationships.

Department Table

Field	Type	Null	Key
dept_id	varchar(10)	NO	PRI
dept_name	char(50)	NO	UNI
manager_id	int	YES	UNI

Employee Table

Field	Type	Null	Key
employee_id	int	NO	PRI
f_name	char(30)	NO	
l_name	char(30)	NO	
Designation	char(50)	NO	
hire_date	date	NO	
Salary	int	NO	
Email	varchar(40)	NO	UNI
DOB	date	NO	
gender	enum('Male','Female')	YES	
city	char(30)	YES	
Pincode	int	YES	
Dept_id	varchar(10)	YES	MUL

Employee_Contact Table

Field	Type	Null	Key
Employee_id	int	NO	PRI
Contact_no	bigint	NO	PRI

Payroll Table

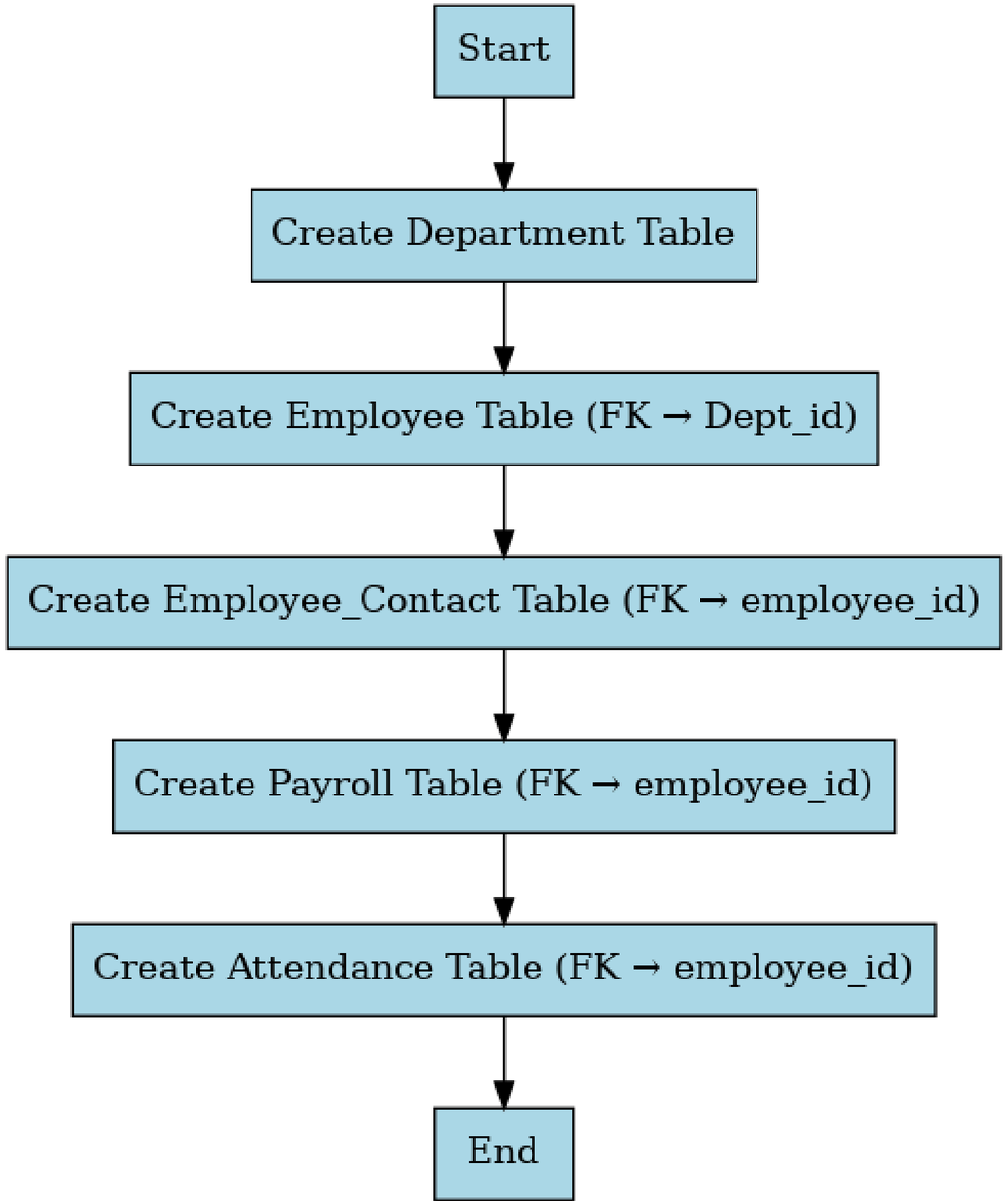
Field	Type	Null	Key
payroll_id	int	NO	PRI
payroll_period_start	date	NO	
payroll_period_end	date	NO	
salary_date	date	NO	
deduction	int	YES	
Employee_id	int	YES	MUL

Attendance Table

Field	Type	Null	Key
attendance_id	int	NO	PRI
attend_date	date	NO	
Status	enum('Present','Absent','Holiday','Leave')	NO	
Check_in_time	time	YES	
check_out_time	time	YES	
leave_type	varchar(100)	YES	
leave_status	enum('Pending','Approved','Rejected')	YES	
Employee_id	int	NO	MUL

SQL Implementation : Table Creation

Step-by-Step Process for Creating Relational Tables



SQL Implementation: Table Creation

Key rules and best practices followed while converting the ER diagram into relational tables to ensure data integrity and efficient database design.

i) Rule for Primary Keys:

Employee_id is underlined in the Employee entity, indicating it's the primary key. The rule is that each table should have a primary key to uniquely identify its rows.

ii) Rule for Derived Attributes:

The Employee table includes Age as a derived attribute, which should generally not be stored in the database to avoid redundancy and maintain data integrity, as it can be calculated from existing attributes (i.e. DOB).

iii) Rule for Multi-valued Attributes:

In the ER diagram, since an employee can have multiple contact numbers, a separate table should be created to store these values, with a foreign key linking back to the Employee table.

iv) Rule for One-to-Many Relationships:

In the ER diagram, the one-to-many relationship between Employee and Payroll means one employee can have many payroll records. The rule is that the primary key from the "one" side (Employee) becomes a foreign key in the "many" side (Payroll).

Table Creation & Sample Data

SQL tables were created to store and manage employee, payroll, attendance, and department records efficiently.

Department Table

dept_id	dept_name	manager_id
D101	Human Resources	201
D102	Finance	202
D103	Information Technology	203
D104	Sales and Marketing	204
D105	Research and Development	205

Employee_Contact Table

Employee_id	Contact_no
101	9123456789
101	9876543210
102	9898989898
103	9988776655
104	9556677889
105	9001234567

Employee Table

attendance_id	attend_date	Status	Check_in_time	check_out_time	leave_type	leave_status	Employee_id
1	2024-02-01	Present	09:00:00	17:00:00	NULL	NULL	101
2	2024-02-01	Present	09:15:00	17:30:00	NULL	NULL	102
3	2024-02-01	Absent	NULL	NULL	NULL	NULL	103
4	2024-02-01	Holiday	NULL	NULL	NULL	NULL	104
5	2024-02-02	Leave	NULL	NULL	Sick Leave	Approved	105

Payroll Table

employee_id	f_name	l_name	Designation	hire_date	Salary	Email	DOB	gender	city	Pincode	Dept_id
101	Amit	Sharma	HR Manager	2020-01-15	75000	amit.sharma@email.com	1985-05-10	Male	Delhi	110001	D101
102	Priya	Iyer	Finance Manager	2019-03-12	80000	priya.iyer@email.com	1987-09-20	Female	Mumbai	400001	D102
103	Rajesh	Verma	IT Head	2018-07-25	95000	rajesh.verma@email.com	1982-11-05	Male	Bengaluru	560001	D103
104	Meera	Nair	Marketing Head	2021-05-30	72000	meera.nair@email.com	1990-02-14	Female	Pune	411000	D104
105	Suresh	Patil	Research Scientist	2017-09-10	88000	suresh.patil@email.com	1984-07-18	Male	Pune	NULL	D105

Attendance Table

payroll_id	payroll_period_start	payroll_period_end	salary_date	deduction	Employee_id
1	2024-01-01	2024-01-31	2024-02-01	2000	101
2	2024-01-01	2024-01-31	2024-02-01	1500	102
3	2024-01-01	2024-01-31	2024-02-01	0	103
4	2024-01-01	2024-01-31	2024-02-01	1200	104
5	2024-02-01	2024-02-28	2024-03-01	2500	105

Implementation- queries

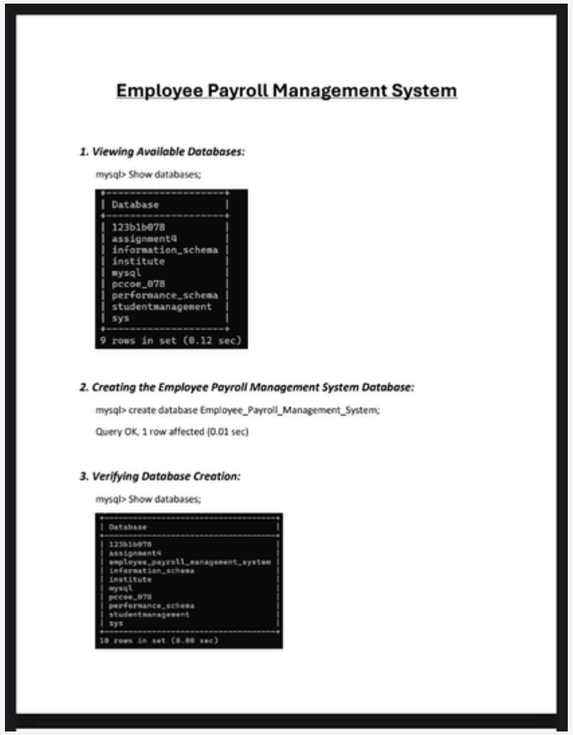
SQL queries were used to create, modify, and manage the Employee Payroll Management System database.

The queries include:

- **DDL (Data Definition Language)** – Creating & modifying tables.
- **DML (Data Manipulation Language)** – Inserting, updating, and deleting records.
- **DQL (Data Query Language)** – Retrieving data using SELECT

Full Query Document:
https://drive.google.com/file/d/1cyWP6lzK0f468m_GUgJwCvOmApWDuH1q/view?usp=drive_link

All SQL queries, including table creation, data insertion, updates, and constraints, are available in the linked document.



Employee Payroll Management System

1. Viewing Available Databases:

mysql> Show databases;

```
+--------------------+
| Database           |
+--------------------+
| 123b1b078          |
| assignment4        |
| information_schema |
| institute          |
| mysql              |
| pccoe_078          |
| performance_schema |
| studentmanagement  |
| sys                |
+--------------------+
9 rows in set (0.12 sec)
```

2. Creating the Employee Payroll Management System Database:

mysql> create database Employee_Payroll_Management_System;

Query OK, 1 row affected (0.01 sec)

3. Verifying Database Creation:

mysql> Show databases;

```
+--------------------+
| Database           |
+--------------------+
| 123b1b078          |
| assignment4        |
| employee_payroll_management_system |
| information_schema |
| institute          |
| mysql              |
| pccoe_078          |
| performance_schema |
| studentmanagement  |
| sys                |
+--------------------+
10 rows in set (0.00 sec)
```

Implementation- queries

SQL queries were used to create, modify, and manage the Employee Payroll Management System database.

Example Queries:

Creating Table (DDL):

```
mysql> create table department (
-> dept_id varchar(10) primary key,
-> dept_name char(50) unique not null,
-> manager_id int unique
-> );
```

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| dept_id    | varchar(10) | NO   | PRI | NULL    |       |
| dept_name  | char(50)    | NO   | UNI | NULL    |       |
| manager_id | int         | YES  | UNI | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.04 sec)
```

Inserting Data (DML Example):

```
mysql> insert into Department values
-> ('D101', 'Human Resources', 201),
-> ('D102', 'Finance', 202);
```

```
+---------+-----------------------------+------------+
| dept_id | dept_name                   | manager_id |
+---------+-----------------------------+------------+
| D101    | Human Resources             |        201 |
| D102    | Finance                     |        202 |
```

Implementation- queries

SQL queries were used to create, modify, and manage the Employee Payroll Management System database.

Retrieving Data (DQL- SELECT):

mysql> select employee_id, f_name from employee where salary between 45000 and 50000;

```
+-------------+---------+
| employee_id | f_name  |
+-------------+---------+
|         108 | Neha    |
|         109 | Arjun   |
|         111 | Ravi    |
+-------------+---------+
3 rows in set (0.00 sec)
```

Updating Records (DML):

mysql> update employee set salary = 90000 where Employee_id = 105;

```
+-------------+---------+--------+
| employee_id | f_name  | salary |
+-------------+---------+--------+
|         101 | Amit    |  75000 |
|         102 | Priya   |  80000 |
|         103 | Rajesh  |  95000 |
|         104 | Meera   |  72000 |
|         105 | Suresh  |  90000 |
+-------------+---------+--------+
```

Deleting a Record (DML Example):

mysql> delete from employee_contacts where employee_id = 109;

Ensuring Data Integrity & Validation:

mysql> delete from employee_contacts where employee_id = 109;

```
mysql> select * from employee where dept_id is null;
Empty set (0.04 sec)
```

Constraints & Data Integrity

Constraints maintain data accuracy and consistency by enforcing rules and relationships in the database.

Primary Key (Entity Integrity)	Ensures each record is unique and acts as the table's identifier. employee_id uniquely identifies each employee.
Foreign Key (Referential Integrity)	Links two tables and maintains referential integrity. dept_id in Employee table links to Department table, ensuring valid assignments.
Unique Constraint (Domain Integrity)	Ensures no duplicate values in a column. email in Employee table ensures each employee has a unique email ID.
NOT NULL (Domain Integrity)	Prevents storing NULL values in critical columns. Salary cannot be NULL, ensuring every employee has a salary.
CHECK Constraint (Domain Integrity)	Ensures values meet specific conditions. Salary > 0 ensures that salary is always a positive number.
ENUM (Domain Integrity)	Restricts column values to predefined options. gender ENUM('Male', 'Female', 'Other')

CONCLUSION:

- Successfully designed and implemented a structured relational database for an Employee Payroll Management System.
- Applied Normalization (1NF, 2NF, 3NF) to eliminate redundancy and improve efficiency.
- Ensured data integrity using Primary Key, Foreign Key, Constraints (NOT NULL, UNIQUE, CHECK, ENUM).
- Used SQL queries to manage payroll, attendance, and employee records efficiently.

THANK YOU!

This project demonstrates how a structured database can enhance payroll management by ensuring data accuracy, efficiency, and automation.