Employee Database

SQL Project Documentation

1. Introduction

1.1 Project Overview

The Employee Database Project is a practical exercise in relational database management, designed to address core human resources and payroll queries using SQL. The project demonstrates advanced query-writing skills for workforce analytics, benchmarking, and pattern detection. The focus is on leveraging data to provide insights that inform HR decision-making, optimize workforce management, and enhance organizational efficiency

Problem Statement & Objectives

Organizations depend on accurate, timely information about their employees to manage payroll, track recruitment, and measure departmental performance. This project aims to answer key business questions about employee compensation, tenure, departmental structure, and workforce diversity through targeted SQL queries. The core objectives are:

- To design efficient SQL queries for classic HR analytics problems.
- To accurately retrieve, aggregate, and analyse employee and departmental data.
- To demonstrate advanced SQL techniques such as window functions, ranking, grouping, and string manipulation.

Scope

The project focuses on the analysis of employee and department records through a series of ten targeted SQL queries. It involves querying a single centralized employee information table, emp_info, housed within a PostgreSQL or MySQL relational database environment. The work emphasizes the data layer and does not extend to a user interface.

1.2 Technologies Used

- Database: MySQL
- Language: SQL (DQL, Aggregations, Window functions, CASE Statements, Sub-Queries, Sort functions).

2.Database Schema

Table: emp_info

Stores all details for each employee in the organization.

Column	**Data Type**	**Description**
emp_id	INT (PK)	Unique Employee Identifier
Full_name	VARCHAR	Employee Name
age	INT	Employee Age
department	VARCHAR	Department Name
salary	NUMERIC	Salary Amount
Hire_date	DATE/VARCHAR	Date of Joining

This single-table structure is used for all queries, providing a unified source for employee and departmental analytics.

3. SQL Queries and Business Insights

Each question is listed with the business motivation, the SQL approach, the code, and the purpose of the analysis, following the style of the Online Book Store project.

SQL Queries:

1. Retrieve the Second-Highest Salary of an Employee

Business Question: What is the second-highest salary among all employees?

Approach: This common HR query identifies salary tiers and is useful for benchmarking. Techniques include subqueries, window functions, and offset/limit.

```
-- Method 1: Subquery

SELECT MAX (salary)

FROM emp_info

WHERE salary < (SELECT MAX (salary) FROM emp_info);
```

```
-- Method 2: Window Function

SELECT salary

FROM (

  SELECT salary, DENSE_RANK () OVER (ORDER BY salary DESC) AS rnk

  FROM emp_info

) AS ranked

WHERE rnk = 2;
```

-- Method 3: Offset

```
SELECT DISTINCT salary

FROM emp_info

ORDER BY salary DESC

LIMIT 1 OFFSET 1;
```

Insight: These approaches reliably deliver the second-highest salary, supporting compensation benchmarking

2. Find the Second-Highest Salary for Each Department

Business Question: What is the second-highest salary in each department?

Approach: Use window functions (DENSE_RANK () with partitioning) for department-wise ranking.

SQL Query:

```
SELECT department, salary FROM (SELECT department, salary, DENSE_RANK ()

OVER (PARTITION BY department ORDER BY salary DESC) AS rnk FROM emp_info)

 AS Ranked

WHERE rnk = 2;
```

Insight: Identifying runner-up salaries guides fair compensation policies across departments.

3.Find the Maximum Salary of the Top 5 Highest-Paid Employees in Each Department.

Business Question: What is the maximum salary among the top 5 earners in every department?

Approach: Use DENSE_RANK () over each department, limit to the top 5, and aggregate with MAX.

SQL Query:

```
SELECT department, MAX (salary) FROM (SELECT department, salary, DENSE_RANK()
OVER (PARTITION BY department ORDER BY salary DESC) AS rnk FROM emp_info)
AS ranked WHERE rnk <= 5
GROUP BY department LIMIT 5;
```

Insight: Highlights top tier pay per department to assist with leadership compensation benchmarking.

4. Find the Department with the Highest Number of Employees

Business Question: Which department employs the most people?

Approach: Aggregate with COUNT (*), group by department, and pick the highest count.

SQL Query:

```
SELECT department, COUNT (*)
FROM emp_info
GROUP BY department
ORDER BY COUNT (*) DESC
LIMIT 1;
```

Insight: Informs team size distribution and identifies departments with the greatest management complexity.

5. Fetch All Employees Whose Salary Is Greater Than the Minimum Salary

Business Question: Who earns more than the Minimum salary employee?

Approach: Compare each salary to a subquery calculating the minimum.

SQL Query:

```
SELECT * FROM emp_info

WHERE salary > (SELECT MIN (salary) FROM emp_info);
```

Insight: Reveals high-performers and aids compensation planning. (Note: Provided SQL compared to minimum salary; adjusted here for average salary per business context.).

6. Find All Employees Who Joined in the Year 2020

Business Question: Which employees were recruited in 2020?

Approach: Parse the year from the Hire_date and filter.

SQL Query:

```
SELECT * FROM emp_info

WHERE YEAR(STR_TO_DATE(Hire_date,'%d/%m/%Y')) = 2020.
```

Insight: Tracks hiring dynamics, helps with cohort analysis and anniversary recognition.

7. Fetch the First and Last Record from the Table

Business Question: What are the earliest and most recent records in the employee list?

Approach: Order by emp_id and use LIMIT.

SQL Query:

```
(SELECT * FROM emp_info ORDER BY emp_id ASC LIMIT 1)

UNION

(SELECT * FROM emp_info ORDER BY emp_id DESC LIMIT 1);
```

Insight: Useful for auditing the oldest and newest entries in the workforce roster.

8. Display Employees Grouped by Their Age Brackets (20-30, 31-40, etc.)

Business Question: How are employees distributed by age group?

Approach: Categorize with CASE expressions and aggregate.

SQL Query:

```
SELECT CASE

    WHEN age BETWEEN 20 AND 30 THEN '20-30'

    WHEN age BETWEEN 31 AND 40 THEN '31-40'

    ELSE '41+'

  END AS age_bracket,

  COUNT (*) FROM emp_info

  GROUP BY age_bracket;
```

Insight: Supports demographic reporting and diversity tracking in HR planning.

9. Fetch the Details of Employees with the Same Salary

Business Question: Which employees share their compensation levels?

Approach: Find salaries with more than one employee, then retrieve full details.

SQL Query:

```
SELECT * FROM emp_info
```

```
WHERE salary IN (SELECT salary FROM emp_info

GROUP BY salary

HAVING COUNT (*) > 1);
```

Insight: Highlights duplicate salaries, important for identifying salary bands and equity

10. Retrieve Employee Names and Salaries in a Single String

Business Question: How to display names and salaries together for reporting?

Approach: Concatenate employee name and salary fields.

SQL Query:

```
SELECT CONCAT (Full_name, salary) AS employees_information FROM emp_info;
```

Insight: Enhances presentation and rapid information dissemination, especially useful in dashboards and exports.

4. Key Learnings and Challenges

- **Advanced SQL**: Proficiency with window functions, grouping, and string operations strengthens analytical capabilities.
- **Handling Edge Cases**: Careful use of joins, partitions, and date parsing is vital for correctness, especially in cohort and ranking queries.
- **Data Quality**: Consistent formatting of fields like Hire_date ensures reliable results and parsing success.

5. Conclusion and Future Scope

This Employee Database SQL Project demonstrates the practical value of SQL for HR and payroll analytics. By addressing a variety of business questions through structured queries, it enables organizations to gain actionable insights from their workforce data. Future enhancements might include:

- Automating key reports with stored procedures.
- Extending analytics to include performance and absenteeism.
- Integration with business intelligence tools for real-time dashboards

Thank you for your time