TITLE

EMPLOYEES PERFORMANCE

1. INTRODUCTION

This project focuses on managing and analyzing employee data using SQL. It involves creating tables, inserting records, updating data, and writing various queries to extract meaningful insights from the data.

2. DATABASE SETUP

Creating a database named office.

```
Create database office;

Use office;
```

3. TABLE STRUCTURE

EMPLOYEES TABLE

Column	Data Type	Description
id	INT	Employee ID
name	VARCHAR	Employee Name
age	INT	Age
gender	VARCHAR	Gender
department	VARCHAR	Department Name
salary	DECIMAL	Salary Amount
joining_date	DATE	Joining Date
performance_score	DECIMAL	Performance Score

experience	INT	Years of Experience
status	VARCHAR	Employment Status
location	VARCHAR	Location
session	VARCHAR	Shift Session

BONUS TABLE

Column	Data Type	Description
employee_id	INT	Linked to employees.id
name	VARCHAR	Employee Name
bonus_amount	DECIMAL	Bonus Amount
department	VARCHAR	Department Name
bonus_date	DATE	Date of Bonus

4. SQL QUERIES

-- SELECT & WHERE --

-- 1.Select all columns for employees from the 'IT' department.

```
select * from employees where department="IT";
```

-- 2. Retrieve the names and salaries of employees whose salary is greater than 8000.

```
select name , salary from employees where salary>8000;
```

-- 3.Show employees who joined after '2020-01-01'.

```
select * from employees where joining_date>'2020-01-01';
```

-- 4. List employees who are either in the 'HR' or 'Finance' department.

```
select * from employees where department="IT" or department="Finance";
```

-- 5. Find employees whose performance score is NULL.

```
select * from employees where performance_score is null;
```

-- 6. List all 'Inactive' employees who are located in 'Los Angeles'.

```
select * from employees where location="Los Angeles";
```

-- Aggregate Functions & GROUP BY --

-- 7. What is the average salary of employees in each department?

```
select  avg(salary) as avg_salary from employees;
```

-- 8. Count how many employees work in each location.

```
select location ,count(*) as location_count from employees group by location;
```

-- 9. What is the highest and lowest salary in the 'Sales' department?

```
select min(salary) as lowest_salary , max(salary) as highest_salary from employees where
department="Sales";
```

-- 10. Calculate total salary paid to 'Active' employees.

```
select department, sum(salary) as sum_of_salary from employees where status = "active" group by
department;
```

-- ORDER BY & LIMIT –

-- 11. Show top 5 highest-paid employees.

select salary as top_salaries from employees order by salary desc limit 5;

-- 12. List employees sorted by joining date, newest first.

select joining_date as newest_joining from employees order by joining_date desc;

-- 13. Get the names and departments of the 10 youngest employees.

select age as youngest_employees from employees order by age asc limit 10;

-- ALTER & DISTINCT --

-- 13. Alter the main table to add a new column email.

alter table employees add column email varchar(20);

-- 14. List all unique departments in the company.

select distinct department as unique_departments from employees;

-- 15. Show the unique combinations of department and location across all employees.

select distinct department, location from employees;

-- IN/NOT IN & LIKE --

-- 16. List all employees whose name starts with 'J'.

select * from employees where name like "j%";

-- 17. Find employees not in 'Morning' or 'Evening' sessions.

```
select * from employees where session not in ('Morning','Evening');
```

-- 18. Show employees whose location is in ('New York', 'Phoenix') and are 'Active'.

```
select * from employees where location in ('New York', 'Phoenix') and status="Active";
```

-- Subqueries --

-- 19.Find employees whose salary is above the average salary of all employees.

```
select * from employees where salary>(select avg(salary) from employees);
```

-- 20. Get employees who have the same experience as the most experienced employee.

```
select * from employees where experience = (select max(experience) from employees);
```

-- CASE & IF --

-- 21. Write a query that adds a column 'Seniority' as 'Junior', 'Mid', or 'Senior' based on experience:

-- Junior: <= 2 years , Mid: 3–6 years , Senior: > 6 years

```
alter table employees add column Seniority varchar(20);

select name , exprience , if(experience<=2 , "Junior" , if(experience>=3 , "Mid", if("experience" >6 ,
"Junior" ,  " "))) as emp_exp from employees;
```

-- 22. Use CASE to label salary ranges:

-- <4000: 'Low' , 4000–8000: 'Medium' , 8000: 'High'

```
select name , salary,

case

when salary<4000  then "Low"

when salary>=4000 and salary<=8000 then "Medium"
```

```
else "High" end salary_range from employees;
```

-- HAVING & COUNT --

-- 23. Find departments where average salary is more than 7000.

```
select department , avg(salary) as high_salary from employees group by department having
avg(salary)>7000;
```

-- 24. Show locations having more than 5 employees.

```
select location, count(*) as employee_count from employees group by location having count(*)>5;
```

-- UPDATE & DELETE --

```
set sql_safe_updates=0;
```

-- 25. Update the status of all employees with experience greater than 7 to "Retired".

```
 update employees set status = "Retired" where experience>7;
```

-- 26. Delete all employees whose salary is below 3000.

```
delete from employees where salary<3000;
```

-- 27. Update performance score to 3.0 where it is currently NULL.

```
update employees set performance_score = 3.0 where performance_score is null;
```

-- Joins --

-- 28. Show all employees who received a bonus, along with their department, bonus amount, and bonus date.

```
select employees.name, employees.department, bonus.bonus_amount, bonus.bonus_date from
employees inner join bonus on employees.id = bonus.employee_id;
```

-- 29. List employees who didn't receive any bonus.

```
select employees.id, employees.name, employees.department from employees left join bonus on
employees.id = bonus.employee_id where Bonus.employee_id is null;
```

-- 30. For each department, show employee names and bonus amounts, but only for bonuses greater than 600.

```
select employees.name, employees.department, bonus.bonus_amount from employees inner join
bonus on employees.id = bonus.employee_id where bonus_amount>600;
```

5. LEARNINGS

- Practiced creating and altering SQL tables
- Learned how to use GROUP BY, HAVING, CASE, JOIN, and UPDATE
- Understood real-world usage like filtering data and working with NULLs
- Created a bonus table and joined it with the employee table to find relationships

6. CONCLUSION

This project helped me understand how SQL can be used to manage and analyze employee data effectively. From data creation to updates and complex joins, I practiced a wide range of SQL queries. The project shows how structured data can provide valuable insights when queried correctly.