DDL

1. **Create a Table:**
   Write a query to create a table named employees with columns id (INT, Primary Key), name (VARCHAR(100)) NOT NULL, designation (VARCHAR(50)), and salary (DECIMAL) with default value 5000.
2. **Add a Column:**
   Add a column joining_date (DATE) to the employees table.
3. **Modify a Column:**
   Modify the data type of the salary column in the employees table to FLOAT.
4. **Drop a Column:**
   Write a query to drop the joining_date column from the employees table.
5. **Rename a Column:**
   Rename the column designation to job_title in the employees table.
6. **Rename a Table:**
   Rename the employees table to company_employees.
7. **Drop a Table:**
   Write a query to drop the employees table.
8. **Truncate a Table:**
   Write a query to truncate all data from the employees table without dropping its structure.

DML

9. **Insert three employees with the following details:**
   (3, 'Alice Brown', 'Analyst', 7000), (4, 'Bob White', 'Tester', 8000), (5, 'Charlie Black', 'Designer', 6000)
10. **Select All Rows:**
    Write a query to fetch all the columns and rows from the employees table.
11. **Select Specific Columns:**
    Write a query to fetch only the name and designation columns from the employees table.
12. **Select with Condition:**
    Write a query to fetch details of employees whose salary is greater than 8000.
13. **Select with LIKE Operator:**
    Write a query to find employees whose name starts with 'J'.
14. **Update Salary:**
    Update the salary of the employee with id = 1 to 12000.
15. **Update Multiple Columns:**
    Update the designation to 'Senior Developer' and salary to 15000 for the employee with id = 2.
16. **Delete a Row:**
    Delete the record of the employee with id = 5.
17. **Delete Rows with a Condition:**
    Delete all employees whose salary is less than 7000.
18. **Select with ORDER BY:**
    Write a query to fetch all rows from the employees table and order them by salary in descending order.

DISTINCT, AND, OR, NOT, BETWEEN and LIKE

DISTINCT

19. List unique designations in the employees table:
20. Find the unique combinations of designation and salary:
21. Count the number of unique designations:

AND

22. Find employees with a salary greater than 5000 and a designation of 'Manager':
23. Retrieve employees whose name starts with 'A' and salary is between 5000 and 10000:
24. Find employees with a designation of 'Developer' and a salary less than 20000:

OR

25. Find employees with a salary greater than 50000 or a designation of 'Intern':
26. Retrieve employees whose name starts with 'J' or salary is exactly 5000:
27. List employees who are either 'Manager' or 'Team Lead':

NOT

28. Find employees who do not have a salary equal to 5000:
29. Retrieve employees whose designation is not 'Developer':
30. Find employees whose name does not contain the letter 'e':

BETWEEN

31. Find employees whose salary is between 10000 and 30000:
32. Retrieve employees with id values between 5 and 15:
33. Find employees whose salary is not between 5000 and 20000:

LIKE

34. Find all employees whose names start with the letter `A`.
35. Find all employees whose names end with the letter `n`.
36. Find all employees whose names contain the substring `John`.
37. Find all employees whose names are exactly 5 characters long.
38. Find all employees whose designations start with `M` and end with `r`.
39. Find all employees whose names have `e` as the second letter.
40. Find all employees whose names do not contain the letter `z`.

Aggregate Functions

41. Write a query to calculate the total salary of all employees in the `employees` table.
42. Write a query to find the average salary of employees.
43. Write a query to count the total number of employees in the table.
44. Write a query to find the highest salary among employees.
45. Write a query to find the lowest salary in the `employees` table.
46. Write a query to count the number of employees.

Group by, Having, Ordery by

Table – employees(employee_id, name, department, salary, hire_date)

47. Write a query to find the total salary per department.
48. Find the average salary of employees in the 'HR' department.
49. List departments with more than 1 employee and their average salary.
50. Order employees by salary in descending order.
51. Find the highest salary in each department.
52. List all employees who have a salary greater than 5000 and order them by name.
53. Count the number of employees in each department and only show departments with more than 1 employee.
54. Find the employees who were hired after '2020-01-01' and order them by hire_date.
55. Get the sum of salaries of employees hired in 2020 or later.
56. List employees who earn less than 5000 and order them by salary in ascending order.
57. Find the department with the maximum average salary.
58. List the departments that have employees with a salary greater than 6000.

Stored Procedure, Views and Triggers - 01

Tables:

1. `employees`

employee_id	name	department	salary
1	Alice	HR	5000
2	Bob	IT	6000
3	Charlie	HR	5500

2. `departments`

department_id	department_name
1	HR
2	IT

3. `projects`

project_id	project_name	department_id
101	Project A	1
102	Project B	2

1. Create a Stored Procedure to Get Employee Salary by Department
2. Create a Stored Procedure to Update Employee Salary
3. Create a Stored Procedure to Insert a New Employee
4. Create a Stored Procedure to Get Employees with Salary Greater Than a Certain Amount
5. Create a Stored Procedure to Delete Employee by ID
6. Create a View to Show Employees and Their Departments
7. Create a View to Show Employees with Salary Above 6000
8. Create a View to Show Projects with Their Department Names

9. Create a View to Get Employee and Project Details
10. Create a View to Show the Total Salary of Employees in Each Department
11. Create a Trigger to Automatically Update Employee Salary History **(Note : when update salary salary history table will be updated : salary_history_table(employee_id, old_salary, new_salary, change_date))**
12. Create a Trigger to Prevent Deleting Employees with Salary Above 7000
13. Create a Trigger to Automatically Update Department's Average Salary After Salary Change
14. Create a Trigger to Insert a Record in the Audit Table After Employee Insert **(Note : audit_table(employee_id, action, action_date) – Example(1,'Insert','1-1-2025))**
15. Create a Trigger to Update Project's Department When Department ID Changes

Stored Procedure, Views and Triggers - 02

Database Schema Overview

- **Students**: StudentID, Name, Department, DOB, Email
- **Courses**: CourseID, CourseName, Credits, Department
- **Faculty**: FacultyID, FacultyName, Department, Email
- **Enrollments**: EnrollmentID, StudentID, CourseID, Grade
- **Teaches**: FacultyID, CourseID, Semester

Stored Procedures Questions

1. Write a stored procedure to insert a new course into the Courses table, specifying the course ID, name, credits, and department.
2. Create a stored procedure to retrieve the list of students enrolled in a particular course.
3. Develop a stored procedure to update the grade of a student for a specific course.
4. Write a stored procedure to delete a student record based on their StudentID.
5. Create a stored procedure to fetch all courses taught by a specific faculty member in a given semester.

Views Questions

1. Create a view that displays the list of students and their respective departments.
2. Design a view to show the details of courses, including the faculty teaching them and the semester in which they are offered.
3. Write a view to display all students along with the courses they are enrolled in and their grades.
4. Create a view that lists the average grade for each course.
5. Design a view that shows all faculty members and the number of courses they are teaching.

Triggers Questions

1. Write a trigger to automatically update the total number of students in a department when a new student is added to the Students table.
2. Create a trigger to prevent deletion of a course if students are currently enrolled in it.
3. Write a trigger to log changes (INSERT, UPDATE, DELETE) made to the Enrollments table into a separate EnrollmentLogs table.
4. Create a trigger to set the grade to 'F' if it is entered as NULL or an invalid value in the Enrollments table.
5. Write a trigger to send an alert (log entry) whenever a faculty member is assigned more than 5 courses in a single semester.

Sub-Query (From Above Table)

1. Find Employees with Salary Greater Than the Average Salary of Their Department
2. Get Employees Who Do Not Belong to the 'IT' Department
3. **List Employees Who Are Assigned to a Project**
4. Find the Employee with the Highest Salary in the 'HR' Department
5. Get Departments That Have More Than 2 Employees
6. List Employees Whose Salary is Above the Average Salary of All Employees
7. Find Employees Who Are Not Assigned to Any Project
8. List Departments That Do Not Have Any Employees with a Salary Above 6000
9. Get Employees Who Earn More Than the Employee with the Highest Salary in the IT Department
10. **Find Projects That Do Not Belong to the HR Department**

JOIN

1. `Employees`

Column Name	Data Type	Description
EmployeeID	INT PRIMARY KEY	Unique ID for each employee
Name	VARCHAR(50)	Name of the employee
DepartmentID	INT	Foreign key to Departments
Salary	DECIMAL(10, 2)	Salary of the employee

2. `Departments`

Column Name	Data Type	Description
DepartmentID	INT PRIMARY KEY	Unique ID for each department
DepartmentName	VARCHAR(50)	Name of the department
ManagerID	INT	ID of the department manager

3. `Projects`

Column Name	Data Type	Description
ProjectID	INT PRIMARY KEY	Unique ID for each project
ProjectName	VARCHAR(50)	Name of the project
EmployeeID	INT	Foreign key to Employees

1. **Inner Join**: List all employees and their departments.
2. **Inner Join with Aggregate**: Find the total salary paid in each department.
3. **Left Join**: List all employees, including those not assigned to any department.
4. **Right Join**: List all departments and their employees (including empty departments).
5. **Full Outer Join**: Find all employees and departments, even if no match exists.
6. **Cross Join**: Create all possible pairs of employees and projects.
7. **Self Join**: Find employees who work in the same department as their colleagues.
8. **Inner Join with Projects**: List all employees assigned to projects.
9. **Join with Aggregate and HAVING Clause**: Find departments with more than 2 employees.
10. **Join with Subquery**: Find the highest-paid employee in each department.
11. **Anti Join**: Find employees not assigned to any project.
12. **Join with Aggregate Function**: Find the average salary for each department.
13. **Multiple Joins**: List all employees, their departments, and projects.
14. **Join with MAX Function**: Find the department with the highest total salary.
15. **Join on Multiple Conditions**: List employees working under a specific manager.