SQL TASK

1. SELECT ALL: Retrieve all columns and all rows from the Employees table.

```
SELECT * FROM Employees;
```

2. SELECT SPECIFIC COLUMNS: Retrieve only the name and salary columns.

```
SELECT name, salary FROM Employees;
```

3. SELECT DISTINCT: Retrieve only unique department IDs present in the Employees table.

```
SELECT DISTINCT department_id FROM Employees;
```

4. COUNT ALL ROWS: Calculate the total number of employees.

```
SELECT COUNT(*) AS total_employees FROM Employees;
```

5. WHERE EQUAL: Retrieve all employees from department 3.

```
SELECT name, department_id FROM Employees WHERE department_id = 3;
```

6. WHERE GREATER THAN: Retrieve employees whose salary is over 60000.

```
SELECT name, salary FROM Employees WHERE salary > 60000;
```

7. WHERE NOT EQUAL: Retrieve employees who are NOT in department 1.

```
SELECT name, department_id FROM Employees WHERE department_id <> 1;
```

8. WHERE AND: Retrieve employees with salary > 50000 AND who are in department 2.

SELECT name FROM Employees WHERE salary > 50000 AND department_id = 2;

9. WHERE OR: Retrieve employees in department 1 OR department 4.

SELECT name FROM Employees WHERE department_id = 1 OR department_id = 4;

10. WHERE IS NULL: Find employees whose manager ID is not yet assigned.

SELECT name FROM Employees WHERE manager_id IS NULL;

11. WHERE BETWEEN: Retrieve employees hired between the two specified dates (inclusive).

SELECT name, hire_date FROM Employees WHERE hire_date BETWEEN '2023-01-01' AND '2023-12-31';

12. WHERE IN: Retrieve employees who are in a list of specific department IDs.

SELECT name FROM Employees WHERE department_id IN (1, 3, 5);

13. WHERE LIKE (WILDCARD %): Find employees whose names start with 'A'.

SELECT name FROM Employees WHERE name LIKE 'A%';

14. ORDER BY ASCENDING: Retrieve names, ordered alphabetically (A-Z).

SELECT name FROM Employees ORDER BY name ASC;

15. ORDER BY DESCENDING: Retrieve salaries, ordered from highest to lowest.

SELECT name, salary FROM Employees ORDER BY salary DESC;

16. LIMIT/TOP: Retrieve the top 5 highest-paid employees (uses LIMIT, common in MySQL/PostgreSQL).

SELECT name, salary FROM Employees ORDER BY salary DESC LIMIT 5;

17. SUM: Calculate the total salary expenditure across all employees.

SELECT SUM(salary) AS total_payroll FROM Employees;

18. AVG: Calculate the average salary of all employees.

SELECT AVG(salary) AS avg_salary FROM Employees;

19. MAX: Find the maximum salary.

SELECT MAX(salary) AS max_salary FROM Employees;

20. MIN: Find the minimum salary.

SELECT MIN(salary) AS min_salary FROM Employees;

21. GROUP BY: Group employees by department and count the number of employees in each.

SELECT department_id, COUNT(*) AS employee_count

FROM Employees

GROUP BY department_id;

22. HAVING: Find departments that have an average salary greater than 70000.

SELECT department_id

FROM Employees

GROUP BY department_id

HAVING AVG(salary) > 70000;

23. INNER JOIN (Simple): Combine Employees and Departments records where the IDs match.

```
SELECT E.name, D.dept_name
FROM Employees E
INNER JOIN Departments D
  ON E.department_id = D.department_id;
```

24. LEFT JOIN (Simple): Retrieve ALL departments, including those with no employees.

```
SELECT D.dept_name, E.name
FROM Departments D
LEFT JOIN Employees E
  ON D.department_id = E.department_id;
```

25. INSERT: Add a new employee record.

```
INSERT INTO Employees (employee_id, name, department_id, salary)
VALUES (101, 'Jane Doe', 3, 75000);
```

26. UPDATE: Change the salary of a specific employee (Jane Doe) to 80000.

```
UPDATE Employees
SET salary = 80000
WHERE name = 'Jane Doe';
```

27. DELETE: Remove an employee record based on their ID.

```
DELETE FROM Employees
WHERE employee_id = 101;
```

28. CREATE TABLE (DDL): Define a new table structure.

```
CREATE TABLE Projects (
    project_id INT PRIMARY KEY,
    project_name VARCHAR(100) NOT NULL,
    start_date DATE
);
```