### Where Clause

1. Retrieve all employees whose age is greater than 30 from the employees table.

2. Find all orders from the orders table where the total amount is less than 500.

---

### Like

3. Fetch all products from the products table where the product name starts with 'A'.

4. Retrieve all customers from the customers table whose email ends with '@gmail.com'.

---

### Between

5. Find all orders from the orders table where the order date is between '2025-01-01' and '2025-01-31'.

6. Retrieve all employees whose salary is between 40,000 and 60,000 from the employees table.

---

### Is Null / Is Not Null

7. List all customers from the customers table where the phone number is NULL.

8. Find all records in the orders table where the shipping address is NOT NULL.

---

### Aggregate Functions

9. Calculate the total salary of all employees in the employees table.

10. Find the maximum, minimum, and average price of products in the products table.

11. Count the total number of orders placed in the orders table.

---

### Alter Table

12. Add a new column date\_of\_birth to the employees table.

13. Modify the data type of the phone column in the customers table to VARCHAR(15).

---

### Update Set

14. Update the salary of all employees in the employees table by 10%.

15. Set the status of all orders in the orders table to 'completed' where the total amount is greater than 1000.

---

### Limit

16. Retrieve the top 5 highest-paid employees from the employees table.

17. Fetch the first 10 records from the products table.

---

### Order By

18. List all customers in the customers table sorted by their last name in ascending order.

19. Fetch all products from the products table ordered by price in descending order.

---

### Alias

20. Retrieve all employees with an alias for the full\_name column as "Employee Name".

21. Display all products with an alias for the price column as "Product Cost".

---

### Joins

22. Fetch all orders along with the customer names from the orders and customers tables using an INNER JOIN.

23. List all employees and their respective department names using a LEFT JOIN on the employees and departments tables.

---

### Group By

24. Group all orders in the orders table by customer ID and calculate the total order amount for each customer.

25. Find the number of employees in each department from the employees table grouped by department.

---

### Having

26. Retrieve all customers who have placed more than 5 orders from the orders table (use GROUP BY and HAVING).

27. Fetch departments from the departments table where the average salary of employees is greater than 50,000.

---

### Constraints

28. Create a table users with constraints for primary key on user\_id and unique constraint on email.

29. Add a foreign key constraint to the orders table referencing the customers table.

---

### Union / Union All

30. Retrieve all distinct product names from the products and backup\_products tables using UNION.

31. Fetch all product names (including duplicates) from the products and backup\_products tables using UNION ALL.

### ****SQL Topics****

#### 1. ****SQL INDEX****

* **Problem 1**: Create an index on the email column of a Users table to improve query performance when searching for a specific email. Verify the query execution time before and after creating the index.
* **Problem 2**: Create a composite index on the Orders table for the customer\_id and order\_date columns. Use it to optimize queries fetching orders for a customer within a specific date range.

#### 2. ****SQL STORED PROCEDURE****

* **Problem 1**: Write a stored procedure GetEmployeeDetails that accepts an employee\_id as input and returns the employee's name, department, and salary.
* **Problem 2**: Create a stored procedure UpdateStock that updates the stock quantity in a Products table after a sale. Pass product\_id and quantity\_sold as input parameters.

#### 3. ****CASE STATEMENT****

* **Problem 1**: Write a query to categorize employees from an Employees table based on their salary:
  + < 3000: "Low Income"
  + 3000-7000: "Middle Income"
  + > 7000: "High Income".
* **Problem 2**: Use a CASE statement to calculate a dynamic discount for a Sales table based on sale\_amount:
  + > 1000: 10%
  + 500-1000: 5%
  + < 500: No discount.

#### 4. ****STRING FUNCTIONS****

* **Problem 1**: Write a query to extract the domain name from email addresses in a Users table.
* **Problem 2**: Use string functions to capitalize the first letter of each word in a product\_name column from a Products table.

#### 5. ****SUBQUERY****

* **Problem 1**: Write a query using a subquery to find the names of employees who earn more than the average salary in their department.
* **Problem 2**: Use a subquery to list all customers who have placed more than three orders.

#### 6. ****MASKING****

* **Problem 1**: Mask the first six digits of credit card numbers in a Transactions table with X while displaying the last four digits.
* **Problem 2**: Write a query to replace the email addresses in a Users table with \*\*\*\*\*@domain.com while keeping the domain name intact.

#### 7. ****DATE FUNCTION****

* **Problem 1**: Write a query to find all orders placed in the last 30 days from an Orders table.
* **Problem 2**: Use date functions to calculate the age of customers from their date\_of\_birth column in a Customers table.

### ****PL/SQL Topics****

#### 1. ****STORED PROCEDURE****

* **Problem 1**: Create a stored procedure CalculateBonus that calculates the annual bonus for employees based on their salary and years of service.
* **Problem 2**: Write a stored procedure DeleteInactiveUsers to delete records from a Users table where the last\_login date is more than a year ago.

#### 2. ****FUNCTIONS****

* **Problem 1**: Write a PL/SQL function GetProductPrice that takes a product\_id as input and returns the price of the product from the Products table.
* **Problem 2**: Create a function CalculateAge that calculates the age of a person given their date\_of\_birth.

#### 3. ****CURSOR****

* **Problem 1**: Write a PL/SQL block using an explicit cursor to fetch and display employee names and salaries from an Employees table where the salary is greater than a specified amount.
* **Problem 2**: Create a cursor to iterate through all orders in an Orders table and update the status of orders placed more than 30 days ago to "Archived".

#### 4. ****TRIGGERS****

* **Problem 1**: Create a trigger that automatically updates the last\_updated column of a Products table whenever a product’s details are modified.
* **Problem 2**: Write a trigger to prevent deletion of rows from the Employees table if the employee has pending tasks in the Tasks table.