

1. Basic SQL Queries

1. Select All Data

Query to retrieve all columns from the **users** table.

```
1 SELECT *
2 FROM users;
```

2. Select Specific Column

Retrieve only the **first_name** column from the **customers** table.

```
1 SELECT first_name
2 FROM customers;
```

3. Filter Records by Condition

Fetch all users where the first name is "john" from the **user_details** table.

```
1 SELECT *
2 FROM user_details
3 WHERE first_name = 'john';
```

4. Sort Records

Get all product names sorted by price in descending order from the products table.

```
1 SELECT product_name
2 FROM products
3 ORDER BY price DESC;
```

5. Filter Data by Range

Retrieve all employees whose age is between 25 and 40 from the **employees** table.

```
1 SELECT *
2 FROM employees
3 WHERE age BETWEEN 25 AND 40;
```

6. Find Null Values

Select users from the **customers** table who don't have an email address.

```
1 SELECT *
2 FROM customers
3 WHERE email IS NULL;
```

7. Remove Duplicate Entries

Fetch unique **departments** from the departments table.

```
1 SELECT DISTINCT department
2 FROM departments;
```

4. Joining Tables

16. Inner Join

Retrieve employee names along with their department names.

```
1 SELECT e.first_name, d.department_name
2 FROM employees e
3 INNER JOIN departments d
4 ON e.department_id = d.department_id;
```

17. Left Join

Get all employees and their assigned projects, even if some employees don't have projects.

```
1 SELECT e.first_name, p.project_name
2 FROM employees e
3 LEFT JOIN projects p
4 ON e.employee_id = p.employee_id;
```

18. Right Join

List all projects and the employees assigned to them, including projects without employees.

```
1 SELECT p.project_name, e.first_name
2 FROM projects p
3 RIGHT JOIN employees e
4 ON p.project_id = e.project_id;
```

19. Full Outer Join

Retrieve all employees and all projects, including those with no matching records.

```
1 SELECT e.first_name, p.project_name
2 FROM employees e
3 FULL OUTER JOIN projects p
4 ON e.employee_id = p.employee_id;
```

6. Window Functions

23. Rank Over Partition

Rank products based on sales within each category.

```
1 SELECT product_id, category_id, sales_amount,
2        RANK() OVER (PARTITION BY category_id ORDER BY
3        sales_amount DESC) AS rank
4 FROM products;
```

24. Running Total

Calculate the cumulative sales for each product over time.

```
1 SELECT product_id, sale_date, sales_amount,
2        SUM(sales_amount) OVER (ORDER BY sale_date) AS
3        running_total
4 FROM sales;
```

25. Lag Function

Find the previous order amount for each user.

```
1 SELECT user_id, order_date, order_amount,
2        LAG(order_amount, 1) OVER (PARTITION BY user_id
3        ORDER BY order_date) AS previous_order
4 FROM orders;
```

8. Date and Time Functions

30. Current Date

Retrieve all users who joined today.

```
1 SELECT *
2 FROM users
3 WHERE join_date = CURRENT_DATE;
```

31. Extract Year

Get the year from the **join_date** column.

```
1 SELECT EXTRACT(YEAR FROM join_date)
2 FROM users;
```

32. Date Difference

Calculate the number of days between **order_date** and today.

```
1 SELECT age(CURRENT_DATE, order_date)
2 FROM orders;
```

33. Add Date Interval

Find all users who signed up in the last 30 days.

```
1 SELECT *
2 FROM users
3 WHERE join_date > CURRENT_DATE - INTERVAL '30 days';
```

2. Aggregation Functions

8. Count Records

Count the total number of orders from the **orders** table.

```
1 SELECT COUNT(*)
2 FROM orders;
```

9. Sum Column Values

Calculate the total sales from the **sales** table.

```
1 SELECT SUM(sale_amount)
2 FROM sales;
```

10. Find Maximum Value

Retrieve the highest salary from the **salaries** table.

```
1 SELECT MAX(salary)
2 FROM salaries;
```

11. Find Minimum Value

Retrieve the lowest score from the **exam_scores** table.

```
1 SELECT MIN(score)
2 FROM exam_scores;
```

12. Calculate Average

Get the average order amount from the **orders** table.

```
1 SELECT AVG(order_amount)
2 FROM orders;
```

3. Grouping

13. Group by and Count

Count the number of users per country from the **users** table.

```
1 SELECT country, COUNT(*)
2 FROM users
3 GROUP BY country;
```

14. Group by and Sum

Calculate the total sales per product category from the **product_sales** table.

```
1 SELECT category, SUM(sales)
2 FROM product_sales
3 GROUP BY category;
```

15. Group by and Filter (HAVING)

Get departments where the total salary is greater than \$1,000,000.

```
1 SELECT department, SUM(salary)
2 FROM salaries
3 GROUP BY department
4 HAVING SUM(salary) > 1000000;
```

5. Subqueries

20. Subquery in WHERE

Fetch all products with a price greater than the average price.

```
1 SELECT *
2 FROM products
3 WHERE price > (SELECT AVG(price) FROM products);
```

21. Subquery in SELECT

Retrieve employee names and their corresponding department names using a subquery.

```
1 SELECT first_name,
2        (SELECT department_name
3         FROM departments
4         WHERE department_id = employees.department_id) AS
5        department
6 FROM employees;
```

22. Subquery in FROM

Find the total sales per product using a subquery.

```
1 SELECT department, SUM(salary)
2 FROM salaries
3 GROUP BY department
4 HAVING SUM(salary) > 1000000;
```

7. String Functions

26. Concatenate Strings

Combine first name and last name to display full name.

```
1 SELECT CONCAT(first_name, ' ', last_name) AS full_name
2 FROM users;
```

27. String Length

Get the length of the product name.

```
1 SELECT LENGTH(product_name)
2 FROM products;
```

28. Substring

Extract the first 3 characters of the product name.

```
1 SELECT SUBSTRING(product_name, 1, 3)
2 FROM products;
```

29. Upper and Lower Case

Display customer names in uppercase.

```
1 SELECT UPPER(customer_name)
2 FROM customers;
```

9. Advanced SQL

34. Case Statements

Display gender as 'M' or 'F' based on the **gender** column.

```
1 SELECT first_name, last_name,
2        CASE
3          WHEN gender = 'male' THEN 'M'
4          WHEN gender = 'female' THEN 'F'
5          END AS gender_abbr
6 FROM users;
```

35. Coalesce Function

Replace null values in the **address** column with 'unknown'.

```
1 SELECT COALESCE(address, 'unknown')
2 FROM customers;
```

36. Casting and Conversion

Convert **order_amount** to a decimal value with 2 decimal places.

```
1 SELECT CAST(order_amount AS DECIMAL(10, 2))
2 FROM orders;
```

37. Recursive Query

Generate a sequence of numbers from 1 to 10.

```
1 WITH numbers AS (
2   SELECT 1 AS number
3   UNION ALL
4   SELECT number + 1
5   FROM numbers
6   WHERE number < 10
7 )
8 SELECT number
9 FROM numbers;
```

38. Cross Join Example

Retrieve a combination of every employee with every project.

```
1 SELECT e.first_name, p.project_name
2 FROM employees e
3 CROSS JOIN projects p;
```

39. Cross Join with Filter

Perform a cross join between employees and projects, but only show combinations where the employee's department matches the project's department.

```
1 SELECT e.first_name, p.project_name
2 FROM employees e
3 CROSS JOIN projects p
4 WHERE e.department_id = p.department_id;
```