Here’s a set of \*\*50 questions\*\* covering a wide variety of SQL concepts you can practice using your `customer`, `product`, and `sales` tables. These questions will help you build your portfolio project while also improving your querying skills:

---

### \*\*1. Basic SELECT Statements\*\*

1. Select all columns from the `customer` table.

2. Retrieve only the `customer\_name`, `age`, and `region` from the `customer` table.

---

### \*\*2. DISTINCT\*\*

3. List all unique `ship\_modes` from the `sales` table.

4. Find the distinct `categories` from the `product` table.

---

### \*\*3. WHERE Conditions\*\*

5. Retrieve customers whose `age` is greater than 30.

6. Find all orders where `sales` are greater than 500 and `discount` is less than 10%.

---

### \*\*4. IN and BETWEEN\*\*

7. Retrieve customers from specific `regions` (e.g., 'East', 'West') using the `IN` condition.

8. List products where the `profit` is between 100 and 1000.

---

### \*\*5. LIKE Condition\*\*

9. Find customers whose `customer\_name` starts with the letter 'A'.

10. Retrieve products whose `sub\_category` contains the word "chair."

---

### \*\*6. ORDER BY and LIMIT\*\*

11. Retrieve the top 5 most expensive sales (`sales` column).

12. Retrieve the youngest 10 customers (`age`) in ascending order.

---

### \*\*7. Aliasing\*\*

13. Rename `sales` as `Total\_Sales` and `profit` as `Net\_Profit` in the output.

14. Show `customer\_name` as `Name` and `region` as `Location`.

---

### \*\*8. Aggregate Functions\*\*

15. Find the total sales made.

16. Calculate the average profit per order.

17. Count the total number of customers.

---

### \*\*9. GROUP BY and HAVING\*\*

18. Group products by their `category` and calculate the total sales for each category.

19. Find all `regions` with total `sales` exceeding 10,000.

---

### \*\*10. CASE Statements\*\*

20. Create a new column that categorizes `profit` into "High Profit" (above 500) and "Low Profit" (below 500).

21. Add a column that determines if an order is "Discounted" or "No Discount" based on whether `discount` is greater than 0.

---

### \*\*11. Joins\*\*

22. Retrieve all orders along with the corresponding `customer\_name`.

23. Find all sales along with the `product\_name` for each `product\_id`.

24. List all orders along with the customer and product details.

---

### \*\*12. Subqueries\*\*

25. Find the `product\_name` of the most sold product (based on `quantity`).

26. Retrieve the names of customers who placed orders with a `profit` above the average profit.

---

### \*\*13. Creating Views\*\*

27. Create a view that shows `customer\_name`, `region`, and total `sales` for each customer.

28. Create a view to display `category`, `sub\_category`, and average `profit`.

---

### \*\*14. String Functions\*\*

29. Extract the first three letters of `customer\_name`.

30. Convert `category` to uppercase in the result.

31. Find the length of the `product\_name`.

---

### \*\*15. Date and Time Functions\*\*

32. Extract the year from the `order\_date`.

33. Calculate the number of days between `order\_date` and `ship\_date`.

34. Find all orders placed in the last 30 days.

---

### \*\*16. Window Functions\*\*

35. Rank all customers based on their total sales using `RANK()`.

36. Use `ROW\_NUMBER()` to assign a unique rank to orders based on their `sales` in descending order.

37. Calculate the cumulative sum of `sales` for each region.

---

### \*\*17. Data Type Conversions\*\*

38. Convert the `sales` column into a string and concatenate it with " USD".

39. Round the `profit` column to 2 decimal places.

---

### \*\*18. Altering Tables\*\*

40. Add a new column to the `customer` table to store their phone numbers.

41. Modify the `sales` table to set `sales` as NOT NULL.

---

### \*\*19. Adding Constraints\*\*

42. Add a primary key constraint to the `customer\_id` column in the `customer` table.

43. Create a foreign key constraint between `sales.customer\_id` and `customer.customer\_id`.

---

### \*\*20. Indexing\*\*

44. Create an index on the `order\_date` column in the `sales` table.

45. Add a composite index on `region` and `sales` in the `customer` table.

---

### \*\*21. Other Advanced Queries\*\*

46. Retrieve all customers who haven’t placed any orders.

47. Find products that haven’t been sold at all.

48. Calculate the profit margin for each product (`profit`/`sales`).

49. Write a query to show the total profit for each product sub-category in descending order.

50. Write a query to find the customer who generated the highest revenue.

---

### Tips for Practice:

1. Write the query first and validate it by running it on your database.

2. After writing the queries, document the question and the SQL solution as part of your portfolio.

3. Try to explain each query in plain language as if presenting it to someone without SQL knowledge.

If you need help with specific questions, let me know!

Here are \*\*25 more advanced SQL queries\*\* based on your database schema to further improve your skills in SQL querying:

---

### \*\*Advanced Joins and Subqueries\*\*

1. Find the `customer\_name` of customers who have purchased all products in the `category = "Furniture"`.

2. List customers who have placed more than 5 orders, along with their total `sales` and `profit`.

3. Identify products that have been ordered by customers from multiple `regions`.

4. Write a query to find the top 3 most profitable products (`product\_id`) for each `category`.

5. Retrieve customers who have spent more than the average `sales` in their region.

---

### \*\*Nested Subqueries\*\*

6. Retrieve the `product\_name` of the second-highest selling product based on `quantity`.

7. Find the `customer\_name` of customers who have placed the least number of orders.

8. Find customers whose total `profit` is less than the average `profit` across all customers.

9. Write a query to find the `product\_name` of products that have never been purchased.

10. List customers who have placed an order only in 2023 but not in any other year.

---

### \*\*Advanced Aggregate Functions\*\*

11. Calculate the contribution of each `sub\_category` to the total sales for its `category`.

12. For each `region`, calculate the percentage of total `sales` contributed by each `customer`.

13. Find the `category` with the highest average `profit` per order.

14. Determine the average `discount` per product across all orders.

15. Calculate the moving average of `sales` over the last 5 orders for each `customer\_id`.

---

### \*\*Window Functions\*\*

16. Rank customers by their total `profit` within each `region`.

17. Compute the running total of `sales` for each `product\_id` in descending order of `order\_date`.

18. Calculate the difference in `profit` between consecutive orders for each customer using `LAG()`.

19. Identify the first and last products purchased by each customer using `FIRST\_VALUE()` and `LAST\_VALUE()`.

20. Use `NTILE()` to divide customers into 4 groups based on their total `sales`.

---

### \*\*Analytical and Conditional Queries\*\*

21. Write a query to classify customers into "High Spenders" and "Low Spenders" based on whether their total `sales` exceed the median `sales`.

22. Identify the `ship\_mode` that has the shortest average delivery time (`ship\_date - order\_date`).

23. Determine which `region` consistently generates the highest `profit` over the years.

24. Write a query to analyze the seasonal trend in `sales` by calculating the monthly total `sales`.

25. Find products with a `sales`-to-`quantity` ratio greater than the average ratio across all products.

---

### \*\*Portfolio Use\*\*

Each of these queries represents a real-world use case, so make sure to document:

1. \*\*Business Problem:\*\* The scenario/question being answered.

2. \*\*Query:\*\* The SQL code.

3. \*\*Insight:\*\* The result of the query and its interpretation.

Feel free to share if you need help implementing any of these! 😊