Amazon Sales Analysis SQL Problems

1. Basic Select: Retrieve the names of all products in the products table.

2. Simple Join: Write a query to find the full name of customers (first_name + last_name) and the names of the products they ordered. Use a JOIN between customers, orders, and order_items.

3. Conditional Select: List all products with a price greater than 100. Display the product name and price.

4. Inner Join: List all orders along with customer names and product names. Use INNER JOIN between orders, customers, and order_items.

5. Left Join: Retrieve all customers and their corresponding orders. Include customers who haven't placed any orders.

6. Right Join: Retrieve all orders and their corresponding customers. Include orders without customer information.

7. Join with Filtering: List all products sold by sellers originating from 'USA.' Include product names and seller names.

8. Multi-table Join: Write a query to find the total amount paid for each order. Include the orders, order_items, and payments tables.

9. Join with Subquery: List the customers who have ordered products in the 'electronics' category. Use a subquery to find the category ID.

10. Cross Join: Write a query to list all combinations of category and sellers.

11. Count Function: Count the total number of unique customers in the customers table.

12. Sum and Group By: Find the total revenue generated by each seller. Display the seller name and total revenue.

13. Average Function: Calculate the average price of products in the products table.

14. Group By with Having: List all sellers who have sold more than 500 products. Display seller names and total products sold.

15. Group By Multiple Columns: Find the total revenue generated by each seller for each category. Display seller names, category names, and total revenue.

16. Count and Distinct: Find the total number of distinct products sold in each category.

17. Join with Aggregation: Write a query to find the total number of orders and the total revenue generated for each customer.

18. Aggregate Functions and CASE: Find the number of orders for each order status ('Inprogress,' 'Delivered,' etc.). Use CASE to categorize the statuses.

19. Nested Aggregation: Find the category with the highest total revenue.

20. Conditional Aggregation: Count the number of successful and failed payments for each customer.

21. Simple Subquery: Find the product with the highest price. Use a subquery to get the maximum price.

22. Correlated Subquery: Find all products whose price is above the average price in their category.

23. Subquery in WHERE Clause: Retrieve the names of customers who have ordered at least one product in the 'Pet Supplies' category.

24. Subquery in SELECT Clause: For each product, display its name and the total number of times it has been ordered.

25. Subquery with EXISTS: List all customers who have made at least one order.

26. IN Clause with Subquery: Find the names of sellers who have sold 'Apple' products.

27. NOT IN Clause: List all customers who have not placed any orders.

28. Subquery with JOIN: Find the names of products that are out of stock. Use a subquery to get product IDs with stock = 0 in the inventory table.

29. Subquery with HAVING: Retrieve sellers who have an average selling price of their products greater than 300.

30. Multi-level Subqueries: Find the product that has generated the highest revenue. Use nested subqueries to calculate revenue.

31. RANK() Function: For each category, rank the products based on their total sales amount.

32. DENSE_RANK() Function: List the top 5 customers based on the total amount spent. Use DENSE_RANK().

33. ROW_NUMBER() Function: Assign a row number to each product in the products table, ordered by price descending.

34. NTILE() Function: Divide all customers into 4 quartiles based on the total amount they have spent.

35. OVER Clause: For each order, calculate the running total of sales for the corresponding customer.

36. PARTITION BY Clause: Find the total revenue generated by each seller in each year.

37. LEAD() Function: For each product, find the next higher-priced product in the same category.

38. LAG() Function: For each product, find the previous lower-priced product in the same category.

39. Cumulative Sum: Calculate the cumulative sum of sales for each seller.

40. Window Function with Aggregation: Find the average order amount for each customer and compare it with their individual orders.

41. Date Filtering: List all orders placed in the current month. Include order ID, order date, and customer name.

42. Extract and Group By: Find the number of orders placed in each year. Use the EXTRACT() function to group by year.

43. DATEDIFF Function: Calculate the average delivery time for all delivered orders.

44. DATE_TRUNC Function: Find the total sales amount for each month in the current year.

45. Age Function: Find customers who have not placed any orders in the last 6 months.

46. Date Conversion: Convert the order_date to a different format (e.g., 'YYYY-MM-DD') and display it with the order ID.

47. Date Arithmetic: Calculate the total number of days between the order date and shipping date for each order.

48. Current Date Usage: Find all orders that are overdue for payment. Assume payment is due within 30 days of the order date.

49. Weekend Orders: Retrieve all orders that were placed on weekends.

50. Next Day Delivery: List all orders that were delivered the next day after shipping.