RETAIL DATA

Total: 55 Questions

EASY:

1. Retrieve all columns from the Sales table.

2. Retrieve the product_name and unit_price from the Products table.

3. Retrieve the sale_id and sale_date from the Sales table.

4. Filter the Sales table to show only sales with a total_price greater than $100.

5. Filter the Products table to show only products in the 'Electronics' category.

6. Retrieve the sale_id and total_price from the Sales table for sales made on January 3, 2024.

7. Retrieve the product_id and product_name from the Products table for products with a unit_price greater than $100.

8. Calculate the total revenue generated from all sales in the Sales table.

9. Calculate the average unit_price of products in the Products table.

10. Calculate the total quantity_sold from the Sales table.

11. Count Sales Per Day from the Sales table

12. Retrieve product_name and unit_price from the Products table with the Highest Unit Price

13. Retrieve the sale_id, product_id, and total_price from the Sales table for sales with a quantity_sold greater than 4.

14. Retrieve the product_name and unit_price from the Products table, ordering the results by unit_price in descending order.

15. Retrieve the total_price of all sales, rounding the values to two decimal places.

16. Calculate the average total_price of sales in the Sales table.

17. Retrieve the sale_id and sale_date from the Sales table, formatting the sale_date as 'YYYY-MM-DD'.

18. Calculate the total revenue generated from sales of products in the 'Electronics' category.

19. Retrieve the product_name and unit_price from the Products table, filtering the unit_price to show only values between $20 and $600.

20. Retrieve the product_name and category from the Products table, ordering the results by category in ascending order.

INTERMEDIATE:

1. Calculate the total quantity_sold of products in the 'Electronics' category.

2. Retrieve the product_name and total_price from the Sales table, calculating the total_price as quantity_sold multiplied by unit_price.

3. Identify the Most Frequently Sold Product from Sales table

4. Find the Products Not Sold from Products table

5. Calculate the total revenue generated from sales for each product category.

6. Find the product category with the highest average unit price.

7. Identify products with total sales exceeding 30.

8. Count the number of sales made in each month.

9. Retrieve Sales Details for Products with 'Smart' in Their Name

10. Determine the average quantity sold for products with a unit price greater than $100.

11. Retrieve the product name and total sales revenue for each product.

12. List all sales along with the corresponding product names.

13. Retrieve the product name and total sales revenue for each product.

14. Rank products based on total sales revenue.

15. Calculate the running total revenue for each product category.

16. Categorize sales as "High", "Medium", or "Low" based on total price (e.g., > $200 is High, $100-$200 is Medium, < $100 is Low).

17. Identify sales where the quantity sold is greater than the average quantity sold.

18. Extract the month and year from the sale date and count the number of sales for each month.

19. Calculate the number of days between the current date and the sale date for each sale.

20. Identify sales made during weekdays versus weekends.

Advanced:

1. List the Top 3 Products by Revenue Contribution Percentage

RETAIL DATA

2. Write a query to create a view named Total_Sales that displays the total sales amount for each product along with their names and categories.

3. Retrieve the product details (name, category, unit price) for products that have a quantity sold greater than the average quantity sold across all products.

4. Explain the significance of indexing in SQL databases and provide an example scenario where indexing could significantly improve query performance in the given schema.

5. Add a foreign key constraint to the Sales table that references the product_id column in the Products table.

6. Create a view named Top_Products that lists the top 3 products based on the total quantity sold.

7. Implement a transaction that deducts the quantity sold from the Products table when a sale is made in the Sales table, ensuring that both operations are either committed or rolled back together.

8. Create a query that lists the product names along with their corresponding sales count.

9. Write a query to find all sales where the total price is greater than the average total price of all sales.

10. Analyze the performance implications of indexing the sale_date column in the Sales table, considering the types of queries commonly executed against this column.

11. Add a check constraint to the quantity_sold column in the Sales table to ensure that the quantity sold is always greater than zero.

12. Create a view named Product_Sales_Info that displays product details along with the total number of sales made for each product.

13. Develop a stored procedure named Update_Unit_Price that updates the unit price of a product in the Products table based on the provided product_id.

14. Implement a transaction that inserts a new product into the Products table and then adds a corresponding sale record into the Sales table, ensuring that both operations are either fully completed or fully rolled back.

15. Write a query that calculates the total revenue generated from each category of products for the year 2024.