Grouping & Multi-table Queries

Grouping Results, Summarizing and Aggregating Data, Counting results, Adding Results, Averaging Results, MAX & MIN functions, using HAVING clause with GROUP BY Statements, Implicit Versus Explicit Groups, Counting DISTINCT Values

Simple Joins/ Equi-Joins, Parent / child queries, Inner Joins, Multiple Joins, Cross Joins, Self Joins, Outer Joins, Right Joins, Left Joins, Full-outer Joins, Creating joins with more than two tables, Equi-Joins Versus Non-Equi Joins, Union operations

Quick Summary -

- Grouping and multi-table queries are advanced SQL concepts used to **combine and manipulate data from multiple tables**.
- **Grouping results** involves using the GROUP BY clause to group data by one or more columns, and then applying aggregate functions like SUM, AVG, MAX, MIN to calculate values based on those groups.
- The HAVING clause can be used with the GROUP BY clause **to filter groups based on specific condition**s.
- Implicit groups are created when no GROUP BY clause is used, and the query returns aggregate values for the entire table. Explicit groups are created when the GROUP BY clause is used. ● Joins are used to combine data from multiple tables based on a common column or set of columns. There are several types of joins, including inner joins, outer joins, and cross joins. ● Inner joins return only the rows that have matching values in both tables, while outer joins return all rows from one table and matching rows from the other table.
- Self joins are used when a table needs to be joined to itself, and multiple joins are used when more than two tables need to be joined.
- Non-equi joins are used to join tables based on non-equality conditions, while equi-joins are used to join tables based on equality conditions.
- Union operations are used to combine the results of two or more SELECT statements into a single result set.

Note: All the questions must be answered with the help of examples.

Questions:

Remembering (Easy):

1. What is the purpose of the HAVING clause in SQL?
2. What is the difference between an implicit and explicit group in SQL?
3. What is the MAX function in SQL, and how is it used?
4. How do you count the number of distinct values in a column using SQL?
5. What is the purpose of a simple join in SQL?

Solution:

1. The HAVING clause is used to filter the results of a GROUP BY query based on a specific condition that applies to an aggregated column. It is similar to the WHERE clause, but it operates on the results of an aggregation rather than on individual rows.

2. An **implicit** group is created when an aggregation function is used without a GROUP BY clause. In this case, the entire table is treated as a single group.

   On the other hand, an **explicit** group is created when a GROUP BY clause is used, which groups the table by one or more columns and creates separate groups for each unique combination of values in those columns.

3. The MAX function in SQL is used to return the maximum value of a specified column. It can be used in combination with the SELECT statement to retrieve the highest value from a particular column in a table. For example, the following query returns the highest salary from the employees table:

SELECT MAX(salary) FROM employees;

4. To count the number of distinct values in a column using SQL, you can use the COUNT() function with the DISTINCT keyword. For example, the following query returns the number of distinct values in the "city" column of the customers table:

SELECT COUNT(DISTINCT city) FROM customers;

5. A simple join in SQL is used to combine two or more tables based on a common column or columns. It allows you to retrieve data from multiple tables as a single result set. For example, the following query joins the orders and customers tables on the "customer_id" column to return a list of all orders with the corresponding customer information:

SELECT * FROM orders
JOIN customers ON orders.customer_id = customers.customer_id;
Understanding (Moderate):

1. How do you use the GROUP BY statement in SQL to group query results?
2. What is the purpose of summarizing and aggregating data in SQL, and how does it relate to GROUP BY?
3. How do you use the COUNT function in SQL to count query results?
4. What is the difference between an inner join and an outer join in SQL?
5. How do you use the UNION operator to combine query results in SQL?

Solutions:

1. The GROUP BY statement is used to group query results based on one or more columns. It is typically used with aggregate functions such as COUNT, SUM, AVG, MAX, and MIN to summarize

the data in each group. Here is an example:

```
SELECT department, AVG(salary)
FROM employees
GROUP BY department;
```

This query will group the employees by department and calculate the average salary for each department.

2. Summarizing and aggregating data in SQL allows you to analyze large datasets and extract meaningful insights. The GROUP BY statement is a key component of this process, as it allows you to group the data by one or more columns and apply aggregate functions to each group.

This can help you answer questions like "What is the average sales per region?" or "How many customers did we have in each state?"

3. The COUNT function is used to count the number of rows returned by a query. It can be used with the GROUP BY statement to count the number of rows in each group. Here is an example:

```
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

This query will count the number of employees in each department.

4. An inner join returns only the rows that have matching values in both tables being joined. An outer join, on the other hand, returns all rows from one table and matching rows from the other table.

There are three types of outer joins: left outer join, right outer join, and full outer join.
Here is an example of an inner join:

```
SELECT *
FROM orders
INNER JOIN customers
ON orders.customer_id = customers.customer_id;
```

This query will return only the orders that have a matching customer in the customers table.

Here is an example of a left outer join:

```
SELECT *
FROM orders
LEFT OUTER JOIN customers
ON orders.customer_id = customers.customer_id;
```

This query will return all orders, including those that do not have a matching customer in the customers

table.

5. The UNION operator is used to combine the results of two or more SELECT statements into a single result set. The SELECT statements must have the same number of columns and compatible data types. Here is an example:

```
SELECT name, address
FROM customers
WHERE state = 'CA'
UNION
SELECT name, address
FROM suppliers
WHERE state = 'CA';
```

This query will return the names and addresses of all customers and suppliers located in California. The results of the two SELECT statements will be combined into a single result set, with duplicates removed.

Applying (Moderate):

1. Create a SQL query that calculates the average value of a specific column for each distinct value of another column.
2. Write a SQL query that counts the number of rows in a table that meets a specific condition.
3. Create a SQL query that joins three tables using a parent-child relationship. 4. Write a SQL query that updates the values of a specific column in a table based on a specific condition.
5. Create a SQL query that returns the top 10 rows from a table ordered by a specific column in descending order.

Solution :

SQL query to calculate average value for each distinct value in another column:

```
SELECT column1, AVG(column2)
FROM table
GROUP BY column1;
```

Here, column1 is the column for which distinct values are to be found and column2 is the column whose average is to be calculated.

SQL query to count the number of rows in a table that meet a specific condition:

```
SELECT COUNT(*)
FROM table
WHERE condition;
```

Here, condition is the specific condition that needs to be met and * indicates that all rows are to be counted.

SQL query to join three tables using a parent-child relationship:

```
SELECT *
FROM parent_table
JOIN child_table1 ON parent_table.key = child_table1.foreign_key
JOIN child_table2 ON parent_table.key = child_table2.foreign_key;
```

Here, parent_table is the table that contains the primary key and child_table1 and child_table2 are tables that contain foreign keys to the parent table.

SQL query to update values of a specific column based on a specific condition:

```
UPDATE table
SET column = new_value
WHERE condition;
```

Here, column is the specific column that needs to be updated, new_value is the new value to be set and condition is the specific condition that needs to be met.

SQL query to return top 10 rows from a table ordered by a specific column in descending order:

```
SELECT *
FROM table
ORDER BY column DESC
LIMIT 10;
```

Here, column is the specific column that the results are to be ordered by in descending order and LIMIT 10 indicates that only the top 10 rows should be returned.

SCENARIO BASED CHALLENGE QUESTIONS: SELF PROBLEM-SOLVING TASK

01. **Scenario:** You have been asked to create a report that shows the total number of orders and the average order value for each customer in your company's database. However, the database is large and contains thousands of orders from hundreds of customers. How would you approach this task using SQL?

Challenge question: Write a SQL query that produces a report showing the total number of orders and the average order value for each customer in the database. In addition, sort the results in descending order by the total number of orders.

02.Scenario: You have been asked to update the price of a specific product in your company's database.

However, there are thousands of orders that have already been placed for this product, and updating the price could have a significant impact on the business. How would you approach this task using SQL?

Challenge question: Write a SQL query that updates the price of a specific product in the database. However, to ensure data integrity and avoid negative impacts on the business, the update should only apply to future orders, not past orders. In addition, the query should return the number of orders that will be affected by the price change.

—----------All theBest!!—---------