Part - 1

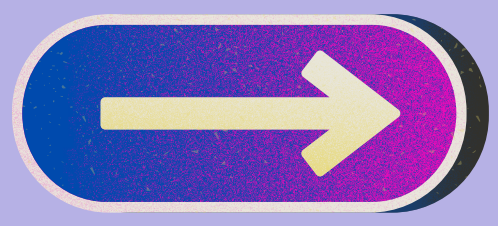
Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Explain different types of joins?

When you're working with databases, joins are essential for combining rows from two or more tables based on a related column.



1. INNER JOIN:

➡ What it Does;

- **Combines rows from both tables where there is a match in the specified columns.**
- **If there is no match, the row is not included in the result.**

➡ When to Use:

- **When you need to retrieve data that exists in both tables.**

➡ Example:

- **Find all employees who have a department assigned.**

```
SELECT employees.name, departments.department_name
FROM employees
INNER JOIN
departments ON
    employees.department_id = departments.id;
```

- **This returns employees only if they have a matching department.**

2. LEFT JOIN:

➡ What it Does;

- **Returns all rows from the left table and matched rows from the right table.**
- **If there's no match, NULL values are returned for columns from the right table.**

➡ When to Use:

- **When you need all records from the left table, regardless of whether there is a match in the right table.**

➡ Example:

- **List all employees and include department information if available.**

```
SELECT employees.name, departments.department_name
FROM employees
LEFT JOIN departments
    ON employees.department_id = departments.id;
```

- **This returns all employees, including those without a department.**

3. RIGHT JOIN:

➡ **What it Does;**

- **Returns all rows from the right table and matched rows from the left table.**
- **If there's no match, NULL values are returned for columns from the left table.**

➡ **When to Use:**

- **When you need all records from the right table, regardless of whether there is a match in the left table.**

➡ **Example:**

- **List all departments and include employee information if available.**

```sql
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
    ON employees.department_id = departments.id;
```

- **This returns all departments, including those without employees.**

4. FULL JOIN

➡ **What it Does;**

- **Returns rows when there is a match in either the left or right table.**
- **If there is no match, NULL values are returned for non-matching rows from both tables.**

➡ **When to Use:**

- **When you need a complete view of both tables, including all matched and unmatched rows.**

➡ **Example:**

- **Get a comprehensive list of all employees and departments, showing matches and unmatched records.**

```
SELECT employees.name, departments.department_name

FROM employees

FULL JOIN departments

    ON employees.department_id = departments.id;
```

- **This returns all employees and all departments, showing NULLs where there is no match**

5. CROSS JOIN:

➡ **What it Does;**

- **Returns the Cartesian product of the two tables,**
- **meaning it combines all rows from the left table with all rows from the right table.**

➡ **When to Use:**

- **When you need every possible combination of rows from the two tables, often for testing or generating combinations.**

➡ **Example:**

- **Generate a list of all possible employee-department pairings.**

```
SELECT employees.name, departments.department_name
FROM employees
CROSS JOIN departments;
```

- **This returns every possible combination of employees and departments.**

6. SELF JOIN:

➡ What it Does;

- **A self join is a regular join but the table is joined with itself.**

➡ When to Use:

- **When you need to compare rows within the same table, such as finding employees who are also managers.**

➡ Example:

- **Identify employees who report to the same manager within the employees table.**

```sql
SELECT A.name AS EmployeeName, B.name AS ManagerName
FROM employees A
JOIN employees B
    ON A.manager_id = B.id;
```

- **This returns employees along with their managers, assuming managers are also listed in the employees table.**

wanna see some

Counter Questions

1. How does the performance of different JOIN types compare?

➡ **Performance varies with JOIN types. INNER JOINs are typically faster than OUTER JOINs, as they involve fewer rows.**

CROSS JOINs can generate large result sets, impacting performance, especially with large tables.

2. How do NULL values affect JOIN operations?

➥ **In INNER JOINs, rows with NULL values in the join columns are excluded.**

In OUTER JOINs, NULL values are preserved, filling gaps where matches are missing between tables.

3. What is an Equi-Join and how is it different from other JOINs?

➡ **An Equi-Join is a join that uses the equality operator (=) to match rows between tables.**

It differs from Non-Equi-Joins, which use other comparison operators like <, >, etc., for matching rows.

4. Can you explain how indexing impacts JOIN performance?

➥ **Indexes can significantly improve JOIN performance by speeding up row lookups, particularly for large tables.**

Without indexes, the database must perform full table scans, which are much slower.

5. What are the risks of using a FULL JOIN on large datasets?

➡ **A FULL JOIN can result in a large number of rows when combining two large datasets, leading to increased memory usage and slower performance.**

Efficient indexing and query optimization become critical in such cases.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

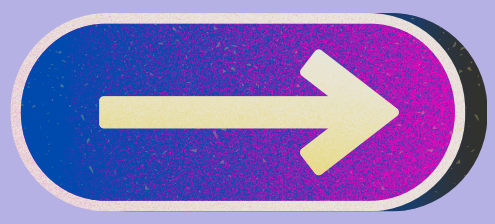
Part - 2

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Tell me about the use case of cross join



What is a CROSS JOIN?

- **A CROSS JOIN combines every row from the first table with every row from the second table, creating a Cartesian product.**

If one table has 5 rows and another has 3, the result will have 15 rows (5 x 3).

When should you use a CROSS JOIN?

Use of CROSS JOIN;

- **CROSS JOIN is used when you need to create all possible combinations between two sets of data.**

A common use case is generating combinations of product options like colors and sizes.

Can you explain it with an example ?

A real-world example

Suppose you work for a clothing company, and you have two tables: one with different shirt colors and another with different sizes.

You want to create a list of all possible combinations of shirt colors and sizes to ensure every possible product variant is available.

Color
Red
Blue
Green

Size
Samll
Medium
Large

What should be the result for this

Result;

Query:

```
SELECT colors.Color, sizes.Size
FROM ShirtColors AS colors
CROSS JOIN Sizes AS sizes;
```

Result:

Color	Size
Red	Small
Red	Mediam
Red	Large
Blue	Small
Blue	Mediam
Blue	Large
Green	Small
Green	Mediam
Green	Large

wanna see some

Counter Questions

1. Why is it important to consider all combinations in business scenarios?

➡ **By considering all combinations, businesses can ensure they don't miss any product variants or pairings,**

which is crucial for comprehensive inventory management, product planning, and marketing.

2. How does CROSS JOIN differ from INNER JOIN?

➡ **A CROSS JOIN combines every row from two tables without any condition, producing a Cartesian product.**

An INNER JOIN only combines rows where there is a match between the specified columns in both tables.

3. Can CROSS JOIN lead to performance issues?

➡ **Yes, since CROSS JOIN creates every possible combination of rows, it can lead to very large result sets.**

If both tables are large, the resulting data can be massive, potentially affecting performance.

4. Can you limit the results of a CROSS JOIN?

➡️ **Yes, you can limit the results of a CROSS JOIN using a WHERE clause to filter specific combinations or a LIMIT clause to restrict the number of rows returned.**

5. What are potential risks of using CROSS JOIN in production queries?

➥ **The main risk is creating unnecessarily large datasets if the tables are big, which can slow down the database and consume excessive resources.**

It’s crucial to use CROSS JOIN only when all combinations are genuinely needed.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

Part - 3

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Tell me about the use case of self-join

Q. Can a Self-Join result in duplicate data?

What is a SELF-JOIN?

➡ **A self-join is a regular SQL join where a table is joined with itself.**

This is used when comparing rows within the same table,

such as showing hierarchical relationships or pairing related data.

When should you use a SELF JOIN?

Use of SELF-JOIN;

➥ **You should use a self-join when you need to compare rows in the same table.**

A typical use case is finding relationships within the data,

- **like employees and**
- **their managers.**

Can you explain it with an example ?

A real-world example

Suppose you work in an organization where each employee reports to a manager,

and you want to create a list showing each employee along with their respective manager.

EmployeeID	EmployeeName	ManagerID
1	John	3
2	Sarah	5
3	Michael	NULL
4	Tom	1

What should be the result for this

Result;

Query:

```
SELECT e1.EmployeeName AS Employee,
e2.EmployeeName AS Manager
FROM  Employees e1
LEFTJOIN Employees e2
      ON e1.ManagerID = e2.EmployeeID;
```

Result:

Employee	Manager
John	Michael
Sarah	Michael
Michael	NULL
Tom	John

Explanation:

- **John and Sarah report to Michael.**
- **Michael has no manager (NULL).**
- **Tom reports to John.**

wanna see some

Counter Questions

1. Why is Self-Join useful?

➡ **Self-joins are useful when dealing with hierarchical data,**

like organizational structures or comparing data within the same table.

It allows you to connect related rows, such as finding managers and employees or comparing similar records.

2. What is the difference between Self-Join and INNER JOIN?

➡ **A self-join is when a table joins with itself, whereas an INNER JOIN connects two different tables based on matching values.**

Self-join is useful for comparing rows within a single table, while INNER JOIN compares rows between tables.

3. Can Self-Joins be used with multiple tables?

➡ **No, a self-join is strictly used to join a table with itself.**

If you need to join more than one table,

you'd use other joins like INNER JOIN or OUTER JOIN

4. How do you avoid performance issues with Self-Joins?

➡️ **To avoid performance issues, ensure that the columns you're joining on,**

such as EmployeeID and ManagerID, are properly indexed.

This speeds up the query execution by making lookups more efficient.

5. Is it possible to use other types of joins (e.g., LEFT JOIN) in a Self-Join?

➡ **Yes, you can use any type of join, like**

- **LEFT JOIN or**
- **RIGHT JOIN, in a self-join.**

For example, using LEFT JOIN ensures that all employees are listed, even if they don't have a manager.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

Part - 4

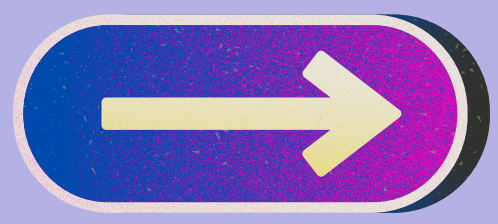
Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

What are aggregate functions in SQL?

Aggregate functions in SQL perform calculations on multiple rows of a table's column and return a single value. These functions are used to perform summary operations and are vital in data analysis and reporting.



1. COUNT():

Purpose: Counts the number of rows in a result set.

Example:

```
SELECT COUNT(*) AS TotalEmployees
FROM Employees;
```

Explanation: Counts the total number of employees in the Employees table. It can also count non-NULL values in a specific column.

Let's talk about SUM() Function

2. SUM():

Purpose: Adds up the values in a numeric column.

Example:

```
SELECT SUM(Salary) AS TotalSalary
FROM Employees;
```

Explanation: Calculates the total sum of all salaries in the **Employees** table. Useful for financial and numeric aggregations.

Let's talk about AVG() Function

3. AVG():

Purpose: Calculates the average value of a numeric column.

Example:

```
SELECT AVG(Salary) AS AverageSalary
FROM Employees;
```

Explanation: Computes the average salary of employees in the Employees table. Important for finding mean values

Let's talk about MIN() Function

4. MIN():

Purpose: **Finds the minimum value in a column.**

Example:

```
SELECT MIN(Salary) AS LowestSalary
FROM Employees;
```

Explanation: **Identifies the lowest salary in the Employees table. Helps in identifying the minimum values in datasets.**

Let's talk about MAX() Function

5. MAX():

Purpose: Finds the maximum value in a column.

Example:

```
SELECT MAX(Salary) AS HightestSalary
FROM Employees;
```

Explanation: Identifies the highest salary in the Employees table. Useful for finding maximum values in datasets.

How do Aggregate Functions work with GROUP BY?

Work with GROUP BY

➡ **Aggregate functions are often used with GROUP BY to calculate values for specific groups,**

such as:

- **counting employees by department or**
- **averaging sales by region.**

➡ **Example with GROUP BY:**

```
SELECT employees.name, departments.department_name
FROM employees
RIGHT JOIN departments
    ON employees.department_id = departments.id;
```

wanna see some

Counter Questions

1. Can aggregate functions be used with non-numeric data?

➡ **Yes, but only certain aggregate functions like COUNT() or MIN()/MAX() can be used on non-numeric data.**

For example, COUNT() can count the number of rows in a column containing strings.

2. What happens if there are NULL values in a column when using aggregate functions?

➥ **Aggregate functions like SUM(), AVG(), MIN(), and MAX() ignore NULL values by default.**

However, COUNT() can include or exclude NULL values depending on how it's used.

3. How is GROUP BY different from ORDER BY when using aggregate functions?

➡ **GROUP BY groups rows that share a value into summary rows, allowing aggregate functions to work on each group.**

ORDER BY simply sorts the result set based on specified columns but doesn’t affect aggregation.

4. Can we use multiple aggregate functions in a single query?

➡ **Yes, you can use multiple aggregate functions in one query.**

For example, you can calculate both SUM() and COUNT() in the same query to get the total salary and the number of employees.

5. What is the difference between COUNT(*) and COUNT(column_name)?

➥ **COUNT(*) counts all rows, including those with NULL values.**

COUNT(column_name) counts only non-NULL values in a specific column.

This distinction is important when you have incomplete data.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

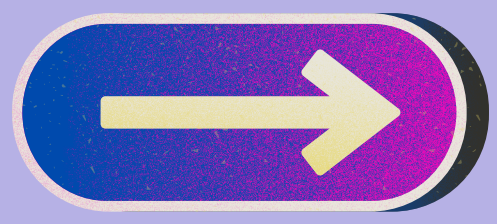
Part - 5

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

How do we use Case statement in SQL? Give example



What is a CASE statement in SQL?

➡ **The CASE statement in SQL is used to add conditional logic to a query.**

It checks each condition in sequence and returns a value based on the first condition that is true,

similar to an if-then-else statement in programming.

What happens if no conditions in a CASE statement are true?

If no conditions in a CASE statement are true

- **If no conditions are true, the CASE statement returns the value in the ELSE clause.**

If there is no ELSE clause, it returns NULL.

What is the syntax of a CASE statement in SQL?

Syntax:

```sql
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ELSE default_result
END
```

Can you give an example of using a CASE statement in SQL?

Example:

Let’s say you have a table of students and their scores. You want to assign a grade based on the score.

```sql
SELECT
    StudentID,
    Name,
    Score,
    CASE
        WHEN Score >= 90 THEN 'A'
        WHEN Score >= 80 THEN 'B'
        WHEN Score >= 70 THEN 'C'
        WHEN Score >= 60 THEN 'D'
        ELSE 'F'
    END AS Grade
FROM Students;
```

What does the output of the query look like?

5. MAX():

For example, if the Students table contains the following data:

StudentID	Name	Score
1	John	85
2	Jane	92
3	Jim	67
4	Jill	78

Then the output would be

Output:

StudentID	Name	Score	Grade
1	John	85	B
2	Jane	92	A
3	Jim	67	D
4	Jill	78	C

Summary:

- **The CASE statement adds conditional logic to SQL.**
- **It checks conditions in order and returns the matching result.**
- **If no conditions are met, it returns a default value or NULL.**

wanna see some

Counter Questions

1. Can we use CASE in the WHERE clause?

➡ **Yes, you can use a CASE statement in the WHERE clause to add conditional logic to filter rows.**

It allows you to perform more complex filtering based on conditions.

2. Can you nest CASE statements in SQL?

➡ **Yes, you can nest CASE statements within other CASE statements for more complex logic.**

However, it's important to manage readability when nesting multiple conditions.

3. What is the difference between CASE and IF in SQL?

➡ **CASE is more versatile than IF in SQL. IF is mostly used in procedural SQL (like stored procedures),**

while CASE is used within queries to add conditional logic directly in SELECT, WHERE, and other clauses.

4. Can a CASE statement return multiple columns?

➡ **No, a CASE statement can only return a single value (or column).**

However, you can write multiple CASE statements within a query to create different conditional columns.

5. How does CASE handle NULL values?

➥ **In a CASE statement, NULL can be part of the conditions,**

and you can use it to check for NULL values specifically.

If none of the conditions are met and there's no ELSE clause, the result will be NULL.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

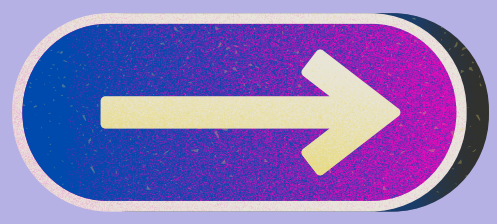
Part - 6

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Difference and Similarities between General Case Statement and Case Expression in Sql.?



What is the similarity between the General CASE Statement and the CASE Expression in SQL?

- **Both are used to add conditional logic in SQL. They evaluate conditions and return results based on those conditions.**

In essence, they help categorize data or derive new data columns based on the conditions.

What is the primary difference in between themn ?

The primary difference

- **General CASE Statement**: Does not return a value itself but controls the execution of other SQL statements or code blocks.

CASE Expression: Returns a single value and is used within a single SQL statement to evaluate a hardcoded expression.

What is the syntax difference between them ?

Syntax difference:

General CASE Statement

```
CASE
        WHEN condition1 THEN result1
        WHEN condition2 THEN result2
         ...
         ELSE default_result
END
```

CASE Expression

```
CASE Expression
        WHEN value1 THEN result1
        WHEN value2 THEN result2
        WHEN value3 THEN result3
         ...
         ELSE other_result
END;
```

Can you give an example of a General CASE Statement?

Example of a General CASE Statement:

```sql
SELECT customer_id, amount,
       CASE
         WHEN amount > 100 THEN 'Expensive product'
         WHEN amount = 100 THEN 'Moderate product'
         ELSE 'Inexpensive product'
       END AS ProductStatus
FROM payment;
```

- **This categorizes product prices based on conditions.**

Can you give an example of a CASE Expression?

Example of a CASE Expression:

```sql
SELECT customer_id,
       CASE amount
         WHEN 500 THEN 'Prime Customer'
         WHEN 100 THEN 'Plus Customer'
         ELSE 'Regular Customer'
       END AS CustomerStatus
FROM payment;
```

This categorizes customers based on the specific value of the amount field.

How does the General CASE Statement process conditions?

General CASE Statement process conditions

- **The General CASE Statement evaluates each condition sequentially, and once a condition is met,**

it returns the corresponding result and stops evaluating further conditions.

How does the CASE Expression differ in its approach?

CASE Expression differ in its approach:

- **The CASE Expression works by directly comparing a value,**

(like a specific column or expression)

with a series of hardcoded values and returning the result associated with the first match.

wanna see some

Counter Questions

1. Can CASE Expressions be nested inside each other, and what would be a real-world use case for doing so?

➡ **Yes, CASE Expressions can be nested. A real-world use case would be handling multiple complex conditions,**

such as assigning grades based on different criteria like attendance and scores.

2. What happens if a CASE Statement or CASE Expression does not have an ELSE clause, and none of the conditions are met?

➡ **if there is no ELSE clause and no condition is met,**

The result will be NULL.

3. Is there a performance difference between using a General CASE Statement versus a CASE Expression in large datasets?

➡ **There is generally no significant performance difference.**

Both work similarly, but CASE Expressions are typically used within a SELECT statement for returning values.

4. How can the CASE Statement be used in conjunction with the GROUP BY clause to categorize grouped data?

➡ **You can use the CASE Statement to categorize grouped data,**

like grouping sales data by region and using CASE to label regions as "High" or "Low" performing based on sales.

5. In what scenarios would you prefer to use a General CASE Statement over a CASE Expression, or vice versa?

➡ **Use a General CASE Statement when controlling multiple SQL operations, and**

a CASE Expression when you only need to return a value within a query.

you completed one interview question with me,

can you do me a favour

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

Part - 7

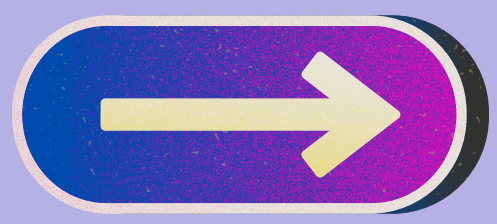
Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Find the total number of records in output when you join two tables,

assuming that there are duplicate key values.



Answer:

- **To determine the total number of records in the output when joining two tables with duplicate key values,**

  it is essential to understand the type of join being used and how it handles duplicates.

In next slide there is a quick overview of different types of joins and how they handle duplicate key values:

Are you excited to see the types of joins ?

Types of Joins:

- **INNER JOIN**
- **LEFT JOIN (or LEFT OUTER JOIN)**
- **RIGHT JOIN (or RIGHT OUTER JOIN)**
- **FULL JOIN (or FULL OUTER JOIN)**
- **CROSS JOIN**

Let's understand with Practical examples ?

Example:

Consider two tables, TableA and TableB

TableA:

ID	Name
1	Alice
2	Bob
2	Carol
3	Dave

Table B:

ID	Product
2	Leptop
2	Tablet
3	Phone
4	Monitor

Can you give an example of Inner Join?

Inner Join:

Query:

```
SELECT * FROM TableA a
INNER JOIN TableB b
    ON a.ID = b.ID;
```

Result:

ID	Name	Product
2	Bob	Laptop
2	Bob	Tablet
2	Carol	Laptop
2	Carol	Tablet
3	Dave	Phone

Explanation:

- The ID '2' appears twice in both TableA and TableB, resulting in 4 combinations (2 from TableA × 2 from TableB).
- The ID '3' appears once in both tables, resulting in 1 combination.

Total records: 4 (for ID '2') + 1 (for ID '3') = 5 records

LEFT JOIN Example:

Query:

```
SELECT * FROM TableA a
LEFT JOIN TableB b
    ON a.ID = b.ID;
```

Result:

ID	Name	Product
1	Alice	NUll
2	Bob	Laptop
2	Bob	Tablet
2	Carol	Laptop
2	Carol	Tablet
3	Dave	Phone

Explanation:

- The ID '1' appears in TableA but not in TableB, so it results in a row with NULL in the Product column.
- IDs '2' and '3' result in the same combinations as in the INNER JOIN example.

Total records: 1 (for ID '1') + 4 (for ID '2') + 1 (for ID '3')
= 6 records.

RIGHTJOIN Example:

Query:

```
SELECT * FROM TableA a
RIGHT JOIN TableB b
    ON a.ID = b.ID;
```

Result:

ID	Name	Product
2	Bob	Laptop
2	Bob	Tablet
2	Carol	Laptop
2	Carol	Tablet
3	Dave	Phone
4	NULL	Monitor

Explanation:

- The ID '4' appears in TableB but not in TableA, so it results in a row with NULL in the Name column.
- IDs '2' and '3' result in the same combinations as in the INNER JOIN example.

Total records: 4 (for ID '2') + 1 (for ID '3') + 1 (for ID '4')
= 6 records.

FULL JOIN Example:

Query:

```
SELECT * FROM TableA a
FULL JOIN TableB b
    ON a.ID = b.ID;
```

Result:

ID	Name	Product
1	Alice	NuLL
2	Bob	Laptop
2	Bob	Tablet
2	Carol	Laptop
2	Carpl	Tablet
3	Dabe	Phone
4	NuLL	Monitor

Explanation:

- **Includes all records from both tables.**
- **NULLs for non-matching rows in each table.**

Total records: 1 (for ID '1') + 4 (for ID '2') + 1 (for ID '3') + 1 (for ID '4') = 7 records.

CROSS JOIN Example:

Query:

```
SELECT * FROM TableA a
CROSS JOIN TableB b
```

Result:

ID	Name	ID	Product
1	Alice	2	Laptop
1	Alice	2	Tablet
1	Alice	3	Phone
1	Alice	4	Monitor
2	Bob	2	Laptop
2	Bob	2	Tablet
2	Bob	3	Phone
2	Bob	4	Monitor
2	Carol	2	Laptop
2	Carol	2	Tablet

CROSS JOIN Example:

2	Carol	3	Phone
2	Carol	4	Monitor
3	Dave	2	Laptop
3	Dave	2	Tablet
3	Dave	3	Phone
3	Dave	4	Monitor

Explanation:

- **Each row of TableA is combined with each row of TableB.**

Total records: 4 rows in TableA × 4 rows in TableB = 16 records.

Conclusion:

The total number of records in the output when joining two tables depends on the type of join and the presence of duplicate key values.

Understanding the join type is crucial to predict the number of resulting records.

wanna see some

Counter Questions

1. How does a LEFT JOIN differ from an INNER JOIN when there are no matching records in the second table?

➡ **In a LEFT JOIN, unmatched rows from the first table are included with NULL values for the second table's columns,**

while INNER JOIN only includes matching rows.

2. What happens when you perform a CROSS JOIN on two tables with no common keys?

➡ **A CROSS JOIN produces a Cartesian product,**

where every row from the first table is combined with every row from the second table.

3. How can you handle duplicate records resulting from an INNER JOIN?

➡ **You can use DISTINCT in the SELECT clause or implement specific filtering conditions to eliminate duplicates.**

4. In what scenarios would a FULL JOIN be more useful than a LEFT or RIGHT JOIN?

➡ **A FULL JOIN is useful when you need to retrieve all records from both tables,**

regardless of matching conditions, and include NULL for non-matching rows.

5. Why might the number of records increase significantly when using a JOIN with duplicate key values?

➡ **Duplicate key values multiply combinations between tables,**

as each matching row in one table is paired with every matching row in the other.

you completed one interview question with me,

can you do me a favour

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

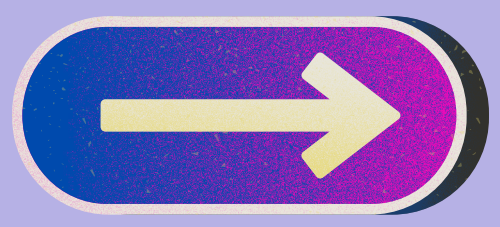
Part - 8

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

What are the rules to follow when using the UNION operator?



1. Number of columns

Every SELECT statement in a UNION query must have the same number of columns.

If the first query returns three columns, the second one should also return exactly three columns, ensuring consistency.

What should be ensured about data types in a UNION query?

2. Data types in Union

➥ **The columns from each SELECT statement must have compatible data types.**

For example, if the first column is of VARCHAR type in one query,

it must also be VARCHAR or a compatible type in the other query.

Does the order of columns matter in a UNION query?

3.Order of columns

➡ **Yes, the order of columns must be the same in each SELECT statement.**

The first column from each query is combined, then the second, and so on.

What happens to duplicate records in UNION?

4. Duplicate records in UNION

➡ **By default, UNION removes duplicate rows from the combined result set.**

If you want to keep duplicates, you should use UNION ALL instead of just UNION.

Can you use column aliases in all SELECT statements in UNION?

5. Column aliases in all SELECT statements

No, column aliases should only be defined in the first SELECT statement.

These aliases will be applied to the final result of the union.

Example:

```
SELECT FirstName, LastName FROM Employees
UNION
SELECT FirstName, LastName FROM Managers;
```

This query combines the FirstName and LastName columns from both the Employees and Managers tables, returning only unique rows.


wanna see some
Counter Questions


FOLLOW US
krishan kumar

1. What is the difference between UNION and UNION ALL?

- **UNION removes any duplicate rows that appear in the result set,**

whereas UNION ALL includes all records, even if they are duplicated across the different SELECT queries.

UNION ALL can be faster since it doesn't need to check for duplicates.

2. Can you perform a UNION between tables with different data types in corresponding columns?

➡ **No, you cannot. The data types in corresponding columns across the SELECT queries must be compatible.**

For instance, if one column is an integer in the first query, the matching column in the second query should also be an integer or a compatible numeric type.

3. How does the performance of UNION compare to UNION ALL?

➡ **UNION ALL is generally faster than UNION because UNION needs to check for and remove duplicates,**

which requires additional processing. UNION ALL skips this step and directly combines the results.

4. Can you use ORDER BY in each SELECT statement of a UNION query?

➡ **No, ORDER BY is only allowed once at the end of the UNION query, after all the SELECT statements have been combined.**

It cannot be used within individual SELECT queries in the UNION.

5. What happens if the column order is different in the SELECT statements?

➥ **If the column order is different between the SELECT statements in a UNION query, the query will fail or return incorrect results because SQL matches columns based on their positions, not names.**

Therefore, the order must be the same in all the queries.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

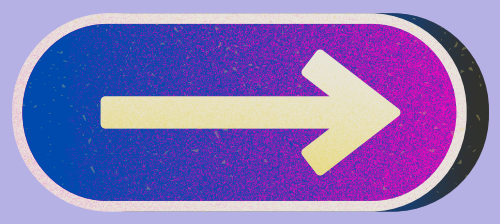
Part - 9

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

What is the difference and similarity between UNION and UNION ALL?



The main difference between UNION and UNION ALL

UNION removes duplicate records from the combined result, showing only unique rows,

while UNION ALL retains all records from both result sets, including duplicates.

UNION is useful when you want to ensure no duplicate records,

whereas UNION ALL is better if you want to keep all records, regardless of duplication.

Which one is faster between UNION and UNION ALL?

Who is faster⏩

➡️ **UNION ALL is generally faster than UNION because it doesn't need to perform the additional step of checking and removing duplicate records.**

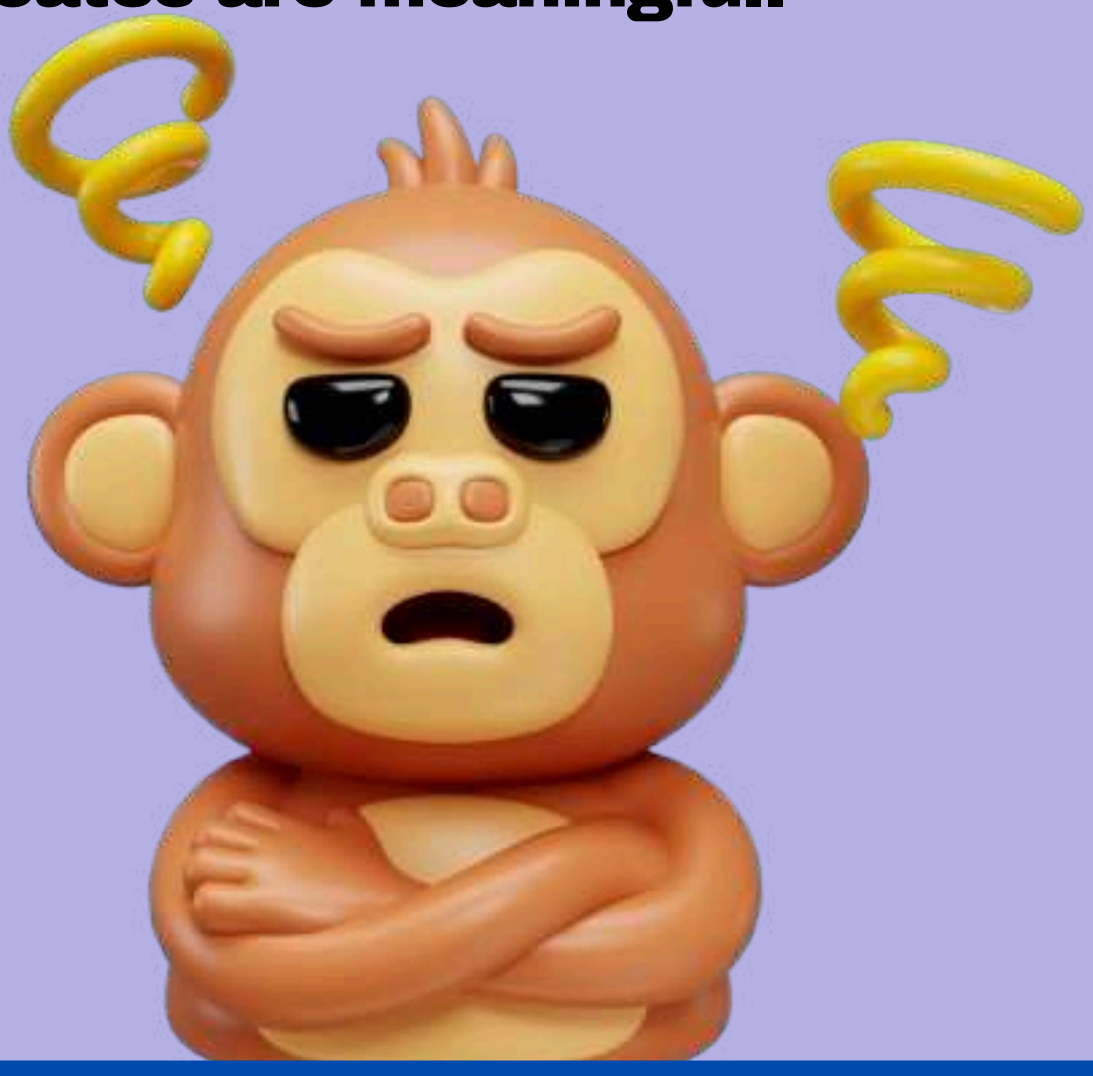
UNION, on the other hand, takes more time as it has to process and remove duplicates from the final result set.

When should you use UNION versus UNION ALL?

Where to use:

- **Use UNION when you need to combine data sets but eliminate duplicate rows. This is often necessary when you want a clean, unique result set.**

Use UNION ALL when you need to combine data sets and keep all records, including duplicates, which can be useful when analyzing raw data or when duplicates are meaningful.



What is the similarity between UNION and UNION ALL?

Similarities:

- **Both UNION and UNION ALL are used to combine the result sets of two or more SELECT queries.**

They require the same number of columns in each query, and the data types of these columns must be compatible across all SELECT statements for a successful query.

What is a simple example of using UNION and UNION ALL?

Examples:

Example of UNION:

```
SELECT FirstName, LastName FROM Employees
UNION
SELECT FirstName, LastName FROM Managers;
```

- **This query combines the Employees and Managers tables but removes any duplicate entries, ensuring each record appears only once in the final result.**

Example of UNION:

```
SELECT FirstName, LastName FROM Employees
UNION ALL
SELECT FirstName, LastName FROM Managers;
```

- **This query combines the Employees and Managers tables and includes all duplicate entries, allowing repeated records to appear in the final result.**

wanna see some

Counter Questions

1. What happens if the number of columns is different in the SELECT statements of a UNION query?

➥ **The query will fail because both SELECT statements in a UNION or UNION ALL must have the same number of columns.**

Each SELECT must return the same number of columns in order to combine the result sets.

2. Can UNION and UNION ALL work with tables having different data types?

➡ **No, the columns in the SELECT statements must have compatible data types.**

For example, if the first column is an integer in one SELECT query, it must also be an integer (or a data type that can be converted to integer) in the other.

3. Can UNION or UNION ALL change the order of the result?

➥ **No, the UNION and UNION ALL operations do not sort the results.**

To control the order of the final output, you must use the ORDER BY clause at the end of the UNION or UNION ALL query.

4. What is the effect of using UNION on performance compared to UNION ALL?

➡ **UNION can slow down performance because it requires additional processing to remove duplicate rows.**

In contrast, UNION ALL is faster as it doesn't remove duplicates, making it more efficient for large datasets where performance is critical.

5. Can you use aliases in the second SELECT statement in UNION queries?

➥ **Column aliases should be defined in the first SELECT statement of a UNION query.**

These aliases are applied to the entire result set, so there's no need to redefine them in subsequent SELECT statements.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

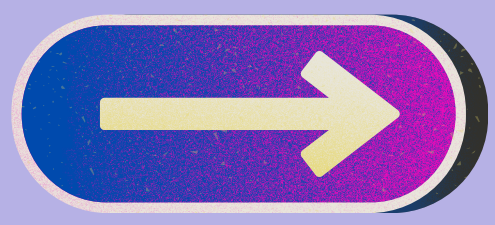
Part -10

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Explain the difference between Join and Union.



What is the main purpose of a JOIN versus a UNION?

JOIN combines related rows from two or more tables based on a related column between them. It retrieves data horizontally by merging columns from different tables into a single row.

UNION, on the other hand, combines the result sets of two or more SELECT queries into one, adding rows from the second query below the rows from the first query, making the result set longer.

How do JOIN and UNION handle data?

JOIN and UNION handle data

- **JOIN merges columns from different tables into a single result based on specified conditions. This approach widens the dataset.**

UNION stacks the results from multiple queries vertically, adding one result set after another, which extends the number of rows.

When should you use JOIN versus UNION?

When to use:

➡ **Use a JOIN when you need to retrieve data from multiple tables by establishing a relationship between them using related columns,**

like matching CustomerID in Customers and Orders tables.

Use a UNION when you want to combine the output of multiple queries that return a similar structure, like merging lists of employees and managers.

What is the syntax difference between JOIN and UNION?

Syntax:

➡ **In a JOIN, you specify the tables to join and the join condition using the JOIN keyword and ON clause:**

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
JOIN Customers
    ON Orders.CustomerID = Customers.CustomerID;
```

➡ **In a UNION, you combine two SELECT statements using the UNION keyword:**

```
SELECT FirstName, LastName
FROM Employees
UNION
SELECT FirstName, LastName FROM Managers;
```

What are the similarities between JOIN and UNION?

Similarities between JOIN and UNION?

- **Both are used to combine data from multiple sources into a single result set.**

  While JOIN combines data horizontally, merging columns,

  UNION combines data vertically, merging rows.

  Both require compatible data types for combining results.

wanna see some

Counter Questions

1. Can you use different numbers of columns in a JOIN or UNION?

- **For JOIN, the number of columns can differ, as the columns are selected individually.**

  For UNION, the number of columns must be the same in each SELECT statement.

2. What happens if there are no matching records in a JOIN?

➡ **If no matching records are found, the JOIN result may show NULL values in columns from the non-matching table,**

depending on the type of JOIN used for example,

- **LEFT JOIN,**
- **RIGHT JOIN).**

3. Can you use WHERE conditions with both JOIN and UNION?

➡ **Yes, you can apply WHERE conditions in both cases. In a JOIN, it filters the combined dataset based on conditions.**

In a UNION, you can apply different WHERE conditions for each SELECT query before combining.

4. Does UNION remove duplicate records?

➡ **By default,**

UNION removes duplicate rows.

To keep duplicates,

use UNION ALL.

5. Can JOIN combine more than two tables?

➡ **Yes, you can join multiple tables using additional JOIN clauses,**

specifying relationships between each pair of tables.

you completed one interview question with me,

can you do me a favour

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

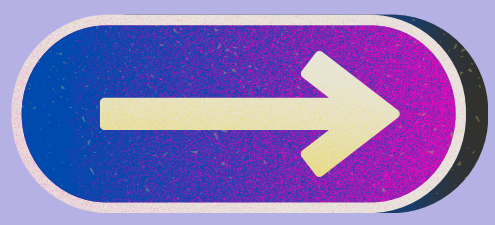
Part - 11

Data Retrieval Interview Questions and Answers..!

krishna kumar

@Krishan kumar

Can we use Aggregate functions as Window functions?



What is a Window function in SQL?

➥ **Yes, aggregate functions like SUM, AVG, COUNT, MIN, and MAX can be used as Window functions.**

A Window function performs calculations across a set of rows related to the current row without collapsing the data.

This lets you perform calculations over different sections of your dataset (windows) while still showing all the original rows

How are Window functions different from traditional Aggregate functions?

Window functions different from traditional Aggregate functions but how?

➡ **Aggregate functions, such as SUM or AVG, calculate values over a group of rows and return a single value for each group.**

When used with GROUP BY, the dataset is collapsed to one row per group.

In contrast, Window functions perform calculations across a subset of rows, defined by the OVER() clause, without collapsing the original rows.

Can you give an example of using SUM() as a Window function?

SUM() as a Window function:

➡ **This query calculates the total salary for each department and retains the original rows, showing each employee's individual salary and department along with the total salary for the department.**

```sql
SELECT
    EmployeeID,
    DepartmentID,
    Salary,
    SUM(Salary) OVER (PARTITION BY DepartmentID) AS TotalDepartmentSalary
FROM Employees;
```

What are some options you can use with the OVER() clause?

some options you can use with the OVER() clause

➡ **The OVER() clause can include:**

- **PARTITION BY: Splits the data into partitions for applying the Window function. Similar to GROUP BY, but rows are not collapsed.**

- **ORDER BY: Specifies the order of rows within each partition, useful for functions like ROW_NUMBER(), RANK(), or calculating running totals.**

- **ROWS/RANGE: Defines a sliding frame within the partition, allowing for calculations like moving averages.**

What are the similarities btw Aggregate and Window functions?

Similarities between Aggregate and Window functions

- **Both can use functions like SUM, AVG, COUNT, etc., to perform calculations.**

While Aggregate functions summarize data across groups of rows, Window functions apply the same calculations over a defined window without collapsing the rows.

wanna see some

Counter Questions

1. What happens if you don't specify PARTITION BY in a Window function?

➡ **Without PARTITION BY, the function applies to the entire dataset, calculating across all rows.**

This is useful for overall metrics, like getting a cumulative sum for all rows together.

2. Can you use Window functions and Aggregate functions together in a query?

➡️ **Yes, you can use both. While Aggregate functions summarize data (e.g., with GROUP BY),**

Window functions perform additional calculations, like rankings, on individual rows within the summarized groups.

3. How does the ORDER BY clause affect Window functions?

➡ **ORDER BY defines the sequence of rows within partitions,**

impacting functions like RANK() and running totals.

Without it, results may not follow the intended row order.

4. What are some common use cases for Window functions?

➡ **Window functions are used for running totals, moving averages, rankings, and percentiles.**

They’re helpful in time-series analysis and trend reporting without collapsing individual rows.

5. Can Window functions be nested?

➡ **No, they can’t be nested directly.**

However, you can use multiple Window functions separately or combine them with other functions in the same query.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

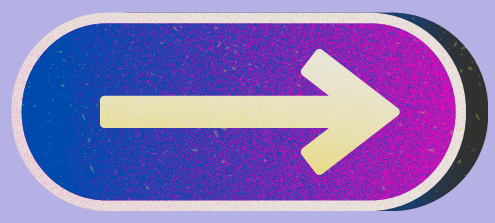
Part - 12

Data Retrieval Interview Q & A ..!

krishan kumar

@Krishan kumar

How can you fetch common records from two tables?



What is an INNER JOIN?

➥ **An INNER JOIN is used in SQL to fetch records that have matching values in both tables.**

It returns only the rows where there is a match between specified columns.

Why use INNER JOIN for this task?

Use of INNER JOIN

- **INNER JOIN is ideal for fetching common records because it only includes rows that satisfy the matching condition across both tables.**

If there's no match, the rows are excluded from the results.

Can you give an example of using it for better understanding?

Example Scenario:

➡ **Suppose we have two tables, TableA and TableB, which both contain an ID column:**

TableA:

ID	Name
1	John
2	Alice
3	Bob

TableB:

ID	Product
2	Laptop
3	Tablet

What are SQL query for this problem lets see.

SQL Query to Fetch Common Records:

```sql
SELECT
    TableA.ID,
    TableA.Name,
    TableB.Product
FROM
    TableA
INNER JOIN
    TableB
ON
    TableA.ID = TableB.ID;
```

Result:

ID	Name	Product
2	Alice	Laptop
3	Bob	Tablet

Let's provide us a Summary:

Summary:

- **INNER JOIN: Returns rows with matching values in both tables based on the specified condition.**

- **Use Case: Ideal for identifying overlapping data, like shared records between different datasets.**

wanna see some

Counter Questions

1. What happens if there are no matching records?

➥ **If there are no matching values, an INNER JOIN will return an empty result set.**

This makes it different from LEFT JOIN or RIGHT JOIN, which would still include non-matching rows from one of the tables.

It is useful when you only need rows with exact matches.

2. Can we use multiple columns for joining?

➡ **Yes, you can join tables using multiple columns by specifying multiple conditions in the ON clause.**

For example,
ON TableA.ID = TableB.ID
AND TableA.Name = TableB.Name.

This helps to ensure more specific matches between the tables.

3. How is INNER JOIN different from OUTER JOIN?

➡ **An INNER JOIN returns only matching rows, while an OUTER JOIN (like LEFT JOIN or RIGHT JOIN) includes all rows from one table and matching rows from the other.**

OUTER JOIN helps in scenarios where you want to keep all records from one table regardless of matches.

4. Can we fetch common records from more than two tables?

➡ **Yes, you can use multiple INNER JOIN clauses to fetch common records from three or more tables. Just add another INNER JOIN for each additional table.**

This is useful in complex queries where multiple relationships exist.

5. What if the column names are different in both tables?

➡ **You can still join the tables by specifying the correct column names in the ON clause, like:**

ON TableA.ID_A = TableB.ID_B.

Using aliases for table names can also make the query easier to read and manage.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

Part - 13

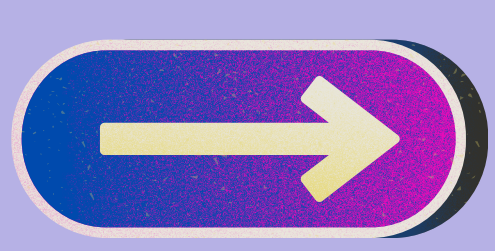
Data Retrieval Interview Q & A ..!

krishan kumar

@Krishan kumar

How can you fetch alternate records from a table?

Fetching alternate records (every second record) can be done using various techniques depending on the SQL system.
There are some examples using different approaches.



Example Scenario:

Let’s assume we have an Employees table as shown below:

ID	Name
1	John
2	Alice
3	Bob
4	Charlie
5	David
6	Eve

Let’s solve with SQL Server😎

1. SQL Server (Using ROW_NUMBER()):

➡️ **To fetch alternate records in SQL Server, the ROW_NUMBER() function can be used to generate sequential row numbers.**

```sql
WITH NumberedEmployees AS (
    SELECT *,
           ROW_NUMBER() OVER (ORDER BY ID) AS RowNum
    FROM Employees
)
SELECT ID, Name
FROM NumberedEmployees
WHERE RowNum % 2 = 1;  -- Fetches every second record starting from the first
```

➡️ **Explanation:**

- **ROW_NUMBER() OVER (ORDER BY ID):** Assigns a unique sequential number to each row based on the order of ID.

- **RowNum % 2 = 1:** Filters the rows to return only those where the row number is odd, resulting in alternate records.

Can we solve this using MySQL?

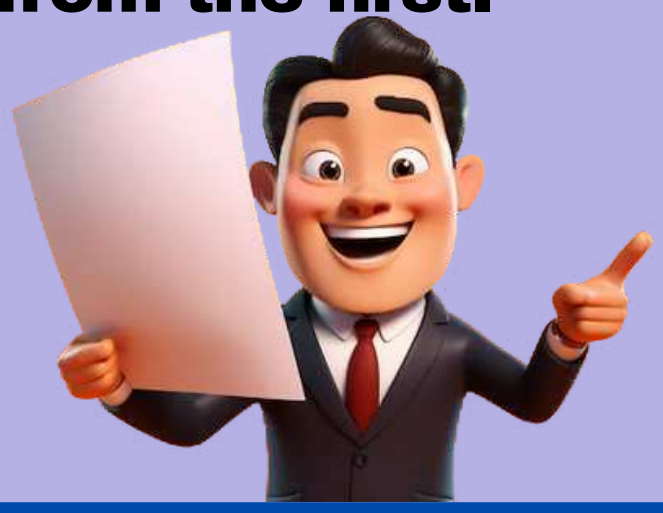
2. MySQL (Using MOD() function):

➡ **In MySQL, the MOD() function can be used in combination with the AUTO_INCREMENT or a manually created row number.**

```sql
SELECT *
FROM Employees
WHERE MOD(ID, 2) = 1;  -- Fetches records where ID is odd
```

➡ **Explanation:**

MOD(ID, 2) = 1:
Returns records where the remainder of dividing ID by 2 is 1, which effectively selects every other row starting from the first.



Can we solve this using PostgreSQL ?

3. PostgreSQL (Using ROW_NUMBER()):

➡️ **Similar to SQL Server, PostgreSQL can also use ROW_NUMBER().**

```sql
SELECT ID, Name
FROM (
    SELECT *,
           ROW_NUMBER() OVER (ORDER BY ID) AS RowNum
    FROM Employees
) AS NumberedEmployees
WHERE RowNum % 2 = 1;
```

I'm Confuesed what to use...!

Summary:

- **Approach:** **Use functions like ROW_NUMBER() or MOD() to filter alternate rows.**
- **Applicability:** **Different techniques apply to different SQL systems.**

wanna see some

Counter Questions

1. Why use ROW_NUMBER() for fetching alternate records?

- **ROW_NUMBER() allows assigning a sequential number to rows,**

making it easy to filter based on custom criteria like odd/even row numbers.

This approach offers flexibility and is supported in various databases, allowing different ordering conditions.

2. How does using MOD() help in fetching alternate records?

➡ **The MOD() function computes the remainder when a column value is divided by a number.**

In cases where the table's column values (like ID) are sequential,

MOD() can filter alternate rows by returning rows where the remainder is 1.

3. What if the IDs in the table are not sequential?

➡ **If IDs are not sequential, ROW_NUMBER() becomes useful because it generates sequential numbers based on a specified order.**

This way, you can still fetch alternate rows regardless of the original values in the column.

4. What is the difference between ROW_NUMBER() and RANK() for fetching alternate records?

➥ **While both functions assign row numbers, ROW_NUMBER() always increments sequentially without considering duplicates,**

whereas RANK() considers duplicate values and assigns the same rank, potentially leading to gaps in numbering.

For fetching alternate records, ROW_NUMBER() is more appropriate.

5. Can we start fetching from the second record instead of the first?

➡ **Yes, modifying the filtering condition to**

RowNum % 2 = 0 or MOD(ID, 2) = 0

will start fetching from the second row. This small adjustment helps switch between odd and even rows.

you completed one interview question with me,

can you do me a favour

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

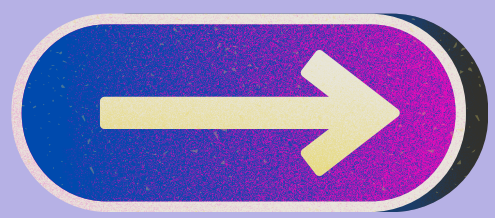
Part -14

Data Retrieval Interview Q & A ..!

krishan kumar

@Krishan kumar

Name the Operator Used for Pattern Matching in SQL?



Like Operator:

- **In SQL, the operator used for pattern matching is the LIKE operator.**

  It allows you to filter data based on a specific pattern,

  where the pattern may include wildcard characters that represent variable or unknown characters.

Let's See the Sntex of Like operator😎

Syntax:

Basic Syntax of the LIKE Operator:

```sql
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

Pattern Matching Wildcards:

- **%: Matches any sequence of characters (including zero characters).**

- **_: Matches exactly one character.**

Let's see the example for it ?

Examples:

- **LIKE 'a%':** Matches any value that starts with "a".
- **LIKE '%a':** Matches any value that ends with "a".
- **LIKE '%or%':** Matches any value that contains the sequence "or".
- **LIKE '_r%':** Matches any value where "r" is the second character.

Example Query:

```sql
SELECT *
FROM customers
WHERE customer_name LIKE 'J%';
```

Explanation:

The above query retrieves all rows from the "customers" table where the customer_name starts with the letter "J".

Swipe right for surprise 🥳

Swipe right for the jackpot of the day 😉

wanna see some

Counter Questions

1. What is the difference between % and _ wildcards in the LIKE operator?

- **The % wildcard matches any sequence of characters, including an empty string, whereas the _ wildcard matches exactly one character.**

For instance, LIKE 'A%' matches "Apple", "Ape", and "Ant", while LIKE 'A_' would only match two-letter words starting with "A".

2. Can the LIKE operator be case-sensitive?

➡ **It depends on the database system. In most cases, the LIKE operator is case-insensitive by default, such as in MySQL.**

However, databases like PostgreSQL offer case-sensitive pattern matching with the ILIKE operator.

Additionally, collations in some databases can affect case sensitivity.

3. How can the LIKE operator be used to filter data based on multiple patterns?

➥ **You can combine multiple LIKE conditions using the OR keyword.**

For example, to match values starting with "A" or "B", you can use
WHERE column_name LIKE 'A%' OR column_name LIKE 'B%'.

This filters results based on either pattern.

4. What is an alternative approach for complex pattern matching in SQL?

➥ **For complex patterns, the REGEXP (Regular Expression) operator can be used.**

It provides more advanced pattern matching capabilities compared to LIKE, such as specifying character ranges and repeated patterns.

However, REGEXP is not available in all SQL databases.

5. How does LIKE perform with large datasets, and can it affect query performance?

The LIKE operator can slow down query performance, especially with leading wildcards (e.g., %pattern).
This is because the database needs to scan each row. To improve performance, indexing strategies or using full-text search features may be considered.

you completed one interview question with me,

can you do me a favour

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar

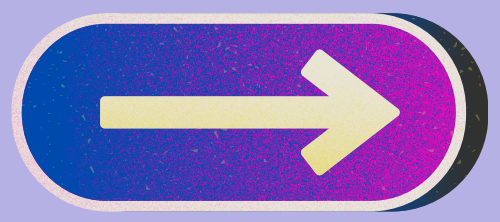
Part - 15

Data Retrieval Interview Q & A ..!

krishan kumar

@Krishan kumar

How can you fetch the first 5 characters from a string?



SUBSTRING :

➡ **To fetch the first 5 characters from a string in SQL, you can use the SUBSTRING (or SUBSTR in some databases) function.**

This function allows you to extract a specific portion of a string starting from a specified position and for a specified length.

Let's See the Sntex of SUBSTRING😎

Syntax:

Basic Syntax of the SUBSTRING:

```
SELECT SUBSTRING(column_name, start_position, length)
FROM table_name;
```

Let's see the example for it ?

Examples:

Let's say we have an Employees table with a column named Name:

ID	Name
1	Krishna
2	Manisha
3	Radhika
4	Manish
5	Puneet
6	Kavya

Swipe right for the Query 🥳

Query:

➡️ SQL Query to Fetch the First 5 Characters:

```sql
SELECT
        ID,
        Name,
        SUBSTRING(Name, 1, 5) AS FirstFiveChars
FROM
        Employees;
```

➡️ Explanation:

SUBSTRING(Name, 1, 5):

- **Name:** The column from which the substring is to be extracted.
- **1:** The starting position in the string (1 indicates the first character).
- **5:** The number of characters to extract from the starting position.

Swipe for the result😱

Result:

➡️ **Let's say we have an Employees table with a column named Name:**

D	Name	FirstFiveChars
1	Krishna	Krish
2	Manisha	Manis
3	Radhika	Radhi
4	Manish	Manis
5	Puneet	Punee
6	Kavya	Kavya

Swipe right for the surprise 🥳

Swipe right for the jackpot of the day 😉

wanna see some

Counter Questions

1. What if the string has fewer than 5 characters?

- **If the string has fewer than 5 characters, the SUBSTRING function will return all the available characters without throwing an error.**

For example, if the name is "Eve," the result will simply be "Eve."

2. Can we use a different function to achieve the same result?

➡ **Yes, you can use LEFT function in many databases.**

The LEFT(column_name, number_of_characters) function extracts a specified number of characters from the left side of the string.

For example, LEFT(Name, 5) will give the same result as SUBSTRING(Name, 1, 5).

3. What is the difference between SUBSTRING and LEFT functions?

➡ **The primary difference is flexibility. While LEFT extracts characters from the left side of the string,**

SUBSTRING allows you to specify both the starting position and the length, making it more versatile.

SUBSTRING can be used to extract characters from any position within the string, not just the beginning.

4. Is there a way to handle cases where the starting position is larger than the string length?

- **If the starting position specified in the SUBSTRING function exceeds the string length, the result will be an empty string.**

For instance, if SUBSTRING('Hello', 10, 5) is used, the output will be an empty string, as there is no 10th character in "Hello."

5. How can you fetch characters from the end of a string?

➡ **To fetch characters from the end of a string, you can use the RIGHT function in most databases.**

For example, RIGHT(Name, 5) will extract the last 5 characters of the Name column.

Find This Useful

Time to hit that like button
and give it some love! 🤩

Visit my Linkedin for such amazing Content😉

in krishan kumar