Part - 13

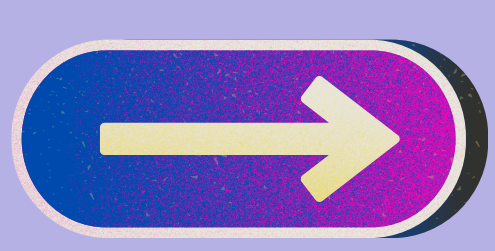
Data Retrieval Interview Q & A ..!

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?

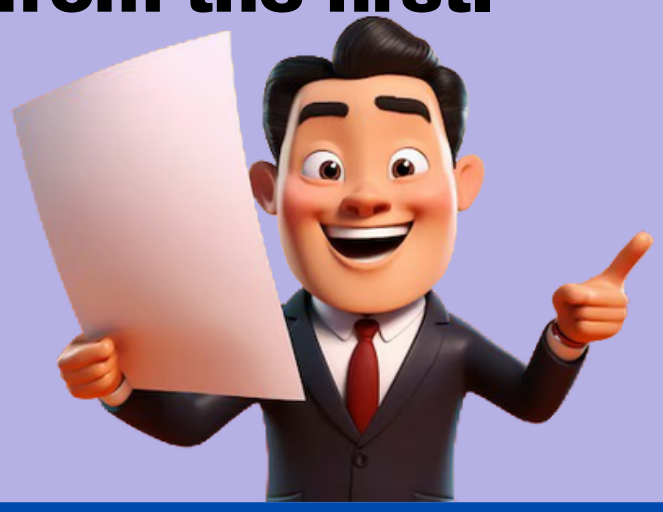
2. MySQL (Using MOD() function):

➡ **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.

Find This Useful

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

Visit my Linkedin for such amazing Content😉

in krishan kumar