SQL Query Optimization

Agenda:

In today's session, we'll cover essential topics, including:-

- Busting a common myth about the COUNT()
- Selecting only the columns that you truly need
- Why LIMIT can be a hidden pitfall
- Harnessing EXISTS() instead of COUNT()
- Leveraging APPROX_COUNT_DISTINCT for efficient counting
- Swapping Self-Joins with Windows Functions
- The art of trimming your data early and frequently
- When to choose MAX() over RANK()
- Importance of ordering your JOINs, from larger to smaller tables
- Does the WHERE clause sequence really matter?
- Should we push ORDER BY to the end of the query?

Summary of Previous Lecture:

Ad-hoc reporting

In the data analysis world, being asked questions, exploring a database, writing SQL statements to find and pull the data needed to determine the answers, and conducting the analysis of that data to calculate the answers to the questions, is called **ad-hoc reporting.**

Common Table Expressions

- Common Table Expressions (**CTEs**) allow us to create temporary named results sets that exist temporarily within the execution scope of SQL statements such as SELECT, INSERT, UPDATE, DELETE, and MERGE.
- The **WITH** clause in SQL provides a better way to write the auxiliary/helper statement which can be later used in larger queries.
- These statements are referred to as common table expressions which are nothing but defining a temporary relational table to be used later in a SQL statement.
- The table is being called temporary because it exists only during the scope of the SQL statement written after CTEs.

The syntax for CTEs is:

```
WITH [query_alias] AS (
[query]
),
[query_2_alias] AS (
[query_2]
)
SELECT [column list]
FROM [query_alias]
```

... [remainder of query that references aliases created above]

-> where "[**query_alias**]" is a placeholder for the name you want to use to refer to a query later, and "[**query**]" is a placeholder for the query you want to reuse.

Advantages of CTE:

- Making recursive queries.
- Hold a query output virtually in a temporary area named as given while definition.
- No need to save Metadata.
- Useful when there is a need to do more operations on some query output.

Views

- Another approach to CTEs is Views.
- A view is treated just like a table in SQL, the only difference being that it has run when it's referenced to dynamically generate a result set (where a table stores the data instead of storing the query),
- So queries that reference views can take longer to run than queries that reference tables.
- However, the view is retrieving the latest data from the underlying tables each time it is run, so you are working with the freshest data available when you query from a view.

Syntax:

```
CREATE VIEW [db_name.]view_name [(column_list)]
AS
  select-statement;
```

- If you want to store your dataset as a view, you simply precede your SELECT statement by replacing the bracketed statements with the actual schema name, and the name you are giving the view.

When should you use each one?

Although some differences exist, common table expressions and views seem to perform very similarly.

- **Ad-hoc queries.**
    - For queries referenced occasionally (or just once), it's usually better to use a CTE.
    - If you need the query again, you can just copy the CTE and modify it if necessary.
- **Frequently used queries.**
    - Creating a corresponding view is a good idea if you often reference the same query.
    - However, you'll need to create view permission in your database to create a view.
- **Access management.**
    - A view might restrict particular users' database access while allowing them to get the necessary information.
    - You can give users access to specific views that query the data they're allowed to see without exposing the whole database.
    - In such a case, a view provides an additional access layer.