MySQL Setup & Intro to DDL, DML commands

Agenda:

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

- MySQL Workbench Setup
- Types of SQL commands
- Constraints
- Data Types
- Data Definition Language (DDL) commands:
    - CREATE
    - ALTER
    - TRUNCATE
    - DROP
- Data Manipulation Language (DML) commands:
    - INSERT
    - UPDATE
    - DELETE

Summary of Previous Lecture:

Busting a Common Myth about COUNT()

- Compared COUNT(*) and COUNT(1) for performance.
- Found no significant difference in performance between them.

Only Select the Columns You Really Need

- Emphasized the importance of selecting only necessary columns to improve efficiency.
- Showcased a query with fewer selected columns outperforming one with SELECT *.

LIMIT is a Trap

- Explained that LIMIT improves performance but doesn't reduce costs.
- The row restriction of the LIMIT clause is applied after SQL databases scan the full range of data.
- Demonstrated how LIMIT can significantly reduce shuffle time.

Use EXISTS() Instead of COUNT()

- Compared EXISTS() and COUNT() for checking the existence of a value.
- Showed that EXISTS() is more efficient when only existence matters and we don’t need to know how frequently the value occurs.

Use APPROX_COUNT_DISTINCT Instead of COUNT(DISTINCT) for large datasets

- Highlighted the efficiency (not accuracy) of APPROX_COUNT_DISTINCT() over COUNT(DISTINCT).
- In cases when data volumes are significant, do consider the option of trading accuracy for performance by utilizing the approximate aggregate functions.

Replace Self-Join with Windows Function

- Compared self-join and window function for analyzing data.
- Showed that window functions are more efficient and simpler.

Trim Your Data Early and Often

- Advised filtering and aggregating data early in the query to reduce shuffling.

Use MAX() Instead of RANK()

- Showed how MAX() can replace RANK() for certain scenarios, improving query simplicity and efficiency.

Order Your JOINs from Larger to Smaller Tables

- Explained the importance of table join order and demonstrated its impact on performance.
- Table join order matters for reducing the number of rows that the rest of the query needs to process.
- In general practice, keeping the larger table in that initial query may boost the query performance.

Sequence of WHERE Clauses matters?

- Explored the sequence of WHERE clauses and concluded that BigQuery's SQL Optimizer is smart enough to run the most selective WHERE clause regardless of how we wrote the query.

Push ORDER BY to the End of the Query?

- Discussed why unnecessary ORDER BY clauses should be removed.
- ORDER BY should be postponed until the outermost query.
- Confirmed that BigQuery's SQL Optimizer is again smart enough to figure out the redundant clauses and automatically exclude them from the computation.
- However, some other legacy databases may not have the same capability.