

DAY 3

# Takeaways

- **Sub Queries** are queries which generate output that will be used as input to the main query
- Queries that provide **a single record**, list or even a table as output can be used as a subquery

# Takeaways

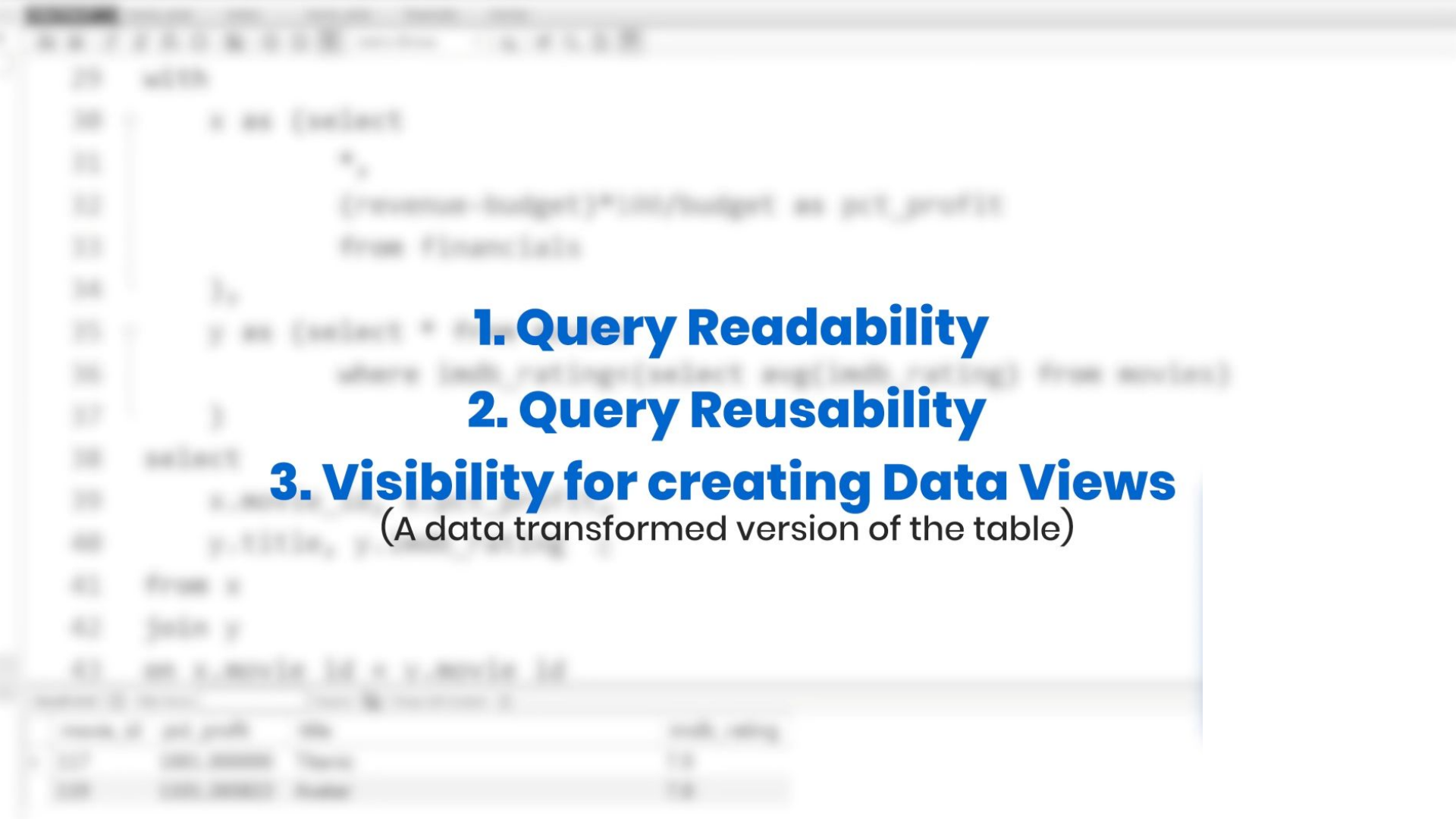
- **IN, ANY & ALL** clauses expect a list as input
- **ANY** clause executes the condition for any one of the values on the list that meets the condition which is the minimum value by default
- **ALL** clause executes the condition where all the values on the list meet the condition which is the maximum value of the list

# Takeaways

- **A subquery is called a co-related query** when its execution depends upon the statement(s) written after the bracket
- One needs to choose between writing a **subquery or a co-related query** depending on its performance
- **EXPLAIN ANALYSE clause** before any query will provide the query execution plan through which one can understand the query performance

# Takeaways

- **Common Table Expression (CTE)** creates a temporary table within a query
- **WITH** and **AS** clauses are used in combination to create **CTE**
- One can create **multiple CTEs** inside a query

- 
- The background of the slide is a blurred screenshot of a SQL query editor. The query is a complex SELECT statement with multiple subqueries and joins. It includes table names like 'Financials', 'Sales', and 'Marketing', and various SQL keywords like 'SELECT', 'FROM', 'JOIN', 'WHERE', 'WITH', and 'AS'. The text is in a monospaced font, typical of code editors.
- 1. Query Readability**
  - 2. Query Reusability**
  - 3. Visibility for creating Data Views**  
(A data transformed version of the table)

# Takeaways

- There are multiple benefits of writing **CTEs** such as **Query readability, reusability** and **creating views**
- **Recursive subqueries** in **CTEs** have several applications involving data series generation



**And... its done!**