📢 Day 2 of SQL Series: Implicit and Explicit Joins! 🤝

Welcome back to our SQL Series! Today, we'll explore the concepts of implicit and explicit joins. Joining tables allows us to combine data from multiple tables based on specified conditions.

Explicit join syntax is the more traditional way of writing SQL joins. It is more verbose, but it is also more explicit and easier to understand.

Implicit join syntax is a newer way of writing SQL joins. It is shorter and more concise, but it can be more difficult to understand and debug.

Implicit joins use comma-separated table names in the FROM clause, and the join condition is specified in the WHERE clause. However, explicit joins use the JOIN keyword followed by the table name and the ON keyword to specify the join condition explicitly.

“””

In real-world business applications, SQL queries typically have 7-8 joins with 12-16 join conditions. Nested joins or other more advanced cases are used in about 1 out of every 10 or 20 queries. Explicit join syntax is much easier to maintain, debug, and develop than implicit join syntax. This is important for business software, where speed and safety are critical. Implicit join syntax is somewhat easier to code if you create SQL statements dynamically through application code. However, explicit join syntax is generally the preferred option for most business applications.

“””

Tip: Try using EXPLAIN statement it will help to optimizing queries and debugging problems.

✨ Join + ON:

The JOIN operation combines rows from two or more tables based on a related column between them. The ON clause specifies the condition for the join. An INNER JOIN is often referred to as just a JOIN.

When using joins, it's important to understand the order of operations:

🗒The WHERE clause filters records after the join has taken place.

🗒The ON clause filters records from the right table before the join.  
Try this out, you will get a better understanding: <https://dbfiddle.uk/91VQS9x0>

✨ Left Join == Left Outer Join:

A LEFT JOIN returns all records from the left table (the one specified before the JOIN keyword), and the matched records from the right table based on the specified join condition. If there's no match, NULL values are returned for the right table.

✨ Right Join == Right Outer Join:

A RIGHT JOIN returns all records from the right table and the matched records from the left table. Any unmatched rows from the left table will contain NULL values.

✨ Full Outer Join==Full join:

A FULL OUTER JOIN returns all records when there is a match in either the left or the right table. If there's no match, NULL values are returned for the respective table. Some of the use cases: Finding mismatched or orphaned data, Running exception reports etc…

✨ Cross Join-Cartesian product:

A CROSS JOIN, also known as a Cartesian product, combines each row from the first table with every row from the second table, resulting in a new table with the total number of rows being the product of the rows in both tables. Cross joins don’t use ON generally but it is allowed to use ON condition whereas natural join is not. Cross joins are **expensive.**

https://dbfiddle.uk/0ONmdMKL

✨Natural join(❌ON):

Natural join is performed automatically by the database. The database will find the join condition based on the column names of the two tables being joined. The natural join is similar to an inner join, but it does not require the user to specify the join condition.

The natural join will only join columns that have the same name and data type. If there are no columns with the same name, then the database will perform a cross join. If there are multiple columns with the same name, then the database will join all of the columns with the same name.

Using ON together with natural join will throw u error.  
NOTE: MySQL, Oracle, SQLite, PostgreSQL these are the only servers support natural joins

https://dbfiddle.uk/Kc3uZrUE

✨Self join(Aliases is important):

A self join is a join that is performed on the same table. This can be useful for finding relationships between rows in the same table. To perform a self join, you need to use the JOIN keyword twice.Remember to provide clear aliases for tables when performing self joins to differentiate between the different instances of the same table.

For example,

--you could use a self join to find all of the employees who report to a specific manager.

Solution here: https://dbfiddle.uk/CgP9YCmo

Stay tuned for more SQL knowledge in the upcoming posts! If you have any questions or specific topics you'd like to explore further, feel free to leave them in the comments below.

Happy querying! 💡

#SQLSeries #JoinOperations #SQLTipsAndTricks #DatabaseManagement