W09 Paper: Case Study Working with Outer Joins in SQL queries.

You have had a profitable ninth week at your new company. You mastered how to write outer joins.

Your boss decided to see if you now have the skills to write outer joins and explain them to those uninitiated in SQL. Specifically, your manager would like you to explain:

* How left and right joins return result sets.
* How combining outer join tables requires using a series of only left joins unless you rely on the foreign key values of an association table in a three table join.
* How cross joins let you fabricate result sets.
* How natural joins rely on existing column names.

You should return and report with a 3-5 paragraph report that clearly explains what you learned about writing outer joins.

Report:

During this week, I have learned how to write outer joins in SQL. Outer joins mainly include left joins and right joins. A left join returns all records from the left table and matching records from the right table; if there is no match, the result will show NULL in the columns of the right table. Conversely, the right join returns all records from the right table and matching records from the left table, filling with NULL where there is no match.

When we combine tables using outer joins in complex configurations, such as three-table joins, we exclusively use a series of left joins, unless we rely on foreign key values from an association table. This is because left joins ensure that we preserve all records from the main table, while trying to include related data from other tables. This approach is crucial for preserving data integrity and completeness, especially when the association table acts as a bridge between the other two tables, facilitating a more accurate and efficient data merge.

Cross joins make it possible to produce result sets by combining each row of one table with all the rows of another, generating a Cartesian product. This operation requires no join conditions, which is ideal for creating large volumes of comprehensively combined data. Crossovers are useful in analysis and testing, where it is necessary to generate all possible combinations between data sets to evaluate performances or explore all potential interactions between data from different sources.

Natural joins rely on existing column names that are common to both tables involved in the join. This type of join automatically matches columns with the same name and data type, simplifying queries by omitting the need to specify explicit join conditions. Although natural joins can make query writing more efficient, they require high discipline in database design and a thorough understanding of table structures to avoid unexpected or incorrect results due to implicit column matching.