


☐

I'm not robot


reCAPTCHA

I'm not robot!

p.project_ID, e.employee_ID; +-----+ | PROJECT_NAME | EMPLOYEE_NAME | |-----| | COVID-19 Vaccine | Terry Smith | | COVID-19 Vaccine | Maria Inverness | | Malaria Vaccine | Pat Wang | +-----+ Important Although the two queries in this example produce the same output when they use the same condition (e.project_id = p.project_id) in different clauses (WHERE vs. FROM ... ON ...), it is possible to construct pairs of queries that use the same condition but that do not produce the same output. The most common examples involve outer joins. If you execute table1 LEFT OUTER JOIN table2, then for rows in table1 that have no match, the columns that would have come from table2 contain NULL.

A filter like WHERE table2.ID = table1.ID filters out rows in which either table2.id or table1.id contains a NULL, while an explicit outer join in the FROM ... ON ... clause does not filter out rows with NULL values. In other words, an outer join with a filter might not actually act like an outer join. A natural join is used when two tables contain columns that have the same name and in which the data in those columns corresponds. In the employees and projects tables shown above, both tables have columns named "project_ID".

A natural join implicitly constructs the ON clause: ON projects.project_ID = employees.project_ID. If two tables have multiple columns in common, then all the common columns are used in the ON clause. For example, if you had two tables that each had columns named "city" and "province", then a natural join would construct the following ON clause: ON table2.city = table1.city AND table2.province = table1.province. The output of a natural join includes only one copy of each of the shared columns. For example, the following query produces a natural join containing all columns in the two tables, except that it omits all but one copy of the redundant project_ID column: SELECT * FROM projects NATURAL JOIN employees ORDER BY employee_ID; +-----+-----+ | PROJECT_ID | PROJECT_NAME | EMPLOYEE_ID | EMPLOYEE_NAME | |-----+-----+ | 1000 | COVID-19 Vaccine | 10000001 | Terry Smith | | 1000 | COVID-19 Vaccine | 10000002 | Maria Inverness | | 1001 | Malaria Vaccine | 10000003 | Pat Wang | +-----+-----+ A natural join can be combined with an outer join. A natural join cannot be combined with an ON clause because the join condition is already implied. However, you can use a WHERE clause to filter the results of a natural join. Syntactically, there are two ways to join tables: Use the JOIN operator in the ON sub-clause of the FROM clause. Use the WHERE with the FROM clause. Snowflake recommends using the ON sub-clause in the FROM clause. The syntax is more flexible. And specifying the predicate in the ON clause avoids the problem of accidentally filtering rows with NULLs when using a WHERE clause to specify the join condition for an outer join. Was this page helpful?Privacy NoticeSite Terms© 2023 Snowflake, Inc. All Rights Reserved.IntroductionTypes of JoinsImplementing JoinsJOINFROMWHERELateral Join