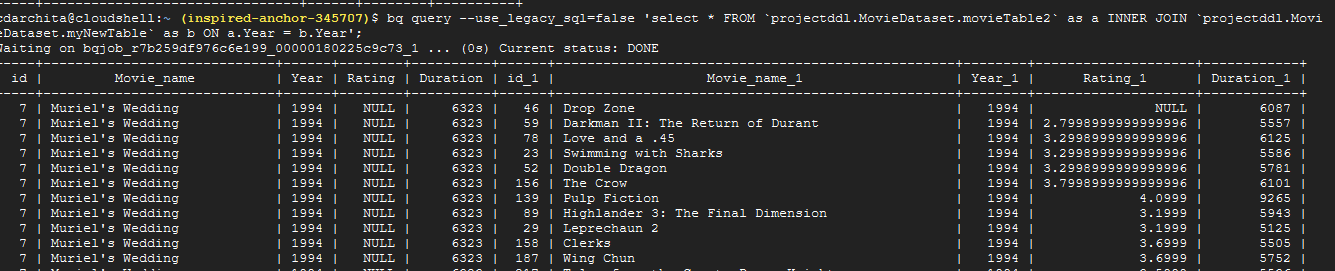
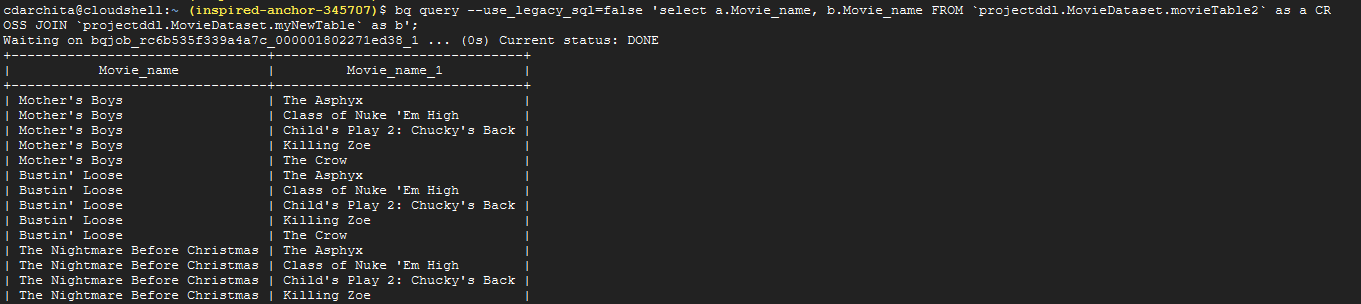
**Inner join:** takes the Cartesian product of two tables and discards all rows that do not meet the join condition.



**CROSS JOIN**

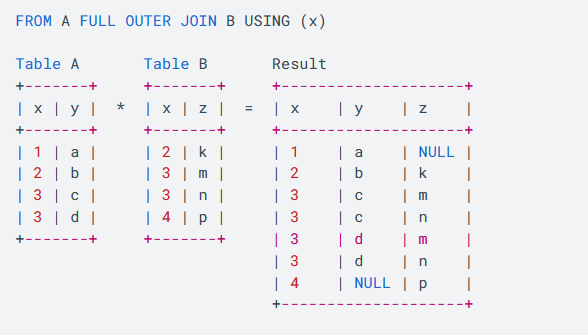
It returns Cartesian product of two tables. If the rows of the two tables are independent, then the result has M \* N rows, given M rows in one and N in the other.



**FULL [OUTER] JOIN**

FULL indicates that all rows from both the tables are returned, even if they do not meet the join condition. **Outer** indicates if a given row does not join to any row in other, the row returns NULLs for all columns from the 2ND table.

from a left outer join b on a.w = b.y



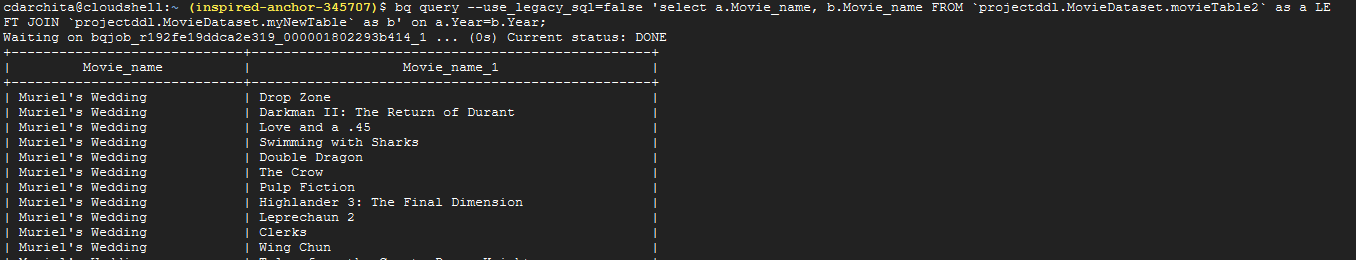
In an outer join, **unmatched rows in one or both tables can be returned**. There are a few types of outer joins:

* LEFT JOIN returns only unmatched rows from the left table.
* RIGHT JOIN returns only unmatched rows from the right table.
* FULL OUTER JOIN returns unmatched rows from both tables.

**LEFT [OUTER] JOIN**

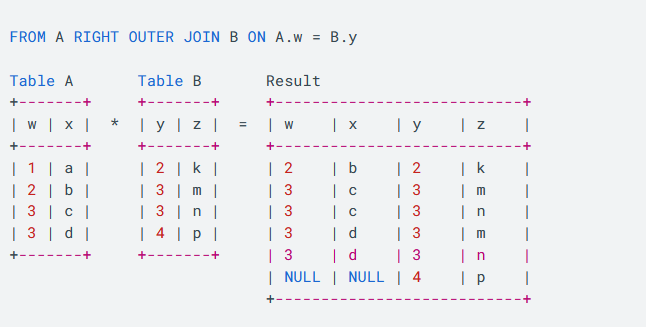
The result of a LEFT OUTER JOIN (or simply LEFT JOIN) for two tables always retains all rows from left table in the JOIN operation, even if no rows in the right table satisfy the join predicate.

bq query --use\_legacy\_sql=false 'select a.Movie\_name, b.Movie\_name FROM `projectddl.MovieDataset.movieTable2` as a LEFT JOIN `projectddl.MovieDataset.myNewTable` as b' on a.Year=b.Year;

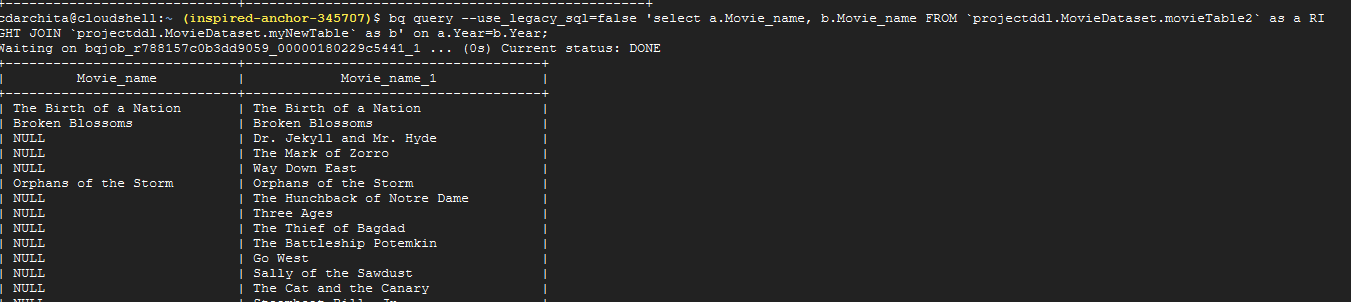


**Right outer join**

It preserves the unmatched rows from the second (right) table, joining them with a NULL in the shape of the first (left) table.



bq query --use\_legacy\_sql=false 'select a.Movie\_name, b.Movie\_name FROM `projectddl.MovieDataset.movieTable2` as a RIGHT JOIN `projectddl.MovieDataset.myNewTable` as b' on a.Year=b.Year;



**With**

In order to find out how much each customer has spent in a given month, you would perform an OUTER JOIN between **Transactions** fact table with **Customer** dimension table to get the results.

* Using [WITH](https://cloud.google.com/bigquery/docs/reference/standard-sql/query-syntax#with-clause) clause allows to name a subquery and use it in subsequent queries such as the SELECT statement

Q. **We use RIGHT OUTER JOIN between Customer and Transactions to get a list of all the customers with their total spend.**

with

transaction as

(

SELECT 1001 AS id, '12-08-2017' AS timestamp, 65401 AS customerid, 'ABC123456' AS sku, 3 AS quantity, 36.3 AS price

UNION ALL SELECT 1001, '12-08-2017', 65401, 'TBL535522', 1, 878.4

UNION ALL SELECT 1001, '12-08-2017', 65401, 'CHR762222', 6, 435.6

UNION ALL SELECT 1002, '12-16-2017', 74682, 'GCH635354', 4, 345.7

UNION ALL SELECT 1002, '12-16-2017', 74682, 'GRD828822', 2, 9.6

),

customer\_dimension as

(

SELECT 65401 AS customerid, 'John Doe' AS customername, 'Faraway' AS location

UNION ALL SELECT 74682, 'Jane michael', 'Nearland'

UNION ALL SELECT 63636, 'Jose carlos', 'Nearland'

)

SELECT

c.customername,

SUM(t.price) AS total\_purchased

FROM transaction AS t

RIGHT OUTER JOIN customer\_dimension AS c

ON t.customerid = c.customerid

GROUP BY c.customername;

**o/p**

|  |  |
| --- | --- |
| CUSTOMER NAME | TOTAL PURCHASE |
| |  |  | | --- | --- | |  | John Doe | | 1350.3 |
| Jane michael | 355.3 |
| Jose carlos | null |

**Array\_agg()**

with Emp as

(

select 1 as empno,'jack' as employee, 35 as age,'CS' as deptName

    union all

select 2 as Empno,'zill' as employee, 45 as age,'CS' as deptName

    union all

select 3 as Empno,'david' as employee, 65 as age,'CIV' as deptName

)

select deptName, array\_agg(employee) Nm from Emp group by deptName;

array\_agg(employee)packs a set of employees.

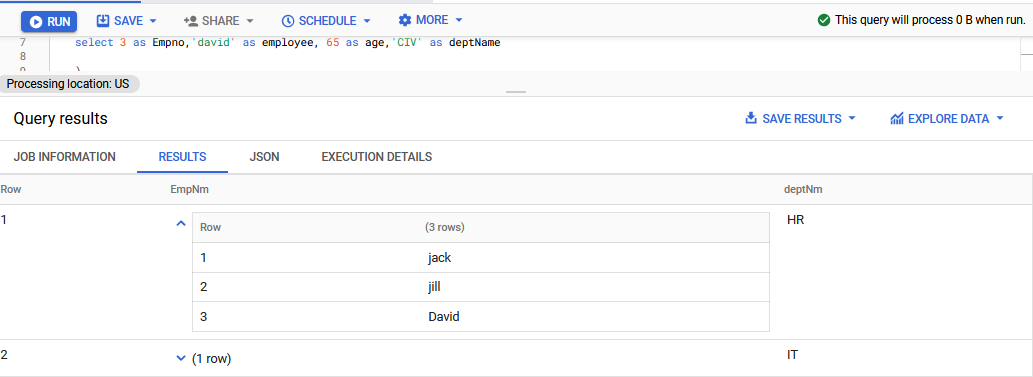
….**union two arrays**……………….

select array['jack','jill','David'] as EmpNm , 'HR' as deptNm

union all

select array['George'] as EmpNm, 'IT' as deptNm

;



* Using arrays and structs to return the question each time with all its answers.
* with agg\_answer as (
* select parent\_id, ARRAY\_AGG(struct(id,body)) from `bigquery-public-data.stackoverflow.posts\_answers` group by parent\_id
* )
* select q.tags question\_tag, q.id question\_id, q.title question\_title, q.body question\_body, a.\*
* from
* `bigquery-public-data.stackoverflow.posts\_questions` as q
* left join
* agg\_answer as a
* on q.id = a.parent\_id
* where regexp\_contains(q.tags,'tensorflow');