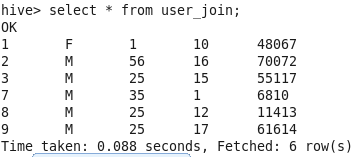
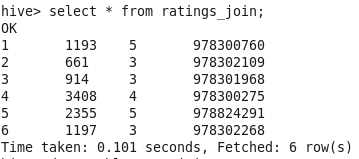
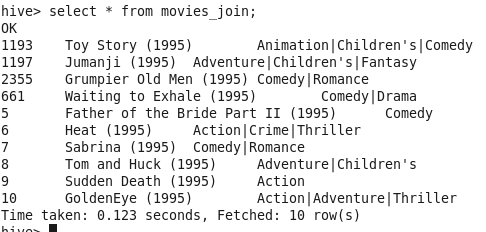
**User:**



**Rating:**

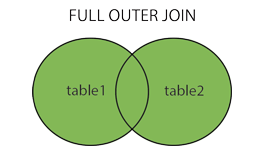


**Movies:**

****

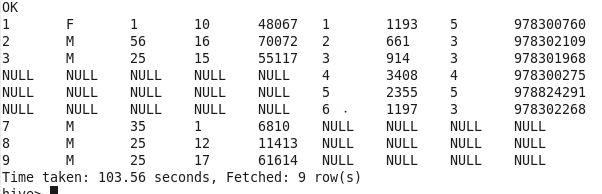
**FULL OUTER JOIN:(Und the position of null)**

Common rows that are present in both tables will be returned once based on matching condition. All other records from both the tables will be returned with Null values. While displaying output, query will be analyzed left to right and displays rows in left table first.

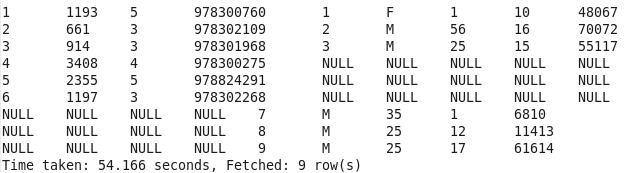
****

**Full-Outer-Join without where:**

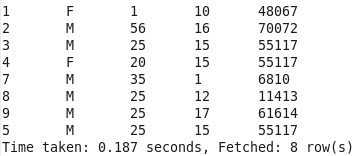
select \* from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid;

****

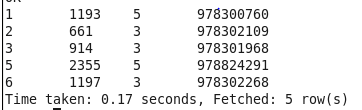
select \* from ratings\_join u FULL OUTER JOIN user\_join r on u.userid = r.userid;



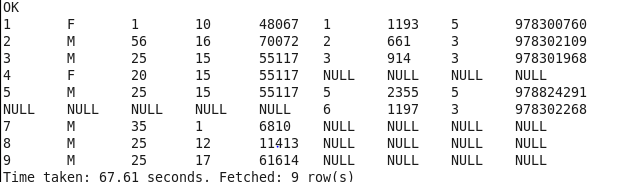
**user-Join1:**

****

**rating\_join1:**

****

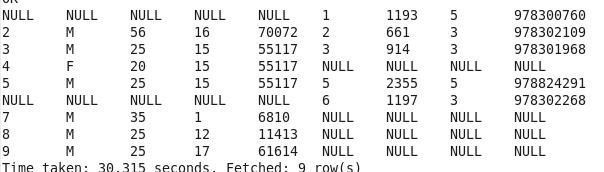
**Outer-Join Precedence Example:**

select \* from user\_join1 u FULL OUTER JOIN ratings\_join1 r on u.userid = r.userid; ****

In precedence, the key that is placed in "ON" clause from the left table is considered first and checked with other table. If a match is found in the other table the entire column is returned after joining.

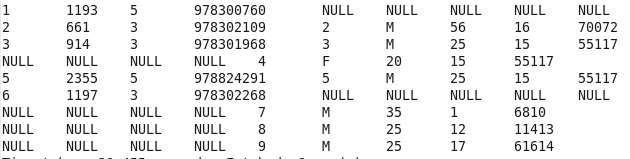
**After deleteing ID 1 from user\_join1:**

select \* from user\_join1 u FULL OUTER JOIN ratings\_join1 r on u.userid = r.userid;



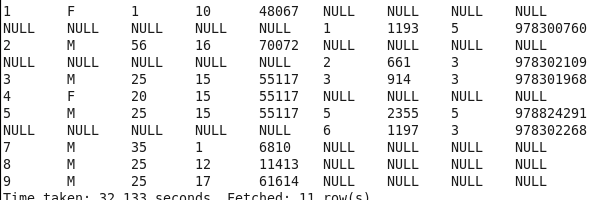
**Changing the position of table name:**

select \* from ratings\_join1 u FULL OUTER JOIN user\_join1 r on u.userid = r.userid;



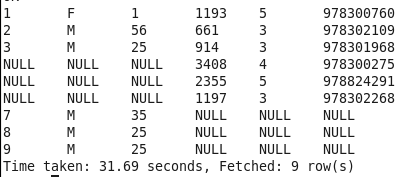
**Using AND inside ON clause:**

Null is display below ID 1 and 2 because it matches one condition **"u.userid = r.userid"** in AND clause.

select \* from user\_join1 u FULL OUTER JOIN ratings\_join1 r on ( u.userid = r.userid AND u.userid = r.rating );****

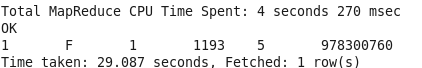
**Full-Outer-Join without where:**

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid;



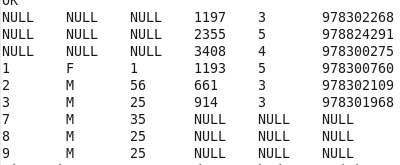
**FULL OUTER JOIN with ON and Where:**

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid where u.userid = 1;



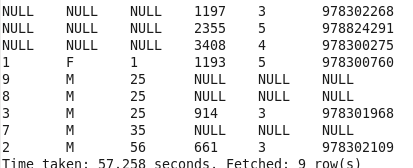
**Full Outer Join with ON and SORT-BY:**

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid SORT BY u.userid;



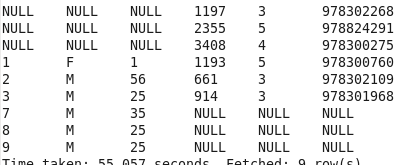
**Full outer Join - ON, Sort BY dual Column:**

select u.userid as uid,u.Gender as gen, u.age,r.movieid, r.rating, r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid SORT BY gen,u.age;



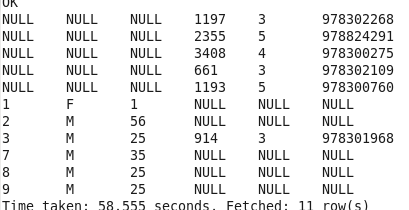
**Full Outer Join - ON, Order By:**

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid ORDER BY u.userid;



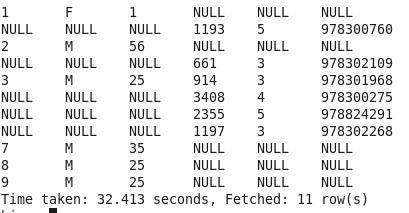
**Full Outer Join - ON, Order By:**

select u.userid,u.Gender,u.age,r.movieid, r.rating ,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on ( u.userid = r.userid AND u.userid = r.rating ) ORDER BY u.userid;

  
**Full Outer Join - ON:**

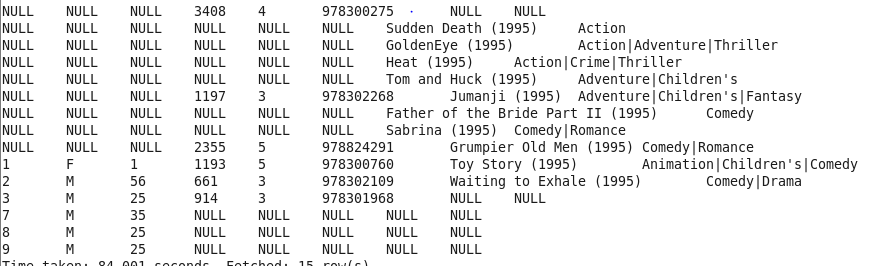
The rows that satisfies single condition in "ON" clause appear first while displaying with NULL first (second and fourth row in the output below)

select u.userid,u.Gender,u.age,r.movieid, r.rating ,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on ( u.userid = r.userid AND u.userid = r.rating );



**Two FULL OUTER JOIN clause which joins all three table:(Not Understood)**

select u.userid,u.Gender,u.age,r.movieid, r.rating, r.timestamp\_rating, m.title, m.genres from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid FULL OUTER JOIN movies\_join m on r.movieid = m.movieid ORDER BY u.userid,m.genres,r.movieid;



**Illegal:**

select u.userid as uid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid <= r.userid;

**#Invalid:** primarily because it is difficult to implement these kinds of joins in MapReduce. It turns out that Pig offers a cross product feature that makes it possible to implement this join, even though Pig’s native join feature doesn’t support it, either.

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on ( u.userid = r.userid OR u.userid = r.userid ) ORDER BY u.userid;

Hive does not currently support using OR between predicates in ON clauses.

select u.userid,u.Gender,u.age,r.movieid, r.rating,r.timestamp\_rating from user\_join u FULL OUTER JOIN ratings\_join r on u.userid = r.userid Group BY u.userid;

#GroupBY is not working here because no aggregate function is used.