INNER JOIN

An INNER JOIN returns only the rows that have matching values in both tables.

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

OR

```
SELECT columns
FROM table1
JOIN table2
ON table1.column = table2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

OR

```
SELECT columns
FROM table1 t1
INNER JOIN table2 t2
ON t1.column = t2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

OR

```
SELECT columns
FROM table1 t1
JOIN table2 t2
ON t1.column = t2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

Algorithm

```
For every Row in table1:
      For every Row in table2:
            If JOIN condition matches:
                  Put the merged row in resultant table
```

Time Complexity O(N x M)

Therefore, joins are expensive.

SELF JOIN

A self-join is a specific type of SQL join where a table is joined with itself. In other words, you treat a single table as if it were two separate tables, and you join the rows within that table based on a relationship defined by a common column.

```
SELECT t1.column1, t2.column2
FROM table_name t1
JOIN table_name t2 ON t1.common_column = t2.common_column
WHERE condition
GROUP BY columns
ORDER BY column;
```

STUDENTS

ID	Name	PeerReviewerID
1	A	5
2	B	2
3	C	4
4	D	3
5	E	1

LEFT JOIN

A LEFT JOIN returns all the rows from the left table and the matching rows from the right table. If there is no match in the right table, NULL values are returned.

```
SELECT columns
FROM table1
LEFT JOIN table2 ON table1.column = table2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

Algorithm for Left Join:

1) Initialize an empty result set or table to store the joined rows.

2) Iterate through each row in the left (first) table:

   A) Take the current row from the left table.

   B) For each row in the right (second) table:

      I) Check if there is a match between the common columns.

      II) If there is a match, create a new row by combining columns from both tables.

      III) Add the newly created row to the result set.

   C) If no match is found in the right table, create a new row by combining columns from the left table and adding NULL values for columns from the right table.

   D) Add the newly created row to the result set.

3) Continue this process for all rows in the left table.

4) The result set now contains all rows from the left table and matched rows from the right table (with NULL values for non-matching rows).

5) Return the result set as the result of the left join operation.

RIGHT JOIN

A right join, also known as a right outer join, is a type of join operation in SQL that combines rows from two tables based on a common column while including all the rows from the right (second) table and matching rows from the left (first) table. If there is no match in the left table, NULL values are included for columns from the left table.

```
SELECT columns
FROM table1
RIGHT JOIN table2 ON table1.column = table2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

Algorithm for Right Join:

1) Initialize an empty result set or table to store the joined rows.

2) Iterate through each row in the right (second) table:

   A) Take the current row from the right table.

   B) For each row in the left (first) table:

      I) Check if there is a match between the common columns.

      II) If there is a match, create a new row by combining columns from both tables.

      III) Add the newly created row to the result set.

   C) If no match is found in the left table, create a new row by combining columns from the right table and adding NULL values for columns from the left table.

   D) Add the newly created row to the result set.

3) Continue this process for all rows in the right table.

4) The result set now contains all rows from the right table and matched rows from the left table (with NULL values for non-matching rows).

5) Return the result set as the result of the right join operation.

FULL JOIN or FULL OUTER JOIN

A FULL JOIN returns all the rows when there is a match in either the left or the right table. If there is no match in one of the tables, NULL values are returned for columns from the non-matching table.

```
SELECT columns
FROM table1
FULL JOIN table2 ON table1.column = table2.column
WHERE condition
GROUP BY columns
ORDER BY column;
```

Algorithm for Full Outer Join:

1) Initialize an empty result set or table to store the joined rows.

2) Iterate through each row in the left (first) table:

   A) Take the current row from the left table.

   B) For each row in the right (second) table:

      I) Check if there is a match between the common columns.

      II) If there is a match, create a new row by combining columns from both tables.

      III) Add the newly created row to the result set.

   C) If no match is found in the right table, create a new row by combining columns from the left table and adding NULL values for columns from the right table.

   D) Add the newly created row to the result set.

3) Continue this process for all rows in the left table.

4) Iterate through each row in the right (second) table:

   A) Take the current row from the right table.

   B) For each row in the left (first) table:

      I) Check if there is a match between the common columns.

      II) If there is a match, we've already included it in the result set in step 2. Skip it.

   C) If no match is found in the left table, create a new row by combining columns from the right table and adding NULL values for columns from the left table.

   D) Add the newly created row to the result set.

5) Continue this process for all rows in the right table.

6) The result set now contains all rows from both the left and right tables, including matched rows (with combined columns) and unmatched rows (with NULL values for columns from the non-matching table).

7) Return the result set as the result of the full outer join operation.

CROSS JOIN

In SQL, a CROSS JOIN, also known as a Cartesian join or cross product, is a join operation that combines all rows from one table with all rows from another table, creating a Cartesian product of the two tables. Unlike other types of joins, there is no specific join condition specified in a CROSS JOIN, and it simply pairs each row from the first table with every row from the second table.

```
SELECT columns
FROM table1
CROSS JOIN table2
WHERE condition
GROUP BY columns
ORDER BY column;
```

INNER JOIN

If we want to fetch only those customers who have orders in our system.

```
SELECT COLUMNS
FROM CUSTOMER C
INNER JOIN ORDERS O
ON C.CUSTOMER_ID = O.CUSTOMER_ID;
```

OUTER JOIN

If we want to fetch customers whether they have any orders in our system or not.

If we want to fetch all the details of left table i.e. customer then we use LEFT JOIN.

```
SELECT COLUMNS
FROM CUSTOMER C
LEFT JOIN ORDERS O
ON C.CUSTOMER_ID = O.CUSTOMER_ID;
```

If we want to fetch all the details of right table i.e. orders then we use RIGHT JOIN.

```
SELECT COLUMNS
FROM CUSTOMER C
RIGHT JOIN ORDERS O
ON C.CUSTOMER_ID = O.CUSTOMER_ID;
```

USING

```
SELECT COLUMNS
FROM CUSTOMER C
INNER JOIN ORDERS O
ON C.CUSTOMER_ID = O.CUSTOMER_ID;

SELECT COLUMNS
FROM ORDERS O
INNER JOIN PRODUCTS P
ON O.ORDER_ID = P.ORDER_ID AND O.PRODUCT_ID = P.PRODUCT_ID;


SELECT COLUMNS
FROM CUSTOMER C
INNER JOIN ORDERS O
USING(CUSTOMER_ID);

SELECT COLUMNS
FROM ORDERS O
INNER JOIN PRODUCTS P
USING(ORDER_ID, PRODUCT_ID);
```

NATURAL JOIN

```
SELECT columns
FROM table1
NATURAL JOIN table2;
```

CROSS JOIN

```
SELECT columns
FROM table1
CROSS JOIN table2;
```