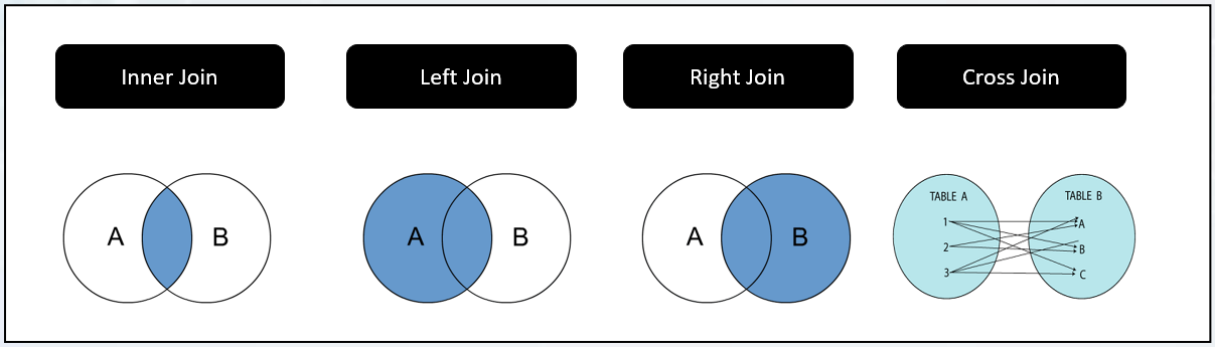
Summary – Day 7

Joins:

- Joins are used to combine the data from multiple tables in the database.
- We can join the tables from same / different databases.
- The data types and the values must be same for the columns of the two tables we are joining.

Types:

- Inner Join
- Left Join
- Right Join
- Full Outer Join
- Cross Join
- Self Join



Inner Join:

- The inner join clause compares each row from the first table with every row from the second table.
- If values from both rows satisfy the join condition, the inner join clause creates a new row whose column contains all columns of the two rows from both tables and includes this new row in the result set. In other words, the inner join clause includes only matching rows from both tables.

Left Join:

- The left join selects data starting from the left table. For each row in the left table, the left join compares with every row in the right table.
- If the values in the two rows satisfy the join condition, the left join clause creates a new row whose columns contain all columns of the rows in both tables and includes this row in the result set.
- If the values in the two rows are not matched, the left join clause still creates a new row whose columns contain columns of the row in the left table and NULL for columns of the row in the right table.
- In other words, the left join selects all data from the left table whether there are matching rows exist in the right table or not.
- In case there are no matching rows from the right table found, the left join uses NULLs for columns of the row from the right table in the result set.

Right Join:

- The right join clause is similar to the left join clause except that the treatment of left and right tables is reversed. The right join starts selecting data from the right table instead of the left table.
- The right join clause selects all rows from the right table and matches rows in the left table. If a row from the right table does not have matching rows from the left table, the column of the left table will have NULL in the final result set.

Full Outer Join:

- The FULL OUTER JOIN combines the results of both left and right outer joins and returns all (matched or unmatched) rows from the tables on both sides of the join clause.
- In MySQL, we don't have Full Outer Join keyword so we can use Union clause to do full outer join.
- We can write full outer join in below manner.
    - SELECT * FROM T1 LEFT JOIN T2 ON T1.C1 = T2.C2

      UNION

      SELECT * FROM T1 RIGHT JOIN T2 ON T1.C1 = T2.C2;

- Full outer join tis the combination of left and right joins.

Cross Join:

- Unlike the inner join, left join, and right join, the cross join clause does not have a join condition.
- The cross join makes a Cartesian product of rows from the joined tables. The cross join combines each row from the first table with every row from the right table to make the result set.
- Suppose the first table has n rows and the second table has m rows. The cross join that joins the tables will return nxm rows.
- The cross join is useful for generating planning data. For example, you can carry the sales planning by using the cross join of customers, products, and years.

```
SELECT column-lists
FROM table1
CROSS JOIN table2;
```

Self Join:

- A SELF JOIN is a join that is used to join a table with itself. In the previous sections, we have learned about the joining of the table with the other tables using different JOINS, such as INNER, LEFT, RIGHT, and CROSS JOIN. However, there is a need to combine data with other data in the same table itself. In that case, we use Self Join.
- We can perform Self Join using table aliases. The table aliases allow us not to use the same table name twice with a single statement. If we use the same table name more than one time in a single query without table aliases, it will throw an error.
- The table aliases enable us to use the temporary name of the table that we are going to use in the query. Let us understand the table aliases with the following explanation.

```
SELECT s1.col_name, s2.col_name...
FROM table1 s1, table1 s2
WHERE s1.common_col_name = s2.common_col_name;
```