Assignment - 5

⇒ Title:- SQL queries with join Operation.

⇒ Aim:- Write Select command using Join Operation to retrieve requested data from tables.

⇒ Objective:- To Study different join operation and Nested Queries.

⇒ Problem Statement:- Create tables with appropriate constraints like primary key, foreign key and write queries with Join Operations as per the batch wise.

⇒ Theory :-

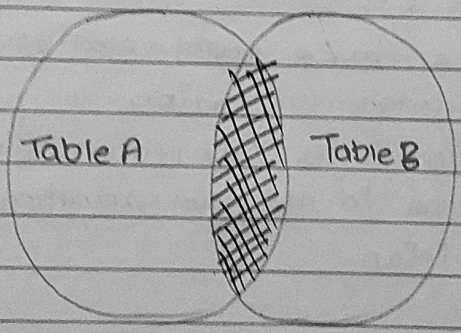
Join Operations :-
└ Join operations take two relations and return as a result another relation.
A join operation is a cartesian product which requires that tuples in the two relations match (under some condition.

- Join Condition :-
  └ defines which tuples in the two relations match and what attributes are present in the result of the join.
- Join type :-
  └ defines how tuples in each other relation (based on join).

Types of Join Operations :-

=> Inner join :-
Selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

Syntax :-
```
Select
table1.col1, tal1, cal2, tab2.col1, .... from
table1
Inner join
    table2 ON
table1.matching_col = table2.matching cal;
```



Eg:- Select StudentCourse.Course_ID, Student.nam
Student.age from Student
Inner join StudentCourse ON
Student.Roll_no = Studentcourse.Roll_no;

=>

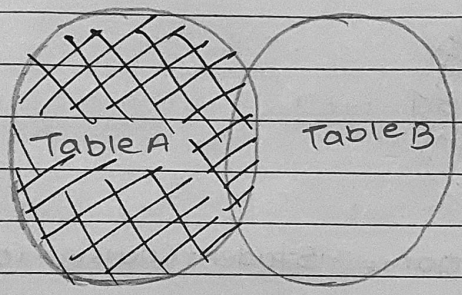
Course Id	Name	age
1	Abc	18
2	Yash	20
3	Yashvi	11
4	Veni	21

⇒ Left join :-

Returns all rows from the left table, along with matching rows from the right table. If there is no match, NULL values are returned from columns from right table. This also known as Left Outer Join.

└ Syntax :-

```
Select
tab1.col1, tab1.col2, tab2.col1 .....
from table1
Left Join table2
ON tab1. matching_col = tab2.matching_col;
```



Eg:- Select student.Name, StudentCourse.CouseID from Student
Left join Student Course ON Student+Course. ROLL_NO = Student . ROLL_NO .

⇒

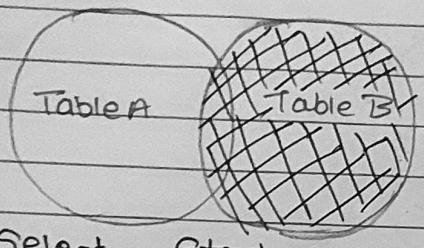
Name	Course ID
Yashi	1
Yashvi	2
Yash	3
Yashshvi	1
Meenaja	2
Rani	NULL
Renu	NULL

⇒ Right join :-

Returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. It is also known as Right outer join. If no matching row on left result-set will contain null.

└ Syntax :-

```
Select
tab1.col1, tab1.col2, col2, tab2.col1 ....
from table1
Right join table2
ON tab1.matching_col = tab2.matching_col;
```



Eg :- Select Student.name, StudentCourse.CourseID
from Student
Right join StudentCourse ON StudentCourse.Roll_NO = Student.Roll_NO;

⇒

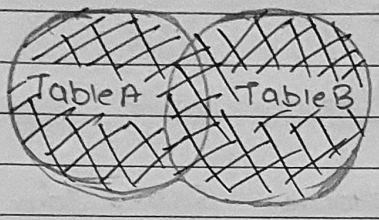
Name	Course_ID
Yashy	1
Yashi	2
Yashvi	3
Yash	4
Neelu	4
Neelam	1
NULL	1
NULL	2
NULL	3

⇒ Full join :-

Creates the result-set by combining results of both left join and right join. The result-set will contain all the rows from both tables.
For the rows for which there is no matching the result-set will contain NULL values. (full outer)

└ Syntax :-

```
select
tab1.col1, tab1.col2, tab2.col1 ....
from table1
Full join table2
ON tab1.matching_col = tab2.matching col;
```



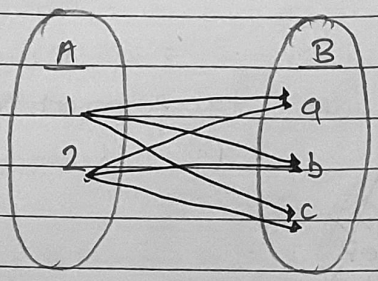
Eg:- Select Student.name, StudentCourse.CourseID from student Full join StudentCourse ON StudentCourse.Roll_No = Student.Roll_No;

⇒

Name	CourseID
Yashi	1
Yashvi	2
Yashy	2
Yash	3
Neely	1
Neelum	NULL
Renu	NULL
NULL	4
NULL	5
NULL	4

⇒ Cross join :-
Generates the cartesian product of two tables, meaning each row from the first table is paired with every row from second. It is used alongside where clause to filter results into meaningful pairs.

Syntax:-
```
Select * from tab1
    cross join tab2;
```



Eg:- Select * from customer cross join orders;

⇒

ID	name	age	order	amount
1	Yashi	15	101	10000
2	Yashvi	20	102	200000
3	yashu	19	103	1000
4	Yash	84	104	40000

- Cross join combines every row from tab1 with every row from tab2.
The result will contain n x n = n rows.

FAQ's :-

1] How do you handle ambiguous column when joining tables?

Ans] when joining two tables have cols with same name. This causes ambiguity in the query because the database engine can't tell which column youre refering to. Hence we can do the following :-

- Use table aliases :-
  - Assign short names & prefix the col with table name.

```
Eg:- Select a.id, a.name, b.salary
     from emp as a
     Join salary as b
     ON a.id = b.emp_id;
```

- Use fully qualified names:-
  - Specify the table name explicitly.

```
Eg:- select emp.id, emp.name, sal.salary
     from emp
     Join sal
     ON emp.id = sal.emp_id;
```

- Use the Using clause:-
  - This avoids ambiguity if col names are same as in both tables.

```
Eg:- Select id, name, salary
     from emp
     Join sal
     Using (emp_id);
```

2] What are the performance considerations when using different types of JOINS?

Ans] Different types of Joins :-

- INNER JOIN :-
  - Returns only matching rows.
  - Usually faster because it processes fewer as non-matching are discarded.
- LEFT JOIN / OUTER LEFT JOIN :-
  - Returns all rows from the left table & matching rows from right table.
- RIGHT / OUTER RIGHT JOIN :-
  - Returns all rows from the right & matching from left table.
- FULL OUTER JOIN :-
  - Returns all from both tables & NULL where not match.
- CROSS JOIN :-
  - Produces a Cartesian product.
- PERFORMANCE CONSIDERATIONS :-
  - Indexing ensures join col are indexed to improve lookup speed.
  - Apply where clauses before joins if possible.
  - Avoid unnecessary columns and rows.
  - Use tools like explain to see how the database engine processes the query.

3] What is the difference between ON and USING in joins?

Ans] • ON Clause :-
  - provides a condition explicity specifying

now columns should be matched.

Eg:- Select a.id, b.dept
from emp as a
Join dept as b
ON a.dept_id = b.id;

- USING Clause :-
  - Used when the col have the same name in both tables. Simplifies syntax and automatically eliminate duplicate columns.

Eg:- Select id, name
from emp
Join dept
using (id);

⇒ Conclusion:-

Thus, we have learned SQL Join operations with SQL operators thoroughly.