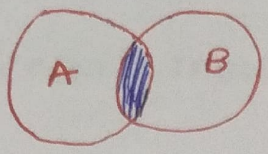
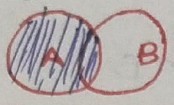
1. Inner Join = returns records that have matching values in both tables.
(A ∩ B)



Syntax:
```
SELECT column(s)
FROM tableA
INNER JOIN tableB
ON tableA.col_name = tableB.col_name;
```

2. Left Join = returns all records that from the left table, and the matched recorded from the right table.



Syntax:
```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name;
```
(A is dominant)

Eg:

stud_id	name
101	adam
102	bob
103	casey

stud_id	course
102	english
105	math
103	science
107	computer science

Result:

stud_id	name	course
101	adam	null
102	bob	english
103	casey	science

3. Right Join

Syntax:
```
SELECT column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name;
```
(B is dominant)

4. Full Join : Left Join ∪ Right Join.
5. Left Exclusive Join where b.id is null.
6. Right Exclusive Join where a.id is null.

7. Self Join

Syntax:

```
SELECT column(s)
FROM table as a
JOIN table as b
ON a.col_name = b.col_name;
```

8. Union

Syntax:

```
SELECT column(s) FROM tableA
UNION  ← Remove duplicate
SELECT column(s) FROM tableB
```

9. UNION ALL

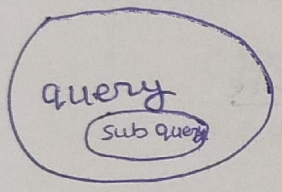
→ same

ONLY CONTAINS DUPLICATE

UNION ALL

SQL Sub Queries (nested query)

It involves 2 select statements



Syntax:

```
SELECT column(s)
FROM table_name
WHERE col_name operator
(subquery);
```

↑ normal but nested

Eg:

```
SELECT name, marks
FROM student
WHERE marks > (SELECT AVG (marks) FROM student);
```

Eg:

```
SELECT MAX(marks)
FROM student
WHERE city = "Mumbai";
```

VIEWS: A view is a virtual table based on the result set of an SQL statement.

```
CREATE VIEW view1 AS
SELECT rollno, name FROM student,
SELECT * FROM view1;
```