CLAUSES

SQL CLAUSES

SQL BETWEEN Clause

SELECT column1, column2....columnN FROM table_EID WHERE column_EID BETWEEN val-1 AND val-2;

SQL IN Clause

SELECT column1, column2....columnN

FROM table_EID

WHERE column_EID IN (Val1, Val2... Valn);

SQL Like Clause

SELECT column1, column2....columnN FROM table_EID WHERE column_EID LIKE { PATTERN}

SQL COUNT Clause

SELECT COUNT(column_EID) FROM table_EID WHERE CONDITION;

SQL DISTINCT Clause

SELECT DISTINCT (column) FROM table_EID;

SQL CLAUSES

SQL ORDER BY Clause

```
SELECT column1, column2....columnN
FROM   table_EID
WHERE  CONDITION
ORDER BY column_EID {ASC|DESC};
```

SQL GROUP BY Clause

its for grouping data based on an aggrigation funciton

```
SELECT SUM(column_EID)
FROM   table_EID
WHERE  CONDITION
GROUP BY column_EID;
```

SQL HAVING Clause

```
SELECT SUM(column_EID)
FROM table_EID
WHERE  CONDITION GROUP BY column_EID
HAVING (arithematic function condition);
```

Is it mantatory to use order by with group by: NO
Can where and having use interchangeably: NO
is it mantatory to use groupby along with having: NO
is it mantatory to use group by if i am using having: yes

order of clauses:

1. where
2. group by
3. having
4. order by

ASSIGNMENT – 4

In the EMP table display :

CITY WISE COUNT OF EMPLOYEES ARRANGED IN DESCENDING ORDFER

DETAILS OF THE EMPLOYEES WHO DOES NOT HAVE AN ACCOUNT ON YAHOO DOMAIN

From the Emp_Sal table display:

DESIGNATION WISE TOTAL COST AND NUMBER OF MEMBERS ARRANGED IN DESCENDING ORDER OF THE TOTAL COST

JOINS

SQL Joins

joins are required when we need to retrive or display data from more than 1 table

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

SQL Join Types:

- INNER JOIN: returns rows when there is a match in both tables.
- LEFT JOIN: returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: returns rows when there is a match in one of the tables.
- CARTESIAN JOIN: returns the cartesian product of the sets of records from the two or more joined tables.
- SELF JOIN: is used to join a table to itself, as if the table were two tables, temporarily renaming at least one table in the SQL statement.

INNER JOIN

The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN..

```
SELECT table1.column1, table2.column2...

FROM table1

INNER JOIN table2

ON table1.common_filed = table2.common_field;
```

LEFT JOIN

The SQL Left Join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching.

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_filed = table2.common_field;
```

RIGHT JOIN

The SQL Right Join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching.

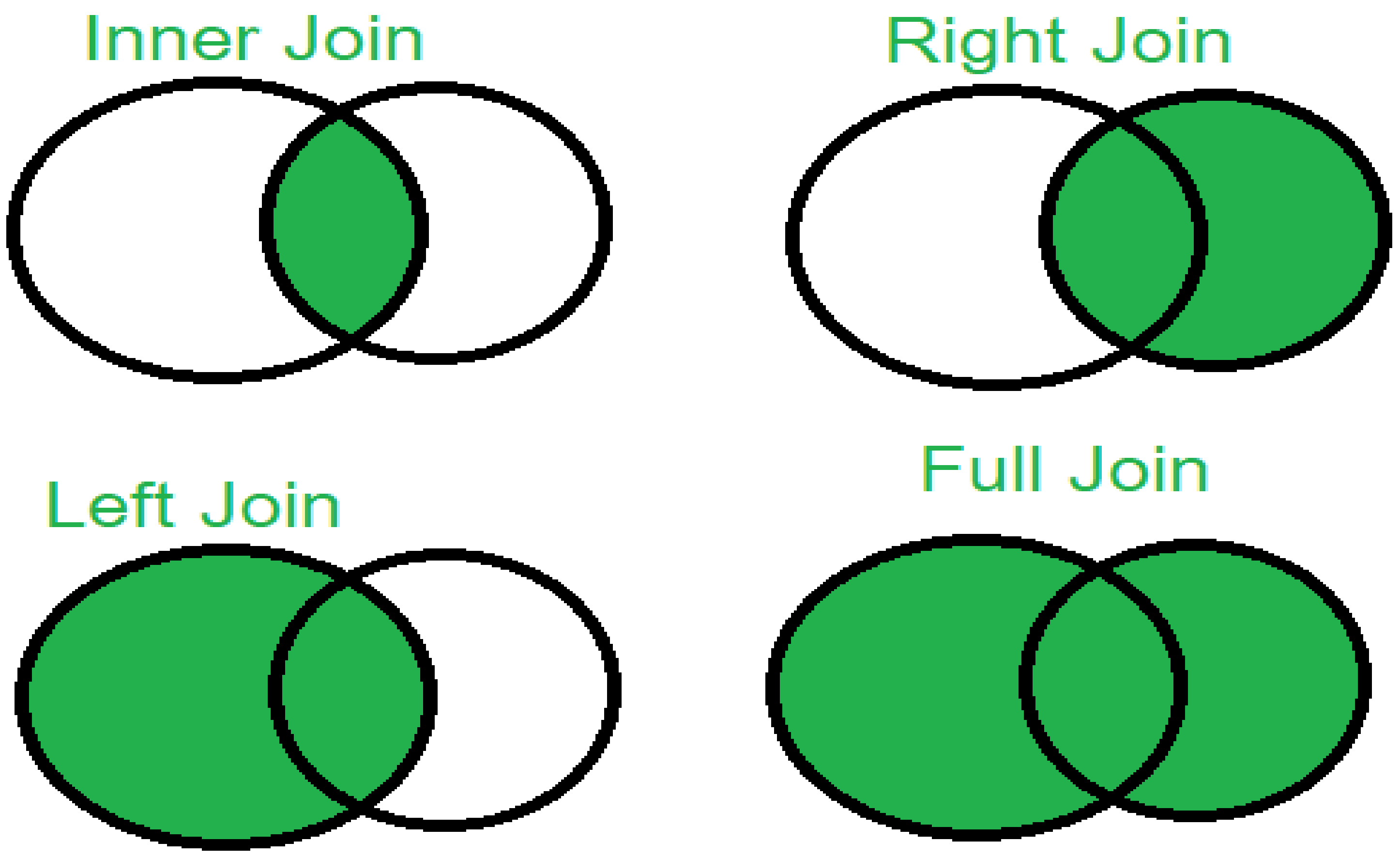
```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_filed = table2.common_field;
```

FULL JOIN

The SQL FULL JOIN combines the results of both left and right outer joins.

```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_filed = table2.common_field;
```

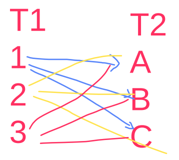
JOINS



CARTESIAN JOIN

- The CARTESIAN JOIN or CROSS JOIN returns the cartesian product of the sets of records from the two or more joined tables.
- It produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN.
- If WHERE clause is used with CROSS JOIN, it functions like an INNER JOIN.

```
SELECT table1.column1, table2.column2...
FROM table1
CROSS JOIN table2
```



```
SELECT table1.column1, table2.column2...
FROM table1
CROSS JOIN table2
WHERE table1.common_filed = table2.common_field;
```

Cross join with where condition of common elements gives inner join

SELF JOIN

The SQL SELF JOIN is used to join a table to itself, as if the table were two tables, temporarily renaming at least one table in the SQL statement.

```
SELECT a.column_EID, b.column_EID...
FROM table1 a, table1 b
 WHERE a.common_filed = b.common_field;
```

SELF JOIN

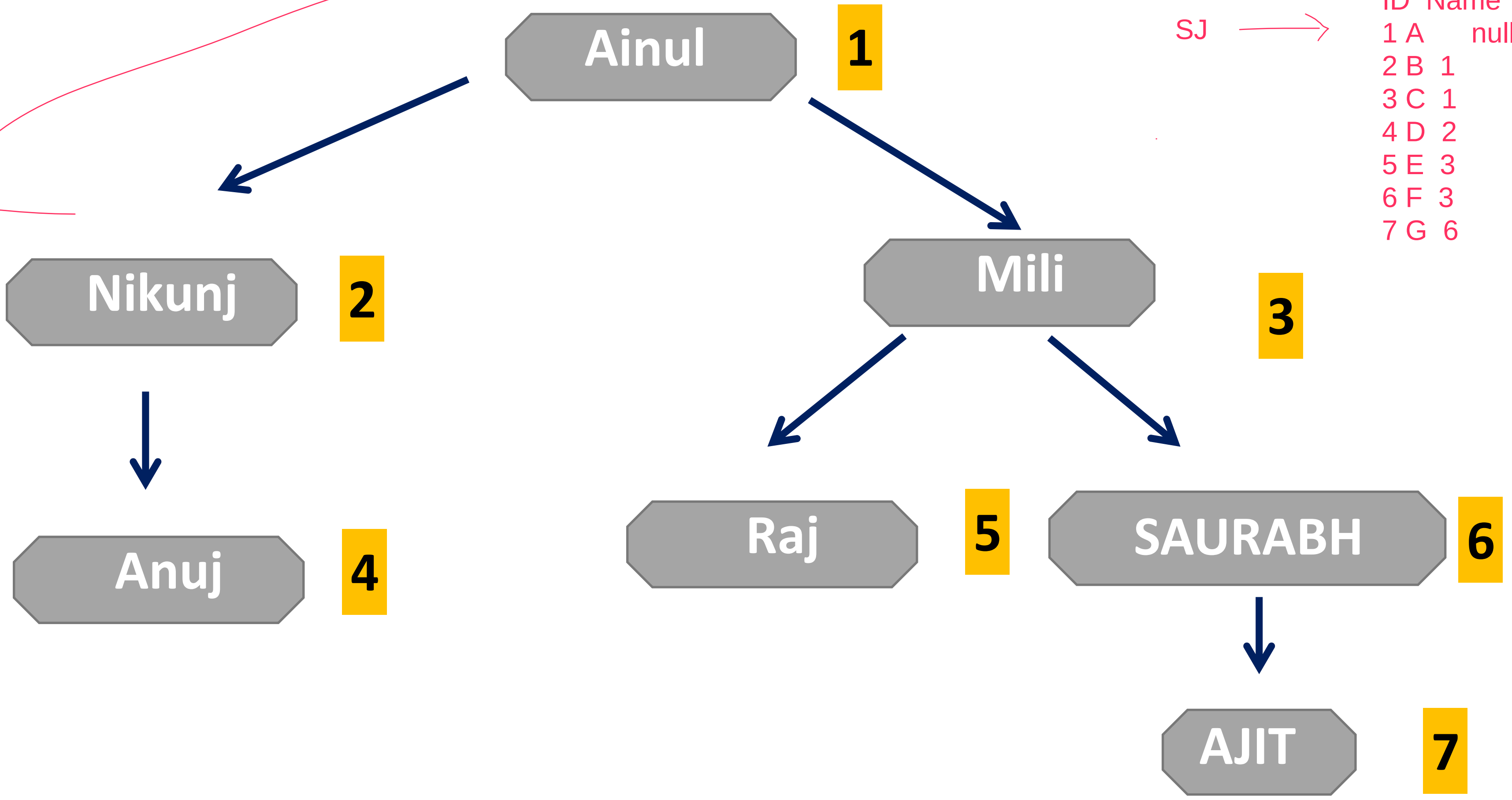
```
select s1.id,s1.name,s2.name as 'Boss name'
from sj  s1
left join sj s2
on s1.bossid=s2.id
```

result

ID	name	Boss name
1	A	null
2	B	A
3	c	A
4	d	B
5	e	c
6	f	c
7	g	f

SJ →

ID	Name	Bossid
1	A	null
2	B	1
3	C	1
4	D	2
5	E	3
6	F	3
7	G	6


Ainul
1
Nikunj
2
Mili
3
Anuj
4
Raj
5
SAURABH
6
AJIT
7