By → Sapna Malik
PGT (CS)

Difference Between ORDER BY and GROUP BY.

ORDER BY	GROUP BY.
1.) Used to display information either in ascending or descending order	1.) Used to group data on the basis of particular column.
2.) Used for sorting data in the table.	2.) Used for grouping of data.
3.) Not mandatory to use aggregate functions with order by.	3.) Mandatory to use aggregate functions in order to use GROUP BY command.
4.) Eg. → select * from student ORDER BY marks;	4.) Eg. → select name, sum(marks) from student GROUP BY name;

Difference between WHERE and HAVING

WHERE	HAVING.
1.) Used for selecting specific row/ rows based on the condition.	1.) HAVING clause can't be used without GROUP BY command.
2.) It can be used without GROUP BY.	2.) HAVING clause selects rows after grouping.
3.) WHERE clause can't contain aggregate functions.	3.) Contains aggregate functions.
4.) Select * from student WHERE marks > 80;	4.) eg.→ select name, max(marks) from student GROUPBY name having max(marks) > 70;

DELTA Pg No.
Date / /

GROUP BY

Syntax:- Select <column name>, aggregate func^n
from <tablename> GROUP BY <columnname>;

Example:- 1.) Select name, marks from student GROUP BY name;

2.) Select name, marks from student where marks >= 70
GROUP BY name;

3.) Select name, sum(marks) from student GROUP BY name;

4.) " " , " " " " having sum(mark
>75;

5.) Select name, count(city) from student group by name;

6.) " " " " " " " "
having count(city) > 1;

DELTA Pg No.
Date / /

SQL JOINS.

⇒ These ~~sql~~ SQL joins are used to join rows from two or more tables.

1.) Cartesian Product. → used to join rows from one table to another.

⇒ Also called cross-join or cross-product.

⇒ It's a binary operation and represented as $\boxed{X}$

e.g. $A = \{1, 2, 3\}$

$B = \{a, b, c\}$

$C = A \times B$
↳ cartesian product.

A	B
1	a
2	b
3	c

$= C = A \times B$

(1, a)
(1, b)
(1, c)
(2, a)
(2, b)
(2, c)
(3, a)
(3, b)
(3, c)

Eg.⇒ We have 2 tables named stud and games.

Stud.

Rollno.	name.
1	A
2	B
3	C.

games.

gno	gname
11	Tennis
12	Hockey.

Syntax:- select *. from table1, table2;

eg.→ select * from stud, games;

---

② Equi-Join:- It uses $\boxed{=}$ operator and used to establish the relationship between two tables on the basis of primary key and foreign key concept.

Syntax. Select * from table1, table2 where
table1.primary key = table2. foreign key;
Or table1. columnname = table2. columnname;

Example.

Primary Key ↓ Student

RollNo.	Name	City	Marks
1	A	Delhi	70
2	B	goa	20
3	C	Vizag	30
4	D	MUM	50

Primary Key ↓ BOOKS Foreign Key ↓

BookID	RollNo.	BookName
10	1	CS
20	2	Chem.
30	4	PHY.

```
Select * from student, books where
        student.rollno = books.rollno ;
```

We can use alias name as well, ~~et~~ lets say s for student table and b for books table. So, the same query can be written as.

$\Downarrow$

```
Select * from student s, books b where
        s.rollno = b.rollno;
```

Natural JOIN:- It is equivalent to Equi-JOIN only difference it eliminates the duplicacy of the same column name used for primary and foreign key.

Syntex:- select * from table1 NATURAL JOIN table2;

eg.→ select * from student NATURAL JOIN Books;