SQL CLAUSES

SQL CLAUSES

- Select
- From
- Where
- Group by
- Having
- Order by

ORDER OF EXECUTION

- From
- Where
- Group by
- Having
- Order by
- Select

WHERE CLAUSE

The SQL **WHERE** clause is used to filter the results and apply conditions in a select, insert, update or delete statement.

The following operators can be performed by using **Where** clause, they are:

=,<,<=,>,>=, Between, Like ,and ,IN,OR.

Sample Relation: Students_info

```
SQL> Select * from students_info;

      S_ID S_NAME     DEPT  ADDRESS         GENDER       CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       201 Rohit      IT    Chennai         Male           72.5 15-MAY-00
       206 Karthick   CSE   Chennai         Male           55.7 12-JAN-01
       213 Lakshmi    IT    Madurai         Female         82.5 25-JUL-00
       211 Rohit      CSE   Chennai         Male           62.4 04-MAY-99
       207 Haritha    ECE   CBE             Female         92.5 17-JAN-01
       215 Rohini     EEE   Madurai         Female         57.5 01-MAY-00
       219            EEE   Salem                          57.5 11-MAY-00

7 rows selected.
```

ORDER BY CLAUSE

- The **ORDER BY** clause is used in a **SELECT** statement to sort results either in ascending or descending order.
- ORDER BY is used to sort the records while retrieving.
  - The data stored in the table will not be sorted
  - BY default the order is **ASCENDING.**
- Oracle sorts query results in **ascending order by default.**

ORDER BY Example[ASC]

Display the details of student Sort the result based on their ID in ascending order

```
SELECT *
FROM students_info
ORDER BY S_ID ASC;
```

(OR)

```
SELECT *
FROM students_info
ORDER BY S_ID;
```

ORDER BY Example[ASC]

```
SQL> SELECT *
  2  FROM students_info
  3  ORDER BY S_ID ASC;

      S_ID S_NAME     DEPT  ADDRESS         GENDER        CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       201 Rohit      IT    Chennai         Male           72.5 15-MAY-00
       206 Karthick   CSE   Chennai         Male           55.7 12-JAN-01
       207 Haritha    ECE   CBE             Female         92.5 17-JAN-01
       211 Rohit      CSE   Chennai         Male           62.4 04-MAY-99
       213 Lakshmi    IT    Madurai         Female         82.5 25-JUL-00
       215 Rohini     EEE   Madurai         Female         57.5 01-MAY-00
       219            EEE   Salem                          57.5 11-MAY-00

7 rows selected.

SQL> SELECT *
  2  FROM students_info
  3  ORDER BY S_ID;

      S_ID S_NAME     DEPT  ADDRESS         GENDER        CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       201 Rohit      IT    Chennai         Male           72.5 15-MAY-00
       206 Karthick   CSE   Chennai         Male           55.7 12-JAN-01
       207 Haritha    ECE   CBE             Female         92.5 17-JAN-01
       211 Rohit      CSE   Chennai         Male           62.4 04-MAY-99
       213 Lakshmi    IT    Madurai         Female         82.5 25-JUL-00
       215 Rohini     EEE   Madurai         Female         57.5 01-MAY-00
       219            EEE   Salem                          57.5 11-MAY-00

7 rows selected.
```

ORDER BY Example[DESC]

By default, the ORDER BY Clause sorts data in **ascending order**.

If we want to sort the data in descending order, we must explicitly specify.

```
SELECT *
FROM students_info
ORDER BY S_ID DESC;
```

ORDER BY Example[DESC]

```
SQL> SELECT *
  2  FROM students_info
  3  ORDER BY S_ID DESC;

      S_ID S_NAME     DEPT  ADDRESS         GENDER       CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       219            EEE   Salem                        57.5 11-MAY-00
       215 Rohini     EEE   Madurai         Female       57.5 01-MAY-00
       213 Lakshmi    IT    Madurai         Female       82.5 25-JUL-00
       211 Rohit      CSE   Chennai         Male         62.4 04-MAY-99
       207 Haritha    ECE   CBE             Female       92.5 17-JAN-01
       206 Karthick   CSE   Chennai         Male         55.7 12-JAN-01
       201 Rohit      IT    Chennai         Male         72.5 15-MAY-00

7 rows selected.
```

Example [ASC AND DESC]

Display the details of students, sort the result based on S_ID in descending order and CIA_I marks in Ascending order.

```
SELECT *
 FROM students_info
 ORDER BY S_ID DESC,CIA_I ASC;
```

Example [ASC AND DESC]

```
SQL> SELECT *
  2   FROM students_info
  3   ORDER BY S_ID DESC,CIA_I ASC;

      S_ID S_NAME     DEPT  ADDRESS         GENDER        CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       219            EEE   Salem                          57.5 11-MAY-00
       215 Rohini     EEE   Madurai         Female         57.5 01-MAY-00
       213 Lakshmi    IT    Madurai         Female         82.5 25-JUL-00
       211 Rohit      CSE   Chennai         Male           62.4 04-MAY-99
       207 Haritha    ECE   CBE             Female         92.5 17-JAN-01
       206 Karthick   CSE   Chennai         Male           55.7 12-JAN-01
       201 Rohit      IT    Chennai         Male           72.5 15-MAY-00

7 rows selected.
```

ORDER BY[COLUMN POSITION]

We can represent the columns in the ORDER BY clause by specifying the position of a column in the SELECT list, instead of writing the column name.

```
SELECT *
FROM students_info
ORDER BY 1 DESC,4 ASC;
```

ORDER BY [COLUMN POSITION]

```
SQL> SELECT *
  2  FROM students_info
  3  ORDER BY 1 DESC,4 ASC;

      S_ID S_NAME     DEPT  ADDRESS         GENDER        CIA_I DOB
---------- ---------- ----- --------------- -------- ---------- ---------
       219            EEE   Salem                          57.5 11-MAY-00
       215 Rohini     EEE   Madurai         Female         57.5 01-MAY-00
       213 Lakshmi    IT    Madurai         Female         82.5 25-JUL-00
       211 Rohit      CSE   Chennai         Male           62.4 04-MAY-99
       207 Haritha    ECE   CBE             Female         92.5 17-JAN-01
       206 Karthick   CSE   Chennai         Male           55.7 12-JAN-01
       201 Rohit      IT    Chennai         Male           72.5 15-MAY-00

7 rows selected.
```

ORDER BY

If we want both columns in descending ,we need to specify :

```
SELECT *
FROM students_info
ORDER BY S_ID DESC, CIA_I DESC;
```

GROUP BY

Why Group data?

- **Grouping data is the process of combining columns with duplicate values in a logical order.**
- For example, a database may contain information about employees; many employees live in different cities, while some employees live in the same city. we may want to execute a query that shows employee information for each particular city. We are grouping employee information by city, and a summarized report is created.
- Grouping data is accomplished through the use of the **GROUP BY** clause of a SELECT statement (query).

GROUP BY

The position of the GROUP BY clause in a query is as follows:

- SELECT
- FROM
- WHERE
- GROUP BY
- ORDER BY

Syntax:

SELECT COLUMN1, COLUMN2 FROM TABLE1, TABLE2

WHERE CONDITIONS

GROUP BY COLUMN1, COLUMN2

ORDER BY COLUMN1, COLUMN2;

GROUP BY EXAMPLE

Question 1:

Display the number of male and female student.

Query:

```
SELECT COUNT(S_id)
FROM STUDENTS_INFO
GROUP BY GENDER;
```

(OR)

```
SELECT COUNT(S_id),GENDER
FROM STUDENTS_INFO
GROUP BY GENDER;
```

GROUP BY EXAMPLE

Question 1: output

```
SQL> SELECT COUNT(S_id)
  2  FROM STUDENTS_INFO
  3  GROUP BY GENDER;

COUNT(S_ID)
-----------
          1
          3
          3

SQL> SELECT COUNT(S_id),GENDER
  2  FROM STUDENTS_INFO
  3  GROUP BY GENDER;

COUNT(S_ID) GENDER
----------- ---------
          1
          3 Male
          3 Female
```

GROUP BY EXAMPLE

Question 2:

Find the total CIA_I marks obtained by the students based on gender.

Query:

```
SELECT  SUM(CIA_I),GENDER
FROM STUDENTS_INFO
GROUP BY GENDER;
```

```
SQL> SELECT   SUM(CIA_I),GENDER
  2   FROM STUDENTS_INFO
  3   GROUP BY GENDER;

SUM(CIA_I) GENDER
---------- --------
      57.5
     190.6 Male
     232.5 Female
```

GROUP BY EXAMPLE

Question 3:

Find the total CIA_I marks obtained by the students based on gender. Exclude the null Value from the result.

Query:

SELECT SUM(CIA_I),GENDER
FROM STUDENTS_INFO
WHERE GENDER IS NOT NULL
GROUP BY GENDER;

```
SQL> SELECT  SUM<CIA_I>,GENDER
  2  FROM STUDENTS_INFO
  3  WHERE GENDER IS NOT NULL
  4  GROUP BY GENDER;

SUM<CIA_I> GENDER
---------- ---------
     190.6 Male
     232.5 Female
```

GROUP BY EXAMPLE

Question 4:

Find number of students who born in SAME MONTH.

Query:.

SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))

FROM STUDENTS_INFO

GROUP BY EXTRACT(MONTH FROM DOB);

```
SQL> SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))
  2  FROM STUDENTS_INFO
  3  GROUP BY EXTRACT(MONTH FROM DOB);

EXTRACT(MONTHFROMDOB) COUNT(DISTINCT(S_ID))
--------------------- -----------------------
                    1                       2
                    5                       4
                    7                       1
```

GROUP BY EXAMPLE

Question 5:

Find number of students who born in SAME MONTH.

Display the result in decreasing order of number of students.

Query:.

SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))

FROM STUDENTS_INFO

GROUP BY EXTRACT(MONTH FROM DOB)

ORDER BY COUNT(DISTINCT(S_ID)) DESC;

```
SQL> SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))
  2  FROM STUDENTS_INFO
  3  GROUP BY EXTRACT(MONTH FROM DOB)
  4  ORDER BY COUNT(DISTINCT(S_ID)) DESC;

EXTRACT(MONTHFROMDOB) COUNT(DISTINCT(S_ID))
--------------------- -----------------------
                    5                       4
                    1                       2
                    7                       1
```

GROUP BY EXAMPLE

Question 6:

Find number of students who born in SAME MONTH. Month should be either may or july. display the result in decreasing order of number of students.

Query:.

```
SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))
FROM STUDENTS_INFO
WHERE EXTRACT(MONTH FROM DOB) IN(5,7)
GROUP BY EXTRACT(MONTH FROM DOB)
ORDER BY COUNT(DISTINCT(S_ID)) DESC;
```

```
SQL> SELECT EXTRACT(MONTH FROM DOB),COUNT(DISTINCT(S_ID))
  2  FROM STUDENTS_INFO
  3  WHERE EXTRACT(MONTH FROM DOB) IN(5,7)
  4  GROUP BY EXTRACT(MONTH FROM DOB)
  5  ORDER BY COUNT(DISTINCT(S_ID)) DESC;

EXTRACT(MONTHFROMDOB) COUNT(DISTINCT(S_ID))
--------------------- ---------------------
                    5                     4
                    7                     1
```

HAVING CLAUSE

- SQL HAVING clause specifies a search condition for a group or **an aggregate.**
- HAVING is usually used in a GROUP BY clause.

How a HAVING clause works?

- The select clause specifies the columns.
- The from clause supplies a set of potential rows for the result.
- The where clause gives a filter for these potential rows.
- The group by clause divide the rows in a table into smaller groups.
- The having clause gives a filter for these group rows.

HAVING CLAUSE

EXAMPLE:

Write a query to get the years in which more than 2 students born in same Year.

Error:

SELECT EXTRACT(YEAR from DOB)

FROM STUDENTS_INFO

WHERE COUNT(S_ID)>2;

```
SQL> SELECT EXTRACT(YEAR from DOB)
  2  FROM STUDENTS_INFO
  3  WHERE COUNT(S_ID)>2;
WHERE COUNT(S_ID)>2
      *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

HAVING CLAUSE

SELECT EXTRACT(YEAR from DOB)

FROM STUDENTS_INFO

GROUP BY EXTRACT(YEAR from DOB)

HAVING COUNT(S_ID)>2;

```
SQL> SELECT EXTRACT(YEAR from DOB)
  2  FROM STUDENTS_INFO
  3  GROUP  BY EXTRACT(YEAR from DOB)
  4  HAVING COUNT(S_ID)>2;

EXTRACT(YEARFROMDOB)
--------------------
                2000
```

HAVING CLAUSE

QUESTION 1:

Display the Department and Average CIA_I marks obtained by the student whose Average CIA_I mark is greater than 60.

```
SELECT dept, AVG(CIA_I)
FROM STUDENTS_INFO
GROUP BY dept
HAVING AVG(CIA_I)>60;
```

```
SQL> SELECT DEPT,AVG(CIA_I)
  2  FROM STUDENTS_INFO
  3  GROUP BY DEPT
  4  HAVING AVG(CIA_I)>60;

DEPT   AVG(CIA_I)
------ -----------
IT            77.5
ECE           92.5
```

HAVING CLAUSE

Question 2:

Display the address and number of students with more than a student from Same city and their total CIA_I (sum of all CIA_I) is not less than .

```
SELECT ADDRESS,COUNT(S_ID)
FROM STUDENTS_INFO
GROUP BY ADDRESS
HAVING SUM(CIA_I)>150;
```

```
SQL> SELECT ADDRESS,COUNT(S_ID)
  2  FROM STUDENTS_INFO
  3  GROUP BY ADDRESS
  4  HAVING SUM(CIA_I)>150;

ADDRESS            COUNT(S_ID)
------------------ -----------
Chennai                      3
```

HAVING CLAUSE

Question :

and Average CIA_I marks obtained by the student

whose department Display the Department name is either ECE or IT and Average CIA_I mark is

greater than 60.sort the result in **decreasing order of Average CIA_I.**

```
SELECT dept,AVG(CIA_I)
 FROM STUDENTS_INFO
 WHERE dept IN('ECE','IT')
 GROUP BY dept
 HAVING AVG(CIA_I)>60
 ORDER BY AVG(CIA_I) DESC;
```

HAVING CLAUSE

```
SQL> SELECT dept,AVG(CIA_I)
  2  FROM STUDENTS_INFO
  3  WHERE dept IN('ECE','IT')
  4  GROUP BY dept
  5  HAVING AVG(CIA_I)>60
  6  ORDER BY AVG(CIA_I) DESC;

DEPT  AVG(CIA_I)
----- -----------
ECE          92.5
IT           77.5
```

DIFFERENCE BETWEEN **WHERE AND HAVING**

WHERE	**HAVING**
WHERE CLAUSE **can be used without the GROUP BY clause.**	The **HAVING** clause **cannot be used without the GROUP BY** clause.
The **WHERE** clause selects rows ***before*** grouping.	The **HAVING** clause selects rows ***after*** grouping.
The **WHERE** clause ***cannot*** **contain aggregate functions.**	The **HAVING** clause ***can*** **contain aggregate functions.**
Example: SELECT * FROM STUDENTS_INFO WHERE CIA_I>60;	**Example:** SELECT COUNT(S_ID),GENDER FROM STUDENTS_INFO WHERE CIA_I>60 GROUP BY GENDER HAVING COUNT(S_ID)>1;