CLAUSES IN SQL: We can add these to a query for adding additional options like filtering the records, sorting records and grouping the records with in a table. These clauses contains the following clauses are,

WHERE: This clause is used for filter or restricts the records from the table.

Ex: SELECT * FROM EMP WHERE SAL=10000

TOP (n) CLAUSE: This clause is used to fetch a top n number of records from a table.

Ex: SELECT TOP (3) * FROM EMP

Ex: UPDATE TOP (3) EMP SET ENAME='SAI'

Ex: DELETE TOP (3) FROM EMP

ORDER BY: The order by clause is used to sort or arrange the data in ascending or descending order with in table. By default order by clause arrange or sort the data in ascending order only.

- If we want to arrange the records in a descending order then we use Desc keyword.
- We can apply order by clause on integer and string columns.

Ex: SELECT * FROM EMP ORDER BY EID (For Ascending Order)

Ex: SELECT * FROM EMP ORDER BY ENAME DESC (For Descending Order)

EX: SELECT Empid, EmpName, Salary, salary*12 AS "AnnSalary"

FROM Employee ORDER BY "AnnSalary"

Note : ORDER BY is only clause that can refer to the column Alias

EX:

SELECT Empid, EmpName, Salary, salary*12 AS "AnnSalary"

FROM Employee

ORDER BY 4 DESC

EX:

SELECT Empid, EmpName, Salary, salary*12 AS "AnnSalary"

FROM Employee

ORDER BY 7 DESC

EX:

SELECT *, 'madhu'

FROM Employee

ORDER BY 8 DESC

-- can I order by on the column name which is not specifed in the select clause

EX:

SELECT Empid, EmpName, EmpDesignation

FROM employee

ORDER BY salary

GROUP BY: Group by clause will use for to arrange similar data into groups. when we apply group by clause in the query then we use group functions like count(),sum(),max(),min(),avg().

If we use group by clause in the query, first the data in the table will be divided into different groups based on the columns and then execute the group function on each group to get the result.

Ex1: WAQ to find out the number of employees working in the organization

Sol: SELECT COUNT (*) FROM EMP

Ex2: WAQ to find out the number of employees working in each group in the organization.

Sol: SELECT DEPT, COUNT=COUNT (*) FROM EMP GROUP BY DEPT

Ex3: WAQ to find out the total salary of each department in the organization

Sol: SELECT DEPT, TOTALSALARY=SUM (SALARY) FROM EMP GROUP BY DEPT (Like this we can find max, min, avg salary in the organization)

HAVING CLAUSE: Having clause is also used for filtering and restricting the records in a table just like where clause.

Ex: WAQ to find out the number of employees in each department only if the count is greater than 3

Sol: SELECT DEPT, COUNT=COUNT (*) FROM EMP GROUP BY DEPT HAVING COUNT (*) >3

Differences Between WHERE and HAVING Clause:

WHERE	HAVING
WHERE clause is used to filter and restrict the records before grouping	HAVING clause is used to filter and restrict the records after grouping
If restriction column associated with A aggregative function then we cannot use WHERE clause there	But we can use HAVING clause at this situations
WHERE clause can apply without group by clause	HAVING clause cannot be applied without a group by clause
WHERE clause can be used for restricting	Where as HAVING clause is used along with group

individual rows	by clause to filter or restrict groups

ROLLUP:-

A rollup is an extension to the group by clause used to calculate and return subtotals and grant total as rows of the query efficiently.

- These additional rows are the rows that would be created by the two union select portions of a pure sql solution.
- It is a group by operation and is used to produce subtotals at any level of the aggregation.
- It also calculates a grand total.
- ROLLUP is a simple extension to the GROUP BY clause, so its syntax is extremely easy to use.
- The ROLLUP extension is highly efficient, adding minimal overhead to a query.
- The total is based on a one dimensional data hierarchy of grouped information.
- If "n" is the number of columns listed in the ROLLUP, there will be n+1 levels of subtotals.
- Null values in the output of rollup operations typically mean that the rows contains sub total or grant total information
- Use nvl function for proper meaning.

Syntax:-

ROLLUP appears in the GROUP BY clause in a SELECT statement. Its form is:

Group by rollup(column1,column2);

or

SELECT ... GROUP BY

ROLLUP(grouping_column_reference_list)

When to Use ROLLUP:-

Use the ROLLUP extension in tasks involving subtotals.

- It is very helpful for subtotaling along a hierarchical dimension such as time or geography. For instance, a query could specify a ROLLUP of year/month/day or country/state/city.
- It simplifies and speeds the population and maintenance of summary tables. Data warehouse administrators may want to make extensive use of it. Note that population of summary tables is even faster if the ROLLUP query executes in parallel.

Examples:-

Sql> select deptno,sum(sal) from emp group by rollup(deptno);

Sql> select job,sum(sal) from emp group by rollup(job);

Passing multiple columns to rollup:-

When multiple column are passed to rollup, the rollup,groups the rows into blocks with same columns values.

```
Sql> select deptno, job,sum(sal) salary from emp
        group by rollup(deptno,job);
sql> select  job, deptno,sum(sal) salary from emp
          group by rollup(job,deptno);
sql> SELECT deptno,job, count(*), sum(sal)FROM
 emp GROUP BY ROLLUP(deptno,job);
```

CUBE:-

- it is an extension similar to rollup.
- CUBE enables a SELECT statement to calculate subtotals for all possible combinations of a group of dimensions.
- It also calculates a grand total.
- If "n" is the number of columns listed in the CUBE, there will be 2n subtotal combinations.
- This is the set of information typically needed for all cross-tabular reports, so CUBE can calculate a cross-tabular report with a single SELECT statement. Like ROLLUP, CUBE is a simple extension to the GROUP BY clause.
- The implementation cube is also called as N-dimensional cross tabulation.

When to Use CUBE

- Use CUBE in any situation requiring cross-tabular reports. The data needed for cross-tabular reports can be generated with a single SELECT using CUBE. Like ROLLUP, CUBE can be helpful in generating summary tables. Note that population of summary tables is even faster if the CUBE query executes in parallel.
- CUBE is especially valuable in queries that use columns from multiple dimensions rather than columns representing different levels of a single dimension. For instance, a commonly requested cross-tabulation might need subtotals for all the combinations of month/state/product. These are three independent dimensions, and analysis of all possible subtotal combinations will be commonplace. In contrast, a cross-tabulation showing all possible combinations of year/month/day would have several values of limited interest, since there is a natural hierarchy in

the time dimension. Subtotals such as profit by day of month summed across year would be unnecessary in most analyses.

Syntax

CUBE appears in the GROUP BY clause in a SELECT statement. Its form is:

```
SELECT ...  GROUP BY
CUBE (grouping_column_reference_list);
```

Examples:-

```
Sql> select deptno,job,sum(sal) salary from emp
group by cube(deptno,job);
sql> select job,deptno,sum(sal) salary from emp
group by cube(job,deptno);
```