CLAUSES: (26-10-2024)
========
- It is a statement to add to sql query for providing some additional facilities like filtering rows,sorting values,grouping similar data,fetching top most rows,finding subtotal and grand total based on column / columns automatically.
- SQLserver supports the following clauses are,
> WHERE
> ORDER BY
> GROUP BY
> HAVING
> TOP n
> ROLLUP
> CUBE

syntax:
======
<SQL query> + <Clause statement>;

WHERE:
======
- filtering rows before grouping the data in a table.(i.e fetching one - by - one row)
- it can use in "SELECT,UPDATE,DELETE" commands only.

syntax:
======
where <filtering condition>;

EX:
```
SELECT * FROM EMP WHERE EMPNO=7788;
UPDATE EMP SET SAL=35000 WHERE JOB='MANAGER';
DELETE FROM EMP WHERE SAL=5000;
```

ORDER BY:
=========
- to arrange a specific column values either in ascending order or in descending order.
- by default order by clause will arrange the values in ascending order only.
- if we want to arrange the values in descending order then we use a keyword is called as "DESC".
- order by clause is used in "SELECT" command only.

syntax:
=======
select * from <table name> order by <column name1> <asc/desc>,<column name2> <asc/desc>,....................;

EX:
```
SELECT * FROM EMP ORDER BY SAL;
SELECT * FROM EMP ORDER BY SAL DESC;

SELECT * FROM EMP ORDER BY ENAME;
SELECT * FROM EMP ORDER BY ENAME DESC;

SELECT * FROM EMP ORDER BY HIREDATE;
SELECT * FROM EMP ORDER BY HIREDATE DESC;
```

EX:
waq to display employees who are working under deptno is 20 and arrange those employees salaries in descending order?
```
SELECT * FROM EMP WHERE DEPTNO=20 ORDER BY SAL DESC;
```

EX:
waq to arrange deptno's in ascending order and those employees salaries in descending order from a table?
```
SELECT * FROM EMP ORDER BY DEPTNO,SAL DESC
```

NOTE:
======
- order by clause can apply on not only column names even though we can also apply on the position of column in "SELECT query".

EX:
```
SELECT EMPNO,ENAME,SAL FROM EMP ORDER BY 3 ;
SELECT EMPNO,ENAME,SAL FROM EMP ORDER BY 2 DESC;
SELECT EMPNO,ENAME,SAL FROM EMP ORDER BY 1 DESC;
```

28-10-2024:
===========
GROUP BY:
==========
- it is used to make groups based on column / columns.
- when we use "group by" clause we must use "aggregative functions" in the query to get the final result.(sum(),avg(),min().max(),count()).
- it is used in "SELECT" command only.

syntax:
======
```
select <column name1>,<column name2>,........,<aggregative function name1>,....................
from <table name> group by <column name1>,<column name2>,........;
```

Ex:
waq to find out no.of employees are working in the organization?

```
SELECT COUNT(*) FROM EMP;
```

Ex:
waq to find out no.of employees are working as a "MANAGER" in the organization?

```
SELECT COUNT(*) FROM EMP WHERE JOB='MANAGER';
```

Ex:
waq to find out no.of employees are working under each job wise?

```
SELECT JOB,COUNT(*) AS NO_OF_EMPLOYEES FROM EMP GROUP BY JOB;
```

Ex:
waq to find out no.of employees are working under each job along with deptno wise?

```
SELECT JOB,DEPTNO,COUNT(*) AS NO_OF_EMPLOYEES FROM EMP GROUP BY
DEPTNO,JOB;
```

Ex:
waq to display sum of salaries of each deptno wise?

```
SELECT DEPTNO,SUM(SAL) AS SUM_OF_SALARY FROM EMP GROUP BY DEPTNO;
```

Ex:
waq to display no.of employees,sum of salary,average salary,minimum salary,maximum salary of the employees from each deptno wise?

```
SELECT DEPTNO,COUNT(*) AS NO_OF_EMPLOYEES,
SUM(SAL) AS SUM_OF_SALARY,
AVG(SAL) AS AVG_SALARY,
MIN(SAL) AS MIN_SALARY,
MAX(SAL) AS MAX_SALARY FROM EMP GROUP BY DEPTNO;
```

HAVING:
=======

- filtering rows after grouping data in a table.
- whenever we use "having" clause we must use "group by" clause.

syntax:
======

```
select <column name1>,<column name2>,........,<aggregative function name1>,....................
from <table name> group by <column name1>,<column name2>,.......having<filtering
condition>;
```

Ex:

waq to display jobs in which job the no.of employees are working more than 3 employees?

```
SELECT JOB,COUNT(*) FROM EMP GROUP BY JOB HAVING COUNT(*)>3;
```

Ex:
waq to display deptno's from emp table if the sum of salary of the deptno is less than to 10000?

```
SELECT DEPTNO,SUM(SAL) FROM EMP GROUP BY DEPTNO HAVING SUM(SAL)<10000;
```

WHERE vs HAVING:
================

WHERE ======	HAVING ========
1. filtering rows before grouping data in a table.	1. filtering rows after grouping data in a table.
2. where clause condition is executing on individual rows in a table.	2. having clause condition is executing on group of from a table.
3. it does not supports "aggregative functions".	3. it supports "aggregative functions".
4. without "group by" we can use "where" clause condition.	4. without "group by" we cannot use "having" clause condition.

TOP n clause:
============

- to fetch top n rows from a table.here "n" means no.of rows.

syntax:
======

```
        top (n)
```

EX:

```
SELECT TOP(5) * FROM EMP;
UPDATE TOP(3) EMP SET SAL=5000;
DELETE TOP(2) FROM EMP;
```

Using all clauses in a single SELECT query:
=====================================

syntax:
======

```
select top(n) <column name1>,<column name2>,........,<aggregative function
name1>,....................
from <table name> [ where <filtering condition>
```

```
                    group by <column name1>,<column name2>,.......
                            having<filtering condition>
                                    order by <column name1> <asc/desc>,...........
            ];
```

EX:

```
SELECT TOP(1) DEPTNO,COUNT(*) FROM EMP  WHERE SAL>1000
                                        GROUP BY DEPTNO
                                                HAVING COUNT(*)>3
                                                        ORDER BY DEPTNO DESC;
```

ROLLUP & CUBE:
===============

- these are two special clauses which are used to find out sub total and grand total.
- these are implementing along with "group by" clause only.
  - ROLLUP : finding sub total & grand total based on a single column only.
  - CUBE : finding sub total & grand total based on multiple columns.

syntax for ROLLUP:
================

```
select  <column name1>,<column name2>,........,<aggregative function name1>,....................
from <table name> group by rollup(<column name1>,<column name2>,.......);
```

Example on ROLLUP with a single column:
====================================

```
SELECT DEPTNO,COUNT(*) FROM EMP GROUP BY ROLLUP(DEPTNO);
```

Example on ROLLUP with multiple columns:
=====================================

EX:

```
SELECT DEPTNO,JOB,COUNT(*) NO_OF_EMPLOYEES FROM EMP GROUP BY
ROLLUP(DEPTNO,JOB);
SELECT JOB,DEPTNO,COUNT(*) NO_OF_EMPLOYEES FROM EMP GROUP BY
ROLLUP(JOB,DEPTNO);
```

Example on CUBE with a single column:
====================================

```
SELECT DEPTNO,COUNT(*) FROM EMP GROUP BY CUBE(DEPTNO);
```

Example on CUBE with multiple columns:
=====================================

```
SELECT DEPTNO,JOB,COUNT(*) FROM EMP GROUP BY CUBE(DEPTNO,JOB);
SELECT JOB,DEPTNO,COUNT(*) FROM EMP GROUP BY CUBE(JOB,DEPTNO) ORDER BY
```

DEPTNO;
===
======