Chapter 7: Querying Using SQL

Ordering Records in Result----order by clause

- **The result set generated by the SQL SELECT statement is not ordered in any form by default.**
- **If we want to sort or order the result set, we can use the ORDER BT clause of SQL SELECT statement as per following format:**
  Select <comma separated select list> FROM <table>
  [WHERE <condition>]
  ORDER BY <fieldname> [ASC|DESC][,<fieldname>[ASC|DESC],….];
- **Keywords ASC and DESC denote the order ---ASC stands for ascending and the DESC stands for descending .**
- **If we do not specify any order keword ASC or DESC, then by default, the ORDER BY clause sorts**
- **The result set in ascending order.**
- **Eg**

```
select *                    select *
from data                   from  data
order by marks              order by marks ASC;
```

Ordering Data on Multiple Columns

- **To order the result set on multiple columns, we can specify the multiple column names in ORDER by clause along with the desired sort order , i.e as:**

- **Eg Select ***
  From data
  Order by section ASC, marks DESC

Ordering Data on the Basis of an Expression

- Some times we need to display the result of a calculation or a mathematical expression in the result set.
- In such case we may want or need to arrange our result set in the order of the calculated expression.
- The ORDER BY clause allows we to include the mathematical expression to order the result set by it.
- To arrange a result set on the basis of a mathematical expression, we should preferably (through not a necessity but preferably) include the mathematical expression in the select list so that it becomes easy to comprehend the result :
- SELECT rollno, name, grade , section , marks *.35 from data
  where marks>70
  order by section ASC, marks*0.35 DESC;

Aggregate Functions

- Aggregate functions work upon groups of rows, rather than on single rows , that is why, these functions are sometimes also called multiple row functions
- AVG
  - This function computes the average of given data.
  - AVG([DISTINCT| ALL]N)
  - Return average value of parameter(s) n.
- COUNT
  - This function counts the number of rows in a given column or expression.
  - Count({* [DISTINCT | ALL] EXPR})
  - Return THE NUMBER OF ROWS IN THE QUERY.
  - If we specify argument expr, this function returns rows where expr is not null.
  - We can count either all rows, or only distinct values of expr.
  - If we specify the *, this function returns all rows, including duplicates and nulls.
- MAX
  - This function returns the maximum value from a given column or expression.
  - MAX([DISTINCT|ALL] expr)
  - Returns maximum value of argument expr.

- MIN
  - This function returns the minimum value from a given column or expression.
  - MIN([DISTINCT|ALL] expr)
  - Returns minimum value of expr.
- SUM
  - This function returns the sum of values in a given column or expression.
  - SUM([DISTINCT|ALL] n)
  - Returns sum of values of n.

Grouping Result—Group BY

- The group by clause combines all those records that have identical values in a particular field or a group of fields , this grouping results into one summary record per group if group functions are used with it .
- GROUP BY clause is used in select statements to divide the table into groups.
- Grouping can be done by a column name , or with aggregate functions in which case the aggregate produces a value for each froup.
- Eg Select job , count(*)
  From empl
  Group by job;

NESTED GROUPS—GROUPING on multiple Columns

- **With GROUP BY clause , we can create groups within groups.**
- **Such type of grouping is called Nested grouping.**
- **Eg Select COUNT(EMPNO) from EMPL**
  GROUP BY dePTNO

PLACING CONDITIONS ON GROUPS---HAVING CLAUSE

- **The HAVING clause places conditions on groups in contrast to WHERE clause that places conditions on individual rows. While WHERE conditions cannot include aggregate functions, HAVING conditions can do so.**
- **Eg**

```
SELECT
AVG(GROSS),SUM(GROSS)

FROM EMPLOYEE
```

To display the jobs where the number of employees is less than 3 , we use the command

```
SELECT JOB, COUNT(*)

FROM empl

GROUP BY job

HAVING count(*)<3;
```

NON-GROUP EXPRESSION WITH group by

Eg

```
SELECT ENAME,SUM(SAL)

FROM EMPL

GROUP BY DEPTNO;
```