Experiment No: 6

Title : Study & Implementation of

- Group by & Having Clause
- Order by Clause
- Indexing

***Objective*:**

To learn the concept of group functions

***Theory*:**

- **GROUP BY:** This query is used to group to all the records in a relation together for each and every value of a specific key(s) and then display them for a selected set of fields the relation.

Syntax: SELECT <set of fields> FROM <relation_name>
GROUP BY <field_name>;

Example: SQL> SELECT EMPNO, SUM (SALARY) FROM EMP GROUP BY
EMPNO;

GROUP BY-HAVING : The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

Syntax: SELECT column_name, aggregate_function(column_name) FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value;

***Example* :** SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM (Orders
INNER JOIN Employees
ON Orders.EmployeeID=Employees.EmployeeID) GROUP BY LastName
HAVING COUNT (Orders.OrderID) > 10;

JOIN using GROUP BY: This query is used to display a set of fields from two relations by matching a common field in them and also group the corresponding records for each and every value of a specified key(s) while displaying.

Syntax: SELECT <set of fields (from both relations)> FROM relation_1,relation_2
WHERE relation_1.field_x=relation_2.field_y GROUP BY field_z;

Example:

SQL> SELECT empno,SUM(SALARY) FROM emp,dept
WHERE emp.deptno =20 GROUP BY empno;

- **ORDER BY:** This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

Syntax: SELECT <set of fields> FROM <relation_name>
ORDER BY <field_name>;

Example: SQL> SELECT empno, ename, job FROM emp ORDER BY job;

JOIN using ORDER BY: This query is used to display a set of fields from two relations by matching a common field in them in an ordered manner based on some fields.

Syntax: SELECT <set of fields (from both relations)> FROM relation_1, relation_2
WHERE relation_1.field_x = relation_2.field_y ORDER BY field_z;

Example: SQL> SELECT empno,ename,job,dname FROM emp,dept
WHERE emp.deptno = 20 ORDER BY job;

- **INDEXING**: An *index* is an ordered set of pointers to the data in a table. It is based on the data values in one or more columns of the table. SQL Base stores indexes separately from tables.

  An index provides two benefits:

  - It improves performance because it makes data access faster.
  - It ensures uniqueness. A table with a unique index cannot have two rows with the same values in the column or columns that form the index key.

Syntax:

CREATE INDEX <index_name> on <table_name> (attrib1,attrib 2….attrib n);

Example:

CREATE INDEX id1 on emp(empno,dept_no);

LAB PRACTICE ASSIGNMENT:

Create a relation and implement the following queries.

1. Display total salary spent for each job category.
2. Display lowest paid employee details under each manager.
3. Display number of employees working in each department and their department name.
4. Display the details of employees sorting the salary in increasing order.
5. Show the record of employee earning salary greater than 16000 in each department.
6. Write queries to implement and practice the above clause.