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 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);

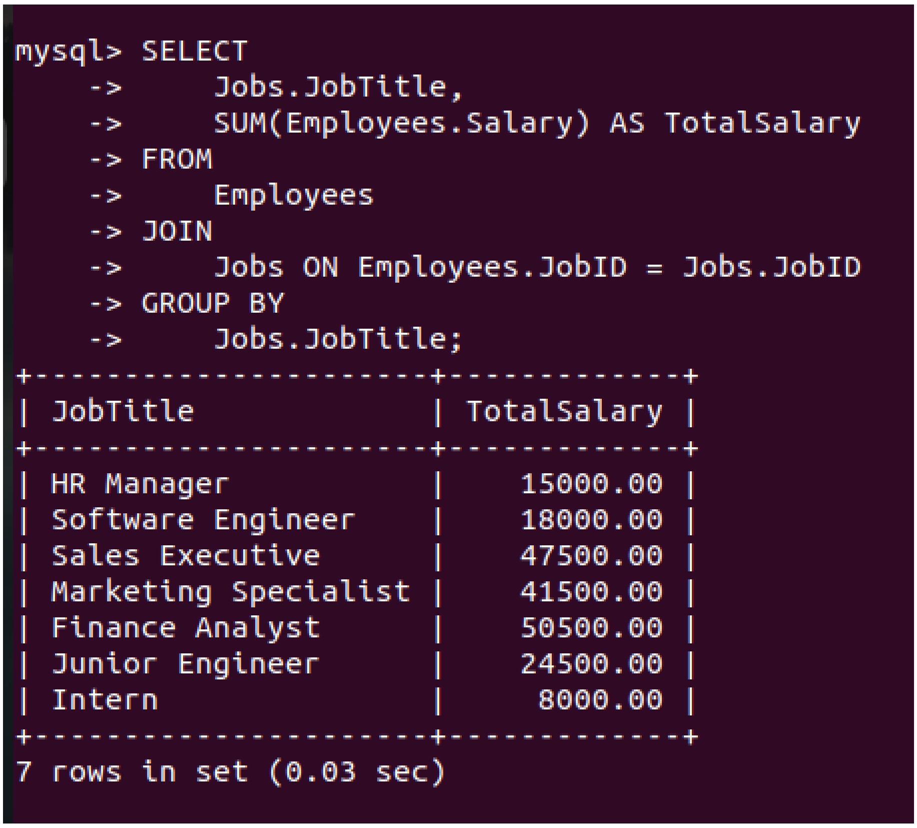
Example:

CREATE INDEX id1 on emp(empno,dept_no);

LAB PRACTICE ASSIGNMENT:

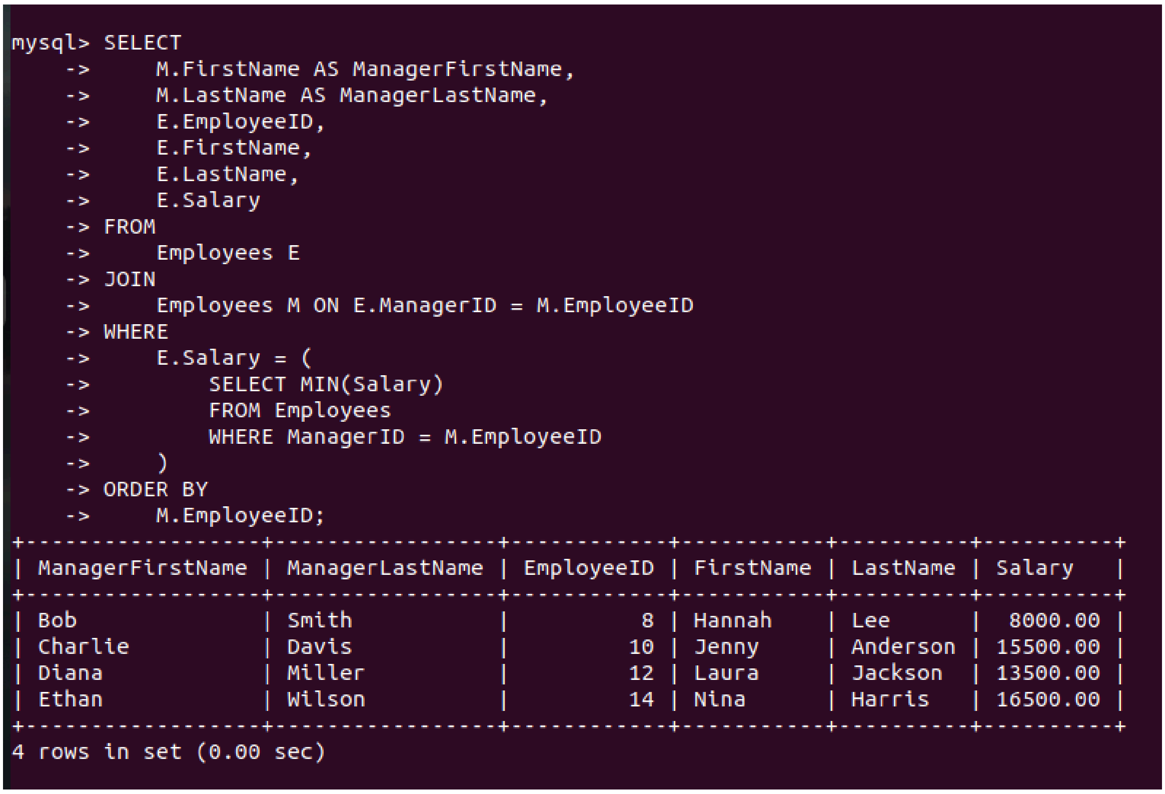
Create a relation and implement the following queries.

1. Display total salary spent for each job category.



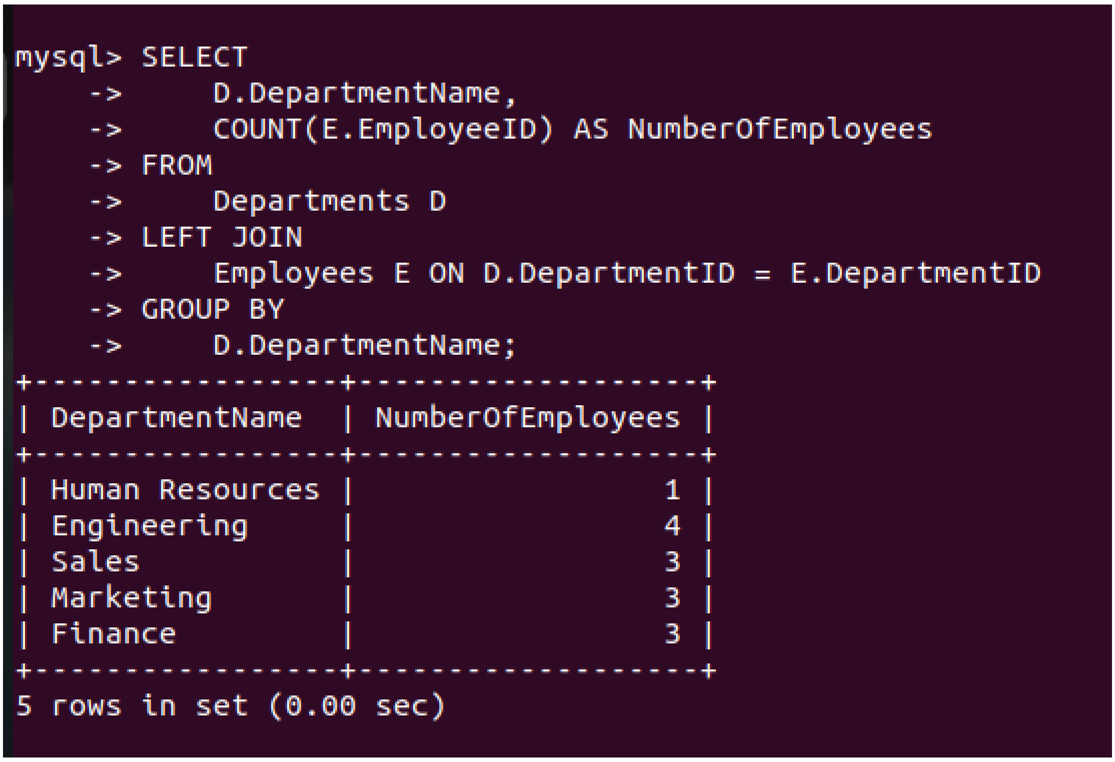
```
mysql> SELECT
    ->     Jobs.JobTitle,
    ->     SUM(Employees.Salary) AS TotalSalary
    -> FROM
    ->     Employees
    -> JOIN
    ->     Jobs ON Employees.JobID = Jobs.JobID
    -> GROUP BY
    ->     Jobs.JobTitle;
+----------------------+-------------+
| JobTitle             | TotalSalary |
+----------------------+-------------+
| HR Manager           |    15000.00 |
| Software Engineer    |    18000.00 |
| Sales Executive      |    47500.00 |
| Marketing Specialist |    41500.00 |
| Finance Analyst      |    50500.00 |
| Junior Engineer      |    24500.00 |
| Intern               |     8000.00 |
+----------------------+-------------+
7 rows in set (0.03 sec)
```

2. Display lowest paid employee details under each manager.



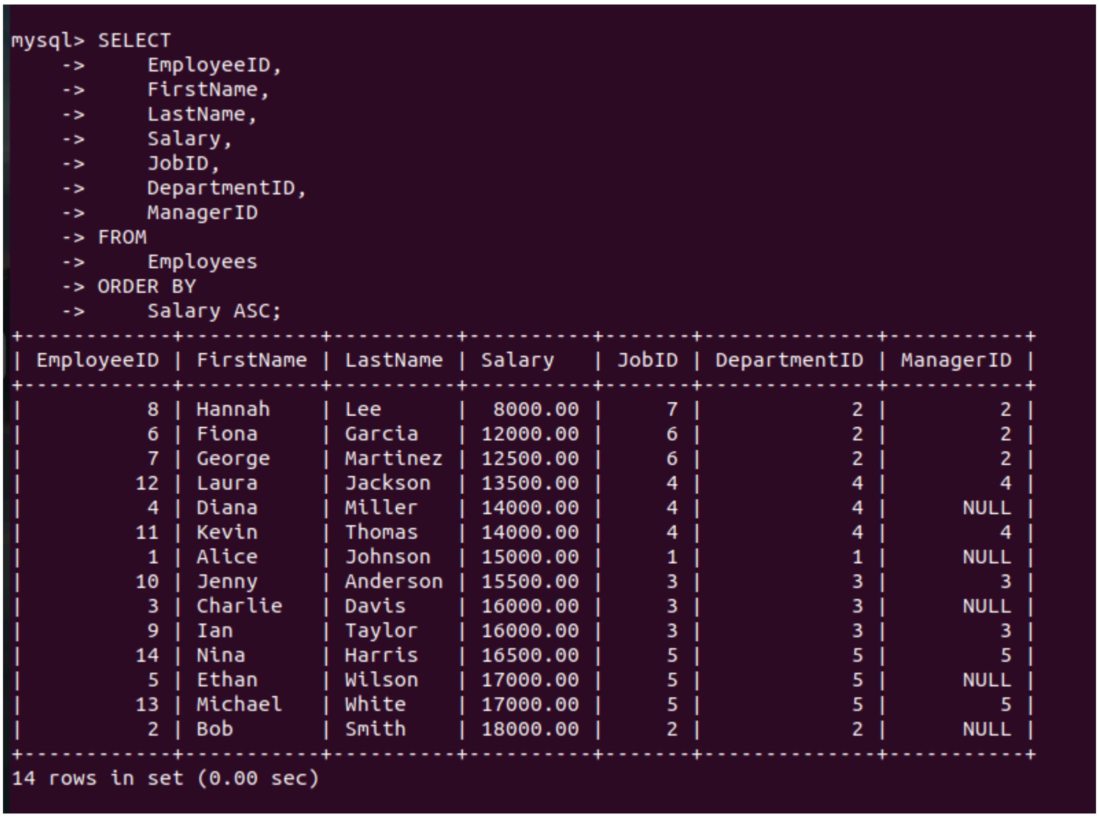
```
mysql> SELECT
    ->     M.FirstName AS ManagerFirstName,
    ->     M.LastName AS ManagerLastName,
    ->     E.EmployeeID,
    ->     E.FirstName,
    ->     E.LastName,
    ->     E.Salary
    -> FROM
    ->     Employees E
    -> JOIN
    ->     Employees M ON E.ManagerID = M.EmployeeID
    -> WHERE
    ->     E.Salary = (
    ->         SELECT MIN(Salary)
    ->         FROM Employees
    ->         WHERE ManagerID = M.EmployeeID
    ->     )
    -> ORDER BY
    ->     M.EmployeeID;
+------------------+-----------------+------------+-----------+----------+----------+
| ManagerFirstName | ManagerLastName | EmployeeID | FirstName | LastName | Salary   |
+------------------+-----------------+------------+-----------+----------+----------+
| Bob              | Smith           |          8 | Hannah    | Lee      |  8000.00 |
| Charlie          | Davis           |         10 | Jenny     | Anderson | 15500.00 |
| Diana            | Miller          |         12 | Laura     | Jackson  | 13500.00 |
| Ethan            | Wilson          |         14 | Nina      | Harris   | 16500.00 |
+------------------+-----------------+------------+-----------+----------+----------+
4 rows in set (0.00 sec)
```

3. Display number of employees working in each department and their department name.



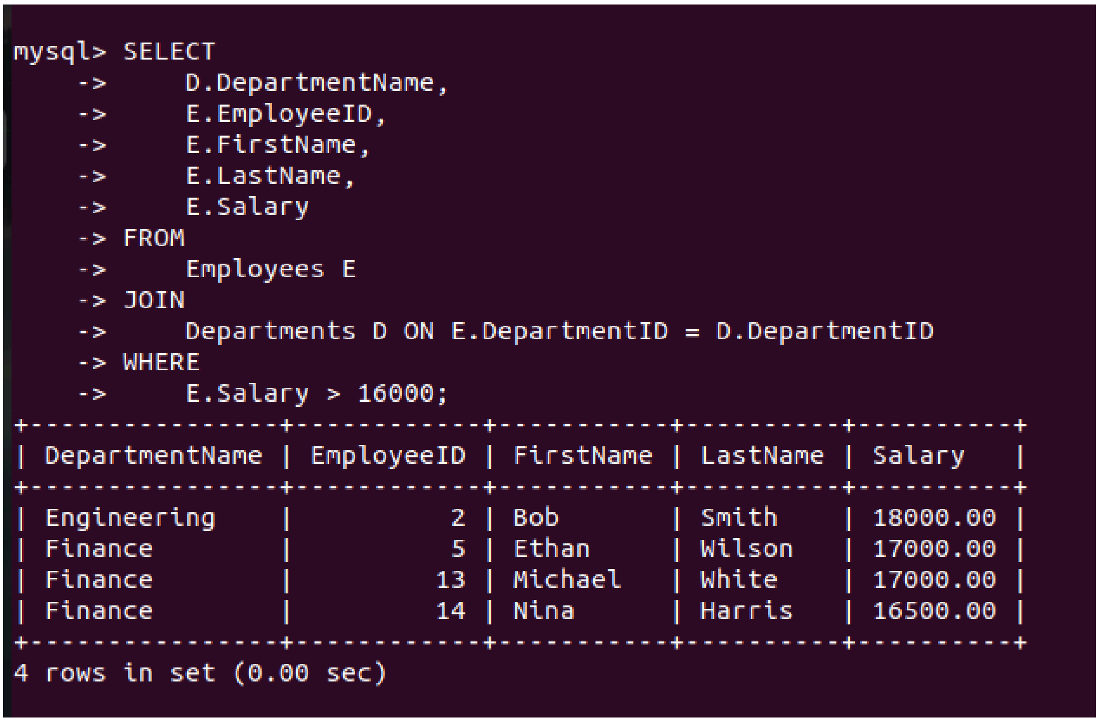
```
mysql> SELECT
    ->     D.DepartmentName,
    ->     COUNT(E.EmployeeID) AS NumberOfEmployees
    -> FROM
    ->     Departments D
    -> LEFT JOIN
    ->     Employees E ON D.DepartmentID = E.DepartmentID
    -> GROUP BY
    ->     D.DepartmentName;
+-----------------+-------------------+
| DepartmentName  | NumberOfEmployees |
+-----------------+-------------------+
| Human Resources |                 1 |
| Engineering     |                 4 |
| Sales           |                 3 |
| Marketing       |                 3 |
| Finance         |                 3 |
+-----------------+-------------------+
5 rows in set (0.00 sec)
```

4. Display the details of employees sorting the salary in increasing order.



```
mysql> SELECT
    ->     EmployeeID,
    ->     FirstName,
    ->     LastName,
    ->     Salary,
    ->     JobID,
    ->     DepartmentID,
    ->     ManagerID
    -> FROM
    ->     Employees
    -> ORDER BY
    ->     Salary ASC;
+------------+-----------+----------+----------+-------+--------------+-----------+
| EmployeeID | FirstName | LastName | Salary   | JobID | DepartmentID | ManagerID |
+------------+-----------+----------+----------+-------+--------------+-----------+
|          8 | Hannah    | Lee      |  8000.00 |     7 |            2 |         2 |
|          6 | Fiona     | Garcia   | 12000.00 |     6 |            2 |         2 |
|          7 | George    | Martinez | 12500.00 |     6 |            2 |         2 |
|         12 | Laura     | Jackson  | 13500.00 |     4 |            4 |         4 |
|          4 | Diana     | Miller   | 14000.00 |     4 |            4 |      NULL |
|         11 | Kevin     | Thomas   | 14000.00 |     4 |            4 |         4 |
|          1 | Alice     | Johnson  | 15000.00 |     1 |            1 |      NULL |
|         10 | Jenny     | Anderson | 15500.00 |     3 |            3 |         3 |
|          3 | Charlie   | Davis    | 16000.00 |     3 |            3 |      NULL |
|          9 | Ian       | Taylor   | 16000.00 |     3 |            3 |         3 |
|         14 | Nina      | Harris   | 16500.00 |     5 |            5 |         5 |
|          5 | Ethan     | Wilson   | 17000.00 |     5 |            5 |      NULL |
|         13 | Michael   | White    | 17000.00 |     5 |            5 |         5 |
|          2 | Bob       | Smith    | 18000.00 |     2 |            2 |      NULL |
+------------+-----------+----------+----------+-------+--------------+-----------+
14 rows in set (0.00 sec)
```

5. Show the record of employees earning salaries greater than 16000 in each department.



```
mysql> SELECT
    ->     D.DepartmentName,
    ->     E.EmployeeID,
    ->     E.FirstName,
    ->     E.LastName,
    ->     E.Salary
    -> FROM
    ->     Employees E
    -> JOIN
    ->     Departments D ON E.DepartmentID = D.DepartmentID
    -> WHERE
    ->     E.Salary > 16000;
+----------------+------------+-----------+----------+----------+
| DepartmentName | EmployeeID | FirstName | LastName | Salary   |
+----------------+------------+-----------+----------+----------+
| Engineering    |          2 | Bob       | Smith    | 18000.00 |
| Finance        |          5 | Ethan     | Wilson   | 17000.00 |
| Finance        |         13 | Michael   | White    | 17000.00 |
| Finance        |         14 | Nina      | Harris   | 16500.00 |
+----------------+------------+-----------+----------+----------+
4 rows in set (0.00 sec)
```