Table 1: employee Note: We'll refer this table for all the queries in this sheet					
employee_id [integer]	**first_name [characters]**	**last_name [characters]**	**salary [integer]**	**department [characters]**	**city [characters]**
1	John	Abraham	1000000	Banking	Delhi
2	Michael	Clarke	800000	Insurance	Bangalore
3	Roy	Thomas	700000	Banking	Gujarat
4	Tom	Jose	600000	Insurance	Delhi
5	Jerry	Pinto	650000	Insurance	Bangalore
6	Philip	Mathew	750000	Services	Chandigarh
7	Amir	Khan	650000	Services	Delhi

ORDER BY

ORDER BY statement is used to sort the column either in ascending or descending order. By default, it sort the data in ascending order.

Query Syntax:

```
SELECT column_1, column_2, ... column_n
FROM table_name
ORDER BY column_name order;
```

Note: Possible values for order: ASC (ascending) DESC (descending). If no order is specified then data is sorted according to ascending order.

Example 1: Get all employee details from the employee table sorted by first_name in ascending order.

Query:

```
1  SELECT *
2  FROM employee
3  ORDER BY first_name ASC;
```

Result:

```
+-------------+------------+-----------+---------+------------+-------------+
| employee_id | first_name | last_name | salary  | department | city        |
+-------------+------------+-----------+---------+------------+-------------+
|           7 | Amir       | Khan      |  650000 | Services   | Delhi       |
|           5 | Jerry      | Pinto     |  650000 | Insurance  | Bangalore   |
|           1 | John       | Abraham   | 1000000 | Banking    | Delhi       |
|           3 | Jones      | Thomas    |  700000 | Banking    | Gujarat     |
|           2 | Michael    | Clarke    |  800000 | Insurance  | Bangalore   |
```

```
|           6 | Philip      | Mathew    |  750000 | Services   | Chandigarh  |
|           4 | Tom         | Jose      |  600000 | Insurance  | Delhi       |
+-------------+-------------+-----------+---------+------------+-------------+
7 rows in set (0.05 sec)
```

Example 2: Get all employee details from the employee table sorted by first_name in descending order.

Query:

```
1  SELECT *
2  FROM employee
3  ORDER BY first_name DESC;
```

Result:

```
+-------------+-------------+-----------+---------+------------+-------------+
| employee_id | first_name  | last_name | salary  | department | city        |
+-------------+-------------+-----------+---------+------------+-------------+
|           4 | Tom         | Jose      |  600000 | Insurance  | Delhi       |
|           6 | Philip      | Mathew    |  750000 | Services   | Chandigarh  |
|           2 | Michael     | Clarke    |  800000 | Insurance  | Bangalore   |
|           3 | Jones       | Thomas    |  700000 | Banking    | Gujarat     |
|           1 | John        | Abraham   | 1000000 | Banking    | Delhi       |
|           5 | Jerry       | Pinto     |  650000 | Insurance  | Bangalore   |
|           7 | Amir        | Khan      |  650000 | Services   | Delhi       |
+-------------+-------------+-----------+---------+------------+-------------+
7 rows in set (0.00 sec)
```

Example 3: Get unique cities in ascending order from employee table.

Query:

```
1  SELECT DISTINCT city
2  FROM employee
3  ORDER BY city;
```

Note: Default order is ascending.

Result:

```
+-------------+
| city        |
+-------------+
| Bangalore   |
| Chandigarh  |
| Delhi       |
| Gujarat     |
+-------------+
4 rows in set (0.00 sec)
```

GROUP BY

The SQL GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.

Query Syntax:

```
SELECT column_1, column_2, ... column_n
FROM table_name
GROUP BY column_name;
```

Example 4: Get different department name from the employee table.

Query:

```
1  SELECT department
2  FROM employee
3  GROUP BY department;
```

Result:

```
+------------+
| department |
+------------+
| Banking    |
| Insurance  |
| Services   |
+------------+
3 rows in set (0.00 sec)
```

Example 5: Get department, total salary (as Dept_Group and Total_Salary) according to departments from employee table.

Query:

```
1  SELECT
2     department AS Dept_Group,
3     SUM(salary) AS Total_Salary
4  FROM employee
5  GROUP BY department;
```

Result:

```
+------------+--------------+
| Dept_Group | Total_Salary |
+------------+--------------+
| Banking    |      1700000 |
| Insurance  |      2050000 |
| Services   |      1400000 |
+------------+--------------+
3 rows in set (0.00 sec)
```

Example 6: Get department and total salary with respect to a department from employee table order by total salary in descending.

Query:

```
1  SELECT
2     department,
3     SUM(salary) AS Total_Salary
4  FROM employee
5  GROUP BY department
6  ORDER BY Total_Salary DESC;
```

Result:

```
+------------+--------------+
| department | Total_Salary |
+------------+--------------+
| Insurance  |      2050000 |
| Banking    |      1700000 |
| Services   |      1400000 |
+------------+--------------+
3 rows in set (0.00 sec)
```

Example 7: Get department, no of employees in a department and total salary with respect to a department from employee table sorted by total salary in descending order.

Query:

```
1  SELECT
2     department AS Dept_Group,
3     COUNT(first_name) AS Employees,
4     SUM(salary) AS TotalSalary
5  FROM employee
6  GROUP BY department
7  ORDER BY TotalSalary DESC;
```

Result:

```
+------------+-----------+-------------+
| Dept_Group | Employees | TotalSalary |
+------------+-----------+-------------+
| Insurance  |         3 |     2050000 |
| Banking    |         2 |     1700000 |
| Services   |         2 |     1400000 |
+------------+-----------+-------------+
3 rows in set (0.00 sec)
```

Example 8: Get department wise average salary from employee table order by salary ascending.

Query:

```
1  SELECT
2     department,
3     AVG(salary)
4  FROM employee
5  GROUP BY department
6  ORDER BY AVG(salary) ASC;
```

Result:

```
+------------+-------------+
| department | AVG(salary) |
+------------+-------------+
| Insurance  | 683333.3333 |
| Services   | 700000.0000 |
| Banking    | 850000.0000 |
+------------+-------------+
3 rows in set (0.01 sec)
```

LIMIT

The LIMIT clause is used to specify the number of records to return. It take one or two arguments, both are non-negative integer constant.

1) With two arguments, the first argument specifies the offset(starting row) of the first row to return, and the second specifies the maximum number of rows to return. The offset of the initial row is 0 (not 1). For example: in LIMIT(1,3) 1 indicates row number starts from 1 and 3 indicates total 3 rows have to be returned from 1.
2) With one argument, it returns top n rows of the table. For example: LIMIT 3 means it has to return top 3 rows of the table.

NOTE: row number of any table starts from 0.

Query Syntax: Top N rows

```
SELECT column_1, column_2, ... column_n
FROM table_name
LIMIT n;
```

Example 9: Select top 1 salary from Employee table.

Query:

```
1  SELECT salary
2  FROM employee
3  ORDER BY salary DESC
4  LIMIT 1;
```

Result:

```
+---------+
| salary  |
+---------+
| 1000000 |
+---------+
1 row in set (0.00 sec)
```

Example 10: Return all detail of employee from employee table, return row from 2 and return maximum 3 row order by salary in ascending order.

Query:

```
1  SELECT *
2  FROM employee
3  ORDER BY salary ASC
4  LIMIT 1,3;
```

Result:

```
+-------------+------------+-----------+--------+------------+-----------+
| employee_id | first_name | last_name | salary | department | city      |
+-------------+------------+-----------+--------+------------+-----------+
|           5 | Jerry      | Pinto     | 650000 | Insurance  | Bangalore |
|           7 | Amir       | Khan      | 650000 | Services   | Delhi     |
|           3 | Jones      | Thomas    | 700000 | Banking    | Gujarat   |
+-------------+------------+-----------+--------+------------+-----------+
3 rows in set (0.00 sec)
```

Example 11: Return city of employee from employee table, return row from 0 and return maximum 2 row order by salary in descending order.

Query:

```
1  SELECT city
2  FROM employee
3  ORDER BY salary DESC
4  LIMIT 0,2;
```

Result:

```
+-----------+
| city      |
+-----------+
| Delhi     |
| Bangalore |
+-----------+
2 rows in set (0.00 sec)
```

Example 12: Select 2nd highest salary from employee table.

Query:

```
1  SELECT salary AS 2ndMin
2  FROM employee
```

Result:

```
+--------+
| 2ndMin |
+--------+
```

```
3  ORDER BY salary DESC
4  LIMIT 1,1;
```

```
| 800000 |
+--------+
1 row in set (0.01 sec)
```

Problem 1	**Refer table structure given below.**

Table Structure

Table 2: student

student_id [integer]	first_name [characters]	last_name [characters]	age [integer]	city [characters]	fees [integer]
1	Vikas	Kumar	18	Delhi	2000
2	Neha	Jain	14	Gurugram	4500
3	Girish	Sharma	15	Delhi	3000
----more rows---					

Problem 1.1	**Get all details of students where first name is arranged alphabetically.**
Problem 1.2	**Get list of different cities present as group_city.**
Problem 1.3	**Get city wise total fees of students from students table in ascending order.**
Problem 1.4	**Get city wise maximum fees as Max_Fees from students table.**
Problem 1.5	**Get student details of two students having minimum age.**
Problem 1.6	**Get details of top three students having maximum fees.**
Problem 1.7	**Get complete names of students having 3rd to 7th minimum fees.**
Problem 1.8	**Get city wise average fees from students table arranged according to average fees in ascending order.**
Problem 1.9	**Get top three ages of students with all details.**
Problem 1.10	**Get recent 2 rows entered into our student table.** **Note: You can use student id (which is already present in ascending order) to get last rows entered.**