LAB 1

This material gives an overview of basic SELECT operations to retrieve data.

Generic SELECT query

```
SELECT
    [DISTINCT]
    [Column1,Column2,...| * ]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position} [ASC | DESC]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position} [ASC | DESC], ...]
```

Note: For executing example queries please create an EMPLOYEE table with following attributes and data-

```
+--------+------------+------+-----+--------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+--------+-------+
| id     | int(11)    | NO   | PRI | NULL    |       |
| name   | varchar(40) | NO   |     | NULL    |       |
| dept   | varchar(40) | NO   |     | NULL    |       |
| salary | int(11)    | YES  |     | NULL    |       |
+--------+------------+------+-----+--------+-------
```

```
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
| 500 | Ranjit | Technology |   6000 |
| 501 | Ritu   | Accounting |   NULL |
+-----+--------+------------+--------+
```

Basic SELECT Queries

1. **Select all rows and columns.** To get details of all the employees, one can use the following SELECT query. ‘*’ means all the columns shall be displayed.

```
mysql> SELECT * FROM employee;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
| 500 | Ranjit | Technology |   6000 |
| 501 | Ritu   | Accounting |   NULL |
+-----+--------+------------+--------+
```

2. **Select specific columns.** If you want to see only name and salary; you have to query: SELECT name, salary from employee;

```
mysql> SELECT name, salary FROM employee;
+--------+--------+
| name   | salary |
+--------+--------+
| Thomas |   5000 |
| Jason  |   5500 |
| Sanjay |   7000 |
| Nisha  |   9500 |
| Ranjit |   6000 |
| Ritu   |   NULL |
+--------+--------+
```

3. **Mathematical operations using SELECT.** We can perform a few basic mathematical operations inside MySql. Just look at the following example:

```
mysql> SELECT 2+3;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
```

Condition based SELECT

1. Basic WHERE Condition to Restrict Record

Instead of displaying all the records from a table, you can also use WHERE condition to view only records that match a specific condition as shown below.

```
mysql> SELECT * FROM employee WHERE salary > 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
+-----+--------+------------+--------+
```

Similar to “greater than >” you can also use “less than=”, “not equal to !=” as shown below.

```
mysql> SELECT * FROM employee WHERE salary < 6000;

mysql> SELECT * FROM employee WHERE salary >= 6000;

mysql> SELECT * FROM employee WHERE salary = 6000;

mysql> SELECT * FROM employee WHERE salary != 6000;
```

2. Conditions on String fields in WHERE Clause

The previous example displays how to restrict records based on numerical conditions. This example explains how to restrict records based on string values.

The exact match of strings works like numeric match using “equal to =” as shown below. This example will display all employees who belong to Technology department.

```
mysql> SELECT * FROM employee WHERE dept = 'Technology';
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Ranjit | Technology |   6000 |
+-----+--------+------------+--------+
```

Please note that this is case insensitive comparison. So, the following is exactly the same as above select command.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY';
```

You can also use != to display all the employee who does not belong to Technology department as shown below.

```
mysql> SELECT * FROM employee WHERE dept != 'TECHNOLOGY';
```

You can also perform partial string match using % in the keywords. The following will display all employees who’s last name begins with “John”.

```
mysql> SELECT * FROM employee WHERE name LIKE 'JOHN%';
```

The following will display all employees whos name ends with “Smith”.

```
mysql> SELECT * FROM employee WHERE name LIKE '%SMITH';
```

You can also give % at both beginning and end. In which case, it will search for the given keyword anywhere in the string. The following will display all employees who contain “John” in their name anywhere.

```
mysql> SELECT * FROM employee WHERE name LIKE '%JOHN%';
```

3. Combine multiple conditions in WHERE clause using OR, AND

You can also use OR, AND, NOT in WHERE condition to combine multiple conditions. The following example displays all employees who are in “Technology” department AND with salary >= 6000. This will display records only when both the conditions are met.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY' AND
salary >= 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 300 | Sanjay | Technology |   7000 |
| 500 | Ranjit | Technology |   6000 |
+-----+--------+------------+--------+
```

The following is same as above, but uses OR condition. So, this will display records as long as any one of the condition matches.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY' OR
salary >= 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
| 500 | Ranjit | Technology |   6000 |
+-----+--------+------------+--------+
```

Ordering and Grouping Clauses

These clauses are used to order or group the results.

1. GROUP BY in Select Command

Group By commands will group records based on certain conditions. The following example displays the total number of employees in every department.

```
mysql> SELECT DEPT, COUNT(*) FROM employee GROUP BY DEPT;
+------------+----------+
| dept       | count(*) |
+------------+----------+
| Accounting |        1 |
| Marketing  |        1 |
| Sales      |        1 |
| Technology |        3 |
+------------+----------+
```

Please note that when you use GROUP BY, you can use certain functions to get more meaningful output. In the above example, we’ve used count(*) group by commands. Similarly you can use sum(), avg(), etc, when you specify GROUP BY.

2. Use HAVING along with GROUP BY

When you use GROUP BY, you can also use HAVING to restrict the records further.

In the following example, it displays only the departments where the number of employee is more than 1.

```
mysql> SELECT COUNT(*) AS CNT, DEPT FROM employee GROUP BY
DEPT HAVING CNT > 1;
+-----+------------+
| CNT | dept       |
+-----+------------+
|   3 | Technology |
+-----+------------+
```

3. Sort Records using ORDER BY

The following records will ordered in alphabetical order based on dept column.

```
mysql> SELECT * FROM employee ORDER BY DEPT;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 501 | Ritu   | Accounting |   NULL |
| 400 | Nisha  | Marketing  |   9500 |
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Ranjit | Technology |   6000 |
+-----+--------+------------+--------+
```

Please note that by default it will sort by ascending order. If you want to sort by descending order, specify the keyword “DESC” after “ORDER BY” as shown below.

```
mysql> SELECT * FROM employee ORDER BY DEPT DESC;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Ranjit | Technology |   6000 |
| 100 | Thomas | Sales      |   5000 |
| 400 | Nisha  | Marketing  |   9500 |
| 501 | Ritu   | Accounting |   NULL |
+-----+--------+------------+--------+
```

You can also order by multiple columns as shown below.

```
mysql> SELECT * FROM employee ORDER BY DEPT, SALARY DESC;
```

4. Get Unique Values from a Column

To display all unique values from a column, use DISTINCT.

The following example will display all the unique dept values from the employee table.

```
mysql> SELECT DISTINCT DEPT FROM employee;
+------------+
| dept       |
+------------+
| Sales      |
| Technology |
| Marketing  |
| Accounting |
+------------+
```

<u>Aggregate Functions</u>

Aggregate Functions act on a group of data to get group related values

1. Count total number of records

Use count(*) in select command to display the total number of records in a table.

```
mysql> SELECT COUNT(*) FROM employee;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
```

2. Sum of all Values in a Column

To add all the values from a column, use SUM() function.

The following example will display the sum of salary column for all the employees who belong to Technology department.

```
mysql> SELECT SUM(SALARY) FROM employee WHERE DEPT =
'TECHNOLOGY';
+-------------+
| sum(salary) |
+-------------+
|       18500 |
+-------------+
```

3. Average of all Values in a Column

To average all the values from a column, use AVG() function.

The following example will display the average salary of each and every department. This combines GROUP BY with AVG() function.

```
mysql> SELECT DEPT,AVG(SALARY) FROM employee GROUP BY DEPT;
+------------+-------------+
| dept       | avg(salary) |
+------------+-------------+
| Accounting |        NULL |
| Marketing  |   9500.0000 |
| Sales      |   5000.0000 |
| Technology |   6166.6667 |
+------------+-------------+
```

4. Maximum value from a column

The following example will display the maximum salary of each and every department. This combines GROUP BY with MAX() function.

```
mysql> SELECT DEPT,MAX(SALARY) FROM employee GROUP BY DEPT;

+------------+-------------+
| DEPT       | max(SALARY) |
+------------+-------------+
| accounting |        NULL |
| marketing  |        9500 |
| sales      |        5000 |
| technology |        7000 |
+------------+-------------+
```

5. Minimum value from a column

The following example will display the minimum salary of each and every department. This combines GROUP BY with MIN() function.

```
mysql> SELECT DEPT,MIN(SALARY) FROM employee GROUP BY DEPT;

+------------+-------------+
| DEPT       | min(SALARY) |
+------------+-------------+
| accounting |        NULL |
| marketing  |        9500 |
| sales      |        5000 |
| technology |        5500 |
+------------+-------------+
```