LIKE Clause

LIKE clause is used in the condition in SQL query with the WHERE clause. LIKE clause compares data with an expression using wildcard operators to match pattern given in the condition.

Wildcard operators

- **Percent sign %**: represents zero, one or more than one character.
- **Underscore sign _**: represents only a single character.

Example:

SELECT * FROM Student WHERE s_name LIKE 'a%';

Above query will return all records where **s_name** starts with character 'a‘.

```
MySQL localhost:3306 arnikostudent SQL > select *from student where name LIKE '%a';
+--------+---------+-----------+------------+
| std_id | name    | address   | contact_no |
+--------+---------+-----------+------------+
|      1 | Reeya   | Bhaktapur | 9812345678 |
|      4 | Govinda | Lalitpur  | 9862232432 |
+--------+---------+-----------+------------+
2 rows in set (0.0011 sec)
```

ORDER BY Clause

Order by clause is used with **SELECT** statement for arranging retrieved data in sorted order. The **Order by** clause by default sorts the retrieved data in ascending order. We can also use asc to sort data in ascending order. To sort the data in descending order DESC keyword is used with Order by clause.

Syntax:

SELECT column-list|* FROM table-name ORDER BY column-name ASC | DESC;

Example:

SELECT * FROM student order by name desc;

```
4 rows in set (0.0005 sec)
MySQL localhost:3306 arnikostudent SQL > select * from student order by name ;
+--------+---------+-----------+------------+
| std_id | name    | address   | contact_no |
+--------+---------+-----------+------------+
|      4 | Govinda | Lalitpur  | 9862232432 |
|      2 | Junu    | Kathmandu | 9862537256 |
|      3 | Nitesh  | Kathmandu | 9862534352 |
|      1 | Reeya   | Bhaktapur | 9812345678 |
+--------+---------+-----------+------------+
4 rows in set (0.0009 sec)
MySQL localhost:3306 arnikostudent SQL > select * from student order by name desc;
+--------+---------+-----------+------------+
| std_id | name    | address   | contact_no |
+--------+---------+-----------+------------+
|      1 | Reeya   | Bhaktapur | 9812345678 |
|      3 | Nitesh  | Kathmandu | 9862534352 |
|      2 | Junu    | Kathmandu | 9862537256 |
|      4 | Govinda | Lalitpur  | 9862232432 |
+--------+---------+-----------+------------+
```

GROUP BY Command

Group by clause is used to group the results of a SELECT query based on one or more columns. It is also used with SQL functions to group the result from one or more tables.

Syntax:

SELECT column_name, function(column_name)

FROM table_name

WHERE condition

GROUP BY column_name;

Example:

SELECT name, address FROM student GROUP BY contact_no;

```
MySQL localhost:3306 arnikostudent SQL > select name, address from student group by contact_no;
+---------+-----------+
| name    | address   |
+---------+-----------+
| Reeya   | Bhaktapur |
| Govinda | Lalitpur  |
| Nitesh  | Kathmandu |
| Junu    | Kathmandu |
+---------+-----------+
4 rows in set (0.0011 sec)
```

HAVING Clause

Having clause is used with SQL Queries to give more precise condition for a statement. It is used to mention condition in Group by based SQL queries, just like WHERE clause is used with SELECT query.

Syntax:

SELECT * from student having contact_no=”9812345678”;

```
+---------+-----------+
4 rows in set (0.0011 sec)
MySQL localhost:3306 arnikostudent SQL > select * from student having contact_no = "9812345678";
+--------+-------+-----------+------------+
| std_id | name  | address   | contact_no |
+--------+-------+-----------+------------+
|      1 | Reeya | Bhaktapur | 9812345678 |
+--------+-------+-----------+------------+
1 row in set (0.0008 sec)
```

DISTINCT Keyword

The distinct keyword is used with SELECT statement to retrieve unique values from the table. Distinct removes all the duplicate records while retrieving records from any table in the database.

Syntax:

SELECT DISTINCT column-name FROM table-name;

Example:

SELECT DISTINCT contact_no FROM student;

```
1 row in set (0.0008 sec)
MySQL localhost:3306 arnikostudent SQL > select distinct contact_no from student;
+------------+
| contact_no |
+------------+
| 9812345678 |
| 9862537256 |
| 9862534352 |
| 9862232432 |
+------------+
4 rows in set (0.0009 sec)
```

AND Operator

AND operator is used to set multiple conditions with the WHERE clause, alongside, SELECT, UPDATE or DELETE SQL queries.

Example:

SELECT name from student WHERE contact_no =”9812345678” and address=”Bhaktapur”;

```
4 rows in set (0.0009 sec)
MySQL localhost:3306 arnikostudent SQL > select name from student where contact_no="9812345678" and address="Bhaktapur";
+-------+
| name  |
+-------+
| Reeya |
+-------+
1 row in set (0.0016 sec)
```

AVG() Function

Average returns average value after calculating it from values in a numeric column.

Syntax:

SELECT AVG(column_name) FROM table_name;

Example:

SELECT avg(std_id) from student;

```
1 row in set (0.0016 sec)
MySQL localhost:3306 arnikostudent SQL > select avg(std_id)from student student;
+-------------+
| avg(std_id) |
+-------------+
|      2.5000 |
+-------------+
1 row in set (0.0009 sec)
```

COUNT() Function

Count returns the number of rows present in the table either based on some condition or without condition.

Syntax:

SELECT COUNT(column_name) FROM table-name;

Example:

SELECT COUNT(name) FROM student;

```
1 row in set (0.0009 sec)
MySQL localhost:3306 arnikostudent SQL > select count(name)from student;
+-------------+
| count(name) |
+-------------+
|           4 |
+-------------+
1 row in set (0.0006 sec)
```

UCASE() Function

UCASE function is used to convert value of string column to Uppercase characters.

Syntax:

SELECT UCASE(column_name) from table-name;

Example:

SELECT UCASE(name) FROM student;

```
1 row in set (0.0007 sec)
MySQL localhost:3306 arnikostudent SQL > select ucase(name) from student;
+-------------+
| ucase(name) |
+-------------+
| REEYA       |
| JUNU        |
| NITESH      |
| GOVINDA     |
+-------------+
4 rows in set (0.0007 sec)
```

LCASE() Function

LCASE function is used to convert value of string columns to Lowecase characters.

Syntax:

SELECT LCASE(column_name) from table-name;

Example:

SELECT LCASE(name) FROM student;

```
4 rows in set (0.0007 sec)
MySQL localhost:3306 arnikostudent SQL > select lcase(name) from student;
+-------------+
| lcase(name) |
+-------------+
| reeya       |
| junu        |
| nitesh      |
| govinda     |
+-------------+
4 rows in set (0.0008 sec)
```

MID() Function

MID function is used to extract substrings from column values of string type in a table.

Syntax:

SELECT MID(column_name, start, length) from table-name;

Example:

SELECT MID(name,2,3) FROM student;

```
4 rows in set (0.0008 sec)
MySQL localhost:3306 arnikostudent SQL > select mid(name, 2,3) from student;
+----------------+
| mid(name, 2,3) |
+----------------+
| eey            |
| unu            |
| ite            |
| ovi            |
+----------------+
4 rows in set (0.0006 sec)
```

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Syntax:

SELECT column-name-list FROM

table-name1 INNER JOIN table-name2

WHERE table-name1.column-name = table-name2.column-name;

Example

SELECT * from student INNER JOIN teacher;

```
MySQL  localhost:3306  arnikostudent  SQL > select * from student inner join teacher;
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
| id | name    | address   | contact_no | id | name        | address   | contact_no | age | course_id | t_gender |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
|  1 | Reeya   | Bhaktapur | 9812345678 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  4 | Govinda | Lalitpur  | 9862232432 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  2 | teacher 2   | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  2 | Junu    | Kathmandu | 9862537256 |  2 | teacher 2   | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  2 | teacher 2   | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  4 | Govinda | Lalitpur  | 9862232432 |  2 | teacher 2   | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  3 | teacher 3   | Lalitpur  | 86763435   |  25 |         1 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  3 | teacher 3   | Lalitpur  | 86763435   |  25 |         1 | M        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  3 | teacher 3   | Lalitpur  | 86763435   |  25 |         1 | M        |
|  4 | Govinda | Lalitpur  | 9862232432 |  3 | teacher 3   | Lalitpur  | 86763435   |  25 |         1 | M        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  4 | teacher 4   | Kathmandu | 867234545  |  27 |         1 | F        |
|  2 | Junu    | Kathmandu | 9862537256 |  4 | teacher 4   | Kathmandu | 867234545  |  27 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  4 | teacher 4   | Kathmandu | 867234545  |  27 |         1 | F        |
|  4 | Govinda | Lalitpur  | 9862232432 |  4 | teacher 4   | Kathmandu | 867234545  |  27 |         1 | F        |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
16 rows in set (0.0006 sec)
```

Natural Join

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

Syntax:

SELECT * FROM

table-name1 NATURAL JOIN table-name2;

Example:

```
MySQL localhost:3306 arnikostudent SQL > select * from student join teacher on student.id = teacher.id;
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
| id | name    | address   | contact_no | id | name        | address   | contact_no | age | course_id | t_gender |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
|  1 | Reeya   | Bhaktapur | 9812345678 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  4 | Govinda | Lalitpur  | 9862232432 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
4 rows in set (0.0189 sec)
MySQL localhost:3306 arnikostudent SQL > select * from student join teacher on student.address = teacher.address;
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
| id | name    | address   | contact_no | id | name        | address   | contact_no | age | course_id | t_gender |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
|  4 | Govinda | Lalitpur  | 9862232432 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  4 | Govinda | Lalitpur  | 9862232432 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
5 rows in set (0.0008 sec)
```

```
MySQL localhost:3306 arnikostudent SQL > select * from student join teacher;
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
| id | name    | address   | contact_no | id | name        | address   | contact_no | age | course_id | t_gender |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
|  1 | Reeya   | Bhaktapur | 9812345678 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  4 | Govinda | Lalitpur  | 9862232432 |  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  2 | Junu    | Kathmandu | 9862537256 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  4 | Govinda | Lalitpur  | 9862232432 |  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  2 | Junu    | Kathmandu | 9862537256 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  4 | Govinda | Lalitpur  | 9862232432 |  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |
|  1 | Reeya   | Bhaktapur | 9812345678 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
|  2 | Junu    | Kathmandu | 9862537256 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
|  3 | Nitesh  | Kathmandu | 9862534352 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
|  4 | Govinda | Lalitpur  | 9862232432 |  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |
+----+---------+-----------+------------+----+-------------+-----------+------------+-----+-----------+----------+
```

Outer Join

Outer Join is based on both matched and unmatched data.

Outer Joins subdivide further into.

1. Left Outer Join
2. Right Outer Join

1.Left Outer Join

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns.

Syntax:

SELECT column-name-list FROM

table-name1 LEFT OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

Example

SELECT t.*, c.* FROM teacheras t LEFT OUTER JOIN course as c on t.id = c.course;

```
MySQL localhost:3306 arnikostudent SQL > select t.*, c.* from teacher as t left outer join course as c on t.id = c.course_id;
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
| id | name        | address   | contact_no | age | course_id | t_gender | course_id | course_name |
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
|  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |         1 | BCA         |
|  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |         2 | BBA         |
|  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |         3 | BBS         |
|  4 |  teacher 4  | Kathmandu | 867234545  |  27 |         1 | F        |      NULL | NULL        |
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
4 rows in set (0.0011 sec)
```

2. Right Outer Join

The right outer join returns a result set table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.

Syntax:

SELECT column-name-list FROM

table-name1 RIGHT OUTER JOIN table-name2

ON table-name1.column-name = table-name2.column-name;

Example

```
MySQL localhost:3306 arnikostudent SQL > select t.*, c.* from teacher as t right outer join course as c on t.id = c.course_id;
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
| id | name        | address   | contact_no | age | course_id | t_gender | course_id | course_name |
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
|  1 | Bishwo karn | lalitpur  | 4566878    |  30 |         2 | M        |         1 | BCA         |
|  2 |  teacher 2  | Bhaktapur | 8676646545 |  32 |         1 | F        |         2 | BBA         |
|  3 |  teacher 3  | Lalitpur  | 86763435   |  25 |         1 | M        |         3 | BBS         |
+----+-------------+-----------+------------+-----+-----------+----------+-----------+-------------+
3 rows in set (0.0007 sec)
```