create database scaler;

create database if not exists scaler;

use scaler;

create table Students(

id int primary key,

name varchar(30),

email varchar(30)

);

create table if not exists Students(

id int primary key,

name varchar(30),

email varchar(30)

);

create table if not exists Batches(

id int primary key,

name varchar(30)

);

drop table Students;

create database if not exists scaler;

use scaler;

Create

create table if not exists Persons(

ID int,

FirstName varchar(255),

LastName varchar(255),

City varchar(255),

Address varchar(255)

);

Insert

Insert can be done in two ways.

1. Insert into specific columns

INSERT INTO TABLE\_NAME(COL1, COL2, COL3, …, COLN) VALUES (VAL1, VAL2, VAL3, …, VALN);

1. Insert into all columns.

INSERT INTO TABLE\_NAME VALUES(VAL1, VAL2, VAL3, …, VALN);

INSERT INTO PERSONS VALUES(1, "ANUBHAV", "GUPTA", "MANSA", "PUNJAB");

INSERT INTO PERSONS VALUES(2, "IQBAL", "SINGH", "PATIALA", "PUNJAB");

INSERT INTO PERSONS VALUES(3, "TARUN", "SAINI", "PATIALA", "PUNJAB");

INSERT INTO PERSONS VALUES(4, "SAKSHI", "SINGLA", "PATIALA", "PUNJAB");

INSERT INTO PERSONS VALUES(5, "RUCHI", "VISHAVKARMA", "MUMBAI", "MAHARASHTRA");

INSERT INTO PERSONS(ID, FIRSTNAME, LASTNAME, CITY, ADDRESS) VALUES

(1, "ANUBHAV", "GUPTA", "MANSA", "PUNJAB"),

(2, "IQBAL", "SINGH", "PATIALA", "PUNJAB"),

(3, "TARUN", "SAINI", "PATIALA", "PUNJAB"),

(4, "SAKSHI", "SINGLA", "PATIALA", "PUNJAB");

(5, "RUCHI", "VISHAVKARMA", "MUMBAI", "MAHARASHTRA");

Read

SELECT \*DISTINCT {COLUMNS}

FROM TABLE\_NAME

WHERE {CONDITIONS}

GROUP BY {COLUMNS}

ODER BY {COLUMNS}

LIMIT X;

We can use the indexing to speed up the search.

SELECT CITY FROM PERSONS;

SELECT DISTINCT CITY FROM PERSONS;

Ascending Order

SELECT DISTINCT CITY FROM PERSONS ORDER BY CITY;

Descending Order

SELECT DISTINCT CITY FROM PERSONS ORDER BY CITY DESC;

SELECT 10;

10

SELECT 10\*10;

100

Relational Operators

=, !=, >, <, >=, <=

Logical Operators

AND, OR, NOT

NOT: IS, IS NOT VALUE

IN, NOT IN and BETWEEN

SELECT {COLUMNS} FROM TABLE\_NAME WHERE COLUMN2 IN (VALUE1, ...);

SELECT {COLUMNS} FROM TABLE\_NAME WHERE COLUMN2 NOT IN (VALUE1, ...);

SELECT {COLUMNS} FROM TABLE\_NAME WHERE COLUMN2 BETWEEN A AND B;

Note: A and B both are inclusive.

UPDATE

UPDATE TABLE\_NAME

SET COLUMN1 = NEW\_VALUE1, COLUMN2 = NEW\_VALUE2, ...

WHERE CONDITION;

DELETE

DELETE FROM TABLE\_NAME

WHERE CONDITION;

WILDCARDS  
A wildcard character is used to substitute one or more characters in a string. Wildcard characters are used with the [LIKE](https://www.w3schools.com/sql/sql_like.asp) operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

### Wildcard Characters

|  |  |
| --- | --- |
| Symbol | Description |
| % | Represents zero or more characters |
| \_ | Represents a single character |
| [] | Represents any single character within the brackets |
| ^ | Represents any character not in the brackets |
| - | Represents any single character within the specified range |

|  |  |
| --- | --- |
| Symbol | Example |
| % | bl% finds bl, black, blue, blob etc. |
| \_ | h\_t finds hot, hat, hit, hut etc. |
| [] | h[oa]t finds hot and hat but not hit and hut. |
| ^ | h[^oa]t finds hit and hut but not hot and hat. |
| - | c[a-d]t finds cat, cbt, cct and cdt |

All the wildcards can also be used in combinations!

Here are some examples showing different LIKE operators with '%' and '\_' wildcards:

|  |  |
| --- | --- |
| LIKE Operator | Description |
| WHERE CUSTOMERNAME LIKE 'a%' | Finds the customer name that starts with a |
| WHERE CUSTOMERNAME LIKE '%a' | Finds the customer name that ends with a |
| WHERE CUSTOMERNAME LIKE '%or%' | Finds the customer name that have or in any position |
| WHERE CUSTOMERNAME LIKE '\_r%' | Finds the customer name that contains second character as r |
| WHERE CUSTOMERNAME LIKE 'a\_%' | Finds the customer name that  starts with "a" and are at least 3 characters in length |
| WHERE CUSTOMERNAME LIKE 'a%o' | Finds the customer name that starts with a and ends with o |

Examples:

1. Using the % Wildcard

The following SQL statement selects all customers with a City starting with "ber":

SELECT \* FROM Customers WHERE City LIKE 'ber%'

The following SQL statement selects all customers with a City containing the pattern "es":

SELECT \* FROM Customers WHERE City LIKE '%es%';

1. Using the \_ Wildcard

The following SQL statement selects all customers with a City starting with any character, followed by "ondon":

SELECT \* FROM Customers WHERE City LIKE '\_ondon';

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

SELECT \* FROM Customers WHERE City LIKE 'L\_n\_on';

1. Using the [charlist] Wildcard

The following SQL statement selects all customers with a City

starting with "b", "s", or "p":

SELECT \* FROM Customers WHERE City LIKE '[bsp]%';

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

SELECT \* FROM Customers WHERE City LIKE '[a-c]%';

1. Using the [!charlist] Wildcard

The two following SQL statements select all customers with a City NOT starting with "b", "s", or "p":

SELECT \* FROM Customers WHERE City LIKE '[!bsp]%';

OR

SELECT \* FROM Customers WHERE City NOT LIKE '[bsp]%';