-- SQL(structured query language)- means format would be rows and columns

-- Categories of datatypes
-- String, Numeric, Date & Time

-- Create Tables

-- int includes numbers and varchar includes numbers, alphabets and special characters

-- TABLE SYNTAX --

--- create table name(
--column1 datatype,
--column2 datatype,
--column3 datatype,
--so on.....
--)/

```
create table person
(
id int,
name varchar(255),
email varchar(150),
password varchar(100),
contact varchar(15),
address text,
dob date
);
```

--Abh data insert krna h
-- Inserting Data into single row
--insert into table_name
--(column_name,column_name,so_on....)
--values
--("value1","value2","so_on....");

--Inserting Data into multiple rows
--sab same h bs values me comma dal ke krna h
--("value1","value2","so_on....),
--("value1","value2","so_on....),
--("value1","value2","so_on....).....

```
INSERT INTO person (id,name,email,password,contact,address,dob)
values (201,"shiv","ss@gmail.com","123","2299229922","mohali","2001-01-10"),
       (20,"shi","su@gmail.com","12","2299269922","mohali","2001-01-9"),
       (2,"sh","s@gmail.com","1","2299229982","chandigarh","2001-01-1");


--Select Query(used in crud operations):helps to read data from the table

SELECT * FROM person --(this will show all the data of the table)
SELECT id, name FROM person --(this will only show the data of the columns mentioned)



--Where clause used to filter records
--CONDITIONS in Where clause(Basic)
--  =    equals
-- !=    not equal
-- >=   greater than or euqals to
-- <=   less than or equals to
-- >     greater than
-- <     less than

SELECT * FROM person WHERE address = "chandigarh";      --(yeh wohi data dikhaye ga
jisme address chandigarh h )



--CONSTRAINTS in SQL (these are basically some check that we apply before taking the
input)
--BASIC:-
--  NOT NULL(states that value should not be null)
--  UNIQUE(states that value should be unique)
--  DEFAULT(sets default value)
--  CHECK(used to make a condition satisfy to input data)

--Here is an example

create table person
(
id int not null unique,
name varchar(255) not null,
email varchar(150) not null,
age int check (age>=18),
status boolean default 1
);
INSERT INTO person (id,name,email,age)
values (201,"shiv","ss@gmail.com",18)
```

```
--OPERATORS
--AND, OR and NOT

-- here is an example

create table person
(
id int,
name varchar(255),
email varchar(150),
password varchar(100),
contact varchar(15),
address text,
dob date
);
INSERT INTO person (id,name,email,password,contact,address,dob)
values (201,"shiv","ss@gmail.com","123","2299229922","mohali","2001-01-10"),
       (20,"shi","su@gmail.com","12","2299269922","mohali","2001-01-9"),
       (2,"sh","s@gmail.com","1","2299229982","chandigarh","2001-01-1");

SELECT * FROM person
where address="mohali" AND id=201
--(you can replace AND with OR, NOT  )



--IN Operator

--same purpose as the OR operator but instead of using too many OR operators you can
use just                one IN operator
--SELECT * FROM table_name
--WHERE column_name IN(value1,value2,...)

--here is an example

CREATE TABLE students (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    age INT,
    email VARCHAR(100),
    phone_number VARCHAR(15),
    major VARCHAR(50)
);
INSERT INTO students (student_id, first_name, last_name, age, email, phone_number,
major)
```

```
VALUES
(1, 'John', 'Doe', '18', 'john.doe@example.com', '123-456-7890', 'Computer Science'),
(2, 'Jane', 'Smith', '20', 'jane.smith@example.com', '987-654-3210', 'Biology'),
(3, 'Alice', 'Johnson', '18', 'alice.johnson@example.com', '555-555-5555', 'History');

SELECT * FROM students;
WHERE age IN(18,20);
```

```
--LIKE Operator
--when we have to find the particular information from the table where the word sarts with particular letter or somme combination we use LIKE("%") operator
--("%") rpresents zero, single or multiple chracters
--("_") represents just single chracter
```

LIKE Operator	Description
LIKE 'a%'	Starts with "a"
LIKE '%a'	End with "a"
LIKE '%or%'	Have "or" in any position
LIKE '_r%'	Have "r" in the second position
LIKE 'a_%'	Starts with "a" and are at least 2 characters in length
LIKE 'a__%'	Starts with "a" and are at least 3 characters in length
LIKE 'a%o'	Starts with "a" and ends with "o"

```
--here is an example

CREATE TABLE peoples(
  sno int,
  names varchar(255),
  email varchar(255)
  );
 INSERT into peoples(sno,names,email)
 VALUES(1,"Sumit","sumit123@gmail.com"),
 (2,"Amit","amit@gmail.com"),
 (3,"Sohail","sohu@123@gmail.com"),
 (4,"Raj","raj11@gmail.com");
 SELECT * FROM peoples WHERE names LIKE("s%")
```

--BETWEEN and NOT BETWEEN

--between operator is used to show data within a specifie range and not between operator is used to show data that is not included in the specified range

--here is an example

```
CREATE TABLE students(
srno int,
name varchar,
age int
);
INSERT into students(srno,name,age)
values(1,"Shiv",15),
(2,"ram",14),(3,"raj",16),(4,"Shim",17),(5,"Sh",17),(6,"uyu",18),(7,"Shiv",20),
(8,"kalesh",15);
SELECT * FROM students WHERE age BETWEEN 15 AND 18
```

--ORDER BY and DISTINCT

--order by is used to display the selected specified column in ascending or descending order
--here is an example

```
CREATE TABLE students(
srno int,
name varchar,
age int
);
INSERT into students(srno,name,age)
values(1,"shiv",15),(2,"ram",14),(3,"ajay",16),(4,"bhavi",17),(5,"charu",17),
(6,"uyu",18),(7,"puru",20),(8,"kalesh",15);
SELECT * FROM students ORDER BY name ASC
```

--distinct is used to show data of the specified column that are distinct(i.e shows all the different values from each other without repetition)

--here is an example

```
CREATE TABLE Students (
    StudentName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL
);
INSERT INTO Students (StudentName, City) VALUES
('John Doe', 'New York'),
('Jane Smith', 'Los Angeles'),
('Michael Johnson', 'Chicago'),
('Emily Davis', 'New York'),
('David Brown', 'Houston'),
('Emma Wilson', 'London'),
('Christopher Lee', 'Chicago'),
('Sophia Miller', 'Madrid'),
('Daniel Taylor', 'Miami'),
('Olivia Martinez', 'Chicago'),
('Matthew Anderson', 'Miami'),
('Ava Taylor', 'New York'),
('Andrew White', 'Los Angeles'),
('Isabella Harris', 'Houston'),
('Ethan Thomas', 'Chicago'),
('Mia Turner', 'New York'),
('William Jackson', 'Los Angeles'),
('Grace Davis', 'Houston'),
('James Martin', 'Chicago'),
('Chloe Rodriguez', 'Miami');

SELECT DISTINCT city FROM students
```

--SELECT DISTINCT city FROM students ORDER BY city ASC(use this if we want to show distinct and order wise data ASC|DESC)

```
--IS NULL and NOT NULL
--is null is used to show the records with data null and not null is used to show the records with data not null
--here is an example

CREATE TABLE Students (
    StudentID INT ,
    StudentName VARCHAR(50) NULL,
    Age INT NULL,
    City VARCHAR(50) NULL
);


INSERT INTO Students (StudentID, StudentName, Age, City) VALUES
(1, 'John Doe', 25, 'New York'),
(2, 'Jane Smith', 30, 'Los Angeles'),
(4, 'Emily Davis',NULL , 'ghgh');

SELECT * FROM Students WHERE age IS NULL




--LIMIT and OFFSET

-- if we want to limit the number of results that are returned you can simply use LIMITcommand with several rows to LIMIT by
--here is an example

CREATE TABLE Students (
    StudentName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL
);
INSERT INTO Students (StudentName, City) VALUES
('John Doe', 'New York'),
('Jane Smith', 'Los Angeles'),
('Michael Johnson', 'Chicago'),
('Emily Davis', 'dff'),
('David Brown', 'Houston'),
('Emma Wilson', 'London'),
('Christopher Lee', 'asa'),
('Sophia Miller', 'Madrid'),
('Daniel Taylor', 'Miami'),
('Olivia Martinez', 'Chicago'),
('Matthew Anderson', 'Miami'),
('Ava Taylor', 'New York'),
('Mia Turner', 'New York'),
SELECT * FROM Students ORDER BY City ASC LIMIT 5;
```

```
--offset basically sets an index from where we have to show the data
--here is an example

CREATE TABLE Students (
    StudentName VARCHAR(50) NOT NULL,
    City VARCHAR(50) NOT NULL
);
INSERT INTO Students (StudentName, City) VALUES
('John Doe', 'New York'),
('Jane Smith', 'Los Angeles'),
('Michael Johnson', 'Chicago'),
('Emily Davis', 'dff'),
('David Brown', 'Houston'),
('Emma Wilson', 'London'),
('Christopher Lee', 'asa'),
('Sophia Miller', 'Madrid'),
('Daniel Taylor', 'Miami'),
('Olivia Martinez', 'Chicago'),
('Matthew Anderson', 'Miami'),
('Ava Taylor', 'New York'),
('Andrew White', 'Los Angeles'),
('Isabella Harris', 'Houston'),
('Ethan Thomas', 'Chicago'),
('Mia Turner', 'New York'),
('William Jackson', 'Los Angeles'),
('Grace Davis', ‘Houston');
SELECT * FROM Students LIMIT 5 OFFSET 1;
```

—FUNCTIONS

COUNT( )	Returns the number of rows in a database table
SUM( )	Returns the total sum of a numeric column
AVG( )	Calculates the average of a set of values
MIN( )	Returns the minimum values in a set of non-NULL values
MAX( )	Return the maximum value in a set of non-NULL values

```
--here is an example

CREATE TABLE students (
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    salary DECIMAL(10, 2) NOT NULL
);

INSERT INTO students (name, age, salary) VALUES
    ('John Doe', 20, 15000.00),
    ('Jane Smith', 22, 18000.50),
    ('Bob Johnson', 21, 16000.75),
    ('Alice Brown', 23, 20000.25),
    ('Charlie Davis', 22, 17500.00),
    ('Eva Miller', 21, 16500.50),
    ('David Wilson', 24, 19000.00),
    ('Sophia Lee', 20, 15500.75),
    ('Michael Taylor', 23, 18500.25),
    ('Olivia Garcia', 22, 17000.50),
    ('Daniel Martinez', 21, 16000.00),
    ('Emma Anderson', 24, 19500.75),
    ('Liam Hernandez', 20, 15200.50),
    ('Isabella Smith', 23, 18200.25),
    ('William Davis', 22, 16800.00);

    SELECT COUNT(salary) FROM students;;
    SELECT SUM(salary) FROM students;
    SELECT AVG(salary) FROM students;
    SELECT MIN(salary) FROM students;
    SELECT MAX(salary) FROM students;




—UPDATE

--it is used to update the data in the table
--here is an example

CREATE TABLE students (

    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL
);

INSERT INTO students (name, email) VALUES
    ('John Doe', 'john.doe@example.com'),
```

```
    ('Jane Smith', 'jane.smith@example.com'),
    ('Bob Johnson', 'bob.johnson@example.com'),
    ('Alice Brown', 'alice.brown@example.com'),
    ('Charlie Davis', 'charlie.davis@example.com'),
    ('Eva Miller', 'eva.miller@example.com'),
    ('David Wilson', 'david.wilson@example.com'),
    ('Sophia Lee', 'sophia.lee@example.com'),
    ('Michael Taylor', 'michael.taylor@example.com'),
    ('Olivia Garcia', 'olivia.garcia@example.com');
UPDATE students
SET email = 'new.email@example.com'
WHERE name = 'John Doe';
```