SQL (Structured query language)

-- Data means some information

-- What is a Database in SQL?

A database is like a big container (storage) where we keep and organize data.

It stores data in tables (rows and columns), just like data in an Excel sheet.

-- It is used to communicate between your application and database

-- Application is used to various purpose

-- means e. g. paint it is used to draw, to create some graphic, some

Edit picture, resizing images and explain something

Media player it is used to play song

-- Also various purpose of application

-- Also various purpose of database

-- Communication with database

-- 1. **DDL** there are various command that help us the create a table

-- 1. Create command:

-- It is help the create a database, create a table, create procedure,

Create a trigger

2. Alter: which will allow u to modified the structure

-- changes the data

3. Drop: permanently delete the structure

-- These are 3 command help to create a design of structure

-- Creation is very important

-- First create a name of table (that we have decided)

-- E.g. make a user table -->

-- store all information of user like user first name, last name,

Gender, pan id

-- So that purpose u need to create that column

-- give the names for column.

-- In particular column which type information u have to save

-- That u should know datatypes

-- Which kind of information we will going to store in to that particular column

-- Rule and Regulation:

-- Constraints: are nothing but rules and regulations that we apply on specific column

-- help the to get clean data into our database

-- All part of database design that

==========

2. **Data Manipulation** (DDL)

-- Data manipulation language

-- E.g. flat -that empty but after some things inserted

-- Same like that database, empty database

-- insert data means data insertion

-- insert command

-- Update command - wrong name then modify the name

-- Delete -- no longer required

==========

3. DQL (Data Query Language) → Fetches data from database

-- SELECT → Retrieve data from tables

-- Select -- data retrieval

==========

4. Transactional Control / data :(TCL / DCL)

-- Commit and rollback

-- Commit means- save

-- Rollback - undue

==========

DDL (Data Definition Language) → defines the structure of the database.

Examples:

CREATE → Create database objects (table, view, etc.)

ALTER → Modify structure of a table

DROP → Delete database objects

TRUNCATE → Remove all data from a table quickly

DML (Data Manipulation Language) → Used to manage data inside tables.

Examples:

INSERT → Add new records

UPDATE → Modify existing records

DELETE → Remove records

==========

DQL (Data Query Language) → Fetches data from database.

Example:

SELECT → Retrieve data from tables

==========

DCL (Data Control Language) → Controls access/permissions.

Examples:

GRANT → Give access rights

REVOKE → Remove access rights

==========

TCL (Transaction Control Language) → Manages transactions in the database.

Examples:

COMMIT → save changes permanently

ROLLBACK → Undo changes

SAVEPOINT → Set a point to rollback

==========

In short:

DDL → Structure

DML → Data

DQL → Query

DCL → Permissions

TCL → Transactions

==========

Categories of SQL Datatypes

Category	Description	Examples
1. Numeric	Store numbers (used for calculations)	`INT`, `FLOAT`, `DECIMAL`, `NUMERIC`, `BIGINT`, `SMALLINT`
2. Character / String	Store text or characters	`CHAR`, `VARCHAR`, `TEXT`

Category	Description	Examples
3. Date and Time	Store date, time, and timestamps	`DATE`, `TIME`, `DATETIME`, `TIMESTAMP`, `YEAR`
4. Binary	Store binary data (like images, files)	`BINARY`, `BLOB`
5. Boolean	Store true/false values	`BOOLEAN`
6. JSON / XML	Store structured data	`JSON`, `XML`

□ Examples

Example	Datatype	Description
`EmployeeID INT`	`INT`	Whole numbers only
`Name VARCHAR(50)`	`VARCHAR(50)`	Text up to 50 characters
`Salary DECIMAL(10,2)`	`DECIMAL(10,2)`	Up to 10 digits total, 2 after decimal
`HireDate DATE`	`DATE`	Stores only date (`YYYY-MM-DD`)
`LoginTime DATETIME`	`DATETIME`	Stores date and time
`IsActive BOOLEAN`	`BOOLEAN`	True or False
`Details JSON`	`JSON`	Stores structured JSON data

==========

command

1. show databases;
   -- show the tables

1. if u have create a database : create database fortune;
2. if u have show again database : show databases;
3. if u have work one database: use fortune;
4. if u have see the tables means column : show tables;
5. if create a column / tables :

-- Syntax : create table table_name(column_name1 datatype constraint,
column_name2 datatype,
column_name2 datatype);

-- write : create table Person(pid int not null, pname text,
city varchar(20));

-- if u have show the table in database : show tables;

-- if u have show the columns in table : desc Person;

Syntax: describe table_name;

write : desc Person;

==========

For example:

-- create table Emplyoee(emp_id int primary key auto_increment, emp_name text, salary decimal(10, 2), city varchar(20));

-- show tables; 2 tables

-- desc Emplyoee;

==========

1.Task:
-- Table name : Department

-- column : dept_id int, dept_Name text, emp_id int

==========

1. task:
   -- Create vehical

-- column : vehicle_id int not null, brand text, model text,

date_of_manifacturing date

create table Vahicale(vehicle_id int noy null, brand text, model test,

date_of_manufacturing data);

-- show tables;

==========

If u have to insert the data in table

-- Syntax :
-- insert into Person(column_name1, column_name2, column_name) values
(value1, value2, value3);

-- u have to add the data respective according datatypes

-- insert into Person(pid, pname, city)values(101, ram, pune);

-- it will say 1 row affected

==========

If u have show the data that u have already added

-- syntax :
-- select * from Table_name;

-- select * from Person;

==========

If u have store multiple data at a time

-- insert into Person (pid, pname, city) values
(101, 'Anup','Pune'), (102, 'Ram','Pune'), (101, 'Arun','Pune');

-- u will get the msg 3 rows affected

-- select * from Person;

==========

-- if u have no need to person table

-- Syntax : drop table_name;

-- Drop table Person;

-- show tables;

==========

-- If u have delete the multiple tables

-- Syntax : drop table employee, vehicle ;

-- comma separated

-- shows tables;

==========

-- daily base uses command

-- show databases : which will allow us to list down the databases into sql server

-- create database database_name : if u have create database

-- use database_name : choose / select the database

-- show tables : display table / it is list the table

-- create, drop. insert, desc : DDL command

==========

Numerical datatypes:

-- Integers(Whole numbers 0-9 digits)

-- int

-- 2 types :

-- Singed and unsigned (any buddy's know?)

-- e.g. singed : its allow to store both positive and negative data

-- when we create a normal variable that by default signed

-- unsigned : only positive

-- Example 1: Age of a Person

A person’s age cannot be negative.

So, we should use UNSIGNED.

-- for example : 0 to infinity

-- Example 2: Bank Account Balance

-- Balance can go negative (overdraft) or positive.

-- So, we use SIGNED.

-- for example : 13, -13 u can able to store

-- tinyint, smallint, mediumint, int, bigint (size and range different)

-- tinyint- allow me to store 3 digit number till 127

signed : (-128 to 127)

unsigned : (0 - 255)

-- this is the range of tiny int

-- smallint - allow me to store 5 digit number

-- mediumint - allow me to store 8 digit number

-- int - allow me to store 10 digit number

-- bigint : allow me to store 90 digit number

==========

Decimal --> allow u to store both positive and negative fractional

point values

-- e.g. : salary decimal(10, 2);

-- 10 indicates -- total length of a decimal number

-- 2 indicates -- fractional data after decimal points

-- e.g. : 10000000.00

-- salary decimal(5, 3);

-- e.g. : 10.000

-- salary decimal(3, 3);

-- e.g. : 0.123

-- length is not fixed , u can mention 1.456

-- but decimal is fixed , after decimal point fix

-- fractional value means those values which are not a whole number

==========

-- Boolean datatypes:

-- true -> represent as 1 in memory

-- false -> represent as 0 in memory

After creating that Boolean but internally save in numeric value

That's why Boolean datatype consider as a tinyint

==========

text

Types.4 :

-- tiny text, text, medium text and longtext

-- this is a fixed sized

-- when ever enter a data in text field u have to insert the data in

single coat

-- varchar : consider as a trilling space

-- char : not consider as a trilling space

in case of char and varchar less memory wastage

-- in case of text more memory wastage

-- Tiny text : fixed size --> 255 bytes

-- text : 64 Kb (kilobyte) --> above 64 pages

-- medium text : 16 mb

-- longtext : 4 Gb

==========

Enum datatype :(anam)

-- enumerated list -- fixed type of data

-- it will accept only data from the specified

-- e. g. status enum('high', 'medium', 'low');

-- not specify other data

-- exam : gender

-- we can able to use this kind of data where we have fixed data

-- country, month, day

-- one kind of restriction outsider data is not allow

-- amazon status : ship, transit, nearest hub, out of delivery

and delivery, return

==========

Datatype : Date and time -- important

-- transaction order, birthdate, employee join date, order date

-- How to store date into database

-- dob date constraints (optional)

-- dob- column name, date -> datatype

-- describe specific format : DD-MM-YY / YY-MM--DD

-- 2025-10-12

-- while inserting the data u don't need to provided dash

-- without dash it will work , date will gate inserted

-- whenever u provided data dash separated and without dash separated

it should be always visible year, month, date

-- 00-69 --> 2069

-- 70-99 --> 1970-1999

-- that data accessible by this format

-- we show the date which is Accept by database

-- but if u have show the data as per user then u need to use the

function to display that data.

-- that we will see

-- logintime time not null (to store a time)

-- specific format --> HH:MM:SS

-- Combining the both are store data and time that we can use

datetime database

-- u need to write the data in single coat

-- store the individual data

-- store the individual time

-- datetime -> both store

-- timestamp -> it is also same like your datetime datatype

-- datetime is save current date and time

-- means if we have store one data related time and date with

the help of date time datatype then its visible Indian date time.

if that save in India

-- but u want to show that in USA then its visible as Indian time.

-- and if u have use the timestamp datatypes that data visible the

in respective place.

-- u can access date different places

==========

functions :

-- select now();

-- it will provide current system date and time

-- if u have only time

-- select date(now()); (date of now)

-- it will return only date

-- 2nd option :

-- CURDATE(); --> it will display current date directly

-- current time --> CURTIME();

-- How to display a date in specific format

-- DATE_FORMAT();

-- which will allow u to format a date in a specific way

-- It is accept two argument

1. which date u want modify

2. which format u want specify

-- how u want a format to visible date

-- show Time and date

-- %b --> it shows 3 latter. e.g. Aug

-- %Y --> complete year

-- date : if u want show the weekdays

-- %a --> only shows 3 latter. e. g. mon, sun

-- %W --> show total name of day. e.g. Monday

==========

I want find out the 2 different date

-- DATEDIFF();

-- it accept two arguments : from date - to date

-- current date and history date , future date

==========

-- DATE_ADD();

-- u need to use INTERVAL keyword

-- Which will allows to display a date after specific interval

-- that interval could be day, week, month, year

back date interval

-- DATE_SUB(CURDATE(), INTERVAL 1 MONTH)

Why we use :

-- separate date, month, year, hour, minute

-- universal standard time and current time (Indian standard)

-- select curtime(), UTC_time();

==========

datetime datatype

-- it contain both date and time

-- select now() ==> date and time

-- datetime and timestamp both are visible same

-- why we set the timezone , we use datatype that time we change the the timezone then it same data.

-- when we use timestamp then this change according timezone

-- datetime - It is not affected by time zones.

-- timestamp - it is affected by time zone.

-- timestamp - MySQL converts the time automatically between your system's local time and UTC (Universal Time).

-- that we need use to settimezone command

-- How to set timezone command let see

-- Syntax : set time_zone='+00:00';

-- 00 means universal time zone

-- Use DATETIME → When you just want to store a date & time as it is.

-- Use TIMESTAMP → When you want MySQL to track creation or update time automatically.

-- update itself

-- TIMESTAMP automatically saves current date and time.

-- TIMESTAMP = "Store the exact time when the record was added automatically."

-- "when" something happened.

1️⃣ Show current Universal Time (UTC)

```
SELECT UTC_TIMESTAMP();
```

Example output:

+---------------------+

| UTC_TIMESTAMP() |

+---------------------+

| 2025-10-14 06:00:30 |

+---------------------+

🕒 → This shows current time in UTC, regardless of your local time zone.

✅ 2 Compare UTC and Local Time

SELECT NOW() AS Local_Time, UTC_TIMESTAMP() AS UTC_Time;

📊 Example output:

Local_Time UTC_Time

2025-10-14 11:30:30 2025-10-14 06:00:30

✅ Example: Using DATETIME to store event time

CREATE TABLE events (

event_id INT AUTO_INCREMENT PRIMARY KEY,

event_name VARCHAR(50),

event_time DATETIME

);

✅ Insert example data

INSERT INTO events (event_name, event_time)

VALUES

('Meeting', '2025-10-14 10:30:00'),

('Birthday Party', '2025-12-05 19:00:00');

✅ View the data

```
SELECT * FROM events;
```

Using TIMESTAMP

```
CREATE TABLE login_details (
  user_name VARCHAR(50),
  login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

✅ Insert Data

```
INSERT INTO login_details (user_name)
VALUES ('Rutuja'), ('Amit');
```

✅ View Data

```
SELECT * FROM login_details;
```

==========

Binary datatype :

-- how to upload image

-- nokari profile : u can record short video of your bio and put that
  in noukari profile

-- then u can upload image as well

-- this all action go to in database

-- basically video , audio, imgs that all this data saved in binary

-- tinyblob, blob, mediumblob, longblob

-- blob stand for -- binary large object

-- u need to use syntax to upload binary data

-- select @@secure_file_priv;

-- priv -> privileges

we can load the image that we can use the path

-- how it will use? u can use a one function. i.e. LOAD_FILE()

==========

Json datatype

-- Type of file

-- json is collection of array

-- multiple array store in one format

-- MySQL support JSON datatype and where u can able to store

multiple value for single attribute in to your datatype

-- we can able to store n number of value at the same place

-- for single attribute u can give the multiple values that u

have to provide json

-- We use the JSON datatype in SQL when we want to store flexible, structured data inside a single column

==========

u want see the headers of array in json column

-- Array start from 0 indexing value

-- use command : select **JSON_KEYS(json column name)** from table_name;

-- select JSON_KEYS(details)from product;

==========

-- if u want see all color

-- Command -->

select JSON_EXTRACT(JSON column,"$.jsonColumn_Name") from tableName

-- select JSON_KEYS(details,"$.colors")from product;

---

u want to see particular value

-- u want to see particular size or color of that product

-- only select one colour - blue

-- Command -->

select JSON_EXTRACT(JSON column,"$.jsonColumn_Name[]") from tableName

-- [size]==> provide size of array (indexing value)

-- select JSON_KEYS(details,"$.colors[1]")from product;

---

if u see the json file format data then use the command

-- JSON_PRETTY(column name)from tables_name;

-- JSON_PRETTY(details) from products;

---

if u have converted the normal record into json format

-- select JSON_ARRAY(column1, column2, column3) from table_name;

-- select JSON_ARRAY(id, name, Dob) from person;

Constraints

-- A constraint in SQL is a rule you define on a table or column to restrict what data can be stored there.

--constraints keep the data clean and correct.

1. not null
2. default
3. check
4. primary key (Auto increment : with numerical primary key)
5. unique key
6. Foreign key

==========

- 1.Not null:

-- the columns for which we can not insert empty data

-- * --> mandatory

==========

2. default

-- with apply text, numeric,

The **DEFAULT constraint** is used to **give a column a default value** — if no value is provided when inserting data.

It helps to **avoid NULL values** and ensures that every record has a valid entry.

Syntax

```
CREATE TABLE table_name (
  column_name datatype DEFAULT default_value
);
```

==========

CREATE TABLE Employees (

```
  EmpID INT,
  Name VARCHAR(50),
  City VARCHAR(30) DEFAULT 'Pune'
);
```

---

3. Check

-- u want to data in specific range then u can achieve check constraint

-- to verify specific condition

-- discrepancy in your age

-- voting table should accept the entries all the users which having age greeter than 18

-- less than 18 they don't give the entry

-- this all constraints work at the time of data insertion

-- check the condition is match or not

-- Example 1: Bank Account (Minimum Balance)

Rule: Every account must always have at least 1000 balance.

Balance DECIMAL(10,2) CHECK (Balance >= 1000)

Balance = 500 → rejected.

-- Example 2: Employee Age (Company Policy)

Rule: Employee age must be between 18 and 60 years.

Age INT CHECK (Age >= 18 AND Age <= 60)

Age = 15 → rejected.

-- if u want see where u have apply the check constraint

-- command --> show create table table_name;

-- u will able to see applied constraint in your particular table

-- complete details of your table

-- it will exactly show u definition of that u have created in

particular table

-- check will simple check the data at the time of inserting in

to your table

Primary key

-- primary key is unique value which allow us to find out

each record uniquely

-- if the point out the one value then we will get the one record

not more than one

-- associate one value with one record

-- which is unique value of each record

-- the value which will be going to be unique for each record we can make them as a primary key

-- A Primary Key is a column (or set of columns) in a table

that uniquely identifies each row.

-- It means no two rows can have the same value in that column, and it cannot be empty (NULL).

Real Life Example of Primary Key

Think about your Aadhaar Number (or Social Security Number in some countries):

Every person has a unique Aadhaar number.

Two people cannot have the same Aadhaar number.

It cannot be empty.

So, Aadhaar Number acts like a Primary Key for all citizens.

Another Example

In a School:

Each student has a Roll Number.

Roll Number is unique (no two students in the same class share it).

Every student must have a Roll Number.

That Roll Number is the Primary Key in the Students table.

In short:

A Primary Key = Roll Number / Aadhaar Number / Passport Number → something that is unique and cannot be empty.

Rules:

-- In a single table there is only one primary key

-- we can make only one column as primary

-- multiple primary key is not allow

-- but u can make multiple column as primary

-- primary key is by default not null

(i can not able to keep empty)

-- primary key dos not accept duplicate value

-- so this is background information of primary key

-- que : how many primary key i can able to create in side a

single table?

-- your ans will be only one

-- one table can have only one primary key

-- we can make single column primary key

-- we can make multi column primary key

Unique

-- unique is same like your primary key

-- it will have value of each record

-- unique could be null and not null

-- for example : no mobile number then skip it

Real-Life Example

Email ID in a Company

Every employee must have a different email ID.

No two employees can use the same email.

But, if someone doesn't have an email yet, that place can be left NULL once.

Here, Email ID column can be declared as UNIQUE.

Another Example

Mobile Number in a School Database

Parents may register their mobile number.

Each mobile number must be unique so that messages go to the right parent.

If some parent hasn't given a number yet, it can be left NULL.

-- u can make multiple unique key per table

-- i can able of create multiple unique key

-- unique value contain null values

-- but not allowed duplicate data

-- by default null

Foreign Key

-- It is used to connect two tables.

-- It makes sure the data between tables stays consistent.

-- primary key of one table could be foreign in another table

-- one column of one table its links with another table can be make as a foreign key

-- A Foreign Key is a column in one table that links to the Primary Key of another table.

-- parent child relationship / relationship established

-- interlink the two table

-- foreign key allow us to establish child and parent relation between the table inside the database

-- this way by linking the two table i can access the additional information from the another table

-- common column

-- parent class exist first

-- retrieve the data from two table at a time

-- Real-Life Example

Imagine a School Database:

Students Table

| StudentID (PK) | Name | Class |

|----------------|--------|-------|

| 1 | Riya | 10 |

| 2 | Aman | 9 |

Here, StudentID is the Primary Key (unique for every student).

Fees Table

| FeeID | StudentID (FK) | Amount |

|-------|----------------|--------|

| 101 | 1 | 5000 |

| 102 | 2 | 4500 |

In the Fees Table, StudentID is a Foreign Key because it refers back to the Primary Key (StudentID) in the Students table.

Real-Life Meaning

In real life, a Roll Number (Primary Key in Student table) is used in the Fee Record table to know which student paid the fee.

Without a Foreign Key, the system won't know which fee belongs to which student.

Syntax:

Constraint name of key Foreign key (column name)

references table_name(column_name)

--> Constraint fk_key Foreign key (deptid)

references department(deptid)

==========

auto_increment attribute / constraint

-- we can apply the auto_increment with primary

-- autu_increment attribute we can able to apply only primary key column

-- it will work with only primary column no one others

-- Automatically insert the value

-- by default 1 it will visible

-- Que : how many column we can apply auto_increment attribute

-- ans. only one and that column should be primary key

Types of Constraints in SQL

Constraint	Purpose	Example
NOT NULL	Makes sure a column **cannot have NULL (empty)** values	`Name VARCHAR(50) NOT NULL`
UNIQUE	Makes sure **all values are different** in a column	`Email VARCHAR(100) UNIQUE`
PRIMARY KEY	Uniquely identifies each row (cannot be NULL + must be unique)	`EmpID INT PRIMARY KEY`
FOREIGN KEY	Links two tables (creates relationship)	`DeptID INT FOREIGN KEY REFERENCES Departments(DeptID)`
CHECK	Ensures values meet a condition	`Age INT CHECK (Age >= 18)`
DEFAULT	Sets a default value if none is given	`City VARCHAR(30) DEFAULT 'Pune'`

Example

```
CREATE TABLE Employees (
  EmpID INT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  Age INT CHECK (Age >= 18),
  Email VARCHAR(100) UNIQUE,
  City VARCHAR(30) DEFAULT 'Pune'
);
```

Explanation:

- `EmpID` → uniquely identifies each employee
- `Name` → can't be empty
- `Age` → must be 18 or above
- `Email` → no duplicates allowed
- `City` → if not provided, defaults to "Pune"

Why Constraints Are Important

- Keep data **accurate** (no wrong or duplicate info)
- Keep data **reliable** (follow business rules)
- Keep data **secure** (relationships between tables stay valid)

==========