

MySQL

* Database :- Database is a collection of data in a format that can be easily accessed (digital).

A software application used to manage our DB is called DBMS (Database Management System)

* Types of Databases :-

Relational

Data stored in tables

Eg. MySQL, ORACLE,
SQL Server, PostgreSQL.

Non-Relational (NoSQL)

data not stored in tables

Eg:- MongoDB

Swipe <<<

* What is SQL?

Structured Query Language

SQL is a programming language used to interact with relational databases.

It is used to perform ~~CRE~~CRUD operations:

Create

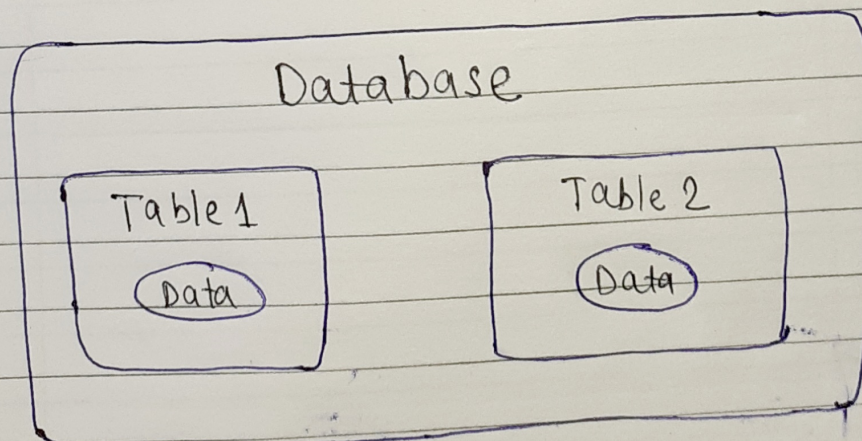
Read

Update

Delete

* Database Structure

Swipe <<<



* What is a Table?

Student Table

Roll no	Name	class	DOB	Gender	city	Marks
1	Nanda	X	1995-06-06	M	Agra	551
2	Neha	XII	1993-05-07	F	Delhi	624
3	Nishant	XI	1994-12-12	M	Pune	629
4	Mohan	XII	1993-06-12	M	Mumbai	377

* Creating Our First Database:~

Our first SQL Query

CREATE DATABASE db_name;

DROP DATABASE db_name;

* Creating our first Table:~

USE db_name;

Swipe <<<

Syntax:-

```
CREATE TABLE table_name (  
    column_name1 datatype constraint,  
    column_name2 datatype constraint,  
    column_name3 datatype constraint  
);
```

Example :-

```
CREATE TABLE student (  
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    age INT NOT NULL  
);
```

Swipe <<<

Swipe <<<

* SQL Datatypes

Data type	Description	Usage
CHAR	can store characters of fixed length	CHAR(50)
VARCHAR	can store characters up to given length	VARCHAR(50)
BLOB	can store binary large object	BLOB(1000)
INT	integer (-2,147,483,648 to 2,147,483,647)	INT
TINYINT	integer (-128 to 127)	TINYINT
BIT	can store x-bit values, x can range from 1 to 64	BIT(2)
FLOAT	Decimal number	FLOAT
DOUBLE	Decimal number	DOUBLE
BOOLEAN	Boolean values 0 & 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE

* Signed & Unsigned

TINYINT (-128 to 127)

TINYINT UNSIGNED (0 to 255)

Swipe <<<

* Types of SQL Commands:~

- DDL (Data Definition Language):~ create, alter, rename, truncate & drop.
- DQL (Data Query Language):~ select
- DML (Data Manipulation Language):~ select, insert, update & delete
- DCL (Data Control Language):~ grant, & revoke permission to users.
- TCL (Transaction Control language):~ start transaction,

commit, rollback etc.

* Database related Queries :-

CREATE DATABASE db_name;

CREATE DATABASE IF NOT EXISTS db_name;

Eg:-

CREATE DATABASE IF NOT EXISTS college;

DROP DATABASE db_name;

DROP DATABASE IF EXISTS db_name;

SHOW DATABASES;

SHOW TABLES;

Swipe <<<

* TABLE related Queries:-

i) Create

Eg:-

```
CREATE TABLE student(
```

```
rollno INT PRIMARY KEY,
```

```
name VARCHAR(50)
```

```
);
```

ii) Select & View ALL columns

Eg:-

```
SELECT * FROM table-name;
```

iii) Insert

Eg:-

```
Insert INTO table-name (colname1,  
colname2) values  
col1-v1, col2-v1),  
col1-v2, col2-v2);
```

Swipe <<<


```
INSERT INTO student (rollno, name) VALUES
(101, "Karan"),
(102, "Arjun");
```

* KEYS

Swipe <<<

Primary key

It is a column (or set of columns) in a table uniquely identifies each row. (a unique id)
There is only 1 PK & it should be NOT NULL.

Foreign key

A foreign key is a column (or set of columns) in a table that refers to the primary key in another table.

There can be multiple FKs. FKs can have duplicate & null values.

Eg:-

table 1 - Student

table 2 - City

id	name	cityId	city
101	Karan	1	Pune
102	Arjun	2	Mumbai

id	city-name
1	Pune
2	Mumbai

* Constraints

SQL constraints are used to specify rules for data in a table.

i) NOT NULL:~ columns cannot have a null value

Eg:~ col1 int NOT NULL

ii) UNIQUE:~ all values in column are different

Eg:~ col2 int UNIQUE

Swipe <<<

iii) PRIMARY KEY:~ makes a column unique & not null but used only for one.

Eg:~ id INT PRIMARY KEY

iv) FOREIGN KEY:~ prevent actions that would destroy links between tables.

CREATE TABLE temp {

cust_id int,

FOREIGN KEY (cust_id) references customer(id)

);

v.) DEFAULT :~ sets the default value of a column

Salary INT DEFAULT 25000

vi.) CHECK :~ it can limit the values allowed in a column

CREATE TABLE city (

id INT PRIMARY KEY,

city VARCHAR(50),

age INT,

CONSTRAINT age_check CHECK (age >= 18 AND city = "Delhi")

);

Swipe <<<

Page 12

Create this Sample data

```
CREATE DATABASE college;  
USE college;
```

```
CREATE TABLE student (
```

```
rollno INT PRIMARY KEY,  
name VARCHAR(50),  
marks INT NOT NULL,  
grade VARCHAR(1),  
city VARCHAR(20)  
);
```

Swipe <<<

```
INSERT INTO student (rollno, name, marks,  
grade, city) VALUES  
(101, "anil", 78, "C", "Pune"),  
(102, "bhumika", 93, "A", "Mumbai"),  
(103, "chetan", 85, "B", "Mumbai"),  
(104, "dhruv", 96, "A", "Delhi"),  
(105, "emanuel", 12, "F", "Delhi"),  
(106, "farah", 82, "B", "Delhi");
```


* Select in Detail

Used to select any data from the database

Syntax :-

```
SELECT col1, col2 FROM table_name;
```

```
SELECT * FROM table_name;
```

* Where Clause

To define some conditions

```
SELECT col1, col2 FROM table_name  
WHERE conditions;
```

Eg:-

```
SELECT * FROM student WHERE  
marks > 80;
```

Swipe <<<

* Using Operators in WHERE

Arithmetic Operators: ~ + (addition), - (subtraction), * (multiplication), / (division), % (modulus).

Comparison Operators: ~ = (equal to), != (not equal to), >, <=, >, >=

Logical Operators: ~ AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY

Bitwise Operators: ~ & (Bitwise AND), | (Bitwise OR)

Swipe <<<

* Operators.

- AND (to check for both conditions to be true)
↳ SELECT * FROM student WHERE marks > 80 AND city = "Mumbai";
- OR (to check for one of the conditions to be true)
↳ SELECT * FROM student WHERE marks > 90 OR city = "Mumbai";
- BETWEEN (selects for a given range)
↳ SELECT * FROM student WHERE city IN ("Delhi", "Mumbai");
- NOT (to negate the given condition)
↳ SELECT * FROM student WHERE city NOT IN ("Delhi", "Mumbai");

* Limit Clause

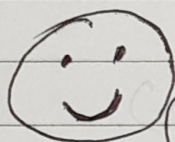
Sets an upper limit on number of (tuples) rows to be returned.

↳ SELECT * FROM student LIMIT 3;

* Order By Clause

To sort in ascending (ASC) or descending order (DESC)

↳ SELECT * FROM student ORDER BY city ASC;



By default order by clause follows ascending order if you ^{not} mention ASC.

Swipe <<<

* Aggregate Functions

Aggregate functions perform a calculation on a set of values, and return a single value.

- COUNT()

- MAX()

- MIN()

- SUM()

- AVG()

Examples:-

→ SELECT max(marks) FROM student;

→ SELECT avg(marks) FROM student;

Swipe <<<

FOLLOW ME



@Lovely Kumari