DAISY PANIMARIYA

PAGE NO.: 1
DATE:

everyday new day!

Database

Intro

Structured Query Language (SQL)

* ~~MySQL → Database~~
* SQL → Language

*SQL is used to ~~tool used to~~ access or modify the data in the database.

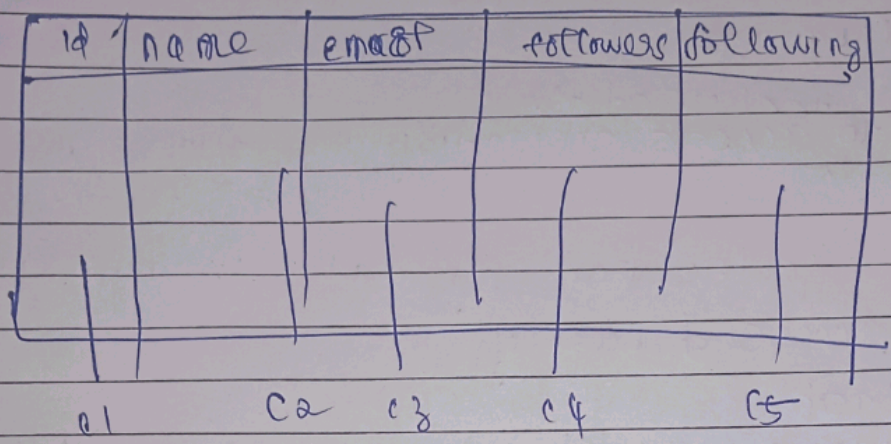
* ~~Database : collection of data in a format~~ that can be accessed easily.
* Why Database?
  * can store large data (expand depending upon usage)
  * features like security, scalability etc
  * Easier to Insert, update or delete data, search data.

SQL	NOSQL
*Relational Database	*Non Relational Database
* Data stored in Tables	* Data stored in document/ key-val pair/ graphs etc
* Eg :- MySQL, Oracle, PostgreSQL	* Eg :- MongoDb, cassandra, Neo4j etc

SQL - is a programming language used to Interact with database

PAGE NO.: 2
DATE: / /

Table in SQL



Column => design (schema) Structure of the table

row => information of users (each user) => tuple

```
CREATE DATABASE db_name; # create database
DROP DATABASE db_name; # Delete database
USE db_name; # access database
```

Table creation

```
CREATE TABLE t_name
{
    column_name 1 datatype constraint,
    column_name 2 datatype constraint,
};
```

Advised to use uppercase commands

PAGE NO.: 8
DATE: / /

INSERT

SYNTAX

```
INSERT INTO $table_name
VALUES
(                    ),
(                    );
```

~~print the table~~

```
SELECT * FROM table_name;
```

TABLE Queries

- Create
- Insert
- Update
- Alter
- Truncate – Data delete
- Delete – Table delete

CREATE TABLE

SYNTAX:

```
CREATE TABLE table_name
(
    column_name 1 datatype constraint,
    column_name2 datatype constraint

);
```

Eg: name

Datatypes

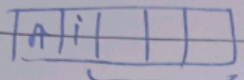
PAGE NO. :
DATE : / /

STRING

fixed length

* char => string (0-255) -> char(50)
* VARCHAR => string (0-255) given length -> VARCHAR(50)

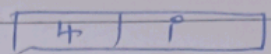
CHAR(5)



reserved

waste

VARCHAR(5)



no reservation

* ~~BLOB => string (0-65535)~~

INTEGER

- INT
- TINYINT (-128 to 127)
- BIGINT

BIT

till 64 size
64 bit values
ex: BIT(2)

FLOAT

Precision to 23 digits

DOUBLE

Precision 24 to 53 digits

BOOLEAN

0 or 1
T or F

NOT exist in SQL but achieved using TinyInt

DATE

Format
YYYY-MM-DD
1000-01-01
9999-12-31

YEAR

4 digits
1901 to 2155

UNSIGNED 0-256
TinyINT -128 to 127
-129 ✗
128 ✗

NO TINYINT UNSIGNED
no negative number
{1 -128} 128
1 to 127 + 128 ~~spaces~~ numbers

Constraints

Rules for data in the table

PAGE NO.:
DATE: / /

- NOT NULL : columns cannot have NULL values
- UNIQUE : all values in column are different
- Default : sets the default value of a column
- check : it can limit the values allowed in a column

example:-

```
(name VARCHAR(30) NOT NULL,
email VARCHAR(50) UNIQUE,
followers INT DEFAULT = 0,
following INT DEFAULT = 0);
```

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

PRIMARY KEY:
makes a column unique & not NULL but used only for one.

Key
⇓
Unique or special columns

```
CREATE TABLE table_name
(       INT
    id NOT NULL,
    PRIMARY KEY(id)
);
```

```
id INT PRIMARY KEY  → method I
PRIMARY KEY(id)     → method II
```

FOREIGN KEY : (Link Table)

Prevent actions that would destroy links between tables

Student

id	name	t_id
101	adam	1
102	bob	2

Teacher

id	name
1	Suraj
2	Shyam

└ Primary key. └ Foreign key └ Primary key.

~~Primary key~~ of another table is foreign key when it's inherited.

~~SYNTAX~~

FOREIGN KEY (t_id) REFERENCE Teacher (id)

One PK and Many FK possible.

INSERT

```
INSERT INTO table_name
Values
(          ),
(          );
```

```
INSERT INTO table_name
(column_name1, column_name2)
VALUES
(column1_v1, col1_v2)
(col2_v1, col2, v2);
```

* Select

All/whole SELECT * FROM table_name;

* particular col

SELECT col_name FROM table_name;

* DISTINCT

Unique element

SELECT DISTINCT Age FROM table_name;

* where clause

Select age from table_name
where condition;

* Arithmetic: +, −, *, /, %
* Comparision: =, !=, >, >=, <, <=
* Relational => AND, OR, NOT, BETWEEN, IN, ALL, LIKE, ANY
* Bitwise operators: &, |

* Limit Clause:

Select col_name FROM table_name
LIMIT 2;

* Order By clause

Select col1 from tablename
ORDER BY colname ASC;

SELECT col1 FROM tab_name
ORDER BY COLUMN_NAME DESC;

Default Ascending Order

Aggregate functions

- calculate set of values and return a single value
- Inbuilt functions

COUNT() MAX() MIN() SUM()
AVG()

GROUP BY CLAUSE

⇒ Group rows that have same values into summary rows
⇒ It collects data from multiple records and groups the result by one or more column.
⇒ Generally we use group by with some aggregation functions.

Having clause only for group by (necessary)

* Similar to where applies some condition on rows
* only used after group by

```
Select col1 col2
From tab6_name
Group by col_name
Having condition
```

General Order for clause

PAGE NO.:
DATE: / /

```
Select col_name(s)
From table_name
Where condition
Group by column(s)
Having condition
Order
ORDER BY column(s) ASC;
```