classmate
Date 26/9/2024
Page

SQL

⇒ Database is collection of data in a format that can be easily accessed (Digital).

⇒ A Software application used to manage a DB is called DBMS

⇒ Type of Databases.

Relational	Non - Relational
⇒ Data Stored in table	⇒ Data Not Stored, it has.
My SQL	Mongo DB

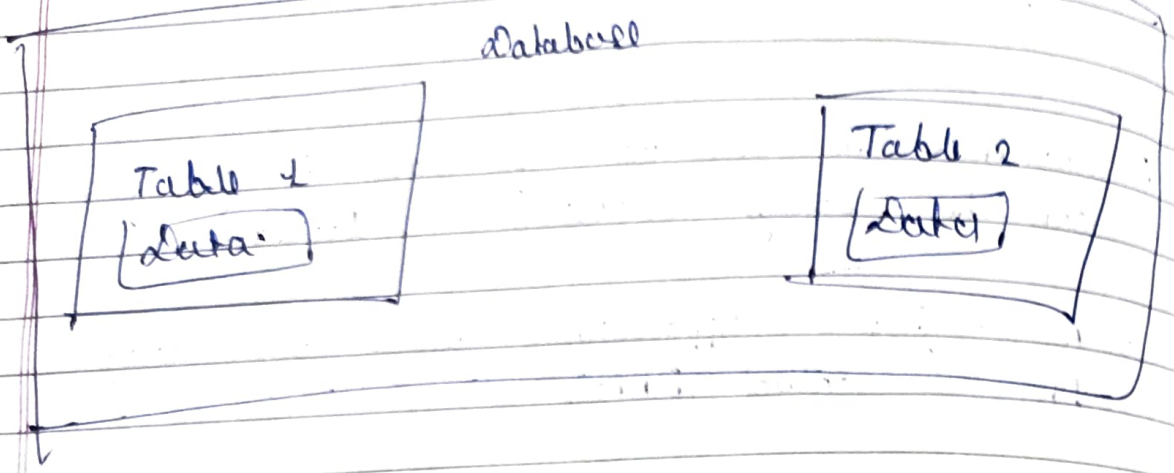
⇒ What is SQL

⇒ SQL Stand for Structured Query Language.

⇒ SQL is a programming language used to interact with relational database.

⇒ It use to perform CRUD operation
Create
Read
Update
Delete

Database Structure:



⇒ Can be inter related table.

⇒ It contain
Rows ⇒ And columns ↓

Roll	Name	Class
1	A	I
2	B	II
3	C	III

Creating our first Database

⇒ Create Database

```
CREATE DATABASE DB_Name;
```

~~Creat~~

⇒ Drop Database (Deleting Database)

```
DROP DATABASE DB_Name;
```

⇒ ~~Use table (using)~~

⇒ To use Database.

```
USE db_Name;
```

⇒ Creating Our first Table

```
USE db_name;

CREATE TABLE table_name (
    Coulum_name1 datatype constraint,
    .
    .
    .
);
```

```
CREATE TABLE student (
    id INT Primary key,
    Name VARCHAR(50),
    age INT Not NULL
),
```

⇒ To Insert Data.

~~Insert~~ `INSERT INTO student VALUES(1, "Amuan", 26);`

`INSER INO student VALUES(2, "SHEDHA", 26);`

To print whole table

```
SELECT * FROM table_Name;
```

SQL Data Type

1	CHAR	⇒ Character related type (0-255) Fix length CHAR(50)
* 2	VARCHAR	⇒ " can store variable
3	BLOB	⇒ can store binary large obj. (0, 65535) BLOB(4000)
4	INT	⇒ (-, +)
5	TINYINT	⇒ (-128 to 127)
6	BIGINT	⇒ Large Range
7	BIT	⇒ can store x-bit values Range: (1 to 62) BIT(2) ⇒ (0, 2) BIT(2) - {00 01 10 11}
8	FLOAT	⇒ Decimal value
9	Double	⇒ Decimal.
10	BOOLEAN	⇒ 0 or 1
11	DATE	⇒
12	YEAR	

classmate
Date
Page

⇒ SQL Data Type Signed & unsigned

TINYINT UNSIGNED (0 to 255)

⇒ Range changed

TINYINT (-128 to 127)

⇒ Type of SQL Comamand

⇒ DDL (Data Definition Language) :-
Create, alter, rename, truncate & drop

⇒ DQL (Data Quary Lanuage) :-
Select

⇒ DML (Data Manipulation language) :-
insert, update & delete

⇒ DCL (Data control language) :-
grant & revoke permission to users

⇒ TCL (Transaction control language)
Start transaction, commit, roll back &

Database Related Queries.

⇒ CREATE DATBASE db_Name ;

CREATE DATABASE IF NOT EXISTS db_Name ;

⇒ ~~CREA~~ DROP DATABASE db_Name ;

DROP DATA BASE IF EXISTS db_Name ;

⇒ ~~Show~~ SHOW DATABASES ; { All Database. }

⇒ SHOW TABLE ; { To ~~sess~~ See table which we are using }

Table Related Queries

~~Create~~
CREATE TABLE table_Name (
    Column_Name1 columtype Constraint ,
    Colum_Name2 datatype constraint ,
);

{ Create ~~DB~~ table student (
    RollNo INT Primary key,
    Name VARCHAR (30 }

⇒ To ~~drop tap~~

⇒ To delete table

DROP TABLE ~~Student~~ table_Name;

⇒ Select & View ALL Colums

SELECT * FROM table_Name;

⇒ Insert Data into Queries.

```
INSERT INTO table_name
  ( column1 , column2)
VALUES
  ( col1-v1 , col2-v2),
  ( col1-v2 , col2-v2);
```

} To add Multiple Data

```
INSERT INTO Student
  (rollno, name)
VALUES
  (101 , "Kuruva")
  (102 , "Arjuna")
```

```
INSERT INTO table_name value ( " " )
```

To Add Single data.

Keys

⇒ Primary key

- It is a column (or set of columns) in a table that 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 primary key in the Another table
- There can be multiple FKs.
- FKs can have duplicate & null values

Constraints.

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

- NOT NULL —
  Columns cannot have Null value
  Ex:- col1 INT NOT NULL
- UNIQUE —
  all values in column are different
  Ex 1 ⇒ col2 INT UNIQUE

- Primary key (PRIMARY KEY)
  make a column unique & not Null but used only for one.

way 1
```
id INT PRIMARY KEY
```

Ex:

way 1
```
CREATE TABLE temp (
    id int not Null, Primary Key(id)
);
```

> Combian of two primary key

```
Create Table temp (
    id INT,
    name VARCHAR (50),
    city VARCHA (20),
    PRIMARY KEY (id, Name).
);
```

- FOREIGN KEY

Prevent actions that would destroy links between table.

```
CREATE TABLE temp (
    Cust-id int,
    FOREIGN KEY ((ust-id) references
        customer(id)
);
```

- DEFAULT :-
  Set the default value of a column.

  > Salary INT DEFAULT 25,000

- CHECK :-
  It can limit the values allowed in a column.

WAY 1 =>

```
CREATE TABLE city (
    Id IN PRIMARY KEY,
    city VARCHAR(50),
    age INT,
    CONSTRAINT age-check
    CHECK ( age >= 18 AN city = 'Delhi')
```

WAY 2 >

```
Create TABLE ID New Tab (
    age INT CHECK (age >= 13)
);
```

Select

using to select any data from the database.

Basic Syntax

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

> To Select All Column

```
SELECT * FROM table_name;
```

> To find DISTINCT
↳ Unique

```
SELECT DISTINCT col_Name FROM table_Na
```

⓪ Where Clause :-

To define some conditions.

Syntax

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

Example:

```
SELECT * FROM table_Name WHERE mark > 80;
SELECT * FROM table_Name WHERE city = "M
```

AND is used to combine two condio.

```
Select * From table_name WHERE mark > 80 AND
                              city = "Mumbai"
```

Where Clause

using Operators in WHERE

- Arithmetic Operators: +, -, *, /, %
- comparison Operator: =, !=, >=, <, >, <=
- Logical operato- : AND, OR, NOT, IN, BETWEEN

```
SELECT * FROM Name_table where colum_namt 10 > 30;
```

- Between operator

```
SELECT * FROM Name_table WHERE Marks BETWEEN
                              80 AND 90;
```

- IN operator

(matches any values in the list)

```
SELECT * FROM Name_table where city IN ("Delhi",
                                        "Mumbai")
```

- NOT IN operator.

(To negate the given condition

```
SELECT * FROM Name_table where city NOT IN
                              ("Delhi", "Mumbai")
```

Limit Clause.

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

SELECT * FROM Name_Table LIMIT number;

Syntax:

```
SELECT col1, col2, FROM table_Name
    LIMIT Number;
```

where AND LIMIT combine use Example.

```
SELECT *
FROM Student
WHERE marks > 75
LIMIT 3;
```

classmate
Date
Page

Order By Clause. ORDER BY

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

```
SELECT * FROM student
ORDER BY city ASC ;
```

```
SELECT col1, col2 FROM table_Name
ORDER BY col_name(s) ASC ;
```

Aggregate function.

Aggregate function Perform calculation on a set of values, and return a single value.

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

Get Max Marks

```
SELECT MAX(marks)
from student
```

Get Average

```
SELECT avg(marks)
from student
```

Syntax

```
SELECT Aggregate func(col)
from table_name.
```

Group By Clause. (GROUP BY)

> Groups rows that have the 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 which any aggregation function.

Count number of students in each city.

```
SELECT city , count(name)
FROM student
GROUP BY city;
```

If there is No Aggreiate function then we have to put that column name in GROUP By

```
SELECT city , name , count (name)
from student
GROUP BY city , name;
```

Q Practice Question

Write the query to find avg marks in each city in ascending order.

```
SELECT city, avg (marks)
FROM student
GROUP BY city
ORDER BY city asc;
```

- **Having Clause** (HAVING)

> Similar to where i.e applies some condition on row

> used when we want to apply any condition after grouping.

> Count number of student in each city where max marks crosses 90.

```
SELECT count(name), city
FROM student
GROUP BY city
HAVING max(marks) > 90;
```

General Order.

SELECT Column(s)
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition.
ORDER BY column(s) ASC

SFWGHO

Select city
FROM Student
where grade = "A"
GROUP BY city
HAVING Max(marks) >= 99
ORDER BY city DESC

Table Related Queries.

1) Update (UPDATE)

= To update existing rows

Examp

```
UPDATE  Student
SET     grade = "O"
WHERE   grade = "A";
```

Syntax

```
UPDATE  table_name
SET     col1 = val1, col2 = val2
WHERE   condition;
```

Ex = 2,

```
Update  Student
set.    marks = 82
where   rollno = 105
```

Ex 3 → Set New Age

Where marks Between 80 AND 90 ;

Ex ⇒ 4

```
update  student
set     marks = marks + 40;
```

2) Delete (DELETE FROM)

To Delete existing row

Syntax ⇒

```
DELETE FROM table_name
WHERE condition ;
```

```
DELETE FROM Student
WHERE marks > 30;
```

DELETE FROM table name → whole column data will be delete

Revisiting ~~foreign~~ foreign key

Course

id	name
101	Science
102	English
103	Hindi

Parent Table

← FK ←

teacher

id	Name	department (FK)
101	Adam	101
102	Bob	103
103	Chris	102
104	Donald	102

Child Table

- On Delete Cascade

  When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key

- On Update Cascade

  When we create a foreign key using UPDATE CASCADE referencing rows are updated in the child table when the referenced row is updated in the pres

```
CREATE TABLE Student (
    id INT PRIMARY KEY
    CourseID INT
    FOREIGN KEY (CourseID)
    REFERENCES table_Name ( column_name
    ON DELETE CASADE
    ON UPDATE CASCADE
);
```

3) Alter :-

to Change the Schema.

> ADD column

```
ALTER TABLE table_name
ADD COLUMN column_name datatyp constrai;
```

> Drop Column

```
ALTER TABLE table_name
DROP COLUMN Column_name;
```

> Rename Table

```
ALTER TABLE table_name.
RENAME TO New_table_name;
```

> Change CHANGE column (rename)

A

```
ALTER TABLE table_name
CHANGE COLUMN old name New_name -
- New_datype
- New_constraint
```

> ~~Modi~~ MODIFY Column (modify datatype /constraint)

```
ALTER TABLE table_name
MODIFY col_name New_datatype New_const.
```

Table Related Queries

> ~~Truncat~~ Truncate (to delete table data)

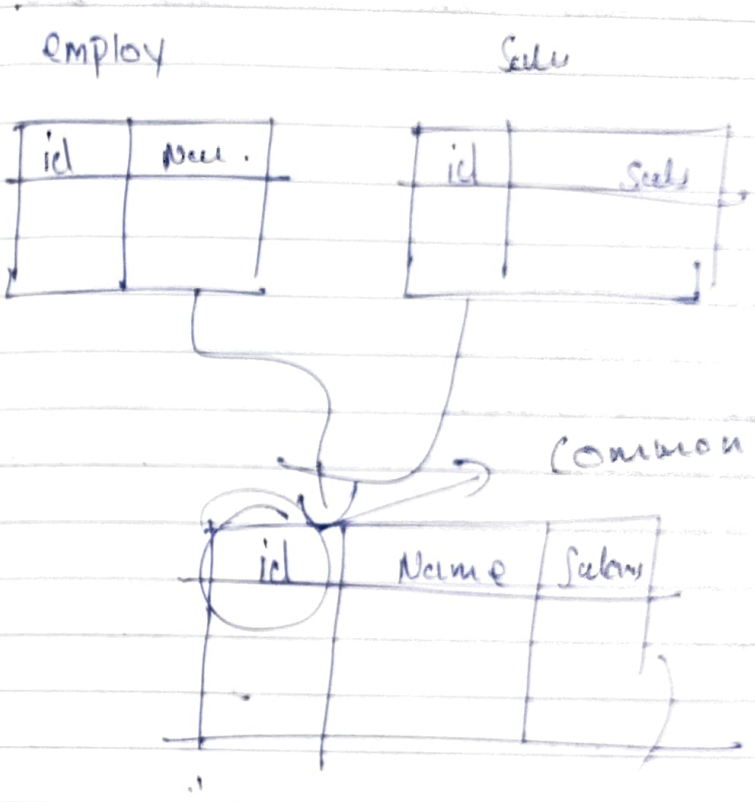
```
TRUNCATE TABLE table_name ;
```

Delete all data of the Table

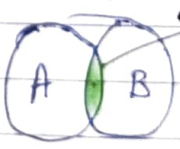
Drop Delete the whole Table.

SQL (Joins in SQL)

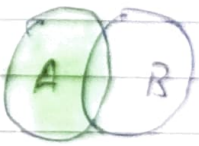
Join is used to combine rows from two or more table, based on a related column between them.



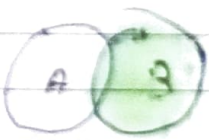
Type of Joinds. (Venn diagram.



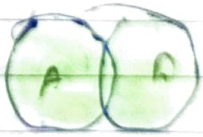
Inner Joint



Left Joint



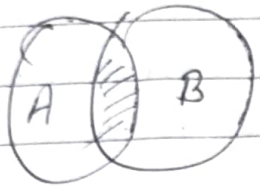
Right Join



Full Join

Outer Join

1) Inner Joint



Return record that have matching value in both table

* Sytax :-

```
SELECT column'(s)
FROM tableA
INNER JOIN TableB
ON tableA.col-Name = table.col-Name.
```

Student

id	name
101	a
102	b
103	c

Course

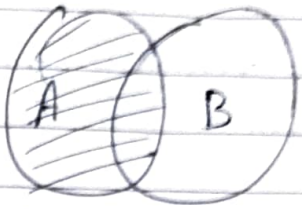
id	cours
102	engl's
103	Scin
104	Comsi

```
) Select *
) FROM Student
) INNER JOIN Course
) ON Student.~~Student~~id = course.id
```

Inner

id	Name	cours
102	b	engl's
103	c	Scino

2) Left Joint.



Return all Records from the left table, and the matched records from the Right tables.

* Sytax :-

```
SELECT  Column(s)
FROM table A
LEFT JOINT table B
 ON TableA . col-name = table . col-name .
```

* Example.

```
SELECT *
FROM Student as c
LEFT JOIN course as c
 ON  S . Student-id = c . Student-id ;
```

Student

Student	Id
101	adam
102	bob
103	casey

Course

Student Id	course
102	English
105	Math
103	Sci...
107	Comp Sc

Result

Student-id	Name	course
101	adam	Null
102	bob	eng
103	casey	Sci

3) Right Join.

Return all records from the right table and matched record of left Table

* Sytax :-

```
SELECT Columns)
FROM table A
RIGHT JOIN table B
 ON table A. col-name = tableB. col-name.
```

* Exampl

```
SELECT *
FROM Student as S
RIGHT JOIN course as C
ON S. Student-id = C. student-id;
```

Student	Course	Name
102	english	bob
105	Math	Null
103	Scien	Casey
107	Computer Sir	Null

4) Full Join.

Return all records when there is a match in either left or Right table.

* Sytax :-

```
SELECT * from Student as a
LEFT JOIN course as b
ON a.id = b.id
UNION
SELECT * FROM Student as a
RIGHT JOIN course as b
ON a.id = b.id;
```

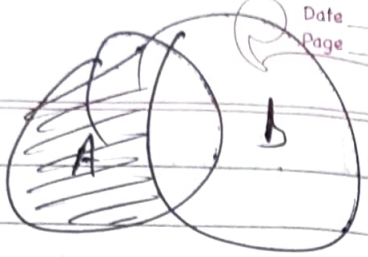
LEFT JOIN
UNION
RIGHT JOIN

* Example.

Student	Name	Course
101	adam	Null
102	bob	English
103	Casey	Science
105	Null	Math
107	Null	Computer Sci.

Extra

5) Left Exclusive Joint



Syntax :-

```
SELECT *
FROM Student as a
LEFT JOIN Course as b
 ON a.id = b.id
WHERE b.id IS NULL
```

6) Right Exclusive Joint

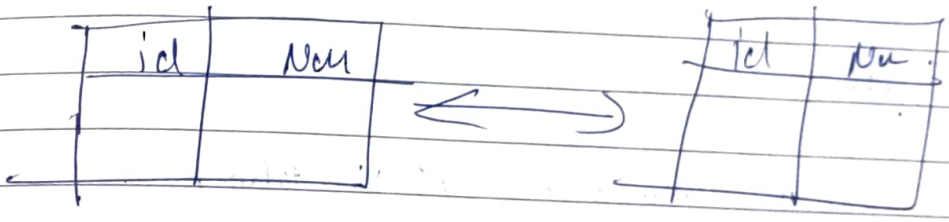

```
Select *
FROM Student as S
RIGHT JOIN Course as C
ON S.Student id = C.Student.id
 where a.id is NULL
```

Self Join

It is a regular join but table is joined with it self.

Syntax:

```
SELECT column(s)
FROM table as a
JOIN table as b
 ON a.col_name = b.col_name;
```



8) Union.

It is used to combine the result-set of two or more SELECT statements.

Gives UNIQUE record.

To use it:
- every SELECT should have same no. colum.
- columns must have similar data type.
- columns in every SELECT should be in same order.

Syntax:

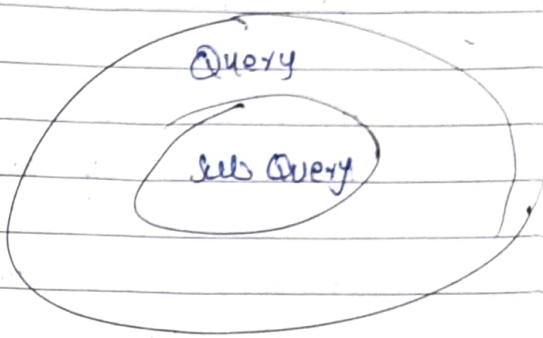
```
SELECT column(s) FROM table A
UNION
SELECT Column(s) FROM table B
```

SQL Sub Queries.

⇒ A subquery or Inner query or the Nested query is a query within another SQL query
⇒ It involves 2 Select statement.

Syntax

```
Select column(s)
FROM table name
WHERE col-name operator
(Sub-query);
```



Example

Given name of all student who scored more than Class average.

Step 1. Find the avg of class
Step 2. Find the names of students with marks > avg

```
Select Avg(marks)
from Student
```

Step (1)

Rollno	name	Marks
101	anil	78
102	abhishek	53
103	chetan	85
104	dravv	96
105	emanuel	92
106	farukh	82

```
Select name, marks
from Studnt
where marks > 87.667
```

Step 2

Step (1) And Step (2) Com.

```
SELECT name, marks
from Studen
WHERE mark > (SELECT Avg(marks) from Studen);
```

MY SQL views

A views is a virtual table based on the Result-Set of an SQL Statement

```
CREATE VIEW view2 AS
SELECT Rollno, name FROM Studen:

SELECT * FROM view1
```