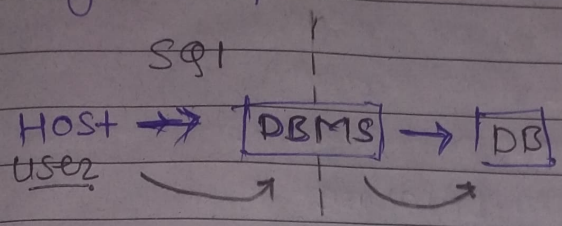
:- Database -:

Database is collection of data in a formate that can be easily accessd (Digital)
A ~~softwear~~ software application used to manage our DB is called DBMS (Database manggment system)



* Types of Databases.
1) Relational (SQL) 2) ~~DB~~ Non-relational (NOSQL)

1] Relational
Data stored in tables

*** We use SQL to work with relational DBMS
Ex. MySQL, SQL Server, ORACLE, Postgre SQL etc

2) Non-relational (NOSQL)
data not stored in tables.

Ex. mongo DB, etc.

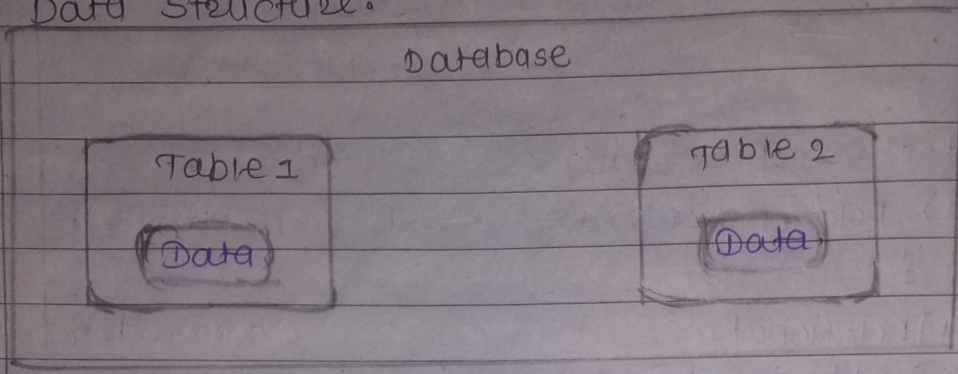
- What is SQL ?
Structured Query Language.
SQL is a ~~pro~~ programming language used to interact with relational databases.

It is used to perform CRUD operations.
- Create
- Read
- Update
- Delete

SEQUEL
Structured
English
Query
Language

SQL
Structured
Query
Language.

Data Structure.



Column => Structure / schema (design)
row → individual data.

Creating our First database
our First SQL query

~~Create~~
CREATE DATABASE db-name;
DROP DATABASE db-name;

Creating our First Table
USE db-name;
CREATE TABLE table_name (
column_name1 datatype constraint,
column_name2 datatype constraint,
Column_name2 datatype Constraint
);

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

~~(22'48)~~

SQL Datatypes

- They define the type of values that can be stored in a column.

Datatype	Description	Usage.
~~CHAR~~ CHAR	String (0-255), can store characters of fixed length	CHAR (50)
VARCHAR	String (0-255), can store characters up to given length	VARCHAR (50)
BLOB	String (0-65535), 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
BIGINT	integer (-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	BIGINT
BIT	can store x-bit values. x can range from 1 to 64	BIT (2)
FLOAT	Decimal number - with precision to 23 digits	FLOAT
DOUBLE	Decimal number - with 24 to 53 digits	DOUBLE
BOOLEAN	Boolean value 0 or 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
YEAR	year in 4 digits format ranging from 1901 to 2155	YEAR

- SQL Datatypes
  signed & Unsigned

  TINYINT UNSIGNED (0 to 255)
  TINYINT (-128 to 127)

- 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): geant & revoke permision to users.
  TCL (Transaction Control Language): Start transaction, commit, rollback etc.

- Database related Queries
  CREATE DATABASE db-name;
  CREATE DATABASE IF NOT EXISTS db-names

  CREATE DATABASE IF NOT EXISTS College;

  DROP DATABASE db-name;
  DROP DATABASE IF EXISTS db-name;

  SHOW DATABASES;
  SHOW TABLES;

~~TABLE~~

Table related queries.

Create

```
CREATE TABLE table-name (
   column_name1 datatype constraint,
   column_name2 datatype constraint,
);
```

```
CREATE TABLE student (
  rollno  INT PRIMARY KEY,
  name VARCHAR (50)
);
```

Table related queries.

Select & view All columns

→ All select

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

```
SELECT * FROM student;
```

Table

Table related queries.

Insert

```
INSERT INTO table_name
 (colname1, colname2)
 VALUES
 (col1-v1, col2-v1),
 (col1-v2, col2-v2);
```

	col1	col2
v1	101	kazan
v2	102	arjun

```
INSERT INTO student
 (rollno, name)
 VALUES
  (101, "kazan"),
  (102, "arjun");
```

* INSERT INTO Stedew VALUES (104, "Laxman");

Q] Create a database for your company named XYZ.

→ Step1: Create table inside the DB to store employee info (id, name and salary).

Step2: Add following information in the DB:

1, "adam", 25000
2, "bob", 30000
3, "casey", 40000

Step3: Select & view all your table data.

Codes

```
CREATE DATABASE xyz;

CREATE DATABASE xyz_company;
USE xyz_company;

CREATE TABLE employee (
id INT PRIMARY KEY,
name VARCHAR(100),
salary INT
);

INSERT INTO employee
(id, name, salary)
VALUES
(1, "adam", 25000),
(2, "bob", 30000),
(3, "casey", 40000);

SELECT * FROM employee;
```

- 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 the primary key in another table there can be multiple FKs.

FKs can have duplicate & null values.

- Example of keys.

table1 - student

id (PK)	name	cityId (FK)	city
101	karan	1	pune
102	arjun	2	Mumbai
103	ram	1	pune
104	shyam	9	Delhi

table2 - city

id (PK)	city_name
1	pune
2	~~Mumbai~~
3	Delhi

(01:01:10)

- Constraints

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

~~NOT NULL columns cannot have a null col1 int NOT~~ ~~value~~ ~~value~~

NOT NULL columns cannot have a null value col1 int NOT NULL

UNIQUE all values in column are different.

col2 int UNIQUE

PRIMARY KEY makes a column unique & not null but used only for one id put PRIMARY KEY

```
CREAT TABLE temp (
 id iwt not null,
PRIMARY KEY (id)
);
```

Constraints.

FOREIGN KEY: Prevent actions that would destroy links between tables.

```
CREATE TABLE temp (
    cust_id iwt,
    FOREIGN KEY (cust_id) Reference customer (id
);
```

DEFAULT: sets the default value of a column

```
salary INT DEFAULT 25000
```

- CHECK it can limit the value allowed in a column.

```
CREAT TABLE city (
id INT PRIMARY KEY,
city VARCHAR(50),
age INT,
CONSTRAINT age-check CHECK
  (age >=18 AND city = "Delhi")
);
```

```
CREAT TABLE newTab(
age INT CHECK (age >=18)
);..
```

CO1.10.35).

- **Constraints:**
  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")
);
```

```
CREATE TABLE newTAB (
    age INT CHECK (age >= 18)
);
```

- **Select in Detail**
  used to select any data from the database

Basic Syntax
SELECT col1, col2 FROM table_name;

To select All
SELECT * FROM table_name;

Where clause
To define some conditions

SELECT Col1, Col2 FROM table_name
WHERE conditions;

SELECT * FROM Student WHERE marks > 80;
SELECT * FROM student WHERE city = "mumbai";

Shorts

SELECT * FROM student WHERE marks > 80
AND city = "Amalner";

Where clause.
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).

* Operators.

AND: (to check for both conditions to be true)
SELECT * FROM student WHERE marks > 80 AND
city = "Amalner";

OR: (to check for one of the conditions to be true)
SELECT * FROM student WHERE marks > 90 OR
city = "Amalner";

Between: (select for a given range)
SELECT * FROM student WHERE marks BETWEEN 80 AND 90;

In: (matches any value in the list)
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;

SELECT col1, col2 FROM table-name
LIMIT number;
```

Order By clause

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

1]
```
SELECT * FROM student
ORDER BY city ASC;

SELECT col1, col2 FROM table-name
ORDER BY col-name(s) ASC;
```

2]
```
SELECT * FROM student
ORDER BY city DESC;

SELECT col1, col2 FROM table_name
ORDER BY col-name(s) DESC;
```

Aggregate Functions.

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

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

Get maximum marks.

```
SELECT max(marks)
FROM student;
```

Get Average marks.

```
SELECT avg(marks)
FROM student;
```

Group By Clause

Groups rows that have the same value 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 function.

Count number of students in each city.

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

Q] Write the Query to find avg marks in each city in acceding order.

⇒

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

Q] For the given table, find the total payment according to each payment method.

customer_id	customer	mode	city
101	laxman shirse	Net Banking	Amalgaon
102	jitu patil	credit card	Menergaon
103	Jayesh pwar	Debit card	Dapner
104	Vijay Devare	cash	Someshwar
105	Raghav More	cash	Delhi
106	Lina pawar	cheque	pune
107	Andrew Tate	cash	Malegaon
108	Vinay Kale	Net Banking	Mdn Jalod
109	Manoj Pawar	credit card	Pingalwade
110	Sandip Sone	Debit card.	Dapur

E2E02 1178
OF - SET SQL-SAFE-UPDATES=0;
ON - SET SQL-SAFE-UPDATES=1;

* Having clause.

similar to where i.e. applies some condition on rows used when we want to apply any condition after grouping.

count number of students in each city where max marks cross 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;
```

Table related queries.

Update (to update existing rows)

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

for grade

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

for marks change.

```
UPDATE student
SET marks = 82
WHERE roll no = 101;
```

Table related Queries.

Delete (to delete existing rows)

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

```
DELETE FROM student
WHERE marks < 33;
```

* ~~FK.~~
* Revisiting FK.

department (dept)

id	name
101	Science
102	History
103	English

↑ parent table

teacher.

id	name	dept_id
101	Davide	101
102	Mary	102
103	Rouie	102
104	John	103

↑ child table

Cascading for FK.

~~on~~ update ~~cascade~~ on delete Cascade.

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

~~on~~ ~~delete~~ ~~cascade~~ on update Cascade

When we create to a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

2:07:10
Practice
Stop
only theory
Written

```
CREATE TABLE Student (
 id INT PRIMARY KEY,
 courseID INT,
 FOREIGN KEY (courseID) REFERNCES Course(cid)
  ON DELETE CASCADE
  ON UPDATE CASCADE
 );
```

* Table related queries:
Alter (to change the seen schema)
↳ design (column, datatype, constraints)

~~Add~~ ADD column
```
ALTER TABLE table-name
ADD COLUMN column_name datatype constraint;
```

DROP column
```
ALTER TABLE table_name
DROP COLUMN column_name;
```

RENAME table
```
ALTER TABLE table_name
RENAME TO new_table-name;
```

CHANGE column (rename)
```
ALTER TABLE table-name
CHANGE COLUMN old-name new_name new_datatype
                                new-constraint;
```

MODIFY column (modify datatype/constraint)
```
ALTER TABLE table-name
MODIFY col-name new-datatype new-constraint;
```

Query:

ADD column
ALTER TABLE student
ADD COLMN age INT NOT NULL DEFAULT 19;

MODIFY column
ALTER TABLE student
MODIFY age VARCHAR(2);

CHANGE column (rename)
ALTER TABLE student
CHANGE age stu_age INT;

DROP column
ALTER TABLE student
DROP COLUMN stu_age;

RENAME table
ALTER TABLE student
RENAME TO stu;

- Truncate (to delete table's data)

TRUNCATE TABLE table_name;

~~UPDATE~~
UPDATE student
SET grade = "O"
WHERE grade = "A";

Q7] In the student's table:
a] change the name of column "name" to "full_name".
b] delete all the students who scored marks less than 80.
c] delete the column ~~for~~ for grades.

→

a] ALTER TABLE student
CHANGE name full_name VARCHAR(50);

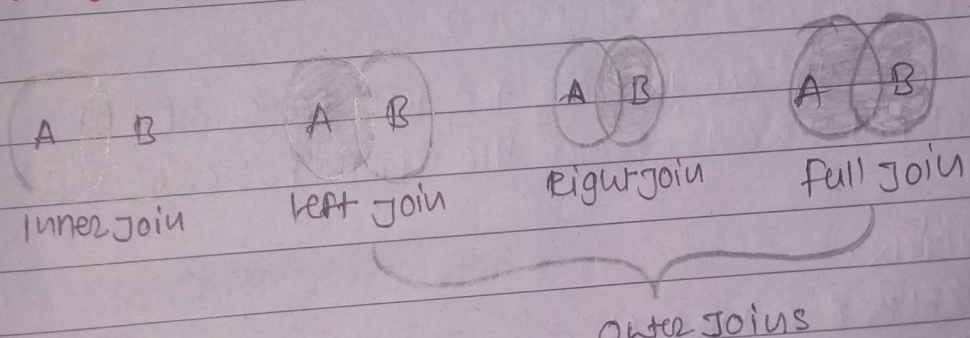
b] DELETE FROM student
WHERE marks < 80;

c] ALTER TABLE student
DROP COLUMN grade;

Joins in SQL

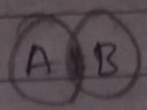
Join is used to ~~count~~ combine rows from two or more tables, based on a related column between them.

Types of Joins (venn Diagram)



Innee Join

Reterins records that have matching values in both rables.



Syntax

```
SELECT Column(s)
FROM tableA
INNER JOIN tableB
ON tableA.col-name = tableB.col_name;
```

alias
↳ alternate name.

Ex:

Student

student Id	name
101	adam
102	bob
103	casey

Couese

student_id	couse
102	English
105	Math
103	Science
107	Computer Science

RESULT

student_id	name	couse
102	bob	English
103	casey	Science

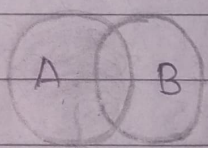
```
SELECT *
FROM student
INNER JOIN couse
ON student.student_id = couse.stud
```

* Left Join.

Reterens all records from the left table, and the matched Records from the right table.

Syntax

```
SELECT Column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col-name = table.B.col-names
```



Eg: student (left)

student_id	name
101	adam
102	bob
103	casey

course (Right)

student_id	course
102	english
105	math
103	Science
107	Computer science

Result

student_id	name	course
101	adam	null
102	bob	english
103	casey	science

```
SELECT *
FROM student as s
LEFT JOIN course as c
ON s.student_id = c.student_id;
```

* Right table.

Returns all records from the right table, and the matched records from the left table.

Syntax

```
SELECT column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

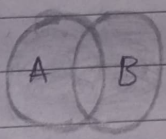
Result

student-id	course	name
102	english	bob
105	math	null
103	Science	casey
107	Computer science	null

```
SELECT *
FROM student as S
RIGHT JOIN course as C
ON S.student_id = C.student_id ;
```

* Full JOIN

Returns all records when there is a match in ~~matter~~ either left or right table.



Syntax in Mysql

```
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

Ex:

student

student_id	name
101	adam
102	bob
103	casey

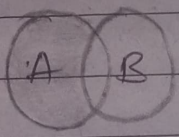
course

student_id	course
102	english
105	math
103	science
107	computer science

Result

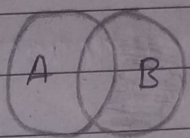
student_id	name	course
101	adam	null
102	bob	english
103	casey	science
105	null	math
107	null	computer science

Q7] Write SQL commands to display the eight exclusive join:



Left Exclusive Join

```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```



Right Exclusive Join

```
SELECT
FROM student as a
RIGHT JOIN course as b
ON a.id = b.id
WHERE a.id IS NULL;
```

* SELF JOIN

It is a regular join but the table is joined with itself

Syntax

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

Ex:

Employee

id	name	manager_id
101	adam	103
102	bob	104
103	casey	null
104	donald	103

```
SELECT a.name as manager_name, b.name
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```

→ * Union.

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

Gives UNIQUE records.

To use it

- every SELECT should have same no. of columns
- columns must have similar data types.
- columns in every SELECT should be in same order

Syntax

```
SELECT column(s) FROM tableA
UNION
SELECT column(s) FROM tableB
```

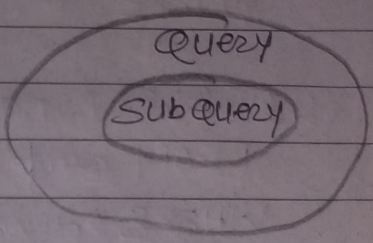
* SQL Sub Queries.

A ~~Subqueries~~ subquery or Inner query or a Nested query is a query within another SQL query

It involves 2 select statements.

Syntax

```
SELECT column(s)
FROM table-name
WHERE col-name operator
(subquery);
```



Ex:

Get names of all students who scored more than class average.

rollno	name	marks
101	anil	78
102	bhumika	93
103	chetan	85
104	dhruv	96
105	emanuel	92
106	Pazan	82.

step1: Find the avg of class

step2: Find the names of students with marks > avg.

1] SELECT AVG (marks)
FROM Student;

2] SELECT name
FROM Student
WHERE marks > avg mark;

(on dynamic process)

```
SELECT name, marks
FROM student
WHERE marks > (SELECT AVG (marks) FROM student);
```

ex:

Find the names of all student with even 2011 numbers

Step 1: Find the even 2011 numbers

Step 2: Find the names of students with even 2011 no.

```
1] SELECT 2011 no.
   FROM Student
   WHERE 2011 no %2=0;

2] SELECT name
   FROM Student
   WHERE 2011 no IN
   (102, 104, 106);
```

2011 no	Name	Marks
101	anil	78
102	bhumika	93
103	chetan	85
104	dhruv	96
105	emanuel	92
106	Farah	82

```
SELECT name, 2011 no
FROM Student
WHER 2011 no IN (
        SELECT 2011 no
        FROM Studen
        WHERE 2011 no %2=0);
```

Example with FROM

Find the max marks then the students of Delhi

Step 1: Find the students of Delhi

Step 2: Find their max marks during the sublist in step 1.

2011 no	name	marks	city
101	anil	78	pune
102	bhumika	93	mumbai
103	chetan	85	mumbai
104	dhruv	96	Delhi
105	emanuel	92	Delhi
106	Farah	82	Delhi

1] SELECT *
FROM student
WHERE city = "Delhi";

2] SELECT MAX(marks)
FROM (SELECT * FROM student WHERE city="Delhi")
AS temp;

~~Short~~

SELECT MAX(marks)
FROM student
WHERE city = "Delhi";

* Example with SELECT

SELECT (SELECT MAX(marks) FROM student), name
FROM student;

* MySQL views

A view is a virtual table based on the result-set of an SQL statement.

CREATE VIEW View1 AS
SELECT roll no, name FROM student;
~~SET~~
SELECT * FROM View1;

* A view always shows up-to-date data. The database engine recreates the view, every time a user queries it.