Constraints:

Constraints in a database are rules enforced on data columns to ensure the integrity, accuracy and reliability of the data stored in the database.

Types of constraints:

① primary key:

- ensure each row in a table has a unique identifier.
- No Null value allowed

Example:

```
CREATE TABLE students(
    StudentID INT PRIMARY KEY,
     Name     VARCHAR(100)
);
```

② foreign key:

- Ensures the referential integrity of data in one table to match values in another table.

Example:

```
CREATE TABLE Orders (
   OrderID INT PRIMARY KEY,
   CustomerID INT
   FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

③ **Unique:**

- Ensures all values in a column are unique.
- Allows one NULL value.

Ex:

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
        Email VARCHAR(100) UNIQUE
);
```

④ **Not NULL Constraint:**

- Ensures a column can't have a NULL value.

Example:

```
CREATE TABLE Products (
    productID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL
);
```

⑤ CHeck :

- Ensures the values in a column meet a specific condition.

Example:

```
CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    Balance DECIMAL (10,2),
    CHECK (Balance >= 0)
);
```

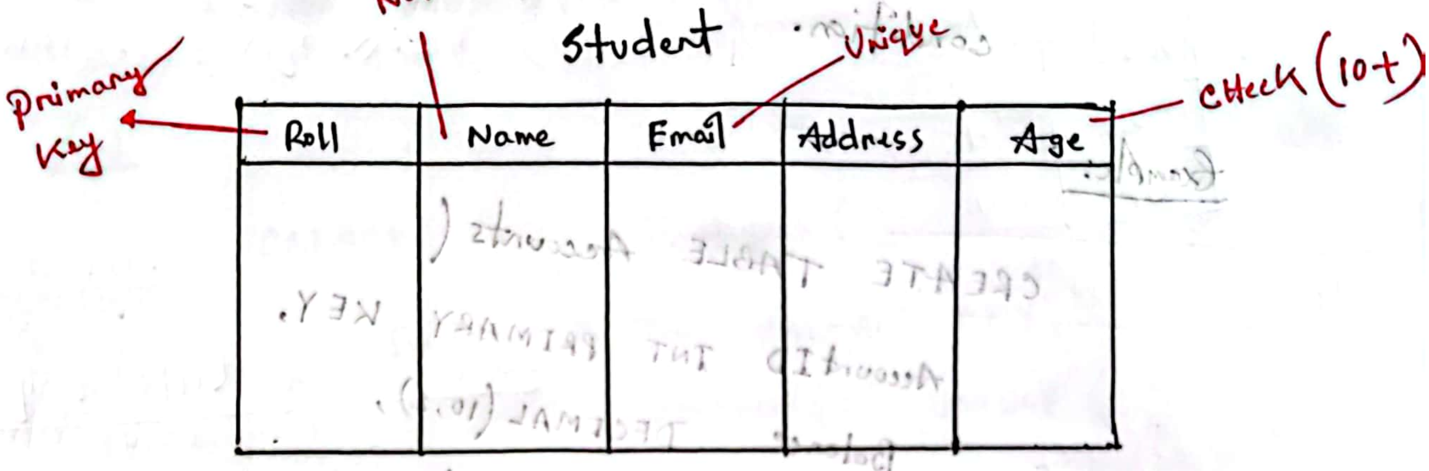
⑥ Default :

- provides a default value for a column when no value is specified.

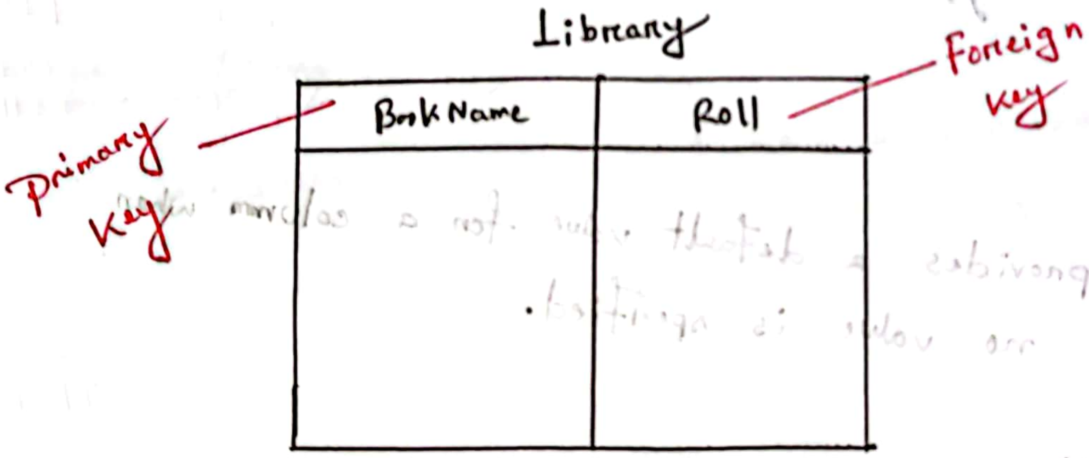
Example:

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE DEFAULT GETDATE()
);
```

#



Student
NOT NULL
Unique
Primary Key
Check (10+)
Roll
Name
Email
Address
Age



Library
Foreign key
Primary Key
BookName
Roll

SELECT Query :

Syntax:

```
SELECT column1, column2 . . . . .
  FROM table_name ;
```

- **selecting all columns:**

```
SELECT * FROM table_name ;
```

- **selecting specific columns:**

```
SELECT Roll, Name FROM students ;
```

- **Using WHERE clause:**

```
SELECT * FROM Students WHERE Age > 18 ;
```

- **Arithmatic Operators (+, -, *, ÷, %):**

```
SELECT CSE+ME, CSE-ME, CSE*ME, (CSE+ME)/2
FROM Marks
WHERE Roll = 104 ;
```

Marks

Roll	CSE	ME
101	10	20
102	20	40
103	30	60
104	7	10

• **Comparison Operators (=, <, >, ⩽, ⩾, !=):**

```
SELECT * FROM Student
WHERE  Age >=15;
```

Student

Roll	Age
101	10
102	20
103	30
104	40

• **Logical AND, OR operators:**

```
SELECT  Name
FROM    Employee
WHERE   Salary >9000 AND Age >24
```

Employee

ID	Name	Salary	Age	Designation
1	A	10000	22	Snr. SWE
2	B	15000	23	Jnr. SWE
3	C	20000	24	Team lead
4	D	12000	25	Pr. Manager
5	E	9000	24	Snr. SWE

• **DISTINCT:**

```
SELECT DISTINCT Designation
FROM Employee;
```

$\longrightarrow$ Selects all the distinct designation.

- **Note:**

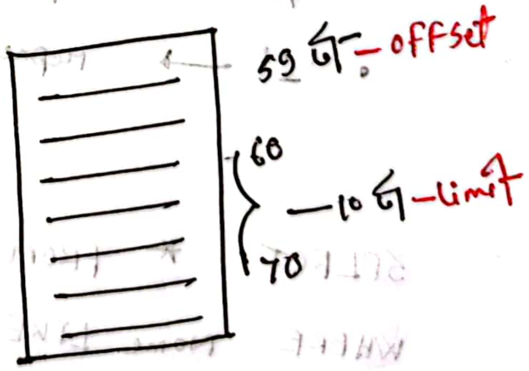
```
SELECT
FROM
WHERE
ORDER BY
LIMIT
OFFSET
```
$\downarrow$

- ~~Order~~ ORDER BY:

```
SELECT   Name
FROM     Employee
ORDER BY   Salary   ASC;  → Ascending order
                    DESC; → Descending order
```

- Limit offset:

```
SELECT
FROM
WHERE
ORDER BY
LIMIT    10   |  LIMIT  59,10
OFFSET   59   |
```



- **IN:**

```
SELECT *
FROM students
WHERE Roll IN (1,3,5)
```
— select students with Roll number 1,3,5

- **NOT IN:**

```
SELECT *
FROM students
WHERE Roll NOT IN (1,2,3)
```
— select students whose roll numbers aren't 1, 3 or 5.

- **LIKE:**

  - To search for a specified pattern in a column.
  - '%' → represents zero or more characters
  - '_' → represents a single character.

```
SELECT * FROM students
WHERE Name LIKE 'A%';
```
→ select students whose names start with 'A'

```
SELECT * FROM students
WHERE Email LIKE '%gmail%';
```
→ select students whose email contains gmail

- **AS :**

used to rename a coloumn or table with an alias.

```
SELECT Name AS StudentName
FROM Students;
```

→ selects the name column but rename it to StudentName in the result set.