SQL --> Structured Query Language
Not a case sensitive language but data(value) within SQL is case sensitive
char(20) ==== CHAR(20)
stud.name ["suresh" != "SURESH"]

Data - is a raw fact which is used to describe the attribute of an entity.
raw-fact --> unprocessed/unchanged data
attribute --> properties | behaviour
entity --> anything with physical existance | object

Database -- place in memory where data are stored in systematic/organised manner

EG.. PEN
attributes|properties -- [color, size, brand, price type]
data|raw-fact|values -- [blue, Medium, Cello, 10, Dot]

CRUD OPERATION in db
Create | Insert
Read | Retrive
Update | Modify
Delete | Drop

DBMS -- s/w used to maintain and manage our database.
The two kwy features offered by DBMS are --
Security and Authorization

we require query language in order to communicate with DBMS
data going to be stored in file format.

DBMS -- not much appreciated
RDBMS(introduced by EDGAR FRANK CODD) -- mostly used
SQL(introduced by RAYMOND BOYACE in 1970) --

Some well known types of DBMS --
1. Hierarchical
2. Object oriented
3. Networking
4. RDBMS

RDBMS -- Relational Database Management System
Relation -it is a type of DBMS software which is used to maintain and manage our database in form of relation
SQL -- to communicate with relational database MS
it provides security and Authorisation.
Data stored in form of Relation(tables)

model was introduced -- Relation called Relational model

First R Model then SQL was introduced

Difference DBMS vs RDBMS
1. Abbreviation DBMS and RDBMS
2. Query Language QL vs SQL
3. file format vs Table format
4. ease of crud OPERATION

Logical organisation of data in form of rows and columns
1. Columns|Attributes|Fields|Properties
2. Rows|Records|Tuples

E F CODD rules for RDBMS
Mainly 4 rules --

1. Single valued Atomic data stored in each cell
2. in RDBMS every data is stored in form of Table including metadata.
   metadata(further info about data) -- auto generated by system.
   metadata(stored in metaTable)
3. In RDBMS we can store data in multiple table if necessary/required we can establish connection between them using
   Key Attributes.
   Student ---------> Teacher (Key Attribute)
4. entered data need to be validated in two ways. (constraints)
   a. by assigning attribute data type (mendatory)
   b. by assigning constraints (optional)

types of DataType
1. Char (Character -- 'A-Z','a-z','0-9','@#$_')
2. Numeric (Number)
3. Varchar (Variable Character)
4. Date
5. Large Object (CLOB(char large obj), BLOB(Binary Large Obj))

1. ### CHAR
fixed length memory allocation
before using char we need to mention size of it
syntax --- char(20) upto 2000 Character
must be enclosed in single Character ['Vinod']
follows fixed length memory allocation
Adhar PNR PAN BankAccountNo ID

-- faster execution than varchar

2. ### VARCHAR
CHAR + 'AlphaNumeric'
stand for variable character (same like char without fixed size)
syntax -- varchar(20) upto 2000
variable length memory allocation
name(10) 'chetan' rest free memory space
(checks length used to free up wastes time hence slower than char)

3. ### VARCHAR2 -- updated version of varchar
size upto 4000 character

upto 2000 ----> char | varchar
> 2000   ----> VARCHAR2

varchar(5) --> means varchar2(5) compiler considers by self

4. ### Number
syntax -- NUMBER(precision, [scale])
precision --> integer value (range upto 38 digit) mendatory
scale   --> decimal | floating value (range -84 to 127) optional

case1### NUMBER(6) // 9 digit number
max : 999999 and min : -999999

case2### NUMBER(4, 2) _ _ _ _
max: 99.99 and min : -99.99

eg: NUMBER(7, 5) max : 99.99999 and min : -99.99999
    NUMBER(39, 4) failed as max limit exceeded

case3### precision == scale 0.9999999--
precision < scale
Number(2, 5) memory block allocated = max(precision, scale) = max(2, 5) = 5
_ _ _ _ _
0.00099 max and -0.00099 min

case4### scale < 0 or negative
NUMBER(4, -1) memory block allocated = add(prec, abs(scale)) == add(4, 1) =
5 _ _ _ _ _
0 9 9 9 9 --> roundofwith(last 9) = 10000
NUMBER(5, -2) _ _ _ _ _ _ _

0 0 9 9 9 (9 9) --> round(99) ---> 0100000

5. ### DATE
syntax: -- DATE
2 formats available --
1. 'DD-MM-YYYY' --> 1235 === 1235
2. 'DD-MM-YY'   --> 24 === 2024

6. ### LARGE OBJECT

1. CLOB -- Character Large Object [store large amount of character > 4000 character]
max size : 4GB [for more than 4GB -- goto MYSQL (size limit 8GB)]
syntax: CLOB

2. BLOB -- Binary Large Object [store large binary form of multimedia data eg. image video audio etc.]
max size: 4GB [for more than 4GB -- goto MYSQL (size limit 8GB)]
syntax: BLOB

Note :
in order to create a table 2 things required
1. attribute data type (mendatory)
2. constraints (optional)

CONSTRAINTS --
set of rules assigned to column | attributes in order to validate data.
1. UNIQUE constraint to avoid repeated or duplicate value in defined attribute
2. NOT NULL constraint to avoid null or unfilled data field
3. CHECK constraint for extra validation of condition to be satisfied :: CHECK(condition)
4. PRIMARY Key
5. FORIEGN Key

phone_no number(10) unique not null constraint valid_phone check(length(phone_no) = 10 && phone_no > 0)

PRIMARY KEY ---
attribute that can define all other attributes in unique manner

Characterstics --
1. wont allow repeated or duplicate value -- UNIQUE
2. wont allow null value -- NOT NULL
PRIMARY KEY == [UNIQUE + NOT NULL]

FOREIGN KEY ---
also known as referential integrity constraints
a column to become a foreign key, it must be primary key of its own table.

Characterstics --
1. allow repeated or duplicate value
2. allow null value

In order to establish connection bw two table, 2 steps are required
1. we have to add a column
2. assign foreign key to that column

CUSTOMER [cid, cname, phn, add] PROD [pid, pname, price, loc]
CUSTOMER [cid, cname, phn, add, pid] PROD [pid, pname, price, loc]

pid --> foreign key in CUST(may be null and duplicate), primary key in PROD.

SQL statements : --
the 5 SQL statements are as follows.
1. DDL [Data Definition Language] (define_structure | create)
2. DML [Data Manipulation Language] (insert | delete | update)
3. TCL [Transaction Control Language]
   (Defining and using Transactions(DML operation))
4. DCL [Data Contrl Language]
   (control over flow of data bw users)
5. DQL [Data Query Language]
   (question.answer | data_show | select querry)

TCL stands for Transaction control language
   Transaction -- DML operation performed on database.
(insert|update|delete)

Data Control Language --
   controls user who will operate data and who will use the data.
   1. grant -- give access permission

   2. revoke -- withdraw access permission

Data Query Language --
   read or retrieve our data

DQL -- (data query language)

1. SELECT   --> only rows
2. PROJECTION --> retriving data by selecting only column
3. SELECTIONS --> used to retrieve data by selecting both rows and column
4. JOINS   --> retrive data from multiple tables simultaneously

PROJECTION ---
Q. Write querry to display differnt salary wihout using distinct clause
    Select unique sal ;

SELECTION ---

```
SELECT */[DISTINCT] COLNAME /EXPRESSION[ALIAS]
FROM TABLE
WHERE <Filter_Condition> ;
```

where clause
where clause execute row by row
we can pass multiple condition with help of logical operator...

Q: write a query to display employee earning more than 10k
A: select * from employee where salary > 10000 ;

Order of execution --
1. from
2. Where
3. select

SQL operators --
7 types of operators --
1. Arithmetic operators
2. Relational operators (<, > >=)
3. Comparison operators (=, !=)
4. Concatenation op (merge 2 strings) (||)
5. Logical Operator (and or not nor)

6. Special operator
        IN   NOT IN
        BETWEEN   NOT BETWEEN
        IS   IS NOT
        LIKE   NOT LIKE
7. SubQuery Operator --
        ALL
        ANY
        EXISTS
        NOT EXISTS

CONCATENATION OPEARTOR ---
|| -- concatenation operator
```
select "Hi " || ENAME from EMP ;
select "Hi " || ENAME || " Thank You! Ma Chuda!.." ;
```

Hi Lawda
Hi Lassan

LOGICAL OPERATOR ---
1. AND operator
2. OR operator
3. NOT operator (unary operator)

```
select ename, deptno from employee where job = 'Manager' AND deptno = 10 ;

select ename, deptno, sal*12 as AnnualSalary
from employee
where deptno = 10 AND sal*12 > 14000 AND job = 'SALESMAN' ;
```

IN OPERATOR --
multi valued operator used to accept multiple value.
syntax -- colName IN (multiple values[n1, n2, n3, ...])

BETWEEN operator --
it is used when we have ranges of value.
between lower value and upper values

Note - it includes the ranges.
```
select salary from employee where salary between 1000 and 2000 ;
```

```
mysql> select *from cricket where cricket_id between 2 and 4 ;


+------------+---------+
| cricket_id | name    |
+------------+---------+
|          2 | Michael |
|          3 | Johnson |
|          4 | flaming |
+------------+---------+


select name, deptno from employee where deptno = 10 and hiredate between
'01-Jan-2019' and '31-Dec-2019' ;


IS - used to compare only null value
select ename from emp where comm is null ;

IS NOT - not null value


LIKE -- pattern matching..
% -- n no of char(0 char also)
_ -- single char


select *from employee where name like '_m%' ;


NOT LIKE


select ename from emp where sal > (
        select sal from employee where ename in "SCOTT" and
        deptno in (
                select deptno from dept where ename in "LAWDA"
        )
) ;


TYPES OF SUBQUERRY..
There are 2 types subquerry --
1. Single row subquerry
        if the subquerry return exactly one row or value, is called as SRSQ
        it returns only one value, then we can use normal/special
operators. (=:"normal", "IN": special)
```

2. Multi-row subquerry.
        if the subquerry return more than one record/values, called as
        if it returns more than one value we cannot use normal operator.

WAQ to display dname of "ALEN" ?
```
select dname from emp where deptno = (
        select deptno from emp where ename = "ALEN"
) ;
```

subquerry opertaor ---
1. ALL : special operator used along with relational operator to compare the value.
   all operators returns true if all values at RHS has satisfied the condition.

   > ALL, < ALL, >= ALL, <=ALL.

```
select ename, sal from emp where sal > ALL (
        select sal from emp where deptno in 10
) ;
```

sal > all (800 and 1300)

WAQ to display name of the emp if emp earn less than emp working as SALESMAN.
```
select ename from emp where sal < ALL (
        select sal from emp where job = "SALESMAN"
) ;
```

2. ANY : it is special operator used along relational operator.
         returns true if one of the values at the RHS has satisfied the condition.

```
select ename, sal from emp where sal > ANY (
        select sal from emp where deptno in 10
) ;
```
sal > ANY (800 and 1300)

NESTED QUERRY ---
we can nest upto 255 sub querries.

display second maximum salary.
```
select max(sal) from emp where sal < (select max(sal) from emp) ;
```

```
## Find the third max salary..
select max(salary) as ThirdMaxSalary from employee
        where salary < (select max(salary) from employee
                where salary < (select max(salary) from employee)) ;


## Third Minimum salary ..
select min(salary) as ThirdMinSalary from employee
        where salary > (select min(salary) from employee
                where salary > (select min(salary) from employee)) ;
```

```
### EMPLOYEE MANAGER RELATIONSHIP ---

WAQD name of smith manager ??
select ename from employee where empno = (select mgrno frm employee where
ename = "SMITH") ;
```

```
#### JOINS
The process of retrieval of data from multiple table simultaneously is
known as join.
data of both tables cannot be displayed at a time, hence needed join.

Types of joins --
1. Cartesian join (Cross Join)
2. Inner Join     (equi join)
3. Outer Join
   -- Left Outer Join
   -- Right Outter Join
   -- Full Outer Join

4. self join
5. natural join


### 1. CARTESIAN JOIN
In cartesian join, a record from table 1 will be merged with all the
records of table 2

Number of columns in result table = n(table1.columns) + n(table2.columns)
```

Number of rows in result table = n(table1.rows) * n(table2.rows)
Syntax --
1. ANSI -- American National
select column_names from

2. INNER JOIN
it is used to obtain only matching records which has pair.
ANSI syntax - select column_name from table_name1 inner join table_name2 on {join condition} ;
Oracle syntax - select column_name from table_name1, table_name2 on {join condition} ;

3. OUTER JOIN
it is used to obtain unmatched records.
3 types --
1. left-outer join
2. right-outer join
3. full-outer join

Left Outer Join --
used to obtain unmatched records of left table along with matching records

Right Outer Join ---
it is used to obtain unmatched records of right table along with matching records

Full Outer Join --
it is used to match records from both left and right
waqd names of all employee dept even though employees dont work in any dept and dept with no employee.

self join --
joining a table by itself
why? | when?
whenever the data to select is in same table but present in different records.

syntax --
select column_name from table_name t1 join table_name t2 on <join_condition> ;

Natural Join ---
it behaves as inner join. if there is a relation bw the given two tables. else it behaves as cross join.
ANSI syntax -- select column_name from table1 natural join table2 ;

Correlated Sub-Queries --
a query written inside another query such that outer querry and inner querry are dependent on each other
is known as correlated sub-querry.

first outer-querry execute partialy and gives result to inner as input, further inner gives op to outer
querry

Working principle --
let us consider 2 queries inner and outer queries respectively
1. outer query execute first but partially.
2. partially executed output is given as an input to the inner querry
3. the inner query execute completely and generates an output
4. the output of inner querry is fed as input to the outer query and outer query produce the result.
5. Therefore we can state that the outer query and the inner query both are interdependent.

Note --
1. in correlated subquery a join condition is must, and must be written only in the inner query.
2. correlated subquery works with the principle of both subquery and joins

question: WAQD dnames in which there are employees working

```
select dname
from dept
where deptno IN (
   select deptno
   from emp
   where emp.deptno = dept.deptno
) ;
```

question: WAQD dname in which there are no emplyee working

```
select dname
from dept
where deptno not in (
   select deptno
   from emp
   where emp.deptno = dept.deptno
)  ;
```

Difference bw correlated subquery and subquery --

S1. inner query execute first
C1. outer query execute first

s2. outer query is dependent on inner query
c2. both are interdependent

S3. join condition not mendatory
C3. join condition is mendatory and must be written in subquerry

S4. outer query exceede once
C4. outer query executes twice

EXISTS and NOT EXISTS operator.

EXISTS : --
1. operator is unary operator which can accept one operand towards right hand side and that operand has to be
   correlated subquerry.
2. exist operator returns true if the subquery returns any value other then null|empty.

```
select dname
from dept
where exists (
   select deptno
   from emp
   where emp.deptno = dept.deptno
) ;
```

NOT EXISTS : --
1. operator is unary operator which can accept one operand towards right hand side and that operand has to be
   correlated subquerry.
2. not exists operator returns true if subquery returns null

```
select dname
from dept
where not exists (
   select deptno
   from emp
   where emp.deptno = dept.deptno
) ;
```

To find max/min salary --

```
select sal
from emp e1
where (
   select count(distinct sal)
   from emp e2
   where e1.sal < e2.sal
) = N - 1 ;
```

3rd, 4th and 5th

```
select sal
from emp e1
where (
   select count(distinct sal)
   from emp e2
   where e1.sal < e2.sal
) IN(3-1, 4-1, 5-1)  ;
```

```
select salary
from employee as e1
where(
   select count(distinct salary)
   from employee as e2
   where e1.salary > e2.salary
```

```
) IN (1, 2, 3) ;

OUTPUT --
+--------+
| salary |
+--------+
|  54000 |
|  59400 |
|  84000 |
+--------+
3 rows in set (0.01 sec)

mysql> select salary from employee order by salary ;
+--------+
| salary |
+--------+
|  41000 |
|  54000 |
|  59400 |
|  84000 |
|  88000 |
+--------+
```

DDL -- Data Definition Language
is used to construct an object/table in the database and deals with the structure of the object.
it has 5 statements --
1. CREATE
2. RENAME
3. ALTER
4. TRUNCATE
5. DROP

1. ### CREATE
it is used to build/construct an object.

How to create a table..
1. name of the table (tables cannot have same names)
2. no_of_columns
3. name_of_columns
4. assign dataTypes to columns
5. assign constraints(not mendatory)

```
CREATE TABLE TABLE_NAME (
  COL_NAME1 DATA_TYPE1 CONSTRAINT1,
  COL_NAME2 DATA_TYPE2 CONSTRAINT2,
  .... so on

) ;

create table student(
  SID varchar(20),
  SNAME varchar(30),
  AGE numeric,
  PhoneNo varchar(11)
) ;

desc student ;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| SID     | varchar(20)   | YES  |     | NULL    |       |
| SNAME   | varchar(30)   | YES  |     | NULL    |       |
| AGE     | decimal(10,0) | YES  |     | NULL    |       |
| PhoneNo | varchar(11)   | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+


2. ### RENAME
it is used to change the name of the object.
syntax --
RENAME TABLE TABLE_NAME TO NEW_TABLE_NAME ;

rename table student to stud ;

desc stud ;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| SID     | varchar(20)   | YES  |     | NULL    |       |
| SNAME   | varchar(30)   | YES  |     | NULL    |       |
| AGE     | decimal(10,0) | YES  |     | NULL    |       |
| PhoneNo | varchar(11)   | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+


3. ### ALTER
it is used to modify the structure of existing table.
-- TO ADD A COLUMN
```

```
ALTER TABLE TABLE_NAME ADD COLUMN DATA_TYPE CONSTRAINTS ;
```

```
alter table student add column class numeric ;
desc student ;
+---------+---------------+------+-----+---------+-------+
| Field   | Type          | Null | Key | Default | Extra |
+---------+---------------+------+-----+---------+-------+
| SID     | varchar(20)   | YES  |     | NULL    |       |
| SNAME   | varchar(30)   | YES  |     | NULL    |       |
| AGE     | decimal(10,0) | YES  |     | NULL    |       |
| PhoneNo | varchar(11)   | YES  |     | NULL    |       |
| class   | decimal(10,0) | YES  |     | NULL    |       |
+---------+---------------+------+-----+---------+-------+
```

```
-- TO DROP A COLUMN
```