\*\*\*\*\*\*\*\*\*\*DDL\*\*\*\*\*\*\*\*\*\*\*

1.create database ustglobal; //creating a database

2.show databases;

3.use ustglobal; // using particular database

4.show tables;

5.create table student(id int, name varchar(120), emailId varchar(150)); //creating a table

6.show tables; //we can see the tables present in the database

7.desc student; //describing the table

8.drop table student; //delete the records as well as structure of the table

9.drop database ustglobal; //drop the database

10.alter table student add column mobileNo int; //we can add one column in existing table

11.alter table student add address varchar(120); //we can add columns without using 'column'

12.alter table student drop address; //deleting one column

13.alter table student change name stdname varchar(120); //changing the column name

14.alter table student add address varchar(120); //adding one column to existing table

15. rename table student to student\_info; //change the table name

\*\*\*\*\*\*\*\*\*\*\*DML\*\*\*\*\*\*\*\*\*

16.insert into student\_info(id,stdname,emailId,mobileNo)values(100,'suma','suma.v@gmail.com',9876); //to insert a row

17.insert into student\_info values(101,'vani','vani.r@gmail.com',987654);

18.insert into student\_info values(102,'navya','navya.v@gmail.com',954236),(104,'kavitha','kavitha.v@gmail.com',123455); //we can insert multiple rows at a time

19.select \* from student\_info; //to display the records in the table

20. update student\_info set mobile=987654; //to update the existing data with new data

21. delete from student\_info; //delete all the records in the table

22. truncate table student\_info; //to delete all the records in the table

23. select id,stdname from student\_info; //if we want to retrieve particular column

Q. WAQ to display id, name from the student\_info whose name is vani

A. select id, stdname from student\_info where stdname='vani';

//updating the record based on given condition

24. update student\_info set stdname='meghana' where id=104;

//delete the record where id=104

25. delete from student\_info where id=104;

**Q. write a query to give a temporary name to a blood group as bg**

ANS. select bloodgroup AS bg from employee;

**Q. write a query to display employee name, salary, designation and his annual salary.**

**ANS. select name, salary,designation,salary\*12 as annualsalary from employee; (OR)**

select name, salary,designation,salary\*12 annualsalary from employee;

**Q. Write a query to give the to give the alias name for all the columns present in your employee table.**

**ANS.**

**Q. Write a query to display unique deptId from employee table**

**ANS.** select distinct deptId from employee;

**Q. Write a query to display unique values for deptId and designation.**

ANS. select distinct deptId,designation from employee;

Q. **Write a query to display unique values for deptId and designation, age.**

**ANS.** select distinct deptId,designation,age from employee;

**OPERATORS**

**Q. Write a query to display name salary and salary increment by 5000.**

ANS. select name, salary, salary+5000 as inc\_sal from employee;

**Q. . Write a query to display name salary and salary decrement by 5000.**

ANS. select name, salary, salary-5000 as dec\_sal from employee;

**Q. . Write a query to display name salary and annual salary.**

ANS. select name, salary, salary\*12 as annual\_sal from employee;

**Q. Write a query to display the salary for one day.**

ANS. select name, salary, salary/30 as day\_sal from employee;

**Q.Write a query to display all the records excludes deptId 10.**

ANS. select \* from employee where deptId!=10;

(OR)

select \* from employee where deptId!<>10;

**Q. Write a query to display name, designation, mobile num, age whose salary greaterthan 5000**.

ANS. select name, designation, mobileno, age from employee where salary>5000;

**Q. Write a query to display name, designation, mobile num, age whose salary lessthan 5000.**

**ANS.** select name, designation, mobileno, age from employee where salary<5000**;**

**Q. Write a query to display the salary lessthan or equal to 12000.**

ANS. select name, designation, mobileno, age, salary from employee where salary<=12000;

**Q. Write a query to display the salary greaterthan or equal to 12000.**

select name, bndesignation, mobileno, age, salary from employee where salary>=12000;

**Q. Write a query to display all the records who is having depiId 20 only if salary is greaterthan 10000.**

ANS. select \* from employee where deptId=20&&salary>10000;

**Q. Write a query to display all the reords of the employee table who is working as salesman and date of birth is 1994.**

ANS. select \* from employee where designation='salesman'&&dateofbirth='1994';

Q. Write a query to display all the records from employee table whose salary is 10000 as well as designation is analyst.

ANS. select \* from employee where salary=10000&& designation='analyst';

Q. Write a query to display all the records who is working in dept number 20 or dept number 60.

ANS. select \* from employee where deptId=20 or deptId=60;

Q. write a query to display id, name, email id who is working as a tester or age is greaterthan 20.

ANS. select id,name, emailId from employee where designation='tester' or age>20;

Q. write a query to display who is working as a hr in deptId 30 or whose designation is clerk and salary >20000.

ANS. select \* from employee where designation='HR' and deptId = 30 or designation ='clerk' && salary>20000; //first AND will execute then OR will execute

Q. write a query to display all records whose salary is 10000 and designation is developer

ANS. select \* from employee where salary=10000 and designation='developer';

Q. write a query to display all the employees except who is working as a developer.

ANS. select \* from employee where not designation ='developer';

Q. write a query to display all the employees excluding deptId 30 or 20

ANS. select \* from employee where not (deptId=20 or deptId=30);

//order NOT=> AND => OR

Q write a query to display all the employees execluding salesman who all are working in dept no 40.

ANS. select \* from employee where not(designation='salesman' and deptId=40);

Q. write a query to display all the employee details who is working as a salesman for dept number 20 or 30

ANS. select \* from employee where designation='salesman'and(deptId=20 or deptId=30);

Q. write a query to display all the employees who is working in dept no 20 only their salary is greaterthan 10000 and lessthan 25000 including 10000 and 25000.

ANS. select \* from employee where deptId=20 and (salary>=10000 and salary<=25000);

Q. Write a query to display all the records who is working as a developer and date of birth is 1994.

ANS. select \* from employee where designation ='developer' and(dateofbirth>='1994-01-01' or dateofbirth<='1994-12-31');

**Special Operators**

**IN**

Q. write a query to display employee who is working in dept 10 or 20 or 30.

ANS. select \* from employee where deptId in(10,20,30);

Q. Write a query to display employee whose job designation is Hr , analyst, tester.

ANS. select \* from employee where designation in('tester','HR','analyst');

Q write a query to display all the employee details who are working as a clerk in dept 20,30,40,60,70

ANS. select \* from employee where designation='clerk' and deptId in(20,30,40,60,70);

**NOT IN**

Not in operator is a multivalued operator

Syntax :

Select \* from table\_name where column\_name not in(val1,val2………);

Q. Write a query to display all the employee details who are working as a clerk and does not belong to dept 20,30,40,60,70

ANS. select \* from employee where designation='clerk' and deptId not in(20,30,40,60,70);

Q. . Write a query to display all the employee details who are working in dept 20,40,60 excluding the salesman or analyst

ANS. select \* from employee where deptId in(20,40,60) and designation not in('salesman','analyst');

**BETWEEN**

Between is used for searching based on range of values.

Syntax:

Select \* from tab\_name where col\_name Between lower-limit AND upper-limit

Q. Write a query to display all the details of employee whose salary in range of 20000 to 50000.

ANS. select \* from employee where salary between 20000 and 50000;

Q. Write a query to display all the details of employee who hired 2017

ANS. select \* from employee where dateofjoin between '2017-01-01' and '2017-12-31';

**NOT BETWEEN**

**Q.** Write a query to display all the details of employee excluding who is hired in 2019.

ANS. select \* from employee where dateofjoin not between '2019-01-01' and '2019-12-31';

**Q**. Write a query to display employee name, salary and deptId of all the employees who is belong to deptId 10 or 20 excluding employee whose salary is greater than or equal to 15000 and salary is less than or equal to 50000.

ANS. select name, deptId, salary from employee where deptId in(10,20) and salary not between 15000 and 50000;

**IS Operator**

ISoperator is used to check the null value. If record is null then it will return true, if record is not null then it will return false.

Q. write a query to display annual salary his emailId is null.

ANS. select salary\*12 from employee where emailId is null;

**LIKE**

Like operator is used to perform the pattern matching.

Syntax : Select \* from table\_name where col\_name like ‘\_\_’ .

Characteristics are two types : 1.Ordinary Character Type

2.Special Character Type(wild card character).

* %(percentage) – matches 0 or ‘n’ characters
* \_ (underscore) – matches exactly one character

Syntax : select \* from table\_name where col\_name like ’%/\_’ ;

Q. Write a query to display all the details of employees whose name starts with ‘s’.

ANS. select \* from employee where name like 's%';

Q. write a query to display all the details of an employee whose name contains only five characters.

ANS. select \* from employee where name like '\_\_\_\_\_';

Q. Write a query to display name, salary date of birth, date of joining whose name starts with s and end with s.

ANS. select name, salary, dateofbirth, dateofjoin from employee where name like 's%s';

Q. write a query to display all the records of employee whose second character is ‘p’.

ANS. select \* from employee where name like '\_p%';

Q. write a query to display all the employees where designation have atleast two ‘e’

ANS. select \* from employee where designation like '%e%e%';

Q. write a query to select all the strings of length the name starts with ‘s’ and second last character is ‘I’.

ANS. select \* from employee where name like 's\_\_l\_';

Q. Write a query to display all the details of employee if the name of employee starts with ‘r’ or starts with ‘z’ character .

ANS. select \* from employee where name like 'r%' or name like 'z%';

Q. write a query to display all the employees whose names in the third character should be ‘c’ and sixth character should be ‘I’.

ANS. select \* from employee where name like '\_\_c\_\_i%';

**NOT LIKE**

Not Like is used to select all the strings which should not match with the given string

Syntax: select \* from table\_name where col\_name not like ‘%/\_’;

Q. write a query to display all the employees excluding the details of employee whose name starts with ‘a’.

ANS. select \* from employee where name not like 'a%';

**FUNCTIONS**

Function is a block of code which is used to perform specific task.

Function attributes : It has input arguments function name and return type

Functions are two types: 1.Scalar functions (single row functions)

2.aggregate functions(multi row functions).

**AGGREGATE FUNCTIONS**

**-------🡪**

**-----🡪 -----🡪**

Aggregate functions takes multiple input values and returns only single output

Types of Aggregate functions :

* AVG()- Returns average value.
* COUNT()-It counts the number of rows.
* MAX()-Returns maximum value of the selected column.
* MIN()-returns minimum value of the selected column.
* SUM()-Returns sum of all the values of the selected column.

Q. Write a query to display the average salary of employee.

ANS. select avg(salary) from employee;

Q. Write a query to display min salary, maximum salary, sum of all the employees salaries.

ANS. select max(salary) from employee;

select min(salary) from employee;

select sum(salary) from employee;

select count(\*) from employee; //it will return the number of rows

**SCALAR FUNCTION/SINGLE ROW FUNCTION**

-----------------------🡪 -------🡪

---------------------🡪 ------🡪

----------------------🡪 ------🡪

Types of scalar functions:

UPPER()- it converts the value of a field to uppercase.

LOWER()-it converts the value of a field to lowercase.

LENGTH()-it returns the length of the value in a given field.

CONCAT()-it is used to concatenate two strings.

REVERSE()-

**Assignment-1**

**Q.** WAQ to display all the details of software developer in dept 30.

ANS. select \* from employee where deptId=30 and designation='developer';

Q. WAQ to display the list for all the software developer in dept no 40 and having salary greater than 5000.

ANS. select \* from employee where designation='developer' and deptId=40 and salary>5000;

Q. WAQ to display to list all the employees except those who are working in dept 10 & 20.

ANS. select \* from employee where deptId not in(10,20);

Q. WAQ to display list the employees who are not working as developer and clerks in dept 10,20 with the salary in the range of 1000 and 3000

ANS. select \* from employee where designation not in('developer','clerk') and deptId in(10,20) and salary between 1000 and 3000;

Q. WAQ to display the list of employees whose salary is not in the range of 10000 and 20000 in dept 10,20,30 except all salesman

ANS. select \* from employee where salary not between 10000 and 20000 and deptId in(10,20,30) and designation!='salesman';

Q. WAQ to arrange all the employees by their salary in descending order.

ANS. select \* from employee order by salary;

Q. WAQ to display the maximum salary, minimum salary, total salary from employee.

ANS. select max(salary),min(salary),sum(salary) from employee;

Q. WAQ to display the list of the number of software developer in dept 20.

ANS. select \* from employee where designation='developer' and deptId=20;

Q. WAQ to display the list of the highest and lowest salary earned by salesman.

ANS. select max(salary) as heighest\_sal,min(salary) as lowest\_sal from employee where designation='salesman';

Q. WAQ to display the total salary of all departments.

ANS. select sum(salary) from employee group by deptId;

**Assignment-2**

Q. WAQ to display all the employees working in dept 20.

ANS. select \* from employee where deptId=20;

Q. WAQ to display the employees earning more than RS.25000.

ANS. select \* from employee where salary>25000;

Q. WAQ to display all the employees whose job type is developer.

ANS. select \* from employee where designation='developer';

Q. WAQ to display all the employees working in dept 10,20,40.

ANS. select \* from employee where deptId in(10,20,40);

Q. WAQ to display all the employees whose name is having letter ‘l’ as second character.

ANS. select \* from employee where name like '\_l%';

Q. WAQ to display all the employees whose first name is having atleast 2 ‘a’ character.

ANS. select \* from employee where name like '%a%a%';

Q. WAQ to display all the employees whose name is having letter ‘e’ as the second last character.

ANS. select \* from employee where name like '%a\_';

Q. WAQ to display all the employees who are having exactly five characters in their name.

ANS. select \* from employee where name like '\_\_\_\_\_';

Q. WAQ to display all the employees whose salary is between 20000 and 30000.

ANS. select \* from employee where salary between 20000 and 30000;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Q. Write a query to display number of employees hired in after 2016.

ANS. select \* from employee where dateofjoin>'2016-12-31';

Q. WAQ to display average salary and the highest salary of dept 20,30,40 and 60.

ANS. select avg(salary),max(salary) from employee where deptId in(20,30,40,60);

Q. WAQ to display date of first and last hired employee.

ANS. select min(dateofjoin) as first\_hired, max(dateofjoin) as last\_hired from employee;

**Scalar functions are divided into two types**

1.Case manipulation functions

* Lower()
* Upper()
* Initcap() //initcap() is not supported in mysql

2.Character manipulation functions.

* Length()
* Concat()
* Reverse()
* Substr()—it is a function used to extract a string from a given string. It accepts three arguments.

Syntax:--- select substr(string, position, length) from table\_name;

String: it is an arg which we passed.

Position: it is a number which specifies the starting position of a string.

Length: the length of the string.

* Instr()-it is used to check whether a substring is present in a string or not. If a substring is not present then this function will return 0 otherwise it will return the integer number(position of the substring).

Syntax:--

Select instr(string,substring) from tablename;

String:- it is a string in which we have to search.

Substring :--it is a substring which we have to search.

* Replace()---It is used to replace the character or a substring in a given string with the new substring .

Syntax:--select replace(string,substring1,substring2) from tablename;

EXAMPLES :--

Q. WAQ to display all the names in upper case

ANS. select upper(name) from employee;

Q. WAQ to display all the names in Lower case

ANS. select lower(name) from employee;

Q. . WAQ to display the length of the names of an employees.

ANS. select length(name) from employee;

Q. WAQ to merge name and salary.

ANS. select concat(name,salary) from employee;

Q. WAQ to reverse all the names of an employees.

ANS. select reverse(name) from employee;

Q. WAQ to display name of the employee only three characters.

ANS. select substr(name,1,3) from employee;

Q. WAQ to display all the details of an employee whose name begin with consonant

ANS. select \* from employee where substr(name,1,1) not in('a','e','i','o','u');

Q. WAQ to search the position of ‘a’ in the name column.

ANS. select instr(name,'a') from employee;

Q. WAQ to replace where ‘a’ is present in the name with ‘j’.

ANS. select replace(name,'a','j') from employee;

Q. WAQ to display number of employees who is working in dept 30 and salary is 15000.

ANS. select count(\*) from employee where deptId=10 and salary=20000;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**GROUP BY**

The group by statement group the rows that have the same values into summary rows.

The group by statement is often used with aggregate functions (count,min,max,sum,avg) to group the result-set by one or more columns.

Syntax:--

Select col\_name(s) from table\_name where condition group by col\_name(s);

EX. select count(\*) from employee group by deptId;

NOTE : -- group by clause execute row by row. After the execution of group by the records are grouped. Therefore, all the clauses are executed after the execution of group by clause.

Group by clause is often is used with navigate functions.

Q. WAQ to display highest salary of employee in each department.

ANS. select deptId, max(salary) as max\_salary from employee group by deptId;

Q. WAQ to display number of developers working in each department.

ANS. select deptId, designation, count(\*) from employee where designation='developer' group by deptId;

Q. WAQ to display number of employees hired on each date.

ANS. select count(\*) from employee group by dateofjoin;

Q. WAQ to display minimum salary in each designation

ANS. select designation, min(salary) from employee group by designation;

**HAVING**

Having clause is often used with the group by clause to filter groups based on a specified condition.

In having clause we can use only column, i.e., used in group by clause

Syntax: --

Select col\_name(s) from table\_name where condition GROUP BY col\_name(s) HAVING condition.

**order of execution** : from=>where=>group by=>having=>select

**NOTE** :-- having clause executes after the execution of group by clause. Therefore it checks the given condition after group by.

Having clause can we use only in column that is used by group by.

Q. WAQ to display dept number which has atleast four working employees.

ANS. select deptId from employee group by deptId having count(\*)>=4;

Q. WAQ to display dept number which has total salary greater than 20000

ANS. select sum(salary),deptId from employee group by deptId having sum(salary)>20000;

Q. WAQ to display dept no and avg salary of all the departments whose avg salary is greater than 20000 excluding the employee whose name begins with ‘s’.

ANS . select deptId, avg(salary) from employee where name not like('s%') group by deptId having avg(salary)>20000 ;

**ORDER BY**

Order by clause is used to sort the results in the ascending or descending order.

By default it will sort the results in ascending order. To sort the records in descending order use the DESC keyword.

Syntax:--

Select col1,col2,….. from table\_name order by col1,col2,…..ASC/DESC;

**Order of execution**: from=>where=>select=>order by

Q. WAQ to display all the records of the employee salary should be in descending order.

ANS. select \* from employee order by salary desc;

Order by clause always executes after the execution of select clause.

Order by clause should be always last clause to be return in the syntax.

If the group by clause is used then order by clause we can use aggregate function or multi row function Only the column name which is use in the group by clause.

Q. WAQ to display name, designation, date of join for all employees who are working in either any of the departments 10,20,60,50 and sort the results in descending order of their salary.

ANS. select name, designation, dateofjoin from employee where deptId in(10,20,50,60) order by salary desc;

Q. WAQ to display deptId along with number of employees in it.

ANS. select deptId, count(\*) from employee group by deptId;

Q. WAQ to display all the details of an employee whose designation is developer or tester.

ANS. select \* from employee where designation in('developer','tester');

**SUB QUERES**

A Query within another query is known as sub-query. It is also called as inner query or nested query.

Sub-query is also a another way to fetch the data from multiple tables.

Whatever the inner query we are writing it should be written in parenthesis.

NOTE :--- Always inner query will execute first.

The output of an inner query will be taken as input for the outer query and then the final result we will get from the outer query.

Q. When we use sub query in where clause ?

Scenario-1: If the given condition contain unknown value then we can use sub query to obtain the unknown value.

Scenario-2: if the column to be selected and the condition to be filter a given for two different tables we can use sub-query.

Two types are there in sub-queries

1. Single row subquery
2. Multi row subquery.

1.Single row subquery:-----

If the single value operators such as equal to, less than, greater than, equal to , not equal to.

We will call it as “single row subquery”.

Syntax:--Select col1,col2,……from table where col1 operator(select col from table where exp1=value);

2.Multi row subquery :--

If the operator such as multi value operator(IN) are used.

We can call it as “multi row subquery”.

Syntax:--

Select col1,col2,…….. from table where col1 operator(select column from table where exp1=value.

Q. WAQ to display name of the employee who is working for research dept.

ANS. select name from employee where deptId=(select deptId from department where dname='research');

Q. WAQ to display employee who is working in location which has atleast one ‘a’ character in its name.

ANS. select name from employee where deptId IN(select deptId from department where location like('%a%'));

Q. WAQ to obtain second maximum salary from the employee table.

ANS. select max(salary) from employee where salary <(select max(salary) from employee);

Q. WAQ to display the third maximum salary

ANS. select max(salary) from employee where salary <(select max(salary) from employee where salary<(select max(salary) from employee));

**JOINS**

Join is used to fetch the data from two or more tables.

It is used for combining column from two or more tables by using values common to both the tables.

**JOIN** keyword is used in SQL queries for joining two or more tables.

Minimum required condition for joining tables is, at least number of tables should be two.

**Types of joins:**

1. **Cross join/cartesian join :--**

Record from one table is merged with each record with the another table (or) join two tables records from the left table will merge with each and every record with the right table.

Syntax :----

Select col\_name-list from tab1 cross join tab2;

Q. WAQ to display to display all the records using the Cartesian

ANS. select \* from employee cross join department;

2**. Inner join: --**

Joining two tables with the help of join(common column) column is known as inner join.

It will compare the records from left table with the records from right table if both the tables have the common records then it will select that column.

Syntax: --

Select \* from table1 inner join table2 on condition.

Ex: --

select \* from employee inner join department where salary>20000;

**3. Outer join: --**

i. left-outer join.

ii. Right-outer join.

iii. full-outer join.

**i. left-outer join: --**

left outer join gives inner join plus a unmatched record of left table. The record do not have pair in the opposite table with respective joins.

Syntax: --

Select \* from table1 left join table2 on condition;

Ex: --

Select \* from employee left join department on employee.deptId=department.deptId;

ii. **right-outer join: --**

right outer join gives inner join plus an unmatched record of right table. The record do not have pair in the opposite table with respective joins.

Syntax: --

Select \* from table1 right join table2 on condition;

Ex: --

Select \* from employee right join department on employee.deptId=department.deptId;

**iii. Full outer join: --**

full outer join if inner join as well as all the records from both the table which do not have any pair.

Syntax: --

**Key in RDBMS**

**1. key attributes: --**

Key attributes is used to find the attributes using which we can uniquely determine the record in a table.

**2. Non-key attributes: --**

All the attributes except key attributes are known as non-key attributes.

**3. Primary key attributes: --**

A key attributes which is choosen to be a main attributes to determine the records uniquely in the table is known as primary key attributes.

**4. Non-Primary key attributes: --**

All the attributes except primary key attributes are known as non primary attributes.

**5. Composite key attributes: --**

The combination of two or more attributes which determines a record uniquely is known as composite key.

**6. Foreign key attributes: --**

Foreign key attributes are used to relate two tables.

**Functional Dependency**

In functional dependency a relation exists such that an attribute determine another attribute uniquely is known as functional dependency.

Ex: -- R={x, y}

Y is dependent on X

X ---> determinant

Y ----> dependent

Types of functional dependency :--

1. Total functional dependency

2. Partial functional dependency

3. Transitive functional dependency.

**1. Total Functional Dependency: --**

All the attributes of a relation is determine by key attributes is known as total functional dependency.

R={a, b, c, d}

if a -->b, c, d then a-->b, a-->c, a-->d

**2. Partial Functional Dependency: --**

Relation is said to have partial functional dependency if 1. It consists of composite attribute

2. they exists a dependency such that an attribute can determine by another attribute which is a part of composite key.

R={a, b, c, d}

(a, b) ---->c

(a, b) ---->d

(a, b) ---->c, d

a ---->c

**3. Transitive Functional Dependency: --**

Relation is said to be in transitive functional dependency if there exists a relation such that an attribute is determine by non-key attributes which interns is determined by key attributes.

R= { a, b, c, d}

a --->c

c ---->d

a---->d

**Normalization**

1. Data Redundancy

2. anomaly

The side effects that occurring while performing DML operation.

Types of anomaly insert, update, delete.

The process of decomposing the table into smaller tables in order to remove dependency and anomalies by identifying dependencies is known as normalization.

(or)

The process of reducing the table into normal forms is known as normalization .

**Normal Form: --**

A state of table without table redundancies and anomalies is known as normal form.

Types of Normal forms:

1. 1 normal form

2. 2nd normal form

3. 3rd normal form

4. BCNF

5. 4th normal form

**1. 1st normal form (1NF):**

A table is said to be in 1NF if it satisfies the following conditions.

1. a table does not contain duplicates

2. every cell in a row should be single value.

**2. 2nd normal form (2NF):**

a table is in 2NF if it follow the following conditions

1. A table should be in 1NF

2. A table should not have partial functional dependency(should not have composite key).

Note: -- If the table consists partial functional dependency the attributes which are possible are remove from the table.

**3. 3nd normal form (3NF):**

A table is in 3NF if the following conditions is satisfied

1. it should be in 2NF

2. A table should not have transitive functional dependency.

Note: --- 1NF, 2NF and 3NF these are based on key attributes and functional dependency of relational schema.

**ER Diagrams**

ER diagram is an entity relationship diagram describes a structure of a database with the help of diagram is known as ER diagram. It is a blue print of a database that can be later used to implement a database.

Components of ER diagrams:

1. Entity

2. Attributes.

3. Relationship

Entity Attributes

Relationship It combines the components

studying

student

college

Relationships: --

1. one to one

2. one to many

3. many to one

4. many to many

**View**

View is a logical or virtual table that can be created from the existing table.

View do not occupy memory.

Syntax: --

Create view view-name as select \* from table-name;

=> select \* from view-name; //to retrieve the records.

Ex: --

Crea;te view emp1 as select \* from department;

select \* from emp1 where deptId=10;

Note: -- Any DML operation perform on view it will reflect on table.

**TCL(Transaction Control Language):**

Commit

Rollback

savepoint: it only work with rollback.

**Stored Procedure: --**

Syntax: --

DECLIMITER$$

Create procedure <procedure Name>

As

Begin

<SQL statements>;

End $$

DECLIMITER;