**\*\*\*\*\*\*\*\*\*\*\*Database Testing\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**What is SQL?**

- SQL stands for structured query Languages'

- SQL Was developed by IBM in 1970s.

- SQL is standard language used to communicate with RDBMS (Relational Database Management system).

- Famouse RDHBMS as like oracle, mysql, sql server, ms access, mongo DB, db2, sybage, aws db etc...

**What is Use of SQL query?**

- create database using SQL query.

- create table in database.

- insert data in table

- update data in table

- delete data from table

- retrieve the data from table

- add a new columns / remove the columns

- join the tables

- group the data

**SQL Queries are classified into the groups based on their nature.**

we have different groups in SQL

1) Data Definition Language (DDL)

2) Data Manipulation Language (DML)

3) Data Query Language (DQL)

4) Data Control Language (DCL)

5) Transitional Control Language (TCL)

1) create new database by using SQL Query

create database databasename;

e.g

create database facebook;

2) create new table

create table register (firstname varchar, lastname varchar, username varchar,password varchar, date int,month varchar, year int,boolean varchar )

SQL

there different groups in SQL

1) Data Definition Language (DDL)

2) Data Manipulation Language (DML)

3) Data Query Language (DQL)

4) Data control Language (DCL)

5) Transitional Control Language (TCL)

**1) Data Definition Language (DDL)**

there are different query present in DDL group

i) create

ii) Alter

iii) Drop

iv) Truncate

i) create Query

- there are 2 use of create query

i) by using create query we can create Database

ii) by using create query we can tables, (always we create table inside the database).

one datase contains we have multiple tables

and each and every table contains we have multiple rows and multiple columns.

i) by using create query we can create Database

syntax:

create database databasename;

e.g.

create database mastercard;

how to select database using query/command

syntax:

use databasename;

e.g

use mastercard;

ii) by using create query we can tables,

it is used to create table structure.

syntax":

create table tablename(columnName1 Datatype1,columnName2 Datatype2,columnName3 Datatype3,columnName4 Datatype4);

e.g

create table employee (id int, firstname varchar(20), lastname varchar(20), email varchar(30), age int);

**3) Data Query Language (DQL)**

i) select

it is used to retrieve the table information

syntax:

select \* from tablename;

e.g

select \* from employee;

**2) Data Manipulation Language (DML)**

there are 3 main query /command in DML

1) insert

2) update

3) delete

1) insert

it is used to insert the new records/rows in table

syntax:

insert into tablename values (value1,value2, value3,value4,value5);

e.g

insert into employee values (1,'Nikita','Patil','nikita@gmail.com',23);

syntax:

insert into tablename (columnName1,columnName2,columnName3,columnName4) values (value1,value2,value3,value4);

e.g

insert into emp (id,firstname,email,age) values (4,'Shital','shital@gmail.com',23);

e.g

insert into emp (age,firstname,id,email) values (23,'shital',4,'shital@gmail.com');

create database mastercard;

use mastercard;

create table emp (id int, firstname varchar(20) , lastname varchar(20), emaild varchar(30), age int);

select \* from emp;

insert into emp values (1,'Nikita','Patil','nikita@gmail.com',23);

select \* from emp;

insert into emp values (2,'Ajit', 'Sathe', 'ajit@gmail.com', 25);

select \* from emp;

insert into emp values (3, 'Rohit', 'Kumbhar','rohit@gmail.com',27);

select \* from emp;

insert into emp (id,firstname,emaild,age) values (4,'shital','shital@gmail.com',23);

select \* from emp;

select firstname,lastname from emp;

select \* from emp;

select id, firstname,age from emp;

select \* from emp;

select \* from emp where id=2;

select \* from emp where firstname='Rohit';

select \* from emp where age=23;

select \* from emp;

select id, lastname,age from emp where firstname='Nikita';

create table personalInfo (id int,firstname varchar(20),city varchar(20),mobileNumber bigint);

select \* from personalInfo;

insert into personalInfo values (101,'Shubham','Pune',90909090);

insert into personalInfo (id,firstname,city,mobileNumber) values (102,'Anjali','Mumbai',808080);

insert into personalInfo(city,id,firstname)values ('Mumbai',103,'Omkar');

select \* from personalInfo;

select \* from personalInfo where city='Mumbai';

SQL Queries classified into multiple group based on their nature.

**1) Data Definition Language (DDL) group**

i) Create

2) Alter

3) drop

4) truncate

**2) Data Maniupulation Language (DML) group**

1) insert

2) update

3) delete

**3) Data Query Language (DQL) group**

1) select

**4) Data Control Language (DCL) group**

1) grant

2) revoke

**5) Transitional Control Language (TCL) group**

1) commit

2) roll back

3) save point/save transaction

1) Create Query:

- we can create the table as well as database

e.g

create database databasename;

e.g.

create database amazon;

syntax:

create table tablename (columnName datatype,columnName datatype,columnName datatype,columnName datatype,columnName datatype )

e.g

create table employee (id int identity(1001,1), firstname varchar(10),lastname varchar(10),address varchar(30),mobileNumber bigint,emailId varchar(30), DOB date);

identity(1001,1) - Auto increment the Id by 1 and starts from 1001

date YYYY-MM-DD

2) Insert Query:

- there are 2 ways we can add to the data/test data/inputs to table

syntax;

insert into tablename values (value1,value2,value3,value4,value5);

e.g

insert into employee values('Sayali','Patil','Pune',90909090,'sayali@gmail.com','2010-12-24');

syntax:

insert into tablename (columnName1,columnName2,columnName3,columnName4,columnName5) values (value1,value2,value3, value4,value5);

e.g

insert into employee (firstname,lastname,address,mobileNumber,emailId,DOB) values ('Sayali','Patil','Pune',90909090,'sayali@gmail.com','2010-12-24');

3) select query

- it is used to retrieve the data from table

syntax:

select \* from tablename;

e.g

select \* from employee

4) update query

- it used to update the records in table

syntax:

update tablename set columnName=value where columnName=value;

e.g

update employee set address ='Delhi' where id=1003;

e.g

update employee set address ='Delhi' where firstname='Rahul';

5) alter query

- it used to add or remove the columns

syntax

- alter table tablename add columnname datatype;

e.g

alter table employee add salary int;

syntax:

alter table tablename drop column columnname;

e.g

alter table employee drop column address;

6) delete query:

- it used to delete the records from tables

- we can delete the records as per the row wise and table wise.

- data can be roll back using roll back query.

- It is present inside the DML group

syntax:

delete from tablename where columnName=value;

e.g

delete from employee where id=1004;

delete from employee;

7) truncate query

-is used to delete the all records from table but as per the table wise.

- it is faster than delete query.

- data can not be roll back

- it is present inside the DDL group.

syntax:

truncate table tablename;

e.g

truncate table employee;

8) drop query:

- it used to delete/drop the table as well as database.

- data can not be roll back .

- it is present inside the DDL group.

syntax:

drop table tablename;

drop database databasename;

e.g

drop table employee;

drop database facebook;

create database cybage;

use cybage;

create table employee (id int identity(1001,1),firstname varchar(10),lastname varchar(10),address varchar(30)

,emailId varchar(30),DOB date);

select \* from employee;

insert into employee values ('Sayali','Patil','Pune','sayali@gmail.com','2010-12-24');

insert into employee (firstname,lastname,emailId,DOB)values ('Aboli','Shinde','aboli@gmail.com','2011-01-18');

select \* from employee;

insert into employee (firstname,lastname,emailId,DOB)values ('Rahul','Naik','rahul@gmail.com','2001-03-20');

select \* from employee where id=1003;

update employee set address='Delhi' where id=1003;

update employee set address='Dubai' where firstname='Rahul';

select \* from employee;

update employee set emailId='rahulnaik@gmail.com' where id=1005;

alter table employee add salary int;

select \* from employee;

update employee set salary =100000;

select \* from employee;

update employee set salary=200000 where id=1003 or id=1005;

update employee set salary=3000000 where address='Pune' or address='Delhi';

select \* from employee;

update employee set address='Mumbai';

alter table employee drop column address;

alter table employee add gender varchar(10);

select \* from employee;

update employee set gender='female' where id=1001 or id=1003;

update employee set gender='male' where firstname='Rahul';

alter table employee drop column gender;

select \* from employee;

delete from employee where id=1004;

insert into employee values

('Anita','Walke','anita@gmail.com','2010-10-10',2000000),

('Nikhil','Patil','nikhil@gmail.com','2014-10-10',2200000),

('Prajakta','Shah','prajakta@gmail.com','2015-10-10',2500000),

('Punam','Mahajan','puname@gmail.com','2016-04-10',1700000);

select \* from employee;

delete from employee where lastname='Patil';

delete from employee;

select \* from employee;

truncate table employee;

drop table employee;

drop database cybage;

use master;

drop database abc;

Ass;

diff between DDL and DML group

SQL Query

1) Like operator

- it used to search specified pattern in columns

- there are 2 types of wild cart in SQL

1) % symbol it used to ignore the multiple characters

2) \_ symbol it used to ignore the single character

syntax:

select \* from tablename where columnName like 'pattern';

e.g

select \* from employee where firstname like 'r%'

it search all employee whose firstname starts with r

select \* from employee where firstname like '%a'

it search all employee whose firstname ends with a

select \* from employee where firstname like '\_a%'

it search all employee whose firstname second character is a

select \* from employee where firstname like '\_a%p\_'

it search all employee whose firstname second character is a and second last character is p.

select \* from employee where firstname like '\_p\_a%'

it search all employee whose firstname second character is p and 4th character is a.

2) order by clause

- it used to, order the columns as per the ascending or descending order.

syntax:

select \* from tablename order by columnname asc;

select \* from tablename order by columnname desc;

- default order is ascending order.

select \* from employee order by salary asc;

it display salary in ascending order

select \* from employee order by salary;

it display salary in ascending order

select \* from employee order by salary desc;

it display salary in descending order

select \* from employee order by firstname desc;

it display the firstname in descending order

select \* from employee order by firstname asc;

it display the firstname in ascending order

3) max() function

- it used to find the maximum columns values.

syntax

select max(columnname) from tablename;

e.g

select max(salary) from employee;

e.g

select max(firstname) from employee;

it find the max firstname values.

can we use max() function for varchar datatype?

yes

4) min() function

- it is used to find the minimum column values

syntax

select min(columnname) from tablename;

e.g

select min(salary) from employee;

e.g

select min(firstname) from employee;

can we use min() function for varchar datatype?

yes

5) avg() function

- it used to display the average values

syntax

select avg(int columnname) from tablaname;

e.g

select avg(salary) from employee

can we use avg unction for varchar datatype?

No

6) count() function

- it used to count how many records in table

syntax:

select count(columnName) from tablename;

e.g

select count(\*) from employee;

select count(id) from employee;

select count(firstname) from employee;

7)distinct() function

it is used to remove the duplicate values

syntax:

select distinct(columname) from tablename;

e.g

select distinct(id) from employee;

select distinct(firstname) from employee;

8) dense-rank() function

it will count total records in table and it will group duplicate values.

over(order by salary desc)

it will display salary column in descending order.

with clause : it used to create temporary table.

syntax:

with temporaryTablename as

(

select \*,dense\_rank() over (order by columnName desc) as temporaryColumnName from tablename

)

select \* from temporaryTablename;

e.g.

with ctc as

(

select \*, dense\_rank() over (order by salary desc) as temprank from employee

)

select \* from ctc where temprank=1;

with ctc as

(

select \*, dense\_rank() over (order by salary desc) as temprank from employee

)

select \* from ctc where temprank=4;

with ctc as

(

select \*, dense\_rank() over (order by salary desc) as temprank from employee

)

select \* from ctc where temprank=26;

with ctc as

(

select \*, dense\_rank() over (order by salary desc) as temprank from employee

)

select \* from ctc where temprank=101;

create database cybage;

use cybage;

create table employee (id int identity(1001,1),firstname varchar(10),lastname varchar(10),address varchar(30)

,emailId varchar(30),DOB date);

select \* from employee;

insert into employee values ('Sayali','Patil','Pune','sayali@gmail.com','2010-12-24');

insert into employee (firstname,lastname,emailId,DOB)values ('Aboli','Shinde','aboli@gmail.com','2011-01-18');

select \* from employee;

insert into employee (firstname,lastname,emailId,DOB)values ('Rahul','Naik','rahul@gmail.com','2001-03-20');

select \* from employee where id=1003;

update employee set address='Delhi' where id=1003;

update employee set address='Dubai' where firstname='Rahul';

select \* from employee;

update employee set emailId='rahulnaik@gmail.com' where id=1005;

alter table employee add salary int;

select \* from employee;

update employee set salary =100000;

select \* from employee;

update employee set salary=200000 where id=1003 or id=1005;

update employee set salary=3000000 where address='Pune' or address='Delhi';

select \* from employee;

update employee set address='Mumbai';

alter table employee drop column address;

select \* from employee;

alter table employee drop column address;

select \* from employee;

insert into employee values

('Anita','Walke','anita@gmail.com','2010-10-10',2000000),

('Nikhil','Patil','nikhil@gmail.com','2014-10-10',2200000),

('Prajakta','Shah','prajakta@gmail.com','2015-10-10',2500000),

('Punam','Mahajan','puname@gmail.com','2016-04-10',1700000);

select \* from employee;

select \* from employee where firstname like 'P%';

select \* from employee where firstname like '%a';

select \* from employee where firstname like '\_r%';

select \* from employee where firstname like '\_\_k%';

select \* from employee where firstname like '%h%'

select \* from employee where firstname like 's%' or lastname like 'p%';

select \* from employee where firstname like '%t\_';

select \* from employee where salary like '2%';

select \* from employee order by salary asc;

select \* from employee order by firstname asc;

select \* from employee order by firstname desc;

select \* from employee order by salary desc;

select \* from employee order by DOB asc;

select \* from employee order by DOB desc;

select \* from employee order by salary;

select max(salary) from employee;

select \* from employee where salary = 2500000;

select \* from employee where salary in (select max(salary) from employee);

select MAX(firstname) from employee;

select \* from employee where firstname in (select max(firstname) from employee);

select min(salary) from employee;

select \* from employee where salary in (select min(salary) from employee);

select min(lastname) from employee;

select avg(salary) from employee;

select avg(firstname) from employee;

select avg(DOB) from employee;

select avg(id) from employee;

select \* from employee;

select count(\*) from employee;

update employee set id=1005 where firstname='Punam';

update employee set salary=2700000 where id=1007;

select \*, DENSE\_RANK() over (order by salary desc) as temprank from employee

with pune as

(

select \*,DENSE\_RANK() over (order by salary desc) as tempRank from employee

)

select \* from pune where tempRank=1;

with ctc as (

select \*,DENSE\_RANK() over (order by salary desc) as mumbai from employee

)

select \* from ctc where mumbai=2;

select \* from employee;

select top 10 \* from employee;

with pune as

(

select \*, DENSE\_RANK() over (order by salary desc) as tempRank from employee

)

select \* from pune where tempRank=2;

**Join Query** **VVVVVVVVIMMPPPPPPPPPPPPPPPPPP**

- Join query is used to join 2 or more tables.

- there are 4 main types of joins in SQL

1) inner Join

2) Left join

3) right join

4) full join

**1) inner Join**

- it is also known as default join.

- if we join 2 table by using inner join query then it will display matching records from both tables depends on id.

syntax:

select \* from tableName1 inner join tablename2 on tableName1.id = tableName2.id;

e.g

select \* from employee inner join dept on employee.empid=dept.deptid;

select \* from employee as e inner join dept as d on e.empid=d.deptid;

select employee.empid,employee.fname,employee.city,dept,deptName,dept.projectName from employee inner join dept on employee.empid=dept.deptid;

select e.empid,e.fname,e.city,d.deptName,d.projectName from employee as e inner join dept as d on e.empid=d.deptid;

- it is also known as default join.

select e.empid,e.fname,e.city,d.deptName,d.projectName from employee as e join dept as d on e.empid=d.deptid;

**2) Left join**

- if we join 2 table by using left join query then it will display all records from left table and matching records from other table depends on id.

- if it is unmatching records then it display values as null.

syntax:

select \* from tableName1 left join tablename2 on tableName1.id = tableName2.id;

e.g.

select \* from employee left join dept on employee.empid=dept.deptid;

select \* from employee as e left join dept as d on e.empid=d.deptid;

select employee.empid,employee.fname,employee.city,dept,deptName,dept.projectName from employee left join dept on employee.empid=dept.deptid;

select e.empid,e.fname,e.city,d.deptName,d.projectName from employee as e left join dept as d on e.empid=d.deptid;

**3) right join**

- if we join 2 table by using right join query then it will display all records from right table and matching records from other table depends on id

- if it is any unmatching records then it display value as null.

syntax:

select \* from employee right join dept on employee.empid=dept.deptid;

select \* from employee right join dept on employee.empid=dept.deptId;

select dept.deptId,dept.deptName,employee.fname,employee.city from employee right join dept on employee.empid=dept.deptId;

select employee.fname,dept.deptId,dept.deptName,employee.city from employee right join dept on employee.empid=dept.deptId;

select dept.deptId,employee.fname,dept.deptName,employee.city from employee right join dept on employee.empid=dept.deptId;

select d.deptId,d.deptName,e.fname,d.salary,e.city,d.projectName from employee as e right join dept as d on e.empid=d.deptId;

**4) full join**

- if we join 2 table by using full join query then it will display all records from both the tables depends on id

- if it is any unmatching records then it display value as null.

syntax:

select \* from employee full join dept on employee.empid=dept.deptid;

select e.empid,e.city,d.salary,d.projectName from employee as e full join dept as d on e.empid=d.deptId;

5) cross join

select \* from employee cross join dept;

Have u used cross join?

No,

Create database amdocs;

use amdocs;

create table employee (empid int, fname varchar(10),lname varchar(10),city varchar(20));

insert into employee values

(1, 'Ajit','Patil','Pune'),

(2, 'Sonali','Khalate','Pune'),

(3, 'Jyoti','More','Mumbai'),

(4, 'Rohit','Sathe','Delhi'),

(10, 'Punam','Patil','Dubai'),

(11, 'Om','Joshi','USA'),

(12, 'Sohil','Kamble','UK'),

(13, 'Sagar','Chavan','USA');

create table dept (deptId int, deptName varchar(20), salary int, projectName varchar(20));

insert into dept values

(1,'Testing',1800000,'CRM Application'),

(2,'Dev',2200000,'CRM Application'),

(3,'BA',1500000,'CRM Application'),

(20,'DevOps',1600000,'BSS Application'),

(21,'BA',1100000,'BSS Application '),

(22,'Testing',1700000,'BSS Application'),

(23,'Dev',2400000,'OSS Application'),

(24,'DevOps',1500000,'OSS Application');

select \* from employee;

select \* from dept;

select \* from employee inner join dept on employee.empid= dept.deptId;

select \* from employee as e inner join dept as d on e.empid=d.deptId;

select employee.empid, employee.fname, dept.deptName,dept.projectName from employee inner join dept

on employee.empid=dept.deptId;

select e.fname,e.city,d.deptName,d.salary from employee as e inner join dept as d on e.empid=d.deptId;

select \* from employee;

select \* from dept;

select \* from employee left join dept on employee.empid=dept.deptId;

select e.empid, e.fname, d.deptName,d.salary from employee as e left join dept as d on e.empid=d.deptId;

select employee.empid,employee.fname,employee.city ,dept.projectName,dept.salary from employee left join dept

on employee.empid=dept.deptId;

select \* from employee right join dept on employee.empid=dept.deptId;

select dept.deptId,dept.deptName,employee.fname,employee.city from employee right join dept on employee.empid=dept.deptId;

select employee.fname,dept.deptId,dept.deptName,employee.city from employee right join dept on employee.empid=dept.deptId;

select dept.deptId,employee.fname,dept.deptName,employee.city from employee right join dept on employee.empid=dept.deptId;

select d.deptId,d.deptName,e.fname,d.salary,e.city,d.projectName from employee as e right join dept as d on e.empid=d.deptId;

select \* from employee full join dept on employee.empid=dept.deptId;

select e.empid,e.city,d.salary,d.projectName from employee as e full join dept as d on e.empid=d.deptId;

select \* from employee cross join dept;

create table customer(custId int, custName varchar(20),custAddress varchar(20));

insert into customer values

(1,'Sonali','INDIA'),

(2,'Pavan','UK'),

(101,'TOM','USA'),

(102,'Sajid','Dubai'),

(103,'Manoj','UK');

insert into customer values

(10,'Aboli','INDIA'),

(20,'Sahil','INDIA');

select \* from customer;

select \* from employee inner join dept on employee.empid=dept.deptId

inner join customer on employee.empid=customer.custId;

select \* from employee inner join dept on employee.empid=dept.deptId

inner join customer on dept.deptId=customer.custId;

select \* from employee left join dept on employee.empid=dept.deptId

inner join customer on employee.empid=customer.custId;

select \* from employee left join dept on employee.empid=dept.deptId

left join customer on employee.empid=customer.custId;

select \* from employee left join dept on employee.empid=dept.deptId

right join customer on employee.empid=customer.custId;

select \* from employee right join dept on employee.empid=dept.deptId

inner join customer on dept.deptId=customer.custId;

select \* from employee right join dept on employee.empid=dept.deptId

left join customer on employee.empid=customer.custId;

select \* from employee right join dept on employee.empid=dept.deptId

right join customer on employee.empid=customer.custId;

select \* from employee full join dept on employee.empid=dept.deptId

inner join customer on employee.empid=customer.custId;

select \* from employee full join dept on employee.empid=dept.deptId

inner join customer on dept.deptId=customer.custId;

select \* from employee cross join dept cross join customer

Database Testing: //DAO database Access Object by using Spring boot framework

========================================================================================================

SQL Constraints: VVVVVVVVIMMMPPPPPPP

-------------------------------------------------------------------------------------------------------------------------------------------------------------------

SQL Constraints are used to apply rules on table columns.

there are different constraints in SQL

1) Primary key it is combination of unique key and not nulll key.

2) not null we can not add values as null/ blank column.

3) unique ` each and every column values should be unique.

4) identity auto increment , it will increment the values automatically.

5) check if u want to check if person age is above 21 or not? if above then create entity

6) default if we have default valus for any object then we use default constraints

7) foreign key if reference key of primary key.

-------------------------------------------------------------------------------------------------------------------------------------------------------------------

create table employee (empid int primary key identity(101,1), fullName varchar(20) not null, email varchar(30) unique, age int check(age>21),

city varchar(20) default 'Pune');

insert into employee values ('Sonali Khalate', 'sonali@gmail.com',24);

create table dept (empid int foreign key references employee(empid), deptName varchar(30), salary int not null, projectName varchar(30));

create database citiBank;

create table employee (empid int primary key identity(101,1),fullName varchar(20) not null, email varchar(30) unique,

age int check(age>21), city varchar(20) default 'Pune');

select \* from employee;

insert into employee

(fullName,email,age) values ('Sujit Kande','sujit@gmail.com',22);

create table dept (empid int foreign key references employee(empid),deptName varchar(30), salary int not null,

projectName varchar(30));

insert into dept values (101,'Testing',180000,'CRM Application');

insert into dept values (102,'Dev',190000,'BSS Application');

insert into dept values (106,'DevOps',2200000,'CRM Application');

select \* from employee;

select \* from dept;

select \* from employee inner join dept on employee.empid=dept.empid;

select \* from employee left join dept on employee.empid=dept.empid;

select \* from employee full join dept on employee.empid=dept.empid;

**Store Procedure:** **VVVVVVVVIMMPPPP**

if we are using same query again and again then we store the specific query in store procedure.

there are 2 types of store procedure

1) Store Procedure without parameters

2) Store Procedure with parameter;

syntax:

create procedure procedureName

as

Your query

go;

if u want to run above query

syntax:

exec procedureName;

1) Store Procedure without parameters

syntax:

create procedure abc

as

select \* from employee inner join dept on employee.empid=dept.empid

go;

exec abc;

2) Store Procedure with parameter;

syntax:

create procedure procedureName

@procedureParameterName int

as

select \* from employee where empid=@procedureParameterName

go;

e,g

create procedure xyz

@id int

as

select \* from employee where empid=@id

go

exec xyz @id=102

scenario:

retrieve single employee information select \* from employee where empid=101

update single employee age update employee set age=25 where empid=101

get updated single employe information select \* from employee where empid=101

create procedure pune

@id int, @age int

as

select \* from employee where empid=@id

update employee set age=@age where empid=@id

select \* from employee where empid=@id

go;

exec pune @id=101, @age=30;

exec pune @id=102, @age=27;

create database citiBank;

create table employee (empid int primary key identity(101,1),fullName varchar(20) not null, email varchar(30) unique,

age int check(age>21), city varchar(20) default 'Pune');

select \* from employee;

insert into employee

(fullName,email,age) values ('Anjali More','anjali@gmail.com',22);

create table dept (empid int foreign key references employee(empid),deptName varchar(30), salary int not null,

projectName varchar(30));

insert into dept values (101,'Testing',180000,'CRM Application');

insert into dept values (102,'Dev',190000,'BSS Application');

insert into dept values (106,'DevOps',2200000,'CRM Application');

select \* from employee;

select \* from dept;

select \* from employee inner join dept on employee.empid=dept.empid;

select \* from employee left join dept on employee.empid=dept.empid;

select \* from employee full join dept on employee.empid=dept.empid;

create procedure abc

as

select \* from employee inner join dept on employee.empid=dept.empid

go

exec abc;

exec abc;

create procedure pqr as select \* from employee go

exec pqr;

create procedure xyz

@id int

as

select \* from employee where empid=@id

go

exec xyz @id=111;

select \* from employee;

create procedure pune

@id int, @age int

as

select \* from employee where empid=@id

update employee set age=@age where empid=@id

select \* from employee where empid=@id

go

exec pune @id=101, @age=30;

exec pune @id=108, @age=25;

exec pune @id=111, @age=38;

select \* from employee;

create procedure updateCityInfo

@empid int, @city varchar(20)

as

select \* from employee where empid=@empid

update employee set city=@city where empid=@empid

select \* from employee where empid=@empid

go

exec updateCityInfo @empid=101, @city='Dubai';

exec updateCityInfo @empid=102, @city='USA';

select \* from employee;

create procedure mumbai

@empid int, @age int, @city varchar(20)

as

select \* from employee where empid=@empid

update employee set age=@age where empid=@empid

update employee set city=@city where empid=@empid

select \* from employee where empid=@empid

go

exec mumbai @empid=101, @age=22, @city='UK';

exec mumbai @empid=108, @age=26, @city='USA';

**DCL group:**  DB Admin We dont have access of DCL command

1) grant

2) revoke

**TCL group**  Yes we have access

1) commit

2) rollback

3) Save Transaction

Create Dont have access

alter dont have access

Drop dont have access

truncate dont have access

Grant dont have access

Revoke dont have access

select yes we have access

update yes we have access

delete yes we have access

insert yes we have access

commit yes we have access

rollback yes we have access

Save Transaction yes we have access

Database -- 20/30/40+

Database contains we have 50-70-80 tables

one table contains we have 30-80 columns

one table contains we have 1 lakhs rows

1) Find all employee who is working in Testing department --

select \* from employee where department\_name='Testing';

2) find all employee who is working in HR and Development department --

select \* from employee where department\_name='HR' or department\_name='Development';

3) find all employee whose salary is 22,00,000

select \* from employee where salary=2200000;

4) find all employee whose salary is below 20,00,000

select \* from employee where salary<2000000

5) find all employee whose salary is above 20,00,000

select \* from employee where salary>2000000

6) find all employee whose salary between 22,00,000 to 30,00,000

select \* from employee where salary>2200000 and salary<3000000

or

select \* from employee where salary between 2200000 and 3000000

create database infosys;

use infosys;

create table employee (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(100),

department\_name VARCHAR(100),

salary int

);

insert into employee (employee\_id, employee\_name, department\_name, salary) values

(1, 'Monika', 'Testing', 1500000),

(2, 'Santosh', 'Development', 1700000),

(3, 'Rucha', 'HR', 1400000),

(4, 'Aboli', 'Testing', 1900000),

(5, 'Amol', 'Development', 2500000),

(6, 'Amruta', 'HR', 2200000),

(7, 'Sakshi', 'Testing', 2300000),

(8, 'Rutuja', 'Development', 2700000),

(9, 'Suraj', 'Development', 2900000),

(10, 'Rohit', 'HR', 1900000),

(11, 'Neha', 'Testing', 2000000),

(12, 'Nikhil', 'DevOps', 2100000),

(13, 'Monika', 'HR', 2400000),

(14, 'Manoj', 'Testing', 2500000),

(15, 'Gauri', 'DevOps', 2800000),

(16, 'Pankaj', 'HR', 2900000),

(17, 'Pravin', 'DevOps', 2300000),

(18, 'Pragati', 'Development', 1800000),

(19, 'Ravi', 'Development', 2300000),

(20, 'Rohan', 'DevOps', 2800000);

select \* from employee;

-- Find all employee who is working in Testing department

select \* from employee where department\_name='Testing';

-- find all employee who is working in HR and Development department --

select \* from employee where department\_name='HR' or department\_name='Development';

-- find all employee whose salary is 22,00,000

select \* from employee where salary= 2200000;

-- find all employee whose salary is below 20,00,000

select \* from employee where salary <2000000;

-- find all employee whose salary is equal or below 20,00,000

select \* from employee where salary <=2000000;

-- find all employee whose salary is above 20,00,000

select \* from employee where salary >200000;

-- find all employee whose salary is equal and above 20,00,000

select \* from employee where salary >= 200000;

-- find all employee who is salary between 25,00,000 to 30,00,000

select \* from employee where salary > 2500000 and salary<3000000;

select \* from employee where salary >= 2500000 and salary<= 3000000;

select \* from employee where salary between 2500000 and 3000000;

group by clause

- it used to groups row data that have same value.

- group by clause we use with aggregate functions as like count, avg, min, max, sum etc.

syntax:

select aggregatefunction(columnName) from tablename group by columnName;

e.g

count how many employee working in each and every department

select department\_name, count(department\_name) from employee group by department\_name;

count total salary paid to the employee as per department wise.

select department\_name, sum(salary) from employee group by department\_name;

count avg salary paid to the employee as per department wise.

select department\_name, avg(salary) from employee group by department\_name;

having clause we use after the group by clause

count how much salary given to HR Dept only

select department\_name, sum(salary) from employee group by department\_name having department\_name='HR'

find max salary as per the department\_name wise.

select department\_name, max(salary) from employee group by department\_name;

**TCL group**

begin transaction;

1) commit store data permanent in database.

syntax:

commit;

2) rollback go back to specific save transaction or save point

syntax:

rollback transaction transactionName;

3) save transaction create save transactionn so we can perform rollback operation

syntax

save transaction transactionName;

begin transaction;

select \* from employee;

save transaction a1;

select \* from employee;

update employee set department\_name='Testing';

select \* from employee;

rollback transaction a1;

select \* from employee;

commit;

begin transaction;

select \* from employee;

save transaction p1;

select \* from employee;

update employee set salary=100 where employee\_id=2;

save transaction p2;

select \* from employee;

delete from employee where employee\_id=2;

select \* from employee;

rollback transaction p2;

select \* from employee;

rollback transaction p1;

select \* from employee;

commit;

begin transaction;

select \* from employee;

save transaction s1;

select \* from employee;

update employee set salary=100 where employee\_id=2;

select \* from employee;

save transaction s2;

select \* from employee;

delete from employee where employee\_id=2;

select \* from employee;

rollback transaction s1;

select \* from employee;

--rollback transaction s2;

commit;

create database infosys;

use infosys;

create table employee (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(100),

department\_name VARCHAR(100),

salary int

);

insert into employee (employee\_id, employee\_name, department\_name, salary) values

(1, 'Monika', 'Testing', 1500000),

(2, 'Santosh', 'Development', 1700000),

(3, 'Rucha', 'HR', 1400000),

(4, 'Aboli', 'Testing', 1900000),

(5, 'Amol', 'Development', 2500000),

(6, 'Amruta', 'HR', 2200000),

(7, 'Sakshi', 'Testing', 2300000),

(8, 'Rutuja', 'Development', 2700000),

(9, 'Suraj', 'Development', 2900000),

(10, 'Rohit', 'HR', 1900000),

(11, 'Neha', 'Testing', 2000000),

(12, 'Nikhil', 'DevOps', 2100000),

(13, 'Monika', 'HR', 2400000),

(14, 'Manoj', 'Testing', 2500000),

(15, 'Gauri', 'DevOps', 2800000),

(16, 'Pankaj', 'HR', 2900000),

(17, 'Pravin', 'DevOps', 2300000),

(18, 'Pragati', 'Development', 1800000),

(19, 'Ravi', 'Development', 2300000),

(20, 'Rohan', 'DevOps', 2800000);

select \* from employee;

-- Find all employee who is working in Testing department

select \* from employee where department\_name='Testing';

-- find all employee who is working in HR and Development department --

select \* from employee where department\_name='HR' or department\_name='Development';

-- find all employee whose salary is 22,00,000

select \* from employee where salary= 2200000;

-- find all employee whose salary is below 20,00,000

select \* from employee where salary <2000000;

-- find all employee whose salary is equal or below 20,00,000

select \* from employee where salary <=2000000;

-- find all employee whose salary is above 20,00,000

select \* from employee where salary >200000;

-- find all employee whose salary is equal and above 20,00,000

select \* from employee where salary >= 200000;

-- find all employee who is salary between 25,00,000 to 30,00,000

select \* from employee where salary > 2500000 and salary<3000000;

select \* from employee where salary >= 2500000 and salary<= 3000000;

select \* from employee where salary between 2500000 and 3000000;

select count(\*) from employee where department\_name ='Testing' or department\_name='HR';

select count(department\_name) from employee group by department\_name;

select department\_name, count(department\_name) from employee group by department\_name;

select department\_name, sum(salary) from employee group by department\_name;

select department\_name, avg(salary), SUM(salary),COUNT(department\_name) from employee group by department\_name;

select department\_name,sum(salary) from employee group by department\_name having department\_name='HR';

select department\_name,sum(salary) from employee group by department\_name having department\_name='HR'

or department\_name='Testing';

select department\_name, max(salary) from employee group by department\_name;

select \* from employee;

begin transaction;

select \* from employee;

save transaction a1;

select \* from employee;

update employee set department\_name='Testing';

select \* from employee;

rollback transaction a1;

select \* from employee;

commit;

begin transaction;

select \* from employee;

save transaction p1;

select \* from employee;

update employee set salary=100 where employee\_id=2;

save transaction p2;

select \* from employee;

delete from employee where employee\_id=2;

select \* from employee;

rollback transaction p2;

select \* from employee;

rollback transaction p1;

select \* from employee;

commit;

begin transaction;

select \* from employee;

save transaction s1;

select \* from employee;

update employee set salary=100 where employee\_id=2;

select \* from employee;

save transaction s2;

select \* from employee;

delete from employee where employee\_id=2;

select \* from employee;

rollback transaction s1;

select \* from employee;

--rollback transaction s2;

commit;

begin transaction;

select \* from employee;

save transaction r1;

select \* from employee;

delete from employee where employee\_id=1;

select \* from employee;

update employee set salary=10 where employee\_id=2;

select \* from employee;

save transaction r2;

select \* from employee;

update employee set department\_name='Sales';

select \* from employee;

rollback transaction r1;

select \* from employee;

commit;

5) Data control language DCL --> this command is used by Database Admin

i) grant assign persmission/access to the users

syntax:

grant queryName table to username

allocate create permission to user (arati)

syntax:

grant create table to arati;

grant update table to arati;

grant delete table to arati;

ii) revoke remove the permission from users.

syntax:

revoke queryName table from username

e.g

revoke delete table from arati;

revoke alter table from arati;

tester Permission

i) insert

ii) update

iii) delete

iv) select

v) commit

vi) save transaction

vi) roll back

Denied permission for tester

1) Create

2) Alter

3) drop

4) truncate

5) grant

6) revoke

How many databases present in previous project

int between 30-40

how table in each and every database?

50-80 tables in each and every database

how many coumns in each and every table

30-80 columns in each and every table

do same database for SIT and UAT Enviroment?

No, always we have different database for SIT And UAT Enviroment

Which type of database u have used in previous company?

i) mysql

or

ii) sql server

or

ii) MS access

or

iv) Mongo DB

or

v) oracle 19g

or

vi) Db2

or

vi) aws db

or

vii) cloud db

etc..

|  |  |  |
| --- | --- | --- |
| Delete | Drop | Truncate |
| It is present inside the DML group | It is present inside the DDL Group | It is present inside the DDL Group |
| By using delete query we can delete rows as well as table records | By using drop query we can drop table as well as database | By using truncate query we can delete all records from table |
| It is slower than truncate and drop | It is faster than delete and truncate | It is slower than drop and faster than delete |
| Once we delete records by using delete query we can roll back using rollback query | Once we delete records using drop query we can not roll back. | Once we delete records using truncate query we can not roll back. |