SQL

[Structure Query language]

→ Sql is stands for structure Query language.

→ Sql is a language, which is used to Communicate with database in form of Queries

Data Base :-

Collection of data will be stored in the data base.

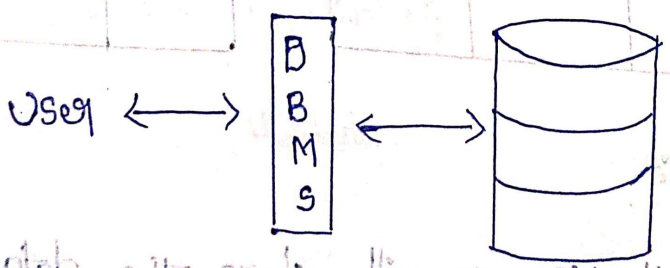
DBMS :-

→ DBMS stands for Database management system.

→ DBMS is a Software

→ DBMS is responsible for all the types of Operatic Such as inserting the data, deleting the data and modifying the data etc.

→ DBMS is a intermidiate between User and the database.



Types of DBMS :-

1) RDBMS
2) HDBMS
3) NDBMS

(1) RDBMS:-

Relational Database management System

(2) HDBMS:-

Hierarchial Database management System

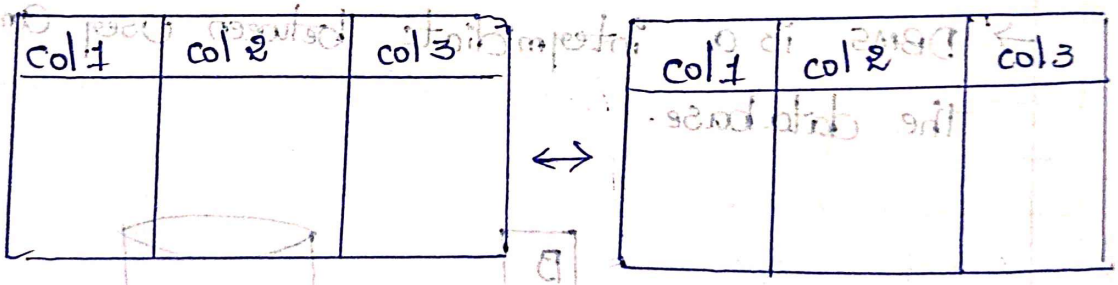
(3) NDBMS:-

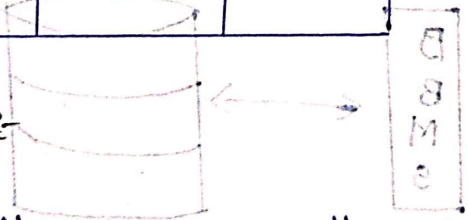
Network Database management System

→ RDBMS:-

In RDBMS, we will stored the data in table form and those table might have some relationship.

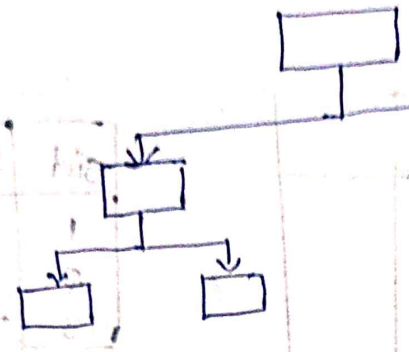
table-1 table-2





→ HDBMS:-

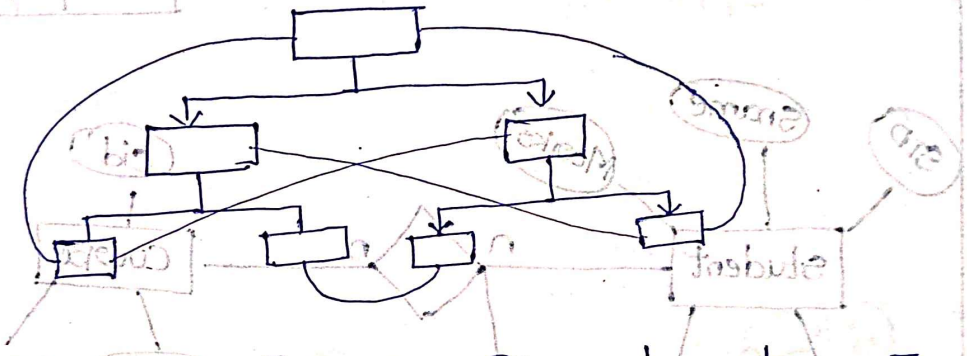
In HDBMS we will store the data in the form of tree structure.



→ Tree

→ NDBMS :-

In NDBMS we can store the datas in the form of tree structure. But all the datas are inter Connected with each other.



ER Diagram :- [Entity Relationship diagram]

→ Entity

→ Relation Ship

→ Atributes

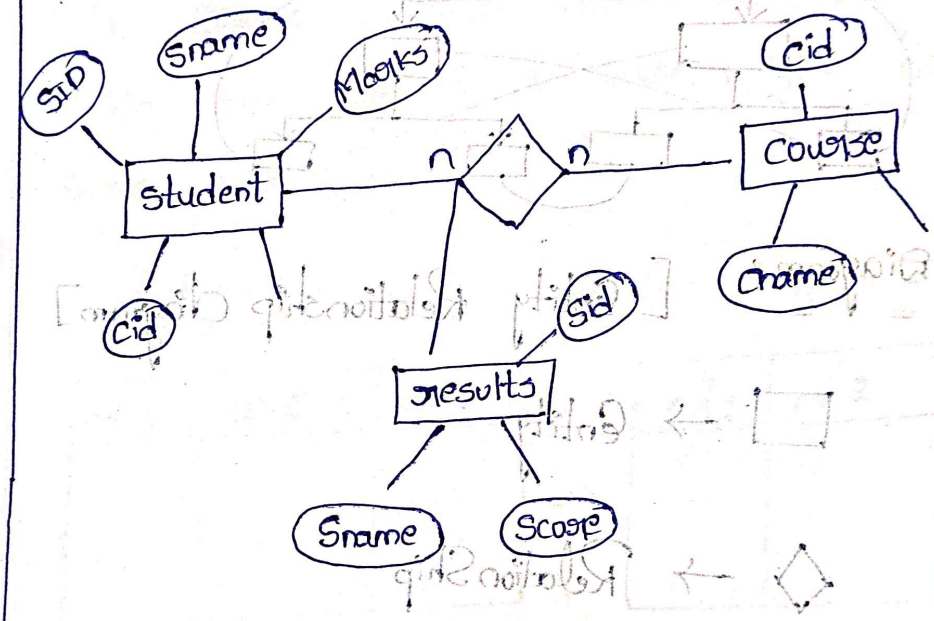
Sid	Sname	marks	cid	
101	Mouni	70	1	
102	Bha	80	2	
103	chinna	60	3	
104	Teja	90		
105	manu	50		

Results

Sid	Sname	
1	Mou	50
2	Bha	80
3	Tej	10

Course

cid	cname	
1	dava	F
2	oak	D
3	Web	B
4	python	D



Dept

Did

Dname

loc

Emp

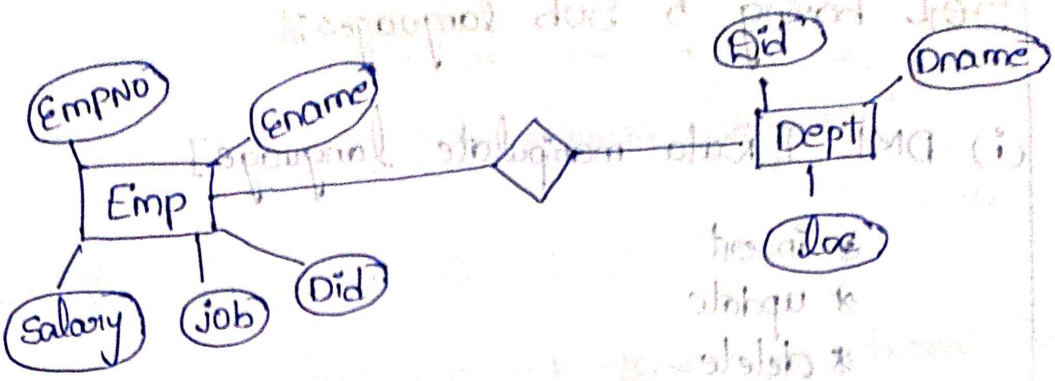
Empno

Ename

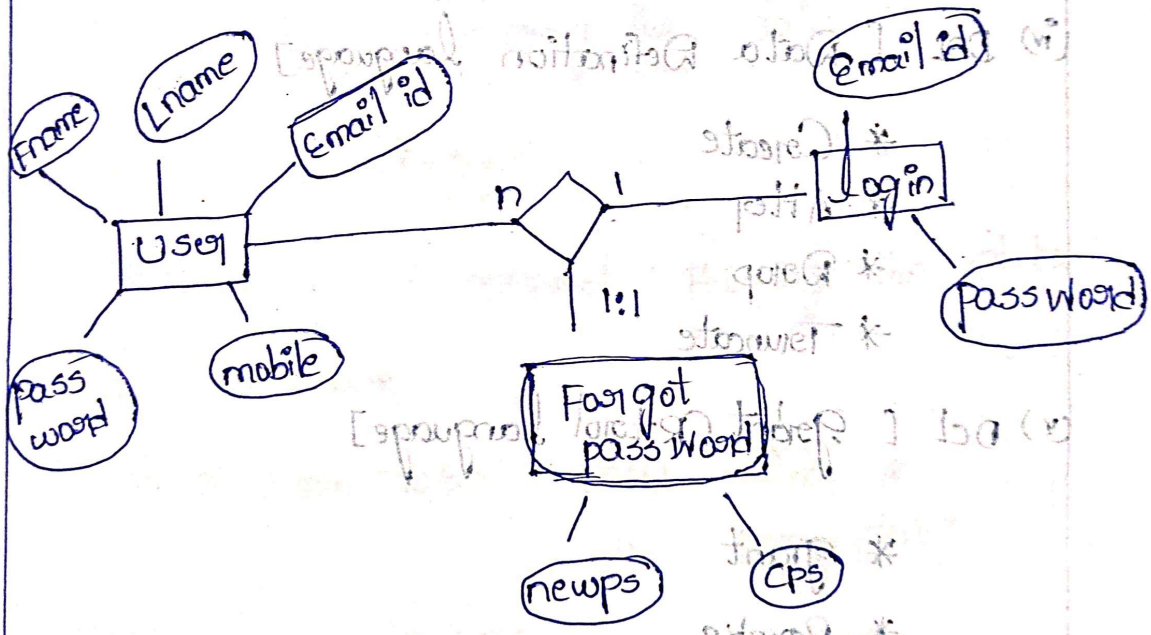
Salary

dob

Did



Usej	Login	Forgot password
Fname	Email id	email id
L name	Pass word	New ps
Email id	cps	
pass word		
mobile		



SQL having 5 Sub languages:

(i) DML: [Data manipulate language]

* insert
* update
* delete

(ii) DQL [Data Query language]

* Select
* From
* where

(iii) TCL [Transaction Control language]

* Savepoint
* Commit
* Rollback

(iv) DDL [Data Defination language]

* Create
* Alter
* Drop
* Truncate

(v) DCL [Data Control language]

* grant
* Revoke

☆☆ In SQL is not completely case sensitive where in SQL Records are case sensitive

1. DQL

[Data Query language]

→ The version of SQL is "21c"

→ The DQL Sub language there are three Commands

1) Select
2) From
3) Where

→ Select :- Fetching the columns

→ From :- From is used to Fetching the table name

→ Where :- where is used to provide Condition and Filetering the data.

imp

★ What is a Query :-

Query is a request Fecthing the data from database.

→ Select * From tab; (or) select * From tab;

TNAME	TABTYPE	CLUSTERID
DEPT	TABLE	
EMP	TABLE	
BONUS	TABLE	
SALGRADE	TABLE	

→ Select *
From dept;

DEPT NO	DNAME	LOC
10	ACCOUNTING	NEWYORK
20	RESERCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

→ Selet*
From Emp;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARIN	MANAGER	7698	28-SEP-81	1250	1400	30
7698	BLAKE	SALESMAN	7839	01-MAY-81	2850		30
7 82	CLARK	MANAGER	7839	09-JUNE-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500		30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

→ Select eid from emp;

Select eid from emp;

*

ERROR at line 1:

ORA: 00904: "EID": invalid identifier

→ Select empno from emp;

*

Empno

7369

7499

14 rows selected

→ write a Query to fetch all the details of the employees?

Set pages 20 lines 200;

Select *

from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

14 rows selected

→ Select empno from empl;

Select empno from empl

*

ERROR at line 1:

table or view does not exist

→ Write a query to fetch empname, salary and designation (Job) from emp table?

```
Select empname, salary, job
from emp;
```

O·P :-
```
Select empname, salary, job
          *
ERROR at line 1:
ORA-00904: "SALARY": invalid identifier.
```

→
```
Select ename, sal, job
from emp;
```

ENAME	SAL	JOB

14 rows selected

→ Write a query to fetch deptNo, and joining data of all the employees from emp table.

```
Select deptNo, hiredate
from emp;
```

O·P :-

DEPTNO	HIREDATE

14 rows selected

→ Write a Query to Fetch deptname, and location from dept table?

```
Select dname, loc
from dept;
```

DNAME	LOC
ACCOUNTING	NEWYORK
RESEARCH	DALLAS
SALES	CHICAGO
ORERATIONS	BOSTON

→ Write a Query to fetch all the details of the dept?

```
Select *
from dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEWYORK
20	RESERCH	DALLAS
30	SALES	CHICAGO
40	ORERATIONS	BOSTON

→ As Operator:

→ As Operator One type of keyword.

→ As is One of the "special operator"

→ Another name is "Aliasing"

→ Aliasing is used to give a temporary name for the columns

→ Aliasing will be done by the As keyword.

→ Select *
From emp

→ Select empno eno
From emp;

O.P :- Empno
14 rows selected

→ Select ename as ename
From emp;

NAME
14 rows selected

→ Select ename as DeptNO
From emp;

O.P :- DEPTNO
14 rows selected

→ Select ename as name, job as designation, sal as Salary
From emp;

→ Operation Operators :-

Arithmatic Operators :-

+, -, *, /

→ Select ename, Sal + 2000
From emp;

→ Write a query to company decided to give diwali bonus of 2000/- for every employee display the name and salary?

```
Select ename, sal + 2000
from emp;
```

O.P:

ENAME	SAL+2000
SMITH	2800

14 rows selected

→ Write a query to fetch annual salary of the employees select sal * 12?

```
Select sal * 12
from emp;
```

SAL*12
9600

14 rows selected

→ Write a query to fetch display company decided to give a hike of 10 percent for every employee display their name and salary?

```
Select ename, sal + (sal * 10/100)
from emp;
```

ENAME	SAL + (SAL * 10/100)

Select sal * 1.1 from emp;
Select sal * 0.1 from emp;

↳ Optional

→ Write a query to display company decided to donate 500/- from every employee display sal after the deduction and display the ename and job?

```
Select ename, sal-500, job
from emp;
```

O.P: 14 rows selected

→ Select empno + 20 from emp;

```
EMPNO +20
---------
14 rows selected
```

→ Charecter (or) concodination operator:

To join the records

Litrus:

Litrus nothing but a number, string, charecter, date formatt, boolean is called as litrus. →

[based on their language]

→ Data types:

1) Number
2) Varchar
3) date formate

Number :- Convert to decimal into integer

Ex :- 10.3 (10-3=7)

→ 10po737.222 (any number)

Ex :- 7.2 (7-2=5)

→ 10000.22

Number → int

Number (4,2)

Varchar :- In other language is string. which you want to be string abo, character abo, we will use single quote only.

Ex :- 'Hello' varchar (10)

'H' varchar (20)

Date-formate :-

→ very important to "SQL"

Ex :- '01-JAN-22' (SQL Date-formate)

Name	Null ?	type.
DEPT NO	Not Null	NUMBER (2)
D NAME		VARCHR2 (14)
LOC		VARCHR2 (13)

→ desc emp;

Name	Null ?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME		VARCHAR2 (10)
JOB		VARCHAR2 (9)
MGR		
HIREDATE		Number (4)
SAL		
COMM		
DEPTNO		

→ Select * from bonus;

no rows selected

Charecter Operator :- (||)

Concoplination - Joining or merging the data

→ set pages 200 lines 200;

→ Select 's' || 'ql'
from emp;

O·p :- s
sql
14 rows selected

→ select 's' || 'ql'
from dept;

s
sql
4 rows selected

→ Select *
From dual
[dual is One of the dummy table)

O·P

D
X

→ Select 's' || 'a' || 'b'
from dual;

's'
sab

imp * **Dual** :-

Dual is an dummy table it is used to work with an literus.

→ select 123||45
from dual;

O·P :-

123	
12345

→ Select 'R' || 'J'
from dual;

O·P :-

R'
RJ

→ Select allb
from dual;
O·P :- Invalid identifier

→ select '12' || 34
from dual;

O·P :-

12
1234

→ Select ename || job from emp;

O·P :-

ENAME || JOB
SMITH CLERK

14 rows selected

→ Select ename || ' ' || job total from emp;

O·P:- TOTAL
SMITH CLERK
14 rows selected

→ Write a Query to fetch first name and last name do it as a single column complete name from xyz table?

Select First-name || ' ' || Last-Name as completeName
from xyz;

→ write a Query to display as follows Smith working as clerk?

Select ename || 'Working as a' || job
from emp

other way:-

Select ename || ' ' || 'working as a' || ' ' || job
from emp;

→ Write a Query to display as follows Smith working in a dept 20 and getting a salary of 800?

Select ename || ' working in a dept ' || deptno ||
' and getting a salary of ' || sal
from emp;

→ Write a Query to display as follows smith employee id is 7369 and he join the company 17-Dec-80 and getting a sal of 800?

Select ename || ' emp id is ' || empno || ' and join in the company ' || hire date || ' and getting salary of ' || sal

from emp;

→ write a Query to fetch as follows smith working as a clerk use proper aliasing?

Select ename as name, 'working as a ' || job as desi from emp;

→ write a Query to display as follow smith working in a dept 20 and getting a salary of 800 do proper aliasing?

Select ename as name, 'working in a dept ' || dept no as did, ' and getting a salary ' || sal as sala from emp;

Comparision Operator :-

→ This operator is used to check the particular condition

→ In this >, <, >=, <=, =, != (or) <> (or) ^=

→ This Operator is used to select one 'where' clause

→ where clause is providing Condition and filter the records.

Syntax :- Select ....
from table name
where cond;

→ Select *
From emp
where ename = 'SMITH';

→ Select *
From emp
where ename = 'ALLEN';

→ Select *
From emp
where job = 'SALSEMAN';

→ Select *
From Emp
where job = 'CLERK';

→ Select ENAME
from emp
where JOB = 'SALSEMAN';

→ Select JOB, sal
from emp
where ename = 'ALLEN';

① Write a Query to fetch name, salary, dept no of all the clerks?

```
Select ename, sal, deptno
from emp
where JOB = 'CLERK';
```

② Write a Query to fetch all the details of employee whose salary is greater than 2000?

```
Select *
from emp
where sal > 2000;
```

③ write a Query to fetch name, salary, dept No of all the clerks?

```
Select ename, sal, deptNo
from emp
where job = 'CLERK';
```

④ write a Query to display name of the employee whose salary is >=3000?

```
Select ename
from emp
where sal >=3000;
```

(5) Write a query to fetch all the details of the employees except manager?

```
Select *
From emp
where job != 'MANAGER';
```

(6) Write a Query to fetch all the details of the employee who are working in department 20?

```
Select *
From emp
where deptno = 20;
```

* Logical Operators:-

(1) OR → Fetch same column
(2) AND → Fetch different column.
(3) NOT → Equal column we can use 'and'

→
```
Select *
from emp
where job = 'SALSEMAN' AND job = 'MANAGER';
```

O.P:- No rows selected

→
```
Select *
From emp
where job = 'SALESEMAN' or job = 'MANAGER';
```

O.P:- 3 rows selected

→ Select *
From emp
where sal >= 1000 and sal <= 3000;

O.P:- 9 rows selected

→ Select *
From emp
where sal >= 1000 or sal <= 3000;

O.P:- 14 rows selected

→ Select *
From emp
where ename = 'ALLEN' and sal >= 1000 and sal <= 3000;

O.P:- 1 row selected

→ Write a query to fetch all the details of the employee who salary is >500, <2500.

Select *
From emp
where sal >= 500 and sal <= 2500;

→ write a Query to fetch all the details of the salseman, manager, clerk?

Select *
From emp
where job = 'SALESEMAN' or job = 'MANAGER' or job = 'CLERK';

O.P:- 7 rows selected

→ Write a query to fetch all the details of emplo the department 10 and 20 working as salse man and manager?

```
Select *
from emp
where (DeptNO =10 or DeptNO =20) and (Job = 'SALS
        MAN' or job = 'MANAGER');
```

O·P: 2 rows selected

→ Write a query to fetch all the details of smith, allen, jones and blake, and working in a department 30 and 10 and getting a salary

```
Select *
From emp
where (ename = 'SMITH' or ename = "ALLEN' or
        ename = 'BBACK') and (DEPTNO =30 or
        DeptNO = 10) and (sal > 1000 and sal <3000);
```

O·P: 2 rows selected

→ Write a query all the details of manager, clerk, salseman working in a dept 10, 20, 30.

```
Select *
From emp
where (job = 'MANAGER' or job = 'CLERK' or
        job = 'SALSEMAN') and (DeptNO = 10 or
        deptNO = 20 or deptNO = 30);
```

O·P :- 7 rows selected

→ Write a Query to fetch all the details of the employee in deptNO 10 & 30

```
Select *
From emp
where job(deptNO = 10 or deptNO = 30);
```

→ write a Query to fetch all the details of the employe except manager in dept 10 & 30

```
Select *
From emp
where (job != 'MANAGER') and (DeptNO = 10 or DeptNO != 30);
```

O·P :- 7 rows selected

→
```
Select *
From emp
where (deptNO = 30) and job = 'MANAGER';
```

O·P :- 5 rows selected

→
```
Select *
from emp
where Not job = 'MANAGER';
```

O·P :- 11 rows selected.

In Operator :

→ In operator is used to fetch multiple records from a same column.

→ In operator same work as an 'or' Operator.

→ Select *
From emp
where ename in ('SMITH', 'ALLEN', 'WARD');

(or)

select *
From emp
where ename = SMITH or ename='ALLEN' or ename='WARD';

O.P :- 3 rows selected

→ Select *
From emp
where not deptNO in(10,20);

(or)

select *
From emp
where deptNo not in (10,20);

(or)

select *
From emp
where deptNO != 10 and deptNO != 20;

O.P :- 6 rows selected.

→ Write a query to fetch all the debits of the salseman, manager, analist and working in a depart ment 10 and 30 and getting a salary more than 1000 less than 3500;

```
Select *
From emp
where job in ('SALESEMAN', 'MANAGER', 'ANALIST')
      and deptno in (10,30) and SAL>1000
      and SAL<3500;
```

O.P:- 6 rows selected

→ Write a query to fetch all the details of the department accounting, salse, research in location newyork and chicago?

```
Select *
From dept
where dname in ('ACCOUNTING', 'RESEARH', 'SALSES')
      and Loc in ('NEWYORK', 'CHICAGO');
```

O.P:- 2 rows selected.

Like Operator:

→ like Operator is used to when we have partial information

→ like operator will work for a varchar (string)

→ like operator is used for to find out missing characts.

→ we have a two types of operators:

(1) % = percentage is used for many characters

(2) _ = underscore is used for single character.

→ Select ename
from emp
where ename like 'S%';

→ Select *
from emp
where ename like '%T';

→ Select *
from emp
where empNo like '7%8';

→ Select *
From emp
where ename like 'A-L%';

→ Write a Query to fetch name of the employee whos 6th character is i?

```
Select ename
from emp
where ename like '____i%';
```

→ Write a query to fetch all the debails of the employee who joined in the month of Dec.

```
Select *
From emp
where hiredate like '%DEB%';
```

o.p :- 3 rows selected.

→ Write a Query to fetch all the details of the employee whose reporting manager debails ending with an a character 8 ?

```
Select *
From emp
where mgr like '%8';
```

→ Write a Query to fetch deptName 6th character is N ending with an 'G'.

```
Select dname
from dept
where dname like '_____N%G';
```

→ Write a Query to fetch DeptName it is starting with an a character 'O' 8th character is O

```
Select dname
from dept
where dname like 'O______O%';
```

→ Write a Query to fetch Name of the employe whose name last but one character is 'm' and starting with an 'a' character?

```
Select ename
from emp
where ename like 'a%.m-';
```

→ Write a Query to fetch location which stating with an a N character 4 character is 'y' ending with an a 'k'?

```
Select loc
from dept
where loc like 'N__y%.k';
```

→ **Between Operator:-**

Between Operator for finding the range between one value to another value.

Syn:- Select ....
from tablename
where columnname low between low range and high range;

→ Select *
From emp
where sal between 3000 and 1000;

→ Select ename
from emp
where ename between 'A' and 'FORD';

→ Select ename
from emp
where ename between 'A' and 'FORD';

→ Select ename
from emp
where ename between 'ADAMS' and 'FORD';

→ Select ename
from emp
where ename between 'ALLEN' and 'FORD';

→ Select Hiredate
from emp
where Hiredate between '17-DEC-80' and '23-JAN-8

→ Write a Query to fetch all the details of the manager and the salseman who are working in dept 10 & 30 and getting salary between the range of 1000-4000 and their reporting manager number ending with an alive

Select *
From emp
where job in ('MANAGER', 'SALSEMAN') and dept in (10,30) and Sal between 1000 and 4000 and mgr like '%9';

→ Write a Query to fetch all the details of the employee they join in the company between the range of 01-JAN-81 to 09-JAN-83

```
Select *
from emp
where hiredate between '01-JAN-81' and '09-JAN-83';
```

is Operator :-

"Is" Operator is used for fetching the null value.

→ Write a Query to fetch all the details of the employee who are not getting any commition

```
Select *
from emp
where comm is null;
```

→ Write a Query to fetch all the details of the employe who don't have reporting manager?

```
Select *
from emp
where mgr is null;
```

DML [Data Manipulating language]

Syntax:-

1) insert into tablename
   values (val1, val2 .....);
2) insert into tablename (col1, col2...)
   values (val1, val2 ....);

→ insert into emp
values (101, 'abc', 'clerk', 1001, '26-oct-22', 50000,
1000, 20);

O.P:- 1 row Created

→ Select *
from emp;

1 row Created

→ insert into emp
values (101, 'abc', 'clerk', 1001, '26-oct-22', 50000,
1000, 20);

Error at line 1
ORA-00001 : Unique Constraint (SCOTT.PK-Emp violated)

→ insert into emp
values (10003, 'xyz', 'clerk', 1000, '26-oct-22',
50000, 1000, 20);

Error at line 2:
value larger than specified precision allowed for this column.

→ insert into emp
values (1003, 'xyz', 'CLERK', 102, '28/Jan/22',
40000, 1000, 20);

1 row created

→ Select * From emp;

→ insert into emp
values (1005, 'MNO', 'SALSEMAN', 103, 26-Oct-2022, 60000,
5000, 30);

1 row created

→ insert into into emp
values (1006, 'MNO', 'SALSEMAN', 104, '26-Oct-2022',
60,000, 40);

Error at line 2:
value larger than specified precision allowed
for this column.

→ insert into emp
values (1006, 'MNO', 'SALSEMAN', 104, '26-Oct-2022',
70000.3909032988, 4000, 40);

1 row created.

→ insert into emp
values (1008, 'yyy', 'SALSEMAN', 104, '26-Oct-2022',
70000.3909032988, 4000, 70);

Error at line 1:
integrity constraint (SCOTT.FK_DEPTNO) violated
parent key not fo

→ insert into dept
values (70, 'dev', 'INDIA');

1 row Created

→ insert into emp
values (10008, 'yyy', 'SALESEMAN', 104, '26-oct-2022',
70000.3909032988, 4000, 70);

o/p 1 row Created

→ insert into emp
values (1009, 'yyy', 'SALSEMAN', 104, '26-oct-2022',
70000.3909032988, 4000);

Error at line 1:
not enough values.

→ insert into emp
values (1009, 'yyy', 'SALSEMAN', 104, '26-oct-2022',
70000.3909032988, 4000, null);

o/p :- 1 row Created

→ Select * From emp;

→ insert into emp
values (null, 'yyy', 'SALSEMAN', 104, 26-oct-2022, 70000
390903298 88, 4000, null);

Error at line 2:
Cannot insert null into ("SCOTT"."emp"."empno")

→ insert into dept
values (null, 'aaa', 'india');

Error at line 2
Cannot insert null into ("SCOTT". "DEPT". "DEPTNO")

→ insert into dept
values (70, 'aaa', 'india');

Error at line 1
Unique Constraint (SCOTT.PK-DEPT) violated.

→ insert into emp (empno, ename, deptno)
values (100, 'a', 40);

O.P :- row Created

→ select * From emp

→ insert into emp (empno)
values (1010);

O.P :- row Created

→ insert into emp (ename)
values (b);

Error at line 1
cannot insert in null into ("SCOTT")

Update:

Update is used to update and modify the values in a table.

Syn:- Update tableName set $col_1 = val_1, col_2 = val_2 \ldots$
where condition;

sid	sname	age
10	Ram	22
20	sitha	23
30	Janaki	24

→ Update the sitha age is 24

```
update stud set age =24
where sname = 'SITHA';
```

→ write a Query to update the salary of SCOTT to 40000.

```
update emp set sal =40,000
where ename = 'SCOTT';
```

→ Write a Query to update the designation of Smith to accountant and hiredate to 01-sep-202

```
update emp set job= 'ACCOUNTANT', hiredate =
                                '01-sep-2021'.
where ename = 'SMITH';
```

→ write a Query to update the dept name to hr of dept no 40.

```
Update dept set dname = 'HR'
where deptNO = 40;
```

→ update the employee number of martin to 7676

```
update emp Set empNO = 7676
where ename = 'MARTIN';
```

1 Delete :-

Delete command is used to delete the partical or record from table.

Syn:- delete from tablename
where condition;

→ Write a Query to fetch delete Smith, Scott, and allen data from the data?

```
delete from emp
where ename in ('SMITH', 'SCOTT', 'ALLEN');
```

→ write a Query to delete all the salseman in the table?

```
delete from emp
where job = 'SALSEMAN';
```

→ Write a Query to delete all the employees whose salary is 3000?

```
delete from emp
where ~~ename~~ SAL = 3000;
```

→ Write a Query to delete all the department information whose dept name is research?

```
delete from dept
where dname = 'RESERCH';
```

→ write a Query to delete all the student information whose student id is 10 and 20?

```
delete from stud
where sid in (10,20);
```

→ write a Query to delete all the course details of SQL?

```
delete from course
where cname = 'SQL';
```

Course:

Cid	Cname	Trainer
1	SQL	JYOTHI
2	PYTHON	JANTHOSH
3	JAVA	RAVEESH

☆ for updating and deleting values where condition is mandatory.

→ TCL [Transaction Control language]

- → Commit
- → Roll back
- → save point

Commit :- Save the transaction permanantly we can use commit.

Save point :- Save point is used to save the the transaction for temporary

Syn :-

```
Save point save point name;
```

Roll back :-

```
Roll back;
Roll back to Savepoint;
```

→ Roll back is used to delete all the transaction till Commit

→ **Roll back to Savepoint :-** It is used to delete the transaction till specific save point

★ Savepoint & roll back → Temporary

★ Commit is permanant

→ Select * from bonus;

no rows selected

→ insert into bonus
values ('A', 'SALSEMAN', 5000, 1000);

1 row Created

→ insert into bonus
values ('A', 'SALSEMAN', 5000, 1000);

1 row Created

→ insert into bonus
values ('A', 'SALSEMAN', 5000, 1000);

1 row created

→ insert into bonus
values ('A', 'SALSEMAN', 5000, 1000);

1 row Created

→ Save point a;

Save point Created

→ Select * from bonus;

4 rows Created

→ insert into bonus
values ('B', 'CLERK', 40000, 1000);

1 row Created

→ insert into bonus
values ('B', 'CLERK', 40000, 1000);

1 row Created.

→ insert into bonus
values ('b', 'CLERK', 4000, 1000);

1 row Created

→ insert into bonus
values ('b', 'CLERK', 4000, 1000);

1 row Created

→ insert into bonus
values ('b', 'CLERK', 4000, 1000);

1 row Created.

→ Save point b;

Savepoint Created

→ Select * from bonus;

9 rows selected

→ insert into bonus
values ('c', 'MANAGER', 70000, 30000);

1 row Created

→ insert into bonus
values ('c', 'MANAGER', 70000, 30000);

1 row Created

→ insert into bonus
values ('c', 'MANAGER', 70000, 30000);

1 row Created

→ Select * From bonus

12 rows Created

→ Save point c;
Save point Created

→ insert into bonus (ename)
values ('d');
1 row Created

→ insert into bonus (ename)
values ('d');
1 row Created

→ insert into bonus (ename)
values ('d');
1 row Created

→ Select * from bonus;
15 rows selected

→ roll back to c;
12 rows selected roll back completed.

→ Select * from bonus;
12 rows selected

→ roll back to b;
roll back completed.

→ Select * from bonus;
9 rows selected

→ Roll back;
roll back Completed.

→ select * from bonus;

no rows selected

→ insert into bonus
values ('c', 'MANAGER', 70000, 30,000);

1 row Created

→ insert into bonus
values ('c', 'MANAGER', 70000, 30,000);

1 row Created

→ insert into bonus
values ('c', 'MANAGER', 70000, 30,000);

1 row created

→ Commit;

Commit Completed

→ select * from bonus;

2 rows selected

→ roll back;

roll back Completed

→ select * from bonus;

2 rows Created

→ insert into bonus
values ('b', 'clerk', 40000, 1000);

1 row Created.

→ insert into bonus
values ('b', 'clerk', 40000, 1000);
1 row created

→ Select * from bonus;
5 rows created

→ save point a;
save point created

→ Roll back to a;
Error (because we are not save point 'a')

→ DDL [Data defination language]

(1) Create
(2) Alter → add, drop, rename, modify
(3) Drop
(4) Truncate

→ Type of Constraints :-

(1) not null
(2) Default
(3) check
(4) primary key
(5) unique
(6) foreign key

It is used to provide the rules for the dates

Not Null :-

→ it will accept not null values.
→ it will accept and allow duplicate values
→ it will never accept null values
→ In a one table one or more column can be declare as a not null

Default :-

→ it is a used to set the default value if no values is specified
→ In a one table one or more column can be declared as default.

Check :-

→ check is used to provide the condition.
→ In a table one or more column can be declared as a 'check'.

Unique :-

→ It is will accept unique values
→ It will never accept duplicate values.
→ It will accept null values
→ In a table one or more column can be declared as a "unique"

primary :-

→ It is a combination of Uniaue and not null.

→ It is a column whos data declare the reco rds uniquely

→ In a table one column can be declared as as a primary key

foriegn key :-

→ It is a column in a table, whose data refered the data of another table uniquely.

→ It will expect duplicate values and null values

→ we cannot insert any data which is not present refere table

→ In a table one or more columns can be declare as a foriegn key.

(1) Create :-

Syn :- Create table tablename (

Create table workkey (

wid number (10),

wname varchar (20),

job varchar (10),

sal number (7,2));

Table created.

desc worker;

NAME	Null	Type
wid		number (10)
wname		varchar (20)
job		varchar (20)
sal		Number (7,2)

→ insert into worker
values (10001, 'abc', 'clerk', 4000);
1 row created

→ insert into worker
Vales (1001, 'abc', 'clerk', 4000);
1 row created

→ insert into worker
values (10001, 'abc', 'clerk', 4000);
1 row created

→ Select * From worker;

WID	WNAME	JOB	SAL
10001	abc	clerk	4000
10001	abc	clerk	4000
10001	abc	clerk	4000

→ Create table course(
cid number(10) primary key,
cname varchar(20) not null,
tri varchar(20));

Table Created.

Name	Null	Type
CID	NotNull	Number(10)
CNAME	NotNull	varchar(20)
TRI		varchar(20)

→ Insert into course
values (1, 'java', 'a');

1 row created

→ insert into course
values (1, 'java', 'a');

Error at line 1:
ORA-00001: unique constraint (SCOTT, SYS_C0005468 voolated).

→ insert into course
values (2, 'sql', 'b');

1 row created

→ insert into course(cname)
values ('python');

Error at line 1:
ORA - 00947 : not enough values.

→ insert into course (cid)
values (3);

Error at line 1:
ORA - 01400: cannot insert null into ('scott'.'course'.CNAME)

→ insert into course (cid, cname)
values (3, 'java');

1 row created.

→ Create a table of jspiders consist of jid, name, age (>=20), gender (male and female) country colum (default should be india)

```
Create table jspiders (
jid      number(10)   primary key,
dname    varchar(20)  notnull,
age      number(2)    check age (a>=20),
gender   varchar(10)  check (gender in ('m','f'));
country  varchar(10)  Default ('India');
```

→ insert into jspiders
values (101, 'a', 18, 'm', 'usa');

Error at line -1
check constraint (scott.sys - c0056829) violated

→ insert into jspiders
values (101, 'a', 20, 'm', 'usa');

1 row Created

→ Select * from jspiders;

JID	JNAME	AGE	GENDER	COUNTRY
101	a	20	m	usa

→ insert into jspiders
values (102, 'b', 22, 'F', 'chinna');

Error at
check constraint scott.sys-c0056630) violated

→ insert into jspiders
values (102, 'b', 22, 'm', 'chinna');

1 row Created

→ insert into jspiders (jid, jname, age)
values (103, 'c', 23);

1 row Created

→ Select * from jspiders;

JID	JNAME	AGE	GENDER	COUNTRY
101	a	20	m	USA
102	b	22	m	chinna
103	c	23		India

→ insert into jopidey
values (105, 'd', 24, 'f', null);

1 row Created.

→ Select * from jopidey;

JID	JNAME	AGE	GENDER	COUNTRY
101	a	20	m	USA
102	b	22	m	china
103	c	23		india
105	D	24	f	

→ Create a table of product in that product id, Product name, price?

Create table product (
pid number(5) Unique,
Pname varchar(20),
price number(4,2));

Table Created.

→ desc product;

NAME	NULL	Type
PID		number(5)
PNAME		varchar(20)
PRICE		number(4,2)

→ insert into product
values (101, 'pen', 20);

Error at line : 1
Unique Constraint

→ insert into product (pname)
values ('book');

1 row Created

→ select * from product;

PID	PNAME	PRICE
101	pen	20
	book	

→ insert into product (pname)
value ('bag');

1 row Created

→ Select *, from product;

PID	PNAME	PRICE
101	pen	20
	book	
	bag	

→ Create table product (
pid number (5) Unique,
pname varchar (10),
price number (4,2));

Create table product (
*
Error : name already used by an existing Object

→ Create table p (
pid number(5) Unique,
Pname varchar (10)
price number (4,2));
Create a table.

→ insert into p
values (01, 'pen', 20);
1 row Created.

→ insert into p(Pname)
values ('bag');
insert into p(name)
*
Error line 1
cannot insert Null into 'scott'."P"."PID")

→ Select * from course;

Name	Type
CID	
CNAME	
TRI	

→ Create table student (
sid number(5) primary key,
Sname varchar(20),
marks number(4,2),
cid number(5) references Course(cid));
Table Created.

→ insert into student
values (101, 'a', 70, 1);

Error at line 1:
integrity Constraint (parent key is not found)

→ insert into student (sid, sname, marks)
values (101, 'a', 70);

1 row Created

→ insert into Course
values (1, 'sql', 'z');

1 row Created

→ Select * from course;

CID	CNAME	TRI
1	sql	z

→ insert into student
values (102, 'b', 80, 1);

1 row Created

→ insert into student
values (103, 'b', 50, 1);

1 row Created

→ Select * from student;

SID	SNAME	MARKS	CID
101	a	70	
102	b	80	1
103	b	50	1

→ Select * from emp;

14 rows selected

Alter :- → add
→ drop
→ Rename
→ modify

Add :- [add column]

Syn:- Alter table tablename
add columnName datatype (size) constraint;

→ Alter table emp
add gender varchar (1) check (gender in ('M', 'F'));

→ Select * From emp;

→ alter table emp
add age number (2) not null;

Error at line 1
table must be empty to add mandatory
(Not null) column.

→ alter table emp
add age number(2) primary key;

Error at line 2
table can be have Only One primary key

* Select * from tab;

TNAME	TABTYPE	CLUSTERID
DEPT	TABLE	
BONUS	TABLE	
SALGRADE	TABLE	
STU	TABLE	
.....	.....	

→ desc course;

NAME
CID
CNAME
TRI

→ alter table course
add age number(3) not null;
table Created

→ desc course;

NAME
CID
CNAME
TRI
AGE

→ alter table course
add salary number(7,2);
table created

→ desc Course;

NAME
CID
CNAME
TRI
AGE

→ alted table course
add salary number (7,2);
table created

→ desc course;

NAME
CID
CNAME
TRI
AGE
SALARY

Drop :- [delete column]

Syn :- alter table tablename
drop column columnName;

→ alter table emp
drop column gender;
Table altered.

→ alter table course
drop column cid;
Error
Cannot drop "parent" key column.

→ Select * From tab;

TNAME	TABTYPE	CLUSTERED
DEPT	TABLE	
BONUS	TABLE	
⋮	⋮	

→ alter table Course
drop column cid;

Error at line 2
cannot drop "parent" key column

→ alter table student
drop column cid;
Table altered

→ alter table student
drop column sid;
Table altered

→ alter table course,
drop column age;
Table alted

→ alter table course
drop column salary;
table altered

→ desc course;

NAME
CNAME
TRI

★ Rename :- [Renaming the column]

Syn :- alter table tablename
rename column oldcol to newcol;

→ alter table emp
rename column empno to eid;
Table is altered

→ Alter table emp
rename column ename to Name;
Table is altered

→ alter table emp
rename column deptno to did;
Table as altered.

★ modify :- [changing the data]

→ if the records already present it will through error. if we try to change one data type to another data type.

→ Only we can increase and decrese the size if records are already present.

→ data must be empty to change one data type to another data type.