21/6/22

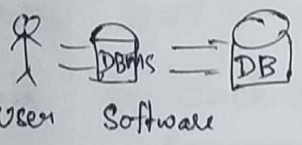
Oracle SQL

Databases:

Oracle DB
My SQL
IBM DB2
MongoDB (cloud)
Post grey
SQL Server
⋮

SQL - Structured Query language

- SQL stand for Structured query language
- SQL is used to communicate with database in the form of queries.
- Database is nothing but storage location



DBMS - Database Management System

→ DBMs is a (software means collection of programs)

→ DBMs is responsible for all type of operations such as inserting the data, fetching, updating and deleting the data.

→ DBMs is intermediate b/w programmer & D.B.

Types of DBMS:

→ Relational DBMS (Oracle SQL)

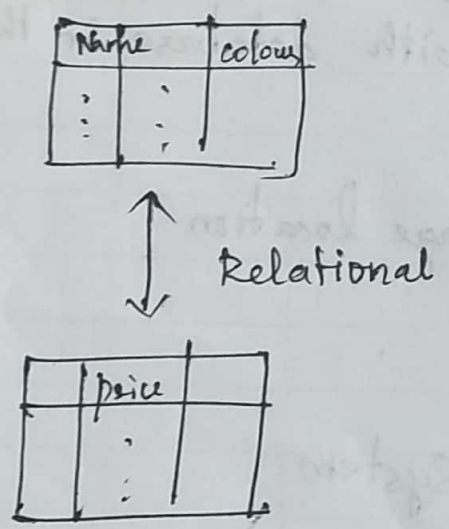
→ Object Oriented DBMS

→ Hierarchial DBMS

→ Network DBMS

Relational DBMS:

→ In this DBMS we store data in the format of Tables.



Rows — Records / Tuples

columns - Fields / Attributes

Objected Oriented DBMS:

JSON – { shoe name: ... ; price: ... ; }

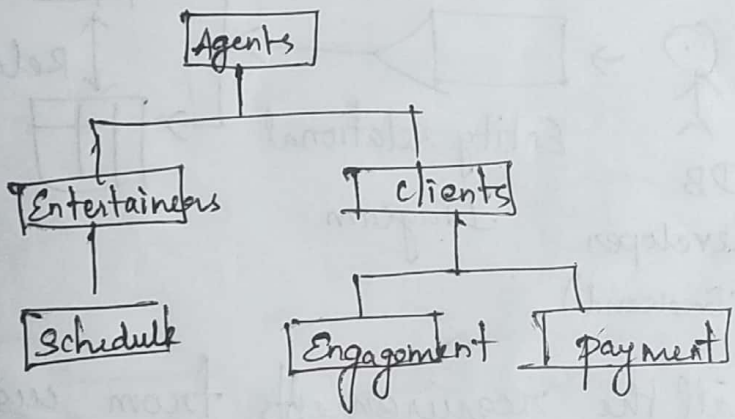
- In this DBMS we store data in object format.

JSON – Javascript Object Notation

```
shoe – {
        name : ----;
        price : ---;
        :
        }
```

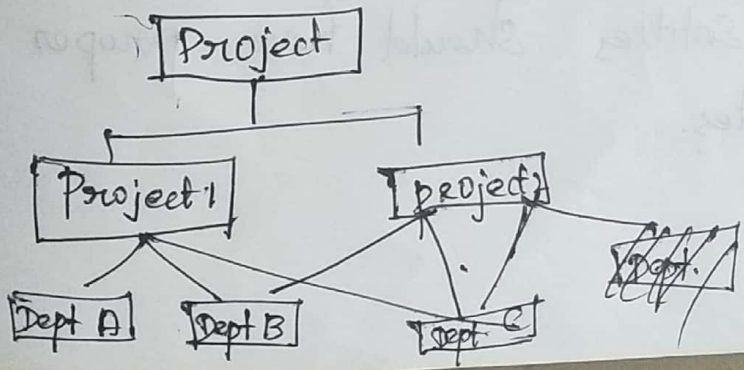
Hierarchical DBMS:

- In this DBMS we store data in tree structure format.



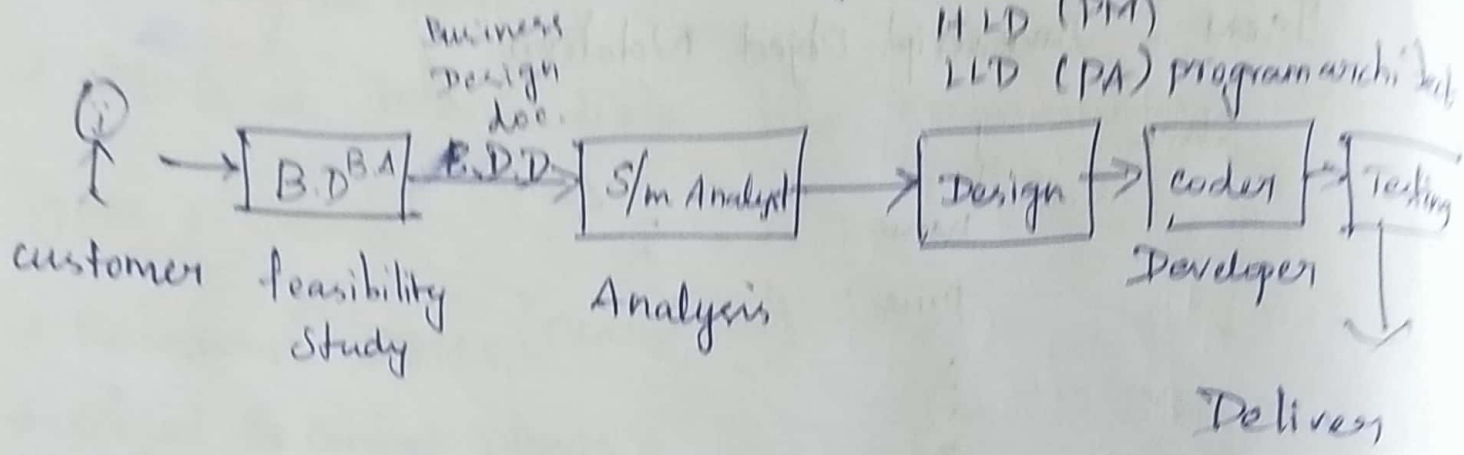
Network DBMS:

- In this DBMS, we store data in the format of tree structure but all the data will be interconnected to each other data.

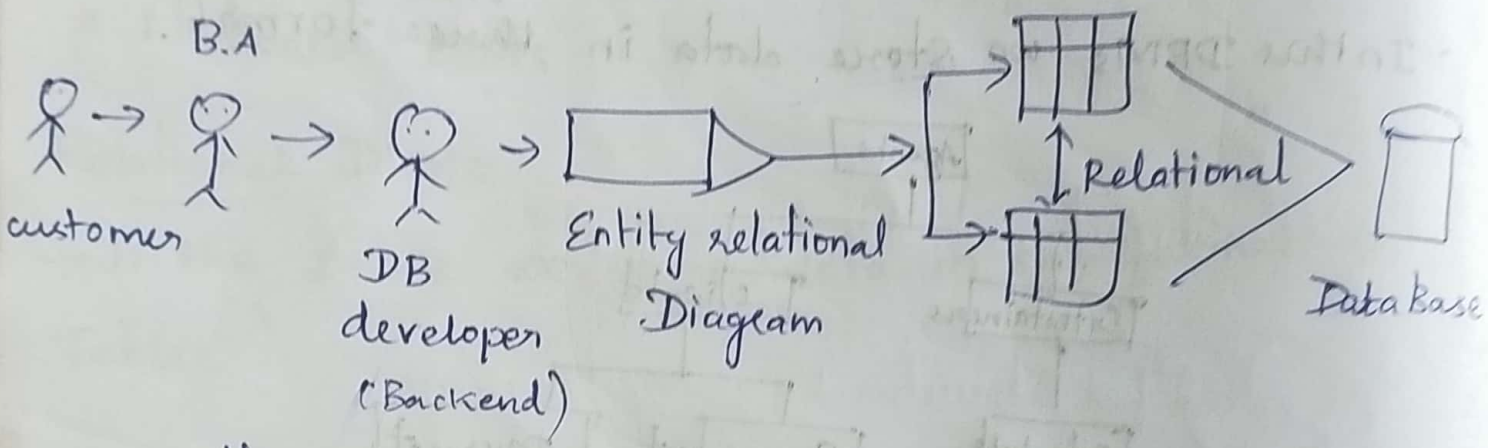


22/6

Software development life cycle



Database Model



→ collect all the requirements from customers.

→ while collecting the requirements there should be proper entities.

→ Entities (objects) can be living (or) Non-living things.

→ Each Entities should have proper attributes.

(objects) To represent

Entity → Rectangle

Attribute → oval

Relational → Rhombus

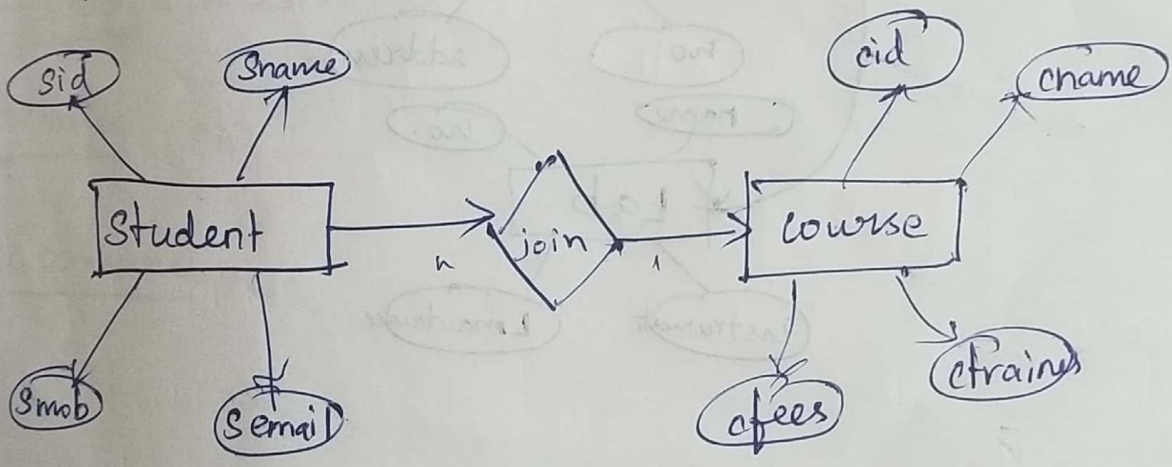
ER diagram: (entity)

Student (living)

- sid
- sname
- smobileno
- semail
- saddress
- ⋮

course (non_living)

- cid
- cname
- cduration
- cfees
- ctrainer

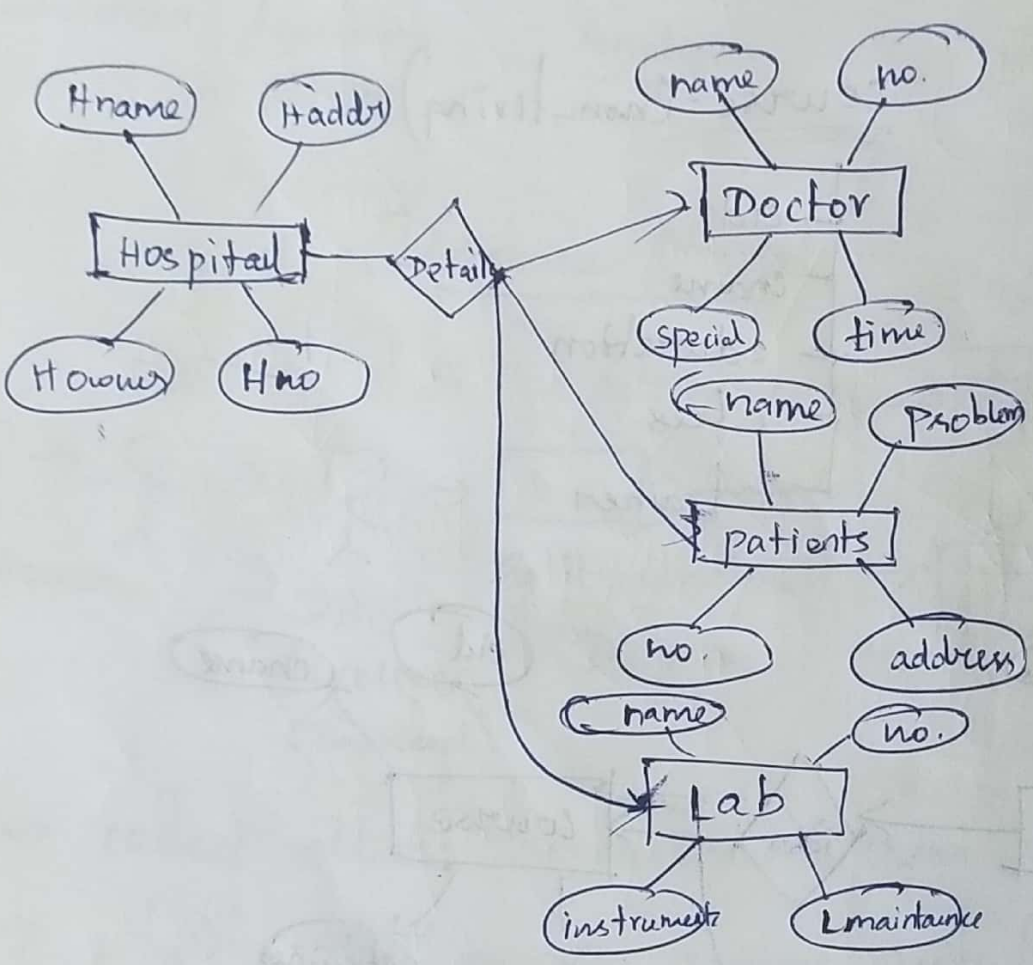


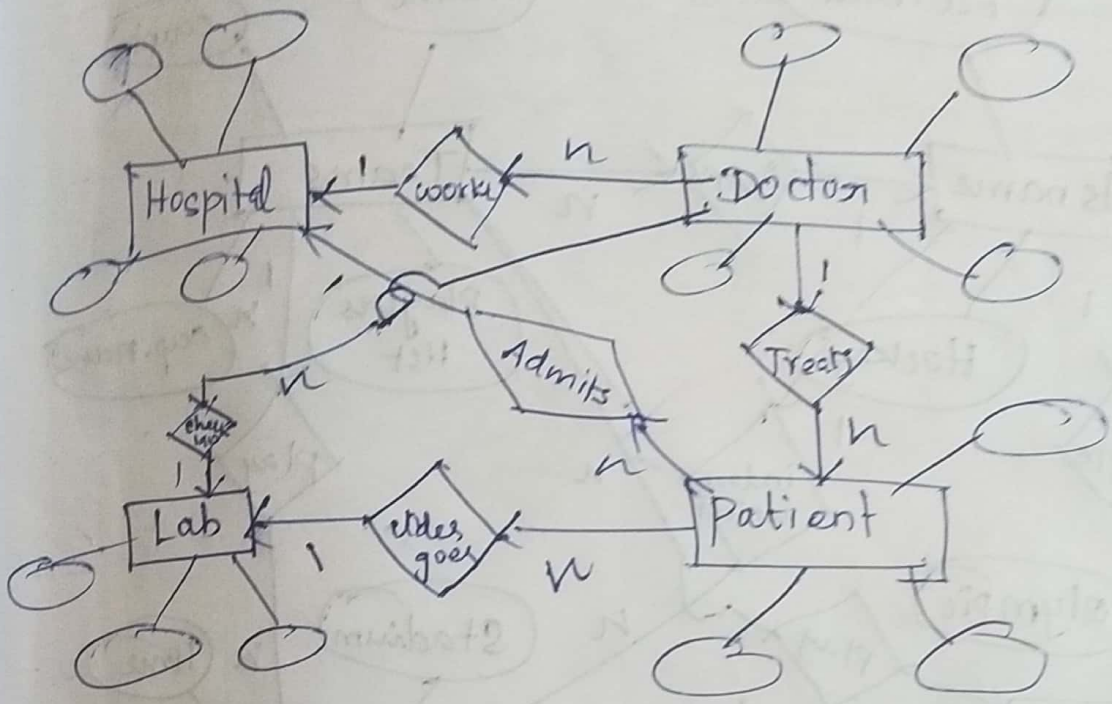
Lab (nliving)
- Lname
- L no.
- L instrumats
- L maintenane

Hospital (non living)
- H name
- Haddress
- H no.
- H owner

Doctor (living)
- D. name
- D no
- D specialization
- D time

Patients (livings)
- p name
- P problem
- P. no
- p. addres



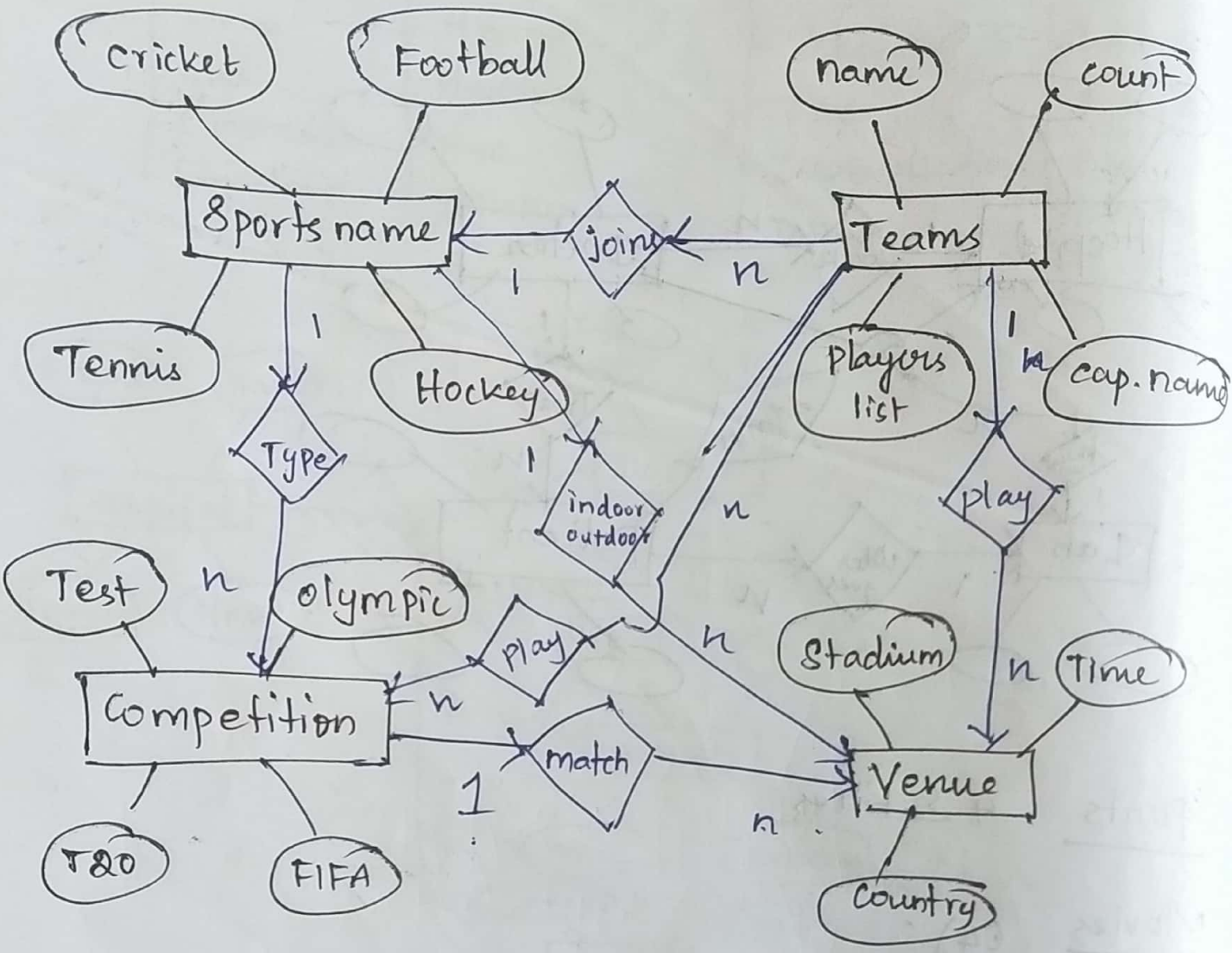


Sports . 4 Entities

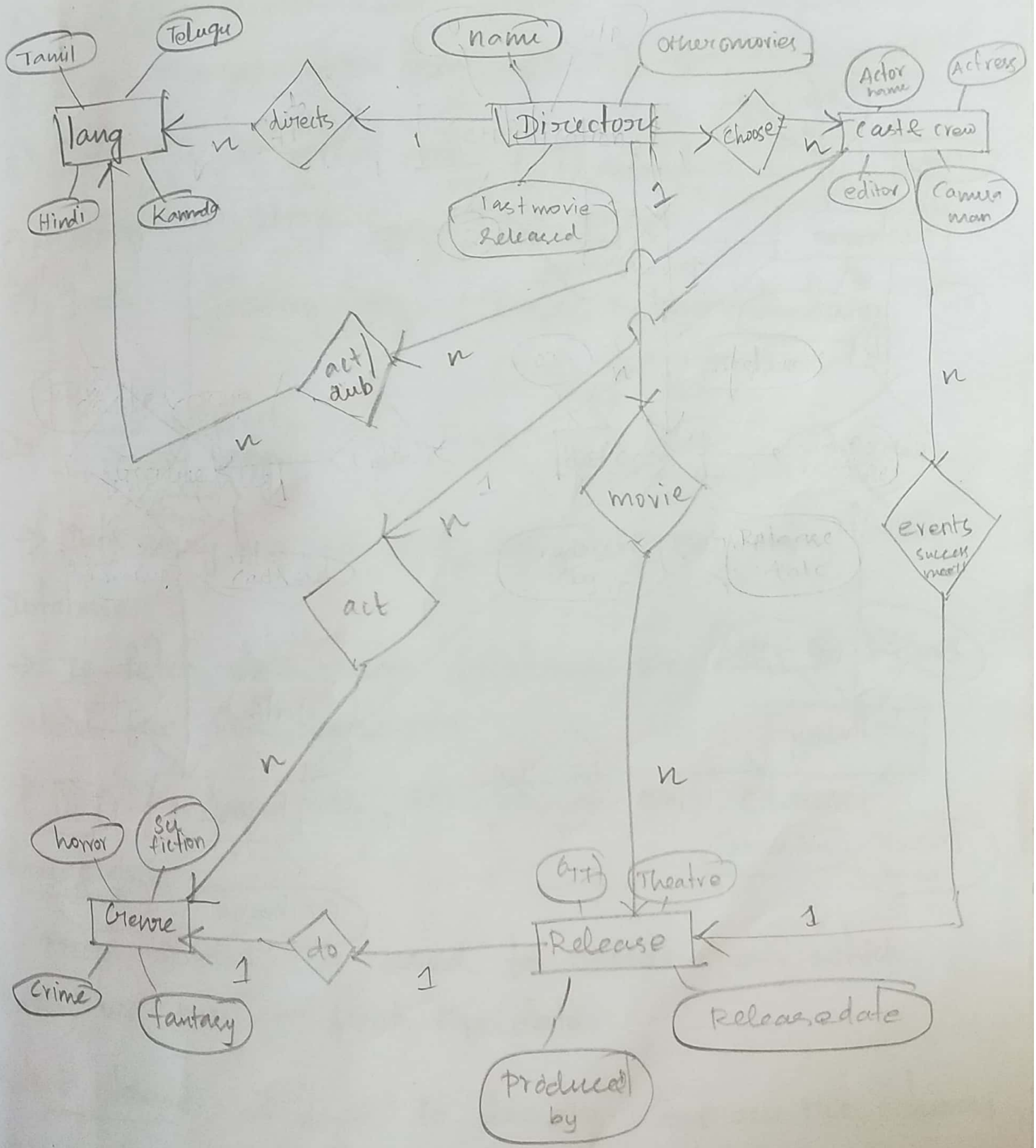
Movies (4)

Ecommerce (4)

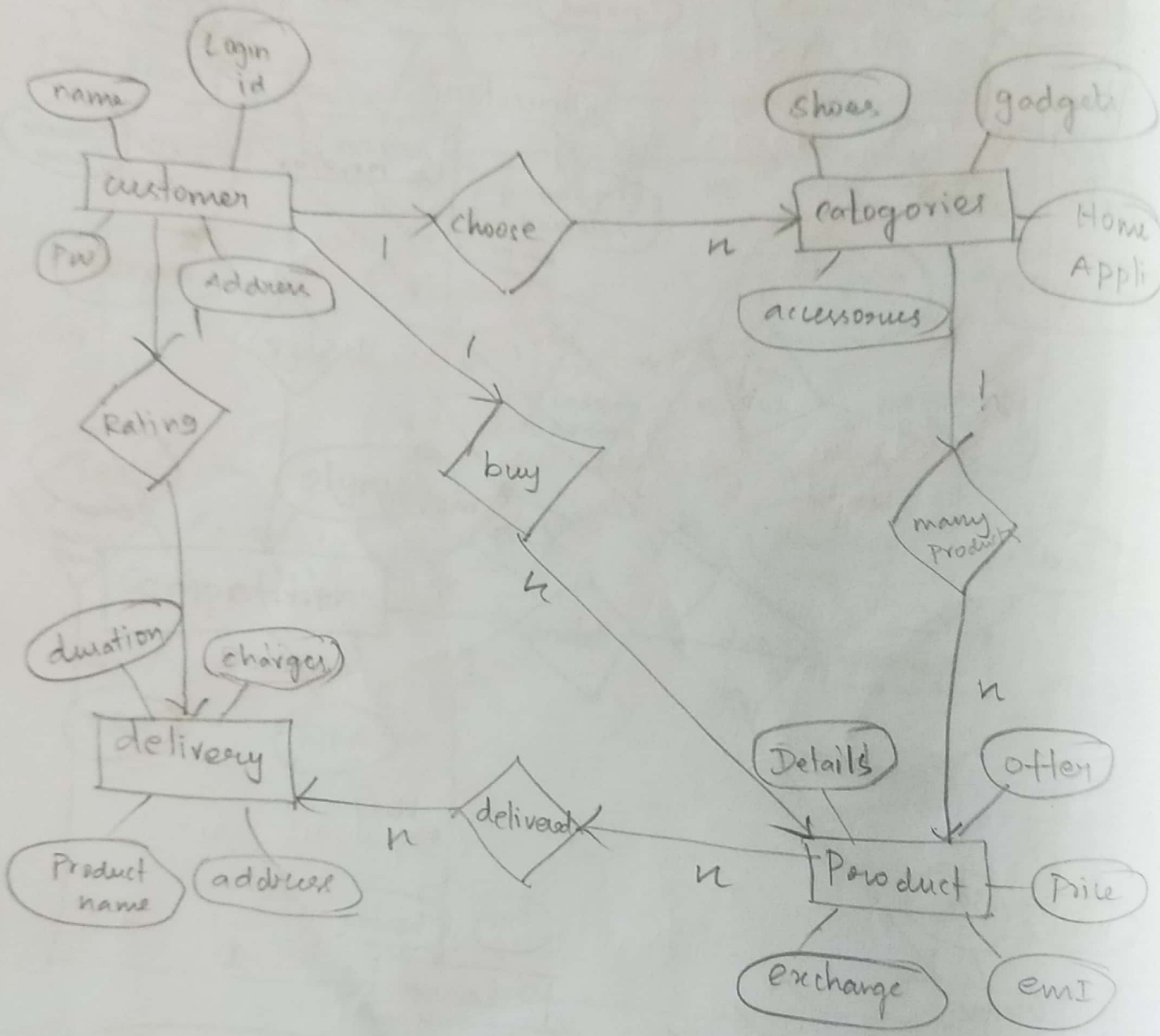
Sports:



Movies



E-commerce:



23/6

Sub languages of SQL

1) Data Definition lang. (DDL)
2) Data Manupulation lang. (DML) insert, delete, update
3) Data Transaction lang. / Transaction control lang. (DTL / TCL)
4) Data control lang. (DCL) Grant, Permission
5) Data Query lang. / Data retrieval lang. (DQL / DRL)

Data Query lang.:

→ Data query lang. is used to fetch the data from Database.

→ To fetch data from database we need to know about few SQL Keywords.

→ In SQL Keywords are known as 'clauses'.

From clause:

From clause is used to decide from which table we need to fetch the data.

Select clause:- is used to decide what are the columns we need to fetch from the selected table.

```
Select *
from tab;
```

* → means fetching all the information.

tab → is a keywoed which is used to check what all the tables present in existing user.

```
desc tablename;
```

By using desc command we can check table structure

```
Select *
from emp;
```

```
Select *
from employee;
```

Error: Table doesnot exist.

→ SQL (lang.) is not a case sensitive but data presenting DB is format sensitive.

→ SQL queries should ends with semicolon.

Write a query to fetch employee names and their jobs from employee table

```
select empnames, empjobs
from emp;
```

error: empnames - invalid identifier

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

```
set pagesize 243;
set linesize 232;
select * from emp;
```

→ Setting pagesize & linesize is temporary changes.

24/6

Write a query to fetch employee name, salary and their joining dates from employee table

```
Select Ename, Sal, Hiredate
from emp;
```

The process of fetching the data in column wise is known as 'projection'.

→ Projection can be achieved by using 'Select' clause

→ By using 'select' clause we can decide which column to select which col not to select.

Aliasing: means changing (or) renaming the column or table names.

→ To perform aliasing we use "as" Keyword

→ col aliasing will be done in 'select' clause.
Table aliasing will be done in 'from' clause.

```
select ename from emp;

ename
:
:
:
```

(Aliasing)

```
select ename as empname
from emp;

EMP NAME
:
:
:
```

```
select emp name from emp;

Error: 'emp name': is invalid identifier
```

→ Aliasing changes are temporary changes means if we perform col aliasing that aliasing names will not be affected to table col names.

Q Write a query to fetch employee name & hiredate while fetching alias hiredate as joiningdate.

```
select hiredate as joiningdate , ename
from emp;
```

```
Select ename, hiredate joiningdate
from emp;
```

EName	joiningDa
⋮	⋮

Note:

→ While performing col aliasing 'as' keyword is optional.

→ If we are not providing 'as' keyword then next word will be considering as aliasing name, if there is single word.

```
Select ename, hiredate joining date
from emp;
```

ERROR: from keyword not found where expected

```
Solect ename , hiredate joining_date
from emp;
```

ENname	joiningDate

To maintain the same case we use " ".

```
select ename, hiredate "joining date"
from emp;
```

ENAME	joining date

Operators:

1) Arithemetic operators:

+, -, *, /, % → pattern Matching

→ All arithmetic operators can be used in 'select' clause. but not in 'from' clause.

Q. Write a query to fetch annual salary for each employee:

```
select  sal * 12
from emp;
```

```
select ename, sal, sal * 12
from emp;
```

EName	sal	sal * 12
:	:	:
:	:	:
:	:	:

```
select ename, sal, sal * 12 as Asal
from emp;
```

EName	Sal	ASal
:	:	:
:	:	:
:	:	:

Q. Write a query to fetch employee name & their salaries while fetching the data add 500 to each emp salary and alias added salary as incremented salary.

```
Select ename, sal + 500 as incsal
from emp;
```

Q. Write a query to fetch empname & their daily Salaries

```
Select ename, sal/30 as dailysal
from emp;
```

sal

Literals:

→ Literal is nothing but data.

→ That data can be taken from tables (or) Outside the tables.

→ There are 3 types of literals:

1) Number
2) String (varchar) - combination of no. string specialchar
3) Date

→ All the type of data must be enclosed within single quotes ' '.

→ Any types of Literal can be used in 'select' clause.

The property of literal says that if we are using any type of literal in 'select' clause the same literal will be executed for each & every row of a given table.

```
select 'jspiders' from emp;

'JSPIDERS
---------
Jspiders
Jspiders
:
:
:
:

select 'ename' from emp;

ENAME
----------
ename
ename
:
:

select '10' from emp;

'10
10
:
:
:
(14 rows)

select 10 from emp;

10
10
10
:
:
:
:

select '19-DEC-98' from emp;

19-DEC-98
19-DEC-98
:
:
:
```

→ String and data type of literals must be mentioned in single quotes but no. type of literal is optional to be mentioned in single quotes.

Concatination (character operator)

→ Concatination means joining (or) combining two (or) more columns (or) two or more literals.

→

```
Select sql||plus from emp;
ERROR : 'PLUS' - invalid identifier

Select 'sql'||'plus' from emp;
'sql||
 sqlplus
 sql plus
  :
  :
  :
 Select 'sql' '||' 'plus' from emp;
 'sql plus'
  :
  :
  :
 Select 10||12 from emp;
 1011
 1012
```

Select ename || 'Works as ' || job from emp;

ENAME || 'WORKAS' || Job

Smith works as clerk

.

.

.

Write a query to display all the employee details as follows Smith is Working in dept no. and get a Sal of 800Rs use proper aliasing.

Select ename Name || 'Working in' || dept no || 'and get a' || sal as Salary of package

dept number

Select ename || 'Working in departnumber' || deptno || 'get a salary of ' || sal

from emp;

Select ename name, 'Working in a department number' || deptno did, 'get a salary of' || Sal Salary

from emp;

Dual table: is dummy table used to work with an literals.

```
Select 1234||567
from Dual;
```

1234561

```
Select 'SQ'||'L'
from Dual;
```

'SQ
SQL

```
select *
from Dual;
```

D
—
X.

```
Select '787JKLM'||789
from dual;
```

787JKLM789

Comparasion or Relational Operator:

>, <, >=, <=, !=, =

Not equal → !=, <>, ^=

Using this operator 'where' is executed.

Syntax for 'where' clause:

```
select col1, col2,
from tablename
where (condition);
```

where clause is used to fetch the data in row-wise.

```
Select *
from emp
where ename = 'SMITH';
```

```
Select * from emp;
where job = 'MANAGER';
```

ENAME
SMITH ....

```
select *
from emp
where ename = 'smith';
```

No rows selected

Write a query to fetch all the details of clerk.

```
Select *
from emp
where job = 'clerk';
```

Write a query to fetch name of the employee, job, dept. no. ~~name~~ of employee King.

```
Select *
from emp
where ename = 'KING', job = ', deptno
```

```
Select ename, job, dept-no
from emp
where ename = 'KING';
```

Write a query to fetch salary, job, of employee Allen.

```
Select salary, job from emp
where ename = 'ALLEN';
```

Write a query to fetch all the details of employee who gets the salary more than 1000

```
Select * from emp
where Sal > 1000;
```

Write a query to fetch all the details of employee who gets salary less than equal to 3500

```
Select * from emp
where Sal <= 3500;
```

write a query to fetch all details of employee except clerk.

```
Select * from emp
where ename/Job != 'CLERK';
```

Write a query to fetch name, designation dept.no & who gets the salary > 2000

```
select ename, deptno, job from emp
where sal > 2000;
```

Logical Operators:

AND, OR, NOT

AND →

con1	con2	R
T	T	T
T	F	F
F	T	F
F	F	F

OR →

T	T	T
T	F	T
F	T	T
F	F	F

Write a query to fetch Smith and Allen all the records.

```
Select * from emp
where ename = 'SMITH' AND ename = 'ALLEN';
```

No rows selected.

```
Select * from emp
where ename = 'SMITH' OR ename = 'ALLEN';
```

Write a query to fetch all the details of employee who gets Salary of more than 1000 & less than 3500

```
select * from emp
where Sal > 1000 AND Sal < 3500;
```

write a query to fetch all clerk, salesman, manager records.

```
Select * from emp
where job = 'CLERK' OR job = 'SALESMAN' OR
                    job = 'MANAGER';
```

write a query to fetch all details of employee Sal >= 2000 only fetch manager records.

✓
```
select * from emp
where Sal >= 2000 AND/OR job = 'MANAGER';
```

```
Select *
from emp
where NOT job = 'CLERK'
```

```
select *
from emp
where NOT job = 'CLERK' OR job = 'MANAGER'
```

28/6

Write a query to fetch all the details manager, Salesman except clerk & getting Sal. of more than 1500 <= 4000

```
select *
from emp
where not job = 'clerk' (job = 'MANAGER' (OR)
                                    job = 'SALESMAN'
                         AND (job != 'CLERK')
where
        AND (Sal > 1500 AND Sal <= 4000;)
```

```
where not job = 'CLERK' AND (job = 'MANAGER' OR
                             Job = 'SALESMAN')
        AND (Sal > 1500 AND Sal <= 4000);
```

Write a query to fetch all details of employee like james, scott, Adams, Smith & Sal should be >= 1000.

```
Select *
from emp
where (ename = 'JAMES' OR ename = 'SCOTT' OR
       ename = 'SMITH' OR ename = 'ADAMS')
```

AND Sal >= 1000;

Write a query to fetch all the details of employee who are getting commission >= 300

```
Select *
from emp
where COMM >=300;
```

IN Operator (Special Operator)

→ Fetching the Multiple data from the same column.

→ (Same as 'OR' operator)

```
Select * from emp
where ename IN ('SMITH', 'ALLEN', 'FORD');

Select * from emp
where ename NOT IN ('SMITH', 'ALLEN');
```

⇓

```
where ename != 'SMITH' AND ename != 'ALLEN'
```

LIKE Operator: (Special operator)

→ It is used when you know some particular information.

→ only used for String (Varchar)

_ → used for single character. - know particular information

% → used for many character.

↓

Don't know anything

```
Select *ename
from emp
where ename like 'S%';
```

ENAME

SMITH

SCOTT

Starting with 'S' ends with 'T'

```
Select ename
from emp
where ename like 'S%T';
```

ENAME

SCOTT

→ Like operator used for giving a pattern for the columns.

→ It works on varchar (string)

→ We use 'like' operators when we have particular information.

→ '_' & '%' indicates the missing character.

Q1 Write a query to fetch name of the employee starting with an K letter & ending with an 'G' letter.

```
Select ename from emp
where ename like 'K%G';
```

Q2. Write a query to fetch name of the employee his name ending with an S

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

29/6

Write a query to fetch dept name starting with an 'S' third character is 'L'

```
select deptname from dept
where deptname like 'S_L%';
```

W. Q. to fetch deptname 6th character is 'T' ending with an S

```
Select dname
from dept
where dname like '_____T%S';
```

W.Q.T. Fetch reporting manager number ending with an character 8

```
Select MGR
from emp
where MGR like '%8';
```

W.Q.T. fetch all details of employee who joined in dec

```
Select *
from emp
where Hiredate like '%DEC%';
```

W.Q.T fetch all the details of employees like Salesman, Manager in dept 10, 20, 30 and reporting manager no. ending with an 2.

```
Select job, deptno, mgr
from emp
where job like deptno in ('10', '20', 30)
where mgr like '%2';
```

```
Select *
from emp
where job in ('SALESMAN', 'MANAGER') and deptno in (10,20,30)
                and mgr like '%8';
```

W.Q.T fetch job of the employee 8th character is 'N' the word consists of 9 letters.

```
Select job
from emp
where ~~ename~~ job like '_______N';
```

```
Insert into (empno, enjob)
```

between (special operator)

→ It used for finding the range between the values.

→ It works on number, string and character but not in

SYNTAX:

```
Select col1, col2
from table name
where colname between [lowrange and highrange]
```

```
Select *
from emp
where sal between 1000 and 2000;

Select *
from emp
where ename between 'A' and 'F';

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

Select deptNo
from emp
where deptno between '10' and 30;

Select hiredate
from emp
where hiredate between '01-JAN-80' and '01-DEC-80';
```

30/6

Write a query to fetch all the details of employe who gets salary between the range of 1500 to 3500

```
Select *  from emp
where  sal  between  1500  and 3500 ;
```

Write a query to fetch all the details of employees like salesman, manager, clerk and joining data between the range of '01-JAN-80' to '01-JAN-83'. in dept 10, 20, 30

```
Select * job, joining date
from emp
where job like in ('salesman', manager, clerk)
    and joining date (between '01-JAN-80' and
                      '01-JAN-83')
                and deptno in (10,20,30)
```

Write a query to fetch employee name start from 'A' to 'M':

```
Select ename
from emp
where ename 'A between 'A' and 'M';
```

Write a query to fetch all the details of employee like salesman, clerk salary lies b/w 800 to 1500

```
Select *
from emp
where job in ('SALESMAN', 'CLERK') and
      Sal between 800 and 1500;
```

IS Operator (Special operator)

IS (or) IS null - used to fetch the null values.

Write a query to fetch all the details of employee who don't have reporting manager.

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

Write a query to fetch all the details of employee who not getting a commission.

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

Data Manipulation language: (insert, update, delete)

Insert:

Syntax

1) insert into tablename
   values (colval1, colval2, ...)
2) insert into tablename (col1, col2, col3, ...)
   Values (val1, val2, ...)

Syntax : should be in order emp, empname----

```
insert into emp
values ('101', 'abc', 'clerk', 5900, '01-Jun-22',
                                        100, 30);
```

TCL command - 'commit' used for permanent data
(insert, update, delete)

insert into emp
Values (1001, 'abc', 'clerk', ---);

Error: not specified precision in the col.

Error: Unique constraint

hiredate as '01/Jun/22'

insert into emp
Values (1001, null, null, ----)

insert into emp
values (null, 'abc', -----)

Error: can't insert Null into ('scott', 'emp', 'empno')

Number (7, 2)

70000.12
(7-2)
5

hiredate '01/JAN/22' -

'01-JAN-22'.

01/07

insert into bonus
Values ('abc', 'clerk', 1000, 100);

insert into emp (empno, job)
Values (101, 'clerk');

insert ~~table~~ into dept (deptno, job)
values (60, 'clerk');

Update:

Syntax:

```
Update  tablename
Set  col1 = value1 , col2 = value2, ....
where condition;
```

Eg:

```
Update  emp
Set  sal = 10,000
where ename = 'SMITH';

Update  emp
Set  job = 'SALESMAN', MGR = 1001
where  ename = 'SMITH';
```

Write a query to ~~fetch~~ ~~display~~ update manager salary for 30,000

```
Update  emp
set  sal = 30000
where job = 'MANAGER';
```

02/7

```
Update emp
Set ename = 'modi';
```

Delete:

Syntax:

```
Delete from tablename
where condition;
```

```
Delete from emp
where empno in (101, 102);
```

Note:

→ DML command is temporary we can undo the operation to save the data permanently we'll use TCL command 'commit'.

→ Delete and update compulsarily we need to use 'where' condition otherwise whole data will be vanished.

TCL - Transaction Control language: → control the every transaction

1) commit
2) Rollback
3) Savepoint

Commit is used for save the transaction permanently.

Syntax:

```
commit ;
Savepoint;
Savepoint savepointname;
Rollback ;
Rollback to savepoint;
```

Rollback, Savepoint are temporary.

Commit - used to save the data transaction permanently.

Savepoint - used to save the transaction temporarily.

Rollback;

↳ used for undo-ing the transaction till 'commit'.

Rollback to savepoint;

↳ used for undo-ing the transaction till savepoint.

Eg:

Savepoint a;

Savepoint b;

Rollback;

insert into bonus

Values ('ram', 'clerk', 5000, 200);

insert...

insert...

Savepoint a;

insert into bonus

Values (...

Rollbacks

04/7

Data Definition language: (DDL)

1) Create
2) Alter → drop, add, rename, modify } columns
3) drop
4) truncate

Constraints:

- → Primary key
- → foreign key
- → check
- → default
- → unique
- → not null

→ Constraints: is used for providing the rules for the data.

→ Not null: Ensure that that column can't have null values.

→ Default: set the default value for the column if no value is specified.

→ Check: Ensure that the values in a column satisfies specific condition.

→ Unique: Ensure that all values in a column are different.

→ You cannot insert duplicate values.

→ It accepts null values.

→ In one table, one (or) more unique can be present.

Primary key:

→ It is a column whose data determine the particular records uniquely.

→ It is the combination of unique and not null.

→ In a table only one column declare as primary key.

Foreign key:

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

→ Foreign key column can contain duplicate & it can contain null values.

→ We cannot insert any data which is not present in referring table column.

→ In one table more than one foreign key can be present.

5/6

Syntax for creating the table:

```
create table tablename (
    );
```

Eg: (w/o constraints)

```
Create table prod (
    pid     number(4),
    Pname   varchar(20),
    price   number(4,2));
```

Create course ((default value)

cid number primary key,

cname varchar (20) not null,

tri Varchar (20));

Create a table Vaccination consists of vid, vname, age, Gender, and country. (unique id)

Create table Vaccination (

✗ vid number(10) unique key,

vname varchar (20),

age ~~Varchar (20)~~ number (3), check >= 5,

Gender Varchar (10) check (Male (or) Female),

country Varchar (10) ~~default~~ (India));

✓ create table Vacci (

vid number (10) unique ~~key~~,

Vname Varchar (20),

age number (3) check (age >= 5),

gender Varchar (20) check (gender in ('m', 'F')),

Country Varchar (20) default (India));

```
insert into Vacci
Values (1007, 'a', 7, 'usa');

insert into Vacci (vname)
values ('abc');
```

Relation b/w tables:

```
Create table Student(
Sid     number(20) primary key,
Sname   Varchar(20),
marks   number(4,2),
cid     number(10) references course(cid));
                   (foreign key)
```

b/w

Alter (permanent)

- → add
- → drop (remove)
- → Rename
- → Modify

} columns

(i) Add: (adding new column)

```
Alter table tablename
add new column datatype(size) consi;
    name
```

Eg:

```
Alter table emp
add sid number(20) notnull;
```

```
Alter table emp
add age number(2);
```

Error: not null

notnull, default will be used in Empty table.

(ii) drop (deleting the column)

```
Alter table tablename
drop column columnname;
```

Eg:

```
Alter table emp
drop column sid;
```

Delete the foreign key first then primary key because we have references.

```
alter table Stude
drop column age;
```

(iii) Rename: (Renaming the col name)

```
Alter table tablename
rename column oldcolumn to newcolumn;
```

Eg:

```
alter table emp
rename column ename to name;
```

iv) Modify - used for changing the datatype.

```
alter table tablename
modify column datatype(size);
      name
```

Note:

- For changing any datatype the column should be empty.
- Only we can modify the datatype size, if records are present.

Eg:

```
alter table stud
modify sid number(20);
```

```
rename old table to new table;
       name        name
rename Stude to S;
```

} Renaming the table.

Drop - used for deleting the table.

Sy:

```
drop table tablename;
```

Show recyclebin; (only store the deleted tables.)

flashback table tablename to before drop;
(restoring the deleted table)

Purge table tablename; (delete the table from recycle permanently.)

Purge recyclebin; (delete all the tables from recycle bin)

Eg:

```
drop table s;

flashback table s to before drop;

show recyclebin;
Select * from tab;
```

7/4 Truncate: (deleting the all records from table permanently.)
(Not in recyclebin)

Syntax:

```
truncate table tablename;
```

Eg:

Truncate Table Sal;

DELETE - is the DML command

- It is not autosaved
- we can undo the every operation
- It is slower than truncate
- In DELETE we use 'where' condition to delete the specific record

Truncate - is the DDL command

- Truncate command delete the record permanently.
- It is faster than delete
- In Truncate we can't use 'where' condition so that we cannot delete specific record.

SQL Functions: - inbuilt fn. to do a specific task.

→ Single row function - work on every single row gives Multiple answer

→ Multi row function - " gives single answer

Single row function: (not permanent)

- number fn:
  - mod()
  - abs()
  - sqrt()
  - power()
  - floor()
  - ceil()
  - round()
- character function:
  - length()
  - upper()
  - lower()
  - substr()
  - replace()
  - trim()
- date fn:
  - month_between()
  - session timezone()
- conversion fn:
  - to_date()

current_date()

oracle sysdate. sysdate()

Here, we can use function in 'where' condition.

no function

Multiraw function: (In where condition used)

- max()
- min()
- avg()
- count()
- Sum()

8/7

number function:

mod():

→ Select mod (24, 2)
from dual;

→ Select empno, mod (empno, 2)
from emp;

→ Select mod (24, 2)
from emp;

→ Select mod (sal, 2)
from emp;

abs():

→ Select abs (14.35)
from dual;

→ Select abs(-123.56)
from dual;

Sqrt():

→ Select Sqrt(45)
from dual;

→ Select Sqrt(25)
from dual;

→ Select Sqrt(Sal)
from emp;

Power():

Select Power(4,2)
from dual;

select power(20,2)
from dual;

floor():

Select floor(4.2)
from dual;

Select floor(5.7)
from dual;

ceil():

Select ceil(4.1)
from dual;

Select ceil(-5.1)
from dual; -5

Select floor(-5.1)
from dual; -6

round(): (combination of floor & ceil)

Select round(6.4)
from dual;

Select round(6.8)
from dual;

Select round(-4.1)
from dual;

character function:

length():

```
Select length ('JANANI')
from dual;

Select dname, length(dname)
from dual;

select length ('name is Jane')
from dual;
```

upper():

```
Select upper ('hello')
from dual;
```

lower():

```
select lower ('STRUCTURED')
from dual;
```

12/7

Replace():

1) replace (source, old delchar, newchar);

2) replace (source, delchar);

1) Select replace ('apple', 'p', 'x')
from dual;

Select replace ('apple', 'ap', 'g')
from dual;

Select ename, replace (ename, 'A', 'c')
from ~~dual~~ emp;

Select replace ('good morneng', 'morning', 'evening')
from dual;

2) replace (soure, delchar)

Select replace ('apple', 'pp')
from dual;

Select replace ('applepie', 'pp')
from dual;

Trim(): - To delete the starting and Ending space.

Select trim (' java ')
from dual;

rtrim - functions.
ltrim

Select trim ('j ava')
from dual;

IQ

Substr():

1) Substr (source, Position);

2) Substr (Source, position, length);

→ To fetch particular character.

length - +ve

Eg:

select substr ('goodmorning', 8)
from dual;

select substr ('goodmorning', -7)
from dual;

select substr ('goodmorning', -11)
from dual;

select ename, substr (ename, 2)
from dual;

1 2 3 4 5 6 7 8 9 10 11
g o o d m o r n i n g

-11 -10 9 -8 -7 -6 -5 -4 -3 -2 -1

select substr ('goodmorning',
from dual;

empty

```
Select substr ('good morning', 5, 2)
from dual;

Select substr ('good morning', -10, 2)
from dual;

Select substr ('goodmorning', -11, 1)
from dual;

Select ename, substr (ename, 1, 3)
from emp;
```

Date function:

```
Select sysdate                          12-JUL-22
from dual;

Select current_date                     12-JUL-22
from dual;

Select sessiontimezone                  +5.30
from dual;

Select months_between ('12-jul-22', '01-jan-22')
from dual;
```

current — previous/past

months_between - to calculate the months experience.

Select months_between (current_date, (or) sysdate
from dual; '01-JAN-22');

Select months_between ('01-JAN-22', sysdate)
from dual;

-6.37

Select ename, (months_between ('12-Jul-22', Hiredate))/12
from emp;

Select ename, months_between (current_date, hiredate)/12
from emp;

Write a query to display the no. of year of experience of all the employee.

Select months_between (current_date, hiredate))/12
From emp;

13/7

Conversion function:

to_date

→ Converts any ~~normal~~ format date to Sql date format.

```
Select to_date ('20-05-21', 'dd-mm-yy')
from dual;

Select to_date ('11-04-2022', 'mm-dd-yyyy')
from dual;
```

Multi row function: - (doesn't allow other col in select clause)-

1) Write a query to fetch the maximum salary.

```
Select max(sal)
from emp;

Select ename, max(sal)
from emp
```

→ Error - not a single row fn.

2) Write a query to fetch ~~who joined the last?~~ max hiredate?

```
Select max(hiredate)
from emp;

Select max(deptno)
from emp;
```

Select max(ename) (Don't use)
from emp;

WARD

(By ASCII Value)

min():

W.Q.T.F min salary.

Select min(sal)
from emp;

800

W.Q.T.F ~~value~~ joined the min hiredate

Select min(hiredate)
from emp;

'17-DEC-80'

avg():

Select avg(sal)
from emp;

2073.214

W.Q.T.F avg. sal among the manager.

Select avg(~~mgr~~ sal)
from emp
where job = 'MANAGER';

Sum():

```
select sum(sal)
from Emp;

select sum(empno)
from emp;
```

count():

```
select count (mgr)
from emp;                    13

select count (comm)
from emp;                    4
```

Note:

Single row fundion can be used in select, where having etc..

Multirow fn. can be used in select having groupby but it cannot be used in where clause

Sub Query:-

↓

Query inside the query

Based on one result, we are fetching another query result.

Syntax:

1) Select col
   from tablename
   where condition (Sub Query);

2) Select
   from (sub query)
   where condition;

(same table)

(We can merge two syntax if we need)

1) W.Q.T.F who is getting maximum salary.

```
Select *
from emp
where sal = ( select max(sal)
              from emp);
```

2) W.Q.T.F all the details of employee who is getting minimum salary.

```
Select *
from emp
where Sal = ( select min (sal)
              from emp);
```

14) W.A.T.F all the details of employee who are working in newyork city.

```
Select *
from dept emp
where location = 'Newyork'; (select *
                             from dept)
                             where to
```

```
select *
from emp
 where dept_no = (select deptno
                  from dept
                  where loc = 'Newyork');
```

W.Q.T.F all the details of employee who works in Salesdept.

```
Select *
from emp
where deptno = (Select deptno
                from dept
                where ~~loc = 'chicago'~~)
                (or)
                dname = 'sales');
```

10 = 30
20 = 30
✓ 30 = 30

W.Q.T.F Second highest Salary.

```
select max(sal)
from emp
where sal < (select max(sal)
             from emp);
```

(or)

```
where sal =! (select max(sal)
              from emp);
```

~~5000~~ = 15000
3000 = 15000

5000 < 5000
3000 < 5000

W.Q.T.F all the details of employee who is getting second highest salary.

~~Select * ename, max(sal)~~
~~from emp~~
~~where sal != (select max(sal)~~
~~from emp);~~

```
Select * from emp
where sal = ( select max(sal)
              from emp
              where sal < (select max(sal)
                           from emp))
```

Orderby:

1) Select
   from table name
   orderby column asc/desc , col asc/dec. ... ;

2) Select
   from tablename
   where condition
   Orderby col asc/desc , col asc/desc ....

```
Select sal
from emp
Order by sal desc;
```

(by default - it takes as asc)

```
Select sal
from emp
Order by sal asc;
```

```
Select sal
from emp
Order by sal;
```

```
Select ename
from emp
Order by ename;
```

(It'll arrange by ASCII value)

If we use multiple columns it will arrange based on the first column.

```
Select ename, sal
from emp
Order by sal asc, ename desc;
```

```
Select ename
from emp
where ename in ('SMITH', 'SCOTT', 'KING');
Order by ename desc;
```

```
select ename
from emp
order by ename desc;
where ename in ('SMITH', 'SCOTT', 'FORD', 'KING');
```

→ Error: SQL command not properly ended.

15/7

Distinct:

- removes the duplicate values & gives unique values.
- can take 2 or more columns. but it takes as one column.

```
Select distinct sal
from emp;
```

```
Select distinct sal, deptno
from emp;
```

✓ 5000	10
✓ 4000	20
✓ 3000	30
✓ 3000	10
✗ 4000	20

```
Select distinct sal, job, deptno
from emp;
```

```
Select distinct *
from emp;
```

```
select distinct *
from bonus;
```

rownum: → Virtual col present in every table and everytime it starts from fetching first records.

→ Only '<' (or) '<=' is used.

fetch start from
→ It won't Middle Values.

```
Select *
from emp
where rownum < 5;
```

```
Select *
from emp
where rownum <= 3;
```

```
Select *
from emp
where rownum = 1;
```

```
Select *
from emp
where rownum = 2;
```

↓
no rows selected.

>= (will not work)

W.R.T.F deptname, location of employee Smith.

```
Select dname, loc
from dept
where deptno = (select deptno
                from emp
                where ename = 'SMITH');
```

W.Q.T.F all the details of Smith's manager.

(Mock)

```
Select *
from emp
where empno
mgr = ( Select mgr
        from emp
        where ename = smith );
```

Third highest salary.

```
Select max(sal)
from emp
where sal < ( select max(sal)
              from emp
              where sal < (select max(sal)
                           from emp) );
```

```
Select min(sal)
from emp
where sal = (select distinct sal
             from emp orderby sal desc
             where rownum <= 3);
```

5000
3000
2975

2nd Syntax:

```
Select min(sal)
from ~~emp~~ ( select distinct sal
               from emp
               order by desc )
where rownum <=3;
```

Third ~~second~~ highest salary.

2975

duplicates	
5000	5000
3000	3000
3000	2975
2975	2850
2850	2450
2450	1600
1600	1500
1500	1300
1300	1250
1250	1100
1250	950
1100	800
950	
800	

5th

```
select min(sal)
from ( select distinct sal from emp
       order by desc) where rownum <=5;
```

16/7

3rd highest:

```
Select *
from ( Select *
from (select * from emp
      order by sal desc)
      where rownum <=3
      order by sal)
      where rowcolumn <=1;
```

O/P: (duplicate value)

empno, ename, job, ...

3000

```
Select *
from emp
where sal = ( select *
              from (select *
                    from ( select distinct sal from emp
                           Order by sal desc
                           where rownum <= 3
                           Order by sal)
                    where row column <= 1);
```

* all details

2975

3rd

empno	ename	job	....	sal	comm	dno
~~2975~~	Jones			2975		

```
Selcit *
from emp
where sal = ( select min (sal)
              from (select distinct sal
                    from emp
                    order by desc)
              where rownum <= 3 );
```

W.Q.T.F all the details of employee who is getting more than avg salary.

```
Select *
from emp
where sal > (select avg(sal)
             from emp)
```

W.Q.T.F dname of ~~all details~~ employee allen

```
Select dname
from dept.
where deptno = ( select deptno
                 from emp
                 where ~~dnam~~
                 ename = 'Allen')
```

(↓ IN)

W.Q.T.F dname, loc of Allen, Smith, ward.

```
Select dname, loc
from dept
where deptno = ( select deptno
```

from emp
where ename in ('Allen', 'smith', 'ward'));

Errors: Single-row Sub-query

select dname, loc
from dept
where deptno IN (");

W.Q.T. display the employee whose loc which has one 'o' in it

select * ~~loc like '%o%'~~
from dept
~~where~~ loc ~~like~~ '%o%'
where deptno ~~in~~ in (select deptno
from ~~emp~~); dept
where loc like '%o%');

~~select *~~
~~from (select~~ deptno

~~where loc like '%o%'~~

W.Q.T.F loc of employee working as a Salesman

```
Select loc
from dept
where deptno in (select deptno
                 from emp
                 where job='salesman');
```

18/7

W.Q.T.F 5th highest salary

```
Select *
from (select *
from (select sal
      from emp
      order by desc
      where rownum <=5
      order by sal)
where rownum <=1);
```

```
(Select * min(sal)
from (select distinct sal
      from emp
      order by desc
      where ~~rownum <=5~~
      where rownum <=5)
```

5000
4000
3000
2~
1000

W.Q.T.F display all the employees salary greater than avg salary of deptno 20.

```
Select
    ( select *
      from emp
      where deptno=20 and
      sal > (Select avg(sal)
             from emp
             where deptno = 20) );
```

~~where deptno = 20);~~

```
select *
from ( select * from emp
       where sal > (select avg(sal) from emp
                    where deptno = 20) )
where deptno = 20;
```

W.Q.T.F display empno and name for employee working as clerk and earning highest salary among the clerk.

~~Select empno, ename~~

```
(Select empno, ename
 from emp where job
```

Select empno, ename, sal ≠ 'CLERK' and

from emp

~~job 'clerk' and~~

where sal >= (select max(sal)

job = 'clerk' and from emp

where job = 'CLERK')

✓ Select empno, ename.

from emp

where sal = (select max(sal)

empno ename

7934 Miller

from emp

where job = 'CLERK');

where ~~job = 'clerk'~~

W.Q.T. display name of salesman whose is earning the salary more than highest sal of clerk.

select ename from emp.

where job = 'salesman' and

sal > (select max(sal)

from emp

where job = 'CLERK')

W.Q.T.F avg salary among the clerk.

```
select sal
from emp
where sal = (select avg(sal)
             from emp
             where job = 'clerk');
```

W.Q.T.F display all details of employee of salesman & manager where loc != dallas.

```
Select *
from emp
where job in ('SALESMAN', 'MANAGER') and deptno in (select deptno
                                                    from dept
                                                    where loc != 'Dallas');
```

19/7

joins

Multiple different tables

→ used to fetch the data from

→ 7 types of joins

Types:

→ cross join (or) cartesian join

→ Inner join

I.Q↑

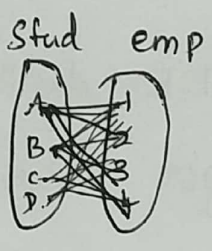
→ self join

→ equi join

→ Outer join

→ left join

→ Right join



Cross join:

→ No relation (or) connection between tables.

→ It check with every data.

→ If there is no connection with table also it can fetch data. by every data of one table checks with every data of another table.

Syntax:

1) Select table.col, table.col, table.col ....

from table1, table2

8) select table.col, table.col, ...
from table1 Cross join table2;

'as' keyword works in 'select' not in 'from' clause

Aliasing should be done in 'from' clause in joins.

Two table all data → select *

→ select stud.sname, stud.marks, emp.ename, emp.job
from stud, emp;

→ select s.sname, s.marks, e.ename, e.job
from stud s, emp e
where s.ename = 'ram';

→ select s.name, s.marks, e.ename, e.job
from stud s cross join empe
where s.sname = 'ram';

→ select d.dname, e.ename
from dept d, empe;

56 rows selected
14 ename × 4 dname
Smith ACC
Allen ACC
ACC
14
Smith Research
Rese...
14

Smith Sales
Smith Operations

W.Q.T display all the courses which 'abc' is a student will go through in full stack development.

```
Select stud.sname, cour.* 
from stud, cour
where stud.sname = 'abc';
```

W.Q.T display all student name who are done with an java

```
Select s.sname, c.cname
from studs, cour c
where c.cname = 'java';
```

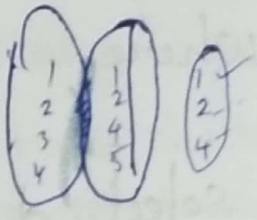
20/7

(*) Equi join: used to fetch common column (or) data

- It needs connection
- If we want another condition use 'and'.

Syntax:

```
Select table.col, table.col
from table1, table2
where table1.commoncol = table2.common col.
```



```
Select *
from stud s course c
where s.cid = c.cid;

Select S.Sname , C.cname
from stud S, cours C
where S.cid = c.cid;

Select e.ename , e.deptno, d.deptno, d.dname
from emp e, dept d
where e.deptno = d.deptno;
```

W.Q.T.F all the details of student and current course details.

fetch the column only if common col equals

cid cid
201 = 201
202 = 202

```
Select *
from stud s course c
where s.cid = c.cid;
```

W.Q.T display employee name & working location.

ename	dloc
Smith	Chicago
Allen	dallas
Ward	Newyork

```
Select e.ename, d.doc
from emp e, dept d
where e.deptno = d.deptno;
```

W.G.T display Smith, Allen, all the records along with deptname & location.

```
Select e.*, d.dname, d.loc
from emp e, dept d
where e.deptno = d.deptno and
      e.ename in ('SMITH', 'ALLEN');
```

e.* dname loc
Smith .... R dallas
Allen .... S chicago

W.G.T display all the details of Salesman, and manager. and their working location

```
Select e.*, d.loc
from empe, deptd
where e.deptno = d.deptno and
      e.job in ('SALESMAN', 'MANAGER');
```

e.* dloc
Job
Salesman D
Salesman N
manager C

21/7 Inner join: combination of equi joins

Syntax:

```
Select table.col, table.col
From table1 inner join (or) join table2 on
Table1.commoncol = Table2.commoncol;
```

A B
1 2 3 4
2 4

= ! > <

select e.ename, e.deptno, d.dname, d.deptno
from empe innerjoin deptd on
e.deptno = d.deptno;

ename	deptno	Dname	dno
Smith	20	R	20
Allen	30	S	30
ward	30		30

equi join

select e.ename, e.deptno, d.dname, d.deptno
from empe, deptd
where e.deptno = d.deptno;

ename	deptno	dname	deptno
Smith	20	R	20
Allen	30	S	30
Ward	30	S	30

select e.ename, e.deptno, d.dname, d.deptno
from empe innerjoin deptd on
e.deptno =! d.deptno;

ename	dno	dname	dno
Smith	20	A	10
Allen	30	A	10
Allen	30		20
Smith	20		30
Smith	20		40

Select e.ename, e.deptno, d.dname, d.deptno
from empe innerjoin deptd on
e.deptno =! d.deptno
where e.ename = 'SMITH';

ename	deptno	dname	dno
Smith	20	Acc	10
Smith	20	Sales	30
Smith	20	Operation	40

Select e.ename, e.deptno, d.dname, d.deptno
from empe innerjoin deptd on
e.deptno > d.deptno;

Smith	20	Sales	30
Smith	20	op	40
Allen	30	op	40

(Same example as equi joins)

W.Q.T display all the details of student 'abc' display rest of the courses still he left to do.

sid	sname	mark	cid
101	ram	98	202

cid	cname	Tid
201	java	ram
203	sql	...
204	html	...

```
Select S.*, C.*
from stud s innerjoin course c on
where S.cid != C.cid,
    where S.Sname = 'abc';
```

22/7

Self join:

emp emp

- It will joins itself
- create duplicate table
- Aliasing is mandatory
- common col should be there

Syntax:

```
Select  table.col, table.col
from    table t1, table t2
where   t1.common col = t2.common col;
```

W.A.Q.T. fetch name, job, salary and manager name of employee 'Smith'.

1) Select e1.name, e1.job, e1.sal, e2.ename
from emp e1, emp e2
where e1.mgr = e2.empno and e1.ename = 'SMITH';

2) Select e1.ename, 'Manger is' || E2.ename
from emp e1, emp e2
where e1.mgr = e2.empno and e1.ename = 'SMITH';

3) Select e1.ename, 'Manager is' || e2.ename
from emp e1, emp e2
where e1.mgr = e2.empno;

→ Same o/p as 2 but 14 rows selected.

1)

ename	Sal	job	ename
Smith	5000	clerk	Ford

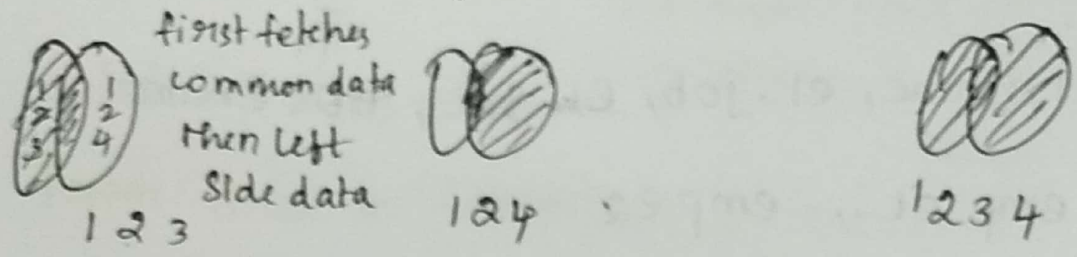
2)

ENAME	'MANAGER IS'	ENAME
SMITH	manager is	FORD

25/7

(connection is compulsory)

left join / Right join — Outer join



leftjoin:
```
Select table.col, table.col
from table1 leftjoin table2
on table1.common col = table2.commoncol;
```

Rightjoin:
```
Select table.col, table.col
from table1 fullRightjoin table2
on table.1 commoncol = table2.commoncol;
```

Outerjoin:
```
Select table.col, table.col
from table1 full outerjoin table2
on table.1 common col = table2.commoncol;
```

```
Select S.sname, S.marks, S.cid, C.cid, C.cname
from stud S leftjoin course c
on S.cid = c.cid;
```

```
select S.sname, S.cid, S.marks, C.cname, C.cid, C.cname
from stud s right join course c
on S.cid = C.cid;
```

```
select S.sname, S.marks, S.cid, C.cname, C.cid, C.cname
from studs outer join course c
on S.cid = C.cid;
```

W.Q.T. display all the details of all students display course name if they are attending any course currently.

```
select S.*, C.cname
from stud s left join course c
on S.cid = C.cid;
```

W.Q.T display all the student information who are attending some course and display course which are available in institute.

```
select *
from stud s right join course c
on S.cid = C.cid;
```

W.Q.T. display all the student details and display all the course details which are available in institute.

```
Select *
from stud s Outerjoin course c
on s.cid = c.cid;
```

Index: - 2 types fetching the data quickly.

- It can be used to create index only for column not tables.
- can create one index for one column.
- Same col multiple conditions.

To reduce cpu ~~load~~ loads.

Syntax:

```
Create Index Indexname on TableName
                          (columnname);
```

Eg:

```
Create Index jobs on emp(job);
```

(Backend details like time, col

```
Set autotrace on explain;
Set autotrace off explain;
        (or)
set autotrace off
```

Drop index indexname;

Drop index dep

Types:

clustered index – one table one index

Non-cluster index – one table more than one index.

2) cluster data stored in sortby, Non-clustes occupies more order so it takes less space. Space because it not storing in sortby order

3) cluster is faster than non-cluster

Non-cluster is slower than cluster

Set autotrace on explain;

Select * from emp

where job='MANAGER';

create index dep on emp(deptno);

Select * from emp

where deptno=10;

26/7

Normalization:

- It is used divide large table into smaller table.
- It will build a relation b/w two tables.
- It removes insertion anomaly
  deletion anomaly
  updation anomaly
- It minimize ~~removes~~ redundancy.

Example:

sid	sname	cname	cid	marks	tfin
101	x	Sql			
102	y	css		80.4	
103	z	Java		65.4	
101					

deleting anamoly

insertion anamoly

updation anamoly

sid	sname	cname	cid

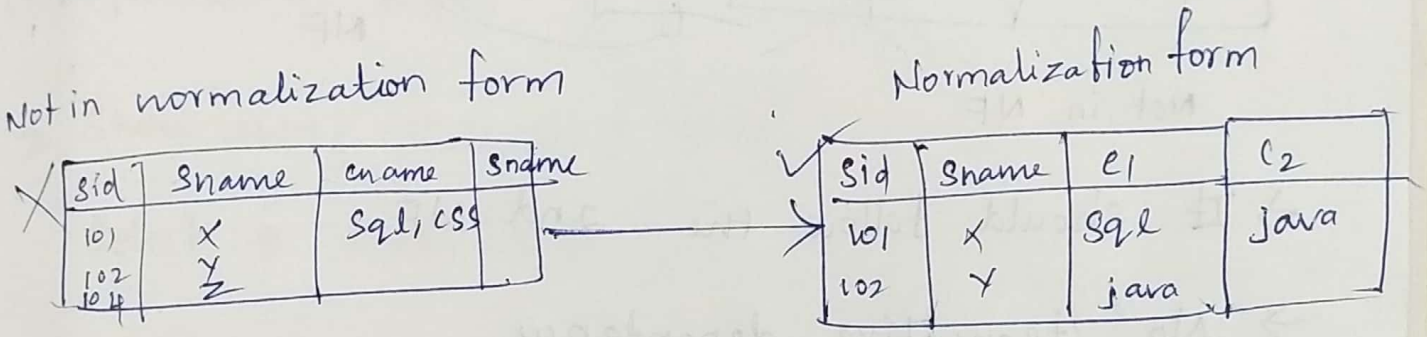
cid	cname	Tfin

→ Normalization divides larger table into smaller tables and links them using the relationship.

→ It will removes the insertion anamoly, updation anamoly & deletion anamoly.

→ It minimize the redundancy.

1st Normal Form : (1NF)

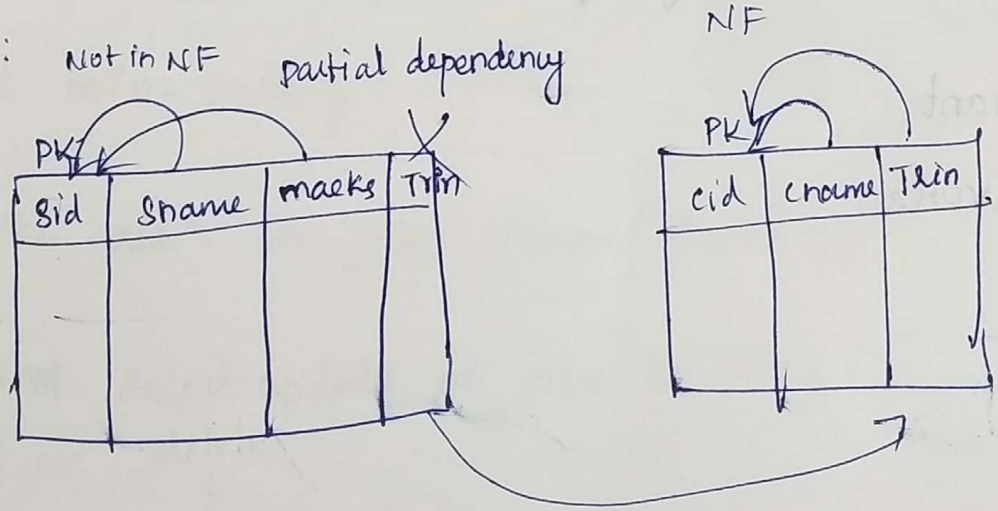


→ Single attributes (records)

→ Unique column names should be present

→ Order is not mandatory.

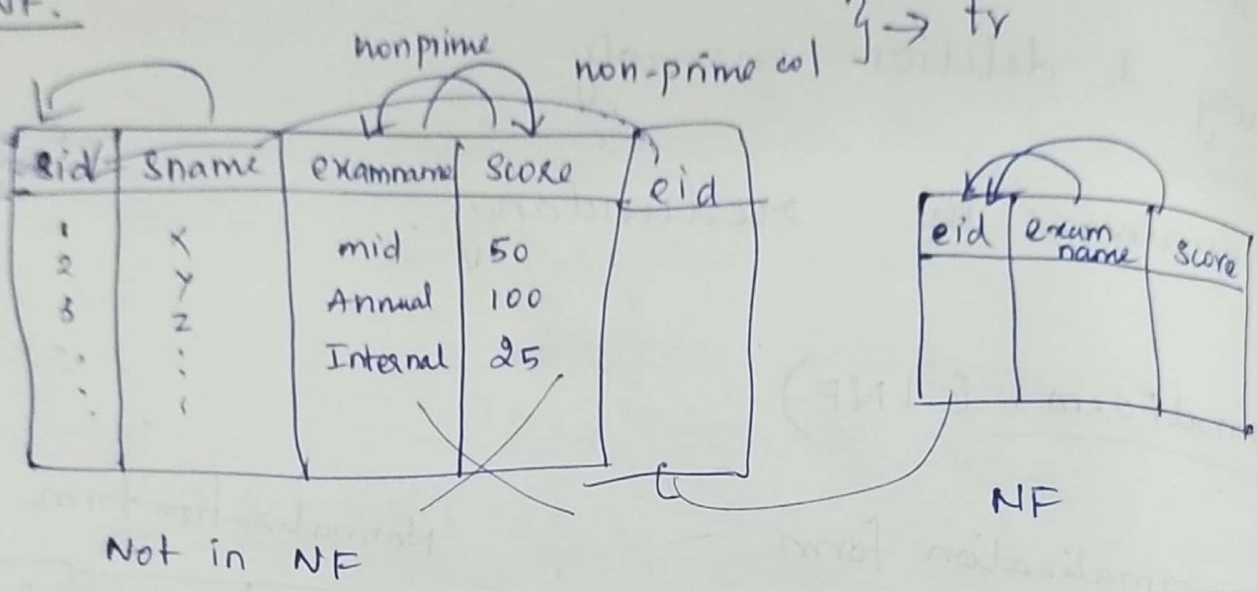
2NF:



→ It should follow the 1st normal form.

→ No partial dependancy.

3NF:



→ It should follow the 2nd NF.

→ No transitive dependency.

Non-peime column depending on each other is transitive dependency.

27/7

Data Control language:

→ Grant

→ Revoke

Syntax: (permission) (syntax, tablename, views)

1) Grant privilages on object to user;

2) Revoke privilages on object from user;

→ Select * from all_users; (to check users present in database)

→ grant select on emp to HR;

→ conn hr; (connecting hr)

Enter password: tiger

→ show user; (current user)

Select * from tab;

Select * from Scott.emp;

Select ename, job from Scott.emp;

grant update on ~~emp~~ jobs to ~~HR~~ scott;

update Scott.jobs
Set min_salary = 10000;

Revoke all on jobs from Scott;

grant select, update on jobs to HR;
delete
↓
all

Delete from scott.emp
where ename = 'SMITH';

delete from hr.jobs
where Job_id = 'MK_MAN';

Error: integrity constraint (Itz.emp_job_FK)

Grant: Grant: command used to provide privileg
to database object for users.

→ This command also allows user to grant permi
to other user.

Revoke: command is used to withdraw user
Privileges on database objeet if any granted.

To Create a new user:

Create user username identified by passwo

Eg:

Create user abc identified by 1234;

(you cannot create under any user
connect to System (admin) then create)

connabc; error.

Grant Connect to abc; → system user.

conn abc;

connected.

28/7

→ Grant all privileges to newuser;

→ grant all privileges to abc;

create table emp (eId number(4));

Table created.

Drop user username cascade;

(to delete user after connecting to system)

Drop user abc cascade;

Select * from all_user;

To unlock:

→ alter user username account unlock;

→ alter user username identified by password;

conn system;

Password: tiger

Set Operators: -

Records should match each other
column "
datatype "

(Unique values)

→ Union
→ Union all (alldata)
→ Intersect (common)
→ minus (Non-common)

→ Two tables
→ same tables
→ same data & datatype
→ diff data & datatype
→ No connection

Union:

```
Select deptno from emp
Union
Select deptno from dept;
```

(unique values)

same data, diff table

10
20
30
40

```
Select deptno from emp
union
Select deptno from emp;
```

Same table

```
Select ename from emp
union
Select dname from dept;
```

Union all

```
Select deptno from emp
union all
Select deptno from dept;
```

10
20
30
20
40
10

Intersect:

```
select deptno from emp
Intersect
select deptno from dept;
```

minus:

```
Select deptno from emp
minus
select deptno from dept;
```

No rows selected.

```
Select deptno from dept
minus
Select deptno from demp;
```

40

Views

```
create view V1
as
Select ename, empno, sal
from emp;
```

Error: insufficient privileges

```
conn system
Password: tiger
```

→ grant Create view to Scott;
→ conn Scott;
→ create view V1
as
Select ename, job, empno
from emp;

View created.

Select * from V1;
Select * from tab;

Views is Virtual table (DML command)

→ View is a resultant set of stored query.
→ Read only versus update.

Advantages of views:

→ To restrict the data access (security)
→ To make complex query easy.
→ To provide different views of same data

drop View Viewname;

Groupby: - to group the data
- to work with multi row fn.
- depends on select clause.

syntax: group by (col name given in select clause);

```
select job, count(*)
from emp
group by job;
```

having: to put condition
it can be used only with groupby

Syntax:

1) select col...
2) From tablename
3) where Condition
4) group by
5) having...
6) Order by col (asc/desc)

1) 2) 3) 4)

```
select job, count(*)
from emp
where ~~ename~~ = 'SMITH'; job in ('SALESMAN', MA
group by job;
```

```
Select job, deptno , count(*)
from emp
group by job, deptno;
having count(*) >= 2;
```