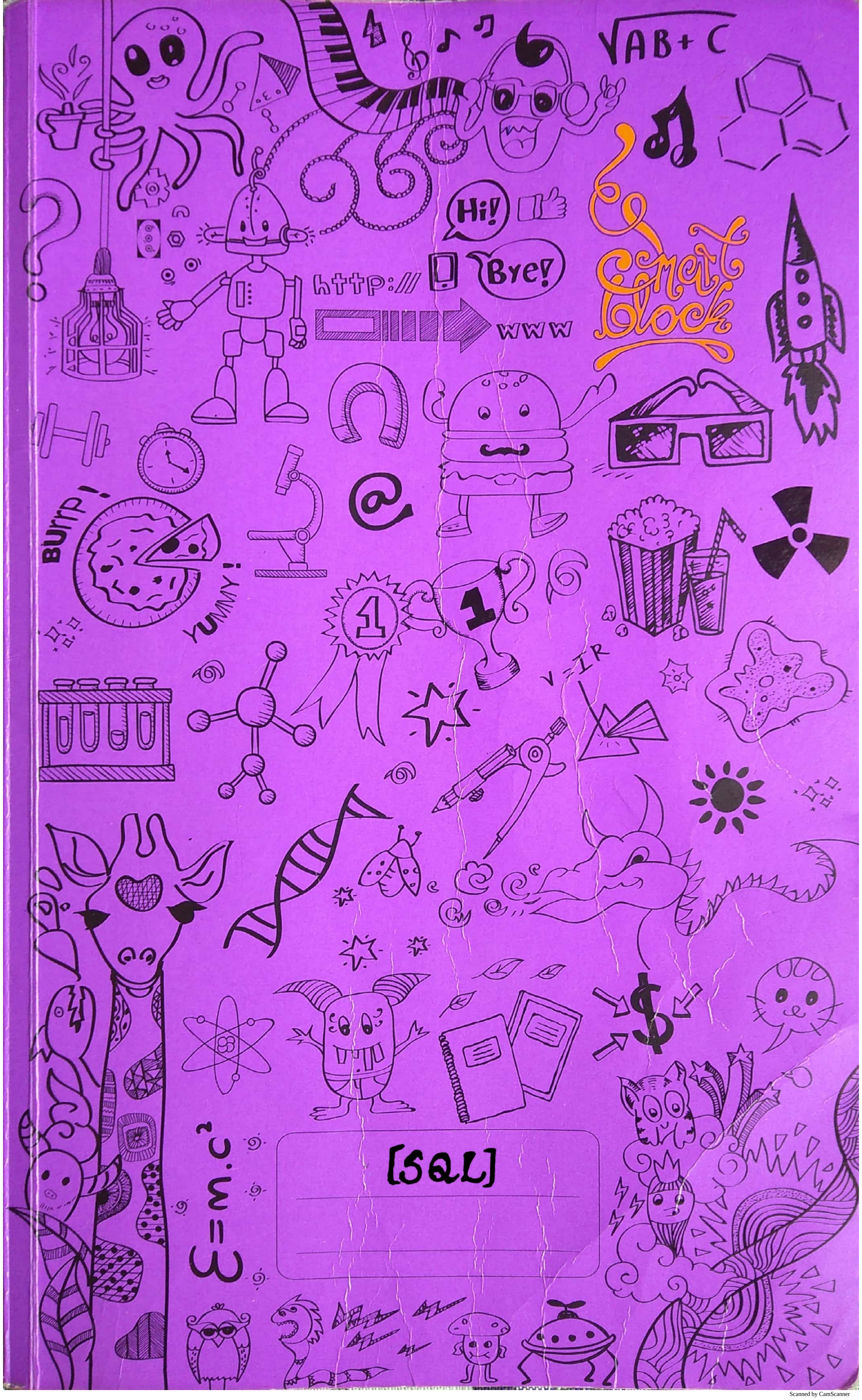


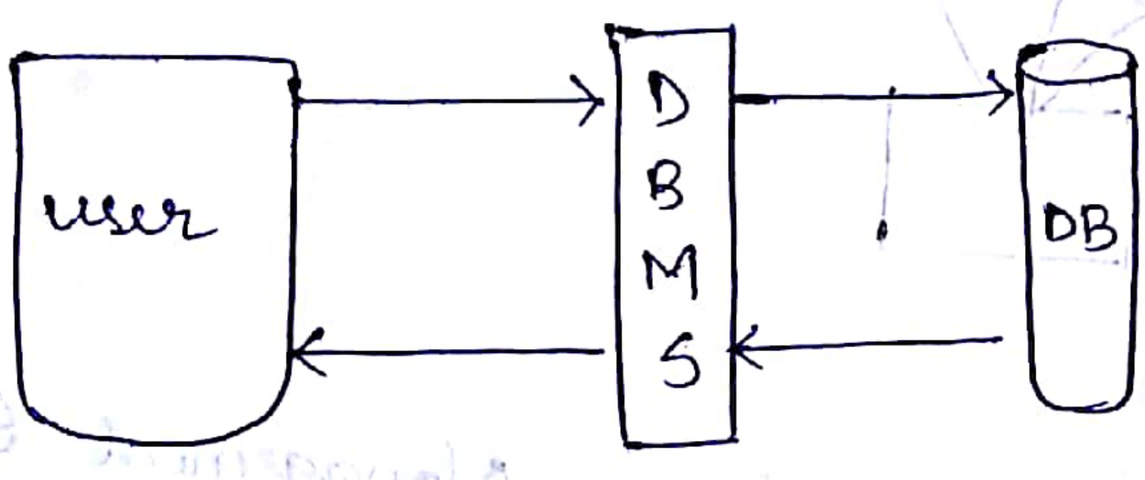
Scanned by CamScanner

Data :-

→ Data is a useful information and those datas are stored in centralized location called data base.

DBMS :(Data Base Management System) :-

→ DBMS will be responsible for all type of operation such as retrieving the information, fetching the information, modifying the data, inserting the data / removal of data.

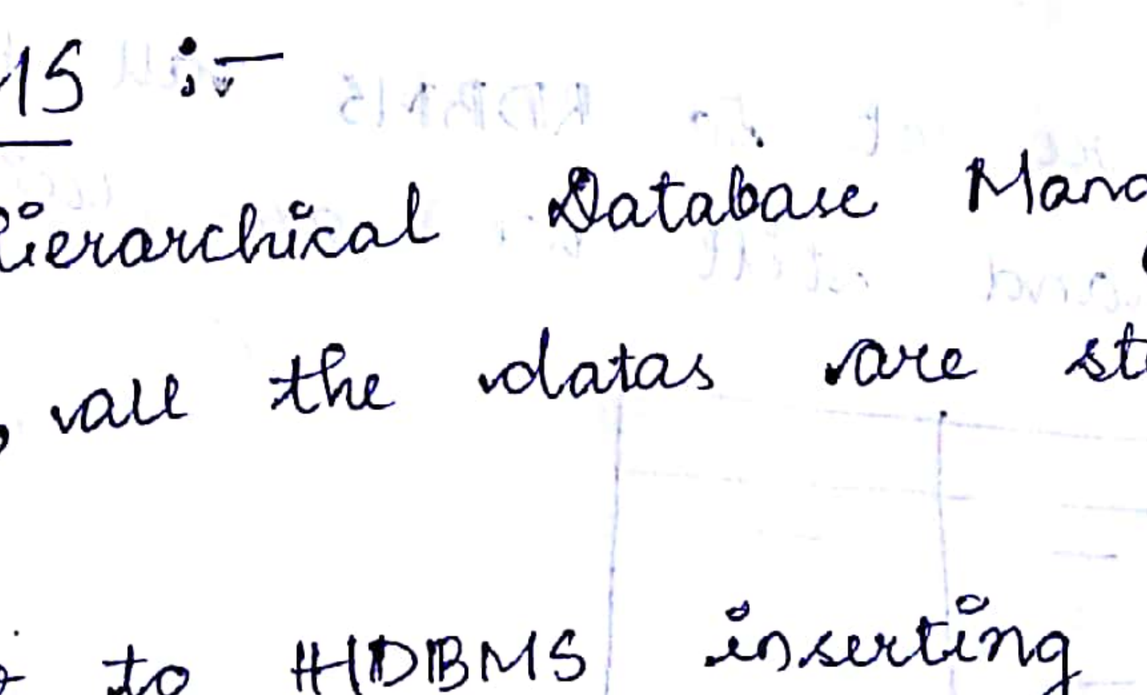


→ DBMS acts like intermediate between user and database.

Types of DBMS :-

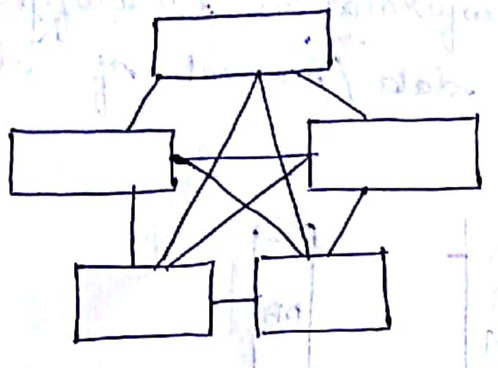
(i) HDBMS : (Hierarchical Database Management System)

- In HDBMS, all the datas are stored in tree structure.
- With respect to HDBMS inserting a data is easy but retrieval of data is time consuming.



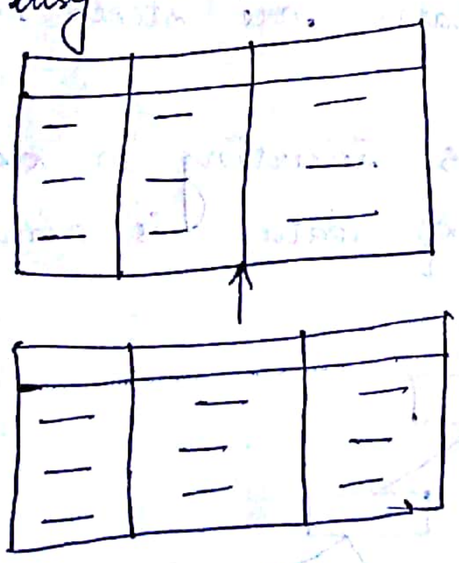
(ii) NDBMS : (Network Database Management System)

- In NDBMS all the datas are stored in tree structure, but are interconnected in network passion.
- So with respect to NDBMS retrieving a data is easy but insertion is time consuming.



iii) RDBMS : (Relational Database Management System)

- In RDBMS all the datas are stored in tables and those tables might have some relation.
- With respect to RDBMS all the operations are easy and still we are using RDBMS.



SQL : (Structured Querry Language)

→ SQL is a language used to interact with DBMS or database in the form of query.

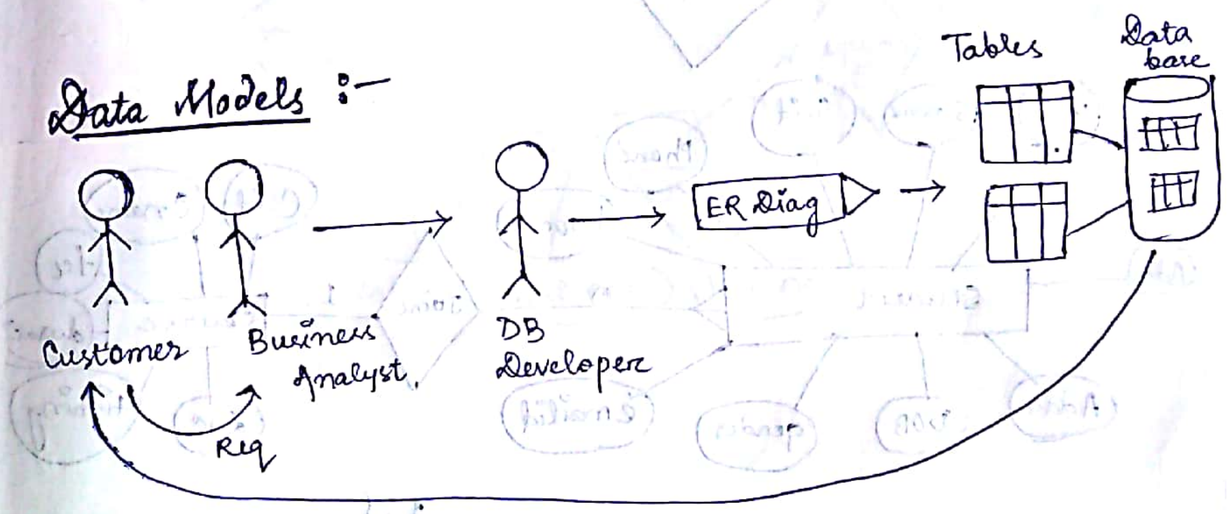
Database :—

- Oracle
  - 10G (older version)
  - 11G, 12C } newer version (paid, easy)
- My SQL
- MongoDB
- DB 2
- IBM2
- Sky SQL

SQL → Language

MySQL → Database

Data Models :—



Database Collection Process :—

→ Collect all the requirements from the customer.

→ While collecting the requirements specifying entity and attributes.

→ Entity might be living or nonliving things.

→ Attributes are features of those entity.

Scanned by CamScanner

Entity (table name)

Student
- S.name
- Reg no
- DOB
- Ph no
- Branch
- Aadhar
- gender
- e-mail
- YOJ

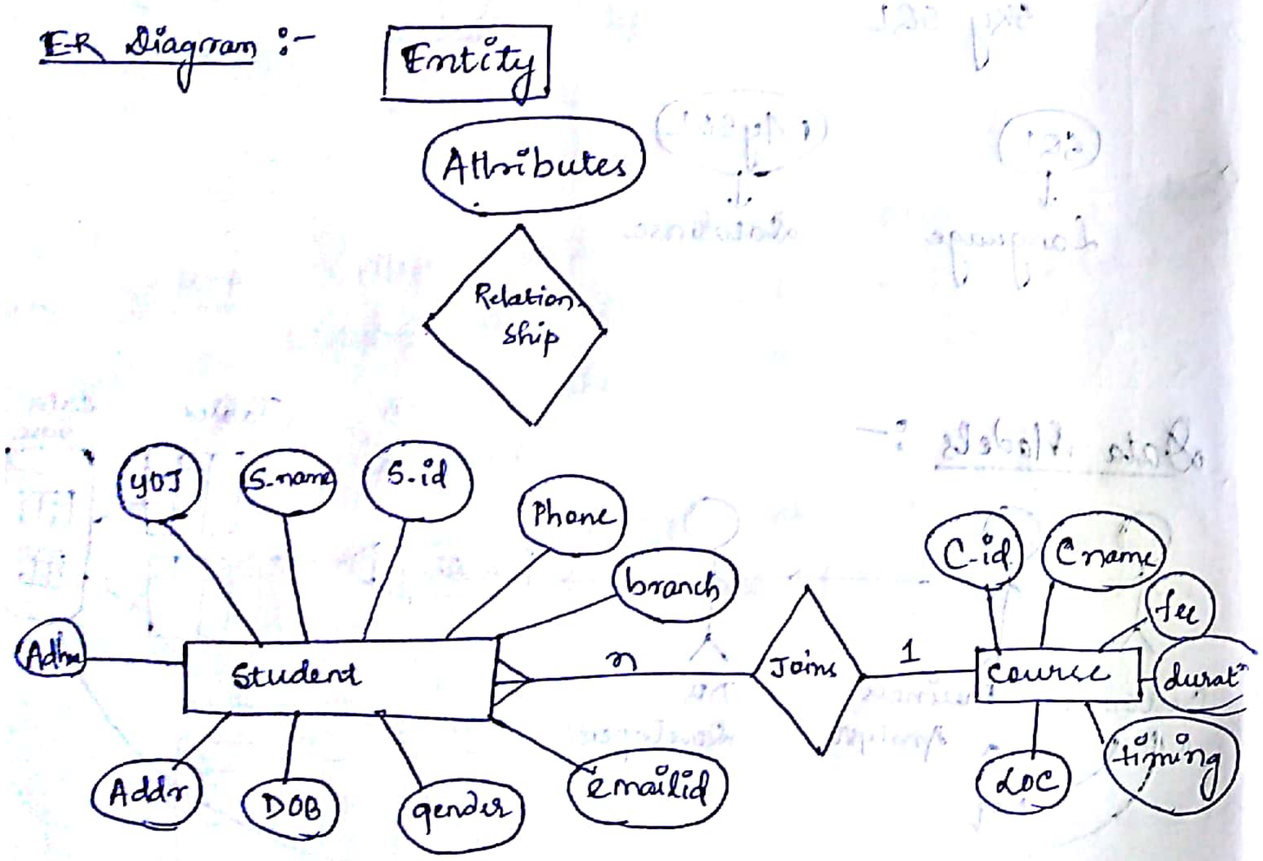
Course
- C-id
- Cname
- duration
- fees
- f-name
- Timing
- Location

Employee
- emp-id
- e-name
- Sal
- DOB
- Ph no
- DOJ
- JOB
- emailid
- dept no.

Department
- Dept no.
- Dept name
- no. of emps
- TL
- Mgr
- Location.

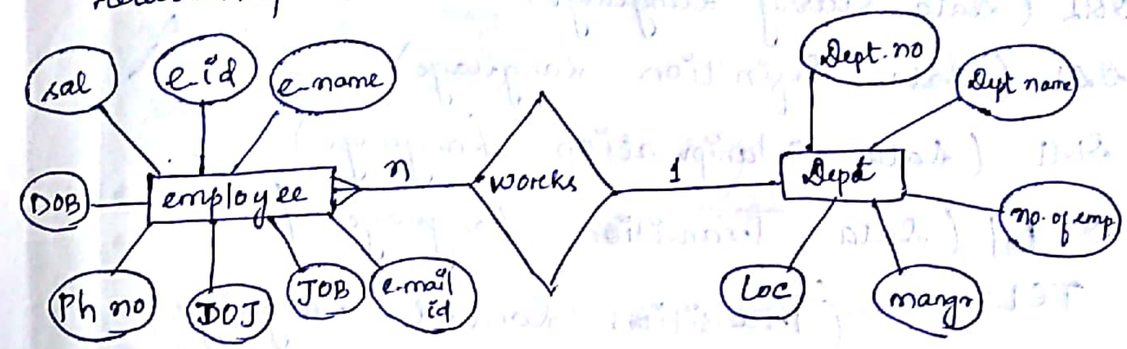
attributes (Column name)

E-R Diagram :-

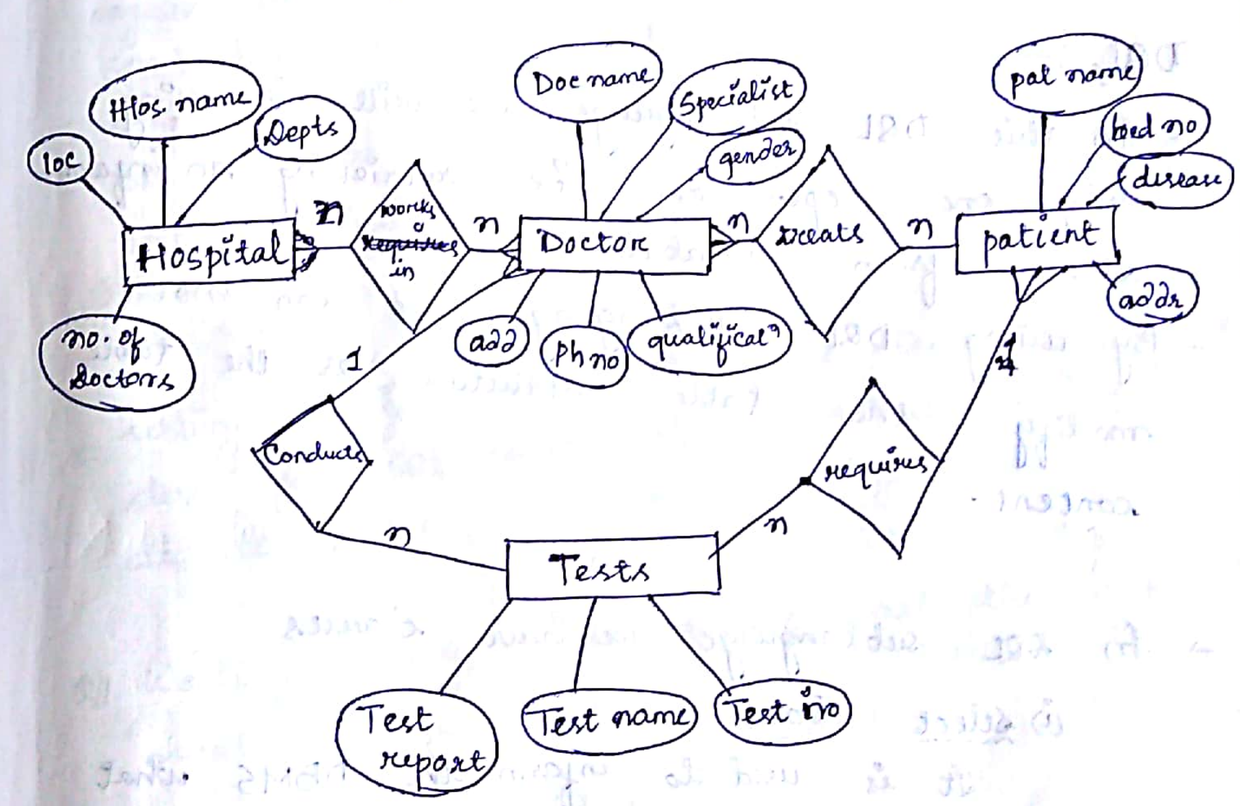


1 : 1
1 : n
n : 1 ✓
n : n

Q/ Write an E-R diagram for employee & department relationship.



Q/ Write E-R Diagram for hospital Management.



Sub Languages of SQL :-

- DQL (Data Query Language) - 90 %
- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- DTL/ (Data Transition Language)/
  TCL (Transition Control Language)
- DCL (Data Control Language)

DQL :-

- In this DQL sublanguage we will be using only one operation i.e retrieving an information from database.
- By using DQL sublanguage we can not modify the table structure or the table content.

→ In DQL sublanguage we have clauses.

(i) select clause
It is used to inform the DBMS what all data to be retrieve.

(ii) from clause
It is used to inform the DBMS from which table we have to retrieve the information.

```
select * from tab ;
```

The above query retrieves all the existing table information.

- * means retrieve everything.
- tab is a keyword to display existing table names.
- All the SQL queries should end with ';'.

```
SQL> show pagesize ;
pagesize 14
SQL> show linesize ;
linesize 80
SQL> set pagesize 150 ;
SQL> set linesize 150 ;
SQL>
SQL> select * from emp ;
```

```
SQL> set pages 150 lines 150 ;
```

setting pagesize and linesize is temporary if we close out SQL editor these are set back to default values.

Q) Write a query to retrieve all the information about employee.

```
SQL> select * from emp ;
```

Q) Write a query to retrieve all the information from department table.

```
select * from dept ;
```

Scanned by CamScanner

Employee Table :-

EMP NO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

Department Table

DEPTNO DNAME LOC

Note : SQL language is not a case sensitive language. We can use both upper case and lower case to write a query.

→ By using select clause we can perform projection, selection and joins.

(i) Projection :-

Projection means selecting column wise.

- By using select clause we can achieve projection.
- By using select clause we can decide which column to display and which column not to display.

Querry

SQL> SELECT ENAME FROM EMP ;

By the above querry we are informing two things to DBMS (i) to select the column ENAME (ii) to reject all other columns.

Q/ Write a query to retrieve employee name and their salaries.

SQL> select ENAME , SAL ~~from~~ FROM EMP ;

Aliasing the Column Name :-

→ By using 'as' keyword we can alias a column name.

SQL> select ename as emp-name from emp ;

Q/ Write a query to retrieve ename and sal while retrieving alias, ename as emp-name and sal as salary.

SQL> select ename as emp-name , sal as salary from emp ;

SQL> SELECT ENAME EMP-NAME FROM EMP ;

EMP-NAME

⋮

While aliasing 'as' keyword is optional. if we miss as keyword immidiate next word will be considered as alias name.

Q/ Write a query to retrieve ename while retrieving alias it as emp name.

SQL> select ename as "emp name" from emp ;
(we have to put double quotes)

Emp name

⋮

We have to use double quotes for alias name in two situations.

(i) if the alias names are more than one word.

(ii) if we want to maintain the same case.

Scanned by CamScanner

Write a query to check what all jobs are there is a company.

```
SQL> select job from emp;
JOB
salesman
Manager
Manager
Clerk
:
```

```
SQL> select distinct job from emp;

JOB
Clerk
Salesmana
:
```

→ Distinct is a keyword we can use the column name with same clause to retrieve each value only once.

```
SQL> select distinct job, distinct ename from
     emp;
     select distinct job, distinct ename
     *
     error.
```

Spool Command :-

```
SQL> set pages 150 lines 150;
SQL> spool D:\sql2\1.txt
SQL> select *
     from emp;

SQL> select *
     from dept;

SQL> spool off;
```

→ While using the spool command the path should not have a spaces.

Arithmatic operator :-

+
-
*
/

select
x from

→ Arithmatic operators can be used in select clause but not in from clause.

Q/ Write a query to retrieve employee name and salary while retrieving increased salary by 300.

```
SQL> select ~~salary~~ ename, sal, sal+300 from emp;
```

Q/ Write a query to retrieve employee name and sal while retrieving deduct salary by 200.

```
SQL> select ename, sal, sal-200 from emp;
```

Q/ Write a query to retrieve employee name, monthly salary and annual salary.

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

Editor :

```
SQL> ed
     select ename, sal "Monthly Sal", (sal*12) "Annual Sal"
     from emp;
SQL> / ↵
```

Scanned by CamScanner

Q// Write a query to retrieve employe name, monthly salary and annual salary with bonus of 600 for every half year.

```
SQL> select empname , sal , (sal*12) + (600*2)
     from emp ;
```

Literals :-

→ Literals means data it might be there in table or outside the table.

- There are 3 types of literals.
  - (i) String literal - Name, Addr, email id, loc
  - (ii) Number literal - Phone no, age, Adhar no.
  - (iii) Date literal - DOB, DOJ, MFG Date, Expire date.

→ Number literals we can use directly but string and date literals should be enclosed with double quotes.

Properties of Literals :-

→ If we are using any literals in select clause, the same literal will be executed for each and every row of given table.

```
SQL> select 'good morning' from emp ;
```

o/p :- 'GOODMORNING'
good morning
⋮
14 times

→ We have some situation where we need to use string literal to ~~use~~ increase readability.

Concatenation :-

→ This means joining one or more entities.
→ Entities might be literals or column names.
→ For joining two entities we use parallel lines.

```
X SQL> select 5||8
       from emp ;
```

5|
58
58
⋮
14 times.

```
X SQL> select ename || sal
       from emp ;
```

ENAME || SAL
SMITH5000

```
X SQL> select ename || ' ' ||
       sal from emp ;
```

ENAME||' '||SAL
SMITH 5000

```
X SQL> select 'sql' || 'plus'
       from emp ;
```

'SQL' ||
sqlplus
⋮

```
X SQL> select 'sql' || 'plus'
       from emp ;
```

'SQL' ||
sql plus
⋮

Scanned by CamScanner

SQL> select ename || ' EARNS ' || sal
from emp ;

ENAME || ' EARNS' || SAL

SMITH EARNS 800

Q. Write a query to retrieve as 'SMITH WORKS AS CLERK'

SQL> select ename || ' WORKS AS ' || JOB
from emp ;

Q. Write a query to display as below :
SMITH WORKS AS CLERK AND EARNS 800 Rs SALARY

SQL> select ename || ' WORKS AS ' || JOB || ' AND '
EARNS ' || SAL || ' RS SALARY '
from emp ; ;

- SQL> select * , ename
from emp ;

ofe error

- SQL> select ename sal
from emp ;

SAL

- SQL> select '*'
from emp ;

*
⋮
14 times.

- SQL> select 5||9
from dept ;

5|

59
59
59
59

- SQL> select 6+9
from dept ;

6+9

15
15
15
15

- SQL> select '5||4'
from emp ;

'5||
5||4
⋮
14 times

- SQL> select '6+3'
from emp ;

'6+

6+3
⋮
14 times.

- SQL> select sql+9
from dept ;
(error)

SQL> select 'sql' +9
from dept ;
(error)

- SQL> select 6273+7282
from emp ;

6273+

13555
⋮
14 times.

- SQL> select 6273 +7282
from dual ;

6273+7282

13555

- dual is a special table where we can use whenever we are dealing only with the literals to display output only once.

'5' || '4'

54
54

A+9

Selection :-

We use concept of selection to display particular rows.

projection - col
selection - rows

example :

Student

id	Name	perc.
10	Sukant	75.67
11	Shubham	84.61
12	Shahuda	95.86
13	Laxmi	78.78

```
select name
from student
where id = 12 ;
```

o/p Shahuda

Q/ Write a query to display all managers name.

```
SQL> SELECT ENAME
     FROM EMP
     WHERE JOB = 'manager' ;
```

no rows selected.

```
SQL> select ename
     from emp
     where job = 'MANAGER';
```

Note

- SQL is not a case sensitive language
- data should be same as given in table. So it is case sensitive.

Q/ Write a query to display all information of 'SCOTT'

```
SQL> select *
     from emp
     where ename = 'SCOTT' ;
```

Q/ Write a query to display emp name, job and hire date who join on 9th June 81.

```
SQL> select ename , job , hiredate
     from emp
     where hiredate = '09-JUN-81' ;
                      or
                      '09-jun-81' ;
```

Q/ Write a query to display emp name who joined on 23rd jan 1982.

```
SQL> select ename
     from emp
     where hiredate = '23-JAN-1982' ;
```

String → ename, job etc } case sensitive

Date } format sensitive [DD-MON-YY]

Q/ Write a query to display empname who work in dept no. 20.

```
SQL> select ename
     from emp
     where DEPTNO = '20' ;
```

Scanned by CamScanner

Logical operator :-

1. AND
2. OR
3. NOT

AND

i/p	i/p	o/p
T	T	T
F	T	F
T	F	F
F	F	F

Q) Write a query to display emp name, job, sal of all clerk who work in dept 10.

```
SQL> select ename, job, sal
     from emp
     where job = 'CLERK'
     AND deptno = 10 ;
```

OR

i/p	i/p	o/p
T	T	T
F	T	T
T	F	T
F	F	F

Q) Write a query to display empname, deptno of those people who belongs to deptno 10 and 20.

```
SQL> select ename, deptno
     from emp
     where deptno = 10
     OR
     DEPTNO = 20 ;
```

NOT

i/p	o/p
T →	F
F →	T

Relational Operator :-

< <=
> >=
= !=

- These relational operators can be used in where condition but not in select clause.

* Write a query to retrieve all the ~~queries~~ employees who is earning less than 3000.

```
SQL> select *
     from emp
     where sal < 3000 ;
```

* Write a query to retrieve all the employees whose salary is atleast 1600.

```
SQL> select *
     from emp
     where sal >= 1600 ;
```

* Write a query to retrieve all the employees who does not belongs to dept 20.

```
SQL> select *
     from emp
     where deptno != 20 ;
```

Scanned by CamScanner

→ We can achieve selection using where condition.
→ Where condition will be executed for each and every row of given table, if the condition satisfy, it will select the row or else it will reject the row.

Order by Clause :-

→ By using order by clause we can rearrange the output in ascending / descending order.
→ By default it will be in ascending order. if we want to display in descending order we have to write the keyword 'desc'.

* select *
  from emp
  order by sal ;

* select *
  from emp
  order by sal desc ;

* select *
  from emp
  order by ename ;

* Write a query to display ename and sal while retrieving alias ename as emp-name and sal as salary while retrieving order with respect to sal.

SQL> select ename emp-name , sal salary
  from emp
  order by sal /salary ;

SQL> select *
  from emp
  order by deptno , sal ;

→ We can use more than one column in order by clause. in that case first it will order with respect to 1st column name and within that it will order w.r.t 2nd column name.

Note : If you are using 2 column names in order by clause. 1st column should have more duplicate values to see the difference in 2nd column.

SQL> select *
  from emp
  order by deptno desc , sal desc ;

* Write a query to retrieve employee name and salary for all the employees who is earning exactly 3000 and while retrieving alias ename as emp-name and sal as salary.

SQL> select ename emp-name , sal salary
  from emp
  where sal = 3000 ;

Note :- Alias names are ~~alondir~~ allowed in order by clause but not in where condition.

Scanned by CamScanner

* Write a query to retrieve all salesman details while retrieving arrange the records w.r.t salary.

```
SQL> select salesman
     from emp
     order by sal ;
```

```
SQL> select *
     from emp
     where job = 'salesman'
     order by sal ;
```

* Write a query to retrieve all the managers who belongs to dept 30.

```
SQL> select *
     from emp
     where job = 'manager' and deptno = 30 ;
```

* Write a query to retrieve all the managers and analyst.

```
SQL> select *
     from emp
     where job = 'MANAGER' ~~and~~ OR job = 'ANALYST' ;
```

* Write a query to retrieve all the ~~query~~ employees who joined in year 81.

```
SQL> select *
     from emp
     where hiredate >= '01-JAN-81'
     and hiredate <= '31-DEC-81' ;
```

* Write a query to retrieve the information of employee whose eid is 7900, 7902, 7698, 7788.

```
SQL> select *
     from emp
     where empno = 7900
       or empno = 7902
       or empno = 7698
       or empno = 7788 ;
```

Special Operators :-

(i) In operator :-

We go for in operator to eliminate or conditions.

```
SQL> select *
     from emp
     where empno in (7900, 7982, 7698, 7788) ;
```

* Write a query to retrieve all employees whose salary in the range of 1600 - 3000.

```
SQL> select *
     from emp
     where sal >= 1600 AND sal <= 3000 ;
```

(ii) Between and operator :-

- We go for between and operator whenever we have range of values.
- It is inclusive operator.

```
SQL> select *
     from emp
     where sal between 1600 and 3000 ;
```

Scanned by CamScanner

* Write a query to retrieve all the empname whose name start with the letter 'S'.

```
SQL> select ename
     from emp
     where ename like 'S%';
```

(iii) Like operator :-

- It is also known as pattern matching operator or wildcard operator.
- Whenever we know some part of a content, this is used.

_ → 1 char

% → 0 char / n char

* Write a query to retrieve all the employee name whose name ends with 'N'.

```
SQL> select ename
     from emp
     where ename like '%N';
```

* Write a query to retrieve all the employee whose job contains 'man' in it.

```
SQL> select *
     from emp
     where job like '% MAN %';
```

* Write a query to retrieve all the employees whose name second character is 'I'.

```
SQL> select *
     from emp
     where ename like '_I%';
```

* Write a query to retrieve all emp names whose name is having 'A' as 3rd character.

```
SQL> select *
     from emp
     where ename like '__A%';
```

* Write a query to retrieve all employee name whose name last but one character is 'E'.

```
SQL> select *
     from emp
     where ename like '%E_';
```

* Write a query to retrieve all the employees whose name is having exactly 4 characters in it.

```
SQL> select *
     from emp
     where ename like '____';
```

* We can't combine 2 special operators.

(iv) is null operator :-

We go for is null operator whenever we want to retrieve information based on null value.

Whenever we are dealing with null value we have some condition.

- null ≠ null
- null ≠ 0
- null ≠ —

- null is a unpredictable value.

```
SQL> select *
     from emp
     where comm = null;
```

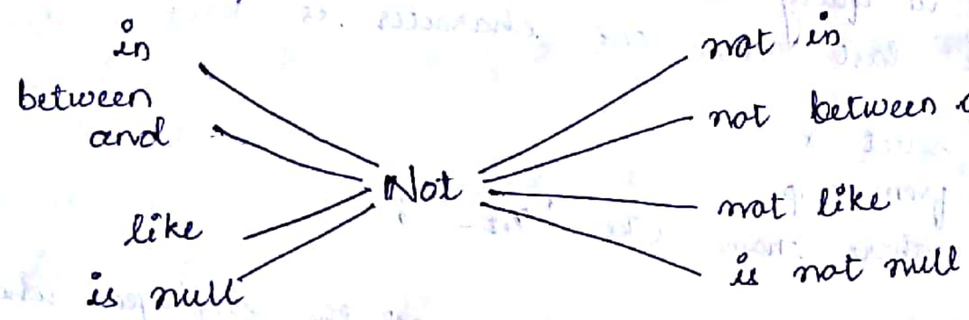
×

```
SQL> select *
     from emp
     where comm is null;
```

Scanned by CamScanner

* Write a query to retrieve all the employees who don't have manager.

```
SQL> select *
     from emp
     where mgr is null ;
```



* Write a query to reject all the employees who belongs to dept 10 and 30.

```
SQL> select *
     from emp
     where deptno not in (10,30) ;
```

* Write a query to reject all the employees whose salary is in a range of 1000 to 2000.

```
SQL> select *
     from emp
     where sal not between ~~and~~ 1000 and 2000 ;
```

* Write a query to retrieve all the employees whose name does not start with 'J'.

```
SQL> select *
     from emp
     where ename not like 'J%' ;
```

* Write a query to retrieve all the employees who are getting some commision.

```
SQL> select *
     from emp
     where comm is not null ;
```

* Write a query to retrieve all the managers who belongs to dept 20 and 30.

```
SQL> select *
     from emp
     where job = 'MANAGER' and deptno in (20,30) ;
```

* Write a query to retrieve all the employees who are getting some commision with their degn. neighter manager not analyst.

```
SQL> select *
     from emp
     where comm is not null and job not in
     ('MANAGER', 'ANALYST') ;
```

* Write a query to retrieve all the employees whose job is manager and they don't have manager.

```
SQL> select *
     from emp
     where job = 'MANAGER' and mgr is null ;
```

* Write a query to retrieve all the employees whose total salary is more than 2000.

```
SQL> select *
     from emp
     where sal + comm > 2000 ;

SQL> select *
     from emp
     where sal + comm > 2000 or sal > 2000 ;
```

Scanned by CamScanner

* Write a query to retrieve all the manager whose name does not start with 'B' and 'C'.

```
SQL> select *
     from emp
     where job = 'MANAGER'
     and ~~where~~ ename not like 'B%'
     and ename not like 'C%'
```

* Write a query to retrieve all the employees whose first character of the salary is 2.

```
SQL> select *
     from emp
     where sal like '2%' ;
```

* Write a query to retrieve all the employees whose annual salary is ending with zero.

```
SQL> select *
     from emp
     where (sal*12) like '%0' ;
```

* Write a query to retrieve odd salaries.

```
SQL> select *
     from emp
     where sal/2 like '%.%' ;
```

* Write a query to retrieve even salary.

```
SQL> select *
     from emp
     where sal/2 not like '%.%' ;
```

* Write a query to retrieve all employees who joined in year 81.

```
SQL> select *
     from emp
     where hiredate like '%81' ;
```

* Write a query to retrieve all employees who joined in december.

```
SQL> select *
     from emp
     where hiredate like '%DEC%' ;
```

* Write a query to retrieve all the salesman whose name last but one character is E and having salary exactly 4 characters in it.

```
SQL> select *
     from emp
     where job = 'SALESMAN'
     and ename like '%E_'
     and sal like '____' ;
```

Functions :-

→ Functions are predefined actions.

→ There are 2 types of functions.

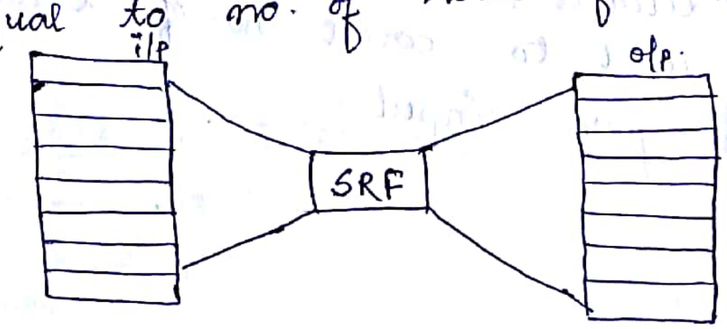
(i) Single row function

(ii) Multi row function.

(i) Single row function :-

→ This means for each and every row the corresponding output will be provided.

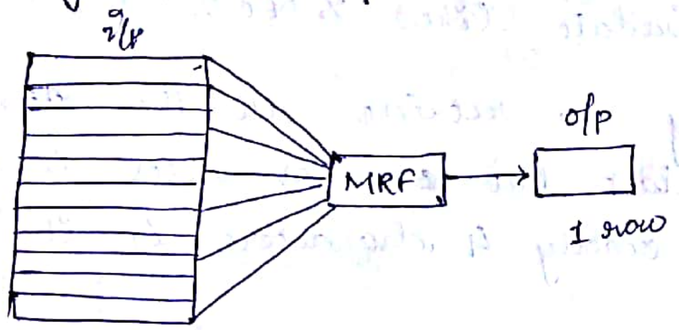
→ In S.R.F no. of rows of input will be equal to no. of rows of output.



Scanned by CamScanner

(ii) Multi row function :-

→ It takes multiple inputs but always provides only one output.



Single Row Function :-

ⓐ Case Manipulation

1. lower
2. upper
3. initcap

SQL> select lower ('GOOD MORNING'), upper ('good morning'), initcap ('good morning')
from dual ;

SQL> select lower (ename), upper (ename), initcap (ename)
from emp ;

→ initcap converts every word, first letter in upper case and remaining in lower case.

ⓑ Character Manipulation

1. length (Literals / Column)

→ It is used to count no. of characters in the given input.

SQL> select length ('sql')
from dual ; → 3

SQL> select length ('sql course')
from dual ; → 9

SQL> select ename, length (ename)
from emp ;

* Write a query to retrieve all the employee names whose name is having exactly 4 characters in it.

SQL> select *
from emp
where length (ename) = 4 ;

② Concat function

concat (arg1, arg2)

arg 1, arg 2 > literals / Column

* SQL> select concat (5,8)
from dual ;

CO
58

* SQL> select concat ('sql', 'plus')
from dual ;

CONCAT('
sql plus

* SQL> select concat (ename, sal)
from emp ;

CONCAT(ENAME,SAL)
SMITH800

* SQL> select ~~ename~~ concat (ename, ' ', sal)
from emp ;

error.

→ For concat function, we can not pass more than one argument.

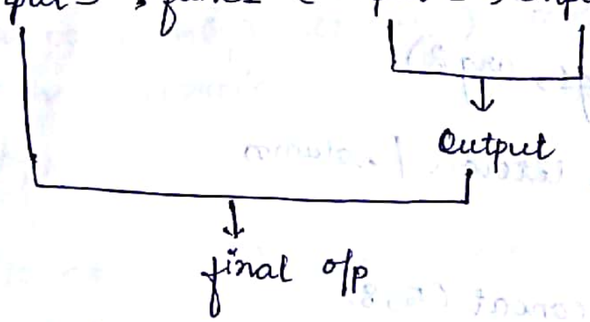
→ If we have situation that to use more than 2 arguments we have to go for nested function.

Nested function :-

→ Using one function within another function is known as nested function.

→ In that case always inner function executes first.

func 2 (Input 3 , funct (Input 1 , Input 2))



SQL> select concat (ename, concat (' ', sal))
from emp ;

CONCAT(ENAME,CONCAT.(' ',SAL))

SMITH 800

* Write a query to retrieve as below.

SMITH EARNS 800.

SQL> select concat (ename, concat ~~(' EARNS ' EA~~
(' EARNS ', *SAL')

from emp ;

* Write a query to retrieve as below.

SMITH WORKS AS CLERK

SQL> select concat (ename, concat (' WORKS AS ', *JOB*
from emp ;

* Write a query to retrieve as below.

Smith works as clerk and earns 800 Rs Salary.

~~SQL> select initcap~~

SQL> select concat (initcap (ename), concat (' works as ', concat
(lower (job), concat (' and earns ', concat (sal, ' rs
salary ')))))
from emp ;

③ Reverse function :-

Reverse (columnname)

→ It is used to reverse the content in the present column.

SQL> select reverse (ename)
from emp ;

SQL> select *
from emp
where reverse (ename) = ename ;

④ Replace function :-

Replace (arg1, arg2, arg3)

arg 1 → Lateral / colname
arg 2 → old char / string
arg 3 → new char / string

Scanned by CamScanner

* SQL> select replace ('java', 'j', 'm')
from dual ;

→ mava

* SQL> select replace ('java', 'a', 'c')
from dual ;

→ jcvc

* SQL> select replace ('c developer', 'c', 'java')
from dual ;

→ java developer

* SQL> select replace ('java', 'd', 'p')
from dual ;

→ java

* SQL> select replace ('java', 'a')
from dual ;

→ jv

Note :
In replace function if 3rd argument is missed out then old character will be completely removed from a string.

* Write a query to retrieve employee name and job while retrieving replace all the salesman with sales woman.

~~SQL> select ename, job from emp where replace (job, 'salesman', 'saleswomen')~~

SQL> select ename replace (job, 'salesman', 'saleswoman')
from emp ;

* Write a query to retrieve all the ename while retrieving S to #.

SQL> select ename, replace (ename, 'S', '#')
from emp ;

* Write a query to retrieve all the ename while retrieving S to # and A to @.

SQL> select ename, replace (replace (ename, 'S', '#'), 'A', '@')
from emp ;

ADAMS @D@M#

* Write a query to retrieve ename and the dept no. while retrieving replace dept 10 to 30 and 30 to 10.

~~SQL> select ename, replace (replace (deptno, '10', '30')~~

SQL> select ename, deptno, replace (replace (replace (deptno, 10, '@'), 30, 10), '@', 30)
from emp ;

⑤ Substring function :-

substr (arg1, arg2, arg3)

arg1 → Literal / colname
arg2 → start position (sp)
arg3 → Number of char to fetch from sp.

* select substr ('developer', 1, 5)
from dual ;

→ devel

* select substr ('developer', 3, 5)
from dual ;

→ velop

Scanned by CamScanner

SQL> select substr ('developer', 3, 1)
from dual ;

5

→ V

SQL> select substr ('developer', 5)
from dual ;

SUBST

→ Loper

Note :-

→ In substr function if the third argument is missed out, then it will fetch till end.

SQL> select substr ('developer', -4, 2)
from dual ;

1 2 3 4 5 6 7 8 9
d e v e l o p e r
-9 -8 -7 -6 -5 -4 -3 -2 -1

SU

op

→ In substr function second argument must be positive or negative value but third argument should be positive always.

Q/ Write a query to retrieve first and last character of employee name.

SQL> select ename, substr (ename, 1, 1), substr (ename, -1, 1)
from emp ;

Q/ Write a query to retrieve all the employee name whose name ends with 'S'.

SQL> select ename
from emp
where substr (ename, -1, 1) = 'S' ;

⑥ instring function :-

instr (arg1, arg2, arg3, arg4)

arg1 → Literal / colname
arg2 → searching char / String
arg3 → Start position
arg4 → which occurance.

SQL> select instr ('developer', 'e', 1, 1)
from dual ;

→ 2

SQL> select instr ('developer', 'e', 3, 1)
from dual ;

→ 4

SQL> select instr ('developer', 'e', 1, 3)
from dual ;

→ 8

SQL> select instr ('developer', 'x', 1, 1)
from dual ;

→ 0

SQL> select instr ('developer', 'e', 1, 9)
from dual ;

→ 0

SQL> select instr ('developer', 'e', 4, 2)
from dual ;

→ 8

→ In instr function if the searching character or searching occurance will not found the o/p will be 0.

SQL> select instr ('developer', 'e')
from dual ;

→ 2

Scanned by CamScanner

→ In instr function if the third and fourth argument is missed out by default it will consider as 1, 1.

SQL> select instr ('developer', 'e', -4, 1)
from dual ;

→ 4

1	2	3	4	5	6	7	8	9
d	e	v	e	l	o	p	e	r
-9	-8	-7	-6	-5	-4	-3	-2	-1

MAN

→ If the starting position is negative value, in case of substr function evaluation will take place from left to right but in instr function it is from right to left.

Solve using functions :-

Q) Write a query to retrieve all the employee name whose name is having 'L' as 3rd character.

SQL> select *
from emp
where substr (ename, 3, 1) = 'L' ;

or

SQL> select *
from emp
where instr (ename, 'L', 3, 1) = 3 ;

Q) WAQ to retrieve all the employees whose job contains 'MAN' in it.

SQL> select *
from emp
where substr (JOB, 1, 3) = 'MAN'
or substr (JOB, -3, 3) = 'MAN';

Or

SQL> select *
from emp
where instr (job, 'MAN', 1, 1) != 0 ;

Q) WAQ to retrieve all the employee names while retrieving first three characters of the name should be in lower case and remaining characters in upper case.

SQL> select concat (lower (ename, 1, 3)), upper(substr (ename, 4)))
from emp ;

→ smiTH

Q) WAQ to retrieve all the employee name while retrieving first and last character in upper case and remaining in lower case.

SQL> select concat (initcap (substr (ename, 1, length (ename) - 1)), upper (substr (ename, -1, 1)))
from emp ;

Q) WAQ to retrieve all the employees who joined in february.

SQL> select *
from emp
where substr (hiredate, 4, 3) = 'FEB' ;

Scanned by CamScanner

Q/ Write a query to retrieve all the employees who joined in year 81.

```
SQL> select *
     from emp
     where substr (hiredate , -2, 2) = 81 ;
```

or

8, 2

Q/ WAQ to count no. of ~~year~~ 'L' in each name.

```
SQL> select length (Ename) - length (replace (ename, 'L')),
     Ename
     from emp ;
```

ALLEN	AEN	No. of L
↓	↓	
5 −	3	= 2

Q/ WAQ to retrieve all the employee name, whose name is having exactly two L in it.

```
SQL> select *
     from emp
     where length (ename) - length (replace (ename, 'L'))
                                                = 2 ;
```

~~from~~

(c) Number Function :—

(i) Round function :

It is used to round up the decimal value. If the decimal value is from .1 to .4, it will round up to previous value, if the value is from .5 to .9, it will round up to next value.

```
SQL> select round (8.2) , round (8.5) , round (8.9)
     from dual ;
```

ROUND (8.2)	ROUND (8.5)	ROUND(8.9)
8	9	9

(ii) Trunc function :

It is used to eleminate the decimal value.

```
SQL> select trunc (8.2), trunc (8.5), trunc (8.9)
     from dual ;
```

TRUNC (8.2)	TRUNC(8.5)	TRUNC(8.9)
8	8	8

(iii) mod function :

- Mod function always retrieves a reminder.

```
SQL> select mod (12, 2)
     from dual ;
```

→ 0

```
SQL> select mod (12, 1)
     from dual ;
```

→ 0

```
SQL> select mod (12, 0)
     from dual ;
```

→ 12

Q/ WAQ to retrieve even salary.

~~SQL> select mod (sal, 2)~~

```
SQL> select ename, sal
     from emp
     where mod ( sal, 2) = 0 ;
```

Scanned by CamScanner

Q WAQ to retrieve odd salary.

```
SQL> Select *
     from emp
     where mod (sal , 2) ! = 0 ;
```

(iv) <u>nvl function</u> :

nvl (arg1 , arg 2)

→ It is used to replace null value with actual value.

→ In nvl function if arg1 is null, it provides 2.

→ If arg1 is not null, it provides same value.

same
↑
not null
↑
nvl (arg1 , arg 2)
comm , 600
↓
null

```
*SQL> select comm, nvl( comm, 600)
      from emp ;
```

(v) <u>nvl2 function</u> :

nvl (arg1 , arg 2 , arg 3)

→ It is used to replace null value as well as actual values.

null
↑
nvl (arg1 , arg 2 , arg 3)
comm 800
↓
not null

→ In nvl2 function if arg1 is not null, it provides 2.

→ If arg1 is not null, it provides 3.

```
SQL> select comm, nvl2(comm , 800, 1000)
     from emp ;
```

(vi) <u>Rownum , RowID function</u> :

→ Rownum is a sequence no. of each and every row.

→ RowID is a unique and sequence address.

→ RowID consists of 18 characters.

```
* SQL> select rownum , rowid
       from emp ;
```

ROWNUM	ROWID
1	AAAM5PAAEAAAAAgAAA
2	AAAM5PAAEAAAAAgAAB
3	⋮
4	⋮
⋮	

```
* SQL> select rownum , rowid
       from dept ;
```

```
* SQL> select *
       from emp
       where rownum <= 5 ;
```

[1st 5 record]

```
* SQL> select *
       from emp
       where rownum = 5 ;
```

~~[5th record only]~~

[no rows selected]

```
SQL> select *
     from emp
     where rownum = 1 ;
```

[1st record]

```
SQL> select *
     from emp
     where rownum <= &N ;

Enter the value for n : 2
old   3 : where rownum <= &N
new   3 : where rownum <= 2
```

⑥ Date function :

(i) sysdate ( )

→ It retrieves system date.

→ Always the default format will be DD-MON-YY.

```
SQL> select sysdate
     from dual ;

SYSDATE
---------
26-SEP-18
```

Arithmatic operation with sysdate function :-

```
SQL> select sysdate + 3
     from dual ;
```

→ 29-SEP-18

```
SQL> select sysdate - 10
     from dual ;
```

→ 16-SEP-18

```
SQL> select sysdate + 20
     from dual ;
```

→ 16-OCT-18

* multiplication /division is not possible.

```
SQL> select sysdate * 2
     from dual ;
```

→ error.

Q/ Write a query to retrieve enames, and their experience.

```
SQL> select ename, (sysdate - hiredate)
     from emp ;
```

Q/ Write a query to retrieve employee name and their experience with respect to years.

```
SQL> select ename, round((sysdate - hiredate)/365)
     from emp ;
```

Q/ Write a query to retrieve employees name and their experience in years whose having experience more than 35 years and while retrieving ascending order.

```
SQL> select ename, round((sysdate - hiredate)/365) "Exp"
     from emp
     where round((sysdate - hiredate)/365) > 35
     order by exp
```

(ii) systimestamp

→ It display with timestamp details.

```
SQL> select systimestamp
     from dual ;

SYSTIMESTAMP
-------------------------------------------
01-OCT-18 03.00.14.649000 PM +05:30
```

```
SQL> select substr(systimestamp, 11, 5)
     from dual ;

SUBST
-----
03.03
```

Scanned by CamScanner

(iii) to_char function :

→ This is required to convert any date into required format.

```
SQL> select to-char (sysdate , 'dd-mm-yy')
     from dual ;

TO-CHAR(
-01-10-18
```

```
SQL> select to-char (sysdate , 'dd/mm/yy')
     from dual ;
```

→ 01/10/18

```
SQL> select to-char ( sysdate , 'dd mm yy')
     from dual ;
```

→ 01 10 18

```
SQL> select to-char (sysdate , 'dd-mm-yyyy')
     from dual ;
```

→ 01-10-2018

```
SQL> select to-char (sysdate, 'yyyy')
     from dual ;
```

→ 2018

```
SQL> select to_char(sysdate , 'ddth mon yyyy')
     from dual
```

→ 01st oct 2018

```
SQL> select to_char(sysdate , 'dd month year')
     from dual ;
```

→ 01 october twentyeighteen

```
SQL> select to_char (sysdate , 'day')
     from dual ;
```

→ Monday.

Q) Write a query to retrieve all the employees who joined in seventh month.

```
SQL> select *
     from emp
     where to_char (hiredate , 'mm') = 07 ;
```

Q) Write a query to retrieve all the employees who joined in march.

```
SQL> select *
     from emp
     where to_char (hiredate , 'month') = 'march' ;
     or
     (hiredate , 'MON') = 'MAR'
```

Q) Write a query to retrieve all the employees who joined on sunday.

```
SQL> select *
     from emp
     where to_char (hiredate , 'day') = 'sunday' ;
```

Q) Write a query to retrieve all the employees who joined in year 81.

```
SQL> select *
     from emp
     where to_char (hiredate, 'yy') = 81 ;
```

Q) Write a query to retrieve all the employees who joined between 1983-1988 ;

```
SQL> select *
     from emp
     where to_char (hiredate , 'yyyy') between 1983 and
                                               1988 ;
```

Scanned by CamScanner

(v) Months_between function :

SQL> select months_between ('20-mar-2017', '14-apr-2018')
from dual ;

→ -12.806452

SQL> select ename, months_between (sysdate, hiredate)
from emp ;

Q// Write a query to retrieve all employee names and experience in months for all the employees who is having experience more than 440 months.

SQL> select ename, round (months_between (sysdate, hiredate))
from emp
where round (months_between (sysdate, hiredate)) >440 ;

SQL> select to_char (systimestamp, 'hh:mi')
from dual ;

TO_CH
03:46

SQL> select to_char (systimestamp, 'hh:mi:ss')
from dual ;

TO_CHAR (
03:46:36

(vi) Add_months : (~~sysdate~~

→ We can add the months for random date or for present date.

SQL> select add_months (sysdate, 6)
from dual ;

ADD_MONTH
02-APR-19

(vi) Last_day :

SQL> select last_day ('12-jun-2018')
from dual ;

LAST_DAY (
30-JUN-18

SQL> select last_day (sysdate)
from dual ;

LAST_DAY(
31-OCT-18

Q// Write a query to retrieve all the employees who joined in the last day of the particular month.

SQL> select *
from emp
where hiredate = lastday (hiredate) ;

(vii) Next_day :

SQL> select next_day (sysdate, 'monday')
from dual ;

NEXT_DAY(
08-OCT-18

Scanned by CamScanner

Q// WAQ to list all the employees who are working for department 10 or 20 with designation clerk or analyst with the salary 3 or 4 digits with an experience more than 20 years but not belongs to month of march, april and september and working for some manager and their employee no. is not ending with 88 or 56.

```
SQL> select *
     from emp
     where deptno in (10,20) and
       job in ('CLERK','ANALYST')
       ~~and sal >= 100 or sal <= 9999~~
       and length(sal) in (3,4)
       and round(sysdate - hiredate)/365 > 20
       and to_char(hiredate,'month') = ('mar','apr','sep')
       and mgr is not null
       and substr(empno,-2,2) not in (88,56);
```

Q// WAQ to retrieve all the employees whose first 2 characters of hiredate is equal to last 2 characters of salary.

```
SQL> select *
     from emp
     where substr(hiredate,1,2) = substr(sal,-2,2);
```

Q// WAQ to retrieve all the employees whose 10% of salary is equal to year of joining.

```
SQL> select *
     from emp
     where (sal * 0.1) = to_char(hiredate,'yy');
```

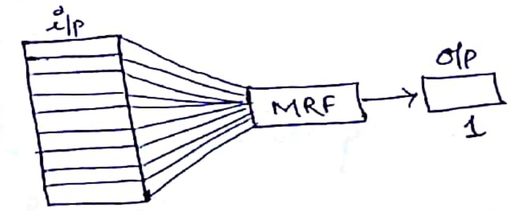
Q// WAQ to retrieve all the employee names while retrieving first 50% of names should be in lower case, second 50% of names should be in upper case.

```
SQL> select concat(lower(substr(ename,1,round(length
       (ename)/2))) upper(substr(ename,round(length
       (ename)/2)+1)))
```

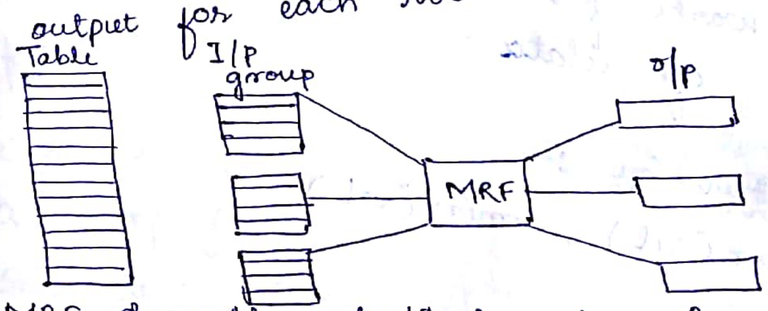
o/p
- SMITH
- ALLEN

(ii) Multi row Function :-

→ It takes multiple input but produces only one output.



→ It takes multiple roots as input and produces output for each root.



→ MRF considers duplicate value but not null values.

⊗ ~~Count Function~~ :

[Aggregate Function]

ⓐ Count Function :-

SQL> select count (empno)
from emp ;

→ 14

SQL> select count (ename)
from emp ;

→ 14

SQL> select count (hiredate)
from emp ;

→ 14

SQL> select count (comm)
from emp ;

→ 4

SQL> select count (job)
from emp ;

→ 14

→ Count function works for number, string as well as date type of data.

ⓑ Max and Min Function :-

SQL> select max (sal), min (sal)
from emp ;

→ 5000 800

SQL> select max (ename), min (ename)
from emp ;

→ WARD ADAMS

SQL> select max(hiredate), min (hiredate)
from emp ;

→ Max and min functions ~~are~~ works for number, string as well as date type character.

ⓒ Sum() and average() :-

SQL> select sum (sal), avg (sal)
from emp ;

SUM(SAL)	AVG(SAL)
29025	2073.21429

SQL> select sum (ename), avg (ename)
from emp ;

→ error

SQL> select sum(hiredate), avg (hiredate)
from emp ;

→ error

→ Sum and average function only works for number type of data.

Note

→ '*' is a special operator where we can use with count function.

Q1 Write a query to count number of analysts of a company.

SQL> select count (job)
from emp
where job = 'ANALYST' ;

Q2 Write a query to count no. of sales man in department 30.

SQL> select count (job)
from emp
where job = 'SALESMAN'
and deptid = 30 ;

Scanned by CamScanner

Q/ Write a query to retrieve highest salary with respect to manager.

```
SQL> select max(sal)
     from emp
     where job = 'MANAGER';
```

```
SQL> select max(sal), ename
     from emp
X    where job = 'MANAGER';
```

→ error

→ We can not use combination of multirow function in col name in select clause because the no. of rows of o/p does not match.

Q/ Write a query to retrieve lowest salary with respect to clerk.

```
SQL> select min(sal)
     from emp
     where job = 'CLERK';
```

Q/ WAQ to retrieve average salary of all the clerk.

```
SQL> select avg(sal)
     from emp
     where job = 'CLERK';
```

→ 1037.5

Q/ WAQ to find lowest salary in department 20.

```
SQL> select min(sal)
     from emp
     where deptno = 20;
```

→ 800

Q/ WAQ to find average salary of department 20.

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

→ 2175

Q/ WAQ to find average salary of each department

```
SQL> select avg(salary)
     from emp
     group by deptno;
```

Group by Clause :-

→ We go for groupby clause whenever we want to divide the table into multiple groups.

```
SQL> select avg(sal), deptno
     from emp
     group by deptno;
```

AVG(SAL)	DEPTNO
1566.66667	30
2175	20
2916.66667	10

Note: We can use a combination of multirow function if and only if it is a member of group by clause.

Q/ WAQ to find highest salary in each department

```
SQL> select max(sal), deptno
     from emp
     group by deptno;
```

Scanned by CamScanner

Q// WAQ to find average salary of each job.

SQL> select avg(sal), job
from emp
group by job ;

Q// WAQ to count no. of employees in each department.

SQL> select count(*), deptno
from emp
group by deptno ;

Q// WAQ to find least salary in each department except department 20.

SQL> select min(sal), dept no
from emp
where deptno != 20
group by deptno ;

Q// WAQ to count each job in each department.

SQL> select count(job), deptno, job ;
from emp
group by deptno, job ;

Q// WAQ to count no. of employees in each dept and select only those department which no. of count is more than 3.

SQL> select count(*), deptno
from emp
~~where~~
group by deptno
having count(*) > 3 ;

where → select rows
reject rows

having → select the groups.

Having Clause :

→ It is required to decide which group to select, which group not to select.

Q// WAQ to find highest salary for each job except analyst.

SQL> select max(sal), job
from emp
group by job
having job != 'ANALYST' ;

select max(sal), job
from emp
where job != 'ANALYST'
group by job ;

Q// WAQ to find average salary of each job except president and avg sal should be more than 2000 and while retrieving order with respect to job.

SQL> ~~select avg(sal), job~~
~~from emp~~
~~group by job~~
~~having job != 'PRESIDENT'~~

MRF
↓
[select, having]

SQL> select avg(sal), job
from emp
where job != 'PRESIDENT'
group by job
having avg(sal) > 2000
order by job ;

Note : MRF are not allowed in where condition.

Scanned by CamScanner

Clause	Order of Execution
Select	5
from	1
where	2
group by	3
having	4
order by	6

alias — not used in where; used in select and order by clause.

Q// WAQ to retrieve all the ~~queries~~ employees who belongs to dept 20.

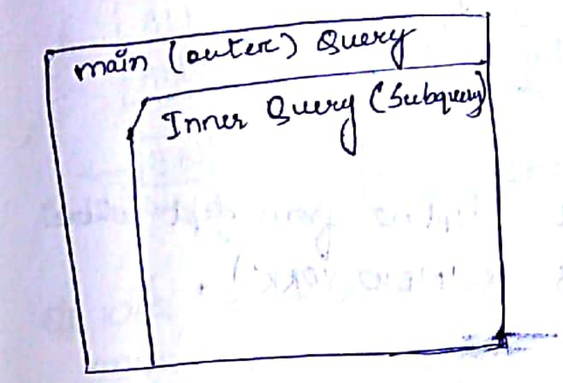
Ans
```
SQL> select *
     from emp
     where deptno = 20 ;
```

Q// WAQ to retrieve all the employees whose department no. is same as 'ADAMS' department no.

Ans
```
SQL> select *
     from emp
     where deptno = (select deptno from emp where
                     ename = 'ADAMS') ;
```

Subquery : It means using one query within another query.

- In that case always inner query executes first and based on the output of inner query the output of outer query is executed.
- We go for subquery whenever we want to retrieve information based on unknown information.



Q// WAQ to retrieve all the employees whose job is same as JAMES JOB.

```
SQL> select *
     from emp
     where job = (select job from emp where ename =
                  'JAMES') ;
```

Q// WAQ to retrieve all the employees who earns more than MILLER.

```
SQL> select *
     from emp
     where sal > (select sal from emp where
                  ename = 'MILLER') ;
```

Q// WAQ to retrieve employee, name, job and salary for all the employees whose job is same as FORD's job and earns more than 'ALLEN'.

```
SQL> select ename, job, sal
     from emp
     where job = (select job from emp where
                  ename = 'FORD')
     and sal > (select sal from emp where
                  ename = 'ALLEN') ;
```

Q// WAQ to retrieve all the employees whose salary is more than average salary of department 20.

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

Scanned by CamScanner

Q// WAQ to retrieve all the employees who works in New York.

```
SQL> select *
     from emp
     where deptno = (select deptno from dept where
                     locat~~s~~ = 'NEW YORK');
```

Q// WAQ to retrieve department information of 'KING'.

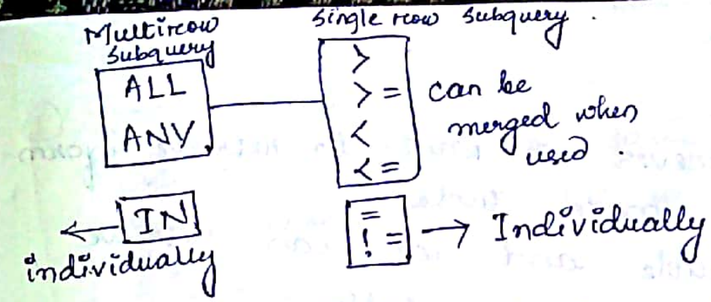
```
SQL> select *
     from emp
     where deptno = (select deptno from dept
                     where ename = 'KING');
```

Q// WAQ to retrieve all the employees whose salary is more than average salary of all the departments.

```
SQL> select *
     from emp
     where sal > all (select avg(sal) from emp
                      group by deptno);
```

Note : If the subquery returns more than one output, those are known as multirow subquery. In that case we have to use multirow operators such as ALL, ANY, IN based on requirement.

- If the inner query retrieves only one output those are known as single row subquery.
  - In that case we have to use single row operators <, <=, >, >=, =, !=. based on requirement.



Q// WAQ to retrieve all the employees whose salary is more than average salary of any one dept.

```
SQL> select *
     from emp
     where sal > any (select avg(sal) from emp
                      group by deptno);
```

Q// WAQ to retrieve all the employees whose salary is more than any one of salesman salary.

```
SQL> select *
     from emp
     where sal > (any (select sal from emp where
                       job = 'SALESMAN');
```

Q// WAQ to retrieve employee information who belongs to research and sales department.

```
SQL> select *
     from emp
     where deptno in (select deptno from dept
                      where Dname in('RESEARCH',
                                     'SALES'));
```

Q// WAQ to retrieve department table information of all the clerk and analyst.

```
SQL> select *
     from dept
     where deptno in (select dept no
                      from emp where job in
                      ('CLERK','ANALYST'));
```

Scanned by CamScanner

Joins :-

→ We go for joins whenever we want to retrieve information from one or more table.

→ We can join the table and we can retrieve the information from both the tables.

Emp

Emp no	Ename	Deptno
1	Dinga	10
2	Dingi	30
3	Ranga	20
4	Rangi	30

Dept

Deptno	Dname
10	HR
20	Dev
30	Test
40	ROD

$\boxed{\text{Emp . Deptno = Dept . Deptno}}$

```
select *
from emp, dept
where emp.deptno = dept.deptno ;
```

Types of joins :-

1. Inner joins
   - (i) equi joins
   - (ii) Non-equi joins
   - (iii) Self joins
2. Cartition join
3. Outer join
   - (i) left outer join
   - (ii) Right outer join
   - (iii) Full outer join

1. Inner joins :

→ This means the rows satisfies the ~~rows~~ join condition only those rows will be displayed in the output.

(i) equi joins :

→ If the joining condition contains equal sign those are considered as a equi join.

Q/ WAQ to retrieve employee name and their department names.

```
SQL> select ename, dname
     from emp, dept
     where emp.deptno = dept.deptno ;
```

```
SQL> select ename, dname
     from emp e, dept d
     where e.deptno = d.deptno ;
```

} aliasing is used in table.

Q/ WAQ to retrieve department table information for MILLER.

→ SQL> Select d.* or select d.deptno, dname, loc
from emp, dept
where emp.deptno = dept.deptno
and ename = 'MILLER' ;

Q/ WAQ to retrieve employee table information who belongs to DALLAS.

→ SQL> select e.*
from emp e, dept d
where e.deptno = d.deptno
and loc = 'DALLAS' ;

Scanned by CamScanner

Q// WAQ to retrieve employee and dept information for all the employees whose salary is more than 1000 and their department names second character is A.

```
SQL> select *
     from emp e, dept d
     where e.deptno = d.deptno
     and sal > 1000
     and dname like '_A%' ;
```

Non-equi joins:

→ If the joining condition does not contain equal sign are known as non-equi joins.

```
SQL> select ename, sal, grade
     from emp, salgrade
     where sal between losal and hisal ;
```

Self join:

→ If we are joining same table with another copy of same table, we call it as self join.

Emp e

Emp no	Ename	Mgr
1	A	3
2	B	3
3	C	2
4	D	1
5	E	2

Emp m

Emp no	Ename	Mgr
1	A	3
2	B	3
3	C	2
4	D	1
5	E	2

```
select e.ename, m.ename.
from emp e, emp m
where e.mgr = m.empno ;
```

Q// WAQ to retrieve employee name, employee desg and their managers name.

```
SQL> select e.ename, e.job, m.ename
     from emp e, emp m
     where e.mgr = m.empno ;
```

Q// WAQ to retrieve ename, employees dept name, manager name.

```
SQL> select e.ename, d.dname, m.ename
     from emp e, emp m, Dept d
     where e.mgr = m.empno
     and e.deptno = d.deptno ;
```

Note: If we are retrieving 'n' tables there must be 'n-1' joining condition compulsory.

Q// WAQ to retrieve employee name, employees dept name, manager name and manager's department name.

```
SQL> select e.ename, d.dname, m.dname, D2.dname
     from emp e, emp m, Dept d1, Dept d2
     where e.mgr = m.empno
     and e.deptno = d.deptno
     and m.deptno = d2.deptno ;
```

Cartition joins:

→ Joining each and every row of one table with each and every row of another table.

→ We can achieve cartition join whenever the joining condition is missed out.

```
SQL> select *
     from emp, dept ;
```

Scanned by CamScanner

Outer joins

→ It means the rows which satisfies the join condition as well as which does not satisfy the join condn will be displayed in the output based on which type of outer join.

ⓐ left outer join = inner join rows + left out rows of left table.

```
SQL> select *
     from emp e, dept d
     where e.deptno = d.deptno(+);
```

```
SQL> select e.ename, m.ename
     from emp e, emp m
     where e.mgr = m.empno (+) ;
```

ⓑ right outer join = inner join rows + left out rows of right table.

```
SQL> select *
     from emp e, dept d
     where e.deptno (+) = d.deptno ;
```

ⓒ full outer join = inner join rows + left out rows of both table.

```
SQL> select *
     from emp e, dept d
     where e.deptno (+) = d. deptno(+) ;
```

→ Full outer join is not possible using oracle joins. If we have to achieve full outer join, we have to go for ANSI join.

ANSI joins :

American National Standard Institute

→ Oracle joins works for oracle database only but ANSI joins works for any database.

Q/ WAQ to retrieve all the employee details who belongs to research and sales dept.

```
SQL> select e.*
     from emp e, dept d
     ~~where e.deptno = d.deptno~~
     ~~and~~ where e.deptno = d.deptno
     and dname in ('RESEARCH', 'SALES') ;
```

Q/ WAQ to retrieve dept information of all the CLERKS and ANALYST.

```
SQL> select d.*
     from emp e, dept d
     where e.deptno = d.deptno
     and job in ('CLERK', 'ANALYST') ;
```

Q/ WAQ to retrieve dept name and location for all the employees who is working under CLARK.

```
SQL> select dname, loc
     from emp e, dept d, emp m
     where e.mgr = m.emp no
     and e.deptno = d.deptno
     and m.ename = 'CLARK' ;
```

Scanned by CamScanner

Q// WAQ to retrieve employee information of all the salesman who is not located at 'DALLAS'.

```
SQL> select e.*
     from emp e, dept d
     where e.deptno = d.deptno
     and job = 'SALESMAN'
     and d.loc != 'DALLAS';
```

Q// WAQ to retrieve all the employee details who works in a same department as ADAMS and while retrieving reject ADAMS information

```
SQL> select f.*
     from emp e, emp f
     where e.deptno = f.deptno
     and e.ename = 'ADAMS'
     and f.ename != 'ADAMS';
```

Q// WAQ to retrieve employee name and salary for all the employees who is earning more than their manager.

```
SQL> select e.ename, e.sal
     from emp e, emp f
     where ~~e.deptno~~ e.mgr = f.empno
     and e.sal > f.sal;
```

Q// WAQ to retrieve all the employee details who is earning same salaries.

```
SQL> select e.ename, e.sal
     from emp e, emp f
     where e.sal = f.sal
     where ~~e.ename = f.ename;~~
     e.empno != f.empno;
```

Q// WAQ to retrieve all the details whose job is same as 'BLAKE's job and while retrieving reject BLAKE information.

```
SQL> select f.*;
     from emp e, emp f
     where e.job = f.job
     and e.ename = 'BLAKE'
     and f.ename != 'BLAKE';
```

Assignment

Q// WAQ to list all the information of employee with location and grade of all employee belongs to the grade range from 2 - 4 and working for those department which are not starting from 'OP' or 'S' and their employees are joined in 1981 but not in the month of MAR or SEP and salary not ends with '00' displaying ascending order of grades.

→

```
select
from emp e, dept d, salgrade s
where e.deptno = d.deptno.'
and (d.dname not like in ('OP%', 'S%')
and (to_char (hiredate 'YYYY') = '1981'
and to_char (hiredate, 'MON') not in ('MAR','SEP');
```

Scanned by CamScanner

Q// WAP to retrieve highest salary.

```
SQL> select max (SAL)
     from emp ;
```

Q// WAP to retrieve 2nd highest salary.

```
SQL> select max (SAL)
     from emp
     where sal < (select max(SAL) from emp);
```

Co-related sub-query :-

- Co-related sub-query is both combination of sub-query as well as joins.
- In case of co-related subquery always outer query executes 1st because of joining condition is inner query.
- 1st outer query executes and retrieve the table and give it to the inner query and inner query executes based on the condition again it will pass output to the outer query, the outer query executes again that is outer query executes twice.
- We go for co-related sub-query whenever we want to compare two values.

empno	ename	sal
1	A	600
2	B	500
3	C	400
4	D	200
5	E	500

y sal > = X sal	count = 50
600 > = 600	1
600 > = 500	0
600 > = 400	0
600 > = 100	0
	1

empno	ename	sal
1	A	600
2	B	500
3	C	400
4	D	200
5	E	500

y sal > = X sal	
500 > = 600	1
500 > = 500	2
500 > = 400	0
500 > = 200	0
	2

y sal > = X sal	
400 > = 600	1
400 > = 500	2
400 > = 400	3
400 > = 200	0
	3

y sal > = X sal	
200 > = 600	1
200 > = 500	2
200 > = 400	3
200 > = 200	4
	4

→ WAQ to retrieve 2nd highest salary.

```
SQL> select *
     from emp X
     where 2 = (select count (distinct sal) from emp Y
                where Y.sal > = X.sal) ;
```

→ WAQ to retrieve 6th highest sal.

```
select *
from emp X
where 6 = (select count (distinct (sal)
           from emp Y
           where Y.sal > = X.sal) ;
```

→ WAQ to retrieve nth highest sal?

```
select *
from emp X
where &n = (select count (distinct sal)
            from emp X
            where Y.sal > = X.sal) ;
```

→ WAQ to retrieve 3rd and 8th highest sal.

```
select *
from emp X
where 3 = (select count (distinct sal) from
           emp Y where Y.sal >= X.sal)
or 8 = (select count (distinct SAL) from
        emp Y where Y.sal > = X.SAL) ;
```

Scanned by CamScanner

Q// WAQ to retrieve 4th least sal :

```
select *
from emp X
where 4 = (select count (distinct SAL) from emp Y
           where Y. SAL >= X. SAL) ;
```

Q// WAQ to retrieve 3 ~~least~~ ~~SA~~ highest salaries.

```
Select *
from emp X
where 3 >= (select count (Distinct sal)
            from emp Y
            where Y. sal >= X. sal);
```

Q// WAQ to retrieve 4 least salaries.

```
select *
from emp X
where 4 >= (select count (Distinct sal)
            from emp Y
            where Y. sal <= X. sal) ;
```

Q// WAQ to retrieve 5th emp who joined a company

```
select ename
from emp X
where 5 = (select count (distinct hiredate)
           from emp Y
           where Y. hiredate <= X. hiredate);
```

Q// WAQ to retrieve all the emp whose sal is more than avg sal of their own dept.

```
select * from emp x
where sal > (select avg(sal)
             from emp y
             where x. deptno. = y. deptno) ;
```

Q// WAQ to retrieve all the emp whose sal is more than avg sal of their own job.

```
select * from emp x
where sal > (select avg(sal)
             from emp y
             where x. job = y. job) ;
```

Data types :-

(a) Numbers :-

integer → (4digits) [4]

decimal no. → [4.2]

(b) Strings :-

char (4) — SQL X ← wasted (fixed size)

SQL X ← can be reused (variable size)

after 10g 4000 in both: 2000 ← varchar (4); 4000 ← varchar 2 (char capacity)

(c) date :- Date with a fixed format "dd-Mon-YY". [default]

Constraints :-

Constraints are required to restrict the data which is being inserted in a table.

(i) NOT NULL Constraint :

- If we declare any column as "not null" in that column 'Null' values are not allowed but duplicate values are allowed.

  Ex : emp name, student name etc.

(ii) UNIQUE Constraint :

This type if we declare any column as 'unique' constraint in that column 'null' value is allowed but 'duplicate' value is not allowed.

Ex : phno.

We can add 'n' no. of null values.

(iii) Primary key : [Not null and unique is allowed]

- One primary key is 'mandatory'.
- If we declare any column as primary key in that column "Null values" and "duplicate" values are not allowed.
- We can create any one primary key for table.
- We can create a table without using a 'primary key' but we may face some issues during joins.

(iv) Check Constraint : [It will check condition]

- It provides extra restriction on a column.

  (age) ← extra restriction

  check (age >=16 and age <=20)

(v) Candidate key :

Column that are eligible to become 'primary key' are known as candidate key.

(vi) Alternate key :

Columns that are eligible to become primary key not selected as a primary key i.e known as alternate key.

Alternate key = Candidate key - primary key.

(vii) Foreign key : [Referential integrity]

- It is used for joining 1 or more table.
- By using this key we can form referencial integrity between table.
- Foreign key applied on a column, the current table becomes child table and the reference table is called as parent table.
- Foreign key applied on a column that becomes child column and refering column becomes parent column.
- Any data inserted in child key column should be present in parent key column.
- We can create one or more foreign key in a single table.
- In foreign key column both 'null value' and 'duplicate values' are allowed.

(viii) Composite Primary key :

- Primary key consists of more than one column are known as composite primary key.

Scanned by CamScanner

DDL :→ [Data Definition Language]

Table creation :

syntax:
create table table-name (column1 datatype (size) constraint, column2 datatype(size) constraint...);

* create table course (cid number(2) primary key, cname varchar2(20) not null);

desc course

CID	Notnull	Number
CNAME	Notnull	Varchar 2

* create table student (sid number(2) primary key, sname varchar(20) not null, marks number(2), age number (2) check (age >= 16), cid number(2) reference course (cid));

desc student

SID
SNAME
MARKS
AGE
CID

* To copy a whole table →
from emp to emp2

// create table emp2 as select * from emp;

- The above statement copies both structure as well as content of a table.

// create table emp3 as select * from emp where 1 = 2;
↓
any false statement.

The above statement copies only the 'structure' of the table.

Creation commands

(a) < Alter command > :

- By using this command we can modify the structure of the table.
- By using this command we can add a new column to the existing table.

syntax : Alter table table-name
add colname, datatype (size) constraints;

* alter table emp3
add remarks varchar (30);

→ By using alter command we can 'modify' a 'data type'.

→ By using this we can also 'add' a constraint.

syntax :

Alter table table-name
modify colname new datatype (size) constraints;

* alter table emp3
modify remarks number (2) not null;

→ By using alter command we can rename a column name.

Scanned by CamScanner

syntax : alter table table-name
rename column old id name to new column name;

* alter table emp3
rename column remarks to rating ;

→ By using alter command we can drop /delete a column.

syntax : alter table tablename
drop column col-name ;

* alter table emp3
drop column comm ;

(b) <Rename command> :

Syntax : Rename old table-name to new table name;

ex :- rename emp3 to emp4 ;

(c) <Truncate command> :

syntax : truncate table table-name ;

→ Truncate command removes only the content of a table but the structure remains same

ex :- truncate table emp2 ;

(d) <Drop command> :

syntax : drop table table_name ;

→ This command removes both content as well as structure of a table.

ex:- drop table emp3 ;

[select * from tab;] all tables will be shown.

(e) <Flash Back>

retrieve from recycle bin.
(recyclebin to database back) [select * from recyclebin

flashback table "addr from recyclebin" to before drop ;

(f) <Purge command>

- It completely delete the table from recycle bin

purge recyclebin ;

DML :→ [Data Manipulation Language]

① Insert command :

syntax :-
insert into table-name (col1, col2, ... coln)
values (val1, val2, ..., valn) ;

ex- insert into course (cid, cname)
values (1, 'development') ;
↓
should be in one particular case.

ex- insert into course (cid, cname)
values (2, 'testing') ;

ex- insert into student (sid, sname, marks, age, cid)
values (1111, 'Smruti', 80, 23, 1) ;

Scanned by CamScanner

To avoid repratation for each rows and value I (/)

ex insert into student (sid, sname, marks, age, cid)
values (&cid, &sname, &marks, &age, &cid);

o/p
again "/" for recent (last) query.

Scanned by CamScanner

(II) update command :

- By using update command we can modify the content of a table.

syntax : update table-name
set col1 = val1, col2 = val2
where condition ;

SQL> update emp2
set sal = 6000
where sal = 3000 ;

SQL> update emp2
set sal = 4000, comm = 400
where sal = 6000 ;

SQL> update emp2
set comm = 400 ;

- In update command if where condition is missed out then all the rows will be updated.

(III) delete command :

syntax : Delete table-name
where condition.

SQL> delete emp2
where ename = 'JAMES' ;

SQL> delete emp2
where sal > 3000 ;

Truncate	Delete
- It is permanent	- It is temporary
- It comes under DDL sublanguage	- It comes under DML sublanguage
- Where condition is not allowed in Truncate.	- Where condition is allowed in delete command.
- We can not delete particular records in truncate command.	- We can delete particular records in delete command.
- It can not be rolled back.	- It can be rolled back.

DTL / TCL :→ [Data Transition Language] [Transition Control Languge].

(i) Commit
(ii) Roll back
(iii) Save point

- All these commands works only with DML command

i) Commit Command

- By using commit command we can save all the DML commands permanently.

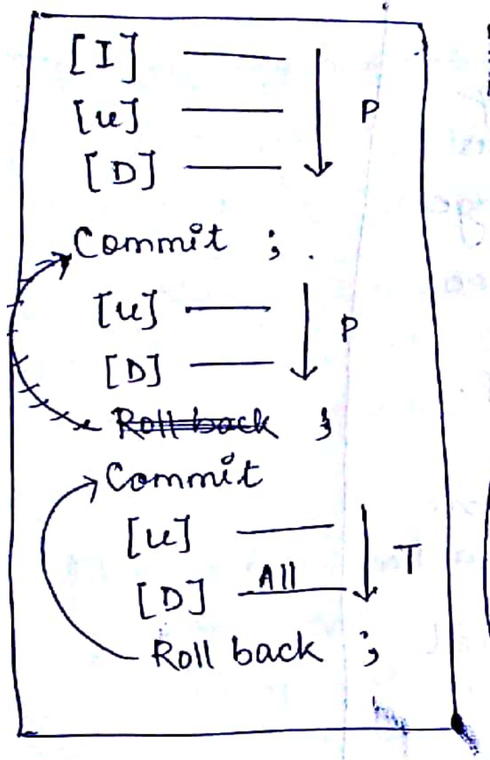
ii) Roll back Command

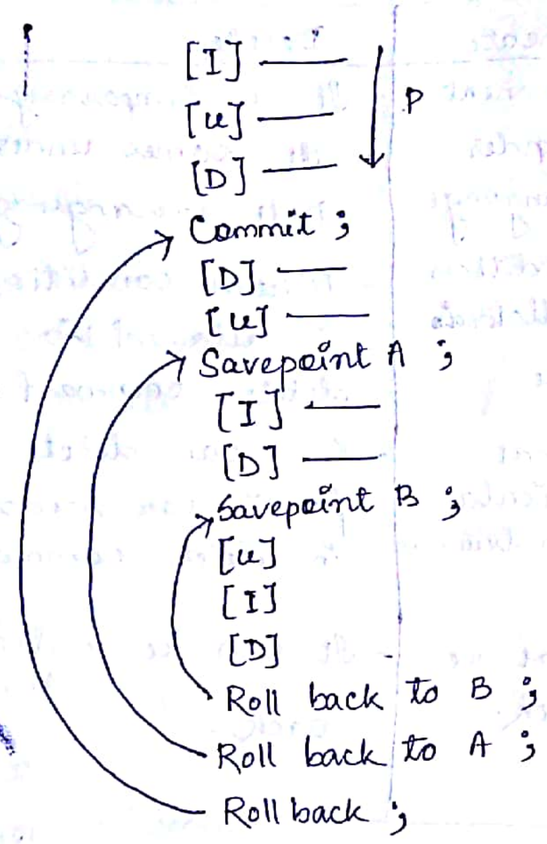
- It is used to take back us to the previous commit statement.

iii) Save point Command

- It is used to create multiple save points.

Scanned by CamScanner





Note :

We can close SQL plus editor in two ways.

(i) click on close button

(ii) exit command

- exit command is having auto commit feature i.e if we are executing exit command all the DML commands will save permanently without using commit command.

DCL :- [Data Control Languge]

- Grant
- Revoke

- Grant command is used to grant a permission to other user.
- Revoke command is used to take back those permission.

```
SQL> Create user xyz identified by xyz123 ;
  error

SQL> conn system/tiger ;
Connected
SQL> Show user ;
USER is "SYSTEM"
SQL> create user xyz identified by xyz123 ;
User created.

SQL> select *
     from all-users ;

SQL> grant connect to xyz ;
Grant succeeded.

SQL> conn xyz/xyz123 ;
Connected.
SQL> show user ;
USER is "XYZ".
SQL> select *
     from scott.emp ;
  error
SQL> conn scott/tiger ;
Connected.
SQL> grant select on emp to xyz ;
Grant succeeded.
```

Scanned by CamScanner

```
SQL> conn xyz /xyz123;
Connected.
SQL> select *
     from scott.emp;
SQL> conn scott / tiger
Connected.
SQL> revoke select on emp from xyz;
Revoke succeeded.
SQL> conn xyz /xyz123;
Connected.
SQL> select *
     from scott.emp;
error
SQL> conn system / tiger;
Connected.
SQL> drop user xyz;
User dropped.
SQL> Select *
     from all_users;
```

Q// WAQ to retrieve first 5 records.

```
select *
from emp
where rownum <= 5;
```

Q// WAQ to retrieve last 'n' records.

```
select *
from emp
minus
select *
from emp
where rownum <= (Select count (*) - &n from emp)
```

```
Enter value for n : 6
old 6 : where rownum <= (Select count (*) - &n from emp)
new 6 : where rownum <= (Select count (*) - 6 from emp)

SQL> Select *
     from (select * from emp order by rownum desc)
     where rownum <= 6
     order by rownum;
```

Q// WAQ to retrieve particular records.

```
SQL> Select *
     from emp
     where rownum <= 5
     minus
     select *
     from emp
     where rownum <= 4;
```

Q// WAQ to retrieve unique records.

emp X

Emp no	Ename	Rowid
01	ALLEN	0A
02	WARD	0B
03	MARTIN	0C
04	WARD	0D

emp Y

Emp no	Ename	Rowid
01	ALLEN	0A
02	WARD	0B
03	MARTIN	0C
04	WARD	0D.

```
SQL> select *
     from emp X
     where rowid = (select max (rowid)
                    from emp Y
                    where X.empno = Y.empno);
```

0A
0B

0B
0D (max)

Q) WAQ to retrieve duplicate records.

SQL) Select *
from emp X
where rowid != (Select max (rowid)
from emp Y
where X.empno = Y.empno);

Normalization :-

→ Normalization are just a guide lines.
→ If we are following the guide lines we can achieve all the operations properly.
→ If we are not following the guide lines we may face some issues during retrieval, insertion, modifications or deletions.

(i) 1st Normal Form : [1NF]
- Duplicate columns are not allowed.
- Create a primary key.
- No multivalue attributes.

(ii) 2nd Normal Form : [2NF]
- The table should be in 1st Normal form stage.
- Split the table and create the foreign key.

(iii) 3rd Normal Form : [3NF]
- The table should be in second normal form stage
- Remove all the columns that are not related to the particular table.

(iv) 3.5 NF : [BCNF]
(Boyce Codd Normal Form)
- All the columns should be a candidate key.

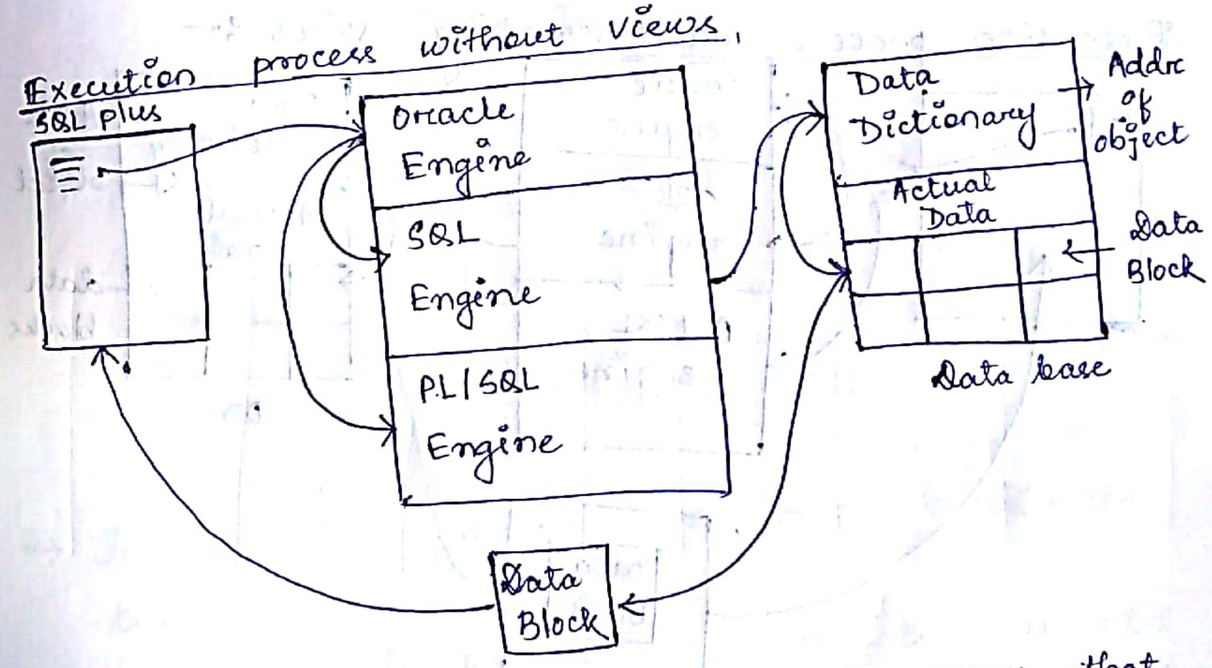
(v) 4th Normal Form
- The table should be in 3NF stage.
- No composite primary key.

(vi) 5th Normal Form
- Table should be in 4NF stage.
- For joining two tables use candidate keys.

PL/SQL [Procedural language / Programming language]

Views :
- Views are virtual tables.
- By using views we can achieve two things.
  * We can increase database performance.
  * We can provide security to our data.

Execution process without Views.



→ In SQL plus we will write a SQL query, that will executed by oracle engine.
→ In oracle engine it contains 2 types of engine.
(i) SQL engine (ii) PL/SQL engine.
→ SQL queries will be executed by SQL engine and PL/SQL queries will be executed by PL/SQL engine.

Scanned by CamScanner

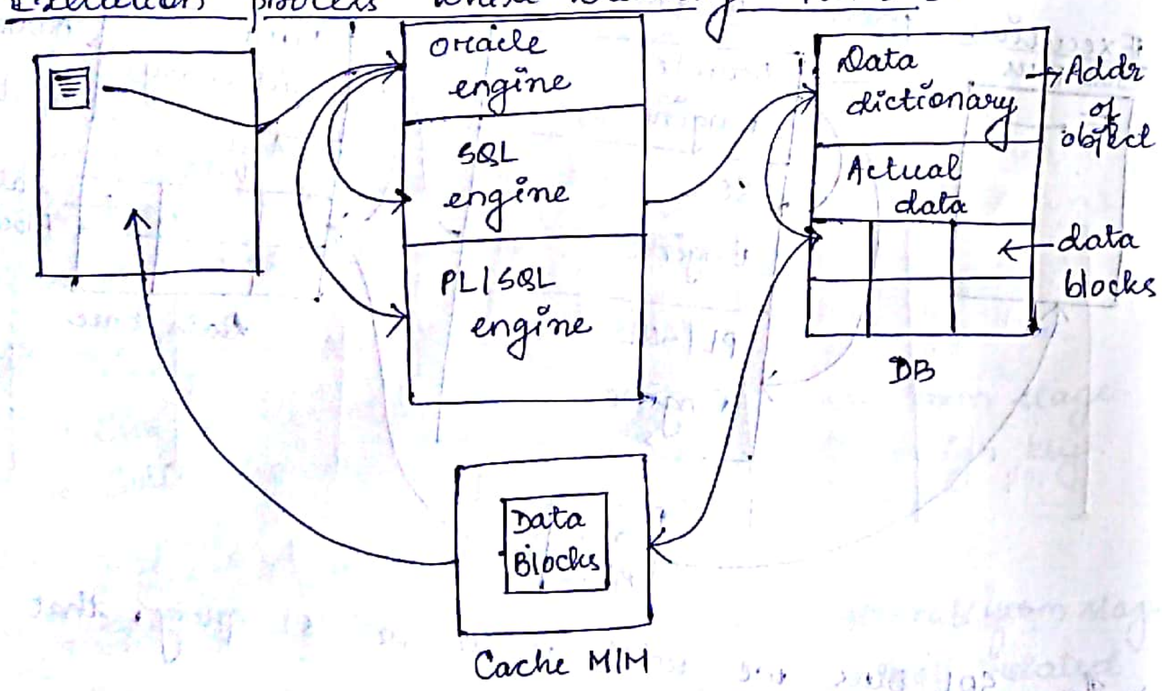
→ Then the executed queries will go to data base.
→ The data base contains two parts.
(i) Data dictionary – contains address of object.
(ii) Actual Data – contains data blocks.

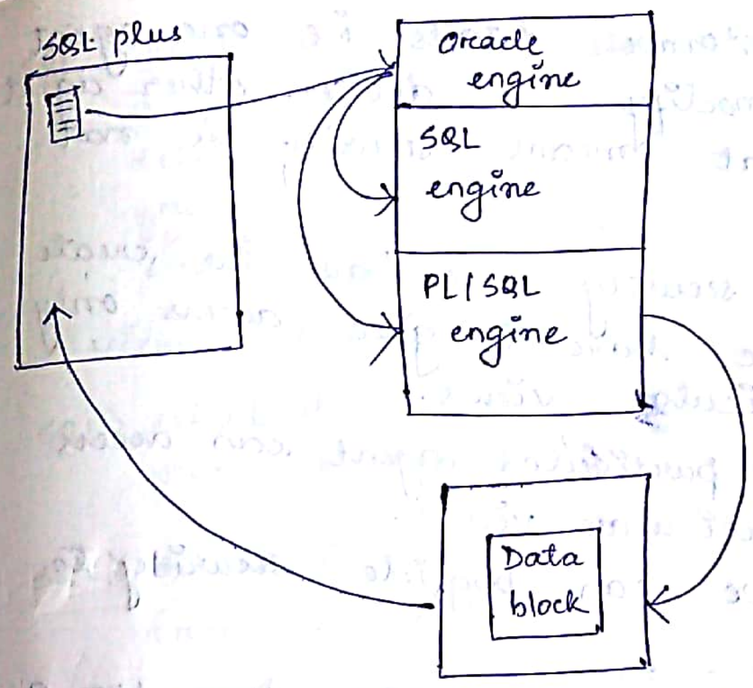
→ The data will go to data block and the output will be displayed.

- The same above process will be followed each and every time when we execute queries.
- At the number of hits increases to the database then the database performance will be decreased.

Example : Youtube

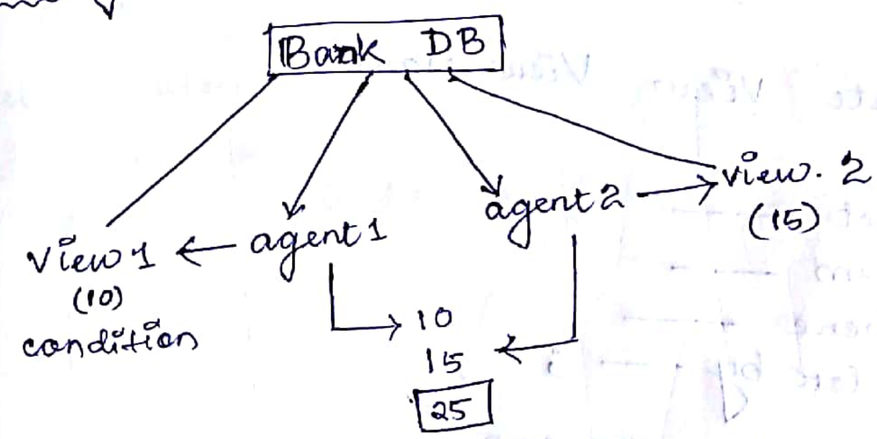
Execution process while creating views :–





[Execution process after creating views]

ⓑ Security :



→ In the above example we have bank data base.
→ For bank db we have 2 agents. – agent 1, agent 2
→ agent 1 is having 10 customers and agent 2 is having 15 customers.
→ All the 25 records will be stored in BDB.
→ For the agents if we are giving direct access to the BDB any agent can execute any operation

Scanned by CamScanner

to all the customer details. i.e one agent can access, modify or delete other agent information. that means security is not provided.

- For providing security we have to create views and we have to give access only for that particular views.
- That means, the particular agent can access only their particular views.
- In this way we can provide security to data.
- Views are dynamic, modification done for a view will be reflected in original database as well as modification change in original database will reflect in view.

```
Create View View_Name
as
Select ——
from ——
where ——
order by —— ;
```

```
SQL> create view vw_emp
     as
     select *
     from emp
     order by sal ;
```

View created.

```
SQL> select *
     from tab ;
```

```
SQL> create view vw_sales
     as
     select *
     from emp
     where job = 'SALESMAN'
     order by sal ;
```

- View created.

```
SQL> select *
     from vw_emp ;

SQL> select *
     from vw_sales ;

SQL> create view vw_hide_cols
     as
     select empno, ename, job, mgr, hiredate,
                                        deptno
     from emp
     order by ename ;
```

- View created.

```
SQL> select *
     from vw_hide_cols ;
```

Scanned by CamScanner