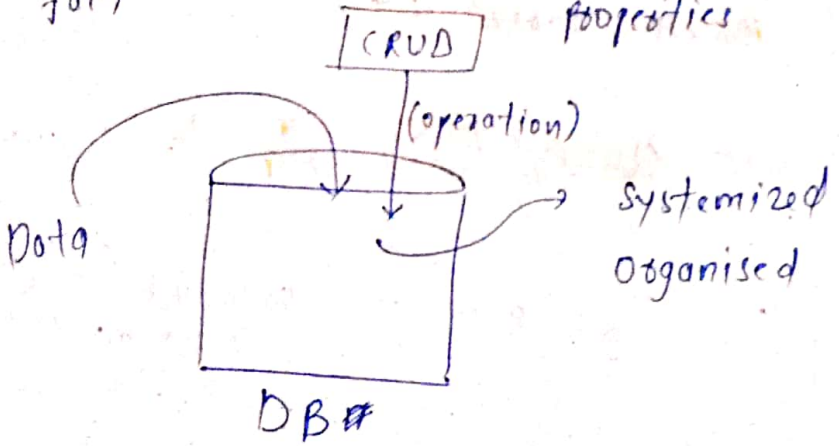
Data: Data is Raw fact that describe the property of ~~information~~ an object/entity.

Properties/Attributes.

Data $\xrightarrow[\text{fact}]{\text{Raw}}$ describes $\longrightarrow$ Attribute/properties $\longrightarrow$ Entity/object.



Database:

- It is place where data will be stored in systematic and organized manner.

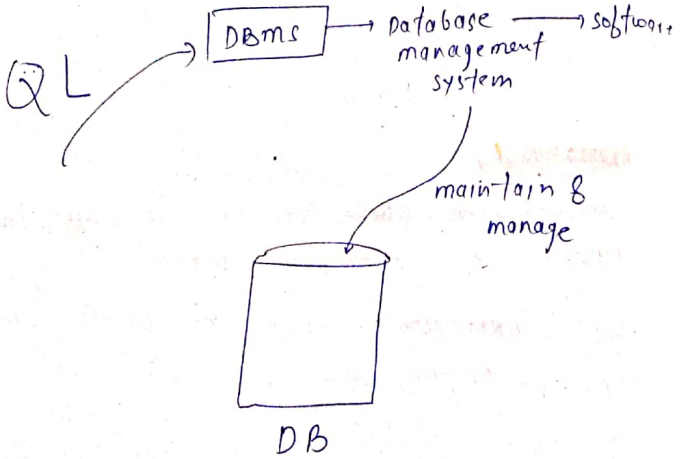
Basic operation performed on Database are:-

1) Create / Insert
2) Read / Retrieve
3) update / modify
4) Delete / Drop.

* Generally known as CRUD operation.

DBMS (Database management system)

- DBMS is a software we use to maintain & manage Database.
- Two important factor are.
  1) Security
  2) Authorization
- we use Query Language to communicate



- There are 4 different Types of DBMS Software:
  1) NDBMS (Network DBMS)
  2) HDBMS (Heirarichle DBMS)
  3) OODBMS (object oriented DBMS)
  4) RDBMS (Relational DBMS)

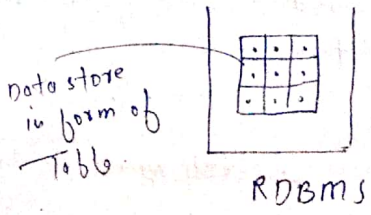
RDBMS:

- It is a Type of DBMS software were data will be stored in Form of Tables.
- Two Important Factor are
  1) security
  2) Authorization.
- we use SQL (structure Query Language) to communicate it.

Relational Model:

- It is a concept designed by data scientist "E.F.Codd"
- In Relational model we store the data in the Form of Table's.
- * A DBMS software which follow's Relational model Becomes Relational DBMS (RDBMS)

DBMS → Follows → (RM) → RDBMS



Column/Attribute/Fields

STD	SName	SBranch	Perc...
1	Shaala	mech	69
2	Shakeela	CIVIL	70
3	Laila	CSE	85
4	Alaina	EEE	90
5	Pooja	ECE	95

Row/Records/Tuples

1. Table
2. Column
3. Row
4. Cell
5. Entity

Table:

- Table is a logical organization of data which consist of ROWS AND COLUMNS

Column:

- Column is also refered as attribute or Fields.
- A column is used to represent the one property of All the entities.

ROWS:

- Row is also refered as Record or Tupples.
- A Row is used to represent the properties of single entity.

Cell:

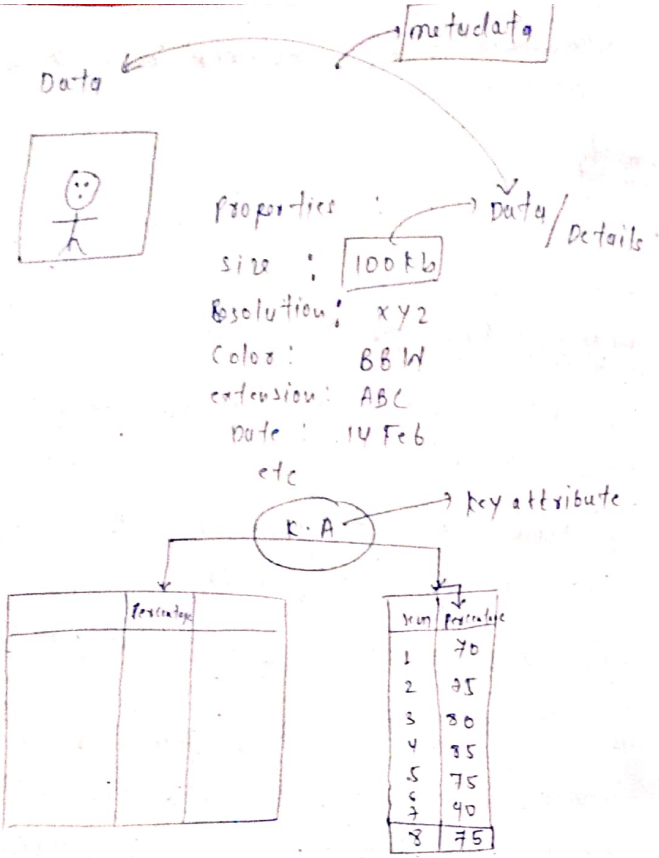
- Cell is the smallest unit in table in which we store data.
- The intersection of ROWS & column generate cell.

Entity:

- Anything which has its existance

Rules of E.F codd:

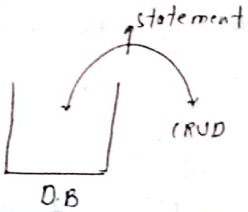
1. The data stored in the cell must be a single value data.
2. In RDBMS we store everything in the form of Table including META DATA.
   (The detail about the Data is meta Data)
3. According to E.F codd we can store data in multiple Table, If Needed we can establish connection b/w two table using key attributes
4. We can validate the data entered into the Table in Two steps
   a) By assigning Data Types
   b) By assigning Constraints



Statement: statement are the ones which are used to perform "CRUD" operation in DB.

Statement in SQL:

1. Data Definition Language (DDL)
2. Data manipulation language (DML)
3. Data control language (DCL)
4. Transaction control language (TCL)
5. Data Query language (DQL)



Data Query language:

- This statement is used to retrieve the data from database.
- There are 4 statements:
  1. Select
  2. Projection
  3) Selection
  4. Joins

Select:

- This statement is used to retrieve the Data from Database and Display it.

Projection:

- This statement is used to retrieve the Data from Database by selecting only column.

* All the value ~~of~~ In the column will be selected by default.

selection:

- This statement is used to retrieve the Data from Database by selecting both column as well as records.

Joins

- This statement is used to retrieve the Data from multiple Tables simultaneously.

Syntax for Projection:

```
SELECT */ [DISTINCT] colum.Name/ Expression
FROM   Table_Name;
```

student

SID	Sname	Branch	Percentage
1.	sheela	CS	35
2.	Jaya	EC	60
3.	Rekha	ECE	90
4.	sakshi	CE	75
5.	Sushma	civil	80

Q. write a Query to display SID from student Table.

```
Select S.ID
From student;
```

o/p :

SID
1
2
3
4
5

Q. write a Query to display s.Name from student Table..

```
Select S_Name
From student;
```

o/p :

s.Name
sheela
Jaya
Rekha
sakshi
Sushma

Q. write a Query to display branch from student Table.

→ select branch
From student;

o/p:

branch
CS
EC
EEE
CE
CIVIL

Q. write a Query to Display S.ID and Branch from student Table

→ select S-ID
select Branch
From Student;

→ Select S-ID, Branch
From student;

o/p:

SID	Branch

Q. write a Query to display S-Name & percentage From student Table.

→ select S.Name, percentage
From student;

o/p:

S-Name	Percentage

Q. write a Query to display all the details From student Table.

→ select *.
From Student;

Q. write a Query to display branch from student Table.

→ select branch
From student;

o/p:

branch
CS
EC
EEE
CE
CIVIL

Q. write a Query to Display S.ID and Branch from student Table.

→ select * S-ID
select Branch
From Student;

→ Select S-ID, Branch
From student;

o/p:

SID	Branch

Q. write a Query to display S-Name & percentage From student Table.

→ select S.Name, percentage
From student;

o/p:

S-Name	Percentage

Q. write a Query to display all the details From student Table.

→ select *.
From Student;

I
```
Select *
From Tab;
```
→ how many no. of Table in D.B.

II
```
DESC table_name;
```
→ col & datatype assigned to col's.

III
```
Select *
From Table_name
```
→ entire Table structure

EmpNo	ENome	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

Asterisk (*) :

It is used to Determine "To select all the columns".

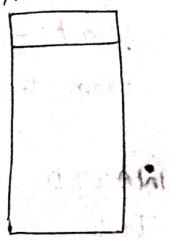
semi-colon (;)

It is used to determine "The end of the statement"

Q. WAQTD Name of All the Employees

⇒
```
Select ENome
From EMployee Table;
```

o/p



Q. WAQTD Name & salary given to all the employee

→
```
select ENome, SAL
From Employee Table;
```

Q. WAQTD Employee ID. AND DEPARTMENT number given to all the employees

→
```
select Emp No., Dept No
From Employee Table;
```

Q INAQTD EName and HIREDATE of the employee

```
select Ename, HIREdate
From Employee Table;
```

Q INAQTD Name and Designation of all the employee

→ select Ename, Job
From Employee Table;

Q. INAQTD all the details From the employee Table.

→ select *
From employee Table;

Q. INAQTD Annual salary of all the employee

→ select 12*sal
From employee Table;

Q. INAQTD EName and Annual salary given to all the employees

→ select Ename, 12*sal
From Eployee Table;

* We can write complete query in one line also.

Q INAQTD Name and Halfterm salary of all the employee.

→ select Ename, 6*sal
From employee Table;

Q INAQTD Ename and sal and also sal with 25% hike.

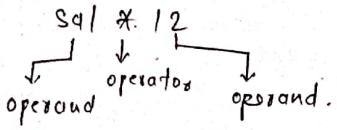
→ select Ename, sal + 0.25*sal) → sal + sal * 25/100
From employee Table;

Q INAQTD Ename and sal, sal with deduction of 12% For all the emp.

→ Select Ename, sal − sal * 12/100
From employee Table;

Expression:

- statement which give us Result is known as expression.
- Expression consist of Two Types
  1) operand
  2) operator (+, -, *, /)
- operand consit of two types
  1) Column Name
  2) Literals (Direct Value)
- Literals are of Three Types
  1) Number Literal
  2) Character Literal
  3) Data Literal
- Character Literal and Data Literal should be enclosed within single quotes.



Q. WAQTD Ename and sal and Also sal with 18% Hike.

→ select Ename, sal, sal + 18/100 * sal.

→ From Employee Table;

Q WAQTD Ename and Annual sal with bonus of 5000.

→ select Ename, sal * 12 + 5000

From employee Table;

ALIAS:

- ALIAS is used to Indicate for another name.

Q1) WAQTD Name, sal of the employee along with their annual salary.

```
Select Ename, Sal, sal*12 as Annual salary
From emp;
```

↳ (It is not mandatory)

"Annual Salary"

Annual_salary

Q2) WAQTD Ename and Job for all the employee with their halfterm salary.

→ Select Ename, Job, sal*6 "Half term salary"
From emp;

Q4) WAQTD Name salary and salary with hike of 10%.

→ select Ename, sal, sal+ sal*10/100 "Salary with hike of 10%"
From emp;

Q5) WAQTD Name & salary with Deduction of 25%

→ Select Ename, sal−sal*25/100 "salary with deduction of 25%"
From emp;

Q6) WAQTD Name and salary with monthly hike of Rs 50.

Select EName, sal + 50 "salary with monthly hike of 50%"

7) WAQTD Name & Annual salary with Deduction of 10%

→ Select Ename, sal*12 - (sal* - sal * 10/100)
From emp;

→ Select Ename, sal*12 - 12*sal*10/100 "As annual salary ded"
From emp;

8) WAQTD Total salary Given to each employee (sal + comm).

→ select sal+comm as "Total salary"
From emp;

empty all → Null operation → null

1500 + 0 → 1500
1500 + null → null

Note: This question can be solved by using a concept i.e NVL stands for Null value logic.

Q. WAQTD name and designation Along with 100 penalty in salary.

→ select Ename, Job as "Designation", sal-100 as "Penalty"
From emp;

Q. WAQTD All the details of the employee Along with an Annual salary.

→ select *, sal*12 as "Annual salary"
From emp;

No output:

select emp.*, sal*12 as "Annual salary"
From emp;

Q. WAQTD All the detail of employee table along with annual salary with bonus of 5000.

select emp.*, sal*12 + 5000 as "Annual sal with 5% bonus"
From emp;

Note: If we want to display any other column along with all the display we can use table-name.*;

Here . represent's database i.e it's a command to a compiler saying that go to DB search for the table and select all the details.

Q. WAQTD percentage from student ~~column~~ Table.

⇒ Select Percentage
From student;

o/p:

Percentage
85
75
80
81
80

Q. WAQTD the Different percentage from student Table.

DISTINCT → remove repeated/duplicated.

Select Distinct Percentage
From student;

Percentage
70
60
80

student:

SID	sName	Branch	Percentage
1	A	CS	70
2	B	EC	60
3	C	mech	80
4	D	EC	60
5	A	CS	70
6	F	~~mech~~ T.C	80

Q. WAQTD different branch from student Table.

→ select distinct branch
From student;

Branch
CS
EC
mech
TC

Q. WAQTD different branch & percentage from student table.

→ select distinct branch, ~~distinct~~ percentage
From student;

o/p

Branch	percentage
CS	70
EC	60
mech	80
T.C	80

Q. WAQTD Two different student name, Branch percentage from student Table

→ select distinct sName, Branch, Percentage
From student;

sName	Branch	Percen
A	CS	70
B	EC	60
C	mech	80
D	EC	60
F	T.C	80

different
unique
} Distinct.

Q. WAQTD only different salaries Given to employees Table.

→ select distinct sal
From emp;

Q. WAQTD two different Designation that are present in the Emp Table.

→ select distinct Job
From emp;

Q. WAQTD Different Dept No. as well as Salaries that are present in the Table.

→ select distinct Dept No., sal
From emp;

Q. WAQTD Ename who is Working in Deptno. 10.

Ename	Deptno.
Sheela	20
Leela	10
maala	30
Laila	10
Shakeela	20
Taquila	10

Row by Row

Selection:

where → filter the records

③ select Ename
① From Emp
② where deptno = 10;

{ where clause is used to filter the records. }

o/p of where clause:

Ename	Deptno
Leela	10
Laila	10
Taquila	10

o/p of select

Ename
Leela
laila
Taquila

NOTE:
- For where clause we can pass Filter condition as an argument.
- where clause executes Row by Row.
- where clause after the execution of From clause
- We can pass multiple condition for where clause using logical operators.

Q) WAQTD Ename working on Depart no. 20.

→ Select Ename
From emp
where dept.no=20;

o/p:

Ename
sheela
shakeela

---

where clause:

Syntax:

SELECT */ [DISTINCT] COLUMN_NAME/Expression [ALIAS]
From TABLE_NAME
where <filter_condition>;

order of execution:

① From
② where
③ select

Q. WAQTD sal of all the emps.

→ Select sal
From emp;

Q. WAQTD sal of Emp whose name is smith.

→ select sal
From emp
where Ename='SMITH';

* {We can pass words with single quotes}
* {Ename, Job title provide in where clause must be in upper case.}

Ex- select sal
From emp
where Ename='SMITH';

Q. WAQTD sal of an employe who are working on Department no. 20.

→ Select sal
From emp
where deptno=20;

'20' single Quotes
↳ still output is show but it's not standard way that's why we can not write with no.

Q. WAQTD name of the employee working as Clerk.

```
select Ename
From emp
where Job = 'CLERK';
```

Q. WAQTD the annual salary of the employee who's name is smith.

```
Select sal * 12 as "annual sal"
From emp
where Ename = 'SMITH';
```

Q. WAQTD Detail of Emp whose name is Jones.

```
select *
From emp
where Ename = 'JONES';
```

Q. WAQTD Salary of the employee who are working as salesman.

```
→ Select Sal
  From emp
  where Job = 'SALESMAN';
```

Q4.) WAQTD Details of the empl who earn more than 2000.

```
→ select *
  From emp
  where (SAL > 2000);
```

Q5) WAQTD Detail of the emp. whose name is Jones.

```
→ select *
  From emp
  where Ename = 'JONES';
```

Q8) WAQTD EmpNO. of the employees who are working in Dept 30.

```
→ select EmpNO.
  From emp
  where Dept.NO = 30;
```

Q10) WAQTD Detail of the employees working as manager.

```
select *
From emp
where Job = 'MANAGER';
```

Q11) WAQTD Name and salary Given to an employee If employee earns a commision of Rupees 1400

→
```
select Ename, sal
From emp
where comm = 1400;
```

Q14) WAQTD Details of employee working as an analyst.

```
select *
From emp
where Job = 'ANALYST';
```

Q15) WAQTD Details of employee Earning sal more than 2000 Rupees per month.

```
select *
From emp
where sal > 2000;
```

Q7. WAQTD Name & sal along with his annual Salary if the annual sal is more than 12000

→
```
select Ename, sal, sal*12 as "Annual sal"
From emp
where sal*12 > 12000;
```

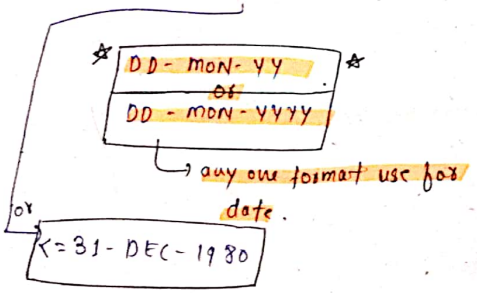
Note: We can't pass ALIAS name in where clause because ALIAS name is given select clause but select clause execute after executing of where clause.

Q8) WAQTD Details of the emp who was hired after 01-Jan-81.

```
select *
From emp
where HireDate >'01-JAN-81';
```

Q9) WAQTD Ename and Hiredate if they are Hired before 1981.

```
Select Ename, Hiredate
From emp
where Hiredate < '01-JAN-1981';
```



Q13) WAQTD EmpNo. of employee Hired before the Year 87.

```
select EmpNo.
From Emp
where HiredDate < '01-JAN-87';
```

Q. WAQTD Ename and Hiredate if they are hired after 1981

```
Select Ename
From emp
where Hiredate >'31-Dec-1981';
                or
      >='01-Dec-1982';
```

Q WAQTD EmpNo. of employees hired after the Year 85

```
select EmpNo.
From emp
where hiredate > '31-DEC-1985';
                or
      >= 01-JAN-1986';
```

* Ed / ED → edit the Query in Notepad then save it.

put forward slash enter.

NOTE: If we want to edit the query just type ED/ed compiler automatically take us to the notepad. In notepad edit the query but don't terminate the query by using semicolon, rather type ctrl+s or save the query and exit the notepad. automatically the edited query will by copy pasted to the software after that put forward slash.

Q. WAQTD Details of the employee having commission more than salary.

```
select *
From emp
where comm > sal ;
```

Operators:

1. Arithmetic operators (+, −, *, /)
2. Comparison operator (=, !=)
3. Relational operators (<, >, <=, >=)
4. logical operator (AND, OR, NOT)
5. Concatenation operator (||)
6. Special operator (IN, NOT IN, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE, IS, IS NOT)
7. Sub Query operator (ALL, ANY, EXIST, NOT EXISTS)

Logical operator:

AND operator:

- Binary multiplication.
- AND operator returns if both the condition are True
- AND operator should always be used between conditions.

0 → F
1 → T

Input		output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

I/p		o/p
C_1	C_2	C
F	F	F
F	T	F
T	F	F
T	T	T

C_1 AND C_2

OR operator:

- Binary addition
- OR operator returns True if any one of the condition is satisfied (True)
- OR operator should always be used b/w conditions.

0 → F
1 → T

Input		output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

I/p		o/p
A	B	C
F	F	F
F	T	T
T	F	T
T	T	T

NOT operator:

- It is used Negation.

I/p	o/p
T	F
F	T

Q1. WAQTD Details of the employee working as clerk and earning less than 1500.

```
Select *
From emp
where Job='CLERK' AND sal<1500;
```

Q2. WAQTD Name & Hiredate of the employee working as managers in Dept 30.

```
select Ename, Hiredate
From emp
where Job='MANAGER' AND Dept=30;
```

Q3. WAQTD all the details of the employee working in Dept 30 or Analyst.

```
select *
From emp
where Dept=30 or Job='ANALYST';
```

Q5) WAQTD Name of the employees whose salary is less than 1100 AND their Designation is clerk.

```
select Ename
From emp
where sal<1100 AND Job='CLERK';
```

Q6 WAQTD Name and sal, Annual sal and DeptNo. If DeptNo=20 earning more than 1100 and Annual salary exceed 12000

```
select Ename, sal, sal*12 As "Annual sal", DeptNo
From emp
where DeptNo=20 ANDsal>1100 AND sal*12>12000;
```

Q7) WAQTD EmpNo. and Names of the employee working as manager in Dept 20.

```
select EmpNo, Ename
From emp
where Job='MANAGER' AND Dept=20;
```

Q 8) WAQTD Details of Employee working in Dept no 20 or 30.

```
select *
From emp
where Dept No = 20 OR Dept No = 30;
```

Q10) WAQTD details of employee working as president with salary of of Rupees 4000.

```
select *
From emp
where Job: 'PRESIDENT' AND Sal: 4000;
```

Q13) WAQTD name of employees working in dept 10, 20, 30, 40.

```
select Ename
From emp.
where dept: 10 OR dept: 20 OR dept: 30 OR
                              dept: 40;
```

Q14) WAQTD Details of employees with empNO. 7902, 7839

```
select *
From emp
where EmpNO: 7902 OR EmpNO: 7839;
```

Q WAQTD Names of Employees along with annual salary For the employees working as manager OR clerk into Dept 10 OR 30.

```
Select Ename, sal*12 as "Annual sal"
From emp
where (Job: 'MANAGER' OR Job: 'CLERK')
                    AND.
      (Dept = 10 OR Dept: 30)
```

Q WAQTD Details of the employee Along with Annual salary if they are working in Dept 30 as Salesman and their annual salary has to be greater than 1400.

```
Select emp.*, sal*12 as "Annual salary"
From emp
where DeptNo = 30 AND Job: 'SALESMAN' AND
      sal*12 > 1400;
```

Q. WAQTD Names of employees Hired after 81 into Dept 10 or 30.

```
Select Ename
From emp
where Hire Date > 31 Dec 1981 AND (Dept: 10 OR
                                   Dept: 30);
```

Q16. WAQTD Names of employees Hired after 81 and Before 87.

```
select Ename
From emp
where (HireDate > '31-Dec-81') AND (HireDate < '1-JAN-87');
```

Q. WAQTD The employee name of an employee who is working on Dept 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

```
select Ename
From emp
where Dept=10 OR Dept=20 OR ..... OR Dept=100;
```

* IN Operator's : (special operators)

- IN operator is a multi valued operator in which we can pass multiple values at RHS.
- i.e IN operator can accept multiple value at RHS
- IN return True, if any one of the condition satisfied.
- IN operators ~~normally~~ Allows the value present at LHS to be compared with all the values present at RHS.

Syntax:

```
column_name/expression IN (V1, V2, V3 ... Vn);
```

Q. WAQTD Details of employee working in Department 10, 20, 30, 40, 50, 60.

```
Select *
From emp
where DeptNO IN (10, 20, 30, 40, 50, 60);
```

Q. WAQTD Ename and Job of emp's who are working as manager or salesman.

```
Select Ename, Job
From emp
where Job IN ('manager', 'salesman');
```

*

NOTE: IN operator works as replacement for ~~OR~~ '=' operator and works as replacement for OR operator

NOT IN:

- NOT IN operator is similar to IN operator But it Rejects the value instead of selecting it.

Syntax:

```
column_name/expression NOT IN (V1, V2, V3 ... Vn);
```

Q. WAQTD Emp's Name excluding the emp's working in Dept 10 OR 20.

```
select Ename
From emp
where DeptNo. NOT IN (10,20);
```

	Enome	Dept No.
x	sheela	10
x	leela	20
x	meela	10
	Laila	30

```
select Ename
① From emp
② where DeptNO. NOT IN (10,20)
```

II
10 IN (10,20) → T → NOT → F
20 IN (10,20) → T → NOT → F
10 IN (10,20) → T → NOT → F
30 IN (10,20) → F → NOT → T

O/p: Laila.

Q1. WAQTD Emp's Name if They are working in Dept 20,30 AND hired after 1980

```
select Ename
From emp
where DeptNO IN (20,30) AND Hiredate > 31-Dec-1980;
```

Q2) WAQTD Details of the employee if they are working as president, manager, salesman

```
select *
From emp
where Job IN ('PRESIDENT','MANAGER','SALESMAN');
```

Q3) WAQTD Ename if emps are working in Dept=10 or 20 AND as president or analyst.

```
select Ename
From emp
where Dept IN (10,20) AND Job IN ('PRESIDENT','ANALYST');
```

Q4) WAQTD details of emp's Except who are working as salesman or analyst.

```
select *
From emp
where Job NOT IN ('SALESMAN','ANALYST');
```

Q. WAQTD all the detail of employee who is earning salary more than 1250 and salary less than 3000.

```
select *
From emp
where sal > 1250 AND sal < 3000;
```

BETWEEN OPERATOR:

* Between operator is used when ever we have ranges.
* Between operator works including the Ranges.
* The range can't be interchanged.

Syntax:

```
Column_name/expression BETWEEN lower_range AND
                               higher_range;
```

Q. WAQTD emp's name and sal who are earning salary more than 1250 and less than 3000.

```
select ename, sal
From emp
where sal Between 1251 and 2999;
```

2) WAQTD Ename, if Emps are earning salary Between 1250 and 3000.

```
select Ename
From emp
where sal Between 1250 AND 3000;
```

3) WAQTD Ename and sal and comm if emp earning comm in the range of 300 to 1000.

```
select Ename, sal, comm
From emp
where comm Between 300 AND 1000;
```

4) WAQTD Ename, hiredate who were hired after 1981 and befor 1987 ?

→
```
select ename, hiredate
From emp.
where hiredate BETWEEN 01-JAN-1982 AND
      31-DEC-1986.;
```

5) WAQTD Ename, hiredate who were hired in the year 1982

→
```
select ename, hiredate
From emp
where hiredate BETWEEN 01-JAN-1982
      AND 31-DEC-1982;
```

Q.) WAQTD Ename, hiredate except who were hired in the year 1980.

→ select Ename, hiredate
From emp
where hiredate NOT BETWEEN
01-JAN-1980 AND 31-DEC-1980;

Q) WAQTD Ename who is not earning any commission.

→ select Ename
From emp
where comm. is Null;

⇒ **Is operator:**

* Is operator is used only to compare with null.

syntax:

column-name / expression is null;

Q.) WAQTD the ename who are earning commission?

→ select Ename
From emp
where comm is NOT null;

Q.) WAQTD Ename who is not earning any sal.

Select Ename
From Emp
where sal is Null;

⇒ **IS-NOT**

- Is Not operator is similar to Is operator but it reject the value instead of selecting it.
- syntax:

column-name / expression IS-NOT NULL;

Q) WAQTD details of emp who are not represent to any manager.

→ select *
From emp
where ~~Ename~~ ~~is~~ ~~NOT~~ Is Null;
manager

Q. WAQTD details of emp who are ~~not~~ represented manager.

```
select *
From emp
where manager IS NOT null;
```

Q. WAQTD details of emp who was hired after 1981 but before 1984.

```
select *
From emp
where hiredate BETWEEN 01-JAN-1982
                   AND 31-DEC-1986.;
```

WAQTD the Ename starting with A.

```
select Ename
From emp
where Ename like 'A%';
```

'A%'

Like operator:

- Like operator is used when ever we need to "match the pattern".

- Syntax:

```
column.name/expression LIKE 'pattern_to_match';
```

- To achieve the pattern matching we use special character such as

1. percentile (%): It can ~~also~~ accept any character, Any NO. of time or NO characters.

2. Underscore (_):

- It can any Accept character But only once.

Q) WAQT Ename which has having character A at last.

```
Select Ename
From emp
where Ename like %A ;
                 %A ;
```

% A ;

Q) WAQTD Ename having character A.

```
Select Ename
From emp
where Ename like '%A%' ;
```

Q) WAQTD the Ename who has having two consecutive LL in their name

```
Select Ename
From emp
where Ename like '%LL%' ;
```

Q) WAQTD the Ename who having two A's in their name

```
select Ename
From emp
where Ename like '%A%A%' ;
```

Q. WAQTD Ename who has having character A in the second place.

```
→ select Ename
  From emp
  where Ename like '_A%' ;
```

Q. WAQTD Details of the emp having 's' in the last place.

```
→ select *
  From emp
  where Ename like '%S' ;
```

Q. WAQTD Ename of Emp's having 'E' in the 4th place.

```
→ select Ename
  From emp
  where Ename like '___E%' ;
```

Q. WAQTD Ename who are having character 'A' in First place and s in the last place.

```
→ select Ename
  From Emp
  where Ename like 'A%S' ;
```

Q. WAQTD Ename, sal if they are earning 3 digits salary.

→ select Ename, sal
From emp
where sal like '__$_3$__$_2$__$_1$';

Q. WAQTD Ename, Hiredate if they were hired in the year 82.

→ select Ename, Hiredate
From emp
where hiredate between '1-Jan-82' AND '31-Dec-82'
OR
where hiredate like '%82';
OR
where hiredate like '__-___-82';

⇒ NOT LIKE:

- It is similar to like operator but rejects the value instead of selecting it.

Syntax:

column-name/expression NOT LIKE 'pattern-to-match';

Q. WAQTD Ename ~~the~~ except who are having character 'A' in their name.

→ select Ename
From emp
where Ename NOT LIKE '%A%';

Q. WAQTD Ename except who are having character 'S' in their last name.

→ select Ename
From emp
where Ename NOT like '%S';

Q. WAQTD Ename, Hiredate Except those who were hired in the Year 81.

→ select Ename, Hiredate
From emp
where hiredate NOT like '%81';

Q. WAQTD detail of emp who are having character 'A' in 1st place or emps who are having character 'S' in the last place.

→ select *
From emp
where Ename LIKE 'A%' OR Ename like '%S';

Q. WAQTD Ename if they are having 'A' in first place and working in dept 10 or 20.

→ select Ename
From emp
where Ename like 'A%' AND DeptNO IN(10,20);

Q. WAQTD Ename, sal if they are earning 4 Digit sal.

→ select Ename, sal
From emp
where sal like '____';

Q. WAQTD Ename hiredate if they are hired in the Year 81 and earning sal more than 2000.

→ select Ename, hiredate
From emp
where hiredate like '%81' AND sal > 2000;

Q. WAQTD detail of emp if they having string 'MAN' in Job.

→ select *
From emp
where Job like '%MAN%';

ESCAPE:

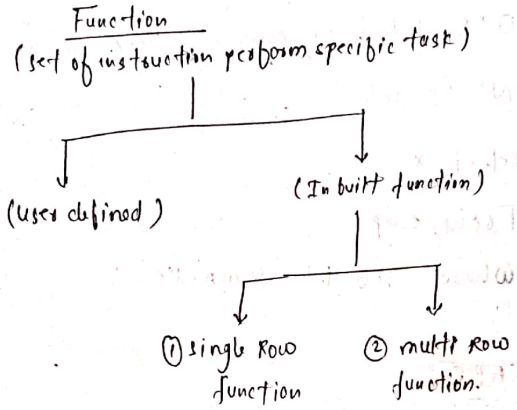
- Escape character is used to remove the special character which is present next to it.

Syntax:

```
column-name / expression LIKE 'Pattern-to-match'
                    Escape 'char';
```

Function

- It is the list of Instruction that are used to perform a specific task.



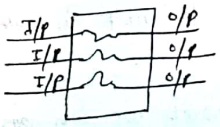
Single Row function

- Single Row function execute Row by Row.
- It take one input execute and generate one output then goes to the next ~~step~~ input.
- if we pass 'n' no. of input to single Row function. It return 'n' no. of output.

multi Row function:

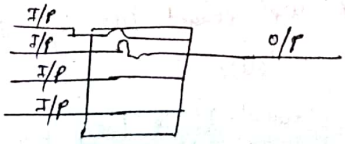
* multi Row function is also known as Group function or Aggregate function.
* It ~~execute~~ Group by Group.
* It take all the input once aggregate It compile and generate an output.
* If we pass 'n' no. of ~~output~~ Input to multiRow function it returns a single output.

SRF (single Row Function)



'n' I/p ⟶ 'n' o/p

MRF



'n' I/p ⟶ '1' o/p

List of the multi Row function/aggregate function:

1) MAX()
2) MIN()
3) SUM()
4) AVG()
5) Count()

NOTE!

* multi Row function can accept only a single argument that is column name an expression.
* MAX() and MIN() function can be used for all the following datatype problem.

  i.e char, varchar and Date.
* sum() and AVG() function can only take number column as an argument
* MRF ignore the null value.
* we cannot use any column name with MRF in select clause.
* Count() is the only MRF to which we can pass "As an argument".

Q. WAQTD number of employee getting salary less than 2000 in deptno 10.

```
Select count(*)
From emp
where Sal<2000 AND deptno=10;
```

Q WAQTD Number of emps working on dept 10.

```
Select count(*)
From emp
where dept=10;
```

Q. WAQTD Number of emps working in Dept 30.

```
Select count(*)
From emp
where dept=30;
```

Q WAQTD

```
select count(*)
From emp
where Dept IN
```

Group by Clause:

- We use group by clause to group the Records.
- It executes Row by Row.
- For Group by clause we can pass column name or an expression or an argument.

→ We can write Group by expression Along with multi-Row Function in select clause.

→ Group By expression :- Any column name or expression which is written in Group by ——— clause is known as Group by expression.

→ After the execution of group by clause it create group and if any clause.

→ Execute after group by clause it executes Group by Group....

Q. WAQTD the no. of employee working in each department.

Row by Row

I

Ename	Dept. No.
Ramya	10
Ashika	20
Tamannah	30
Karina	20
Kajal	10
Alia	20

Group by ⟶ Group the Records

II

```
select count(*), DeptNo.
From emp
Group by (Dept No.);
```

Group by expression.

III

10: Ramya, Kajal

30: Tamannah

20: Ashika, Karina, Alia

o/p → Group by clause.

IV o/p select.

count(*)	Deptno
2	10
3	20
1	30

Syntax:

Select group_by_expression / group_function
From Table_name
[Where <Filter_condition>]
Group by column_name/expression ;

order of execution:

1) From
2) where (if used) (Row by Row)
3) Group by (Row by Row)
4) select ~~(Row by Row)~~ → (Group by Group)

Q1) WAQTD Number of employee working in each department except president.

→ Select (*), DeptNo.
From emp
where Job NOT IN {'PRESIDENT'}
Group by deptNo ;

Q2) WAQTD Total salary needed to pay all the employees in Each Job.

→ select sum(sal), Job
From emp
Group by Job ;

Q5) WAQTD Number of employee having characters 'A' in their name in each Job

→ select count(*), Job
From emp
where Ename LIKE '%A%'
Group by Job ;

Q6) WAQTD Number of employee and Avg. salary needed to pay the whos salary is greater than 2000 In each department.

→ select count(*), ~~sum~~ Avg(sal)
From emp
where Sal > 2000
Group by department ;

Q7) WAQTD number of employee and total salary given to all the salesman in each dept.

```
select count(*), sum(sal)
From emp
where Job = 'SALESMAN'
group by deptNO ;
```

Q8) WAQTD Number of employees with their maximum salaries in each Job.

```
Select count(*), max(sal)
From emp
Group by Job ;
```

Q9) WAQTD maximum salaries given to an employee working in each dept.

```
select max(sal)
From emp
group by deptNO ;
```

Q10) WAQTD number of times the salaries are present in employee Table.

```
select count(sal), sal
From emp ;
Group by sal ;
```

Q11) WAQTD Number of employee working in each department having atleast 2 Emp's in each dept.

```
select count(*), DeptNo
From emp
Group by deptNO
having count(Ename) >= 2 ;
```

⟹ **Having Clause:**

- Filters the groups.

(I)

Ename	DeptNO
Rajkumar	10
Ajith	30
HRK	20
Allu	10
Dhoni	20
Baloyya	10

(II)

(4) select count(*), deptNO.
(1) From emp
(2) Group by DeptNO
(3) having count(*) >= 2 ;

(III) o/p Group by

10: Rajkumar 10, Allu 10, Baloyya 10
20: HRK 20, Dhoni 20
30: Ajith 30

(IV) Having clause : count(*) >= 2

3 >= 2 ✓ (Rajkumar 10, Allu 10, Baloyya 10)

2 >= 2 ✓ (HRK 20, Dhoni 20)

(V) o/p select

count	deptno
3	10
2	20

Having Clause:

- We use having clause to filter the group.
- We can pass multi-Row function codintion in having clause.
- It execute group by group
- If using having clause it should be used after group by clause.
- It cannot be used without group by clause.

Syntax:

```
Select group-by-expression / group-function
From Table-name
[where < filter condition > ]
Group by column-name / expression
having < group-filter-condition >
```

Group By - expression condition / multi-Row-function condition.

ORder of execution:

1. From
2. where (if used) (Row by Row)
3. Group by (Row by Row)
4. having (Group by Group)
5. Select (Group by Group)

Q11) WAQTD Number of employie hired on the same day in to the same department.

```
→ select count(*), hiredate, DeptNO
  From emp
  Group by hiredate, deptNO;
```

Q12) WAQTD Number of employees getting the same salary in the same department.

```
→ select count(*), sal, DeptNO
  From emp
  Group by sal, DeptNO;
```

Q2) WAQTD DeptNo and total salary needed to pay all emp in each dept. if they are atleast 4 emp in each dept.

```
→ select DeptNO, Sum(sal)
  From emp
  Group by deptNO
  having count(*) >= 4;
```

Q3) WAQTD Number of Emp earning sal more than 1200 in each Job and the total sal needed to pay of each Job must exceed 3800.

→ Select count(*), ~~sum(sal)~~ Job
From emp
where sal > 1200
Group by Job
having sum(sal) > 3800 ;

Q4) WAQTD DeptNO and Number of emp working only if there are 2 Emp working in each dept as manager.

→ select count(*), DeptNO.
From emp
where Job = 'MANAGER'
Group by deptNO.
having count(*) = 2 ;

Q5) WAQTD JOB and max sal of emp in each Job if the max sal exceeds 2600.

→ select Job, max(sal)
From emp
Group by Job
having max(sal) > 2600 ;

Q6) WAQTD The salaries which are repeated in emp table.

→ Select count(*), sal
From emp
Group by sal
having count(*) > 1 ;

Q7) WAQTD The hiredate which are duplicated in emp table

→ select hiredate
From emp
Group by hiredate
having count(*) > 1 ;

Q8) WAQTD AVG salary of each dept if avg sal is less than 3000.

→ select Avg(sal), DeptNO
From emp
Group by deptNO
having Avg(sal) < 3000 ;

Q10) WAQTD min and max salaries of each Job If min sal is more than 1000 and max sal is less than 5000.

→ select min(sal), max(sal), Job
From emp
Group by Job
having min(sal) > 1000 AND max(sal) < 5000;

~~Q10) WAQTD MIN and MAX Salaries of each Job if min sal is more than~~

Q9) WAQTD DeptNO if there are atleast 3 emp in each dept whos name has char 'A or S'.

→ Select DeptNO
From emp
where Ename like '%A%' OR Ename like '%S%'
Group by deptNO
having count(*) > 3 ;

Q11) WAQTD Ename who is earning more than smith.

→ select Ename
From emp

Difference b/w where clause and having clause.

Q) WAQTD empname. whos earning more than Teju.

(I)

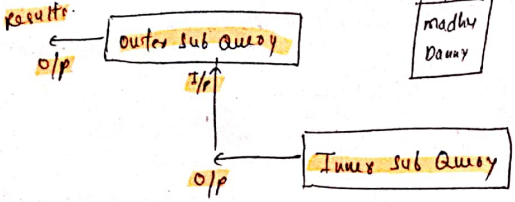
Ename	Sal
Bipasha	1000
Lucky	2000
Teju	2500
madhu	3000
Danny	3500

(II)

⑥ select Ename
④ From emp
⑤ where sal > ③ select sal (2500)
① From emp
② where Ename = 'Teju'

(III)

o/p
madhu
Danny



Sub Query:

- A Query which is written inside another Query is known as Sub-Query.

Working procedure:

→ There we will be having minimum of 2 queries.

1) Outer
2) Inner Query / sub Query.
3) Inner Query will execute first and generate the output.
4) The o/p generated by the inner query will be given as I/p to the outer query.

→ The outer Query will execute and generated the o/p.

→ This o/p will be the Result.

→ By this we can say the outer query is dependent on inner query.

Q) WAQTD a emp name who is earning more than Dany.

→
```
select empname
From emp
where sal > ( select sal
              From emp
              where Ename = 'Dany' ; )
```

WHY SUB-Query:

Cases: When ever we have unknown.

Q1) WAQTD Name of the employee earning more than ADAMS.

```
→ select Ename
  From emp
  where sal > (select sal
               From emp
               where Ename='ADAMS');
```

Q2) WAQTD Name and salary of the employees earning less than King.

```
→ select Ename, Sal
  From emp
  where Sal < (select Sal
               From emp
               where Ename='KING');
```

Q3) WAQTD Name and DeptNO of the employee if they are working in the same Dept as JONES.

```
→ select Ename, DeptNO
  From emp
  where DeptNO = (select DeptNO
                  From emp
                  where Ename='JONES');
```

Q4) WAQTD Name and Job of all the employee working in the same designation as James.

```
→ select Name, Job
  From emp
  where Job = (select Job
               From emp
               where Ename='James');
```

Q5) WAQTD EmpNO and Ename along with annual salary of all the employee if their annual salary is greater than ward's annual salary.

```
→ select EmpNO, Ename, sal*12 as 'Annual sal'
  From emp
  where sal*12 > (select Sal*12
                  From emp
                  where Ename='WARD');
```

Q6) WAQTD Name and Hiredate of the employee if they are hired before Turner.

```
→ select Ename, hiredate
  From emp
  where hiredate < (select hiredate
                    From emp
                    where ename='TURNER');
```

Q7) WAQTD Name and hiredate of the employ. if they are hired after the President.

```
→ select Ename, hiredate
  From emp
  where hiredate > ( select hiredate
                     From emp
                     where Job='President' );
```

Q8) WAQTD Name and sal of the employee if they are earning sal less than the employee whos empNO is 7839.

```
→ select Ename, sal
  From emp
  where sal < ( select sal
                From emp
                where empNO = 7839 );
```

Q9) WAQTD All the detail of the employees if the employees are hired before MILLER.

```
→ select *
  From emp
  where hiredate < ( select hiredate
                     From emp
                     where Ename='MILLER' );
```

Q10) WAQTD Ename and empno of the employees ~~if they are~~ if employees are earning more than ALLEN.

```
→ select Ename, empno
  From emp
  where sal > ( select sal
                From emp
                where ename='ALLEN' );
```

Q11) WAQTD Number of employees hired after KING.

```
→ select count(*)
  From emp
  where hiredate > ( select hiredate
                     From emp
                     where Ename='KING' );
```

Q12) WAQTD Total salary given to the employees working in the same dept as WARD

```
→ select  sum(sal)
  From emp
  where
  group by dept
  having count(*) > 1 ;
```

```
→ select sum(sal)
  From emp
  where deptno = (
  select deptno
  From emp
  where Ename='WARD'
  );
```

Q) WAQTD Ename and salary of all the employees who are earning more than miller but less than ALLEN.

→ select Ename, sal
From emp
where sal >
(select sal
From emp
where Ename like 'MILLER');
AND sal < (select sal
From emp
where Ename like 'ALLEN');

NOTE:
(*Use Like and In operator in place of equal)

Q12) WAQTD all the detail of employee working in dept no 20 and working in the same designation as SMITH.

→ select *
From emp
where deptNO = 20 And Job =
(select Job
From emp
where ename like 'SMITH');

Q13) WAQTD All the detail of the employee working as manager in the same dept as TURNER.

→ select *
From emp
where Job = 'MANAGER' AND deptNO =
(select deptno
From emp
where Ename like 'Turner');

Q14) WAQTD Name and Hiredate of the employees Hired after 1980 AND befor KING.

→ select Name, hiredate
From emp
where Hiredate > '31-DEC-1980' AND Hiredate <
(Select hiredate
From emp
where Ename like 'KING');

Q15) WAQTD Name and sal along with annual sal for all emp whose sal is less than Blake or emp's earning more than more than 3500.

→ select Name, sal, sal*12 as 'annual sal'
From emp
where sal < (select sal
From emp
where ename like '%BLACK%';)
OR
sal > 3500;

Q16) WAQTD All the detail of employee who earn more than SCOTT but less than king

Select *
From emp
where sal > select sal AND sal< select
From emp
where ename like 'SCOTT';

Q17) WAQTD Name of the employee whos name start with 'A' and work in the same Dept as Blake.

→ select Ename
From emp
where Ename like 'A%' and Deptno = Select Deptno
From emp
where ename like 'BLAKE';

Q18) WAQTD Name and comm if employees earn commission and work in the same designation as SMITH.

→ Select Ename, comm
From emp
where Job = Select Job
From emp
where ename like 'SMITH';
AND comm = select comm
From emp
where ename like 'SMITH';
where comm is NOT NULL and Job = select Job
From emp
where Ename = 'SMITH';

Q19) WAQTD Detail of all employee working as CLERK in the same dept as Turner

```
→ select *
  From emp
  where Job like 'CLERK' AND
        dept no the select deptno
                 From emp
                 Where ename='Turner';
```

Q20) WAQTD Ename, sal and Designation of the employee whos annual salary is more than SMITH and less than KING.

```
→ Select Ename, sal, ~~Desig~~ Job
  From emp
  where sal*12 > select sal*12
                 From emp
                 where ename in 'SMITH';
                 AND
        sal*12 < select sal*12
                 From emp
                 where ename like 'KING';
```

Q. WAQTD DeptNames of an employee whos name is GITA.

⇒ Emp

Enames	DeptNo
Nita	10
Sita	30
Gita	20
Rita	10

Dept

DeptNO	DeptNames
10	Sales
20	Accounting
30	HR

```
→ ⑤ select DeptNames
  ④ From emp
  ⑥ where DeptNO IN ③select DeptNO
                    ① From emp
                    ② where Enames = 'GITA';)
```

Q. WAQTD the DeptName of an employee NITA

```
Select DeptNames
From emp
where DeptNO IN ( select Dept No.
                  From dept
                  where ename = 'NITA'; )
```

Q) WAQTD the emp name who is working in DeptName is HR

```
select Ename
From emp
where DeptNO IN(select DeptNo
                From Dept
                where DeptName='HR';)
```

Q21) WAQTD Dname of the employee whos name is SMITH.

→
```
select Dname
From dept
where DeptNO IN select DeptNo
                From ~~emp dept~~ emp
                where Ename='SMITH';
```

Q22) WAQTD Dname and loc of the employee whos Ename is KING

→
```
select Dname, loc
From Dept
where DeptNO IN select DeptNo
                From emp
                where ename IN'KING';
```

Q24) WAQTD Dname and loc along with DeptNo of the employee whos name end with 'R'

→
```
select Dname, loc, DeptNo
From Dept
where DeptNO IN ~~Dept~~ select DeptNo
                From emp
                where ename like '%R';
```

Q28) WAQTD details of the employees working in Sales → DeptName

→
```
select *
From emp where DeptNo IN (select DeptNo
~~where Job like 'SALESMAN';~~ From Dept
                where Dname
                like 'sales';)
```

Q27) WAQTD Ename & salaries of the employee who are working in the location 'CHICAGO'

→
```
select Ename, Sal
From emp
where DeptNO IN (select DeptNo
                From Dept
                where Loc='CHICAGO';)
```

Case-II.

Case-II problem deals with whenever data need to be selected and condition need to be executed But data present in two different table go for sub query.

Q31) WAQTD Name of the employee earning more than SCOTT IN accounting Dept.

```
Select Ename
From emp
where sal > select sal  AND  DeptNO IN select
          From emp                    DeptNo
          where Ename like 'SCOTT'    From dept
                                      where
                                      Dname = 'Accounting';
```

Q32) WAQTD Details of the employee working as manager and working in location chicago

```
→ ~~From~~ select *
  From emp
  where Job IN 'MANAGER' AND
                 select
        DeptNO IN DeptNO
        From Dept
        where loc = 'CHICAGO';
```

Q33) WAQTD Name and salary of the employee Earning more than KING in the dept Accounting.

```
→ select ename, sal
  From emp
  where sal > (select sal      AND  DeptNo In
               From emp             select DeptNo
               where ename = 'KING';  From Dept
                                      where
                                      Dname = 'Accounting');
```

Q34) WAQTD Details of the employee working as salesman in the Department sales

```
→ select *
  From ~~dept~~ emp
  where ~~Job~~ = ~~'SALESMAN'~~ AND  DeptNo IN
        ~~Dname = 'sales'~~           select ~~DeptNo~~ DeptNo
        Job = 'salesman';             From dept
                                      where ~~Dname = 'sales'~~;
```

Q35) WAQTD Name, sal, Job, Hiredate of the employee working in operations department and hired before KING.

```
→ select ename, sal, Job, Hiredate
  From emp
  where ~~hiredate like~~ deptNO IN select deptNO
                                   From dept
  Hiredate< select Hiredate  AND   where dname = 'Operation';
            From emp
            where ename = 'KING';
```

Q36) Display all the employees whose department name ending with 'S'.

→
```
select *
From emp
where deptNo. IN select deptNO
                From dept
                where dname like '%S';
```

Q37) WAQTD Dname of the employee who names has character 'A' in it

→
```
select Dname
From Dept
where deptno. IN select deptno
                Foom emp
                where ename like '%A%';
```

Q38) WAQTD Dname ~~of the~~ And loc of the employee whos salary is Rupees 800.

→
```
select dname , loc
From dept
where dept no IN select dept No.
                From emp
                where sal = 800 ;
```

Q39) WAQTD Dname of the employee who earn commission.

→
```
select Dname
From Dept
where deptno IN select deptNO
                From emp
                where comm > 0;
                or
                comm is NOT NULL;
```

Q40) WAQTD loc of the employee if they are earn commission in dept 40.

→
```
select loc
From dept
where deptNO IN select deptno      AND deptNO=40;
                From emp
                where comm > 0;
                      or
                comm is NOT Null;
```

⇒

Ename	Sal
Radhika	2000
Sam	1000
Deepika	5000
Tanmoush	3000
Ashika	4000

Q) WAQTD the max(sal) from emp table

→ select max(sal)
From emp;

Q) WAQTD 2nd max ~~table~~ sal from emp table.

→ select max(sal)
From emp
where sal < select max(sal)
From emp;

Q) WAQTD 3rd max sal from emp table.

→ select max(sal)
From emp
where sal < (select max(sal)
From emp
where sal < (select max(sal)
From emp));

Q) WAQTD the Ename who is earning 2nd max salary.

→ select Ename, max(sal)
From emp
where sal < (select max(sal)
From emp);
Group by max(sal)

→ select Ename
From emp
where sal IN (select max(sal)
From emp
where sal < (select max(sal)
From emp));

Q) WAQTD the minm salary from emp table

→ select min(sal)
From emp;

Q) WAQTD the 2nd minm salary from emp table

→ select min(sal)
From emp
where sal > (select min(sal)
From emp);

Q43) WAQTD Name and hiredate of the employee hired before all the employees

→ select ename, Hiredate
From emp
where hiredate IN(select min(Hiredate)
From emp);

Q65) WAQTD Departname of an employee Getting and max sal.

→ select Dname
From Dept
where Deptno IN sal In (select max(sal)
From emp
where sal <(select max(sal)
From emp));
select Deptno
From emp
where

Q44) WAQTD Name and Hiredate of the employees hired at the last.

→ select Ename, Hiredate
From emp
where hiredate IN(select max(Hiredate)
From emp);

Q45) WAQTD Name, comm of the employee who earns min comm.

→ select Ename, comm
From emp
where comm IN (select MIN(comm)
From emp);

Q46) WAQTD Name, sal and comm of the employee Earning maximum commision.

→ select ename, sal, comm
From emp
where comm IN(select max(comm)
From emp);

Q47) WAQTD Details of the employee who has greatest EmpNo.

→ Select *
From emp
where EmpNo IN (select Max(EmpNo)
From emp);

Q48) WAQTD details of the employee having the least hiredate.

→ select *
From emp
where hiredate IN (select MIN(hiredate)
From emp);

Q49) WAQTD detail of the emp earning least annual salary.

→ select *
From emp
where sal*12 IN (select MIN(sal*12)
From emp);

Q50) WAQTD details of the employee who was hired end.

→
Select *
From emp
where hiredate IN (select hiredate
From emp
where hiredate IN select MIN(hiredate)
From emp
where hiredate IN select MIN(hiredate)
From emp)

Q51) WAQTD Name of the emp hired before the last employee.

→ select Ename
From emp
where hiredate < (select hiredate
From emp
where hiredate IN select max(hiredate)
From emp
where hiredate IN select max(hiredate)
From emp)

Q52) WAQTD loc of the employee who was hired First

→
select loc
From Dept
where Deptno IN select DeptNO
From emp
where hiredate IN (select hiredate
From emp
where hiredate IN select MIN(hiredate)
From emp)

Q53) WAQTD details of the employee earning 7th minimum salary.

⟹

```
Select *
From emp
where sal IN
select MIN(sal) (4)
From emp
where sal IN
select MIN(sal) (6)
From emp
where sal IN
select MIN(sal)
From emp
where sal IN select MIN(sal)
        From emp
        where sal IN select MIN(sal)
                From emp
                where sal IN select MIN(sal)
                        From emp
                        where sal IN select MIN(sal)
                                From emp
```

Q70) INAQTD Dname of employee getting and maximum salary.

```
→ select Dname
  From Dept
  where Deptno IN select Deptno
                  From emp
                  where sal IN select max(sal)
                                Fromemp
                                where sal IN select max(sal)
                                             Fromemp
```

Nested Sub Query:

- A sub Query written inside another sub-Query is known as Nested-sub Query.
- We can nest about 255 sub-Queries.

Type of subQuery:

There are two Type

1. single Row sub-Query
2. multi Row sub Query

1. single Row sub-Query:

- A sub Query which return exactly one o/p as single Row sub-Query. we can use operator's such as IN, NOTIN, ALL, ANY.

ex- INAQTD Ename earning more than SCOTT.

```
→ select Ename
  From emp
  where sal > select sal
              From emp
              where ename = 'SCOTT';
```

Multi Row sub-Query:

- A sub Query which returns more than one output known as multi Row sub-Query. We must use operator's such as IN, NOTIN, ALL, ANY.

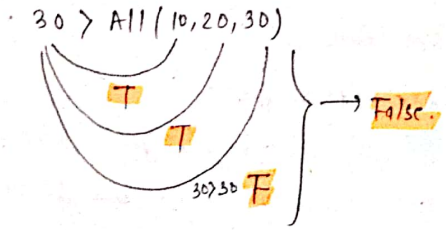
ALL operator:

- All operator is a special operator which can accept multiple values at RHS. It will return true only if the condition at RHS is satisfied.

Syntax:

Column_name / Expr.. RelationalOPr. All (V1, V2, .. Vn);

Ex: All



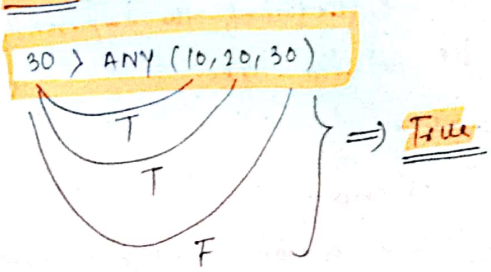
ANY operator:

- Any operator is a special operator which can accept multiple values at RHS. It will Return true if any one of condition at RHS is satisfied.

Syntax:

Column_Name / expr.. Relational OP ANY (V1, V2, V3, .. Vn);

Ex: ANY = OR



Q) WAQTD EmpName who is earning more than * all the salesman

```
Select Ename
From emp
where sal > All (select sal
                 From emp
                 where Job IN 'SALESMAN');
```

Q) WAQTD ename & sid if is earning more than any of salesman.

→ select ename
From emp
where sal > ANY select sal
From emp
where Job IN 'SALESMAN';

Q52) WAQTD Detail of the employee hired after all the clerks

→ select *
From emp
where hiredate > All select hiredate
From emp
where Job IN 'CLERK';

Q53) WAQTD Name and salary of all the employee if they are earning less than atleast a MANAGER

→ select ename, sal
From emp
where sal < ANY select sal
From emp
where Job = 'manager';

Q54) WAQTD name and hiredate of employee hired before all the managers

Select ename, hiredate
From emp
where hiredate < All select hiredate
From emp
where Job IN 'MANAGER'

Q55) WAQTD Names of the employee hired after All the managers and earning salary more than All the clerks.

→ Select ename
From emp
where hiredate > All select hiredate
From emp
where Job = 'MANAGER' AND

sal > All select sal
From emp
where Job IN 'CLERK';

Q56) WAQTD Details of the employee working as clerk and hired before atleast a salesman.

→ select *
From emp 'CLERK'
where Job IN (select Job From emp where) AND hiredate < ANY (select hiredate From emp where Job IN 'salesman';

Q57) WAQTD details of the employee working IN accounting OR sales Dept.

→ select *
From emp
where DeptNO IN select DeptNO From Dept where Dname='sales'; OR
select DeptNO From dept where Dname=('sales', Accounting)
select DeptNO From Dept where Dname='Accounting';

Q58) WAQTD Department names of the employee ~~working in accounting or sales dept~~ with name SMITH, KING and MILLER

→ Select Dname
From Dept
where Depart no. IN (select DeptNO From emp where Enames='KING' AND ANY (select DeptNo From emp where Enames='MILLER'; AND (select DeptNO From emp where Enames='SMITH')))
select DeptNo From emp where Enames IN ('KING','SMITH','MILLER')

Q59) WAQTD Details of the employee working Newyork or chicago

→ select *
From emp
where deptno IN select Deptno From dept where loc='chicago' OR deptno IN select deptNO From dept where loc='Newyork'
select DeptNo From dept where loc IN ('chicago','Newyork')

Q60) WAQTD Emp Names if employee are hired after all the employee of dept 10.

→ Select Ename
From emp
where hiredate > ANY select hiredate From emp where deptNO = 10;

Q) WAQTD the manager name of Ritu

Empno	Ename	mgr
1	Sunny	2
2	Sweta	4
3	Rithu	4
4	Anu	5
5	Divya	—

⑥ select Ename
④ From emp
⑤ where Empno = ③select mgr
① From emp
② where ename = 'Rithu'

o/p => ANU

Q) WAQTD the manager name of Sunny.

select Ename
From emp
where empno = select mgr
From emp
where ename = 'SUNNY';

o/p = sweta.

Q) WAQTD the employee name who is reporting to divya.

select ename
From emp
where mgr = select empno
From emp
where ename = 'divya';

Emp-MGR

Q71) WAQTD SMITH reporting manager Name

→ select ename
From emp
where empno = select mgr
From emp
where ename = 'SMITH'

Q74) WAQTD MILLER MANAGER Salary

select sal
From emp
where empno = select mgr
From emp
where ename = 'MILLER';

Q72) WAQTD Adams manager manager Name.

where
From emp
where mgr In Select empno
From emp
where ~~select~~ mgr IN Select empno
~~From emp~~ From emp
~~where~~ where ename = 'ADAMS'
~~empno~~

~~Select mgr~~

→ Select ename
From emp
where Empno = select mgr
From emp
where empno = select mgr
From emp
where ename = 'ADAMS'

Q73) WAQTD Dname of JONES MANAGERS

→ select Dname
From Dept
where Deptno IN select Deptno
From emp
where ~~deptno~~ empno = select mgr
From emp
where ename = 'JONES'

JOIN's

* This statement is used to retrive the data from multible tables simultaneously

Types

1) Cartesian Join or ~~Equi~~ Cross Join
2) Inner Join or Equi Join
3) outer Join
   a) Right outer Join
   b) left outer Join
   c) full outer Join
4) Self Join
5) Natural Join

1) ~~cartesian join or~~ Inner Join :

Syntax :

1. ANSI : select column_name
From Table_Name1 INNER JOIN Table_Name2
ON <JOIN-CONDITION>;

2. ORACLE :
SELECT column_Name
From Table_Name-1, Table_Name-2
where <JOIN-condition>;

ANSI

```
select *
From EMP Inner Join Dept
ON Emp.DeptNO = Dept.DeptNO ;
```

ORACLE

```
select *
From EMP, Dept
where emp.deptno = Dept.Deptno ;
```

Q1. Name of the employee and his location of all the employee

→

```
select ename, loc
From emp, dept
where emp.deptNO = dept.deptNO ;
```

Ex: Inner Join

Boys

BID	Bname	GID
1	Yash	22
2	Vedant	33
3	Dhoni	44

Girls

GID	Gname
33	Saksi
22	Radhika
44	Anushka

o/p: of where clause

BID	Bname	GID	GID	Gname
1	Yash	22	22	Radhika
2.	Vedant	33	33	Saksi
3.	Dhoni	44	44	Anushka

Actual o/p:

Bname	Gname
Yash	Radhika
Vedant	saksi
Dhoni	Anushka

```
(3) Select Bname , Gname
(1) From Boys, Girls
(2) where Boys.GID = Girls.GID ;
```

1. Cartesian Join or Cross Join :

- In cartesian join a record from table-1 will be merged with all the records of table-2.
- No. of column in result table will be sum of column present in table1 and table-2.
- No. of ~~colum~~ records in result table will be Product of records present in table-1 and table-2.
- In this Join we will be getting error records.

Syntax:

1. ANSI:

```
Select column-name
From Table-name1 Cross Join Table-name2.
```

2. oracle

```
select column-name
From Table-name1, Table-name2;
```

Ex- Let us consider two table Boys and Girls

BID	Bname	GID
1	Yash	22
2	Virat	33
3	Dhoni	11

GID	Gname
11	Sakshi
22	Radhika
33	Anushka

Error Records

O/P

BID	Bname	GID	GID	Gname
1	Yash	22	11	Sakshi
1	Yash	22	22	Radhika
1	Yash	22	33	Anushka
2	Virat	33	11	Sakshi
2	Virat	33	22	Radhika
2	Virat	33	33	Anushka
3	Dhoni	11	11	Sakshi
3	Dhoni	11	22	Radhika
3	Dhoni	11	33	Anushka

Q. WAQTD Bname, Gname from Boys table and Girls Table.

2) Inner Join: or (Equi Join):-

- We use inner Join to obtain only the ~~method~~ matched records or the records which is pair.
- We use Join condition to obtain the matched records.

Join condition:

~~We use~~ Table_Name1.column-name = Table_Name2.column-name.

- It is a condition which is used to merge two table to get only the matched records.

Table_Name1.col_name = Table_Name2.col_name;

Ex- Emp.Deptno = Dept.Deptno

Ex: Inner Join:

Boys

BID	Bname	GID
1	Yash	22
2	Virat	33
3	Dhoni	11

Girls

GID	Gname
11	Sakshi
22	Radhika
33	Anushka

O/P: of where clause:

BID	Bname	GID	GID	Gname
1	Yash	22	22	Radhika
2	Virat	33	33	Anushka
3	dhoni	11	11	Sakshi

Outer Join:

- In outer Join we get the unmatched records along with the matched records.

1. Left outer Join:

- In left outer join we get unmatched records of left table along with matched records.

1. ANSI : Select column_Name
From Table_Name1 LEFT[outer] JOIN
Table_Name2
ON <JOIN_CONDITION>;

Ex: Select *
From emp E left outer Join dept D
ON E.DeptNO = D.DeptNO;

2. ORACLE: Select column Name
From Table_Name1, Table_Name2
where Table_Name1.col_Name =
Table_Name2.col_Name(+);

Ex: Select *
From Emp E, Dept D
where E.DeptNO = D.DeptNO(+);

Emp

Ename	DeptNo.
A	20
B	Null
C	30
D	Null
E	10

Dept

DeptNo.	Dname
10	D_1
20	D_2
30	D_3
40	D_4

Q. WAQTD all the details from emp and dept table along with unmatched records from emp table.

→ Select *
From emp E, Dept D
where E.deptNO = D.deptNO(+)*;

o/p:

Ename	DeptNo	DeptNo	Dname
A	20	20	D_2
C	30	30	D_3
E	10	10	D_1
B	Null	Null	Null
D	Null	Null	Null

2. RIGHT OUTER JOIN:

- In Right outer join we get unmatched records of right Table along with matched ~~records~~ records

ANSI: Select column_Name
From Table_Name1 Right [OUTER] Join
Table_Name2
ON <JOIN_CONDITION>

ex: Select *
From emp E Right outer join DeptD
ON E.DeptNO = D.DeptNO;

ORACLE: select column_Name
From Table_Name1, Table_Name2
where Table_Name1.col_Name(+) =
Table_Name2.col_Name;

Ex: Select *
From Emp E, Dept D
where E.DeptNo(+) = D.DeptNo;

Q. WAQTD all the details from ~~emp~~ or matched Records from emp and ~~magic~~ dept table along with unmatched record from dept table.

→ Select *
From Emp E, Dept D
where E.DeptNo = D.DeptNo;

o/p

Ename	DeptNo	DeptNo	Dname
A	20	20	D_2
C	10	10	D_1
E	30	30	D_3
Null	Null	40	D_4

matched

unmatched

---

3) Full outer Join:

- To obtain unmatched records of both the table along with matched Records.

1. ANSI: select column-name
From Table-Name1 Full [outer] Join
Table-Name2
ON <Join-conditions>;

Ex: select *
From Emp E Full outer Join Dept D
ON E.Deptno = D.Deptno;

Q. WAQTD the matched records from both emp and dept table along with unmatched records from both the table.

→ select *
From Emp E, ~~Full outer Join~~ ~~Dept D~~
~~select~~
~~ON~~

→ select *
From emp E Full outer Join Dept D
ON E.DeptNO = D.DeptNo;

o/p:

Ename	DeptNo	deptNo	Dname
A	20	20	D_2
C	10	10	D_1
E	30	30	D_3
B	Null	Null	Null
D	Null	Null	Null
Null	Null	40	Dy

} unmatched records from both table

Q. WAQTD Ename & his manager name From emp table of all the emp

Self Join:

- Self Join is used to Join the same two table or the table itself.
- WHY WE USE SELF JOIN?
  - ↳ If the Data to be selected and condition to be executed is present in same table but in different Record we use self Join.

Syntax:

ANSI: select column.Name
From Table.Name T_1 JOIN Table.Name T_2
ON <Join_condition>;

Ex: select *
From Emp E_1 Join Emp E_2
ON <Join_condition>;
↳ E_1.mgr = E_2.EmpNO;

ORACLE: select column_Name
From Table.Name T_1, Table.Name T_2
where <Join condition>;

Ex:
select *
From Emp E, Emp 2
where E_1.mgr = E_2.EmpNO;

Emp E_1

EmpNO	Ename	mgr
1	sheela	2
2	leela	4
3	maala	4
4	laila	5
5	shakila	—

Emp E_2

EmpNO	Ename	mgr
1	sheela	2
2	leela	4
3	maala	4
4	laila	5
5	shakila	—

→ ⑤ select E_1.Ename, E_2.Ename mgr.name
① From Emp E_1, Emp E_2
② where E_1.mgr = E_2.empNO;

o/p:

E_1.name	mgr.name
sheela	leela
leela	laila
maala	laila
laila	shakeela

Q1) WAQTD Name of the employee and his manager Name if employee is work as clerk.

→ select E_1.Ename, E_2.Ename mgr.Name
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_1.Job = 'CLERK';

Q2) WAQTD Name of the employee and manager Designation if manager work in dept 10 OR 20.

→ select E_1.name, E_2.Job
From Emp E_1, Emp E_2
where E_1.mgr = E_2.empNO AND E_1.DeptNo IN (10, 20);

Q2) WAQTD Emp Name and manager Hiredate if Employee was hired before 1982

→ select E_1.name, E_2.Hiredate
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND
E_1.hiredate < '01-Jan-1982';

Q3) WAQTD Name of the emp and manager Salary if employee and manager both earn more than 2300.

→ select E_1.name, E_2.sal mgr sal
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_2.sal > 2300 AND E_1.sal > 2300;

Q6) WAQTD Emp Name and manager Name and their salaries if employee earn more than their manager.

→ select E_1.Ename, E_2.Ename mgr_name, E_1.sal, E_2.sal
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_1.sal > E_2.sal;

Q7) WAQTD EmpName and Hiredate, manager Name Hiredate if manager was hired before employee.

→ select E_1.Ename, E_1.hiredate, E_2.Ename mgr_name, E_2.Hiredate
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_2.hiredate < E_1.hiredate;

Q8) WAQTD EmpName and manager Name if both are working in same Job.

→ select E_1.Ename, E_2.Ename mgr_Name
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_1.Job = E_2.Job

Doubt Q9) WAQTD EmpName and manager name if both manager is working as actual manager.

→ select E_1.Ename, E_2.Ename mgr_Ename
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_2.Job = 'MANAGER'

Q10) WAQTD EmpName and manager Name along with their annual salaries if employee works in Dept 10, 20 and manager sal is greater than emp sal.

→ select E_1.Ename, E_1.sal*12, E_2.Ename mgr_Name, E_2.sal*12
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND Deptno IN (10,20)
AND E_2.sal > E_1.sal;

Q11) WAQTD Ename and manager designation for all the employee.

→ select E_1.Ename, E_2.Job mgr_Job
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNo ;

Q12) WAQTD Employee Name and manager salary for all the employee if manager salary end with 50.

→ select E_1.Ename, E_2.sal mgr_sal
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNO AND E_2.sal like '%50';

Joining Multiple Table :

13. WAQTD Ename and Manager Name (2 Tables).

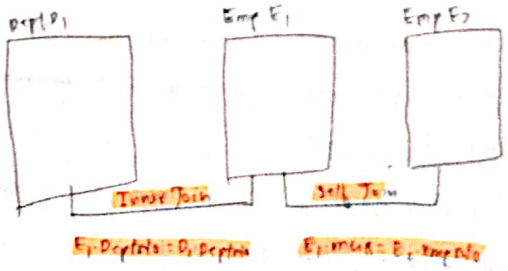
→ select E_1.Ename, E_2.Ename mgr.name
From Emp E_1, Emp E_2
where E_1.mgr = E_2.EmpNo ;

14. WAQTD Ename and Dname (2 Tables)

→ select [redacted].Ename, [redacted].Dname
From Emp E_1, Dept D_1
where E_1.DeptNo = D_1.Deptno.

15. WAQTD Ename, manager name and Emp Dname.

→ select E_1.Ename, E_2.Ename mgr_name, g.Dname
From Emp E_1, Emp E_2, Dept D_1
where E_1.mgr = E_2.EmpNO AND E_1.deptno = D_1.DeptNo ;



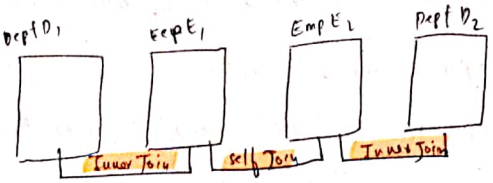
NOTE:
- If we want to Join 'n' no of table we will be writing (n-1) Join condition.
- we can Join upto 256 Tables using Inner Join

Q12. WAQTD Ename, manager Name and EmpDname and manager Dname.

→ select E_1.Ename, E_2.Ename mgr_name, D_1.Dname, D_2.Dname mgr_dname

From Emp E_1, Emp E_2, Dept D_1, Dept D_2

where E_1.mgr = E_2.EmpNO AND E_1.deptno = D_1.Deptno AND E_2.deptno = D_2.deptno.



Q13) WAQTD Ename, manager name and this Dname if emp earns more than 2000 and manager working in dept 20.

→ select E_1.Ename, E_2.Ename mgr_name, D_1.Dname, D_2.Dname

From Emp E_1, Emp E_2, Dept D_1

where E_1.mgr = E_2.EmpNO AND

E_1.Deptno = D_1.Deptno AND E_1.sal > 2000 AND E_2.Deptno = D_2.Deptno AND E_2.Dept IN 20;

Q13. WAQTD Ename, manager name and their Dname if emp earning more than smith and manager earning more than ALLEN.

→ select E_1.Ename, E_2.Ename mgr_name, D_1.Dname

From Emp E_1, Emp E_2, Dept D_1

where E_1.mgr = E_2.EmpNO AND ~~E1.sal~~

D_1.Deptno = E_1.deptNO AND

E_1.sal > select sal

From emp E_3

where E_3.ename = 'SMITH'

E_2.sal > select sal

From emp E_2

where E_2.ename = 'ALLEN';

Natural Join:

- In Natural Join we wont be writing any Join condition.
- If the table contains similar column we get the o/p of inner Join.
- If the table is not having similar column we will get the o/p of cartesian Join.
- Why OR When we use natural join?
  ↳ whenever there is no table structure we use natural Join.
- column that are present — Table structure.

syntax:

ANSI: select. column_name
From Table_Name1 Natural Join Table name2;

E.G: select *
From emp Natural Join Dept;

OR

select *.
From emp Natural Join salgrade;

Q) WAQTD salary From emp table.

→ select sal
From emp;

Q) WAQTD salary in ascending order From emp Table.

→ select sal
From emp

~~select sal from emp~~

Order by
↓
ASC / DESC

② select sal
① From emp
③ order by sal ASC.

o/p
501
1000
2000
2500
3000
3500

Order by:

- It is used to sort the records in ascending or Descending order.
- Order by clause must be written as last clause in the statement.
- order by clause execute after the select clause.
- By default order by clause sort the records in Ascending order.

- We can pass column_name or Expression as an argument in order by clause.

#

- We can pass alias name in order by clause.

Q1) WAQTD Ename in ASC ORDER.

```
Select Ename
From emp
order by Ename ASC;
```

Q2) WAQTD sal in DESC order

```
Select sal
From emp
order by sal DESC;
```

Q3) WAQTD sal in Ascending order

```
Select sal
From emp
Order by sal ASC;
```

Q4) WAQTD Ename in DESC order

```
Select Ename
From emp
order by Ename DESC;
```

Q5) WAQTD Annual sal in DESC order

```
Select sal * 12 Annual sal
From emp
order by Annual sal Desc;
```

Q6) WAQTD Deptno AND sal in ASC.

```
Select Deptno, sal
From emp
order by Deptno Asc AND sal ASC;
```

Deptno, sal ASC
↳ sal asc order

sal, Deptno ASC
↳ sal asc order.

(sal ascending order ~~if~~ (Draw back))

#

Concatination:

- This operator is used to join the Given two String.

Q1) WAQTD The output in the Following Format

a. 'MR. ABC Your salary is RS. XYZ'

⇒ select 'MR.' || ENAME || 'Your Salary is Rs.' || sal

From emp;

b. 'MR. SMITH Your salary is RS. XYZ'

⇒ select 'MR.' || Ename || 'Your salary is Rs' || sal

From emp

where Ename = 'SMITH';

Mr. King, you Designated as President

→ select 'Mr.' || Ename || 'You Designated as' || Job
From emp
where Ename = 'KING' AND Job = 'PRESIDENT';

d. Mr. X You have been promoted as Manager.

→ select 'Mr.' || Ename || 'You have been promoted as' || Job
From emp
where Job = 'MANAGER';

Single Row Function:

- Dual:
- Dual is Dummy Table.
- To print the result of any mathematical operation Done.

1. Upper():
- This function is used to convert the given string into upper case.

Syntax: UPPER('STRING')

Ex: ① select upper('vivek')
From Dual;
o/p: VIVEK

② select upper('vivek@123')
From Dual;
o/p VIVEK@123

2. Lower():
- This Function is used to convert the given string into lower case.

Syntax: Lower('string')

Ex: ① select lower('VIVEK')
From Dual
o/p: vivek

② select lower('ViVeK')
From Dual
o/p: vivek

3. INITCAP():
- This Function is used to convert the initial character of the given string into upper case.

Syntax: INITCAP('STRING')

Ex: ① select INITCAP('wakar ali')
From Dual;

o/p: Wakar Ali

② select INITCAP('Vivek Singh')
From Dual

o/p: Vivek Singh

4. length():

- This Function is used to count the number of character that are present in the string.

Syntax: length('STRING')

Ex: ① select length('Wakar Ali')
From Dual;

o/p: 9

Q. WAQTD no. of character present in Enames For all the Emp's.

→ select length(Enames), Ename
From emp;

Q. WAQTD Enames who are having only 5 characters in their name using SRF

→ select ~~length(Enames)~~, Enames
From Emp
where length(Enames) = 5;

without SRF:

select Enames
From emp
where EnaMe like '_____';

Q. WAQTD Ename and sal of emp who are getting 3 digit sal Using SRF.

→ select Enames, sal
From emp
where length(sal) = 3;

or

select Enames, sal
From emp
where sal like '___';

Q. WAQTD Enames and comm of emps who are getting 3 digit comm.

→ select Enames, comm
From emp
where length(comm) = 3;

or

select Enames, comm
From emp
where comm like '___';

6. substring:

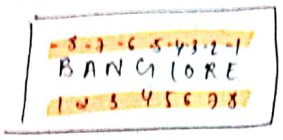
- This Function is used to extract the part of the string from the given original string.

syntax: substr('original_string', position, [length])

Ex: ① substr('BANGLORE', 1, 1) ⟶ B

② substr('BANGLORE', 1, 4) ⟶ BANG

③ substr('BANGLORE', 1) ⟶ BANGLORE



④ substr('BANGLORE', -1, 1) ⟶ E

⑤ substr('BANGLORE', -2, 2) ⟶ RE

(Always take left to right)

⑥ substr('BANGLORE', -3, 2) ⟶ OR

Q. WAQTD the First character of all the emp From emp table

⟶ select substr(Ename, 1, 1)
From emp;

Q2) WAQTD Details of the emp if their name starts A using SRF.

⟶ select *
From emp
where Ename like 'A%';

OR

select *
From emp
where substr(Ename, 1, 1) = 'A';

Q3) WAQTD The First Three characters of all the emps

⟶ select substr(Ename, 1, 3)
From emp;

~~select~~

Q4) WAQTD Details of emp's if there name start with A or S using SRF.

⟶ select *
From emp
where substr(Ename, 1, 1) = 'A' OR
substr(Ename, 1, 1) = 'S';

or

substr(Ename, 1, 1) In ('A', 'S');

Q5) WAQTD Ename whose names start with vowels (a,e,i,o,u)

→ select Ename
From emp
where substrin(Ename,1,1) IN('a','e','i','o','u');

Q8) WAQTD Ename and Job of emps if the Job start with "MAN" string OR end with string MAN.

→ select Ename, Job
From emp
where substr(Job,1,3) = 'MAN' OR
substr(Job,-3,3) = 'MAN';

7. Reverse():

- This Function is used to reverse the given string

Syntax: Reverse('string')

Ex: ① select Reverse('hlabar')
From dual;

② select Reverse('madam')
From dual;

Q. WAQTD Ename in lower case and Job in reverse Format if the emp name having 6 characters.

→ select Ename
From emp
where lower(Ename) AND Reverse(Job) AND
substr(Ename,1,6);

→ select lower(Ename), Reverse(Job)
From emp
where length(Ename) = 6;

8) MOD():

- This Function is used to obtain modulus of given Number (Reminder)

Syntax:

MOD(m,n)

Ex: mod(17,2) → o/p: 1
mod(121,2) → o/p: 1
mod(8,2) → o/p: 0

Q. WAQTD Details of emp whose empNo is an odd No

→ select *
From emp
where MOD(empNO, 2) = 1 ;
(empNO, 2)

Q. WAQTD Details of even Records/empNo.

→ select *
From emp
where MOD(empNO, 2) = 0 ;

9) To_char() :

- This function is used to convert the given Date to string Format.

Syntax :

To_char(Date, 'Format_models')

Format_model's :

1. YEAR	9. DD
2. YYYY	10. D → 7 day
3. YY	11. HH24
4. MONTH	12. HH12
5. MON	13. MI
6. MM	14. SS
7. DAY	
8. DY	

Today's date : 11/04/2022

1. YEAR : To_char(sysDate, 'YEAR') → o/p : Twenty twenty two
2. YYYY : To_char(sysdate, 'YYYY') → o/p : 2022
3. YY : To_char(sysdate, 'YY') → o/p : 22
4. MONTH : " " → " : APRIL
5. MON : " " : APR
6. MM : " " " : 04
7. DAY : " " " : MONDAY
8. DY : " " " : MON
9. DD : " " " : 11
10. D : " " " : 2

Q. WAQTD Enames of the emp who were hired in the month FEB using SRF

→ select Ename
From emp
where To_char(hiredate, 'MON') = 'FEB' ;

Q. WAQTD Enames of the Emp who were hired in the month oct, NOV, Dec

→ select Ename
From Emp
where To_char(hiredate, 'MON') IN ('OCT', 'NOV', 'DEC') ;

WAQTD details of the emp hired on wednesday or MONDAY.

→ select *
From emp
where To_Char(hiredate, 'DAY') IN ('MONDAY', 'WEDNESDAY');

Q WAQTD details of emp hired in the year 81, 83, 87

→ select *
From emp
where To_Char(hiredate, 'YY') IN (81, 83, 87);

Q WAQTD details of emp. if Hired on 17 & 19 & ON 22nd.

→ Select *.
From emp
where To_Char(hiredate, 'DD') IN (17, 19, 22);

SYSDATE:

- This is used to obtain the present date.

→ Sysdate / current_date

- This command is used to obtain the current date from database.

To_Date():

- This Function is used to convert the Date string to Date Format.

syntax:

To_Date ('Date_STR')

Ex: select To_char(To_Date('26-APR-1998'), 'DAY')
From Dual
o/p: sunday.

10) NVL(): (Non Value logic)

syntax: NVL(ARG1, ARG2)

- It can accept 2 arguments
  - ↳ In arg1 we must write column name or expression that can be null
  - ↳ In arg2 we must write a value that can be substituted in place of Null.

If Arg1 is NOT Null, NVL returns same value present in Arg1.

ename	sal	comm
sheela	2000	500
leela	2500	—
meela	3000	400
leela	3500	—
sheela	4000	0

Q. WAQTD total sal from emp table

→ [sal+com]

~~select sum~~

select sal+com as 'Total sal'

From emp;

o/p: 2500

—

3200

—

4000

② select sal+NVL(comm, 0)

① From emp;

o/p: 2500
2500
3000
3500
4000

① sal+NVL(500, 0)
2000 + 500 = 2500

② sal+NVL(null, 0)
2500 + 0 = 2500

③ sal+NVL(400, 0)
3000 + 400 = 3200

INSTR():

- This Function is used to obtain Index value of the substring which is present in the original string

Syntax:

INSTR('ORIGINAL_STRING', 'SUB_STR', position, [nth occurance])

* nth occurance - No of Times it is present.

Example: INSTR('BANANA', 'A', 1, 1) → o/p - 2

INSTR('BANANA', 'A', 1, 2) → 4

INSTR('BANANA', 'A', 1, 3) → 6

INSTR('BANANA', 'A', 1, 4) → 0

↳ Index out of bounds

INSTR('BANANA', 'AN', 1, 1) → 2

Q) WAQTD Details of emp if their name having character 'A' using INSTR

→ select *
From emp
where instr(ename, 'A', 1, 1) > 0;

with INSTR

→ select *
or From emp
where ename like '%A%';

w/o INSTR

WAQTD Enames of Emp in lower case if Emps are having character 'A' present atleast twice in the rows using SRF

→ select lower(Ename)
From emp
where Instr(Ename, A, 1, 2) >0 ;
or
Instr(Ename, A, 1, 2) >= 1 ;

PSEUDOCOLUMNS

- Pseudo columns are the false columns that are present in each and every Table and must be called explicitely.
- Pseudo columns cannot be seen without calling Them.
- Types of Pseudo column:
  1. ROWID
  2. ROWNUM

ROWID:

- ROWID is an 18 Digit Address in which the records is present or the Records is stored in the memory.

```
select ROWID, Emp.*
From Emp
```

NOTE:

→ ROWID is one of the way to access OR delete The Record.
→ ROWID is unique.
→ ROWID is present for each and every Records.
→ ROWID is generated at the time of Insertion of Records.
→ ROWID cannot be inserted, deleted or updated.
→ Empty table will not be having ROWID
→ ROWID is static in nature (constant)
→ ROWID can be used to identify a record uniquely from the table there is no key attribute as primary key.

Rownum:

- Rownum act as serial number to the result Table.

```
select Rownum, Emp.*
From Emp;
```

NOTE:

→ Rownum is used as record number that is assigned to the Result Table.

→ Rownum is Dynamic in nature (keeps on changing)

→ Rownum is generated at the time of execution.

→ Rownum always start with 1

→ Rownum cannot be duplicated.

→ Rownum gets incremented after it is assigned.

Q. WAQTD The First three records from emp Table.

→
```
select *
From Emp
where RowNum <= 3 ;
```

```
Select *
From emp
where Rownum <= 3;
    or
RowNum IN (1,2,3);
    or
Rownum < 4;
```

Ename	Deptno
Shastk	40
Tarik	30
manik	10
Lalik	40

Rownum < 4

1	Shastk	40	1 < 4
2	Tarik	30	2 < 4
3	manik	10	3 < 4
4	Lalik	40	4 < 4 X

```
where Rownum, Emp.*
From emp
where Rownum < 4 ;
```

Q WAQTD the First row of emp table.

```
Select *Rownum, emp.*
From emp
where ROWNUM IN (1);
```

Q WAQTD the third Records From emp table

```
select emp.*
From emp
where Rownum = 3;
```

↳ RowNum > 2 AND RowNum <= 3

Rownum is dynamic that's why it is wrong

⇒ To make Rownum as STATIC:

- Take a table and assign Rownum to a given Table.
- Change the Rownum to any other name by using Alias (Serno)
- use this as a subquery in From clause of outer query.

• In the outer query use the alias name in the condition.

→ Rownum as static

```
select Rownum SINO, Emp.*
From emp;
```

SINO	Ename	DeptNo
1	sheela	20
2	leela	30
3	mala	10
4	laila	20
5	shakila	30

→
```
select *
From (select RownumSINO, Emp.*
      From Emp)
where SINO = 3;
```

Q WAQTD the seventh Records From emp Table

→
```
select *
From (select Rownum SRINO, Emp.*
      From Emp)
where SINO = 7;
```

Q6) WAQTD First Four Records From emp table

```
select Rownum, emp.*
From emp
where Rownum <= 4;
```

Q4) WAQTD of 1st, 3rd, 5th & 8th Record

→
```
select *
From (select Rownum SINO, Emp.*
      From emp)
where SINO IN(1,5,7,3);
```

Q5) WAQTD Enames of 3th, 7th AND 6th Record

→
```
select Ename
From (select Rownum SINO, Emp.*
      From emp)
where SINO IN(6,7,3);
```

Q7) WAQTD last 3 Records From Emp Table.

→
```
select *
From (select Rownum, Emp.*
      From emp
      order by Rownum DESC)
where Rownum <= 4;
```

Rownum	Ename	Deptno
5	shakila	30
4	laila	20
3	mala	10
2	leela	30
1	sheela	20

Q. WAQTD the last Five Records From Emp Table

```
→ select *
  From (select Rownum, emp.*
        From emp
        order by Rownum DESC)
  where Rownum < 6 ;
```

Q. WAQTD The max salary of employee table

```
select max(sal)
From emp;
```

Q. WAQTD the 2nd max sal From emp table.

```
→ select max(sal)
  From emp
  where sal < select max(sal)
              From emp
```

Q. WAQTD the 9th max sal From emp table

Syntax: For nth max:

```
select sal
From (select Rownum slno, sal
      From (select DISTINCT sal
            From emp
            order by sal DESC))
where slno = n ;
```

9th max sal

```
→ select sal
  From (select Rownum slno, sal
        From (select Distinct sal
              From emp
              order by sal Desc))
  Where slno = 9 ;
```

Q. WAQTD 13th max sal From Emp Table

```
→ select sal
  From (select Rownum slno, sal)
        From (select Distinct sal
              From emp
              order by sal Desc))
  where slno = 13 ;
```

Q. WAQTD 9th min sal

Syntax: For nth MIN

```
→ select sal
  From (select Rownum slno, sal
        From (select distinct sal
              From emp
              order by sal ASC))
  where slno = n ;
```

8th min sal

```
Select sal
From (select Rownum s.no, sal
      From (select Distinct sal
            From emp
            order by sal asc))
where s.no = 8 ;
```

Data Types:

- Data Types are used to determine what type or kind of data will be stored in a particular memory allocation.

Datatypes in SQL:

1. Char
2. Varchar / Varchar2
3. Numbers
4. Date
5. Large object
   a. character large object (CLOB)
   b. Binary large object (BLOB)

∴ SQL is NOT case-sensitive

1. Char:

Char datatype can accept characters such as 'A-Z', 'a-z', '0-9' OR special characters (#, $, -)

Syntax:

char(size)

Size: • It is used to determine the no. of characters that we can store.

- Whenever we mention char Datatype we have to mention size for it
- The max size that we can store is 2000.

***

- It is type of fixed memory allocation.

Drawback of char

'WASTAGE OF MEMORY'

2. Varchar/Varchar2:

Varchar datatype can accept characters such as

- A-Z
- a-z
- 0-9 or
- special character (#, *, $, . .)

Syntax: Varchar(size)

Size:

It is used to determine the number of character that we can store.

- whenever we mention Varchar datatype we have to mention size for it.
- The max size that we can store is 4000
- It is a type of "Variable length memory Allocation"

Varchar2:

- Varchar2 is the updated version of Varchar.
- i.e The size is updated From ~~2000 to~~ 4000

syntax:

Varchar2(size)

- The max size we can store is 4000.

Number:

- This datatype is used to store numerical Values
- It can accept two argument
  a) Precision
  b) scale

syntax:

Number (Precision, [scale])

Precision:

- It is used to determine the Digits we are going to store in numerical place.

scale

- It is used to Determine the No. of Digit we are going to store in Decimal place within the precision

* The max Precision we can store is ~~38~~ 38.
* The max scale we can stor is 127

case-I (P > S)

Number (3,1)

9 9 . 9

Ex Number (5*)
↳ bydefault scale is 0
9 9 9 9 9

case-II (P = S)

Number (2,2)

0 . 9 9

case-III (P < S) (Here we give priority to scale)

Number (2,4)

0 . 0 0 9 9

4. Date

Syntax - **DATE**

The two oracle specified Date Formats are

a) 'DD-MON-YY'

b) 'DD-MON-YYYY'

- Date should always be enclosed within single quotes

5. Large object :

a) CLOB (Character large object)

- This is used to store the characters upto 4GB of size
- Syntax - **CLOB**

---

b) BLOB (Binary large object)

- This is used to store binary number of images, Videos, Files, etc
- upto 4 GB of size

Syntax - **BLOB**

SLOB	Number(3)	Varchar2(20)	Number(8,2)	Number(10)	Date
Sno	Sno	Ename	Sal	Ph No	Doj
1	1	Rafi	40000.00	9876543210	14-Feb-21
2	2	Anitha	45000.00	9876543210	10-NOV-21
3	3	Vijay	30000.00	9876543210	14-DEC-21
4	4	Mahesh	60000.00	0123456789	21-JAN-22
5	5	Ajju	10000.00	9999999999	21-Feb-22

D.T ⟶ to Validate date.

CONSTRAINTS:

- Constraints are the condition that are assigned to a particular column to validate the data.
- Types of constraints

1. Unique
2. Not NULL
3. Check
4. Primary key
5. Foreign key

1. Unique:

- Unique is a constraint which is assigned to a particular column which cannot accepted Repeated or Duplicate Value.

2. Not Null:

- Not Null is a constraints which is assigned to a particular column which cannot be Null or which are Mandatory.

3. Check:

- Check is a constraints which is assigned to a particular column for Extra Validation.
- Check constraints is assigned with condition. if the condition is true the value get accepted else rejected.

Ex: 1. check (length(phn no) = 10)
2. check (sal > 0)

NOTE:
- NULL is a keyword which represents empty cell.
- Any operation preformed with NULL, the result will be Null.

4. Primary key:

- Primary key is a constraint which is assigned to a column to identify a record uniquely from the table.

Characteristics of Primary key:

1. We can have only one primary key
2. It cannot repeated or duplicate value
3. It cannot accept Null
4. It is a combination of unique and NOT Null
5. Primary key is not mandatory but recomended to have one in Table.

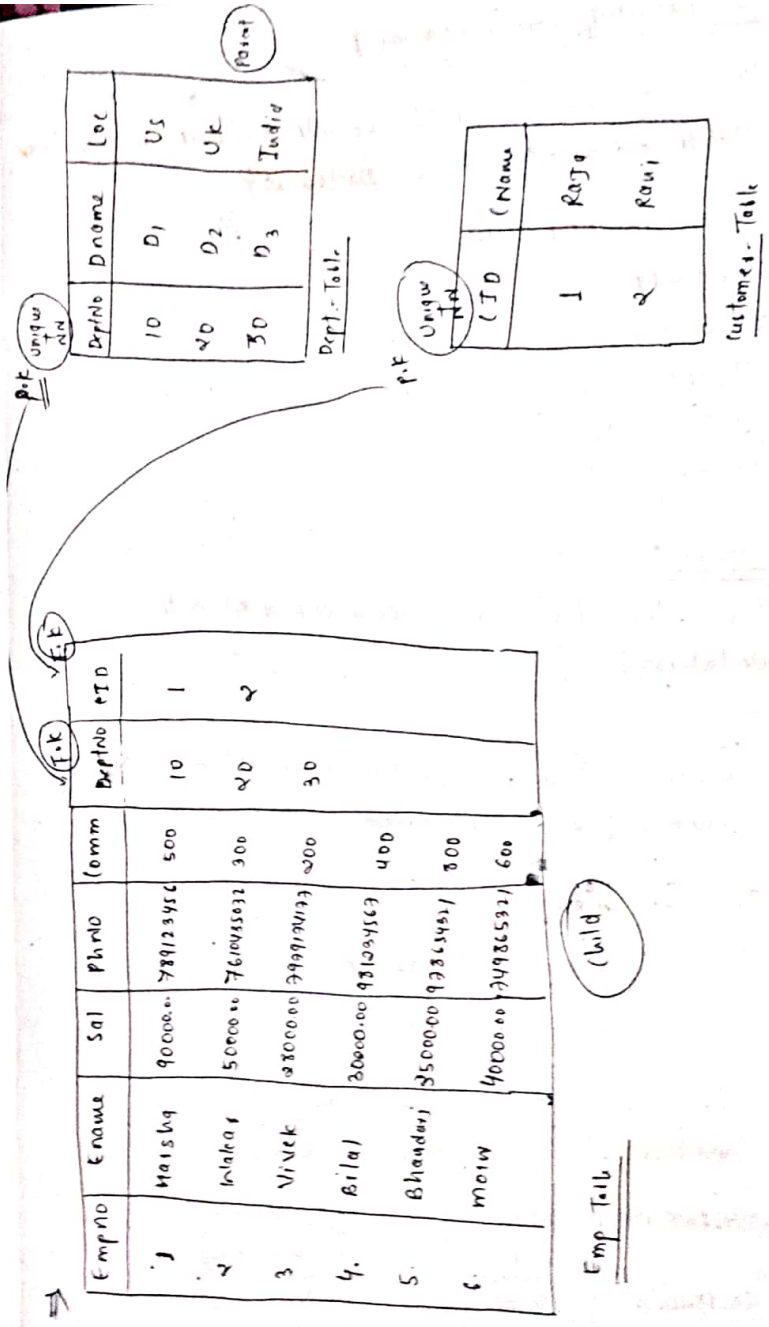
5) Foreign key:

- It is a constraint which is used to establish the connection between two tables

Characteristics of Foreign key:

1. We can have 'n' nos of foreign key in Table.
2. It can accept Repeated or Duplicated Value
3. It can accept null.
4. It is not a combination of unique and Not null
5. It is present in child Table but actually belongs to parent table
6. It is also referred as "Referential integrity Constraints".

- Only primary key can travel to another Table. When it travel it becomes Foreign key.



DDL (Data Definition Language)

- This statement is used to **create, rename, alter, or Delete an object From Database**.

There are **5** statements:

1) **Create**
2) Rename
3) Alter — modify
4) Truncate
5) Drop

1. Create:

This statement is used to **create an object in Database**.

Q. Create a Table with the name customes having columns like "CID, CName, CNum, loc".

→ **Blueprint**

Table name = **customes**

No. of column = **4**

	CID	CName	CNum	loc
DataType	Number(4)	Varchar(50)	Number(10)	Varchar(70)
NULL/NOT NULL	NOT NULL	NOT NULL	NOT NULL	NOT NULL
Constraints	Primary key CID_PK		check(length (CNum)=10) CNum_CK	

```
Create Table customes
(
CID     number(4)   NOT NULL,
CName   Varchar(50) NOT NULL,
CNum    number(10)  not null,
loc     varchar(70) not null,
constraints  CID_PK  Primary key(CID),
constraints  CNum_CK check(length(CNum)=10)
);
```

Q. Create a Table name Product having 4 column PID, Pname, Price, Discount.

→ Blueprint: Table name = Product

No. of col = 4

	PID	Pname	Price	Discount
data Type	Number(4)	Varchar(50)	Number(8)	Number(4)
Null/NOT NULL	NOT NULL	NOT NULL	NOT NULL	NOT NULL
Constraints	Primary key PID_PK,		check(Price>0) Price_CK,	

```
→ Create Table Product
(
PID      Number(4)   NOT NULL,
Pname    Varchar(60) NOT NULL,
Price    Number(8)   NOT NULL,
Discount Number(4)   NOT NULL,
constraints  PID_PK  Primary key(PID),
constraints  Price_CK check(Price>0) );
```

* If you want to save the table in Database we command `COMMIT;`
  → store permanent in Database
* Always write constraints reference name should be Unique.

2. Rename:

- This statement is used to rename the current Table name to New Name.

Syntax: `Rename current_table_name to new_table_name;`

Ex: Rename customer to cust;

3. Alter:

- This statement is used to modify the objects in Database

1. To add a column:
   Ex: Alter Table customer
   Add email Varchar(45) NOT NULL;

2. To Drop a column:
   Ex: Alter Table customer
   Drop column Loc;

Command for Describe
`Desc customer;`

3. To ~~modify~~ change the Data Type in column

Ex: Alter table customer
modify Ename Number(50);

Ex: Alter table customer
modify Lname Varchar(60);

4. To change NOT NULL constraints:

Ex: Alter table customer
modify Cmail Varchar2(45) NULL;

Ex. Alter table customer
modify Cnum Number(10) NULL;

5. To Rename the column:

Ex. Alter Table customer
Rename column Cnum to num;

Ex. Alter Table customer
Rename column Cname to name;

6. To Modify Constraints

1) Alter Table customer
Add PID Number(3) NOT NULL

Add a Foreign key:
Alter Table customer
Add constraint PID_FK Foreign key(PID)
References Product(PID)

Truncate:

- It is used to Delete all the Records From table permanently.

syntax: Truncate Table Table_name;

Ex: Truncate Table customer; (Don't perform in the system)

- If we perform truncate all the records present in that particular table will be Deleted permanently from the database and we cannot retrieve the data's again we will be left out with only the column names and the corresponding data types that they assigned.

5. DROP:

- This statement is used to delete the object i.e Table from the Database along with table structure.

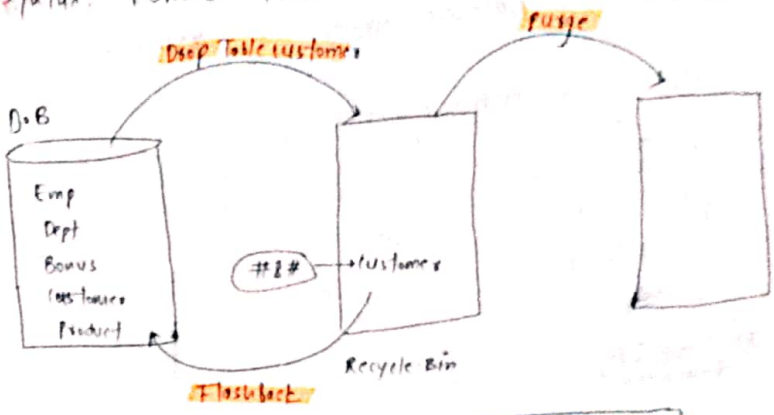
Syntax: Drop Table Table_name;

To Recover the Table: (only in oracle)

syntax: Flashback Table table_name
To Before Drop
[Rename To new_name];

To Drop the Table From Recycle bin:

Syntax: PURGE Table table_name;



Show Recyclebin;
(command show all the records in recycle Bin.

DML (Data manipulation language):

- This statement is used to insert, update or Delete the records from the table
- There are three statements
  1) Insert
  2) update
  3) Delete

1) Insert:

- This statement is used to insert the record into the table.

syntax: 1. Insert into table_name Values ($V_1, V_2, \ldots V_n$);

2. INSERT INTO table_name (col1, col2, ... coln)
   VALUES ($V_1, V_2, \ldots V_n$);
   Or
3. INSERT INTO table_name (col1, col2, ... coln)
   VALUES (&col1, &col2, &col3, ... &coln);

- whenever we know the sequence of column we used 1. syntax.
- whenever we know only column names but we dont know sequence of columns we use 2. syntax.
- whenever we want to insert n no. of records we use 3rd. syntax.

1. syntax:
Ex. Insert Into Product Values (1, 'covaxin', 1000, 20);

(commit; (for save in the Database)

2. syntax:

Insert Into Product (PID, Pname, Discount, price)
Values (10, 'SHOES', 10, 3000);

3. syntax

Insert Into Product (PID, Pname, price, Discount)
Values (&PID, &Pname, &price, &Discount);

Enter value for PID : 11
Enter value for Pname : 'BOTTLE'
Enter value for price : 300
Enter value for Discount: 10

* If you want to ~~Insert~~ value in multiple row then use 3 syntax.

prefer / slash (Forward)

---

2. Update:

- This statement is used to update the Records in the Table.

syntax: Update table_name
set col1 = V_1, col2 = V_2 .... coln = V_n
[where <filter condition>];

Ex: update Product
set Pname = 'Glass', Discount = 20
where Pname = shoes;
OR
PID = 9

3. Delete:

- This statement is used to delete a particular record from the table.

Syntax: Delete
From table_name
[where <filtercondition>];

Q. Difference b/w Truncate, Drop, Delete

4. Data Control Language:

- This statement is used to give the permission or take back permission from another user.

There are 2 statements:-

1.) Grant:

- This statement is used to give the permission to another user.

Syntax: Grant sql-statement ON table_name
To user_name;

2.) Revoke:

This statement is used to take back the permission from another user.

Syntax: Revoke sql-statement ON table_name
From user_name;

SCOTT	
Emp	Table
Dept	Table
Bonus	Table
SalGrade	Table
Product	Table

HR	
Regions	Table
Countries	Table
Location	Table
operation	Table
Department	Table
Jobs	Table
Employees	Table
Emp-Details-View	View

Select *
From HR.employee;

permission → Grant select on employees
To Scott;

Revoke select on employees
From Scott;

⇒ Transaction control language:

There are three statements:

1. Commit
2. Savepoint
3. Rollback

1. Commit:

Syntax: Commit;

This statement is used to save transaction on database.

2. Savepoint:

Syntax: Savepoint savepoint_name;

This statement is used to mark the position on Database.

3. Rollback:

Syntax: Rollback;

Syntax: Rollback to savepoint_name;

This statement is used to go back or undo to the previous savepoint.

Ex: insert into Product values (11, 'BANANA', 500, 50);

1 row created.
Savepoint R_1;
insert into Product values (13, 'NINJA', 1000, 50);
Savepoint R_2;
Insert into Product values (14, 'Bitwise', 180, 20);
Savepoint R_3;
Rollback to R_1;

Attributes:

1. Key attributes:

- An attribute which is used to identify the record uniquely from the ~~database~~ Table is called key attribute.

2. Non-key attributes:

- All the attribute except key attribute are preferred as non-key attributes.

3. Prime-key attributes:

- Among the key attributes an attribute is choosen to be the main attribute to Identify the record uniquly from the Table.

4. Non prime key attribute:

All the key attribute except prime key attribute is refeared non prime key attribute.

5. Composition key attribute:

- It is a composition of two or more non key attributes which is used to identify the Record unique from the table.

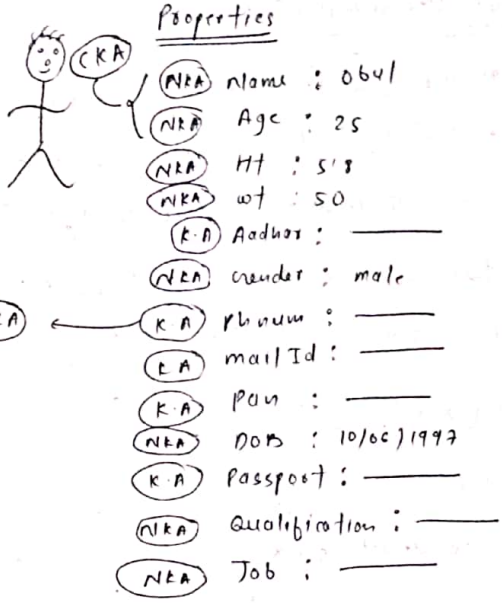
6. Super key attribute:

It is the set of all the key attributes.

7. Foreign key attributes:

It behaves as an attribute of another entity to represent the relation.

Ex-



Functional Dependency:

let us consider the relation 'R' with two attributes 'x' and 'y' respectively in which attribute 'x' determine attribute 'y'.

- or in other words 'y' dependent on 'x' Then exist functional dependency.

- $R \longrightarrow \{x, y\}$

  $x \longrightarrow y$

  y is dependent on x.

Types of Functional Dependency:

1. Total Functional Dependency
2. Partial Functional Dependency
3. Transitive Functional Dependency.

1. Total Functional Dependency:

- If all the attributes in a relation are determined by a single attribute which is a key attribute then there exist total Functional Dependency.

- In total Functional Dependency there are no anomaly and Redundancy.

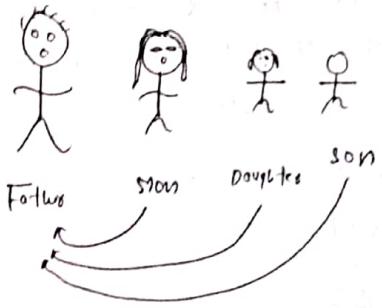
Anomaly: These are the side effect which are caused during the DML operations.

Redundancy: These are the repeated or Duplicated.

Ex: $R \longrightarrow \{A, B, C, D\}$

$A \longrightarrow \{B, C, D\}$

$A \longrightarrow B$
$A \longrightarrow C$
$A \longrightarrow D$



2. Partial Functional Dependency:

- For a partial Functional Dependency to exist there must be a composite key attribute.
- One of the attribute in composite key relation determines another attribute separated and this is known as partial Functional dependency.
- In partial Functional dependency we have Redundancy and Anomaly.

Ex: Let us consider a relation 'R' with 4 attributes A, B, C, D IN which A & B are composite key attribute

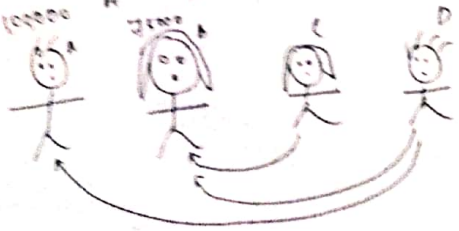
Ex: $R \longrightarrow \{A, B, C, D\}$

$(A, B) \longrightarrow (C, D)$

$B \longrightarrow C$

$B \longrightarrow D$

$A \longrightarrow D$



3. Transitive Functional Dependency:

- If an attribute is determined by a non-key attribute which in turn is determined by a key attribute, then there exists transitive functional dependency.
- In transitive functional Dependency we have redundancy and anomaly.

Ex: Let us consider a relation with 4 attribute A, B, C, D in which A is a key attribute.

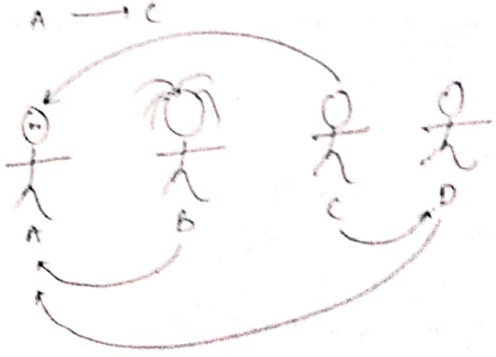
$R \longrightarrow (A, B, C, D)$

$(F.D)$ $A \longrightarrow B$

$A \longrightarrow D$

$B \longrightarrow C$

$A \longrightarrow C$



Normalization:

- It is a process of producing a longer table into smaller table in order to remove the redundancy and anomaly by identifying their functional dependency.

Or

- It is a process of decomposing a large Table into smaller table to remove redundancy and anomaly.

Normal Form:

~~It is a process of reducing the table of its~~

- A table w/o redundancy and anomaly is said to be a normal Form.

Level of normal Form:

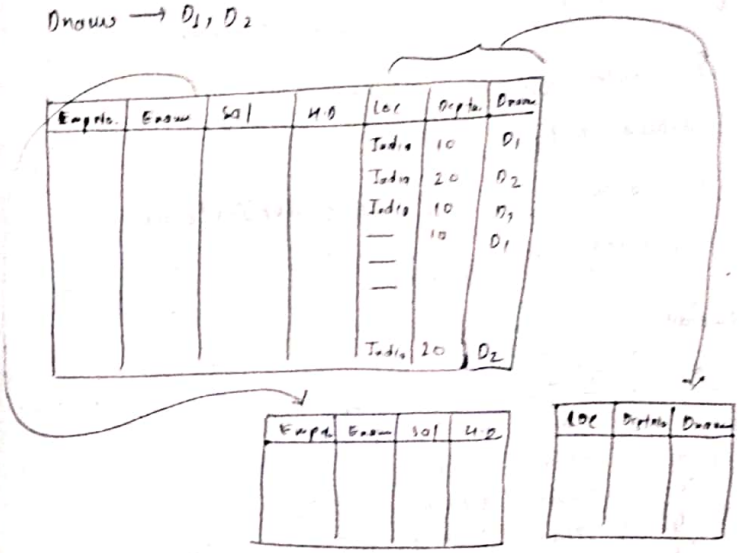
1. First normal Form (1NF)
2. Second normal Form (2NF)
3. Third normal Form (3NF)
4. Boyce code normal Form (BCNF)

R → { Empno, Enames, Sal, H.D, Loc, Deptno, Dnames }

Loc → India

DeptNo → 10, 20

Dnames → D_1, D_2



NOTE: A Table is said to be Normalized if we reduce to ~~3rd~~ 3rd Normal Form.

1NF:

- A table is said to be in 1st Normal Form.
- If it satisfies the following condition.
  - ↳ A table should not consist of multi-valued data.
  - ↳ A table should not have duplicate or repeated values.

student

Sid	Sname	Skills
1	shaila	SQL
2	leela	Java, SQL
3	maala	PY, SQL
4	laila	Java
5	shakila	Java

1NF

Sid	Sname	Skills 1 (SQL)	Skills 2 (Java)	Skills 3 (PY)
1	shaila	SQL	Java	
2	leela	SQL	Java	
3	maala	SQL		PY
4	laila		Java	
~~5~~	~~shakila~~		~~Java~~	

2NF:

- A Table is said to be in 2nd Normal Form.
- If it satisfies the following conditions
  - → The table should be in 1st Normal Form
  - → The table should not have partial Functional dependency.

NOTE

If the table consist of partial Functional dependency then the attribute responsible are removed from the table.

Ex:

R ⟶ {empno, Enamw, sal, H.D, Deptno, Dnamw, Loc}

empno ⟶ Enamw, sal, H.D

Deptno ⟶ Dnamw, Loc

Empno, Deptno ⟶ {Enamw, sal, H.D, Dnamw, Loc}

CKA

PFD

R_1 ⟶ {Empno, Enamw, sal, H.D} (KA)

R_2 ⟶ {Deptno, Dnamw, Loc} (KA, PK)

3NF :

- A table is said to be 3rd Normal Form

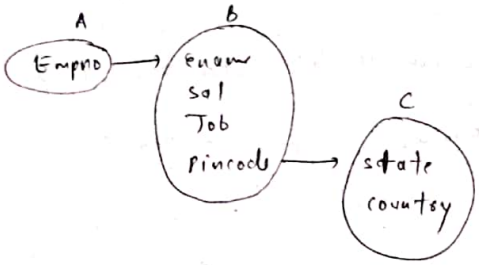
If it satisfies the following condition

→ A table should be in 2nd Normal Form.

→ A table should not have transitive Functional dependency

Note If the table consist of transitive Functional dependency then the attribute responsible are removed from the table.

R ⟶ {Empno, Ename, sal, Job, Pincode, state, country}



A ⟶ B

B ⟶ C

R_1 ⟶ {Empno, Ename, sal, Job} (K A)

R_2 ⟶ {Pincode, state, country} (K A)

4. BCNF

- It is the updated version of 3NF
- Also called 3.5 NF