classmate
Date 09/02/21
Page

Data :-

Data is a raw fact which describes the attributes of an Entity.

Entity (or) objects :-

Anything which has its existance

Attributes :- properties of an object.

Data Base :-

Data base is a place (or) medium where we can store the data in systematic and organized manner.

We can perform some basic operations in data base

1) Create / Insert
2) Read / retrive
3) update / modify
4) delete / Drop.

These operations are known as CRUD operations.

DBMS :- [Data base Management System].

DBms is a Software which is used to maintain & manage the database.

DBms provides two important features they are 1) Security → protection 2) Authorization → verification.

To Communicate with DBMS Software we need a language called Query Language.

Types of DBMS.

1) RDBMS. (RDBMS)
2) Network.
3) object oriented.
4) hierarchy.

RDBMS

RDBMS :- [Relational data base management system].

RDBMS is a type of DBMS software in which we can store the data in the form of tables / Relations.

(or)

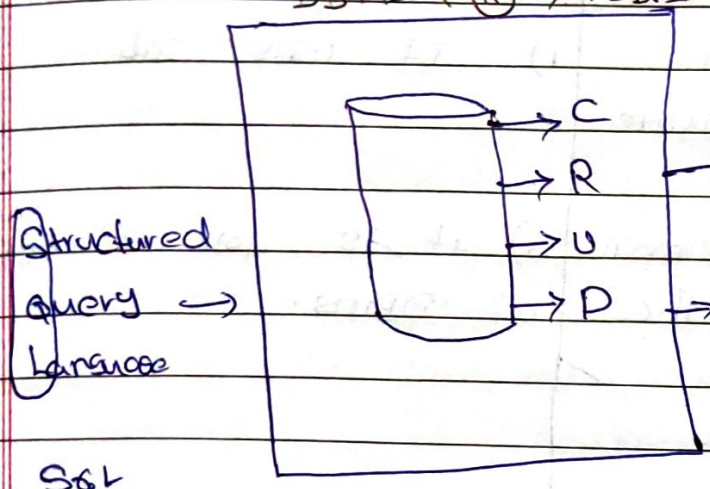
If DBMS follows relational model then it becomes RDBMS.

(or)

If DBMS follows E.F. CODD rules then also it becomes RDBMS.

To Communicate with RDBMS software we need a language called SQL.
[Structured Query Language].

E.F CODD → DBMS + (R) ⇒ RDBMS



Relational Model :-

* Relation model was designed by E.F.CODD.
* In Relational Model (or) RDBMS we can store the data in the form of tables (or) relations.
* In Relational Model we can also store the "Meta data" in the form of data.

Meta data :-

Data about the data is called as meta data.

Ex:- picture

Size = 5 mB.
Time = 09 Am.
Data = 01-01-20.
Resolution = 1630 x 20.

EF CODD rules :-

1) Data Entered into the cell must be Single Valued data (or) Atomic data.

2) We can store the data in multipe tables and we can establish the connection b/w any two tables by using "key attributes".

3) We can validate the data which are enter in to the tables by assigning.

(1) Data-types

(2) Constraints.

4) Data types are mandatory, but constraints are optional.

Students :-1

ID	name	Stream	per
01	Mahesh	ECE	64.1.
04	Sasi	Dog	70.1.
07	Ravi	Team	65.1.
08	Krish	haus	75.1.

DoB	Adhar	Blood group	Age
09	111	o+	20
10	20	o+	20
22	40	o+	20
64	70	o+	20

Data types :-

Data types are the type of data (or) kind of data which are used to store in memory location.

Types of datatypes in sql :-

<1> char
<2> varchar. (or) varchar2.
<3> large object → i) char LOB ii) Binary LOB.
4> Date.
5> Number.

1. char datatype :-

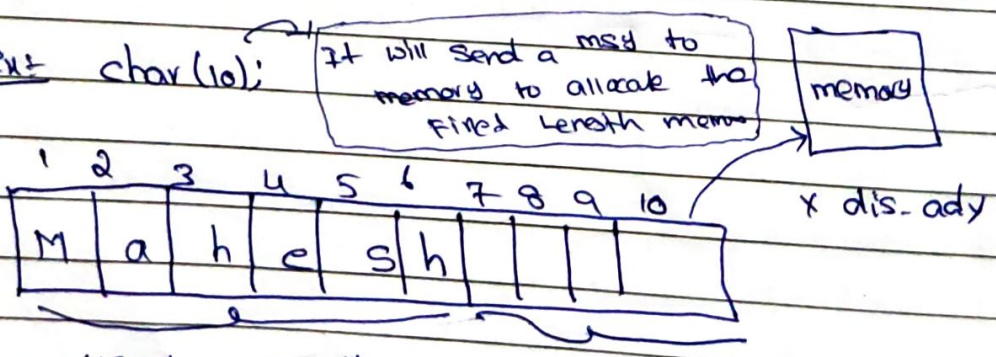
char datatype is used to store the characters

<1> 'A-Z' upper case alphat.
<2> 'a-z' lower
(3) '$-@' special char.
<4> '0-9' numbers.

Syntax :-

char(size); 2000 max.

Ex :- char(10);



x dis-adv

To overcome this disadvantage we will use varchar.

Var char :-

Varchar is used to store the characters.

(1) 'A-Z'.

(2) 'a-z'

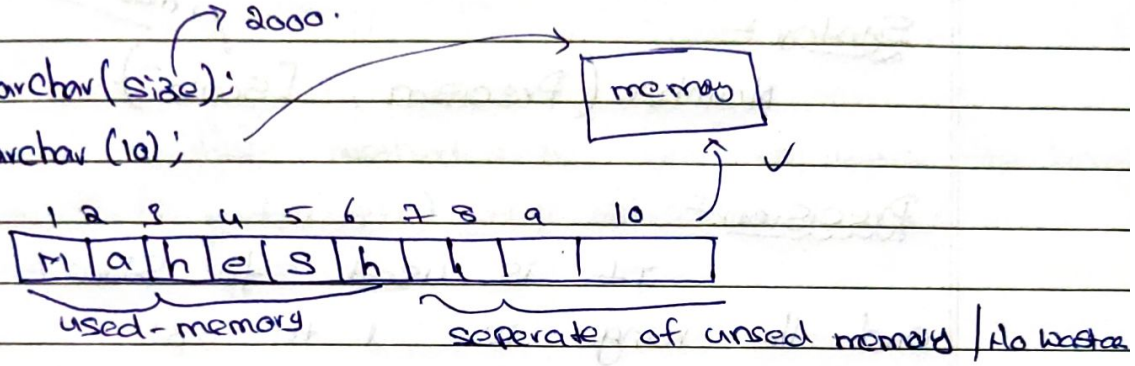
(3) '@-@'

(4) '0-9'.

Syntax :-

Varchar(size);

Ex :- Varchar(10);



Note :-

1) char and varchar can except max size of 2k

2) Varchar2, it is just an updation of varchar, which can except max size 4k.

Date data-type :-

Syntax :- Date.

The standard format which is given by oracle is

'DD-MON-YY' ⇒ '6-FEB-20' ↔ For current cent.

'DD-MON-YYYY' ⇒ '22-APR-1998' ↔ Past Centuary.

Rdms.

(1) oracle → SQL*plus, my SQL

(2) microsoft → SQL server

(3) IBM → DB2

(4) Sybase

language	Software
SQL	MySQL
SQL	J-SQL

classmate
Date
Page

Number :-

This datatype is used to store numerical values.

It can accept two arguments :-

a) Precision.

b) scale

Syntax :-

Number (Precision, [Scale]).

↑ max range 127.

Precision :-

It is used to store integer values and the range is 1 to 38.

Scale :-

It is not mandatory & which is used to store decimal values in the precision.

1) Number (5) ± 9 9 9 9 9 → max.
2) Number (4,0) ± 9 9 9 9
3) Number (5,2) ± 9 9 9 9 9
4) Number (4,4) ± 0.9 9 9 9
5) Number (2,5) ± 0.0 0 0 9 9
6) Number (3,7) ± 0.0 0 0 0 9 9 9
7) Number (38, 127) ± 0.0 0 0 0 ... etc

Large objects (LOB)

(1) char LOB :-

This method is used to store the char up to 4 GB.

Syntax: CLOB;

(2) Binary LOB :-

This method is used to store the binary values (like audio, video etc) up to 4GB.

Syntax : BLOB.

Constraints :-

Constraints are the conditions 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 accept repeated or duplicate values.

2. NOT NULL :-

NOT NULL is a constraint which is assigned to a particular column which cannot be null (or) which are mandatory.

3. check :-

check is a Constraint which is assigned to a particular column for extra validations.

It is assigned with a condition, if the condition is true the value gets accepted, else rejected.

check(length(phno=10)

	NOT NULL		check(sal>0)	NOT NULL
	Unique	NOT NULL	NOT NULL	UNIQUE.
Photo	EID	EName	Sal	Phno	HireDate
BLOB	Number(3)	Varchar(25)	Number(7,2)	number(10)	Date
	1	AJITH	35,000.00	9999999999	01-JAN-20
	2	NIKITHA	35,000.00	9988765422	01-FEB-21
	3	MATHESH	40,000.00	8309906848	01-FEB-21
	5	SonSay	50,000.00	9999991765	01-Jan-20.

classmate
Date 12/08/21
Page

Primary key Constraint :-

It is used to uniquely identify a record from the table.

characteristics of Primary key Constraints :-

1) The table can be having only one Primary key.
2) It should be unique.
3) It should be not null.
4) It is always a combination of unique and not null.
5) It is not mandatory. But design wise Preferable.

Foriegn key Constraints :-

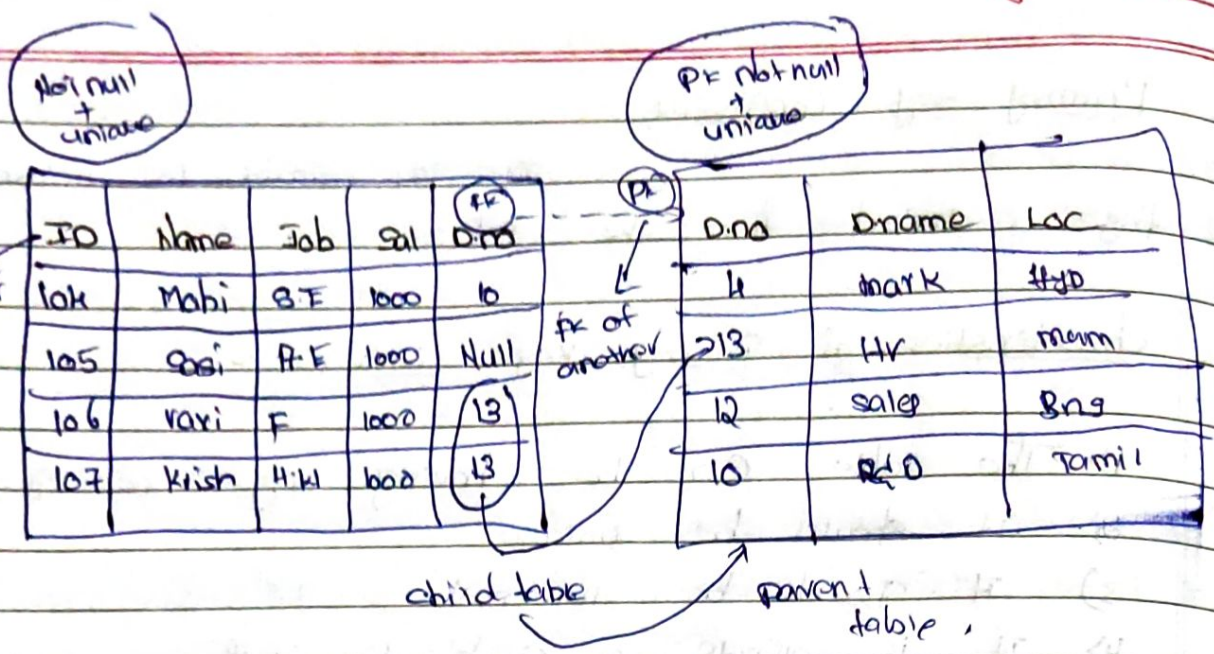
It is used to establish the Connection between any two tables.

characteristics of foriegn key Constraints :-

* The table can be having any no. of foriegn key.
* It can accept duplicate values.
* It is not a combination of unique and not null
* It is also not mandatory.
* Foriegn key always present in child table but actually belongs to parent table.

*** foreign key is also known as refferential integrity Constraint.

* only Primary key can travel to another table, when it travels it becomes foriegn key.



15/02/21.

Statements in SQL

1) Data definition language (DDL)
2) Data manipulation language (DML)
3) Transaction Control language (TCL) (Grant, revoke)
4) Data Control language (DCL) (commit, rollback)
5) Data query language (DQL).

1. Data definition language (DDL):-

This Statement is used to create, rename, alter or delete an object from database.

There are five statements:

1. Create.
2. Rename.
3. Alter
4. Truncate
5. Drop.

Create:-

This Statement is used to create an object in database.

Example → table view

Syntax

```
Create Table tablename
(
 column_name1  DATATYPE  NOT NULL/[NULL],
 column_name2  DATATYPE  NOT NULL/[NULL],
    :
    :
 column_nameN  DATATYPE  NOT NULL/[NULL],
 CONSTRAINT / Constraint_ref_name unique (column_name),
 constraint  constraint_ref_name check (Condition),
 Constraint  Constraint_ref_name primarykey( column_name),
 constraint  Constraint_ref_name foriegnkey ( column_name)
 References  parent_table_name (column_name)
 );
```

Blue print:

Table name : customer [No. of cols : 4]

col-Name / data-types	CID	CNAME	PH NO	LOC
	Number(3)	Varchar(30)	Number(10)	Varchar(40)
NULL/NOT NULL	NOT NULL	NOT NULL	NOT NULL	NULL.
constraints.	primary key [cid-pk]		UNIQUE [ph-u] CHECK Long(ph#) [ph-c]	

Example → bookSide :

```
Create table Customer
(
    CID Number(3) Not NULL,
    CName Varchar(20) Not NULL,
    PHNO Number(10) Not null,
    Loc Varchar(50) Not null,
    Constraint CID-PK primary key (CID),
    Constraint PH-U unique (PHNO),
    Constraint PH-U check (Length (PhNo) =10)
);
```

→ Table Created.

* DES Customer:

Name	NULL	TYPE
CID	NOT NULL	Number(3)
CName	NOT NULL	varchar2(30)
PHNO	Not NULL	Number(10)
Loc	NOT NULL	varchar2(50).

Rename:

This Statement is used to rename the current table name to new name.

Syntax:

Rename current_table_name to new_table_name;

Eg:- Rename customer to cost;
table renamed.

Alter:
This statement is used to modify the structure in database.

Syntax:
1) To add a col
Alter table table-name
Add mail, varchar(50) null;

2) Drop
Alter table cust
Drop column loc;

Truncate :- permanently

Used to delete all the records permanently

Syntax:- Truncate name table table_name;

Drop :- or

This statement is used to delete the object i.e Tables from the database along with table structure.

Syntax:-

Drop table table_name;

To recover table: (only in oracle)

Syntax :- flashback table table_name :
To before drop
[Rename to new_name];

To drop table from recycle bin

Syntax:

Purge table table_name;

Example:-

Drop table Customer;
Table dropped.

Flashback Table Customer
To Before drop
rename to cust.

Data manipulation language (DML) :

This statement is used to insert, update or delete the records from the table.

There are 3 statements.

1. Insert.
2. update.
3. delete.

Insert :

This statement is used to insert the record in to the table.

Syntax : 1) Insert into table_name values(v1, v2 ... vn);

2) Insert into table_name (col1, col2 ... coln)
values (v1, v2 .... vn);

(or)

Insert into table_name (col1, col2 ... coln)
values (&col1 &col2 .... &coln);

Examples :

insert into cust values(5, 'F', 9966275544, '2cgm');
1. row Created.

insert into cust (CID, MOBNO, Cname, mailID)
values (6, 7766889955, 'F', 'HBYN');
1. row created.

Insert into cust (CID, ~~cust_id~~, mobno, Cname, mailID)
values (&CID, &mobno, &cname, &mailID);
1 row created.

commit

Update:-

This Statements is used to update the records in the table.

Syntax:

```
update table_name
set col1 = v1, col2 = v2 ... coln = vn
[where <filter_condition>];
```

Example:

```
update cust
set mobno = 8899787765, mailID = 'ASEDRFG'
where cID = 1;
1 row updated
commit → to save permanently.
```

Delete:-

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

Syntax:

```
Delete
from table_name
[where <filter_condition>];
```

Example:-

```
Delete
from cust
where cID = 6;
1 row deleted
commit.
```

classmate
Date 17/02/21
Page

Data query language (DQL).

This statement is used to retrive the data from database

There are 4 statements.

1. Select
2. Projection
3. Selection
4. Joins.

Select :-

This statement is used to retrive the data from the database and display it

Projection :-

This statement is used to retrive the data from the database by selecting only Column.

All the column will be selected by default.

Selection :-

This statement is used to retrive the data from the database by selecting both column as well as records.

Joins :-

This statement is used to retrive the data from database from multiple tables simultaneously.

Projection :-

Working procedure

1. From clause starts the execution.
2. For from class we can pass table name as an Argument.

3) The Job of from clause is to go to the database and search for the table and put the table under execution.

4) Select clause executes after the execution of from clause.

5) For select clause we can pass ~~ASTERISK~~(*), column name and expression as an argument.

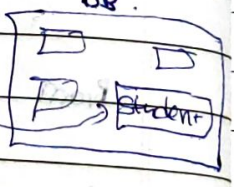
6) The Job of select clause is go to the table which is underexecution and display.

7) select clause is responsible for the table.

3)

1) WAQTD all the names of the student from std table

```
Select sname
from Student;
```



Student

SID	Sname	Branch	Per
1	Mahi	ECE	63
2	chinnu	BSc	75
3.	Sasi	BCom	70
4.	Pratheesh	ECE	55

o/p

Sname
mahi
chinnu
Sasi
Pratheesh

2) WAQTD Student name & branch of all the Students.

```
Select sname, branch
from Student;
```

3) WAQTD all the details of Students.

```
select *
from student;
```

Desc Dept;

Name	Null?	type
Deptno	Not Null	Number(2)
Dname		Varchar2(14)
Dc		Varchar2(13)

1. WAQTD all the details from Emp table.
2. WAQTD name of all the employees.
3. WAQTD name & salary given to the all employees.
4. WAQTD name & annual salary given to all the employees

1) Select *
from emp;

set pages 100 Lines 100

2) select ename
from emp;

3) select ename, sal
from emp;

4) Select ename, sal * 12
from emp;

5) WAQTD ID and deptno of all the emp in emp table.

Select empno, deptno
from emp;

6. WAQTD ename and hiredate of all the emp

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

7) WAQTP name and designation of all the emp

```
select ename, job
from emp.
```

8) WAQTP name & half-term sal of all the emp

```
select ename, sal * 6.
from emp
```

9) WAQTP ename and sal and sal with 25% hike.

```
select ename, sal, sal/4+sal
from emp;
```

10) WAQTP ename And sal, sal with deduction of 12% for all to emp.

```
Select ename, sal, sal - sal * 12/100
from emp;
```

$100 - 100 \times \frac{12}{100}$

$= 2$ ₹

Expression :-

A Statement which gives us result is known as expression.

Expression Consists of two types.

1. operand.
2. operators (+, -, *, /)

ALIAS :-

1) Alias is an alternative name given to a column name (or) expression in the result table.

2) Alias name can be used with (or) without using 'As' keyword.

3) Alias name should be single word or a string enclosed with double quote.

4) It is not mandatory but recommended to provide.

1) WAQTD sal as salary, Hiredate as Date of Joining.

```
select sal salary, Hiredate Date_of_Joining
from emp;
```

2) WAQTD name, sal of the emp along with their annual salary

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

3) WAQTD name, Job for all the emp with their Half term Salary.

```
select ename, Job, sal * 12 as "half term Salary"
from emp;
```

4) WAQTD all the details of emp along with an annual bonus of 2000.

```
Select *, sal*12 +2000 as "annual bonus"
from emp;
```

5) WAQTD name, sal and sal with a hike of 10%.
6) WAQTD name & sal with deduction of 25%.
7) WAQTD name & sal with monthly hike of Rs. 50.
8) WAQTD name & Annual sal with deduction of 10%.
9) WAQTD Total sal given to each employee (Sal + Com)
10) WAQTD details of all the emp along with annual sal
11) WAQTD name & designation along with 100 penality in sal.

1) WAQ to display different per from Student table

```
select Distinct per
from student;
```

Distinct :-

To remove repeated values or duplicated values in result table we use distinct clause.

* For distinct clause we can pass column name or an expression as an argument.
1) Distinct clause should be used as first argument in the select clause.
3) We can pass multiple columns for distinct clause.
4) It removes the combination of duplicates from all the columns.

Student :-

SID	Sname	Branch	Per
1	A	CS	70
2	B	EC	60
3	C	TC	80
4	D	EC	60
5	A	CS	70
6	E	ME	80

I

Per
70
60
80
60
70
80

II

Per
70
60
80

III

2) WAQTD diff branch, per from Student table

```
Select distinct per, branch
from Student;
```

branch	per
CS	70
EC	60
TC	80
ME	80

working

Branch	per
CS	70
EC	60
TC	80
EC	60
CS	70
ME	80

Assignment

1) WAQTD diff Studentname, branch & per from Student table.
2) WAQTD Diff Deptno from emp table.
3) WAQD Diff sal from emp table.
4) WAQD Diff Designations from emp table.

1) Select distinct sname, branch, per from Student.

Sname	branch	per
A	CS	70
B	EC	60
C	TC	80
D	EC	60
A	CS	70
E	ME	80

Sname	Branch	Per
A	CS	70
B	EC	60
C	TC	80
D	EC	60
E	ME	80

Selection :

Where clause :

This used to filter the records.

Note :

1. For where clause we can pass filter condition as an argument.
2. Where clause executes row-by-row.
3. Where clause executes after the execution of from clause.
4. We can pass multiple conditions for where clause using logical operators.

Syntax :

```
Select *, [Distinct] column_name / Expression [ALIAS]
From Table_name
Where <filter_condition>;
```

Order of execution :

1. From
2. Where
3. Select.

1) WAQTD emp name who is working in deptno 20.

I

③ Select Ename
① From emp
② Where Dno = 20;

II o/p of from clause. III

Ename	Sal	Deptno
RAJU	2000	20
Ajith	3000	10
Vijay	1000	30
Balayya	5000	20
Pawan	4000	10
Umesh	8500	30

row-by-row

Deptno
20=20 T
10=20 F
30=20 F
20=20 T
10=20 F
30=20 F

IV o/p of where clause

Ename	Sal	Deptno
RAJU	8000	20
Balayya	5000	20

V o/p of select

Ename
RAJU
Balayya

10. WAQTD Details of the Employ working as manager.

```
select *
from emp
where Job = 'manager';
```

11) WAQTD name & sal given to an emp, if Emp earns a commission of Rupees 1100.

```
select ename, sal
from emp
where comm = '1100';
```

12) WAQTD Details of Employs where Commission more than salary.

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

13) WAQTD empno of employees hired before the year 87.

```
select empno
from emp
where hiredate < '01-JAN-87';
```

14) WAQTD details of employees working as analyst

```
select *
from emp
where Job = 'analyst';
```

classmate
Date 23-Jan-21
Page

Operators :-

1). Arthmatic operators (+, -, *, /)
2) Comparison operator (=, !=)
3) Relation operator (<, > , <= , >=)
4) Logical operator (AND , OR, NOT)
5) Concatination operator (||)
6) Special op, (IN, NOT IN, Between, NOT between, Like, not Like, IS, IS NOT).
7). Sub-query operator (ALL, Any, EXISTS, NOT EXISTS).

1) logical operator :- (

AND :-

i) AND operator returns true if both the conditions are true. → (Binary multiplication).

AND operator should always be used between conditions.

I/p		O/p
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

0-F
1-T

I/p		O/p
C1	C2	R
F	F	F
F	T	F
T	F	F
T	T	T

ii) OR operator : (Binary Addition).

→ OR operator return ~~retrue~~ true if any one of the condition is satisfied (true).

→ OR operator is used always be used between conditions.

I/p		o/p
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

iii) NOT operator :

It is used negation

I/p	o/p
0	1
1	0

I/p	o/p
F	T
T	F

1) WAQTD ename if the emp is working in dept 20 and earning sal more than 1500

```
select ename
from emp
where deptno = 20 AND sal > 1500;
```

Logical Operator

1) WAPTD Details of the emp working as a clerk and Earning less than 1500

```
Select *
From Emp
Where Job = 'clerk' AND Sal < 1500;
```

2) WAPTD name and hiredate of the emp working as manager in Dept 30.

```
select ename, hiredate
from Emp
where Job = 'Manager' and Deptno = 30;
```

3) WAPTD details of the emp along with annual salary if they are working in Deptno 30 as Salesman & their Annual Sal has to be greater than 14000

```
Select Emp.*, Sal*12 as Annual Sal
from Emp
where Sal*12 > 14000 AND deptno = 30 And Job='Salesman';
```

4) WAPTD all the details of emp working in dept 30 (or) as a Analyst

```
Select *
from Emp
where deptno = 30 or Job = 'ANALYST';
```

9) WAQTD Details of Emp working as Analyst in DEPT 10

```
Select *
from emp
where Job = 'ANALYST' AND DEPTNO = 10;
```

10) WAQTD Details of Emp working as president with salary of rupees 4000.

```
select *
from emp
where Job = 'PRESIDENT' AND Sal = 4000;
```

11) WAQTD names and deptno, Job of emp working as clerk in dept 10 or 20.

```
Select ename, deptno, Job
from emp
where Job = 'CLERK' AND (deptno = 10
                         or deptno = 20);
```

Whenever we have both the conditions we should write in ()

12) WAQTD details of Emp working as clerk or manager in dept 10.

```
select *
from emp
where (Job = 'CLERK' or Job = 'Manager') AND deptno
= 10;
```

Concatination is used to join the given two Strings.

Interview

'MR. ABC YOUR salary is RS. XYZ'

↑

1) WAQTO the o/p in the following format

```
select 'MR.' || Ename || ' your salary is RS.' || sal
  from emp;
```

2). 'MR. SMITH your salary is RS. XYZ'.

```
select 'MR.' || Ename || ' your Salary is RS.' || Sal
  from emp
 where ename = 'Smith';
```

3). 'MR. king your designated as president.

```
Select 'MR.' || ename || ' your designated as ' || Job
  from emp
 where ename = 'king';
```

Special operator :-

In operator :-

→ In operator is a multi valued operator in which we can pass multiple values at RHS.

i.e In operator can accept multiple values at RHS.

In operator return true, if any one of the condition is satisfied.

In operator allows the value present at LHS to be compared with all the values present at RHS.

Syntax :-

Column_name / expression IN (v1, v2, ... vn);

Eg: 100 IN (50, 100, 150, 200).

Ex:

10 = 10,20
10 = 10 ✓
10 = 20 ✗

100 IN (100, 50, 150, 200)
100 = 50 ✗
= 100 ✓
= 150 ✗
= 200 ✗
} True.

2) WAQTD Fname and Job of emp's who are working as manager or salesman.

```
Select ename, Job
from emp
where Job IN ('manager', 'salesman');
```

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, ---vn);

1) WAQTD emp's name excluding the emp's working in dept 10 or 20 :

```
Select ename
from emp
where deptno NOT IN (10, 20);
```

10 NOT IN (10,20)

10 = 10 T
10 = 20 F
} True NOT → False.

Between :-

→ Between operator is used when ever we have ranges.

→ Between operator works including the ranges.

→ the range cannot be inter changed.

Syntax :-

Column name / Expression between lower_range AND higher_range.

1. WAQTD Emp's name and sal who are earning sal more than 1250 and less than 3000.

```
select ename , sal
from emp
where sal between 1250 and 2999;
```

2) WAQTD ename, sal if emp's are earning sal between 1250 and 3000. [it includes range]

between operator includ range

```
select ename, sal
from emp
where sal >= 1250 and sal <= 3000;
        ↓
    sal between 1250 and 3000;
```

NOT Between:

→ NOT Between is similar to between operator but it rejects the value instead of selecting it.

Syntax:

Column name, Expression NOT Between lower_range & h-r

1) WAQTD emp's name & sal if emp's are earning sal less than 1250 and emp's who are earning sal more than 3000.

```
Select ename, sal
  from emp
 where sal not between 1250 and 3000;
```

3) WAQTD ename, comm if they are getting comm less than 500 and emp's earning comm more than 700

< 500 & > 700

```
select ename, comm
  from emp
 where comm NOT Between 500 and 700;
```

3) WAQTD ename, hiredate who were hired after 1981 & before 1987

```
Select ename, hiredate
  from emp
where hiredate between '01-JAN-1982' and
                       '31-DEC-1986';
```

4) 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';
```

5) 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';
```

6) WAQTD ename, hiredate if they are working in dept no 10 and hired in hired in the year 1981.

7) WAQTD ename, hiredate who were hired after 1980 but before 1987 and working as manager or salesman.

```
Select ename, hiredate
  from emp
where deptNo = 10 or (hiredate between '01-JAN-1981'
                      and '31-DEC-1981');
```

IS OPERATOR :

→ IS operator is used only to compare with NULL.

Syntax :

Column_name | Expression is NULL;

1. WAQTD ename ~~if they —~~ if they doesn't get any sal.

```
select ename
from emp
where sal is null;
```

classmate
Date
Page

IS NOT:

→ IS NOT operator is similar to IS operator but it reverse the value instead of selecting it.

Syntax:

Column name / expression IS NOT NULL;

WAQTD ename and comm of emp's if they are earning comm.

```
Select ename, comm
    from emp
Where comm IS NOT null;
```

classmate
Date 01/03/21
Page

*** Like operators :

Like operator is used whenever we need to match the pattern

Syntax : Column_name / Expression like 'Pattern-To-Match' ;

To achieve pattern matching, we use special operators such as

1) % - Percentile : It can accept any character any number of times, even no character.

2) _ (under Score) :

underscore can accept any character but "only once".

*

1) WAQTD emp names who are having char A in their name.
2) WAQTD emp names who are having char A in the first place of their name
3) WAQTD emp name if they are having char A at the second place of their name.
4) WAQTD emp name if they are having char S in the last place of their name.
5) WAQTD emp name if they are having char A in the fourth place of their names.
6) WAQTD emp names if they are having char E in the second last of their name.

1) Select ename
from emp
where ename like '%A%' ;

2) select ename
from emp
where ename like 'A.%';

3) select ename
from emp
where ename like '_A.%';

4) select ename
from emp
where ename like '%g';

5) select ename
from emp
where ename like '___A.%';

6) Select ename
from emp
where ename like '%E_';

7) write the who are having char A in the first place and s in the last place

8) write sal, emp names of emp who are earning 3 digits sal.

a) write emp name & hiredate if they a
the year 82.

Not like :

Not like operator is similar to like operator, it rejects the value instead of selecting.

Syntax:

Column name / expression Not like.

1) WAQTD emp names who are having character '%' in their name.

```
Select ename
from emp
where ename like '%!_%!_%' escape '!';
```

classmate
Date
Page

Escape Character :

It is used to remove the special behaviour of a special character which is present next to it and treat it as ordinary char.

→ column_name/expression Like 'pattern' Escape 'CHAR';

1) WAQTD ename names who are having character percentile in the first place of their name

```
Select ename
From emp
Where ename like '!%%' Escape '!';
```

2) WAQTD ~~ename~~ who are having char (-) ^ in their name
Last

```
select ename
from emp
where ename Like '%!_%' Escape '!';
```

Functions.

Functions are nothing but list of instructions that are used to perform a particular task.

Functions

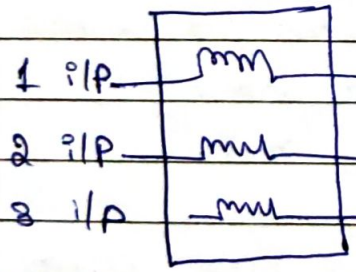
Inbuilt Function.

User defined

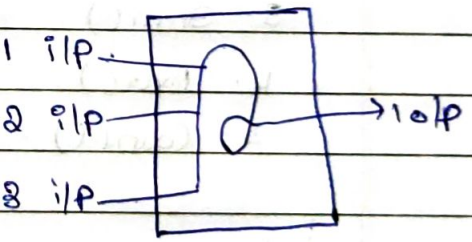
Single row

multi row (Group)
~~Aggregate~~ function.

Single row function:



multi row function.



↓ ingle

→ Single row function excutes row-by-row.

→ It takes one input and generates one output and then goes to the next input

→ If we pass 'i' no of inputs to Single row function, it returns 'i' no of outputs.

Multi-row function :

→ multi row function is also known as group function (or) aggregate functions.

→ It creates group by group.
→ It takes all the input at once aggregate it (combine) and generate one output.

→ If you pass 'n' of inputs to multirow function it returns a single o/p.

List of multirow functions :

1. Max()
2. Min()
3. Sum()
4. AVG()
5. Count().

Note :

1. multirow function can accept only a single argument that is a column name or an expression

2. max() and min() functions can be used for all the following datatypes also.
   ie char, varchar, Number and Date.

3. Sum() and Avg() Functions can only take number column as an argument.

4. multirow functions will ignore the null value.

5. We cannot use multirow functions in where clause.

6. We cannot use any column name with multi row functions in select clause.

7. Count() is the only MRF to which we can pass '*' as an argument.

1. WAQTD the max sal from emp table.
   Select max(sal)
   from emp;

2. WAQTD min sal from emp table.
   select min(sal)
   from emp;

3). WAQTD Total sal from emp table
   select Sum(sal)
   from emp;

4) WAQTD Avg sal from emp table
   select Avg(sal)
   from emp;

5) WAQTD max sal in dept 10.

```
Select max(sal)
from emp
where deptno = 10;
```

6) WAQTD no of emp's in emp table.

```
Select count(deptno) or (*) or (ename)
from emp
```

1. WAQTD number of employees getting sal less than 2000 in deptno 10.
2. WAQTD total sal needed to pay employees working as clerk.
3. WAQTD Avg sal needed to pay all employees.
4. WAQTD no of emp's having 'A' as their first character.

1.
```
Select Count (*)
from emp
where sal < 2000 And deptno = 10;
```

2.
```
Select sum(sal)
from emp
where Job = 'clerk';
```

3.
```
Select Avg (sal)
from emp;
```

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 expression as 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 creates groups and if any clause execute after the groupclause it execute group-by-group.

Syntax :

```
Select group_by_expression | group_function
  from table_name
  [where <filter_condition>]
  Group By column_name / expression;
```

Order of Execution:

1- Form
2- Where (if used) [Row-by-Row]
3. Group by [Row-by-Row]
4. Select [Group-by-Group].

10) WAQTD no. of Empls work in each dept.

③ Select Count(*), Dno
① from Emp
② group by Dno; (G-by exp)

II

ename	sal	DNo
Tommy	3000	20
John	2000	10
Sirkesh	5000	30
Kirsh	4000	20
James	1000	10

2·10
2·30
1·30

III

o/p of Group by

20

Tommy	3000	20	1
Kish	4000	20	2

10

John	2000	10	1
James	1000	10	2

30

Srilesh	5000	30	1

IV

o/p of select

Count(*)	DNO
2	20
2	10
1	30

Having Clause

→ We use having class to filter the groups.
→ We can pass multi-row Function Condition in having clause.
→ It executes group by group.
→ If you are 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)
3. Group by
4. Having
5. Select.

11) WAQTD no of emps in each dept if there are atleast 2 emp's working in each dept.

```
Select count(*), deptno
  from emp
  Group By deptno
  having count(*) >= 2;
```

emp

Ename	DNO
John	20
ANTO	10
Jack	30
Smith	20
James	10
King	20

o/p of from

o/p of GroupBy

20

John	20
Smith	20
King	20

10

ANTO	10
James	10

30

Jack	30

o/p of having count(*) >= 2

1 ✗
2 ✓
2 ✓

o/p select

Count(*)	DNO
3	20
2	10

1) WAQTD DNO & no of emp working in each dept. & if there are Atleast 2 clerks in each dept

```
Select count(*), deptno
From Emp
Group by deptno
where Job = 'CLERK'
having count(*) >= 2;
```

2) WAQTD Dno and total sal needed to pay all the emp in each dept & if there are Atleast 4 emp in each dept

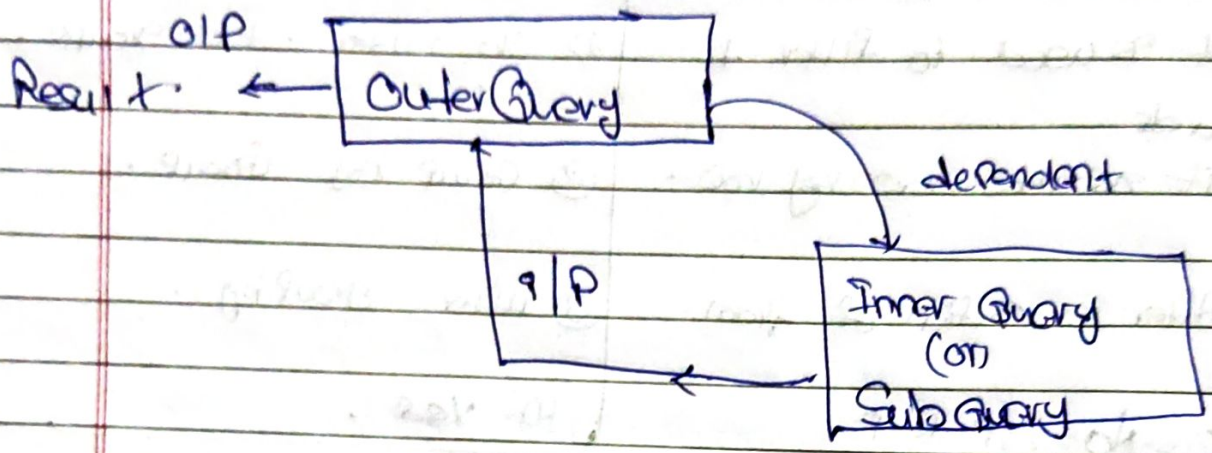
```
Select sum(sal), deptno
from emp
group by deptno
having count(*) >= 4;
```

3) WAQTD no of emp's earning sal more than 1200 in each Job. & total sal needed to pay emp of each Job must exceed 3800.

```
Select count(*), Job
from emp
where sal > 1200
group by Job
having sum(sal) > 3800;
```

Sub Query :-

* A query which is written inside another query is called sub-query.



1) Here we will be having minimum of 2 queries
   a) Inner query
   b) Outer query.

2) Inner query executes first and generates its o/p.

3) The o/p generated by the inner query will be given q/p to the outer query.

4) The outer query will execute and generate its o/p.

5) This o/p will be the result.

6) Hence outer query is dependent on inner query.

Why is Sub-query.

① Whenever we have unknown.

1) WAQTD emp name who are earning more than John.

```
select ename
from emp
where sal > (select sal
             from emp
             where ename = 'John');
```

2) WAQTD name & sal of the emp earning less than king

```
select ename, sal
from emp
where sal < (select sal
             from emp
             where ename = 'king');
```

3) WAQTD ename & deptno of the emp's if they are working in the same dept as Jones.

```
select ename, deptno
from emp
where deptno = (select deptno
                from emp
                where ename = 'Jones');
```

Page

WAQTD deptname of vinay.

```
select deptname
from emp
where deptno IN ( select deptno
                  from emp
                  where ename = 'vinay');
```

Case-ii

The data to be selected and the condition to be executed are present in different tables we use sub-query.

1. WAQTD dname and location of the emp whose name is King.

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

2. WAQTD loc of the emp whose employee number is 7902

```
select loc
from dept
where deptno in ( select deptno
                  from emp
                  where empno = 7902);
```

Nested Sub-Query :-

→ A Sub-Query Written inside another Sub-Query is known as nested Sub-Query.
→ We can nest about 255 Sub-queries.

5) WAQTD emp name who is earning 3rd max sal.

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

6) WAQTD 3rd min sal.

```
select min(sal)
from emp
where sal > (select min(sal)
             from emp
             where sal > (select min(sal)
                          from emp));
```

To find ~~4th~~ max or min sal, we will write (N-1) sub queries.

64) Write empno of the emp earning 2nd max sal.

65) Write deptname of an emp getting 4th max sal.

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

42) ~~Write~~

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

43)
```
Select ename, hiredate
from emp
where hiredate = (select Min(hiredate)
                  from emp);
```

44)
```
Select ename, hiredate
from emp
where hiredate = (select max(hiredate)
                  from emp);
```

classmate
Date 12-03-21
Page

Types of Sub-Query.

Two types

① Single row Sub-query.
② Multi row Sub-query.

1) Single row Sub-query :-

A sub-query which returns exactly one output is known as single-row-subquery.
We can use operations such as IN, Not IN, ALL, Any.

Ex :-

1. WAQ to ename earning more than SCOTT.

```
Select ename
from emp
where sal > (select sal
             from emp
             where ename = 'SCOTT');
```

2) Multi row-Sub-query :-

A Sub-Query which returns more than one o/p is known as multi-row subquery. We must use operators such as IN, NOT IN, ALL, Any.

A) ALL operator :-

ALL operator is a special operator which can accept multiple values at RHS, it will return true only if all the condition at the RHS is satisfied.

Syntax :-

Column_Name/Exp ... Relational op. ALL(v1, v2 ... vn);

Col_name / exp Relational op ALL (v1, v2, ... vn);

100 > ALL (50, 75, 90, 150)

100 > 50 T
> 75 T
> 90 T
> 150 F

} False.

b) Any operator :-

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

Syntax

Column Name / exp Relational op Any (v1, v2, ... vn);

100 > 50 T
> 75 T
> 90 T
> 150 F

} True.

Fname	Sal	Job
A	2000	PM
B	1000	PM
C	5000	manager
D	3000	PM
E	4000	manager

1) WAQTD the empnames who are earning more than all the salesman.

```
Select ename
from emp
where sal > (select max(sal)
             from emp
             where Job = 'salesman');
```

(or)

```
select ename
from emp
where sal > All ( select sal
                  from emp
                  where Job = 'salesman');
```

2000, 1000, 1300

× 2000 > All (2k, 1k, 3k)

✓ 1000 > All

✓ 5000 > All

✓ 4000 > All

2) WAQTD 'ename' who are earning more than any of the salesman.

```
select ename
from emp
where sal > Any (select sal
                 from emp
                 where Job = 'salesman');
```

```
select ename
from emp
where sal > ( select min(sal)
              from emp
              where Job = 'salesman');
```

classmate
Date 16/03/22
Page

Joins :

This statement is used to retrive the data from multiple tables simultaneously.

Types of Joins.

1) Cartesian Join (or) Cross Join.
2) Inner Join (or) Equi Join.
3) outer Join.
   a) Left outer Join.
   b) Right outer Join.
   c) Full outer Join.
4) Self Join.
5) Natural Join.

1) Cartesian Join (or) Cross Join :

In Cartesian Join a record from table1 will be merged with all the record of table2.

Syntax :

1) ANSI : Select Column_name
   From table_name1 Cross Join Table_name2 ;

2) ORACLE : Select Column_name
   From Table_name1 , Table_name2 ;

1) WAQTO Bname and Gname.

(I) select bname, gname
from Boys, Girls;

(II)

Boys. 3

BID	Bname	GID
1	virat	22
2	ABHI	33
3	Ranveer	11

Girls (2)

GID	Gname
11	DEEPIKA
22	ANUSHKA
33	AISH

m = 3

(III)

BID	Bname	GID	GID	Gname
1	virat	22	11	DEEPIKA
1	virat	22	22	ANUSHKA
1	virat	22	33	AISH
2	ABHI	33	11	DEEPIKA
2	ABHI	33	22	ANUSHKA
2	ABHI	33	33	AISH
3	RANVEER	11	11	DEEPIKA
3	RANVEER	11	22	ANUSHKA
3	RANVEER	11	33	AISH

→ No. of columns in Result table will be sumation of columns Present in table1 and table2.

→ No. of 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.

Inner Join :-

We use Join Condition on which we merge two tables to get only the matched records.

Syntax :

Table_name 1.Column_name = Table_name2.Col_Name;

Eg:

Emp.DNo = Dept.DNo.

Select BName, Gname
From boys, girls
Where boys.GID = girls.GID;

Join condition.

BID	Bname	GID	GID	GName
1	virat	22	22	Anushka
2	abhi	33	33	Anish
3	Ranbir	11	11	peepika

Table = girls table

22 = 11 F
22 = 22 T
22 = 33 F

33 = 11 F
= 22 F
= 33 F

11 = 11 T
= 22 F
= 33 F

Syntax:

1) Oracle:

```
select Column_name
from Table_name1, Table_name2
where <Join_Condition>;
```

2) ANSI

```
select Column_name
from table_name1 Inner Join Table_name2
ON <Join_Condition>;
```

ex:

```
select *
from emp Inner Join dept
ON emp.deptno = dept.deptno;
```

1) WAQTD ename, sal and dname of all the emp
2) WAQTD ename, deptname, deptno.
3) WAQTD ename, dname if emp's working in dept 30.

1)

```
Select ename, sal, Dname
  from emp, dept
where emp.deptno = dept.deptno;
```

2)

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

Natural Join:

In natural join we won't be writing any join condition.

If the table contains similiar columns we get the o/p of inner join. else we get o/p of Cartesian Join.

Why or when we use natural join?
Whenever there is no table structure we use natural join.
↑ when you don't know column names previously

Column that are present ... table structure.

Syntax:

ANSI : Select column_name
From Table_Name1 Natural Join table_Name2;

Eg: select *
from emp Natural Join dept;

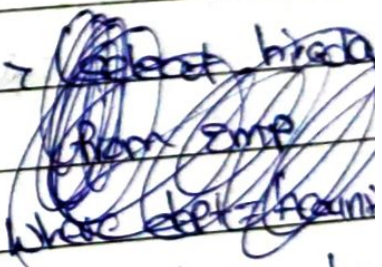
OR: select *
from emp Natural Join SALGRADE;

1) WAQTD Dname and mgr-no for employees reporting to 7839.

```
Select Dname, mgr
from emp e , dept d
where e.empno = d.deptno and mgr = 7839 ;
```

2) WAQTD Dname & hiredate for emp hired after 83 in to Accounting or research dept

```
Select Dname, hiredate
from emp e, dept d.
where e.empno = d.deptno and hiredate > '31-Dec-1983' and dname in('Acc.
```



Outer Join :

In outer join we get the unmatched records along with matched records.

1. Left outer Join :

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

Left outer Join (Syntax) :

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 (+);

1) WAQTD all the details of employees and department along with unmatched of employee table.

emp - I

Ename	DNO
A	20
B	null
C	30
D	null
E	10

dept

Dname	DNO
D1	10
D2	20
D3	30
D4	40

II

matched records

Ename	DNO	Dname	DNO
A	20	D2	20
C	30	D3	30
E	10	D1	10
B	null	null	null
D	null	null	null

unmatched records from left table.

```
Select * (ename)
from emp e, dept d
where e.deptno = d.deptno(+);
```

2) WAQTD ename from who is not working in any dept using joins.

```
Select *
from emp e, dept d
where e.deptno = d.deptno(+) and e.deptno = null;
```

2. Right outer Join :

In this Join we get unmatched records of right table along with matched records.

Syntax :

ANSI

1) Select Column_name ,
from table_name1 Right [outer Join] Table_name2.
on <Join_condition> ;

Ex :

Select *
from emp e right 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_name2 ;

Ex: select *
from emp e, dept d
where e.deptno(+) = d.deptno ;

1) WAQTD all the * from Emp and dept table along with unmatched records from dept table

Select *
from emp e , dept d
where e.deptno(+) = d.deptno

Full outer Join :

To obtain unmatched record of both the tables along with match records.

Syntax :

1. Ansi : Select Column_name
from table_name1 FULL [OUTER] JOIN table_name2
ON <JOIN_Condition> ;

Ex :- Select *
From EMP E FULL OUTER JOIN DEPT D
on E.DEPTNO = D.DEPTNO ;

no syntax in oracle

Select *
From emp e full outer join dept d
on e.deptno = d.deptno ✓

table for both

classmate
Date 22/08/24
Page

Self Join :-

It is used to join the same two tables (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 :

~~Ans~~ :
oracle

```
select Column_name
from Table_Name1, [redacted] Table_Name T2
[redacted] where <Join_Condition> ;
```

Ex

```
select *
from Emp E1, Emp2
where E1.mgr = E2.empno ;
```

1) WAQTD emp name and mgr name from same table.

Ans

(I)

```
Select E1.Name , E2.Name
from Emp E1, Emp E2
where E1.mgr = E2.Empno ;
```

Emp E1 (II)

EID	Ename	mgr
1	A	2
2	B	4
3	C	4
4	D	5
5	E	null

Emp E2

EID	Ename	mgr
1	A	2
2	B	4
3	C	4
4	D	5
5	E	null

(III)

02 = 1F
= 2T
= 3
= 4
= 5
} F

4 = 1F
= 2F
= 3F
= 4T V
= 5F
} F

4 = 1F
= 2F
= 3F V
= 4V
= 5
} F

5 = 1F
= 2F
= 3F
= 4F
= 5V
} F

null = 1F
= 2F
= 3
= 4
= 5

Anything × null = null

e1. EID	e1. Ename	e1. mgr	e2. EID	e2. Ename	e2. mgr
1	A	2	2	B	4
2	B	4	4	D	5
3	C	4	4	D	5
4	D	5	5	E	null

O/P

e1.ename	e2.ename
A	B
B	D
C	D
D	E

1) Display name of the emp & his manager's name if emp is working as clerk

```
Select e1.name , e2.ename mgr_name
from emp e1 , emp e2
where e1.mgr = e2.empno and e1.job = 'clerk';
```

2) Display name of the Emp & manager's designation if manager work in dept no 10 or 20.

```
select e1.ename, e2.job mgr-job
from emp e1 , empe2
where e1.mgr = e2.empno and e2.deptno in (10,20);
```

3) select name of the emp and manager sal no emp and mgr both earn more than 2300.

```
select e1.ename, e2.sal mgr_sal
from emp e1 , empe2
where e1.mgr = e2.empno and e1.sal >2300 and
e2.sal > 2300;
```

Single Row Function :

→ Single row function executes Row-by-row.
→ It takes one input and generates one o/p and then goes to the next Input.

Dual :

Dual is a dummy table to print the result of any mathematical operations alone.

1) Select upper(ename)
from emp; all in upper
if lower → all in lower.
initCap → Mahesh

2) Select initCap('JSPIDERS')
from dual;
o/p → Jspider.

if ('My Earth')
→ My Earth.

1) Upper :

These function is used to convert the given String in to upper case.

Syntax: UPPER('STRING')

2) Lower :-

This function is used to convert the given String in to lowercase.

Syntax :-

Lower ('string')

3) InitCap :-

This function is used to convert the initial character of the given string in to the upper case.

Syntax :- InitCap ('string').

4) Length() :-

This function is used to count the number of character present in the string

Syntax :- Length ('string').

1) WAPTD no. of characters present in name's for all the emp's

select length(ename)
from emp;

2) WAQTD ename who are having only 5 character in their name using SRF

```
Select ~~length(5)~~ ename
from emp
where length(enam) =5 ;
```

3) WAQTD ename and sal of Emp who are getting 3 digit sal using SRF

```
Select ename , sal
from emp
where length(sal) =3 ;
```

4). WAQTD ename and comm of emps who are getting 3 digit comm

```
Select ename , comm
from emp
where length(comm) = 3 ;
```

Reverse:

This function is used to reverse the given string.

Syntax: Reverse('STRING').

Ans

```
Select reverse('Mahesh')
from dual;
```

6. SUBSTR ( ):

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

Syntax : SUBSTR ('original_string', Position, [Length]).

SUBSTR ('BANGALORE', 1, 1) → B
1, 4) → BANG
9, 5) → E
1) → BANGALORE
6) → LORE
7, 5) → ORE

SUBSTR('BANGALORE', -1, 1) → E
-3, 3) → ORE
-7, 4) → NGAL
-5)
-8).

7. WAQTD the first character of all the emp

```
select substr(ename, 1, 1)
from emp;
```

6) select ename, job
from emp
where substr(job, 1, 3) = 'MAN' or substr(job, -3, 3) = 'MAN';

7) INSTR() :=
This function is used to obtain index value of the substring which is present on the original string.
Index value - position of character.

Syntax :=
INSTR ('ORIGINAL_STRING', 'SUB_STR', POSITION, [Nth occurance]).

Nth occurance - no. of times it is present.

INSTR ('BANANA', 'A', 1, 1) → 2
'A', 1, 2) → 4
'N', 3, 1) → 3
'AN', 1, 1) → 2
'AN', 3, 1) → 4
'A', 1, 4) → 0

BANANA
1 2 3 4 5 6
-ve positions available.

If you don't provide occurance by default it will select first occurance.

9) To_char()

This function is used to Convert the given date to string format.

Syntax:

To_char(date, 'format_models')

Format models:

1. year	- twenty two	7. day - Thursday	13) MI - 12
2. YYYY	- 2022	8. dy - Thu	14) SS - 16
3. YY	- 22	9. DD - 25	
4. month	- August	10 D - 05	
5. mon	- aug	11) HH24 - 12	
6. MM	- 3	12) HH12 - 12	

SYSDATE:

This is used to obtain the present date

→ This command is used to obtain the Current Date from database.

→ sysdate / current - date.

→

Select To_char(To_date('15-AUG-1947', Day)
From Dual;

→ To_Date() :

→ This function is used to convert the Date string to Date format.

Syntax :
To_Date ('Date_str')

▷ WAQ TO total sal given to each Emp.

```
Select Sal+Comm
from emp
where Comm != NotNull;
```

NVL() : [Null value logic]

Syntax : NVL (Arg1, Arg2)

It Can accept 2 Arguments

→ In Arg1 we must write a 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.

Emp

(I)

Name	Sal	Comm
A	500	50
B	700	
C	800	100
D	600	

(II)

550
900

(IV)

Sal + NVL (comm, 0)
550
700
900
600.

(III)

Select Sal + NVL (Comm, 0)

500 + nvl (50, 0) → 500+50 → 500+50 = 550.

700 + NVL (NULL, 0) → 700 + 0 = 700

800 + NVL (100, 0) → 800+100 = 900.

600 + NVL (NULL, 0) → 600+0 = 600.

(v) Replace () :-

This function is used to replace the substring with newstring in given original string.

Syntax:

Replace ('ORIGINAL_string', 'SUB_STR', ['NEW_STR'])

classmate
Date 29/03/21
Page

Pseudo Columns.

Pseudo Columns are the false Columns that are present in each and every table and must be called explicitly.

Pseudo Columns cannot be seen without calling them.
There are two types
1. Row ID
2. Row Number.

1. **ROWID:**

RowID is an 18 digit address in which the record is present or the record 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 record.
→ RowID is generated at the time of insertion of records.
→ It can be inserted, updated or deleted.
→ Empty table will not be having RowID.
→ RowID is static in nature.
→ RowID is uniquely identify a record from the table when there is no key attribute or Primary key.

Rownum :-

Rownum acts as a serial number to 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 starts with 1.
→ Rownum cannot be duplicated.
→ Rownum gets incremented after it is assigned.

1) WAQTD details of first three records.

Rownum = 1

emp

EName	DNO
A	20
B	10
C	30
D	20
E	10

Rownum < 4

1 A 20 → 1<4 T
2 B 10 → 2<4 T
3 C 30 → 3<4 T
D 20 → 4<4 F
E 10

```
Select *
from emp
where RowNum <4;
```

WAP to find the third record from emp table.

```
Select RowNum, emp.*
from emp
where RowNum = 3;
```

RowNum=1

RowNum=3

emp

Ename	DNo
A	20
B	10
C	30
D	20
E	10

	A	20
	B	10
	C	30
	D	20
	E	10

No Row Selected.

To make row num as static:

→ Take a table and assign rownum to a given table
→ Change the rownum to any other name by using alias (SLNo).
→ Use this as a subquery in from clause of outer query.
→ In the outer query use the alias name in the Condition.

1> Write the third record from emp table.

(I) emp

Ename	DNo
A	20
B	10
C	30
D	20

(II)

```
Select Rownum Slno, emp.*
From emp;
```

→

Slno	Ename	DNo
1	A	20
2	B	10
3	C	30
4	D	20

(III)

```
Select *
from ( Select Rownum Slno, emp.*
       from emp)
where slno =3;
```

1) WAQTD 5th Record from the emp table.
2) WAQTD first four record from emp table.
3) WAQTD ename, sal from seventh Record.
4) WAQTD of 1st, 3rd, 5th and 8th record.
5) WAQTD ename of 8th, 7th and 6th Rearord.

7) WAQTD 12th max sal from emp table.

```
Select Sal
from (select rownum SLNo, Sal
      from emp
      order by sal desc)
where SLNo = 12;
```

order by clause :

```
Select sal                    In Ascending
from emp
order by Sal Asc;

select sal
from emp
order by sal desc;
```

To find N^{th} min & max sal using Rownum Concept.

Syntax for N^{th} max sal.

```
Select sal
from ( Select Rownum Slno, sal
       From (select Distinct sal
             from emp
             order by sal desc))
where slno = n;
```

1) WAQTD 3rd max sal.

```
select sal
from ( select Rownum slno , sal
       from ( select Distinct sal
              from emp
              order by sal desc))
where slno = 3;
```

To find n^{th} min sal syntax

```
Select sal
from ( select Rownum slno , sal
       from (select Distinct sal
             from emp
             order by sal ASC))
where slno = n;
```