CLOB : It stores plain character data upto 4 GB
This data is readable.
From 11g, 1 Terabyte of data can be stored.

BFILE : is similar to BLOB, Stores Files, Music Files, videos, Images

BFile just stores the path but file is actually stored at the OS Level ie., BFile is called external LOB whereas BLOB & CLOB are called internal LOBs, BLOB stores data inside the DB, BFile stores data External to the DB.

STATEMENTS

DQL [Data Query Language] / DRL [Data Retrieval Language]
=> SELECT

DDL [Data Definition Language]
=> CREATE, ALTER, DROP, RENAME, TRUNCATE

DML [Data Manipulation Language]
=> INSERT, UPDATE, DELETE, MERGE, EXPLAIN PLAN

DCL [Data Control Language]
=> GRANT, REVOKE

TCL [Transaction Control Language]
=> COMMIT, ROLLBACK, SAVEPOINT

SCL [Session Control Language]
=> ALTER SESSION

SCL [System Control Language]
=> ALTER SYSTEM

① DDL's helps to create & alter the structure of the db object.
They are implicitly commited & Saved.

② DML's They helps to manipulate the data in the table we need to explicitly commit them.
Any unwanted changes performed by DML's can be rolled back, & Undone. whereas DDL's changes cannot be rolled back.

DDL'S

Create :

Prod	Ord
pid (PK)	Ordid (PK)
Pnm (NN)	Pid (FK)
	Qty (>0)
	Ord-dt

=> Create table Prod
(
Pid Number Primary Key,
Pnm Varchar(9) Not Null
)
/

=> Create table Ord
(
Ordid Number(5) Primary Key,
Pid Number References Prod(Pid),
Qty Number(2) Checks (Qty > 0),
Ord-dt Date
) price Number(9,2) default 0
/

Column level Constraints

→ Create table Ord
(
Ordid Number(5),
Pid Number,
Qty Number(2),
Ord_dt Date Not Null → Table Level Constraints
Constraint Ord_PK1 Primary Key (Ordid),
Constraint Ord_FK1 Foreign Key (Pid) References Prod(Pid),
Constraint Ord_Chk1 Check (Qty>0)
)

Methods of providing Constraints ① COLUMN LEVEL ② TABLE LEVEL

* Differences between Column Level & Table level constraints

→① Column level constraints are given along with the column whereas Table level constraints are provided after providing all the columns.

→② In Column level constraints, Constraint name is not Mandatory whereas in Table level constraint, Constraint name is Mandatory.

→③ In Column level constraints we cannot provide the composite keys whereas in Table level constraints, we can provide composite keys.

→④ In table level constraints we can provide user defined constraints, but we cannot provide user defined constraints in column level constraints it will take default table level constraints

* CTAS : Create Table As Select

Create table temp
as
Select *
From dept;

Copies both data & Structure of table dept.
(Taking backup & copying table)

Here all the rows & columns are copied but not the constraints, however only Not Null constraint will be copied.

* To copy only the structure but not the data.

→ Create table temp100
as
Select * From dept
where 1=2; → Universal False Condition
(or) where 100=200 (or) 'a'='b'

DROP & TRUNCATE

DROP : Removes both data & structure of the table.
(Removes the table from db)

TRUNCATE : Removes only the data but not the structure

→ Drop table t1; SQL> Desc t1;
→ Truncate table t2; SQL> Select * from t2;

→ Create table t1 as select * from dept;
→ Create table t2 as select * from dept;

* Drop can be flashed back from 10g onwards. whereas Truncate cannot be flashed back.

Oracle 10g Recyclebin

Table → Drop → Recyclebin → Flashback → Restore it
Recyclebin → Purge → Remove permanently

=) Flashback table t1 to before drop;
=) drop table t1; - Remove from db
=) Purge table t1; - Remove from Recyclebin
=) drop table t1 purge;
=) Show recyclebin;
=) Purge recyclebin;

Rename - It renames a table
=) Rename t2 to t99;

ALTER: is used to change the structure of the table

Adding a Column
Alter table Prod Add Prod_Comments Varchar(20) not null;

Dropping a Column
Alter table Prod Drop Column Prod_Comments;

Rename a Column
Alter table Prod Rename Column Pid to Prod_id;

Add / Remove a Constraint

Alter table Prod Add Constraint Prod_PK Primary Key(Pid);

Alter table Ord Add Constraint Ord_FK Foreign Key (Pid) References Prod (Pid);

Alter table Ord Drop Constraint Ord_FK;

Disabling / Enabling a Constraint

Alter table Ord Disable Constraint Ord_chk;

Alter table Ord Enable Constraint Ord_chk;

Modify (should be used along with Alter)

- Changes the datatype
- Changes the Size
- Adds / Removes the default / Not Null constraints only

Alter table Prod Modify Pid Number(6); Size (5 to 6)
Alter table Cost Modify Phone Varchar(10);
↳ From Number to Varchar

(#) To change the datatype, column should be empty

Alter table Prod Modify Prod_desc Not Null;
(#) => No value should be Null

Alter table Prod Modify Prd_desc Null;

Alter table Prod Modify Prd_desc default 'A';
↳ To give default

Alter table Prod Modify Prod_desc default Null;
↳ To remove default

DML's

INSERT: inserts a record to a table

=) Insert into Prod values (1001, 'desktop');

=) Insert into Prod (Pid, Pname) values (1008, 'laptop');

=) Insert into Prod values (#Pid, '# Pnm', '# Prod_desc');
↳ Varchar fields

→ Insert into Emp1 Select * From Emp where deptno=20;

→ insert into Emp1 (Empno, Ename, Sal, deptno)
Select Empno, Ename, Sal, deptno
From Emp
where deptno = 10;

→ Insert into Ord (Ord_id, Prd_id, Qty)
values (7777, 1001, 2);

UPDATE → Updates One or More columns

Update Prod
Set Pname = 'DVD', Prod_desc = 'player'
where Pid = 101;

Update Emp
Set Comm = (Select Max(comm) From Emp
where deptno = 30);

Update Emp
Set Sal = Null; → Truncating a column

Update Emp
Set Ename = 'Scott', Sal = 9000
where Empno = 7788;

NOTE: We cannot delete a Master record when it is having associated child records

DELETE

Delete from Emp where deptno = 20;

Delete from Emp;

Sequence of Create & Delete

Create & Insert data ① Master
② Child

Delete ① Child
② Master

Prod – Master Table Pid-PK
Ord – Child Table Prd_id-FK
Oid-PK

On Delete Cascade

→ Alter table Ord Add Constraint Ord_FK2
Foreign key (Prd_id) References Prod(Pid) on delete cascade

Generally we cannot delete the Master data, when it is having associated child records.
If we want to delete All the child records along with the Master record, we have to use on delete cascade option. It should be given during the creation of foreign key.

Drop table Prod cascade constraints;

We cannot drop the Master table, when it is having associated child tables, If we want to drop the same then we have to use Cascade Constraints

TCL - Transaction Control Language

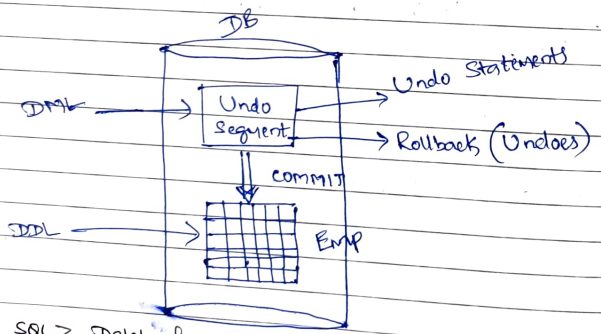
These commands help to manage the DML's statements
✓ Rollback ✓ Commit ✓ Savepoint

Rollback:

⇒ It rollsback any uncommitted DML statements
⇒ When we issue a rollback, the data will be rolled back till previous rollback (or) till previous commit (or) till start of the session.
⇒ The rollback never depends on the table but depends on the statements.

COMMIT:

⇒ It saves any DML changes permanently to the db.
⇒ Once we commit we cannot rollback & vice-versa
⇒ The DDL statement cannot be rollbacks as they are implicitly committed. (SAVE)



Eg: SQL> Delete from Emp;
SQL> Select * from Emp;
O/P: No data
SQL> Rollback;

SQL> Insert ...
SQL> Insert ...
SQL> Commit;

Scenarios:

① Delete from X;
Update Y set Sal = 2000;
Insert into Z values (1000);
Rollback;
} All the statements will be Rolled backs.

② Delete from X;
Update Y Set Sal = 2000;
commit;
Insert into Z values (1000);
Rollback;
} Only insert is rolled backs & rest are committed.

③ Delete from X;
Update Y Set Sal = 2000;
Alter table abc Add comm Number;
insert into Z values (1000);
Rollback;
} Only insert is rolled back & rest are committed

(*) DDL Statement, It will commit itself and commits the above DML statements implicitly.

④ Delete from X;
Commit;
Rollback; → It is No use
} Nothing is rolled backs as commit has occured first

(#) ASSIGNMENT

① Difference between Truncate & Delete
② Difference between DDL & DML
③ Difference between Drop & Truncate
④ Difference between Char & Varchar

SAVE POINT: is like a pointer (marker) till where the DML's statements can be rolled back.

* It helps in selective rollback.

① Insert
Update
Save point X;
Delete
Roll back to X;

Partial Rollback
Rolled back till X
→ Only Delete is rolled back

② Insert
Update
Savepoint X; → Killed
Delete
Roll backs;

Complete Rollback

③ Insert
Update
Save point X; → Killed
Delete
Commit; → All are commited & Savepoint is killed
Roll back to X; → Error
⇒ Commit's all the DML's and throws an error "Savepoint X" not exists"

④ SQL> Insert
Update
Savepoint X;
Delete
Rollback to X;
Commit;

Only Delete is rolled back & Insert & Update is Commited

⑤ Insert
Update
Savepoint X;
Delete
Insert
Savepoint Y;
Insert
Update
Rollbacks to Y
Rollback to X

NOTE
* We can rollback to a savepoint we cannot commit to a savepoint.
* In order to rollback first 2 statements [Insert, Update] before savepoint X, we need to rollback entire statements including savepoint X & savepoint Y i.e., why savepoints are not so flexible.

TRANSACTION:

⇒ A transaction is a set of DML statements which can be committed (or) Rolled back at once.

→ Transaction will be ended whenever we issue a commit (or) Rollback (or) DDL Statements

① Insert → Txn1 Starts
Update → —u— (continues)
Delete → —u—
Rollback | commit; → —u— ends

② Insert → txn2 Starts
Update → —u— continues
Alter → —u— ends
DDL Statement

NOTE
→ Incase of abnormal exit (power off → (Power crash (or) Click (x) button)) oracle automatically Rollback any uncommitted DML Statements
→ In case of normal exit, it automatically commits. (SQL> Exit;)

Sub-Queries

-> Subquery means nested query (Query within a Query)

Select * From X → Outer Query
Where ... (Select * From Y where .....) → Inner Query

NOTE: Output of Inner Query is passed as Input to the Outer Query

⇒ Display all the Emp from Research dept.

```
Select * From Emp
Where deptno = ( select deptno from dept
                 where deptname = 'Research');
```

```
Select * From Emp
Where deptno = ( Select deptno from dept
where dname in ('Accounting', 'Sales'));
```

o/p: Error (Inner Query returning more than One value)

Modify Query

```
Select * From Emp
Where deptno IN ( Select deptno from dept
where deptname in ('Accounting', 'Sales'));
```

⇒ Here IN is used to retrieve multi-record

① Display the deptname of the King

```
Select dname from dept
where deptno = ( Select deptno from Emp
      where Ename = 'King');
```

NOTE: We can have 32 Levels of nested subquery

② Display Emp who are earning same sal as of Ward

```
Select * From Emp
Where Sal = ( Select Sal from Emp
    where Ename = 'Ward')
And Ename != 'Ward'; → When ward is not required to display
```

③ Display 2nd highest Sal

```
Select Max(Sal) from Emp
where sal < max(sal); ⇒ We cannot use Aggregate function in where

Select Max(Sal) from Emp
where Sal < ( Select Max(Sal) from Emp);
```

④ Display 3rd highest Sal

```
Select Max(Sal) from Emp
where Sal < ( Select Max(Sal) from Emp
      where Sal < ( Select Max(Sal) from Emp)
);
```

⑤ Display 1st highest Sal

```
Select Max(Sal) from Emp;
```

⑥ Display the Mgr Name of Adams

Name from Mgrno - 2nd
Find Mgrno 1st

```
Select Ename from Emp
where Empno = ( Select Mgrno from Emp
      where Ename = 'Adams');
```

⑦ Display all the employees reporting to Blake

```
Select Ename from Emp
Where MgrNo = (Select Empno from Emp
      where Ename = 'Blake');
```

(*) Paired columns in the Sub-query

```
Select * From tablename1
where (col1, col2) in (Select col1, col2 from tablename2)
Here, Tab1.col1 = Tab2.col2
and Tab1.col2 = Tab2.col2
```

(1) Display all the emp who are earning highest salary in respective depts.

```
=> Select * From Emp
   where (deptno, Sal) in (Select deptno, Max(Sal)
                           From Emp
                           Group by deptno);
```

SUB-QUERY OPERATORS

(1) Single Row Operator

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

(2) Multi-Row Operator → Cannot be used alone

IN, NOT IN, (Any, ALL, <Any, >Any, <ALL, >ALL).

Ename	Job	Sal
A	Manager	2000
B	Manager	5000
C	Clerk	9000
D	Manager	11000
E	Clerk	1000
F	Sales Manager	6000
G	Sales Manager	12000

(1) Display the emp who are earning more than any of the managers

```
=> Select * From Emp
   where Sal > ( Select Min(Sal) from Emp
                 where Job = 'Manager')
                 And Job != 'Manager';
```

o/p: C, F, G

(OR)

```
Select * From Emp
where Sal > Any ( Select Sal from Emp
                  where Job = 'Manager')
                  AND Job != 'Manager';
```

(2) Display the emp earning more than All the managers

```
Select * From Emp
where Sal > ALL ( Select Sal from Emp
                  where Job = 'Manager')
                  And Job != 'Manager';
```

Restriction of Sub-query

=> We cannot use ORDER BY CLAUSE inside the inner query but ORDER BY can be used in outer query

=> Sub-query cannot fetch the data from 2 different tables. This problem can be overcome by JOINS

=> Subqueries will not fetch the data from multiple tables at once.

Assignments

1. List the dname which is having No employees (use NOT IN)

```
=> Select dname from dept
   where deptno NOT IN ( Select distinct deptno from Emp);
```

(1) Select dname from dept
where deptno IN (Select deptno from dept
MINUS
Select deptno from Emp);

(2) List the deptname which is having atleast one employee [use. IN]
Select dname from dept
where deptno IN (Select distinct deptno from Emp);

(3) List the emp who dont have any reportees under them.
Select * From Emp
where Empno NOT IN (Select Mgr from Emp
where Mgr is not null);

(or)
Select Empno from Emp
Minus
select Mgr from Emp;

(4) Select all the employees who are in dallas
Select * from Emp
where deptno IN (Select deptno from dept
where Loc = 'Dallas');

(5) Select all the departmental information for all the managers
Select * from dept
where deptno IN (Select deptno from Emp
~~where Mgr is not null~~),
where Empno IN (Select Mgr from Emp
where Mgr is not null)
);

(*) : When 'Job' column not available in Employee table & Going by 'Mgr' column.

(*) If 'Job' column is available in Emp table
Select * from dept
Where deptno IN (Select deptno from Emp
where Job = 'Manager');

(6) List the managers & clerks who works in Accounts & Sales dept.
Select * from Emp
where Job IN ('Managers', 'Clerks')
And deptno IN (Select deptno from dept
where dname IN ('Accounts', 'Sales')
);

(7) List the Salesman who are not located at dallas
Select * from Emp
where Job = 'Salesman'
and deptno IN (Select deptno from dept
where Loc <> 'Dallas');
(or) and deptno Not IN (Select deptno from dept
where Loc = 'Dallas');

(8) List employees who are earning same as Smith but Smith should not be displayed
Select * from Emp
where Sal = (Select Sal from Emp
where Ename = 'Smith')
And Ename != 'Smith';

(9) List the employee name who is earning 1st max salary
Select * from Emp
where Sal = (Select max(sal) from Emp);

10. List the deptname who is earning the 1st max salary.

```
Select dname from dept
where deptno ≠N (Select deptno from Emp
                 where sal = (Select max(sal) from Emp)
                 );
```

11. List the 2nd least salary

~~Select min(sal) from Emp~~
~~where sal > (~~
Select Min(sal) from Emp
where Sal > (Select Min(sal) from Emp);

12. List top 2 Salaries (top1 union top2)

```
Select max(sal) from Emp
UNION
Select max(sal) from Emp
where sal < (Select max(sal) from Emp);
```

~~13.~~ ~~List the Emp who don't have any reportees under them.~~

13. List the employees who are earning more than the average Salaries of all the employees.

```
Select * from Emp
where Sal > (Select Avg(Sal) from Emp);
```

14. List the Emp name from Accounts & Sales depts.

```
Select Ename from Emp
where deptno IN (Select deptno from dept
                 where dname IN ('Accounts', 'Sales'));
```

15. List the mgr name who is having atleast 2 employees under him (Reporting to him)

```
Select Ename from Emp
where Empno IN (Select Mgr from Emp
                Group by Mgr
                having count(*) ≥ 2);
```

16. List the employees who are earning the min salary in each of their Job (paired columns)

```
Select * from Emp
where (Job, Sal) IN (Select Job, Min(Sal) from Emp
                     Group by Job);
```

17. List the employee name who has joined the company first?

```
Select * from Emp
where DOJ = (Select Min(DOJ) from Emp);
```

18. List the departname which has made the most recent hiring.

```
Select * from dept
where deptno IN (Select deptno from Emp
                 where DOJ = (Select Max(DOJ) from Emp)
                 );
```

19. List the clerks & Managers who are located in New yorks & reporting to King

```
Select * from Emp
where Job IN ('Clerks', 'Managers')
And Deptno = (Select deptno from dept
              where loc = 'New yorks')
And Mgr = (Select Empno from Emp
           where Ename = 'King');
```

20. Display all the emp who work in same dept. as of SCOTT

```
Select * from Emp
where deptno = (select deptno from Emp
                where Ename = 'SCOTT');
```

21. Select Empno, Job & Salaries of all the analyst, who are earning more than any managers

```
Select Empno, Job, Sal from Emp
where Job = 'Analyst'
And Sal > Any (select Sal from Emp
               where Job = 'Managers');
```

22. Display deptname which is having more than 4 Employees

```
Select dname from dept
where deptno IN (select deptno from Emp
                 Group by deptno
                 having count(*) > 4);
```

23. Display the manager name of 'MARTIN'

```
Select Ename from Emp
where Empno = (select Mgr from Emp
               where Ename = 'Martin');
```

24. Display the no. of employees reporting to blake.

```
Select count(*) from Emp
where Mgrno = (select Empno from Emp
               where Ename = 'Blake');
```

25. Display employees who are having atleast 3 person reporting to them.

```
Select * from Emp
where Empno IN (select Mgr from Emp
                Group by Mgr
                Count(*) >= 3);
```

26. Display 1st & 2nd highest salary in single query

27. List the 3rd Maximum Salary

28. List the employees who have joined in the same day (like Monday/Tuesday) as of Scott.

JOINS

Joins are used to fetch the data from multiple tables.

It works on cartesian product principle.

In cartesian product, each and every element of Set-1 is multiplied with each & every element of Set-2

* when we use joins each & every record of 1st table will be joined or combined with each & every record of another table.

A = { 1, 2, 3 }

B = { 4, 5 }

A * B = { (1,4), (1,5), (2,4), (2,5), (3,4), (3,5) }

* In joins each and every record of the first table is combined with each & every record of the 2nd table.

Types of Joins

1. Cartesian / Cross Join
2. Equi / Inner join
3. Non - Equi Join
4. Outer Join → Left Outer Join / Left Join
   → Right Outer Join / Right Join
   → Full Outer Join / Full Join
5. Natural Join
6. Self Join

(1) CARTESIAN / CROSS JOIN

→ It returns total possible combinations of the records. (No. of records in 1st table * No. of records in 2nd table)

→ Generally it will not have a where clause. (If it will have incorrect condition)

→ We should not use cartesian join in real time.

→ Cartesian Join might be useful to get the test data as shown below. Create table table1 as select A.id, B.name from A, B;

Cartesian Join = No. of records in A X No. of records in B where A & B are tables.

Working of Cartesian Join

Emp table

EmpNo	Ename	DeptNo
101	A	10
102	B	20
103	C	10
104	D	20
105	E	30

(5)

Dept table

DeptNo	DName
10	Accounts
20	Research
30	Sales
40	Operations

(4)

O/P: 5 x 4 = 20 records will be displayed

O/P: Display EmpName & their dept name using cross join

```
Select A.Ename, B.DName from Emp A, dept B;
```

(or)

```
Select Emp.Ename, dept.Dname from Emp, dept;
```

O/P:

A.Ename	B.dname
A	Accounts
A	Research
A	Sales
A	Operations
B	Accounts
B	Research
B	Sales
B	Oper
C	Accounts
C	Research
C	Sales
C	Oper
D	Accounts
D	Research
D	Sales
D	Oper
E	Accounts
E	Research
E	Sales
E	Oper

(*) Required O/p

A.Ename	B.dname
A	Accounts
B	Research
C	Accounts
D	Research
E	Sales

(**) This problem can be overcome by providing a condition that condition is called Join Condition, then that query is called as an Inner Join

(***) CJ contains both correct & incorrect set of data, we have to eliminate the incorrect set of data & retain the correct set of data. This can be done using INNER JOINS

② INNER JOINS

It returns the matching records b/w the tables, it will have a join condition in it.

Ex:

① Display the Ename & their dname

Select A.Ename, B.dname from Emp A, dept B
where A.deptno = B.deptno; → Join Condition (Common columns are Joined)

If a table A is having 10 records & table B is having No records then cartesian product result in (0) zero records. Create table table1 as Select A.id, B.name from A,B;
Select * from A,B; → Results in zero records

O/p: 10x0 = 0

NOTE:
① To write an inner joins we should have a common columns b/w the table.
② Inner Join will avoid Cartesian product and fetches the correct no. of records.
③ If we don't have common columns b/w the tables, then we cannot write Inner Joins

② Display Ename, dname, Sal, Location for all employees earning more than 2000.

→ Select A.Ename, B.dname, A.Sal, B.loc
from Emp A, dept B → table alias
where A.deptno = B.deptno → Join condition
And A.Sal > 2000; → Filter condition

O/p: Returns only matching records

Alias helps to provide short names for bigger tablenames and also helps in better performance

ANSI Style Inner Joins

Select A.Ename, B.dname
from Emp A Join dept B
On (A.deptno = B.deptno) → JOIN condition
And A.Sal > 2000; → Filter condition

ASSIGNMENT

① Display Emprame, dname, sal, Job for all the employees located in New yorks Earning less than 2000

Select A.Ename, B.dname, A.Sal, A.Job from Emp A, dept B
Where A.deptno = B.deptno
And B.loc = 'New yorks'
And A.Sal < 2000;

② Display all the employees from Research dept (Sub query)

Select * from Emp
where deptno = (Select deptno from dept
where dname = 'Research');

③ Display Ename, dname who is earning the first highest Salary.

Select A.Ename, B.dname from Emp A, dept B
where A.deptno = B.deptno
And (A.Empno, A.deptno, A.Sal) IN
(Select Empno, deptno, Max(Sal) from Emp
Group by Empno, deptno);

OR

Select A.Ename, B.dname from Emp A, dept B
where A.deptno = B.deptno
And A.Sal = (Select Max(Sal) from Emp);

④ Display the dept+name & their total Sal (ANSI Style also)

```
Select B.dname, Sum(sal) from Emp A, dept B
where A.deptno = B.deptno
Group by B.dname;
```

OR

```
Select B.dname, Sum(A.sal) from Emp A Join dept B
on (A.deptno = B.deptno)
Group by B.dname;
```

⑤ Display dname wise no. of emp in a table

```
Select B.dname, Count(A.empno) from Emp A, dept B
where A.deptno = B.deptno
Group by B.dname;
```

INNER JOIN SCENARIOS

Scenario #1

A	B	C
X Y Z	X P Q	X U V

o/p: Y | P | U

A	B	C
1	1	1
2	3	3
	5	6

Matches across all 3 tables A=B=C

Common column = X

```
Select A.Y, B.P, C.U from A, B, C
where A.X = B.X
AND B.X = C.X; => A.X = C.X

A.X = B.X = C.X => wrong
```

ANSI STYLE

```
Select A.Y, B.P, C.U
From A Join B on (A.X = B.X)
     Join C on (A.X = C.X); => B.X = C.X
```

Scenario #2

A	B	C
X Y Z	X Y P	X U V

o/p: Z | P | U

```
Select A.Z, B.P, C.U From A,B,C
where A.X = B.X
And A.Y = B.Y
And A.X = C.X => B.X = C.X
```

ANSI STYLE

```
Select A.Z, B.P, C.U
From A Join B On (A.X = B.X And A.Y = B.Y)
     Join C On (B.X = C.X);
```

Scenario #3

A	B	C	D
X Y Z	P Q R	R S T	U Y S

o/p: Z | Q

B=C, C=D, D=A. SO A=B

B → C → D → A
A → D → C → B

```
Select A.Z, B.Q From A,B,C,D
where A.Y = D.Y
And C.R = B.R
And D.S = C.S;
```

[No common column b/w A & B So we make use of C & D to fetch data]

ANSI STYLE

```
Select A.Z, B.Q
From A Join D on (A.Y = D.Y)
     Join C On (C.S = D.S)
     Join B On (B.R = C.R);
```

ANSI STYLE

```
Select A.Z, B.Q
From A Join D On (A.Y = D.Y)
     Join B On (B.R = C.R)
     Join C On (C.S = D.S);
```

Since
⇒ B. Can't access tables below to it

Sequence should be ⇒ A → D → C → B

NOTE: ① Number of joins equals to Number of common columns

② Minimum No. of Joins = No. of tables − 1

3. NATURAL JOIN

It is similler to inner join but we don't provide a Join condition explicitly, it is formed implicitly by Oracle
Here, Join condition is formed implicitly by Oracle. but the common column names must be same.
If the common column names are not same & if we write Natural Join, it will not through any error but results in Cartesian Join (or) Cross Join.

```
Select A.Gname, B.Dname
From Grp A Natural Join Dept B;
```

Natural Join	Inner Join
① Here, Join condition is formed Implicitly	① we have to provide the Join condition Explicitly
② Can be written only in ANSI Style	② Can be written in ANSI Style as well as in Traditional Style
③ The common columns names must be same, otherwise results in Cartesian Join	③ The common column names may (or) may not be same.

Scenario #1

A			B			o/p:	Z	Q
X	Y	Z	X	Y	Q			

```
Select A.Z, B.Q
From A Natural Join B;
```

NOTE: Both X & Y are used in the Join condition Implicitly.
[Where A.X = B.X And A.Y = B.Y]

Scenario #2

To use only one common column b/w the tables, we have to use "USING" clause. In this case we shouldn't use NATURAL keyword, but just use JOIN keyword.

```
Select A.Z, B.Q
From A Join B Using(X);
```

⇒ Now only X is used in the JOIN condition.

Scenario #3

To use Natural Join b/w more than 2 tables

```
Select * From Table1
Natural Join Table2
Natural Join Table3;
```

4. OUTER JOIN

It returns both matching & Non matching records

Outer Join = Inner Join + Non Matching records

Non-matching means data present in one table but absent in another.

TYPES OF OUTER JOIN

① Left Join : Matching + Non-matching records from L.H.S table

② Right Join : Matching + Non-matching records from R.H.S table

③ Full Join : Matching + Non-matching records from both tables

NOTE

① If 2 tables are related, Master table contains the Matching records as well as Non-matching records, whereas Child table contains only Matching records.

② If 2 tables are not related, then the Non-matching records would exist in both the tables, in that case we can make use of FULL JOIN

① Display all the dept names whether it is having Employee or Not? If it is having Employee then display Employee name.

```
Select A.Ename , B.DName
From Emp A Right Join Dept B      → Non-matching records (operations)
On A.deptno = B.deptno;
```

```
Select A.Ename , B.DName
From Dept B Left Join Emp A       → Non-matching records
On A.deptno = B.deptno;
```

```
Select A.Ename , B.DName
From Dept B Full Join Emp A
On A.deptno = B.deptno;
```

A — 5 records, B — 9 records ⇒ If A has 5 records & B has 9 records

2 = 2 → Matching

3 != , != 7 → Non-matching

A CJ B ⇒ 45

A IJ B ⇒ 2 A∩B

A LJ B ⇒ 2+3=5

A RJ B ⇒ 2+7=9

A FJ B ⇒ 2+3+7=12 A∪B

ASSIGNMENT

① Modify the previous query to display, No Employees against operations.

O/P :- No Employees Operations

② Write a query to get Full Join without using full Join Keyword (LJ union RJ)

③ Display only Non-matching records [Minus]

Methods
- ① Minus with Outer Join
- ② Not In
- ③ Intersect
- ④ Not Exists

④ Prod_New

Pid	Pname
101	Camera
102	CC
103	LT
104	DT
105	Ipad

Ord_New

Ordid	Pid	Qty
9001	101	2
9002	101	3
9003	102	5
9004	105	6
9005	108	2

→ write all types of Joins in ANSI (CJ, IJ, NJ, LJ, RJ, FJ) & analyze the o/p. (display PName & Order Id

A — 5, B — 5

3=4
2!=
!=1

CJ = 5×5 = 25
IJ = ④ - 9001, 9002, 9003, 9004
NJ = ④ - 9001, 9002, 9003, 9004
LJ = ⑥ - 9001, 9002, 9003, 9004, 103, 104
RJ = ⑤ = 9001, 9002, 9003, 9004, 9005
FJ = ⑦ - 9001, 9002, 9003, 9004, 9005, 103, 104

ANSWERS ① Select B.dname, NVL(To_char(A.Ename), 'No Employee')
From Dept B Right Join Emp A
On (A.deptno = B.deptno);

← compare

① Select NVL(A.Ename, "No Employee"), B.Dname
From Emp A Right Join Dept B
On (A.deptno = B.deptno);

② Select A.Pid, B.Pid From A Left Join B
On (A.Pid = B.Pid)
UNION
Select A.Pid, B.Pid From A Right Join B
On (A.Pid = B.Pid)

Table A → Matching + Nonmatching Records
+
Table B → Matching + Nonmatching Records

③ Select A.Pid, B.Pid
From A Full Join B On A.Pid = B.Pid
MINUS
Select A.Pid, B.Pid
From A Join B On A.Pid = B.Pid

④ Cartesian Join

Select A.Pname, B.Ordid
From Prod-new A ~~Cross Join~~ Ord-New B;
Select A.Pname, B.Ordid
From Prod-new A CROSS JOIN Ord-New B;

Inner Join

Select A.Pname, B.Ordid
From Prod-New A Join Ord-New B
On A.Pid = B.Pid;

NATURAL JOIN

Select A.Pname, B.Ordid
From Prod-New A Natural Join Ord-New B;

Left Join

Select A.Pname, B.Ordid
From Prod-New A LEFT JOIN Ord-New B
On A.Pid = B.Pid;

RIGHT JOIN

Select A.Pname, B.Ordid
From Prod-New A Right Join Ord-New B
On A.Pid = B.Pid;

FULL JOIN:

```
Select A.Pname, B.Ordid
From Prod_New A Full Join Ord_New B
On A.Pid = B.Pid;
```

Outer Join without ANSI Style

Outer Join should be represented with (+)

```
Select A.Ename, B.Dname
From Emp A, Dept B
where A.deptno (+) = B.deptno  ⇒ Right Join
```

```
Select A.Ename, B.Dname
From Emp A, Dept B
where B.deptno = A.deptno (+)  ⇒ Left Join
```

Full Join Restrictions:

① where B.deptno (+) = A.deptno (+)
↳ Can't use (+) on both sides

② where B.deptno = A.deptno (+)
OR A.deptno = B.deptno (+)
↳ Can't use (+) symbol in OR Operator

Full Join can be achieved using UNION Operator as shown

```
Select A.Ename, B.Dname From Emp A, Dept B
Where A.deptno (+) = B.deptno ⇒ Right Join
UNION
Select A.Ename, B.Dname From Emp A, Dept B
Where A.deptno = B.deptno (+) ⇒ Left Join
```

Q. Display all Dname along with their total salaries if a dept. doesn't have a emp, display his total salary as "0"?

```
Select A.deptname, Sum(Nvl(Sal,0)) as Total_Sal
From Dept A Left Join Emp B
On A.deptno = B.deptno;
```

5 SELF JOIN

It means joining a table to itself, It is similar to an Inner join but same table will be used more than Once.

Ex: From Emp A, Emp B
(OR) From Emp A JOIN Emp B

Q1. Display Ename & their managers names?

```
⇒ Select A.Ename, B.Ename as Mgr_Name
From Emp A LEFT JOIN Emp B
On (A.Mgr = B.Empno);
```

Q2. Display the employee name & their Manager names. If an employee does not have manager display him as "CEO"

NVL (to_char(B.Ename), 'CEO')

```
⇒ Select A.Ename, Nvl(B.Ename, 'CEO') as Mgrname
From Emp A Left Join Emp B
On A.Mgr = B.Empno;
```

```
⇒ Select A.Ename as Emp1, B.Ename Emp2
From Emp A JOIN Emp B
where A.Sal = B.Sal;
```

where A.Sal = B.Sal
And A.Empno != B.Empno;

ASSIGNMENT:

1. Display the employee earning same salary
2. Display the employee working in same dept, as of SCOTT but SCOTT should not be displayed.
3. Display the employees earning more than reporting manager.
4. Display the employees who have joined before their reporting manager.
5. Display the employees who are earning same salary as of SCOTT

⑥ NON EQUI JOIN

→ It is a JOIN which is joined without equal Symbol.

→ This join can be used even though we do not have a common column, This type of Join is very close to CJ, that's why it should be written very carefully.

Emp

Empno	Ename	Sal
101	Scott	2000
102	Ford	5000
103	Miller	9000
104	John	2000
105	Adams	1000

Tax

Min_Sal	Max_Sal	Tax
0	2500	10
2501	5000	20
5501	1000	30

Q. Display the ename & his tax percentage

Ename	Tax
Scott	10
Ford	20
Miller	30
John	10
Adams	10

⇒ Select Distinct A.Ename, B.Tax
From Emp A JOIN Tax B
On (A.Sal between B.Min_Sal And B.Max_Sal)

⇒ Select A.* From Emp A JOIN Emp B
On (A.deptno = B.deptno)
And (B.Ename = 'SCOTT');

⇒ Select A.Ename, B.deptno From Emp A, Emp B
where A.Empno = B.Empno
And A.deptno = (Select deptno from Emp
where Ename = 'SCOTT');

⇒ Select A.Ename, B.Ename as Mgrname, A.Hiredate as Emphdt, B.Hiredate as Mgrdt
From Emp A, Emp B
where A.Mgr = B.Empno
And A.Hiredate < B.Hiredate;

⇒ Select A.Ename, B.Sal From Emp A, Emp B
where A.Empno = B.Empno
And A.Sal = (Select Sal From Emp
Group by Sal
having count(*) > 1);

⇒ Select Max (Distinct (A.Sal)) From Emp A
where (n-1) = (Select Count (Distinct (B.Sal)) From Emp B
where B.Sal > A.Sal);

ADVANCED SQL

ORACLE SCHEMA ARCHITECTURE:

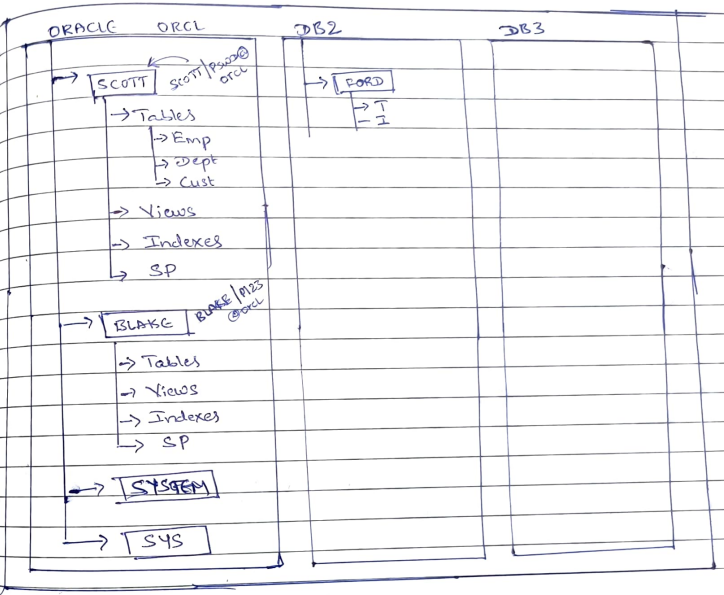
SCHEMA: means USER or an ACCOUNT or OWNER

* A Schema is a logical connection of certain Database objects like tables, Views, Indexes, Sequences, SP (Stored procedures), Function, Triggers etc.
* A Schema is always associated with a password
* A Database is organized into many Schemas which inturn can connect many DB objects.

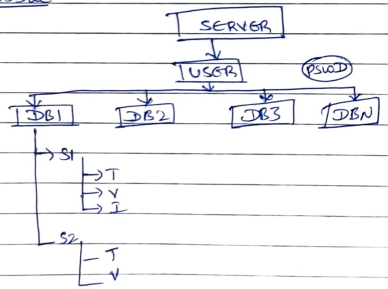
DEFAULT USER

* When we create the database the following 2 Users (default) will be created
  1. System - Password Manager (tiger) - passwords
  2. Sys - No Password set (Super User (Admin))

* Sys is more powerful than System, owns the data dictionary
* These 2 users are Admin users [To perform Administrator Tasks]
* They cannot be dropped or removed from the database



MSSQL



- A DB will have many Schemas,
- DB Server can have many db which inturn can have many users & user can have many db objects

CREATING NEW USER

Connect as System/Manager

SQL> Conn System/Manager or System/tiger

STEP 1: Create User User_name identified by PSWD;

SQL> User created

STEP 2: Grant Connect, resource to User_name;

Ex: Create User King identified by K123;

Grant Connect, resource to King;

To change the Password

System Admin can change the password of user (or) to change the password (or) Resetting password

SQL> Conn System/Manager
Connected

SQL> Alter user King identified by KK;
User altered

If we want to lock User

SQL> Alter User King account Lock;
User altered

If we want to unlock User

SQL> Alter user King account unlock;
User altered

If we want to Drop user → Drop user King;

SQL> Drop user King Cascade;
User dropped

If want to login without PSWD (Forgot password)

Enter Username: /as Sysdba
then it is connected

SQL> Show User;
User is "Sys"

SQL> password
SQL> Conn Sys as Sysdba
SQL> Conn /as Sysdba
Sys as Sysdba

PRIVILEGES & ROLES & DCL

PRIVILEGE: is a permission to perform certain DB activity (or) tasks

Types:
1. System privilege (Given by DBA)
2. Object privilege

System privilege is given by administrator to perform certain DB tasks (Create or Modify certain database objects)
These are generally provided by the Super User
Ex: Create table
Create View — Drop table
Create User — Drop View
Create Procedure — - - -
Create Sequence

Object privilege is given to particular DB objects like Table, View, Function, SP, etc.,
- Generally provided on a particular object to perform Select & DML operation
- Only owner of the object can provide privileges to other users

Ex: SELECT
INSERT
UPDATE
DELETE
ALL — [Includes all the privileges]
EXECUTE [Applicable for SP, Functions]
ALTER

DCL Commands [DATA CONTROL LANGUAGE]

These commands help to provide or Revoke permission to a user or a role.

GRANT: It Grants or Provides a privilege to a user

REVOKE: It takes back privilege from user

* To Grant a Privilege we should be owner of the object or should be Superuser
* Grant & Revoke will implicitly commit the DML changes above them (not DDL)

Ex:

USER1 [SCOTT]

SQL > Grant Select on Emp to FORD;

USER2 [FORD]

SQL > Select * from Scott.Emp; ⇒ works

NOTE: [Username.TableName on granted table]

SQL > Select * from Emp; ⇒ Error

SQL > Delete from Scott.Emp; ⇒ Error (Insufficient privileges)

USER1 [SCOTT]

SQL > Revoke Select on Emp from FORD;

SQL > Select * from Scott.Emp; → Error [As SCOTT is having Emp table no need to mention]

USER2 [FORD]

SQL > Select * from Scott.Emp; ⇒ Error

SQL > Select * from Emp; ⇒ Error

USER1 [SCOTT]

SQL > Grant all On Emp to FORD;

* Multiple Grants to multiple users at once, but on same object

Grant delete, Insert On Emp to FORD, USER3;

** Not possible to provide on multiple objects

Grant insert on Emp, delete on dept to Scott → Error

From System

SQL > Grant Create table, Create View to User2;

SQL > Revoke Create table From User2;

ROLES

* A Role is group of privileges which can be Granted or Revoke to multiple users at once.
* A Role can contain both System & Object privileges
* Any change (Add/Remove) in privilege to the Roles will automatically effect all the users who have Granted with the Role.
* It Resolves the Effort of the DBA to Grant & Revoke the privileges to multiple users.

① Creating a Role

Create Role test_Role;

② Granting a Role

Grant create table, Create View to test_Role;

③ Grant the Role to Users

Grant test_Role to User1, User2;

Revoke Create View from Test_Role;

DEFAULT ROLES

- CONNECT → Permits to Connect to DB
- RESOURCE → Helps to create basic db objects like Tables/Synonyms/etc.,
- DBA → As good as Sys user (He will have all the System privileges)

SQL > Grant dba to Scott;

A role can contain both System & Object privileges

PSEUDO - COLUMN

→ They are also called as virtual columns or FALSE columns
→ They appears to be non-existing in any of the table, but they are actually parts of all the tables. (They Exist)

Ex: Select Ename, Rownum, Rowid from Emp;

Ename	Rownum	Rowid
King	1	AAANNAA
Blake	2	AAANNAB

ROWID [Row Identifier]

✓ Is an unique address for a row in the entire database
Eg: It acts like a pointer to a Row
✓ It is a 18 digit Alpha Numeric Value

=> Rowid will not change whenever a record is getting updated.
=> Rowid will be Removed only when a record is deleted. (Static)
=> Whenever a new record is inserted to a table, the Rowid is automatically Generated & Stored.

Architecture of ROWID

Objectid (6)	Fileid (3)	Blockid (6)	Slotid (3)
AAAAAB	AAE	AAAAEA	AAA

* DB can have more than One database file
* A File can have many blocks
* A Block can have many slots

Eg: 7788 Scott 300 AAABB1AA
7789 Blake 500 AAABB1AB

* In a given table first row will have Lowest Rowid and Last row will have highest Rowid (arranged)
* Data in the table is ordered according to the Rowid

Display the 1st Record

Select * from Emp
where Rowid = (Select Min(Rowid) from Emp);

To pick the Last row from table (Last record)

Select * from Emp
where Rowid = (Select Max(Rowid) from Emp);

Assignments

① Display the 2nd Row
② Display First & Last row in a same query.

ROWNUM [Row Number]

→ is a sequential no. generated as and when a record is retrieved (given) from the table.

→ ROWID	ROWNUM
① It is a static value	① It is a dynamic value
② Rowid will not change	② Rownum will change
③ It is stored in the database	③ Not stored in database
④ Returns a Alphanumeric value	④ Returns the Integer value

```
Select Ename, Rowid, Rownum from Emp;
```

Ename	Rowid	Rownum
King	AAAAYNAAABIERD	1
Blake	AAAYNJBSCIERDK	2

* Is not stored in the db, it's generated on the fly (Run time)

Restrictions

① When we fetch the data based on Rownum. The 1st Record will always be fetched.

* ROWNUM is used for fetching n records from the table.

```
Select * from Emp where Rownum = 2;   => No records
Select * from Emp where Rownum > 1;   => No records
Select * from Emp where Rownum <= 2;  => Works
Select * from Emp where Rownum = 1;   => Works
```

Display 2nd record only

```
Select * from Emp where Rownum <= 2
MINUS
Select * from Emp where Rownum = 1;
```

Display any One record from dept 30

```
Select Ename from Emp where deptno = 30 AND Rownum = 1;
```

Assignment

① Display 6th record from table
② Display 2nd & 4th record from table (Minus & Union)

① Display any one employee who is getting 3000 Rs.

```
Select * from Emp
  where Sal = 3000;
```

Olp: We will get multiple records, if we want to display One record

```
Select * from Emp
  where Sal = 3000 AND Rownum = 1;
```

③ Display first 5 records

```
Select * from Emp
where Rownum <= 5;
```

④ Display last 5 Records

```
Select * from Emp
MINUS
Select * from Emp
where Rownum <= (Select count(*) - 5 from Emp);
```

⑤ Delete from Emp where Rownum = 1;
⑥ Delete from Emp where Rownum <= 5;

ASSIGNMENT

③ Write 5 differences b/w Rowid & Rownum
④ Display the no. of records without using count function
⑤ Display alternate records from the table (Rowid, Rownum, MOD function with paired columns in Sub-queries)
⑥ Display 2nd record from the table (Using ROWID)
⑦ Display Last record from the table (Using Rowid)