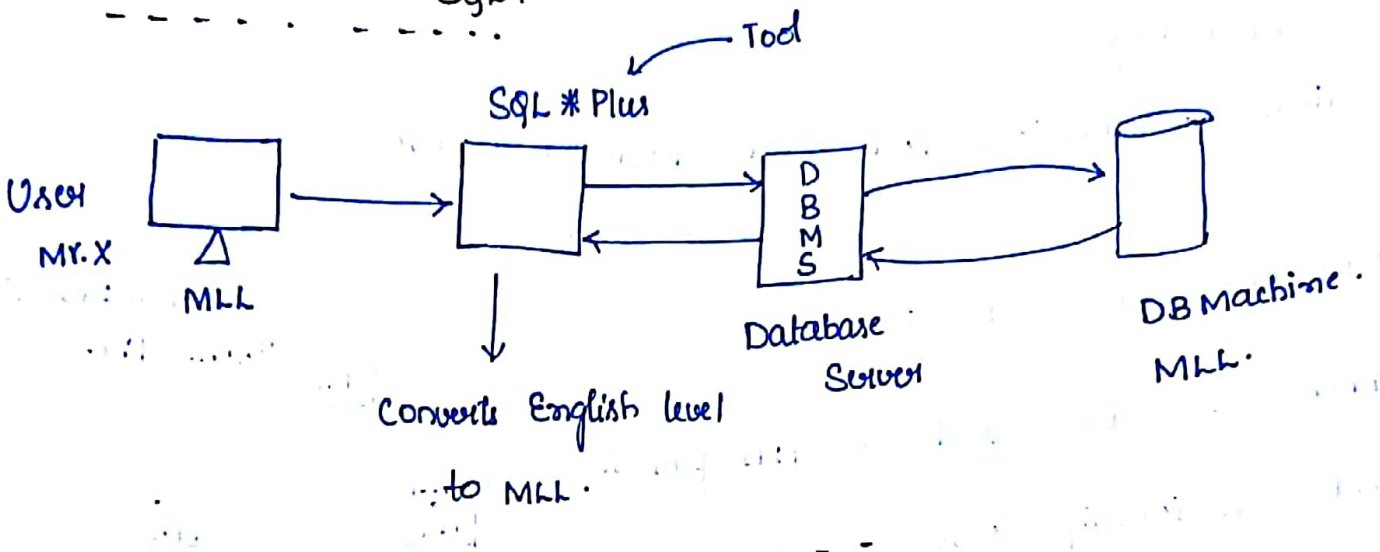
18th AUG 2017

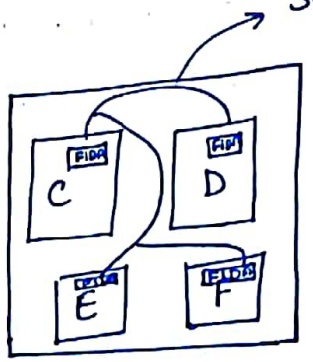
** Overview Of SQL:



** Types Of DBMS:

* Flat-File Model:

→ In Flat-file Model, the data is stored in the form of file format.



Same file can be placed in different storage location.

* Advantages:

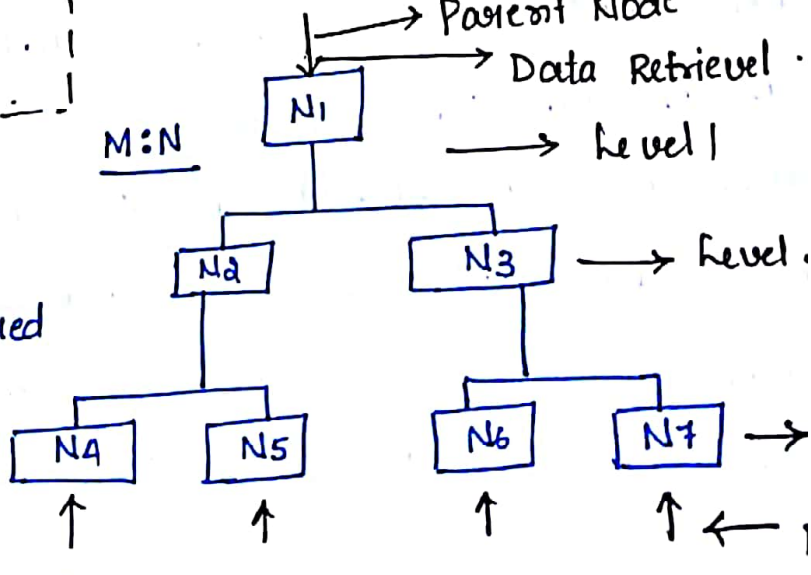
→ The disadvantage of Flat-file Model is Data Redundancy (dupli

data or copy of same data.

* Hierarchical - DBMS

→ In HDBMS, the data is stored in the form of file format

also called as M:N



* Advantages:

→ Overcoming the problem of Data Redundency.

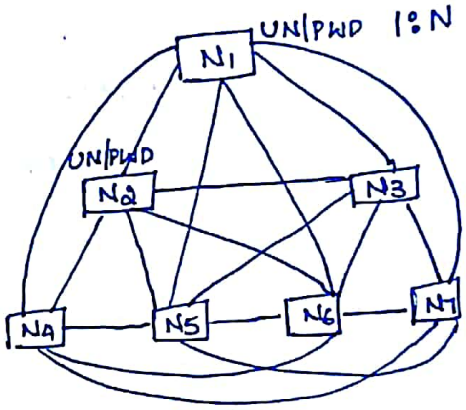
* Disadvantages:

→ Time consumption for data retrieval is more.

---

* NETWORK - DBMS (N-DBMS)

1 Parent : N child



→ NDBMS is also called as 1:N parent child relationship (one parent - many child relationship).

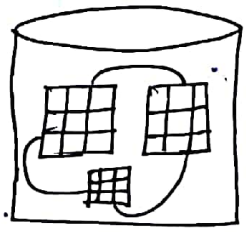
→ Data is stored in file format.

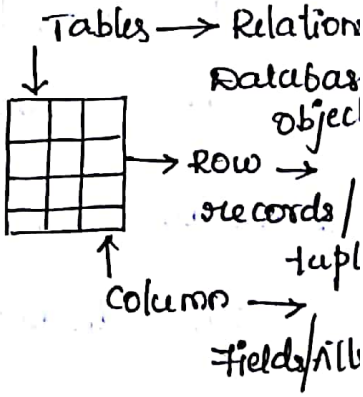
→ Time consumption for data retrieval is less.

→ The disadvantage is complexity of the network.

---

* Relational - DBMS (R-DBMS)





→ In RDBMS, data is stored in the form of 2-dimensional rows & columns.

→ It is the advanced database management system.

→ In RDBMS, data can be fetched either referring the columns or records of table or relations.

* Tables are also called as Relations / Database objects.

* Rows are also called as Records / Tuples.

* Columns are also called as Fields / Attributes.

→ The Intersection of rows & columns is called as Cells, which contains d

---

** What is Data ?

→ A useful information is a data.

→ Small set of information becomes data, this set of information helps to make decision.

→ For example your name, age, height, weight etc are some data related to you. A picture, image, file, pdf etc can also be considered as data.

** What is Database ?

→ A Database is a systematic collection of usefull information

→ Database supports storage and manipulation of data.

→ Database can also be defined as collection of one or more tables. Ex: Mobile, Human brain etc.

→ Database make data management easy.

** What is Database Management System ?

DBMS is a program that stores, retrieves and modi data in the database on request. It is also called as stud of different techniques of design, techniques development & mainten of the database.

21 Aug 2017

** RELATIONAL DATABASE CONCEPT

→ It is collection of relations or two dimensional tables.

→ Dr. EF codd proposed the relational model for database system 19

Relational Model Consists of the following

* Collection of objects or Relations
* Set of operators to act on the relations.
* Data integrity for accuracy and consistency.

(Data Integrity: It means which maintains the correct data in the database).

To restrict storing wrong data in database we use Data integ

✓ We can Preserve data integrity by two types.

1) Data Type.
2) Constraints.

Data Type: Which ensure type of data which you store in each column.

CONSTRAINTS: These are the restrictions or conditions that are used on the columns of table to preserve the data correctness.

TYPES: NOT NULL (NOT BLANK)
UNIQUE
PRIMARY KEY
FOREIGN KEY
CHECK.

NULL → is undefined or unknown value. It is used to [illegible] empty values.

1) NOT NULL: This ensures that atleast some value should be present for an attribute, can have more than one not null constraints on a table.

Ex: Name cannot be left blank / NULL in a table.

2) UNIQUE: It checks for duplicate values.

Ex: EMPLOYEEID / MOBILE # / MAILID should / could be unique & NOT BLANK / NULL.

PRIMARY KEY: Is used for identifying a record uniquely in a table. It is the combination of NOTNULL & UNIQUE constrain You can have only one Primary Key in a per table.

Ex: Employee ID or Mobile / Mail ID can be chosen as Primary Key.

→ The columns that are eligible to become PK are calle Candidate Keys.

→ A column which is eligible to become PK but not choo as PK is called Alternate Key.

CK = PK + AK

AK = CK − PK.

4) CHECK: Check is used for enforcing some additional condition with respect to business requirement.

Ex: SAL > 3000, AGE < 14.

5) FOREIGN KEY: It is a referential integrity constraint which creates the relationship between the tables. To create a FK in a child table, master table should have PK or the common column.

→ Foreign key can take both NULL and duplicate values. There can be more than one FK per table.

Ex: Deptno from DEPT table to Deptno in Employee table.

PRIMARY KEY	FOREIGN KEY
→ It is unique & Not Null	→ It is non-unique & also can contain null values.
→ In a relation, we can have only one primary key.	→ In a relation, we can have more than one F.K.
→ Primary key may or may not refer Foreign key.	→ Foreign key always has to refer primary key.
→ It is used to identify each record uniquely in a table.	→ It is used to preserve referential integrity constraint.

→ In a relationship always a pri - mary key table is called as Master table

→ In a relationship always Foreign key table is called as child table.

** SQL:

→ Sql is developed by group IBM

→ SQL → Structured Query Language. The first name of Sql is SEQUEL (Simple or Structured english Query language).

→ SQL → It is the one and only language which is used to interact with the database.

→ TO work on SQL, a DB software (RDBMS) is required. SQL is not case sensitive.

What is SQL?

Structured Query language (SQL) pronounced as "S-Q-L" or sometimes as "See-Quel" is actually the standard langua for dealing with Relational Database. SQL can be effectively used to insert, search, update, delete database records.

** SQL SUB-LANGUAGES:

1. Data Query Language : Select.

2) Data Definition Language (DDL) → CREATE, ALTER, DROP, RENAME, TRUNCATE.

3) Data Manipulation Language (DML) → INSERT, UPDATE, DELETE.

4) Transaction Control Language (TCL) → COMMIT, ROLLBACK, SAVEPOINT.

5) Data Control language (DCL) → GRANT, REVOKE

** COMMANDS IN SQL

1) clear screen or cl scr → It clears the screen of SQL*+ tool.

2) describe or desc → It is used to display the structure of the table (ie structure means column names, datatypes applied on each column & only the NOT NULL constraints).

Ex: SQL > describe emp;

Name	Null ?	Type
EMPNO	NOT NULL	NUMBER(4)

ENAME	VARCHAR2(10)
JOB	VARCHAR2(9)
MGR	NUMBER(4)
HIREDATE	DATE
SAL	NUMBER(7,2)
COMM	NUMBER(7,2)
DEPTNO	NUMBER(2)

3) Show ⟶ It is used to view the default settings of SQL * plus tool.

Ex: SQL> Show pages
pagesize 200

SQL> Show linesize
linesize 200

SQL> Show user
User is "SCOTT".

4) set ⟶ It is used to change the default settings of SQL * plus

Ex: SQL> Set pages 300 lines 300
SQL> Show lines
linesize 300

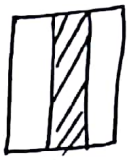
5) ; ⟶ It is SQL statement terminator, which says the end of SQL statement.

** DATA QUERY LANGUAGE :

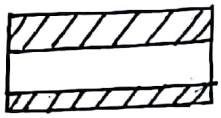
SELECT ⟶ It is used to read the data from the database.

There are 3 capabilities of SELECT class.

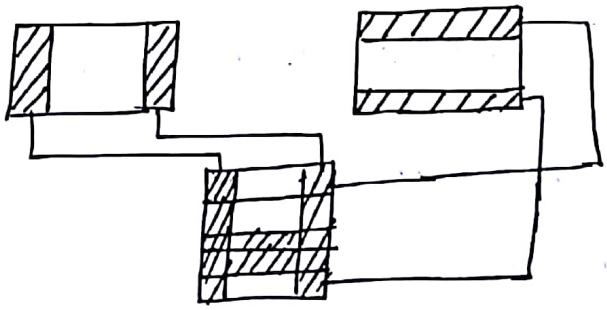
1) PROJECTION : Projection means selecting only the columns of tabel.



2) SELECTION : Selection means selecting only the rows of tabel.



3) JOINS : It is a condition used to display rows & columns (data) from different tables.



** PROJECTION :

Syntax :

All or Everything

```
SELECT * | { Distinct [columns] | Expression [Alias] }
FROM tablename ;
```

Q) Display all the columns from the employee table.

```
SQL> Select *
     from emp;
```

O/P:

EmpNO	Ename	Job	Mgr	Hiredate	. . . .
⋮	⋮	⋮	⋮	⋮	

Q) Display EmpNO, EmpName, Job from Employee table.

```
SQL> Select empno, ename, job
     from emp;
```

O/P:

EMPNO	ENAME	JOB
7369	.....	...
⋮		

Q) Display EmpNO, DeptNO, Salary, Hiredate from emp table.

```
SQL> Select empno, deptno, sal, hiredate
     from emp;
```

O/P: EMPNO DEPTNO SAL HIREDATE.

Q) Display all the columns of emp table except empno & Salary column

```
SQL> Select ename, job, mgr, hiredate, comm, deptno
     from emp;
```

Q)

** LITERALS :

A literal can be a number, character, special character or string or date data, which can be included in the select class

→ Character, special character, string and date literals must be enclosed with 'single quote'.

→ If the literals are included in the SELECT clause, then those literals will be the column heading and will be displayed as result for each row/records once.

Ex: Select 1
from dept;

1
1
1
1
1

SQL> Select 'a'
from dept;

→ 'a' : column heading depends on how many char inside the single quote.

→ Single quote

a
a
a
a
a

SQL> Select 'a '
from emp;

'A'

24th Aug 2017

```
SQL> Select 'hello'
     from dept;
```

```
'HELL
----
hello
hello
hello
hello
```

```
SQL> Select 'hello__'
     from dept;
```

(→ Spaces)

```
'HELLO
------
hello
hello
hello
hello
```

** OPERATOR PRECEDENCE:

(+ - x %. /)

→ x^ & division takes priority over addition and subtraction.

→ ( ) → indicates highest priority.

```
SQL> Select 2*3+4
     from dept;
```

```
2*3+4
-----
   10
   10
   10
   10
```

```
SQL> Select (2+3)*3
     from dept
```

```
(2+3)*3
-------
     15
     15
     15
     15
```

** COLUMN ALIASING

→ Column Aliasing is nothing but renaming the column name which can be done in 3 ways 1) Using AS keyword b/w old column name & new column name.

Ex: old columnname AS new column name

2) Using space between old column name and new column name

old column name _ new column name

3) If the new column name contains space between the words, then it has to be enclosed with double quotes.

oldcolumnname " new_column name "

space

" " is imp.

→ It is used for the purpose of calculation

Ex: Select ename as employeename
from emp;

EMPLOYEENA
SMITH
ALLEN
⋮

Q) Display empno as employee number, job as Designation, sal as Salary from employee table.

```
Select empno as employee_number,
       job as designation,
       sal as Salary
from emp;
```

OR

```
Select empno "employee number"
       job designation
       sal Salary
from emp;
```

Q) Display empno as employee name sal as monthly salary, MGR as manager from emp table.

```
Select empno "employee name",
       sal "monthly salary",
       mgr "manager number"
from emp;
```

Q) Display Salary as monthly salary, comm as monthly commission. Hiredate as Joining Date Dept as Department Number.

```
Select  Sal "monthly salary",
        comm "Monthly commission",
        Hiredate as joining date
        from emp;
```

Q) Display employee name, employeenumber, job, monthly salary, annual salary for all the employees.

```
Select  ename, empno, job, Sal
        Sal*12 as "Annual_Sal"
        from emp;
```

Q) Display ename, job, Mgr number, Monthly salary and annual Salary along with commission of 300.

```
Select  ename, job, mgr,
        Sal as
        monthlysalary,
        Sal*12+300 as annualsalary
        from emp;
```

$1 = 1/2 + 1/2$

$1/4 + 1/4 + 1/4 + 1/4$.

Q) Display employee number, job, Hiredate, monthly salary, annual salary with quaterly commission of 428.

```
Select  empno, job, hiredate, Sal as monthly_salary
        (Sal*12)+(428*4) as annualsalary
from    emp;
```

Q) Display ename, monthly salary, monthly commission & annual salary with commission of 800 given for two quadrants.

```
Select ename, sal as monthlysalary, COMM as commission
       (Sal*12) + (800*2) as annualsalary,
from emp;
```

** CONCATINATION OPERATOR: (||)

→ It is used to concatenate columns or columns with a String.

→ It is represented by double vertical lines & it is also called as pipe operator.

Ex:
```
SQL> Select ename || job
     from emp;
```

ENAME || JOB
SmithClerk
AllenSalesman.

```
SQL> Select ename || ' ' || job
     from emp;
```

ENAME || ' ' || JOB
Smith Clerk
Allen Salesman.

```
SQL> Select ' hello ' || ename
     from emp;
```
(space before hello, space after hello)

'HELLO' || ENAME
hello Smith
hello Allen.

Q) 'ALLEN is a Salesman'. Display the string for all employees

Select ename || ' is a ' || job (space) from emp;

ENAME || 'IS A' || JOB
- - - - - - - - -
Smith is a clerk.
ALLEN is a Salesman.

Q) Display the following for all employees 'ALLEN IS A SALESMAN AND HIS SALARY IS 800'.

Select ename || ' is a ' || job || 'and his Salary is ' || Sal from emp;

Qi) Display the following string for all the employees

ALLEN IS A SALESMAN AND HIS SALARY IS 800 AND WORKING IN DEPARTMENT NO 20. Select ename || ' is a ' || JOB || ' and his Salary is ' || Sal || 'and working in department number' || DeptNo from emp;

Q2) Display all rows and all columns of employee table. Select * from emp;

Q3) Display any 2 columns of employee table. Select empno, ename from emp;

Q4) IS the following statement is correct.

Select ename, deptno from emp; True

Select *, ename from emp; False

Select ename deptno from emp; True.

Q5) What is CK AK PK FK.

Q6) Difference b/w PK & FK.

27th Aug 2017

** OPERATORS

Operators are classified into

Relational Operators ⟶ (>, <, >=, <=, =, <> or !=)

LOGICAL Operators → NOT, AND, OR (NOA)

Special operators. → IN, LIKE, BETWEEN, IS (LIBI)

Order Evaluated	operator.
1	Arithmetic operator.
2	Concatenation
3	Comparison
4	IS, Like, In
5	Between
6	NOT
7	AND
8	OR.

** SELECTION: Limiting the rows of the table.

Syntax:

```
SELECT * | { Distinct [columns] | Expression (Alias) .. }
FROM tablename
WHERE condition (?);
```

WHERE → WHERE clause restricts the number of records in the table and returns the records only for the specified condition.

→ Where clause checks condition for each and every record once.

→ If the condition is true, it returns the record.

→ If the condition is false, it discards the record and checks condition for the next record of the table.

Ex:
```
Select *
from emp,
where Deptno = 10;
```

```
Select *
from emp
where ename = 'allen';
```
(x)

→ ✓ 'ALLEN';

Note: Data inside the table is case Sensitive.

WHERE clause is used to limit the records of the table.

Q) Display all the employees whose job is SALESMAN.

```
SQL> Select *
     from emp
     where JOB = 'SALESMAN';
```

Q) Display employee no, employee name, job, sal whose manager no is 7698.

```
SQL> Select empno, ename, job, sal
     from emp
     where mgr = 7698 ;
```

Q) Display all the employees whose sal is >= (greater than or equals) to 1600.

```
SQL> Select *
     from emp
     where SAL >= 1600 ;
```

Q) Display the manager no for those employees who are working in deptno 30.

```
SQL> Select *Mgr
     from emp
     where deptno = 30 ;
```

Q) Display all the employees who joined after 22-FEB-81.

```
SQL > Select ename *
      from emp
      where HIREDATE > '22-FEB-81';
```

before <

Q) Display all the employees who doesnot belong to deptno 30.

```
SQL> Select *
     from emp
     where Deptno <> 30 ;
```

<> → equals !=

Q) Display all the employees whose commision is >= 0.

```
SQL> Select *
     from emp
     where COMM >= 0 ;
```

Q) Display all the employees whose employee no is more than 7521

```
SQL> Select *
     from emp
     where empno > 7521 ;
```

Q) Display all the employees who joined on 03-DEC-81

```
SQL> Select *
     from emp
     where HIREDATE = '03-DEC-81' ;
```

** Use of Distinct :

```
Select distinct (Deptno)
from emp ;
```

O/P

10
20
30

```
Select distinct (JOB)
from emp ;
```

** EMPLOYEE TABLE:

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

29 AUG 2017

** LOGICAL OPERATORS

* AND : If both the conditions is TRUE, it returns result.

Q) Display all the employees whose deptno is 20 & manager no is 7698.

```
Select *
from emp
where deptno = 20 AND Mgr = 7698 ;
```

0 rows selected.

Q) Display all the employees whose Job is Mgr and mgr no is 7839.

```
Select *
from emp
where Job = 'Manager' AND MGR = 7839 ;
            'MANAGER'
```

Q) Display all the employees whose sal is greater than 1600 and commision ≥ 200.

```
Select *
from emp
where Sal > 1600 AND COMM ≥ 200 ;
```

0 rows selected.

Q) Less than 1600 & commission ≥ 200

```
Select * from emp
where Sal < 1600 AND COMM ≥ 200 ;
```

Q) Display all the employees who does not belong to deptno 30 & Job is Analyst.

```
Select *
from emp
where DeptNo <> 30 AND Job = 'ANALYST';
```

Q) Display all the employees who are getting some commission & DeptNO is 30.

```
Select *
from emp
where COMM != NULL AND DeptNO = 30;
      ⇓
      >0
```

One Null is not equal to another null in Oracle

Q) Display all the employees who are working in dept no 10 & 20.

```
Select *
from emp
where Deptno = 10 AND Deptno = 20;
```

0 rows selected.

Q) Display all the employees whose mgr no is 7689 and getting some commission & working in deptno. 30.

7698

```
Select *
from emp
where MGR = 7689 AND COMM >= 0 AND DeptNO = 30;
          7698
```

Q) Display all the employees whose JOB is Analyst and Salary is >= 3000 and Mgr no is 7566.

```
Select *
from emp
where JOB = 'ANALYST' AND SAL ≥ 300 AND MGR = 7566';
```

---

* OR ⟶ If the either of the condition is true, it returns the records.

Q) Display all the employees who are working in deptno 10 or 30.

```
Select *
from emp
where deptno = 10 OR deptno = 30;
```

Q) Display all the employees who joined before 23-JAN-82 and or Salary less than 1500.

```
Select *
from emp
where HIREDATE <'23-JAN-82' OR SAL <1500';
```

Q) Display all the employees who are getting some commission or working in deptno 20.

```
Select *
from emp
where COMM ≥0 OR Deptno = 20;
```

Q) Display all the employees whose job is manager or Salary >= 2300 or working in deptno 20.

```
Select *
from emp
where JOB = 'MANAGER' OR SAL ≥ 2300 OR Deptno = 20';
```

Q) Display all the employees who are working in deptno 10, 20 or 30

```
Select *
from emp
where deptno = 10 OR deptno = 20 OR deptno = 30;
```

---

* IN → IN special operator is used to evaluate multiple values of same column.

```
Select *
from emp
where deptno IN (10, 20, 30);
```

```
Select *
from emp
where deptno IN 20 ✓
```

```
Select *
from emp
where deptno IN 10 20 30
```

This returns an error, it should enclosed with brackets if more than one value is evaluated

Q) Display all the employees whose Mgr no is 7839 or 7566 AND Salary is >= 2400.

```
Select *
from emp
where MGR IN (7839, 7566) AND SAL >= 2400;
```

Q) Display all the employees who doesnot belong to deptno 10 or 20 and Job is clerk or Salesman from the employee table.

```
Select *
from emp
where DeptNO NOT IN (10,20) AND JOB IN ('CLERK', 'SALESMAN');
```

Q) Display all the employees who joined on 3-DEC-81 or 20-FEB 81 and JOB is Analyst or Salesman.

```
Select *
from emp
where HIREDATE ('3-DEC-81', '20-FEB-81') AND JOB IN ('ANALYST'
'SALESMAN')
```

30/08/2017

Q) Display all the employees who are getting some commission & working in deptno 10 or 20 and the manager number is 7782 and 7698 from the employee table.

```
Select * from emp
where COMM>=0 AND DeptNO IN (10, 20) AND MGR = 7782 AND MGR
{7788, 7698}.
```

** Is :→ Is operator is used to evaluate null values in a column

→ It is used to evaluate null values only.

→ If Is Special operator is used it always expects 'NULL' Keyword.

Ex: 1) SQL > Select * from emp
where comm is null; } Displays the records where comm is Null.

2) SQL > Select * from emp
where comm is 100; } Error
Missing NULL Keyword.

Q) Display all the employes whose job is NOT as Manager or president & not getting COMM IN employe table.

SQL> Select * from emp
where JOB NOT IN ('MANGER', 'PRESIDENT') AND COMM IS NULL;

| >=0 OR IS NOT NULL |

Q) Display all the employees who joined from '28-SEP-81' and getting some COMM and the job is Salesman or clerk.

SQL> Select * from emp
where HIREDATE >= '28-SEP-81' AND COMM IS NOT NU
AND JOB IN ('SALESMAN', 'CLERK');

8) Display all the employees whose department no is not has 30, 60 90 and the manager number is not has 7839 and 7902 and not getting COMM from the employee table.

```
SQL> Select * from emp
     where Deptno NOT IN (30,60,90) AND MGR NOT IN (7839, 7902)
     AND COMM IS NULL;
```

9) Display all the employees whose Salary is more than 2500 and works as a ANALYST in deptno 20 or 40, without commission.

```
SQL> Select * from emp
     where SAL > 2600 AND JOB = 'ANALYST' AND DEPTNO IN
     (20, 40) AND COMM IS NULL;
```

9) Display all the employees with Salary more than 1,500. and working under 7780 or 7966 with some COMM.

```
SQL> Select * from emp
     where SAL > 1500 AND MGR IN(7780, 7966) AND
     COMM IS NOT NULL;
```

** BETWEEN: Between operator is used to evaluate range of value

Between Lowest Range Value AND Highest Range Value.
↓
Mandatory

→ If the BETWEEN Operator is used, then AND Logical operator is mandatory.

Ex: SQL> Select * from emp
where SAL BETWEEN 600 AND 3000

O/P: Some rows are selected

SQL> Select * from emp
where SAL BETWEEN 3000 AND 600

O/P: NO rows selected.

Q) Display all the employees whose salary in the range of 2450 and 6000 and not getting commission or the manager no is not as 7698 or 7566.

SQL> Select * from emp.
where SAL BETWEEN 2450 AND 6000
AND COMM IS NULL ~~AND~~ OR MGR NOT IN (7698, 7566);

Q) Display all the employees whose mgr no is not in the range of 7695 & 7782 and getting some commission or working in dept no 10 OR 20.

SQL> Select * from emp
where MGR NOT BETWEEN 7695 AND 7782 AND COMM IS NOT NULL OR DEPTNO IN (10,20);

8) Display all the employees whose employee number is in the range of 7566 & 7902 and salary >= 3000 and working has President or Analyst.

```
SQL> Select *
     from emp where empno BETWEEN 7566 AND 7902
       AND SAL >= 300 AND JOB IN ('PRESIDENT', 'ANALYST');
```

9) Display all the employees whose employee number is not in the range of 7698 and 7902 and Mgr no is in the range of 7670 and 7840 and Job is not has clerk or Analyst from the employee table.

```
SQL> Select * from emp
     where empno NOT BETWEEN 7698 AND 7902
       AND MGR BETWEEN 7670 AND 7840
       AND JOB NOT IN ('CLERK', 'ANALYST');
```

31st Aug-2017

** LIKE: Like operator is used for pattern matching.

% → % means 0 or more (char/number/special character/string)

_ → _ means exactly one (_____ but not string).

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

O/P

Smith
Scott.

Q) Display all the employees whose names ends with S.

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

Jones
Madams.

Q) Display all the employees whose name having I has a second character

```
Select * from emp
where ename LIKE '_I%';
```

Q) Display all the employees whose name having E has last but one character.

```
Select * from emp
where ename LIKE '%E_';
```

Q) Display all the employees whose job having Substring MAN in it.

It can be anywhere.

```
Select * from emp
where JOB LIKE '%MAN%';
```

Q) Display all the employees whose job having L has 3rd character and S has 5th character from the employee table.

```
Select * from emp
where JOB LIKE '__L_S%';
```

Q) Display all the employees whose job having E has 4th character & getting some commission & working in deptno 30 or 40.

```
Select * from emp where JOB LIKE '___E%' AND COMM IS
NOT NULL AND DEPTNO IN (30,40);
```

8) Display all the employees whose employee number is in the range 7698 & 7900 & Hiredate having substring FEB & Manager NO. is 7698 or 7782.

```
Select * from emp
where EMPNO BETWEEN 7698 AND 7900 AND HIREDATE LIKE
'%FEB%' AND MGR IN (7698, 7782);
```

9) Display all the employees whose name doesnot start with A and S and not getting COMM from emp table.

```
Select * from emp
where ename NOT LIKE 'A%' AND ename NOT LIKE 'S%'
AND COMM IS NULL';
```

10) Display all the employee whose job is NOT as SALESMAN OR CLERK and Hiredate having Substring 81. and Salary is not in the range of 1000 and 2900.

```
Select * from emp
where JOB NOT IN('SALESMAN','CLERK') AND HIREDATE LIKE '%81%'
AND SAL NOT BETWEEN 1000 AND 2900;
```

11) Display all the employees whose name having A has 3rd character and manager no not has 7878 or 7698 and Salary having 0 in the last position.

```
Select * from emp where ename LIKE '__A%' AND MGR
NO$ IN (7878, 7698) AND SAL LIKE '%0';
```

8) Display all the employees whose Salary is in the range of 800 and 2800 and job having L has 3rd character & S has 5th character & A has last but one character & getting some COMM from the employee table.

```
Select * from emp
where SAL BETWEEN 800 AND 2800 AND
      JOB LIKE '__L_S%A_' and COMM IS NOT NULL;
```

01-Sep-2017

** ORDER-BY:

Syntax:

```
#  SELECT  SELECTLIST
1  FROM    tablename
2  WHERE   condition (?)
3  ORDER BY Column Name  Sorting technique;
```

→ Order By clause is used to [Solve] Sort the data either in ascending or descending order.

→ By default, the data will be sorted in ascending order.

→ ASC: It sorts the data in Ascending order.

DESC: It sorts the data in Descending order

→ Always the Order By clause must be in the last line of the SQL statement / Query.

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

Salary column will be displayed in ascending order.

```
SQL> Select ename
     from emp
     Order by ename desc;
```

Ename column will be displayed in descending order.

Q) Display all the columns from the employee table for those employees who are working in deptno 20 or 30 and sort the salary column data in descending order.

```
Select *
from emp
where deptno IN (20, 30)
Order By Sal desc;
```

Q) Display employeename, job, salary for those employees whose job is Salesman, or manager and sort the job in descending order.

```
Select ename, job, sal
from emp
where job IN ('SALESMAN', 'MANAGER')
Order By JOB desc;
```

Q) Display all the columns of a table for those employees whose manager no is 7698, 7839 or 7566 & sort the employee no in descending order.

```
Select * from emp
where MGR IN (7698, 7839, 7566)
Order By empno desc;
```

Ex :

```
SQL> Select SAL, ENAME
     From Emp
     Order By Sal Desc, Ename;
```

SAL	ENAME
5101	KING
3000	FORD
3000	SCOTT
2975	JONES
2850	BLAKE
2450	CLERK

```
SQL> Select SAL, ENAME
     from Emp
     Order By Sal Desc,
              Ename Desc;
```

SAL	ENAME
5101	KING
3000	SCOTT
3000	FORD
2975	JONES

→ More than one column can be passed in Order By clause but always first priority will be given to the first column in Order By clause to sort the data.

→ If the duplicate values or the same values present in the first column then 2nd priority will be given to second column to sort the data

**

→ Data can be sorted using literals in the Order by clause.

```
SQL> Select JOB, SAL, COMM
     from emp
     Order by 2;
```

} Here SAL (Column) will be sorted.

```
     Order by H;
```

→ Throws error since only 3 columns are selected

```
SQL> Select *
     from emp
     Order by 8;
```

} 8th column of emp table will be sorted.

```
SQL> Select *
     from emp
     Order by 10;
```

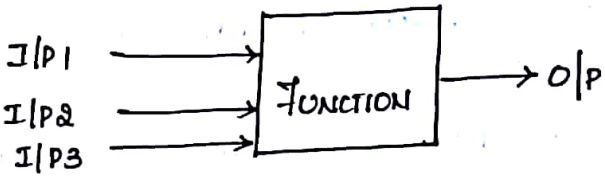
} Throws error since 10 columns are not there in emp table.

** FUNCTIONS:

Functions are reusable programs

→ In SQL, Functions are very powerful features which is used to perform the following

* It is used for calculation.
* Modifies individual data items.
* Manipulates the o/p for group of records.
* Formats Dates and Numbers for display.



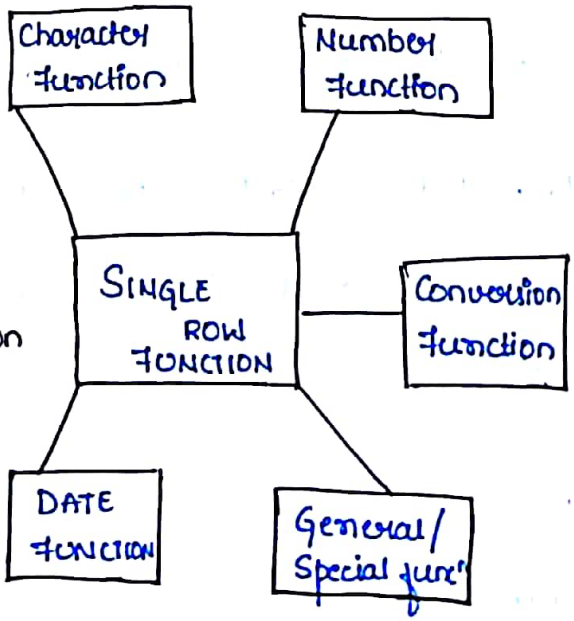
→ There are 2 types of functions. 1) Pre-defined functions.
2) User-defined functions.

→ Pre-defined functions are of two types 1) Single row function &
2) Multi row function.

** Single Row Function:

* CHARACTER FUNCTION:

- Case Manipulation Function
  - UPPER ()
  - LOWER ()
  - INITCAP ()
- Character Manipulation Function
  - LENGTH ()
  - CONCAT ()
  - REPLACE ()
  - SUBSTR ()
  - INSTR ()



** NESTED FUNCTION:

F2 (F1 (arg1, arg2) , arg2)

Result of F1 is IP to F2

argument 2 of Function 2.

→ Function inside a function is known as Nested function.

* In Nested function, always the innermost function will be executed first and the result of the innermost function will be used as an input argument for the outer function.

OH|sep|2017

** DUAL TABLE:

SQL > Select *
from dual;

D
X

Select 2*3
from dual;

2*3
6

→ An empty table.

To display only once the o/p in a row

SQL> connect hr/tiger
connected.

→ It is a dummy table used for the purpose of calculation without using user table (employee table as user table).

→ describe (Structure of a table).

SQL > desc dual;

Name	NULL?	Type
DUMMY		VARCHAR2(1)

DUAL table is maintained by Oracle cop.

** CASE MANIPULATION FUNCTIONS: → { return string datatype }

* UPPER()

→ It takes single arguments.

→ An argument can be string or columns of the table.

→ UPPER() converts the data into upper case. if the i/p argument is in the lowercase.

SQL> select upper('jspiders'), upper(ename)
from emp;

UPPER('J	UPPER(ENA
JSPIDERS	SMITH
JSPIDERS	ALLEN.

* LOWER()

→ It takes single arguments.

→ An argument can be string or columns of the table.

→ Lower() converts the data into lower case, if the i/p argument is in the uppercase.

SQL > select lower('JSPIDERS'), LOWER(ENAME)
from emp;

LOWER('J	LOWER(ENA
jspiders	Smith
jspiders	allen.

* INITCAP()

→ Initcap() displays 1st character of each argument in uppercase & rest of the argument in lower case.

Ex: SQL> Select INITCAP(ENAME)
from emp;

INITCAP(EN
Smith
Allen

SQL> Select initcap('JSPIDERS')
from emp;

INITCAP
Jspiders.

Q) Display employe name in lowercase, & first character of each job in uppercase & rest of the characters in lowercase from the employee table.

SQL> Select lower(ename), initcap(job)
from emp;

SQL > select Lower(Hiredate)
from emp;
(,Sal,)

All the data are considered as String & passed

** CHARACTER MANIPULATION FUNCTION:

* LENGTH() ← {* returns int data type}
* takes only one argument}

→ Returns length of input argument.

→ Argument can be any type of data (Ex: Sal, Hiredate, ename)

→ Ex: Select Length(ename)
from emp;

Length(ENAME)
5
5
4

Q) Display all the employees whose name having more than 4 characters.

```
Select *
from emp
where length (ename) > 4 ;
```

To execute the last query use SQL> /

Q) Display all the employees whose job having exactly 5 characters.

```
Select *
from emp
where length (Job) = 5 ;
```

** CONCAT FUNCTION : → It takes exactly/only two arguments

→ It return type is String datatype

```
SQL > Select CONCAT (ENAME, JOB)
      from emp ;
```

O/P (CONCAT (ENAME, JOB)

SMITHCLERK

ALLENSALESMAN.

```
SQL> Select concat (ename)
     from emp ;
```

```
SQL> Select concat (ename, job,
                    sal)
     from emp.
```

Throws error (Invalid argument).

** → To leave the space between the arguments of concat function we need to use Nested functions.

SQL> Select concat (Ename, ' ')
from emp;

↳ This leaves a single space

SQL> Select concat (concat (Ename, ' '), job)
from emp;

Q) Display the following string for all the employees.

ALLEN IS A SALESMAN.

SQL > Select concat (Ename 'IS A')
from emp;

o/p

EnameIs A

✓*
SQL> Select concat (concat (Ename 'Is A'), job)
from emp;

Q) Display the following for all the employees.

Hello Smith your Salary is 800.

SQL > Select concat (concat ('Hello', Ename), ' your Salary is ', SAL)
from emp;

Flow of execution:

Select concat ('Hello', Ename)
from emp;

Select concat (concat ('Hello', ename), 'your salary Is'), Sal)
from emp;

** REPLACE ()

Syntax: Replace (arg1 , arg2 , arg3)

arg1 ↑ Main string

arg2 ↑ Search string

arg3 ↑ char/num/special character/ String to be replaced with and argument data.

Ex: Select 'JSPIDERS', REPLACE ('JSPIDERS', 'S', '*')
from dual;

'JSPIDER REPLACE (

JSPIDERS J*PIDER*

Case Sensitive

SQL> Select 'JSPIDERS', REPLACE ('JSPIDERS', 'S', '*')
from dual

JSPIDERS JSPIDERS

(2nd & 3rd argument should be enclosed with single quote except numbers)**

Q) Relplace S with * in each employee name.

SQL> Select ename, Replace (ename , 'S' , '*') ;
from emp;

Q) Replace S with '*' & A with '&' with each job.

SQL> Select job, Replace (Replace (job , 'S' , '*'), 'A' , '&')) ;
from emp;

Q) Replace deptno 20 with 30 and 30 with 20 from the employee table.

```
Select deptno Replace (Replace (Replace (20, ' '), 30, 20), ' ', 30)
                                          deptno,
from emp;
```

Dept NO	R1	R2	R3
20	' '	' '	30
30	20	20	20

Q) Display the employee name and the count of number of L's present in each name.

```
SQL> Select Ename, length (ename) - Length (Replace (ename, 'L'))
     from emp;
```

05-Sep-2017

→ Replace function can also take 2 arguments. If it takes two arguments Replace function acts as Remove function.

If we pass only two arg in Replace () it acts as a Remover

Ex: L character will be removed from ename.

Q) Replace Deptno 20 with 30 and 30 with 20 and 10 with 40 from the emp table.

```
Select deptno, Replace (Replace (Replace (Replace (Deptno, 20, ' '), 30, 20), 10, 40),
                                                    ' ', 30)
from emp;
```

Q) Replace the 2nd character of each name with *

~~Select *ename Replace from emp~~

```
Select Substr (ename, 1, 1) || '*' || Substr (ename, 3, length (ename))
from emp;
```

** SUBSTR ():

** Syntax: Substr (arg1, arg2, arg3)

- arg1 → Main String
- arg2 → Position
- arg3 → No of character to print (displayed)

** SUBSTR ( )

⟶ It takes 3 arguments

⟶ Return type is String.

Example:

-8	-7	-6	-5	-4	-3	-2	-1
J	S	P	I	D	E	R	S
1	2	3	4	5	6	7	8

Always Left to Right (- or + position)

Substr ('JSPIDERS', 1, 3) = JSP

('Jspiders', 1, 1) = J

('Jspiders', 1, 5) = JSPID

('JSPIDERS', 8, 1) = S

('JSPIDERS', -1, 1) = S

('JSPIDERS', -2, 2) = RS

('JSPIDERS', -7, 4) = SPID

('JSPIDERS', -6, 3) = PID

Q) Display the first character of each employee name.

[Select * from emp where Substr]

```
Select ename, Substr (ename, 1, 1)
from emp;
```

Q) Display the last character of each employee job.

```
Select job, Substr (job, -1, 1) from emp;
```

OR

```
Select job, substr (job, length(job), 1) from emp;
```

Q) Display the last two character of each name.

```
Select ename, substr (ename, length(ename)-1, 2) from emp;
```

OR

```
Select ename, substr (ename, -2, 2) from emp;
```

Q) Display all the employees whose name has S has first character using Substr function.

```
Select * from emp where SUBSTR (ename, 1, 1) = 'S' ;
```

Q) Display all the employees whose name having character E has last but one character in each name.

```
Select * from emp
where SUBSTR (ename, -2, 1) = 'E' ;
```

Q) Display all the employees whose job having M has last but 2nd character from emp table.

```
Select * from emp
where SUBSTR (job, -3, 1) = 'M' ;
```

Q) Display the first 3 characters of each job in lowercase.

```
Select job, lower (Substr (job, 1, 3))
from emp;
```

Q) Display the first 3 characters of each job in uppercase & rest in lowercase.

```
Select concat (upper (Substr (job, 1, 3)), lower (Substr (job, 4)))
from emp;
```

Q) Display the first & last character of each name in uppercase & rest of the letters in lowercase.

```
Select concat ( concat (upper (Substr (ename, 1, 1)), lower (substr
(ename, 2, length (ename) -2))), upper (substr(ename,
-1, 1)) from emp;
```

06-Sep-2017

Q) Display the first two character & the last two character of each job in uppercase & rest of the character in lower case.

```
Select job,concat(concat(upper(job,1,2)),lower(substr(job,3,length(job)
-4))),upper(substr(job,-2,2))) from emp;
```

CLERK

** INSTR() ⟶ It takes 4 arguments.

⟶ Its return type is integer value (position of the string).

⟶ Substr() also takes two arguments. if it takes two arguments, the argument must be mainstring and position.

⟶ If it takes two arguments from the specified position, it prints rest of the characters in the mainstring.

** Syntax: INSTR(arg1, arg2, arg3, arg4)

arg1 ↑ Main String
arg2 ↑ Search String
arg3 ↑ Position
arg4 ↑ No. of occurence

```
-8 -7 -6 -5 -4 -3 -2 -1
J  S  P  I  D  E  R  S
1  2  3  4  5  6  7  8
```

If position is +ve L → R

INSTR('JSPIDERS', 'S', 1, 1) = 2

INSTR('JSPIDERS', 'S', 1, 2) = 8

INSTR('JSPIDERS', 'S', 3, 1) = 8

('JSPIDERS', 'S', 3, 2) = 0

('JSPIDERS', 'S', 2, 2) = 8

('JSPIDERS', 'S', 1, 3) = 0

('JSPIDERS', 'M', 1, 1) = 0

('JSPIDERS', 'E', 3, 1) = 6

Left to Right

If position is -ve L ← R

INSTR('JSPIDERS', 'S', -1, 1) = 8

INSTR('JSPIDERS', 'S', -1, 2) = 2

INSTR('_____', 'I', -6, 1) = 0

_____ 'P', -3, 1) = 3

_____ 'I', -5, 1) = 4

_____ 'S', -3, 1) = 2

_____ 'S', -1, 3) = 0

Right to Left

→ INSTR () search for the string in the main string and returns position of the string based on the string occurences.

→ If position is +ve then reading of the character takes place from left to right.

→ If the position is -ve then reading of the character takes place from right to left.

Q) Display the first occurence of first A in each employe name.

Select ename, instr (ename, 'A', 1, 1) from emp;

Q) Display the second occurence of A in each job.

Select job, instr (job, 'A', 1, 2) from emp;

Q) Display the last occurence of A in each job

Select job, instr (job, 'A', -1, 1) ~~length (right)~~

from emp;

Q) Display all the employes whose name having A has first character using instr function.

Select ename, where instr (ename, 'A', 1, 1)=1 from emp;

OR Select * from emp

where instr (ename, 'A', 1, 1) = 1;

Q) Display all the employees whose name having S has last character using instr().

INSTR

Select * from emp where (ename, 'S', -1, 1) = Length (Ename);

Q) Display all the employees whose name having E has last but one character using instr().

Select * from emp where Instr (Ename, 'E', -2, 1) = Length (Ename)-1;

Q) Display all the employees whose job having E has fourth character using INSTR().

Select * from emp where INSTR (Job, 'E', 4, 1) = 4;

Q) Display all the employees whose job having substring MAN in it using INSTR().

Select * from emp where INSTR (JOB, 'MAN', 1, 1) <> 0;

OR ↓ OR
>0 =1.

Q) Display all the employees whose job having substring SID in it using INSTR().

Select * from emp where INSTR (JOB, 'SID', 1, 1) > 0.

07/sep/2017

** NUMBER()

- ROUND()
- TRUNC()
- MOD()

* ROUND() → It rounds out the value to the specified integer based on decimal value.

→ If the decimal value is 0 to 4 range it rounds off value to previous integer.

→ If the decimal value is 5 to 9 range it rounds off value to next integer. Ex: 14.45 → 14
14.58 → 15

```
SQL> Select Round(14.455), Round(14.66)
     From dual;

ROUND(14.455)  ROUND(14.66)
-------------  ------------
           14            15
```

---

* TRUNC() : → It truncates a value to specified integer.

```
SQL> Select Trunc(14.466), Trunc(14.999)
     from dual;

TRUNC(14.466)  TRUNC(14.999)
-------------  -------------
           14             14
```

---

* MOD() : Mod() always returns a remainder.

```
SQL> Select Mod(13,2), Mod(13,9)
     from dual;

MOD(13,2)  MOD(13,9)
---------  ---------
         1          4
```

---

* DATE() : Oracle db Server stores date in internal numeric format. Century : Year : Month : DAY : HOUR : MIN : SEC.

* Default Date format is : DD-MM-YY

```
SQL> Select HIREDATE
     from emp
     where ename like 'A%';
```

```
HIREDATE
---------
20-FEB-81
23-MAY-87
```

---

* ARITHMETIC WITH DATES

→ Add or Subtract a date to or from a number, to get the resultant date value or result.

→ Subtract two dates to get the no. of days between two dates.

---

* SYS DATE :

→ Sys Date is a function which returns the system date and time in which Oracle DB server is installed. while returning the o/p, it returns only the date.

```
SQL> Select SYSDATE
     FROM DUAL;
```

```
SYSDATE
---------
07-Sep-17
```

```
SQL> Select SYSDATE+3
     From dual;
```

```
SYSDATE+3
---------
10-Sep-17
```

```
SQL> Select Sysdate-2
     from dual;
```

```
Sysdate-2
---------
05-Sep-17
```

```
SQL> Select Sysdate - Hiredate
     from emp;
```

```
Sysdate - Hiredate
------------------
13413.3895
```

```
SQL> Select TRUNC (Sysdate - Hiredate) from emp.
```

→ It eliminates the decimal value.

Change for every

Q) Write a query to display an experience of an employee with respect to days.

```
SQL> Select Trunc (Sysdate - Hiredate)
     from emp;
```

O/P
13078
13011

Q) Write a query to display an experience of employee in terms of years.

```
SQL> Select Trunc((Sysdate - Hiredate)/365) from emp;
```

O/P
36
36

Q) Display all the employees who are more than 35 year old from an employee table.

```
Select * from emp where Trunc ((Sysdate - Hiredate)/365) > 35;
```

**
Q) Display all the employees who are having 30 years of experience from emp table.

```
Select * from emp where Trunc ((Sysdate - Hiredate)/365) = 30;
```

---

* SYSTIME STAMP : Systime Stamp() returns System date and time in hrs, minutes, seconds, milliseconds & including the time zone as well.

```
SQL> Select Systimestamp from dual;
```

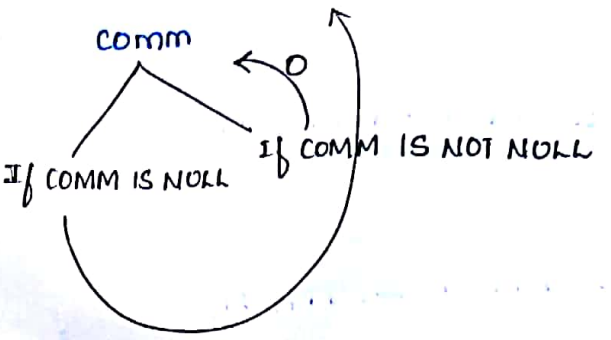
O/P
07-Sep-17 09:34.39.140000 AM +5.30

** GENERAL / SPECIAL ()

→ These functions works with any datatype and pertain (applicable) to using null values.

Anything added with Null it returns Null.

* NVL() → Null value() NVL (arg1, arg2)



→ NVL () is used to convert Null value to an actual value.

SQL> Select comm, NVL(COMM, 0)
from emp;

COMM	NVL (COMM, 0)
	0
300	300
500	500
	0
1400	1400
	0

→ The data type must match

SQL> Select COMM, NVL (COMM, 'A')
from emp;

Q) Display the total monthly salary from emp table.

SQL> Select SAL + NVL (COMM, 0) totalmonthlysalary from emp;

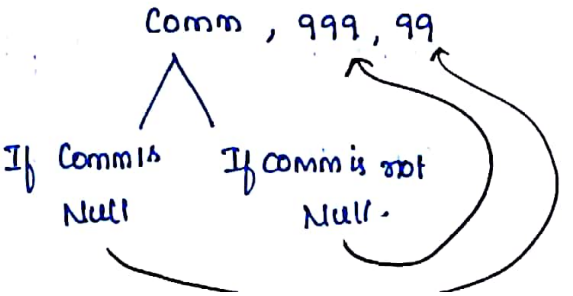
OR

SQL> Select NVL (SAL+ COMM, SAL) TotalMonthly Salary from emp;

08/Sep/2017

* NVL2

NVL2 (arg1, arg2, arg3)



→ NVL2() is used to convert Null & Not Null value to an actual value.

→ The datatype may or may not match

Select comm, nvl2(comm, 999, 99)
from emp;

comm	[NVL2(comm, 999, 99)
▭	99
300	999
500	999
▭	99

o/p.

Q) Display the total monthly salary of all the employees.

SQL> Select sal + nvl2(comm, comm, 0) totalmonthlysalary from emp;

OR

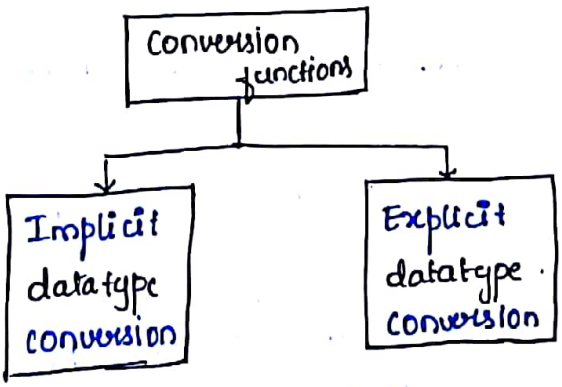
SQL> Select nvl2(comm, sal+comm, sal)

If commision is Null, NVL2 return 3rd arg

Q) Display the message as 'Not getting commission' for those employees whose comm is Null else display the message as 'getting some commission'.

SQL> Select comm, nvl2(comm, 'getting some commission', not getting commission) from emp;

** CONVERSION FUNCTIONS():



→ Data can be converted from one datatype to another datatype using Conversion functions.

→ Data can be converted implicitly by DB Server or by DB user using explicit datatype conversion function.

SQL> Select 10+'20' → Implicit datatype conversion done by DB Server.
from dual;

```
10+'20'
-------
     30
```

SQL> Select 10+'A'
from dual;

Error
Invalid number.

SQL> Select 10+ascii('A') → explicit datatype conversion done by DB user
from dual;

```
10+ascii('A')
-------------
           75
```

---

* to-number:

to-number is a function which converts string type of data to integer type of data.

SQL> Select 25+TO-NUMBER('20') from dual;

O/P:

```
---
45
```

SQL> Select * from emp where HIREDATE > '03-JUN-81';
→ Implicit datatype conversion by DB Server

SQL> Select SYSDATE - '03-JUN-81' from dual;
→ Invalid number.

NOTE:

In the above query the dbms is failing to convert the data implicitly. hence it is always recommended to make use of explicit data type conversion function, to convert data from one datatype to another datatype

* TO_DATE() → TO-Date is a function which converts String type of data into date type of data.

```
SQL> Select
     Sysdate - To_date ('03-Jun-81') from dual;

O/P
10439.077
```

** Using to_char function with Dates:

Syntax: to_char (arg1, arg2)

arg1 → DATE

arg2 → 'format model' (always should be enclosed with single quote)

→ to_char() is used to convert Date type of data into String format.

→ In to_char(), the format must be enclosed with side quotes.

* Elements of Date format Model:

DD → It displays date in numeric value.

DY → It displays 1st 3 characters of Day

DAY → It displays full day in words.

MM → It displays month in numeric value.

MON → It displays 1st 3 characters of month.

MONTH → It displays full month name in words

YY → It displays last 2 digits of the year

YYY → It displays last 3 digits of the year

YYYY → It displays the year in Numeric values.

YEAR → It displays full year in words.

SQL> Select Sysdate, To_char (SYSDATE, 'DD DY DAY MM MON MONTH YY YYY YYYY YEAR') from dual;

O/P:

01-JAN-10 01 FRI FRIDAY 01-JAN JANUARY 10 010 2010 TwentyTen

Q) Display the Systemdate in the following date format.

i) DY MON AND YYY

ii) DAY MONTH YEAR

iii) DD-MONTH-YY

iv) DD-MONTH-YYYY

SQL> Select Sysdate, To-char (SYSDATE, 'DY MON YYY') from dual;

SQL> Select Sysdate, To-char (Sysdate, 'DAY MONTH YEAR') from dual;

SQL> Select Sysdate, To_char (Sysdate, 'DD MONTH YY') from dual;

SQL> Select Sysdate, To-char (Sysdate, 'DD MONTH YYYY') from dual;

* Only to display Month.

SQL> Select Sysdate, TO-char (Sysdate, 'MONTH');

Q) Display the string for all the employees. 'ALLEN joined in 1981'.

Select concat (ename||' joined in' || TO-char (Hiredate, 'YYYY') from emp;

ename, ' '),

concat

Select (concat (ename, 'joined in), To-char (Hiredate, 'YYYY')) from emp;

Q) Display the foll string for all the employees 'ALLEN JOINED ON 12th of September';

Select ename || 'joined on' || TO-char (Hiredate, 'DD') || 'OF' || TO-char (Hiredate, 'MONTH') from emp;

SQL> Select concat(concat(concat(concat(ename, 'joined on'), TO-char (Hiredate, 'DD')), ' OF '), TO-char (Hiredate, 'Month'))
from dual;

Q) Display the String for all the employees 'ALLEN joined on Monday as a Salesman in the year 1981.

SQL> Select ename || 'joined on' || TO-char (HIREDATE, 'DAY') || 'as a' || job || 'in the year' || TO-char (HIREDATE, 'YYYY') from emp;

SQL> Select concat(concat(concat(concat(concat(concat(ename, 'joined on), to_char(Hiredate, 'Day'), 'as a'), job), 'in the year), to-char(Hiredate, 'YYYY')) from emp;

Q) Display the following date '03-DEC-81' in the following date format 'DAY-MONTH-YEAR'.

SQL> Select TO-char ('03-DEC-81', 'DAY, MONTH YEAR') from dual
Invalid number.

SQL> Select TO-char (TO-Date ('03-Dec-81'), 'Day Month Year')
from dual;

11/09/2017

** GROUP FUNCTIONS OR MULTIROW FUNCTIONS OR AGGREGATE FUNCTIONS

1) MIN()
2) MAX()
3) AVG()
4) COUNT()
5) SUM().

SQL> ED
Note pad will open

Group() takes group of records as an input and returns single o/p.

There are 5 Group functions
* Min()
* Max()
* Count()
* Avg()
* Sum().

→ MIN, MAX, COUNT Group functions works with any type of data.

→ Group functions ignores the NULL values

→ To consider the Null values we have to use general functions inside the Group functions

SQL> Select count(NVL(comm,0))
from emp.

* COUNT() : If the argument is '*' Count group() returns the count of number of records present inside the table.

→ If the argument is 'ColumnName' Count group() returns the count of actual records present inside the column.

SQL> Select count(*), count(COMM)
from emp.

Count(*)	Count(COMM)
14	4.

Q) Display the minimum Salary, Hiredate and maximum Salary, Hiredate

Select Min (sal), MIN(HIREDATE), MAX (SAL), MAX (HIREDATE) from emp;

NOTE:

SQL> Select ename, max(sal)

from emp; Error: not a single-group group function.

We cannot pass the group function along with the column name in Select clause. without using 'Group by clause'.

13|Sep|2017

** GROUP BY: Syntax: 5 Select xxxx [column Name], Group function.

1 From tableName

2 Where Condition (?)

3 Group by ColumnName

4 Order By ColumnName, Sorting technique;

→ Group By clause creates Sub-groups in a table.

→ The columns in the Select clause must be there in the group By clause

Ex: SQL> Select deptno, Count(*)

from emp

Group By deptno; → Should be same

SQL> Select deptno, count(*)

from emp

Group By sal; → X → This executes error.

→ The column in the Group By clause need not to be there in the select clause.

Ex: Select count(*)
from emp
Group By Deptno;

o/p
6
5
3

Q) Display departmentwise no. of employees.

Select deptno, count(*) from emp Group By Deptno;

Q) Display jobwise no of employees.

Select job, count(*) from emp Group By job;

Q) Display salarywise number of employees.

Select Sal, count(*) from emp Group By Sal;

Q) Display Deptno wise no of employees, Max Salary, total Salary from emp table.

Select Deptno, count(*), Max(sal), SUM(Sal) from emp
Group by Deptno;

Q) Display Deptnowise no of employees, max-sal, Avg sal of all employees whose job is Salesman, Analyst, Manager from emp table.

Select Deptno, count(*), Max(sal), Avg(sal) from emp
where job IN ('SALESMAN', 'ANALYST', 'MANAGER') Group By deptno

Q) Display jobwise no of employees, MAX Sal, total salary for all the employees whose mgr no is 7698, 7839, 7566 from emp table.

Select job, count(*), Max(Sal), Sum(Sal) from emp where MGR IN (7698, 7839, 7556) group by job;

8) Display Salarywise number of employes min Salary & Average salary for those employes whose employee no is in the range of 7521 and 7900 and joined after 20-FEB-81 from emp table.

Select Sal, count(*), MIN(SAL), AVG(SAL) from emp where empno between 7521 AND 7900 AND HIREDATE > '20-FEB-81' Group By Sal;

9) Display Salarywise no of employees, total salary and minimum salary of all employes who doesnot belong to deptno 30 or 60 and Sal greater than 2400.

Select Sal, count(*), SUM(Sal), MIN(Sal) from emp where Deptno NOT IN (30, 60) AND SAL > 2400) Group by Sal;

NOTE:

We can pass more than one column in Group by clause. If we pass more than 1 column, it groups the data based on both the columns.

SQL> Select Deptno, job, count(*) from emp Group By Deptno, job;

O/P

20	Salesman
30	President
20	clerk

Q) Display deptno wise max Salary and count of max salary in each deptno.

{ Select deptno, MAX(SAL), count((max)(sal)) from emp Group by deptno; } Wrong -

Explanation:

case 1> Select deptno, count(*)
from emp
group by deptno;

30	6
20	5
10	3

case 2> Select deptno, count(*)
where job in ('SALESMAN', 'ANALYST', 'MANAGER')
group by deptno;

30	5
20	3
10	1

FINAL:

```
Select Deptno, max(Sal), count(*)
from emp
where Sal in (Select Max(Sal) from emp Group by Deptno)
Group by deptno;
```

13/09/2017

** HAVING CLAUSE :

Syntax:

```
Select ***x [column Name], Group function
from tableName
where condition (?)
Group By column Name
Having column Name / Group function
Order by column Name, Sorting technique
```

→ Having clause is used to filter group results.

→ Group functions can be used in the Having clause to eliminate or filter the group result.

Q) Display Deptno wise number of employees for those departments which are having more than 3 employees in it.

```
Select Deptno, count(*) from emp Group by Deptno
Having Count(*) > 3;
```

o/p

DeptNO	count(*)
30	6
20	5

Q) Display jobwise no of employees max Salary for those jobs whose average Salary >= 1600.

```
Select job, count(*), MAX(Sal) from emp
Group by job Having AVG(Sal) >=1600;
```

Q) Display Salary wise no of employees for those Salaries which is greater than 1500

```
Select Sal, count(*) from emp Group by Sal
Having Sal > 1500;
```

Q) Find the duplicate Salaries from the emp table.

```
Select Sal, count(*)
from emp
Group by Sal
Having count(*) >1;
```

→ This returns only the duplicate values.

Q) Find the duplicate job from employee table.

```
Select job, count (*)
from emp  Group by job.
Having count (*) >1;
```

Q) Display jobwise no of employees, Max (Sal), Avg (Sal) for those employees whose manager no is 7698, 7566, 7782 or 7902. If total salary is greater than 1900.

```
Select job, count (*), Max (Sal), Avg (sal) from emp
where MGR IN ('7698, 7566, 7782, 7902), Group by job
Having Sum (sal) > 1900;
```

Q) Display dept no wise no of employees, total salary, min salary for all the Salesman, Analyst, clerk if the max sal is <= 3500.

```
Select deptno, count(*), Sum (sal), Min (sal) from emp
where job IN ('SALESMAN', 'ANALYST', 'CLERK') Group by deptno
Having Max(Sal) <= 3500;
```

Q) Display Salarywise no of employees, Min Sal, Avg Sal for all the employees who are joined from '2-APR-81'. If the Avg salary is less than 2000 and job is Mgr Salesman or clerk.

```
Select Sal, count (*), Min(Sal), Avg (Sal) from emp
where Hiredate >='2-APR-81' Group by Sal
Having AVG (sal) X 2000 AND JOB IN ('MANAGER', 'SALESMAN',
                                    'CLERK');
```

This gives an error

NOTE:

The column in the Group by clause can only be passed in the HAVING clause.

Q> Display jobwise no of employees. Max(Sal), Min(Sal) and AVG(Sal) for those employees whose employee number is in the range of 7566 and 7900 if total salary is less than equals to 4000 And job is MGR, President Or clerk.

```
Select job, count(*), Max(Sal), Min(Sal), AVG(Sal)
from emp
where Sal Between 7566 AND 7900
Group by job
Having SUM(Sal) <=4000 AND
job IN('MANAGER', 'PRESIDENT', 'CLERK');
```

Cannot pass Group fns in where clause

** SUB-QUERIES.

Query inside a query is known as Sub-Query Or Nested Queries.

→ In Sub queries always the inner query executes first, the result of the inner query will be used as a input argument to the outer query.

→ Sub-queries are used whenever we have to find the unknown values.

Example: Select *
from emp
where Sal = (Select Max(Sal) from emp);

Q> Display all the employees information who are getting maximum salary from emp table.

Select *
from emp where Sal = (Select Max (Sal) from emp);

Q> Display all the employes whose sal is greater than Allen.

Select *
from emp where Sal > (Select sal from emp where ename = 'ALLEN

14/09/2017

Syntax of Sub-Query:

Select Selectlist
from tablename
where Expression operator (Select Selectlist from tablename);

column Name

Q> Display Allens Manager information.

Select * from emp where empno = (Select MGR from emp where ename = 'ALLEN');

Q) Display Allen's Manager Manager's information.

```
Select * from emp
    where empno = (Select MGR from emp
                   where empno = (Select MGR from emp
                                  where ename = 'ALLEN'));
```

(7839, 7698)

Q) Display Smith Manager Manager's Manager's information.

```
Select * from emp
    where empno = (Select MGR from emp
                   where empno = (Select MGR from emp
                                  where empno = (Select MGR from emp
                                                 where ename = 'SMITH')));
```

Q) Display all the employees whose salary is same as Scott & job is same as Ford.

```
Select * from emp
    where SAL = (Select Sal from emp where ename = 'SCOTT')
    AND JOB = (Select job from emp where ename = 'FORD');
```

Q) Display all the employees who are working in location Dallas.

```
Select * from emp
    where DeptNO = (Select Deptno from Dept where LOC = 'DALL
```

Q) Display all the employees who are working in Research Dept.

```
Select * from emp
    where DeptNO = (Select Deptno from Dept where dname = 'RESEAR
                                                          CH');
```

Q) Display the departmental information for those employees whose mgr no is 7698.

```
Select * from Dept where Deptno = (Select Deptno from emp
                                   where MGR = 7698);
```

Error: Single row Subquery returns more than one value..

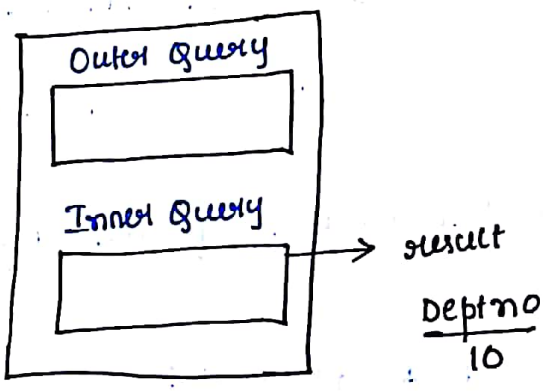
```
Select * from dept
        where deptno IN (Select Deptno from emp where MGR =
                                                    7698);
```

→ There are 2 types of Sub-Queries.

1) Single row Sub-Query
2) Multiple row Sub-Query.

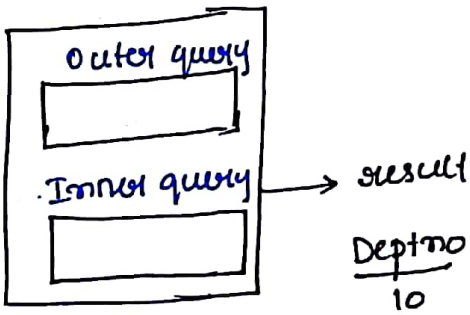
* Single row Sub-Query

→ If the Subquery returns only one result, then it is known as Single row Subquery.



* Multiple Row Sub-Query.

→ If the Subquery returns more than one result, then it is known as Multi row Subquery.



Q) Display the dept names which are having 3 employees in it.

```
Select Dname
from dept where deptno IN ( Select Deptno, from emp group
                            by deptno Having Count(*)=3);
```

→ If (Select Deptno, count(throws error.

Q) List the department names which are having more than 3 employees in it.

```
Select Dname from dept
where Deptno IN (Select Deptno from emp group by deptno
                 Having Count(*) > 3);
```

Q) Display the information of an employee whose salary is second maximum.

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

Q) Display the information of an employee whose getting fifth maximum Salary.

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

Q) Display all the employees who are working in Research & Sales department.

IN (Since it returns more than one value)

Select * from emp where Deptno = (Select DEPTNO from dept where Dname IN ('RESEARCH', 'SALES'));

Q) Display all the employees whose Salary is greater than Avg salary of location Newyork or Boston.

Select * from emp where Sal > (Select Avg (Sal) from emp where deptno IN (Select deptno from dept where LOC IN ('NEYORK', 'BOSTON')));

Q) Display all the employees whose Salary is greater than Max Salary of ACCOUNTING AND SALES Deptno

Select * from emp where Sal > (Select Max (Sal) from emp where deptno IN (Select deptno from dept where Dname IN ('ACCOUNTING', 'SALES')));

Q) List the deptnames for those employees whose deptno is same as the employee who is getting max salary from the location 'NEWYORK' 'MYSORE'

Select * from dept where deptno IN (Select Deptno from emp where Sal = (Select MAX (sal) from emp where deptno IN (Select deptno from dept where LOC IN ('NEWYORK', 'MYSORE'))));

1> List the employees who has salary greater than Allen MGR's salary.

2> List the employees working in 'Research' department.

3> List the department names that are having analyst.

4> List the employees in Research and Sales department.

5> List the department names which are having salesman in it.

6> Display the employees whose location is having at least one 'o' in it.

7> List the department names that are having atleast 1 employee in it.

8> List the department names that are having atleast 4 employees in it

9> Display the department names which are having atleast 3 clerks in it.

10> List the department names that are having no employees at all.

11> Display all the employees whose job is same as Scott.

12> Display Scott managers managers department name.

13> List the employees whose job is same as Scott and their salary greater than Smith's salary.

14> Display all the employees whose job is same as Scott and Allen.

15> Display all the employees who are actual managers.

16> Display who are all the employees reporting to Scott.

17> Display all the employees who are not manager.

Q2) Select * from emp where deptno = (Select deptno from dept where dname = 'RESEARCH');

Q3) Select * from dept where deptno IN (Select deptno from emp where job = 'ANALYST');

Q4) Select * from emp where deptno IN (Select deptno from dept where dname IN ('RESEARCH', 'SALES') Order by deptno;

Q5) Select * from dept where deptno in (Select deptno from emp where job = 'SALESMAN');

Q6) Select * from emp where deptno in (Select deptno from dept where LOC LIKE '%O%') order by deptno;

Q7) Select dname from dept where deptno IN (select deptno from emp group by deptno having count(*) > 0);

Q8) Select dname from dept where deptno in (Select deptno from emp group by deptno having count(*) >= 4);

Q9) Select dname from dept where deptno in (Select deptno from emp where job = 'CLERK' group by deptno having count ('CLERK') = 2);

Q10) Select * from dept where deptno not IN (Select deptno from emp);

Peer Note: where deptno is not null;

Q11) Select * from emp where job = (Select job from emp where ename = 'SCOTT');

Q12> Select dname from dept where deptno = (Select deptno from emp where empno = (Select MGR from emp where empno = (Select MGR from emp where ename = 'SCOTT')));

Q14> Select * from emp where job IN (Select job from emp where ename in ('SCOTT', 'ALLEN'));

Q13> Select * from emp where job = (Select job from emp where ename = 'SCOTT') AND SAL > (Select Sal from emp where ename = 'SMITH');

Q11> Select * from emp where SAL > (Select Sal from emp where empno = (Select MGR from emp where ename = 'ALLEN'));

Q16> Select * from emp where MGR = (Select empno from emp where ename = 'SCOTT');

Q17> Select * from emp where empno NOT IN (Select MGR from emp);

Q15> Select * from emp where empno IN (Select MGR from emp);

The Subquery may or may not return result if the value is Null

Q> Display all the employees whose Salary is greater than any of the Salesman Salary

Min

Select * from emp where Sal > (Select (s from emp where b = 'SALESMAN');

15/09/2017

OR

Select * from emp where SAL > ANY (Select Sal from emp where job = 'SALESMAN');

ALL / ANY (KEYWORDS OR CLAUSE)

Q) Display all the employees whose salary is greater than all the Salesman salary.

Select * from emp where SAL > ALL (Select Sal from emp where job = 'SALESMAN');

Q) Display all the employees whose salary is greater than any of the managers salary.

Select * from emp where SAL > ANY (Select Sal from emp where job = 'MANAGER');

NOTE:

ALL / ANY (Keywords or clauses) → All / Any keywords are used along with relational operators in where clause to handle multiple values returned by Sub-queries.

** PSEUDO COLUMNS / GHOST COLUMNS:

→ Pseudo columns are the columns present in all the tables of RDBMS but not visible until we explicitly call those columns.

* ROWNUM : The main functionality of ROWNUM () is to check whether the record is present in table or not.

→ Using RowNum () it is possible to display the first record or first few records or all the records but not the intermediate records.

Q) Display the first record of the table using Rownum ().

Select * from emp where ROWNUM = 1;

Q) Display the top 5 records of the employee table.

Select * from emp where ROWNUM <= 5;

* ROWID : Row ID is the address given to each record of the table.

→ Each record has unique address.

→ Using RowId it is possible to display the first or last or the intermediate records of the table.

Q) Display the last record of the table using RowId.

Select * from emp where ROWID = (Select Max(ROWID) from emp);

Q) Display the first record of the table using RowId

Select * from emp where ROWID = (Select Min(ROWID) from emp);

** IN-LINE SUBQUERIES | IN-LINE VIEW SUBQUERIES

→ If we pass subqueries from subclass then it is known as Inline View Subqueries.

Syntax: Select Selectlist
From (Select Selectlist from tableName);

Ex: SQL> Select * from (Select empno, ename, job from emp);

o/p displayed

SQL> Select sal from (Select empno, ename, job from emp);

Displays error.

Q) Display the top 25% salary employees record from emp table.

Select * from (Select * from emp Order by sal Desc) where
ROWNUM <= (select count(*)/4 from emp);

Q) Display the least 25% salary employees record from emp table.

Select * from (Select * from emp Order by sal) where
ROWNUM <= (Select count(*)/4 from emp);

Q) Display the middle 50% salary employees record from emp table.

Select * from emp where empno not in (Select empno from
(Select * from emp order by sal) where rownum <= (Select
count(*)/4 from emp)) AND empno not in (Select empno from
(select * from emp Order by sal desc) where rownum <=
(Select count(*)/4 from emp));

** JOINS

Joins is one of the capabilities of SELECT clause.

→ It is used to display rows and columns from different tables.

→ According to Joins, there are two types: 1) Inner Joins
2) Outer Joins.

** INNER JOIN Versus OUTER JOIN

INNER JOINS

→ Inner joint condition which returns only the matching records of the table known as Inner joins.

OUTER JOIN

→ A Joint condition which returns both matching & un-matching records of the table based on join condition.

Q) Display the employee name with their respective dname for all the employees.

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

Q) Display employee name, deptname, location for all the employees.

SQL> Select ename, dname, LOC
from emp, dept
where emp.deptno = dept.deptno;

Q> Display employee name, dept name, loc for all the employees whose sal is greater than equals to 2400.

```
Select ename, dname, loc from emp, dept
where emp.deptno = dept.deptno AND emp.sal >= 2400;
```

Q> Display emp name, dept name, loc for all the employees whose salary is greater than Scott salary.

```
Select ename, dname, loc from emp, dept
where emp.deptno = dept.deptno And SAL >= (Select sal from emp
      where ename = 'SCOTT');
```

Q> Display employee name with their respective Salary Grade.

```
Select ename, grade from emp, Salgrade
where sal between losal and hisal;
```

Q> Display empname, dname, Loc, grade for all the employees

```
Select ename, dname, loc, Grade
from emp, Dept, Salgrade
where emp.deptno = dept.deptno AND SAL Between Losal And Hisal
```

NOTE:

To display the data from 'n' tables we have to write 'n-1' join condition.

Q> Display ename, dname, loc, grade for all the employees whose job is same as Scotts Manager job from emp table.

SQL> Select ename, dname, loc, Grade

from emp, dept, salgrade

where emp.deptno = dept.deptno AND SAL between Losal and

Hisal AND JOB = (Select job from emp where MGR EMPNO =

(Select Mgr from emp where ename = 'SCOTT'));

Q> Display empname, dname, loc, Grade for all the employees whose loc is 'NEWYORK', 'DALLAS' OR 'BOSTON' AND SAL Grade in the range of 3 and 5.

SQL> Select ename, dname, Loc, Grade

from emp, dept, salgrade

where emp.deptno = dept.deptno AND SAL between LOSal And Hisal

LOC IN ('NEW YORK', 'DALLAS' 'BOSTON') AND GRADE BETWEEN

3 AND 5;

Q> Display dept name wise number of employees.

SQL> Select dname, Count(*) from emp, dept

from emp, dept

where emp.deptno = dept.deptno

Group by Dname;

** NOTE:

Sub-query may or may not return the o/p. if the inner query result contain Null values.

Q1) Select * from emp where empno NOT IN (Select MGR from emp where MGR IS NOT Null); o/p :: 8 rows selected.

19/09/2017

** TYPES OF ORACLE JOINS

1) EQUI JOIN
2) NON-EQUI JOIN
3) OUTER JOIN
4) SELF JOIN.

** EQUI-JOIN: Join condition which is written using equal to (=) operator is known as Equijoin.

Ex: SQL> Select ename, dname
from emp, dept
where emp.deptno = dept.deptno;

** NON-EQUI JOIN: Join condition written apart from using = condition is known as Non-equi join.

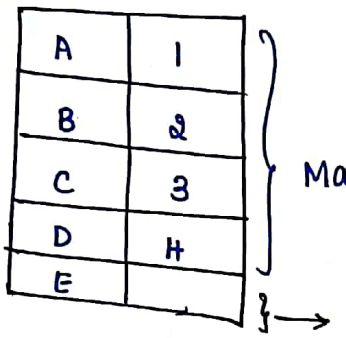
Ex: SQL> Select ename, grade
from emp, salgrade
where sal between Losal and Hisal;

** Outer Joins:

- Left OJ
- Right OJ
- Full OJ

* Left OJ:

Left outer join condition displays the result of inner join ie (matching records) and unmatching record from left table.



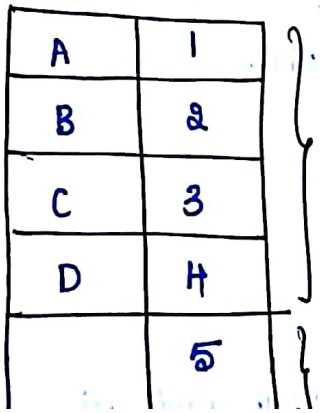
Example:

```
Select ename, dname
from emp, dept
where emp.deptno (+) = dept.deptno;
```

(+) → Left

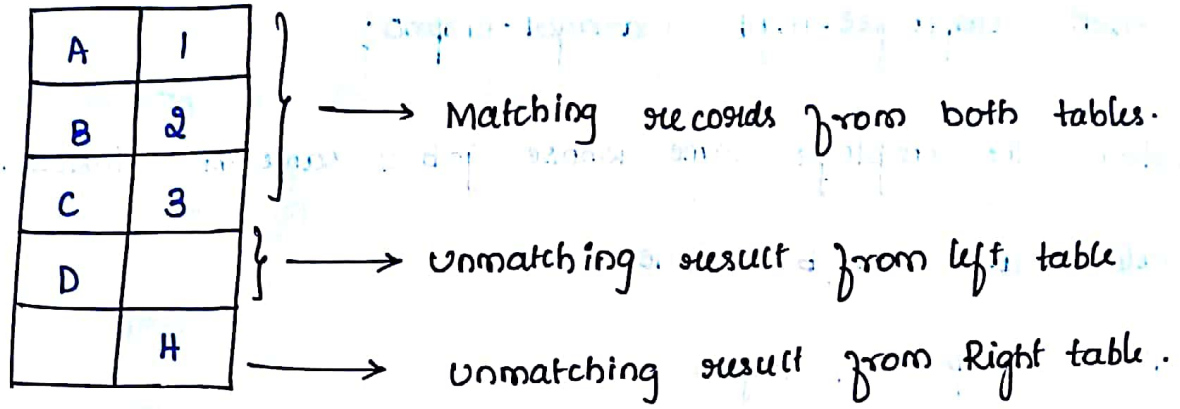
* Right OJ:

Right outer join condition displays the result of inner join ie (matching records) & unmatching record from right table.



* FULL OUTER JOIN:

Full outer join returns the result of inner join & unmatching record from left & Right table.




```
SQL> Select   ename, dname
     from     emp, dept
     where    emp.deptno = dept.deptno(+)
     UNION
     Select   ename, dname
     from     emp, dept
     where    emp.deptno(+) = dept.deptno ;
```

Left table.

Right table.

** SELF-JOIN:

A join within a table is known as SELF-Join.

OR Displaying the data from the same table using same column or multiple columns is known as Self-Joins.

→ Table Aliasing is must for Self-Joins.

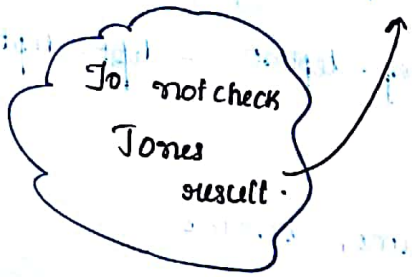
```
Ex: Select ename, x.mgr
    from   emp x ;
```

Q: Display employee name with their respective Managers.

```
Select employee.ename, Manager.ename
from emp employee, emp manager
where employee.mgr = manager.empno;
```

Q: Display the employee name whose job is same as 'JONES'.

```
Select a.ename, b.ename
from emp a, emp b
where a.job = b.job AND a.ename = 'JONES';
      AND B.ename <> 'JONES';
```



Q: Display all the employees whose Sal is same as MARTIN Salary

```
Select B.* from emp a emp b
where A.Sal = B.Sal And A.ename = 'MARTIN' AND B.ENAME
                                          <> 'MARTIN';
```

Q: Display the employee name who joined before their own manager.

```
Select A.ename, B.ename
From emp A, emp B
where A.MGR = B, EmpNO AND A.HIREDATE < B.HIREDATE';
```

Q) Display employee name with their Manager name whose salary is less than their own manager salary.

```
Select A.ename , B.ename
from emp A , emp B
where A.MGR = B.empno and A.Sal < B.Sal ;
```

Q) Display all the employees who joined after their own manager.

```
Select A.ename, B.ename
from emp , emp B.
where A.MGR = B.empno and A.Hiredate > B.Hiredate ;
```

Q) Display all the employees whose salary is greater than their own mgr.

```
Select A.ename, B.ename
from emp A , emp B
where A.MGR = B.empno and A.Sal > B.Sal ;
```

** CARTESIAN JOINS :

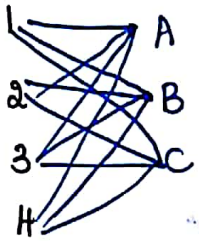
→ It is cross product of two table.

→ Cartesian join is formed whenever the join condition is omitted.

→ It is an invalid join condition.

Ex: SQL> Select ename, dname
from emp , dept ;

O/P: 56 rows selected. (14 X 4)



L.T R.T

** ANSI JOINS:

A Join condition written using 'ON' clause is known as ANSI Joins.

Ex:
```
Select ename , dname
from emp join Dept
ON emp.deptno = dept.deptno;
```
} Example for Inner Join using ANSI Joins.

OR
```
Select ename, dname
from emp inner join dept
ON emp.deptno = dept.deptno;
```

Ex: To display the data from more than two tables using ANSI Join.

```
SQL> Select ename, dname, Grade
     from emp join Dept on emp.Deptno = Dept.deptno
     Join Salgrade on Sal between Losal and Hisal;
```

```
SQL> Select ename, dname, Grade
     from emp join Dept
```

Q> Display the employee name with their respective dname for all the employees.

SQL> Select ename, dname from emp join dept ON emp.deptno = dept.deptno;

Q> Display employee name, dept name, location for all the employees.

SQL> Select ename, deptname as dname, LOC from emp join dept

ON emp.deptno = dept.deptno;

Q> Display employee name, dept name, loc for all the employees whose sal is greater than equals to 2400.

SQL> Select ename, deptname as dname, LOC

from emp join Dept

ON emp.deptno = dept.deptno AND SAL >= 2400;

Q> Display emp name, dept name, LOC for all the employees whose salary is greater than Scott Salary.

SQL> Select ename, dname, LOC from emp join Dept

ON emp.deptno = dept.deptno AND SAL >= (Select Sal from emp where ename = 'SCOTT');

Q> Display employee name with their respective Salary Grade.

SQL> Select ename, grade from emp join Salgrade ON Sal between Losal and Hisal;

Q> Display ename, dname, loc, grade for all the employees.

SQL> Select ename, dname, LOC, Grade from emp join Dept on emp.deptno = Dept.Deptno Join Salgrade ON Sal between Losal and Hisal;

Q) Display ename, dname, Loc, grade for all the employees whose job is same as Scott Manager job from emp table.

SQL> Select ename, dname, Loc, grade from emp join dept
ON emp.deptno = dept.deptno AND JOB = (Select job from emp
where empno = (Select Mgr from emp where ename = 'SCOTT'))
JOIN Salgrade on Sal between Losal and Hisal;

Q) Display ename, dname, Loc, Grade for all the employees whose Location is 'NEWYORK', 'DALLAS' OR 'BOSTON' AND Salgrade in the range of 3 and 5.

SQL> Select ename, dname, Loc, grade from emp
join dept on emp.deptno = dept.deptno and LOC IN ('NEWYORK',
'DALLAS', 'BOSTON') join Salgrade ON Sal between losal and
hisal and grade between 3 and 5.

Q) Display dept namewise number of employees.

Select dname, count(*)
from dept join emp
ON emp.deptno = dept.deptno
Group by dname;

** Example for Right/Left outerjoin using ANSI Syntax :

```
Select ename, dname
from emp RIGHT/Left OUTER JOIN Dept
ON emp.Deptno = Dept.Deptno;
```

OR → RIGHT JOIN / LEFT JOIN

** Example for Full outer join using ANSI Syntax :

```
Select ename, dname
from emp FULL OUTER JOIN Dept
ON emp.Deptno = Dept.Deptno;
```

OR → FULL JOIN

** SEMI JOINS

Displaying the data from one table using the reference from another table is known as SEMI JOINS.

All Subqueries are Sub-joins

** ANTI JOINS

If we use not Logical operators in SEMI join then it is known as ANTI JOINS.

Ex: Select * from emp
where empno NOT IN (Select MGR from emp);

** CROSS JOIN : It is the cross product of two tables. It is similar to Cartesian Join.

Ex in ANSI : Select ename, dname
from emp CROSS JOIN Dept ;

** ORACLE SERVER SET OPERATORS.

20/09/17

- Union
- Union All
- Intersection
- Minus.

A = {1, 2, 3, 4, 5, 6}

B = {4, 5, 6, 7}

A ∪ B = {1, 2, 3, 4, 5, 6, 7}

A ∪ALL B = {1, 2, 3, 4, 5, 6, 4, 5, 6, 7}

A ∩ B = {4, 5, 6}

A - B = {1, 2, 3}

B - A = {7}

A = {CL, MI, JO, TU, WA, KI}

B = {TU, WA, KI, FD}

A ∪ B = {CL, MI, JO, TU, WA, KI, FD}

A ∪ALL B = {All names of A & B}

A ∩ B = {WA, TU, KI}

A - B = {CL, MI, JO}

B - A = {FD}.

Set Operators are used within the Sql Statement, but the Set Operators operates on the result of the Sql Statement

Ex: Statement 1
Set operator
Statement 2
} Result
Set operator
Statement 3

* UNION: Use union operator between the statement to return unique values from both the statements.

* UNIONALL: Use UnionAll operator to return all the values or result from both the statements.

* INTERSECT: Use Intersect operator to return common results or sharing the result between the statement.

* MINUS: Use MINUS operator to return unique result from statement 1 which is not present in statement 2.

Example: Select ename from emp whose Sal >= 1300
minus / union / union all / intersect
select ename from emp whose deptno = 30;

Ex 2: Select ename from emp where deptno = 30 → String type

UNION

Select deptno from emp where ..... → int type.

O/P: Error occurs because of data mismatch

Q) Display only the unmatching records from left table.

SQL> Select ename, dname

from emp left outer join dept

on emp.deptno = dept.deptno

MINUS

Select ename, dname

from emp inner join dept

on emp.deptno = dept.deptno;

O/P

Smith

Q) Dislay only the unmatching records from Right table.

SQL> Select ename, dname

from emp right outer join dept

on emp.deptno = dept.deptno

MINUS

Select ename, dname

from emp inner join dept

on emp.deptno = dept.deptno.

O/P

Operations.

Q) Display the unmatching records from both the table.

```
Select ename, dname
from emp full outer join dept
on emp.deptno = dept.deptno
MINUS
Select ename, dname
from emp join dept deptno
ON emp.deptno = dept.deptno;
```

Q) Display the last record of the table using Set operators.

```
Select * from emp
minus
Select * from emp where rownum < (Select count(*) from emp);
```

Q) Display the last but one record of employee table using Set operators.

```
Select * from emp where rownum<(Select count(*) from emp)
                                ↓ <= (Select count(*)-1 from emp)
MINUS
Select * from emp where rownum < (Select count(*)-1 from emp);
                                ↓
- - - -                         <= (Select count(*)-2 from emp);
```

Q) Display the 10th record of the table using SET Operators.

```
Select * from emp where rownum <= 10 MINUS
Select * from emp where rownum <= (Select count(*) -5 from
                                   emp);
```

Q> Display 10th & 12th record of a table using Set Operator.

```
(Select * from emp where rownum <= 12
minus
Select * from emp where rownum <= (Select count(*) - 3 from emp))
Union all
(Select * from emp where rownum <= 10
minus
Select * from emp where rownum <= (Select count(*) - 5 from emp))
```

Q> Display the middle 50% of salary using Set Operator.

```
Select * from emp
minus
(Select * from (Select * from emp order by sal desc)
where rownum <= (Select count(*)/4) from emp)
UNION
Select * from (Select * from emp order by sal) where rownum <=
(Select count(*)/4 from emp));
```

OR

```
Select * from emp
minus
Select * from (Select * from emp order by sal desc) where
rownum <= (Select count(*)/4 from emp) minus
Select * from (Select * from emp order by sal) where rownum <=
```

select count(*)/4 from emp) ;

** CO-RELATED SUB-QUERIES

→ It is a combination of Subqueries and Self joins.

→ Co-related Subqueries are used for row by row processing whenever the unknown value changes for each & every record of Inner query.

→ In Co-related Subquery always the outer query execute first and the result of the outer query (ie the value present in the column specified in Self join condition of Outer query) will be compared with each and every values of the inner query (ex: emp.sal column values).

Q: Write a co-related sub query to display 4th Maximum Salary from emp table:

```
Select * from emp X
    Where 4 = (Select count (Distinct sal)
                from emp
                where X.sal <= emp.sal);
```

X.Sal	emp.sal	4 = 12 X	4 = 6 X
800	800	X.Sal <= emp.sal	X.sal <= emp.sal
1600	1600	800 <= 800	1600 <= 800
1250	1250	800 <= 1600	1600 <= 1600
2975	2975	800 <= 1250	1600 <= 1250
1250	1250	800 <= 2975	1600 <= 2975
2850	2850	800 <= 1250	1600 <= 1250
2450	2450	800 <= 2850	1600 <= 2850
3000	3000	800 <= 2450	1600 <= 2450
5000	5000	800 <= 3000	1600 <= 3000
1500	1500	800 <= 5000	1600 <= 5000
1100	1100	800 <= 1500	
950	3000 950 1300	800 <= 1100	
		800 <= 950	
		800 <= 1300	
		1600 <= 1600	

1600 <= 1100
1600 <= 950
1600 <= 1300

4 = 9 X

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

4 = 3 X

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

4 = 4 ✓

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

Q. Write a co-related subquery to display 5th max salary from emp table

```
Select * from emp x
where 5 = (Select count (Distinct Sal)
from emp where x.Sal <= emp.sal);
```

x.Sal <= emp.Sal

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

5 = 5

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

. . . . . .

21/09/2017

Q) Write a Corelated subquery to display 5th & 7th Maximum salary.

```
Select * from emp x
Where (Select count(distinct sal) from emp
      where x.sal <= emp.sal) in (5,7);
```

Q) Write a corelated subquery to display 3rd minimum salary from emp table.

```
Select * from emp x
Where 3 = (Select count (Distinct sal)
from emp where x.sal >= emp.sal :
```

Q) Write a corelated subquery to display 3rd, 7th & 9th minimum sal.

```
Select * from emp x
where (Select count(distinct sal) from emp
      where x.sal >= emp.sal) in (3,7,9);
```

O/P

Allen
Blake
Adams.

Q) Write a corelated subquery to display top 5 minimum salaries from emp table.

```
Select * from emp x
where (Select count(distinct sal) from emp where x.sal >= emp.sal)
<= 5.
```

O/P: 6 rows selected (one duplicate value since it is returning outer query result).

Q) Write a co-related subquery to display 5th record of the table.

```
Select * from emp x where 5 = (Select count(rowid) from emp
      where x.rowid >= emp.rowid);
```

count ①

X.rowid ≥ emp.rowid

AAA ≥ AAA

AAA ≥ AAB

AAA ≥ AAC

AAA ≥ AAD

AAA ≥ AAE

count = 2

X.rowid ≥ emp.rowid

AAA ≥ AAB

≥ AAB

≥ AAC

≥ AAD

≥ AAE.

Q) Write a co-related subquery to display top 3 records from the emp table.

```
Select * from emp x
where 3>= (select count(rowid) from emp where x.rowid>=emp.rowid)
```

Q) Write a co-related query to display nth max salary.

```
SQL> Select * from emp x
where &n = (select count(distinct sal) from emp
            where x.sal<=emp.sal);
```

Don't write only &

Enter the value of n : 12

O/P : 12th record is displayed.

Q) Write a co-related subquery to display the employees who joined before their own manager.

```
SQL> Select * from emp x
where x.Hiredate < (select emp.hiredate from emp where
                    x.MGR = emp.empno);
```

17-DEC-80 < 03-DEC-81

Q) Write a co-related subquery who are earning more than their own manager.

```
Select * from emp x
where x.sal > (Select emp.sal from emp where x.MGR = emp.empno);
```

O/P : 2 rows selected.

Q) Write a co-related subquery to display first employee in each dept

```
Select * from emp x
where x.hiredate in (Select min (hiredate) from emp group by deptno);
```

O/P : Display 4 rows. (It also group Null values.)

OR

```
Select * from emp x
where x.hiredate = (select min (hiredate) from emp
                    where x.deptno = emp.deptno);
```

O/P : Display 3 rows.

OR

```
Seled * from emp x
where x.hiredate in (Select min (hiredate) from emp where deptno is
                     NOT NULL group by deptno);
```

O/P : Displays 3 rows.

Not a co-related query bcoz it does not contain Equi-join condition

Q) Write a co-related subquery to display minimum salary from each dept.

```
Select * from emp x
where x.Sal = (Select min (sal) from emp where x.deptno = emp.
               deptno);
```

Q) Write a corulated subquery to display bottom 5 records from emp table.

```
Select * from emp x where 5 >= (Select count (rowid) from emp
                                where x.rowid <= emp.rowid);
```

OR

```
Select * from emp x
    where (select count(rowid) from emp where x.rowid <= emp.row
id) <= 5;
```

Q1) Display the top 3 minimum salary from emp table.

2) Write a corelated subquery to display minimum COMM from each dept & commission must be greater than 100.

3) Write a corelated subquery to display 7, 8th & 9th max salary.

4) Write a corelated subquery to display employee who joined after their own managers.

5) Write a corelated subquery who are earning their own managers.

6) Display the last employee from each dept.

7) Write the corelated subquery to display nth max salary from emp table.

NOTE:

Select * from tab; → Displays the database objects.
(tables, views)

Connect System/tiger OR Connect Scott/tiger

Q1) Select * from emp x

where (Select count (distinct sal) from emp where x.sal >= emp.sal)
<= 3 ;

Q2) Select * from emp x

where COMM IN (Select MIN(COMM) from emp group by deptno);

Q3) Select * from emp x

where (Select count (distinct sal) from emp where x.sal >= emp.sal)
in (7, 8, 9);

Q4) Select * from emp x

Where x.hiredate > (Select emp.hiredate from emp where x.Mgr =
emp.empno);

Q5) Select * from emp x

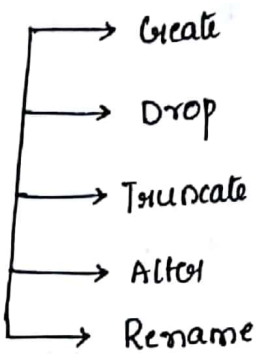
where x.Sal < (Select sal from emp where x.Mgr = emp.empno);

Q6) Select * from emp x where x.hiredate = Any (Select max (hiredate)
from emp where deptno is not null group by deptno);

Q7) Select * from emp x where &n = (Select count (distinct sal) from
emp where x.sal <= emp.sal);

** DATA DEFINITION LANGUAGE :

The various commands in DDL are :



* CREATE : It creates the database objects (table, view, index etc).

Some of the basic Datatypes we use in SQL are

1) CHAR : It stores the fixed length character data - 2000 bytes.

2) VARCHAR : → It stores the variable length character data.

It can store alphanumeric data.

Another difference is

In Char, maximum value we can store is 2000 characters.

In Varchar, maximum value we can store 4000 characters.

3) NUMBER : → It stores Numeric data.

For ex : 1) Sal number(4)

Here the maximum possible value is 9999.

2) Sal number (6,2)

Here, 2 → Scale (total number of decimal places)

6 → precision (total number of digits)

sal number (4,3);

sal number (2,2).

4) DATE : ⟶ It stores date and time.

⟶ No need to specify any length for this type.

Date is always displayed in the default format dd-month-yyyy.

5) BLOB : Stands for ⟶ Binary Large Object.

It stores binary data (images, movies, music files) within the database. It stores upto 4GB.

6) CLOB :

Stands for ⟶ Character Large Object.

It stores plain character data like Varchar field upto 4GB.

NOTE :

Varchar2 ⟶ from 10g, Varchar & Varchar2 are the same.

Earlier, Varchar was supporting upto 2000 characters & varchar2 was supporting upto 4000 characters.

** CREATE

Syntax :

```
Create table tableName (
        ColumnName  datatype,
        ColumnName  datatype,
        ColumnName  datatype,
        :
        ColumnName  datatype);
```

Q.: Create the following tables

PRODUCTS

ProductID (PK)

ProdName (NOT NULL)

Qty (chk > 0)

Description

ORDERS

ProductID (FK from products)

OrderID (PK)

Qty_sold (chk > 0)

Price

Order Date

```
Create table Products (
    ProductID    Number (200)   primary key,
    ProdName     Varchar (30)   NOT NULL,
    Qty          Number (10)    check ( Qty > 0),
    Description  varchar (10) ) ;

Create table Orders (                    for Oracle
                                         depends on Database.
    ProductID    Number(20)     references products (ProductID).
    OrderID      Number (10)    primary key,
    Qty_sold     Number(10)     check (Qty > 0),
    Price        Number(10,4),
    Order_Date   Date ) ;
```

Q) Create the following tables

a) Table name : STUDENTS { regno (PK), name (NN), Semester, DOB, phone }

b) Table name : BOOKS { bookno (PK), bname, author }

c) Table name : LIBRARY { RegNO (FK from students, bookno (FK from books), DOI - date of Issue, DOR → Date of Return }

```
Create table Students (
    RegNO       Varchar (15)    primarykey,
    Name        Varchar (20)    NOTNULL,
    Semester    Varchar (5),
    DateofBirth Date,
    Phone       Number (15));
```

```
Create table Books(
    Bookno      Number (20) primarykey,
    Bookname    Varchar (25),
    Author      Varchar (20));
```

```
Create table Library (
    RegNO   Varchar(15)  references Students(RegNO),
    Bookno  Varchar (20) references Books (Bookno),
    DOI     Date,
    DOR     Date, );
```

22/09/17

** TRUNCATE :

**

Q** CREATE A DUPLICATE COPY OF A TABLE & copy both structure and data of a table.

Create table empll as Select * from emp ;

↓ New tablename

↓ table to be copied.

* NO constrains will be copied. (Ex: Not Null).

Q. Create a duplicate copy of a table and copy only the structure of table.

Create table empl2 as select * from emp where 1 = 2 ;

↓ This checks the condition all the records will be false.

TRUNCATE statement deletes all the records of the table permanently but the structure of the table remains same.

Syntax : Truncate table tablename ;

Ex: Truncate table empll ;

O/P : Table truncated.

SQL> Show recyclebin

All the drop tables will be stored in Recyclebin.

** DROP

Drop statement deletes both Structure & Data inside the table.

SQL> purge table emp1;
→ Delete the table from recyclebin

→ All the dropped tables will be stored in recycle bin which can be restored back to the database users.

Syntax: drop table tablename ;

Ex: drop table empla ;

** FLASH BACK

Flash Back command is used to restore the database object in recyclebin back to the database user.

Syntax: flashback table tablename to before drop ;

Ex: flashback table empla to before drop ;

Syntax for dropping a table permentely from the database without storing in RecycleBin

Syntax: drop table empla purge ;

** RENAME: Rename statement is used to rename the database objects name in the database.

Syntax: rename tablename to newtablename ;

Ex: rename t1 to t2 ;

** ALTER : Alter statement is used to change the structure of the table.

Syntax to rename the tablename using ALTER statement :

Syntax : Alter table tablename rename to newtablename ;

Ex : Alter table t1 rename to t2 ;

Syntax for adding a column to a table

Syntax : Alter table tablename [Add] columnname [Datatype] ;

Ex : Alter table emp Add Bonus Number (10) ;

Syntax for dropping a column from a table :

Syntax : Alter table tablename Drop column Columnname ;

Ex : Alter table emp Drop column Bonus ;

Syntax for renaming a column name.

Syntax : Alter table tablename Rename Column Columnname To NewColumnname ;

Ex : Alter table emp Rename column Salary to Sal ;

- - - - - - - - - - - - - - - - -

** USER - CONSTRAINTS :

It contains constraints related information of the objects present in the database.

Syntax : DESC USER - CONSTRAINTS ;

Query to check other constraints of a columns of a table.

```
SQL> Select Constraint_Name, Constraint_Type, Search_condition
     from user_constraints
     where Table_Name = 'EMP';
```

Case

* CONSTRAINT_NAME: It is a name given to each and every constraint in the database in order to store each type of constraint in the form of object.

* CONSTRAINT_TYPE: It displays the type of constraints applied on the columns of the table.

* SEARCH_CONDITION: It displays the additional conditions applied on the columns of the table.

* ADDING OR REMOVING CONSTRAINT:

1) PRIMARY KEY:

Syntax: Alter Table Tablename ADD CONSTRAINTS ConstraintName Primary Key (columnName);

Example: Alter Table Empl2 Add constraints CNS1 PRIMARY KEY (EMPNO);

Alter Table Dept2 Add constraints CNSS1 Primary Key (DeptNO);

Table altered.

2) FOREIGN KEY :

Syntax : Alter Table Tablename ADD Constraints ConstraintName
Foreign key (columnName) References TableName (columnName);

Example : Alter Table EMP2 ADD CNS2 FOREIGN KEY (DeptNO)
References Dept2 (DeptNO);

Table Altered.

3) UNIQUE :

Syntax : Alter Table Tablename ADD Constraints ConstraintName
Unique (columnName);

Example : Alter Table Emp2 ADD Constraints UKS UNIQUE (ENAME);

4) CHECK :

Syntax : Alter Table TableName Add Constraints ConstraintName
CHECK (COLUMNNAME, CONDITION (?));

Example : Alter Table Emp2 ADD CONSTRAINTS CHS CHECK (SAL<=7000);

5) NOT NULL :

Syntax : Alter Table Tablename Modify (columnname, NOTNULL/NULL);

Example : Alter Table emp2 Modify (ename, null);

** DROPPING CONSTRAINTS : Query.

Syntax : Alter Table Tablename Drop constraints constraintname

Example : Alter Table empa drop constraints CNS1;

To see whether altered constraints are applied use a query

```
SQL> Select constraint_Name, constraint_type, Search_condition
     from User_constraints
     where table_name = 'EMP2';
```

O/P :

Constraint_Name	C	Search_condition
CNS1	P	
CNS2	R	
CNS3	U	
	C	SAL <= 1000
	C	"ENAME" IS NOT NULL

** DATA MANIPULATION LANGUAGE (DML) :

* CASE EXPRESSION :

Syntax ⇒

```
Select SelectList, CASE
WHEN comparison condition THEN
                    ReturnCondition
:
:
WHEN ..... THEN ....
ELSE ReturnCondition [option]
END AS ALIASNAME FROM TableName
```

→ Case expression provides facility of if. else and then Condition.

→ Case Expression are written inside/within the SQL statement to write the conditional statement.

→ In case Expression Else is optional.

Q. Write a CASE expression to display the message as

→ If Experience is greater than 35 then message is eligible for pension.

→ If the experience = 35, then message is eligible after a year.

→ If the experience < 35, not eligible for pension.

```
Select Ename, Trunc((Sysdate - Hiredate)/365), CASE
WHEN TRONC ((Sysdate - Hiredate)/365) > 35 Then 'Eligible for pension'
When Trunc ((Sysdate - Hiredate)/365) = 35 Then 'Eligible After a year'
else 'NOT eligible for pension'
END AS PensionEligibility. from emp;
```

Q. Display the Salary hike for all the employees based on job.

→ If job is president, then Sal hike of 30%.

→ If job is Manager, then Sal hike of 25%.

→ If job is Analyst, then Sal hike of 20%.

→ If job is Salesman, then Sal hike of 35%.

else sal like of 15%. for rest of all job.

```
Select ename, job, sal "Salary Before Hike",
CASE
When (select count(*) from emp where job = 'PRESIDENT') != 0
then  sal * 1.30
When (Select count(*) from emp where job = 'MANAGER') != 0
then  sal * 1.25
When (Select count(*) from emp where job = 'ANALYST') != 0
then  sal * 1.20
When (Select count(*) from emp where job = 'SALESMAN') != 0
the   sal * 1.15
else  sal * 1.50
END "Salary After Hike"
from emp :
```

25/09/2017

** DML :

* INSERT : Type 1 :

```
INSERT INTO tablename (column1, column2, column3 .....)
                values (value1, value2, value3 .... );
```

Ex:
```
INSERT INTO EMP (sal, Mgr, Hiredate, comm, Ename, DeptNO,
                job, EmpNO)
        values (10000, 1234, '03-DEC-81', 110, 'Sunny Deol', 30,
                Null, 2345);
```

Typ Inserting Data only to 3 columns:

```
Insert into Emp (Empno, Ename, job)
        values  (3456, 'SHAM', 'SALESMAN');
```

NOTE: The above insert statement is used, if the column order is unknown.

→ Using this INSERT statement, we can insert data into Selected columns or all columns of the table.

→ To insert a NULL value into a column, make use of "NULL" Keyword.

Type 2: INSERT INTO tableName values (value1, value2, value3...)

Here we need to pass data to all columns of the table else it throws an error.

EX: Insert into Dept values (50, 'AAB', 'MLORE', NULL); ← Address

NOTE: If the column Order is known, we can use the above Insert statement to insert the record into the table.

But by using this approach of inserting data we have to pass values to the reference/column of the table

---

Type 3 : INSERT into Dept tablename values (&Deptno, &Dname, &LOC); &ColumnName, &ColumnName ....

Enter value for deptno : 50

Enter value for Dname : ERRT

Enter value for LOC : HASSAN

Ex: Insert into Dept values (&Deptno, &Dname, &LOC);

NOTE: To write insert statement once and insert multiple records we can use above INSERT statements.

ALL DML statements are temporary NOT Permanent

** DELETE :

Delete statement can delete single, multiple or all records from a table.

Syntax : Delete from table-Name
where condition (?)

Example : Delete from emp
where empno = 7369;

1 Record Deleted

Delete from emp
where deptno = 30;

6 Record / Multiple Deleted

Delete from emp;

All records Deleted

Q. Delete all the records of the employee whose name start with S & A.

```
Delete from emp
where ename LIKE 'S%' OR ename LIKE 'A%';
```

Q. Delete the records of the employee whose job is same as ADAMS job.

```
Delete from emp
where job = (Select job from emp where ename = 'ADAMS');
```

Q. Delete the records of the employee whose HIREDATE is same as the Hiredate of an employee whose job is 'CLERK'.

```
Delete from emp
where HIREDATE IN (Select Hiredate from emp where job = 'CLERK');
```

---

* Difference b/w Truncate And Delete.

TRUNCATE	DELETE
→ Truncate statement deletes all the records at a time.	→ Delete statement can delete one, multiple or all records of a table.
→ It releases the memory space back to the database.	→ It never releases the memory space back to the database.
→ It can never rollback.	→ It can rollback.
→ It is a DDL Statement	→ It is a DML Statement.

** UPDATE :

Syntax : Update tableName
SET expression
where condition(?) ;

Ex① : Update emp
Set Sal = 3000.
where job = 'MANAGER';

Update statement can update one record, multiple records or all records of the table.

Q: Update salaries of all the employees if their Loc is NEWYORK.

```
UPDATE emp, dept
Set sal = 3000
where deptno = (Select deptno from dept where LOC = 'NEWYORK');
```

Ex 2 : To update all the employees record.

```
Update emp set Sal = 5000;
```

Ex 3 : To update Multiple Columns at a time.

```
Update Emp
Set Hiredate = '04-DEC-81', SAL = 9000.
    where ename = 'KING'.
```

NOTE :

Using update statement, we can update multiple fields as shown in above Ex3.

Q: Update the salary with the maximum salary for those employees whose Hiredate matches with the Hiredate of a employee who is getting max salary.

```
Update emp
    Set Sal = (Select Max(Sal) from emp)
        where Hiredate IN (Select Hiredate from emp
                           where Sal = (Select Max(Sal) from emp));
```

→ If we have not closed the bracket here itself it updates all the records Sal column.

** TRANSACTION CONTROL LANGUAGE → Commit, Savepoint, Rollback.

26/09/17

Transaction : It is a process of business logic which is used to perform a particular task.

** Transaction Control Statements are used to manage the changes made by DML statements.

In SQL Insertion, deletion, or updation are known as SQL Transactions.

* COMMIT : Used to save the transaction or Work done.

* ROLLBACK : It restores the database to the original state, since the last commit.

* SAVEPOINT : It identifies a point in a transaction to which you can rollback later

Example for COMMIT :

SQL> insert into dept values (60, 'stret', 'sffsaq');

1 row created

SQL> commit; commit completed. (To save the data permanently in DB).

Example for SAVEPOINT :

SQL> delete from emp where empno in (7369, 7499);

2 rows deleted.

SQL> Savepoint a;

Savepoint created

SQL> delete from emp where deptno = 10;

3 rows deleted

SQL> Savepoint b;

Save_point created

SQL> delete from emp where deptno = 20;

2 rows deleted

SQL> Savepoint c;

Save point created.

SQL> rollback to b ;

Rollback complete

**

SQL> rollback ;

↳ It completely returns all the records no matter what rollback or Savepoint is done.

** DATA CONTROL LANGUAGE

Data Control language is used to control the privelige [access or permission] in database to perform any operations such as creating the table views, sequences etc.

* GRANT : It is used to give the permission from one database user to another database user.

* REVOKE : It is used to take back the granted access or permission from the database user.

---

* Creating a User :

Create user Username identified by password ;

SQL > Connect system/tiger

Connected

Go to SuperUser & Create the User.

SQL> Create user SACHIN identified by tiger ; / User created.

** SCHEMA

→ It is nothing but users/partitions in the database.

→ It is a dedicated memory space, where we can create our own database objects.

After creating the object / User

SQL> grant all privileges to Sachin.

Grant succeeded.

** SYNTAX OF REVOKE & GRANT:

Grant privileges ON < database object > to < database User >;

Ex: Grant Select ON emp to hr;

Revoke privileges ON < database object > from < database User >;

Ex: Revoke Select ON emp from hr;

* Example for Multiple Grant access

SQL> Grant Select, delete ON dept to hr;

Grant succeeded.

SQL> Select * from scott.dept;

SQL> Insert into Scott.dept values (60, 'ABC', 'PQR');

Insufficient privileges → bcoz only select & delete privilege is given.

SQL> delete from Scott.dept where deptno = 50;

1 row deleted.

connect Scott/tiger.

SQL> Revoke select, delete dept to hr;

Revoke succeeded;

- - - - - - - - - - - - - - -

SQL> Grant all on emp to hr; → Granting all permission

SQL> revoke select, delete on emp from hr; → Taking few perm back

SQL> revoke all on emp from hr; → Taking all perm back.

- - - - - - - - - - - - - - -

NOTE:

Even though DML Commands are not permanently affected in the database. If we use DML commands from another

* COMPOSITE PRIMARY KEY (CPK)

A primary key which is having more than one column is known as CPK OR

A key which is having more than 1 P.K column in a table is known as CPK.

* Creating a Composite primary Key.

```
SQL> create table t1 (
       fNM varchar (10),
       lNM varchar (10),
       gender char (1),
       Qualification varchar(10),
       primary key (fNM, lNM));
```

Table created.

fNM	LNM	Gender	Qual
Sunny	Srinidhi		
Sunny	S		
Chetan	A		
Chetan	S		

Both together results in UNIQUE.

```
SQL> insert into t1 values (&fNM, &lfNM, &Gender, &qualification);
Enter value for fNM: 'SUNNY'
Enter value for lNM: 'SRINI'
Enter value for Gender : 'M'
Enter value for Qualification : 'BE'
1 row created.
```

** NORMALIZATION :

It is process of efficiently organizing the data in the database is known as Normalization

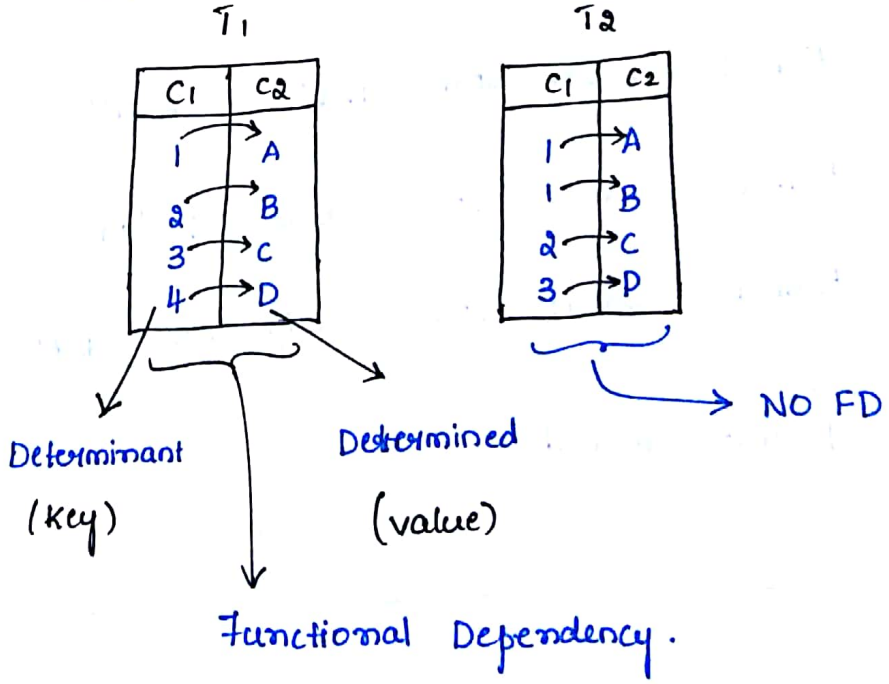
OR

It is the process of dividing the bigger tables into smaller table.

The main goal of the Normalization is to reduce the data redundency with the database.

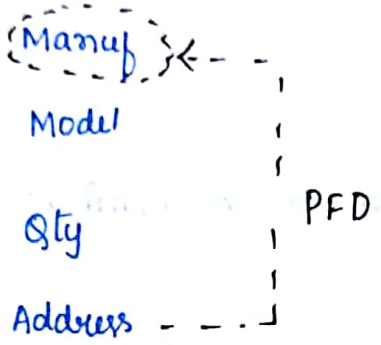
* FUNCTIONAL DEPENDENCY :

The value in one column should have only one asso-ciated value in the adjacent value is known as functional Dependency.



* Partial Functional Dependency :-

If the Non-prime Attributes depends on part or portion of Prime attributes is known as Partial functional dependency.



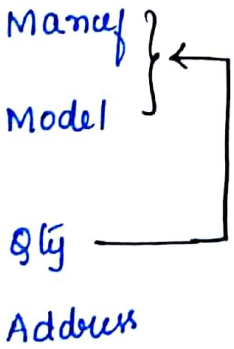
PK OR Composite P.K

Manufac	Model	Qty	Addr
Sam	TV	10	S.K
Sam	Mobile	30	S.K
App	TV	20	US
App	Mobile	10	US
Sony	Mobile	40	JP

Prime Attributes — Non-prime Attributes.

* Fully Functional Dependency :

If the Non-prime Attributes depends on complete prime attributes is known as Fully functional Dependency.



* Transitive Functional Dependency :

In a relation, if the Non-prime attributes indirectly depend on prime attributes or another non-prime attributes it is known as Transitive functional Dependency.

* UN-NORMALIZED TABLE :

Sl.No	City	Status	Phone	Qty.
1	Blr	30	P_1 P_2	10, 20
2	Mum	20	P_2 P_3	20, 30
3	Del	10	P_3 P_4	30, 40
4	Blr	30	P_3 P_4	40, 50

1. NF → 1st Normal Form says, there must be no multi value attributes in a relation OR

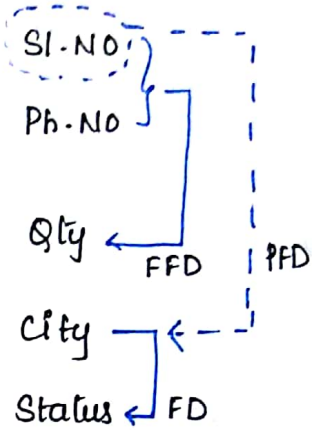
Each attribute in a relation must have atomic (→ Single) value.

After 1st N.F ⇩

Sl.NO	City	Status	Phone	Qty
1	Blr	30	P_1	10
1	Blr	30	P_2	20
2	Mum	20	P_2	20
2	Mum	20	P_3	30
3	Del	10	P_3	30
3	Del	10	P_4	40
4	Blr	30	P_3	40
4	Blr	30	P_4	50

PA | NPA

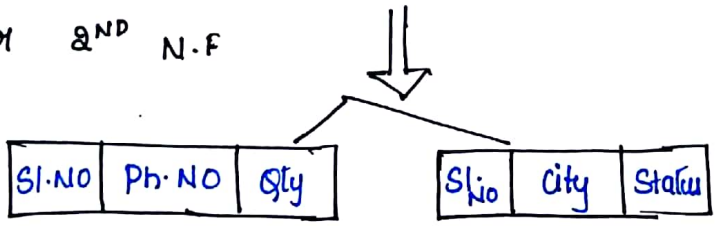
Sl.NO	PhNO	City	Status	Qty
1	P_1	B	30	10
1	P_2	B	30	20
2	P_2	M	20	20
2	P_3	M	20	30
3	P_3	D	10	30
3	P_4	D	10	40
4	P_3	B	30	40
4	P_4	B	30	50



2ND N.F : 2nd Normal Form follows 1st N.F.

→ It says, in a relation, there must be no partial function dependency.

After 2ND N.F



3rd Normal Form : It follows a 2nd Normal Form.

It says, there must be no transitive functional dependency.

After 3rd N.F