SQL

Database:- It is a medium where we can store the data in the organised and systematic manner.

DBMS:- It is a software which is used to maintain and manage the database.

Basic Operations done on a Database:-

1)Create 2) Read 3) Update 4)Delete (CRUD)

These operations are referred as CRUD operations.

How to Manage the Database?

1) We use query language to communicate or interact with DBMS.

2) Security and Authorization are the two imp features provided by DBMS.

Entity:- It is a substance which has its existence in real world.

Eg;- Student.

Attribute :- Properties that describes an entity.

Eg;- SID,SNAME.

RDBMS:-

1) It is a type of DBMS software which is used to store the data in the form of relations or tables.

2) Any DBMS that follows relational model is known as RDBMS.

3) We use SQL to communicate with RDBMS.

Relational Model:- It was designed by E.F Codd.

A Relational Model is used to arrange the data in the form of

relations or tables including metadata.

List the difference between Database and DBMS

List the difference between DBMS and RDBMS

List the difference between Relational Model and RDBMS

Table:- Collection of columns and rows. Also known as an entity.

Column :- It is also called as an attribute/field.

Row :- Row is also called as records/tuples.

Cell:- Cell is the intersection of row and columns.

Rules of E.F Codd:-

1) The data that is stored in the cell must be single value data.

2) According to him we can store the data in multiple tables If needed we can establish a connection between two tables using key attributes.

3) We can validate the data entering into the table in two steps

By using data types or by using constraints.

Data types are mandatory whereas constraints are optional.

DATA TYPES :-

It is used to specify the type of the data that we are going to store in a particular memory location.

DATA TYPES in SQL:-

1) CHAR 2)VARCHAR /VARCHAR2 3)NUMBER 4)DATE 5)LARGE OBJECT

CHAR :- It Is a data type that can accept ‘A-Z’ or ‘a-z’ or any special characters in single quotes.

Whenever we use char data types we have to mention the size for it.

Syntax :- CHAR(Size) Eg- Char(10) Store any character of size under 10 if used they are called used memory and if not un-used.

1) Char data type can accept about 2000 characters.

VARCHAR :- It is similar to char but they are variable length data types.

Data gets allocated at compile time for char but dynamically for VARCHAR.

So there is a memory loss for CHAR but not for VARCHAR.

VARCHAR2 :- It is an updated version of VARCHAR in which it can accept 4000 characters.

NUMBER :- Number data type can accept 2 arguments i.e., Precision and Scale.

Syntax :- NUMBER(PRECISION,SCALE).

Precision :- It determines the number of digits required to store in the value .Eg:- NUMBER(5)

It generates a cell of size 5 with index starting from 1 to 5.

The maximum precision can be 38

SCALE :- It is used to determine the number of digits required to store the decimal value in the precision.

Eg :- NUMBER(5,2) If P>S it can store 999.99 with scale being 2.

If P==S It can 0.99999 with scale being 5.

If P<S it can store up to the value but fills the unfilled Precision

values with 0’s.

The maximum scale can be 127.

DATE Data Type :- It is used to store the date.

The Oracle Specified date format are ‘DD-MON-YY’ or ‘DD-MON-YYYY’.

Syntax :- DATE.

LARGE OBJECT:- They are used to store values up to 4Gb of size.

1) CLOG – Character Large Object :- Used to store characters up to 4GB.

2) BLOB – Binary Large Object :- Used to store binary values of image,mp3,mp4 up to 4GB of size.

CONSTRAINTS:-

NULL :- nothing or empty .Null Is a key word used to represent empty cell or null table.

Characteristics :- Null does not represent 0 or space.

It does not consume memory. We can not equate 2 nulls. Any arithmetic operations performed on null will result in null itself.

Constraints :- They are the extra validation that are provided to the columns to validate the data .

TYPES OF CONSTRAINTS :-

UNIQUE :- It is assigned to a column that cannot accept duplicate values or repeated values.

Eg ;- ID,AADHAR NO,PHNO.

NOT NULL :- It is assigned to a particular column which is mandatory.

Eg :- ID,NAME,Percentage,YOP,DOB……

CHECK :- Check constraints is an additional validation given to a column with the condition . If the condition Is satisfied then the value is accepted else rejected .

Eg :- CHECK (age >21) , CHECK(marks > 60).

PRIMARY KEY :- A P.K constraint is used to identify the record uniquely from the table.

We can have 1 primary key in a table.

It does not accept duplicate values. It does not accept null.

It is always a combination of UNIQUE and NOT NULL.

NOTE ☺ Primary key is not Mandatory but it is recommended to have one.

Foreign Key:- They are used to establish a connection between two tables . We can have any number of foreign keys in a table.

Foreign key can accept duplicate , null values. It is not a combination of Unique and Not Null.

Foreign Key is present in child table. But actually belongs to parent. It is also known as REFERENTIAL INTEGRITY CONSTRAINT.

Ass:-1) Diff between P.K and F.K

Overview of SQL Statements :-

DQL (DATA QUERY LANGUAGE)

DDL (DATA DEFINITION LANGUAGE)

DML (DATA MANIPULATION LANGUAGE)

DCL (DATA CONTROL LANGUAGE)

TCL (TRANSACTION CONTROL LANGUAGE)

DATA QUERY LANGUAGE :- In SQL we have four statements

1) SELECT 2) Projection 3) Selection 4)JOIN

SELECT : This statement is used to retrieve the data from the table and display the result.

Projection :- The retrieval of data by selecting only the column .Here all the records present in the column are by default selected.

Syntax :- SELECT * / Col_Name / Expression/Alias From table_name;('/'<- symbol for or)

Selection : The retrieval of the data by selecting by both column as well as rows .

JOIN :- The retrieval of the data from multiple tables is known as join.

Dual is a dummy table used for arithmetic calculations.(Eg :- 2*3)

Instead of performing an Expression operation on a defined table one can use Dual table.

Order of Execution :-

From clause starts the execution.

For from clause we can pass table name as an argument.

From clause is used to go to the database, search for the table and put the table under execution .

SELECT :-

It executes after the execution of From clause.

For select clause we can pass * or Column name or Expression.

It goes to the table which is under execution and select the columns

It is responsible for repairing the result table.

NOTE ☺ '*' means to select all the column names along with records.

It has to be used as the first argument and the only argument for select clause.

';' - It is used to determine the end of the query or statements.

If the column name is incorrect in the select clause the error given by the compiler is invalid identifier.

With the table names incorrect in the clause the compiler gives an error (Table does not exist).

Eg :- SELECT * FROM STUD;(If Stud is not created)

Example :- Invalid Indentifier :- SELECT SID From EMP;

SELECT ENAME FROM STUDENT;

DISTINCT KEYWORD :- It is used to remove the duplicate records present in the result table.

It has to be used as first argument to the Select clause.

For Distinct keyword we can pass multiple column names as an argument.

It removes the combination of the column in which the records are duplicated .

QUERIES :- Write a query to display all the details of the student .

Select * from Student;

Write a query to display all the names of the student.

Select Sname from Student.

Write a query to display name and percentage of the students

Select Sname,Percentage from Student.

Write a query to display student percentage by deducting 2% of it.

Select percentage -2 from students .

Write a query to display branch and percentage from Student.

Select Distinct Branch , Percentage from Student.

Write a query to display all the names of employee table.

Select Ename from Employee;

Write a query to display name and salary given to all the emloyees.

Select Name, Salary from Employee.

Write a query to display Ename and Annual sal given to all the employees.

Select Ename,Sal*12 from Employee

Write a query to display name and designation of all the employees.

Select Ename,Job from Emloyee.

Write a query to display name and Half term sal of all the employees.

Select Ename,Sal*6 HALFTERMSAL from Employee.

Write a query to display Ename and Salary and also Salary with

25 % hike.

Select Ename,Salary ,Sal+sal*0.25 from Employee.

Write a query to display Ename ,sal and Sal with 12 % deduction.

Select Ename,Sal ,Sal –sal*.12 from Employee.

Write a query to display all the details of the employees along with annual salary.

Select emp.*, sal*12 from AnnualSal from emp;

Write a query to display details of the employees along with annual bonus of 2000.

Write a query to display total salary given to each employee.

Select Sal+ Comission TotalSal from Dual;

Where clause :- Syntax Where <filter_condtion>

It is used to filter the condition. It executes row by row.

CONCATINATION OPERATOR :-

☺ IT is used to join two strings or two column of the string.

←----------☺☺☺☺----------→

DATE :- 30-03-2019

OPERATORS : LOGICAL OPERATOR.

AND , OR AND NOR.

AND OPERATOR :- It is used between conditions and it selects the records only if the conditions are satisfied else it rejects or discards the records.

Example:- Write a query to display details of the employees if the employee sal is 3000 in deptno 20.

OR OPERATOR :- It Is used whenever any of the condition has to be true. If one of the condition is true then it selects the records.

Example :-

Write a query to display name and job of all the emps working as manager or analysts.

☺ Select ename,job from emp where (job='MANAGER' or job='ANALYST') ;

Write a query to display all the details of the emp working as manager or clerk in deptno 20;

☺ Select * from emp where (job='MANAGER' or job='CLERK') and deptno=20;

Write a query to display where sal <1100 and job=clerk;

☺ Select * from emp where sal<1100 and job='CLERK';

Write a query to display name,sal,annual sal and deptno if deptno is 20 having more than 1100 and annual salary is more than 12000

☺ Select ename,sal,sal*12 ANNUALSAL ,deptno from emp where deptno=20 and sal>1100 and ANNUALSAL >12000.

Write a query to display details of the employee enames working in deptno 10,20,30,40.

☺ Select ename ,deptno from emp where deptno in (10,20,30,40);

Write a query to display name and designation of the employees working as manager,salesman,clerk and analyst.

☺ Select ename,job from emp where job

in('MANAGER.'SALESMAN','CLERK','ANALYST');

Write a query to display ename and annual sal as the employees whose salaries are 1100 3000 950 in emp table.

☺ Select ename ,sal*12 ANNUALSAL from emp where sal in(1100,3000,950);

SPECIAL OPERATORS :-

IN OPERATOR :- This operator is a multi-valued operator which can accept multiple values at the RHS. It is used in evaluating multiple values. It is nothing but '=' operator. It is used to compare one value of LHS with all values of RHS. If one of the condition satisfies it returns and select the records.

Syntax:- col_name/expression in(Val1,Val2,Val3,.....);

Example:- Sal in (1000,2000,3000)

3000 in (1000,2000,3000);

Write a query to display name and job of the emp working as manager or analyst.

☺ Select ename,job from emp where job='MANAGER' or job='ANALYST';

Write a query to display ename job and deptno to all the employees working as clerk in deptno 10 or 30.

☺ Select ename,job,deptno from emp

NOT IN OPERATOR :- It is an operator which rejects the record instead of selecting it.

Syntax :- col_name/expression NOT IN (Val1,Val2,Val3,……)

Example :- Sal not in(1000,2000,3000)

3000 not in(1000,2000,3000)

Write a query to display name of the emp and dept no excluded and employees of the dept no 20 and 10.

☺ Select ename,deptno from emp where deptno not in(10,20);

Write a query to display all the details of the emp working as manager or clerk in any dept excluding deptno 20.

BETWEEN :- It is used to whenever we have range of values.

It works including the range.

Syntax :- col_name/expression between lower_range and higher_range.

Write a query to display name and sal only if the emp sal starts from 1250 and ends at 3000.

Write a query to display ename and the hiredate if the emp was hired during 1981.

NOT BETWEEN :- It is similar to between operator but rejects the values instead of selecting it. It works excluding the range.

Syntax :- col_name/expression not between lower_range and higher_range.

Write a query to display details of the emp excluding the employees hiring from 1981.

IS OPERATOR :-

It is a special operator that we used to verify whether the value present at LHS is null.

Syntax :- Col_name/Expression is NULL.

NOT IS OPERATOR :- It is used to verify whether the value present at LHS is not null.

Syntax :- Col_name/Expression not is NULL.

LIKE OPERATOR :- It is an operator used for pattern matching. It is done through two ways '%' and '_'.

'%' is used to match more than one characters.

'_' is used to match only one character.

Syntax :- Col_name/expression like '%','_'.

Example :- Select ename from emp where ename like '_D%';

NOTE ☺☺ Escape :- This character is used to remove special behaviour given to a special character.

Syntax :- 'pattern_to_match' ESCAPE 'char';

Example :- Select ename from emp where ename like('%!_%') ESCAPE '!';

Here we have used !<- we can use any special character.

FUNCTIONS :-

Block of codes or Set of instructions used to perform a specific task.

Major components of a function are :-

1) Function Name.

2) Type of Arguments.

3) Return type.

Types of functions in SQL

1) Single row

2)Multi-row /Aggregate/Group Functions.

SINGLE ROW FUNCTIONS :-

In this function an input is taken it is processed and executed and the output is given and then the second input is processed.

If we pass ‘n’ number of inputs to a single row functions we get ‘n’ number of outputs.

Example :- Select length(‘TIGER’) from dual ;

Output :-5

Invalid arguments Example :- Select length(‘TIGER’,’LON’) from dual;

Output :- Invalid number of arguments.

MULTI-ROW FUNCTIONS :-

These functions aggregate all the input at once executes it and returns a single output.

If we pass ‘n’ number of inputs only one output is given.

TYPES OF MULTI-ROW FUNCTIONS :-

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

NOTES ☺☺

1) These functions can accept only one argument at a time.

2) We can pass column name as an argument for a multi-row

function however we cannot write a column name along with the multi-row function.

Eg :- Select MAX(ename) ,ename can’t be written as its not a structured table.

3) We can use any number of multi-row functions together.

4) These functions cannot be nested. (MAX(MIN)) <- WRONG

5) These functions ignore null values.

6) We cannot use MRF in where clause.

←----------☺☺☺☺----------→

DATE:- 31-03-2019

ORDER OF EXECUTION:- 1)FROM ->2) WHERE ->3) GROUP BY ->4) SELECT.

GROUP BY CLAUSE:-

It is used to execute row by row . In group by clause we can write only name or expression. The col_name/expression used in group by clause can only be written in select clause. After the execution of group by clause we get group of output.

Write a query to display no of employees in each dept.

Write a query to display no of employees in each dept excluding the manager.

Write a query to display max salary given to the employees of each job excluding the employees whose salary Is less than 2100.

Write a query to display number of clerks working in each dept.

Write a query to display number of employees working in each

dept except president.

Write a query to display total sal needed to pay all the emp in each job.

Write a query to display number of emp working as manager in each dept.

Write a query to display avg sal needed to pay for all the employees in each dept excluding the employees of deptno 20.

Write a query to display number of emp having character ‘a’ in there names in each job.

Write a query to display total sal needed to pay and number of salesman in each job.

HAVING CLAUSE :-

Order of Execution :-

{

FROM

WHERE

GROUP BY

HAVING

SELECT }

☺ This clause Is used to filter the groups.

Syntax :- Select group function

From table_name

Where <Filter_conditon>

Group by col_name/espression

Having <Group by Filter condition>

Write a query to display deptno and number of employees working in each dept if there are two employees in each dept.

☺ select deptno,count(*) from emp group by deptno having count(*) >2

Write a query to display name of all the employees if the name is repeated.

☺ select ename,count(ename) from emp group by ename having count(ename) >1;

Write a query to display the repeated salary from the employee table.

☺ select sal,count(sal) from emp group by sal having count(sal) >1

Write a query to display the duplicated dates from employee table

☺ select hiredate,count(hiredate) from emp group by hiredate having count(hiredate) >1

Write a query to display deptno and number of emp working in each dept if there are atleast two clerks in each dept.

☺select deptno,count(*) from emp where job='CLERK' group by deptno having count(*)>=2

Write a query to display deptno and total salary needed to pay

all the employees in each dept if there are atleast 4 employee in each dept.

☺ select deptno,sum(sal),count(*) from emp group by deptno having count(*)>=4

Write a query to display no of employees working in each dept along with the total sal if the total sal of each dept is greater than 6000.

☺ select deptno,count(*),sum(sal) from emp group by deptno having sum(sal)>6000

Write a query to display dept number and number of employees working only if there are 2 employees working in each dept as manager.

☺ select deptno,count(*) from emp where job='MANAGER' group by deptno having count(*)=2

Write a query to display job and max sal of emp in each job if max sal excludes 2600.

☺ select job,max(sal) from emp group by job having max(sal) >2600

Write a query to display the salary which are repeated in emp table.

☺ select sal,count(sal) from emp group by sal having count(sal) >1

Write a query to display avg salary of each dept if avg sal is < 3000.

☺ select deptno,sal,avg(sal) from emp group by deptno,sal having avg(sal)<3000

Write a query to display hiredate which are duplicated In emp table.

☺ select hiredate,count(hiredate) from emp group by hiredate having count(hiredate) >1

Write a query to display deptno if there are atleast 3 employees in each dept whose name has character ‘a’ or ‘s’.

☺ select deptno,count(*) from emp where ename like ('%A%') or ename like ('%S%') group by deptno having count(*)>=3

Write a query to display min and max sal of each job if min sal > 1000 and max sal < 5000.

☺ select job,min(sal),max(sal) from emp group by job having min(sal) > 1000 and max(sal) <5000

List the difference between where and having clause.

ORDER BY CLAUSE :-

Syntax :- Select col_name

From table_name

Where <filter_condition>

Group by col_name/expression

Having <Group by filter_condition>

Order By Sorting Techniques

Order of Execution :-

{

FROM

WHERE

GROUP BY

HAVING

SELECT

ORDER BY }

This clause executes after the execution of all the clauses. It has to be written as last statement in the query. For this clause we can pass column name or expression to sort the records. By default this clause sorts the records in ascending order. We can use ALIAS names in this clause but we cannot assign or change the ALIAS names.

Write any ten egs on order by clause.

Write a query to display to order the enames in the table.

☺ select ename from emp order by ename

Write a query to display ename and order the hiredate in the emp table.

☺ select ename,hiredate from emp order by hiredate.

Write a query to display the job of the employees in an order.

☺ select ename,job from emp order by job

Write a query to display the details of an employee with odering of ename.

☺ select * from emp order by ename

Write a query to display the enames according to the order of sal.

☺ select ename,sal from emp order by sal desc.

Write a query to display the enames and hiredate with ordering of enames desc.

☺ select ename,hiredate from emp order by ename desc

CONDITION 2 for sub-query:-

Whenever the data to be selected and condition to be executed are present in two different tables then we use sub-query method.

Example :- Select dname from dept where dept = (select dno from emp where ename=’A’);

Write a query to display the deptname of MILLER.

☺ select dname from dept where deptno = (select deptno from emp where ename='MILLER')

Write a query to display details of the employees working in research dept.

☺ select * from emp where deptno = (select deptno from dept where dname='RESEARCH')

Write a query to display loc of the employees with empid 7902.

☺ select loc from dept where deptno =(select deptno from emp where empno=7902)

Write a query to display number of employees hired after smith into dept ‘Accounting’.

☺ select count(*) from emp where deptno = (select deptno from dept where

dname='ACCOUNTING' and hiredate > '17-DEC-80') group by deptno

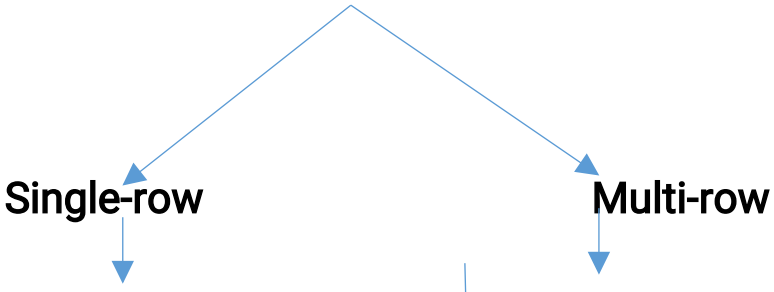
OR

select count(*) from emp where hiredate > (select hiredate from emp where ename ='SMITH') and deptno = (select deptno from dept where dname ='ACCOUNTING');

Write a query to display dname of SMITH and KING.

☺ select dname from dept where deptno in (select deptno from emp where ename in('SMITH','KING'))

TYPES OF SUB-QUERIES



A sub-query that returns exactly one Record. If then we can use any type Of operator to write the condition.

If the sub-query returns more than one record then its multi row sub-query. So we cannot use normal operators we have use special operators like (IN,NOT IN)

Select * from emp where deptno = (select deptno from dept where dname ='ACCOUNTING');

select * from emp where deptno = (select deptno

from emp where empid=7902); from emp where sal>2000);

ALL and ANY OPERATOR :-

ALL operator is a SQ operator that is used along with relational operator. It is used to compare the value of LHS with all the values present at RHS. If all the values at the RHS satisfy the condition then only all operator returns true.

Syntax :- Col_name Relational Operator ALL (Val1,Val2,Val3,….)

Example :-

Write a query to display details of the emp whose salary is greater than sal of all the salesman.

☺ Select * from emp where sal > ALL (select sal from emp where job ='SALESMAN')

ANY OPERATOR

It is used along with relational operator .It is used to compare the values of LHS with all the values present at RHS.

If any of the value at RHS satisfy the condition then any operator returns true.

Write a query to display name and sal of the emp who are having salary < SALESMAN.

☺ select ename,sal from emp where sal < ALL (select sal from emp where job ='SALESMAN')

Write a query to display all the details of the emp if there sal > atleast a manager.

☺ select * from emp where sal > ANY (select sal from emp where job='MANAGER')

Write a query to display name and hiredate of the emp hired before the emp of the dept 10 .

☺ select ename,hiredate from emp where hiredate < ALL (select hiredate from emp where deptno =10)

Write a query to display name and hiredate along with annual sal if hired after president into deptno = 20.

☺ select ename,hiredate,sal*12 ANNUALSAL from emp where hiredate > ALL (select hiredate from emp where job = 'PRESIDENT' and deptno in (20))

Write a query to display all the details of the emp whose annual sal is more than emp of dept 10.

☺ select * from emp where sal*12 > ANY (select sal*12 from emp where deptno=10)

Write a query to display number of emp having sal < atleast 'CLERK'.

☺ select count(*) from emp where sal < ANY (select sal from emp where job='CLERK')

NESTED SUB-QUERY :-

A sub –query written inside another sub-query is called as nested sub-query.

We can nest about 255 sub-queries.

Write a query to display 7th max sal .

☺ select *

from

(select

```
    sal,dense_rank() over (order by sal asc) ranking
  from   emp)  where ranking = 3
```

Write a query to display third min sal.

☺ select *

```
 from
 ( select
    sal,dense_rank() over (order by sal asc) ranking
  from   emp)  where ranking = 7
```

Write a query to display name of the employees earning 2nd max sal.

☺ select *

```
 from
 ( select
    Ename,sal,dense_rank() over (order by sal asc) ranking
  from   emp)  where ranking = 2
```

Write a query to display getting name of the emp third min sal.

☺ select *

```
 from
 ( select
    sal,dense_rank() over (order by sal asc) ranking
  from   emp)  where ranking = 3
```

Write a query to display details of the emp earning 2nd max sal.

☺ select *

from

(select

sal,dense_rank() over (order by sal asc) ranking

from emp) where ranking = 2

Write a query to display dname of the emp for earning minimum sal.

☺ Select dname

Employee Manager Realtionship.

Write a query to display manager name of BLAKE.

☺ select ename from emp where empno = (select mgr from emp where ename='BLAKE')

Write a query to display MILLER'S Manager details.

☺ select * from emp where empno = (select mgr from emp where ename= 'MILLER')

Write a query to display Salesman managers name.

☺

Write a query to display manager's manager name of MARTIN

☺ Select ename from emp where empno = (select mgr from emp where empno = (select mgr from emp where ename='MARTIN'));

Write a query to display name of the employees reporting to KING.

☺ select ename from emp where mgr = (select empno from emp where ename= 'KING')

Write a query to display name of the employees reporting to analyst.

☺ select ename from emp where empno = (select mgr from emp where job=’ANALYST’);

Write a query to display location of the emp reporting to salesman.

☺ select loc from emp from dept where deptno = ALL (select deptpno from emp where job =’SALESMAN’);

Write a query to display number of employees reporting to salesman’s manager

☺ select count(*) from emp where mgr in (select mgr in emp where job=’SALESMAN’);

Write a query to display name of an employee who are reporting to Blake manger.

☺ select ename from emp where mgr = (select empno from emp where ename='BLAKE')

Write a query to display dname of emp who are reporting to Miller’s manager.

☺ select dname from dept where deptno in (select deptno from emp where mgr =(select mgr from emp where ename='MILLER'))

Write a query to display name and hiredate of the emp hired after the first employee.

☺ select ename,hiredate from emp where hiredate > (select min(hiredate)from emp);

Write a query to display name and hiredate of the emp if the emp was hired on the next day of the first emp.

☺ select ename,hiredate from emp where hiredate > (select

min(hiredate)from emp);

Write a query to display details of the emp hired after the month of first emp.

☺ select ename,hiredate from emp where hiredate > (select min(hiredate)+30from emp);

Write a query to display number of emp hired after 2nd emp.

☺ select ename from emp where hiredate >(select min(hiredate) from emp where hiredate >(select min(hiredate) from emp))

Write a query to display number of emp hired on the same day.

☺ select ename from emp where hiredate in

(select hiredate from emp group by hiredate having count(hiredate)>1)

Write a query to display name of the emp if the emp is getting same sal

☺ select ename,sal from emp where sal in (select sal from emp group by sal having count(sal) >1);

SINGLE ROW FUNCTIONS :-

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

Syntax :- upper('STRING');

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

Syntax :- lower('STRING');

3) Initcap() :- It is used to convert first letter of the given string to uppercase and rest of the character or letter in lower case.

Syntax :- Initcap('String');

4) Reverse() :- This function Is used to reverse the given String.

5) Length() :- It is used to obtain the length of given string.

Write a query to display the name of the employees in lower case if emp getting 4 digit sal.

☺ select lower(ename) from emp where length(sal)=4;

Write a query to display job in reverse format when ename has 6 characters.

☺ select reverse(job) from emp where length(ename)=6;

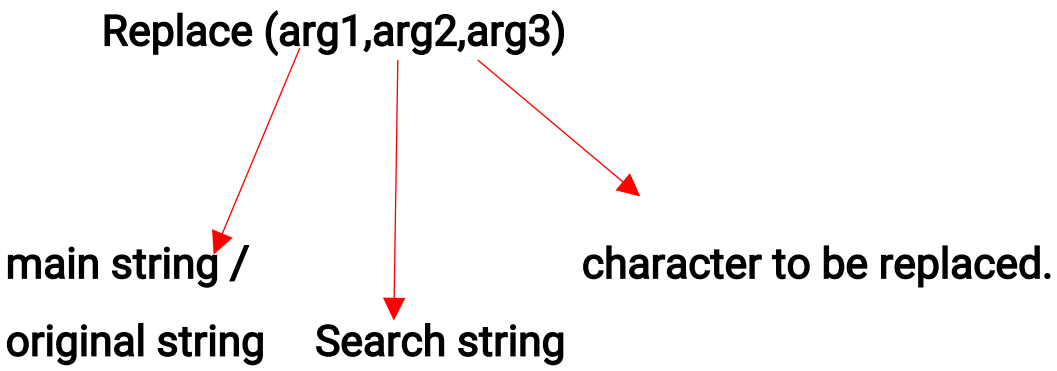
CONCAT FUNCTION ():- similar to concat operator .

Syntax :- concat(arg1,arg2)

Example : concat('HI ' ,ename) .

Write a query to display following Hi smith your sal is and workind in deptno .

REPLACE FUNCTION :-



Example : Replace('Bangalore', 'A','*')

It searches for a search string in a main string and replaces it with third arguments data.

Write a query to display to replace a with * in each ename.

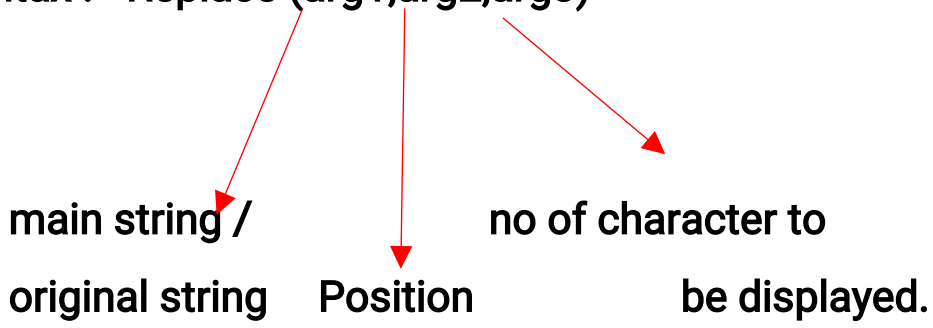
☺ select replace(ename,'A','*') from emp.

Write a query to display to replace f with * and a with & in each job.

☺ select replace(replace(job,'S','*'),'A','&') from emp.

SUBSTRING () :- Used to retrieve a sub-string of a given string.

Syntax :- Replace (arg1,arg2,arg3)



It is used to extract a part of the string.

Write a query to display the name of the employees if his name starts with vowel

☺ select ename from emp where substr(ename,1,1) in ('A','E','I','O','U');

Write a query to display the first half of emp name.

☺ select substr(ename,1,length(ename)/2) from emp;

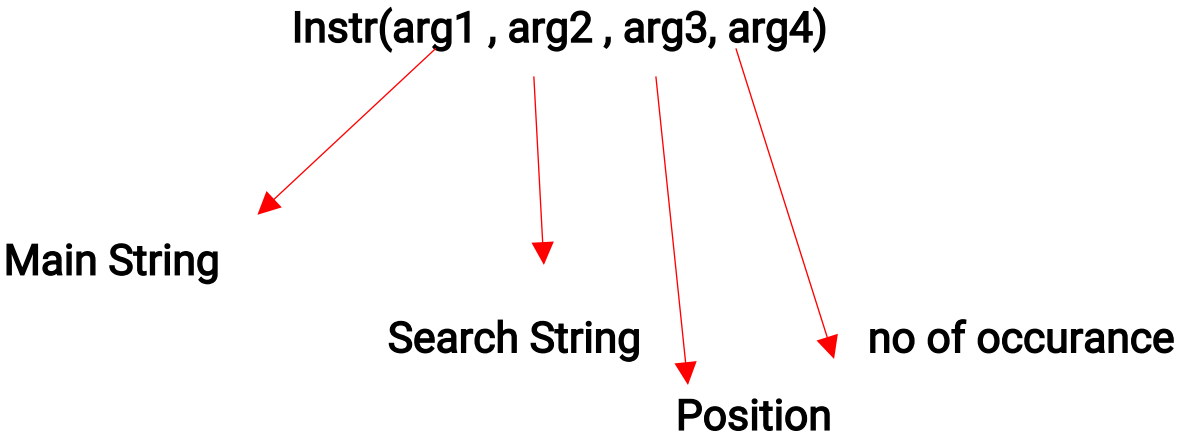
Write a query to display to get first and last character.

☺ select substr(ename,1,1) || substr(ename,-1,1) from emp

Write a query to display ename and job if the last 3 char of job ends with man or erk.

☺ select ename, job from emp where substr(job,-3) in ('MAN','ERK');

INSTR() :-



Searches the string from a main string with inclusives of position and occurances and returns the position at the which the string is seen.

NOTE ☺ In instr the string traversal can take place from right to left.

Write a query to display first occurance of a in each name.

☺ select instr(ename,'A',1,1) from emp.

Write a query to display only the name of the empl starting with a using instr

☺ select ename from emp where substr(ename,'A,1,1) =1;

Write a query to display e as last but first using instr

☺ select ename from emp where instr(ename,'E',1,1) =-1

Write a query to display dept no of the location which is having space char in it.

☺ select loc from dept where instr(loc,' ',1,1) !=0

NVL FUNCTION :- Used to convert null values to actual values.

Syntax :- NVL (arg1,arg2)

Exaample :- NVL(sal+comm,sal)

MOD FUNCTION :- Used to find the modulus of any two integer numbers.

PSUEDO COLUMNS :- They are the false column that are present in each and every table and must be called explicitly.

We have two columns :- 1) Rowid 2)Rownum

Rowid is an 18 digit address that is present for each and every record and it is nothing but a physical memory location on which record is stored.

It is generated at the time of the insertion of the record. It is unique.It is used to uniquely identify the record from the table. It is fastest way to access or delete the record. It is static (it does not change)

Eg :- select rowid,emp.* from emp;

Rownum is a sequential number allocated to the result table.

It is generated at the time of execution. It can be changed as it Is dynamic. It cannot be same. It always starts with 1.

Write a query to display third record from the table.

☺ select * from (select rownum slno,emp.* from emp) where slno=3;

Write a query to display the last five records from the table.

☺ select * from

(select rownum slno,emp.* from emp)

where slno > (select count(*) -5 from emp);

Write a query to display 4^{th} maximum salary.

☺ select distinct sal from

Retrieval of data from multiple tables is known as **JOIN.**

TYPES :-

1) CARTESIAN JOIN /CROSS JOIN 2)INNER JOIN /EQUI-JOIN 3) NATURAL JOIN 4) OUTER JOIN has 3 types LEFT-OUTER ,RIGHT-OUTER,FULL OUTER-JOIN. 5) SELF-JOIN

CARTESIAN JOIN :- In cartesian join the records of table one will be merged with all records of table 2.

No of columns in the resultant table will be equal to the summation of number of columns present in both the tables. The number of records in the resultant table will be product of number of records in both the tables.

Syntax :- Using ANSI

Select col_name from table_name 1 cross join table_name 2

Syntax :- Using Oracle

Select col_name from table_name 1 , table_name 2

INNER JOIN / EQUI JOIN :- Inner join are used to obtain only the matching record or the records which have a pair .

It is nothing but Cartesian join but It has a condition .

Syntax :- Using ANSI

Select col_name from table_name 1 inner join table_name 2 on <join_condition>

Eg :- Select * from emp on emp.dno = dept.dno.

Syntax :- Using Oracle

Select col_name from table_name 1 inner join table_name 2 where <join_condition>

Eg :- Select * from emp where emp.dno = dept.dno.

Write a query to display the name and deptno of all the employee.

☺ select ename,dept.deptno

from emp,dept

where emp.deptno=dept.deptno

Write a query to display sal and loc of all the emp.

☺ select sal,loc from emp e ,dept d where e.deptno=d.deptno

Write a query to display name and dname for all the emp who works in dept 10.

☺ select dname,ename from emp e,dept d where d.deptno=d.deptno and d.deptno=10

Write a query to display name,dname and loc for all the emp who works in research dept.

☺ select dname,ename,loc from emp e,dept d where d.deptno=e.deptno and d.dname='RESEARCH'

Write a query to display name,sal and annual sal along with loc if their loc are Dallas or Chicago.

☺ select distinct ename,dname,sal,sal*12 annualsal,loc from dept d,emp e where e.deptno=d.deptno and d.loc in ('DALLAS','CHICAGO')

Write a query to display ename and dname if emp are earning more than 2000.

☺ select distinct ename,dname from emp e,dept d where e.deptno=d.deptno and e.sal >2000;

When do we use sub-queries and join ?

It totally depends on requirement.

Write a query to display who are working dname research.

For this we can use join or sub-query.

Write a query to display employees information along with their dname.

In this type we cannot use sub-query method.

NOTE ☺ If we join n number of tables we have n-1 number of join condition.

NATURAL JOIN :- It is similar to inner join we don't write any join condition

If there are common col present in both the tables it returns inner join output.(only matching records) else it returns Cartesian join output.(matching and un-matching records)

In Natural join we have ANSI syntax

Syntax :- Using ORACLE

Select from table_name 1 , table_name 2 where condition .

Syntax :- Using ANSI

Select col_name from table_name 1 natural join table_name 2 on condition .

Eg :- For Oracle Syntax :- Select ename,dname from emp e,dept d where e.deptno=10.

OUTER JOIN :- They are used to obtain un matched record or the records which do not have pair

LEFT OUTER JOIN :- It is used to obtain un matched records of left table along with the matched records.

Syntax :- Select col_name from table_name 1 LEFT OUTER JOIN table_name 2 on join condition.

Syntax :- Using Oracle

Select col_name from table_name 1 ,

table_name 2 where join condition.

Write a query to display employees who doesn't work in any dept.

RIGHT OUTER JOIN :- It is used to um-matched record of right table and record of right table.

Eg :- Select * from emp e,dept d where e.deptno=d.deptno and ename=null;

FULL OUTER JOIN :- It is used to obtain un-matched record of both the table along with matched record.

NOTE ☺ We don't have syntax for full outer join in Oracle.

Syntax :- Using ANSI

Select col_name from table_name 1 full outer join table_name 2 condition.

Select * from emp full outer join on emp.deptno=dept.deptno join condition.

SELF-JOIN :- Joining a table with itself is self join.

Whenever we want to display emp name with manager we have to go with self join

Write a query to display ename and manager name for all the emp.

☺ select e1.ename,e2.mgrname from emp e1,emp e2 where e1.empno=e2.mgr .

Write a query to display mgr name ,ename, along with his salary if the emp is working in dept no=10;

Write a query to display name of the emp if the emp working as clerk.

Select ename from emp e where e.job='CLERK';

Write a query to display ename and managers designation if manager works in dept 10 or 20.

Write a query to display ename and managers hiredate if emp was hired before 1982.

☺ Select ename,e2.hiredate from emp e1,emp e2 where e1.empno=e2.mgr and e1.hiredate < '31-DEC-1981'

Write a query to display ename and managers commission if emp works as salesman and manager and dept no 30

☺ Select ename,e2.comm from emp e1,emp e2 where e1.empno = e2.mgr and e1.job='SALESMAN' and e1.deptno=30;

Write a query to display ename and manager name and their sal if emp are more than manager

☺ Select e1.ename,e2.ename,e1.sal,e2.sal from emp e1,emp e2 where e1.empno=e2.mgr and e1.sal>e2.sal.

Write a query to display ename and hiredate and manager name and hiredate if manager was hired before emp.

☺ Select e1.ename,e1.hiredate,e2.hiredate,e2.ename from emp e1,emp e2 where e1.empno=e2.mgr and e2.hiredate<e1.hiredate.

Write a query to display ename and manager name if both are working in same job.

Select e1.ename,e2.ename from emp e1,emp e2 where e1.empno=e2.mgr and e1.job=e2.job.

DDL COMMANDS :-

Create :- Used to create an object in the database.

Eg :- Creation of a table.

Syntax :- Create table table_name (Col_name Datatype NOTNULL/NULL,col_name2 datatype NotNULL/NULL upto……

Col_name n datattype NotNULL/NULL,Constraint constraint_ref_name,unique (col_name),Constraint_ref_name check(condition),Constraint Const_ref_name P.K(col_name),Constraint const F.K(col_name) references Parent table name(col_name));

Customer Table

CID	CNAME	PH_NO	Address
Number(3)	Varchar(20)	Number(10)	Varchar(20)
Not Null	Not Null	Not Null	Not Null
Constraints P.K (c_id)		UNIQUE {Phone no} Check Ph_no=10	

Common rules to create table :-

1) Table cannot have same name.

2) Table should have at least one column and data types.

Table name **PRODUCT** number of cols 4.

PID	PNAME	PRICE	CID
Number(3)	Varchar(20)	Number(10)	Number(10)
Not Null	Not Null	Not Null	Null
Constraints P.K (p_id)		Check pr >0	

RENAME :- Used to change the name of the existing object.

Eg :- You can rename a table

Syntax :- Rename current table name to new table name.

ALTER :- This statement is used to modify the structure of the table.

Syntax :- Alter table table_name

Add col_name followed by data type (Null/NotNULL)

To Drop the column we used

Alter table table_name drop column col_name;

To change the data type

Alter table table_name modify Column_name new datatype.

To change the NULL or NOT NULL Constraint

Alter table table_name modify col_name existing data type null/notnull.

To rename the column.

Alter table table_name rename column current col_name to new col_name.

To modify constraints.(Similar to change null/notnull)

Alter table table_name add new

constraint const_ref_name new constraint name (col_name).

TRUNCATE :- It is used to remove all the records from the table permanently.

Syntax :- Truncate table table_name;

NOTE ☺ To create a new table similar to an existing one use Create table_name as (select * from existing table_name)

DROP :- It is used to remove the table from the database along with records.

Syntax :- DROP table table_name;

FLASHBACK :- It is used recover the table from bin.

Create a new table by copying the values from an existing one

Drop it using drop command.

You can retrieve the table again by using the following command

FLASHBACK TABLE TABLE_NAME TO BEFORE DROP;

Eg :- FLASHBACK table emp to before drop;

PURGE :- To move the table from bin to the trash.

Transaction Control Language (TCL):-

These statements are used to monitor the transactions done on the database.

In this we have 3 statements :- **COMMIT ,ROLLBACK ,SAVEPOINT.**

COMMIT :- This statement is used to save all the transaction on the database.

Syntax :- Commit;

ROLLBACK :- Used to go back to the latest committed or saved statement.

ROLLBACK TO SAVEPOINT :-

Syntax :- Rollback to savepoint name

SAVEPOINT:- It is used to mark the position on the database

Syntax :- Savepoint savepoint_name.

Eg:- Savepoint a;

SQL> select * from emp1;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	SMI	CLASS					
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

15 rows selected.

SQL> delete from emp1 where ename='ALLEN';

1 row deleted.

SQL> savepoint a;

Savepoint created.

SQL> delete from emp1 where comm >0;

2 rows deleted.

SQL> savepoint b;

Savepoint created.

SQL> delete from emp1 where empno in (7902,7934);

2 rows deleted.

SQL> savepoint c;

Savepoint created.

SQL> delete from emp1 where sal=(select max(sal) from emp1);

1 row deleted.

SQL> savepoint d;

Savepoint created.

SQL> select count(*) from emp1;

  COUNT(*)

         9

SQL> rollback to d;

Rollback complete.

SQL> select count(*) from emp1;

```
  COUNT(*)
--------
         9

SQL> rollback to c;

Rollback complete.

SQL> select count(*) from emp1;

  COUNT(*)
--------
        10

SQL> rollback to b;

Rollback complete.

SQL> select count(*) from emp1;

  COUNT(*)
--------
        12
```

SQL> rollback to a;

Rollback complete.

SQL> select count(*) from emp1;

COUNT(*)

14

Note ☺ DDL Statements are auto-commit statements.

Data Control Language (DCL) :-

These are used to control the flow of data between the users.

We have two statements :-

1) Grant

2) Revoke

GRANT :- It is used to provide permission from owner user to other users.

Syntax :- Grant privilege_list on table_name to user_name;

UPDATE :- Used to change a value in a database.

Syntax :- Update table_name

Set col1=v1,col2=v2 …… coln=val_n)

Where (filter condition);

Eg:- Update set sal=sal*2 where ename='SMITH';