* Introduction
* Data Types
* Number
* Char
* V archar ( or ) Varchar
* Date
* Time Stamp
* Long
* Raw
* Long Raw
* Lob ( Clob, Blob, Bfile & NCLob )

1. **Number:**
   * It allows only numeric values
   * Maximum size is 38 digits  
     **Syntax :**X Number ( P, (S));  
     P => It allows how many digits to store  
     S => Size  
     **Ex :**X Number (5,2)
2. **Char:**
   * It allows alphanumeric characters ( Numbers + Characters )
   * Maximum size is 2000 Bytes/ Characters  
     **Syntax :**X Char( S );
3. **Varchar2 ( or ) Varchar2:**
   * It allows alphanumeric characters
   * Max size 4000 Bytes/ Characters
   * Memory allocation is dynamic   
     **Syntax :**X Varchar2( S );
4. Date:
   * It is used to store date values
   * Max size is 7 Bytes  
     **Syntax : X**Date;
5. Timestamp:
   * It is used to store date along with fraction of seconds.  
     Syntax : X Timestamp;
6. Long:
   * It is used to store information
   * Max size is 2 GB.
   * Only once we have to use in entire table.   
     **Syntax :**X Long;
7. Raw:
   * It is used to store images
   * Max size is 2000 Bytes.   
     **Syntax :**X Raw;
8. Longraw:
   * It is used to store information as well as images.
   * Max size is 2GB  
     **Syntax :**X Longraw;
9. Lob
   * Clob :
     + It is used to store huge information
     + Max size is 4 GB  
       **Syntax :**X Clob;
   * Blob :
     + It is used to store images but in the form of binary format.
     + Max size is 4 GB   
       **Syntax :**X Blob;
   * Bfile :
     + It is used to store the files.
     + Max size is 4 GB.   
       **Syntax :**X Bfile;
   * NCLob:
     + It is used to store multiple languages ( Unicode Format )

**SQL Statements**

* DDL ( Data Definition Language )
* DML ( Data Manipulation Language )
* DQL ( Data Query Language )
* TCL ( Transaction Control Language )
* DCL ( Data Control Language )

1. DDL :
   * These are auto commit commands.
   * These are session independent.
   * These are used to define database objects.
     1. Create
        + It is used to create the database object   
          **Syntax :**Create table tablename ( Col-1 Datatype, Col-2 Datatype,………);
     2. Alter
        + It is used to alter the structure of table
          1. Add   
             : It is used to add the columns in a table   
             **Syntax :**Alter table tablename add colname Datatype ( S );
          2. Modify  
             It is used to modify the column in a table   
             **Syntax :**Alter table tablename modify colname Datatype ( S );
          3. Drop  
             : It is used to drop a column in a table   
             **Syntax :**Alter table tablename drop column colname Alter table tablename drop ( Col-1, Col-2,…… );
          4. Rename  
             : It is used to rename a column name in a table.   
             **Syntax :**Alter table tablename rename column oldcol to newcol;
     3. Drop
        + It is used to drop table from the database.

**Syntax :**Drop table tablename;

* + 1. Rename
       - It is used to rename the table name

**Syntax :**Rename oldtablename to newtablename

1. DML :
   * It is used to handle the data in database object.
   * These are non auto commit commands.
   * These are session dependent.
     1. Insert
        + It is used to insert the data into table.
        + We can insert the data into table in 2 methods
          1. Direct Method  
             : It is used to insert the data directly to a table.  
             **Syntax :**Insert into tablename ( Col-1,Col-2 ) Values ( Val-1, Val-2 )
          2. Reference Method : It is used to insert the data into table thorugh prompt.  
             **Syntax :**Insert into tablename values ( &Col-1, &Col-2 ).
     2. Update
        + It is used to update the data in a table.  
          **Syntax :**Update tablename set Col-1=Val-1,Col-2=Val-2…… where condition.
     3. Delete
        + It is used to delete the data in a table.  
          **Syntax :**Delete from tablename where condition
2. DQL :
   * It is used to retrive the data from a table.  
     **Syntax :**Select \* from tablename;
3. TCL :
   * It is used to save the transactions on a table.
     1. Commit
        + It is used to save the data permanently in a database.
          1. Implicit Commit  
             : It is applied by the system.
          2. Explicit Commit   
             : It is applied by the user.
     2. Rollback
        + It is used to cancel the previous transactions.
     3. Save Point
        + It is used to mark a specific record.
        + It is only for tempary purpose.  
          **Syntax :**Savepoing S1;
     4. Truncate
        + It work like a Delete + Commit. **Syntax :**truncate table tablename.
4. DCL :
   * It is used to provide the access to users.
     1. Grant
        + It is used to provide the permissions to users.
     2. Revoke
        + It is used to cancel the permissions to users.

**Clauses :**

* Select Clause
* From Clause
* Where Clause
* Group by Clause
* Having Clause
* Order by Clause
* Distinct Clause

1. Select Clause
   * It is used to retrieve the data from table.
2. From Clause
   * It is used to retrieve the data from which tables.
3. Where Clause
   * It is used to provide the conditions.
   * It is used to filter the data from grouped records.
   * It won’t allow group functions and alias names.  
     **Syntax :**select \* from emp where deptno=10;
4. Group By Clause
   * It is used to make the data into group format.
   * It is not possible to provide the group functions along with normal columns in a select statement without using group by clause.  
     **Syntax :**select \* from emp group by deptno;
5. Having Clause
   * It is used to provide the conditions.
   * It is used to filter the data from grouped data based on condition.  
     **Syntax :**select \* from emp group by deptno having count(\*) > 1;
6. Order by Clause
   * It is used to make the data in order  
     **Syntax :**select \* from emp order by sal;
7. Distinct Clause
   * It is used to restrict the duplicate records.  
     **Syntax :**select distinct ( empno ) from emp;

**Operators**

* Arthemetic Operators
* Logical Operators
* Relational Operators
* Special Operators
* Set Operators

1. Arthemetic Operators ( /, +, \*, - )
   * It is used to do the mathematical functions Ex : select 2+2,2\*2 from dual;
2. Logical Operators ( And, Or, Not )
   * Ex-1 : select \* from emp where deptno=10 and sal>1000;
   * Ex-2 : select \* from emp where deptno=10 or deptno=20;
   * Ex-3 : select \* from emp where not deptno=20;
3. Relational Operators ( =, <, >, <=, >=, != )  
   Ex-1 : select \* from emp where deptno <=20;
4. Special Operators ( Is, In, Like, Between )
   * Ex-1 : select \* from emp where comm is null;
   * Ex-2 : select \* from emp where comm is not null;
   * Ex-3 : select \* from emp where sal in ( 800, 1500, 2000);
   * Ex-4 : select \* from emp where sal between 1000 and 2000;
   * Ex-4 : select \* from emp where ename like ‘—‘;
   * Ex-5 : select \* from emp where ename like ‘s%’;

NULL :

* + It is unmeasured value
  + It is neither ‘0’ or ‘empty’
  + Every null value is uniquely considered by oracle engine.
  + Any data type will support to store null values
  + It display as blank or prompt.
  + If you calculate any value with null finally we are getting null only.

1. Set Operators
   * By using set operators we join more than one query such queries are called compound queries.
   * In each of select statement there must be same number of columns and same data type but must not be same size.
     1. Union All :
        + It displays the all values along with duplicate values also.
        + Two queries must have equal number of columns  
          **Syntax :**select \* from query1 union all select \* from query2;
     2. Union :
        + It is similar to that of union all, but it wont display the duplicate values.  
          **Syntax :**select \* from query1 union select \* from query2;
     3. Intersect :
        + It displays the common values from two queries  
          **Syntax :**select \* from query1 intersect select \* from query2;
     4. Minus :
        + It displays the first query records, which are not found in the second query records.  
          **Syntax :**select \* from query1 minus select \* from query2;

**Functions**

* Number Functions
* String Functions
* Date Functions
* Conversion Functions
* General Functions
* Aggrigate Functions

1. Number Functions
   1. Power (M, N)  
      **Syntax :**select power( 25, 2 ) from dual;

DUAL :

* + - It is a dummy table which is provided by Oracle engine.
    - It has only one column which is associated with Varchar data type.
  1. Sqrt (M)  
     **Syntax :**select sqrt( 625 ) from dual;
  2. Mod (M, N)  
     **Syntax :**select mod( 5, 2 ) from dual;
  3. Ascii ( C )  
     **Syntax :**select ascii( ‘a’ ) from dual;
  4. Ceil (M)
     + It displays the next highest value  
       **Syntax :**select ceil ( 12.45 ) from dual.
  5. Floor (M)
     + It displays the next lowest value  
       **Syntax :**select floor ( 13.65) from dual;
  6. Round (M, N)
     + It rounds the value up to given number of position. That is if last eliminating value is >=5 then it simply add one value to the left adjacent value.
     + It check the condition.  
       **Syntax :**select round ( 15.2345, 2 ) from dual;
  7. Trunc ( M, N )
     + It work similar to that of round, but it won’t check the condition. **Syntax :**select trunk ( 12.567, 2 ) from dual;

1. **Sting Functions**
   1. Length ( S )
      * It is used to display the number of characters in a given string.  
        **Syntax :**select length( ‘ebs’ ) from dual;
   2. Reverse ( S )
      * It is used to reverse the given string.  
        **Syntax :**select reverse ( ‘ebs’ ) from dual;
   3. Upper ( S)
      * It is used to convert the string into upper characters.  
        **Syntax :**select upper( ‘ebs’ ) from dual;>
   4. Lower ( S )+
      * It is used to convert the string into lower characters.  
        **Syntax :**select lower ( ‘EBS’ ) from dual;
   5. Initcap ( S )
      * It is used to convert the first character into upper character in a given string.  
        **Syntax :**select initcap ( ‘business’ ) from dual;
   6. Concat ( S1, S2 )
      * It is used to merge the two strings. And we have to use ‘||’ symbol while merge the two strings.  
        **Syntax :**select concat ( ‘ebs’, ’solutions’ ) from dual;  
        **Syntax :**select ‘ebs’ || ‘business’ || ‘solutions’ from dual;
   7. Ltrim ( S, C )
      * It is used to remove the character from left end of the given string, if the character is found.  
        **Syntax :**select ltrim ( ‘ebsebs’ , ‘e’ ) from dual;
   8. Rtrim ( S, C )
      * It is used to remove the character from right end of the given string, if the character is found.  
        **Syntax :**select rtrim ( ‘ebsess’ , ‘s’ ) from dual;
   9. Trim
      * It is used to remove the characters from both sides in a given string.  
        **Syntax :**select trim ( ‘e’ from ‘eebse’ ) from dual;
   10. Lpad
       * It is used to add the character from left end.  
         **Syntax :**select lpad ( ‘ebs’, 5 , ‘&’ ) from dual;
   11. Rpad
       * It is used to add the character from rightend.  
         **Syntax :**select rpad ( ‘ebs’, 7 , ‘&’ ) from dual;
   12. Translate ( S, C, C )
       * It is used to translate the character wise in a given string, if the character is found.
       * It is not possible to translate entire string.  
         **Syntax :**select translate ( ‘welcome’ , ‘w’ , ‘t’) from dual;
   13. Replace ( S, S ,S )
       * It is used to replace entire string.
       * It is not possible to replace more than one string.  
         **Syntax :**select replace ( ‘e business solutions’, ‘business’, ‘ebs’ ) from dual;
   14. Decode ( Column, Condition, Do1,…………….. Column)
       * It is used replace more than one string.
       * It works like as a if condition but it does not allow the relational operators.  
         **Syntax :**select job, decode ( job, ‘manager’, ‘mgr’, ‘clerk’, ‘clk’, ‘salesman’, ‘sls’, job ) from dual;
   15. Case ( when condition then result else default value )
       * It is used to replace more than one string by using relational operator.  
         **Syntax :**select case when deptno=10 and job=’MANAGER’ then ‘mgr’ else job end j from emp;
   16. Substr ( S, M, N )
       * It is used to display the set of characters from a given string.  
         S = String   
         M = Position  
         N = No of Characters   
         **Syntax :**select substr ( ‘welcome’, 1,3 ) from dual;
   17. Instr ( S, C, M, N )
       * It is used to display the position number of a given character.  
         S = String  
         C = Character  
         M = Position  
         N = Occurance   
         **Syntax :**select instr ( ‘welcome’, ‘e’, 1, 1 ) from dual;
2. Data Functions
   1. Sysdate :
      * It is used to display the system date.  
        **Syntax :**select sysdate from dual;
   2. Current\_Date :
      * It is used to display the next day. **Syntax :**select current\_date from dual;
   3. Add\_Months :
      * It is used to add or substract number of months for a given date.   
        **Syntax :**select add\_months( sysdate, 1) from dual;
   4. Months\_Between ( Date1, Date2 ):
      * It is used to display the number of months between two dates   
        **Syntax :**select months\_between ( sysdate, hiredate ) from emp;
   5. Next\_Day ( Date, ‘format’ )
      * It is used to display the next day date based on the format.  
        **Syntax :**select next\_day ( sysdate, ‘sun’ ) from dual;
   6. Last\_Day ( Date )
      * It is used to display the last day of the given month.  
        **Syntax :**select last\_day ( sysdate ) from dual;

**Date Formats :**

|  |  |  |
| --- | --- | --- |
| D | ⇒ | Number of day in the week |
| DD | ⇒ | Number of day in the month |
| DDD | ⇒ | Number of day in the year |
| DY | ⇒ | First 3 Characters of the day - SUN |
| Dy | ⇒ | First 3 Characters of the day - Sun |
| Dy | ⇒ | First 3 Characters of the day - sun |
| DAY | ⇒ | Complete Characters of the day |
| Day | ⇒ | Complete Characters of the day |
| Day | ⇒ | Complete Characters of the day |
| MM | ⇒ | Number of the month in the year. |
| MON | ⇒ | First 3 Characters of the month |
| Mon | ⇒ | First 3 Characters of the month |
| Mon | ⇒ | First 3 Characters of the month |
| MONTH | ⇒ | Complete Charaters of the month |
| Month | ⇒ | Complete Charaters of the month |
| Month | ⇒ | Complete Charaters of the month |
| Y | ⇒ | Last digit of the year |
| YY | ⇒ | Last two digits of the year |
| YYYY | ⇒ | Last three digits of the year |
| YYYY | ⇒ | Four digits of the year |
| YEAR | ⇒ | Year in the character format. |
| HH | ⇒ | An hour of the day |
| HH24 | ⇒ | 24 Hours format. |
| MI | ⇒ | Minits of the Hour |
| SS | ⇒ | Seconds of the minute. |
| SSSS | ⇒ | Seconds since starting of the day |
| FS | ⇒ | Fraction of Seconds |
| W | ⇒ | Week ot the month |
| WW | ⇒ | Week of the year |
| Q | ⇒ | Quarter of the year |

1. **Conversion Functions**
   1. To\_Char ( Date, ‘format’ )
      * It is used to convert system format in to user format  
        **Syntax :**select to\_char ( sysdate, ‘day’ ) from dual;
   2. To\_Date ( ‘C’, ‘format’ )
      * It is used to convert user format into system format  
        **Syntax :**select to\_date ( ‘21’, ‘DD’ ) from dual;   
        Select to\_date ( ‘december’, ‘MM’ ) from dual;
   3. To\_Number
      * It is used to translate a value of char or varchar data type to number format.  
        **Syntax :**select to\_number ( ‘20’ ) from dual;
2. General Funtions
   1. User & Uid
      * Select user,uid from dual;
   2. Greatest & Least
      * Select greatest ( 1,2,3 ), least ( 1, 2, 3 ) from dual;
   3. NVL ( Col1, Val )
      * It is used to handle the null values
      * It work like as a if condition  
        **Syntax :**select sal, comm, sal+nvl(comm, 0) from emp;
   4. NVL2 ( Col1, Val1, Val2 )
      * It is a advanced of nvl
      * It work like as a if then else condition  
        **Syntax :**select sal, comm, nvl2 ( comm, 0, 100 ) from emp;
3. Aggregate Functions
   1. Min
      * Syntax : select min ( sal ) from emp;
   2. Max
      * Syntax : select max ( sal ) from emp;
   3. Avg
      * Syntax : select avg ( sal ) from emp;
   4. Sum  
      Syntax : select sum ( sal ) from emp;
   5. Count ( \* )
      * It is used to count of the all records from a table  
        **Syntax :**select count( \* ) from emp;
   6. Count ( column )
      * It is used to count the given column values  
        **Syntax :**select count ( empno ) from emp;

**Constraints**

* Primary Key
* Composite Primary Key
* Unique
* Not Null
* Check
* Default
* Foreign Key / Reference Key
* Constraints are rules which are used to allow the valid data

1. Primary Key
   * It won’t allow duplicate records and null values  
     **Syntax :**create table tablename ( sno number(5) primary key )
2. Composite Primary Key
   * It is used to create primary key on multiple columns  
     **Syntax :**create table tablename ( sno number(5), sname varchar2(20) primary key ( sno, sname );
3. Unique
   * It is allow only unique values.
   * It does not allow duplicate records  
     **Syntax :**create table tablename ( sno number (5) unique );
4. Not Null
   * It is allow only not null values
   * It does not allow null values  
     **Syntax :**create table tablename ( sno number ( 5 ) not null );
5. Check
   * It is used to check the condition  
     **Syntax :**create table tablename ( sno number ( 5 ), check ( sno>0 ) );
6. Default
   * It is used to insert default values  
     **Syntax :**create table tablename ( sno number ( 5 ), grade char ( 2 ) default ‘A’ );
7. Foreign Key
   * It is used to maintain a reference from one table to another table.  
     **Syntax :**create table table1 ( sno number (5) primary key ) Create table table2 ( dno number ( 5 ), dname varchar2(10), sno number(5) references table1 (sno)

**Joins**

* Simple Join
* Self Join
* Outer Join

1. Simple Join
   1. Equi Join
      * It is used to join two tables based on equal condition.  
        **Syntax :**select \* from emp, dept where emp.deptno=dept.deptno;
   2. Non Equi Join
      * It is used to join two tables based on not equal condition  
        **Syntax :**select \* from emp, dept where emp.deptno != dept.deptno;
2. Seft Join
   1. It is used to join the table itself.  
      **Syntax :**select \* from emp e1, emp e2 where e1.deptno=e2.deptno;
3. Outer Join
   1. Left Outer Join
      * It is used to display the full details of the left table and matched records of the right table.  
        **Syntax :**select \* from emp e,dept d where emp.deptno = dept.deptno(+);
   2. Right Outer Join
      * It is used to display the full details of the right table and matched records of the left table.  
        **Syntax :**select \* from emp e,dept d where emp.deptno(+)=dept.deptn;
   3. Full Outer Join
      * If you join left and right outer joins with union operators such joins are called full outer join.  
        **Syntax :**select \* from emp e,dept d where emp.deptno(+) = dept.deptno   
        Union  
        select \* from emp e,dept d where emp.deptno(+) = dept.deptno

**Synonyms**

* Private Synonym
* Public Synonym
* It is used to hide the owner of the table.
* It work like as a mirror image of the tables.
* It does not have a own structure.
* It is depend on the tables.
* We can possible to create the synonym on tables but we can’t create the synonym
* All synonyms are stored in user\_synonyms table.

1. Private Synonym
   * It is used to create private synonym in current schema and accessed within that schema only.  
     **Syntax :**create synonym synonym\_name for table\_name;
2. Public Synonym
   * It is used to create public synonym in current schema and accessed from other schemas also.  
     **Syntax :**create public synonym synonym\_name for table\_name;

**Views**

* Simple View
* Complex View
* Force View
* Vertical View
* Horizantal View
* Functional View
* Partition View
* Materialized View
* Inline View
* These are the advanced of synonyms
* It is a virtual table to hide the base table and it work like a mirror image of the table.
* It doesn’t have own structure
* It is not possible to modify the structure of a table by using views
* We can define view on synonyms and synonym on views.
* We can possible to define the view on particular columns only.
* All views are stored in all\_views.

1. Simple View
   * It is used to define a view on single table that views are called simple view.  
     **Syntax :**create view view\_name as select \* from table\_name;
2. Complex View
   * It is used to define a view on multiple tables that views are called complex view.  
     **Syntax :**create view view\_name as select \* from emp,dept where  
     emp.deptno=dept.deptno;
3. Force View
   * It is used to define a view without base table.  
     **Syntax :**create force view view\_name as select \* from non existing table;
4. Vertical View
   * It is used to define a view on specific columns in a table.  
     **Syntax :**create view view\_name as select empno,ename,job,sal from emp;
5. Horizantal View
   * It is used to define a view on specific records in a table.  
     **Syntax :**create view view\_name as select \* from emp where deptno=10;
6. Functional View
   * It is used to define a view with functions on table.  
     **Syntax :**create view view\_name ( col1, col2 ) as select fun1, fun2 frome mp;   
     **Syntax :**create view v1 ( a , b ) as select as select min ( sal ), max ( sal ) from emp;
7. Partition View
   * It is used to define a view on compound queries.

**Syntax :**create view view\_name as query1 union query2

1. Materialized View
   * It is one of the view which is having the own structure.
   * It doesn’t allow the dml operations on views.
   * It is used to store the historical data.
   * We can define the view on table which is having the primary key.

**Syntax :**create materialized view view\_name as select \* from emp;

1. Inline View
   * It work like as a query, which is having the query in from clause or instead of table.

**Syntax :**select \* from ( select \* from emp );   
Ex-1 : First 5 Records :

* + Select \* from ( select emp.\*,rownum r from emp ) where r<=5;

Ex-2: Last 5 Records :

* + Select \* from ( select emp.\*,rownum r from emp ) where r> ( select max(rownum) - &n from emp );

Ex-3: Random Records:

* + Select \* from ( select emp.\*,rownum r from emp ) where r in ( 1,3,5 );

Ex-4: Even no of Records :

* + Select \* from ( select emp.\*,rownum r from emp ) where mod(r,2) = 0;

Ex-5: Last Record

* + Select \* from (select emp.\*,rownum r from emp) where r= ( select count(\*) from emp);

**Indexes**

* Simple Index
* Complex Index
* Unique Index
* Functional Index
* Bitmap Index
* It is one of the object which is used to retrieve the data from the database fastly.
* It is used to increase the performance while retrieve the date from the database.
* It will make the use of user\_id’s.
* All indexes are stored in all\_indexes.

1. Simple Index
   * It is used to create a index on single column of a table.

**Syntax :**create index index\_name on table\_name ( column\_name )

1. Complex Index
   * It is used to create a index on multiple columns of a table.

**Syntax :**create index index\_name on table\_name ( col1, col2 )

1. Unique Index
   * It is used to create a index on columns which are having unique data.

**Syntax :**create unique index index\_name on table\_name ( col1 );

1. Functional Index
   * It is used to create a index on columns while making use of the functions.

**Syntax :**create index index\_name on table\_name ( function ( column ) ); Create index index\_name on emp ( length ( ename ) );

1. Bitmap Index
   * It is used to create a bit map index on column.

**Syntax :**create bitmap index index\_name on emp ( empno );

**Clusters**

* It is a logical boundary which is used to improve the overall performance of the database.
* We can create the cluster on tables, but we can’t create on columns.
* We can possible to create a index on cluster.
* All clusters are stored in all\_clusters **Syntax :** create cluster cluster\_name ( column\_name datatype );  
  Create cluster cl1 ( sno number(5) );  
  Create table table\_name ( column\_name datatype(n)) cluster cluster\_name ( column\_name );  
  Create table t1 ( sno number(5)) cluster cl1 ( sno );  
  Create index index\_name on cluster cluster\_name;

**Sequence**

* It is used to create sequence on columns in a table.
* While insert the data into tables we use the sequence.
* All sequences are stored in all\_sequences. **Syntax :** create sequence sequence\_name;  
  Create sequence sequence\_name increment by 1 start with 1;  
  **Currval :** it is used to insert current value.  
  **Nextval :** it is used to insert next value  
  Create table t1 ( sno number(5), cno number(5));  
  Insert into t1 values ( seq1.currval, seq1.nextval );
* Sub Queries
  + Query within the query is called as a sub query.
    - Simple Sub Query
    - Co related Sub Query
  + Simple Sub Query
    - In simple sub query first inner query is executed independently, based on inner query value outer query is executed.
    - Outer query is depend on inner query but inner query doesn’t depend on outer query.

**Syntax :**select \* from emp where empno= ( select \* from emp );

Ex-1 : Display the employees who are working in research department?

* + - Select empno,ename from emp where deptno=(select deptno from dept where dname='RESEARCH');

Ex-2 : Display the employee details who are getting maximum salary?

* + - select \* from emp where sal = (select max(sal) from emp)

Ex-3 : Display the employee details who are getting second maximum salary?

* + - select \* from emp where sal = (select max(sal) from emp where sal<(select max(sal) from emp))

Ex-4 : Display the employees details to get the particular maximum salary employee?

* + - select \* from emp e where &n=(select count(distinct(sal)) from emp where sal>=e.sal)

Ex-5 : Display the maximum salary emp data in particular dept;

* + - Select \* from emp e where sal=(select max(sal) from emp where deptno=10);

Ex-6 : Display the maximum salary emp details in dept wise.

* + - Select \* from emp e where sal=(select max(sal) from emp group by deptno);

Ex-7 : Display the employees who are reporting to KING?

* + - select \* from emp where mgr=(select empno from emp where ename='KING')

Ex-8 : Display the Department details which are having more than 5 employees?

* + - select \* from dept where deptno in (select deptno from emp group by deptno having count(\*)>=5)

Ex-9 : Display the employees who are having at least 2 reporting?

* + - select \* from emp where mgr in (select mgr from emp group by mgr having count(\*)>=2) order by mgr

Ex-10: Display the dept details which are having at least 3 salesmans?

* + - select \* from dept where deptno =(select distinct (deptno) from emp where job='SALESMAN')

Ex-11 : Display the duplicate records in a table?

* + - select \* from emp where rowid not in ( select max(rowid) from emp group by empno);
  + Co related Sub Query
    - In this query first outer query get executes based on outer query value inner query get executed and return a value and very finally based on the inner query value outer query value will be displayed.

**Syntax :**select \* from emp e where 1=(select count(\*) from emp where e.sal<=sal); Select \* from emp e where 1=(select count(\*) from emp where e.sal>=sal);