1.SQL Components

1.1) DDL.

1.2) DML.

1.3) DCL.

1.4) TCL.

1.5) CLAUSES.

2. Data Types.

3. Operators.

4. Constraints.

5. SQL Functions.

6. Joins.

7. Sub Queries.

8. Analytical Functions.

9. View

10. Synoyms.

11. Sequence.

12. Tables

12.1)External Table.

12.2)GTT

12.3)Partition Table.

12.4)Normal Table.

13. User Defined Data Types.

14. Directory.

15. DB link.

16. Indexes.

17. Hierarchical Query.

19. SQL Loader.

20.Null

21.Pseudo Columns.

**1) SQL Components:-**

1.1)DDL:- This language have commands which is used to create db object, modify db object structure and removing db object.The db objects are table, view, index, synonym, sequence..etc. The commands are

1. Create

2. Truncate.

3. Drop

4. Flashback

5. Purge

6. Rename

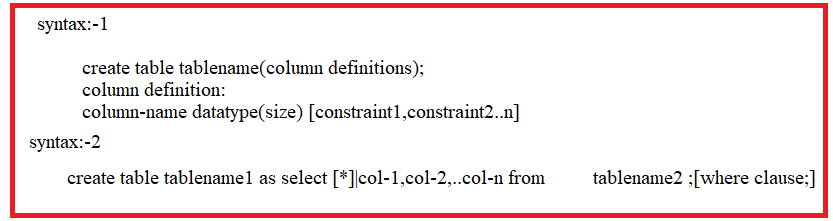
7. Alter

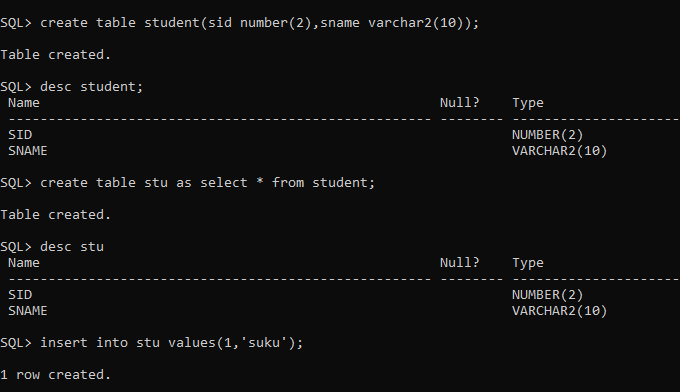
8. Comment.

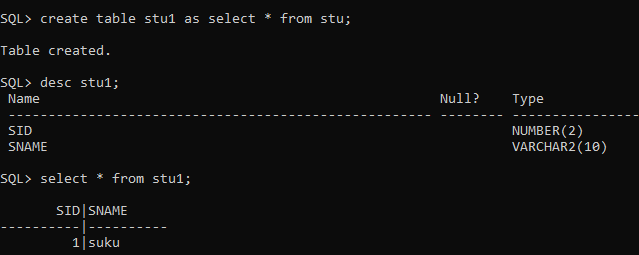
NOTE:-1 The DDL commands directly works on physical object(DATABASE).There is no involvement of buffer while DDL command is being executed.

2. Before starting execution and after execution of DDL command , implictly commit command is executed.

**1.CREATE**:- we define dbobject structure using create command.This command creates table with specified structure.But the table does't has rows.

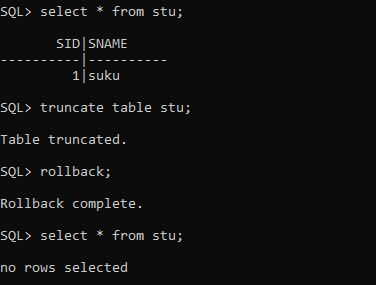






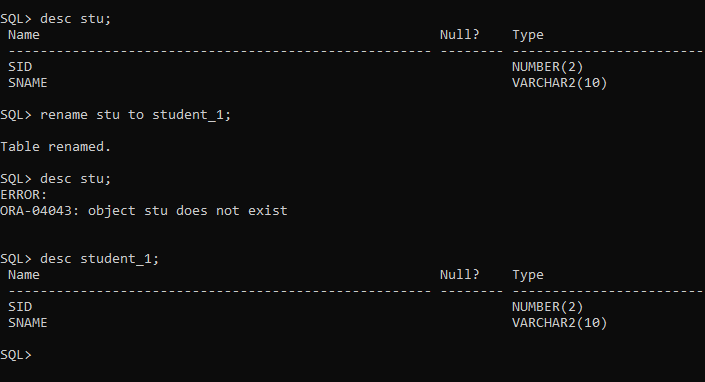
**2)TRUNCATE**:- This command removes all records from the db object. We can't use where clause in the truncate command. Truncate operation can't be roll backed.





**3)RENAME**:- This statement changes dbobjectname.





**4)DROP**:- This stament removes dbobject from database.

If recyclebin feature is enabled , then removed data base object will go to recycle bin.

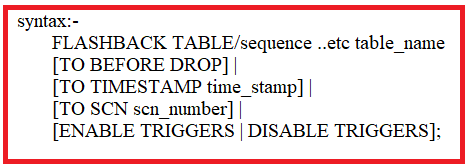
If recyclebin feature is not disabled, then removed data base object will not go to recycle bin and permanently removed.

The removed table will not go to recylebin when purge command is used in drop statement although recyclebin feature has been enabled.





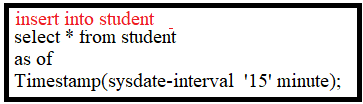
**5)FLASHBACK:-**



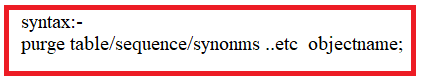
\* when we delete table using drop command, The table is removed from database and it is kept in recycle bin. The flashback restore table from recycle bin to database.

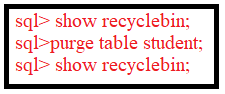


\* when we removed records from the table using truncate or delete command , we can get back the removed records in table in following way.

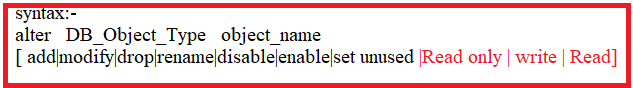


**6)PURGE**:- This statement permanently removes object from recyclebin.

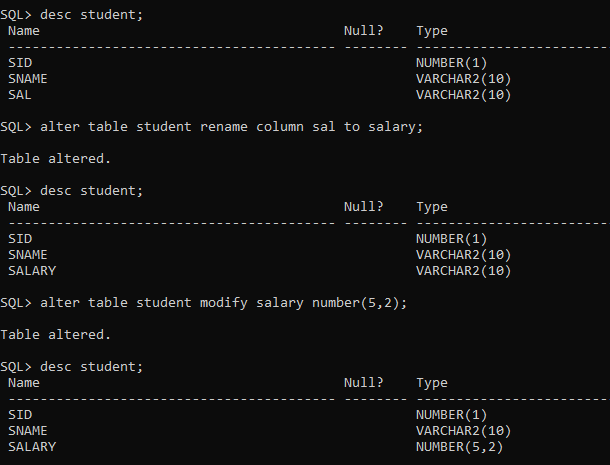


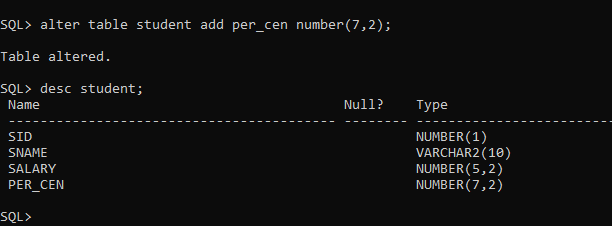


**7)ALTER**:- This command modifies structure of database object.



enable|disable are used to enable constraint and disable constraints respectively.

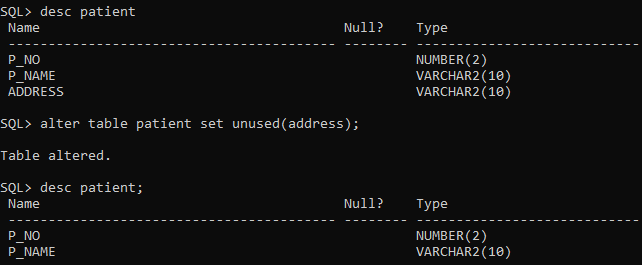


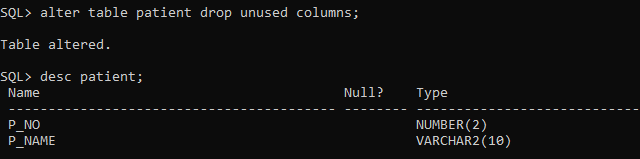




**UnUsed**:- This statement marks one or more columns as unused, but does not actually remove the target column data or restore the disk space occupied by these columns. However, a column that is marked as unused is not displayed in queries or data dictionary views.

Later we can drop unused columns from table structure.





**1.2)DML**:- DML stands for data manipulation language. DML have statements/commands which is used to manipulate data ,select data from database object,keep the data in database object.

The dml statements are

1.insert

2.update

3.delete

4.merge

5.select.

**1.INSERT**:- This command place data in database object.

syntax1:-

insert into tablename(column-list)values(value-list);

syntax2:

insert into tablename values(value-list);

syntax3:

insert into tablename(column-list) select |column-list from tablename [where clause];

(or)

insert into tablename select \* from tablename [where clause];

syntax4:

insert into tablename(value-list)values(&var1,&var2,...);

->@var1,...are substitute variable.when dbms reads substitution variable at runtime. While executing query ,DBMS takes value from user & substitue

corresponding variable with taken value. That value will be placed in corresponding column.

column-list:- The column-list contain one or more than one column which should be in database object.The columns are separated by ,.

value-list:- The value-list contain one ore more than one value which should be compatable with datatypes of appropriate column. The values are separated by ,.

EXAMPLES:-

SQL> create table sukumar(id number(7),name varchar2(7),salary number(7,2));

Table created.

SQL> insert into sukumar(id,name,salary)values(1,'suku',7000);

1 row created.

SQL> insert into sukumar values(2,'veena',9000);

1 row created.

SQL> insert into sukumar values(3,'samba',9000);

1 row created.

SQL> insert into sukumar (id,name)values(4,'rock');

1 row created.

SQL> insert into sukumar (id,name)values(5,'rock',8000);

insert into sukumar (id,name)values(5,'rock',8000)

\*

ERROR at line 1:

ORA-00913: too many values

SQL> insert into sukumar (id,name,salary)values(5,'rock');

insert into sukumar (id,name,salary)values(5,'rock')

\*

ERROR at line 1:

ORA-00947: not enough values

SQL> create table sukumar1(id number(7),name varchar2(7),salary number(7,2));

Table created.

SQL> insert into sukumar1 select id,name from sukumar;

insert into sukumar1 select id,name from sukumar

\*

ERROR at line 1:

ORA-00947: not enough values

SQL> insert into sukumar1 select \* from sukumar;

4 rows created.

SQL> select \* from sukumar1;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

SQL> insert into sukumar values(5,'gov',7000);

1 row created.

SQL> insert into sukumar1(id,name) select id,name from sukumar where id=5;

1 row created.

SQL> select \* from sukumar1;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

5 gov

SQL> insert into sukumar values(&id,&name,&salary);

Enter value for id: 6

Enter value for name: 'sv1'

Enter value for salary: 9000

old 1: insert into sukumar values(&id,&name,&salary)

new 1: insert into sukumar values(6,'sv1',9000)

1 row created.

SQL> /

Enter value for id: 7

Enter value for name: 'sv2'

Enter value for salary: 10000.2

old 1: insert into sukumar values(&id,&name,&salary)

new 1: insert into sukumar values(7,'sv2',10000.2)

1 row created.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

5 gov 7000

6 sv1 9000

7 sv2 10000.2

7 rows selected.

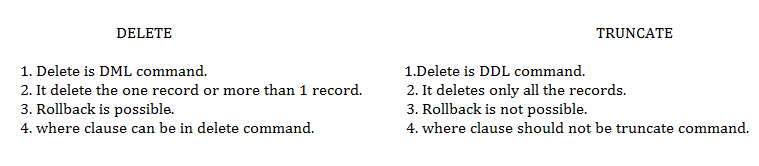
SQL> spool off;

================================================================================================================================================

**2.DELETE**:- This statement removes records from table.

syntax :- delete from table name [where clause];

diff b/w delete and truncate command:



Example:-

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

SQL> delete from sukumar;

4 rows deleted.

SQL> select \* from sukumar;

no rows selected

SQL> rollback;

Rollback complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

SQL> delete from sukumar where id=4;

1 row deleted.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

SQL> rollback;

Rollback complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

SQL> spool off;

==================================================================================================================================

**3.UPDATE**:- This command modifies /updates data in table.

syntax:-

update tablename set columnname= value [,colname2=value ,.....] [where clause];

Example :-

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 9000

3 samba 9000

4 rock

SQL> update sukumar set salary=7000;

4 rows updated.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7000

2 veena 7000

3 samba 7000

4 rock 7000

SQL> update sukumar set salary=7700 where id=1;

1 row updated.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7700

2 veena 7000

3 samba 7000

4 rock 7000

SQL> update sukumar set salary=7700,name='samba' where id=4;

1 row updated.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7700

2 veena 7000

3 samba 7000

4 samba 7700

SQL> spool off;

============================================================================================================================

**4. SELECT** :- This statement selects records from database objects.

syntax:-

select {\*|col-list} from tablename

[where clause

for update

group by clause

having clause

order clause;]

col-list:- It contains the single element or more than one element. It can also contains single row function ,multi row function , case conditional statement, case statements.

element definition:-

{columnname|columnname as aliasname}

Syntax to Case Statement:

Case columnname

When value1 then

Statement1

When value2 then

Statement 2

------

------

Else

Statements

End;

Syntax to case conditional statement:

Case

When condition then

Statement 1

When condition then

Statement 2

---

---

Else

Statements

End;

Example:-

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

1 suku 7700

2 veena 7000

3 samba 7000

4 samba 7700

SQL> select id,name from sukumar;

ID NAME

---------- -------

1 suku

2 veena

3 samba

4 samba

SQL> select id as identity ,name as studentname from sukumar;

IDENTITY STUDENT

---------- -------

1 suku

2 veena

3 samba

4 samba

SQL> spool off;

SQL> select \* from sukumar;

ID NAME SAL

---------- ---------- ----------

1 sukumar 7000

2 veena 6000

2 veena 6000

3 samba 5000

SQL> select id,name,sal,case sal

2 when 7000 then 'first'

3 when 6000 then 'second'

4 else 'third' end

5 as grade from sukumar;

ID NAME SAL GRADE

---------- ---------- ---------- ------

1 sukumar 7000 first

2 veena 6000 second

2 veena 6000 second

3 samba 5000 third

SQL> select id,name,sal,case

2 when sal=6000 then 'second'

3 when sal=7000 then 'first'

4 else 'third' end as grade

5 from sukumar;

ID NAME SAL GRADE

---------- ---------- ---------- ------

1 sukumar 7000 first

2 veena 6000 second

2 veena 6000 second

3 samba 5000 third

1.3)DCL:- DCL stands for data control language.

1.what is privilige & explain them?

A) A privilege is a right to execute a particular type of SQL statement or to access another user's object.The privilege is also called permission.The DBA has all priviliges on db.The dba can give all permissions or some permission to other users .Not only granting priviliges,The dba can cancel all or some permissions to other users.

There are two categories of privileges

1.System Privileges

2.Object Privileges.

\*\*\* Granted privileges information is maintained in USER\_TAB\_PRIVS\_MADE Table.

**1.System privileges**:- The following are few system priviliges.

The following are system privileges to create,alter,delete,drop,update dbobject in another user schema.

a)alter any table

b)alter any index

c)alter any indextype

d)alter any materialized view

e)alter any sequence

f)alter any role

g)alter any database

f)alter any procedure

g)alter user

f)back any table

g)create any cluster ,drop any cluster,alter any cluster.

h)create any table ,delete any table, drop any table.

i)create any sequence,drop any sequence,alter any sequence

j)create any directory ,drop any directory

k)create any index,drop any index,alter any index

l)create any synonym,drop any synonym

m)create any trigger | alter any tigeeer

n)create any view |drop any view

o)flashback any table

p)grant any privilege

q)grant any role

r)lock any table

s)insert any table

t)select any table

u)select any sequence

v)select any dictionary

w) update any table.

x) create session,drop session

y)unlimited tablespace.

The following are system privileges to create,alter,delete,drop,update dbobject in your schema.

a)alter table,create table,drop table

b)alter index,create index,drop index

c)alter sequence,create sequence,drop sequence

d)create dictionary,drop dictionary.

e)alter role,create role,drop role

f)flashback table

g)lock table

**2. Object Priviliges**:- These privileges allows users to do operation on object. The following are object priviliges.

a)insert

b)select

c)update

d)delete

f)alter

g)execute

h)reference

i)index.

The DCL has commands/statements which is

1) grant 2)revoke 3) setrole.

**1. Grant**:- This command grants priviliges to users.

syntax1:

grant systempri-1[,systempri-2,..systempri-3] to user1[,user2,..usern] [with grant];

syntax2:

grant objpri-1[,objpri-2,objpri-3,...objpri-n] on database object to user1[,usr2,..usern][with grant];

**2. Revoke**:- This command take back priviliges from users.

syntax1:

revoke syspri-1[,stspri-2,syspri-3,...syspri-n] from user1[,user2,..usern];

syntax2:

revoke objpri-1[,objpri-2,....objpri-n] on database object from user1[,user2,...usern];

**3.Role**:- Role is group of priviliges.

3.1) create role:-

syntax:- create role rolename [identified by pwd/not identified]

\*not identified is default value.

3.2) Add priviliges to role:-

syntax-1:- grant object-p1[,p2,p3,..pn] on database object name to rolename.

syntax-2:- grant sysp1[,p2,p3...pn] to rolename;

3.3) grant role to users/other roles:-

syntax:- grant rolename to user1|rolename1[,user2|rolename2,user3|rolename3];

3.4) Add password to role:-

syntax:- alter role rolename identified by pwd;

3.5) set role to current user:-

syntax:- set role|all [rolename identified by pwd];

3.6) unset role to current user:-

syntax:- set role none;

syntax:- set role all except rollname;

Examples:-

==============================================================USER-SYS AS SYSDBA==================================

SQL> show user

USER is "SYS"

SQL> create user sv identified by rock

2 default tablespace users

3 quota unlimited on users;

User created.

SQL> grant create session to sv;

Grant succeeded.

SQL> revoke create session from sv;

Revoke succeeded.

SQL> grant create session to sv;

Grant succeeded.

SQL> grant create table to sv;

Grant succeeded.

SQL> revoke create table from sv;

Revoke succeeded.

SQL> grant select on scott.sukumar to sv;

Grant succeeded.

SQL> revoke select on scott.sukumar from sv;

Revoke succeeded.

SQL> create role xyz identified by rock;

Role created.

SQL> grant select,insert,delete on scott.sukumar to xyz;

Grant succeeded.

SQL> grant xyz to sv;

Grant succeeded.

SQL> revoke xyz from sv;

Revoke succeeded.

SQL> spool off;

======================================================USER-SV==================================

SQL> conn

Enter user-name: sv

ERROR:

ORA-01045: user SV lacks CREATE SESSION privilege; logon denied

Warning: You are no longer connected to ORACLE.

SQL> conn

Enter user-name: sv

Connected.

SQL> show user;

USER is "SV"

SQL> select \* from scott.sukumar;

select \* from scott.sukumar

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> create table sample(id number(8));

create table sample(id number(8))

\*

ERROR at line 1:

ORA-01031: insufficient privileges

SQL> create table sample(id number(8));

Table created.

SQL> insert into sample values(9);

1 row created.

SQL> delete \* from sample;

delete \* from sample

\*

ERROR at line 1:

ORA-00903: invalid table name

SQL> delete from sample;

1 row deleted.

SQL> alter table sample modify id number(10);

Table altered.

SQL> desc sample;

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NUMBER(10)

SQL> create table sample1(id number(2));

create table sample1(id number(2))

\*

ERROR at line 1:

ORA-01031: insufficient privileges

SQL> select \* from scott.sukumar;

select \* from scott.sukumar

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> select \* from scott.sukumar;

ID NAME SALARY

---------- ------- ----------

5 rock 80000

7 sv 888

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> insert into scott.sukumar values(3,'rock',9999);

insert into scott.sukumar values(3,'rock',9999)

\*

ERROR at line 1:

ORA-01031: insufficient privileges

SQL> delete from scott.sukumar where id=5;

delete from scott.sukumar where id=5

\*

ERROR at line 1:

ORA-01031: insufficient privileges

SQL> select \* from scott.sukumar;

select \* from scott.sukumar

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> select \* from scott.sukumar;

select \* from scott.sukumar

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> set role xyz identified by rock;

set role xyz identified by rock

\*

ERROR at line 1:

ORA-01924: role 'XYZ' not granted or does not exist

SQL> set role xyz identified by rock;

Role set.

SQL> select \* from scott.sukumar;

ID NAME SALARY

---------- ------- ----------

5 rock 80000

7 sv 888

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> insert into scott.sukumar values(3,'rock',9999);

1 row created.

SQL> select \* from scott.sukumar;

ID NAME SALARY

---------- ------- ----------

5 rock 80000

7 sv 888

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

7 rows selected.

SQL> delete from scott.sukumar where id=5;

1 row deleted.

SQL> select \* from scott.sukumar;

ID NAME SALARY

---------- ------- ----------

7 sv 888

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> select \* from scott.sukumar;

ID NAME SALARY

---------- ------- ----------

7 sv 888

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> delete from scott.sukumar where id=5;

0 rows deleted.

SQL> unset role xyz idetified by rock;

SP2-0734: unknown command beginning "unset role..." - rest of line ignored.

SQL> set role none;

Role set.

SQL> delete from scott.sukumar where id=5;

delete from scott.sukumar where id=5

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> select \* from scott.sukumar;

select \* from scott.sukumar

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> spool off;

**1.4)TCL**:- TCL stands for transaction control language.

It has following commands.

1.commit

2.savepoint

3.rollback

4.set transaction.

**1.commit**:- This command make permanet the changes in db.

syntax:- commit;

**2.rollback**:- This command restore db to original since last commit.

syntax:- rollback [ to savepoint]

**3.savepoint**:-

syntax:- savepoint savepoint name;

This command saves & marks the current point in processing of transaction. single transaction can have multiple savepoints. we can rollback to that particular savepoint whenever nessary.

Example:-

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

7 sv 888

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> delete from sukumar;

6 rows deleted.

SQL> select \* from sukumar;

no rows selected

SQL> rollback;

Rollback complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

7 sv 888

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

7 ab 7780

6 rows selected.

SQL> delete from sukumar where id=7;

2 rows deleted.

SQL> commit;

Commit complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

SQL> savepoint a;

Savepoint created.

SQL> delete from sukumar where id=1;

1 row deleted.

SQL> savepoint b;

Savepoint created.

SQL> delete from sukumar where id=2;

1 row deleted.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

3 rock 9999

3 samba 7000

SQL> rollback b;

rollback b

\*

ERROR at line 1:

ORA-02181: invalid option to ROLLBACK WORK

SQL> rollback to b;

Rollback complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

3 rock 9999

2 veena 7000

3 samba 7000

SQL> rollback to a;

Rollback complete.

SQL> select \* from sukumar;

ID NAME SALARY

---------- ------- ----------

3 rock 9999

1 suku 7700

2 veena 7000

3 samba 7000

SQL> spool off

1.5)CLAUSES:- The sql statement have clauses. The clauses have been divided into two types.

1.Mandatory clauses.

2.optional clauses.

The statement can be processed without having optional clauses. The following are few optional clauses.

1. where clause.

2. Group by clause.

3. Having clauses

4. order by clause.

5. for update clause:- Goto 3rd unit last in DBMS.

Group By clause:- The GROUP BY Statement in SQL is used to arrange identical data into groups. i.e if a particular column has same values in different rows then it will arrange these rows in a group.

* If select statement have both where clause and group by clause , then group by clause should appear after the where clause.

In this case,group by clause is processed after processing where clause.

* The select statement can have group by clause with out having where clause.
* In output, the groups are kept in ascending order by default.
* The columns which is in group by clause does not need to be in select –list.
* The column names which is in select-list except aggregate functions should be in group by clause .but order of columns is not mandatory.

Example:

Table Name: Employee

|  |  |  |
| --- | --- | --- |
| Ename | Dept | Salary |
| Suku | Mca | 7000 |
| Suresh | Mca | 9000 |
| Venkatesh | Cse | 9000 |
| Vasavi | Cse | 19000 |

SQL> select dept,sum(salary) from employee group by dept;

DEPT SUM(SALARY)

---------- -----------

cse 28000

mca 16000

SQL> select dept,max(salary) from employee group by dept;

DEPT MAX(SALARY)

---------- -----------

cse 19000

mca 9000

SQL> select dept,min(salary)as min\_sal from employee group by dept;

DEPT MIN\_SAL

---------- ----------

cse 9000

mca 7000

**Having clause:-**

Syntax:- having qualification.

* The having clause is optional.
* It can appear before the group by clause (or) after the group by clause.
* It is always processed after group by clause.
* It eliminates groups which does not meet qualification of having clause.

Ex:-

SQL> select dept,max(salary)as max\_sal from employee group by dept having max(salary) > 17000;

DEPT MAX\_SAL

---------- ----------

cse 19000

Order by:-

Syntax: order by col,col2,…coln[ASC|DESC]

-> We write this clause to arrange rows of result in ascending order or desc order.

-> It is optional clause.

-> The select statement can have order clause with out having group by and having clause.

-> If select statement have where,groupby,having and order clauses, then order clause is evaluated after processing all other clauses.

SQL> select \* from sailors order by rating;

SID SNAME RATING AGE

-------- ---------- ---------- ----------

4 a 1 50

3 z 1 11

1 x 7 38

2 y 8 35

SQL> select \* from sailors order by rating desc;

SID SNAME RATING AGE

---------- -------- ---------- ----------

2 y 8 35

1 x 7 38

4 a 1 50

3 z 1 11

SQL> select rating,max(age) from sailors group by rating ;

RATING MAX(AGE)

---------- ----------

1 50

8 35

7 38

SQL> select rating,max(age) from sailors group by rating order by rating desc;

RATING MAX(AGE)

---------- ----------

8 35

1. 38

1 50

2.DATA TYPES:-

2.1)Character Data types:- Character data type allows only A-Z,a-z,0-9, and special characters.There are different character data types in oracle. Those are

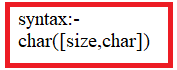
a)char

b)Nchar

c)varchar2

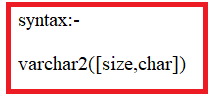
d)Nvarchar2

**a) char**:-



* size is optional.If we don't specify size ,then DBMS allocates only one byte to column.
* The dbms allocates atmost 2000 bytes to char type column.
* It is fixed length data type.(i.e) The data length in that column is less than size, rest of spaces filled with blank spaces.
* If we don't pass second argument,then size is taken as bytes. Only single byte characters are stored in column. single byte character means the character occpies only one byte memory space.
* The column does not allow multibyte character.
* if we pass second argument, then column allows both single byte character and multibyte character.

**b)varchar2**:-



* **size is mandatory**.
* The dbms allocate atmost 4000 bytes to varchar2 column.
* It is variable length data type.
* If we don't pass second argument,then size is taken as bytes. Only single byte characters are stored in column. single byte character means the character occpies only one byte memory space.
* The column does not allow multibyte character.
* if we pass second argument, then column allows both single byte character and multibyte character.
* If character requires 4 bytes space, such character is not allowed by column.

**c) Nchar:-**



* size is optional.If we don't specify size ,then DBMS allocates only one byte to column.
* The dbms allocates atmost 1000 bytes to char type column.
* The size is taken as char instead of byte.Therefore this column allows 1byte character,2byte character,4byte character..etc.
* It is fixed length data type.

**d) Nvarchar2**:-



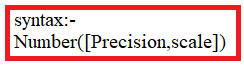
* size is mandatory.
* Dbms allocates atmost 2000 bytes to varchar2 type column.
* It is variable length data type.
* The size is taken as char so column allows 1byte character,2byte character, 4bytes character ...etc.

**II. NUMBER DATA TYPE**:- The numbers are formed with digits 0-9,+,- and (.) decimal pooint. Oracle have following number data types.

1) Number

2) Float.

**1) Number**:-



* Precision is total number of digits in number(integerpart+decimal part)

Ex:- 3338.7879

precision of above number is 8.

precision range is 1 to 38.Default precision is 1.That means column allows only one digit number.

* Scale is no.of digits after decimal point.

Ex:- 338.7879

scale of above number is 4.

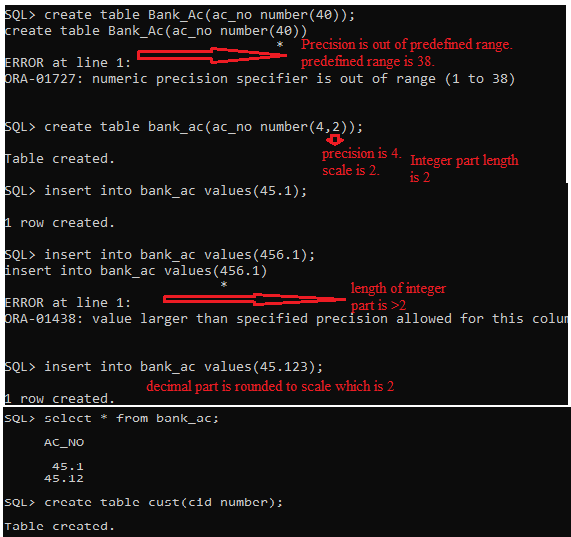
Both arguments are optional.

DBMS allocates atmost 22 bytes.

Note:- 1. Id colum will contain both number which has either integer part or both integer part and decimal part.

2.Precision is 7-3=4. So integer part has atmost 4digits(9999).If it contain more than 4 digits , we get error message.

3.scale is 3. So decimal part has atmost 3 digits(999).If decimal part has more than 3 digits,then decimal part is only rounded.



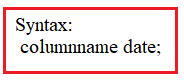
**III.DATE&TIME data type:-**

There are two date types .

1.DATE

2.TIME STAMP.

1. **DATE**:- The oldest calender was julean calender. There ws no calender before julean calender. that calender started from 01-Jan-4712BC.There was't historical records before that day. Thatwhy oracle took that date as starting date.



* date type column value occupies 7 bytes.
* The date type value is internally stored in the following format.

BC/AD DD-MON-YYYY HH:MM:SS AM/PM.

* RANGE of Date type value is 01-jan-4712BC TO 31-DEC-9999AD.

Q.How do u know default date format?

SQL> select value from V$NLS\_PARAMETERs where parameter='NLS\_DATE\_FORMAT';

VALUE

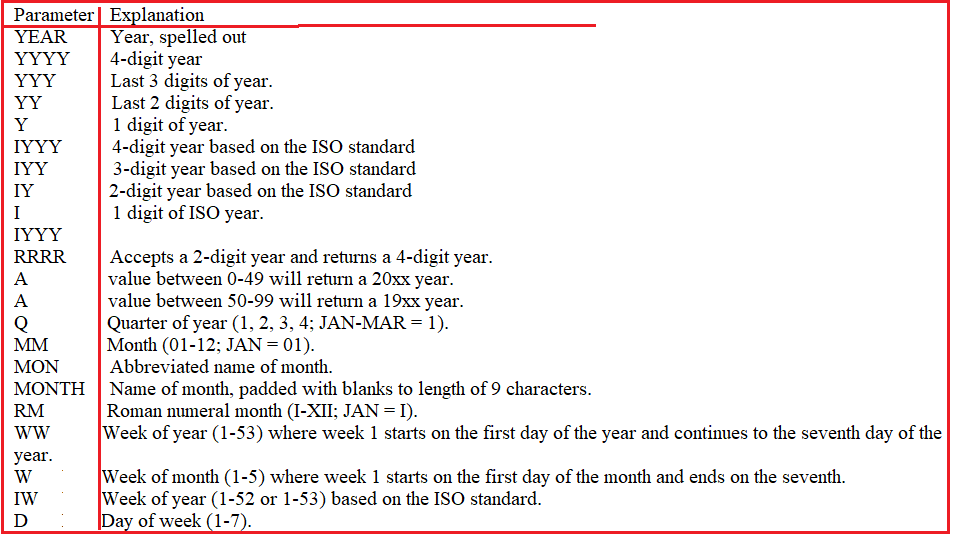
----------------------------------------------------------------

DD-MON-RR

SQL> prompt "Change date format for current session"

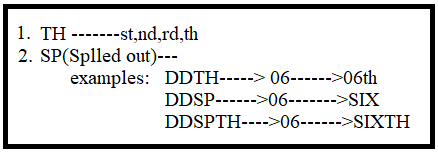
"Change date format for current session"

SQL> alter session set NLS\_DATE\_FORMAT='DD-MM-YYYY';

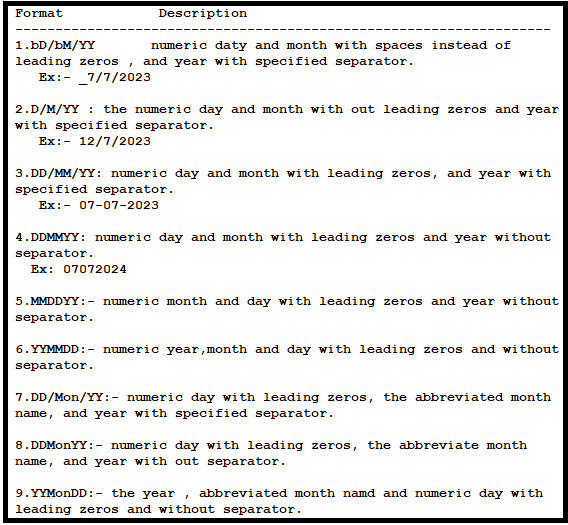




Parameter successors:-



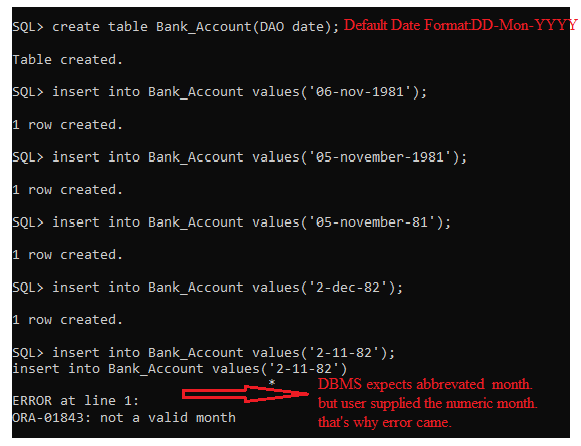
Examples to Date format:

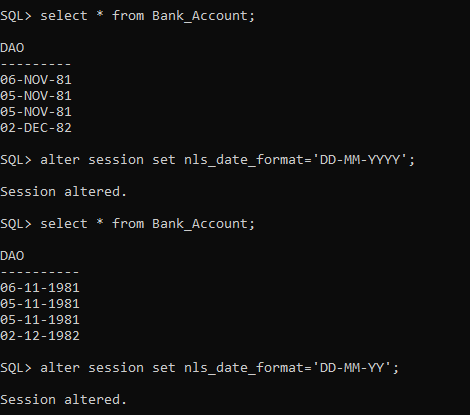


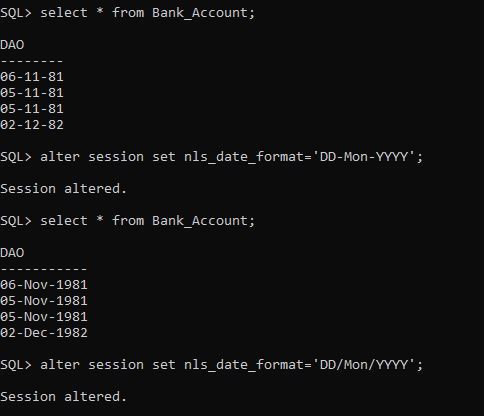
Note:-

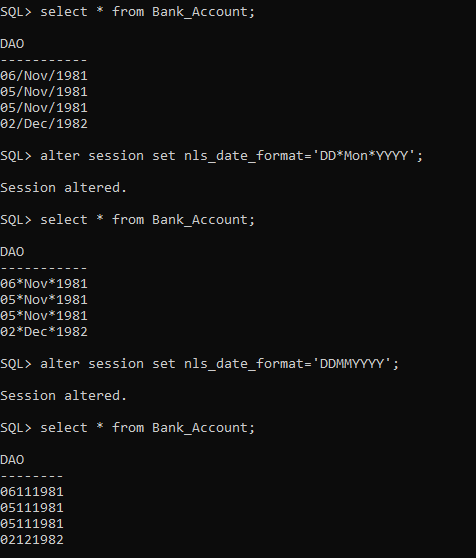
1. mm can match with numeric month , abbreviated month and full month name.
2. Mon can only match with abbreviated month and full month name.

**Display the date in Required Format:**

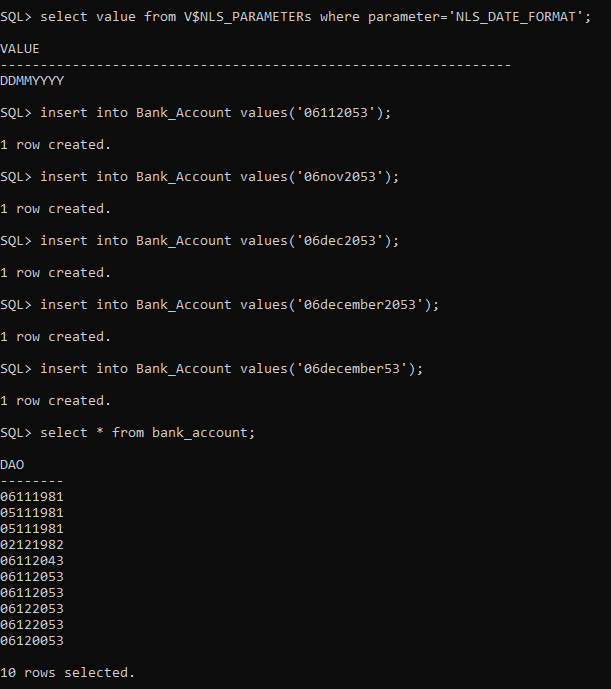


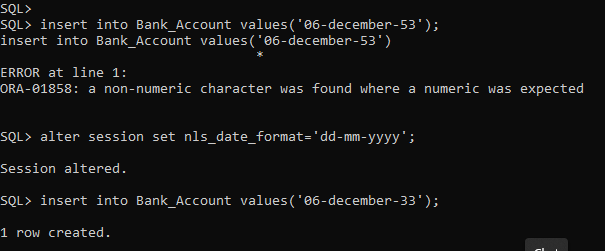


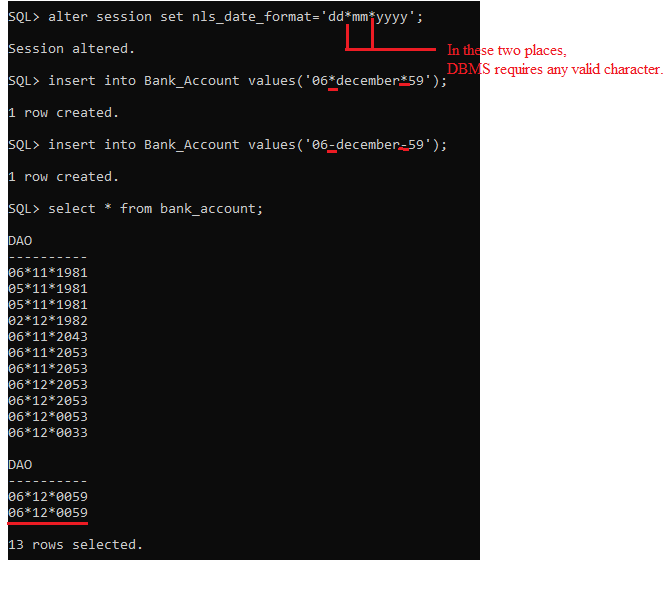




**Insert data into Date type column:**







**2.TIMESTAMP**:-

* By default,The date type value format is 'DD-MM-RR'.SO we can only store date in the date type column. we can't store time in date type column.
* But If we changed format using NLS\_DATE\_FORMAT,then only we can keep both time and date in date type column.
* We can store only date and both date and time in Timestamp column. **But we can’t store only time in Timestamp column.**

1) The default format of timestamp is

dd-mm-rr [hh:mi:ss[.fs] [AM/PM]]

we can store both time and date in timestamp column.



francational seconds range is from 1 to 9.

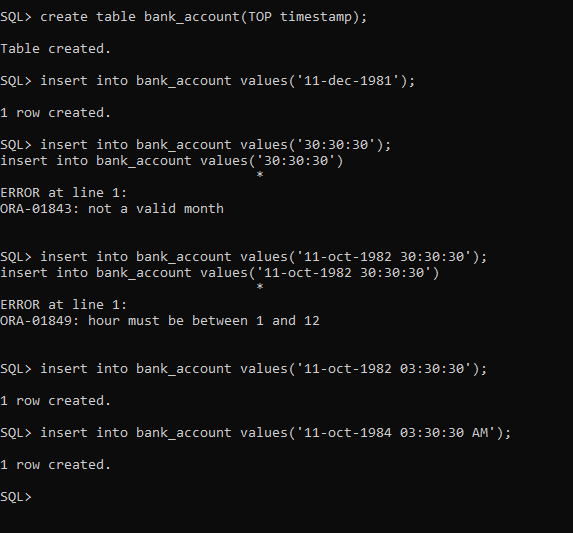
SQL> select value from V$NLS\_PARAMETERS WHERE PARAMETER='NLS\_TIMESTAMP\_FORMAT';

VALUE

----------------------------------------------------------------

DD-MON-RR HH.MI.SSXFF AM

SQL> alter session set NLS\_TIMESTAMP\_FORMAT='MM-DD-YYYY HH:MI:SSXFF PM';



**III) LONG**:-

syntax:- column name Long;

1) It variable length data type.

2)Dbms allocates atmost 2GB space for long column.

3)The table can has atmost one long column.

4)This type column allows only single byte characters.

**IV) RAW & LONG RAW**:- Both are depricated data types in oracle 9i.

**V) LOB**:-Large object Binary Files are a set of data types that are designed to hold large amounts of data. A LOB can hold up to a maximum size ranging from 8 terabytes to 128 terabytes depending on how your database is configured.

Storing data in LOBs enables you to access and manipulate the data efficiently in your application. Lobs stores structured and unstructured data.

The LOB types are

1. BLOB

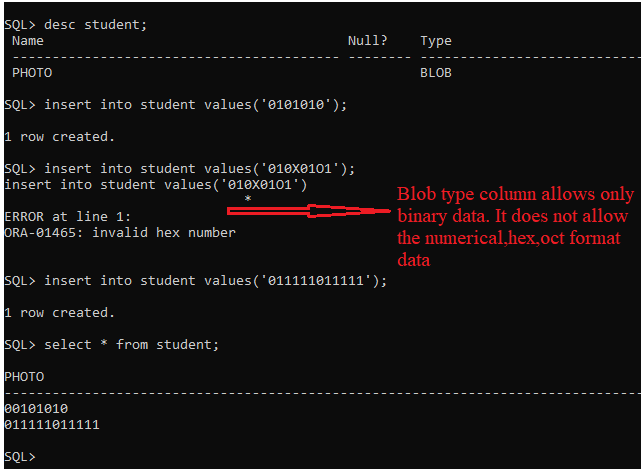
2.CLOB

3.BFILE.

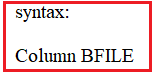
**1)BLOB**:-



* Blob column stores binary data(images,audio,video,hexadecimal data or Null.
* Single table can have more than one blob columns.
* Dbms allocates atmost 4gb space to column.
* we can't define constraint over BLOB column Except NOTNULL.
* Blob is internal lob.(i.e) semistructured and unstructured data is stored in column of table. which is not stored in file system of OS.

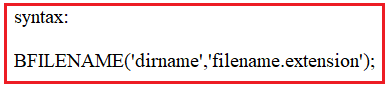


**2)BFILE**:- The column which is Bfile type stores only LOCATOR of binary files(imagefiels,audiofiles,vidoefiles),txtfiles,zipfiles and csv files.Actual files are in Opeating system file system instead of database.That'why Bfile type is called external lob.

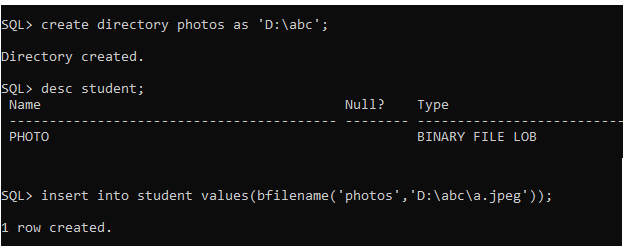


* Bfile column does't participate in transaction.
* we can't define constraints over Bfile column except NOT NULL.
* This column allows null or ' '.

To keep locator of file, we use the BFILENAME function.



It retruns locator of binary.



**VI) Data Type Conversion:-** Converting one data type into another data type is called conversions.

Oracle has 2 types of conversion:

1.Implicit conversion

2.Explicit conversion

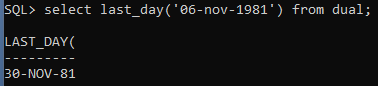
1.Implicit conversion:-

a. In oracle when an expression contains string representing pure number(eg:’123’) then oracle server automatically converts “string type” into “Number type”.

b. In oracle whenever we are passing number into character functions then oracle server automatically converts number type into character type.

c. Whenever we are passing DATE string into Date functions then oracle server automatically converts “Date String” into “Date Type”. But passed parameter must be in “Default Date Format”.





2. Explicit Conversion:- we should use function to convert the one data type into another data type.

Example:-

1. To\_date(-,-)
2. To\_char(-,-)
3. To\_number(-)
4. To\_timeStamp(-)

**3)OPERATORS**:- The operator is keyword in sql and pl/sql.The each operator has special meaning and own functionality.The operator is single character,word & group of characters.The operator has been divided into 3 types.

1.Unary operators.

2.Binary operators.

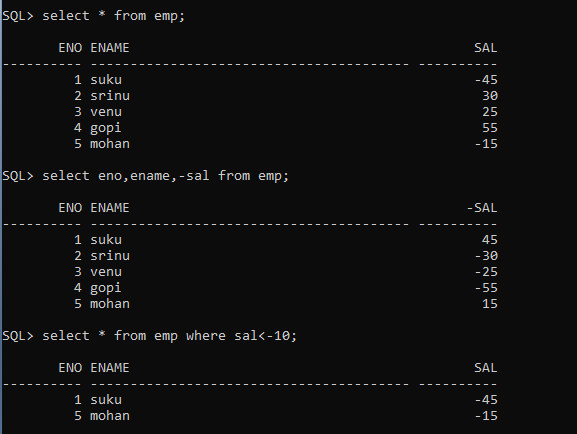
3.Set operators.

**3.1)Unary operators**:- Unary operator associates with only one operand.This unary operators has highest priority than binary and set operators.

The unary operators are unaryplus(+) and unary minus(-).



* unary plus makes positive the given operand.
* unary minus make negitive the given operand.



**3.2)BINARY OPEATORS**:-



Binary operators always receives two operands. The binary operator is always between two operands. The binary operator has highest priority than set operators but has low priority than unary operators.

The binary operators are

1) Arithmetic operators.

2) Concatenation operators.

3) Single valued relational operators.

4) Multi valued relational operator.

5) Logical operators.

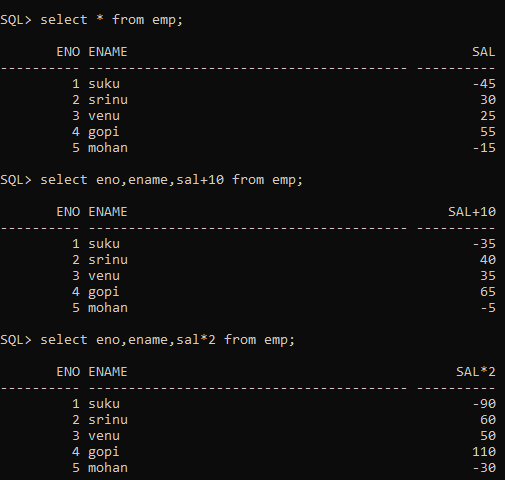
1.Arithmetic operators:- The arithmetic opreators are +,-,\*,/.

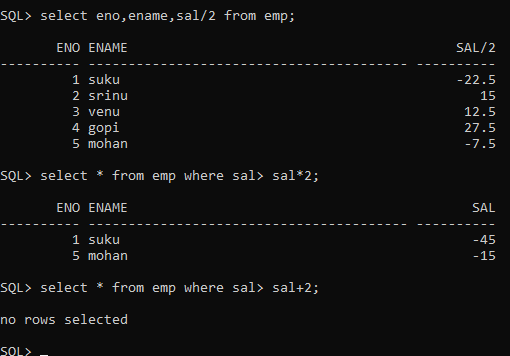
\*,/ has highest priority than +,-.

Associativity is left to right.

oracle does not has % operator.

Example:-



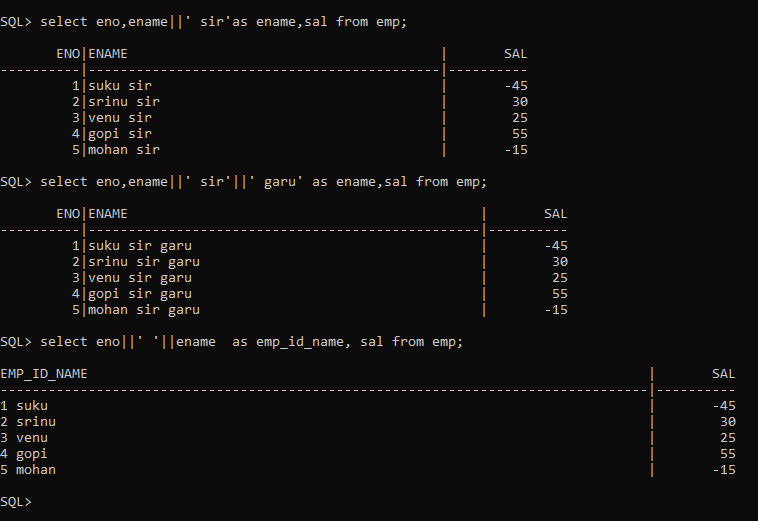


2. CONCATENATION:- This operator concats two columns.The operator is (||).

The result of concatenation is character expression.



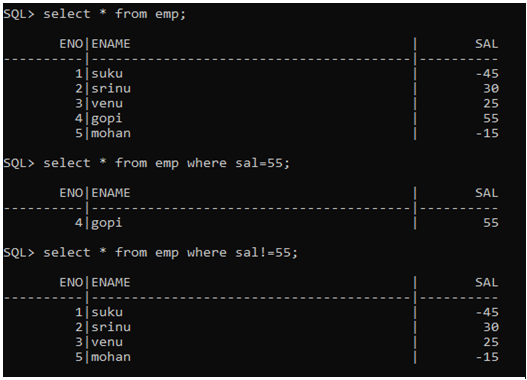
Example:-

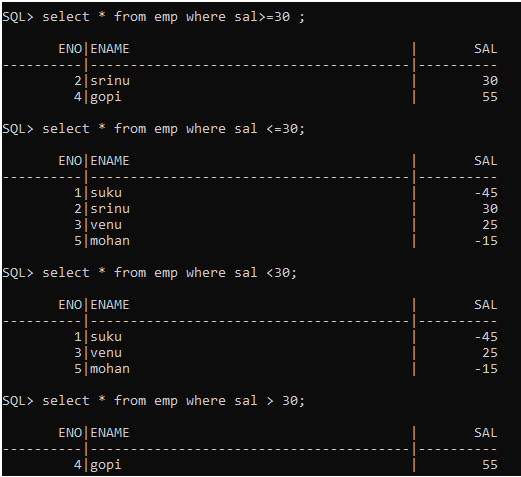


3. SINGLE VALUE RELATIONAL OPERATOR:- The operators are =,!=,<,>,>=,<=,<>.

These operators compares left side operand with only one operand which is rightside to operator. Therefore these oprators are called as single value relational operator.

Example:-

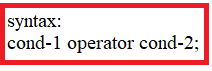




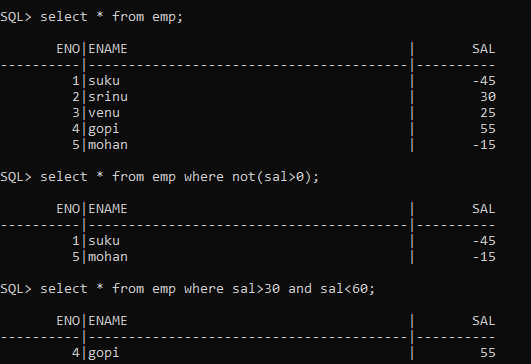
4.LOGICAL OPERATORS:-

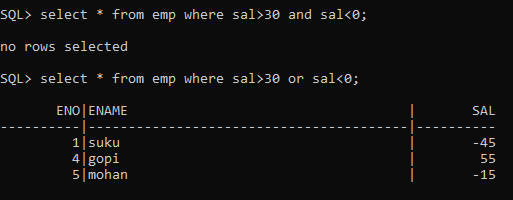
The logical operators are

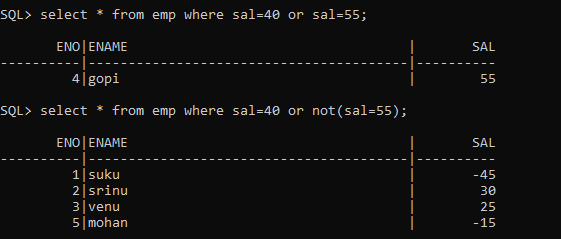
1)NOT 2) AND 3 )OR.



Examples:-







5.MULTIVALUED RELATIONAL OPERATORS:- The operators compars liftside operand with more than one values.

Therefore these operators are called as multivalue relational operator. The operators are

1)IN

2)NOT IN

3)ALL

4)ANY

5)EXISTS

6)BETWEEN

7)NOT EXISTS

8)LIKE

9)NOT LIKE.

10)NOT BETWEEN.

SYNTAX:- operand relational operator mulivalue relational operator(list of values);

1.ALL:- It returns true ,if condition becomes true for everyvalue in list.

2.ANY:- It returns true, if condition becomes true for atleast one value in list.

3.IN:- It returns true, if operand value is equal to atleast one value in list.

4.NOT IN:- It return true, if operand value is not equal to any value in list.

5.BETWEEN:- IT returns true,if operand value is with in range which incluse both min and max range.

6.NOT BETWEEN:- it return true, if operand values is not with in range.

7.LIKE:- There are two wild characters.

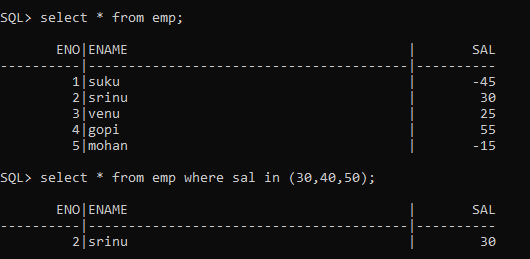
% matches any number of characters.

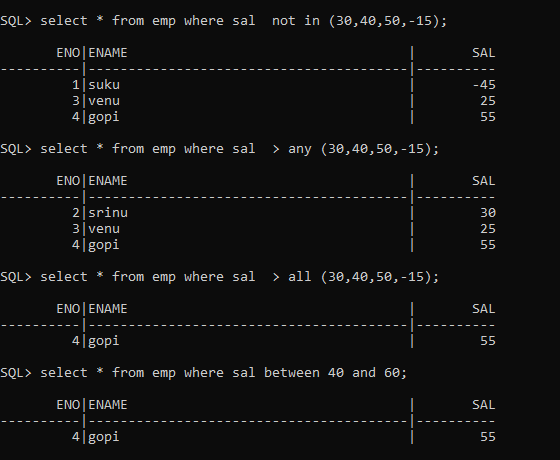
- matches a single character.

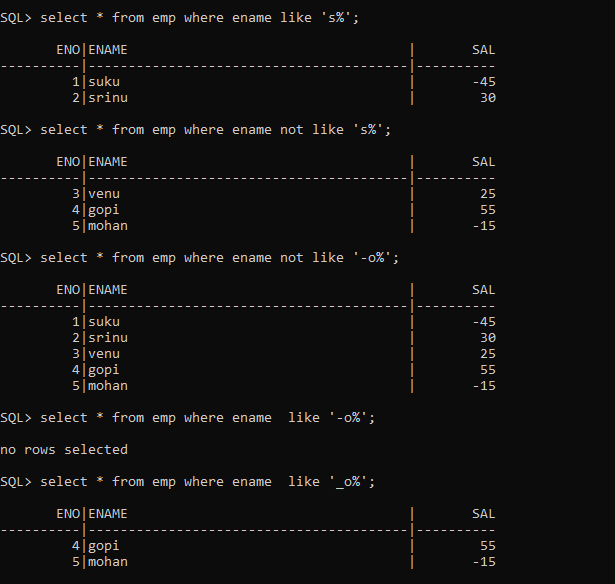
8. EXISTS:- This operator returns true if subquery returns record; otherwise it returns false.

syntax:- where [not] exists(subquery);

EXAMPLES:-







**3.3)SET OPERATORS**:- The following standard operations on sets are also available in relational algebra:

*union* (*U*)

*intersection* (*∩*)

*set-difference* (*−*)

and *cross-product* (*×*).

Relation Name: Cricket (R)

|  |  |  |
| --- | --- | --- |
| Id | Name | Remuneration |
| 1 | X | 1000 |
| 2 | Y | 2000 |
| 3 | Z | 3000 |

Relation Name: Football(S)

|  |  |  |
| --- | --- | --- |
| Id | Name | Remuneration |
| 001 | X | 1000 |
| 002 | A | 7000 |
| 003 | B | 6500 |

The union-compatible which is rule must be satisfied by all binary operators except cross product.

R&S are two distinct relations.

If the following two conditions holds, R&S are union compatible.

1. They have same no.of columns.The column names may be

different.

1. Corresponding fields ,from left to right, have same domain.

**Union**: *R U S* returns a relation instance containing all tuples that occur in *either*relation instance *R* or relation instance *S* (or both). *R* and *S* must be *union-compatible*, and the schema of the result is defined to be identical to the schema of *R*.

Syntax: RUS.

SQL QUERY:-

Sql> select id,name,remuneration from cricket union select id,name,remuneration from football;

Ex:- Cricket U FootBall

|  |  |  |
| --- | --- | --- |
| Id | Name | Remuneration |
| 1 | X | 1000 |
| 2 | A | 7000 |
| 3 | B | 6500 |
| 2 | Y | 2000 |
| 3 | Z | 3000 |

**Intersection**: *R∩S* returns a relation instance containing all tuples that occur in *both R* and *S*. The relations *R* and *S* must be union-compatible, and the schema of theresult is defined to be identical to the schema of *R*.

Syntax: R *∩* S.

sqlQuery:

sql> select id,name,remuneration from cricket Intersect select id,name,remuneration from football;

Ex:- Cricket ∩ FootBall

|  |  |  |
| --- | --- | --- |
| Id | Name | Remuneration |
| 1 | X | 1000 |

**Set-difference**: *R−S* returns a relation instance containing all tuples that occur in *R* but not in *S*. The relations *R* and *S* must be union-compatible, and the schema of the result is defined to be identical to the schema of *R*.

Syntax:- R-S

**SQL QUERY:-**

Sql > select id,name,remuneration from cricket minus select id,name,remuneration from football;

Ex:- Cricket - FootBall

|  |  |  |
| --- | --- | --- |
| Id | Name | Remuneration |
| 2 | Y | 1000 |
| 3 | Z | 3000 |

**Cross-product**: *R × S* returns a relation instance whose schema contains all the fields of *R* (in the same order as they appear in *R*) followed by all the fields of *S* (in the same order as they appear in *S*). The result of *R × S* contains one tuple *〈r, s〉* (theconcatenation of tuples *r* and *s*) for each pair of tuples *r∈R, s∈S*. The cross-productoperation is sometimes called Cartesian product.

Syntax:-R X S

**SQL Query:**

Sql> select \* from cricket ,football;

Ex: Cricket XFootBall

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Name | Remuneration | Id | Name | Remuneration |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

4)CONSTRAINTS:- The constraint is business rule (or) restriction. The constraints can be defined on single column,multiple column and table.

syntax:- constraint constraint-name.

This syntax is optional. If you don't specify, dbms creaes constraint name. That constraint name start with SYS\_C------.This constrint informationis not stored in table. That information is stored in **user\_constraint** view.

Q.)What is table-level/out-of-line style constraint?

The constraint which can be defined after all column definition in create statement is called table-level constraint.

EX:- create table tablename( col1-def, col2-def,...coln-def,constraint definition);

Q.)what is column-level/in line style constraint?

The constraint which is immediately defined after column definition in create statement is called column-level constraint.

Ex:- create table tablename(col1-def constraint definiton, col2-def,....col3-def);

In oracle, constraints have been devided into 3 types.

1.DOMAIN INTEGRITY CONSTRAINTS.

2.ENTITY INTEGRITY CONSTRAINTS.

3.REFERENTIAL INTEGRITY CONSTRAINTS.

1.DOMAIN INTEGRITY CONSTRAINTS:- These constraints validates entries for given column.

The constraints are

a)default

b)not null

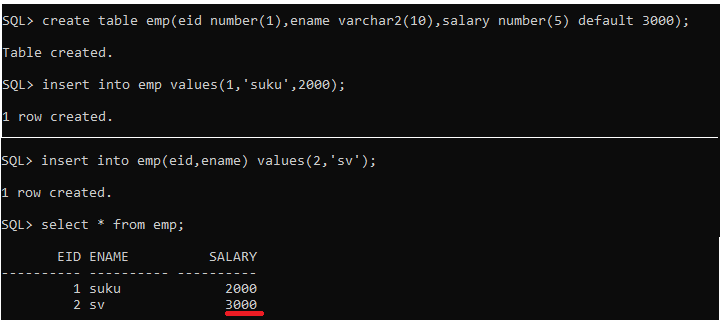
c)check

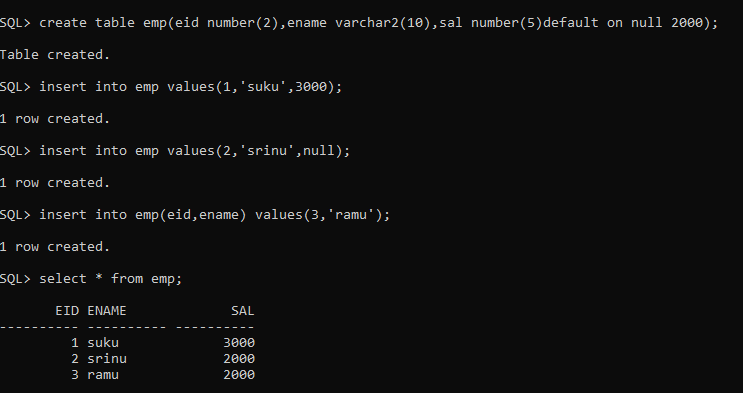
d) type of value is restricted through data type of column.

a)Default:-



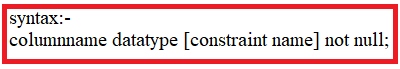
* it can be defined at only column-level
* If we don't specify value for column ,DBMS inserts default value into that column.
* If i specify value for column, the specified value has high priority than default value. so DBMS inserts specified value in column.
* In single table, more than one column can have default constraints.





Note :- we can't remove existing default constraint.

**b)NOTNULL**:-



* IT is column level constraint.
* In single table, more than one column can have default constraint.
* The column does't allow null values .If we try to insert Null ,we get error message.
* Default constraint has more priority than notnull constraint.The single column can has both constraints.If we want to define both constraints on single colun,then we should first define default and next define notnull constraint.

Example:-

SQL> create table sample(id number(7),name varchar2(10) constraint abc not null);

Table created.

SQL> desc sample

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NUMBER(7)

NAME NOT NULL VARCHAR2(10)

SQL> insert into sample values(7,null);

insert into sample values(7,null)

\*

ERROR at line 1:

ORA-01400: cannot insert NULL into ("SCOTT"."SAMPLE"."NAME")

SQL> alter table sample drop constraint abc;

Table altered.

SQL> insert into sample values(7,null);

1 row created.

SQL> desc sample

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NUMBER(7)

NAME VARCHAR2(10)

SQL> alter table sample modify name varchar2(10)constraint abc not null;

alter table sample modify name varchar2(10)constraint abc not null

\*

ERROR at line 1:

ORA-02296: cannot enable (SCOTT.ABC) - null values found

SQL> select \* from sample;

ID NAME

---------- ----------

7

SQL> delete from sample;

1 row deleted.

SQL> alter table sample modify name varchar2(10) constraint abc not null;

Table altered.

SQL> desc sample;

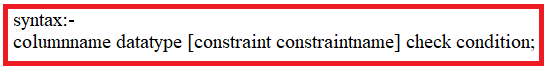
Name Null? Type

----------------------------------------- -------- ----------------------------

ID NUMBER(7)

NAME NOT NULL VARCHAR2(10)

**C)CHECK**:-



* + This constraint is both column level constraint & table level constraint.
  + when dbms inserts value in to column,dbms examines condition. If condition is true, DBMS places value into column; otherwise it displays error message.
  + The default constraint has more priority than not null & check constraint.Not null &check constraint have same priority.
  + when we want to define 3 constraints on single column,we first should define default constraint.Later not null & check can be defined in any order.

EXAMPLE:-

SQL> create table sample (id number(7),sal number(7)constraint abc check (sal>7000));

Table created.

SQL> insert into sample values(7,8000);

1 row created.

SQL> insert into sample values(7,6000);

insert into sample values(7,6000)

\*

ERROR at line 1:

ORA-02290: check constraint (SCOTT.ABC) violated

SQL> alter table sample drop constraint abc;

Table altered.

SQL> insert into sample values(7,6000);

1 row created.

SQL> alter table sample modify sal number(7)constraint abc check(sal>5000);

Table altered.

SQL> insert into sample values(7,4000);

insert into sample values(7,4000)

\*

ERROR at line 1:

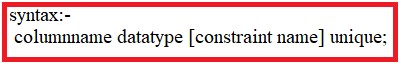
ORA-02290: check constraint (SCOTT.ABC) violated

**II.ENTITY INTEGRITY CONSTRAINTS**:- The constraints which is defined at object level is called entity integrity constraint. There are two entity intgrity constraints.

1.UNIQUE key

2.PRIMARY key.

1.UNIQUE KEY:-



* when we define unique constraint on column,that column does not allow duplicate value. But column allows null value.
* unique key constraint can be defined at column level or table level.
* The table allows more than one unique key constraints on different columns.
* we can define unique key constraint on single column or multiple columns. The multiplicolumns all together is called composite key.

NOTE:- 1.WHEN we want to define unique key constraint on single column, it can be defined at column level or table level.

2. when we want to define unique key constraint on multiple column, it should only be defined at table level.

EXAMPLE:

SQL> create table sample(id number(7) constraint abc unique,name varchar2(7));

Table created.

SQL> insert into sample values(7,'sula');

1 row created.

SQL> insert into sample values(7,'sula');

insert into sample values(7,'sula')

\*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.ABC) violated

SQL> insert into sample values(null,'sula');

1 row created.

SQL> select \* from sample;

ID NAME

---------- -------

7 sula

sula

SQL> alter table sample drop constraint abc ;

Table altered.

SQL> insert into sample values(7,'sula');

1 row created.

SQL> select \* from sample;

ID NAME

---------- -------

7 sula

sula

7 sula

SQL> alter table sample modify id number(7) constraint abc unique;

alter table sample modify id number(7) constraint abc unique

\*

ERROR at line 1:

ORA-02299: cannot validate (SCOTT.ABC) - duplicate keys found

SQL> delete from sample;

3 rows deleted.

SQL> alter table sample modify id number(7) constraint abc unique;

Table altered.

SQL> insert into sample values(7,'suku');

1 row created.

SQL> insert into sample values(7,'suku');

insert into sample values(7,'suku')

\*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.ABC) violated

SQL> create table sample (id number(7),name varchar2(7),sal number(7) ,constraint abc unique(id,sal));

Table created.

SQL> insert into sample values(7,'suku',7000);

1 row created.

SQL> insert into sample values(7,'suku',7000);

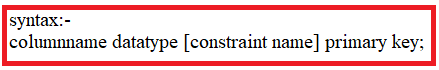
insert into sample values(7,'suku',7000)

\*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.ABC) violated

**2.PRIMARY KEY**:-



* + when we define primary key constraint on column,that column does not allow duplicate value and null value.
  + primary key constraint can be defined at column level or table level.
  + The table allows more than one primary key constraints on different columns.
  + we can define primary key constraint on single column or multiple columns. The multiplicolumns all together is called composite key.

NOTE:- 1.WHEN we want to define primary key constraint on single column, it can be defined at column level or table level.

2. when we want to define primary key constraint on multiple column, it should only be defined at table level.

Difference between unique key and primary key:

1. unique key does't allow duplicate value. 1.Primary key does't allow duplicate row.

2. It allows null value. 2.it does't allow null value.

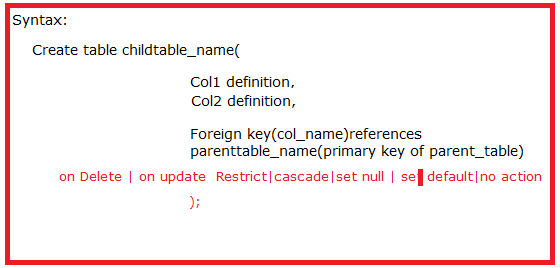
3. It can be defined on single column and multiple column 3.It can be defined on single column and multiple column

**III.REFERENTIAL INTEGRITY CONSTRAINTS**:-

**1.FOREIGN KEY:-** The referential integrity constraint is specified between two tables .

Foreign key constraint:-A Foreign key is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another relation. The foreign key is used to establish and enforce link between the data in two relations to control data that can be stored in the foreign key.The table with the foreign key is called the child table(referencing), and the table with the primary key is called the referenced or parent table.

The primary key name in referenced table and foreign key name in refrencing table need not to be same. But their data type must be same.



**CASCADE:** Delete or update the row from the parent table and automatically delete or update the matching rows in the child table.

**SETNULL:** Delete or update the row from the parent table and set the foreign key column or columns in the child table to **NULL**. **The foreign key must allow the null values.**

**RESTRICT:** Rejects the delete or update operation for the parent table.

**SET DEFAULT: oracle does not support this clause.**

**NoAction is default value.**

Example:

Create table student(roll\_no number(2) primary key,

Sname varchar2(10),

Dob date,

Age number(2),

M\_no number(11));

Create table address(roll\_no number(2) primary key,

Village varchar2(5),

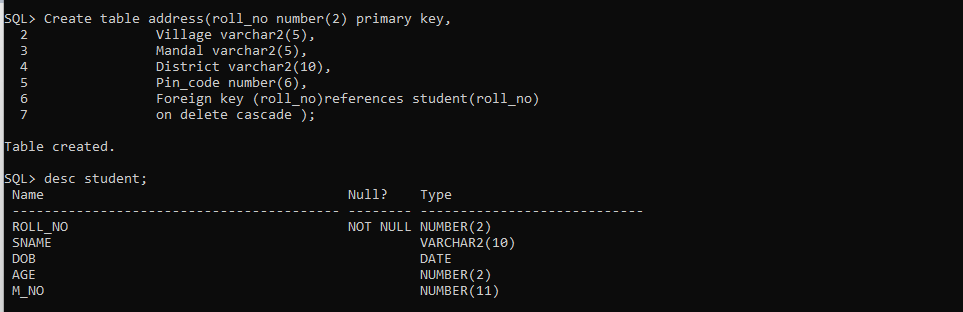
Mandal varchar2(5),

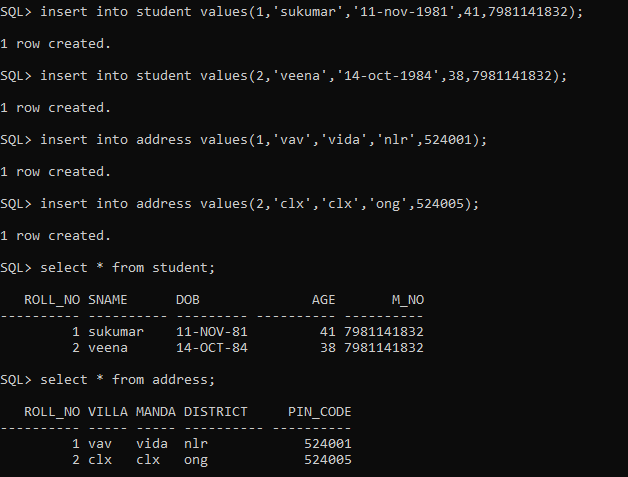
District varchar2(10),

Pin\_code number(6),

Foreign key (roll\_no)references student(roll\_no));

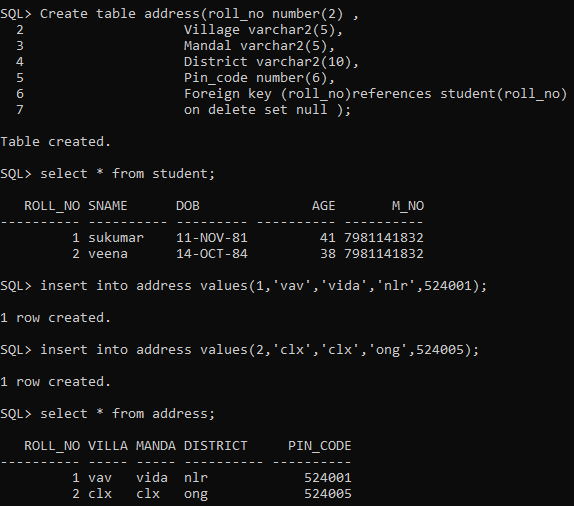
Case1: On delete cascade

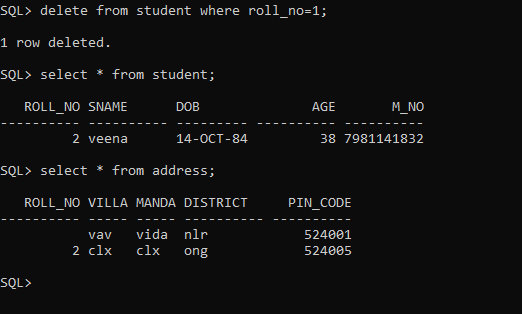




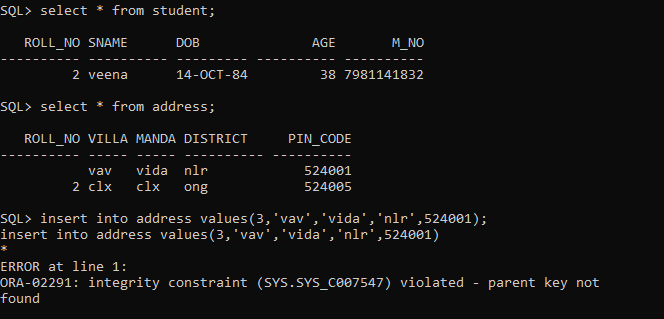


**Case:2 On delete set null**

****

****

Note:- Without student record in student table, we can’t keep student address in address table.



**Note:- truncate table student;**

**Error:Unique primary key in table referenced by enabled foreign keys.**

**Solution:**

**SQL> truncate table student cascade.**

**To work it the child table “Foreign key “ must have “on delete cascade” clause.**

5)SQL FUNCTIONS:- The functions which we are going to learn are built-in functions. when we inovke function, the functionality(particular task) is performed. In sql, built-in functions have been usually classified into 2 categories. The categories are

1.Single-row functions.

2.Multi-row functions.

**1.Single-row functions**:- The functions act on each single row and gives one output per each row. The single row function can be nested upto any level. It can be used in select, where and order by clause.

**2. Multiple-row functions**:- The functions acts on group of rows and returns one output per each group. The multiple-row functions can be maximum nested up to 2 levels. They can appear in select, order by, and having clause. These functions ignores null values.

The sql built-in functions have also been divided into 6 types based on their input data.

1.character functions.

2.number functions

3.date functions.

4.conversion functions

5.general functions

6.Time stamp functions.

**5.2.1)CHARACTER FUNCTIONS**:-

1)UPPER(COLUMNNAME/STRING)

2)LOWER(COLUMNNAME/STRING)

3)INITCAP(COLUMNNAME/STRING,SEPARATOR)

It converts starting letter of every word into uppercase. In string, word is identified based on space,(.),(,) and any special character.

4) Length (column/string): This function returns length of string or no.of characters in given string.

5) concat(columnname1/string1,columnname2/string)

This function joins only two strings. It returns joined string.

Diff b/w concat operator and concat function.

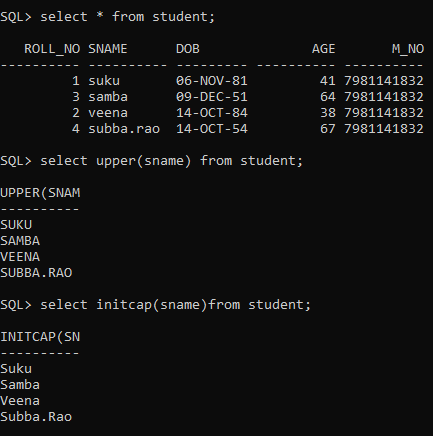
1) concat function can joins only two strings.

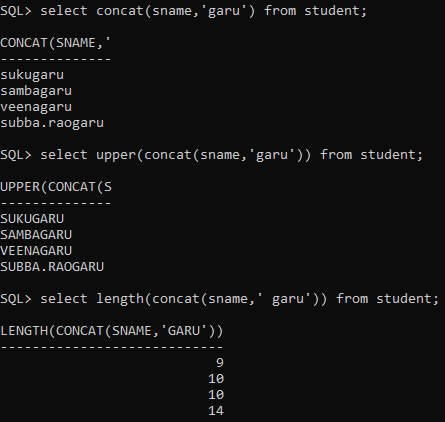
2) concat operator can join more than 2 strings.

if we want to join more than two n strigsusing concat function then we should use nested concat functions.

Ex:- concat(concat('suku','mar'),'rajaka');

o/p: sukumar rajaka.



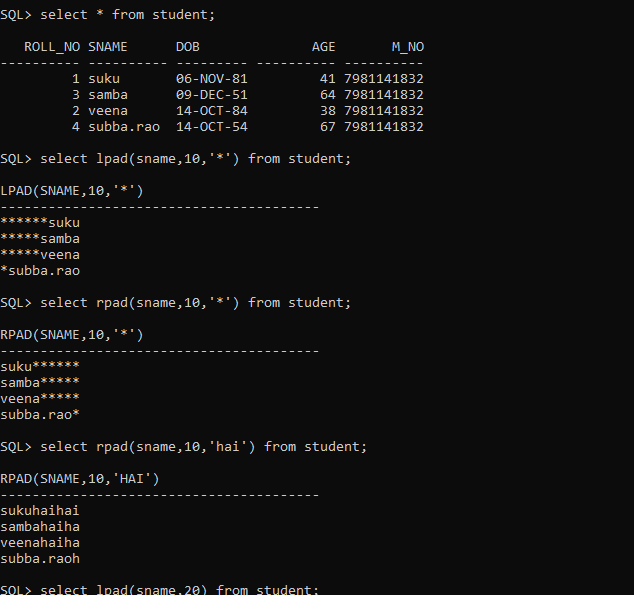


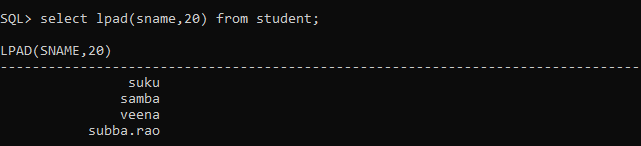
6. PADDING:- padding means adding extra characters at left,right or both sided of value in column.

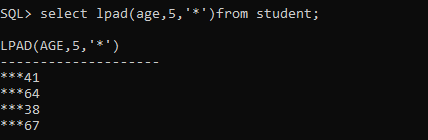
6.1)LPAD (columnname/string,n,character);

6.2)RPAD (columnname/string,n,character);

* Third parameter is optional. It is single character which is to be added to given string. The default character is 'space'.
* Second parameter is integer value. It specify that total length of value.
* When length of first argument is < second argument, the rest of place in output are filled with 3 rd argument to leftside|rightside of 1 st argument in output.





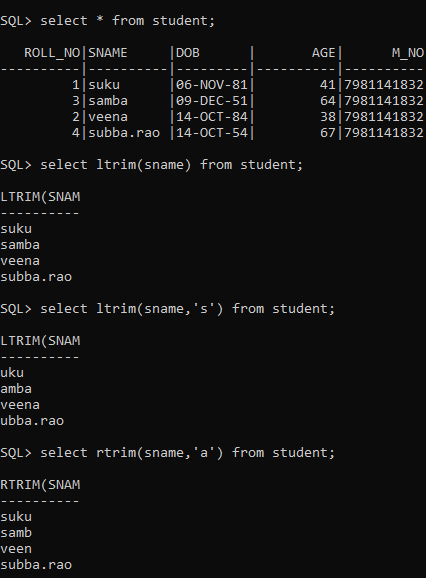


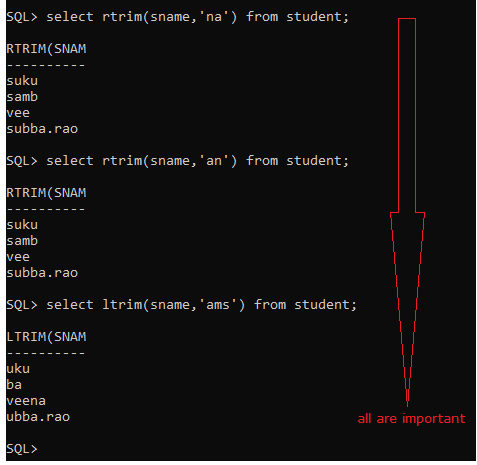
7. TRIMMING:- Trimming is removing unwanted/unnecessary character[s] from right most,left most or both sides of given string. By default , ‘space’ is unwanted character.

7.1) LTRIM(column name,trim chatacter|trim string)

7.2) RTRIM(column name,trim character|trim string)

7.3)TRIM(leading/trailing/both,'character' from columnname/string);



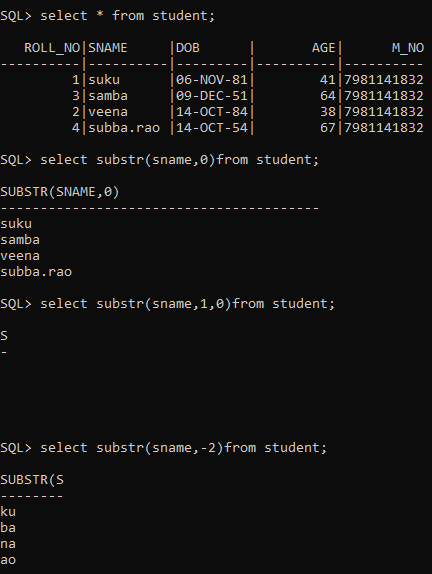


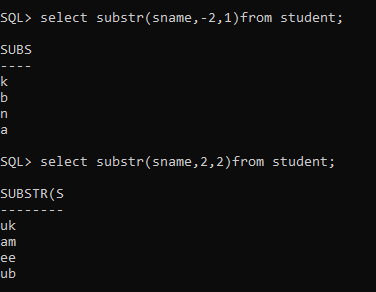
8. SUBSTR:-



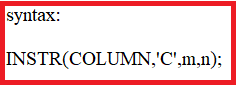
* This function returns char or string type.
* It extracts portion of string from m position.
* If m=0 , it is treated as 1. M is either +,-.
* n specifies no.of characters,if m is positive.
* if n is not specified, oracle returns all char from m to end of string.
* if n is 0 or <0 , null is returned.
* n is either float or int.
* if m is negative,then extract m characters from right .
* if m is negative and n is not zero or negative then we get only n character from right.

Examples:-

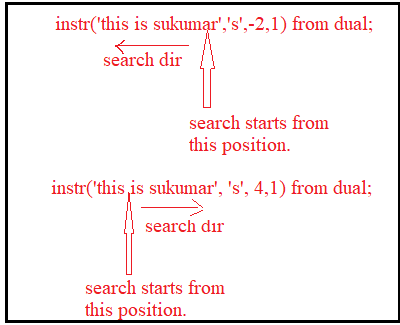




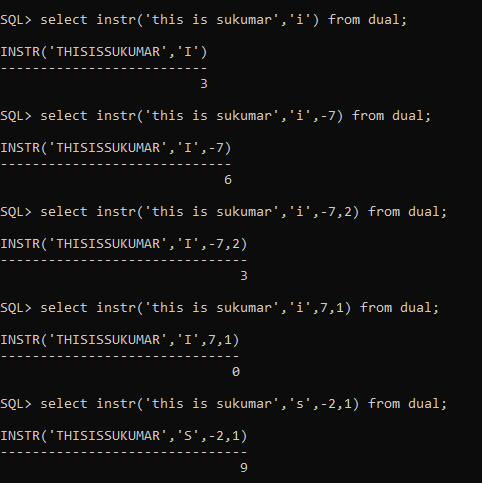
9.INSTRING:-



This function returns position of character, if it is found; otherwise it returns 0. Oracle Database begins the search—that is, the ‘m’ of the first character of the first substring to compare with *substring*. If *‘m’* is negative, then Oracle counts backward from the end of *string* and then searches backward from the resulting position.

* n should be positive. It represents occurrence of second argument. it should be >0.Default value is 1.
* 

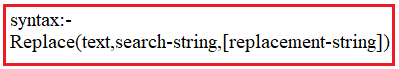
EXAMPLES:-



10.REVERSE(colnam/string)

it returns string in reverse.

11.REPLACE:-



* It returns string in which all occurrence of search-string have been replaced with replacement -string.
* Replacement-string is optional. If replacement-string is omitted or null, then search string will be removed from text.
* There is no need to equal no.of characters in search-string and replacement-string.
* This function is string replacement function.It is not character by character replacemnet.

EXAMPLES:-

SQL> select replace('this is sukumar','sukumar') from dual;

REPLACE(

--------

this is

SQL> select replace('this is sukumar','sukumar','veena') from dual;

REPLACE('THIS

-------------

this is veena

SQL> select replace('this is sukumar','suku','veena') from dual;

REPLACE('THISISS

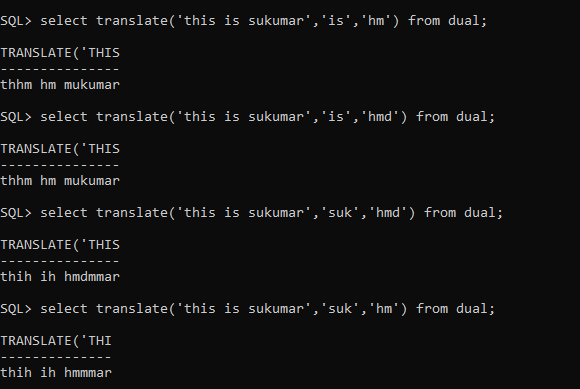
----------------

this is veenamar

12.TRANSALATE:-



* The no.of characters in string1 must equal to no.of characters in string2.
* if no.of characters in search-string is < no.of chatacters in replacement-string then extra character will be ignored.
* if no.of characters in search-string is > no.of characters in replacement-string ,then string replacement is done and character by character replacement is also done.



**5.2.2)NUMBER FUNCTIONS**: The numeric functions always manipulate/processes only numeric values and return numeric value as result.

1. abs:-This function returns absolute value. This function doesnot consider sign of input value. it always return positive value.

SQL> select abs(10) from dual;

ABS(10)

----------

10

SQL> select abs(-10) from dual;

ABS(-10)

----------

10

SQL> select abs(10.4) from dual;

ABS(10.4)

----------

10.4

SQL> select abs(-10.4) from dual;

ABS(-10.4)

----------

10.4

2.CEIL:- The CEIL() function returns the nearest greatest integer.

SQL> select ceil(10) from dual;

CEIL(10)

----------

10

SQL> select ceil(10.2) from dual;

CEIL(10.2)

----------

11

SQL> select ceil(10.6) from dual;

CEIL(10.6)

----------

11

SQL> select ceil(-10.2) from dual;

CEIL(-10.2)

-----------

-10

SQL> select ceil(-10.7) from dual;

CEIL(-10.7)

-----------

-10

3.FLOOR:- syntax:-floor(number)

The FLOOR() function returns the nearest lowest integer value SQL> select floor(3.4) from dual;

FLOOR(3.4)

----------

3

SQL> select floor(3.7) from dual;

FLOOR(3.7)

----------

3

SQL> select floor(-3.4) from dual;

FLOOR(-3.4)

-----------

-4

SQL> select floor(-3.7) from dual;

FLOOR(-3.7)

-----------

-4

4.power(base,powervalue).

5.sqrt(v1) .v1 should be positive value.

6.sin

7.con

8.Acos

9.Asin

10.Atan

11.cosH

12.tan

**5.2.3)AGGREGATEFUNCTIONS/GROUP FUNCTIONS/MULTIVALUED FUNCTIONS**:-

1.min(colname/group of values)

2.max(colname/group of values)

3.sum(colname/group of values)

4.avg(colname/group of values)

5.count(colname/group of values)

EXAMPLES:

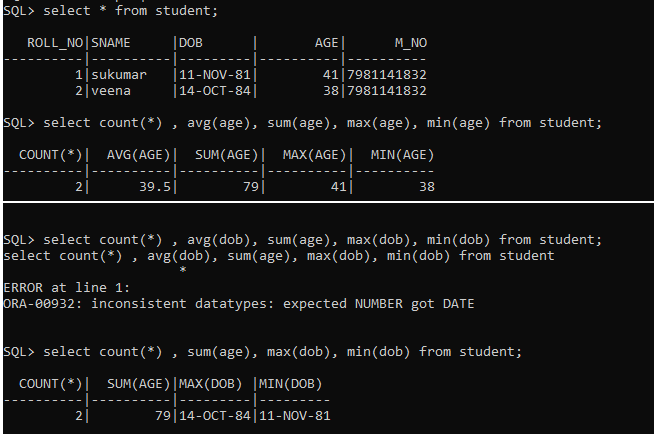
SQL> select count(empno) ,job from emp;

select count(empno) ,job from emp

\*

ERROR at line 1:

ORA-00937: not a single-group group function

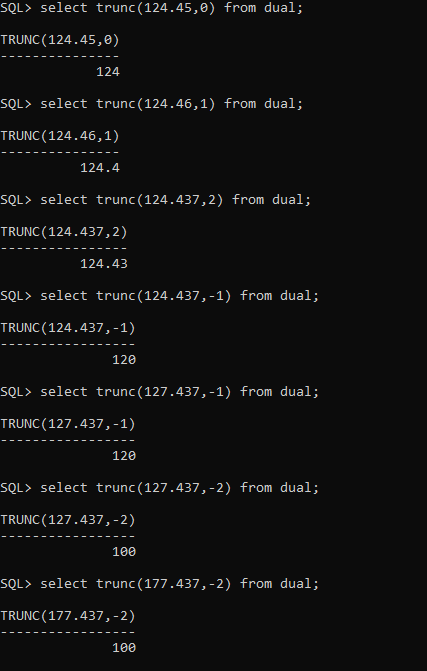


**5.2.4).OTHER FUNCTIONS**:- The functions are

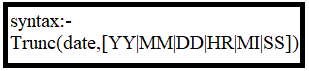
1. Truncate:-

syntax: Trun(m,n)

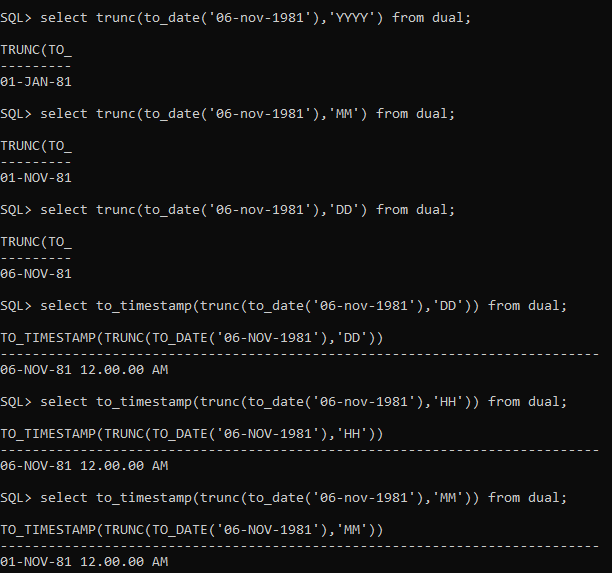
* + n is optional.
  + n is 0,positive or negative number. Default value is 0.
  + If n is 0 then decimal part is completely discarded and only integer part will be returned.
  + if n is -value, then this function avoids decimal part and truncates unitplace, tensplace,....in integer part based on n value.
  + if n is +value, then this function keeps n digits in decimal part and discards remaining digits.

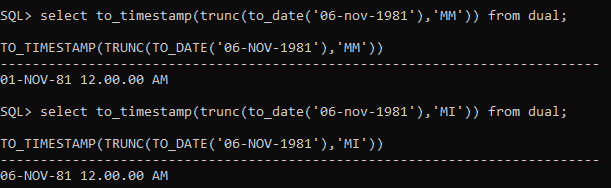


Note:- The truncate function can also be used to truncate the date & interval.



* The first argument should be date data type
* when date is truncated based on year, we get starting date of year.
* when date is truncated based on month, we get starting date of month.
* when date is truncated baed on day, we get starting time of day.
* when date is truncated based on hour, in time, minutes & seconds becomes zeros.
* when date is truncated based on minutes,In time, only seconds becomes zero.





2. ROUND:-



This function rounds integer part or decimal part of m based on n.

* n is optional.
* n is 0,positive or negative number. Default value is 0.
* If n is 0 then m is rounded to nearest integer and it will be returned.
* if n is -value, then this function avoids decimal part and round unit place, tens place,....in integer part based on n value.
* if n is +value, then this function keeps n digits in decimal part and round decimal part only if necessary.

EXAMPLES:-

SQL> select round(10.1) from dual;

ROUND(10.1)

-----------

10

SQL> select round(10.7) from dual;

ROUND(10.7)

-----------

11

SQL> select round(10.777,2) from dual;

ROUND(10.777,2)

---------------

10.78

SQL> select round(10.773,2) from dual;

ROUND(10.773,2)

---------------

10.77

SQL> select round(127.773,-1) from dual;

ROUND(127.773,-1)

-----------------

130

SQL> select round(127.773,-2) from dual;

ROUND(127.773,-2)

-----------------

100

Note:- Round function can also be used to round the date and interval.



The first argument should be date datatype.

-> when date is rounded based on year, the month is checked .If month is less than or equal to 6. we get startng date of current year.other wise we get starting date of next year.

SQL> select round(to\_date('11-nov-1991'),'yy') from dual;

ROUND(TO\_DATE('11

-----------------

01-01-92 12:00:00

SQL> select round(to\_date('11-apr-1991'),'yy') from dual;

ROUND(TO\_DATE('11

-----------------

01-01-91 12:00:00

-> when date is rounded based on month,the day is checked.If day is less than or equal to 15 . we get satring date of month.otherwise we get starting date of next month.

SQL> select round(to\_date('11-apr-1991'),'mm') from dual;

ROUND(TO\_DATE('11

-----------------

01-04-91 12:00:00

SQL> select round(to\_date('16-apr-1991'),'mm') from dual;

ROUND(TO\_DATE('16

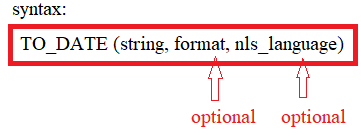
-----------------

01-05-91 12:00:00

-> when date is rounded based on day,the Hours is checked.If hour is less than or equal to 6|12 . we get date of that day.otherwise we get starting date of next day.

**5.2.5. Conversion Function:**

**1.TO\_DATE**:-



Arguments

The TO\_DATE() function accepts three arguments:

1) string

is a string value which is converted to a DATE value. It can be a value of any data type CHAR, VARCHAR2, NCHAR, or NVARCHAR2.

2) format is the date and time format for the string.

The format argument is optional. If you omit the format, the string must be in the standard date format which is DD-MON-YY e.g., 31-DEC-2000

Noted that if format is J, which is for Julian, then the string must be an integer.

For the detailed information on how to construct the format, check it out the Oracle date format.

3) nls\_language

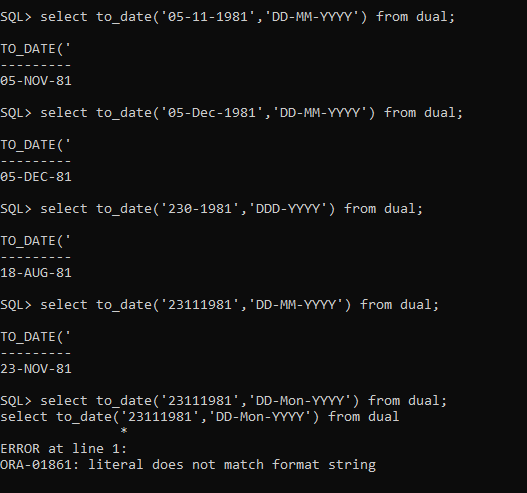
is an expression that specifies the language for day and month names in the string. This nls\_language argument has the following form:

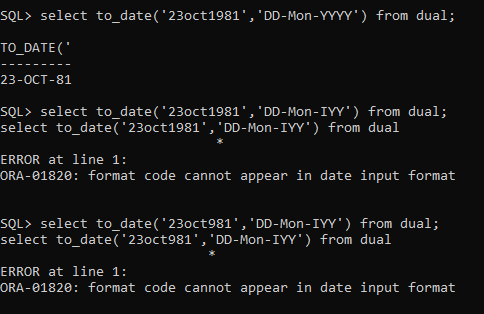
NLS\_DATE\_LANGUAGE = language

This ls\_language argument is optional. If you omit it, the TO\_DATE() function will use the default language for your session.

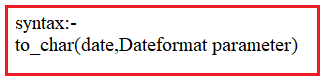
Return value

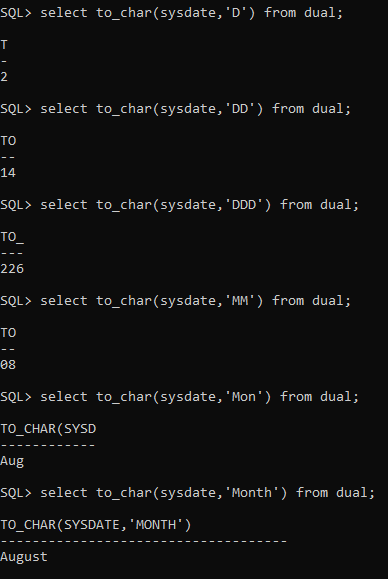
The TO\_DATE() function returns a DATE value which is corresponding to the input string.



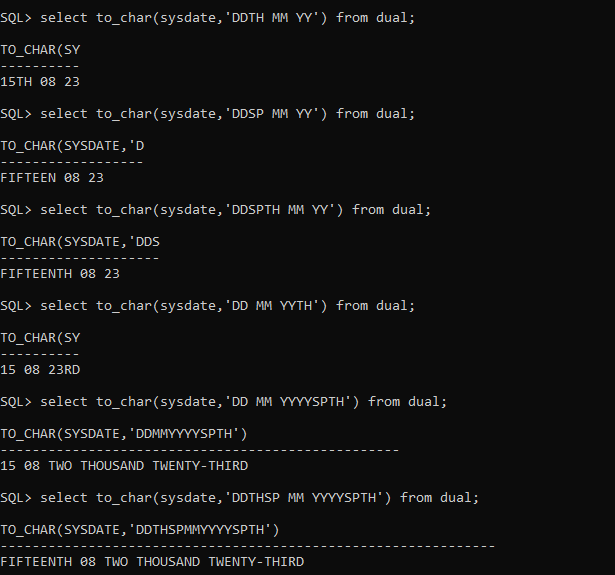


4.TO\_CHAR:- This function converts a DATE or INTERVAL value to a string in a specified date format. It returns date whose type is string.





===🡺Important





**5.2.6) DATE FUNCTIONS:-** The date functions always takes date type value as input. The date functions returns values. The value type is

DATE,integer or character.

1. sysdate:- It returns date base on dbserver zone.

2. current\_date:- it returns date based on client system zone.

note:- The db server is in us and client system is in india. The sysdate give the usdate and current\_date gives indiadate.If both are in same zone,sysdate and current\_date give same date.

Example:-

SQL> select sysdate from dual;

SYSDATE

---------

25-FEB-20

SQL> select current\_date from dual;

CURRENT\_D

---------

25-FEB-20

3.ADD months:-

syntax:- add\_months(date,+/-num);

This functions adds or subtract no.of months to given date.

Example:-

SQL> select add\_months(sysdate,2) from dual;

ADD\_MONTH

---------

25-APR-20

SQL> select add\_months(sysdate,-1) from dual;

ADD\_MONTH

---------

25-JAN-20

4.Months\_between:-

syntax:- months\_between(date1,date2)

-> if both months are same,then it returns zero.

-> This function subtracts date2 from date2 and gives result. The result is interms of months. The result is 0,positive or negative.

Examples:-

SQL> select months\_between(sysdate,sysdate) from dual;

MONTHS\_BETWEEN(SYSDATE,SYSDATE)

-------------------------------

0

SQL> select months\_between(sysdate,'25-nov-2020') from dual;

MONTHS\_BETWEEN(SYSDATE,'25-NOV-2020')

-------------------------------------

-9

SQL> select months\_between(sysdate,'25-jan-2020') from dual;

MONTHS\_BETWEEN(SYSDATE,'25-JAN-2020')

-------------------------------------

1

5.LAST\_DAY:- This function returns last date of give month based on date.

syntax:-last\_day(date)

SQL> select last\_day(sysdate) from dual;

LAST\_DAY(

---------

29-FEB-20

SQL> select last\_day('01-jan-2020') from dual;

LAST\_DAY(

---------

31-JAN-20

6.NEXT\_DAY:-

Syntax:- next\_day(date,'weekday');

it returns date. The next weekday will be on that returned date.

Example:-

SQL> select next\_day('01-jan-2020','mon') from dual;

NEXT\_DAY(

---------

06-JAN-20

SQL> select next\_day('01-jan-2020','wed') from dual;

NEXT\_DAY(

---------

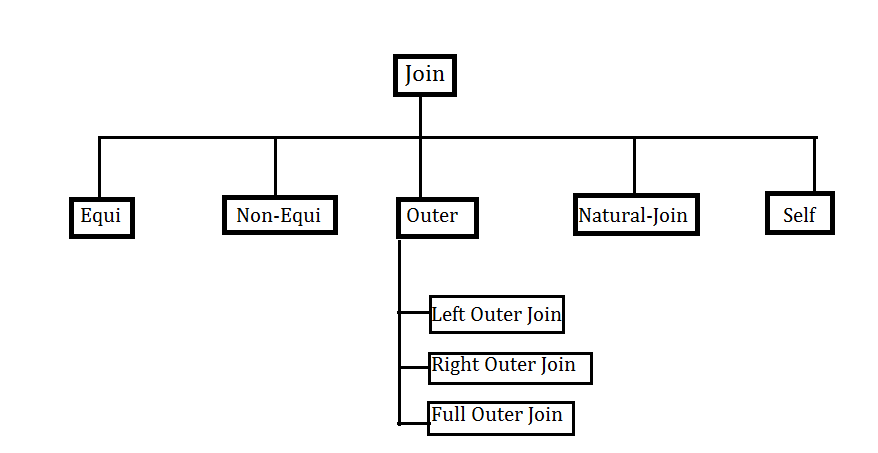
08-JAN-20

**6)JOINS**: The join is denoted by *⊲⊳ symbol.* The join is used to retrieve data from two relation instances. The join can be defined as cross product followed by Selection and projection.

Note:- The result of cross product is always larger than result of join.

There are 5 joins.

1. Equi join/Inner Join
2. Non-Equi join
3. Natural join
4. Self Join.
5. Outer join



1. Non-Equi join:-

Syntax: Relation1 *⊲⊳*c *Relation2*

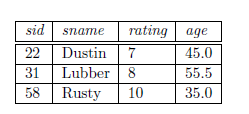
The join operator accept two relations and condition. It will gives one relation instance as Result.Where c is condition. The condition is made with operator and attributes of Relation1 and relation2. The attribute can be referred as **Relation-name. Attribute | position number**. The operator may be >,<,<=,>=,!=, between, in …etc.

It is equalient to *Relation1 ⊲⊳c Relation2* =*σc*(*Relation1 × Relation2*)

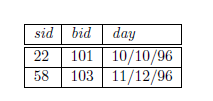
The result instance of condition join contains attributes of Relation1 followed by attributes of Relation2.

Example:-

Relation-Name: Sailors



Relation-Name: Reserves



Relation AlgebraQuery : Sailors *⊲⊳*sailors.sid<Reserves.sidReserves.

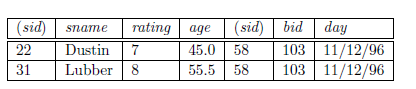
SQL Queries:

a)selects.sid,s.sname,s.rating,s.age,r.sid,r.bid,r.day from

sailors s,reserves r where s.sid<r.bid;

Result:

----------



2.Equi join:-

Syntax: Relation1 *⊲⊳*c *Relation2*

The join operator accept two relations and condition. It will gives one relation instance as Result.Where c is condition. The condition contains only equlityoperator(=) and attributes of Relation1 and relation2. The attribute can be referred as **Relation-name. Attribute | position number**.

It is equalient to *Relation1 ⊲⊳c Relation2* =*σc*(*Relation1 × Relation2*)

The result instance of condition join contains attributes of Relation1 followed by attributes of Relation2 that don’t appear in condition.

Example:-

Query : Sailors *⊲⊳*sailors.sid=Reserves.sidReserves

SQL Queries:

a)selects.sid, s.sname, s.rating, s.age, r.bid, r.day from

sailors s, reserves r where s.sid=r.sid;

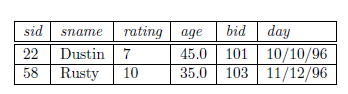
(or)

Select \* from sailors join reserves on sailors.sid=reserves.sid;

(or)

Select \* from sailors join reserves using (sid);

Result:



3.Natural Join:-

Syntax: Relation1 *⊲⊳ Relation2*

The join operator accepts only two relations. It will gives one relation instance as Result. In this case, we can simply omit the join condition; the default is that the join condition is a collection of equalities on all common fields.

Example:-

R(a:number,b:string,c:string)

S(a:number,c:string,d:number,e:number)

Ex:1Relational Algebra Query: R *⊲⊳ S*

SQL Query: select \* from R natural join S;

*The above query is equal to R ⊲⊳*R.a=S.a AND R.c*=S.c S.*

Schema of Result instance:R *⊲⊳S(a:number,b:string,c:string,d:number,e:number)*

Note:- In Oracle, we are not allowed to use ALIAS name on joining conditional column.

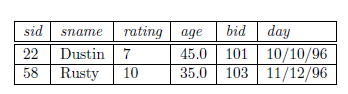
Ex:2

Relational Algebra query Sailors *⊲⊳ Reserves*

Sql query: select \* from sailors natural join reserves;

Output:

---------



4.outer join:-Outer Joins subdivide further into,

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

A)LEFT Outer Join:The left outer join returns a result set table with the **matched data** from the two tables and then the remaining rows of the **left** table and null from the **right** table's columns.

Syntax: R1  R2

SQL Query Syntax:

Select column-name-list from

Table-name1 left outer join table-name 2

On table-name 1. Common-col-name=table-name2.common-col- name;

Ex:

Relational Algebra Query: sailors  reserves

Sql Query: select s.sid , s.name , s.rating , s.age,

r.sid,r.bid,r.day

sailors s left outer join reserves r

on s.sid=r.sid;

Relation Instance:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sid | Sname | Rating | Age | Sid | Bid | day |
| 22 | Dustin | 7 | 45 | 22 | 101 | 10-oct-96 |
| 58 | Rusty | 10 | 35 | 58 | 103 | 11-dec-96 |
| 31 | Lubber | 8 | 55 | Null | Null | Null |

B)Right outer join**:-**The right outer join returns a resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.



Syntax: R1  R2

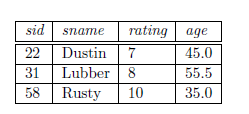
SQL Query Syntax:

Select column-name-list from

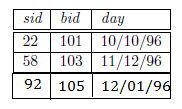
Table-name1 right outer join table-name 2

On table-name 1.Common-col-name=table-name2.common-col- name;

Relation-Name: Sailors



Relation-Name: Reserves



Ex:

Relation algebra query: sailors  reserves

SQL Query: select s.sid , s.name , s.rating , s.age

r.sid,r.bid,r.day

sailors s right outer join reserves r

on s.sid=r.sid;

Relation Instance of result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sid | Sname | Rating | Age | Sid | Bid | day |
| 22 | Dustin | 7 | 45 | 22 | 101 | 10-10-96 |
| 58 | Rusty | 10 | 35 | 58 | 103 | 11-12-96 |
| Null | Null | Null | Null | 92 | 105 | 12-01-96 |

### C)Full Outer Join:The full outer join returns a resultset table with the matched data of two table then remaining rows of both left table and then the right table.



Syntax: R1  R2

SQL Query Syntax:

Select column-name-list from

Table-name1 full outer join table-name 2

On table-name 1. Common-col-name=table-name2.common-col- name;

Ex:

Relational Algebra Query: Sailors  reserves

Sql Query:select s.sid , s.name , s.rating , s.age

r.sid,r.bid,r.day

sailors s full outer join reserves r

on s.sid=r.sid;

Relation Instance of Result:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sid | Sname | Rating | Age | Sid | Bid | day |
| 22 | Dustin | 7 | 45 | 22 | 101 | 10-10-96 |
| 58 | Rusty | 10 | 35 | 58 | 103 | 11-12-96 |
| 31 | Lubber | 8 | 55 | Null | Null | Null |
| Null | Null | Null | Null | 92 | 105 | 12-01-96 |

**7.SUB QUERIES/NESTED QUERY/INNER QUERY**:- A nested query is query that has another query embedded within it. The embedded query is called sub query.

* The sub query may has another sub query embedded within it.
* The sub query appears in where clause, from-clause or having clause.
* The Inner query is completely independent of outer query.
* The Inner query output becomes an input to outer query.

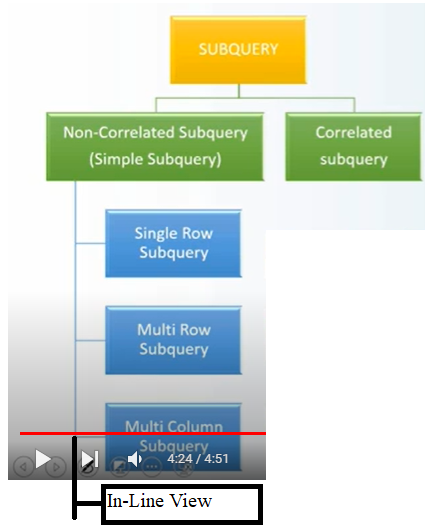
There are few rules that sub queries must follow:-

1. sub queries must be enclosed with in parentheses.
2. ORDER BY clause cannot be used in sub query.
3. Subqueries that return more than one row can only be used with multiple value operators such as the IN operator.

All relational databases having two types of sub query. These are

1.Non-Corelated Sub Query.

2.Corelated sub Query.



3.1)Non-Corelated Sub Query:- In Non-corelated sub query, “child query/Inner Query” is executed first then only “parent query” is executed.

a) single row subquery.

b) Multiple row subquery.

c) Multiple column subquery.

d) Inline-View.

3.1.a)Single row sub query: The child query always returns single value. In single row sub query we are using =, != ,> ,>=, >=,< ,<=, between operators.

Ex:

1.Write a query to display the employees who are getting more salary than average salary from emp table.

SQL> select ename,sal from emp where sal>(select avg(sal) from emp);

2.Wrie a query to display the senior most employee details from emp table.

SQL> select \* from emp where exp=(select max(exp) from emp);

3.1.b)Multiple row Sub query: The child query returns multiple values. In multiple row sub query, we are using “in, not in, all, any, Exist, not Exist operators.

EX:

1.write a query to display the employees who are working in “sales or research” department from emp , dept table.

SQL>select ename, deptno from emp

Where deptnoin(select deptno from dept where dname=’sales’ or

Dname=’research’);

2.Write a query to display employees who are working as “manager” from emp table.

SQL> select \* from emp where eno in (select mgr from emp);

3.1.c)Multiple Column subquery: In all relational databases we can also compare multiple column values of the child query with multiple column of parent query. This type of sub query is also called as “Multiple column subquery”.

Syntax: select \* from tablename where(col1,col2,…)

In

(select col1,col2,… from tablename where condition);

Ex:

1.Write a query to display employees who are getting maximum salary in each department from emp table by using multiple row sub query.

SQL> select deptno,sal,ename

From emp

Where (deptno, sal) in (select deptno,max(sal) from emp group by deptno);

3.1.d) InLineView:- Generally , in oracle we are not allowed to use “alias name” in where clause. In oracle, if you want to use column ALIAS NAME is “where “ clause then we must specified ALIAS NAME query in place of tbale name within parent query. This type of subquery is called “Inline view”.

Ex:

1.Write a query to display employee who are getting more than 30000 annual sal from emp table.

SQL> select \* from( selectename,sal,sal\*12 as annual\_sal from emp)

Where annual\_sal>30000;

3.2)Co-related Nested queries:- In co-related sub query “parent query “is executed first then only “child query” is executed.The parent query and inner query are not independent queries.(i.e) The inner query refers the column[s] of the outer query.

**Note:- For each row of outer query, the Inner query is executed once.**

Example:

1. Find names of sailors who have reserved boat 103.

SELECT S.sname

FROM Sailors S

WHERE EXISTS ( SELECT\*

FROM Reserves R

WHERE R.bid = 103

AND R.sid = S.sid );

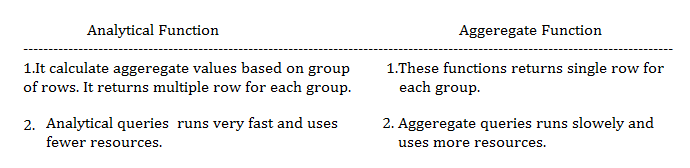
The sub query that used s.sid from outer query is called co-related sub query.

**8. Analytical Functions**:- \* The Analytical functions were introduced since oracle8i.

\* Analytical functions are also called OLAP functions.

\* Oracle analytic functions calculate an aggregate value based on a group of rows and return multiple rows for each group.

Diff between analytical functions and aggregate functions.



Syntax:-

There are some variations in the syntax of the individual analytic functions, but the basic syntax for an analytic function is as follows.

analytic\_function name([ arguments ]) OVER (analytic\_clause)

The analytic\_clause breaks down into the following optional elements.

[ query\_partition\_clause ] [ order\_by\_clause [ windowing\_clause ] ]

1.)Query Partition: clause:- The query\_partition\_clause divides the result set into partitions, or groups, of data. The operation of the analytic function is restricted to the boundary imposed by these partitions, similar to the way a GROUP BY clause affects the action of an aggregate function.

If the query\_partition\_clause is omitted, the whole result set is treated as a single partition.

Example:-

SQL> select empno,deptno,sal,avg(sal) over () as suku from emp;

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7369 20 800 2132.69231

7499 30 1600 2132.69231

7521 30 1250 2132.69231

7566 20 2975 2132.69231

7654 30 1250 2132.69231

7698 30 2850 2132.69231

7782 10 2450 2132.69231

7788 20 3000 2132.69231

7839 10 5000 2132.69231

7844 30 1500 2132.69231

7876 20 1100 2132.69231

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7900 30 950 2132.69231

7902 20 3000 2132.69231

13 rows selected.

SQL> select empno,deptno,sal,avg(sal) over (partition by deptno) as suku from emp;

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7782 10 2450 3725

7839 10 5000 3725

7788 20 3000 2175

7876 20 1100 2175

7369 20 800 2175

7902 20 3000 2175

7566 20 2975 2175

7499 30 1600 1566.66667

7844 30 1500 1566.66667

7698 30 2850 1566.66667

7654 30 1250 1566.66667

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7900 30 950 1566.66667

7521 30 1250 1566.66667

13 rows selected.

2.)Order by clause:-The order\_by\_clause is used to order rows, or siblings, within a partition.

SQL> select empno,deptno,sal,avg(sal) over (partition by deptno) as suku from emp;

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7782 10 2450 3725

7839 10 5000 3725

7788 20 3000 2175

7876 20 1100 2175

7369 20 800 2175

7902 20 3000 2175

7566 20 2975 2175

7499 30 1600 1566.66667

7844 30 1500 1566.66667

7698 30 2850 1566.66667

7654 30 1250 1566.66667

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7900 30 950 1566.66667

7521 30 1250 1566.66667

13 rows selected.

SQL> select empno,deptno,sal,avg(sal) over (partition by deptno order by sal desc) as suku from emp;

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7839 10 5000 5000

7782 10 2450 3725

7788 20 3000 3000

7902 20 3000 3000

7566 20 2975 2991.66667

7876 20 1100 2518.75

7369 20 800 2175

7698 30 2850 2850

7499 30 1600 2225

7844 30 1500 1983.33333

7521 30 1250 1690

EMPNO DEPTNO SAL SUKU

---------- ---------- ---------- ----------

7654 30 1250 1690

7900 30 950 1566.66667

13 rows selected.

3) Windowing\_clause:- The windowing\_clause is an extension of the order\_by\_clause and as such, it can only be used if an order\_by\_clause is present.

The windowing\_clause has two basic forms.

RANGE BETWEEN start\_point AND end\_point.

ROWS BETWEEN start\_point AND end\_point.

Possible values for "start\_point" and "end\_point" are:

UNBOUNDED PRECEDING : The window starts at the first row of the partition, or the whole result set if no partitioning clause is used. Only available for start points.

UNBOUNDED FOLLOWING : The window ends at the last row of the partition, or the whole result set if no partitioning clause is used. Only available for end points.

CURRENT ROW : The window starts or ends at the current row. Can be used as start or end point.

Note:-The default windowing\_clause is RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW, not ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW.

The Analytical functions are

.Rank

.percent\_rank

.row\_number

.First

.First\_value

.last

.last\_value

.lag

.lead

.dense\_rank

.Ntile

.Median

.stddev

.feature\_details

.cluster\_id

.Avg(\*)

.sum(\*)

.Min

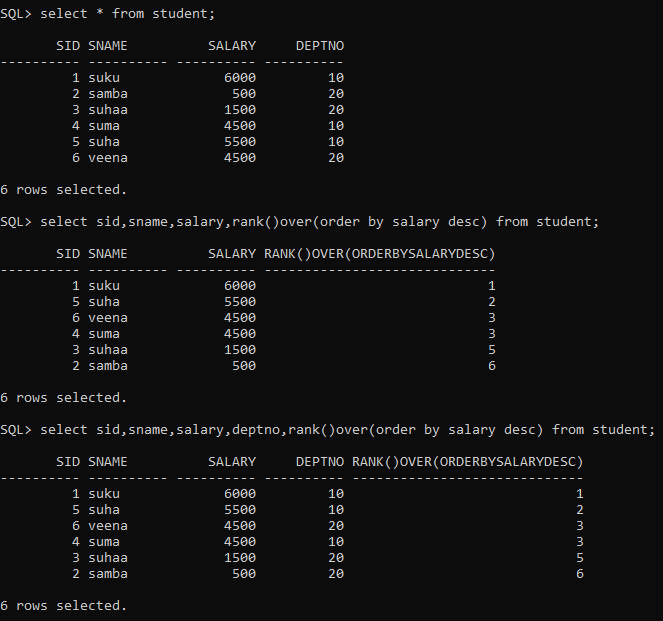
.Max

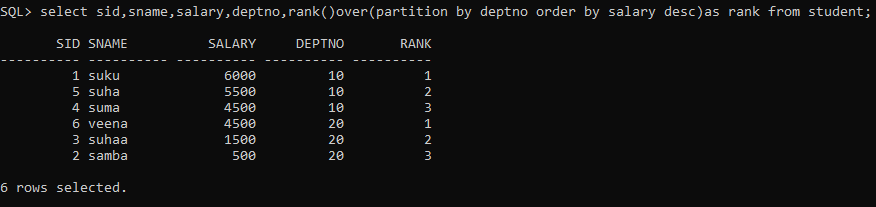
.Count(\*)

**1.Rank**:- This function calculates rank of rows with in group.

syntax:1 rank()over(order by clause);

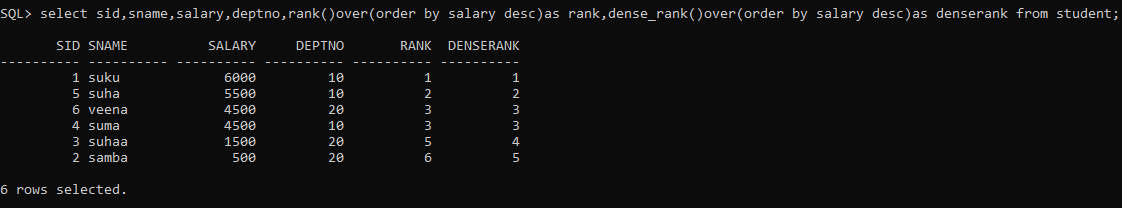
syntax:2 rank()over([partion clause,]order by clause[,window clause])

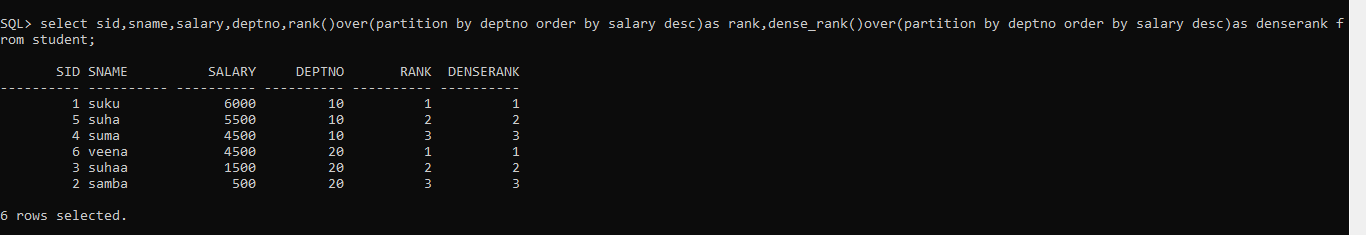




**2) Dense\_rank():-** It is similar to rank() function.

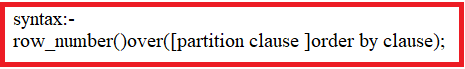
diff b/w dense\_rank() and rank()

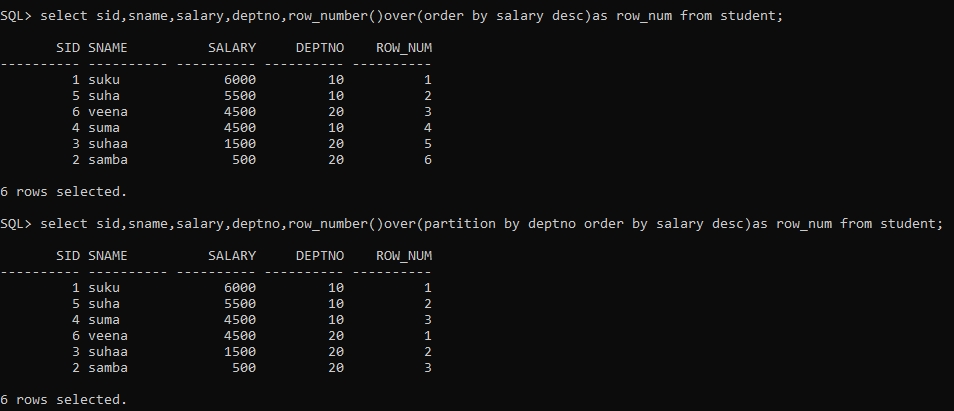




if two values are same,then gap in rank sequence.To avoid such problem we use dense\_rank.

**4.row\_number():-**The ROW\_NUMBER() is an analytic function that assigns a sequential unique integer to each row to which it is applied, either each row in the partition or each row in the result set.





**4)Percent\_rank**:- It returns the position or rank of records in group of records.

The first row always get rank 0.

The percent\_rank is calculated as: (position of row-1/total rows-1)

Example:-

SQL> select empno,ename,deptno,sal,percent\_rank()over(order by sal) from emp;

EMPNO ENAME DEPTNO SAL PERCENT\_RANK()OVER(ORDERBYSAL)

---------- ---------- ---------- ---------- ------------------------------

7369 SMITH 20 800 0

7900 JAMES 30 950 .083333333

7876 ADAMS 20 1100 .166666667

7521 WARD 30 1250 .25

7654 MARTIN 30 1250 .25

7844 TURNER 30 1500 .416666667

7499 ALLEN 30 1600 .5

7782 CLARK 10 2450 .583333333

7698 BLAKE 30 2850 .666666667

7566 JONES 20 2975 .75

7902 FORD 20 3000 .833333333

7788 SCOTT 20 3000 .833333333

7839 KING 10 5000 1

13 rows selected.

SQL> select empno,ename,deptno,sal,percent\_rank()over(partition by deptno order by sal) from emp;

EMPNO ENAME DEPTNO SAL PERCENT\_RANK()OVER(PARTITIONBYDEPTNOORDERBYSAL)

---------- ---------- ---------- ---------- -----------------------------------------------

7782 CLARK 10 2450 0

7839 KING 10 5000 1

7369 SMITH 20 800 0

7876 ADAMS 20 1100 .25

7566 JONES 20 2975 .5

7902 FORD 20 3000 .75

7788 SCOTT 20 3000 .75

7900 JAMES 30 950 0

7521 WARD 30 1250 .2

7654 MARTIN 30 1250 .2

7844 TURNER 30 1500 .6

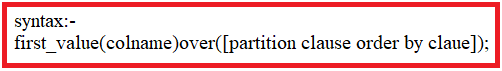
7499 ALLEN 30 1600 .8

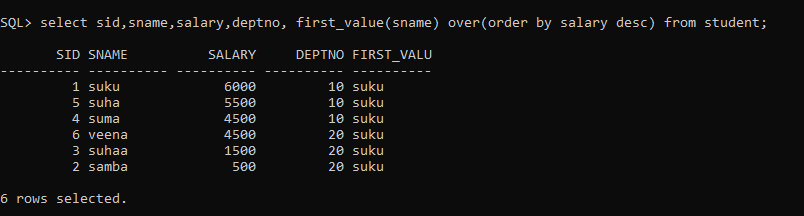
7698 BLAKE 30 2850 1

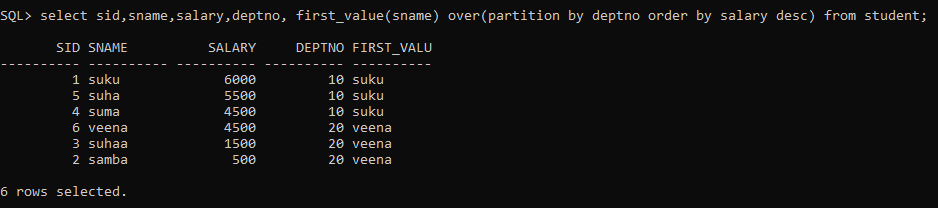
13 rows selected.

**5.first\_value():**-The Oracle PL/SQL FIRST\_VALUE function is an analytic function which selects the first record from the partitioned and ordered set of rows.

You can specify the IGNORE NULLS parameter to ignore the NULL values of the operand column and force consideration of NOT NULL values.

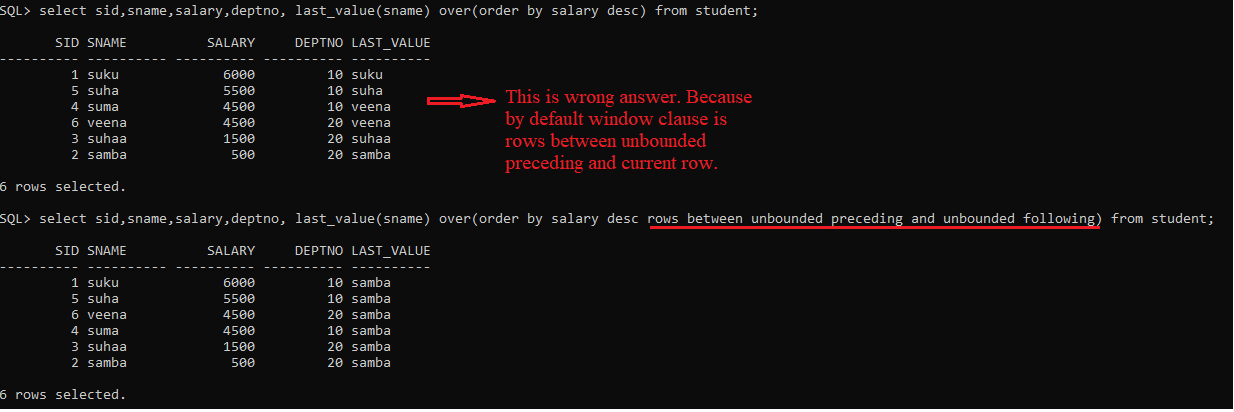




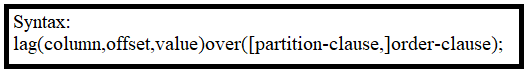


6.last\_value():



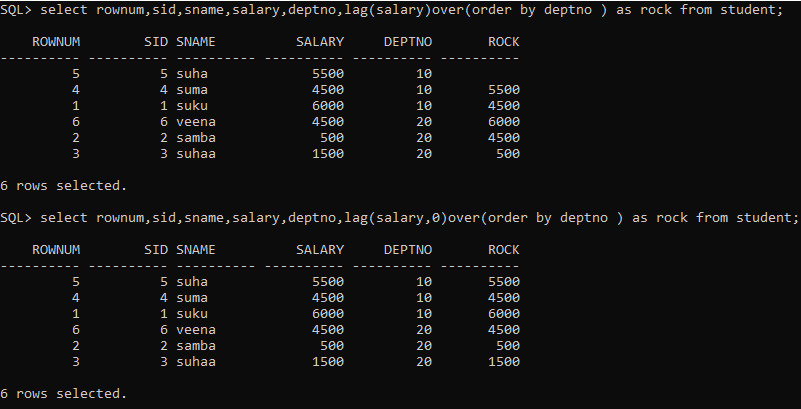


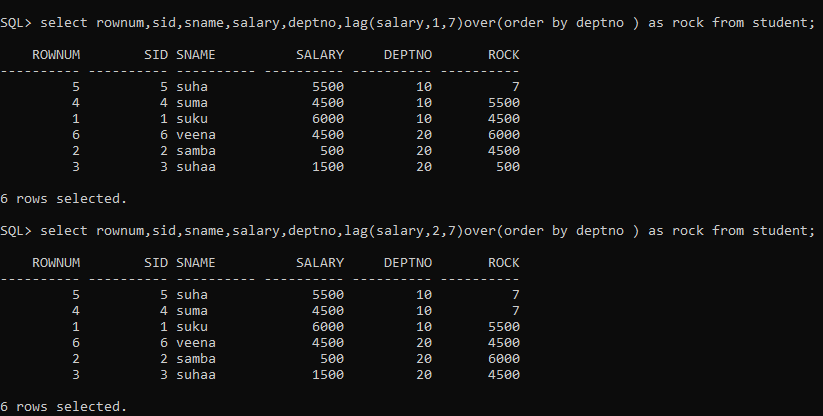
7.Lag():- It is used to access the data from previous specific row and place in current row.



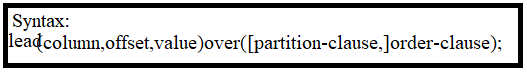
offset:- it is the physical offset from the current row, default value is 1.

Value:-Optional. It is the value that is returned if the offset goes out of the bounds of the table. If this parameter is omitted, the default is null.



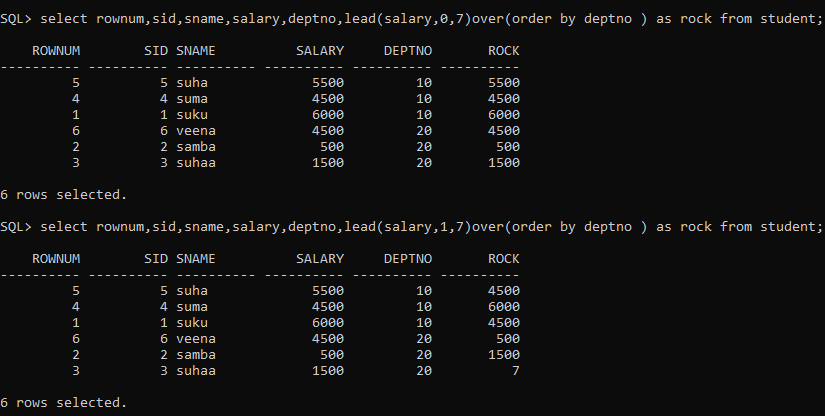


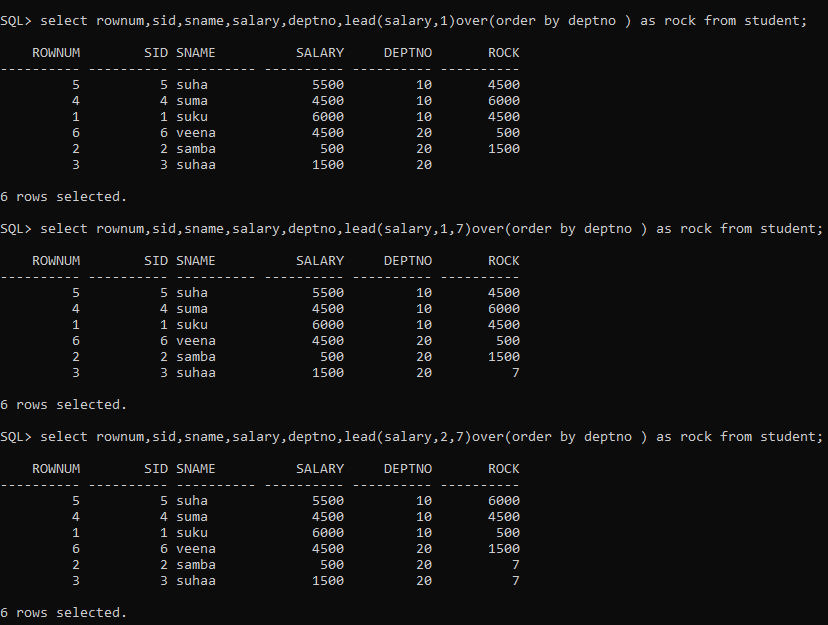
8.Lead():- It is used to access the data from subsequent specific row and place in current row.



offset:- it is the physical offset from the current row, default value is 1.

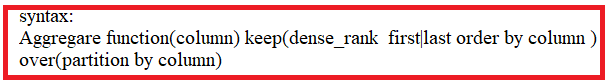
Value:-Optional. It is the value that is returned if the offset goes out of the bounds of the table. If this parameter is omitted, the default is null.

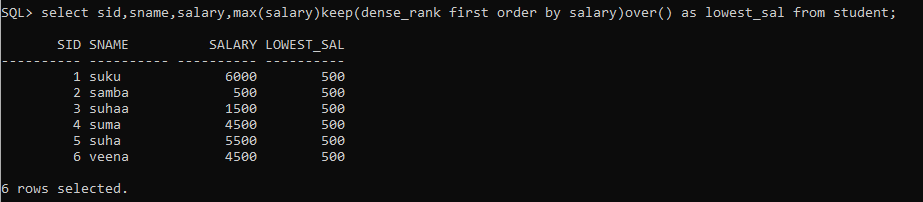


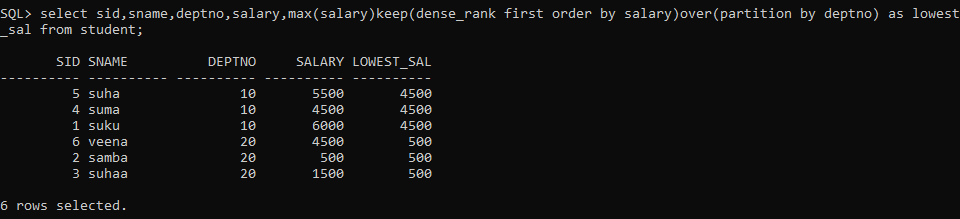


10.First():- The First analytic function can be used to take the first value from an ordered sequence of partition and to place in specific column of current row.

11.last():- The last analytic function can be used to take the last value from an ordered sequence of partition and to place in specific column of current row.







10.Synonym:- THE SYNONYM is database object. The synonym is alias or another name of object. The object may be table,view sequence,function or any other db object.

\* The synonym can only be created for entire DB object.(i.e) we can create synonym for entire table. but we can't create synonym for specific rows or set of colunmns with in table.

\* The synonym does not store any data. The synonym can be either public synonym or private synonym.

\* The user synonyms information is stored in USER\_SYNONYMS view.

\* **we can't add and drop column to base object through synonym.**

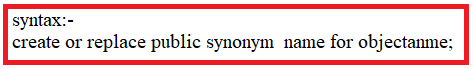
**\* We can do DML operation on base object through synonym.**

**10.1)PRIVATE SYNONYM**:-

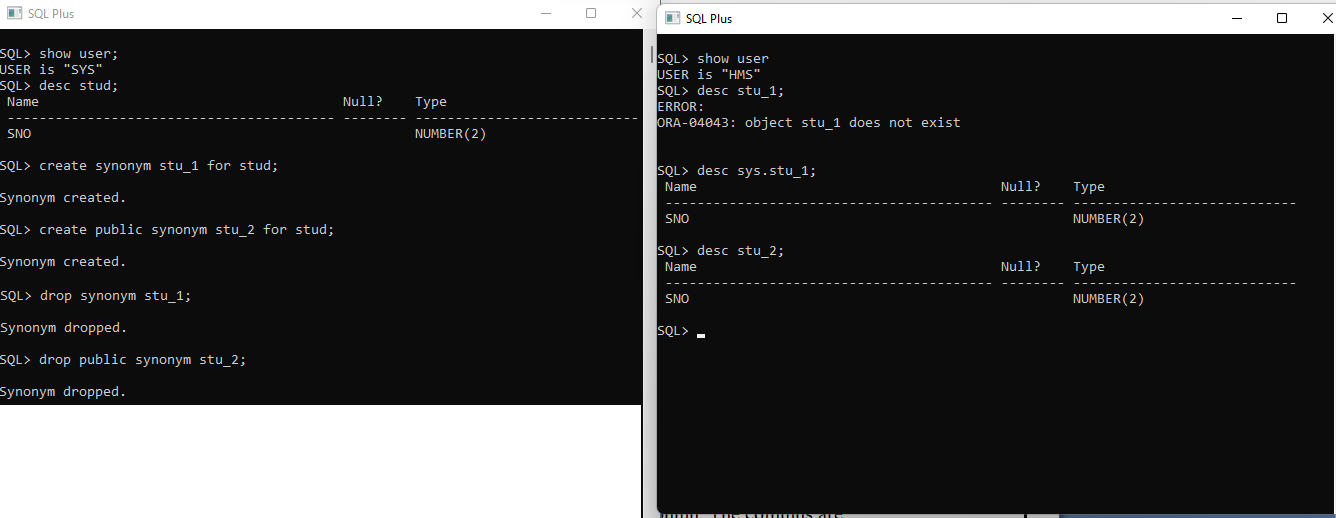


Note:-The private synonym can only be accessed with in current user schema.

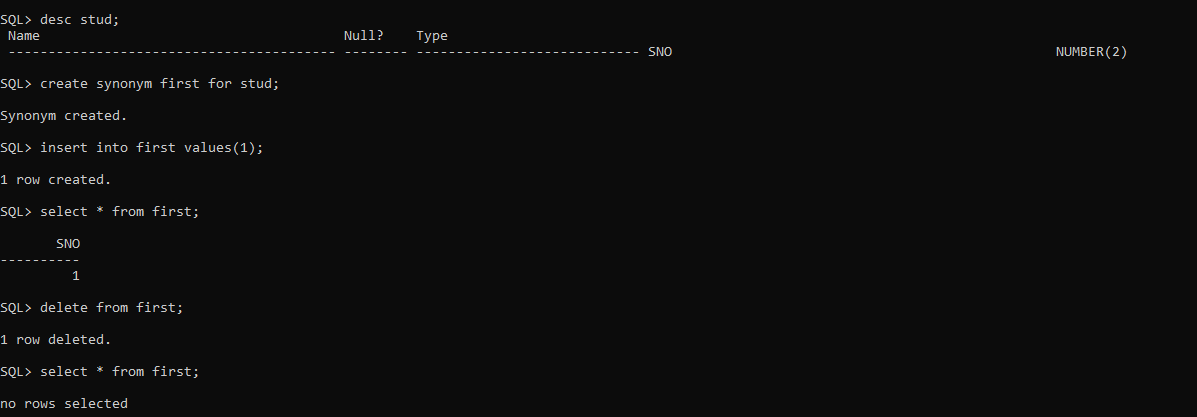
**10.2)PUBLIC SYNONYM**:- PUBLIC synonym is available in all schemas of sameDB. To create public synonym, the user must be DBA,the user should has DBA priviliges or user shold has privileges of create public synonym.



Note:- The public synonym can be accessed with in current user schema and another user schema.



The DML Operations can be done using synonym.



**11)SEQUENCE**:- The sequence is used to generat unique sequential values. Sequence has two pseudocoloumn. The columns are

1) CURRVAL

2)NEXTVAL

**1.currval**:- The currval has value which is most recently generated by sequence. we created sequence, but sequence did not generate any value. when if we try to get currval, then we get error message.

**2.nextval**:- It gives the next available value in sequence to generate.



* Incremented by:- The default value is 1.It can be positive number(or) negative number. It should not be zero.

If increment by value is positive, then the sequence is called incremental sequence and dbms adds this value to min\_value.

if increment by value is negative,the sequence is called decrement sequence and dbms add its value to max\_value.

* startwith: Default value is MINValue. This value can be altered. sequence should starts from this number for first cycle. Rest of cycle starts from min\_value.
* Minvalue:- Default value is 1.

if sequence is incremental sequence, then min\_value indicates starting number of sequence.

if sequence is decremental sequence, the min\_value indicates ending number of sequence.

* MAXVALUE:- default value is big number(99999..9).

if sequence is incremental sequence, then max\_value becomes ending number of sequence.

if sequence is decremental sequence, then max\_value becomes starting number of sequence.

* CACHE:- Deault value is 20. Instead of generating single value, sequence generates specified no.of values. These values are stored in buffer.

Cache value should be greather than 0.

* **cycle:-** Default value is nocycle.

case1: when sequence reaches the max\_value, it examines cycle properties value. If cycle property value isno cycle,it stops generating next value & display message to user;other wise it starts next sequence from MIN\_value.

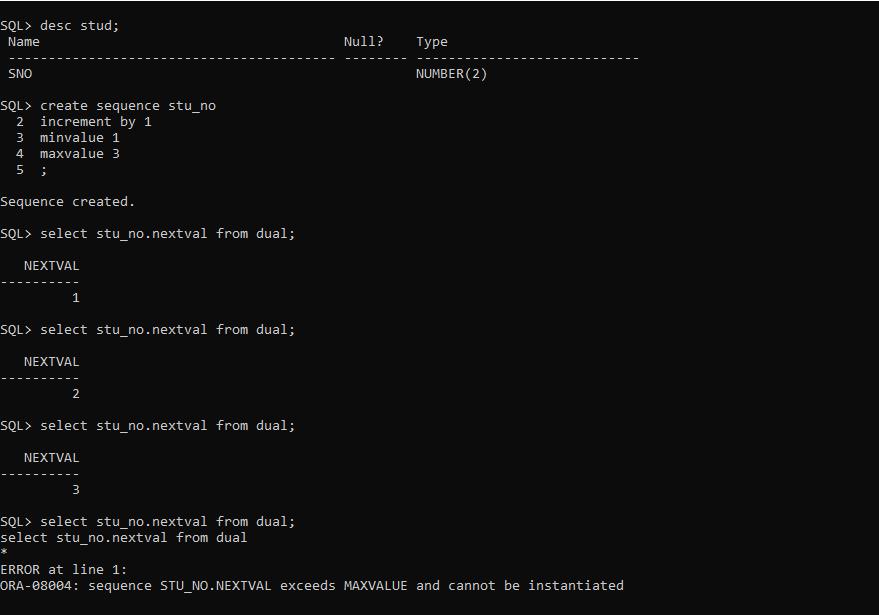
case2:when sequence reaches min\_value,it examines cycle properties value.If cycle property value is nocycle, it stops generating next value & display message to user; othe wise it begins next sequence(cycle) from max \_value.

* ORDER: default value is NO-ORDER.guarantees the sequence numbers to be generated in the order of request.
* No-order:- Does not guarantee the sequence number with order.
* LAST\_NUMBER:- default value is 1.when sequence generate value for first tiem, it becomes cache value + 1.

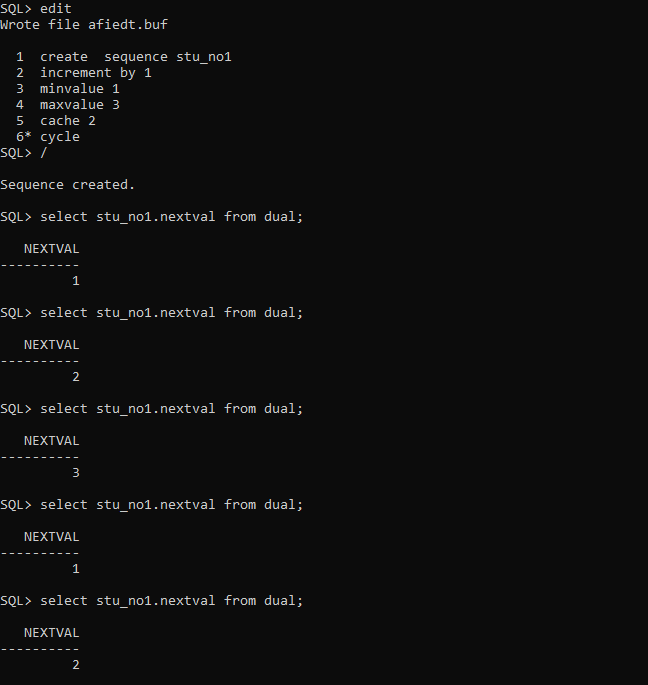
NOTE:- 1.user created sequences .The sequences information is stored in USER\_SEQUENCES table.

2.Once sequence is created with start with clause, we can't modify startwith clause value using alter statement.

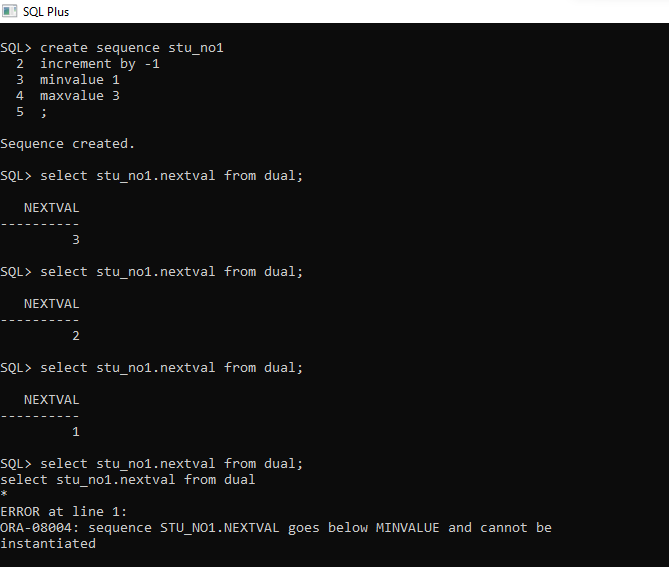
1. Incremental Sequence with no-cycle.



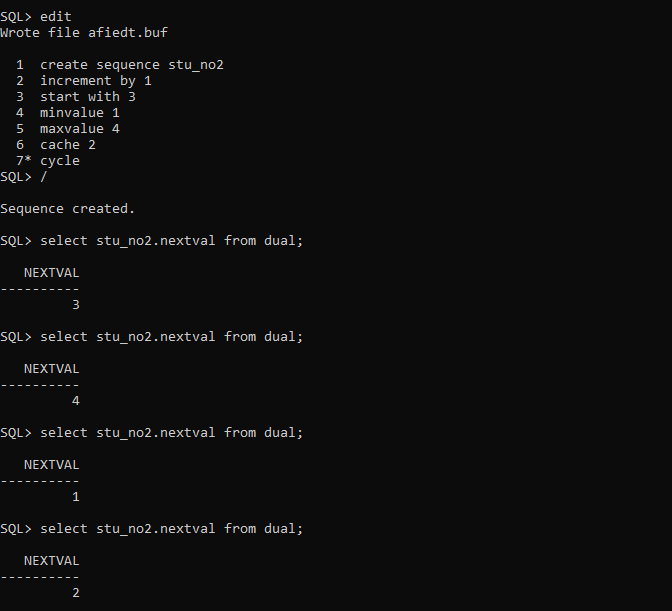
2.Incremental sequence with cycle



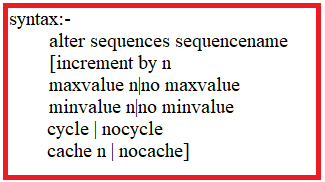
3.Decremental sequence with no cycle

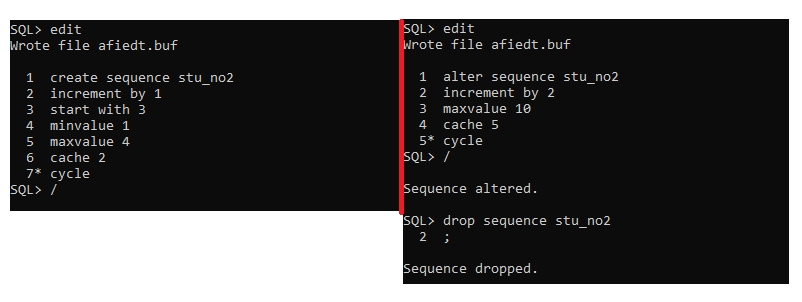


1. Incremental sequence with start with clause.



ALTER the sequence:-



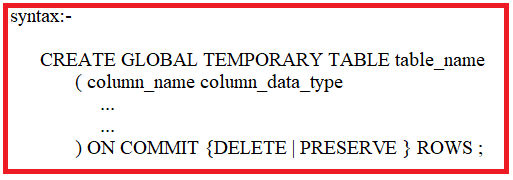


**12)TABLES:-** There are 4 types of tables**.**

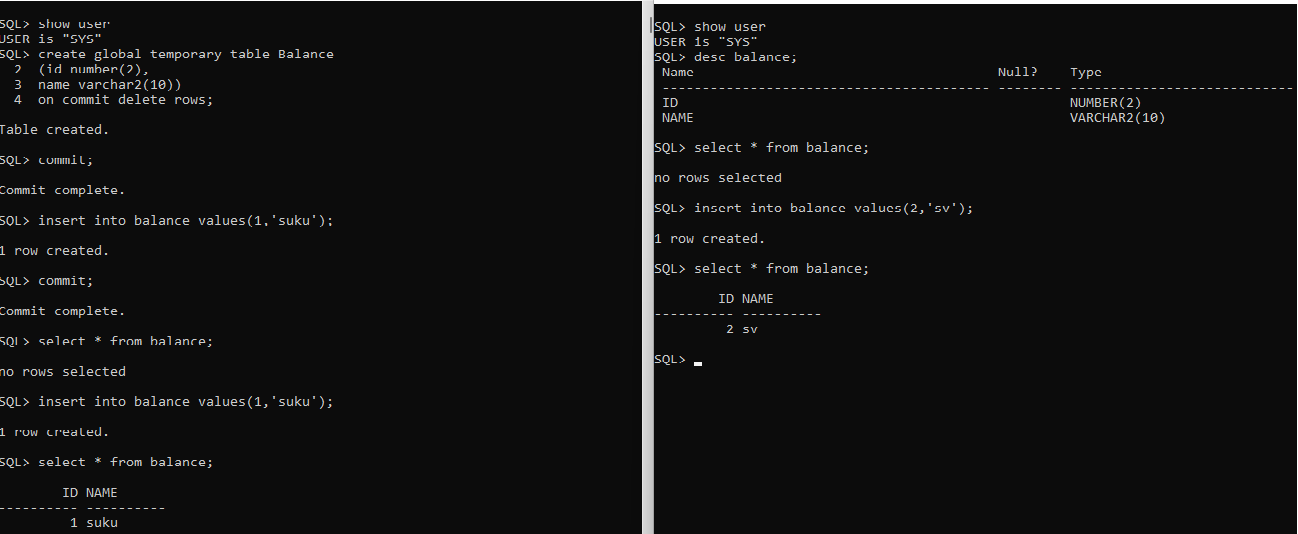
12.1**) Normal Table.**

**12.2) GTT:-**

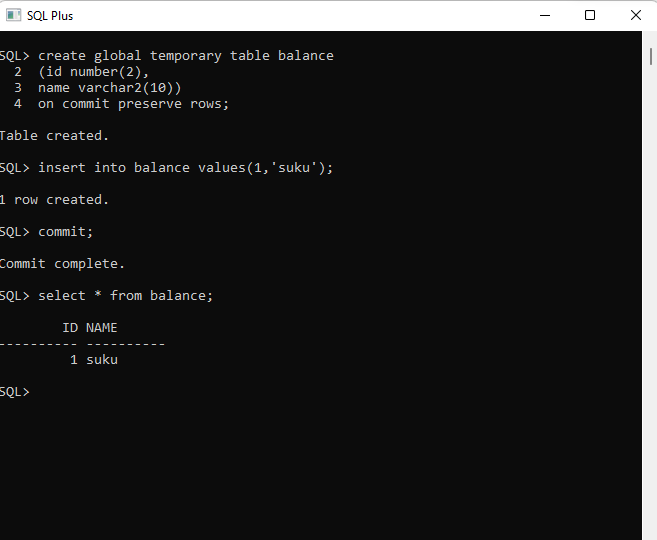
* Temporary table is also like ordinary table.The temporary table definition is shared among the multiples users and multiple sessions of same user.
* The rows are private to multiple users and multiple sessions of same user.
* The data which is in temporary tables is automatically deleted at end of session or end of transaction.
* Temporary tables has all features like ordinary table. But we can't estable foreign key relation between two temporary tables.



1. Global table with on commit delete rows option.



2.Global table with on commit preserve rows.



BENEFITS:-

-----------

1. THE records are generated while transaction is being processed. Those records will not be useful after transaction completed. To keep such records, we use global temporary tables.

2. Two users are working over the same table at the same time. The restriction is that no user access the other user data in same table.In this situation, we use global temporary table.

**12.3)External Table:-** The external table structure is perminent but the data which is external table is not perminent. The external table has data which is loaded from external file.

The data is unloaded from external table into external file.The external file should be in operating system file structure. The external file may be .txt file,.csv file.

Syntax:- create table tablename (columns definition)

ORGANIZATION ) EXTERNAL

( TYPE ORACLE\_LOADER | ORACLE\_DATAPUMP

DEFAULT DIRECTORY <ORACLE\_DIRECTORY\_OBJECT\_NAME>

ACCESS PARAMETERS

(

RECORDS DELIMITED BY NEWLINE

BADFILE <FILENAME>

DISCARDFILE <FILENAME>

LOGFILE <FILENAME>

[ READSIZE <BYTES>

[ SKIP <NUMBER\_OF\_ROWS>]

FIELDS TERMINATED BY 'TERMINATOR'

REJECT ROWS WITH ALL NULL FIELDS

MISSING FIELD VALUES ARE NULL

(<COLUMN NAME LIST>))\LOCATION('<FILENAME1> <FILENAME2' ...))

REJECT LIMIT <UNLIMITED | INTEGER >;

1. TYPE:- It specifies type of external table.There are two types of external tables which is

1)oracle\_loader

2)oracle \_datapump

Each type external table is supported by its own access driver.

-> oracle \_loader access driver is default. it can performs only loading operation.

-> oracle \_ datapump is another access driver.It can perform loading and unloading operation.

2. Default directory:- Here we should specify directory which contains datafiles.

3. BADFILE:- The badfile contains rejected records .The records may be rejected by mismatching datatype,mismatching colunmn length.

4. DISCARDFILE:- This file contains records which don't meet condition in where clause.

5. LOGFILE:- It contains that when loading is done & etc.

6. SKIP :- no.of lines will be skipped from the begining of the file.

7. LOCATION:- we specify file name onwhich loading or unloading done.

8. REJECT:- if we specify unlimited then bad records are moved to badfile and loading is being done.

if we specify n then when loading is terminated,n bad records are occured.

default value is unlimited.

* Constraints(primary key,foreign key,check,notnull,..etc) can't be defined on column of external table.
* Truncate & comment is not allowed on external table.

EXAMPLES:-

SQL> create directory sv as 'd:\sv';

directory created.

This direcoty is oracle object.The directory has not been created in d:\ by operating system file system.Therefore we should create directory.This direcotry

contains data files,logfiles,badfile,discard file.

Note:- Directory and datafile should be created before create the external table.

SQL>

1 create table emp\_load2(id number(7),name varchar2(7))

2 organization external

3 (type oracle\_loader

4 default directory sv

5 access parameters

6 (

7 records delimited by ';'

8 badfile 'a.txt'

9 discardfile 'b.txt'

10 logfile 'c.txt'

11 skip 1

12 fields terminated by ','

13 )

14\* location ('abc.txt'))

Table created.

SQL> select \* from emp\_load2;

ID NAME

---------- -------

2 veena

3 sula

4 suma

5 suha

SQL> create table emp\_load3(id number(7),name varchar2(7))

2 organization external

3 (type oracle\_loader

4 default directory sv

5 access parameters

6 (

7 records delimited by ';'

8 fields terminated by','

9 )

10 location('abc.txt', 'raji.txt')

11 );

Table created.

SQL> select \* from emp\_load3;

ID NAME

---------- -------

1 suku

2 veena

3 sula

4 suma

5 suha

9 'samba'

10 'sunil'

7 rows selected.

SQL> spool off;

=========================================================DATA PUMP==================================================

sql>

1 create table emp\_unload

2 organization external

3 (type oracle\_datapump

4 default directory sv

5 location ('suku.dmp')

6\* ) as select \* from emp\_load3

Table created.

SQL> select \* from emp\_unload;

ID NAME

---------- -------

1 suku

2 veena

3 sula

4 suma

5 suha

9 'samba'

10 'sunil'

7 rows selected.

SQL> spool off;

NOTE:-The suku.dmp file has been created in sv direcotry.when rows are fetched from suku.dmp,we try to select rows from emp\_unload.

**12.4) Partition Table**:- The large table is divided into smaller and manageble piecies.These piecies are called partitions.Each partition has same logical attributes(columnnames,datatypes,constrints...etc).

But Each partiton has different physical attributes such as (tablespace,pcttree,pctused).

Advantages:-

1.Maintaince operations focus only on particular portion of table instead of entire table.

example:

1) a database administrator could compress a single partition containing say the data for the year 2006 of a table, rather than compressing the entire table.

2) A data base administrator could backup/recovery single partition instead of entire table.

2.Another advantage of using partitioning is when it is time to remove data, an entire partition can be dropped which is very efficient and fast, compared to

deleting each row individually.

3.Partitioning improves query performance. In many cases, the results of a query can be achieved by accessing a subset of partitions, rather than the entire table. For some queries, this technique (called partition pruning) can provide order-of-magnitude gains in performance.

4. You can also run concurrent SELECT and DML operations against partitions that are unaffected by maintenance operations.

Partition Key:-

Each row in a partitioned table is unambiguously assigned to a single partition. The partition key is a set of one or more columns that determines the partition for each row. Oracle automatically directs insert, update, and delete operations to the appropriate partition through the use of the partition key.

A partition key:

* Consists of an ordered list of 1 to 16 columns
* Cannot contain a LEVEL, ROWID, or MLSLABEL pseudocolumn or a column of type ROWID
* Can contain columns that are NULLable

When to Partition a Table:-

Here are some suggestions for when to partition a table:

* Tables greater than 2GB should always be considered for partitioning.
* Tables containing historical data, in which new data is added into the newest partition. A typical example is a historical table where only the current month's data is updatable and the other 11 months are read only.

Partition methods:-

Oracle provides following partition.

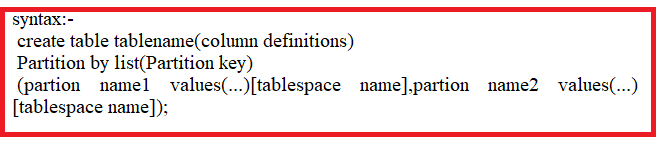
1.List partition

2.Range partition.

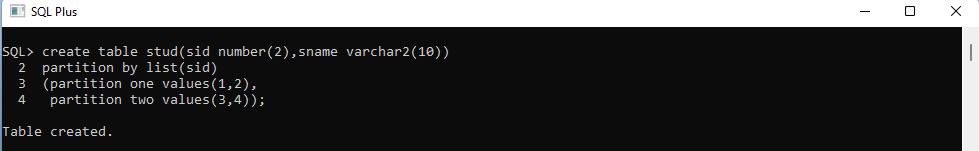
3.Hash partition.

4.Composite partition.

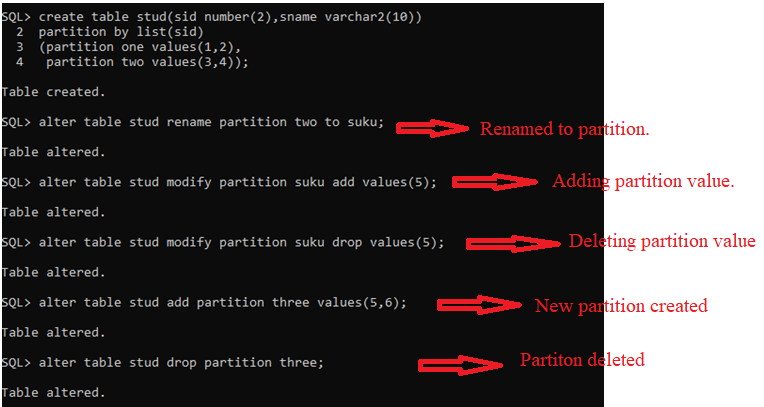
**1.List partition:-** The data distribution is defined by a list of values of the partitioning key. A special 'DEFAULT' partition can be defined to catch all values for a partition key that are not explicitly defined by any of the lists.



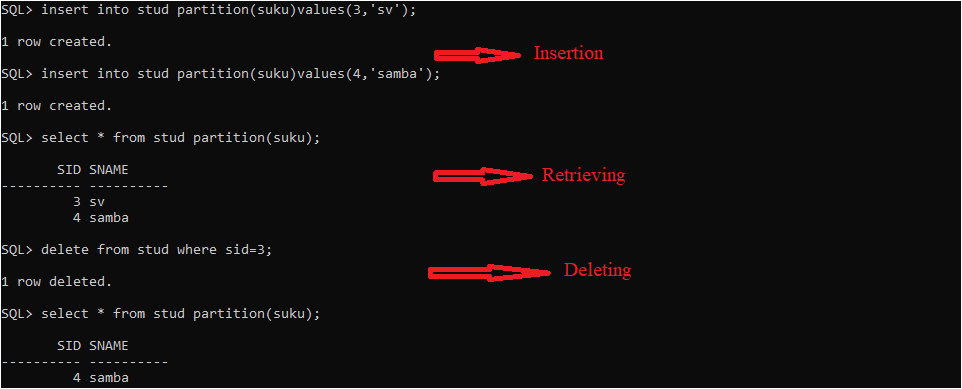
Example:-

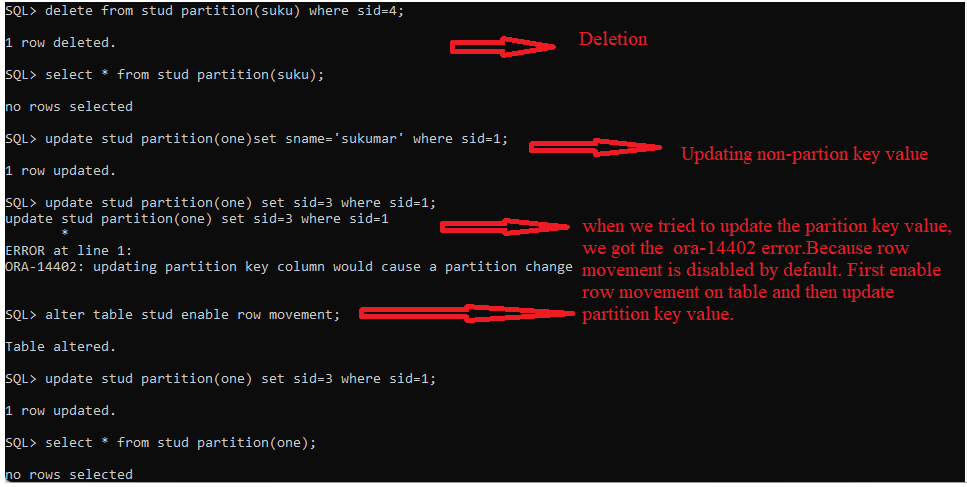




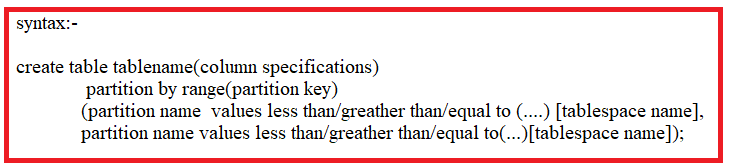




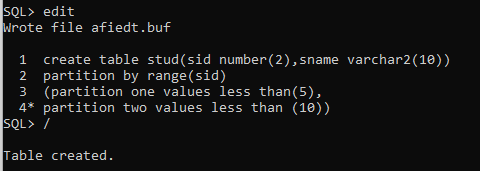




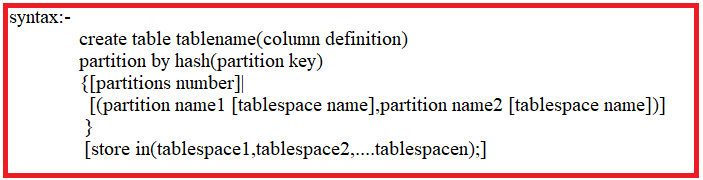
2.Range Partition:-The data is distributed based on a range of values of the partitioning key (for a date column as the partitioning key, the 'January2007' partition contains rowswith the partitioning-key values between '01-JAN-2007' and '31-JAN-2007').



Example:-



3.Hash:- A hash algorithm is applied to the partitioning key to determine the partition for a given row. Unlike the other two data distribution methods, hash does not provide any logical mapping between the data and any partition.



Example:-



4. Composite Partition:-There are several composite partitions. few of them are

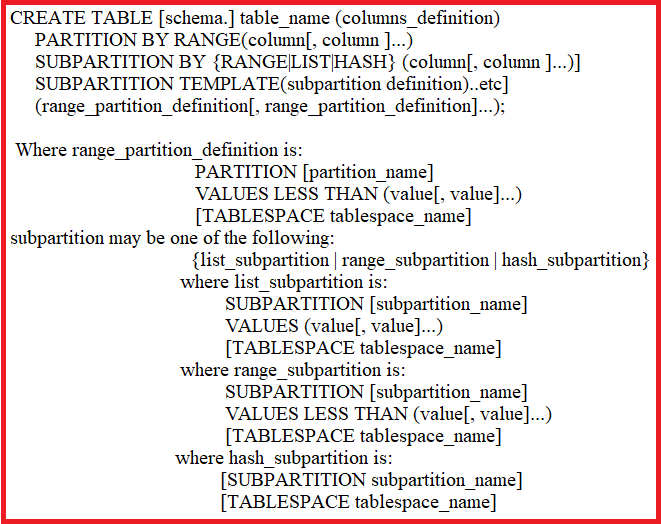
4.1)Range-list partition

4.2)Range-hash partition.

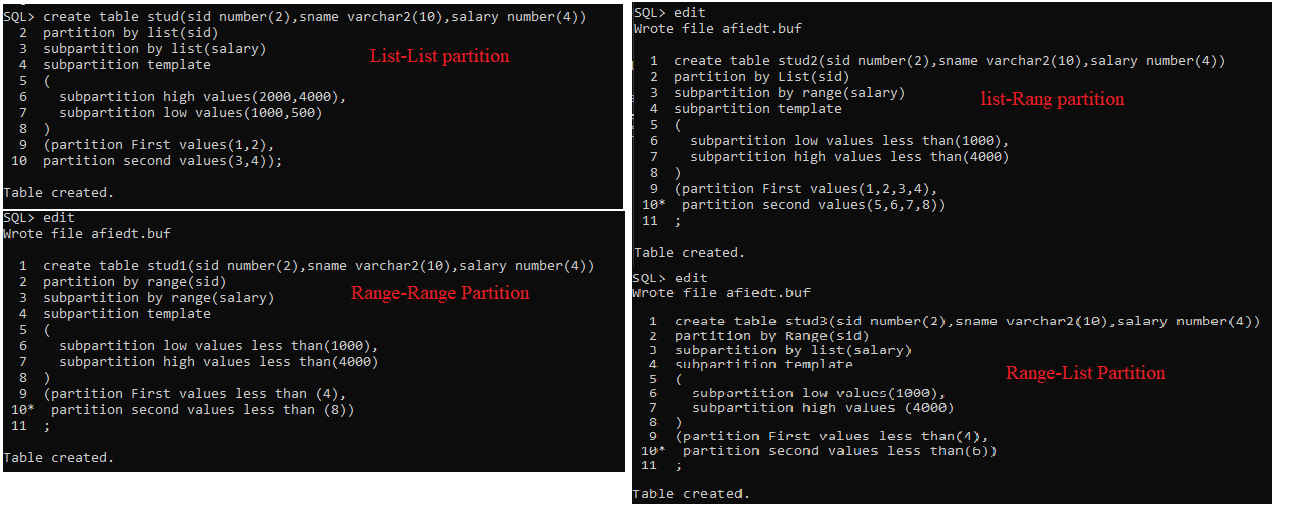
4.3)Range-Range partition.

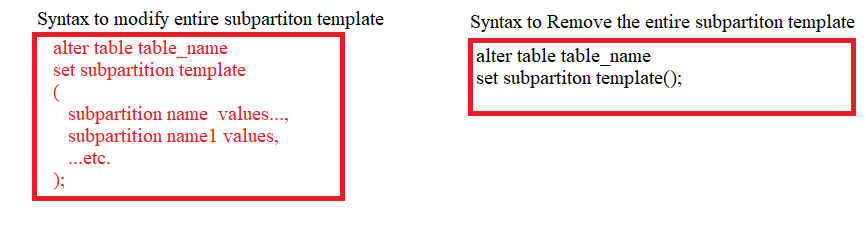
4.4)list-range partition.

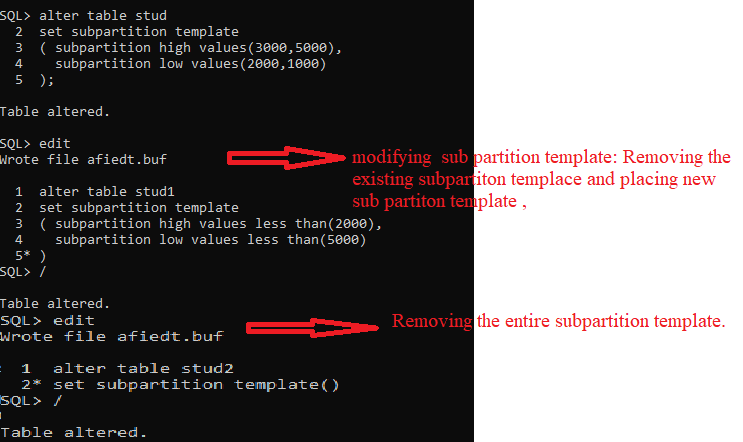
4.1) Range-List,Range-hash,Range-Range partition:-



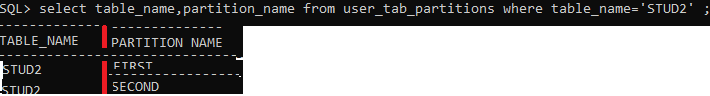
Example: List-List partition.







Note:- All partitons information of user schema is stored in “user\_tab\_schema” table. From this table, we can select the partitions of specific table.



**13)USER DEFINED DATATYPE**:-

=> when pre-defined data types will not meet the business requirements, we can create our own data types.

Oracle 8i onwards, There are two types of user-defined data types.

1. composite type/Objec type.
2. Collection.

a.varray.

b. Associative array.

**OBJECT TABLES**

-> IF table contains object type as column ,then such table is called OBJECT TABLE.iT HAS UNIQUE ID WHICH IS CALLED (OID).

-> The object table automatically INHERITS columns of object type.

-> Each row in object ASSOCIATES WITH object identifier(OID) which is unique through out db.

-> OID is generated and assigned by oracle,when row is created.

->The row in object table can be referenced by other object(object type).

-> object table information is maintained in USER\_OBJECTS data dictionary.

create table name(col1 built\_indatatype,col2 object type name {[not null|defualt|check]},....);

-->WE CAN'T DEFINE PRIMARY AND UNIQUE CONSTRAINTS ON USER DEFINED DATA TYPE.

**13.1.Object Type:-**

-> The Object type information is stored permanetly in data dictonary which is USER\_TYPES.

-> The object type allows group of elements of different data types.

Syntax:-

CREATE OR REPLACE TYPE NAME

AS OBJECT

(COLS DEFINITIONS);/

Examples:-

SQL> create or replace type address as object(dno number(2),town varchar2(10));

2 /

Type created.

SQL> create table student\_1(sid number(2),sname varchar2(10),addr address);

Table created.

SQL> set describe depth 4 linenum off indent on;

SQL> desc address;

Name Null? Type

----------------------------------------- -------- ----------------------------

DNO NUMBER(2)

TOWN VARCHAR2(10)

SQL> desc student\_1

Name Null? Type

----------------------------------------- -------- ----------------------------

SID NUMBER(2)

SNAME VARCHAR2(10)

ADDR ADDRESS

DNO NUMBER(2)

TOWN VARCHAR2(10)

SQL> insert into student\_1 values(1,'suku',address(12,'nlr'));

1 row created.

SQL> insert into student\_1 values(1,'veena',address(11,'clx'));

1 row created.

SQL> select sid,sname,r.addr.dno,r.addr.town from student\_1 r;

SID SNAME ADDR.DNO ADDR.TOWN

---------- ---------- ---------- ----------

1 suku 12 nlr

1 veena 11 clx

SQL> drop type address;

Type dropped.

----\*\*\* WE CAN'T INSERT VALUES INTO TYPE OBJECT .

**13.2. Collections:-**

1. **Varray:-**

**Syntax:**

**Create type type\_name**

**Is**

**Varray(array\_size)of data\_type([element\_size]);**

**Example:-**

SQL> create type ph\_no is

2 varray(3)of number(2);

3 /

Type created.

SQL> create table emp(eid number(2),p\_no ph\_no);

Table created.

SQL> desc emp;

Name Null? Type

----------------------------------------- -------- ----------------------------

EID NUMBER(2)

P\_NO PH\_NO

SQL> set describe depth 4 linenum off indent on;

SQL> desc emp;

Name Null? Type

----------------------------------------- -------- ----------------------------

EID NUMBER(2)

P\_NO PH\_NO

SQL> insert into emp values(1,ph\_no(1,2));

1 row created.

SQL> insert into emp values(1,ph\_no(3,4));

1 row created.

SQL> select \* from emp;

EID

----------

P\_NO

--------------------------------------------------------------------------------

1

PH\_NO(1, 2)

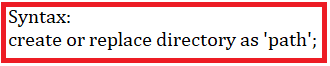
1

PH\_NO(3, 4)

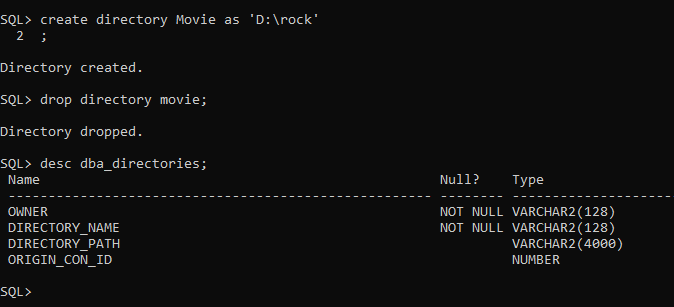
**14)DIRECTORY**:- The directory is DB object. This directory name is alias(another name) for full path of directory which physically existed in OS. DBMS does not physically create folder at specified location.

when we write data from db to external files or when we read data from external files to database,we need to intreact with os file(external file).These files are in folder to which we should create alias in oracle.

Examples:-



-> View name is DBA\_DIRECTORIES which contains information of oracle directories.



Q) CAN WE CHANGE DIRECTORY NAME?

NO. WE can't change directory name. but we can alter path of directory.

**15)DB-LINK:-**

---> The database link establish communication channel b/w only two different databases.

---> Now we connect to one of two database which is called local db and other database is called remote database.

---> After communication channel is established b/w local & remote db, we can bring the data from remote database into local database.

---> The remote DB should be up and running otherwise we can't get data into local db from remote db.

---> To store remote database data in local database, we materilazed view with database link.

There are four types of database links:

1.Private(It is default value and Dblink is only private to user).

2.public (It is accessable by any user which is in any database).

3.shared (when multiple connections are going to be opened,dblink is shared by multiple connections).

4.Global (It is used by any user of current database.)

Syntax:-1

create or replace {public|private|shared|global} database link name

connect to current\_user

using 'dbname';

Syntax:-2

create or replace {public|private|shared|global} database link name

connect to username

identified by pwd

[Authenticated by username identified by pwd]-> it is only for shared type dblink.

using 'dbname';

-->although dbname doe's not exist,database link is created.

-->although username does not exist,database link is created.

-->although pwd is rong,database link is created.

--> But we face problem when we try to acces the data from remote database.

Examples:-

SQL> create database link abc

2 connect to scott

3 identified by sulamaha

4 using 'orcl2';

Database link created.

SQL> create database link abc1

2 connect to current\_user

3 using 'orcl2';

Database link created.

SQL> select \* from person@orcl2;

select \* from person@orcl2

\*

ERROR at line 1:

ORA-02019: connection description for remote database not found

**16)INDEXES**:- Index is database object.The Index contains entry for each value that appears

in the indexed columns of the table or cluster.

Advantages:-

1.Query performance is improved in searching,updating,deleting and insertion operation.

->The Index information is maintained in two tables which is USER\_INDEXES AND USER\_IND\_COLUMNS.

->If table is temporary table then index is also temporary index with same scope.

->either on single column or more than single column(UPTO 32 COLUMNS) ,Index is created.If index is created on more than one column

then such column is called COMPOSITE INDEX.

\*\*\*->The data is copied from data segment into index segment of indexed column.By default data which in index segment is arranged in ascending order.

SYNTAX:-

CREATE {[UNIQUE]/[BITMAP]}

INDEX indexname

ON

Tablename(columnname[,columnname...])

Tablespace tablespacename;

Q) when do we create index?

-> The column contains wide range of values.

-> The column contains lage no.of null values.

-> The column is frequently used in where clause or join condition.

Q) when don't we create index?

-> table is too small.

-> column value is frequently updated.

16.1)INDEX TYPES:-

**16.1.1)NORMAL INDEX**:-

-> We create index on column with out specfing index type. Such index is called normal index.

-> Except BLOB type columns,we can create index on basic type(number,varchar2,date,...) columns.

-> The column on which index created can contain NULL values and DUPLICATE VALUES.

Examples:-

SQL> DESC SAMPLE;

Name Null? Type

----------------------------------------- -------- ----------------------------

ID NUMBER(2)

NAME NUMBER(2)

SQL> CREATE INDEX ONE ON SAMPLE(ID);

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE

------------------------------ ------------------------------ ---------------------------

ONE SAMPLE NORMAL

PK\_EMP EMP NORMAL

PK\_DEPT DEPT NORMAL

SQL> INSERT INTO SAMPLE VALUES(1);

1 row created.

SQL> INSERT INTO SAMPLE VALUES(1);

1 row created.

SQL> INSERT INTO SAMPLE VALUES(null);

1 row created.

SQL> SELECT \* FROM SAMPLE;

ID

----------

1

1

SQL> DROP INDEX ONE;

INDEX DROPED.

SQL> CREATE INDEX ONE ON SAMPLE(ID,NAME);

Index created.

**16.1.2)UNIQUE INDEX**:- we create unique index on column(s) which don't allow duplicate value but can contain null values.

EXAMPLES:-

SQL> DESC SAMPLE;

Name Null? Type

----------------------------------------------------------------- -------- --------------------------------------------

ID NUMBER(2)

NAME VARCHAR2(7)

SQL> CREATE UNIQUE INDEX ONE ON SAMPLE(ID);

CREATE UNIQUE INDEX ONE ON SAMPLE(ID)

\*

ERROR at line 1:

ORA-01452: cannot CREATE UNIQUE INDEX; duplicate keys found

SQL> DELETE \* FROM SAMPLE;

DELETE \* FROM SAMPLE

\*

ERROR at line 1:

ORA-00903: invalid table name

SQL> DELETE FROM SAMPLE;

3 rows deleted.

SQL> CREATE UNIQUE INDEX ONE ON SAMPLE(ID);

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES

------------------------------ ------------------------------ --------------------------- ---------

ONE SAMPLE NORMAL UNIQUE

PK\_EMP EMP NORMAL UNIQUE

PK\_DEPT DEPT NORMAL UNIQUE

SQL> INSERT INTO SAMPLE VALUES(1,'SUKU');

1 row created.

SQL> INSERT INTO SAMPLE VALUES(1,'SUKU');

INSERT INTO SAMPLE VALUES(1,'SUKU')

\*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.ONE) violated

**16.1.3)BIT MAP INDEX**:-

->Bit map index is created on low cardinality column(s).Cardinality is no.of distinct values in column.

->Bitmap indexes store rowid's associated with key value as bitmap

->Bitmap indexes should be used only when data is infrequently updated.

->The column onwhich bitmap index created allows duplicate values.

Restrictions:-

->Bitmap cannot be specified with unique index.

->Bitmap index can't be specified when creating global partition table.

->Bitmap index can't be specified for domain index.

->bitmap index can't be specified for table which involves OLTP.

EXAMPLES:-

SQL> create table abc(id number(2)) partition by list(id)(partition one values(1));

Table created.

SQL> create bitmap index a on abc(id);

create bitmap index a on abc(id)

\*

ERROR at line 1:

ORA-25122: Only LOCAL bitmap indexes are permitted on partitioned tables

SQL> create bitmap index two on abc(id);

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES

------------------------------ ------------------------------ --------------------------- ---------

ONE SAMPLE NORMAL UNIQUE

PK\_EMP EMP NORMAL UNIQUE

PK\_DEPT DEPT NORMAL UNIQUE

TWO ABC BITMAP NONUNIQUE

**16.1.4)FUNCTION BASED NORMAL INDEX:-**

Let us assumed,We created normal index on column(name).

Q:)-select \* from tablename where name='suku'.

while executing above query, query optimizer uses the index of NAME.

Q:) select \* from tablename where upper(name)='suku'.

while executing above query ,query optimizer DOES NOT USE index of NAME. Therefore query performence will be low.

if the index column is used with dbms\_built in function or index column is used in expressions,then query optimizer doe's not user bitmap index.

To overcome above problem,Another index has been introduced.That is function based normal index.

Examples:-

SQL> create index one on sample(upper(name));

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES

------------------------------ ------------------------------ --------------------------- ---------

ONE SAMPLE FUNCTION-BASED NORMAL NONUNIQUE

PK\_EMP EMP NORMAL UNIQUE

PK\_DEPT DEPT NORMAL UNIQUE

**16.1.5)FUNCTION BASED BITMAP INDEX**:-

let us assumed,We created BITMPA-index on column(name).

Q:)-select \* from tablename where name='suku'.

while executing above query, query optimizer uses the BITMAP index of NAME.

Q:) select \* from tablename where upper(name)='suku'.

while executing above query ,query optimizer DOES NOT USE BITMAP index of NAME. Therefore query performence will be low.

if the index column is used with dbms\_built in function or index column is used in expressions,then query optimizer doe's not user bitmap index.

To overcome above problem,Another index has been introduced.That is function based BITMAP index.

EXAMPLE:-

SQL> CREATE BITMAP INDEX ONE ON SAMPLE(UPPER(NAME));

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES

------------------------------ ------------------------------ --------------------------- ---------

ONE SAMPLE FUNCTION-BASED BITMAP NONUNIQUE

PK\_EMP EMP NORMAL UNIQUE

PK\_DEPT DEPT NORMAL UNIQUE

**16.1.6)REVERSE KEY INDEX**:-

->The data is copied from data segment into index segment in reverse order.

DATASEGMENT DATA:

-----------------

AA1

AA2

AA3

INDEXSEGMENT DATA:

-----------------

1AA

2AA

3AA

Syntax:- create [bitmap]|unique index on tablename(col1[,col2])reverse;

Examples:-

SQL> create index one on sample(id)reverse;

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES

------------------------------ ------------------------------ --------------------------- ---------

ONE SAMPLE NORMAL/REV NONUNIQUE

PK\_EMP EMP NORMAL UNIQUE

PK\_DEPT DEPT NORMAL UNIQUE

**16.1.7)GLOBAL INDEX**:-

->Global index can be created on normal indexed column(s) which is in normal table or partitioned table.

->global index can't be created on bitmap indexed column(s) which is also in normal table or partitioned table.

Syntax:- create [unique] index on tablename(col1[,col2])global;

Examples:-

SQL> create index one on sample(id)global;

Index created.

SQL> create table rock(id number(7))partition by list(id)(partition one values(2));

Table created.

SQL> create index two on rock(id)global;

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS,partitioned FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES PARTITIONED

------------------------------ ------------------------------ --------------------------- --------- -----------

ONE SAMPLE NORMAL NONUNIQUE NO

TWO ROCK NORMAL NONUNIQUE NO

PK\_EMP EMP NORMAL UNIQUE NO

PK\_DEPT DEPT NORMAL UNIQUE NO

**16.1.8)LOCAL INDEX**:-

-> local index can only be created column(s) which should be in PARTITIONED TABLE.

-> The local index is also divided into partitions.No of index partitions is equal to no.of table partitions.Ever index partition is belong to one appropriate table partition.

Syntax:- create [unique] index on tablename(col1[,col2])local;

Examples:-

SQL> create table rock(id number(7))partition by list(id)(partition one values(2));

Table created.

SQL> create index one on rock(id)local;

Index created.

SQL> select INDEX\_NAME,TABLE\_NAME,INDEX\_TYPE,UNIQUENESS,partitioned FROM USER\_INDEXES;

INDEX\_NAME TABLE\_NAME INDEX\_TYPE UNIQUENES PAR

------------------------------ ------------------------------ --------------------------- --------- ---

ONE ROCK NORMAL NONUNIQUE YES

PK\_EMP EMP NORMAL UNIQUE NO

PK\_DEPT DEPT NORMAL UNIQUE NO

**16.1.9)SORT/UNSORT:-**

Syntax:- create index indexname on tablename(col1[,col2])SORT|NOSORT;

\*sort is default value.

9.1)SORT:-

->IF the data in column is not in either ascending order or descending order and we want to create index on such colum,THEN ONLY we have to use sort option .

-> Now The data which is in index column is first sorted in ascending order then after the ordered data is copied into indexed segment.

EXAMPLE:-

SQL> select \* from sample;

ID NAME

---------- -------

10 sv

37 sv1

32 rock1

1 SUKU

32 rock

SQL> create index two on sample (id)sort;

Index created.

SQL> select \* from sample 2 ;

ID NAME

---------- -------

1 suku

2 sv

3 rock

SQL> create index three on sample(id) sort;

Index created.

9.2)UNSORT:-

->If the data in column is already in either ascending order or descending order and we want to create index on such column,

THEN ONLY we have to use unsort option.Other wise we get error message.

-> The data is first unsorted then after it is copied into index segement.

EXAMPLE:-

ID NAME

---------- -------

10 suku

7 sv

15 sv1

SQL> create index two on sample (id)nosort;

create index two on sample (id)nosort

\*

ERROR at line 1:

ORA-01409: NOSORT option may not be used; rows are not in ascending order

SQL> select \* from sample;

ID NAME

---------- -------

1 suku

2 sv

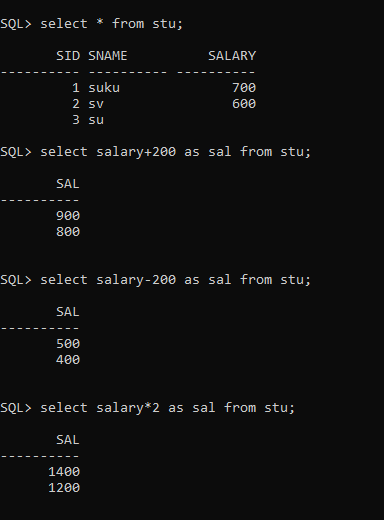
SQL> create index two on sample (id)nosort;

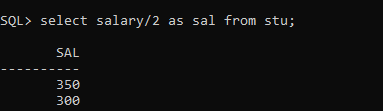
Index created.

**20**.NULL Value: The sailor joined in club. The sailor does not yet has rating value, while his record is being inserted into sailor table.

For such situation, Sql provides special value which is NULL. We keep null value in column, when column value is **unknown**.

**20.1.Arithmetic operators +,-,\*,&,/ return null , if one of their argument is ‘null’.**

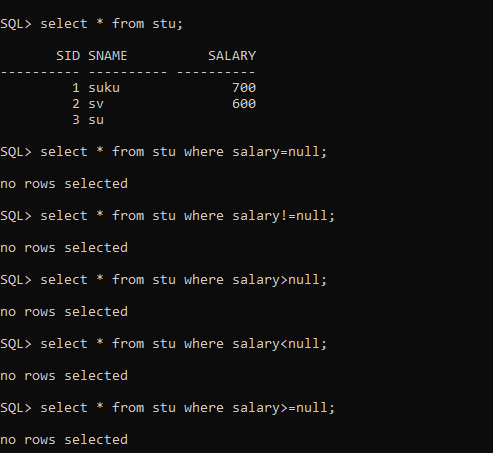




**20.2)Comparison using NULL:**

\*If we compare two null values using <,>,>=,<=,!=,= ,then result is always unknown.

\* If one of operand value is unknown in comparison,the result is also unknown.



20.3.AND,OR,NOT:

1.NOT unknown result is always unknown.

2.

a) unknown v unknown

It’s truth value is unknown.

b)unknown V false

It’s truth value is unknown.

c)false Vs unknown

It’s truth value is unknown.

d)unknown v true

It’s truth value is true.

e)true vs unknown

It’s truth value is true.

3.

a)Unknown ^ true

Truth value is unknown.

b)true ^ unknown

Truth value is unknown.

c)unknown ^ unknown

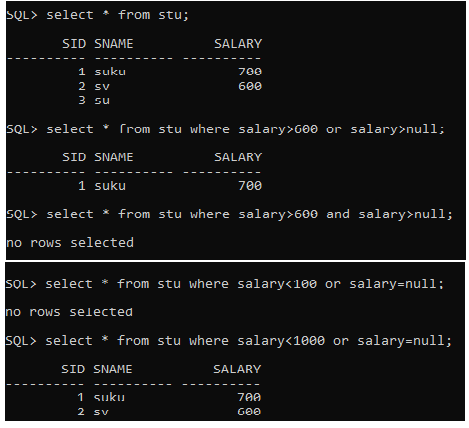
Truth value is unknown.

d)False ^ unknown

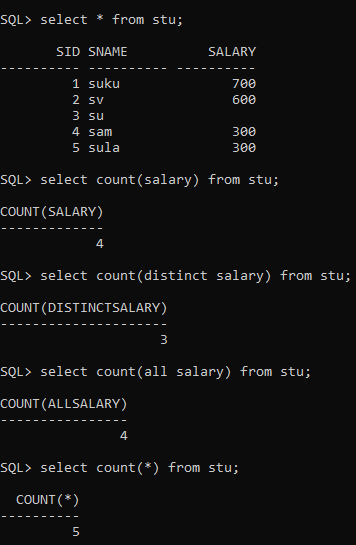
Truth value is false.

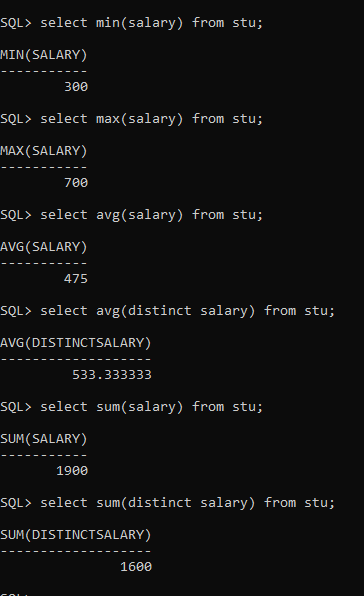
e)unknow ^ False

Truth value is false.



**20.4. All aggregate operators(count,sum,avg,min,max) discard (0r) ignore null values by default.**

****

****

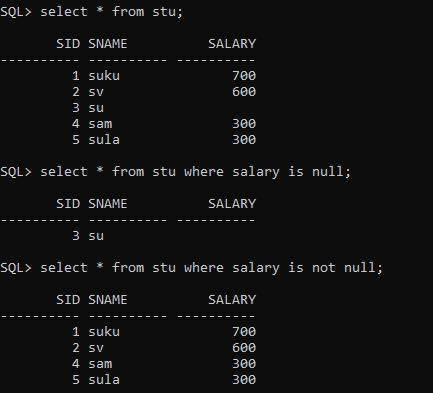
**20.5. Oracle provides two operator to test whether column value is null or not.**

1.ISNULL

2.ISNOTNULL

SQL provides IS NULL operator to test whether column value is null.

SQL provides IS NOT NULL operator to test whether column value is Not Null.



20.4)Disallowing Null values:- If we specify NOT NULL in field definition, that field willnot allow NULL value.

20.5 nvl Functions:- The oracle provided following null value functions.

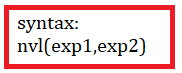
a.nvl()

b.nvl2()

c.nullif()

d. coalesce()

A.nvl():-It allows you to replace null with user supplied value in column.



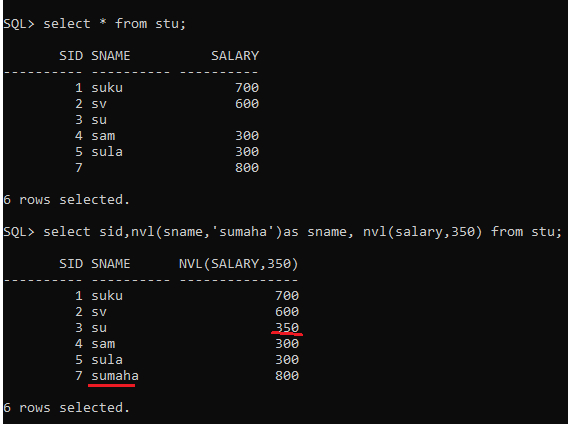
It accepts two arguments. If exp1 evaluates to null, then nvl() function returns exp2. If exp1 evaluates to non-null, nvl()function returns exp1.

The exp1and exp2 can have same or different data types. If their data types are different, oracle implicit converts one to other according to following rules:

i.if the data type of exp1 is character , oracle converts exp2 to data type of exp1.

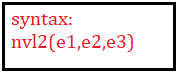
ii. if the data type of e1 is numeric, oracle determines which argument has the highest numeric precedence , implicitly converts the other argument to that dat type , and returns that data type.

iii. if oracle can not implicitly convert one data type to another data type, it raises an error.



B.NVL2():-The Oracle NVL2() function is an extension of the [NVL()](https://www.oracletutorial.com/oracle-comparison-functions/oracle-nvl/) function with different options based on whether a NULL value exists.

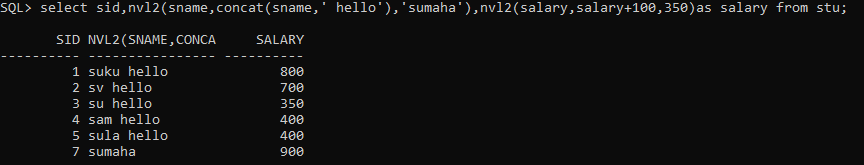
The Oracle NVL2() function accepts three arguments. If the first argument is not null, then it returns the second argument. In case the first argument is null, then it returns the third argument.



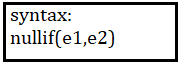
In this syntax, the first argument e1 can be a value of any data type. The second and third arguments can be values of any data types except LONG.

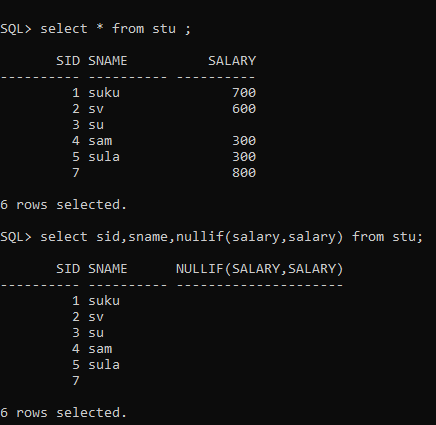
In case the data types of e2 and e3 are different:

* If e2 is numeric, Oracle implicitly converts either e2 or e3 to the data type of the higher precedence and returns a value of that data type.
* If e2 is character data, Oracle converts e3 to the data type of e2 before comparing them and returns VARCHAR2 in the character set of e2. Oracle does not carry this conversion in case e3 is a null constant because it is not necessary.



C.NullIF():-The Oracle NULLIF() function accepts two arguments. It returns a null value if the two arguments are equal. In case the arguments are not equal, the NULLIF() function returns the first argument.

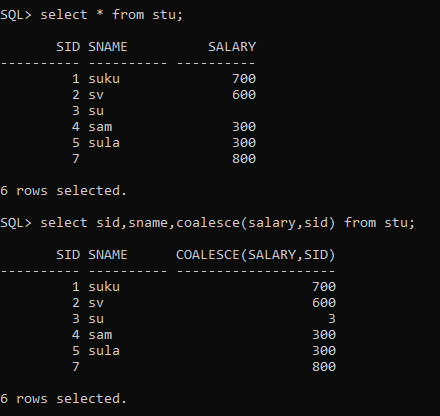




D.Coalesec():-The Oracle COALESCE() function accepts a list of arguments and returns the first one that evaluates to a non-null value.



Example:-



Diff between coalesce() and nvl():-

* **Coalesce() is ANSI standard function. It takes n no.of arguments.**
* **Nvl() is oracle function.It takes only 2 arguments.**
* **If coalesce() have only two parameters, then it works like nvl() function.**

**21.Pseudo Columns:-** A **pseudocolumn** behaves like a table column, but is not actually stored in the table. You can select from pseudocolumns, but you cannot insert, update, or delete their values. pseudocolumns typically return a different value for each row.

a. Sequence pseudo columns

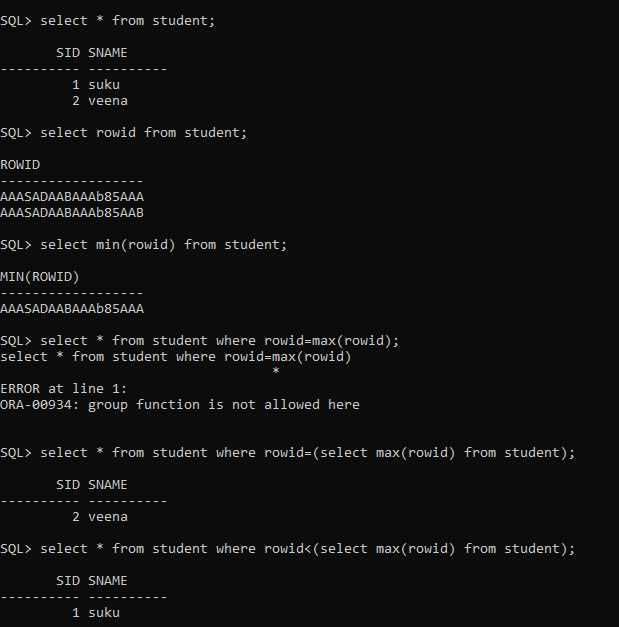
b. ROWID pseudo column

c. ROWNUM pseudo column

d. Hierarchical Pseudo columns.

**21.a.RowId:-** The ROWID stores physical address of a row in table. In oracle whenever we are inserting row into a table then oracle server internally automatically generates a UNIQUE identification number for identifying a record uniquely in hexa decimal format. This is called “ROWADDRESS” OR “ROWID”.

We can retrieve records very fastly from table using rowed. It improves the query performance.



**21.b.RowNum:-**  This column have the numbers of each row in table.

At the time of record selection, DBMS automatically assign number to row and store in “ROWNUM” pseudo column.

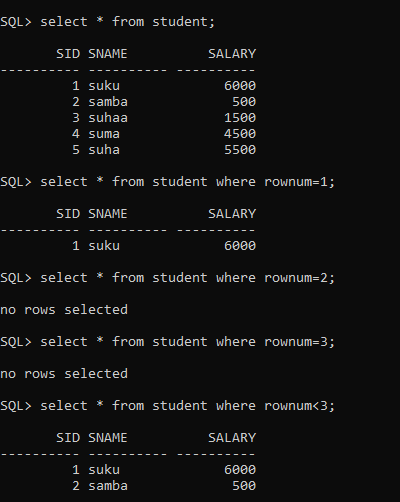
Note:- In eqvalient operation, ROWNUM work with only value “one”. But It

does not work more than one.

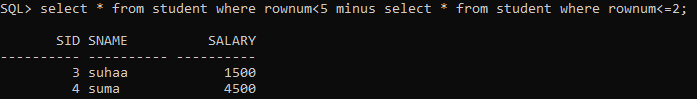
Ex: ROWNUM=1 //valid

ROWNUM=2 or >=2 //valid but no row selected from table.

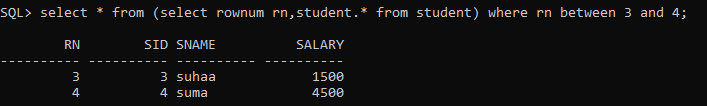
It works with < and <= operator.



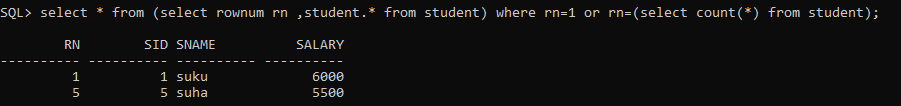
Q: Write a Query to display records between 2 to 5 from student table.



Or



Q.Write a query to display the first and last record from student table.



Q.Write a Query to display the 3rd highest salary.

