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.

**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.

syntax:-1

create table tablename(column definitions);

column definition:

column-name datatype(size) [constraint1,constraint2..n]

syntax:-2

create table tablename1 as select [\*]|col-1,col-2,..col-n from tablename2 ;[where clause;]

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

Table created.

SQL> desc sample

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> create table sample1 as select \* from sample;

Table created.

SQL> desc sample1;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> create table sample2 as select id from sample;

Table created.

SQL> desc sample2

Name Null? Type

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

ID NUMBER(7)

SQL> insert into sample1 values(1,'suku');

1 row created.

SQL> insert into sample1 values(2,'sv');

1 row created.

SQL> insert into sample1 values(3,'sula');

1 row created.

SQL> create table sample3 as select \* from sample where id >=2;

Table created.

SQL> select \* from sample3;

no rows selected

SQL> desc sample3;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> create table sample4 as select \* from sample1 where id > =2;

Table created.

SQL> select \* from sample4;

ID NAME

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

2 sv

3 sula

**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.

syntax:- truncate table tablename;

SQL> truncate table sample4;

Table truncated.

SQL> rollback;

Rollback complete.

SQL> select \* from sample4;

no rows selected

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

syntax:- Rename oldname to newname

SQL> rename sample4 to sample41;

Table renamed.

SQL> select \* from sample4;

select \* from sample4

\*

ERROR at line 1:

ORA-00942: table or view does not exist

SQL> select \* from sample41;

no rows selected

**4)DROP**:- This stament removes dbobject from database. The removed object is now kept in recyclebin.

syntax:- drop table/sequence/..etc objectname [purge];

SQL> drop table sample41;

Table dropped.

**5)FLASHBACK:-**

syntax:-

FLASHBACK TABLE/sequence ..etc table\_name

[TO BEFORE DROP] |

[TO TIMESTAMP time\_stamp] |

[TO SCN scn\_number] |

[ENABLE TRIGGERS | DISABLE TRIGGERS];

\* 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 truncate table using drop command,all records are deleted. Flashback restores table to prior version using timestamp or scn\_numbers.

EXAMPLES:-

SQL> select \* from sample;

ID NAME

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

1 suku

2 veena

SQL> drop table sample;

Table dropped.

SQL> show recyclebin;

ORIGINAL NAME RECYCLEBIN NAME OBJECT TYPE DROP TIME

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

ROCK BIN$LoVtYFXZTLy0gESZQ0IjLQ==$0 TABLE 2020-02-21:18:11:28

SAMPLE BIN$UcLpREIsSpmVadDCSRea9w==$0 TABLE 2020-02-22:09:36:41

SAMPLE1 BIN$JqlPw2auSIGvWh3hfo78ug==$0 TABLE 2020-02-22:09:34:22

SAMPLE1 BIN$JYrJkBa9TfiC8x6dSMcTqA==$0 TABLE 2020-02-21:18:11:00

SAMPLE2 BIN$lT1NtTsERC+gyGUmVHe1qw==$0 TABLE 2020-02-22:09:34:25

SAMPLE3 BIN$C4+JOKIpSQWir78f8TZctA==$0 TABLE 2020-02-22:09:35:16

SAMPLE41 BIN$tG2/4J7/SAijAxG9q71Y9A==$0 TABLE 2020-02-22:09:18:44

SUKUMAR BIN$b4LXfh3mSDS3fdiFObZ8JA==$0 TABLE 2020-02-21:11:46:22

SUKUMAR BIN$b88ZoDYSQWWdAD9Ab83kaw==$0 TABLE 2020-02-21:07:41:26

SUKUMAR BIN$ozh9q4sXTwGWw/ON254dZw==$0 TABLE 2020-02-21:07:25:24

SUKUMAR BIN$tutDFJqtSi6uI5LaAD2erQ==$0 TABLE 2020-02-21:07:24:02

SUKUMAR BIN$S7giN23rQ5ehoMy89QfAaQ==$0 TABLE 2020-02-20:19:13:12

SUKUMAR1 BIN$qZnn35SQRMWEBvPM2ZHbfA==$0 TABLE 2020-02-20:20:09:14

SV BIN$L8OI9Ab5QgWHKU054uiJ3w==$0 TABLE 2020-02-21:11:54:50

SV BIN$Z3i9kGaMTQGfn2mc3dVGOw==$0 TABLE 2020-02-21:11:46:11

SV BIN$tIyJHEY+S2qlRnqqKsjo8A==$0 TABLE 2020-02-21:10:03:00

SV BIN$xZ6x4laXQkuW3lsOeLHfiA==$0 TABLE 2020-02-21:09:41:25

SV BIN$0gdjpgQYQvOcdV2YiJ1uVA==$0 TABLE 2020-02-21:09:25:14

SV1 BIN$lOcxalk8RJmK2UFxD13amg==$0 TABLE 2020-02-21:11:46:16

SV1 BIN$fNxbU/UCTKutTum4XjPNGg==$0 TABLE 2020-02-21:09:41:29

SV1 BIN$FvIPM1O0R/u94jNAYvCSPA==$0 TABLE 2020-02-21:09:38:54

SV1 BIN$DxWYsZ4MSn6m0mp07DQBug==$0 TABLE 2020-02-21:09:25:20

SV1 BIN$tTslk2kuQvSJuBJHXZwXng==$0 TABLE 2020-02-21:09:20:52

SQL> flashback table sample to before drop;

Flashback complete.

SQL> select \* from sample;

ID NAME

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

1 suku

2 veena

SQL> truncate table sample;

Table truncated.

SQL> select \* from sample;

no rows selected

SQL> flashback table sample to timestamp(systimestamp-interval '3' MINUTE);

flashback table sample to timestamp(systimestamp-interval '3' MINUTE)

\*

ERROR at line 1:

ORA-08189: cannot flashback the table because row movement is not enabled

SQL> select table\_name,row\_movement from dba\_tables where table\_name='sample';

select table\_name,row\_movement from dba\_tables where table\_name='sample'

\*

ERROR at line 1:

ORA-00942: table or view does not exist

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

syntax:- purge table/sequence/synonms ..etc objectname;

SQL> show recyclebin;

ORIGINAL NAME RECYCLEBIN NAME OBJECT TYPE DROP TIME

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

ROCK BIN$LoVtYFXZTLy0gESZQ0IjLQ==$0 TABLE 2020-02-21:18:11:28

SAMPLE1 BIN$JqlPw2auSIGvWh3hfo78ug==$0 TABLE 2020-02-22:09:34:22

SAMPLE1 BIN$JYrJkBa9TfiC8x6dSMcTqA==$0 TABLE 2020-02-21:18:11:00

SAMPLE2 BIN$lT1NtTsERC+gyGUmVHe1qw==$0 TABLE 2020-02-22:09:34:25

SAMPLE3 BIN$C4+JOKIpSQWir78f8TZctA==$0 TABLE 2020-02-22:09:35:16

SAMPLE41 BIN$tG2/4J7/SAijAxG9q71Y9A==$0 TABLE 2020-02-22:09:18:44

SUKUMAR BIN$b4LXfh3mSDS3fdiFObZ8JA==$0 TABLE 2020-02-21:11:46:22

SUKUMAR BIN$b88ZoDYSQWWdAD9Ab83kaw==$0 TABLE 2020-02-21:07:41:26

SUKUMAR BIN$ozh9q4sXTwGWw/ON254dZw==$0 TABLE 2020-02-21:07:25:24

SUKUMAR BIN$tutDFJqtSi6uI5LaAD2erQ==$0 TABLE 2020-02-21:07:24:02

SUKUMAR BIN$S7giN23rQ5ehoMy89QfAaQ==$0 TABLE 2020-02-20:19:13:12

SUKUMAR1 BIN$qZnn35SQRMWEBvPM2ZHbfA==$0 TABLE 2020-02-20:20:09:14

SV BIN$L8OI9Ab5QgWHKU054uiJ3w==$0 TABLE 2020-02-21:11:54:50

SV BIN$Z3i9kGaMTQGfn2mc3dVGOw==$0 TABLE 2020-02-21:11:46:11

SV BIN$tIyJHEY+S2qlRnqqKsjo8A==$0 TABLE 2020-02-21:10:03:00

SV BIN$xZ6x4laXQkuW3lsOeLHfiA==$0 TABLE 2020-02-21:09:41:25

SV BIN$0gdjpgQYQvOcdV2YiJ1uVA==$0 TABLE 2020-02-21:09:25:14

SV1 BIN$lOcxalk8RJmK2UFxD13amg==$0 TABLE 2020-02-21:11:46:16

SV1 BIN$fNxbU/UCTKutTum4XjPNGg==$0 TABLE 2020-02-21:09:41:29

SV1 BIN$FvIPM1O0R/u94jNAYvCSPA==$0 TABLE 2020-02-21:09:38:54

SV1 BIN$DxWYsZ4MSn6m0mp07DQBug==$0 TABLE 2020-02-21:09:25:20

SV1 BIN$tTslk2kuQvSJuBJHXZwXng==$0 TABLE 2020-02-21:09:20:52

SQL> purge table sample1;

Table purged.

SQL> show recyclebin;

ORIGINAL NAME RECYCLEBIN NAME OBJECT TYPE DROP TIME

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

ROCK BIN$LoVtYFXZTLy0gESZQ0IjLQ==$0 TABLE 2020-02-21:18:11:28

SAMPLE1 BIN$JqlPw2auSIGvWh3hfo78ug==$0 TABLE 2020-02-22:09:34:22

SAMPLE2 BIN$lT1NtTsERC+gyGUmVHe1qw==$0 TABLE 2020-02-22:09:34:25

SAMPLE3 BIN$C4+JOKIpSQWir78f8TZctA==$0 TABLE 2020-02-22:09:35:16

SAMPLE41 BIN$tG2/4J7/SAijAxG9q71Y9A==$0 TABLE 2020-02-22:09:18:44

SUKUMAR BIN$b4LXfh3mSDS3fdiFObZ8JA==$0 TABLE 2020-02-21:11:46:22

SUKUMAR BIN$b88ZoDYSQWWdAD9Ab83kaw==$0 TABLE 2020-02-21:07:41:26

SUKUMAR BIN$ozh9q4sXTwGWw/ON254dZw==$0 TABLE 2020-02-21:07:25:24

SUKUMAR BIN$tutDFJqtSi6uI5LaAD2erQ==$0 TABLE 2020-02-21:07:24:02

SUKUMAR BIN$S7giN23rQ5ehoMy89QfAaQ==$0 TABLE 2020-02-20:19:13:12

SUKUMAR1 BIN$qZnn35SQRMWEBvPM2ZHbfA==$0 TABLE 2020-02-20:20:09:14

SV BIN$L8OI9Ab5QgWHKU054uiJ3w==$0 TABLE 2020-02-21:11:54:50

SV BIN$Z3i9kGaMTQGfn2mc3dVGOw==$0 TABLE 2020-02-21:11:46:11

SV BIN$tIyJHEY+S2qlRnqqKsjo8A==$0 TABLE 2020-02-21:10:03:00

SV BIN$xZ6x4laXQkuW3lsOeLHfiA==$0 TABLE 2020-02-21:09:41:25

SV BIN$0gdjpgQYQvOcdV2YiJ1uVA==$0 TABLE 2020-02-21:09:25:14

SV1 BIN$lOcxalk8RJmK2UFxD13amg==$0 TABLE 2020-02-21:11:46:16

SV1 BIN$fNxbU/UCTKutTum4XjPNGg==$0 TABLE 2020-02-21:09:41:29

SV1 BIN$FvIPM1O0R/u94jNAYvCSPA==$0 TABLE 2020-02-21:09:38:54

SV1 BIN$DxWYsZ4MSn6m0mp07DQBug==$0 TABLE 2020-02-21:09:25:20

SV1 BIN$tTslk2kuQvSJuBJHXZwXng==$0 TABLE 2020-02-21:09:20:52

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

syntax:- alter table/synon/sequence/..object name [ add|modify|drop|rename|disable|enable|set unused ];

enable|disable are used to enable trigger and disable the triggers respectively.

SQL> desc sample;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> alter table sample add salary number(7,2);

Table altered.

SQL> desc sample

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SALARY NUMBER(7,2)

SQL> alter table sample modify salary number(9,3);

Table altered.

SQL> desc sample

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SALARY NUMBER(9,3)

SQL> alter table sample rename column salary to salary1;

Table altered.

SQL> desc sample;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SALARY1 NUMBER(9,3)

SQL> alter table sample rename to sample1;

Table altered.

SQL> desc sample

ERROR:

ORA-04043: object sample does not exist

SQL> desc sample1;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SALARY1 NUMBER(9,3)

SQL> alter table sample1 drop column salary1;

Table altered.

SQL> desc sample1;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> alter table sample add name varchar2(7);

Table altered.

SQL> insert into sample values(1,'suku');

1 row created.

SQL> insert into sample values(2,'suku');

1 row created.

SQL> alter table sample set unused column name;

Table altered.

SQL> desc sample;

Name Null? Type

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

ID NUMBER(7)

SQL> select \* from sample;

ID

----------

1

2

SQL> alter table sample drop unused columns;

Table altered.

SQL> alter table emp1 add constraint pk1 primary key(empno);

Table altered.

SQL> insert into emp1 values(1,'suku',7000);

1 row created.

SQL> insert into emp1 values(1,'suku',7000);

insert into emp1 values(1,'suku',7000)

\*

ERROR at line 1:

ORA-00001: unique constraint (SCOTT.PK1) violated

SQL> alter table emp1 drop constraint pk1;

Table altered.

SQL> insert into emp1 values(1,'suku',7000);

1 row created.

SQL> insert into emp1 values(1,'suku',7000);

1 row created.

SQL> alter table emp1 disable constraint pk1;

Table altered.

SQL> insert into emp1 values(1,'suku',7000);

1 row created.

SQL> insert into emp1 values(1,'suku',7000);

1 row created.

SQL> alter table emp1 enable constraint pk2;

Table altered.

**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.

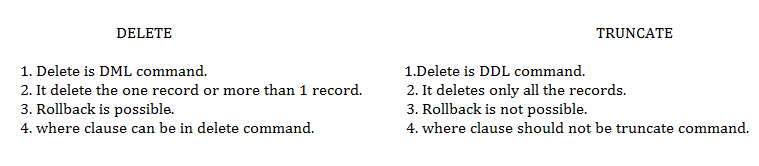
SQL> spool off;

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

**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.

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**:-

syntax:- char([size,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**:-

syntax:- varchar2([size,char])

-> 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:-**

syntax:- Nchar([size])

-> 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**:-

syntax:- varchar2([size])

-> 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**:-

syntax:- Number([Precision,scale])

-> 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 in decimal part of number.

Ex:- 338.7879

scale of above number is 4.

scale range is from -84 to +126.

-> Both arguments are optional.

-> DBMS allocates atmost 22 bytes.

SQL> create table sample(id number(7));

Table created.

Note :- If you enter value which has decimal part, number is rounded to nearest integer.

SQL> insert into sample values(4.7);

1 row created.

SQL> insert into sample values(7);

1 row created.

SQL> select \* from sample;

ID

----------

5

7

SQL> drop table sample;

Table dropped.

SQL> create table sample(id number(7,3));

Table created.

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.

SQL> insert into sample values(7);

1 row created.

SQL> insert into sample values(7.56);

1 row created.

SQL> insert into sample values(7.5989);

1 row created.

SQL> select \* from sample;

ID

----------

7

7.56

7.599

SQL> insert into sample values(8.00);

1 row created.

SQL> select \* from sample;

ID

----------

7

7.56

7.599

8

SQL> drop table sample;

Table dropped.

SQL> create table sample(id number(2,1));

Table created.

SQL> insert into sample values(32);

insert into sample values(32)

\*

ERROR at line 1:

ORA-01438: value larger than specified precision allowed for this column

SQL> insert into sample values(2);

1 row created.

SQL> insert into sample values(2.1);

1 row created.

SQL> select \* from sample;

ID

----------

2

2.1

SQL> spool off

**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.

Syntax: columnname 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.

SQL> prompt "Default format of DATE"

"Default format of DATE"

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

VALUE

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

DD-MON-RR

SQL> crate table rock(id date);

SP2-0734: unknown command beginning "crate tabl..." - rest of line ignored.

SQL> create table rock (id date);

Table created.

SQL> insert into rock values('06-nov-1981');

1 row created.

SQL> insert into rock values('06-11-1981');

insert into rock values('06-11-1981')

\*

ERROR at line 1:

ORA-01843: not a valid month

SQL> insert into rock values('nov-06-1981');

insert into rock values('nov-06-1981')

\*

ERROR at line 1:

ORA-01858: a non-numeric character was found where a numeric was expected

SQL> prompt "Change date format for current session"

"Change date format for current session"

SQL> alter session set NLS\_DATE\_FORMAT='yyyy-mon-dd 12HH:MI:SS';

ERROR:

ORA-01821: date format not recognized

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

Session altered.

SQL> insert into rock values('nov-06-1981');

1 row created.

SQL> insert into rock values('06-nov-1981');

insert into rock values('06-nov-1981')

\*

ERROR at line 1:

ORA-01858: a non-numeric character was found where a numeric was expected

SQL> select \* from rock;

ID

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

11-06-1981 12:00:00

11-06-1981 12:00:00

SQL> insert into rock values('06-nov-1981 09:12:34');

insert into rock values('06-nov-1981 09:12:34')

\*

ERROR at line 1:

ORA-01858: a non-numeric character was found where a numeric was expected

SQL> insert into rock values('nov-06-1981 09:12:34');

1 row created.

SQL> select \* from rock;

ID

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

11-06-1981 12:00:00

11-06-1981 12:00:00

11-06-1981 09:12:34

SQL> spool off

**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.

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.

syntax:-

column name timestamp([fracational seconds]);

-> 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> create table rock(id timestamp);

Table created.

SQL> insert into rock values('06-nov-1981');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37 AM');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37.1234 AM');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37.12345 AM');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37.123457 AM');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37.12345789 AM');

1 row created.

SQL> insert into rock values('06-nov-1981 7:07:37.1234578978 AM');

insert into rock values('06-nov-1981 7:07:37.1234578978 AM')

\*

ERROR at line 1:

ORA-01855: AM/A.M. or PM/P.M. required

SQL> select \* from rock;

ID

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

06-NOV-81 12.00.00.000000 AM

06-NOV-81 07.07.37.000000 AM

06-NOV-81 07.07.37.123400 AM

06-NOV-81 07.07.37.123450 AM

06-NOV-81 07.07.37.123457 AM

06-NOV-81 07.07.37.123458 AM

6 rows selected.

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';

Session altered.

SQL> select \* from rock;

ID

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

11-06-1981 12:00:00.000000 AM

11-06-1981 07:07:37.000000 AM

11-06-1981 07:07:37.123400 AM

11-06-1981 07:07:37.123450 AM

11-06-1981 07:07:37.123457 AM

11-06-1981 07:07:37.123458 AM

6 rows selected.

SQL> SPOOL OFF

**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**:- Syntax:- columnname 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.

SQL> create table sample(id BLOB);

Table created.

SQL> insert into sample values('123');

1 row created.

SQL> insert into sample values('ghij');

insert into sample values('ghij')

\*

ERROR at line 1:

ORA-01465: invalid hex number

SQL> insert into sample values('0100010');

1 row created.

SQL> insert into sample values(0100010);

insert into sample values(0100010)

\*

ERROR at line 1:

ORA-00932: inconsistent datatypes: expected BLOB got NUMBER

SQL> insert into sample values('AF012D');

1 row created.

SQL> select \* from sample;

ID

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

0123

00100010

AF012D

SQL> spool off

**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.

syntax:-

Column BFILE

-> 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.

syntax:

BFILENAME('dirname','filename.extension');

SQL> conn

Enter user-name: sys as sysdba

Connected.

SQL> prompt "1.Create a director";

"1.Create a director"

SQL> create directory rock as 'E:\rock';

Directory created.

SQL> grant read,write on directory rock to scott;

Grant succeeded.

SQL> create table sample1(photo BFILE);

Table created.

SQL> conn

Enter user-name: scott

Connected.

SQL> create table sample2(photo BFILE);

Table created.

SQL> insert into sample2 values(Bfilename('E:\rock','abc.jpg'));

1 row created.

SQL> select \* from sample2;

PHOTO

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

bfilename('E:\rock', 'abc.jpg')

SQL> spool off;

**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(-).

syntax:-

unaryoperator operand.

-> unary plus makes positive the given operand.

-> unary minus make negitive the given operand.

Example:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select -sal from emp;

-SAL

----------

-800

-1600

-1250

-2975

-1250

-2850

-2450

-3000

-5000

-1500

-1100

-950

-3000

-1300

14 rows selected.

SQL> select +sal from emp;

SAL

----------

800

1600

1250

2975

1250

2850

2450

3000

5000

1500

1100

950

3000

1300

14 rows selected.

SQL> select -sal from emp where sal > 3000;

-SAL

----------

-5000

SQL> select -sal from emp where sal>-sal ;

-SAL

----------

-800

-1600

-1250

-2975

-1250

-2850

-2450

-3000

-5000

-1500

-1100

-950

-3000

-1300

14 rows selected.

SQL> select -sal from emp where -sal>sal ;

no rows selected

SQL> spool off

**3.2)BINARY OPEATORS**:-

SYNTAX:- operand1 operator operand2

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:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select empno,sal+1000 as newsalary,deptno from emp;

EMPNO NEWSALARY DEPTNO

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

7369 1800 20

7499 2600 30

7521 2250 30

7566 3975 20

7654 2250 30

7698 3850 30

7782 3450 10

7788 4000 20

7839 6000 10

7844 2500 30

7876 2100 20

7900 1950 30

7902 4000 20

7934 2300 10

14 rows selected.

SQL> select empno,sal\*2 as newsalary,deptno from emp where deptno=20;

EMPNO NEWSALARY DEPTNO

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

7369 1600 20

7566 5950 20

7788 6000 20

7876 2200 20

7902 6000 20

SQL> select empno,sal\*3-1000 as newsalary,deptno from emp where deptno=20+10;

EMPNO NEWSALARY DEPTNO

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

7499 3800 30

7521 2750 30

7654 2750 30

7698 7550 30

7844 3500 30

7900 1850 30

6 rows selected.

SQL> select empno-sal as newsalary from emp where deptno=10;

NEWSALARY

----------

5332

2839

6634

SQL> select empno-sal as newsalary from emp where deptno=deptno-10;

no rows selected

SQL> spool off

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

The result of concatenation is character expression.

syntax:-

col-1|constant-1 || col-2|constant-2;

Example:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select ename||'-'||job as name from emp;

NAME

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

SMITH-CLERK

ALLEN-SALESMAN

WARD-SALESMAN

JONES-MANAGER

MARTIN-SALESMAN

BLAKE-MANAGER

CLARK-MANAGER

SCOTT-ANALYST

KING-PRESIDENT

TURNER-SALESMAN

ADAMS-CLERK

JAMES-CLERK

FORD-ANALYST

MILLER-CLERK

14 rows selected.

SQL> select empno||'-'||deptno from emp;

EMPNO||'-'||DEPTNO

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

7369-20

7499-30

7521-30

7566-20

7654-30

7698-30

7782-10

7788-20

7839-10

7844-30

7876-20

7900-30

7902-20

7934-10

14 rows selected.

SQL> select emp||'-'||ename from emp;

select emp||'-'||ename from emp

\*

ERROR at line 1:

ORA-00904: "EMP": invalid identifier

SQL> select empno||'-'||ename from emp;

EMPNO||'-'||ENAME

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

7369-SMITH

7499-ALLEN

7521-WARD

7566-JONES

7654-MARTIN

7698-BLAKE

7782-CLARK

7788-SCOTT

7839-KING

7844-TURNER

7876-ADAMS

7900-JAMES

7902-FORD

7934-MILLER

14 rows selected.

SQL> spool off;

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:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select \* from emp where empno>7500;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

12 rows selected.

SQL> select \* from emp where sal>1000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

12 rows selected.

SQL> select \* from emp where sal=1000;

no rows selected

SQL> select \* from emp where sal<>1000;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> spool off

4.LOGICAL OPERATORS:-

The logical opetators are

1)NOT 2) AND 3 )OR.

syntax: cond-1 operator cond-2;

examples:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select \* from emp where sal<3000 and deptno=30;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7900 JAMES CLERK 7698 03-DEC-81 950 30

6 rows selected.

SQL> select \* from emp where sal<3000 or deptno=30;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7934 MILLER CLERK 7782 23-JAN-82 1300 10

11 rows selected.

SQL> select \* from emp where (sal<3000 or deptno=30)and empno<7500;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

SQL> spool off;

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:-

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> select \* from emp where sal > any(2000,2500,3000);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7566 JONES MANAGER 7839 02-APR-81 2975 20

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7902 FORD ANALYST 7566 03-DEC-81 3000 20

6 rows selected.

SQL> select \* from emp where sal > all(2000,2500,3000);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7839 KING PRESIDENT 17-NOV-81 5000 10

SQL> select \* from emp where sal in(2000,2500,3000);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7902 FORD ANALYST 7566 03-DEC-81 3000 20

SQL> select \* from emp where sal between 1000 and 2500;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

7 rows selected.

SQL> select \* from emp where sal not between 1000 and 2500;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7566 JONES MANAGER 7839 02-APR-81 2975 20

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7 rows selected.

SQL> select \* from emp where sal between 800 and 1250;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

Note:- Between includes both 800 and 1250 value in above example.

SQL> select \* from emp where ename like 's%';

no rows selected

SQL> select \* from emp where ename like 'S%';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

SQL> select \* from emp where ename like '%S';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7566 JONES MANAGER 7839 02-APR-81 2975 20

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

SQL> select \* from emp where ename like '--S%';

no rows selected

SQL> select \* from emp where ename like '\_\_S%';

no rows selected

SQL> select \* from emp where ename like '\_\_A%';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

SQL> select \* from emp where ename like 'S\_A%';

no rows selected

SQL> select \* from emp where ename like 'B\_A%';

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

SQL> spool off

SQL> select \* from emp where exists(select \* from emp where sal=sal+2);

no rows selected

SQL> select \* from emp where exists(select \* from emp where sal=sal);

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

**3.3)SET OPERATORS**:- Go to DBMS Material.

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.

syntax:- columnname datatype default value;

SQL> create table sample(id number(7),m\_status varchar2(10) default 'unmarried');

Table created.

SQL> desc sample;

Name Null? Type

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

ID NUMBER(7)

M\_STATUS VARCHAR2(10)

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

1 row created.

SQL> insert into sample (id)values(8);

1 row created.

SQL> select \* from sample;

ID M\_STATUS

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

7 married

8 unmarried

SQL> alter table sample modify m\_status varchar2(10)default 'u\_marry';

Table altered.

SQL> insert into sample (id)values(8);

1 row created.

SQL> select \* from sample;

ID M\_STATUS

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

7 married

8 unmarried

8 u\_marry

SQL> alter table sample modify id number(7) default 77;

Table altered.

Note :- we can't remove existing default constraint.so we have to set default value as null.

**b)NOTNULL**:-

syntax:- columnname datatype [constraint name] not null;

->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**:-

syntax:-columnname datatype [constraint constraintname] check condition;

-> 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:-

syntax:- columnname datatype [constraint name] unique;

-> 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**:-

syntax:- columnname datatype [constraint name] 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:- Goto DMBS Material.**

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.

SQL> select \* from emp;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

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

INITCAP('THISIS

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

This Is Sukumar

SQL> select initcap('this.is.sukumar') from dual;

INITCAP('THIS.I

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

This.Is.Sukumar

SQL> select initcap(ename) from emp;

INITCAP(EN

----------

Smith

Allen

Ward

Jones

Martin

Blake

Clark

Scott

King

Turner

Adams

James

Ford

Miller

14 rows selected.

SQL> select initcap('this.is#sukumar')from dual;

INITCAP('THIS.I

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

This.Is#Sukumar

4) Length (column/string);

This function retruns 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.

EXAMPLE:-

SQL> select concat('sukumar','rajaka') from dual;

CONCAT('SUKUM

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

sukumarrajaka

SQL> select concat(concat('suku','mar'),'rajaka')from dual;

CONCAT(CONCAT

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

sukumarrajaka

SQL> select concat(ename,'works in some where') from emp;

CONCAT(ENAME,'WORKSINSOMEWHER

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

SMITHworks in some where

ALLENworks in some where

WARDworks in some where

JONESworks in some where

MARTINworks in some where

BLAKEworks in some where

CLARKworks in some where

SCOTTworks in some where

KINGworks in some where

TURNERworks in some where

ADAMSworks in some where

JAMESworks in some where

FORDworks in some where

MILLERworks in some where

14 rows selected.

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

6.1)LPAD(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 output.

-> When length of first argument is < second argument, the rest of place in output are filled with 3 rd argument to leftside of 1 st argument in output.

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

EXAMPLES:-

SQL> select lpad('this',7) from dual;

LPAD('T

-------

this

SQL> select lpad('this',8) from dual;

LPAD('TH

--------

this

SQL> select lpad('this',8,'\*') from dual;

LPAD('TH

--------

\*\*\*\*this

SQL> select rpad('this',8) from dual;

RPAD('TH

--------

this

SQL> select rpad('this',8,'\*') from dual;

RPAD('TH

--------

this\*\*\*\*

7. TRIMMING:- Trimming is removing unwanted/unnecessary characters from right most,left most or both sides of given string.

7.1) LTRIM(column name/string,string/chatacter)

7.2) RTRIM(column name/string,string/character)

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

Examples:-

SQL> select 'sukumar' from dual;

'SUKUMA

-------

sukumar

SQL> select ltrim(' sukumar') from dual;

LTRIM('

-------

sukumar

SQL> select ' sukumar' from dual;

'SUKUMAR'

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

sukumar

SQL> select ltrim(' sukumar','s') from dual;

LTRIM('SUKUMA

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

sukumar

SQL> select ltrim('sukumar','s') from dual;

LTRIM(

------

ukumar

SQL> select ltrim('sukumar','su') from dual;

LTRIM

-----

kumar

SQL> select ltrim('ssssukumar','s') from dual;

LTRIM(

------

ukumar

SQL> select ' sukumar ' from dual;

'SUKUMAR'

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

sukumar

SQL> select rtrim(' sukumar ') from dual;

RTRIM('SUKUM

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

sukumar

SQL> select rtrim(' sukumar') from dual;

RTRIM('SUKUM

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

sukumar

SQL> select rtrim(' sukumar','r') from dual;

RTRIM('SUKU

-----------

sukuma

SQL> select rtrim(' sukumar','ar') from dual;

RTRIM('SUK

----------

sukum

8. SUBSTR:-

Syntax: substr(col/string,m ,n)

-> 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:-

SQL> select substr('this is sukumar',5) from dual;

SUBSTR('THI

-----------

is sukumar

SQL> select substr('this is sukumar',8) from dual;

SUBSTR('

--------

sukumar

SQL> select substr('this is sukumar',8,4) from dual;

SUBS

----

suk

SQL> select substr('this is sukumar',8,0) from dual;

S

-

SQL> select substr('this is sukumar',8,-2) from dual;

S

-

SQL> select substr(name,-3) from person;

SUB

---

mar

ena

tha

tha

9.INSTRING:-

syntax:- INSTR(COLUMN/STRING,'C',m,n);

This function returns position of character, if it is found; otherwise it returns 0.

-> m is optional . m is either positive or negative.

if m is positive,seaching starts from leftmost side.

if m is negative,searching starts from rightmost side.

The default value is 1.

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

EXAMPLES:-

SQL> select instr('this is sukumar','u',1,3) from dual

2 ;

INSTR('THISISSUKUMAR','U',1,3)

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

0

SQL> select instr('this is sukumar','u',1,2) from dual;

INSTR('THISISSUKUMAR','U',1,2)

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

12

SQL> select instr('this is sukumar','u',1,1) from dual;

INSTR('THISISSUKUMAR','U',1,1)

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

10

SQL> select instr('this is sukumar','u',-2,1) from dual;

INSTR('THISISSUKUMAR','U',-2,1)

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

12

SQL> select instr('this is sukumar','u',0,1) from dual;

INSTR('THISISSUKUMAR','U',0,1)

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

0

SQL> select instr('this is sukumar','u',1,1) from dual;

INSTR('THISISSUKUMAR','U',1,1)

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

10

10.REVERSE(colnam/string)

it returns string in reverse.

11.REPLACE:-

syntax:- Replace(text,search-string,[replacement-string])

-> 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:-

syntax:- Translate(co;/string,string1,string2)

-> The no.of characters in string1 must equal to no.of characters in string2.

-> if no.of characters in string1 is < no.of chatacters in string 2 then extra character will be ignored.

-> if no.of characters in string1 is > no.of characters in string 2 ,then string replacement is done and character by character replacement is also done.

EXAMPLES:-

SQL> select translate('this is sukumar','s','v') from dual;

TRANSLATE('THIS

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

thiv iv vukumar

SQL> select translate('this is sukumar','su','ve') from dual;

TRANSLATE('THIS

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

thiv iv vekemar

SQL> select translate('this is sukumar','su','vex') from dual;

TRANSLATE('THIS

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

thiv iv vekemar

SQL> select translate('this is sukumar','su','v') from dual;

TRANSLATE('TH

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

thiv iv vkmar

**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) AGGREGATE FUNCTIONS/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

SQL> select count(empno) as Tot-Emp,avg(sal) as average-sal,sum(sal)as toal-sal ,max(sal),min(sal) from emp;

select count(empno) as Tot-Emp,avg(sal) as average-sal,sum(sal)as toal-sal ,max(sal),min(sal) from emp

\*

ERROR at line 1:

ORA-00923: FROM keyword not found where expected

SQL> select count(empno) as TotEmp,avg(sal) as averagesal,sum(sal)as toal-sal ,max(sal),min(sal) from emp;

select count(empno) as TotEmp,avg(sal) as averagesal,sum(sal)as toal-sal ,max(sal),min(sal) from emp

\*

ERROR at line 1:

ORA-00923: FROM keyword not found where expected

SQL> select count(empno) as TotEmp,avg(sal) as averagesal,sum(sal)as toalsal ,max(sal),min(sal) from emp;

TOTEMP AVERAGESAL TOALSAL MAX(SAL) MIN(SAL)

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

14 2073.21429 29025 5000 800

**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.

EXAMPLE:-

SQL> select trunc(10) from dual;

TRUNC(10)

----------

10

SQL> select trunc(10.3) from dual;

TRUNC(10.3)

-----------

10

SQL> select trunc(10.333,1)from dual;

TRUNC(10.333,1)

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

10.3

SQL> select trunc(123.333,-1)from dual;

TRUNC(123.333,-1)

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

120

SQL> select trunc(123.333,-2)from dual;

TRUNC(123.333,-2)

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

100

SQL> select trunc(123.333,-3)from dual;

TRUNC(123.333,-3)

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

0

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

syntax:- Trunc(date,[YY|MM|DD|HR|MI|SS])

note :- The first argument should be date data type. if you need ,use to\_date function to convert the string into data type.

-> when date is truncated based on year, we get starting date of year.

SQL> select to\_char(trunc('11-nov-dec','yy')) from dual;

select to\_char(trunc('11-nov-dec','yy')) from dual

\*

ERROR at line 1:

ORA-01722: invalid number

SQL> select trunc(to\_date('11-nov-2020'),'yy') from dual;

TRUNC(TO\_

---------

01-JAN-20

-> when date is truncated based on year, we get starting date of month.

SQL> select trunc(to\_date('11-nov-2020'),'mm') from dual;

TRUNC(TO\_

---------

01-NOV-20

-> 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.

SQL> select trunc(sysdate,'hh') from dual;

TRUNC(SYSDATE,'HH

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

25-02-20 06:00:00

-> when date is truncated based on minutes,In time, only seconds becomes zero.

SQL> select trunc(sysdate,'mi') from dual;

TRUNC(SYSDATE,'MI

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

25-02-20 06:15:00

-> we can't truncate seconds.

2. ROUND:-

syntax:- Round(m,n)

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 unitplace,tensplace,....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.

syntax:- round(date,[yy|mm|dd|HH|MI|SS]);

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.

3.TO\_DATE:-

1

TO\_DATE (string, format, nls\_language)

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:

1

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.

SQL> select to\_date('06111981','dd-mm-yy') from dual;

TO\_DATE('

---------

06-NOV-81

SQL> select to\_date('06 11 1981','dd mm yy') from dual;

TO\_DATE('

---------

06-NOV-81

SQL> select to\_date('06 11 1981','dd-mm-yy') from dual;

TO\_DATE('

---------

06-NOV-81

SQL> select to\_date('06 nov 1981','dd-mm-yy') from dual;

TO\_DATE('

---------

06-NOV-81

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.

syntax:- to\_char(date/num,[D|DD|DDD|yy|YYYY|HH|MI|SS]or[DAY|DY|MONTH|MON|YEAR])

Returns Number As output:

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

D-day of week.

DD-day of month.

DDD-day of year.

YY,YYYY,HH,MI,SS.

Returns Character As output:

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

DAY

DY

MONTH

MON

YEAR.

Examples:-

SQL> select to\_char(sysdate,'D') from dual;

T

-

6

SQL> select to\_char(sysdate,'DD') from dual;

TO

--

22

SQL> select to\_char(sysdate,'DDD') from dual;

TO\_

---

112

SQL> select to\_char(sysdate,'DAY') from dual;

TO\_CHAR(S

---------

FRIDAY

SQL> select to\_char(sysdate,'DY')from dual;

TO\_

---

FRI

SQL> select to\_char(sysdate,'YY')from dual;

TO

--

22

SQL> select to\_char(sysdate,'YYYY')from dual;

TO\_C

----

2022

SQL> select to\_char(sysdate,'MON')from dual;

TO\_

---

APR

SQL> select to\_char(sysdate,'MONTH')from dual;

TO\_CHAR(S

---------

APRIL

5.2.5) 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**: Goto DBMS Material.

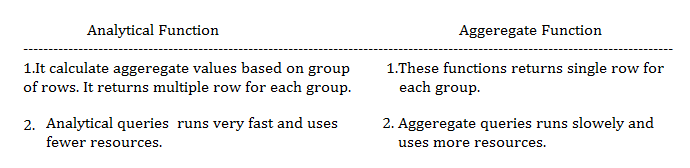
**7.SUB QUERIES**:- Go to DBMS material.

**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])

Example:-

SQL> select empno,ename,sal,rank()over(order by sal asc) from emp;

EMPNO ENAME SAL RANK()OVER(ORDERBYSALASC)

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

7369 SMITH 800 1

7900 JAMES 950 2

7876 ADAMS 1100 3

7521 WARD 1250 4

7654 MARTIN 1250 4

7844 TURNER 1500 6

7499 ALLEN 1600 7

7782 CLARK 2450 8

7698 BLAKE 2850 9

7566 JONES 2975 10

7902 FORD 3000 11

7788 SCOTT 3000 11

7839 KING 5000 13

13 rows selected.

SQL> select empno,ename,deptno,sal,rank()over(partition by deptno order by sal asc) from emp;

EMPNO ENAME DEPTNO SAL RANK()OVER(PARTITIONBYDEPTNOORDERBYSALASC)

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

7782 CLARK 10 2450 1

7839 KING 10 5000 2

7369 SMITH 20 800 1

7876 ADAMS 20 1100 2

7566 JONES 20 2975 3

7902 FORD 20 3000 4

7788 SCOTT 20 3000 4

7900 JAMES 30 950 1

7521 WARD 30 1250 2

7654 MARTIN 30 1250 2

7844 TURNER 30 1500 4

7499 ALLEN 30 1600 5

7698 BLAKE 30 2850 6

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

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

values dense\_rank rank

5000 1 1

4000 2 2

3000 3 3

3000 3 3

3000 3 3

2000 6 4

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

**3) 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.

syntax:- row\_number()over([partition clause ]order by clause);

examples:-

SQL> select empno,ename,deptno,sal,row\_number()over(order by sal asc) from emp;

EMPNO ENAME DEPTNO SAL ROW\_NUMBER()OVER(ORDERBYSALASC)

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

7369 SMITH 20 800 1

7900 JAMES 30 950 2

7876 ADAMS 20 1100 3

7521 WARD 30 1250 4

7654 MARTIN 30 1250 5

7844 TURNER 30 1500 6

7499 ALLEN 30 1600 7

7782 CLARK 10 2450 8

7698 BLAKE 30 2850 9

7566 JONES 20 2975 10

7902 FORD 20 3000 11

7788 SCOTT 20 3000 12

7839 KING 10 5000 13

13 rows selected.

SQL> select empno,ename,deptno,sal,row\_number()over(partition by deptno order by sal asc) from emp;

EMPNO ENAME DEPTNO SAL ROW\_NUMBER()OVER(PARTITIONBYDEPTNOORDERBYSALASC)

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

7782 CLARK 10 2450 1

7839 KING 10 5000 2

7369 SMITH 20 800 1

7876 ADAMS 20 1100 2

7566 JONES 20 2975 3

7902 FORD 20 3000 4

7788 SCOTT 20 3000 5

7900 JAMES 30 950 1

7521 WARD 30 1250 2

7654 MARTIN 30 1250 3

7844 TURNER 30 1500 4

7499 ALLEN 30 1600 5

7698 BLAKE 30 2850 6

13 rows selected.

**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.

syntax:- first\_value(colname)over([partition clause order by claue]);

SQL> select empno,ename,deptno,sal,first\_value(sal)over() from emp;

EMPNO ENAME DEPTNO SAL FIRST\_VALUE(SAL)OVER()

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

7369 SMITH 20 800 800

7499 ALLEN 30 1600 800

7521 WARD 30 1250 800

7566 JONES 20 2975 800

7654 MARTIN 30 1250 800

7698 BLAKE 30 2850 800

7782 CLARK 10 2450 800

7788 SCOTT 20 3000 800

7839 KING 10 5000 800

7844 TURNER 30 1500 800

7876 ADAMS 20 1100 800

7900 JAMES 30 950 800

7902 FORD 20 3000 800

13 rows selected.

SQL> select empno,ename,deptno,sal,first\_value(sal)over(order by sal desc) from emp;

EMPNO ENAME DEPTNO SAL FIRST\_VALUE(SAL)OVER(ORDERBYSALDESC)

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

7839 KING 10 5000 5000

7788 SCOTT 20 3000 5000

7902 FORD 20 3000 5000

7566 JONES 20 2975 5000

7698 BLAKE 30 2850 5000

7782 CLARK 10 2450 5000

7499 ALLEN 30 1600 5000

7844 TURNER 30 1500 5000

7521 WARD 30 1250 5000

7654 MARTIN 30 1250 5000

7876 ADAMS 20 1100 5000

7900 JAMES 30 950 5000

7369 SMITH 20 800 5000

13 rows selected.

SQL> select empno,ename,deptno,sal,first\_value(sal)over(partition by deptno order by sal desc) from emp;

EMPNO ENAME DEPTNO SAL FIRST\_VALUE(SAL)OVER(PARTITIONBYDEPTNOORDERBYSALDESC)

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

7839 KING 10 5000 5000

7782 CLARK 10 2450 5000

7788 SCOTT 20 3000 3000

7902 FORD 20 3000 3000

7566 JONES 20 2975 3000

7876 ADAMS 20 1100 3000

7369 SMITH 20 800 3000

7698 BLAKE 30 2850 2850

7499 ALLEN 30 1600 2850

7844 TURNER 30 1500 2850

7521 WARD 30 1250 2850

7654 MARTIN 30 1250 2850

7900 JAMES 30 950 2850

13 rows selected.

SQL> select empno,ename,deptno,sal,first\_value(mgr)ignore nulls over(partition by deptno order by sal desc) from emp;

EMPNO ENAME DEPTNO SAL FIRST\_VALUE(MGR)IGNORENULLSOVER(PARTITIONBYDEPTNOORDERBYSALDESC)

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

7839 KING 10 5000

7782 CLARK 10 2450 7839

7788 SCOTT 20 3000 7566

7902 FORD 20 3000 7566

7566 JONES 20 2975 7566

7876 ADAMS 20 1100 7566

7369 SMITH 20 800 7566

7698 BLAKE 30 2850 7839

7499 ALLEN 30 1600 7839

7844 TURNER 30 1500 7839

7521 WARD 30 1250 7839

7654 MARTIN 30 1250 7839

7900 JAMES 30 950 7839

13 rows selected.

9)VIEW:- Goto DBMS Material and durgasoft material.

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 obly be created for entire object.(i.e) we can create synony 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.

SQL> desc user\_synonyms;

Name Null? Type

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

SYNONYM\_NAME NOT NULL VARCHAR2(30)

TABLE\_OWNER VARCHAR2(30)

TABLE\_NAME NOT NULL VARCHAR2(30)

DB\_LINK

**10.1)PRIVATE SYNONYM**:-

syntax:- create or replace synonym synonymname for object;

Example:-

SQL> create or replace synonym abc1 for emp;

Synonym created.

SQL> select \* from user\_synonyms;

SYNONYM\_NAME TABLE\_OWNER TABLE\_NAME

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

DB\_LINK

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

ABC2 SCOTT EMP

SQL> select \* from abc1;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1300 10

14 rows selected.

SQL> update abc1 set sal=1500 where empno=7934;

1 row updated.

SQL> select \* from abc1;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

7934 MILLER CLERK 7782 23-JAN-82 1500 10

14 rows selected.

SQL> delete from abc1 where empno=7934;

1 row deleted.

SQL> select \* from abc1;

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7369 SMITH CLERK 7902 17-DEC-80 800 20

7499 ALLEN SALESMAN 7698 20-FEB-81 1600 300 30

7521 WARD SALESMAN 7698 22-FEB-81 1250 500 30

7566 JONES MANAGER 7839 02-APR-81 2975 20

7654 MARTIN SALESMAN 7698 28-SEP-81 1250 1400 30

7698 BLAKE MANAGER 7839 01-MAY-81 2850 30

7782 CLARK MANAGER 7839 09-JUN-81 2450 10

7788 SCOTT ANALYST 7566 19-APR-87 3000 20

7839 KING PRESIDENT 17-NOV-81 5000 10

7844 TURNER SALESMAN 7698 08-SEP-81 1500 0 30

7876 ADAMS CLERK 7788 23-MAY-87 1100 20

EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO

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

7900 JAMES CLERK 7698 03-DEC-81 950 30

7902 FORD ANALYST 7566 03-DEC-81 3000 20

13 rows selected.

SQL> rename abc1 to abc2;

Table renamed.

SQL>drop synonym abc1;

Table dropeed

NOTE:- 1.we can't add and drop column to base object through synonym.

2. The private synonym can only be access with in current 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.

syntax:- create or replace public synonym name for objectanme;

SQL> create public synonym abc4 for emp;

Synonym created.

**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.

Sytax:- creat sequence sequence\_name[incremented by value1 min\_value value2 max\_value value3 cache value4| no cache cycle|nocycle order|noorder];

->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 sequenceand dbms add tis value to max\_value.

-> start\_with: Default value is MIN\_Value. This value can be altered. sequence should starts from this number for first cycle.Rest of cycle starts from min\_value.

-> Min\_value:- 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.

-> MAX\_VALUE:- default value is bignumber(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.

EXAMPLE:-

SQL> desc user\_sequences;

Name Null? Type

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

SEQUENCE\_NAME NOT NULL VARCHAR2(30)

MIN\_VALUE NUMBER

MAX\_VALUE NUMBER

INCREMENT\_BY NOT NULL NUMBER

CYCLE\_FLAG VARCHAR2(1)

ORDER\_FLAG VARCHAR2(1)

CACHE\_SIZE NOT NULL NUMBER

LAST\_NUMBER NOT NULL NUMBER

SQL> select \* from user\_sequences;

no rows selected

SQL> create sequence abc1;

Sequence created.

SQL> set linesize 100

SQL> select \* from user\_sequences;

SEQUENCE\_NAME MIN\_VALUE MAX\_VALUE INCREMENT\_BY C O CACHE\_SIZE LAST\_NUMBER

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

ABC1 1 1.0000E+28 1 N N 20 1

SQL> select abc1.currval from dual;

select abc1.currval from dual

\*

ERROR at line 1:

ORA-08002: sequence ABC1.CURRVAL is not yet defined in this session

SQL> select abc1.nextval from dual;

NEXTVAL

----------

1

SQL> select abc1.nextval from dual;

NEXTVAL

----------

2

SQL> desc abc1

SP2-0381: DESCRIBE sequence is not available

SQL> select abc1.currval from dual;

CURRVAL

----------

2

SQL> create sequence abc2 increment by 2 minvalue 7 maxvalue 14 cycle cache 4;

Sequence created.

SQL> select abc2.nextval from dual;

NEXTVAL

----------

7

SQL>

SQL> select abc2.nextval from dual;

NEXTVAL

----------

9

SQL> select abc2.nextval from dual;

NEXTVAL

----------

11

SQL> select abc2.nextval from dual;

NEXTVAL

----------

13

SQL> select abc2.nextval from dual;

NEXTVAL

----------

7

SQL> create sequence abc3 increment by -2 minvalue -10 maxvalue -1 ;

Sequence created.

SQL> select abc3.nextval from dual;

NEXTVAL

----------

-1

SQL> select abc3.nextval from dual;

NEXTVAL

----------

-3

SQL> select abc3.nextval from dual;

NEXTVAL

----------

-5

SQL> select abc3.nextval from dual;

NEXTVAL

----------

-7

SQL> select abc3.nextval from dual;

NEXTVAL

----------

-9

SQL> select abc3.nextval from dual;

select abc3.nextval from dual

\*

ERROR at line 1:

ORA-08004: sequence ABC3.NEXTVAL goes below MINVALUE and cannot be instantiated

-------------------DDL on SEQUENCE--------------------------------

SQL> drop sequence abc1;

Sequence dropped.

ALTER:-

syntax:-

alter sequences sequencename

[increment by n

maxvalue n|no maxvalue

minvalue n|no minvalue

cycle | nocycle

cache n | nocache]

SQL> select \* from user\_sequences;

SEQUENCE\_NAME MIN\_VALUE MAX\_VALUE INCREMENT\_BY C O CACHE\_SIZE LAST\_NUMBER

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

ABC2 7 14 2 Y N 4 15

ABC3 -10 -1 -2 N N 20 -11

SQL> run

1 alter sequence abc2

2 maxvalue 16

3\* cache 3

Sequence altered.

SQL> select \* from user\_sequences;

SEQUENCE\_NAME MIN\_VALUE MAX\_VALUE INCREMENT\_BY C O CACHE\_SIZE LAST\_NUMBER

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

ABC2 7 16 2 Y N 3 9

ABC3 -10 -1 -2 N N 20 -11

SQL> rename abc2 to abc4;

Table renamed.

SQL> select \* from user\_sequences;

SEQUENCE\_NAME MIN\_VALUE MAX\_VALUE INCREMENT\_BY C O CACHE\_SIZE LAST\_NUMBER

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

ABC3 -10 -1 -2 N N 20 -11

ABC4

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

12.1**) Normal Table.**

**12.2) GTT:-**

-> Temporary table is also like ordernary table.The temporary table definition is shared by 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.

syntax:- CREATE GLOBAL TEMPORARY TABLE table\_name

( column\_name column\_data\_type

...

...

) ON COMMIT {DELETE | PRESERVE } ROWS ;

--------------------------------------------------------------SESSION-1--------------------------------------------------------- ;

SQL> CREATE GLOBAL TEMPORARY TABLE

2 SUKUMAR(ID NUMBER(7));

Table created.

SQL> INSERT INTO SUKUMAR VALUES(1);

1 row created.

SQL> INSERT INTO SUKUMAR VALUES(2);

1 row created.

SQL> SELECT \* FROM SUKUMAR;

ID

----------

1

2

SQL> SPOOL OFF

--------------------------------------------------------------SESSION-2--------------------------------------------------------- ;

SQL> INSERT INTO SUKUMAR VALUES(77);

1 row created.

SQL> INSERT INTO SUKUMAR VALUES(87);

1 row created.

SQL> SELECT \* FROM SUKUMAR;

ID

----------

77

87

SQL> SPOOL OFF

SQL>--------------------------------------------------------------------------------------------------------------------------------;

SQL> CREATE GLOBAL TEMPORARY TABLE ROCK1(ID NUMBER(7)) ON COMMIT DELETE ROWS;

Table created.

SQL> INSERT INTO ROCK1 VALUES(8);

1 row created.

SQL> INSERT INTO ROCK1 VALUES(9);

1 row created.

SQL> INSERT INTO ROCK1 VALUES(10);

1 row created.

SQL> SELECT \* FROM ROCK1;

ID

----------

8

9

10

SQL> COMMIT;

Commit complete.

SQL> SELECT \* FROM ROCK1;

no rows selected

SQL> SPOOL OFF

SQL> CREATE GLOBAL TEMPORARY TABLE ROCK2 (ID NUMBER(7) ) ON COMMIT PRESERVE ROWS;

Table created.

SQL> INSERT INTO ROCK1 VALUES(3);

1 row created.

SQL> INSERT INTO ROCK1 VALUES(4);

1 row created.

SQL> COMMIT;

Commit complete.

SQL> SELECT \* FROM ROCK1;

no rows selected

SQL> SPOOL OFF

SQL > EXIT;

SQL>------------------------------------------------------------------------SESSEION CLOSED & REOPENED --------------------------------------

SQL> SELECT \* FROM ROCK1;

NOROWS SELECTED;

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 badrecords 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.

For 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.

syntax:- 1.create table tablename(column definitions) Partition by list(Partition key)

2.(partion name1 values(...)[tablespace name],partion name2 values(...)[tablespace name]);

Example:-

SQL> create table sukumar (id number(7))partition by list(id)(partition one values(1,2,3),partition two values(4,5,6));

Table created.

SQL> insert into sukumar values(1);

1 row created.

SQL> insert into sukumar values(2);

1 row created.

SQL> insert into sukumar values(5);

1 row created.

SQL> insert into sukumar values(6);

1 row created.

SQL> select \* from sukumar partition(one);

ID

----------

1

2

SQL> select \* from sukumar partition(two);

ID

----------

5

6

SQL> select \* from sukumar;

ID

----------

1

2

5

6

SQL> spool off;

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').

syntax:- 1.create table tablename(column specifications)partition by range(partition key)

2.(partition name values less than/greather than/equal to (....) [tablespace name],

partition name values less than/greather than/equal to(...)[tablespace name]);

example:-

1 create table sukumar(id number(7))partition by range(id)

2 (partition one values less than(10),

3\* partition two values less than(20))

SQL> /

Table created.

SQL> insert into sukumar values(9);

1 row created.

SQL> insert into sukumar values(12);

1 row created.

SQL> insert into sukumar values(13);

1 row created.

SQL> insert into sukumar values(7);

1 row created.

SQL> select \* from sukumar;

ID

----------

9

7

12

13

SQL> select \* from sukumar partition(one);

ID

----------

9

7

SQL> select \* from sukumar partition(two);

ID

----------

12

13

SQL> spoll off

SP2-0042: unknown command "spoll off" - rest of line ignored.

SQL> spool off;

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.

syntax:- 1.create table tablename(column definition) partition by hash(partition key)

2.{[partitions number]| [(partition name1 [tablespace name],partition name2 [tablespace name]);]}[store in(tablespace1,tablespace2,....tablespacen);]

Example:-

SQL> ed

Wrote file afiedt.buf

1 create table sukumar1(id number(7),name varchar2(8))partition by hash(id)

2 (partition one,

3 partition two,

4\* partition three)

SQL> /

Table created.

SQL> insert into sukumar1 values(1,'suku');

1 row created.

SQL> insert into sukumar1 values(2,'veena');

1 row created.

SQL> insert into sukumar1 values(3,'sula');

1 row created.

SQL> insert into sukumar1 values(4,'sumaha');

1 row created.

SQL> select \* from sukumar1;

ID NAME

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

1 suku

3 sula

4 sumaha

2 veena

SQL> select \* from sukumar1 partition(one);

no rows selected

SQL> select \* from sukumar1 partition(two);

ID NAME

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

1 suku

3 sula

4 sumaha

SQL> select \* from sukumar1 partition(three);

ID NAME

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

2 veena

SQL> spool off;

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:-

syntax:-

CREATE TABLE [ schema. ]table\_name

table\_definition

PARTITION BY RANGE(column[, column ]...)

[SUBPARTITION BY {RANGE|LIST|HASH} (column[, column ]...)]

[SUBPARTITION TEMPLATE(subpartition definition)..]

(range\_partition\_definition[, range\_partition\_definition]...);

Where range\_partition\_definition is:

PARTITION [partition\_name]

VALUES LESS THAN (value[, value]...)

[TABLESPACE tablespace\_name]

subpartition may be one of the following:

{list\_subpartition | range\_subpartition | hash\_subpartition}

where list\_subpartition is:

SUBPARTITION [subpartition\_name]

VALUES (value[, value]...)

[TABLESPACE tablespace\_name]

where range\_subpartition is:

SUBPARTITION [subpartition\_name]

VALUES LESS THAN (value[, value]...)

[TABLESPACE tablespace\_name]

where hash\_subpartition is:

[SUBPARTITION subpartition\_name]

[TABLESPACE tablespace\_name]

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

DDL ON PARTITIONS

1. Maintenance operations on LIST PARTTITION:-

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

SQL> ed

Wrote file afiedt.buf

1 create table sv(id number(7),name varchar2(8))partition by list(name)

2\* (partition p1 values('suku','veena'))

SQL> /

Table created.

SQL> prompt "================ADD NEW PARTITION======================="

"================ADD NEW PARTITION======================="

SQL> alter table sv add partition p2 values('sula','suma','suha');

Table altered.

SQL> insert into sv values(1,'suku');

1 row created.

SQL> insert into sv values(2,'sula');

1 row created.

SQL> insert into sv values(3,'suma');

1 row created.

SQL> select \* from sv partition (p1);

ID NAME

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

1 suku

SQL> select \* from sv partition (p2);

ID NAME

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

2 sula

3 suma

SQL> prompt"=============================TRUNCATE========================================"

"=============================TRUNCATE========================================"

SQL> alter table sv truncate partition p2;

Table truncated.

SQL> select \* from sv partition(p2);

no rows selected

SQL> prompt"==============================DROP==============================================="

"==============================DROP==============================================="

SQL> alter table sv drop partition p2;

Table altered.

SQL> alter table sv split partition p1 values ('suku') into (partition p11,partition p12);

Table altered.

SQL> insert into sv values(8,'suku');

1 row created.

SQL> insert into sv values(9,'suku');

1 row created.

SQL> insert into sv values(10,'suku');

1 row created.

SQL> insert into sv values(10,'veena');

1 row created.

SQL> insert into sv values(11,'veena');

1 row created.

SQL> insert into sv values(12,'veena');

1 row created.

SQL> select \* from sv partition (p11);

ID NAME

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

1 suku

8 suku

9 suku

10 suku

SQL> select \* from sv partition (p12);

ID NAME

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

10 veena

11 veena

12 veena

SQL> prompt =================================EXCHANGE DATA IN PARTITION========================================

=================================EXCHANGE DATA IN PARTITION========================================

SQL> create table sv1(id number(7),name varchar2(7));

Table created.

SQL> create table sv1(id number(7),name varchar2(8));

Table created.

SQL> alter table sv exchange partition p12 with table sv1;

Table altered.

SQL> select \* from sv1;

ID NAME

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

10 veena

11 veena

12 veena

SQL> select \* from sv partition (p12);

no rows selected

SQL> spool off

2.Maintenance operation on RANGE PARTITION:-

SQL> create table sv(id number(7),name varchar2(7)) partition by range(id) (partition p1 values less than(5),

2 partition p2 values less than(10));

Table created.

SQL> prompt "==========================================ADD PARTITION===================================="

"==========================================ADD PARTITION===================================="

SQL> alter table sv add partition p3 values less than(15);

Table altered.

SQL> insert into sv values(1,'suku');

1 row created.

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

1 row created.

SQL> insert into sv values(11,'veena');

1 row created.

SQL> insert into sv values(7,'veena');

1 row created.

SQL> select \* from sv;

ID NAME

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

1 suku

2 veena

7 veena

11 veena

SQL> select \* from sv partition(p3);

ID NAME

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

11 veena

SQL> alter table sv modify partition p3 add values(4);

Table altered.

SQL> alter table sv modify partition p3 drop values(4);

Table altered.

SQL> prompt =============================================TRUNCATE============================================

=============================================TRUNCATE============================================

SQL> alter table sv truncate partition (p3);

Table truncated.

SQL> select \* from sv partition (p3);

no rows selected

SQL> prompt ========================================DROP PARTITION===========================

========================================DROP PARTITION===========================

SQL> alter table sv drop partition(p3);

Table altered.

SQL> select \* from sv partition(p3);

select \* from sv partition(p3)

\*

ERROR at line 1:

ORA-02149: Specified partition does not exist

SQL> prompt =======================================SPLIT PARTITION ==================================

=======================================SPLIT PARTITION ==================================

SQL> alter table sv split partition p1 at(3) into (partition p11,partition p12);

Table altered.

SQL> insert into sv values('4','rock');

1 row created.

SQL> insert into sv values('2','hhh');

1 row created.

SQL> select \* from sv partition(p11);

ID NAME

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

1 suku

2 veena

2 hhh

SQL> select \* from sv partition(p12);

ID NAME

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

4 rock

SQL> prompt ========================================EXCHANGE ======================================

========================================EXCHANGE ======================================

SQL> create table sv1(id number(7),name varchar2(8));

Table created.

SQL> alter table sv exchange partition p11 with table sv1;

alter table sv exchange partition p11 with table sv1

\*

ERROR at line 1:

ORA-14097: column type or size mismatch in ALTER TABLE EXCHANGE PARTITION

SQL> desc sv;

Name Null? Type

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

ID NUMBER(7)

NAME VARCHAR2(7)

SQL> drop table sv1;

Table dropped.

SQL> create table sv1(id number(7),name varchar2(7));

Table created.

SQL> alter table sv exchange partition p11 with table sv1;

Table altered.

SQL> select \* from sv1;

ID NAME

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

1 suku

2 veena

2 hhh

SQL> spool off;

===========================================================================COMPOSITE PARTITIONS==========================================================

EXample:- List -Range partition

SQL> create table sv(id number(7),name varchar2(7))

2 partition by list (name)

3 subpartition by range(id)

4 subpartition template(

5 subpartition one values less than(5),

6 subpartition two values less than(10))

7 (partition p1 values('suku','veena'),

8 partition p2 values('sula','suma','suha'));

Table created.

SQL> insert into sv values(3,'suku');

1 row created.

SQL> insert into sv values(5,'veena');

1 row created.

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

1 row created.

SQL> insert into sv values(6,'sula');

1 row created.

SQL> insert into sv values(6,'suha');

1 row created.

SQL> select \* from sv;

ID NAME

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

3 suku

5 veena

7 sula

6 sula

6 suha

SQL> alter table sv modify partition p1 add subpartition three values less than (20);

Table altered.

SQL> alter table sv drop subpartition three;

Table altered.

SQL> create table sv1(id number(7),name varchar2(7));

Table created.

SQL> ALTER TABLE sv MODIFY SUBPARTITION one ADD VALUES (30);

Table altered.

SQL> spool off;

**13)USER DEFINED DATATYPE**:-

-> user defined data type(UDT) object(OR) Object type is data base object.

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

-> This Object type is also called as collection in oracle. It is REUSABLE.

Syntax:-

CREATE OR REPLACE TYPE NAME

AS OBJECT

(COLS DEFINITIONS);/

Examples:-

SQL> create type area\_type as object

2 (street\_name varchar2(10),landmark varchar2(10));

3 /

Type created.

SQL> create type address as object

2 (hno number(10),area area\_type,city varchar2(10));

3 /

Type created.

SQL> set describe depth 3 linenum off indent on;

SQL> desc area\_type

Name Null? Type

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

STREET\_NAME VARCHAR2(10)

LANDMARK VARCHAR2(10)

SQL> desc address

Name Null? Type

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

HNO NUMBER(10)

AREA AREA\_TYPE

STREET\_NAME VARCHAR2(10)

LANDMARK VARCHAR2(10)

CITY VARCHAR2(10)

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

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.

Examples:-

SQL> create table student(no number(2),name varchar2(10),addr address);

Table created.

SQL> desc student;

Name Null? Type

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

NO NUMBER(2)

NAME VARCHAR2(10)

ADDR ADDRESS

HNO NUMBER(10)

AREA AREA\_TYPE

STREET\_NAME VARCHAR2(10)

LANDMARK VARCHAR2(10)

CITY VARCHAR2(10)

SQL> INSERT INTO STUDENT(NO,NAME,ADDR)VALUES(7,'SUKU',ADDRESS(77,AREA\_TYPE('RAJAKA','PETROLB'),'NLR'));

1 row created.

SQL> INSERT INTO STUDENT VALUES(8,'SV',ADDRESS(88,AREA\_TYPE('CHAKALI','WATERTANK'),'CHI'));

1 row created.

SQL> SELECT \* FROM STUDENT;

NO NAME ADDR(HNO, AREA(STREET\_NAME, LANDMARK), CITY)

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

7 SUKU ADDRESS(77, AREA\_TYPE('RAJAKA', 'PETROLB'), 'NLR')

8 SV ADDRESS(88, AREA\_TYPE('CHAKALI', 'WATERTANK'), 'CHI')

SQL> SELECT NO,NAME,F1.ADDR.HNO FROM STUDENT F1;

NO NAME ADDR.HNO

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

7 SUKU 77

8 SV 88

SQL> SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME FROM STUDENT F1;

NO NAME ADDR.HNO ADDR.AREA.

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

7 SUKU 77 RAJAKA

8 SV 88 CHAKALI

SQL> SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME,F1.ADDR.CITY FROM STUDENT F1;

NO NAME ADDR.HNO ADDR.AREA. ADDR.CITY

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

7 SUKU 77 RAJAKA NLR

8 SV 88 CHAKALI CHI

SQL> SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME,F1.ADDR.CITY FROM STUDENT F1 WHERE F1.ADDR.HNO=7;

no rows selected

SQL> SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME,F1.ADDR.CITY FROM STUDENT F1 WHERE F1.ADDR.HNO=77;

NO NAME ADDR.HNO ADDR.AREA. ADDR.CITY

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

7 SUKU 77 RAJAKA NLR

SQL> UPDATE STUDENT F1 SET F1.ADDR.HNO=89 WHERE F1.ADDR.HNO=88;

1 row updated.

SQL>SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME,F1.ADDR.CITY FROM STUDENT F1;

NO NAME ADDR.HNO ADDR.AREA. ADDR.CITY

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

7 SUKU 77 RAJAKA NLR

8 SV 89 CHAKALI CHI

SQL> DELETE FROM STUDENT F1 WHERE F1.ADDR.HNO=89;

1 row deleted.

SQL> SELECT NO,NAME,F1.ADDR.HNO ,F1.ADDR.AREA.STREET\_NAME,F1.ADDR.CITY FROM STUDENT F1;

NO NAME ADDR.HNO ADDR.AREA. ADDR.CITY

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

7 SUKU 77 RAJAKA NLR

SQL> CREATE TABLE ROCK1(ADDR ADDRESS);

Table created.

SQL> DROP TYPE ADDRESS;

DROP TYPE ADDRESS

\*

ERROR at line 1:

ORA-02303: cannot drop or replace a type with type or table dependents

SQL> DROP TABLE ROCK1;

Table dropped.

SQL> DROP TYPE ADDRESS;

Type dropped.

**14)DIRECTORY**:- 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 should be folder. This folder is called directory.

Syntax: create or replace directory as 'path';

Examples:-

SQL> create directory sukumar1 as 'd:\abcde';

Directory created.

NOTE:- Directory file is created in oracle folder. DBMS does not physically create folder at specified location. Therefore we should explicitly create folder with specified name.

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

owner of directory,directory name & directory path.

SQL> show user;

USER is "SYS"

SQL> desc dba\_directories;

Name Null? Type

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

OWNER NOT NULL VARCHAR2(30)

DIRECTORY\_NAME NOT NULL VARCHAR2(30)

DIRECTORY\_PATH VARCHAR2(4000)

SQL> select \* from dba\_directories where directory\_name='SUKUMAR1';

OWNER DIRECTORY\_NAME

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

DIRECTORY\_PATH

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

SYS SUKUMAR1

d:\abcde

SQL> DROP DIRECTORY SUKUMAR1;

Directory dropped.

-----> 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.