SIR C.R.REDDY COLLEGE OF ENGINEERING ELURU – 534 007
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DATABASE MANAGEMENT SYSTEM

LABORATORY MANUAL

III/ IV B.Tech (CSE): II - SEMESTER

Submitted
By

G.Satyanarayana,M.Tech

LAB OBJECTIVE:

Upon successful completion of this Lab the student will be able to:

- Creating database objects
- Modifying database objects
- Manipulating the data
- Retrieving the data from the database server
- Performing database operations in a procedural manner using pl/sql
- Performing database operations (create, update, modify, retrieve, etc.,) using front-end tools l
- Design and Develop applications like banking, reservation system, etc.,

GENERAL INSTRUCTIONS TO STUDENTS

1. Students should be regular and come prepared for the lab practice.

2. In case a student misses a class, it is his/her responsibility to complete that missed experiment(s).

3. Students should bring the observation book, lab journal and lab manual.

   Prescribed textbook and class notes can be kept ready for reference if required.

4. They should implement the given experiment individually.

5. While conducting the experiments students should see that their programs would meet the following criteria:

   - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
   - Programs should perform input validation (Data type, range error, etc.) and give appropriate error messages and suggest corrective actions.
   - Comments should be used to give the statement of the problem and every function should indicate the purpose of the function, inputs and outputs
   - Statements within the program should be properly indented
   - Use meaningful names for variables and functions.
   - Make use of Constants and type definitions wherever needed.

6. Once the experiment(s) get executed, they should show the program and results to the instructors and

   copy the same in their observation book.

7. Questions for lab tests and exam need not necessarily be limited to the questions in the manual, but

   could involve some variations and / or combinations of the questions.

Note: Above mentioned instructions can be modified based on the context of the lab.

INDEX

Introduction to SQL

Structure Query Language(SQL) is a programming language used for storing and managing data in RDBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) uses **SQL** as the standard database language. SQL is used to perform all type of data operations in RDBMS.

SQL contains different data types those are

- 1. char(size)
- 2. varchar(size)
- 3. varchar2(size)
- 4. date
- 5. number(p,s) //** P-PRECISION S-SCALE **//
- 6. number(size)
- 7. raw(size)
- 8. raw/long raw(size)

Different types of commands in SQL:

- A).**DDL commands: -** To create a database objects
- B).**DML commands: -** To manipulate data of a database objects
- C).**DQL command: -** To retrieve the data from a database.
- D).**DCL/DTL commands: -** To control the data of a database…

SQL Command

SQL defines following data languages to manipulate data of RDBMS.

DDL : Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Command	**Description**
create	to create new table or database
alter	for alteration

truncate	delete data from table
drop	to drop a table
rename	to rename a table

DML : Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Command	Description
insert	to insert a new row
update	to update existing row
delete	to delete a row
merge	merging two rows or two tables

TCL : Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Command	Description
commit	to permanently save

rollback	to undo change
savepoint	to save temporarily

DCL : Data Control Language

Data control language provides command to grant and take back authority.

Command	**Description**
grant	grant permission of right
revoke	take back permission.

DQL : Data Query Language

Command	**Description**
select	retrieve records from one or more table

Experiment:1

AIM:

To create a DDL to perform creation of table, alter, modify and drop column.

Data Definition Language (DDL) statements are used to define the database structure or schema. Some examples:

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

CREATION OF TABLE:

SYNTAX:

```
create table<tablename>(c olumn1 datatype,column2 datatype...);
```

EXAMPLE:

```
SQL>create table std(sno number(5),sname varchar(20),age number(5),sdob date,sm1

number(4,2),sm2 number(4,2),sm3 number(4,4));

Table created.

SQL>insert into std values(101,’AAA’,16,’03-jul-88’,80,90,98);

1 row created.

SQL>insert into std values(102,’BBB’,18,’04-aug-89’,88,98,90);

1 row created.
```

OUTPUT:

```
Select * from std;

SNO SNAME AGE SDOB SM1 SM2 SM3

101 AAA 16 03-jul-88 80 90 98

102 BBB 18 04-aug-89 88 98 90
```

ALTER TABLE WITH ADD:

```
SQL>create table student(id number(5),name varchar(10),game varchar(20));

Table created.
SQL>insert into student values(1,’mercy’,’cricket’);

1 row created.
```

SYNTAX:

```
alter table<tablename>add(col1 datatype,col2 datatype..);
```

EXAMPLE:

```
SQL>alter table student add(age number(4));

SQL>insert into student values(2,’sharmi’,’tennis’,19);
```

OUTPUT:

```
ALTER: select * from student;

ID NAME GAME

1 Mercy Cricket
```

ADD: select * from student;

```
ID NAME GAME AGE

1 Mercy cricket

2 Sharmi Tennis 19
```

ALTER TABLE WITH MODIFY:

SYNTAX:

```
Alter table<tablename>modify(col1 datatype,col2 datatype..);
```

EXAMPLE:

```
SQL>alter table student modify(id number(6),game varchar(25));
```

OUTPUT:

MODIFY

```
desc student;
NAME NULL? TYPE

Id              Number(6)

Name          Varchar(20)

Game          Varchar(25)

Age          Number(4)
```

DROP:

SYNTAX: drop table<tablename>;

EXAMPLE:

```
SQL>drop table student;

SQL>Table dropped.
```

RESULT:

Thus the DDL commands have been executed successfully.

Experiment:2

AIM:

To create a DML to perform inseration,delete,update.

DML COMMANDS

DML commands are the most frequently used SQL commands and is used to query and manipulate the existing database objects. Some of the commands are Insert, Select, Update, Delete.

Insert Command This is used to add one or more rows to a table. The values are separated by commas and the data types char and date are enclosed in apostrophes. The values must be entered in the same order as they are defined.

Select Commands It is used to retrieve information from the table. It is generally referred to as querying the table. We can either display all columns in a table or only specify column from the table.

Update Command It is used to alter the column values in a table. A single column may be updated or more than one column could be updated.

Delete command After inserting row in a table we can also delete them if required. The delete command consists of a from clause followed by an optional where clause.

Q1: Insert a single record into dept table.

```
Ans: SQL> insert into dept values (1,'IT','Tholudur');
1 row created.
```

Q2: Insert more than a record into emp table using a single insert command.

```
Ans: SQL> insert into emp values(&empno,'&ename','&job',&deptno,&sal);
Enter value for empno: 1
Enter value for ename: Mathi
Enter value for job: AP
Enter value for deptno: 1
Enter value for sal: 10000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(1,'Mathi','AP',1,10000)
1 row created.
SQL> / Enter value for empno: 2
Enter value for ename: Arjun
Enter value for job: ASP
Enter value for deptno: 2
Enter value for sal: 12000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(2,'Arjun','ASP',2,12000)
1 row created.
SQL> / Enter value for empno: 3
Enter value for ename: Gugan
Enter value for job: ASP
Enter value for deptno: 1
Enter value for sal: 12000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(3,'Gugan','ASP',1,12000)
1 row created.
```

Q3: Update the emp table to set the salary of all employees to Rs15000/- who are working as ASP

```
Ans: SQL> select * from emp;
EMPNO ENAME JOB DEPTNO SAL
```

```
---------- -------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 12000
3 Gugan ASP 1 12000
SQL> update emp set sal=15000 where job='ASP'; 2 rows updated.
SQL> select * from emp;
EMPNO ENAME JOB DEPTNO SAL
---------- -------------------- ------------- ---------- ----------
1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
```

Q4: Create a pseudo table employee with the same structure as the table emp and insert rows into the table using select clauses.

Ans:
```
SQL> create table employee as select * from emp;
Table created.
SQL> desc employee;
Name Null? Type
--------------------------------------- -------- ----------------------------
EMPNO NUMBER(6)
ENAME NOT NULL VARCHAR2(20)
JOB NOT NULL VARCHAR2(13)
DEPTNO NUMBER(3)
SAL NUMBER(7,2)
```

Q5: select employee name, job from the emp table

Ans:
```
SQL> select ename, job from emp;
  ENAME          JOB
-------------------- -------------
Mathi             AP
Arjun             ASP
Gugan             ASP
Karthik           Prof
Akalya            AP
Suresh            lect
6 rows selected
```

Experiment:3

AIM:To create a DCL to perform Grant and Revoke

Data Control Language (DCL) statements. Some examples:

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

Oracle provides extensive feature in order to safeguard information stored in its tables from

unauthorized viewing and damage. The rights that allow the user of some or all oracle resources on the server are called privileges.

GRANT:

a) Grant privileges using the GRANT statement

The grant statement provides various types of access to database objects such as tables, views and sequences and so on.

Syntax:

```
GRANT <object privileges>
ON <objectname>
TO<username>
```

b) **REVOKE :**

b) Revoke permissions using the REVOKE statement:

The REVOKE statement is used to deny the Grant given on an object.

Syntax:

```
REVOKE<object privilege> ON FROM<user name>;
```

AIM:- To perform TCL operations

TCL(TRNSACTION CONTROL LANGUAGE)

SAVEPOINT:

Write a query to implement the save point.

Syntax for save point:

```
SQL> SAVEPOINT <SAVE POINT NAME>;
SQL> SAVEPOINT S1;
Savepoint created.
SQL> SELECT * FROM EMP;
EMPNO ENAME DESIGNATIN SALARY
---------- ------------ - --------- ----------
101 NAGARAJAN LECTURER 16000
102 SARAVANAN ASST. PROF 16000
104 CHINNI HOD, PROF 45000
SQL> INSERT INTO EMP VALUES(105,'PARTHASAR','STUDENT',100);
1 row created.
SQL> SELECT * FROM EMP;
EMPNO ENAME DESIGNATIN SALARY
----- ------------ ---------- ----------
105 PARTHASAR STUDENT 100
```

PREPARED BY M.NAGARAJAN. VALLIAMMAI ENGG.COLLEGE

```
101 NAGARAJAN LECTURER 16000
102 SARAVANAN ASST. PROF 16000
104 CHINNI HOD, PROF 45000
```

ROLL BACK

Write a query to implement the Rollback.

Syntax for save point:

```
SQL> ROLL BACK <SAVE POINT NAME>;
SQL> ROLL BACK S1;
Rollback complete.
SQL> SELECT * FROM EMP;
EMPNO ENAME DESIGNATIN SALARY
---------- ------------ ---------- ----------
101 NAGARAJAN LECTURER 16000
102 SARAVANAN ASST. PROF 16000
103 PANNERSELVAM ASST. PROF 20000
104 CHINNI HOD, PROF 45000
```

COMMIT

Write a query to implement the Rollback.

Syntax for commit:

```
SQL> COMMIT;
```

PREPARED BY M.NAGARAJAN. VALLIAMMAI ENGG.COLLEGE

QUERY: 09

```
SQL> COMMIT;

        Commit complete
```

Result:
Thus the DCL commands have been successfully executed and the results are verified.

Experiment:4

AIM: TO PERFORM FORIEGNKEYS

Definition: Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

PRIMARY KEY:A primary key is a column (or columns) in a table that uniquely identifies the rows in that table.

CANDIDATE KEY:

A candidate key is a column that meets all of the requirements of a primary key. In other words, it has the potential to be a primary key.

```
SQL> desc student5;

 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SNO                                       NOT NULL NUMBER(38)
 SNAME                                              CHAR(30)
 LOGIN                                              VARCHAR2(30)
 SAGE                                               NUMBER(38)
 CGPA                                               FLOAT(63)



SQL> select * from student5;


       SNO SNAME                          LOGIN                                SAGE
CGPA
---------- ------------------------------ ------------------------------ ---------- ----------
         1 sai                            sai@gmail.com                          21        7.5
         2 adi                            adi@gmail.com                          22        6.6
         3 kum                            kum@gmail.com                          21        6.4
         4 san                            san@gmail.com                          20        6.8
         5 sar                            sar@gmail.com                         23          5.9



SQL> create table course
  2   (
  3    SNO int,
  4    Cno int,
  5    Cname char(30),
  6    primary key(SNO,Cno),
  7   foreign key(sno) references student5
  8   );

Table created.

SQL> desc course;
 Name
Null?     Type
 -------------------------------------------------------------------------------------------------
 SNO
NOT NULL NUMBER(38)
 CNO
NOT NULL NUMBER(38)
 CNAME
CHAR(30)

SQL> insert into course values(&SNO,&Cno,'&Cname');
Enter value for sno: 1
Enter value for cno: 1
Enter value for cname: civil
```

```
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(1,1,'civil')

1 row created.

SQL> /
Enter value for sno: 2
Enter value for cno: 3
Enter value for cname: cse
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(2,3,'cse')

1 row created.

SQL> /
Enter value for sno: 3
Enter value for cno: 2
Enter value for cname: eee
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(3,2,'eee')

1 row created.

SQL> /
Enter value for sno: 4
Enter value for cno: 7
Enter value for cname: ece
old   1: insert into course values(&SNO,&Cno,'&Cname')
```

```
new   1: insert into course values(4,7,'ece')


1 row created.


SQL> /

Enter value for sno: 5

Enter value for cno: 4

Enter value for cname: mech

old   1: insert into course values(&SNO,&Cno,'&Cname')

new   1: insert into course values(5,4,'mech')


1 row created.


SQL> select * from course;


       SNO        CNO CNAME

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

         1          1 civil

         2          3 cse

         3          2 eee

         4          7 ece

         5          4 mech


SQL> insert into student5 values(,'sa','sa@gmail.com',29,4.9);

insert into student5 values(,'sa','sa@gmail.com',29,4.9)

                            *

ERROR at line 1:

ORA-00936: missing expression
```

```
SQL> insert into course values(20,6,'it');
insert into course values(20,6,'it')
*
ERROR at line 1:
ORA-02291: integrity constraint (CSE179.SYS_C0015305) violated - parent key not found



SQL> update table student5 SET sno=15 where sno=1;
update table student5 SET sno=15 where sno=1
       *
ERROR at line 1:
ORA-00903: invalid table name



SQL>   update student5 SET sno=15 where sno=1;
  update student5 SET sno=15 where sno=1
*
ERROR at line 1:
ORA-02292: integrity constraint (CSE179.SYS_C0015305) violated - child record found



SQL>   insert into student5 values(null,'sa','sa@gmail.com',29,4.9);
  insert into student5 values(null,'sa','sa@gmail.com',29,4.9)
                              *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("CSE179"."STUDENT5"."SNO")
```

```
SQL> insert into student5 values(20,'sa','sa@gmail.com',29,4.9);

1 row created.

SQL>   insert into course values(20,6,'it');

1 row created.

SQL>
SQL> insert into student5 values(7,'sak','sak@gmail.com',28,4.9);

1 row created.

SQL> update
  2   student5 SET sno=15 where sno=7;

1 row updated.

SQL> select * from student5;

       SNO SNAME                          LOGIN                                SAGE
CGPA
---------- ------------------------------ ------------------------------ ---------- ----------
         1 sai                            sai@gmail.com                          21        7.5
         2 adi                            adi@gmail.com                          22        6.6
         3 kum                            kum@gmail.com                          21        6.4
         4 san                            san@gmail.com                          20        6.8
         5 sar                   sar@gmail.com                23          5.9
        20 sa                    sa@gmail.com                 29          4.9
        15 sak                   sak@gmail.com                28          4.9

7 rows selected.

SQL> delete table student5 where sno=1;
delete table student5 where sno=1
       *
ERROR at line 1:
ORA-00903: invalid table name



SQL> delete student5 where sno=5;
delete student5 where sno=5
*
ERROR at line 1:
ORA-02292: integrity constraint (CSE179.SYS_C0015305) violated - child record found



SQL> delete student5 where sno=15;

1 row deleted.

SQL> drop table course;

Table dropped.
```

```
SQL>

SQL>   create table course

  2    (

  3      SNO int,

  4      Cno int,

  5      Cname char(30),

  6      primary key(SNO,Cno),

  7    foreign key(sno) references student5

  8      on delete cascade

  9   );


Table created.


SQL> insert into course values(&SNO,&Cno,'&Cname');

Enter value for sno: 1

Enter value for cno: 1

Enter value for cname: civil

old   1: insert into course values(&SNO,&Cno,'&Cname')

new   1: insert into course values(1,1,'civil')


1 row created.


SQL> /

Enter value for sno: 2

Enter value for cno: 3

Enter value for cname: cse

old   1: insert into course values(&SNO,&Cno,'&Cname')

new   1: insert into course values(2,3,'cse')
```

```
1 row created.

SQL> /
Enter value for sno: 3
Enter value for cno: 7
Enter value for cname: eee
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(3,7,'eee')

1 row created.

SQL> /
Enter value for sno: 4
Enter value for cno: 2
Enter value for cname: ece
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(4,2,'ece')

1 row created.

SQL> /
Enter value for sno: 5
Enter value for cno: 6
Enter value for cname: mech
old   1: insert into course values(&SNO,&Cno,'&Cname')
new   1: insert into course values(5,6,'mech')
```

```
1 row created.


SQL> /

Enter value for sno: 20

Enter value for cno: 5

Enter value for cname: it

old   1: insert into course values(&SNO,&Cno,'&Cname')

new   1: insert into course values(20,5,'it')


1 row created.


SQL> select * from course;


       SNO        CNO CNAME

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

         1          1 civil

         2          3 cse

         3          7 eee

         4          2 ece

         5          6 mech

        20          5 it


6 rows selected.


SQL> delete student5 where sno=20;


1 row deleted.
```

```
SQL> select * from course;


       SNO        CNO CNAME
---------- ---------- ------------------------------
         1          1 civil
         2          3 cse
         3          7 eee
         4          2 ece
         5          6 mech


SQL>
```

Experiment:5

AIM: To perform View operations

In database theory, a view is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object. This pre-established query command is kept in the database dictionary. Unlike ordinary base tables in a relational database, a view does not form part of the physical schema: as a result set, it is a virtual table computed or collated dynamically from data in the database when access to that view is requested. Changes applied to the data in a relevant underlying table are reflected in the data shown in subsequent invocations of the view. In some NoSQL databases, views are the only way to query data.

```
SQL> create table emp3
  2  (
  3  eid int primary key,
  4  ename char(30),
  5  age int,
  6  sal real
  7  );

Table created.

SQL> insert into emp3 values(&eid,'&ename',&age,&sal);
Enter value for eid: 1
Enter value for ename: uma
Enter value for age: 20
Enter value for sal: 2500
old   1: insert into emp3 values(&eid,'&ename',&age,&sal)
new   1: insert into emp3 values(1,'uma',20,2500)

1 row created.

SQL> /
Enter value for eid: 2
Enter value for ename: prabhala
Enter value for age: 21

Enter value for sal: 2560

old    1: insert into emp3 values(&eid,'&ename',&age,&sal)

new     1: insert into emp3 values(2,'prabhala',21,2560)


1 row created.


SQL> /

Enter value for eid: 3

Enter value for ename: swathi

Enter value for age: 22

Enter value for sal: 2900

old    1: insert into emp3 values(&eid,'&ename',&age,&sal)

new     1: insert into emp3 values(3,'swathi',22,2900)


1 row created.


SQL> /

Enter value for eid: 4

Enter value for ename: maha

Enter value for age: 20

Enter value for sal: 2600

old    1: insert into emp3 values(&eid,'&ename',&age,&sal)

new     1: insert into emp3 values(4,'maha',20,2600)


1 row created.
```

```
SQL> /
Enter value for eid: 5
Enter value for ename: mouni
Enter value for age: 21
Enter value for sal: 2655
old   1: insert into emp3 values(&eid,'&ename',&age,&sal)
new   1: insert into emp3 values(5,'mouni',21,2655)

1 row created.

SQL> /
Enter value for eid: 6
Enter value for ename: manu
Enter value for age: 21
Enter value for sal: 1500
old   1: insert into emp3 values(&eid,'&ename',&age,&sal)
new   1: insert into emp3 values(6,'manu',21,1500)

1 row created.

SQL> /
Enter value for eid: 7
Enter value for ename: manasa
Enter value for age: 22
Enter value for sal: 2459
old   1: insert into emp3 values(&eid,'&ename',&age,&sal)
new   1: insert into emp3 values(7,'manasa',22,2459)
```

```
1 row created.


SQL> /

Enter value for eid: 8

Enter value for ename: umap

Enter value for age: 21

Enter value for sal: 2980

old   1: insert into emp3 values(&eid,'&ename',&age,&sal)

new   1: insert into emp3 values(8,'umap',21,2980)


1 row created.


SQL> /

Enter value for eid: 9

Enter value for ename: mno

Enter value for age: 34

Enter value for sal: 14678

old   1: insert into emp3 values(&eid,'&ename',&age,&sal)

new   1: insert into emp3 values(9,'mno',34,14678)


1 row created.


SQL> /

Enter value for eid: 10

Enter value for ename: san

Enter value for age: 24
Enter value for sal: 23456
old   1: insert into emp3 values(&eid,'&ename',&age,&sal)
new   1: insert into emp3 values(10,'san',24,23456)

1 row created.

SQL> select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
         1 uma                                    20       2500
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                  21       2655
         6 manu                                   21       1500
         7 manasa                                 22       2459
         8 umap                                   21       2980
         9 mno                                    34      14678
        10 san                                    24      23456

10 rows selected.

SQL>  select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
```

```
         1 uma                                        20          2500
         2 prabhala                                  21          2560
         3 swathi                                   22          2900
         4 maha                                       20          2600
         5 mouni                                     21          2655
         6 manu                                       21          1500
         7 manasa                                     22          2459
         8 umap                                       21          2980
         9 mno                                        34         14678
        10 san                                       24        23456

10 rows selected.

SQL> create view cse as
  2  (
  3  select eid ename
  4  from emp3
  5  );

View created.

SQL> select * from cse;

     ENAME
----------
         1
         2
         3
         4
         5
         6
         7
         8
         9
        10

10 rows selected.



SQL> create view cse1 as
  2   (
  3   select eid,ename
  4   from emp3
  5   );

View created.

SQL> select * from cse1;

       EID ENAME
---------- ------------------------------
         1 uma
         2 prabhala
         3 swathi
         4 maha

         5 mouni

         6 manu

         7 manasa

         8 umap

         9 mno

        10 san


10 rows selected.




SQL> create view cse2(eno,ename) as

  2    (

  3    select eid,ename

  4    from emp3

  5    );


View created.


SQL> select * from cse2;


       ENO ENAME

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

         1 uma

         2 prabhala

         3 swathi
         4 maha

         5 mouni

         6 manu

         7 manasa

         8 umap

         9 mno

        10 san


10 rows selected.


SQL>   create view cse3(eno,name) as

  2    (

  3    select eid,ename

  4    from emp3

  5    );


View created.


SQL> select * from cse3;


       ENO NAME

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

         1 uma

         2 prabhala

         3 swathi

         4 maha

         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san

10 rows selected.

SQL> drop view cse1;

View dropped.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
         1 uma
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
```

```
10 rows selected.

SQL> insert into cse3 values(11,'abc');

1 row created.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
         1 uma
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
        11 abc

11 rows selected.

SQL> select * from emp3;

       EID ENAME                                AGE        SAL
```

```
---------- ------------------------------ ---------- ----------
         1 uma                                    20       2500
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                  21       2655
         6 manu                                   21       1500
         7 manasa                                 22       2459
         8 umap                                   21       2980
         9 mno                                    34      14678
        10 san                                    24      23456
        11 abc

11 rows selected.

SQL> insert into emp3 values(12,'nikki',22,2400);

1 row created.

SQL> select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
         1 uma                                    20       2500
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                    21         2655
         6 manu                                     21         1500
         7 manasa                                   22         2459
         8 umap                                     21         2980
         9 mno                                      34        14678
        10 san                                      24        23456
        11 abc

       EID ENAME                                   AGE          SAL
---------- ------------------------------ ---------- ----------
        12 nikki                                    22         2400

12 rows selected.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
         1 uma
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
        11 abc

       ENO NAME
---------- ------------------------------
        12 nikki

12 rows selected.

SQL> update emp3
  2  set eid=30
  3  where eid=1;

1 row updated.

SQL> select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
        30 uma                                    20       2500
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                  21       2655
         6 manu                                   21       1500
         7 manasa                                 22       2459
         8 umap                                   21       2980
         9 mno                                   34      14678
        10 san                                   24      23456
        11 abc

       EID ENAME                                AGE        SAL
---------- ------------------------------ ---------- ----------
        12 nikki                                 22       2400

12 rows selected.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
        30 uma
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
        11 abc

       ENO NAME
```

```
---------- ------------------------------
        12 nikki

12 rows selected.

SQL> update cse3
  2  set eno=1
  3  where eno=30
  4  ;

1 row updated.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
         1 uma
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
        11 abc
       ENO NAME
---------- ------------------------------
        12 nikki

12 rows selected.

SQL> select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
         1 uma                                    20       2500
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                  21       2655
         6 manu                                   21       1500
         7 manasa                                 22       2459
         8 umap                                   21       2980
         9 mno                                    34      14678
        10 san                                    24      23456
        11 abc

       EID ENAME                                 AGE        SAL
---------- ------------------------------ ---------- ----------
        12 nikki                                  22       2400
```

```
12 rows selected.

SQL> delete emp3
  2  where eid=1;

1 row deleted.

SQL> select * from emp3;

       EID ENAME                                 AGE        SAL
---------- ----------------------------- ---------- ----------
         2 prabhala                               21       2560
         3 swathi                                 22       2900
         4 maha                                   20       2600
         5 mouni                                  21       2655
         6 manu                                   21       1500
         7 manasa                                 22       2459
         8 umap                                   21       2980
         9 mno                                    34      14678
        10 san                                    24      23456
        11 abc
        12 nikki                                  22       2400

11 rows selected.

SQL> select * from cse3;
```

```
       ENO NAME
---------- ------------------------------
         2 prabhala
         3 swathi
         4 maha
         5 mouni
         6 manu
         7 manasa
         8 umap
         9 mno
        10 san
        11 abc
        12 nikki

11 rows selected.

SQL> delete cse3
  2  where eno=2;

1 row deleted.

SQL> select * from cse3;

       ENO NAME
---------- ------------------------------
         3 swathi
         4 maha
         5 mouni

         6 manu

         7 manasa

         8 umap

         9 mno

        10 san

        11 abc

        12 nikki


10 rows selected.


SQL> select * from emp3;


       EID ENAME                                 AGE        SAL

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

         3 swathi                                22       2900

         4 maha                                  20       2600

         5 mouni                                 21       2655

         6 manu                                  21       1500

         7 manasa                                22       2459

         8 umap                                  21       2980

         9 mno                                   34      14678

        10 san                                   24      23456

        11 abc

        12 nikki                                 22       2400


10 rows selected.
```

```
SQL> drop view cse3;

View dropped.

SQL> select * from emp3;

       EID ENAME                                AGE        SAL
---------- ---------------------------- ---------- ----------
         3 swathi                                22       2900
         4 maha                                  20       2600
         5 mouni                                 21       2655
         6 manu                                  21       1500
         7 manasa                                22       2459
         8 umap                                  21       2980
         9 mno                                   34      14678
        10 san                                   24      23456
        11 abc
        12 nikki                                 22       2400

10 rows selected.

SQL> commit;

Commit
```

Experiment:6

AIM:To perform SQL queries in Sailors DataBase

Create table for sailors

```
SQL> create table sailors
  2  (
  3  sid int primary key,
  4  sname char(20),
  5  rating int,
  6  age real
  7  );
Table created.
SQL> insert into sailors values(&sid,'&sname',&rating,&age);
Enter value for sid: 22
Enter value for sname: dustin
Enter value for rating: 7
Enter value for age: 45.0
old   1: insert into sailors values(&sid,'&sname',&rating,&age)
new   1: insert into sailors values(22,'dustin',7,45.0)
1 row created.
SQL> /
Enter value for sid: 29
Enter value for sname: brutus
Enter value for rating: 1
Enter value for age: 33.0
old   1: insert into sailors values(&sid,'&sname',&rating,&age)
new   1: insert into sailors values(29,'brutus',1,33.0)

1 row created.
SQL> /
Enter value for sid: 31
Enter value for sname: lubber

Enter value for rating: 8

Enter value for age: 55.5

old    1: insert into sailors values(&sid,'&sname',&rating,&age)

new     1: insert into sailors values(31,'lubber',8,55.5)


1 row created.


SQL> /

Enter value for sid: 32

Enter value for sname: andy

Enter value for rating: 8

Enter value for age: 25.5

old    1: insert into sailors values(&sid,'&sname',&rating,&age)

new     1: insert into sailors values(32,'andy',8,25.5)

1 row created.

SQL> /

Enter value for sid: 58

Enter value for sname: rusty

Enter value for rating: 10

Enter value for age: 35.0

old    1: insert into sailors values(&sid,'&sname',&rating,&age)

new     1: insert into sailors values(58,'rusty',10,35.0)


1 row created.


SQL> /

Enter value for sid: 64
Enter value for sname: horatio

Enter value for rating: 7

Enter value for age: 35.0

old      1: insert into sailors values(&sid,'&sname',&rating,&age)

new      1: insert into sailors values(64,'horatio',7,35.0)


1 row created.


SQL> /

Enter value for sid: 71

Enter value for sname: zorba

Enter value for rating: 10

Enter value for age: 16.0

old      1: insert into sailors values(&sid,'&sname',&rating,&age)

new      1: insert into sailors values(71,'zorba',10,16.0)

1 row created.

SQL> /

Enter value for sid: 74

Enter value for sname: horatio

Enter value for rating: 9

Enter value for age: 35.0

old      1: insert into sailors values(&sid,'&sname',&rating,&age)

new      1: insert into sailors values(74,'horatio',9,35.0)


1 row created.


SQL> /

Enter value for sid: 85
Enter value for sname: art
Enter value for rating: 3
Enter value for age: 25.5
old   1: insert into sailors values(&sid,'&sname',&rating,&age)
new   1: insert into sailors values(85,'art',3,25.5)

1 row created.

SQL> /
Enter value for sid: 95
Enter value for sname: bob
Enter value for rating: 3
Enter value for age: 63.5
old   1: insert into sailors values(&sid,'&sname',&rating,&age)
new   1: insert into sailors values(95,'bob',3,63.5)

1 row created.

SQL> select * from sailors;

       SID SNAME                    RATING        AGE
---------- -------------------- ---------- ----------
        22 dustin                        7         45
        29 brutus                        1         33
        31 lubber                        8       55.5
        32 andy                          8       25.5
        58 rusty                        10         35
        64 horatio                       7         35
        71 zorba                              10          16
        74 horatio                             9          35
        85 art                                 3        25.5
        95 bob                                 3        63.5

10 rows selected.

SQL> 10 rows selected.

SQL>

SQL> create table boats
  2  (
  3  bid int primary key,
  4  bname char(30),
  5  color char(30)
  6  );

Table created.

SQL> insert into boats values(&bid,'&bname','&color');
Enter value for bid: 101
Enter value for bname: interlake
Enter value for color: blue
old   1: insert into boats values(&bid,'&bname','&color')
new   1: insert into boats values(101,'interlake','blue')

1 row created.

SQL> /
Enter value for bid: 102
Enter value for bname: interlake

Enter value for color: red

old      1: insert into boats values(&bid,'&bname','&color')

new      1: insert into boats values(102,'interlake','red')


1 row created.


SQL> /

Enter value for bid: 103

Enter value for bname: clipper

Enter value for color: green

old      1: insert into boats values(&bid,'&bname','&color')

new      1: insert into boats values(103,'clipper','green')


1 row created.


SQL> /

Enter value for bid: 104

Enter value for bname: marine

Enter value for color: red

old      1: insert into boats values(&bid,'&bname','&color')

new      1: insert into boats values(104,'marine','red')


1 row created.


SQL> select * from boats;


       BID BNAME                          COLOR
```

```
---------- ----------------------------- -----------------------------
       101 interlake                     blue
       102 interlake                     red
       103 clipper                        green
       104 marine                          red

SQL>




SQL> create table reserves
  2  (
  3  sid int,
  4  bid int,
  5  day date,
  6  primary key(sid,bid,day),
  7  foreign key (sid) references sailors,
  8  foreign key (bid) references boats
  9  );

Table created.

SQL> insert into reserves values(&sid,&bid,'&day');
Enter value for sid: 22
Enter value for bid: 101
Enter value for day: 10-oct-98
old   1: insert into reserves values(&sid,&bid,'&day')
new   1: insert into reserves values(22,101,'10-oct-98')
```

```
1 row created.

SQL> insert into reserves values(&sid,&bid,'&day');

Enter value for sid: 22

Enter value for bid: 102

Enter value for day: 10-oct-98

old   1: insert into reserves values(&sid,&bid,'&day')

new   1: insert into reserves values(22,102,'10-oct-98')


1 row created.


SQL> /

Enter value for sid: 22

Enter value for bid: 103

Enter value for day: 10-aug-98

old   1: insert into reserves values(&sid,&bid,'&day')

new   1: insert into reserves values(22,103,'10-aug-98')


1 row created.


SQL> /

Enter value for sid: 22

Enter value for bid: 104

Enter value for day: 10-jul-98

old   1: insert into reserves values(&sid,&bid,'&day')

new   1: insert into reserves values(22,104,'10-jul-98')


1 row created.
```

```
SQL> /
Enter value for sid: 31
Enter value for bid: 102
Enter value for day: 11-oct-98
old   1: insert into reserves values(&sid,&bid,'&day')
new   1: insert into reserves values(31,102,'11-oct-98')

1 row created.

SQL> /
Enter value for sid: 31
Enter value for bid: 103
Enter value for day: 11-jun-98
old   1: insert into reserves values(&sid,&bid,'&day')
new   1: insert into reserves values(31,103,'11-jun-98')

1 row created.

SQL> /
Enter value for sid: 31
Enter value for bid: 104
Enter value for day: 11-dec-98
old   1: insert into reserves values(&sid,&bid,'&day')
new   1: insert into reserves values(31,104,'11-dec-98')

1 row created.
```

```
SQL> /

Enter value for sid: 64

Enter value for bid: 101

Enter value for day: 9-may-98

old     1: insert into reserves values(&sid,&bid,'&day')

new     1: insert into reserves values(64,101,'9-may-98')


1 row created.


SQL> /

Enter value for sid: 64

Enter value for bid: 102

Enter value for day: 9-aug-98

old     1: insert into reserves values(&sid,&bid,'&day')

new     1: insert into reserves values(64,102,'9-aug-98')


1 row created.


SQL> /

Enter value for sid: 74

Enter value for bid: 103

Enter value for day: 9-aug-98

old     1: insert into reserves values(&sid,&bid,'&day')

new     1: insert into reserves values(74,103,'9-aug-98')


1 row created.


SQL> select * from reserves
```

```
  2  ;

       SID        BID DAY
---------- ---------- ---------
        22        101 10-OCT-98
        22        102 10-OCT-98
        22        103 10-AUG-98
        22        104 10-JUL-98
        31        102 11-OCT-98
        31        103 11-JUN-98
        31        104 11-DEC-98
        64        101 09-MAY-98
        64        102 09-AUG-98
        74        103 09-AUG-98

10 rows selected.

SQL> select * from boats;

       BID BNAME                          COLOR
---------- ------------------------------ ------------------------------
       101 interlake                      blue
       102 interlake                      red
       103 clipper                        green
       104 marine                         red

SQL> select * from sailors;
```

```
       SID SNAME                    RATING        AGE
---------- -------------------- ---------- ----------
        22 dustin                        7         45
        29 brutus                        1         33
        31 lubber                        8       55.5
        32 andy                          8       25.5
        58 rusty                        10         35
        64 horatio                       7         35
        71 zorba                        10         16
        74 horatio                       9         35
        85 art                           3       25.5
        95 bob                           3       63.5

10 rows selected.


SQL> select distinct s.sname,s.age
  2  from sailors s
  3  where rating>7;

SNAME                       AGE
-------------------- ----------
andy                       25.5
rusty                        35
zorba                        16
horatio                      35
lubber                     55.5


SQL>  select s.sname,s.age
```

```
  2   from sailors s
  3   where rating>7;

SNAME                    AGE
-------------------- ----------
lubber                  55.5
andy                    25.5
rusty                     35
zorba                     16
horatio                   35

SQL> select distinct s.sname,s.age
  2

SQL> select distinct s.sname,s.age
  2   from sailors s;

SNAME                    AGE
-------------------- ----------
andy                    25.5
rusty                     35
zorba                     16
horatio                   35
art                     25.5
brutus                    33
lubber                  55.5
bob                     63.5
dustin                    45
```

```
9 rows selected.


SQL> desc sailors
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SID                                       NOT NULL NUMBER(38)
 SNAME                                              CHAR(20)
 RATING                                             NUMBER(38)
 AGE                                                FLOAT(63)


SQL> select s.sid,s.sname,s.rating,s.age
  2  from sailors s
  3  where rating>7;


       SID SNAME                    RATING        AGE
---------- -------------------- ---------- ----------
        31 lubber                        8       55.5
        32 andy                          8       25.5
        58 rusty                        10         35
        71 zorba                        10         16
        74 horatio                       9         35


SQL> select sname
  2  from sailors s,reserves r
  3  where s.sid=r.sid AND r.bid=103;


SNAME
```

```
--------------------

dustin

lubber

horatio



SQL>   select r.sid

  2     from boats b,reserves r

  3   where r.bid=b.bid AND b.color='red';


       SID

----------

        22

        22

        31

        31

        64


SQL> select distinct r.sid   from boats b,reserves r

  2     where r.bid=b.bid AND b.color='red';


       SID

----------

        22

        31

        64


SQL> select s.sname
```

```
  2    from boats b,reserves r,sailors s
  3   where s.sid=r.sid AND b.bid=r.bid AND b.color='red';

SNAME
--------------------
dustin
dustin
lubber
lubber
horatio

SQL> select b.color
  2    from boats b,reserves r,sailors s
  3   where s.sid=r.sid AND b.bid=r.bid AND s.sname='lubber';

COLOR
------------------------------
red
green
red

SQL> select s.sname
  2   from reserves r,sailors s
  3   where s.sid=r.sid;

SNAME
--------------------
dustin
```

```
dustin

dustin

dustin

lubber

lubber

lubber

horatio

horatio

horatio


10 rows selected.


SQL>   select distinct s.sname

  2   from reserves r,sailors s

  3   where s.sid=r.sid;


SNAME

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

horatio

dustin

lubber


SQL>

SQL> select s.sname

  2   from sailors s

  3   order by sname;


SNAME
```

```
--------------------
andy
art
bob
brutus
dustin
horatio
horatio
lubber
rusty
zorba

10 rows selected.

SQL> select s.sname
  2   from sailors s
  3   order by sname desc;

SNAME
--------------------
zorba
rusty
lubber
horatio
horatio
dustin
brutus
bob
```

```
art

andy


10 rows selected.


SQL> select s.sname

  2   from sailors s

  3   order by sname asc;


SNAME

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

andy

art

bob

brutus

dustin

horatio

horatio

lubber

rusty

zorba


10 rows selected.


SQL> select s.sname

  2   from sailors s

  3   where sname like 'r%';

SNAME
```

```
-------------------
rusty

SQL> select s.sname
  2  from sailors s
  3  order by sname desc;


SNAME
--------------------
Zorba
rusty
lubber
horatio
horatio
dustin
brutus
bob
art
andy
10 rows selected.

SQL>  select *
  2  from sailors s
  3  order by sname desc;


       SID SNAME                      RATING        AGE
---------- -------------------- ---------- ----------
        71 zorba                        10         16
        58 rusty                        10         35
        31 lubber                        8       55.5
        64 horatio                       7         35
        74 horatio                       9         35
        22 dustin                          7            45
        29 brutus                          1            33
        95 bob                             3          63.5
        85 art                             3          25.5
    32 andy                        8        25.5
```

Experiment:7

AIM:To perform Aggregate functions

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning or measurement such as a set, a bag or a list.

Common aggregate functions include :

- Average() (i.e., arithmetic mean)
- Count()
- Maximum()
- nanmean() (mean ignoring NaN values, also known as "nil" or "null")
- Median()
- Minimum()
- Mode()
- Sum()

```
SQL> select count(*)

  2  from sailors;

  COUNT(*)

----------

        10

SQL>  select count(*)

  2   from boats;

  COUNT(*)

----------

         4

SQL>  select count(*)

  2   from reserves;

  COUNT(*)

----------

        10




SQL>  select distinct count(*)

  2  from boats;
  COUNT(*)
----------
         4
SQL> select count(sname)
  2  from sailors;
COUNT(SNAME)
------------
          10
SQL> select count(distinct sname)
  2  from sailors;
COUNT(DISTINCTSNAME)
--------------------
                   9
SQL> select sum(age)
  2  from sailors;
  SUM(AGE)
----------
       369
SQL> select sum(distinct age)
  2  from sailors;
SUM(DISTINCTAGE)
----------------
           273.5
SQL>  select sum(rating)
  2   from sailors;
```

```
SUM(RATING)
-----------
         66
SQL>   select sum(distinct rating)
  2   from sailors;
SUM(DISTINCTRATING)
-------------------
                 38
SQL> select avg(age)
  2   from sailors;
  AVG(AGE)
----------
      36.9
SQL> select avg(distinct age)
  2   from sailors;
AVG(DISTINCTAGE)
----------------
      39.0714286
SQL> select avg(rating)
  2   from sailors;

AVG(RATING)
-----------
        6.6


SQL> select avg(distinct rating)
  2   from sailors;
```

```
AVG(DISTINCTRATING)
-------------------
          6.33333333

SQL> select max(age)
  2   from sailors;

  MAX(AGE)
----------
      63.5

SQL> select max(rating)
  2   from sailors;

MAX(RATING)
-----------
          10
SQL> select min(age)
  2   from sailors;
  MIN(AGE)
----------
        16
```

Experiment:8

Aim:To perform UNION,INTERSECT, EXCEPT operations

The UNION, INTERSECT, and EXCEPT clauses are used to combine or exclude like rows from two or more tables. They are useful when you need to combine the results from separate queries into one single result. They differ from a join in that entire rows are matched and, as a result, included or excluded from the combined result.

Overview

These operators can be used on any query; however, a couple simple of conditions must be met:

1. The number and order columns must be the same in both queries
2. The data types must be the same or compatible.

UNION Operator

The Union operator returns rows from both tables. If used by itself, UNION returns a distinct list of rows. Using UNION ALL, returns all rows from both tables. A UNION is useful when you want to sort results from two separate queries as one combined result. For instance if you have two tables, Vendor, and Customer, and you want a combined list of names, you can easily do so using:

INTERSECT Operator

Use an intersect operator to returns rows that are in common between two tables; it returns unique rows from both the left and right query. This query is useful when you want to find results that are in common between two queries. Continuing with Vendors, and Customers, suppose you want to find vendors that are also customers. You can do so easily using:

```
select * from sailors;
  select * from reserves;
select * from boats;
SQL> select s1.sname from sailors s1,reserves r1,boats b1 where s1.sid=r1.sid and b1.bid=r1.bid and
b1.color='red'
  2  union
  3  select s2.sname from sailors s2,reserves r2,boats b2 where s2.sid=r2.sid and b2.bid=r2.bid and
b2.color='green';
```

```
SNAME
--------------------
dustin
horato
lubber

SQL> select s1.sname from sailors s1,reserves r1,boats b1 where s1.sid=r1.sid and b1.bid=r1.bid and
b1.color='red'
     intersect
     select s2.sname from sailors s2,reserves r2,boats b2 where s2.sid=r2.sid and b2.bid=r2.bid and
b2.color='green';

SNAME
--------------------
dustin
horato
lubber




SQL> select s1.sname from sailors s1,reserves r1,boats b1 where s1.sid=r1.sid and b1.bid=r1.bid and
b1.color='red'
       minus
     select s2.sname from sailors s2,reserves r2,boats b2 where s2.sid=r2.sid and b2.bid=r2.bid and
b2.color='green';
```

```
no rows selected
```

AIM:To perform nested queries

```
SQL> select s.sname from sailors s where s.sid in
  2  (select r.sid from reserves r where r.bid=103);

SNAME
--------------------
dustin
lubber
horato

SQL> select s.sname from sailors s where s.sid in
  2  (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='red');

SNAME
--------------------
dustin
lubber
horato
```

```
SQL> select s.sname from sailors s where s.sid not in
  2  (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='red');
```

```
SNAME
--------------------
brurus
andy
rusty
zorba
horato
art
bod

7 rows selected.

SQL> select s.sname from sailors s where s.sid not in
  2  (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='green');

SNAME
--------------------
brurus
andy
rusty
horato
zorba
art
bod

7 rows selected.
```

```
SQL> select s.sname from sailors s where s.sid not in
  2  (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='green' and b.color='red');


SNAME
--------------------
dustin
brurus
lubber
andy
rusty
horato
zorba
horato
art
bod

10 rows selected.

SQL> select s.sname from sailors s where s.sid in
  2  (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='green' and b.color='red');


no rows selected

SQL> select * from sailors;
```

```
       SID SNAME                     RATING        AGE
---------- -------------------- ---------- ----------
        22 dustin                        7         45
        29 brurus                        1         33
        31 lubber                        8       55.5
        32 andy                          8       25.5
        58 rusty                        10         35
        64 horato                        7         35
        71 zorba                        10         16
        74 horato                        9         35
        85 art                           3       25.5
        95 bod                           3       63.5

10 rows selected.

SQL> select s.sname from sailors s where s.sid in
  2    (select r.sid from reserves r,boats b where r.bid=b.bid and b.color='green' and b.color='red');


no rows selected
```

Experiment:9

AIM;To perform nested queries.

NESTED QUERIES

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

Subqueries must be enclosed within parentheses.

A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.

Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.

The SELECT list cannot include any references to values that evaluate to a BLOB, ARRAY, CLOB, or NCLOB.

A subquery cannot be immediately enclosed in a set function.

The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement:

Subqueries are most frequently used with the SELECT statement. The basic syntax is as follows:

```
SELECT column_name [, column_name ]

FROM    table1 [, table2 ]

WHERE   column_name OPERATOR

        (SELECT column_name [, column_name ]

        FROM table1 [, table2 ]

        [WHERE])
```

1)

```
select s.sname from sailors s where s.sid in
(select r.sid from reserves r where r.bid=103);
2)
select s.sname from sailors s where s.sid in
(select r.sid from reserves r,boats b where r.bid=b.bid and b.color='red');
3)
select s.sname from sailors s where s.sid not in
(select r.sid from reserves r,boats b where r.bid=b.bid and b.color='red');
4)
select s.sname from sailors s where s.sid not in
(select r.sid from reserves r,boats b where r.bid=b.bid and b.color='green');
5)
select s.sname from sailors s where s.sid in
(select r1.sid from reserves r1,boots b1 where r1.bid=b1.bid and b1.color='green'
intersect
select r2.sid from reserves r2,boots b2 where r2.bid=b2.bid and b2.color='red');
6)
select s.sname from sailors s where s.sid in
(select r.sid from reserves r,boots b where r.bid=b.bid and b.color='red' and
r.sid not in
(select r.sid from reserves r,boots b where r.bid=b.bid and b.color='green'));
```

ANSWER

```
SNAME
--------------------
Horatio
```

Experiment:10

AIM:TO Perform Trigger operations

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).

A database definition (DDL) statement (CREATE, ALTER, or DROP).

A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers could be defined on the table, view, schema, or database with which the event is associated.

Benefits of Triggers

Triggers can be written for the following purposes:

Generating some derived column values automatically

Enforcing referential integrity

Event logging and storing information on table access

Auditing

Synchronous replication of tables

Imposing security authorizations

Preventing invalid transactions

Creating Triggers

The syntax for creating a trigger is:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
    Declaration-statements
BEGIN
    Executable-statements
EXCEPTION
    Exception-handling-statements
END;
```

```
SQL> create table emp123
  2   (
  3   eno int primary key,
  4   ename char(20),
  5   eage int,
  6  eadd varchar2(30)
  7  ,
  8  ephone int
  9  );

Table created.

SQL> create or replace trigger cse
  2  after update or delete or insert on emp123
  3  for each row
  4  begin
  5  if updating then
  6  dbms_output.put_line('table is updated');
  7  elsif inserting then
  8    dbms_output.put_line('table is inserting');
  9  elsif deleting then
 10    dbms_output.put_line('row is deleted');
 11  end if;
 12  end;
 13  ;
 14

SQL>   create trigger cse
  2   after update or delete or insert on emp123
  3   for each row
  4   begin
  5   if updating then
  6   dbms_output.put_line('table is updated');
  7   elsif inserting then
  8     dbms_output.put_line('table is inserting');
  9   elsif deleting then
 10     dbms_output.put_line('row is deleted');
 11   end if;
 12   end;
 13
 14
 15

SQL>

SQL> SET SERVEROUTPUT ON
SQL>   create trigger cse
  2   after update or delete or insert on emp123
  3   for each row
  4   begin
  5   if updating then
  6   dbms_output.put_line('table is updated');
  7   elsif inserting then
  8     dbms_output.put_line('table is inserting');
  9   elsif deleting then
 10     dbms_output.put_line('row is deleted');
 11   end if;
 12   end;
 13
 14
 15

SQL>   create or replace trigger cse
  2    after update or delete or insert on emp123
  3    for each row
  4    begin
  5    if updating then
  6    dbms_output.put_line('table is updated');
  7    elsif inserting then
  8      dbms_output.put_line('table is inserting');
  9    elsif deleting then
 10      dbms_output.put_line('row is deleted');
 11    end if;
 12    end;
 13
 14
 15  /

Trigger created.

SQL> insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);
Enter value for eno: 1
Enter value for ename: uma
Enter value for eage: 20
Enter value for eadd: elr
Enter value for ephone: 8019
old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)
new   1: insert into emp123 values(1,'uma',20,'elr',8019)
```

```
table is inserting

1 row created.

SQL> /
Enter value for eno: 2
Enter value for ename: manu
Enter value for eage: 21
Enter value for eadd: chpd
Enter value for ephone: 9581
old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)
new   1: insert into emp123 values(2,'manu',21,'chpd',9581)
table is inserting

1 row created.

SQL> /
Enter value for eno: 3
Enter value for ename: maha
Enter value for eage: 20
Enter value for eadd: rrpet
Enter value for ephone: 9683
old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)
new   1: insert into emp123 values(3,'maha',20,'rrpet',9683)
table is inserting

1 row created.
```

```
SQL> select * from emp123;

       ENO ENAME                                EAGE EADD                                      EPHONE
---------- -------------------- ---------- ------------------------------ ----------
         1 uma                                  20 elr                                          8019
         2 manu                                 21 chpd                                         9581
         3 maha                                 20 rrpet                                        9683

SQL> update emp123
  2   set ename='mouni'
  3   where ename='maha'
  4   ;
table is updated

1 row updated.

SQL> select * from emp123;

       ENO ENAME                                EAGE EADD                                      EPHONE
---------- -------------------- ---------- ------------------------------ ----------
         1 uma                                  20 elr                                     8019
         2 manu                                 21 chpd                                     9581
         3 mouni                                20 rrpet                                   9683

SQL>   insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);
```

```
Enter value for eno: 4

Enter value for ename: abc

Enter value for eage: 12

Enter value for eadd: acb

Enter value for ephone: 12334

old   1:  insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)

new   1:  insert into emp123 values(4,'abc',12,'acb',12334)

table is inserting


1 row created.


SQL> delete emp123

  2  where


SQL> select * from emp123;


       ENO ENAME                      EAGE EADD                                EPHONE

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

         1 uma                          20 elr                                  8019

         2 manu                         21 chpd                                 9581

         3 mouni                        20 rrpet                                9683

         4 abc                          12 acb                                 12334


SQL> delete emp123

  2  where eno=4;

row is deleted


1 row deleted.
```

```
SQL>

SQL> select * from emp123;

       ENO ENAME                              EAGE EADD                                       EPHONE
---------- ------------------- ---------- ----------------------------- ----------
         1 uma                                  20 elr                                      8019
         2 manu                                 21 chpd                                       9581
         3 mouni                               20 rrpet                                    9683



SQL> create or replace trigger cse1
  2  before   update or delete or insert on emp123
  3    for each row
  4    begin
  5    if updating then
  6    dbms_output.put_line('table is updated');
  7    elsif inserting then
  8      dbms_output.put_line('table is inserting');
  9    elsif deleting then
 10      dbms_output.put_line('row is deleted');
 11    end if;
 12    end;
 13  /

Trigger created.

SQL> insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);
```

```
Enter value for eno: 4
Enter value for ename:
Enter value for eage: 23
Enter value for eadd: abc
Enter value for ephone: 9441
old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)
new   1: insert into emp123 values(4,'',23,'abc',9441)
table is inserting
table is inserting

1 row created.

SQL> update emp123
  2  set ename='maha'
  3  where eno=4;
table is updated
table is updated

1 row updated.

SQL> select * from emp123;

       ENO ENAME                    EAGE EADD                           EPHONE
---------- -------------------- ---------- ------------------------------ ----------
         1 uma                        20 elr                              8019
         2 manu                       21 chpd                             9581
         3 mouni                      20 rrpet                            9683
         4 maha                       23 abc                              9441
```

```
SQL> delete emp123
  2   where eno=4;
row is deleted
row is deleted

1 row deleted.

SQL> select * from emp123;

       ENO ENAME                      EAGE EADD                           EPHONE
---------- -------------------- ---------- ------------------------------ ----------
         1 uma                          20 elr                             8019
         2 manu                         21 chpd                            9581
         3 mouni                        20 rrpet                           9683

SQL>




SQL>   create table hospital
  2    (
  3   hid int,
  4   hname char(30),
  5   hadd varchar2(40)
  6   );
```

```
Table created.

SQL> create or replace trigger medical
  2       before   update or delete or insert on emp123
  3         for each row
  4         begin
  5         if updating then
  6         dbms_output.put_line('hospital data is updated');
  7         elsif inserting then
  8           dbms_output.put_line('hospital data is inserting');
  9         elsif deleting then
 10         dbms_output.put_line('hospital data is deleted');
 11       end if;
 12   end;
 13   /

Trigger created.

SQL> create or replace trigger medical1
  2       before   update or delete or insert on hospital
  3         for each row
  4         begin
  5         if updating then
  6       nsertrhospital data is updated');
  7         elsif inserting then
  8           dbms_output.put_line('hospital data is inserting');
  9         elsif deleting then
 10         dbms_output.put_line('hospital data is deleted');
 11          end if;
 12       end;
 13       /

Trigger created.

SQL> insert into hospital values(&hid,'&hname','&hadd');
Enter value for hid: 1
Enter value for hname: kamineni
Enter value for hadd: bza
old     1: insert into hospital values(&hid,'&hname','&hadd')
new     1: insert into hospital values(1,'kamineni','bza')
hospital data is inserting

1 row created.

SQL> /
Enter value for hid: 2
Enter value for hname: apolo
Enter value for hadd: hyd
old     1: insert into hospital values(&hid,'&hname','&hadd')
new     1: insert into hospital values(2,'apolo','hyd')
hospital data is inserting

1 row created.

SQL> update hospital
  2    set hid=01
  3   where hid=1;
hospital data is updated


1 row updated.


SQL> select * from hospital;


       HID HNAME                          HADD
---------- ------------------------------ ----------------------------------------
         1 kamineni                       bza
         2 apolo                          hyd


SQL>   update hospital
  2     set hid=07
  3     where hid=1;
hospital data is updated


1 row updated.


SQL> select * from hospital;


       HID HNAME                          HADD
---------- ------------------------------ ----------------------------------------
         7 kamineni                       bza
         2 apolo                          hyd


SQL> delete hospital
  2   where hid=7;
```

```
hospital data is deleted

1 row deleted.

SQL> select * from hospital;

       HID HNAME                          HADD
---------- ------------------------------ ----------------------------------------
         2 apolo                          hyd

SQL> create trigger cse127
  2  after insert on emp123
  3  for each row
  4  begin
  5  if(:new.eage>0)then
  6    dbms_output.put_line('valid age');
  7  else
  8  dbms_output.put_line('invalid age');
  9  end if;
 10  end;
 11  /

Trigger created.

SQL> insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);
Enter value for eno: 5
Enter value for ename: xyz
Enter value for eage: -20
Enter value for eadd: asd

Enter value for ephone: 2345

old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)

new   1: insert into emp123 values(5,'xyz',-20,'asd',2345)

hospital data is inserting

invalid age

table is inserting


1 row created.


SQL> drop trigger medical;


Trigger dropped.



SQL> insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);

Enter value for eno: 6

Enter value for ename: qwe

Enter value for eage: 12

Enter value for eadd: qer

Enter value for ephone: 12345

old   1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)

new   1: insert into emp123 values(6,'qwe',12,'qer',12345)

valid age

table is inserting


1 row created.
```

```
create trigger age
  after insert on emp123
    for each row
  begin
  if(:new.eage<0)then
        raise_application_error(-20000,'no negative age on data');
end if;
end;
/
```

```
SQL> create trigger age
  2    after insert on emp123
  3      for each row
  4    begin
  5    if(:new.eage<0)then
  6          raise_application_error(-20000,'no negative age on data');
  7  end if;
  8  end;
  9  /

Trigger created.

SQL> insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone);
```

```
Enter value for eno: 7

Enter value for ename: ert

Enter value for eage: -54

Enter value for eadd: wer

Enter value for ephone: 12121

old      1: insert into emp123 values(&eno,'&ename',&eage,'&eadd',&ephone)

new      1: insert into emp123 values(7,'ert',-54,'wer',12121)

insert into emp123 values(7,'ert',-54,'wer',12121)

            *

ERROR at line 1:

ORA-20000: no negative age on data

ORA-06512: at "CSE127.AGE", line 3

ORA-04088: error during execution of trigger 'CSE127.AGE'
```

Experiments:10

AIM:To perform PL/SQL programs

1.AIM:To perform sum of 2 numbers

```
SQL> set serveroutput on;
SQL> declare
  2     a number(2);
  3     b number(2);
  4     c number(2);
  5     begin
  6     dbms_output.put_line('enter a number');
  7     a:=&a;
  8     dbms_output.put_line('enter b number');
  9     b:=&b;
 10     c:=a+b;
 11     dbms_output.put_line('the sum of two numbers are:'||a||'+'||b||'='||c);
 12     end;
 13     /
Enter value for a: 23
old     7:   a:=&a;
new      7:   a:=23;
Enter value for b: 24
old     9:   b:=&b;
new      9:   b:=24;
enter a number
enter b number
the sum of two numbers are:23+24=47


PL/SQL procedure successfully completed.
```

2)largest of 2 numbers

```
SQL> declare
```

```
  2  a number;
  3  b number;
  4  begin
  5  dbms_output.put_line('enter a number');
  6  a:=&number;
  7  dbms_output.put_line('enter bnumber');
  8  b:=&number;
  9  if a>b then
 10  dbms_output.put_line('a is largest'  ||a);
 11  else
 12  dbms_output.put_line('b is largest'  ||b);
 13  end if;
 14  end
 15  ;
 16  /
Enter value for number: 12
old   6:  a:=&number;
new   6:  a:=12;
Enter value for number: 21
old   8:  b:=&number;
new   8:  b:=21;
enter a number
enter bnumber
b is largest21

PL/SQL procedure successfully completed.
```

3) largest of 3 numbers

```
SQL> declare
```

```
  2     a number :=&a;
  3   b number :=&b;
  4   c number :=&c;
  5   begin
  6   if a>b and a>c then
  7     dbms_output.put_line(a||'is greatest');
  8   elsif b>a and b>c then
  9     dbms_output.put_line(b||'is graetest');
 10   else
 11     dbms_output.put_line(c||'is greatest');
 12   end if;
 13   end;
 14   /
Enter value for a: 12
old      2:    a number :=&a;
new      2:    a number :=12;
Enter value for b: 21
old      3: b number :=&b;
new      3: b number :=21;
Enter value for c: 8
old      4: c number :=&c;
new      4: c number :=8;
21is graetest

PL/SQL procedure successfully completed.
```

4) sum of avg of 3 numbers

```
SQL> declare
```

```
  2        a number :=&a;
  3      b number :=&b;
  4      c number :=&c;
  5    sm number;
  6    av number;
  7    begin
  8    sm:=a+b+c;
  9    av:=sm/3;
 10      dbms_output.put_line('sum='||sm);
 11      dbms_output.put_line('avg='||av);
 12    end;
 13    /
Enter value for a: 2
old   2:       a number :=&a;
new   2:       a number :=2;
Enter value for b: 4
old   3:     b number :=&b;
new   3:     b number :=4;
Enter value for c: 6
old   4:     c number :=&c;
new   4:     c number :=6;
sum=12
avg=4

PL/SQL procedure successfully completed.
```

5) sum of digits of a number

```
  SQL> declare
    2    n number;
    3    s number:=0;
    4    r number;
    5    begin
    6    n:=&n;
    7    while n<>0 loop
    8    r:=mod(n,10);
    9    s:=s+r;
   10    n:=trunc(n/10);
   11    end loop;
   12      dbms_output.put_line('the sum of digits is '||s);
   13    end;
   14    /
Enter value for n: 5
old      6: n:=&n;
new      6: n:=5;
the sum of digits is 5

PL/SQL procedure successfully completed.
```

6) multiplication table

```
SQL> declare
    2     a number(2):=&a;
    3     b number(2):=1;
    4     c number(3);
    5     begin
    6     while b<=10
```

```
  7     loop
  8     c:=a*b;
  9   dbms_output.put_line(a||'*'||b||'='||c);
 10   b:=b+1;
 11   end loop;
 12   end;
 13   /
Enter value for a: 5
old      2:    a number(2):=&a;
new      2:    a number(2):=5;
5*1=5
5*2=10
5*3=15
5*4=20
5*5=25
5*6=30
5*7=35
5*8=40
5*9=45
5*10=50

PL/SQL procedure successfully completed.
```

7) print in reverse order

```
SQL> declare
```

```
  2    n number(5):=&n;
  3    rev number(5):=0;
  4    r number(5):=0;
  5    begin
  6    while n!=0
  7    loop
  8    r:=mod(n,10);
  9    rev:=rev*10+r;
 10    n:=trunc(n/10);
 11    end loop;
 12    dbms_output.put_line('the reverse number is '||rev);
 13    end;
 14    /
Enter value for n: 456
old   2:   n number(5):=&n;
new   2:   n number(5):=456;
the reverse number is 654

PL/SQL procedure successfully completed.
```

8) PALINDROME ARE NOT

```
SQL> declare
  2  len number;
  3  str varchar2(20):='&input_string';
  4  chkstr varchar2(20);
  5  begin
  6  len:=length(str);
  7  for i in reverse 1..len loop
  8  chkstr:=chkstr||substr(str,i,1);
  9  end loop;
 10  if chkstr=str then
 11  dbms_output.put_line(str||'is a palindrome!');
 12  else
 13  dbms_output.put_line(str||'is not a palindrome!');
 14  end if;
 15  end;
 16  /
Enter value for input_string: 121
old   3: str varchar2(20):='&input_string';
new   3: str varchar2(20):='121';
121is a palindrome!

PL/SQL procedure successfully completed.
```

9) Number prime or not

```
SQL> declare
  2  n number;
  3  i number;
  4  counter number;
  5  begin
  6  n:=&n;
  7  i:=1;
  8  counter:=0;
  9  if n=1
 10  then
 11  dbms_output.put_line('1 is neither prime nor composit,');
 12  elsif n=2
 13  then
 14  dbms_output.put_line('2 is even prime');
 15  else
 16  for i in 1..n loop
 17  if mod(n,i)=0
 18  then counter:=counter+1;
 19  end if;
 20  end loop;
 21  end if;
 22  if counter=2
 23  then dbms_output.put_line(n||'is a prime no.');
 24  else
 25  dbms_output.put_line(n||'is a not prime no,');
 26  end if;
```

```
  27    end;
  28    /
Enter value for n: 121
old      6: n:=&n;
new      6: n:=121;
121is a not prime no,

PL/SQL procedure successfully completed.
```

10) prime numbers up to n numbers

```
SQL> declare
```

```
  2    num number;

  3    prime integer;

  4    begin

  5    num:=&num;

  6    for i in 1..num loop

  7    prime:=1;

  8    for j in 2..i-1

  9    loop

 10    if mod(i,j)=0 then

 11    prime:=0;

 12    end if;

 13    exit when prime=0;

 14    end loop;

 15    if prime=1 then

 16    dbms_output.put_line(i);

 17    end if;

 18    end loop;

 19    end;

 20    /

Enter value for num: 6

old      5: num:=&num;

new      5: num:=6;

1
2
3
5
PL/SQL procedure successfully completed.
```

11) Number is Armstrong or not

```
SQL> declare
```

```
  2   pnum number(5);
  3   tot number(5);
  4   ip number(3);
  5   tmp number(5);
  6   begin
  7   pnum:=&pnum;
  8   tmp:=pnum;
  9   tot:=0;
 10   while tmp>0
 11   loop
 12   ip:=tmp mod 10;
 13   tot:=tot+(ip*ip*ip);
 14   tmp:=floor(tmp/10);
 15   end loop;
 16   if(tot like pnum) then
 17   dbms_output.put_line(pnum||'is armstrong,');
 18   else
 19   dbms_output.put_line(pnum||'is not armstrong,');
 20   end if;
 21   end;
 22   /
Enter value for pnum: 121
old     7: pnum:=&pnum;
new     7: pnum:=121;
121is not armstrong,
PL/SQL procedure successfully completed.
```

12) count the number of characters and words

```
SQL>    declare
```

```
  2    str varchar2(20):='&str';
  3    noc number(4):=0;
  4    now number(4):=1;
  5    s char;
  6    begin
  7    for i in 1..length(str)
  8    loop
  9    s:=substr(str,i,1);
 10    noc:=now+1;
 11      if s="then
 12      now:=now+1;
 13      end if;
 14      end loop;
 15      dbms_output.put_line('the no.of chars'||noc);
 16      dbms_output.put_line('the no.of words'||now);
 17    end;
 18      /
Enter value for str: adi
old      2: str varchar2(20):='&str';
new      2: str varchar2(20):='adi';
the no.of chars2
the no.of words1
PL/SQL procedure successfully completed.
```

13) accept and concat the two strings

```
  SQL> declare
  2   str varchar2(20):='&str';
  3   str1 varchar2(20):='&str1';
  4   v varchar2(40);
  5   begin
  6   v:=str||"||str1;
  7   dbms_output.put_line(v);
  8   end;
  9   /
Enter value for str: adi
old   2: str varchar2(20):='&str';
new   2: str varchar2(20):='adi';
Enter value for str1: tya
old   3: str1 varchar2(20):='&str1';
new   3: str1 varchar2(20):='tya';
aditya

PL/SQL procedure successfully completed.
```

Experiment:10

AIM:To perform cursors operations

Oracle creates a memory area, known as context area, for processing an SQL statement, which contains all information needed for processing the statement, for example, number of rows processed, etc.

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set.

You can name a cursor so that it could be referred to in a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

1.Implicit cursors

2.Explicit cursors

Implicit Cursors
Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Whenever a DML statement (INSERT, UPDATE and DELETE) is issued, an implicit cursor is associated with this statement. For INSERT operations, the cursor holds the data that needs to be inserted. For UPDATE and DELETE operations, the cursor identifies the rows that would be affected.

In PL/SQL, you can refer to the most recent implicit cursor as the SQL cursor, which always has the attributes like %FOUND, %ISOPEN, %NOTFOUND, and %ROWCOUNT. The SQL cursor has additional attributes, %BULK_ROWCOUNT and %BULK_EXCEPTIONS, designed for use with the FORALL statement. The following table provides the description of the most used attributes:

Attribute	Description
%FOUND	Returns TRUE if an INSERT, UPDATE, or DELETE statement affected one or more rows or a SELECT INTO statement returned one or more rows. Otherwise, it returns FALSE.
%NOTFOUND	The logical opposite of %FOUND. It returns TRUE if an INSERT, UPDATE, or DELETE statement affected no rows, or a SELECT INTO statement returned no rows. Otherwise, it returns FALSE.
%ISOPEN	Always returns FALSE for implicit cursors, because Oracle closes the SQL cursor automatically after executing its associated SQL statement.
%ROWCOUNT	Returns the number of rows affected by an INSERT, UPDATE, or DELETE statement, or returned by a SELECT INTO statement.

Any SQL cursor attribute will be accessed as sql%attribute_name as shown below in the example.

Example:
We will be using the CUSTOMERS table we had created and used in the previous chapters.

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
+----+----------+-----+-----------+----------+
```

The following program would update the table and increase salary of each customer by 500 and use the SQL%ROWCOUNT attribute to determine the number of rows affected:

```
DECLARE
   total_rows number(2);
BEGIN
   UPDATE customers
   SET salary = salary + 500;
   IF sql%notfound THEN
      dbms_output.put_line('no customers selected');
   ELSIF sql%found THEN
      total_rows := sql%rowcount;
      dbms_output.put_line( total_rows || ' customers selected ');
   END IF;
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
6 customers selected

PL/SQL procedure successfully completed.
```

If you check the records in customers table, you will find that the rows have been updated:

```
Select * from customers;

+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2500.00 |
|  2 | Khilan   |  25 | Delhi     |  2000.00 |
|  3 | kaushik  |  23 | Kota      |  2500.00 |
|  4 | Chaitali |  25 | Mumbai    |  7000.00 |
|  5 | Hardik   |  27 | Bhopal    |  9000.00 |
|  6 | Komal    |  22 | MP        |  5000.00 |
+----+----------+-----+-----------+----------+
```

Explicit Cursors

Explicit cursors are programmer defined cursors for gaining more control over the context area. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

The syntax for creating an explicit cursor is :

```
CURSOR cursor_name IS select_statement;
```

Working with an explicit cursor involves four steps:

Declaring the cursor for initializing in the memory

Opening the cursor for allocating memory

Fetching the cursor for retrieving data

Closing the cursor to release allocated memory

Declaring the Cursor

Declaring the cursor defines the cursor with a name and the associated SELECT statement. For example:

```
CURSOR c_customers IS
   SELECT id, name, address FROM customers;
```

Opening the Cursor

Opening the cursor allocates memory for the cursor and makes it ready for fetching the rows returned by the SQL statement into it. For example, we will open above-defined cursor as follows:

```
OPEN c_customers;
```

Fetching the Cursor

Fetching the cursor involves accessing one row at a time. For example we will fetch rows from the above-opened cursor as follows:

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

Closing the Cursor

Closing the cursor means releasing the allocated memory. For example, we will close above-opened cursor as follows:

```
CLOSE c_customers;
```

Example:

Following is a complete example to illustrate the concepts of explicit cursors:

```
DECLARE
   c_id customers.id%type;
   c_name customers.name%type;
   c_addr customers.address%type;
   CURSOR c_customers is
      SELECT id, name, address FROM customers;
BEGIN
   OPEN c_customers;
   LOOP
      FETCH c_customers into c_id, c_name, c_addr;
      EXIT WHEN c_customers%notfound;
      dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
   END LOOP;
   CLOSE c_customers;
END;
/
```

When the above code is executed at SQL prompt, it produces the following result:

```
1 Ramesh Ahmedabad
2 Khilan Delhi
3 kaushik Kota
4 Chaitali Mumbai
5 Hardik Bhopal
6 Komal MP

PL/SQL procedure successfully completed.
```

```
create table customers(id int primary key,name char(20),age int,addr varchar2(30),sal real);
SQL> insert into customers values(&id,'&name',&age,'&address',&salary);
```

```
Enter value for id: 2

Enter value for name: khilan

Enter value for age: 25

Enter value for address: delhi

Enter value for salary: 1500.00

old      1: insert into customers values(&id,'&name',&age,'&address',&salary)

new      1: insert into customers values(2,'khilan',25,'delhi',1500.00)


1 row created.


SQL> /

Enter value for id: 3

Enter value for name: kaushik

Enter value for age: 23

Enter value for address: kota

Enter value for salary: 2000.00

old      1: insert into customers values(&id,'&name',&age,'&address',&salary)

new      1: insert into customers values(3,'kaushik',23,'kota',2000.00)


1 row created.


SQL> /

Enter value for id: 4

Enter value for name: chaitali

Enter value for age: 25

Enter value for address: mumbai

Enter value for salary: 6500.00

old      1: insert into customers values(&id,'&name',&age,'&address',&salary)
```

```
new      1: insert into customers values(4,'chaitali',25,'mumbai',6500.00)

1 row created.

SQL> /

Enter value for id: 5

Enter value for name: hardik

Enter value for age: 27

Enter value for address: bhopal

Enter value for salary: 8500.00

old      1: insert into customers values(&id,'&name',&age,'&address',&salary)

new      1: insert into customers values(5,'hardik',27,'bhopal',8500.00)


1 row created.


SQL> /

Enter value for id: 6

Enter value for name: komal

Enter value for age: 22

Enter value for address: mp

Enter value for salary: 4500.00

old      1: insert into customers values(&id,'&name',&age,'&address',&salary)

new      1: insert into customers values(6,'komal',22,'mp',4500.00)


1 row created.


SQL> select * from customers;


        ID NAME                        AGE ADDRESS                        SALARY

---------- -------------------- ---------- ------------------------- ----------
```

```
        2 khilan                    25 delhi                        1500
        3 kaushik                   23 kota                         2000
        4 chaitali                  25 mumbai                       6500
        5 hardik                    27 bhopal                       8500
        6 komal                     22 mp                           4500
        1 ramesh                    32 ahmedabad                    2000

6 rows selected.

SQL> update customers
  2   set id=7
  3   where name='ramesh';

1 row updated.

SQL> select * from customers;

        ID NAME                     AGE ADDRESS                     SALARY
---------- -------------------- ---------- ------------------------- ----------
        2 khilan                    25 delhi                        1500
        3 kaushik                   23 kota                         2000
        4 chaitali                  25 mumbai                       6500
        5 hardik                    27 bhopal                       8500
        6 komal                     22 mp                           4500
        7 ramesh                    32 ahmedabad                    2000

6 rows selected.
```

```
SQL> update customers set id=1
  2  where name='ramesh';

1 row updated.

SQL> select * from customers;

        ID NAME                 AGE ADDRESS                   SALARY
---------- -------------------- ---------- ------------------------- ----------
         2 khilan                25 delhi                      1500
         3 kaushik               23 kota                       2000
         4 chaitali              25 mumbai                     6500
         5 hardik                27 bhopal                     8500
         6 komal                 22 mp                         4500
         1 ramesh                32 ahmedabad                  2000

6 rows selected.

SQL> select id
  2  from customers
  3  order by id;

        ID
----------
         1
         2
         3
         4
         5
         6

6 rows selected.

SQL>   select *
  2     from customers
  3     order by id;

        ID NAME                    AGE ADDRESS                    SALARY
---------- ------------------- ---------- ------------------------ ----------
         1 ramesh                   32 ahmedabad                  2000
         2 khilan                   25 delhi                      1500
         3 kaushik                  23 kota                       2000
         4 chaitali                 25 mumbai                     6500
         5 hardik                   27 bhopal                     8500
         6 komal                    22 mp                         4500

6 rows selected.

SQL> set serveroutput on
SQL> declare
  2   total_rows number(2);
  3   begin
  4   update customers
  5   set salary=salary+500;
  6   if sql%notfound then
  7   dbms_output.put_line('no customers selected');
  8    elsif    sql%found then
  9    total_rows:=sql%rowcount;
 10      dbms_output.put_line(total_rows||'customers selected');
 11    end if;
 12    end;
 13    /
6customers selected

PL/SQL procedure successfully completed.

SQL> select * from customers;

        ID NAME                 AGE ADDRESS                       SALARY
---------- -------------------- ---------- ------------------------- ----------
         2 khilan                25 delhi                   2000
         3 kaushik               23 kota                    2500
         4 chaitali              25 mumbai                  7000
         5 hardik                27 bhopal                  9000
         6 komal                 22 mp                      5000
         1 ramesh                32 ahmedabad               2500

6 rows selected.


SQL> declare
  2    c_id customers.id%type;
  3    c_name customers.name%type;
  4    c_addr customers.address%type;
```

```
cursor c_customers is
select id,name,address from customers;
begin
open c_customers;
loop
fetch c_customers into c_id,c_name,c_addr;
exit when c_customers%notfound;
  dbms_output.put_line(c_id||"||c_name||"||c_addr);
end loop;
close c_customers;
end;
/
2khilan                    delhi
3kaushik                   kota
4chaitali                mumbai
5hardik                    bhopal
6komal                       mp
1ramesh                     ahmedabad

PL/SQL procedure successfully completed.

SQL> commit;

Commit complete.
```

Experiment:13

AIM:To perform join in SQL

Join in SQL

SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables. **Join** Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is**(n-1)** where **n**, is number of tables. A table can also join to itself known as, **Self Join**.

Types of Join

The following are the types of JOIN that we can use in SQL.

- Inner
- Outer
- Left
- Right

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT  column-name-list
from  table-name1
CROSS  JOIN
table-name2;
```

Example of Cross JOIN

The **class** table,

ID	NAME
1	abhi

2	adam
4	alex

The **class_info** table,

ID	**Address**
1	DELHI
2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

```
SELECT *
 from class,
 cross JOIN class_info;
```

The result table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI

1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the query.

Inner Join Syntax is,

```
SELECT column-name-list
from table-name1
INNER JOIN
table-name2
WHERE table-name1.column-name = table-name2.column-name;
```

Example of Inner JOIN

The **class** table,

ID	**NAME**
1	abhi
2	adam
3	alex
4	anu

The **class_info** table,

ID	**Address**

1	DELHI
2	MUMBAI
3	CHENNAI

Inner JOIN query will be,

```
SELECT * from class, class_info where class.id = class_info.id;
```

The result table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

Natural Join Syntax is,

```
SELECT *
from table-name1
NATURAL JOIN
table-name2;
```

Example of Natural JOIN

The **class** table,

ID	**NAME**

1	abhi
2	adam
3	alex
4	anu

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Natural join query will be,

```
SELECT * from class NATURAL JOIN class_info;
```

The result table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have ID column(same name and same datatype), hence the records for which value of ID matches in both the tables will be the result of Natural Join of these two tables.

Outer JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join

The left outer join returns a result table with the **matched data** of two tables then remaining rows of the **left** table and null for the **right** table's column.

Left Outer Join syntax is,

```
SELECT column-name-list
from table-name1
LEFT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

Left outer Join Syntax for **Oracle** is,

```
select column-name-list
from table-name1,
table-name2
on table-name1.column-name = table-name2.column-name(+);
```

Example of Left Outer Join

The **class** table,

ID	**NAME**
1	abhi
2	adam
3	alex
4	anu

5	ashish

The **class_info** table,

ID	**Address**
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Left Outer Join query will be,

```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);
```

The result table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

Right Outer Join

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.

Right Outer Join Syntax is,

```
select column-name-list
from table-name1
RIGHT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

Right outer Join Syntax for **Oracle** is,

```
select column-name-list
from table-name1,
table-name2
on table-name1.column-name(+) = table-name2.column-name;
```

Example of Right Outer Join

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	Address

1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

```
SELECT * FROM class RIGHT OUTER JOIN class_info on (class.id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a result table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Full Outer Join Syntax is,

```
select column-name-list
from table-name1
FULL OUTER JOIN
```

```
table-name2
on table-name1.column-name = table-name2.column-name;
```

Example of Full outer join is,

The **class** table,

ID	**NAME**
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	**Address**
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info on (class.id=class_info.id);
```

The result table will look like,

ID	**NAME**	**ID**	**Address**
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT