(A1547) DATABASE MANAGEMENT SYSTEMS LAB

B.Tech IV Sem

L	T	P	C
0	0	3	2

Objectives:

- To teach the student database design and query and PL/SQL.
- To get the Knowledge on Normalization
- To get the Knowledge on Data Integrity

Experiment 1: Working with ER Diagram and Normalization

Example: ER Diagram for Sailors Database

Entities:

1. Sailor (sid,sname,rating,age)

2. Boat (bid,bname,bcolour)

Relationship: Reserves(sid,bid,date)

Primary Key Attributes:

1. SID (Sailor Entity)

2. BID (Boat Entity)

Experiment 2: Working with DDL, DML, DCL and Key Constraints

Creation, Altering and Dropping of Tables and Inserting Rows into a Table (Use Constraints While Creating Tables) Examples Using Select Command.

Experiment 3: Working with Queries and Nested QUERIES

Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSET, Constraints

Experiment 4: Working with Queries USING Aggregate Operators & views

Queries using Aggregate Functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and Dropping of Views

Experiment 5: Working with Conversion Functions & String Functions

Queries using Conversion Functions (to_char, to_number and to_date), String Functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), Date Functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)

Experiment 6: Working with Triggers using PL/SQL

Develop Programs using BEFORE and AFTER Triggers, Row and Statement

Triggers and INSTEAD OF Triggers

Experiment 7: Working with PL/SQL Procedures

Programs Development using Creation of Procedures, Passing Parameters IN and OUT of PROCEDURES

Experiment 8: Working with LOOPS using PL/SQL and Exception Handling

Program Development using WHILE LOOPS, Numeric FOR LOOPS, Nested Loops using ERROR Handling, BUILT-IN Exceptions, USE Defined Exceptions, RAISE- APPLICATION ERROR

Experiment 9: Working with Functions Using PL/SQL

Program Development using Creation of Stored Functions, Invoke Functions in SQL Statements and Write Complex Functions.

Experiment 10: Working with CURSORS

Develop Programs using Features Parameters in a CURSOR, FOR UPDATE CURSOR, WHERE CURRENT of Clause and CURSOR Variables

Outcomes: After the completion of the course, the students would be able to:

- Ability to working DDL,DML,DCL Commands
- Ability to normalize database
- Ability to working with GUI

Textbooks:

1. Oracle PL/SQL by Example, Benjamin Rosenzweig, Elena Silvestrova, Pearson Education 3rd Edition
2. Oracle Database LogG PL/SQL Programming, Scott Urman, Tata Mc-Graw Hill.
3. SQL and PL/SQL for Oracle 10g, Black Book, Dr .P.S. Deshpande.

Experiment-1

ER Diagram for Sailors Database

The goal of the "BoatClub" database is to enable members of a boat club to reserve boats for trips lasting several hours.

The two major entities are:

- Sailors—members of the boat club who reserve boats; and
- Boats—boats in the club's inventory.

In this problem we need to know what boats are reserved by what sailors on a given day. Thus, "reservation" is obviously an important relationship in this simple problem.

Attributes of the Sailor Entity

Attribute	Description
SID	A sailor—each sailor is assigned a unique ID
name	The sailor's name
rating	The sailor's rating, ranging from 1 (low) to 10 (high)
age	The sailor's age

Attributes of the Boat Entity

Attribute	Description
BID	A boat ID—each boat is assigned a unique ID (painted on the bow)
name	The name of the boat (also painted on the bow)
color	The color of the boat

A sailor can make many reservations (*) but a reservation involves only a single sailor. Similarly, a boat can be allocated to many reservations, but only one boat is allocated to a particular reservation.

Q1. Draw an ER diagram that captures the above information.

Q2.Convert above ER diagram in to relations (tables)
Ans:-

Q3. Write SQL statements to create above relations (tables).

Ans:-

create table Sailors(sid varchar2(20) primary key,name varchar2(20),rating number(10) check(rating between 1 and 10),age number(3))

create table boats1(bid varchar2(20) primary key,name varchar2(20),colour varchar2(10))

create table reserves3(sid varchar2(20) not null,bid varchar2(20) not null,day date,primary key(sid,bid),foreign key(sid) references sailors(sid),foreign key(bid) references boats1(bid));

Q4.Insert the following data into above created tables.
Ans:-
Insert into sailors(22,'Dustin',7,45);
1row created

Insert into sailors(29,'Brutus',1,33);
1row created

Insert into sailors(31,'Lubbur',8,55.5);
1row created

Insert into sailors(32,'Andy',8,25.5);
1row created

Insert into sailors(58,'Rusty',10,35);
1row created

Insert into sailors(64,'Horatio',7,35);
1row created

Insert into sailors(22,'Dustin',7,45);
1row created

Select * from sailors;
Select *from Boat;

Select *from Reserve;

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

Boats

Bid	Bname	Color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

SID	BID	Day
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	10-11-1998
31	103	06-11-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998

Q5.Write the following Queries in SQL

5.1Find the names and ages of all sailors.
Ans:- select sname,age from sailor

Output :

SNAME	AGE
dustin	45
brutus	56
lubber	56
andy	26
busty	35
rusty	35
zordo	16
horatio	35
art	26
bob	64

10 rows returned in 0.00 seconds

5.2 Find all sailors with a rating above 7.
Ans:- select * from sailor where rating>7;

Output:-

S_ID	SNAME	RATING	AGE
31	lubber	8	56
32	andy	8	26
58	busty	10	35
64	rusty	10	35
71	zordo	10	16
74	horatio	9	35

6 rows returned in 0.00 seconds

CSV Export

5.3Find the names of sailors who have reserved boat number 103.

Ans:- select s.sname from sailor s, reserves r where s.s_id=r.sid and r.bid=103;

Output:

SNAME
Dustin
Lubber
Horatio

3 rows returned in 0.00 seconds

5.4 Find the *sids* of sailors who have reserved a red boat.

Ans:- select s_id from sailor,reserves,boat where s_id=sid and bid=b_id and b_colour='red'

Output:-

S_ID
22
22
31
31
64

4 rows returned in 0.03 seconds

5.5 Find the names of sailors who have reserved a red boat.

Ans:- select sname from sailor,reserves,boat where s_id=sid and bid=b_id and b_colour='red'

Output:-

SNAME
Dustin
Dustin
Lubber
Lubber
Rusty

5 rows returned in 0.00 seconds

5.6 Find the colors of boats reserved by Lubber.
Ans:- select b_colour from sailor,reserves,boat where s_id=sid and bid=b_id and sname='lubber'

Output:-

B_COLOUR
Red
Green
Red

3 rows returned in 0.00 seconds

5.7 .Find the names of sailors who have reserved at least one boat.

Ans:- select sname from sailor,reserves,boat where s_id=sid and bid=b_id

Output:-

SNAME
Dustin
Dustin
Dustin
Dustin
Lubber
Lubber
Lubber
Rusty
Rusty
Horatio

10 rows returned in 0.00 seconds

5.8 Compute increments for the ratings of persons who have sailed two different boats on the same day.

Ans:- select sname,rating+1 as rating from sailor,reserves r1,reserves r2 where s_id=r1.sid and r1.sid=r2.sid and r1.bid<>r2.bid and r1.day=r2.day

Output:-

SNAME	RATING
dustin	8
dustin	8

2 rows returned in 0.00 seconds

5.9.Find the ages of sailors whose names begins and ends with the B and

has at least three characters.

Ans:- select age from sailor where sname like 'b_%b';

Output:-

AGE
64

1 rows returned in 0.00 seconds

EXPERIMENT 3:

Experiment-2

Creation, Altering and Dropping of Tables and Inserting Rows into a Table (Use Constraints While Creating Tables)

Examples Using Select Command.

Q.1: login into Oracle data base using ***system*** user and create a user with your ***Roll no*** and grant permission to create the tables in SQL.
Ans: Create user <username> identified by <password>
Grant dba to <username>

Q.2 login into oracle data base using your roll no and create the following tables.

1. Sailors (sid:integer)
2. Boats (bid:integer);
3. Reserves (sid:integer,bid:integer,day:date);

Ans:

create table sailor2(sid number(10))

create table boats2(bid number(10));

create table reserves5(sid number(10),bid number(10),day date)

Write the following DQL statements.

1. Add new column sailor name and rating to sailors table.

Ans: alter table sailor2 add(name varchar2(10),rating number(10))

2.Add new column boat name and color to Boats table.

Ans:- alter table boats2 add(name varchar2(20),colour varchar2(20));

3.Add a primary key to sailors table after table creation

Ans:- alter table sailor2 add primary key(sid);

4. Add a primary key to Boats table after table creation

Ans:- alter table boats2 add primary key(bid);

5. Remove a Primary Key from Sailors table

Ans:- alter table sailor2 drop primary key;

6. Remove a Primary Key from Boats table

Ans:- alter table boats2 drop primary key

7. Add a not null constrain on Sailors, Boats Table

Ans:- alter table sailor2 modify(name not null);
alter table boats2 modify(name not null);

8. Drop not null constraints from Sailors, Boats Table.

Ans:- alter table sailor2 modify(name null);

alter table boats2 modify(name null);

9. Add a constraint check to the rating column
Ans:- alter table sailor2 add constraint ck_ra_s check

(rating between 1 and10)

10. Drop the above check constraint from rating column.
Ans:- alter table sailor2 drop constraint ck_ra_s;

11. Add a primary key constraint on Sailors, Boats tables.
Ans:- alter table sailor2 add constraint pk_sid_s primary key(sid);

alter table boats2 add constraint pk_bid_s primary key(bid);

12. Drop primary key constraints from sailors ,Boats tables.
Ans:- alter table sailor2 drop constraint pk_sid_s;

alter table boats2 drop constraint pk_bid_s ;

13. Add foreign key constraints to reserves table.
Ans:- alter table reserves5 add constraint fr_sid_s foreign key(sid) references sailor2(sid)

14. Drop foreign key constraints from reserve table

Ans:- alter table reserves5 add constraint fr_bid_s foreign key(bid) references boats2(bid)

Experiment 3

Queries (along with sub Queries) using ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSET, Constraints

Create the following relations (tables) and write the following queries.

Q1.Find the names of sailors who have

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

Boats

Bid	Bname	Color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

SID	BID	Day
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	10-11-1998
31	103	06-11-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998

reserved a red or a green boat.

Ans: SELECT distinct S.sname FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor = 'red' union SELECT distinct S.sname FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor='green';

Q2. Find the names of sailors who have
reserved a red and a green boat.

Ans: SELECT distinct S.sname FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor = 'red' intersect
SELECT distinct S.sname
FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor='green';

Q3. Find the names of sailors who have reserved

a red but not green boats.

SELECT S.sid
FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor = 'red' minus
SELECT S.sid FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor='green';

Q4.Find all sids of sailors who have a rating of

10 or reserved boat 104.

Ans: select distinct sid from sailors where rating=10 union select distinct S.sid from sailors S,reserve R, boats B where S.sid=R.sid AND R.bid = B.bid and b.bid=104;

Q5.Find the names of sailors who have reserved boat 103
using independent nested query.

Ans: select distinct S.sname from sailors S,reserve R, boats B where S.sid=R.sid AND R.bid = B.bid and b.bid=103;

(OR)nested query=

```
SELECT S.sname FROM Sailors S
WHERE S.sid IN (SELECT R.sid  FROM Reserve RWHERE R.bid IN(SELECT
B.bid  FROM  Boats B  WHERE B.bid =103));
```

Q6.Find the names of sailors who have reserved a red boat.

```
Ans:      SELECT distinct S.sname
FROM Sailors S, Reserve R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.bcolor  = 'red';
```

Q7.Find the names of sailors who have not reserved a red boat.

```
Ans:    SELECT  S.sname   FROM Sailors S
WHERE S.sid not IN (SELECT R.sid  FROM Reserve R WHERE R.bid
IN(SELECT B.bid FROM Boats B WHERE B.bcolor ='red'));
```

Q8.Find the names of sailors who have reserved boat number 103 using correlated nested query.

Ans: select S.sname from sailors S where EXISTS (select *from reserve R where R.sid=S.sid and R.bid =103);

Q9.Find sailors whose rating is better than some sailor called ‘Horatio’.

Ans: select *from sailors S where S.rating=ANY(select rating from sailors where sname='horatio');

Q10.Find the sailors with the highest rating.

Ans: select *from sailors where rating>= ALL(select rating from sailors);

Q11.Find the names of sailors who have reserved both a red and a green boat using nested queries.

Ans: select S.sname from sailors S where sid IN (select sid from reserve R,boats B where B.bcolor='red' and R.bid=B.bid) INTERSECT (select S.sname from sailors S where S.sid IN (select R.sid from reserve R,boats B,sailors S where R.sid=S.sid and B.bid=R.bid and B.bcolor='green'));

Q12.Find the names of sailors who have reserved all boats.

Ans: select S.sname from sailors S where not exists ((select bid from boats) MINUS (select R.bid from reserve R where R.sid=S.sid));

Experiment 4

Queries using Aggregate Functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and Dropping of Views

Create the following relations (tables) and write the following queries.

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

Boats

Bid	Bname	Color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Reserves

SID	BID	Day
22	101	10-10-1998
22	102	10-10-1998
22	103	10-08-1998
22	104	10-07-1998
31	102	10-11-1998
31	103	06-11-1998
31	104	11-12-1998
64	101	09-05-1998
64	102	09-08-1998
74	103	09-08-1998

Q1.Find the average age of all sailors.

Ans .select avg(age) from sailors

Q2Find the average age of sailors with a rating of 10.

Ans .select avg(age) from sailors where rating=10

Q3.Find the name and age of the oldest sailor.

Ans .select sname, age from sailors where age=(select max(age) from sailors)

Q4.Count the number of sailors.

Ans . select count(sid) from sailors

Q5.Count the names of different sailor names.

Ans .select count(distinct sname) from sailors

Q6.Find the names of sailors who are older than the oldest sailor with a rating of10.

Ans .select s.sname from sailors s where s.age>(select max(s2.age) from sailors s2 where s2.rating=10)

Q7.Find the age of the youngest sailors for each rating level.

Ans .select rating, min(age) from sailors group by rating

Q8.Find the age of the youngest sailor who is eligible to vote for each rating level with at least two such sailors.

Ans .select min(age) from sailors where age>=18 group by rating having count (*)>1

Q9.For each red boat, find the number of reservations for this boat.

Ans .select r.bid, count(*) from reserves r, boats b where r.bid=b.bid AND b.bcolor='Red' group by r.bid

Q10.Find the average age of sailors for each rating level that has at least two sailors.

Ans .select rating, avg(age) from sailors group by rating having count

(*)>1

Q11.Find the average age of sailors who are of voting age for each rating level that has at least two sailors.

Ans .select rating, avg(age) from sailors where age>=18 group by rating having count
(*)>1

Q12.Find those ratings for which the average age of sailors is the minimum overall ratings.

Ans .select t.rating, t.avgage from (select s.rating,avg(s.age) as avgage from sailors s group by s.rating)t where
t.avgage =(select min(t.avgage) from t)

Q13.Define a view for finding sailors whose rating is above7. .Insert some rows into above view.

Ans .create view v1 as (select * from sailors where rating>7)
select * from v1

Q.14.Restrict updates on above view.

Ans .insert into v1
values (7,'Harsh',7,18)
select * from sailors

Q15.Drop the view created in

Ans create or replace view v1 (Select * from sailors where rating > 7)with Read

Only

16.Drop view v1

Experiment 5

Queries using Conversion Functions (to_char, to_number and to_date), String Functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), Date Functions (Sysdate, next_day, add_months, last_day, months_between, least, greatest, trunc, round, to_char, to_date)

Q1.Create the following table:

Table Name::Staff Record

Staff_ID	Name	DOB	Sex	Salary	Award	District	Department
1001	Jeffrey Lee	23/02/1978	M	28463.40	3	Tai Kok Tsui	Sales
1002	Hugo Cheung	08/04/1976	M	14598.50	2	Central	Sales
1003	Jennifer Wong	29/03/1978	F	39850.00	6	Tai Po	Sales
1004	Melinda Ma	28/08/1982	F	7783.00	6	Tai Po	Purchase
1005	Hilda Leung	24/10/1982	F	45670.50	2	Westren	Sales
1006	Nelly Tam	10/10/1973	F	4530.80	4	Shatin	Sales
1007	Mable Mee	30/08/1979	F	3549.40	1	Tai Kok Tsui	Purchase
1008	Barnabv Nge	12/05/1980	M	8327.30	5	Hunghom	Account
1009	Luaretta Tai	23/09/1982	F	32445.42	3	Tai Wai	Account
1010	Gregory tai	22/10/1972	M	35542.40	4	Tai Wo	Purchase

Write the following SQL queries

1.Write a SQL statement to produce a list of male staff only, showing their names in upper case, department in
lower case and the number of characters in the department.

Ans:- select upper(staffname),lower(dept),length(dept) from staff where sex='m';

2. Write a SQL statement to produce a list of all staff member with their names appended with first letter of their
Department.
Ex Jeffrey Lee (S)

Ans:- select concat(concat(staffname,'('), concat(substr(dept,1,1),')')) from staff

3. Write a SQL statement to print a list of first names of staff
Ex. Jeffrey

Ans:- select substr(staffname,1,instr(staffname,' ')) from staff

4. Write a SQL statement to print a list of districts that consists of a single word.The list should not consist of
repeating items and is arranged in descending alphabetical order.
Ex Central

Ans:-select distinct district from staff where instr(district,' ')=0 order by district desc;

5. Given that the bonus of staff is calculated by
Bonus=SQRT(Salary* Award)
Write a SQL statement to print a list of salary and bonus for each staff.
Ans:- select staffname,salary, to_char(sqrt(salary*award),'$999.99') as bonus from staff

Experiment 6: Working with Triggers using PL/SQL

Develop Programs using BEFORE and AFTER Triggers, Row and Statement Triggers and INSTEAD OF Triggers

TRIGGER

A Trigger is PL/SQL block which can be executed without explicit calling at the time of DML operations on a table.

- Types of Triggers:

Triggers are categorized base on when they are fired:

- Before
- After
- For each row
- For each statement(default)

Syntax:

Create or replace trigger <trigger name> {before/after/instead of} {Insert/delete/update [of column1 [, column2...]]} on <table name/view name > [for each statement/ row] [when <condition>]

DECLARE
<Declarations>
BEGIN

<Executable statements>

END;

Write a trigger to ensure that dept table doesnot contain duplicate or null values in deptno.

```
create or replace trigger trig1 before insert on dept for each row

declare

        a number;
BEGIN
        if(:new.deptno is Null) then
                raise_application_error(-20001,'error::deptno cannot be null');
        else
                select count(*) into a from dept where deptno=:new.deptno;
        if(a=1) then
                raise_application_error(-20002,'error:: cannot have duplicate
                deptno');
        end if;
        end if;
END;
```

/

OUTPUT:

SQL> @trigger

Trigger created.

SQL> select * from dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL> insert into dept values(&deptnp,'&dname','&loc');

Enter value for deptnp: null

Enter value for dname: marketing

Enter value for loc: hyd

```
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(null,'marketing','hyd')
insert into dept values(null,'marketing','hyd')
            *
ERROR at line 1:
ORA-20001: error::deptno cannot be null
ORA-06512: at "SCOTT.TRIG1", line 5
ORA-04088: error during execution of trigger 'SCOTT.TRIG1'
SQL> /
Enter value for deptnp: 10
Enter value for dname: manager
Enter value for loc: hyd
old   1: insert into dept values(&deptnp,'&dname','&loc')
new   1: insert into dept values(10,'manager','hyd')
insert into dept values(10,'manager','hyd')
            *
ERROR at line 1:
ORA-20002: error:: cannot have duplicate deptno
ORA-06512: at "SCOTT.TRIG1", line 9
```

ORA-04088: error during execution of trigger 'SCOTT.TRIG1'

SQL> /

Enter value for deptnp: 50

Enter value for dname: MARKETING

Enter value for loc: HYDERABAD

old 1: insert into dept values(&deptnp,'&dname','&loc')

new 1: insert into dept values(50,'MARKETING','HYDERABAD')

1 row created.

SQL> select * from dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

50 MARKETING HYDERABAD

Experiment 7: Working with PL/SQL Procedures

Programs Development using Creation of Procedures, Passing Parameters IN and OUT of PROCEDURES

SUBPROGRAMS

A subprogram is used to reduce the program in size. We use the same set of statements to be executed for the different values, so we write the same set of statements as a subprogram.

A subprogram can be a function or procedure in PL/SQL which has to be created separately as a database object.

- **FUNCTION**

Basically function is to return a value of various data types available. To get the function result we have to assign function to returntype variable.
Function can't return more than one value.

Example: Create or replace function add (a number, b number)return number is

C number;

```
Begin
        C: =a+b;
        Return C;
End;
```

- **PROCEDURE**

Basically a procedure is used to perform some task which may return one value or more than one value or no value. we use parameters to get the result of procedures.

- MODES OF PARAMETER:

IN	Read only value	Right side of: =
	Literal, expression, variable	
OUT	Write only value	Left side of: =
	Only UN initialized variable.	
IN OUT	Read Write Value	Left & Right side of := Only Initialized variable

Example: Create or replace procedure add (a in number, b in number, c out number) is

```
Begin
        C: =a+b;
End;
```

- **PROCEDURE PROGRAM**

```
set serveroutput on;
Create or replace procedure factor (n in number out number)
        f number (4):=1;
        i number(2)
        as
        begin
         for i in 1...n loop
                  f:=f*i;
        end loop;
        declare
                a number(3):=&enterano;
                b number(3):=&enterbno;
                npr number(4);
                x number(4);
                y number(4);
```

```
begin
        factor(a,x)
        c:=a-b;
        factor(c,y)
        npr=x/y;
end;
```

Experiment 8: Working with LOOPS using PL/SQL and Exception Handling

Program Development using WHILE LOOPS, Numeric FOR LOOPS, Nested Loops using ERROR Handling, BUILT-IN Exceptions, USE Defined Exceptions, RAISE- APPLICATION ERROR

Control Structures:

1. If <condition> then
   <Statements>
   End if;

- **Eg. Write a PL/SQL block to check the given number is Even or Odd**

```
DECLARE
        num number(5);
        rem number;
BEGI
num:=&num;
        rem:=mod(num,2);
         if rem=0
         then
                dbms_output.put_line(' Number '||num||' is Even');
        else
                dbms_output.put_line(' Number '||num||' is Odd');
        end if;
END;

/
```

OUTPUT:

```
SQL>start even

Enter value for num: 6

old   5:   num:=&num;

new   5:   num:=6;

Number 6 is Even

PL/SQL procedure successfully completed.

SQL> /
```

```
Enter value for num: 3

old   5:   num:=&num;

new   5:   num:=3;

Number 3 is Odd


PL/SQL procedure successfully completed.
```

2. If <condition> then

```
        <Statements>
elsif <condition> then
        <Statements>
else
   <Statements>
End if;
```

- **Eg. Write a PL/SQL Program to find the Maximum of given three numbers.**

```
Declare
        a number;
        b number;
        c number;
Begin
       a := &a;
       b := &b;
```

```
c :=&c;
if(a>b and a>c) then
        dbms_output.put_line('the maximum is'||a);
elsif(b>a and b>c) then
        dbms_output.put_line('the maximum is'||b);
else
        dbms_output.put_line('maximum is'||c);
end if;
END;
```

/Loops:

1. GOTO:

```
<<lablename>>
<Statements>
go to lablename;
```

2. Simple loop:

```
Loop
        <Statements>
        exit <condition>
End loop
```

3. For loop:

```
For <counter> in <start value>... <End value>
Loop
        <Statements>
```

End loop

- **Eg.Write a PL/SQL Program to generate the first five natural numbers.**

```
Declare
        I  Num;
Begin
    for I in 1..5
    loop
        dbms_output.put_line(i);
    end loop;
END;
 /
```

Output:

1
2
3
4
5

4. While loop:

While <condition>
Loop

```
        <Statements>
    End loop;
```

- **Write a PL/SQL block to Generate Fibonacii Series**

```
DECLARE
    num number(5);
    f1 number(5):=0;
    f2 number(5):=1;
    f3 number(5);
    i number(5):=3;
BEGIN
    num:=&num;
    dbms_output.put_line('THE FIBONACCI SERIES IS:');
    dbms_output.put_line(f1);
    dbms_output.put_line(f2);
    while(i<=num)
    loop
        f3:=f1+f2;
        dbms_output.put_line(f3);
        f1:=f2;
        f2:=f3;
        i:=i+1;
    end loop;
END;
```

/

OUTPUT:

```
SQL> start fib
Enter value for num: 10
old   8:  num:=&num;
new   8:  num:=10;
THE FIBONACCI SERIES IS:
0
1
1
2
3
5
8
13
21
34
```

- PL/SQL procedure successfully completed.

- **Write a PL/SQL Program to insert records into a student table**

```
/* Inserting students marks into table */
declare
    a stu_marks.sno%type:=&enterno;
    b  stu_marks.sname%type:='&name';
    c stu_marks.markstype:=&marks;
begin
    insert into stu_marks values(a,b,c);
commit;
end
```

PL/SQL block for INSERTING ROWS INTO EMPDET TABLE WITH THE Following Calculations:

HRA=50% OF BASIC

DA=20% OF BASIC

PF=7% OF BASIC

NETPAY=BASIC+DA+HRA-PF

```
declare
   eno1 empdet.eno%type;
   ename1 empdet.name%type;
   deptno1  empdet.deptno%type;
   basic1 empdet.basic%type;
   hra1 empdet.hra%type;
   da1 empdet.da%type;
   pf1 empdet.pf%type;
   netpay1 empdet.netpay%type;
BEGIN
```

```
        eno1:=&eno1;
        ename1:='&ename1';
        deptno1:=&deptno1;
        basic1:=&basic1;
        hra1:=(basic1*50)/100;
        da1:=(basic1*20)/100;
        pf1:=(basic1*7)/100;
        netpay1:=basic1+hra1+da1-pf1;
        insert into empdet
        values(eno1,ename1,deptno1,basic1,hra1,da1,pf1,netpay1);
    END;
    /
```

OUTPUT:

```
sql> @basic
enter value for eno1: 104
old  11: eno1:=&eno1;
new  11: eno1:=104;
enter value for ename1: srinivas reddy
old  12: ename1:='&ename1';
new  12: ename1:='srinivas reddy';
enter value for deptno1: 10
old  13: deptno1:=&deptno1;
```

new 13: deptno1:=10;

enter value for basic1: 6000

old 14: basic1:=&basic1;

new 14: basic1:=6000;

pl/sql procedure successfully completed.

Working with Functions Using PL/SQL

Program Development using Creation of Stored Functions, Invoke Functions in SQL Statements and Write Complex Functions.

- **FUNCTION**

Basically function is to return a value of various data types available. To get the function result we have to assign function to returntype variable.
Function can't return more than one value.

Example: Create or replace function add (a number, b number)return number is

```
C number;
Begin
    C: =a+b;
    Return C;
End;
```

```
SQL> CREATE OR REPLACE FUNCTION student_details_func
 2  RETURN VARCHAR(20);
```

```
3 IS
4 student_name VARCHAR(20);
5 BEGIN
6 SELECT sname INTO student_name FROM student WHERE sid = 100;
7 RETURN emp_name;
8 END;
9 /
```

OUTPUT

Function created sucessfully.

```
SQL> Create or replace function get_sal
2 (v_id in emp.empno%type) return number is v_salary emp.sal % type: =0;
3
4 begin
5 select sal into v_salary from emp where
6 empno=v_id;
7 return(v_salary);
8 end get_sal;
9 /
```

OUTPUT

Function created sucessfully.

Experiment 10: Working with CURSORS

Develop Programs using Features Parameters in a CURSOR, FOR UPDATE CURSOR, WHERE CURRENT of Clause and CURSOR Variables

CURSORS:

Oracle uses work areas to execute SQL statements and store processing information. A PL/SQL construct called cursor lets us you name a work area and access its stored information. There are 2 types of cursors.

- IMPLICIT CURSOR:
  Is the cursor which is always generated?
  Implicitly by the oracle. We can use the implicit cursor as SQL, which is its name.

- EXPLICIT CURSORS
  These cursors are always defined by the user in PL/SQL block with the following syntax.

```
DECLARE
        CURSOR <Cursor name> IS <Select statement>;
BEGIN
        < Select statements>;
END;
```

Write a Cursor to display List of Employees from Emp Table in PL/SQL block

```
DECLARE
        cursor c is select empno,ename,deptno,sal from emp ;
        i emp.empno%type;
        j emp.ename%type;
        k emp.deptno%type;
        l emp.sal%type;
BEGIN
        open c;
        dbms_output.put_line('Empno,name,deptno,salary of employees are:=
        ');
        loop
        fetch c into i,j,k,l;
        exit when c%notfound;
        dbms_output.put_line(i||' '||j||' '||k||' '||l);
        end loop;
        close c;
END;
```

OUTPUT:

```
SQL> @EMP
Empno,name,deptno,salary of employees are:=
```

7369	SMITH	20	800
7499	ALLEN	30	1600
7521	WARD	30	1250
7566	JONES	20	2975
7654	MARTIN	30	1250
7698	BLAKE	30	2850
7782	CLARK	10	2450
7788	SCOTT	20	3000
7839	KING	10	5000
7844	TURNER	30	1500
7876	ADAMS	20	1100
7900	JAMES	30	950
7902	FORD	20	3000
7934	MILLER	10	1300

PL/SQL procedure successfully completed.

PARAMETRIC CURSORS:

Parameterized cursor is used to give parameters while opening the cursor. These parameters are to substitute in cursor declaration.

We use OPEN, FETCH, CLOSE statements to control a Cursor.

- **CURSOR ATTRIBUTES:**

  %FOUND
  %NOTFOUND
  %ISOPEN
  %ROWCOUNT

  Using the above attributes we can know the different states of cursor.

- **CURSOR FOR LOOP**:

```
FOR VARIABLE IN CURSOR
LOOP
    <Statements>
END LOOP;
```

- **FOR UPDATE CLAUSE:**

Is used to restrict update on particular columns by selecting those columns in for update clause.