Note :I am not responsible for any mistakes in this document

concerning

1. If you can't understand my hand writing please verify the DBMS record pdf given by college

WRITE ALL AT **RIGHT SIDE** ONLY

-SETHU

Page No.: 1
Date:
Practical No.: 1

Name of the Experiment: Implementation of Referential

Aim: TO Implement the Referential integrity.

Procedure:

→ Create a master table (course)

```
SQL > create table course
2 (cno number (5) primary key,
   3 cname varchar 2 (20));
Table created

SQL> desc course;

Name          NULL?          TYPE
-----         -----          ----
CNO           NOT NULL       NUMBER(5)
CNAME                        VARCHAR2(20)

SQL> insert into course values (&cno,'&cname');
   Enter value for cno : 1001
   Enter value for cname : BSC
   old 1 : insert into course values (&cno,'&cname')
   new 1 : insert into course values (1001,'BSC')
   1 row created.

SQL>
Enter value for cno : 1002
Enter value for cname : BCOM
old 1 : insert into course values (&cno,'&cname')
new 1 : insert into course values (1002,'BCOM')
```

1 row created.

SQL > /

Enter Value for cno : 1003

Enter value for cname : BCA

old 1 : insert into course values (&cno, &cname')

new 1 : insert into course values (1003, 'BCA')

1 row created.

SQL > /

Enter value for cno : 1004

Enter value for cname : BA

old 1 : insert into course values (&cno, '&cname')

new 1 : insert into course values (1004, 'BA')

1 row created.

SQL > commit;

commit complete.

SQL > select * from course;

CNO	CNAME
1001	BSC
1002	BCOM
1003	BCA
1004	BA

→ creating a child table (student)

```
SQL > create table student
  2 (sno number(5) primary key,
  3  sname varchar2(30);
  4  dob date,
  5  cno number (5) references course (cno));
Table created.
```

SQL > desc student;

Name	NULL ?	Type
SNO	NOT NULL	NUMBER(5)
SNAME		VARCHAR2(30)
DOB		DATE
CNO		NUMBER(5)

Page No.: 4
Date:
Name of the Experiment :
Practical No.:

SQL> insert into student values (&sno,'&sname','&dob'&cno);

Enter value for sno : 1

Enter value for sname : Bhanu

Enter value for dob : 10-Jan-1995

Enter value for cno : 1001.

old 1 : 'insert into student values (&sno,'&sname','&dob',&cno)

new 1 : insert into student values (1,'Bhanu','10-Jan-1995',1001)

1 row created.

SQL> /

Enter value for sno : 2

Enter value for sname : swathi

Enter value for dob : 26-Aug-1998

Enter value for cno : 1002

old 1 : insert into student values (&sno,'&sname','&dob'&cno)

new 1 : insert into student values (2,'swathi','26-Aug-1998',1002)

1 row created

```
SQL> /
Enter value for sno : 3
Enter value for sname : Mahesh
Enter value for dob : 12-Apr-1999
Enter value for cno : 1003
old 1: insert into student values (&sno,'&sname','&dob',&cno)
new 1: insert into student values (3,'mahesh','12-Apr-1999',1003)
  1 row created.

SQL> /
Enter value for sno : 4
Enter value for sname : Aahil
Enter value for dob : 08-Oct-2000
Enter value for cno : 1004
old 1: insert into student values (&sno,'&sname','&dob',&cno).
new 1: insert into student values (4,'Aahil','08-Oct-2000',1004)

  1 row created.

SQL> commit;
  commit complete.
```

Name of the Experiment :

Page No.: 6
Date:
Practical No.:

output:

```
SQL> select * from student;
```

SNO	SNAME	DOB	CNO.
1	Bhanu	10-JAN-95	1001
2	Swathi	26-AUG-98	1002
3	mahesh	12-APR-99	1003
4	Aahil	08-OCT-00	1004

Name of the Experiment : Implementation of Aggregate

Practical No.: 2

Aim:

To implement the aggregate functions.

Procedures:

SQL provide the various Built-in functions like shown below table:

SNO.	Built-in-function	USE
1	COUNT	to count the number of rows of the relation.
2.	MAX	to find the maximum value of the attribute (column)
3.	MIN	to find the minimum value of the attribute.
4.	SUM	to find the sum of values of the attributes provided the data type of the attribute is number
5.	AVG	to find the average of n values, ignoring null values.

Now let us try to view the table PERSON and the contents of the table PERSON as shown in fig.4.11.

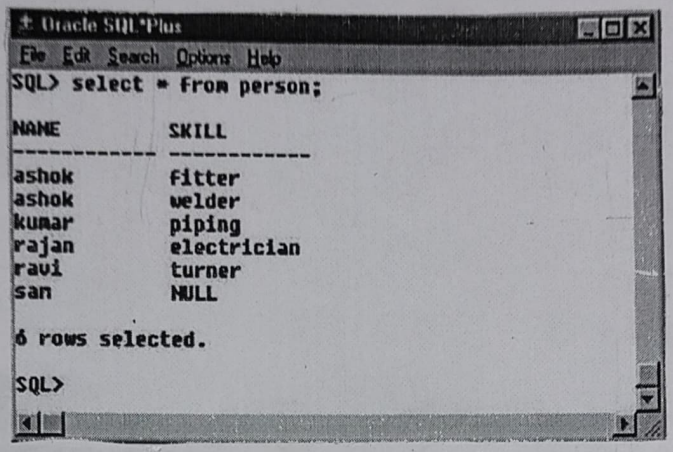


Fig. 4.11. PERSON table

From the figure, it is clear that the number of rows of the table is six.

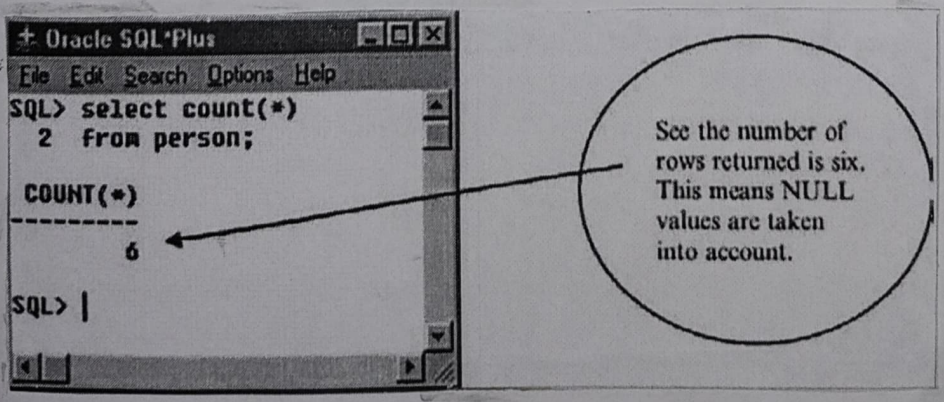
COUNT COMMAND:

Now let us use the COUNT(*) function to view the no. of rows of the relation PERSON. The SQL command and the corresponding output are shown in Fig 4.12.

The syntax of this function is given by.

Select count (attribute name) From table name;

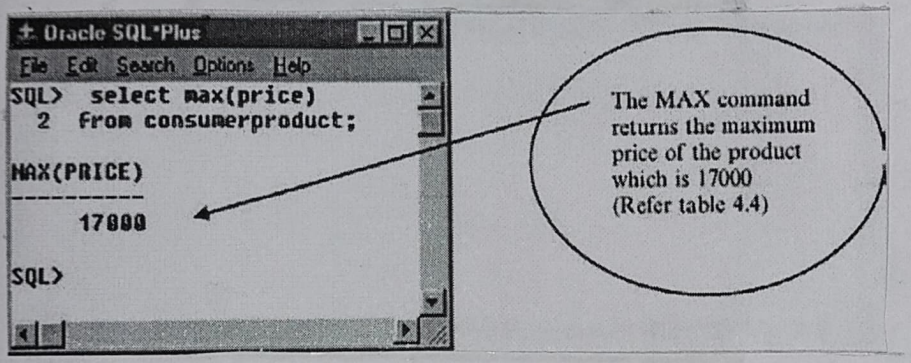
Example:



MAX command.

The MAX command stands for maximum value. The MAX command returns the maximum value of an attribute. The syntax of MAX command is:

Select max (attribute name) from table name;

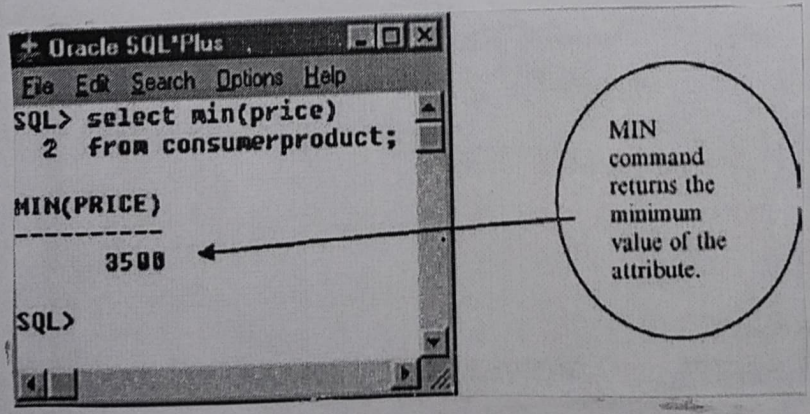


MIN Command.

The MIN command is used to return the minimum value of an attribute. The syntax of MIN command is same as MAX command.

Syntax of MIN command is.

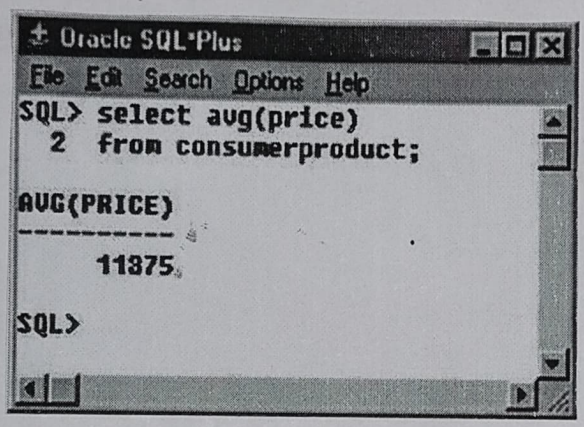
Select min (attributes name) from table name;



AVG Command

The AVG command is used to get the average value of an attribute. The syntax of AVG command is:

select AVG (attribute name) from table name;



Page No.: 11

Date:

Name of the Experiment: Demonstration on Joins

Practical No.: 3

Aim:

To Implement the joins concept.

Procedure:

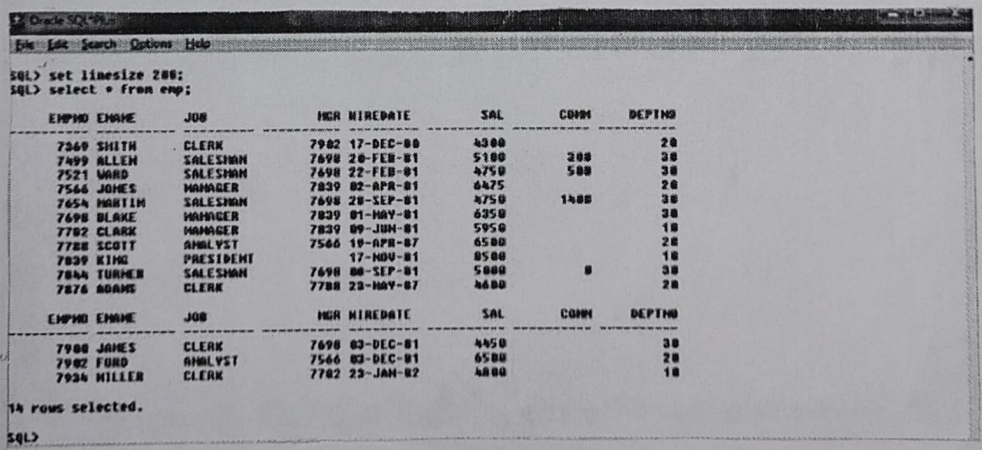
Syntax for joining table is:

Select table.1.coulum, table.2.column, ... tablen.column,
from table.1, table.2 ... tablen
where table.1.column1 = table.2.column2;

→ Equi join:

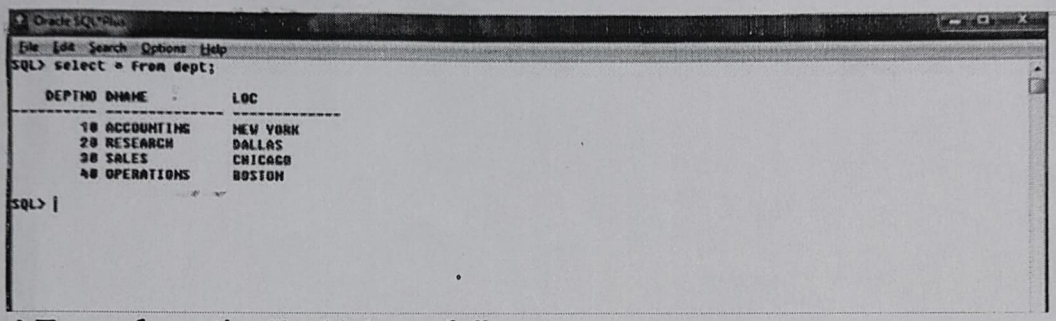
Consider the following tables.

Table: EMP



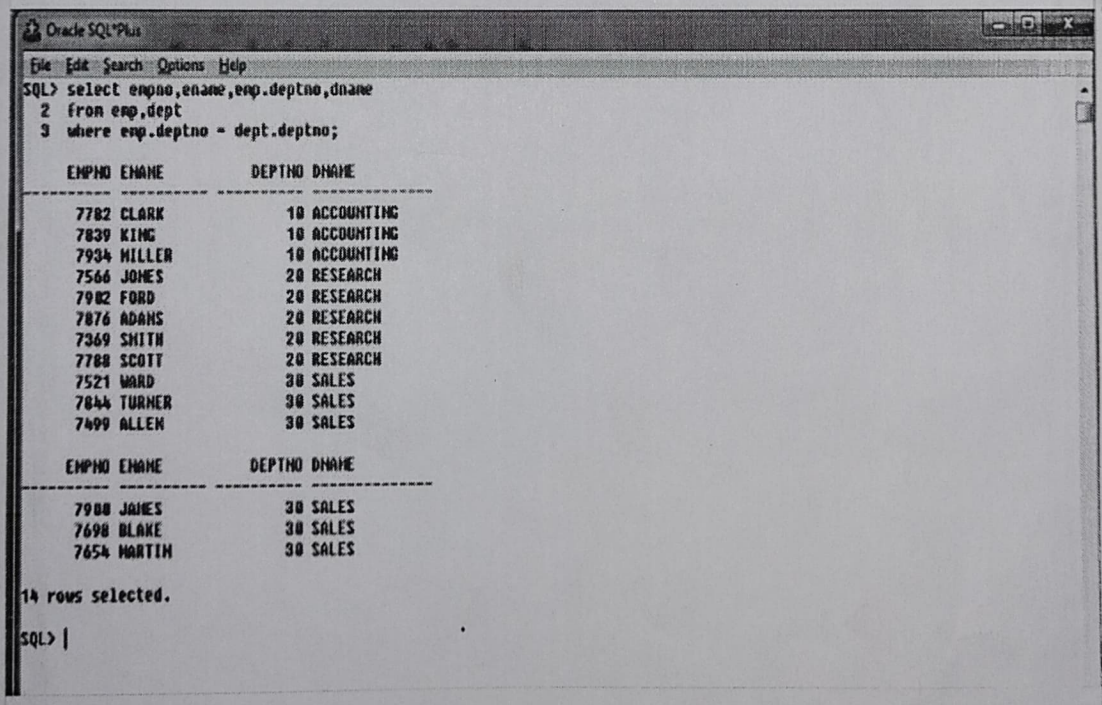
```
Oracle SQL*Plus
File  Edit  Search  Options  Help

SQL> set linesize 200;
SQL> select * from emp;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      MANAGER         7839 01-MAY-81       2850                    30
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7788 SCOTT      ANALYST         7566 19-APR-87       3000                    20
      7839 KING       PRESIDENT            17-NOV-81       5000                    10
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500          0         30
      7876 ADAMS      CLERK           7788 23-MAY-87       1100                    20

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
      7902 FORD       ANALYST         7566 03-DEC-81       3000                    20
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

14 rows selected.

SQL>
```

Table : DEPT



```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> select * from dept;

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

SQL>
```

→To perform the Equi join as follows:



```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> select empno,ename,emp.deptno,dname
  2  from emp,dept
  3  where emp.deptno = dept.deptno;

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7782 CLARK              10 ACCOUNTING
      7839 KING               10 ACCOUNTING
      7934 MILLER             10 ACCOUNTING
      7566 JONES              20 RESEARCH
      7902 FORD               20 RESEARCH
      7876 ADAMS              20 RESEARCH
      7369 SMITH              20 RESEARCH
      7788 SCOTT              20 RESEARCH
      7521 WARD               30 SALES
      7844 TURNER             30 SALES
      7499 ALLEN              30 SALES

     EMPNO ENAME          DEPTNO DNAME
---------- ---------- ---------- --------------
      7900 JAMES              30 SALES
      7698 BLAKE              30 SALES
      7654 MARTIN             30 SALES

14 rows selected.

SQL>
```

→ To perform the left outer join as follows:

Page No.: 13

Date:

Name of the Experiment :

Practical No.:

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> select empno,ename,emp.deptno,dname,loc
  2  from emp,dept
  3  where emp.deptno(+) = dept.deptno;

     EMPNO ENAME          DEPTNO DNAME          LOC
---------- ---------- ---------- -------------- -------------
      7782 CLARK              10 ACCOUNTING     NEW YORK
      7839 KING               10 ACCOUNTING     NEW YORK
      7934 MILLER             10 ACCOUNTING     NEW YORK
      7566 JONES              20 RESEARCH       DALLAS
      7902 FORD               20 RESEARCH       DALLAS
      7876 ADAMS              20 RESEARCH       DALLAS
      7369 SMITH              20 RESEARCH       DALLAS
      7788 SCOTT              20 RESEARCH       DALLAS
      7521 WARD               30 SALES          CHICAGO
      7844 TURNER             30 SALES          CHICAGO
      7499 ALLEN              30 SALES          CHICAGO

     EMPNO ENAME          DEPTNO DNAME          LOC
---------- ---------- ---------- -------------- -------------
      7900 JAMES              30 SALES          CHICAGO
      7698 BLAKE              30 SALES          CHICAGO
      7654 MARTIN             30 SALES          CHICAGO
                                 OPERATIONS     BOSTON

15 rows selected.

SQL>
```

→ To Perform the Right outer join as follows:

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> select empno,ename,emp.deptno,dname,loc
  2  from emp,dept
  3  where emp.deptno = dept.deptno(+);

     EMPNO ENAME          DEPTNO DNAME          LOC
---------- ---------- ---------- -------------- -------------
      7934 MILLER             10 ACCOUNTING     NEW YORK
      7839 KING               10 ACCOUNTING     NEW YORK
      7782 CLARK              10 ACCOUNTING     NEW YORK
      7902 FORD               20 RESEARCH       DALLAS
      7876 ADAMS              20 RESEARCH       DALLAS
      7788 SCOTT              20 RESEARCH       DALLAS
      7566 JONES              20 RESEARCH       DALLAS
      7369 SMITH              20 RESEARCH       DALLAS
      7900 JAMES              30 SALES          CHICAGO
      7844 TURNER             30 SALES          CHICAGO
      7698 BLAKE              30 SALES          CHICAGO

     EMPNO ENAME          DEPTNO DNAME          LOC
---------- ---------- ---------- -------------- -------------
      7654 MARTIN             30 SALES          CHICAGO
      7521 WARD               30 SALES          CHICAGO
      7499 ALLEN              30 SALES          CHICAGO

14 rows selected.

SQL>
```

Page No.: 17
Date:
Name of the Experiment : Demonstration on Control structures
Practical No.: 4

Aim:

To implement the control structures.

Procedure:

→ Conditional control

IF-THEN

Syntax: IF condition THEN
Sequence of statements;
END IF;

HINT: SQL> SET SERVEROUTPUT ON;

```
/* Program to find a person is major or not using if statement */
declare
age number;
begin
age := &age;
if age >= 18 then
dbms_output.put_line ('you are a major');
end if;
End;
/
```

output:

Enter value for age : 19

old 4 : age = &age;

new 4 : age = 19;

you are a major

PL/SQL procedure successfully completed.

IF-THEN-ELSE

Syntax: IF condition THEN

Sequence of statements 1;

ELSE

Sequence of statement 2;

END IF;

/* program to find a person is major or minor using if-else-statement */

```
declare
age number;
 begin
age := &age;
 if age >= 18 then
dbms_output.put_line ('you are a major');
```

```
else
dbms_output.put_line ('you are a minor');
else end if;
  end;
  /
```

Output:

```
Enter value for age : 20
old 4: age:=&age;
new 4: age:=20;
you are a major
PL/SQL procedure successfully completed.
```

→ Iterative control

WHILE - LOOP.

Syntax:

```
WHILE condition LOOP
  Sequence of statements;
  END LOOP;
```

```
/* program to demonstrate the use of while loop */
Declare
    a number (2) := 10;
Begin
    while a<20 loop
        dbms_output_line ('value of a:' ||a);
        a:=a+1;
    END LOOP;
End;
```

Output:-

```
value of a : 10
value of a : 11
value of a : 12
value of a : 13
value of a : 14
value of a : 15
value of a : 16
value of a : 17
value of a : 18
value of a : 19
```

PL/SQL procedure successfully completed.

FOR-LOOP.

Syntax:

```
For counter IN lower limit... Higher limit loop.
Sequence of statements;
END LOOP;
```

/* Program to print the table of a given number using for loop */

```
DECLARE
        n number;
        i number;
        P number;
BEGIN
        n:=&n;
        for i in 1... 10
        loop
            P:=n*i;
           dbms_output.put_line(n||'x'||i||'='||P);
           end loop;
end;
/
```

Output:

```
Enter value for n: 10
old 6: n:=&n;
new 6: n:= 10;
```

10 X 1 = 10
10 X 2 = 20
10 X 3 = 30
10 X 4 = 40
10 X 5 = 50
10 X 6 = 60
10 X 7 = 70
10 X 8 = 80
10 X 9 = 90
10 X 10 = 100

P2 / SQL procedure successfully completed.

Page No.:20
Date:
Name of the Experiment : Demonstration on cursors.
Practical No.:5

AIM:

To Create the Electricity Bill calculation using cursors.

Procedure:

→ creation of Table (ebill):

```
SQL> create table ebill (name varchar2(10),
  2 address varchar2(20),
  3 city varchar2(20),
  4 unit number(10));

table created.

SQL> insert into ebill values (
  '&name',
  '&address',
  '&city',
  '&unit');
Enter value for name : madhu
old 2:'&name',
new 2: 'madhu',
Enter value for address: 1st on Road
old 3: '&address',
new 3: '1st on Road',
Enter value for city : Tirupati
```

old 4 : '&city',
new 4 : 'Tirupati',
Enter value for unit : 100
old 5 : '&unit'
new 5 : '100
1 row created.

SQL>/
Enter value for name : Riyaz
old 2 : '&name',
new 2 : 'Riyaz',
Enter value for address : VRC CENTRE
old 3 : '&address',
new 3 : 'VRC center',
Enter value for city : NELLORE
old 4 : '&city',
new 4 : 'Nellore',
Enter value for unit : 200
old 5 : '&unit')
new 5 : '200')
1 row created

SQL>/
Enter value for name : Lakshmi
old 2 : '&name'

new 2 : 'Lasthmi',
Enter value for address : Nehru Street
old 3 : '&address',
new 3 : 'Nehru Street';
Enter value for city : VizaQ.
old 4 : '&city',
new 4 : 'VizaQ'
Enter value for unit : 300
old 5 : '&unit')
new 5 : '300')
1 row created.
SQL> /
Enter value for name : Rama Rao
old 2 : '&name',
new 2 : 'Rama Rao',
Enter value for address : Banjara Hills
old 3 : '&address',
new 3 : 'Banjara Hills',
Enter value for city : Hyderabad.
old 4 : '&city',
new 4 : 'Hyderabad',
Enter value for unit : 400.
old 5 : '&unit',)
new 5 : '400')

1 row created

SQL> commit;

Commit complete.

SQL> select * from ebill;

Name	Address	city	unit
madhu	ISKON Road	Tirupati	100
Riyaz	VRC centre	Nellore	200
Lakshmi	Nehru street	Vizag	300
Rama Rao	Banjara hills	Hydexabad	400.

SQL> set serveroutput on;

SQL> Declare

cursor c is select * from ebill;

b bill %ROWTYPE;

begin

open c;

dbms_output.put_line ('Name Address city unit Amount');

dbms_output.put_line ('<=========>');

loop

fetch c into b;

if (c% notfound) then

exit;

```
else
 if (b.unit <=100) then
dbms_output.put_line (b.name||' '||b.address||' '||b.city||' '||
                      b.unit||' '||b.unit*1);
else if (b.unit >100 and b.unit <=200) then
dbms_output.put_line (b.name||'&'||b.address||' '||b.city||' '||
                      b.unit||' '||b.unit*2);
else if (b.unit >200 and b.unit <=300) then
dbms_output.put_line (b.name||' '||b.address||' '||b.city||' '||
                      b.unit||' '||b.unit*3);
elseif (b.unit >300 and b.unit <=400) then
dbms_output.put_line (b.name||' '||b.address||' '||b.city||' '||
                      b.unit||' '||b.unit*4);
elseif (b.unit >400 and b.unit <=500) then
dbms_output.put_line (b.name||' '||b.address||' '||b.city||' '||
                      b.unit||' '||b.unit*5);

  end if;
  end if;
  end loop;
  close c;
  end;
  /
```

output:-

Name	Address	City	unit	Amount
madhu	Iskon Road	Tirupati	100	100
Riyaz	VRC center	Nellore	200	400
Lakshmi	nehru street	Vizag	300	900
Rama Rao	Banjara hills	Hyderabad	4000	1600

PL/SQL procedure successfully completed.

Page No.: 26

Date:

Name of the Experiment : Demonstration on functions in PL/SQL

Practical No.: 6

Aim:

To implement the functions in PL/SQL.

Procedure:

The General syntax of creating the function is:

```
CREATE [OR REPLACE] FUNCTION function-name
[(parameter-name [IN|OUT|IN OUT] type [,...])]
RETURN return-datatype
{IS|AS}
 BEGIN
    <Function-body>
 END
[function-name];
```

```
/* program to find the sum of two numbers using functions*/
SQL> create or replace function sump(a number, b number)
return number is
r number;
Begin
    r:=a+b;
    return(r);
End;
/
Function created.
```

Page No.: 27

Date:

Name of the Experiment :

Practical No.:

```
/* Function Execution */
declare
n1 number;
n2 number;
s number;
begin
n1:=&n1;
n2:=&n2;
s:=sump(n1,n2);
dbms_output.put_line('the sum of two numbers are ='||s);
end;
/
Enter value for n1:25
old 6: n1:=&n1;
new 6: n1:=25;
Enter value for n2:75
old 7: n2:=&n2;
new 7: n2:=75;
the sum of two numbers are=100.

PL/SQL procedure successfully completed.
```

Page No.: 28
Date:
Name of the Experiment : Demonstration on packages in PL/SQL
Practical No.: 7

Aim:

To implement the package in PL/SQL

Procedure:

→ Defining Package specification Syntax:

```
CREATE [OR REPLACE] PACKAGE package_name IS|AS
    [variable_declaration ...]
  [PROCEDURE [schema..] procedure_name
        [(parameter {IN, OUT, IN OUT} datatype [parameter])]
  ]
  [FUNCTION [schema..] function_name
        [(parameter{IN, OUT, IN OUT} datatype [parameter])]
        RETURN return_datatype;
  ]
  END [package_name];
```

→ Create Package Body Syntax:

```
CREATE [OR REPLACE] PACKAGE BODY Package_name
  IS|AS
  [private_variable_declaration ...]
    Begin
        [initialization_statement]
        [PROCEDURE [schema..] Procedure_name
              [(parameter [parameter])]
```

```
        IS|AS
            variable declarations;
        BEGIN
            statements(s);
        EXCEPTION
            WHEN...
        END
[FUNCTION[schema..]function-name
        [(parameter[parameter])]
        RETURN return_datatype;
      IS|AS
          variable declarations;
      BEGIN..
          statement(s);
      EXCEPTION
          WHEN...
      END
END;
/
```

/* Program to Demonstration of statistical functions using package

```
SQL> Create or replace package stat-fn as
    procedure sump(a number, b number, c number);
    Procedure avgp(a number, b number, c number);
    end stat-fn;
    /
package created.
SQL> create or replace package body stat-fn is
```

```
procedure sump (a number, b number, c number) is
  d number;
  begin
  d:=a+b+c;
dbms_output.put_line('the sum of given numbers are ='||d);
 end sump;
procedure avgp(a number, b number, c number) is
  d number;
  e number;
 begin
d:=a+b+c;
e:=d/3;
dbms_output.put_line('the average of given numbers are='||e);
 end avgp;
 end stat_fn;
 /
Package body created.
SQL> set serveroutput on;
SQL>/* calling the procedure's */
 SQL> declare
     n1 number;
     n2 number;
     n3 number;
```

```
begin
n1:=&n1;
n2:=&n2;
n3:=&n3;
stat-fn.sump(n1,n2,n3);
stat-fn.avgp(n1,n2,n3);
end;
/
```

output:

```
Enter value for n1: 10
old 6: n1:=&n1;
new 6: n1:=10;
Enter value for n2: 20
old 7: n2:=&n2;
new 7: n2:=20;
Enter value for n3: 30
old 8: n3:=&n3;
new 8: n3:=30;
```

The sum of given numbers are = 60

The Average of given numbers are = 30.

PL/SQL procedure successfully completed.

Page No.: 31

Date:

Name of the Experiment : Demonstration on triggers in PL/SQL

Practical No.: 8

Aim:

To implement the triggers in PL/SQL

procedure:

The General syntax to create the Triggers as:

```
CREATE [OR REPLACE] TRIGGER trigger-name
      BEFORE | AFTER
    [INSERT, UPDATE, DELETE [COLUMN NAME..]
     ON table_name
    Referencing [OLD AS OLD | NEW AS NEW]
    FOR EACH ROW | FOR EACH STATEMENT [WHEN condition]
DECLARE
  [declaration-section
   variable declarations;
   constant declarations;
 ]
Begin
   [executable-section
            PL/SQL execute / subprogram body ]
Exception
  [exception-section
            PL/SQL Exception block]
 END;
```

Name of the Experiment :

```
SQL> create table student(
     rollno number(5),
     name varchar(20),
     m1 number(3),
     m2 number(3),
     m3 number(3),
    tot number(3),
     avg number(3),
    result varchar(10));
    table created.
SQL> create or replace trigger t1 before insert on student
     for each row
      begin
     :new.tot:=:new.m1+:new.m2+:new.m3;
     :new.avg:=:new.total/3;
    if(:new.m1>=35 and :new.m2>=35 and :new.m3>=35) then
    :new.result:='pass';
    else
     :new.result:='fail';
     end if;
     end;
     /
    Trigger created.
```

Name of the Experiment :

SQL> insert into student values (101, 'Goutham', 67, 89, 99, '', '', '');
1 row created.

SQL> insert into student values (102, 'Swaroop', 85, 79, 99, '', '', '');
1 row created.

SQL> insert into student values (103, 'chandini', 89, 99, 99, '', '', '');
1 row created.

SQL> insert into student values (104, 'Dwaraka', 23, 45, 33, '', '', '');
1 row created.

SQL> commit;
Commit complete.

Output:

SQL> set linesize 200;
SQL> select * from student;

ROLL NO	NAME	M1	M2	M3	TOT	AVG	RESULT
101	Goutham	67	89	99	255	85	Pass
102	Swaroop	85	79	99	263	88	Pass
103	chandini	89	99	99	287	96	Pass
104	Dwaraka	23	45	33	101	34	Fail