PL*SQL

Exercise 7

1. Write a PL*SQL block that prompts the user to enter the salary of an employee. Your program should display the name of the employee (from the EMP table) who's getting that salary. If more than 1 employee is receiving that salary, or if no employees exist getting that salary, your program should display appropriate messages. Use *too_many_rows* and *no_data_found* exceptions to achieve this. Display the results on the screen using dbms_output.put_line.

```
declare
v_sal number;
v_name varchar2(40);
begin
v_sal:=:v_sal;
select emp_name into v_name from employ_227 where b_sal=v_sal;
dbms_output.put_line(v_sal||'  '||v_name);

exception
when too_many_rows then
dbms_output.put_line('multiple data can not be displayed');

when no_data_found then
dbms_output.put_line('data can not be displayed');
end;
```

2. Write a PL*SQL block to check if any employee from EMP table is receiving a salary greater than 9999.99. Make the use of *value_error* exception to achieve this. Display the results on the screen using dbms_output.put_line.

```
declare
v_sal number;
cursor c1 is select * from employ_227;
begin
for i in c1
loop
if i.b_sal > 1000 then
raise value_error;
end if;
end loop;
exception
when value_error then
dbms_output.put_line('Sal is greater than 1000');
```

```
end;
```

3. Create a user-defined exception by the name of *exp_check*. Select the ename and hiredate of all employees into a cursor. Your program should calculate the experience of all the employees in years, and insert the ename and experience of each employee into tempp table. If any employee has experience less than 2 years, the program should be aborted with a suitable message. Raise the user-defined exception *exp_check* to achieve this. Display the results on the screen using dbms_output.put_line.

```
declare
cursor c1 is select * from employ_227;
temp number(5);
exp_check exception;
begin
for i in c1
loop
temp:=TRUNC(MONTHS_BETWEEN(SYSDATE,i.hire_date)/12);
if temp<2 then
raise exp_check;
else
insert into tempp_227 values(i.emp_name,temp);
end if;
end loop;
exception
when exp_check then
dbms_output.put_line('Less Experience');
end;
```

4. Write a PL*SQL function to take three parameters, the sides of a triangle. The sides of the triangle should be accepted from the user. The function should return a Boolean value:- *true* if the triangle is valid, *false* otherwise. A triangle is valid if the length of each side is less than the sum of the lengths of the other two sides. Check if the dimensions entered by the user can form a valid triangle. Display the results on the screen using dbms_output.put_line.

```
create or replace function ptriangle_227(v_a number,v_b number,v_c number)
return varchar2
is
arr varchar2(10);
begin
IF v_a+v_b < v_c OR v_b+v_c < v_a OR v_c+v_a < v_b THEN
return 'false';
ELSE
```

```
return  'true';
END IF;
end;

declare
arr1 varchar2(10);
begin
arr1:=ptriangle_227(10,11,12);
if arr1='true' then
 dbms_output.put_line('Valid Triangle');
else
 dbms_output.put_line('Not A Valid Triangle');
end if;
end;
```

5. Write a function that generates a random number between 1 and 10. Use any logic of your choice to achieve this. Display the results on the screen using dbms_output.put_line.

```
create or replace function rndm(a1 number) return number
as
a number(10);
begin
a:=trunc(DBMS_RANDOM.value(low => 1, high => 10));
return a;
end;

declare
var number(8);
begin
var:=rndm(1);
DBMS_OUTPUT.put_line('value=' || var);
end;
```

6. Design a structure to store length in yards, feet, and inches (for example, 7 yards, 2 feet, 3 inches). Your program should accept 2 length measurements from the user. Write a PL*SQL procedure to find the difference between two measurements as represented by these structures. Display the results on the screen using dbms_output.put_line.

```
create or replace procedure cnv(v_i number)
as
v_i_1 number(10,0);
v_y number(10,0);
v_f number(10,0);
v_temp varchar2(20);
```

```
v_temp1 number(10,0);
begin
v_i_1:=MOD(v_i,12);
v_f:=v_i/12;
v_y:=v_f/3;
v_f:=MOD(v_i,3);
dbms_output.put_line(v_y||' Yard '||v_f||' Feet '||v_i_1||' Inches ');
end;

begin
cnv(124);
end;
```

7. Create a function that accepts a string of *n* characters and exchanges the first character with the last, the second with the next – to – last, and so forth until *n* exchanges have been made. What will the final string look like? Write the function to verify your conclusion. Display the results on the screen using dbms_output.put_line.

```
create or replace function rev(str varchar2) return varchar2
as
str2 varchar2(20);
begin
select REVERSE(str) into str2 from dual;
return str2;
end;

declare
str1 varchar2(30);
begin
str1:=rev('qwerty');
dbms_output.put_line(str1);
end;
```