SENECA COLLEGE DBS 501 Course

Assigment#2

* Due date is Feb 28th 11:59 pm before midnight.
* Late submission will be losing some points.
* Please include all the codings here under each question and also include your execution Screen shots…
* Each question is 25 mark

**1.** Create a visitor’s application that prompts the user for the number of visitors each day for the past 5 days and then displays the average number of visitors per day. (use & to accept new value)

Use PL/SQL Array (NUMBER type basic composite array type ) and array has to have 5 members only. The application output should look like similar to following: (**25 marks)**

Test: If you receive below numbers from user then your average should be 234. Display average on screen.

1: 150

2: 220

3: 180

4: 300

5: 320

The average number of visitors is: 234

Hint :

*Once you create Array*

*DECLARE*

*TYPE my\_array\_type IS TABLE OF number INDEX BY PLS\_INTEGER;*

*v\_n\_array my\_array\_type;*

*Then in the BEGIN and END block accept new value for each array members. DO NOT Use LOOP to accept new & as it is not design to get new value via loop.. Instead use separate 5 different &substitution value to get each values from user input*

*BEGIN*

*v\_n\_array(1):= &n1*

*v\_n\_array(2):=&n2*

*…………….*

*…………*

*Collect all v\_n\_array members (5 different numbers) and then do calculations like finding average.*

*v\_avg := …………*

*Then Display them*

*DBMS\_OUTPUT.PUT\_LINE (‘*The average number of visitors is: ‘ || v\_avg *)*

*END;*

|  |
| --- |
| declare  type my\_array\_type is table of number index by pls\_integer;  v\_n\_array my\_array\_type;  v\_avg number;  begin  v\_n\_array(1) := &n1;  v\_n\_array(2) := &n2;  v\_n\_array(3) := &n3;  v\_n\_array(4) := &n4;  v\_n\_array(5) := &n5;  v\_avg := (v\_n\_array(1) + v\_n\_array(2) + v\_n\_array(3) + v\_n\_array(4) + v\_n\_array(5)) / 5;  dbms\_output.put\_line('The average number of visitors is: ' || v\_avg);  end; |

**2.** Create an odds/even program that prompts the user for a number. This number will be top (cap) number. And your program will find out all the even and odd numbers in between 1 to this max number.

The program should continue to find odd and even numbers from 1 to maximum number given (entered) by user… If user input is less than or equal to zero Program should not even start doing its task and right away print ‘Sorry I can not calculate odd and even numbers for negative or zero’ … But when user entered any positive number then the program should display all numbers as either even or odd. **(25 marks)**

***Assume user entered 12. This will be cap number***

***From 1 to 12 all the numbers in between your program will be finding out each of the number either as odd or even***

***1 is odd number***

***2 is even number***

***3 is odd number***

***4 is even number***

***5 is odd number***

***6 is even number***

***7 is odd number***

***8 is even number***

***9 is odd number***

***10 is even number***

***11 is odd number***

***12 is even number***

|  |
| --- |
| declare  maxNum number;  counter number := 1;  begin  maxNum := &max\_number;  if maxNum > 0 then  while counter <= maxNum loop  if mod(counter, 2) = 0 then  dbms\_output.put\_line(counter || ' is an even number');  else  dbms\_output.put\_line(counter || ' is an odd number');  end if;  counter := counter + 1;  end loop;  else  dbms\_output.put\_line('Sorry, I cannot calculate odd and even numbers for negative or zero numbers.');  end if;  end; |

**3.**

1. First;

Create (Clone) another version of DEPARTMENTS table with Create table as select command .something like this below

***CREATE TABLE new\_departments AS SELECT \* FROM DEPARTMENTS WHERE 1=2;***

this unmatcheched WHERE condition will be able to create same table as DEPARTMENT but no data in it. Empty version.

b) build your PL/SQL block…

Create a PL/SQL block that selects the record which belongs to maximum department ID in the departments table and stores entire record info to composite variable as RECORD. (Lets say cv\_dept)

Create this composite variable same ROWTYPE of DEPARTMENT table.

Declare a variable, **v\_max\_deptno** , of type NUMBER in the declarative section.

1. Start the executable section with the BEGIN keyword and include a SELECT statement to retrevive the maximum **department\_id** from the departments table.
2. By using this **v\_max\_deptno** use this number in WHERE condition of SELECT statement and make sure you only bring one row from DEPARTMENTS table to composite variable
3. You will now INSERT this max department number record info to newly created table **new\_departments** table…
4. To do that you will use RECORD feature

INSERT INTO new\_departments VALUES cv\_dept ;

Select new\_departments table and make sure new record is there and show this with screen shot

|  |
| --- |
| create table new\_departments as select \* from hr\_departments where 1=2;  declare  v\_max\_deptno number;  cv\_dept hr\_departments%rowtype;  begin  select max(department\_id) into v\_max\_deptno from hr\_departments;  select \* into cv\_dept  from hr\_departments  where department\_id = v\_max\_deptno;  insert into new\_departments values cv\_dept;  end; |

***4) Create a metric distant conversion program..***

*Ask user three questions ansqd collect them with & substitution values*

*1st: from where (it could be cm , mt, km , mm)*

*2nd : to where: (it could be cm , mt, km , mm)*

*3rd : amount*

*For instance if 1st and 2nd entry same send a print message “ same metric conversion is not possible “ and stop the program*

*If for instance, 1st variable value is* ***from km*** *then 2nd variable is to where cm and 3rd variable amount is 1 then result will be 1km = 1000cm*

|  |
| --- |
| declare  from\_conv char(2);  to\_conv char(2);  amount number;  conv number;  begin  from\_conv := lower('&from');  to\_conv := lower('&to');  amount := &amount;  if from\_conv = to\_conv then  dbms\_output.put\_line('same metric conversion is not possible');  else  case from\_conv  when 'km' then  case to\_conv  when 'mt' then  conv := amount \* 1000;  when 'cm' then  conv := amount \* 100000;  when 'mm' then  conv := amount \* 1000000;  end case;  when 'mt' then  case to\_conv  when 'km' then  conv := amount \* 0.001;  when 'cm' then  conv := amount \* 100;  when 'mm' then  conv := amount \* 1000;  end case;  when 'cm' then  case to\_conv  when 'km' then  conv := amount \* 0.00001;  when 'mt' then  conv := amount \* 0.01;  when 'mm' then  conv := amount \* 10;  end case;  when 'mm' then  case to\_conv  when 'km' then  conv := amount \* 0.000001;  when 'mt' then  conv := amount \* 0.001;  when 'cm' then  conv := amount \* 0.1;  end case;  end case;  dbms\_output.put\_line(amount || from\_conv || ' = ' || conv || to\_conv);  end if;  end; |