Zensar 20th Sep PLSQL

Rule:

- PLSQL is not case sensitive
- **begin** and **end;** is mandatory
- all operator used in SQL, also used in PLSQL.
- **** operator can be used only in PLSQL
- single row function can be used in PLSQL block, except decode.
- group function is not allowed in PLSQL, only in SQL
- DML(Insert, update, delete,merge) is allowed in PLSQL
- TCL(commit, rollback, savepoint) is allowed in PLSQL
- DDL and DCL is not allowed in PLSQL

program 1: To add 2 number

```
DECLARE
  num1    NUMBER;
  num2    NUMBER;
  res     NUMBER;
BEGIN
  num1 := 10;
  num2 := 20;
  res := num1+num2;
  dbms_output.put_line('res: '||res);
END;
/
```

program 1(a): To add 2 number

```
DECLARE
  num1    NUMBER := 10;
  num2    NUMBER := 20;
```

```
  res     NUMBER;
BEGIN
  res := num1+num2;
  dbms_output.put_line('res: '||res);
END;
/
```

program 1(b): To add 2 number

```
DECLARE
  num1    NUMBER default 10;
  num2    NUMBER default 20;
  res     NUMBER;
BEGIN
  res := num1+num2;
  dbms_output.put_line('res: '||res);
END;
/
```

** operator can be used only in PLSQL

```
DECLARE
  num1    NUMBER;
  num2    NUMBER;
  res     NUMBER;
BEGIN
  num1 := 10;
  num2 := 5;
  res := num1 ** num2;  --num1 power num2  ==>10000
  dbms_output.put_line('res: '||res);
END;
/

SELECT 10**3 FROM dual;  --not allowed in sql
```

NESTED block

```
DECLARE
  num1    NUMBER := 10;
  num2    NUMBER :=20;
BEGIN
   --
  DECLARE
     num1  NUMBER := 111;
     num3  NUMBER := 222;
```

```
  BEGIN
    dbms_output.put_line('num1='||num1);  --111
    dbms_output.put_line('num2='||num2);  --20
    dbms_output.put_line('num3='||num3);  --222
  END;

  dbms_output.put_line('num1='||num1);  --10
  dbms_output.put_line('num2='||num2);  --20
  --dbms_output.put_line('num3='||num3);  --error
END;
/
```

```
BEGIN
  <<OUTER>>
  DECLARE
    num1    NUMBER := 10;
    num2    NUMBER :=20;
  BEGIN
    <<INNER>>
    DECLARE
       num1  NUMBER := 111;
       num3  NUMBER := 222;
    BEGIN
      dbms_output.put_line('num1='||outer.num1);  --10
      dbms_output.put_line('num2='||num2);  --20
      dbms_output.put_line('num3='||num3);  --222
    END;

    dbms_output.put_line('num1='||num1);  --10
    dbms_output.put_line('num2='||num2);  --20
    --dbms_output.put_line('num3='||num3);  --error
  END;
END;
/
```

single row function can be used in PLSQL block, except decode,nvl2.

```
--single row function can be used in PLSQL block, except decode,nvl2.
DECLARE
   str       VARCHAR2(20) := 'Sample string';
   NUM       NUMBER;
   output    VARCHAR2(100);
BEGIN
   NUM := LENGTH(str);
   output := SUBSTR(str,5);
   dbms_output.put_line(output);

   --output := MIN(str); --group function is not allowed in plsql
```

```
   --output := DECODE('a','b','c','d','e');  --decode is not allowed in plsql
   output := NVL2('a','b','c');
END;
/
```

IF ELSIF ELSE

IF

```
DECLARE
  a NUMBER := &NUM;
BEGIN
  IF a>=0 THEN
    dbms_output.put_line(a||' is postive number');
  END IF;
END;
/
```

IF ELSE

```
DECLARE
  a NUMBER := &NUM;
BEGIN
  IF a>=0 THEN
    dbms_output.put_line(a||' is postive number');
  ELSE
    dbms_output.put_line(a||' is negative number');
  END IF;
END;
/
```

IF ELSIF

```
--max of three number
DECLARE
  a NUMBER := 10;
  b NUMBER := 20;
  c NUMBER := 3;
BEGIN
  IF a>b AND a>c THEN
    dbms_output.put_line(a||' is greatest');
  ELSIF b>c THEN
    dbms_output.put_line(b||' is greatest');
  ELSE
    dbms_output.put_line(c||' is greatest');
```

```
  END IF;
END;
/
```

--alternate wayrt

```
DECLARE
  a NUMBER := 10;
  b NUMBER := 20;
  c NUMBER := 3;
BEGIN
  dbms_output.put_line(GREATEST(a,b,c)||' is greatest');
END;
/
```

Assignment 1:

```
/*
enter the mark from user.
print the grade as per mark
mark>=80  => print Grade: A
mark>=60   and mark <80 => print Grade: B
mark>=40   and mark <60 => print Grade: C
mark>=0   and mark <40 => print Grade: Fail
*/
declare
  mark number := 10;
  ...
begin


end;
/
```

Assignment 2:

```
--check year is leap year
eg:
2000  -> leap year
2100  -> not a  leap year
2004  -> leap year
2003  -> not a leap year
```

LOOP:

- simple loop
- while
- for

simple loop

```
DECLARE
   a NUMBER := 1;
BEGIN
  LOOP
    dbms_output.put_line(a);
    EXIT WHEN a=10;
    a := a+1;
  END LOOP;
END;
```

```
DECLARE
   a NUMBER := 1;
BEGIN
  LOOP
    EXIT WHEN a>10;
    dbms_output.put_line(a);
    a := a+1;
  END LOOP;
END;
/
```

While

```
DECLARE
   a NUMBER := 1;
BEGIN
   WHILE a<=10
   LOOP
    dbms_output.put_line(a);
    a := a+1;
   END LOOP;
END;
/
```

FOR loop

```
BEGIN
   FOR i IN 1..10 LOOP
     --i := i+3; --i cannot be used as assingment
     dbms_output.put_line(i);
```

```
   END LOOP;
END;
/
```

```
/*
*
**
***
****
*/
BEGIN
   FOR i IN 1..4 LOOP
    FOR j IN 1..i LOOP
      DBMS_OUTPUT.PUT('*');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
   END LOOP;
END;
/
```

```
--wap to print A to Z
BEGIN
   FOR I IN 1..26 LOOP
   DBMS_OUTPUT.PUT_LINE(CHR(64+I));
   END LOOP;
END;
/
```

--example of continue

```
--continue;
--break  use EXIT
BEGIN
   FOR i IN 1..8 LOOP
    FOR j IN 1..i LOOP
      IF j>4 THEN
        CONTINUE;
      END IF;
      DBMS_OUTPUT.PUT('*');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
   END LOOP;
END;
/
```

--exit from the outer loop from inner loop

```
BEGIN
   <<OUTER>>
   FOR i IN 1..8 LOOP
    <<INNER>>
    FOR j IN 1..i LOOP
      IF j>4 THEN
        EXIT OUTER;
      END IF;
      DBMS_OUTPUT.PUT('*');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('');
END;
/
```

```
BEGIN
   <<OUTER>>
   FOR i IN 1..8 LOOP
    <<INNER>>
    FOR j IN 1..i LOOP
      EXIT OUTER WHEN j>4;
      DBMS_OUTPUT.PUT('*');
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('');
   END LOOP;
   DBMS_OUTPUT.PUT_LINE('');
END;
/
```

Assignment 3:

```
--print 10 to 1 using
a. simple loop
b. while loop
c. for loop
```

C

```
WAP to print fibbonaci series
0 1 1 2 3 5 8 13 .... upto 100
```

Assignment 4:

```
...A
..BCD
.EFGHI
JKLMNOP
.QRSTU
..VWX
...Y
```

CASE expression:

```
DECLARE
  mark NUMBER := 90;
  grade VARCHAR2(100);
BEGIN
  grade := CASE WHEN mark > 80 THEN 'First class'
                WHEN mark > 60 THEN 'Second class'
                WHEN mark >= 40 THEN 'PASS'
                ELSE 'fail' END;

  dbms_output.put_line('Grade: '||grade);
END;
```

Assignment 5:

```
--execute assignment 1 using case expression
```

CASE statement:

```
DECLARE
  mark NUMBER := 90;
  grade VARCHAR2(100);
BEGIN
 CASE WHEN mark > 80 THEN
           dbms_output.put_line('Grade: '||'First class');
           --more statement is allowed
      WHEN mark > 60 THEN
           dbms_output.put_line('Grade: '||'Seconde class');
      WHEN mark >= 40 THEN
           dbms_output.put_line('Grade: '||'Pass');
      ELSE
        dbms_output.put_line('Grade: '||'Fail');
      END CASE;
END;
/
```

Assignment 6:

```
--execute assignment 1 using case statement
```

DML and TCL is allowed in PLSQL

```
BEGIN
   DELETE emp WHERE ROWNUM<=3;
   COMMIT;

   INSERT INTO emp(id, name, salary, did) VALUES (1,'Jack',3000,10);
   COMMIT;

   UPDATE emp
   SET salary = salary+100;
   ROLLBACK;
END;
/
```

DDL is not allowed in PLSQL

```
BEGIN
 CREATE TABLE emp1234(ID NUMBER);  --DDL not allowed in PLSQL
END;
/
```

DCL is not allowed in PLSQL

```
BEGIN
  GRANT SELECT ON employees TO SYSTEM; --DCL not allowed in PLSQL
END;
/
```