Practical - 2

Conditional and Loop structure

1) IF/ElSE

Syntax :

```
IF condition1 THEN
      sequence_of_statements1
ELSIF condition2 THEN
   sequence_of_statements2
ELSE
   sequence_of_statements3
END IF;
```

Example :

```
Declare
       Grade char(1);
Begin
Grade:= '&grade';
  IF grade = 'A' THEN
   dbms_output.put_line('Excellent');
ELSIF grade = 'B' THEN
   dbms_output.put_line('Very Good');
ELSIF grade = 'C' THEN
   dbms_output.put_line('Good');
ELSIF grade = 'D' THEN
   dbms_output. put_line('Fair');
ELSIF grade = 'F' THEN
   dbms_output.put_line('Poor');
ELSE
   dbms_output.put_line('No such grade');
END IF;
End ;
```

2) Case

Syntax :

```
CASE selector
    WHEN expression1 THEN sequence_of_statements1;
    WHEN expression2 THEN sequence_of_statements2;
    ...
     WHEN expressionN THEN sequence_of_statementsN;
    [ELSE sequence_of_statementsN+1;]
 END CASE ;
```

```
Example
     Declare
        Grade char(1);
     Begin
        Grade:='&grade';
         CASE grade
               WHEN 'A' THEN dbms_output.put_line('Excellent');
               WHEN 'B' THEN dbms_output.put_line('Very Good');
               WHEN 'C' THEN dbms_output.put_line('Good');
               WHEN 'D' THEN dbms_output.put_line('Fair');
               WHEN 'F' THEN dbms_output.put_line('Poor');
               ELSE dbms_output.put_line('No such grade');
          END CASE;

    END;
```

3) While

Syntax

```
WHILE condition LOOP

        sequence_of_statements

END LOOP;
```

Example

```
 DECLARE
 countr       NUMBER := 1;
 BEGIN
   WHILE countr < 11 LOOP
     dbms_output.put_line('Square root of ' || countr ||' is '|| SQRT(countr) );
     countr := countr + 1;
  END LOOP;
  dbms_output.put_line('End of Calculations.');
END;
```

4) For loop

Syntax

```
FOR loop_variable IN [REVERSE] lower_bound..upper_bound LOOP
    statements
END LOOP;
```

```
BEGIN
    FOR i IN REVERSE 1..10 LOOP  -- i starts at 10, ends at 1
      DBMS_OUTPUT.PUT_LINE(i); -- statements here execute 10 times
    END LOOP;
END;
```

<u>How to accept number from the user</u>

To read the user input and store it in a variable, for later use, you can use sqlplus command `ACCEPT`.

```
Accept <your variable> <variable type if needed [number|char|date]> prompt
'message'

accept x number prompt 'Please enter something: '
```

Exercise

1	Take the employee number from the user , if it is already into the table then, calculate the HRA , DA, and Net salary. (HRA 10%) DA = 60% Net salary = Basic + HRA+DA. Hint : use count() function to check empno is in table or not.
2	Take the empno from the user, if salary less than 10,000 print massage less salary Salary > 10,000 and Salary < 30,000 then print massage medium salary Salary >30,000 and Salary < 60,000 then print massage high salary Salary > 60,000 then print massage very high salary.
3	Write a pl/sql block using **case** which check the month of hiredate of given empno. And print month in word format e.g January ,march,july) use while loop for repetition of input, **Hint :** select **extract(year from hiredate)** from emp;
4	Find the reverse of the number which is input by the user.
5	product_master(prod_id, prod_name, prod_price, qty) old_price (prod_id, old_price, new_price, change_date) Accept new_price from the user. Change the price of the product P00001 to new_price if the price is less than 4000 in product_master table. The change is recorded in the old_price table along with prod_id and the change_date on which the price was last changed.