INTRODUCTION TO PL/SQL

PL/SQL stands for PROCEDURAL Language Extensions to SQL.
PL/SQL extends SQL by adding programming structures and subroutines available in any high level language.
PL/SQL can be used for both server-side and Client side Development.
PL/SQL has syntax and rules that determine how programming statements work together.
PL/SQL is not a stand alone Programming Language.
PL/SQL is a part of the ORACLE RDBMS and hence can reside in two environments,the CLIENT and the SERVER.
Any MODULE that is developed using PL/SQL can be moved easily between SERVER SIDE and CLIENT SIDE applications.
Either in CLIENT/SERVER environments any PL/SQL Block or the PL/SQL Engine processes Subroutine.
PL/SQL Engine is a special component that processes and executes any PL/SQL statements and sends any SQL statement to the SQL statement processor.
The SQL statement processes are always located on the ORACLE SERVER.

As per the necessity the PL/SQL Engine can be located either at

SERVER
CLIENT

When PL/SQL Engine is locted upon the SERVER,the whole PL/SQL block is passed to the PL/SQL Engine on the ORACLE SERVER.
When the PL/SQL Engine is located upon the CLIENT,the PL/SQL processing is done on the CLIENT SIDE.All SQL statenents that are embedded within the PL/SQL block,are sent to the ORACLE SERVER for further processing.

If the PL/SQL block does not contain any SQL statements,the entire block is executed on the CLIENT SIDE.

PL/SQL BLOCK

```
DECLARE
--Declarations of memory variables,constants,cursors etc.,in PL/SQL
BEGIN
--SQL executable statements
--PL/SQL executable statements
EXCEPTION
/*SQL or PL/SQL code to handle errors that may arise during the execution of the code block between BEGIN and EXCEPTION section
END;
```

SYNTAX's of CONTROL STATEMENTS in PL/SQL

1. BRANCHING
2. SELECTION
3. LOOPING

BRANCHING STATEMENTS

1.Simple IF
2.ELSIF
3.ELSE IF

SIMPLE IF

```
IF condition THEN
   statement1;
   statement2;
END IF;
```

IF-THEN-ELSE STATEMENT

```
IF condition THEN
   statement1;
ELSE
   statement2;
END IF;
```

ELSIF STATEMENTS

```
IF condition1 THEN
  statement1;
ELSIF condition2 THEN
  statement2;
ELSIF condition3 THEN
  statement3;
ELSE
```

```
  statementn;
END IF;
```

NESTED IF

```
IF condition THEN
   statement1;
ELSE
  IF condition THEN
    statement2;
  ELSE
    statement3;
  END IF;
END IF;
ELSE
  statement3;
END IF;
```

SELECTION IN PL/SQL

SIMPLE CASE

```
CASE SELECTOR
    WHEN Expr1 THEN statement1;
    WHEN Expr2 THEN statement2;
          :
          :
          :
ELSE
     statementn;

END CASE;
```

SEARCHED CASE

```
CASE
  WHEN  searchcondition1 THEN statement1;
  WHEN  searchcondition2 THEN statement2;
        :
        :
        :
ELSE
        statementn;
END CASE;
```

ITERATIONS IN PL/SQL

SIMPLE LOOP

```
LOOP
 statement1;
EXIT [ WHEN Condition];
END LOOP;
```

WHILE LOOP

```
WHILE condition LOOP
        statement1;
        statement2;
END LOOP;
```

FOR LOOP

```
FOR counter IN [REVERSE]
   LowerBound..UpperBound
LOOP
        statement1;
        statement2;
END LOOP;
```

WRITE A PL/SQL PROGRAM TO SWAP TWO NUMBERS WITH OUT TAKING THIRD VARIABLE

```
declare
a number(10);
b number(10);
begin
a:=&a;
b:=&b;
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
a:=a+b;
b:=a-b;
a:=a-b;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

OUTPUT:

```
SQL> @ SWAPPING.SQL
 17  /
Enter value for a: 5
old   5: a:=&a;
new   5: a:=5;
Enter value for b: 3
old   6: b:=&b;
new   6: b:=3;
THE PREV VALUES OF A AND B WERE
5
3
THE VALUES OF A AND B ARE
3
5

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO SWAP TWO NUMBERS BY TAKING THIRD VARIABLE

```
declare
a number(10);
b number(10);
c number(10);
begin
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
a:=&a;
b:=&b;
c:=a;
a:=b;
b:=c;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

OUTPUT:

```
SQL> @ SWAPPING2.SQL
 19  /
Enter value for a: 5
old   6: a:=&a;
new   6: a:=5;
Enter value for b: 3
old   7: b:=&b;
new   7: b:=3;
THE PREV VALUES OF A AND B WERE
5
3
THE VALUES OF A AND B ARE
3
5

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO FIND THE LARGEST OF TWO NUMBERS

```
declare
a number;
b number;
begin
a:=&a;
b:=&b;
if a=b then
dbms_output.put_line('BOTH ARE EQUAL');
elsif   a>b   then
dbms_output.put_line('A IS GREATER');
else
dbms_output.put_line('B IS GREATER');
end if;
end;
```

OUTPUT:

```
SQL> @ GREATESTOF2.sql
 13  /
Enter value for a: 5
old   5: a:=&a;
new   5: a:=5;
Enter value for b: 2
old   6: b:=&b;
new   6: b:=2;
A IS GREATER

PL/SQL procedure successfully completed.
```


WRITE A PL/SQL PROGRAM TO FIND THE LARGEST OF THREE NUMBERS

```
declare
a number;
t number;
arm number;
d number;
begin
a:=&a;
t:=a;
arm:=0;
while t>0
loop
d:=mod(t,10);
arm:=arm+power(d,3);
t:=trunc(t/10);
end loop;
if arm=a then
dbms_output.put_line('given no is an armstrong no'||a);
else
dbms_output.put_line('given no is not an armstrong no');
end if;
end;
```

OUTPUT:

```
SQL> @ ARMSTRONGNUM.sql
Enter value for a: 407
old   7: a:=&a;
new   7: a:=407;
given no is an armstrong no407

PL/SQL procedure successfully completed.

SQL> /
Enter value for a: 406
old   7: a:=&a;
new   7: a:=406;
given no is not an armstrong no

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO FIND THE SUM OF DIGITS IN A GIVEN NUMBER

```
declare
```

```
a number;
d number:=0;
sum1 number:=0;
begin
a:=&a;
while a>0
loop
d:=mod(a,10);
sum1:=sum1+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('sum is'|| sum1);
end;
```

OUTPUT:

```
SQL> @ SUMOFDIGITS.sql
 16  /
Enter value for a: 564
old   7: a:=&a;
new   7: a:=564;
sum is15

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO DISPLAY THE NUMBER IN REVERSE ORDER

```
declare
a number;
rev number;
d number;
begin
a:=&a;
rev:=0;
while a>0
loop
d:=mod(a,10);
rev:=(rev*10)+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('no is'|| rev);
end;
```

OUTPUT:

```
SQL> @ REVERSE2.sql
 16  /
Enter value for a: 536
old   6: a:=&a;
new   6: a:=536;
no is635

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO DISPLAY NUMBER IN REVERSE ORDER USING STRING FUNCTION

```
declare
gn varchar2(5):=4567;
sl number(2);
rv varchar2(5);
begin
sl:=length(gn);
for i in reverse 1..sl
loop
rv:=rv||substr(gn,i,1);
end loop;
dbms_output.put_line('given no r is'||gn);
dbms_output.put_line('given no in reverse order is'||rv);
end;
```

OUTPUT:

```
SQL> @ REVERSE.sql
 14  /
given no r is4567
given no in reverse order is7654

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO CHECK WHETHER THE GIVEN NUMBER IS PRIME OR NOT

```
declare
```

```
a number;
c number:=0;
i number;
begin
a:=&a;
for i in 1..a
loop
if mod(a,i)=0 then
c:=c+1;
end if;
end loop;
if c=2 then
dbms_output.put_line(a ||'is a prime number');
else
dbms_output.put_line(a ||'is not a prime number');
end if;
end;
```

OUTPUT:

```
SQL> @ PRIME.SQL
 19 /
Enter value for a: 11
old   6: a:=&a;
new   6: a:=11;
11is a prime number

PL/SQL procedure successfully completed.
```

WRITE A PL/SQL PROGRAM TO FIND THE FACTORIAL OF A GIVEN NUMBER

```
declare
n number;
```