PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Disadvantages of SQL:

- **SQL doesn't provide the programmers with a technique of condition checking, looping and branching.**
- **SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.**
- **SQL has no facility of error checking during manipulation of data.**

Features of PL/SQL:

1. **PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.**
2. **PL/SQL can execute a number of queries in one block using single command.**
3. **One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.**
4. **PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.**
5. **Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.**
6. **PL/SQL Offers extensive error checking.**

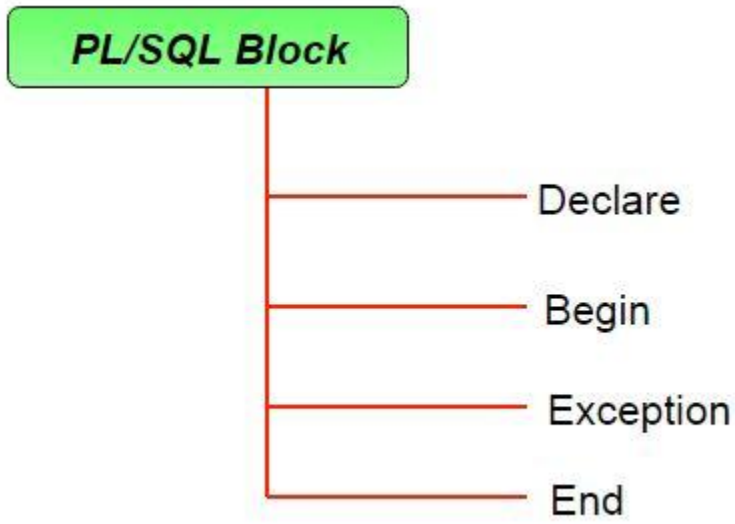
Differences between SQL and PL/SQL:

SQL	PL/SQL
SQL is a single query that is used to perform DML and DDL operations.	PL/SQL is a block of codes that used to write the entire program blocks/ procedure/ function, etc.
It is declarative, that defines what needs to be done, rather than how things need to be done.	PL/SQL is procedural that defines how the things needs to be done.
Execute as a single statement.	Execute as a whole block.
Mainly used to manipulate data.	Mainly used to create an application.
Cannot contain PL/SQL code in it.	It is an extension of SQL, so it can contain SQL inside it.

Structure of PL/SQL Block:

PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The

basic unit in PL/SQL is a block. All PL/SQL programs are made up of blocks, which can be nested within each other.



Typically, each block performs a logical action in the program. A block has the following structure:

```
DECLARE
    declaration statements;

BEGIN
    executable statements

EXCEPTIONS
    exception handling statements

END;
```

- **Declare section starts with DECLARE keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.**
- **Execution section starts with BEGIN and ends with END keyword.This is a mandatory section and here the program logic is**

written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL*PLUS built-in functions as well.

- Exception section starts with EXCEPTION keyword.This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

PL/SQL identifiers

There are several PL/SQL identifiers such as variables, constants, procedures, cursors, triggers etc.

Variables:
Like several other programming languages, variables in PL/SQL must be declared prior to its use. They should have a valid name and data type as well.
Syntax for declaration of variables:

```
variable_name datatype [NOT NULL := value ];
```

1. Example to show how to declare variables in PL/SQL :

```
SQL> SET SERVEROUTPUT
ON;

SQL> DECLARE
    var1 INTEGER;
    var2 REAL;
    var3 varchar2(20) ;

BEGIN
```

```
    null;
END;
/
```

Output:

```
PL/SQL procedure successfully completed.
```

2.

Explanation:

- **SET SERVEROUTPUT ON: It is used to display the buffer used by the dbms_output.**
- **var1 INTEGER : It is the declaration of variable, named *var1* which is of integer type. There are many other data types that can be used like float, int, real, smallint, long etc. It also supports variables used in SQL as well like NUMBER(prec, scale), varchar, varchar2 etc.**
- **PL/SQL procedure successfully completed.: It is displayed when the code is compiled and executed successfully.**
- **Slash (/) after END;: The slash (/) tells the SQL*Plus to execute the block.**

3.

1.1) INITIALISING VARIABLES:

The variables can also be initialised just like in other programming languages. Let us see an example for the same:

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
     var1 INTEGER := 2 ;
     var3 varchar2(20) := 'I Love
GeeksForGeeks' ;

  BEGIN
     null;

  END;
  /
```

Output:

```
PL/SQL procedure successfully completed.
```

4.

Explanation:

- **Assignment operator (:=) : It is used to assign a value to a variable.**

5.

Displaying Output:

The outputs are displayed by using DBMS_OUTPUT which is a built-in package that enables the user to display output, debugging information, and send messages from PL/SQL blocks, subprograms, packages, and triggers.

Let us see an example to see how to display a message using PL/SQL :

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
     var varchar2(40) := 'I love
GeeksForGeeks' ;

   BEGIN
      dbms_output.put_line(var);

   END;
   /
```

Output:

```
I love GeeksForGeeks

PL/SQL procedure successfully completed.
```

6.

Explanation:

- ***dbms_output.put_line*** **: This command is used to direct the PL/SQL output to a screen.**

7.

Using Comments:

Like in many other programming languages, in PL/SQL also, comments can be put within the code which has no effect in the code. There are two syntaxes to create comments in PL/SQL :

- **Single Line Comment: To create a single line comment , the symbol – – is used.**
- **Multi Line Comment: To create comments that span over several lines, the symbol /* and */ is used.**

8. **Example to show how to create comments in PL/SQL :**

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE

     -- I am a comment, so i will be
ignored.
     var varchar2(40) := 'I love
GeeksForGeeks' ;

  BEGIN
     dbms_output.put_line(var);

  END;
  /
```

Output:

```
I love GeeksForGeeks

PL/SQL procedure successfully completed.
```

9.

10.

Taking input from user:

Just like in other programming languages, in PL/SQL also, we can take input from the user and store it in a variable. Let us see an example to show how to take input from users in PL/SQL:

```
SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

      -- taking input for
variable a
      a number := &a;

      -- taking input for
variable b
      b varchar2(30) := &b;

  BEGIN
      null;

  END;
  /
```

Output:

```
Enter value for a: 24
old   2: a number := &a;
new   2: a number := 24;
Enter value for b: 'GeeksForGeeks'
old   3: b varchar2(30) := &b;
new   3: b varchar2(30) := 'GeeksForGeeks';

PL/SQL procedure successfully completed.
```

11.

12.

(***) ***Let us see an example on PL/SQL to demonstrate all above concepts in one single block of code.***

```
--PL/SQL code to print sum of two numbers taken from
the user.
SQL> SET SERVEROUTPUT ON;

SQL> DECLARE

     -- taking input for variable a
     a integer := &a ;

     -- taking input for variable b
     b integer := &b ;
     c integer ;

  BEGIN
     c := a + b ;
     dbms_output.put_line('Sum of '||a||' and '||b||'
is = '||c);

  END;
  /


Enter value for a: 2
Enter value for b: 3

Sum of 2 and 3 is = 5

PL/SQL procedure successfully completed.
```

13.

PL/SQL Execution Environment:

The PL/SQL engine resides in the Oracle engine.The Oracle engine can process not only single SQL statement but also block of many statements.The call to Oracle engine needs to be made only once to execute any number of SQL statements if these SQL statements are bundled inside a PL/SQL block.

Greatest number among three given numbers in PL/SQL

In PL/SQL code groups of commands are arranged within a block. A block group related declarations or statements. In declare part, we declare variables and between begin and end part, we perform the operations. Given three numbers, the task is to find greatest among them.

Examples:

```
Input: a = 46, b = 67, c = 21
Output: 67

Input: a = 9887, b = 4554, c = 212


Output: 9887
```

Approach is to consider the first number and compare it with other two numbers. Similarly, check with second and third.

```
--To find the greatest number
-- among given three numbers
DECLARE
    --a assigning with 46
    a NUMBER := 46;
    --b assigning with 67
    b NUMBER := 67;
    --c assigning with 21
    c NUMBER := 21;
BEGIN
    --block start
    --If condition start
    IF a > b
       AND a > c THEN
      --if a is greater then print a
      dbms_output.Put_line('Greatest
number is '
                           ||a);
    ELSIF b > a
          AND b > c THEN
      --if b is greater then print b
      dbms_output.Put_line('Greatest
number is '
                           ||b);
    ELSE
      --if c is greater then print c
      dbms_output.Put_line('Greatest
number is '
                           ||c);
    END IF;
--end if condition
END;
--End program
```

Output: `Greatest number is 67`

Print all odd numbers and their sum from 1 to n in PL/SQL

In PL/SQL code groups of commands are arranged within a block. A block group related declarations or statements. In declare part, we declare variables and between begin and end part, we perform the operations.

Given a number N, the task is to display all the odd numbers from 1 to N and their sum.

Examples:

```
Input: 3
Output: 1, 3

Input: 5
Output: 1, 3, 5
```

Approach is to initialize a *num* with 1 and *sum* with 0 and keep incrementing *num* by 2 and *sum* by *num* until num <= N.

```
-- display all odd number from 1 to n
DECLARE
    -- declare variable num
    num NUMBER(3) := 1;
    sum1 NUMBER(4) := 0;
BEGIN
    WHILE num <= 5 LOOP
        -- display odd number
        dbms_output.Put_line(num);

        -- the sum of all odd numbers
         sum1 := sum1 + num;

        --next odd number
        num := num + 2;

    -- end  loop
    END LOOP;
dbms_output.Put_line('Sum of all odd numbers is
'|| sum1);
END;
```

Output:

```
1
3
5
Sum of all odd numbers is 9
```