* SQL:
  + A *declarative* language used to perform a single database operation at a time, such as querying, inserting, updating, or deleting data.
  + Focuses on *what* needs to be done (e.g., select this data), not how.
* PL/SQL:
  + A *procedural language* extension to SQL provided by Oracle.
  + Allows grouping multiple SQL statements and procedural logic like loops, conditionals (if-else), and error handling into a block of code.
  + Used for writing complex programs such as stored procedures, functions, triggers, and batch jobs inside the database.

**2. Execution**

* SQL:
  + Executes one statement at a time.
  + Each query or command is sent to the database engine individually.
* PL/SQL:
  + Executes a block of statements together as a unit.
  + This reduces network traffic between the application and the database by sending multiple instructions in one go.
  + Supports control structures to manage flow of logic inside the block.

**3. Structure**

* SQL:
  + Consists of individual commands like SELECT, INSERT, UPDATE, DELETE.
* PL/SQL:
  + Organized in blocks with logical sections:

text

DECLARE -- optional variable declarations

BEGIN -- executable statements (SQL + procedural code)

EXCEPTION -- optional error handling

END;

* + Allows declaring variables, constants, cursors, and handling exceptions (errors).

A variable is like a container that holds data during program execution. Each variable must have a valid name and a specific **data type**.

**Syntax for declaration of variables:**

*variable\_name datatype [NOT NULL := value ];*

* **variable\_name**: The name of the variable.
* **datatype**: The data type of the variable (e.g., INTEGER, VARCHAR2).
* **NOT NULL**: This optional constraint means the variable cannot be left empty.
* **:= value**: This optional assignment assigns an initial value to the variable.

The command SET SERVEROUTPUT ON is used in Oracle SQL\*Plus or similar tools to enable the display of output generated by the DBMS\_OUTPUT package in PL/SQL.

**What it does:**

* When you run PL/SQL code that uses DBMS\_OUTPUT.PUT\_LINE to print messages, by default these messages are not displayed on the SQL\*Plus or SQL Developer output console.
* Executing SET SERVEROUTPUT ON turns on the output display, so any messages written to the DBMS\_OUTPUT buffer are shown on your screen after the PL/SQL block completes.

**Why is it needed?**

* The DBMS\_OUTPUT.PUT\_LINE procedure writes messages to an internal buffer, but this buffer is not shown by default.
* SET SERVEROUTPUT ON directs the client tool to fetch and display those buffered messages.
* Without this, your PUT\_LINE calls will not produce visible output.

*SQL> SET SERVEROUTPUT ON;  
SQL> DECLARE  
var varchar2(40) := 'I love GeeksForGeeks' ;  
  
BEGIN  
dbms\_output.put\_line(var);  
  
END;  
/*

**Comments in PL/SQL**

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.

**Example: Adding Comments**

*-- This is a single-line comment  
  
/\*  
This is a multi-line comment  
that spans over multiple lines.  
\*/*

**1. Using Substitution Variables (&)**

* When you use &variable in your PL/SQL block or SQL query, it prompts the user to enter a value at runtime.
* The value entered is then substituted in the code.

SET SERVEROUTPUT ON;

DECLARE

a NUMBER := &a; -- User prompted to input a number

b VARCHAR2(30) := '&b'; -- User prompted to input text (note the single quotes)

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Number entered: ' || a);

DBMS\_OUTPUT.PUT\_LINE('Text entered: ' || b);

END;

/

* **Anonymous Blocks:** This are the blocks which do not have any name associated with the block. The most disadvantage of the PL/ SQL anonymous blocks is that this block can be used only once as they are not stored anywhere in the oracle database.
* **Named Blocks:** This are generally the procedures and functions which can be executed and used again. This are reusable block structures in PL/SQL. Along with the name of the procedure, we can also optionally specify the list of parameters used for input. Each of the individual parameter declared inside the parenthesis can be either OUT/ IN or INOUT parameter. This are called as the modes of parameters. This mode helps us to specify whether the parameter that we have created is used for writing to or reading from the procedure.

SET SERVEROUTPUT ON;

-- Create Procedure with parameters

CREATE OR REPLACE PROCEDURE show\_student\_details(

p\_name VARCHAR2,

p\_rollno NUMBER

) IS

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Student Name: ' || p\_name);

DBMS\_OUTPUT.PUT\_LINE('Roll Number: ' || p\_rollno);

END;

/

-- Method 1: EXECUTE command

EXECUTE show\_student\_details('John Doe', 101);

-- Method 2: Anonymous block

BEGIN

show\_student\_details('Alice', 202);

END;

/

A computer screen with white and green text

AI-generated content may be incorrect.A computer screen shot of a program

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

What is %TYPE in PL/SQL?

* + %TYPE is an attribute in PL/SQL that lets you declare a variable with the same data type as a column in a database table (or as another variable).

ow it works here

* customers.id%TYPE → The variable will have the same data type as the id column from the customers table.
* customers.name%TYPE → Same data type as the name column.
* customers.address%TYPE → Same data type as the address column.

A screenshot of a computer program

AI-generated content may be incorrect.