**PL/SQL Language Basics**

Like other programming languages, PL/SQL has a character set, reserved words, punctuation, datatypes, and fixed syntax rules.

**Character Sets and Lexical Units**  
PL/SQL programs are written as lines of text using a specific set of characters:  
■ Upper- and lower-case letters A .. Z and a .. z  
■ Numerals 0 .. 9  
■ Symbols ( ) + – \* / < > = ! ~ ^ ; : . ‘ @ % , ” # $ & \_ | { } ? [ ]  
■ Tabs, spaces, and carriage returns

PL/SQL keywords are not case-sensitive, so lower-case letters are equivalent to corresponding upper-case letters except within string and character literals.

A line of PL/SQL text contains groups of characters known as lexical units:  
■ **Delimiters (simple and compound symbols)** – A delimiter is a simple or compound symbol that has a special meaning to PL/SQL. For example, you use delimiters to represent arithmetic operations such as addition and subtraction.  
**■ Identifiers(which include reserved words)** – We use identifiers to name PL/SQL program items and units, which include constants, variables, exceptions, cursors, cursor variables, subprograms, and packages.  
■ **Literals** – A literal is an explicit numeric, character, string, or BOOLEAN value not represented by an identifier.  
■ **Comments** – The PL/SQL compiler ignores comments, but you should not. Adding comments to your program promotes readability and aids understanding. Generally, you use comments to describe the purpose and use of each code segment. PL/SQL supports two comment styles: single-line and multi-line.

***Interested in learning SQL? Go through this***[***SQL Tutorial***](https://intellipaat.com/blog/tutorial/sql-tutorial/)***!***

**Declarations**  
You can declare variables and constants in the declarative part of any PL/SQL block, subprogram, or package. Declarations allocate storage space for a value, specify its data type, and name the storage location so that you can reference it.  
**Some examples follow:**

DECLARE

birthday DATE;

emp\_count SMALLINT := 0;

* **Constants –**To declare a constant, put the keyword CONSTANT before the type specifier.

**Example –**

DECLARE

credit\_limit CONSTANT REAL := 5000.00;

* **Using DEFAULT**

You can use the keyword DEFAULT instead of the assignment operator to initialize variables. For example, the declaration

blood\_type CHAR := 'O';

can be rewritten as follows:

blood\_type CHAR DEFAULT 'O';

* **Using NOT NULL –**

Besides assigning an initial value, declarations can impose the NOT NULL constraint:

DECLARE

acct\_id INTEGER(4) NOT NULL := 9999;

* **Using the %TYPE Attribute**

The %TYPE attribute provides the datatype of a variable or database column. This is particularly useful when declaring variables that will hold database values. For example, assume there is a column named last\_name in a table named employees.  
To declare a variable named v\_last\_name that has the same datatype as column title, use dot notation and the %TYPE attribute, as follows:

v\_last\_name employees.last\_name%TYPE;

* **Using the %ROWTYPE Attribute**

The %ROWTYPE attribute provides a record type that represents a row in a table or view. Columns in a row and corresponding fields in a record have the same names and datatypes. However, fields in a %ROWTYPE record do not inherit constraints, such as the NOT NULL or check constraint, or default values.

DECLARE

dept\_rec departments%ROWTYPE; -- declare record variable

* **Restrictions on Declarations**

PL/SQL does not allow forward references. You must declare a variable or constant before referencing it in other statements, including other declarative statements.

DECLARE

-- Multiple declarations not allowed.

-- i, j, k, l SMALLINT;

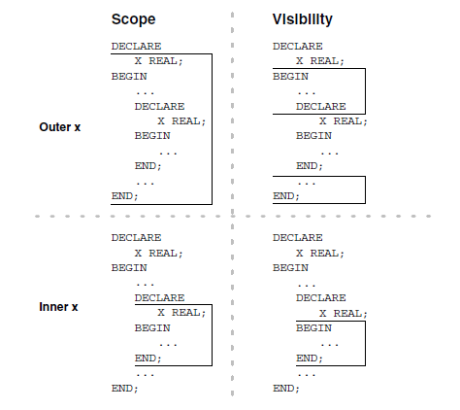
-- Instead, declare each separately.

i SMALLINT;

j SMALLINT;

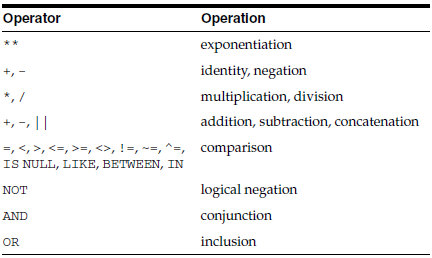
-- To save space, you can declare more than one on a line.

k SMALLINT; l SMALLINT;

[](https://cdn.intellipaat.com/wp-content/uploads/2015/09/scope-and-visibility.png)

***Have you got more queries? Come to our***[***Community***](https://intellipaat.com/community/)***and get them clarified today!***

**PL/SQL Expressions and Comparisons**  
Expressions are constructed using operands and operators. An operand is a variable, constant, literal, or function call that contributes value to an expression. An example of a simple arithmetic expression follows:  
-X / 2 + 3  
Unary operators such as the negation operator (-) operate on one operand; binary operators such as the division operator (/) operate on two operands. PL/SQL has no ternary operators.  
**Table: Order of Operations**

[](https://cdn.intellipaat.com/wp-content/uploads/2015/09/order-of-operations.png)

**Logical Operators**  
The logical operators AND, OR, and NOT.AND and OR is binary operators; NOT is a unary operator.

* **IS NULL Operator**

The IS NULL operator returns the BOOLEAN value TRUE if its operand is null or FALSE if it is not null. Comparisons involving nulls always yield NULL. Test whether a value is null as follows:

IF variable IS NULL THEN ...

* **LIKE Operator**

You use the LIKE operator to compare a character, string, or CLOB value to a pattern. The case is significant. LIKE returns the BOOLEAN value TRUE if the patterns match or FALSE if they do not match.  
The patterns matched by LIKE can include two special-purpose characters called wildcards. An underscore (\_) matches exactly one character; a percent sign (%) matches zero or more characters. For example, if the value of last\_name is ‘JOHNSON’, the following expression is true:

last\_name LIKE 'J%S\_N'

* **BETWEEN Operator**

The BETWEEN operator tests whether a value lies in a specified range or not. It means “greater than or equal to a low value and less than or equal to high value.” For example, the following expression is false:

45 BETWEEN 38 AND 44

* **IN Operator**

The IN operator tests set membership. It means “equal to any member of.” The set can contain nulls, but they are ignored. For example, the following expression tests whether a value is part of a set of values:

letter IN ('a','b','c')

* **Concatenation Operator**

Double vertical bars (||) serve as the concatenation operator, which appends one string (CHAR, VARCHAR2, CLOB, or the equivalent Unicode-enabled type) to another.  
For example, the expression

'suit' || 'case'

returns the following value:

'suitcase'

**BOOLEAN Expressions**  
BOOLEAN expressions consist of simple or complex expressions separated by relational operators. Often, BOOLEAN expressions are connected by logical operators AND, OR, and NOT. A BOOLEAN expression always yields TRUE, FALSE, or NULL.  
There are three kinds of BOOLEAN expressions: arithmetic, character, and date.

***Enroll yourself for***[***PL/SQL Training***](https://intellipaat.com/pl-sql-online-training/)***and give a head-start to your career in PL/SQL!***

**CASE Expressions**  
There are two types of expressions used in CASE statements: simple and searched. These expressions correspond to the type of CASE statement in which they are used.

* **Simple CASE expression**

A simple CASE expression selects a result from one or more alternatives and returns the result. Although it contains a block that might stretch over several lines, it really is an expression that forms part of a larger statement, such as an assignment or a procedure call. The CASE expression uses a selector, an expression whose value determines which alternative to return.

* **Searched CASE Expression**

A searched CASE expression lets you test different conditions instead of comparing a single expression to various values. A searched CASE expression has no selector. Each WHEN clause contains a search condition that yields a BOOLEAN value, so you can test different variables or multiple conditions in a single WHEN clause.

**Handling Null Values in Comparisons and Conditional Statements**  
When working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

* Comparisons involving nulls always yield NULL
* Applying the logical operator NOT to a null yield NULL
* In conditional control statements, if the condition yields NULL, its associated sequence of statements is not executed.
* If the expression is a simple CASE statement or CASE expression yields NULL, it cannot be matched by using WHEN NULL. In this case, you would need to use the searched case syntax and test WHEN expression IS NULL.

**Conditional Compilation**  
Using conditional compilation, you can customize the functionality in a PL/SQL application without having to remove any source code. For example, using conditional compilation you can customize a PL/SQL application to:  
■ Utilize the latest functionality with the latest database release and disable the new features to run the application against an older release of the database  
■ Activate debugging or tracing functionality in the development environment and hide that functionality in the application while it runs at a production site

**PL/SQL Web Applications**  
With PL/SQL you can create applications that generate Web pages directly from an Oracle database, allowing you to make your database available on the Web and make back-office data accessible on the intranet.  
The program flow of a PL/SQL Web application is similar to that in a CGI Perl script. Developers often use CGI scripts to produce Web pages dynamically, but such scripts are often not optimal for accessing Oracle Database. Delivering Web content with PL/SQL stored procedures provides the power and flexibility of database processing.  
For example, you can use DML, dynamic SQL, and cursors. You also eliminate the processing overhead of forking a new CGI process to handle each HTTP request. You can implement a Web browser-based application entirely in PL/SQL with PL/SQL Gateway and the PL/SQL Web Toolkit.  
■ **PL/SQL gateway** enables a Web browser to invoke a PL/SQL stored procedure through an HTTP listener. mod\_plsql, one implementation of the PL/SQL gateway, is a plug-in of Oracle HTTP Server and enables Web browsers to invoke PL/SQL stored procedures.  
■ **PL/SQL Web Toolkit** is a set of PL/SQL packages that provides a generic interface to use stored procedures called by mod\_plsql at runtime.

***Get familiar with the top***[***PL/SQL Interview Questions***](https://intellipaat.com/blog/interview-question/pl-sql-interview-questions/)***to get a head start in your career!***

**PL/SQL Server Pages**  
**PL/SQL Server Pages (PSPs)** enables you to develop Web pages with dynamic content. They are an alternative to coding a stored procedure that writes out the HTML code for a web page, one line at a time.  
Using special tags, you can embed PL/SQL scripts into HTML source code. The scripts are executed when the pages are requested by Web clients such as browsers. A script can accept parameters, query or update the database, then display a customized page showing the results.  
During development, PSPs can act like templates with a static part for page layout and a dynamic part for content. You can design the layouts using your favorite HTML authoring tools, leaving placeholders for the dynamic content. Then, you can write the PL/SQL scripts that generate the content. When finished, you simply load the resulting PSP files into the database as stored procedures