**ORACLE PL/SL STORED PROCEDURE AND FUNCTIONS**

**What is the Oracle database?**

Oracle database is a relational database management system. It is also called **OracleDB**, or simply **Oracle**. It is produced and marketed by **Oracle Corporation**. It was created in **1977** by **Lawrence Ellison** and other engineers. It is one of the most popular relational database engines in the IT market for storing, organizing, and retrieving data.

Oracle database was the first DB that designed for **enterprise grid computing** and data warehousing. Enterprise grid computing provides the most flexible and cost-effective way to manage information and applications. It uses SQL queries as a language for interacting with the database.

**Introduction to PL/SQL:**

**PL/SQL stands for “Procedural Language extensions to the Structured Query Language”.** [SQL](https://www.oracletutorial.com/oracle-basics/) is a popular language for both [querying](https://www.oracletutorial.com/oracle-basics/oracle-select/) and [updating data](https://www.oracletutorial.com/oracle-basics/oracle-update/) in the relational database management systems (RDBMS). PL/SQL adds many procedural constructs to SQL language to overcome some limitations of SQL. Besides, PL/SQL provides a more comprehensive programming language solution for building mission-critical applications on Oracle Databases.

PL/SQL is a standard and portable language for Oracle Database development. If you develop a program that executes on an Oracle Database, you can quickly move it to another compatible Oracle Database without any changes.

## What is Procedure in PL/SQL?

A **Procedure** in PL/SQL is a subprogram unit that consists of a group of PL/SQL statements that can be called by name. Each procedure in PL/SQL has its own unique name by which it can be referred to and called. This subprogram unit in the Oracle database is stored as a database object.

 The characteristics of Procedure subprogram unit in PL/SQL:

* Procedures are standalone blocks of a program that can be stored in the [database](https://www.guru99.com/introduction-to-database-sql.html).
* Call to these PLSQL procedures can be made by referring to their name, to execute the PL/SQL statements.
* It is mainly used to execute a process in PL/SQL.
* It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
* It contains declaration part (optional), execution part, exception handling part (optional).
* The values can be passed into Oracle procedure or fetched from the procedure through parameters.
* These parameters should be included in the calling statement.
* A Procedure in SQL can have a RETURN statement to return the control to the calling block, but it cannot return any values through the RETURN statement.
* Procedures cannot be called directly from SELECT statements. They can be called from another block or through EXEC keyword.

Syntax:

CREATE OR REPLACE PROCEDURE

<procedure\_name>

(

<parameterl IN/OUT <datatype>

..

.

)

[ IS | AS ]

<declaration\_part>

BEGIN

<execution part>

EXCEPTION

<exception handling part>

END;

**Example1: Creating Procedure and calling it using EXEC**

In this example, we are going to create an Oracle procedure that takes the name as input and prints the welcome message as output. We are going to use EXEC command to call procedure.

CREATE OR REPLACE PROCEDURE welcome\_msg (p\_name IN VARCHAR2)

IS

BEGIN

dbms\_output.put\_line (‘Welcome '|| p\_name);

END;

/

EXEC welcome\_msg (‘Soumya’);

**Code Explanation:**

* **Code line 1**: Creating the procedure with name ‘welcome\_msg’ and with one parameter ‘p\_name’ of ‘IN’ type.
* **Code line 4**: Printing the welcome message by concatenating the input name.
* Procedure is compiled successfully.
* **Code line 7**: Calling the procedure using EXEC command with the parameter ‘Soumya. Procedure is executed, and the message is printed out as “Welcome Soumya”.

**FUNCTIONS**:

Functions is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.

* Functions are a standalone block that is mainly used for calculation purpose.
* Function use RETURN keyword to return the value, and the datatype of this is defined at the time of creation.
* A Function should either return a value or raise the exception, i.e. return is mandatory in functions.
* Function with no DML statements can be directly called in SELECT query whereas the function with DML operation can only be called from other PL/SQL blocks.
* It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
* It contains declaration part (optional), execution part, exception handling part (optional).
* The values can be passed into the function or fetched from the procedure through the parameters.
* These parameters should be included in the calling statement.
* A PLSQL function can also return the value through OUT parameters other than using RETURN.
* Since it will always return the value, in calling statement it always accompanies with assignment operator to populate the variables.

#### **Syntax:**

CREATE OR REPLACE FUNCTION

<procedure\_name>

(

<parameterl IN/OUT <datatype>

)

RETURN <datatype>

[ IS | AS ]

<declaration\_part>

BEGIN

<execution part>

EXCEPTION

<exception handling part>

END;

**Creating Function and calling it using Anonymous Block**

CREATE OR REPLACE FUNCTION welcome\_msgJune ( p\_name IN VARCHAR2) RETURN VAR.CHAR2

IS

BEGIN

RETURN (‘Welcome ‘|| p\_name);

END;

/

DECLARE

lv\_msg VARCHAR2(250);

BEGIN

lv\_msg := welcome\_msg\_func (‘Soumya);

dbms\_output.put\_line(lv\_msg);

END;

SELECT welcome\_msg\_func(‘Soumya:) FROM DUAL;

**OUTPUT**:

Welcome Soumya

**Procedure Vs. Function: Key Differences**

| **Procedure** | **Function** |
| --- | --- |
| * Used mainly to a execute certain process | * Used mainly to perform some calculation |
| * Cannot call in SELECT statement | * A Function that contains no DML statements can be called in SELECT statement |
| * Use OUT parameter to return the value | * Use RETURN to return the value |
| * It is not mandatory to return the value | * It is mandatory to return the value |
| * RETURN will simply exit the control from subprogram. | * RETURN will exit the control from subprogram and also returns the value |
| * Return datatype will not be specified at the time of creation | * Return datatype is mandatory at the time of creation |

## Built-in Functions in PL/SQL

### Conversion Functions

| **Function Name** | **Usage** | **Example** |
| --- | --- | --- |
| TO\_CHAR | Converts the other datatype to character datatype | TO\_CHAR(123); |
| TO\_DATE ( string, format ) | Converts the given string to date. The string should match with the format. | TO\_DATE(‘2015-JAN-15’, ‘YYYY-MON-DD’);  Output: 1/15/2015 |
| TO\_NUMBER (text, format) | Converts the text to number type of the given format.  Informat ‘9’ denotes the number of digits | Select TO\_NUMBER(‘1234′,’9999’) from dual;  Output: 1234  Select TO\_NUMBER(‘1,234.45′,’9,999.99’) from dual;  Output: 1234 |

### String Functions

These are the functions that are used on the character datatype.

| **Function Name** | **Usage** | **Example** |
| --- | --- | --- |
| INSTR(text, string, start, occurance) | Gives the position of particular text in the given string.   * text – Main string * string – text that need to be searched * start – starting position of the search (optional) * accordance – occurrence of the searched string (optional) | Select INSTR(‘AEROPLANE’,’E’,2,1) from dual  **Output**: 2  Select INSTR(‘AEROPLANE’,’E’,2,2) from dual  Output: 9 (2nd occurance of E) |
| SUBSTR ( text, start, length) | Gives the substring value of the main string.   * text – main string * start – starting position * length – length to be sub stringed | select substr(‘aeroplane’,1,7) from dual  **Output**: aeropla |
| UPPER ( text ) | Returns the uppercase of the provided text | Select upper(‘guru99’) from dual;  **Output**: GURU99 |
| LOWER ( text ) | Returns the lowercase of the provided text | Select lower (‘AerOpLane’) from dual;  **Output**: aeroplane |
| INITCAP ( text) | Returns the given text with the starting letter in upper case. | Select (‘guru99’) from dual  **Output**: Guru99  Select (‘my story’) from dual  **Output**: My Story |
| LENGTH ( text ) | Returns the length of the given string | Select LENGTH (‘guru99’) from dual;  **Output**: 6 |
| LPAD ( text, length, pad\_char) | Pads the string in the left side for the given length (total string) with the given character | Select LPAD(‘guru99’, 10, ‘$’) from dual;  **Output**: $$$$guru99 |
| RPAD (text, length, pad\_char) | Pads the string in the right side for the given length (total string) with the given character | Select RPAD(‘guru99′,10,’-‘) from dual  **Output**: guru99—- |
| LTRIM ( text ) | Trims the leading white space from the text | Select LTRIM(‘ Guru99’) from dual;  **Output**: Guru99 |
| RTRIM ( text ) | Trims the trailing white space from the text | Select RTRIM(‘Guru99 ‘) from dual;  **Output**; Guru99 |

### Date Functions

These are functions that are used for manipulating with dates.

| **Function Name** | **Usage** | **Example** |
| --- | --- | --- |
| ADD\_MONTHS (date, no.of months) | Adds the given months to the date | ADD\_MONTH(‘2015-01-01’,5);  **Output**: 05/01/2015 |
| SYSDATE | Returns the current date and time of the server | Select SYSDATE from dual;  **Output**: 10/4/2015 2:11:43 PM |
| TRUNC | Round of the date variable to the lower possible value | select sysdate, TRUNC(sysdate) from dual;  **Output**: 10/4/2015 2:12:39 PM 10/4/2015 |
| ROUND | Rounds the date to the nearest limit either higher or lower | Select sysdate, ROUND(sysdate) from dual  **Output**: 10/4/2015 2:14:34 PM 10/5/2015 |
| MONTHS\_BETWEEN | Returns the number of months between two dates | Select MONTHS\_BETWEEN (sysdate+60, sysdate) from dual  **Output**: 2 |