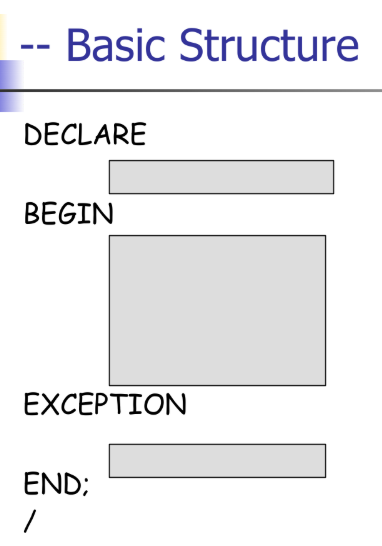
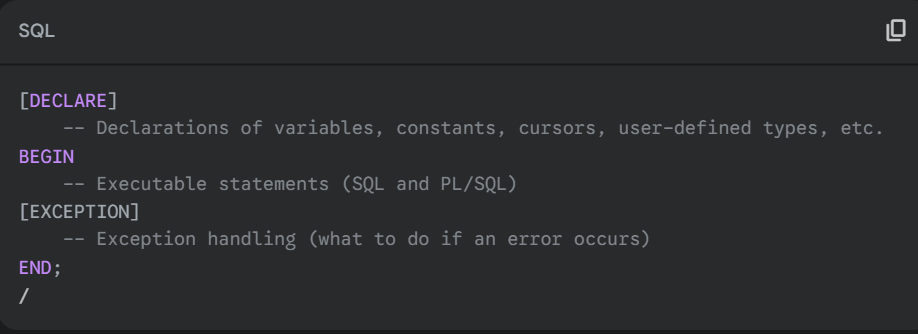
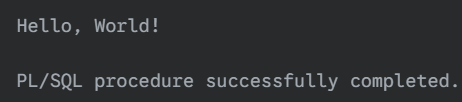
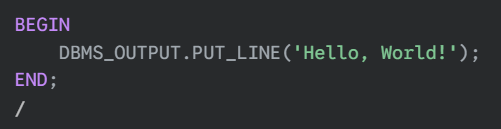
PL/SQL stands for **Procedural Language/SQL**. It's an extension of SQL (Structured Query Language) developed by Oracle Corporation. While SQL is primarily used for managing and querying relational databases, PL/SQL adds procedural capabilities to SQL, allowing you to write more complex programs, control flow, and handle errors.

**PL/SQL** is like a programming language that uses SQL commands, along with features like variables, loops, conditional statements, and error handling, to create sophisticated applications that work with your Oracle database.

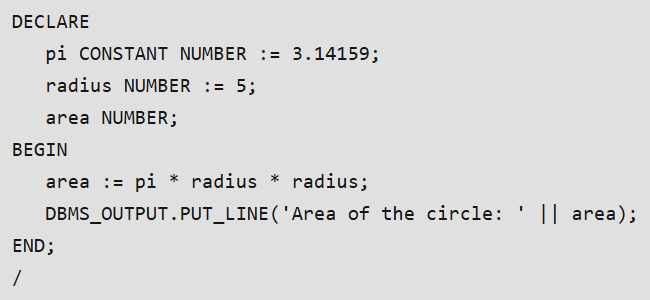
**Basic Syntax/Structure**

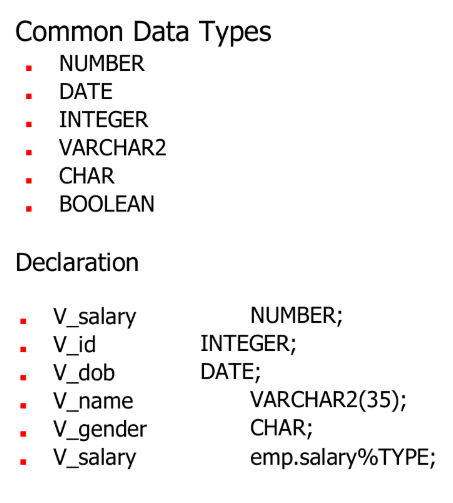


Simple Hello World Program



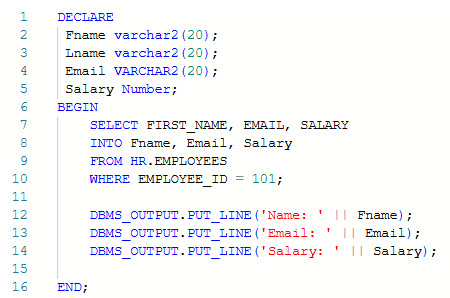
**Constant Attributes:**

****

**Variables:  
*%TYPE* Attribute**: This is a very useful attribute. It allows you to declare a variable with the same data type and size as a specific database column or another variable. This makes your code more robust against schema changes.

**Assignment Operator:** Use **:=** for assignment in PL/SQL.

**Concatenation Operator**: Use **||** to concatenate strings.

****

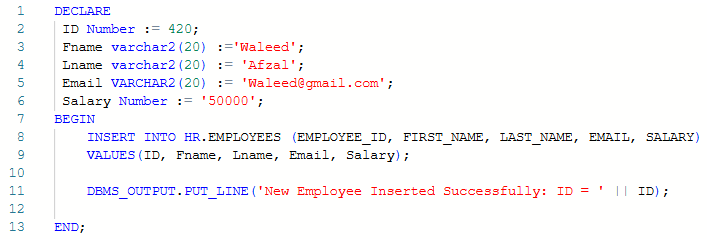
**SELECT INTO:**

The SELECT INTO clause of SQL is used to retrieve one row or set of columns from the Oracle database.

When you use a SELECT statement in PL/SQL, you must provide an INTO clause to store the retrieved data into PL/SQL variables.

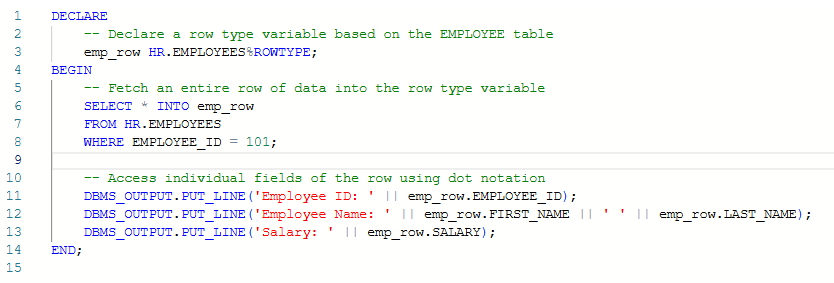
**Important Rule:** A SELECT ... INTO statement *must* return exactly one row. If it returns zero rows or more than one row, it will raise an exception.

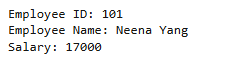
**INSERT INTO:**



**ROWTYPE:**

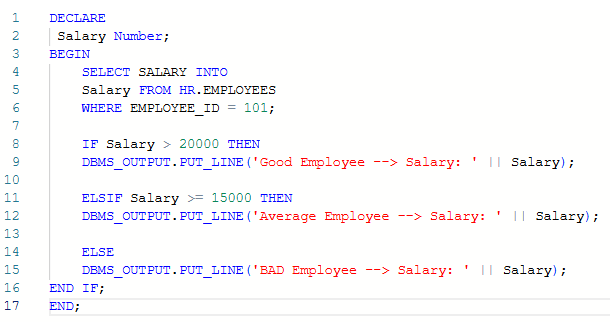
A "row type" refers to a data type that represents an entire row of a table. It allows you to work with a complete row of data as a single entity, rather than dealing with individual columns separately.





In this example, emp\_row is a variable of the row type HR.EMPLOYEES%ROWTYPE, which means it can hold an entire row of data from the EMPLOYEE table. After fetching the data into emp\_row, you can access individual columns using dot notation (e.g., emp\_row.EMPLOYEE\_ID).

**Conditionals (IF ELSIF ELSE):**



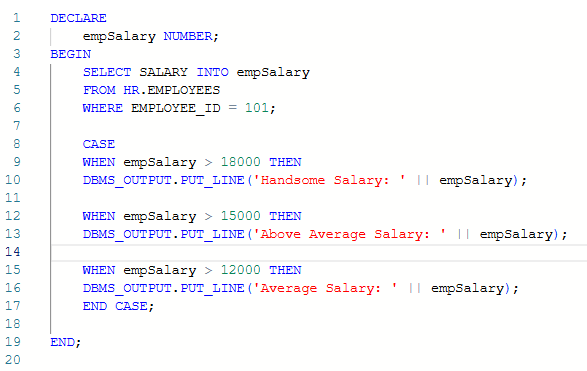
Output:



**CASE STATEMENTS:**

1. **Simple CASE Statement:**

This is most akin to a traditional "switch case" where you have a single value you want to test against different options.

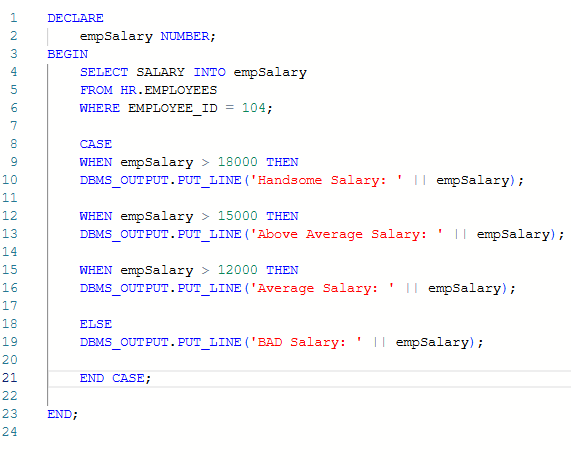
****

****

1. **Searched CASE Statement:**

This form is more flexible as it allows you to specify different Boolean conditions for each WHEN clause, similar to a series of IF-ELSIF-ELSE statements.

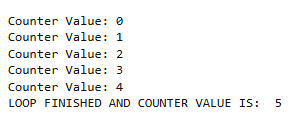
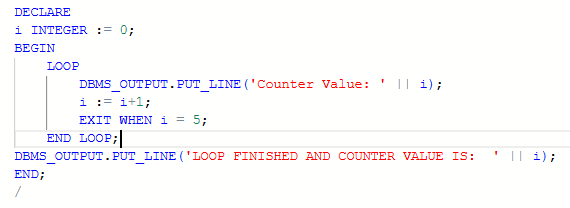




**LOOPS:**

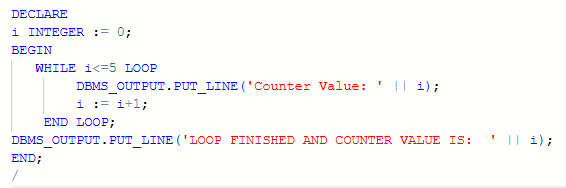
**🡪 Basic Loop:**

This is the most fundamental loop structure. It runs indefinitely until an EXIT or EXIT WHEN statement is encountered within the loop body.

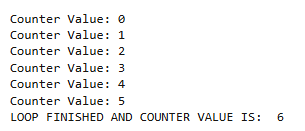


**🡪 While Loop:**

The WHILE loop executes statements repeatedly as long as a specified Boolean condition remains TRUE. The condition is evaluated at the beginning of each iteration. If the condition is initially FALSE or NULL, the loop body is never executed.

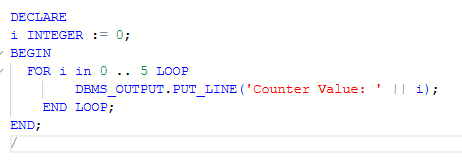






**🡪 FOR Loop:**

The numeric FOR loop is ideal when you know the exact number of iterations required. It iterates over a specified range of integers. The loop counter is automatically declared (you don't declare it in the DECLARE section) and its value is automatically incremented or decremented with each iteration.

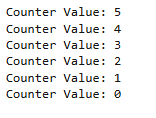
A screenshot of a computer

AI-generated content may be incorrect.

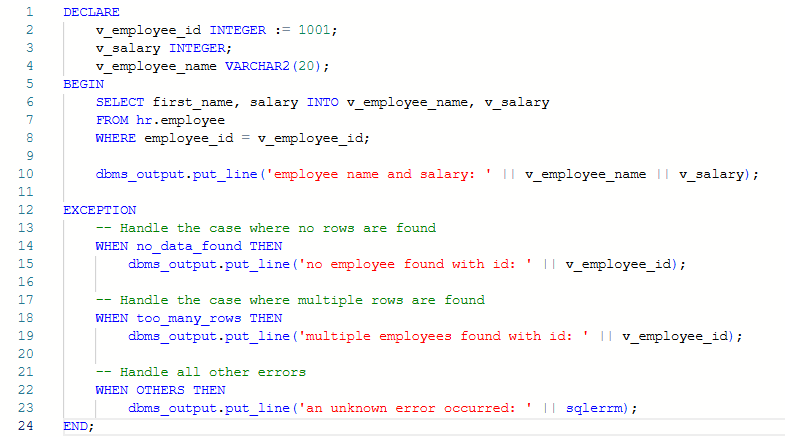
**\*~~\* REVERSE FOR LOOP:**

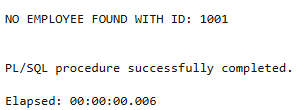
A screenshot of a computer

AI-generated content may be incorrect.



**EXCEPTIONS HANDLING:**





**TAKE USER INPUT:**

