Harmony IT Solution

Database Design and Programming

Tahaluf Training Center 2022

Harmony IT Solution

1. PL/SQL
2. Anonymous Block
3. Data Types
4. Variables
5. Comments

Harmony IT Solution

PL/SQL

- **PL/SQL** stands for “Procedural Language extensions to the Structured Query Language”.

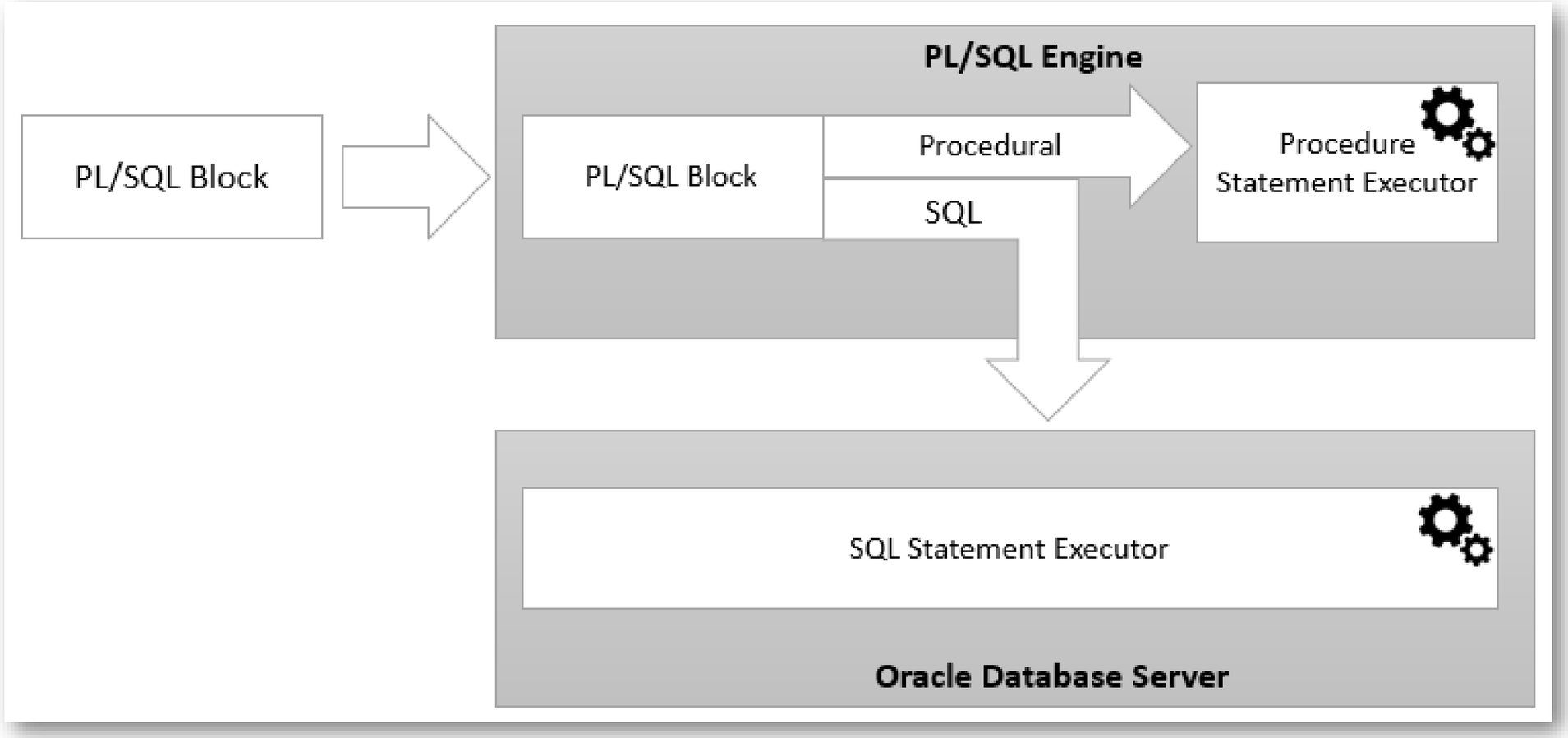
- SQL is a popular language for both querying and updating data in the relational database management systems (**RDBMS**).

- **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 an embedded language.
- **PL/SQL** only can execute in an Oracle Database. It was not designed to use as a standalone language like Java, C#, and C++.

In other words, you cannot develop a PL/SQL program that runs on a system that does not have an Oracle Database.



Anonymous Block

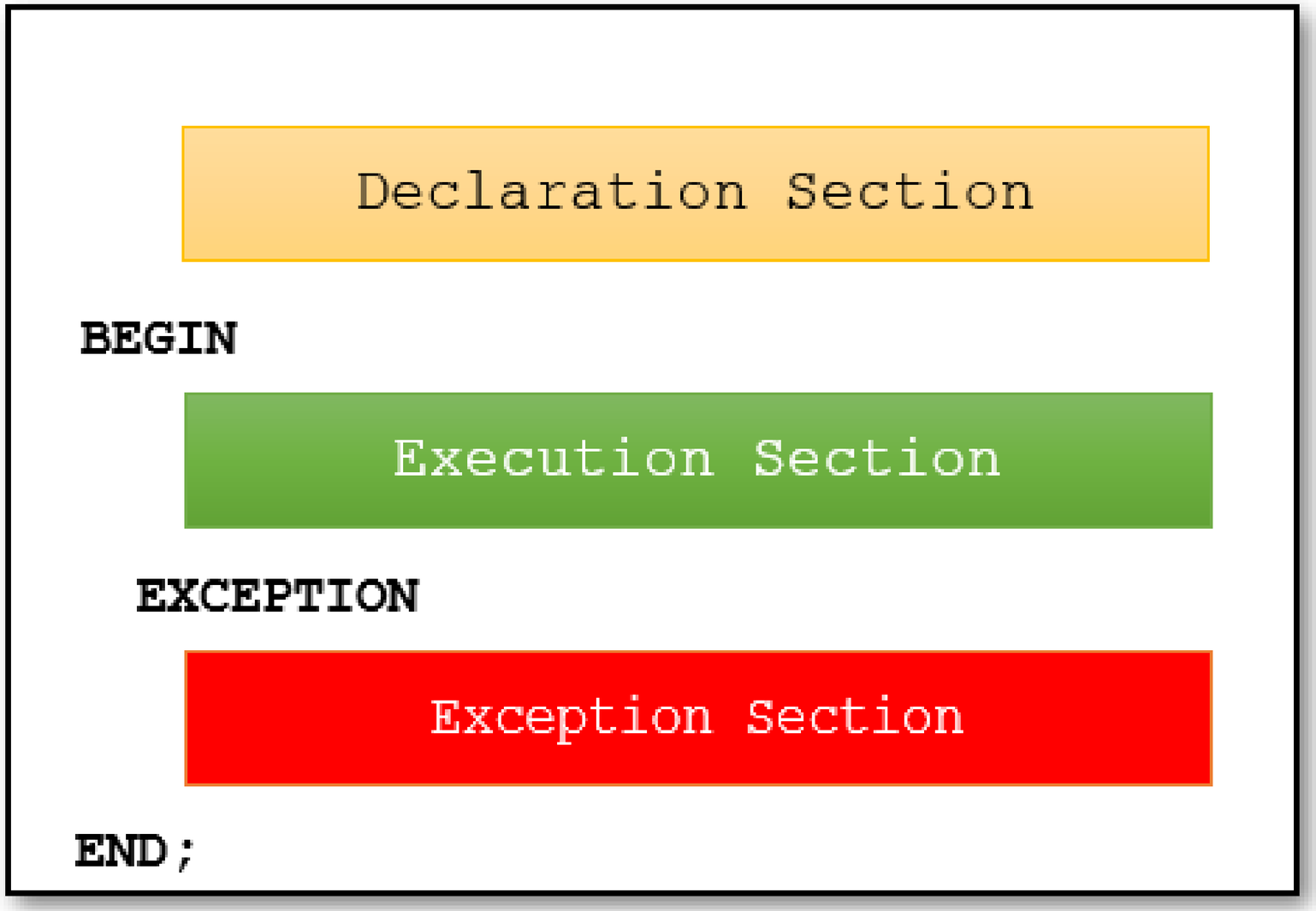
- PL/SQL is a block-structured language whose code is organized into blocks. A PL/SQL block consists of three sections: **declaration**, **executable**, and **exception-handling** sections.

- **In a block**, the executable section is mandatory while the declaration and exception-handling sections are optional.

- A **PL/SQL** block has a name. Functions or Procedures is an example of a named block. A named block is stored into the Oracle Database server and can be reused later.

- A block without a name is an **anonymous block**. An anonymous block is not saved in the Oracle Database server, so it is just for one-time use.

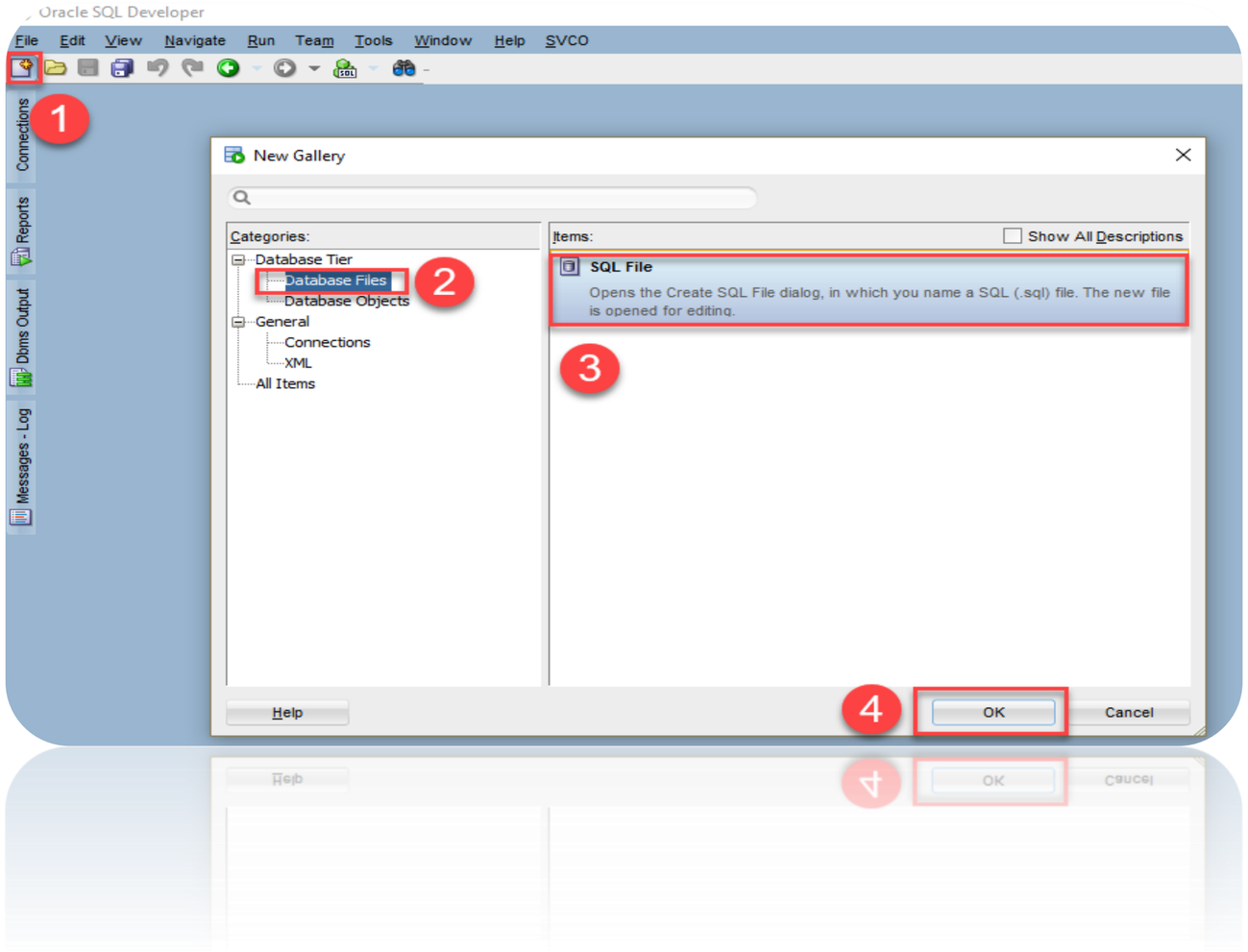
PL/SQL anonymous blocks can be useful for testing purposes.



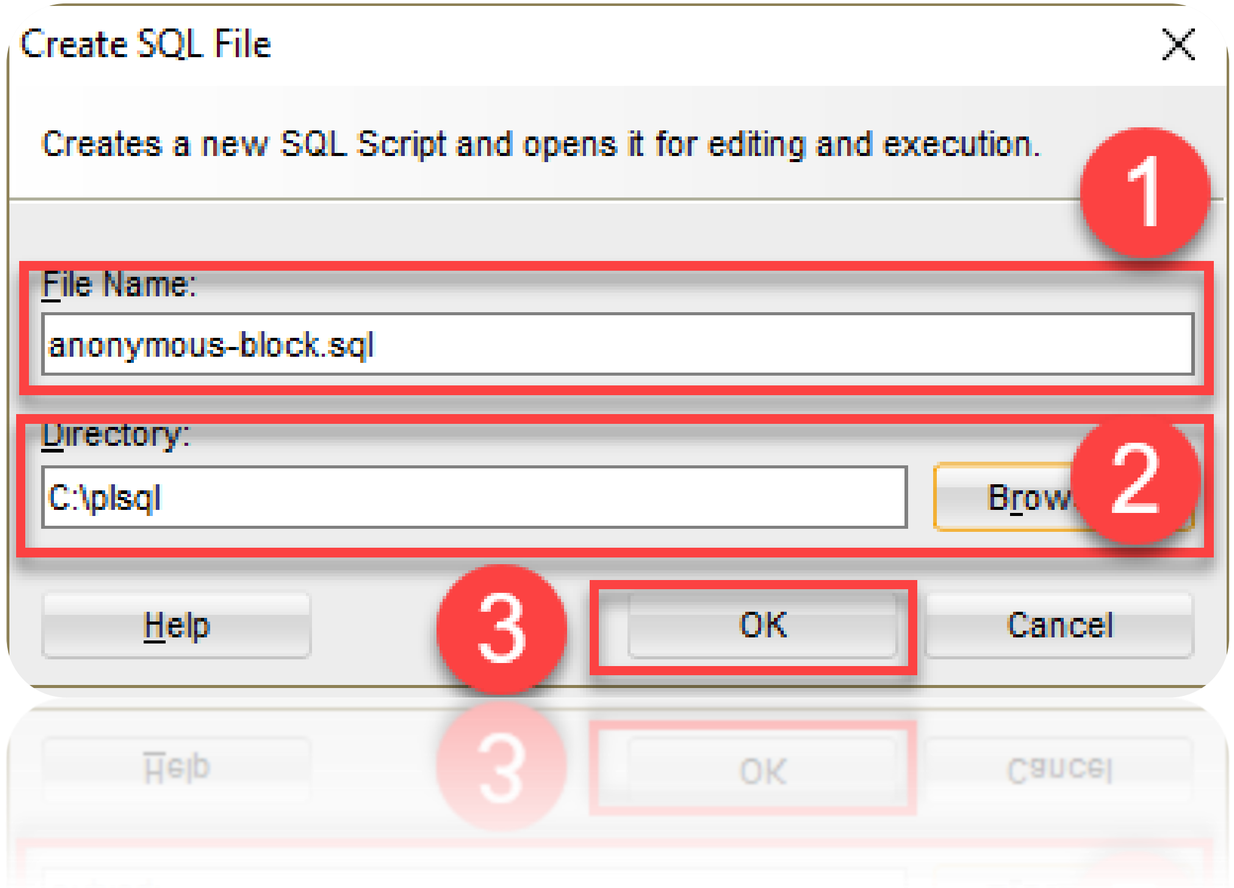
```
set SERVEROUTPUT ON;

BEGIN
DBMS_OUTPUT.PUT_LINE('Welcome In Tahaluf');
DBMS_OUTPUT.PUT_LINE('2021');
DBMS_OUTPUT.PUT('Mutaz');
DBMS_OUTPUT.PUT(' Makhatreh');
dbms_output.new_line();
End;
```

Execute a PL/SQL anonymous block using SQL Developer:

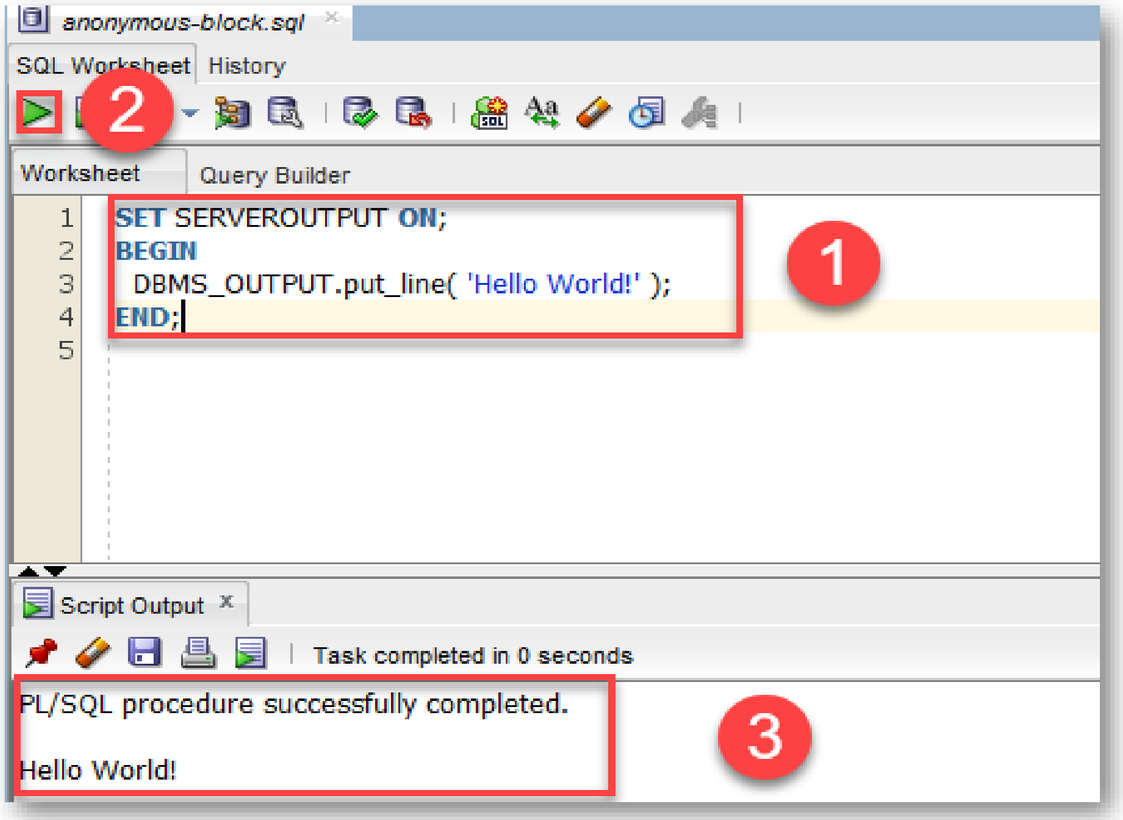


Execute a PL/SQL anonymous block using SQL Developer:



Harmony IT Solution

Execute a PL/SQL anonymous block using SQL Developer:



Example:

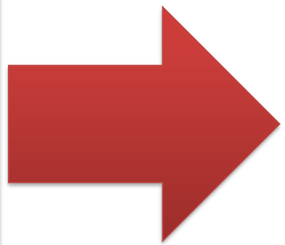
```
set SERVEROUTPUT ON;


declare
My_message varchar(50):='Welcome in tahaluf';
begin
Dbms_output.put_line(My_message);
End;
```

Harmony IT Solution

Example:

```
declare
My_number number ;
begin
My_number := 10/0;
DBMS_OUTPUT.PUT_LINE(My_number);
EXCEPTION
when zero_divide then
DBMS_OUTPUT.PUT_LINE(SQLERRM);
End;
```



```
PL/SQL procedure successfully completed.


ORA-01476: divisor is equal to zero
```

Harmony IT Solution

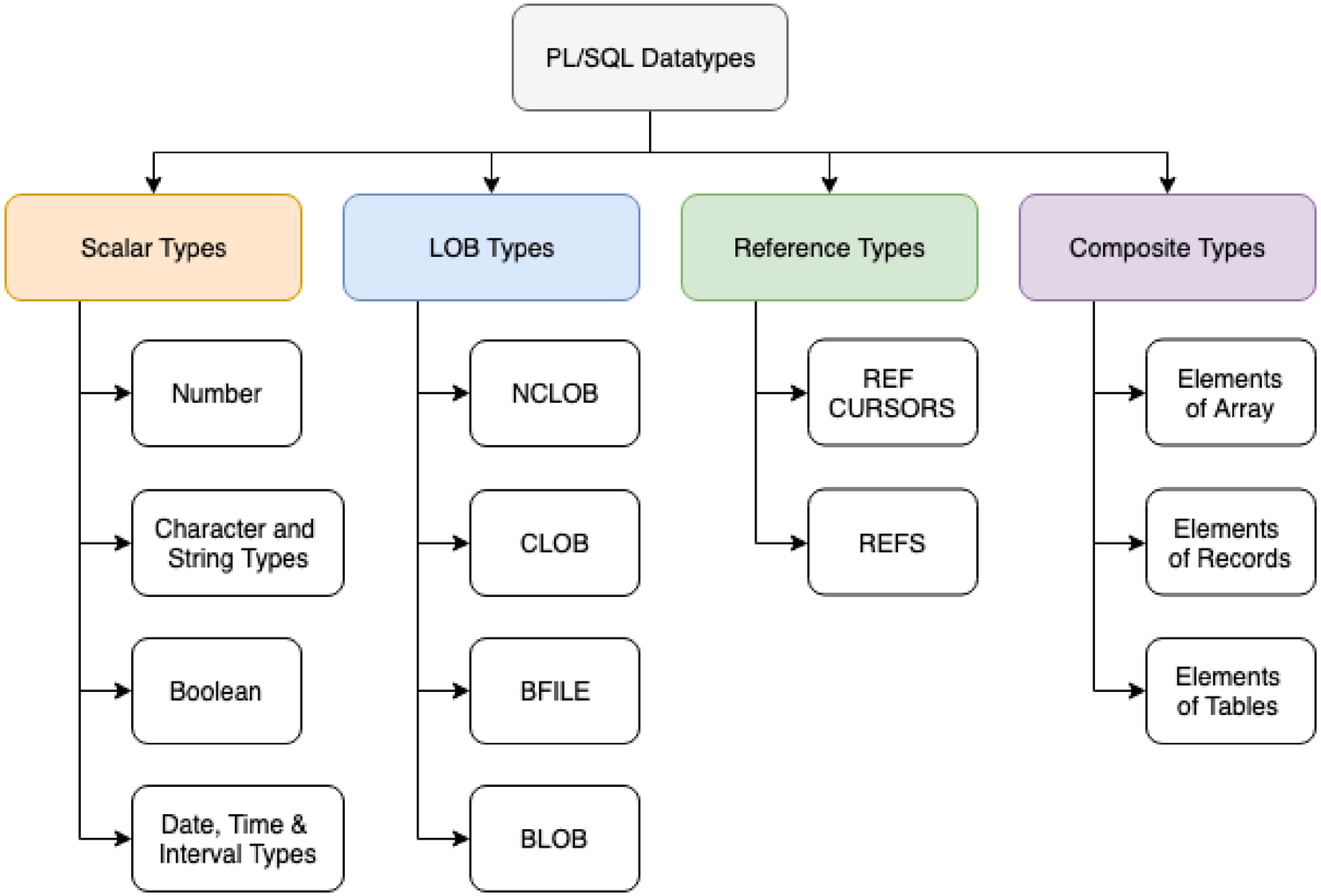
Data Types

- Each value in **PL/SQL** such as a constant, variable and parameter has a data type that determines the storage format, valid values, and allowed operations.

- PL/SQL has two kinds of data types: **scalar** and **composite**.

    1. **scalar** types are types that store single values such as **number, Boolean**, **character**, and **datetime** .

    2. **composite** types are types that store multiple values, for example, **record** and **collection**.



Variables

- **Variable** is named storage location that stores a value of a particular data type. The value of the variable changes through the program. Before using a variable, you must declare it in the declaration section of a block.

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

In the last syntax:

- **First**, specify the name of the variable. The name of the variable should be as descriptive as possible, e.g., **l_Counter**, **l_Name**, and **l_Age**.

- **Second**, choose an appropriate data type for the variable, depending on the kind of value which you want to store, for example, **number**, **character**, **Boolean**, and **datetime**.

- **By convention**, **local** variable names should start with **l_** and **global** variable names should have a prefix of **g_** .

Default values:

PL/SQL allows you to set a default value for a variable at the declaration time. To assign a default value to a variable, you use the assignment operator (:=) or the DEFAULT keyword.

```
declare
l_MyPhone varchar2(100):='Iphone 13 pro max';
begin
DBMS_OUTPUT.PUT_LINE(l_MyPhone);
END;
```

=

```
declare
l_MyPhone varchar2(100) default 'Iphone 13 pro max';
begin
DBMS_OUTPUT.PUT_LINE(l_MyPhone);
END;
```

NOT NULL constraint:

- If you impose the **NOT NULL** constraint on a value, then the variable cannot accept a NULL value.

- Besides, a variable declared with the NOT NULL must be initialized with a non-null value.

Note that PL/SQL treats a zero-length string as a NULL value.

```
declare
l_MyPhone varchar2(100) not null:='Iphone 13 pro max';
begin
l_MyPhone:=null;
--l_MyPhone:='';
DBMS_OUTPUT.PUT_LINE(l_MyPhone);
END;
```

ORA-06502: PL/SQL:
numeric or value error

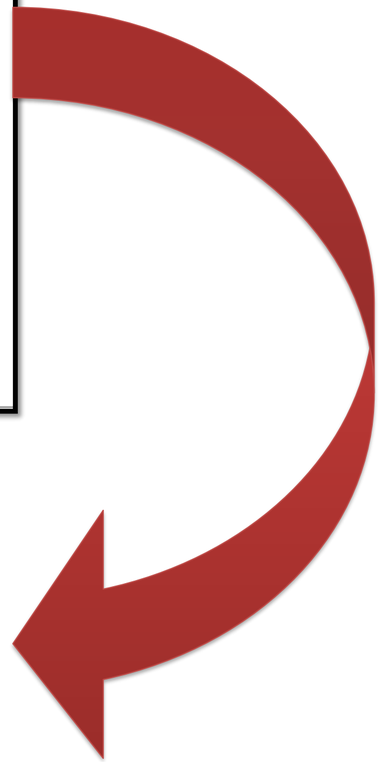
Anchored declarations:

- Typically, you declare a variable and select a value from a table column to this variable.
- If the data type of the table column changes, you must adjust the program to make it work with the new type.
- PL/SQL allows you to declare a variable whose data type anchor to a table column or another

Use Department Table:

DEPARTMENTID	DEPARTMENTNAME	DEPARTMENTLOCATION
2	Cis	Aiman
3	CPE	Shariah
4	HR	iordan
5	AI	Al ain
1	CS	Dubai

```
declare
DName Department.DepartmentName%TYPE; --Anchored decleration
DLocation Department.departmentlocation%TYPE;
begin
select
DepartmentName,departmentlocation
into
DName,DLocation
from
Department
where
DEPARTMENTID=5;
DBMS_OUTPUT.PUT_LINE(DName|| '---->'||dlocation);
END;
```



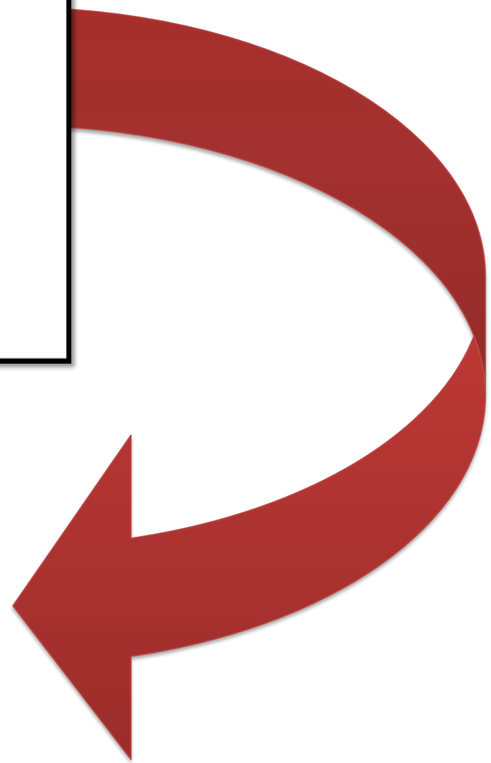
```
AI---->Al ain
```

Harmony IT Solution

Use Student Table:

NAME	MARK	ID	GENDER
fares	70	4	M
Ali	77	5	M
Mutaz	92	1	M
ahmad	85	2	M
Sara	98	3	F

```
declare
l_name student.name%type;
l_min student.mark%type;
l_max student.mark%type;
l_avg student.mark%type;
begin
select name into l_name from student where id =3;
select min(mark),max(mark),avg(mark)
into l_min,l_max,l_avg
from student;
DBMS_OUTPUT.PUT_LINE('Student Name: ' ||l_name);
DBMS_OUTPUT.PUT_LINE('Min mark: '|| l_min);
DBMS_OUTPUT.PUT_LINE('Max mark: '|| l_max);
DBMS_OUTPUT.PUT_LINE('Avg mark: '|| l_avg);
END;
```



```
Student Name: Sara
Min mark: 70
Max mark: 98
Avg mark: 84.4
```

- PL/SQL **comments** allow you to describe the purpose of a line or a block of PL/SQL code.

- When compiling the PL/SQL code, the compiler ignores comments.

- **However**, you should always use comments to make your code more readable and to help you and other developers understand it better in the future.

PL/SQL has two comment styles: **single-line** and **multi-line** comments:

- **Single-line comments:** A single-line comment starts with a double hyphen (**--**) that can appear anywhere on a line and extends to the end of the **line.**

```
-- valued added tax 10%
```

- **Multi-line comments:** A multi-line comment starts with a slash-asterisk (**/***) and ends with an asterisk-slash (***/**), and can span multiple lines.

```
/*
     This is a multi-line commet
     that can span multiple lines
*/
```

Harmony IT Solution