Variables

This lesson sheds light on how to declare and use variables in a stored procedure and also discusses the scope of variables.

We'll cover the following

Variables

A variable is nothing but a data object with a name associated to it. Variables help store a user-defined, temporary value which can be referred to in subsequent statements.

A variable must be declared before it can be used in the code. The **DECLARE** keyword is used to declare variables by providing the data type and an optional default value. If **DEFAULT** is not used at the time of declaration, then the variable will have NULL value.

Assignment of a value to a variable is done using the **SET** keyword. Another way to assign values to a variable is to use it in a query with a **SELECT INTO** statement.

The scope of a variable defines its lifetime after which it becomes unavailable. The scope of a variable created in a stored procedure is local meaning that it will not be accessible after the **END** statement of the stored procedure. Inside the stored procedure, the scope of a variable depends on where it is declared. This means we can have multiple variables of the same name in a stored procedure as long as they have different scopes.

Syntax

> DECLARE **VarName DataType** (**VarLength**) [DEFAULT **DefaultValue**];

> SET **VarName** = value;

> SELECT **ColName**
>
> INTO **VarName**
>
> FROM **TableName**;

> Connect to the terminal below by clicking in the widget. Once connected, the command line prompt will show up. Enter or copy and paste the command **./DataJek/Lessons/52lesson.sh** and wait for the MySQL prompt to start-up.

```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1
DELIMITER **
CREATE PROCEDURE Summary()
BEGIN
        DECLARE TotalM, TotalF INT DEFAULT 0;
    DECLARE AvgNetWorth DEC(6,2) DEFAULT 0.0;

    SELECT COUNT(*) INTO TotalM
    FROM Actors
    WHERE Gender = 'Male';

    SELECT COUNT(*) INTO TotalF
    FROM Actors
    WHERE Gender = 'Female';

    SELECT AVG(NetWorthInMillions) INTO AvgNetWorth
    FROM Actors;

    SELECT TotalM, TotalF, AvgNetWorth;
END**
DELIMITER ;

-- Query 2
CALL Summary();
```

● Terminal

1. To demonstrate how to create a variable, let's assume we are interested in storing the average net worth of all actors in our **Actors** table. We can create a variable **AvgNetWorth** as follows:

   ```
   DECLARE AvgNetWorth DEC(6,2) DEFAULT 0.0;
   ```

   In a stored procedure, this statement creates a variable **AvgNetWorth** of decimal data type and assigns it a default value of 0.0. The values 6 and 2 in parentheses mean that the number can be 6 digits long and the decimal part will have 2 digits.

2. We can create more than one variable in a single **DECLARE** statement provided they have the same data type. Say, we want to store the total number of males and females from our Actors table in two variables **TotalM** and **TotalF**. This can be done using the following statement inside a stored procedure:

   ```
   DECLARE TotalM, TotalF INT DEFAULT 0;
   ```

   The default value of both the integers is set to 0.

3. Once the variables have been created, we can assign values to them in two ways. The first method uses the **SET** keyword.

   ```
   SET TotalM = 6;
   SET TotalF = 4;
   ```

   The values of variables **TotalM** and **TotalF** after assignment are 6 and 4 respectively.

   The second assignment method uses the **SELECT INTO** statement to assign a value to a variable using a query. We can use this method to calculate the average net worth from the **Actors** table and assign it to the **AvgNetWorth** variable from step 1:

   ```
   SELECT AVG(NetWorthInMillions)
   ```

```
INTO AvgNetWorth
FROM Actors;
```

Here we have used the **AVG** aggregate function to find the average value of the **NetWorthInMillions** column of the **Actors** table and then assign that value to the **AvgNetWorth** variable.

4. Now we will define a stored procedure that declares, assigns and displays value to variables as follows:

```
DELIMITER **

CREATE PROCEDURE Summary()
BEGIN
    DECLARE TotalM, TotalF INT DEFAULT 0;
    DECLARE AvgNetWorth DEC(6,2) DEFAULT 0.0;

    SELECT COUNT(*) INTO TotalM
    FROM Actors
    WHERE Gender = 'Male';

    SELECT COUNT(*) INTO TotalF
    FROM Actors
    WHERE Gender = 'Female';

    SELECT AVG(NetWorthInMillions)
    INTO AvgNetWorth
    FROM Actors;

    SELECT TotalM, TotalF, AvgNetWorth;
END**

DELIMITER ;
```

This stored procedure will calculate the total number of males and females in the **Actors** table as well as the average net worth of all actors and display the result when called. The variables are declared in the **BEGIN END** block and will be out of scope after the **END** statement is executed. To execute the **Summary** stored procedure run the following statement:

```
CALL Summary();
```