**What is Variable?**

In MS SQL, variables are the object which acts as a placeholder to a memory location. Variable hold single data value.

## Types of Variable: Local, Global

MS SQL has two types of variables:

1. Local variable
2. Global variable.

However, the user can only create a local variable.

### Local variable:

* A user declares the local variable.
* By default, a local variable starts with **@.**
* Every local variable scope has the restriction to the **current batch or procedure** within any given session.

### Global variable:

* The system maintains the global variable**.**A user cannot declare them.
* The global variable starts with **@@**
* It stores **session related information**.

## How to DECLARE a variable

* Before using any variable in batch or procedure, you need to **declare the variable.**
* DECLARE command is used to DECLARE variable which acts as a placeholder for the memory location.
* Only once the declaration is made, a variable can be used in the subsequent part of batch or procedure.

**TSQL Syntax:**

DECLARE { @LOCAL\_VARIABLE[AS] data\_type [ = value ] }

**Rules:**

* Initialization is an optional thing while declaring.
* By default, DECLARE initializes variable to NULL.
* Using the keyword 'AS' is optional.
* To declare more than one local variable, use a comma after the first local variable definition, and then define the next local variable name and data type.

### Examples of Declaring a variable:

**Query: With 'AS'**

DECLARE @COURSE\_ID AS INT;

**Query: Without 'AS'**

DECLARE @COURSE\_NAME VARCHAR (10);

**Query: DECLARE two variables**

DECLARE @COURSE\_ID AS INT, @COURSE\_NAME VARCHAR (10);

## Assigning a value to a VARIABLE

You can assign a value to a variable in the following**three**ways**:**

1. During variable declaration using DECLARE keyword.
2. Using SET
3. Using SELECT

Let's have a look at all three ways in detail:

### During variable declaration using DECLARE keyword.

**T-SQL Syntax:**

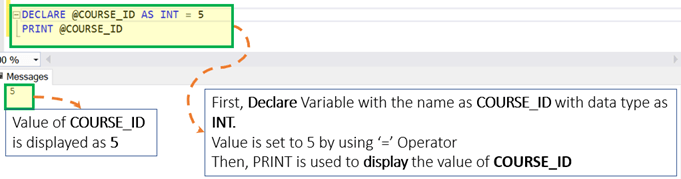
DECLARE { @Local\_Variable [AS] Datatype [ = value ] }

Here, after datatype we can use '=' followed by value to be assigned

**Query:**

DECLARE @COURSE\_ID AS INT = 5

PRINT @COURSE\_ID



### Using SET

Sometimes we want to keep declaration and initialization separate. SET can be used to assign values to the variable, post declaring a variable. Below are the different ways to assign values using SET:

**Example**: Assigning a value to a variable using SET

**Syntax:**

DECLARE @Local\_Variable <Data\_Type>

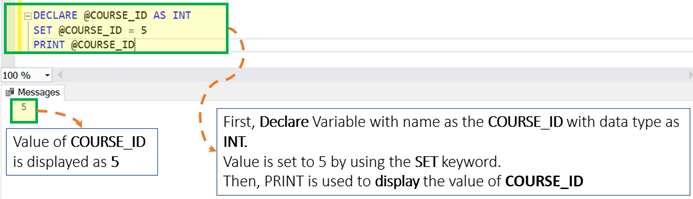
SET @Local\_Variable = <Value>

**Query:**

DECLARE @COURSE\_ID AS INT

SET @COURSE\_ID = 5

PRINT @COURSE\_ID



**Example**: Assign a value to **multiple variables** using SET.

**Syntax:**

DECLARE @Local\_Variable \_1 <Data\_Type>, @Local\_Variable\_2 <Data\_Type>,

SET @Local\_Variable\_1 = <Value\_1>

SET @Local\_Variable\_2 = <Value\_2>

**Rule:**OneSET Keyword can be used to assign a value to only **one variable**.

**Query:**

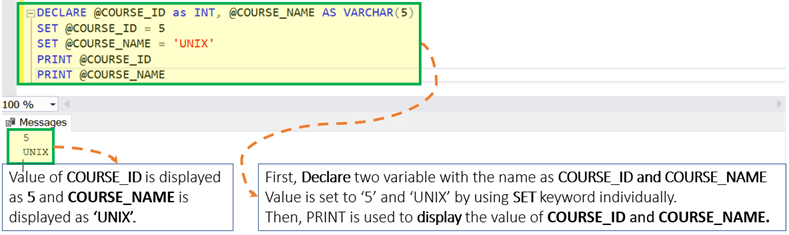
DECLARE @COURSE\_ID as INT, @COURSE\_NAME AS VARCHAR(5)

SET @COURSE\_ID = 5

SET @COURSE\_NAME = 'UNIX'

PRINT @COURSE\_ID

PRINT @COURSE\_NAME



**Example**: Assigning a value to a variable with a **Scalar Subquery** using SET

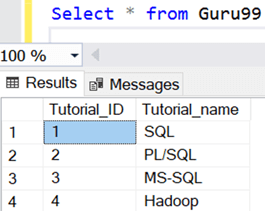
**Syntax:**

DECLARE @Local\_Variable\_1 <Data\_Type>, @Local\_Variable\_2 <Data\_Type>,SET @Local\_Variable\_1 = (SELECT <Column\_1> from <Table\_Name> where <Condition\_1>)

**Rules:**

* Enclose the query in parenthesis.
* The query should be a scalar query. A scalar query is a query with results as just one row and one column. Otherwise, the query will throw an error.
* If the query returns zero rows, then the variable is set to EMPTY, i.e., NULL.

**Assumption:** Assume that we have the table as 'Guru99' with two columns as displayed below:



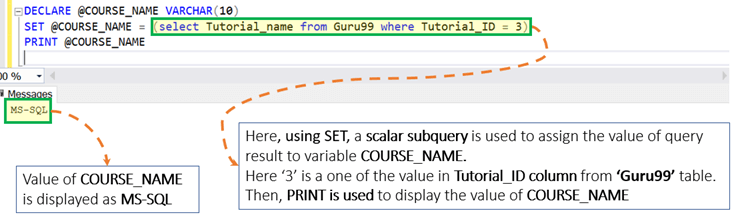
We will use 'Guru99' table in the further tutorials

**Example 1:**When subquery return one row as a result.

DECLARE @COURSE\_NAME VARCHAR (10)

SET @COURSE\_NAME = (select Tutorial\_name from Guru99 where Tutorial\_ID = 3)

PRINT @COURSE\_NAME



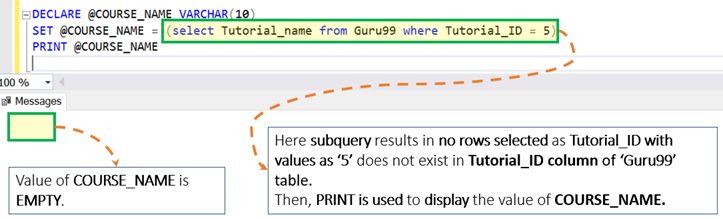
**Example 2: When subquery returns zero row as a result**

DECLARE @COURSE\_NAME VARCHAR (10)

SET @COURSE\_NAME = (select Tutorial\_name from Guru99 where Tutorial\_ID = 5)

PRINT @COURSE\_NAME

In this particular case, the variable value is EMPTY, i.e., NULL.



### USING SELECT

Just like SET, we can also use SELECT to assign values to the variables, post declaring a variable using DECLARE. Below are different ways to assign a value using SELECT:

**Example**: Assigning a value to a variable using SELECT

**Syntax:**

DECLARE @LOCAL\_VARIABLE <Data\_Type>

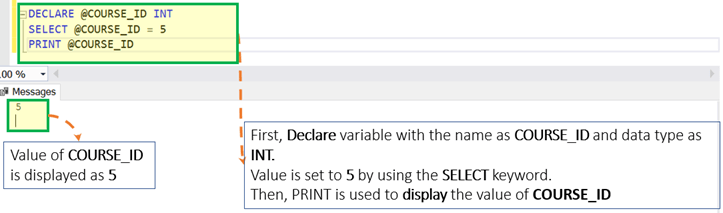
SELECT @LOCAL\_VARIABLE = <Value>

**Query:**

DECLARE @COURSE\_ID INT

SELECT @COURSE\_ID = 5

PRINT @COURSE\_ID



**Example**: Assigning a value to multiple variable using SELECT

**Syntax:**

DECLARE @Local\_Variable \_1 <Data\_Type>, @Local\_Variable \_2 <Data\_Type>,SELECT @Local\_Variable \_1 = <Value\_1>, @Local\_Variable \_2 = <Value\_2>

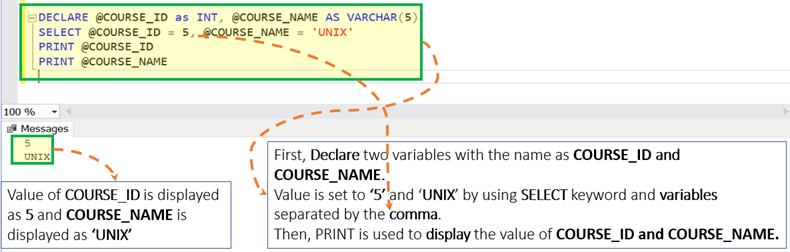
**Rules:** Unlike SET, SELECT can be used to assign a value **to multiple variables**separated by the**comma**.

DECLARE @COURSE\_ID as INT, @COURSE\_NAME AS VARCHAR(5)

SELECT @COURSE\_ID = 5, @COURSE\_NAME = 'UNIX'

PRINT @COURSE\_ID

PRINT @COURSE\_NAME



**Example**: Assigning the value to a variable with a Subquery using SELECT

**Syntax:**

DECLARE @Local\_Variable\_1 <Data\_Type>, @Local\_Variable \_2 <Data\_Type>,SELECT @Local\_Variable \_1 = (SELECT <Column\_1> from <Table\_name> where <Condition\_1>)

**Rules:**

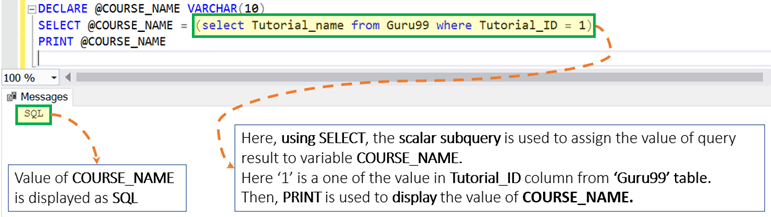
* Enclose the query in Parenthesis.
* The query should be a scalar query. The scalar query is the query with the result as one row and one column. Otherwise, the query will throw an error.
* If the query returns zero rows, then the variable is EMPTY, i.e., NULL.
* Reconsider our 'Guru99' table

**Example 1:**When subquery return one row as a result.

DECLARE @COURSE\_NAME VARCHAR (10)

SELECT @COURSE\_NAME = (select Tutorial\_name from Guru99 where Tutorial\_ID = 1)

PRINT @COURSE\_NAME



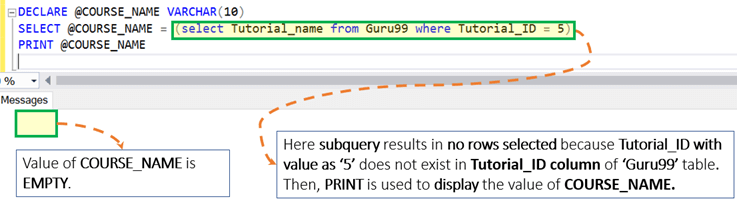
**Example 2:**When subquery return zero row as a result

DECLARE @COURSE\_NAME VARCHAR (10)

SELECT @COURSE\_NAME = (select Tutorial\_name from Guru99 where Tutorial\_ID = 5)

PRINT @COURSE\_NAME

In this particular case, the variable is to EMPTY, i.e., NULL.



**Example 3:**Assign a value to a variable with a regular SELECT statement.

**Syntax:**

DECLARE @Local\_Variable \_1 <Data\_Type>, @Local\_Variable \_2 <Data\_Type>,SELECT @Local\_Variable \_1 = <Column\_1> from <Table\_name> where <Condition\_1>

**Rules:**

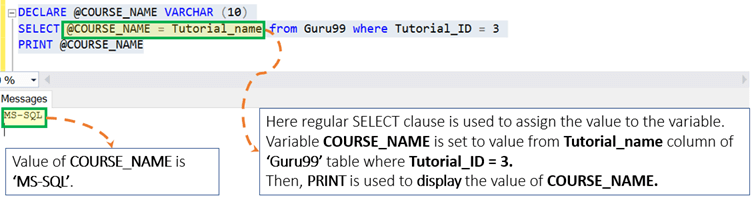
* Unlike SET, if the query results in multiple rows then the variable value is set to the value of the last row.
* If the query returns zero rows, then the variable is set to EMPTY, i.e., NULL.

**Query 1: The query returns one row.**

DECLARE @COURSE\_NAME VARCHAR (10)

SELECT @COURSE\_NAME = Tutorial\_name from Guru99 where Tutorial\_ID = 3

PRINT @COURSE\_NAME



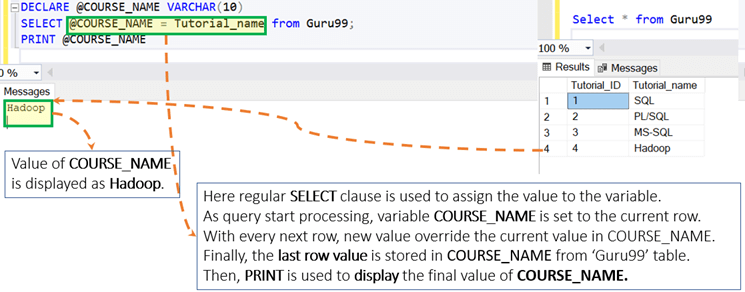
**Query 2: The query returns multiple rows.**

DECLARE @COURSE\_NAME VARCHAR (10)

SELECT @COURSE\_NAME = Tutorial\_name from Guru99

PRINT @COURSE\_NAME

In this special case,variable value is **set to the value of the last row**.



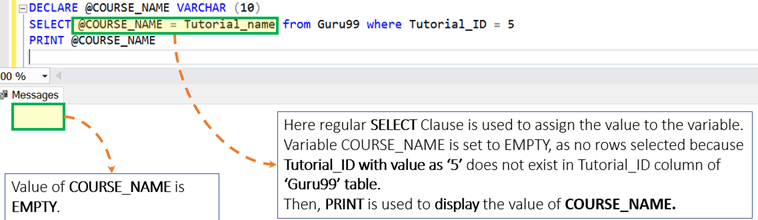
**Query 3: The query returns zero rows.**

DECLARE @COURSE\_NAME VARCHAR (10)

SELECT @COURSE\_NAME = Tutorial\_name from Guru99 where Tutorial\_ID = 5

PRINT @COURSE\_NAME

In this particular case, the variable is EMPTY, i.e., NULL.



### Other Examples

Using variable in the query

**Query:**

DECLARE @COURSE\_ID Int = 1

SELECT \* from Guru99 where Tutorial\_id = @COURSE\_ID

