Variable SQL Server: Déclarer, Définir, Sélectionner, Global, Local [Exemples TSQL]

Quelle est la variable?

Dans MS SQL, les variables sont l'objet qui sert d'espace réservé à un emplacement mémoire. Variable contient une valeur de donnée unique.

Dans ce tutoriel, vous apprendrez:

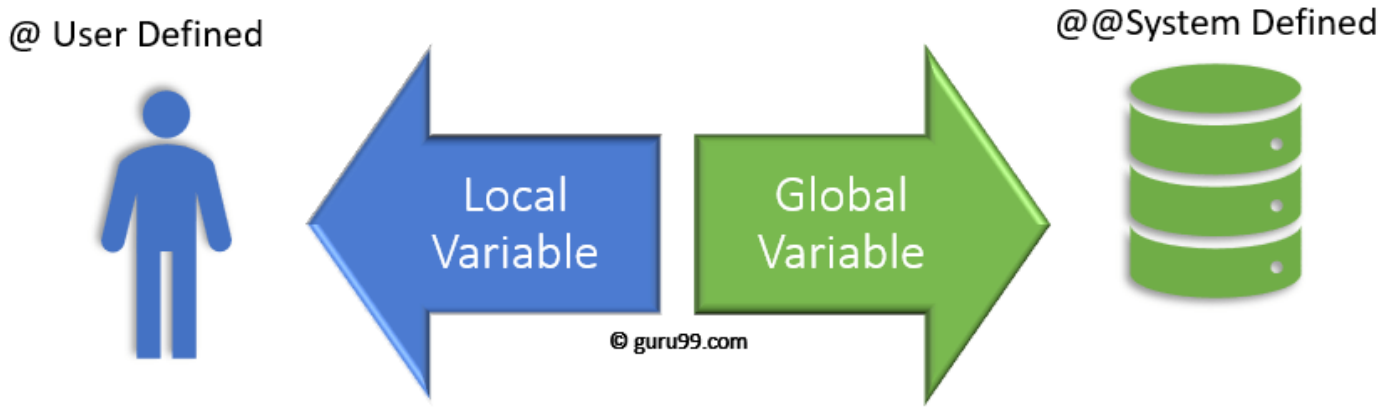
Types de variable: Local, Global

MS SQL a deux types de variables:

1. Variable locale
2. Variable globale.

Cependant, l'utilisateur ne peut créer qu'une variable locale.

La figure ci-dessous explique deux types de variables disponibles dans le serveur MS SQL.



(/images/1/030819_0723_SQLServerVa1.png)
Type de variables dans SQL Server

Variable locale:

- Un utilisateur déclare la variable locale.
- Par défaut, une variable locale commence par **@.**
- La portée de chaque variable locale est limitée au **lot ou à la procédure en cours** dans une session donnée.

Variable globale:

- Le système gère la variable globale **.** Un utilisateur ne peut pas les déclarer.
- La variable globale commence par **@@**
- Il stocke **les informations relatives à la session** .

Comment déclarer une variable

- Avant d'utiliser une variable dans un lot ou une procédure, vous devez **déclarer la 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.

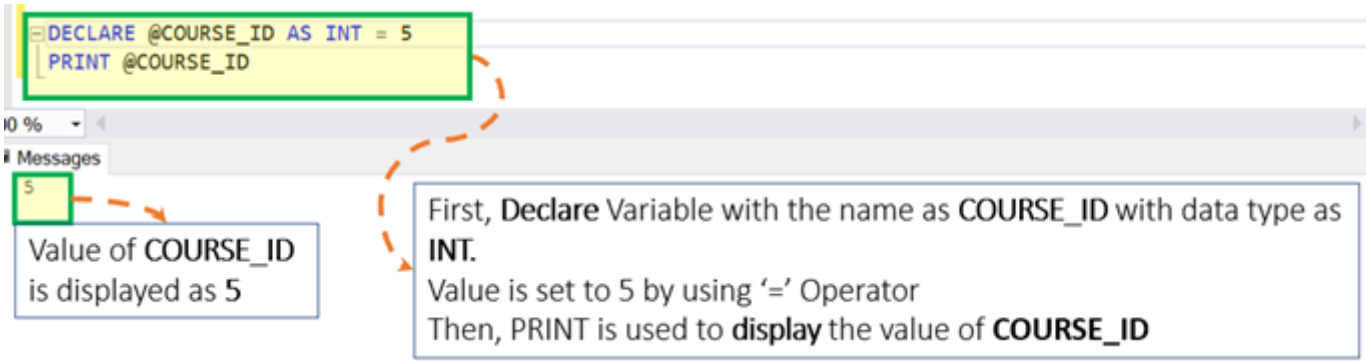
T-SQL Syntax:

```
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
```



(/images/1/030819_0723_SQLServerVa2.png)

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:

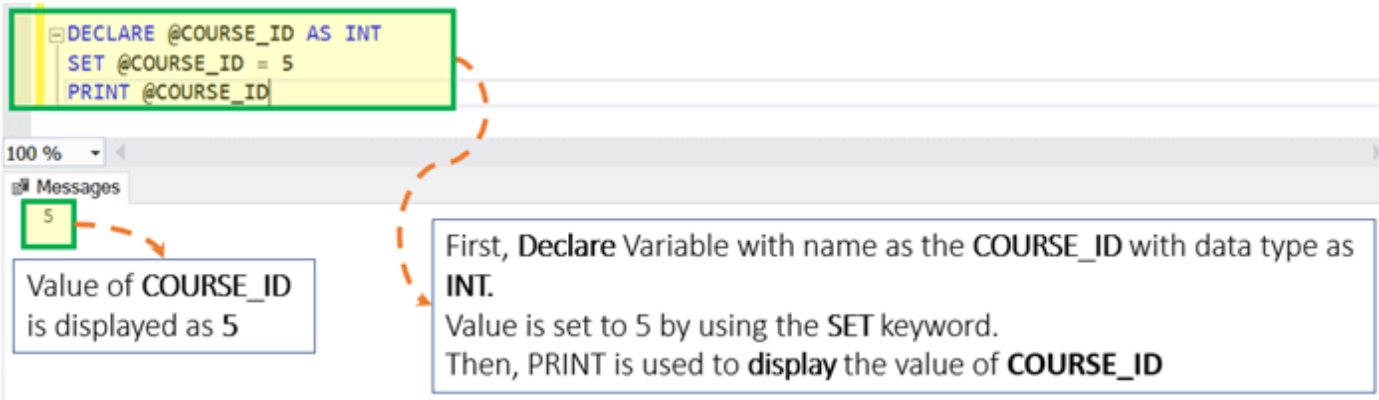
Example: Assigning a value to a variable using SET

Syntax:

```
DECLARE @Local_Variable <Data_Type>
SET @Local_Variable =  <Value>
```

Query:

```
DECLARE @COURSE_ID AS INT
SET @COURSE_ID = 5
PRINT @COURSE_ID
```



(/images/1/030819_0723_SQLServerVa3.png)

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

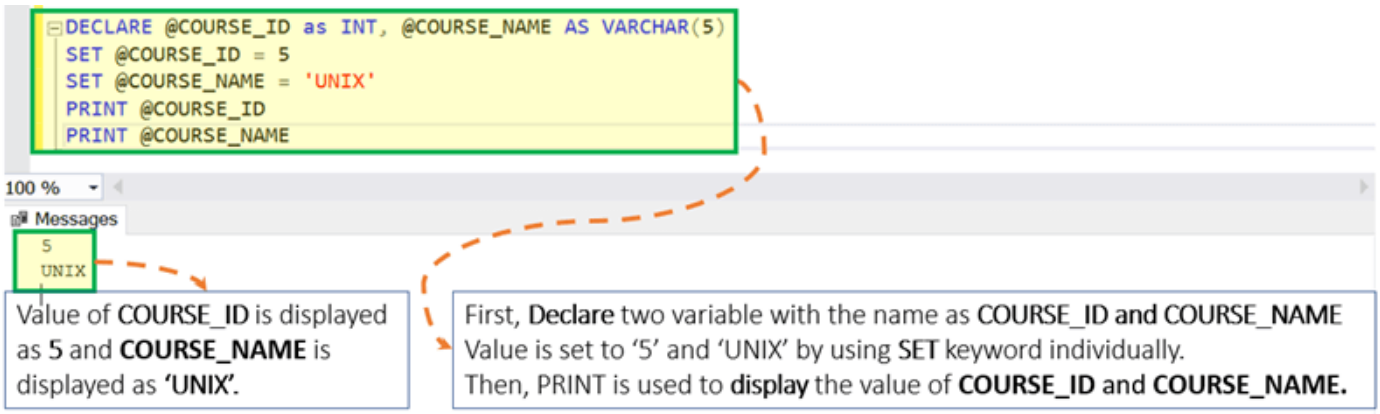
Syntax:

```
DECLARE @Local_Variable _1 <Data_Type>, @Local_Variable_2 <Data_Type>,
SET @Local_Variable_1 = <Value_1>
SET @Local_Variable_2 = <Value_2>
```

Rule: One SET 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
```



(/images/1/030819_0723_SQLServerVa4.png)

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

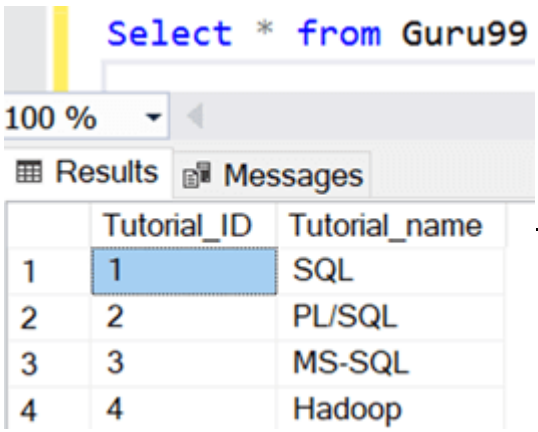
Syntax:

```
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:

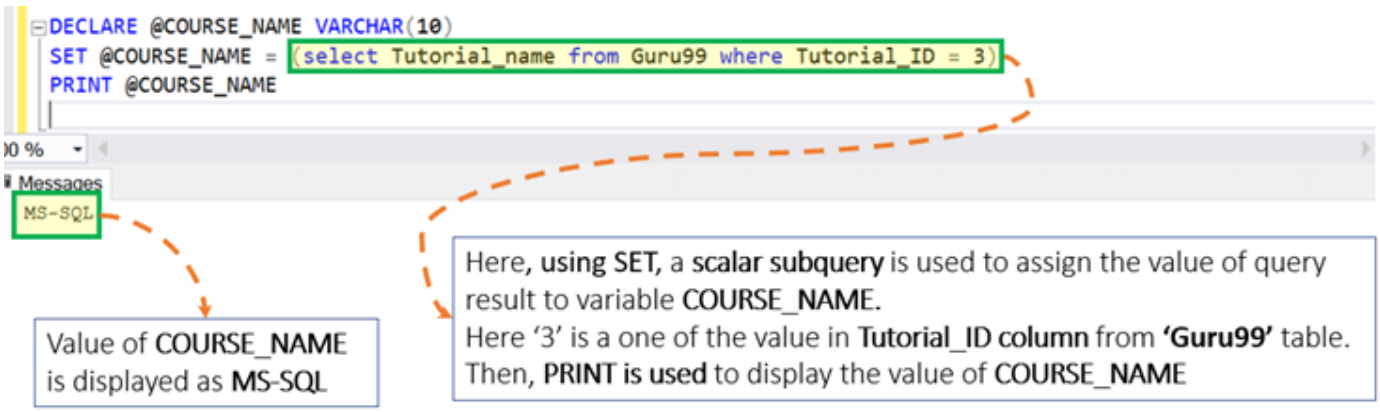


(/images/1/030819_0723_SQLServerVa5.png)

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
```

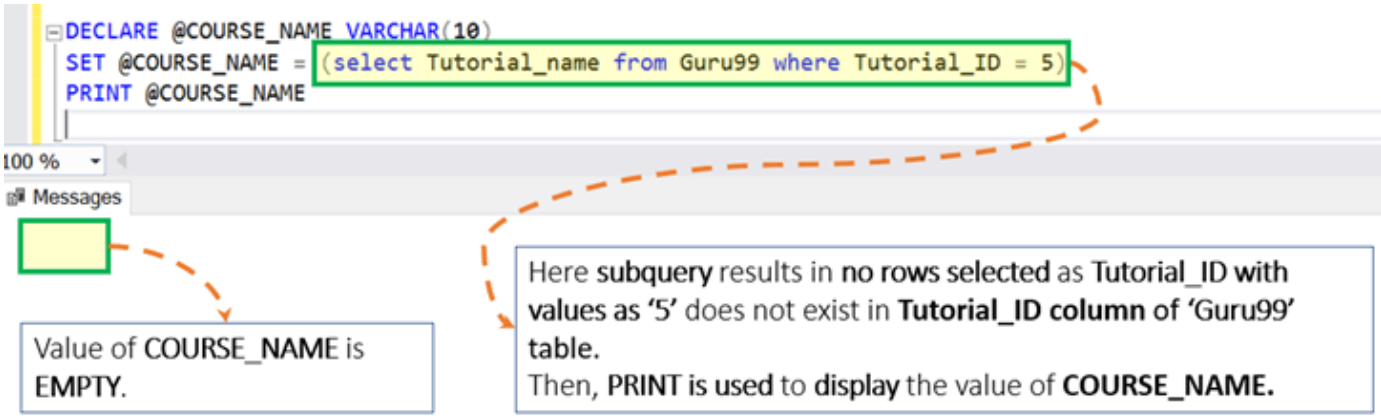


(/images/1/030819_0723_SQLServerVa6.png)

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.



(/images/1/030819_0723_SQLServerVa7.png)

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:

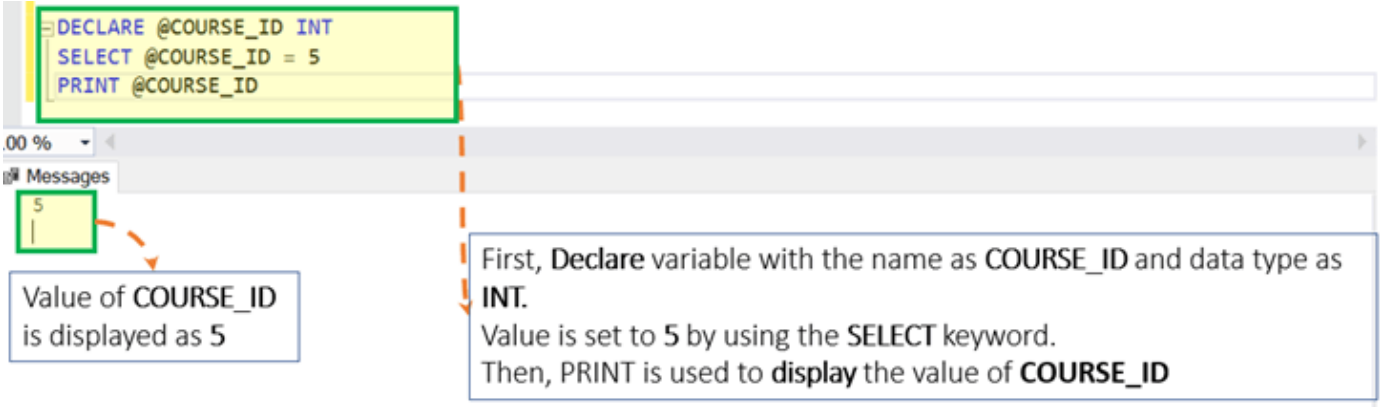
Example: Assigning a value to a variable using SELECT

Syntax:

```
DECLARE @LOCAL_VARIABLE <Data_Type>
SELECT @LOCAL_VARIABLE = <Value>
```

Query:

```
DECLARE @COURSE_ID INT
SELECT @COURSE_ID = 5
PRINT @COURSE_ID
```



(/images/1/030819_0723_SQLServerVa8.png)

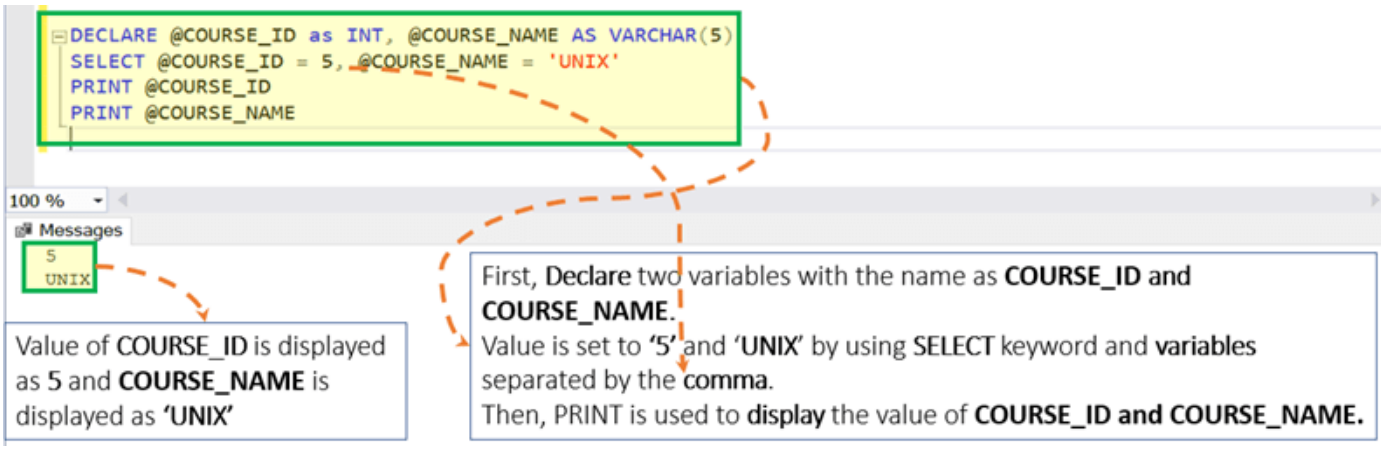
Example: Assigning a value to multiple variable using SELECT

Syntax:

```
DECLARE @Local_Variable _1 <Data_Type>, @Local_Variable _2 <Data_Type>,SELECT @Local_Variab
le _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
```



(/images/1/030819_0723_SQLServerVa9.png)

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

Syntax:

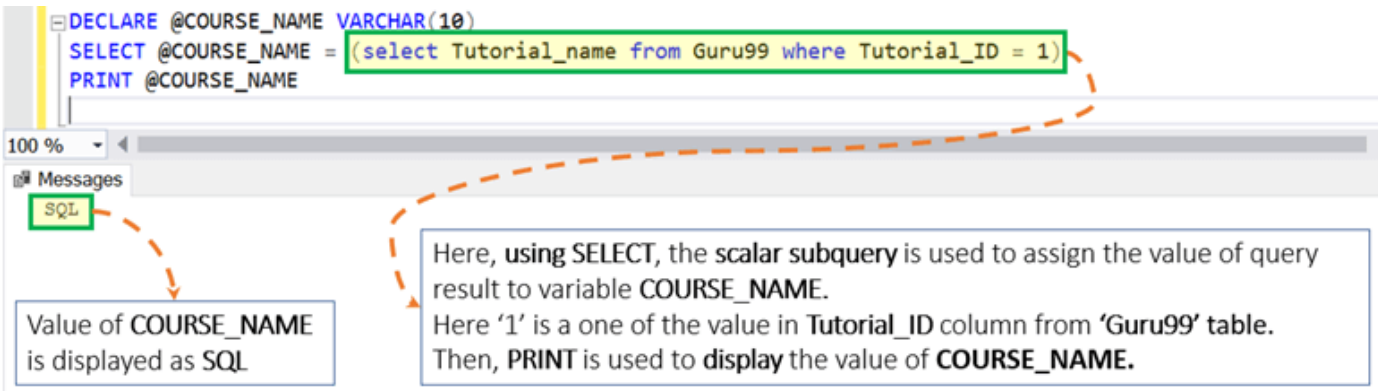
```
DECLARE @Local_Variable_1 <Data_Type>, @Local_Variable _2 <Data_Type>,SELECT @Local_Variabl
e _1 = (SELECT <Column_1> from <Table_name> where <Condition_1>)
```

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
```

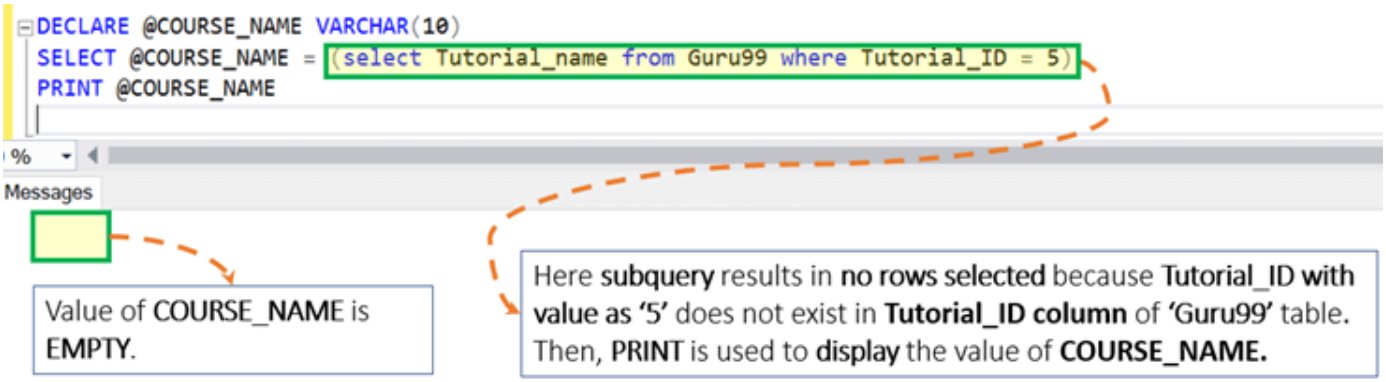


(/images/1/030819_0723_SQLServerVa10.png)

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.



(/images/1/030819_0723_SQLServerVa11.png)

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

Syntax:

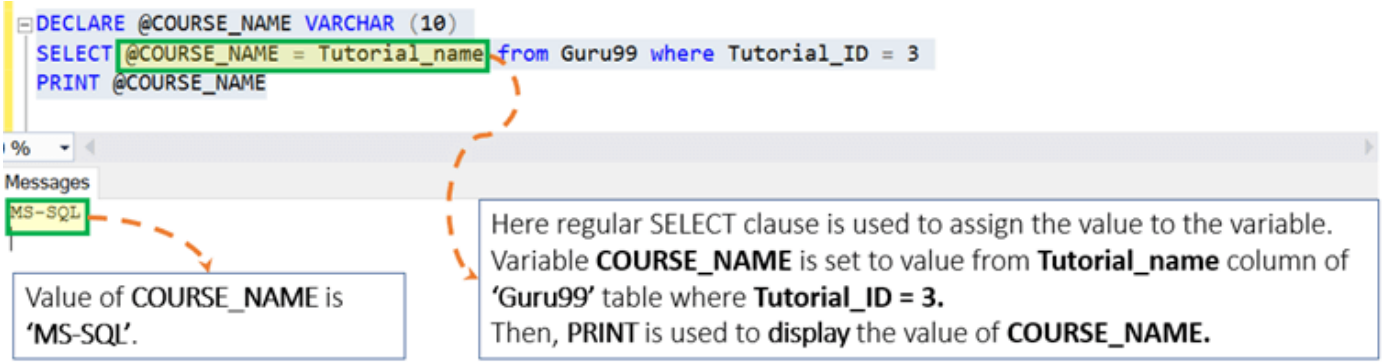
```
DECLARE @Local_Variable _1 <Data_Type>, @Local_Variable _2 <Data_Type>,SELECT @Local_Variab
le _1 = <Column_1> from <Table_name> where <Condition_1>
```

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
```

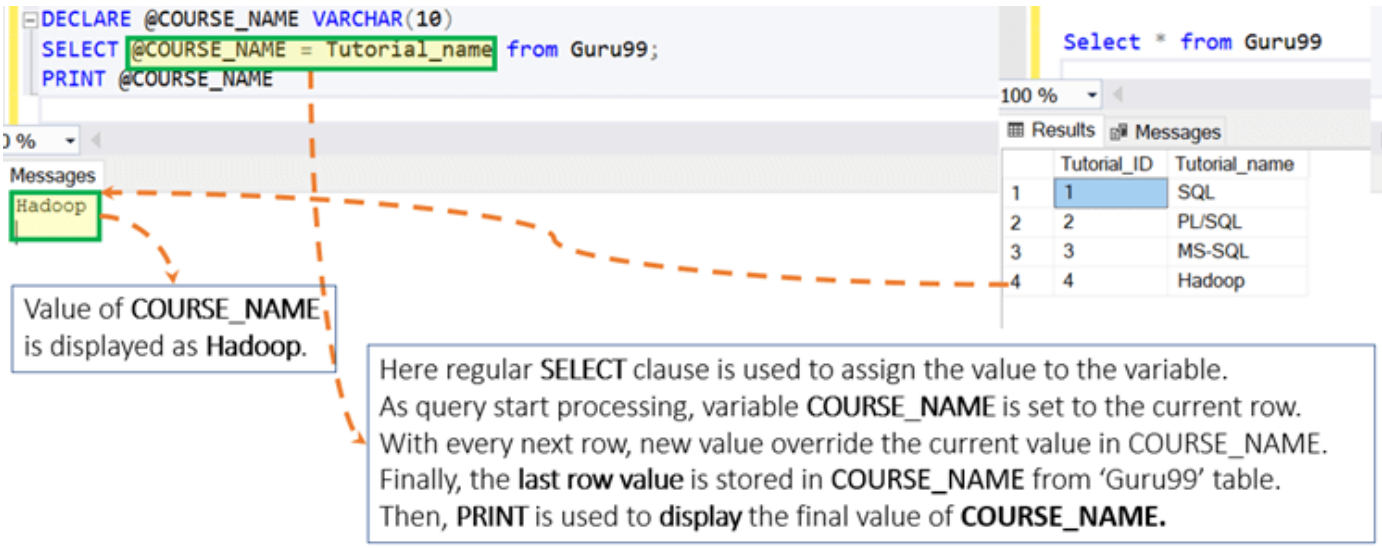


(/images/1/030819_0723_SQLServerVa12.png)

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**.

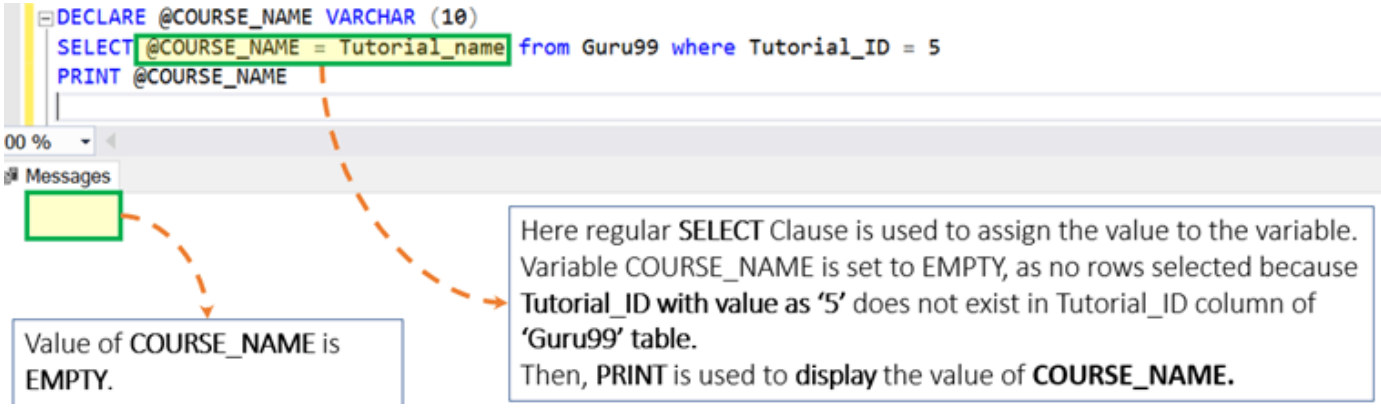


(/images/1/030819_0723_SQLServerVa13.png)

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.



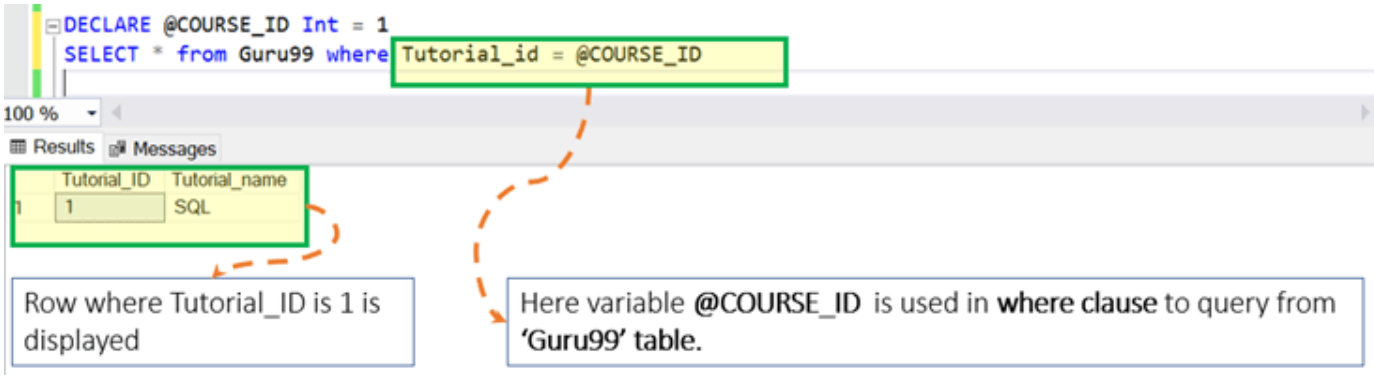
(/images/1/030819_0723_SQLServerVa14.png)

Other Examples

Using variable in the query

Query:

```
DECLARE @COURSE_ID Int = 1
SELECT * from Guru99 where Tutorial_id = @COURSE_ID
```



(/images/1/030819_0723_SQLServerVa15.png)

Interesting Facts!

- A local variable can be displayed using PRINT as well as SELECT COMMAND
- Table Data type doesn't allow the use of 'AS' during declaration.
- SET complies with ANSI standards whereas SELECT does not.
- Creating a local variable with the name as @ is also allowed. We can declare it as, for example:

  ```
  'DECLARE @@ as VARCHAR (10)'
  ```

Summary:

- Variables are the object which acts as a placeholder.

- Two types of Variable exist: Local and Global
- We can assign the variable in the following three ways:
  While using DECLARE
  Using SET
  USING SELECT

Prev (/sql-server-datatype.html)

Report a Bug

Next (/sql-server-table-create-alter-drop.html)

Tutoriel SQL Server

11) SI… autre déclaration (/sql-server-if-else.html)

12) déclaration de cas (/sql-server-case.html)

(https://www.facebook.com/guru99com/)
(https://twitter.com/guru99com)
(https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqFQ)
(https://forms.aweber.com/form/46/724807646.htm)

À propos

À propos de nous (/about-us.html)
Annoncez avec nous (/advertise-us.html)
Écrivez pour nous (/become-an-instructor.html)
Contactez-nous (/contact-us.html)

Suggestion de carrière

(/best-sap-module.html)Test du logiciel (/software-testing-career-complete-guide.html)SAP Career Suggestion Tool en (/best-sap-module.html)
tant que carrière (/software-testing-career-complete-guide.html)

Intéressant

Livres à lire! (/books.html)EBook de (/ebook-pdf.html)
Blog (/blog/)
Quiz (/tests.html)
(/ebook-pdf.html)

Exécuter en ligne

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)