What is a variable

A variable is an object that holds a single value of a specific type e.g., [integer](https://www.sqlservertutorial.net/sql-server-basics/sql-server-int/), [date](https://www.sqlservertutorial.net/sql-server-basics/sql-server-date/), or [varying character string](https://www.sqlservertutorial.net/sql-server-basics/sql-server-varchar/).

We typically use variables in the following cases:

* As a loop counter to count the number of times a loop is performed.
* To hold a value to be tested by a control-of-flow statement such as WHILE.
* To store the value returned by a [stored procedure](https://www.sqlservertutorial.net/sql-server-stored-procedures/) or a function

Declaring a variable

To declare a variable, you use the DECLARE statement. For example, the following statement declares a variable named @model\_year:

DECLARE @model\_year SMALLINT;

Code language: SQL (Structured Query Language) (sql)

The DECLARE statement initializes a variable by assigning it a name and a data type. The variable name must start with the @ sign. In this example, the data type of the @model\_year variable is SMALLINT.

By default, when a variable is declared, its value is set to NULL.

Between the variable name and data type, you can use the optional AS keyword as follows:

DECLARE @model\_year AS SMALLINT;

Code language: SQL (Structured Query Language) (sql)

To declare multiple variables, you separate variables by commas:

DECLARE @model\_year SMALLINT,

@product\_name VARCHAR(MAX);

Code language: SQL (Structured Query Language) (sql)

Assigning a value to a variable

To assign a value to a variable, you use the SET statement. For example, the following statement assigns 2018 to the @model\_year variable:

SET @model\_year = 2018;

Code language: SQL (Structured Query Language) (sql)

Using variables in a query

The following [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement uses the @model\_year variable in the [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause to find the products of a specific model year:

SELECT

product\_name,

model\_year,

list\_price

FROM

production.products

WHERE

model\_year = @model\_year

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)

Now, you can put everything together and execute the following code block to get a list of products whose model year is 2018:

DECLARE @model\_year SMALLINT;

SET @model\_year = 2018;

SELECT

product\_name,

model\_year,

list\_price

FROM

production.products

WHERE

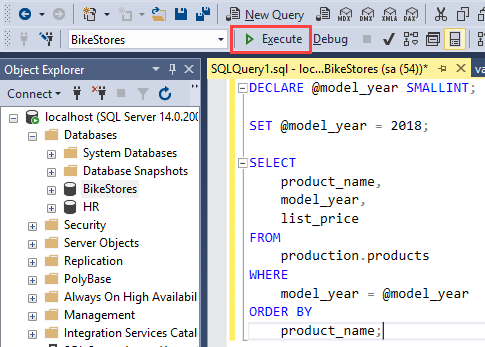
model\_year = @model\_year

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)

Note that to execute the code, you click the Execute button as shown in the following picture:



The following picture shows the output:



Storing query result in a variable

The following steps describe how to store the query result in a variable:

First, declare a variable named @product\_count with the integer data type:

DECLARE @product\_count INT;

Code language: SQL (Structured Query Language) (sql)

Second, use the SET statement to assign the query’s result set to the variable:

SET @product\_count = (

SELECT

COUNT(\*)

FROM

production.products

);

Code language: SQL (Structured Query Language) (sql)

Third, output the content of the @product\_count variable:

SELECT @product\_count;

Code language: SQL (Structured Query Language) (sql)

Or you can use the PRINT statement to print out the content of a variable:

PRINT @product\_count;

Code language: SQL (Structured Query Language) (sql)

or

PRINT 'The number of products is ' + CAST(@product\_count AS VARCHAR(MAX));

Code language: SQL (Structured Query Language) (sql)

The output in the messages tab is as follows:

The number of products is 204

Code language: SQL (Structured Query Language) (sql)

To hide the number of rows affected messages, you use the following statement:

SET NOCOUNT ON;

Code language: SQL (Structured Query Language) (sql)

Selecting a record into variables

The following steps illustrate how to declare two variables, assign a record to them, and output the contents of the variables:

First, declare variables that hold the product name and list price:

DECLARE

@product\_name VARCHAR(MAX),

@list\_price DECIMAL(10,2);

Code language: SQL (Structured Query Language) (sql)

Second, assign the column names to the corresponding variables:

SELECT

@product\_name = product\_name,

@list\_price = list\_price

FROM

production.products

WHERE

product\_id = 100;

Code language: SQL (Structured Query Language) (sql)

Third, output the content of the variables:

SELECT

@product\_name AS product\_name,

@list\_price AS list\_price;

Code language: SQL (Structured Query Language) (sql)

Stored Procedure Variables - assign a record to a variable

Accumulating values into a variable

The following stored procedure takes one parameter and returns a list of products as a string:

CREATE PROC uspGetProductList(

@model\_year SMALLINT

) AS

BEGIN

DECLARE @product\_list VARCHAR(MAX);

SET @product\_list = '';

SELECT

@product\_list = @product\_list + product\_name

+ CHAR(10)

FROM

production.products

WHERE

model\_year = @model\_year

ORDER BY

product\_name;

PRINT @product\_list;

END;

Code language: SQL (Structured Query Language) (sql)

In this stored procedure:

* First, we declared a variable named @product\_list with varying character string type and set its value to blank.
* Second, we selected the product name list from the products table based on the input @model\_year. In the select list, we accumulated the product names to the @product\_list variable. Note that the CHAR(10) returns the line feed character.
* Third, we used the PRINT statement to print out the product list.

The following statement executes the uspGetProductList stored procedure:

EXEC uspGetProductList 2018

Code language: SQL (Structured Query Language) (sql)

The following picture shows the partial output:

