SQL Server Cheatsheet

Contents

[Import a CSV file 1](#_Toc57964021)

[LEAD, LAG 10](#_Toc57964022)

[Restrict the number of rows returned 10](#_Toc57964023)

# Import a CSV file

Create a database file

From https://www.sqlshack.com/importing-and-working-with-csv-files-in-sql-server/

We’ll import data from a file which is available at <https://gist.githubusercontent.com/tijptjik/9408623/raw/b237fa5848349a14a14e5d4107dc7897c21951f5/wine.csv>

You can import a CSV file into a specific database. Let’s first create a dummy database named ‘Bar’ and try to import the CSV file into the Bar database.

**Step 0:** **Create the database**. In the SSMS, execute the following script to create the database:

|  |  |
| --- | --- |
| 1 | CREATE DATABASE Bar |

Now follow these steps to import CSV file into SQL Server Management Studio. This process is valid for all different types of flat files, however for the sake of this demonstration we will use a CSV file.

**Step 1: Selecting the database**

The first step is to go to the Object Explorer -> [Database] -> Tasks and click the “Import Flat File …” as shown in the following figure:

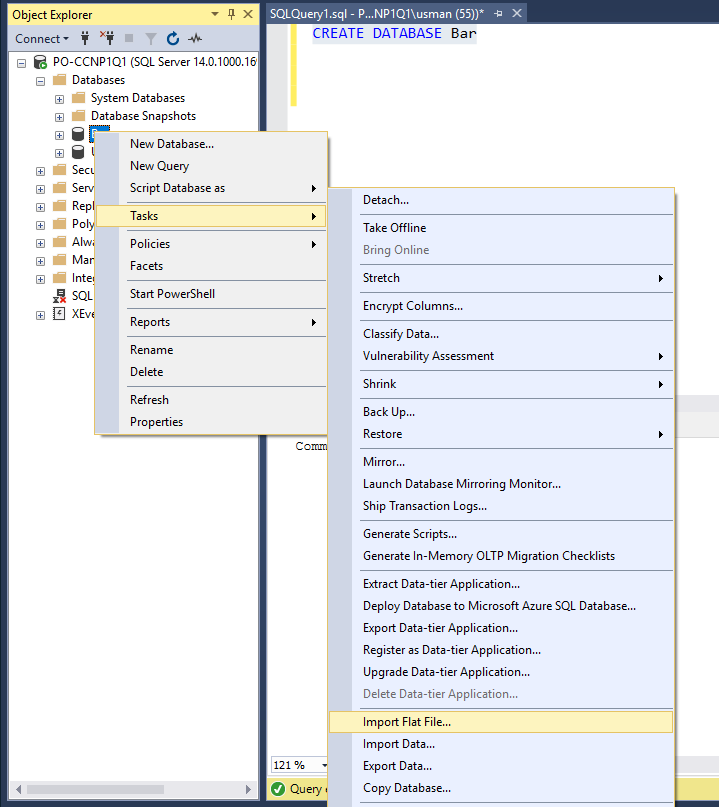
Once you select the “Import Flat File …” option, you will see a new window containing brief information about how to import the flat files, as shown below:

Simply click the “Next” button.

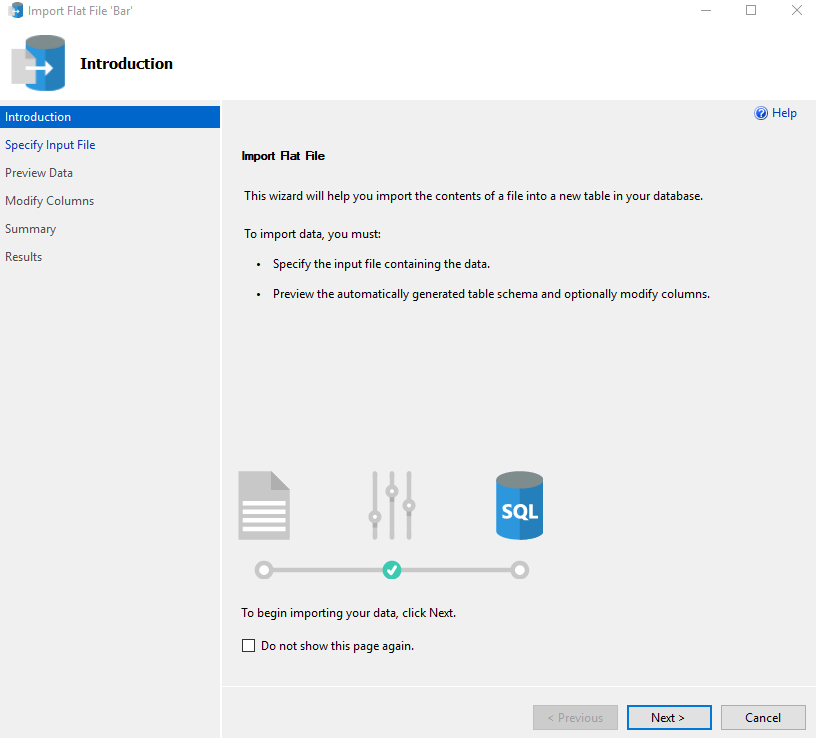
**Step 2: Select the file to import**

When you click the “Next” button, a window will appear, prompting you to select the file to import and to specify the corresponding table name. Select the file and enter the name that you want to give to the table that will contain data from your CSV file. I named the table as “wine” (which is the default name) as shown in the figure below:

Click the “Next” button again.

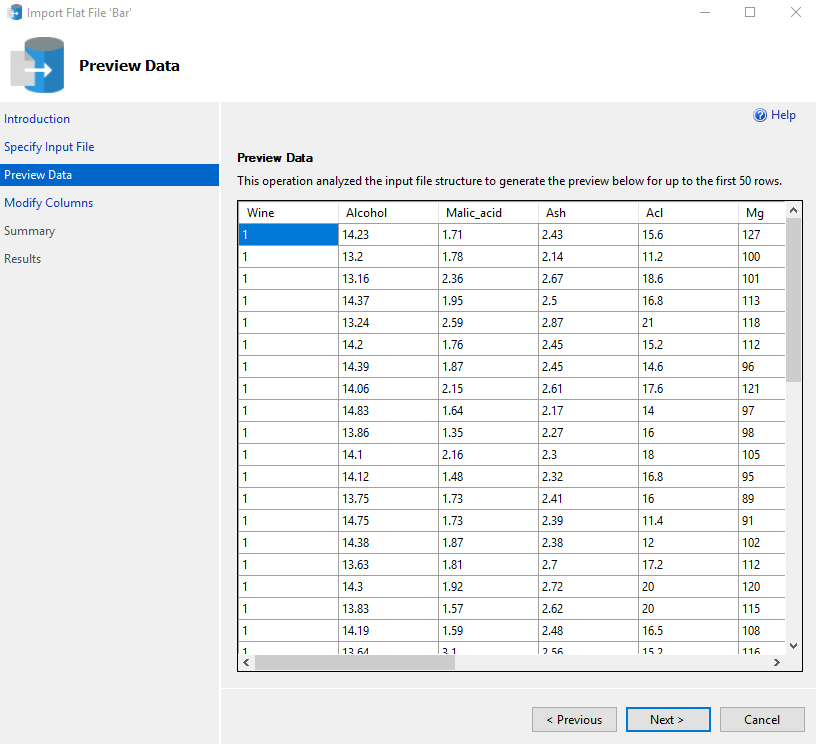


Once you select the “Import Flat File …” option, you will see a new window containing brief information about how to import the flat files, as shown below:



**Step 3: Preview the data**

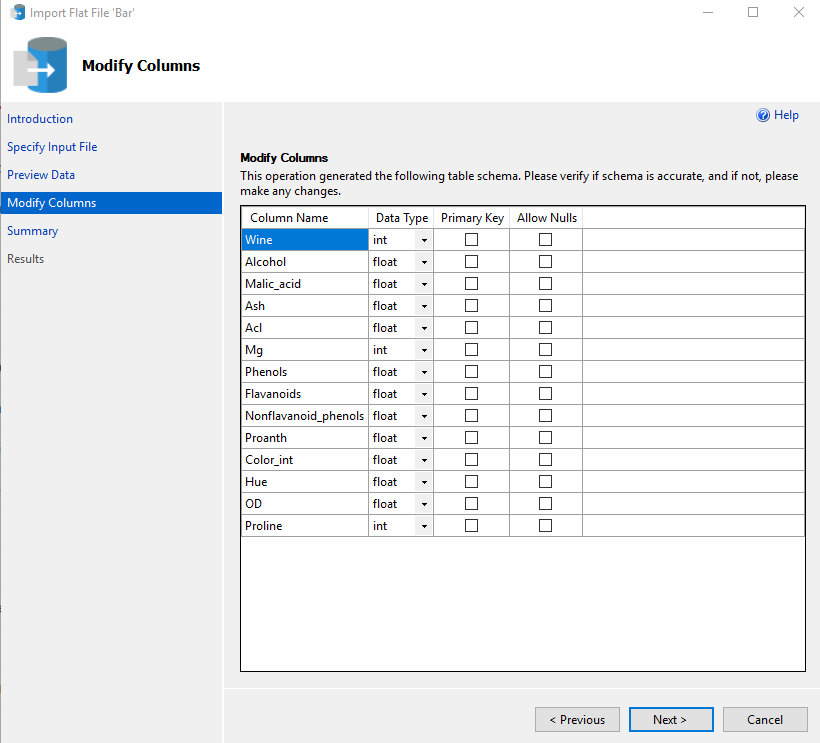
If your file has loaded successfully, a new window will appear containing a preview of your file, as shown in the following figure:



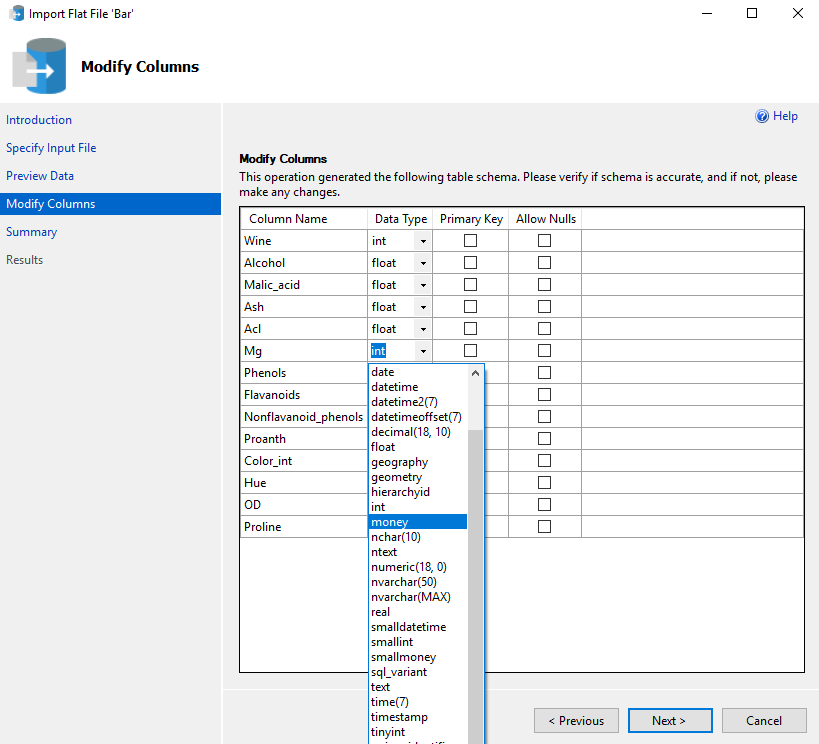
You can see that the preview contains data from our CSV file. Click the “Next” button.

**Step 4: Modify columns**

Once you click the “Next” button on the “Preview Data” screen, you will see the “Modify Columns” screen where you can modify the details of the columns of your table, as shown in the following screenshot:



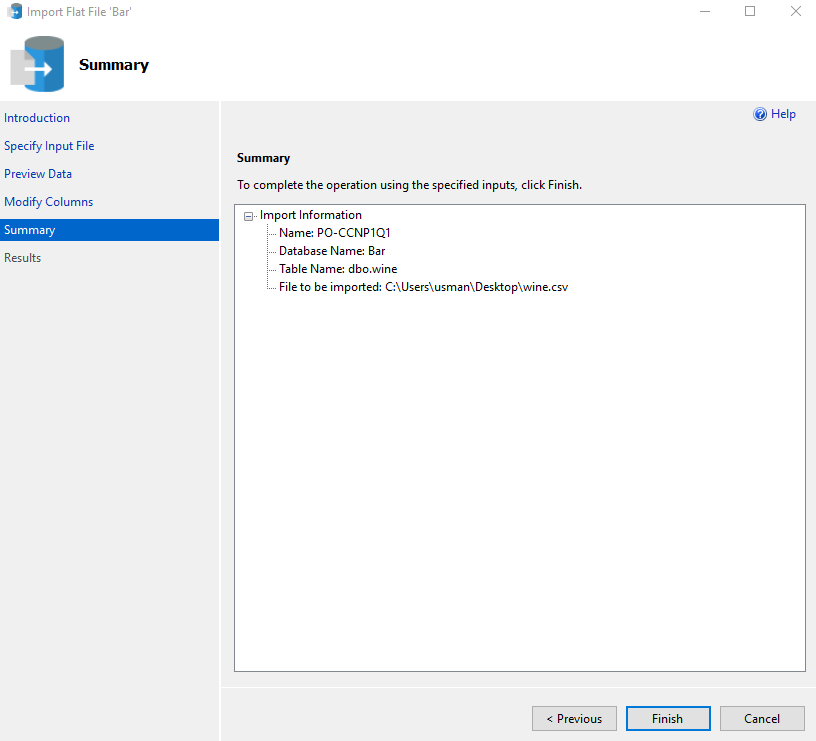
In the above figure, you can see that each column in the CSV file has been assigned a default data type. The default types should work in the most cases. However, if you want, you can change the data type by simply clicking the drop-down list in front of each column as shown below:



Furthermore, you can also set a column as the Primary key and allow Null values for any column. Once you are done with modifying the columns, click the “Next” button.

**Step 5: Viewing the summary**

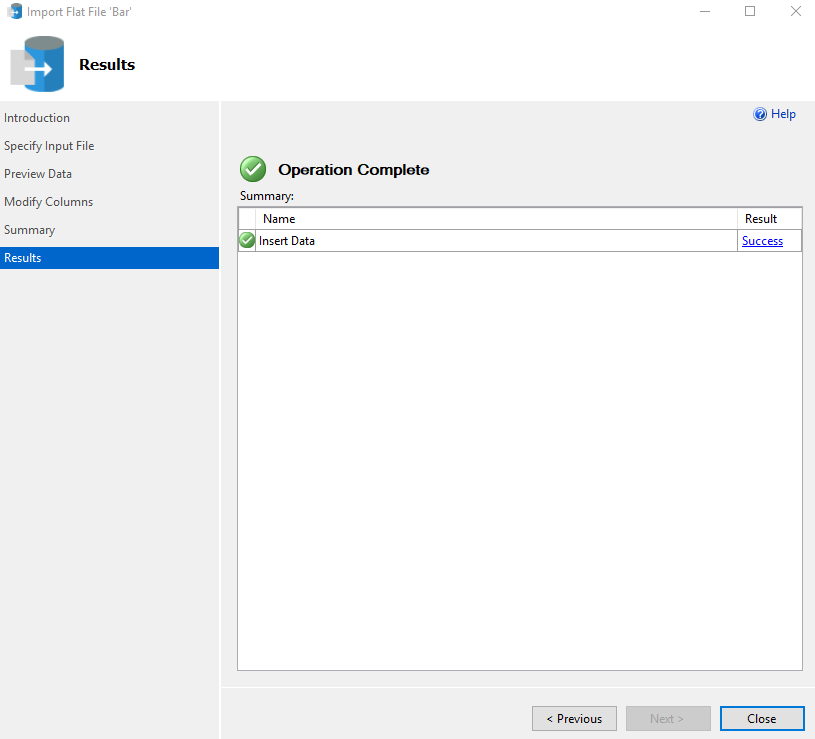
A new window displaying the summary of the operation including the name of the Server, Database, Table and the file that is being imported will be displayed. This is shown in the following screenshot:



Click the “Finish” button.

**Step 6: Process completion**

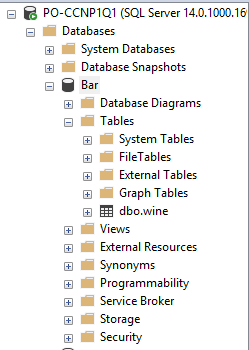
Depending upon the size of the file, the SQL Server will take some time to import the file. Once the file is successfully imported, without any errors, you will see the following screen:



**Working with a CSV file**

We have successfully imported our CSV file to our SQL Server data table. We can now execute SQL queries on this table.

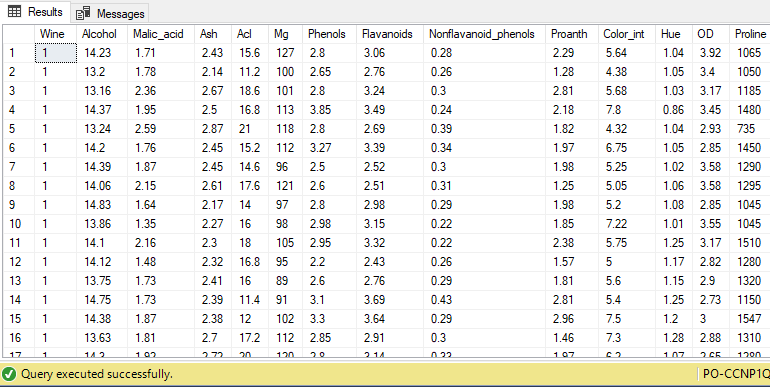
Since we imported the file into our “Bar” database in the “wine” table, the “Bar” database should contain a table named “wine”. Go to Object Explorer-> Databases -> Bar-> Tables, here you should see your data table “dbo.wine”, as shown below:



Execute the following query to SELECT all the records from the newly created “wine” table.

|  |  |
| --- | --- |
| 1  2 | USE Bar  SELECT \* FROM wine |

The output looks like this:



# LEAD, LAG

SELECT YEAR(OrderDate) as 'Year',

SUM(TotalDue) as 'Total Sales',

LAG(SUM(TotalDue)) OVER (ORDER BY YEAR(OrderDate)) As 'Prev Year'

FROM Sales.SalesOrderHeader

GROUP BY YEAR(OrderDate)

ORDER BY Year

Output:

Year Total Sales Prev Year

2005 12693250.6264 NULL

2006 34463848.4353 12693250.6264

2007 47171489.546 34463848.4353

2008 28888197.5082 47171489.546

# Restrict the number of rows returned

SELECT TOP 3 PersonType, COUNT(\*) as 'Person Count'

FROM Person.Person

GROUP BY PersonType

Output:

PersonType Person Count

IN 18484

EM 273

SP 17