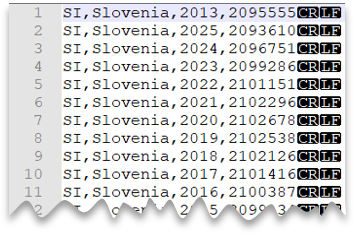
**Prerequisites**

Your first step is to **create a database** where the tables will be created. Then initialize the objects by executing [setup script](https://github.com/Azure-Samples/Synapse/blob/master/SQL/Samples/LdwSample/SampleDB.sql) on that database. This setup script will create the data sources, database scoped credentials, and external file formats that are used in these samples.

**Windows style new line**

The following query shows how to read a CSV file without a header row, with a Windows-style new line, and comma-delimited columns.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '\n'

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

country\_name = 'Luxembourg'

AND year = 2017;

**Unix-style new line**

The following query shows how to read a file without a header row, with a Unix-style new line, and comma-delimited columns. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '0x0a'

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

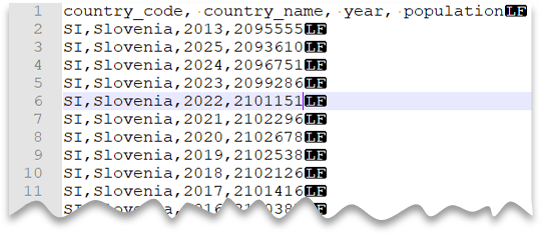
country\_name = 'Luxembourg'

AND year = 2017;

**Header row**

The following query shows how to a read file with a header row, with a Unix-style new line, and comma-delimited columns. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix-hdr/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

HEADER\_ROW = TRUE

) AS [r]

Option HEADER\_ROW = TRUE will result in reading column names from the header row in file. It is great for exploration purposes when you are not familiar with file content. For best performance see [Use appropriate data types section in Best practices](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/best-practices-serverless-sql-pool#use-appropriate-data-types). Also, you can read more about [OPENROWSET syntax here](https://learn.microsoft.com/en-us/azure/synapse-analytics/sql/develop-openrowset#syntax).

**Custom quote character**

The following query shows how to read a file with a header row, with a Unix-style new line, comma-delimited columns, and quoted values. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix-hdr-quoted/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '0x0a',

FIRSTROW = 2,

FIELDQUOTE = '"'

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

country\_name = 'Luxembourg'

AND year = 2017;

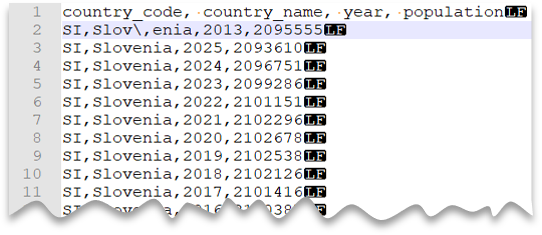
**Note**

This query would return the same results if you omitted the FIELDQUOTE parameter since the default value for FIELDQUOTE is a double-quote.

**Escape characters**

The following query shows how to read a file with a header row, with a Unix-style new line, comma-delimited columns, and an escape char used for the field delimiter (comma) within values. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix-hdr-escape/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '0x0a',

FIRSTROW = 2,

ESCAPECHAR = '\\'

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

country\_name = 'Slovenia';

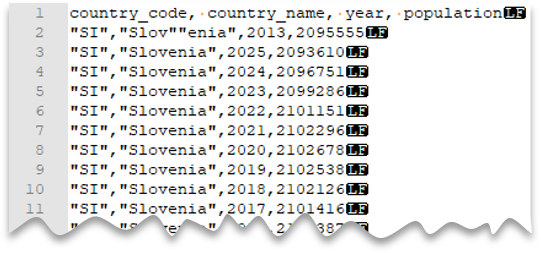
**Note**

This query would fail if ESCAPECHAR is not specified since the comma in "Slov,enia" would be treated as field delimiter instead of part of the country/region name. "Slov,enia" would be treated as two columns. Therefore, the particular row would have one column more than the other rows, and one column more than you defined in the WITH clause.

**Escape quoting characters**

The following query shows how to read a file with a header row, with a Unix-style new line, comma-delimited columns, and an escaped double quote char within values. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix-hdr-escape-quoted/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '0x0a',

FIRSTROW = 2

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

country\_name = 'Slovenia';

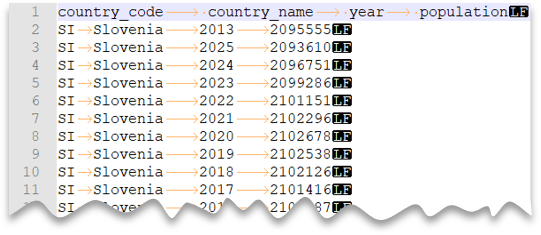
**Note**

The quoting character must be escaped with another quoting character. Quoting character can appear within column value only if value is encapsulated with quoting characters.

**Tab-delimited files**

The following query shows how to read a file with a header row, with a Unix-style new line, and tab-delimited columns. Note the different location of the file as compared to the other examples.

File preview:



SQLCopy

SELECT \*

FROM OPENROWSET(

BULK 'csv/population-unix-hdr-tsv/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR ='\t',

ROWTERMINATOR = '0x0a',

FIRSTROW = 2

)

WITH (

[country\_code] VARCHAR (5) COLLATE Latin1\_General\_BIN2,

[country\_name] VARCHAR (100) COLLATE Latin1\_General\_BIN2,

[year] smallint,

[population] bigint

) AS [r]

WHERE

country\_name = 'Luxembourg'

AND year = 2017

**Return a subset of columns**

So far, you've specified the CSV file schema using WITH and listing all columns. You can only specify columns you actually need in your query by using an ordinal number for each column needed. You'll also omit columns of no interest.

The following query returns the number of distinct country/region names in a file, specifying only the columns that are needed:

**Note**

Take a look at the WITH clause in the query below and note that there is "2" (without quotes) at the end of row where you define the *[country\_name]* column. It means that the *[country\_name]* column is the second column in the file. The query will ignore all columns in the file except the second one.

SQLCopy

SELECT

COUNT(DISTINCT country\_name) AS countries

FROM OPENROWSET(

BULK 'csv/population/population.csv',

DATA\_SOURCE = 'SqlOnDemandDemo',

FORMAT = 'CSV', PARSER\_VERSION = '2.0',

FIELDTERMINATOR =',',

ROWTERMINATOR = '\n'

)

WITH (

--[country\_code] VARCHAR (5),

[country\_name] VARCHAR (100) 2

--[year] smallint,

--[population] bigint

) AS [r]