Gaining Insights with U-SQL at Scale

# Overview

U-SQL is a new Query Language introduced in the Azure Data Lake Analytics service. It has been specifically developed for the requirements of Big Data processing by combining the declarative power of SQL with the expressiveness of C# for custom code integration. It allows you to focus on the business problem and takes over the efficient scaling out of the user’s code on large amounts of data.

In this challenge you will learn to use U-SQL to gain insights into your data in Azure Data Lake at scale. You will learn how U-SQL makes it easy to schematize data on read, scale out your custom code and show how you can apply domain-specific libraries to your data at scale.

# Objectives

* Submit a simple U-SQL script against sample data using the local account environment.
* Extend the simple U-SQL script with custom C# code to extract interesting information from textual data.
* Use the Visual Studio Azure Data Lake tooling to create a custom-library, deploy it to your Azure Data Lake account and use it in your script.

# Prerequisites

* Windows 10
* Visual Studio 2015
* Azure SDK 2.8 or newer
* Azure Data Lake Tool for Visual Studio (<http://aka.ms/adltoolsvs>) with the tweet dataset installed in the sample directory.
* A login account to log into the Azure Data Lake Analytics Account

If you’re using a system setup for the labs, they already have all the tools installed.

The username and password will be provided by the lab.

# Intended Audience

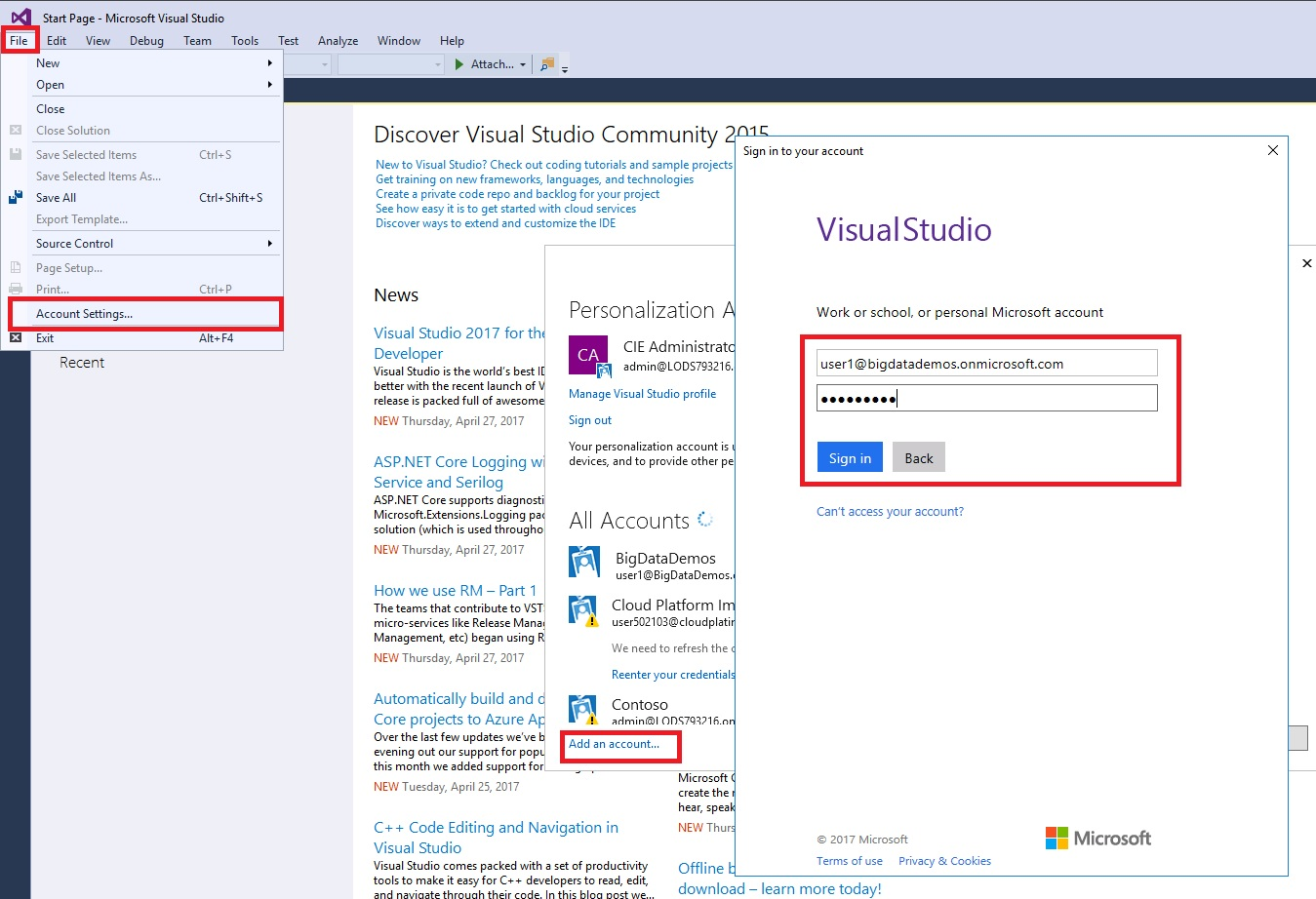
This Quick Start Challenge is intended for data developer’s and data analysts. Familiarity with SQL, C# and Visual Studio is recommended but not required.

# Step 1: Login and Create a new U-SQL Project in Visual Studio

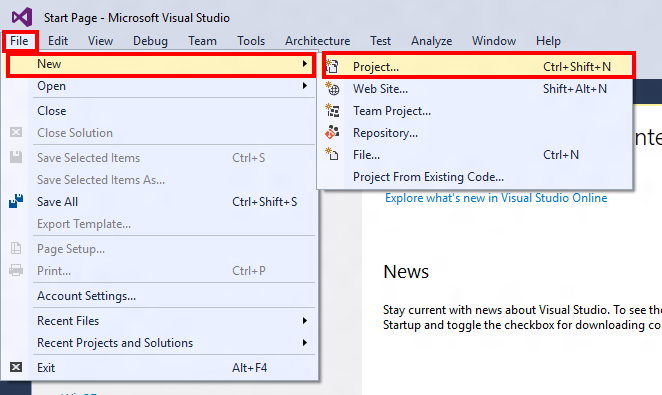
Let’s start by starting Visual Studio, login into your provided Azure account, and creating a new U-SQL project in Visual Studio.

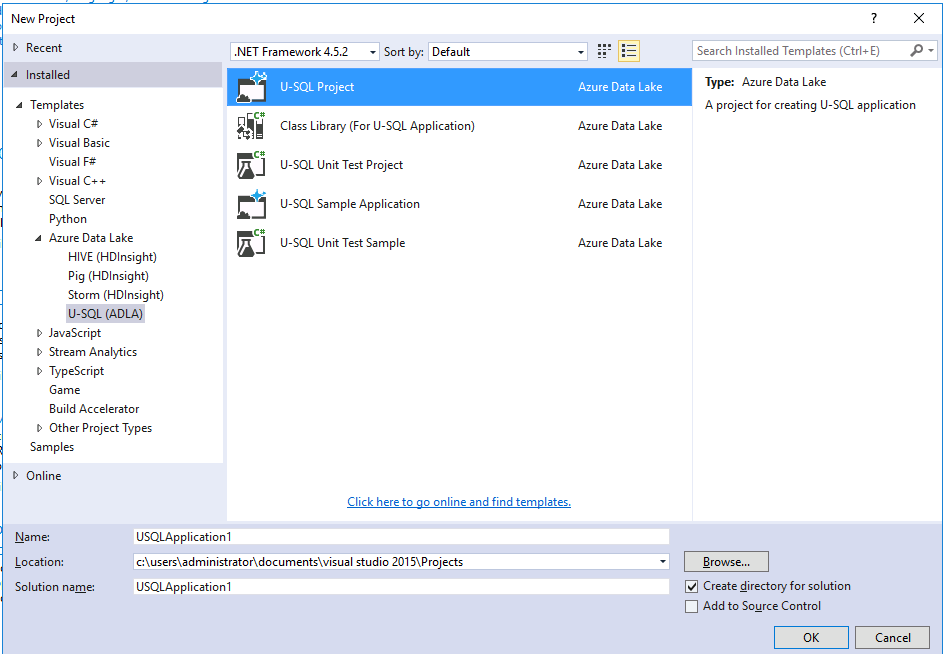
First, launch **Visual Studio 2015** from the Start Menu.

Then log into the provided account by selecting the menu item **File** **→ Account Settings…**, then select Add an account…, enter the provided username and password and sign in.



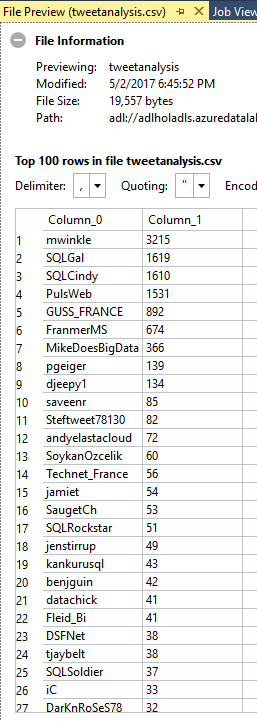
Then, create a new project:



Select **Installed → Templates → Azure Data Lake → U-SQL (ADLA) → U-SQL Project** for your project template: 

Give the project a name and create it.

# Challenge: Write a script in U-SQL to process all Tweet CSV files in the folder to aggregate all the tweets by their respective authors. The result should look like below:



The rest of the page is intentionally left blank to allow the user to think before scrolling to the solution.

# Solution:

# Step 2: Copy the following Script to the Script.usql Window

**Script** to copy:

@t =

EXTRACT date string,

time string,

author string,

tweet string

FROM "/Samples/Data/Tweets/{\*}.csv"

USING Extractors.Csv();

@res =

SELECT author,

COUNT( \* ) AS tweetcount

FROM @t

GROUP BY author;

OUTPUT @res

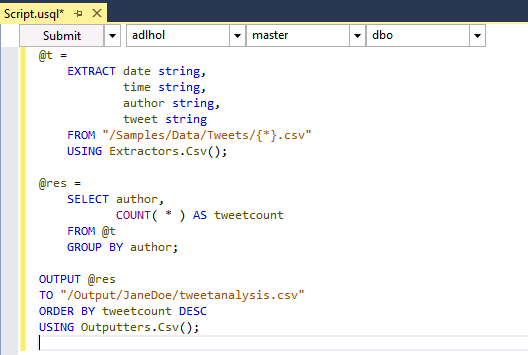
TO "/Output/<insert unique name>/tweetanalysis.csv"

ORDER BY tweetcount DESC

USING Outputters.Csv();

And replace the string <insert unique name> with a unique folder name (e.g., your FirstNameLastName).

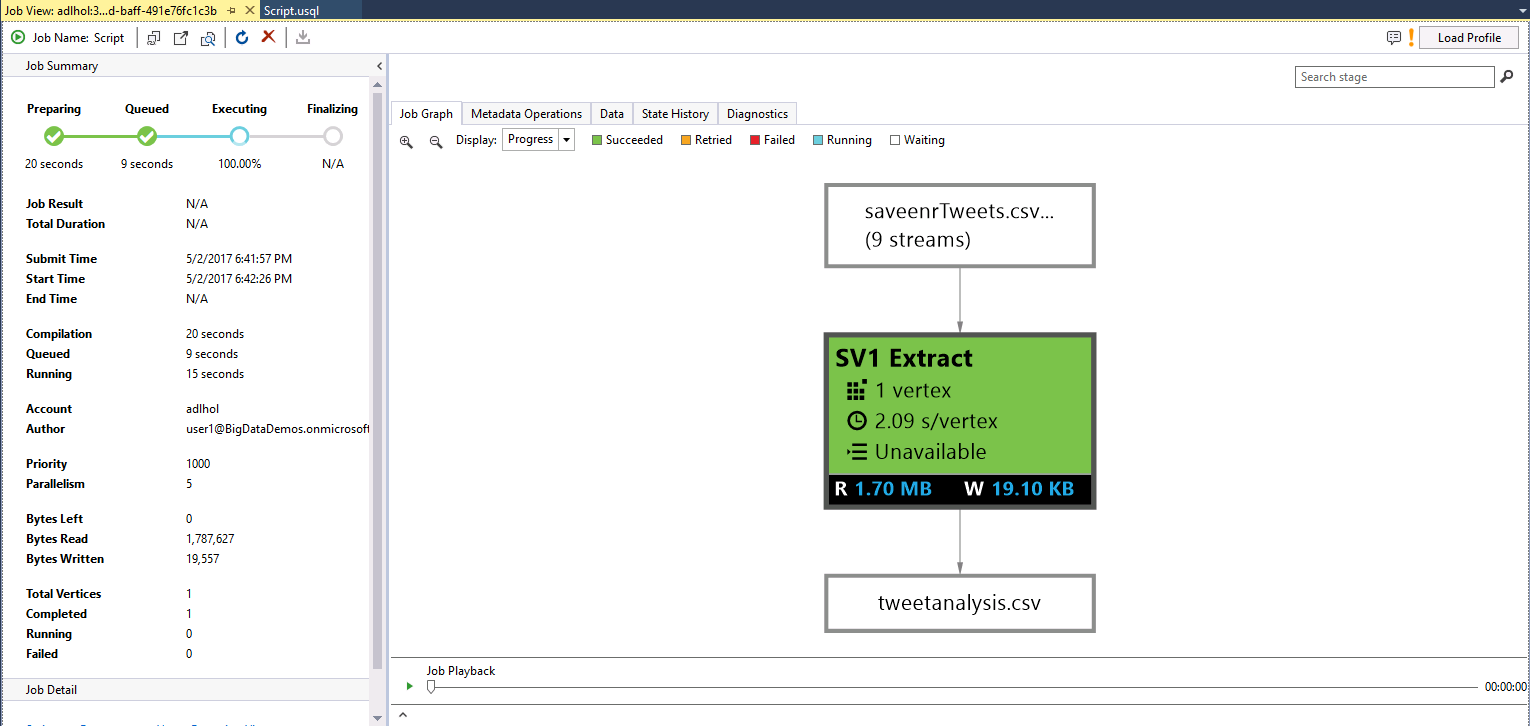
The window will look like:



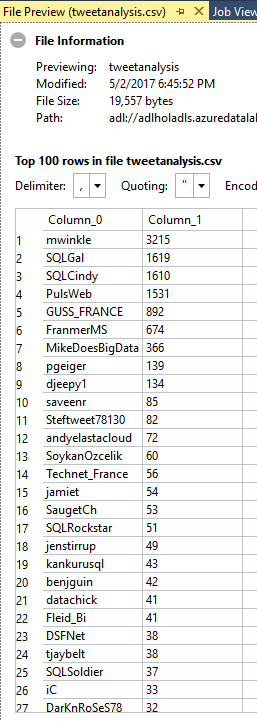
This script schematizes all files matching the file set wildcard expression {\*}.csv in the Samples/Data/Tweets folder, counts the tweets per author and outputs the result in a file called tweetanalysis.csv in the specified output folder.

Make sure you select the **adlhol** Account next to the Submit button (see image above).

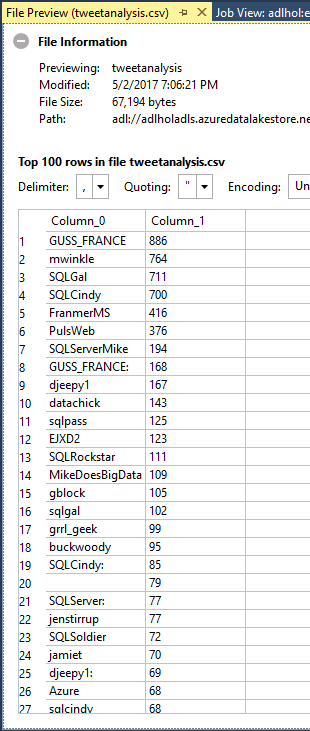
Then submit the U-SQL script by clicking on the Submit button in the Script.usql window. After a short while, the following job graph window should appear:

Once the job has completed running, double-click the tweetanalysis.csv output box in the job graph.

This will open the file preview and show the result of the U-SQL script:



# Challenge: Extract the mentions from the tweet files. The result should look like below:



The rest of the page is intentionally left blank to allow the user to think before scrolling down to the solution on the next page.

# Solution

# Step 3: Extract tweet mentions from the tweets with C#

Now we want to extract more information about who was referenced in the tweets and count the references.

First, replace the content of your existing script file with the following statements:

@t =

EXTRACT date string,

time string,

author string,

tweet string

FROM "/Samples/Data/Tweets/{\*}.csv"

USING Extractors.Csv();

@m =

SELECT SqlArray.Create(tweet.Split(' ')

.Where(x=>x.StartsWith("@")).Select(x=>x.Substring(1)))

AS mentions

FROM @t;

@m =

SELECT mention

FROM @m

CROSS APPLY

EXPLODE(mentions) AS M(mention);

@res =

SELECT mention,

COUNT( \* ) AS tweetcount

FROM @m

GROUP BY mention;

OUTPUT @res

TO "/Output/<insert unique name>/tweetanalysis.csv"

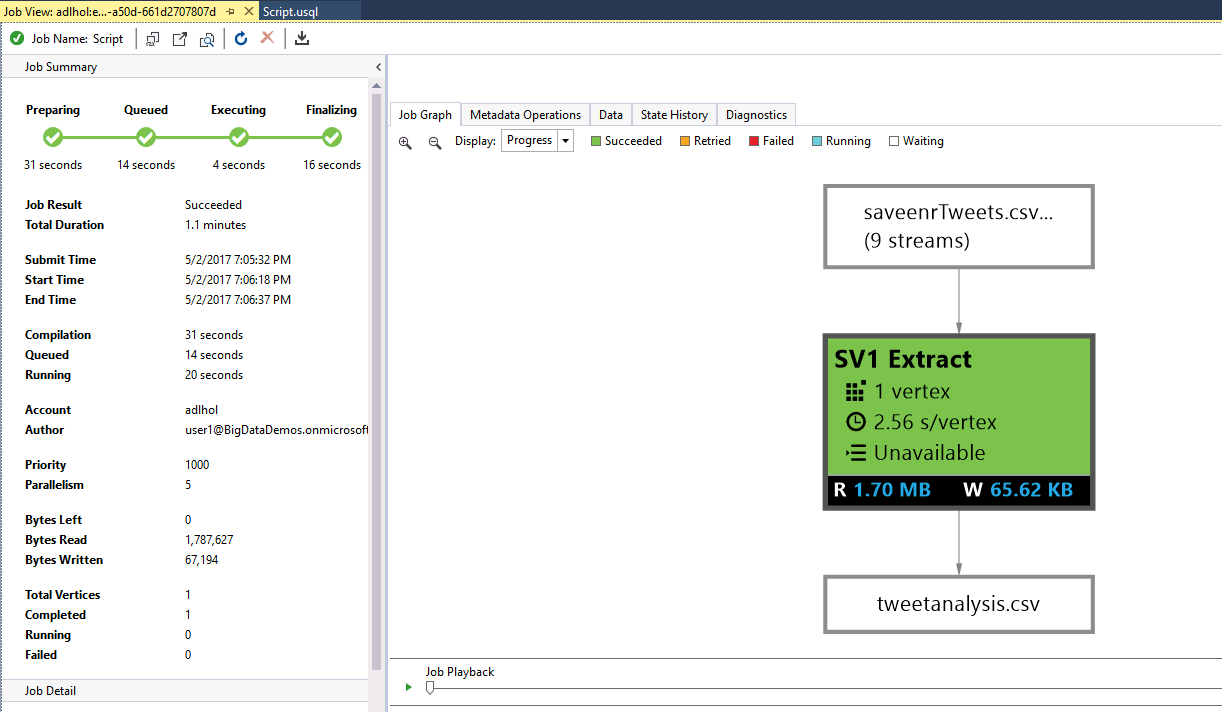
ORDER BY tweetcount DESC

USING Outputters.Csv();

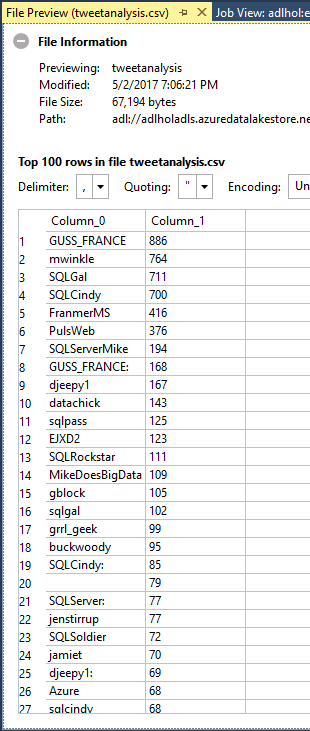
The resulting Script.usql window should look like (red highlight shows difference to previous script):



Then hit the Submit button of the Script.usql window and wait for the job graph window to show successful completion of the job:



Finally open the tweetanalysis.csv file as in Step 2 and see the following result:



# Challenge: Refactor the C# Code into a user-defined Function in a C# Project for U-SQL

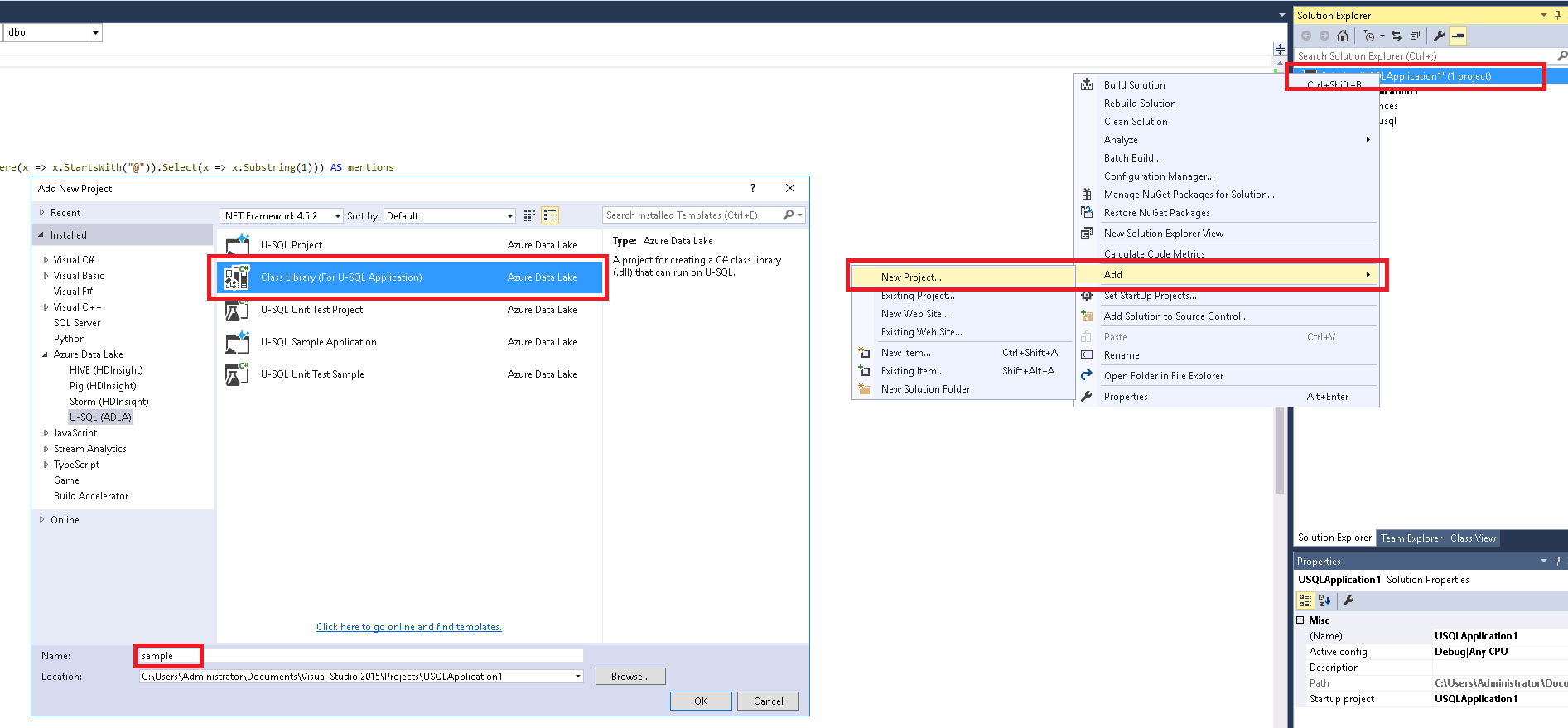
The rest of the page is intentionally left blank to allow the user to think before scrolling down to the solution on the next page.

Solution:

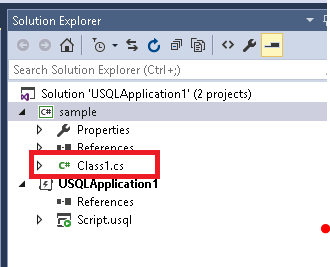
Step 4: Refactor the C# Code into a user-defined Function in a C# Project for U-SQL

Now, let’s refactor the C# code that does the extraction of the mentions from a tweet into a user-defined function into a C# Project for U-SQL.

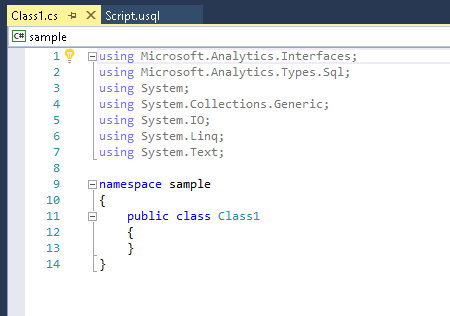
First create a Class Library for U-SQL Application for U-SQL in the existing solution and name it sample by right-clicking on the USQLApplication solution name in the Solution Explorer, select Add… and then New Project:



This also opens the Class1.cs file (shown below in the Solution Explorer):



The file contains some predefined namespace references and an empty namespace declaration for the sample namespace:



Now let’s create a public class tweetanalysis in the sample namespace that exposes the public static method get\_mentions() that implements the user-defined function. Copy the following code into the sample namespace:

public static class tweetanalysis

{

public static SqlArray<string> get\_mentions(string tweet) {

return SqlArray.Create(tweet.Split(' ')

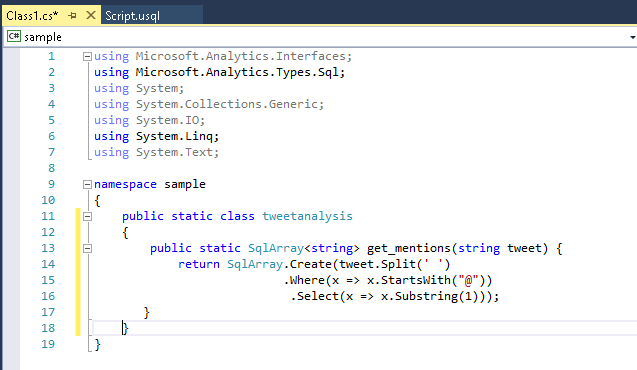
.Where(x=>x.StartsWith("@"))

.Select(x=>x.Substring(1)));

}

}

The resulting file will look like:



# Challenge: Deploy the C# Assembly into a new Database in your Azure Data Lake account

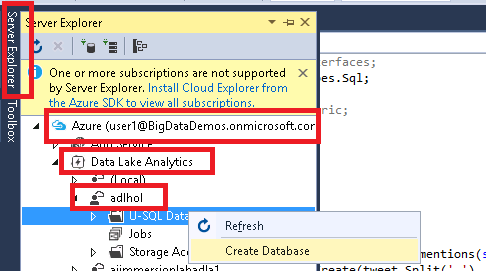
The rest of the page is intentionally left blank to allow the user to think before scrolling down to the solution on the next page.

Solution:

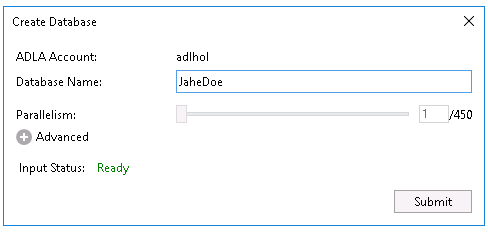
# Step 5: Deploy the C# Assembly into a new Database in your Azure Data Lake account

Now, let’s deploy the C# assembly that does the extraction of the mentions from a tweet into a new database that you create for yourself.

First create a new U-SQL database and name it with a unique database name (for example your FirstnameLastname). You can create a new database by opening the Server Explorer (on the left), then open **Azure → Data Lake Analytics → adlhol**, and then right click on **U-SQL Databases** and then select **Create Database**.

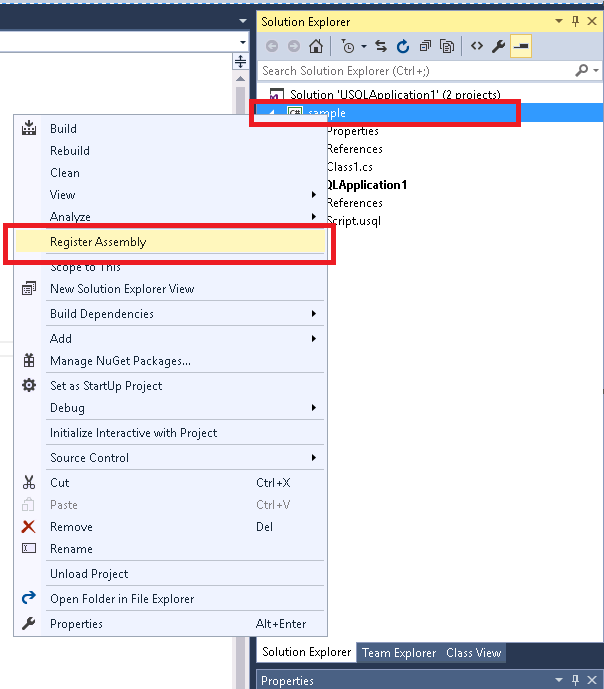


Then you enter the chosen **unique** database name as follows and submit:

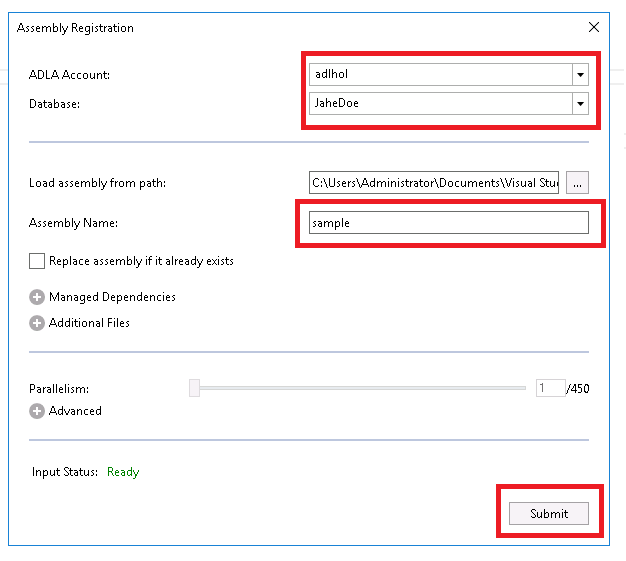


This opens another job window that shows the progress of the database creation script.

Once the database got created, right-click on the **sample** project in the Solution Explorer and select Register Assembly:

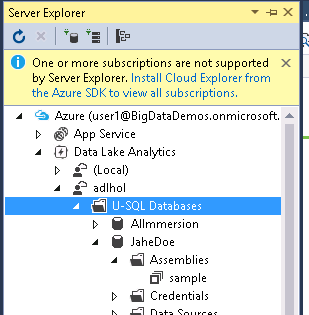


Then select the adlhol Account and your previously created database and submit it for registration:



This will build the project, upload it to ADLA and register it in the U-SQL database as a U-SQL assembly.

Once the registration job has completed successfully, you have registered the assembly. You can check the registration in the Solution Explorer (you may have to refresh the U-SQL Database list by right-clicking and selecting Refresh):



# Challenge: Refactor the U-SQL Script to make use of your custom-code provided in the assembly and run it

The result should look the same as in Step 3.

The rest of the page is intentionally left blank to allow the user to think before scrolling down to the solution on the next page.

Solution:

Step 6: Refactor the U-SQL Script to make use of your custom-code provided in the assembly and run it

Now, let’s update the U-SQL script to make use of the custom U-SQL assembly and its function and let U-SQL scale the function out over all the rows.

Let’s go back to the Script.usql tab in Visual Studio and add a reference to the assembly (replace the <insert unique database name> with the name of the earlier created database) and replace the C# expression with calling the get\_mentions method (remember that the output directory also needs to have a unique name):

REFERENCE ASSEMBLY <insert unique database name>.sample;

@t =

EXTRACT date string,

time string,

author string,

tweet string

FROM "/Samples/Data/Tweets/{\*}.csv"

USING Extractors.Csv();

@m =

SELECT sample.tweetanalysis.get\_mentions(tweet) AS mentions

FROM @t;

@m =

SELECT mention

FROM @m

CROSS APPLY

EXPLODE(mentions) AS M(mention);

@res =

SELECT mention,

COUNT( \* ) AS tweetcount

FROM @m

GROUP BY mention;

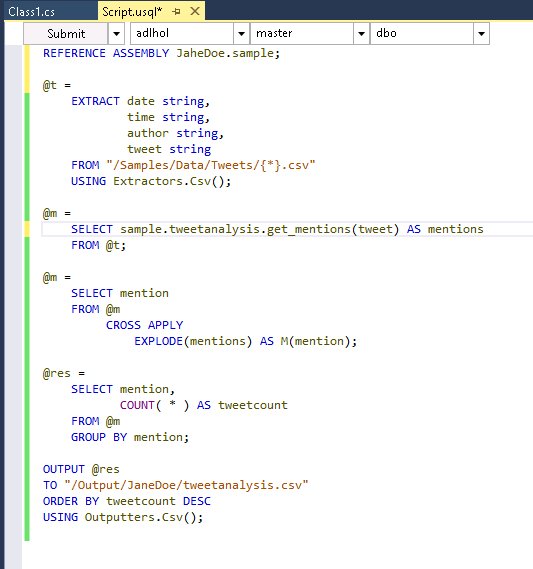
OUTPUT @res

TO "/Output/<insert unique name>/tweetanalysis.csv"

ORDER BY tweetcount DESC

USING Outputters.Csv();

So the script will look like:



Finally hit the submit button, wait for the job graph to show that the script successfully executed and then open the result file as in the previous steps. The result should be the same as in Step 3.

# Summary

Congratulations on completing this Quick Start Challenge! In this lab, you’ve learned how you can use U-SQL to process textual data and use the power of its C# integration to easily extract additional information from texts, register custom assemblies for reuse and then use U-SQL to scale it out.

# Additional Resources

If you are interested in learning more about U-SQL, check out the Azure Data Lake Analytics Service at <http://www.azure.com/datalake> and look at any of the following locations for more information:

**GitHub project and U-SQL community site**: <http://usql.io>

**U-SQL Reference Documentation**: <http://aka.ms/usql_reference>