Assignment 3 – Report

Group - 13

Ahmed Abdelmaksoud

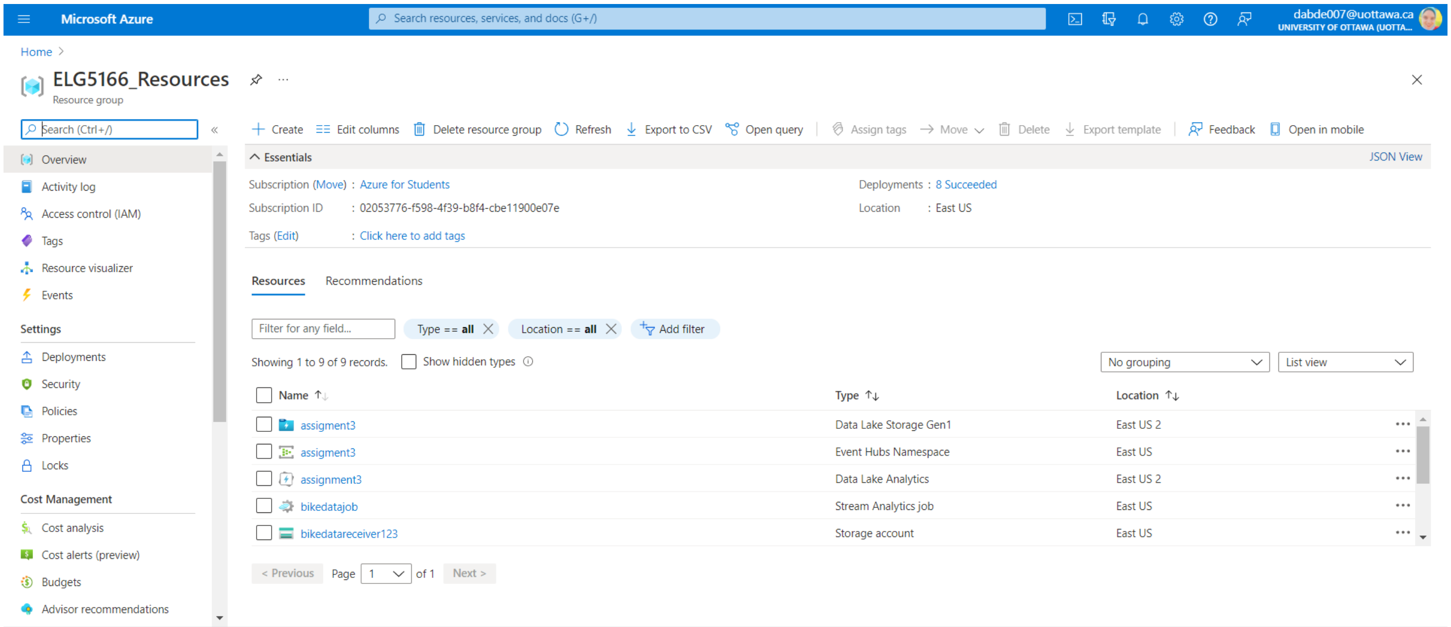
Hassan Ahmed

Dina Abdelhady

Ahmed Abdelsamad

1. Event Hubs Analytics

1. Create a resource group



ELG5166_Resources

Resource group

Essentials

Subscription (Move) : Azure for Students

Subscription ID : 02053776-f598-4f39-b8f4-cbe11900e07e

Tags (Edit) : Click here to add tags

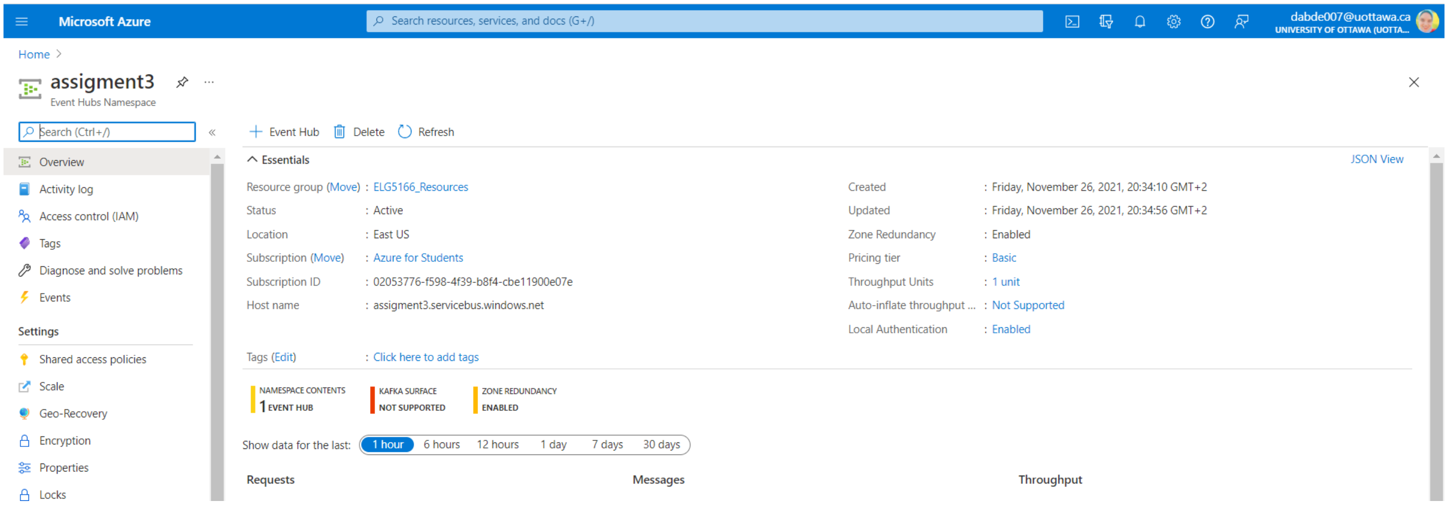
Deployments : 8 Succeeded

Location : East US

Showing 1 to 9 of 9 records. □ Show hidden types

□ Name ↑↓	Type ↑↓	Location ↑↓
□ assigment3	Data Lake Storage Gen1	East US 2
□ assigment3	Event Hubs Namespace	East US
□ assignment3	Data Lake Analytics	East US 2
□ bikedatajob	Stream Analytics job	East US
□ bikedatareceiver123	Storage account	East US

2. Create event hub name space



assigment3

Event Hubs Namespace

Essentials

Resource group (Move) : ELG5166_Resources

Status : Active

Location : East US

Subscription (Move) : Azure for Students

Subscription ID : 02053776-f598-4f39-b8f4-cbe11900e07e

Host name : assigment3.servicebus.windows.net

Created : Friday, November 26, 2021, 20:34:10 GMT+2

Updated : Friday, November 26, 2021, 20:34:56 GMT+2

Zone Redundancy : Enabled

Pricing tier : Basic

Throughput Units : 1 unit

Auto-inflate throughput ... : Not Supported

Local Authentication : Enabled

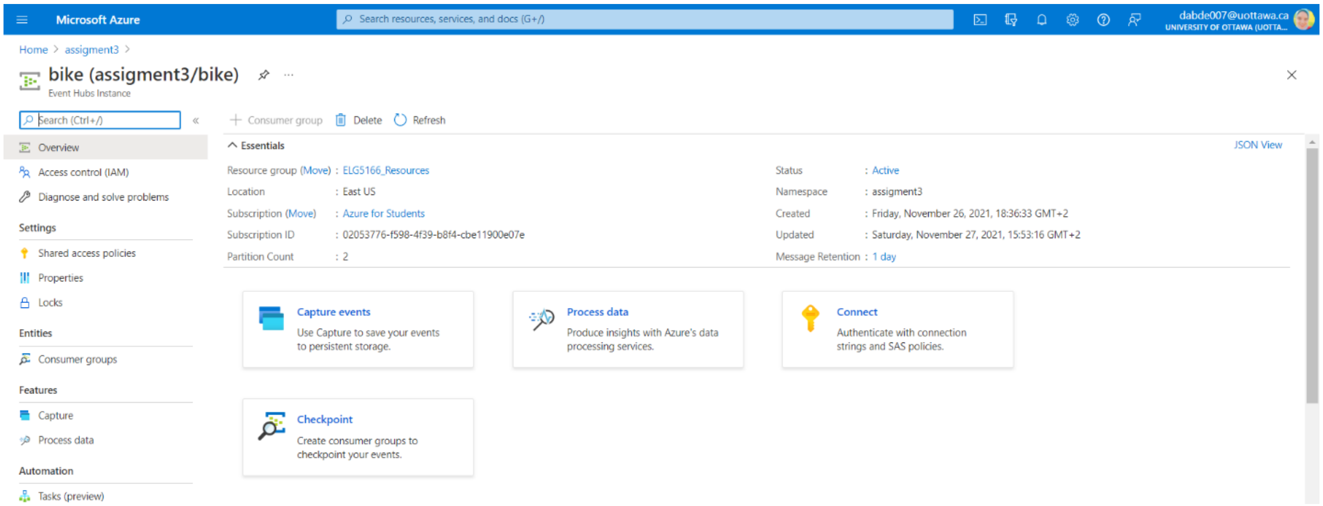
Tags (Edit) : Click here to add tags

NAMESPACE CONTENTS 1 EVENT HUB

KAFKA SURFACE NOT SUPPORTED

ZONE REDUNDANCY ENABLED

3. Create event hub



Resource group (Move)	: ELG5166_Resources	Status	: Active
Location	: East US	Namespace	: assigment3
Subscription (Move)	: Azure for Students	Created	: Friday, November 26, 2021, 18:36:33 GMT+2
Subscription ID	: 02053776-f598-4f39-b8f4-cbe11900e07e	Updated	: Saturday, November 27, 2021, 15:53:16 GMT+2
Partition Count	: 2	Message Retention	: 1 day

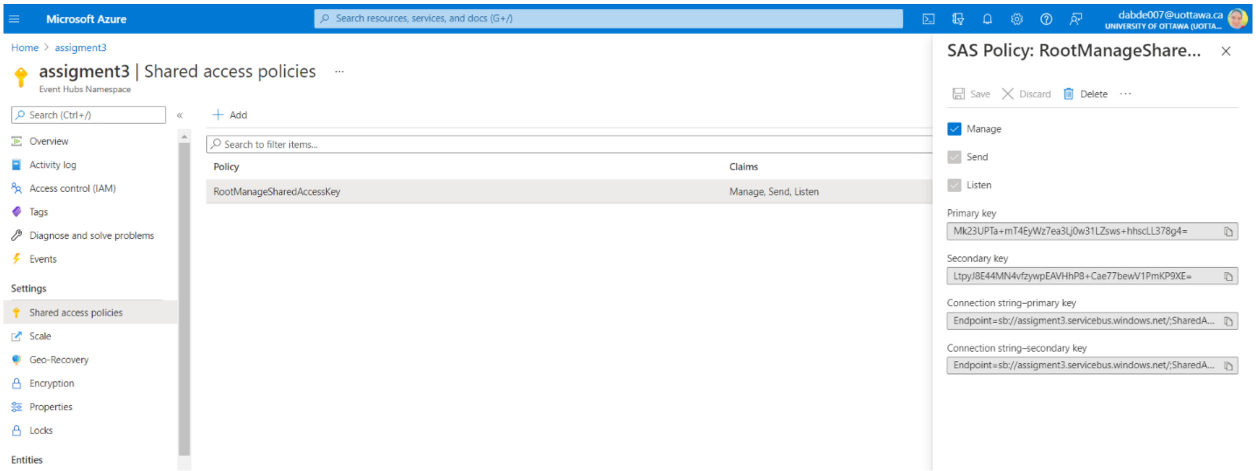
Capture events
Use Capture to save your events to persistent storage.

Process data
Produce insights with Azure's data processing services.

Connect
Authenticate with connection strings and SAS policies.

Checkpoint
Create consumer groups to checkpoint your events.

4. Copy connection string key.



Policy	Claims
RootManageSharedAccessKey	Manage, Send, Listen

SAS Policy: RootManageShare...

☑ Manage

☑ Send

☑ Listen

Primary key
Mk23UPTa+mT4EyWz7ea3Lj0w31LZsws+hhscLL378g4=

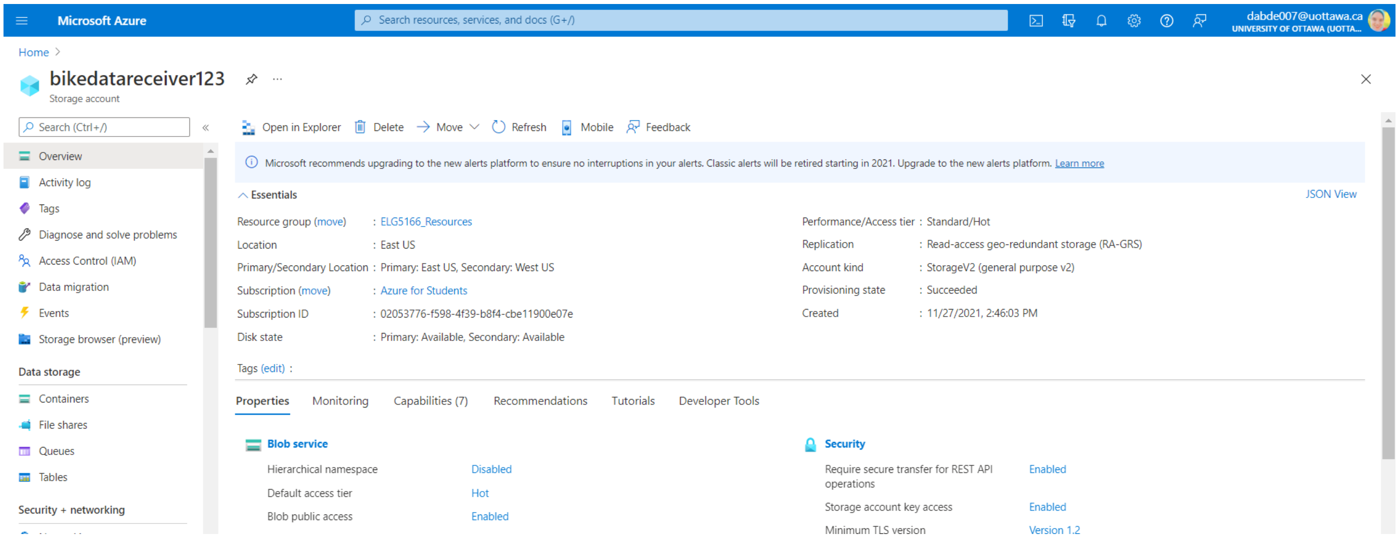
Secondary key
LtpyJ8E44MN4vfzywpEAVHhP8+Cae77bewV1PmKP9XE=

Connection string–primary key
Endpoint=sb://assigment3.servicebus.windows.net/;SharedA...

Connection string–secondary key
Endpoint=sb://assigment3.servicebus.windows.net/;SharedA...

5. Create storage account

After that we need to change the eventHubName, connectionString and cachedEventSource in the event hub sender to be able to read the data and connect to the Azure.



bikedatareceiver123
Storage account

Microsoft recommends upgrading to the new alerts platform to ensure no interruptions in your alerts. Classic alerts will be retired starting in 2021. Upgrade to the new alerts platform. Learn more

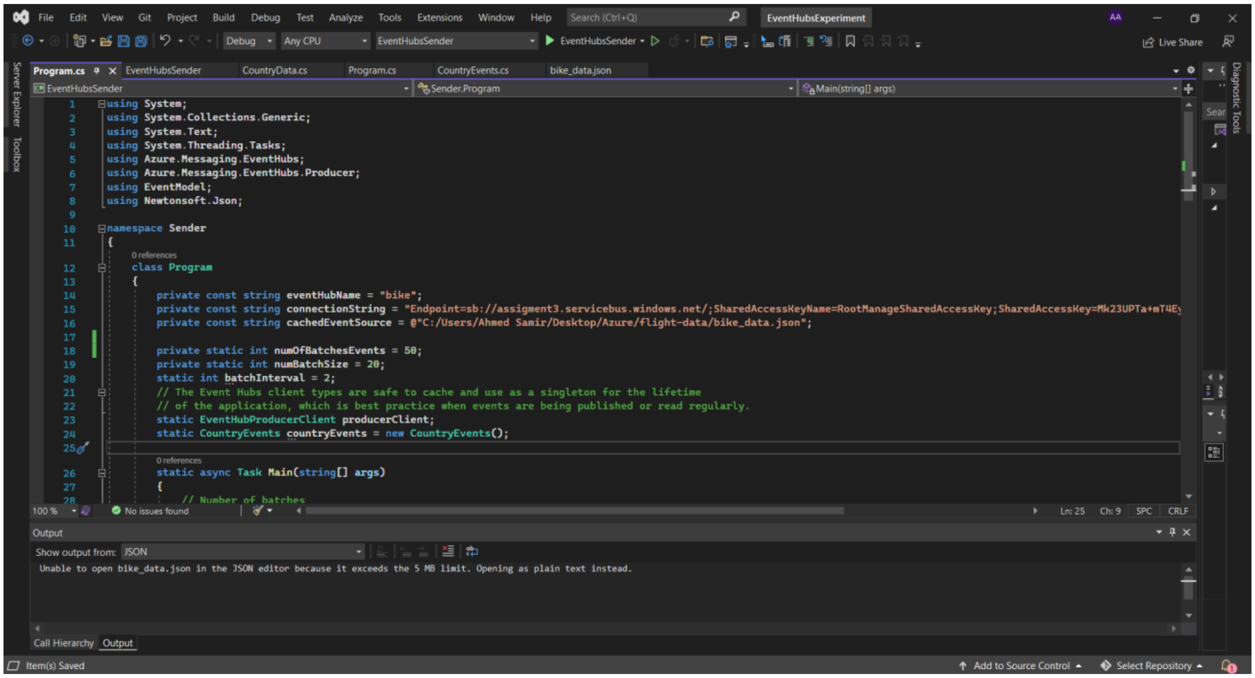
Essentials

Resource group (move) : ELG5166_Resources
Location : East US
Primary/Secondary Location : Primary: East US, Secondary: West US
Subscription (move) : Azure for Students
Subscription ID : 02053776-f598-4f39-b8f4-cbe11900e07e
Disk state : Primary: Available, Secondary: Available
Performance/Access tier : Standard/Hot
Replication : Read-access geo-redundant storage (RA-GRS)
Account kind : StorageV2 (general purpose v2)
Provisioning state : Succeeded
Created : 11/27/2021, 2:46:03 PM
Tags (edit) :

Blob service
Hierarchical namespace Disabled
Default access tier Hot
Blob public access Enabled

Security
Require secure transfer for REST API operations Enabled
Storage account key access Enabled
Minimum TLS version Version 1.2

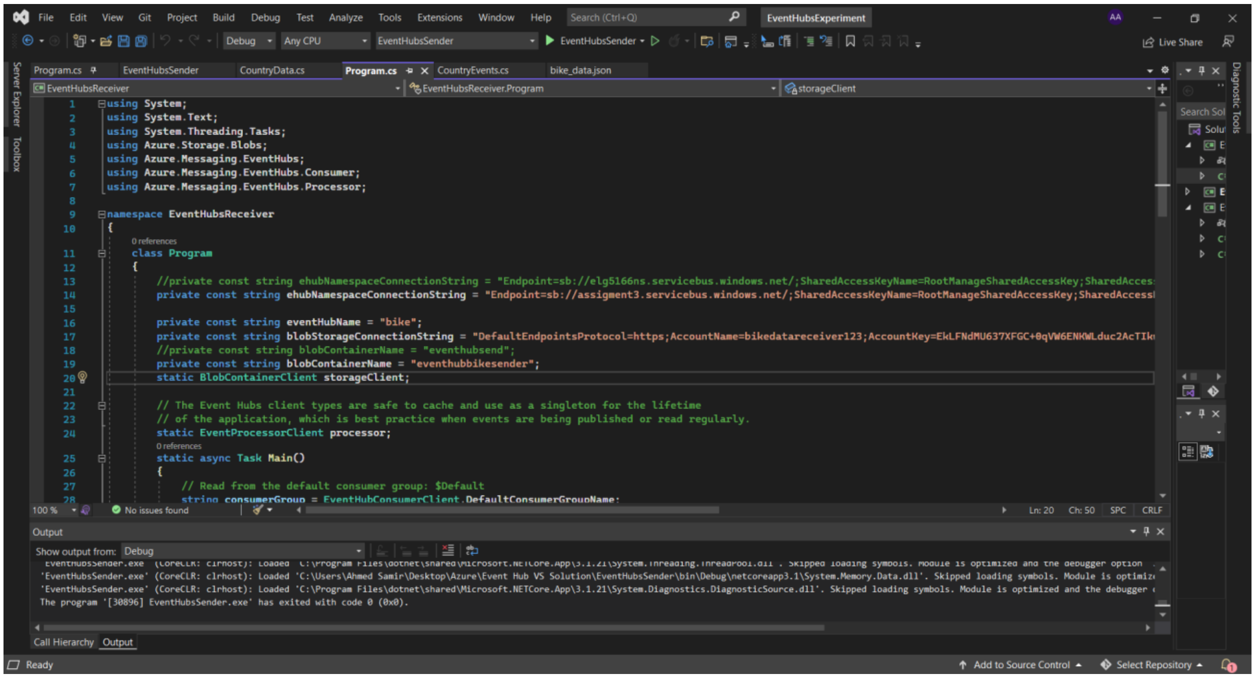
Also, we need to change the connections on the event hub receiver to be able to receive the data.



```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Producer;
using EventModel;
using Newtonsoft.Json;

namespace Sender
{
    class Program
    {
        private const string eventHubName = "bike";
        private const string connectionString = "Endpoint=sb://assigment3.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=Mk23UPTa+mT4Ey
        private const string cachedEventSource = @"C:/Users/Ahmed Samir/Desktop/Azure/flight-data/bike_data.json";

        private static int numOfBatchesEvents = 50;
        private static int numBatchSize = 20;
        static int batchInterval = 2;
        // The Event Hubs client types are safe to cache and use as a singleton for the lifetime
        // of the application, which is best practice when events are being published or read regularly.
        static EventHubProducerClient producerClient;
        static CountryEvents countryEvents = new CountryEvents();

        static async Task Main(string[] args)
        {
            // Number of batches
```

Show output from: JSON

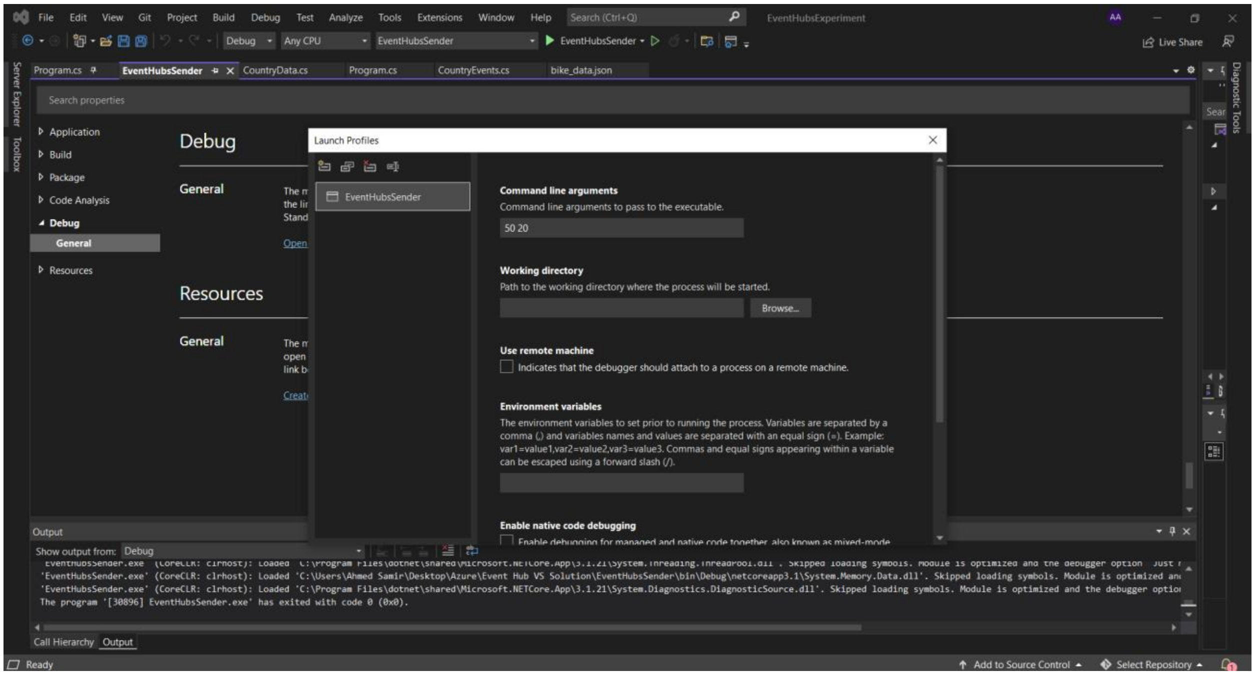
```
Unable to open bike_data.json in the JSON editor because it exceeds the 5 MB limit. Opening as plain text instead.
```



```
using System;
using System.Text;
using System.Threading.Tasks;
using Azure.Storage.Blobs;
using Azure.Messaging.EventHubs;
using Azure.Messaging.EventHubs.Consumer;
using Azure.Messaging.EventHubs.Processor;

namespace EventHubsReceiver
{
    class Program
    {
        //private const string ehubNamespaceConnectionString = "Endpoint=sb://elg5166ns.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAcces
        private const string ehubNamespaceConnectionString = "Endpoint=sb://assigment3.servicebus.windows.net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccess

        private const string eventHubName = "bike";
        private const string blobStorageConnectionString = "DefaultEndpointsProtocol=https;AccountName=bikedatareceiver123;AccountKey=EkLFNdMU637XFGC+0qVW6ENKWLduc2AcTIk
        //private const string blobContainerName = "eventhubsend";
        private const string blobContainerName = "eventhubbikesender";
        static BlobContainerClient storageClient;

        // The Event Hubs client types are safe to cache and use as a singleton for the lifetime
        // of the application, which is best practice when events are being published or read regularly.
        static EventProcessorClient processor;

        static async Task Main()
        {
            // Read from the default consumer group: $Default
            string consumerGroup = EventHubConsumerClient.DefaultConsumerGroupName;
```

```
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Threading.ThreadPool.dll'. Skipped loading symbols. Module is optimized and the debugger option
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Users\Ahmed Samir\Desktop\Azure\Event Hub VS Solution\EventHubsSender\bin\Debug\netcoreapp3.1\System.Memory.Data.dll'. Skipped loading symbols. Module is optimiz
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Diagnostics.DiagnosticSource.dll'. Skipped loading symbols. Module is optimized and the debugger
The program '[30896] EventHubsSender.exe' has exited with code 0 (0x0).
```

a) To change the trip entries to 20 and number of events to 50 we need to change the numBatchSize and numOfBatchesEvents through the EventHubSender Debug (Note the UI is different because we here use VS studio community 2022)



Launch Profiles

EventHubsSender

Command line arguments
Command line arguments to pass to the executable.
50 20

Working directory
Path to the working directory where the process will be started.

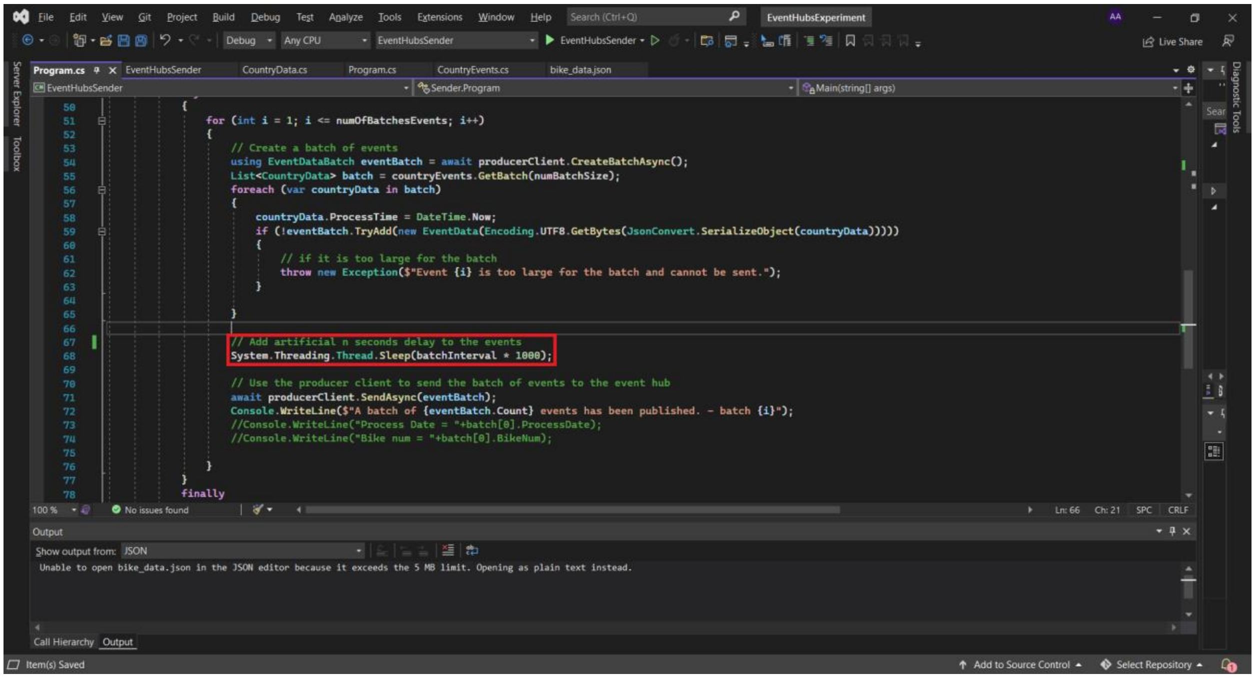
Use remote machine
□ Indicates that the debugger should attach to a process on a remote machine.

Environment variables
The environment variables to set prior to running the process. Variables are separated by a comma (,) and variables names and values are separated with an equal sign (=). Example: var1=value1,var2=value2,var3=value3. Commas and equal signs appearing within a variable can be escaped using a forward slash (/).

Enable native code debugging
□ Enable debugging for managed and native code together, also known as mixed-mode

```
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Threading.ThreadPool.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just M
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Users\Ahmed Samir\Desktop\Azure\Event Hub VS Solution\EventHubsSender\bin\Debug\netcoreapp3.1\System.Memory.Data.dll'. Skipped loading symbols. Module is optimized and
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Diagnostics.DiagnosticSource.dll'. Skipped loading symbols. Module is optimized and the debugger optio
The program '[30896] EventHubsSender.exe' has exited with code 0 (0x0).
```

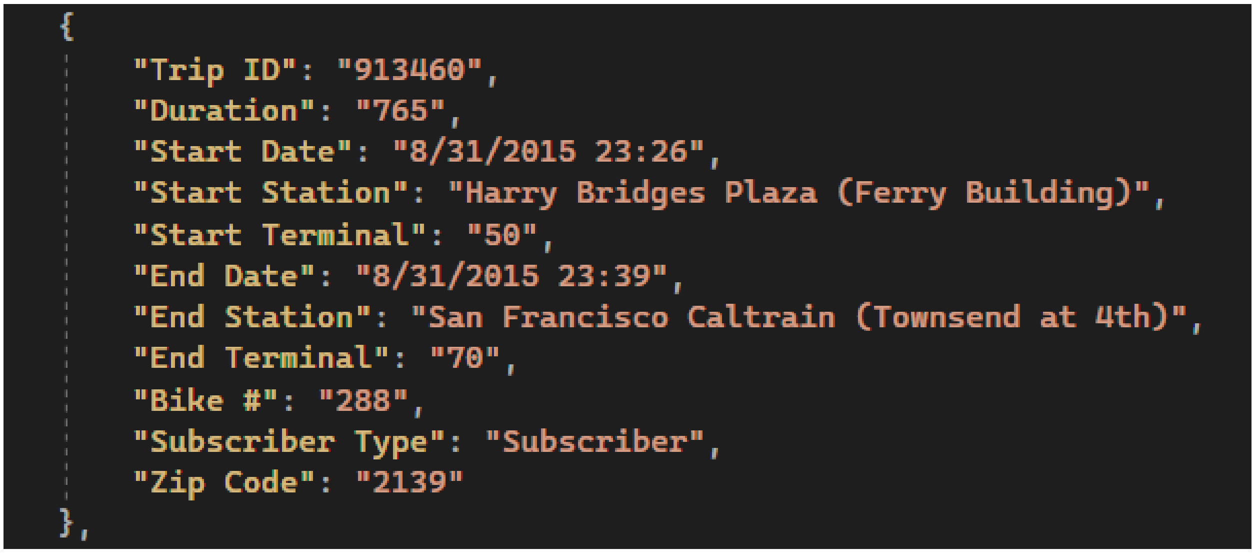
And to change the batch interval we initialized a variable and set its value to 2



```
            {
                for (int i = 1; i <= numOfBatchesEvents; i++)
                {
                    // Create a batch of events
                    using EventDataBatch eventBatch = await producerClient.CreateBatchAsync();
                    List<CountryData> batch = countryEvents.GetBatch(numBatchSize);
                    foreach (var countryData in batch)
                    {
                        countryData.ProcessTime = DateTime.Now;
                        if (!eventBatch.TryAdd(new EventData(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(countryData)))))
                        {
                            // if it is too large for the batch
                            throw new Exception($"Event {i} is too large for the batch and cannot be sent.");
                        }

                    }

                    // Add artificial n seconds delay to the events
                    System.Threading.Thread.Sleep(batchInterval * 1000);

                    // Use the producer client to send the batch of events to the event hub
                    await producerClient.SendAsync(eventBatch);
                    Console.WriteLine($"A batch of {eventBatch.Count} events has been published. - batch {i}");
                    //Console.WriteLine("Process Date = "+batch[0].ProcessDate);
                    //Console.WriteLine("Bike num = "+batch[0].BikeNum);

                }
            }
            finally
```

```
Unable to open bike_data.json in the JSON editor because it exceeds the 5 MB limit. Opening as plain text instead.
```

To send the data we need to look at the Json file

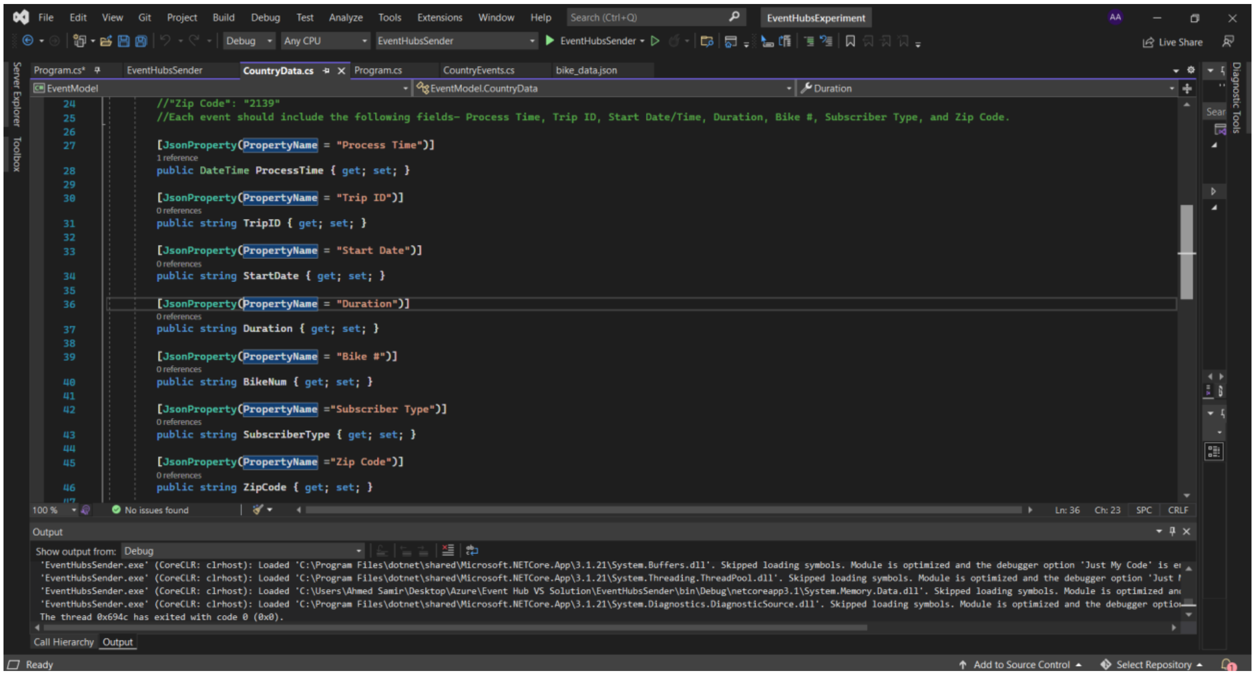


```
{
    "Trip ID": "913460",
    "Duration": "765",
    "Start Date": "8/31/2015 23:26",
    "Start Station": "Harry Bridges Plaza (Ferry Building)",
    "Start Terminal": "50",
    "End Date": "8/31/2015 23:39",
    "End Station": "San Francisco Caltrain (Townsend at 4th)",
    "End Terminal": "70",
    "Bike #": "288",
    "Subscriber Type": "Subscriber",
    "Zip Code": "2139"
},
```

And we can send all the data, but we preferred to send just the data that is required (Process Time, Trip ID, Start Date, Duration, Bike #, Subscriber Type, and Zip Code).

So we changed the class to be able to send these variables and we have added [JsonProperty(PropertyName = variableName)]

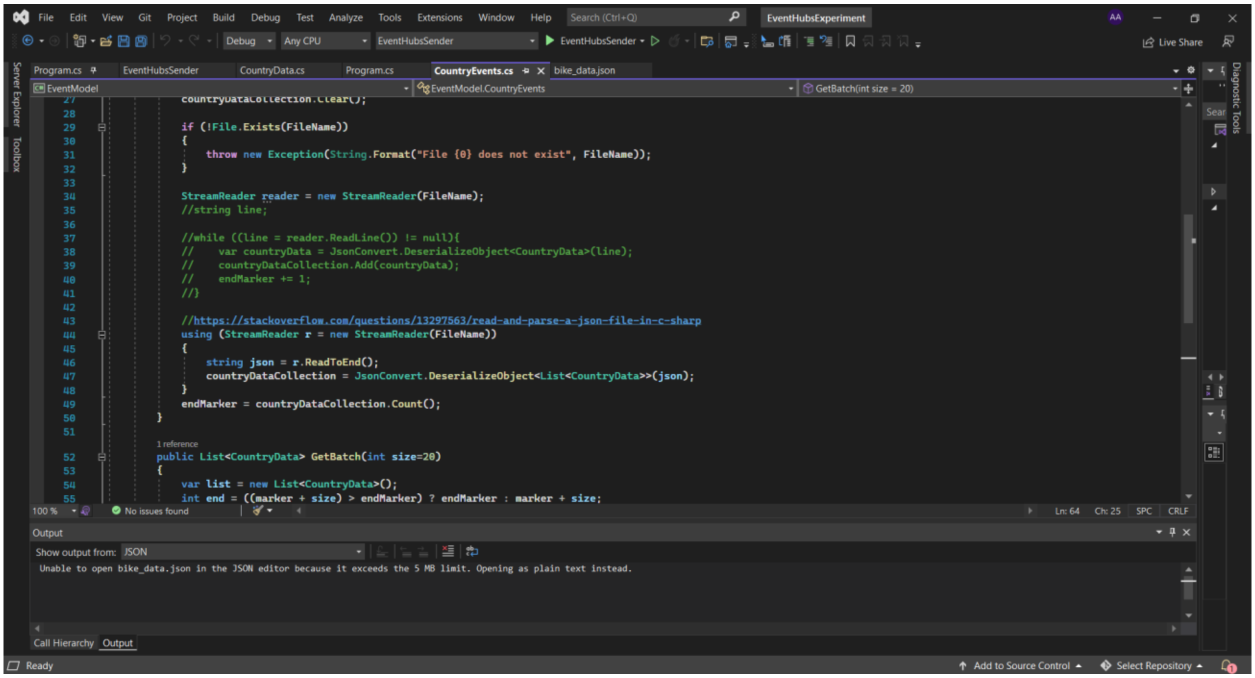
to be able to read and send the columns with spaces in between.



```
            //"Zip Code": "2139"
            //Each event should include the following fields- Process Time, Trip ID, Start Date/Time, Duration, Bike #, Subscriber Type, and Zip Code.

            [JsonProperty(PropertyName = "Process Time")]
            1 reference
            public DateTime ProcessTime { get; set; }

            [JsonProperty(PropertyName = "Trip ID")]
            0 references
            public string TripID { get; set; }

            [JsonProperty(PropertyName = "Start Date")]
            0 references
            public string StartDate { get; set; }

            [JsonProperty(PropertyName = "Duration")]
            0 references
            public string Duration { get; set; }

            [JsonProperty(PropertyName = "Bike #")]
            0 references
            public string BikeNum { get; set; }

            [JsonProperty(PropertyName ="Subscriber Type")]
            0 references
            public string SubscriberType { get; set; }

            [JsonProperty(PropertyName ="Zip Code")]
            0 references
            public string ZipCode { get; set; }
```

```
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Buffers.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is en
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Threading.ThreadPool.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just M
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Users\Ahmed Samir\Desktop\Azure\Event Hub VS Solution\EventHubsSender\bin\Debug\netcoreapp3.1\System.Memory.Data.dll'. Skipped loading symbols. Module is optimized and
'EventHubsSender.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\3.1.21\System.Diagnostics.DiagnosticSource.dll'. Skipped loading symbols. Module is optimized and the debugger option
The thread 0x694c has exited with code 0 (0x0).
```

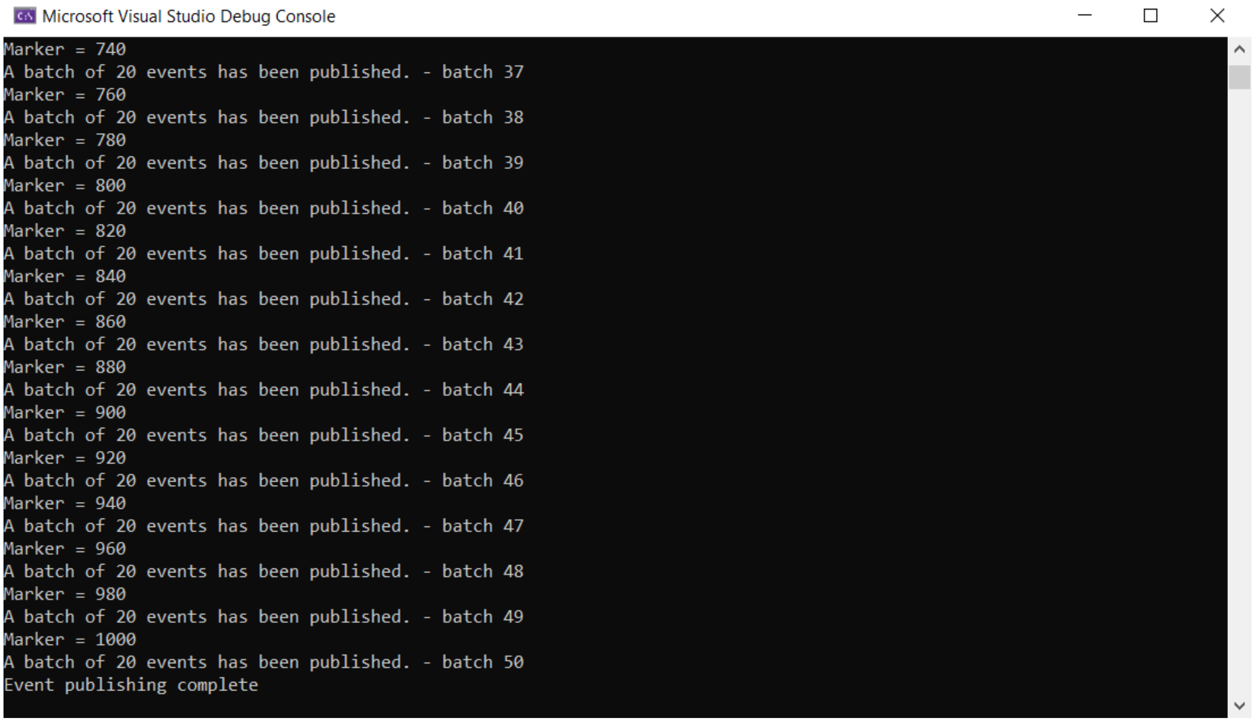
The initial code is reading the data from the Json file line by line but in our case we can read the entire file at once.



```
                countryDataCollection.Clear();

                if (!File.Exists(FileName))
                {
                    throw new Exception(String.Format("File {0} does not exist", FileName));
                }

                StreamReader reader = new StreamReader(FileName);
                //string line;

                //while ((line = reader.ReadLine()) != null){
                //    var countryData = JsonConvert.DeserializeObject<CountryData>(line);
                //    countryDataCollection.Add(countryData);
                //    endMarker += 1;
                //}

                //https://stackoverflow.com/questions/13297563/read-and-parse-a-json-file-in-c-sharp
                using (StreamReader r = new StreamReader(FileName))
                {
                    string json = r.ReadToEnd();
                    countryDataCollection = JsonConvert.DeserializeObject<List<CountryData>>(json);
                }
                endMarker = countryDataCollection.Count();
            }

            1 reference
            public List<CountryData> GetBatch(int size=20)
            {
                var list = new List<CountryData>();
                int end = ((marker + size) > endMarker) ? endMarker : marker + size;
```

```
Unable to open bike_data.json in the JSON editor because it exceeds the 5 MB limit. Opening as plain text instead.
```

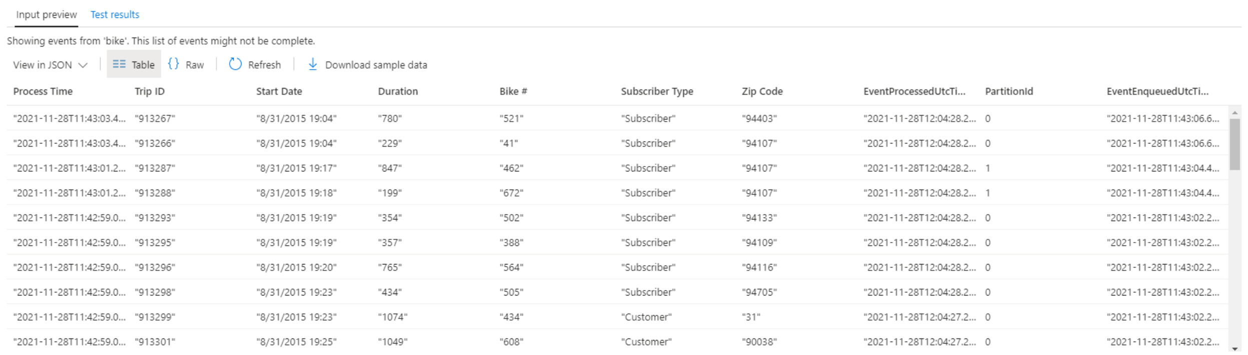
Finally, we can send the batches to the Azure and as we can see it sent 50 batches 1000 record and 50 (20*50)



```
Marker = 740
A batch of 20 events has been published. - batch 37
Marker = 760
A batch of 20 events has been published. - batch 38
Marker = 780
A batch of 20 events has been published. - batch 39
Marker = 800
A batch of 20 events has been published. - batch 40
Marker = 820
A batch of 20 events has been published. - batch 41
Marker = 840
A batch of 20 events has been published. - batch 42
Marker = 860
A batch of 20 events has been published. - batch 43
Marker = 880
A batch of 20 events has been published. - batch 44
Marker = 900
A batch of 20 events has been published. - batch 45
Marker = 920
A batch of 20 events has been published. - batch 46
Marker = 940
A batch of 20 events has been published. - batch 47
Marker = 960
A batch of 20 events has been published. - batch 48
Marker = 980
A batch of 20 events has been published. - batch 49
Marker = 1000
A batch of 20 events has been published. - batch 50
Event publishing complete
```

b) We went to the event hub's namespace and show that the entire message set was received. Provide one or more screenshots.

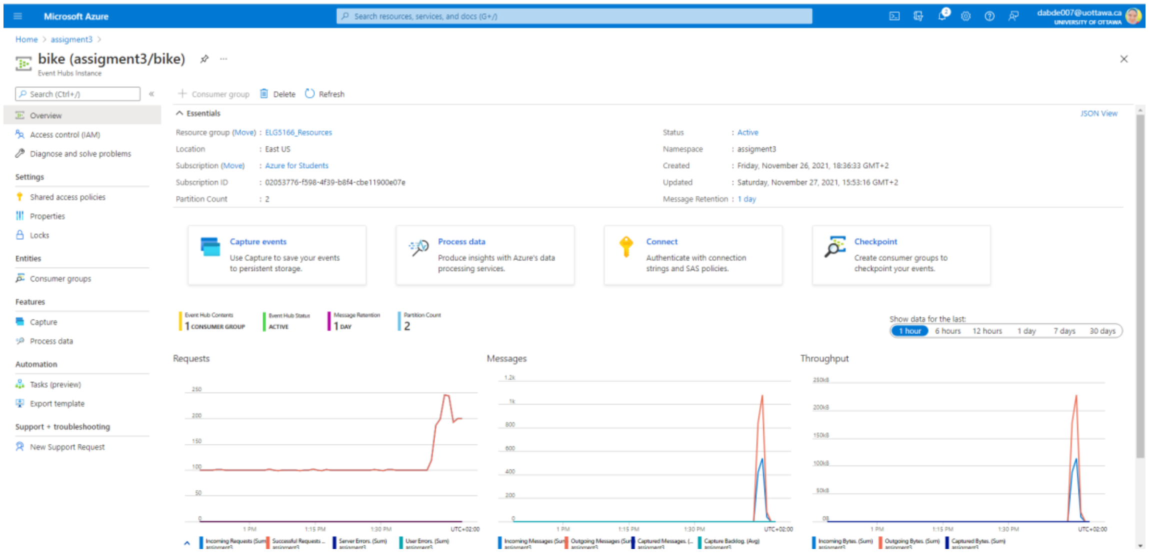
This screen represents a sample of the receive data



Showing events from 'bike'. This list of events might not be complete.

Process Time	Trip ID	Start Date	Duration	Bike #	Subscriber Type	Zip Code	EventProcessedUtcTi...	PartitionId	EventEnqueuedUtcTi...
"2021-11-28T11:43:03.4...	"913267"	"8/31/2015 19:04"	"780"	"521"	"Subscriber"	"94403"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:06.6...
"2021-11-28T11:43:03.4...	"913266"	"8/31/2015 19:04"	"229"	"41"	"Subscriber"	"94107"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:06.6...
"2021-11-28T11:43:01.2...	"913287"	"8/31/2015 19:17"	"847"	"462"	"Subscriber"	"94107"	"2021-11-28T12:04:28.2...	1	"2021-11-28T11:43:04.4...
"2021-11-28T11:43:01.2...	"913288"	"8/31/2015 19:18"	"199"	"672"	"Subscriber"	"94107"	"2021-11-28T12:04:28.2...	1	"2021-11-28T11:43:04.4...
"2021-11-28T11:42:59.0...	"913293"	"8/31/2015 19:19"	"354"	"502"	"Subscriber"	"94133"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:02.2...
"2021-11-28T11:42:59.0...	"913295"	"8/31/2015 19:19"	"357"	"388"	"Subscriber"	"94109"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:02.2...
"2021-11-28T11:42:59.0...	"913296"	"8/31/2015 19:20"	"765"	"564"	"Subscriber"	"94116"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:02.2...
"2021-11-28T11:42:59.0...	"913298"	"8/31/2015 19:23"	"434"	"505"	"Subscriber"	"94705"	"2021-11-28T12:04:28.2...	0	"2021-11-28T11:43:02.2...
"2021-11-28T11:42:59.0...	"913299"	"8/31/2015 19:23"	"1074"	"434"	"Customer"	"31"	"2021-11-28T12:04:27.2...	0	"2021-11-28T11:43:02.2...
"2021-11-28T11:42:59.0...	"913301"	"8/31/2015 19:25"	"1049"	"608"	"Customer"	"90038"	"2021-11-28T12:04:27.2...	0	"2021-11-28T11:43:02.2...

From the event hub's namespace, we can see that the entire message set was received



Home > assigment3 >

bike (assigment3/bike)

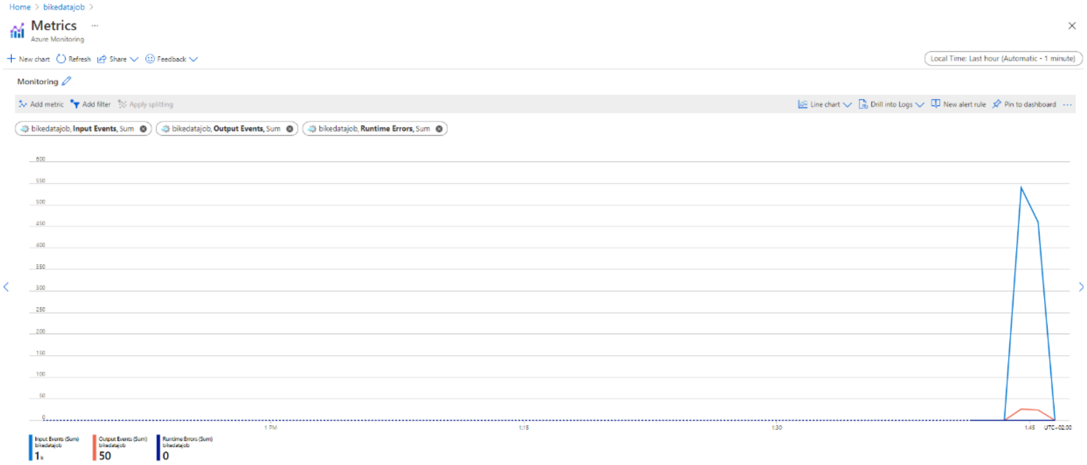
Event Hubs Instance

Essentials

Resource group (Move)	: ELG5166_Resources	Status	: Active
Location	: East US	Namespace	: assigment3
Subscription (Move)	: Azure for Students	Created	: Friday, November 26, 2021, 18:36:33 GMT+2
Subscription ID	: 02053776-f598-4f39-b8f4-cbe11900e07e	Updated	: Saturday, November 27, 2021, 15:53:16 GMT+2
Partition Count	: 2	Message Retention	: 1 day

Event Hub Contents: 1 CONSUMER GROUP
Event Hub Status: ACTIVE
Message Retention: 1 DAY
Partition Count: 2

1000 massage as 50 events



Home > bikedatajob >

Metrics

Azure Monitoring

Local Time: Last hour (Automatic - 1 minute)

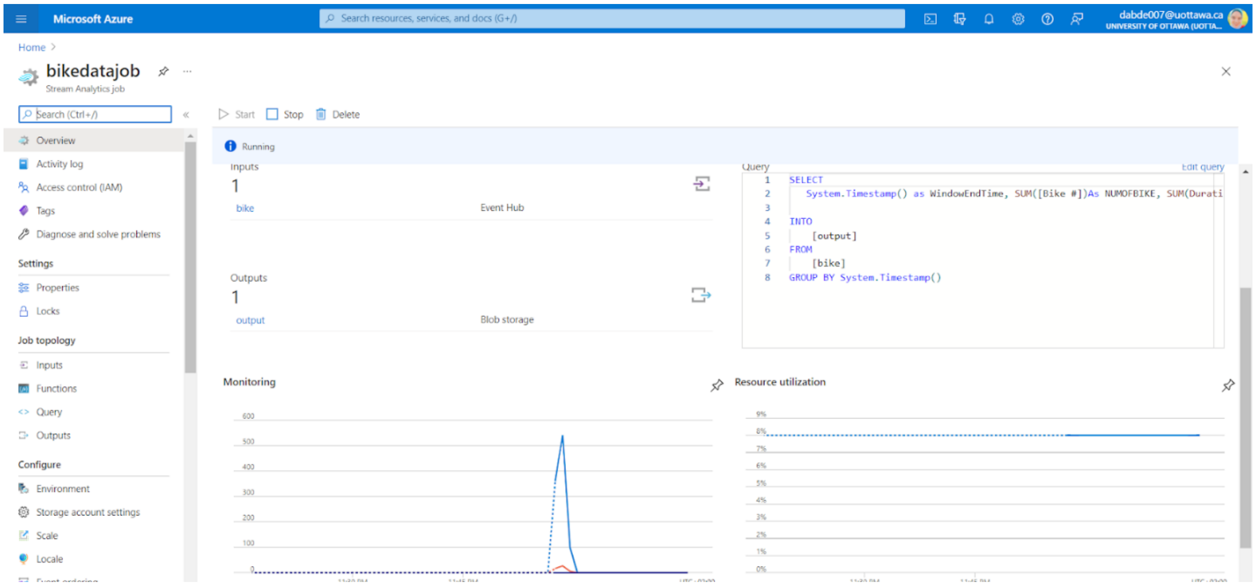
Monitoring

bikedatajob, Input Events, Sum
bikedatajob, Output Events, Sum
bikedatajob, Runtime Errors, Sum

Input Events (Sum) bikedatajob: 1k
Output Events (Sum) bikedatajob: 50
Runtime Errors (Sum) bikedatajob: 0

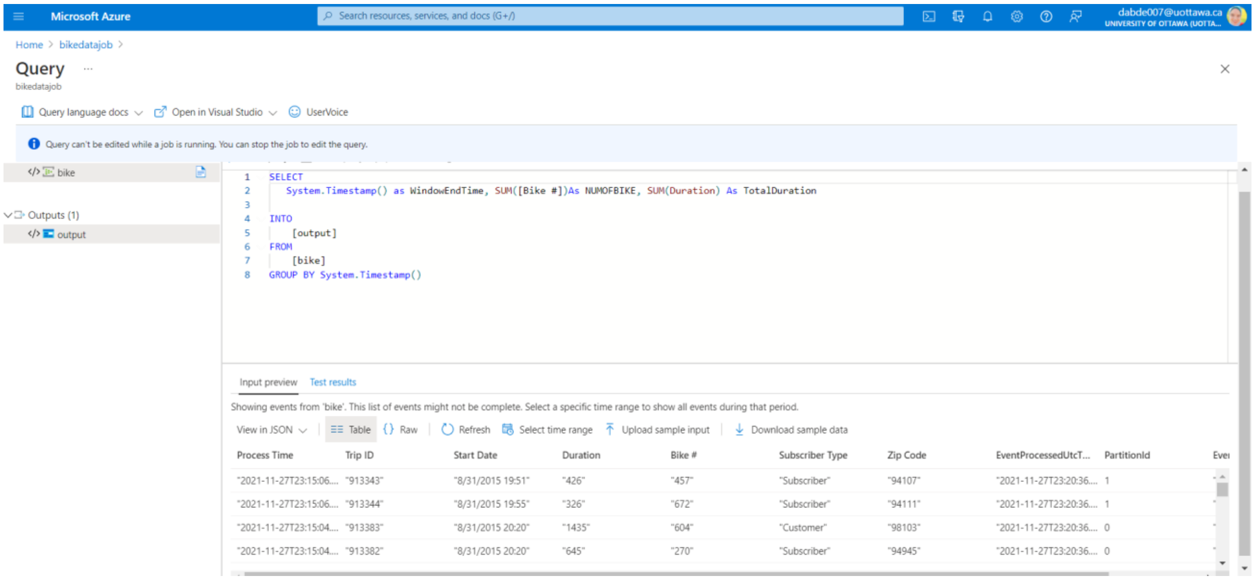
c) Then we Created an Azure Stream Analytics job that uses either a Storage account or SQL Database to store the summary of the event received. Our summary should include the following fields – WindowEnd, Total Bikes, Total Duration for each Batch received. Download this dataset and include both a screenshot and the data as part of your submission.

Then, we created stream analytics job



```
SELECT
   System.Timestamp() as WindowEndTime, SUM([Bike #])As NUMOFBIKE, SUM(Durati

INTO
    [output]
FROM
    [bike]
GROUP BY System.Timestamp()
```

calculate Total Bikes, Total Duration and WindowEnd for each Batch received



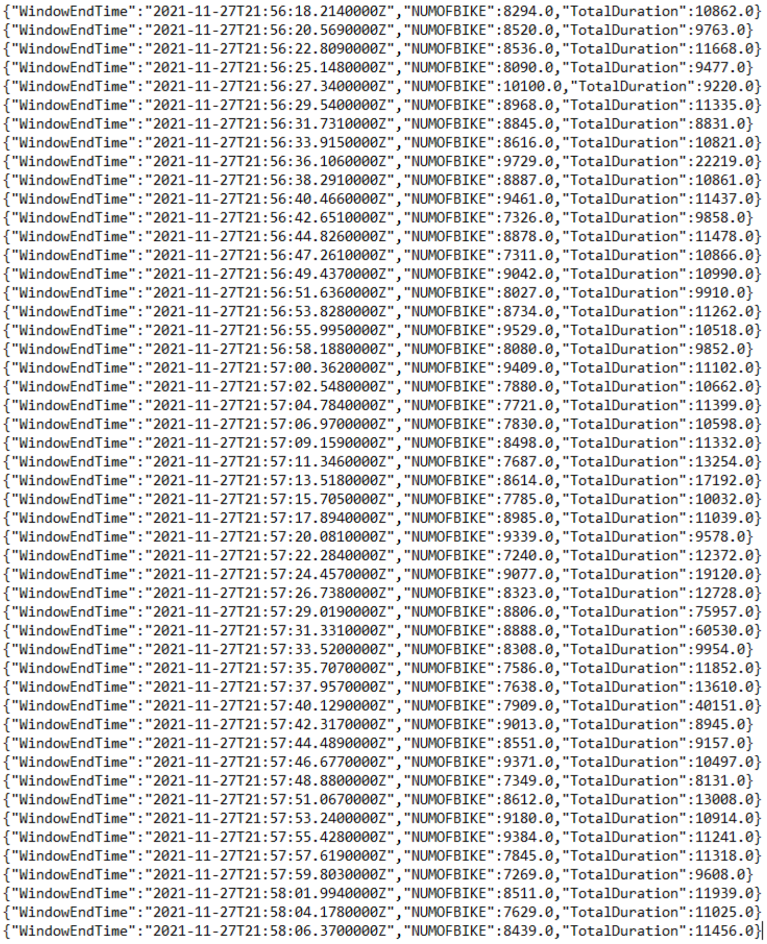
Query can't be edited while a job is running. You can stop the job to edit the query.

```
SELECT
   System.Timestamp() as WindowEndTime, SUM([Bike #])As NUMOFBIKE, SUM(Duration) As TotalDuration

INTO
    [output]
FROM
    [bike]
GROUP BY System.Timestamp()
```

Showing events from 'bike'. This list of events might not be complete. Select a specific time range to show all events during that period.

Process Time	Trip ID	Start Date	Duration	Bike #	Subscriber Type	Zip Code	EventProcessedUtcT...	PartitionId	Evei
"2021-11-27T23:15:06....	"913343"	"8/31/2015 19:51"	"426"	"457"	"Subscriber"	"94107"	"2021-11-27T23:20:36....	1	
"2021-11-27T23:15:06....	"913344"	"8/31/2015 19:55"	"326"	"672"	"Subscriber"	"94111"	"2021-11-27T23:20:36....	1	
"2021-11-27T23:15:04....	"913383"	"8/31/2015 20:20"	"1435"	"604"	"Customer"	"98103"	"2021-11-27T23:20:36....	0	
"2021-11-27T23:15:04....	"913382"	"8/31/2015 20:20"	"645"	"270"	"Subscriber"	"94945"	"2021-11-27T23:20:36....	0	

Data Screenshots:



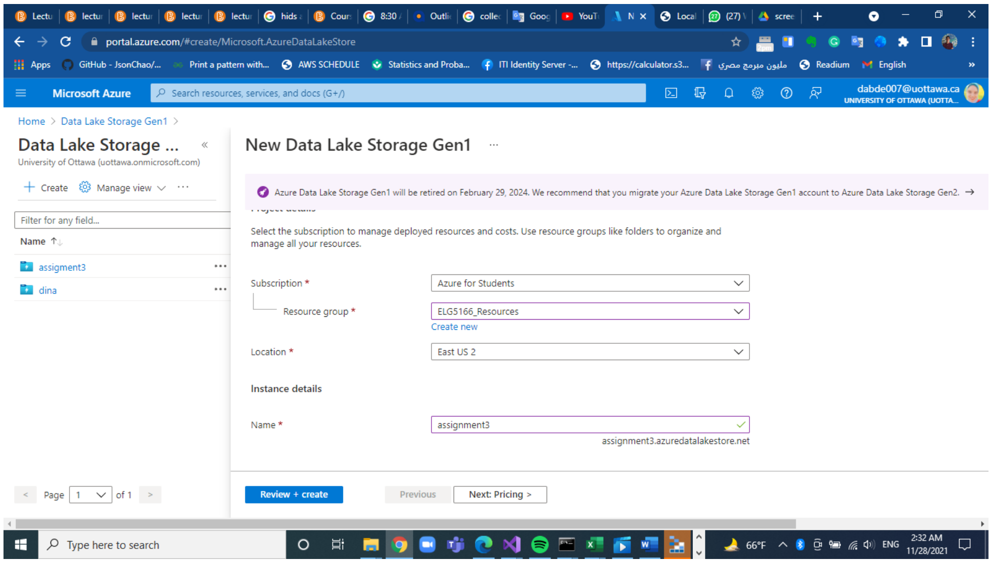
```
{"WindowEndTime":"2021-11-27T21:56:18.2140000Z","NUMOFBIKE":8294.0,"TotalDuration":10862.0}
{"WindowEndTime":"2021-11-27T21:56:20.5690000Z","NUMOFBIKE":8520.0,"TotalDuration":9763.0}
{"WindowEndTime":"2021-11-27T21:56:22.8090000Z","NUMOFBIKE":8536.0,"TotalDuration":11668.0}
{"WindowEndTime":"2021-11-27T21:56:25.1480000Z","NUMOFBIKE":8090.0,"TotalDuration":9477.0}
{"WindowEndTime":"2021-11-27T21:56:27.3400000Z","NUMOFBIKE":10100.0,"TotalDuration":9220.0}
{"WindowEndTime":"2021-11-27T21:56:29.5400000Z","NUMOFBIKE":8968.0,"TotalDuration":11335.0}
{"WindowEndTime":"2021-11-27T21:56:31.7310000Z","NUMOFBIKE":8845.0,"TotalDuration":8831.0}
{"WindowEndTime":"2021-11-27T21:56:33.9150000Z","NUMOFBIKE":8616.0,"TotalDuration":10821.0}
{"WindowEndTime":"2021-11-27T21:56:36.1060000Z","NUMOFBIKE":9729.0,"TotalDuration":22219.0}
{"WindowEndTime":"2021-11-27T21:56:38.2910000Z","NUMOFBIKE":8887.0,"TotalDuration":10861.0}
{"WindowEndTime":"2021-11-27T21:56:40.4660000Z","NUMOFBIKE":9461.0,"TotalDuration":11437.0}
{"WindowEndTime":"2021-11-27T21:56:42.6510000Z","NUMOFBIKE":7326.0,"TotalDuration":9858.0}
{"WindowEndTime":"2021-11-27T21:56:44.8260000Z","NUMOFBIKE":8878.0,"TotalDuration":11478.0}
{"WindowEndTime":"2021-11-27T21:56:47.2610000Z","NUMOFBIKE":7311.0,"TotalDuration":10866.0}
{"WindowEndTime":"2021-11-27T21:56:49.4370000Z","NUMOFBIKE":9042.0,"TotalDuration":10990.0}
{"WindowEndTime":"2021-11-27T21:56:51.6360000Z","NUMOFBIKE":8027.0,"TotalDuration":9910.0}
{"WindowEndTime":"2021-11-27T21:56:53.8280000Z","NUMOFBIKE":8734.0,"TotalDuration":11262.0}
{"WindowEndTime":"2021-11-27T21:56:55.9950000Z","NUMOFBIKE":9529.0,"TotalDuration":10518.0}
{"WindowEndTime":"2021-11-27T21:56:58.1880000Z","NUMOFBIKE":8080.0,"TotalDuration":9852.0}
{"WindowEndTime":"2021-11-27T21:57:00.3620000Z","NUMOFBIKE":9409.0,"TotalDuration":11102.0}
{"WindowEndTime":"2021-11-27T21:57:02.5480000Z","NUMOFBIKE":7880.0,"TotalDuration":10662.0}
{"WindowEndTime":"2021-11-27T21:57:04.7840000Z","NUMOFBIKE":7721.0,"TotalDuration":11399.0}
{"WindowEndTime":"2021-11-27T21:57:06.9700000Z","NUMOFBIKE":7830.0,"TotalDuration":10598.0}
{"WindowEndTime":"2021-11-27T21:57:09.1590000Z","NUMOFBIKE":8498.0,"TotalDuration":11332.0}
{"WindowEndTime":"2021-11-27T21:57:11.3460000Z","NUMOFBIKE":7687.0,"TotalDuration":13254.0}
{"WindowEndTime":"2021-11-27T21:57:13.5180000Z","NUMOFBIKE":8614.0,"TotalDuration":17192.0}
{"WindowEndTime":"2021-11-27T21:57:15.7050000Z","NUMOFBIKE":7785.0,"TotalDuration":10032.0}
{"WindowEndTime":"2021-11-27T21:57:17.8940000Z","NUMOFBIKE":8985.0,"TotalDuration":11039.0}
{"WindowEndTime":"2021-11-27T21:57:20.0810000Z","NUMOFBIKE":9339.0,"TotalDuration":9578.0}
{"WindowEndTime":"2021-11-27T21:57:22.2840000Z","NUMOFBIKE":7240.0,"TotalDuration":12372.0}
{"WindowEndTime":"2021-11-27T21:57:24.4570000Z","NUMOFBIKE":9077.0,"TotalDuration":19120.0}
{"WindowEndTime":"2021-11-27T21:57:26.7380000Z","NUMOFBIKE":8323.0,"TotalDuration":12728.0}
{"WindowEndTime":"2021-11-27T21:57:29.0190000Z","NUMOFBIKE":8806.0,"TotalDuration":75957.0}
{"WindowEndTime":"2021-11-27T21:57:31.3310000Z","NUMOFBIKE":8888.0,"TotalDuration":60530.0}
{"WindowEndTime":"2021-11-27T21:57:33.5200000Z","NUMOFBIKE":8308.0,"TotalDuration":9954.0}
{"WindowEndTime":"2021-11-27T21:57:35.7070000Z","NUMOFBIKE":7586.0,"TotalDuration":11852.0}
{"WindowEndTime":"2021-11-27T21:57:37.9570000Z","NUMOFBIKE":7638.0,"TotalDuration":13610.0}
{"WindowEndTime":"2021-11-27T21:57:40.1290000Z","NUMOFBIKE":7909.0,"TotalDuration":40151.0}
{"WindowEndTime":"2021-11-27T21:57:42.3170000Z","NUMOFBIKE":9013.0,"TotalDuration":8945.0}
{"WindowEndTime":"2021-11-27T21:57:44.4890000Z","NUMOFBIKE":8551.0,"TotalDuration":9157.0}
{"WindowEndTime":"2021-11-27T21:57:46.6770000Z","NUMOFBIKE":9371.0,"TotalDuration":10497.0}
{"WindowEndTime":"2021-11-27T21:57:48.8800000Z","NUMOFBIKE":7349.0,"TotalDuration":8131.0}
{"WindowEndTime":"2021-11-27T21:57:51.0670000Z","NUMOFBIKE":8612.0,"TotalDuration":13008.0}
{"WindowEndTime":"2021-11-27T21:57:53.2400000Z","NUMOFBIKE":9180.0,"TotalDuration":10914.0}
{"WindowEndTime":"2021-11-27T21:57:55.4280000Z","NUMOFBIKE":9384.0,"TotalDuration":11241.0}
{"WindowEndTime":"2021-11-27T21:57:57.6190000Z","NUMOFBIKE":7845.0,"TotalDuration":11318.0}
{"WindowEndTime":"2021-11-27T21:57:59.8030000Z","NUMOFBIKE":7269.0,"TotalDuration":9608.0}
{"WindowEndTime":"2021-11-27T21:58:01.9940000Z","NUMOFBIKE":8511.0,"TotalDuration":11939.0}
{"WindowEndTime":"2021-11-27T21:58:04.1780000Z","NUMOFBIKE":7629.0,"TotalDuration":11025.0}
{"WindowEndTime":"2021-11-27T21:58:06.3700000Z","NUMOFBIKE":8439.0,"TotalDuration":11456.0}
```

We can see that we have 50 record each record represents the number of bikes and the total duration per batch.

2. Azure Data Lake Analytics

2.1. Create a Data Lake Storage Gen2 account

To create ADL (Azure Data Lake), It is required to create Data Lake Storage in Figure-1



New Data Lake Storage Gen1

Azure Data Lake Storage Gen1 will be retired on February 29, 2024. We recommend that you migrate your Azure Data Lake Storage Gen1 account to Azure Data Lake Storage Gen2.

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure for Students

Resource group * ELG5166_Resources

Location * East US 2

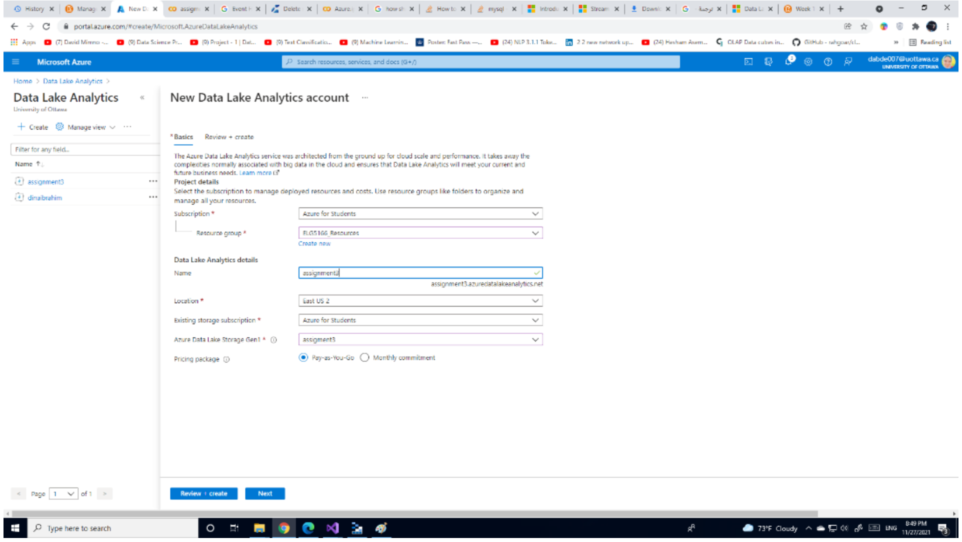
Instance details

Name * assignment3

assignment3.azuredatalakestore.net

Figure 1- Create a Data Lake Storage Gen2 account

2.2. Create Data Analytics Account



New Data Lake Analytics account

The Azure Data Lake Analytics service was architected from the ground up for cloud scale and performance. It takes away the complexities normally associated with big data in the cloud and ensures that Data Lake Analytics will meet your current and future business needs. Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * Azure for Students

Resource group * ELG5166_Resources

Data Lake Analytics details

Name assignment3

assignment3.azuredatalakeanalytics.net

Location * East US 2

Existing storage subscription * Azure for Students

Azure Data Lake Storage Gen1 * assigment3

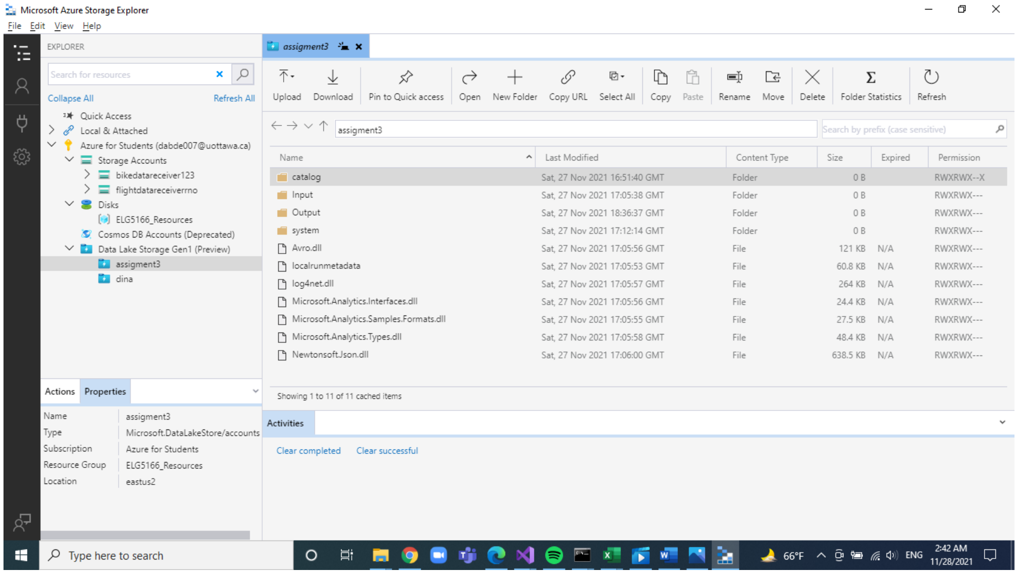
Pricing package ● Pay-as-You-Go ○ Monthly commitment

Figure 2- Set up New Data Lake Analytics Account

2.3. Microsoft Azure Storage Explorer

Moving our data samples to the ADLS requires us first to create the Input and Output folders to upload the rest of the content, So Microsoft Azure Store Explorer was used.

Here in Figure 3, We have finished our connection with Microsoft Azure Storage Explorer with our local data by uploading Input and Output folders and by uploading all DLL files to connect Microsoft Visual Studio with Azure Data Lake Analytics to have live updates over cloud from our local machine.



Name	assigment3
Type	Microsoft.DataLakeStore/accounts
Subscription	Azure for Students
Resource Group	ELG5166_Resources
Location	eastus2

Name	Last Modified	Content Type	Size	Expired	Permission
catalog	Sat, 27 Nov 2021 16:51:40 GMT	Folder	0 B		RWXRWX--X
Input	Sat, 27 Nov 2021 17:05:38 GMT	Folder	0 B		RWXRWX---
Output	Sat, 27 Nov 2021 18:36:37 GMT	Folder	0 B		RWXRWX---
system	Sat, 27 Nov 2021 17:12:14 GMT	Folder	0 B		RWXRWX---
Avro.dll	Sat, 27 Nov 2021 17:05:56 GMT	File	121 KB	N/A	RWXRWX---
localrunmetadata	Sat, 27 Nov 2021 17:05:53 GMT	File	60.8 KB	N/A	RWXRWX---
log4net.dll	Sat, 27 Nov 2021 17:05:57 GMT	File	264 KB	N/A	RWXRWX---
Microsoft.Analytics.Interfaces.dll	Sat, 27 Nov 2021 17:05:56 GMT	File	24.4 KB	N/A	RWXRWX---
Microsoft.Analytics.Samples.Formats.dll	Sat, 27 Nov 2021 17:05:55 GMT	File	27.5 KB	N/A	RWXRWX---
Microsoft.Analytics.Types.dll	Sat, 27 Nov 2021 17:05:58 GMT	File	48.4 KB	N/A	RWXRWX---
Newtonsoft.Json.dll	Sat, 27 Nov 2021 17:06:00 GMT	File	638.5 KB	N/A	RWXRWX---

Showing 1 to 11 of 11 cached items

Figure 3 - Microsoft Azure Storage Explorer Finished Setup

2.4. U-SQL Queries

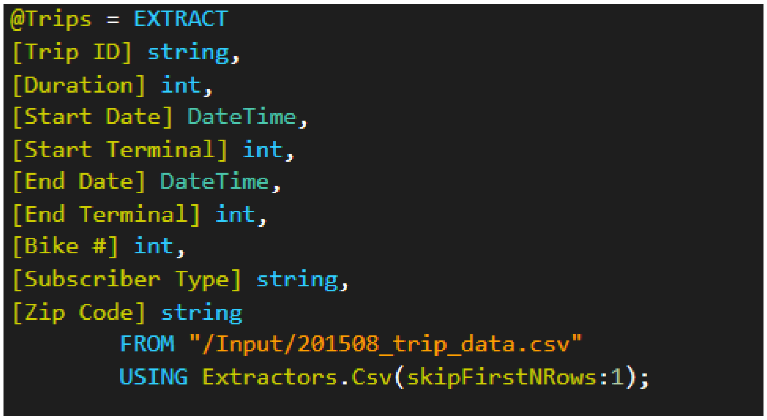
The tables in Tabel 1 and Table 2 represents a bikes rental data, In table 1, It represents bike trips the was done at specific time from starting terminal to end terminal having specific amount of bikes and the type of customers that had that trip that are living at specific zip code as their location.

In table 2, It describes the terminals information.

In this Assignment, We want to derive some meaningful information from this data.

Trip ID	Duration	Start Date	Start Terminal	End Date	End Terminal	Bike #	Subscriber Type	Zip Code
913460	765	8/31/2015 23:26	50	8/31/2015 23:39	70	288	Subscriber	2139
913459	1036	8/31/2015 23:11	31	8/31/2015 23:28	27	35	Subscriber	95032
913455	307	8/31/2015 23:13	47	8/31/2015 23:18	64	468	Subscriber	94107
913454	409	8/31/2015 23:10	10	8/31/2015 23:17	8	68	Subscriber	95113
913453	789	8/31/2015 23:09	51	8/31/2015 23:22	60	487	Customer	9069

Table 1- Bikes Trips

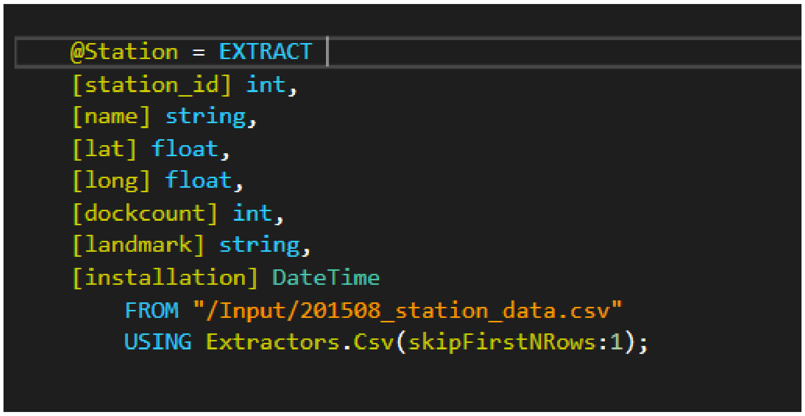


```
@Trips = EXTRACT
[Trip ID] string,
[Duration] int,
[Start Date] DateTime,
[Start Terminal] int,
[End Date] DateTime,
[End Terminal] int,
[Bike #] int,
[Subscriber Type] string,
[Zip Code] string
        FROM "/Input/201508_trip_data.csv"
        USING Extractors.Csv(skipFirstNRows:1);
```

Figure 4 - Trips Table in U-SQL

station_id	name	lat	long	dockcount	landmark	installation
2	San Jose Diridon Caltrain Station	37.32973	-121.902	27	San Jose	8/6/2013
3	San Jose Civic Center	37.3307	-121.889	15	San Jose	8/5/2013
4	Santa Clara at Almaden	37.33399	-121.895	11	San Jose	8/6/2013
5	Adobe on Almaden	37.33142	-121.893	19	San Jose	8/5/2013

Table 2 - Terminal Data



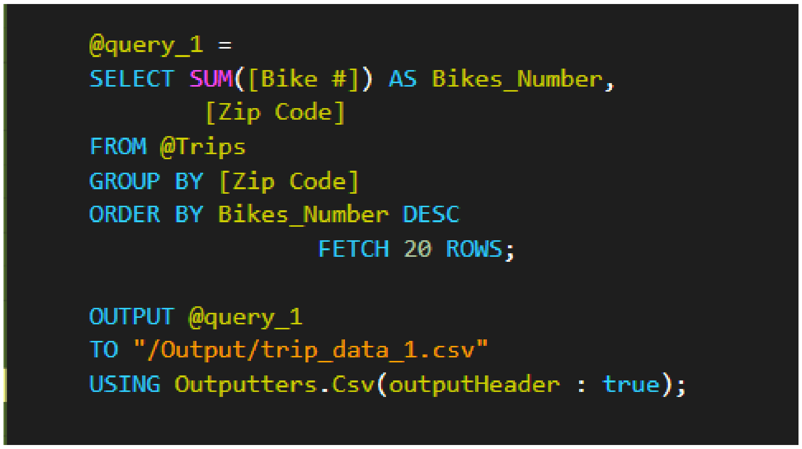
```
@Station = EXTRACT
[station_id] int,
[name] string,
[lat] float,
[long] float,
[dockcount] int,
[landmark] string,
[installation] DateTime
    FROM "/Input/201508_station_data.csv"
    USING Extractors.Csv(skipFirstNRows:1);
```

Figure 5 - Stations Table in U-SQL

Queries

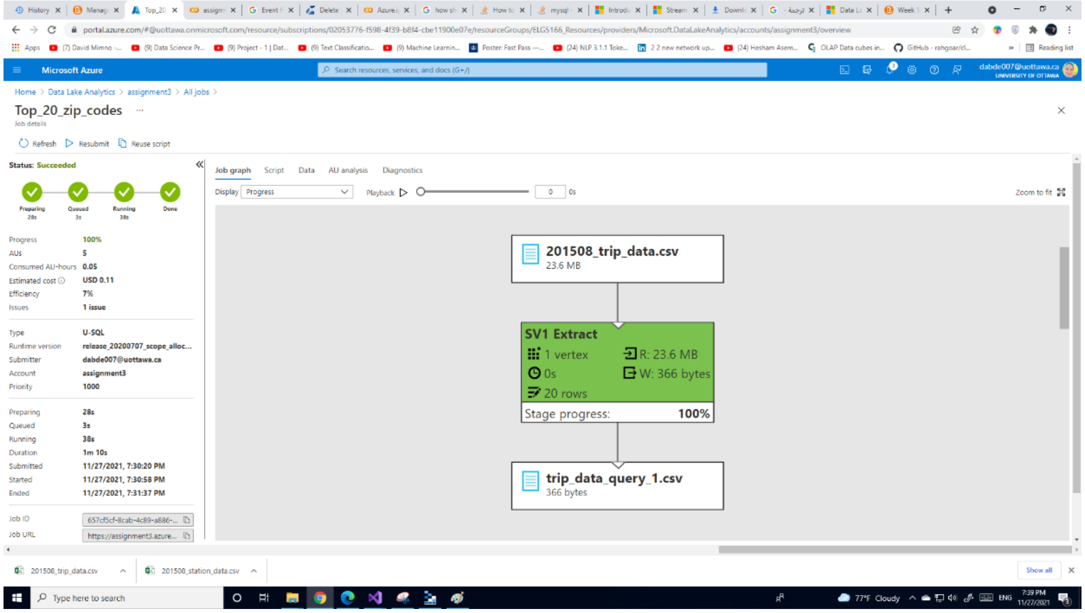
1. Top 20 zip codes where most bikes were rented from

Assumptions: we have assumed [Bike #] is the number of bikes for one trip because it was mentioned “#” right after the “Bike”.
From **Table 1**, the sum over [Bike #] was calculated then group by the [zip code] followed by Order by the sum of bikes descending, and finally fetching the first 20 rows.
Result: In table 4, we have the highest 20 zip codes that have bike number.



```
@query_1 =
SELECT SUM([Bike #]) AS Bikes_Number,
        [Zip Code]
FROM @Trips
GROUP BY [Zip Code]
ORDER BY Bikes_Number DESC
                FETCH 20 ROWS;

OUTPUT @query_1
TO "/Output/trip_data_1.csv"
USING Outputters.Csv(outputHeader : true);
```

Figure 6- Top 20 zip codes query



Top_20_zip_codes
Job details

Status: Succeeded

Preparing 28s · Queued 3s · Running 38s · Done

Progress	100%
AUs	5
Consumed AU-hours	0.05
Estimated cost	USD 0.11
Efficiency	7%
Issues	1 issue
Type	U-SQL
Runtime version	release_20200707_scope_alloc...
Submitter	dabde007@uottawa.ca
Account	assignment3
Priority	1000
Preparing	28s
Queued	3s
Running	38s
Duration	1m 10s
Submitted	11/27/2021, 7:30:20 PM
Started	11/27/2021, 7:30:58 PM
Ended	11/27/2021, 7:31:37 PM
Job ID	657cf5cf-8cab-4c89-a886-...
Job URL	https://assignment3.azure...

Figure 7 - Query 1 graph

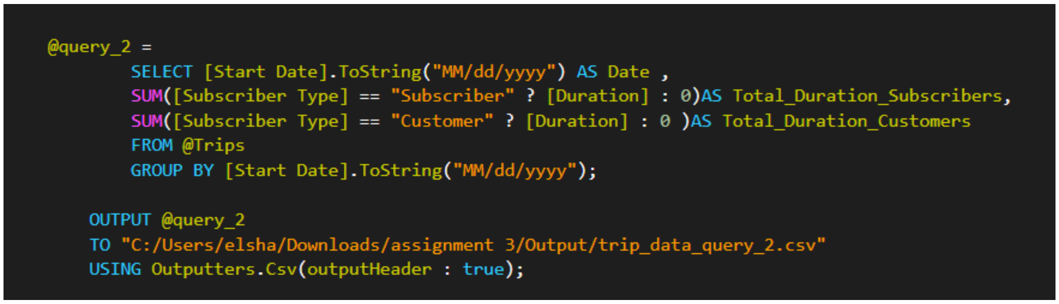
Zip Code	Bikes_Number
94107	20228868
94105	8788689
94133	7077405
94103	6353674
94111	4782022

Table 3 - Top 20 zip codes results

2. Daily duration aggregate across the rental subscriber types.

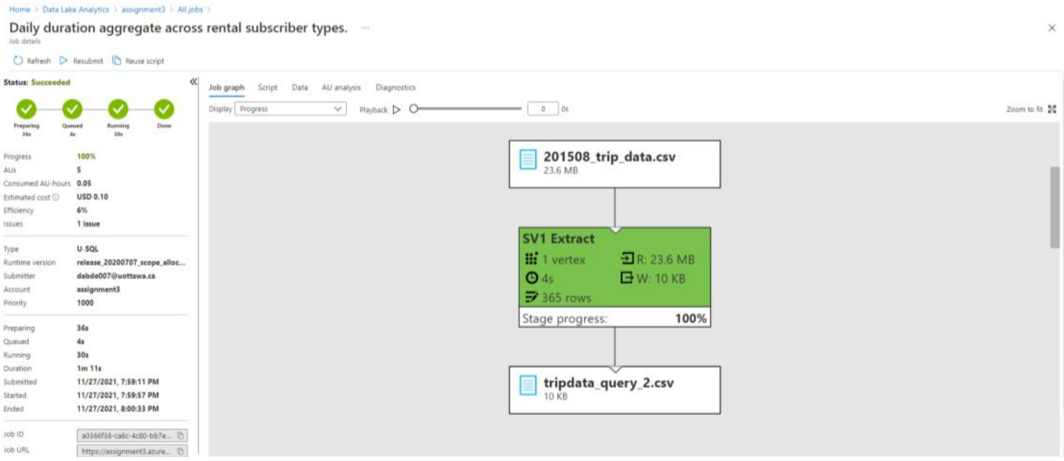
From **Table 1**, The sum over the single day trip durations was calculated separately upon each type by using CASE statement for each subscriber type, then grouping by the day duration was performed

Result: In Table 4, The duration of all trips of each day for both subscriber types.



```
@query_2 =
        SELECT [Start Date].ToString("MM/dd/yyyy") AS Date ,
        SUM([Subscriber Type] == "Subscriber" ? [Duration] : 0)AS Total_Duration_Subscribers,
        SUM([Subscriber Type] == "Customer" ? [Duration] : 0 )AS Total_Duration_Customers
        FROM @Trips
        GROUP BY [Start Date].ToString("MM/dd/yyyy");

    OUTPUT @query_2
    TO "C:/Users/elsha/Downloads/assignment 3/Output/trip_data_query_2.csv"
    USING Outputters.Csv(outputHeader : true);
```

Figure 8 - Daily duration aggregation Query



Home > Data Lake Analytics > assignment3 > All jobs >

Daily duration aggregate across rental subscriber types.

Job details

Status: Succeeded

Progress	100%
AUs	5
Consumed AU-hours	0.05
Estimated cost	USD 0.10
Efficiency	6%
Issues	1 issue
Type	U-SQL
Runtime version	release_20200707_scope_alloc...
Submitter	dabde007@uottawa.ca
Account	assignment3
Priority	1000
Preparing	36s
Queued	4s
Running	30s
Duration	1m 11s
Submitted	11/27/2021, 7:59:11 PM
Started	11/27/2021, 7:59:57 PM
Ended	11/27/2021, 8:00:33 PM
Job ID	a0366f38-ca8c-4c80-bb7e...
Job URL	https://assignment3.azure...

Display: Progress

Figure 9 - Daily Duration Aggregation graph

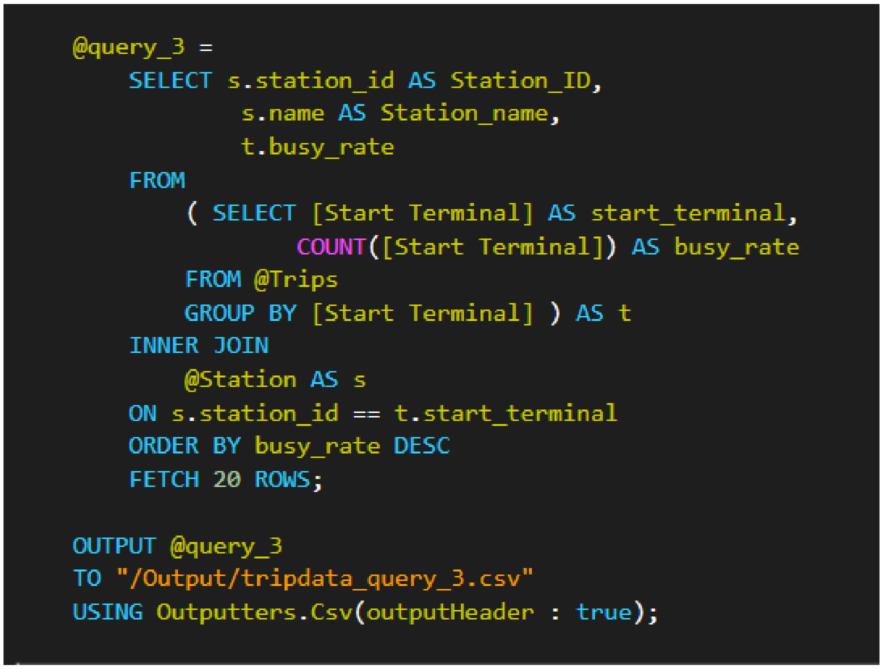
Date	Total_Duration_Subscribers	Total_Duration_Customers
1/1/2015	37943	344997
1/2/2015	154422	772045
1/3/2015	80711	460716
1/4/2015	68021	489093
1/5/2015	594816	159695
1/6/2015	625803	187769

Table 4 - Daily Duration aggregation results

3. The top busiest terminals for bike pickup

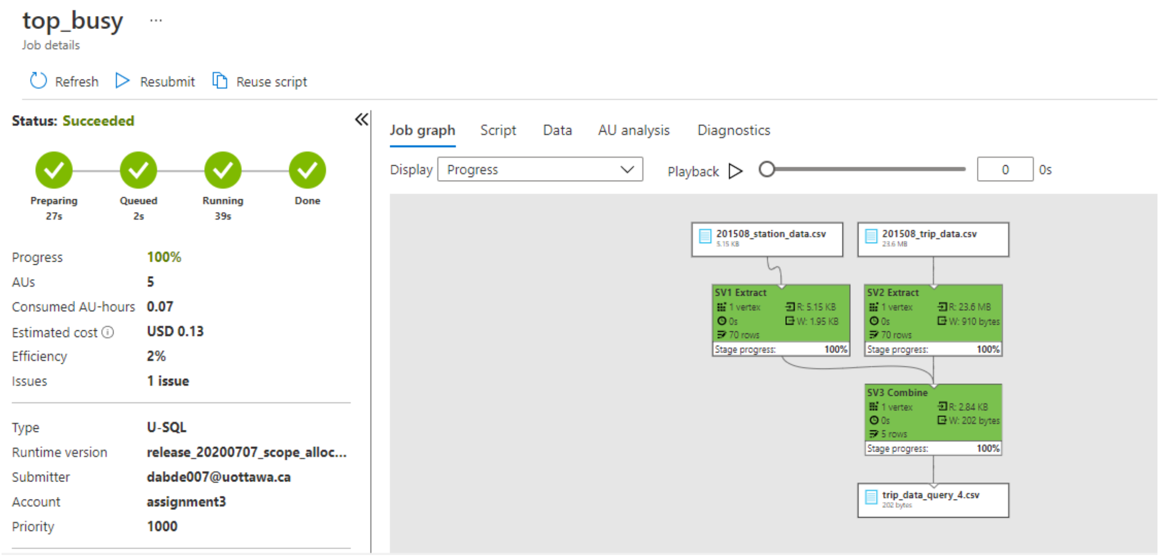
Assumption: we have assumed that bike pickup means that the terminals that the bikes started their trip from, hence the start terminal was used in the query, And the busiest start terminal would have the most occurrences in the data.

From **Table 1**, we have grouped by the [start terminal] and selecting the count of all terminal occurrences to have the frequency of all start terminals and selected the start terminal number.
Then we have performed an "INNER JOIN" with **Table 2** with the [station_id] to get the names of the terminals from **Table 2,** and lastly ordering by the frequency of terminals [busy_rate] descending and then fetching the top 20 records.



```
@query_3 =
    SELECT s.station_id AS Station_ID,
            s.name AS Station_name,
            t.busy_rate
    FROM
        ( SELECT [Start Terminal] AS start_terminal,
                COUNT([Start Terminal]) AS busy_rate
        FROM @Trips
        GROUP BY [Start Terminal] ) AS t
    INNER JOIN
        @Station AS s
    ON s.station_id == t.start_terminal
    ORDER BY busy_rate DESC
    FETCH 20 ROWS;

OUTPUT @query_3
TO "/Output/tripdata_query_3.csv"
USING Outputters.Csv(outputHeader : true);
```

Table 5 - top most busy start teminals query



Status: Succeeded

Progress	100%
AUs	5
Consumed AU-hours	0.07
Estimated cost	USD 0.13
Efficiency	2%
Issues	1 issue
Type	U-SQL
Runtime version	release_20200707_scope_alloc...
Submitter	dabde007@uottawa.ca
Account	assignment3
Priority	1000

Table 6 - top most busy start teminals graph

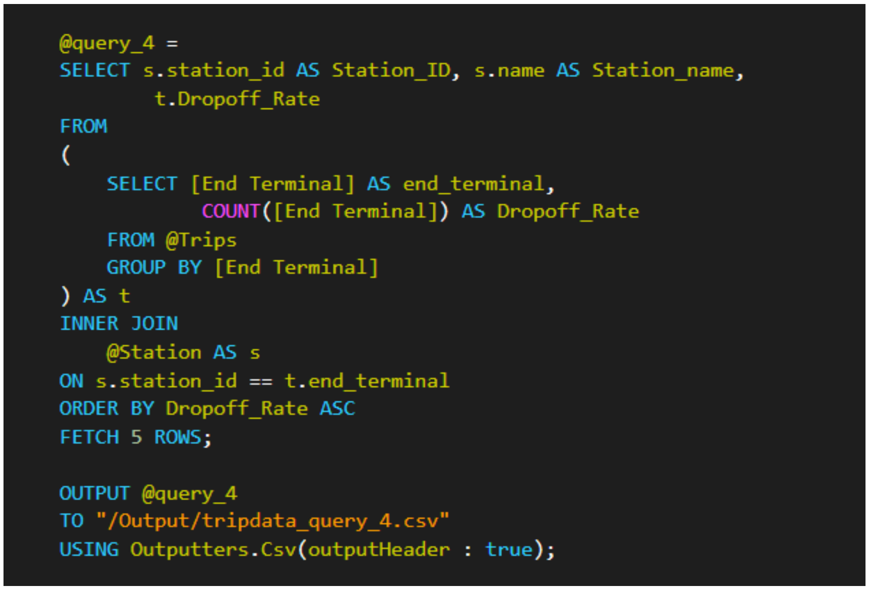
Station_ID	Station_name	busy_rate
70	San Francisco Caltrain (Townsend at 4th)	26304
69	San Francisco Caltrain 2 (330 Townsend)	21758
50	Harry Bridges Plaza (Ferry Building)	17255
55	Temporary Transbay Terminal (Howard at Beale)	14436
60	Embarcadero at Sansome	14158

Table 7 - top most busy start teminals result

4. Which 5 terminal has the least drop-offs?

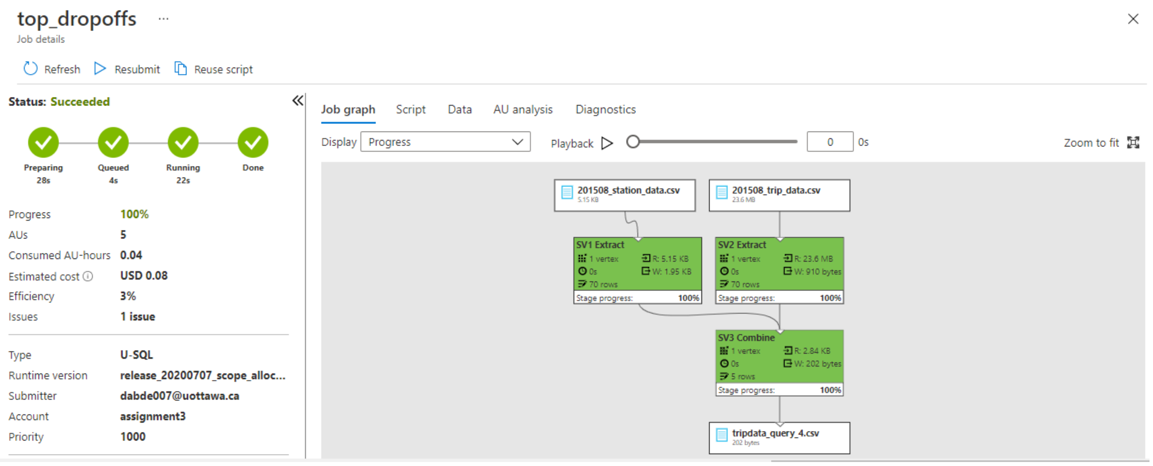
Assumption: we have assumed that bike drop-offs means that the terminals that the bikes ended their trip on, hence the [end terminal] was used in the query, And the least dropoff terminals would have the least occurrences in the dataset.

From **Table 1**, we have grouped by the [end terminal] and selecting the count of all terminal occurrences to have the frequency of all end terminals and selected the end terminal number.
Then we have performed an "INNER JOIN" with **Table 2** with the [station_id] to get the names of the terminals from **Table 2,** and lastly ordering by the frequency of terminals [Dropoff_Rate] ascending and then fetching the top 5 records.



```
@query_4 =
SELECT s.station_id AS Station_ID, s.name AS Station_name,
        t.Dropoff_Rate
FROM
(
    SELECT [End Terminal] AS end_terminal,
            COUNT([End Terminal]) AS Dropoff_Rate
    FROM @Trips
    GROUP BY [End Terminal]
) AS t
INNER JOIN
    @Station AS s
ON s.station_id == t.end_terminal
ORDER BY Dropoff_Rate ASC
FETCH 5 ROWS;

OUTPUT @query_4
TO "/Output/tripdata_query_4.csv"
USING Outputters.Csv(outputHeader : true);
```

Figure 10 - least terminal dropoffs query



top_dropoffs
Job details

Status: Succeeded

Preparing 28s
Queued 4s
Running 22s
Done

Progress	100%
AUs	5
Consumed AU-hours	0.04
Estimated cost	USD 0.08
Efficiency	3%
Issues	1 issue
Type	U-SQL
Runtime version	release_20200707_scope_alloc...
Submitter	dabde007@uottawa.ca
Account	assignment3
Priority	1000

Display: Progress

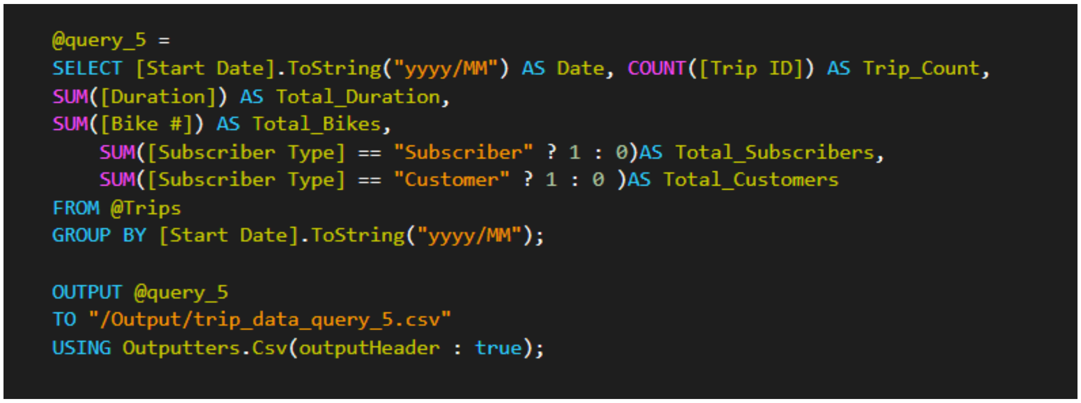
Figure 11 - least terminal dropoffs graph

Station_ID	Station_name	Dropoff_Rate
24	Redwood City Public Library	98
21	Franklin at Maple	100
83	Mezes Park	145
23	San Mateo County Center	187
26	Redwood City Medical Center	230

Table 8 - least terminal dropoffs results

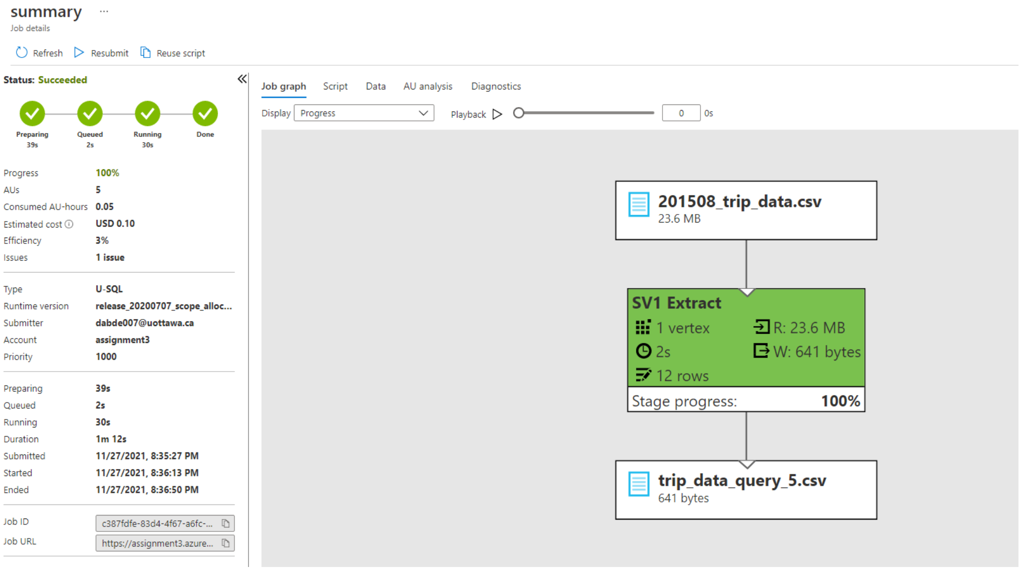
5. What is the monthly summary of bike rentals (format - month/year ex. 06/2020)

Total bikes, total duration (seconds), total trips and total subscriber types was calculated per month.



```
@query_5 =
SELECT [Start Date].ToString("yyyy/MM") AS Date, COUNT([Trip ID]) AS Trip_Count,
SUM([Duration]) AS Total_Duration,
SUM([Bike #]) AS Total_Bikes,
    SUM([Subscriber Type] == "Subscriber" ? 1 : 0)AS Total_Subscribers,
    SUM([Subscriber Type] == "Customer" ? 1 : 0 )AS Total_Customers
FROM @Trips
GROUP BY [Start Date].ToString("yyyy/MM");

OUTPUT @query_5
TO "/Output/trip_data_query_5.csv"
USING Outputters.Csv(outputHeader : true);
```

Figure 12 - summary query



summary

Job details

Status: Succeeded

Preparing 39s — Queued 2s — Running 30s — Done

Progress	100%
AUs	5
Consumed AU-hours	0.05
Estimated cost ⓘ	USD 0.10
Efficiency	3%
Issues	1 issue
Type	U-SQL
Runtime version	release_20200707_scope_alloc...
Submitter	dabde007@uottawa.ca
Account	assignment3
Priority	1000
Preparing	39s
Queued	2s
Running	30s
Duration	1m 12s
Submitted	11/27/2021, 8:35:27 PM
Started	11/27/2021, 8:36:13 PM
Ended	11/27/2021, 8:36:50 PM
Job ID	c387fdfe-83d4-4f67-a6fc-...
Job URL	https://assignment3.azure...

Display: Progress

Playback 0 0s

Figure 13 - summary graph

Date	Trip_Count	Total_Duration	Total_Bikes	Total_Subscribers	Total_Customers
2014/09	31682	33160021	13310244	27456	4226
2014/10	34220	33401099	14386280	29833	4387
2014/11	25516	22454934	10752350	22623	2893
2014/12	19677	41131402	8382802	17386	2291
2015/01	27840	25611358	11756577	25068	2772
2015/02	26401	25633016	11139951	23688	2713
2015/03	31626	29892301	13444028	27752	3874
2015/04	31363	28031940	13313721	28038	3325
2015/05	29540	31584633	12574550	25545	3995
2015/06	31907	34481927	13447867	27868	4039
2015/07	32476	33983062	13718375	27652	4824
2015/08	31904	31088866	13564645	27308	4596

Table 9 - Summary result

3. Definitions

1. Please compare briefly, based on at least 3 criteria, the differences in architecture between Apache Spark Structured Streaming and Azure Event Hubs & Stream Analytics?

Event hub and stream analytics work together as end-to-end solution where the event hub read the stream and feed it to the stream analytics to analyze this data. while spark input File source, Socket Source, Rate Source, Kafka Source

In stream analytics User can extend existing functions as UDFs in JavaScript or C# or User-defined aggregates. While in spark it uses Scala Python R Java

The input data from in-stream analytics Avro, JSON or CSV, UTF-8 encoded, while spark is taking any format using custom code

References:

- [X]Microsoft Docs, https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/data/stream-processing-stream-analytics , Stream processing with Azure Stream Analytics
- [X] Microsoft Docs, https://docs.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/stream-processing ,Choose a stream processing technology in Azure
- [X] Spark apache , https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html ,Structured Streaming Programming Guide

2. Describe briefly 3 benefits of Azure U-SQL over Apache Spark. Illustrate them briefly with some use cases

Although U-SQL is a combination between spark and SQL and it is not an ANSI SQL, It supports constructs and U-SQL has some advantages over spark:

- The first thing is that U-SQL makes the user write normal SQL queries and can be driven to C#. This helps to extend it to us to user-defined functions. C# is statically typed, and it is faster, and it has can do more than python and this is useful in the case of streaming. While spark use with Scala or python, while U-SQL can extend with c# and .Net it allows us to:

  - Add c# expressions inside SELECT.
  - Add user-defined functions, and aggregation,
  - Add user-defiend operators like (processors, extractors, outputters, reducers and combiners).

- U-SQL cost per query. but Apache spark costs keep going as long as the cluster is running by U-SQL we can save more money. As if we. have a large data it's highly recommended to get payed on the job you have done only.
- U-SQL could be used on the local machine in visual basic this helps to show the results before going to the cloud. This could be helpful to monitor the behavior of the query before going to the cloud if something went wrong then we can modify it.
- Uniformally process the data from different sources in Azure (like Azure data lake, Azure block storage, SQL DB, SQL Server instances, and ADLS), so it will be better to use U-SQL on the Azure environment.

References:

- Lecture notes
- [X] stack overflow, https://stackoverflow.com/questions/35575080/azure-spark-sql-vs-u-sql , Azure Spark SQL vs U-SQL

3. What are the 5 characteristics of Azure Data Lake that distinguish it from other Distributed Dataset Storage infrastructure such as Hadoop?

1. **Developer friendly platform**

Managing Hadoop applications is complex. Developers need to write code to each operation that makes it very difficult to work but Azure Data Lake makes this easy by using Azure DevOps, Visual Studio, etc…, so that make Developers use familiar tools to run, debug and trace the code. And see how it runs.

2. Security

Data in the cluster is not always segmented by applications. Hadoop clusters don’t support encrypted data at reset. But Azure Data Lake is Based on HDFS but provides security management & integration with other Azure services while abstracting Hadoop commands. So, it is managed and supported by Microsoft and governance controls.

3. Unlimited Storage

Azure Data Lake Storage can hold any amount of data and unlimited number of files. We can store single or multiple files at terabyte and petabyte. Because the files are divided into blocks. these blokes are distributed across data nodes. Which are unlimited. Other cloud storage options have file and account size limitations of only a few terabytes.

4. Real-time Data Processing

Apache Hadoop is for batch processing, which means it takes a huge amount of data in input, process it and produces the result. But Azure Data Lake has Apache Spark/Storm in HDInsight which processes streaming data for the Azure DL Storage.

5. the data into one place

ADLS can gathering all the big data from disparate sources across cloud and on-premises environments into one place. That make the monitoring and managing the stored data easy.

References:

[1]Lecture Notes , Week 8 - Azure Big Data Landscape & Week 4 - Batch Processing of Big Data

[2] dremio, https://www.dremio.com/data-lake/azure/ -What Is Azure Data Lake Storage (ADLS)?

[3]Dataflair, https://data-flair.training/blogs/13-limitations-of-hadoop/ -13 Big Limitations of Hadoop & Solution To Hadoop Drawbacks

4. What 2 factors influence the use of Azure AUs for U-SQL query processing. Describe with a simple example.

AU number of nodes to process the job the default AUs number on azure is 5 there is a tool called AU Analyzer that helps to perform optimizations. So, depending on the job we want to make you can choose the number of AUs. For example, if we process a huge amount of data then we need to add more AUs. and if we want to make simple things as creating a table then 1 AU could be enough. The two factors that influence choosing the number of AUs are cost and time. but more AUs can make the overhead due to parallelization

For example, if we need to query a huge amount of data in use this data in some real-time application so it is highly recommended to increase the number of AUs. And if we want to get some insight from the data lake (Time not important) we can choose a fewer number of AUs.

References:

- Lecture notes, recording
- [X] azure, https://azure.microsoft.com/en-au/blog/get-started-with-u-sql-it-s-easy/ ,Get started with U-SQL: It’s easy!