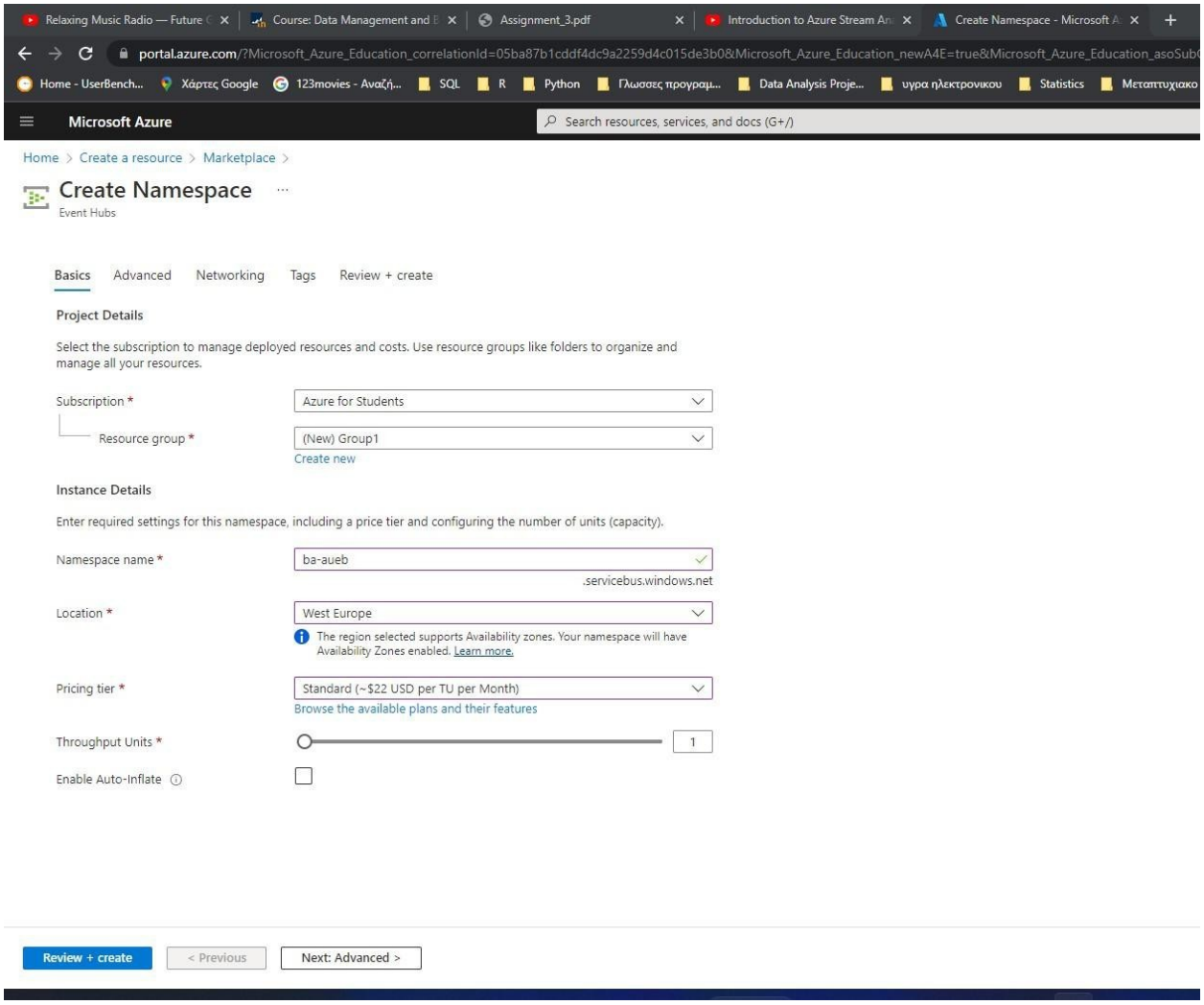
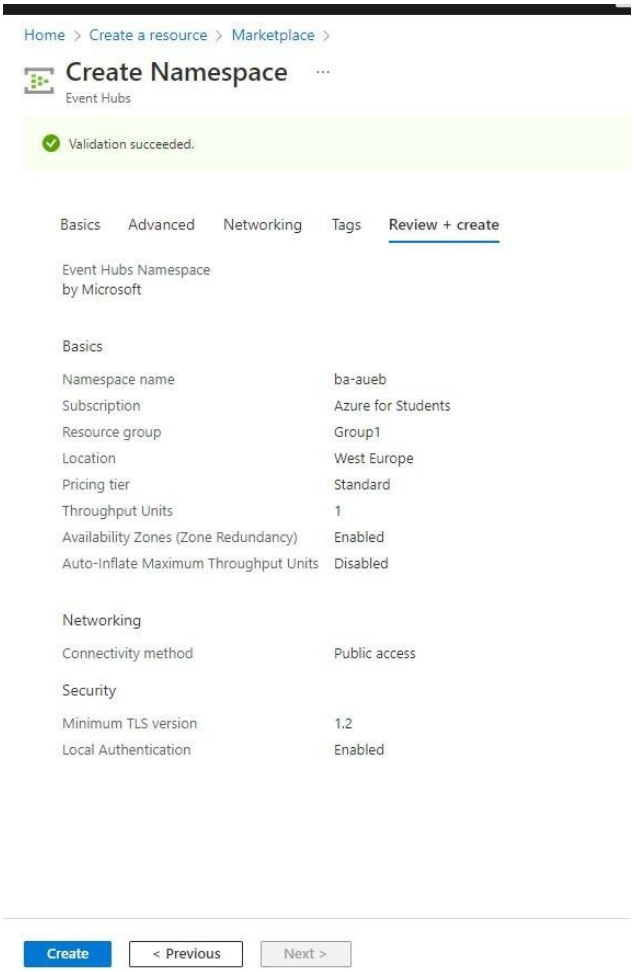
Assignment 3, Data Streaming with Azure.
Vlassis Christos

Part 1 Setting up the Azure:

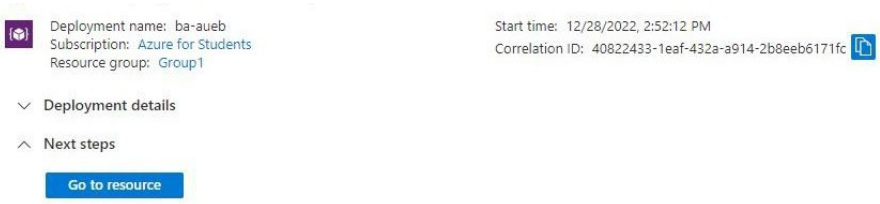
Firstly we create a NameSpace to have out event hub. We also create a resource group(group1) to be used for other purposes also:



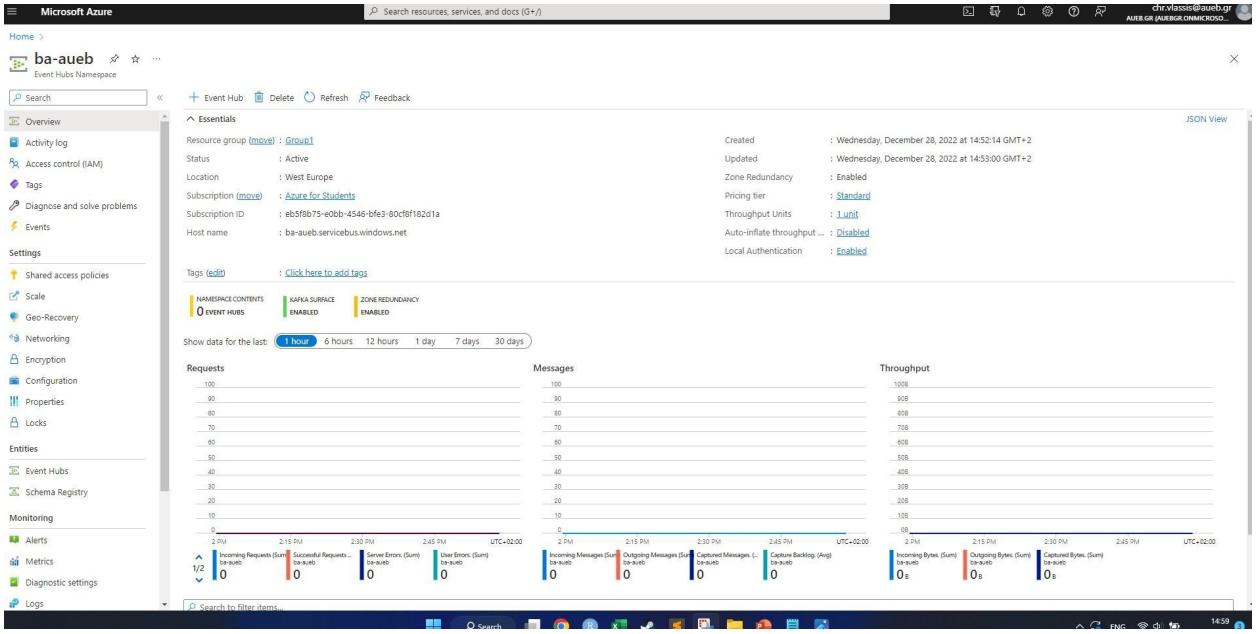
Operation Succeded:



Deployment succeeded:



View of the workspace:



Now, we have to create the Event hub in order to bring the data from the application:

Basics

Name	ba-aueb-stream-hub
Partition count	3

Retention

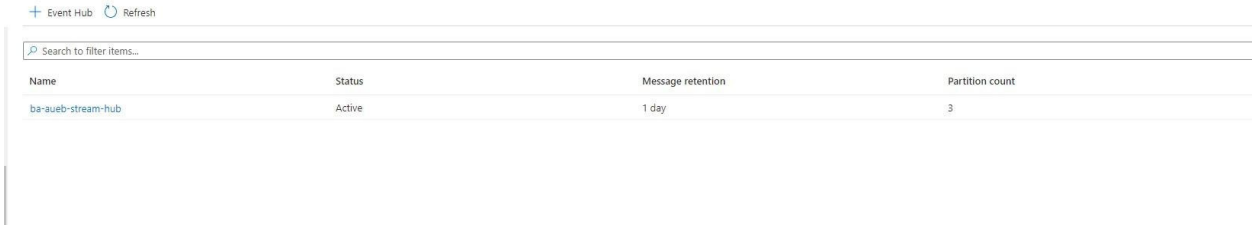
Cleanup policy	Delete
Retention time (hrs)	1

Capture

Capture Status	Off

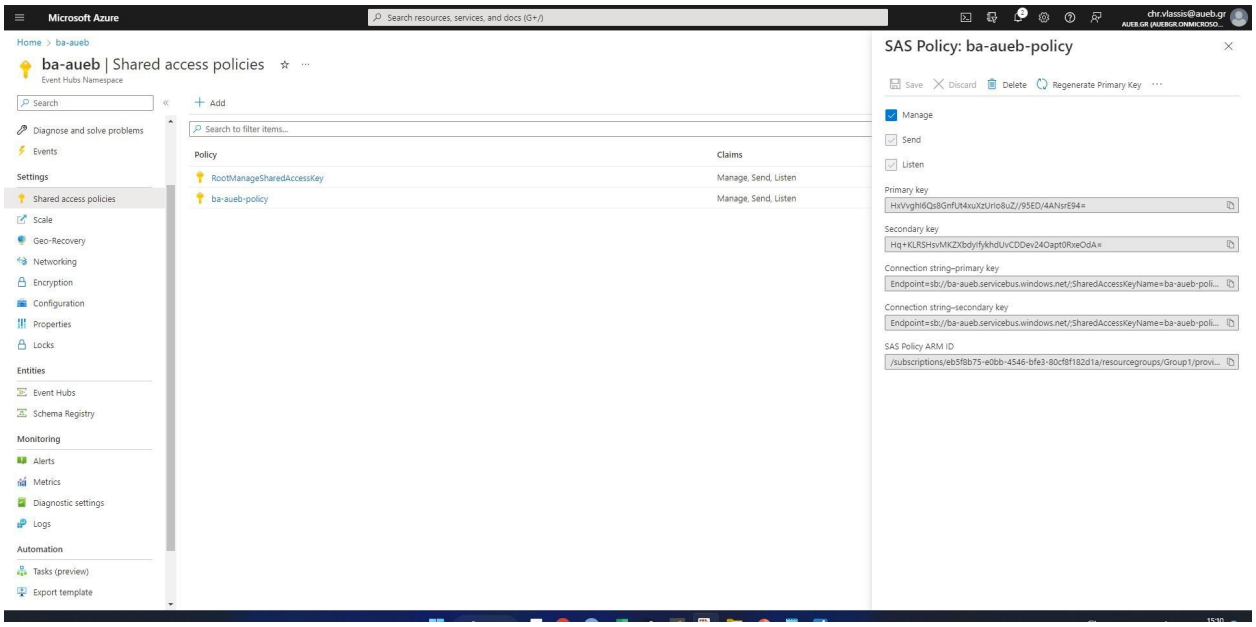
Create < Previous Next >

The event hub View:



Also, we have to create a policy:

Policy View:

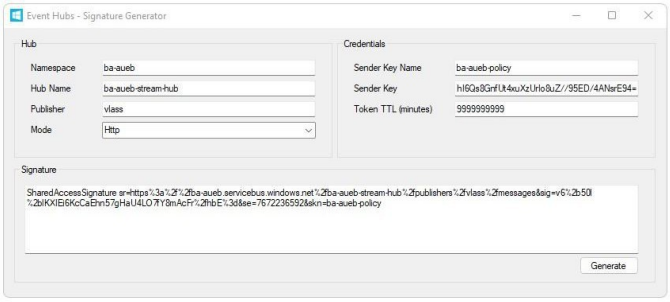


It's time to open the application using notepad and change the appropriate code to send data to our event hub. What follows is the line of code that was changed in order to achieve that.

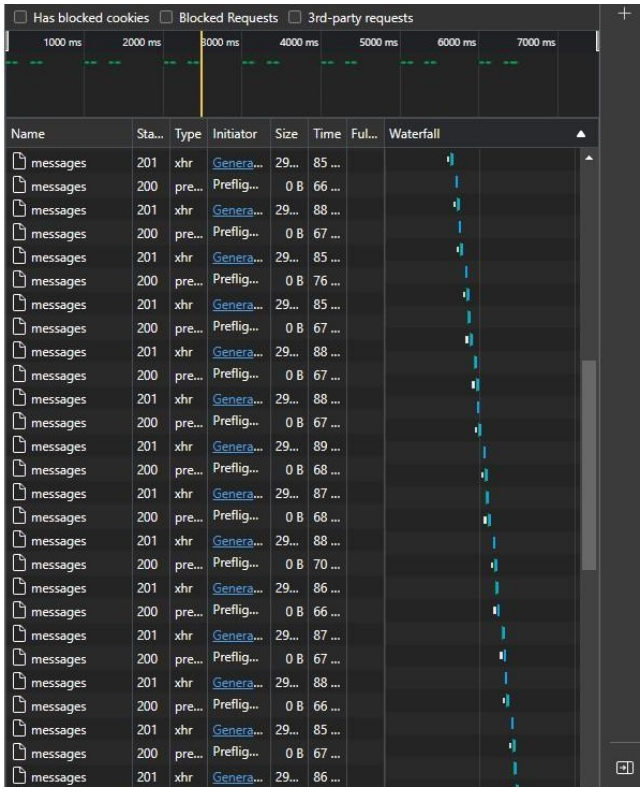
```
//Use the signature generator: https://github.com/sandrinodimattia/RedDog/releases
var sas = "SharedAccessSignature sr=https%3a%2f%2fba-aueb.servicebus.windows.net%2fba-aueb-stream-hub%2fpublishers%2fvlass%2fmessages&sig=v6%2b501%2blKXIEi6KcCaEhn57gHaU4LO7fY8mAcFr%2fhbE%3d&se=7672236592&skn=ba-aueb-policy";
var serviceNamespace = "ba-aueb";
var hubName = "ba-aueb-stream-hub";
var deviceName = "vlass";
```

Creating a signature generator:

Send Data Sent: { "ATMCode": 15 , "CardNumber": 3534633361736454 , "Type": 1 , "Amount": 10 }

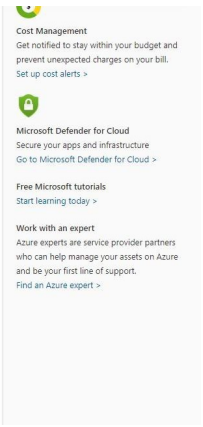


Application works fine:

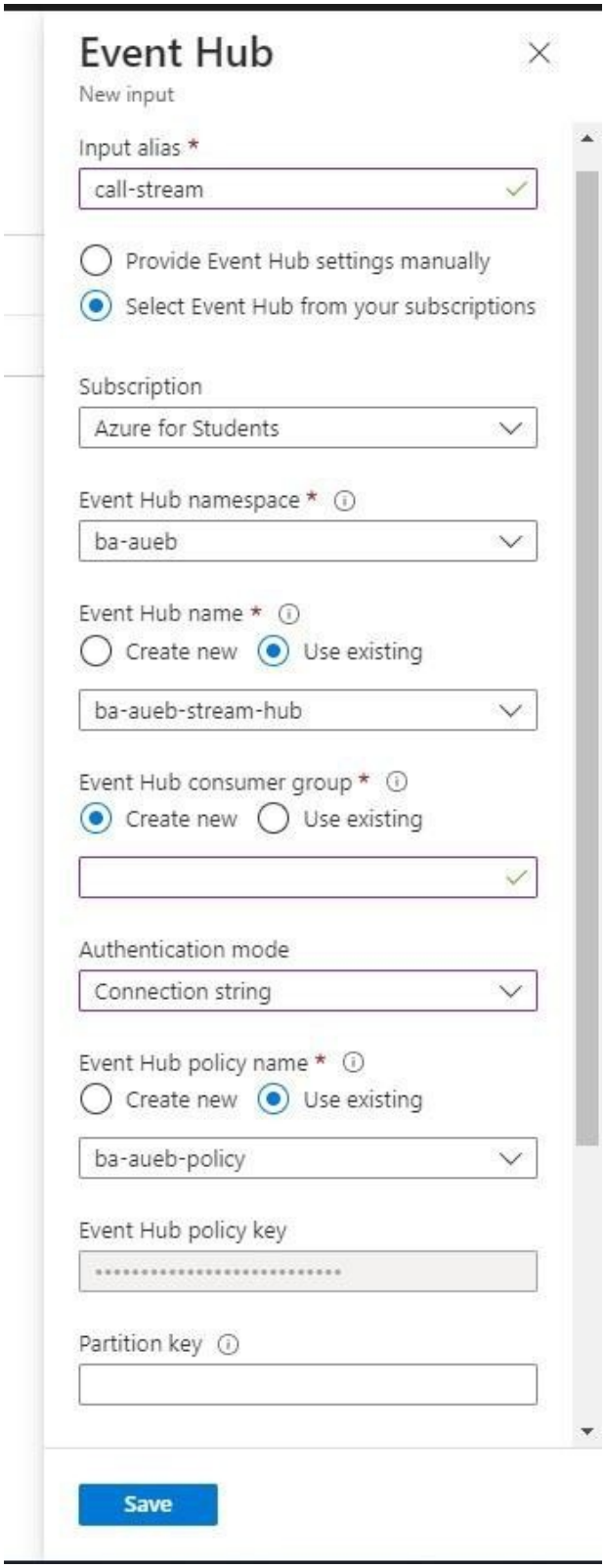


After the connecting the application to the event hub we have to create a storage an a stream analytics job:

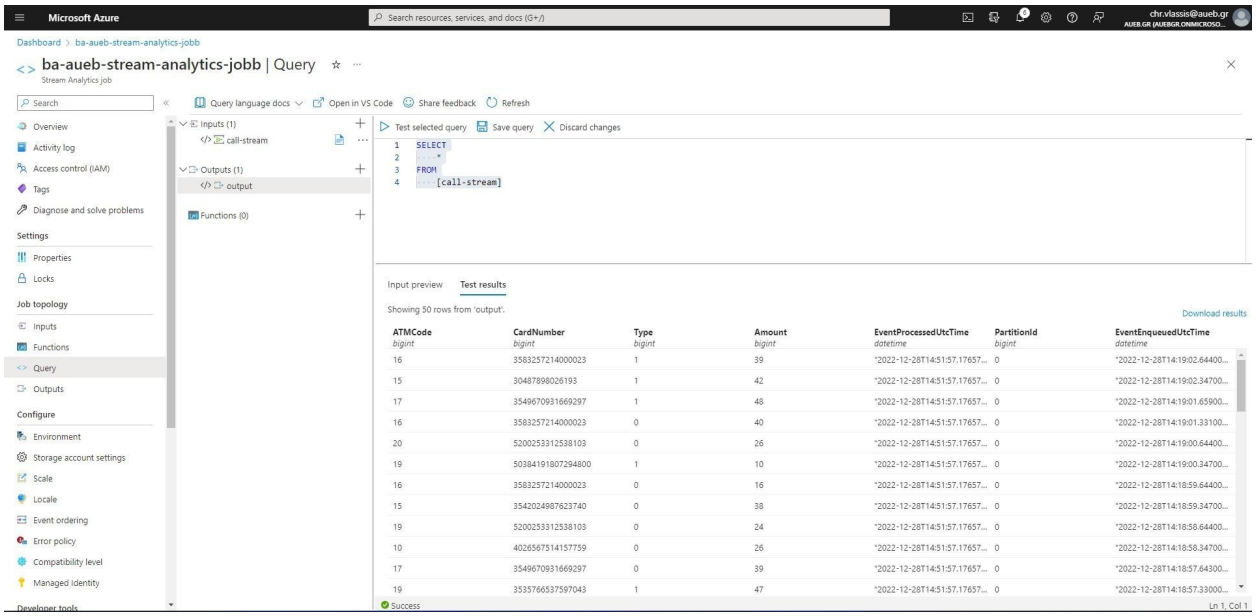
Creating the stream analytics job:



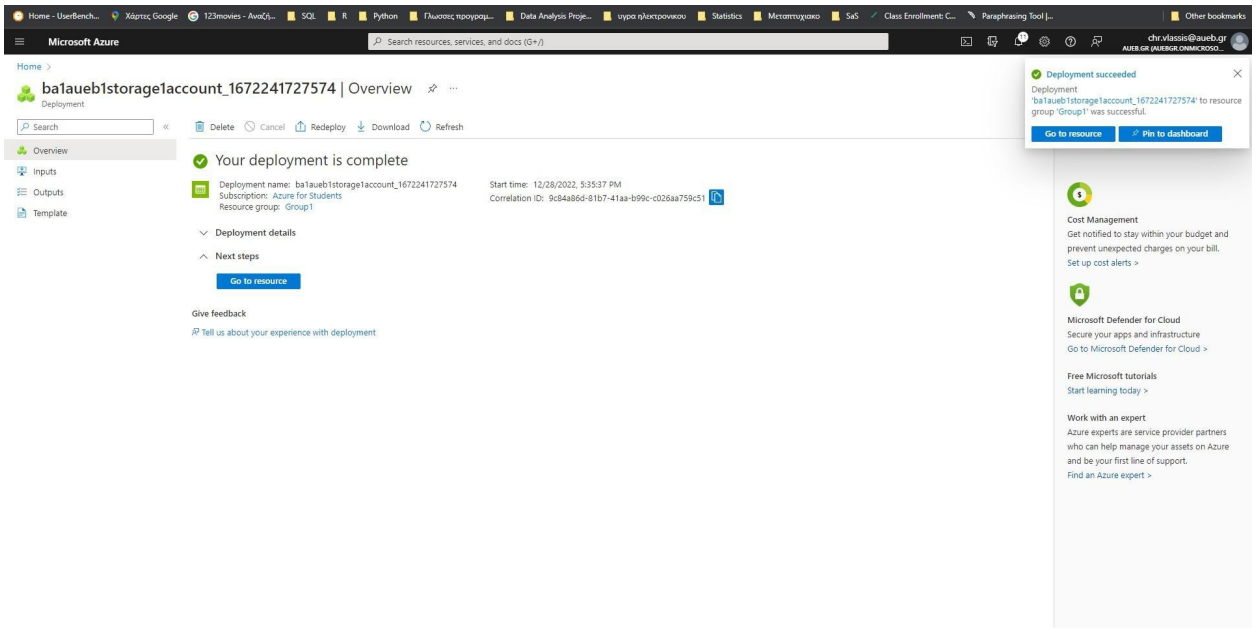
Creating the inputs of the stream:



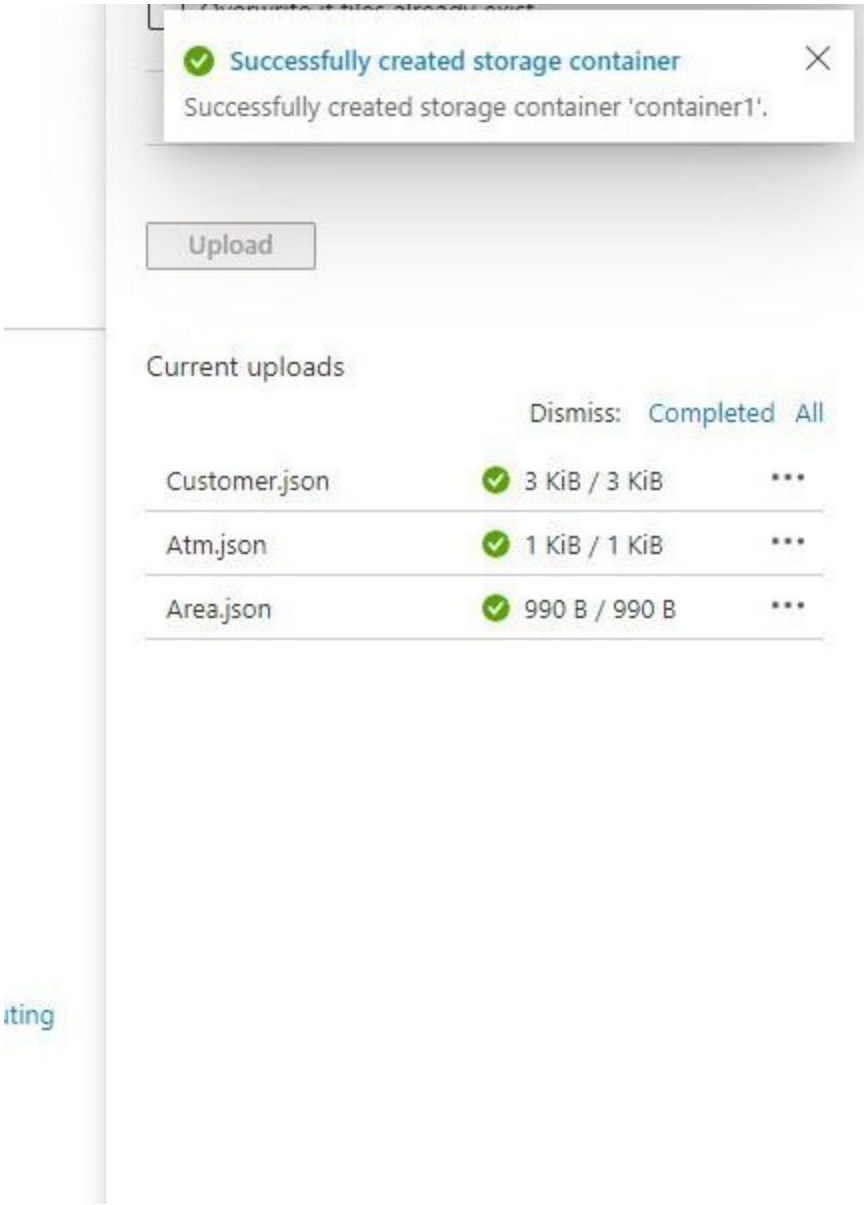
Testing the stream query:



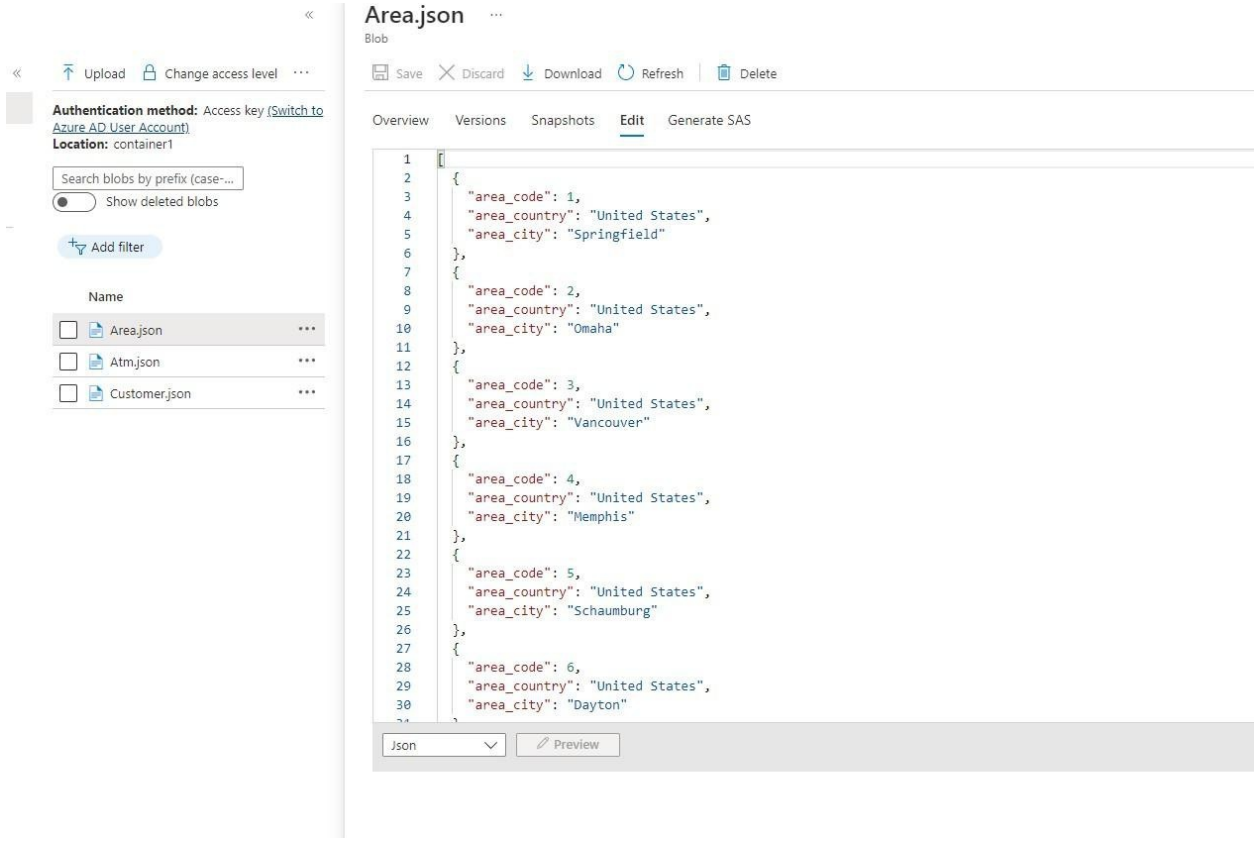
Our stream works fine. Now we have to create the storage account in order to store our output data. Creating the storage:



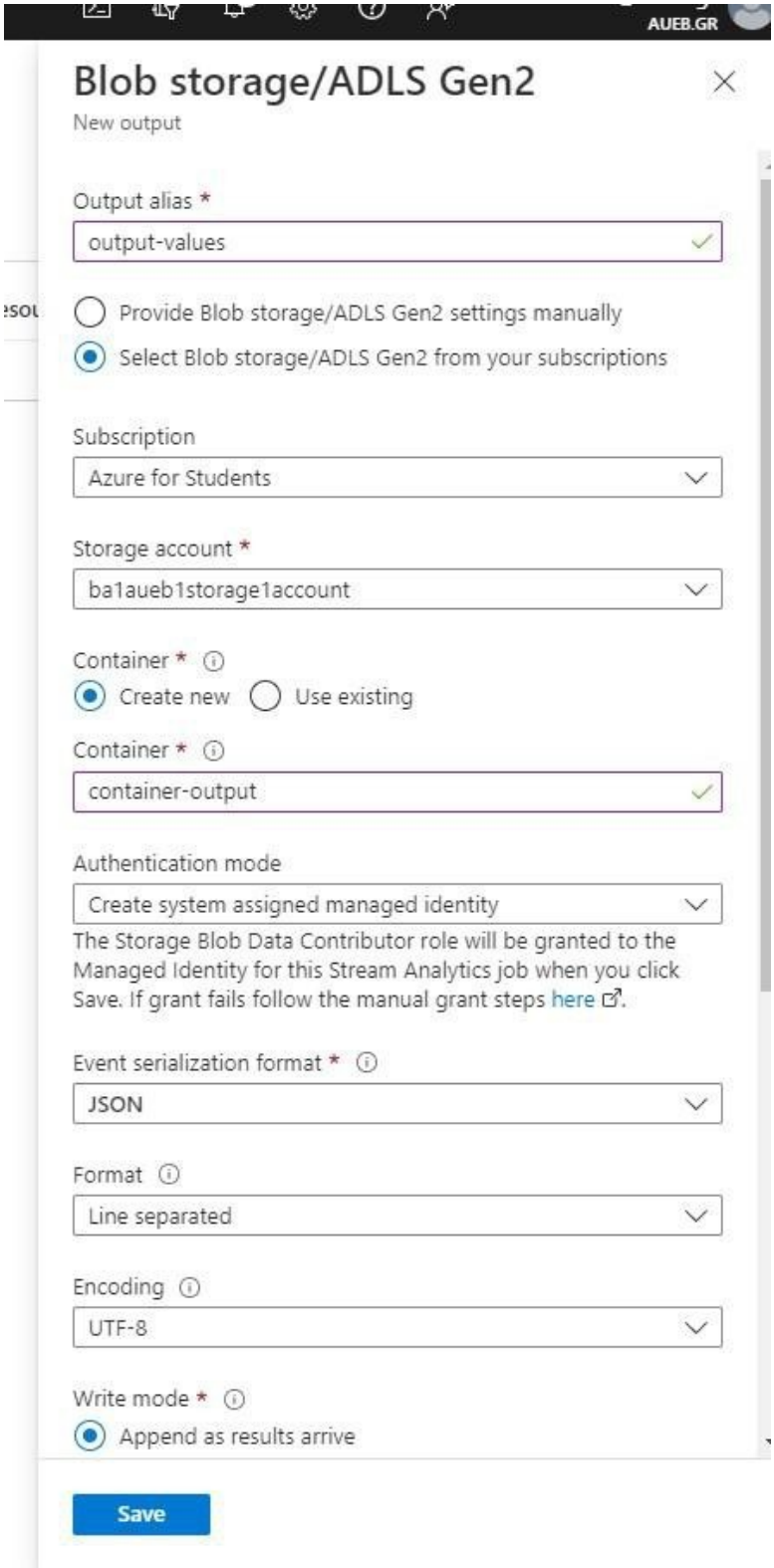
Now we want to upload our static data into the storage account. I will not put them in the same container as the output values, a different one will be used. Uploading the data:



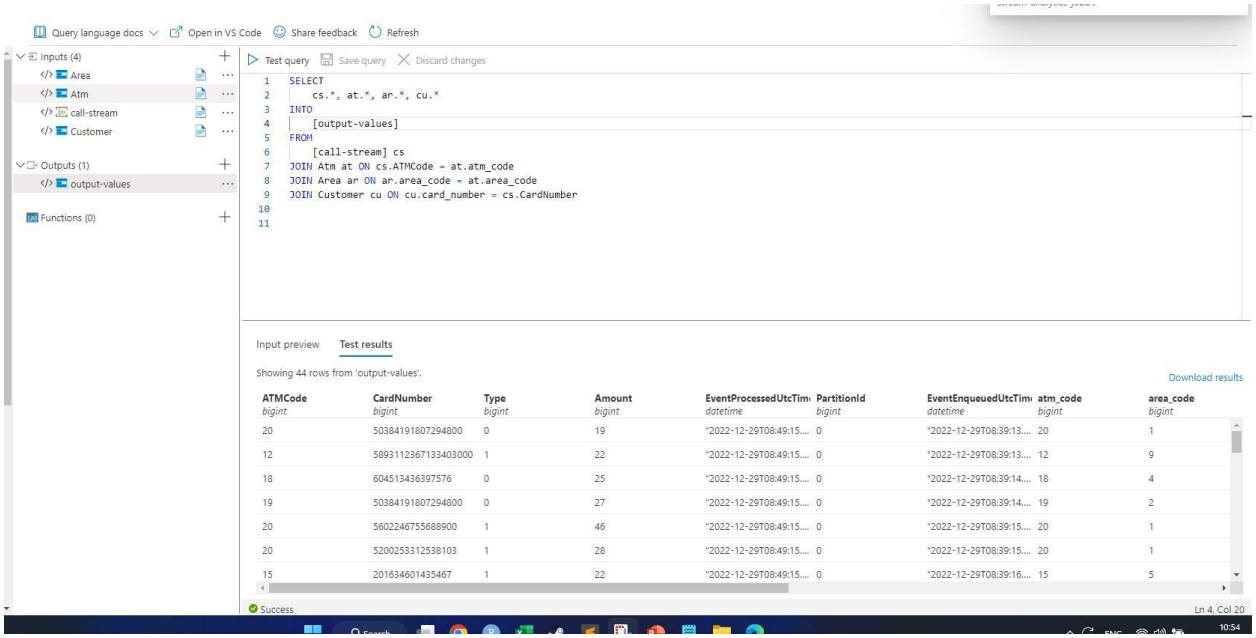
A snapshot of the static data:



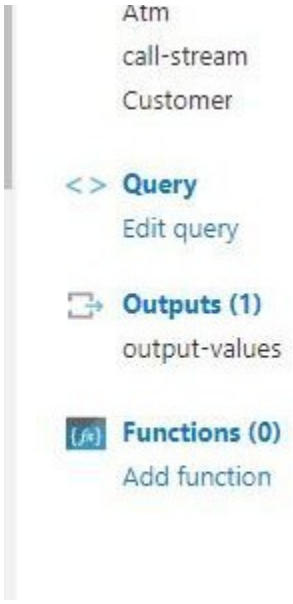
We create the output container of the stream analytics job:



Testing the combination of stream data (from the application level) with the static data (from our storage account). Everything works:



4 inputs and 1 output:

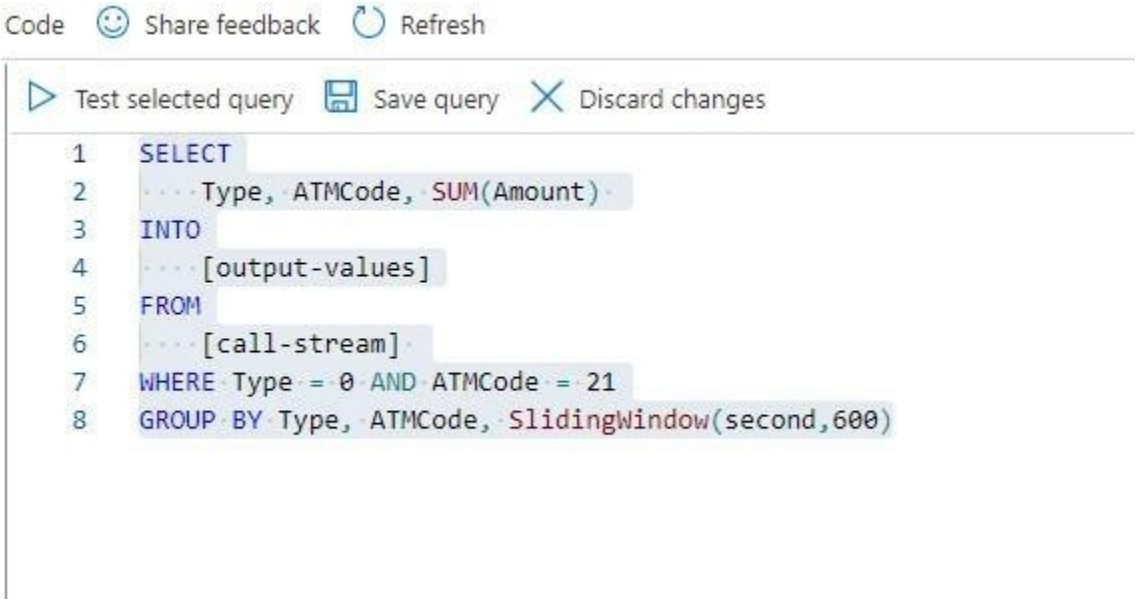


Output photos for the select *:

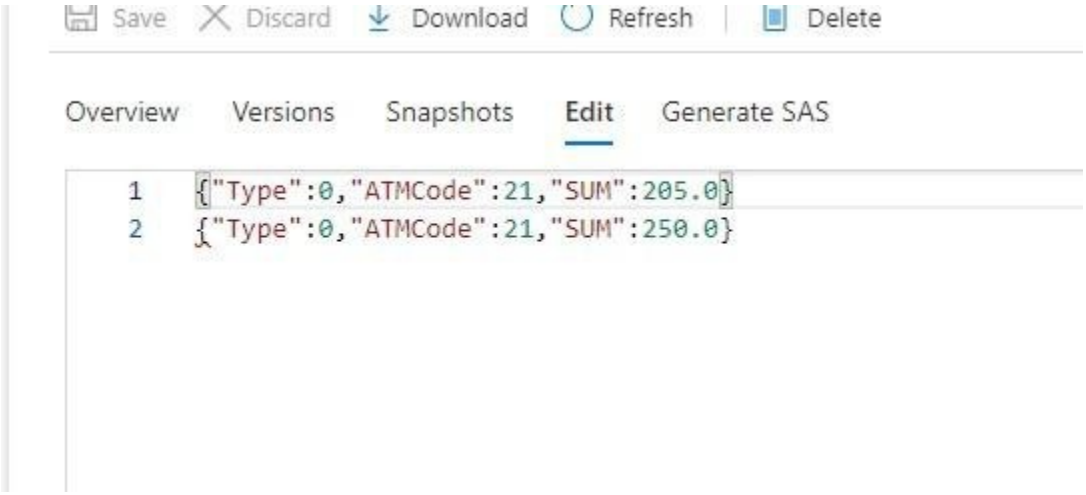
Part 2 Analysis and Queries:

Now that we have tested and we are certain that our stream works with the static data and we storage our output we can start our analysis:

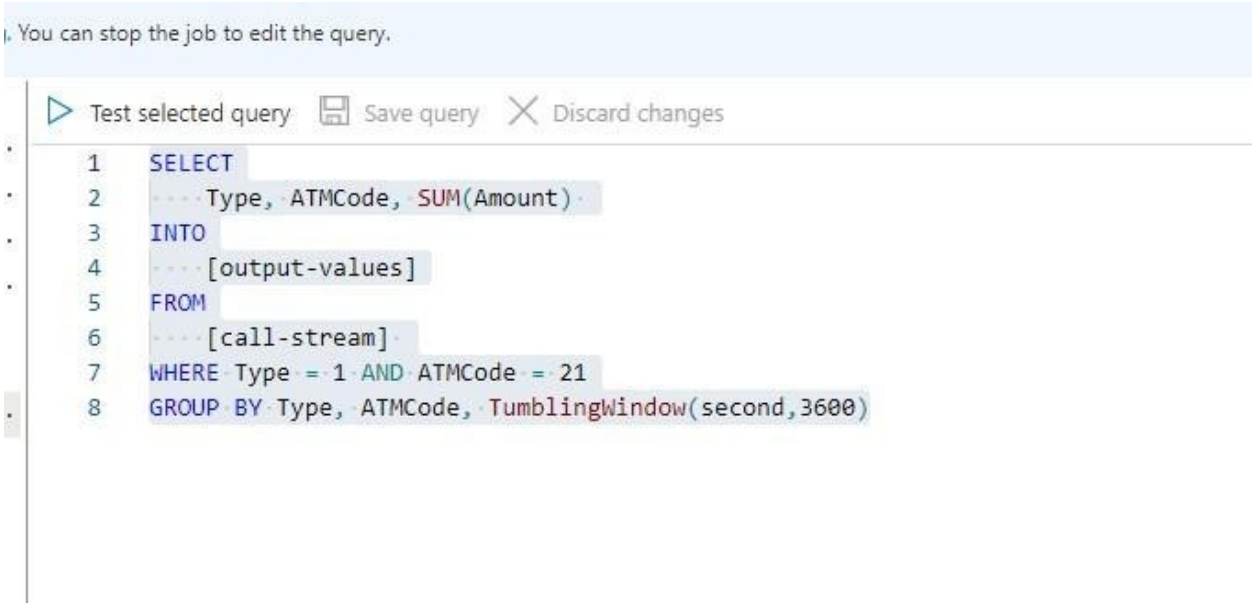
Query 1: Show the total “Amount” of “Type = 0” transactions at “ATM Code = 21” of the last 10 minutes. Repeat as new events keep flowing in (use a sliding window).



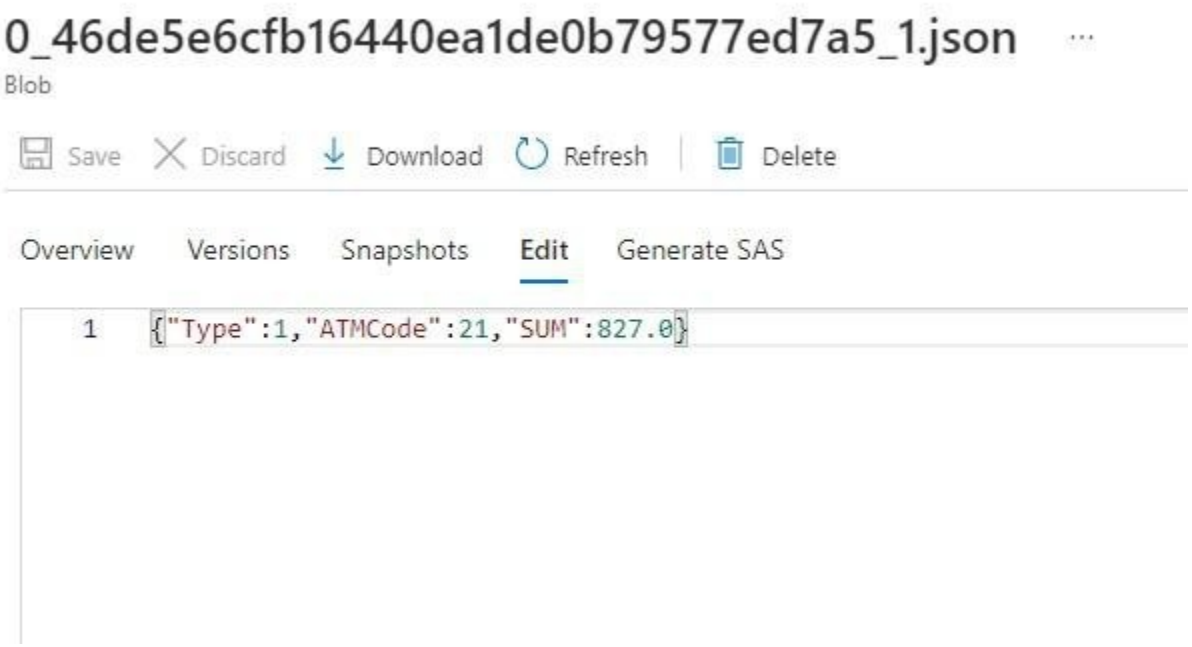
Output:



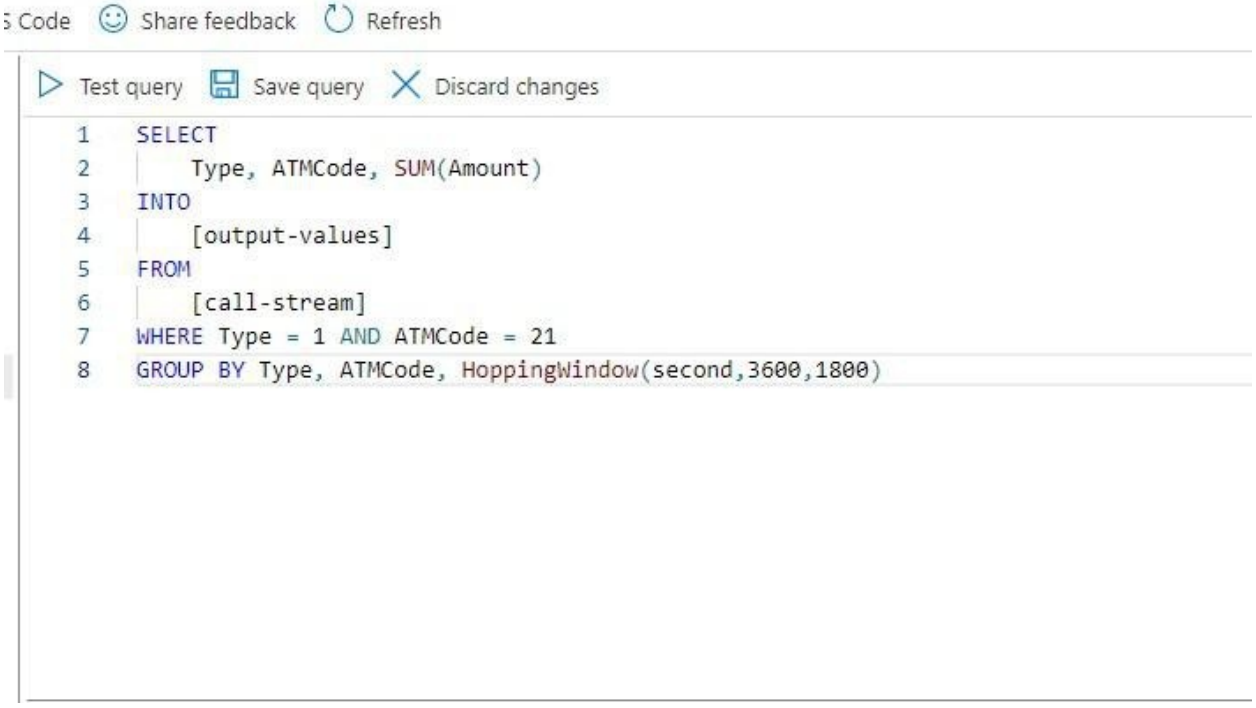
Query 2: Show the total “Amount” of “Type = 1” transactions at “ATM Code = 21” of the last hour. Repeat once every hour (use a tumbling window).



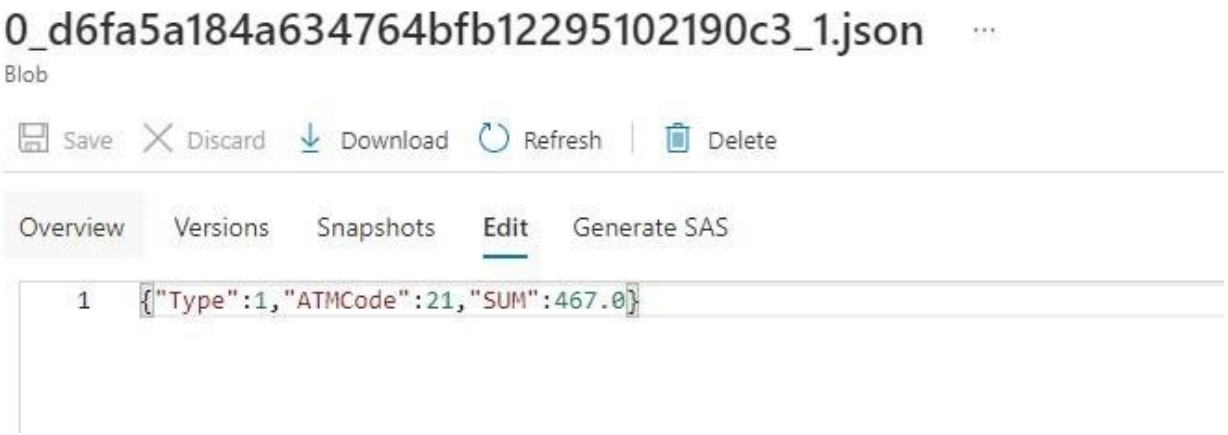
output:



Query 3: Show the total “Amount” of “Type = 1” transactions at “ATM Code = 21” of the last hour. Repeat once every 30 minutes (use a hopping window).



output:



Query 4: Show the total “Amount” of “Type = 1” transactions per “ATM Code” of the last one hour (use a sliding window).

Code Share feedback Refresh

Test selected query Save query Discard changes

```
SELECT
    Type, ATMCode, SUM(Amount)
INTO
    [output-values]
FROM
    [call-stream]
WHERE Type = 1
GROUP BY Type, ATMCode, SlidingWindow(second,3600)
```

Output:

0_bf85f262dd364d56ab0649079809812c_1.json

Blob

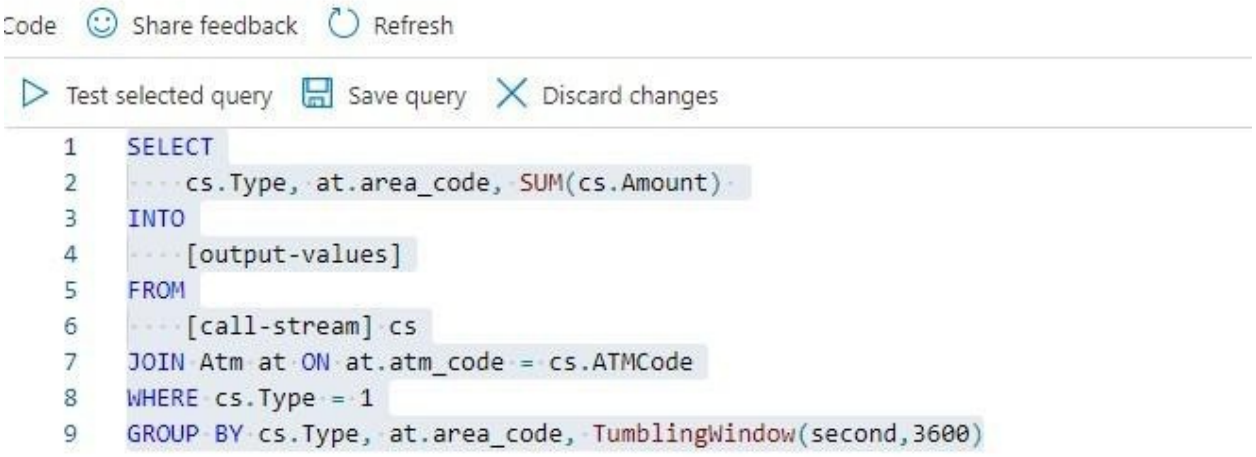
Save Discard Download Refresh Delete

Overview Versions Snapshots Edit Generate SAS


```
{"Type":1,"ATMCode":13,"SUM":599.0}
{"Type":1,"ATMCode":15,"SUM":1639.0}
{"Type":1,"ATMCode":15,"SUM":1665.0}
{"Type":1,"ATMCode":18,"SUM":1219.0}
{"Type":1,"ATMCode":10,"SUM":1154.0}
{"Type":1,"ATMCode":13,"SUM":619.0}
{"Type":1,"ATMCode":10,"SUM":1189.0}
{"Type":1,"ATMCode":13,"SUM":632.0}
{"Type":1,"ATMCode":15,"SUM":1676.0}
{"Type":1,"ATMCode":10,"SUM":1216.0}
{"Type":1,"ATMCode":15,"SUM":1702.0}
{"Type":1,"ATMCode":18,"SUM":1245.0}
{"Type":1,"ATMCode":10,"SUM":1253.0}
{"Type":1,"ATMCode":12,"SUM":584.0}
{"Type":1,"ATMCode":18,"SUM":1256.0}
{"Type":1,"ATMCode":18,"SUM":1288.0}
{"Type":1,"ATMCode":10,"SUM":1286.0}
{"Type":1,"ATMCode":13,"SUM":666.0}
{"Type":1,"ATMCode":18,"SUM":1337.0}
{"Type":1,"ATMCode":18,"SUM":1368.0}
{"Type":1,"ATMCode":15,"SUM":1750.0}
{"Type":1,"ATMCode":15,"SUM":1780.0}
{"Type":1,"ATMCode":10,"SUM":1301.0}
{"Type":1,"ATMCode":10,"SUM":1315.0}
{"Type":1,"ATMCode":19,"SUM":1051.0}
{"Type":1,"ATMCode":10,"SUM":1348.0}
{"Type":1,"ATMCode":10,"SUM":1394.0}
{"Type":1,"ATMCode":18,"SUM":1402.0}
{"Type":1,"ATMCode":20,"SUM":1357.0}
{"Type":1,"ATMCode":13,"SUM":627.0}
```


Json Preview

Query 5: Show the total “Amount” of “Type = 1” transactions per “Area Code” of the last hour. Repeat once every hour (use a tumbling window).



Output:

Query 6: Show the total “Amount” per ATM’s “City” and Customer’s “Gender” of the last hour. Repeat once every hour (use a tumbling window).

Code | Share feedback | Refresh

Test query | Save query | Discard changes

```
SELECT
    ar.area_city, cs.ATMCode, cu.gender, SUM(cs.Amount)
INTO
    [output-values]
FROM
    [call-stream] cs
JOIN Atm at ON at.atm_code = cs.ATMCode
JOIN Area ar ON ar.area_code = at.area_code
JOIN Customer cu ON cu.card_number = cs.CardNumber
WHERE cs.Type = 1
GROUP BY cs.ATMCode, ar.area_city, cu.gender,TumblingWindow(second,3600)
```

Output:

ntainers > container-output >

0_f82589d7097d4a5eadad47b0a4df2175_1.json

Blob

Save | Discard | Download | Refresh | Delete

Overview | Versions | Snapshots | Edit | Generate SAS


```
{"area_city":"Baltimore","ATMCode":13,"gender":"Male","SUM":282.0}
{"area_city":"Memphis","ATMCode":18,"gender":"Male","SUM":592.0}
{"area_city":"Memphis","ATMCode":18,"gender":"Female","SUM":465.0}
{"area_city":"Tacoma","ATMCode":16,"gender":"Male","SUM":40.0}
{"area_city":"Schaumburg","ATMCode":15,"gender":"Female","SUM":1563.0}
{"area_city":"Omaha","ATMCode":19,"gender":"Male","SUM":1108.0}
{"area_city":"Canton","ATMCode":12,"gender":"Female","SUM":29.0}
{"area_city":"Springfield","ATMCode":20,"gender":"Female","SUM":284.0}
{"area_city":"Canton","ATMCode":12,"gender":"Male","SUM":678.0}
{"area_city":"Baltimore","ATMCode":13,"gender":"Female","SUM":292.0}
{"area_city":"Vancouver","ATMCode":17,"gender":"Male","SUM":603.0}
{"area_city":"Springfield","ATMCode":21,"gender":"Female","SUM":215.0}
{"area_city":"Tacoma","ATMCode":16,"gender":"Female","SUM":546.0}
{"area_city":"Springfield","ATMCode":20,"gender":"Male","SUM":746.0}
{"area_city":"Schaumburg","ATMCode":15,"gender":"Male","SUM":406.0}
{"area_city":"Omaha","ATMCode":19,"gender":"Female","SUM":377.0}
{"area_city":"Springfield","ATMCode":21,"gender":"Male","SUM":250.0}
```

Query 7: Alert (Do a simple SELECT “1”) if a Customer has performed two transactions of “Type = 1” in a window of an hour (use a sliding window).

Output:


```
{"CardNumber":3542024987623740,"Alerter":"1"}
{"CardNumber":3542024987623740,"Alerter":"1"}
{"CardNumber":56022176913710210,"Alerter":"1"}
{"CardNumber":4026567514157759,"Alerter":"1"}
{"CardNumber":201634601435467,"Alerter":"1"}
{"CardNumber":3535766537597043,"Alerter":"1"}
{"CardNumber":5446210381593272,"Alerter":"1"}
{"CardNumber":30487898026193,"Alerter":"1"}
{"CardNumber":50384191807294800,"Alerter":"1"}
{"CardNumber":5602246755688900,"Alerter":"1"}
{"CardNumber":5893112367133403000,"Alerter":"1"}
{"CardNumber":560222217915598000,"Alerter":"1"}
{"CardNumber":3549670931669297,"Alerter":"1"}
{"CardNumber":3534633361736454,"Alerter":"1"}
{"CardNumber":3583257214000023,"Alerter":"1"}
{"CardNumber":604513436397576,"Alerter":"1"}
{"CardNumber":3554025590595485,"Alerter":"1"}
{"CardNumber":50383945269330136,"Alerter":"1"}
{"CardNumber":5602238863017460,"Alerter":"1"}
{"CardNumber":5610827137784218,"Alerter":"1"}
{"CardNumber":5200253312538103,"Alerter":"1"}
```

Query 8: Alert (Do a simple SELECT “1”) if the “Area Code” of the ATM of the transaction is not the same as the “Area Code” of the “Card Number” (Customer’s Area Code) - (use a sliding window)

Test query | Save query | Discard changes

```
WITH Alert AS (
SELECT
        SUM(CASE
            WHEN a.area_code != c.area_code THEN 1 ELSE 0 END) AS equality
FROM
    [call-stream] cs
JOIN [Atm] a ON a.atm_code = cs.ATMCode
JOIN [Customer] c ON cs.CardNumber = c.card_number
GROUP BY SlidingWindow(hour,1) )
/* επειδή δεν αναφέρθηκε χρόνος για το sliding window έβαλα 1 hour*/

SELECT '1' as Alerter INTO [output-values] FROM Alert where equality IS NOT NULL
```

Input preview | Test results

Blob storage/ADLS Gen2 reference inputs cannot be previewed. Upload a sample file to test the query.

Table | Raw | Upload sample input | Download sample data

Output:

0_6904e0e84850492a9d48eb42f919a3b8_1.json

Blob

Save | Discard | Download | Refresh | Delete

Overview | Versions | Snapshots | Edit | Generate SAS

```
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
{"Alerter":"1"}
```