Big Data Systems – Data Streams

Assignment 5

Panourgia Evangelia (t8190130)
Papadatos Ioannis (t8190314)
Professor: Chatziantoniou Damianos

Latest Update: May 8, 2022

School of Management Science and Technology,
Athens University of Economics and Business

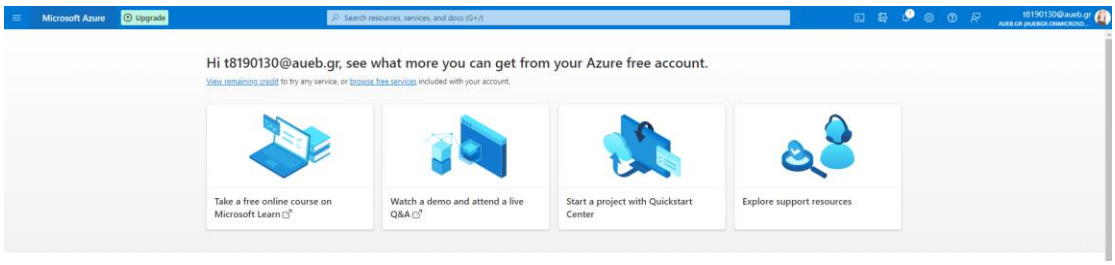
Contents

0.Introduction

Azure Stream Analytics is a fully managed, real-time analytics service designed to help you analyze and process fast moving streams of data that can be used to get insights, build reports or trigger alerts and actions [1]. For this assignment, we used Stream Analytics by setting up a processing environment and running queries using real-time randomly generated data and reference data (data exists in folder named data).

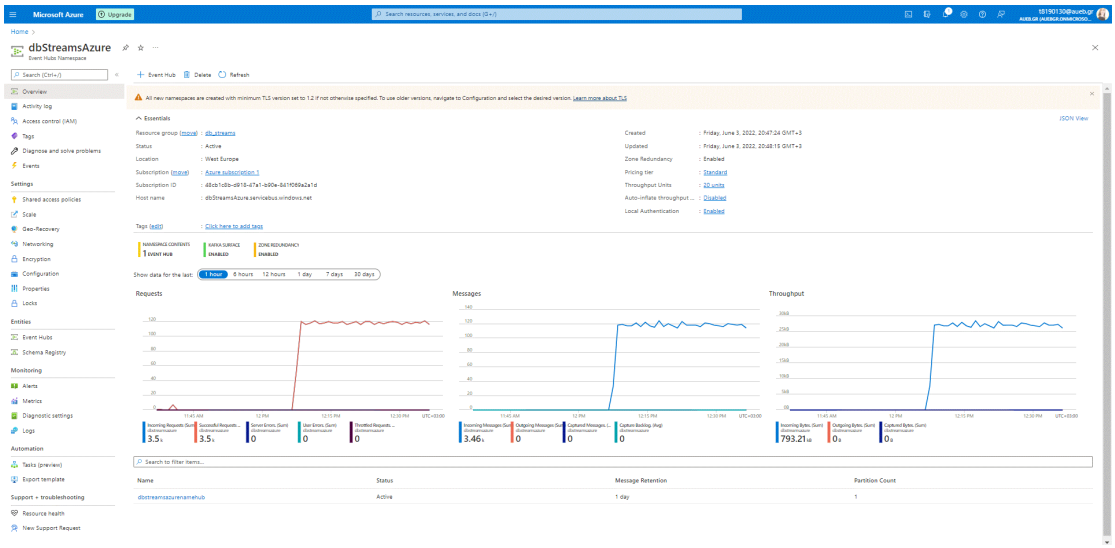
1. Create a Trial Account

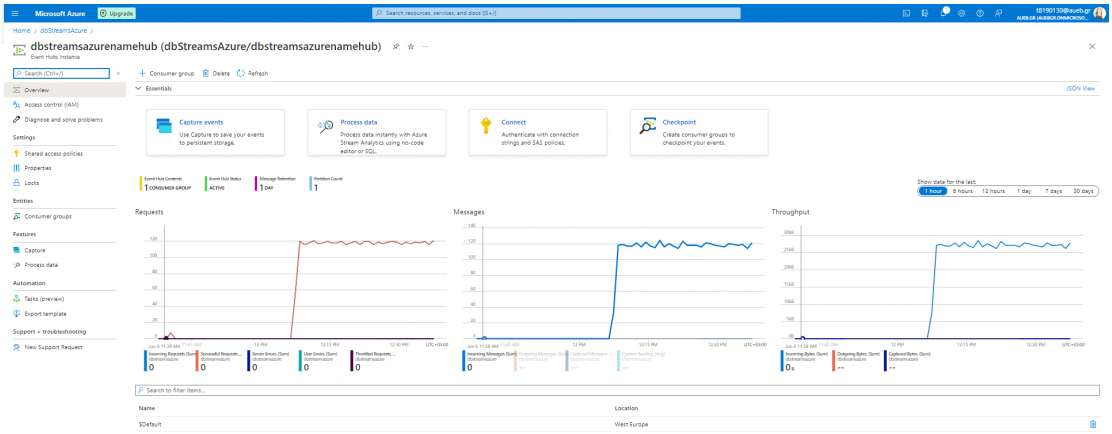
I created a Microsoft Azure account using my university e-mail address.



2. Setup an event hub

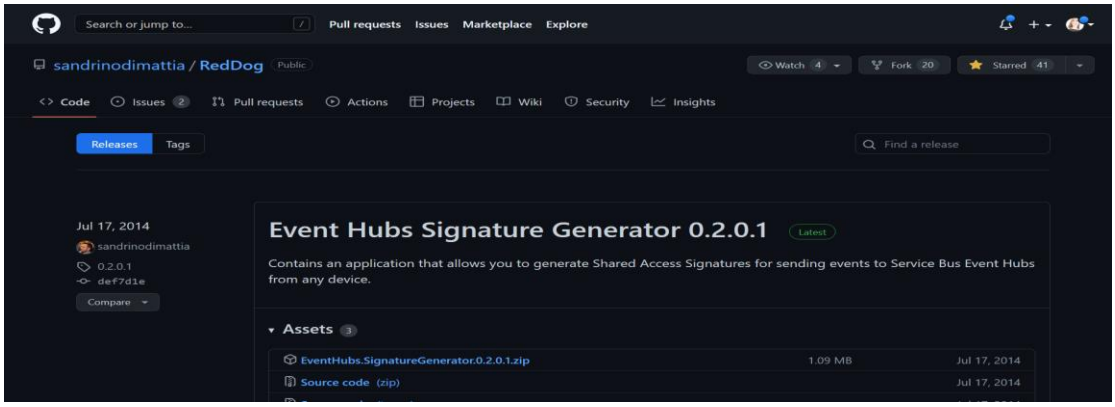
Event Hubs is a fully managed, real-time data ingestion service that's simple, trusted, and scalable [2].

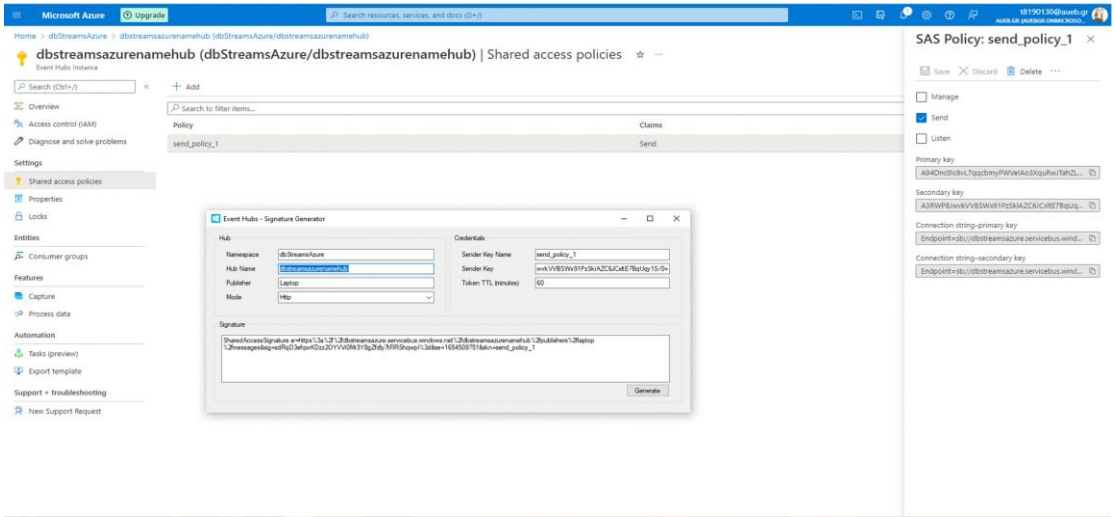




3. Generate a Security Access Signature

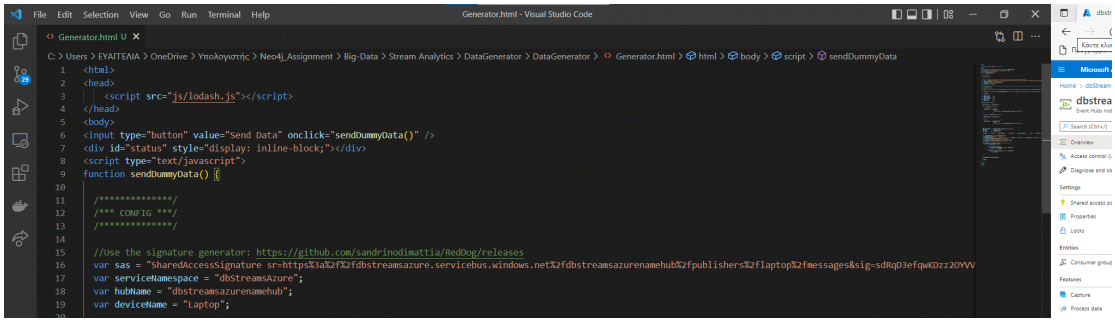
We generated a security access signature. Specifically, we visited GitHub repository named: https://github.com/sandrinodimattia/RedDog/releases which contains an application that allows you to generate Shared Access Signatures for sending events to Service Bus Event Hubs from any device. So, we download EventHubs.SignatureGenerator.0.2.0.1.zip (unzip it)and then we returned to azure interface again and created the security access signature as shows the following pictures.





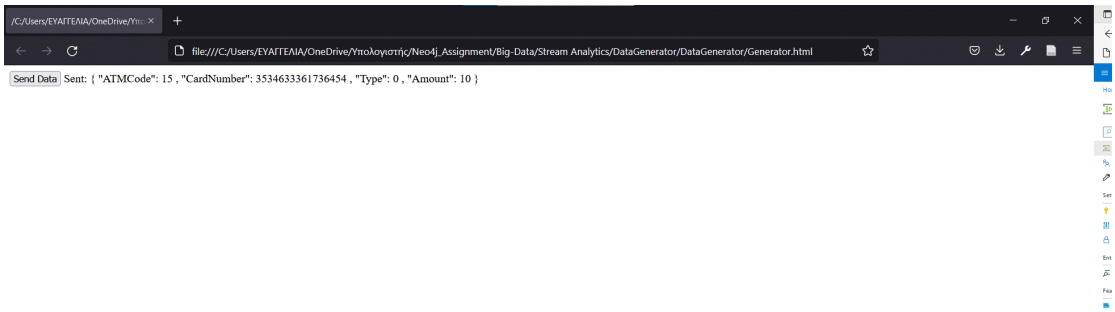
4. Edit Generator.html

We edited Generator.html. We opened it with visual studio and we updated the CONFIG variables with our security access signature bearing in mind the suitable variables values that we created in step 3 (the previous step).



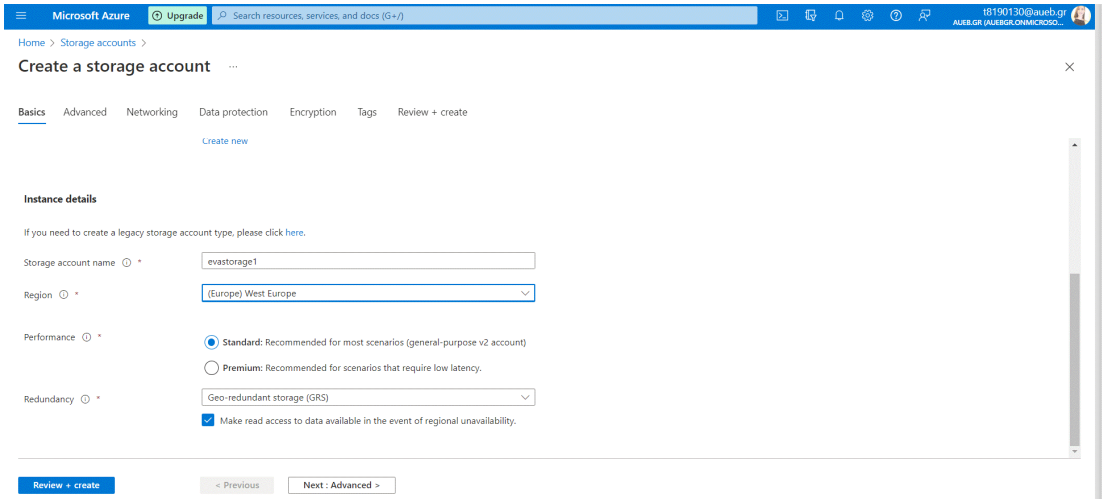
5. Feed the Event Hub

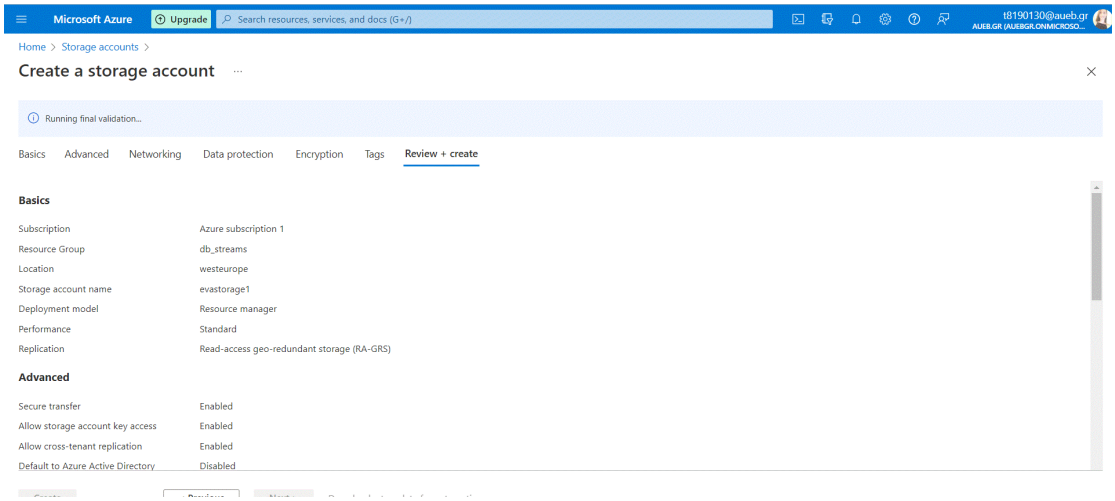
We fed the Event Hub with the use of Generator.html. Specifically, we opened Generator.html in a web browser and pressed the “Send Data” button.

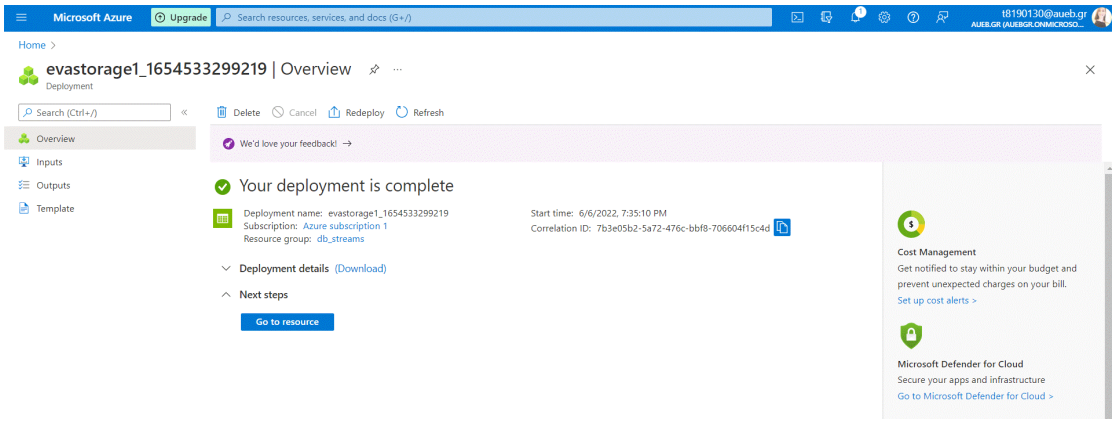


6. Set up a Storage account

An Azure storage account contains all of your Azure Storage data objects, including blobs, file shares, queues, tables, and disks. The storage account provides a unique namespace for your Azure Storage data that's accessible from anywhere in the world over HTTP or HTTPS. Data in your storage account is durable and highly available, secure, and massively scalable [3]. For, the assignment a storage account was created in order to store the reference data to be used in the queries, as well as their output.

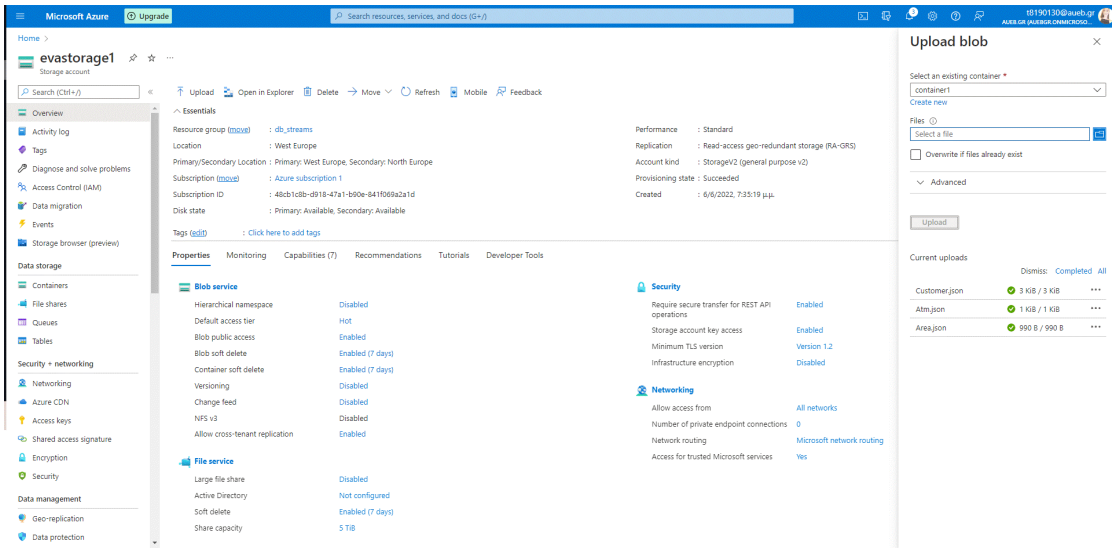

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
t8190130@aueb.gr
AUEB.GR (AUEBGR.ONMICROSO...
Home > Storage accounts >
Create a storage account
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Create new
Instance details
If you need to create a legacy storage account type, please click here.
Storage account name
evastorage1
Region
(Europe) West Europe
Performance
Standard: Recommended for most scenarios (general-purpose v2 account)
Premium: Recommended for scenarios that require low latency.
Redundancy
Geo-redundant storage (GRS)
Make read access to data available in the event of regional unavailability.
Review + create
< Previous
Next : Advanced >

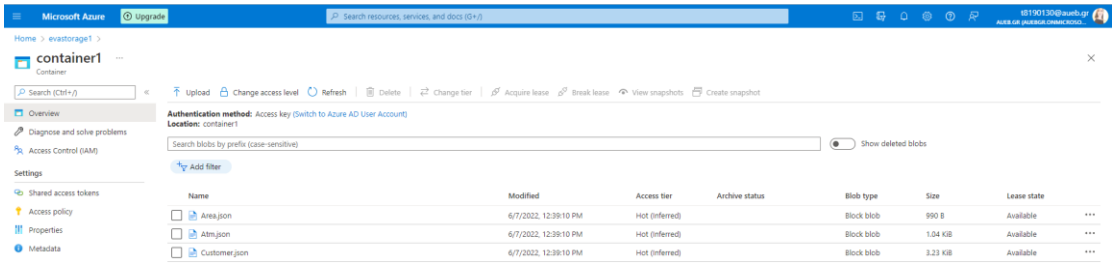

Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
t8190130@aueb.gr
AUEB.GR (AUEBGR.ONMICROSO...
Home > Storage accounts >
Create a storage account
Running final validation...
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Basics
Subscription
Azure subscription 1
Resource Group
db_streams
Location
westeurope
Storage account name
evastorage1
Deployment model
Resource manager
Performance
Standard
Replication
Read-access geo-redundant storage (RA-GRS)
Advanced
Secure transfer
Enabled
Allow storage account key access
Enabled
Allow cross-tenant replication
Enabled
Default to Azure Active Directory
Disabled


Microsoft Azure
Upgrade
Search resources, services, and docs (G+/)
t8190130@aueb.gr
AUEB.GR (AUEBGR.ONMICROSO...
Home >
evastorage1_1654533299219 | Overview
Deployment
Search (Ctrl+/)
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Refresh
We'd love your feedback!
Your deployment is complete
Deployment name: evastorage1_1654533299219
Subscription: Azure subscription 1
Resource group: db_streams
Start time: 6/6/2022, 7:35:10 PM
Correlation ID: 7b3e05b2-5a72-476c-bbf8-706604f15c4d
Deployment details (Download)
Next steps
Go to resource
Cost Management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >
Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

7. Upload the Reference Data files to your storage account

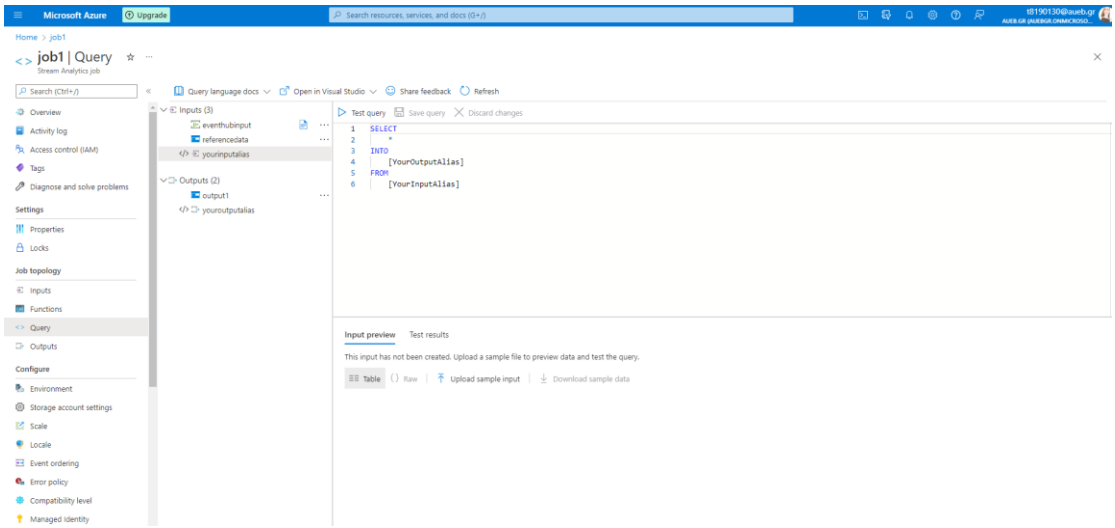
A blob container was created and the reference JSON files were uploaded.





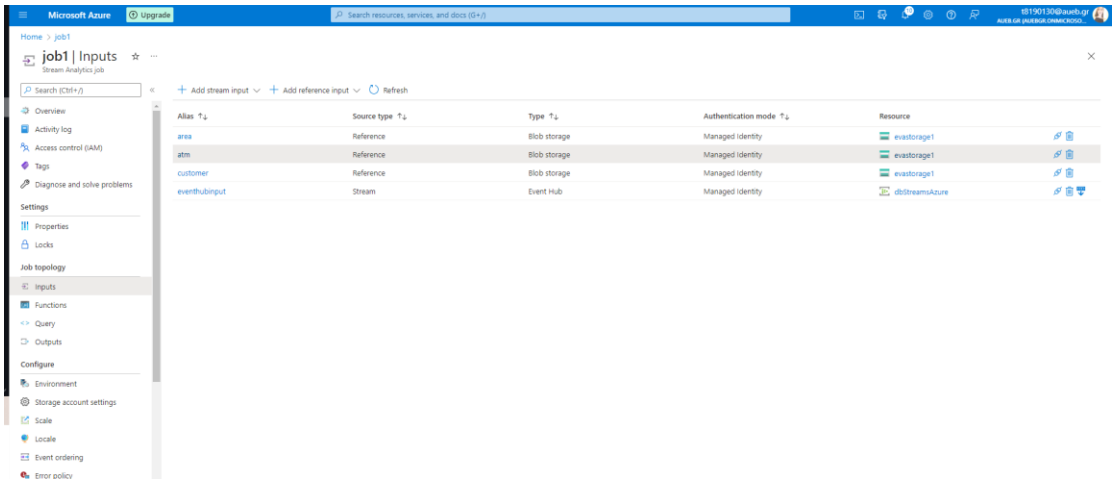
8. Setup a Stream Analytics Job

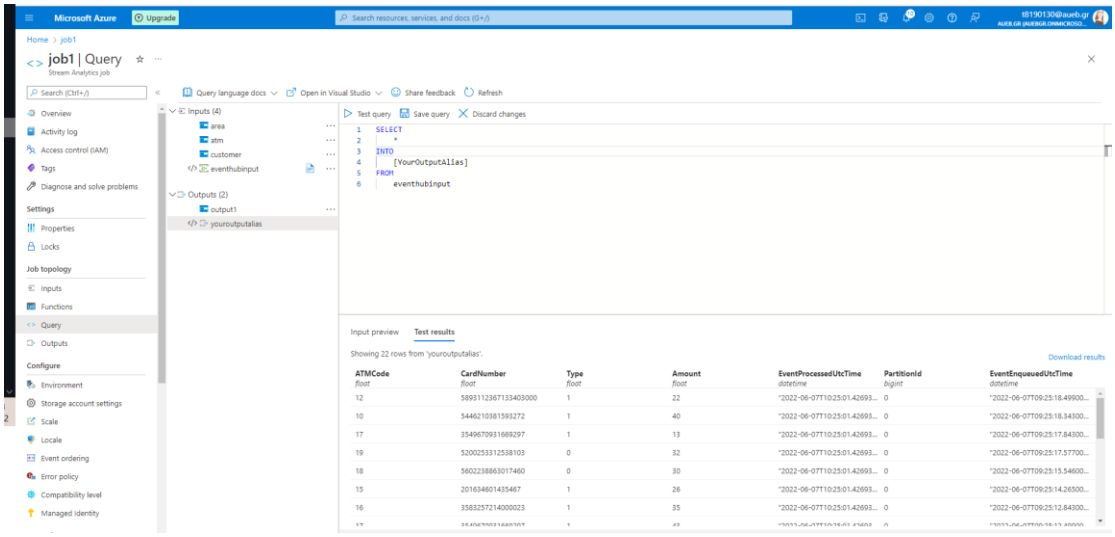
We created a stream analytics job bearing in mind the instructions from link[4].



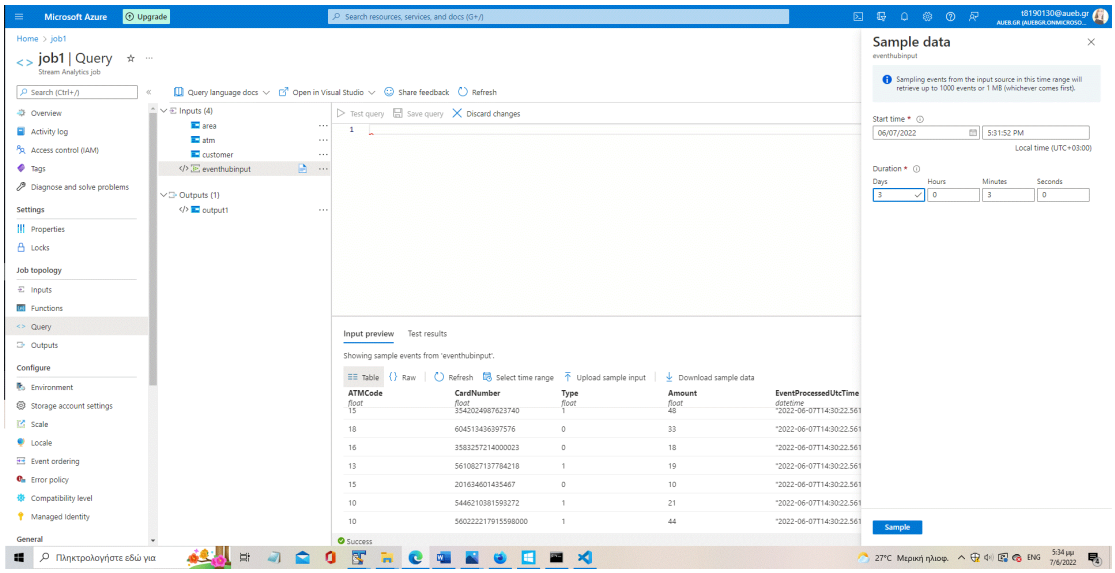
9. Input setup

The reference data, as well as the Event Hub stream were set as inputs for the job. Successful setup of the stream input was confirmed by sampling data from it.



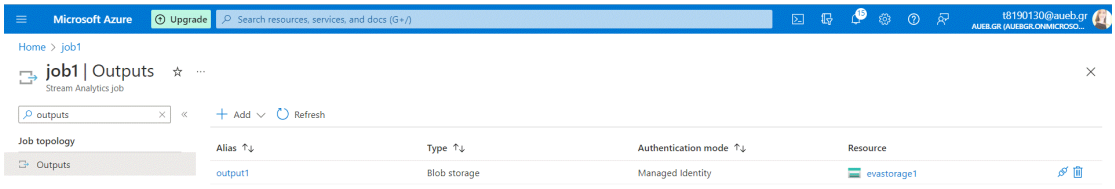


For the sample we changed the range as following :



10. Output setup

We created a stream analytics job bearing in mind the instructions from link[4].

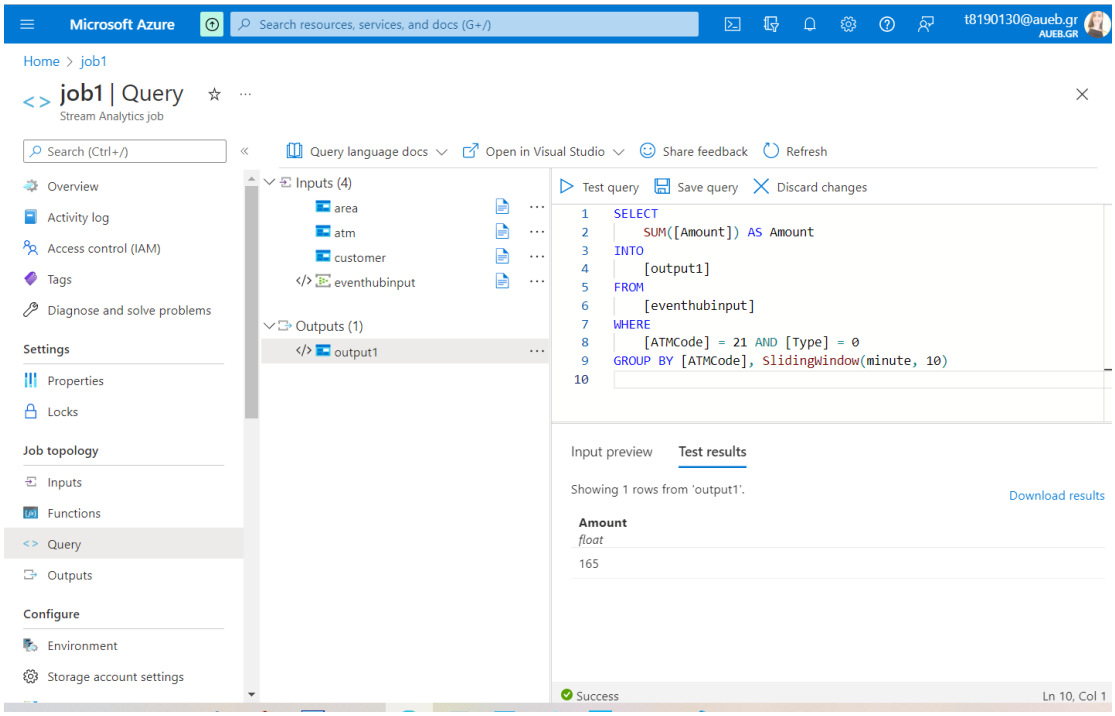


II Queries

In the interest of reproducibility the jobs were tested using the data sampled in Section 9 of this report, however jobs did also produce blobs in the output container, successfully. Both the sampled data and the results of the queries have been provided alongside this report in the folder **output_json**.

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).

```
SELECT
    SUM([Amount]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
WHERE
    [ATMCode] = 21 AND [Type] = 0
GROUP BY [ATMCode], SlidingWindow(minute, 10)
```

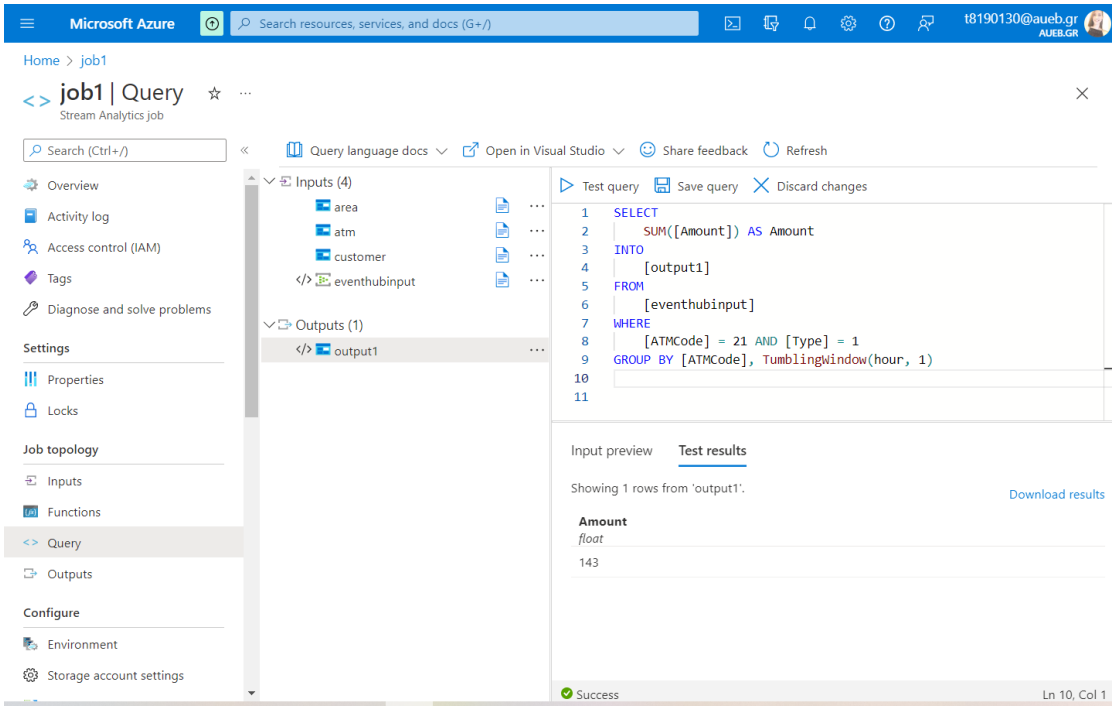


Reference [5]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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).

```
SELECT
    SUM([Amount]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
WHERE
    [ATMCode] = 21 AND [Type] = 1
GROUP BY [ATMCode], TumblingWindow(hour, 1)
```

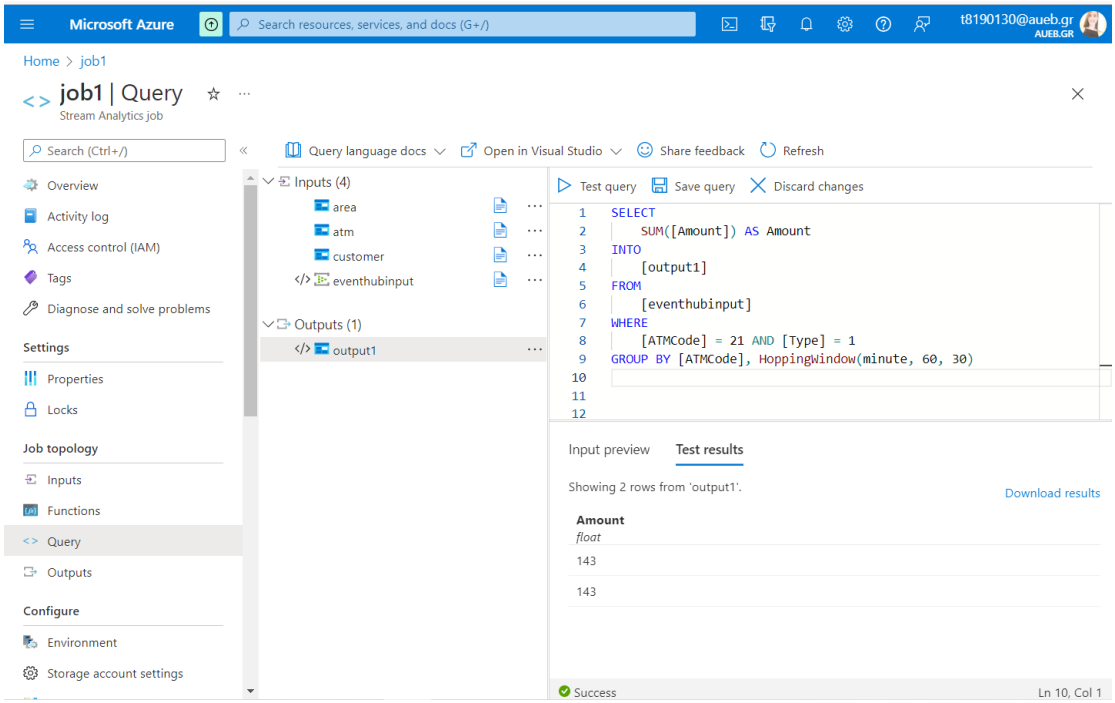


Reference [6]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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).

```
SELECT
    SUM([Amount]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
WHERE
    [ATMCode] = 21 AND [Type] = 1
GROUP BY [ATMCode], HoppingWindow(minute, 60, 30)
```

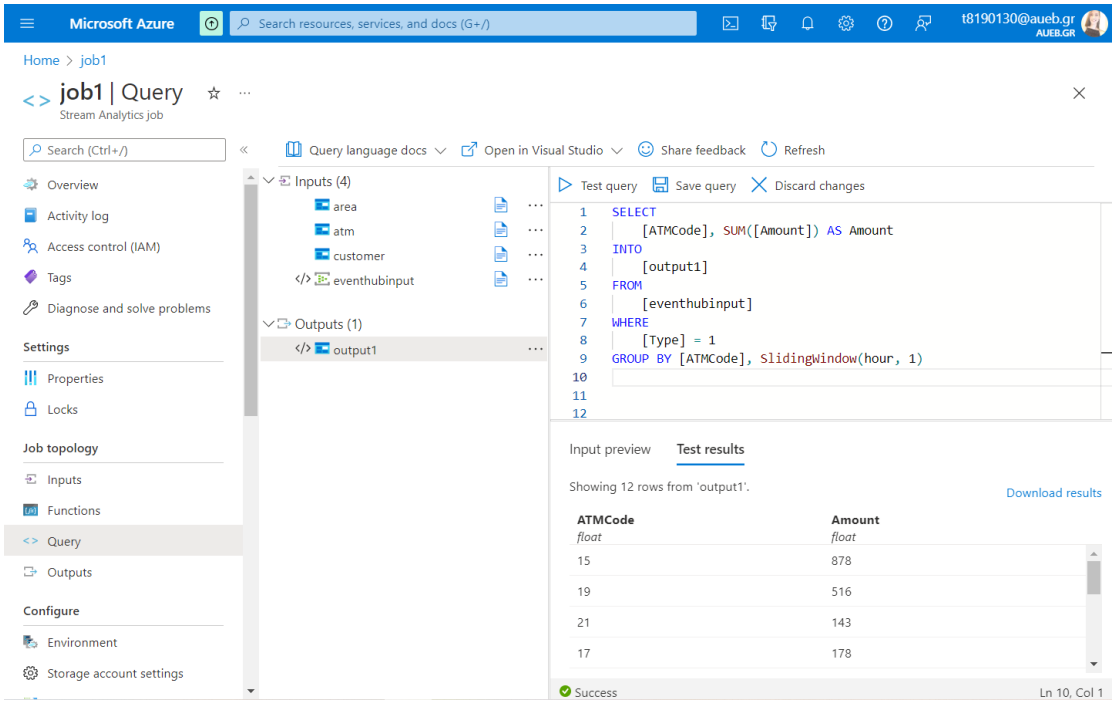


Reference [7]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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

```
SELECT
    [ATMCode], SUM([Amount]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
WHERE
    [Type] = 1
GROUP BY [ATMCode], SlidingWindow(hour, 1)
```

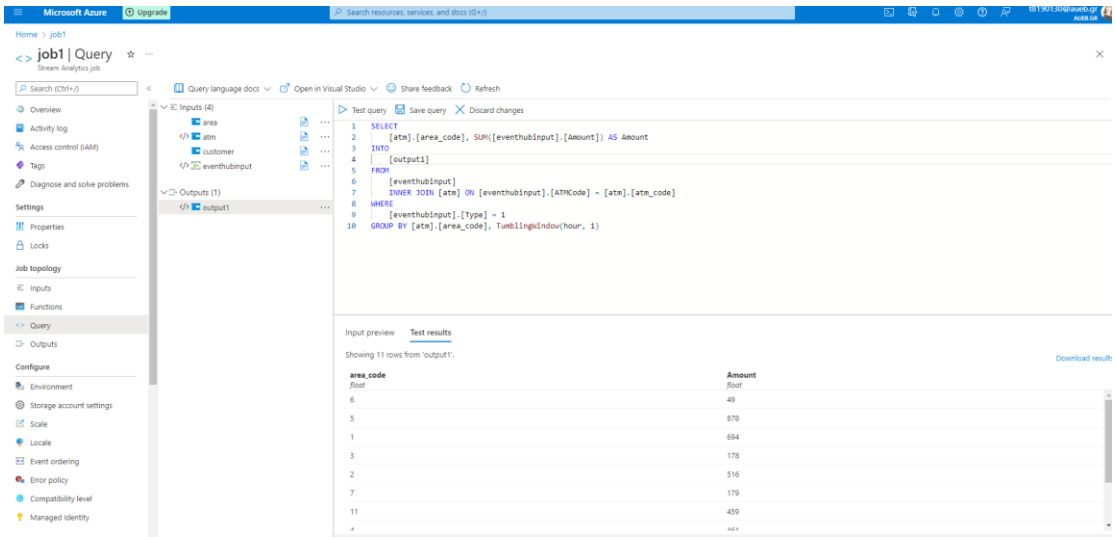


Reference[5]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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).

```
SELECT
    [atm].[area_code], SUM([eventhubinput].[Amount]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
    INNER JOIN [atm] ON [eventhubinput].[ATMCode] = [atm].[atm_code]
WHERE
    [eventhubinput].[Type] = 1
GROUP BY [atm].[area_code], TumblingWindow(hour, 1)
```

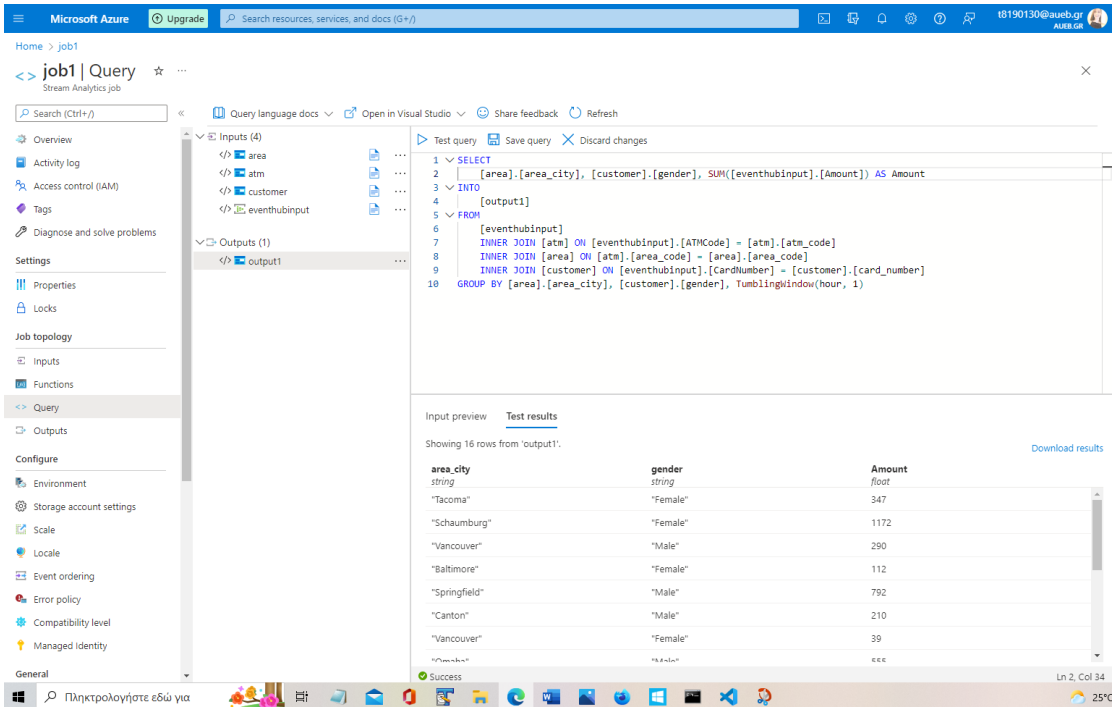


Reference[6]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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).

```
SELECT
    [area].[area_city], [customer].[gender], SUM([eventhubinput].[Amoun
t]) AS Amount
INTO
    [output1]
FROM
    [eventhubinput]
    INNER JOIN [atm] ON [eventhubinput].[ATMCode] = [atm].[atm_code]
    INNER JOIN [area] ON [atm].[area_code] = [area].[area_code]
    INNER JOIN [customer] ON [eventhubinput].[CardNumber] = [customer].
[card_number]
GROUP BY [area].[area_city], [customer].[gender], TumblingWindow(hour,
1)
```

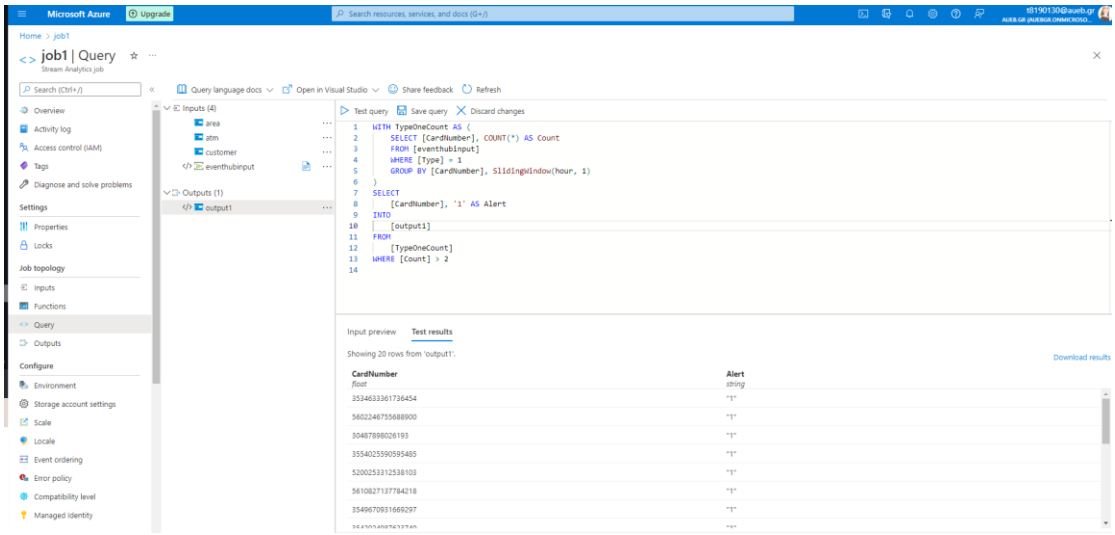


Reference[6]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

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

```
WITH TypeOneCount AS (
    SELECT [CardNumber], COUNT(*) AS Count
    FROM [eventhubinput]
    WHERE [Type] = 1
    GROUP BY [CardNumber], SlidingWindow(hour, 1)
)
SELECT
    [CardNumber], '1' AS Alert
INTO
    [output1]
FROM
    [TypeOneCount]
WHERE [Count] > 2
```

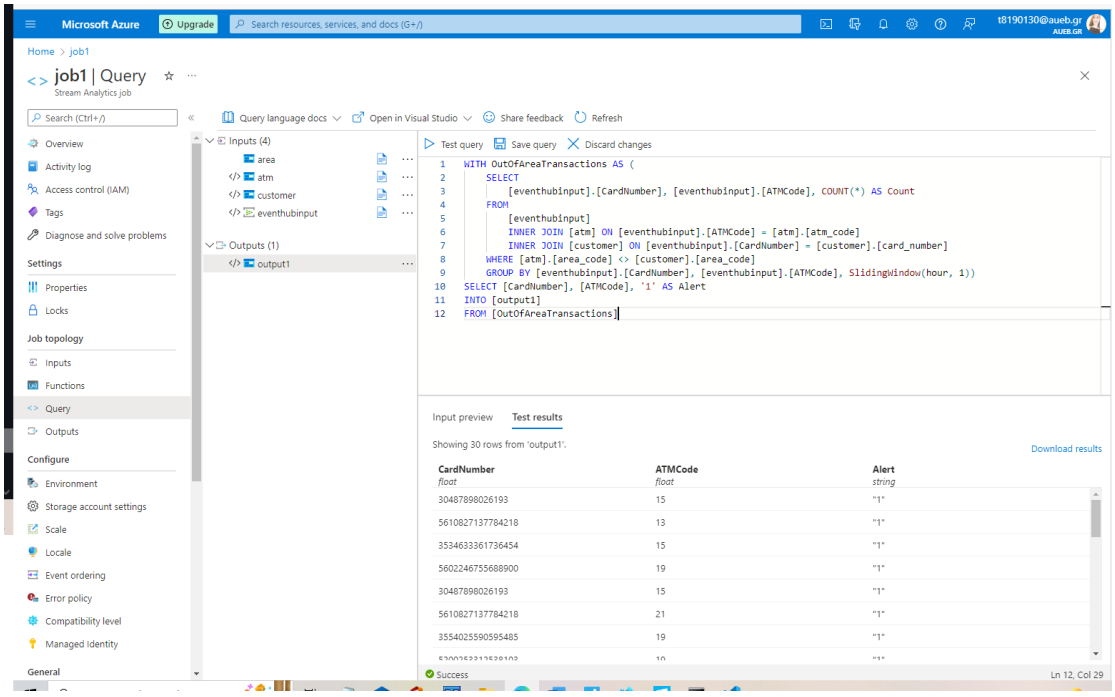


Reference[8]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

Query 8: Alert (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)

```
WITH OutOfAreaTransactions AS (
    SELECT
        [eventhubinput].[CardNumber], [eventhubinput].[ATMCode], COUNT(
*) AS Count
    FROM
        [eventhubinput]
        INNER JOIN [atm] ON [eventhubinput].[ATMCode] = [atm].[atm_code
]
        INNER JOIN [customer] ON [eventhubinput].[CardNumber] = [custom
er].[card_number]
    WHERE [atm].[area_code] <> [customer].[area_code]
    GROUP BY [eventhubinput].[CardNumber], [eventhubinput].[ATMCode], S
lidingWindow(hour, 1))
SELECT [CardNumber], [ATMCode], '1' AS Alert
INTO [output1]
FROM [OutOfAreaTransactions]
```



Reference[8]

Note : in folder output_json exists the exported output from azure interfaces for all queries.

11. Conclusion

In conclusion, as this assignment is the last one, we would like to mention that this course helped as to learn useful tools – systems for big data. Specifically, we worked with Hadoop, Redis, neo4j and azure in practice. Last but not least, we would like to thank our professor Damianos Chatziantoniou for the course. Good summer!

12.References

- [Azure Streams Analytics][1] : Azure Stream Analytics documentation | Microsoft Docs
- [Event Hub][2] : https://azure.microsoft.com/en-us/services/event-hubs/#overview
- [Storage Account][3] : https://docs.microsoft.com/en-us/azure/storage/common/storage-account-overview
- [Create Stream Analytics Job][4] : https://docs.microsoft.com/en-us/azure/stream-analytics/stream-analytics-quick-create-portal
- [Sliding Window][5] : Sliding Window (Azure Stream Analytics) - Stream Analytics Query | Microsoft Docs
- [Tumbling Window][6] : Tumbling Window (Azure Stream Analytics) - Stream Analytics Query | Microsoft Docs
- [Hopping Window][7] : Hopping Window (Azure Stream Analytics) - Stream Analytics Query | Microsoft Docs
- [Sliding Window][8] : Sliding Window (Azure Stream Analytics) - Stream Analytics Query | Microsoft Docs