

# Big Data Management Systems - Semester Project 2025: Data Streams | Azure Stream Analytics

In this project, we demonstrated the use of Azure Stream Analytics to process real-time data streams of ATM transactions.

**Maria Schoinaki, BSc Student**

Department of Informatics, Athens University of Economics and Business  
p3210191@aueb.gr

**Nikos Mitsakis, BSc Student**

Department of Informatics, Athens University of Economics and Business  
p3210122@aueb.gr

## Overview

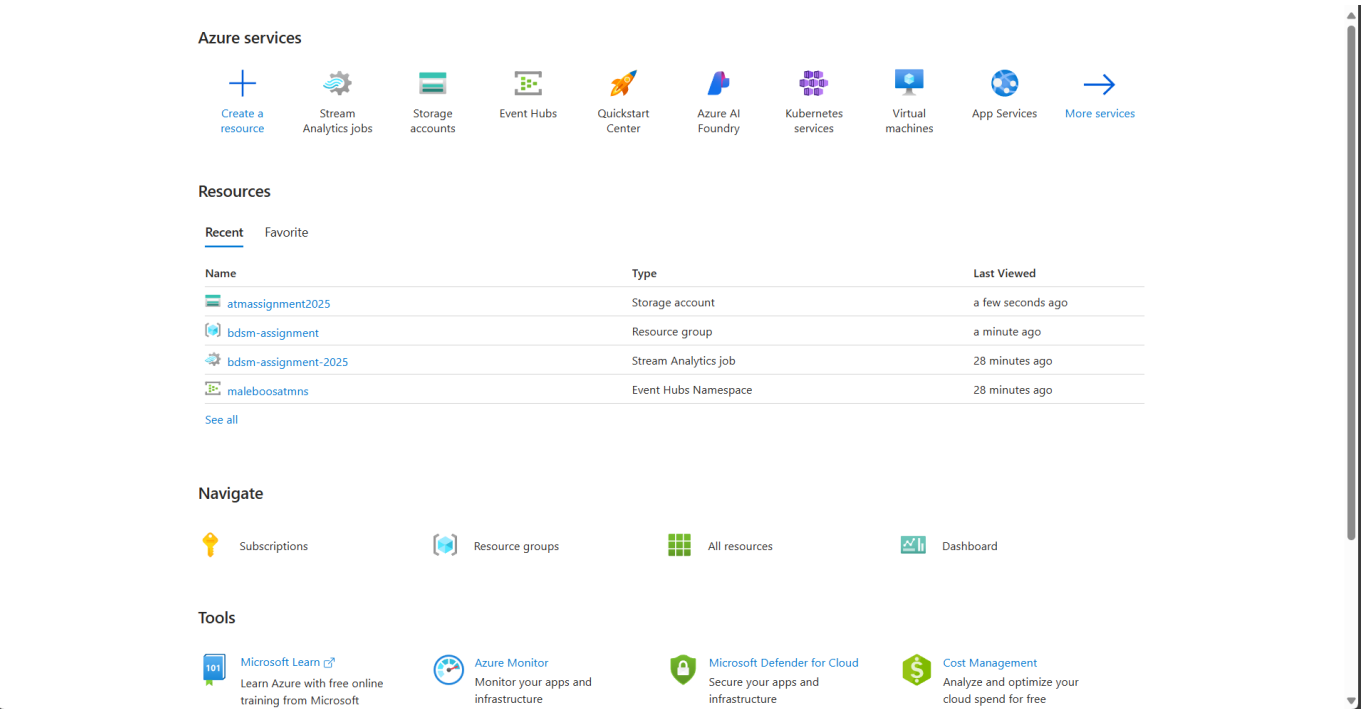
The goal was to configure the Azure environment, ingest streaming data using Event Hub, perform stream analytics queries, and output results to Azure Blob Storage.

Schema of the ATM Transaction Stream: (ATMCode, CardNumber, Type, Amount)

## Step 1: Create an Azure Trial Account

We registered for an Azure Free Trial account to access Azure cloud services required for this assignment.

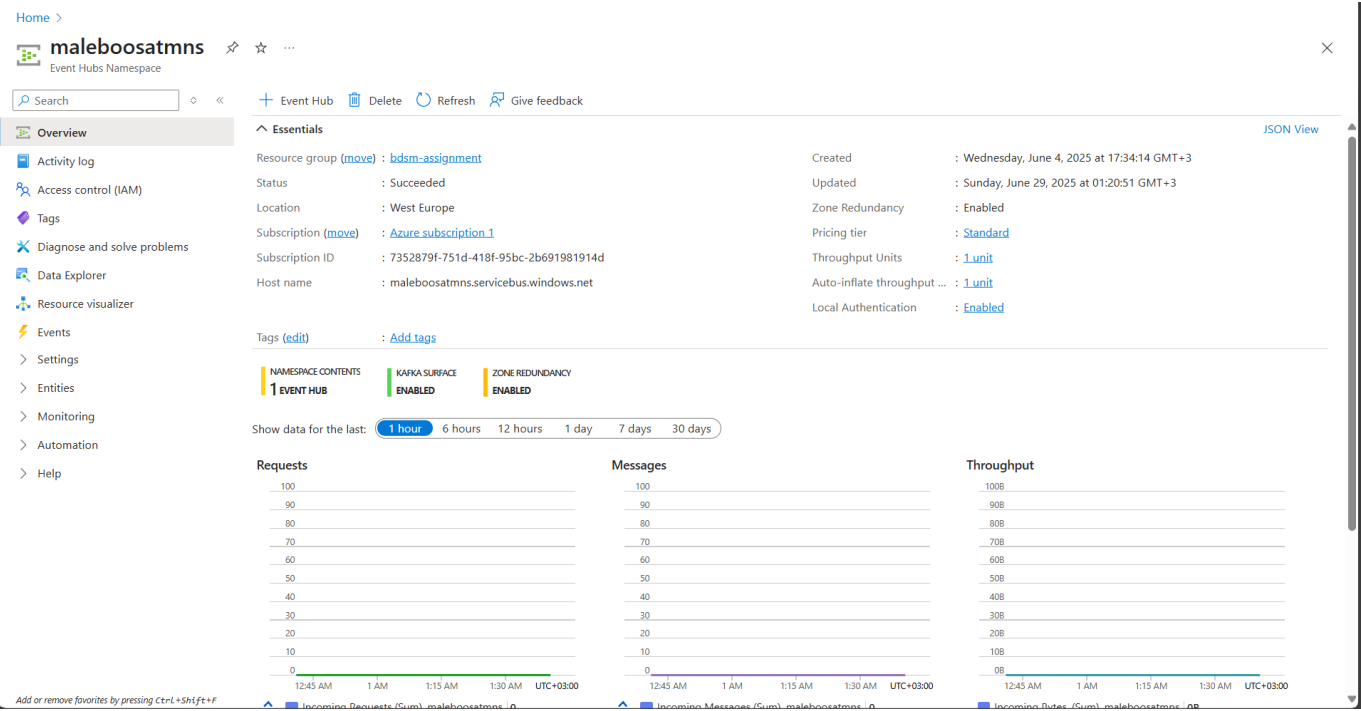
**Screenshot:**



## Step 2: Setup an Event Hub

We created an Event Hub namespace and a hub to receive streaming ATM transaction events.

**Screenshot:**



## Step 3: Generate a Security Access Signature (SAS)

We used the [RedDog Event Hub SAS Generator](#) to create a SAS token for secure data publishing.

**Screenshot:**

The screenshot shows the 'Event Hubs - Signature Generator' application. It has two main sections: 'Hub' and 'Credentials'.

**Hub Configuration:**

- Namespace: maleboosatmns
- Hub Name: atmtransactions
- Publisher: schoin
- Mode: Http

**Credentials Configuration:**

- Sender Key Name: RootManageSharedAccessKey
- Sender Key: (empty field)
- Token TTL (minutes): 60

**Signature:**

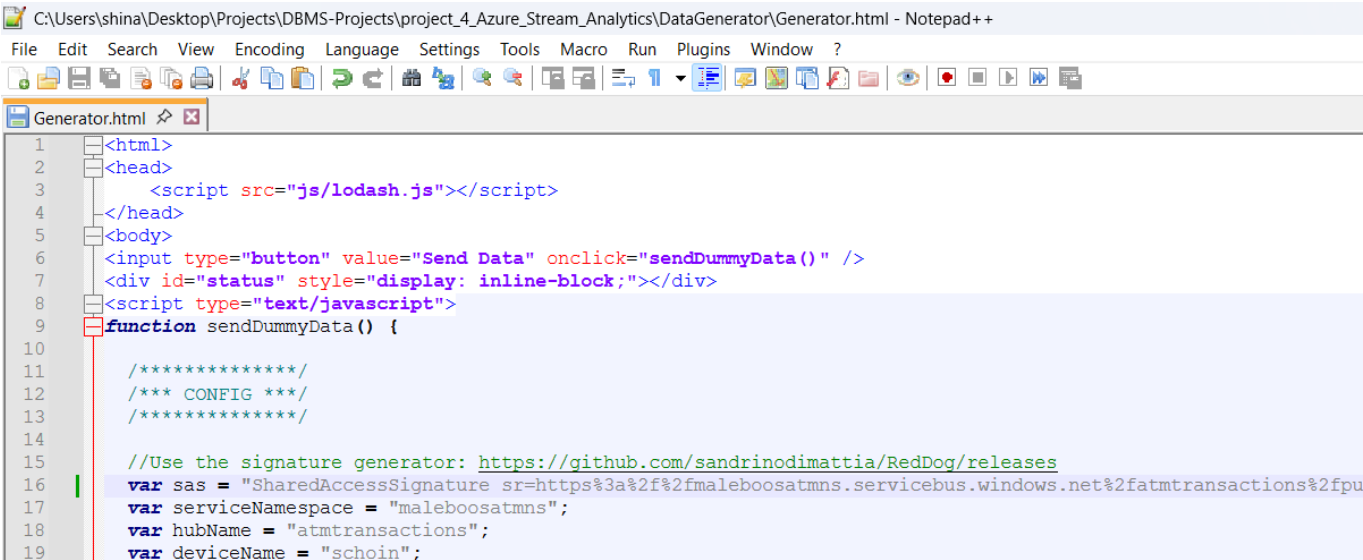
The generated signature is displayed at the bottom:

```
SharedAccessSignature sr=https%3a%2f%2fmaleboosatmns.servicebus.windows.net%2fatmtransactions%2fpublishers%2fschoin
```

## Step 4: Edit Generator.html and Configure Variables

We updated the [Generator.html](#) file with our Event Hub config and the generated SAS token.

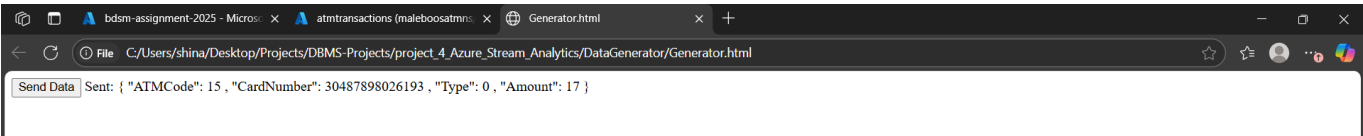
Screenshot:



Step 5: Feed Data into the Event Hub

We opened `Generator.html` in our browser and clicked “Send Data” to generate and send ATM transaction events to the Event Hub.

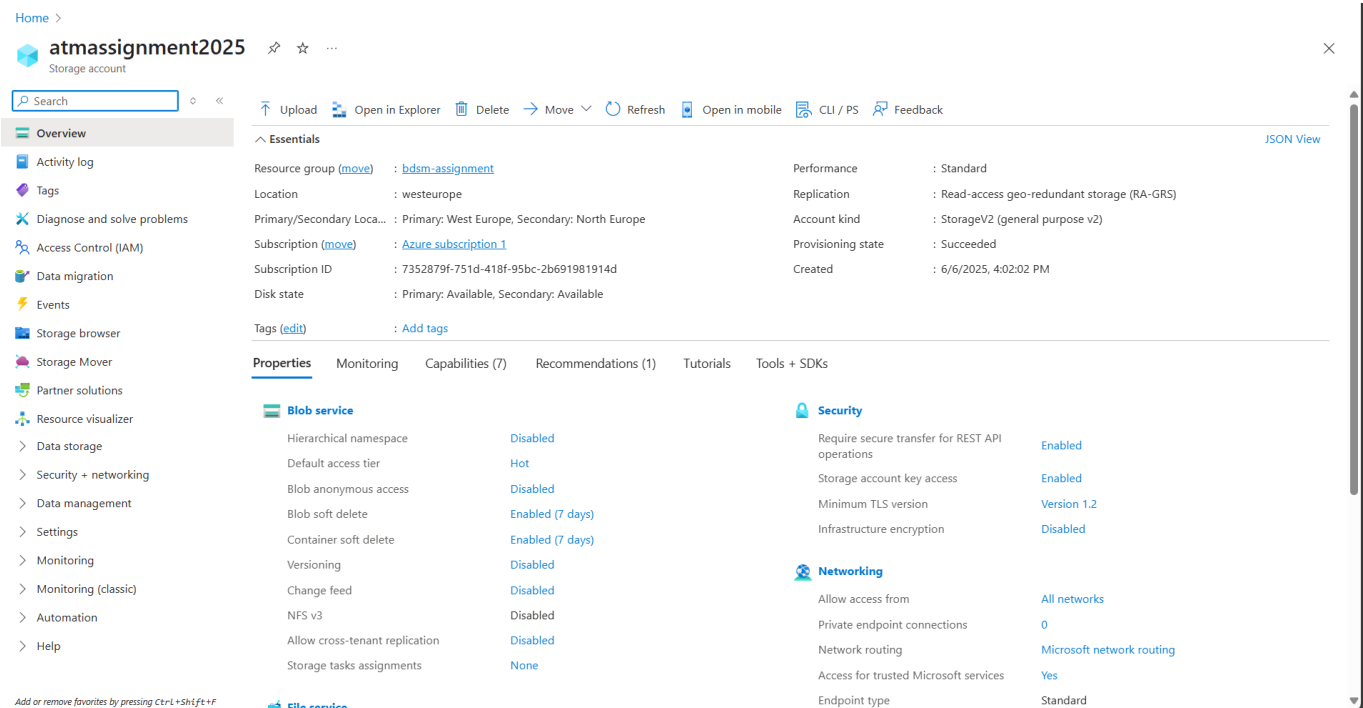
Screenshot:



Step 6: Setup a Storage Account

We created a Storage Account on Azure to store reference and output data.

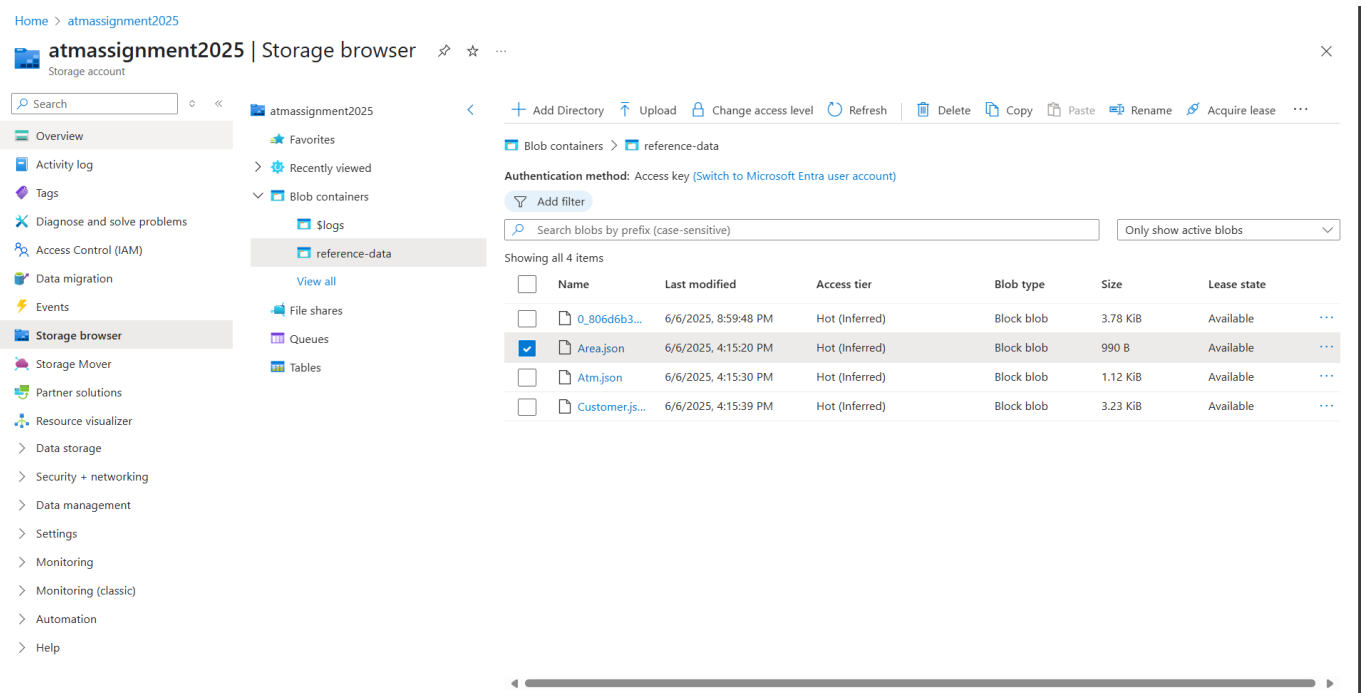
Screenshot:



## Step 7: Upload Reference Data Files

We uploaded the provided reference data files to our Azure Storage account as required for the queries.

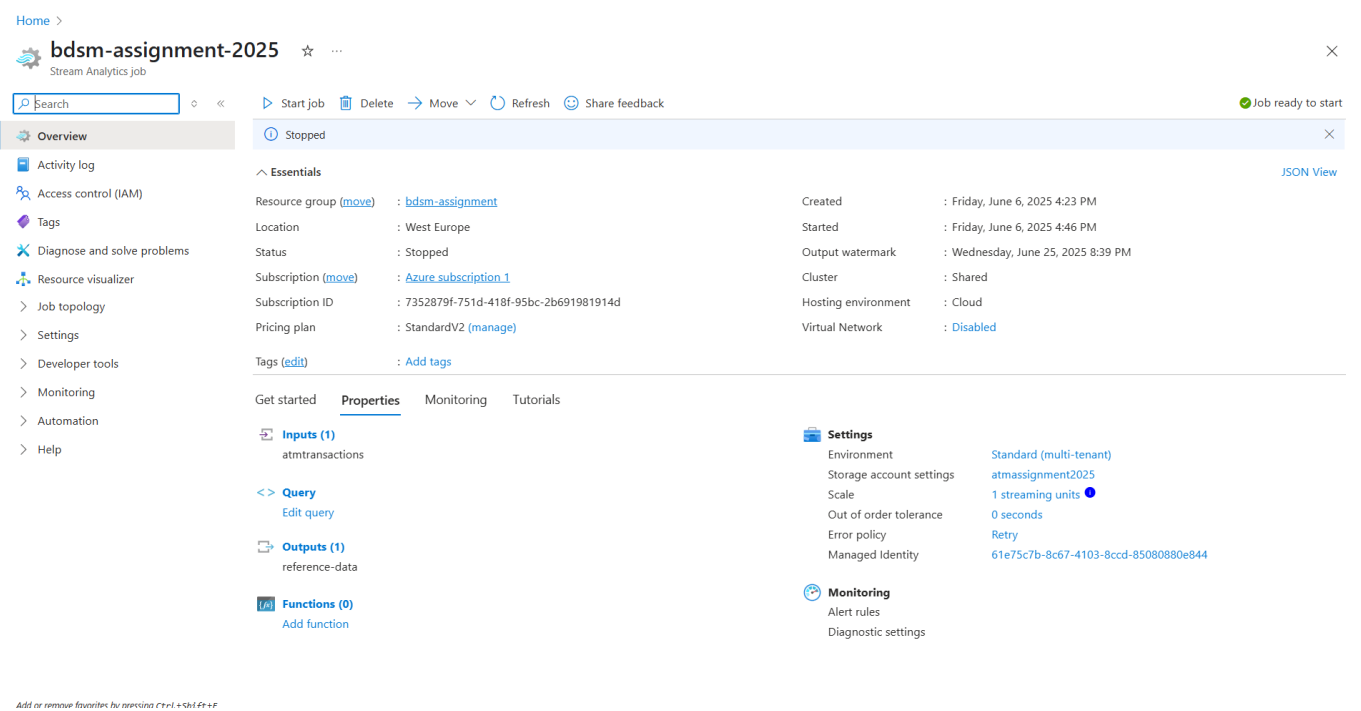
**Screenshot:**



## Step 8: Setup a Stream Analytics Job

We created a Stream Analytics job to process incoming data streams.

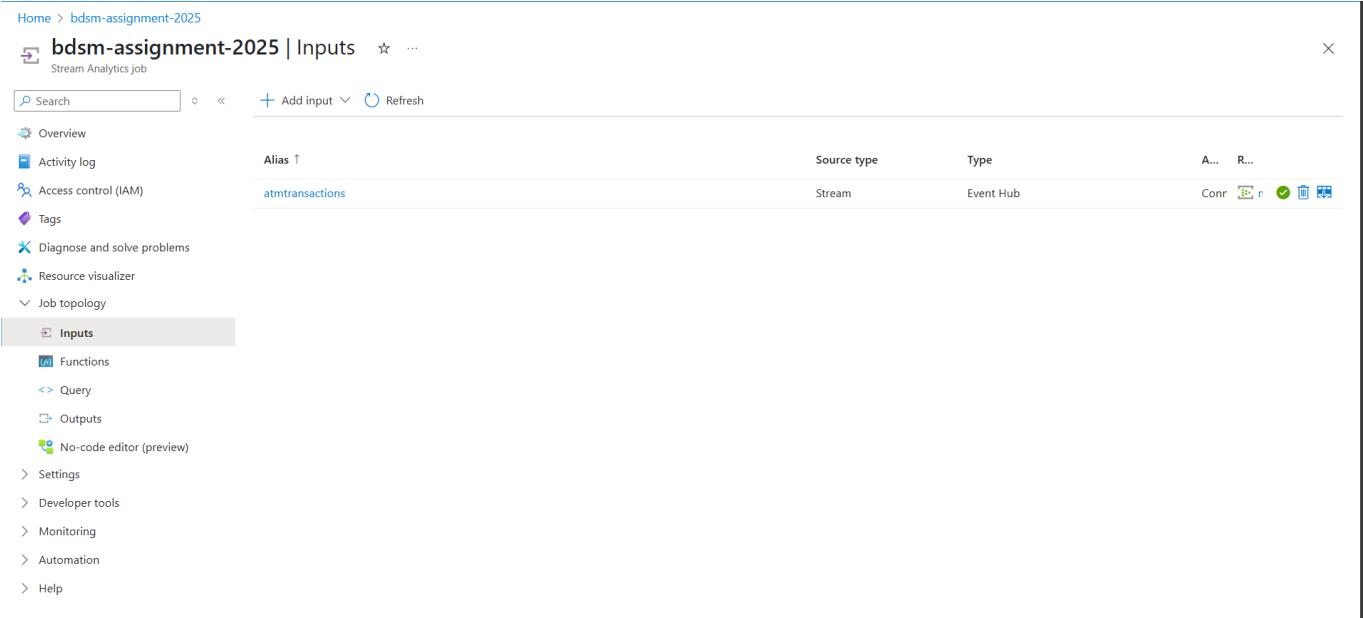
**Screenshot:**



## Step 9: Configure Inputs (Event Hub + Reference Data)

We added our Event Hub as a streaming input and the reference data files as reference inputs in our Stream Analytics job.

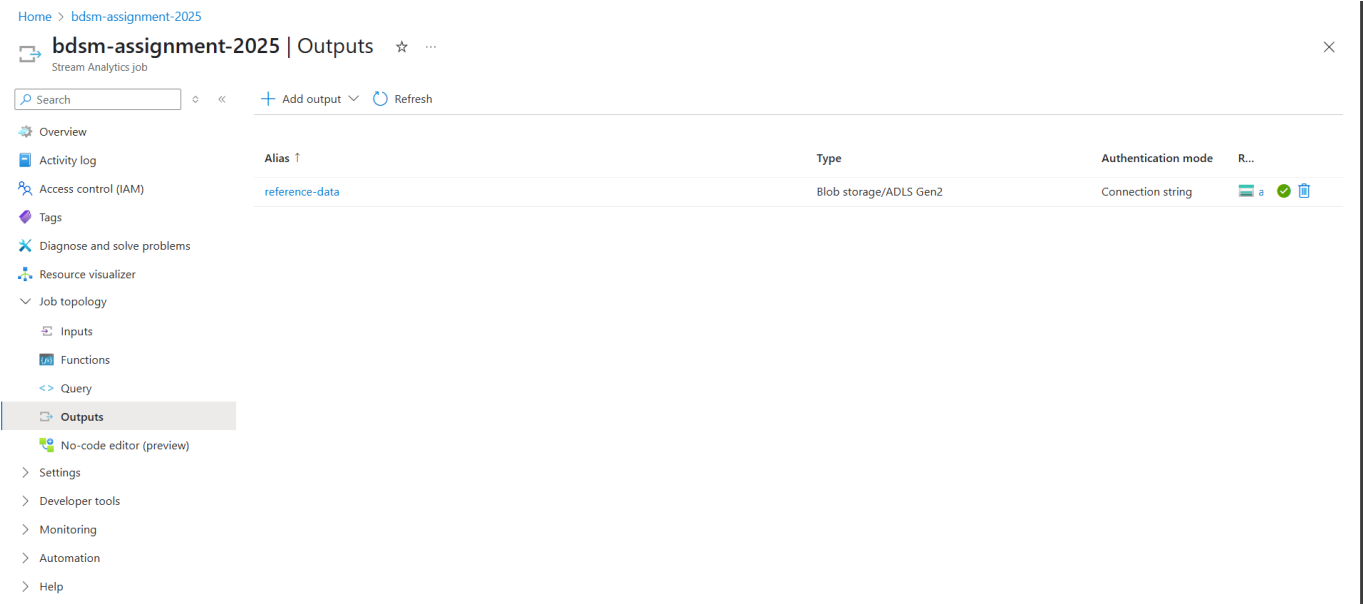
Screenshot:



Step 10: Create Blob Storage Output

We configured a Blob Storage output for the job results.

Screenshot:



Step 11: Run Stream Analytics Queries

We ran the required queries using the Stream Analytics Query Language.

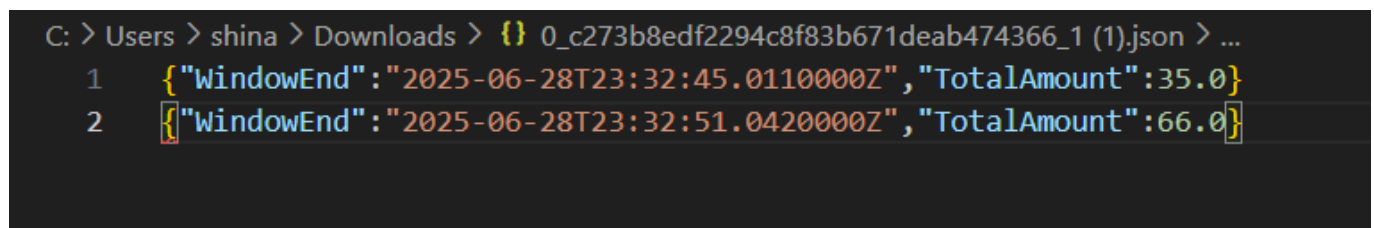
Query 1: Sliding Window (Last 10 Minutes, Type=0, ATMCode=21)

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).

**Query:**

```
SELECT
    System.Timestamp AS WindowEnd,
    SUM(Amount) AS TotalAmount
INTO
    [reference-data] -- output alias
FROM
    [atmtransactions] -- input alias
WHERE
    Type = 0 AND ATMCODE = 21
GROUP BY
    SLIDINGWINDOW(minute, 10)
```

This query outputs the running sum for all matching transactions in the previous 10 minutes. It produces a result whenever a matching event arrives. Output times are not fixed—each output corresponds to an event's time.

**Screenshot:**

```
C: > Users > shina > Downloads > {} 0_c273b8edf2294c8f83b671deab474366_1 (1).json > ...
1  {"WindowEnd":"2025-06-28T23:32:45.011000Z","TotalAmount":35.0}
2  {"WindowEnd":"2025-06-28T23:32:51.042000Z","TotalAmount":66.0}
```

**Query 2: Tumbling Window (Last Hour, Type=1, ATMCODE=21)**

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

**Query:**

```
SELECT
    System.Timestamp AS WindowEnd,
    SUM(Amount) AS TotalAmount
INTO
    [reference-data]
FROM
    [atmtransactions]
WHERE
    Type = 1 AND ATMCODE = 21
GROUP BY
    TUMBLINGWINDOW(hour, 1)
```

This query calculates the total transaction Amount for all transactions where Type = 1 and ATMCODE = 21 for each fixed, non-overlapping 1-hour window. The results are produced once per hour at the end of each tumbling window.

**Screenshot:**

```
C: > Users > shina > Downloads > {} 0_0e8aaaf1d7cf4ca49a50745afbcf3549_1.json > ...  
1 [{"WindowEnd":"2025-06-29T00:00:00.000000Z","TotalAmount":235.0}]
```

**Query 3: Hopping Window (Last Hour, Every 30 Minutes, Type=1, ATMCode=21)**

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).

**Query:**

```
SELECT  
    System.Timestamp AS WindowEnd,  
    SUM(Amount) AS TotalAmount  
INTO  
    [reference-data]  
FROM  
    [atmtransactions]  
WHERE  
    Type = 1 AND ATMCode = 21  
GROUP BY  
    HOPPINGWINDOW(minute, 60, 30)
```

This generates overlapping 1-hour windows every 30 minutes, so each transaction may appear in up to two windows.

**Screenshot:**

```
C: > Users > shina > Downloads > {} 0_dc4f0795769d4e90b010b411e19a8b7d_1.json > ...  
1 [{"WindowEnd":"2025-06-29T00:30:00.000000Z","TotalAmount":199.0}]
```

**Query 4: Sliding Window (Last Hour, Type=1, Per ATMCode)**

Show the total "Amount" of "Type = 1" transactions per "ATM Code" of the last one hour (use a sliding window).

**Query:**

```
SELECT  
    System.Timestamp AS WindowEnd,  
    ATMCode,  
    SUM(Amount) AS TotalAmount  
INTO  
    [reference-data]  
FROM  
    [atmtransactions]  
WHERE
```

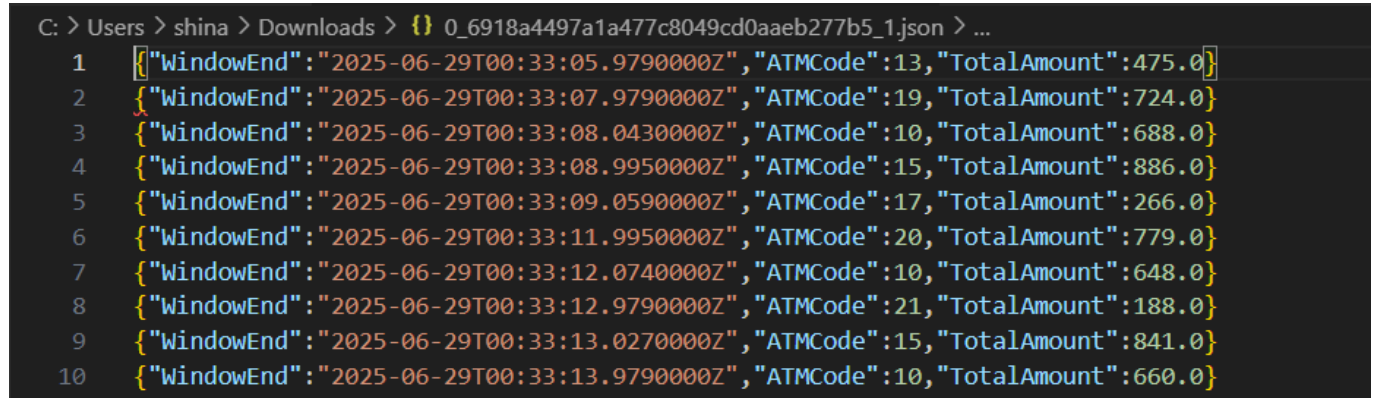
```

    Type = 1
GROUP BY
    ATMCode,
    SLIDINGWINDOW(hour, 1)

```

This shows the running sum per ATMCode for the last hour, updating every time a new event is processed.

#### Screenshot:



```

C: > Users > shina > Downloads > {} 0_6918a4497a1a477c8049cd0aaeb277b5_1.json > ...
1 [{"WindowEnd": "2025-06-29T00:33:05.9790000Z", "ATMCode": 13, "TotalAmount": 475.0}]
2 [{"WindowEnd": "2025-06-29T00:33:07.9790000Z", "ATMCode": 19, "TotalAmount": 724.0}]
3 [{"WindowEnd": "2025-06-29T00:33:08.0430000Z", "ATMCode": 10, "TotalAmount": 688.0}]
4 [{"WindowEnd": "2025-06-29T00:33:08.9950000Z", "ATMCode": 15, "TotalAmount": 886.0}]
5 [{"WindowEnd": "2025-06-29T00:33:09.0590000Z", "ATMCode": 17, "TotalAmount": 266.0}]
6 [{"WindowEnd": "2025-06-29T00:33:11.9950000Z", "ATMCode": 20, "TotalAmount": 779.0}]
7 [{"WindowEnd": "2025-06-29T00:33:12.0740000Z", "ATMCode": 10, "TotalAmount": 648.0}]
8 [{"WindowEnd": "2025-06-29T00:33:12.9790000Z", "ATMCode": 21, "TotalAmount": 188.0}]
9 [{"WindowEnd": "2025-06-29T00:33:13.0270000Z", "ATMCode": 15, "TotalAmount": 841.0}]
10 [{"WindowEnd": "2025-06-29T00:33:13.9790000Z", "ATMCode": 10, "TotalAmount": 660.0}]

```

#### Query 5: Tumbling Window (Last Hour, Type=1, Per Area Code)

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

We added as input stream the Atm.json with alias atmref **Query:**

```

SELECT
    System.Timestamp AS WindowEnd,
    atmref.area_code,
    SUM(t.Amount) AS TotalAmount
INTO
    [reference-data]
FROM
    [atmtransactions] t
    JOIN [atmref] AS atmref
        ON t.ATMCode = atmref.atm_code
WHERE
    t.Type = 1
GROUP BY
    atmref.area_code,
    TUMBLINGWINDOW(hour, 1)

```

This produces a sum per Area Code for each 1-hour period.



**Screenshot:**

```
C: > Users > shina > Downloads > {} 0_7d5a5efa9ac84973bc7482dd6b6d273e_1.json > ...
1  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":5,"TotalAmount":953.0}
2  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":1,"TotalAmount":937.0}
3  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":3,"TotalAmount":125.0}
4  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":11,"TotalAmount":718.0}
5  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":2,"TotalAmount":460.0}
6  {"WindowEnd":"2025-06-29T02:00:00.000000Z","area_code":7,"TotalAmount":339.0}
```

**Query 6: Tumbling Window (Last Hour, Per City & Gender)**

Show the total "Amount" per ATM's "City" and Customer's "Gender" of the last hour. Repeat once every hour (use a tumbling window).

We added as input reference the Customer.json with alias customer, the Area.json with alias area and the Atm.json with alias atmref **Query:**

```
SELECT
    System.Timestamp AS WindowEnd,
    atmcity.area_city AS City,
    cust.gender AS Gender,
    SUM(t.Amount) AS TotalAmount
INTO
    [reference-data]
FROM
    [atmtransactions] t
    JOIN [atmref] AS atmref      -- ATM area code
        ON t.ATMCode = atmref.atm_code
    JOIN [area] AS atmcity      -- ATM city
        ON atmref.area_code = atmcity.area_code
    JOIN [customer] AS cust     -- Customer info
        ON t.CardNumber = cust.card_number
WHERE
    t.Type = 1
GROUP BY
    atmcity.area_city,
    cust.gender,
    TUMBLINGWINDOW(hour, 1)
```

This produces a breakdown per ATM city and gender for each hour.

**Screenshot:**

```
C: > Users > shina > Downloads > {} 0_6a8e09b1f6d345929fb7a0b23a8cd809_1.json > ...
1  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Schaumburg","Gender":"Female","TotalAmount":609.0}
2  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Baltimore","Gender":"Male","TotalAmount":114.0}
3  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Omaha","Gender":"Female","TotalAmount":24.0}
4  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Memphis","Gender":"Male","TotalAmount":544.0}
5  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Tacoma","Gender":"Female","TotalAmount":164.0}
6  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Memphis","Gender":"Female","TotalAmount":219.0}
7  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Springfield","Gender":"Male","TotalAmount":454.0}
8  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Canton","Gender":"Male","TotalAmount":149.0}
9  {"WindowEnd":"2025-06-29T12:00:00.000000Z","City":"Schaumburg","Gender":"Male","TotalAmount":153.0}
```

**Query 7: Sliding Window Alert (Customer, 2+ Transactions in 1 Hour, Type=1)**

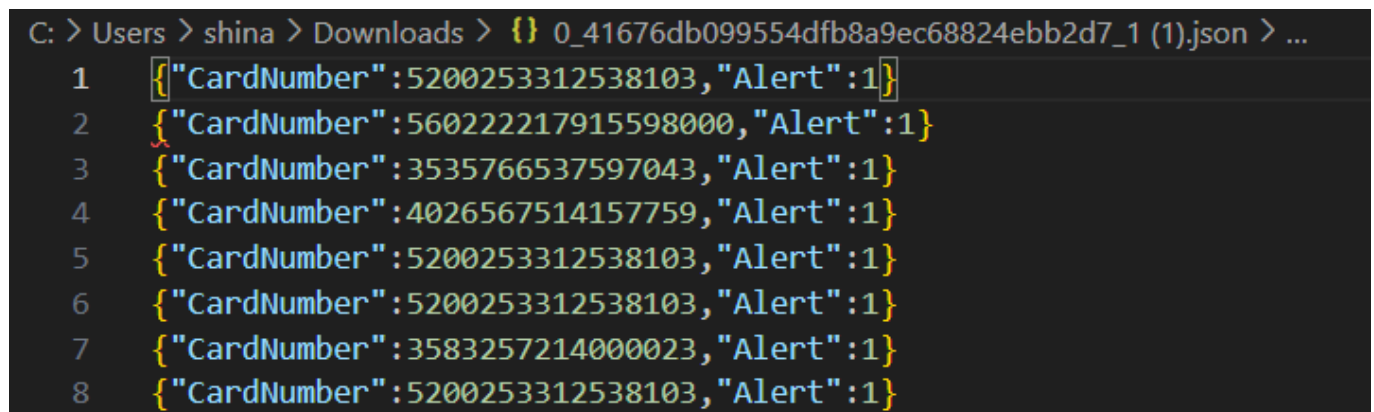
Alert (SELECT "1") if a Customer has performed two transactions of "Type = 1" in a window of an hour (use a sliding window).

#### Query:

```
SELECT
    CardNumber,
    1 AS Alert
INTO
    [reference-data]
FROM
    [atmtransactions]
WHERE
    Type = 1
GROUP BY
    CardNumber,
    SLIDINGWINDOW(hour, 1)
HAVING
    COUNT(*) >= 2
```

This outputs an alert whenever a customer has two or more Type=1 transactions in a 1-hour window.

#### Screenshot:



```
C: > Users > shina > Downloads > {} 0_41676db099554dfb8a9ec68824ebb2d7_1 (1).json > ...
1  {"CardNumber":5200253312538103,"Alert":1}
2  {"CardNumber":560222217915598000,"Alert":1}
3  {"CardNumber":3535766537597043,"Alert":1}
4  {"CardNumber":4026567514157759,"Alert":1}
5  {"CardNumber":5200253312538103,"Alert":1}
6  {"CardNumber":5200253312538103,"Alert":1}
7  {"CardNumber":3583257214000023,"Alert":1}
8  {"CardNumber":5200253312538103,"Alert":1}
```

#### Query 8: Sliding Window Alert (ATM AreaCode ≠ Customer AreaCode)

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)

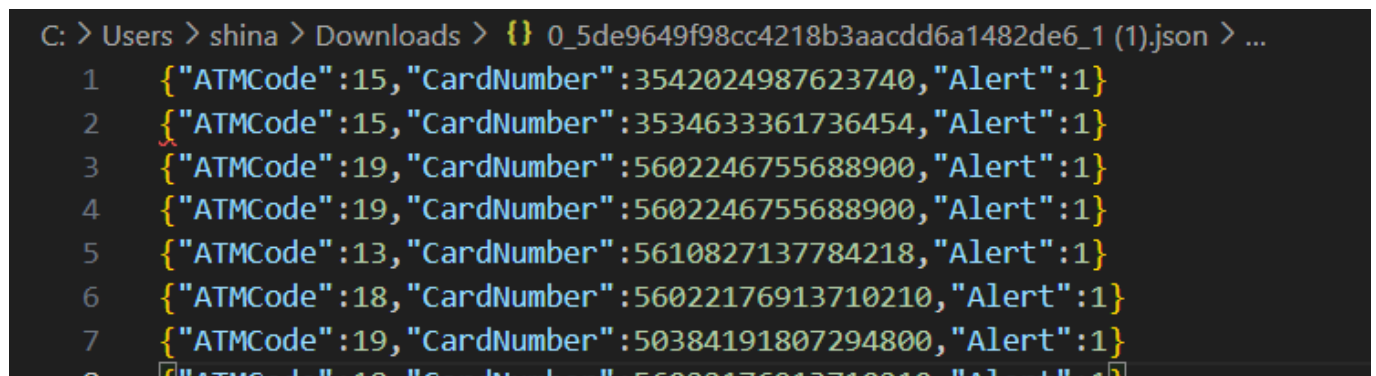
We added as input reference the Customer.json with alias customer and the Atm.json with alias atmref **Query:**

```
SELECT
    t.ATMCode,
    t.CardNumber,
    1 AS Alert
INTO
    [reference-data]
FROM
    [atmtransactions] t
```

```
JOIN [atmref] AS atmref
  ON t.ATMCode = atmref.atm_code
JOIN [customer] AS cust
  ON t.CardNumber = cust.card_number
WHERE
  atmref.area_code <> cust.area_code
GROUP BY
  t.ATMCode, t.CardNumber, SLIDINGWINDOW(hour, 1)
```

This outputs an alert for any transaction where the customer's area code does not match the ATM's area code within a 1-hour sliding window.

#### Screenshot:



The screenshot shows a command prompt window with the following text:

```
C: > Users > shina > Downloads > {} 0_5de9649f98cc4218b3aacdd6a1482de6_1 (1).json > ...
```

Below the command prompt, there is a list of JSON objects, each on a new line, representing transactions where the area code does not match:

```
1 {"ATMCode":15,"CardNumber":3542024987623740,"Alert":1}
2 {"ATMCode":15,"CardNumber":3534633361736454,"Alert":1}
3 {"ATMCode":19,"CardNumber":5602246755688900,"Alert":1}
4 {"ATMCode":19,"CardNumber":5602246755688900,"Alert":1}
5 {"ATMCode":13,"CardNumber":5610827137784218,"Alert":1}
6 {"ATMCode":18,"CardNumber":56022176913710210,"Alert":1}
7 {"ATMCode":19,"CardNumber":50384191807294800,"Alert":1}
8 {"ATMCode":19,"CardNumber":56022176913710210,"Alert":1}
```

## References

- [Azure Stream Analytics Documentation](#)
- [Azure Event Hubs Documentation](#)