

Business Requirement

Scenario: A national weather monitoring organization (ABC), aims to enhance real-time weather analysis for operational efficiency and public safety. The organization needs a robust system to ingest, process, and analyze real-time weather data from sensors distributed across the country. The processed data must provide actionable insights, such as average weather metrics, trends, and conditions, for internal stakeholders and public users.

The primary objectives are:

1. **Real-Time Monitoring:** Collect and process live weather data, including temperature, humidity, wind speed, and precipitation, to ensure immediate responses to severe weather events.
 2. **Data Reliability:** Store the streaming data with high accuracy for historical analysis and future predictions.
 3. **Actionable Insights:** Aggregate data for trend analysis over time intervals to support decision-making.
 4. **Public Dashboard:** Enable visualization of real-time weather trends via an interactive dashboard for internal stakeholders and public safety alerts.
-

Solution

Solution Overview: The Azure-based real-time streaming solution efficiently addresses the business requirement by leveraging structured streaming, scalable cloud services, and interactive visualizations.

1. **Data Ingestion with Azure Event Hub:**
 - The solution collects real-time weather data using Azure Event Hub with a throughput unit designed for a high volume of events.
 - Data is stored temporarily for immediate streaming to processing pipelines.
2. **Data Processing in Azure Databricks:**
 - **Bronze Layer:** Ingests raw weather data and persists it in Delta format for reliability and reprocessing, ensuring all data is stored with exactly-once semantics.
 - **Silver Layer:** Cleans and transforms the data, converting JSON structures into a structured tabular format with specific fields like temperature, humidity, wind speed, and conditions.
 - **Gold Layer:** Aggregates weather metrics into 5-minute intervals, providing summarized insights for trends and historical analysis.
3. **Data Visualization with Power BI:**

- **Silver Layer Visualization:** Offers near-real-time data monitoring for immediate insights.
 - **Gold Layer Visualization:** Enables analytical dashboards with aggregated metrics, supporting trend analysis over time.
4. **Scalability & Fault Tolerance:**
- Implemented checkpointing and Delta Lake for reliable and fault-tolerant data streaming.
 - Scalable architecture allows increasing throughput units or clusters to handle spikes in sensor data volume.
5. **Business Outcomes:**
- **Enhanced Weather Awareness:** Real-time data visualization enables proactive responses to changing weather conditions.
 - **Efficient Decision-Making:** Aggregated metrics support better strategic planning for weather-dependent activities.
 - **Improved Public Safety:** Accurate and timely weather data can trigger alerts, reducing risks associated with extreme weather.

This solution demonstrates how Azure's real-time streaming capabilities, Databricks' processing power, and Power BI's visualization tools work cohesively to meet ClimaSecure's business needs, ensuring both immediate responsiveness and strategic weather insights.

Steps I followed:

I created a resource group in Azure with the name "streaming-rg". This resource group consists of all the azure resources that I will use in this project. In the resource group, I created an event hub service. I gave a name to the namespace "real time-streaming". I choose the same region in which I have created my resource group (UK south). In the pricing tier, I selected "Basic". The "Basic" pricing tier allows only 1 consumer group and message retention period is 1 day. This means that event hubs store the messages only for 1 day. The "Basic" pricing tier does not have the "Capture" capability.

"Capture" capability allows to automatically capture the streaming data and store it to Azure data lake storage for processing and analytics.

For the throughput units, I have selected "1"

Throughput units (TUs) in Azure Event Hub represent a capacity measure used to determine the scale of data ingress (incoming data) and egress (outgoing data) for an Event Hub namespace. They are a key part of managing and scaling your Event Hub service. Each throughput unit provides a certain level of performance capacity for data streaming and is billed on a per-hour basis.

Key Characteristics of Throughput Units

1. Data Ingress (Incoming Events):

- Each throughput unit allows for **up to 1 MB/sec of data ingress** or 1,000 events per second (whichever is reached first).
 - This includes all data sent to an Event Hub instance within the namespace.
2. **Data Egress (Outgoing Events):**
- Each throughput unit allows for **up to 2 MB/sec of data egress**.
 - This includes data read from partitions by consumer applications or through Azure Stream Analytics.
3. **Scalability:**
- You can scale up the number of throughput units to handle larger volumes of data. The maximum number depends on your pricing tier (Basic, Standard, or Dedicated).
 - For example, the Standard tier typically supports up to 20 throughput units by default (more available upon request).
4. **Shared Across Namespace:**
- Throughput units are shared across all Event Hubs within a single namespace, making it a cost-effective way to manage multiple streams.
5. **Burst Mode:**
- If data exceeds the allotted throughput unit capacity, Event Hub may throttle requests, causing potential delays or dropped messages.

How to Determine Throughput Units?

The number of throughput units you need depends on:

- **Event Size:** The size of the events being sent (in kilobytes).
- **Event Frequency:** The number of events per second being sent to or read from the Event Hub.
- **Number of Consumers:** The volume of data being consumed by receivers also counts toward egress.

Example Use Case:

- You have an application sending 500 KB of data every second to Event Hub.
- At the same time, you have a consumer application reading the same amount of data.
- For 1 MB/sec ingress and 1 MB/sec egress, you'd need **at least 1 throughput unit**.

Pricing Impact:

Throughput units affect the cost of running your Event Hub service. Make sure to monitor usage and scale up or down as necessary to optimize performance and costs.

I created the resource once the validation is completed.

The screenshot shows the Microsoft Azure portal with a deployment overview for a service named "realtime-streaming". The deployment is marked as complete with a green checkmark. Key details include:

- Deployment name: realtime-streaming
- Subscription: Free Trial
- Resource group: streaming-rg
- Start time: 11/23/2024, 6:52:06 PM
- Correlation ID: 60b6fa04-9f1c-41a3-b0fa-c35...

The "Deployment details" section shows a table with one item:

Resource	Type	Status	Operat
realtime-streaming	Event Hubs Namespace	OK	Operat

Below the table, there's a "Next steps" section with a "Go to resource" button.

On the right side of the page, there are promotional cards for "Cost management" and "Microsoft Defender for Cloud".

Once the namespace creation is completed, I created an Event Hub in the namespace with the name “streaming”. The partition count is set to “1” and the retention time is set to 1 hour (this can be adjusted) as shown below

The screenshot shows the Microsoft Azure portal with the "Create Event Hub" wizard. The "Event Hub Details" step is active, showing the following configuration:

- Name: streaming
- Partition count: 1

The "Retention" step is also visible, showing:

- Cleanup policy: Delete
- Retention time (hrs): 1

At the bottom of the screen, there are navigation buttons: "Review + create", "< Previous", and "Next: Capture >".

realtime-streaming | Event Hubs

Name	Status	Message retention	Partition count
streaming	Active	1 hour	1

Event hubs also allows to generate data in a desired format such as binary, JSON, XML as shown below

streaming (realtime-streaming/streaming) | Data Explorer (preview)

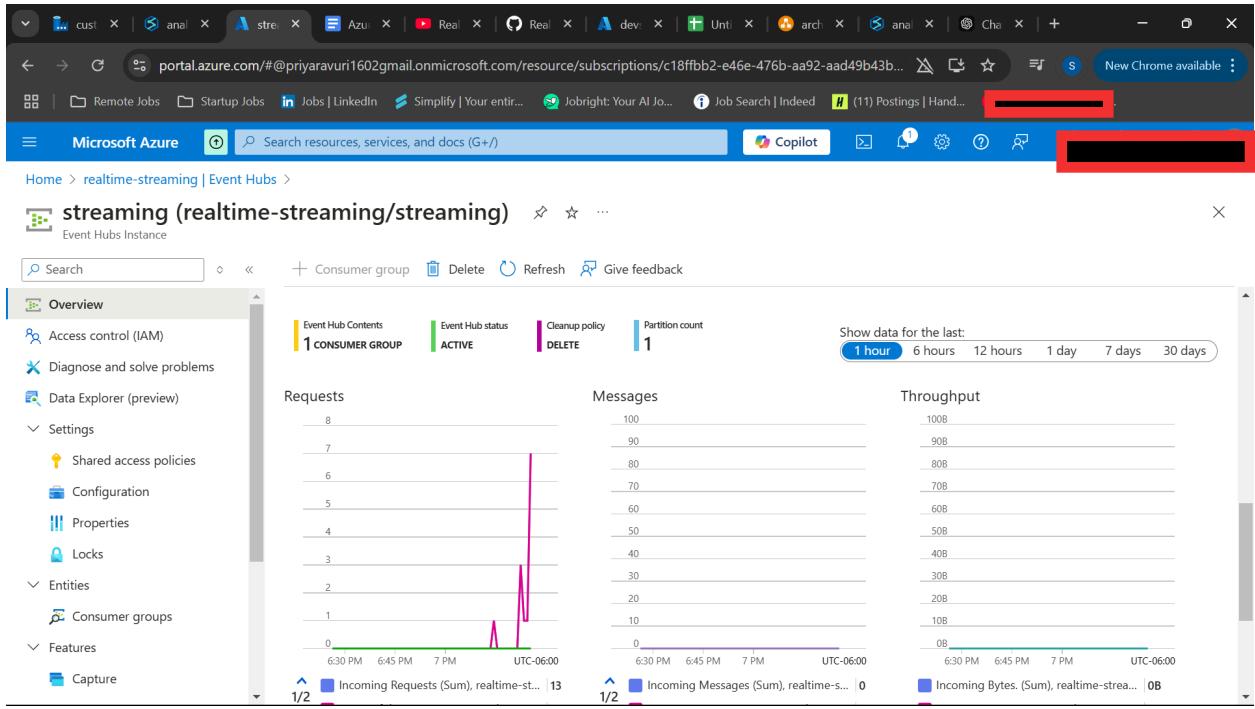
Select Dataset *

Custom payload

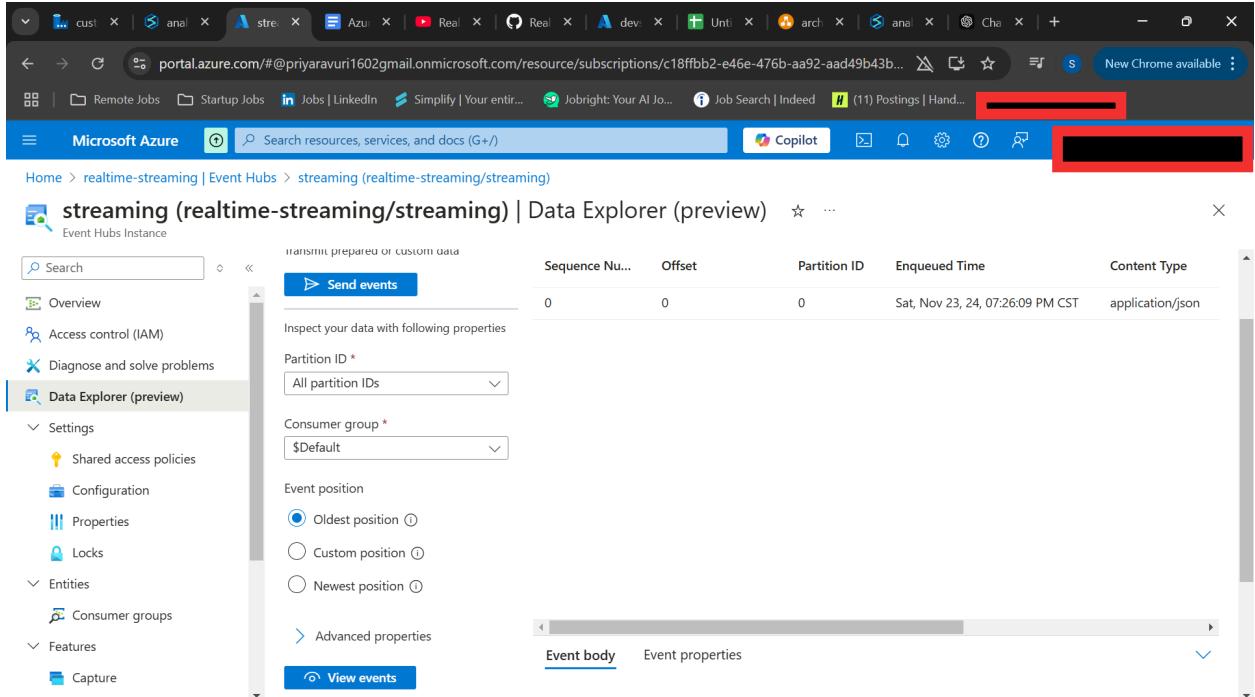
Select Content-Type *

JSON

Once I send an event, I can see a spike in the requests



I can also see the event as shown below



Now, I proceeded to create an azure databricks service with the following configuration

Once the Azure databricks service is deployed, I can see one more resource group which is automatically provisioned with the name “[databricks-rg-streamingspace-err6ub6nrg57o](#)”. This is the resource group that will be used by databricks. It has the following resources provisioned

Resource groups

databricks-rg-streamingspace-err6ub6nrg57o

Name	Type	Location	...
dbmanagedidentity	Managed Identity	UK South	...
dbstoragef1xoh7sdv3nxc	Storage account	UK South	...
unity-catalog-access-connector	Access Connector for ...	UK South	...
workers-sg	Network security group	UK South	...
workers-vnet	Virtual network	UK South	...

I launched the workspace and created a single node cluster with the following configuration

Compute

Configuration

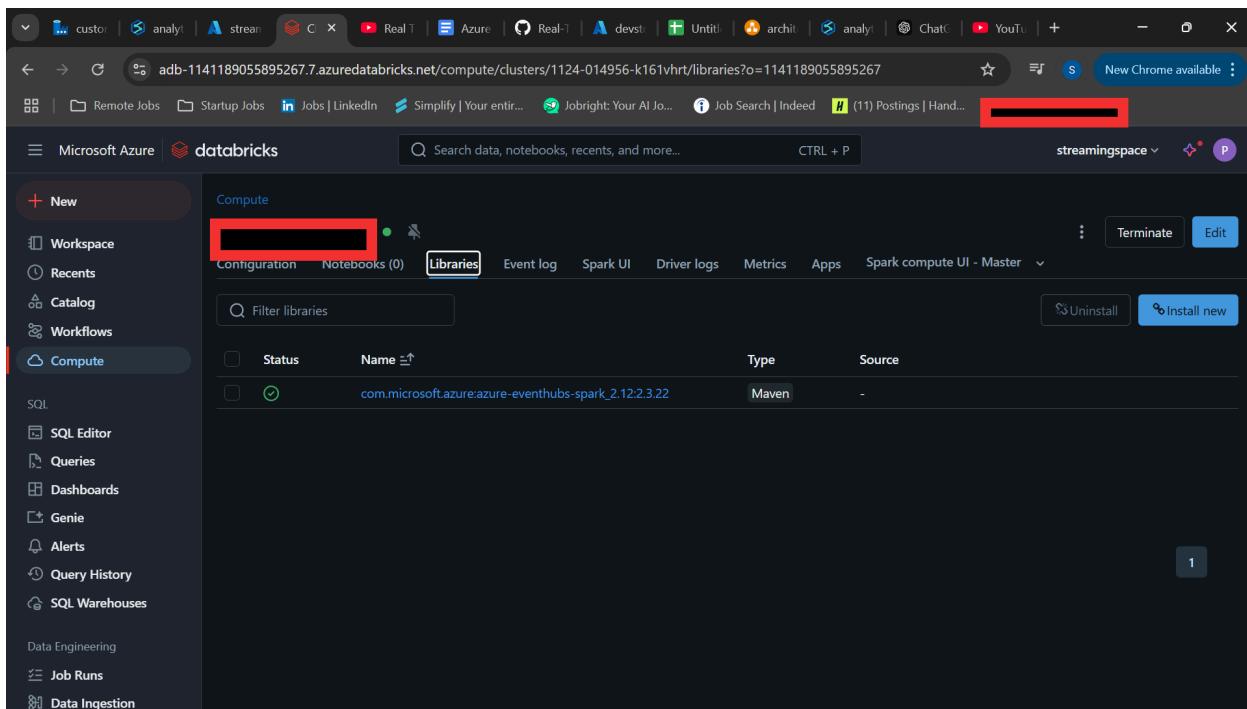
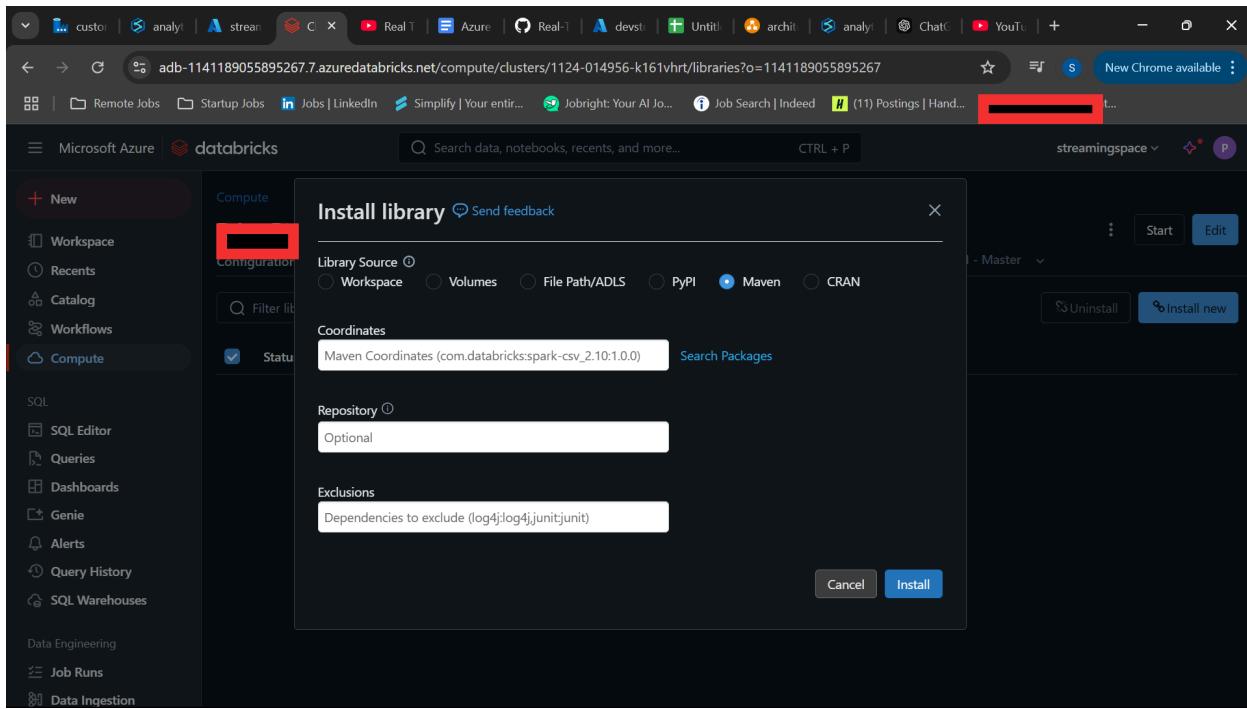
Databricks Runtime Version: 15.4 LTS ML (includes Apache Spark 3.5.0, Scala 2.12)

Node type: Standard_DS3_v2 (14 GB Memory, 4 Cores)

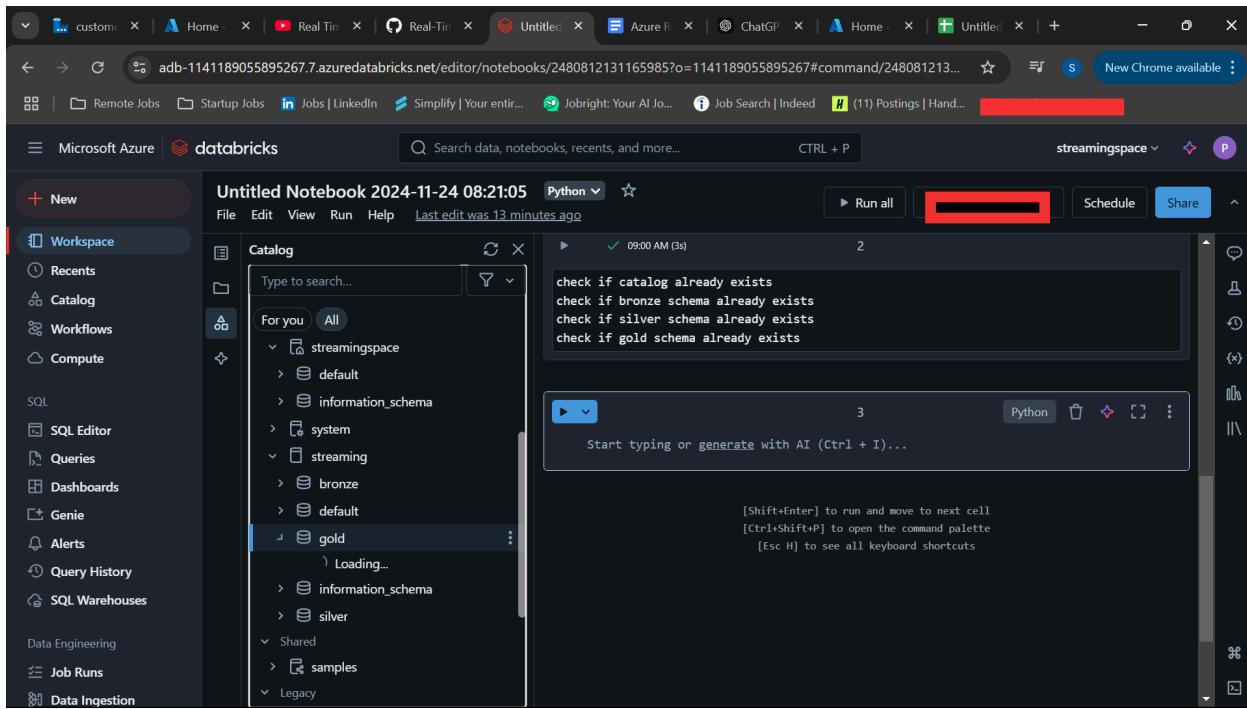
Tags: No custom tags, Automatically added tags

Summary

Then I installed an event hub for spark maven library on this cluster as shown below



Then, I created a catalog with the name "streaming" and I created three schemas with the names "Bronze", "silver", "gold"



I created a shared access policy with the name “db” and “Listen” permission in the event hub which I created earlier.

The screenshot shows the Azure portal interface. The URL is <https://portal.azure.com/#/resource/subscriptions/c18ffbb2-e46e-476b-aa92-aad49b43bbb3/resourceGroups/realtime-streaming/providers/Microsoft.EventHub/eventHubs/streaming>. The page title is "streaming (realtime-streaming/streaming) | Shared access policies". The left sidebar shows navigation options like Overview, Access control (IAM), Diagnose and solve problems, Data Explorer (preview), Settings, Shared access policies, Configuration, Properties, Locks, Entities, Consumer groups, Features, and Capture. The "Shared access policies" section is selected. On the right, a modal window titled "Add SAS Policy" is open, showing fields for "Policy name" (set to "db") and checkboxes for "Manage", "Send", and "Listen" (which is checked). A "Create" button is at the bottom of the modal.

I copied the “Connection string–primary key” and the event hub name and pasted it in the below code in the databricks notebook

```
# Config
# Replace with your Event Hub namespace, name, and key
```

```

connectionString = "Paste connection string here"
eventHubName = "Type event hub name here"

ehConf = {
    'eventhubs.connectionString':
        sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt(connectionString),
    'eventhubs.eventHubName': eventHubName
}

```

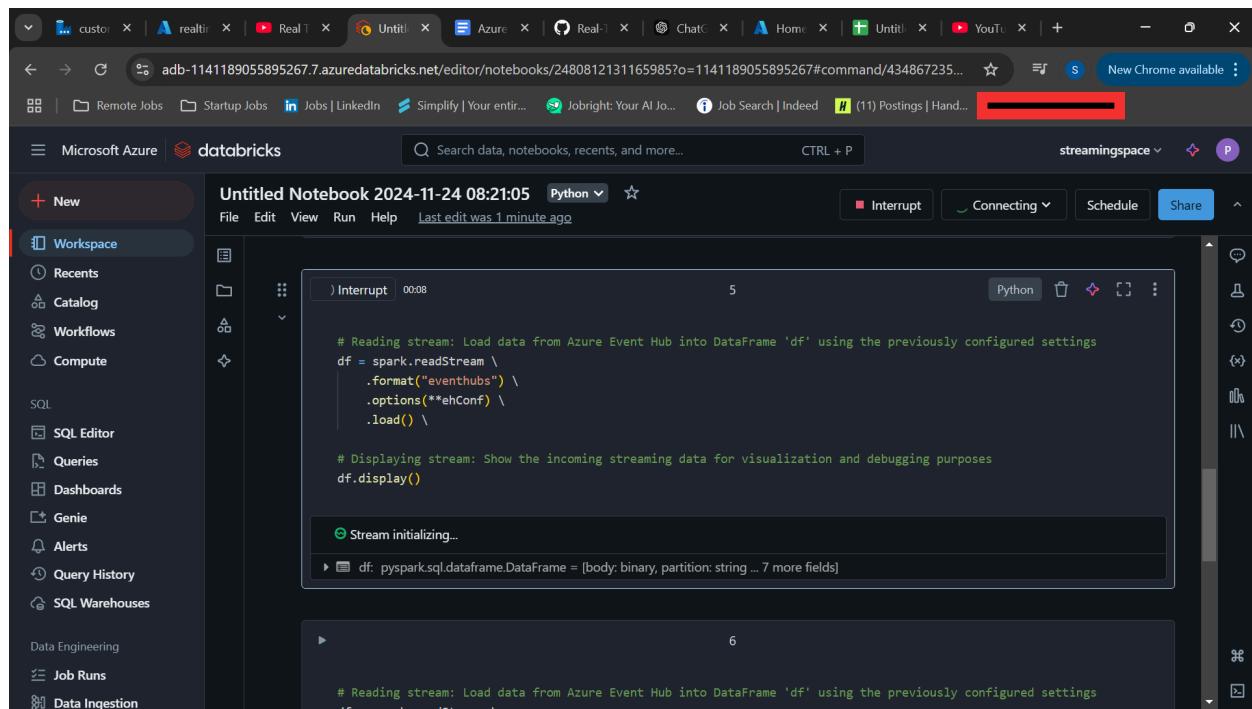
I initiated the read stream with the below code

```

# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previously
configured settings
df = spark.readStream \
    .format("eventhubs") \
    .options(**ehConf) \
    .load()

# Displaying stream: Show the incoming streaming data for visualization and debugging
purposes
df.display()

```



The screenshot shows a Databricks notebook titled "Untitled Notebook 2024-11-24 08:21:05" running in Python. The notebook contains the following code:

```

# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previously configured settings
df = spark.readStream \
    .format("eventhubs") \
    .options(**ehConf) \
    .load()

# Displaying stream: Show the incoming streaming data for visualization and debugging purposes
df.display()

```

The notebook is currently at step 5, with the status "Connecting". A message "Stream initializing..." is visible. Below the code, the output pane shows the command "# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previously configured settings".

I can see that no data is generated in the stream as I haven't generated any new data. The stream is a live stream, so it won't show the historical data I have generated earlier.

Now, I created one more event in azure event hub to check if i can see live stream here in the databricks notebook

The screenshot shows the Microsoft Azure portal with the URL <https://portal.azure.com/#@priyavur1602@gmail.onmicrosoft.com/resource/subscriptions/c18ffbb2-e46e-476b-aa92-aad49b43bbb3/>. The page title is "realtime-streaming | Data Explorer (preview)". The left sidebar shows the "Data Explorer (preview)" section selected. In the main area, there is a "Send events" button and a message stating "Completed sending 1 events" and "Sent 1 events to eventhub 'streaming'". Below this, there is a search bar, a dropdown for "Event Hub" set to "streaming", and a "Create a new Event Hub" button. A "No data to display" message is present with the instruction "Select 'Send events' to transmit data to the event hub or choose 'View events' to inspect data that has already been sent."

Now, I can see the live stream in databricks notebook as below

The screenshot shows a Databricks notebook titled "Untitled Notebook 2024-11-24 08:21:05" in Python. The left sidebar shows the "Workspace" section selected. The main notebook area displays a table named "display_query_1" with one row of data. The data is a long string of characters starting with "ew0KICAgICJ0ZW1wZXJhdHVyZSI6IDwLA0KICAgICl0dW1pZGl0esI6IDYwLA0KICAgICJ3aW5kU3BIZWQjOiaMCwNCiAgICAid2luZERp...". The notebook also contains a comment at the bottom: "# Reading stream: Load data from Azure Event Hub into DataFrame 'df' using the previously configured settings".

When I am using the readstream method, the data generated is temporary, it won't persist in databricks environment. The data will build up in the stream while the stream is active.

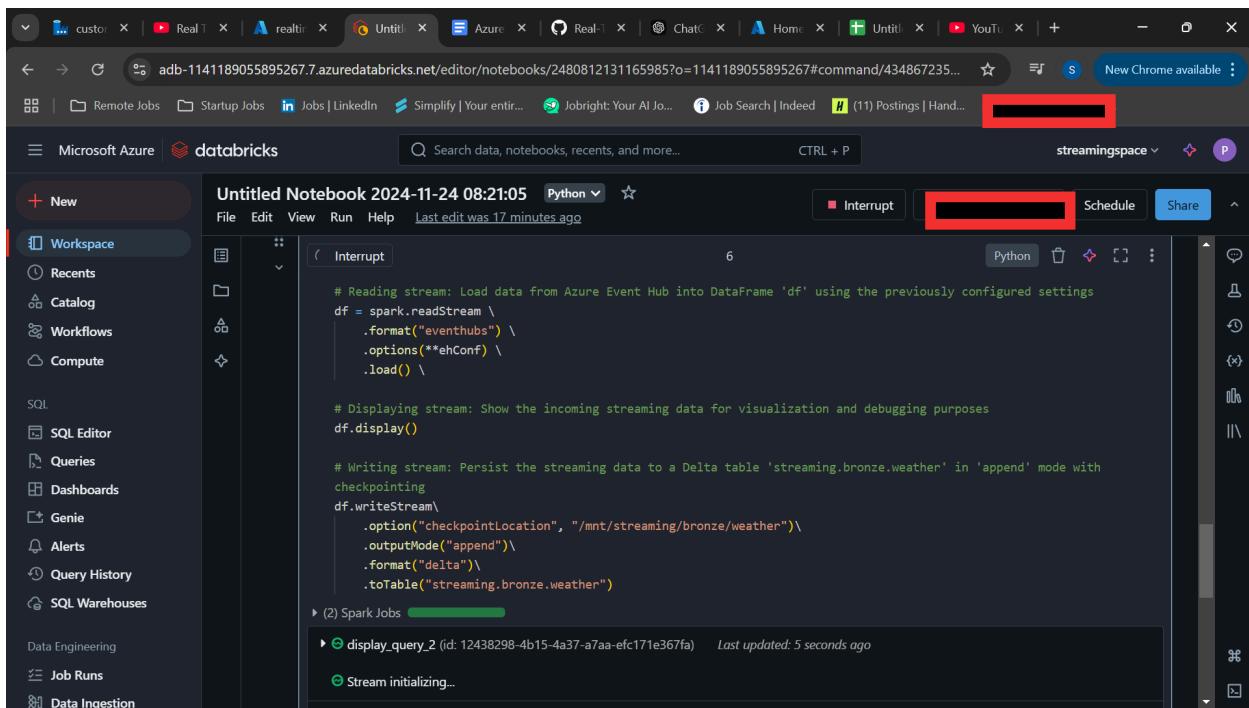
To persist the data, I wrote to storage using the writestream method using the below code

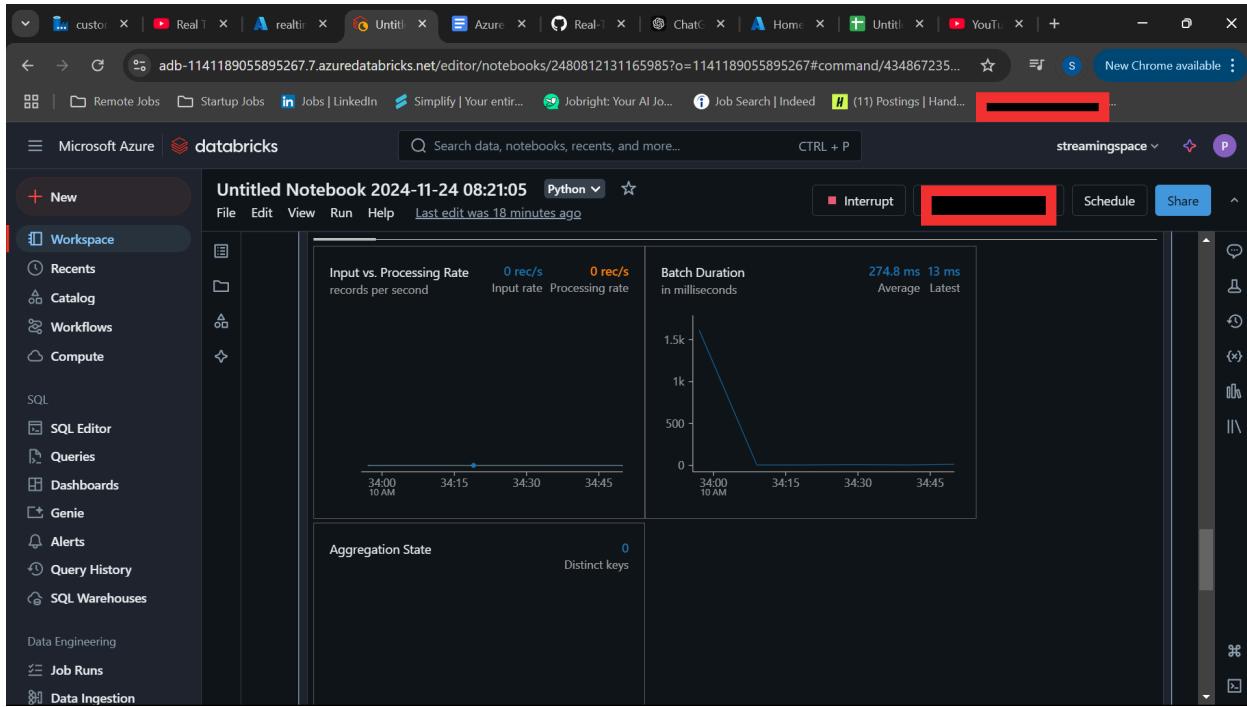
```
# Writing stream: Persist the streaming data to a Delta table 'streaming.bronze.weather' in
'append' mode with checkpointing
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/bronze/weather")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.bronze.weather")
```

I specified a checkpoint location in the databricks file system (DBFS) to store the state and progress of the streaming job.

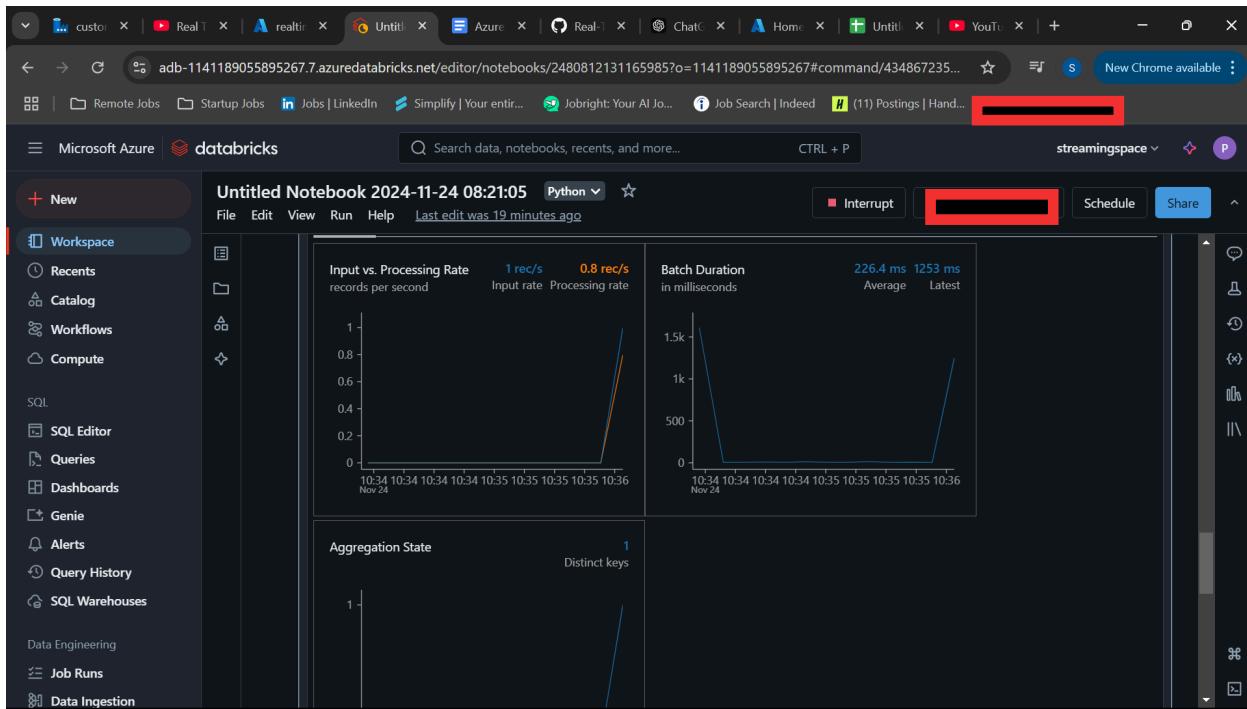
There are different output modes such as “Complete”, “Append”, “Update”

I can see that the stream is initialized





As soon as I generate event , I can see the below image



I can also see the table “weather” is created in the bronze layer with the sample data generated as shown below

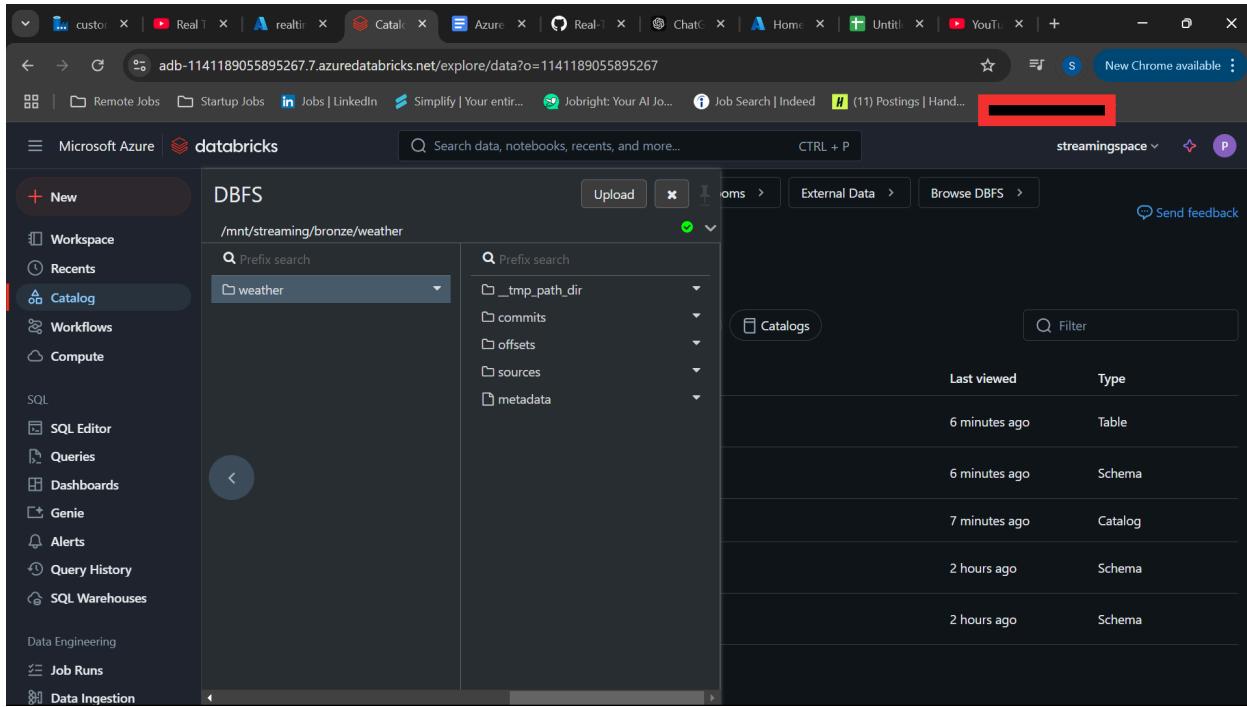
The screenshot shows the Databricks Catalog Explorer interface. On the left sidebar, under the Catalog section, there is a tree view of databases and tables. Under the 'bronze' database, the 'weather' table is selected. The main panel displays the 'Sample Data' tab for the 'weather' table, showing two rows of data. The first row has a timestamp of 2023-01-01T00:00:00Z and a body temperature of 15.5. The second row has a timestamp of 2023-01-01T01:00:00Z and a body temperature of 16.0.

Now, even if I terminate the stream, the data still persists as I have written the stream to weather table in the bronze layer

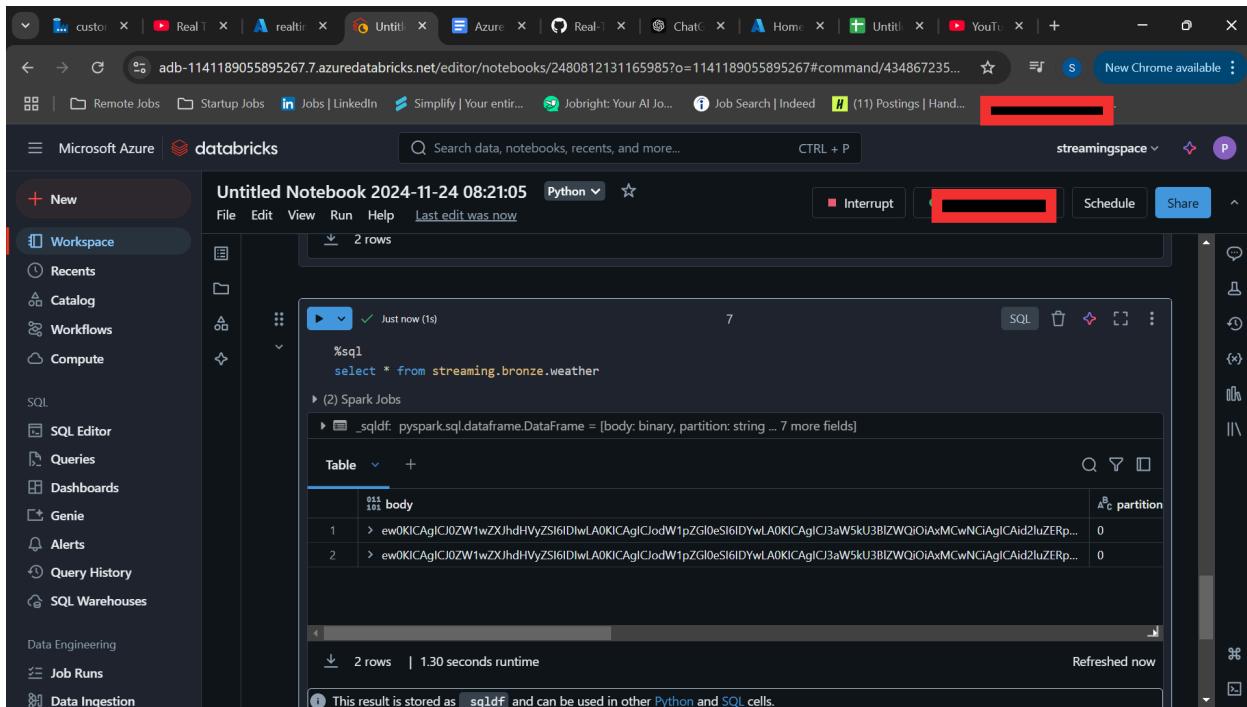
I enabled the DBFS file browser to see the checkpointing information. I can see the checkpointing information as shown in the image below. This keeps track of the state and progress of the stream

The screenshot shows the Databricks DBFS file browser. The left sidebar shows the DBFS structure with 'FileStore' and 'local_disk0' mounted at '/mnt'. The right panel shows the contents of the 'streaming' directory under '/mnt'. A table on the right lists recently viewed items: a catalog last viewed 6 minutes ago, a schema last viewed 6 minutes ago, a schema last viewed 6 minutes ago, a schema last viewed 2 hours ago, and a schema last viewed 2 hours ago.

Last viewed	Type
6 minutes ago	Table
6 minutes ago	Schema
6 minutes ago	Catalog
2 hours ago	Schema
2 hours ago	Schema



I can query the weather table in the bronze layer as shown below



I created the silver layer by initiating the stream from the bronze layer and writing the stream to the checkpoint location. I used the below code to create the silver layer

```
# Defining the schema for the JSON object

json_schema = StructType([
    StructField("temperature", IntegerType()),
```

```

StructField("humidity", IntegerType()),
StructField("windSpeed", IntegerType()),
StructField("windDirection", StringType()),
StructField("precipitation", IntegerType()),
StructField("conditions", StringType())
])

# Reading and Transforming: Load streaming data from the
# 'streaming.bronze.weather' Delta table, cast 'body' to string, parse JSON, and
# select specific fields
df = spark.readStream\
    .format("delta")\
    .table("streaming.bronze.weather")\
    .withColumn("body", col("body").cast("string"))\
    .withColumn("body", from_json(col("body"), json_schema))\
    .select("body.temperature", "body.humidity", "body.windSpeed",
"body.windDirection", "body.precipitation", "body.conditions",
col("enqueuedTime").alias('timestamp'))

# Displaying stream: Visualize the transformed data in the DataFrame for
# verification and analysis
df.display()

# Writing stream: Save the transformed data to the 'streaming.silver.weather'
# Delta table in 'append' mode with checkpointing for data reliability
df.writeStream\
    .option("checkpointLocation", "/mnt/streaming/silver/weather")\
    .outputMode("append")\
    .format("delta")\
    .toTable("streaming.silver.weather")

```

Code Explanation:

Define JSON Schema:

- A schema (`json_schema`) is defined using `StructType` to describe the structure of the incoming JSON objects. It includes fields like `temperature`, `humidity`, `windSpeed`, etc., with appropriate data types.

Reading Streaming Data:

- A streaming DataFrame (`df`) is created by reading data from the Delta table `streaming.bronze.weather`.
- The `body` column (which contains JSON data) is cast to a string and parsed into a structured format using the `from_json` function, applying the `json_schema`.

- Specific fields (`temperature`, `humidity`, `windSpeed`, etc.) are extracted from the parsed JSON along with the `enqueuedTime` (renamed as `timestamp`).

Displaying the Stream:

- The `df.display()` function is used to visualize the real-time streaming data for verification and analysis in an interactive notebook environment (e.g., Databricks).

Writing Transformed Data to Silver Table:

- The transformed data is written to the Delta table `streaming.silver.weather` in "append" mode, meaning new data is continuously added to the table.
- A checkpoint directory (`/mnt/streaming/silver/weather`) is specified to ensure data reliability and fault tolerance during the streaming process.

I can see that the stream is initialized and all the data is converted to tabular format from a nested JSON

```

11 # Displaying stream: Visualize the transformed data in the DataFrame for verification and analysis
12 df.display()
13
14 # Writing stream: Save the transformed data to the 'streaming.silver.weather' Delta table in 'append' mode with
15 # checkpointing for data reliability
16 df.writeStream
17 .option("checkpointLocation", "/mnt/streaming/silver/weather")
18 .outputMode("append")
19 .format("delta")
20 .toTable("streaming.silver.weather")

```

(2) Spark Jobs

Stream initializing...
Stream initializing...

df. pyspark.sql.dataframe.DataFrame = [temperature: integer, humidity: integer ... 5 more fields]
<pyspark.sql.streaming.query.StreamingQuery at 0x7f67551dd10>

[Shift+Enter] to run and move to next cell
[Ctrl+Shift+P] to open the command palette

A screenshot of the Azure Databricks notebook interface. The left sidebar shows various Databricks services like Workspace, Recents, Catalog, Workflows, Compute, SQL, and Data Ingestion. The main area displays an 'Untitled Notebook' titled 'Python'. The code cell contains Python code for writing a stream to a Delta table:`# Writing stream: save the transformed data to the 'streaming.silver.weather' Delta table in 'append' mode with checkpointing for data reliability
df.writeStream\
 .option("checkpointLocation", "/mnt/streaming/silver/weather")\
 .outputMode("append")\
 .format("delta")\
 .toTable("streaming.silver.weather")`

The notebook also lists two Spark Jobs: 'display_query_1' and '5b8a4bf1-94cd-425a-b30b-21853794bc4b'. Below the code, a table view shows historical weather data with columns: temperature, humidity, windSpeed, windDirection, precipitation, conditions, and a timestamp.

	temperature	humidity	windSpeed	windDirection	precipitation	conditions	timestamp
1	20	60	10	NW	0	Partly Cloudy	2024-11-24T08:21:05Z
2	20	60	10	NW	0	Partly Cloudy	2024-11-24T08:21:05Z
3	100	80	30	SC	50	Cloudy	2024-11-24T08:21:05Z

As the spark structured streaming processes the historical data and real time with exactly once semantics, I can see the historical records too in the output. If I generate a new event now, the output will be changed as below

A screenshot of the Azure Databricks notebook interface, identical to the first one but with a new event added. The table view now shows five rows of data, including the three historical entries and two new ones generated by the streaming process.

	temperature	humidity	windSpeed	windDirection	precipitation	conditions	timestamp
1	20	60	10	NW	0	Partly Cloudy	2024-11-24T08:21:05Z
2	20	60	10	NW	0	Partly Cloudy	2024-11-24T08:21:05Z
3	10	30	10	SE	0	Partly Cloudy	2024-11-24T08:21:05Z
4	100	80	30	SC	50	Cloudy	2024-11-24T08:21:05Z

I can also view the sample weather data in the silver layer as below

The screenshot shows the Microsoft Azure Databricks interface. The left sidebar is the navigation menu with options like Workspace, Recents, Catalog (which is selected), Workflows, Compute, SQL, and Data Engineering. The main area is titled 'Catalog' and shows a tree view of databases: 'streamingspace', 'system', 'streaming', 'bronze', 'default', 'gold', 'information_schema', and 'silver'. Under 'silver', there is a folder named 'weather'. A 'Sample Data' tab is open, displaying a table with columns: temperature, humidity, windSpeed, windDirection, and precipitation. The table has 5 rows of sample data.

The data is persisted in the databricks environment. I can see the checkpointing information from DBFS browser as below

The screenshot shows the Microsoft Azure Databricks interface with the DBFS browser. The left sidebar is the same as the previous screenshot. The main area shows the DBFS browser with a path '/mnt/streaming/silver/weather'. A 'Prefix search' dropdown is set to 'weather'. On the right, there is a 'Catalogs' section with a table showing 'Last viewed' and 'Type' for several items: a table 4 minutes ago, a schema 4 minutes ago, a table 5 hours ago, a schema 5 hours ago, a catalog 5 hours ago, and a schema 7 hours ago.

For the gold layer, I aggregated the data into 5 minute intervals using a timestamp column. I created the gold layer using the below code

```
# Aggregating Stream: Read from 'streaming.silver.weather', apply watermarking and windowing, and calculate average weather metrics
```

```

df = spark.readStream\
    .format("delta") \
    .table("streaming.silver.weather") \
    .withWatermark("timestamp", "5 minutes") \
    .groupBy(window("timestamp", "5 minutes")) \
    .agg(avg("temperature").alias('temperature'),
        avg("humidity").alias('humidity'), avg("windSpeed").alias('windSpeed'),
        avg("precipitation").alias('precipitation')) \
    .select('window.start', 'window.end', 'temperature', 'humidity',
    'windSpeed', 'precipitation')

# Displaying Aggregated Stream: Visualize aggregated data for insights into weather trends
df.display()

# Writing Aggregated Stream: Store the aggregated data in 'streaming.gold.weather_aggregated' with checkpointing for data integrity
df.writeStream \
    .option("checkpointLocation", "/mnt/streaming/weather_summary") \
    .outputMode("append") \
    .format("delta") \
    .toTable("streaming.gold.weather_summary")

```

Code Explanation:

1. Reading the Silver Table Stream:

- A streaming DataFrame (`df`) is created by reading data from the Delta table `streaming.silver.weather`.
- **Watermarking** is applied on the `timestamp` column with a 5-minute threshold to handle late-arriving data, ensuring efficient memory usage and consistency.

2. Windowing and Aggregation:

- **Windowing:** The `window` function groups data into 5-minute intervals based on the `timestamp` column.
- **Aggregation:** The `avg` function calculates the average of weather metrics (`temperature`, `humidity`, `windSpeed`, `precipitation`) for each window.
- The resulting DataFrame includes:
 - `window.start` and `window.end` to indicate the time range for each aggregation.
 - Average values of the specified weather metrics.

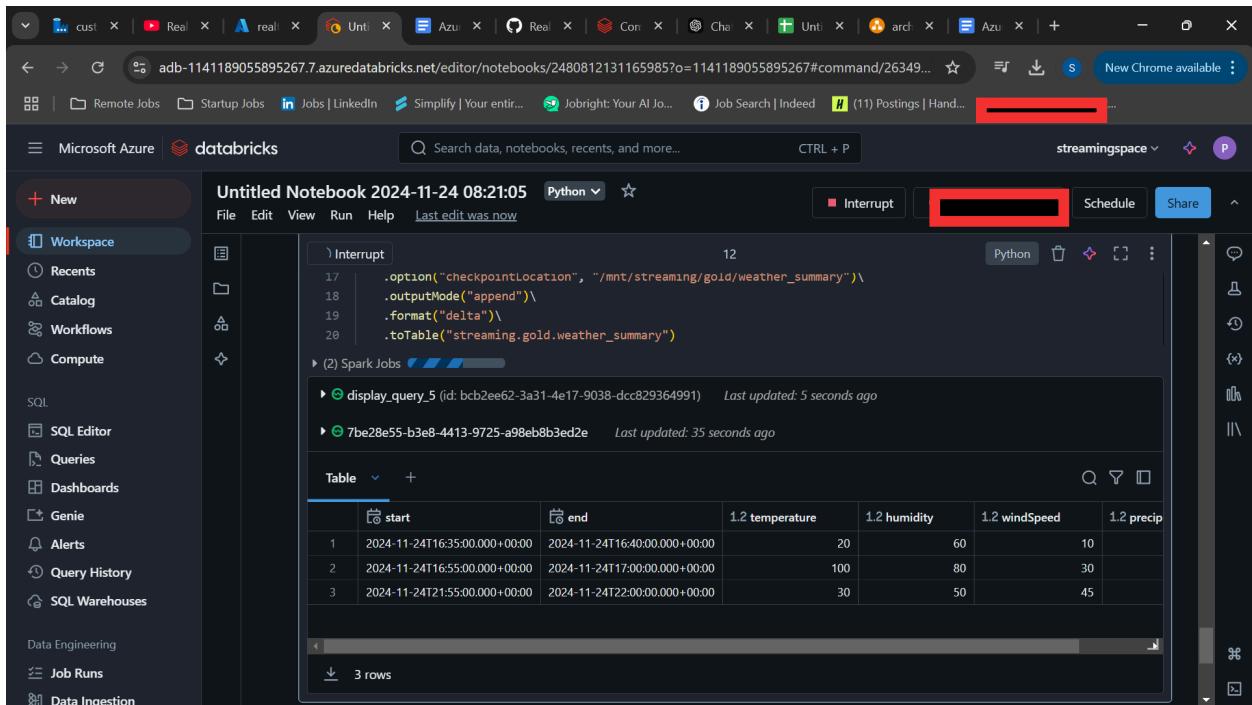
3. Displaying Aggregated Data:

- `df.display()` visualizes the aggregated streaming data in an interactive notebook environment, enabling real-time monitoring of weather trends.

4. Writing Aggregated Data to Gold Table:

- The aggregated data is written to the Delta table `streaming.gold.weather_summary` in "append" mode, continuously appending new summaries as the stream progresses.
- A checkpoint directory (`/mnt/streaming/weather_summary`) ensures data integrity and fault tolerance during the streaming process.

I can see the stream is initialized for gold layer as below



The screenshot shows a Databricks notebook titled "Untitled Notebook 2024-11-24 08:21:05". The code cell contains the following Python code:

```

17 .option("checkpointLocation", "/mnt/streaming/gold/weather_summary")
18 .outputMode("append")
19 .format("delta")
20 .toTable("streaming.gold.weather_summary")

```

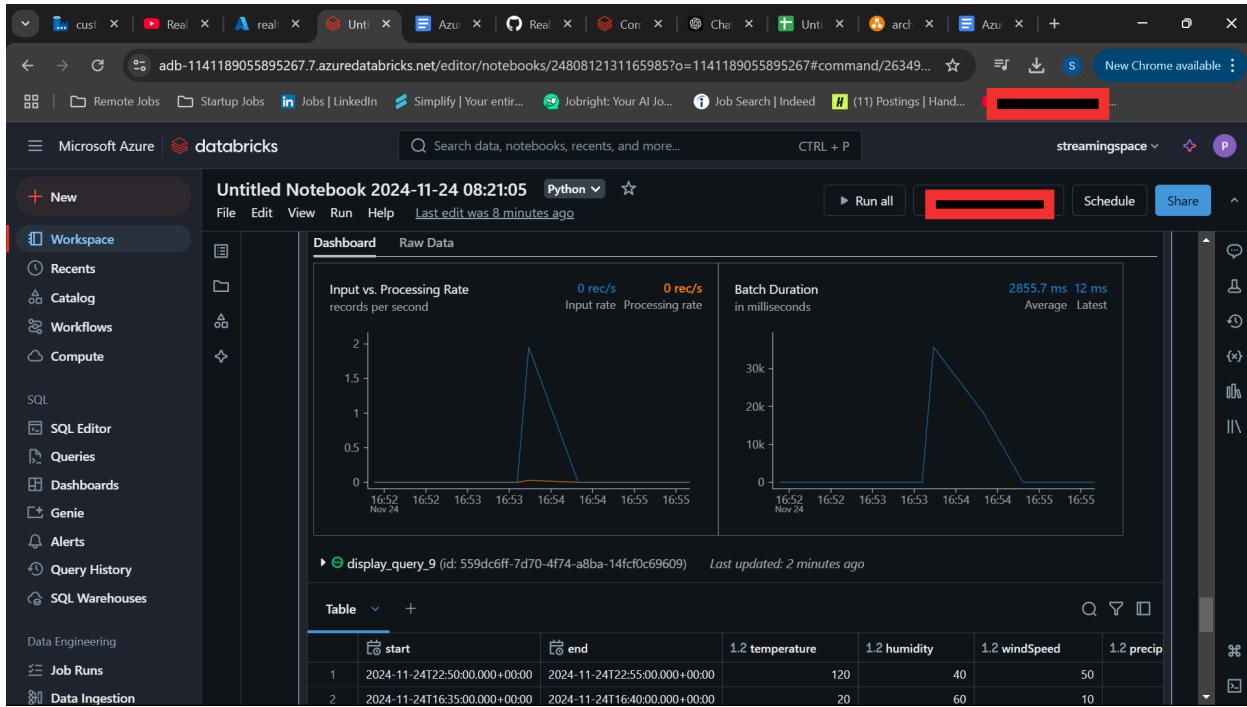
Below the code cell, there are two Spark Jobs listed:

- `display_query_5` (id: bcb2ee62-3a31-4e17-9038-dcc829364991) - Last updated: 5 seconds ago
- `7be28e55-b3e8-4413-9725-a98eb8b3ed2e` - Last updated: 35 seconds ago

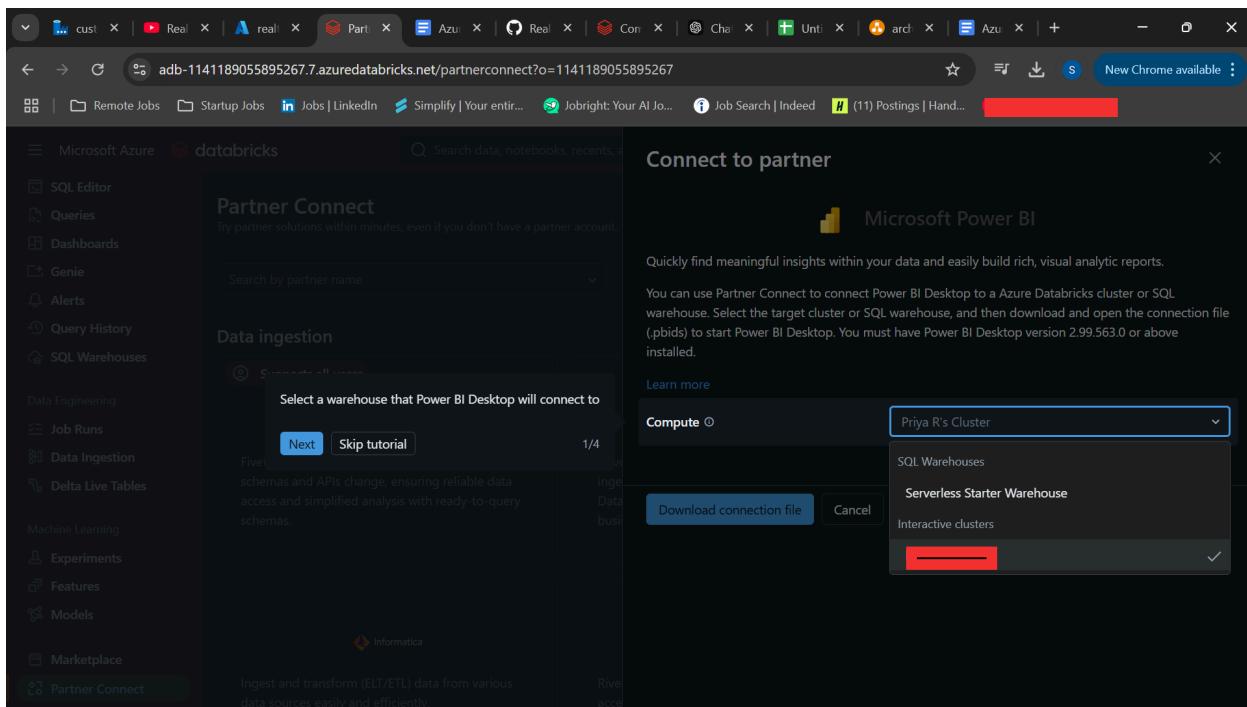
At the bottom of the notebook, a table is displayed with the following data:

	start	end	1.2 temperature	1.2 humidity	1.2 windSpeed	1.2 precip
1	2024-11-24T16:35:00.000+00:00	2024-11-24T16:40:00.000+00:00	20	60	10	
2	2024-11-24T16:55:00.000+00:00	2024-11-24T17:00:00.000+00:00	100	80	30	
3	2024-11-24T21:55:00.000+00:00	2024-11-24T22:00:00.000+00:00	30	50	45	

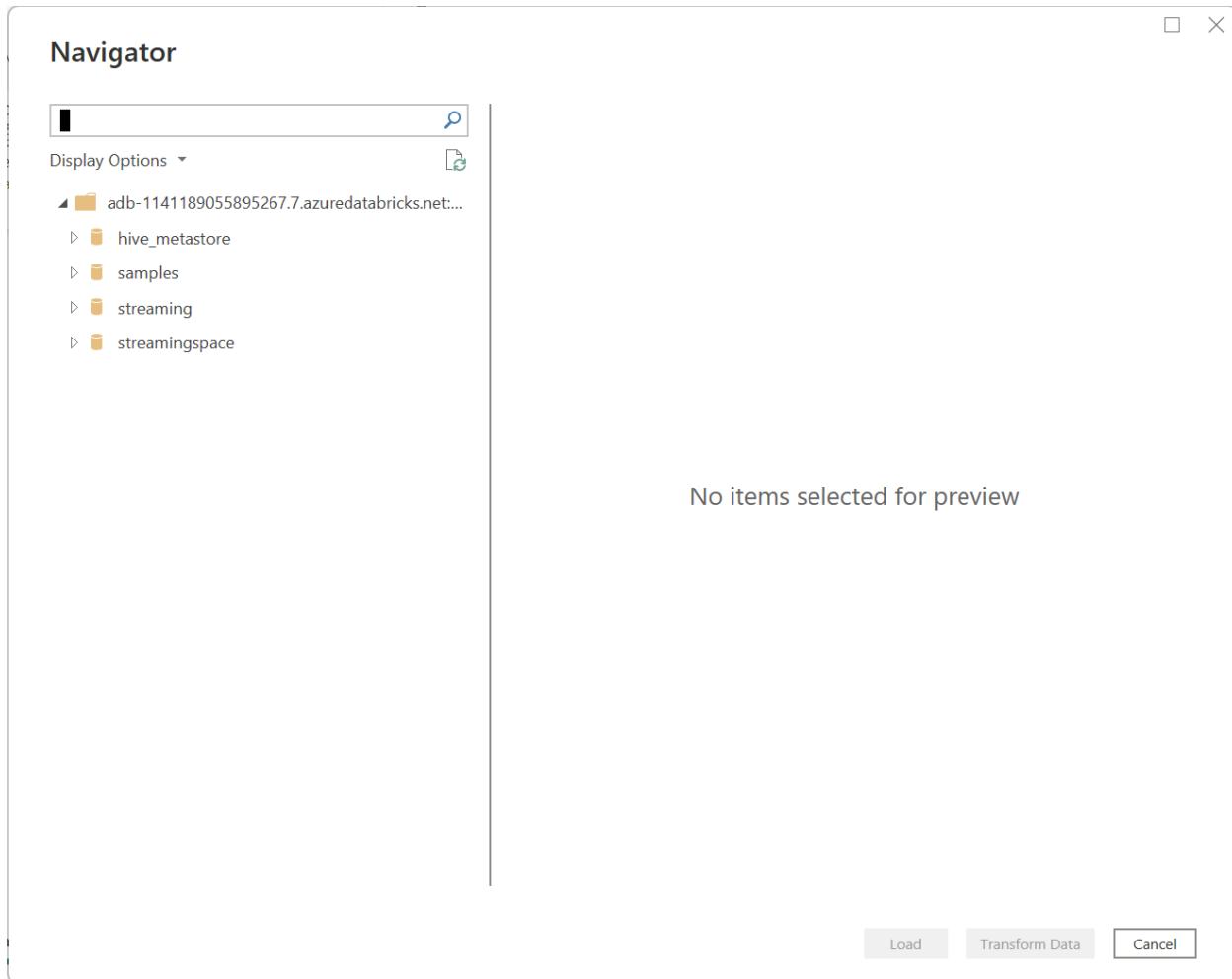
The table shows 3 rows of data.



Now, I connected the gold layer to Power BI inorder to monitor the streaming on a dashboard in real time. For this, I used the “Partner Connect” option as below



I signed in through Azure Active directory and I can see the data in Power BI as below



Hare, I selected the silver layer to observe the changes in real time as the gold layer has a watermark for 5 minutes and it will take longer to observe the changes.

The screenshot shows the Power BI Navigator interface. On the left, there is a navigation pane with a search bar and a "Display Options" dropdown. Below these are several data sources listed as folders:

- adb-1141189055895267.7.azuredatabricks.net... (with a minus sign)
- hive_metastore [1] (with a minus sign)
- default (with a plus sign)
- samples (with a plus sign)
- streaming [4] (with a minus sign)
 - bronze (with a plus sign)
 - default (with a plus sign)
 - gold (with a plus sign)
 - silver [1] (with a minus sign)
 - weather (with a checkmark and a grid icon)
- streamingspace (with a plus sign)

The main area displays a table titled "weather" with the following data:

temperature	humidity	windSpeed	windDirection	precipitation	con
20	60	10	NW	0	Par
20	60	10	NW	0	Par
100	80	30	SC	50	Clo
10	30	10	SE	0	Par
50	70	80	NW	0	Par
120	40	50	SE	0	Sun

At the bottom right of the main area are three buttons: "Load" (green), "Transform Data" (white), and "Cancel" (white).

Here, I was using the “DirectQuery” mode in power BI, so it took time to load the data as the data is not in the power BI’s internal memory. I switched to import mode later to see the changes faster. Also, to see the streaming data change in the power BI dashboard, I refreshed the query and I can see the dashboard below.

Untitled - Power BI Desktop

File Home Insert Modeling View Optimize Help Format Data / Drill

Cut Copy Paste Format painter Clipboard Get data from Excel OneLake workbook data hub Data Server Enter Data Refresh Recent sources Transform data from Query Refresh New visual Text box More visual Insert Calculations New visual calculation New measure Quick measure Sensitivity Share Publish Copilot

Visualizations Data

Build visual

Filters

Search

weather

- conditions
- Sum of humidity
- Sum of precipitation
- Sum of temperature
- Year
- Quarter
- Month
- Day
- windDirection
- Sum of windSpeed

Columns

conditions	Sum of humidity	Sum of precipitation	Sum of temperature	Year	Quarter	Month	Day	windDirection	Sum of windSpeed
Partly Cloudy	190	0	90	2024	Qtr 4	November	24	NW	100
Cloudy	80	50	100	2024	Qtr 4	November	24	SC	30
Partly Sunny	70	20	80	2024	Qtr 4	November	24	West	40
Sunny	40	0	120	2024	Qtr 4	November	24	SE	50
Partly Cloudy	30	0	10	2024	Qtr 4	November	24	SE	10
Total	410	70	400						230

Page 1 +

Page 1 of 1

A screenshot of the Power BI Desktop interface. The main area shows a data grid with columns: conditions, Sum of humidity, Sum of precipitation, Sum of temperature, Year, Quarter, Month, Day, windDirection, and Sum of windSpeed. The data includes rows for Partly Cloudy, Cloudy, Partly Sunny, Sunny, and Partly Cloudy, with totals at the bottom. The Data pane on the right displays the data model with various columns selected. The top ribbon shows the Home tab is selected. The status bar at the bottom indicates Page 1 of 1.