



FINAL PROJECT

**Processing live streaming data via event hub
and Databricks.**

EMPLOYEE ID: 2320213

NAME: MAJJI VIJAY VAMSI

CSDAIA24AZ005

INTRODUCTION

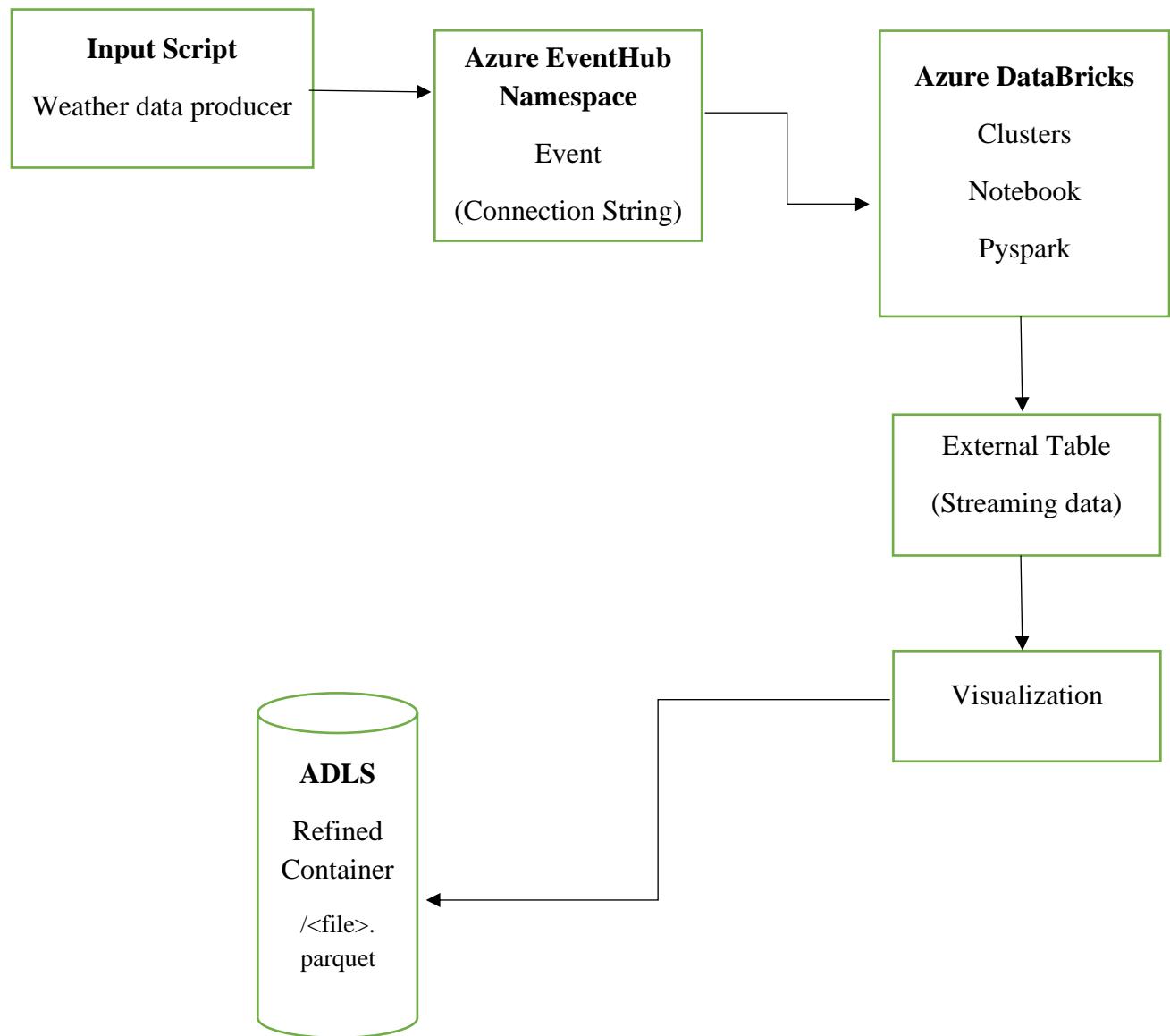
Azure EventHub and Azure DataBricks are the tools were used in the project. With this use of Azure components I learnt how to fetch the live streaming data of weather report from the Azure Event hub and convert that data into to the table in the Azure DataBricks by using the commands and the functions present in the pyspark and I loaded that streaming data in the container which present in the Azure Data Lake Storage(ADLS). By performing this operations I learnt how to get the live streaming data and how to use it based on the business requirement.

OBJECTIVE

The objective of the project is to fetch the live streaming data by using the python script and loading that live streaming data into the Azure DataBricks by using the connection string in the EventHub. After fetching the data in EventHub, giving the output of the EventHub to the Azure Databricks and by making use of pyspark commands in the notebook connecting the both EventHub and ADB. Finally fetching the data from the EventHub and converting the data received into the table format and loading that data into the Adls(refined container) .

PROJECT REQUIREMENT:

BLOCK DIAGRAM:



STEP 1: Create an event-hub namespace and event-hub (namespace-vamsi-event) (eventhub-majji).

The screenshot shows the Microsoft Azure portal interface. The top navigation bar has tabs for 'vamsi-event - Microsoft Azure', 'eventhubs to DB 02 - Databrick...', 'print schema command in pysp...', and 'New Tab'. The user is signed in as '19B91A04C2@srkrec.ac.in' from 'SRKR ENGINEERING COLLEGE (S...)'.

The main page title is 'vamsi-event' under 'Event Hubs Namespace'. On the left, there's a sidebar with 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Generate data (preview)', 'Events', 'Settings' (with 'Shared access policies', 'Scale', 'Geo-Recovery', 'Encryption', 'Configuration'), and 'Configuration'.

The 'Essentials' section displays resource details:

- Resource group: finalproject
- Status: Active
- Location: East US
- Subscription: Azure for Students
- Subscription ID: 60b24090-d7f5-4c2e-b6e4-53fc04c72f96
- Host name: vamsi-event.servicebus.windows.net
- Tags: (edit), Add tags
- Created: Tuesday, April 2, 2024 at 15:38:14 GMT+5:30
- Updated: Tuesday, April 2, 2024 at 15:38:37 GMT+5:30
- Zone Redundancy: Enabled
- Pricing tier: Basic
- Throughput Units: 1 unit
- Auto-inflate throughput units: Not Supported
- Local Authentication: Enabled

At the bottom, there are links for 'NAMESPACE CONTENTS', 'KAFKA CLUSTER', and 'TIERED REDUNDANCY'. The status bar at the bottom right shows '8:18 PM 4/10/2024'.

STEP 2: Create a SAS policy to give a connection string to script to send the events to event-hub.

The screenshot shows the Microsoft Azure portal interface. The top navigation bar has tabs for 'SAS Policy: python - Microsoft /', 'eventhubs to DB 02 - Databrick...', 'print schema command in pysp...', and 'New Tab'. The user is signed in as '19B91A04C2@srkrec.ac.in' from 'SRKR ENGINEERING COLLEGE (S...)'.

The main page title is 'SAS Policy: python' under 'Event Hubs > vamsi-event > majji (vamsi-event/majji)'. On the left, there's a sidebar with 'Overview', 'Access control (IAM)', 'Diagnose and solve problems', 'Settings' (with 'Shared access policies', 'Configuration', 'Properties', 'Locks'), 'Entities' (with 'Consumer groups'), and 'Features' (with 'Capture').

The 'Shared access policies' section shows a table with one row:

Policy	Key
python	9qUnX990zzSzUqYFfgAOjoe8WZmxrmWE+AElhAtNgvQ=

The right pane shows the policy details for 'python':

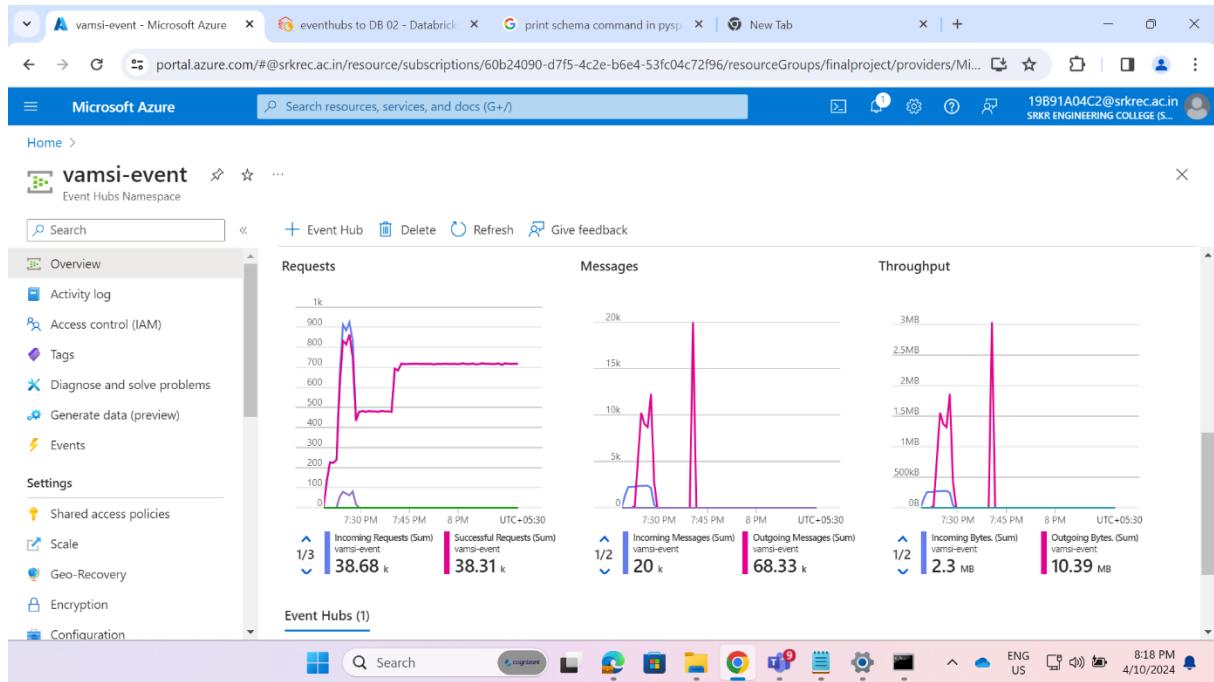
- Manage checkbox is checked.
- Send and Listen checkboxes are unchecked.
- Primary key: 9qUnX990zzSzUqYFfgAOjoe8WZmxrmWE+AElhAtNgvQ=
- Secondary key: ZjItoR3GaBkO0QPoGi5ZNdtJksE9LvoQj9+AElJCj2Vg=
- Connection string-primary key: Endpoint=sb://vamsi-event.servicebus.windows.net/SharedAccessKeyName=python...
- Connection string-secondary key: Endpoint=sb://vamsi-event.servicebus.windows.net/SharedAccessKeyName=python...
- SAS Policy ARM ID: /subscriptions/60b24090-d7f5-4c2e-b6e4-53fc04c72f96/resourceGroups/finalprojec...

The status bar at the bottom right shows '7:43 PM 4/14/2024'.

STEP 3: Install event hub libraries (pip install-eventhub) in cmd and run the script in command prompt (sending the events to event-hub).

```
Command Prompt - sendevent_eventhub_python_script.py
events processing {"id": 3, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 73, "humidity": 84}
events processing {"id": 1, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 90, "humidity": 94}
events processing {"id": 5, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 76, "humidity": 83}
events processing {"id": 2, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 78, "humidity": 87}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 75, "humidity": 82}
events processing {"id": 4, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 88, "humidity": 83}
events processing {"id": 8, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 85, "humidity": 89}
events processing {"id": 4, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 79, "humidity": 89}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:54.831455", "temperature": 93, "humidity": 84}
events processing {"id": 5, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 73, "humidity": 76}
events processing {"id": 2, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 98, "humidity": 96}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 70, "humidity": 74}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 71, "humidity": 99}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 86, "humidity": 78}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 82, "humidity": 73}
events processing {"id": 1, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 75, "humidity": 85}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 92, "humidity": 79}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 74, "humidity": 89}
events processing {"id": 8, "timestamp": "2024-04-14 14:41:55.083710", "temperature": 71, "humidity": 72}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 71, "humidity": 75}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 98, "humidity": 79}
events processing {"id": 4, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 77, "humidity": 80}
events processing {"id": 5, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 93, "humidity": 99}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 99, "humidity": 96}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 86, "humidity": 99}
events processing {"id": 4, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 85, "humidity": 87}
events processing {"id": 3, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 72, "humidity": 86}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 94, "humidity": 81}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.334788", "temperature": 82, "humidity": 81}
events processing {"id": 10, "timestamp": "2024-04-14 14:41:55.570254", "temperature": 79, "humidity": 72}
events processing {"id": 10, "timestamp": "2024-04-14 14:41:55.570254", "temperature": 72, "humidity": 83}
events processing {"id": 2, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 78, "humidity": 94}
events processing {"id": 1, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 76, "humidity": 83}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 85, "humidity": 88}
events processing {"id": 6, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 85, "humidity": 90}
events processing {"id": 4, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 71, "humidity": 80}
events processing {"id": 9, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 72, "humidity": 83}
events processing {"id": 7, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 92, "humidity": 96}
events processing {"id": 8, "timestamp": "2024-04-14 14:41:55.585968", "temperature": 100, "humidity": 78}
```

STEP 4: Receiving events in event-hub in graphical presentation.



STEP 5: Create a data Bricks resource (dbmatch) and Launch Workspace.

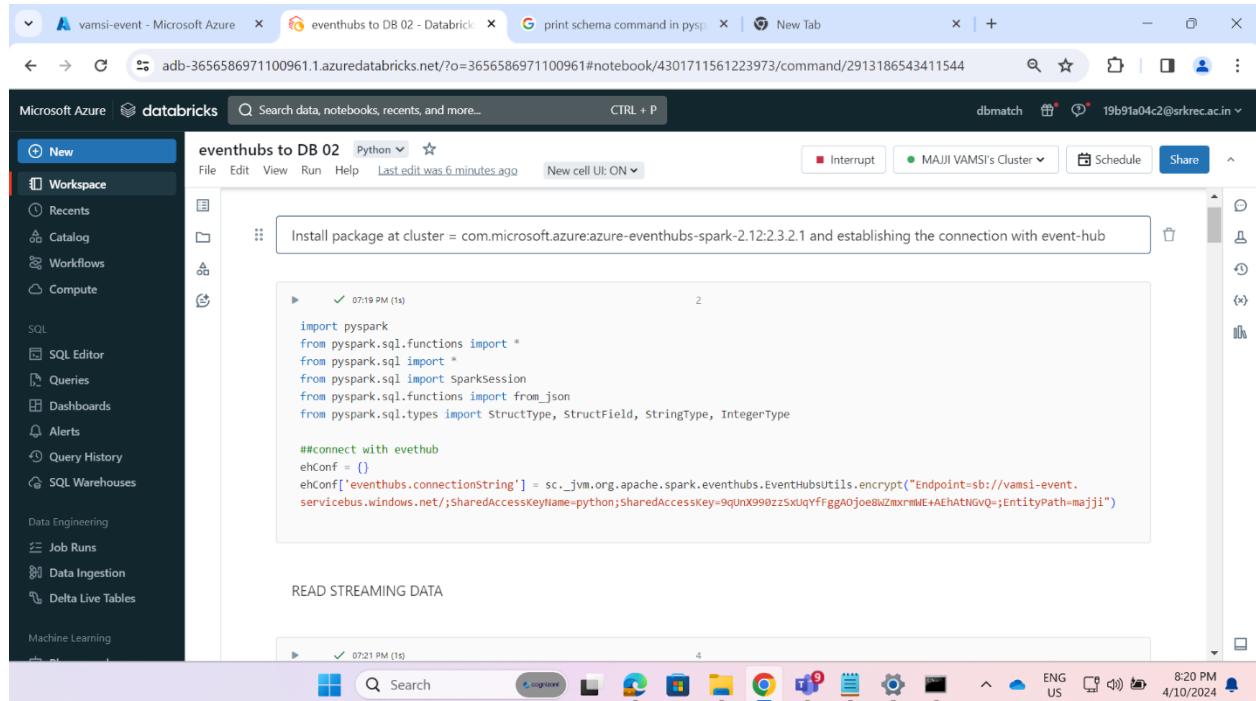
The screenshot shows the Microsoft Azure portal interface. At the top, there are three browser tabs: "Azure Databricks - Microsoft Az", "eventhubs to DB 02 - Databrick", and "print schema command in pysp". The main content area is titled "Microsoft Azure" and "Azure Databricks". A search bar at the top says "Search resources, services, and docs (G+/-)". On the right, a user profile is shown with the email "19b91a04c2@srkrec.ac.in" and "SRKR ENGINEERING COLLEGE (S)". Below the title, there are buttons for "+ Create", "Manage view", "Refresh", "Export to CSV", "Open query", and "Assign tags". A filter bar includes "Filter for any field...", "Subscription equals all", "Resource group equals all", "Location equals all", and "Add filter". The results table shows one record: "dbmatch" (Type: Azure Databricks Service, Resource group: databricks-class, Location: East US, Subscription: Azure for Students). The table has columns: Name, Type, Resource group, Location, and Subscription. At the bottom, there are navigation buttons for "< Previous", "Page 1 of 1", and "Next >".

STEP 6: In data bricks create a cluster (MAJJI VAMSI). Install libraries to connect azure event hub to spark (com.microsoft.azure:azure-eventhubs-spark_2.12:2.3.22).

The screenshot shows the Azure Databricks workspace. The left sidebar has sections like "New", "Workspace", "Recents", "Catalog", "Workflows", "Compute" (selected), "SQL", "SQL Editor", "Queries", "Dashboards", "Alerts", "Query History", "SQL Warehouses", "Data Engineering", "Job Runs", "Data Ingestion", "Delta Live Tables", and "Machine Learning". The main content area is titled "MAJJI VAMSI's Cluster" and shows "Free trial ends in 7 days. Upgrade to Premium in Azure Portal". It has tabs for "Configuration", "Notebooks (2)", "Libraries" (selected), "Event log", "Spark UI", "Driver logs", "Metrics", "Apps", and "Spark compute UI - Master". The "Libraries" tab shows a table with one row: "Status" (green checkmark), "Name" (com.microsoft.azure:azure-eventhubs-spark_2.12:2.3.22), "Type" (Maven), and "Source" (-). Buttons for "More", "Terminate", "Edit", "Uninstall", and "Install new" are available. The bottom navigation bar shows "Search", "CTRL + P", and the date/time "8:19 PM 4/10/2024".

STEP 7: Creating a Notebook as eventhubs to DB 02.

Create a cell in the notebook Import packages to work with pyspark and sql.



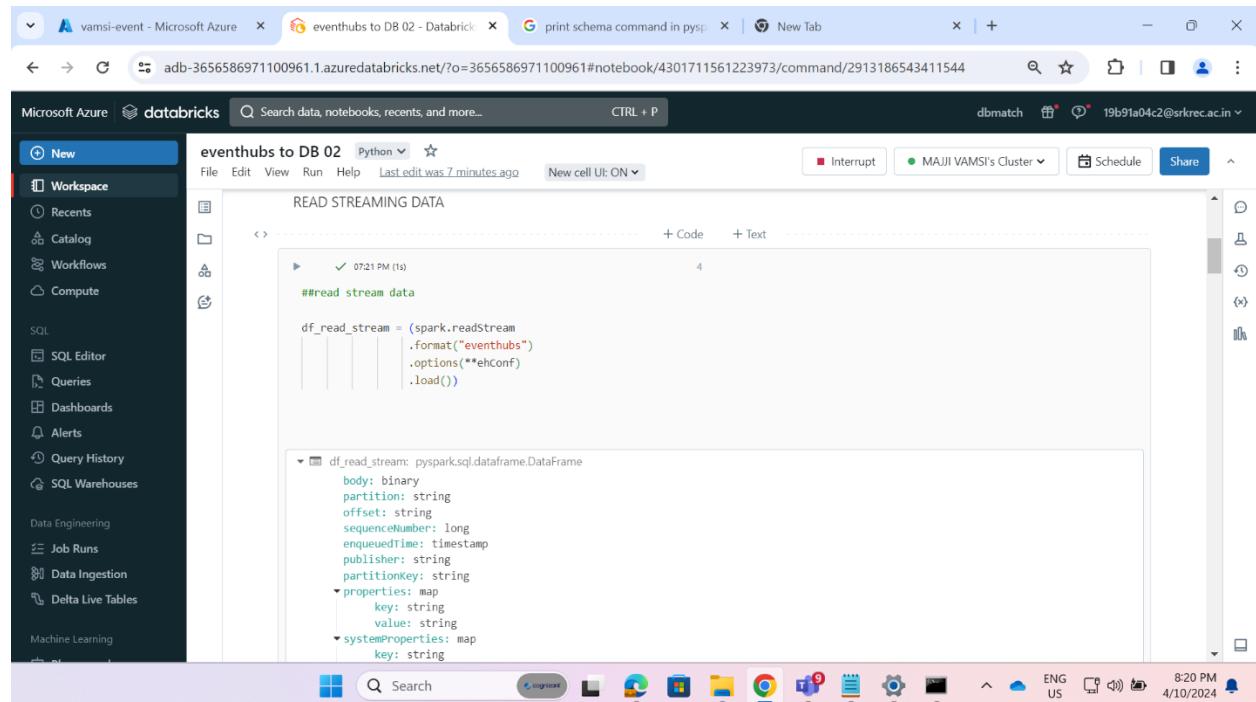
```
Install package at cluster = com.microsoft.azure.eventhubs-spark-2.12:2.3.2.1 and establishing the connection with event-hub
```

```
import pyspark
from pyspark.sql.functions import *
from pyspark.sql import *
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

##connect with eventhub
ehConf = {}
ehConf['eventhubs.connectionString'] = sc._jvm.org.apache.spark.eventhubs.EventHubsUtils.encrypt("Endpoint=sb://vamsi-event.servicebus.windows.net/;SharedAccessKeyName=python;SharedAccessKey=9qunX990zzSxUqYFrGgAOjoe8WZmxrmME+AElAtNGvQ;EntityPath=majji")
```

READ STREAMING DATA

STEP 8: Reading the streaming data form the EventHub.

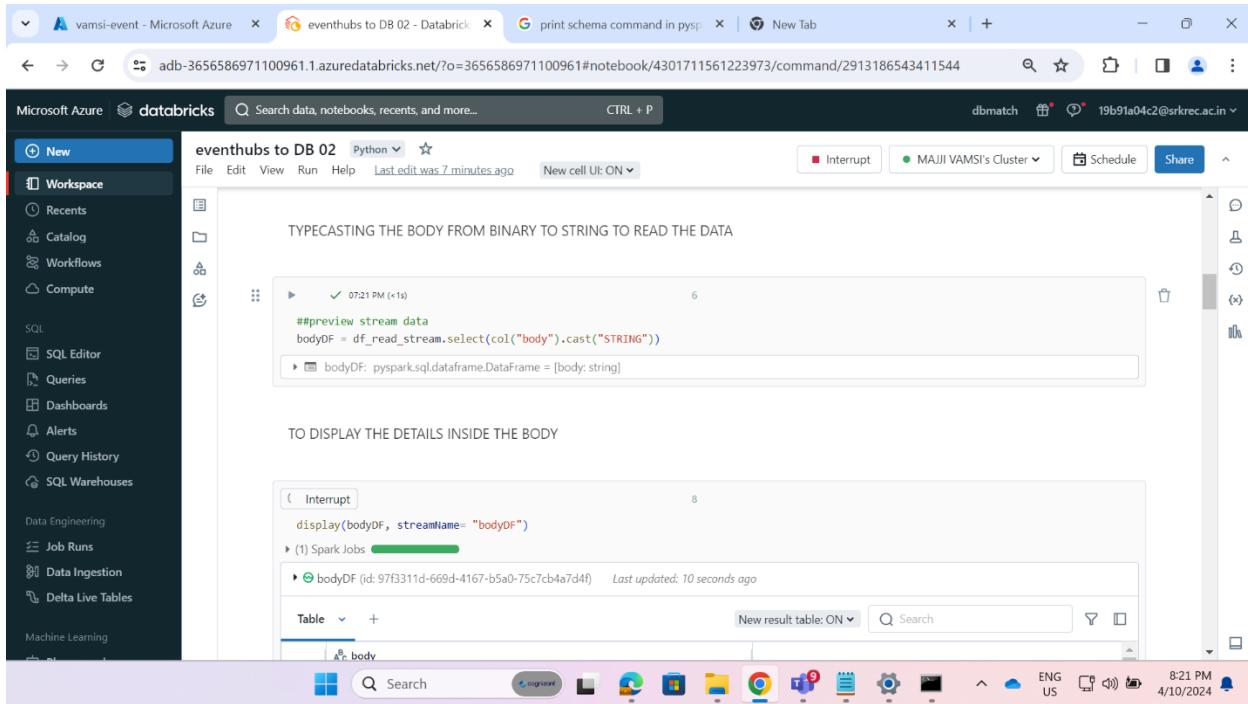


```
##read stream data
```

```
df_read_stream = (spark.readStream
                   .format("eventhubs")
                   .options(**ehConf)
                   .load())
```

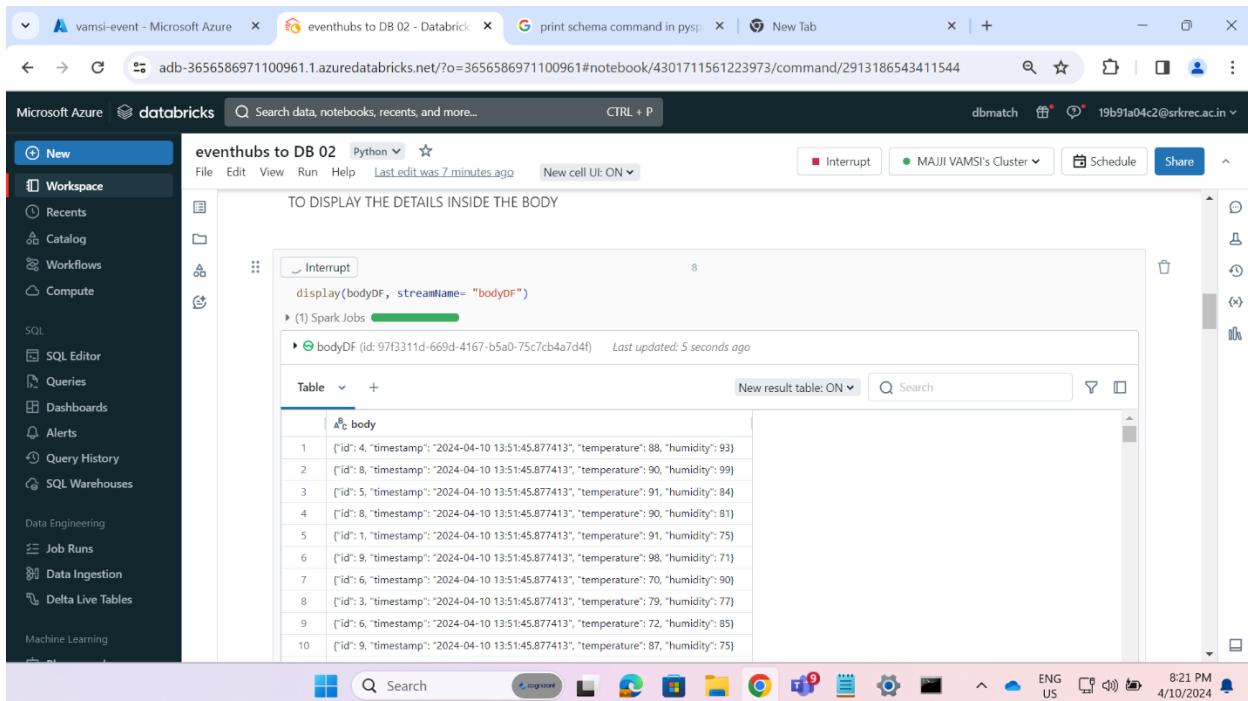
```
df_read_stream: pyspark.sql.dataframe.DataFrame
  body: binary
  partition: string
  offset: string
  sequenceNumber: long
  enqueueTime: timestamp
  publisher: string
  partitionKey: string
  properties: map
    key: string
    value: string
  systemProperties: map
    key: string
```

STEP 9: Preview the streaming data.



The screenshot shows a Databricks notebook titled "eventhubs to DB 02" in Python. The sidebar on the left lists various Databricks services: New, Workspace, Recents, Catalog, Workflows, Compute, SQL (SQL Editor, Queries, Dashboards), Alerts, Query History, SQL Warehouses, Data Engineering (Job Runs, Data Ingestion, Delta Live Tables), and Machine Learning. The main notebook area contains two code cells. The first cell, labeled "6", contains the command `##preview stream data` followed by `bodyDF = df_read_stream.select(col("body").cast("STRING"))`. The second cell, labeled "7", contains the command `display(bodyDF, streamName= "bodyDF")`. Below the cells, a table named "body" is displayed with 10 rows of data. The table has a header row with columns "id", "body". The data rows are as follows:

	body
1	{"id": 4, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 88, "humidity": 93}
2	{"id": 8, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 90, "humidity": 99}
3	{"id": 5, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 91, "humidity": 84}
4	{"id": 8, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 90, "humidity": 81}
5	{"id": 1, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 91, "humidity": 75}
6	{"id": 9, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 98, "humidity": 71}
7	{"id": 6, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 70, "humidity": 90}
8	{"id": 3, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 79, "humidity": 77}
9	{"id": 6, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 72, "humidity": 85}
10	{"id": 9, "timestamp": "2024-04-10 13:51:45.877413", "temperature": 87, "humidity": 75}



This screenshot shows the same Databricks notebook environment. The notebook title is "eventhubs to DB 02" in Python. The sidebar and notebook interface are identical to the previous screenshot. The notebook contains the same two code cells: cell 6 with the typecasting command and cell 7 with the display command. The table "body" is now fully visible, showing 10 rows of JSON data. The data is identical to the table shown in the previous screenshot.

STEP 10: creating schema for the dataframe.

The screenshot shows a Microsoft Azure Databricks notebook titled "eventhubs to DB 02" in Python. The code cell contains the following Python code:

```
from pyspark.sql.types import StructType, StructField, StringType, TimestampType
events_schema = StructType([
    StructField('id', StringType(), True),
    StructField('temperature', StringType(), True),
    StructField('humidity', StringType(), True),
    StructField('timestamp', TimestampType(), True)
])
# mapping the event schema to the decoded body where actual events are present
# we map the schema to df query and and convert the body field from binary to string type
decoded_df = df.read_stream.select(from_json(col('body').cast("string"), events_schema).alias("stringBody"))
display(decoded_df)
```

The notebook interface includes a sidebar with various Databricks services like Recents, Catalog, Workflows, Compute, and Data Engineering. A table named "stringBody" is visible in the results section, showing two rows of data:

	stringBody
1	{"id": "9", "temperature": "78", "humidity": "99", "timestamp": "2024-04-10T13:52:03.412161..."} {"id": "5", "temperature": "76", "humidity": "72", "timestamp": "2024-04-10T13:52:03.412161..."}

STEP 11: Changing the datatypes and storing the streaming data into columns.

The screenshot shows a Microsoft Azure Databricks notebook titled "eventhubs to DB 02" in Python. The code cell contains the following Python code:

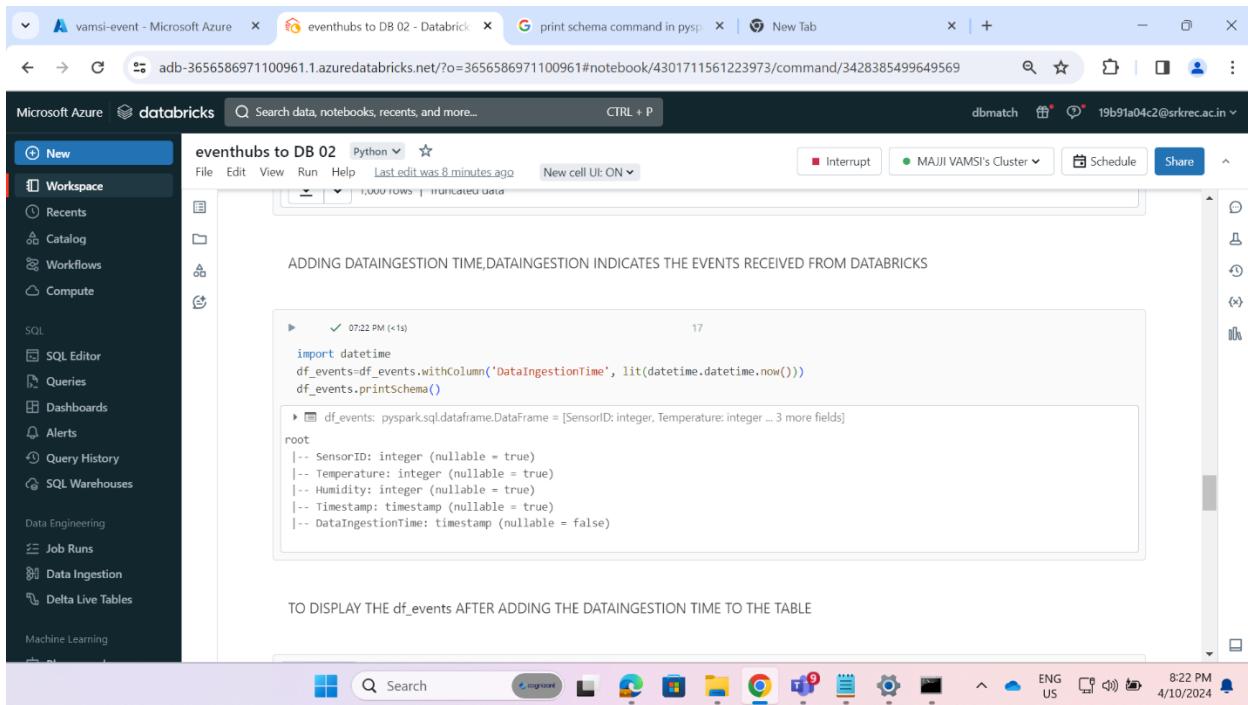
```
df_events = decoded_df.select(
    decoded_df.stringBody['id'].cast("int").alias("SensorID"),
    decoded_df.stringBody['temperature'].cast("int").alias("Temperature"),
    decoded_df.stringBody['humidity'].cast("int").alias("Humidity"),
    decoded_df.stringBody['timestamp'].cast('timestamp').alias("Timestamp")
)

display(df_events)
df_events.printSchema()
```

The notebook interface includes a sidebar with various Databricks services like Recents, Catalog, Workflows, Compute, and Data Engineering. A table named "df_events" is visible in the results section, showing four columns: SensorID, Temperature, Humidity, and Timestamp. The data is as follows:

	SensorID	Temperature	Humidity	Timestamp
1	2	93	81	2024-04-10T13:52:18.224+00:00
2	9	91	96	2024-04-10T13:52:18.224+00:00
3	3	78	88	2024-04-10T13:52:18.224+00:00
4	5	81	100	2024-04-10T13:52:18.224+00:00

STEP 12: Ingesting a new column for data ingestion timing.



The screenshot shows a Microsoft Azure Databricks notebook titled "eventhubs to DB 02". The sidebar on the left includes options like New, Workspace, Recents, Catalog, Workflows, Compute, SQL, and Data Engineering. The main area contains the following Python code:

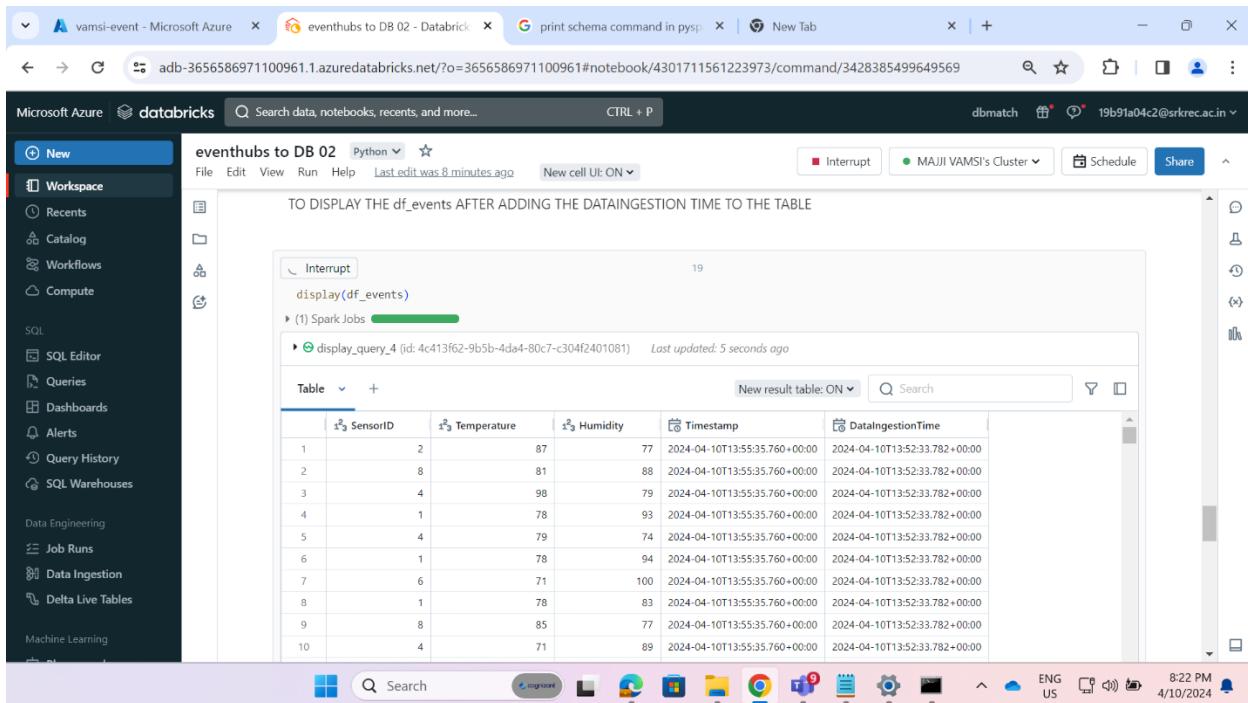
```
import datetime
df_events=df_events.withColumn("DataIngestionTime", lit(datetime.datetime.now()))
df_events.printSchema()
```

Below the code, a preview of the DataFrame is shown:

```
root
 |-- SensorID: integer (nullable = true)
 |-- Temperature: integer (nullable = true)
 |-- Humidity: integer (nullable = true)
 |-- Timestamp: timestamp (nullable = true)
 |-- DataIngestionTime: timestamp (nullable = false)
```

A note below the preview says "TO DISPLAY THE df_events AFTER ADDING THE DATAINGESTION TIME TO THE TABLE".

STEP 13: Display the newly added column to table.



The screenshot shows the same Databricks notebook. The sidebar and code from Step 12 remain the same. The main area now displays the output of the "display(df_events)" command:

```
display(df_events)
```

The resulting table shows 10 rows of data with the newly added "DataIngestionTime" column. The table has columns: SensorID, Temperature, Humidity, Timestamp, and DataIngestionTime. The "DataIngestionTime" column shows values such as 2024-04-10T13:55:35.760+00:00 and 2024-04-10T13:52:33.782+00:00.

	SensorID	Temperature	Humidity	Timestamp	DataIngestionTime
1	2	87	77	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
2	8	81	88	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
3	4	98	79	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
4	1	78	93	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
5	4	79	74	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
6	1	78	94	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
7	6	71	100	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
8	1	78	83	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
9	8	85	77	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00
10	4	71	89	2024-04-10T13:55:35.760+00:00	2024-04-10T13:52:33.782+00:00

STEP 14: Connecting with storage account Databricks and creating a required schema for external table.

```
spark.conf.set("fs.azure.account.key.finalpjdfs.core.windows.net", "6PCfieFMaEtuzXyGaql68if00TcvIB0Ue1ARD5zsgJjashVl6JrP15Mm/RaChulc+VredHoKEusc+AStc+lPQg==")
```

```
spark.sql(""" CREATE TABLE IF NOT EXISTS external_data (
    id INT,
    Temperature INT,
    Humidity INT,
    Timestamp TIMESTAMP,
    DataIngestionTime TIMESTAMP)
USING DELTA LOCATION 'abfss://refined@finalpj.dfs.core.windows.net/External_data' """)
```

STEP 15: Ingesting Streaming data into the external table.

```
spark.conf.set("spark.databricks.delta.schema.autoMerge.enabled", "true")
```

```
df_events.writeStream\
    .outputMode('append')\
    .option("checkpointLocation", '/tmp/delta/_checkpoints/')\
    .option("path", "abfss://refined@finalpj.dfs.core.windows.net")\
    .option("mergeSchema", "true")\
    .table('external_data')\
    .awaitTermination()
```

VISUALIZATION OF DATA:



Result:

The screenshot shows the Microsoft Azure Storage Explorer interface. The left sidebar shows a storage account named "finalpj" with sections for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser. The "Containers" section is currently selected. The main area displays a list of containers with their names, last modified dates, anonymous access levels, and lease states. There is a search bar at the top and a "Show deleted containers" toggle switch. The bottom navigation bar includes icons for Search, Refresh, Delete, Give feedback, and a user profile.

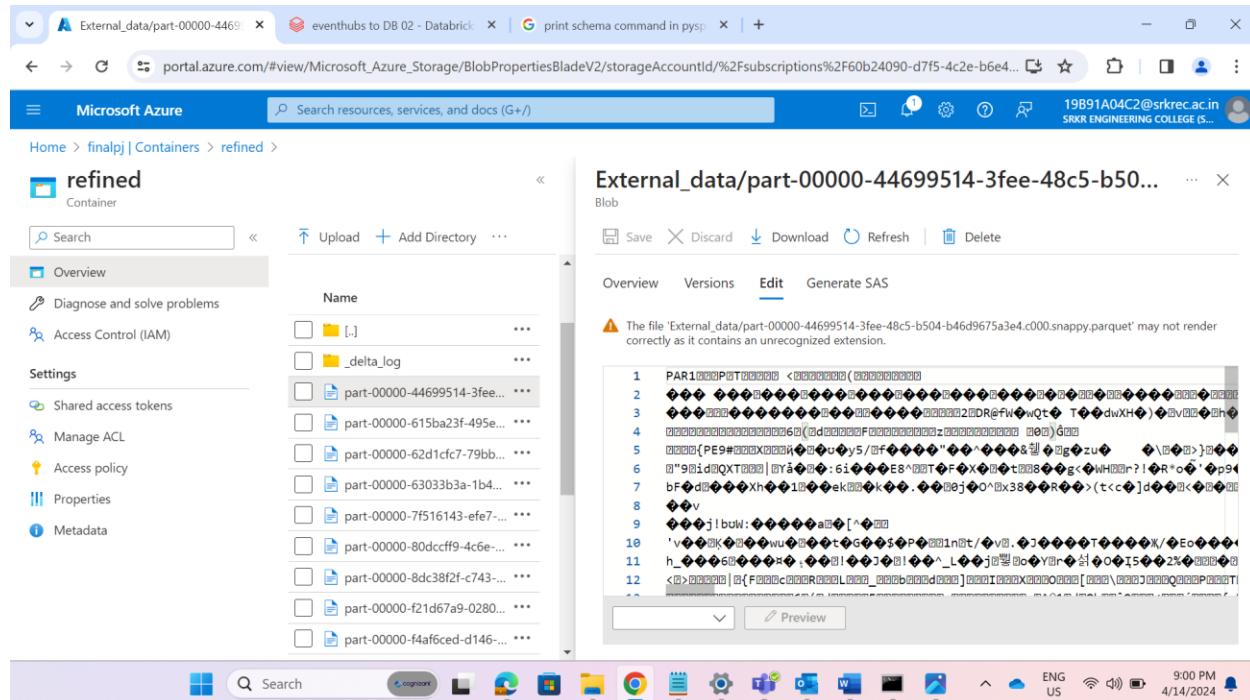
The screenshot shows the Microsoft Azure Storage Container Overview page for a container named "refined". The left sidebar contains navigation links for Overview, Diagnose and solve problems, Access Control (IAM), Settings (Shared access tokens, Manage ACL, Access policy, Properties, Metadata), and a search bar. The main area displays the authentication method as "Access key (Switch to Microsoft Entra user account)" and the location as "refined". A search bar for blobs by prefix is present. The table lists one blob named "External_data".

Name	Modified	Access tier	Archive status	Blob type	Size
External_data					

The screenshot shows the Microsoft Azure Storage Container Overview page for a container named "refined". The left sidebar contains navigation links for Overview, Diagnose and solve problems, Access Control (IAM), Settings (Shared access tokens, Manage ACL, Access policy, Properties, Metadata), and a search bar. The main area displays the authentication method as "Access key (Switch to Microsoft Entra user account)" and the location as "refined". A search bar for blobs by prefix is present. The table lists several blobs, including part files for a larger file named "part-00000-44699514-3fee-48c5-b504-b46d9675a3e...".

Name	Modified	Access tier	Archive status	Blob type	Size
[..]					
_delta_log					
part-00000-44699514-3fee-48c5-b504-b46d9675a3e...	4/10/2024, 7:39:17 PM	Hot (Inferred)		Block blob	4.7
part-00000-615ba23f-495e-41dc-af0d-db44216e83a...	4/10/2024, 7:39:13 PM	Hot (Inferred)		Block blob	7.0
part-00000-62d1fcf7-79bb-47fd-93d5-5c57035a7b7...	4/10/2024, 7:39:19 PM	Hot (Inferred)		Block blob	4.9
part-00000-63033b3a-1b42-40ff-bd64-4310221f09d...	4/10/2024, 7:39:10 PM	Hot (Inferred)		Block blob	6.8
part-00000-7f516143-efe7-4e05-bcb0-dd47adb6da4...	4/10/2024, 7:39:24 PM	Hot (Inferred)		Block blob	5.0
part-00000-80dccff9-4c6e-4686-8815-ed47dfeeba5...	4/10/2024, 7:39:15 PM	Hot (Inferred)		Block blob	5.6
part-00000-8dc38f2f-c743-48e9-9ecc-6ab27de0d22...	4/10/2024, 7:39:11 PM	Hot (Inferred)		Block blob	6.9
part-00000-f21d67a9-0280-4db1-9f4b-4ced4cd8c47...	4/10/2024, 7:39:22 PM	Hot (Inferred)		Block blob	4.9
part-00000-f4af6ced-d146-4b6f-b75b-1f2cc23fcddc...	4/10/2024, 7:39:20 PM	Hot (Inferred)		Block blob	5.0

A Microsoft Azure portal window showing the Blob storage container 'refined' under the 'finalpj' resource group. The container contains several files, including a Parquet file named 'part-00000-44699514-3fee-48c5-b504-b46d9675a3e4.c000.snappy.parquet'. A preview of the file content is displayed, showing binary data.



The screenshot shows the Azure portal interface with the following details:

- Address Bar:** portal.azure.com/#view/Microsoft_Azure_Storage/BlobPropertiesBladeV2/storageAccountId/%2Fsubscriptions%2F60b24090-d7f5-4c2e-b6e4...
19B91A04C2@srkrec.ac.in
SRKR ENGINEERING COLLEGE (S...)
- Container List:** refined
- File List:**
 - [..]
 - _delta_log
 - part-00000-44699514-3fee-48c5-b504-b46d9675a3e4.c000.snappy.parquet
 - part-00000-615ba23f-495e...
 - part-00000-62d1fcf7-79bb...
 - part-00000-6303b3a-1b4...
 - part-00000-7f516143-efe7...
 - part-00000-80dccff9-4c6e...
 - part-00000-8dc38f2f-c743...
 - part-00000-f21d67a9-0280...
 - part-00000-f4af5ced-d146...
- File Preview:** The preview pane shows the binary content of the Parquet file, with lines 1 through 12 visible:
 - PAR1
 - EOF
 - ... (Binary data)
 - ... (Binary data)
 - ... (Binary data)
 - ... (Binary data)
 - b7d5d1810d18e...
 - ...
 - ...
 - ...
 - h_...
 - ...
- Bottom Navigation:** Includes icons for File Explorer, Task View, Taskbar, Task Manager, Control Panel, Settings, Start, and a Cloud icon.