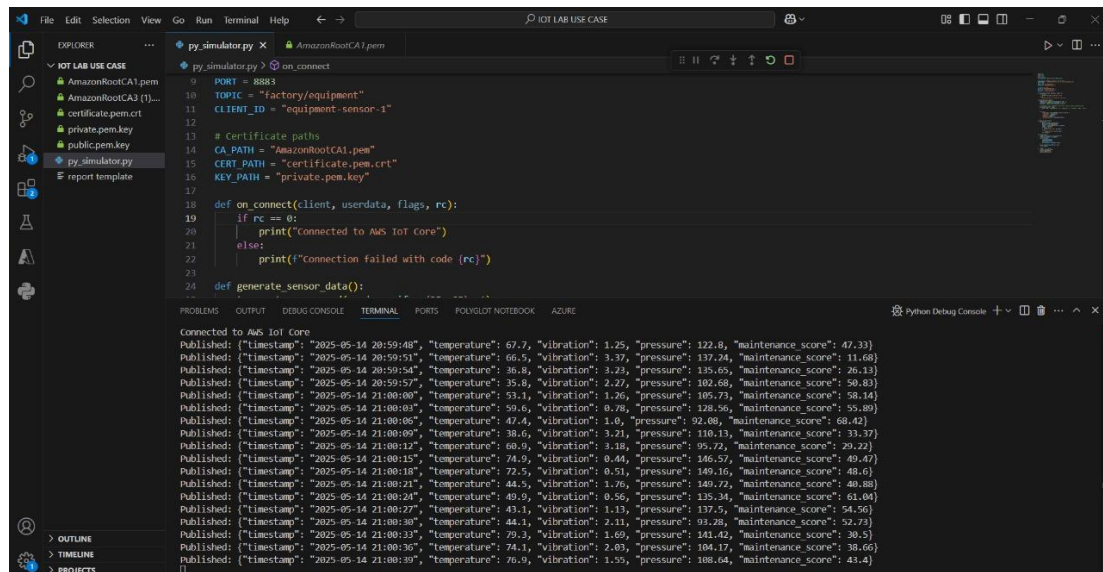


STEP 01 : DATA GENERATION AND INGESTION

1.1: Generating Sensor(Json) Data Using Python Code In Vs Code – Desktop



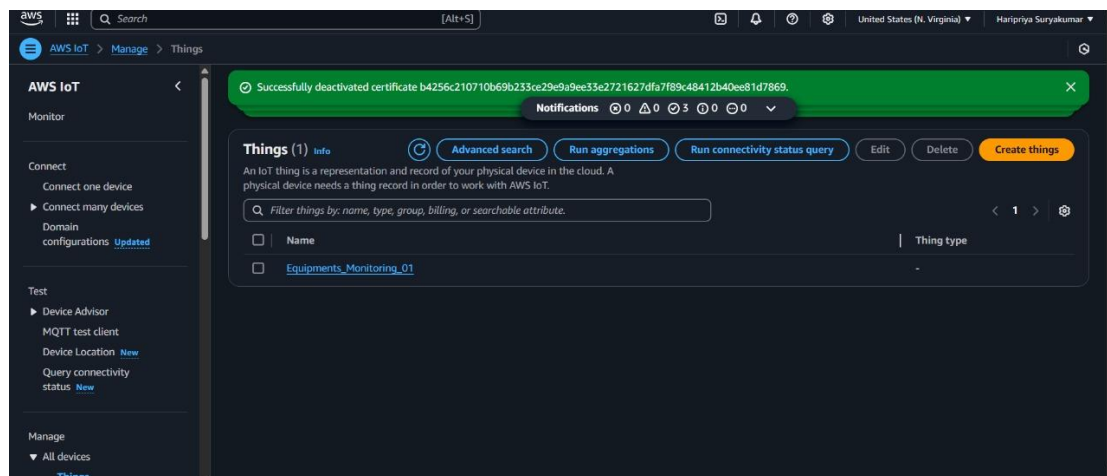
```
File Edit Selection View Go Run Terminal Help
py_simulator.py x AmazonRootCA1.pem

EXPLORER
IoT LAB USE CASE
AmazonRootCA1.pem
AmazonRootCA3 (1)...
certificate.pem.crt
private.pem.key
public.pem.key
py_simulator.py
report template

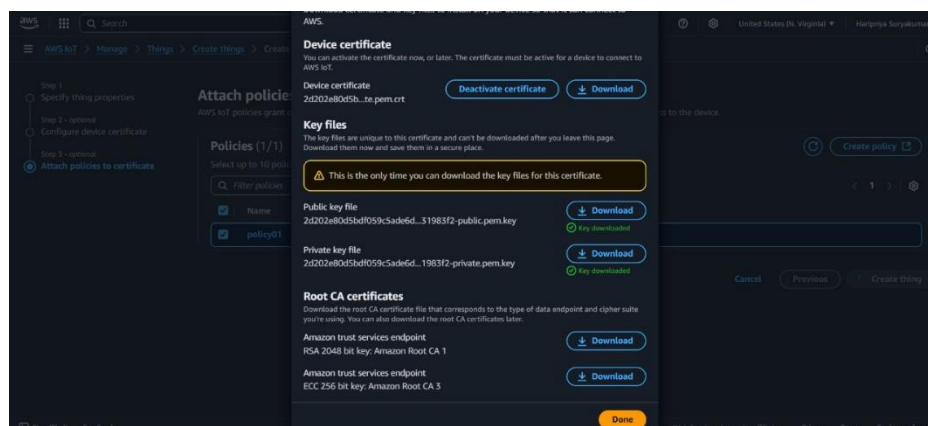
py_simulator.py > on_connect
9 PORT = 8883
10 TOPIC = "factory/equipment"
11 CLIENT_ID = "equipment-sensor-1"
12
13 # Certificate paths
14 CA_PATH = "AmazonRootCA1.pem"
15 CERT_PATH = "certificate.pem.crt"
16 KEY_PATH = "private.pem.key"
17
18 def on_connect(client, userdata, flags, rc):
19     if rc == 0:
20         print("connected to AWS IoT Core")
21     else:
22         print(f"Connection failed with code {rc}")
23
24 def generate_sensor_data():
25     ...

Connected to AWS IoT Core
Published: {"timestamp": "2025-05-14 20:59:48", "temperature": 67.7, "vibration": 1.25, "pressure": 122.8, "maintenance_score": 47.33}
Published: {"timestamp": "2025-05-14 20:59:51", "temperature": 66.5, "vibration": 3.37, "pressure": 137.24, "maintenance_score": 11.68}
Published: {"timestamp": "2025-05-14 20:59:54", "temperature": 36.8, "vibration": 3.23, "pressure": 135.65, "maintenance_score": 26.13}
Published: {"timestamp": "2025-05-14 20:59:57", "temperature": 35.8, "vibration": 2.27, "pressure": 102.68, "maintenance_score": 50.83}
Published: {"timestamp": "2025-05-14 21:00:00", "temperature": 53.1, "vibration": 1.26, "pressure": 105.73, "maintenance_score": 58.14}
Published: {"timestamp": "2025-05-14 21:00:03", "temperature": 59.6, "vibration": 0.78, "pressure": 128.56, "maintenance_score": 55.89}
Published: {"timestamp": "2025-05-14 21:00:06", "temperature": 47.4, "vibration": 1.0, "pressure": 92.08, "maintenance_score": 68.42}
Published: {"timestamp": "2025-05-14 21:00:09", "temperature": 38.6, "vibration": 3.21, "pressure": 110.13, "maintenance_score": 33.37}
Published: {"timestamp": "2025-05-14 21:00:12", "temperature": 60.9, "vibration": 3.18, "pressure": 95.72, "maintenance_score": 29.22}
Published: {"timestamp": "2025-05-14 21:00:15", "temperature": 74.9, "vibration": 0.46, "pressure": 146.57, "maintenance_score": 49.47}
Published: {"timestamp": "2025-05-14 21:00:18", "temperature": 72.5, "vibration": 0.51, "pressure": 149.16, "maintenance_score": 48.6}
Published: {"timestamp": "2025-05-14 21:00:21", "temperature": 44.5, "vibration": 1.76, "pressure": 149.72, "maintenance_score": 40.88}
Published: {"timestamp": "2025-05-14 21:00:24", "temperature": 49.9, "vibration": 0.56, "pressure": 135.34, "maintenance_score": 61.04}
Published: {"timestamp": "2025-05-14 21:00:27", "temperature": 43.1, "vibration": 1.13, "pressure": 137.9, "maintenance_score": 54.56}
Published: {"timestamp": "2025-05-14 21:00:30", "temperature": 48.1, "vibration": 2.11, "pressure": 93.28, "maintenance_score": 32.78}
Published: {"timestamp": "2025-05-14 21:00:33", "temperature": 79.3, "vibration": 1.69, "pressure": 141.42, "maintenance_score": 30.5}
Published: {"timestamp": "2025-05-14 21:00:36", "temperature": 74.1, "vibration": 2.03, "pressure": 104.17, "maintenance_score": 38.66}
Published: {"timestamp": "2025-05-14 21:00:39", "temperature": 76.9, "vibration": 1.55, "pressure": 108.64, "maintenance_score": 43.4}
```

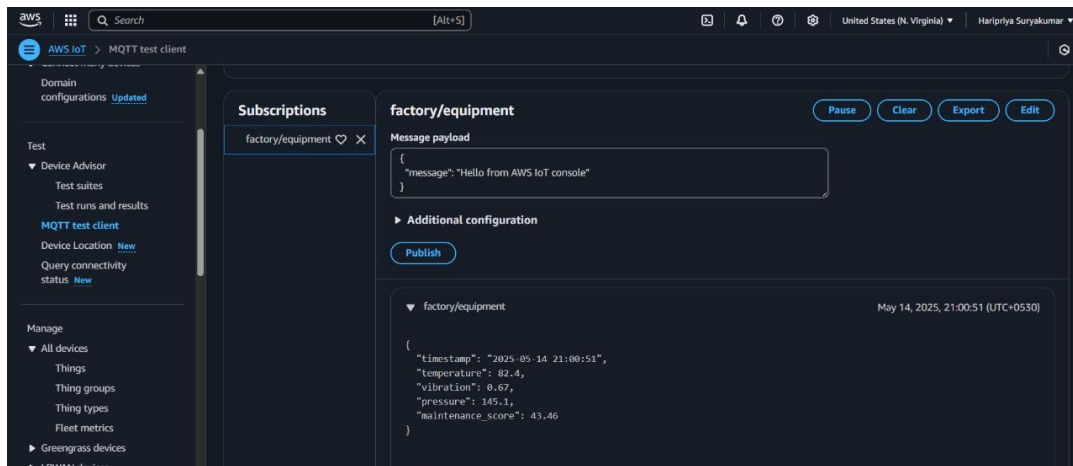
1.2 AWS IOT CORE CREATION



1.3 AWS IOT CORE – CERTIFICATES

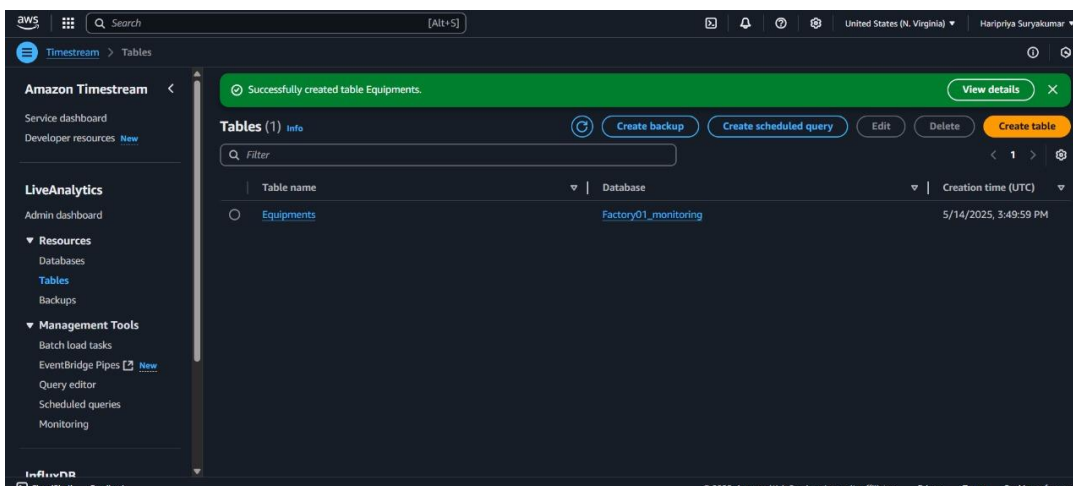
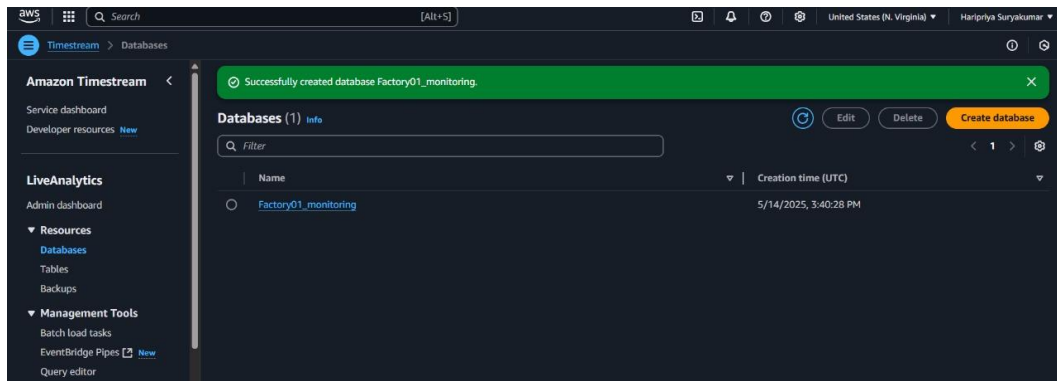


1.4 MQTT TEST CLIENT RECEIVING DATA

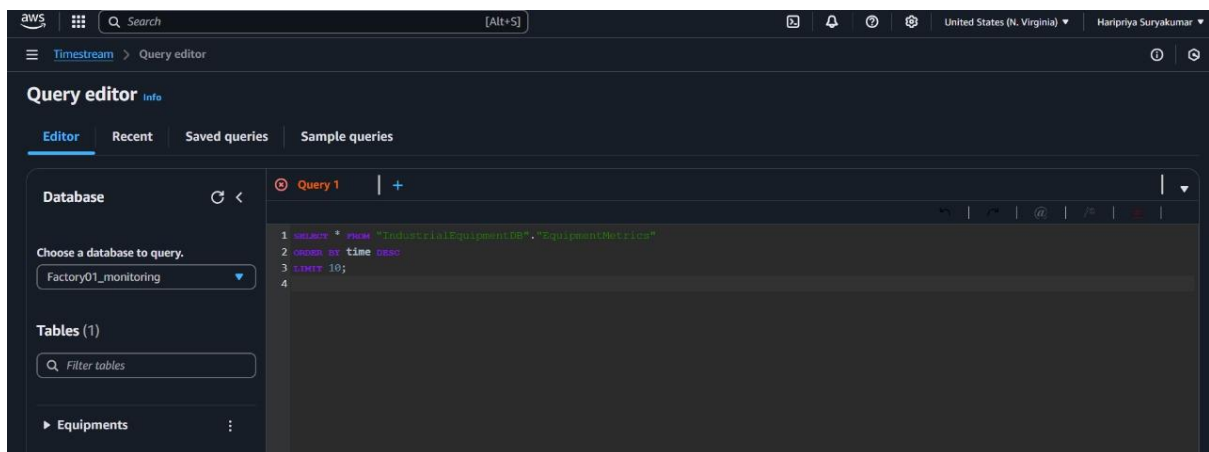


STEP 02: DATA PROCESSING & LOADING

2.1 Storing time-stamped JSON data in AWS Timestream – creating database followed by table inside database

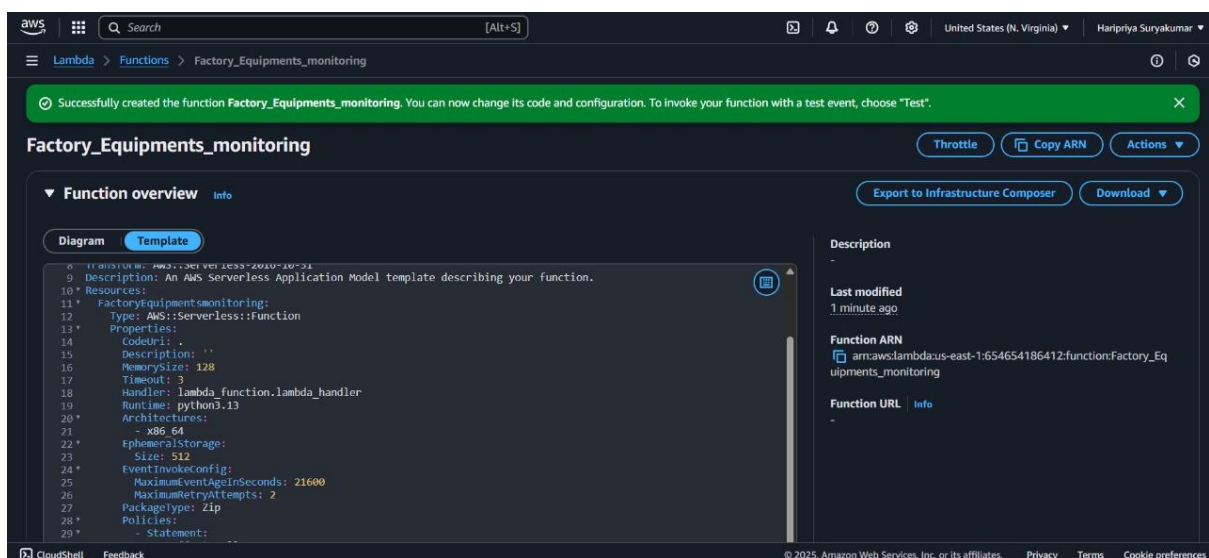


2.2 Filtering data from timestream to AWS Lambda using query in query editor

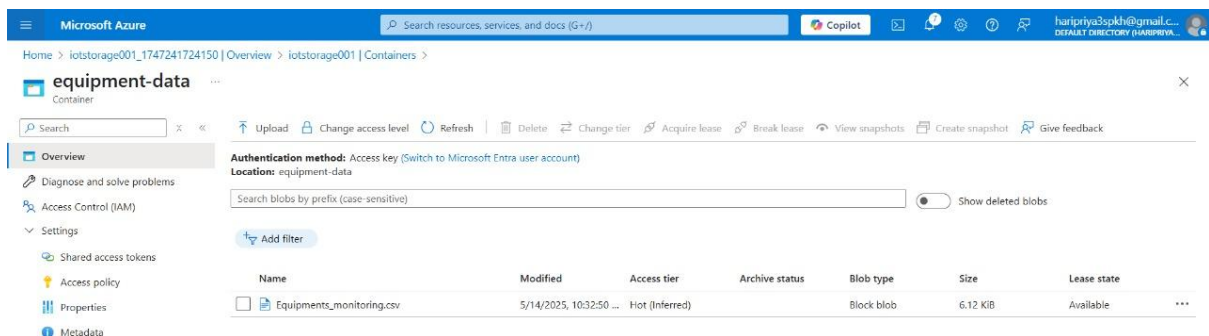


STEP 03: PUSHING DATA FROM AWS TO AZURE BLOB USING AWS LAMBDA

3.1 Connecting the source(AWS Timestream database) and destination(AZURE blob storage - container) to AWS Lambda using python code and their respective URLs and access keys.

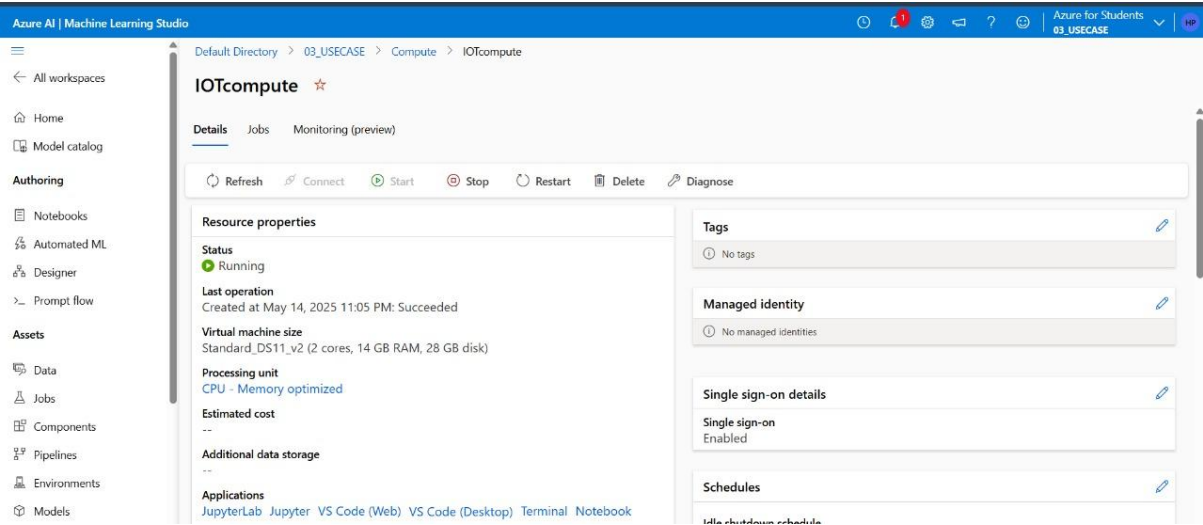


3.2 Data getting stored in azure blob as .csv file

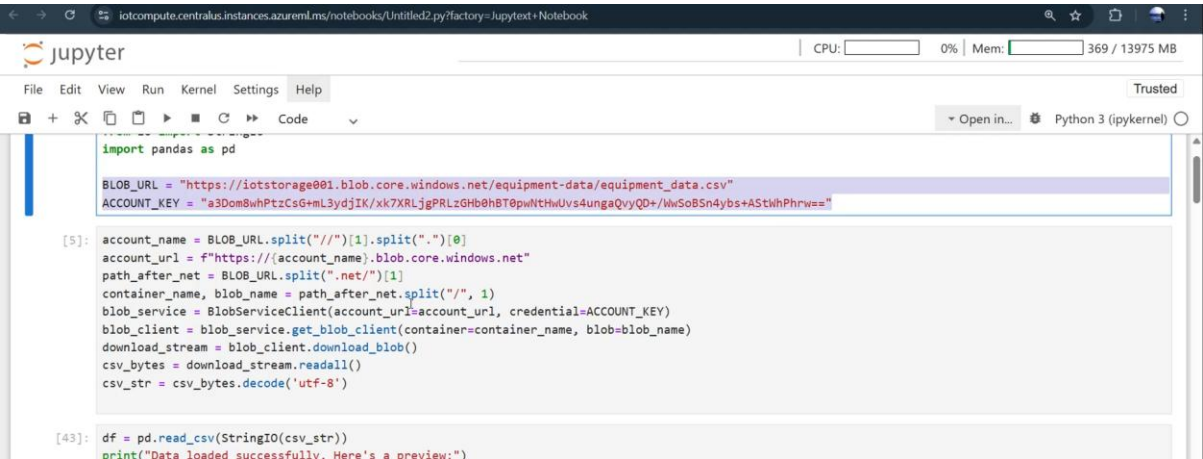


STEP 04: PREDICTIVE MAINTAINANCE IN AZURE ML

4.1 Creating ML workspace Azure ML



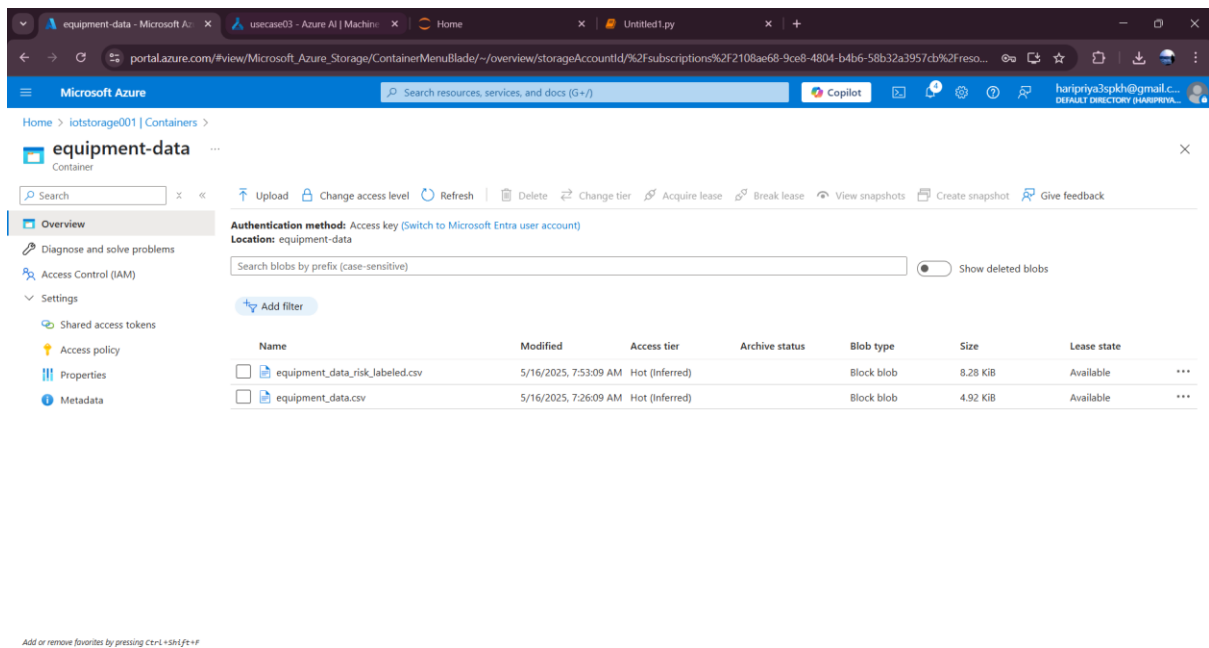
4.2 Model loading, training & testing in Jupyter in Azure ML – Virtual machine



4.3 Classification report

Accuracy: 0.9				
Classification Report:				
	precision	recall	f1-score	support
0	0.79	1.00	0.88	11
1	1.00	0.84	0.91	19
accuracy			0.90	30
macro avg	0.89	0.92	0.90	30
weighted avg	0.92	0.90	0.90	30

4.4 Predictions stored in same container as .csv file



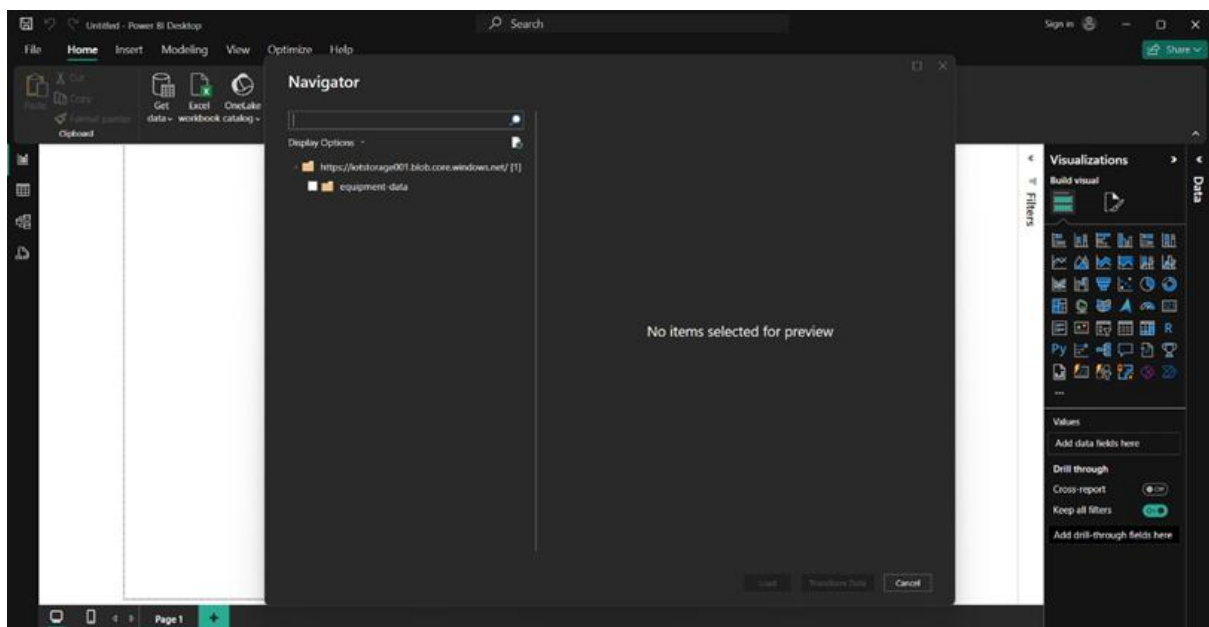
Microsoft Azure portal interface showing the 'equipment-data' container. The container contains two CSV files: 'equipment_data_risk_labeled.csv' and 'equipment_data.csv'.

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
equipment_data_risk_labeled.csv	5/16/2025, 7:53:09 AM	Hot (Inferred)		Block blob	8.28 KiB	Available
equipment_data.csv	5/16/2025, 7:26:09 AM	Hot (Inferred)		Block blob	4.92 KiB	Available

STEP 05: VISUALIZATIONS

5.1 POWER BI

5.1.1 Connecting Azure blob storage to Power BI Desktop



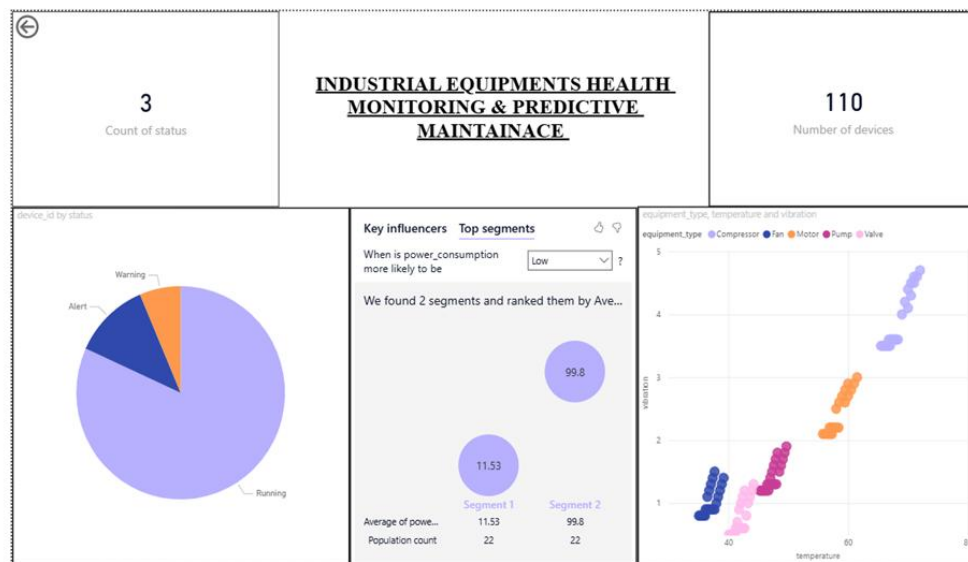
Microsoft Power BI Desktop interface showing the Navigator pane. The Navigator pane displays the connection to the 'equipment-data' container. The Visualizations pane is empty, indicating no items are selected for preview.

5.1.2 Raw data dashboard



5.1.3 Predictive Analysis data dashboard

5.2



GRAFANA DASHBOARD – LIVE DATA STREAM

5.2.1 Table view

The dashboard displays a table view of live data stream. The table has the following columns: **device_id**, **timestamp**, **equipment_type**, **location**, **temperature**, **vibration**, **pressure**, and **power_c**.

device_id	timestamp	equipment_type	location	temperature	vibration	pressure	power_c
DEV001		Pump	Area A	45.8	1.20	5.50	
DEV002		Motor	Area B	56.4	2.10	0.100	
DEV003		Fan	Area C	35.5	0.800	0	
DEV004		Compressor	Area A	66.1	3.50	8.20	
DEV005		Valve	Area B	40.5	0.500	6.10	
DEV006		Pump	Area C	47.1	1.30	5.70	

5.2.2 Dashboard

