



University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING

Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

# TECHNICAL REPORT WEATHER ANALYTICS DASHBOARD



**SEMESTER 2**

# CONTENTS

Weather Analytics Dashboard .....2

Executive Summary .....2

Technical Report.....3

Abstract .....4

Methodology .....4

Results Section .....5

Discussion.....13

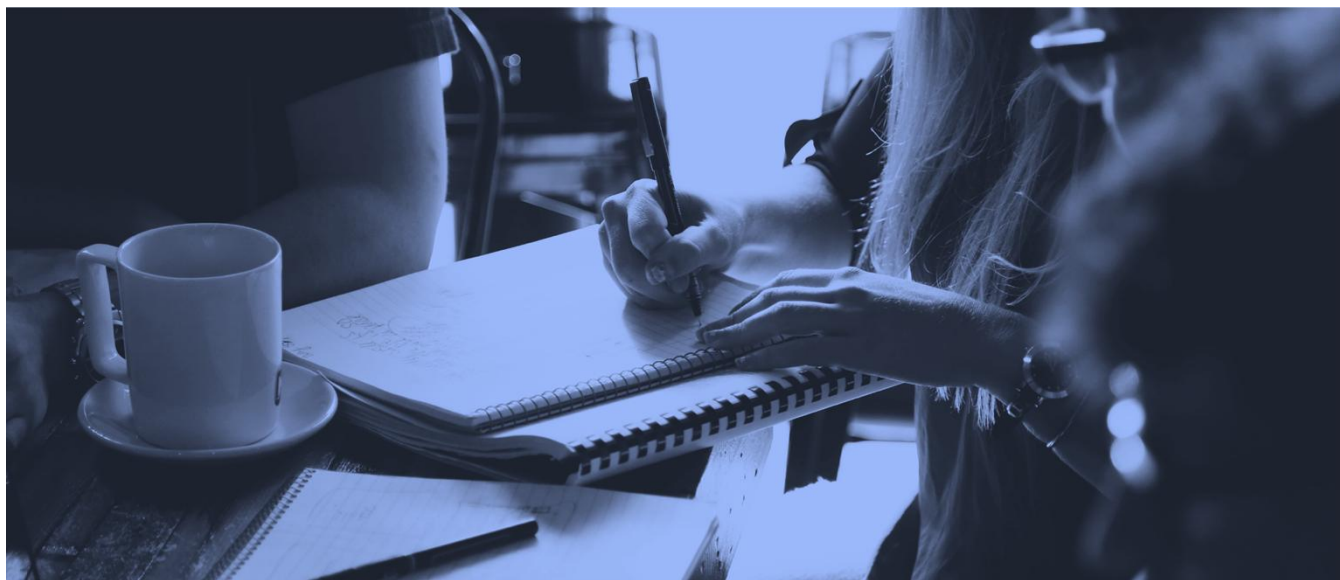
Conclusion .....13

Contributions/References.....14

# Weather Analytics Dashboard

## Executive Summary

This project presents an end-to-end **Weather Analytics Dashboard Pipeline** using the data from the **Open-Meteo API** and processing it through AWS services. Our primary aim was to create a scalable and automated data engineering solution that renders insights for multiple cities in the U.S in both daily and hourly granularity.



### Team Members:

Aldric Pinto

Tanak Patel

Trevor Hitchcock

### Questions?

Contact:

[Aldric Pinto](#)

[Tanak Patel](#)

[Trevor Hitchcock](#)

# Technical Report

## *Weather Analytics Dashboard*

### Highlights of Project

**Problem:** Weather Data is critical for planning in the fields of logistics, public safety and so on. But accessing data in a structured and analysis-ready format is a challenge.

**Objective:** Create a robust pipeline that automates the ETL process (using AWS services) and visualize it in a user-friendly dashboard.

**Impact:** This project shows how data from an external source like an API can be transformed into a powerful analytics tool with the help of cloud infrastructure.



### Submitted on:

April 27<sup>th</sup>, 2025

## Abstract

This project is a weather analytics dashboard powered by a scalable data pipeline built by utilizing Amazon Web Services. The pipeline retrieves weather data daily from the Open-Meteo API, processes it using AWS Glue ETL, and stores the raw data in Amazon S3. AWS Glue Crawler is used for schema detection and AWS Glue Catalog is used to store and organize this data, while Athena allows us to query our data. The resulting data is visualized in a Power BI, showing the user key weather metrics such as temperature, precipitation, UV index, and diurnal patterns across major cities in the United States. The pipeline is updated daily with weather trends from the previous day, giving the user real time insights to enhance routing and risk mitigation strategies in logistics operations. Our project shows how cloud computing can deliver meaningful, timely, and actionable weather data.

## Methodology

Our project followed the CRISP-DM methodology to develop a weather analytics dashboard specifically tailored for logistics companies. The steps included business understanding, data understanding, data preparation, modeling, and evaluation, all integrated within a serverless cloud-based architecture on AWS.

### Business Understanding

The primary objective of our project was to provide logistics companies with weather insights that assist decision-making regarding routing, delivery scheduling, and risk mitigation. The solution aimed to streamline weather data collection, processing, and visualization using a scalable and cost-effective pipeline.

## Results Section

### Data Understanding

The data was sourced from the Open-Meteo API, a free source of historical weather data. We selected variables such as temperature (min, max, and hourly averages), precipitation, UV index, cloud cover, and day/night status, because of their relevancy to logistics planning.

### Data Preparation

A daily ETL process was implemented using AWS Glue, which extracted weather data for several major U.S. cities through API calls. The raw JSON responses were cleaned and transformed into structured tabular formats, and the transformed data was stored in Amazon S3 and crawled using AWS Glue Crawlers, which inferred schema definitions and registered metadata into the AWS Glue Data Catalog for easy access.

### Modeling

Although no predictive modeling was conducted, a data modeling layer was introduced using Amazon Athena to write SQL queries on top of the prepared datasets. These queries supported metric aggregation, temporal filtering, and city-level comparisons, serving as the foundation for the dashboard's visual analytics.

### Evaluation

The final dashboard was developed in Power BI, linked via an Athena ODBC connection to ensure dynamic, query-driven data refresh. The visual interface allows logistics companies to track weather conditions across time and location, evaluate potential disruptions, and make more resilient scheduling decisions. The daily automation and modular design allow scalability to more cities and additional weather variables as needed.

## Data Engineering Pipeline Overview

The project implemented a fully automated data pipeline based on a serverless architecture using AWS services. The schema was designed to capture both daily and hourly weather metrics across multiple cities, organized as follows:

- Date (primary key for daily data)
- Timestamp (for hourly resolution)
- City Name
- Latitude / Longitude
- Temperature (Min/Max/Average)
- Precipitation
- Cloud Cover
- UV Index
- Is Daytime (Boolean)

## Data Ingestion

Weather data was ingested using scheduled API calls to the Open-Meteo API, which returned JSON-formatted weather records for multiple U.S. cities. The ingestion script was embedded within an AWS Glue ETL job, allowing us to use recurring automated fetches.

## Data Storage

Raw data was stored in Amazon S3, with folder paths structured by city and date for logical organization and easy querying. The storage layer was optimized for scalability and low-latency access via Athena.

## Data Processing

The ETL script in AWS Glue cleaned and transformed the JSON responses into structured tabular data. AWS Glue Crawlers automatically inferred schema and populated the AWS Glue Data Catalog, making the datasets accessible through Amazon Athena.

## Data Consumption

The processed data was queried using Athena and connected to Power BI via ODBC. This allowed for flexible consumption of the data and slicing/filtering based on user specific needs.

## Model Deployment

While the project did not involve machine learning models, the pipeline itself was deployed to run daily on schedule, forming a production-ready data infrastructure. This environment supports continuous data updates without manual intervention and could be easily extended to support forecasting models in future iterations.

## Data Visualization

Key insights were delivered through a series of interactive Power BI dashboards:

- Line graphs showing hourly temperature fluctuations by city as well as average cloud cover.
- Stacked area charts representing average temperature every hour
- Bar charts comparing precipitation levels across cities
- Slicers and filters enabling selection by city and day or night

Each visualization was built with interactivity in mind to support data-driven decision-making for logistics professionals.

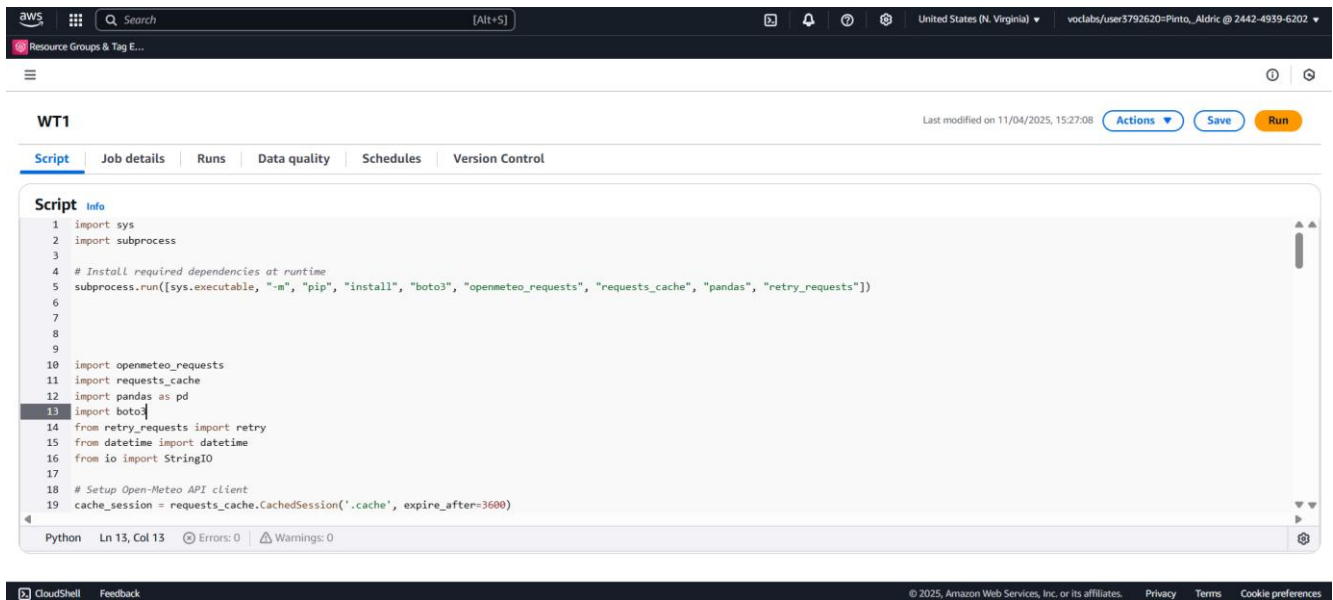
## Deployment

The final dashboard was published in Power BI Service, with visualization updates which are aligned with the pipeline's daily data update. Users can access the dashboard through a secure URL and interact with real-time weather intelligence relevant to their operations.



## Screenshots

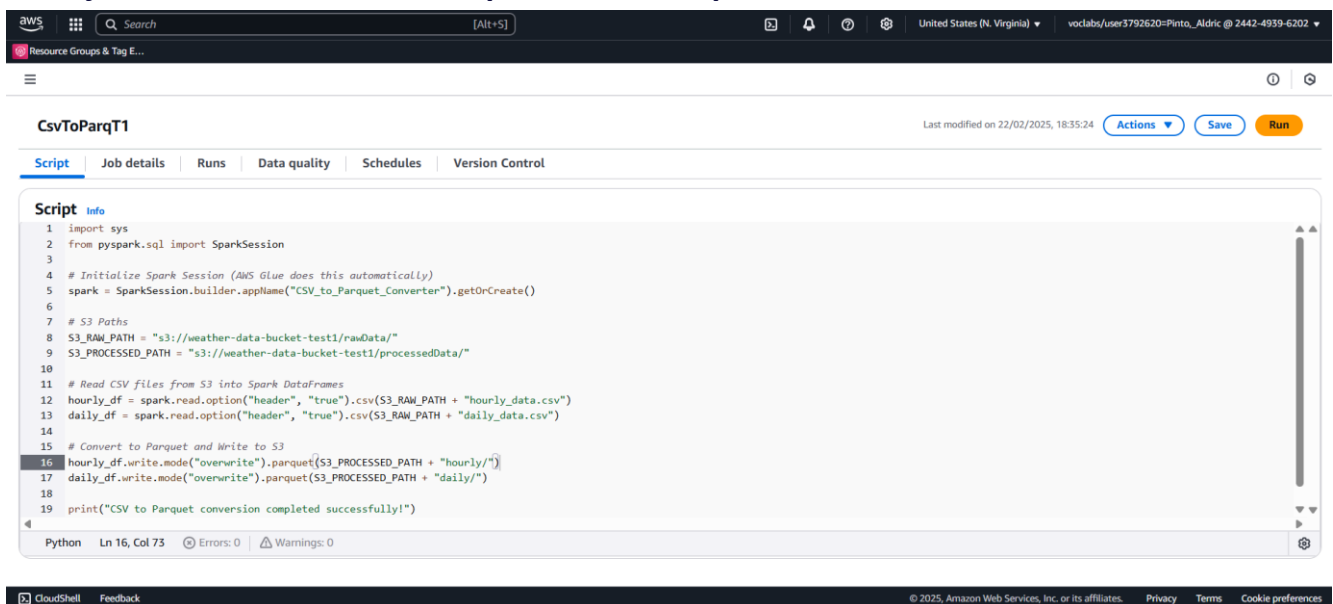
### Glue job to fetch data (CSV format) from Open-Meteo API to raw-data folder in S3:



```
1 import sys
2 import subprocess
3
4 # Install required dependencies at runtime
5 subprocess.run([sys.executable, "-m", "pip", "install", "boto3", "openmeteo_requests", "requests_cache", "pandas", "retry_requests"])
6
7
8
9
10 import openmeteo_requests
11 import requests_cache
12 import pandas as pd
13 import boto3
14 from retry_requests import retry
15 from datetime import datetime
16 from io import StringIO
17
18 # Setup Open-Meteo API client
19 cache_session = requests_cache.CachedSession('.cache', expire_after=3600)
```

Python Ln 13, Col 13 | Errors: 0 | Warnings: 0

### Glue job to convert CSV to Parquet format to processed-data folder in S3:



```
1 import sys
2 from pyspark.sql import SparkSession
3
4 # Initialize Spark Session (AWS Glue does this automatically)
5 spark = SparkSession.builder.appName("CSV_to_Parquet_Converter").getOrCreate()
6
7 # S3 Paths
8 S3_RAW_PATH = "s3://weather-data-bucket-test1/rawData/"
9 S3_PROCESSED_PATH = "s3://weather-data-bucket-test1/processedData/"
10
11 # Read CSV files from S3 into Spark DataFrames
12 hourly_df = spark.read.option("header", "true").csv(S3_RAW_PATH + "hourly_data.csv")
13 daily_df = spark.read.option("header", "true").csv(S3_RAW_PATH + "daily_data.csv")
14
15 # Convert to Parquet and Write to S3
16 hourly_df.write.mode("overwrite").parquet(S3_PROCESSED_PATH + "hourly/")
17 daily_df.write.mode("overwrite").parquet(S3_PROCESSED_PATH + "daily/")
18
19 print("CSV to Parquet conversion completed successfully!")
```

Python Ln 16, Col 73 | Errors: 0 | Warnings: 0

## S3 bucket:

The screenshot shows the AWS Management Console interface for an Amazon S3 bucket named 'weather-data-bucket-test1'. The left sidebar displays the 'Amazon S3' navigation menu with options like 'General purpose buckets', 'Directory buckets', 'Table buckets', 'Access Grants', 'Access Points', 'Object Lambda Access Points', 'Multi-Region Access Points', 'Batch Operations', 'IAM Access Analyzer for S3', 'Block Public Access settings for this account', 'Storage Lens', 'Dashboards', 'Storage Lens groups', 'AWS Organizations settings', and 'Feature spotlight'. The main content area shows the bucket's 'Objects' tab, listing five objects: 'athena/' (Folder), 'processedData\_\$folder\$' (Folder), 'processedData/' (Folder), 'rawData/' (Folder), and 'Unsaved/' (Folder). The 'processedData\_\$folder\$' object is highlighted, showing its last modified date as 'March 14, 2025, 12:05:42 (UTC-04:00)' and its storage class as 'Standard'. The top of the console shows the AWS logo, a search bar, and the user's account information: 'United States (N. Virginia)' and 'voclabs/user3792620=Pinto\_Aldric @ 2442-4939-6202'.

## Amazon Athena:

The screenshot shows the Amazon Athena console interface. The top navigation bar includes 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. The 'Editor' tab is active, showing a SQL query: 'SELECT \* FROM "AwsDataCatalog"."tbl1"."p1daily" limit 10;'. Below the query editor, there are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. The 'Query results' tab is selected, showing a green status bar indicating 'Completed'. The query results are displayed in a table with 10 rows and 7 columns: '#', 'temperature\_2m\_max', 'temperature\_2m\_min', 'uv\_index\_max', 'location', 'date\_only', and 'time\_only'. The first four rows of data are visible. The bottom of the console shows the footer with 'CloudShell', 'Feedback', and copyright information: '© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

#	temperature_2m_max	temperature_2m_min	uv_index_max	location	date_only	time_only
1	18.535	8.935	6.95	New Haven	2025-04-20	04:00:00
2	12.935	5.835	6.85	New Haven	2025-04-21	04:00:00
3	20.075998	9.576	7.15	New Haven	2025-04-22	04:00:00
4	16.776	7.576	7.0	New Haven	2025-04-23	04:00:00

The screenshot displays the Amazon Athena Query Editor interface. At the top, the AWS logo and search bar are visible. The main header shows 'Amazon Athena' and 'Query editor'. Below this, there are tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. The 'Editor' tab is active, showing a SQL query: `SELECT * FROM "AwsDataCatalog"."wt1"."p1shourly" limit 10;`. Below the query editor, there are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. To the right of these buttons, there is a 'Reuse query results' option with a link to 'up to 60 minutes ago'. Below the query editor, there are tabs for 'Query results' and 'Query stats'. The 'Query results' tab is active, showing a green status bar indicating 'Completed'. Below this, there is a search bar for 'Search rows'. The results are displayed in a table with 10 columns: #, temperature\_2m, rain, snow\_depth, cloud\_cover, location, date\_only, time\_only, and is\_day. The table contains 4 rows of data.

#	temperature_2m	rain	snow_depth	cloud_cover	location	date_only	time_only	is_day
1	18.085	0.0	0.0	100.0	New Haven	2025-04-20	04:00:00	1
2	17.935	0.0	0.0	100.0	New Haven	2025-04-20	05:00:00	1
3	18.535	0.0	0.0	57.0	New Haven	2025-04-20	06:00:00	1
4	17.185	0.0	0.0	1.0	New Haven	2025-04-20	07:00:00	1

## ODBC Connector for connecting Athena to Power BI

Simba Athena ODBC Driver DSN Setup

Data Source Name: weatherdata

Description: weather data

AWS Region: us-east-1

Catalog: AwsDataCatalog

Schema: default

Workgroup: primary

Metadata Retrieval Method: Auto

Output Options

S3 Output Location: s3://weather-data-bucket-test1/athena/

Encryption Options: NOT\_SET

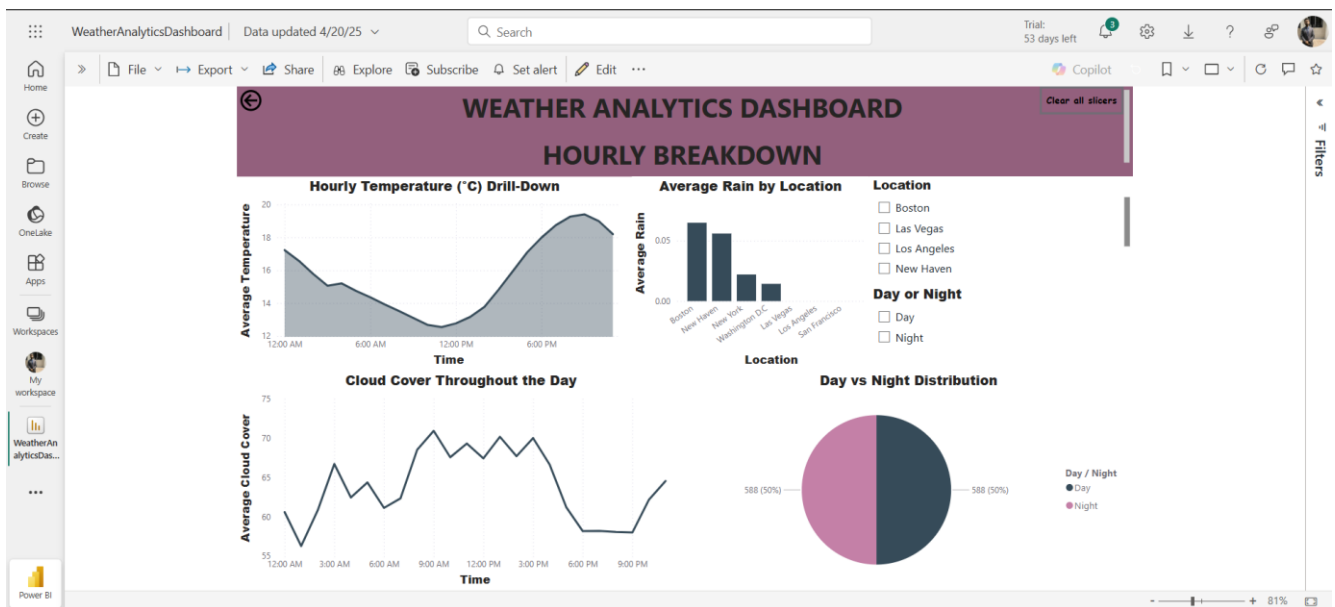
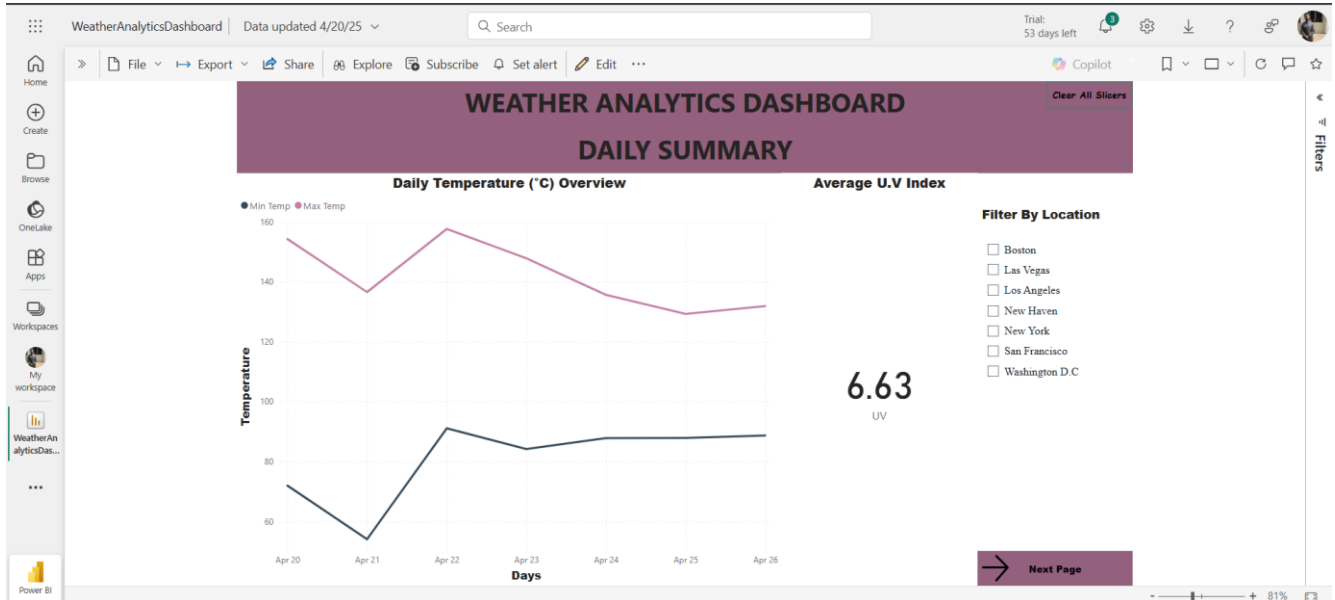
KMS Key:

Authentication Options... Advanced Options... Logging Options...

Endpoint Override Options Proxy Options...

v1.2.3.1000 (64 bit) Test... OK Cancel

## PowerBI dashboard



## Discussion

The result from our weather analytics dashboard helps us answer an important question:

Can we build a system that's fast, affordable and able to handle large amounts of weather data to give us useful insights?

Our answer is **Yes** – and we're getting here.

We used Aws services including AWS Glue, AWS S3, AWS Athena and then connected Power Bi to Athena.

Our main discussion point was that what makes this setup powerful isn't just the tools we used, but how well they work together. This system is cost effective, scalable and mostly automated, which solves many problems that usually come with traditional weather analysis system.

The major issue is data quality. If the data coming in is missing or late, it can mess up the results and the charts we show. This means we need better checks on the data as it comes in and possibly faster ways to collect it.

Even with these limits, this project shows real progress. We've built a solid base using cloud tools to turn raw weather data into something useful. In the future, we could improve it by adding machine learning (using tools like Amazon SageMaker) to predict weather or using real-time streaming (with AWS Kinesis) to make updates faster.

## Conclusion

This project shows that we can build a weather analytics system that is easy to use, low-cost, and can grow with more data, by using cloud tools like Amazon S3, AWS Glue, Athena, and Power BI. In the past, this kind of system needed a lot of setup and manual work, but now it can be done automatically using serverless technology. This makes it much easier and faster to analyze weather data.

Our dashboard takes complex weather information and turns it into simple charts and visuals, so people can easily see patterns and make smart decisions. It shows how today's cloud tools are fast, flexible, affordable, and mostly automatic. This project gives a clear example of how future weather systems can be built using these modern tools.

## Contributions/References

<https://open-meteo.com/>

<https://docs.aws.amazon.com/athena/>

<https://docs.aws.amazon.com/glue/>

<https://excalidraw.com/>