

Data Analysis

Analysis of Global Land Average Temperature Trends

By Vaishali Bokadiya
Vaishalibokadiya19@gmail.com

Project Overview :

Analysis of Global Land Average Temperature Trends

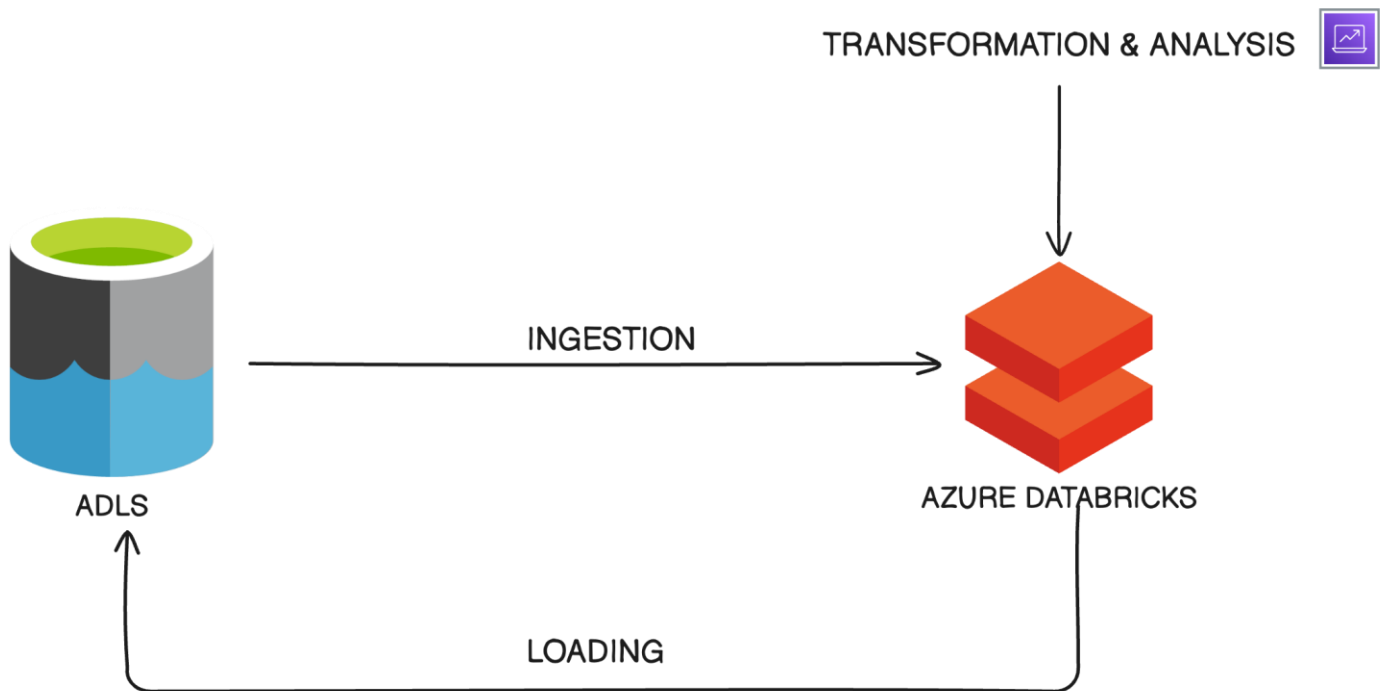
Our project is to implement data analysis using Spark SQL on Azure Databricks and process data for errors, seasonality, and anomalies. Some say climate change is the biggest threat of our age while others say it's a myth based on dodgy science. The goal of this project is to analyze the Global Land Average Temperature dataset to understand long-term temperature trends, detect anomalies, and identify potential factors contributing to climate change.

Project Requirements:

1. **Data Source:** We need access to data that needs to be analysed. This could be stored in various formats such as CSV, Parquet, JSON, or in a database.
2. **Azure Databricks Environment:** Set up an Azure Databricks workspace and create a cluster with appropriate configurations based on the size of your data and computational requirements.
3. **PySparkSQL :** PySparkSQL, which is the Python API for Spark SQL. This includes understanding how to create SparkSession, loading data, performing data transformations, and executing SQL queries.
4. **Data Preparation:** Data often requires preprocessing before analysis. This may involve handling missing values, converting data types, filtering out irrelevant data, and ensuring the data is properly formatted for analysis.
5. **Data Analysis:**
 - Utilize PySparkSQL queries to analyze the temperature data for long-term trends, seasonal variations, and anomalies.
 - Aggregate the data to calculate statistical metrics such as average temperature by year or decade.

6. **Visualization:** Visualize the results of the analysis using plots, charts, and maps to convey insights effectively.

Architecture :



Azure Resources Used for this Project:

AZURE TOOLS :

- **Azure Data Lake Storage** : Azure Data Lake Storage (ADLS) is a scalable and secure cloud-based storage solution provided by Microsoft Azure. It's designed for big data analytics workloads and is optimized for storing large amounts of structured, semi-structured, and unstructured data.
- **Azure Databricks** : Azure Databricks is a fast, easy, and collaborative Apache Spark-based analytics platform optimized for Azure. It provides a fully managed, cloud-based environment that integrates seamlessly with other Azure services, allowing data engineers, data scientists, and analysts to collaborate on big data and machine learning projects.

AZURE TECHNOLOGIES :

- **Pyspark** : PySpark is the Python API for Apache Spark, a distributed computing framework for processing large datasets. It allows developers to write Spark applications using Python programming language, leveraging Spark's distributed computing capabilities.
- **Spark SQL**: Spark SQL is a module in Apache Spark for processing structured data using SQL and DataFrame API. It provides a unified interface for querying structured data sources, enabling seamless integration of SQL queries with Spark's distributed processing engine.

How it Works :

1. Data Ingestion:
Load the dataset into Azure Databricks storage (e.g., Azure Blob Storage).
2. Data Exploration and Cleaning:
 - Explore the structure and contents of the dataset to understand its schema and characteristics.
 - Perform data cleaning steps to handle missing values, outliers, and inconsistencies.
3. Data Analysis:
 - Utilize PySparkSQL queries to analyze the data for errors, seasonality, and anomalies.
 - Aggregate the data to calculate statistics such as average.
4. Error Detection:
Utilize Spark SQL queries or built-in functions to detect errors in your data. This could involve identifying inconsistencies, duplicates, or unexpected values.
5. Seasonality Analysis:
Use Spark SQL functions or libraries like PySpark's pandas or numpy to analyze seasonality patterns in your data. This could involve time series analysis, trend detection, or Fourier transforms.
6. Anomaly Detection:
Implement anomaly detection algorithms using Spark SQL or PySpark libraries.
7. Visualization:
Visualize the results of the analysis using plots, charts, or dashboards to communicate insights effectively.

About Dataset :

Climate Change: Earth Surface Temperature Data

Exploring global temperatures since 1750

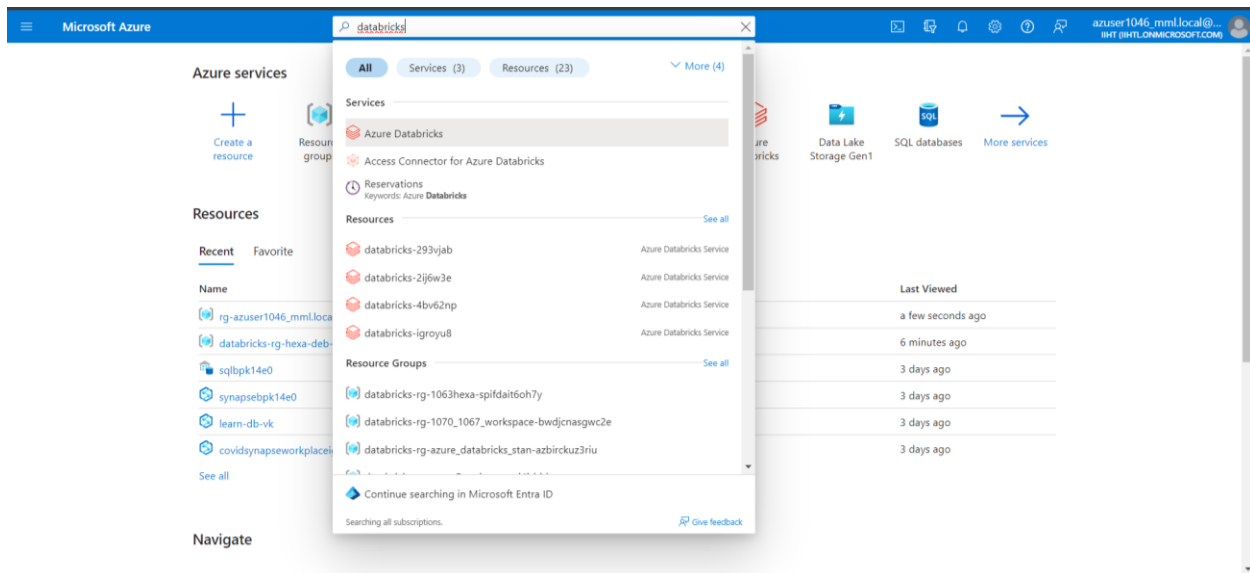
The Berkeley Earth Surface Temperature Study combines 1.6 billion temperature reports from 16 pre-existing archives. It is nicely packaged and allows for slicing into interesting subsets (for example by country). They publish the source data and the code for the transformations they applied. They also use methods that allow weather observations from shorter time series to be included, meaning fewer observations need to be thrown away.

Early data was collected by technicians using mercury thermometers, where any variation in the visit time impacted measurements. In the 1940s, the construction of airports caused many weather stations to be moved. In the 1980s, there was a move to electronic thermometers that are said to have a cooling bias. Our Dataset contains columns Date, Average Temperature, Average Temperature Uncertainty, Country.

```
GlobalLandTemperaturesByCountry.csv X
D: > Hexaware > Data_Engineering-Batch > Data_Engineering_Projects > Datasets > GlobalLandTemperatures > GlobalLandTemperaturesByCountry.csv
1 Date,AverageTemperature,AverageTemperatureUncertainty,Country
2 1743-11-01,4.3839999999999995,2.294,Åland
3 1743-12-01,,Åland
4 1744-01-01,,Åland
5 1744-02-01,,Åland
6 1744-03-01,,Åland
7 1744-04-01,1.53,4.68,Åland
8 1744-05-01,6.702000000000001,1.789,Åland
9 1744-06-01,11.609000000000002,1.577,Åland
10 1744-07-01,15.342,1.41,Åland
11 1744-08-01,,Åland
12 1744-09-01,11.702,1.517,Åland
13 1744-10-01,5.477,1.862,Åland
14 1744-11-01,3.407,1.425,Åland
15 1744-12-01,-2.181,1.641,Åland
16 1745-01-01,-3.85,1.841,Åland
17 1745-02-01,-6.574999999999998,1.36,Åland
18 1745-03-01,-4.195,1.213,Åland
19 1745-04-01,-0.9660000000000002,1.172,Åland
20 1745-05-01,,Åland
21 1745-06-01,,Åland
22 1745-07-01,,Åland
23 1745-08-01,,Åland
24 1745-09-01,,Åland
25 1745-10-01,,Åland
26 1745-11-01,,Åland
27 1745-12-01,,Åland
28 1746-01-01,,Åland
29 1746-02-01,,Åland
30 1746-03-01,,Åland
31 1746-04-01,,Åland
32 1746-05-01,,Åland
33 1746-06-01,,Åland
34 1746-07-01,,Åland
35 1746-08-01,,Åland
36 1746-09-01,,Åland
37 1746-10-01,,Åland
```

Tasks Performed :

Search for Azure Databricks in the resources :



Created Azure Databricks Workspace :

Microsoft Azure

Search resources, services, and docs (G+V)

Home > Azure Databricks >

Create an Azure Databricks workspace

Basics Networking Encryption Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Azure subscription 1

Resource group * ⓘ (New) Hexa_DE_Project
Create new

Instance Details

Workspace name * hexa_project ✓

Region * North Europe

Pricing Tier * ⓘ Premium (+ Role-based access controls)

ⓘ We selected the recommended pricing tier for your workspace. You can change the tier based on your needs.

Review + create < Previous Next : Networking >

Microsoft Azure

Search resources, services, and docs (G+/I)

azuser1046_mml.local@...
BMT (BMT.LON.MICROSOFT.COM)

Home > Azure Databricks >

Create an Azure Databricks workspace ...

Validation Succeeded

BasicsNetworkingEncryptionTagsReview + create

Summary

Basics

Workspace name

hexa_project

Subscription

Azure subscription 1

Resource group

Hexa_DE_Project

Region

North Europe

Pricing Tier

premium

Managed Resource Group name

Networking

Deploy Azure Databricks workspace with Secure Cluster Connectivity (No Public IP)

No

Deploy Azure Databricks workspace in your own Virtual Network (VNet)

No

Create< PreviousDownload a template for automation

Microsoft Azure

Search resources, services, and docs (G+/I)

azuser1046_mml.local@...
BMT (BMT.LON.MICROSOFT.COM)

Home >

Hexa_DE_Project_hexa_project | Overview ...

Deployment

Search<DeleteCancelRedeployDownloadRefresh

OverviewInputsOutputsTemplate

✓ Your deployment is complete

Deployment name : Hexa_DE_Project_hexa_project

Subscription : Azure subscription 1

Resource group : Hexa_DE_Project

Start time : 25/2/2024, 2:44:12 pm

Correlation ID : 3a132e70-181e-435a-8b8c-849a0f5f128d

> Deployment details

> Next steps

Go to resource

Give feedback

Tell us about your experience with deployment

Cost management

Get notified to stay within your budget and prevent unexpected charges on your bill.

Set up cost alerts >

Microsoft Defender for Cloud

Secure your apps and infrastructure

Go to Microsoft Defender for Cloud >

Free Microsoft tutorials

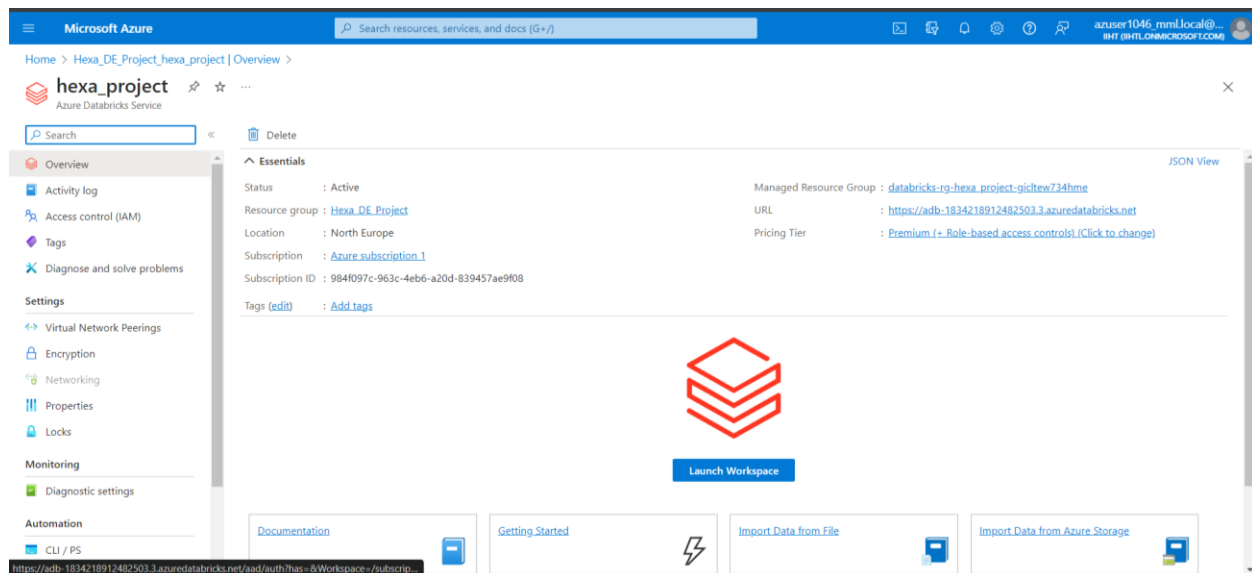
Start learning today >

Work with an expert

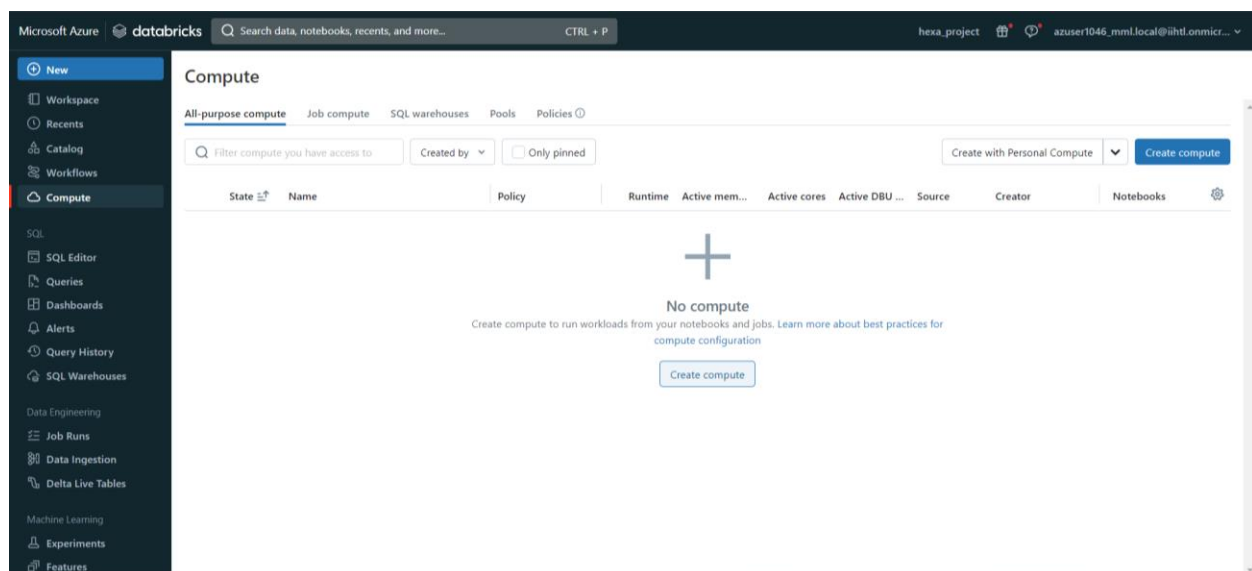
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.

Find an Azure expert >

Launch Databricks Workspace :



Created Cluster in Azure Databricks workspace :



Configuring the Cluster :

The screenshot shows the configuration page for a Databricks cluster. The left sidebar contains navigation options like Workspace, Recents, Catalog, Workflows, Compute, SQL, SQL Editor, Queries, Dashboards, Alerts, Query History, SQL Warehouses, Data Engineering, Job Runs, Data Ingestion, Delta Live Tables, Machine Learning, Experiments, and Features. The main content area is titled 'azuser1046_mml.local@iihtl.onmicrosoft.com's Cluster' and includes tabs for Configuration, Notebooks (0), Libraries, Event log, Spark UI, Driver logs, Metrics, Apps, and Spark compute UI - Master. The 'Configuration' tab is active, showing settings for Policy (Personal Compute), Access mode (Single user access), Single user (azuser1046_mml.local@iihtl.onmicrosoft.com), Performance (Databricks Runtime Version 14.3 LTS ML), Node type (Standard_DS3_v2, 14 GB Memory, 4 Cores), and Tags (No custom tags). A 'Summary' box on the right displays cluster details: 1 Driver, 14 GB Memory, 4 Cores, Runtime 14.3.x-cpu-ml-scala2.12, and a cost of 0.75 DBU/h. Buttons for 'More', 'Terminate', and 'Edit' are visible.

Created Notebook in Azure Databricks workspace :

The screenshot shows the Azure Databricks workspace interface. The left sidebar is the same as in the previous image. The main content area is titled 'azuser1046_mml.local@iihtl.onmicrosoft.com's Cluster' and shows the 'Configuration' tab. A 'New' button is visible in the top left corner of the main content area. The 'Summary' box on the right displays cluster details: 1 Driver, 14 GB Memory, 4 Cores, Runtime 14.3.x-cpu-ml-scala2.12, and a cost of 0.75 DBU/h. Buttons for 'More', 'Terminate', and 'Edit' are visible. The URL bar at the bottom shows the Databricks workspace URL.

Created Azure Data Lake Storage account

Microsoft Azure

Search resources, services, and docs (G+/)

azuser1046_mml.local@...
BHT (BHTLONMICROSOFT.COM)

Home > Storage accounts >

Create a storage account

BasicsAdvancedNetworkingData protectionEncryptionTagsReview

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *
Azure subscription 1

Resource group *
Hexa_DE_Project
[Create new](#)

Instance details

Storage account name *
adls1046project

Region *
(Europe) North Europe
[Deploy to an edge zone](#)

Review

< Previous

Next : Advanced >

[Give feedback](#)

portal.azure.com/#view/HubsExtension/DeploymentDetailsBlade/~/overview/id/%2Fsubscriptions%2F984f097c-963c-4eb6-a20d-839457ae9f08%2Fres...

Microsoft Azure

adls1046project_1709099457934 | Overview

Search

DeleteCancelRedeployDownloadRefresh

Overview

Inputs

Outputs

Template

✓ Your deployment is complete

Deployment name: adls1046project_1709099457934

Subscription: Azure subscription 1

Resource group: Hexa_DE_Project

Start time: 28/2/2024, 11:21:08 am

Correlation ID: 3695c3e6-b223-479a-b6cb-750c9403b348

Deployment details

Next steps

[Go to resource](#)

Give feedback

[Tell us about your experience with deployment](#)

Cost Management

Get notified to stay within your budget and prevent unexpected charges on your bill.
[Set up cost alerts >](#)

Microsoft Defender for Cloud

Secure your apps and infrastructure
[Go to Microsoft Defender for Cloud >](#)

Free Microsoft tutorials

[Start learning today >](#)

Work with an expert

Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
[Find an Azure expert >](#)

Created container in ADLS

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation pane is open, showing the 'Containers' section under 'Data storage'. The main area displays the 'Containers' page for the 'adls1046project' storage account. A 'New container' dialog is open on the right, showing the container name 'container' and the anonymous access level set to 'Private (no anonymous access)'. The dialog also includes a 'Create' button and a 'Give feedback' link.

Home > Hexa_DE_Project > adls1046project

adls1046project | Containers

Search

+ Container Change access level Restore containers Refresh Delete Give feedback

Search containers by prefix

Name	Last modified	Anonymous
<input type="checkbox"/> \$logs	28/2/2024, 11:21:44 am	Private

Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Storage Mover

Data storage

Containers
File shares
Queues
Tables

Security + networking

Networking
Front Door and CDN

New container

Name *

container

Anonymous access level ⓘ

Private (no anonymous access)

The access level is set to private because anonymous access is disabled on this storage account.

Advanced

Create Give feedback

Uploading Dataset in the ADLS

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation pane is open, showing the 'container' section under 'Data storage'. The main area displays the 'container' page for the 'adls1046project' storage account. A 'Upload blob' dialog is open on the right, showing the file 'GlobalLandTemperaturesByCountry.csv' selected for upload. The dialog also includes an 'Upload' button and a 'Give feedback' link.

Home > Hexa_DE_Project > adls1046project | Containers >

container

Search

Upload Change access level Refresh Delete Change tier Acquire lease

Authentication method: Access key (Switch to Microsoft Entra user account)

Location: container

Search blobs by prefix (case-sensitive)

Add filter

Name	Modified	Access tier	Access
No results			

Overview
Diagnose and solve problems
Access Control (IAM)

Settings

Shared access tokens
Access policy
Properties
Metadata

Upload blob

1 file(s) selected: GlobalLandTemperaturesByCountry.csv
Drag and drop files here or Browse for files

Overwrite if files already exist

Advanced

Upload Give feedback

Data Preparation:

- Loaded the global temperature data into Spark DataFrame from our data source Azure Blob Storage.

Load Data from Data Sources:

- For Azure Blob Storage: Use the **spark.read.format("csv").load("wasbs://<container-name>@<storage-account-name>.blob.core.windows.net/<path>")** method to load CSV files from Azure Blob Storage.
- Once the data is loaded and formatted, created a Spark DataFrame to represent the data in a structured format for further analysis.
- Used the **spark.createDataFrame()** method or DataFrame APIs to create the DataFrame from the loaded data.

```
storage_account_name = "adls1046project"
container_name = "container"
file_name = "GlobalLandTemperaturesByCountry.csv"
storage_account_access_key = "KUKAhIfCYt/M/1Co79hLgWIsSzICBxQ7TJQ3s7Dwtr01gZWe/lRA0Rh43+Qmu1+PR31WfvqtRx6w+ASthC50fg=="
spark.conf.set("fs.azure.account.key." + storage_account_name + ".blob.core.windows.net", storage_account_access_key)
df = spark.read.format("csv").load(f"wasbs://{container_name}@{storage_account_name}.blob.core.windows.net/{file_name}", inferSchema=True,
header=True)
df.show()
```

▶ (3) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [Date: date, AverageTemperature: double ... 2 more fields]

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-11-01	4.3839999999999995	2.294	Åland
1743-12-01	NULL	NULL	Åland
1744-01-01	NULL	NULL	Åland
1744-02-01	NULL	NULL	Åland
1744-03-01	NULL	NULL	Åland
1744-04-01	1.53	4.68	Åland
1744-05-01	6.7020000000000001	1.789	Åland
1744-06-01	11.6090000000000002	1.577	Åland
1744-07-01	15.342	1.41	Åland
1744-08-01	NULL	NULL	Åland
1744-09-01	11.702	1.517	Åland
1744-10-01	5.477	1.862	Åland
1744-11-01	3.407	1.425	Åland
1744-12-01	-2.181	1.641	Åland

Data Exploration:

To explore our data using Spark SQL, We followed these steps:

1. Load the data into Spark DataFrame.
2. Used Spark SQL queries to explore the data, including identifying the structure, schema, and basic statistics.

```
▶ ▾ ✓ Just now (<1s) Cell 2 Python ✨ ⌵ ⋮
```

```
# Identify the structure and schema of the dataset
print("Schema of the dataset:")
data.printSchema()
```

```
Schema of the dataset:
root
|-- Date: date (nullable = true)
|-- AverageTemperature: double (nullable = true)
|-- AverageTemperatureUncertainty: double (nullable = true)
|-- Country: string (nullable = true)
```

```
▶ ▾ Python ✨ ⌵ ⋮ Cell 4
```

```
# Use Spark SQL queries to explore the data
# Example 1: Count the number of records in the dataset
record_count = spark.sql("SELECT COUNT(*) AS record_count FROM climate_data").collect()[0]["record_count"]
print("Number of records in the dataset:", record_count)
```

```
Number of records in the dataset: 577462
```

```
▶ ▾ ✓ Just now (2s) Cell 6 Python ✨ ⌵ ⋮
```

```
# Example 3: Display distinct values in a particular column
print("Distinct Countries:")
spark.sql("SELECT DISTINCT Country FROM climate_data").show()
```

▶ (2) Spark Jobs

```
Distinct Countries:
+-----+
| Country|
+-----+
| Anguilla|
| Åland|
| Afghanistan|
| Africa|
| Algeria|
| Argentina|
| Angola|
| Baker Island|
| Albania|
| Bahamas|
| American Samoa|
| Andorra|
| Antarctica|
| Aruba|
| Azerbaijan|
| Armenia|
| Asia|
```

Just now (3s) Cell 7 Python

```
# Example 4: Calculate the average value of a numerical column
print("Average value of a of temperature of the year:")
avg_temp = spark.sql("SELECT DATE_PART('YEAR', Date), AVG(AverageTemperature) AS avg_temp FROM climate_data GROUP BY DATE_PART('YEAR', Date)
ORDER BY DATE_PART('YEAR', Date)")
avg_temp.show()
```

(2) Spark Jobs

avg_temp: pyspark.sql.dataframe.DataFrame = [date_part(YEAR, Date): integer, avg_temp: double]

Average value of a of temperature of the year:

date_part(YEAR, Date)	avg_temp
1743	5.18414
1744	9.8378975
1745	1.3871250000000004
1746	NULL
1747	NULL
1748	NULL
1749	NULL
1750	9.129352727272728
1751	9.167387499999998
1752	4.4133866666666668
1753	8.870820754716977
1754	8.822018957345971
1755	8.530536277602524
1756	9.17988625592417
1757	8.993332283464566
1758	8.13037054263566
1759	0.2612577160000001

Data Cleaning:

- Used Spark SQL queries or DataFrame operations to perform data cleaning operations to inspect the data and to handle missing values, and any inconsistencies in the data.

Cell 2

```
# Handle missing values
# Drop rows with any missing temperature values
cleaned_data = data.dropna(subset=["AverageTemperature"])
cleaned_data.show()
```

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-11-01	4.384	2.294	Åland
1744-04-01	1.53	4.68	Åland
1744-05-01	6.702	1.789	Åland
1744-06-01	11.609	1.577	Åland
1744-07-01	15.342	1.41	Åland
1744-09-01	11.702	1.517	Åland
1744-10-01	5.477	1.862	Åland
1744-11-01	3.407	1.425	Åland
1744-12-01	-2.181	1.641	Åland
1745-01-01	-3.85	1.841	Åland
1745-02-01	-6.575	1.36	Åland
1745-03-01	-4.195	1.213	Åland
1745-04-01	-0.966	1.172	Åland
1750-01-01	1.091	1.119	Åland
1750-02-01	0.809	3.353	Åland
1750-03-01	0.923	4.716	Åland
1750-04-01	3.943	1.434	Åland
1750-05-01	6.265	1.339	Åland

▶ ▾Cell 3Python

Drop rows with any missing date values
cleaned_data = cleaned_data.dropna(subset=["Date"])
cleaned_data.show()

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-11-01	4.384	2.294	Åland
1744-04-01	1.53	4.68	Åland
1744-05-01	6.702	1.789	Åland
1744-06-01	11.609	1.577	Åland
1744-07-01	15.342	1.41	Åland
1744-09-01	11.702	1.517	Åland
1744-10-01	5.477	1.862	Åland
1744-11-01	3.407	1.425	Åland
1744-12-01	-2.181	1.641	Åland
1745-01-01	-3.85	1.841	Åland
1745-02-01	-6.575	1.36	Åland
1745-03-01	-4.195	1.213	Åland
1745-04-01	-0.966	1.172	Åland
1750-01-01	1.091	1.119	Åland
1750-02-01	0.809	3.353	Åland
1750-03-01	0.923	4.716	Åland
1750-04-01	3.943	1.434	Åland
1750-05-01	6.265	1.339	Åland

▶ ▾Just now (1s)Cell 4Python

Remove rows with missing country values
from pyspark.sql.functions import col
cleaned_data = cleaned_data.filter(col("Country").isNotNull())
cleaned_data.show()

▶ (1) Spark Jobs

cleaned_data: pyspark.sql.dataframe.DataFrame = [Date: date, AverageTemperature: double ... 2 more fields]

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-11-01	4.3839999999999995	2.294	Åland
1744-04-01	1.53	4.68	Åland
1744-05-01	6.702000000000001	1.789	Åland
1744-06-01	11.609000000000002	1.577	Åland
1744-07-01	15.342	1.41	Åland
1744-09-01	11.702	1.517	Åland
1744-10-01	5.477	1.862	Åland
1744-11-01	3.407	1.425	Åland
1744-12-01	-2.181	1.641	Åland
1745-01-01	-3.85	1.841	Åland
1745-02-01	-6.574999999999998	1.36	Åland
1745-03-01	-4.195	1.213	Åland
1745-04-01	-0.9660000000000002	1.172	Åland
1750-01-01	1.0910000000000006	1.119	Åland
1750-02-01	0.8090000000000002	3.353	Åland
1750-03-01	0.9229999999999999	4.716	Åland
1750-04-01	3.943	1.4340000000000002	Åland
1750-05-01	6.2650000000000015	1.339	Åland

```
Cell 5
Python

# Drop irrelevant columns
cleaned_data = cleaned_data.drop("AverageTemperatureUncertainty")
cleaned_data.show()
```

1744-05-01	6.702	Åland
1744-06-01	11.609	Åland
1744-07-01	15.342	Åland
1744-09-01	11.702	Åland
1744-10-01	5.477	Åland
1744-11-01	3.407	Åland
1744-12-01	-2.181	Åland
1745-01-01	-3.85	Åland
1745-02-01	-6.575	Åland
1745-03-01	-4.195	Åland
1745-04-01	-0.966	Åland
1750-01-01	1.091	Åland
1750-02-01	0.809	Åland
1750-03-01	0.923	Åland
1750-04-01	3.943	Åland
1750-05-01	6.265	Åland
1750-06-01	12.408	Åland
1750-07-01	16.683	Åland

only showing top 20 rows

Error Detection:

- Utilized Spark SQL queries or built-in functions to detect errors in your data. This involved identifying inconsistencies, duplicates, or unexpected values.

```
# Check for missing values in relevant columns (e.g., Country, AverageTemperature)
missing_values = data.filter((col("Country").isNull()) | (col("AverageTemperature").isNull()))
if missing_values.count() > 0:
    print("Missing values found in the dataset:")
    missing_values.show()
else:
    print("No missing values found in the dataset.")
```

Missing values found in the dataset:

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-12-01	null	null	Åland
1744-01-01	null	null	Åland
1744-02-01	null	null	Åland
1744-03-01	null	null	Åland
1744-08-01	null	null	Åland
1745-05-01	null	null	Åland
1745-06-01	null	null	Åland
1745-07-01	null	null	Åland
1745-08-01	null	null	Åland
1745-09-01	null	null	Åland
1745-10-01	null	null	Åland
1745-11-01	null	null	Åland
1745-12-01	null	null	Åland
1746-01-01	null	null	Åland
1746-02-01	null	null	Åland
1746-03-01	null	null	Åland
1746-04-01	null	null	Åland


```
Cell 3

# Check for duplicates
duplicate_count = data.groupBy(data.columns).count().filter("count > 1").count()
if duplicate_count > 0:
    print("Duplicates found in the dataset.")
else:
    print("No duplicates found in the dataset.")
```

No duplicates found in the dataset.

```
# Check for outliers or unrealistic values in AverageTemperature
# you can set a threshold to identify outliers
min_threshold = -30
max_threshold = 50
outliers = data.filter((col("AverageTemperature") < min_threshold) | (col("AverageTemperature") > max_threshold))
if outliers.count() > 0:
    print("Outliers found in the AverageTemperature column:")
    outliers.show()
else:
    print("No outliers found in the AverageTemperature column.")
```

Outliers found in the AverageTemperature column:

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1823-02-01	-31.746	3.438	Denmark
1831-02-01	-31.28	4.364	Denmark
1832-02-01	-31.259	3.57	Denmark
1834-01-01	-31.572	3.27	Denmark
1835-02-01	-31.831	3.245	Denmark
1836-02-01	-32.625	3.677	Denmark
1838-01-01	-31.057	3.493	Denmark
1839-01-01	-31.251	3.188	Denmark
1839-12-01	-31.045	2.886	Denmark
1840-01-01	-31.082	3.549	Denmark
1841-01-01	-31.097	2.931	Denmark
1844-02-01	-31.911	3.263	Denmark
1845-02-01	-31.081	3.222	Denmark
1845-12-01	-31.152	3.11	Denmark
1848-02-01	-31.245	3.602	Denmark

Seasonality Analysis:

- Use Spark SQL functions or libraries like PySpark's pandas or numpy to analyze seasonality patterns in your data.

```
Cell 2

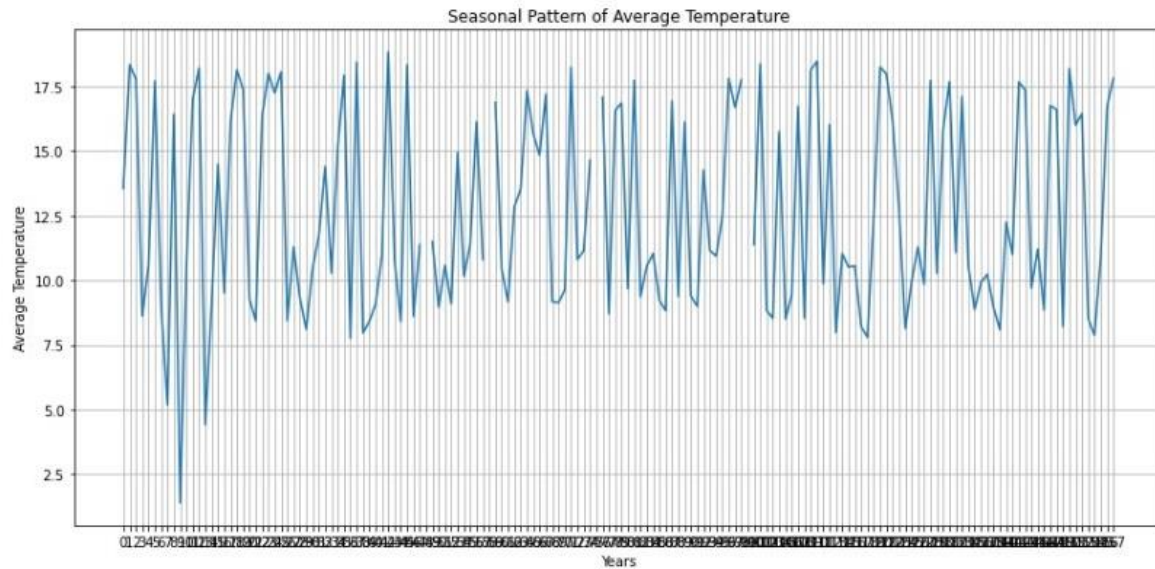
# Ensure 'Date' column is of type DateType
data = data.withColumn("Date", data["Date"].cast("date"))
data.show()
```

Date	AverageTemperature	AverageTemperatureUncertainty	Country
1743-11-01	4.384	2.294	Åland
1743-12-01	null	null	Åland
1744-01-01	null	null	Åland
1744-02-01	null	null	Åland
1744-03-01	null	null	Åland
1744-04-01	1.53	4.68	Åland
1744-05-01	6.702	1.789	Åland
1744-06-01	11.609	1.577	Åland
1744-07-01	15.342	1.41	Åland
1744-08-01	null	null	Åland
1744-09-01	11.702	1.517	Åland
1744-10-01	5.477	1.862	Åland
1744-11-01	3.407	1.425	Åland
1744-12-01	-2.181	1.641	Åland
1745-01-01	-3.85	1.841	Åland
1745-02-01	-6.575	1.36	Åland
1745-03-01	-4.195	1.213	Åland
1745-04-01	-0.966	1.172	Åland

Cell 6

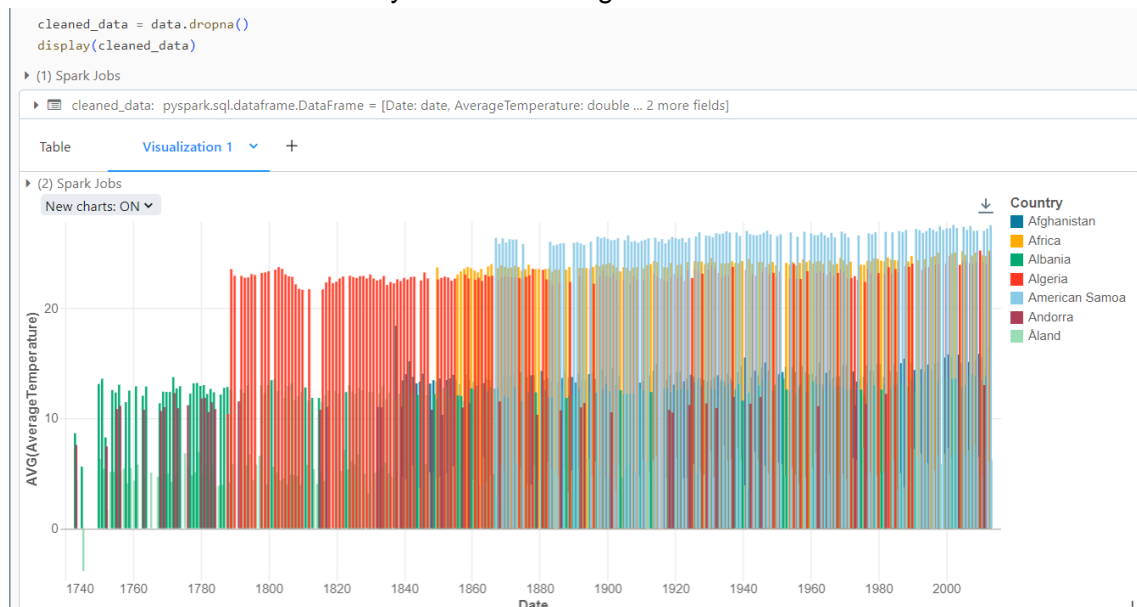
```
# Convert Spark DataFrame to Pandas DataFrame for visualization
yearly_avg_temp_pandas = yearly_avg_temp.toPandas()

# Plot the seasonal pattern of average temperature
plt.figure(figsize=(12, 6))
plt.plot(yearly_avg_temp_pandas["AvgTemperature"])
plt.title("Seasonal Pattern of Average Temperature")
plt.xlabel("Years")
plt.ylabel("Average Temperature")
plt.xticks(range(len(yearly_avg_temp_pandas)))
plt.grid(True)
plt.tight_layout()
plt.show()
```



Visualization:

- Visualized analysis results using Databricks' built-in visualization tools.



Data Analysis:

- Utilized PySparkSQL queries to analyze the data for errors, seasonality, and anomalies.
- Aggregate the data to calculate statistics such as average.

```
3 days ago (1s) Cell 3 Python
```

```
print("Total Countries:")
spark.sql("SELECT Count(DISTINCT Country) as Total_Countries FROM climate_data").show()
```

▶ (3) Spark Jobs

Distinct Countries:

Total_Countries
243

```
3 days ago (1s) Cell 4
```

```
avg_temp = spark.sql("SELECT DATE_PART('YEAR', dt) as Year, AVG(AverageTemperature) AS avg_temp_by_year, Country FROM climate_data GROUP BY DATE_PART('YEAR', dt), Country ORDER BY Country, Year")
display(avg_temp.dropna())
```

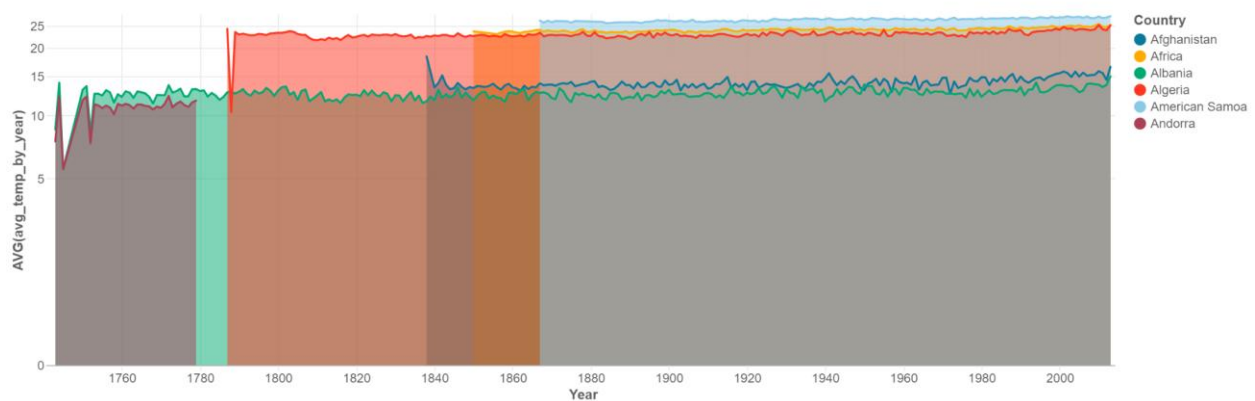
▶ (2) Spark Jobs

avg_temp: pyspark.sql.dataframe.DataFrame = [Year: integer, avg_temp_by_year: double ... 1 more field]

Table Avg_Temp_by_Year + New result table: OFF

	Year	avg_temp_by_year	Country
1	1838	18.379571428571428	Afghanistan
2	1840	13.413454545454545	Afghanistan
3	1841	13.9976	Afghanistan
4	1842	15.154666666666667	Afghanistan
5	1843	13.75625	Afghanistan
6	1844	13.148750000000001	Afghanistan
7	1845	13.305833333333332	Afghanistan

10,000 rows | Truncated data | 1.19 seconds runtime Refreshed 3 days ago



Average temperature by year

▶ 3 days ago (<1s) Cell 5

```
avg_temp.dropna().createOrReplaceTempView("Avg_temp")
```

▶ 3 days ago (1s) Cell 6 Python

```
avg_data = spark.sql("SELECT Avg(avg_temp_by_year) as avg_temp_by_Country, Country from Avg_temp GROUP BY Country ORDER BY Country")
display(avg_data)
```

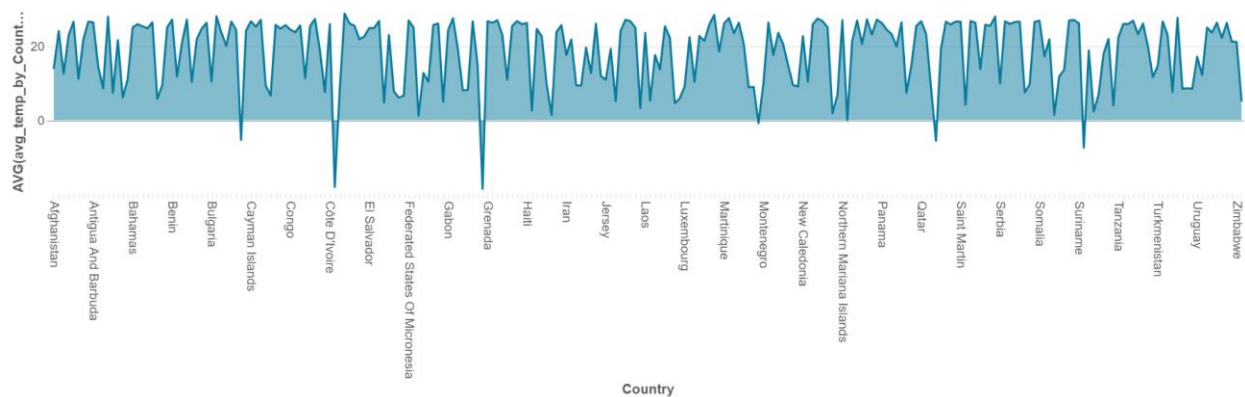
(5) Spark Jobs

avg_data: pyspark.sql.dataframe.DataFrame = [avg_temp_by_Country: double, Country: string]

Table Avg_temp_by_country + New result table: OFF

	avg_temp_by_Country	Country
1	14.061307148423012	Afghanistan
2	24.07212044978083	Africa
3	12.577367012824885	Albania
4	22.939589924576175	Algeria
5	26.602235920614174	American Samoa
6	11.171873552945188	Andorra
7	21.818182360779808	Anqola

242 rows | 1.22 seconds runtime Refreshed 3 days ago



Conclusion:

In conclusion, the project successfully achieved its objectives of analyzing the Global Land Average Temperature dataset using PySpark on Azure Databricks. By leveraging distributed computing capabilities and advanced analytics techniques, the project provided actionable insights into temperature trends and anomalies, contributing to efforts to understand and address climate change.

The project demonstrates the power of using cloud-based data analytics platforms like Azure Databricks and PySpark for processing and analyzing large-scale environmental datasets, enabling data-driven decision-making and fostering scientific research in climate science and related fields.

Reference : <https://learn.microsoft.com/en-us/azure/storage/blobs/create-data-lake-storage-account>
<https://learn.microsoft.com/en-in/azure/synapse-analytics/get-started-create-workspace>
<https://blog.arinti.be/databricks-importing-data-from-a-blob-storage-2b8dc700d029>
https://docs.databricks.com/en/_extras/notebooks/source/data-import/azure-blob-store.html