

Time Series Analysis

-Sarthak Niranjana Kulkarni (Maverick)

sarthakkul2311@gmail.com

(+91) 93256 02791

Data Engineering Project 1: *Implement time series analysis using PySparkSQL on Azure Databricks. Process and analyze time-stamped data for trends, seasonality, and anomalies.*

1. Project Overview:

This project involves analyzing temperature data over time using PySparkSQL on Azure Databricks to identify trends, seasonal patterns, and anomalies. By leveraging distributed computing with Spark, the dataset is processed efficiently, starting with data cleaning and conversion of timestamps into a proper date format. The analysis includes calculating monthly average temperatures to detect long-term trends and seasonal variations. Anomalies are identified using Z-scores, flagging extreme temperature values that deviate significantly from the norm. The results are visualized with line charts for trends, bar charts for seasonality, and scatter plots for anomalies, providing valuable insights into temperature behavior and outliers over time.

2. Data Overview:

- Data: Daily minimum temperatures (°C) from January 1, 1981.
 - Columns: Date and Temperature.
 - Issues: Inconsistent date formats (e.g., MM-DD-YY, M/D/YYYY).
 - Temperature Range: ~2.1°C to 26.3°C, showing seasonal variations.
 - Trends: Lower temperatures mid-year, higher at the start of the year.
-

3. Architecture diagram –ER diagram:



4. Source Data File:

The source data for this project was obtained from Kaggle, where the *Time Series Datasets* dataset is available. The dataset is provided in CSV (Comma Separated Values) format, which is commonly used for structured data storage and analysis.

[Download Ecommerce Dataset](#)

5. Execution Overview:

❖ Spark Session Initialization:

The objective of this step was to create a Spark session that would allow us to use Spark SQL functionality within Databricks. This session acts as the foundation for reading and processing the dataset using Spark. To achieve this, we used the command `SparkSession.builder.appName("TimeSeriesAnalysis").getOrCreate()`. This command initializes the Spark session, enabling us to perform further operations on the data.

❖ Dataset Loading and Exploration

Next, the goal was to load the dataset from a Databricks Delta table to make it available for analysis. We accomplished this by using Spark's Delta format to read the data, utilizing the command `spark.read.format('delta').load(file_path)`. After loading the data, we displayed the first five rows using `df.show(5)` to inspect the dataset's structure and get an initial sense of the columns and data types.

❖ Data Cleaning and Preprocessing

In this step, the objective was to clean and preprocess the data to ensure it was in a usable format. We started by converting the Date column to the correct data type, i.e., Date, using the `to_date()` function. This ensures that we can correctly handle and manipulate the timestamp information in our analysis. We also dropped any rows with missing values using the `dropna()` function to ensure the dataset is complete. Finally, we sorted the data by the Date column using `orderBy()` to make sure the data is in chronological order, which is important for time series analysis.

❖ Trend Analysis

The goal of this step was to extract the monthly trend of the data. To do this, we added two new columns: Year and Month, which were extracted from the Date column using the `year()` and `month()` functions. These new columns allow us to group the data by year and

month. Then, we calculated the monthly average of the Daily minimum temperatures by grouping the data by Year and Month and aggregating it with the `avg()` function. The result was displayed as the first 10 months and their corresponding average temperatures.

❖ **Seasonality Analysis**

In this step, we focused on identifying the seasonal trends in the dataset. To do this, we grouped the data by month and calculated the average of Daily minimum temperatures for each month. This step provided insights into the general seasonal patterns, with each month representing the average temperature for that period.

❖ **Anomaly Detection**

The objective here was to detect anomalies in the dataset based on the Z-score, which can identify data points that deviate significantly from the mean. First, we calculated the mean and standard deviation for the Daily minimum temperatures column. Using these values, we added a new column, `Z_score`, which represents the standardized value of each data point (i.e., how many standard deviations away from the mean the value is). Next, we filtered the dataset to identify anomalies, defined as data points where the Z-score was greater than 3 or less than -3. These anomalies were displayed for further analysis.

❖ **Visualization**

The final step involved visualizing the trends, seasonality, and anomalies in the dataset. To make this possible, we converted the Spark DataFrames (i.e., `monthly_trends`, `seasonality`, and `anomalies`) to Pandas DataFrames for easier manipulation and visualization in Python. Using `matplotlib`, we created three plots:

- **Monthly trends:** A line plot showing the trend of the average temperatures across months.
 - **Seasonality:** A bar chart displaying the average temperature for each month to highlight seasonal variations.
 - **Anomalies:** A scatter plot with red dots representing the anomalies, making it easy to identify outliers in the dataset.
-

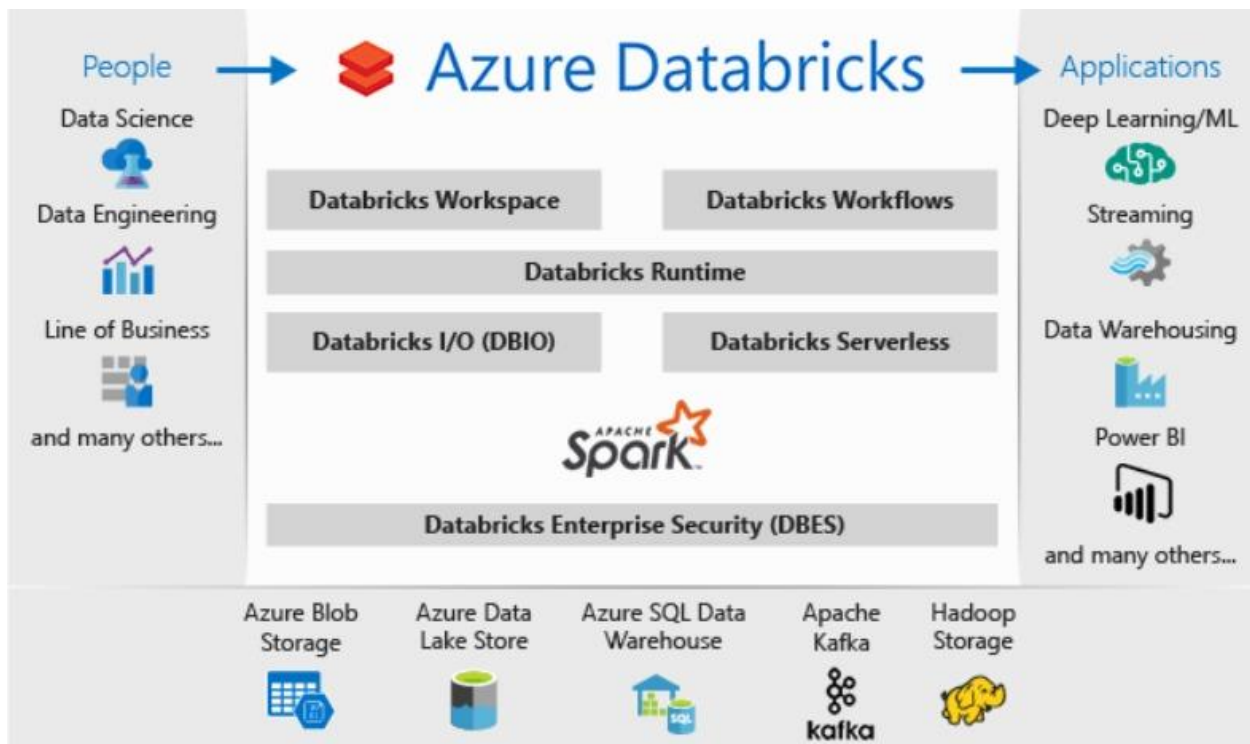
6. Azure resources used for this project:

Azure Databricks (ADB):

Azure Databricks is the core resource used in my project for performing time series analysis and data processing. It is a powerful, Apache Spark-based platform optimized for big data processing and machine learning tasks.

Key Functions:

- Data Processing
- Data Storage
- Notebooks
- Cluster Management



7. Project Requirements:

❖ Software Requirements

- Azure Databricks (ADB)
- Python (3.x)
- Apache Spark

❖ **Data Requirements**

- Temperature Time-Stamped Dataset
- Data in Delta Lake Format

❖ **Hardware Requirements**

- Databricks Cluster
-

8. Tasks Performed:

❖ **Data Acquisition**

- Downloaded the daily temperature dataset (containing timestamps) from Kaggle.
- Uploaded the dataset to Azure Databricks, ensuring compatibility with Delta Lake for efficient processing.

❖ **Environment Setup**

- Initialized a Spark Session to enable PySpark and SparkSQL functionalities for data processing.
- Configured and managed clusters in Azure Databricks for distributed data analysis.

❖ **Data Loading**

- Loaded the dataset from a Databricks Delta Table using PySpark (`spark.read.format('delta')`) for efficient querying and transformations.
- Displayed the initial few rows to inspect the structure and quality of the dataset.

❖ **Data Cleaning and Preprocessing**

- Converted the timestamp column (Date) to a standard date format using `to_date()`.
- Removed rows with missing or null values using `dropna()`.
- Sorted the dataset chronologically by the Date column to ensure time-series consistency.

❖ Time-Series Trend Analysis

- Extracted Year and Month from the Date column using PySpark functions `year()` and `month()`.
- Grouped data by Year and Month to calculate monthly averages of daily minimum temperatures.
- Displayed the results to understand the overall temperature trends.

❖ Seasonality Analysis

- Grouped data by Month to calculate average daily minimum temperatures for each month across all years.
- Identified seasonal trends by analyzing the temperature variations between months.

❖ Anomaly Detection


- Calculated the mean and standard deviation of daily minimum temperatures.
- Added a `Z_score` column to standardize temperature values.
- Filtered out anomalies where the `Z_score` was greater than 3 or less than -3 (statistical outliers).
- Displayed the detected anomalies for further interpretation.

❖ Data Visualization

- Converted PySpark DataFrames (`monthly_trends`, `seasonality`, `anomalies`) to Pandas DataFrames for visualization.
 - Used Matplotlib to create insightful charts:
 - Line Plot for trend analysis (Monthly Average Temperatures over Time).
 - Bar Chart for seasonal patterns (Average Temperature by Month).
 - Scatter Plot for anomaly detection, highlighting temperature outliers with red markers.
-

9. Analysis Result:


▶ (1) Spark Jobs


▶  df: pyspark.sql.dataframe.DataFrame = [Date: date, Daily minimum temperatures: string]

```
+-----+-----+
|      Date|Daily minimum temperatures|
+-----+-----+
|1981-01-01|                20.7|
|1981-01-02|                17.9|
|1981-01-03|                18.8|
|1981-01-04|                14.6|
|1981-01-05|                15.8|
+-----+-----+
```

only showing top 5 rows

▶ (4) Spark Jobs

▶  df: pyspark.sql.dataframe.DataFrame = [Date: date, Daily minimum temperatures: string ... 2 more fields]

▶  monthly_trends: pyspark.sql.dataframe.DataFrame = [Year: integer, Month: integer ... 1 more field]

```
+---+---+-----+
|Year|Month|   Average_Temp|
+---+---+-----+
|1990|  7| 8.183870967741935|
|1987| 10|10.238709677419354|
|1986|  1|13.825806451612904|
|1983|  3|15.777419354838708|
|1984| 12|12.643333333333334|
|1983|  4|10.596666666666668|
|1985|  3|15.877419354838711|
|1990|  4|13.433333333333332|
|1990|  6| 7.719999999999999|
|1984| 10|10.632258064516131|
+---+---+-----+
```

only showing top 10 rows

▶ (4) Spark Jobs

▶  seasonality: pyspark.sql.dataframe.DataFrame = [Month: integer, Average_Temp: double]

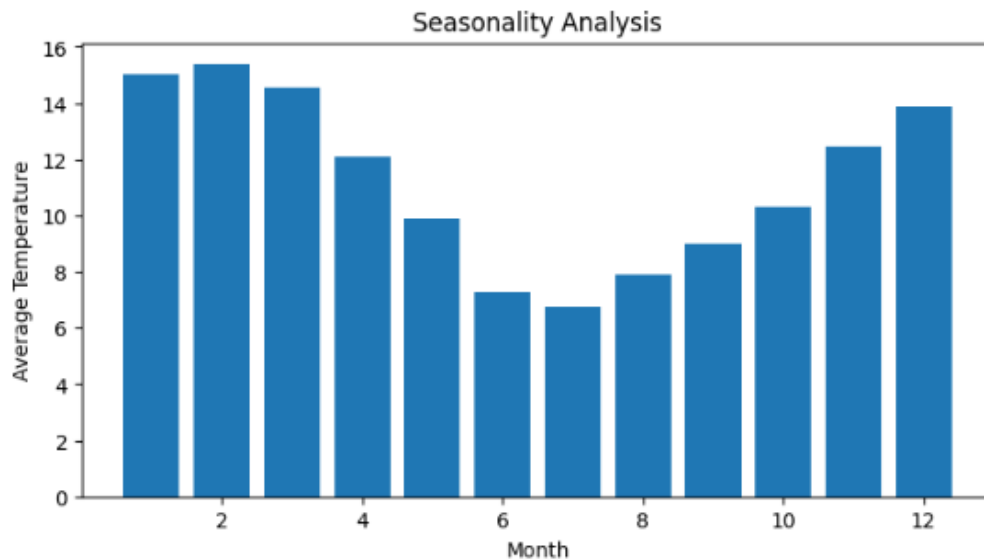
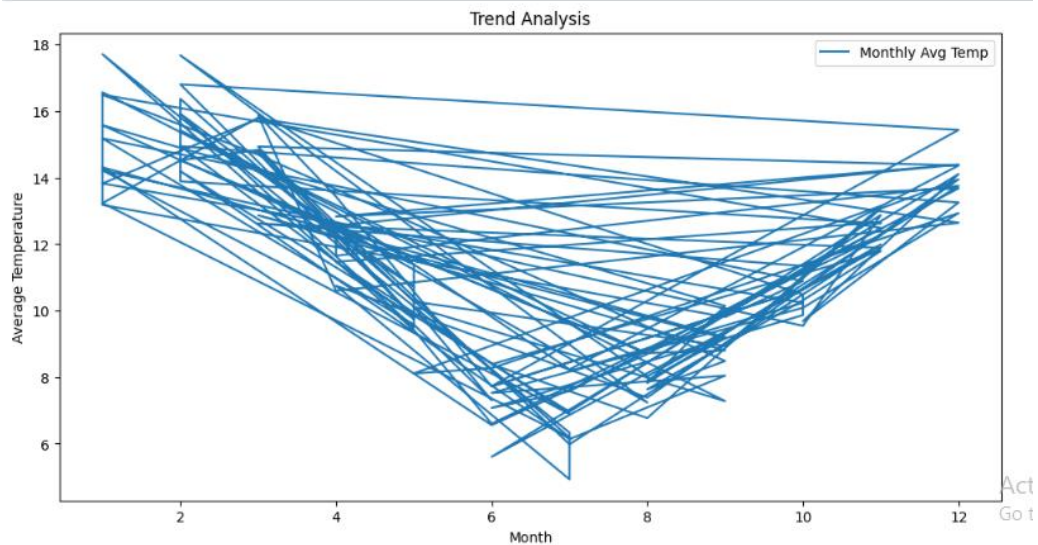
```
+---+-----+
|Month|   Average_Temp|
+---+-----+
| 12|13.851948051948042|
|  1|15.030322580645171|
|  6| 7.278333333333333|
|  3|14.565483870967752|
|  5| 9.866451612903221|
|  9| 8.976333333333334|
|  4|12.088333333333333|
|  8| 7.891290322580645|
|  7| 6.754397394136806|
| 10|10.309354838709684|
| 11|12.479666666666665|
|  2|15.373758865248226|
+---+-----+
```

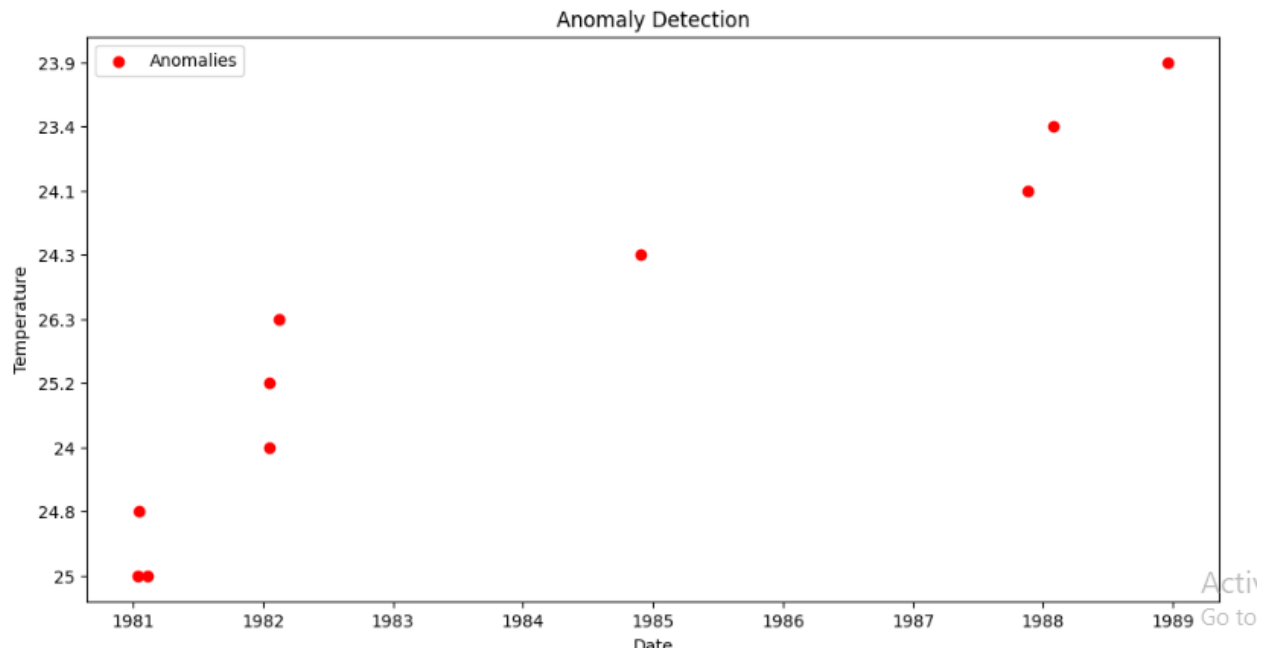
► (5) Spark Jobs

- anomalies: pyspark.sql.dataframe.DataFrame = [Date: date, Daily minimum temperatures: string ... 3 more fields]
- df: pyspark.sql.dataframe.DataFrame = [Date: date, Daily minimum temperatures: string ... 3 more fields]

Date	Daily minimum temperatures	Year	Month	Z_score
1981-01-15	25	1981	1	3.400904267096705
1981-01-18	24.8	1981	1	3.351663446877671
1981-02-09	25	1981	2	3.400904267096705
1982-01-17	24	1982	1	3.1547001660015352
1982-01-20	25.2	1982	1	3.4501450873157387
1982-02-15	26.3	1982	2	3.720969598520426
1984-11-26	24.3	1984	11	3.2285613963300865
1987-11-19	24.1	1987	11	3.179320576111053
1988-01-30	23.4	1988	1	3.006977705344433
1988-12-16	23.9	1988	12	3.130079755892018

► (3) Spark Jobs





10. Project Code:



Final-Project-1.zip

Link to whole project: https://drive.google.com/drive/folders/1qmWbWQg_EbRjeJt0PXV1-T-tyf7Mw_9m?usp=drive_link
