# SOLUTION APPROACH

Requirements:

*You are a data engineer for a renewable energy company that operates a farm of wind turbines. The turbines generate power based on wind speed and direction, and their output is measured in megawatts (MW). Your task is to build a data processing pipeline that ingests raw data from the turbines and performs the following operations: Cleans the data: The raw data contains missing values and outliers, which must be removed or imputed. Calculates summary statistics: For each turbine, calculate the minimum, maximum, and average power output over a given time period (e.g., 24 hours). Identifies anomalies: Identify any turbines that have significantly deviated from their expected power output over the same time period. Anomalies can be defined as turbines whose output is outside of 2 standard deviations from the mean. Stores the processed data: Store the cleaned data and summary statistics in a database for further analysis. Data is provided to you as CSVs which are appended daily. Due to the way the turbine measurements are set up, each csv contains data for a group of 5 turbines. Data for a particular turbine will always be in the same file (e.g. turbine 1 will always be in data\_group\_1.csv). Each day the csv will be updated with data from the last 24 hours, however the system is known to sometimes miss entries due to sensor malfunctions. The files provided in the attachment represent a valid set for a month of data recorded from the 15 turbines. Feel free to add/remove data from the set provided in order to test/satisfy the requirements above. Your pipeline should be scalable and testable; emphasis is based on the clarity and quality of the code and the implementation of the functionality outlined above, and not on the overall design of the application. Your solution should be implemented in Python, using any frameworks or libraries that you deem appropriate. Please provide a brief description of your solution design and any assumptions made in your implementation.*

## Objective

Build a scalable, automated, and reliable data pipeline that:

1. Ingests **daily CSV files** from 15 turbines grouped into 3 files.
2. **Cleans the data** (handles missing values and outliers).
3. **Calculates daily statistics** (min, max, average).
4. **Identifies anomalies** (based on 2 standard deviations from the mean).
5. **Stores processed results** for reporting and dashboarding (e.g., Power BI).

## Assumptions:

1. Daily CSVs are appended for each of the 3 turbine groups (data\_group\_1.csv, etc.).
2. Each turbine is always in the same file and group.
3. There may be missing or corrupt rows due to sensor malfunctions.
4. File schema is consistent.
5. Reports are needed daily by 8 AM.
6. No Need to archive as file names will be different each time

## High-Level Architecture

[ Wind Turbines ]

⬇

[ Azure Blob Storage - raw-data container ]

⬇

[ Azure Event Grid Trigger ]

⬇

[ Azure Data Factory ]

└── Ingest & Process Pipeline

├─ Identify File Group via Filename Pattern

├─ Clean Data (Databricks/ADF)

├─ Calculate Summary Stats

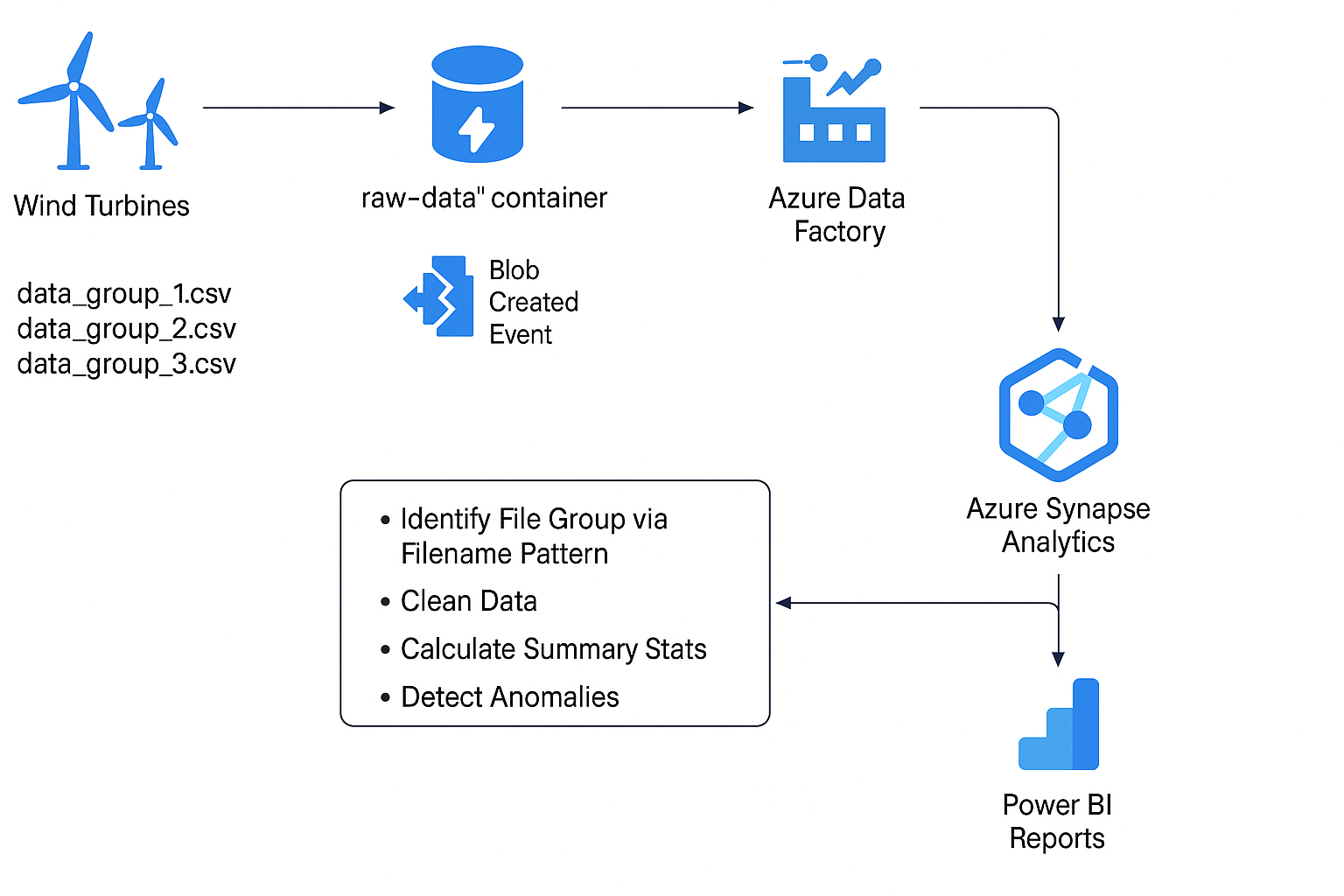
└─ Detect Anomalies

⬇

[ Azure Synapse Analytics ]

⬇

[ Power BI Reports ]



### Azure Storage

* + **Storage Account**: windturbinedatastorage (type blob storage)
  + **Container**: raw-data
  + **Folder Structure**:
    - Data/ 🡪 file will be picked on pattern
  + **Files**:
    - data\_group\_1.csv
    - data\_group\_2.csv
    - data\_group\_3.csv

### Trigger Design – Event Based

1. **Service**: Azure **Event Grid**
2. **Event Type**: Blob Created
3. **Scope**: raw-data container
4. **Action**: Triggers ADF pipeline on new file upload
5. **Trigger Logic** (via ADF pipeline parameter):
   * 1. Extract filename
     2. Route processing logic based on data\_group\_n.csv using a Switch or metadata-driven design

### ADF Pipelines

**Main Pipeline (PL\_MainProcessOnEvent)**

* + 1. **Trigger**: Event Grid → new file in raw-data
    2. **Extract Filename** from event payload
    3. **Metadata Activity**: Determine group ID from filename
    4. **Data Flow / Databricks Notebook**:
       1. Read file based on path (e.g., *@concat('https://<storage>.blob.core.windows.net/raw-data/', pipeline().parameters.filename))*
       2. Clean missing/null values
       3. Remove outliers using Z-score or IQR
       4. Calculate daily min, max, avg
       5. Detect anomalies (outside ±2 std deviation)
    5. **Sink to Synapse Tables**

### Data Modeling (Azure Synapse DDLs)

**1. CleanedTurbineData**

*CREATE TABLE dbo.CleanedTurbineData (*

*Timestamp DATETIME,*

*TurbineID VARCHAR(50),*

*WindSpeed FLOAT,*

*WindDirection FLOAT,*

*PowerOutput FLOAT*

*);*

**2. TurbineSummaryStats**

*CREATE TABLE dbo.TurbineSummaryStats (*

*Date DATE,*

*TurbineID VARCHAR(50),*

*MinPower FLOAT,*

*MaxPower FLOAT,*

*AvgPower FLOAT*

);

**3. TurbineAnomalies**

*CREATE TABLE dbo.TurbineAnomalies (*

*Date DATE,*

*TurbineID VARCHAR(50),*

*AvgPower FLOAT,*

*GroupMean FLOAT,*

*StdDev FLOAT,*

*IsAnomaly BIT*

*);*

#### LOGICS

1. For turbine statistics

*SELECT*

*turbine\_id,*

*CAST(timestamp AS DATE) AS reading\_date,*

*MIN(power\_output) AS min\_output,*

*MAX(power\_output) AS max\_output,*

*AVG(power\_output) AS avg\_output,*

*COUNT(\*) AS total\_records*

*FROM cleaned\_turbine\_data*

*GROUP BY turbine\_id, CAST(timestamp AS DATE)*

1. Calculate mean and standard deviation for each turbine (daily):

*SELECT*

*turbine\_id,*

*CAST(timestamp AS DATE) AS reading\_date,*

*AVG(power\_output) AS mean\_output,*

*STDEV(power\_output) AS stddev\_output*

*FROM cleaned\_turbine\_data*

*GROUP BY turbine\_id, CAST(timestamp AS DATE)*

1. Join back to raw data and flag anomalies:

*SELECT*

*d.\*,*

*CASE*

*WHEN ABS(d.power\_output - s.mean\_output) > 2 \* s.stddev\_output THEN 'Anomaly'*

*ELSE 'Normal'*

*END AS anomaly\_flag*

*FROM cleaned\_turbine\_data d*

*JOIN summary\_stats s*

*ON d.turbine\_id = s.turbine\_id AND CAST(d.timestamp AS DATE) = s.reading\_date*

### Reporting Layer (optional)

* Connect **Power BI** to Synapse views:
* vw\_DailyPerformance
* vw\_AnomalousTurbines
* Enable stakeholders to:
* Monitor daily turbine output
* Identify turbines needing maintenance
* Visualize long-term performance trends

### **Scalability Considerations**

| **Feature** | **Scalability Detail** |
| --- | --- |
| **Storage** | Azure Blob handles PB-scale data |
| **ADF** | Serverless and scalable orchestration |
| **Synapse** | MPP engine supports analytical scale |
| **Modular Design** | Easy to plug in new data groups or models |
| **Partitioned Tables** | Based on TurbineID or Date |
| **Parallel Pipelines** | Process group-wise for better performance |

### **Monitoring & Logging**

* ADF pipeline logs and failure notifications via Azure Monitor
* Log anomalies in a separate log table
* Set alerts for 0 or unusually high/low power outputs