# Data Engineering Homework Assignment

## Objective

Your task is to design and implement a scalable data pipeline using AWS to simulate a real-world data engineering problem. The goal is to ingest, process, and store data in a manner that can handle real-time streams of data efficiently.

## Problem Statement

You are tasked with building a data processing pipeline for a mock renewable energy company. The pipeline will ingest energy generation and consumption data from multiple sites, process the data to identify anomalies, and store the processed data in a database for analysis.

The pipeline should:
1. Simulate a live data feed by continuously uploading new JSON files to an S3 bucket.
2. Process incoming data in real-time whenever a new file is uploaded.
3. Store the processed data in a DynamoDB table.
4. Provide a way to visualize trends in the processed data.
5. Create APIs to allow querying data stored in DynamoDB.

## Requirements

### 1. Simulated Data Feed

- Generate random data for multiple energy sites with the following fields:
  - `site_id` (string): Unique identifier for a site.
  - `timestamp` (string): UTC timestamp of the record.
  - `energy_generated_kwh` (float): Energy generated in kWh.
  - `energy_consumed_kwh` (float): Energy consumed in kWh.
- Continuously upload JSON files containing this data to an S3 bucket every 5 minutes.

### 2. Data Processing

- Process each uploaded file using an AWS Lambda function.
- For each record in the JSON file:
  - Calculate `net_energy_kwh` = `energy_generated_kwh` - `energy_consumed_kwh`.
  - Identify anomalies where:
    - `energy_generated_kwh < 0`
    - `energy_consumed_kwh < 0`.

### 3. Data Storage

- Store the processed data in a DynamoDB table with the following schema:
  - **Partition Key**: `site_id`
  - **Sort Key**: `timestamp`
  - Other attributes:

- `energy_generated_kwh`
- `energy_consumed_kwh`
- `net_energy_kwh`
- `anomaly` (boolean)

#### 4. Data Visualization

- Create visualizations of the processed data to showcase trends and insights. Examples include:
  - Energy generation vs. consumption per site.
  - Distribution of anomalies across sites.
  - Energy trends over time.
- The visualization can be implemented using Python (e.g., Matplotlib, Seaborn, or Plotly).

#### 5. APIs for Querying Data

- Create APIs to allow querying the DynamoDB table. Endpoints could include:
  - Fetch records for a specific site and time range.
  - Retrieve all anomalies for a given site.
- Use any framework of your choice (e.g., FastAPI, Flask).

## Extra Credit

1. Automated Alerting for Anomalies: Implement real-time alerts whenever anomalies are detected in the data.

2. GitHub CI/CD: Automate the deployment process using a CI/CD pipeline integrated with GitHub.

3. Error Handling for Failure Cases: Implement robust error handling to gracefully manage and log failures in the pipeline.

## Deliverables

1. Code:
- Python code for:
  - Simulating the data feed.
  - AWS Lambda function for processing.
  - Visualizing the processed data.
  - API endpoints for querying the DynamoDB table.
- Terraform or AWS CLI scripts to deploy the infrastructure (S3, DynamoDB, Lambda, IAM roles, and event triggers).

2. Documentation:
- Detailed instructions on how to set up and run the pipeline.
- Explanation of design decisions.

3. Optional:
- Implementation details for any of the extra credit features.

## Evaluation Criteria

1. Completeness: Does the pipeline meet all requirements?
2. Scalability: Is the solution scalable for larger data volumes?
3. Code Quality: Is the code clean, efficient, and well-documented?
4. Infrastructure: Is the infrastructure correctly set up using Terraform or AWS CLI?
5. Innovation: Are any extra credit features or additional optimizations implemented?

## Additional Information: Free Tier Usage and Cost Disclaimer

This assignment can be completed using the AWS Free Tier, which provides sufficient resources to handle the tasks without incurring costs if used responsibly. However, please ensure you monitor your AWS Free Tier usage and delete all resources after completing the assignment to avoid unnecessary charges.

1. S3: Free Tier includes 5 GB of storage and 2,000 PUT requests per month.
2. DynamoDB: Free Tier includes 25 GB of storage, 25 Write Capacity Units (WCUs), and 25 Read Capacity Units (RCUs).
3. Lambda: Free Tier includes 1 million requests and 400,000 GB-seconds of compute time per month.
4. CloudWatch Logs: Free Tier includes 5 GB of log ingestion and storage per month.

### Cost Disclaimer

**We are not responsible for any AWS costs incurred during this assignment**. It is your responsibility to monitor usage and adhere to the AWS Free Tier limits. Exceeding the Free Tier or leaving resources active after completion may result in charges for which you will be solely responsible.

## Post-Assignment Resource Teardown

After completing the assignment, ensure you delete all AWS resources to avoid incurring unnecessary costs. Use the following checklist to remove resources:
- Delete the S3 bucket and its contents.
- Delete the DynamoDB table.
- Delete the Lambda function and associated logs.
- Delete IAM roles and policies.
- If using Terraform, run `terraform destroy` to clean up resources.

## Submission

Submit the following:
1. A GitHub repository containing:
   - All code and configuration files.
   - A README file with setup and deployment instructions.
2. A short write-up explaining your solution and design decisions.