Assignment1: Spark Data Processing Orchestrated with Airflow

Objective

The goal of this assignment is to help you learn the basics of **Apache Spark** and **Apache Airflow** by building a simple end-to-end data pipeline.
You will use **Spark** to clean and transform data, and **Airflow** to run, schedule, and monitor the Spark job. By completing this assignment, you will understand how data flows through a pipeline and how orchestration works in a real-world setup.

Dataset

Use the **New York City Taxi Trip Records** dataset, which is publicly available.

- Dataset link: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- Format: Parquet
- Suggested file: Yellow Taxi Trip Records (any one month)

This dataset contains trip-level information such as pickup time, drop-off time, passenger count, trip distance, and fare details.

Task Description

1. **Input Data**
    a. Download one month of the NYC Taxi trip data in **Parquet format**.
    b. Store the file in the configured storage location (local filesystem or object storage).
2. **Create a Spark Job**
    a. Write a Spark job that:
        i. Reads the Parquet file.
        ii. Performs basic data cleaning:
            1. Convert column names to lowercase.
            2. Remove rows with null or invalid values in key columns (for example, pickup time or trip distance).
        iii. Writes the cleaned data to an output location as a new dataset.
    b. Ensure the Spark job can be run independently using `spark-submit`.
3. **Create an Airflow DAG**
    a. Create or update an Airflow DAG that:
        i. Runs the Spark job.
        ii. Defines clear task order and dependencies.

        iii. Configures retries in case the Spark job fails.
        iv. Can be triggered manually or on a schedule.

4. **Trigger the Pipeline**
    a. Configure the DAG so it can be **triggered using an API or from the Airflow UI**.
    b. When triggered, the DAG should execute the Spark job and complete the data processing flow.
5. **Share Information Between Tasks**
    a. Use **XComs** to share simple details such as:
        i. Number of records read from the input file.
        ii. Number of records written to the output dataset.
        iii. Job status (success or failure).
6. **Understand the Platform**
    a. Review existing Airflow DAGs and shared components in the platform.
    b. Follow established patterns for configuration, logging, and folder structure.

Deliverables

- Spark job code.
- Airflow DAG code.
- A short README explaining:
    - What the dataset is.
    - What transformations are performed.
    - How to run the Spark job directly and through Airflow.
- Logs or screenshots showing a successful pipeline run.

Assignment 2: Incremental Spark Pipeline with Data Quality Checks Orchestrated by Airflow

Objective

The goal of this assignment is to **deepen your understanding of Spark and Airflow** by building a more realistic data pipeline. You will process data incrementally, add basic data quality checks, and manage multiple steps using Airflow. By the end of this assignment, you should be comfortable running and debugging a multi-step Spark pipeline on your laptop.

Dataset

Continue using the **NYC Taxi Trip Records** dataset (Parquet format).

- Use **two or more months** of data.
- Each month should be stored as a separate Parquet file.

Task Description

1. **Local Setup**
    a. Run everything locally on your laptop:
        i. Spark in local mode.
        ii. Airflow using LocalExecutor or SequentialExecutor.
    b. No cloud services are required.
2. **Incremental Data Processing with Spark**
    a. Modify your Spark job to:
        i. Read multiple Parquet files (one per month).
        ii. Process data **incrementally** based on pickup date (for example, process only new months).
        iii. Maintain a simple checkpoint or state file that tracks which months have already been processed.
3. **Additional Transformations**
    a. Enhance the Spark transformations:
        i. Filter out trips with invalid values (e.g., negative trip distance or fare).
        ii. Add derived columns, such as:
            1. Trip duration (drop-off time − pickup time).
            2. Average speed (distance / duration).
        iii. Partition the output data by pickup date (or month).
4. **Data Quality Checks**
    a. Add a separate Spark step (or logic) to:

      i. Validate that record counts are greater than zero.
      ii. Ensure key columns do not contain nulls after cleaning.
   b. Fail the job if data quality checks do not pass.

5. **Airflow DAG Enhancement**
   a. Create an Airflow DAG that:
      i. Runs the Spark processing job.
      ii. Runs data quality checks as a separate task.
      iii. Stops the pipeline if quality checks fail.
      iv. Uses XComs to pass metrics such as:
         1. Records processed per month.
         2. Records rejected due to validation.
      v. Includes retries and clear task dependencies.
6. **Triggering and Re-runs**
   a. Allow the DAG to be:
      i. Triggered manually from the Airflow UI.
      ii. Re-run safely without reprocessing already processed data.
7. **Logging and Debugging**
   a. Add meaningful logs in the Spark job to show:
      i. Which months are being processed.
      ii. Record counts before and after filtering.
   b. Ensure logs are visible from Airflow task logs.