# Weather Data Pipeline Project Guide

## Project Overview

Goal: Analyze whether it rains more today than 50 years ago by collecting, transforming, and visualizing historical weather data.

## Data Source

- Provider: OpenWeather API

- Format: JSON → converted to CSV using Python scripts

## AWS S3: Data Storage

- Upload weather CSV files to a private S3 bucket.

- Bucket used for archiving raw data.

## AWS RDS (PostgreSQL) Setup

- Create PostgreSQL DB instance.

- Create user `dataengineer\_user` and grant privileges.

Issues:

- ❌ Permission denied for schema public: solved by GRANTing schema permissions.

- ❌ UndefinedTable: ensured correct schema and search\_path.

## Database Tables

Staging Table: `stg\_weather\_data`

Used for raw ingestion.

Dimension Table: `dim\_location`

City-based deduplicated locations.

Fact Table: `fact\_weather`

Daily aggregated weather metrics with a foreign key to `dim\_location`.

Design Tools:

- DBML for modeling

- dbdiagram.io to visualize relationships

## Python ETL Script

Function: `load\_to\_staging()`

Responsibilities:

- Read CSV

- Clean `dt\_iso` format

- Convert `NaN` to `None` or `0` where appropriate (e.g., `rain\_1h`)

- Batch insert using `execute\_batch`

Issues Faced:

- ❌ NaN caused integer out of range: fixed with `fillna(0)`

- ❌ Permission denied for sequence: fixed with GRANT USAGE

- ❌ Duplicate data insertions: solution → truncate or check before insert

## dbt (Data Build Tool)

Used to transform staging data into dimensional models.

Setup:

- Installed via pip with a virtualenv

- Profile configured with AWS RDS credentials

Models:

- `stg\_weather\_data` (view)

- `dim\_location` (table with `row\_number()`)

- `fact\_weather` (aggregated by `DATE(dt\_iso)`)

Issues:

- ❌ column must appear in GROUP BY: resolved by adding to GROUP BY clause

- ❌ Protobuf version errors: fixed by downgrading protobuf to v5

## Airflow

Used for orchestration.

DAGs Created:

- `load\_weather\_data`: runs Python ETL script

- `clear\_stg\_weather\_data`: truncates staging table

- `load\_to\_dim\_fact`: runs dbt to populate dimension and fact tables

Issues:

- ❌ DAG not showing up: solved by setting `AIRFLOW\_HOME` and restarting scheduler

- ❌ ModuleNotFoundError: pandas: fixed by installing dependencies in Airflow virtualenv

## Power BI (Final Analysis)

Used a Windows machine to:

- Connect to RDS PostgreSQL

- Visualize daily and yearly trends

- Created metrics like `Rainy Days per Year`

Common DAX Issues:

- ❌ A single value for column X cannot be determined: resolved by using aggregation (e.g., MAX, SUM)

## Deployment & Sharing

- Used Tableau Public for Mac experimentation

- Power BI on Windows for main analysis

- Data exported as CSV, zipped if larger than 10MB

## Next Steps

- Automate daily ingestion with Airflow schedule

- Add anomaly detection (e.g., unusual temperature drops)

- Publish results on LinkedIn with diagrams and insights

## Diagram

A flowchart was created to summarize the data journey:

OpenWeather → Python → AWS S3 → RDS (PostgreSQL) → dbt + Airflow → Power BI

✅ Project successfully implemented and running locally with cloud-hosted PostgreSQL database.