

Course: Data Engineering

Final Project Report

Prepared by: Georges Assaf

Project Background

The goal is to develop an end-to-end data engineering solution that automates data extraction, transformation, and loading processes using Apache Airflow, ensuring efficiency, reliability, and scalability. The processed data will be stored in MongoDB, a NoSQL database optimized for flexible and high-performance data storage.

To bring insights to life, the project includes an interactive dashboard built with Dash, allowing users to explore relationships between weather conditions (such as temperature and humidity) and sales trends across different store locations.

Scope of work

The project is divided into three milestones:

1. Data Extraction & Initial Visualization (Week 1)
 - Extract sales data from a CSV file.
 - Develop an initial Dash visualization showing total sales by store location.
2. Data Integration & MongoDB Operations (Week 2)
 - Store sales data in MongoDB and perform CRUD operations.
 - Integrate MongoDB data and enhance Dash visualizations.
3. Full Automation & Final Submission (Week 3)
 - Automate the ETL pipeline using Apache Airflow.
 - Implement error handling and logging mechanisms.
 - Build a comprehensive interactive Dash dashboard which visualize the data from MongoDB.

Project Deliverables

The project deliverables are the followings:

- A fully functional ETL pipeline implemented as a DAG file in Python, automating data extraction, transformation, and loading.
- A Dash application, developed in Colab or Jupyter Notebook, will provide interactive visualizations of the integrated sales and weather data.
- A recorded demo video will showcase the system in action, demonstrating its features and insights.
- A comprehensive project project

Deliverables Description

ETL Pipeline (DAG file)

The DAG file name is `DEProject_etl_pipeline_mongodb.py` (Submitted as part of the deliverables)

In this DAG file I started by importing the required python libraries followed by a definition to setup the logging (location of log file, format,etc.)

Next, I defined the definition `fetch_weather_data` that takes as input 3 arguments, city, date and api_key) and return the weather conditions temperature , humidity and weather description

In addition, this file contains the ETL pipeline which contains three main definitions which are:

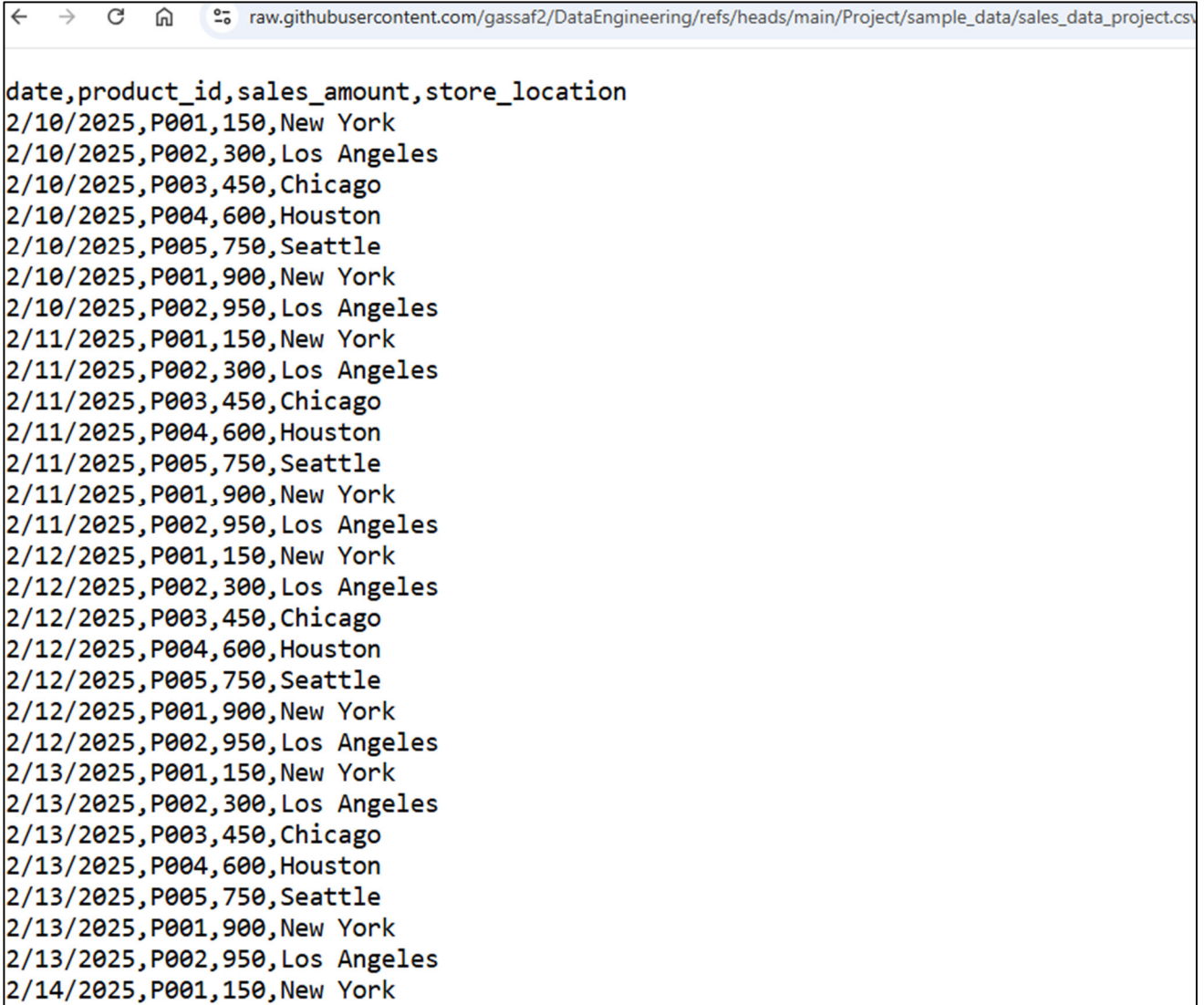
1. **extract_data:** It extracts data from csv file available on github

2. **transform_data**: this task of the pipeline its used to transform the extracted data and add the weather conditions temperature, humidity and weather description
3. **load_data**: this task is used to load the transformed data to a MongoDB "weather_db" to a collection called "sales_weather"

You can find detailed comments on the code in the `DEProject_etl_pipeline_mongodb.py` file.

Dataset Used

I used the latest dataset shared and has the below format(I read it from github repository)



```
date,product_id,sales_amount,store_location
2/10/2025,P001,150,New York
2/10/2025,P002,300,Los Angeles
2/10/2025,P003,450,Chicago
2/10/2025,P004,600,Houston
2/10/2025,P005,750,Seattle
2/10/2025,P001,900,New York
2/10/2025,P002,950,Los Angeles
2/11/2025,P001,150,New York
2/11/2025,P002,300,Los Angeles
2/11/2025,P003,450,Chicago
2/11/2025,P004,600,Houston
2/11/2025,P005,750,Seattle
2/11/2025,P001,900,New York
2/11/2025,P002,950,Los Angeles
2/12/2025,P001,150,New York
2/12/2025,P002,300,Los Angeles
2/12/2025,P003,450,Chicago
2/12/2025,P004,600,Houston
2/12/2025,P005,750,Seattle
2/12/2025,P001,900,New York
2/12/2025,P002,950,Los Angeles
2/13/2025,P001,150,New York
2/13/2025,P002,300,Los Angeles
2/13/2025,P003,450,Chicago
2/13/2025,P004,600,Houston
2/13/2025,P005,750,Seattle
2/13/2025,P001,900,New York
2/13/2025,P002,950,Los Angeles
2/14/2025,P001,150,New York
```

Error Handling Types

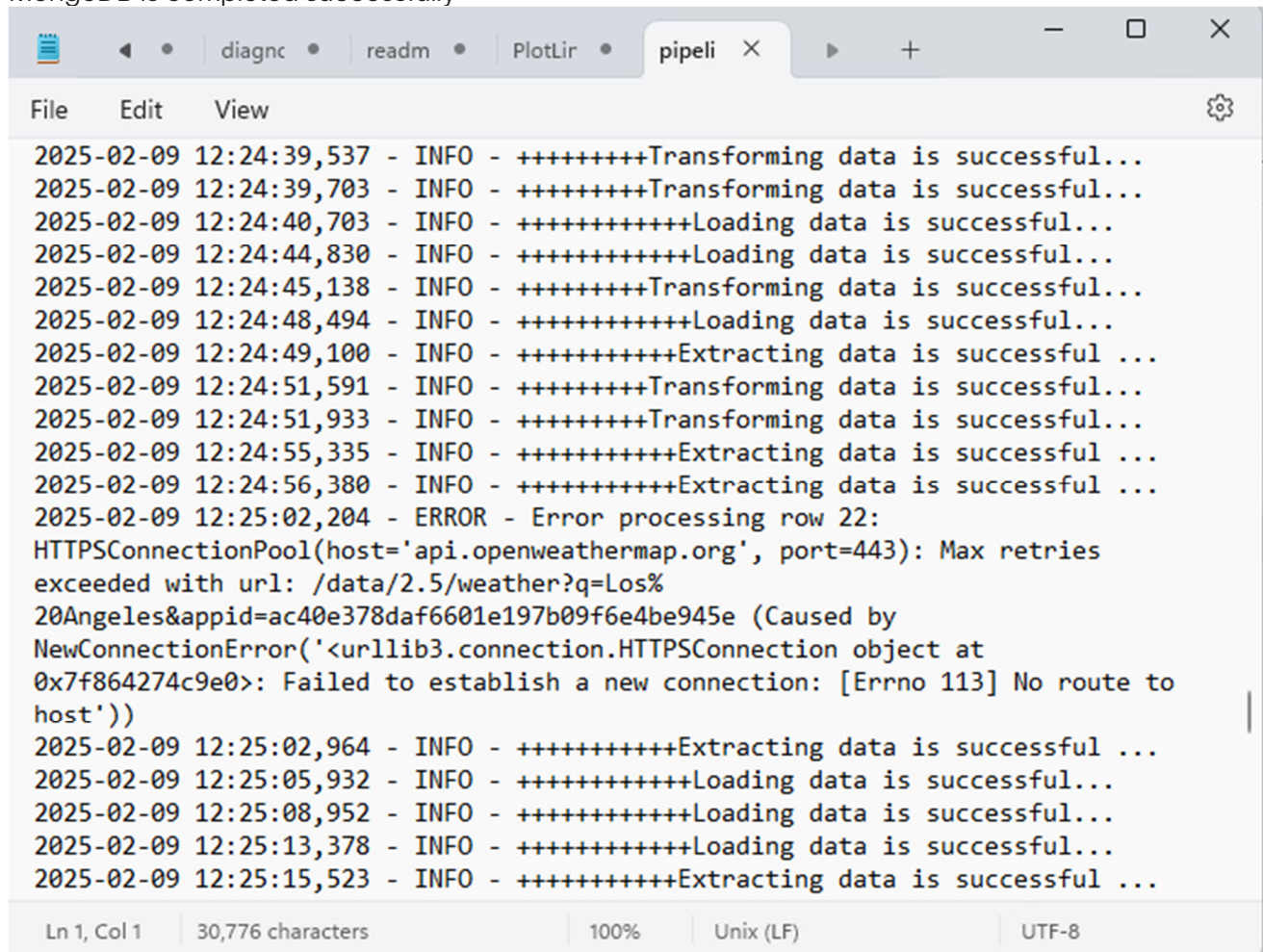
Different types of error handling are called through **exception handling** mechanisms in Python using try-except blocks. Below are the different types of error handling I used in this project

1. **urllib.error.HTTPError** :Catches HTTP errors when making a request to a URL.
2. **urllib.error.URLError**: Handles URL related errors, such as connection failures or invalid URLs.

3. **ConnectionError:** Handles network-related connection failures. It is raised when a request to a server fails due to issues like no internet, server is down
4. **Exception:** Handles general exception handler for unexpected errors that are not covered by the other cases.

Pipeline Logs Generated

The below pipeline.log shows that logging for error and for info is working fine. As we see in the error, the pipeline tries the maximum number of retries to reach and get the weather from the API but it failed. Other INFO messages confirm that task like extracting data, transforming the data and loading to MongoDB is completed successfully



```
2025-02-09 12:24:39,537 - INFO - ++++++Transforming data is successful...
2025-02-09 12:24:39,703 - INFO - ++++++Transforming data is successful...
2025-02-09 12:24:40,703 - INFO - ++++++Loading data is successful...
2025-02-09 12:24:44,830 - INFO - ++++++Loading data is successful...
2025-02-09 12:24:45,138 - INFO - ++++++Transforming data is successful...
2025-02-09 12:24:48,494 - INFO - ++++++Loading data is successful...
2025-02-09 12:24:49,100 - INFO - ++++++Extracting data is successful ...
2025-02-09 12:24:51,591 - INFO - ++++++Transforming data is successful...
2025-02-09 12:24:51,933 - INFO - ++++++Transforming data is successful...
2025-02-09 12:24:55,335 - INFO - ++++++Extracting data is successful ...
2025-02-09 12:24:56,380 - INFO - ++++++Extracting data is successful ...
2025-02-09 12:25:02,204 - ERROR - Error processing row 22:
HTTPSConnectionPool(host='api.openweathermap.org', port=443): Max retries
exceeded with url: /data/2.5/weather?q=Los%
20Angeles&appid=ac40e378daf6601e197b09f6e4be945e (Caused by
NewConnectionError('<urllib3.connection.HTTPSConnection object at
0x7f864274c9e0>: Failed to establish a new connection: [Errno 113] No route to
host'))
2025-02-09 12:25:02,964 - INFO - ++++++Extracting data is successful ...
2025-02-09 12:25:05,932 - INFO - ++++++Loading data is successful...
2025-02-09 12:25:08,952 - INFO - ++++++Loading data is successful...
2025-02-09 12:25:13,378 - INFO - ++++++Loading data is successful...
2025-02-09 12:25:15,523 - INFO - ++++++Extracting data is successful ...
```

Appendix

All 74	Active 2	Paused 72	Running 0	Failed 0	Filter DAGs by tag	Search DAGs	Auto-refresh	
DAG ↕	Owner ↕	Runs	Schedule	Last Run ↕	Next Run ↕	Recent Tasks	Actions	Links
DEProject_etl_pipeline_mongodb	airflow		@ 6***	2025-02-09, 08:48:34	2025-02-09, 06:00:00			...
etl_pipeline_mongodb	airflow		@ 6***	2025-02-08, 06:00:00	2025-02-09, 06:00:00			...

Showing 1-2 of 2 DAGs



Mongo DB Collection with Loaded Data

Atlas

LAU

Access Manager

Billing

blog_platformData ServicesCharts

Overview

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

Goto

LAU > BLOG_PLATFORM > DATABASES

Cluster0

OverviewReal TimeMetricsCollectionsAtlas SearchPerformance AdvisorOnline ArchiveCmd Line ToolsInfrastructure As Code

DATABASES: 17COLLECTIONS: 39

+ Create Database

Search Namespaces

db_blog_platform

e-commerce

health_db

retail_db

retails_db

sales_db

sample_airbnb

sample_analytics

sample_geospatial

sample_guides

sample_mflix

sample_restaurants

sample_supplies

sample_training

sample_weatherdata

test_dog

weather

sales

sales_weather

weather.sales_weather

STORAGE SIZE: 20KBLOGICAL DATA SIZE: 5.6KBTOTAL DOCUMENTS: 29INDEXES TOTAL SIZE: 20KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

Generate queries from natural language in Compass

FilterType a query: { field: 'value' }

store_location: "New York"

Temperature_celsius: 0.34000000000000003183

Humidity_perc: 93

Weather_Description: "snow"

_id: ObjectId('67a86c1be2908cf5312370ea')

date: "2/10/2025"

product_id: "P002"

sales_amount: 300

store_location: "Los Angeles"

Temperature_celsius: 10.9300000000000007

Humidity_perc: 84

Weather_Description: "clear sky"

_id: ObjectId('67a86c1be2908cf5312370eb')

date: "2/10/2025"

product_id: "P003"

sales_amount: 450

store_location: "Chicago"

Temperature_celsius: -2.26999999999999982

Humidity_perc: 88

Weather_Description: "overcast clouds"

_id: ObjectId('67a86c1be2908cf5312370ec')

date: "2/10/2025"

product_id: "P004"

sales_amount: 600

store_location: "Houston"

Temperature_celsius: 22.240000000000001

Humidity_perc: 85

Weather_Description: "broken clouds"

PREVIOUS

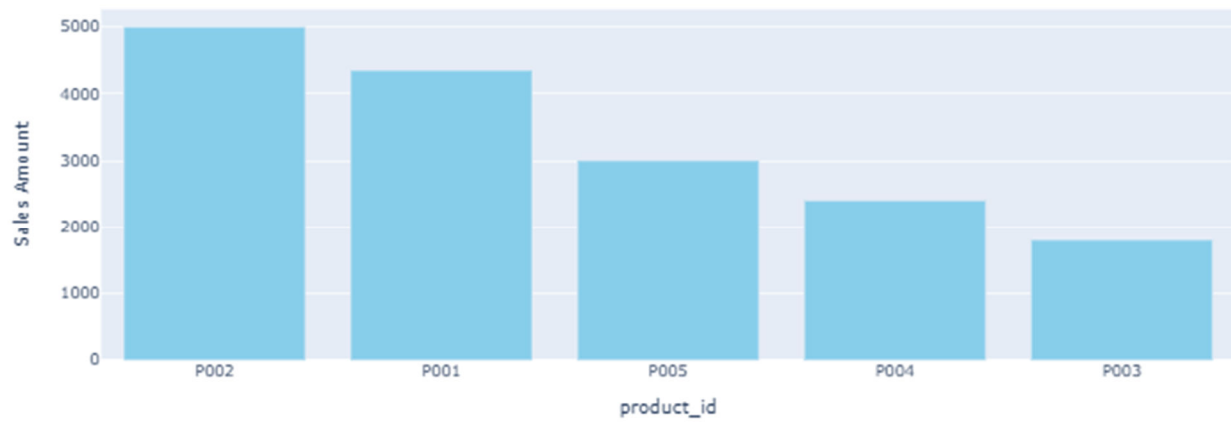
1-20 of many results

Sales Weather Dashboard

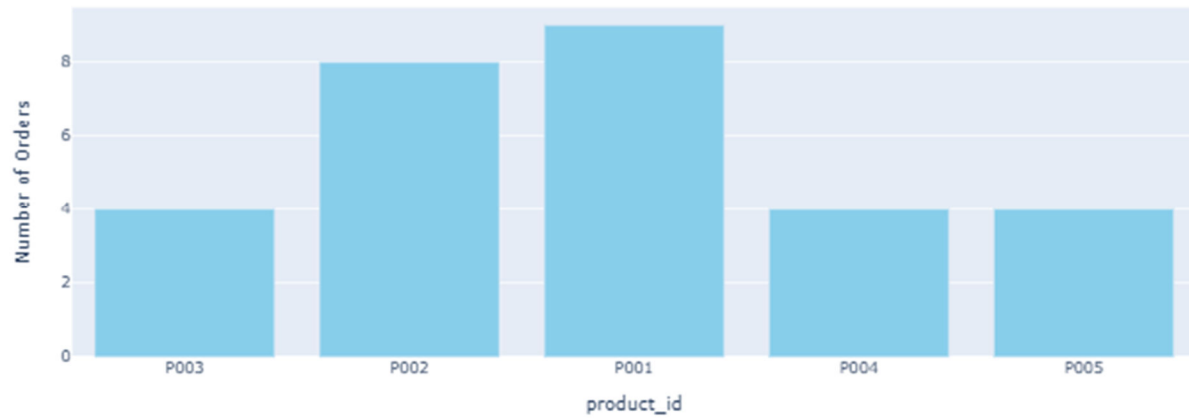
Sales

☐ Store Location ☒ Product

Sales Amount by product_id



Sales Amount by product_id



Sales

● Store Location ○ Product

Sales Amount by store_location



Sales Amount by store_location

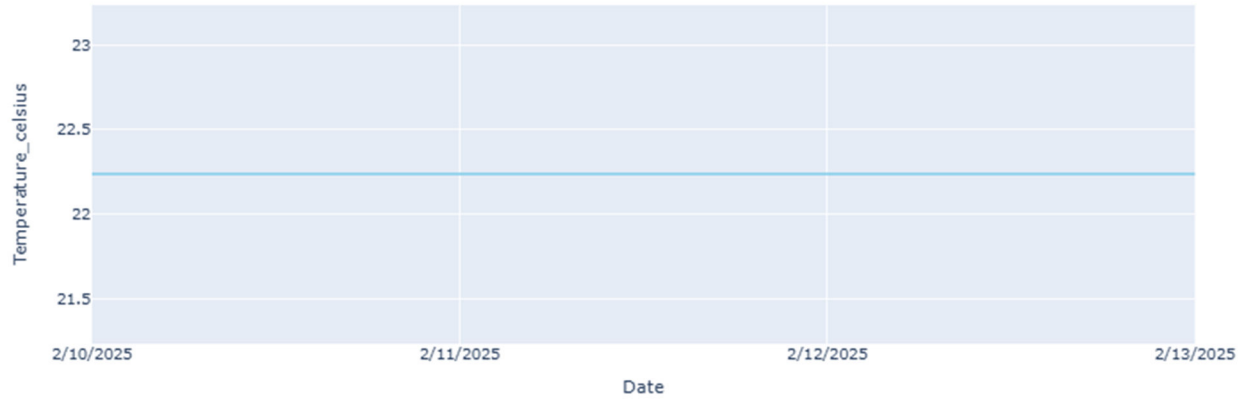


Weather

Houston



Temperature Trend by Date



Legend: Temperature_celsius, Humidity_perc

Temperature & Humidity Values

