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.

1. Data Integration & MongoDB Operations (Week 2)

* Store sales data in MongoDB and perform CRUD operations.
* Integrate MongoDB data and enhance Dash visualizations.

1. 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 **project\_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 project\_etl\_pipeline\_mongodb.py file.**

# Dataset Used

I used the latest dataset shared and has the below format

A screenshot of a white table with black text

Description automatically generated

# 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: H**andles 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.