10 Academy: Artificial Intelligence Mastery

Week 7 Challenge Document

Date: 09 July - 15 July 2024

# Shipping a Data Product: From Raw Telegram Data to an Analytical API

#### An end-to-end data pipeline for Telegram, leveraging dbt for transformation, Dagster for orchestration, and YOLOv8 for data enrichment.

# Overview

## Business Need

You are a Data Engineer at Kara Solutions, a leading data science company. Your team is tasked with building a robust data platform to generate insights about Ethiopian medical businesses, using data scraped from public Telegram channels.

A well-designed data platform significantly enhances data analysis. To achieve this, you will build an end-to-end pipeline that answers key business questions, such as:

* What are the top 10 most frequently mentioned medical products or drugs across all channels?
* How does the price or availability of a specific product vary across different channels?
* Which channels have the most visual content (e.g., images of pills vs. creams)?
* What are the daily and weekly trends in posting volume for health-related topics?

To answer these questions, you will implement a modern ELT (Extract, Load, Transform) framework. Raw data will be extracted from Telegram and loaded into a "Data Lake" storage zone. From there, it will be loaded into a PostgreSQL database, which will serve as your data warehouse. The crucial transformation step will happen inside the warehouse using dbt, where you will clean the data and remodel it into a dimensional star schema optimized for analytical queries. This layered approach ensures your data is reliable, scalable, and ready for analysis.

This project involves scraping, data modeling, object detection with YOLO to enrich the data, and exposing the final insights through an analytical API.

Your job is to build a data product that does the following:

* Develop a reproducible project environment and secure pipeline.
* Develop a data scraping and collection pipeline to populate a raw data lake.
* Design and implement a dimensional data model (star schema) in a PostgreSQL data warehouse.
* Develop a data cleaning and transformation pipeline using dbt.
* Enrich the data using object detection on images with YOLO.
* Expose the final, cleaned data through an analytical API using FastAPI.

## 

## Data and Features

You will be building a data warehouse for this week.

## Learning Outcomes

Skills:

* Telegram API data extraction using Telethon
* Data Modeling: Designing and implementing a Star Schema.
* ELT Pipeline Development: Building layered data pipelines (Raw -> Staging -> Marts).
* Infrastructure as Code (IaC) and environment management using Docker and requirements.txt.
* Data Transformation at scale using dbt (Data Build Tool).
* Data Enrichment using Object Detection (YOLO).
* Analytical API Development with FastAPI.
* Data Pipeline Orchestration with Dagster.
* Testing and validation of data systems.
* Managing credentials and secrets using environment variables.

Knowledge:

* Principles of modern ELT vs. ETL architectures.
* Layered data architecture (Data Lake, Staging, Data Marts).
* Best practices in data cleaning, validation, and transformation.
* Structuring data for efficient analytical queries (Dimensional Modeling).
* Integrating unstructured data (like image detection results) into a structured warehouse.
* Best practices for deploying and maintaining reproducible data pipelines.

Communication:

* Documenting data architecture and modeling decisions.
* Reporting on project outcomes and technical challenges

## Team

Tutors:

* Mahlet
* Rediet
* Kerod
* Rehmet

## Key Dates

* Discussion on the case - 09:00 UTC on Wednesday 09 July 2025. Use #all-week7 to pre-ask questions.
* Interim Solution - 20:00 UTC on Sunday 12 July 2025.
* Final Submission - 20:00 UTC on Tuesday 15 July 2025

# 

# Deliverables

## Task 0 - Project Setup & Environment Management

Before writing any code, set up a professional and reproducible project structure.

* Initialize a Git repository for your project.
* Create a requirements.txt file to manage all your Python dependencies.
* Create a Dockerfile and a docker-compose.yml to containerize your entire application (Python environment, PostgreSQL database). This guarantees that your project runs anywhere.
* Create a .env file to store all your secrets (Telegram API keys, database passwords).
* Add .env to your .gitignore file to ensure you never commit secrets to version control.
* Use a library like python-dotenv to load these secrets into your application as environment variables.

## Task 1 - Data Scraping and Collection (Extract & Load)

1. **Telegram Scraping:** Utilize the Telegram API or write custom scripts to extract data from public Telegram channels relevant to Ethiopian medical businesses. Use the following channels
   * [Chemed Telegram Channel](https://t.me/CheMed123)
   * <https://t.me/lobelia4cosmetics>
   * <https://t.me/tikvahpharma>
   * And many more from <https://et.tgstat.com/medicine>
2. **Image and Scraping:** Collect images from the following telegram channels for object detection:
   * [Chemed Telegram Channel](https://t.me/CheMed123)
   * <https://t.me/lobelia4cosmetics>
3. **Populate the Data Lake:** Store the raw, unaltered scraped data. This data is your source of truth.
   * **Structure:** Organize your raw data in a partitioned directory structure. For example: data/raw/telegram\_messages/YYYY-MM-DD/channel\_name.json. This makes incremental processing much easier.
   * **Format:** Store the raw data as JSON files, preserving the original structure from the API.
4. **Logging:** Implement robust logging to track which channels and dates have been scraped, and to capture any errors (e.g., rate limiting).

## Task 2 - Data Modeling and Transformation (Transform)

Raw data is messy and untrustworthy. [DBT](https://www.getdbt.com/) helps us build a reliable "data factory" to transform this raw material into a clean, trusted data product. It allows us to:

* Transform chaos into clean, structured tables.
* Model the data into a standard star schema, creating a single source of truth for analytics.
* Test our data to prevent bad data from reaching our users, building trust.
* Modularize our logic, making the system scalable and easy to maintain.

Use dbt to transform the raw data and build a structured data warehouse optimized for analysis.

* Write a script to load the raw JSON files from your data lake into a raw schema in your PostgreSQL database.
* Install dbt and its PostgreSQL adapter and set up a DBT project.
  + pip install dbt
* Initialize a [DBT](https://www.getdbt.com/) project connected to your PostgreSQL database.
  + dbt init my\_project
* Develop dbt Models in Layers**:**
  + **Staging Models (** Create models that clean and lightly restructure the raw data. These should perform tasks like casting data types, renaming columns, and extracting key fields from JSON. You should have one staging model per raw source (e.g., stg\_telegram\_messages.sql).
  + **Data Mart Models (** Build your final analytical tables by joining your staging models. Implement a **star schema** with fact and dimension tables.
    - dim\_channels: A dimension table with information about each Telegram channel.
    - dim\_dates: A dimension table for time-based analysis.
    - fct\_messages: A fact table containing one row per message, with foreign keys to your dimension tables and key metrics (e.g., message length, has\_image).
* **Testing and Documentation:**
  + Use dbt's built-in tests (unique, not\_null) to validate your primary keys and critical columns.
  + Write at least one **custom data test** (a SQL query that must return 0 rows to pass) to enforce a key business rule.
  + Use dbt docs generate to document your project.

## Task 3 - Data Enrichment with Object Detection (YOLO)

Use a modern, pre-trained YOLOv8 model to analyze images and integrate the findings into your data warehouse.

* Setting Up the Environment:
  + ​​Install the ultralytics package.
    - pip install ultralytics
* Write a script that scans for new images scraped in Task 1.
* Use the YOLOv8 model to detect objects in the images.
* Create a new fact table in dbt (e.g., fct\_image\_detections) with columns like message\_id (a foreign key to fct\_messages), detected\_object\_class, and confidence\_score. This links the visual content directly to your core data model.

## Task 4 - Build an Analytical API [Fast API](https://fastapi.tiangolo.com/)

* Setting Up the Environment:
  + Install FastAPI and Uvicorn

pip install fastapi uvicorn

* Create a FastAPI Application
  + Set up a basic project structure for your FastAPI application.

my\_project/

├── main.py

├── database.py

├── models.py

├── schemas.py

└── crud.py

* **Develop Analytical Endpoints:** Instead of generic CRUD, create endpoints that answer the business questions from the overview. The API should query your final dbt models (your data marts).
  + GET /api/reports/top-products?limit=10: Returns the most frequently mentioned products.
  + GET /api/channels/{channel\_name}/activity: Returns the posting activity for a specific channel.
  + GET /api/search/messages?query=paracetamol: Searches for messages containing a specific keyword.
* **Data Validation:** Use Pydantic schemas to define the structure of your API responses.

## Task 5 - Pipeline Orchestration

A collection of scripts is not a production pipeline. To make your workflow robust, observable, and schedulable, you will use an orchestrator. Use **Dagster** for its excellent local development experience.

1. **Install Dagster:**pip install dagster dagster-webserver
2. **Define a Job:** Convert your run\_pipeline.sh logic into a Dagster "job". A job is a graph of Python functions (called "ops"). You will have ops for:
   * scrape\_telegram\_data
   * load\_raw\_to\_postgres
   * run\_dbt\_transformations
   * run\_yolo\_enrichment
3. **Launch the UI:** Use the command dagster dev to launch the local UI, where you can inspect, run, and monitor your pipeline.
4. **Add a Schedule:** Use Dagster's scheduling features to configure your job to run automatically (e.g., every day).

# Tutorials Schedule

## Overview

In the following, the colour **purple** indicates morning sessions, and **blue** indicates afternoon sessions.

## Wednesday:

* Introduction to the challenge (Mahlet)
* Telegram scraping and Local database postgres (Rehmet).

## Thursday:

* End-to-End Data Transformation: Data Warehouse Modeling (Star Schema) and dbt (Kerod)
* Data Enrichment with Modern Computer Vision (Rediet)

## Friday:

* Building Analytical APIs with FastAPI and Data Pipeline Orchestration with dagster (Rediet)

# 

# Interim Submission

* Interim report - Covering Tasks 0, 1, and 2s
* A complete GitHub repository link covering **Task 0, 1, and 2**.
* Your repo should show a working setup, a data lake with raw data, and a dbt project with staging and mart models with passing tests.

## Feedback

You may not receive detailed comments on your interim submission but will receive a grade.

# Final Submission

* + A blog post entry (which you can submit for example to Medium publishing) or a pdf report.
    - A visual diagram of your entire data pipeline.
    - A diagram of your star schema.
    - Explanation of your technical choices for each task.
    - A brief reflection on difficulties and key takeaways.
    - Screenshots of your API endpoints working.
* Link to your Github code, and make sure to include screenshots demonstrating anything else you have done.

## Feedback

You will receive comments/feedback in addition to a grade.

# 

# References

**Web Scraping**

1. <https://realpython.com/python-web-scraping-practical-introduction/>
2. <https://realpython.com/beautiful-soup-web-scraper-python/>
3. <https://www.geeksforgeeks.org/python-web-scraping-tutorial/>
4. <https://scrapy.org/>
5. <https://www.selenium.dev/>
6. <https://docs.telethon.dev/en/stable/>

**DBT**

1. <https://www.getdbt.com/>
2. <https://docs.getdbt.com/docs/introduction>
3. <https://www.youtube.com/watch?v=C6BNAfaeqXY>
4. https://www.startdataengineering.com/post/dbt-data-build-tool-tutorial/

**YOLO**

1. Ultralytics YOLOv8 Docs: <https://docs.ultralytics.com/>
2. https://docs.ultralytics.com/tasks/detect/

**Data Modeling & Architecture:**

* The Data Warehouse Toolkit (Kimball): <https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/>
* dbt Docs on Structuring Projects: <https://docs.getdbt.com/guides/best-practices/how-we-structure/1-guide-overview>
* <https://www.projectpro.io/article/how-to-build-an-etl-pipeline-in-python/1131>
* <https://airbyte.com/tutorials/weather-data-stack-with-dbt-dagster-and-bigquery>

**Fast API**

1. <https://fastapi.tiangolo.com/>
2. <https://fastapi.tiangolo.com/tutorial/first-steps/>
3. <https://realpython.com/fastapi-python-web-apis/>
4. [Pydantic and Fast API](https://www.sicara.fr/blog-technique/fastapi-pydantic-powerful-duo#:~:text=FastAPI%20is%20the%20backend%20python,and%20even%20documenting%20your%20API.)
5. <https://medium.com/codenx/fastapi-pydantic-d809e046007f>

**Orchestration & Environment:**

* Dagster (Open-Source): <https://dagster.io/>
* Mage (Open-Source): <https://www.mage.ai/>
* Docker for Python Developers: <https://docs.docker.com/language/python/>