📦 Kara Solutions Data Pipeline Final Report

Project: Shipping a Data Product
Team: Yamlak Negash (Data Engineer)
Submission Date: July 15, 2025

---

1. 🎯 Business Understanding

Kara Solutions is building a unified analytical platform using public Telegram data from Ethiopian medical businesses. The aim is to:

- Monitor product trends and availability.
- Detect the most common medical items using object detection.
- Track channel activity, pricing variation, and visual content frequency.
- Serve insights through a programmatic API for business users.

This end-to-end ELT pipeline transforms messy Telegram data into an analytical warehouse with enriched metadata and images. The solution is containerized, reproducible, and fully orchestrated.

---

2. ⚙️ Technical Architecture Overview

Components Used:

Task	Tool/Tech
Scraping Telegram Data	`Telethon`
Data Storage (Raw)	`JSON`, `data/raw/`
Data Warehouse	`PostgreSQL`
Data Transformation	`dbt`
Object Detection on Images	`YOLOv8` (`ultralytics`)

API Layer	`FastAPI`, `Uvicorn`
Orchestration	`Dagster`
Environment	`Docker`, `.env`, `python-dotenv`

Folder Structure:

pgsql
CopyEdit

```
kara-data-pipeline/
├── data/
│   └── raw/telegram/YYYY-MM-DD/channel.json
├── dbt_project/
│   ├── models/
│   │   ├── staging/
│   │   ├── marts/
│   │   └── yolo/
│   └── dbt_project.yml
├── api/
│   ├── main.py
│   ├── schemas.py
│   └── database.py
├── dags/
│   └── pipeline_dagster.py
├── scripts/
│   ├── scrape_telegram.py
│   ├── load_to_postgres.py
│   ├── detect_objects.py
├── docker-compose.yml
├── Dockerfile
├── requirements.txt
└── README.md
```

---

3. 📊 Data Transformation with dbt

We created three model layers:

- **Staging Models** clean and flatten raw JSON (e.g., message text, post_date, image flag).
- **Mart Models** implement a **Star Schema** with:
    - `dim_channels`
    - `dim_dates`
    - `fct_messages`
- **YOLO Models** join `fct_messages` with detection results from images.

Custom dbt tests were implemented to ensure:

- No nulls in primary keys.
- Logical constraints (e.g., `message_length > 0` if `has_image = True`).

Sample query from API:

sql
CopyEdit

```
SELECT detected_object_class, COUNT(*)
FROM fct_image_detections
GROUP BY detected_object_class
ORDER BY COUNT(*) DESC
LIMIT 10;
```

---

4. 🧠 Image Enrichment via YOLOv8

- 500+ Telegram images processed.
- YOLOv8 detected common drug containers: pills, tubes, bottles.
- `ultralytics` Python API used for real-time inference.
- Detected classes and confidence scores logged.

- Detection results stored in `fct_image_detections`.

---

5. 🧪 API Endpoints (FastAPI)

Endpoint	Purpose
`/api/reports/top-products?limit=10`	Most common product mentions
`/api/channels/{channel}/activity`	Channel-specific post frequency
`/api/search/messages?query=amoxicillin`	Text search in posts

All endpoints use `Pydantic` schemas to validate responses.

FastAPI served with Uvicorn:

bash
CopyEdit
```
uvicorn main:app --reload
```

---

6. 🔄 Pipeline Orchestration (Dagster)

- Defined pipeline as a **Dagster Job** with 4 ops:
    - `scrape_telegram_data`
    - `load_raw_to_postgres`
    - `run_dbt_transformations`
    - `run_yolo_enrichment`
- Pipeline runs via `dagster dev` and is scheduled daily.

- Failure notifications printed via Dagster logging.

---

7. 🧰 Technical Stack

Category	Tool
Language	Python 3.10
Libraries	FastAPI, Telethon, dbt, Pandas, Dagster, YOLOv8
Database	PostgreSQL 14
Containerization	Docker, Docker Compose
Dependency Management	`requirements.txt`, `.env`
Scheduling	Dagster

---

8. ✅ Outcomes & Learnings

✅ Successes:

- Successfully modeled Telegram data into a star schema.
- Integrated YOLOv8 image object detection into analytics.
- Deployed secure, containerized API endpoints.
- Fully orchestrated ELT pipeline using Dagster.

🔍 Challenges:

- Handling rate limits from Telegram API.
- YOLOv8 occasionally misclassified overlapping items.
- Balancing speed and reliability across multiple data layers.

🧠 Learnings:

- dbt enforces good modeling hygiene and testability.
- Dagster provides superior observability vs. custom bash scripts.
- AI-powered enrichment creates value beyond basic data scraping.

---

9. 📸 Screenshots

Include here:

- Screenshot of working API response
- Dagster pipeline run dashboard
- YOLO object detection results

---

10. 📎 References

- dbt Docs
- YOLOv8 Ultralytics
- Dagster Quickstart
- FastAPI