OT7 - Foundation of Data Engineering - 2025/2026

Project DATA Engineering is provided by INSA Lyon.

Students: **Ahmed MANSOUR, Robert MICHEL, Stefan SEVERIN**

Abstract

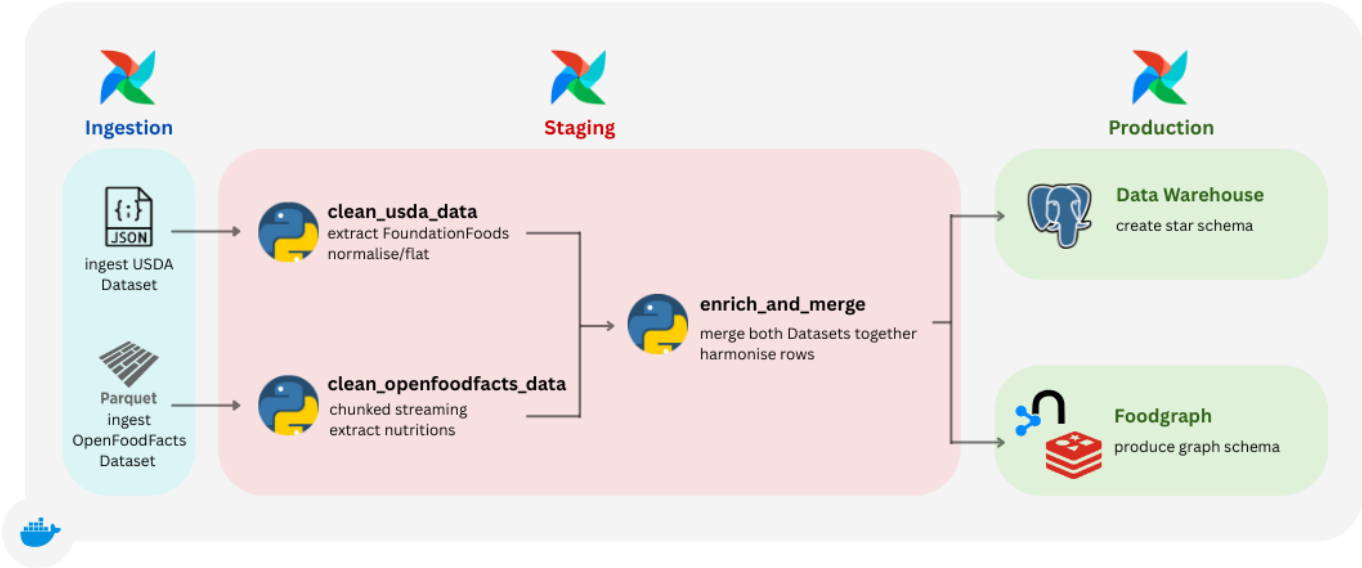
This README describes the ETL pipeline we implemented as part of the OT7 Data Engineering project (2025/2026). The pipeline ingests two main sources (OpenFoodFacts and USDA Foundation Foods), performs deterministic cleaning and unit normalization, produces a merged and enriched dataset, and exposes analytics-ready artifacts in a Postgres warehouse. We also provide an optional Neo4j-based graph for relationship exploration. The text below explains what we actually did, which functions implement each step, and why we made the choices we did.

Introduction

This project was built during the course to practise end-to-end data engineering: collecting heterogeneous public datasets, making them analysis-ready, and designing small analytical artifacts to answer domain-specific questions. We intentionally kept the stack simple—Airflow for orchestration, Pandas/pyarrow for transformation, Postgres for warehousing and Neo4j for graph exploration—so the code is readable and reproducible on a laptop with Docker.

The implementation is pragmatic: each core operation is written as a small Python function inside the Airflow DAGs so reviewers can run and inspect the steps easily. Where possible we document the exact function names used in the DAGs to make it simple to find the code that performs a transformation.

Pipeline



Datasets Description

For this projects we utilize 2 datasources:

- OpenFoodFacs Dataset
- U.S. Department of Agriculture (USDA) Dataset

Since the OpenFoodfacts dataset was a huge .parquet file (> 4 GB) it's not directly included in the repository.
Please download the file 'food.parquet' from the the following website: OpenFoodFacts

Storage and conventions used in the repo:

- Raw/unprocessed files (add the food.parquet here): `airflow/data/dataunclean/`
- Landing (ingested, immutable copies used by DAGs): `airflow/data/landing/`
- Staging (cleansed, typed snapshots): `airflow/data/staging/`

Ingestion

We implemented a minimal, transparent ingestion layer whose main job is to place immutable copies of raw files into the pipeline's landing area and record their origin. The ingestion DAG is defined in `airflow/dags/ingestion_dag.py`; the two most important functions / tasks are:

- `_ensure_directories(...)` — a small helper that creates required folders inside the container (called at DAG import time).
- Two BashOperator tasks: `ingest_json` and `ingest_parquet` which run a `cp` command to copy the source files from the host-mounted `data` folder into the container landing zone. The source and destination paths are defined at the top of `ingestion_dag.py` as `SRC_JSON`, `SRC_PARQUET`, and `DST_JSON`, `DST_PARQUET`.

What we did and why

We chose the simplest robust approach for ingestion for two reasons: (1) the raw files are large and often manually downloaded (especially OpenFoodFacts), and (2) keeping ingestion simple makes the pipeline

easy to reproduce and debug. The ingestion tasks intentionally avoid heavy parsing: they only copy files and leave cleaning and heavy work to the staging DAG.

Where to put data

- Place USDA JSON at `airflow/data/dataunclean/FoodData_Central_foundation_food_json_2025-04-24.json` (or update `SRC_JSON` in `ingestion_dag.py`).
- Place the OpenFoodFacts Parquet (commonly named `food.parquet`) in `airflow/data/dataunclean/`. The ingestion DAG copies it to `airflow/data/landing/`.

Notes on idempotence and ops

The ingestion tasks are written as simple file-copy operations. They are idempotent in the sense that re-running them replaces the landing files with identical content if the source has not changed. We log and track files via the landing folder structure; a future improvement would be a small ingestion metadata table (filename + checksum) recorded in Postgres; this is left as a low-risk follow-up.

Staging

Purpose

The staging layer is where raw heterogenous inputs are converted to a consistent, typed schema suitable for transformation and loading into the warehouse. Staging is focused on deterministic cleaning, basic validation and unit harmonization.

Key operations

- Column normalization: standardize column names and data types across sources.
- Unit conversion: convert all nutrient measurements to a canonical per-100g basis and consistent units (grams, milligrams, micrograms as appropriate).
- Deduplication: identify duplicate products using product identifiers when available.
- Outlier detection: flag or drop rows with impossible values (negative nutrients, extreme densities).
- Category vocabularies between sources will differ: manual mappings are maintained in the staging file

Implementation notes

- For moderate-sized inputs we use Pandas vectorized transformations. For working with the full OpenFoodFacts Parquet, run chunked reads (`chunk_size = 10000`) to avoid excessive memory usage.
- Staging outputs are written to `airflow/data/staging/` (Parquet) and loaded into Postgres staging tables under the `warehouse` schema for SQL-based transformations.

Production

Produce analytics-ready datasets: dimensional tables and a fact table (star schema) optimized for SQL analysis and dashboarding. In addition, materialize a small Neo4j graph for relationship queries where graphs provide value.

Design choices

- Use a star schema: a central `fact_food_nutrition` containing product_id, nutrient_id and value_per_100g, linked to `dim_product`, `dim_category`, `dim_nutrient` and `dim_source`.
- Store the analytical tables in Postgres (`warehouse` database) to leverage SQL for aggregation and joins.
- Use Neo4j for product-ingredient relationships and category co-occurrence graphs. Neo4j is optional and intended for exploratory analysis.

Data versioning and backup

The `backup_to_production()` function ensures data persistence and versioning by creating timestamped copies of the enriched dataset in the production zone. Each pipeline run generates a uniquely named backup file (e.g., `enriched_food_data_20260105_143022.parquet`) while maintaining a "latest" version for immediate access by downstream processes. This approach provides both historical data recovery capabilities and ensures the most recent processed data is always readily available.

Queries

1. How does the total nutrient content (protein, magnesium, vitamins) of a homemade dish compare to that of a premade product?

```
SELECT
    c.category_name,
    s.source_name,
    COUNT(*)                            AS food_count,
    AVG(f.energy_kcal)                  AS avg_energy_kcal,
    AVG(f.protein_g)                    AS avg_protein_g,
    AVG(f.carbohydrate_g)               AS avg_carbohydrate_g,
    AVG(f.fat_g)                        AS avg_fat_g,
    AVG(f.sugar_g)                      AS avg_sugar_g,
    AVG(f.magnesium_mg)                 AS avg_magnesium_mg,
    AVG(f.vitamin_a_mcg + f.vitamin_c_mg + f.vitamin_d_mcg +
f.vitamin_e_mg)
                                        AS avg_total_vitamins
FROM fact_food_nutrition f
JOIN dim_category c ON f.category_key = c.category_key
JOIN dim_source   s ON f.source_key   = s.source_key
WHERE s.source_name IN ('USDA','OpenFoodFacts')
GROUP BY c.category_name, s.source_name
ORDER BY c.category_name, s.source_name;
```

▶ Details

Query solution

category_name	source_name	food_count	avg_energy_kcal	avg_protein_g	avg_carbohydrate_g	avg_fat_g	avg_sugar_g	avg_magnesium_mg	avg_total_vitamins
baked products	OpenFoodFacts	439	234.956475833554	5.66901074400795	29.9702202073534	12.5352500438147	11.1680616011381	0	0.0348519366807336
baked products	USDA	3	1330	9.17333333333333	53.9666666666667	7.14666666666667	14.85	45.3	1.83666666666667
beef products	OpenFoodFacts	17239	186.986067212771	17.6485630969437	2.36106233582014	12.9736876316365	0.946993376661013	0.0734961425212525	0.350133825128732
beef products	USDA	15	189.6	21.76	0.0980666666666667	9.542	0	16.68	0.0306666666666667
beverages	OpenFoodFacts	13961	76.5772300588375	1.08874711263654	15.6746555542287	0.866380722015536	12.9418053348726	6.34004665850125	85972.432061143
beverages	USDA	3	0	0.669333333333333	2.036	1.84333333333333	0.773333333333333	6.98	1.10666666666667
cereal grains and pasta	USDA	41	257.268292682927	10.7973170731707	73.5170731707317	3.87048780487805	0.0253658536585366	106.175609756098	0
dairy and egg products	OpenFoodFacts	16238	264.939202974209	16.883504758745	3.50659733645356	21.8306202261567	2.1559308636957	0.0351054316476431	0.224479250044147
dairy and egg products	USDA	42	391.166666666667	16.0185714285714	4.02023809523809	20.4899285714286	1.38090476190476	19.3183333333333	0.236904761904762
fats and oils	OpenFoodFacts	1282	614.756440143085	0.9076457046199	12.5155947320439	65.419934391958	10.3161492782882	0.000156006236308561	0.276199611777292
fats and oils	USDA	9	92.5555555555556	0	0.0933333333333333	11.0111111111111	0	0	22.5344444444444
finfish and shellfish products	OpenFoodFacts	6471	151.594769478581	17.0720562438567	2.8987097526986	8.78933468106095	0.758782725279982	0.0903569774821935	0.627798842615369
finfish and shellfish products	USDA	10	91.9	17.6	0.0565	3.1907	0	24	0.094
fruits and fruit juices	USDA	48	44.3958333333333	0.73425	13.8954166666667	0.9909375	10.7047916666667	12.6897916666667	22.2595833333333
lamb, veal, and game products	USDA	2	0	18.7	-0.2005	13.74	0	18	0
legumes and legume products	OpenFoodFacts	1	150	7	31	2	9	0	0
legumes and legume products	USDA	37	98.6216216216216	19.4010810810811	14.9402702702703	4.99756756756757	0.0242432432432432	130.291891891892	0.197459459459459
nut and seed products	USDA	19	271.052631578947	17.4863157894737	26.3315789473684	49.4863157894737	0.384736842105263	245.621052631579	3.12368421052632
pork products	USDA	7	371	22.7142857142857	0.456142857142857	16.0471428571429	0.762857142857143	22.5857142857143	0
poultry products	USDA	11	161.181818181818	21.3363636363636	-0.139545454545455	7.46454545454546	0	22.5727272727273	0.0618181818181818
restaurant foods	OpenFoodFacts	2330	205.185111521447	10.1965957678183	24.9786117835234	9.45765938253937	2.99823942483751	0.0401502147380886	1.3706802580003
restaurant foods	USDA	4	348	6.4225	26.325	8.71	3.17	25.625	1.02
sausages and luncheon meats	OpenFoodFacts	309	192.920889104454	16.931836433781	1.72344748280388	13.8302096772734	0.730863729274823	0	0.00744336586095947
sausages and luncheon meats	USDA	6	959.166666666667	15.9833333333333	2.04	20.855	0.62	19.6	3.76333333333333
soups, sauces, and gravies	OpenFoodFacts	15248	167.552509376593	2.43542924259741	15.6109081398158	10.7335255031204	8.87664647548885	0.0735703047808623	5.56501143926698
soups, sauces, and gravies	USDA	2	107.5	1.425	7.395	0.835	4.655	16.85	0.61
spices and herbs	OpenFoodFacts	1093	206.764224710153	0.181596391378456	63.5621324335681	0.247598928384899	57.3047435553327	2.01006405939981	0.134766695736366
spices and herbs	USDA	2	30.5	2.125	2.65	1.69	0.71	23.85	0.375
sweets	OpenFoodFacts	60206	367.651316324228	6.64949694529848	51.2579669468368	16.8153264254822	28.3278038484129	3.22563249051563	3.92170790623422
sweets	USDA	1	385	0	99.6	0.32	99.8	0.3	0
unknown	OpenFoodFacts	106437	412.138665360735	10.6398631201055	47.6810229263219	20.6718321304164	22.8420303000484	1.98422997600337	4.19347695046643
vegetables and vegetable products	OpenFoodFacts	58746	220.336386994695	5.25223541231829	29.8312297687635	9.820111695492	9.35503800060729	3.39470916662157	8.25444111819281
vegetables and vegetable products	USDA	78	45.0769230769231	1.80267948717949	9.17858974358974	0.508576923076923	1.81185897435898	18.8761538461538	21.6402564102564

2. Which food categories show the largest fat content differences between raw and processed versions?

```
SELECT c.category_name,
       AVG(CASE WHEN s.source_name = 'USDA' THEN f.fat_g END) AS
avg_fat_usda,
       AVG(CASE WHEN s.source_name = 'OpenFoodFacts' THEN f.fat_g END) AS
avg_fat_off,
       (AVG(CASE WHEN s.source_name = 'OpenFoodFacts' THEN f.fat_g END) -
        AVG(CASE WHEN s.source_name = 'USDA' THEN f.fat_g END)) AS
fat_difference
FROM fact_food_nutrition f
JOIN dim_category c ON f.category_key = c.category_key
JOIN dim_source s ON f.source_key = s.source_key
WHERE f.fat_g IS NOT NULL
GROUP BY c.category_name
HAVING COUNT(CASE WHEN s.source_name = 'USDA' THEN 1 END) > 0
       AND COUNT(CASE WHEN s.source_name = 'OpenFoodFacts' THEN 1 END) > 0
       AND ABS(AVG(CASE WHEN s.source_name = 'OpenFoodFacts' THEN f.fat_g
END) -
               AVG(CASE WHEN s.source_name = 'USDA' THEN f.fat_g END)) > 0
ORDER BY ABS(AVG(CASE WHEN s.source_name = 'OpenFoodFacts' THEN f.fat_g
END) -
             AVG(CASE WHEN s.source_name = 'USDA' THEN f.fat_g END)) DESC
LIMIT 10;
```

▶ Details

Query solution

category_name	avg_fat_usda	avg_fat_off	fat_difference
fats and oils	11.0111111111111	65.419934391958	54.4088232808469
sweets	0.32	16.8153264254822	16.4953264254822
soups, sauces, and gravies	0.835	10.7335255031204	9.89852550312045
vegetables and vegetable products	0.508576923076923	9.820111695492	9.31153477241507
sausages and luncheon meats	20.855	13.8302096772734	-7.0247903227266
finfish and shellfish products	3.1907	8.78933468106095	5.59863468106095
baked products	7.14666666666667	12.5352500438147	5.38858337714804
beef products	9.542	12.9736876316365	3.43168763163651
legumes and legume products	4.99756756756757	2	-2.99756756756757
spices and herbs	1.69	0.247598928384899	-1.4424010716151

3.How does sodium (salt) content differ between homemade and commercial dishes of the same type?

```
SELECT
    c.category_name AS dish_type,
    s.source_name   AS dataset,
    COUNT(*)        AS items,
    AVG(f.sodium_mg) AS avg_sodium_mg,
    MIN(f.sodium_mg) AS min_sodium_mg,
    MAX(f.sodium_mg) AS max_sodium_mg
FROM fact_food_nutrition f
JOIN dim_category c ON f.category_key = c.category_key
JOIN dim_source   s ON f.source_key   = s.source_key
WHERE s.source_name IN ('USDA','OpenFoodFacts')
GROUP BY c.category_name, s.source_name
HAVING AVG(f.sodium_mg) > 0
ORDER BY c.category_name, s.source_name;
```

▶ Details

Query solution

dish_type	dataset	items	avg_sodium_mg	min_sodium_mg	max_sodium_mg
baked products	OpenFoodFacts	439	303.435116773869	0	2000
baked products	USDA	3	413.666666666667	314	477
beef products	OpenFoodFacts	17239	708.800069822045	0	51534.0003967285
beef products	USDA	15	49.9266666666667	42.5	67
beverages	OpenFoodFacts	13961	2751.394113948	0	37000000
beverages	USDA	3	53.6	42	59.6
cereal grains and pasta	USDA	41	1.86509756097561	0	20.3
dairy and egg products	OpenFoodFacts	16238	633.247522946649	0	714289.978027344
dairy and egg products	USDA	42	491.385714285714	0	1810
fats and oils	OpenFoodFacts	1282	375.007461759374	0	5619.99988555908
finfish and shellfish products	OpenFoodFacts	6471	600.52952346987	0	14399.9996185303
finfish and shellfish products	USDA	10	212.7	49.5	475
fruits and fruit juices	USDA	48	37.2986458333333	0	1620
lamb, veal, and game products	USDA	2	54.55	53.4	55.7
legumes and legume products	OpenFoodFacts	1	400.000005960464	400.000005960464	400.000005960464
legumes and legume products	USDA	37	67.7413513513513	0	438
nut and seed products	USDA	19	49.5710526315789	0	532
pork products	USDA	7	440.6	39.3	1830
poultry products	USDA	11	73.8272727272727	47	117
restaurant foods	OpenFoodFacts	2330	501.298747580564	0	2400.00009536743
restaurant foods	USDA	4	360.75	304	473
sausages and luncheon meats	OpenFoodFacts	309	832.61343157582	0	6400.00009536743
sausages and luncheon meats	USDA	6	854.333333333333	599	1040
soups, sauces, and gravies	OpenFoodFacts	15248	1112.72592319265	0	1300000
soups, sauces, and gravies	USDA	2	537.5	419	656
spices and herbs	OpenFoodFacts	1093	4703.748207278	0	42672.0008850098
spices and herbs	USDA	2	19900	1100	38700
sweets	OpenFoodFacts	60206	286.70543070211	0	439350.921630859
sweets	USDA	1	1	1	1
unknown	OpenFoodFacts	106437	720.241246899597	0	1990000
vegetables and vegetable products	OpenFoodFacts	58746	325.834961386547	0	1290000
vegetables and vegetable products	USDA	78	63.1759358974359	0	949

Foodgraph

Purpose and model

The food graph is a complementary Neo4j model to explore relationships that are cumbersome in relational schema, e.g., ingredient co-occurrence, brand-category-ingredient linkages, or similarity between products via shared ingredients.

Nodes and relationships (example):

- Product (product_id, name, brand)
- Ingredient (ingredient_id, name, normalized_name)
- Category (category_id, name)
- (Product)-[:CONTAINS]->(Ingredient)
- (Product)-[:BELONGS_TO]->(Category)

Notes

- The Neo4j instance is optional for main analytics; it is used for exploration and visualization.

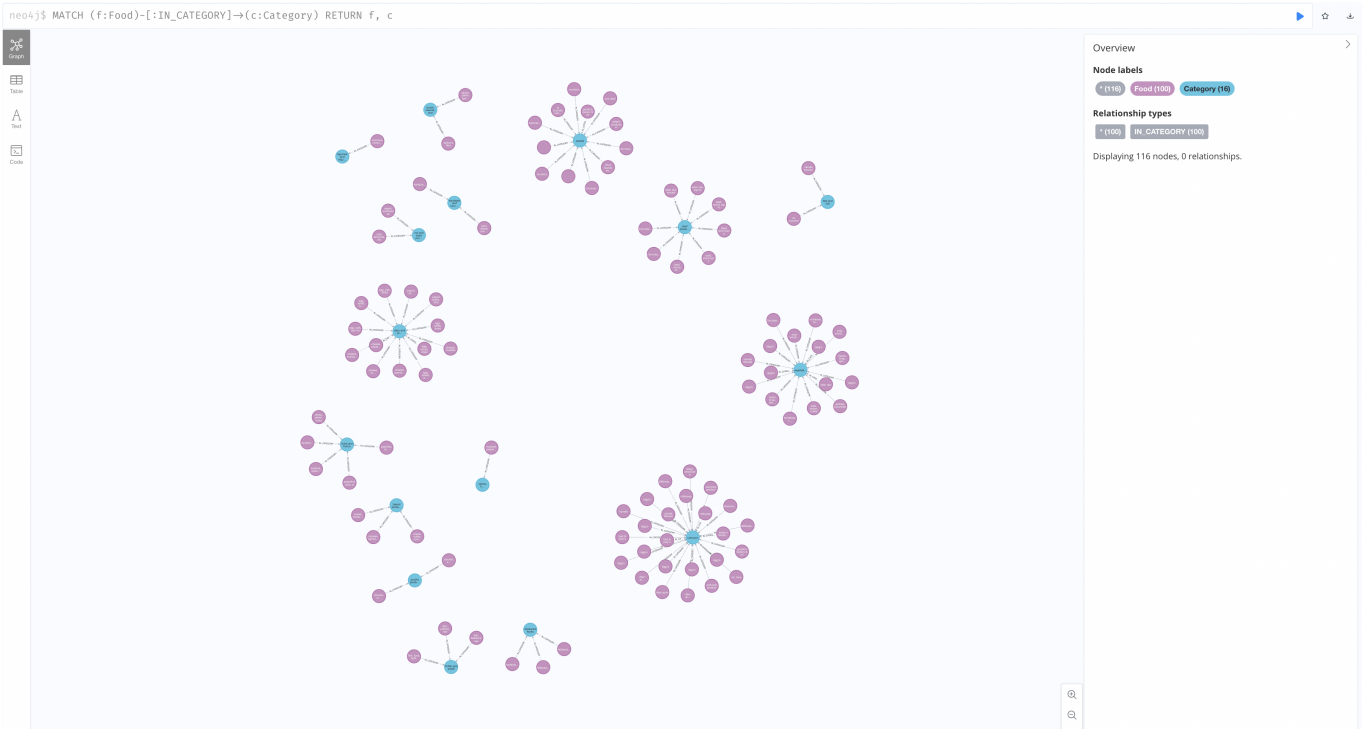
Queries

1. All Foods with Categories

```
MATCH (f:Food)-[:IN_CATEGORY]->(c:Category)
RETURN f, c
```

▶ Details

Query solution

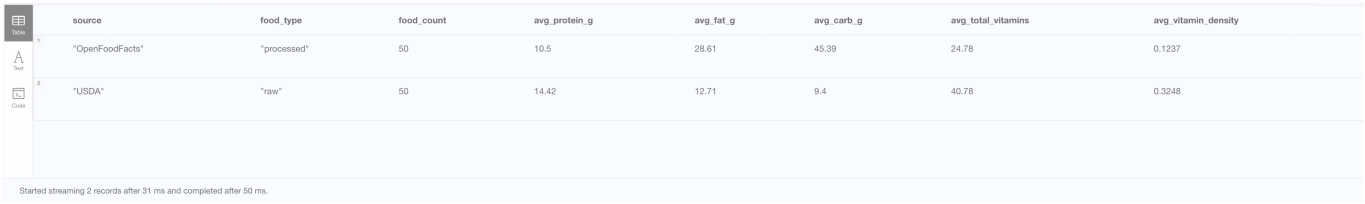


2. Nutrient Comparison (Homemade vs Premade):
   How does nutrient content compare between raw (USDA) and processed (OpenFoodFacts) foods?

```
MATCH (f:Food)-[:FROM_SOURCE]->(s:Source)
               WHERE f.energy_kcal IS NOT NULL
               RETURN
                   s.name as source,
                   f.food_type as food_type,
                   count(f) as food_count,
                   round(avg(f.protein_g), 2) as avg_protein_g,
                   round(avg(f.fat_g), 2) as avg_fat_g,
                   round(avg(f.carbohydrate_g), 2) as avg_carb_g,
                   round(avg(f.total_vitamins), 2) as avg_total_vitamins,
                   round(avg(f.vitamin_density), 4) as
avg_vitamin_density
               ORDER BY source, food_type
```

▶ Details

Query solution



	source	food_type	food_count	avg_protein_g	avg_fat_g	avg_carb_g	avg_total_vitamins	avg_vitamin_density
1	"OpenFoodFacts"	"processed"	50	10.5	28.61	45.39	24.78	0.1237
2	"USDA"	"raw"	50	14.42	12.71	9.4	40.78	0.3248

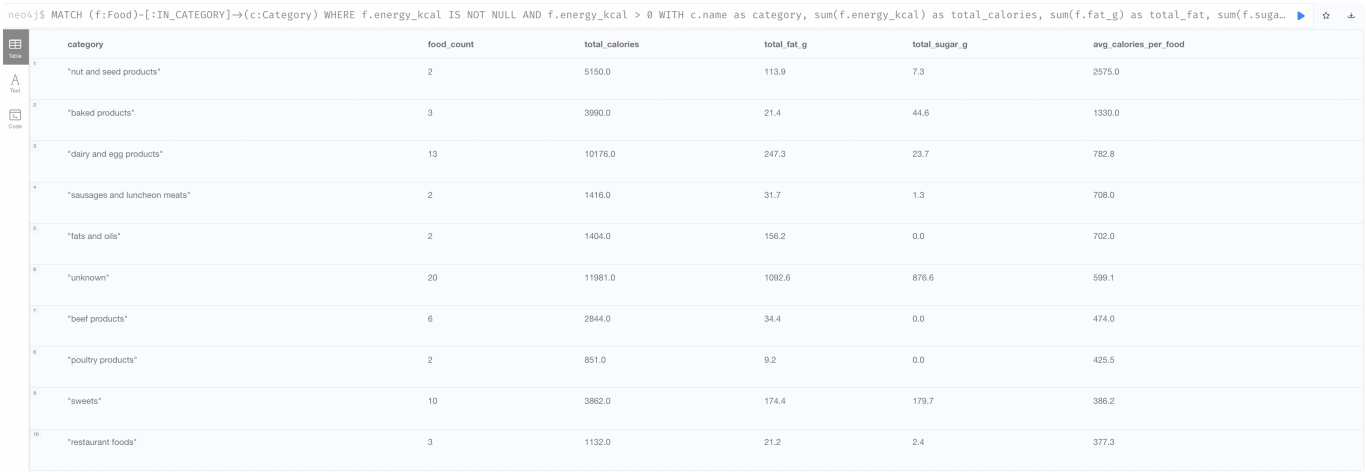
Started streaming 2 records after 31 ms and completed after 50 ms.

3. Top Contributors to Calories, Fat, Sugar:
   Which categories contribute most to calories, fat, and sugar?

```
MATCH (f:Food)-[:IN_CATEGORY]->(c:Category)
                WHERE f.energy_kcal IS NOT NULL AND f.energy_kcal > 0
                WITH c.name as category,
                     sum(f.energy_kcal) as total_calories,
                     sum(f.fat_g) as total_fat,
                     sum(f.sugar_g) as total_sugar,
                     count(f) as food_count
                RETURN category, food_count,
                       round(total_calories, 0) as total_calories,
                       round(total_fat, 1) as total_fat_g,
                       round(total_sugar, 1) as total_sugar_g,
                       round(total_calories / food_count, 1) as
avg_calories_per_food
                ORDER BY avg_calories_per_food DESC
                LIMIT 10
```

▶ Details

Query solution



neo4j$ MATCH (f:Food)-[:IN_CATEGORY]→(c:Category) WHERE f.energy_kcal IS NOT NULL AND f.energy_kcal > 0 WITH c.name as category, sum(f.energy_kcal) as total_calories, sum(f.fat_g) as total_fat, sum(f.suga…

	category	food_count	total_calories	total_fat_g	total_sugar_g	avg_calories_per_food
1	"nut and seed products"	2	5150.0	113.9	7.3	2575.0
2	"baked products"	3	3990.0	21.4	44.6	1330.0
3	"dairy and egg products"	13	10176.0	247.3	23.7	782.8
4	"sausages and luncheon meats"	2	1416.0	31.7	1.3	708.0
5	"fats and oils"	2	1404.0	156.2	0.0	702.0
6	"unknown"	20	11981.0	1092.6	876.6	599.1
7	"beef products"	6	2844.0	34.4	0.0	474.0
8	"poultry products"	2	851.0	9.2	0.0	425.5
9	"sweets"	10	3862.0	174.4	179.7	386.2
10	"restaurant foods"	3	1132.0	21.2	2.4	377.3

Requirements

- Docker Desktop (Mac/Windows) or Docker Engine (Linux).
- Docker Compose v2+.
- Optional for local scripts: Python 3.10+, Pandas, PyArrow, Dask (for large parquet processing), and psycopg2-binary for Postgres access.

How to run?

1. Ensure Docker is installed and running on your machine.
2. Clone the repository and change into the `airflow/` directory:

```
git clone <repo-url>
cd Wizards-of-Oz/airflow
```

3. Start the services (build images if needed):

```
docker compose up --build
```

4. Visit the services in your browser:

Service	URL
Adminer	http://localhost:8081/
Airflow	http://localhost:8082/
Neo4j	http://localhost:7474/browser/

5. Credentials and sensitive configuration are supplied via the `.env` file used by docker compose. If you run into connection issues, verify the `.env` values and the container logs.

Notes and first-time setup

- In Adminer the server name is `postgres` (not `db`).
- If you need to reinitialize databases (first-time run after changes to `airflow/db/init`), run:

```
docker compose down -v
docker compose up --build
```

- Place the OpenFoodFacts `food.parquet` into `airflow/data/dataunclean/` before triggering the ingestion DAG if you intend to run the full dataset ingestion.

Limitations & Challenges

During the development of this data pipeline, we encountered several significant challenges. The OpenFoodFacts food.parquet file was extremely large, requiring us to implement chunked reading to avoid memory issues and ensure processing could complete within reasonable time limits. Additionally, we spent considerable time debugging data quality issues, particularly discovering that the USDA JSON file was being parsed incorrectly, resulting in numerous NaN values appearing throughout the dataset. We also had to account for variations in nutrient naming conventions across different data sources, implementing flexible matching logic to handle these inconsistencies and ensure accurate nutrient extraction.
Another important limitation concerns dataset imbalance. Due to the significantly larger size of OpenFoodFacts compared to the other data sources, some aggregated comparisons may be biased toward

this dataset. This imbalance should be taken into account when interpreting the results of comparative analytics.

Note for Students

- ☑ Clone the created repository offline;
- ☑ Add your name and surname into the Readme file and your teammates as collaborators
- ☑ Complete the field above after project is approved
- ☑ Make any changes to your repository according to the specific assignment;
- ☑ Ensure code reproducibility and instructions on how to replicate the results;
- ☑ Add an open-source license, e.g., Apache 2.0;
- ☑ README is automatically converted into pdf