# **README — Eurostat Datasets**

# **Purpose**

This project converts **Eurostat survey Excel files** into **relational database–ready tables**.  
It extracts, cleans, and normalizes the data to **Third Normal Form (3NF)** for use in EUROSTAT\_load\_db or other analytical databases.

The process:

* Reads all relevant Eurostat sheets automatically.
* Extracts dataset metadata (title, date, frequency, indicators).
* Transforms multi-header layouts into tidy country × year × value format.
* Cleans missing values and enforces nullable data types.
* Outputs three 3NF CSVs (countries, indicators, observations) for SQL import.

## **Input Files**

| **File** | **Description** |
| --- | --- |
| isoc\_r\_eb\_ain2$defaultview\_spreadsheet.xlsx | AI technologies in enterprises by NUTS 2 region |
| isoc\_eb\_ain2$defaultview\_spreadsheet.xlsx | AI technologies by size class and NACE Rev.2 activity |
| isoc\_eb\_ai$defaultview\_spreadsheet.xlsx | Enterprises using AI technologies (aggregated indicators) |

Each workbook includes:

* ≈9-row metadata block (dataset name, last updated, frequency).
* One or more valid data sheets with headers:  
   TIME | GEO (Labels) | 2021 | 2023 | 2024 …
* Non-data sheets (*Summary*, *Flags*, etc.) are ignored.

## **Processing Script — data\_wrangling\_eurostat.py**

### **Requirements**

Python ≥ 3.9

* pip install pandas openpyxl

### **Steps**

1. **Detect** sheets containing both TIME and GEO (Labels).
2. **Extract** metadata: dataset title, last updated, size class, NACE Rev.2, indicator, unit.
3. **Reshape** wide data → long (country\_name, year, value).
4. **Clean** symbols like :, NaN, <N.A> → pd.NA.
5. **Combine** all sheets → one file per workbook.
6. **Normalize** merged dataset into 3NF:  
   * countries.csv — dimension
   * indicators.csv — dimension
   * observations.csv — fact table

## **Output Files**

| **File** | **Description** |
| --- | --- |
| countries.csv | Unique list of countries |
| indicators.csv | Unique indicator + metadata combinations |
| observations.csv | Fact table (country × indicator × year × value) |

All files use **UTF-8**, have headers, and contain no index columns.

## **Database Schema (3NF)**

CREATE TABLE countries (

country\_id SERIAL PRIMARY KEY,

country\_name TEXT UNIQUE

);

CREATE TABLE indicators (

indicator\_id SERIAL PRIMARY KEY,

information\_society\_indicator TEXT,

unit\_of\_measure TEXT,

nace\_rev2 TEXT,

size\_class TEXT,

time\_frequency TEXT

);

CREATE TABLE observations (

observation\_id SERIAL PRIMARY KEY,

country\_id INT REFERENCES countries(country\_id),

indicator\_id INT REFERENCES indicators(indicator\_id),

year INT,

value FLOAT,

dataset TEXT,

last\_updated TEXT,

data\_extracted\_on TEXT,

sheet\_name TEXT

);

**Relations:** countries (1) ───< observations >───(1) indicators

## **Loading Examples**

**PostgreSQL**

\copy countries FROM 'countries.csv' CSV HEADER;

\copy indicators FROM 'indicators.csv' CSV HEADER;

\copy observations FROM 'observations.csv' CSV HEADER;

**MySQL**

LOAD DATA INFILE '/path/to/countries.csv'

INTO TABLE countries

FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

**Foreign Keys**

ALTER TABLE observations

ADD CONSTRAINT fk\_country FOREIGN KEY (country\_id) REFERENCES countries(country\_id),

ADD CONSTRAINT fk\_indicator FOREIGN KEY (indicator\_id) REFERENCES indicators(indicator\_id);

**Validation Checks**

-- Missing country references

SELECT COUNT(\*) FROM observations o

LEFT JOIN countries c ON o.country\_id = c.country\_id

WHERE c.country\_id IS NULL;

-- Count records by dataset

SELECT dataset, COUNT(\*) FROM observations GROUP BY dataset;

| **Check** | **Result** |
| --- | --- |
| Missing values standardized (pd.NA) | DONE |
| UTF-8 encoding | DONE |
| Nullable Int/Float types | DONE |
| Referential integrity (FKs) | DONE |
| Duplicates kept for audit traceability | DONE |

**Workflow Summary**

1. Place all Eurostat workbooks in the working directory.

Run:  
 python data\_wrangling\_eurostat.py

1. Verify row counts in console logs.
2. Import 3NF CSVs into your database.
3. Apply FKs and validate joins.

## **Developer Notes**

* PKs are auto-increment integers (country\_id, indicator\_id, observation\_id).
* FKs maintain referential integrity between dimension and fact tables.
* year provides temporal granularity.
* Data retains **original Eurostat values** — no aggregation or transformation.
* Ready for use alongside other Eurostat or sentiment data sources.