PROJECT TITLE : **CONSTRUCTION OF A DATASET ON ITALIAN MEDICINES**
COURSE NAME : **FUNDAMENTALS OF PROGRAMMING AND DATA MANAGEMENT MODULE B A.Y. 2024/2025**
STUDENT NAME : **MUHAMMAD HASAN [MATRICULA ID: D03000134]**

FINAL REPORT

Data Extraction

The main structured data is downloaded from AIFA website https://www.aifa.gov.it/liste-dei-farmaci as instructed in the project's requirements.

To extract additional information for each medicine, a Python script is written and executed on Google Collab environment.

The link of the code Git repository is https://github.com/thisishasan/prog_mod_b

The script uses following libraries:

1) **Pandas** - Used for loading and manipulating the CSV dataset.
2) **Requests** - Used to send HTTP GET requests to the **AIFA API** and download PDFs.
3) **fitz (from `PyMuPDF`)** - Used to read and extract text from downloaded PDF files. Handles PDF page access and text parsing.
4) **Re** - Python's regular expressions module. Used for cleaning and parsing text (e.g., normalizing whitespace, identifying section headers).
5) **Csv** - Used to specify quoting behavior in `to_csv()` function for properly formatted output.

The data is loaded into a Pandas DataFrame (df), keeping only relevant columns 'Principio Attivo', 'Descrizione Gruppo', 'Denominazione e Confezione', 'Titolare AIC', 'AIC', 'Codice Gruppo Equivalenza'

A specific range of rows is selected from the dataset: from index 470 to 802

```
import pandas as pd

df = pd.read_csv("data.csv")

df = df[['Principio Attivo', 'Descrizione Gruppo', 'Denominazione e Confezione', 'Titolare AIC', 'AIC', 'Codice Gruppo Equivalenza']]

df = df.iloc[472-2:804-2]
```

Performed data quality checks including missing values and duplicates records

```
df.isnull().sum()
```

	0
Principio Attivo	0
Descrizione Gruppo	0
Denominazione e Confezione	0
Titolare AIC	0
AIC	0
Codice Gruppo Equivalenza	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(0)
```

The core logic revolves around iterating through each row (drug) in the DataFrame

1) API Query
   - Uses the AIC code to query the AIFA API https://api.aifa.gov.it/aifa-bdf-eif-be/1.0.0/formadosaggio/ricerca?query={aic}&spellingCorrection=true&page=0 for drug formulation and dosage info.
2) Parses the JSON response to extract:
   - ATC code (Anatomical Therapeutic Chemical classification)
   - Drug description
   - Unique URLs for the online leaflet and downloadable PDF
   -

```
# Build the API URL
url = f"https://api.aifa.gov.it/aifa-bdf-eif-be/1.0.0/formadosaggio/ricerca?query={aic}&spellingCorrection=true&page=0"

# Make the GET request
response = requests.get(url)

# Check for successful response
if response.status_code == 200:
    json_data = response.json()  # Parse the response as JSON

    # Access the nested data field
    content = json_data.get('data', {}).get('content', [])
```

3) These are stored back into the DataFrame.

4) The PDF is downloaded from the AIFA API endpoint https://api.aifa.gov.it/aifa-bdf-eif-be/1.0.0/organizzazione/{codice_sis}/farmaci/{aic6}/stampati?ts=RCP

```
# Print each item in the content list
for item in content:
    id = item.get('id')
    codice_atc = item.get('codiceAtc')[0]
    df.at[index, 'ATC'] = codice_atc
    descrizione_atc = item.get('descrizioneAtc')
    codice_sis = item.get('medicinale').get('codiceSis')
    aic6 = item.get('medicinale').get('aic6')
    leaflet_url = f"https://medicinali.aifa.gov.it/it/#/it/organizzazione/{codice_sis}/farmaci/{aic6}/stampati/FI"
    pdf_url = f"https://api.aifa.gov.it/aifa-bdf-eif-be/1.0.0/organizzazione/{codice_sis}/farmaci/{aic6}/stampati?ts=RCP"
    df.at[index, 'URL'] = leaflet_url
```

5) Using fitz, the script:
   - Reads all pages
   - Combines the text into a single string

- Normalizes whitespace and broken lines

```
# creating a pdf reader object
#doc = PdfReader(pdf_path)
doc = fitz.open(pdf_path)

# Combine all page text into one string
full_text = ""
#for page in doc.pages:
for page in doc:
    #full_text += page.extract_text().strip()
    full_text += page.get_text()
doc.close()

# Normalize the text: fix words broken across lines
text = re.sub(r'([a-zA-Z])\n([a-zA-Z])', r'\1 \2', full_text)  # fix broken words
text = re.sub(r'\n+', '\n', text)
text = re.sub(r'\s{2,}', ' ', text)  # collapse long spaces
```

6) The text is parsed to extract specific regulatory sections based on the headings
   - A regex pattern is dynamically built to match section headers.
   - Text between matched sections is extracted.
   - Each section is stored in a new column of the DataFrame under its corresponding title.

```
# Create a regex pattern for section headers
section_regex = [
    fr"({num}[\.\s]*{re.escape(title)})"
    for num, title in sections
]
pattern = '|'.join(section_regex)
pattern = re.compile(pattern, flags=re.IGNORECASE)

# Find all section starts
matches = list(pattern.finditer(text))

# Extract section contents
extracted = {}
for i in range(len(matches)):
    start = matches[i].start()
    end = matches[i+1].start() if i+1 < len(matches) else len(text)

    header = matches[i].group().strip()
    content = text[start:end].replace(header, '', 1).strip()

    # Match section by header prefix (e.g. "4.2")
    matched_section = next((f"{num} {title}" for num, title in sections if num in header), header)
    extracted[matched_section] = content

# Save each section in a separate variable
for title, content in extracted.items():
    section_name = title.replace(" ", "_").replace(".", "")  # Create a valid variable name
    globals()[section_name] = content  # Dynamically assign the content to a variable
    df.at[index, title] = content.replace(title," ").strip()

    # Print the results
    print(f"\n{'='*80}\n{title}\n{'='*80}\n{content}...\n")
```

Any remaining missing values are replaced with "Not available".

```
[ ]  df = df.fillna("Not available")
```

The enriched dataset is saved as a new CSV file named updated_data.csv with UTF-8 encoding and all fields quoted.

```python
import csv
df.to_csv("updated_data.csv", index=False, encoding="utf-8", quoting=csv.QUOTE_ALL)
```

Relational Database Management System

1) Create a database named ‘medicine_data’

```
CREATE DATABASE IF NOT EXISTS medicine_data;
```

2) Create a table ‘medicines’ into the database

```
CREATE TABLE IF NOT EXISTS medicines (
    id INT AUTO_INCREMENT PRIMARY KEY,
    active_ingredient LONGTEXT,
    group_description LONGTEXT,
    medicine_name_and_packaging LONGTEXT,
    marketing_authorization_holder LONGTEXT,
    aic_code BIGINT,
    equivalence_group_code VARCHAR(20),
    atc VARCHAR(50),
    leaflet_url LONGTEXT,
    pdf_url LONGTEXT,
    therapeutic_indications LONGTEXT,
    posology_and_method_of_administration LONGTEXT,
    contraindications LONGTEXT,
    special_warnings_and_precautions_for_use LONGTEXT,
    interactions_with_other_medicinal_products LONGTEXT,
    fertility_pregnancy_and_lactation LONGTEXT,
    effects_on_ability_to_drive_and_use_machines LONGTEXT,
    undesirable_effects_side_effects LONGTEXT,
    overdose LONGTEXT,
    incompatibilities LONGTEXT
);
```

3) Import the updated CSV into the table ‘medicines’

```
LOAD DATA INFILE '/var/lib/mysql-files/updated_data.csv'
INTO TABLE medicines
CHARACTER SET utf8
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
```

```
(
  active_ingredient,
  group_description,
  medicine_name_and_packaging,
  marketing_authorization_holder,
  aic_code,
  equivalence_group_code,
  atc,
  leaflet_url,
  pdf_url,
  therapeutic_indications,
  posology_and_method_of_administration,
  contraindications,
  special_warnings_and_precautions_for_use,
  interactions_with_other_medicinal_products,
  fertility_pregnancy_and_lactation,
  effects_on_ability_to_drive_and_use_machines,
  undesirable_effects_side_effects,
  overdose,
  incompatibilities
);
```

4) Due to data redundancy, Create separate tables for columns (active_ingredient, atc, equivalence_group_code and marketing_authorization_holder)

```
CREATE TABLE IF NOT EXISTS active_ingredients (
    id INT AUTO_INCREMENT PRIMARY KEY,
    active_ingredient VARCHAR(50) UNIQUE
);

CREATE TABLE IF NOT EXISTS atc_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    atc_code VARCHAR(20) UNIQUE
);

CREATE TABLE IF NOT EXISTS equivalence_group_codes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    equivalence_group_code VARCHAR(5) UNIQUE
);

CREATE TABLE IF NOT EXISTS marketing_authorization_holders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    marketing_authorization_holder VARCHAR(50) UNIQUE
);
```

5) Insert records into new tables (active_ingredients, atc, equivalence_group_codes and marketing_authorization_holders) from medicines tables

```
INSERT INTO active_ingredients (active_ingredient)
SELECT distinct medicines.active_ingredient FROM medicines
ON DUPLICATE KEY UPDATE
active_ingredient = VALUES(active_ingredient);

INSERT INTO atc_codes (atc_code)
SELECT distinct medicines.atc FROM medicines
ON DUPLICATE KEY UPDATE
atc_code = VALUES(atc_code);

INSERT INTO equivalence_group_codes (equivalence_group_code)
SELECT distinct medicines.equivalence_group_code FROM medicines
ON DUPLICATE KEY UPDATE
equivalence_group_code = VALUES(equivalence_group_code);

INSERT INTO marketing_authorization_holders (marketing_authorization_holder)
SELECT distinct medicines.marketing_authorization_holder FROM medicines
ON DUPLICATE KEY UPDATE
marketing_authorization_holder = VALUES(marketing_authorization_holder);
```

6) Alter table ‘medicines’ and add new columns (active_ingredient_id, atc_code_id, equivalence_group_code_id, marketing_authorization_holder_id) into the table

```
ALTER TABLE medicines ADD COLUMN active_ingredient_id INT AFTER active_ingredient;
ALTER TABLE medicines ADD COLUMN atc_code_id INT AFTER atc;
ALTER TABLE medicines ADD COLUMN equivalence_group_code_id INT AFTER
equivalence_group_code;
ALTER TABLE medicines ADD COLUMN marketing_authorization_holder_id INT AFTER
marketing_authorization_holder;
```

7) Update table ‘medicines’ and set values for the new columns (active_ingredient_id, atc_code_id, equivalence_group_code_id, marketing_authorization_holder_id) from the corresponding tables (active_ingredients, atc_codes, equivalence_group_codes, marketing_authorization_holders)

```
UPDATE medicines m
JOIN active_ingredients ai ON m.active_ingredient = ai.active_ingredient
SET m.active_ingredient_id = ai.id;

UPDATE medicines m
JOIN atc_codes ai ON m.atc = ai.atc_code
SET m.atc_code_id = ai.id;

UPDATE medicines m
JOIN equivalence_group_codes ai ON m.equivalence_group_code =
ai.equivalence_group_code
SET m.equivalence_group_code_id = ai.id;

UPDATE medicines m
JOIN marketing_authorization_holders ai ON m.marketing_authorization_holder =
ai.marketing_authorization_holder
SET m.marketing_authorization_holder_id = ai.id;
```

8) Drop columns (active_ingredient, atc, equivalence_group_code, marketing_authorization_holder) from the table ‘medicines’

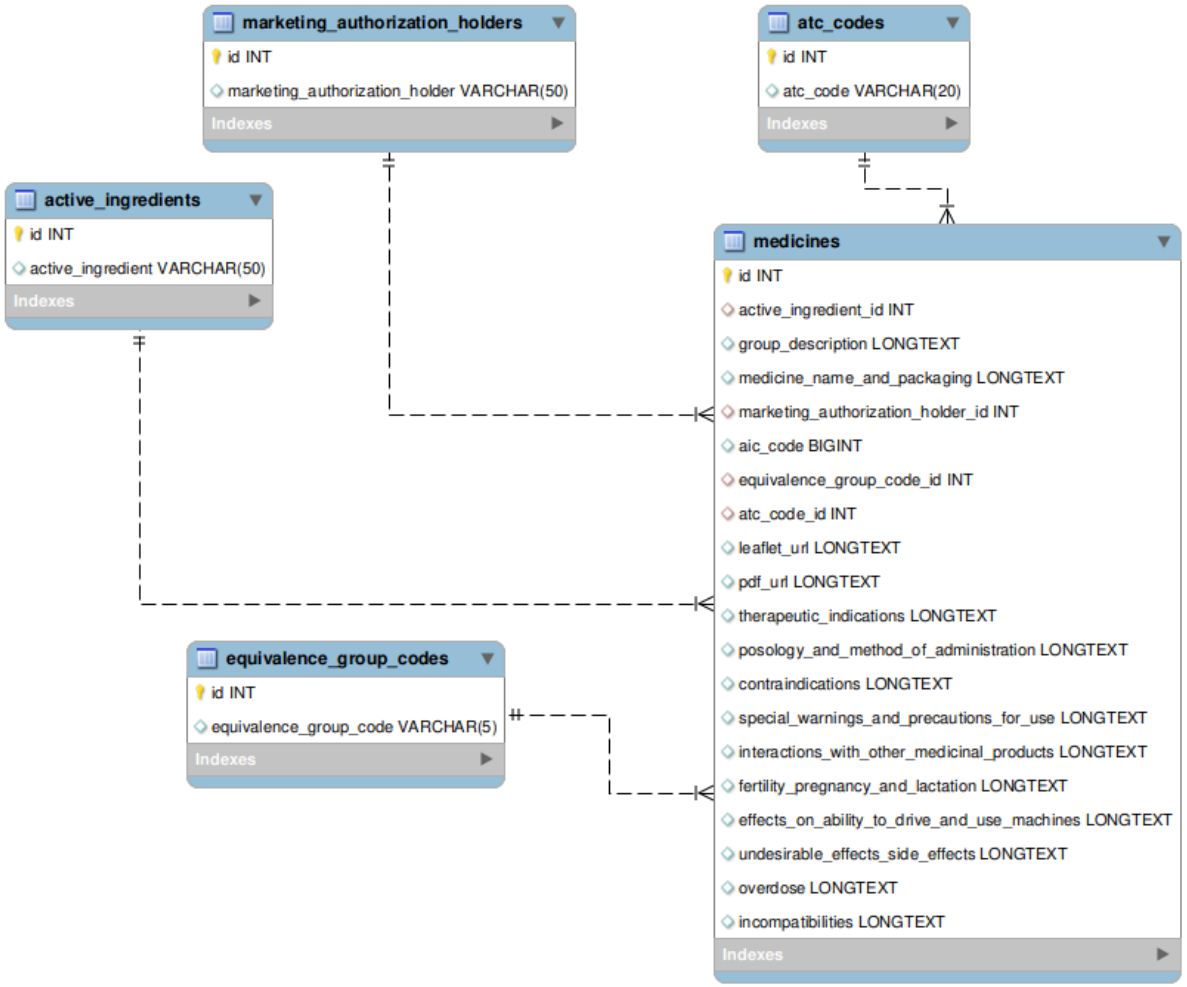
```
ALTER TABLE medicines
DROP COLUMN active_ingredient,
DROP COLUMN atc,
DROP COLUMN equivalence_group_code,
DROP COLUMN marketing_authorization_holder;
```

9) Alter table ‘medicines’ and apply foreign key constraints to the columns (active_ingredient_id, atc_code_id, equivalence_group_code_id, marketing_authorization_holder_id) with referenced tables created earlier (active_ingredients, atc_codes, equivalence_group_codes, marketing_authorization_holders)

```
ALTER TABLE medicines
ADD CONSTRAINT fk_active_ingredient
  FOREIGN KEY (active_ingredient_id) REFERENCES active_ingredients(id),
```

```
ADD CONSTRAINT fk_atc_code
  FOREIGN KEY (atc_code_id) REFERENCES atc_codes(id),
ADD CONSTRAINT fk_equivalence_group_code
  FOREIGN KEY (equivalence_group_code_id) REFERENCES
equivalence_group_codes(id),
ADD CONSTRAINT fk_marketing_authorization_holder
  FOREIGN KEY (marketing_authorization_holder_id) REFERENCES
marketing_authorization_holders(id);
```

10) ERD Diagram of the database



11) Some Analytical SQL Queries

Count of Medicines per Active Ingredient

```
mysql> SELECT ai.active_ingredient, COUNT(m.id) AS medicine_count
    -> FROM medicines m
    -> JOIN active_ingredients ai ON m.active_ingredient_id = ai.id
    -> GROUP BY ai.active_ingredient
    -> ORDER BY medicine_count DESC;
+-------------------------------------+----------------+
| active_ingredient                   | medicine_count |
+-------------------------------------+----------------+
| Amoxicillina/acido clavulanico      |            115 |
| Amlodipina                          |            101 |
| Amoxicillina                        |             30 |
| Aripiprazolo                        |             28 |
| Anastrozolo                         |             19 |
| Amlodipina/Valsartan                |              9 |
| Amlodipina/Valsartan/idroclorotiazide |              7 |
| Amitriptilina                       |              6 |
| Anagrelide                          |              5 |
| Apixaban                            |              5 |
| Ampicillina/sulbactam               |              2 |
| Apomorfina                          |              2 |
| Apremilast                          |              2 |
| Apraclonidina                       |              1 |
+-------------------------------------+----------------+
14 rows in set (0.00 sec)
```

Count of Medicines by Marketing Authorization Holder

```
mysql> SELECT mah.marketing_authorization_holder AS holder_name, COUNT(m.id) AS total_medicines
    -> FROM medicines m
    -> JOIN marketing_authorization_holders mah ON m.marketing_authorization_holder_id = mah.id
    -> GROUP BY mah.marketing_authorization_holder
    -> ORDER BY total_medicines DESC;
+--------------------------------+-----------------+
| holder_name                    | total_medicines |
+--------------------------------+-----------------+
| SANDOZ SpA                     |              18 |
| ZENTIVA ITALIA Srl             |              14 |
| GLAXOSMITHKLINE SpA            |              14 |
| DOC GENERICI Srl               |              14 |
| EG SpA                         |              13 |
| MYLAN SpA                      |              12 |
| TEVA ITALIA Srl                |              10 |
| AUROBINDO PHARMA ITALIA Srl    |               9 |
```

Find Medicines Belonging to a Specific ATC Code (e.g., "N04BC07")

```
mysql> SELECT m.medicine_name_and_packaging, ai.active_ingredient, ac.atc_code
    -> FROM medicines m
    -> JOIN active_ingredients ai ON m.active_ingredient_id = ai.id
    -> JOIN atc_codes ac ON m.atc_code_id = ac.id
    -> WHERE ac.atc_code = 'N04BC07';
+-------------------------------------------------+--------------------+----------+
| medicine_name_and_packaging                     | active_ingredient  | atc_code |
+-------------------------------------------------+--------------------+----------+
| APOFIN STYLO*1 penna preriempita SC 30 mg/3 ml  | Apomorfina         | N04BC07  |
| APOFIN STYLO*5 penne preriempita SC 30 mg/3 ml  | Apomorfina         | N04BC07  |
+-------------------------------------------------+--------------------+----------+
2 rows in set (0.00 sec)
```

Frequency of Incompatibility Mentions Across Medicines

```
mysql> SELECT
    ->     CASE
    ->         WHEN incompatibilities = 'Not available' THEN 'Missing'
    ->         ELSE 'Provided'
    ->     END AS incompatibility_status,
    ->     COUNT(*) AS count
    -> FROM medicines
    -> GROUP BY incompatibility_status;
+------------------------+-------+
| incompatibility_status | count |
+------------------------+-------+
| Provided               |   246 |
| Missing                |    86 |
+------------------------+-------+
2 rows in set (0.00 sec)
```

Top 10 Most Frequently Mentioned Therapeutic Indications

```
mysql> SELECT CONCAT(SUBSTRING(therapeutic_indications, 1, 100), '...') AS therapeutic_indications, COUNT(*) AS count
    -> FROM medicines
    -> GROUP BY therapeutic_indications
    -> ORDER BY count DESC
    -> LIMIT 10;
+----------------------------------------------------------------------------------------------------+-------+
| therapeutic_indications                                                                            | count |
+----------------------------------------------------------------------------------------------------+-------+
| Not available...                                                                                   |    85 |
| Ipertensione Angina pectoris cronica stabile Angina conseguente a vasospasmo (angina di Prinzmetal)... |    28 |
| Augmentin è indicato nel trattamento delle seguenti infezioni negli adulti e nei bambini (vedere par... |     8 |
| Trattamento dell'ipertensione essenziale come terapia sostitutiva in pazienti adulti nei quali la pr... |     7 |
| Clavulin è indicato nel trattamento delle seguenti infezioni negli adulti e nei bambini (vedere para... |     6 |
| Ipertensione Angina pectoris cronica stabile Angina conseguente a vasospasmo (angina di Prinzm... |     6 |
| Trattamento del tromboembolismo venoso (TEV) e prevenzione del TEV ricorrente nei pazienti pediatric... |     5 |
| AMOXICILLINA E ACIDO CLAVULANICO EG STADA è indicato per il trattamento delle seguenti infezioni neg... |     5 |
| Neoduplamox è indicato nel trattamento delle seguenti infezioni negli adulti e nei bambini (vedere p... |     5 |
| - Ipertensione. - Angina pectoris cronica stabile. - Angina conseguente a vasospasmo (angina di Prin... |     5 |
+----------------------------------------------------------------------------------------------------+-------+
10 rows in set, 1 warning (0.01 sec)
```

Most Common Side Effects Keywords (Using LIKE)

```
mysql> SELECT
    ->   COUNT(*) AS count,
    ->   'nausea' AS keyword
    -> FROM medicines
    -> WHERE undesirable_effects_side_effects LIKE '%nausea%'
    ->
    -> UNION ALL
    ->
    -> SELECT
    ->   COUNT(*),
    ->   'headache'
    -> FROM medicines
    -> WHERE undesirable_effects_side_effects LIKE '%headache%'
    ->
    -> UNION ALL
    ->
    -> SELECT
    ->   COUNT(*),
    ->   'rash'
    -> FROM medicines
    -> WHERE undesirable_effects_side_effects LIKE '%rash%';
+-------+----------+
| count | keyword  |
+-------+----------+
|   246 | nausea   |
|     0 | headache |
|   135 | rash     |
+-------+----------+
3 rows in set (0.01 sec)
```

Top 10 Longest Therapeutic Indications (For Review)

```
mysql> SELECT
    ->   id,
    ->   CHAR_LENGTH(therapeutic_indications) AS length,
    ->   SUBSTRING(therapeutic_indications, 1, 200) AS preview
    -> FROM medicines
    -> ORDER BY length DESC
    -> LIMIT 10;
+-----+--------+-------------------------------------------------------------
--------------------+
| id  | length | preview
                   |
+-----+--------+-------------------------------------------------------------
--------------------+
| 270 |   6849 | L’uso del medicinale è indicato per il trattamento delle inf
zioni delle v        |
| 269 |   6849 | L’uso del medicinale è indicato per il trattamento delle inf
zioni delle v        |
| 327 |   1182 | Aripiprazolo Teva è indicato per il trattamento della schizo
pisodi maniac        |
| 312 |   1182 | Aripiprazolo Teva è indicato per il trattamento della schizo
pisodi maniac        |
| 147 |   1160 | Amoxicillina Aurobindo Italia è indicata per il trattamento
Faringite e          |
| 152 |   1160 | Amoxicillina Aurobindo Italia è indicata per il trattamento
Faringite e          |
| 141 |   1150 | Amoxicillina Mylan Generics è indicato per il trattamento de
Faringite e t        |
| 236 |   1148 | Amoxicillina e Acido Clavulanico TecniGen Italia è indicato
.4 e 5.1):           |
| 237 |   1148 | Amoxicillina e Acido Clavulanico TecniGen Italia è indicato
.4 e 5.1):           |
| 239 |   1145 | Amoxicillina e Acido clavulanico PENSA PHARMA è indicato nel
e 5.1): • Sin         |
+-----+--------+-------------------------------------------------------------
--------------------+
10 rows in set (0.00 sec)
```

POWER BI Dashboard