Project: Data Management

Giovanni De Francesco, Roberto Di Lauro, Roberto Tessitore

June 2025

Contents

Dataset

The aim of the project work is to construct a structured dataset based on official data provided by the Italian Medicines Agency (AIFA), including information about regulatory and clinical aspects of medicines that are currently authorized in Italy. It has been first downloaded the dataset but after a few days AIFA website updated the dataset, so it could be seen the differences between the two datasets.

Differences between datasets

First of all it has been uploaded the datasets:

```
import pandas as pd

df_old = pd.read_csv("/home/roberto/Scrivania/data_management/
    Classe_H_per_principio_attivo_01-08-2024.csv", encoding="cp1252", sep=";")
df_old.rename(columns={'Codice \nAIC': 'Codice AIC'}, inplace=True)

df_new = pd.read_csv("/home/roberto/Scrivania/data_management/
    Classe_H_per_principio_attivo_15-10-2024.csv", encoding="cp1252", sep=";")
df_new.rename(columns={'Codice \nAIC': 'Codice AIC'}, inplace=True)
```

Let's take a look to the dataset:

	Principio Attivo	Descrizione Gruppo	Denominazione e Confezione	Prezzo al pubblico €	Prezzo Ex-factory €	Prezzo massimo di cessione €	Titolare AIC	Codice AIC	Codice Gruppo Equivalenza	Metri cubi ossigeno
0	Abacavir	ABACAVIR 20MG/ML 240ML USO ORALE	ZIAGEN*orale soluz flacone 240 ml 20 mg/ml	89,86	54,44	-	VIIV HEALTHCARE BV	34499020	AAB	NaN
1	Abacavir	ABACAVIR 300MG 60 UNITA' USO ORALE	ABACAVIR*60 cpr riv 300 mg	235,86	142,91	-	MYLAN SpA	45354040	AAC	NaN
2	Abacavir	ABACAVIR 300MG 60 UNITA' USO ORALE	ZIAGEN*60 cpr riv 300 mg	336,95	204,16	-	VIIV HEALTHCARE BV	34499018	AAC	NaN
3	Abacavir/lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	170,27	103,16	-	AUROBINDO PHARMA ITALIA Srl	44669012	AAD	NaN
4	Abacavir/Lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	60,00	36,36	-	SUN PHARMACEUTICAL IND.EU.B.V.	45348036	AAD	NaN
...	...	...	...	...	...	...	...	...	...	...
2193	Zidovudina	ZIDOVUDINA 100MG/10ML 200ML USO ORALE	RETROVIR*1 flacone 200 ml 100 mg/10 ml sciroppo	36,45	22,09	-	VIIV HEALTHCARE BV	26697058	BQF	NaN
2194	Zidovudina	ZIDOVUDINA 100MG/10ML 200ML USO ORALE	RETROVIR*1 flacone 200 ml 100 mg/10 ml scropp...	36,45	22,09	-	VIIV HEALTHCARE BV	26697134	BQF	NaN
2195	Zidovudina	ZIDOVUDINA 200MG 5 UNITA' USO PARENTERALE	RETROVIR*5 fiale EV 200 mg 20 ml	78,64	47,65	-	VIIV HEALTHCARE BV	26697072	TFA	NaN
2196	Zidovudina	ZIDOVUDINA 250MG 40 UNITA' USO ORALE	RETROVIR*40 cps 250 mg	144,13	87,33	-	VIIV HEALTHCARE BV	26697159	UZC	NaN
2197	Zolfo esafluoruro	ZOLFO ESAFLUORURO 8MCL/ML 5ML 1 UNITA' USO PAR...	SONOVUE*8 mcl/ml 1 flaconcino + 1 siringa 5 ml...	101,27	61,36	-	BRACCO INTERNATIONAL B.V.	35233028	BQI	NaN

2198 rows × 10 columns

It can be seen how many rows have been added, removed or modified;

```
key = 'Codice AIC'
df_old.set_index(key, inplace=True)
df_new.set_index(key, inplace=True)
```

```
nuove = df_new[~df_new.index.isin(df_old.index)]

rimosse = df_old[~df_old.index.isin(df_new.index)]

comuni_old = df_old[df_old.index.isin(df_new.index)]
comuni_new = df_new[df_new.index.isin(df_old.index)]

modificate = comuni_new[comuni_new.ne(comuni_old)].dropna(how='all')

invariati = comuni_new[comuni_new.eq(comuni_old)].dropna(how='all')
```

```
Added rows: 42
Removed rows: 28
Modified rows: 20
Unchanged rows: 2136
```

It can be seen also which column has been modified more times.

```
common_ids = df_old.index.intersection(df_new.index)
old_common = df_old.loc[common_ids]
new_common = df_new.loc[common_ids]

both_nan = old_common.isna() & new_common.isna()

diff_mask = (old_common != new_common) & ~both_nan

conteggio_modifiche = diff_mask.sum()

conteggio_modifiche = conteggio_modifiche[conteggio_modifiche > 0].sort_values(
    ascending=False)

print(conteggio_modifiche)
```

```
Prezzo Ex-factory €          13
Prezzo al pubblico €         13
Titolare AIC                 10
Principio Attivo              9
Denominazione e Confezione    2
Descrizione Gruppo            1
```

The prices are the most frequent updated values.
At this point it can be viewed for each AIC code, which column has been changed, the old value and also the new one:

```
differenze = []
colonne_comuni = df_old.columns.intersection(df_new.columns)

for idx in old_common.index:
    for col in colonne_comuni:
        old_val = old_common.loc[idx, col]
        new_val = new_common.loc[idx, col]

        if pd.isna(old_val) and pd.isna(new_val):
            continue

        if old_val != new_val:
            differenze.append({
                'Codice AIC': idx,
                'Campo': col,
                'Valore vecchio': old_val,
                'Valore nuovo': new_val
            })
```

```
df_diff = pd.DataFrame(differenze)

df_diff.dropna(how='all', inplace=True)
```

Codice AIC	Campo	Valore vecchio	Valore nuovo
50219056	Titolare AIC	CHIESI ITALIA SpA	CHIESI FARMACEUTICI SpA
49281037	Descrizione Gruppo	CABOTEGRAVIR 600MG 3ML 1 UNITA' USO PARENTERALE RP	CABOTEGRAVIR 600MG 3ML 1 UNITA' USO PARENTERALE
48961027	Prezzo al pubblico €	13038,16	11766,94
48961027	Prezzo Ex-factory €	7900,00	7129,75
48961015	Prezzo al pubblico €	2173,03	1961,16
48961015	Prezzo Ex-factory €	1316,67	1188,30
23288032	Principio Attivo	Fattore II/Fattore IX/Fattore X della coagulazione	Fattore II/Fattore IX/Fattore IX/Fattore X della coagulazione
49054036	Titolare AIC	GALAPAGOS NV	ALFASIGMA SpA
49054012	Titolare AIC	GALAPAGOS NV	ALFASIGMA SpA
36946010	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
36751016	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
36751016	Titolare AIC	IASON GmbH	CURIUM AUSTRIA GMBH
44466011	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
38827022	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
37149010	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
36751028	Principio Attivo	Fluoro 18F desossiglucosio	Fluoro-18F-desossiglucosio
36751028	Titolare AIC	IASON GmbH	CURIUM AUSTRIA GMBH
40383010	Principio Attivo	Fluorodopa 18F	Fluorodopa-18F
35143039	Prezzo al pubblico €	907,72	1072,76
35143039	Prezzo Ex-factory €	550,00	650,00
44244010	Prezzo al pubblico €	83,18	107,28
44244010	Prezzo Ex-factory €	50,40	65,00
35143015	Prezzo al pubblico €	226,93	268,19
35143015	Prezzo Ex-factory €	137,50	162,50
44244022	Prezzo al pubblico €	166,36	214,55
44244022	Prezzo Ex-factory €	100,80	130,00

44244022	Prezzo Ex-factory €	100,80	130,00
44244034	Prezzo al pubblico €	332,72	429,10
44244034	Prezzo Ex-factory €	201,60	260,00
35143027	Prezzo al pubblico €	453,86	536,38
35143027	Prezzo Ex-factory €	275,00	325,00
44244046	Prezzo al pubblico €	665,45	858,21
44244046	Prezzo Ex-factory €	403,20	520,00
35891011	Titolare AIC	GENZYME EUROPE B.V.	SANOFI B.V.
48215014	Prezzo al pubblico €	2888,20	2606,60
48215014	Prezzo Ex-factory €	1750,00	1579,38
48215026	Prezzo al pubblico €	11552,80	10426,40
48215026	Prezzo Ex-factory €	7000,00	6317,50
48215040	Prezzo al pubblico €	4126,00	3723,72
48215040	Prezzo Ex-factory €	2500,00	2256,25
41477011	Principio Attivo	Macroaggregati di albumina umana (macrosalb)	Tecnezio 99m Tc Albumina Umana Soluzione Iniettabile
38705024	Titolare AIC	ASTELLAS PHARMA EUROPE B.V.	SANDOZ PHARMACEUTICALS D.D.
38705012	Titolare AIC	ASTELLAS PHARMA EUROPE B.V.	SANDOZ PHARMACEUTICALS D.D.
40986022	Titolare AIC	IASON GmbH	CURIUM AUSTRIA GMBH
45219019	Titolare AIC	MYLAN PHARMACEUTICALS LIMITED	VIATRIS LIMITED
45452012	Denominazione e Confezione	INFLUVAC S TETRA*1 siringa preriempita IM SC 0,5 ml con ago 2023-2024	INFLUVAC S TETRA*1 siringa preriempita IM SC 0,5 ml con ago 2024-2025
45452036	Denominazione e Confezione	INFLUVAC S TETRA*10 siringhe preriempite IM SC 0,5 ml con ago 2023-2024	INFLUVAC S TETRA*10 siringhe preriempite IM SC 0,5 ml con ago 2024-2025
20430029	Prezzo al pubblico €	17,90	20,38
20430029	Prezzo Ex-factory €	8,13	12,35

Prepare the dataset

It has been downloaded the csv file of Class H medicines from the AIFA website and it has been read in python, with an encoding which is not utf-8 but Windows-1252, and the column name "Codice AIC" had a formatting "" inside so it has been removed.

```
import pandas as pd

df = pd.read_csv("/home/roberto/Scrivania/data_management/
    Classe_H_per_principio_attivo_15-10-2024.csv", encoding="cp1252", sep=";")
df.rename(columns={'Codice \nAIC': 'Codice AIC'}, inplace=True)
```

The number of rows assigned for the database construction are from 1 to 332 and the columns of interest are just:

- Principio Attivo
- Descrizione Gruppo
- Denominazione e Confezione
- Titolare AIC
- Codice AIC
- Codice Gruppo Equivalenza
- Classe (A or H to be added)

```
df = df[0:333]

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

df = df[columns]

df['classe'] = 'H'
```

It has been looked for some empty rows or duplicate entries but none was found.

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

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

```
print("Number of duplicated entries:", df.duplicated().sum())
```

```
Number of duplicated entries: 0
```

The final preprocessed dataset looks like this:

	Principio Attivo	Descrizione Gruppo	Denominazione e Confezione	Titolare AIC	Codice AIC	Codice Gruppo Equivalenza	classe
0	Abacavir	ABACAVIR 20MG/ML 240ML USO ORALE	ZIAGEN*orale soluz flacone 240 ml 20 mg/ml	VIIV HEALTHCARE BV	34499020	AAB	H
1	Abacavir	ABACAVIR 300MG 60 UNITA' USO ORALE	ABACAVIR*60 cpr riv 300 mg	MYLAN SpA	45354040	AAC	H
2	Abacavir	ABACAVIR 300MG 60 UNITA' USO ORALE	ZIAGEN*60 cpr riv 300 mg	VIIV HEALTHCARE BV	34499018	AAC	H
3	Abacavir/lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	AUROBINDO PHARMA ITALIA Srl	44669012	AAD	H
4	Abacavir/Lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	SUN PHARMACEUTICAL IND.EU.B.V.	45348036	AAD	H
...	...	...	...	...	...	...	...
328	Buprenorfina	BUPRENORFINA 8MG 7 UNITA' USO ORALE	BUPRENORFINA*7 cpr sublinguali 8 mg	SUN PHARMACEUTICAL IND.EU.B.V.	40643049	AGM	H
329	Buprenorfina	BUPRENORFINA 8MG 7 UNITA' USO ORALE	SUBUTEX*7 cpr sublinguali 8 mg	INDIVIOR EUROPE LIMITED	33791031	AGM	H
330	Buprenorfina	BUPRENORFINA 2MG 28 UNITA' USO ORALE	PREKISAN*28 cpr sublinguali 2 mg	G.L. PHARMA GmbH	49020035	MND	H
331	Buprenorfina	BUPRENORFINA 4MG 28 UNITA' USO ORALE	PREKISAN*28 cpr sublinguali 4 mg	G.L. PHARMA GmbH	49020074	MNE	H
332	Buprenorfina	BUPRENORFINA 8MG 28 UNITA' USO ORALE	PREKISAN*28 cpr sublinguali 8 mg	G.L. PHARMA GmbH	49020112	MNF	H

Building up the detailed dataset

To build up the detailed dataset it has been searched specific features of each drug on the website, and also from the summary of Package Leaflet download from the same website.
The procedure has been automatized thanks to two algorithms of web scraping and pdf parsing.

Web Scraping

In this case it has been used Selenium, a python package for dynamic web scraping, which simulate the usage of a web browser. It has benn created a function "find_code" which is:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.firefox.options import Options
import time
import os

def find_code(code):

    download_dir = "/home/roberto/Scrivania/data_management/pdf_download"
    os.makedirs(download_dir, exist_ok=True)

    options = Options()
    options.add_argument("--headless")
    profile = webdriver.FirefoxProfile()
    profile.set_preference("browser.download.folderList", 2)  # 2 = usa dir
        personalizzata
    profile.set_preference("browser.download.dir", download_dir)
    profile.set_preference("browser.helperApps.neverAsk.saveToDisk", "application/pdf
        ")
    options.profile = profile

    driver = webdriver.Firefox(options=options)
    driver.get("https://medicinali.aifa.gov.it/it/#/it/risultati?query=0" + str(code)
        )

    wait = WebDriverWait(driver, 20)


    accept = wait.until(EC.element_to_be_clickable((By.ID, "disclaimercheck")))
    accept.click()

    button = wait.until(EC.element_to_be_clickable((By.CLASS_NAME, "btn-outline-
        secondary")))
    button.click()

    time.sleep(3)
    try:
        element = wait.until(EC.element_to_be_clickable((By.CLASS_NAME, "custom-card-
            result")))
        element.click()
    except:
        driver.quit()
        return 'NOT AVAILABLE', 'NOT AVAILABLE', 'NOT AVAILABLE'

    url = driver.current_url

    codice_atc = wait.until(EC.presence_of_all_elements_located((By.CLASS_NAME, "text
        -black.small.ng-star-inserted")))
```

```
    atc = 'NOT AVAILABLE'

    for i in range(len(codice_atc)):
        if '-' in codice_atc[i].text:
            atc = codice_atc[i].text


    try:
        pdf = wait.until(EC.presence_of_all_elements_located((By.CLASS_NAME, "col-sm
            -6")))
        pdf[1].click()

        time.sleep(3)

        tabs = driver.window_handles
        driver.switch_to.window(tabs[1])
    except:
        driver.quit()
        return atc, 'NOT AVAILABLE', url



    pdf = driver.current_url
    driver.quit()

    return atc, pdf, url
```

This function takes as input the AIC code of a drug and this can be used to search on the AIFA website for the specific drug, this has been accomplished with:

```
    driver.get("https://medicinali.aifa.gov.it/it/#/it/risultati?query=0" + str(code)
        )
```

then there are two instruction to accept the terms and conditions of service:

```
    accept = wait.until(EC.element_to_be_clickable((By.ID, "disclaimercheck")))
    accept.click()

    button = wait.until(EC.element_to_be_clickable((By.CLASS_NAME, "btn-outline-
        secondary")))
    button.click()
```

Next step is to find the first result of the searching, and if it is not present, then our function needs to return "NOT AVAILABLE", if it is it clicks on it:

```
    try:
        element = wait.until(EC.element_to_be_clickable((By.CLASS_NAME, "custom-card-
            result")))
        element.click()
    except:
        driver.quit()
        return 'NOT AVAILABLE', 'NOT AVAILABLE', 'NOT AVAILABLE'
```

At this point it can be stored the url in a variable and if the atc code is found it can be stored otherwise it has "NOT AVAILABLE" as value:

```
    url = driver.current_url

    codice_atc = wait.until(EC.presence_of_all_elements_located((By.CLASS_NAME, "text
        -black.small.ng-star-inserted")))



    atc = 'NOT AVAILABLE'

    for i in range(len(codice_atc)):
```

```
        if '-' in codice_atc[i].text:
            atc = codice_atc[i].text
```

Last step is to find and download the pdf of the Summary of Product Characteristics, storing the pdf path which will be useful fot the pdf parsing part:

```
    try:
        pdf = wait.until(EC.presence_of_all_elements_located((By.CLASS_NAME, "col-sm
            -6")))
        pdf[1].click()

        time.sleep(3)

        tabs = driver.window_handles
        driver.switch_to.window(tabs[1])
    except:
        driver.quit()
        return atc, 'NOT AVAILABLE', url

    pdf = driver.current_url
    driver.quit()

    return atc, pdf, url
```

At the end it returns the ATC code, the pdf path and the url of the specific drug. We can apply this to the whole dataset for each row and create 3 new columns:

```
df[['Codice ATC', 'pdf', 'URL']] = df['Codice AIC'].apply(find_code).apply(pd.Series)
```

PDF Parsing

For pdf parsing it has been used the python library PyMuPDF and it has been created a function "find_all_text" to opend each pdf and extract the requested sections, which are:

- 4.1 Therapeutic indications
- 4.2 Posology and method of administration
- 4.3 Contraindications
- 4.4 Special warnings and precautions for use
- 4.5 Interactions with other medicinal products
- 4.6 Fertility, pregnancy and lactation
- 4.7 Effects on ability to drive and use machines
- 4.8 Undesirable effects (side effects)
- 4.9 Overdose
- 6.2 Incompatibilities

Since the name of the section could vary it has been created a dictionary that connect each paragraph number with a list of all possible names of the section(eg. à in a', d'impiego in di impiego), like this:

```
    sezion = {4.1: ["4.1 \nIndicazioni terapeutiche",
                    "4.1\nIndicazioni terapeutiche"],
              4.2: ["4.2 \nPosologia e modo di somministrazione",
                    "4.2 \nPosologia e modo di somministrazione"],
              4.3: ["4.3 \nControindicazioni",
                    "4.3\nControindicazioni"],
              4.4: ["4.4 \nAvvertenze speciali e precauzioni  d impiego ",
                    "4.4 \nAvvertenze speciali e precauzioni di impiego"],
              4.5: ["4.5 \nInterazioni con altri medicinali ed altre forme
                   d interazione ",
```

```
            "4.5 \nInterazioni con altri medicinali ed altre forme di interazione
                "],
        4.6: ["4.6 \nFertilit , gravidanza e allattamento",
              "4.6 \nFertilita', gravidanza e allattamento"],
        4.7: ["4.7 \nEffetti sulla capacit  di guidare veicoli e  s u l l uso  di
            macchinari",
              "4.7 \nEffetti sulla capacita' di guidare veicoli e  s u l l uso  di
                  macchinari"],
        4.8: ["4.8 \nEffetti indesiderati",
              "4.8\nEffetti indesiderati"],
        4.9: ["4.9 \nSovradosaggio",
              "4.9 \nSovradosaggio"],
        5.0: ["5. \nPROPRIET  FARMACOLOGICHE",
              "5. \nPROPRIETA  FARMACOLOGICHE"],
        6.2: ["6.2 \nIncompatibilit ",
              "6.2 \nIncompatibilita'"],
        6.3: ["6.3 \nPeriodo di validit ",
              "6.3 \nPeriodo di validita'"]}
```

then it opens the pdf and read the text from the file:

```
    if pdf_path == "ILABLE":
        return ['NOT AVAILABLE'] * len(sezioni)

    with fitz.open(pdf_path) as pdf:
        for page in pdf:
            testo = page.get_text("text", clip=fitz.Rect(0, 50, page.rect.width, 795)
                )
            testo_totale += testo.strip()

    testo_totale = testo_totale.replace("Agenzia Italiana del Farmaco", "")
    testo_totale = testo_totale.replace("-\n", "-")
    testo_totale = testo_totale.replace("- \n", "-")
    testo_totale = testo_totale.replace("    \n", "    ")
    testo_totale = testo_totale.replace("   \n", "    ")

    testi = []
```

at the end it search for one of the title in the section list for each section and it append each section in a list which is returned at the end of the function, if it doesn't find any it returns "NOT AVAILABLE":

```
    for s in sezioni:
        start_idx = -1
        end_idx = -1
        for i in range(len(sezion[s])):
            if start_idx == -1:
                start_idx = testo_totale.find(sezion[s][i])
                ini = i
            else:
                break
        for i in range(len(sezion[round(s + 0.1, 1)])):
            if end_idx == -1:
                end_idx = testo_totale.find(sezion[round(s + 0.1, 1)][i])
                fin = i
            else:
                break

        sezione = testo_totale[start_idx + len(sezion[s][ini]) :end_idx].strip()

        testi.append(sezione)


    return testi
```

This is applied to the whole dataset

```
sezioni = [4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 6.2]
df[sezioni] = df['pdf'].apply(lambda x: find_all_text(x[7:])).apply(pd.Series)
```

and the "pdf" column is not useful anymore so it can be dropped:

```
df.drop(columns=['pdf'], inplace=True)
```

Now the full dataset has been completed and it has been saved as a csv file.

Building a relational database

Importing the AIFA Dataset into SQL Server using R

The dataset, provided in CSV format and containing information about AIFA-registered drugs (approximately 333 rows and 20 columns), was imported into a Microsoft SQL Server relational database using the R programming language and the odbc package.

The CSV file was read into R using the read_delim function from the readr package, specifying the delimiter as ; according to the European format. The following code was used:

```
library(readr)

finale <- read_delim("C:/Users/wrath/Downloads/finale.csv",
                     delim = ";",
                     escape_double = FALSE,
                     trim_ws = TRUE)
```

Listing 1: Reading the CSV file in R

After loading the data, a connection to the local SQL Server instance was established using dbConnect with the appropriate ODBC driver. The dataframe was then uploaded to the database Progetto using the dbWriteTable function, overwriting any existing table with the same name. After executing the script, the table Aifa_GroupH_1_333 becomes available in the Progetto database, as shown in Figure 1.

```
library(DBI)
library(odbc)

con <- dbConnect(odbc::odbc(),
                 Driver = "ODBC Driver 17 for SQL Server",
                 Server = "LocalHost\\SQLexpress",
                 Database = "Progetto",
                 Trusted_Connection = "Yes")

dbWriteTable(con, "Aifa_GroupH_1_333", finale, overwrite = TRUE, row.names = FALSE)
```

Listing 2: Connecting to SQL Server and writing the table

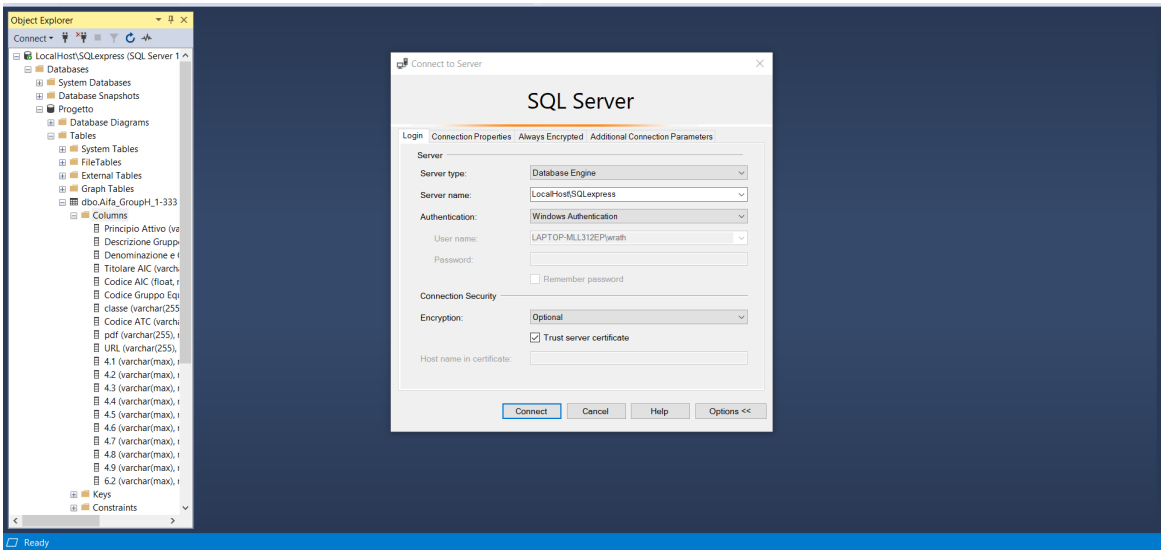


Figure 1: Table `Aifa_GroupH_1_333` visualized in SQL Server Management Studio.

Querying the Database

Once the dataset was successfully imported into the SQL Server database, several SQL queries were executed to explore and analyze the data. Querying the database allows us to retrieve specific information, perform aggregations, and extract meaningful statistics about the pharmaceutical products registered in the AIFA dataset.

The following image shows an example of a SQL query retrieving all records from the imported table, along with the resulting data displayed within SQL Server Management Studio.

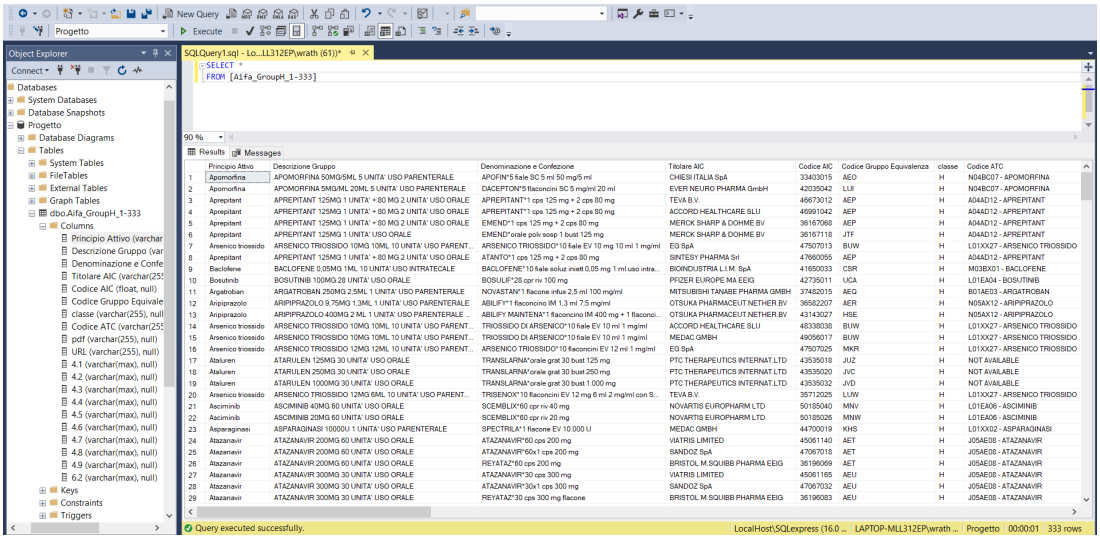


Figure 2: SQL query used to retrieve all records from the `Aifa_GroupH_1_333` table.

In the database are collected information about 333 drugs, belonging to the group H of AIFA drugs. There are 94 distinct active principles, 109 different companies, 203 equivalence group between drugs.

Creating a View to Analyze Drug Equivalence Groups

In order to analyze the competitiveness and composition of drug equivalence groups, we created a view called `EQUIVALENCE_COMPANY`. This view provides enriched information by including, for each drug:

- the name of the company (`Titolare AIC`),
- the active ingredient (`Principio Attivo`),

- the full commercial name (`Denominazione e Confezione`),
- the equivalence group code (`Codice Gruppo Equivalenza`),
- the number of distinct companies offering drugs in that group,
- the total number of drug variants available in that group.

This allows us to focus on equivalence groups where more than one company is present, highlighting areas of potential competition. The query used to define the view is shown below:

```
CREATE VIEW EQUIVALENCE_COMPANY AS
SELECT
    A.[Titolare AIC],
    A.[Principio Attivo],
    A.[Denominazione e Confezione],
    A.[Codice Gruppo Equivalenza],
    X.num_aziende_gruppo,
    Y.num_farmaci_gruppo
FROM
    [Aifa_GroupH_1-333] A
JOIN (
    SELECT
        [Codice Gruppo Equivalenza],
        COUNT(DISTINCT [Titolare AIC]) AS num_aziende_gruppo
    FROM
        [Aifa_GroupH_1-333]
    GROUP BY
        [Codice Gruppo Equivalenza]
) X ON A.[Codice Gruppo Equivalenza] = X.[Codice Gruppo Equivalenza]
JOIN (
    SELECT
        [Codice Gruppo Equivalenza],
        COUNT(*) AS num_farmaci_gruppo
    FROM
        [Aifa_GroupH_1-333]
    GROUP BY
        [Codice Gruppo Equivalenza]
) Y ON A.[Codice Gruppo Equivalenza] = Y.[Codice Gruppo Equivalenza]
WHERE
    X.num_aziende_gruppo > 1;
```

Listing 3: View to analyze drug equivalence group composition

To explore the results of this view, we queried it as follows:

```
SELECT *
FROM EQUIVALENCE_COMPANY
ORDER BY num_aziende_gruppo DESC;
```

Listing 4: Retrieving competitive equivalence groups

This final query returns all drugs belonging to equivalence groups shared by multiple companies, sorted by the number of companies per group. It is useful to highlight competitive areas in the market.

	Titolare AIC	Principio Attivo	Denominazione e Confezione	Codice Gruppo Equivalenza	num_aziende_gruppo	num_farmaci_gruppo
1	MEDAC GMBH	Bortezomib	BORTEZOMIB*1 flaconcino EV SC polv 3,5 mg	AGJ	13	13
2	JANSSEN CILAG INTERNAT.N.V.	Bortezomib	VELCADE*1 flaconcino polv EV SC 3,5 mg	AGJ	13	13
3	HIKMA PHARMA GMBH	Bortezomib	BORTEZOMIB*1 flaconcino polv 3,5 mg	AGJ	13	13
4	EG SpA	Bortezomib	BORTEZOMIB*1 flaconcino EV SC 2,5 mg/ml 3,5 mg 1...	AGJ	13	13
5	DR. REDDY'S Srl	Bortezomib	BORTEZOMIB*1 flaconcino polv 3,5 mg	AGJ	13	13
6	ZENTIVA ITALIA Srl	Bortezomib	BORTEZOMIB*1 flaconcino polv EV SC 3,5 mg	AGJ	13	13
7	EUGIA PHARMA (MALTA) LIMITED	Bortezomib	BORTEZOMIB*1 flaconcino polv EV SC 3,5 mg	AGJ	13	13
8	FRESENIUS KABI DEUTSCH.GMBH	Bortezomib	BORTEZOMIB*1 flaconcino polv EV SC 3,5 mg	AGJ	13	13
9	SUN PHARMACEUTICAL IND.EU.B.V.	Bortezomib	BORTEZOMIB*1 flacone EV SC 3,5 mg	AGJ	13	13
10	MYLAN SpA	Bortezomib	BORTEZOMIB*EV 1 flaconcino polv 3,5 mg	AGJ	13	13
11	EVER VALINJECT GMBH	Bortezomib	BORTEZOMIB*EV SC 1 flaconcino 1 ml 2,5 mg/ml	AGJ	13	13
12	ACCORD HEALTHCARE SLU	Bortezomib	BORTEZOMIB*EV SC 1 flaconcino 1,4 ml 2,5 mg/ml	AGJ	13	13
13	SANDOZ B.V.	Bortezomib	BORTEZOMIB*EV SC 1 flaconcino polv 3,5 mg	AGJ	13	13
14	ZENTIVA ITALIA Srl	Abiraterone	ABIRATERONE*56 cpr riv 500 mg	KII	10	10
15	ACCORD HEALTHCARE SLU	Abiraterone	ABIRATERONE*56x1 cpr riv 500 mg	KII	10	10
16	QILU PHARMA SPAIN S.L.	Abiraterone	ABIRATERONE*14x4 cpr riv 500 mg	KII	10	10
17	MEDAC GMBH	Abiraterone	ABIRATERONE*56 cpr riv 500 mg	KII	10	10

Figure 3: Results from the `EQUIVALENCE_COMPANY` view query.

It is possible to observe how different pharmaceutical companies supply drugs that belong to the same equivalence group. Active ingredients such as Bortezomib and Abiraterone are produced by multiple companies, highlighting a highly competitive and diverse market for these compounds.

Identifying Monopoly Cases within Equivalence Groups

The following query aims to identify **equivalence groups that are exclusively supplied by a single pharmaceutical company**. This may indicate the presence of a monopoly, where no alternative manufacturer exists for a given active principle within the same equivalence group.

By grouping the dataset by both the `Codice Gruppo Equivalenza` and the `Principio Attivo`, and selecting only the groups where exactly one company (`Titolare AIC`) is present, we highlight those situations where competition is completely absent. There are also shown which are the relatives companies.

```
CREATE VIEW UNIQUE_EQUIVALENCE_GROUP_COMPANY AS
SELECT
    [Codice Gruppo Equivalenza],
    [Principio Attivo],
    [Titolare AIC]
FROM
    [Aifa_GroupH_1-333] A
WHERE
    [Codice Gruppo Equivalenza] IN (
        SELECT [Codice Gruppo Equivalenza]
        FROM [Aifa_GroupH_1-333]
        GROUP BY [Codice Gruppo Equivalenza]
        HAVING COUNT(DISTINCT [Titolare AIC]) = 1
    );
```

Listing 5: Query to list monopoly equivalence groups along with the exclusive company

Once the view is created, we can display all such monopoly cases ordered by the number of companies (always equal to 1):

```
SELECT * FROM UNIQUE_EQUIVALENCE_GROUP_COMPANY
ORDER BY [Codice Gruppo Equivalenza];
```

Listing 6: Querying the monopoly groups

	Codice Gruppo Equivalenza	Principio Attivo	Titolare AIC
1	AAB	Abacavir	VIIV HEALTHCARE BV
2	AAF	Abatacept	BRISTOL M.SQUIBB PHARMA EEIG
3	AAJ	Acido diatrizoico	BAYER SpA
4	AAM	Abatacept	BRISTOL M.SQUIBB PHARMA EEIG
5	AAM	Abatacept	BRISTOL M.SQUIBB PHARMA EEIG
6	ABM	Acido levofolinico	MEDAC GMBH
7	ABN	Acido levofolinico	MEDAC GMBH
8	ABT	Acido pamidronico	PHARMATEX ITALIA Srl
9	ABU	Acido pamidronico	PHARMATEX ITALIA Srl
10	ABW	Acido pamidronico	PHARMATEX ITALIA Srl
11	ABZ	Acido zoledronico	MEDAC GMBH
12	ACF	Bcg - Bacillo di Calmette e Guerin	MEDAC GMBH
13	ACH	Adalimumab	CELLTRION HEALTHCARE HUNG.KFT
14	ACO	Agalsidasi Alfa	TAKEDA PHARM.INT.AG IR.BRANCH
15	ACP	Agalsidasi Beta	GENZYME EUROPE B.V.
16	ACT	Aldesleuchina	CLINIGEN HEALTHCARE B.V.
17	ACW	Alfa 1 Antitripsina Umana	INSTITUTO GRIFOLS S.A.

Figure 4: Monopoly equivalence groups, i.e., those supplied by a single pharmaceutical company.

Analyzing Different Equivalence Groups for the Same Active Principle

This query and corresponding view identify active principles that belong to multiple equivalence groups. The goal is to verify whether these different groups correspond to variations in packaging, dosage, or formulation.

Typically, changes in the equivalence group are justified by differences in drug format or dosage. By listing the active principle together with the equivalence group, package description, and company, we can better understand the rationale behind these groupings.

```
CREATE VIEW DIFFERENT_PACKAGES_COMPANY AS
SELECT
    [Principio Attivo],
    [Codice Gruppo Equivalenza],
    [Denominazione e Confezione],
    [Titolare AIC]
FROM
    [Aifa_GroupH_1-333]
WHERE
    [Principio Attivo] IN (
        SELECT [Principio Attivo]
        FROM [Aifa_GroupH_1-333]
        GROUP BY [Principio Attivo]
        HAVING COUNT(DISTINCT [Codice Gruppo Equivalenza]) > 1
    );
```

Listing 7: View to identify active principles assigned to multiple equivalence groups

To view the contents of the view, the following query can be used:

```
SELECT *
FROM DIFFERENT_PACKAGES_COMPANY
ORDER BY
    [Principio Attivo],
    [Codice Gruppo Equivalenza],
    [Denominazione e Confezione];
```

Listing 8: Query to select from the view and order results

	Principio Attivo	Codice Gruppo Equivalenza	Denominazione e Confezione	Titolare AIC
1	Abacavir	AAB	ZIAGEN*orale soluz flacone 240 ml 20 mg/ml	VIIV HEALTHCARE BV
2	Abacavir	AAC	ABACAVIR*60 cpr riv 300 mg	MYLAN SpA
3	Abacavir	AAC	ZIAGEN*60 cpr riv 300 mg	VIIV HEALTHCARE BV
4	Abatacept	AAF	ORENCIA*3 flaconcini EV 250 mg polv + 3 siringhe	BRISTOL M.SQUIBB PHARMA EEIG
5	Abatacept	AAM	ORENCIA*4 penne preriemp clickject SC 125 mg 1 ml	BRISTOL M.SQUIBB PHARMA EEIG
6	Abatacept	AAM	ORENCIA*4 siringhe SC 125 mg 1 ml	BRISTOL M.SQUIBB PHARMA EEIG
7	Abemaciclib	KRK	VERZENIOS*28 cpr riv 100 mg	ELI LILLY NEDERLAND BV
8	Abemaciclib	KRL	VERZENIOS*28 cpr riv 150 mg	ELI LILLY NEDERLAND BV
9	Abemaciclib	KRM	VERZENIOS*28 cpr riv 50 mg	ELI LILLY NEDERLAND BV
10	Abiraterone	AAH	ABIRATERONE*120 cpr 250 mg flacone	ACCORD HEALTHCARE SLU
11	Abiraterone	AAH	ABIRATERONE*120 cpr 250 mg flacone	MEDAC GMBH
12	Abiraterone	AAH	ABIRATERONE*120 cpr riv 250 mg flacone	SANDOZ SpA

Figure 5: Query result showing active principles belonging to multiple equivalence groups, with details on packaging and company.

The results (Figure 5) confirm that, in most cases, the assignment to different equivalence groups corresponds to meaningful differences in dosage or packaging. In other situations, different companies may use slightly different commercial names or formulations for the same active principle, which also contributes to the creation of multiple equivalence groups.

Analysis of Variant Counts per Active Principle

To better understand how drugs vary across formats and manufacturers, we created the view `VARIANT_PRINCIPIO_COMPANY`, which aggregates the number of commercial variants and distinct companies for each active principle.

The goal of this analysis is to identify active principles that:

- are commercialized in multiple dosage forms or packages;
- are offered by multiple companies (i.e., have multiple `Titolare AIC`);
- are likely to be widely used and demanded.

The following SQL view was created to extract these statistics:

```
CREATE VIEW VARIANT_PRINCIPIO_COMPANY AS
SELECT
    [Principio Attivo],
    COUNT(DISTINCT [Denominazione e Confezione]) AS Num_Varianti,
    COUNT(DISTINCT [Titolare AIC]) AS Num_Aziende
FROM
    [Aifa_GroupH_1-333]
GROUP BY
    [Principio Attivo]
```

Listing 9: Creating the VARIANT_PRINCIPIO_COMPANY view

The query counts how many unique packaging variants (`Denominazione e Confezione`) exist for each active principle, and how many different companies are associated with it. This allows us to identify drugs that are highly diversified and commercialized.

The results, ordered by the number of variants, highlight substances like *Adalimumab* and *Bevacizumab*, which show a large number of product variants and are manufactured by several companies, confirming their importance in the pharmaceutical market.

	Principio Attivo	Num_Varianti	Num_Aziende
1	Adalimumab	27	6
2	Bevacizumab	16	8
3	Bendamustina	13	8
4	Antitrombina III Umana	12	5
5	Bortezomib	11	13
6	Abiraterone	10	11
7	Baclofene	9	3
8	Adrenalina	7	3
9	Buprenorfina	7	4
10	Bosutinib	6	2
11	Acido zoledronico	6	8
12	Atosiban	6	4
13	Amoxicillina/acido clavulanico	6	3
14	Atazanavir	6	3
15	Avapritinib	5	1
16	Aflibercept	5	2
17	Alitretinoina	4	2

Figure 6: View `VARIANT_PRINCIPIO_COMPANY` shown in SQL Server.

Interestingly, some active principles show a high number of associated companies but only a limited number of packaging variants. This pattern may indicate a highly standardized formulation, limited clinical flexibility in dosages, or the presence of generic drugs where multiple companies offer the same product variant. Such cases suggest strong market competition over a narrow product range.

Therapeutic Classification (ATC) Analysis

To better understand the therapeutic scope of the dataset, we performed several queries on the `ATC` (Anatomical Therapeutic Chemical) classification codes. This allowed us to analyse the distribution of drugs and companies across different therapeutic classes.

Number of Drugs per ATC Code

To understand which therapeutic classes are most populated, we created a view that counts the number of drugs per ATC code.

```
CREATE VIEW ATC_DRUGS AS
SELECT
    [Codice ATC],
    COUNT(*) AS Num_Farmaci
FROM
    [Aifa_GroupH_1-333]
GROUP BY
    [Codice ATC];

SELECT *
FROM ATC_DRUGS
ORDER BY
    Num_Farmaci DESC;
```

Listing 10: Create view for number of drugs per ATC code

Results Table:

	Codice ATC	Num_Farmaci
1	L04AB04 - ADALIMUMAB	27
2	L01AA09 - BENDAMUSTINA	19
3	L02BX03 - ABIRATERONE ACETATO	16
4	L01FG01 - BEVACIZUMAB	16
5	L01XG01 - BORTEZOMIB	14
6	B01AB02 - ANTITROMBINA III	12
7	L01BC07 - AZACITIDINA	11
8	M03BX01 - BACLOFENE	9
9	M05BA08 - ACIDO ZOLEDRONICO	9
10	G02CX01 - ATOSIBAN	8
11	C01CA24 - ADRENALINA	7
12	N07BC01 - BUPRENORFINA	7
13	NOT AVAILABLE	7
14	J01CR02 - AMOXICILLINA E INIBITORI ENZIMATICI	6
15	L01EA04 - BOSUTINIB	6
16	J05AE08 - ATAZANAVIR	6
17	A04AD12 - APREPITANT	5

Figure 7: Number of drugs per ATC code

Number of Companies per ATC Code

This query counts the number of distinct marketing authorisation holders (*Titolare AIC*) per therapeutic group.

```
CREATE VIEW ATC_COMPANY AS
SELECT
    [Codice ATC],
    COUNT(DISTINCT [Titolare AIC]) AS Num_Aziende
FROM
    [Aifa_GroupH_1-333]
GROUP BY
    [Codice ATC];


SELECT *
FROM ATC_COMPANY
ORDER BY
    Num_Aziende DESC;
```

Listing 11: Create view for number of companies per ATC code

	Codice ATC	Num_Aziende
1	L01XG01 - BORTEZOMIB	13
2	L02BX03 - ABIRATERONE ACETATO	11
3	L01BC07 - AZACITIDINA	9
4	L01AA09 - BENDAMUSTINA	8
5	L01FG01 - BEVACIZUMAB	8
6	M05BA08 - ACIDO ZOLEDRONICO	8
7	L04AB04 - ADALIMUMAB	6
8	B01AB02 - ANTITROMBINA III	5
9	A04AD12 - APREPITANT	4
10	C01BD01 - AMIODARONE	4
11	J02AX06 - ANIDULAFUNGINA	4
12	G02CX01 - ATOSIBAN	4
13	L01XX27 - ARSENICO TRIOSSIDO	4
14	M03BX01 - BACLOFENE	3
15	J01CR02 - AMOXICILLINA E INIBITORI ENZIMATICI	3
16	J05AE08 - ATAZANAVIR	3

Figure 8: Number of companies per ATC code

Analysis of Antibiotic Class (J01)

We filtered drugs belonging to the ATC class J01, which includes antibiotics, in order to examine their composition and company distribution.

```
CREATE VIEW ATC_ANTIBIOTIC AS
SELECT
    [Titolare AIC],
    [Principio Attivo],
    [Denominazione e Confezione],
    [Codice Gruppo Equivalenza]
FROM
    [Aifa_GroupH_1-333]
WHERE
    [Codice ATC] LIKE 'J01%';


SELECT *
FROM ATC_ANTIBIOTIC
ORDER BY
    [Principio Attivo], [Titolare AIC];
```

Listing 12: Create view for antibiotics (J01) class

	Titolare AIC	Principio Attivo	Denominazione e Confezione	Codice Gruppo Equivalenza
1	NEW RESEARCH Srl	Amikacina	NEKACIN*5 fiale IM EV 500 mg 2 ml	ADP
2	PHARMATEX ITALIA Srl	Amikacina	AMICASIL*10 fiale IM EV 500 mg 2 ml	ADO
3	TEVA ITALIA Srl	Amikacina	AMIKACINA*5 fiale 0,5 g 2 ml	ADP
4	IBIGEN Srl	Amoxicillina/acido clavulanico	AMOXICILLINA E ACIDO CLAVULANICO*10 flaconcini po...	GYG
5	IBIGEN Srl	Amoxicillina/acido clavulanico	AMOXICILLINA E ACIDO CLAVULANICO*10 flaconcini EV...	KHV
6	SANDOZ A/S	Amoxicillina/acido clavulanico	AMOXICILLINA E ACIDO CLAVULANICO*1 flaconcino EV ...	ADX
7	SANDOZ A/S	Amoxicillina/acido clavulanico	AMOXICILLINA E ACIDO CLAVULANICO*1 flaconcino EV ...	ADV
8	TEVA ITALIA Srl	Amoxicillina/acido clavulanico	AMOXICILLINA ACIDO CLAVULANICO*1 fiala EV 1.000 ...	ADW
9	TEVA ITALIA Srl	Amoxicillina/acido clavulanico	AMOXICILLINA ACIDO CLAVULANICO*1 flacone EV 2.00...	ADX
10	INCA-PHARM Srl	Azitromicina	AZYLUNG*1 flaconcino EV 500 mg	AFH
11	PFIZER ITALIA Srl	Azitromicina	ZITROMAX*1 flaconcino EV 500 mg	AFH

Figure 9: Antibiotic drugs (ATC code J01)

Active Principles Used in Multiple ATC Categories

This view identifies active principles that appear in more than one ATC category, indicating drugs with multiple therapeutic uses.

```sql
CREATE VIEW ATC_PRINCIPIO_ATTIVO AS
SELECT
    [Principio Attivo],
    COUNT(DISTINCT [Codice ATC]) AS Num_ATC
FROM
    [Aifa_GroupH_1-333]
GROUP BY
    [Principio Attivo]
HAVING COUNT(DISTINCT [Codice ATC]) > 1;

SELECT *
FROM ATC_PRINCIPIO_ATTIVO
ORDER BY
    Num_ATC DESC;
```

Listing 13: Create view for active principles in multiple ATC groups

	Principio Attivo	Num_ATC
1	Aflibercept	2
2	Buprenorfina	2

Figure 10: Active principles used in multiple therapeutic classes

Data Visualization: Power BI Dashboard

To enhance the interpretability of the dataset and make the extracted information more accessible, we developed an interactive dashboard using Power BI. This dashboard was constructed by importing the cleaned and structured data from our SQL database and was driven by the queries previously designed during the data management phase.

The purpose of this visualization layer is to provide both a high-level overview and detailed insights into the dataset, enabling users to explore relationships and distributions within the pharmaceutical data in an intuitive way. Each visual component is connected to the underlying data model and supports interactive filtering, allowing users to dynamically adjust the view based on specific variables or categories.

The dashboard serves as a bridge between raw data and strategic understanding: it transforms complex tables into meaningful visual narratives that support interpretation, pattern recognition, and data-driven decisions.

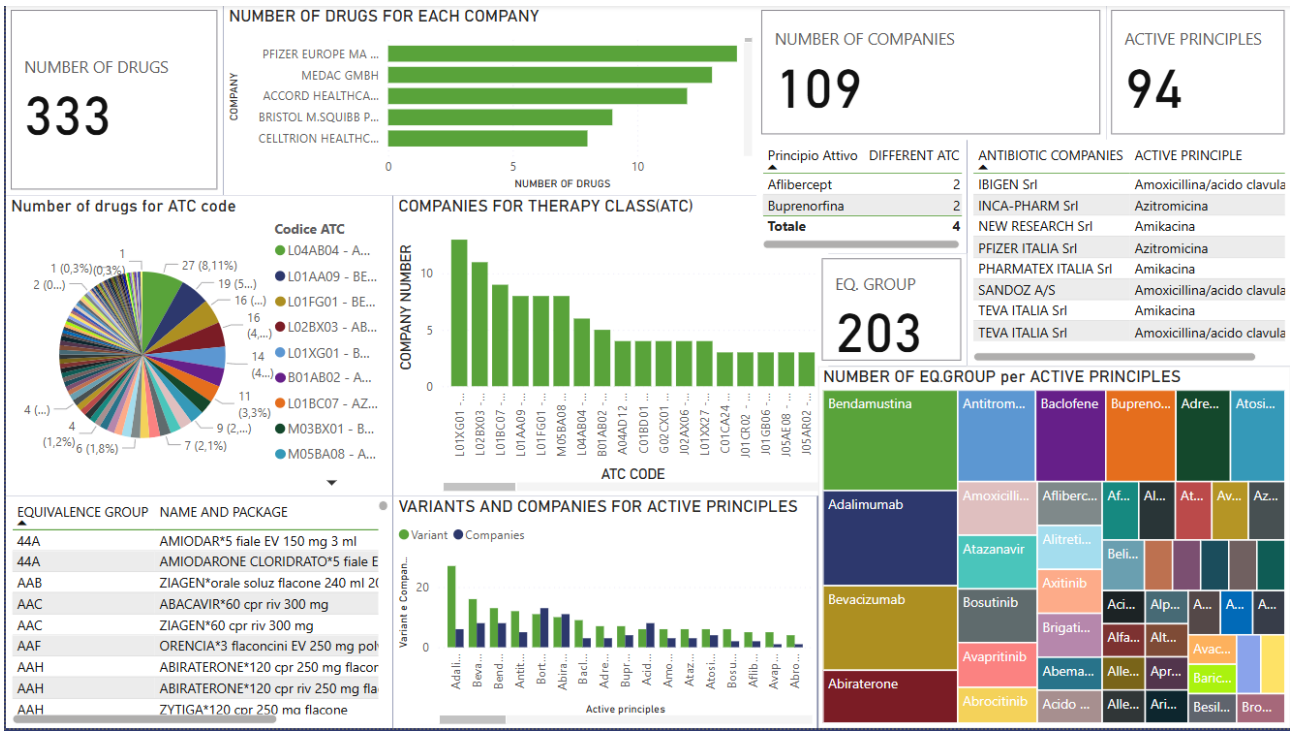


Figure 11: Interactive Power BI dashboard summarizing the pharmaceutical dataset. The visualizations include key metrics, drug distribution by company and ATC code, equivalence groups, and variant analysis per active principle.

The dashboard is composed of several visual components designed to summarize and explore the key dimensions of the pharmaceutical dataset. At the top, three KPIs display the total number of drugs (333), companies (109), and active principles (94), giving an immediate overview of the dataset's scale. A bar chart shows the distribution of drugs across companies, highlighting the most prolific producers such as Pfizer Europe MA and Medac GmbH.

A pie chart presents the number of drugs per ATC code, offering insight into the therapeutic class distribution. Another bar chart shows how many companies are associated with each ATC class, providing a measure of market diversity within each therapeutic area.

Equivalence groups (EQ groups) are analyzed both numerically—203 in total—and visually through a tree map, which links each EQ group to its active principle, revealing the most frequently substituted compounds. An accompanying table lists EQ groups alongside drug names and packaging formats.

Additionally, a dual-axis bar chart displays the number of variants and the number of companies for each active principle, supporting a comparison between chemical diversity and commercial presence.

All charts are fully interactive, allowing users to filter by active principle, company, or therapeutic class to explore relationships within the data dynamically. This interactivity enables a more flexible and user-driven analysis experience.

In addition to the main dashboard, we developed an interactive Sankey diagram to further explore the relationships among the different entities in our dataset. This visualization highlights the flow from marketing authorization holders (Titolar AIC) to active principles, and then to the corresponding equivalence groups. Each path in the diagram represents the number of drugs associated with a specific combination of company, active substance, and equivalence classification. This visual tool allows users to quickly grasp how certain companies are linked to specific active substances and how these are categorized into therapeutic equivalence groups. The Sankey diagram enhances the interpretability of the data by providing a clear and intuitive overview of the distribution and interconnection of drug-related entities.

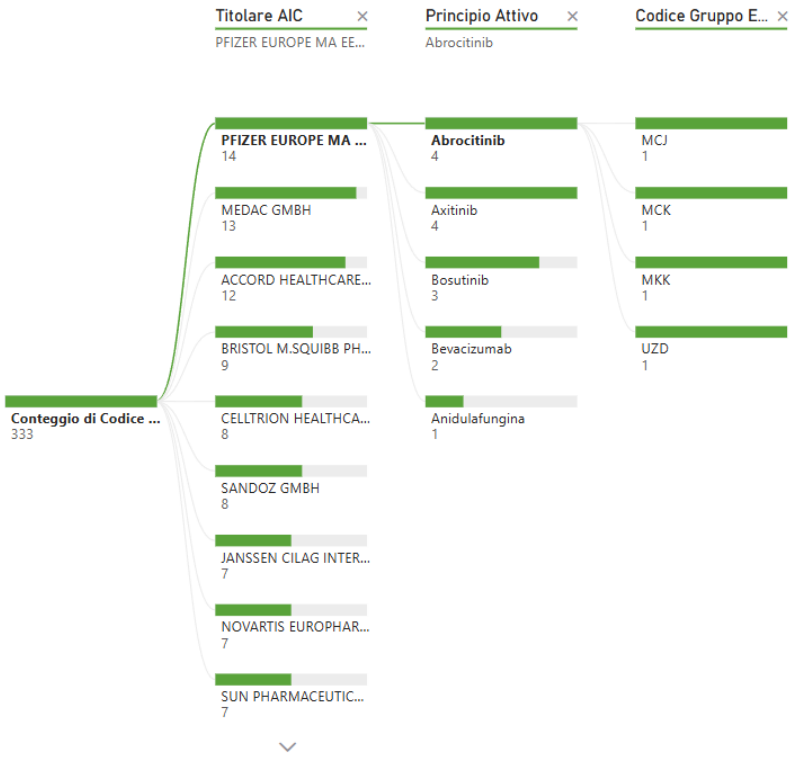


Figure 12: Sankey diagram showing the flow of drugs from marketing authorization holders (Titolar AIC) to active principles and equivalence groups. This interactive visualization helps to explore the relationships and distributions among companies, active substances, and equivalence classes.

Another Sankey diagram was created to visualize the relationship between the AIC codes (marketing authorization identifiers), the drug names and packaging, and their corresponding ATC (Anatomical Therapeutic Chemical) codes. This visualization provides a clear representation of how each AIC code maps to a specific formulation and packaging of a drug, and how these are classified according to therapeutic classes. It is particularly useful for identifying how different product variants (e.g., dosage forms or pack sizes) are distributed across ATC codes. This view enhances the granularity of the analysis, offering a more detailed understanding of the dataset's structure and allowing users to trace the regulatory and therapeutic classification paths of individual products.

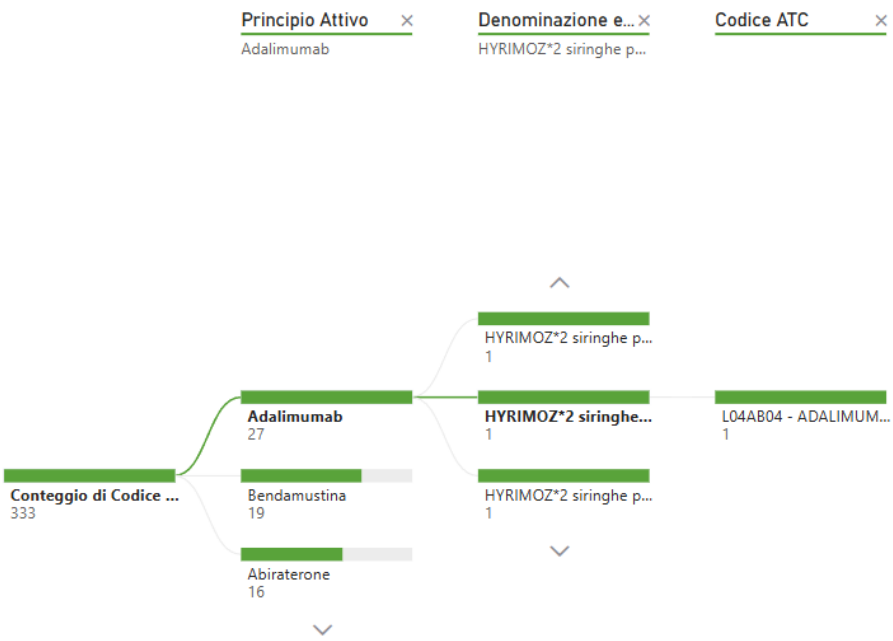


Figure 13: Sankey diagram showing the flow from AIC codes (Marketing Authorization Codes) through drug names and packaging, to ATC codes (Anatomical Therapeutic Chemical classification). This interactive visualization helps explore the relationships and distributions between AIC codes, their corresponding drug names and packages, and the ATC classification of active substances.

LLM in Non-Structured Variables

After creating the dataset in its entirety, it was observed that it contains both structured data (typical of SQL databases) and unstructured data, which is more similar in nature to what is commonly handled in NoSQL systems. Working with unstructured data in a relational database management system (RDBMS) such as SQL Server Management Studio presents significant limitations, as these systems are generally not designed to process and interpret free-form text effectively. There are, however, several possible approaches to interact with unstructured data:

1. Performing simple keyword searches (e.g., using LIKE queries), which can find exact terms but lack contextual understanding.
2. Using a text search engine capable of performing more advanced queries through approximate matching, synonyms, or full-text indexing.
3. Leveraging a Large Language Model (LLM), which can interpret the meaning of the text and answer complex queries accordingly.

Among these, the LLM-based approach provides the highest degree of flexibility and semantic understanding. However, it also introduces challenges such as choosing the most suitable model in terms of accuracy, cost, and performance.
Our initial idea was to search for publicly available APIs that could be integrated into Python scripts. However, most of these services require paid subscriptions, which are not fully justified given the relatively low complexity of our task.
For this reason, Ollama was chosen as a valid compromise. It is a lightweight and user-friendly framework that allows users to run Large Language Models (LLMs) locally on their machines.
Unlike cloud-based services (such as those provided by OpenAI or Google), Ollama runs models like LLaMA, Mistral, Gemma, and others directly on the user's hardware, ensuring greater privacy, local control over data, and no cost per query.
Moreover, Ollama offers a simple API interface, making it easy to integrate with Python scripts and data processing workflows for tasks such as question answering, summarization, and information extraction from unstructured text.
In the following sections, we will illustrate the complete workflow implemented to adopt this approach.

Dataset Upload

First, we load the CSV file into Python, then we drop the attribute **pdf**, as it is not useful for our analysis.

```
import pandas as pd
df=pd.read_csv("C:\\Users\\giovi\\Downloads\\finale.csv",delimiter=";")
df=df.drop("pdf",axis=1)
```

Query Ollama

A custom function (query ollama) is defined to send an HTTP request to the Ollama API. It takes a prompt as input and returns the model's response. This function handles possible connection or timeout errors.

```
def query_ollama(prompt, timeout=180):
try:
    response = requests.post(
        'http://localhost:11434/api/generate',
        json={
            "model": "gemma:2b",
            "prompt": prompt,
```

```
                "stream": False
            },
            timeout=timeout
        )
        return response.json()["response"]
    except Exception as e:
        return f"ERRORE: {type(e).__name__} - {str(e)}"
```

One major challenge was selecting the most suitable LLM model among the many available in Ollama. Several models were too computationally expensive to run efficiently on our hardware. Among the usable ones, we tested Mistral, DeepSeek-R1, and Gemma:2b.
After several trials, Gemma:2b proved to be the best compromise between response quality and execution time. In particular, DeepSeek-R1 was significantly slower because it tends to generate internal reasoning before delivering the actual answer, increasing the overall processing time. Mistral was comparatively faster, but when considering both speed and response quality, Gemma:2b was clearly the most efficient and practical choice for our use case.

Construction of the Prompt

Another function (process row) is defined to build the prompt for a specific row of the dataset and send it to the LLM using the previous function. It also prints a short preview of the result and returns both the index and the generated response.

```
    def process_row(index, testo):
    prompt = f"""
Il seguente testo    stato estratto da un database:
{testo}
Rispondi alla domanda: "Su cosa si basa la dimostrazione dell'utilit  del farmaco?"
Risposta concisa e strutturata:
"""
    risposta = query_ollama(prompt)
    print(f"    Riga {index}: {risposta[:80]}...")
    return index, risposta
```

Saving the Output

It filters the rows where the column is not empty, in these case 4.1. Then, it creates a new empty column called risposta llm, which will be used to store the responses generated by the model.

```
    df = df[df["4.1"].notnull() & (df["4.1"].str.strip() != "")]
    df = df.copy()
    df["risposta_llm"] = ""
```

Batch Division

To avoid overloading the system and to manage large datasets efficiently, the data is processed in small groups called batches. Each batch contains a fixed number of rows (defined by batch size) and is prepared for parallel processing.

```
    batch_size = 10
    max_workers = 2
    for batch_start in range(0, len(df), batch_size):
    batch_end = min(batch_start + batch_size, len(df))
    batch_data = [(i, df.iloc[i]["4.1"]) for i in range(batch_start, batch_end)]
```

Parallel Execution

For each batch, the rows are processed in parallel using Python's ThreadPoolExecutor. This allows multiple requests to be sent to the LLM (Ollama) at the same time, significantly speeding up the processing. Each result is saved in the corresponding row of the DataFrame.

```
        with ThreadPoolExecutor(max_workers=max_workers) as executor:
        futures = {executor.submit(process_row, i, testo): i for i, testo in
            batch_data}
        for future in as_completed(futures):
            i = futures[future]
            try:
                index, result = future.result()
                df.iloc[index, df.columns.get_loc("risposta_llm")] = result
            except Exception as e:
                print(f"Errore elaborazione riga {i}: {e}")
                df.iloc[index, df.columns.get_loc("risposta_llm")] = "ERRORE"
```

Save

Finally, we save the output in a csv file.

```
    df.to_csv("risultati_con_risposte.csv", index=False)
```

New Implementation

This initial implementation was functional, but not very efficient. Completing a full query could take between one hour and one hour and a half, depending on the dataset size and system performance.
The goal then became to optimize the code for better efficiency. In the original version, the LLM was tasked with querying every single row of a specific column, which significantly increased processing time.
The idea was to introduce a pre-filtering step in the code. For example, if we want to answer the question "What is the demonstration of utility based on for a specific drug?", it's inefficient to run the LLM on the entire dataset. Instead, it is much more effective to filter the rows related to the specific drug first, and then run the LLM only on that subset. This significantly reduces computation time and improves the overall responsiveness of the system.

Code Changes

```
import requests
import pandas as pd
def query_ollama(prompt, timeout=180):
    try:
        response = requests.post(
            'http://localhost:11434/api/generate',
            json={
                "model": "gemma:2b",
                "prompt": prompt,
                "stream": False
            },
            timeout=timeout
        )
        return response.json()["response"]
    except Exception as e:
        return f"ERRORE: {type(e).__name__} - {str(e)}"
def interroga_righe_con_parola(parola, df, domanda, colonna_cerca="Denominazione e
    Confezione", colonna_testo="4.1"):
    # Filtro righe che contengono la parola (case-insensitive)
    filtro = df[colonna_cerca].str.contains(parola, case=False, na=False)
    df_filtrato = df[filtro].copy()
    if df_filtrato.empty:
        print(f"    Nessuna riga trovata contenente '{parola}' nella colonna '{
            colonna_cerca}'")
        return pd.DataFrame()
    # Nome della colonna per la risposta = domanda
    nome_colonna_risposta = domanda.strip().replace("?", "").strip()
    df_filtrato[nome_colonna_risposta] = ""
    for idx, row in df_filtrato.iterrows():
```

```
        testo = row[colonna_testo]
        prompt = f"""
Il seguente testo    stato estratto da un database:
{testo}
Rispondi alla domanda: "{domanda}"
Risposta concisa e strutturata:
"""
        risposta = query_ollama(prompt)
        df_filtrato.at[idx, nome_colonna_risposta] = risposta
        # Stampa a video
        print(f"Riga {idx}:")
        print(row.iloc[:8].to_string())
        print(f"{nome_colonna_risposta}: {risposta}\n{'-'*40}")
    # Seleziona solo le prime 8 colonne + colonna_testo + risposta
    colonne_output = list(df.columns[:8]) + [colonna_testo, nome_colonna_risposta]
    return df_filtrato[colonne_output]
if __name__ == "__main__":
    parola_cercata = input("Inserisci la parola da cercare in 'Denominazione e
        Confezione': ").strip()
    domanda = "Su cosa si basa la dimostrazione dell'utilit  del farmaco?"
    df_risultati = interroga_righe_con_parola(parola_cercata, df, domanda)
    if not df_risultati.empty:
        df_risultati.to_csv("risultati_filtrati.csv", index=False)
```

Compared to the previous code, the new version filters the dataset instead of processing it entirely or in fixed batches. Specifically, it selects only the rows where a user-specified keyword appears in the "Denominazione e Confezione" column. The filtering is case-insensitive, allowing more flexible and comprehensive matching.

```
filtro = df[colonna_cerca].str.contains(parola, case=False, na=False)
df_filtrato = df[filtro].copy()
```

The Large Language Model (LLM) is then queried exclusively for these filtered rows, reducing unnecessary API calls and focusing computational resources on relevant data.

```
for idx, row in df_filtrato.iterrows():
testo = row[colonna_testo]
prompt = f"""
Il seguente testo    stato estratto da un database:
{testo}
Rispondi alla domanda: "{domanda}"
Risposta concisa e strutturata:
"""
risposta = query_ollama(prompt)
df_filtrato.at[idx, nome_colonna_risposta] = risposta
```

For each matched row, the code prints the first eight columns to provide context, followed by the generated LLM response. The function returns a DataFrame containing only these first eight columns, the original text column (here "4.1"), and the newly created answer column.

```
nome_colonna_risposta = domanda.strip().replace("?", "").strip()
df_filtrato[nome_colonna_risposta] = ""
print(f"Riga {idx}:")
print(row.iloc[:8].to_string())
print(f"{nome_colonna_risposta}: {risposta}\n{'-'*40}")
colonne_output = list(df.columns[:8]) + [colonna_testo, nome_colonna_risposta]
return df_filtrato[colonne_output]
```

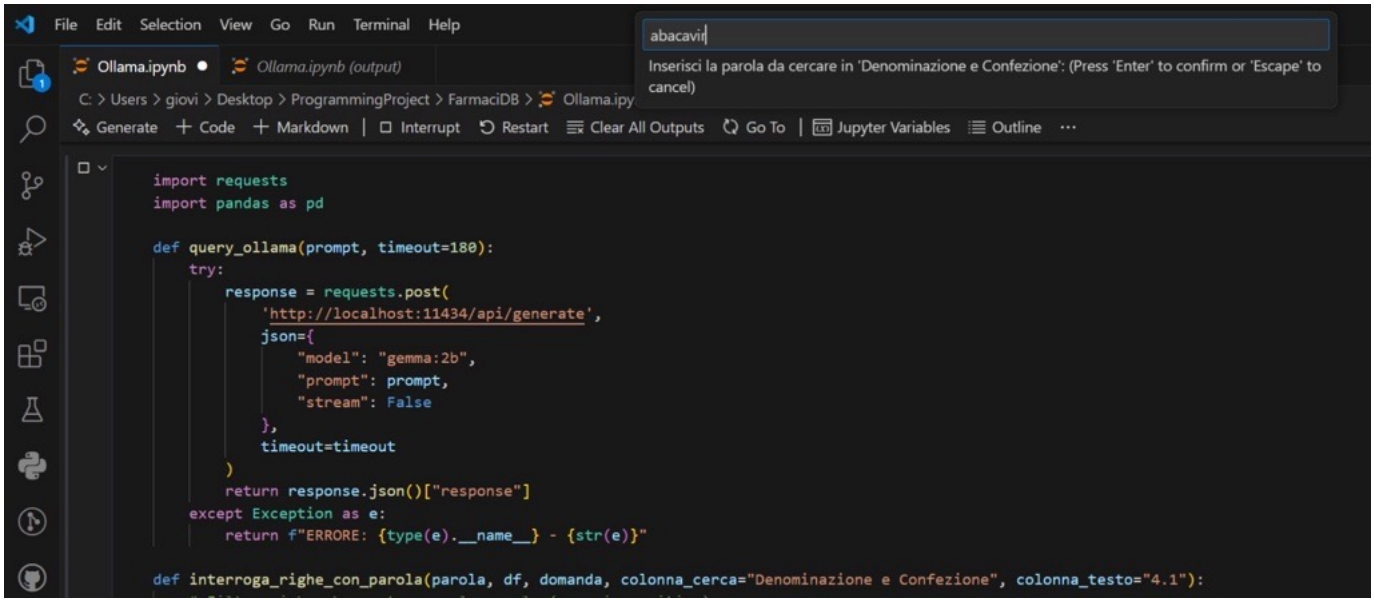
The script prompts the user to input the keyword to search for, making the process more interactive and adaptable to different queries without modifying the code itself.

```
parola_cercata = input("Inserisci la parola da cercare in 'Denominazione e Confezione
    ': ").strip()
domanda = "Su cosa si basa la dimostrazione dell'utilit  del farmaco?"
df_risultati = interroga_righe_con_parola(parola_cercata, df, domanda)
```

Finally, this approach is synchronous and linear (no multithreading), which simplifies debugging and makes it more suitable for targeted queries rather than bulk processing.

Example

When we run the script (with the filter set on the column "Denominazione e Confezione"), Python prompts us to enter the name of the drug we want to use to filter the database.



As a test, we input the name "abacavir". The script then filters the rows containing this keyword in the selected column and processes only those rows. It extracts the content from column "4.1" (also specified in the code) and asks the LLM the question:**"What is the basis for demonstrating the usefulness of the drug?"** Finally, the answers are saved in a CSV file for further analysis.

```
report=pd.read_csv("C:/Users/giovi/Desktop/ProgrammingProject/FarmaciDB/
    risultati_filtrati.csv")
pd.set_option('display.max_columns', 100)
pd.set_option('display.width', 200)
pd.set_option('display.max_colwidth', None)  # mostra testo completo senza puntini
pd.set_option('display.expand_frame_repr', False)
report
```

	Principio Attivo	Descrizione Gruppo	Denominazione e Confezione	Titolare AIC	Codice AIC	Codice Gruppo Equivalenza	classe	Codice ATC	4.1	Su cosa si basa la dimostrazione dell'utilità del farmaco
0	Abacavir	ABACAVIR 300MG 60 UNITA' USO ORALE	ABACAVIR*60 cpr riv 300 mg	MYLAN SpA	45354040	AAC	H	J05AF06 - ABACAVIR	Abacavir Mylan è indicato nella terapia di associazione antiretrovirale per il trattamento \ndell'infezione da Virus dell'Immunodeficienza Umana (HIV) negli adulti, negli adolescenti e nei \nbambini (vedere paragrafi 4.4 e 5.1). \n \nLa dimostrazione del beneficio di abacavir è principalmente basata sui risultati degli studi \neffettuati con un regime posologico di due volte al giorno, in pazienti adulti mai sottoposti a \ntrattamento (naïve) in terapia di associazione (vedere paragrafo 5.1). \n \nPrima di iniziare il trattamento con abacavir, deve essere eseguito uno screening per \ndeterminare la presenza dell'allele HLA-B*5701 in ogni paziente affetto da HIV, a prescindere \ndalla razza (vedere paragrafo 4.4). Abacavir non deve essere utilizzato nei pazienti in cui sia \nnota la presenza dell'allele \nHLA-B*5701.	La dimostrazione dell'utilità del farmaco basa sui risultati degli studi effettuati con un regime posologico di due volte al giorno, in pazienti adulti mai sottoposti a trattamento (naïve) in terapia di associazione (vedere paragrafo 5.1).
1	Abacavir/lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	AUROBINDO PHARMA ITALIA Srl	44669012	AAD	H	J05AR02 - LAMIVUDINA E ABACAVIR	Abacavir e lamivudina Aurobindo è indicato nella terapia di combinazione antiretrovirale per il \ntrattamento di adulti, adolescenti e bambini che pesano almeno 25 kg con infezione da Virus \ndell'Immunodeficienza Umana (Human Immunodeficiency Virus, HIV) (vedere paragrafi 4.4 e 5.1). \n \nPrima di iniziare il trattamento con abacavir, deve essere eseguito uno screening per la presenza \ndell'allele HLA-B*5701 in ogni paziente affetto da HIV, a prescindere dalla razza (vedere paragrafo \n4.4). Abacavir non deve essere utilizzato nei pazienti in cui sia nota la presenza dell'allele HLA-\nB*5701.	La dimostrazione dell'utilità del farmaco è fornita dal testo che descrive la terapia di combinazione antiretrovirale per il trattamento di adulti, adolescenti e bambini che pesano almeno 25 kg con infezione da Virus dell'Immunodeficienza Umana (Human Immunodeficiency Virus, HIV).
2	Abacavir/Lamivudina	ABACAVIR+LAMIVUDINA 600+300MG 30 UNITA' USO ORALE	ABACAVIR E LAMIVUDINA*30 cpr riv 600 mg + 300 mg	SUN PHARMACEUTICAL IND.EU.B.V.	45348036	AAD	H	J05AR02 - LAMIVUDINA E ABACAVIR	NOT AVAILABLE	Non ho accesso il testo, quindi non posso fornire una risposta alla tua domanda.