
NetMeds
OFFICIAL
NetMeds
NetMeds
NetMeds
NetMeds
NetMeds

End To End Netmeds Data Analysis Project Workflow

Web Scraping

Extracted data from the **Netmeds website** and **RxList** drug-related data using **BeautifulSoup** to collect drug-related information.

Excel Dashboard:

Cleaned data using **Power Query** and built an **interactive dashboard** for insights. Conducted a **case study** to suggest strategies for increasing **Netmeds' revenue**.

Sql Analysis

Transferred cleaned data from **Excel to SQL**, wrote **advanced queries** to extract insights, created **visualizations**, and provided **recommendations** to improve business performance.

EDA & Machine Learning

Conducted **Exploratory Data Analysis (EDA)** and built **ML models** using Python to **predict tablet prices** from **Netmeds data** and **drug class** from **RxList data**..

Table of Contents

**Automated Drug Classification with Web Scraping and Machine Learning

In this project, I aimed to predict the drug class of pharmaceutical drugs using a combination of web scraping and machine learning. The data for this project was extracted from the RXList website, which provides detailed information about drugs, including their brand names, generic names, and drug classes. By scraping this data and leveraging machine learning algorithms, I developed a model capable of classifying new drugs based on their names.

The objectives of the project were:

- To scrape and extract drug information from the RXList website.
- To preprocess the data and train a machine learning model.
- To evaluate the model's performance and make predictions.
- To save the trained model, vectorizer, and encoder for future predictions.


```
#importing all necessary libraries
from bs4 import BeautifulSoup
import requests
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
import joblib


alphabets = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
final_url = []

for i in alphabets:
```

```
    url = f"https://www.rxlist.com/drugs/alpha_{i}.htm"
    final_url.append(url)

print(final_url)
```

```
['https://www.rxlist.com/drugs/alpha_a.htm', 'https://www.rxlist.com/drugs/alpha_b.htm', 'https://www.rxlist.com/drugs/alpha_c.htm', 'https://www.rxlist.com/
```

```
links = []
for i in final_url:
    url = i
    page = requests.get(url)
    soup = BeautifulSoup(page.content, 'html.parser')

    div_elements = soup.find_all('div', attrs={"class": "AZ_results"})

    for div in div_elements:
        a_tags = div.find_all('a')
        for a_tag in a_tags:
            links.append(a_tag.get('href'))



Final_link_list = []

for i in links:
    if i and i.startswith("https"):
        Final_link_list.append(i)

print(Final_link_list)
print(len(Final_link_list))
```

```
['https://www.rxlist.com/a-methapred-drug.htm', 'https://www.rxlist.com/kivexa-drug.htm', 'https://www.rxlist.com/ziagen-drug.htm', 'https://www.rxlist.com/e
8797
```

```
for link in Final_link_list:
    print(link)
```

```
Streaming output truncated to the last 5000 lines.
https://www.rxlist.com/boniva-drug.htm
https://www.rxlist.com/boniva-injection-drug.htm
https://www.rxlist.com/ibrance-drug.htm
https://www.rxlist.com/brexafemme-drug.htm
```

https://www.rxlist.com/zevalin-drug.htm
https://www.rxlist.com/imbruvica-drug.htm
https://www.rxlist.com/ibsrela-drug.htm
https://www.rxlist.com/ibuprofen-drug.htm
https://www.rxlist.com/duexis-drug.htm
https://www.rxlist.com/caldolor-drug.htm
https://www.rxlist.com/neoprofen-drug.htm
https://www.rxlist.com/corvert-drug.htm
https://www.rxlist.com/ic-green-drug.htm
https://www.rxlist.com/firazyr-drug.htm
https://www.rxlist.com/sajazir-drug.htm
https://www.rxlist.com/iclevia-drug.htm
https://www.rxlist.com/iclusig-drug.htm
https://www.rxlist.com/extraneal-drug.htm
https://www.rxlist.com/vascepa-drug.htm
https://www.rxlist.com/idacio-drug.htm
https://www.rxlist.com/idamycin-drug.htm
https://www.rxlist.com/idamycin-pfs-drug.htm
https://www.rxlist.com/idamycin-drug.htm
https://www.rxlist.com/idamycin-pfs-drug.htm
https://www.rxlist.com/praxbind-drug.htm
https://www.rxlist.com/abecma-drug.htm
https://www.rxlist.com/zydelig-drug.htm
https://www.rxlist.com/idelvion-drug.htm
https://www.rxlist.com/idhifa-drug.htm
https://www.rxlist.com/idose-tr-drug.htm
https://www.rxlist.com/elaprase-drug.htm
https://www.rxlist.com/ifex-drug.htm
https://www.rxlist.com/ifex-drug.htm
https://www.rxlist.com/igalmi-drug.htm
https://www.rxlist.com/iheezo-drug.htm
https://www.rxlist.com/ilaris-drug.htm
https://www.rxlist.com/ilevro-drug.htm
https://www.rxlist.com/illuccix-drug.htm
https://www.rxlist.com/fanapt-drug.htm
https://www.rxlist.com/ventavis-drug.htm
https://www.rxlist.com/aurlumyn-drug.htm
https://www.rxlist.com/ilotycin-drug.htm
https://www.rxlist.com/ilumya-drug.htm
https://www.rxlist.com/iluvien-drug.htm
https://www.rxlist.com/gleevec-drug.htm
https://www.rxlist.com/imbruvica-drug.htm
https://www.rxlist.com/imcivree-drug.htm
https://www.rxlist.com/imdelltra-drug.htm
https://www.rxlist.com/imdur-drug.htm
https://www.rxlist.com/rytelo-drug.htm
https://www.rxlist.com/imfinzi-drug.htm
https://www.rxlist.com/cerezyme-drug.htm
https://www.rxlist.com/primaxin-im-drug.htm
https://www.rxlist.com/primaxin-iv-drug.htm
https://www.rxlist.com/recarbrio-drug.htm
https://www.rxlist.com/tofranil-drug.htm

https://www.rxlist.com/tofranil-pm-drug.htm

```
drug_info = {}

url = 'https://www.rxlist.com/boniva-drug.htm#description'
page = requests.get(url)

if page.status_code != 200: #  if the page request was successfull or not
    print(f"Failed to retrieve {url}")
else:
    soup = BeautifulSoup(page.content, 'html.parser')

    # Extracting the Brand Name from the <h1> tag as brand name is in h1 tag
    h1_tag = soup.find('h1')
    if h1_tag:
        brand_name = h1_tag.get_text(strip=True)
        drug_info['Brand Name'] = brand_name
        print(f"Brand Name: {brand_name}")

    # as the name of medicine is in  the <div> tags with class 'hero'
    div_elements = soup.find_all('div', attrs={"class": "hero"})

    for div in div_elements:
        # Finding the <ul> inside the <div class="hero"> as inside each hero theres is list which contains genric name,drug class,brand name
        ul_tag = div.find('ul')
        if ul_tag:
            # for all <li> tags inside the <ul>
            li_tags = ul_tag.find_all('li')

            for li in li_tags:
                # Extracting the Generic Name if not there i am giving as none
                if 'Generic Name' in li.get_text():
                    generic_name = li.find('a').get_text(strip=True) if li.find('a') else None
                    drug_info['Generic Name'] = generic_name
                    print(f"Generic Name: {generic_name}")

                # Extract the Drug Classes
                if 'Drug Class' in li.get_text():
                    a_tags = li.find_all('a')
                    drug_classes = [a.get_text(strip=True) for a in a_tags]
                    drug_info['Drug Classes'] = drug_classes
                    print(f"Drug Classes: {', '.join(drug_classes)}")
                    #adding each info to drug_info

print("\nDrug Info Dictionary:")
```

```
print(drug_info)
#this was for a single medicine to check the code now  for multiple link we need to extract
```

```
Brand Name: Boniva
Generic Name: ibandronate sodium
Drug Classes: Bisphosphonate Derivatives, Calcium Metabolism Modifiers

Drug Info Dictionary:
{'Brand Name': 'Boniva', 'Generic Name': 'ibandronate sodium', 'Drug Classes': ['Bisphosphonate Derivatives', 'Calcium Metabolism Modifiers']}
```

```
# Initialize the list to store drug information for all URLs
final_drug_info_list = []

for url in Final_link_list:
    drug_info = {}

    # Sending a request to the URL
    page = requests.get(url)

    # Checking if the page request was successful (status code 200)
    if page.status_code != 200:
        print(f"Failed to retrieve {url}")
        continue  # if the page is not responding then continue

    soup = BeautifulSoup(page.content, 'html.parser')

    # Extractng the Brand Name from the <h1> tag
    h1_tag = soup.find('h1')
    if h1_tag:
        brand_name = h1_tag.get_text(strip=True)
        drug_info['Brand Name'] = brand_name
        print(f"Brand Name: {brand_name}")

    # Finding all the <div> tags with class 'hero' because it contains the ifo about the medicine
    div_elements = soup.find_all('div', attrs={"class": "hero"})

    for div in div_elements:# Finding the <ul> inside the <div class="hero">
        ul_tag = div.find('ul')
        if ul_tag:
            li_tags = ul_tag.find_all('li')

            for li in li_tags:
                if 'Generic Name' in li.get_text():
                    generic_name = li.find('a').get_text(strip=True) if li.find('a') else None# Extracting the Generic Name
                    drug_info['Generic Name'] = generic_name
```

```
                print(f"Generic Name: {generic_name}")

            if 'Drug Class' in li.get_text():
                a_tags = li.find_all('a')
                drug_classes = [a.get_text(strip=True) for a in a_tags]# Extracting the Drug Classes
                drug_info['Drug Classes'] = drug_classes
                print(f"Drug Classes: {', '.join(drug_classes)}")

    # appending the drug information for the current URL to the final list
    final_drug_info_list.append(drug_info)

# final_list
print("\nFinal Drug Info List:")
for info in final_drug_info_list:
    print(info)
```

```
    {'Brand Name': 'Gamunex', 'Generic Name': 'immune globulin intravenous (human) 10%', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Octagam', 'Generic Name': 'immune globulin intravenous (human) 5% liquid preparation', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Carimune', 'Generic Name': 'immune globulin intravenous (human) nanofiltered lyophilized preparation', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Rhophylac', 'Generic Name': 'immune globulin intravenous (human) solution', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Bivigam', 'Generic Name': 'immune globulin intravenous (human), 10%', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Gammaplex', 'Generic Name': 'immune globulin intravenous (human), 5% liquid', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Cutaquig', 'Generic Name': 'immune globulin intravenous, human', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Panzyga', 'Generic Name': 'immune globulin intravenous, human - ifas liquid preparation', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Asceniv', 'Generic Name': 'immune globulin intravenous, human - slra for injection', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Yimmugo', 'Generic Name': 'immune globulin intravenous, human-dira, 10% liquid', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Alyglo', 'Generic Name': 'immune globulin intravenous, human-stwk 10% liquid', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Vivaglobin', 'Generic Name': 'immune globulin subcutaneous (human)', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Hizentra', 'Generic Name': 'immune globulin subcutaneous (human) injection', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Cuvitru', 'Generic Name': 'immune globulin subcutaneous (human), 20% solution', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Xembify', 'Generic Name': 'immune globulin subcutaneous, human – klhw injection', 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Imodium', 'Generic Name': 'loperamide hcl', 'Drug Classes': ['Antidiarrheals']}
    {'Brand Name': 'Imogam Rabies', 'Generic Name': None, 'Drug Classes': ['Immune Globulins']}
    {'Brand Name': 'Imovax', 'Generic Name': 'rabies vaccine', 'Drug Classes': ['Vaccines, Inactivated, Viral', 'Vaccines, Travel']}
    {'Brand Name': 'Impavido', 'Generic Name': 'miltefosine capsules', 'Drug Classes': ['Antileishmaniasis Agents']}
    {'Brand Name': 'Impeklo', 'Generic Name': 'clobetasol propionate lotion', 'Drug Classes': ['Corticosteroids, Topical']}
    {'Brand Name': 'Implanon', 'Generic Name': 'etonogestrel implant', 'Drug Classes': ['Progestins']}
    {'Brand Name': 'Impoyz', 'Generic Name': 'clobetasol propionate cream', 'Drug Classes': ['Corticosteroids, Topical']}
    {'Brand Name': 'Imraldi', 'Generic Name': None}
```

```
import pandas as pd
#importing libraries


drug_info_df = pd.DataFrame([{
    'Brand Name': info.get('Brand Name', ''),
    'Generic Name': info.get('Generic Name', ''),
    'Drug Classes': ', '.join(info.get('Drug Classes', []))  # Use an empty list if 'Drug Classes' is missing
} for info in final_drug_info_list])


drug_info_df
```

	Brand Name	Generic Name	Drug Classes
0	A-Methapred	None	Corticosteroids
1	Kivexa	None	
2	Ziagen	abacavir sulfate	HIV, NNRTIs, Antiretroviral Agents
3	Epzicom	abacavir sulfate and lamivudine tablets	HIV, NNRTIs
4	Trizivir	None	HIV, NNRTIs
...	...	...	...
8792	Zyprexa Relprevv	olanzapine extended release injectable suspension	How Do Second Generation Antipsychotics Work?,...
8793	Zyrtec	cetirizine	How Do Second Generation Antihistamines Work?,...
8794	Zyrtec-D	cetirizine, pseudoephedrine	Antihistamine/Decongestant Combos
8795	Zytiga	None	Antineoplastics, Antiandrogen, Antiandrogens
8796	Zyvox	linezolid	Antibiotics, Other, Oxazolidinones

8797 rows × 3 columns

```
# Saving the DataFrame to a CSV file
drug_info_df.to_csv('drug_info.csv', index=False)  # index=False avoids saving the index as a separate column
```

```
drug_info_df.describe(include=['object', 'category'])
```

	Brand Name	Generic Name	Drug Classes
count	8797	7019	8797
unique	4467	3164	1070
top	A-Methapred	levonorgestrel and ethinyl estradiol tablets	
freq	2	32	1081

```
drug_info_df.isnull().sum()
```

	0
Brand Name	0
Generic Name	1778
Drug Classes	0

dtype: int64

```
print(drug_info_df.isnull().sum())
#there were 1778 null values means there are no generic name for drug mentioned in rx list
```

```
Brand Name         0
Generic Name    1778
Drug Classes       0
dtype: int64
```

```
drug_info_df_cleaned = drug_info_df.dropna()
print(drug_info_df_cleaned.isnull().sum())
```

```
Brand Name      0
Generic Name    0
Drug Classes    0
dtype: int64
```

```
pd.set_option('display.max_rows', None)
```

```
drug_info_df_cleaned.describe(include='object')
```

	Brand Name	Generic Name	Drug Classes
count	7019	7019	7019
unique	3548	3164	1011
top	Ziagen	levonorgestrel and ethinyl estradiol tablets	
freq	2	32	185

```
drug_info_df_cleaned
```

	Brand Name	Generic Name	Drug Classes
2	Ziagen	abacavir sulfate	HIV, NNRTIs, Antiretroviral Agents
3	Epzicom	abacavir sulfate and lamivudine tablets	HIV, NNRTIs
5	Triumeq	abacavir, dolutegravir, and lamivudine film-co...	HIV, ART Combos
6	Tymlos	abaloparatide injection	Parathyroid Hormone Analogs
9	Orencia	abatacept	DMARDs, Immunomodulators
10	ReoPro	abciximab	Antineoplastics CDK Inhibitors, Glycoprotein I...
11	Abecma	idecabtagene vicleucel suspension	CAR-T Cell Therapies
12	Abelcet	amphotericin b injection	Antifungals, Systemic
13	Verzenio	abemaciclib tablets	Antineoplastics CDK Inhibitors
14	Abilify	aripiprazole	Antipsychotics, Second Generation, Antimanic A...
15	Abilify Asimtufii	aripiprazole for extended-release injectable s...	Antipsychotics, Second Generation, Antimanic A...
16	Abilify Maintena	aripiprazole extended-release injectable suspe...	Antipsychotics, Second Generation, Antimanic A...
17	Abilify MyCite	aripiprazole tablets with sensor	Antipsychotics, Second Generation, Antimanic A...
19	Yonsa	abiraterone acetate tablets	Antineoplastics, Antiandrogen, Antiandrogens
22	Dysport	abobotulinumtoxin a injection	Neuromuscular Blockers, Depolarizing, Botulinu...
23	Abraxane	albumin-bound paclitaxel for injectable suspen...	Antineoplastics, Antimicrotubular
24	Abreva	docosanol cream	Antivirals, Topical
25	Abrilada	adalimumab-afzb injection, for subcutaneous use	DMARDs, TNF Inhibitors, Monoclonal Antibodies
26	Cibinqo	abrocitinib tablets	Dermatologics, Other, DMARDs, JAK Inhibitors
27	Abrysvo	respiratory syncytial virus vaccine for injection	Vaccines, Inactivated, Viral
33	Absorica	isotretinoin	Acne Agents, Systemic
34	Absorica LD	isotretinoin capsules	Retinoid-like Agents, Topical, Acne Agents, To...
35	Abstral	fentanyl sublingual tablets	Opioid Analgesics, Opioids, Anilidopiperidine
36	Calquence	acalabrutinib capsules	Antineoplastic Tyrosine Kinase Inhibitors
38	Campral	acamprosate calcium	Psychiatry Agents Other, GABA Analogs
39	Acanya	clindamycin phosphate 1.2% and benzoyl peroxid...	Acne Agents, Topical Combos
40	Precose	acarbose	Antidiabetics, Alpha-Glucosidase Inhibitors
41	Accolate	zafirlukast	Leukotriene Receptor Antagonists

41	Accolate	zafirlukast	Leukotriene Receptor Antagonists
42	Accretropin	somatropin injection	Growth Hormone Analogs
43	Accrufer	ferric maltol capsules	Iron Products
44	AccuNeb	albuterol sulfate inhalation solution	Beta2 Agonists
45	Accupril	quinapril hydrochloride	ACE Inhibitors
46	Accuretic	quinapril hcl/hydrochlorothiazide	ACEIHCTZ Combos
47	Accutane	isotretinoin	Acne Agents, Topical
49	Sectral	acebutolol	Antidysrhythmics, II, Beta-Blockers, Intrinsic...
51	Aceon	perindopril erbumine	ACE Inhibitors
52	Acephen	acephen	Analgesics, Other
53	Acetadote	acetylcysteine injection	Antidotes, Other
54	Tylenol	acetaminophen	Other analgesics, acetaminophen
55	Tylenol-Codeine	acetaminophen and codeine	Analgesics, Opioid Combos
56	Combogesic IV	acetaminophen and ibuprofen injection	Analgesics, Other Combos
57	Combogesic	acetaminophen and ibuprofen tablets	Analgesics, Other Combos
58	Ofirmev	acetaminophen for injection	Analgesics, Other
59	Acephen	acephen	Analgesics, Other
60	Trezix	acetaminophen, caffeine and dihydrocodeine bit...	Analgesics, Opioid Combos
61	Midrin	acetaminophen, isometheptene and dichloralphen...	Analgesics, Other Combos
64	Diamox Tablets and Injection	acetazolamide tablets and injection	Anticonvulsants, Other, Antiglaucoma, Carbonic...
65	Diamox Sequels	acetazolamide xr	Anticonvulsants, Other
66	Acetic Acid	acetic acid	Antibacterials, Topical, Antifungals, Topical
67	Lithostat	acetohydroxamic acid tablets	Antimicrobials, Adjunct
69	Miochol-E	acetylcholine chloride intraocular solution	Miotics, Direct-Acting
71	Legubeti	acetylcysteine for oral solution	Enzymes, Mucolytic
72	Acetadote	acetylcysteine injection	Antidotes, Other
73	Acetylcysteine Solution	n-acetyl-l-cysteine	Pulmonary, Other
75	Achromycin V	tetracycline	Tetracyclines
77	Acidul	fluoride	Minerals, Other

8738	Zonatuss	benzonatate capsules, usp 150 mg	Antitussives
8739	Zonegran	zonisamide	Anticonvulsants, Other
8740	Zonisade	zonisamide oral suspension	Anticonvulsants, Other
8741	Zonegran	zonisamide	Anticonvulsants, Other
8742	Zonisade	zonisamide oral suspension	Anticonvulsants, Other
8743	Zontivity	vorapaxar tablets	, Protease Activated Receptor-1 (PAR-1) Inhibi...
8744	Zorbtive	somatropin rdna origin for injection	Growth Hormone Analogs
8745	Zortress	everolimus	
8746	Zorvolex	diclofenac capsules	
8747	Zoryve	roflumilast cream	Phosphodiesterase-4 Enzyme Inhibitors
8748	Zoryve Foam	roflumilast topical foam	PDE-4 Inhibitors, Topical
8751	Shingrix	zoster vaccine recombinant, adjuvanted suspens...	Vaccines, Inactivated, Viral
8752	Zosyn	piperacillin and tazobactam injection	Penicillins, Extended-Spectrum
8753	Zosyn Injection	piperacillin and tazobactam pharmacy bulk vial	Penicillins, Extended-Spectrum
8755	Zovia	etynodiol diacetate and ethinyl estradiol tablets	Contraceptives, Oral, Estrogens/Progestins
8756	Zovirax	acyclovir	Antivirals, VZV, Antivirals, HSV, Antivirals, ...
8757	Zovirax Cream	acyclovir cream, 5%	Antivirals, HSV, Antivirals, Other
8758	Zovirax Injection	acyclovir for injection	Antivirals, Other
8759	Zovirax Ointment	acyclovir ointment	Antivirals, Topical
8760	Zovirax Suspension	acyclovir	Antivirals, Other
8761	Ztalmy	ganaxolone oral suspension	GABA Analogs
8762	ZTLido	lidocaine	Local Anesthetics, Amides
8763	Zubsolv	buprenorphine and naloxone sublingual tablets	Opioid Antagonists
8764	Zulresso	brexanolone injection, for intravenous use	Antidepressants, Other
8765	Zunveyl	benzgalantamine delayed-release tablets	Acetylcholinesterase Inhibitors, Central
8766	Zuplenz	ondansetron oral soluble film	Antiemetics, Selective 5-HT3 Antagonist
8768	Zurzuvae	zuranolone	Antidepressants, Other
8769	Zurnai	nalmefene for intramuscular for subcutaneous use	Opioid Antagonists, Opioid Reversal Agents
8770	Zurzuvae	zuranolone	Antidepressants, Other

8771	Zutripro	hydrocodone bitartrate, chlorpheniramine malea...	Antitussives, Narcotic Combos
8772	Zyban	bupropion hcl	Antidepressants, Dopamine Reuptake Inhibitors
8773	Zyclara	imiquimod cream	Topical Skin Products
8774	Zydelig	idelalisib tablets	, Antineoplastics PI3K Inhibitors
8776	Zydone	hydrocodone bitartrate and acetaminophen	Opioid Analgesics
8777	Zyflo	zileutin	5-Lipoxygenase Inhibitors
8778	Zyflo CR	zileuton extended release tablets	5-Lipoxygenase Inhibitors
8779	Zyfrel	hydrocodone bitartrate and acetaminophen oral ...	Selective 5-HT3 Receptor Antagonists
8780	Zykadia	ceritinib hard-gelatin capsules	Antineoplastics, Anaplastic Lymphoma Kinase In...
8781	Zylet	loteprednol etabonate and tobramycin	Antibiotics/Corticosteroids, Ophthalmic
8782	Zyloprim	allopurinol	Antigout Agents, Xanthine Oxidase Inhibitors
8784	Zymaxid	gatifloxacin ophthalmic solution	Quinolones, Ophthalmic
8785	Zymfentra	infliximab-dyyb injection, for subcutaneous use	DMARDs, TNF Inhibitors, Monoclonal Antibodies
8786	Zynlonta	loncastuximab tesirine-lpyl for injection	Antineoplastics, Anti-CD19 Monoclonal Antibodies
8787	Zynrelef	bupivacaine and meloxicam	NSAIDs
8788	Zynteglo	betibeglogene autotemcel suspension for iv inf...	Clotting Factors, Gene Therapy
8789	Zynyz	retifanlimab-dlwr	PD-1PD-L1 Inhibitors
8790	Zypitamag	pitavastatin tablets for oral use	HMG-CoA Reductase Inhibitors, Lipid-Lowering A...
8791	Zyprexa	olanzapine	Antipsychotics, Second Generation, Antimanic A...
8792	Zyprexa Relprevv	olanzapine extended release injectable suspension	How Do Second Generation Antipsychotics Work?,...
8793	Zyrtec	cetirizine	How Do Second Generation Antihistamines Work?,...
8794	Zyrtec-D	cetirizine, pseudoephedrine	Antihistamine/Decongestant Combos

```
print((drug_info_df_cleaned['Drug Classes'] == '').sum())   # Empty strings
```

```
185
```

```
# Checking for NaN and empty string values in 'Brand Name', 'Generic Name', and 'Drug Classes'
print("Missing values in 'Brand Name':")
print(drug_info_df_cleaned['Brand Name'].isnull().sum())  # NaN values in 'Brand Name'
print((drug_info_df_cleaned['Brand Name'] == '').sum())   # Empty string values in 'Brand Name'

print("\nMissing values in 'Generic Name':")
print(drug_info_df_cleaned['Generic Name'].isnull().sum())  # NaN values in 'Generic Name'
print((drug_info_df_cleaned['Generic Name'] == '').sum())   # Empty string values in 'Generic Name'

print("\nMissing values in 'Drug Classes':")
print(drug_info_df_cleaned['Drug Classes'].isnull().sum())  # NaN values in 'Drug Classes'
print((drug_info_df_cleaned['Drug Classes'] == '').sum())   # Empty string values in 'Drug Classes')
```

```
Missing values in 'Brand Name':
0
0

Missing values in 'Generic Name':
0
0

Missing values in 'Drug Classes':
0
185
```

```
# Droping dowwn  rows where 'Drug Classes' is an empty string
drug_info_df_cleaned = drug_info_df_cleaned[drug_info_df_cleaned['Drug Classes'] != '']

# Verifying wether  that there are no empty strings left
print(f"Missing values in 'Drug Classes' after cleaning: {drug_info_df_cleaned['Drug Classes'].isnull().sum()}")
print(f"Empty string values in 'Drug Classes' after cleaning: {(drug_info_df_cleaned['Drug Classes'] == '').sum()}")
```

```
Missing values in 'Drug Classes' after cleaning: 0
Empty string values in 'Drug Classes' after cleaning: 0
```

```
#data is cleaned
```

```
drug_info_df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6834 entries, 2 to 8796
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Brand Name    6834 non-null   object
 1   Generic Name  6834 non-null   object
 2   Drug Classes  6834 non-null   object
dtypes: object(3)
memory usage: 213.6+ KB
```

```
drug_info_df_cleaned.shape
```

```
(6834, 3)
```

```
df = drug_info_df_cleaned
#our dataframe
#precheck for missing values
print("Missing values in the dataset:")
print(df.isnull().sum())

# Vectorizing text data (combine Brand Name and Generic Name)
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['Brand Name'] + ' ' + df['Generic Name'])

# Encoding target variable (Drug Classes)
encoder = LabelEncoder()
y = encoder.fit_transform(df['Drug Classes'])

# splitting data into train test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training Random Forest Classifier
model = RandomForestClassifier()
model.fit(X_train, y_train)

# making Prediction  on the test set
y_pred = model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

joblib.dump(model, 'drug_class_model.pkl')
joblib.dump(vectorizer, 'vectorizer.pkl')
```

```
joblib.dump(encoder, 'encoder.pkl')
# Saving the model, vectorizer, and encoder as .pkl files

print("Model, vectorizer, and encoder saved as .pkl files.")
```

```
Missing values in the dataset:
Brand Name      0
Generic Name    0
Drug Classes    0
dtype: int64
Model Accuracy: 86.91%
Model, vectorizer, and encoder saved as .pkl files.
```

```
# List of test inputs (Brand Name + Generic Name)
test_inputs = [
    "Panadol Paracetamol",
    "Advil Ibuprofen",
    "Lipitor Atorvastatin",
    "Nexium Esomeprazole",
    "Prozac Fluoxetine",
    "Zithromax Azithromycin",
    "Lasix Furosemide",
    "Humalog Insulin Lispro",
    'Mounjaro tirzepatide',
    'Otezla apremilast [a-PRE-mi-last]',
    'ZEXIPAG 200 Tablet 10 Selexipag 200 mcg',
    'Zandu Balm 9 ML Ajwain Trachyspermum (0.75 %) + Eucalyptus Oil (0.8 %) + Gaultheria Fragrantissima (12 %) + Pudina Satva (12 %)',
    "Glucophage Metformin",
    'A-doc sp aceclofenac,paracetamol and serraitiopeptidase',
    'Dettol Chloroxylenol',
    'Seromycin Cycloserine',
    'Pyrazinamide Pyrazinamide ',
    'Deltyba Delamanid',
    'Priligy Dapoxetine',
    'Cogniza Cerebroprotein Hydrolysate',
    'CILOBACT Cilostazol ',
    'profine Progesterone'
]

#checking qith various inputs
for test_input in test_inputs:
    # vectorizing the input data as that we did in train
    input_vectorized = vectorizer.transform([test_input])
    predicted_class_encoded = model.predict(input_vectorized)# Predicting the encoded drug class
    predicted_class = encoder.inverse_transform(predicted_class_encoded)
    print(f"Input: {test_input} -> Predicted Drug Class: {predicted_class[0]}")
```

Optimized Web Scraping for Medicine Data Extraction from Netmeds

This project focuses on efficiently scraping data from Netmeds, an online pharmacy, to extract vital information about medicines, including tablet names, prescription requirements, pricing, discounts, and other product details. Initially taking over 15 hours to execute, the project was optimized using multithreading, exception handling, and logging, reducing the execution time to just a few hours. The pipeline was designed to be reusable and scalable, enabling the extraction of data from similar websites. Over 27,000 medicines were successfully scraped, with structured data stored for further analysis on pricing, availability, and discounts.


```
#importing necessary libraries
import requests
from bs4 import BeautifulSoup
```

```
import time
from concurrent.futures import ThreadPoolExecutor, as_completed
import threading
import json
import logging


url = "https://www.netmeds.com/prescriptions"
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

# making a list to store links
all_links = []

# Iterate through all divs in the 'prescriptions_products' section
for section in soup.find_all("div", attrs={"class": "prescriptions_products"}): # Iterating through all 'ul' elements with the class 'alpha-drug-list'
    for ul in section.find_all("ul", attrs={"class": "alpha-drug-list"}):
        links = [a['href'] for a in ul.find_all("a", href=True)]  # Extracting all 'a' tags within each 'ul' and get their 'href' attributes
        all_links.extend(links)
# so there are categories like for each disease there is alphabetical drugs for each medicine so we need to go into each disease then each medicine and info relate


all_links
```

```
['https://www.netmeds.com/prescriptions/adhd',
 'https://www.netmeds.com/prescriptions/acne',
 'https://www.netmeds.com/prescriptions/alcohol-addiction',
 'https://www.netmeds.com/prescriptions/allergies',
 'https://www.netmeds.com/prescriptions/alzheimer',
 'https://www.netmeds.com/prescriptions/amoebiasis',
 'https://www.netmeds.com/prescriptions/anaemia',
 'https://www.netmeds.com/prescriptions/anaesthesia-local',
 'https://www.netmeds.com/prescriptions/anaesthesia-general',
 'https://www.netmeds.com/prescriptions/anal-fissure',
 'https://www.netmeds.com/prescriptions/angina',
 'https://www.netmeds.com/prescriptions/anti-allergic',
 'https://www.netmeds.com/prescriptions/anti-scar',
 'https://www.netmeds.com/prescriptions/antibiotic',
 'https://www.netmeds.com/prescriptions/anxiety',
 'https://www.netmeds.com/prescriptions/apnea',
 'https://www.netmeds.com/prescriptions/appetite',
 'https://www.netmeds.com/prescriptions/arrhythmiasis',
 'https://www.netmeds.com/prescriptions/arthritis',
 'https://www.netmeds.com/prescriptions/asthma-copd',
 'https://www.netmeds.com/prescriptions/atopic-dermatitis-eczema',
 'https://www.netmeds.com/prescriptions/auto-immune-disease',
 'https://www.netmeds.com/prescriptions/ayurvedic-medicine',
 'https://www.netmeds.com/prescriptions/bacterial-infections',
 'https://www.netmeds.com/prescriptions/bladder-and-prostate-disorders',
 'https://www.netmeds.com/prescriptions/bleeding-disorders',
```

```
        'https://www.netmeds.com/prescriptions/blood-clot',
        'https://www.netmeds.com/prescriptions/bone-metabolism',
        'https://www.netmeds.com/prescriptions/burn',
        'https://www.netmeds.com/prescriptions/cns-stimulant',
        'https://www.netmeds.com/prescriptions/cancer-oncology',
        'https://www.netmeds.com/prescriptions/cardiac-arrest',
        'https://www.netmeds.com/prescriptions/cerebral-ischemic-stroke',
        'https://www.netmeds.com/prescriptions/cholelithiasis-gall-stones',
        'https://www.netmeds.com/prescriptions/cleanser',
        'https://www.netmeds.com/prescriptions/colon-cleanser',
        'https://www.netmeds.com/prescriptions/constipation',
        'https://www.netmeds.com/prescriptions/contraception',
        'https://www.netmeds.com/prescriptions/cough-and-cold',
        'https://www.netmeds.com/prescriptions/crack',
        'https://www.netmeds.com/prescriptions/dandruff',
        'https://www.netmeds.com/prescriptions/denture-adhesive',
        'https://www.netmeds.com/prescriptions/depression',
        'https://www.netmeds.com/prescriptions/dewormer',
        'https://www.netmeds.com/prescriptions/diabetes',
        'https://www.netmeds.com/prescriptions/diagnostic',
        'https://www.netmeds.com/prescriptions/diarrhoea',
        'https://www.netmeds.com/prescriptions/dietary-management',
        'https://www.netmeds.com/prescriptions/digestion',
        'https://www.netmeds.com/prescriptions/dry-eye',
        'https://www.netmeds.com/prescriptions/dry-skin',
        'https://www.netmeds.com/prescriptions/ear-conditions',
        'https://www.netmeds.com/prescriptions/ectoparasiticide',
        'https://www.netmeds.com/prescriptions/electrolytes',
        'https://www.netmeds.com/prescriptions/epilepsy-convulsion',
        'https://www.netmeds.com/prescriptions/eye-infections',
        'https://www.netmeds.com/prescriptions/other-eye-conditions',
        'https://www.netmeds.com/prescriptions/fever',

final_link=[]#for storing now each link of specific medicine


headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
}
all_med_links = []

for url in all_links:
    print(f"Scraping URL: {url}")
    try:

        response = requests.get(url, headers=headers) # initializing GET request to the current URL
        if response.status_code == 200: # if request succesfully accepted
            soup = BeautifulSoup(response.text, "html.parser")
            med_links = []
```

```
            # extracting the main section containing all drugs
            for section in soup.find_all("div", attrs={"class": "prescriptions_products"}):
                for ul in section.find_all("ul"):
                    for li in ul.find_all("li", class_="product-item"):
                        # extracting nested <a> tag and extract the href attribute
                        a_tag = li.find("a", href=True)
                        if a_tag:
                            med_links.append(a_tag["href"])

            # Adding the medication links from this page to the main list
            all_med_links.extend(med_links)
            print(f"Found {len(med_links)} medication links on this page.")

        else:
            print(f"Failed to fetch {url}. Status code: {response.status_code}")

    except Exception as e:
        print(f"An error occurred while scraping {url}: {e}")
    time.sleep(2)    # Adding a delay time  between requests to avoid overwhelming the server


print(f"\nTotal medication links extracted: {len(all_med_links)}")
```

```
Found 142 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/liver-supplement
Found 0 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/lubrication
Found 0 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/malarial
Found 168 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/melasma
Found 10 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/migraine
Found 224 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/mucolytic-agent
Found 4 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/multiple-sclerosis
Found 16 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/muscle-cramps-spasticity
Found 408 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/muscle-spasm
Found 434 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/myasthenia-gravis
Found 14 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/mydriasis
Found 56 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/nappy-rash
Found 12 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/nasal-congestion
Found 368 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/neuropathic-pain
Found 1262 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/nootropics-and-neurotrophics
Found 330 medication links on this page.
Scraping URL: https://www.netmeds.com/prescriptions/obese
```

```
# Saving all the links in the 'all_med_links' list to a text file
with open('all_med_links.txt', 'w') as file:
    for link in all_med_links:
        file.write(link + '\n')

print("Links have been saved to 'all_med_links.txt'")
```

```
Links have been saved to 'all_med_links.txt'
```

```
all_med_links
```

```
['https://www.netmeds.com/prescriptions/atrest-25mg-tablet-10-s',
 'https://www.netmeds.com/prescriptions/atrest-25mg-tablet-10-s',
 'https://www.netmeds.com/prescriptions/capnea-injection-1ml',
 'https://www.netmeds.com/prescriptions/capnea-solution-1ml',
```

'https://www.netmeds.com/prescriptions/cognistar-30mg-injection',
'https://www.netmeds.com/prescriptions/cognistar-60mg-injection',
'https://www.netmeds.com/prescriptions/cogniza-tablet-10-s',
'https://www.netmeds.com/prescriptions/capnea-injection-1ml',
'https://www.netmeds.com/prescriptions/capnea-solution-1ml',
'https://www.netmeds.com/prescriptions/cognistar-30mg-injection',
'https://www.netmeds.com/prescriptions/cognistar-60mg-injection',
'https://www.netmeds.com/prescriptions/cogniza-tablet-10-s',
'https://www.netmeds.com/prescriptions/a-ret-0-025-gel-20gm',
'https://www.netmeds.com/prescriptions/a-ret-0-05-gel-20gm',
'https://www.netmeds.com/prescriptions/a-ret-0-1-gel-20gm',
'https://www.netmeds.com/prescriptions/ab-next-gel-20gm',
'https://www.netmeds.com/prescriptions/abaka-gel-10gm',
'https://www.netmeds.com/prescriptions/acnecross-anti-acne-body-spray-50ml',
'https://www.netmeds.com/prescriptions/acnecure-gel-20gm',
'https://www.netmeds.com/prescriptions/acnedap-gel-15gm',
'https://www.netmeds.com/prescriptions/acnedap-plus-gel-15gm',
'https://www.netmeds.com/prescriptions/acnejoy-face-wash-100ml',
'https://www.netmeds.com/prescriptions/acnelak-soap-75gm',
'https://www.netmeds.com/prescriptions/acnelex-medi-acne-cleanser-mask-face-wash-100ml',
'https://www.netmeds.com/prescriptions/acnemoist-cream-60gm',
'https://www.netmeds.com/prescriptions/acnepad-wipes-50-s',
'https://www.netmeds.com/prescriptions/acnerex-face-wash-75ml',
'https://www.netmeds.com/prescriptions/acnerex-soap-75gm',
'https://www.netmeds.com/prescriptions/acnescar-gel-15gm',
'https://www.netmeds.com/prescriptions/acnesol-1-solution-25ml',
'https://www.netmeds.com/prescriptions/acnesol-gel-20gm',
'https://www.netmeds.com/prescriptions/acnesol-a-nano-gel-15gm',
'https://www.netmeds.com/prescriptions/acnesol-nc-gel-20gm',
'https://www.netmeds.com/prescriptions/acnetoin-i-10-softgel-capsule-10s',
'https://www.netmeds.com/prescriptions/acnetoin-i-20mg-capsule-10s',
'https://www.netmeds.com/prescriptions/acnetor-ad-gel-15gm',
'https://www.netmeds.com/prescriptions/acnewar-gel-15gm',
'https://www.netmeds.com/prescriptions/acnewar-plus-gel-15gm',
'https://www.netmeds.com/prescriptions/acneycl-az-spray-50ml',
'https://www.netmeds.com/prescriptions/acnicin-gel-15gm',
'https://www.netmeds.com/prescriptions/acnin-acne-prone-skin-face-pack-50gm',
'https://www.netmeds.com/prescriptions/acnin-pimple-care-face-pack-50gm',
'https://www.netmeds.com/prescriptions/acnoff-anti-acne-bar-100gm',
'https://www.netmeds.com/prescriptions/acnorm-lotion-180ml',
'https://www.netmeds.com/prescriptions/acnovate-gel-15gm',
'https://www.netmeds.com/prescriptions/acnovate-trio-gel-15gm',
'https://www.netmeds.com/prescriptions/acstop-gel-20gm',
'https://www.netmeds.com/prescriptions/acstop-soap-75gm',
'https://www.netmeds.com/prescriptions/actame-10mg-tablet-10s',
'https://www.netmeds.com/prescriptions/actame-20-softgel-10s',
'https://www.netmeds.com/prescriptions/actame-30-capsule-10s',
'https://www.netmeds.com/prescriptions/actame-f-foaming-face-wash-150ml',
'https://www.netmeds.com/prescriptions/actreat-geltopical-20gm',
'https://www.netmeds.com/prescriptions/acutret-10mg-capsule-10-s',
'https://www.netmeds.com/prescriptions/acutret-20mg-capsule-10-s',
'https://www.netmeds.com/prescriptions/acutret-30mg-capsule-10-s',

```
        'https://www.netmeds.com/prescriptions/acutret-5mg-capsule-10-s',
        'https://www.netmeds.com/prescriptions/adafenin-gel-15gm'

valid_links = [link for link in all_med_links if link.startswith("https://")]
print(f"Total valid links: {len(valid_links)}")

# Checking for invalid links by seeing does it starts withh https
invalid_links = [link for link in all_med_links if not link.startswith("https://")]
print(f"Total invalid links: {len(invalid_links)}")
if invalid_links:
    print("Here are some invalid links:")
    print(invalid_links[:10])
```

```
Total valid links: 47796
Total invalid links: 0
```

```
# now we have 47796 drugs for each drug we will write a code to extarct the necassary drug info like name,generic,discount
extracted_data_list = []
failed_links = []

# Thread-safe lock
lock = threading.Lock()

#  logging
logging.basicConfig(filename='scraping_log.txt', level=logging.INFO,
                    format='%(asctime)s - %(levelname)s - %(message)s')

# Fetching page function with retries
def fetch_page(link, retries=3, delay=5):
    for attempt in range(retries):
        try:
            response = requests.get(link, timeout=10)
            response.raise_for_status()
            return response
        except requests.exceptions.RequestException as e:
            logging.warning(f"Error on {link}: {e} (attempt {attempt+1}/{retries})")
            time.sleep(delay)
    return None

# function for single link
def process_link(link):
    response = fetch_page(link)
    if not response:
        with lock:
            failed_links.append(link)
        return None
```

```
    # Parsing and extract data
    soup = BeautifulSoup(response.content, 'html.parser')
    product_name = soup.find('h1', class_='black-txt')
    product_name = product_name.get_text(strip=True) if product_name else "Not Available"

    # Extracting other fields similarly
    disease = soup.find('span', class_='gen_drug')
    disease = disease.get_text(strip=True) if disease else "Not Available"

    rx_required = soup.find('span', class_='req_Rx')
    rx_required = rx_required.get_text(strip=True) if rx_required else "Not Available"

    final_price = soup.find('span', class_='final-price')
    final_price = final_price.get_text(strip=True) if final_price else "Not Available"

    discount = soup.find('span', class_='disc-price')
    discount = discount.get_text(strip=True) if discount else "Not Available"

    country_of_origin = soup.find('span', class_='drug-manu ellipsis origin_text')
    country_of_origin = country_of_origin.get_text(strip=True) if country_of_origin else "Not Available"

    manufacturer = soup.find('span', class_='drug-manu')
    manufacturer = manufacturer.get_text(strip=True) if manufacturer else "Not Available"

    tablet_info = soup.find('div', class_='drug-conf')
    tablet_info = tablet_info.get_text(strip=True) if tablet_info else "Not Available"

    # Creating a dictionary of extracted data and appending
    extracted_data = {
        "Product Name": product_name,
        "Disease": disease,
        "Rx Required": rx_required,
        "Final Price": final_price,
        "Discount": discount,
        "Country of Origin": country_of_origin,
        "Manufacturer": manufacturer,
        "Tablet Info": tablet_info
    }

    # Appending the data to the shared list
    with lock:
        extracted_data_list.append(extracted_data)

    logging.info(f"Extracted data from {link}")

# saving progress if it crashed it would save till recent one
def save_progress():
```

```
    with lock:
        with open('extracted_data.json', 'w') as json_file:
            json.dump(extracted_data_list, json_file)
        with open('failed_links.txt', 'w') as file:
            for link in failed_links:
                file.write(f"{link}\n")
    logging.info("Progress saved.")

# Main function to process linkss in parallel
def main(all_med_links):
    batch_size = 100  # Saving progress every 100 links
    with ThreadPoolExecutor(max_workers=30) as executor:
        futures = [executor.submit(process_link, link) for link in all_med_links]

        for i, future in enumerate(as_completed(futures)):
            try:
                future.result()  #  exceptions if any occurred
            except Exception as e:
                logging.error(f"Error in processing: {e}")

            if (i + 1) % batch_size == 0:
                save_progress()
                print(f"Progress saved after {i+1} links.")


    save_progress()


if __name__ == "__main__":
    main(all_med_links)
```

```
Progress saved after 100 links.
Progress saved after 200 links.
Progress saved after 300 links.
Progress saved after 400 links.
Progress saved after 500 links.
Progress saved after 600 links.
Progress saved after 700 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/alerfix-m-tablet-15-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/aplev-m-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
Progress saved after 800 links.
Progress saved after 900 links.
Progress saved after 1000 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/casmont-lc-tablet-15s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/coekastle-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
Progress saved after 1100 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/covel-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (rea
WARNING:root:Error on https://www.netmeds.com/prescriptions/cetrimonte-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/casmont-lc-tablet-15s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
```

```
WARNING:root:Error on https://www.netmeds.com/prescriptions/cetrimonte-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/cetrimonte-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
Progress saved after 1200 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/ebmont-lc-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
WARNING:root:Error on https://www.netmeds.com/prescriptions/ergomont-l-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/ebmont-lc-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
Progress saved after 1300 links.
Progress saved after 1400 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/infinair-5-10mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed
WARNING:root:Error on https://www.netmeds.com/prescriptions/infinair-5-10mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed
WARNING:root:Error on https://www.netmeds.com/prescriptions/jotair-l-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
WARNING:root:Error on https://www.netmeds.com/prescriptions/jotair-l-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
WARNING:root:Error on https://www.netmeds.com/prescriptions/kastimin-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
WARNING:root:Error on https://www.netmeds.com/prescriptions/l-cetrino-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out.
WARNING:root:Error on https://www.netmeds.com/prescriptions/laveta-m-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
WARNING:root:Error on https://www.netmeds.com/prescriptions/lejet-m-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
WARNING:root:Error on https://www.netmeds.com/prescriptions/lelet-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (rea
WARNING:root:Error on https://www.netmeds.com/prescriptions/infinair-5-10mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed
WARNING:root:Error on https://www.netmeds.com/prescriptions/levokast-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
Progress saved after 1500 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/levomont-lc-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out
WARNING:root:Error on https://www.netmeds.com/prescriptions/levosiz-m-tablet-15-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (
WARNING:root:Error on https://www.netmeds.com/prescriptions/levova-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
WARNING:root:Error on https://www.netmeds.com/prescriptions/lukotas-lc-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out.
WARNING:root:Error on https://www.netmeds.com/prescriptions/locet-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (rea
WARNING:root:Error on https://www.netmeds.com/prescriptions/lasma-lc-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (r
WARNING:root:Error on https://www.netmeds.com/prescriptions/lukotas-lc-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out.
Progress saved after 1600 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/levonair-10-5mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed
WARNING:root:Error on https://www.netmeds.com/prescriptions/lezest-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
WARNING:root:Error on https://www.netmeds.com/prescriptions/locet-m-tablet-10s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (rea
WARNING:root:Error on https://www.netmeds.com/prescriptions/minolast-lc-10-5mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read tim
WARNING:root:Error on https://www.netmeds.com/prescriptions/monlevo-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
WARNING:root:Error on https://www.netmeds.com/prescriptions/montasma-plus-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed ou
WARNING:root:Error on https://www.netmeds.com/prescriptions/montecip-lc-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out
WARNING:root:Error on https://www.netmeds.com/prescriptions/montegress-lc-10mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read tim
Progress saved after 1700 links.
WARNING:root:Error on https://www.netmeds.com/prescriptions/minolast-lc-10-5mg-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read tim
WARNING:root:Error on https://www.netmeds.com/prescriptions/monlevo-tablet-10-s: HTTPSConnectionPool(host='www.netmeds.com', port=443): Read timed out. (re
```

Excel Dashboard on Netmeds Data

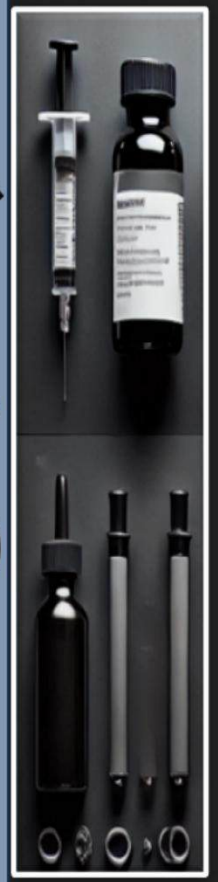
In this project, I created an **interactive Excel dashboard** to analyze **Netmeds pricing data** for various pharmaceutical products. The dataset was cleaned and transformed using **Text to Columns** to categorize products into **tablets, ointments, creams, lotions, and more**.

The dashboard consists of **four sheets**, with the main focus on **pricing analysis**. Key features include:

- **Three slicers** for filtering by **Rx Required, Disease, and Country**
- **Segmentation of product types** for better analysis
- **Data visualization** using Pivot Tables & Charts
- **Enhanced filtering and interactivity** for better insights

This project demonstrates **data cleaning, transformation, and visualization skills** in Excel, making it useful for pharmaceutical pricing analysis. 🚀

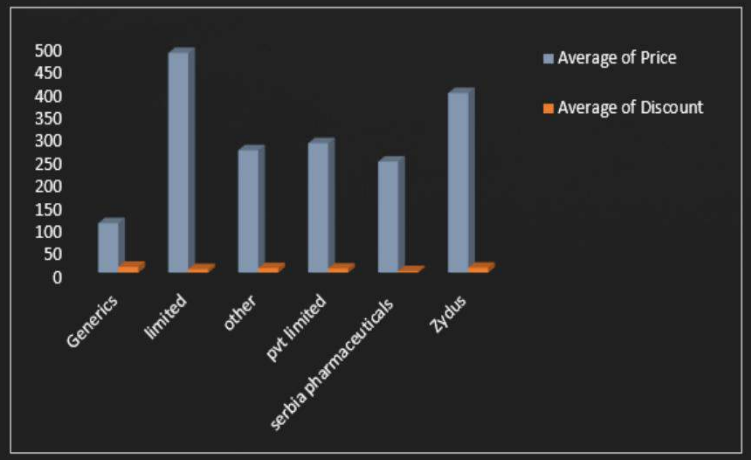
Netmeds Pricing Analysis

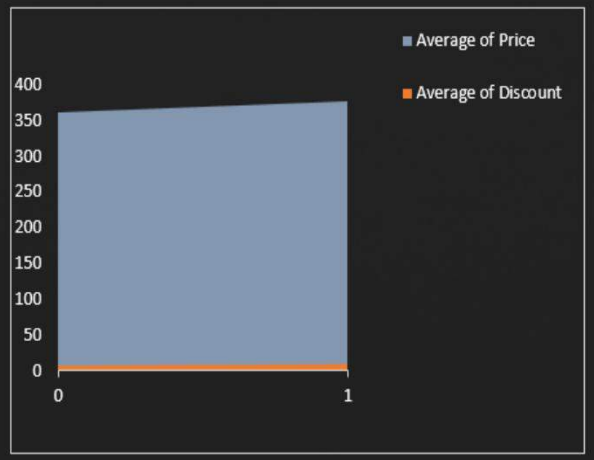


₹ 375
Average Price

₹ 8.72
Average Discount

23,897
Tablet Count





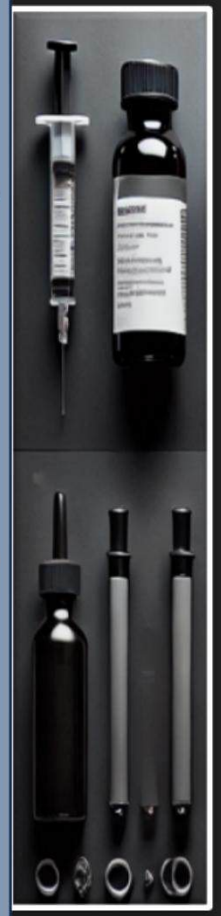
Disease
- Appetite
- Arrhythmiasis
- Arthritis

Manufacturer
- altiuz pharma care
- amazing research laboratories
- amwill healthcare

Rx Required
- 0
- 1

Country of Origin
- Australia
- Belgium
- Canada

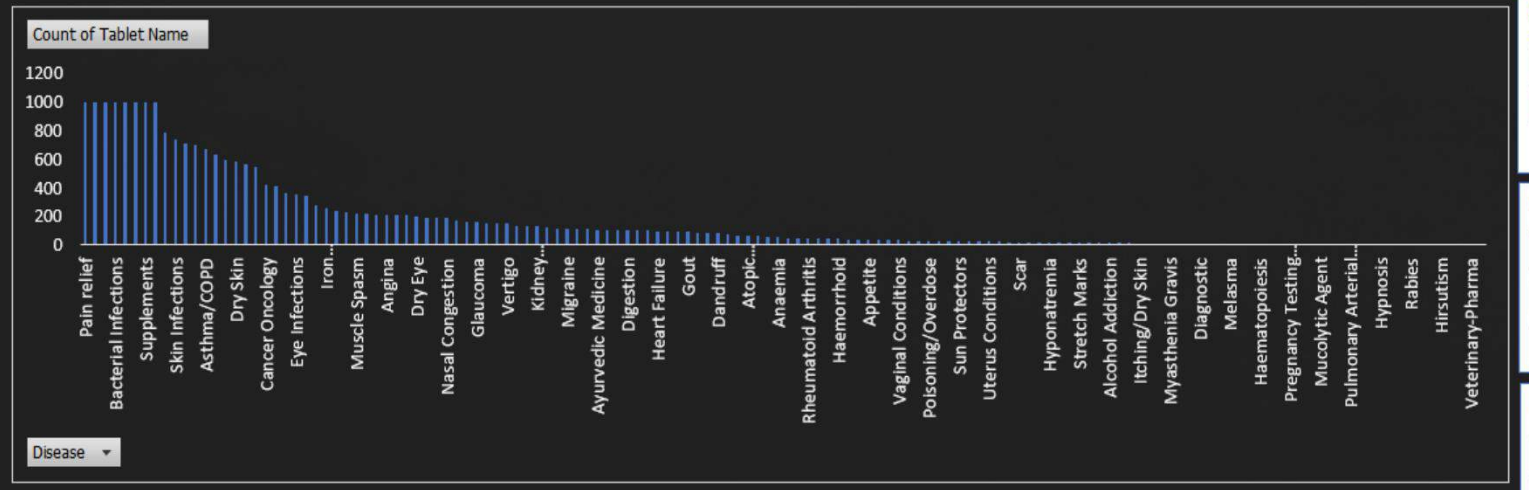
Netmeds Medication Insights



₹ 375
Average Price

₹ 8.72
Average Discount

23,897
Tablet Count



Disease
- Acne
- ADHD
- Alcohol Addiction

Rx Required
- 0
- 1

Country of Origin
- Australia
- Belgium
- Canada

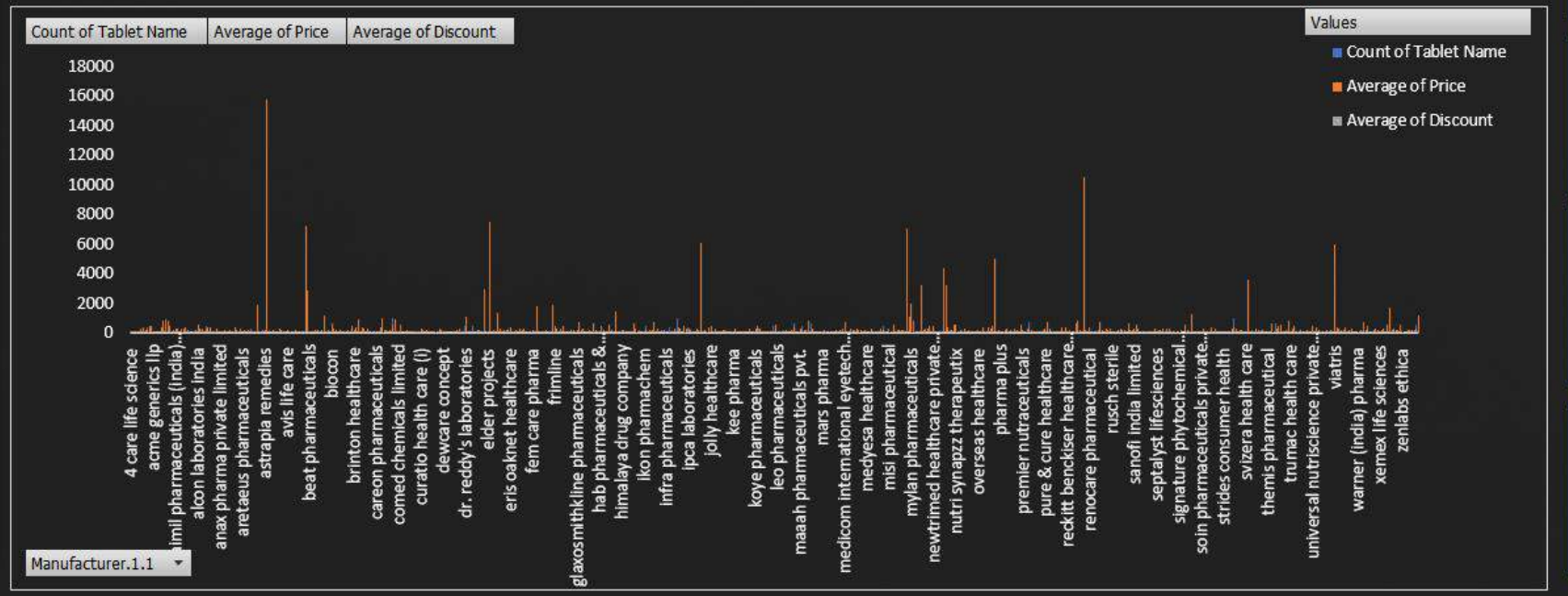
Manufacturer.1.1
- 4 care life science
- a n pharmacia laborat...
- aagam life sciences

Netmeds Manufacturer Insights

₹ 375
Average Price

₹ 8.72
Average Discount

23,897
Tablet Count



Disease
- Acne
- ADHD
- Alcohol Addiction

Rx Required
- 0
- 1

Country of Origin
- Australia
- Belgium
- Canada

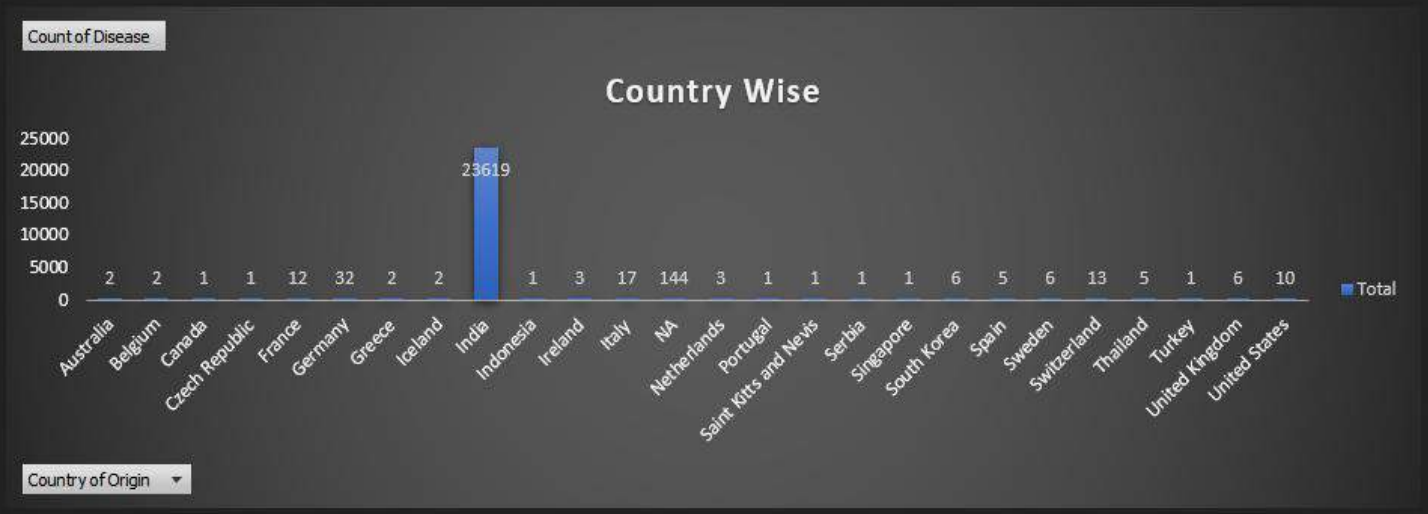
Manufacturer.1.1
- siskan pharma
- skinmedis laboratories
- skn organics

Netmeds Country wise Insights

₹ 375
Average Price

₹ 8.72
Average Discount

23,897
Tablet Count



Disease
- Acne
- ADHD
- Alcohol Addiction

Rx Required
- 0
- 1

Country of Origin
- Australia
- Belgium
- Canada

Manufacturer.1.1
- 4 care life science
- a n pharmacia laboratori...
- aagam life sciences

Netmeds Revenue Model

◈ Commission-Based Revenue (Primary Source)

◈ **How It Works**:

- Netmeds **earns commissions** on the sale of prescription medicines, OTC (over-the-counter) drugs, and FMCG products listed on its platform.
- Pharmaceutical companies, distributors, and third-party sellers pay Netmeds a percentage of their sales revenue for selling through the platform.

◈ **Revenue Contribution**:

- In **FY18, Netmeds earned ₹4.46 crore** from commissions.

◈ The company mainly caters to the **chronic segment** (patients requiring regular medications), a market worth **$10 billion annually** in India.

◈ Marketing & Advertising Revenue

◈ **How It Works**:

- Netmeds earns revenue by offering **promotional space and advertising services** to pharmaceutical companies and healthcare brands.
- This includes:
  ✓ **Featured product placements**
  ✓ **Banner ads on the website/app**
  ✓ **Email and push notification promotions**

◈ **Revenue Contribution**:

- Brands pay **marketing fees** to boost their product visibility and sales on Netmeds' platform.

◈ Shipping & Convenience Charges

◈ **How It Works**:

- Netmeds **charges customers for delivery services**, depending on order size, urgency, and location.
- Standard delivery is **free** for orders above a certain amount, while **express or same-day delivery** comes with additional charges.

◈ B2B Wholesale Business

◈ **How It Works**:

- Netmeds is evaluating a **B2B model**, supplying medicines to local pharmacies at **wholesale rates**.

◈ This model will allow the company to **sell in bulk to retailers and medical stores**, increasing its revenue potential.

◈ Franchise Model (Expansion Revenue Stream)

◈ **How It Works**:

- Netmeds is expanding its **physical presence** by offering franchise opportunities.
- Franchise owners pay an **initial investment (~₹20 lakh)** and share revenue/profit with Netmeds.

Analysis of Netmeds' Prescription-Driven Market and Potential Gaps in Medication Availability

- **Prescription-Based Market:** Around **88.98%** of drugs on Netmeds require a doctor's prescription, highlighting the dominance of prescription-driven sales.
- **Stroke and Tuberculosis (TB) Medications:**
  - In India, approximately **49 out of 1,000** people die due to stroke, and **29 out of 1,000** from TB.
  - While acute cases might not prefer online purchases, post-stroke patients (especially those with hemiplegia or paresis) and TB patients (who require prolonged medication) could benefit from online accessibility.
  - Netmeds should consider adding these essential medications to their stock.
- **Ischemic Heart Disease & Chronic Conditions:**
  - **121 out of 1,000** deaths in India are due to ischemic heart disease.
  - Patients with stents, bypass surgery, or CABG need lifelong medication.
  - While Netmeds has a strong inventory for diabetes, high cholesterol, and hypertension, expanding offerings for cardiac care medications can enhance availability and meet patient needs.

Estimating Netmeds Revenue

- Notified TB patients: 26 lakh TB patients in 2023 (increase from 24.22 lakh cases in 2022).
  Reporting of cases: ~67% reporting being done by the public sector and ~33%) from private sector.
- **Step 1: Estimating 2025 Revenue**
- **Baseline 2024 Revenue**
- Netmeds' **2024 revenue** was **₹67.24 crore** (a drop from 2023).
- **Adding Revenue from TB Medications**
- We estimated a potential **market size of 39,000 TB patients**.
- If **each patient spends ₹3,000 per year** on TB medications, that adds: 39,000×3,000=₹117crore
- If **each patient spends ₹4,000 per year**, that adds: 39,000×4,000=₹156crore
- Assuming **Netmeds' existing revenue also grows slightly (~10-15%)** 67.24×1.15=₹77crore
- **Final Revenue Estimate (Existing + TB Market) = ₹150 - ₹170 crore**

Step 2: Estimating Expenses

- Netmeds' **2024 expenses** were **₹60.83 crore**. With expansion into TB meds, costs will rise due to:
- **Stocking & Procurement** of TB medicines
- **Marketing & Awareness**
- **Logistics & Distribution**
- Let's assume a **50-70% increase** in expenses:

$$60.83 \times 1.8 = ₹110 \text{crore}$$

$$60.83 \times 2.1 = ₹130$$

Thus, **estimated expenses for 2025 = ₹110 - ₹130 crore**

- **Step 3: Estimating Profit**
- Profit is calculated as:
- Profit=Revenue−Expenses
  - (₹150 Cr revenue - ₹130 Cr expenses) = **₹20 Cr profit**
- **Upper bound** (₹170 Cr revenue - ₹110 Cr expenses) = **₹35 Cr profit**
- Thus, the **realistic profit estimate is ₹20 - ₹35 crore.**
- **Profit Increase=151%**

- **Recommendation for Netmeds: Enhancing Accessibility & Visibility of Post-Stroke Medications**
- To better serve patients recovering from **stroke**, Netmeds can implement the following strategies to **increase trust, improve visibility, and drive sales** for essential post-stroke medications.
- **1 ☐ Create a Dedicated "Post-Stroke Care" Section**

- Introduce a separate category for **Post-Stroke Medications**, similar to existing sections for **Diabetes, Hypertension, and Cardiac Care**.
- Include:
  Blood thinners & anticoagulants (e.g., Aspirin, Clopidogrel, Warfarin) to prevent further strokes.
  Cholesterol-lowering drugs (e.g., Statins like Atorvastatin, Rosuvastatin) to reduce heart disease risk.
  Blood pressure medications (e.g., ACE inhibitors, Beta-blockers) to prevent hypertension-related strokes.
  Neuroprotective drugs & nerve regeneration supplements (e.g., Citicoline, Piracetam) to support brain recovery.
  Physical rehabilitation aids (e.g., muscle relaxants, physiotherapy supplements).

- **2 ☐ Cross-Reference Post-Stroke Medications in Related Categories**
- Since stroke-related medicines fall under different categories, Netmeds can **cross-list** them in:
  Cardiac Care – Blood thinners & cholesterol-lowering drugs.
  Hypertension Management – BP medications preventing stroke recurrence.
  Neurology & Brain Health – Cognitive enhancers & nerve recovery drugs.
  Pain & Muscle Relaxants – Medications for post-stroke spasticity and movement recovery.

- **Add a small note/link under relevant medications**:
  → *"This medication is commonly prescribed for stroke recovery. Explore our full Post-Stroke Care range → [link]."*

◈ **3 ☐ Build Trust & Awareness Among Stroke Patients & Caregivers**

- Include an **educational section** on the "Post-Stroke Care" page, addressing:
  Why long-term medication adherence is crucial.
  Commonly prescribed medications and their benefits.
  Lifestyle modifications for stroke survivors.
- **Partner with neurologists or stroke specialists** to provide credibility & recommendations.

◈ **Expected Impact**

◈ **Improved customer confidence** – Stroke survivors & caregivers will see Netmeds as a **trusted platform** for long-term stroke recovery.
Higher conversions – Dedicated categorization & visibility make it easier for customers to find & buy essential medications.
Stronger market positioning – Netmeds can **differentiate itself** by addressing critical healthcare needs beyond general medications.

Unveiling Data-Driven Insights: Advanced SQL Analysis on E-Pharma Products

Unveiling Data-Driven Insights: Advanced SQL Analysis on E-Pharma Products

This report provides deep insights into pharmaceutical product pricing, discounts, and market trends using advanced SQL queries, helping businesses make data-driven decisions

Objective & Business Impact

- Understanding pricing & discount strategies across different company types.
- Identifying market trends for different medicine forms (tablets, injections, creams, etc.).
- Comparing RX vs. Non-RX medicine pricing & discounts.
- Analyzing manufacturer trends across different regions.
- Finding the most competitive and premium-priced pharma brands.

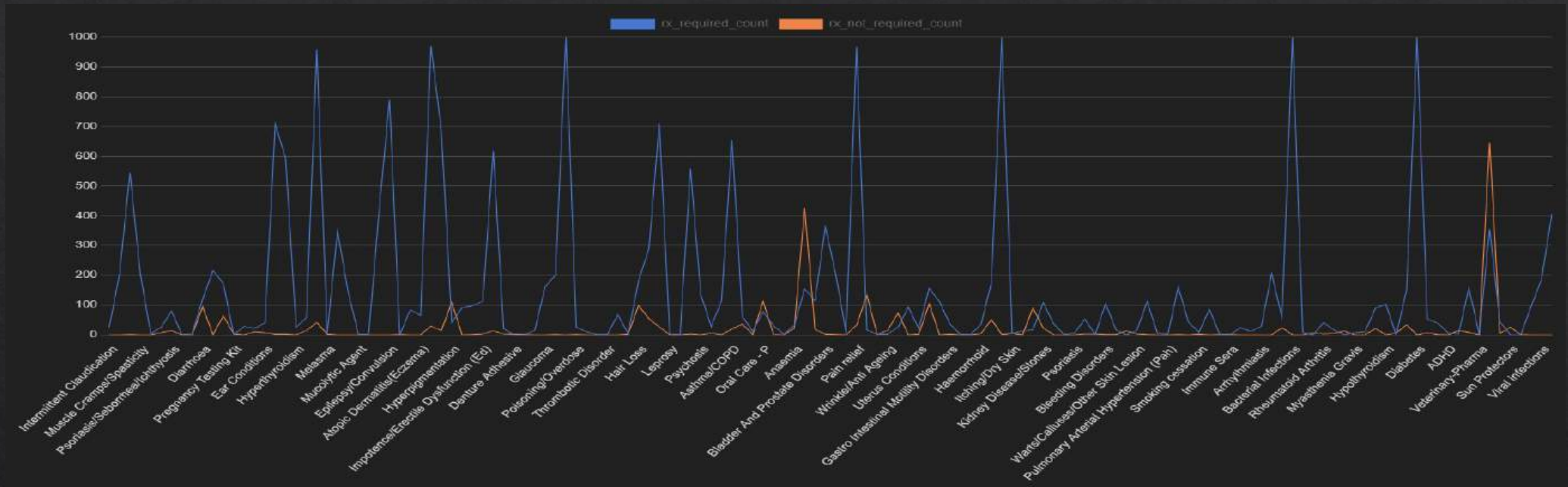
SQL Techniques Used

: Window Functions (rank,row_number,dense_rank)
: Common Table Expressions (CTEs) for structured analysis
: Subqueries for filtering and trend detection
: Aggregations & Grouping for business-level insights

Summary Statistics of Prescription and Non-Prescription Medications

```sql
WITH DiseaseCounts AS (
    SELECT disease,
           SUM(CASE WHEN rx_required = 1 THEN 1 ELSE 0 END) AS rx_required_count,
           SUM(CASE WHEN rx_required = 0 THEN 1 ELSE 0 END) AS rx_not_required_count,
           COUNT(*) AS total_count_per_disease
    FROM netmedsdata
    GROUP BY disease
OverallCounts AS (
    SELECT
        SUM(CASE WHEN rx_required = 1 THEN 1 ELSE 0 END) AS overall_rx_required_count,
        SUM(CASE WHEN rx_required = 0 THEN 1 ELSE 0 END) AS overall_rx_not_required_count,
        COUNT(*) AS overall_total_count
    FROM netmedsdata
)
SELECT dc.disease,
       dc.rx_required_count,
       dc.rx_not_required_count,
       ROUND(CAST(dc.rx_required_count AS NUMERIC) * 100 / dc.total_count_per_disease, 2) AS rx_required_percentage,
       ROUND(CAST(oc.overall_rx_required_count AS NUMERIC) * 100 / oc.overall_total_count, 2) AS overall_rx_required_percentage,
       ROUND(CAST(oc.overall_rx_not_required_count AS NUMERIC) * 100 / oc.overall_total_count, 2) AS overall_rx_not_required_percentage
FROM DiseaseCounts dc, OverallCounts oc;
```

Visualization of output

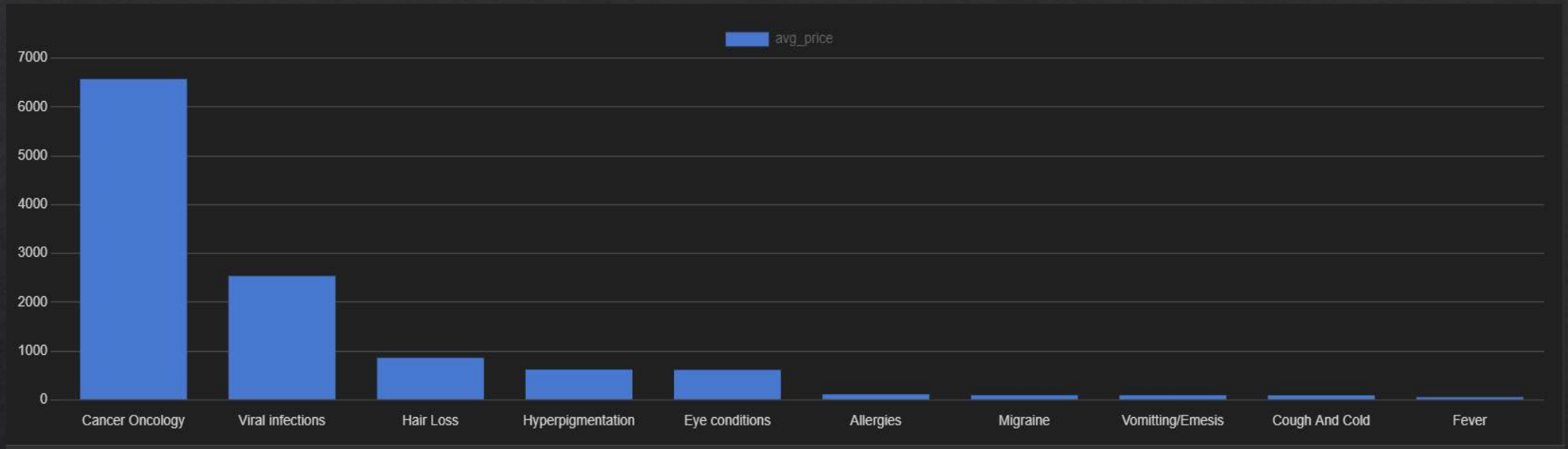


	disease character varying (255)	rx_required_count bigint	rx_not_required_count bigint	rx_required_percentage numeric	overall_rx_required_percentage numeric	overall_rx_not_required_percentage numeric
1	Intermittent Claudication	25	0	100.00	88.95	11.05
2	Auto Immune Disease	206	0	100.00	88.95	11.05
3	Depression	544	1	99.82	88.95	11.05
4	Muscle Cramps/Spasticity	204	0	100.00	88.95	11.05
5	Anal Fissure	3	0	100.00	88.95	11.05
6	Bone Metabolism	25	7	78.13	88.95	11.05
7	Psoriasis/Seborrhea/Ichthyosis	79	14	84.95	88.95	11.05
8	Head Lice	1	0	100.00	88.95	11.05
9	Haematopoiesis	3	0	100.00	88.95	11.05
10	Diarrhoea	120	93	56.34	88.95	11.05

Top 5 and Bottom 5 Diseases by Average Price

```sql
WITH RankedAverages AS (
    SELECT
        disease,
        ROUND(AVG(price), 2) AS avg_price,
        ROW_NUMBER() OVER (ORDER BY AVG(price) DESC) AS rn_desc,
        ROW_NUMBER() OVER (ORDER BY AVG(price) ASC) AS rn_asc
    FROM netmedsdata
    GROUP BY disease
    HAVING COUNT(*) > 100
)
SELECT disease, avg_price
FROM RankedAverages
WHERE rn_desc <= 5
UNION ALL
SELECT disease, avg_price
FROM RankedAverages
WHERE rn_asc <= 5
ORDER BY avg_price DESC;
```

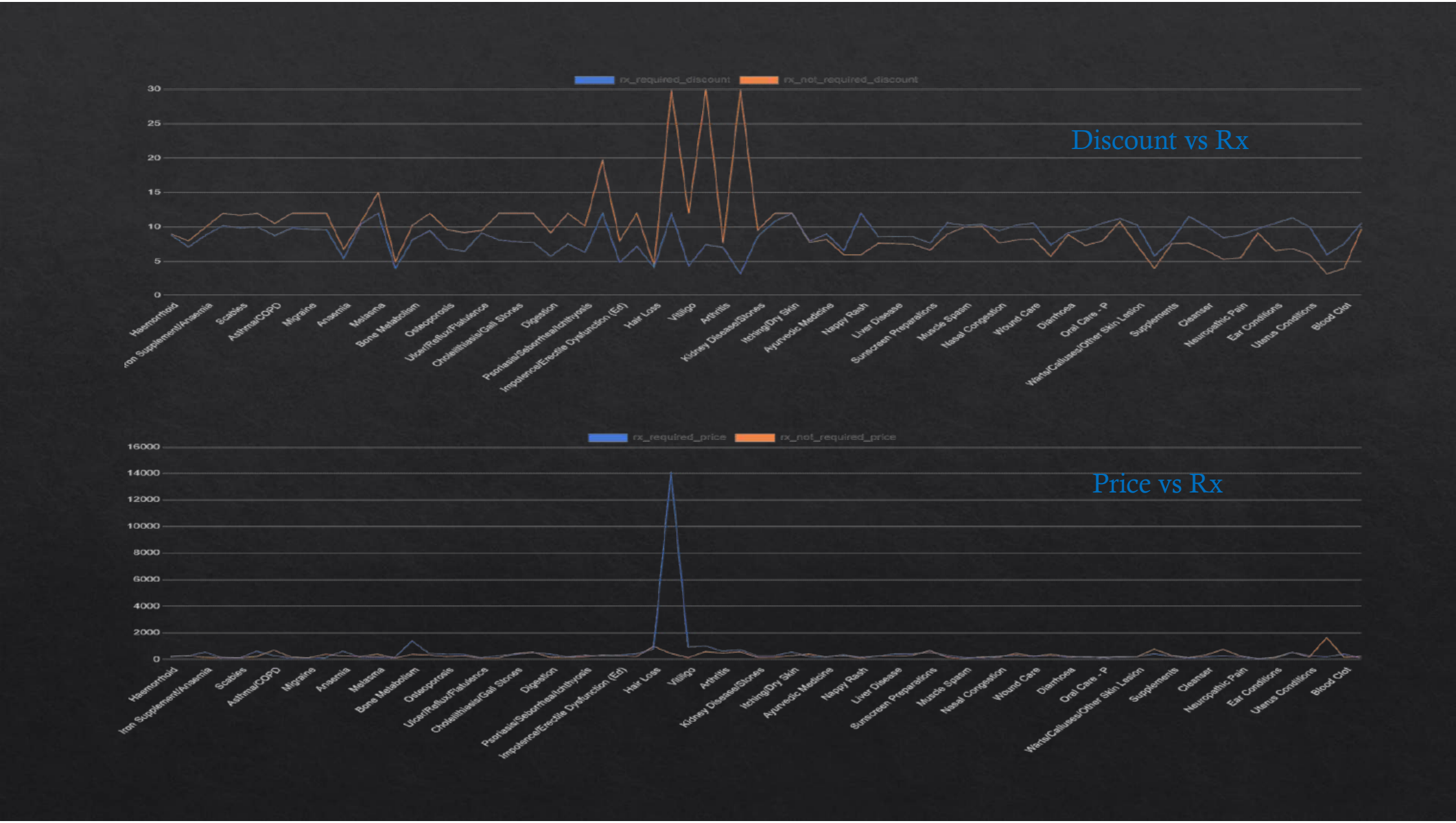
	disease character varying (255)	avg_price numeric
1	Cancer Oncology	6573.01
2	Viral infections	2537.45
3	Hair Loss	861.54
4	Hyperpigmentation	621.17
5	Eye conditions	614.96
6	Allergies	113.26
7	Migraine	96.54
8	Vomitting/Emesis	96.52
9	Cough And Cold	94.26
10	Fever	56.66



- **Cancer Oncology (6573.01):** Indicates resource-intensive cancer care.
- **Anesthesia General (4156.50):** Suggests complex procedures needing specialized expertise.
- **Wrinkle/Anti-Aging (1469.18):** Shows high cost of advanced aesthetic treatments.
- **Parasitic Worms (48.19) to Mydriasis (92.43):** Lower costs for common ailments needing less intensive care.

Average Price and Discount Comparison for Diseases Based on Prescription Requirement

```sql
WITH avg_values AS (
    SELECT disease,
           AVG(CASE WHEN rx_required = 1 THEN price ELSE NULL END) AS rx_required_price,
           AVG(CASE WHEN rx_required = 0 THEN price ELSE NULL END) AS rx_not_required_price,
           AVG(CASE WHEN rx_required = 1 THEN discount ELSE NULL END) AS rx_required_discount,
           AVG(CASE WHEN rx_required = 0 THEN discount ELSE NULL END) AS rx_not_required_discount
    FROM netmedsdata
    WHERE rx_required IS NOT NULL
    GROUP BY disease
)
SELECT disease,

       ROUND(COALESCE(rx_required_price, 0), 2) AS rx_required_price,
       ROUND(COALESCE(rx_not_required_price, 0), 2) AS rx_not_required_price,
       ROUND(
           CASE
               WHEN rx_not_required_price > 0
               THEN (rx_required_price - rx_not_required_price) / rx_not_required_price * 100
               ELSE 0
           END, 2) || '%' AS price_percentage_diff,


       ROUND(COALESCE(rx_required_discount, 0), 2) AS rx_required_discount,
       ROUND(COALESCE(rx_not_required_discount, 0), 2) AS rx_not_required_discount,
       ROUND(
           CASE
               WHEN rx_not_required_discount > 0
               THEN (rx_required_discount - rx_not_required_discount) / rx_not_required_discount * 100
               ELSE 0
           END, 2) || '%' AS discount_percentage_diff
FROM avg_values
WHERE rx_required_price > 0
   AND rx_not_required_price > 0
   AND rx_required_discount > 0
   AND rx_not_required_discount > 0
ORDER BY price_percentage_diff DESC;
```

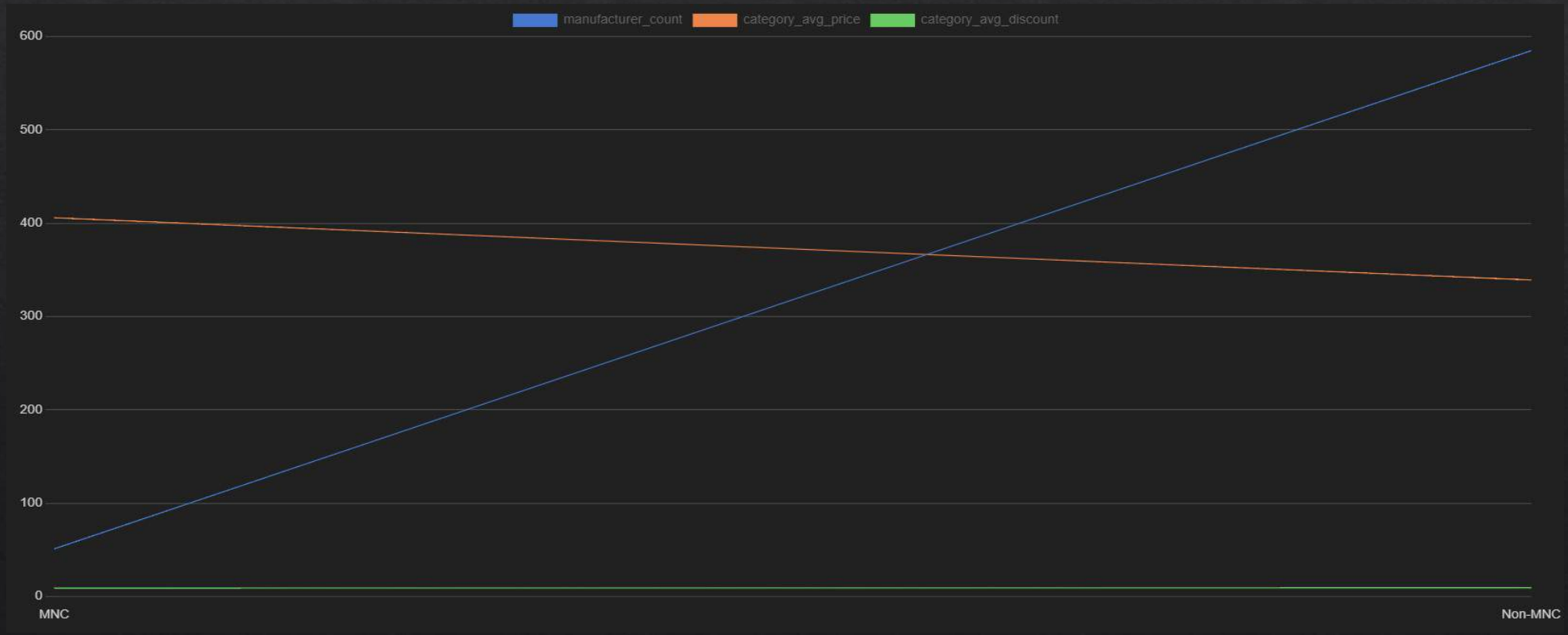

rx_required_discount
rx_not_required_discount
Discount vs Rx
30
25
20
15
10
5
0
Haemorrhoid
Iron Supplement/Anaemia
Scabies
Asthma/COPD
Migraine
Anaemia
Melasma
Bone Metabolism
Osteoporosis
Ulcer/Reflux/Flatulence
Cholelithiasis/Gall Stones
Digestion
Psoriasis/Seborrheal/Ichthyosis
Impotence/Erectile Dysfunction (Ed)
Hair Loss
Vitiligo
Arthritis
Kidney Disease/Stones
Itching/Dry Skin
Ayurvedic Medicine
Nappy Rash
Liver Disease
Sunscreen Preparations
Muscle Spasm
Nasal Congestion
Wound Care
Diarrhoea
Oral Care - P
Warts/Calluses/Other Skin Lesion
Supplements
Cleanser
Neuropathic Pain
Ear Conditions
Uterus Conditions
Blood Clot
rx_required_price
rx_not_required_price
Price vs Rx
16000
14000
12000
10000
8000
6000
4000
2000
0
Haemorrhoid
Iron Supplement/Anaemia
Scabies
Asthma/COPD
Migraine
Anaemia
Melasma
Bone Metabolism
Osteoporosis
Ulcer/Reflux/Flatulence
Cholelithiasis/Gall Stones
Digestion
Psoriasis/Seborrheal/Ichthyosis
Impotence/Erectile Dysfunction (Ed)
Hair Loss
Vitiligo
Arthritis
Kidney Disease/Stones
Itching/Dry Skin
Ayurvedic Medicine
Nappy Rash
Liver Disease
Sunscreen Preparations
Muscle Spasm
Nasal Congestion
Wound Care
Diarrhoea
Oral Care - P
Warts/Calluses/Other Skin Lesion
Supplements
Cleanser
Neuropathic Pain
Ear Conditions
Uterus Conditions
Blood Clot

•Analysis of Manufacturers by Product Count, Average Price, and Average Discount, Grouped by MNC/Non-MNC Category

```sql
WITH ManufacturerCategory AS (
    SELECT manufacturer,
           COUNT(*) AS product_count,
           ROUND(AVG(price), 2) AS avg_price,
           ROUND(AVG(discount), 2) AS avg_discount,
           CASE
               WHEN COUNT(*) > 100 THEN 'MNC'
               ELSE 'Non-MNC'
           END AS category
    FROM netmedsdata
    GROUP BY manufacturer
)
SELECT category,
       COUNT(*) AS manufacturer_count,
       SUM(product_count) AS total_products,
       ROUND(AVG(avg_price), 2) AS category_avg_price,
       ROUND(AVG(avg_discount), 2) AS category_avg_discount
FROM ManufacturerCategory
GROUP BY category
ORDER BY total_products DESC;
```

	category text	manufacturer_count bigint	total_products numeric	category_avg_price numeric	category_avg_discount numeric
1	MNC	51	16064	405.86	8.97
2	Non-MNC	585	7833	339.19	9.24

Visualization of output



Business Recommendations for Online Pharmacy Expansion

1. **Optimize Product Availability:**
    1. MNCs have a dominant presence in terms of product count. To maintain a competitive edge, I recommend **strengthening partnerships with top-selling MNC brands** to ensure a robust product lineup.
    2. Since Non-MNCs focus on affordability, **expanding the Non-MNC range** can help attract budget-conscious customers and increase market penetration.
2. **Strategic Pricing & Discounts:**
    1. MNCs maintain **higher pricing with slightly lower discounts**. I suggest **leveraging brand reputation and quality** to justify premium pricing while offering strategic discounts on selected products to drive conversions.
    2. Non-MNCs rely on **higher discounts** to attract buyers. To maximize sales, I recommend **testing targeted promotions, cashback offers, and bulk purchase discounts** to see if they drive higher revenue.
3. **Market Expansion Strategy:**
    1. MNC products tend to attract **urban, brand-conscious customers**, whereas Non-MNCs cater to **price-sensitive buyers in Tier 2 & Tier 3 cities**.
    2. I recommend **expanding the Non-MNC product portfolio in smaller cities and rural areas** while maintaining a strong MNC presence in urban markets.
4. **Targeted Marketing Approach:**
    1. For **premium customers**, introduce **loyalty programs and exclusive deals on MNC products** to enhance customer retention.

- For **budget-conscious customers**, focus on **promotions and affordability-driven campaigns** to increase engagement with Non-MNC products.

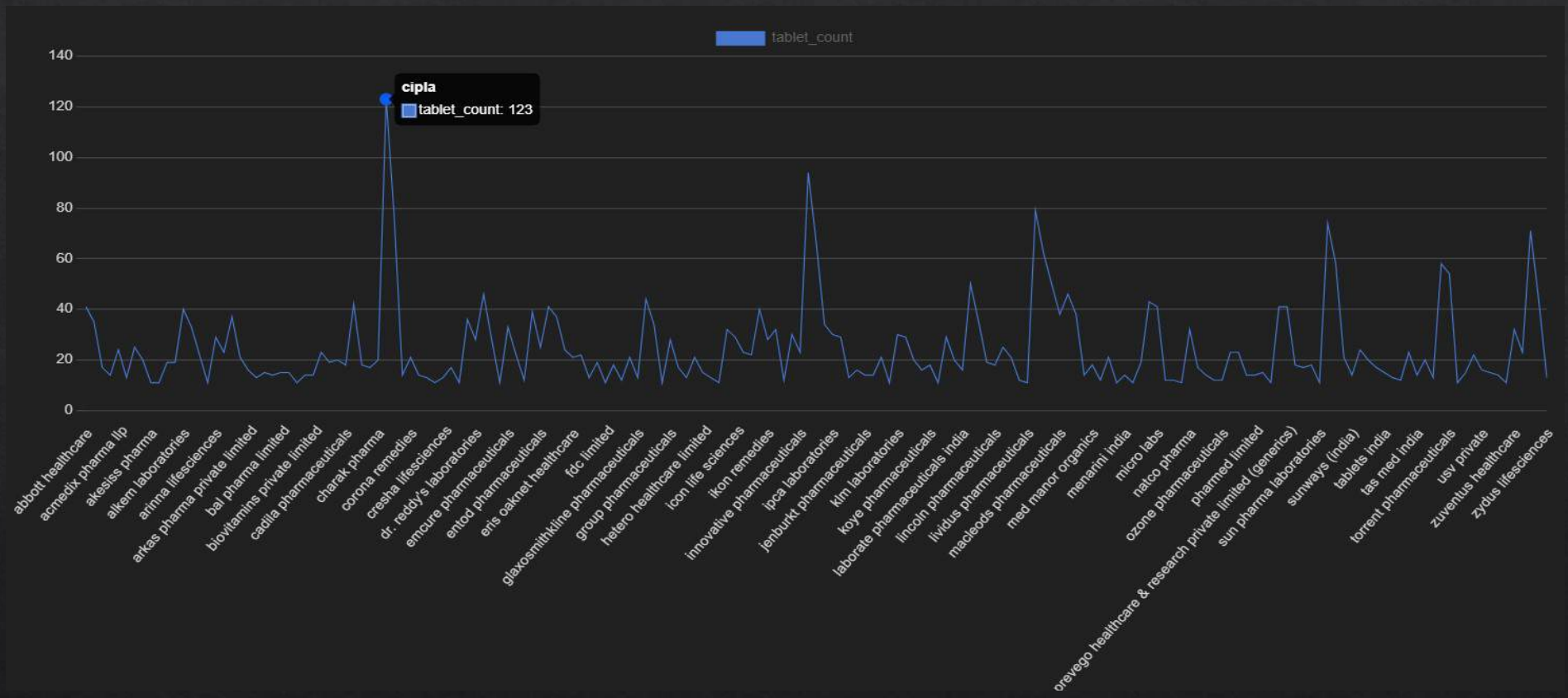
◈ **Next Steps for Growth:**

- **Analyze customer purchase data** to personalize pricing and discount strategies.
- **Balance MNC & Non-MNC offerings** to cater to different customer segments.
- **Run localized marketing campaigns** to expand into new markets effectively.

Top 2 and Bottom 2 Diseases by Tablet Count for Each Manufacturer with Minimum Count Filter

```sql
WITH ranked_diseases AS (
    SELECT
        manufacturer,
        disease,
        COUNT(tablet_info) AS tablet_count,
        ROW_NUMBER() OVER (PARTITION BY manufacturer ORDER BY COUNT(tablet_info) DESC) AS rank_desc,
        ROW_NUMBER() OVER (PARTITION BY manufacturer ORDER BY COUNT(tablet_info) ASC) AS rank_asc
    FROM netmedsdata
    GROUP BY manufacturer, disease
)
SELECT
    manufacturer,
    disease,
    tablet_count
FROM ranked_diseases
WHERE (rank_desc <= 2 OR rank_asc <= 2)
    AND tablet_count > 10
ORDER BY manufacturer, rank_desc;
```

manufacturer character varying (255)	disease character varying (255)	tablet_count bigint
abbott healthcare	Bacterial Infections	41
abbott healthcare	Epilepsy/Convulsion	35
abbott india	Hormonal Therapy	17
abbott india	Hypothyroidism	14
acmedix pharma llp	Diabetes	24
acmedix pharma llp	Hypertension	13
ajanta pharma	Dry Skin	25
ajanta pharma	Hypertension	20


tablet_count
140
120
100
80
60
40
20
0
cipla
tablet_count: 123
abbott healthcare
acmedix pharma llp
akesiss pharma
alkem laboratories
arinna lifesciences
arkas pharma private limited
bal pharma limited
biovitamins private limited
cadila pharmaceuticals
charak pharma
corona remedies
cresha lifesciences
dr. reddy's laboratories
emcure pharmaceuticals
entod pharmaceuticals
eris oaknet healthcare
fdc limited
glaxosmithkline pharmaceuticals
group pharmaceuticals
hetero healthcare limited
icon life sciences
ikon remedies
innovative pharmaceuticals
ipca laboratories
jenburkt pharmaceuticals
klm laboratories
koye pharmaceuticals
laborate pharmaceuticals india
lincoln pharmaceuticals
lividus pharmaceuticals
macleods pharmaceuticals
med manor organics
menarini india
micro labs
natco pharma
ozone pharmaceuticals
pharmed limited
orevegg healthcare & research private limited (generics)
sun pharma laboratories
sunways (india)
tablets india
tas med india
torrent pharmaceuticals
usv private
zuventus healthcare
zydus lifesciences

Top 5 High-Cost Medicines and Their Cheaper Alternatives by Disease

```sql
WITH RankedMedicines AS (
    SELECT
        disease,
        tablet_name,
        tablet_info,
        price,
        ROW_NUMBER() OVER (PARTITION BY tablet_info ORDER BY price DESC) AS rank
    FROM netmedsdata
),
CheaperAlternatives AS (
    SELECT
        rm.disease,
        rm.tablet_name AS high_cost_tablet_name,
        rm.tablet_info AS high_cost_tablet_info,
        rm.price AS high_cost_price,
        alt.tablet_name AS alternative_tablet_name,
        alt.tablet_info AS alternative_tablet_info,
        alt.price AS alternative_price,
        ROW_NUMBER() OVER (PARTITION BY rm.tablet_info ORDER BY alt.price ASC) AS alt_rank
    FROM RankedMedicines rm
    JOIN netmedsdata alt
        ON rm.tablet_info = alt.tablet_info
    AND alt.price < rm.price
SELECT *
FROM CheaperAlternatives
WHERE alt_rank <= 5
ORDER BY high_cost_tablet_info, alternative_price;
```

alternative_tablet_info text	alternative_price integer
4 Butanedisulfonate 400 mg+Ademetionine-1	1190
Abacavir 600 mg+Lamivudine 300 mg	1486
Abiraterone 250 mg	7420
Abiraterone 250 mg	13000
Abiraterone 250 mg	15832
Abiraterone 250 mg	16456
Abiraterone 250 mg	24640
Abiraterone 500 mg	15832
Abiraterone 500 mg	16456

	disease character varying (255)	high_cost_tablet_name character varying (255)	high_cost_tablet_info text
1	Liver Disease	CIRROSAM 400 Tablet 15's	4 Butanedisulfonate 400 mg+Ademetionine-1
2	Liver Disease	CIRROSAM 400 Tablet 15's	4 Butanedisulfonate 400 mg+Ademetionine-1
3	Viral infections	ABEC L Tablet 30's	Abacavir 600 mg+Lamivudine 300 mg
4	Cancer Oncology	Abiratas 250mg Tablet 120'S	Abiraterone 250 mg
5	Cancer Oncology	Abiratas 250mg Tablet 120'S	Abiraterone 250 mg
6	Cancer Oncology	Abiratas 250mg Tablet 120'S	Abiraterone 250 mg

Insights From Query

Higher Sales & Lower Cart Abandonment:

- Customers see **cheaper alternatives**, making them more likely to **complete a purchase**.

Better Customer Experience & Retention:

- Users **save money** and find **substitutes** easily, increasing **trust & loyalty**.

Reduced Lost Sales (Out-of-Stock Solutions):

- If a medicine is unavailable, Netmeds suggests **similar alternatives**, preventing lost revenue.

Data-Driven Pricing & Supplier Negotiation:

- Helps Netmeds track demand for **cheaper substitutes**, enabling **better pricing strategies & bulk purchase deals**.

Result: More purchases, happy customers, and a stronger market position!

Insights From Query

- **Personalized medicine recommendations** based on a customer's past purchase behavior can significantly **boost sales & user satisfaction**.

1. **Customer Segmentation Based on Purchase History:**
    1. **High-spending customers** → Show **premium alternatives** first.
    2. **Budget-conscious customers** → Show **cheaper alternatives** first.
2. **Dynamic Pricing & Recommendations:**
    1. If a user frequently buys **expensive medicines**, prioritize **higher-cost alternatives** in search results.
    2. If a user prefers **affordable options**, highlight the **cheapest substitutes** upfront.
3. **Boosting Conversions & Revenue:**
    1. **Personalized pricing** increases the chance of a sale.
    2. Netmeds can **upsell premium products** to high-spending users.
    3. **Discounts can be targeted** at price-sensitive customers to retain them.

- 🚀**Business Impact:**
- ✓ **Higher conversion rates** (customers see what they prefer).
  ✓ **Increased revenue & profitability** through smart recommendations.
  ✓ **Stronger customer loyalty** by making shopping more relevant.

"Leading Manufacturer with Highest Average Discount Based on Top 20 Diseases"

```sql
WITH RankedDiseases AS (
    SELECT
        manufacturer,
        COUNT(*) AS tablet_count,
        AVG(discount) AS avg_discount
    FROM netmedsdata
    GROUP BY manufacturer
    HAVING COUNT(*) >= 100
),
ManufacturerWithMostDiscount AS (
    SELECT
        manufacturer,
        ROUND(AVG(avg_discount), 2) AS avg_discount
    FROM RankedDiseases
    GROUP BY manufacturer
    ORDER BY avg_discount DESC
    LIMIT 5
)
SELECT dism.manufacturer, m.avg_discount, n.disease, COUNT(n.disease) AS disease_count
FROM ManufacturerWithMostDiscount AS m
INNER JOIN netmedsdata AS n ON m.manufacturer = n.manufacturer
GROUP BY m.manufacturer, m.avg_discount, n.disease
having COUNT(n.disease) > 20
ORDER BY m.manufacturer, m.avg_discount, disease_count DESC;
```

	manufacturer character varying (255)	avg_discount numeric	disease character varying (255)	disease_count bigint
1	arinna lifesciences	11.10	Epilepsy/Convulsion	29
2	arinna lifesciences	11.10	Schizophrenia	23
3	arinna lifesciences	11.10	Depression	23
4	ikon pharmachem	11.22	Depression	23
5	ikon pharmachem	11.22	Schizophrenia	22
6	leeford healthcare	13.24	Fungal Infections	50
7	leeford healthcare	13.24	Pain relief	35
8	leeford healthcare	13.24	Skin Infections	31
9	leeford healthcare	13.24	Ulcer/Reflux/Flatulence	22
10	leeford healthcare	13.24	Allergies	22
11	leeford healthcare	13.24	Bacterial Infections	22

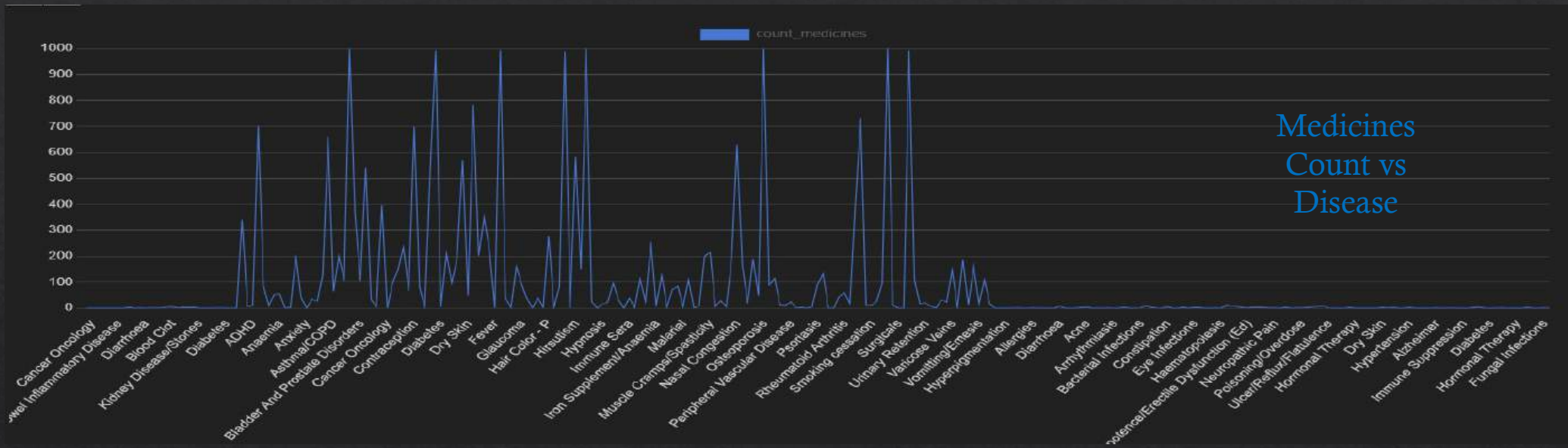
"Pharma Giants: Highest Discount Providers with 20+ Medicines per Disease"

- **Arinna Lifesciences**
  Ikon Pharmachem
  Leeford Healthcare
- **Offer the most discounts** 💰
  Manufacture 100+ medicines 🏭
  Have 20+ medicines per disease category 💊
- **How This Data Helps Netmeds:**
- **Strategic Partnerships** – Netmeds can **partner with these companies** to get **better bulk pricing & exclusive deals**.
  Targeted Discounts & Promotions – Prioritize these brands for **discount campaigns** to attract more buyers.
  Better Inventory Management – Since they manufacture **many medicines**, Netmeds can **stock more alternatives**.
  Personalized Recommendations – Show **discounted medicines** from these brands to **price-sensitive customers**.

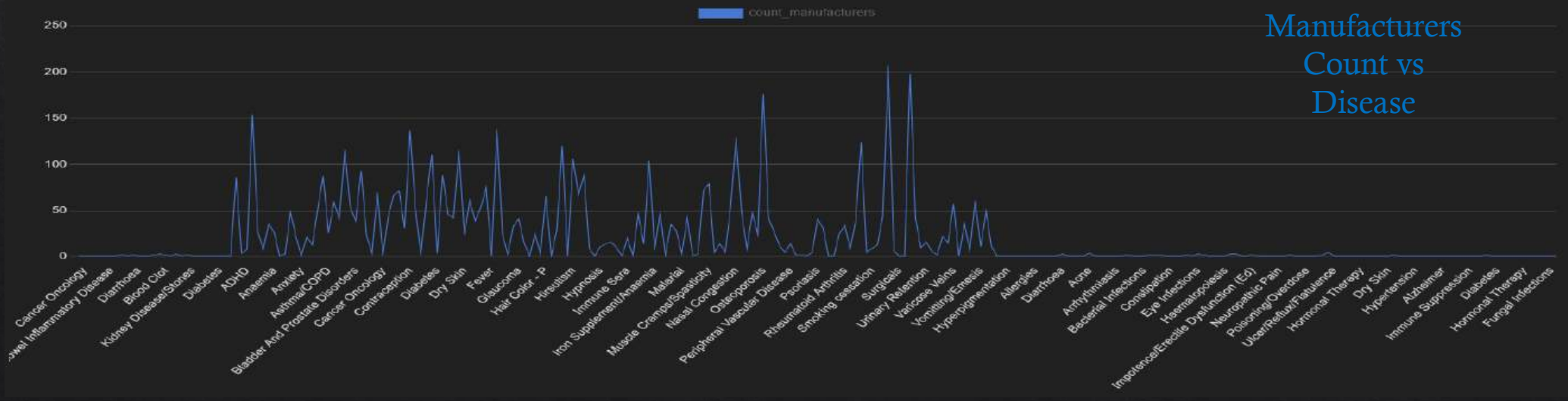
Country-wise Disease Distribution: Manufacturer Count, Average Discount, Price, and Medicine Count

```sql
WITH country_disease_info AS (
    SELECT
        n.country_of_origin,
        n.disease,
        COUNT(DISTINCT n.manufacturer) AS count_manufacturers,
        AVG(n.discount) AS avg_discount,
        AVG(n.price) AS avg_price,
        COUNT(DISTINCT n.tablet_name) AS count_medicines
    FROM netmedsdata n
    WHERE n.country_of_origin <> 'NA'
    GROUP BY n.country_of_origin, n.disease
)
SELECT
    c.country_of_origin,
    c.disease,
    c.count_manufacturers,
    ROUND(c.avg_discount, 2) AS avg_discount,
    ROUND(c.avg_price, 2) AS avg_price,
    c.count_medicines
FROM country_disease_info c
ORDER BY c.country_of_origin, c.disease;
```

country_of_origin character varying (255)	disease character varying (255)	count_manufacturers bigint	avg_discount numeric	avg_price numeric	count_medicines bigint
Australia	Cancer Oncology	1	12.00	129.00	1
Australia	Sunscreen Preparations	1	12.00	1056.00	1
Belgium	Blood Clot	1	12.00	455.00	1
Belgium	Eye conditions	1	12.00	446.00	1
Canada	Asthma/COPD	1	0.00	5486.00	1



Medicines Count vs Disease



Manufacturers Count vs Disease

Insights From Query

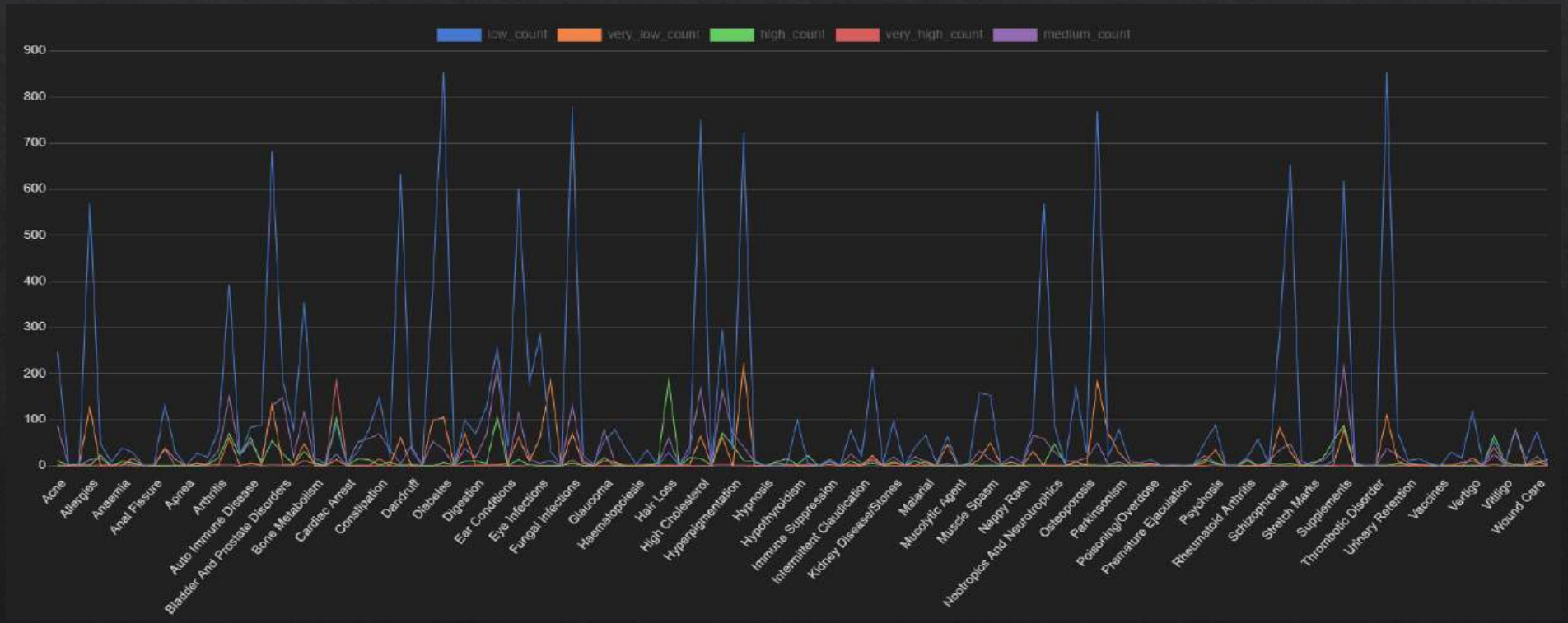
- **High Demand Chronic Conditions:** Ulcer/Reflux/Flatulence, Hypertension, High Cholesterol, Fungal Infections, Diabetes, Bacterial Infections, Asthma/COPD, Neuropathic Pain, Epilepsy are prevalent, with a large number of manufacturers and medicines available. This points to a significant customer base for these chronic conditions.
- **Rare/Acute/Specialized Conditions:** Bleeding Disorders, Hypnosis, Multiple Sclerosis, Pulmonary Hypertension, Heart Failure, Cancer Oncology have significantly fewer manufacturers and medicines, likely due to lower prevalence and potentially more specialized distribution channels.
- **Supplements Popular:** Supplements also have a good number of manufacturers and medicines, suggesting a growing interest in preventive health.
- **Pain Relief Dominance:** Pain relief has an exceptionally high number of medicines (1000), indicating a large and diverse market.

Price Distribution by Disease: Count of Medicines in Each Price Category

```sql
WITH PriceCategories AS (
    SELECT
        disease,
        CASE
            WHEN price < 50 THEN 'Very Low'
            WHEN price >= 50 AND price < 300 THEN 'Low'
            WHEN price >= 300 AND price < 600 THEN 'Medium'
            WHEN price >= 600 AND price < 3000 THEN 'High'
            ELSE 'Very High'
        END AS netmedsdata
    FROM netmedsdata
)

SELECT
    disease,
    COUNT(CASE WHEN netmedsdata = 'Low' THEN 1 END) AS low_count,
    COUNT(CASE WHEN netmedsdata = 'Medium' THEN 1 END) AS medium_count,
    COUNT(CASE WHEN netmedsdata = 'High' THEN 1 END) AS high_count,
    COUNT(CASE WHEN netmedsdata = 'Very High' THEN 1 END) AS very_high_count,
    COUNT(CASE WHEN netmedsdata = 'Very Low' THEN 1 END) AS very_low_count
FROM PriceCategories
GROUP BY disease
ORDER BY disease;
```

disease character varying (255)	low_count bigint	medium_count bigint	high_count bigint	very_high_count bigint	very_low_count bigint
Acne	247	86	10	0	1
ADHD	3	1	2	0	0
Alcohol Addiction	4	0	4	0	3
Allergies	569	13	1	0	127
Alzheimer	47	16	22	1	0

Visualization from output



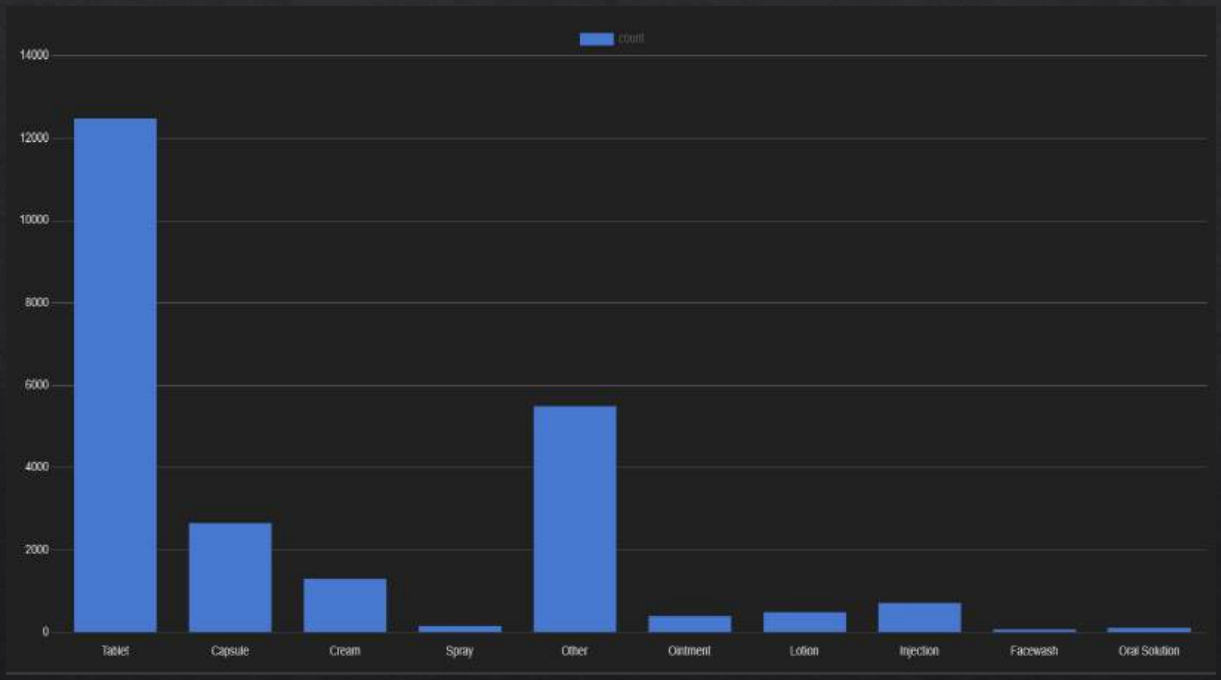
Insights from Query

- **Very High Price Category:** Cancer Oncology has the highest number of medicines (184), followed by Viral Infection and Anemia (12 each). Blood clot treatments also have a significant presence (11).
- **High Price Category:** Hair Loss dominates with 186 medicines, followed by Supplements, Hormonal Therapy, Dry Skin, and Cancer Oncology (80-120).
- **Medium Price Category:** Supplements lead with 216 medicines, followed by Asthma/COPD, Bladder/Prostate Problems, High Cholesterol, and Hormonal Therapy (120-170).
- **Low Price Category:** Ulcer/Reflux/Diabetes have the highest counts (around 850), followed by a range of chronic conditions like Neuropathic Pain, Skin Infections, Hypertension, High Cholesterol, Cough/Cold, Bacterial Infections, and Allergies (600-800).
- **Very Low Price Category:** Hypertension again leads with 223 medicines, followed by Fever and Pain Relief (around 180).

"Analysis of Medicine Types and Pricing by Disease"

```sql
WITH PriceDiscountAnalysis AS (
    SELECT
        disease,
        CASE
            WHEN "oral_solution_ml_per_tube" > 0 THEN 'Oral Solution'
            WHEN "cream_gm_per_tube" > 0 THEN 'Cream'
            WHEN "injection" > 0 THEN 'Injection'
            WHEN "tablet_per_strip" > 0 THEN 'Tablet'
            WHEN "capsule_per_strip" > 0 THEN 'Capsule'
            WHEN "gm_per_lotion" > 0 THEN 'Lotion'
            WHEN "facewash_gm_per_tube" > 0 THEN 'Facewash'
            WHEN "spray_ml_per_tube" > 0 THEN 'Spray'
            WHEN "ointment_gm_per_tube" > 0 THEN 'Ointment'
            ELSE 'Other'
        END AS product_category,

        price,
        discount
    FROM netmedsdata
)
SELECT
    disease,
    product_category,
    COUNT(*) AS product_count,
    round(AVG(price), 2) AS avg_price,
    round(AVG(discount), 2) AS avg_discount
FROM PriceDiscountAnalysis
GROUP BY disease, product_category
ORDER BY disease, product_count DESC;
```

	disease character varying (255)	product_category text	product_count bigint	avg_price numeric	avg_discount numeric
1	Acne	Other	182	233.87	8.05
2	Acne	Capsule	57	217.81	7.16
3	Acne	Cream	55	342.18	7.64
4	Acne	Ointment	14	123.86	9.86
5	Acne	Facewash	13	418.92	6.46
6	Acne	Tablet	12	183.58	8.00
7	Acne	Lotion	8	205.50	12.00
8	Acne	Spray	3	467.33	4.00



Tablets Dominate: Tablets are by far the most common type of medicine, with a count exceeding 12,000. This suggests a strong preference or market prevalence for tablet-based medications.

Patient Preference: The overwhelming preference for tablets could be due to factors like ease of consumption, convenience, established habits, or cost-effectiveness.

"Ranking Companies by Average Price within Each Country of Origin"

```sql
WITH CompanyAvgPrice AS (
    SELECT
        country_of_origin,
        company_type,
        AVG(price) AS avg_price
    FROM netmedsdata
    GROUP BY country_of_origin, company_type
),
RankedCompanies AS (
    SELECT
        country_of_origin,
        company_type,
        avg_price,
        RANK() OVER (PARTITION BY country_of_origin ORDER BY avg_price DESC) AS company_rank
    FROM CompanyAvgPrice
)
SELECT
    country_of_origin,
    company_type,
    round(avg_price,2),
    company_rank
FROM RankedCompanies
ORDER BY country_of_origin, company_rank DESC;
```

	country_of_origin character varying (255)	company_type character varying (255)	round numeric	company_rank bigint
1	Australia	limited	129.00	2
2	Australia	pvt limited	1056.00	1
3	Belgium	pvt limited	446.00	2
4	Belgium	limited	455.00	1
5	Canada	limited	5486.00	1
6	Czech Republic	pvt limited	247.00	1
7	France	serbia pharmaceuticals	339.00	4
8	France	limited	569.25	3
9	France	pvt limited	1143.00	2
10	France	other	1656.33	1

1."Medicine Count by Country of Origin and Disease“ and Medicine Count by Manufacturer and Disease"

```sql
SELECT
    country_of_origin,
    disease,
    COUNT(tablet_name) AS medicine_count
FROM netmedsdata
WHERE country_of_origin != 'NA'
GROUP BY country_of_origin, disease
ORDER BY country_of_origin, medicine_count DESC;
```

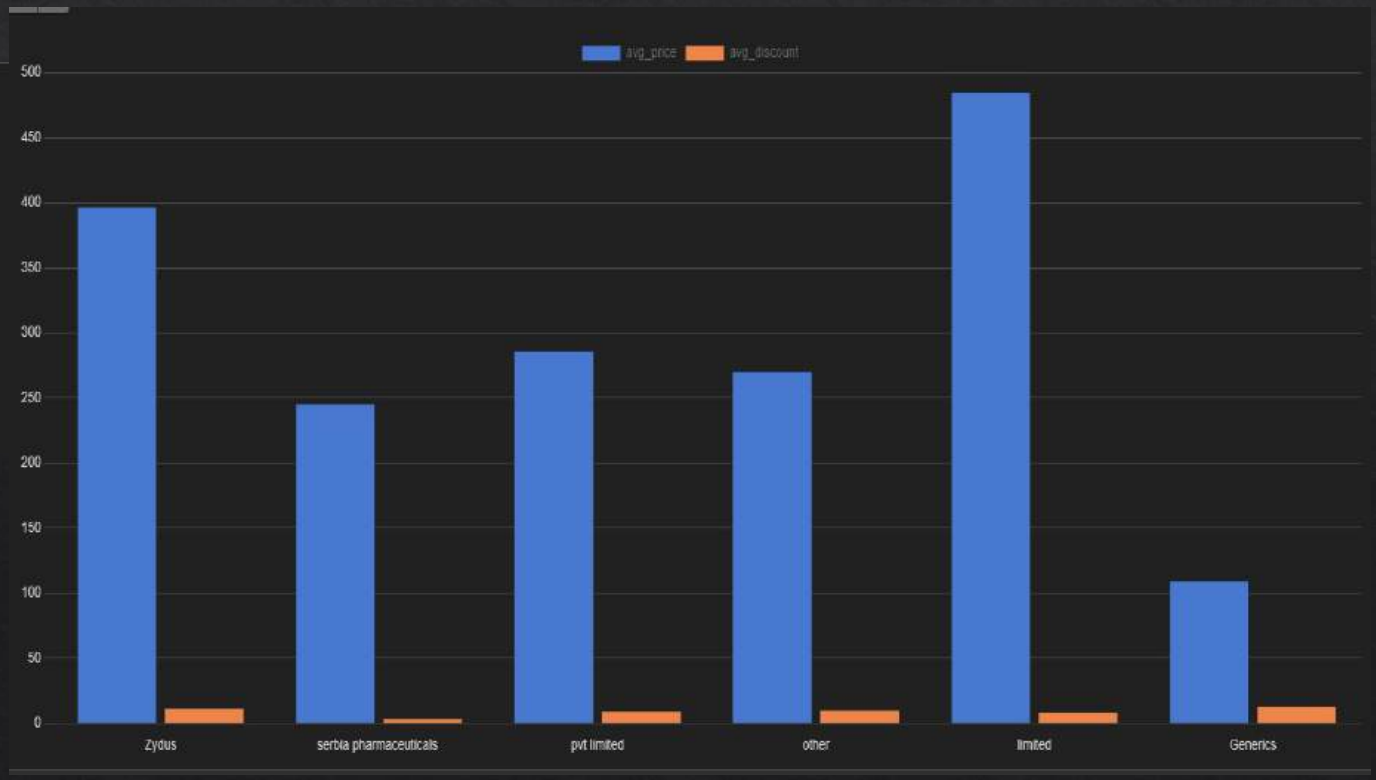
country_of_origin character varying (255)	disease character varying (255)	medicine_count bigint
Australia	Cancer Oncology	1
Australia	Sunscreen Preparations	1
Belgium	Blood Clot	1
Belgium	Eye conditions	1
Canada	Asthma/COPD	1
Czech Republic	Ulcerative Colitis/Bowel Inflammatory Disease	1
France	Blood Clot	4

```sql
SELECT
    manufacturer,
    disease,
    COUNT(tablet_name) AS medicine_count
FROM netmedsdata
WHERE manufacturer IS NOT NULL AND manufacturer != 'NA'
GROUP BY manufacturer, disease
ORDER BY manufacturer, medicine_count DESC;
```

manufacturer character varying (255)	disease character varying (255)	medicine_count bigint
4 care life science	Epilepsy/Convulsion	1
a n pharmacia laboratories	Depression	1
aagam life sciences	Blood Clot	1
aagam life sciences	Allergies	1
aakash pharmaceuticals	Cleanser	1
aarohi pharmaceuticals	Iron Supplement/Anaemia	1
aarux pharmaceuticals	Supplements	2
aarux pharmaceuticals	Varicose Veins	2

"Company-wise Product Summary: Count, Average Price, and Average Discount Analysis"

```sql
SELECT
    company_type,
    COUNT(*) AS count,
    ROUND(AVG(price), 2) AS avg_price,
    ROUND(AVG(discount), 2) AS avg_discount
FROM
    netmedsdata
GROUP BY
    company_type;
```



Project Name - PharmaOptimizer: Data-Driven Pricing Analysis for Smart Pharma Strategies

Project Summary -

Pharmaceutical Price Prediction – Data-Driven Pricing Optimization Project Overview The pharmaceutical industry relies on accurate pricing strategies to balance profitability and affordability. This project leverages data analytics and machine learning to optimize medicine pricing by analyzing key attributes such as drug composition, dosage, and packaging. By employing predictive modeling, the project aims to enhance decision-making for manufacturers, retailers, and consumers, ensuring competitive pricing while maintaining profitability.

Price Optimization & Prediction Optimizing pharmaceutical prices requires understanding various factors influencing costs. This project applies regression models like:

Linear Regression

Random Forest Regressor

XGBoost Regressor

LightGBM Regressor

The models considered multiple pricing factors, including:

 Active ingredients & dosage

 Number of tablets per strip

 Manufacturer details

 Market trends & pricing variations

Through rigorous model evaluation using RMSE, R^2 Score, and MAE, the Random Forest Regressor emerged as the best model, offering reliable price predictions and actionable insights for stakeholders.

Tools & Techniques The project was developed using Python and leading data science libraries, including:

Pandas & NumPy for data preprocessing

Scikit-learn, XGBoost & LightGBM for model building

Matplotlib & Seaborn for data visualization

RandomizedSearchCV for hyperparameter tuning

Data preprocessing involved handling missing values, encoding categorical

variables, and applying log transformation to normalize price distributions.

Key Business Impact This project delivers substantial value by enabling:

 Informed Pricing Strategies – Helping manufacturers and retailers optimize product prices.

 Competitive Market Positioning – Adjusting prices based on data-driven insights.

 Targeted Marketing & Inventory Planning – Segmenting products for effective marketing campaigns.

 Anomaly Detection – Identifying unexpected pricing variations or errors.

 Consumer Benefits – Ensuring affordability through fair pricing and cost predictions.

GitHub Link -

Provide your GitHub Link here.

Problem Statement

The pharmaceutical industry faces complex pricing challenges due to factors such as raw material costs, market demand, competition, government regulations, and regional price variations. Inefficient pricing strategies can lead to profit loss, overpricing, reduced accessibility, and compliance issues.

This project aims to address these challenges by leveraging advanced data analytics and machine learning techniques to develop solutions that enable:

Price Optimization: Predicting optimal prices for pharmaceutical products based on their attributes.

Let's Begin !

1. Know Your Data

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.cluster import KMeans
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

import xgboost as xgb
import lightgbm as lgb
import catboost as cb
import pickle
```

Dataset Loading

```
file_path="/content/web scraping data 2.xlsx"
data = pd.read_excel(file_path, sheet_name="Data")
```

```
/usr/local/lib/python3.11/dist-packages/openpyxl/worksheet/
_reader.py:329: UserWarning: Unknown extension is not supported and
will be removed
  warn(msg)
```

Dataset First View

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23897 entries, 0 to 23896
Data columns (total 18 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Tablet Name                 23897 non-null  object 
 1   Disease                     23897 non-null  object 
 2   Rx Required                 23897 non-null  int64  
 3   Price                       23897 non-null  int64  
 4   Discount                    23897 non-null  int64  
 5   Country of Origin           23897 non-null  object 
 6   Manufacturer                23897 non-null  object 
 7   Tablet Info                 23897 non-null  object 
 8   Gm per lotion               23896 non-null  float64
 9   Facewash - Gm per tube      23897 non-null  int64  
 10  Spray- Ml per tube          23888 non-null  float64
 11  Ointment - Gm per tube.1    23897 non-null  int64  
 12  Oral solution-Ml per tube   23897 non-null  int64  
 13  Cream-Gm per tube           23888 non-null  float64
 14  Injection                   23878 non-null  float64
 15  Tablet per strip            23890 non-null  float64
 16  capsule per strip           23895 non-null  float64
 17  Comapny Type                23897 non-null  object 
dtypes: float64(6), int64(6), object(6)
memory usage: 3.3+ MB


data
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 23897,\n
\"fields\": [\n    {\n      \"column\": \"Tablet Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 23897,\n        \"samples\": [\n
\"BIOKER + Tablet 10's\",\n          \"PANBRIT DSR Capsule 10's\",\n
\"LEVOSALVAC 0.63 Respules 5X2.5ml\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Disease\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
140,\n        \"samples\": [\n          \"Psychosis\",\n
\"Hyponatremia\",\n          \"Contraception\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Rx Required\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Price\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 2302,\n        \"min\": 1,\n        \"max\": 136503,\n
\"num_unique_values\": 1544,\n        \"samples\": [\n          2142,\
```

```
n          599\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Discount\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5,\n        \"min\": 0,\n
\"max\": 45,\n        \"num_unique_values\": 9,\n        \"samples\":
[\n          35,\n          12\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Country of Origin\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 29,\n
\"samples\": [\n          \" Indonesia\",\n          \" Serbia\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Manufacturer\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 636,\n        \"samples\": [\n
\"biomiicron pharma india \",\n          \"raybros medicaments \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Tablet Info\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 6388,\n        \"samples\": [\n
\"Chlorhexidine 0.2 %+Silver Sulphadiazine 1 %\",\n          \"Omega 3
Fatty Acid 74.8 mg+Retinol 170 IU+Tocopherol 4 IU+Zinc 0.4 mg\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Gm per lotion\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
16.59956737244447,\n        \"min\": 0.0,\n        \"max\": 500.0,\n
\"num_unique_values\": 33,\n        \"samples\": [\n          6.0,\n
300.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Facewash - Gm per tube\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 5,\n        \"min\": 0,\n
\"max\": 250,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          70,\n          50\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Spray- Ml per tube\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4.742684184652439,\n        \"min\": 0.0,\n        \"max\": 200.0,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          200.0,\n
120.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Ointment - Gm per tube.1\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3,\n        \"min\": 0,\n
\"max\": 250,\n        \"num_unique_values\": 14,\n
\"samples\": [\n          100,\n          60\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Oral solution-Ml per tube\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
13,\n        \"min\": 0,\n        \"max\": 500,\n
\"num_unique_values\": 17,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
```

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Cream-Gm per tube\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 11.85588894226156,\n        \"min\":
0.0,\n        \"max\": 500.0,\n        \"num_unique_values\": 31,\n
\"samples\": [\n          240.0,\n          200.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Injection\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4.047844444703396,\n        \"min\": 0.0,\n        \"max\": 300.0,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          20.0,\n
43.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Tablet per strip\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 83.53943451589862,\n        \"min\":
0.0,\n        \"max\": 4500.0,\n        \"num_unique_values\": 39,\n
\"samples\": [\n          4500.0,\n          31.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"capsule per strip\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4.154483225215704,\n        \"min\": 0.0,\n        \"max\": 120.0,\n
\"num_unique_values\": 25,\n        \"samples\": [\n          5.0,\n
1.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Comapny Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          \"pvt limited\",\n          \"limited\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```


Dataset Rows & Columns count

```
rows, columns = data.shape

# Print the number of rows and columns
print(f"Number of rows: {rows}")
print(f"Number of columns: {columns}")
```

```
Number of rows: 23897
Number of columns: 18
```

Dataset Information

```
# Dataset Info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23897 entries, 0 to 23896
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
```

```
---  ------                        --------------  -----
 0   Tablet Name                   23897 non-null  object
 1   Disease                       23897 non-null  object
 2   Rx Required                   23897 non-null  int64
 3   Price                         23897 non-null  int64
 4   Discount                      23897 non-null  int64
 5   Country of Origin             23897 non-null  object
 6   Manufacturer                  23897 non-null  object
 7   Tablet Info                   23897 non-null  object
 8   Gm per lotion                 23896 non-null  float64
 9   Facewash - Gm per tube        23897 non-null  int64
 10  Spray- Ml per tube            23888 non-null  float64
 11  Ointment - Gm per tube.1      23897 non-null  int64
 12  Oral solution-Ml per tube     23897 non-null  int64
 13  Cream-Gm per tube             23888 non-null  float64
 14  Injection                     23878 non-null  float64
 15  Tablet per strip              23890 non-null  float64
 16  capsule per strip             23895 non-null  float64
 17  Comapny Type                  23897 non-null  object
dtypes: float64(6), int64(6), object(6)
memory usage: 3.3+ MB
```

Duplicate Values

```
# Dataset Duplicate Value Count
duplicate_count = data.duplicated().sum()

print(f"Number of duplicate rows: {duplicate_count}")

Number of duplicate rows: 0
```

Missing Values/Null Values

```
# Missing Values/Null Values Count
missing_values_count = data.isnull().sum()

print("Missing values count for each column:")
print(missing_values_count)

total_missing_values = data.isnull().sum().sum()
print(f"Total number of missing values: {total_missing_values}")

Missing values count for each column:
Tablet Name                   0
Disease                       0
Rx Required                   0
Price                         0
Discount                      0
Country of Origin             0
Manufacturer                  0
```

```
Tablet Info                    0
Gm per lotion                  1
Facewash - Gm per tube         0
Spray- Ml per tube             9
Ointment - Gm per tube.1       0
Oral solution-Ml per tube      0
Cream-Gm per tube              9
Injection                     19
Tablet per strip               7
capsule per strip              2
Comapny Type                   0
dtype: int64
Total number of missing values: 47
```

What did you know about your dataset?

Total Rows and Columns:

The dataset has 23,897 rows and 8 columns. Data Columns and Types:

Tablet Name: Object (string) type, with no missing values. Disease: Object (string) type, with no missing values. Rx Required: Boolean (True/False) type, with no missing values. Price: Integer type, with no missing values. Discount: Integer type, with no missing values. Country of Origin: Object (string) type, with no missing values. Manufacturer: Object (string) type, with no missing values. Tablet Info: Object (string) type, with no missing values. Missing Values:

There are no missing (null) values in any of the columns, as the count of non-null values in all columns matches the total number of rows. Duplicate Rows:

The dataset has no duplicate rows. Data Integrity:

The dataset is clean in terms of missing data and duplicates, making it suitable for analysis or model training without needing to address those issues. Summary: The dataset appears to be well-structured with no missing values or duplicate records, making it easy to proceed with analysis, feature engineering, or model training without any immediate data cleaning tasks related to nulls or duplicates.

2. Understanding Your Variables

```
data.columns
```

```
Index(['Tablet Name', 'Disease', 'Rx Required', 'Price', 'Discount',
       'Country of Origin', 'Manufacturer', 'Tablet Info', 'Gm per
lotion',
       'Facewash - Gm per tube', 'Spray- Ml per tube',
       'Ointment - Gm per tube.1', 'Oral solution-Ml per tube',
       'Cream-Gm per tube', 'Injection', 'Tablet per strip',
       'capsule per strip', 'Comapny Type'],
      dtype='object')
```

```
# Dataset Describe
data.describe(include=['object',"bool"])
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 4,\n  \"fields\": [\
n    {\n      \"column\": \"Tablet Name\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"23897\",\n          \"Cogniza Tablet
10'S\",\n          \"1\"\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Disease\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          140,\n          \"1000\",\n          \"23897\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Country of Origin\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          29,\n          \"23615\",\n          \"23897\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Manufacturer\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          636,\n          \"1034\",\n          \"23897\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Tablet Info\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          6388,\n          \"200\",\n          \"23897\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Comapny Type\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 4,\n        \"samples\":
[\n          6,\n          \"11472\",\n          \"23897\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Variables Description

Tablet Name: Type: Object (String) Description: This column contains the name of the pharmaceutical tablet or product. It helps identify the product in the dataset. Example Values: "Cogniza Tablet 10'S", "Atrest 25mg Tablet 10'S", etc.

Disease: Type: Object (String) Description: This column indicates the disease or condition that the tablet is used to treat. It helps in understanding the therapeutic category of the tablet. Example Values: "ADHD", "Anxiety", etc.

Rx Required: Type: Boolean (True/False) Description: This column indicates whether a prescription is required to purchase the tablet. It helps identify whether the product is over-the-counter (OTC) or prescription-based. Example Values: True (if a prescription is required), False (if no prescription is required).

Price: Type: Integer Description: This column represents the price of the tablet in monetary units (e.g., INR, USD). It helps in understanding the cost associated with the product. Example Values: 239, 336, 276, etc

Discount: Type: Integer Description: This column indicates the discount offered on the product. It could be a percentage or fixed value depending on how it's structured. Example Values: 0, 12, etc.

Country of Origin: Type: Object (String) Description: This column represents the country where the pharmaceutical tablet is manufactured. It provides insight into the geographical origin of the product, which could impact pricing and availability. Example Values: "India", "USA", etc

Manufacturer: Type: Object (String) Description: This column contains the name of the company that manufactures the tablet. It is useful for understanding brand reputation and potential variations in pricing or quality associated with different manufacturers. Example Values: "Linux Laboratories Pvt Ltd", "Centaur Pharmaceuticals Pvt Ltd", etc

Tablet Info: Type: Object (String) Description: This column provides additional details about the tablet, such as its composition, dosage, and form (e.g., "Cerebroprotein Hydrolysate 90 mg", "Tetrabenazine 25 mg"). Example Values: "Cerebroprotein Hydrolysate 90 mg", "Tetrabenazine 25 mg", etc.

The dataset contains a mix of categorical (string) and numerical (integer, boolean) variables. Categorical variables like "Tablet Name," "Disease," "Country of Origin," "Manufacturer," and "Tablet Info" provide information about the product's identity, origin, and composition. Numerical variables like "Price" and "Discount" help quantify product characteristics. Boolean variable "Rx Required" helps to determine if a prescription is needed for the product.

Check Unique Values for each variable.

```
# Checking Unique Values for each variable.
unique_values = data.nunique()
print(unique_values)
```

```
Tablet Name                  23897
Disease                        140
Rx Required                      2
Price                         1544
Discount                         9
Country of Origin               29
Manufacturer                   636
Tablet Info                   6388
Gm per lotion                   33
Facewash - Gm per tube          10
Spray- Ml per tube              24
Ointment - Gm per tube.1        14
Oral solution-Ml per tube       17
Cream-Gm per tube               31
Injection                       24
Tablet per strip                39
capsule per strip               25
```

```
Comapny Type                        6
dtype: int64
```

4. Data Vizualization, Storytelling & Experimenting with charts : Understand the relationships between variables

Chart - 1

```
# Counting the number of medicines for each disease
disease_counts = data['Disease'].value_counts()

# Sorting the disease counts in descending order
disease_counts = disease_counts.sort_values(ascending=False)

plt.figure(figsize=(24, 12))
sns.barplot(x=disease_counts.index, y=disease_counts.values,
palette='viridis')

plt.xlabel('Disease', fontsize=12, weight='bold', color='black')
plt.ylabel('Number of Medicines', fontsize=12, weight='bold',
color='black')
plt.title('Count of Medicines for Each Disease', fontsize=20,
weight='bold', color='black')

plt.xticks(rotation=90, ha='right', fontsize=10, weight='bold',
color='black')
plt.yticks(fontsize=10, weight='bold', color='black')

plt.tight_layout()
plt.show()
```

```
<ipython-input-12-23027694d868>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=disease_counts.index, y=disease_counts.values,
palette='viridis')
```

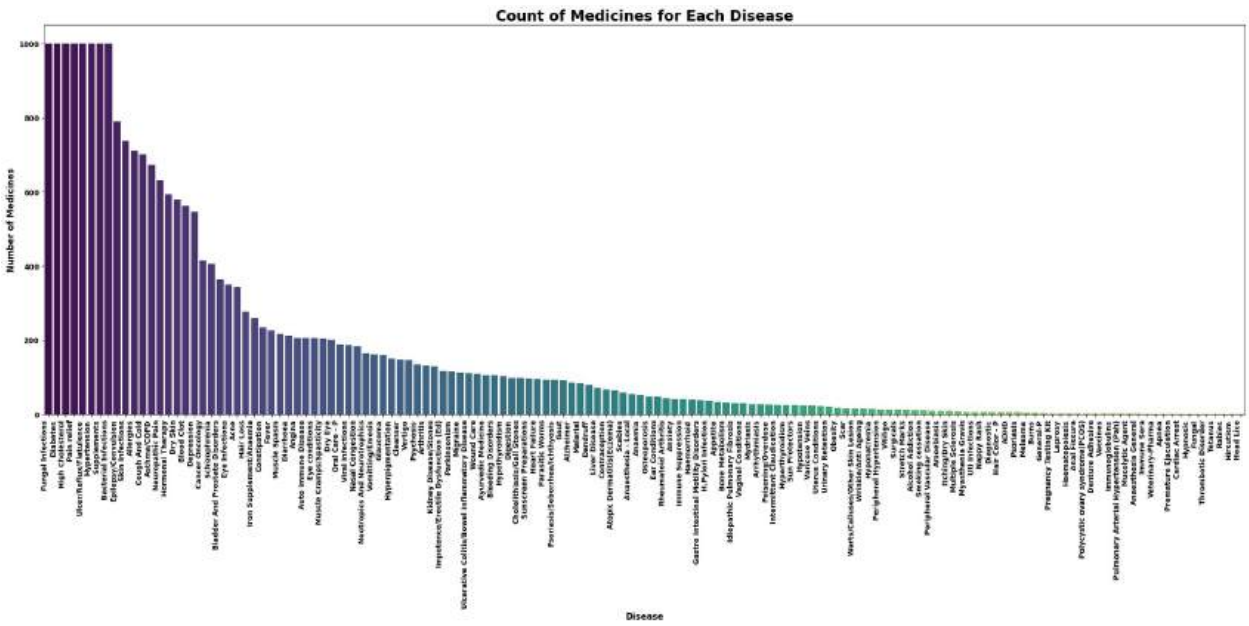


Chart - 2

```
# Count the number of medicines for each 'Rx Required' category
rx_required_counts = data['Rx Required'].value_counts()

plt.figure(figsize=(8, 6))
sns.barplot(x=rx_required_counts.index, y=rx_required_counts.values,
palette='viridis')

plt.xlabel('Prescription Required', fontsize=12)
plt.ylabel('Number of Medicines', fontsize=12)
plt.title('Count of Medicines that Require or Do Not Require
Prescription', fontsize=15)
plt.xticks([0, 1], ['No', 'Yes'], rotation=0)
plt.tight_layout()
plt.show()
```

```
<ipython-input-13-3218043db870>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=rx_required_counts.index, y=rx_required_counts.values,
palette='viridis')
```

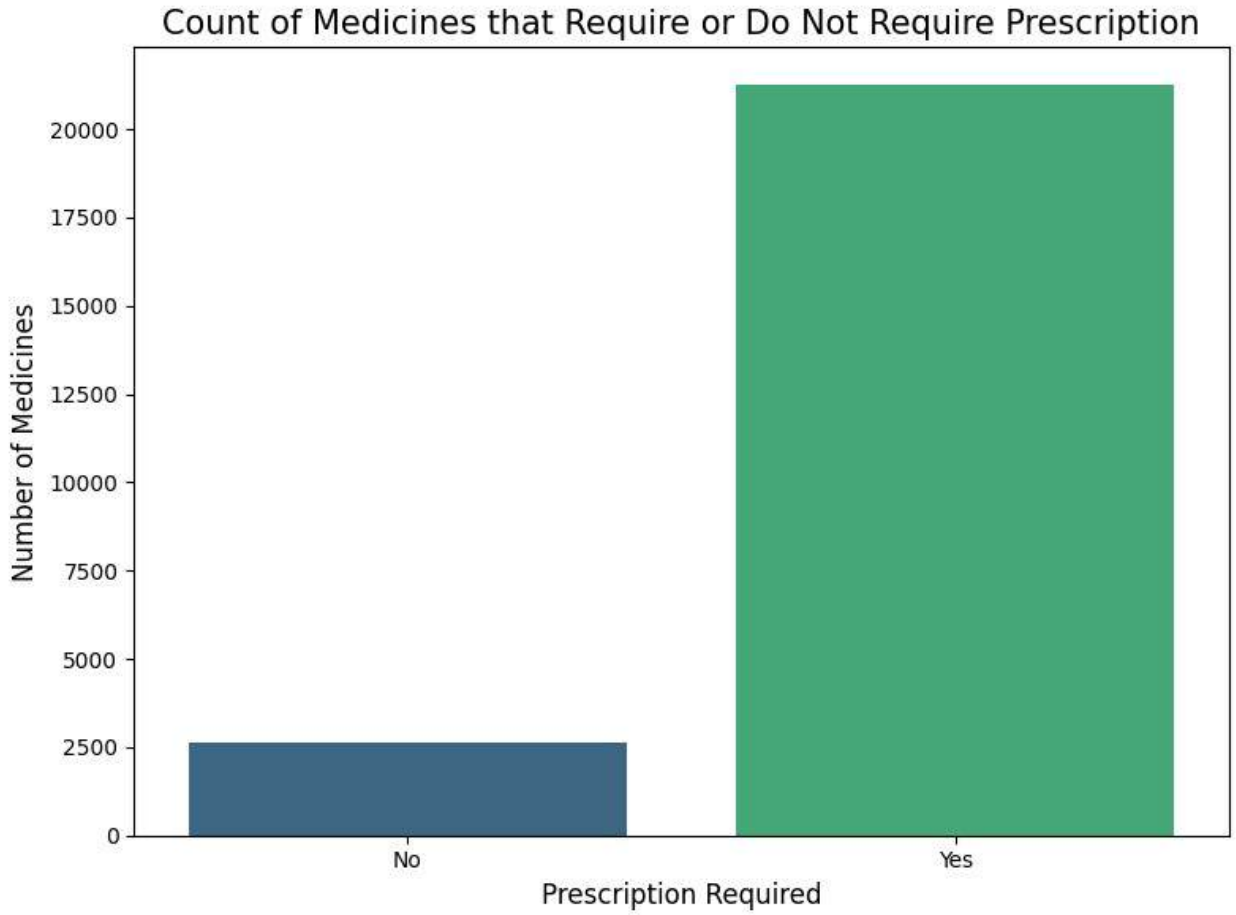


Chart - 3

```
# Group by Disease and Rx Required for  geting the count of medicines
for each combination
disease_rx_counts = data.groupby(['Disease', 'Rx
Required']).size().reset_index(name='Medicine Count')

# Separating diseases with Rx Required as True and False
disease_rx_true = disease_rx_counts[disease_rx_counts['Rx Required']
== True].sort_values(by='Medicine Count', ascending=False)
disease_rx_false = disease_rx_counts[disease_rx_counts['Rx Required']
== False].sort_values(by='Medicine Count', ascending=False)

# Top 10 diseases for Rx Required = True
top_diseases_rx_true = disease_rx_true.head(10)

# Bottom 10 diseases for Rx Required = True
bottom_diseases_rx_true = disease_rx_true.tail(10)

# Top 10 diseases for Rx Required = False
top_diseases_rx_false = disease_rx_false.head(10)
```

```
# Bottom 10 diseases for Rx Required = False
bottom_diseases_rx_false = disease_rx_false.tail(10)

fig, axes = plt.subplots(2, 2, figsize=(18, 12))

sns.barplot(x='Medicine Count', y='Disease',
data=top_diseases_rx_true, ax=axes[0, 0], palette='Blues')
axes[0, 0].set_title('Top Diseases with Prescription Required',
fontsize=15)
axes[0, 0].set_xlabel('Medicine Count', fontsize=12)
axes[0, 0].set_ylabel('Disease', fontsize=12)

sns.barplot(x='Medicine Count', y='Disease',
data=bottom_diseases_rx_true, ax=axes[0, 1], palette='Blues_d')
axes[0, 1].set_title('Bottom Diseases with Prescription Required',
fontsize=15)
axes[0, 1].set_xlabel('Medicine Count', fontsize=12)
axes[0, 1].set_ylabel('Disease', fontsize=12)

sns.barplot(x='Medicine Count', y='Disease',
data=top_diseases_rx_false, ax=axes[1, 0], palette='Oranges')
axes[1, 0].set_title('Top Diseases without Prescription Required',
fontsize=15)
axes[1, 0].set_xlabel('Medicine Count', fontsize=12)
axes[1, 0].set_ylabel('Disease', fontsize=12)

sns.barplot(x='Medicine Count', y='Disease',
data=bottom_diseases_rx_false, ax=axes[1, 1], palette='Oranges_d')
axes[1, 1].set_title('Bottom Diseases without Prescription Required',
fontsize=15)
axes[1, 1].set_xlabel('Medicine Count', fontsize=12)
axes[1, 1].set_ylabel('Disease', fontsize=12)

plt.tight_layout()
plt.show()
```

```
<ipython-input-14-b9dae12666ad>:22: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Medicine Count', y='Disease',
data=top_diseases_rx_true, ax=axes[0, 0], palette='Blues')
<ipython-input-14-b9dae12666ad>:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(x='Medicine Count', y='Disease',
data=bottom_diseases_rx_true, ax=axes[0, 1], palette='Blues_d')
<ipython-input-14-b9dae12666ad>:32: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Medicine Count', y='Disease',
data=top_diseases_rx_false, ax=axes[1, 0], palette='Oranges')
<ipython-input-14-b9dae12666ad>:37: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Medicine Count', y='Disease',
data=bottom_diseases_rx_false, ax=axes[1, 1], palette='Oranges_d')
```

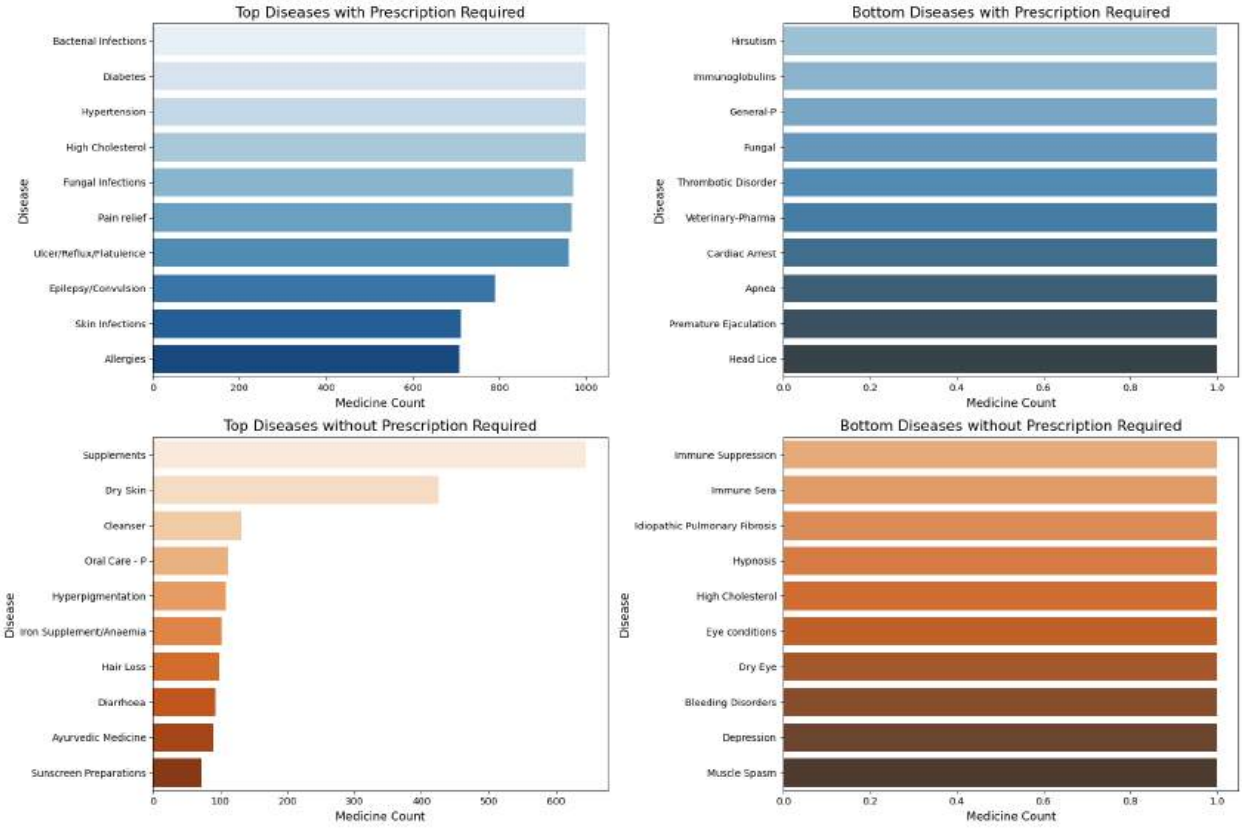


Chart - 4

```
# Grouping by Disease and calculating the average Discount
disease_discount = data.groupby('Disease')
```

```
['Discount'].mean().reset_index()

# Sorting the diseases by Discount in descending order
disease_discount_sorted = disease_discount.sort_values(by='Discount',
ascending=False)

plt.figure(figsize=(16, 24))
sns.barplot(x='Discount', y='Disease', data=disease_discount_sorted,
palette='coolwarm')

plt.xlabel('Average Discount', fontsize=14, fontweight='bold')
plt.ylabel('Disease', fontsize=14, fontweight='bold')
plt.title('Diseases with Highest Average Discount', fontsize=16,
fontweight='bold')

plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
<ipython-input-15-0a75fe012354>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Discount', y='Disease', data=disease_discount_sorted,
palette='coolwarm')
```

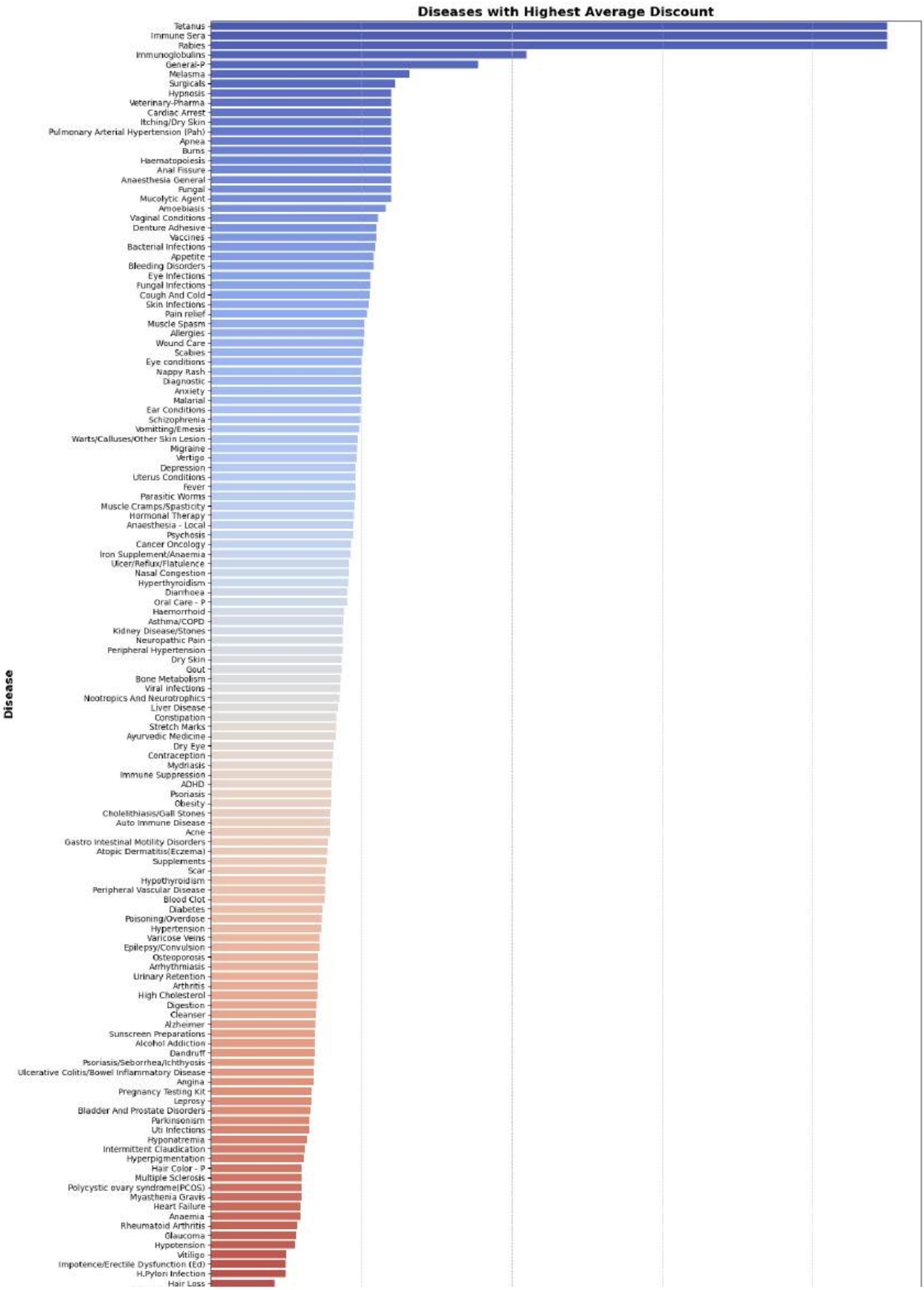

Diseases with Highest Average Discount
Disease
Tetanus
Immune Sera
Rabies
Immunoglobulins
General-P
Melasma
Surgicals
Hypnosis
Veterinary-Pharma
Cardiac Arrest
Itching/Dry Skin
Pulmonary Arterial Hypertension (Pah)
Apnea
Burns
Haematopoiesis
Anal Fissure
Anaesthesia General
Fungal
Mucolytic Agent
Amoebiasis
Vaginal Conditions
Denture Adhesive
Vaccines
Bacterial Infections
Appetite
Bleeding Disorders
Eye Infections
Fungal Infections
Cough And Cold
Skin Infections
Pain relief
Muscle Spasm
Allergies
Wound Care
Scabies
Eye conditions
Nappy Rash
Diagnostic
Anxiety
Malarial
Ear Conditions
Schizophrenia
Vomitting/Emesis
Warts/Calluses/Other Skin Lesion
Migraine
Vertigo
Depression
Uterus Conditions
Fever
Parasitic Worms
Muscle Cramps/Spasticity
Hormonal Therapy
Anaesthesia - Local
Psychosis
Cancer Oncology
Iron Supplement/Anaemia
Ulcer/Reflux/Flatulence
Nasal Congestion
Hyperthyroidism
Diarrhoea
Oral Care - P
Haemorrhoid
Asthma/COPD
Kidney Disease/Stones
Neuropathic Pain
Peripheral Hypertension
Dry Skin
Gout
Bone Metabolism
Viral Infections
Neotropics And Neurotrophics
Liver Disease
Constipation
Stretch Marks
Ayurvedic Medicine
Dry Eye
Contraception
Mydriasis
Immune Suppression
ADHD
Psoriasis
Obesity
Cholelithiasis/Gall Stones
Auto Immune Disease
Acne
Gastro Intestinal Motility Disorders
Atopic Dermatitis(Eczema)
Supplements
Scar
Hypothyroidism
Peripheral Vascular Disease
Blood Clot
Diabetes
Poisoning/Overdose
Hypertension
Varicose Veins
Epilepsy/Convulsion
Osteoporosis
Arrhythmiasis
Urinary Retention
Arthritis
High Cholesterol
Digestion
Cleanser
Alzheimer
Sunscreen Preparations
Alcohol Addiction
Dandruff
Psoriasis/Seborrhea/Ichthyosis
Ulcerative Colitis/Bowel Inflammatory Disease
Angina
Pregnancy Testing Kit
Leprosy
Bladder And Prostate Disorders
Parkinsonism
Uti Infections
Hyponatremia
Intermittent Claudication
Hyperpigmentation
Hair Color - P
Multiple Sclerosis
Polycystic ovary syndrome(PCOS)
Myasthenia Gravis
Heart Failure
Anaemia
Rheumatoid Arthritis
Glaucoma
Hypotension
Vitiligo
Impotence/Erectile Dysfunction (Ed)
H.Pylori Infection
Hair Loss

Chart - 5

```
# Grouping by 'Disease' and 'Manufacturer', and calculating the
average discount
disease_manufacturer_discount = data.groupby(['Disease',
'Manufacturer'])['Discount'].mean().reset_index()

# Sorting the data based on 'Discount' in descending order
sorted_discount =
disease_manufacturer_discount.sort_values(by='Discount',
ascending=False)

# Selecting the top 10 manufacturers with the highest discounts
top_10_discount = sorted_discount.head(10)

plt.figure(figsize=(18, 10))
sns.barplot(x='Disease', y='Discount', hue='Manufacturer',
data=top_10_discount, palette='tab20')

plt.xlabel('Disease', fontsize=14, fontweight='bold')
plt.ylabel('Average Discount', fontsize=14, fontweight='bold')
plt.title('Top 10 Manufacturers with Highest Average Discounts for
Each Disease', fontsize=16, fontweight='bold')

plt.xticks(rotation=90, ha='right', fontsize=12)

plt.legend(title='Manufacturer', loc='upper left', bbox_to_anchor=(1,
1), fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

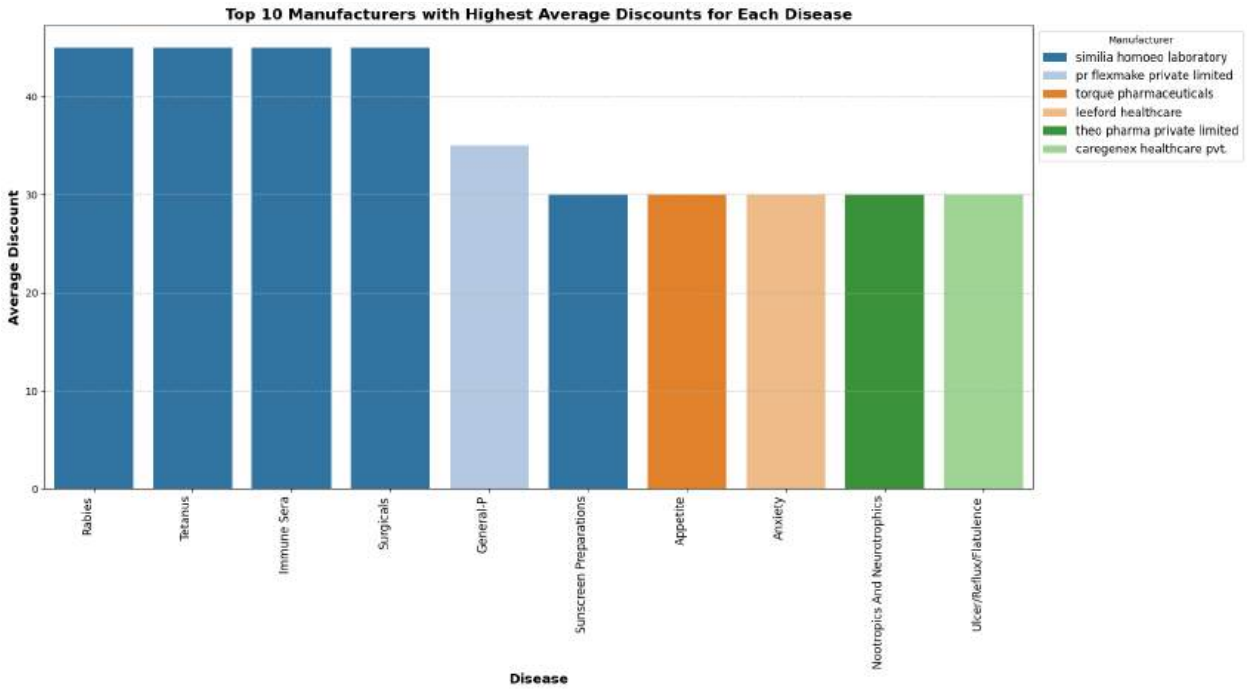


Chart - 6

```
# Grouping by 'Disease' and calculating the average price for each
disease
average_price_by_disease = data.groupby('Disease')
['Price'].mean().reset_index()

# Sorting the results in descending order to show the diseases with
the highest average price
average_price_by_disease_sorted =
average_price_by_disease.sort_values(by='Price', ascending=False)

plt.figure(figsize=(16, 20))
sns.barplot(x='Price', y='Disease',
data=average_price_by_disease_sorted, palette='viridis')

plt.xlabel('Average Price', fontsize=14, fontweight='bold')
plt.ylabel('Disease', fontsize=14, fontweight='bold')
plt.title('Average Price by Disease (Horizontal)', fontsize=16,
fontweight='bold')

plt.grid(axis='x', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

```
<ipython-input-17-37a6e060948e>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Price', y='Disease',
data=average_price_by_disease_sorted, palette='viridis')
```

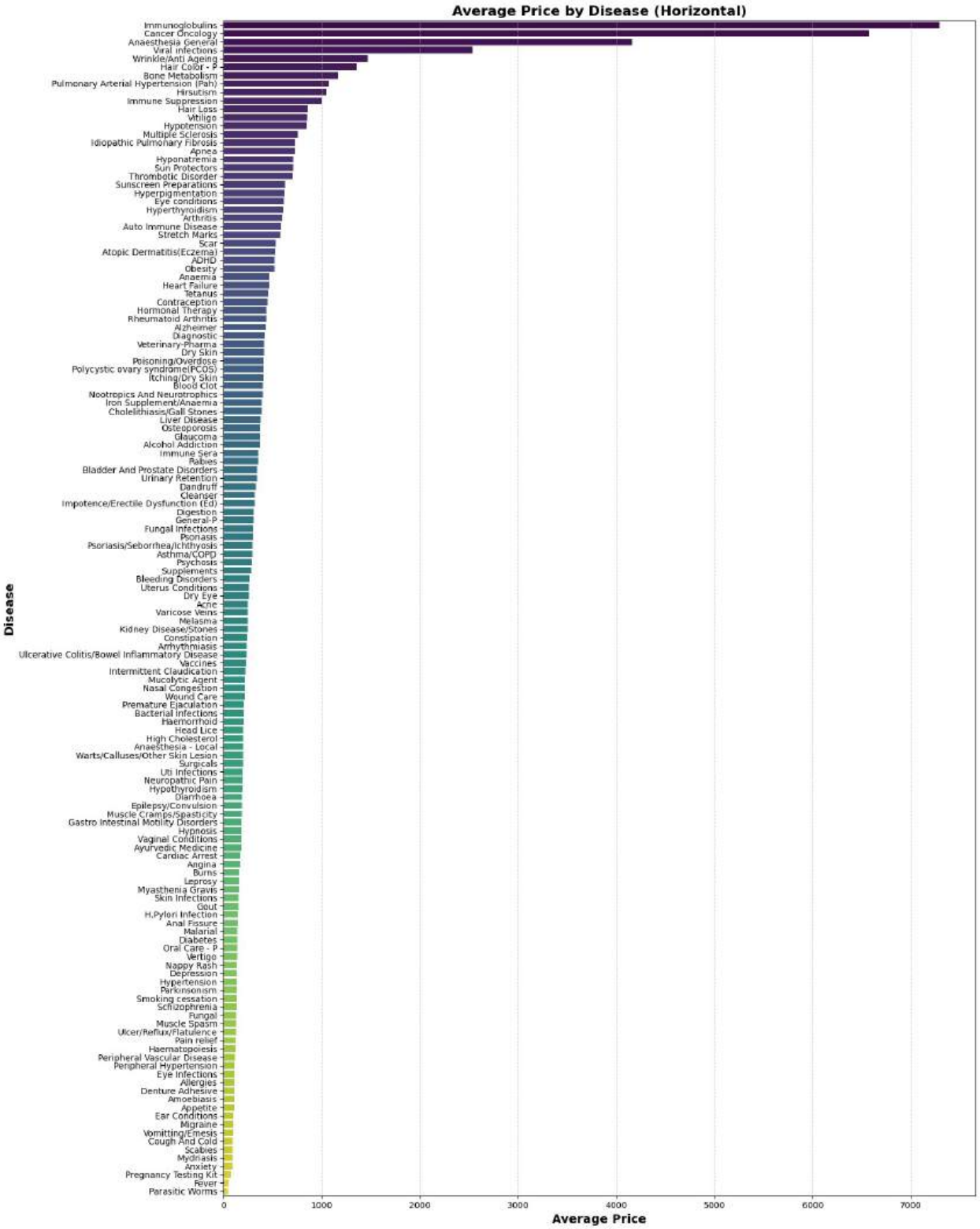


Chart - 7

```
country_price_discount = data.groupby('Country of Origin')[['Price',
'Discount']].mean().reset_index()
```

```
fig, axes = plt.subplots(1, 2, figsize=(18, 8), sharey=False)

sns.barplot(x='Country of Origin', y='Price',
data=country_price_discount, palette='Blues_d', ax=axes[0])
axes[0].set_xlabel('Country of Origin', fontsize=12,
fontweight='bold')
axes[0].set_ylabel('Average Price', fontsize=12, fontweight='bold')
axes[0].set_title('Average Price by Country of Origin', fontsize=15,
fontweight='bold')
axes[0].tick_params(axis='x', rotation=90)
axes[0].grid(axis='y', linestyle='--', alpha=0.7)

sns.barplot(x='Country of Origin', y='Discount',
data=country_price_discount, palette='Oranges_d', ax=axes[1])
axes[1].set_xlabel('Country of Origin', fontsize=12,
fontweight='bold')
axes[1].set_ylabel('Average Discount', fontsize=12, fontweight='bold')
axes[1].set_title('Average Discount by Country of Origin',
fontsize=15, fontweight='bold')
axes[1].tick_params(axis='x', rotation=90)
axes[1].grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.show()
```

```
<ipython-input-18-99a53c94b223>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Country of Origin', y='Price',
data=country_price_discount, palette='Blues_d', ax=axes[0])
<ipython-input-18-99a53c94b223>:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Country of Origin', y='Discount',
data=country_price_discount, palette='Oranges_d', ax=axes[1])
```

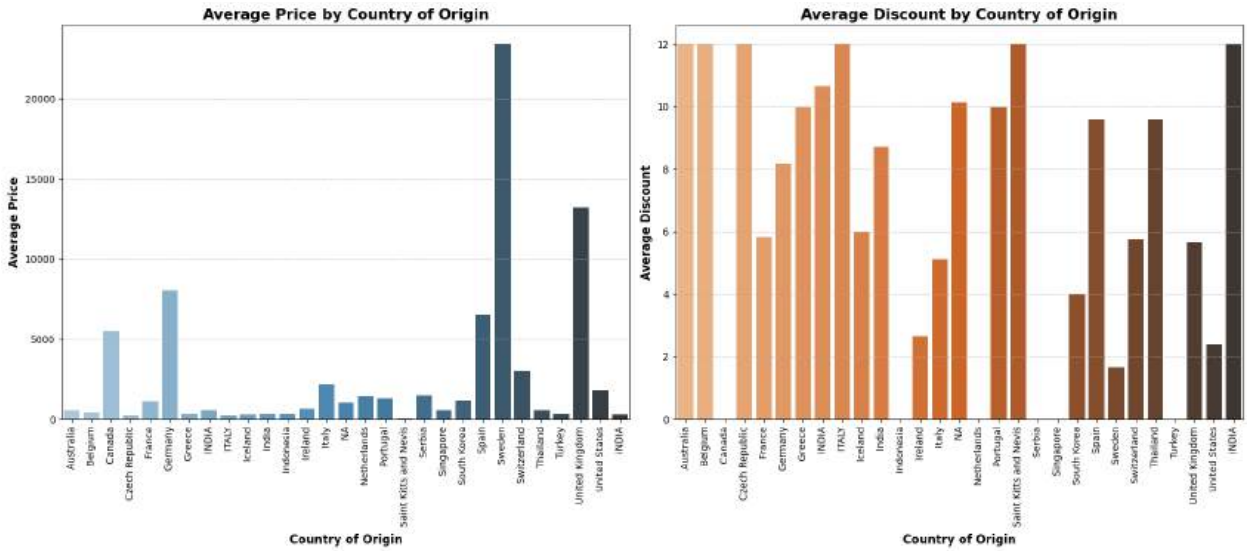


Chart - 9

```
# extracting the top 10 most frequent values in 'Tablet Info'
top_10_tablet_info = data['Tablet Info'].value_counts().head(10)
# Plotting the top 10 Tablet Info values
plt.figure(figsize=(15, 8))
top_10_tablet_info.plot(kind='bar', color=sns.color_palette("Purples",
10), edgecolor='black')
plt.xlabel('Tablet Info', fontsize=14, fontweight='bold')
plt.ylabel('Count', fontsize=14, fontweight='bold')
plt.title('Top 10 Tablet Info Values', fontsize=16, fontweight='bold')
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

plt.show()
```

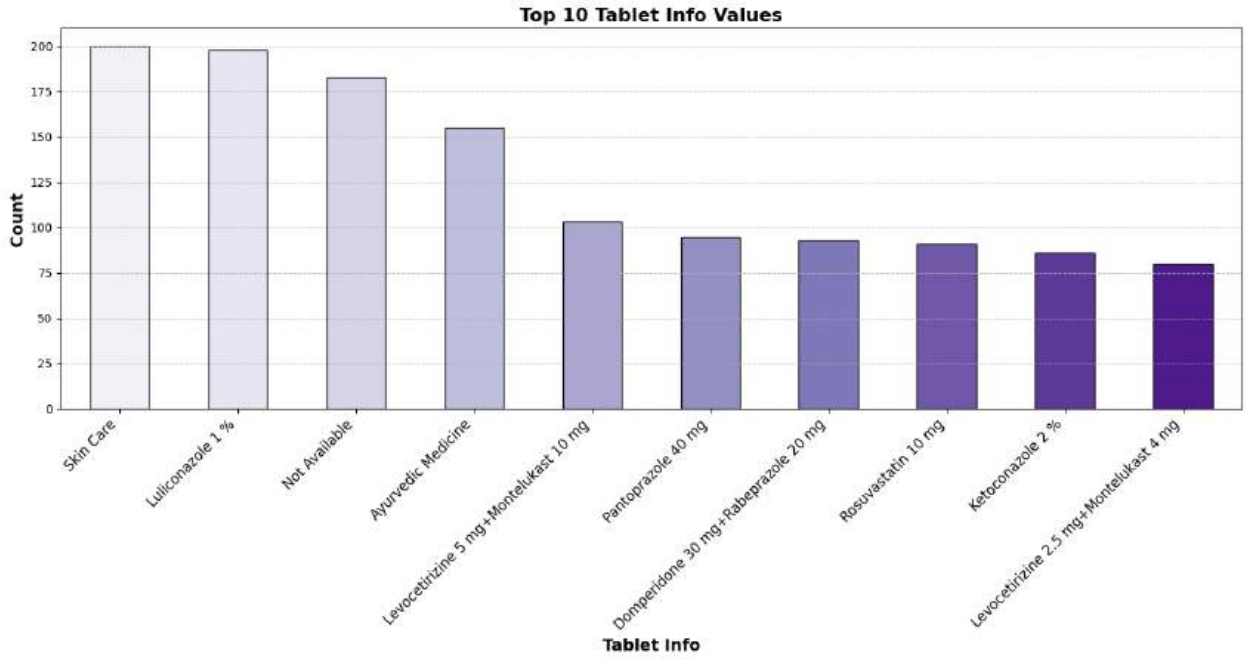


Chart - 10

```
# Grouping data by Country of Origin and Disease, counting tablets
disease_table = data.groupby(['Country of Origin', 'Disease'])['Tablet
Name'].count().reset_index()

# Renaming columns for clarity
disease_table.columns = ['Country of Origin', 'Disease', 'Count of
Tablets']

# Sorting by Count of Tablets in descending order
disease_table = disease_table.sort_values(['Count of Tablets'],
ascending=False)

from tabulate import tabulate

# Converting DataFrame to a formatted table
print(tabulate(disease_table, headers='keys', tablefmt='grid',
showindex=False))
```

```
+-----------------------
+-------------------------------------------------+--------------------+
| Country of Origin     | Disease
|   Count of Tablets |
+=======================+=================================================
==+====================+
| India                 | Supplements
|               1000 |
+-----------------------
+-------------------------------------------------+--------------------+
| India                 | Hypertension
```

```
|                 999 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Pain relief
|                 999 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Bacterial Infections
|                 998 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Fungal Infections
|                 994 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Ulcer/Reflux/Flatulence
|                 992 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Diabetes
|                 992 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | High Cholesterol
|                 987 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Epilepsy/Convulsion
|                 784 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Skin Infections
|                 731 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Allergies
|                 703 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Cough And Cold
|                 699 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Asthma/COPD
|                 660 |
+---------------------
+-------------------------------------------------+--------------------+
| India                 | Neuropathic Pain
|                 629 |
```

NA	1	General-P
NA	1	Eye Infections
NA	1	Dry Skin
NA	1	Dry Eye
NA	1	Constipation
NA	1	Bacterial Infections
NA	1	Ayurvedic Medicine
NA	1	Asthma/COPD
NA	1	Arthritis
NA	1	Anxiety
NA	1	Angina
NA	1	Anaesthesia General

Italy	Wound Care	1
Italy	Immune Suppression	1
Italy	Diarrhoea	1
Italy	Cancer Oncology	1
Italy	Blood Clot	1
Italy	Auto Immune Disease	1
Italy	Allergies	1
Ireland	Immune Suppression	1
Indonesia	Iron Supplement/Anaemia	1
India	Veterinary-Pharma	1
India	Thrombotic Disorder	1
India	Tetanus	1
India	Rabies	

```
|                    1 |
+----------------------
+-------------------------------------------------+--------------------+
| iNDIA                    | Dry Skin
|                    1 |
+----------------------
+-------------------------------------------------+--------------------+
```

Chart - 11

```
# plotting Scatter plot with regression line to visualize the
correlation
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Price', y='Discount', data=data, color='skyblue',
alpha=0.6, label='Data Points')
# Adding regression line
sns.regplot(x='Price', y='Discount', data=data, scatter=False,
color='red', label='Trend Line')
plt.title('Scatter Plot of Price vs. Discount with Regression Line',
fontsize=18, fontweight='bold')
plt.xlabel('Price', fontsize=14)
plt.ylabel('Discount', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend()
plt.tight_layout()
plt.show()
```

```
#  Calculating average Price and Discount for each Country of Origin
average_values = data.groupby('Country of Origin')[['Price',
'Discount']].mean().reset_index()

#Calculating the "Value Score"
average_values['Value_Score'] = average_values['Discount'] /
average_values['Price']

#Identifying the country with the highest Value Score (Best Value for
Money)
best_value_country =
average_values.loc[average_values['Value_Score'].idxmax()]


print("Best Value for Money Country: ", best_value_country['Country of
Origin'])
print("Average Price: ", best_value_country['Price'])
print("Average Discount: ", best_value_country['Discount'])
print("Value Score: ", best_value_country['Value_Score'])

plt.figure(figsize=(16, 6))

sns.barplot(x='Country of Origin', y='Value_Score',
data=average_values, palette='Blues_d')

plt.title('Value for Money: Best Value Countries (Discount/Price
Ratio)', fontsize=18, fontweight='bold')
plt.xlabel('Country of Origin', fontsize=14)
plt.ylabel('Value Score (Discount/Price)', fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.yticks(fontsize=12)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```

```
Best Value for Money Country:   Saint Kitts and Nevis
Average Price:  68.0
Average Discount:  12.0
Value Score:  0.17647058823529413

<ipython-input-22-af90a6a05b5c>:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Country of Origin', y='Value_Score',
data=average_values, palette='Blues_d')
```

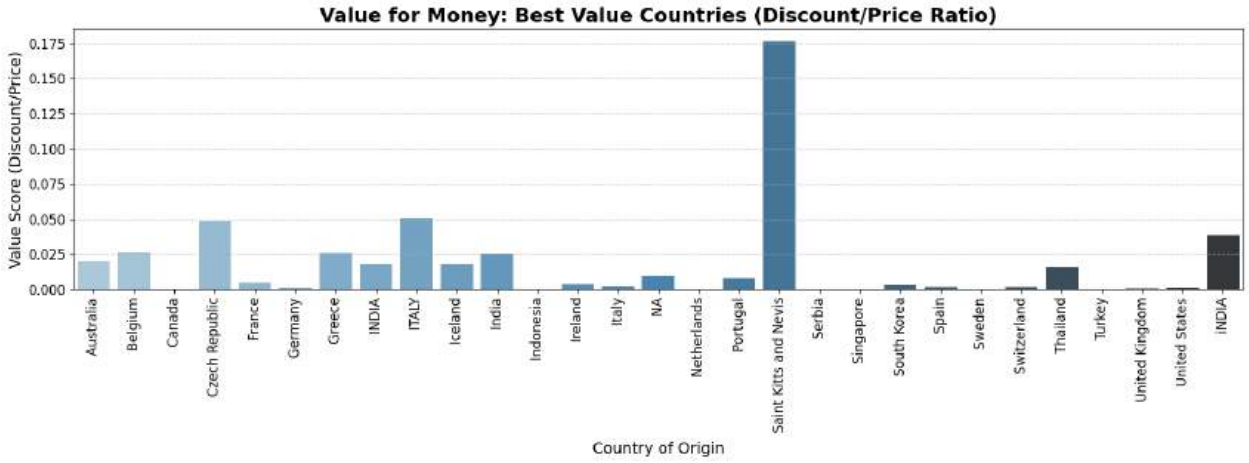
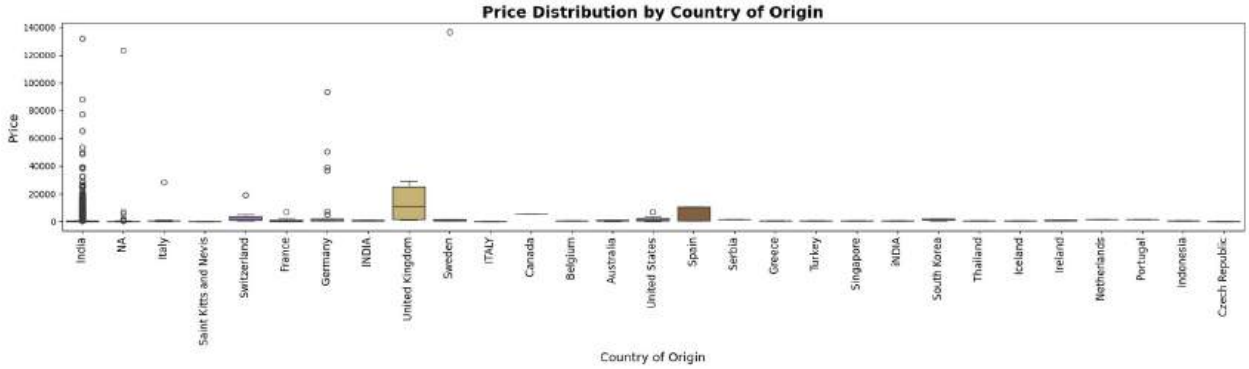


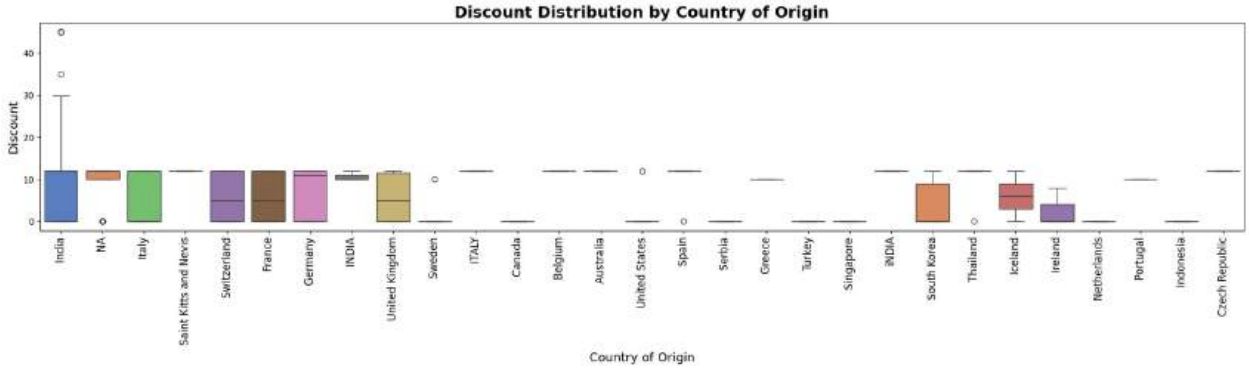
Chart - 13

```
plt.figure(figsize=(20, 6))
sns.boxplot(x='Country of Origin', y='Price', data=data,
palette='muted')
plt.title('Price Distribution by Country of Origin', fontsize=18,
fontweight='bold')
plt.xlabel('Country of Origin', fontsize=14)
plt.ylabel('Price', fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.tight_layout()
plt.show()
plt.figure(figsize=(20, 6))
sns.boxplot(x='Country of Origin', y='Discount', data=data,
palette='muted')
plt.title('Discount Distribution by Country of Origin', fontsize=18,
fontweight='bold')
plt.xlabel('Country of Origin', fontsize=14)
plt.ylabel('Discount', fontsize=14)
plt.xticks(rotation=90, fontsize=12)
plt.tight_layout()
plt.show()
```

```
<ipython-input-23-624ffbb419b6>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='Country of Origin', y='Price', data=data,
palette='muted')
```




```
<ipython-input-23-624ffbb419b6>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x='Country of Origin', y='Discount', data=data, palette='muted')
```




```
# Preparing data for radar chart
countries = average_values['Country of Origin'].tolist()
price = average_values['Price'].tolist()
discount = average_values['Discount'].tolist()

angles = np.linspace(0, 2 * np.pi, len(countries), endpoint=False).tolist()
price += price[:1]
discount += discount[:1]
angles += angles[:1]
fig, ax = plt.subplots(figsize=(10, 14), subplot_kw=dict(polar=True))
ax.plot(angles, price, color='blue', linewidth=2, label='Price')
ax.fill(angles, price, color='blue', alpha=0.25)
ax.plot(angles, discount, color='red', linewidth=2, label='Discount')
ax.fill(angles, discount, color='red', alpha=0.25)
ax.set_yticklabels([])
```

```
ax.set_xticks(angles[:-1])
ax.set_xticklabels(countries, fontsize=12, color='black', rotation=45)
ax.set_title('Radar Chart for Price and Discount by Country of
Origin', fontsize=16, fontweight='bold')
ax.legend(loc='upper right', fontsize=12)

ax.grid(color='grey', linestyle='--', alpha=0.5)
fig.patch.set_facecolor('white')

plt.tight_layout()
plt.show()
```

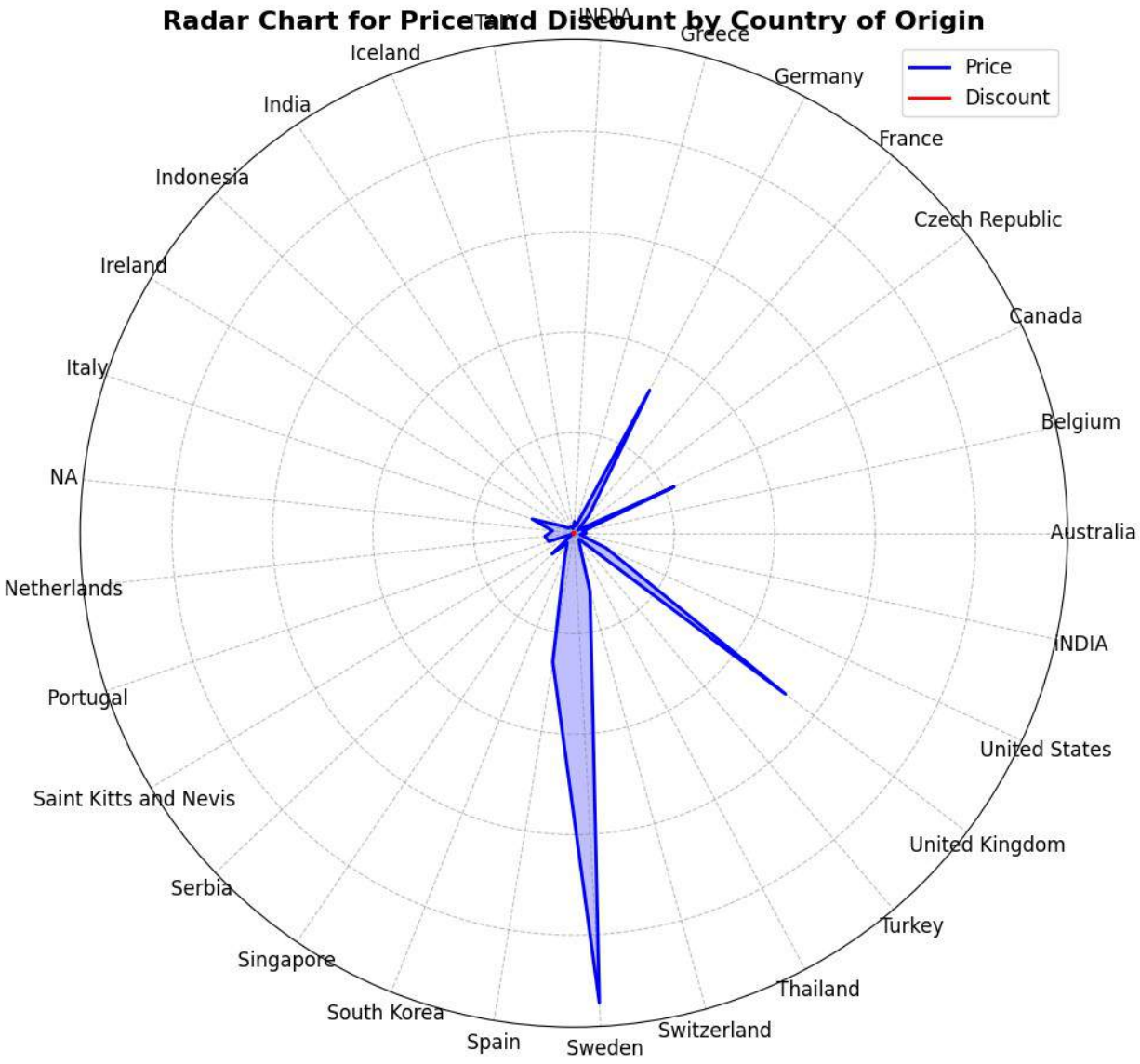
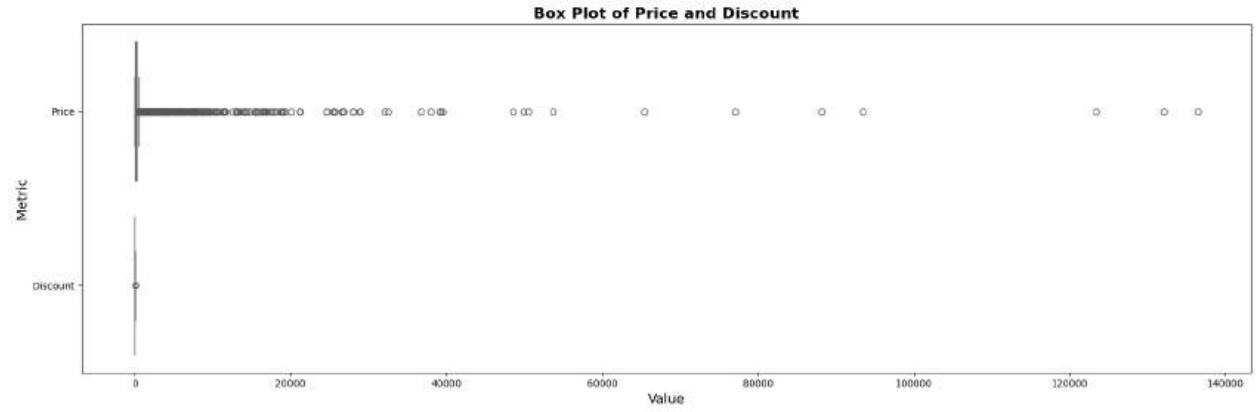


Chart - 15 - Pair Plot

```
# Reshaping the data to a long format suitable for boxplot
df_long = data[['Price', 'Discount']].melt(var_name='Metric',
value_name='Value')
plt.figure(figsize=(18, 6))
sns.boxplot(y='Metric', x='Value', data=df_long, palette='Set2')
plt.title('Box Plot of Price and Discount', fontsize=16,
fontweight='bold')
plt.xlabel('Value', fontsize=14)
plt.ylabel('Metric', fontsize=14)
plt.grid(False)
plt.tight_layout()
plt.show()
```

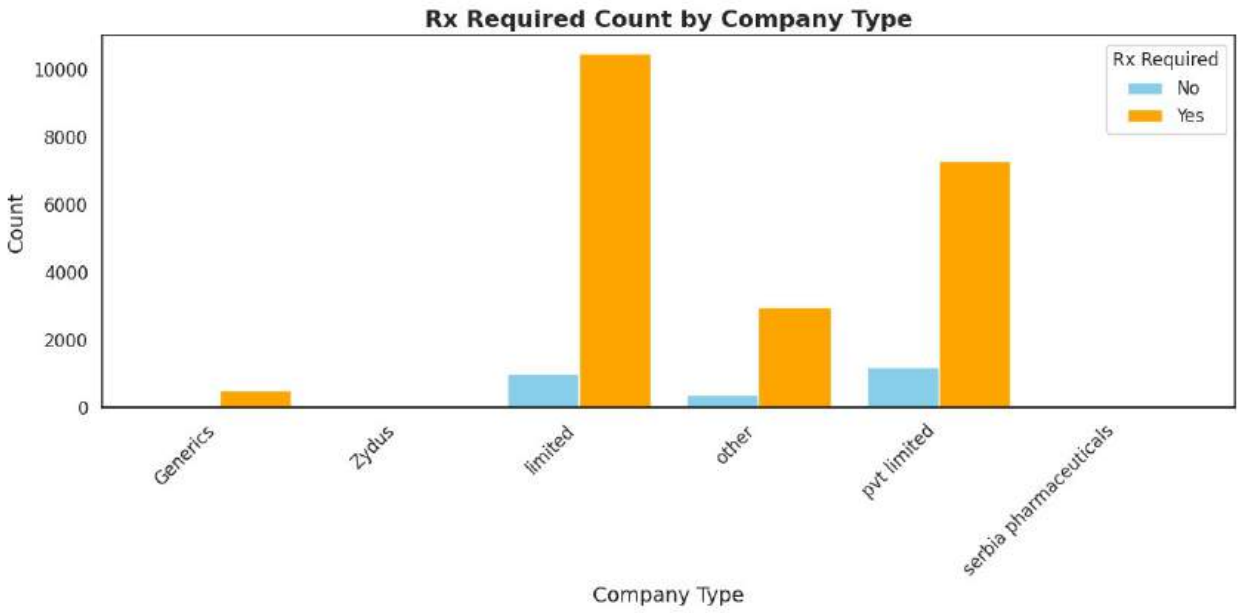
```
<ipython-input-25-ef2e03d74aec>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(y='Metric', x='Value', data=df_long, palette='Set2')
```




```
sns.set(style="white")
data['company type'] = data['Comapny Type'].str.replace(r'\(.*?\)',
'generics', regex=True).str.strip()
company_stats = data.groupby('Comapny Type')[['Price',
'Discount']].mean().reset_index()

# 1. Grouped Bar Chart
plt.figure(figsize=(15, 6))
bar_width = 0.35
x = range(len(company_stats))
plt.bar(x, company_stats['Price'], width=bar_width, label='Average
Price', color='skyblue')
plt.bar([p + bar_width for p in x], company_stats['Discount'],
```

```
width=bar_width, label='Average Discount', color='orange')
plt.xlabel('Company Type', fontsize=14)
plt.ylabel('Average Values', fontsize=14)
plt.title('Average Price and Discount by Company Type', fontsize=16,
fontweight='bold')
plt.xticks([p + bar_width / 2 for p in x], company_stats['Comapny
Type'], rotation=45, ha='right', fontsize=12)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()
plt.figure(figsize=(15, 6))
sns.lineplot(data=company_stats, x='Comapny Type', y='Price',
marker='o', label='Average Price', color='skyblue', linewidth=2)
sns.lineplot(data=company_stats, x='Comapny Type', y='Discount',
marker='o', label='Average Discount', color='orange', linewidth=2)
plt.xlabel('Company Type', fontsize=14)
plt.ylabel('Average Values', fontsize=14)
plt.title('Trend of Average Price and Discount by Company Type',
fontsize=16, fontweight='bold')
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()
```

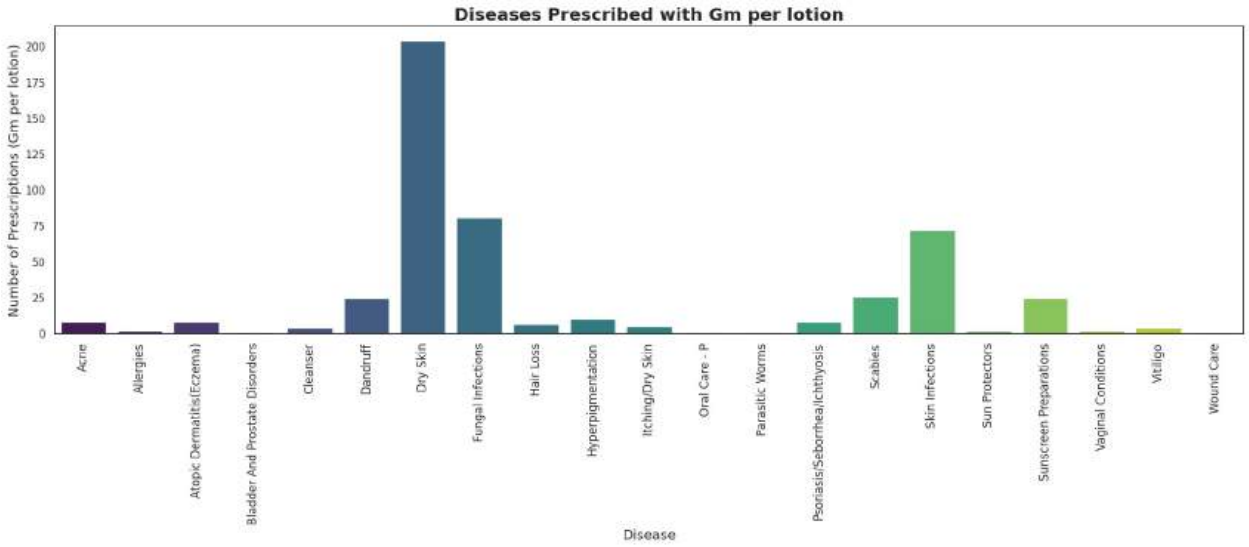

```
sns.set(style="white")
rx_counts = data.groupby(['company type', 'Rx
Required']).size().reset_index(name='Count')
rx_pivot = rx_counts.pivot(index='company type', columns='Rx
Required', values='Count').fillna(0)
ax = rx_pivot.plot(kind='bar', figsize=(12, 6), color=['skyblue',
'orange'], width=0.8)
plt.title('Rx Required Count by Company Type', fontsize=16,
fontweight='bold')
plt.xlabel('Company Type', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)
ax.grid(False)
plt.legend(title='Rx Required', labels=['No', 'Yes'], fontsize=12)
plt.tight_layout()
plt.show()
```



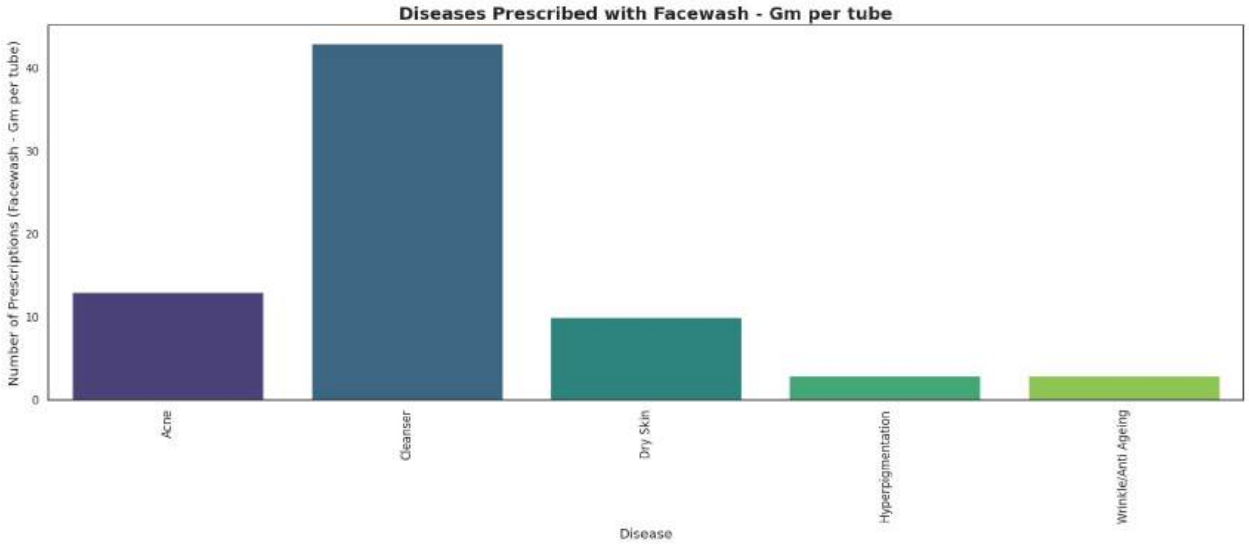

```
columns_to_process = [
    'Gm per lotion',
       'Facewash - Gm per tube', 'Spray- Ml per tube',
       'Ointment - Gm per tube.1', 'Oral solution-Ml per tube',
       'Cream-Gm per tube', 'Injection', 'Tablet per strip',
       'capsule per strip'
]

for column in columns_to_process:
    # Marking all non-zero values as 1
    data[column] = data[column].apply(lambda x: 1 if x > 0 else 0)
    # Grouping by 'Disease' and sum the current column
    grouped_data = data.groupby('Disease')[column].sum().reset_index()
    # Filtering diseases where the column has been prescribed (sum >
0)
    grouped_data = grouped_data[grouped_data[column] > 0]
    plt.figure(figsize=(18, 8))
    sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
    plt.title(f'Diseases Prescribed with {column}', fontsize=18,
fontweight='bold')
    plt.xlabel('Disease', fontsize=14)
    plt.ylabel(f'Number of Prescriptions ({column})', fontsize=14)
    plt.xticks(rotation=90, fontsize=12)
    plt.tight_layout()
    plt.show()
```

```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
```

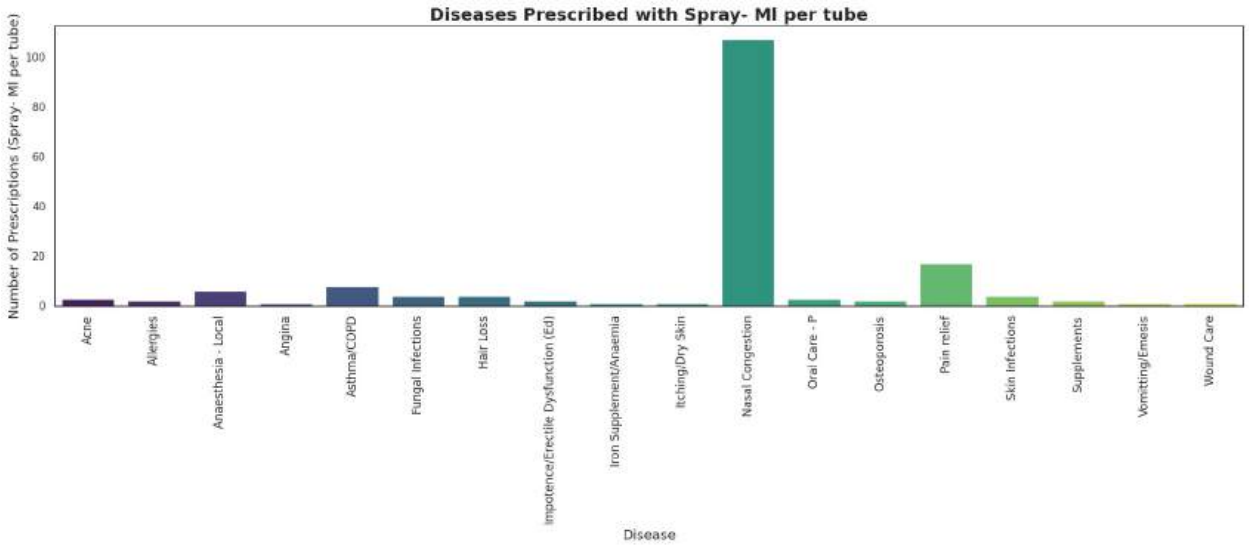
```
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```



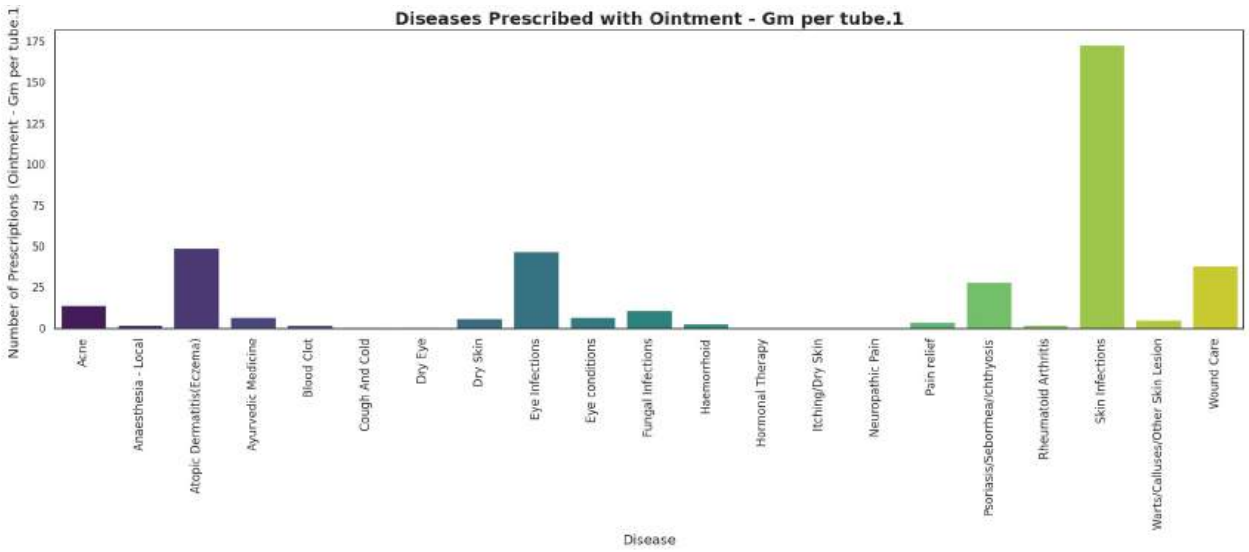

```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```



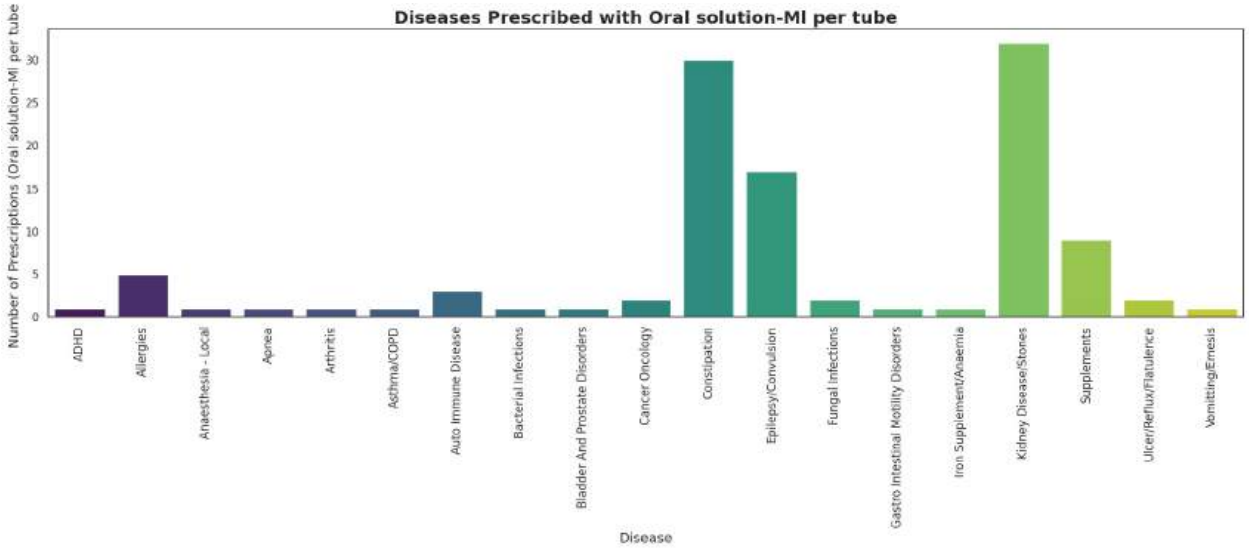
```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column, palette='viridis')
```




```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column, palette='viridis')
```

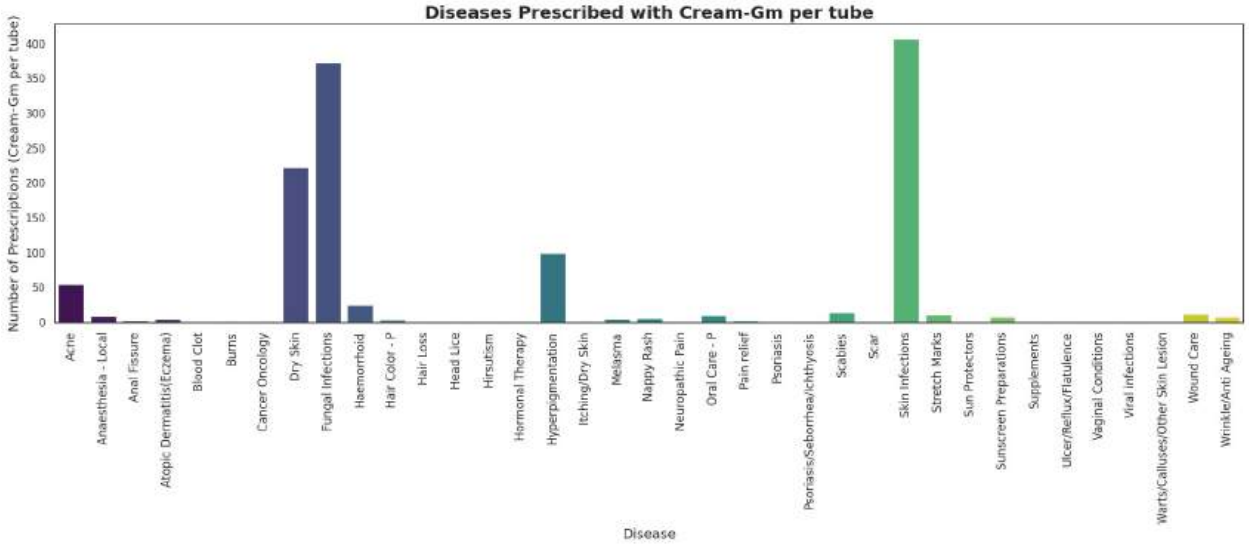



```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```

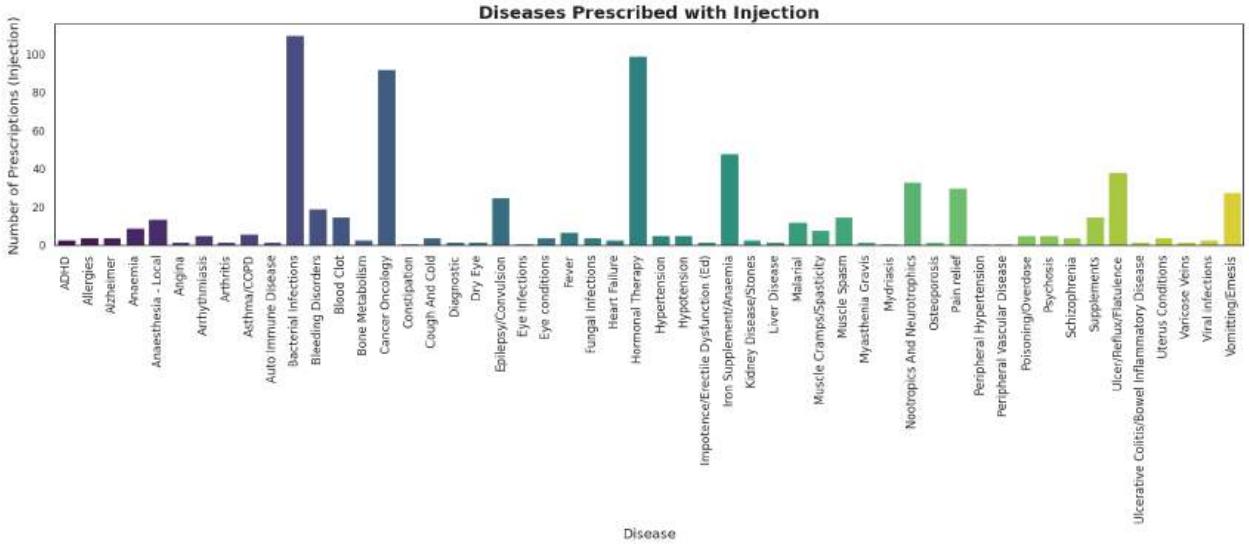



```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```

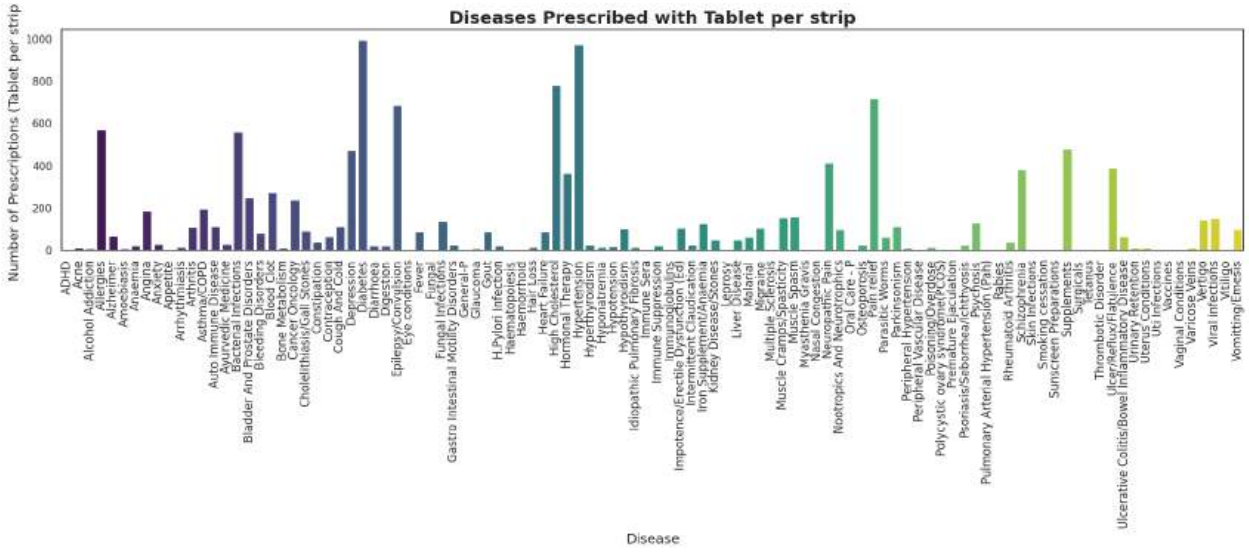



```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```

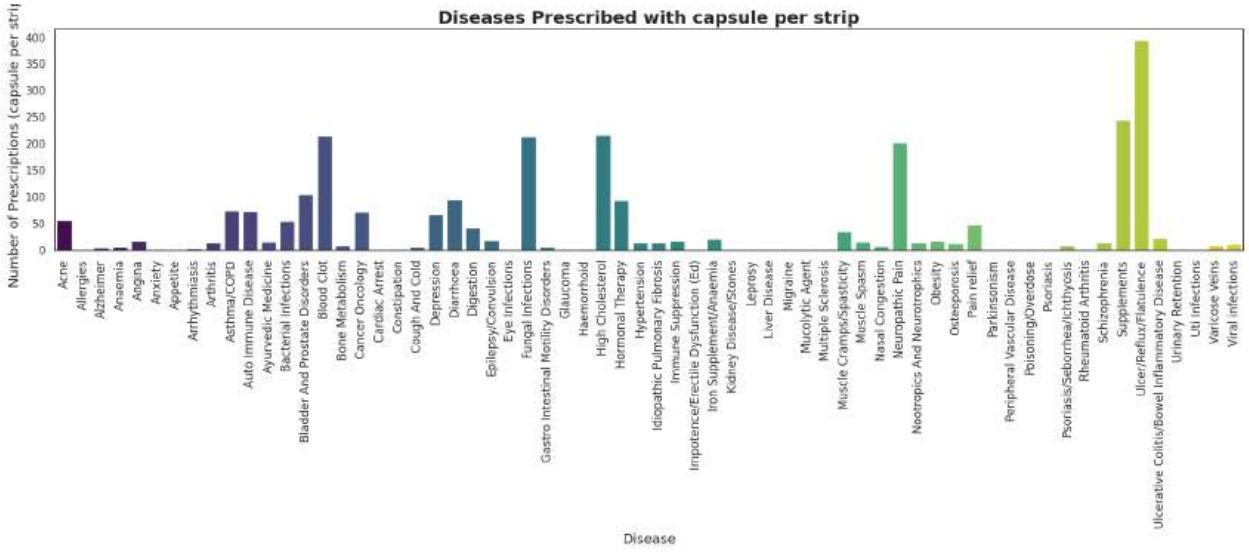



```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```




```
<ipython-input-28-f2a20e9f3d6a>:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(data=grouped_data, x='Disease', y=column,
palette='viridis')
```



```
file_path="/content/Final Cleaned data for model.xlsx"
data = pd.read_excel(file_path, sheet_name="Medicine")

data
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 12437,\n
\"fields\": [\n    {\n      \"column\": \"Tablet name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8781,\n        \"samples\": [\n
\"AMITOR SR \",\n          \"VONO \",\n          \"Olagress MD \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Tablet per Strip\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
18.37592267518446,\n        \"min\": 1.0,\n        \"max\": 1000.0,\n
\"num_unique_values\": 37,\n        \"samples\": [\n          200.0,\n
350.0,\n          15.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug 1\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 769,\n
\"samples\": [\n          \"Niacinamide (Vitamin B\",\n
\"Oxcarbazepine \",\n          \"Sodium Bicarbonate \"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 1 quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3739.6445789527365,\n        \"min\": 0.0,\n        \"max\":
200000.0,\n        \"num_unique_values\": 124,\n        \"samples\":
[\n          1.0,\n          77.0,\n          600.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 2\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
382,\n        \"samples\": [\n          \"Ergocalciferol \",\n
\"Rutoside \",\n          \"paracetamol \"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 2 Quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1590.0340290853978,\n        \"min\": 0.0,\n        \"max\": 50000.0,\
n        \"num_unique_values\": 96,\n        \"samples\": [\n
85.0,\n          480.0,\n          714.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 3\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
215,\n        \"samples\": [\n          \"Sodium Bicarbonate \",\n
\"Tenofovir Disoproxil Fumarate \",\n          \"N-Acetylcysteine \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Drug 3 info\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
9524.929312667316,\n        \"min\": 0.0,\n        \"max\": 150000.0,\
n        \"num_unique_values\": 74,\n        \"samples\": [\n
1.0,\n          5000.0,\n          50.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"Price\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2457,\n
\"min\": 1,\n        \"max\": 136503,\n        \"num_unique_values\":
1070,\n        \"samples\": [\n          11440,\n          16456,\n
357\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Medicine Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Tablet\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
import pandas as pd

# Assuming df is your DataFrame
data['Drug_2_Deliberate_NaN'] = data['Drug 2'].isna().map({True:
'Yes', False: 'No'})
data['Drug_2_Quantity_Deliberate_NaN'] = data.apply(lambda row: 'Yes'
if pd.isna(row['Drug 2 Quantity']) and pd.isna(row['Drug 2']) else
'No', axis=1)
data['Drug_3_Deliberate_NaN'] = data.apply(lambda row: 'Yes' if
pd.isna(row['Drug 3']) and pd.isna(row['Drug 2']) else 'No', axis=1)
data['Drug_3_Info_Deliberate_NaN'] = data.apply(lambda row: 'Yes' if
pd.isna(row['Drug 3 info']) and pd.isna(row['Drug 3']) else 'No',
axis=1)

# Display updated DataFrame
print(data.head())
```

```
  Tablet name  Tablet per Strip                         Drug 1  Drug 1
quantity  \
0    Cogniza               10.0  Cerebroprotein Hydrolysate
90.0
1     Atrest               10.0                Tetrabenazine
25.0
2    SAFERET               10.0                 Isotretinoin
20.0
3    SAFERET               10.0                 Isotretinoin
5.0
4    Glotret               10.0                 Isotretinoin
5.0

  Drug 2  Drug 2 Quantity Drug 3  Drug 3 info  Price Medicine Type  \
0    NaN              NaN    NaN          NaN    239        Tablet
1    NaN              NaN    NaN          NaN    336        Tablet
2    NaN              NaN    NaN          NaN    150        Tablet
3    NaN              NaN    NaN          NaN     80        Tablet
4    NaN              NaN    NaN          NaN     92        Tablet

  Drug_2_Deliberate_NaN Drug_2_Quantity_Deliberate_NaN
```

```
Drug_3_Deliberate_NaN  \
0                   Yes                                Yes
Yes
1                   Yes                                Yes
Yes
2                   Yes                                Yes
Yes
3                   Yes                                Yes
Yes
4                   Yes                                Yes
Yes

  Drug_3_Info_Deliberate_NaN
0                        Yes
1                        Yes
2                        Yes
3                        Yes
4                        Yes
```

```
data.fillna({
    'Drug 2': 'None',
    'Drug 2 Quantity': 0,
    'Drug 3': 'None',
    'Drug 3 info': 0
}, inplace=True)

data.rename(columns={'Drug 3 info': 'Drug 3 Quantity'}, inplace=True)
data
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 12437,\n
\"fields\": [\n    {\n      \"column\": \"Tablet name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 8781,\n        \"samples\": [\n
\"AMITOR SR \",\n          \"VONO \",\n          \"Olagress MD \"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Tablet per Strip\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
18.37592267518446,\n        \"min\": 1.0,\n        \"max\": 1000.0,\n
\"num_unique_values\": 37,\n        \"samples\": [\n          200.0,\n
350.0,\n          15.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug 1\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 769,\n
\"samples\": [\n          \"Niacinamide (Vitamin B\",\n
\"Oxcarbazepine \",\n          \"Sodium Bicarbonate \"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 1 quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3739.6445789527365,\n        \"min\": 0.0,\n        \"max\":
200000.0,\n        \"num_unique_values\": 124,\n        \"samples\":
```

```
[\n          1.0,\n          77.0,\n          600.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 2\",\n      \"properties\":
{\n        \"dtype\": \"category\",\n        \"num_unique_values\":
383,\n        \"samples\": [\n          \"TRIHEXYPEHNIDYL\",\n
\"Bromelain \",\n          \"Magnesium Citrate \"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 2 Quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
959.0261421445576,\n        \"min\": 0.0,\n        \"max\": 50000.0,\n
\"num_unique_values\": 96,\n        \"samples\": [\n          85.0,\n
480.0,\n          714.0\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Drug 3\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 216,\n
\"samples\": [\n          \"Sodium Bicarbonate \",\n
\"Tenofovir Disoproxil Fumarate \",\n          \"Thiamine
Hydrochloride \"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug 3 Quantity\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2813.4998217780626,\n        \"min\":
0.0,\n        \"max\": 150000.0,\n        \"num_unique_values\": 74,\n
\"samples\": [\n          350.0,\n          5000.0,\n          59.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Price\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 2457,\n
\"min\": 1,\n        \"max\": 136503,\n        \"num_unique_values\":
1070,\n        \"samples\": [\n          11440,\n          16456,\n
357\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Medicine Type\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          \"Tablet\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Drug_2_Deliberate_NaN\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"No\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug_2_Quantity_Deliberate_NaN\",\
n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"No\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Drug_3_Deliberate_NaN\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"No\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Drug_3_Info_Deliberate_NaN\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"No\"\n
```

```
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12437 entries, 0 to 12436
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Tablet name                     12435 non-null  object
 1   Tablet per Strip                12411 non-null  float64
 2   Drug 1                          12437 non-null  object
 3   Drug 1 quantity                 12270 non-null  float64
 4   Drug 2                          12437 non-null  object
 5   Drug 2 Quantity                 12437 non-null  float64
 6   Drug 3                          12437 non-null  object
 7   Drug 3 Quantity                 12437 non-null  float64
 8   Price                           12437 non-null  int64
 9   Medicine Type                   12437 non-null  object
 10  Drug_2_Deliberate_NaN           12437 non-null  object
 11  Drug_2_Quantity_Deliberate_NaN  12437 non-null  object
 12  Drug_3_Deliberate_NaN           12437 non-null  object
 13  Drug_3_Info_Deliberate_NaN      12437 non-null  object
dtypes: float64(4), int64(1), object(9)
memory usage: 1.3+ MB
```

```
data.describe()
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 8,\n  \"fields\": [\
n    {\n      \"column\": \"Tablet per Strip\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 4348.11299631753,\
n        \"min\": 1.0,\n        \"max\": 12411.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          12411.0,\
n          12.892272983643542,\n          1000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 1 quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
70007.5949964562,\n        \"min\": 0.0,\n        \"max\": 200000.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
256.22697636511816,\n          40.0,\n          12270.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Drug 2 Quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
17531.388966980867,\n        \"min\": 0.0,\n        \"max\": 50000.0,\
n        \"num_unique_values\": 6,\n        \"samples\": [\n
12437.0,\n          100.18991718260031,\n          50000.0\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug 3 Quantity\",\n      \"properties\": {\n        \"dtype\":
```

\"number\",\n   \"std\": 52433.93045873444,\n   \"min\": 0.0,\n   \"max\": 150000.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   94.81876658358125,\n   150000.0,\n   2813.4998217780626\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Price\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 47656.37672367268,\n   \"min\": 1.0,\n   \"max\": 136503.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   350.3428479536866,\n   144.0,\n   12437.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
data.isnull().sum()
```

```
Tablet name                          2
Tablet per Strip                    26
Drug 1                               0
Drug 1 quantity                    167
Drug 2                               0
Drug 2 Quantity                      0
Drug 3                               0
Drug 3 Quantity                      0
Price                                0
Medicine Type                        0
Drug_2_Deliberate_NaN                0
Drug_2_Quantity_Deliberate_NaN       0
Drug_3_Deliberate_NaN                0
Drug_3_Info_Deliberate_NaN           0
dtype: int64
```

```
data.dropna(inplace=True)
```

```
data.isnull().sum()
```

```
Tablet name                          0
Tablet per Strip                     0
Drug 1                               0
Drug 1 quantity                      0
Drug 2                               0
Drug 2 Quantity                      0
Drug 3                               0
Drug 3 Quantity                      0
Price                                0
Medicine Type                        0
Drug_2_Deliberate_NaN                0
Drug_2_Quantity_Deliberate_NaN       0
Drug_3_Deliberate_NaN                0
Drug_3_Info_Deliberate_NaN           0
dtype: int64
```

```
df=data
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 12250 entries, 0 to 12436
Data columns (total 14 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Tablet name                     12250 non-null  object 
 1   Tablet per Strip                12250 non-null  float64
 2   Drug 1                          12250 non-null  object 
 3   Drug 1 quantity                 12250 non-null  float64
 4   Drug 2                          12250 non-null  object 
 5   Drug 2 Quantity                 12250 non-null  float64
 6   Drug 3                          12250 non-null  object 
 7   Drug 3 Quantity                 12250 non-null  float64
 8   Price                           12250 non-null  int64  
 9   Medicine Type                   12250 non-null  object 
 10  Drug_2_Deliberate_NaN           12250 non-null  object 
 11  Drug_2_Quantity_Deliberate_NaN  12250 non-null  object 
 12  Drug_3_Deliberate_NaN           12250 non-null  object 
 13  Drug_3_Info_Deliberate_NaN      12250 non-null  object 
dtypes: float64(4), int64(1), object(9)
memory usage: 1.4+ MB
```

```
pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.7-cp311-cp311-
manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: graphviz in
/usr/local/lib/python3.11/dist-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (from catboost) (3.10.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in
/usr/local/lib/python3.11/dist-packages (from catboost) (1.26.4)
Requirement already satisfied: pandas>=0.24 in
/usr/local/lib/python3.11/dist-packages (from catboost) (2.2.2)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from catboost) (1.13.1)
Requirement already satisfied: plotly in
/usr/local/lib/python3.11/dist-packages (from catboost) (5.24.1)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-
packages (from catboost) (1.17.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2025.1)
Requirement already satisfied: tzdata>=2022.7 in
```

```
/usr/local/lib/python3.11/dist-packages (from pandas>=0.24->catboost)
(2025.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(4.55.8)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(24.2)
Requirement already satisfied: pillow>=8 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->catboost)
(3.2.1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.11/dist-packages (from plotly->catboost)
(9.0.0)
Downloading catboost-1.2.7-cp311-cp311-manylinux2014_x86_64.whl (98.7
MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 MB 6.3 MB/s eta
0:00:00
```

```python
# Drop some less important columns
df.drop(columns=["Medicine Type",'Drug_2_Deliberate_NaN',
'Drug_2_Quantity_Deliberate_NaN',
       'Drug_3_Deliberate_NaN', 'Drug_3_Info_Deliberate_NaN'],
inplace=True, errors='ignore')

# Encoding categorical variables
label_encoders = {}
categorical_columns = ["Tablet name", "Drug 1", "Drug 2", "Drug 3"]

for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Saving the encoders for later use

X = df.drop(columns=["Price"])
y = df["Price"]
```

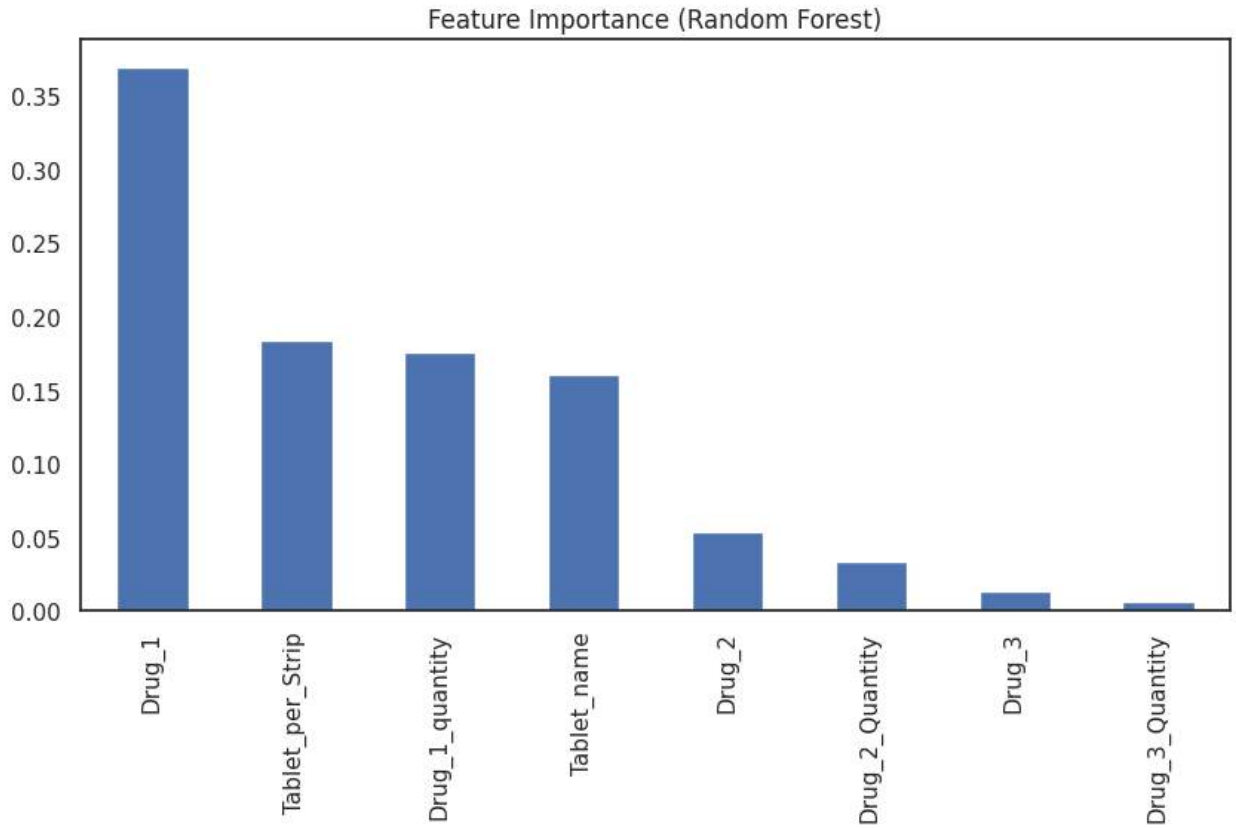
```
# Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
X_train = X_train.replace({'Yes': 1, 'No': 0})
X_test = X_test.replace({'Yes': 1, 'No': 0})
X_train = X_train.astype(int)
X_test = X_test.astype(int)

X_train.columns = X_train.columns.str.replace(' ', '_')
X_test.columns = X_test.columns.str.replace(' ', '_')

y_train = np.log1p(y_train)#applying transformation
y_test = np.log1p(y_test)


# Initializing models
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(n_estimators=100,
random_state=42),
    "XGBoost": xgb.XGBRegressor(objective="reg:squarederror",
n_estimators=100, random_state=42),
    "LightGBM": lgb.LGBMRegressor(n_estimators=100, random_state=42),
    "CatBoost": cb.CatBoostRegressor(verbose=0, random_state=42)
}

# Function to evaluate models
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return mae, mse, r2

# Training & Evaluate All Models
results = {}

for name, model in models.items():
    mae, mse, r2 = evaluate_model(model, X_train, X_test, y_train,
y_test)
    results[name] = {"MAE": mae, "MSE": mse, "R² Score": r2}


results_df = pd.DataFrame(results).T
results_df.sort_values(by="MAE", ascending=True, inplace=True)

print(results_df)
```

```python
from sklearn.model_selection import RandomizedSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10]
}

grid_search = RandomizedSearchCV(RandomForestRegressor(), param_grid,
cv=5, scoring='neg_mean_absolute_error', n_iter=10)
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)

rf_model = RandomForestRegressor(n_estimators=200,
min_samples_split=2, max_depth=None, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print("Random Forest MAE:", mean_absolute_error(y_test, y_pred_rf))
print("Random Forest MSE:", mean_squared_error(y_test, y_pred_rf))
print("Random Forest R² Score:", r2_score(y_test, y_pred_rf))


#this code was for comparing models ad to evalute the performance
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.000835 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 969
[LightGBM] [Info] Number of data points in the train set: 9800, number
of used features: 8
[LightGBM] [Info] Start training from score 4.964672
                      MAE       MSE  R² Score
Random Forest      0.347893  0.290365  0.719464
XGBoost            0.377536  0.326625  0.684432
Decision Tree      0.413883  0.456491  0.558961
CatBoost           0.432257  0.385902  0.627160
LightGBM           0.436314  0.398129  0.615348
Linear Regression  0.748605  1.035163 -0.000123
{'n_estimators': 200, 'min_samples_split': 2, 'max_depth': None}
Random Forest MAE: 0.34645868353370485
Random Forest MSE: 0.2871478735819231
Random Forest R² Score: 0.7225721456414795
```

```python
feature_importance = pd.Series(rf_model.feature_importances_,
index=X_train.columns)
feature_importance.sort_values(ascending=False).plot(kind="bar",
figsize=(10, 5))
plt.title("Feature Importance (Random Forest)")
```

```
plt.show()
```




```
df.drop(columns=["Medicine Type", 'Drug_2_Deliberate_NaN',
'Drug_2_Quantity_Deliberate_NaN',
                 'Drug_3_Deliberate_NaN',
'Drug_3_Info_Deliberate_NaN'], inplace=True, errors='ignore')

# 🔹 Encoding categorical variables
label_encoders = {}
categorical_columns = ["Tablet name", "Drug 1", "Drug 2", "Drug 3"]

for col in categorical_columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

X = df.drop(columns=["Price"])
y = df["Price"]
X_train, X_test, y_train, y_test = train_test_split(X, np.log1p(y),
test_size=0.2, random_state=42)  # Log transformation
#using just 4 model
models = {
    "Random Forest": RandomForestRegressor(n_estimators=200,
```

```
random_state=42),
    "XGBoost": xgb.XGBRegressor(objective="reg:squarederror",
n_estimators=200, random_state=42),
    "LightGBM": lgb.LGBMRegressor(n_estimators=200, random_state=42),
    "CatBoost": cb.CatBoostRegressor(verbose=0, random_state=42)
}

# 🚀 Training & Evaluating Models
best_model = None
best_mae = float("inf")

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)

    if mae < best_mae:
        best_mae = mae
        best_model = model

# 💾 Saving Best Model & Encoders
with open("best_model.pkl", "wb") as f:
    pickle.dump(best_model, f)

with open("label_encoders.pkl", "wb") as f:
    pickle.dump(label_encoders, f)

# 🔮 Function to Predict Price of a Tablet
def predict_price(tablet_name, tablet_per_strip, drug1, drug1_qty,
drug2=None, drug2_qty=None, drug3=None, drug3_qty=None):
    """Predicts the price of a single tablet based on input
details."""

    # Loading trained model & encoders
    with open("best_model.pkl", "rb") as f:
        model = pickle.load(f)
    with open("label_encoders.pkl", "rb") as f:
        label_encoders = pickle.load(f)

    # preprocessing input data
    input_data = {
        "Tablet name": tablet_name,
        "Tablet per Strip": tablet_per_strip,
        "Drug 1": drug1,
        "Drug 1 quantity": drug1_qty,
        "Drug 2": drug2 if drug2 else "Unknown",
        "Drug 2 Quantity": drug2_qty if drug2_qty else 0,
        "Drug 3": drug3 if drug3 else "Unknown",
        "Drug 3 Quantity": drug3_qty if drug3_qty else 0
    }
```

```
    # Encoding categorical values
    for col in categorical_columns:
        if input_data[col] in label_encoders[col].classes_:
            input_data[col] =
label_encoders[col].transform([input_data[col]])[0]
        else:
            input_data[col] = -1  # Unseen data handling

    # Convertingg to DataFrame
    input_df = pd.DataFrame([input_data])

    # Predict log price and reverse transformation
    log_pred_price = model.predict(input_df)[0]
    predicted_price = np.expm1(log_pred_price)  # Reversing log
transformation

    return round(predicted_price, 2)
```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with
underlines
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.001143 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 969
[LightGBM] [Info] Number of data points in the train set: 9800, number
of used features: 8
[LightGBM] [Info] Start training from score 4.964672
```

```
tablet_name = "Saferet"
tablet_per_strip =10
drug1 = "Isotretinoin "
drug2 = "collagen"


predicted_price = predict_price(tablet_name, tablet_per_strip, drug1,
drug1_qty, drug2, drug2_qty)
print(f"Predicted Price for  {tablet_name} tablet: ₹
{predicted_price}")
```

```
Predicted Price for  Saferet tablet: ₹92.76
```