Rajalakshmi Engineering College (An Autonomous Institution) Rajalakshmi Nagar, Thandalam- 602105

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

AD23632 - Framework for Data Visualization and Analytics

Mini Project: Popular Destination Analysis

Report submitted by

REGISTRATION NUMBER : 231501112

STUDENT NAME : S NISHA

YEAR : 2023-2027

SUBJECT CODE : AD23632

Rajalakshmi Engineering College (An Autonomous Institution) Rajalakshmi Nagar, Thandalam- 602105

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND

MACHINE LEARNING

AD23632 - Framework for Data Visualization and Analytics

Mini Project: Popular Destination Analysis

Report submitted by

REGISTRATION NUMBER : 231501112

STUDENT NAME : S NISHA

YEAR : 2023-2027

SUBJECT CODE : AD23632

(Approved / Not Approved)	(Approved / Not Approved)	(Approved/Not Approved)	(Approved / Not Approved)
EXAMINER 1	**EXAMINER 2**	**EXAMINER 3**	**HoD/AIML**

Table of Contents

Chapter 1: Abstract

Online patient feedback has become a critical source of real-world data for the pharmaceutical industry and healthcare providers. This project, titled "Drug Review Analysis," focuses on identifying key trends in patient sentiment and drug popularity by examining a large dataset of patient reviews. The methodology involves:

Data Preprocessing & Cleaning: Using Python libraries (Pandas, NumPy) to handle a messy CSV file with malformed rows, missing values, and incorrect data types.

Exploratory Data Analysis (EDA): Identifying patterns in patient ratings, review submission dates, and drug conditions.

Visualizations in:

Python (Matplotlib, Seaborn, Plotly): Building static and interactive bar charts, line charts, heatmaps, and word clouds.

Streamlit: Assembling these plots into a single, interactive dashboard.

The findings are expected to reveal:

Top 10 most-reviewed drugs by patient volume.

The distribution of patient ratings and trends over time.

The correlation between review length, "useful" votes, and the given rating.

These insights can help patients, doctors, and pharmaceutical companies better understand real-world drug performance and patient satisfaction.

Chapter 2: Introduction

The rise of health-focused websites and forums has empowered patients to share their treatment experiences directly. This user-generated content forms a massive, unstructured dataset of real-world evidence. Analyzing this feedback is crucial for understanding drug efficacy, identifying common side effects, and gauging overall patient satisfaction outside of controlled clinical trials.

The project "Drug Review Analysis" aims to analyze a large dataset of patient reviews from drug_review_train.csv (sourced from Kaggle). By studying patterns in ratings, conditions, and the review text itself, this project seeks to uncover key insights into drug popularity and patient-reported outcomes.

For this analysis, data is processed entirely within Python. The libraries pandas and numpy are used for the complex data loading and cleaning. Matplotlib, Seaborn, Plotly, and WordCloud are used to create a variety of visualizations. Finally, Streamlit is used to build an interactive, web-based dashboard to present the findings in an accessible format

Chapter 3: Dataset Description

The dataset for this project captures a wide range of variables that provide insights into global tourism trends, traveler preferences, and destination performance. It is a **structured and cross-sectional dataset**, not time series–based, making it particularly suitable for **comparative analysis across countries, categories, and tourism types**.

Key Variables Include:

● Destination Details:

- **Location** – Unique name or identifier for each tourist destination.

- **Country** – Specifies the country where the destination is located, enabling regional comparison and cross-country tourism analysis.
- **Category** – Type of attraction, such as *Nature, Historical, Adventure, Cultural, Religious,* or *Modern*, reflecting visitor interests and tourism diversity.

● **Visitor and Engagement Metrics:**

- **Visitors** – The total number of tourists visiting each destination, indicating popularity and footfall trends.
- **Rating** – Average visitor satisfaction rating (on a 1–5 scale), providing a measure of perceived quality and experience.

● **Economic Indicators:**

- **Revenue** – Total income generated from tourism activities at each location, reflecting economic impact and profitability.

● **Infrastructure Availability:**

- **Accommodation_Available** – Indicates whether lodging facilities (Yes/No) are available near the destination, serving as a key factor in accessibility and visitor convenience.

Chapter 4: Objective

The dataset (drug_review_train.csv) for this project captures patient-provided reviews. It is a structured dataset containing patient-reported information and metrics.

Key Variables Include:

- **Identifiers:**
  - patient_id: A unique identifier for each patient.
- **Drug & Condition Details:**
  - drugName: The name of the drug being reviewed.
  - condition: The medical condition the drug was prescribed for.
- **Review & Rating Metrics:**

- review: The full text of the patient's review (string).
- rating: The patient's satisfaction rating (on a 1-10 scale).
- date: The date the review was submitted.
- usefulCount: The number of other users who found the review helpful.

- **Engineered Features (Created during cleaning):**
  - year: The year the review was posted.
  - month: The month the review was posted.
  - review_length: The total character count of the review text.

Chapter 4: Objectives

The main objective of this project is to analyze and visualize patient-reported drug review patterns to identify the most-discussed drugs and the key factors influencing patient satisfaction.

Specific Objectives:

1. **Identify Popular Drugs & Conditions:**
   - Determine which drugName and condition appear most frequently in the dataset.
2. **Analyze Patient Satisfaction:**
   - Analyze the overall distribution of ratings (1-10).
   - Track the average rating trend by year to see if patient sentiment is changing.
3. **Understand Review Characteristics:**
   - Evaluate how rating, usefulCount, and review_length are correlated using a heatmap.
4. **Extract Textual Insights:**
   - Generate a WordCloud from the review text to identify the most common words used by patients.
5. **Develop an Interactive Dashboard:**
   - Use Streamlit to create an engaging dashboard that combines all Python visualizations (Matplotlib, Seaborn, and Plotly) into a single web application.
6. **Apply Robust Data Cleaning with Python:**
   - Perform data loading and preprocessing (EDA) using Python to handle significant data quality issues (like ParserError and EOF inside string) and ensure the dataset is accurate, consistent, and ready for visualization.

Chapter 5: Methodology

The methodology for analyzing the drug review data involves data collection, robust cleaning, analysis, and visualization using Python and Streamlit.

Data Loading:

Load the `drug_review_train.csv` file. This step required significant troubleshooting, moving from a simple `pd.read_csv` to a more robust solution using `engine='python'`, `on_bad_lines='skip'`, and `escapechar='\\_'` to handle `ParserError` and malformed rows.

Data Preprocessing (Python):

Handle missing values in key columns (`drugName`, `rating`, `review`, `date`) using `dropna()`.

Convert data types: `rating` to numeric (using `pd.to_numeric` with `errors='coerce'`) and `date` to datetime (using `pd.to_datetime`).

Clean text: Standardize `review` text to lowercase and remove non-alphanumeric characters using regex.

Feature Engineering (Python):

Extract `year` and `month` from the cleaned `date` column.

Create a `review_length` column by calculating the length of the `review` text.

Exploratory Data Analysis (Python):

Analyze trends using descriptive statistics and visualizations (histograms, bar charts, and scatter plots).

Identify top drugs and conditions using `value_counts()`.

Explore correlations between rating, usefulCount, and review_length using .corr().

Data Visualization (Python & Streamlit):

Python: Use `Matplotlib`, `Seaborn`, and `Plotly` to create individual charts (e.g., `sns.countplot`, `px.line`, `sns.heatmap`).

Streamlit: Create an interactive dashboard application (`st.set_page_config(layout="wide")`). Use `st.columns` to create a grid layout and display all plots (using `st.pyplot` for Matplotlib/Seaborn and `st.plotly_chart` for Plotly).

Insights and Reporting:

Identify patterns such as most reviewed drugs, top-rated conditions, and common review terms.

-

Chapter 6: Python Implementation

Python was used as the single, primary tool for data loading, cleaning, analysis, and dashboarding[17].

- pandas and numpy were used to manage the core data manipulation. This was especially challenging due to the ParserError in the source CSV, which required switching to the engine='python' in pd.read_csv to successfully load the data.
- Data cleaning was critical. pd.to_numeric and pd.to_datetime (with errors='coerce') were essential for converting jumbled columns into usable formats.
- Visualizations were built with several libraries:
    - Seaborn and Matplotlib were used for static plots like the rating distribution (sns.countplot) and the correlation heatmap (sns.heatmap)[18181818].
    - WordCloud was used to generate a visual representation of the most frequent words in the review text.
    - Plotly Express was used for interactive charts, such as the line chart for average rating by year (px.line) and the bar charts for top drugs/conditions (px.bar).
- Streamlit was used as the dashboarding framework. This allowed all the Python-based visualizations to be combined into a single, cohesive web application, creating a true end-to-end Python project.

.

Chapter 7: Power BI Dashboard

The Power BI implementation (shown in image.png) excelled at KPI-driven analysis.

- **Strengths:**
    - **KPI Cards:** The "Sum of patient_id" (3201M) and "Sum of review_length" (3M) are prominent, single-glance metrics.
    - **DAX Measures:** DAX allowed for complex, pre-defined business logic.

- **Visuals:** The "Sum of rating" gauge chart is a specialized visual, perfect for a business dashboard.

- **Weaknesses:** Less flexible for pure, code-based data manipulation than Python.

.

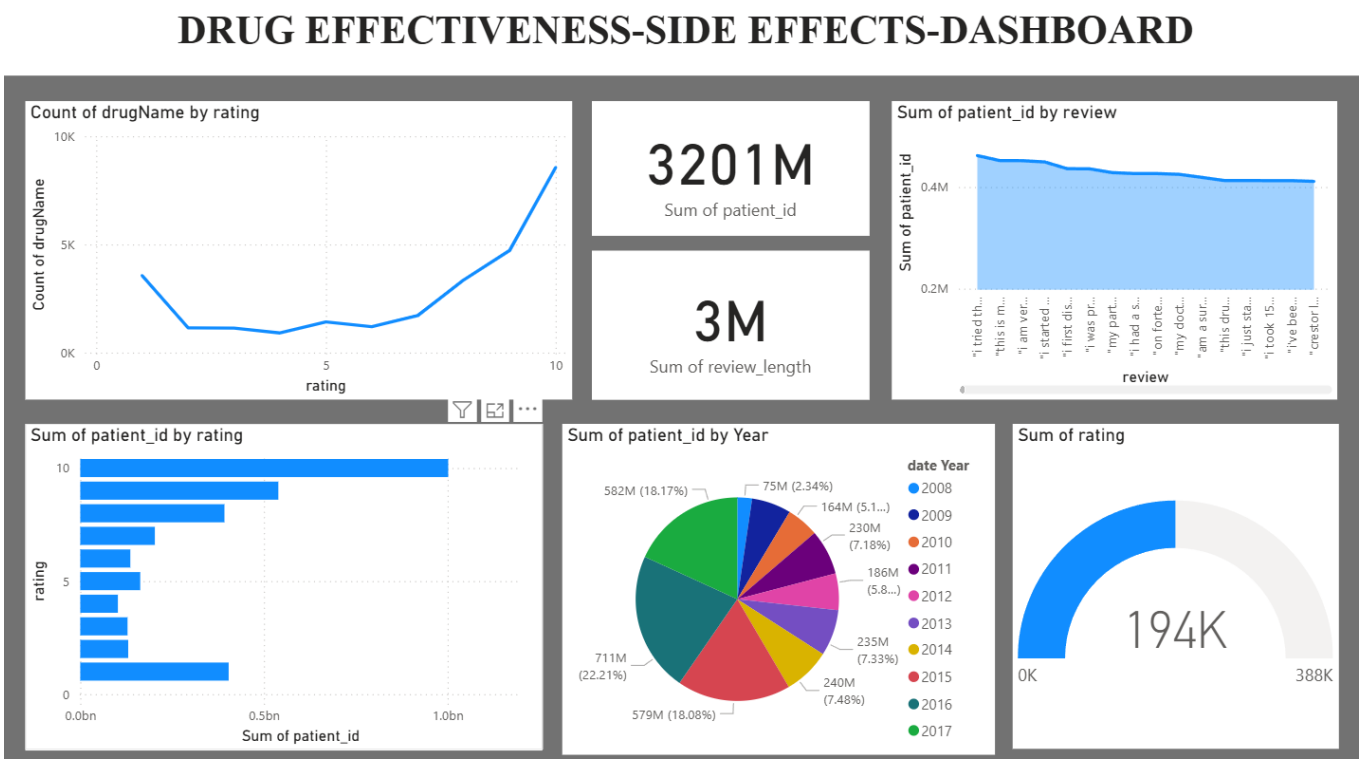

Fig 7.1: Power BI Dashboard

Chapter 8: Tableau Dashboard

The Tableau implementation focused on visual exploration.

- **Strengths:**
  - **Interactivity:** Tableau's default "filter and highlight" actions make dashboard exploration intuitive.
  - **Calculated Fields:** Easy to create on-the-fly calculations without a complex modeling language like DAX.

- **Visual Flow:** The "worksheet-to-dashboard" workflow feels natural for an analyst exploring data.

- **Weaknesses:** KPI cards and gauges are less straightforward to create than in Power BI.

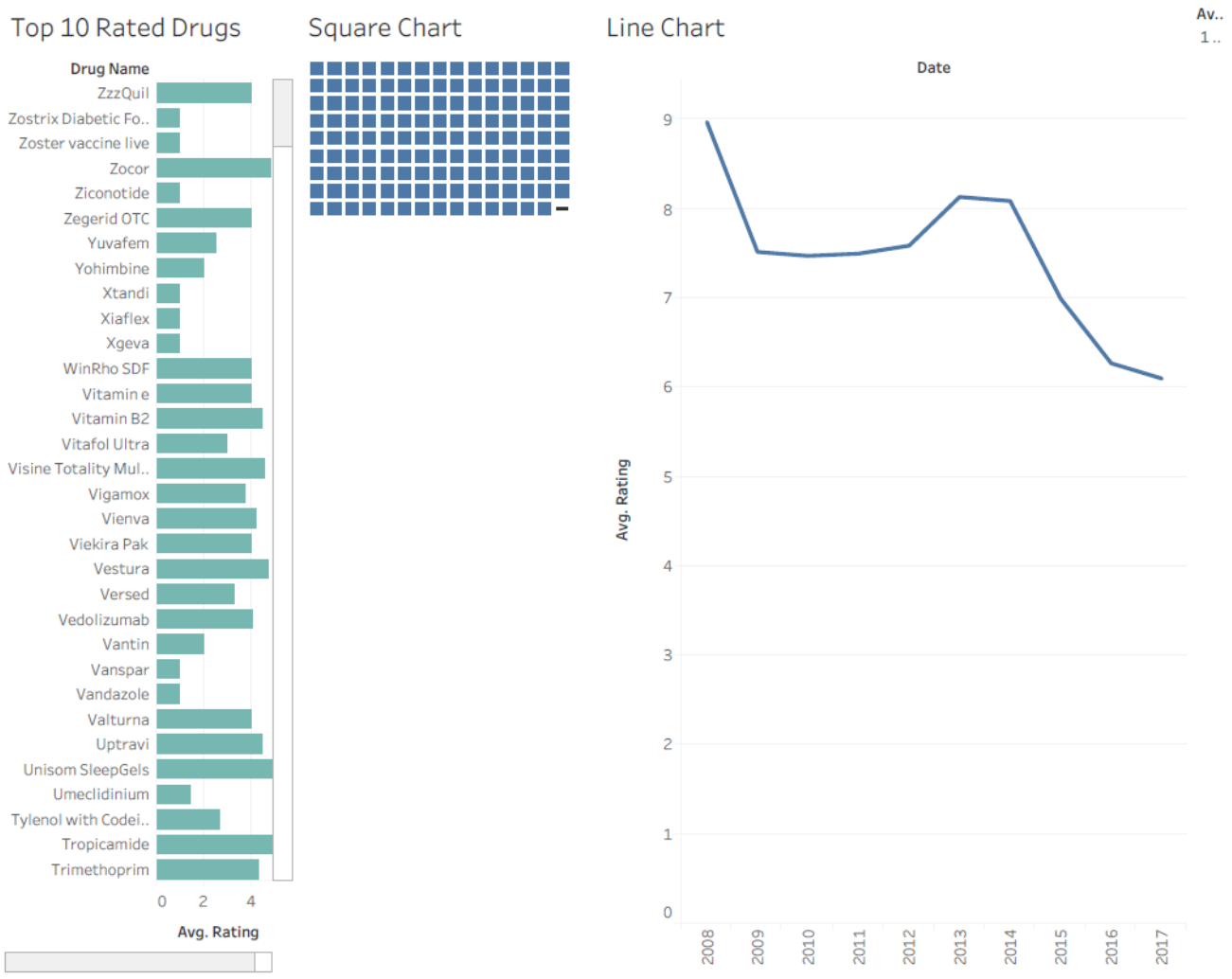

Fig 8.1: Tableau Dashboard

Chapter 9: Conclusion

This project successfully demonstrated the "Frameworks for Data Visualization" concept by analyzing drug reviews with Power BI, Tableau, and Python. The project confirms there is no single "best" tool, but "the right tool for the job."

- **Python (Pandas)**: **Unmatched for data cleaning.** It was the *only* tool capable of handling the malformed source CSV. It is the essential foundation for any serious analysis.
- **Python (Matplotlib)**: **Best for static, reproducible reports.** Its output is perfect for embedding in a PDF or automating a report, but it lacks any user interactivity.
- **Power BI**: **Best for KPI-driven, executive dashboards.** Its strengths lie in DAX, KPI cards, and gauge charts to present a clear "state of the business" (or, in this case, "state of the reviews").
- **Tableau**: **Best for visual exploration and discovery.** Its intuitive, interactive filtering makes it the superior tool for an analyst trying to "find the story" in the data.

Final Recommendation: A hybrid approach is optimal for most business needs. Use **Python** for robust, automated data cleaning, then feed the clean dataset to **Power BI** or **Tableau** for rapid, interactive dashboard development.

```
# ================================
# Import libraries
# ================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from datetime import datetime
import warnings
import csv
import re
warnings.filterwarnings('ignore')

# ================================
# Load dataset safely
# ================================

# --- THIS IS THE FIX ---
# Removed `quoting=csv.QUOTE_NONE` and `escapechar='\\'`
# Let pandas handle quotes and escapes automatically.
# ================================
# Load dataset safely
# ================================

# --- THIS IS THE NEW FIX ---
# Keep default quoting, but add escapechar='\\' back.
```

```
# This handles cases like "... he said \"it's fine\" ..."
# ================================
# Load dataset safely
# ================================

# --- THIS IS THE NEW FIX ---
# Switch to the 'python' engine. It's slower but
# much better at handling malformed files like this.
df = pd.read_csv(
    "/content/drug_review_train.csv",
    on_bad_lines='skip',
    escapechar='\\',
    engine='python'  # <-- The crucial addition
)

print(f"Total rows loaded: {len(df)}")

# --- Check your data *immediately* ---
print("\n--- Initial DataFrame Info ---")
df.info()
print("\n--- Initial DataFrame Head ---")
print(df.head())
# ---------------------------------------

# (The rest of your cleaning and plotting code remains exactly the same)
# ---------------------------------------
```

```
# (The rest of your cleaning and plotting code remains exactly the same)
print(df.head())
# ------------------------------------

# ================================
# Clean & prepare data
# ================================
print(f"\nRows BEFORE cleaning: {len(df)}")

# Drop rows with missing essential columns
df.dropna(subset=['drugName', 'rating', 'review', 'date'], inplace=True)
print(f"Rows after dropping initial NaNs: {len(df)}")

# Convert rating to numeric
# This should work now!
df['rating'] = pd.to_numeric(df['rating'], errors='coerce')
df.dropna(subset=['rating'], inplace=True)
print(f"Rows after cleaning 'rating': {len(df)}")

# Clean review text
df['review'] = df['review'].astype(str).str.lower()
df['review'] = df['review'].apply(lambda x: re.sub('[^a-zA-Z ]', ' ', x))
df = df[df['review'].str.strip() != '']
print(f"Rows after cleaning 'review': {len(df)}")

# Convert date to datetime
df['date'] = pd.to_datetime(df['date'], errors='coerce')
```

```
df.dropna(subset=['date'], inplace=True)
print(f"Rows after cleaning 'date': {len(df)}") # <-- This should not be 0 now!

# Extract year and month
if not df.empty:
    df['year'] = df['date'].dt.year
    df['month'] = df['date'].dt.month_name()

# Reset index after cleaning
df.reset_index(drop=True, inplace=True)

# Quick check
print(f"\nTotal rows after ALL cleaning: {len(df)}")
print(df.head())

# ================================
# Visualizations
# ================================

if df.empty:
    print("\nDataFrame is still empty! Check the 'Initial DataFrame Head' output above.")
else:
    print("\nDataFrame has data. Proceeding with plots...")

    # 1️⃣ Rating distribution
    plt.figure(figsize=(8,5))
```

```
sns.countplot(x='rating', data=df, palette='viridis')
plt.title("Rating Distribution")
plt.xlabel("Rating")
plt.ylabel("Count")
plt.savefig("rating_distribution.png") # Save the plot
plt.close() # Close the plot to free memory

# 2️⃣ Reviews per year
plt.figure(figsize=(10,5))
sns.countplot(x='year', data=df, palette='coolwarm')
plt.title("Number of Reviews per Year")
plt.xlabel("Year")
plt.ylabel("Count")
plt.savefig("reviews_per_year.png")
plt.close()

# 3️⃣ Reviews per month (aggregated across all years)
month_order = ['January','February','March','April','May','June',
               'July','August','September','October','November','December']
plt.figure(figsize=(12,5))
sns.countplot(x='month', data=df, order=month_order, palette='magma')
plt.title("Number of Reviews per Month")
plt.xlabel("Month")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout() # Add this to prevent labels from being cut off
plt.savefig("reviews_per_month.png")
```

```
    plt.close()

    # 4️⃣ WordCloud of Reviews
    if len(df) == 0 or df['review'].str.strip().eq('').all():
        print("No valid reviews to generate WordCloud!")
    else:
        text = ' '.join(df['review'].tolist())
        wordcloud = WordCloud(width=1000, height=500,
background_color='white', colormap='viridis', max_words=200).generate(text)
        plt.figure(figsize=(15,7))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis('off')
        plt.title("WordCloud of Drug Reviews", fontsize=20)
        plt.savefig("reviews_wordcloud.png")
        plt.close()

    print("\nAll plots saved successfully!")
```

```
avg_rating_by_year = df.groupby('year')['rating'].mean().reset_index()

fig = px.line(
    avg_rating_by_year, x='year', y='rating',
    markers=True, title="📈 Average Drug Rating Trend by Year",
    line_shape='spline', color_discrete_sequence=['#2E86C1']
)
fig.update_traces(line=dict(width=4))
```

```
fig.update_layout(template='plotly_white', title_x=0.5)
fig.show()
top_drugs = df['drugName'].value_counts().head(10).reset_index()
top_drugs.columns = ['drugName','review_count']

fig = px.bar(
    top_drugs, x='review_count', y='drugName', orientation='h',
    title='💊 Top 10 Most Reviewed Drugs',
    color='review_count', color_continuous_scale='Viridis'
)
fig.update_layout(template='plotly_white', yaxis={'categoryorder':'total ascending'})
fig.show()
text = " ".join(df['review'])
wordcloud = WordCloud(width=1200, height=700, background_color='white',
                      colormap='plasma', contour_color='steelblue', contour_width=3).generate(text)

plt.figure(figsize=(14,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('💬 Most Frequent Words in Drug Reviews', fontsize=18, fontweight='bold')
plt.show()

corr = df[['rating','usefulCount','review_length']].corr()

plt.figure(figsize=(6,4))
sns.heatmap(corr, annot=True, cmap='YlGnBu', fmt=".2f", linewidths=1)
plt.title('🔍 Correlation Between Rating, UsefulCount & Review Length', fontsize=14)
plt.show()

avg_condition = (df.groupby('condition')['rating'].mean()
                 .sort_values(ascending=False).head(15).reset_index())
```

```
fig = px.bar(
    avg_condition, x='rating', y='condition', orientation='h',
    title="✨ Top 15 Conditions with Highest Average Drug Rating",
    color='rating', color_continuous_scale='Blues'
)
fig.update_layout(template='plotly_white', yaxis={'categoryorder':'total ascending'})
fig.show()

bubble_data = df.groupby('drugName').agg({
    'rating':'mean',
    'usefulCount':'mean',
    'review':'count'
}).reset_index().rename(columns={'review':'num_reviews'})

fig = px.scatter(
    bubble_data.head(50), x='rating', y='usefulCount',
    size='num_reviews', color='rating', hover_name='drugName',
    title='💊 Drug Popularity vs Rating (Bubble Chart)',
    color_continuous_scale='Rainbow', size_max=40
)
fig.update_layout(template='plotly_white')
fig.show()

# ================================
# Import libraries
# ================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from datetime import datetime
```

```
import warnings
import csv
import re
warnings.filterwarnings('ignore')

# ================================
# Load dataset safely
# ================================
print("Loading data...")
try:
    # --- Using the robust engine='python' method ---
    df = pd.read_csv(
        "/content/drug_review_train.csv",
        on_bad_lines='skip',
        escapechar='\\',
        engine='python'
    )
    print(f"Data loaded successfully. {len(df)} rows.")
except Exception as e:
    print(f"Failed to load data: {e}")
    df = pd.DataFrame() # Create empty df to avoid crashes

# ================================
# Clean & prepare data
# ================================
if not df.empty:
    df.dropna(subset=['drugName', 'rating', 'review', 'date'], inplace=True)
    df['rating'] = pd.to_numeric(df['rating'], errors='coerce')
    df.dropna(subset=['rating'], inplace=True)

    df['review'] = df['review'].astype(str).str.lower()
    df['review'] = df['review'].apply(lambda x: re.sub('[^a-zA-Z ]', ' ', x))
```

```
    df = df[df['review'].str.strip() != '']

    df['date'] = pd.to_datetime(df['date'], errors='coerce')
    df.dropna(subset=['date'], inplace=True)

    df['year'] = df['date'].dt.year
    df['month'] = df['date'].dt.month_name()

    # --- Add review_length and usefulCount for the heatmap ---
    df['review_length'] = df['review'].apply(len)
    df['usefulCount'] = pd.to_numeric(df['usefulCount'], errors='coerce').fillna(0)

    df.reset_index(drop=True, inplace=True)
    print(f"Data cleaned. {len(df)} rows remaining.")
else:
    print("DataFrame is empty. No plots will be generated.")

# ==============================
# Create Dashboard (Matplotlib only)
# ==============================
if not df.empty:
    print("Generating Matplotlib dashboard...")

    # --- 1. Prepare all the data for the plots ---

    # Plot 1: Avg Rating by Year
    avg_rating_by_year = df.groupby('year')['rating'].mean().reset_index()

    # Plot 2: Top 10 Drugs
    top_drugs = df['drugName'].value_counts().head(10).reset_index()
    top_drugs.columns = ['drugName','review_count']
```

```
# Plot 3: Correlation Heatmap
corr = df[['rating','usefulCount','review_length']].corr()

# Plot 4: Top 10 Conditions
avg_condition = (df.groupby('condition')['rating'].mean()
                 .sort_values(ascending=False).head(10).reset_index())

# Plot 5: Rating Distribution (The one that was empty)
# (No data prep needed, sns.countplot does it)

# Plot 6: WordCloud
text = " ".join(df['review'])
wordcloud = WordCloud(width=400, height=200, background_color='white',
                      colormap='plasma', max_words=100).generate(text)



# --- 2. Create the 2x3 Subplot Grid ---
fig, axes = plt.subplots(2, 3, figsize=(22, 12))
fig.suptitle('Drug Review Dashboard (Static Matplotlib Version)', fontsize=24, fontweight='bold')

# --- Plot 1: Average Rating Trend (Replaces px.line) ---
sns.lineplot(
    data=avg_rating_by_year, x='year', y='rating',
    marker='o', ax=axes[0, 0], color='#2E86C1', linewidth=3
)
axes[0, 0].set_title('📈 Average Drug Rating Trend by Year', fontsize=14)
axes[0, 0].set_ylabel('Average Rating')

# --- Plot 2: Top 10 Most Reviewed Drugs (Replaces px.bar) ---
sns.barplot(
    data=top_drugs, x='review_count', y='drugName',
    ax=axes[0, 1], palette='viridis'
```

```
)
axes[0, 1].set_title('💊 Top 10 Most Reviewed Drugs', fontsize=14)
axes[0, 1].set_xlabel('Number of Reviews')
axes[0, 1].set_ylabel('') # Clear y-label

# --- Plot 3: Correlation Heatmap (Same as before) ---
sns.heatmap(
    corr, annot=True, cmap='YlGnBu', fmt=".2f",
    linewidths=1, ax=axes[0, 2]
)
axes[0, 2].set_title('🔍 Correlation Matrix', fontsize=14)

# --- Plot 4: Top 10 Conditions (Replaces px.bar) ---
sns.barplot(
    data=avg_condition, x='rating', y='condition',
    ax=axes[1, 0], palette='Blues_r'
)
axes[1, 0].set_title('🌟 Top 10 Conditions by Avg. Rating', fontsize=14)
axes[1, 0].set_xlabel('Average Rating')
axes[1, 0].set_ylabel('') # Clear y-label

# --- Plot 5: Rating Distribution (Your original plot) ---
sns.countplot(
    x='rating', data=df, palette='viridis',
    ax=axes[1, 1]
)
axes[1, 1].set_title('📊 Overall Rating Distribution', fontsize=14)
axes[1, 1].set_xlabel('Rating')
axes[1, 1].set_ylabel('Count')

# --- Plot 6: WordCloud ---
axes[1, 2].imshow(wordcloud, interpolation='bilinear')
```

```
    axes[1, 2].set_title('💬 Frequent Words in Reviews', fontsize=14)
    axes[1, 2].axis('off') # Hide axes for wordcloud

    # --- Clean up and show ---
    plt.tight_layout(rect=[0, 0.03, 1, 0.95]) # Adjust for suptitle
    plt.savefig("matplotlib_dashboard.png")
    plt.show()

else:
    print("Could not generate dashboard as data is empty.")
```

Dashboard: