import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings('ignore')

* Imports Pandas (for data manipulation) and NumPy (for numerical operations)
* Suppresses warning messages to keep output clean

Data Loading

python

df = pd.read\_csv('netflix\_titles.csv')

df.head()

* Reads the Netflix titles dataset from CSV
* Displays first 5 rows to inspect the data

1. Remove Duplicates

python

df = df.drop\_duplicates()

df.head()

* Removes exact duplicate rows
* Shows first 5 rows again to verify

2. Remove Missing Essential Fields

python

df = df.dropna(subset=['title', 'type', 'release\_year'])

* Drops rows where any of these critical columns are missing:
  + title (show/movie name)
  + type (Movie or TV Show)
  + release\_year

3. Clean Categorical Columns

python

cat\_cols = ['type', 'rating', 'country', 'director', 'cast', 'listed\_in']

for col in cat\_cols:

if col in df.columns:

df[col] = df[col].astype(str).replace(r'^\s\*$', np.nan, regex=True)

df[col] = df[col].apply(lambda x: x.strip().title() if isinstance(x, str) else x)

* Processes text columns to:
  1. Convert empty strings to NaN
  2. Strip whitespace
  3. Convert to title case (first letter of each word capitalized)

4. Convert Data Types

python

df['release\_year'] = pd.to\_numeric(df['release\_year'], errors='coerce')

df['date\_added'] = pd.to\_datetime(df['date\_added'], errors='coerce')

df['year\_added'] = df['date\_added'].dt.year

* Converts release\_year to numeric (coercing errors to NaN)
* Converts date\_added to datetime format
* Extracts just the year added into a new column

5. Extract Movie Durations

python

if 'duration' in df.columns:

movie\_mask = df['type'].str.lower() == 'movie'

df.loc[movie\_mask, 'duration\_mins'] = df.loc[movie\_mask, 'duration'].str.extract(r'(\d+)').astype(float)

* For movies only, extracts duration numbers from strings (e.g., "90 min" → 90)
* Stores in new duration\_mins column as float

6. Fill Missing Values

python

df['country'] = df['country'].fillna('Unknown')

df['rating'] = df['rating'].fillna('Not Rated')

df['listed\_in'] = df['listed\_in'].fillna('Unknown Genre')

* Replaces nulls in these columns with sensible defaults

7. Final Blank String Cleaning

python

df = df.replace(r'^\s\*$', np.nan, regex=True)

* Catches any remaining blank/whitespace-only strings and converts to NaN

8. Filter Unrealistic Years

python

current\_year = pd.Timestamp.now().year

df = df[(df['release\_year'] >= 1900) & (df['release\_year'] <= current\_year)]

* Keeps only records with release years between 1900 and current year

9. Save Cleaned Data

python

df.to\_csv('netflix\_titles\_cleaned.csv', index=False)

print("✅ Cleaned file saved as netflix\_titles\_cleaned.csv")

* Saves cleaned dataset to new CSV without row indices
* Prints confirmation

10. Data Summary

python

print(f"Total records: {len(df)}")

print(f"Columns: {list(df.columns)}")

print(f"Missing values per column:\n{df.isnull().sum()}")

print("\nSummary statistics:")

print(df.describe(include='all'))

* Provides key metrics:
  + Final record count
  + Column list
  + Null values per column
  + Statistical summary (for both numeric and categorical columns)

-----------------------------------------------------------------------------------------------------------------------------------------------dashboard code

import streamlit as st *# Web app framework*

import pandas as pd *# Data manipulation*

import plotly.express as px *# Interactive charts*

import numpy as np *# Numerical operations*

from datetime import datetime *# Date handling*

import warnings *# Suppress warnings*

warnings.filterwarnings('ignore') *# Hide warnings*

**Explanation:**

* **Streamlit (**st**)** → Used to build interactive web apps in Python.
* **Pandas (**pd**)** → Data manipulation library (DataFrames, cleaning, filtering).
* **Plotly Express (**px**)** → High-level interface for creating interactive charts.
* **NumPy (**np**)** → Numerical computing (handling arrays, math operations).
* warnings → Suppresses warnings to keep the output clean.

**2. Page Configuration**

python

st.set\_page\_config(

page\_title="Netflix Titles EDA Dashboard",

page\_icon="🎬",

layout="wide",

initial\_sidebar\_state="expanded"

)

**Explanation:**

* Sets up the **browser tab title (**page\_title**)** and **favicon (**page\_icon**)**.
* layout="wide" → Uses full screen width.
* initial\_sidebar\_state="expanded" → Shows filters by default.

**3. Custom CSS Styling**

python

st.markdown("""

<style>

.main-header { color: #E50914; } /\* Netflix red \*/

.metric-container { background: #f0f2f6; } /\* Card styling \*/

</style>

""", unsafe\_allow\_html=True)

**Explanation:**

* Custom CSS for:
  + **Header color** (Netflix red #E50914).
  + **Metric cards** (background, shadows, borders).
* unsafe\_allow\_html=True → Allows HTML/CSS in Streamlit.

**4. Data Loading & Cleaning**

**Pandas (**pd.read\_csv()**)**

python

df = pd.read\_csv('netflix\_titles\_cleaned.csv')

* Reads a CSV file into a **Pandas DataFrame** (structured table).

**Data Cleaning Steps**

python

df['date\_added'] = pd.to\_datetime(df['date\_added']) *# Convert to datetime*

df['release\_year'] = pd.to\_numeric(df['release\_year']) *# Ensure numeric*

df['year\_added'] = df['date\_added'].dt.year *# Extract year*

* pd.to\_datetime() → Converts strings to datetime objects.
* pd.to\_numeric() → Ensures numerical data (e.g., release\_year).
* dt.year → Extracts the year from a datetime column.

**Handling Missing Data**

python

df['country'] = df['country'].fillna('Unknown') *# Replace NaN*

df['rating'] = df['rating'].fillna('Not Rated')

* fillna() → Replaces missing values (NaN) with defaults.

**5. Filtering Logic (Pandas & NumPy)**

**Boolean Masking**

python

mask = pd.Series(True, index=df.index) *# Initialize mask*

mask &= df['type'].isin(selected\_types) *# Filter by type*

mask &= (df['release\_year'] >= year\_range[0]) *# Year range*

* pd.Series(True, index=df.index) → Creates a mask of True values.
* isin() → Filters rows matching selected values.
* &= → Combines multiple conditions (logical AND).

**Applying Filters**

python

filtered = df[mask].copy() *# Create filtered DataFrame*

* df[mask] → Returns only rows where mask == True.
* .copy() → Ensures modifications don’t affect the original.

**6. Visualizations (Plotly Express)**

**1. Bar Chart (Content Type Distribution)**

python

fig\_bar = px.bar(

x=type\_counts.index,

y=type\_counts.values,

color=type\_counts.index, *# Color by type*

title="Movies vs TV Shows"

)

* px.bar() → Creates a bar chart.
* x**&**y → Data for axes.
* color → Differentiates Movies/TV Shows.

**2. Pie Chart (Content Share)**

python

fig\_pie = px.pie(

names=type\_counts.index,

values=type\_counts.values,

hole=0.4, *# Donut chart*

color\_discrete\_sequence=['#E50914', '#F5F5F1'] *# Netflix colors*

)

* px.pie() → Creates a pie/donut chart.
* hole=0.4 → Makes it a donut.
* color\_discrete\_sequence → Custom colors.

**3. Histogram (Release Year Distribution)**

python

fig\_hist = px.histogram(

filtered,

x="release\_year",

color="type", *# Group by Movies/TV Shows*

nbins=30, *# Number of bins*

barmode="group" *# Side-by-side bars*

)

* px.histogram() → Shows distribution over time.
* nbins → Controls granularity.
* barmode="group" → Groups bars by type.

**4. Choropleth Map (Global Distribution)**

python

fig\_map = px.choropleth(

country\_title\_count,

locations='country', *# Country names*

locationmode='country names', *# Geographic mode*

color='num\_titles', *# Color intensity by count*

color\_continuous\_scale='Reds' *# Netflix-themed*

)

* px.choropleth() → World map visualization.
* locationmode='country names' → Matches country names.
* color\_continuous\_scale → Red gradient for Netflix style.

**7. Key Pandas & NumPy Functions Used**

| **Function** | **Purpose** | **Example** |
| --- | --- | --- |
| pd.read\_csv() | Load CSV into DataFrame | df = pd.read\_csv('data.csv') |
| df.dropna() | Remove missing values | df.dropna(subset=['rating']) |
| df.fillna() | Replace missing values | df['country'].fillna('Unknown') |
| pd.to\_datetime() | Convert to datetime | df['date\_added'] = pd.to\_datetime(...) |
| df.groupby() | Group data for aggregation | df.groupby('type').size() |
| np.where() | Conditional operations | np.where(df['type'] == 'Movie', 1, 0) |
| pd.Series.str.split() | Split strings into lists | df['country'].str.split(', ') |

**8. Streamlit UI Components**

**1. Columns Layout**

python

col1, col2 = st.columns(2) *# Split into 2 columns*

with col1:

st.plotly\_chart(fig\_bar) *# Chart in left column*

with col2:

st.plotly\_chart(fig\_pie) *# Chart in right column*

* st.columns() → Creates a multi-column layout.
* with col1: → Places content in the first column.

**2. Metric Cards**

python

st.metric(

label="Total Titles",

value=f"{len(filtered):,}", *# Formatted count*

delta="+10%" *# Optional change indicator*

)

* st.metric() → Displays a key metric in a card.

**3. Expander (Collapsible Section)**

python

with st.expander("📋 Data Table"):

st.dataframe(filtered.head(100)) *# Show first 100 rows*

* st.expander() → Creates a toggleable section

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------\*\***Pandas: A Simple Explanation\*\***

Pandas is like a \*\*super-powered Excel for Python\*\*. It helps you \*\*work with data easily\*\*—whether it's numbers, names, dates, or anything else.

### \*\*Who Made Pandas?\*\*

- Created in \*\*2008 by Wes McKinney\*\*, a programmer who wanted better tools for analyzing financial data.

- He was frustrated that Python didn’t have something as good as \*\*R (a language for statistics)\*\*, so he built Pandas!

### \*\*Why Was Pandas Created?\*\*

- Before Pandas, working with tables of data in Python was \*\*slow and messy\*\*.

- Pandas made it \*\*fast and simple\*\*, like using Excel but with coding.

### \*\*How Did Pandas Grow?\*\*

1. \*\*2009-2012\*\*: First versions came out, adding key features like:

- \*\*DataFrames\*\* (tables with rows & columns).

- \*\*Easy filtering\*\* (like Excel filters but way stronger).

2. \*\*2013-2015\*\*: Became \*\*super popular\*\* in data science.

- Used by big companies (\*\*Google, Facebook, banks\*\*).

- Got faster and could handle \*\*bigger datasets\*\*.

3. \*\*2016-Today\*\*: Now used \*\*everywhere\*\*—AI, business, research.

- Keeps improving (\*\*faster loading, better missing data handling\*\*).

### \*\*Why Do People Love Pandas?\*\*

✅ \*\*Easy to use\*\* (like Excel, but for coders).

✅ \*\*Works with all kinds of data\*\* (CSV, Excel, databases).

✅ \*\*Fast\*\* (thanks to smart programming tricks).

✅ \*\*Used by everyone\*\* (data scientists, analysts, programmers).

### \*\*Will Pandas Stay Popular?\*\*

Yes! Even though new tools (like \*\*Polars\*\*) are coming, Pandas is still the \*\*#1 choice\*\* because:

- It’s \*\*easy to learn\*\*.

- It works with \*\*almost every data tool\*\*.

- It’s \*\*always getting better\*\*.

### \*\*In Short\*\*

Pandas is \*\*the best tool for playing with data in Python\*\*, and it’s not going away anytime soon! 🚀

Want to see how it works? Try it—you’ll love it! 😊

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------# \*\*NumPy: A Simple Explanation\*\*

NumPy (short for \*Numerical Python\*) is like a \*\*super-powered calculator\*\* for Python. It helps you work with \*\*numbers, lists, and tables of data\*\* really fast—much faster than regular Python!

---

## \*\*Who Made NumPy?\*\*

- Created in \*\*2005 by Travis Oliphant\*\*, who wanted Python to be better at math and science.

- Built on an older tool called \*\*Numeric\*\*, which was too slow for big calculations.

---

## \*\*Why Was NumPy Created?\*\*

- Before NumPy, doing math in Python was \*\*slow\*\* (especially with big datasets).

- NumPy made it \*\*fast\*\* by using \*\*C\*\* (a speedy programming language) behind the scenes.

- It also made working with \*\*arrays (lists of numbers)\*\* super easy.

---

## \*\*What Can NumPy Do?\*\*

### 1. \*\*Fast Math Operations\*\*

- Works with \*\*big lists of numbers\*\* in one go (no slow loops!).

- Example:

```python

import numpy as np

numbers = np.array([1, 2, 3, 4])

print(numbers \* 2) # Output: [2, 4, 6, 8] (super fast!)

```

### 2. \*\*Multi-Dimensional Arrays\*\*

- Handles \*\*tables (2D), cubes (3D), and more\*\* easily.

- Example:

```python

table = np.array([[1, 2], [3, 4]]) # A 2x2 table

print(table + 10) # Adds 10 to every number

```

### 3. \*\*Scientific & AI Tools\*\*

- Used in \*\*machine learning (TensorFlow, PyTorch)\*\*.

- Helps with \*\*statistics, physics, and engineering\*\*.

- Example (calculating averages):

```python

data = np.random.rand(100) # 100 random numbers

print(np.mean(data)) # Finds the average instantly

```

---

## \*\*Why Do People Love NumPy?\*\*

✅ \*\*Blazing fast\*\* (up to \*\*100x faster\*\* than regular Python lists).

✅ \*\*Simple syntax\*\* (easy to learn, hard to forget).

✅ \*\*Works with Pandas, SciPy, and AI tools\*\*.

✅ \*\*Used by NASA, Google, and scientists worldwide\*\*.

---

## \*\*NumPy vs. Regular Python Lists\*\*

| Feature | NumPy Arrays | Python Lists |

|------------------|--------------------|--------------------|

| \*\*Speed\*\* | ⚡ Super fast | 🐢 Slow for big data |

| \*\*Math Operations\*\* | ✅ Easy (e.g., `array \* 2`) | ❌ Needs loops |

| \*\*Memory Usage\*\* | 🏎️ Efficient | 🚗 Wastes space |

---

## \*\*Fun Fact\*\*

Did you know \*\*Pandas (for data tables) is built on NumPy\*\*? That’s why both are best friends in data science! 🤝

---

## \*\*Try It Yourself!\*\*

```python

import numpy as np

numbers = np.array([5, 10, 15])

print(numbers \*\* 2) # Squares each number: [25, 100, 225]

```

NumPy is \*\*the secret sauce\*\* behind almost every data science and AI project. Want to work with numbers in Python? \*\*Learn NumPy first!\*\* 🚀

Need help with something specific? Ask away! 😊