```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
from collections import Counter
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans

sns.set(style="whitegrid")


from google.colab import files
uploaded = files.upload()  # choose your CSV here
```

Choose Files Netflix_mov…ustering.csv

- **Netflix_movies_and_tv_shows_clustering.csv**(text/csv) - 2936713 bytes, last modified: 8/14/2025 - 100% done

```
Saving Netflix_movies_and_tv_shows_clustering.csv to Netflix_movies_and_tv_shows_clustering.csv
```

```
csv_path = list(uploaded.keys())[0]
df = pd.read_csv(csv_path)
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	14-Aug-20	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	23-Dec-16	2016	TV-MA	93 min	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	20-Dec-18	2011	R	78 min	Horror Movies, International Movies	When an army recruit is found dead, his fellow...

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.shape, df.columns.to_list()
```

```
((7787, 12),
 ['show_id',
  'type',
  'title',
  'director',
  'cast',
  'country',
  'date_added',
  'release_year',
  'rating',
  'duration',
  'listed_in',
  'description'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       7787 non-null   object
 1   type          7787 non-null   object
 2   title         7787 non-null   object
 3   director      5398 non-null   object
 4   cast          7069 non-null   object
 5   country       7280 non-null   object
 6   date_added    7777 non-null   object
```

```
 7   release_year  7787 non-null   int64
 8   rating        7780 non-null   object
 9   duration      7787 non-null   object
 10  listed_in     7787 non-null   object
 11  description   7787 non-null   object
dtypes: int64(1), object(11)
memory usage: 730.2+ KB
```

```
df.isnull().sum().sort_values(ascending=False)
```

	0
director	2389
cast	718
country	507
date_added	10
rating	7
title	0
show_id	0
type	0
release_year	0
duration	0
listed_in	0
description	0

dtype: int64

```
df = df.drop_duplicates().copy()
for c in df.select_dtypes(include='object').columns:
    df[c] = df[c].fillna("Unknown")
df.head(2)
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	Unknown	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	14-Aug-20	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
				Jorge	Demián						Dramas,	After a

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
if 'date_added' in df.columns:
    df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
    df['year_added'] = df['date_added'].dt.year
    df['month_added'] = df['date_added'].dt.month
df[['date_added','year_added','month_added']].head(3) if 'date_added' in df.columns else "date_added not present"
```

```
/tmp/ipython-input-2111463386.py:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `d
  df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

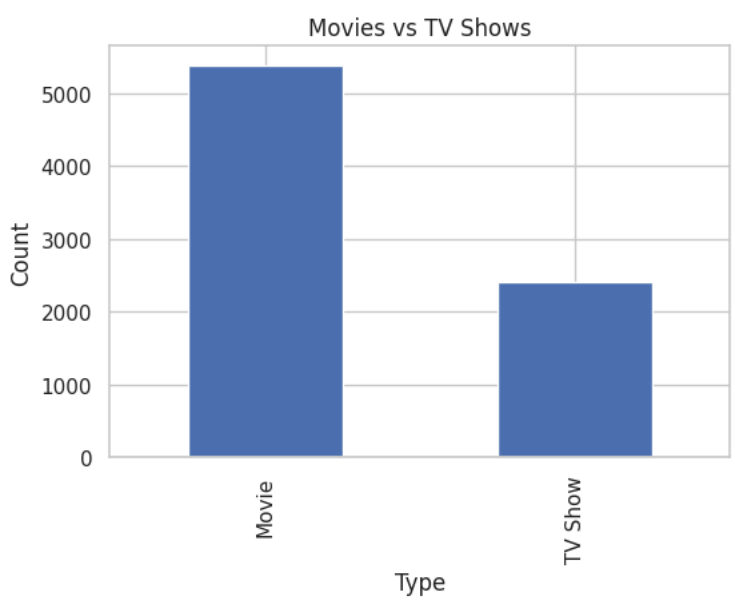
	date_added	year_added	month_added
0	2020-08-14	2020.0	8.0
1	2016-12-23	2016.0	12.0
2	2018-12-20	2018.0	12.0

```
df['type'].value_counts()
```

type	count
Movie	5377
TV Show	2410

dtype: int64

```
plt.figure(figsize=(6,4))
df['type'].value_counts().plot(kind='bar')
plt.title('Movies vs TV Shows')
plt.xlabel('Type'); plt.ylabel('Count')
plt.show()
```

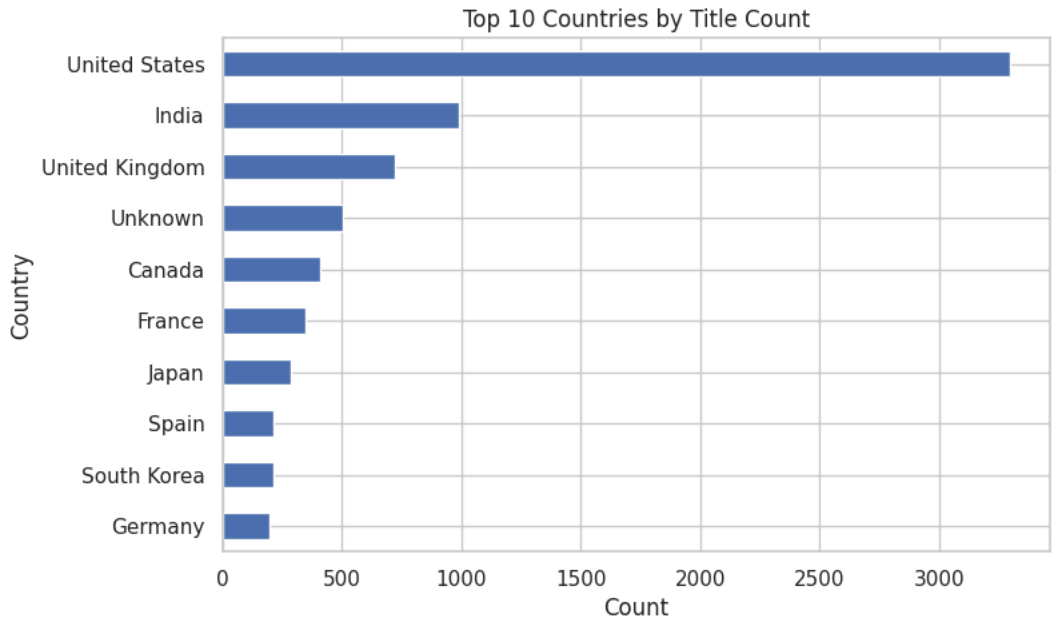



```
countries = df['country'].astype(str).str.split(',').explode().str.strip()
top_countries = countries.value_counts().head(10)
top_countries
```

country	count
United States	3297
India	990
United Kingdom	723
Unknown	507
Canada	412
France	349
Japan	287
Spain	215
South Korea	212
Germany	199

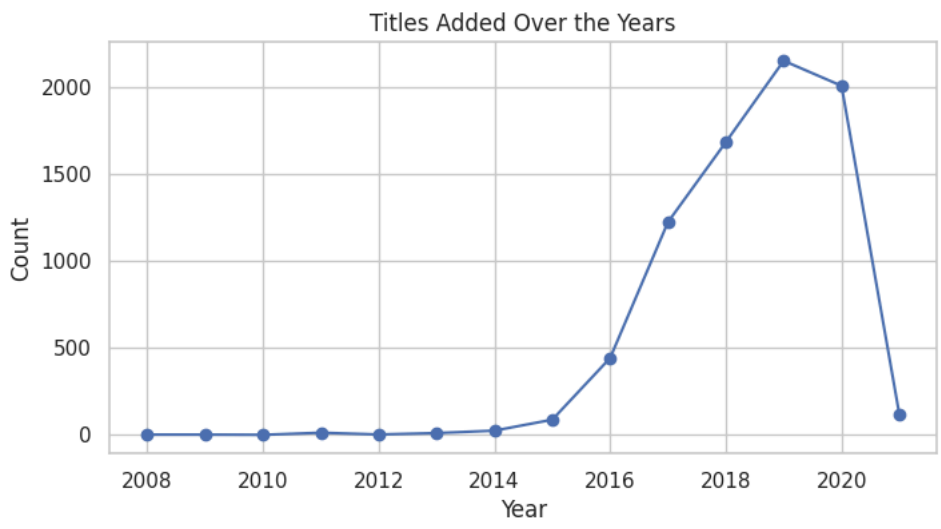
dtype: int64

```
plt.figure(figsize=(8,5))
top_countries.plot(kind='barh')
plt.title('Top 10 Countries by Title Count')
plt.xlabel('Count'); plt.ylabel('Country')
plt.gca().invert_yaxis()
plt.show()
```



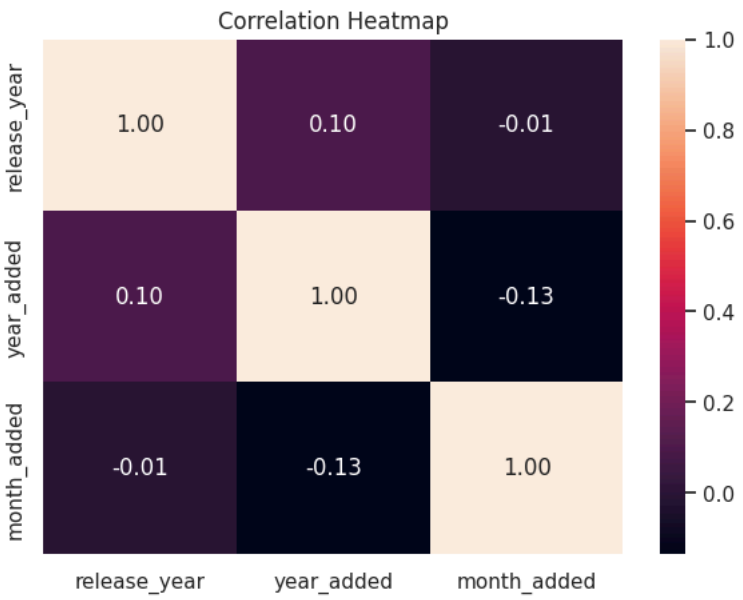

```
if 'year_added' in df.columns:
    yearly = df['year_added'].dropna().value_counts().sort_index()
    yearly
else:
    "year_added not available"
```

```
if 'year_added' in df.columns:
    plt.figure(figsize=(8,4))
    yearly.plot(kind='line', marker='o')
    plt.title('Titles Added Over the Years')
    plt.xlabel('Year'); plt.ylabel('Count')
    plt.show()
else:
    print("year_added not available")
```




```
genre_text = ' '.join(df['listed_in'].astype(str))
wc = WordCloud(width=800, height=400, background_color='black').generate(genre_text)
plt.figure(figsize=(10,5))
plt.imshow(wc, interpolation='bilinear'); plt.axis('off')
plt.title('Most Common Genres (Word Cloud)')
plt.show()
```

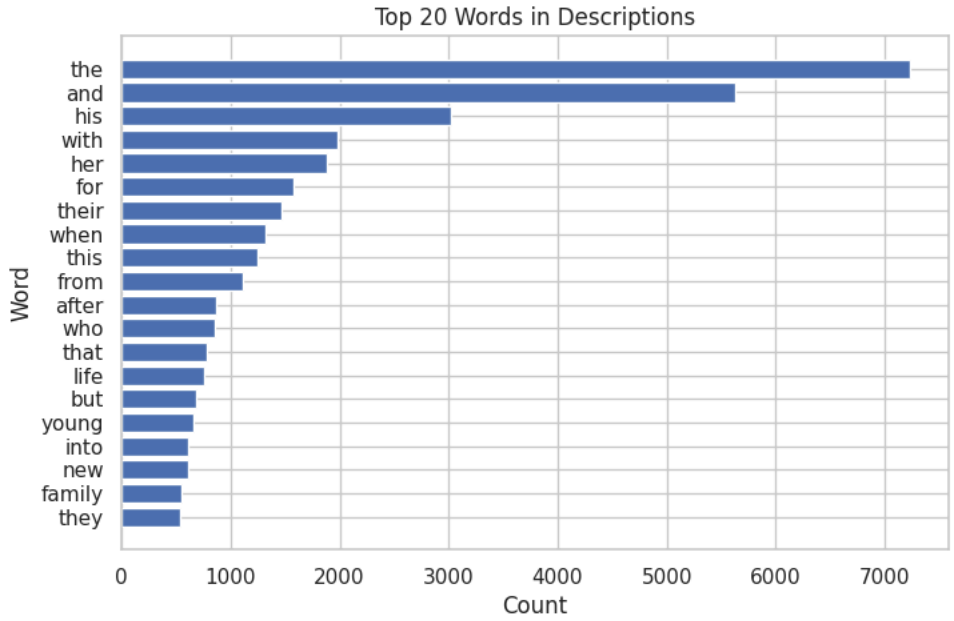

```
num_df = df.select_dtypes(include=np.number)
if num_df.shape[1] > 0:
    plt.figure(figsize=(7,5))
    sns.heatmap(num_df.corr(), annot=True, fmt=".2f")
    plt.title('Correlation Heatmap')
    plt.show()
else:
    print("No numeric columns available for correlation.")
```




```
text_all = ' '.join(df['description'].astype(str))
tokens = re.findall(r'\b[a-z]{3,}\b', text_all.lower())
common20 = Counter(tokens).most_common(20)
pd.DataFrame(common20, columns=['word','count'])
```

	word	count
0	the	7226
1	and	5629
2	his	3020
3	with	1976
4	her	1884
5	for	1579
6	their	1472
7	when	1325
8	this	1246
9	from	1114
10	after	865
11	who	856
12	that	781
13	life	757
14	but	684
15	young	655
16	into	615
17	new	613
18	family	553
19	they	535

```
word_df = pd.DataFrame(common20, columns=['word','count'])
plt.figure(figsize=(8,5))
plt.barh(word_df['word'][::-1], word_df['count'][::-1])
plt.title('Top 20 Words in Descriptions')
plt.xlabel('Count'); plt.ylabel('Word')
plt.show()
```




```
mask = df['description'].astype(str) != "Unknown"
desc_series = df.loc[mask, 'description'].astype(str)

tfidf = TfidfVectorizer(stop_words='english', max_df=0.8, min_df=5)
X = tfidf.fit_transform(desc_series)

kmeans = KMeans(n_clusters=5, random_state=42, n_init="auto")
```

```
labels = kmeans.fit_predict(X)

df['cluster'] = -1
df.loc[mask, 'cluster'] = labels
df['cluster'].value_counts().sort_index()
```

cluster	count
0	515
1	5557
2	410
3	803
4	502

dtype: int64

```
for k in sorted(df['cluster'].unique()):
    print(f"\nCluster {k}:")
    titles = df.loc[df['cluster']==k, 'title'].head(5).tolist()
    print(titles)
```

```
Cluster 0:
['#Alive', '#FriendButMarried', '10 Days in Sun City', '100 Meters', '100% Halal']

Cluster 1:
['3%', '23:59', '9', '21', '46']

Cluster 2:
['15-Aug', '#Roxy', '100 Days My Prince', '21 Thunder', '365 Days']

Cluster 3:
['7:19', '122', '187', '1922', '1994']

Cluster 4:
['#blackAF', '12 Years Promise', '14 Cameras', '20 Minutes', '3 Türken & ein Baby']
```

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

```
'Netflix_cleaned_with_clusters.csv'
```