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

 #encoding='cp1252'
n= pd.read_excel("./Data/netflix_titles.xlsx")
n.head(1)
```

```
----------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Cell In[2], line 2
      1 #encoding='cp1252'
----> 2 n= pd.read_excel("./Data/netflix_titles.xlsx")
      3 n.head(1)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\util\_decorators.py:211, in
deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args,
**kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\util\_decorators.py:331, in
deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*arg
s, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327
msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\io\excel\_base.py:482, in
read_excel(io, sheet_name, header, names, index_col, usecols, squeeze,
dtype, engine, converters, true_values, false_values, skiprows, nrows,
na_values, keep_default_na, na_filter, verbose, parse_dates,
date_parser, thousands, decimal, comment, skipfooter, convert_float,
mangle_dupe_cols, storage_options)
    480 if not isinstance(io, ExcelFile):
```

```
    481     should_close = True
--> 482     io = ExcelFile(io, storage_options=storage_options,
engine=engine)
    483 elif engine and engine != io.engine:
    484     raise ValueError(
    485         "Engine should not be specified when passing "
    486         "an ExcelFile - ExcelFile already has the engine set"
    487     )

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\io\excel\_base.py:1652, in
ExcelFile.__init__(self, path_or_buffer, engine, storage_options)
   1650     ext = "xls"
   1651 else:
-> 1652     ext = inspect_excel_format(
   1653         content_or_path=path_or_buffer,
storage_options=storage_options
   1654     )
   1655     if ext is None:
   1656         raise ValueError(
   1657             "Excel file format cannot be determined, you must
specify "
   1658             "an engine manually."
   1659         )

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\io\excel\_base.py:1525, in
inspect_excel_format(content_or_path, storage_options)
   1522 if isinstance(content_or_path, bytes):
   1523     content_or_path = BytesIO(content_or_path)
-> 1525 with get_handle(
   1526     content_or_path, "rb", storage_options=storage_options,
is_text=False
   1527 ) as handle:
   1528     stream = handle.handle
   1529     stream.seek(0)

File ~\AppData\Local\Packages\
PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-
packages\Python310\site-packages\pandas\io\common.py:865, in
get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    856         handle = open(
    857             handle,
    858             ioargs.mode,
   (...)
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
--> 865         handle = open(handle, ioargs.mode)
    866     handles.append(handle)
    868 # Convert BytesIO or file objects passed with an encoding

FileNotFoundError: [Errno 2] No such file or directory:
'./Data/netflix_titles.xlsx'
```

```
# Convert 'DateColumn' to datetime format
n['ExtractedDate'] = pd.to_datetime(n['date_added'], format='%B %d,
%Y')
n = n.dropna(subset=['ExtractedDate'])

n['ExtractedDate']=n['ExtractedDate'].dt.year
n['ExtractedDate'].isnull().sum()
#n.head(1)
```

```
0
```

```
content_freq_year=[2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2
018,2019,2020,2021]
n2=n[n.ExtractedDate.isin(content_freq_year)]
n2 = n2.groupby(["ExtractedDate", "type"])["type"].count()
n2 = n2.unstack()
n2
```

```
type             Movie  TV Show
ExtractedDate
2008               1.0      1.0
2009               2.0      NaN
2010               1.0      NaN
2011              13.0      NaN
2012               3.0      NaN
2013               6.0      5.0
2014              19.0      5.0
2015              56.0     26.0
2016             253.0    176.0
2017             839.0    349.0
2018            1237.0    412.0
2019            1424.0    592.0
2020            1284.0    595.0
2021             993.0    505.0
```

```
n3=n[n.ExtractedDate.isin(content_freq_year)]
n3.isnull().sum()
n3=n3[n3["country"].isnull() ==False]
```

```
#n.isnull().sum()
n3.isnull().sum()
```

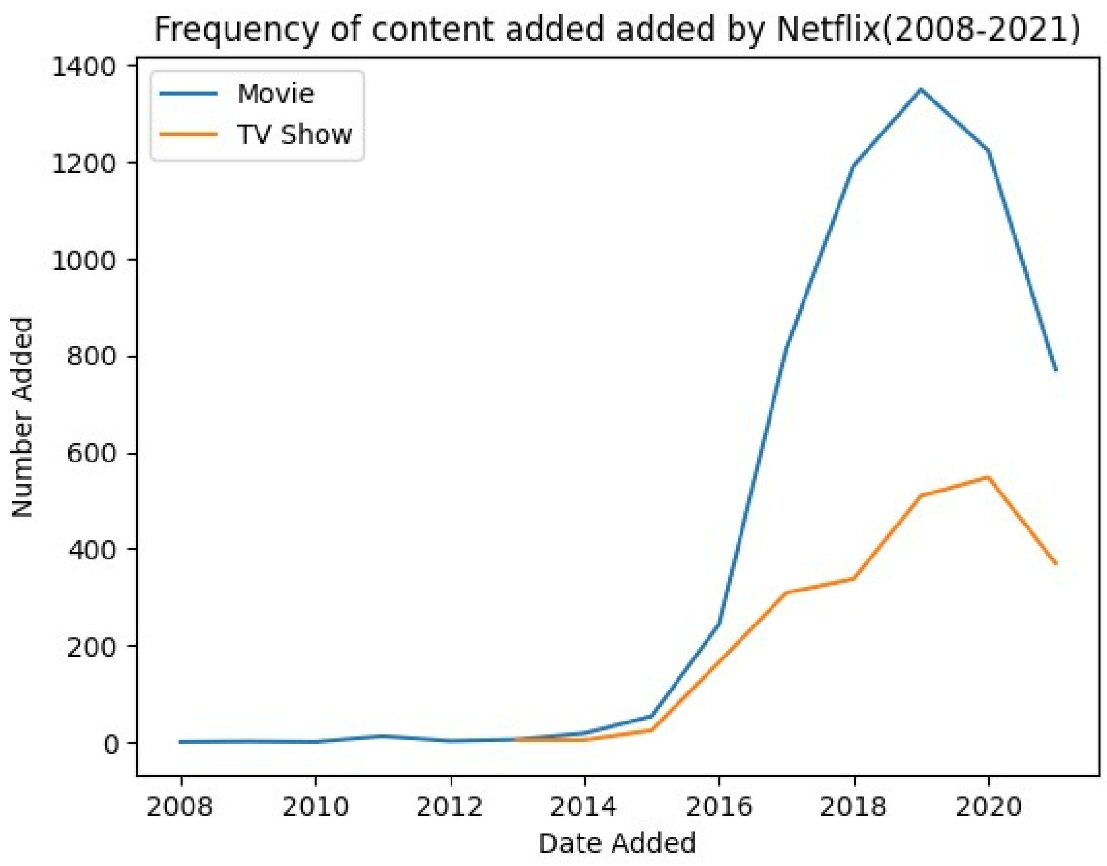
```
show_id              0
type                 0
title                0
director          2216
cast               671
country              0
date_added           0
release_year         0
rating               3
duration             3
listed_in            0
description          0
ExtractedDate        0
dtype: int64
```

```
n3= n3.groupby(["ExtractedDate", "type"])["type"].count()
n3 = n3.unstack()
n3
```

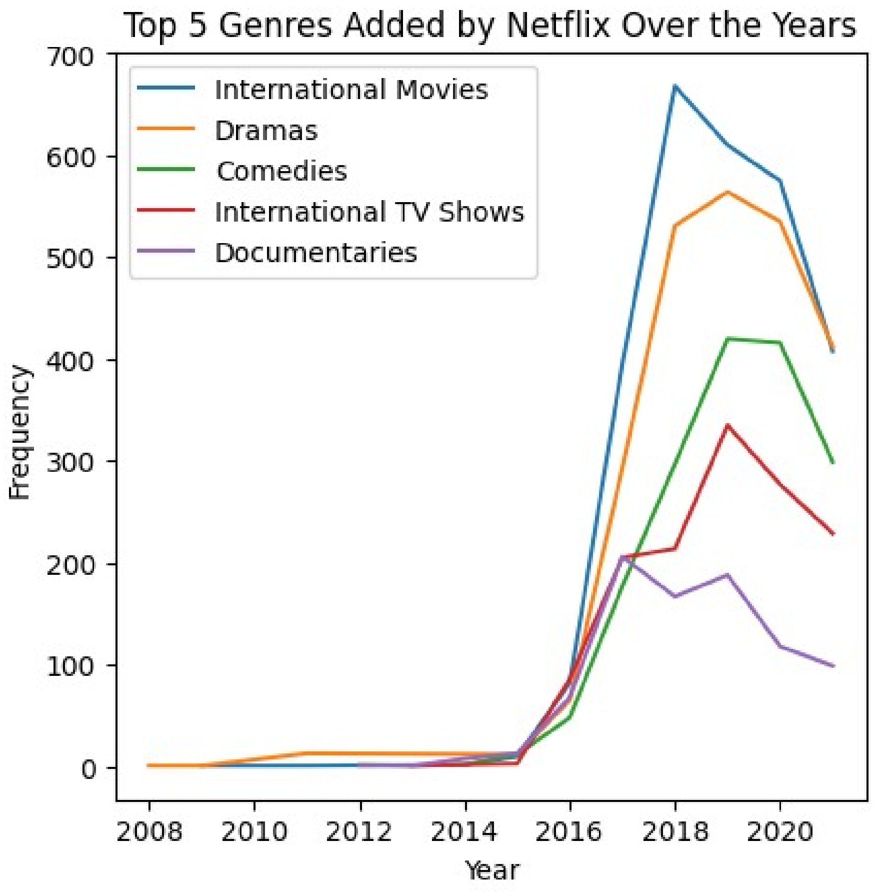
```
type            Movie  TV Show
ExtractedDate
2008              1.0      1.0
2009              2.0      NaN
2010              1.0      NaN
2011             13.0      NaN
2012              3.0      NaN
2013              6.0      5.0
2014             19.0      5.0
2015             54.0     25.0
2016            244.0    166.0
2017            814.0    309.0
2018           1192.0    338.0
2019           1349.0    509.0
2020           1223.0    548.0
2021            770.0    370.0
```

```
#n2.plot.line(marker='o')
n3.plot.line()
# Adding labels and title
plt.xlabel('Date Added')
plt.ylabel('Number Added')
plt.title('Frequency of content added added by Netflix(2008-2021)')

# Adding a legend
plt.legend()
```

```
# Display the plot
plt.show()
```

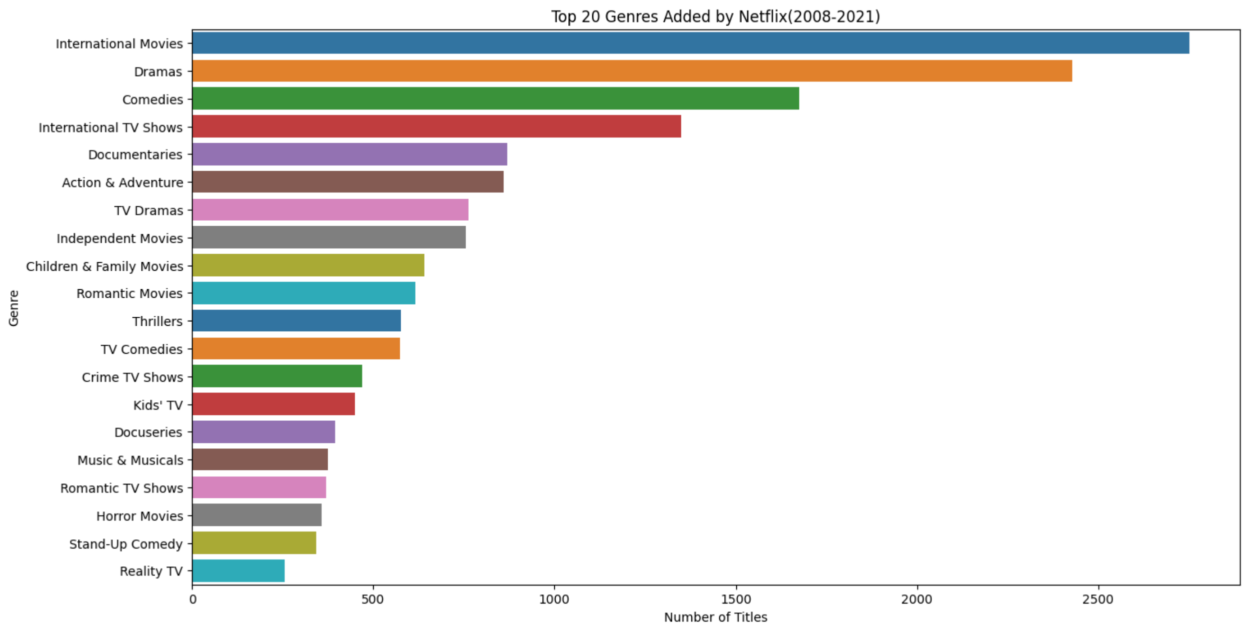


```
# Assuming "listed_in" contains lists
content_freq_year = [2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
2016, 2017, 2018, 2019, 2020, 2021]
n4 = n[n['ExtractedDate'].isin(content_freq_year)]

# Splitting "listed_in" values and exploding the DataFrame
n4['listed_in'] = n4['listed_in'].apply(lambda x: [item.strip() for
item in x.split(',')])
n4 = n4.explode('listed_in')

# Counting occurrences of each genre
genre_counts = n4['listed_in'].value_counts()

# Selecting the top 5 genres
top_5_genres = genre_counts.head(5)
plt.figure(figsize=([5,5]))
# Plotting a line plot for each genre
for genre in top_5_genres.index:
    genre_data = n4[n4['listed_in'] ==
genre].groupby('ExtractedDate').size()
    plt.plot(genre_data.index, genre_data.values, label=genre)

# Adding labels and title
plt.xlabel('Year')
plt.ylabel('Frequency')
plt.title('Top 5 Genres Added by Netflix Over the Years')

# Adding a legend
plt.legend()

# Displaying the plot
plt.show()
```



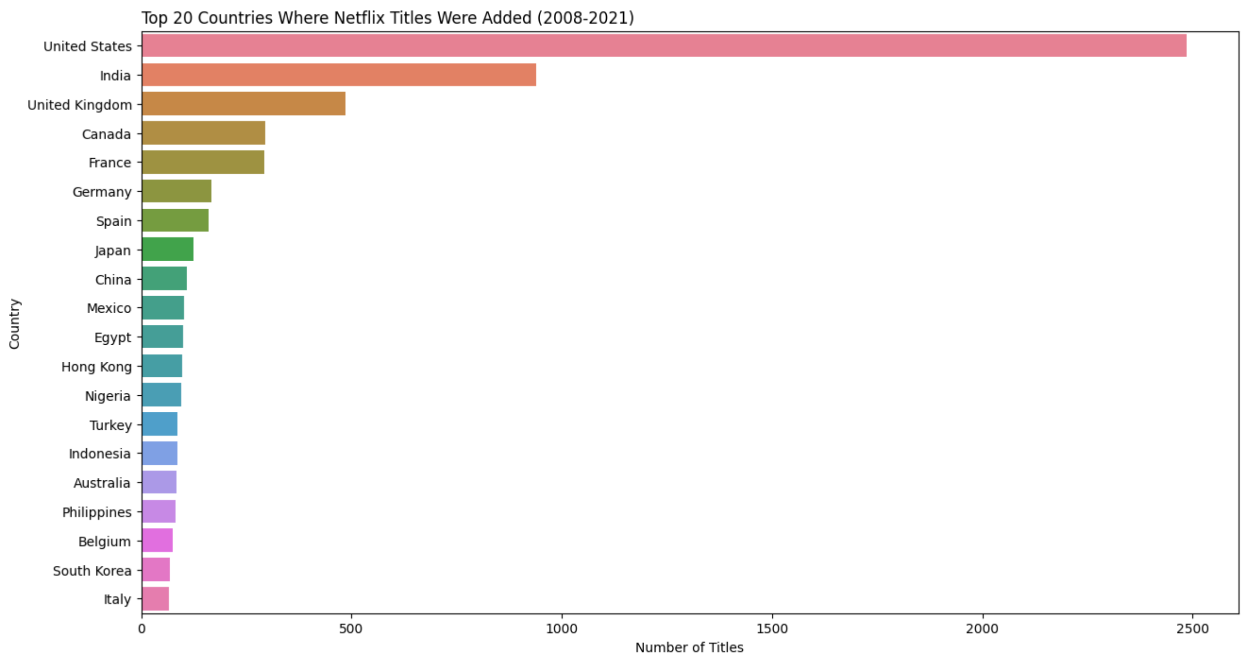
```
# Assuming "listed_in" contains lists
plt.figure(figsize=(15, 8))

content_freq_year = [2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
2016, 2017, 2018, 2019, 2020, 2021]
n4 = n[n['ExtractedDate'].isin(content_freq_year)]
```

```
# Splitting "listed_in" values and exploding the DataFrame
n4['listed_in'] = n4['listed_in'].apply(lambda x: [item.strip() for
item in x.split(',')])
n4 = n4.explode('listed_in')

# Counting occurrences of each genre
genre_counts = n4['listed_in'].value_counts()

# Selecting the top 20 genres and sorting in descending order
top_20_genres = genre_counts.head(20).sort_values(ascending=False)

# Creating a horizontal bar plot with different colors for each bar
sns.barplot(x=top_20_genres.values, y=top_20_genres.index,
palette=sns.color_palette("tab10")
)
# Adding labels and title
plt.xlabel('Number of Titles')
plt.ylabel('Genre')
plt.title('Top 20 Genres Added by Netflix(2008-2021)')
# Displaying the plot
plt.show()
```



```
#Top 20 Countries where Netflix Titles Added From(2008 -2021)

# Assuming "country" column contains information about the countries
plt.figure(figsize=(15, 8))

content_freq_year = [2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
2016, 2017, 2018, 2019, 2020, 2021]
n4 = n[n['ExtractedDate'].isin(content_freq_year)]
```

```
n4 =n4.dropna()
n4['SplitCountries'] = n4['country'].str.split(',').apply(lambda x:
[country.strip() for country in x])
n4 = n4.explode('SplitCountries')



# Counting occurrences of each country
country_counts = n4['SplitCountries'].value_counts()

# Selecting the top 20 countries and sorting in descending order
top_20_countries =
country_counts.head(20).sort_values(ascending=False)

# Custom color palette
custom_palette = sns.color_palette("husl", len(top_20_countries))

# Creating a horizontal bar plot with custom colors
sns.barplot(x=top_20_countries.values, y=top_20_countries.index,
palette=custom_palette)

# Adding labels and title
plt.xlabel('Number of Titles')
plt.ylabel('Country')
plt.title('Top 20 Countries Where Netflix Titles Were Added (2008-
2021)',loc='left')

# Displaying the plot
plt.show()
```


Top 20 Countries Where Netflix Titles Were Added (2008-2021)
Country
United States
India
United Kingdom
Canada
France
Germany
Spain
Japan
China
Mexico
Egypt
Hong Kong
Nigeria
Turkey
Indonesia
Australia
Philippines
Belgium
South Korea
Italy
Number of Titles