```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'exploratory-data-analysis-on-netflix-data:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F280

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading exploratory-data-analysis-on-netflix-data, 1984591 bytes compressed
[==================================================] 1984591 bytes downloaded
Downloaded and uncompressed: exploratory-data-analysis-on-netflix-data
Data source import complete.
```

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
```

```
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a versi
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/exploratory-data-analysis-on-netflix-data/netflix img.png
/kaggle/input/exploratory-data-analysis-on-netflix-data/netflix_titles_2021.csv
```

```
import pandas as pd
data=pd.read_csv('/kaggle/input/exploratory-data-analysis-on-netflix-data/netflix_titles_2021.csv')
data.head()
```

	show_id	type	title	director	cast	country	date_added	release_
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	
		TV	Jailbirds				September	

```
data.describe()
```

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

```
data.columns
```

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

```
data.dtypes
```

```
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
```

```
rating          object
duration        object
listed_in       object
description     object
dtype: object
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Double-click (or enter) to edit

```
from matplotlib import pyplot as plt
import seaborn as sns
```

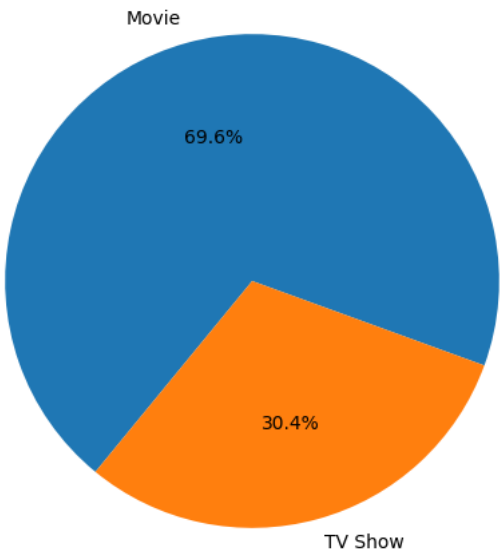
Double-click (or enter) to edit

```
- What types of shows or movies are uploaded on Netflix?
```

```
Object `Netflix` not found.
```

```
Label_types=data["type"].value_counts()
sns.colors=sns.color_palette("pastel")
plt.figure(figsize=(6,6))
plt
plt.pie(Label_types,labels=Label_types.index,autopct="%1.1f%%", startangle=-20)
```

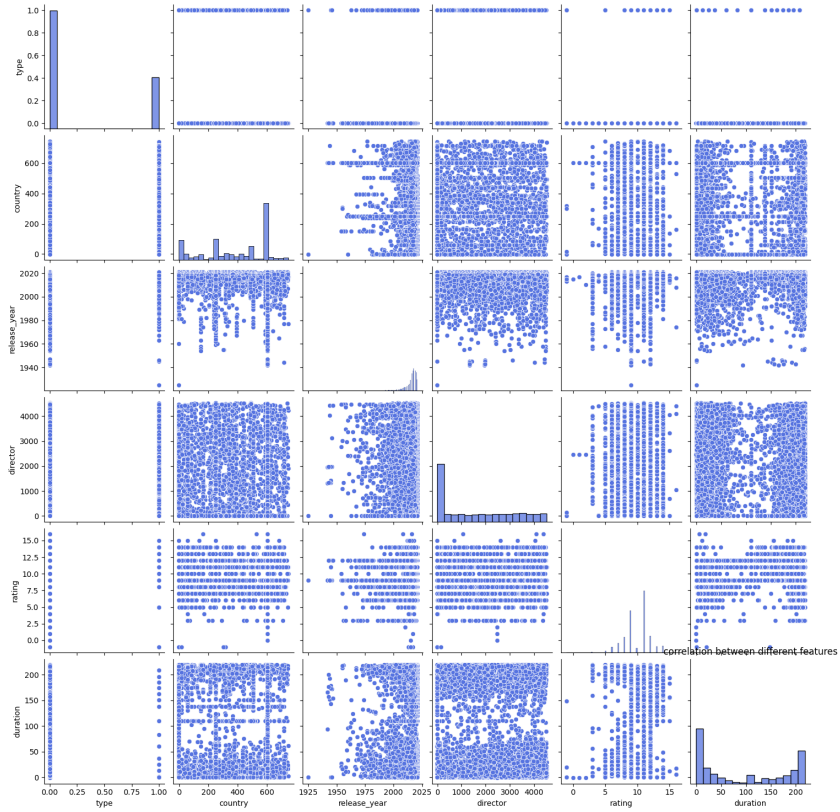
```
([<matplotlib.patches.Wedge at 0x7c19d6bb52a0>,
  <matplotlib.patches.Wedge at 0x7c19d6b646a0>],
 [Text(-0.2903925789508779, 1.0609769790576318, 'Movie'),
  Text(0.2903925789508773, -1.060976979057632, 'TV Show')],
 [Text(-0.1583959521550243, 0.5787147158496173, '69.6%'),
  Text(0.15839595215502397, -0.5787147158496173, '30.4%')])
```



What is the correlation between features?

```
import seaborn as sns
import matplotlib.pyplot as plt

selected_columns = data[["type", "country", "release_year", "director", "rating", "duration"]]
custom_palette = sns.color_palette("coolwarm", 10)
sns.set_palette(custom_palette)
plt.figure(figsize=(15, 15))

# Convert 'object' type columns to numerical if they represent categories
for col in selected_columns:
    if selected_columns[col].dtype == 'object':
        selected_columns[col] = selected_columns[col].astype('category').cat.codes

sns.pairplot(selected_columns)
plt.title("correlation between different features")
plt.show()
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  selected_columns[col] = selected_columns[col].astype('category').cat.code
<ipython-input-63-3db962b23873>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  selected_columns[col] = selected_columns[col].astype('category').cat.code
<ipython-input-63-3db962b23873>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  selected_columns[col] = selected_columns[col].astype('category').cat.code
<ipython-input-63-3db962b23873>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
  selected_columns[col] = selected_columns[col].astype('category').cat.code
<Figure size 1500x1500 with 0 Axes>
```



How many movies have a "TV-14" rating in Canada?

```
tv_14_rating=data[data["rating"]<="TV-14"]
print(tv_14_rating)
```

Show hidden output

```
len(tv_14_rating)
```

Show hidden output

```
sorted_country=data[data["country"]=="India"]
print(sorted_country)
```

Show hidden output

this is giving error since TV-14 is not a seprate column so we are applying wrong code here instead we need to filter rows for the "TV-14" rating

```
print("show titles in India:",sorted_country["title"].value_counts())
```

```
show titles in India: title
Kota Factory             1
Aitraaz                  1
Mumbai Cha Raja          1
Harud                    1
Umrika                   1
                        ..
Oh! Baby                 1
Article 15               1
Care of Kancharapalem    1
Ee Nagaraniki Emaindi    1
Zubaan                   1
Name: count, Length: 972, dtype: int64
```

Filter the rows for TV-14 rating

```
tv_14_rating_count = data[data['rating'] == 'TV-14']
print(tv_14_rating_count)
```

Show hidden output

```
print("Tv-14 ratings movies/shows in Canada:",tv_14_rating_count["country"].value_counts())
```

Show hidden output

```
# What is the show ID and director for 'House of Cards'?
```

Show hidden output

```
House_of_cards=data[data["title"]=="House of Cards"]
House_of_cards
```

Show hidden output

```
List_of_movies=data[data["release_year"]== 2000]
print(List_of_movies)
```

Show hidden output

```
Actor= data[data["cast"]== "Tom Cruise"]
print(Actor)
```

```
Empty DataFrame
Columns: [show_id, type, title, director, cast, country, date_added, release_year, rating, duration, listed_in, descripti
```

```
Index: []
```

Double-click (or enter) to edit

Start coding or generate with AI.

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

```
data["rating"]
```

Show hidden output

```
rating_distribution=data["rating"].apply(lambda x:x if x in["PG","PG-13","TV-14","TV-MA","TV-G"]else"others")
print(rating_distribution)
```

Show hidden output

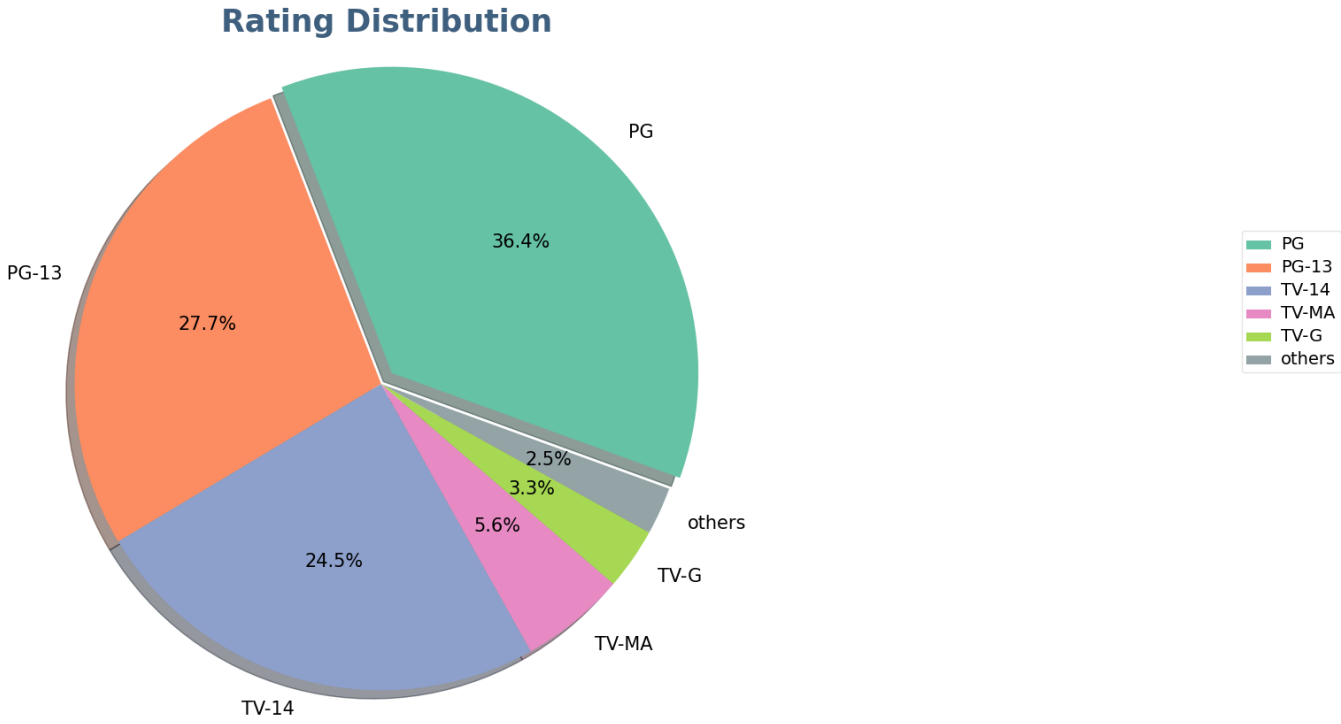
```
rating_data=rating_distribution.value_counts().tolist()
rating_data
```

```
[3207, 2443, 2160, 490, 287, 220]
```

```
labels="PG","PG-13","TV-14","TV-MA","TV-G","others"
len(labels)

sizes=[]
sizes=[]
for i in rating_data:
      percent=(i*100)/len(data["rating"])
      sizes.append(percent)
sizes
```

```
[36.414215964573636,
 27.73929828545475,
 24.525945270807313,
 5.563756103099807,
 3.2587714318156014,
 2.498012944248893]
```

```
fig,ax=plt.subplots(figsize=(10,10))
patches,texts,autotexts=ax.pie(sizes,labels=labels,autopct='%1.1f%%',startangle=-20,
                               shadow=True,explode=(0.05,0,0,0,0,0),
                               colors=sns.color_palette("Set2",8)[:5]+
                               [(0.58,0.64,0.65)],
                               textprops={"fontsize":15,"weight":"light","color":"k"})
ax.axis("equal")
plt.title("Rating Distribution",fontsize=25,pad=-70,weight="bold",
          color=sns.cubehelix_palette(8,start=.5,rot=-.75)[-3])
legend=ax.legend(loc="lower right",framealpha=0.5,bbox_to_anchor=(1.8,0.5,0.1,1))
for text in legend.get_texts():
    text.set_fontsize(14)

plt.show()
```




```
data.columns
```

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'date_added',
       'release_year', 'rating', 'duration', 'listed_in', 'description'],
      dtype='object')
```

```
no_of_years=data["release_year"].unique()
no_of_years
len(no_of_years)
```

```
74
```

```
movies_made_per_year=data["release_year"].unique()/len(no_of_years)
movies_made_per_year
```

```
array([27.2972973 , 27.31081081, 26.93243243, 27.27027027, 26.97297297,
       27.        , 26.98648649, 27.16216216, 27.2027027 , 27.25675676,
       26.68918919, 26.72972973, 26.7972973 , 26.85135135, 27.18918919,
       27.04054054, 27.21621622, 27.05405405, 27.06756757, 27.08108108,
       27.17567568, 27.13513514, 27.14864865, 27.12162162, 27.09459459,
       27.10810811, 26.94594595, 27.22972973, 27.28378378, 27.24324324,
       26.78378378, 26.87837838, 26.89189189, 26.90540541, 27.01351351,
       26.83783784, 26.91891892, 26.81081081, 26.75675676, 26.5       ,
       27.02702703, 26.95945946, 26.82432432, 26.7027027 , 26.47297297,
       26.86486486, 26.77027027, 26.64864865, 26.54054054, 26.28378378,
       26.40540541, 26.74324324, 26.45945946, 26.43243243, 26.52702703,
       26.62162162, 26.66216216, 26.01351351, 26.67567568, 26.48648649,
       26.56756757, 26.63513514, 26.51351351, 26.60810811, 26.71621622,
       26.58108108, 26.59459459, 26.55405405, 26.2972973 , 26.24324324,
       26.41891892, 26.27027027, 26.31081081, 26.25675676])
```

Movies released Per year

```
# Calculate the total number of years (span)
no_of_years_span = data['release_year'].max() - data['release_year'].min() + 1

# Calculate the average number of movies made per year
Avg_movies_made_per_year = len(data) / no_of_years_span

# Print the result
print(Avg_movies_made_per_year)
```

```
90.79381443298969
```

total number of unique years are 74 so length of data

```
len(data) "no_ of _movies made over 74 years "
```

```
8807
```

```
films_made_per_year=len(data)/len(no_of_years)
films_made_per_year
```

```
119.01351351351352
```

```
best_month=data["date_added"].max()
best_month
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-51-191c3e5284de> in <cell line: 1>()
----> 1 best_month=data["date_added"].max()
      2 best_month

                              7 frames
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py in _amax(a, axis, out, keepdims, initial, where)
     39 def _amax(a, axis=None, out=None, keepdims=False,
     40           initial=_NoValue, where=True):
---> 41     return umr_maximum(a, axis, None, out, keepdims, initial, where)
     42
     43 def _amin(a, axis=None, out=None, keepdims=False,

TypeError: '>=' not supported between instances of 'str' and 'float'
```

```
# Convert 'date_added' to datetime objects if it's not already
data['date_added'] = pd.to_datetime(data['date_added'])

best_month = data["date_added"].max()
print(best_month)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-52-06335be22d35> in <cell line: 2>()
      1 # Convert 'date_added' to datetime objects if it's not already
----> 2 data['date_added'] = pd.to_datetime(data['date_added'])
      3
      4 best_month = data["date_added"].max()
      5 print(best_month)

                              5 frames
/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslibs/strptime.pyx in
pandas._libs.tslibs.strptime.array_strptime()

ValueError: time data " August 4, 2017" doesn't match format "%B %d, %Y", at position 1442. You might want to try:
    - passing `format` if your strings have a consistent format;
    - passing `format='ISO8601'` if your strings are all ISO8601 but not necessarily in exactly the same format;
    - passing `format='mixed'`, and the format will be inferred for each element individually. You might want to use
`dayfirst` alongside this.
```

```
# Convert 'date_added' to datetime objects, handling errors
data['date_added'] = pd.to_datetime(data['date_added'], errors='coerce')

best_month = data["date_added"].max()
print(best_month)
```

```
2021-09-25 00:00:00
```

```
unique_directors=data["director"].unique()
unique_directors
len(unique_directors)
```

```
4529
```