

1. IMPORTAR BASE DE DATOS

(Entorno Pandas y Numpy)

```
# IMPORTAR LIBRERÍAS NECESARIAS
import pandas as pd      #Manejo de data estructurada (Dataframe)
import numpy as np       #Manejo de matrices
print(pd.__version__)
```

2.2.2

Breakdown of this notebook:

1. Importing Libraries

2. Data Cleaning

3. Deleting redundant columns

- Dropping duplicates. (Eliminación de duplicados.)
- Cleaning individual columns. (Limpieza de columnas individuales.)
- Remove the NaN values from the dataset (Eliminar los valores NaN del conjunto de datos)
- Some Transformations (Algunas transformaciones)

4. Data Visualization: Using plots to find relations between the features

- Type: Movie and TV Shows (Tipo de datos: Películas y programas de TV)
- Rating (Clasificación)
- Relation between Type and Rating (Relación entre tipo y clasificación)

5. Word Cloud

- Country (País)
- Cast (Reparto)
- Director (Director)
- Category (Categoría)

```
## 2.IMPORTAR BASE DE DATOS
##### VIDEOGAMES SALES ANALYSUS & VISUALIZATION
# Crear DataFrame de Pandas
netflix_titles = "/content/netflix_titles.csv"
df_net = pd.read_csv(netflix_titles)

df_net.head(2) # Primera y últimas filas
```



	show_id	type	title	director	cast	country	date_added	release_year
0	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson,	United States, India, South Korea, China	September 9, 2019	2019

Pasos siguientes:

[Generar código con df_net](#) [Ver gráficos recomendados](#) [New interactive sheet](#)

About this Dataset

show_id : ID

type : tipo

title : título

director : director

cast : Elenco (reparto)

country : país

date_added : fecha_agregada

release_year : año_estreno

rating : clasificación

duration : duración

listed_in : listado_en

description : descripción

df_net.info() # Información de la data

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6234 entries, 0 to 6233
Data columns (total 12 columns):
#   Column          Non-Null Count  Dtype
---  -
0   show_id         6234 non-null   int64
1   type            6234 non-null   object
2   title           6234 non-null   object
3   director        4265 non-null   object
4   cast            5664 non-null   object
5   country         5758 non-null   object
6   date_added      6223 non-null   object
7   release_year    6234 non-null   int64
8   rating          6224 non-null   object
9   duration        6234 non-null   object
10  listed_in       6234 non-null   object
11  description     6234 non-null   object
```

```
dtypes: int64(2), object(10)
memory usage: 584.6+ KB
```

```
df_net.shape # Filas x Columnas (Dimensiones)
```

```
(6234, 12)
```

```
df_net.columns # Nombres de columnas
```

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

✓ 1. VISUALIZANDO EL DATASET

```
# A) Cantidad de:
```

```
df_net.duplicated().sum()      # VALORES DUPLICADOS    --> Mostrar
```

```
0
```

```
# B) Cantidad de:
```

```
df_net.nunique()              # VALORES ÚNICOS
```

```
df_net.isnull().sum()         # VALORES NULOS    --> Mostrar
```

```
0
```

	0
show_id	0
type	0
title	0
director	1969
cast	570
country	476
date_added	11
release_year	0
rating	10
duration	0
listed_in	0
description	0

```
dtype: int64
```

```
# C) Visualizar valores faltantes o vacíos (NaN):
```

```
A3 = df_net[df_net['director'].isnull()]
```

```
A3.head(4)
```



	show_id	type	title	director	cast	country	date_added	release_year
1	80117401	Movie	Jandino: Whatever it Takes	NaN	Jandino Asporaat	United Kingdom	September 9, 2016	20
2	70234439	TV Show	Transformers Prime	NaN	Peter Cullen, Sumalee Montano, Frank Welker, J...	United States	September 8, 2018	20

```
# Suma de los valores faltantes (NaN) en cada columna del DataFrame "df"
```

```
missing_value = df_net.isnull().sum()
```

```
missing_value
```

```
#NOTA:
```

```
# df.isnull() ---> Crea un DataFrame booleano donde: (True) indica un valor faltante
# (False) indica un valor presente.
```

```
# sum() ---> Suma los valores True en cada columna, dando el conteo de valores falt
```



	0
show_id	0
type	0
title	0
director	1969
cast	570
country	476
date_added	11
release_year	0
rating	10
duration	0
listed_in	0
description	0

```
dtype: int64
```

```
# D) Observar los missing
```

```
#def missing(df_net):
```

```
# Esta línea de  
# que toma un DataFr
```

```
# 1. Calculate missing value and their percentage for each column
```

```
#####
```

```
missing_value = df_net.isnull().sum()
```

```
missing_value = missing_value.reset_index()
```

```
#NOTA:
```

```
#####
```

```
# reset_index() : Convierte el índice del resultado anterior (que eran los nombres de las  
missing_value
```



	index	0	
0	show_id	0	
1	type	0	
2	title	0	
3	director	1969	
4	cast	570	
5	country	476	
6	date_added	11	
7	release_year	0	
8	rating	10	
9	duration	0	
10	listed_in	0	
11	description	0	

Pasos
siguientes:

[Generar código con missing_value](#)



[Ver gráficos recomendados](#)

[New interactive sheet](#)

```
# Eliminamos (DELETE) variable "missing_value"  
del missing_value
```

```
missing_value = df_net.isnull().sum()
```

```
missing_value = missing_value.reset_index().rename(  
columns={'index':'Column', 0:'Missing_value_count'})
```

```
#NOTA:
```

```
#####
```

```
# rename(columns={'index':'Column', 0:'Missing_value_count'}) : Cambia el nombre: column  
# column
```

```
missing_value
```



	Column	Missing_value_count	
0	show_id	0	
1	type	0	
2	title	0	
3	director	1969	
4	cast	570	
5	country	476	
6	date_added	11	
7	release_year	0	
8	rating	10	
9	duration	0	
10	listed_in	0	
11	description	0	

Pasos
siguientes:

[Generar código con missing_value](#)



[Ver gráficos recomendados](#)

[New interactive sheet](#)

```
# 2. Calculate missing value percentage for each column
##### # Calcula el por
missing_percent = df_net.isnull().sum() * 100 / df_net.shape[0]
```

#NOTA:

#####

```
# df.isnull().sum() ---> Obtiene el conteo de valores faltantes como
# df.isnull().sum() * 100 / df.shape[0] ---> Multiplica el conteo por 100 y lo divide por
# (df.shape[0]) = obteniendo así el porcentaje.
missing_percent = missing_percent.reset_index().rename(
    columns={'index':'Column', 0:'Missing_Percentage (%)'}).round(2)
```

#NOTA:

#####

```
# reset_index() ---> Convierte el índice
# rename(columns={'index':'Column', 0:'Missing_Percentage (%)'}) ---> Cambia los nombres
# round(2) ---> Redondea el percent
missing_percent
```



	Column	Missing_Percentage (%)	
0	show_id	0.00	
1	type	0.00	
2	title	0.00	
3	director	31.58	
4	cast	9.14	
5	country	7.64	
6	date_added	0.18	
7	release_year	0.00	
8	rating	0.16	
9	duration	0.00	
10	listed_in	0.00	
11	description	0.00	

Pasos
siguientes:

[Generar código con missing_percent](#)



[Ver gráficos recomendados](#)

[New interactive sheet](#)

```
# 3. MERGE : Combina los dos DataFrames calculados anteriormente
#           ("missing_value" y "missing_percent") en un solo DataFrame llamado
#####
missing_Final = missing_value.merge(missing_percent, how = 'inner', left_on = 'Column', r

#NOTA:
#####
# .merge( ) ---> Realiza un "inner join" basado en la columna "Column"
#               (que contiene los nombres de las columnas en ambos DataFrames).
missing_Final
```



	Column	Missing_value_count	Missing_Percentage (%)	
0	show_id	0	0.00	
1	type	0	0.00	
2	title	0	0.00	
3	director	1969	31.58	
4	cast	570	9.14	
5	country	476	7.64	
6	date_added	11	0.18	
7	release_year	0	0.00	
8	rating	10	0.16	
9	duration	0	0.00	
10	listed_in	0	0.00	
11	description	0	0.00	

Pasos
siguientes:

[Generar código con missing_Final](#)



[Ver gráficos recomendados](#)

[New interactive sheet](#)

4. Sort the data frame

#####

```
missing_Final = missing_Final.sort_values(by = 'Missing_Percentage (%)', ascending = False)
```

#NOTA:

#####

```
# .sort_values    ---> Ordena el DataFrame "Final" de manera descendente
#                según el porcentaje de valores faltantes ('Missing_Percentage (%)')
```

```
missing_Final
```

#NOTA:

#####

```
# return          ---> La función devuelve el DataFrame "missing_Final"
#                (que contiene el conteo y el porcentaje de valores faltantes) para cada columna
```




	Column	Missing_value_count	Missing_Percentage (%)	
3	director	1969	31.58	
4	cast	570	9.14	
5	country	476	7.64	
6	date_added	11	0.18	
8	rating	10	0.16	
0	show_id	0	0.00	
1	type	0	0.00	
2	title	0	0.00	
7	release_year	0	0.00	
9	duration	0	0.00	
10	listed_in	0	0.00	
11	description	0	0.00	

Pasos
siguientes:

[Generar código con missing_Final](#)



[Ver gráficos recomendados](#)

[New interactive sheet](#)

✓ 2. ANALIZAR BASE DE DATOS

```
# Realizar una copia para analizar
df = df_net.copy()
```

```
df.columns
```



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

Manejar

**** -Datos faltantes (BLANCO)****

**** -Datos incorrectos(TIENE DATO PERO ESTÁ COMPLETAMENTE MAL)****

**** -Datos inválidos (TIENE DATO PERO INCUMPLE LO PEDIDO)****

```
# Películas --> 1. Datos faltantes
#####
```

```
df.isnull().sum()
```



	0
show_id	0
type	0
title	0
director	1969
cast	570
country	476
date_added	11
release_year	0
rating	10
duration	0
listed_in	0
description	0

dtype: int64

```
# A) Dropping rows with NaN (información faltante) - (Blanco)
#####
df = df.dropna(subset=["director", "cast", "country","date_added","rating"])
# Check to see if all NaN values are resolved
df.isnull().sum()
```



	0
show_id	0
type	0
title	0
director	0
cast	0
country	0
date_added	0
release_year	0
rating	0
duration	0
listed_in	0
description	0

dtype: int64

```
df.shape # Ya se quitó los missing
```

```
(3774, 12)
```

```
A1 = df[df['director'].isnull()] # Comprobando
```

```
A1
```

```
show_id type title director cast country date_added release_year rating dur
```

```
# B) Replacing missing values with statistic mode (información faltante)
```

```
#####
```

```
# df['country'] = df['country'].fillna(df['country'].mode()[0])
```

```
# df['cast'].replace(np.nan, 'No Data',inplace = True)
```

```
# df['director'].replace(np.nan, 'No Data',inplace = True)
```

```
# df.dropna(inplace=True)
```

```
# C) Delete duplicateds
```

```
#####
```

```
# df.drop_duplicates(inplace= True)
```

```
# Películas --> 2. Datos incorrectos
```

```
#####
```

```
# Filtro 1: QUE INICIE CON " # "
```

```
df = df[~df['title'].str.startswith('#')] # Elimina filas que empiezan con '
```

```
# Filtro 2: QUE CONTENGAN SOLO NÚMEROS Y DEL 1 AL 99999
```

```
df = df[~df['title'].str.match(r'^\d{1,5}$')] # Elimina filas que contienen '199
```

```
# Filtro 3: QUE NO INICIE CON UNA LETRA " ~(A-Z) "
```

```
df = df[df['title'].str.match('^[a-zA-Z]')] # Elimina filas que no empiezan co
```

```
df.reset_index().tail(5)
```



	index	show_id	type	title	director	cast	country	date_added
3688	6142	80063224	TV Show	The Great British Baking Show	Andy Devonshire	Mel Giedroyc, Sue Perkins, Mary Berry, Paul Ho...	United Kingdom	August 30, 2019
3689	6158	80164216	TV Show	Miraculous: Tales of Ladybug & Cat Noir	Thomas Astruc	Cristina Vee, Bryce Papenbrook, Keith Silverst...	France, South Korea, Japan	August 2, 2019
3690	6167	80115328	TV Show	Sacred Games	Vikramaditya Motwane, Anurag Kashyap	Saif Ali Khan, Nawazuddin Siddiqui, Radhika Ap...	India, United States	August 15, 2019
Ho-dong								

df.shape



(3693, 12)

```
# CASO ADICIONAL 1.
# Eliminar filas que cumplen alguna de las condiciones (1-99999) | símbolos: # $ % /
# df = df[~(df['title'].str.match('[1-9][0-9]{0,4}$') | df['title'].str.match('[#\$\%\//])

# CASO ADICIONAL 2.
# Converting data type to float
# lst = ["Coarse wool Price", "Coarse wool price % Change", "Copra Price", "Copra price %
# df[lst] = df[lst].astype("float")
# df.dtypes
```

Rating (Clasificación de programas)

PG-13: Los padres están fuertemente advertidos de que algunas escenas pueden no ser apropiadas para niños menores de 13 años.

TV-MA: Para adultos. Puede contener material sexual explícito, violencia intensa, lenguaje crudo y otros temas adultos.

PG: Orientación parental sugerida. Puede contener material que algunos padres podrían considerar inapropiado para niños pequeños.

TV-14: Para mayores de 14 años. Puede contener escenas de violencia, lenguaje fuerte, situaciones sexuales sugestivas o contenido controvertido.

TV-PG: Orientación parental sugerida para televisión. Similar a PG, pero específicamente para televisión.

TV-Y: Para niños de todas las edades.

TV-Y7: Para niños de 7 años en adelante.

R: Restringida. Los menores de 17 años requieren el acompañamiento de un padre o tutor adulto.

TV-G: General para televisión. Similar a G, pero específicamente para televisión.

G: General. Apto para todo público, incluyendo niños.

NC-17: No apto para menores de 17 años. Contenido sexual explícito.

TV-Y7-FV: Para niños de 7 años en adelante, con violencia fantasía.

74 min

84 min

66 min

NR

UR

```
[ ] netflix_data['rating'] = netflix_data['rating'].replace({
    'PG-13': 'Teens - Age above 12',
    'TV-MA': 'Adults',
    'PG': 'Kids - with parental guidance',
    'TV-14': 'Teens - Age above 14',
    'TV-PG': 'Kids - with parental guidance',
    'TV-Y': 'Kids',
    'TV-Y7': 'Kids - Age above 7',
    'R': 'Adults',
    'TV-G': 'Kids',
    'G': 'Kids',
    'NC-17': 'Adults',
    'NR': 'NR',
    'UR' : 'UR'
})
```

```
# Películas --> 3. Datos inválidos
#####
```

```
# Usando REPLACE
```

```
df['rating'] = df['rating'].replace({'TV-Y7-FV': 'TV-Y7'})
```

```
# Usando REPLACE
```

```
df['rating'] = df['rating'].replace({'PG-13': 'Teens - age above 12',
    'TV-MA': 'Adults',
    'PG': 'Kids - with parental guidance',
    'TV-14': 'Teens - Age above 14',
    'TV-PG': 'Kids - with parental guidance',
    'TV-Y': 'Kids',
    'TV-Y7': 'Kids - Age above 7',
```

```

'R': 'Adults',
'TV-G': 'Kids',
'G': 'Kids',
'NC-17': 'Adults',
'NR': 'NR',
'UR' : 'UR'})

df['rating'].unique()

⇒ array(['Kids - with parental guidance', 'Adults', 'NR',
        'Teens - Age above 14', 'Teens - age above 12',
        'Kids - Age above 7', 'Kids', 'UR'], dtype=object)

```

✓ 3. ANÁLISIS EXPLORATORIO DE DATOS (EDA)

```

#Importacion de Librerias para GRÁFICOS
import matplotlib.pyplot as plt # Graficas
import seaborn as sns          # Graficas mas sencillas
%matplotlib inline

```

BAR-PLOT

```

# Diagrama de Barras (BAR-PLOT)  -----> Variable "TYPE"
#####
plt.figure(figsize=(8,4), tight_layout=True)      # plt.figure() ---> Crea una nueva
                                                    # figsize=(8,4) ---> Establece el t
                                                    # tight_layout=True ---> Ajusta aut
                                                    # para que n
sns.set_style("darkgrid")                        # sns.set_style() ---> Establece el estilo
                                                    # "darkgrid" ---> Aplica un estilo con
ax = sns.countplot(x="type", data=df, palette="mako") # sns.countplot() ---->
                                                    # x="type" ---->
                                                    # (
                                                    # data=df ---->
                                                    # palette="mako" ---->
ax.set(title='BARPLOT - type', xlabel='TIPO', ylabel='CANTIDAD') # ax.set() --
                                                    # title='BARPLOT' ---> Esta
                                                    # xlabel='TIPO' ---> Esta
                                                    # ylabel='CANTIDAD' ---> Esta
plt.show()          # plt.show() ---> Muestra el gráfico generado.

```



<ipython-input-29-fd72ffe240e1>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.countplot(x="type", data=df, palette="mako") # sns.countplot()
```

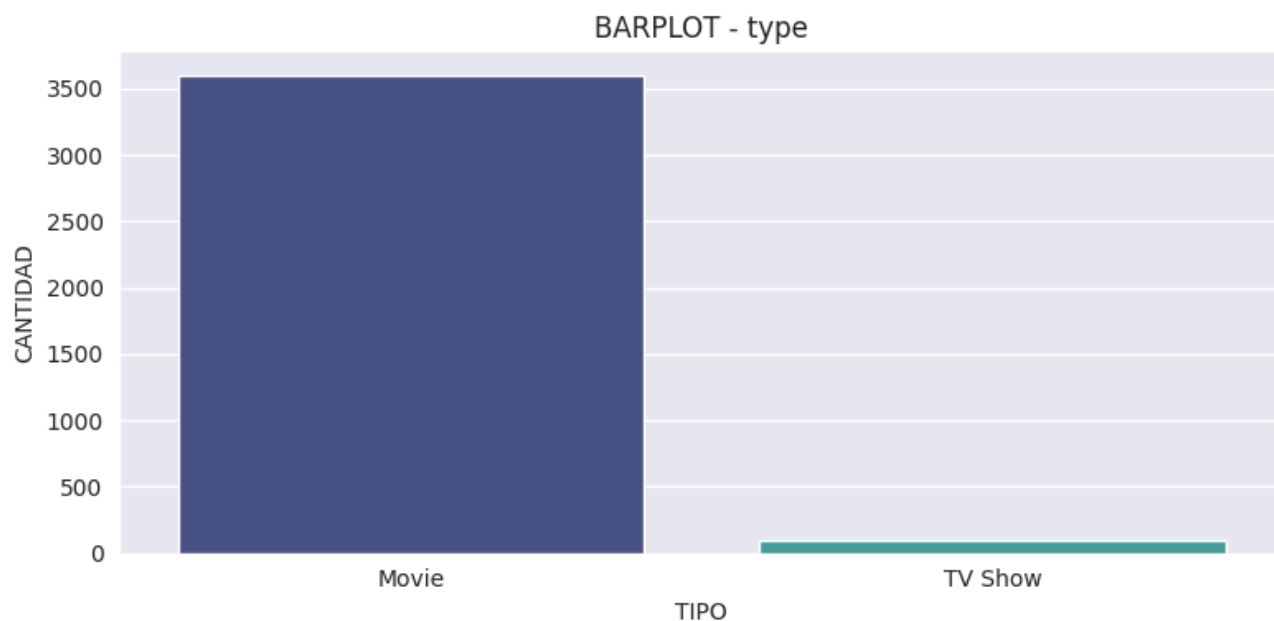


Diagrama de Barras (BAR-PLOT) -----> Variable "RATING"

#####

```
plt.figure(figsize=(20,12))
```

```
sns.set(style="whitegrid")
```

```
ax = sns.countplot(x="rating", data=df, palette="colorblind", order=df['rating'].value_cc
```

```
ax.set(title='BARPLOT - rating', xlabel='CLASIFICACIÓN', ylabel='CANTIDAD')
```

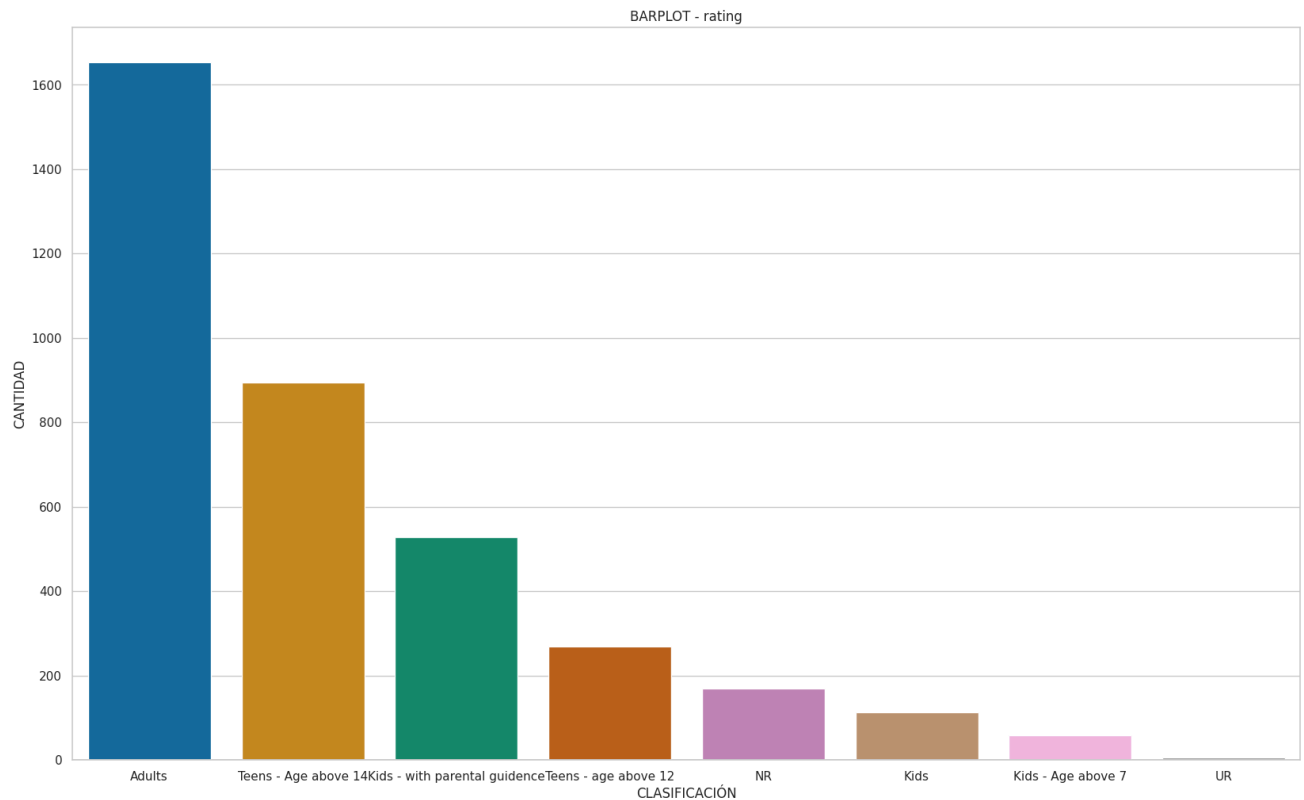
```
plt.show()
```



<ipython-input-50-03661c4485d5>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.


```
ax = sns.countplot(x="rating", data=df, palette="colorblind", order=df['rating'].va
```



```
plt.figure(figsize=(20,12))  
sns.set(style="whitegrid")  
ax = sns.countplot(x="rating", data=df, palette="colorblind", order=df['rating'].value_co
```

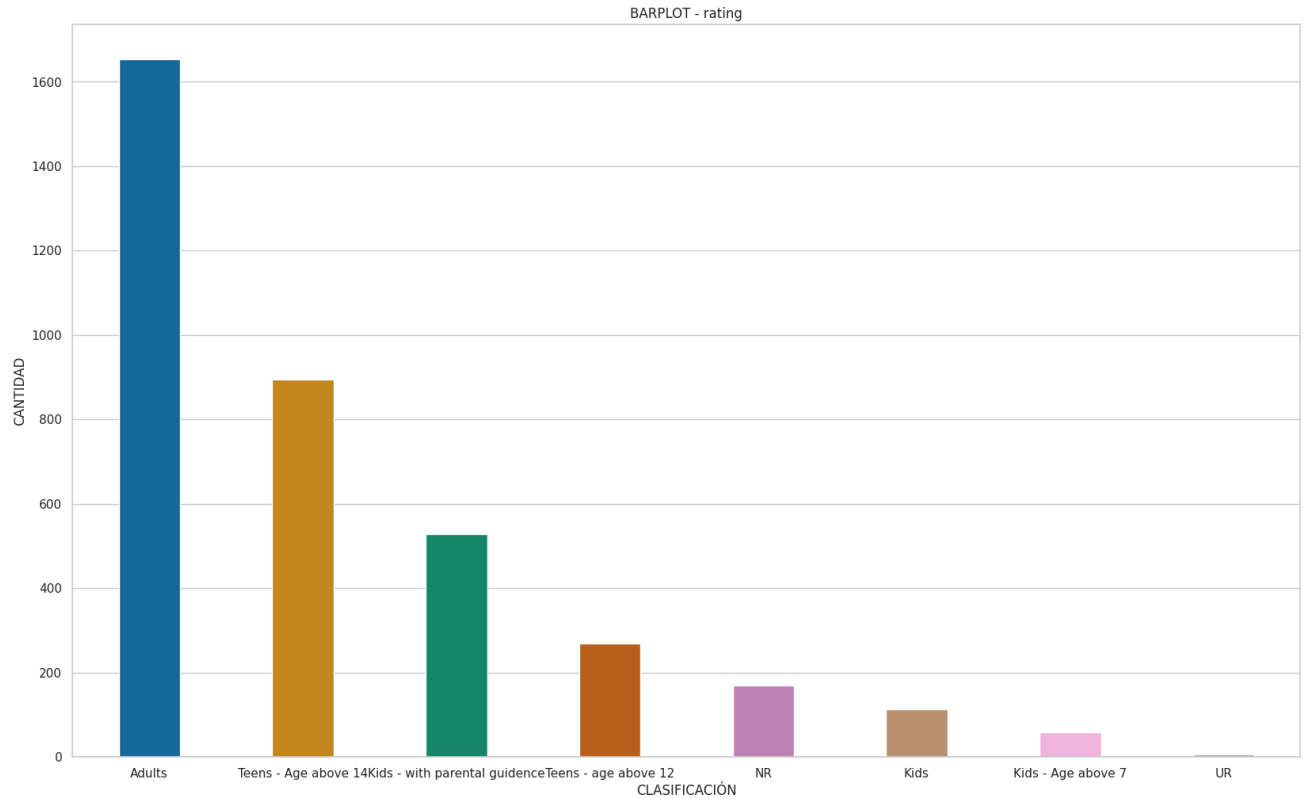


```
ax.set(title='BARPLOT - rating', xlabel='CLASIFICACIÓN', ylabel='CANTIDAD')  
#plt.xticks(rotation=45, fontsize=8) # ---> rotación de las palabras y tamaño de la letra  
plt.show()
```

 <ipython-input-51-2c4f1c15953b>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
ax = sns.countplot(x="rating", data=df, palette="colorblind", order=df['rating'].va
```



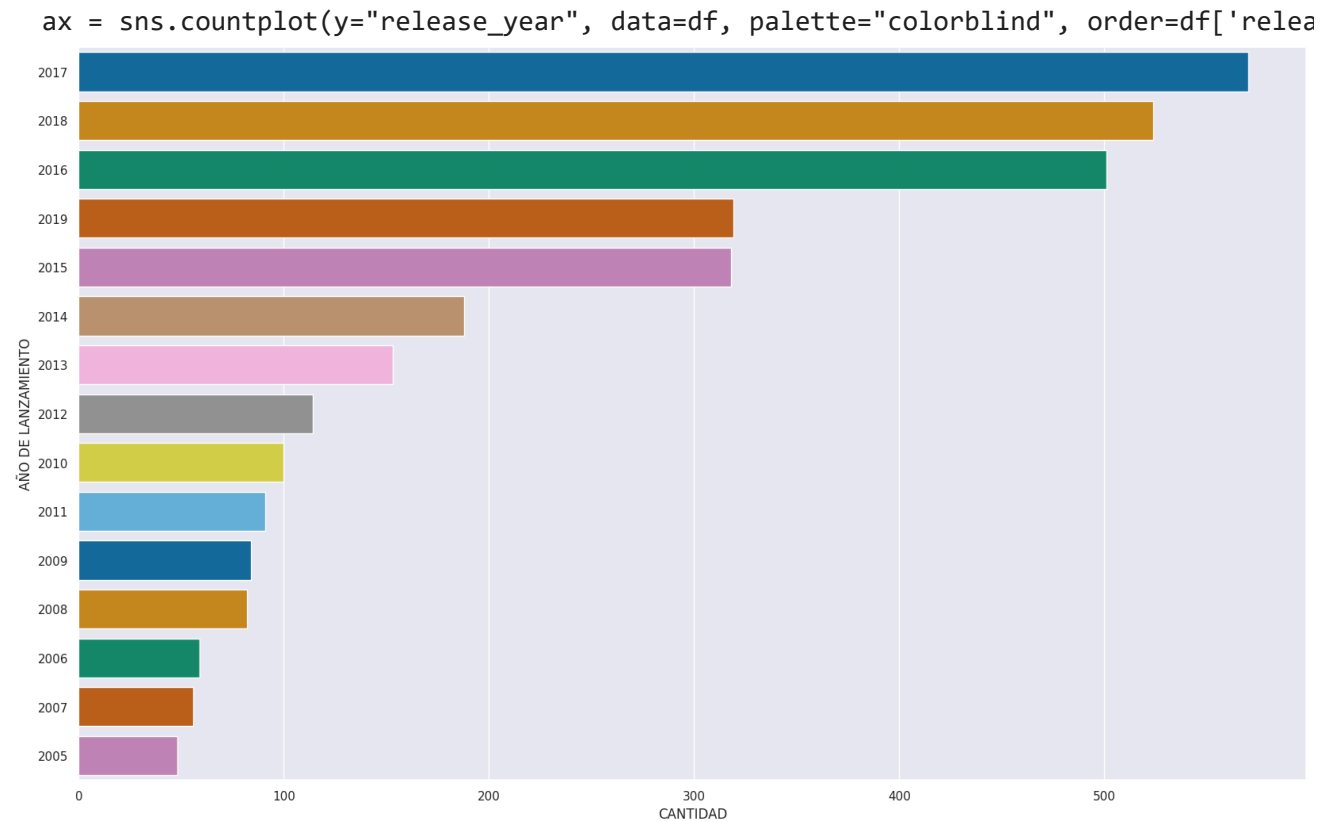
BAR-PLOT INVERTIDO

```
# Diagrama de Barras invertido (INVERTED BAR-PLOT) -----> Variable "release_year"
#####
plt.figure(figsize=(20,12))
sns.set(style="darkgrid")
ax = sns.countplot(y="release_year", data=df, palette="colorblind", order=df['release_yea
ax.set(xlabel='CANTIDAD', ylabel='AÑO DE LANZAMIENTO') # y = variable de gráfica - x = Ca
plt.show()
```



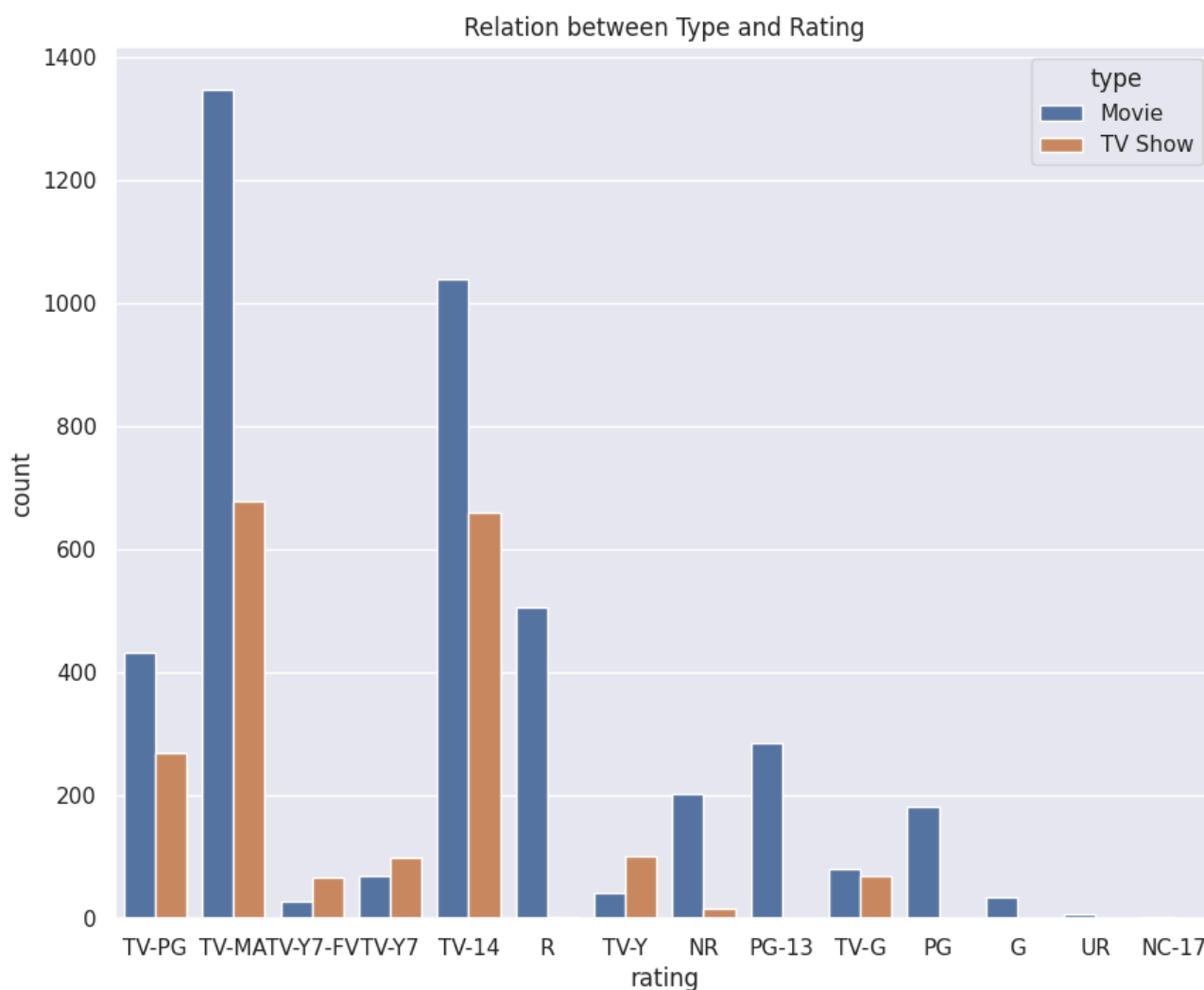
<ipython-input-52-db8fc6d20063>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.



BAR-PLOT (RELACIÓN DE VARIABLES)

```
# BAR-PLOT) -----> Variables | type | rating |
#####
plt.figure(figsize=(10,8))
sns.countplot(x='rating',hue='type',data=df_net) # Eje X ---> variable "rating" | # Leye
plt.title('Relation between Type and Rating')    # Título para la gráfica
plt.show()
```



PIE CHART

```
# PIE CHART -----> Variables | type |
#####
labels = ['Movie', 'TV show'] # etiquetas
size = df['type'].value_counts() # tamaño
colors = plt.cm.Wistia(np.linspace(0, 1, 2)) # color
```

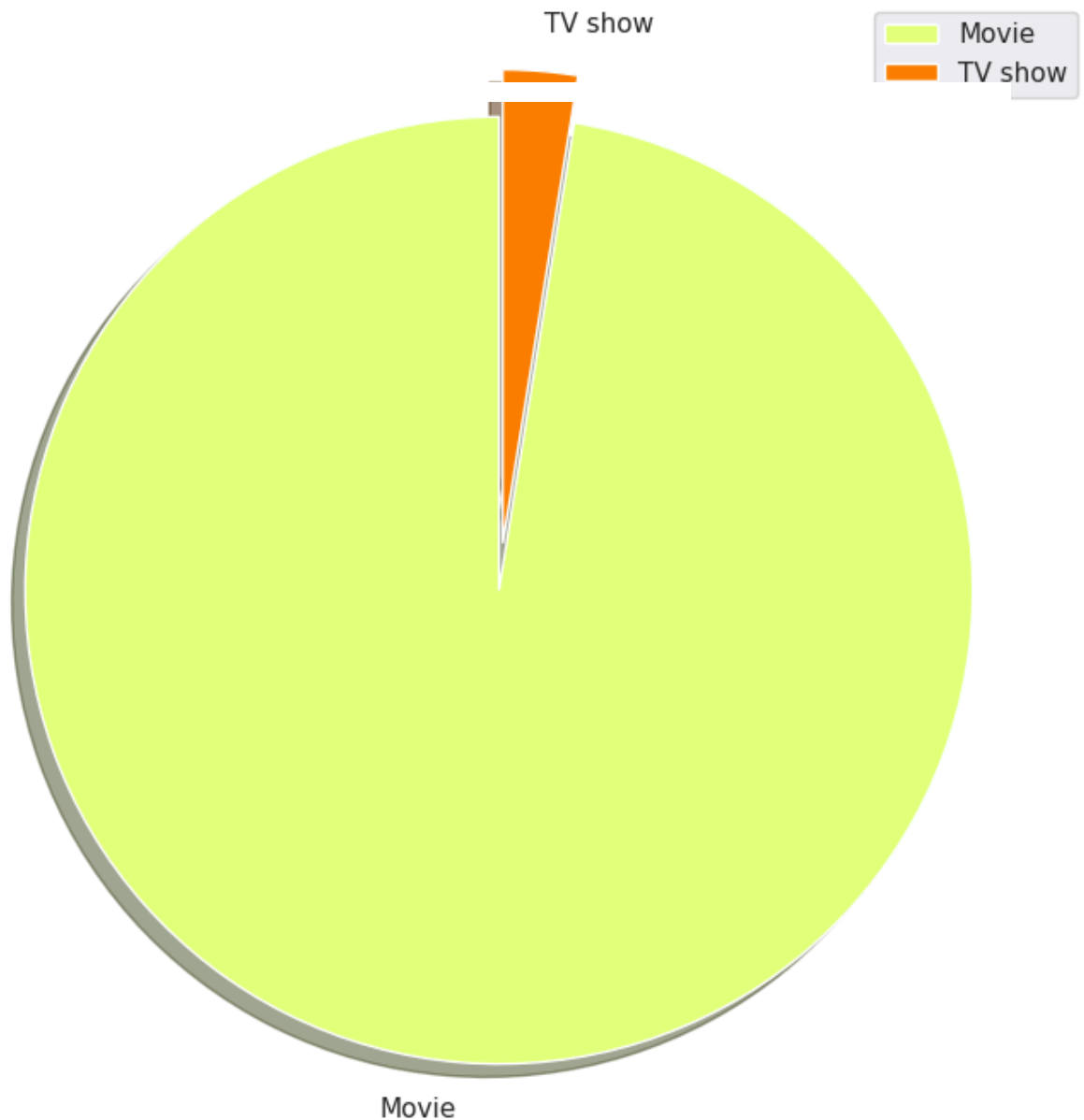
```

explode = [0, 0.1]
plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(size, labels=labels, colors = colors, explode = explode, shadow = True, startangle
plt.title('Distribution of Type', fontsize = 25)           # Título
plt.legend()                                              # Leyenda
plt.show()

```



Distribution of Type



```

# PIE CHART (with %) -----> Variables | type |
##### Pie-char
df['rating'].value_counts().plot.pie(autopct='%1.1f%%', shadow=True, figsize=(10,8))
plt.show()

```

```

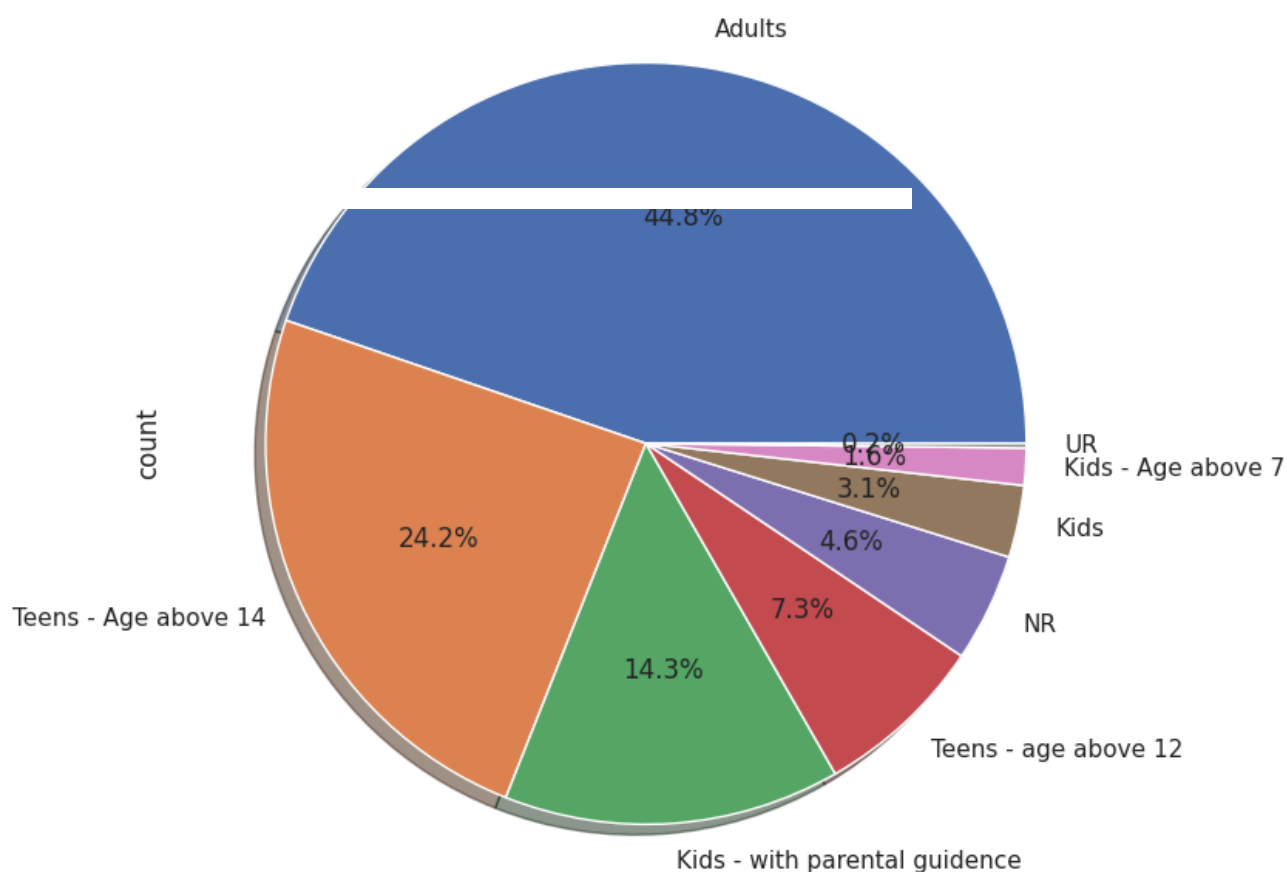
# .value_counts() ---> Cuenta la frecuencia de cada valor único en la columna 'rating',
# creando una nueva serie donde el índice son los valores únicos y

```

```
# Es decir, te dice cuántas veces aparece cada calificación.

# .plot ---> Este método se utiliza para crear diferentes tipos de gráficos.

# En este caso, al utilizar .pie(), estamos especificando que queremos
# autopct='%1.1f%%' ---> Este argumento formatea los porcentajes que se muestran dentro de
# %1.1f ---> Indica que se mostrará un número con un decimal.
# %% ---> Escapa el símbolo de porcentaje para que sea mostrado literalmente
# shadow=True ---> Agrega una sombra a cada porción del gráfico de pie para darle un
# figsize=(10,8) ---> Establece el tamaño de la figura en 10 pulgadas de ancho y 8 pulgadas de alto
```



VISUALIZACIÓN --> NUBE DE PALABRAS

```
# WordCloud
#####
from wordcloud import WordCloud # Genera visualizaciones de texto en forma de nube de p
```

```

# NUBE DE IDEAS -----> Variables | type |
#####
plt.subplots(figsize=(25,15))          # Crea una nueva figura para el gráfico
                                         # con un tamaño de 25 pulgadas de ancho y 15 pu
wordcloud = WordCloud(
    background_color='white',
    width=1920,
    height=1080
).generate(" ".join(df.director))
# WordCloud()          ---> Crea un objeto WordCloud con las sigui
# background_color='white' ---> Define el color de fondo de la nube de
# width=1920            ---> Establece el ancho de la nube de palab
# height=1080          ---> Establece el alto de la nube de palab
# .generate(" ".join(df.country)) ---> Genera la nube de palabras a partir de
#                               la unión de todos los países en la col
# La función join une los elementos de la lista con un espacio en blanco como
plt.imshow(wordcloud)      # Muestra la nube de palabras creada anteriormente.
plt.axis('off')           # Oculta los ejes del gráfico (x y y) ya que no son necesario
plt.savefig('director.png') # Guarda la nube de palabras como una imagen PNG llamada "ti
plt.show()                # Muestra la nube de palabras en pantalla (opcional, ya que s

```

