Import necessary libraries

In [1]:

```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objs as go
import plotly.figure_factory as ff
import calendar
from collections import Counter
from plotly.subplots import make_subplots
```

Read data from a CSV file into a pandas DataFrame and handle defects

In [2]:

```
df = pd.read_csv('netflix_titles.csv')
df.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
```

Check and display null rates for each column in the DataFrame

In [3]:

```
for i in df.columns:
    null_rate = df[i].isna().sum() / len(df) * 100
    if null_rate > 0 :
        print(f"{i} null rate: {round(null_rate, 2)}%")
```

```
director null rate: 29.91%
cast null rate: 9.37%
country null rate: 9.44%
date_added null rate: 0.11%
rating null rate: 0.05%
duration null rate: 0.03%
```

Dealing with missing data

- Replace NaN values in specific columns with 'No Data'
- Drop rows with NaN values in any column
- Drop duplicate rows from the DataFrame

In [4]:

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

df.dropna(inplace=True)

df.drop_duplicates(inplace= True)
```

In [5]:

```
df.head()
```

Out[5]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	No Data	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	No Data	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	No Data	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	No Data	No Data	No Data	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	No Data	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

Dealing with dates, seasons and durations

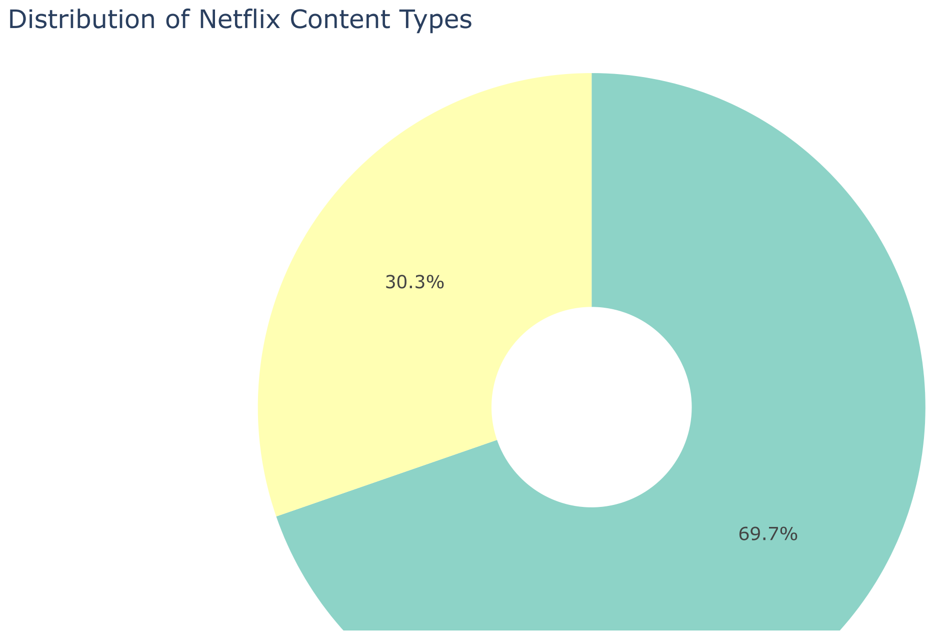
In [6]:

```
df["date_added"] = pd.to_datetime(df['date_added'], format='%B %d, %Y', errors='coerce')
df['year_added'] = df['date_added'].dt.year.fillna(0).astype(int)
df['month_added'] = df['date_added'].dt.month.fillna(0).astype(int)

df['season_count'] = df.apply(lambda x: x['duration'].split(" ")[0] if "Season" in x['duration'] else "",
df['duration'] = df.apply(lambda x: x['duration'].split(" ")[0] if "Season" not in x['duration'] else "",
df.head()
```

Out[6]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	No Data	United States	2021-09-25	2020	PG-13	90	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	No Data	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	2021-09-24	2021	TV-MA		International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	No Data	2021-09-24	2021	TV-MA		Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	No Data	No Data	No Data	2021-09-24	2021	TV-MA		Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	No Data	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	2021-09-24	2021	TV-MA		International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

In [7]:
```
grouped_data = df.groupby('type').size().reset_index(name='count')

fig = px.pie(grouped_data, names='type', values='count', title='Distribution of Netflix Content Types',
             hole=0.3, color_discrete_sequence=px.colors.qualitative.Set3)


fig.update_layout(margin=dict(l=20, r=20, t=60, b=20), showlegend=True)

fig.show()
```



Distribution of Netflix content types

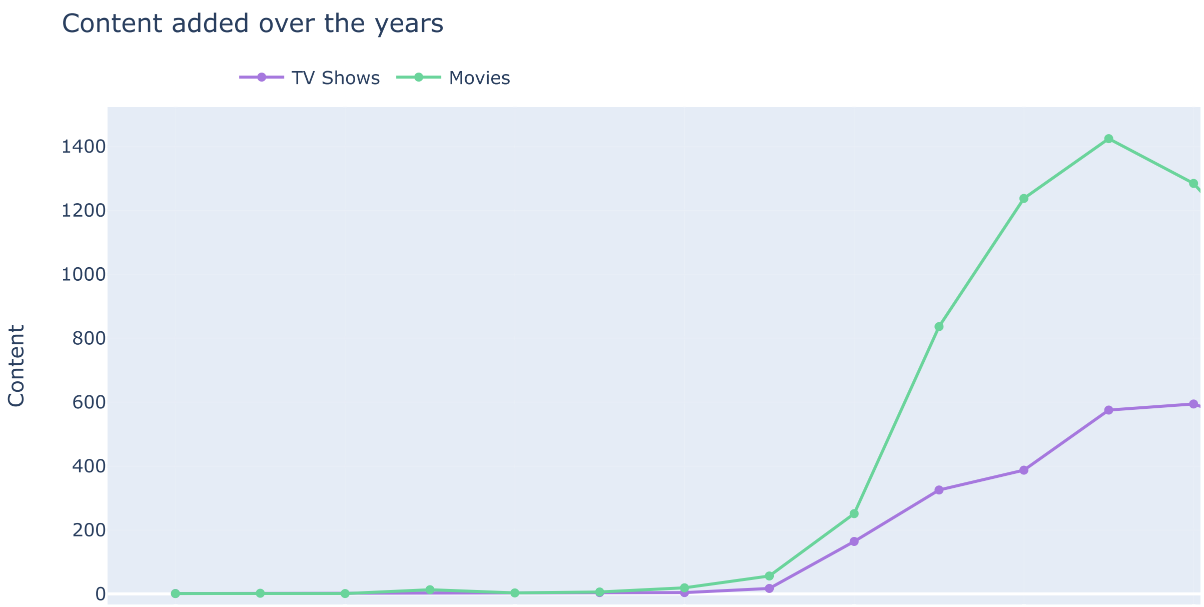
This code generates a pie chart using Plotly to illustrate the distribution of Netflix content types. It begins by grouping the DataFrame by 'type' and calculating the count for each type. The pie chart is then created with specific settings such as title, colors, and layout adjustments. Finally, the interactive plot is displayed.

In [8]:
```
col = "year_added"

d1 = df[(df[col] > 0) & (df["type"] == "TV Show")]
d2 = df[(df[col] > 0) & (df["type"] == "Movie")]

tv_shows = d1[col].value_counts().reset_index()
tv_shows['percent'] = tv_shows['count'].apply(lambda x : 100 * x / sum(tv_shows['count']))
tv_shows = tv_shows.sort_values(col)

movies = d2[col].value_counts().reset_index()
movies['percent'] = movies['count'].apply(lambda x : 100 * x / sum(movies['count']))
movies = movies.sort_values(col)
```

Trend of content added over the years for TV Shows and Movies.

In [9]:

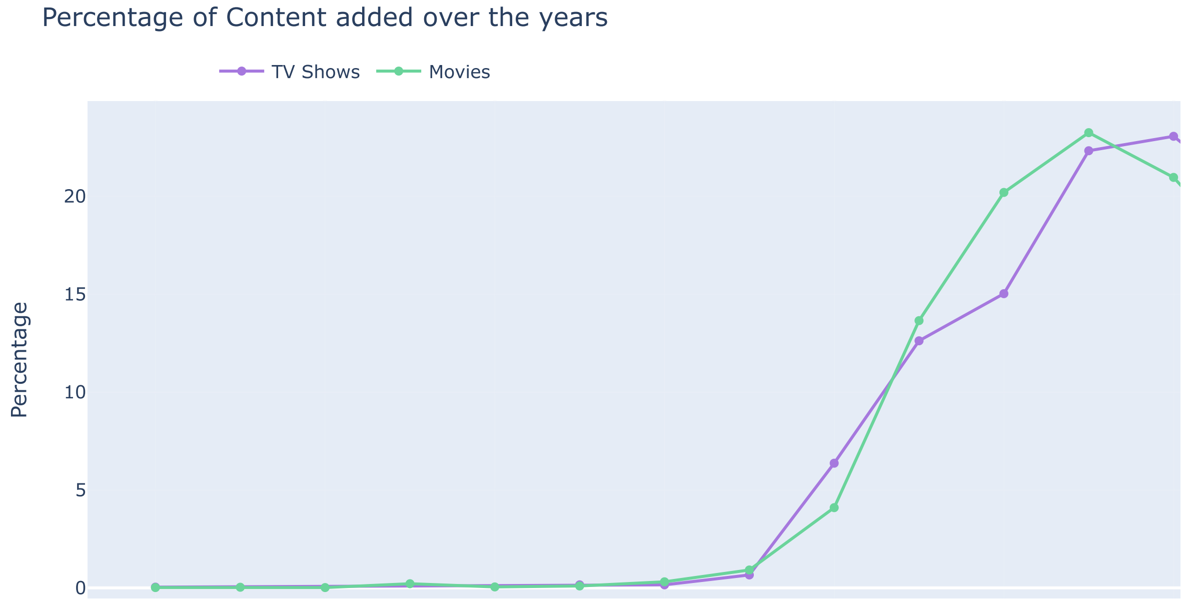
```
trace1 = go.Scatter(x=tv_shows[col], y=tv_shows["count"], name="TV Shows", marker=dict(color="#a678de"))
trace2 = go.Scatter(x=movies[col], y=movies["count"], name="Movies", marker=dict(color="#6ad49b"))

layout = go.Layout(title="Content added over the years",
                   xaxis=dict(title='Year'), yaxis=dict(title='Content'),
                   legend=dict(x=0.1, y=1.1, orientation="h"))

fig = go.Figure(data=[trace1, trace2], layout=layout)
fig.show()
```



Percentage of TV shows and movies added over the years.

In [10]:

```
trace3 = go.Scatter(x=tv_shows[col], y=tv_shows["percent"], name="TV Shows", line=dict(color="#a678de"))
trace4 = go.Scatter(x=movies[col], y=movies["percent"], name="Movies", line=dict(color="#6ad49b"))

layout = go.Layout(title="Percentage of Content added over the years",
                   xaxis=dict(title='Year'), yaxis=dict(title='Percentage'),
                   legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data=[trace3, trace4], layout=layout)
fig.show()
```



This code determines the year in which the maximum number of TV shows and movies were added, respectively.

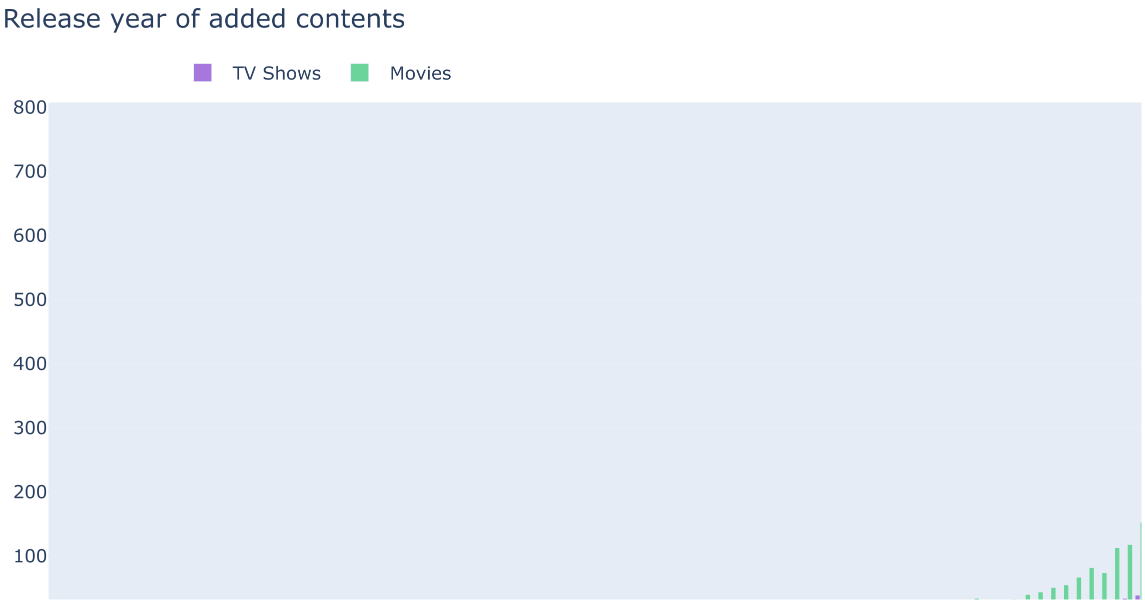
In [11]:

```
max_tv_shows_year = tv_shows.loc[tv_shows['count'].idxmax()][col]
max_movies_year = movies.loc[movies['count'].idxmax()][col]

print(f"The maximum number of TV Shows was added in {int(max_tv_shows_year)}.")
print(f"The maximum number of Movies was added in {int(max_movies_year)}.")
```

```
The maximum number of TV Shows was added in 2020.
The maximum number of Movies was added in 2019.
```

Distribution of release years for added TV shows and movies, with counts represented by different bars for each content type.

In [12]:
```
col = "release_year"

d1 = df[(df[col] > 0) & (df["type"] == "TV Show")]
d2 = df[(df[col] > 0) & (df["type"] == "Movie")]

tv_shows = d1[col].value_counts().reset_index()
tv_shows['percent'] = tv_shows['count'].apply(lambda x : 100 * x / sum(tv_shows['count']))
tv_shows = tv_shows.sort_values(col)

movies = d2[col].value_counts().reset_index()
movies['percent'] = movies['count'].apply(lambda x : 100 * x / sum(movies['count']))
movies = movies.sort_values(col)

trace1 = go.Bar(x=tv_shows[col], y=tv_shows["count"], name="TV Shows", marker=dict(color="#a678de"))
trace2 = go.Bar(x=movies[col], y=movies["count"], name="Movies", marker=dict(color="#6ad49b"))
data = [trace1, trace2]
layout = go.Layout(title="Release year of added contents", legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data, layout=layout)
fig.show()
```



Titles and release years of the first 15 entries for Movies.

```
In [13]: result = df[df['duration'] != ""].sort_values("release_year", ascending=True)[['title', 'release_year']][:
         result
```

Out[13]:

	title	release_year
7790	Prelude to War	1942
8205	The Battle of Midway	1942
8763	WWII: Report from the Aleutians	1943
8660	Undercover: How to Operate Behind Enemy Lines	1943
8739	Why We Fight: The Battle of Russia	1943
8640	Tunisian Victory	1944
8419	The Memphis Belle: A Story of a\nFlying Fortress	1944
8436	The Negro Soldier	1944
7219	Know Your Enemy - Japan	1945
7575	Nazi Concentration Camps	1945
7930	San Pietro	1945
7294	Let There Be Light	1946
8587	Thunderbolt	1947
2375	The Blazing Sun	1954
1699	White Christmas	1954

Titles and release years of the first 15 entries for TV shows

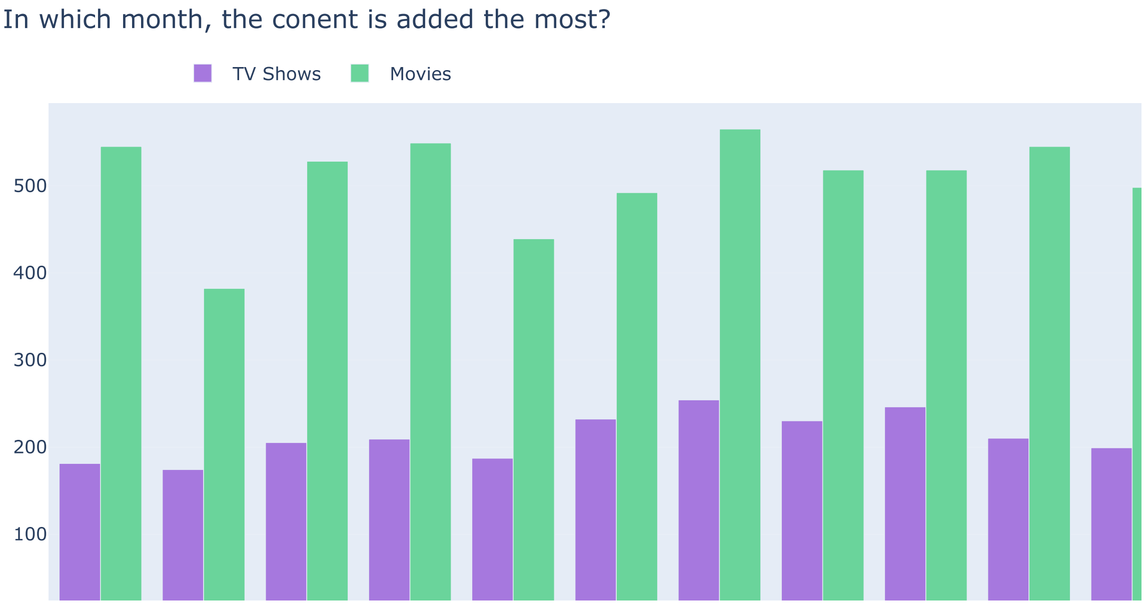
```
In [14]: result = df[df['season_count'] != ""].sort_values("release_year", ascending=True)[['title', 'release_year'
         result
```

Out[14]:

	title	release_year
4250	Pioneers: First Women Filmmakers*	1925
1331	Five Came Back: The Reference Films	1945
7743	Pioneers of African-American Cinema	1946
8541	The Twilight Zone (Original Series)	1963
8189	The Andy Griffith Show	1967
4550	Monty Python's Fliegender Zirkus	1972
4551	Monty Python's Flying Circus	1974
6549	Dad's Army	1977
6674	El Chavo	1979
7588	Ninja Hattori	1981
7878	Robotech	1985
2740	Saint Seiya	1986
7993	Shaka Zulu	1986
5299	High Risk	1988
6970	Highway to Heaven	1988

Displays the distribution of added contents per month

In [15]:
```
col = 'month_added'

d1 = df[(df[col] > 0) & (df["type"] == "TV Show")]
d2 = df[(df[col] > 0) & (df["type"] == "Movie")]

tv_shows = d1[col].value_counts().reset_index()
tv_shows['percent'] = tv_shows['count'].apply(lambda x : 100 * x / sum(tv_shows['count']))
tv_shows = tv_shows.sort_values(col)

movies = d2[col].value_counts().reset_index()
movies['percent'] = movies['count'].apply(lambda x : 100 * x / sum(movies['count']))
movies = movies.sort_values(col)

trace1 = go.Bar(x=tv_shows[col], y=tv_shows["count"], name="TV Shows", marker=dict(color="#a678de"))
trace2 = go.Bar(x=movies[col], y=movies["count"], name="Movies", marker=dict(color="#6ad49b"))

data = [trace1, trace2]
layout = go.Layout(title="In which month, the conent is added the most?", legend=dict(x=0.1, y=1.1, orient
fig = go.Figure(data, layout=layout)

fig.show()
```



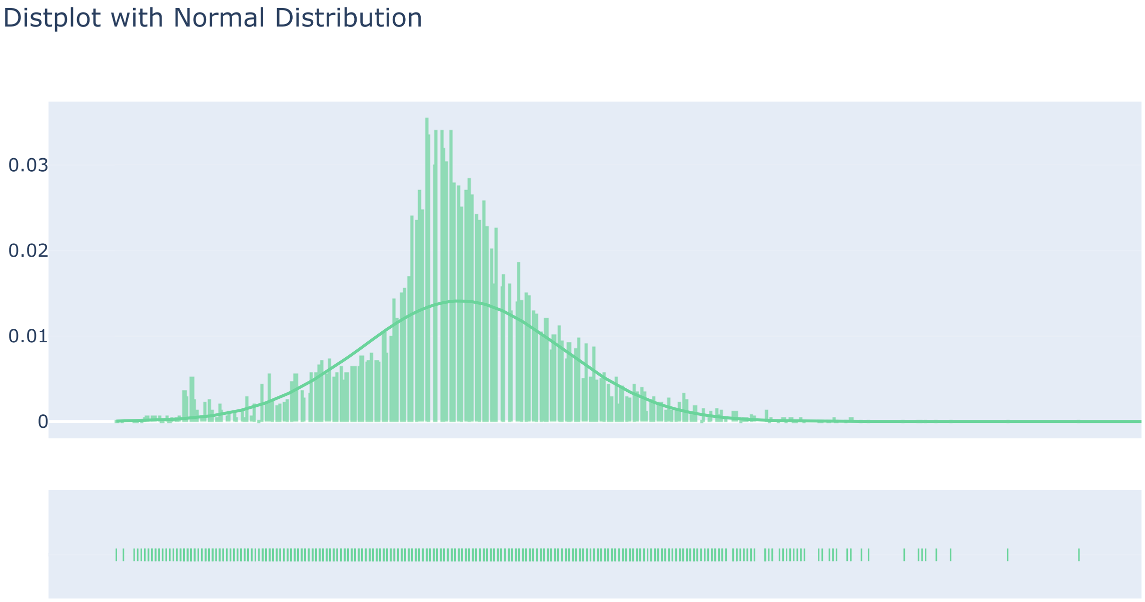
Print the month in which the most TV shows and movies were added.

In [16]:
```
max_tv_shows_year = tv_shows.loc[tv_shows['count'].idxmax()][col]
max_movies_year = movies.loc[movies['count'].idxmax()][col]

print(f"The most of the TV Shows was added in {calendar.month_name[int(max_tv_shows_year)]}.")
print(f"The most of the Movies was added in {calendar.month_name[int(max_movies_year)]}.")
```

```
The most of the TV Shows was added in July.
The most of the Movies was added in July.
```

Distribution of movie durations

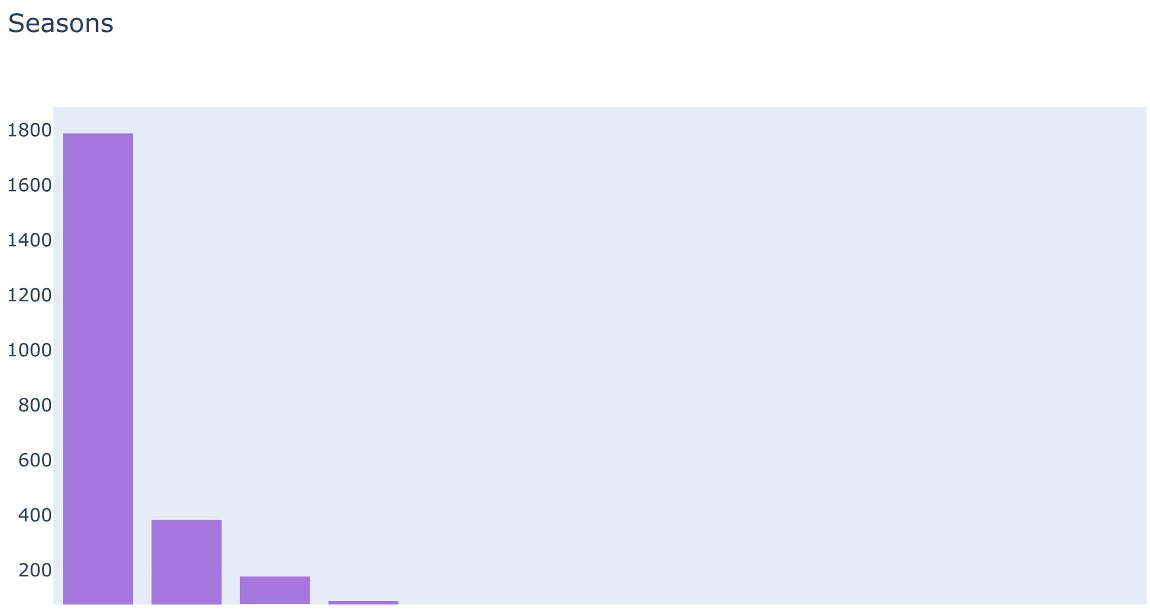
In [17]:
```
x1 = d2['duration'].fillna(0.0).astype(float)
fig = ff.create_distplot([x1], ['a'], bin_size=0.7, curve_type='normal', colors=["#6ad49b"])
fig.update_layout(title_text='Distplot with Normal Distribution')
fig.show()
```



Distribution of the number of seasons for TV shows.

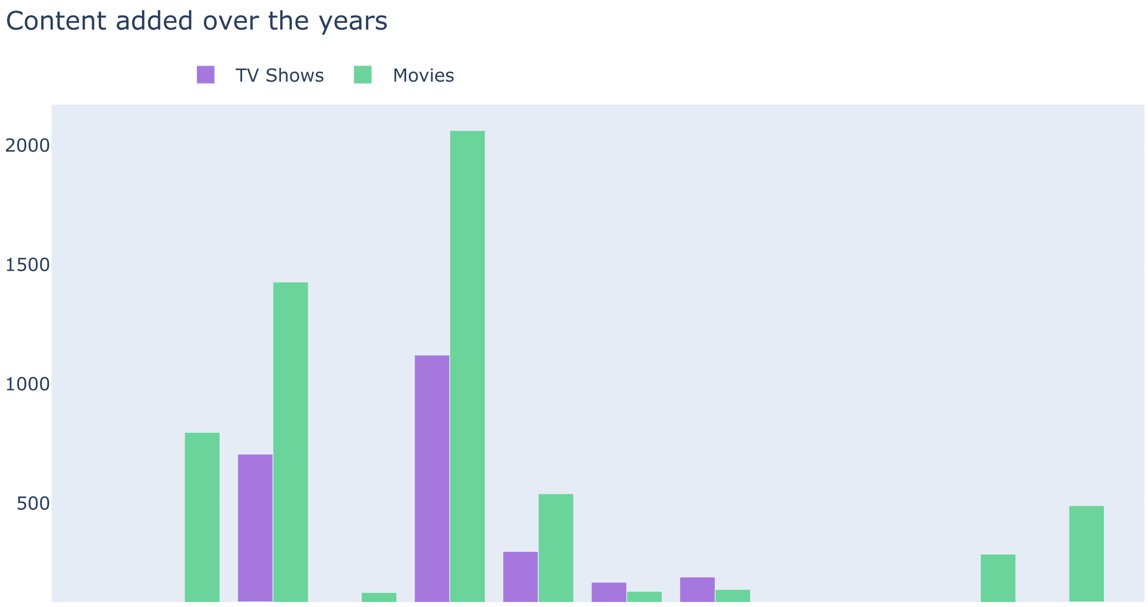
In [18]:

```
col = 'season_count'
tv_shows = d1[col].value_counts().reset_index()
tv_shows['percent'] = tv_shows['count'].apply(lambda x : 100*x/sum(tv_shows['count']))

trace1 = go.Bar(x=tv_shows[col], y=tv_shows["count"], name="TV Shows", marker=dict(color="#a678de"))
data = [trace1]
layout = go.Layout(title="Seasons", legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data, layout=layout)
fig.show()
```



Compare the distribution of ratings for TV shows and movies

In [19]:
```
col = "rating"

tv_shows = d1[col].value_counts().reset_index()
tv_shows['percent'] = tv_shows['count'].apply(lambda x : 100*x/sum(tv_shows['count']))
tv_shows = tv_shows.sort_values(col)

movies = d2[col].value_counts().reset_index()
movies['percent'] = movies['count'].apply(lambda x : 100*x/sum(movies['count']))
movies = movies.sort_values(col)

trace1 = go.Bar(x=tv_shows[col], y=tv_shows["count"], name="TV Shows", marker=dict(color="#a678de"))
trace2 = go.Bar(x=movies[col], y=movies["count"], name="Movies", marker=dict(color="#6ad49b"))
data = [trace1, trace2]
layout = go.Layout(title="Content added over the years", legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data, layout=layout)
fig.show()
```

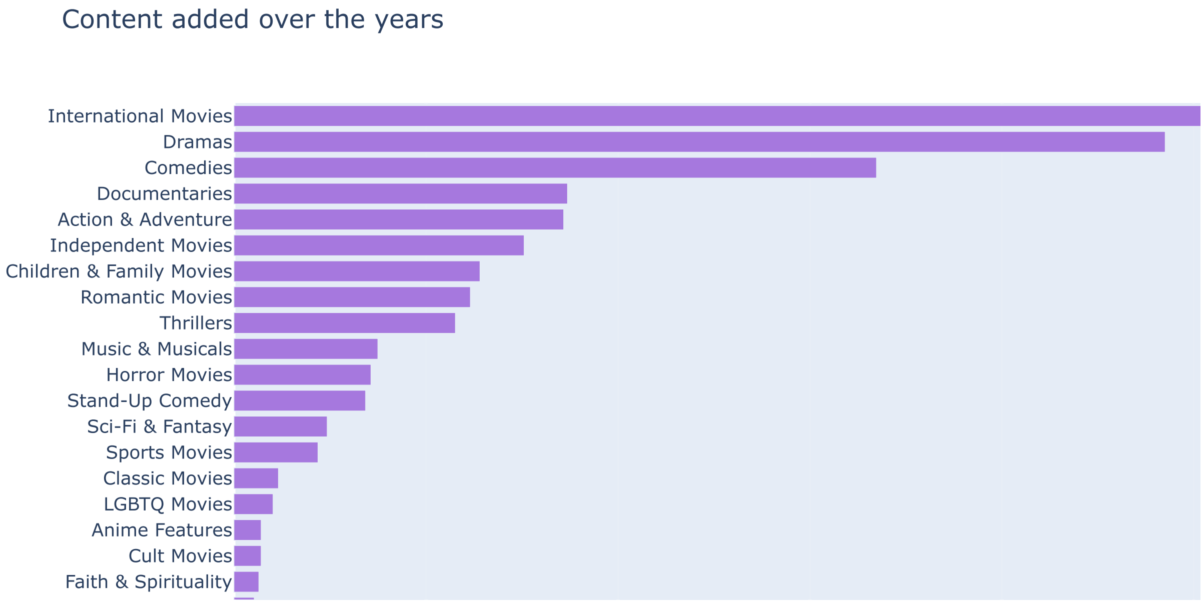


TV-MA: Mature Audience Only. Intended for adults and may be unsuitable for children under 17.

TV-14: This program contains some material that many parents would find unsuitable for children under 14 years of age.

Visualize the distribution of content categories (genres) for movies

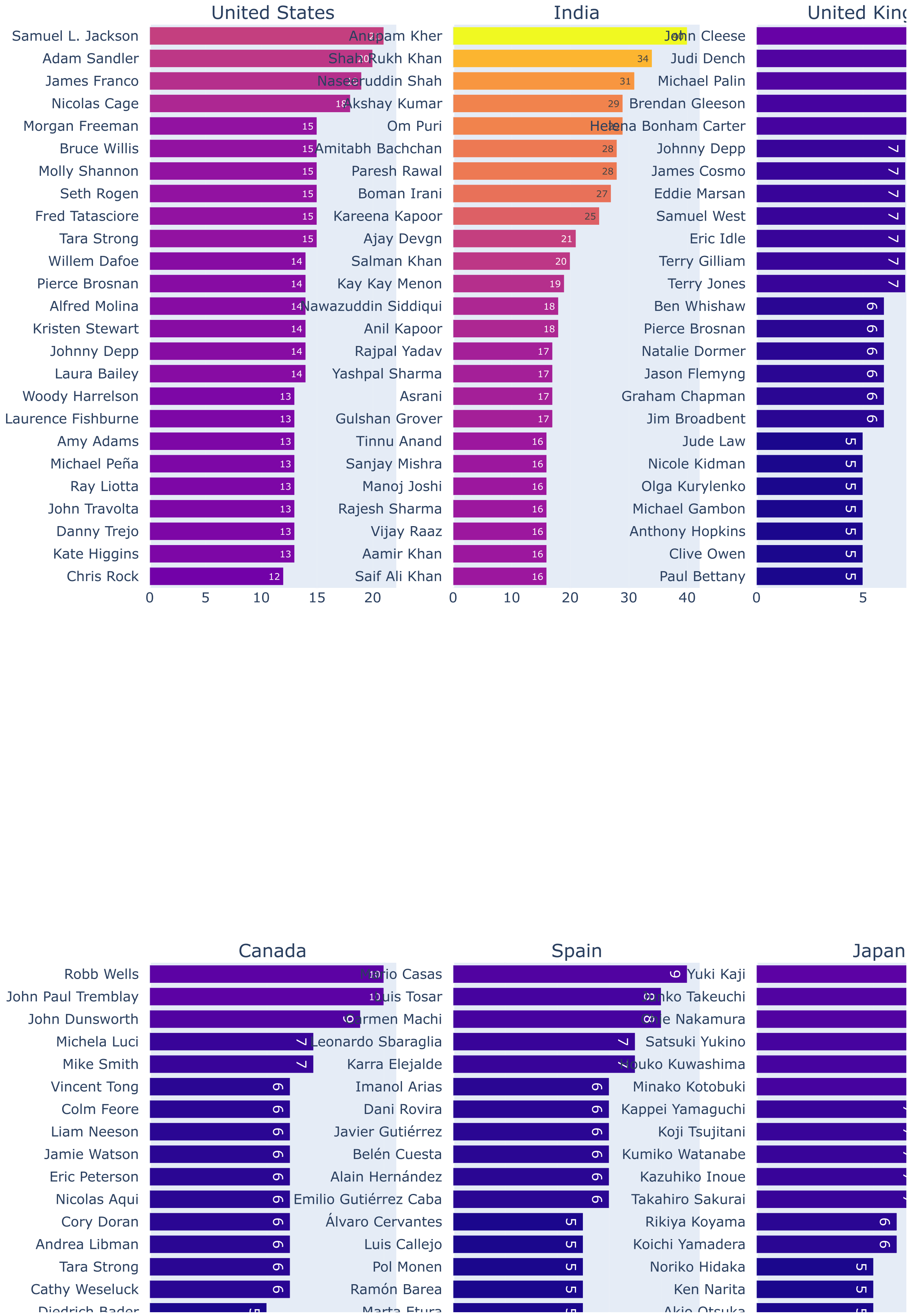
In [20]:
```
col = "listed_in"
counter_list = Counter(d2[col].str.split(", ").explode().tolist()).most_common(50)
labels = [_[0] for _ in counter_list][::-1]
values = [_[1] for _ in counter_list][::-1]
trace1 = go.Bar(y=labels, x=values, orientation="h", name="TV Shows", marker=dict(color="#a678de"))

data = [trace1]
layout = go.Layout(title="Content added over the years", legend=dict(x=0.1, y=1.1, orientation="h"))
fig = go.Figure(data, layout=layout)
fig.show()
```



Most common cast members in movies or TV shows from the United States, India, the United Kingdom, Canada, Spain, and Japan

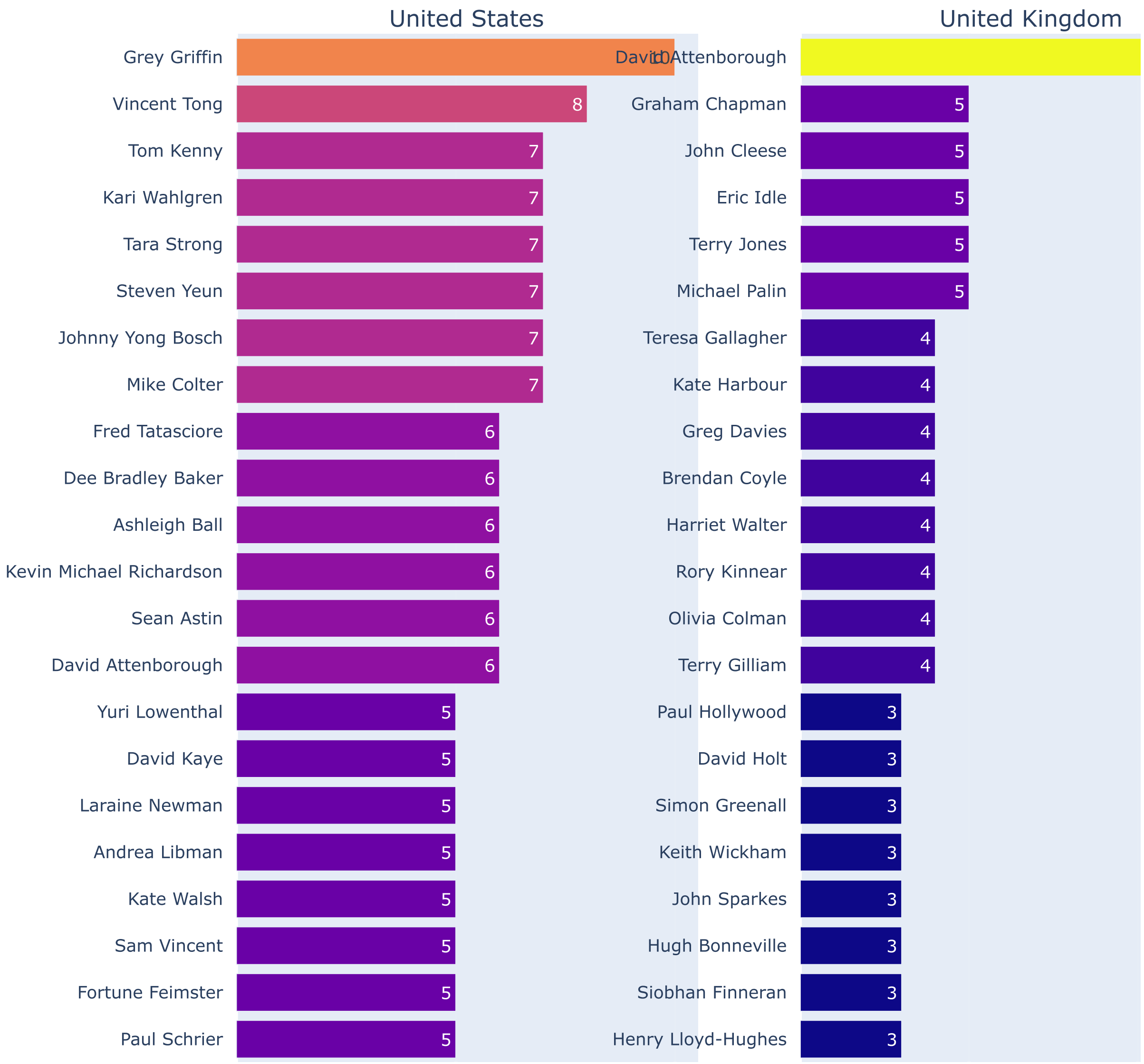
In [21]:

```
def country_trace(country, flag="movie"):
    filtered_df = df[(df["cast"] != "No Data") & (df['country'].fillna("").str.lower().apply(lambda x: 1 i

    if flag == "movie":
        filtered_df = filtered_df[filtered_df["duration"] != ""]
    else:
        filtered_df = filtered_df[filtered_df["season_count"] != ""]

    tags = Counter(filtered_df['cast'].str.split(", ").explode().to_list()).most_common(25)
    tags = [_ for _ in tags if _[0] != ""]

    labels, values = [_[0] + "  " for _ in tags], [_[1] for _ in tags]

    trace = px.bar(y=labels[::-1], x=values[::-1], orientation="h", labels={'x': 'Count', 'y': 'Cast'},
                   title=f'Most Common Cast Members in {country}', text=values[::-1], color=values[::-1])

    return trace

countries = ["United States", "India", "United Kingdom", "Canada", "Spain", "Japan"]

fig = make_subplots(rows=2, cols=3, subplot_titles=countries)
for i, country in enumerate(countries):
    traces = [country_trace(country)]
    for trace in traces:
        fig.add_trace(trace.data[0], row=i // 3 + 1, col=i % 3 + 1)

fig.update_layout(showlegend=False, height=1500)
```


United States
India
United King
Samuel L. Jackson 21
Adam Sandler 20
James Franco 19
Nicolas Cage 18
Morgan Freeman 15
Bruce Willis 15
Molly Shannon 15
Seth Rogen 15
Fred Tatasciore 15
Tara Strong 15
Willem Dafoe 14
Pierce Brosnan 14
Alfred Molina 14
Kristen Stewart 14
Johnny Depp 14
Laura Bailey 14
Woody Harrelson 13
Laurence Fishburne 13
Amy Adams 13
Michael Peña 13
Ray Liotta 13
John Travolta 13
Danny Trejo 13
Kate Higgins 13
Chris Rock 12
Anupam Kher 40
Shah Rukh Khan 34
Naseeruddin Shah 31
Akshay Kumar 29
Om Puri 29
Amitabh Bachchan 28
Paresh Rawal 28
Boman Irani 27
Kareena Kapoor 25
Ajay Devgn 21
Salman Khan 20
Kay Kay Menon 19
Nawazuddin Siddiqui 18
Anil Kapoor 18
Rajpal Yadav 17
Yashpal Sharma 17
Asrani 17
Gulshan Grover 17
Tinnu Anand 16
Sanjay Mishra 16
Manoj Joshi 16
Rajesh Sharma 16
Vijay Raaz 16
Aamir Khan 16
Saif Ali Khan 16
John Cleese
Judi Dench
Michael Palin
Brendan Gleeson
Helena Bonham Carter
Johnny Depp 7
James Cosmo 7
Eddie Marsan 7
Samuel West 7
Eric Idle 7
Terry Gilliam 7
Terry Jones 7
Ben Whishaw 6
Pierce Brosnan 6
Natalie Dormer 6
Jason Flemyng 6
Graham Chapman 6
Jim Broadbent 6
Jude Law 5
Nicole Kidman 5
Olga Kurylenko 5
Michael Gambon 5
Anthony Hopkins 5
Clive Owen 5
Paul Bettany 5
0
5
10
15
20
0
10
20
30
40
0
5
Canada
Spain
Japan
Robb Wells 10
John Paul Tremblay 10
John Dunsworth 9
Michela Luci 7
Mike Smith 7
Vincent Tong 6
Colm Feore 6
Liam Neeson 6
Jamie Watson 6
Eric Peterson 6
Nicolas Aqui 6
Cory Doran 6
Andrea Libman 6
Tara Strong 6
Cathy Weseluck 6
Diedrich Bader 5
Mario Casas 9
Luis Tosar 8
Carmen Machi 8
Leonardo Sbaraglia 7
Karra Elejalde 7
Imanol Arias 6
Dani Rovira 6
Javier Gutiérrez 6
Belén Cuesta 6
Alain Hernández 6
Emilio Gutiérrez Caba 6
Álvaro Cervantes 5
Luis Callejo 5
Pol Monen 5
Ramón Barea 5
Marta Etura 5
Yuki Kaji
Junko Takeuchi
Chie Nakamura
Satsuki Yukino
Houko Kuwashima
Minako Kotobuki
Kappei Yamaguchi
Koji Tsujitani
Kumiko Watanabe
Kazuhiko Inoue
Takahiro Sakurai
Rikiya Koyama 6
Koichi Yamadera 6
Noriko Hidaka 5
Ken Narita 5
Akio Otsuka 5

Most common cast members in TV shows from the United States, the United Kingdom

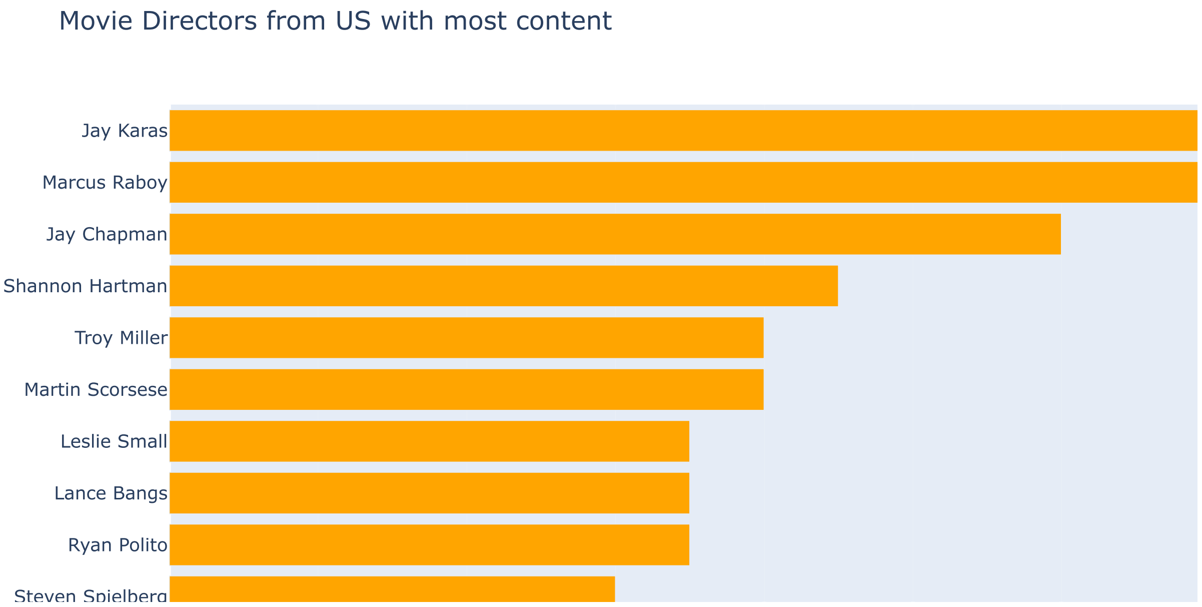
In [22]:
```
traces = []
titles = ["United States", "United Kingdom"]
for title in titles:
        traces.append(country_trace(title, flag="tv_shows"))

fig = make_subplots(rows=1, cols=2, subplot_titles=titles)
for i, trace in enumerate(traces):
    fig.add_trace(trace.data[0], row=1, col=i+1)

fig.update_layout(height=1000, showlegend=False)
fig.show()
```



Display the most prolific movie directors from the United States based on the amount of content they have produced

In [23]:

```
col = "director"
filtered_df = df[(df[col] != "No Data") & (df["type"] == "Movie") & (df["country"] == "United States")]

counter_list = Counter(filtered_df[col].fillna("").str.split(", ").explode().to_list()).most_common(10)
counter_list = [_ for _ in counter_list if _[0] != ""]
labels = [_[0] for _ in counter_list][::-1]
values = [_[1] for _ in counter_list][::-1]
trace1 = go.Bar(y=labels, x=values, orientation="h", name="TV Shows", marker=dict(color="orange"))

data = [trace1]
layout = go.Layout(title="Movie Directors from US with most content", legend=dict(x=0.1, y=1.1, orientatio
fig = go.Figure(data, layout=layout)
fig.show()
```



Contents of the movie directior with the most content.

In [24]:
```
tag = counter_list[0][0]
df["relevant"] = df['director'].fillna("").apply(lambda x : 1 if tag in x else 0)
filtered_df = df[df["relevant"] == 1]
filtered_df[['title', 'release_year', 'listed_in', 'director']]
```

Out[24]:

	title	release_year	listed_in	director
2695	The Main Event	2020	Children & Family Movies, Comedies, Sports Movies	Jay Karas
3646	Demetri Martin: The Overthinker	2018	Stand-Up Comedy	Jay Karas, Demetri Martin
3733	Adam Devine: Best Time of Our Lives	2019	Stand-Up Comedy	Jay Karas
4803	Bill Burr: You People Are All the Same	2012	Stand-Up Comedy	Jay Karas
4863	Ali Wong: Hard Knock Wife	2018	Stand-Up Comedy	Jay Karas
5086	Tom Segura: Disgraceful	2018	Stand-Up Comedy	Jay Karas
5230	Christina P: Mother Inferior	2017	Stand-Up Comedy	Jay Karas
5622	Bill Burr: Walk Your Way Out	2017	Stand-Up Comedy	Jay Karas
5808	Jeff Foxworthy and Larry the Cable Guy: We've ...	2016	Stand-Up Comedy	Jay Karas
5817	Jim Gaffigan: Mr. Universe	2012	Stand-Up Comedy	Jay Karas
5847	Ali Wong: Baby Cobra	2016	Stand-Up Comedy	Jay Karas
5875	Tom Segura: Mostly Stories	2016	Stand-Up Comedy	Jay Karas
5894	Anjelah Johnson: Not Fancy	2015	Stand-Up Comedy	Jay Karas
5899	Demetri Martin: Live (At the Time)	2015	Stand-Up Comedy	Jay Karas
5921	Bill Burr: I'm Sorry You Feel That Way	2014	Stand-Up Comedy	Jay Karas

In [25]:
```
tag = Counter(filtered_df['listed_in']).most_common(1)[0][0]
df["relevant"] = df['listed_in'].fillna("").apply(lambda x : 1 if tag.lower() in x.lower() else 0)
filtered_df = df[df["relevant"] == 1]
filtered_df[filtered_df["country"] == "United States"][["title", "country","release_year"]].head(10)
```

Out[25]:

	title	country	release_year
359	The Original Kings of Comedy	United States	2000
511	Chelsea	United States	2017
826	Bo Burnham: Inside	United States	2021
1189	Nate Bargatze: The Greatest Average American	United States	2021
1191	The Fluffy Movie	United States	2014
1278	Brian Regan: On the Rocks	United States	2021
1352	Tiffany Haddish Presents: They Ready	United States	2021
1450	Eddie Murphy: Raw	United States	1987
1502	London Hughes: To Catch a D*ck	United States	2020
1530	Schulz Saves America	United States	2020

In [ ]:

In [ ]: