In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
netflix_df=pd.read_csv('netflix1 (2).csv')
```

In [4]:
```
netflix_df.head()
```

Out[4]:

	show_id	type	title	director	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	PG-13	90 min	Documentaries
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	TV-MA	1 Season	TV Dramas, TV Horror, TV Mysteries
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	TV-PG	91 min	Children & Family Movies, Comedies
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	TV-MA	125 min	Dramas, Independent Movies, International Movies

In [5]:
```
netflix_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

In [6]:
```
netflix_df.isnull().sum()
```

Out[6]:
```
show_id         0
type            0
title           0
director        0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
dtype: int64
```

In [7]:
```
netflix_df.shape
```

Out[7]:
```
(8790, 10)
```

In [8]:
```
netflix_df.isnull().value_counts()
```

Out[8]:
```
show_id  type   title  director  country  date_added  release_year  rating  duration  listed_in
False    False  False  False     False    False       False         False   False     False        8790
Name: count, dtype: int64
```

In [9]:
```
data= netflix_df.drop_duplicates()
```

In [10]:
```
data.head()
```

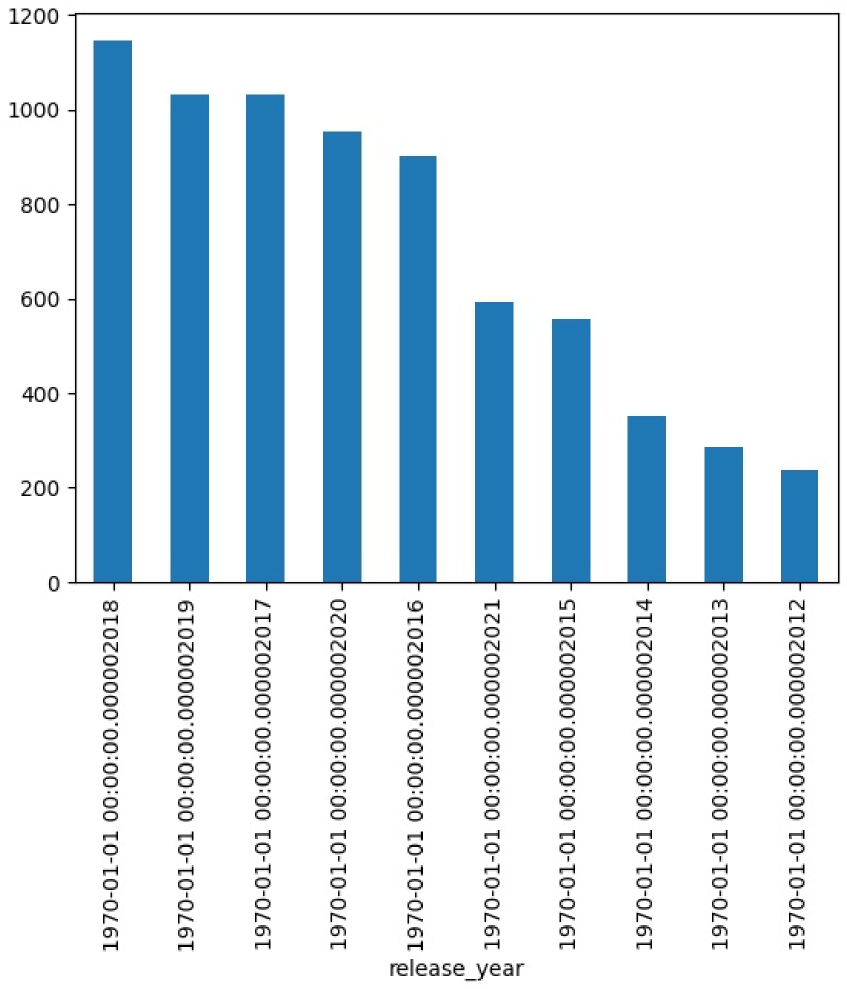
Out[10]:

	show_id	type	title	director	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	PG-13	90 min	Documentaries
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	TV-MA	1 Season	TV Dramas, TV Horror, TV Mysteries
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	TV-PG	91 min	Children & Family Movies, Comedies
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	TV-MA	125 min	Dramas, Independent Movies, International Movies

In [11]:
```
data['release_year']=pd.to_datetime(data['release_year'])
data['Years']=data['release_year'].dt.year
```

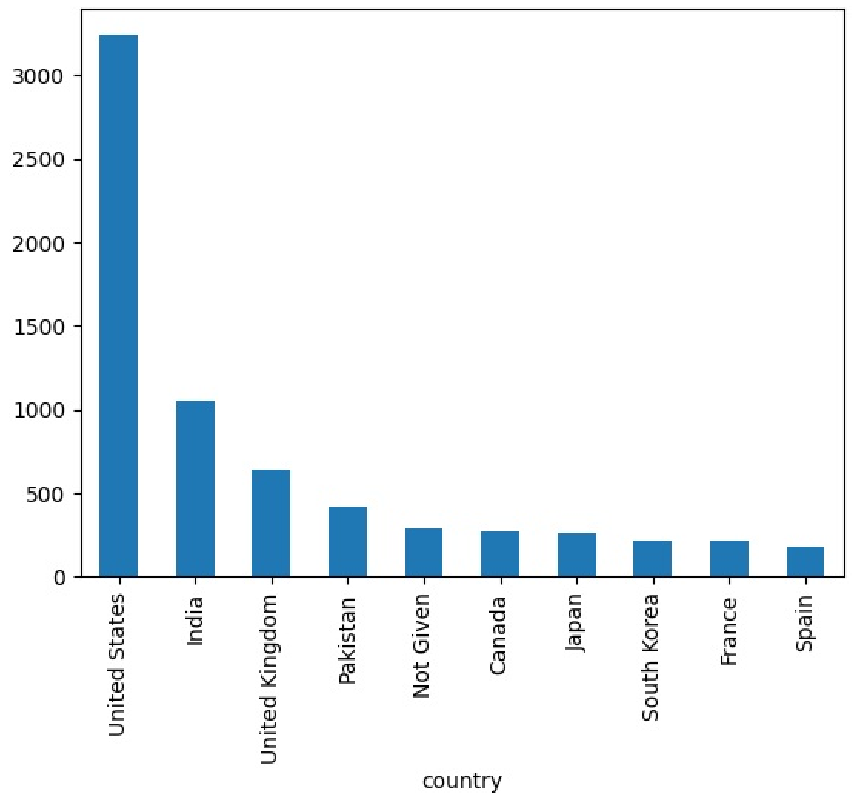
In [12]:
```
year = data.release_year.value_counts()
year[:10].plot(kind = 'bar')
```

Out[12]:
```
<Axes: xlabel='release_year'>
```



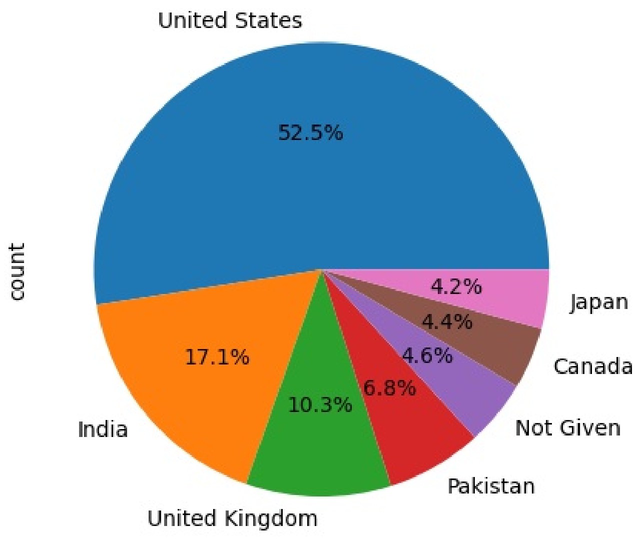
In [13]:
```
top_10_country = data.country.value_counts()
top_10_country[:10].plot(kind = 'bar')
```

Out[13]:
```
<Axes: xlabel='country'>
```



```
In [14]: data_country = data.country.value_counts()
         data_country[:7].plot(kind = 'pie',autopct ='%1.1f%%')
```

```
Out[14]: <Axes: ylabel='count'>
```

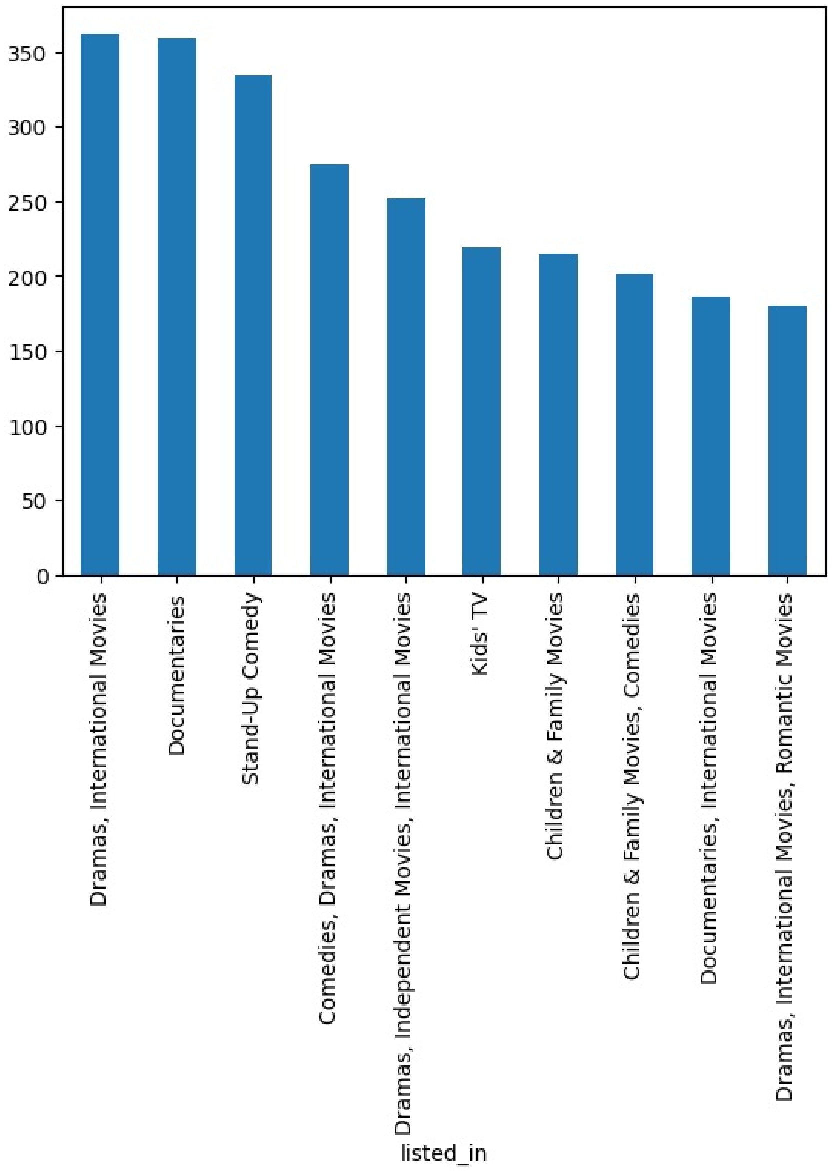


```
In [15]: data.rating.value_counts()
```

Out[15]:
```
rating
TV-MA       3205
TV-14       2157
TV-PG        861
R            799
PG-13        490
TV-Y7        333
TV-Y         306
PG           287
TV-G         220
NR            79
G             41
TV-Y7-FV       6
NC-17          3
UR             3
Name: count, dtype: int64
```

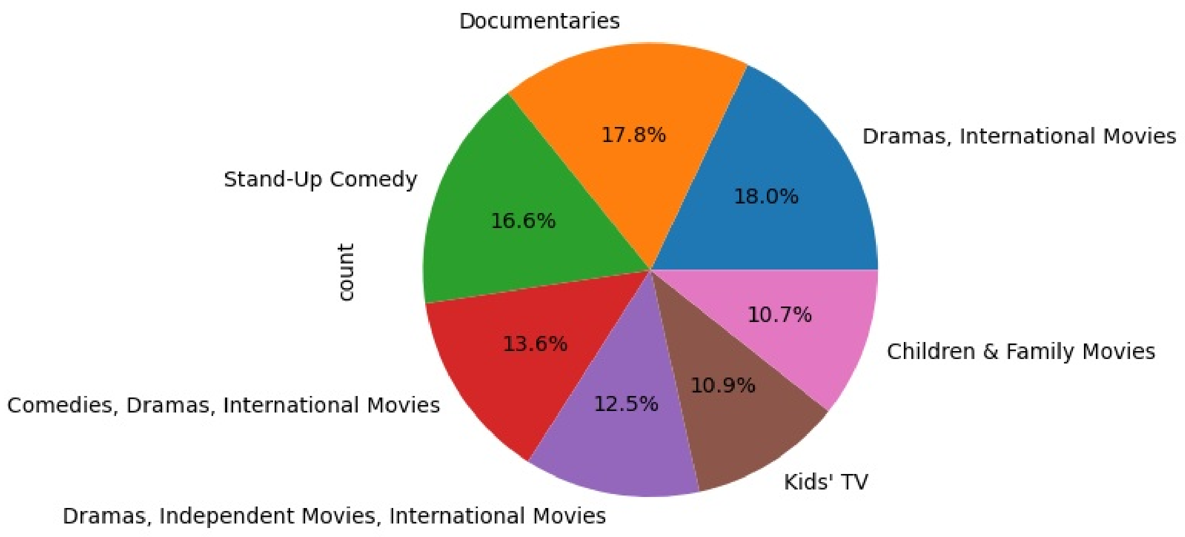
In [16]:
```python
##Most ratings are given to the TV shows in the Dataset


top_10_types = data.listed_in.value_counts()
top_10_types[:10].plot(kind = 'bar')
```

Out[16]:
```
<Axes: xlabel='listed_in'>
```



In [17]:
```python
data_list= data.listed_in.value_counts()
data_list[:7].plot(kind = 'pie',autopct ='%1.1f%%')
```

Out[17]:
```
<Axes: ylabel='count'>
```



Netflix Data: Cleaning, Analysis, and Visualization (Beginner ML Project) This project involves loading, cleaning, analyzing, and visualizing data from a Netflix dataset. We'll use Python libraries like Pandas, Matplotlib, and Seaborn to work through the project. The goal is to explore the dataset, derive insights, and prepare for potential machine learning tasks.

```
In [23]: from wordcloud import WordCloud
```

```
In [26]: df = pd.read_csv('netflix1 (2).csv')
         df.head(10)
```

Out[26]:

	show_id	type	title	director	country	date_added	release_year	rating	duration	listed_in
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	PG-13	90 min	Documentaries
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	TV-MA	1 Season	TV Dramas, TV Horror, TV Mysteries
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	TV-PG	91 min	Children & Family Movies, Comedies
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	TV-MA	125 min	Dramas, Independent Movies, International Movies
5	s9	TV Show	The Great British Baking Show	Andy Devonshire	United Kingdom	9/24/2021	2021	TV-14	9 Seasons	British TV Shows, Reality TV
6	s10	Movie	The Starling	Theodore Melfi	United States	9/24/2021	2021	PG-13	104 min	Comedies, Dramas
7	s939	Movie	Motu Patlu in the Game of Zones	Suhas Kadav	India	5/1/2021	2019	TV-Y7	87 min	Children & Family Movies, Comedies, Music & Mu...
8	s13	Movie	Je Suis Karl	Christian Schwochow	Germany	9/23/2021	2021	TV-MA	127 min	Dramas, International Movies
9	s940	Movie	Motu Patlu in Wonderland	Suhas Kadav	India	5/1/2021	2013	TV-Y7	76 min	Children & Family Movies, Music & Musicals

```
In [27]: # To see the high level data details
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8790 entries, 0 to 8789
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8790 non-null   object
 1   type          8790 non-null   object
 2   title         8790 non-null   object
 3   director      8790 non-null   object
 4   country       8790 non-null   object
 5   date_added    8790 non-null   object
 6   release_year  8790 non-null   int64 
 7   rating        8790 non-null   object
 8   duration      8790 non-null   object
 9   listed_in     8790 non-null   object
dtypes: int64(1), object(9)
memory usage: 686.8+ KB
```

In [28]:
```
def missing_pct(df):
    # Calculate missing value and their percentage for each column
    missing_count_percent = df.isnull().sum() * 100 / df.shape[0]
    df_missing_count_percent = pd.DataFrame(missing_count_percent).round(2)
    df_missing_count_percent = df_missing_count_percent.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_Percentage (%)'
                    }
                )
    df_missing_value = df.isnull().sum()
    df_missing_value = df_missing_value.reset_index().rename(
                    columns={
                            'index':'Column',
                            0:'Missing_value_count'
                    }
                )
    # Sort the data frame
    #df_missing = df_missing.sort_values('Missing_Percentage (%)', ascending=False)
    Final = df_missing_value.merge(df_missing_count_percent, how = 'inner', left_on = 'Column', right_on = 'Col
    Final = Final.sort_values(by = 'Missing_Percentage (%)',ascending = False)
    return Final

missing_pct(df)
```

Out[28]:

	Column	Missing_value_count	Missing_Percentage (%)
0	show_id	0	0.0
1	type	0	0.0
2	title	0	0.0
3	director	0	0.0
4	country	0	0.0
5	date_added	0	0.0
6	release_year	0	0.0
7	rating	0	0.0
8	duration	0	0.0
9	listed_in	0	0.0

The function missing_pct takes a data frame as an input and returns a data frame, where each row corresponds to a column in the original dataframe and contains column's name, number of missing values in that column as well as percentage of the missing values.

This is a standard template that I use for every dataset that I want to analyze.

Handling the missing data and deleting duplicates It is important to handle missing data because any statistical results based on a dataset with non-random missing values could be biased. So you really want to see if these are random or non-random missing values.

Drop the columns which has high number of missing values.

We can impute(filling the missing values using the available information such as mean, median) but we should carefully see the pattern of the column before doing imputation.

For example - You want to fill the height of a person who male. Simpley adding 0 in the missing column would not make sense. So we can take the averega of male height and use that value inplace of missing values.

Rating - manually filling the data usin data from Netflix website

Country - replacing blank countries with the most common country

Cast - replacing null value with "Data not available"

Director - replacing null value with "Data not available"

```
In [30]: # Rating data is mentioned incorrectly for few titles in the input file. Hence correcting it by checking the Ma

         df['rating'] = df['rating'].replace({'74 min': 'TV-MA', '84 min': 'TV-MA', '66 min': 'TV-MA'})
         df['rating'] = df['rating'].replace({'TV-Y7-FV': 'TV-Y7'})
```

```
In [31]: df['rating'].unique()
```

```
Out[31]: array(['PG-13', 'TV-MA', 'TV-PG', 'TV-14', 'TV-Y7', 'TV-Y', 'PG', 'TV-G',
                'R', 'G', 'NC-17', 'NR', 'UR'], dtype=object)
```

```
In [32]: # Renaming vaules for Rating for better understanding
         # Source : https://help.netflix.com/en/node/2064
         df['rating'] = df['rating'].replace({
                         'PG-13': 'Teens - Age above 12',
                         'TV-MA': 'Adults',
                         'PG': 'Kids - with parental guidence',
                         'TV-14': 'Teens - Age above 14',
                         'TV-PG': 'Kids - with parental guidence',
                         'TV-Y': 'Kids',
                         'TV-Y7': 'Kids - Age above 7',
                         'R': 'Adults',
                         'TV-G': 'Kids',
                          'G': 'Kids',
                         'NC-17': 'Adults',
                         'NR': 'NR',
                         'UR' : 'UR'

         })
```

```
In [33]: df['rating'].unique()
```

```
Out[33]: array(['Teens - Age above 12', 'Adults', 'Kids - with parental guidence',
                'Teens - Age above 14', 'Kids - Age above 7', 'Kids', 'NR', 'UR'],
               dtype=object)
```

```
In [37]: import numpy as np
         import pandas as pd

         # Example: check if columns exist before applying operations
         if 'country' in df.columns:
             df['country'] = df['country'].fillna(df['country'].mode()[0])

         if 'cast' in df.columns:
             df['cast'].replace(np.nan, 'No Data', inplace=True)

         if 'director' in df.columns:
             df['director'].replace(np.nan, 'No Data', inplace=True)

         # Drop any remaining missing values
         df.dropna(inplace=True)

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

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\2006500955.py:12: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


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

```
In [38]: print(df.columns.tolist())
```

```
['show_id', 'type', 'title', 'director', 'country', 'date_added', 'release_year', 'rating', 'duration', 'listed_
in']
```

```
In [39]: # splitting the genres in different rows to use it in the viz later

         #df_genre = df[df['title'].isin(['Blood & Water', 'Dick Johnson Is Dead', 'Ganglands' ])]
         df_genre = df[['show_id', 'title','type', 'listed_in' ]]
         df_genre = (df_genre.drop('listed_in', axis=1)
                      .join
                      (
                      df_genre.listed_in
                      .str
```

```
                 .split(', ',expand=True)
                 .stack()
                 .reset_index(drop=True, level=1)
                 .rename('listed_in')
                 ))
```

In [40]:
```
# Creating new columns

df['month'] = pd.DatetimeIndex(df['date_added']).month
```

In [41]:
```
# Total Shows and movies

df_count = df['show_id'].count().sum()
print(df_count)
# Split of showes and TV
df_type = df.groupby('type')['show_id'].count().reset_index()
df_type = df_type.rename(columns = {"show_id":"count_showids"})
```

```
8790
```

In [43]:
```
from plotly.subplots import make_subplots
```

In [44]:
```
import plotly.graph_objects as go
```

In [45]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Example DataFrame (replace with your actual data)
# df_type = your dataframe with 'type' and 'count_showids'

# Create subplot: 1 row, 2 columns
fig = make_subplots(rows=1, cols=2, specs=[[{'type': 'bar'}, {'type': 'pie'}]])

# Add horizontal bar chart
fig.add_trace(
    go.Bar(
        x=df_type['count_showids'],
        y=df_type['type'],
        orientation='h',
        marker=dict(color=["Maroon", "Grey"]),
        showlegend=False,
        text=df_type['count_showids'],
        textposition='auto'
    ),
    row=1, col=1
)

# Add pie chart
fig.add_trace(
    go.Pie(
        labels=df_type['type'],
        values=df_type['count_showids'],
        marker_colors=["Maroon", "Grey"]
    ),
    row=1, col=2
)

fig.update_layout(title_text="Content Type Distribution")
fig.show()
```

In [47]:

```python
# splitting the countries in different rows
#df_genre = df[df['title'].isin(['Blood & Water', 'Dick Johnson Is Dead', 'Ganglands' ])]
df_country = df[['show_id', 'title','type', 'country' ]]
df_country = (df_country.drop('country', axis=1)
             .join
             (
             df_country.country
             .str
             .split(', ',expand=True)
             .stack()
             .reset_index(drop=True, level=1)
             .rename('country')
             ))
```

In [54]:

```python
import plotly.express as px
```

In [55]:

```python
df_country_viz = df_country[["title", "country"]]
df_country_viz = df_country_viz.groupby(['country'])["title"].count().reset_index().sort_values('title', ascend
```

In [58]:

```python
import plotly.express as px

# Step 1: Total content count per country (top 10)
df_country_viz_total = df_country[["title", "country"]]
df_country_viz_total = (
    df_country_viz_total.groupby(['country'])["title"]
    .count()
    .reset_index()
    .sort_values('title', ascending=False)
    .head(10)
    .rename(columns={"title": "total_content_count"})
)

# Step 2: Content count per country split by type (movie/TV)
df_country_viz1 = df_country[["title", "type", "country"]]
df_country_viz1 = (
    df_country_viz1.groupby(['country', 'type'])["title"]
    .count()
    .reset_index()
    .rename(columns={"title": "movies_count"})
)

# Step 3: Merge and calculate percentage share
final1 = df_country_viz_total.merge(df_country_viz1, how='left', on='country')
final1['percentage'] = (final1['movies_count'] / final1['total_content_count']) * 100
final1['percentage'] = final1['percentage'].round(1)
final1['percent_string'] = final1['percentage'].astype(str) + '%'

# Step 4: Plot
fig2 = px.bar(
    final1,
    x='country',
    y='percentage',
    color='type',
    title='Top 10 countries with Movie/TV Show split'
)
fig2.show()
```

In [60]:

```
import plotly.express as px

# Step 1: Get top 10 countries by content count
df_country_viz_total = (
    df_country[["title", "country"]]
    .groupby("country")["title"]
    .count()
    .reset_index()
    .rename(columns={"title": "total_content_count"})
    .sort_values("total_content_count", ascending=False)
    .head(10)
)

# Step 2: Get content count by country and type (movie or TV show)
df_country_viz1 = (
    df_country[["title", "type", "country"]]
    .groupby(["country", "type"])["title"]
    .count()
    .reset_index()
    .rename(columns={"title": "type_count"})  #  Use a clear column name
)

# Step 3: Merge the two datasets
final1 = df_country_viz_total.merge(df_country_viz1, how="left", on="country")

# Step 4: Calculate percentage share by type
final1["percentage"] = (final1["type_count"] / final1["total_content_count"]) * 100
final1["percentage"] = final1["percentage"].round(1)
final1["percent_string"] = final1["percentage"].astype(str) + "%"

# Step 5: Plot
fig2 = px.bar(
    final1,
    x="country",
    y="percentage",
    color="type",
    title="Top 10 Countries: Movie/TV Show Content Split (%)"
)
fig2.show()
```

```
In [61]: df_2 = df.query("type == 'Movie'")
         df_2 = df_2[["title", "rating"]]
         df_2 = df_2.groupby(['rating'])["title"].count().reset_index().sort_values('title', ascending = False)
         df_2 = df_2.rename(columns = {"title": "movies_count"})
         px.bar(df_2, x='rating', y='movies_count', color_discrete_sequence=px.colors.sequential.RdBu,
                title='For which category the maximum content(Movies) are uploaded? ')
```

```
In [62]: df_3 = df.query("type == 'TV Show'")
         df_3 = df_3[["title", "rating"]]
         df_3 = df_3.groupby('rating')["title"].count().reset_index().sort_values('title', ascending = False)
         df_3 = df_3.rename(columns = {"title": "movies_count"})
         px.bar(df_3, x='rating', y='movies_count', color_discrete_sequence=['grey'],
                title='For which category the maximum content(TV Shows) are uploaded?')
```

```
In [63]: df_5 = df.query("release_year >= 2007")
         df_5 = df_5.groupby("release_year")["show_id"].count().reset_index()

         fig = px.area(df_5, x='release_year', y='show_id', color_discrete_sequence=px.colors.sequential.RdBu,
               title='Overall content release Trend')
         fig.show()
```

In 2007, Netflix introduced streaming media and video on demand. We see a slow in the beginning but then it picked up in 2014-2015 and there is a rapid increase till 2018.

By 2018, the content on netlix was 13 times of 2007 year's content. But it has declined since 2019 since the beginning of covid. The other factor could be - In 2019, Disney plus was also launched. Films and television series produced by The Walt Disney Studios and Walt Disney Television, such as Marvel movies moved to Disney plus.

```
In [64]: df_4 = df.query("type == 'Movie'")
         df_4 = df.query("release_year >= 2007")
         df_4 = df_4.groupby(["type","release_year"])["show_id"].count().reset_index()
```

```
fig  = px.line(df_4, x='release_year', y='show_id', color = 'type',
       title='Movies/TV Show release yearly Trend')
fig.show()
```

In [65]:
```
#df_4 = df.query("type == 'Movie'")
df_4 = df.query("release_year >= 2007")
df_4 = df_4.groupby(["type","release_year"])["show_id"].count().reset_index()
df_4_movie = df_4.query("type == 'Movie'")
df_4_show = df_4.query("type == 'TV Show'")

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=  df_4_movie['release_year'],
    y= df_4_movie['show_id'],
    showlegend=True,
    text = df_4_movie['show_id'],

    name='Movie',
    marker_color='Maroon'

))
fig.add_trace(go.Scatter(
    x=  df_4_show['release_year'],
    y= df_4_show['show_id'],
    showlegend=True,
    text = df_4_show['show_id'],

    name='TV Show',
    marker_color='Grey'
))

fig.update_traces( mode='lines+markers')
fig.update_layout(title_text = 'Movies/TV Show release yearly Trend' )
fig.show()
```

```
In [66]: df_4 = df.query("release_year >= 2007")

         df_4 = df_4[["type","month",'release_year', "show_id"]]
         df_4 = df_4.groupby(['release_year', 'month', 'type'])['show_id'].count().reset_index()
         df_4 = df_4.rename(columns = {"show_id": "total_shows"})
         df_4 = df_4.groupby(['month', 'type'])['total_shows'].mean().reset_index()



         fig  = px.line(df_4, x='month', y='total_shows', color = 'type',
                title='All years Movies/TV Show release Month Trend')
         fig.show()
```

```
In [67]: df_4 = df.query("release_year >= 2007")

         df_4 = df_4[["type","month",'release_year', "show_id"]]
         df_4 = df_4.groupby(['release_year', 'month', 'type'])['show_id'].count().reset_index()
         df_4 = df_4.rename(columns = {"show_id": "total_shows"})
         df_4 = df_4.groupby(['month', 'type'])['total_shows'].mean().reset_index()
```

```python
df_4_movie = df_4.query("type == 'Movie'")
df_4_show = df_4.query("type == 'TV Show'")

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=  df_4_movie['month'],
    y= df_4_movie['total_shows'],
    showlegend=True,
    text = df_4_movie['total_shows'],
    name='Movie',
    marker_color='Maroon'

))
fig.add_trace(go.Scatter(
    x=  df_4_show['month'],
    y= df_4_show['total_shows'],
    showlegend=True,
    text = df_4_show['total_shows'],
    name='TV Show',
    marker_color='Grey'
))

fig.update_traces( mode='lines+markers')
fig.update_layout(title_text = 'Movies/TV Shows average release monthly trend' )
fig.show()
```

In [68]:

```python
def trend_yearwise(year):

    title = (f'Movies/TV Show release Month Trend for year {year}' )
    df_6 = df.query("release_year == @year")
    df_6 = df_6.groupby(["type","month"])["show_id"].count().reset_index()
    df_6_movie = df_6.query("type == 'Movie'")
    df_6_show = df_6.query("type == 'TV Show'")

    fig = go.Figure()
    fig.add_trace(go.Scatter(
    x=  df_6_movie['month'],
    y= df_6_movie['show_id'],
    showlegend=True,
    text = df_6_movie['show_id'],
    name='Movie',
    marker_color='Maroon'

    ))
    fig.add_trace(go.Scatter(
    x=  df_6_show['month'],
    y= df_6_show['show_id'],
    showlegend=True,
    text = df_6_show['show_id'],
    name='TV Show',
    marker_color='Grey'
    ))

    fig.update_traces( mode='lines+markers')
    fig.update_layout(title_text =  title )
    fig.show()

trend_yearwise(2019)
```

In [69]:

```
df_genre_viz = df_genre[["title", "type", "listed_in"]]
df_genre_viz = df_genre_viz.groupby(['listed_in', 'type'])["title"].count().reset_index().sort_values('title')
df_genre_viz = df_genre_viz.rename(columns = {"title": "movies_count", "listed_in": "Genre"})

df_genre_movie = df_genre_viz.query("type == 'Movie'")
df_genre_tvshow = df_genre_viz.query("type == 'TV Show'")
# fig1 = px.bar(df_genre_movie, x='movies_count', y='Genre', color_discrete_sequence=px.colors.sequential.RdBu,
#         title='For which Genre the maximum content(Movies) are uploaded? ', height=600)
# fig2 = px.bar(df_genre_tvshow, x='Genre', y='movies_count', color_discrete_sequence=['Grey'],
#         title='For which Genre the maximum content(Shows) are uploaded? ')
#fig1.show()
#fig2.show()
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'bar'}, {'type':'bar'}]],
                   subplot_titles = ['For which Genre the maximum Movies are uploaded?', 'For which Genre the ma
                   horizontal_spacing = 0.3)
fig.add_trace(

    go.Bar(x= df_genre_movie['movies_count'], y= df_genre_movie['Genre'], orientation = 'h', marker_color='Maro
           text = df_type['count_showids'], textposition='auto'),
    row=1, col=1)

fig.add_trace(

    go.Bar(x= df_genre_tvshow['movies_count'], y= df_genre_tvshow['Genre'], orientation = 'h', marker_color = '(
    row=1, col=2)

fig.update_layout( height = 600)
fig.show()
```

```
In [70]: df_9 = df.query("type == 'TV Show'")
         df_9 = df_9[[ "title", "duration"]]
         df_9 = df_9.groupby(['duration'])["title"].count().reset_index().sort_values('title', ascending = False)
         #df_9 = df_9['duration'].replace("seasons", "")
         df_9 = df_9.rename(columns = {"title": "TV Shows", "duration" : "Seasons"})


         df_10 = df.query("type == 'Movie'")
         df_10['duration'] = df_10['duration'].fillna("0")
         df_10['duration'] = df_10['duration'].str.split(" ").str[0].astype(int)




         fig_show = px.bar(df_9, x='Seasons', y='TV Shows', color_discrete_sequence=['grey'],
                title='TV Shows seasons ')
         fig_Movie = px.histogram(df_10, x="duration" , nbins = 20, color_discrete_sequence=px.colors.sequential.RdBu
                           , title = "Movie Duration")

         fig_Movie.show()
         fig_show.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\326621815.py:9: 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/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy

C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\326621815.py:10: 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/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
```

The duration for most movies on netflix falls between 80-120 mins with very few movies more than 150 mins.

Most shows on Netflix has only season1.

Conclusion We did exploratory data analysis on Netflix Movie Data. We found a lot of insights from the data. This is the first step in our series.

Next, we will engineer useful features and begin developing our recommendation model.

In [71]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_style('ticks')
palette = sns.color_palette("ch:s=.15,rot=-.15")
```

In [72]:
```
df.head()
```

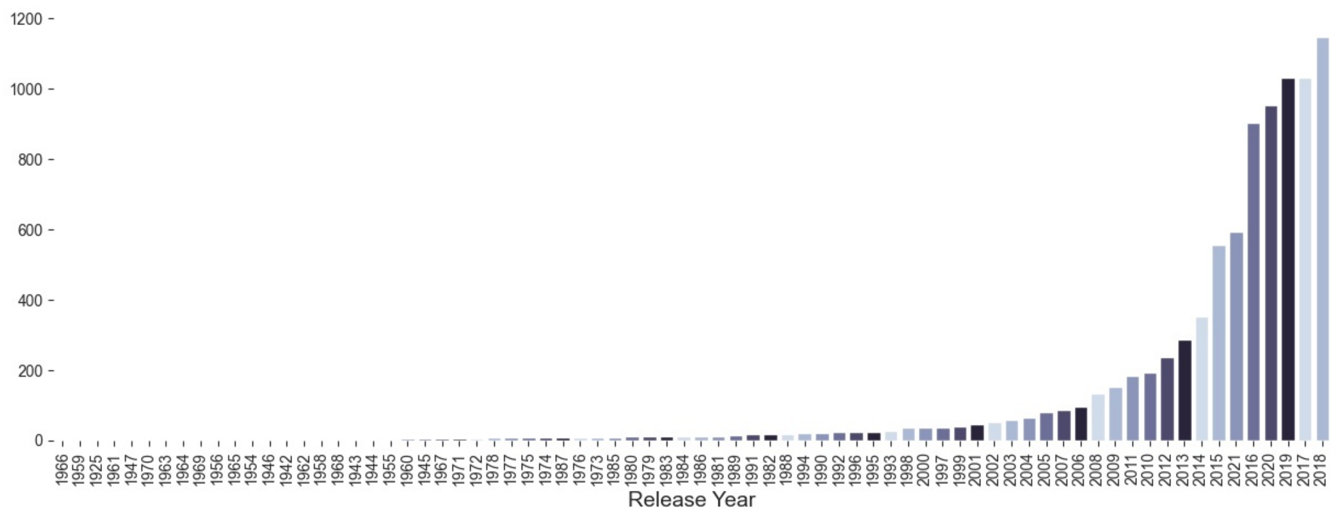
Out[72]:

	show_id	type	title	director	country	date_added	release_year	rating	duration	listed_in	month
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	Teens - Age above 12	90 min	Documentaries	9
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	Adults	1 Season	Crime TV Shows, International TV Shows, TV Act...	9
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	Adults	1 Season	TV Dramas, TV Horror, TV Mysteries	9
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	Kids - with parental guidence	91 min	Children & Family Movies, Comedies	9
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	Adults	125 min	Dramas, Independent Movies, International Movies	9

In [73]:
```python
#UNIVARIATE ANALISYS

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
sns.countplot(data = df, x = 'release_year',ax=ax, order = df['release_year'].value_counts(ascending=True).inde
ax.set_xlabel(xlabel='Release Year', size=14)
ax.set_ylabel(ylabel=" ")
sns.despine(bottom=True, left=True)
plt.xticks(rotation=90)
fig.text(0.5, 1,"Release Year Distribution")
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\4191136668.py:4: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.


C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\4191136668.py:4: UserWarning:


The palette list has fewer values (6) than needed (74) and will cycle, which may produce an uninterpretable plot
.
```

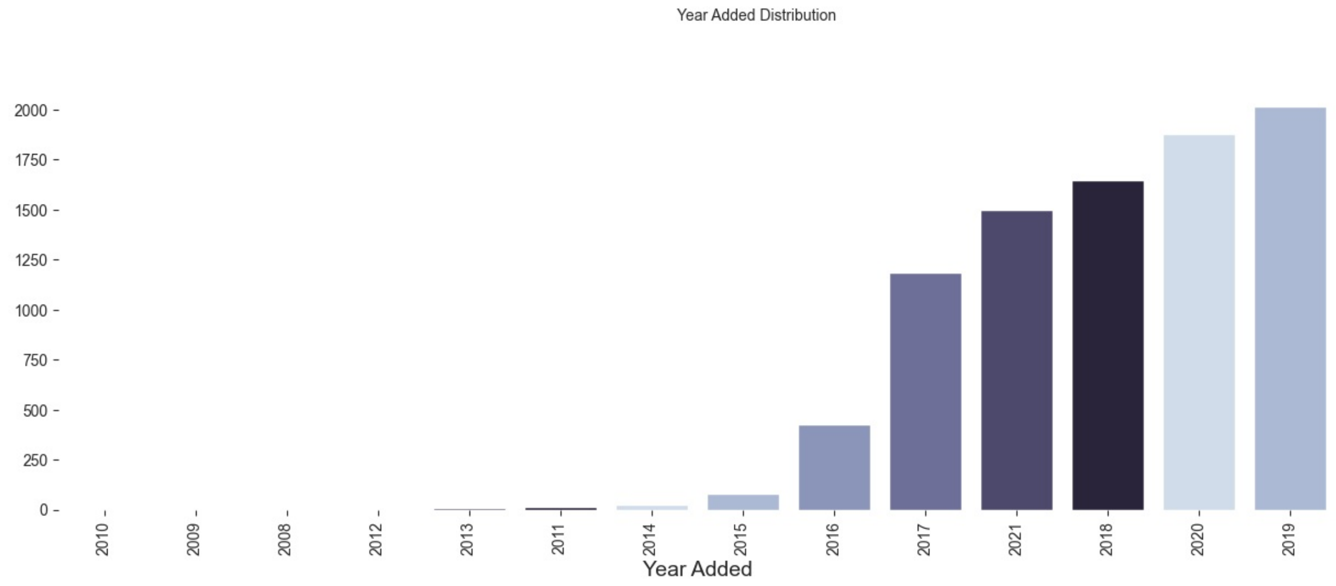
Release Year Distribution



In [74]:
```python
df_year_added = df[df['date_added'].notna()]
df_year_added['year_added'] = pd.DatetimeIndex(df_year_added['date_added']).year
```

In [75]:
```python
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
sns.countplot(data = df_year_added, x = 'year_added',ax=ax, order = df_year_added['year_added'].value_counts(as
ax.set_xlabel(xlabel='Year Added', size=14)
ax.set_ylabel(ylabel=" ")
sns.despine(bottom=True, left=True)
plt.xticks(rotation=90)
fig.text(0.5, 1,"Year Added Distribution")
plt.show()
```

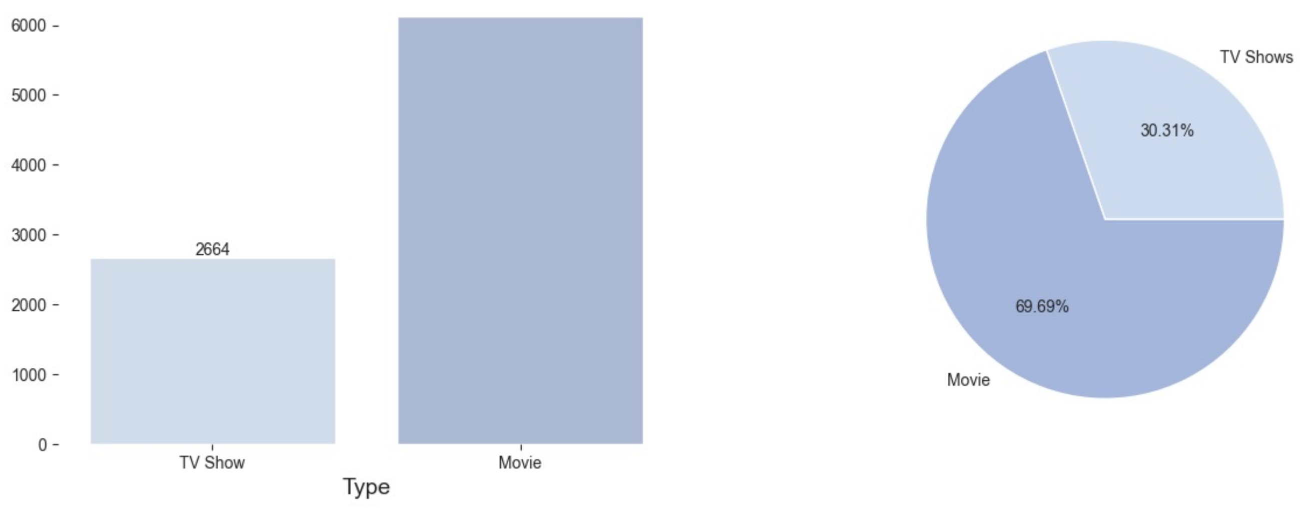
```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\1260906006.py:2: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.


C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\1260906006.py:2: UserWarning:


The palette list has fewer values (6) than needed (14) and will cycle, which may produce an uninterpretable plot
.
```



In [76]:
```python
fig, ax = plt.subplots(1, 2, figsize=(15, 5))
sns.countplot(data = df, x = 'type',ax=ax[0], order = df['type'].value_counts(ascending=True).index,palette=pal
ax[0].set_xlabel(xlabel='Type', size=14)
ax[0].set_ylabel(ylabel=" ")
ax[0].bar_label(ax[0].containers[0])
sns.despine(bottom=True, left=True)
df['type'].value_counts(ascending=True).plot(kind='pie',ax=ax[1],autopct="%.2f%%",colors=palette,labels=['TV Sh
ax[1].set_xlabel(xlabel=" ")
ax[1].set_ylabel(ylabel=" ")
fig.text(0.5, 1,"Type Distribution")
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3722943828.py:2: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.


C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3722943828.py:2: UserWarning:

The palette list has more values (6) than needed (2), which may not be intended.
```

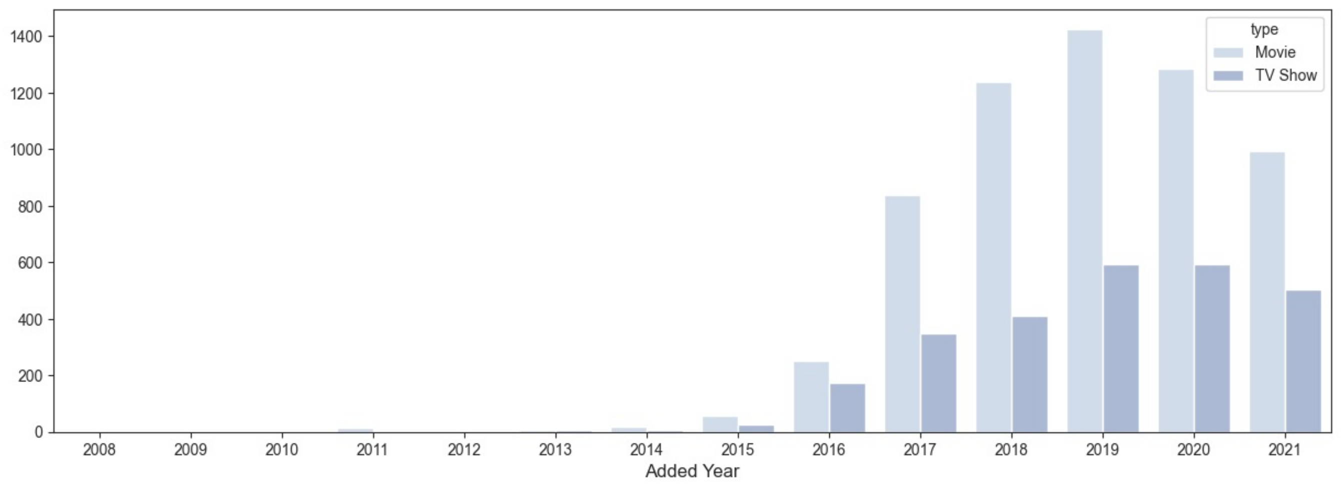
Type Distribution



In [77]:
```
#MULTIVARIATE ANALISYS: Type by added year

fig, ax = plt.subplots(1, 1, figsize=(15, 5))
sns.countplot(data=df_year_added,x='year_added',hue='type',palette=palette)
ax.set_xlabel(xlabel='Added Year', size=12)
ax.set_ylabel(ylabel=' ')
fig.text(0.5, 1,"Type added by year")
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\877318736.py:4: UserWarning:

The palette list has more values (6) than needed (2), which may not be intended.
```

Type added by year



In [80]:
```
#GENRES ANALISYS
```

```
#Get Main Genres from all set of data. Listed_in column consist of multiple genres. For Wordclouds visualizatio

def get_secondary_genre(text):
    if len(text.split(","))>1:
        secondary= text.split(",")[1].strip()
    else:
        secondary = text.split(",")[0].strip()
    return secondary


df['main_genre']= df['listed_in'].apply(lambda x: x.split(",")[0])
df['secondary_genre']= df['listed_in'].apply(lambda x: get_secondary_genre(x))
```

In [81]:
```
df.head()
```

Out[81]:

	show_id	type	title	director	country	date_added	release_year	rating	duration	listed_in	month	main_ge
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	United States	9/25/2021	2020	Teens - Age above 12	90 min	Documentaries	9	Documenta
1	s3	TV Show	Ganglands	Julien Leclercq	France	9/24/2021	2021	Adults	1 Season	Crime TV Shows, International TV Shows, TV Act...	9	Crime Sh
2	s6	TV Show	Midnight Mass	Mike Flanagan	United States	9/24/2021	2021	Adults	1 Season	TV Dramas, TV Horror, TV Mysteries	9	TV Dra
3	s14	Movie	Confessions of an Invisible Girl	Bruno Garotti	Brazil	9/22/2021	2021	Kids - with parental guidence	91 min	Children & Family Movies, Comedies	9	Childre Family Mo
4	s8	Movie	Sankofa	Haile Gerima	United States	9/24/2021	1993	Adults	125 min	Dramas, Independent Movies, International Movies	9	Dra

In [82]:
```
#MAPPING GENRES

mapping_genres_dict ={'Documentaries':'Documentaries',
     'British TV Shows': 'International',
     'International TV Shows':'International',
     'Crime TV Shows':'Crime',
    'Docuseries':'Documentaries',
     'TV Dramas':'Dramas',
     'Children & Family Movies':'Children & Family Movies',
     'Dramas':'Dramas',
     'Comedies':'Comedies',
     'TV Comedies':'Comedies',
     'Thrillers':'Thrillers',
     'TV Thrillers':'Thrillers',
     'Horror Movies':'Horror',
     "Kids' TV":"Kids' TV",
     'Action & Adventure':'Action & Adventure',
     'Reality TV':'Reality TV',
     'Anime Series':'Anime',
     'International Movies':'International',
     'Sci-Fi & Fantasy':'Sci-Fi & Fantasy',
       'Classic Movies':'Classic',
     'TV Shows':'TV Shows',
     'Stand-Up Comedy':'Stand-Up Comedy & Talk Shows',
       'TV Action & Adventure':'Action & Adventure',
     'Movies':'Movies',
     'Korean TV Shows':'International',
     'Stand-Up Comedy & Talk Shows':'Stand-Up Comedy & Talk Shows',
       'Classic & Cult TV':'Classic',
     'Anime Features':'Anime',
       'Cult Movies':'Cult',
     'Classic Movies':'Classic',
     'Independent Movies':'Independent Movies',
     'TV Horror':'Horror',
        'Music & Musicals':'Music & Musicals',
       'LGBTQ Movies':'LGBTQ',
     'Sports Movies':'Sport',
          'Spanish-Language TV Shows':'International',
         'Romantic TV Shows':'Romantic',
         'Romantic Movies':'Romantic',
         'TV Action & Adventure':'Action & Adventure',
         'TV Sci-Fi & Fantasy': 'Sci-Fi & Fantasy',
      'International TV Shows':'International',
     'Faith & Spirituality':'Faith & Spirituality',
      'Science & Nature TV':'Science & Nature'
                     }

df['main_genre']=df['main_genre'].map(mapping_genres_dict)
df['secondary_genre']=df['secondary_genre'].map(mapping_genres_dict)
```

In [83]:
```
df['main_genre'].value_counts()
```

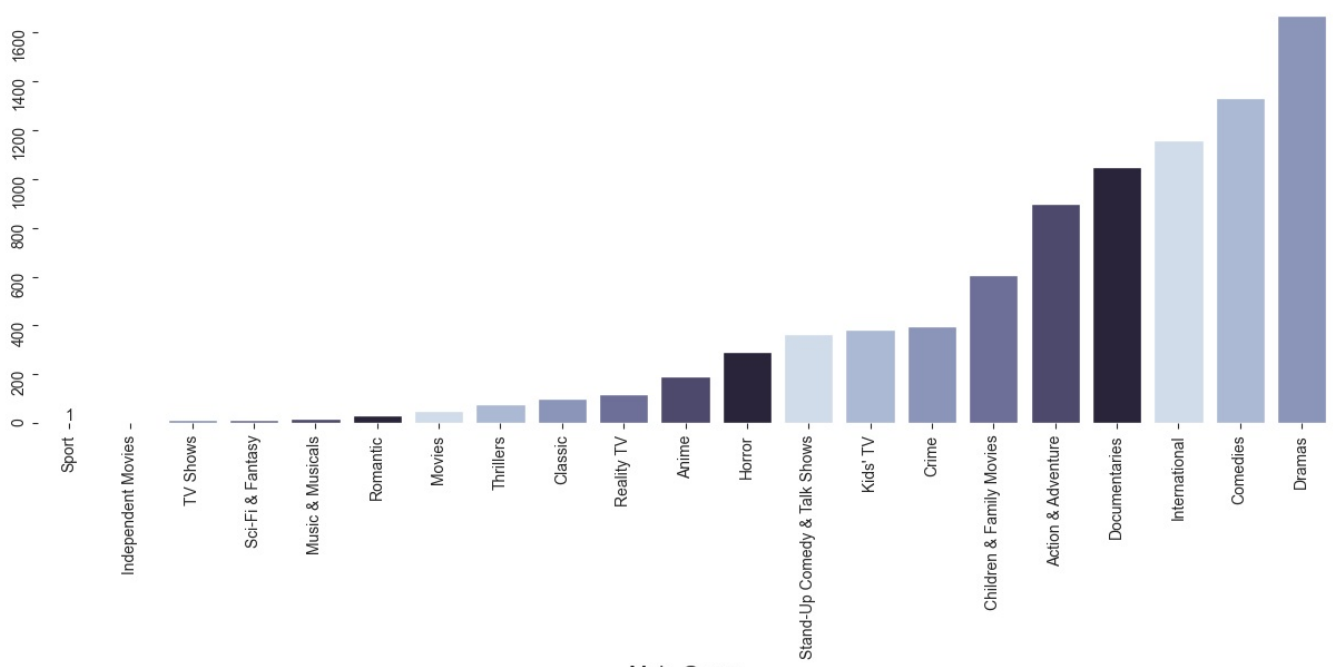
Out[83]:
```
main_genre
Dramas                          1666
Comedies                        1329
International                   1155
Documentaries                   1049
Action & Adventure               898
Children & Family Movies         605
Crime                            399
Kids' TV                         385
Stand-Up Comedy & Talk Shows     368
Horror                           286
Anime                            195
Reality TV                       120
Classic                          100
Thrillers                         65
Movies                            53
Romantic                          35
Independent Movies                20
Music & Musicals                  18
TV Shows                          16
Sci-Fi & Fantasy                  14
Cult                              12
LGBTQ                              1
Sport                              1
Name: count, dtype: int64
```

In [84]:
```
##For all rare genres (LGBTQ, Cult, Sport,Independent Movies),if exists, I'm going to choose secondary genre

df.loc[df['main_genre']=='LGBTQ']
df.loc[df['main_genre'] == 'LGBTQ', ['main_genre']] = df[df['main_genre'] == 'LGBTQ']['secondary_genre']
df.loc[df['main_genre'] == 'Cult', ['main_genre']] = df[df['main_genre'] == 'Cult']['secondary_genre']
df.loc[df['main_genre'] == 'Sport', ['main_genre']] = df[df['main_genre'] == 'Sport']['secondary_genre']
df.loc[df['main_genre'] == 'Independent Movies', ['main_genre']] = df[df['main_genre'] == 'Independent Movies']
```

In [85]:
```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
sns.countplot(data = df, x = 'main_genre',ax=ax, order = df['main_genre'].value_counts(ascending=True).index,pa
ax.set_xlabel(xlabel='Main Genre', size=14)
ax.set_ylabel(ylabel=" ")
ax.bar_label(ax.containers[0])
sns.despine(bottom=True, left=True)
ax.tick_params(labelrotation=90)
fig.text(0.5, 1,"Main Genre Distribution")
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3630741976.py:2: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.


C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3630741976.py:2: UserWarning:


The palette list has fewer values (6) than needed (21) and will cycle, which may produce an uninterpretable plot
.
```

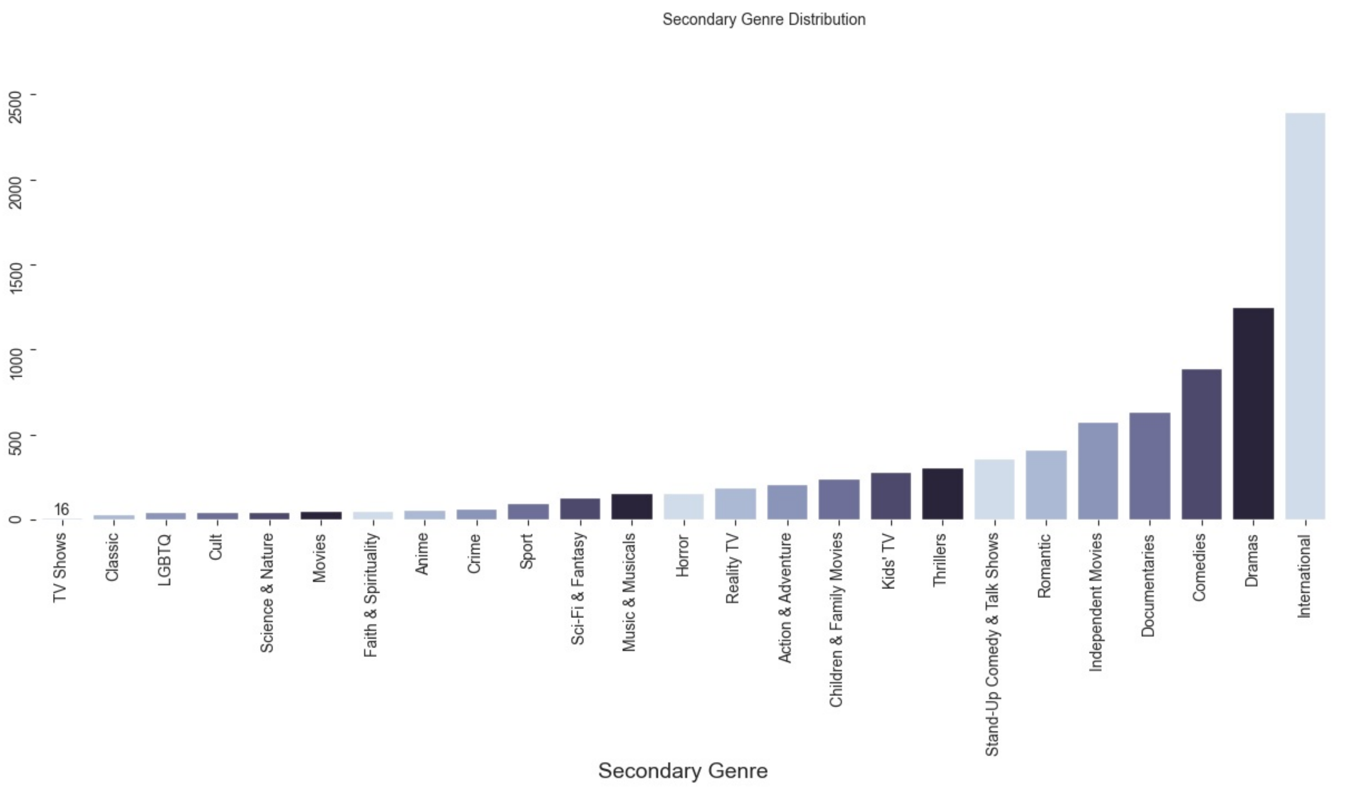
Main Genre Distribution



Main Genre

In [86]:
```
fig, ax = plt.subplots(1, 1, figsize=(15, 5))
sns.countplot(data = df, x = 'secondary_genre',ax=ax, order = df['secondary_genre'].value_counts(ascending=True
ax.set_xlabel(xlabel='Secondary Genre', size=14)
ax.set_ylabel(ylabel=" ")
ax.bar_label(ax.containers[0])
sns.despine(bottom=True, left=True)
ax.tick_params(labelrotation=90)
fig.text(0.5, 1,"Secondary Genre Distribution")
plt.show()
```

```
C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3006812061.py:2: FutureWarning:



Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable
to `hue` and set `legend=False` for the same effect.


C:\Users\LENOVO\AppData\Local\Temp\ipykernel_32556\3006812061.py:2: UserWarning:


The palette list has fewer values (6) than needed (25) and will cycle, which may produce an uninterpretable plot
.
```



In [88]:
```python
print(df.columns)
```

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

In [89]:
```python
import plotly.express as px

# Step 1: Get total content count per country (Top 10)
df_country_viz_total = (
    df_country[["show_id", "country"]]
    .groupby("country")["show_id"]
    .count()
    .reset_index()
    .rename(columns={"show_id": "total_content_count"})
    .sort_values("total_content_count", ascending=False)
    .head(10)
)

# Step 2: Get type-wise count (Movie/TV Show) for these countries
df_country_viz1 = (
    df_country[df_country["country"].isin(df_country_viz_total["country"])]
    .groupby(["country", "type"])["show_id"]
    .count()
    .reset_index()
    .rename(columns={"show_id": "type_count"})
)

# Step 3: Merge and calculate percentage
final1 = df_country_viz_total.merge(df_country_viz1, on="country", how="left")
final1["percentage"] = (final1["type_count"] / final1["total_content_count"]) * 100
final1["percentage"] = final1["percentage"].round(1)
final1["percent_string"] = final1["percentage"].astype(str) + "%"

# Step 4: Visualize
fig2 = px.bar(
    final1,
    x="country",
    y="percentage",
    color="type",
    text="percent_string",
    title="Top 10 Countries: Movie/TV Show Content Split (%)",
    labels={"percentage": "Content Share (%)"}
)
fig2.update_traces(textposition="outside")
fig2.update_layout(barmode="stack", uniformtext_minsize=8, uniformtext_mode='hide')
fig2.show()
```

In [91]:
```
df['pseudo_description'] = df['title'] + ' is a ' + df['main_genre'] + ' show listed under ' + df['listed_in']
```

In [92]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[92]:
```
True
```

In [93]:
```
def text_preprocessing(columns_pandas, stop_words):
    if pd.isnull(columns_pandas):
        return ""
    tokens = nltk.word_tokenize(columns_pandas)
    tokens = [w for w in tokens if w.isalpha()]
    tokens = [word for word in tokens if word.lower() not in stop_words]
    return " ".join(tokens)
```

In [95]:
```
df['listed_in']  # This is a valid column
```

Out[95]:
```
0                                           Documentaries
1       Crime TV Shows, International TV Shows, TV Act...
2                          TV Dramas, TV Horror, TV Mysteries
3                          Children & Family Movies, Comedies
4        Dramas, Independent Movies, International Movies
                              ...
8785                         International TV Shows, TV Dramas
8786                                                 Kids' TV
8787    International TV Shows, Romantic TV Shows, TV ...
8788                                                 Kids' TV
8789                                                 Kids' TV
Name: listed_in, Length: 8790, dtype: object
```

In [107…
```
!pip install nltk
```

```
Requirement already satisfied: nltk in c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages
(3.9.1)
Requirement already satisfied: click in c:\users\lenovo\appdata\local\programs\python\python313\lib\site-package
s (from nltk) (8.1.8)
Requirement already satisfied: joblib in c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packag
es (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in c:\users\lenovo\appdata\local\programs\python\python313\lib\si
te-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in c:\users\lenovo\appdata\local\programs\python\python313\lib\site-packages
(from nltk) (4.67.1)
Requirement already satisfied: colorama in c:\users\lenovo\appdata\local\programs\python\python313\lib\site-pack
ages (from click->nltk) (0.4.6)
```

```
[notice] A new release of pip is available: 25.0.1 -> 25.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [108…

```python
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\LENOVO\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[108…

```
True
```

In [ ]: