In [ ]:
```
Exploratory Data Analysis of Disney + Hotstar
```

In [ ]:
```
Mohammad Areeb
Linkedin : www.linkedin.com/in/mohammadareeb2544
Github : https://github.com/areeb399
```

In [ ]:
```
Objective:
    The aim of case study is to analyze content type in recent years and how Disney
    among audience.
```

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

In [2]:
```
#reading a CSV file:

df = pd.read_csv("titles.csv")
df
```

Out[2]:

	id	title	type	description	release_year	age_certification	runtime	g
0	tm74391	Fantasia	MOVIE	Walt Disney's timeless masterpiece is an extra...	1940	G	120	ani
1	tm67803	Snow White and the Seven Dwarfs	MOVIE	A beautiful girl, Snow White, takes refuge in ...	1937	G	83	
2	tm82546	Pinocchio	MOVIE	Lonely toymaker Geppetto has his wishes answer...	1940	G	88	ani
3	tm79357	Bambi	MOVIE	Bambi's tale unfolds from season to season as ...	1942	G	70	ani
4	tm62671	Treasure Island	MOVIE	Enchanted by the idea of locating treasure bur...	1950	PG	96	
...	...	...	...	...	...	...	...	
1530	tm1071287	Killer Shark vs. Killer Whale	MOVIE	Scientists dive deep on the mysterious and unu...	2021	PG-13	44	docume
1531	tm1091101	Far From Raven's Home	MOVIE	Our gang is off for an exotic vacation of a li...	2021	G	11	ani
1532	tm1075680	World's Most Dangerous Shark?	MOVIE	The Great White has a scary reputation, but Ex...	2021	PG-13	44	docume
1533	tm1133820	The Magic Maker	MOVIE	Famed magician Adam Trent breaks the number on...	2021	PG	43	
1534	tm1091117	Smoky Mountain	MOVIE	Park Rangers	2021	PG	42	docume

id	title	type	description	release_year	age_certification	runtime	
	Park Rangers		work to protect and manage black ...				

1535 rows × 20 columns

In [3]:
```
#display the first 10 rows

df.head(10)
```

Out[3]:

	id	title	type	description	release_year	age_certification	runtime	genre1	
0	tm74391	Fantasia	MOVIE	Walt Disney's timeless masterpiece is an extra...	1940	G	120	animation	
1	tm67803	Snow White and the Seven Dwarfs	MOVIE	A beautiful girl, Snow White, takes refuge in ...	1937	G	83	fantasy	
2	tm82546	Pinocchio	MOVIE	Lonely toymaker Geppetto has his wishes answer...	1940	G	88	animation	
3	tm79357	Bambi	MOVIE	Bambi's tale unfolds from season to season as ...	1942	G	70	animation	
4	tm62671	Treasure Island	MOVIE	Enchanted by the idea of locating treasure bur...	1950	PG	96	family	
5	tm61729	The Adventures of Ichabod and Mr. Toad	MOVIE	The Wind in the Willows: Concise version of Ke...	1949	G	68	fantasy	
6	tm61052	Cinderella	MOVIE	Cinderella has faith her dreams of a better li...	1950	G	74	fantasy	ar
7	tm87946	Dumbo	MOVIE	Dumbo is a baby elephant born with over-sized ...	1941	G	63	animation	
8	tm4623	The Three Caballeros	MOVIE	For Donald's birthday he receives a box with t...	1944	G	71	animation	
9	tm77826	The Reluctant Dragon	MOVIE	Humorist Robert Benchley	1941	G	74	fantasy	ar

	id	title	type	description	release_year	age_certification	runtime	genre1
				attempts to find Walt				

In [4]:
```
#display the last 10 rows

df.tail(10)
```

Out[4]:

	id	title	type	description	release_year	age_certification	runtime	
1525	ts313961	Vets on the Beach	SHOW	Four vets from across Australia come together ...	2021	TV-PG	44	do
1526	tm1076899	A Spark Story	MOVIE	A documentary film providing an exclusive and ...	2021	NaN	87	do
1527	tm1064170	Shark Gangs	MOVIE	Scientists have discovered and investigate the...	2021	PG-13	44	do
1528	tm999348	Own the Room	MOVIE	Follows five young star students on their jour...	2021	PG	91	do
1529	tm1002614	Built for Mars: The Perseverance Rover	MOVIE	BUILT FOR MARS: THE PERSEVERANCE ROVER goes be...	2021	PG-13	88	do
1530	tm1071287	Killer Shark vs. Killer Whale	MOVIE	Scientists dive deep on the mysterious and unu...	2021	PG-13	44	do
1531	tm1091101	Far From Raven's Home	MOVIE	Our gang is off for an exotic vacation of a li...	2021	G	11	
1532	tm1075680	World's Most Dangerous Shark?	MOVIE	The Great White has a scary reputation, but Ex...	2021	PG-13	44	do
1533	tm1133820	The Magic Maker	MOVIE	Famed magician Adam Trent breaks the number on...	2021	PG	43	
1534	tm1091117	Smoky Mountain Park Rangers	MOVIE	Park Rangers work to protect and manage black ...	2021	PG	42	do

In [6]:
```
df.isna()
```

Out[6]:

	id	title	type	description	release_year	age_certification	runtime	genre1	genre2	ger
0	False	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
1530	False	False	False	False	False	False	False	False	True	
1531	False	False	False	False	False	False	False	False	False	
1532	False	False	False	False	False	False	False	False	True	
1533	False	False	False	False	False	False	False	True	True	
1534	False	False	False	False	False	False	False	False	True	

1535 rows × 20 columns

In [7]:
```
#calculate the number of missing values (NaN or None) in each column:

df.isna().sum()
```

Out[7]:
```
id                        0
title                     0
type                      0
description               6
release_year              0
age_certification       325
runtime                   0
genre1                   28
genre2                  514
genre3                  774
genre4                 1070
genre5                 1289
genre6                 1438
production_countries      0
seasons                1120
imdb_id                 402
imdb_score              427
imdb_votes              430
tmdb_popularity          11
tmdb_score              109
dtype: int64
```

In [8]:
```
#retrieve the dimensions of a DataFrame:

df.shape
```

Out[8]:
```
(1535, 20)
```

In [9]:
```
#display a concise summary of the DataFrame's information:

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1535 entries, 0 to 1534
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    1535 non-null   object 
 1   title                 1535 non-null   object 
 2   type                  1535 non-null   object 
 3   description           1529 non-null   object 
 4   release_year          1535 non-null   int64  
 5   age_certification     1210 non-null   object 
 6   runtime               1535 non-null   int64  
 7   genre1                1507 non-null   object 
 8   genre2                1021 non-null   object 
 9   genre3                761 non-null    object 
 10  genre4                465 non-null    object 
 11  genre5                246 non-null    object 
 12  genre6                97 non-null     object 
 13  production_countries  1535 non-null   object 
 14  seasons               415 non-null    float64
 15  imdb_id               1133 non-null   object 
 16  imdb_score            1108 non-null   float64
 17  imdb_votes            1105 non-null   float64
 18  tmdb_popularity       1524 non-null   float64
 19  tmdb_score            1426 non-null   float64
dtypes: float64(5), int64(2), object(13)
memory usage: 240.0+ KB
```

In [10]:
```
#different types of content present:

content_type = df['type'].unique()
content_type
```

Out[10]:
```
array(['MOVIE', 'SHOW'], dtype=object)
```

In [ ]:
```

```

In [11]:
```
#calculate the count of unique content types present:

content_type_count = df['type'].nunique()
print("Count of Content Type : ", content_type_count)
```

```
Count of Content Type :  2
```

In [ ]:
```

```

In [12]:
```
df.describe()
```

Out[12]:

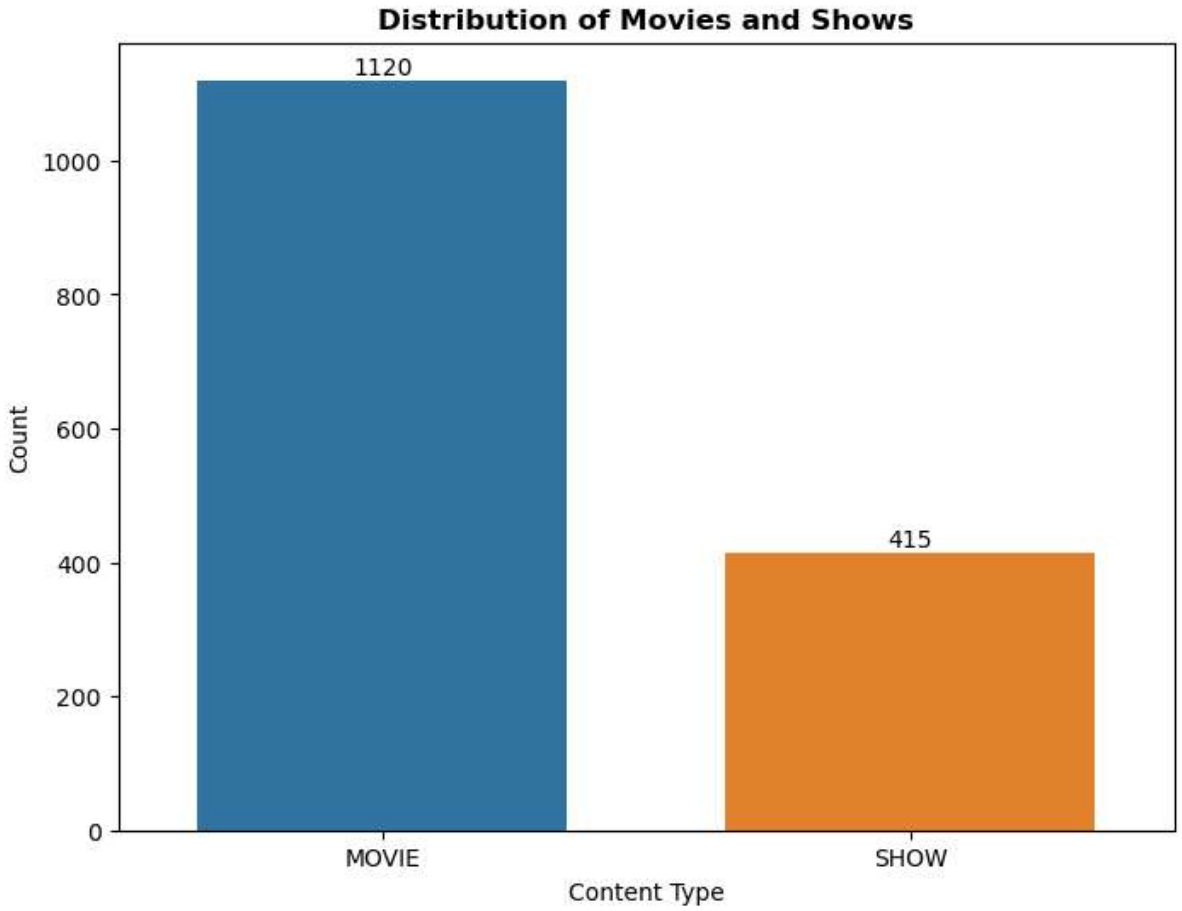
	release_year	runtime	seasons	imdb_score	imdb_votes	tmdb_popularity	tmdb_s
count	1535.000000	1535.000000	415.000000	1108.000000	1.105000e+03	1524.000000	1426.00
mean	2003.577850	60.163518	2.583133	6.605144	7.635004e+04	40.238486	6.91
std	21.678528	39.904824	3.013208	1.068985	1.885893e+05	263.149723	1.02
min	1928.000000	1.000000	1.000000	1.600000	5.000000e+00	0.600000	1.00
25%	1999.000000	23.000000	1.000000	5.900000	4.900000e+02	3.870750	6.30
50%	2012.000000	53.000000	2.000000	6.600000	4.219000e+03	9.383500	6.90
75%	2019.000000	93.000000	3.000000	7.400000	4.163900e+04	25.436750	7.50
max	2022.000000	182.000000	34.000000	9.700000	1.353907e+06	9323.832000	10.00

In [ ]:

In [13]:
```
Content_Count = df['type'].value_counts()
Content_Count
```

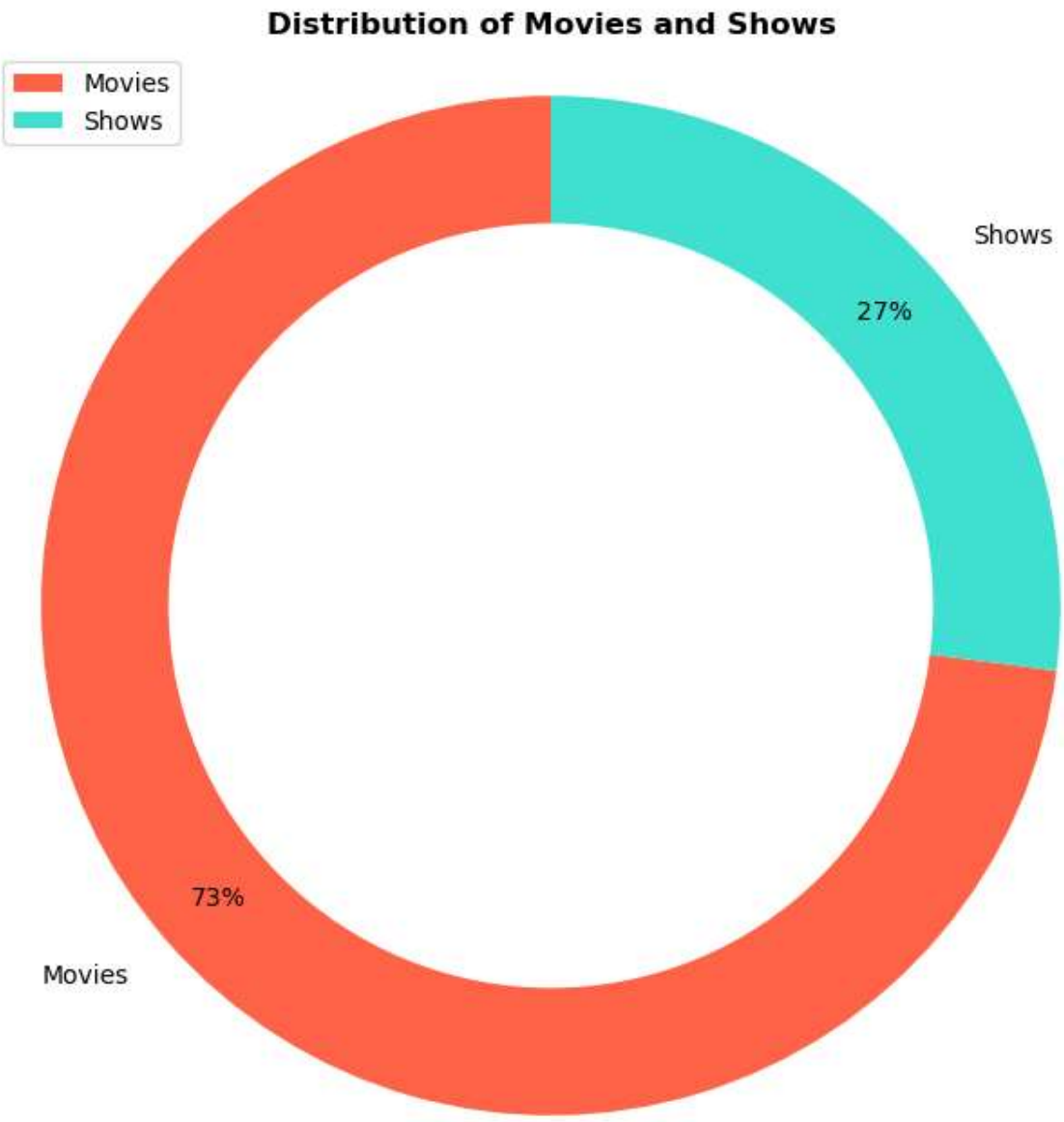
Out[13]:
```
MOVIE    1120
SHOW      415
Name: type, dtype: int64
```

In [14]:
```
# Set the figure size
plt.figure(figsize=(8, 6))

# Create a bar plot using Seaborn
ax = sns.barplot(x = Content_Count.index , y = Content_Count.values, width = 0.7)

for bars in ax.containers:
    ax.bar_label(bars)

# Set labels and title
plt.xlabel("Content Type")
plt.ylabel("Count")
plt.title("Distribution of Movies and Shows", fontweight = 'bold')

plt.show()
```



In [15]:
```
df['type'] = df['type'].map({"MOVIE" :"Movies", "SHOW" : "Shows"})

type = df['type'].value_counts()
labels = type.index
counts = type.values



fig, ax = plt.subplots(figsize=(8, 8))  # Adjust the width and height as needed

# Create a simple pie chart
ax.pie(counts, labels=labels, autopct="%0.0f%%", startangle=90, pctdistance=0.87, c

# Equal aspect ratio ensures that the pie is drawn as a circle.
ax.axis('equal')

# Add a legend
ax.legend(labels, loc="upper left")

# Add a title
plt.title("Distribution of Movies and Shows", fontweight = 'bold')

# Hole - a white-colored circle of radius 0.75
hole = plt.Circle((0, 0), 0.75, facecolor='white')

plt.gcf().gca().add_artist(hole)
```

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



In [ ]:
```
Conclusion:
    The quantity of movies surpasses that of TV shows on Disney+ Hotstar.
```

In [ ]:
```

```

In [16]:
```
#retrieves the unique values present in the 'release_year' column:

df['release_year'].unique()
```

Out[16]:
```
array([1940, 1937, 1942, 1950, 1949, 1941, 1944, 1947, 1928, 1948, 1938,
       1933, 1932, 1939, 1934, 1936, 1935, 1943, 1946, 1965, 1977, 1976,
       1964, 1979, 1955, 1969, 1954, 1973, 1980, 1951, 1960, 1970, 1961,
       1963, 1953, 1971, 1957, 1967, 1959, 1968, 1975, 1974, 1978, 1958,
       1952, 1962, 1972, 1966, 1956, 1987, 1989, 1990, 1988, 1985, 1984,
       1986, 1982, 1983, 1981, 1999, 1993, 2000, 1994, 1992, 1991, 1995,
       1998, 1997, 1996, 2003, 2009, 2004, 2006, 2010, 2008, 2007, 2005,
       2002, 2001, 2011, 2012, 2014, 2013, 2015, 2016, 2017, 2018, 2019,
       2020, 2022, 2021])
```

In [17]:
```
#calculates the count of occurrences of each unique release year :

release_year = df['release_year'].value_counts()
release_year
```

Out[17]:
```
2021    136
2020    126
2019     96
2017     67
2018     61
       ...
1943      1
1928      1
1970      1
1982      1
1958      1
Name: release_year, Length: 91, dtype: int64
```

In [ ]:
```

```

In [18]:
```
# Group by 'release_year' and 'type' and count the occurrences
type_count = df.groupby(['release_year', 'type']).size().reset_index(name='count')
type_count
```

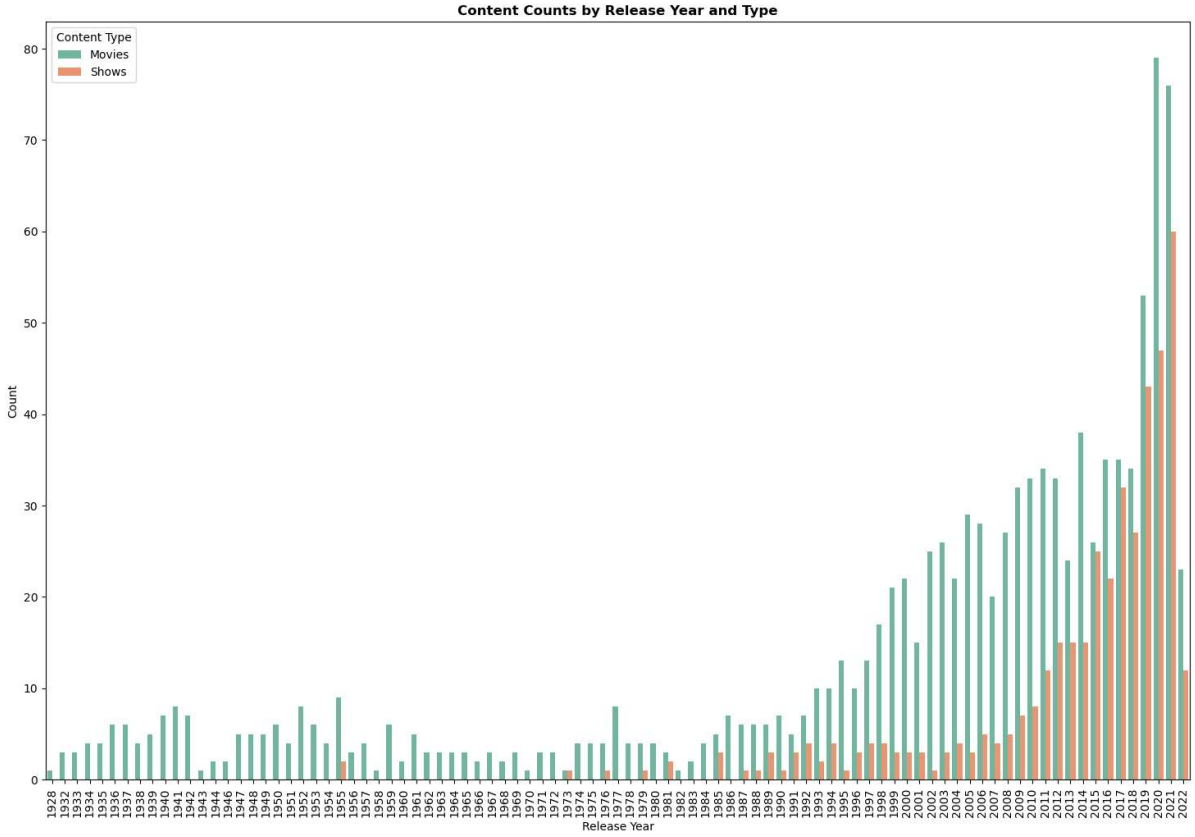
Out[18]:

	release_year	type	count
0	1928	Movies	1
1	1932	Movies	3
2	1933	Movies	3
3	1934	Movies	4
4	1935	Movies	4
...	...	...	...
128	2020	Shows	47
129	2021	Movies	76
130	2021	Shows	60
131	2022	Movies	23
132	2022	Shows	12

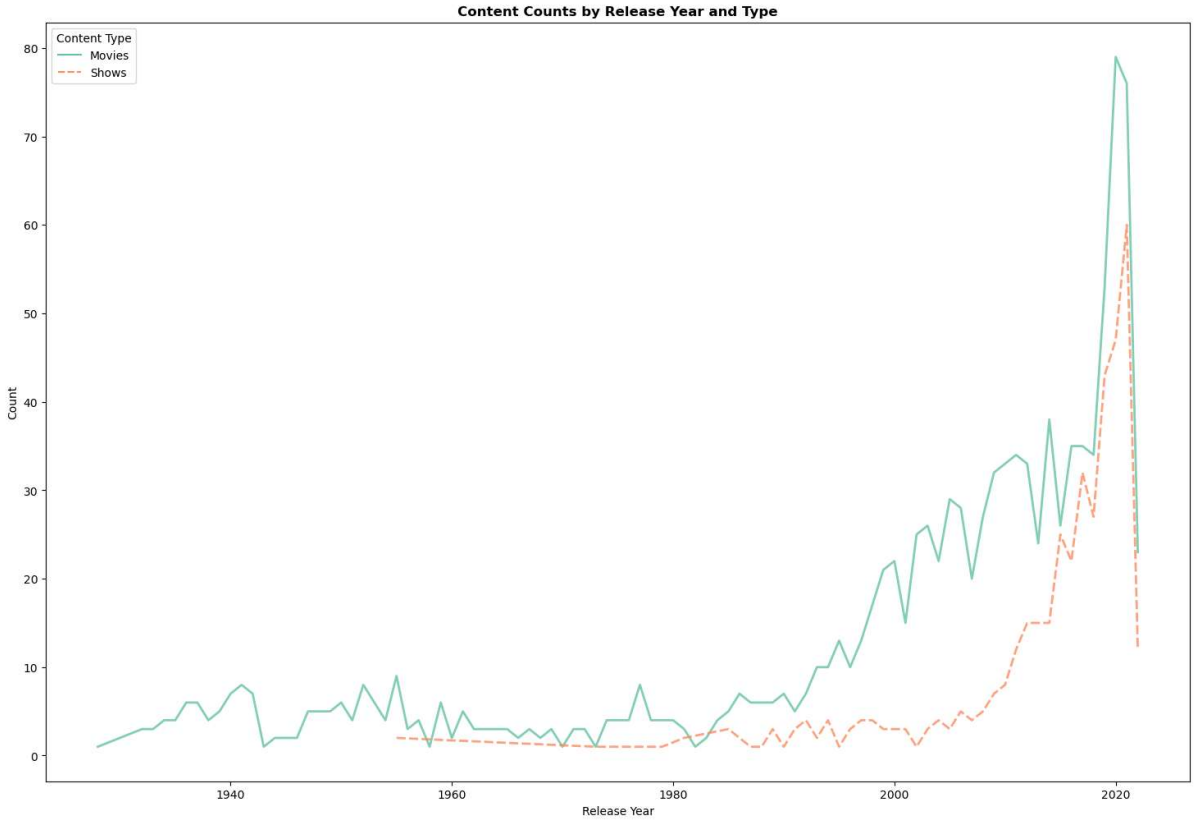
133 rows × 3 columns

In [19]:

```
# Create a bar plot using Seaborn
plt.figure(figsize=(18, 12))
sns.barplot(x='release_year', y='count', hue='type', data=type_count, palette="Set2

# Set labels and title
plt.xlabel("Release Year")
plt.ylabel("Count")
plt.title("Content Counts by Release Year and Type", fontweight= 'bold')

# Rotate and adjust fontsize of x-axis tick labels
plt.xticks(rotation = 90, fontsize=10)

# Add legend
plt.legend(title="Content Type", loc="upper left")

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



In [20]:
```
# Create an area plot using Seaborn
plt.figure(figsize=(18, 12))  # Set the figure size
pivot_type_count = type_count.pivot(index='release_year', columns='type', values='c
sns.lineplot(data=pivot_type_count, palette="Set2", linewidth=2, alpha=0.8)

# Set labels and title
plt.xlabel("Release Year")
plt.ylabel("Count")
plt.title("Content Counts by Release Year and Type", fontweight='bold')

# Add legend
plt.legend(title="Content Type")

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



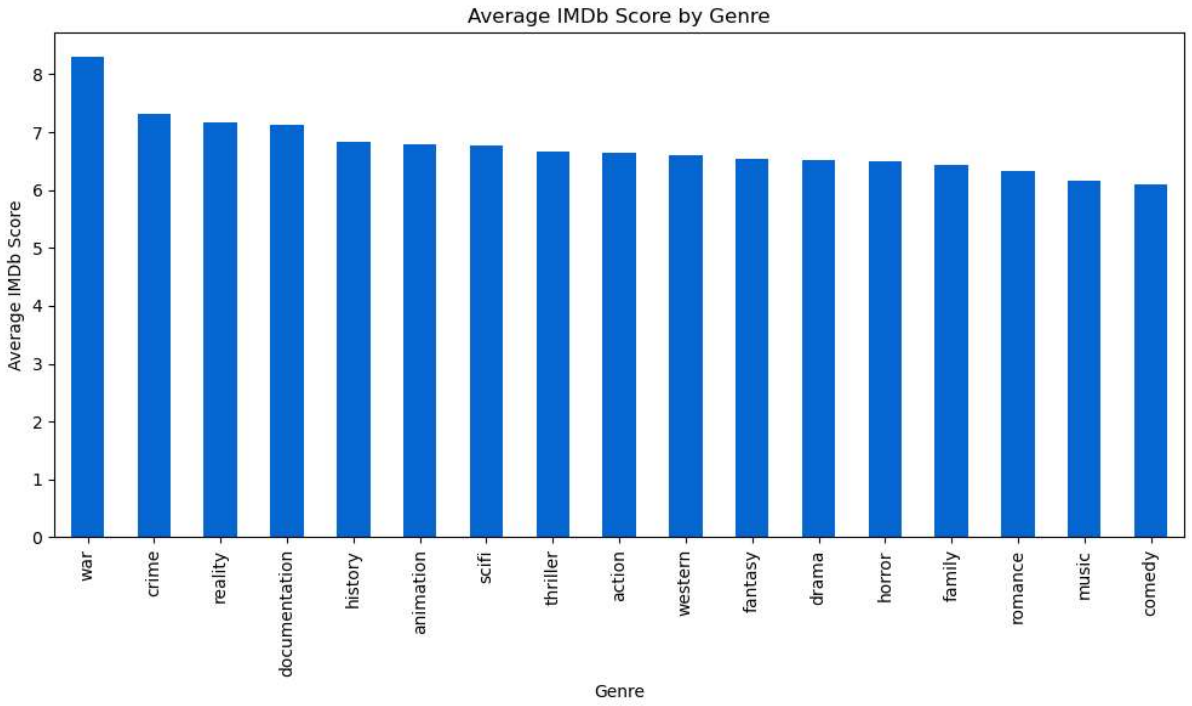
In [ ]:
```
Conclusion:
    Observations reveal that amidst the COVID-19 pandemic, the number of content re
```

In [ ]:
```

```

In [21]:
```
# Group data by genre and calculate average IMDb score for each genre
genre_avg_scores = df.groupby('genre1')['imdb_score'].mean().sort_values(ascending=
genre_avg_scores
```

Out[21]:
```
genre1
war              8.300000
crime            7.325000
reality          7.170833
documentation    7.131000
history          6.825000
animation        6.795876
scifi            6.777876
thriller         6.663636
action           6.635294
western          6.608333
fantasy          6.542187
drama            6.522222
horror           6.500000
family           6.431250
romance          6.319355
music            6.160000
comedy           6.087554
Name: imdb_score, dtype: float64
```

In [22]:
```
# Create a bar chart to visualize average IMDb scores by genre
plt.figure(figsize=(10, 6))

hotstar_colors = ['#0566D2', '#FF9900']

# Set the custom color palette using seaborn
sns.set_palette(hotstar_colors)

genre_avg_scores.plot(kind='bar')
plt.title('Average IMDb Score by Genre')
plt.xlabel('Genre')
plt.ylabel('Average IMDb Score')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



In [ ]:
```
Conclusion:
    The genre type 'War' boasts the highest average IMDb score, followed by 'Crime'
```

In [ ]:
```

```

In [57]:
```
# grouping the data by 'age_certification' and 'type' columns and calculates the co

grouped_data = df.groupby(['age_certification', 'type']).size().reset_index(name='c
grouped_data
```

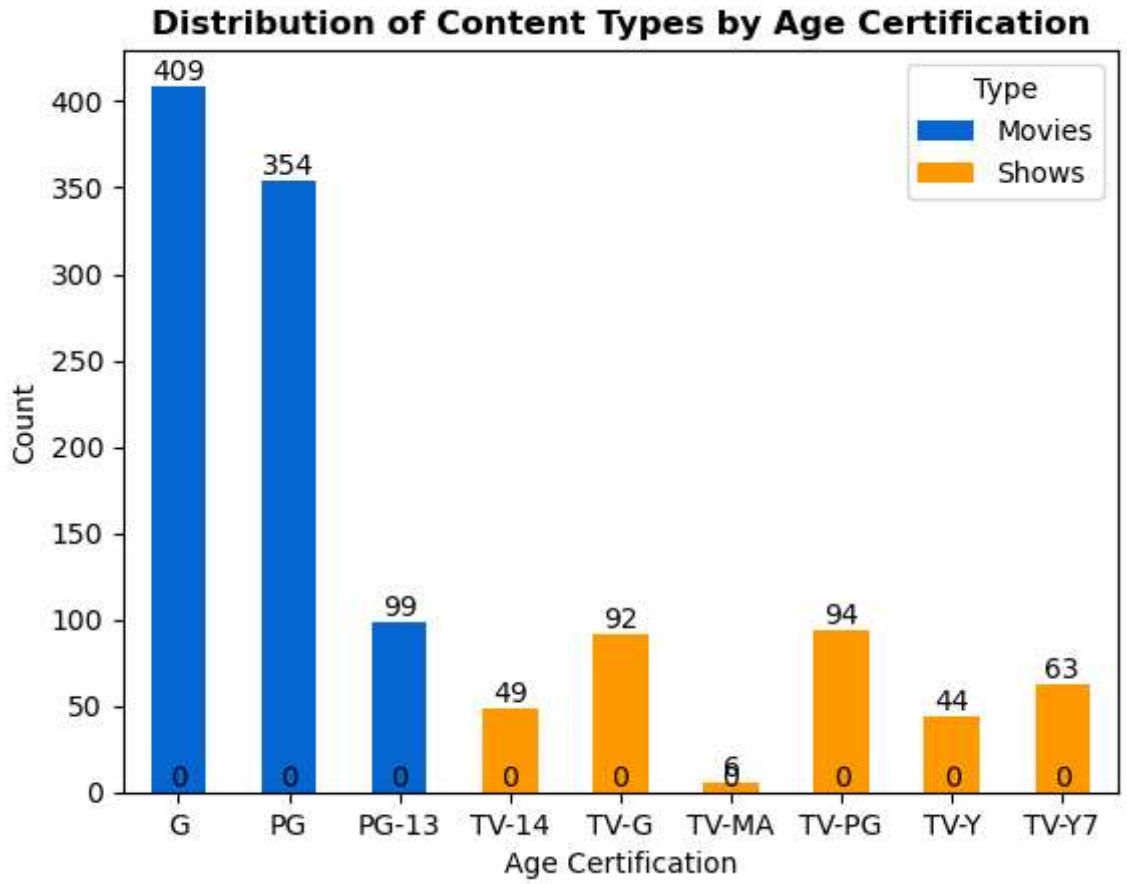
Out[57]:

	age_certification	type	count
0	G	Movies	409
1	PG	Movies	354
2	PG-13	Movies	99
3	TV-14	Shows	49
4	TV-G	Shows	92
5	TV-MA	Shows	6
6	TV-PG	Shows	94
7	TV-Y	Shows	44
8	TV-Y7	Shows	63

In [ ]:
```
G (General Audiences): This rating indicates that the film is suitable for all ages

PG (Parental Guidance Suggested): PG-rated films may contain material that parents

PG-13 Rating (Parents Strongly Cautioned): The "Parents Strongly Cautioned" rating
                                          may not be appropriate for children under

TV-Y Rating (All Children): This rating is meant for all children, regardless of ag

TV-Y7 Rating (Directed to Older Children): The "Directed to Older Children" rating
                                         children aged 7 and older.

TV-G Rating (General Audience): Similar to the G movie rating, TV-G signifies that

TV-14 Rating (Parents Strongly Cautioned): The "Parents Strongly Cautioned" rating
                                          inappropriate for children under 14 year

TV-MA Rating (Mature Audience): The "Mature Audience" rating is intended for viewer
```

In [24]:
```
plt.figure(figsize=(18, 12))

# Pivot the data for easy plotting
pivot_data = grouped_data.pivot(index='age_certification', columns='type', values='

# Plotting the grouped bar chart
ax = pivot_data.plot(kind='bar', stacked=True, color=['#0566D2', '#FF9900'])

for bars in ax.containers:
    ax.bar_label(bars)

plt.title('Distribution of Content Types by Age Certification', fontweight = 'bold'
plt.xlabel('Age Certification')
plt.ylabel('Count')
plt.legend(title='Type')
plt.xticks(rotation=0)

plt.show()
```

```
<Figure size 1800x1200 with 0 Axes>
```



In [ ]:
```
Conclusion:
    G and PG ratings are prevalent among movies, while TVG and TV-PG ratings are co
```

In [ ]:
```

```

In [25]:
```
#count the occurrences of different values in the 'production_countries':

production_count = df['production_countries'].value_counts()
production_count
```

Out[25]:

USA	1292
Great Britain	60
[]	47
Canada	36
['FR']	10
Germany	7
France	6
Australia	6
['AU']	5
India	4
['AR']	4
Japan	3
['PH']	2
['IN']	2
['US', 'CN']	2
['MX', 'US']	2
Newzland	2
['NZ']	2
['JP', 'US']	2
['JP']	2
Brazil	2
['DE', 'US', 'FR', 'GB', 'ZA']	1
['GR']	1
['CL']	1
['DK', 'US', 'CA']	1
['FR', 'GB']	1
['BR', 'AR']	1
['JP', 'US', 'CA', 'CN', 'FR']	1
['US', 'KI']	1
['BW']	1
['ZA', 'US']	1
['NZ', 'GB', 'US']	1
['KR']	1
['US', 'CA', 'DK']	1
['MX', 'US', 'JP']	1
['ES', 'US']	1
['US', 'AU', 'GB']	1
['CN', 'GB', 'US']	1
['ES', 'CH', 'FR', 'MC']	1
['SE', 'US', 'NO']	1
['US', 'AT']	1
Scotland	1
Soviet Union	1
['PL', 'SI', 'US', 'CZ']	1
['SG', 'US']	1
['GB', 'US', 'HU']	1
['GB', 'US', 'DE', 'IE']	1
['XX']	1
['US', 'FR']	1
Hong Kong	1
['NL']	1
['FR', 'PT']	1
['US', 'FR', 'KR']	1
['BE', 'GB', 'US']	1
['GB', 'DE', 'US']	1
['AU', 'GB', 'US']	1

```
['IT']                                1
['ZA']                                1
Name: production_countries, dtype: int64
```

In [ ]:
```
Conclusion:
    The United States takes the lead with a significant count of 1278 in crafting D
    followed closely by Great Britain securing the second position.
```

In [ ]:
```

```

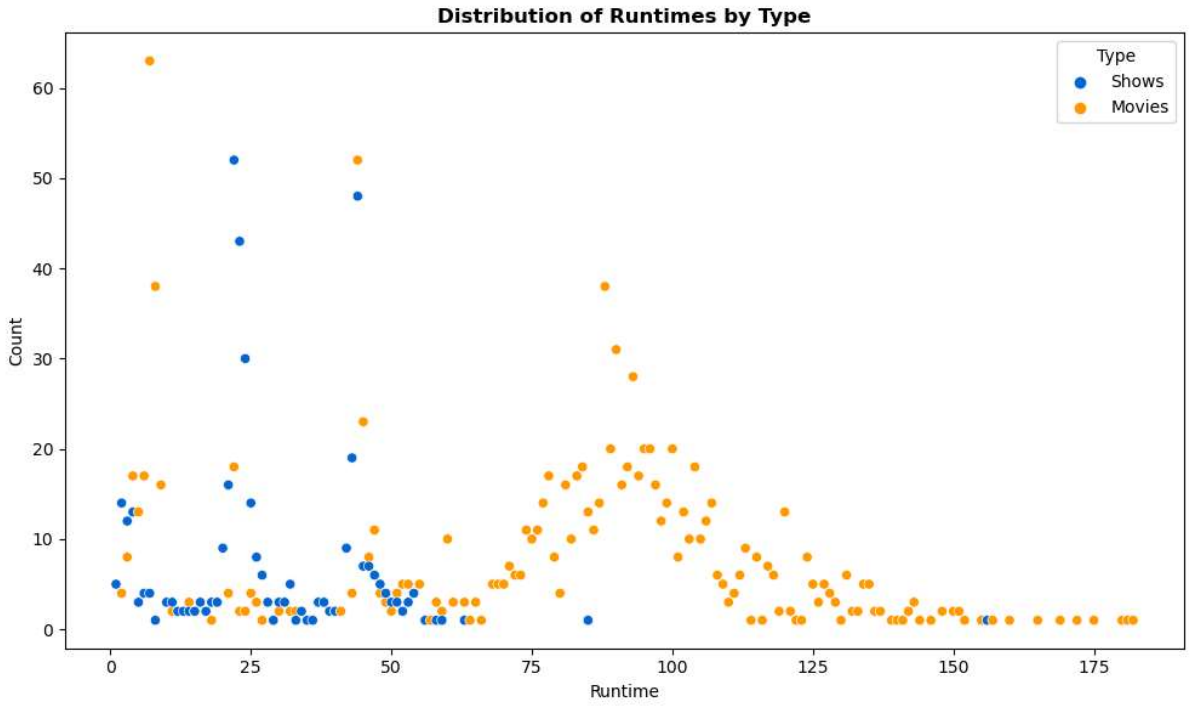
In [35]:
```
# Group by 'runtime' and 'type', and calculate the count for each group
grouped_data = df.groupby(['runtime', 'type']).size().reset_index(name='count')
grouped_data
```

Out[35]:

	runtime	type	count
0	1	Shows	5
1	2	Movies	4
2	2	Shows	14
3	3	Movies	8
4	3	Shows	12
...	...	...	...
197	172	Movies	1
198	175	Movies	1
199	180	Movies	1
200	181	Movies	1
201	182	Movies	1

202 rows × 3 columns

In [36]:
```
#Remove rows with missing runtime values
grouped_data = grouped_data.dropna(subset=['runtime'])

# Create a scatter plot to visualize the distribution of runtimes
plt.figure(figsize=(10, 6))
sns.scatterplot(data=grouped_data, x='runtime', y='count', hue='type', palette=['#0
plt.title('Distribution of Runtimes by Type', fontweight = 'bold')
plt.xlabel('Runtime')
plt.ylabel('Count')
plt.legend(title='Type')
plt.tight_layout()
plt.show()
```



In [ ]:
```
Conclusion:
    Predominantly, movies on Disney+ are typically released with a runtime ranging
```

In [ ]:
```

```

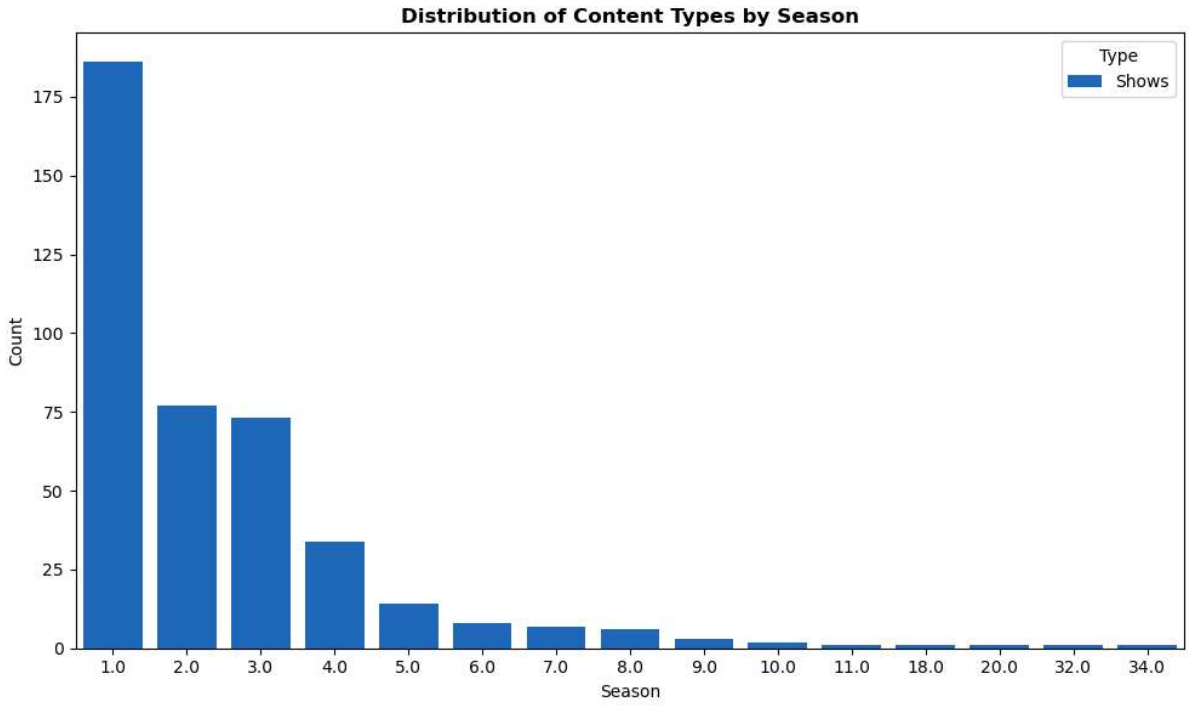
In [38]:
```
# Group by 'type' and 'season', and calculate the count for each group
grouped_data = df.groupby(['type', 'seasons']).size().reset_index(name='count')
grouped_data
```

Out[38]:

	type	seasons	count
0	Shows	1.0	186
1	Shows	2.0	77
2	Shows	3.0	73
3	Shows	4.0	34
4	Shows	5.0	14
5	Shows	6.0	8
6	Shows	7.0	7
7	Shows	8.0	6
8	Shows	9.0	3
9	Shows	10.0	2
10	Shows	11.0	1
11	Shows	18.0	1
12	Shows	20.0	1
13	Shows	32.0	1
14	Shows	34.0	1

In [44]:

```
# Create a bar plot to visualize the distribution of content types by season
plt.figure(figsize=(10, 6))
sns.barplot(data=grouped_data, x='seasons', y='count', hue='type', palette=['#0566D
plt.title('Distribution of Content Types by Season', fontweight = 'bold')
plt.xlabel('Season')
plt.ylabel('Count')
plt.legend(title='Type')
plt.tight_layout()
plt.show()
```



In [ ]:
```
Conclusion:
    Many TV shows on Disney+ extend beyond 10 seasons, often spanning 20 or even 30
```

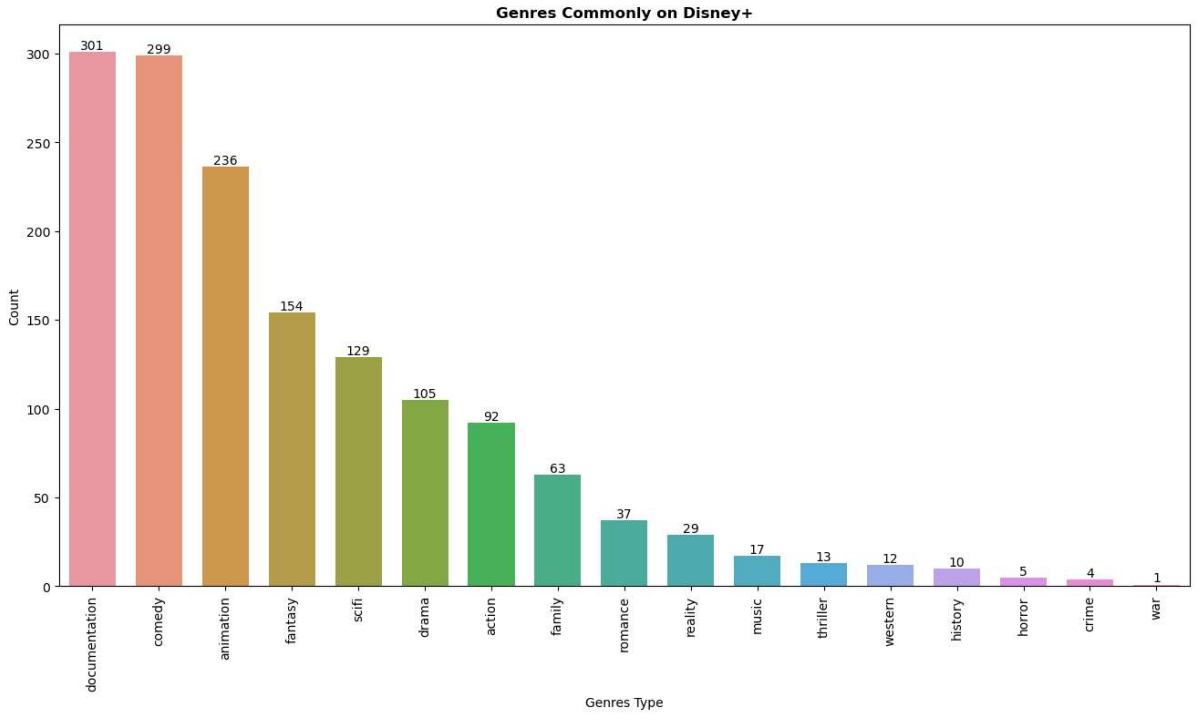
In [ ]:
```
```

In [48]:
```
genre_counts = df['genre1'].value_counts()
genre_counts
```

Out[48]:
```
documentation    301
comedy           299
animation        236
fantasy          154
scifi            129
drama            105
action            92
family            63
romance           37
reality           29
music             17
thriller          13
western           12
history           10
horror             5
crime              4
war                1
Name: genre1, dtype: int64
```

In [52]:
```
# Set the figure size
plt.figure(figsize=(16,8))

# Create a bar plot using Seaborn
ax = sns.barplot(x = genre_counts.index , y = genre_counts.values, width = 0.7)
```

```
for bars in ax.containers:
    ax.bar_label(bars)

# Set labels and title
plt.xlabel("Genres Type")
plt.ylabel("Count")
plt.title("Genres Commonly on Disney+", fontweight = 'bold')
plt.xticks(rotation = 90)

plt.show()
```



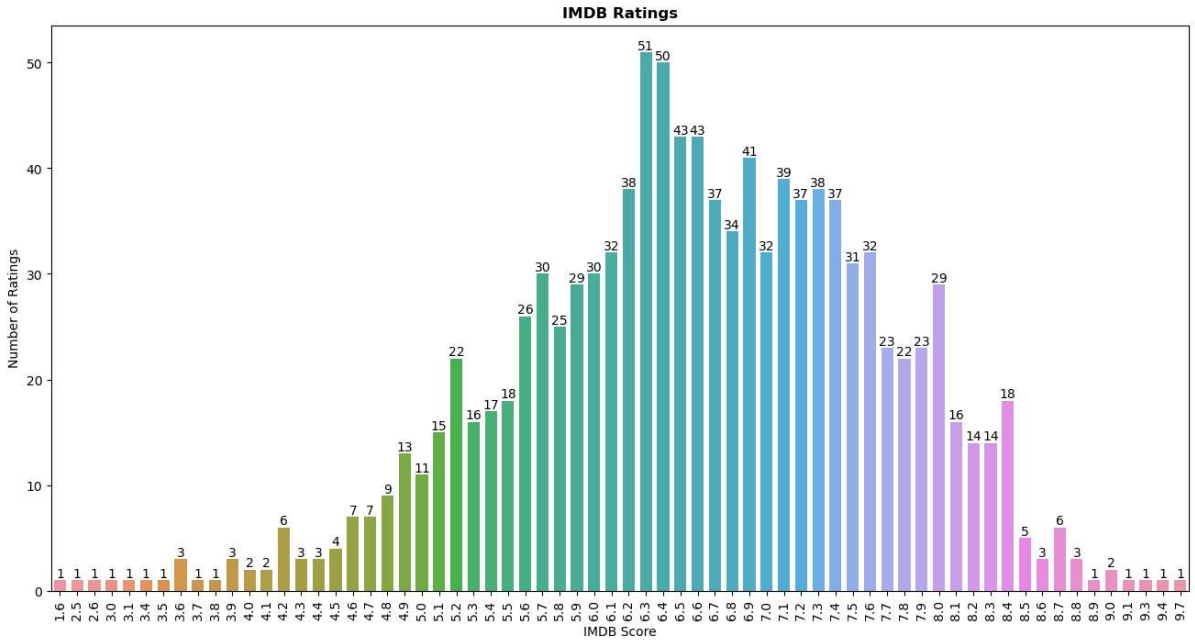
In [ ]:
```
Conclusion:
    Across both movies and TV shows, the most prominent genres are documentation, c
```

In [ ]:
```
```

In [55]:
```
#calculates the count of occurrences for each unique IMDb score present in the 'imd

imdb_count = df['imdb_score'].value_counts()
imdb_count
```

Out[55]:
```
6.3    51
6.4    50
6.6    43
6.5    43
6.9    41
       ..
3.5     1
1.6     1
2.6     1
3.0     1
8.9     1
Name: imdb_score, Length: 66, dtype: int64
```

In [56]:
```
# Set the figure size
plt.figure(figsize=(16,8))

# Create a bar plot using Seaborn
ax = sns.barplot(x = imdb_count.index , y = imdb_count.values, width = 0.7)

for bars in ax.containers:
    ax.bar_label(bars)

# Set labels and title
plt.xlabel("IMDB Score")
plt.ylabel("Number of Ratings")
plt.title("IMDB Ratings", fontweight = 'bold')
plt.xticks(rotation = 90)

plt.show()
```



In [ ]:
```
Conclusion:
    Most of the IMDb ratings for TV shows and movies lie within the range of 6 to 8
```

In [ ]:
```
```

In [ ]:
```
Insights:
    >>The number of films dominate significantly on this platform this may be due t

    >>In recent 10 years the number of films and tv shows have increased considerab
      mobiles, computers and the popularity of the media content rose.

    >>Disney plus hotstar should also focus on other age groups like pg-13 tv-14 an

Dataset Link : https://drive.google.com/drive/folders/1jAWnQYGi4MTO90F5JwHgzBDlyfc6
```