In [1]:
```
# !pip install polars pandas plotly pyarrow nbformat
```

In [2]:
```
import polars as pl
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
```

In [3]:
```
dataset = pl.read_csv(
    './data/imdb.tsv',
    separator='\t',
    null_values="\\N",
    quote_char=None
)

rating_dataset = pl.read_csv(
    './data/imdb_rating.tsv',
    separator='\t',
    null_values="\\N",
    quote_char=None
)
```

In [4]:
```
# theme = 'plotly_dark'
theme = 'plotly_white'
```

In [5]:
```
rating_preference = 2
```

In [6]:
```
df = dataset.join(rating_dataset, on='tconst', how='inner')
df
```

Out[6]: shape: (1_306_106, 11)

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	averageRating	n
str	str	str	str	i64	i64	str	i64	str	f64	
"tt0000001"	"short"	"Carmencita"	"Carmencita"	0	1894	null	1	"Documentary,Short"	5.7	
"tt0000002"	"short"	"Le clown et ses chiens"	"Le clown et ses chiens"	0	1892	null	5	"Animation,Short"	5.8	
"tt0000003"	"short"	"Pauvre Pierrot"	"Pauvre Pierrot"	0	1892	null	4	"Animation,Comedy,Romance"	6.5	
"tt0000004"	"short"	"Un bon bock"	"Un bon bock"	0	1892	null	12	"Animation,Short"	5.6	
"tt0000005"	"short"	"Blacksmith Scene"	"Blacksmith Scene"	0	1893	null	1	"Comedy,Short"	6.2	
…	…	…	…	…	…	…	…	…	…	
"tt9916730"	"movie"	"6 Gunn"	"6 Gunn"	0	2017	null	116	null	8.3	
"tt9916766"	"tvEpisode"	"Episode #10.15"	"Episode #10.15"	0	2019	null	43	"Family,Game-Show,Reality-TV"	7.0	
"tt9916778"	"tvEpisode"	"Escape"	"Escape"	0	2019	null	null	"Crime,Drama,Mystery"	7.2	
"tt9916840"	"tvEpisode"	"Horrid Henry's Comic Caper"	"Horrid Henry's Comic Caper"	0	2014	null	11	"Adventure,Animation,Comedy"	8.8	
"tt9916880"	"tvEpisode"	"Horrid Henry Knows It All"	"Horrid Henry Knows It All"	0	2014	null	10	"Adventure,Animation,Comedy"	8.2	

In [7]:
```
df.schema
```

Out[7]:
```
Schema([('tconst', String),
        ('titleType', String),
        ('primaryTitle', String),
        ('originalTitle', String),
        ('isAdult', Int64),
        ('startYear', Int64),
        ('endYear', String),
        ('runtimeMinutes', Int64),
        ('genres', String),
        ('averageRating', Float64),
        ('numVotes', Int64)])
```

Type Frequency

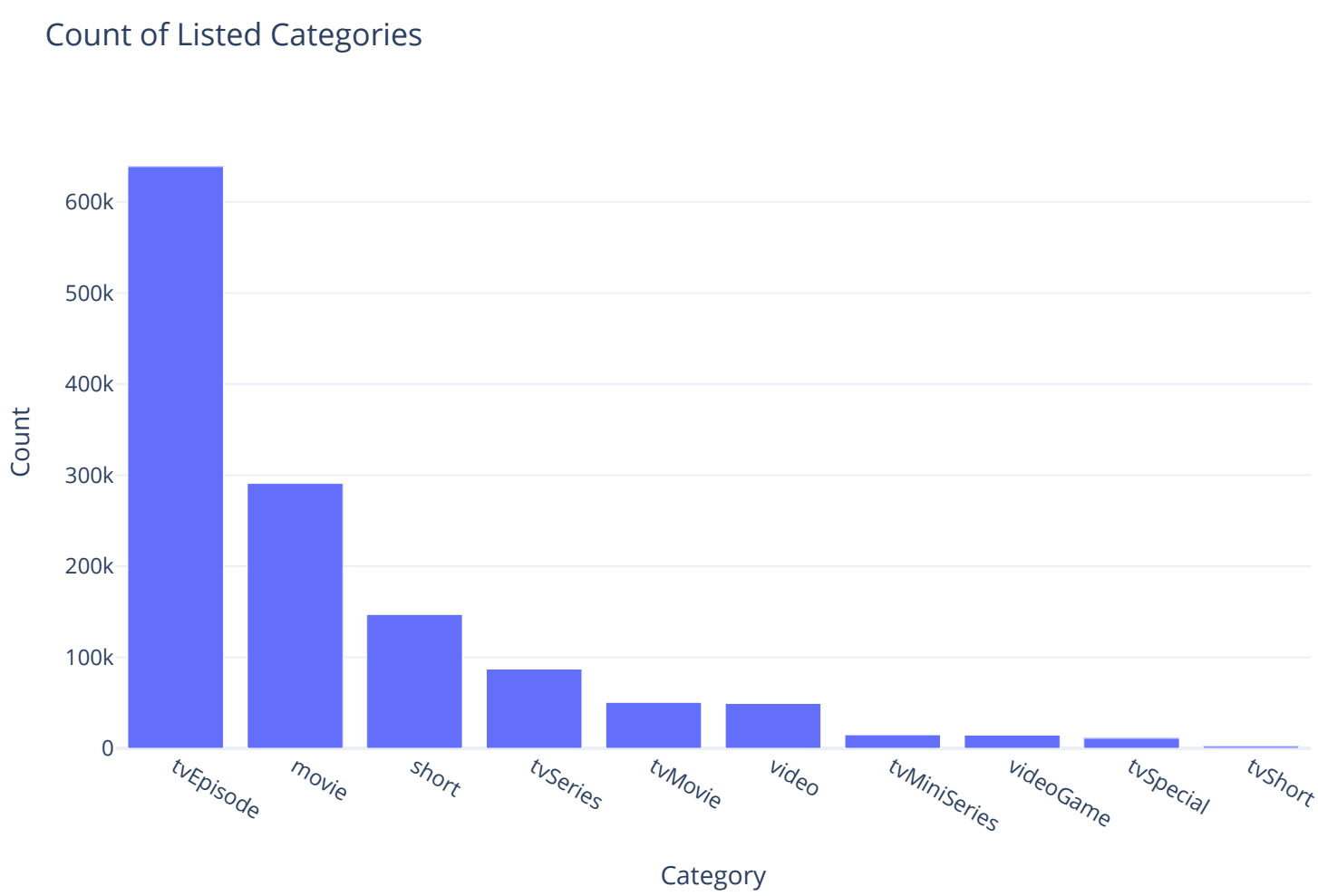
In [8]:
```
# Assuming df is your DataFrame
type_count = (
    df.lazy()
    .group_by('titleType')
    .agg([
```

```
        pl.len().alias('count')  # Only include the count aggregation
    ]).sort("count", descending=True)
)

# Collecting the result
type_count.collect()
```

Out[8]: shape: (10, 2)

titleType	count
str	u32
"tvEpisode"	638904
"movie"	290903
"short"	146929
"tvSeries"	87044
"tvMovie"	50410
"video"	49288
"tvMiniSeries"	14842
"videoGame"	14528
"tvSpecial"	11069
"tvShort"	2189

In [9]:
```
type_count_in_df = type_count.collect().to_pandas()

# Create a bar chart using Plotly
fig = px.bar(type_count_in_df, x="titleType", y="count", title="Count of Listed Categories", labels={"titleType": "

fig.update_layout(template=theme)
fig.show(renderer='notebook')
```



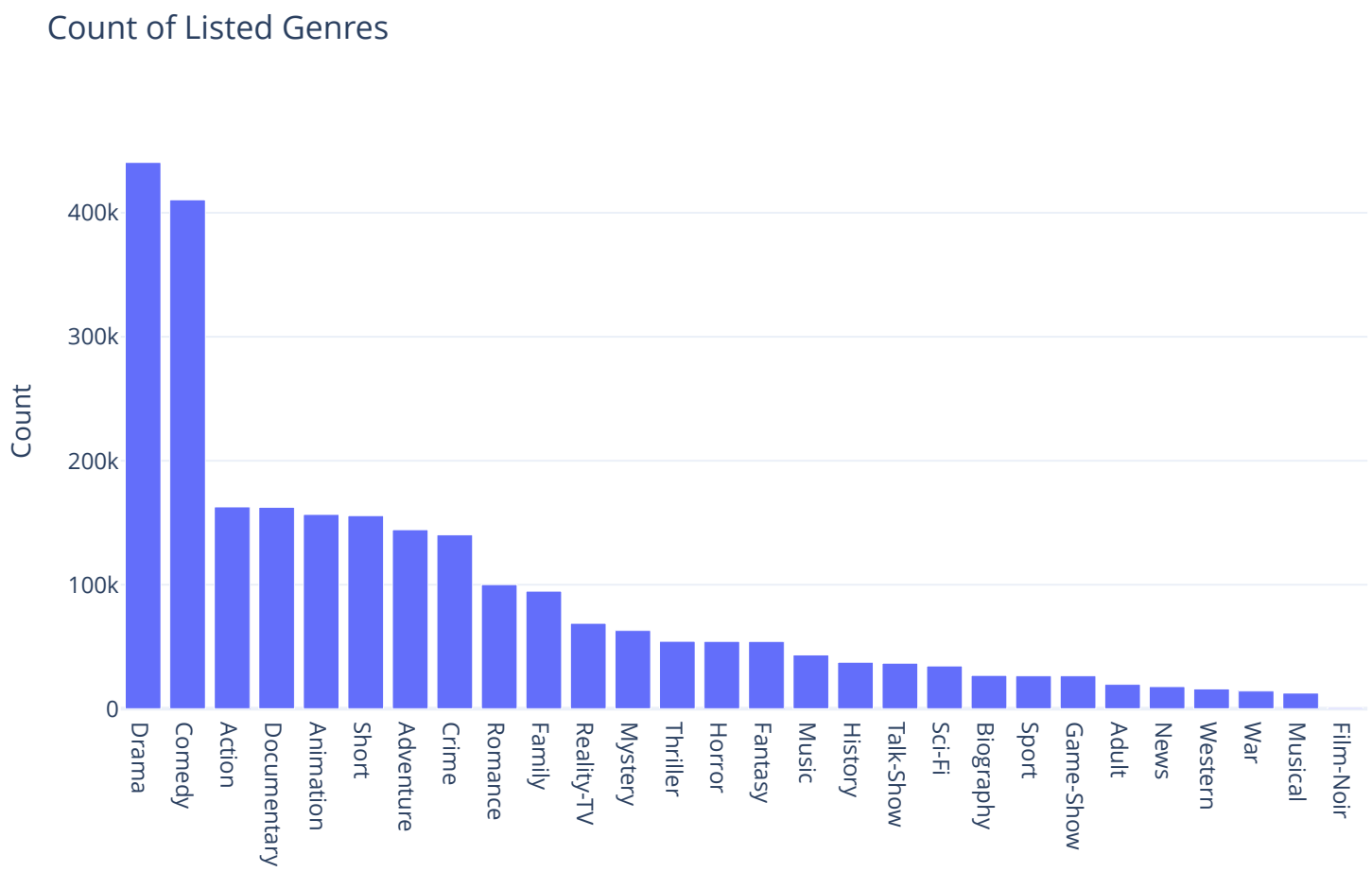
Genres Frequency

In [10]:
```
genres_count = (
    df.lazy()
    .select(pl.col("genres").str.split(","))
    .explode("genres")
    .group_by("genres")
    .agg([pl.len().alias("count")])
    .sort("count", descending=True)
)

genres_count.collect()
```

Out[10]: shape: (29, 2)

genres	count
str	u32
"Drama"	440604
"Comedy"	410433
"Action"	162759
"Documentary"	162438
"Animation"	156700
...	...
"News"	17817
"Western"	15948
"War"	14247
"Musical"	12595
"Film-Noir"	882

In [11]:
```
genres_count_in_df = genres_count.collect().to_pandas()

# Create a bar chart using Plotly
fig = px.bar(genres_count_in_df, x="genres", y="count", title="Count of Listed Genres", labels={"genres": "Genres",
# Apply dark theme
fig.update_layout(template=theme)
fig.show(renderer='notebook')
```



Top Few Titles

In [12]:
```
top_few_movies = (
    df.lazy()
    .select(["primaryTitle", "titleType", "averageRating", "numVotes", (pl.col('averageRating') ** rating_preferenc
    .sort("weightedRating", descending=True)
    .limit(25)
)

top_few_movies.collect()
```

Out[12]: shape: (25, 5)

primaryTitle	titleType	averageRating	numVotes	weightedRating
str	str	f64	i64	f64
"The Shawshank Redemption"	"movie"	9.3	2732446	236329.25454
"The Dark Knight"	"movie"	9.0	2705636	219156.516
"Inception"	"movie"	8.8	2401515	185973.3216
"Game of Thrones"	"tvSeries"	9.2	2154178	182329.62592
"Breaking Bad"	"tvSeries"	9.5	1964594	177304.6085
…	…	…	…	…
"The Godfather Part II"	"movie"	9.0	1295390	104926.59
"Saving Private Ryan"	"movie"	8.6	1417607	104846.21372
"Star Wars: Episode IV - A New …	"movie"	8.6	1387837	102644.42452
"Inglourious Basterds"	"movie"	8.3	1484315	102254.46035
"Batman Begins"	"movie"	8.2	1497020	100659.6248

Top Few Titles by Type

In [13]:
```
top_few_movies_by_types = (
    df.lazy()
    .filter(pl.col("titleType") == "tvSeries")
    .select(["primaryTitle", "titleType", "averageRating", "numVotes", (pl.col('averageRating') ** rating_preferenc
    .sort("weightedRating", descending=True)
    .limit(25)
)

top_few_movies_by_types.collect()
```

Out[13]: shape: (25, 5)

primaryTitle	titleType	averageRating	numVotes	weightedRating
str	str	f64	i64	f64
"Game of Thrones"	"tvSeries"	9.2	2154178	182329.62592
"Breaking Bad"	"tvSeries"	9.5	1964594	177304.6085
"Stranger Things"	"tvSeries"	8.7	1236064	93557.68416
"Friends"	"tvSeries"	8.9	1022109	80961.25389
"Sherlock"	"tvSeries"	9.1	947111	78430.26191
…	…	…	…	…
"Prison Break"	"tvSeries"	8.3	549661	37866.14629
"Westworld"	"tvSeries"	8.5	514348	37161.643
"House"	"tvSeries"	8.7	476875	36094.66875
"The Sopranos"	"tvSeries"	9.2	419973	35546.51472
"Narcos"	"tvSeries"	8.8	439323	34021.17312

Top Few Titles by Genres

In [14]:
```
# top few titles by genres

top_few_movies_by_genres = (
    df.lazy()
    .filter(pl.col("genres").str.contains("Crime"))
    .select(["primaryTitle", "titleType", "averageRating", "numVotes", (pl.col('averageRating') ** rating_preferenc
    .sort("weightedRating", descending=True)
    .limit(25)
)

top_few_movies_by_genres.collect()
```

Out[14]: shape: (25, 5)

primaryTitle	titleType	averageRating	numVotes	weightedRating
str	str	f64	i64	f64
"The Dark Knight"	"movie"	9.0	2705636	219156.516
"Breaking Bad"	"tvSeries"	9.5	1964594	177304.6085
"Pulp Fiction"	"movie"	8.9	2099696	166316.92016
"The Godfather"	"movie"	9.2	1899931	160810.15984
"Se7en"	"movie"	8.6	1689151	124929.60796
…	…	…	…	…
"Catch Me If You Can"	"movie"	8.1	1014752	66577.87872
"12 Angry Men"	"movie"	9.0	808525	65490.525
"Scarface"	"movie"	8.3	861605	59355.96845
"Snatch"	"movie"	8.2	870931	58561.40044
"A Clockwork Orange"	"movie"	8.3	844868	58202.95652

Genres vs Rating

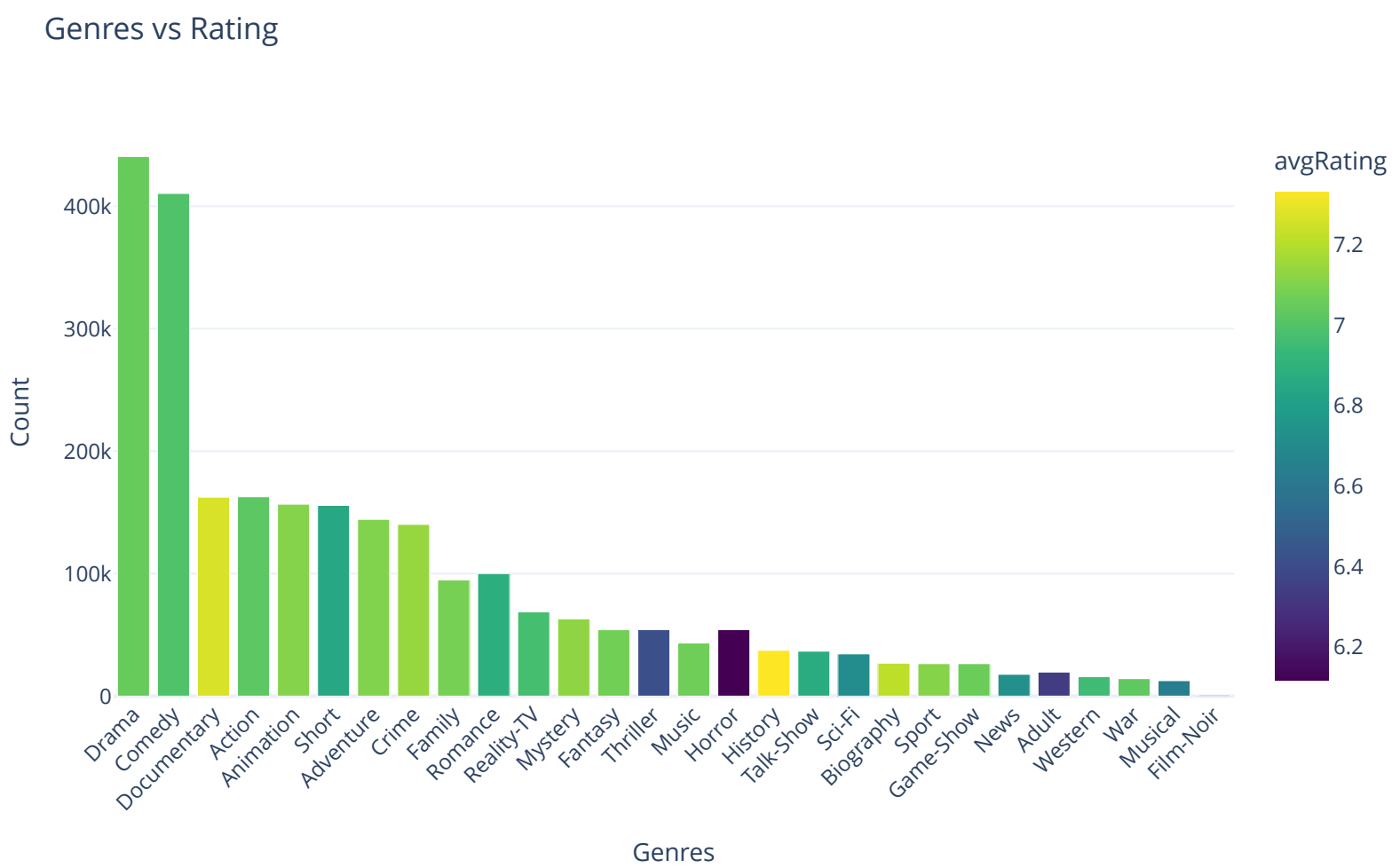
In [15]:
```
genres_avg_rating = (
    df.lazy()
    .select([pl.col("genres"), pl.col("averageRating")])
    .with_columns([pl.col("genres").str.split(",").alias("genres")])
    .explode("genres")
    .group_by("genres")
    .agg([
        pl.col("averageRating").mean().alias("avgRating"),
        pl.len().alias("count"),
    ])
    .with_columns([
        (pl.col("avgRating") ** rating_preference * pl.col("count") / 1000).alias("weightedRating")
    ])
    .sort("weightedRating", descending=True)
)

genres_avg_rating.collect()
```

Out[15]: shape: (29, 4)

genres	avgRating	count	weightedRating
str	f64	u32	f64
"Drama"	7.047944	440604	21886.351802
"Comedy"	6.993148	410433	20071.866867
"Documentary"	7.264498	162438	8572.329017
"Action"	7.020403	162759	8021.748551
"Animation"	7.106294	156700	7913.257247
…	…	…	…
"Adult"	6.338054	19635	788.756304
"Western"	6.953399	15948	771.081834
"War"	7.029726	14247	704.044589
"Musical"	6.634538	12595	554.395224
"Film-Noir"	6.467574	882	36.893627

In [16]:
```
fig = px.bar(
    genres_avg_rating.collect(),
    x="genres",
    y="count",
    title="Genres vs Rating",
    labels={"genres": "Genres", "count": "Count"},
    color="avgRating",  # Color by avgRating
    color_continuous_scale="Viridis"  # Choose color scale
)

# Customize layout for better aesthetics
fig.update_layout(
    xaxis_title="Genres",
    yaxis_title="Count",
    xaxis_tickangle=-45,  # Rotate x-axis labels for readability
    template=theme  # Dark mode
)
```

```
fig.show(renderer='notebook')
```



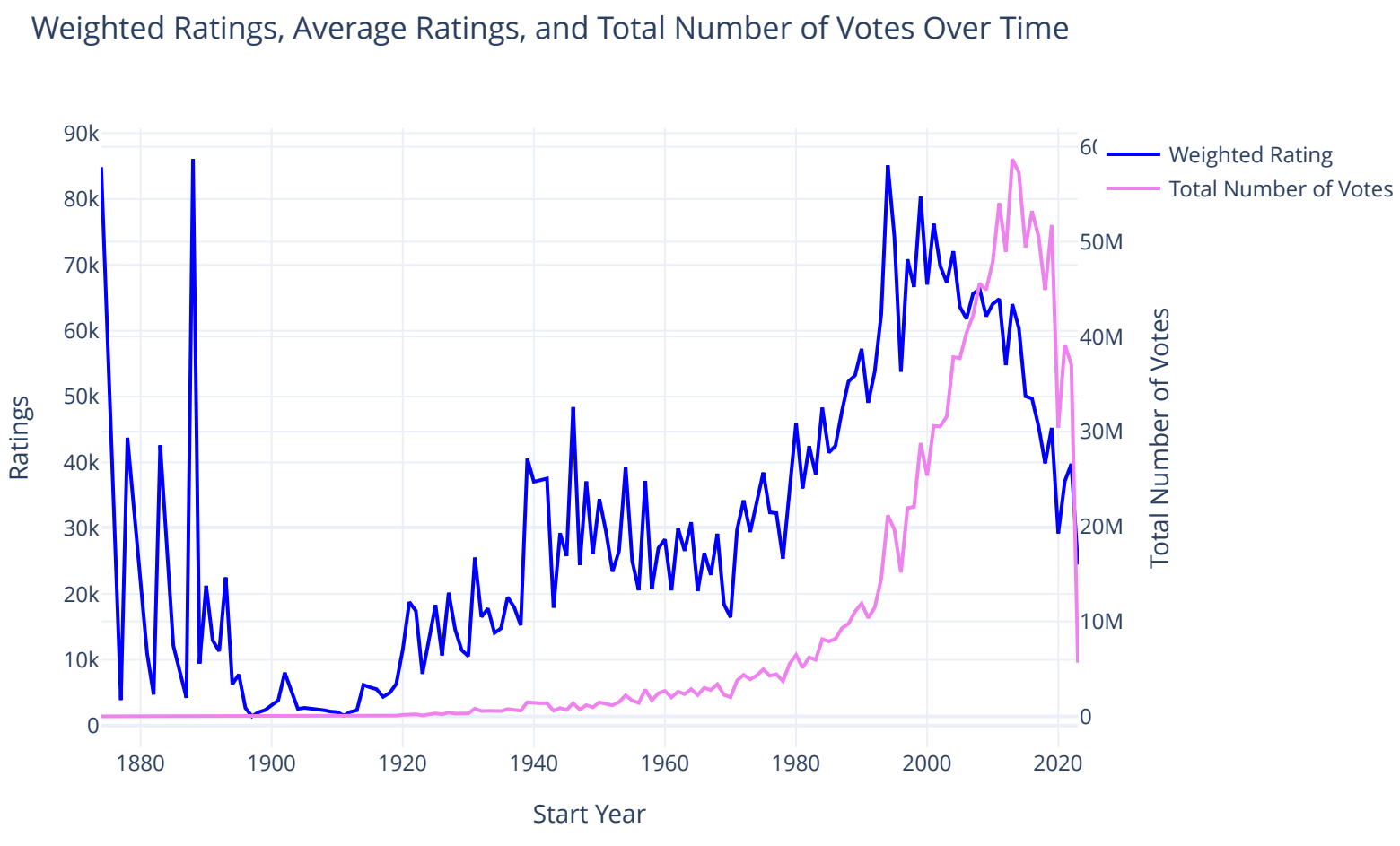
In [17]:
```
rating_time = (
    df.lazy()
    .filter(pl.col("startYear").cast(pl.Int64).is_not_null())  # Filter out null values in startYear
    .with_columns([
        pl.col("startYear").cast(pl.Int64),  # Ensure startYear is an integer
    ])
    .group_by("startYear")
    .agg([
        pl.len().alias("count"),
        pl.col("averageRating").mean().alias("avgRating"),
        pl.col("numVotes").sum().alias("totalNumVotes"),
    ])
    .with_columns([
        (pl.col("avgRating") ** rating_preference * pl.col("totalNumVotes") / pl.col("count")).alias("weightedRatin
    ])
    .sort("startYear", descending=False)  # Sort by startYear for time-based analysis
)

rating_time.collect()
```

Out[17]: shape: (144, 5)

startYear	count	avgRating	totalNumVotes	weightedRating
i64	u32	f64	i64	f64
1874	1	6.8	1835	84850.4
1877	4	5.6	496	3888.64
1878	3	6.033333	3605	43742.001852
1881	2	5.6	698	10944.64
1882	2	5.6	301	4719.68
…	…	…	…	…
2019	58262	7.135761	51750908	45228.6017
2020	52634	7.109942	30413985	29210.507123
2021	53688	7.134991	39149477	37122.362334
2022	49502	7.287984	37050760	39754.769032
2023	12656	7.399802	5659167	24484.78512

In [18]:
```
rating_time_df = rating_time.collect().to_pandas()

# Create the figure and add multiple traces
fig = go.Figure()

# Add line for Weighted Rating
fig.add_trace(go.Scatter(
    x=rating_time_df["startYear"],
```

```
    y=rating_time_df["weightedRating"],
    mode="lines",
    name="Weighted Rating",
    line=dict(color="blue")
))

# Add line for Total Number of Votes (scaled for visualization if needed)
fig.add_trace(go.Scatter(
    x=rating_time_df["startYear"],
    y=rating_time_df["totalNumVotes"],
    mode="lines",
    name="Total Number of Votes",
    line=dict(color="violet"),
    yaxis="y2"  # Using a secondary y-axis for totalNumVotes if needed for scale
))

# Set the layout
fig.update_layout(
    title="Weighted Ratings, Average Ratings, and Total Number of Votes Over Time",
    xaxis_title="Start Year",
    yaxis_title="Ratings",
    yaxis2=dict(title="Total Number of Votes", overlaying="y", side="right"),
    template=theme
)

# Show the plot
fig.show(renderer='notebook')
```



In [19]:

```
df_genres = (
    df.lazy()
    .filter(pl.col("genres").is_not_null())
    .with_columns([pl.col("genres").str.split(",").alias("genre_list")])
    .filter(pl.col("genre_list").len() > 1)
)

df_genres.collect()
```

Out[19]: shape: (1_286_401, 12)

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	averageRating	n
str	str	str	str	i64	i64	str	i64	str	f64	
"tt0000001"	"short"	"Carmencita"	"Carmencita"	0	1894	null	1	"Documentary,Short"	5.7	
"tt0000002"	"short"	"Le clown et ses chiens"	"Le clown et ses chiens"	0	1892	null	5	"Animation,Short"	5.8	
"tt0000003"	"short"	"Pauvre Pierrot"	"Pauvre Pierrot"	0	1892	null	4	"Animation,Comedy,Romance"	6.5	
"tt0000004"	"short"	"Un bon bock"	"Un bon bock"	0	1892	null	12	"Animation,Short"	5.6	
"tt0000005"	"short"	"Blacksmith Scene"	"Blacksmith Scene"	0	1893	null	1	"Comedy,Short"	6.2	
…	…	…	…	…	…	…	…	…	…	
"tt9916708"	"tvEpisode"	"Horrid Henry Goes Gross"	"Horrid Henry Goes Gross"	0	2012	null	null	"Adventure,Animation,Comedy"	8.6	
"tt9916766"	"tvEpisode"	"Episode #10.15"	"Episode #10.15"	0	2019	null	43	"Family,Game-Show,Reality-TV"	7.0	
"tt9916778"	"tvEpisode"	"Escape"	"Escape"	0	2019	null	null	"Crime,Drama,Mystery"	7.2	
"tt9916840"	"tvEpisode"	"Horrid Henry's Comic Caper"	"Horrid Henry's Comic Caper"	0	2014	null	11	"Adventure,Animation,Comedy"	8.8	
"tt9916880"	"tvEpisode"	"Horrid Henry Knows It All"	"Horrid Henry Knows It All"	0	2014	null	10	"Adventure,Animation,Comedy"	8.2	

In [20]:
```
df_genres = (
    df.lazy()
    .filter(pl.col("genres").is_not_null())  # Remove rows with null genres
    .with_columns([pl.col("genres").str.split(",").alias("genre_list")])  # Split genres into lists
    .explode("genre_list")  # Explode to have each genre in its own row
)

df_genres.collect()
```

Out[20]: shape: (2_575_480, 12)

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	averageRating	n
str	str	str	str	i64	i64	str	i64	str	f64	
"tt0000001"	"short"	"Carmencita"	"Carmencita"	0	1894	null	1	"Documentary,Short"	5.7	
"tt0000001"	"short"	"Carmencita"	"Carmencita"	0	1894	null	1	"Documentary,Short"	5.7	
"tt0000002"	"short"	"Le clown et ses chiens"	"Le clown et ses chiens"	0	1892	null	5	"Animation,Short"	5.8	
"tt0000002"	"short"	"Le clown et ses chiens"	"Le clown et ses chiens"	0	1892	null	5	"Animation,Short"	5.8	
"tt0000003"	"short"	"Pauvre Pierrot"	"Pauvre Pierrot"	0	1892	null	4	"Animation,Comedy,Romance"	6.5	
...	...	...	...	...	...	...	...	...	...	
"tt9916840"	"tvEpisode"	"Horrid Henry's Comic Caper"	"Horrid Henry's Comic Caper"	0	2014	null	11	"Adventure,Animation,Comedy"	8.8	
"tt9916840"	"tvEpisode"	"Horrid Henry's Comic Caper"	"Horrid Henry's Comic Caper"	0	2014	null	11	"Adventure,Animation,Comedy"	8.8	
"tt9916880"	"tvEpisode"	"Horrid Henry Knows It All"	"Horrid Henry Knows It All"	0	2014	null	10	"Adventure,Animation,Comedy"	8.2	
"tt9916880"	"tvEpisode"	"Horrid Henry Knows It All"	"Horrid Henry Knows It All"	0	2014	null	10	"Adventure,Animation,Comedy"	8.2	
"tt9916880"	"tvEpisode"	"Horrid Henry Knows It All"	"Horrid Henry Knows It All"	0	2014	null	10	"Adventure,Animation,Comedy"	8.2	

In [21]:
```
df_pairs = (
    df_genres
    .join(df_genres, on="tconst", suffix="_pair")  # Join on tconst to get all genre pairs within the same movie
    .filter(pl.col("genre_list") < pl.col("genre_list_pair"))  # Filter to avoid duplicate pairs (A, B) and (B, A)
    .select([
        pl.concat_str([pl.col("genre_list"), pl.col("genre_list_pair")], separator=", ").alias("genre_pair"),  # Co
        pl.col("averageRating"),
    ])
)

df_pairs.collect()
```

Out[21]: shape: (1_742_968, 2)

genre_pair	averageRating
str	f64
"Documentary, Short"	5.7
"Animation, Short"	5.8
"Animation, Comedy"	6.5
"Animation, Romance"	6.5
"Comedy, Romance"	6.5
...	...
"Adventure, Comedy"	8.8
"Animation, Comedy"	8.8
"Adventure, Animation"	8.2
"Adventure, Comedy"	8.2
"Animation, Comedy"	8.2

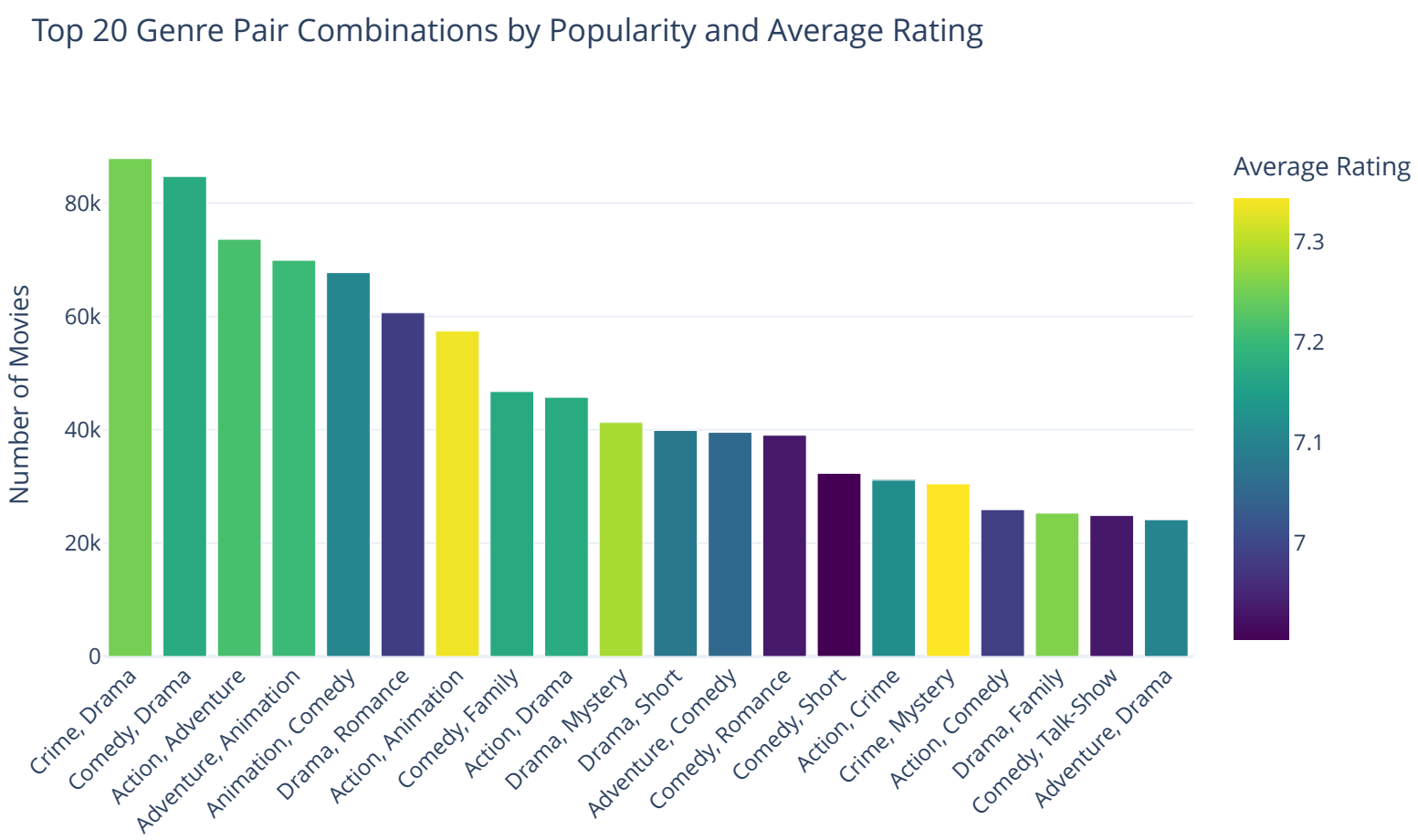
In [22]:
```
genre_pair_stats = (
    df_pairs
```

```
    .group_by("genre_pair")
    .agg([
        pl.col("averageRating").mean().alias("avgRating"),  # Calculate the average rating per genre pair
        pl.col("genre_pair").count().alias("count")  # Count of each genre pair
    ])
    .with_columns([
        (pl.col("avgRating") ** rating_preference * pl.col("count") / 1000).alias("weightedRating")  # Calculate a
    ])
    # .sort("count", descending=True)
)

genre_pair_stats.collect()
```

Out[22]: shape: (362, 4)

genre_pair	avgRating	count	weightedRating
str	f64	u32	f64
"Short, Talk-Show"	7.087291	299	15.018678
"Adult, Sport"	6.543333	30	1.284456
"Musical, Romance"	6.331863	1814	72.727782
"Adult, Reality-TV"	6.28125	16	0.631266
"Action, Talk-Show"	7.266667	9	0.47524
…	…	…	…
"Adult, Adventure"	6.130876	217	8.156517
"Animation, Comedy"	7.109504	67726	3423.214318
"Comedy, Horror"	6.43344	9103	376.765379
"Reality-TV, Thriller"	6.798585	212	9.7988
"News, War"	6.457143	14	0.583726

In [23]:
```
# Select the top genre pairs by count for better visualization
top_genre_pairs = genre_pair_stats.collect().sort("count", descending=True).head(20)

# Create a bar chart showing average rating and count for each genre pair
fig = px.bar(
    top_genre_pairs.to_pandas(),  # Convert Polars DataFrame to Pandas for compatibility with Plotly
    x="genre_pair",
    y="count",
    color="avgRating",
    title="Top 20 Genre Pair Combinations by Popularity and Average Rating",
    labels={"genre_pair": "Genre Pair", "count": "Number of Movies", "avgRating": "Average Rating"},
    color_continuous_scale="Viridis",
    template=theme
)

# Update layout for readability
fig.update_layout(
    xaxis_tickangle=-45,
    xaxis_title="Genre Pair",
    yaxis_title="Number of Movies",
    coloraxis_colorbar=dict(title="Average Rating"),
)

# Show the plot
fig.show(renderer='notebook')
```



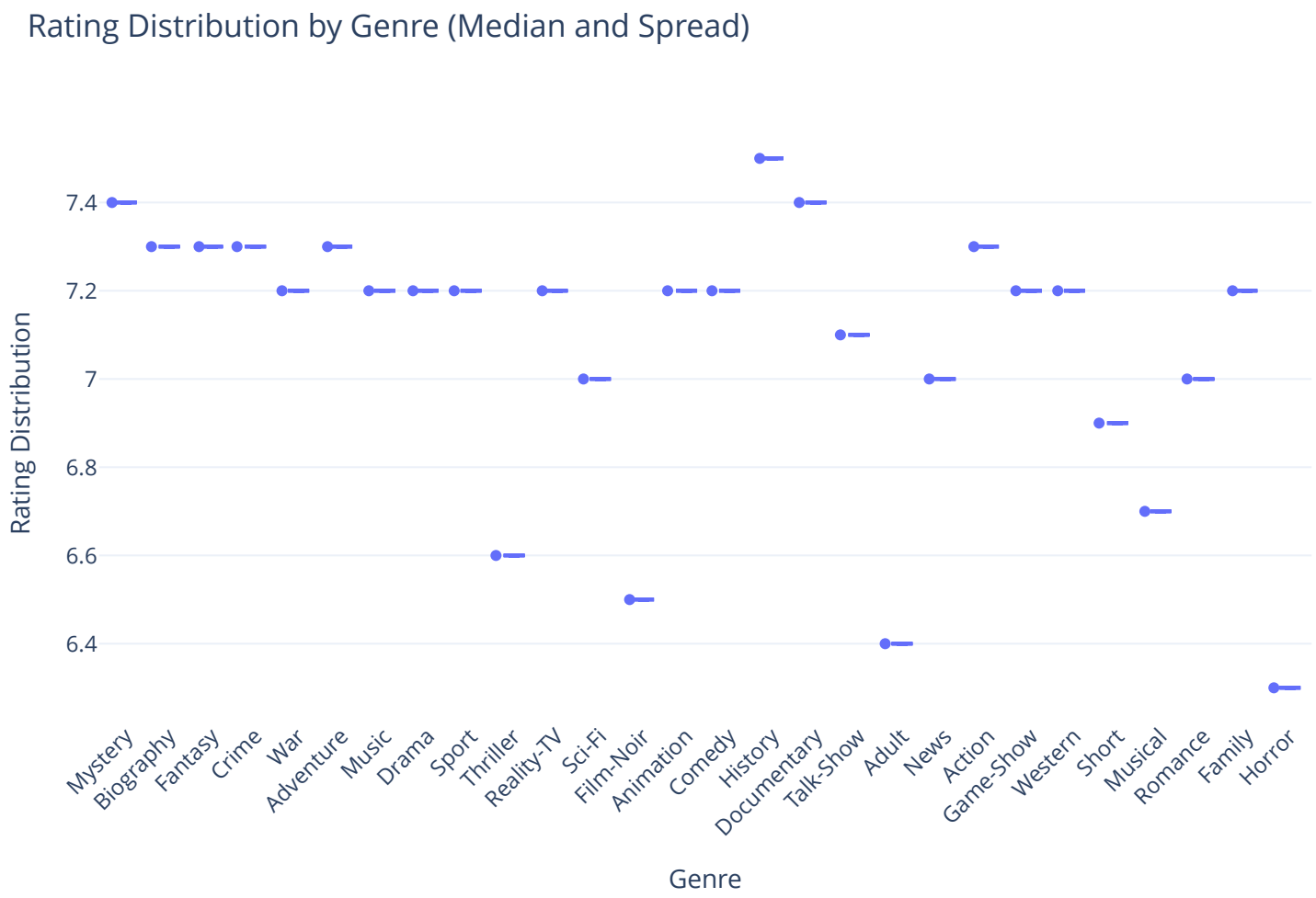
In [24]:
```
rating_distribution = (
    df.lazy()
    .select([pl.col("genres"), pl.col("averageRating")])
    .with_columns(pl.col("genres").str.split(",").alias("genres"))  # Split genres into lists
    .explode("genres")  # Expand list to separate rows
    .group_by("genres")
    .agg([
        pl.col("averageRating").median().alias("medianRating"),
        pl.col("averageRating").quantile(0.25).alias("q1Rating"),  # 25th percentile
        pl.col("averageRating").quantile(0.75).alias("q3Rating"),  # 75th percentile
        pl.col("averageRating").std().alias("stdDevRating")  # Standard deviation
    ])
)

rating_distribution.collect()
```

Out[24]: shape: (29, 5)

genres	medianRating	q1Rating	q3Rating	stdDevRating
str	f64	f64	f64	f64
"History"	7.5	6.8	8.1	1.13295
"Sport"	7.2	6.4	7.9	1.19994
"Musical"	6.7	5.7	7.7	1.419761
"Game-Show"	7.2	6.4	7.9	1.311489
"Film-Noir"	6.5	6.1	6.9	0.695806
…	…	…	…	…
"Comedy"	7.2	6.3	7.9	1.33825
"Fantasy"	7.3	6.4	8.0	1.318804
null	6.6	5.5	7.6	1.557652
"War"	7.2	6.3	7.9	1.252456
"Animation"	7.2	6.5	7.9	1.213502

In [25]:
```
# Collect the data and convert to pandas DataFrame
rating_distribution_df = rating_distribution.collect().to_pandas()

# Create a box plot to show the rating distribution by genre
fig = px.box(
    rating_distribution_df,
    x="genres",
    y="medianRating",
    title="Rating Distribution by Genre (Median and Spread)",
    labels={"genres": "Genre", "medianRating": "Median Rating"},
    points="all"  # Show all data points for better spread indication
)

# Customize layout
```

```
fig.update_layout(
    xaxis_tickangle=-45,  # Rotate x-axis labels for readability
    yaxis_title="Rating Distribution",
    template=theme  # Change the theme if desired
)

# Show the plot
fig.show(renderer='notebook')
```



In [ ]: