FDS - Final Project (Steam Game Dataset)

The main objective of this project is to investigate the steam game dataset and manipulate it by ML techniques.

Step 1: Load and Merge Data

Dataset using:

- games.csv
- tags.csv
- reviews.csv

Loading:

1. load steam game dataset into panda dataframe
2. identify problematic line
3. extract it out and reframe the frame

In [2]:

```python
import pandas as pd
from csv import reader


# Create the DataFrame using the correct header and fixed data
games = pd.read_csv('dataset/games.csv')

# Continue with your processing for tags and reviews
categories = pd.read_csv('dataset/categories.csv', on_bad_lines='skip')
genres = pd.read_csv('dataset/genres.csv', on_bad_lines='skip')
reviews = pd.read_csv('dataset/reviews.csv', on_bad_lines='skip')
tags = pd.read_csv('dataset/tags.csv')

# Check duplicate item
print(games['app_id'].duplicated().sum())        # Should be 0
print(tags['app_id'].duplicated().sum())         # Should be 0
print(reviews['app_id'].duplicated().sum())      # Should be 0
print(categories['app_id'].duplicated().sum())   # Should be 0
print(genres['app_id'].duplicated().sum())       # Should be 0

# Remove duplicate item in each dataset
tags = tags.drop_duplicates(subset=['app_id'])
reviews = reviews.drop_duplicates(subset=['app_id'])
categories = categories.drop_duplicates(subset=['app_id'])
genres = genres.drop_duplicates(subset=['app_id'])

# Merge games and reviews
games['app_id'] = games['app_id'].astype(str)
tags['app_id'] = tags['app_id'].astype(str)
merged_df = pd.merge(games, tags, on='app_id', how='left')

reviews['app_id'] = reviews['app_id'].astype(str)
merged_df = pd.merge(merged_df, reviews, on="app_id", how='left')
```

```python
categories['app_id'] = categories['app_id'].astype(str)
merged_df = pd.merge(merged_df, categories, on='app_id', how='left')

genres['app_id'] = genres['app_id'].astype(str)
merged_df = pd.merge(merged_df, genres, on='app_id', how='left')

# Check duplicate after merging

# Drop rows with missing review scores
merged_df.dropna(subset=['review_score'], inplace=True)

print(merged_df)
```

```
/var/folders/z0/t_vv9z0908vc5j6k_b12pwn40000gp/T/ipykernel_60271/279373437
7.py:6: DtypeWarning: Columns (3) have mixed types. Specify dtype option o
n import or set low_memory=False.
  games = pd.read_csv('dataset/games.csv')
/var/folders/z0/t_vv9z0908vc5j6k_b12pwn40000gp/T/ipykernel_60271/279373437
7.py:11: DtypeWarning: Columns (0,1,3,4,5,9,11,12) have mixed types. Speci
fy dtype option on import or set low_memory=False.
  reviews = pd.read_csv('dataset/reviews.csv', on_bad_lines='skip')
```

```
0
1627127
22
388189
230881
         app_id                        name release_date is_free  \
0            10              Counter-Strike   2000-11-01       0   
1            20       Team Fortress Classic   1999-04-01       0   
2            30               Day of Defeat   2003-05-01       0   
3            40          Deathmatch Classic   2001-06-01       0   
4            50   Half-Life: Opposing Force   1999-11-01       0   
...         ...                         ...          ...     ...   
140077  3297700                  Hacky Demo           \N       1   
140078  3297890         Quantum of Hope Demo           \N       1   
140079  3298020       A Night With: Succubus           \N       0   
140080  3298610                心所向往的北极星 Demo           \N       1   
140081  3298710  S.X.E. Slider: Hard Ridin'           \N       0   

       price_overview      tag review_score review_score_description posit
ive  \
0                  19    1980s            9  Overwhelmingly Positive   235
403   
1                  99   1990's            8            Very Positive     7
315   
2                  99   Action            8            Very Positive     6
249   
3                  99   1990's            8            Very Positive     2
542   
4                  99   1990's            9  Overwhelmingly Positive    22
263   
...               ...      ...          ...                      ...      
...   
140077            NaN      NaN            0          No user reviews      
0   
140078            NaN       3D            0           1 user reviews      
0   
140079            NaN       2D            0          No user reviews      
0   
140080            NaN      NaN            0          No user reviews      
0   
140081            NaN  America            0          No user reviews      
0   

       negative   total metacritic_score reviews recommendations  \
0          6207  241610               88      \N          153259   
1          1094    8409               \N      \N            6268   
2           672    6921               79      \N            4146   
3           524    3066               \N      \N            2218   
4          1111   23374               \N      \N           20144   
...         ...     ...              ...     ...             ...   
140077        0       0               \N      \N              \N   
140078        1       1               \N      \N              \N   
140079        0       0               \N      \N              \N   
140080        0       0               \N      \N              \N   
140081        0       0               \N      \N              \N   

       steamspy_user_score steamspy_score_rank steamspy_positive  \
0                        0                  \N            235397   
1                        0                  \N              7314   
2                        0                  \N              6246   
```

```
3                        0                  \N            2541
4                        0                  \N           22260
...                    ...                 ...             ...
140077                   0                  \N               0
140078                   0                  \N               0
140079                   0                  \N               0
140080                   0                  \N               0
140081                   0                  \N               0

       steamspy_negative                category   genre
0                   6207          Family Sharing  Action
1                   1092          Family Sharing  Action
2                    672          Family Sharing  Action
3                    525          Family Sharing  Action
4                   1112          Family Sharing  Action
...                  ...                     ...     ...
140077                 0               Game demo     NaN
140078                 0  Full controller support  Action
140079                 0           Single-player  Casual
140080                 0               Game demo     NaN
140081                 0  Full controller support  Casual

[140066 rows x 20 columns]
```

Step 2: Filtering and Exploring(Visualizing) the dataset

Data cleaning and preprocessing:

1. clean the dataset: drop duplicates rows
2. remove unnecessary information
3. visualize it

In [3]:
```python
from IPython.display import display

# Data frame cleaning
merged_df.shape
merged_df.drop_duplicates(subset="app_id", inplace=True)
merged_df = merged_df[merged_df['name'] != 'NaN']
merged_df = merged_df[merged_df['review_score'] != 0]
merged_df.rename(columns={'tag': 'game_type'}, inplace=True)
merged_df['name'] = merged_df['name'].str.split(',').str[0]
merged_df = merged_df.drop(columns=["steamspy_user_score", "steamspy_scor

# Strip leading/trailing spaces to clean up empty strings
merged_df['is_free'] = merged_df['is_free'].str.strip()

# Replace empty values with NaN or a meaningful default (e.g., 0)
merged_df['is_free'] = merged_df['is_free'].replace('', '0')

print(merged_df.info())
print("============================================================")
display(merged_df)

# Create a review dataset
review_df = merged_df[['app_id', 'name', 'reviews', 'category', 'genre']]
```

```python
# select only reviews = ! "\N"
review_df = review_df[review_df['reviews'] != '\\N']
print(review_df.info())
print("==========================================================")
display(review_df)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 86116 entries, 0 to 139986
Data columns (total 16 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   app_id                    86116 non-null  object
 1   name                      86116 non-null  object
 2   release_date              86116 non-null  object
 3   is_free                   85324 non-null  object
 4   price_overview            60733 non-null  object
 5   game_type                 78929 non-null  object
 6   review_score              86116 non-null  object
 7   review_score_description  86116 non-null  object
 8   positive                  86116 non-null  object
 9   negative                  86116 non-null  object
 10  total                     86116 non-null  object
 11  metacritic_score          86116 non-null  object
 12  reviews                   86116 non-null  object
 13  recommendations           86109 non-null  object
 14  category                  84494 non-null  object
 15  genre                     83026 non-null  object
dtypes: object(16)
memory usage: 11.2+ MB
None
==========================================================
```

	app_id	name	release_date	is_free	price_overview	game_type	re
0	10	Counter-Strike	2000-11-01	0	19	1980s	
1	20	Team Fortress Classic	1999-04-01	0	99	1990's	
2	30	Day of Defeat	2003-05-01	0	99	Action	
3	40	Deathmatch Classic	2001-06-01	0	99	1990's	
4	50	Half-Life: Opposing Force	1999-11-01	0	99	1990's	
...	...	...	...	...	...	...	
139901	3282750	-Fell in love with the Nobility girl- Demo	2024-10-11	NaN	NaN	Adventure	
139903	3282810	Underworld Overseer Demo	2024-10-14	NaN	NaN	Action RPG	
139919	3283400	Streetball Fury Demo	2024-10-13	NaN	NaN	2D	
139946	3284630	Campervan Simulator Demo	2024-10-16	NaN	NaN	NaN	
139986	3286590	ExoFrontier: Venus (Demo)	2024-10-13	NaN	NaN	1990's	

86116 rows × 16 columns

```
<class 'pandas.core.frame.DataFrame'>
Index: 10894 entries, 24 to 139903
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   app_id    10894 non-null  object
 1   name      10894 non-null  object
 2   reviews   10894 non-null  object
 3   category  10834 non-null  object
 4   genre     10834 non-null  object
dtypes: object(5)
memory usage: 510.7+ KB
None
==========================================================
```

	app_id	name	reviews	category	genre
24	570	Dota 2	“Современный многопользовательский шедевр.”<br...	Включён античит Valve	Бесплатные
30	1200	Red Orchestra: Ostfront 41-45	<strong>"... RO is also one of the market...	Family Sharing	Action
36	1510	Uplink	75 – Metacritic 	Family Sharing	Indie
51	1900	Earth 2160	“It may not replace "Star Craft" in ...	Co-op	Strategy
52	1930	Two Worlds Epic Edition	“The big player alongside Oblivion and Gothic ...	Co-op	RPG
...	...	...	...	...	...
138341	3238120	Lonely Mountains: Snow Riders Demo	“Lonely Mountains: Snow Riders is a slick ski ...	Full controller support	Indie
138504	3242370	101 Cats Hidden in Australia	“🏆 I loved! ❤️❤️❤️ ” \n\\nvaldi \n\\n...	Family Sharing	Casual
139128	3259160	FREE DUROV - DEMO	“Add to wishlist and save Pasha” CyberTopor...	Game demo	Adventure
139231	3261900	Dinocop Demo	“It’s a lot of fun and the game does a great j...	Full controller support	Adventure
139903	3282810	Underworld Overseer Demo	“Underworld Overseer is the Dungeon Keeper spi...	Game demo	Indie

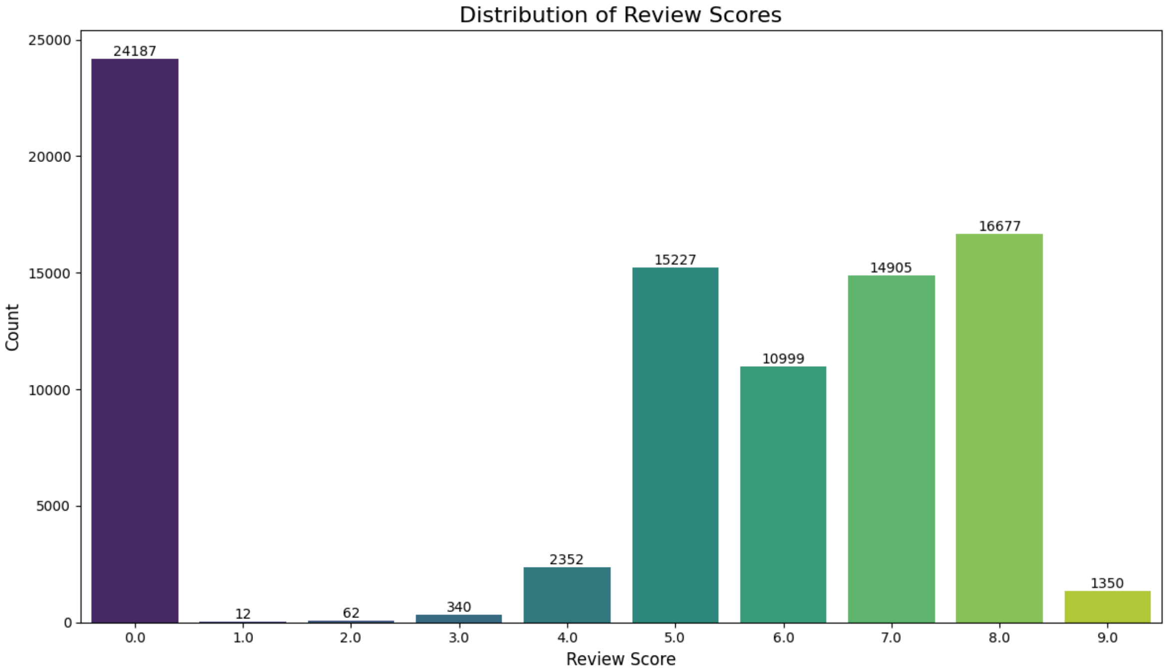
10894 rows × 5 columns

In [4]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Convert 'review_score' to numeric, coercing errors to NaN
merged_df['review_score'] = pd.to_numeric(merged_df['review_score'], erro

# Drop NaN values that resulted from the conversion
merged_df.dropna(subset=['review_score'], inplace=True)

# Plot the distribution of review scores
plt.figure(figsize=(12, 7))
ax = sns.countplot(data=merged_df, x='review_score', order=sorted(merged_
plt.title('Distribution of Review Scores', fontsize=16)
plt.xlabel('Review Score', fontsize=12)
plt.ylabel('Count', fontsize=12)

# Annotate bars with counts
```

```
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2.
                ha='center', va='center', fontsize=10, color='black', xyt
                textcoords='offset points')

plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

```
/var/folders/z0/t_vv9z0908vc5j6k_b12pwn40000gp/T/ipykernel_60271/26846426
2.py:12: FutureWarning:

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

  ax = sns.countplot(data=merged_df, x='review_score', order=sorted(merged
_df['review_score'].unique()), palette="viridis")
```



In [5]:

```
# Plot the top 10 games with the highest review scores
merged_df['review_score'] = pd.to_numeric(merged_df['review_score'], erro
top_10_games = merged_df.nlargest(10, 'review_score')

sns.set_theme(style="whitegrid")

fig, ax = plt.subplots(figsize=(12, 6))
ax.axis('tight')
ax.axis('off')

table_data = top_10_games[['name', 'review_score', 'release_date', 'revie
table = ax.table(cellText=table_data.values, colLabels=table_data.columns
table.auto_set_font_size(False)
table.set_fontsize(12)
table.scale(1.2, 1.2)

# Set font family to serif for all text elements
plt.title('Top 10 Games with Highest Review Scores', fontsize=16, fontwei
for key, cell in table.get_celld().items():
    cell.set_text_props(fontfamily='serif')
```

```
plt.show()
```

Top 10 Games with Highest Review Scores

name	review_score	release_date	review_score_description	price_overview
Counter-Strike	9.0	2000-11-01	Overwhelmingly Positive	19
Half-Life: Opposing Force	9.0	1999-11-01	Overwhelmingly Positive	99
Half-Life	9.0	1998-11-08	Overwhelmingly Positive	0
Half-Life 2	9.0	2004-11-16	Overwhelmingly Positive	0
Counter-Strike: Source	9.0	2004-11-01	Overwhelmingly Positive	75
Half-Life 2: Episode One	9.0	\N	Overwhelmingly Positive	0
Portal	9.0	2007-10-10	Overwhelmingly Positive	75
Half-Life 2: Episode Two	9.0	2007-10-10	Overwhelmingly Positive	59
Left 4 Dead	9.0	2008-11-17	Overwhelmingly Positive	75
Left 4 Dead 2	9.0	2009-11-16	Overwhelmingly Positive	75

Step 3: Apply ML techniques

Key Steps to Start

1. Clean the dataset (fix malformed fields, handle missing values).
2. Feature engineering: Convert languages to a one-hot encoded list (e.g., "English", "French").
3. Normalize/scale numerical features (e.g., price_overview, review_score).
4. Exploratory Data Analysis (EDA): Visualize correlations between variables (e.g., price vs. reviews)

ML techniques

1. Regression tasks - Linear Regression
2. Classification tasks - Logistic Regression (Popular vs Unpopular)
3. Recommendation systems - content-based filtering

Linear Regression - find out whether a future game will be popular

1. Data Preprocessing Before applying ML models, you need to clean and preprocess the dataset:

Convert columns with numerical values (price_overview, review_score, positive, negative, total, metacritic_score, recommendations) to numeric types. Handle missing values (e.g., drop rows with too many missing values or use imputation methods). Convert categorical columns (genre, category, game_type) into numerical form (one-hot encoding or label encoding). Convert release_date to a proper datetime format and extract useful features like year of release.

Use linear regression to predict a game's future popularity based on features like price_overview, genre, release_date, and recommendations.

- Target Variable: recommendations (proxy for popularity)
- Features:
  - price_overview (game price)
  - genre (one-hot encoded)
  - game_type
  - release_date (year)
  - review_score
  - metacritic_score

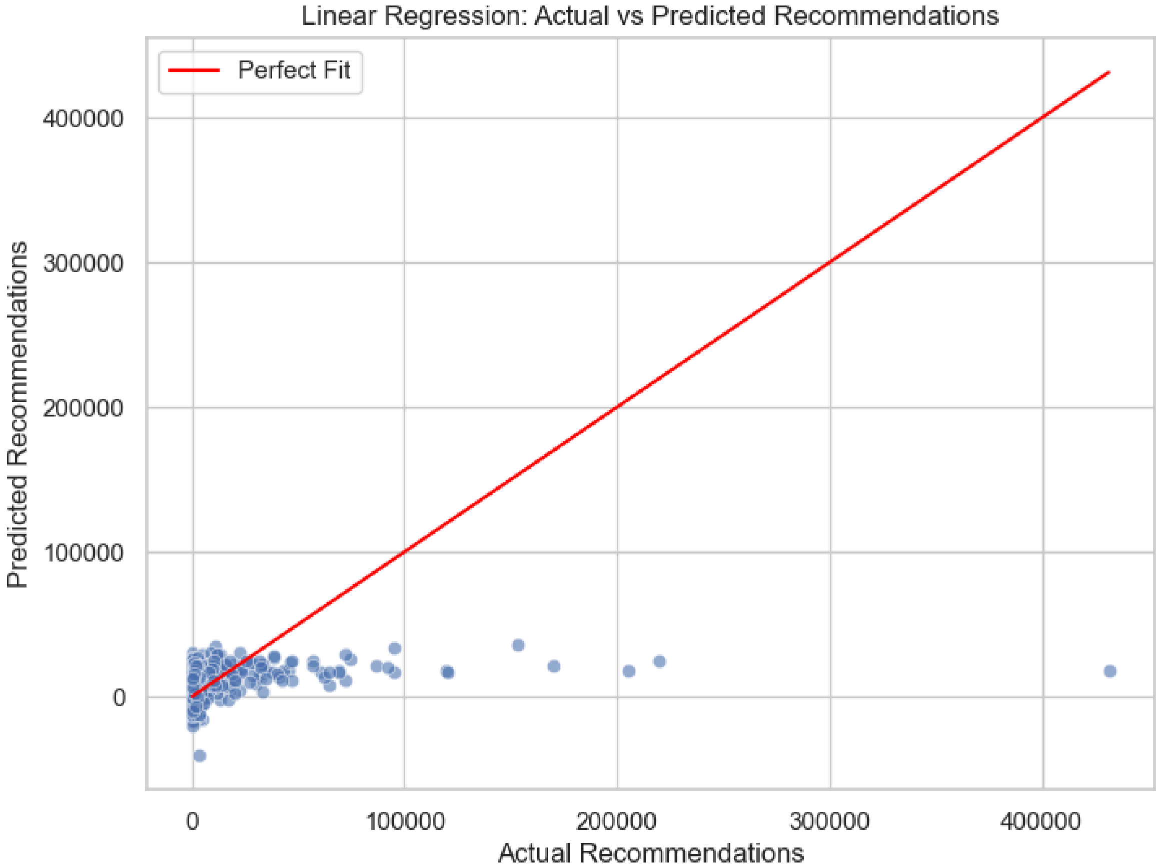
In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import accuracy_score

import seaborn as sns
import matplotlib.pyplot as plt

linear_regression_df = merged_df

# Convert numerical columns
linear_regression_df["price_overview"] = pd.to_numeric(linear_regression_
linear_regression_df["recommendations"] = pd.to_numeric(linear_regression
linear_regression_df["review_score"] = pd.to_numeric(linear_regression_df
linear_regression_df["metacritic_score"] = pd.to_numeric(linear_regressio

# Convert release_date to datetime and extract year
linear_regression_df["release_date"] = pd.to_datetime(linear_regression_d
linear_regression_df["release_year"] = linear_regression_df["release_date

# One-hot encoding for categorical data
linear_regression_df = pd.get_dummies(linear_regression_df, columns=["gen

# Drop missing values
linear_regression_df = linear_regression_df.dropna()

# Predicting the future popular trend
# Define features and target
X = linear_regression_df[["price_overview", "review_score", "metacritic_s
y = linear_regression_df["recommendations"]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, 

# Train Linear Regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

# Predictions
y_pred = lin_reg.predict(X_test)

print(y_pred)

# Scatter plot of actual vs. predicted recommendations
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_test, y=y_pred, alpha=0.6)
```

```
sns.lineplot(x=y_test, y=y_test, color='red', label="Perfect Fit")  # Ide

plt.xlabel("Actual Recommendations")
plt.ylabel("Predicted Recommendations")
plt.title("Linear Regression: Actual vs Predicted Recommendations")
plt.legend()
plt.show()
```

```
[ 1.50603413e+04  1.73240055e+04  4.32751779e+03  1.35485543e+04
  1.61169185e+04  1.89271740e+04  1.87851202e+04  3.05828034e+04
  1.52272774e+04  1.17906440e+04  1.03581869e+04  2.02501724e+04
  2.24168012e+04  2.06029451e+04  4.55536203e+03  1.26813117e+04
  6.55579208e+03  2.53512058e+04  1.30435060e+04  1.82437933e+04
  1.46742825e+04  7.32752105e+03  2.27161357e+04  1.43127609e+04
  1.54949725e+04  2.07704170e+04  1.55923331e+04  2.95344394e+04
  1.43351771e+04 -3.34119601e+03  1.28218926e+04  2.18187811e+04
  6.38439505e+03 -1.15942170e+03  1.61161671e+04  1.72951926e+04
  1.84366163e+04  2.91017907e+03  4.19565001e+03  9.12719419e+03
  1.74690611e+04  1.11520358e+04  6.80333902e+03 -3.80294084e+03
  1.76991630e+04  1.51082886e+03  1.19195249e+04  1.16742461e+04
  3.04340910e+04  1.79633529e+04  7.30154533e+03  9.94050561e+03
 -8.06203251e+03  9.86140930e+02 -9.76772848e+02  2.39266435e+02
  2.44316814e+04  1.50708103e+04  9.96785245e+03  1.81270907e+04
  1.47919913e+04  1.55031068e+03  9.29094073e+03  1.97220530e+04
  7.14168445e+03  1.53242900e+04  3.13406487e+03  1.52496936e+04
  1.81270907e+04  5.63461197e+03 -8.80513363e+03  1.74934491e+04
 -6.46383868e+03  1.40391290e+04  2.04792694e+03  1.67477335e+04
  2.17379399e+03  9.31601614e+03  1.67127801e+04 -4.07082815e+04
  4.76415399e+03  1.47030954e+04  1.19819621e+04  1.77143400e+04
  2.09465312e+03  4.93448840e+03  2.28058315e+02  5.22459971e+03
  4.44712903e+03  1.60527788e+04  2.64209156e+04  5.75497498e+03
  1.89321476e+04  1.76141168e+04  8.27266696e+03  4.13667454e+03
 -2.74636363e+03  1.67568769e+04 -1.92821305e+02  1.31417574e+04
 -2.86826391e+03  2.39266435e+02 -5.95089800e+03  1.03387656e+04
  7.72960278e+03  3.38435857e+03  1.41450856e+04  2.14192102e+04
  2.19354837e+04  2.32478641e+03  5.35251048e+03  2.21058905e+04
 -5.03486733e+03  3.46673718e+04  1.65169514e+04  1.57114385e+04
 -1.63480241e+03  1.59189313e+04 -5.30727486e+03  1.07885424e+04
 -9.91614093e+03  6.13911622e+03  2.51160646e+04  1.68110316e+04
  1.29597090e+04  1.03581869e+04  1.43127609e+04  9.63004615e+03
 -8.31717192e+02  1.04696182e+04  5.50670120e+03  9.15987746e+02
  1.75487540e+04  5.59807223e+03 -2.60503246e+03  6.32739112e+03
  7.27722216e+03  7.60839598e+03  1.10562374e+04  1.91754547e+04
  1.69448791e+04 -1.35589921e+03  2.40463548e+04 -1.55914412e+03
  1.77367562e+04  9.56630984e+03  2.84303167e+03 -2.94355306e+02
  1.16518298e+04  1.32980212e+04  4.62903001e+03  2.44419436e+04
  1.22014562e+04 -5.36872292e+03 -4.66042020e+03  3.93715008e+03
 -4.61594379e+03  1.66659759e+04  3.78088630e+03  4.08741701e+03
  1.70899348e+04  1.53947493e+04  5.70697812e+02 -4.76321172e+03
  6.83396843e+03  7.06613845e+03 -1.57426812e+04  2.65179405e+04
  1.89939122e+04  1.66712473e+04  1.07261052e+04  1.51270542e+04
  1.44239888e+04  1.92616073e+04  1.63187537e+04  1.27342780e+04
  2.32390008e+04  2.59982736e+04  3.59946390e+04  1.79116466e+03
  3.34938249e+04  1.51270542e+04  1.60988473e+04  1.96996368e+04
  2.23718723e+04  3.60701782e+03  1.77367562e+04  1.74143314e+04
  1.07435065e+04  8.66300145e+03  2.91777293e+04  2.71512089e+03
  1.73832444e+04  2.45735387e+04  1.55806191e+04 -1.65797864e+03
  2.49545660e+03  2.94690732e+04  2.16849335e+04  2.16849335e+04
  1.86818952e+04 -9.90227365e+03  2.01123875e+04  2.58599109e+04
  1.06915069e+04  1.85286333e+04  1.64094890e+04  7.07157322e+03
 -4.38956005e+03  3.94034837e+03  3.49561975e+03  9.45187665e+03
  1.74914774e+04  1.36523407e+03  1.56176119e+04  1.71901456e+04
  5.23098400e+03  5.21958484e+03  1.19227109e+04  8.41045193e+03
  1.42901413e+04  1.68110316e+04  1.44466084e+04  6.02464179e+03
  7.41067864e+03  8.91127523e+03  1.66620071e+04  4.54801155e+03
  2.16565635e+04  1.40197871e+04  1.76160885e+04  1.46885188e+04
  2.27140796e+04  1.77367562e+04 -6.60218325e+01 -4.11065686e+03
  2.24656025e+04  1.37408839e+04  1.02612866e+03  5.74878173e+03
```

```
  2.44396912e+04 -2.55308441e+03  7.63448458e+02  1.10874712e+03
  8.60299201e+03  2.68235287e+04  2.08581091e+04  1.78818119e+04
  9.62874703e+03  1.50412601e+04  4.42770774e+03  2.88579108e+03
  3.43436109e+03  1.68416540e+04  8.65179333e+03  2.98357588e+04
  1.87851202e+04  8.35346885e+03  2.23705494e+04 -1.81745126e+03
  1.44578165e+04  1.34990033e+04  1.33651558e+04  2.54122008e+04
 -6.83752357e+03  9.20136726e+03  2.11110402e+04  1.29598668e+04
  3.98668331e+03  5.61219573e+03  1.33651558e+04  8.88660317e+03
  2.27220895e+04  1.13202020e+04  1.74069477e+04  2.11383353e+04
  1.95531259e+04 -2.52669933e+03 -2.42984457e+03  2.32913004e+04
  4.25840421e+02 -4.67166154e+03  7.19868838e+03  1.22160328e+04
  1.00334774e+04  7.19761094e+03  1.06177021e+04  7.72906365e+03
  1.68222397e+04  1.35433000e+04  3.63583065e+03 -1.53523467e+04
  3.73113971e+02 -4.07682916e+03  2.02238188e+04 -1.73040191e+04
  1.76253249e+04  6.31799264e+03  1.49455117e+04  1.23484836e+04
  1.17662561e+04  6.78092278e+03  5.49549307e+03  1.73464217e+04
 -1.26232036e+04 -3.35506330e+03  1.21790400e+04  1.82721463e+04
 -1.38225601e+04  4.31239073e+03  2.67744274e+04 -7.91696243e+02
  1.09479939e+04  2.01011794e+04  2.17535761e+04  7.71785553e+03
  1.93596072e+04  1.66883922e+04 -5.34673553e+03  5.25518094e+03
  5.01024744e+02  1.75917005e+04 -1.50361404e+03  2.03464582e+04
  1.43239690e+04  1.37826421e+04  1.68446559e+04  2.25710800e+04
 -3.26002223e+03  1.71730131e+04 -2.56318620e+03 -1.16942807e+04
  7.56660663e+03  1.40080398e+04  2.20672582e+04  7.76562758e+03
  1.73496200e+04 -9.85385823e+02 -8.41411146e+02  1.44227955e+04
  4.27628872e+03  1.94207336e+04  1.29122246e+04  1.98366826e+04
  9.29040160e+03  1.89829146e+04  9.04709451e+03  5.33603100e+03
  9.82582497e+03  1.36659394e+04  9.58371121e+03  2.30091989e+04
  1.16630379e+04  5.88656670e+03  5.76618310e+03  1.45705095e+04
  9.67774114e+03  9.59372600e+03 -1.22522904e+04  2.59291251e+04
 -2.88640075e+02  8.76322463e+03  4.03737325e+03  4.64151458e+03
  4.53328158e+03 -3.82034221e+03  1.21725111e+04  5.87761439e+03
  9.58054614e+03  2.28447289e+04  7.07331050e+03  2.90503940e+04
  1.69448791e+04  1.33665059e+04  1.29619522e+04  1.66659636e+04
 -7.51203261e+03  2.50792419e+04  7.87609105e+03  1.88965515e+04
  1.93093023e+04  6.01146193e+03  1.60673657e+04  2.41043305e+03
  4.30506833e+03  2.22456323e+03  1.13577698e+04  9.26179211e+03
  7.46456688e+03  2.72632901e+03 -1.68392461e+04  1.69548484e+04
  9.55190343e+03  1.14833840e+04  1.00440446e+04  1.30079099e+04
  3.84322091e+02  1.91183234e+03  2.67421919e+04  2.36622205e+04
  1.48275032e+04  1.74690611e+04  6.66949149e+03  1.56176119e+04
  2.46984206e+03  2.01348037e+04  2.08328769e+04  1.20601048e+04
  7.15894010e+03  1.50189176e+04  2.27049446e+04  6.86278126e+03
 -4.35573235e+03  4.44712903e+03  1.24377701e+04  1.65545446e+04
  4.11923173e+03 -1.26570313e+04  1.22496571e+04  1.73352136e+04
  3.54878734e+03  5.60372608e+03 -6.61212585e+03  1.27514229e+04
  2.13227272e+04  2.12609747e+04  8.52094075e+03  6.95159297e+03
  2.24656025e+04  1.02643603e+04  1.74578530e+04  1.96996368e+04
 -6.99926205e+03 -7.38959653e+03  2.43090420e+04  2.91829439e+04
  1.49339555e+04  1.06845947e+04  3.15667215e+03  1.93986531e+04
  1.55285968e+04  1.55887990e+04  8.33384440e+01  6.84090081e+03
  1.82741181e+04  1.82721463e+04  4.51887516e+03  5.55518955e+03
  1.85114714e+04  1.00822787e+04  2.45867186e+04  6.00025381e+03
  1.20807886e+04  7.99679195e+03  9.72240419e+02  6.80034407e+03
 -7.95060121e+03  4.81712034e+03  7.49199799e+03  1.37885788e+04
  1.21457095e+04  5.13254152e+03 -4.31071938e+03  1.45710162e+04
 -1.07059943e+03  2.27596573e+04  1.74961730e+04  1.82385220e+04
  7.45837363e+03  7.09045538e+03  1.02161595e+04  7.98555060e+03
  1.05576594e+04  2.98900921e+03  8.52291249e+03 -2.01294271e+04
  1.94319417e+04  1.18963258e+04  1.49451636e+04  7.97434248e+03
```

```
 1.11246889e+04  4.59218469e+03  1.15291904e+04  1.43351771e+04
 2.15592922e+04  1.57776097e+04  1.52198033e+04  1.72125742e+04
 2.63629936e+04 -2.45175491e+03  1.39005792e+04  6.68891278e+03
 1.50014129e+04  1.07773342e+04  3.90852830e+03  1.74802692e+04
-8.27779765e+02  1.13015163e+04  1.36483347e+04  7.57279988e+03
 1.69755016e+04  4.32448961e+03  1.87627040e+04 -1.09024036e+04
-1.27908788e+04  2.70391277e+03  5.80320910e+03  9.68894926e+03
 3.09404392e+03  2.87138467e+03  5.23777957e+03  9.20478818e+03
 1.66169713e+04  2.06668920e+04  2.68317006e+04 -1.24446792e+03
 4.20062365e+03 -2.67192511e+03  7.74030500e+03  6.80333902e+03
 8.39401831e+00  1.80044513e+04 -9.16611010e+03  9.13319889e+03
 1.57474906e+04  2.42540763e+04  1.39112183e+04  2.30161382e+04
-3.51263674e+03  5.23079296e+03 -6.87588350e+03  1.60079464e+04
-4.50399860e+03  1.58853069e+04  1.11087656e+04 -1.39725513e+04
 7.85170307e+03  1.70660768e+03  4.73133804e+02  4.07776160e+03
 1.14985610e+04  1.64319052e+04  1.45772454e+04  1.84987430e+04
-4.08803728e+03  1.02275254e+04  1.57738756e+04  1.87294154e+04
 1.65789326e+04  7.70570848e+03 -7.08807376e+03  1.02612866e+03
 2.34057336e+04  8.77939944e+03 -2.15898443e+03  1.07425676e+04
 1.36522722e+04  1.63842102e+04  2.11963283e+04  1.62644210e+04
 8.13881943e+03  8.61497068e+03  5.10159801e+02 -4.22188482e+03
 2.20567892e+04  1.21710812e+04  8.16133206e+03  8.13079731e+03
 1.18369064e+04  2.16737254e+04  2.65622500e+03 -4.67112241e+03
 2.16983941e+04  1.21459216e+04  2.97160784e+03  1.43127609e+04
 1.98753149e+04  2.07950757e+03  1.53723330e+04  9.14534594e+03
 1.59209030e+04 -1.25343919e+04  2.47113237e+04  1.77387279e+04
-1.19076879e+04  7.07103409e+03  2.77166145e+04  8.99443283e+03
 2.24768106e+04  2.23078970e+03  8.23082944e+03  9.54916496e+03
-1.26682394e+04  1.62545360e+04  8.50080091e+03  1.87963284e+04
 1.14015361e+04  2.36666805e+04 -7.56023350e+03 -6.84873169e+03
 8.03378472e+03  1.89189678e+04  1.08661460e+04  1.73404849e+04
 1.50044148e+04  1.15916276e+04  2.09858056e+04  2.11495434e+04
 4.75174315e+03 -1.09937904e+03  1.81323620e+04  2.42886817e+04
-8.52951076e+03  1.35082790e+04  1.11308338e+04  2.46011250e+04
-1.01249329e+04  7.58400800e+03  2.08900546e+04  1.95883990e+03
 1.35320919e+04 -2.96123708e+02  6.52497496e+03  1.30191180e+04
 1.55173887e+04  1.04870196e+04  1.24499002e+04  1.25663463e+04
-6.18476537e+03]
```



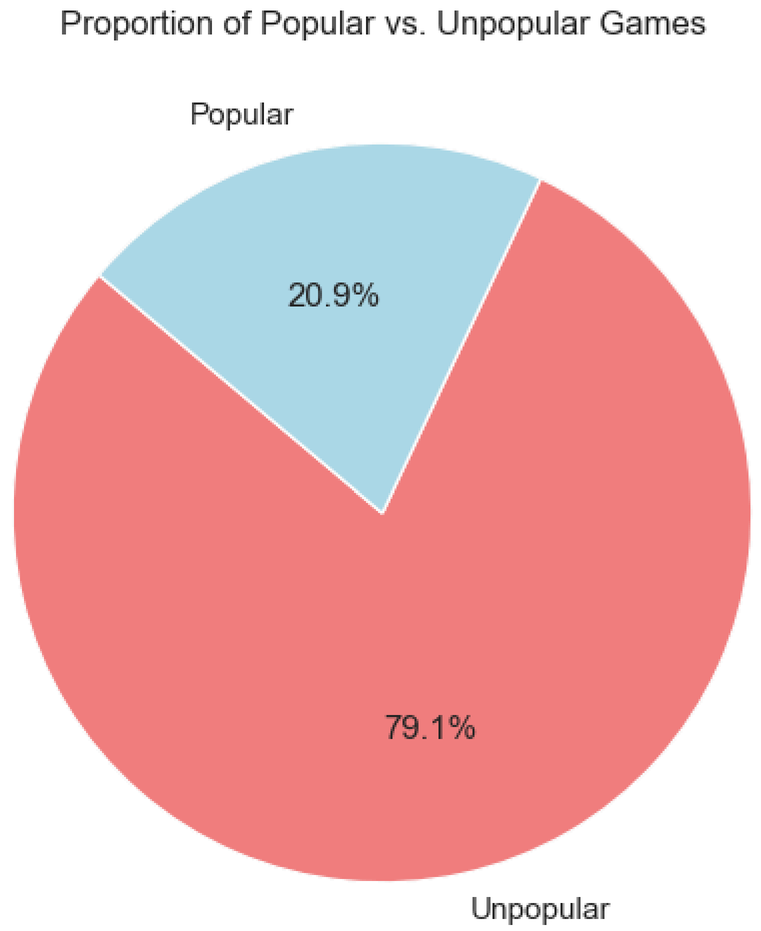
Classify the popular and unpopular games - logistic regression

Finally using a pie chart to represent

In [7]:
```
logistic_regression_df = merged_df

# Define target variable: 1 if review_score > 7, else 0
logistic_regression_df["popular"] = (logistic_regression_df["review_score

# Features
X = logistic_regression_df[["price_overview", "metacritic_score", "releas
y = logistic_regression_df["popular"]

# Impute missing values using the mean - fix missing data
X = X.fillna(X.mean())

# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# Train Logistic Regression model
log_reg = LogisticRegression(max_iter=1000)  # Increase max_iter if neede
log_reg.fit(X_train, y_train)

# Predictions and accuracy
y_pred = log_reg.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))

# Count unique values
popular_counts = logistic_regression_df["popular"].value_counts()
```

```
# Pie chart
plt.figure(figsize=(6, 6))
plt.pie(popular_counts, labels=["Unpopular", "Popular"], autopct="%1.1f%
plt.title("Proportion of Popular vs. Unpopular Games")
plt.show()
```

```
Logistic Regression Accuracy: 0.7984091041049759
```

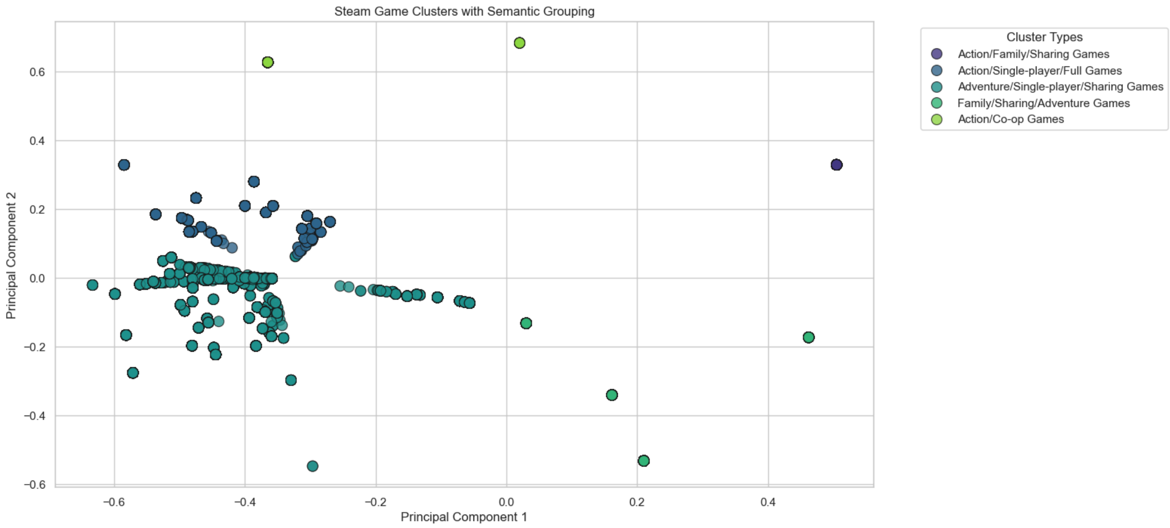


Content-Based Game Recommendation System - K-means

In [9]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Assuming merged_df contains your game data with "name", "genre", and "c
# Prepare features
content_based_df = merged_df.copy()
content_based_df["combined_features"] = content_based_df["genre"].fillna(

# TF-IDF Vectorization
tfidf = TfidfVectorizer(stop_words="english", max_features=500)
tfidf_matrix = tfidf.fit_transform(content_based_df["combined_features"])

# Dimensionality Reduction with PCA
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(tfidf_matrix.toarray())

# K-means Clustering
kmeans = KMeans(n_clusters=5, random_state=42)
clusters = kmeans.fit_predict(reduced_features)
```

```python
# Add cluster information to DataFrame
content_based_df["cluster"] = clusters

# Generate cluster descriptions
def get_cluster_labels(df, n_terms=3):
    cluster_labels = {}
    for cluster in sorted(df["cluster"].unique()):
        # Get top terms for the cluster
        terms = " ".join(df[df["cluster"] == cluster]["combined_features"
        top_terms = pd.Series(terms).value_counts().head(n_terms).index.t
        # Create descriptive label
        cluster_labels[cluster] = f"{'/'.join(top_terms)} Games"
    return cluster_labels

# Get and apply cluster labels
cluster_labels = get_cluster_labels(content_based_df)
content_based_df["cluster_label"] = content_based_df["cluster"].map(clust

# Visualize clusters with descriptions
plt.figure(figsize=(14, 8))
scatter = sns.scatterplot(
    x=reduced_features[:, 0],
    y=reduced_features[:, 1],
    hue=content_based_df["cluster_label"],
    palette="viridis",
    s=100,
    alpha=0.8,
    edgecolor="k",
    legend="full"
)
plt.title("Steam Game Clusters with Semantic Grouping")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', title="Cluster Typ
plt.show()

# Enhanced cluster analysis
def print_cluster_details(df):
    for cluster in sorted(df["cluster"].unique()):
        cluster_data = df[df["cluster"] == cluster]

        print(f"\nCluster {cluster} ({cluster_labels[cluster]})")
        print("-" * 40)
        print("Top Features:")
        terms = " ".join(cluster_data["combined_features"]).split()
        print(pd.Series(terms).value_counts().head(5).to_string())

        print("\nExample Games:")
        print(cluster_data["name"].head(5).to_string(index=False))

print_cluster_details(content_based_df)

# Visualize cluster distribution with labels
plt.figure(figsize=(12, 6))
sns.countplot(x="cluster_label", data=content_based_df, palette="viridis"
plt.title("Game Distribution by Cluster Type")
plt.xlabel("Cluster Type")
plt.ylabel("Number of Games")
plt.xticks(rotation=45, ha='right')
```

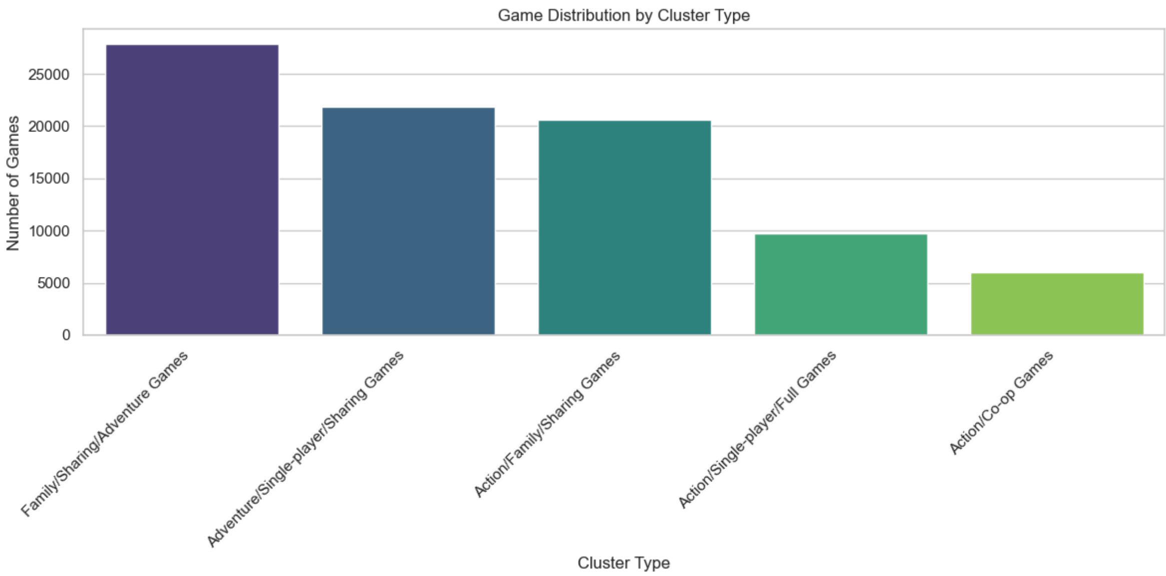
```
plt.tight_layout()
plt.show()
```



```
Cluster 0 (Action/Family/Sharing Games)
----------------------------------------
Top Features:
Action     20656
Family     20656
Sharing    20656

Example Games:
           Counter-Strike
    Team Fortress Classic
            Day of Defeat
       Deathmatch Classic
Half-Life: Opposing Force

Cluster 1 (Adventure/Single-player/Sharing Games)
----------------------------------------
Top Features:
Adventure       5511
Single-player   5100
Sharing         4161
Family          4161
Casual          4032

Example Games:
 Half-Life 2: Lost Coast
Half-Life 2: Episode One
                  Dota 2
        Counter-Strike 2
   Rag Doll Kung Fu Demo

Cluster 2 (Family/Sharing/Adventure Games)
----------------------------------------
Top Features:
Family       27878
Sharing      27878
Adventure    12419
Casual       10585
Indie         4748

Example Games:
                          Rag Doll Kung Fu
                                  Darwinia
                                    Uplink
                Gumboy - Crazy Adventures™
Safecracker: The Ultimate Puzzle Adventure

Cluster 3 (Action/Co-op Games)
----------------------------------------
Top Features:
Action    6027
Co-op     6019

Example Games:
               Killing Floor
                       Quake
                    Quake II
       ThreadSpace: Hyperbol
Judge Dredd: Dredd vs. Death

Cluster 4 (Action/Single-player/Full Games)
```

```
----------------------------------------
Top Features:
Action           8485
Single-player    2327
Full             2267
controller       2267
support          2267

Example Games:
           Half-Life 2: Demo
                 Half-Life 2
      Counter-Strike: Source
       Day of Defeat: Source
Half-Life Deathmatch: Source
```

```
/var/folders/z0/t_vv9z0908vc5j6k_b12pwn40000gp/T/ipykernel_60271/174414209
2.py:76: FutureWarning:

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

  sns.countplot(x="cluster_label", data=content_based_df, palette="viridi
s", order=content_based_df["cluster_label"].value_counts().index)
```



In [ ]: