In [74]:
```
#Data is before 2024
```

In [27]:
```
pip install pandas mysql-connector-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in c:\users\satye\appdata\roaming\python\pytho
n313\site-packages (2.2.3)
Requirement already satisfied: mysql-connector-python in c:\users\satye\appdata\roam
ing\python\python313\site-packages (9.2.0)
Requirement already satisfied: numpy>=1.26.0 in c:\users\satye\appdata\roaming\pytho
n\python313\site-packages (from pandas) (2.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\satye\appdata\roam
ing\python\python313\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\satye\appdata\roaming\python
\python313\site-packages (from pandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\satye\appdata\roaming\pyth
on\python313\site-packages (from pandas) (2025.1)
Requirement already satisfied: six>=1.5 in c:\users\satye\appdata\roaming\python\pyt
hon313\site-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Note: you may need to restart the kernel to use updated packages.
```

In [29]:
```
# Import necessary libraries
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sqlalchemy

# MySQL Connection
db_config = {
    "host": "localhost",
    "user": "Satyen",
    "password": "Satyen@789",
    "database": "Bakugo"
}

# Connect to MySQL ,Database
conn = mysql.connector.connect(host=db_config["host"], user=db_config["user"], pass
cursor = conn.cursor()
cursor.execute("CREATE DATABASE IF NOT EXISTS Bakugo")
cursor.close()
conn.close()

# Connect with database
conn = mysql.connector.connect(**db_config)
cursor = conn.cursor()

# Creating table for data storing
cursor.execute("""
CREATE TABLE Katsuki (
    Score REAL,
    Popularity INTEGER,
    `Rank` INTEGER,
    Members INTEGER,
```

```
    Description TEXT,
    Synonyms TEXT,
    Name TEXT NOT NULL,
    Type TEXT,
    Episodes INTEGER,
    Status TEXT,
    Aired TEXT,
    Premiered TEXT,
    Broadcast TEXT,
    Producers TEXT,
    Licensors TEXT,
    Studios TEXT,
    Source TEXT,
    Genres TEXT,
    Demographic TEXT,
    Duration TEXT,
    Rating TEXT
);

""")

conn.commit()
cursor.close()
conn.close()

print("Database and table created successfully!")
```

```
Database and table created successfully!
```

In [3]:

```
import mysql.connector
import pandas as pd

# MySQL Connection
db_config = {
    "host": "localhost",
    "user": "Satyen",
    "password": "Satyen@789",
    "database": "Bakugo"
}

# Connect to database
conn = mysql.connector.connect(**db_config)
cursor = conn.cursor()

# Reading CSV
csv_file = "C://Users//satye//Downloads//AnimeProc.csv"  # Change this to your actu
df = pd.read_csv(csv_file)

# Fixing column issues
df["Name"] = df["Name"].fillna("Unknown")  # Fix for Name column
df["Episodes"] = pd.to_numeric(df["Episodes"], errors='coerce').fillna(0).astype(in

# SQL Query to insert data
query = """
INSERT INTO Katsuki (Score, Popularity, `Rank`, Members, Description, Synonyms, Nam
                     Premiered, Broadcast, Producers, Licensors, Studios, Source, G
```

```
VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
"""

# Inserting all data in table
for _, row in df.iterrows():
    cursor.execute(query, tuple(row))

conn.commit()
cursor.close()
conn.close()

print("Data inserted successfully!")
```

```
Data inserted successfully!
```

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

Out[4]:

| | Score | Popularity | Rank | Members | Description | Synonyms | Name | Type | Epis |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 9.38 | 284 | 1 | 710 | During their decade-long quest to defeat the D... | Frieren at the Funeral | Frieren: Beyond Journey's End | TV | |
| **1** | 9.09 | 3 | 2 | 3 | After a horrific alchemy experiment goes wrong... | Hagane no Renkinjutsushi: Fullmetal Alchemist,... | Fullmetal Alchemist: Brotherhood | TV | |
| **2** | 9.07 | 13 | 3 | 2 | Eccentric scientist Rintarou Okabe has a never... | NaN | Steins;Gate | TV | |
| **3** | 9.06 | 342 | 4 | 630 | Gintoki, Shinpachi, and Kagura return as the f... | Gintama' (2015) | Gintama Season 4 | TV | |
| **4** | 9.05 | 21 | 5 | 2 | Seeking to restore humanity's diminishing hope... | NaN | Attack on Titan Season 3 Part 2 | TV | |

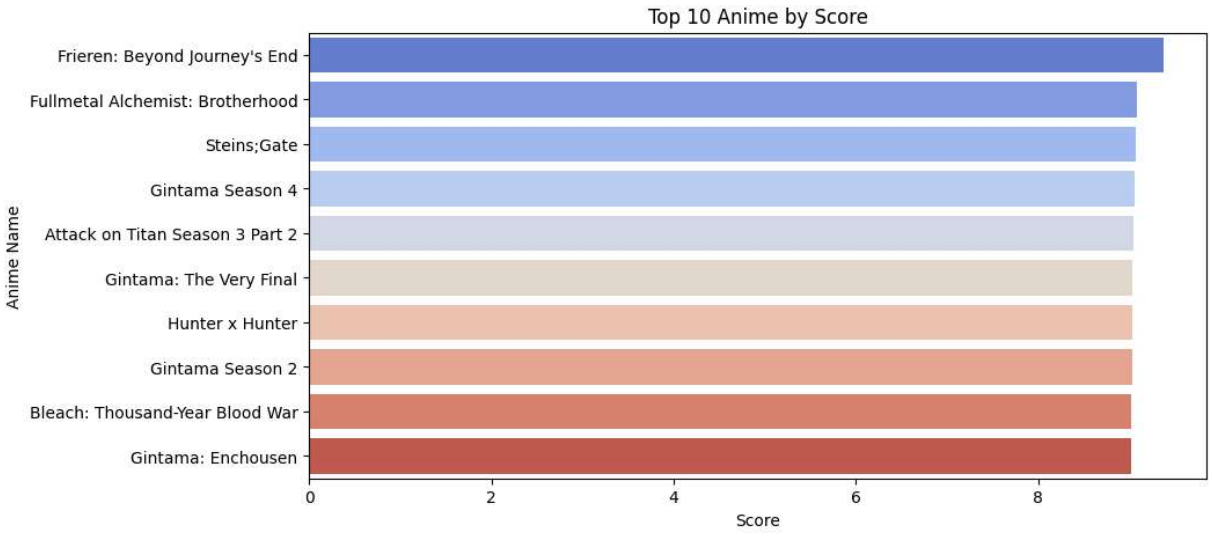
5 rows × 21 columns

In [6]:
```
df.shape
```

Out[6]: (1000, 21)

In [7]:
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# MySQL Connection
db_config = {
    "host": "localhost",
    "user": "Satyen",
    "password": "Satyen@789",
    "database": "Bakugo"
}

conn = mysql.connector.connect(**db_config)
query = "SELECT DISTINCT Name, Score FROM Katsuki ORDER BY Score DESC LIMIT 10"
df = pd.read_sql(query, conn)
conn.close()

# Plot
plt.figure(figsize=(10, 5))
sns.barplot(x="Score", y="Name", data=df, hue="Name", palette="coolwarm")
plt.xlabel("Score")
plt.ylabel("Anime Name")
plt.title("Top 10 Anime by Score")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_11896\593053914.py:16: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df = pd.read_sql(query, conn)
```

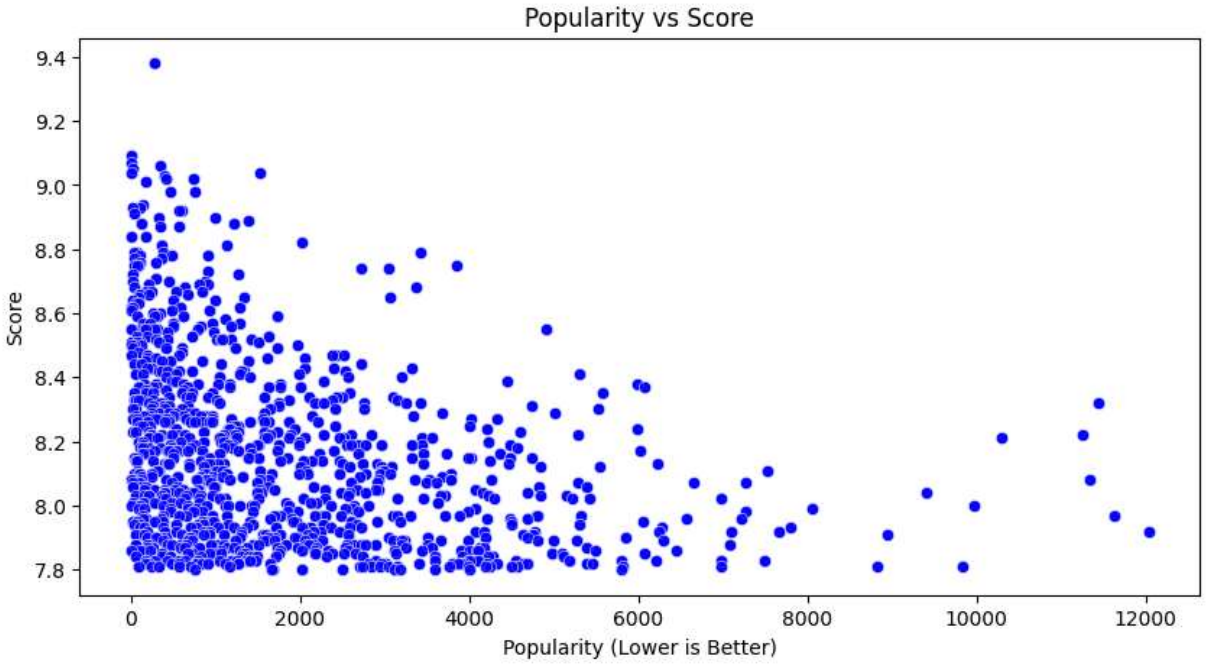



In [ ]:

In [5]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Popularity, Score FROM Katsuki"
df = pd.read_sql(query, conn)
```

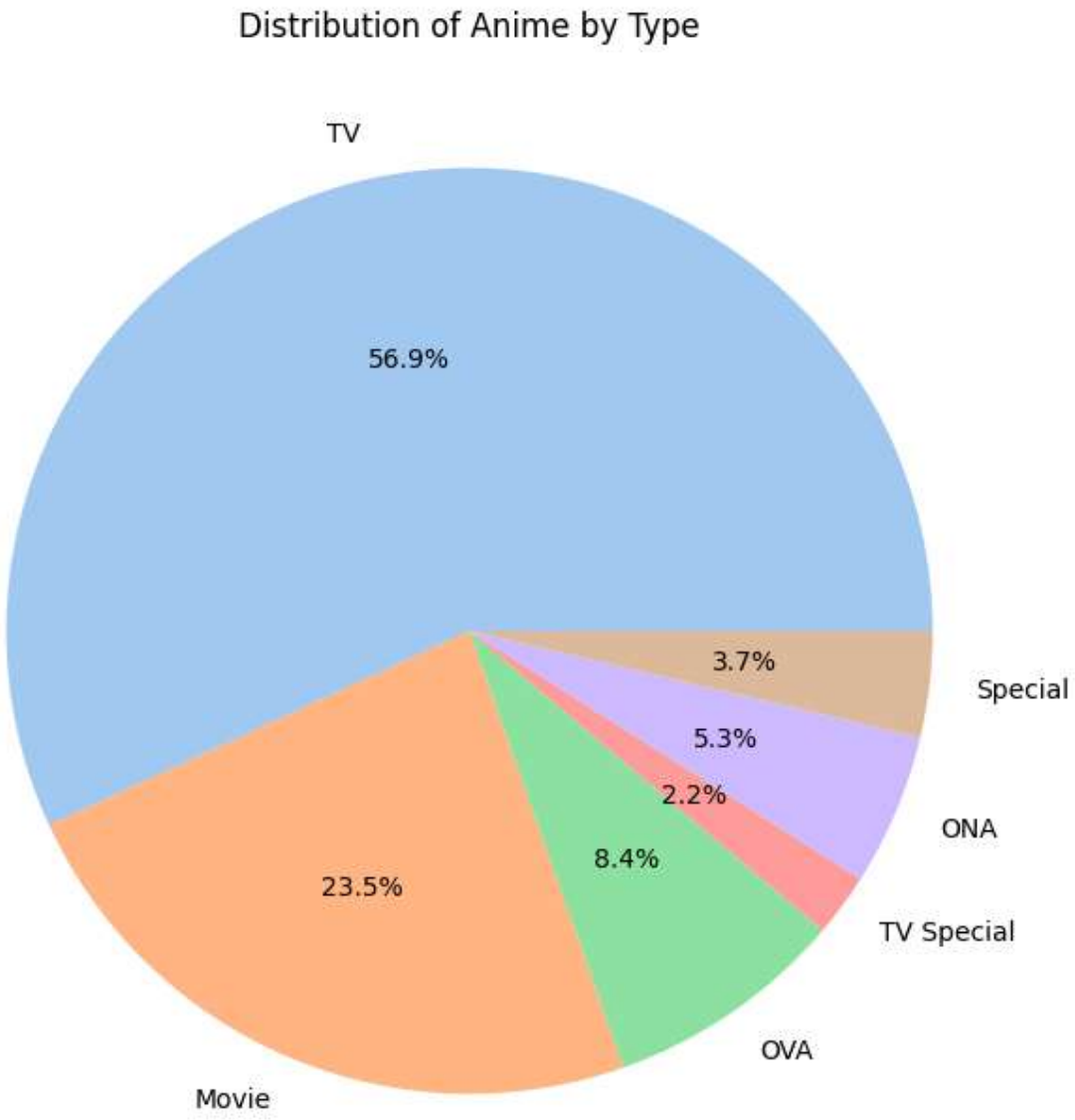
```
conn.close()

# Plot
plt.figure(figsize=(10, 5))
sns.scatterplot(x=df["Popularity"], y=df["Score"], color="blue", alpha=0.7)
plt.xlabel("Popularity (Lower is Better)")
plt.ylabel("Score")
plt.title("Popularity vs Score")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\509813824.py:3: UserWarning: panda
s only supports SQLAlchemy connectable (engine/connection) or database string URI or
sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider usin
g SQLAlchemy.
  df = pd.read_sql(query, conn)
```



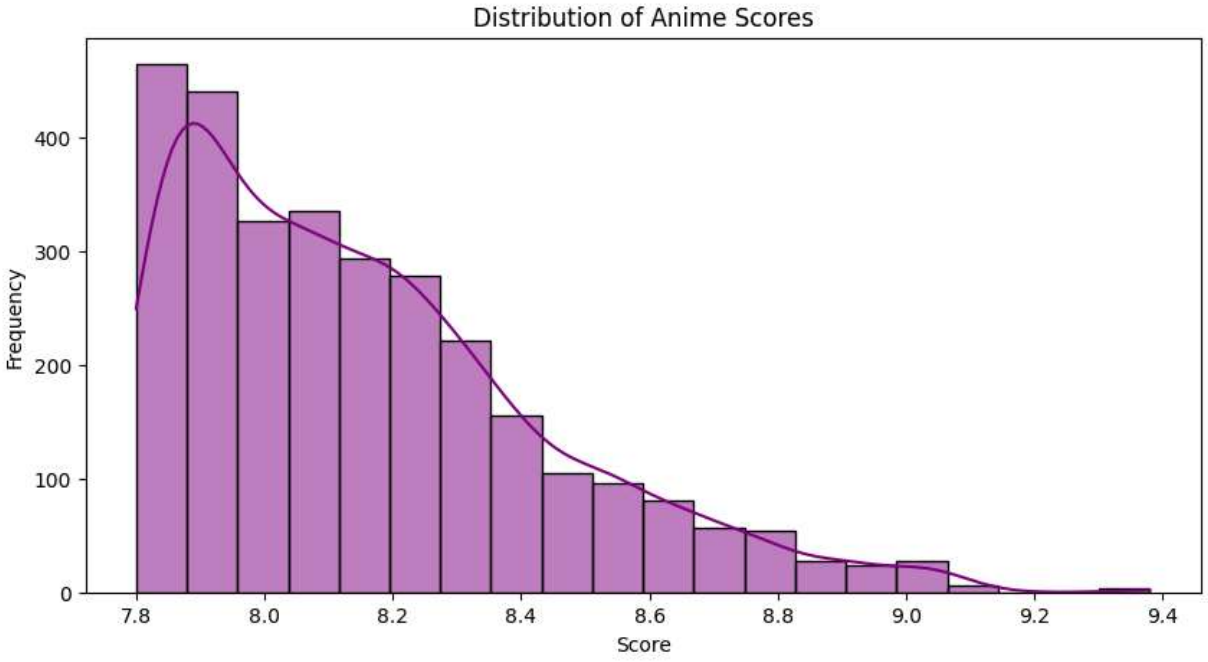

In [6]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Type, COUNT(*) as Count FROM Katsuki GROUP BY Type"
df = pd.read_sql(query, conn)
conn.close()

# Plot
plt.figure(figsize=(8, 8))
plt.pie(df["Count"], labels=df["Type"], autopct="%1.1f%%", colors=sns.color_palette
plt.title("Distribution of Anime by Type")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\1780068210.py:3: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df = pd.read_sql(query, conn)
```



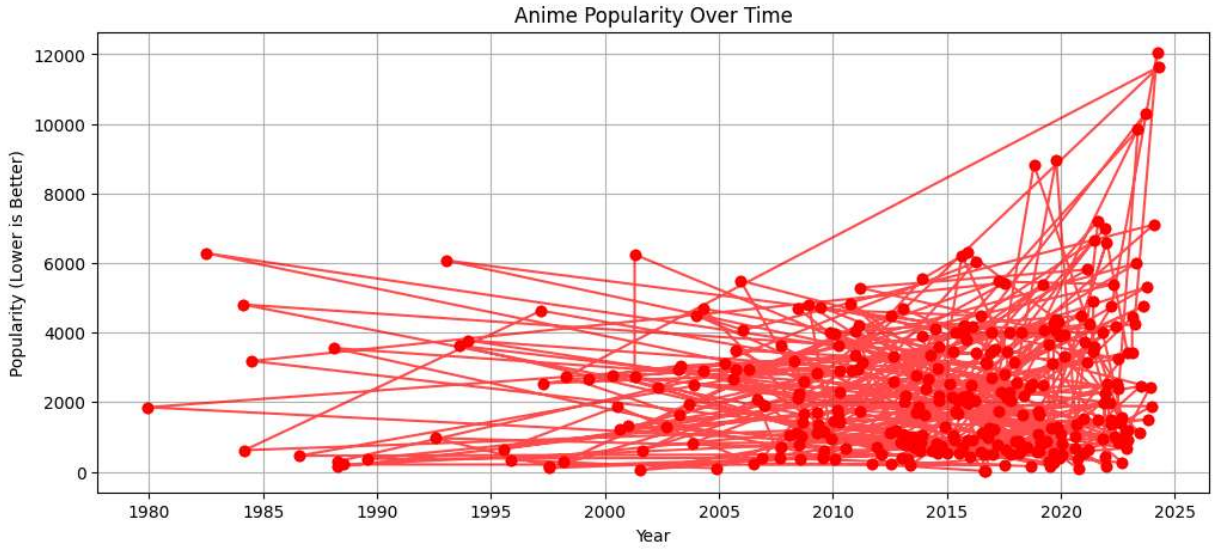

In [7]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Score FROM Katsuki"
df = pd.read_sql(query, conn)
conn.close()

# Plot
plt.figure(figsize=(10, 5))
sns.histplot(df["Score"], bins=20, kde=True, color="purple")
plt.xlabel("Score")
plt.ylabel("Frequency")
plt.title("Distribution of Anime Scores")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\362730680.py:3: UserWarning: panda
s only supports SQLAlchemy connectable (engine/connection) or database string URI or
sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider usin
g SQLAlchemy.
  df = pd.read_sql(query, conn)
```



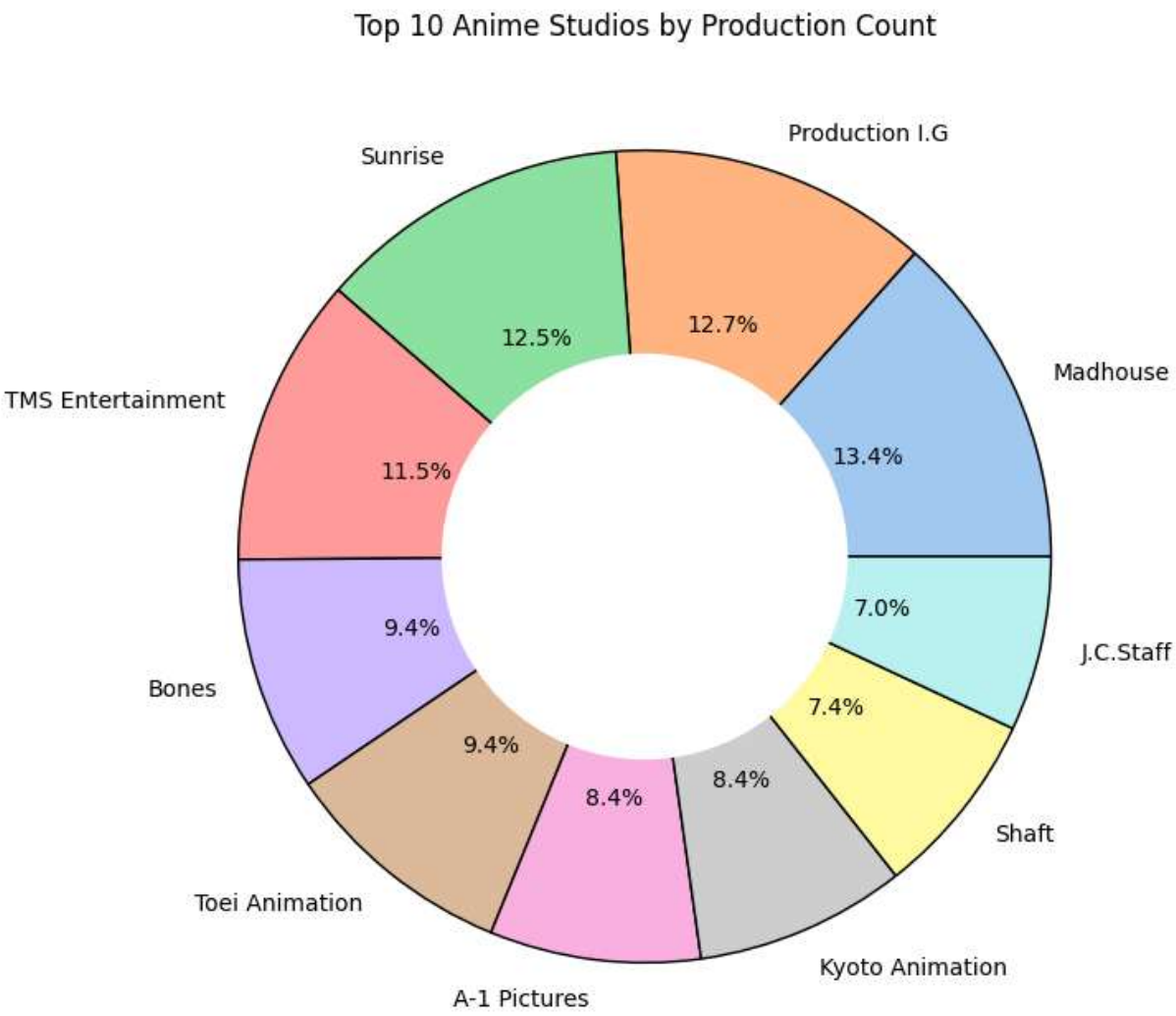

In [9]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Aired, Popularity FROM Katsuki WHERE Aired IS NOT NULL ORDER BY Air
df = pd.read_sql(query, conn)
conn.close()

# Convert Aired column to datetime
df["Aired"] = pd.to_datetime(df["Aired"], errors='coerce')
df = df.dropna()

# Plot
plt.figure(figsize=(12, 5))
plt.plot(df["Aired"], df["Popularity"], marker='o', linestyle='-', color='red', alp
plt.xlabel("Year")
plt.ylabel("Popularity (Lower is Better)")
plt.title("Anime Popularity Over Time")
plt.grid()
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\782699659.py:3: UserWarning: panda
s only supports SQLAlchemy connectable (engine/connection) or database string URI or
sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider usin
g SQLAlchemy.
  df = pd.read_sql(query, conn)
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\782699659.py:7: UserWarning: Could
not infer format, so each element will be parsed individually, falling back to `date
util`. To ensure parsing is consistent and as-expected, please specify a format.
  df["Aired"] = pd.to_datetime(df["Aired"], errors='coerce')
```



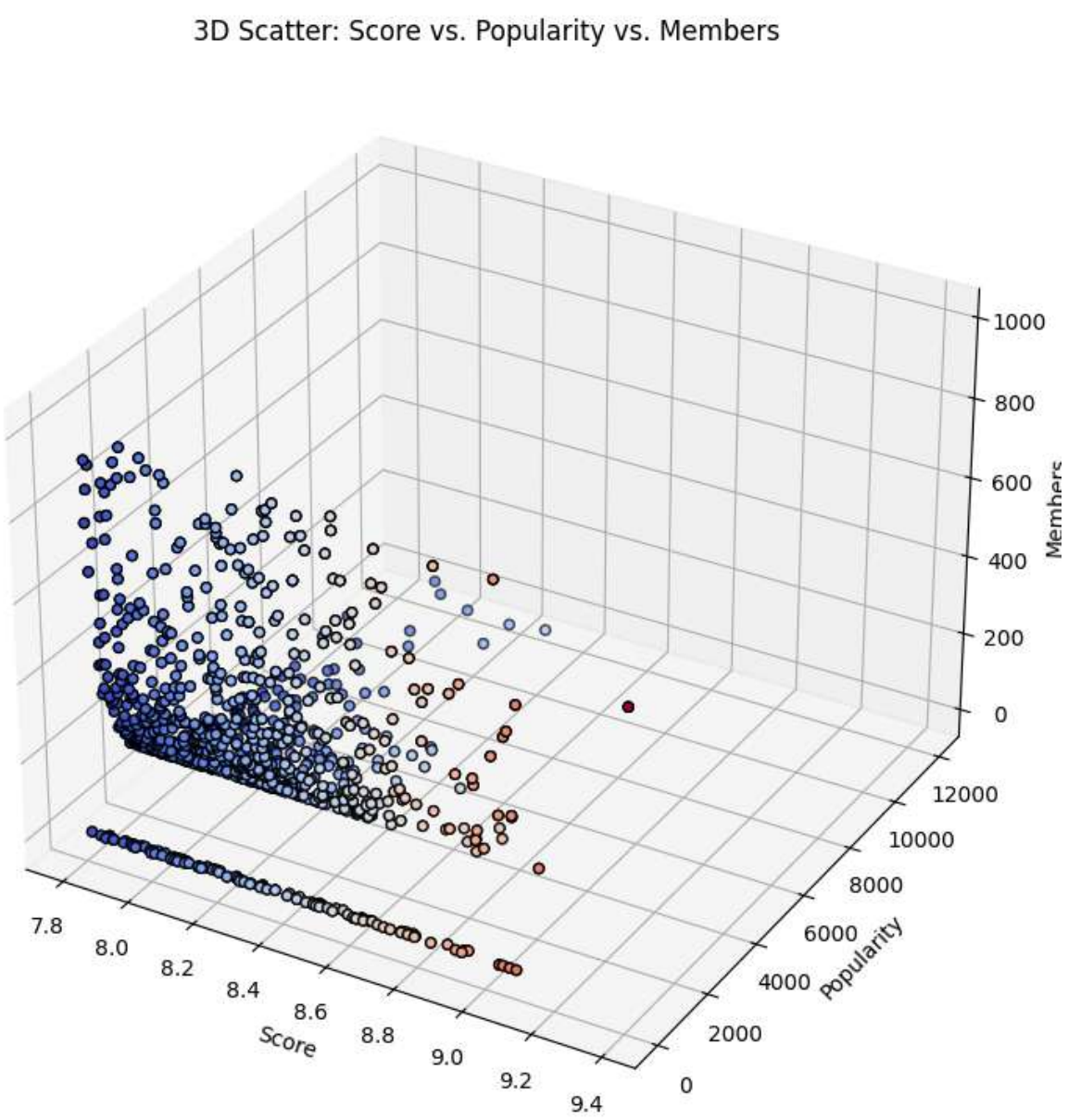

In [15]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Studios, COUNT(*) as Count FROM Katsuki GROUP BY Studios ORDER BY C
df = pd.read_sql(query, conn)
conn.close()

# Plot
plt.figure(figsize=(8, 8))
plt.pie(df["Count"], labels=df["Studios"], autopct="%1.1f%%", colors=sns.color_pale
plt.gca().add_artist(plt.Circle((0,0),0.5,fc='white'))  # Donut hole
plt.title("Top 10 Anime Studios by Production Count")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\4004705431.py:3: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df = pd.read_sql(query, conn)
```



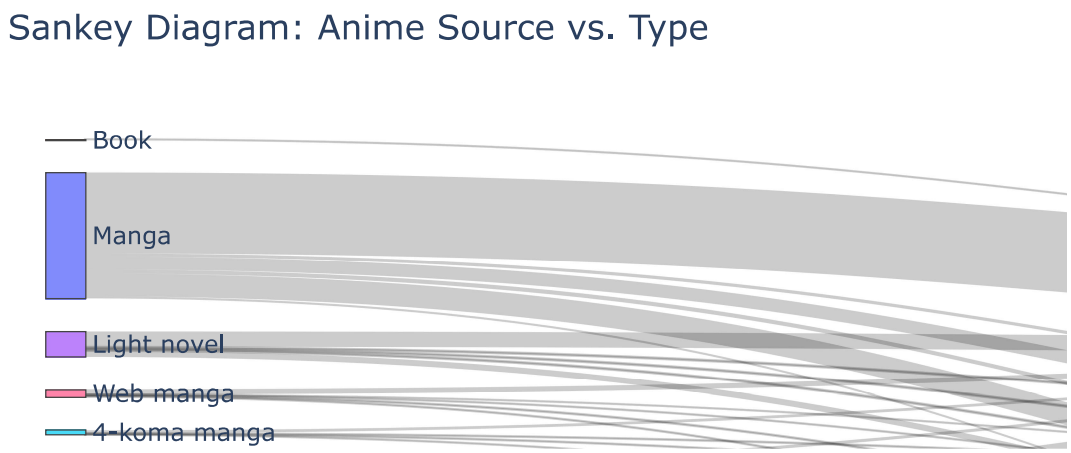

In [45]:
```
from mpl_toolkits.mplot3d import Axes3D

conn = mysql.connector.connect(**db_config)
query = "SELECT Score, Popularity, Members FROM Katsuki"
df = pd.read_sql(query, conn)
conn.close()

fig = plt.figure(figsize=(14, 9))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df["Score"], df["Popularity"], df["Members"], c=df["Score"], cmap="coolw

ax.set_xlabel("Score")
ax.set_ylabel("Popularity")
ax.set_zlabel("Members")
ax.set_title("3D Scatter: Score vs. Popularity vs. Members")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\2226309509.py:5: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df = pd.read_sql(query, conn)
```




In [97]:
```
import plotly.graph_objects as go

conn = mysql.connector.connect(**db_config)
query = "SELECT Source, Type, COUNT(*) as Count FROM Katsuki GROUP BY Source, Type"
df = pd.read_sql(query, conn)
conn.close()

source_labels = list(df["Source"].unique())
type_labels = list(df["Type"].unique())
labels = source_labels + type_labels

source_indices = [labels.index(src) for src in df["Source"]]
target_indices = [labels.index(typ) for typ in df["Type"]]

fig = go.Figure(go.Sankey(node={"label": labels},
                          link={"source": source_indices, "target": target_indices,
fig.update_layout(title_text="Sankey Diagram: Anime Source vs. Type")
fig.show()
```

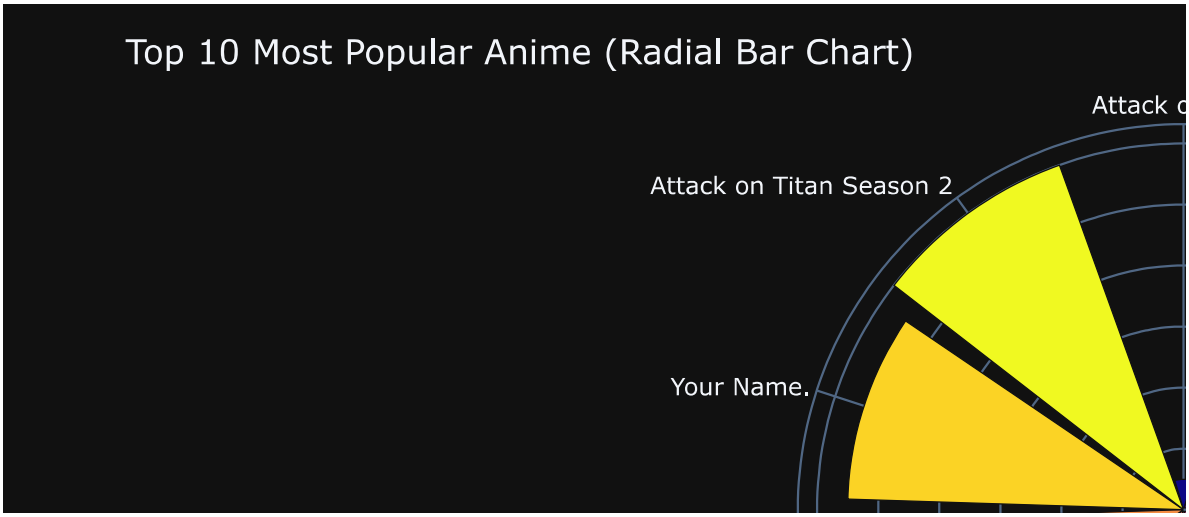
```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\3094127980.py:5: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```




In [107…

```
import plotly.express as px

conn = mysql.connector.connect(**db_config)
query = "SELECT DISTINCT Name, Popularity FROM Katsuki ORDER BY Popularity ASC LIMI
df = pd.read_sql(query, conn)
conn.close()

fig = px.bar_polar(df, r="Popularity", theta="Name", color="Popularity", template="
fig.update_layout(title="Top 10 Most Popular Anime (Radial Bar Chart)")
fig.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\272402436.py:5: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```

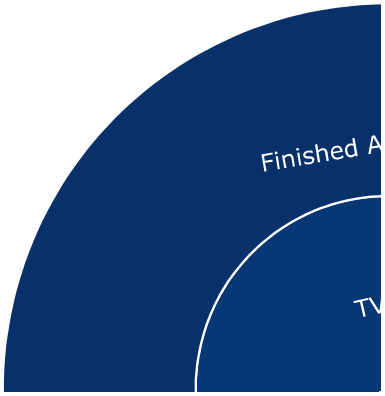



In [108…

```
import plotly.express as px

conn = mysql.connector.connect(**db_config)
query = "SELECT Type, Status, COUNT(*) as Count FROM Katsuki GROUP BY Type, Status"
df = pd.read_sql(query, conn)
conn.close()

fig = px.sunburst(df, path=["Type", "Status"], values="Count", color="Count", color
fig.update_layout(title="Anime Distribution by Type and Status")
fig.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\252164522.py:5: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```

Anime Distribution by Type and Status



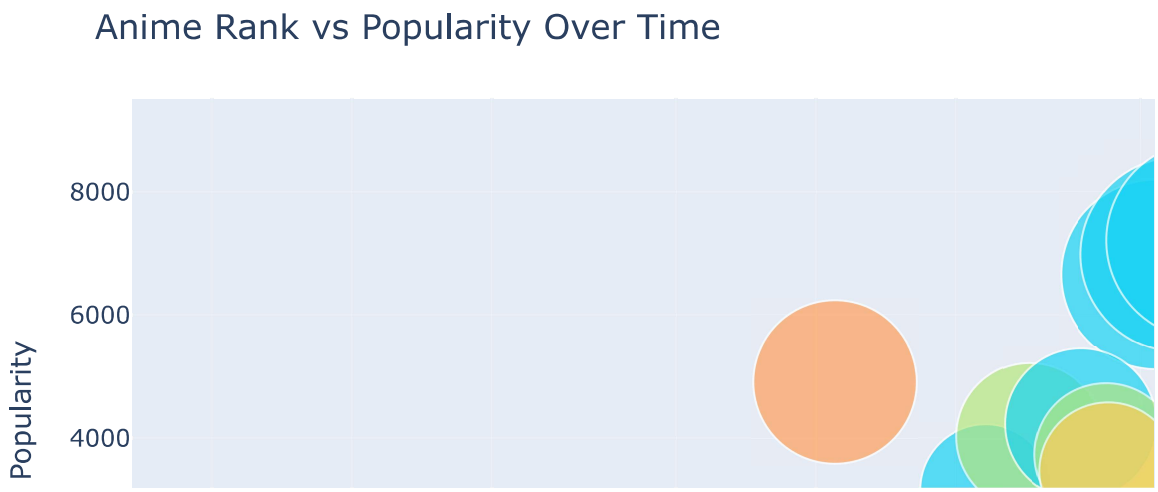

In [125…

```
import mysql.connector
import pandas as pd
import plotly.express as px
conn = mysql.connector.connect(**db_config)

# Fetch Data
query = "SELECT DISTINCT Name, `Rank`, Popularity, Aired FROM Katsuki WHERE Aired I
df = pd.read_sql_query(query, conn)
conn.close()

# Convert Data Types
df["Aired"] = pd.to_datetime(df["Aired"], errors="coerce")
df.dropna(subset=["Aired"], inplace=True)  # Remove NaN values
df["Aired"] = df["Aired"].dt.year          # Keep only the year
df["Rank"] = pd.to_numeric(df["Rank"], errors="coerce")  # Convert Rank to number

# Debug: Check unique years
print(df["Aired"].unique())  # Ensure multiple years exist

# Create Animated Scatter Plot
fig = px.scatter(df, x="Rank", y="Popularity", animation_frame="Aired",
                 size="Popularity", color="Name", hover_name="Name",
                 log_x=True, size_max=90)
```

```
fig.update_layout(title="Anime Rank vs Popularity Over Time")
fig.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\4110817274.py:8: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.

C:\Users\satye\AppData\Local\Temp\ipykernel_19440\4110817274.py:12: UserWarning:

Could not infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a forma
t.
```

```
[2021 2016 2013 2020 2022 2017 2001 2004 1997 2010 2015 2019 2009 2012
 2018 1998 2014 2008 1988 2023 1993 1984 2011 2002 2003 1995 2007 1986
 1989 2000 1979 2006 2005 1999 2024 1992 1982]
```

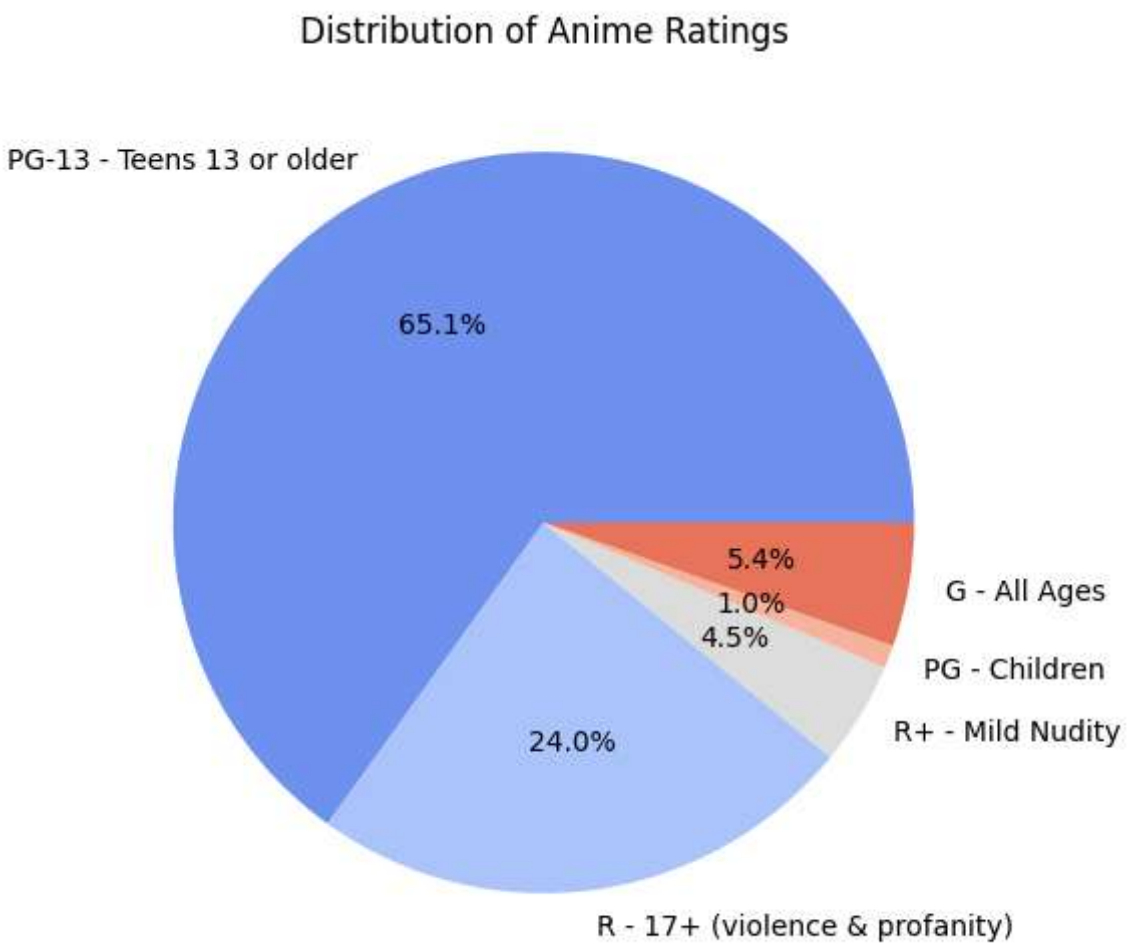



In [126…

```
conn = mysql.connector.connect(**db_config)
query = "SELECT Rating, COUNT(*) as Count FROM Katsuki WHERE Rating IS NOT NULL GRO
df = pd.read_sql(query, conn)
conn.close()

plt.figure(figsize=(10, 6))
```

```
plt.pie(df["Count"], labels=df["Rating"], autopct="%1.1f%%", colors=sns.color_palet
plt.title("Distribution of Anime Ratings")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\1626137058.py:3: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```

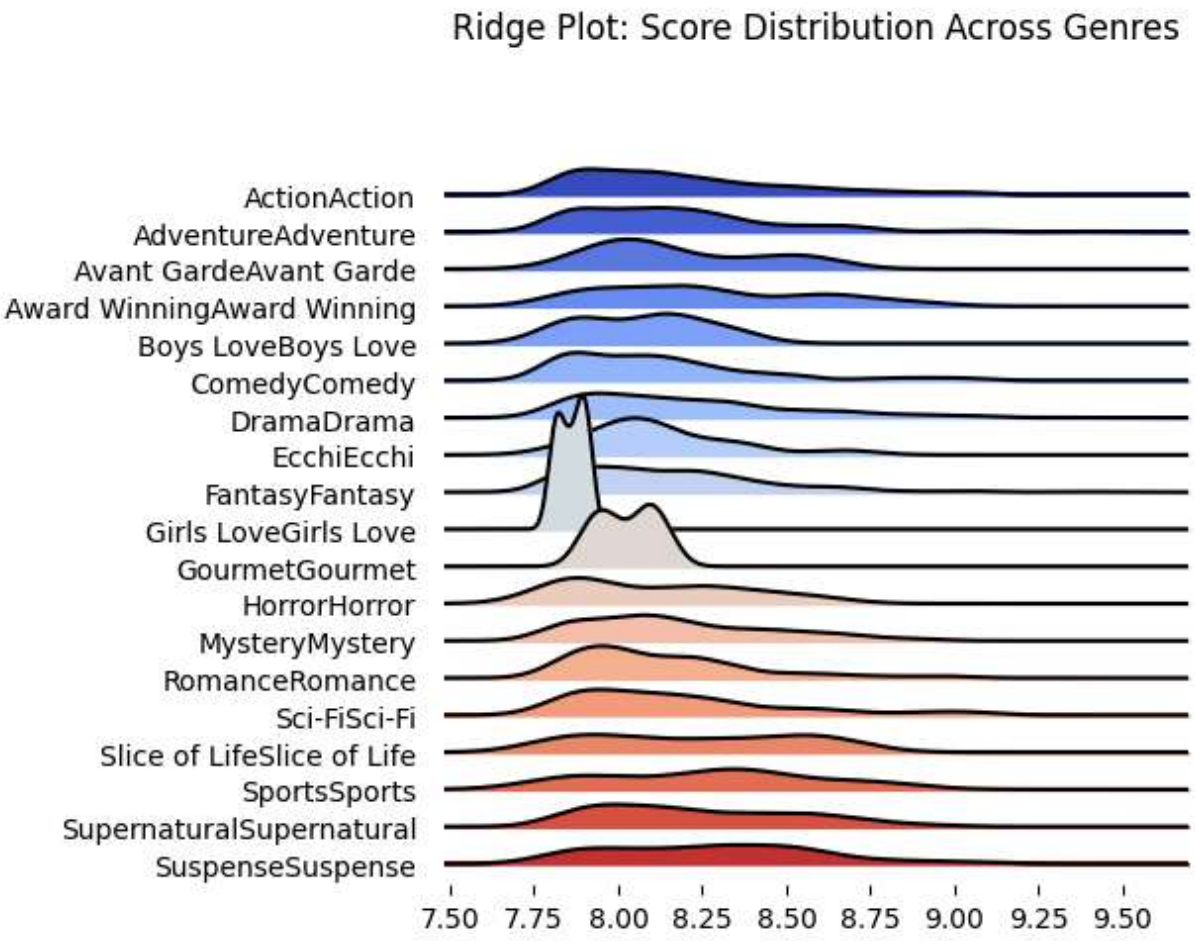



In [131…

```
import joypy

conn = mysql.connector.connect(**db_config)
query = "SELECT Genres, Score FROM Katsuki WHERE Score > 0"
df = pd.read_sql(query, conn)
conn.close()

df["Genres"] = df["Genres"].str.split(", ")
df = df.explode("Genres")

plt.figure(figsize=(17, 8))
joypy.joyplot(data=df, by="Genres", column="Score", colormap=plt.cm.coolwarm)
plt.title("Ridge Plot: Score Distribution Across Genres")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\2877176961.py:5: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```

```
<Figure size 1700x800 with 0 Axes>
```

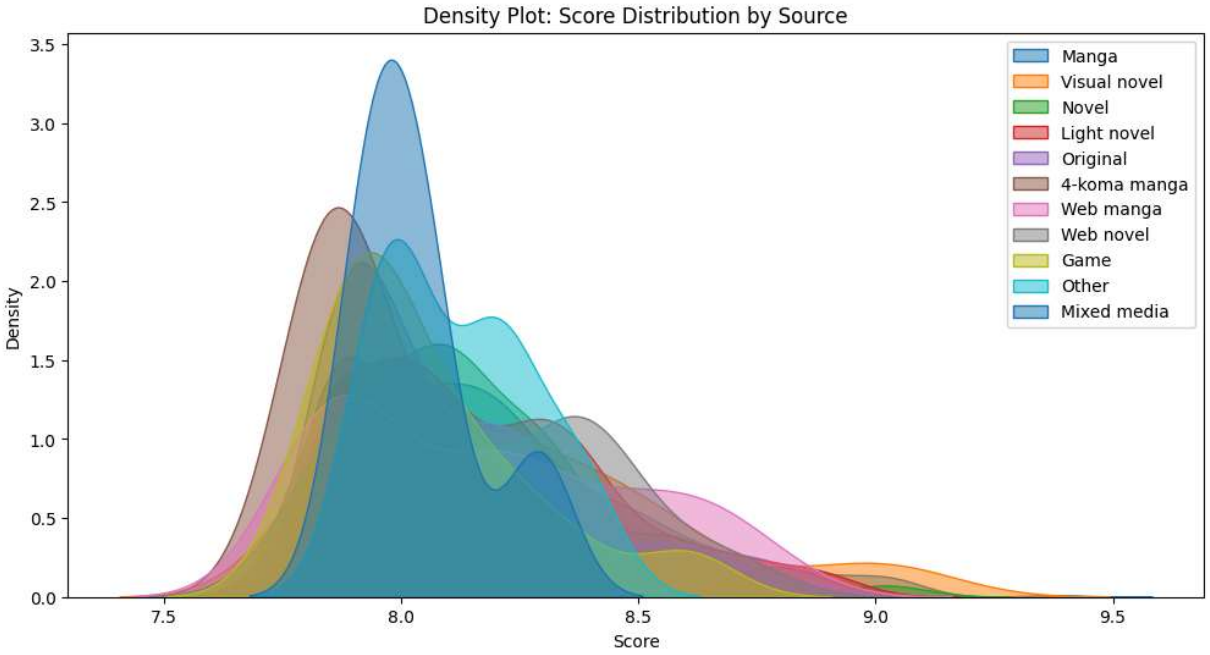



In [138…

```python
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Connect to Database
conn = mysql.connector.connect(**db_config)
query = "SELECT Source, Score FROM Katsuki WHERE Score > 0 AND Source IS NOT NULL"
df = pd.read_sql(query, conn)
conn.close()

# Identify and Remove Zero Variance Groups
df_var = df.groupby("Source")["Score"].std()  # Compute standard deviation
valid_demos = df_var[df_var > 0].index  # Keep only those with variance
df = df[df["Source"].isin(valid_demos)]  # Filter dataframe

# Plot KDE
plt.figure(figsize=(12, 6))
for demo in df["Source"].unique():
```

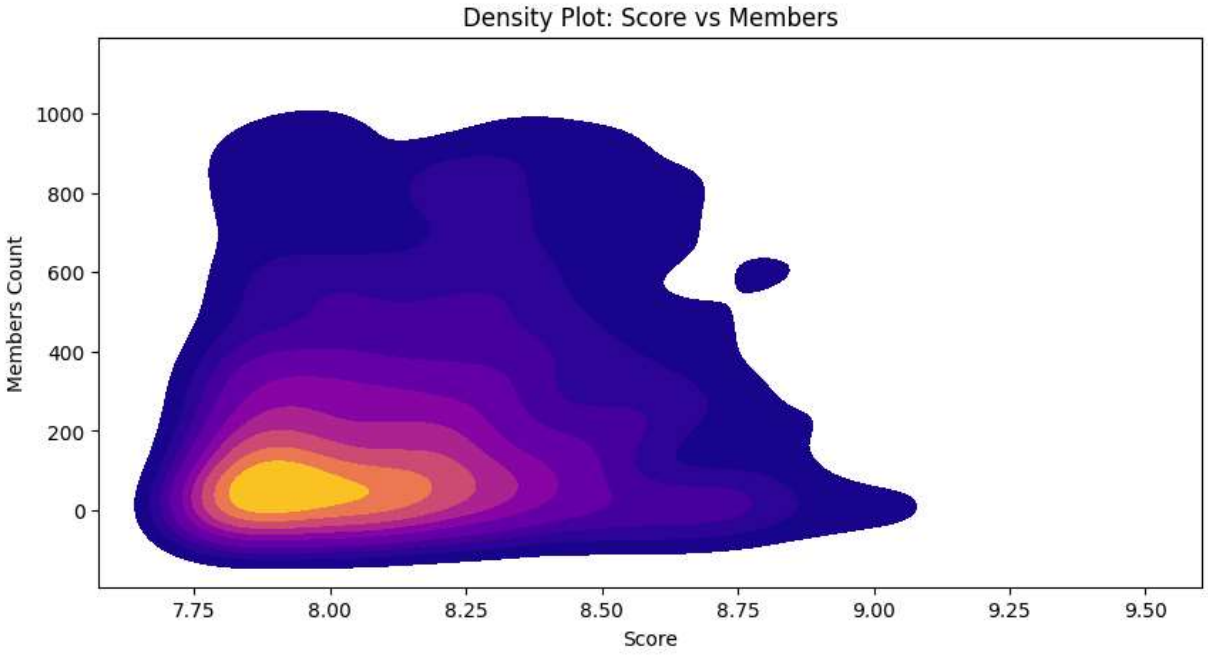
```
    subset = df[df["Source"] == demo]
    sns.kdeplot(subset["Score"], label=demo, fill=True, alpha=0.5)

plt.xlabel("Score")
plt.ylabel("Density")
plt.title("Density Plot: Score Distribution by Source")
plt.legend()
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\1209186137.py:9: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```




In [91]:
```
conn = mysql.connector.connect(**db_config)
query = "SELECT Score, Members FROM Katsuki WHERE Score > 0"
df = pd.read_sql(query, conn)
conn.close()

plt.figure(figsize=(10, 5))
sns.kdeplot(x=df["Score"], y=df["Members"], cmap="plasma", fill=True)
plt.xlabel("Score")
plt.ylabel("Members Count")
plt.title("Density Plot: Score vs Members")
plt.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\4130330793.py:3: UserWarning: pand
as only supports SQLAlchemy connectable (engine/connection) or database string URI o
r sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please consider us
ing SQLAlchemy.
  df = pd.read_sql(query, conn)
```




In [145…

```
import plotly.express as px

conn = mysql.connector.connect(**db_config)
query = "SELECT Source, COUNT(*) as Count FROM Katsuki WHERE Source IS NOT NULL GRO
df = pd.read_sql(query, conn)
conn.close()

fig = px.treemap(df, path=["Source"], values="Count", color="Count", color_continuo
fig.update_layout(title="Treemap: Anime Count by Source")
fig.show()
```

```
C:\Users\satye\AppData\Local\Temp\ipykernel_19440\2624610599.py:5: UserWarning:

pandas only supports SQLAlchemy connectable (engine/connection) or database string U
RI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are not tested. Please conside
r using SQLAlchemy.
```

## Treemap: Anime Count by Source


In [8]:
```
#Conclusion
```

In [ ]:
```
'''Conclusions from Anime Data Analysis---

- The highest-rated anime tend to stand out in popularity, reflecting audience pref
- Popularity and ratings do not always go hand in hand, though some highly-rated an
- Anime scores follow a distinct distribution, with certain rating ranges being mor
- Audience preferences evolve over time, with noticeable trends in popularity acros
- A few top studios dominate anime production, consistently delivering a high volum
- There is a clear relationship between an anime’s score, popularity, and audience
- Different anime sources (manga, light novels, etc.) influence the type of content
- The most popular anime stand out significantly, with a few titles consistently le
- Anime formats and their completion status vary widely, indicating differences in
- Anime rankings and popularity shift over time, reflecting changing audience taste

This analysis provides valuable insights into anime trends, audience behavior, and
```

In [ ]:
```
#Future Prediction
```

In [ ]:
```
'''
Future Based on the Analysis--

Evolving Genre Preferences- As audience tastes shift over time, genres like isekai,
```

```
                                may continue to dominate, while newer themes could eme

Increased Studio Competition- With a few studios producing most of the content, new
                                  and differentiate to gain traction in the industry.

Shifting Popularity Dynamics- The correlation between scores and popularity suggest
                                  through improved marketing and global distribution.

Content Adaptation Trends- The influence of different source materials (manga, ligh
                                with streaming platforms driving demand for diverse st


By looking at these insights, the anime industry can anticipate trends and adapt to

'''
```

In [ ]: