In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
movies=pd.read_csv("movies.csv")
```

In [3]:

```
movies.head()
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [4]:

```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62423 entries, 0 to 62422
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   movieId  62423 non-null  int64 
 1   title    62423 non-null  object
 2   genres   62423 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.4+ MB
```

In [5]:

```
movies.isnull().sum()
```

Out[5]:

```
movieId    0
title      0
genres     0
dtype: int64
```

In [6]:

```
movies.describe()
```

Out[6]:

	movieId
count	62423.000000
mean	122220.387646
std	63264.744844
min	1.000000
25%	82146.500000
50%	138022.000000
75%	173222.000000
max	209171.000000

In [7]:

```
ratings=pd.read_csv("ratings.csv")
```

In [8]:

```
ratings.head()
```

Out[8]:

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510

In [9]:

```
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331701 entries, 0 to 331700
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   userId     331701 non-null  int64  
 1   movieId    331701 non-null  int64  
 2   rating     331701 non-null  float64
 3   timestamp  331701 non-null  int64  
dtypes: float64(1), int64(3)
memory usage: 10.1 MB
```

In [10]:

```
ratings.isnull().sum()
```

Out[10]:

```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

In [11]:

```
ratings.describe()
```

Out[11]:

	userId	movieId	rating	timestamp
count	331701.000000	331701.000000	331701.000000	3.317010e+05
mean	1178.344168	20858.639688	3.548551	1.208615e+09
std	652.540007	38707.888770	1.055842	2.334659e+08
min	1.000000	1.000000	0.500000	7.896520e+08
25%	626.000000	1132.000000	3.000000	9.923919e+08
50%	1185.000000	2791.000000	4.000000	1.182966e+09
75%	1748.000000	8360.000000	4.000000	1.446621e+09
max	2298.000000	208793.000000	5.000000	1.574254e+09

In [12]:

```
print(movies.duplicated().sum())
print(ratings.duplicated().sum())
```

```
0
0
```

In [13]:

```
 #Extract year from title
movies['year'] = movies['title'].str.extract(r'\((\d{4})\)', expand=False)
```

In [14]:

```
# Convert 'year' to integer
movies['year'] = movies['year'].dropna().astype(int)
```

In [15]:

```
# Merge movies and ratings on movieId
data = pd.merge(movies, ratings, on='movieId')
```

In [16]:
```
data
```

Out[16]:

	movieId	title	genres	year	userId	rating	timestam
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	2	3.5	11414158
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	3	4.0	14394722
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	4	3.0	15739442
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	5	4.0	8586259
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	8	4.0	8904925
...	...	...	...	...	...	...	
331696	207309	Fractured (2019)	Thriller	2019.0	1068	3.0	15716104
331697	207309	Fractured (2019)	Thriller	2019.0	2290	5.0	15716760
331698	207642	Kabir Singh (2019)	Action|Drama|Romance	2019.0	2290	5.0	15716202
331699	208002	The Kill Team (2019)	Drama|War	2019.0	973	3.5	15723640
331700	208793	Watchman (2019)	Drama|Thriller	2019.0	1652	3.5	15735908

331701 rows × 7 columns

In [17]:
```python
from sklearn.preprocessing import LabelEncoder

# Encoding movie titles
le = LabelEncoder()
data['title'] = le.fit_transform(data['title'])
```

In [18]:
```python
# genres were separated by '|', first split them
movies['genres'] = movies['genres'].str.split('|')
movies_exploded = movies.explode('genres')

# Merge exploded genres with ratings
data = pd.merge(movies_exploded, ratings, on='movieId')

# One-Hot Encoding
data = pd.get_dummies(data, columns=['genres'])
```

In [19]:
```python
# Fit and transform the 'title' column
data['title'] = le.fit_transform(data['title'])
```

In [20]:

```
data
```

Out[20]:

	movieId	title	year	userId	rating	timestamp	genres_(no genres listed)	genres_Action	genres_Adventu
0	1	14205	1995.0	2	3.5	1141415820	0	0	
1	1	14205	1995.0	3	4.0	1439472215	0	0	
2	1	14205	1995.0	4	3.0	1573944252	0	0	
3	1	14205	1995.0	5	4.0	858625949	0	0	
4	1	14205	1995.0	8	4.0	890492517	0	0	
...	...	...	...	...	...	...	...	...	
899588	207642	7219	2019.0	2290	5.0	1571620212	0	0	
899589	208002	13509	2019.0	973	3.5	1572364057	0	0	
899590	208002	13509	2019.0	973	3.5	1572364057	0	0	
899591	208793	14916	2019.0	1652	3.5	1573590803	0	0	
899592	208793	14916	2019.0	1652	3.5	1573590803	0	0	

899593 rows × 26 columns

In [21]:

```
from sklearn.preprocessing import StandardScaler

# Initialize scaler
scaler = StandardScaler()

# Scaling
data[['rating', 'year']] = scaler.fit_transform(data[['rating', 'year']])
```

In [22]:

```
data
```

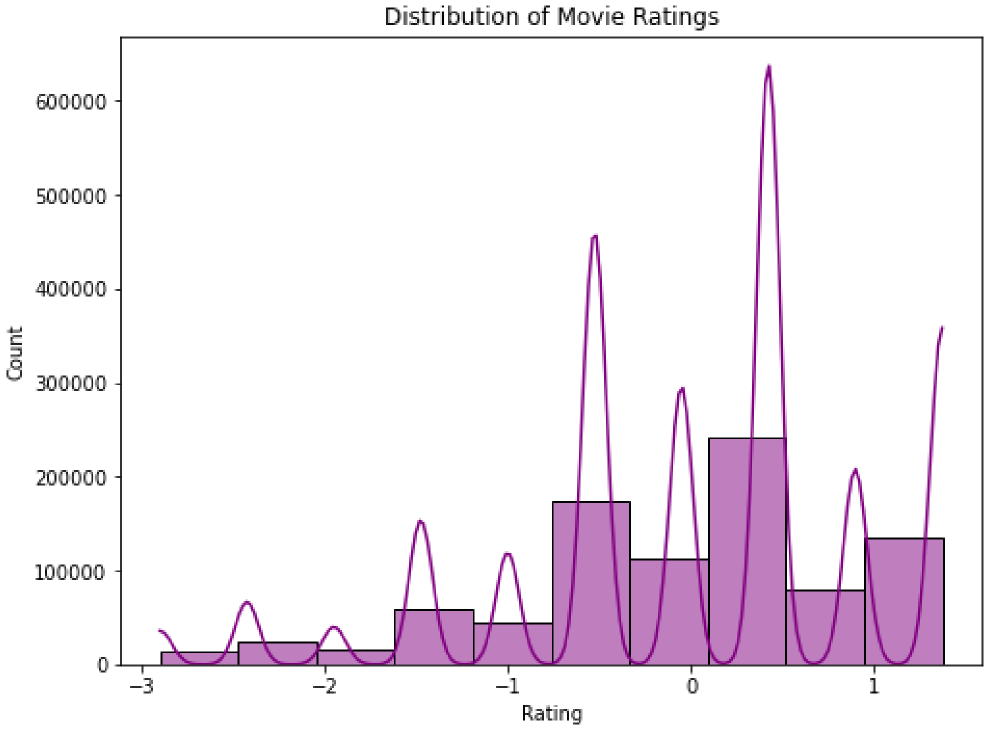
Out[22]:

	movieId	title	year	userId	rating	timestamp	genres_(no genres listed)	genres_Action	genres_Ad
0	1	14205	0.014773	2	-0.052442	1141415820	0	0	
1	1	14205	0.014773	3	0.421775	1439472215	0	0	
2	1	14205	0.014773	4	-0.526659	1573944252	0	0	
3	1	14205	0.014773	5	0.421775	858625949	0	0	
4	1	14205	0.014773	8	0.421775	890492517	0	0	
...	...	...	...	...	...	...	...	...	
899588	207642	7219	1.641360	2290	1.370209	1571620212	0	0	
899589	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899590	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899591	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	
899592	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	

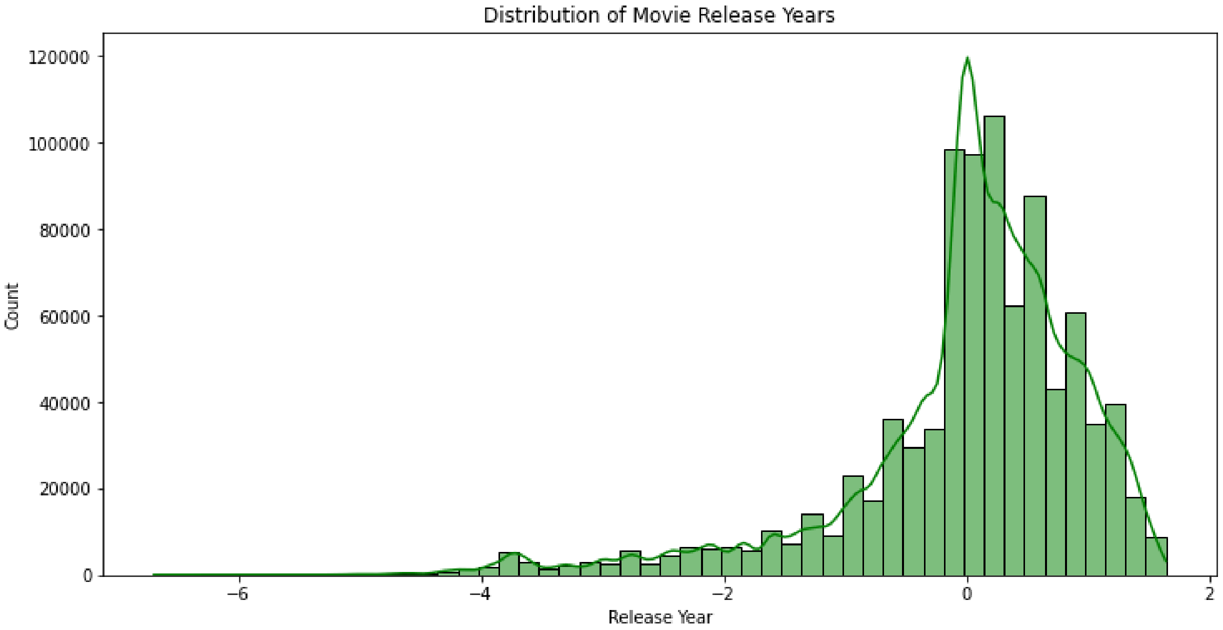
899593 rows × 26 columns

In [23]:

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

# Plot Rating distribution
plt.figure(figsize=(8,6))
sns.histplot(data['rating'], bins=10, kde=True, color='purple')
plt.title('Distribution of Movie Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```

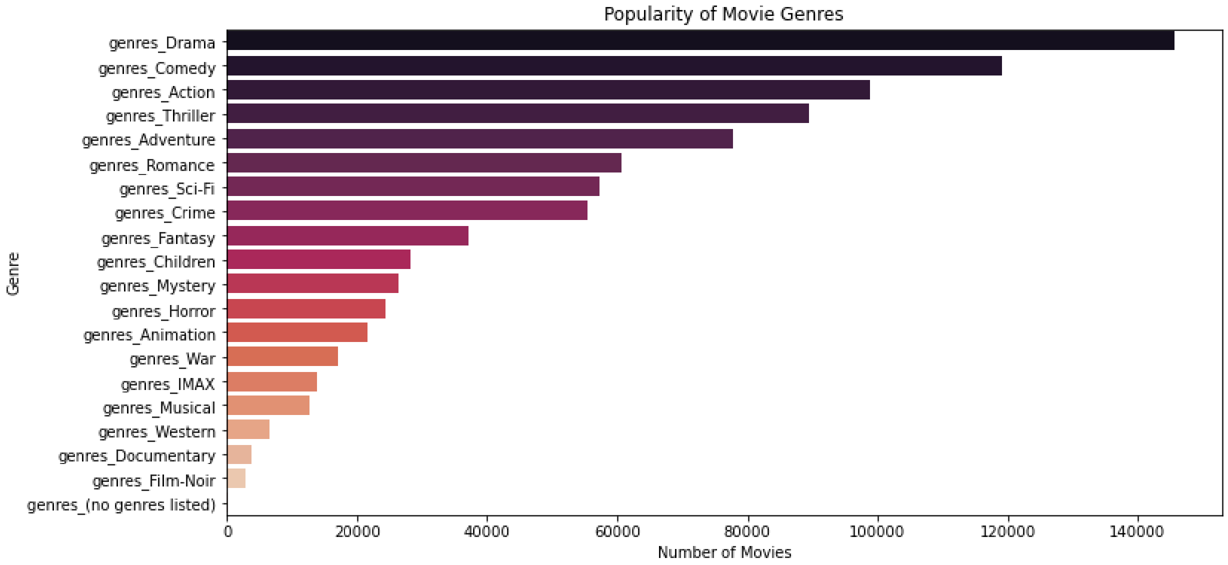


```
In [24]:  #Plot year distribution
         plt.figure(figsize=(12,6))
         sns.histplot(data['year'], bins=50, kde=True, color='green')
         plt.title('Distribution of Movie Release Years')
         plt.xlabel('Release Year')
         plt.ylabel('Count')
         plt.show()
```



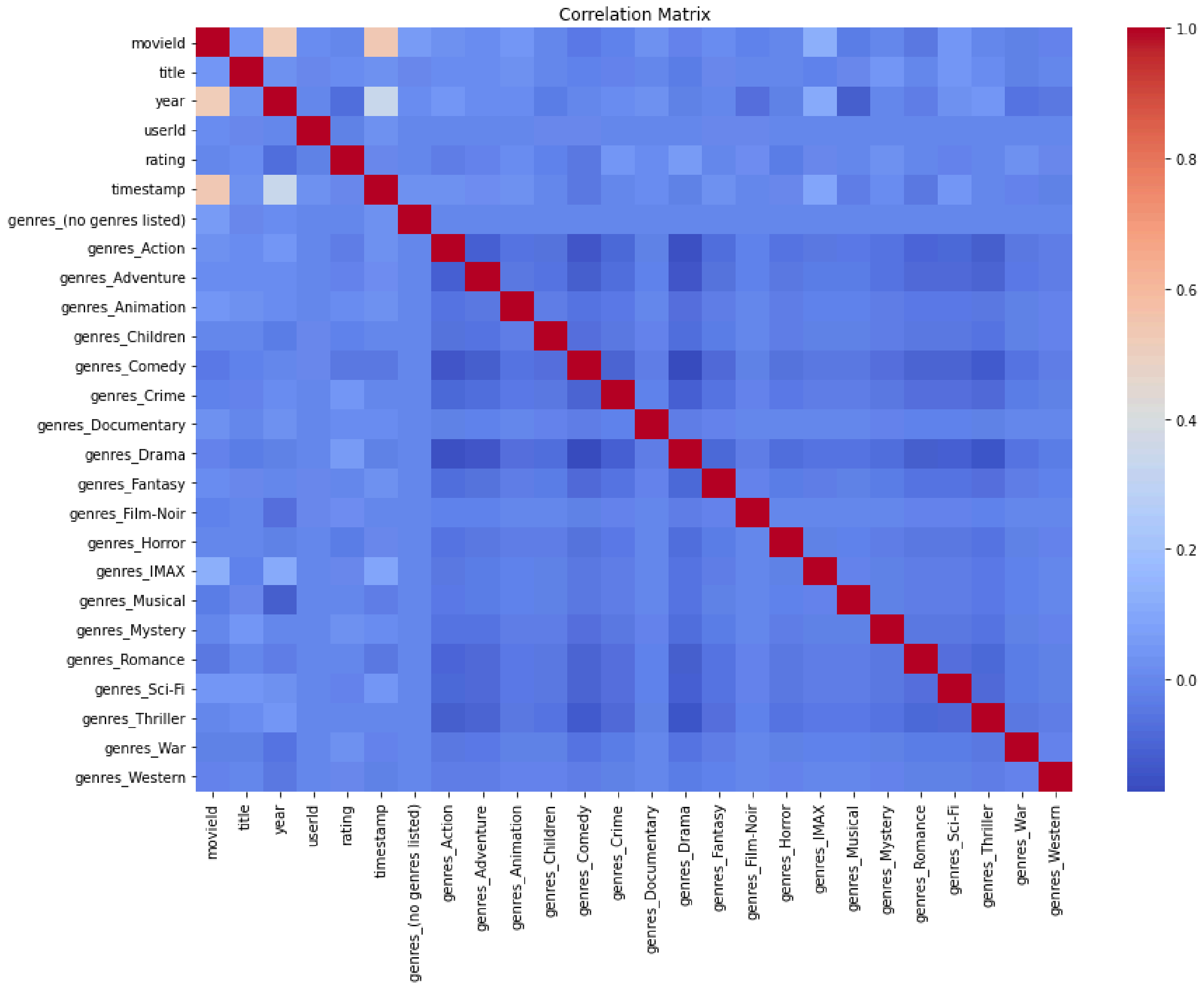
In [25]:

```
# Plot top genres count
genre_columns = [col for col in data.columns if 'genres_' in col]
genre_counts = data[genre_columns].sum().sort_values(ascending=False)

plt.figure(figsize=(12,6))
sns.barplot(x=genre_counts.values, y=genre_counts.index, palette='rocket')
plt.title('Popularity of Movie Genres')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')
plt.show()
```



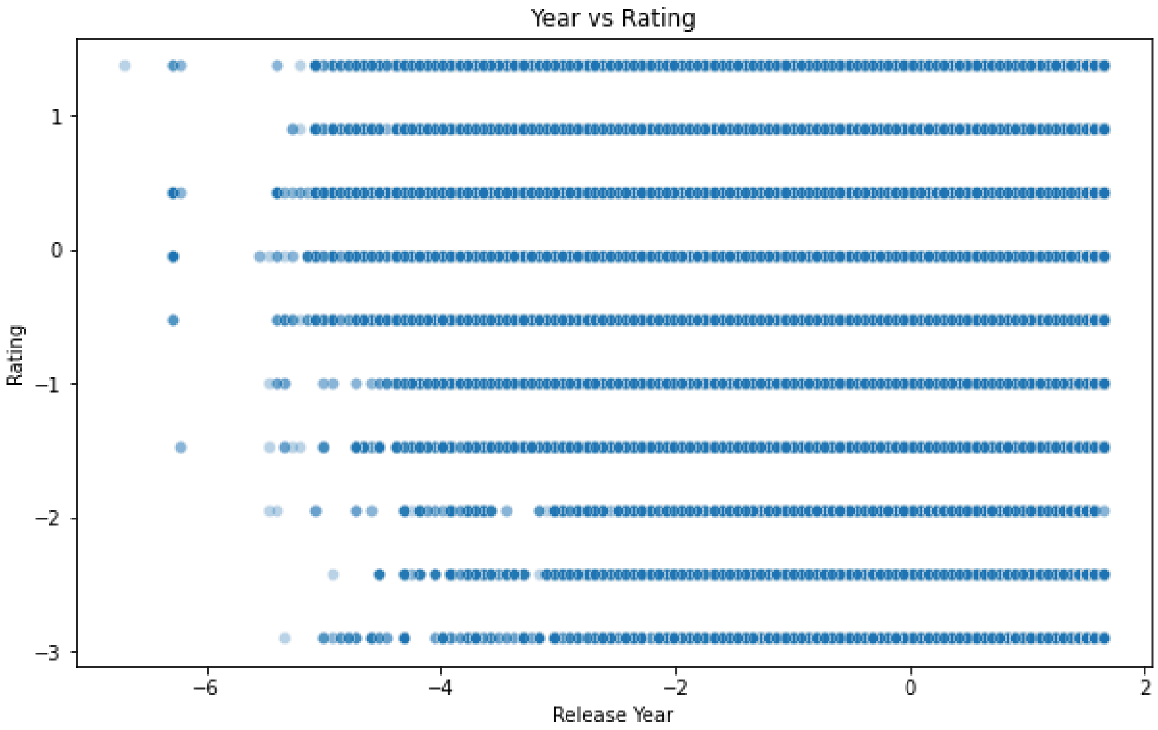
In [26]:

```
# Correlation heatmap
plt.figure(figsize=(14,10))
corr_matrix = data.corr()
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False)
plt.title('Correlation Matrix')
plt.show()
```



In [27]:

```
# Scatter plot
plt.figure(figsize=(10,6))
sns.scatterplot(x=data['year'], y=data['rating'], alpha=0.3)
plt.title('Year vs Rating')
plt.xlabel('Release Year')
plt.ylabel('Rating')
plt.show()
```



In [28]:

```
# Create 'era' bins
bins = [1900, 1950, 1970, 1990, 2010, 2025]
labels = ['1900s-50s', '50s-70s', '70s-90s', '90s-2010', '2010s+']
data['year_bin'] = pd.cut(data['year'], bins=bins, labels=labels)

# One-hot encode era
data = pd.get_dummies(data, columns=['year_bin'])
```

In [29]:
```
data
```

Out[29]:

	movieId	title	year	userId	rating	timestamp	genres_(no genres listed)	genres_Action	genres_Ad
0	1	14205	0.014773	2	-0.052442	1141415820	0	0	
1	1	14205	0.014773	3	0.421775	1439472215	0	0	
2	1	14205	0.014773	4	-0.526659	1573944252	0	0	
3	1	14205	0.014773	5	0.421775	858625949	0	0	
4	1	14205	0.014773	8	0.421775	890492517	0	0	
...	...	...	...	...	...	...	...	...	
899588	207642	7219	1.641360	2290	1.370209	1571620212	0	0	
899589	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899590	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899591	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	
899592	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	

899593 rows × 31 columns

In [30]:

```
data.isnull().sum()
```

Out[30]:

```
movieId                        0
title                          0
year                         220
userId                         0
rating                         0
timestamp                      0
genres_(no genres listed)      0
genres_Action                  0
genres_Adventure               0
genres_Animation               0
genres_Children                0
genres_Comedy                  0
genres_Crime                   0
genres_Documentary             0
genres_Drama                   0
genres_Fantasy                 0
genres_Film-Noir               0
genres_Horror                  0
genres_IMAX                    0
genres_Musical                 0
genres_Mystery                 0
genres_Romance                 0
genres_Sci-Fi                  0
genres_Thriller                0
genres_War                     0
genres_Western                 0
year_bin_1900s-50s             0
year_bin_50s-70s               0
year_bin_70s-90s               0
year_bin_90s-2010              0
year_bin_2010s+                0
dtype: int64
```

In [31]:

```
data['year'] = data['year'].fillna(data['year'].mode()[0])
```

```
In [32]: data.isnull().sum()
```

```
Out[32]: movieId                       0
         title                         0
         year                          0
         userId                        0
         rating                        0
         timestamp                     0
         genres_(no genres listed)     0
         genres_Action                 0
         genres_Adventure              0
         genres_Animation              0
         genres_Children               0
         genres_Comedy                 0
         genres_Crime                  0
         genres_Documentary            0
         genres_Drama                  0
         genres_Fantasy                0
         genres_Film-Noir              0
         genres_Horror                 0
         genres_IMAX                   0
         genres_Musical                0
         genres_Mystery                0
         genres_Romance                0
         genres_Sci-Fi                 0
         genres_Thriller               0
         genres_War                    0
         genres_Western                0
         year_bin_1900s-50s            0
         year_bin_50s-70s              0
         year_bin_70s-90s              0
         year_bin_90s-2010             0
         year_bin_2010s+               0
         dtype: int64
```

```
In [33]: from sklearn.preprocessing import PolynomialFeatures

         # Example with 2 features
         poly = PolynomialFeatures(degree=2, include_bias=False)
         poly_features = poly.fit_transform(data[['year', 'rating']])
```

In [34]:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Select numeric columns for PCA
X = data.select_dtypes(include=[np.number]).drop(columns=['userId', 'movieId'])

# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=0.95)  # Keep 95% variance
X_pca = pca.fit_transform(X_scaled)

print(f"PCA reduced to {X_pca.shape[1]} features.")
```

```
PCA reduced to 22 features.
```

In [35]:

```
from sklearn.model_selection import train_test_split

# Features and Target
X = data.drop(columns=['userId', 'movieId', 'timestamp', 'rating'])  # Drop irreleva
y = data['rating']  # Target variable

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
```

In [ ]:

```
from sklearn.linear_model import LinearRegression

# Initialize and train
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

# Predict
y_pred_lr = lr_model.predict(X_test)
print("y_pred_lr",y_pred_lr)
```

In [ ]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Evaluation for Linear Regression
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)

print(f"Linear Regression - MAE: {mae_lr:.4f}, RMSE: {rmse_lr:.4f}, R²: {r2_lr:.4f}"
```

In [37]:

```python
from sklearn.ensemble import RandomForestRegressor

# Initialize and train
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict
y_pred_rf = rf_model.predict(X_test)
print(y_pred_rf)
```

```
[-1.36666039 -0.37238449 -0.06237143 ... -1.28435013  0.62862553
 -0.21638797]
```

In [45]:

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Evaluation for Linear Regression
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)

print(f"Linear Regression - MAE: {mae_lr:.4f}, RMSE: {rmse_lr:.4f}, R²: {r2_lr:.4f}"
```

```
Linear Regression - MAE: 0.7896, RMSE: 0.9919, R²: 0.0164
```

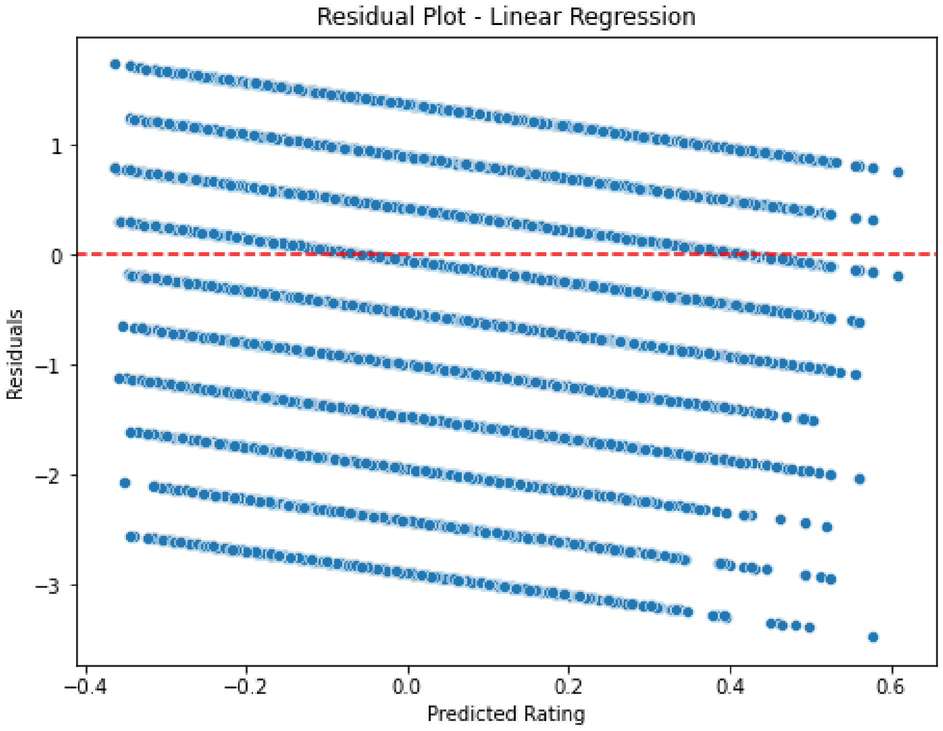
In [39]:

```python
# Evaluation for Random Forest
mae_rf = mean_absolute_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest - MAE: {mae_rf:.4f}, RMSE: {rmse_rf:.4f}, R²: {r2_rf:.4f}")
```

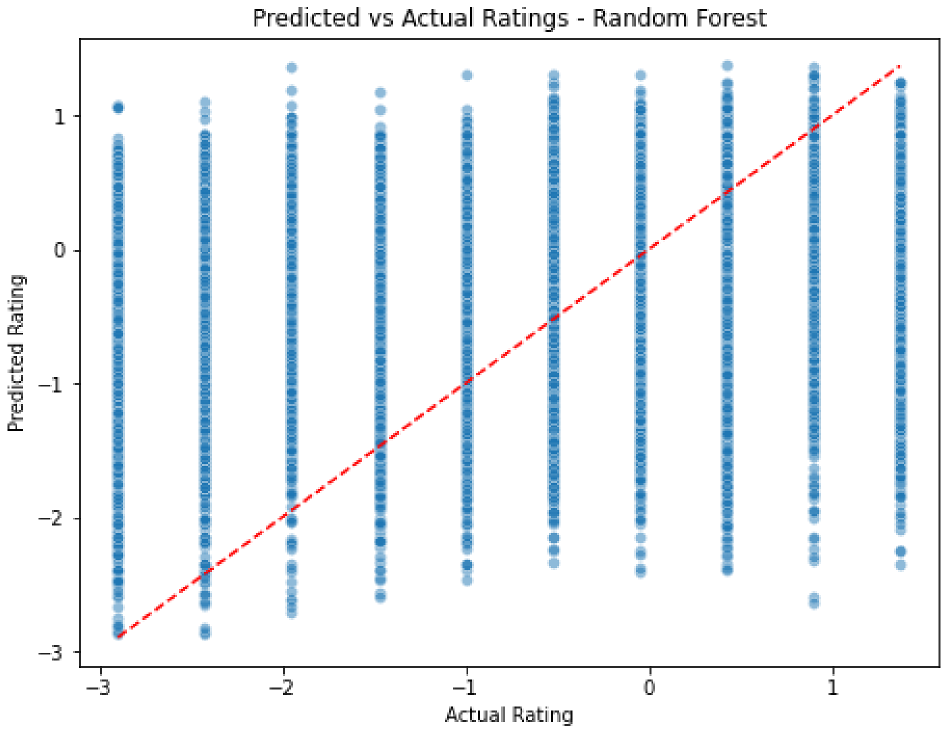
```
Random Forest - MAE: 0.7165, RMSE: 0.9236, R²: 0.1472
```

In [40]:

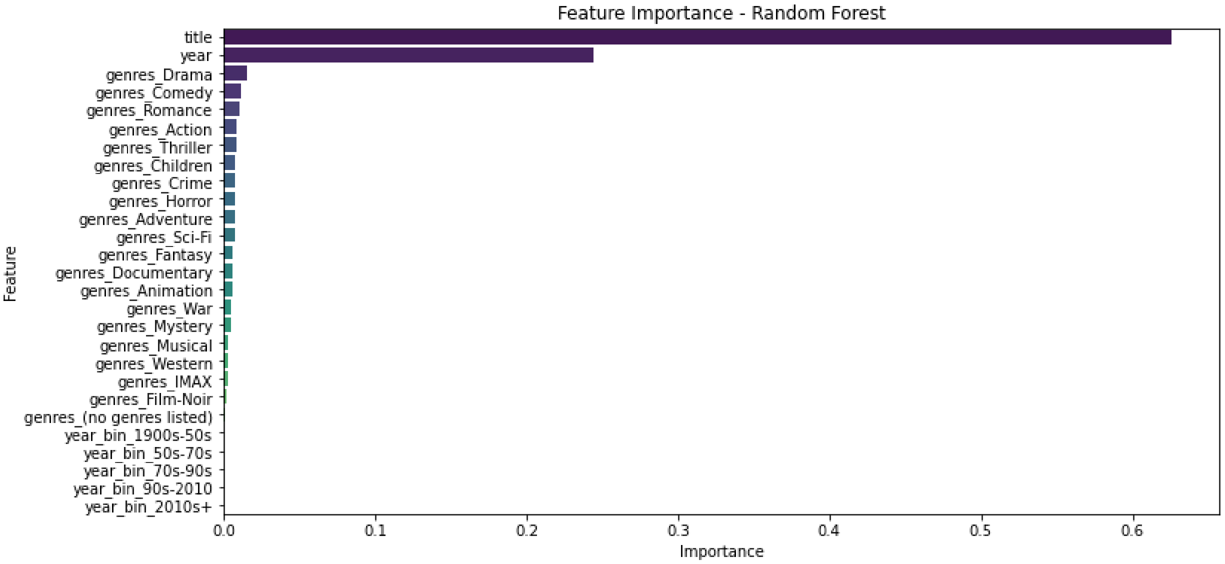
```
import matplotlib.pyplot as plt
import seaborn as sns

# Residuals
residuals = y_test - y_pred_lr

plt.figure(figsize=(8,6))
sns.scatterplot(x=y_pred_lr, y=residuals)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residual Plot - Linear Regression')
plt.xlabel('Predicted Rating')
plt.ylabel('Residuals')
plt.show()
```



In [41]:
```
# Plot Predicted vs Actual for Random Forest
plt.figure(figsize=(8,6))
sns.scatterplot(x=y_test, y=y_pred_rf, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Line
plt.title('Predicted vs Actual Ratings - Random Forest')
plt.xlabel('Actual Rating')
plt.ylabel('Predicted Rating')
plt.show()
```

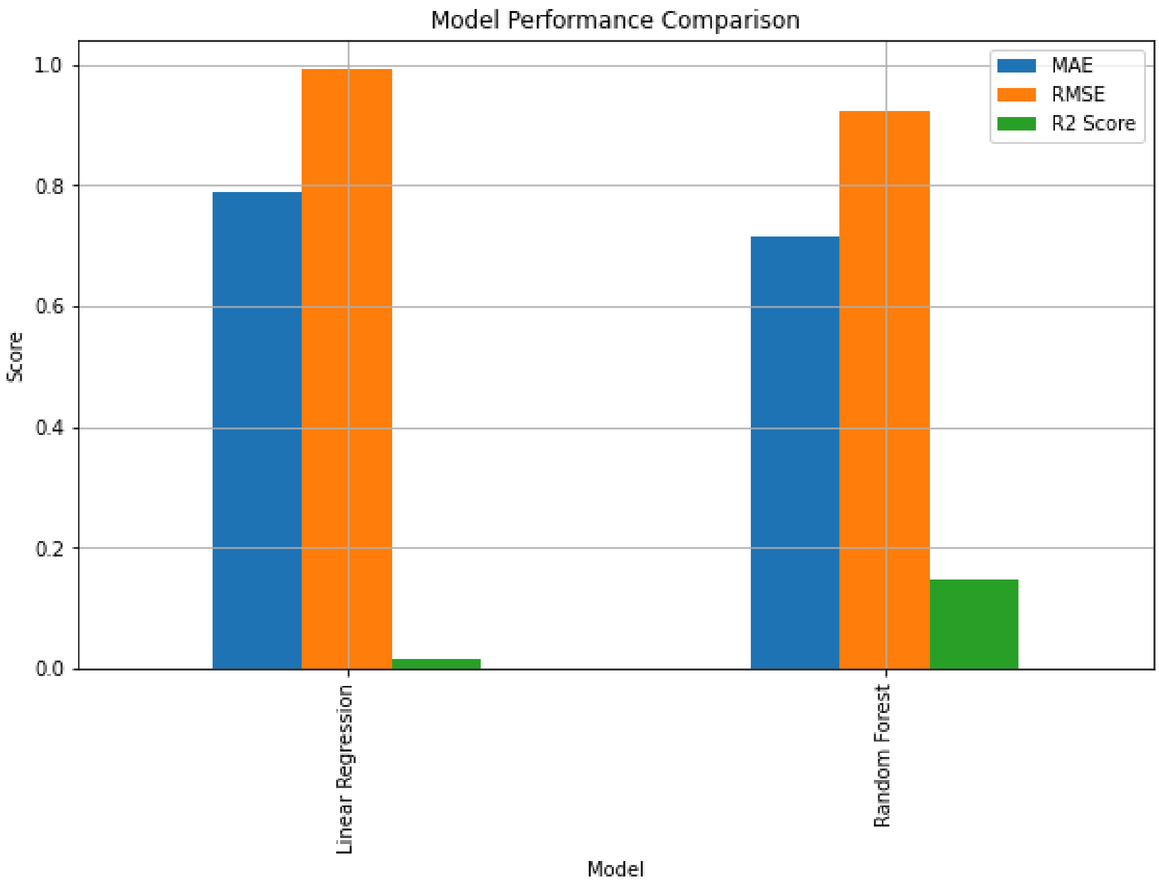


```
In [42]: # Get feature importance
         importances = rf_model.feature_importances_
         features = X.columns

         # Create DataFrame
         feat_imp = pd.DataFrame({'Feature': features, 'Importance': importances})
         feat_imp = feat_imp.sort_values('Importance', ascending=False)

         # Plot
         plt.figure(figsize=(12,6))
         sns.barplot(x='Importance', y='Feature', data=feat_imp, palette='viridis')
         plt.title('Feature Importance - Random Forest')
         plt.xlabel('Importance')
         plt.ylabel('Feature')
         plt.show()
```



In [43]:

```
# Create performance table
metrics_df = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest'],
    'MAE': [mae_lr, mae_rf],
    'RMSE': [rmse_lr, rmse_rf],
    'R2 Score': [r2_lr, r2_rf]
})

# Bar plot
metrics_df.set_index('Model').plot(kind='bar', figsize=(10,6))
plt.title('Model Performance Comparison')
plt.ylabel('Score')
plt.grid(True)
plt.show()
```



In [ ]:

In [ ]: