```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')

# Set style for better visualizations
plt.style.use('seaborn-v0_8')
sns.set_palette("husl")

# Load the dataset
# Update this path with your actual dataset path
data = pd.read_csv('IMDB-Movie-Data.csv')

print("IMDB Movie Dataset Analysis")
print("=" * 40)

# 1. BASIC DATA EXPLORATION
print("\n1. DATASET OVERVIEW")
print("-" * 20)
print(f"Dataset shape: {data.shape}")
print(f"Number of rows: {data.shape[0]}")
print(f"Number of columns: {data.shape[1]}")

print("\nFirst 5 rows:")
print(data.head())
print("\nDataset Info:")
print(data.info())

print("\nColumn names:")
print(data.columns.tolist())

print("\nBasic Statistics:")
print(data.describe())

# 2. DATA CLEANING AND PREPROCESSING
print("\n2. DATA CLEANING")
print("-" * 20)

# Check for missing values
print("Missing values in each column:")
print(data.isnull().sum())

# Check data types
print("\nData types:")
print(data.dtypes)

# Remove duplicates if any
data_cleaned = data.drop_duplicates()
print(f"\nRows after removing duplicates: {data_cleaned.shape[0]}")

# 3. EXPLORATORY DATA ANALYSIS
print("\n3. EXPLORATORY DATA ANALYSIS")
print("-" * 30)

# Assuming common IMDB dataset columns
# Adjust column names based on your actual dataset structure
rating_col = 'Rating' if 'Rating' in data.columns else 'IMDB Rating'
genre_col = 'Genre' if 'Genre' in data.columns else 'Genres'
title_col = 'Title' if 'Title' in data.columns else 'Movie Title'
year_col = 'Year' if 'Year' in data.columns else 'Release Year'

print(f"\nRating statistics for '{rating_col}':")
if rating_col in data.columns:
    print(f"Mean rating: {data[rating_col].mean():.2f}")
    print(f"Median rating: {data[rating_col].median():.2f}")
    print(f"Min rating: {data[rating_col].min():.2f}")
    print(f"Max rating: {data[rating_col].max():.2f}")

# 4. VISUALIZATION SECTION
print("\n4. CREATING VISUALIZATIONS")
print("-" * 25)

# Set up the plotting area
fig, axes = plt.subplots(2, 2, figsize=(15, 12))
fig.suptitle('IMDB Movie Dataset Analysis', fontsize=16, fontweight='bold')

# Plot 1: Rating Distribution
if rating_col in data.columns:
    axes[0, 0].hist(data[rating_col].dropna(), bins=20, alpha=0.7, color='skyblue',
edgecolor='black')
```

```
    axes[0, 0].set_title('Distribution of Movie Ratings')
    axes[0, 0].set_xlabel('Rating')
    axes[0, 0].set_ylabel('Frequency')
    axes[0, 0].grid(True, alpha=0.3)

# Plot 2: Movies by Year (if year column exists)
if year_col in data.columns:
    year_counts = data[year_col].value_counts().sort_index()
    axes[0, 1].plot(year_counts.index, year_counts.values, marker='o', linewidth=2)
    axes[0, 1].set_title('Number of Movies by Year')
    axes[0, 1].set_xlabel('Year')
    axes[0, 1].set_ylabel('Number of Movies')
    axes[0, 1].grid(True, alpha=0.3)

# Plot 3: Top 10 Genres (if genre column exists)
if genre_col in data.columns:
    # Handle multiple genres per movie (assuming they're separated by commas or pipes)
    all_genres = []
    for genres in data[genre_col].dropna():
        if isinstance(genres, str):
            genre_list = genres.replace('|', ',').split(',')
            all_genres.extend([genre.strip() for genre in genre_list])

    genre_counts = pd.Series(all_genres).value_counts().head(10)
    axes[1, 0].barh(genre_counts.index, genre_counts.values, color='lightcoral')
    axes[1, 0].set_title('Top 10 Movie Genres')
    axes[1, 0].set_xlabel('Number of Movies')
    axes[1, 0].grid(True, alpha=0.3)
```

```
# Plot 4: Rating vs Revenue correlation (if revenue column exists)
revenue_cols = ['Revenue', 'Revenue (Millions)', 'Box Office', 'Gross']
revenue_col = None
for col in revenue_cols:
    if col in data.columns:
        revenue_col = col
        break

if revenue_col and rating_col in data.columns:
    clean_data = data[[rating_col, revenue_col]].dropna()
    axes[1, 1].scatter(clean_data[rating_col], clean_data[revenue_col], alpha=0.6,
color='green')
    axes[1, 1].set_title(f'Rating vs {revenue_col}')
    axes[1, 1].set_xlabel('Rating')
    axes[1, 1].set_ylabel(revenue_col)
    axes[1, 1].grid(True, alpha=0.3)
else:
    axes[1, 1].text(0.5, 0.5, 'Revenue data not available',
                   horizontalalignment='center', verticalalignment='center',
                   transform=axes[1, 1].transAxes, fontsize=12)
    axes[1, 1].set_title('Revenue Analysis')

plt.tight_layout()
plt.show()

# 5. ADVANCED ANALYSIS TASKS
print("\n5. ADVANCED ANALYSIS")
print("-" * 20)


# Task 1: Top 5 Rated Movies Overall
print("\nTask 1: TOP 5 RATED MOVIES OVERALL")
print("-" * 35)
if rating_col in data.columns and title_col in data.columns:
    top_5_movies = data.nlargest(5, rating_col)[[title_col, rating_col]]
    print(top_5_movies.to_string(index=False))


    # Visualization for top 5 movies
    plt.figure(figsize=(12, 6))
    plt.barh(top_5_movies[title_col], top_5_movies[rating_col], color='gold',
edgecolor='black')
    plt.title('Top 5 Highest Rated Movies', fontsize=14, fontweight='bold')
    plt.xlabel('Rating')
    plt.ylabel('Movie Title')
    plt.grid(axis='x', alpha=0.3)


    # Add rating labels on bars
    for i, v in enumerate(top_5_movies[rating_col]):
        plt.text(v + 0.01, i, str(v), va='center', fontweight='bold')


    plt.tight_layout()
    plt.show()


# Task 2: Top 10 Rated Movies in Comedy Genre
print("\nTask 2: TOP 10 RATED COMEDY MOVIES")
print("-" * 32)
```

```
if genre_col in data.columns and rating_col in data.columns and title_col in data.columns:
    # Filter comedy movies (case-insensitive)
    comedy_movies = data[data[genre_col].str.contains('Comedy', case=False, na=False)]

    if len(comedy_movies) > 0:
        top_10_comedy = comedy_movies.nlargest(10, rating_col)[[title_col, rating_col, 
genre_col]]
        print(top_10_comedy.to_string(index=False))

        # Visualization for top 10 comedy movies
        plt.figure(figsize=(12, 8))
        plt.barh(range(len(top_10_comedy)), top_10_comedy[rating_col], color='lightblue', 
edgecolor='black')
        plt.yticks(range(len(top_10_comedy)), top_10_comedy[title_col])
        plt.title('Top 10 Highest Rated Comedy Movies', fontsize=14, fontweight='bold')
        plt.xlabel('Rating')
        plt.ylabel('Movie Title')
        plt.grid(axis='x', alpha=0.3)

        # Add rating labels on bars
        for i, v in enumerate(top_10_comedy[rating_col]):
            plt.text(v + 0.01, i, str(v), va='center', fontweight='bold')

        plt.tight_layout()
        plt.show()
    else:
        print("No comedy movies found in the dataset.")

# 6. ADDITIONAL ANALYSIS
```

```
print("\n6. ADDITIONAL INSIGHTS")
print("-" * 22)

# Genre Analysis
if genre_col in data.columns:
    print("\nGenre-wise Average Rating:")
    genre_ratings = {}
    for idx, row in data.iterrows():
        if pd.notna(row[genre_col]) and pd.notna(row[rating_col]):
            genres = row[genre_col].replace('|', ',').split(',')
            for genre in genres:
                genre = genre.strip()
                if genre not in genre_ratings:
                    genre_ratings[genre] = []
                genre_ratings[genre].append(row[rating_col])

    avg_genre_ratings = {genre: np.mean(ratings) for genre, ratings in genre_ratings.items()}
    genre_df = pd.DataFrame(list(avg_genre_ratings.items()), columns=['Genre', 'Average
Rating'])
    genre_df = genre_df.sort_values('Average Rating', ascending=False).head(10)
    print(genre_df.to_string(index=False))

# Year-wise trends
if year_col in data.columns and rating_col in data.columns:
    print(f"\nYear-wise Average Rating (last 10 years):")
    recent_years = data[data[year_col] >= (data[year_col].max() - 10)]
    yearly_avg = recent_years.groupby(year_col)[rating_col].mean().round(2)
    print(yearly_avg.to_string())
```

```
# 7. SUMMARY STATISTICS

print("\n7. SUMMARY REPORT")

print("-" * 18)

print(f"Total movies analyzed: {len(data)}")

if rating_col in data.columns:

    print(f"Average movie rating: {data[rating_col].mean():.2f}")

    print(f"Highest rated movie: {data.loc[data[rating_col].idxmax(), title_col] if title_col in
data.columns else 'N/A'}")

    print(f"Lowest rated movie: {data.loc[data[rating_col].idxmin(), title_col] if title_col in
data.columns else 'N/A'}")


if year_col in data.columns:

    print(f"Year range: {data[year_col].min()} - {data[year_col].max()}")


print("\nAnalysis completed successfully!")

print("=" * 40)
```