Netflix! What started in 1997 as a DVD rental service has since exploded into one of the largest entertainment and media companies.

Given the large number of movies and series available on the platform, it is a perfect opportunity to flex your exploratory data analysis skills and dive into the entertainment industry.

You work for a production company that specializes in nostalgic styles. You want to do some research on movies released in the 1990's. You'll delve into Netflix data and perform exploratory data analysis to better understand this awesome movie decade!

You have been supplied with the dataset `netflix_data.csv`, along with the following table detailing the column names and descriptions. Feel free to experiment further after submitting!

The data

netflix_data.csv

Column	Description
`show_id`	The ID of the show
`type`	Type of show
`title`	Title of the show
`director`	Director of the show
`cast`	Cast of the show
`country`	Country of origin
`date_added`	Date added to Netflix
`release_year`	Year of Netflix release
`duration`	Duration of the show in minutes
`description`	Description of the show
`genre`	Show genre

```
## NAME OF PROGRAMMER: Vanessa Nwankwo
## PROGRAMMING LANGUAGE: Python
## DATE MODIFIED: 31st March 2025
## DATASET: Netflix Movies Dataset
## PROJECT TITLE: Investigating Netflix Movies

# Importing pandas and matplotlib
import pandas as pd
import matplotlib.pyplot as plt

# Read in the Netflix CSV as a DataFrame
netflix_df = pd.read_csv("netflix_data.csv")
```

```
#STEP 1: PREPARING THE DATA
# Check the data types and non-null counts for each column
#print(netflix_df.info())

# Calculate the number of missing values in each column
missing_values = netflix_df.isnull().sum()
print(missing_values)

# check datatypes of the columns
print(netflix_df.dtypes)

# Defining a function to standardize the 'date_added' column entries
def standardize_date(date_str):
    # If the value is missing or not a string, return NaT
    if pd.isnull(date_str):
        return pd.NaT
    date_str = str(date_str).strip()  # Remove extra whitespace
    # If the date_str is just a year (4 characters), assume January 1st of that
year
    if len(date_str) == 4 and date_str.isdigit():
        date_str = date_str + '-01-01'
    # Convert the string to a datetime object
    return pd.to_datetime(date_str, errors='coerce')

# Apply the function to the 'date_added' column
netflix_df['date_added'] = netflix_df['date_added'].apply(standardize_date)

# Verify the conversion
print(netflix_df['date_added'].head())

# Count how many duplicate rows are present
duplicate_count = netflix_df.duplicated().sum()
print("Number of duplicate rows:", duplicate_count)

# This line creates a Boolean Series that marks True for duplicate 'show_id'
entries (after the first occurrence)
duplicates = netflix_df.duplicated(subset='show_id')

# To see how many duplicates there are, you can sum the Boolean Series (since
True counts as 1)
duplicate_count = duplicates.sum()
print("Number of duplicate rows based on 'show_id':", duplicate_count)

# To see duplicate rows (only the duplicate occurrences), filter the DataFrame:
duplicate_rows = netflix_df[duplicates]
print(duplicate_rows)

# Filter the DataFrame to keep only the rows where:
# 1. The 'type' column is 'Movie'
# 2. The 'release_year' is greater than or equal to 1990 and less than 2000
(i.e., movies released in the 1990s)
```

```
movies_1990s = netflix_df[(netflix_df['type'] == 'Movie') &
(netflix_df['release_year'] >= 1990) & (netflix_df['release_year'] < 2000)]

# Display the first few rows of the filtered DataFrame to verify the results
print(movies_1990s.head())

# QUESTION 1: CALCULATE THE FREQUENCY OF MOVIE DURATION IN THE 1990s

# A. Identify the most frequent movie duration and its frequency

duration_counts = movies_1990s['duration'].value_counts() # This counts how many
times each duration value appears.


duration = duration_counts.idxmax()   # returns the index label of the maximum
value.
frequency = duration_counts.max()                    # returns the maximum value
itself, the highest count.

# Print the most frequent movie duration and its frequency
print("The most frequent movie duration in the 1990s is:", duration,
      "minutes, with", frequency, "movies.")

# Plot a histogram to visualize the distribution of movie durations in the 1990s
plt.figure(figsize=(10, 6))  # Set the figure size for better readability

# B. Plot the histogram for the 'duration' column (you may adjust the number of
bins as needed)
plt.hist(movies_1990s['duration'], bins=47, color='skyblue', edgecolor='black')

# Add a vertical dashed red line at the most frequent duration to highlight it
plt.axvline(duration, color='red', linestyle='dashed', linewidth=2,
            label=f"Most Frequent: {duration} min")

# Label the axes and add a title (ctrl + /)

plt.xlabel("Movie Duration (minutes)")
plt.ylabel("Number of Movies")
plt.title("Distribution of Movie Durations in the 1990s")

# Add a legend to explain the highlighted line
plt.legend()

# Display the plot
plt.show()


# C. Get frequency counts for each unique duration
duration_counts = movies_1990s['duration'].value_counts().sort_index()

# Plot the exact frequencies
plt.figure(figsize=(12, 6))
```

```
plt.bar(duration_counts.index, duration_counts.values, color='skyblue',
edgecolor='black')
plt.axvline(duration_counts.idxmax(), color='red', linestyle='--', label=f"Most
Frequent: {duration_counts.idxmax()} min")

# Add titles and labels
plt.title('Exact Frequency of Movie Durations (1990s)')
plt.xlabel('Duration (minutes)')
plt.ylabel('Number of Movies')
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()


# Create duration bins
bins = list(range(60, 181, 10))
labels = [f"{b}-{b+9}" for b in bins[:-1]]

# Bin durations into ranges
movies_1990s['duration_range'] = pd.cut(movies_1990s['duration'], bins=bins,
labels=labels, right=False)

# Count movies in each range
range_counts = movies_1990s['duration_range'].value_counts().sort_index()

# Define neon-like colors
neon_colors = ['#39FF14', '#FF10F0', '#0FF0FC', '#F5FF10', '#FF5F1F', '#FF3131',
'#7D00FF', '#00FFFF', '#FFB347', '#ADFF2F', '#FFD700', '#00FF7F']

# Plot
plt.figure(figsize=(12, 6))
bars = plt.bar(range_counts.index, range_counts.values,
color=neon_colors[:len(range_counts)], edgecolor='black')

# Annotate bars with exact values
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2.0, yval + 0.5, int(yval),
ha='center', va='bottom', fontsize=10, color='white')

plt.title('🎬 Movie Duration Ranges in the 1990s', fontsize=14)
plt.xlabel('Duration Range (minutes)')
plt.ylabel('Number of Movies')
plt.grid(axis='y', linestyle='--', alpha=0.4)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()



# QUESTION 2: Analysing yearly trends
```

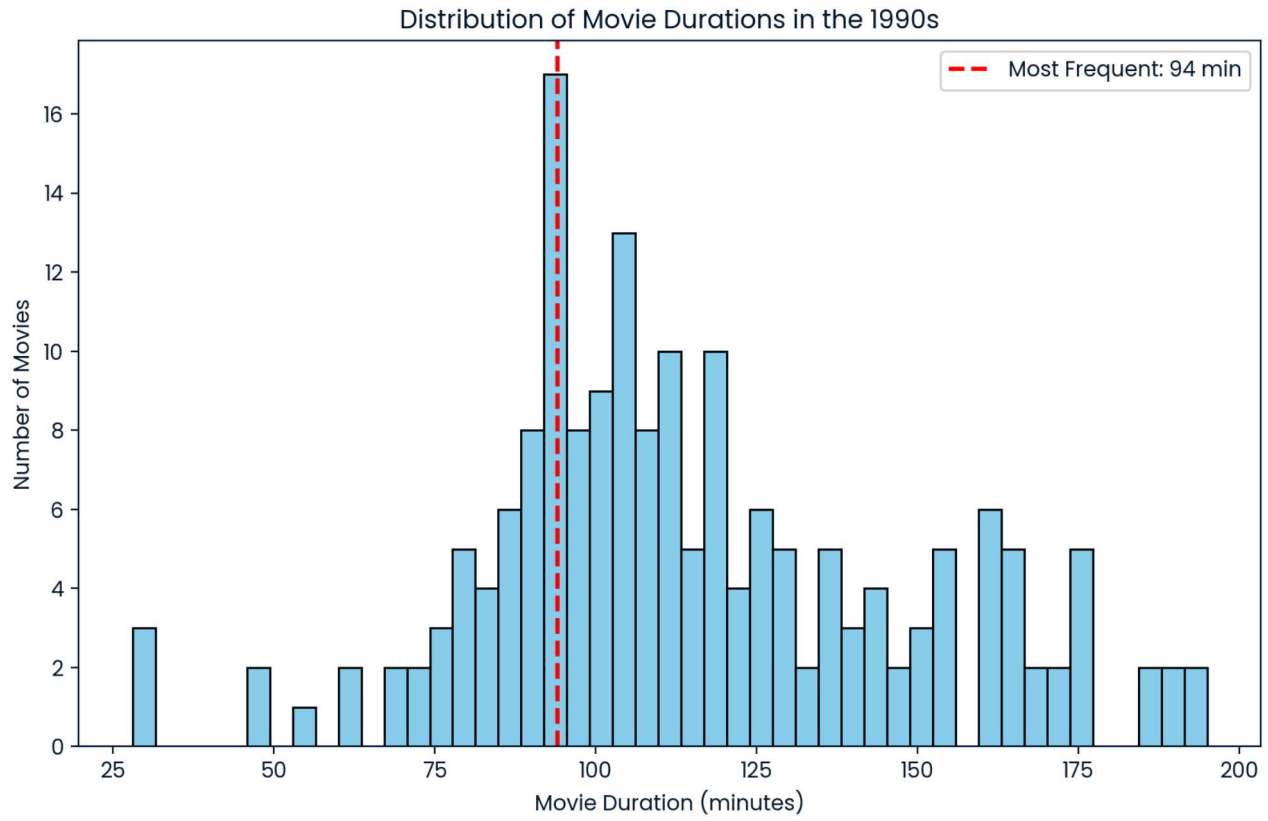
```
# A. Group the data by 'release_year' and count the number of movies for each
year.
movies_count_by_year = movies_1990s.groupby('release_year').size()
# Print the count of movies per year in the 1990s.
print("Movies count per year in the 1990s:")
print(movies_count_by_year)


#  Filter movies with duration == 94 mins
movies_94mins = movies_1990s[movies_1990s['duration'] == 94]

#  Group by release year and count
yearly_counts = movies_94mins['release_year'].value_counts().sort_index()

# Plot
plt.figure(figsize=(10, 5))
plt.bar(yearly_counts.index, yearly_counts.values, color='limegreen',
edgecolor='black')

# Add labels and title
plt.title('Number of Movies with 94-Minute Duration by Year (1990s)',
fontsize=14)
plt.xlabel('Release Year')
plt.ylabel('Number of Movies')
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.xticks(yearly_counts.index, rotation=45)
plt.tight_layout()
plt.show()


# B. Group the movies by 'release_year' and calculate duration statistics: mean,
median, and standard deviation.
# The .agg() function applies multiple aggregate functions to the 'duration'
column.
duration_stats_by_year = movies_1990s.groupby('release_year')
['duration'].agg(['mean', 'median', 'std'])
# Print the duration statistics for each year in the 1990s.
print("\nDuration statistics per year in the 1990s:")
print(duration_stats_by_year)

# Create a line plot to visualize the duration trends over the 1990s.
plt.figure(figsize=(10, 6))  # Set the size of the plot for clarity

# Plot the mean movie duration per year
plt.plot(duration_stats_by_year.index, duration_stats_by_year['mean'],
marker='o', label='Mean Duration')
# Plot the median movie duration per year
plt.plot(duration_stats_by_year.index, duration_stats_by_year['median'],
marker='o', label='Median Duration')
# Plot the standard deviation of movie duration per year
```

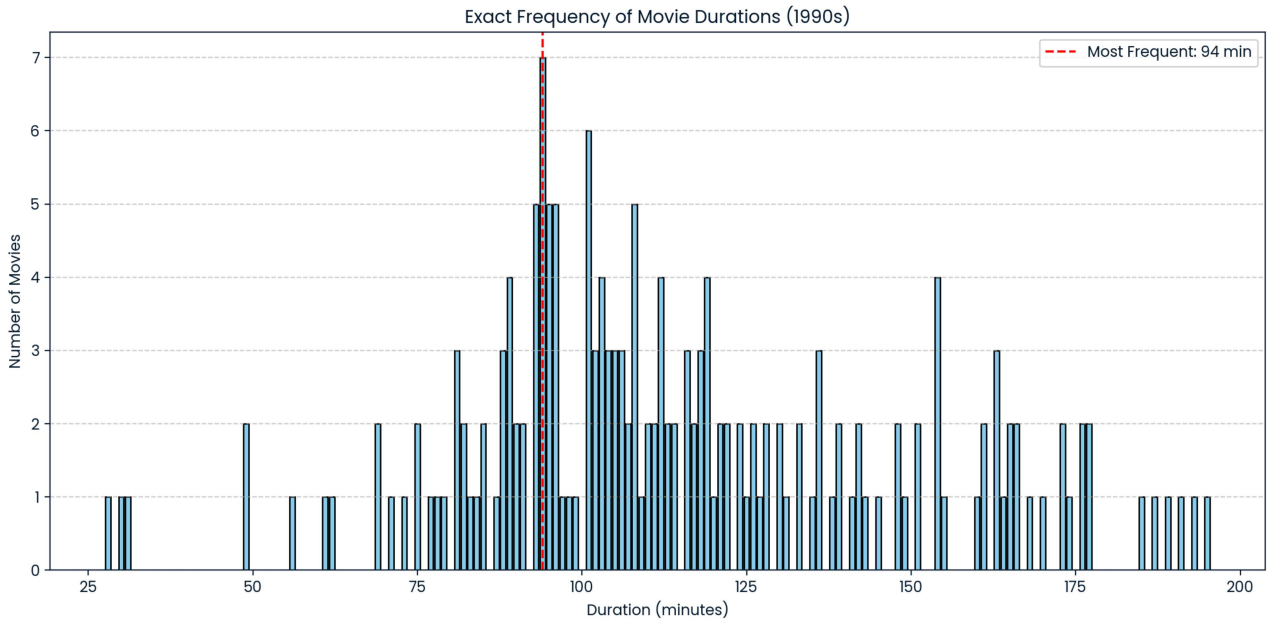
```
plt.plot(duration_stats_by_year.index, duration_stats_by_year['std'], marker='o',
label='Std Dev of Duration')

# Label the axes and the plot title
plt.xlabel("Release Year")

plt.title("Movie Duration Trends in the 1990s")
plt.legend()  # Add a legend to identify each line
plt.grid(True)  # Enable grid for easier reading of values

# Display the plot
plt.show()

# QUESTION 3: Genre Analysis

# A. Genre Frequency
# Split the 'genre' column on commas and 'explode' the list into separate rows.
# .str.strip() is used to remove any extra whitespace.
genre_series = movies_1990s['genre'].str.split(',').explode().str.strip()

# Count how many times each genre appears
genre_counts = genre_series.value_counts()

# Print the genre frequency counts
print("Genre Frequency in 1990s Movies:")
print(genre_counts)

# Plot a bar chart of genre frequencies
plt.figure(figsize=(10, 6))  # Set the figure size
genre_counts.plot(kind='bar', color='skyblue', edgecolor='black')
plt.xlabel('Genre')        # Label for x-axis
plt.ylabel('Number of Movies')  # Label for y-axis
plt.title('Frequency of Genres in 1990s Movies')
plt.xticks(rotation=45)    # Rotate x-axis labels for better readability
plt.tight_layout()         # Adjust layout for neatness
plt.show()                 # Display the plot

# TO DETERMINE SHORT ACTION MOVIES IN THE 1990S
#Further subset the movies_1990s DataFrame to keep only those with "Action" in
their genre
# This uses str.contains to search for the substring "Action" in a case-
insensitive way.
action_movies_1990s = movies_1990s[movies_1990s['genre'].str.contains('Action',
case=False, na=False)]

# Initialize a counter for short movies (movies with a duration less than 90
minutes)
short_movie_count = 0

# Iterate through each row in the action_movies_1990s DataFrame
for index, row in action_movies_1990s.iterrows():
    # Check if the 'duration' of the movie is less than 90 minutes
```

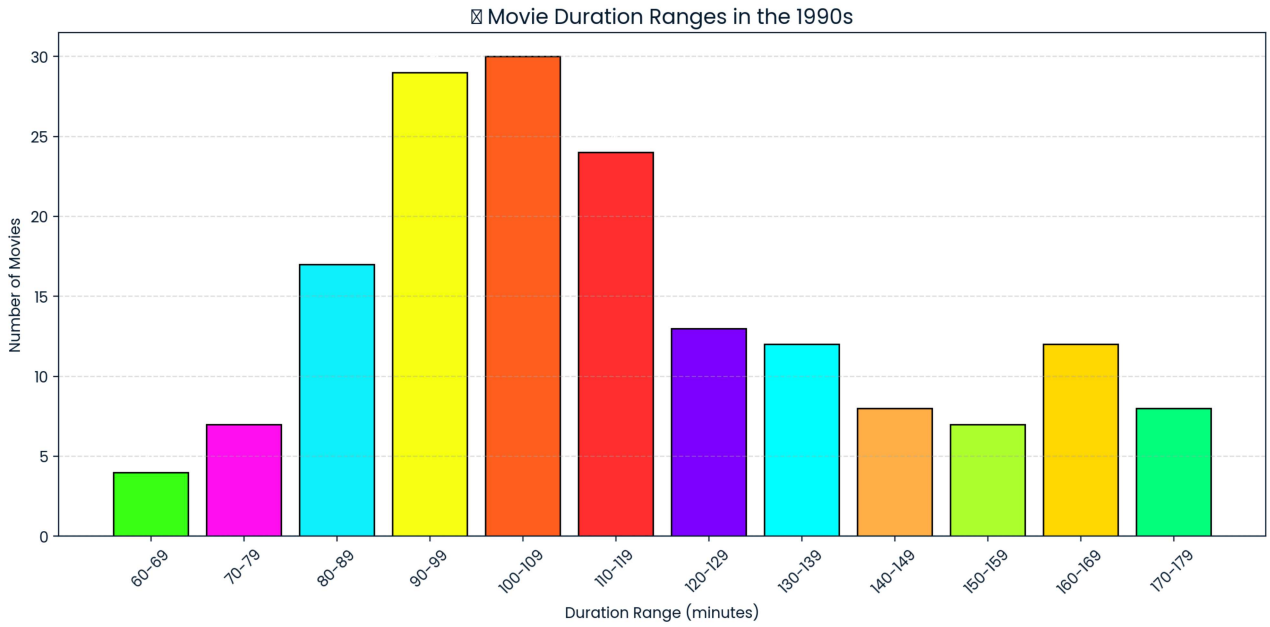
```
    if row['duration'] < 90:
        # If so, increment the counter
        short_movie_count += 1

# Print the number of short action movies from the 1990s
print("Number of short action movies (less than 90 minutes) from the 1990s:",
short_movie_count)

# B. Genre vs Duration
# Create a new DataFrame that 'explodes' the genre column so that each movie-
genre combination is its own row.
movies_genres = movies_1990s.copy()                  # Make a copy of the 1990s
movies DataFrame
movies_genres['genre'] = movies_genres['genre'].str.split(',')  # Split the
'genre' column into a list of genres
movies_genres = movies_genres.explode('genre')       # Expand lists into separate
rows
movies_genres['genre'] = movies_genres['genre'].str.strip()  # Remove extra
whitespace from genre names

# Create a box plot to compare the distribution of movie durations across genres.
plt.figure(figsize=(12, 8))                    # Set the figure size for the box
plot
movies_genres.boxplot(column='duration', by='genre', rot=45)  # Create a box plot
with 'duration' grouped by 'genre' and rotate labels
plt.xlabel('Genre')                            # Label for x-axis
plt.ylabel('Duration (minutes)')               # Label for y-axis
plt.title('Distribution of Movie Durations by Genre (1990s)')  # Plot title
plt.suptitle('')  # Remove the default 'Boxplot grouped by genre' subtitle
plt.tight_layout()   # Adjust layout for neatness
plt.show()           # Display the plot
```

```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
duration        0
description     0
genre           0
dtype: int64
show_id         object
type            object
title           object
director        object
cast            object
country         object
date_added      object
release_year     int64
```

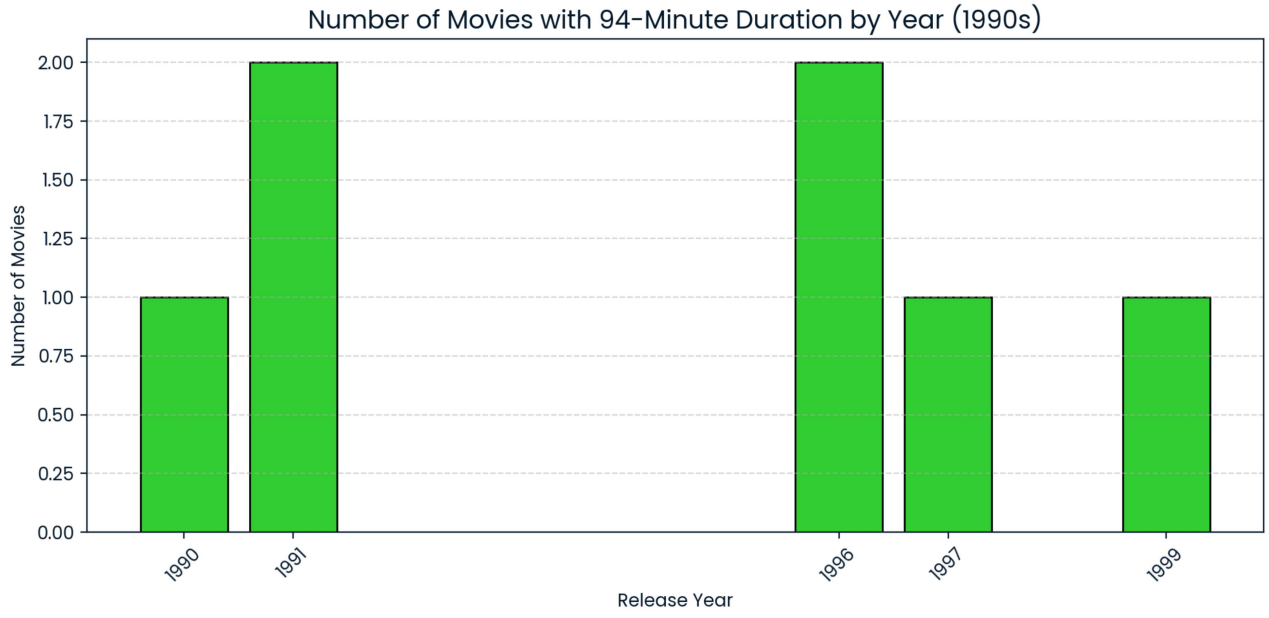
```
duration         int64
description     object
genre           object
dtype: object
0   2016-12-23
1   2018-12-20
2   2017-11-16
3   2020-01-01
4   2017-07-01
```





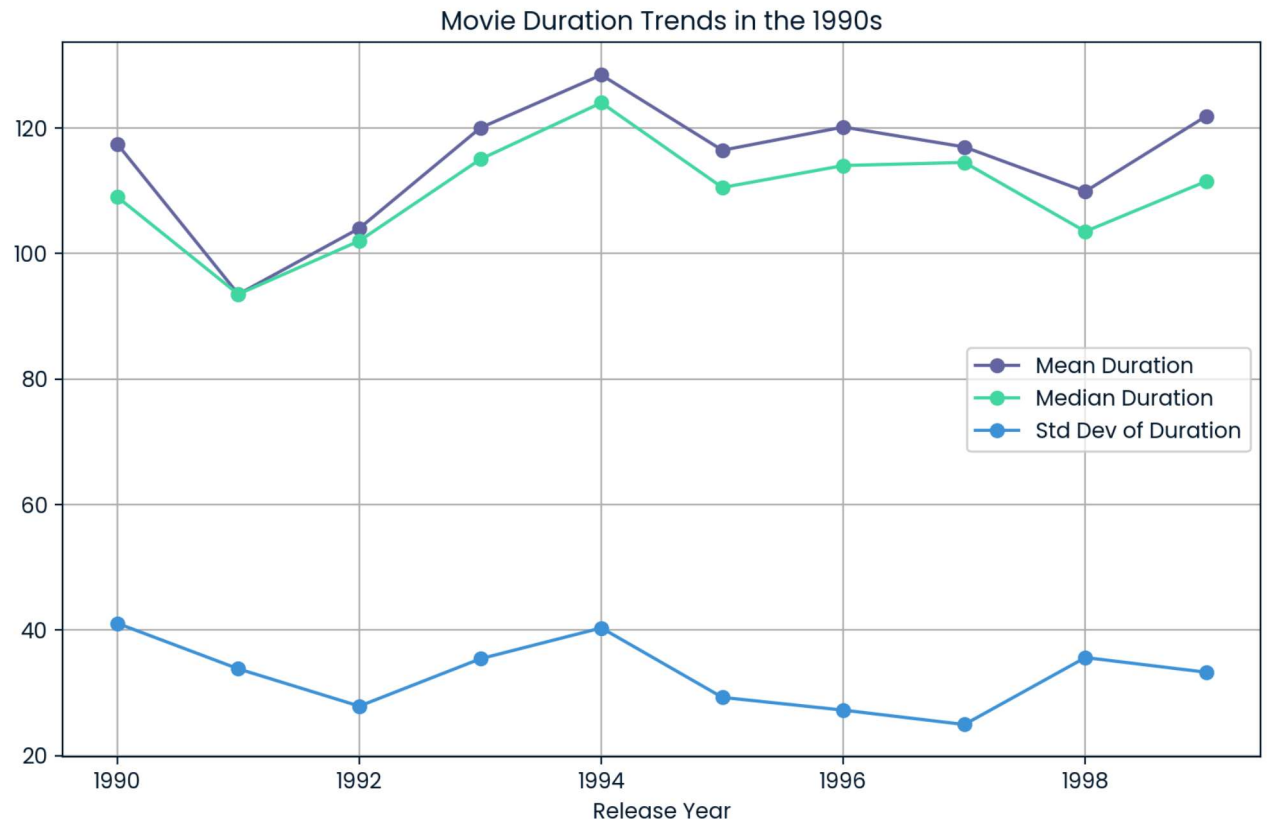



```
Movies count per year in the 1990s:
release_year
1990    14
1991    14
1992    16
1993    16
1994    14
1995    16
1996    15
1997    26
1998    26
1999    26
dtype: int64
```

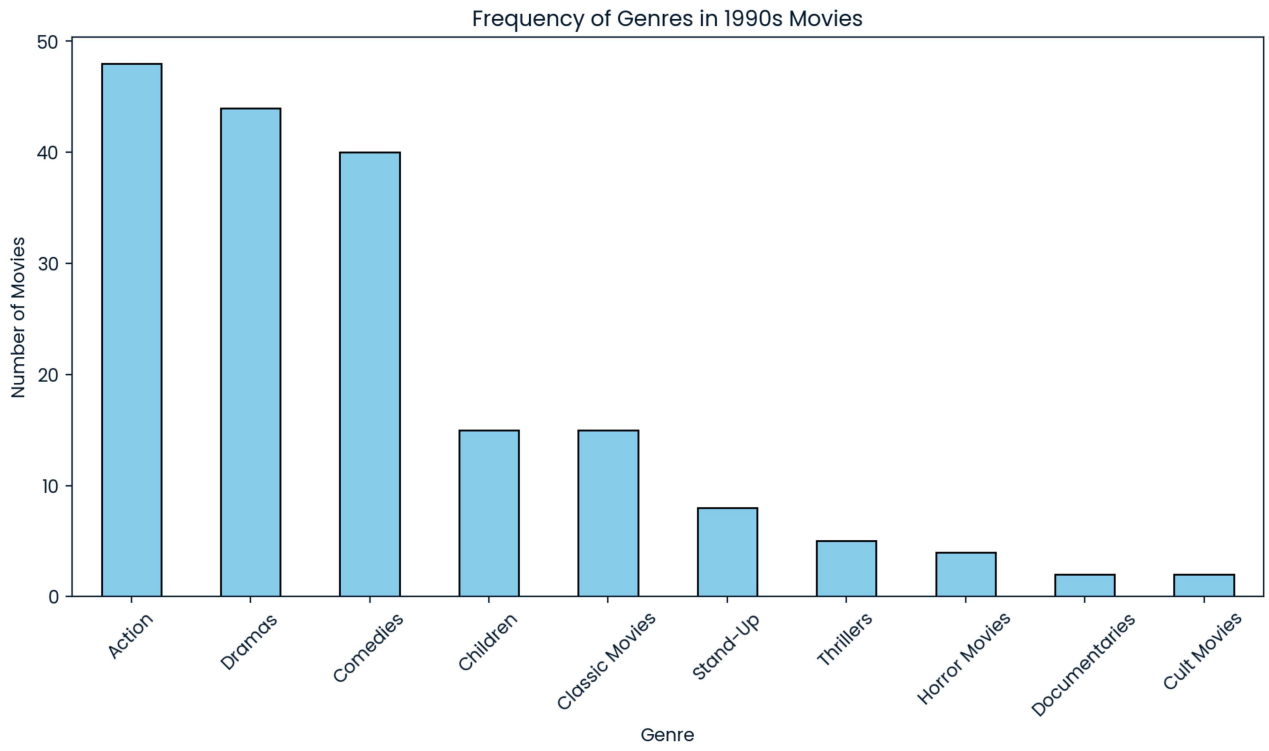



```
Duration statistics per year in the 1990s:
                  mean  median        std
```

```
release_year
1990          117.428571   109.0  41.084742
1991           93.500000    93.5  33.831597
1992          104.000000   102.0  27.892651
1993          120.000000   115.0  35.432565
1994          128.428571   124.0  40.349844
1995          116.437500   110.5  29.289290
1996          120.133333   114.0  27.252435
1997          116.961538   114.5  24.964744
1998          109.884615   103.5  35.631814
1999          121.884615   111.5  33.287027
```

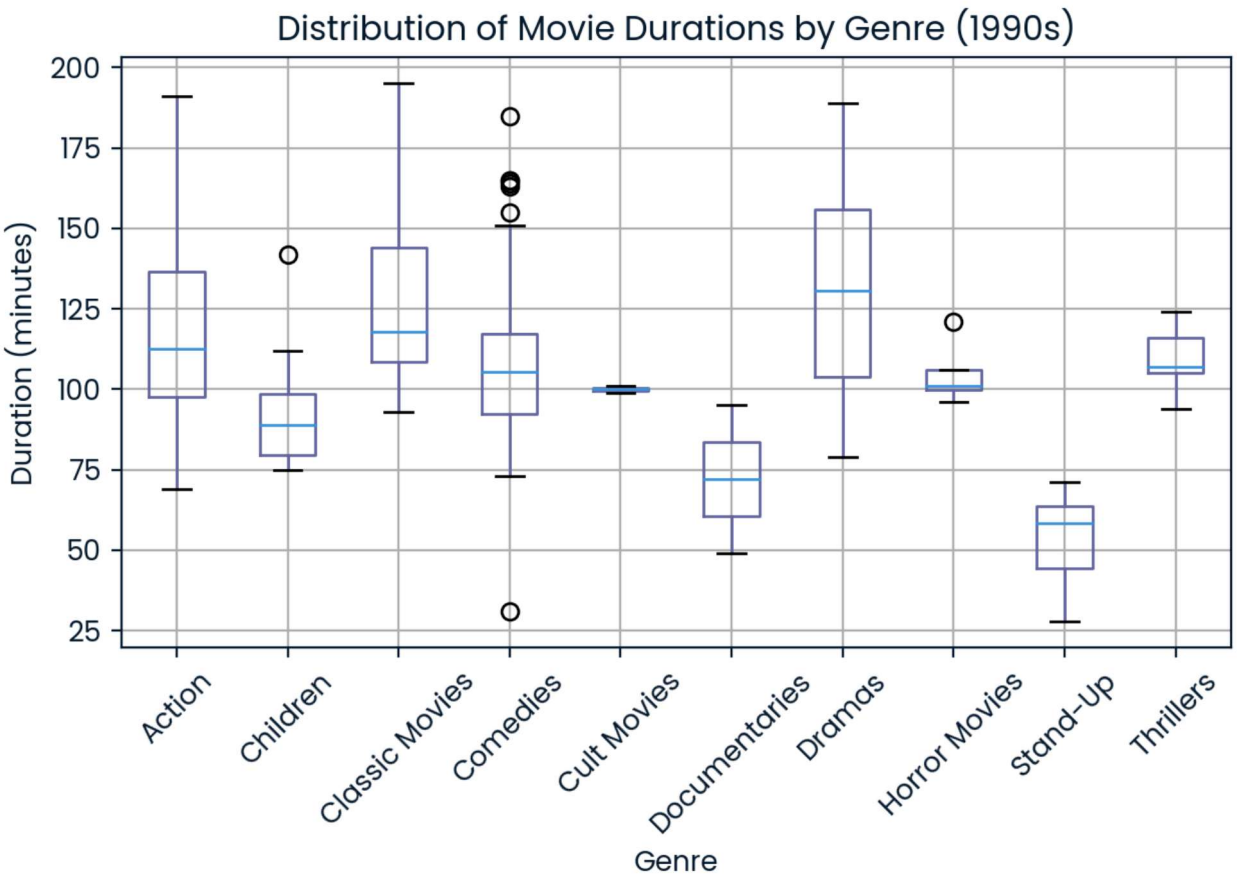



```
Genre Frequency in 1990s Movies:
Action            48
Dramas            44
Comedies          40
Children          15
Classic Movies    15
Stand-Up           8
Thrillers          5
Horror Movies      4
Documentaries      2
Cult Movies        2
Name: genre, dtype: int64
```




```
Number of short action movies (less than 90 minutes) from the 1990s: 7
```

```
<Figure size 1200x800 with 0 Axes>
```



```python
# QUESTION 4: Correlation between release year and duration

# Since 'release_year' and 'duration' are already of datatype int, the Pearson's correlation between these two columns is calculated

correlation = movies_1990s[['release_year', 'duration']].corr()

# Print the correlation matrix, which shows the correlation coefficient between release_year and duration.
print("Correlation between release_year and duration in 1990s movies:")
print(correlation)
```

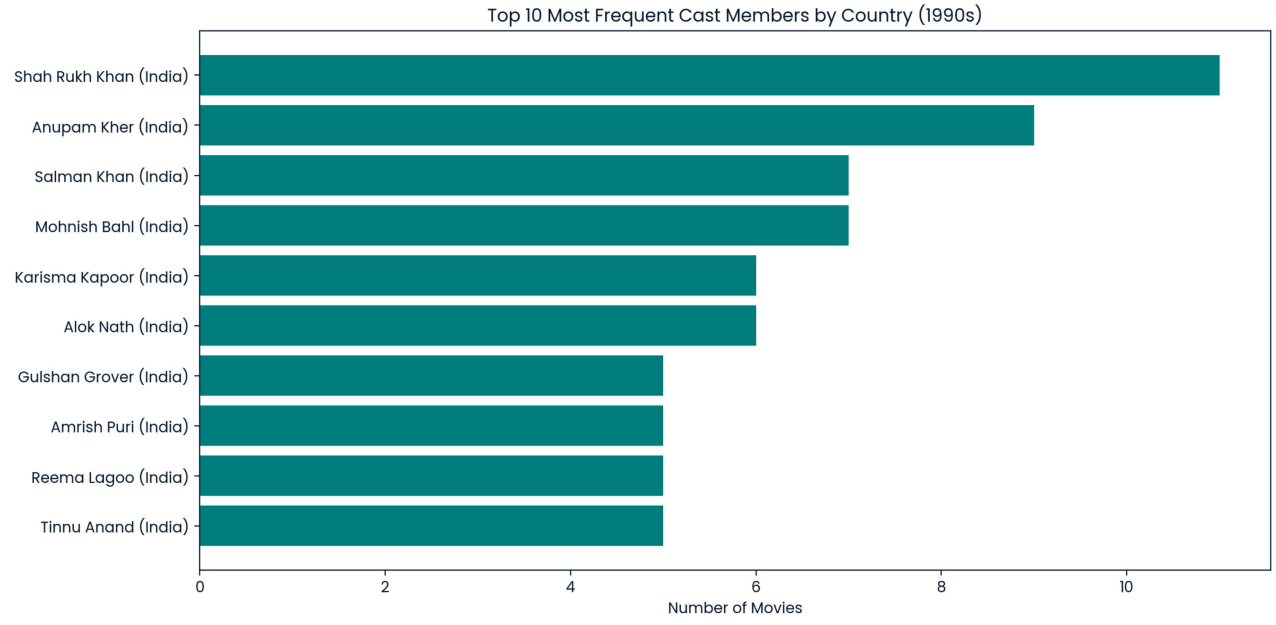
```
Correlation between release_year and duration in 1990s movies:
              release_year  duration
release_year      1.000000  0.096448
duration          0.096448  1.000000
```

```python
# QUESTION 5: Investigating Movie Production Geography and Cast Analysis

   # A. Count the number of movies directed by each director in the 1990s subset
# Remove any missing values from the 'director' column
# Split the comma-separated director names into lists,
# then use explode() to create a new row for each director,
# and finally use str.strip() to remove any extra whitespace.
directors =
movies_1990s['director'].dropna().str.split(',').explode().str.strip()

# Count how many times each director appears in the dataset
director_counts = directors.value_counts()

# Print the top 10 directors by movie count
print("Top Directors in 1990s Movies:")
print(director_counts.head(10))


# B. First, drop missing values in the 'cast' column so that we only work with
available data
# Similarly, remove missing values from the 'cast' column,
# split the comma-separated cast names into lists,
# explode the lists into individual rows, and strip extra whitespace.
cast_members = movies_1990s['cast'].dropna().str.split(',').explode().str.strip()

# Count how many times each cast member appears in the dataset
cast_counts = cast_members.value_counts()

# Print the top 10 most frequent cast members
print("\nMost Frequent Cast Members in 1990s Movies:")
print(cast_counts.head(10))
```

```
Top Directors in 1990s Movies:
Umesh Mehra           4
Johnnie To            4
Mahesh Bhatt          3
Rajkumar Santoshi     3
Gregory Hoblit        3
Youssef Chahine       3
Sooraj R. Barjatya    3
Subhash Ghai          3
Rajiv Mehra           2
Indra Kumar           2
Name: director, dtype: int64

Most Frequent Cast Members in 1990s Movies:
Shah Rukh Khan    12
Anupam Kher        9
Salman Khan        7
Mohnish Bahl       7
```

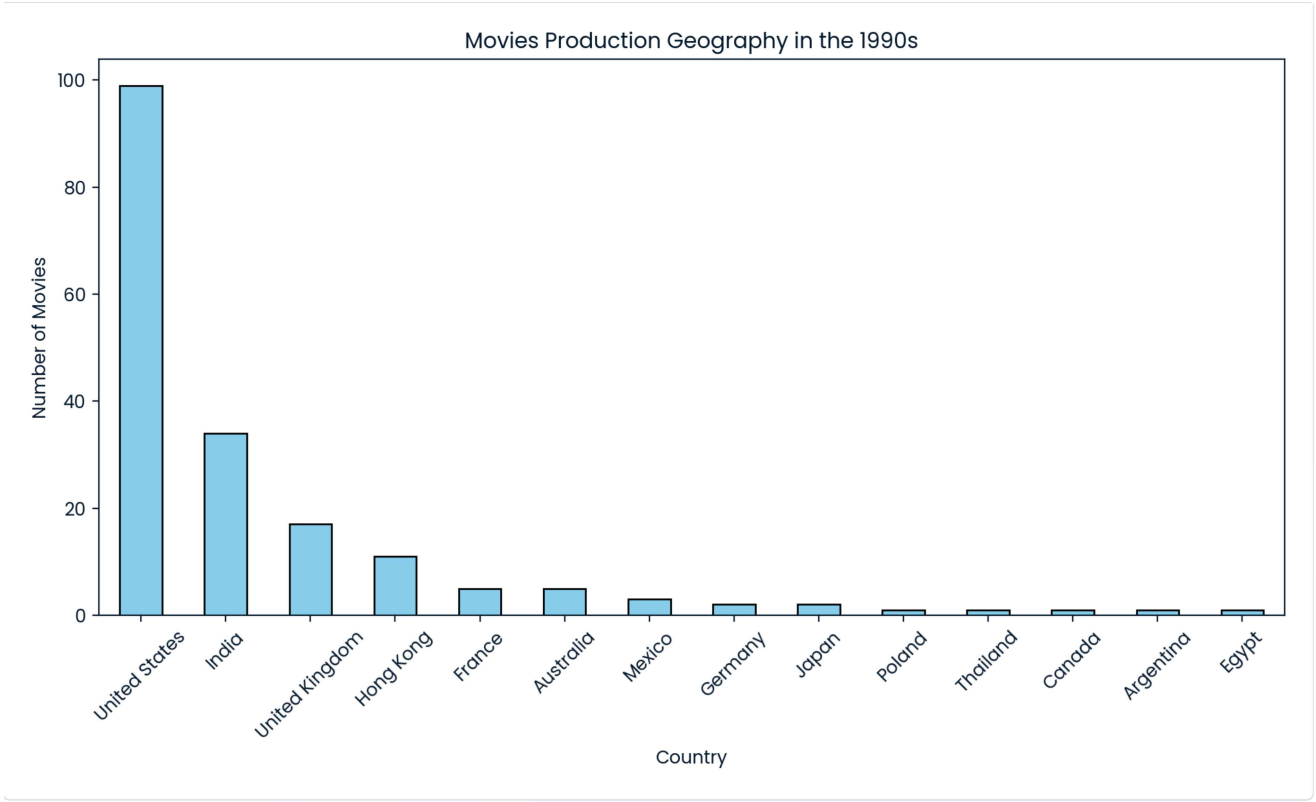
```
Karisma Kapoor     6
Alok Nath          6
Tinnu Anand        5
Gulshan Grover     5
Reema Lagoo        5
Paresh Rawal       5
Name: cast, dtype: int64
```

```
# Step 1: Make a copy and preprocess cast and country columns
movies_cast_country = movies_1990s[['cast', 'country']].dropna()

# Step 2: Split and explode the cast and country columns
movies_cast_country['cast'] = movies_cast_country['cast'].str.split(',')
movies_cast_country['country'] = movies_cast_country['country'].str.split(',')

movies_cast_country = movies_cast_country.explode('cast')
movies_cast_country = movies_cast_country.explode('country')

# Step 3: Clean up whitespaces
movies_cast_country['cast'] = movies_cast_country['cast'].str.strip()
movies_cast_country['country'] = movies_cast_country['country'].str.strip()

# Step 4: Count appearances of cast members by country
cast_counts = movies_cast_country.groupby(['cast',
'country']).size().reset_index(name='count')

# Step 5: Get the top 10 most frequent cast-country combinations
top_cast_country = cast_counts.sort_values('count', ascending=False).head(10)

# Step 6: Plot
plt.figure(figsize=(12, 6))
plt.barh(
    top_cast_country['cast'] + ' (' + top_cast_country['country'] + ')',
    top_cast_country['count'],
    color='teal'
)
plt.xlabel('Number of Movies')
plt.title('Top 10 Most Frequent Cast Members by Country (1990s)')
plt.gca().invert_yaxis()  # Show the most frequent on top
plt.tight_layout()
plt.show()
```



```python
# FOUR: Production Geography

# Group the 1990s movies by the 'country' column and count how many movies each
country produced.
# Each row has only one country.
country_counts = movies_1990s['country'].value_counts()

# Print the country counts to verify the results
print("Number of movies per country in the 1990s:")
print(country_counts)

# Create a bar chart to visualize the production geography.
plt.figure(figsize=(10, 6))         # Set the size of the plot for better
readability
country_counts.plot(kind='bar',      # Plot a bar chart using the country counts
                    color='skyblue',  # Set the color of the bars
                    edgecolor='black')# Add black edges to the bars for clarity
plt.xlabel("Country")               # Label the x-axis
plt.ylabel("Number of Movies")      # Label the y-axis
plt.title("Movies Production Geography in the 1990s")  # Set the title of the
plot
plt.xticks(rotation=45)             # Rotate the x-axis labels by 45 degrees for
readability
plt.tight_layout()                  # Adjust the layout to fit everything nicely
plt.show()                          # Display the plot
```

```
Number of movies per country in the 1990s:
United States     99
India             34
United Kingdom    17
Hong Kong         11
France             5
Australia          5
Mexico             3
Germany            2
Japan              2
Poland             1
Thailand           1
Canada             1
Argentina          1
Egypt              1
Name: country, dtype: int64
```


Movies Production Geography in the 1990s
Number of Movies
0
20
40
60
80
100
United States
India
United Kingdom
Hong Kong
France
Australia
Mexico
Germany
Japan
Poland
Thailand
Canada
Argentina
Egypt
Country

```
# FIVE: Movie Description Analysis

# Word Cloud on Descriptions
from wordcloud import WordCloud
from textblob import TextBlob

# Combine all non-missing movie descriptions into a single string.
descriptions_text = " ".join(movies_1990s['description'].dropna().tolist())

# Create a WordCloud object specifying width, height, and background color.
# Then generate the word cloud from the combined text.
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(descriptions_text)

# Plot the generated word cloud:
plt.figure(figsize=(10, 6))                    # Set the size of the plot
plt.imshow(wordcloud, interpolation='bilinear')  # Display the image with smooth
interpolation
plt.axis('off')                                # Hide the axis lines and labels
plt.title("Word Cloud of 1990s Movie Descriptions")  # Set the plot title
plt.show()                                     # Show the word cloud plot
```