In [50]:

```
import requests
from bs4 import BeautifulSoup
```

In [51]:

```
user_agent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36'
Link_url = 'https://www.imdb.com/chart/top'
response = requests.get(Link_url, headers = {'User-Agent':user_agent,'Accept_language':'en-US, en;q=0.5'})
print(response)
```

```
<Response [200]>
```

In [52]:

```
soup = BeautifulSoup(response.content,'html.parser')
movie_data = soup.findAll('li',attrs = {'class':'ipc-metadata-list-summary-item sc-59b6048d-0 jemTre cli-parent'})
movie_rank = []
movie_name = []
released_year = []
duration = []
rated_type = []
ratings = []
votes = []
```

In [53]:

```
# To convert votes count from Millions and thousands to numbers
def convert_to_int (vote):
    if 'm' in vote:
        return int(float(vote.replace('m','')) * 1000000)
    elif 'k' in vote:
        return int(float(vote.replace('k','')) * 1000)
    else:
        return int(vote)

# Convert duration into minutes
def convert_to_min(r_time):
    total_minutes = 0

    parts = r_time.split()
    for part in parts:
        if 'h' in part:
            total_minutes += int(part.replace('h','')) * 60
        elif 'g' in part:
            total_minutes += int(part.replace('g',''))
    return total_minutes


for movie in movie_data:
    #  Name and rank
    rank_name = movie.find('div',class_="ipc-metadata-list-summary-item__c").div.a.text
    rank = int(rank_name.split('. ')[0])
    name = rank_name.split('. ')[1]
    movie_rank.append(rank)
    movie_name.append(name)

    # year
    year = int(movie.find('div',class_="ipc-metadata-list-summary-item__c").find('span').find_next('span').text)
    released_year.append(year)

    # Duration of movie
    runtime = movie.find('div',class_="ipc-metadata-list-summary-item__c").findAll('span')[2].text
    runtime = convert_to_minutes(runtime)
    duration.append(runtime)

    # Rated Type
    r_type = movie.find('div',class_="ipc-metadata-list-summary-item__tc").find('span').find_next('span').find_next('span').find_next('span'
    rated_type.append(r_type)

    # Ratings
    rating = movie.find('div',class_="ipc-metadata-list-summary-item__c").find('span',class_='ipc-rating-star ipc-rating-star--base ipc-rati
    rating = float(rating)
    ratings.append(rating)

    # votes
    vote = movie.find('div',class_="ipc-metadata-list-summary-item__c").find('span',class_='ipc-rating-star ipc-rating-star--base ipc-rating
    vote = vote.replace('(','').replace(')','')
    vote = convert_votes_to_int(vote)
    votes.append(vote)
```

In [54]:

```
import pandas as pd
import numpy as np
df = pd.DataFrame({"Rank": movie_rank, 'Movie_Name' : movie_name,'Duration(Minutes)' : duration, 'Released_Year' : released_year,'Certificat
df.head(10)
```

Out[54]:

	Rank	Movie_Name	Duration(Minutes)	Released_Year	Certification	Ratings	Votes
0	1	The Shawshank Redemption	142	1994	A	9.3	2800000
1	2	The Godfather	175	1972	A	9.2	2000000
2	3	The Dark Knight	152	2008	UA	9.0	2800000
3	4	The Godfather: Part II	202	1974	A	9.0	1300000
4	5	12 Angry Men	96	1957	U	9.0	834000
5	6	Schindler's List	195	1993	A	9.0	1400000
6	7	The Lord of the Rings: The Return of the King	201	2003	U	9.0	1900000
7	8	Pulp Fiction	154	1994	A	8.9	2200000
8	9	The Lord of the Rings: The Fellowship of the Ring	178	2001	U	8.8	1900000
9	10	Il Buono, Il Brutto, Il Cattivo	161	1966	A	8.8	791000

In [55]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Rank               250 non-null    int64  
 1   Movie_Name         250 non-null    object 
 2   Duration(Minutes)  250 non-null    int64  
 3   Released_Year      250 non-null    int64  
 4   Certification      250 non-null    object 
 5   Ratings            250 non-null    float64
 6   Votes              250 non-null    int64  
dtypes: float64(1), int64(4), object(2)
memory usage: 13.8+ KB
```

In [56]:

```
df.shape
```

Out[56]:

```
(250, 7)
```

In [57]:

```
df.size
```

Out[57]:

```
1750
```

In [58]:

```
df.ndim
```

Out[58]:

```
2
```

In [59]:

```
df.describe()
```

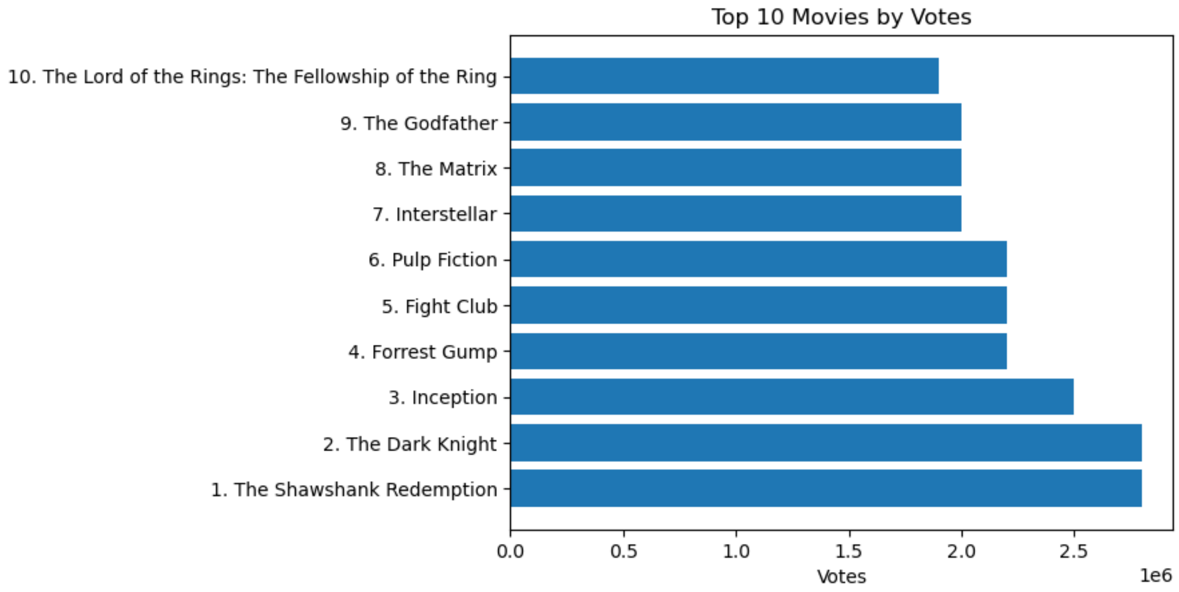
Out[59]:

	Rank	Duration(Minutes)	Released_Year	Ratings	Votes
count	250.000000	250.000000	250.000000	250.000000	2.500000e+02
mean	125.500000	129.012000	1986.816000	8.307600	6.719320e+05
std	72.312977	29.756236	25.387086	0.232462	5.418230e+05
min	1.000000	45.000000	1921.000000	8.000000	3.600000e+04
25%	63.250000	107.250000	1966.250000	8.100000	2.295000e+05
50%	125.500000	126.500000	1994.500000	8.200000	5.370000e+05
75%	187.750000	145.750000	2007.000000	8.400000	9.910000e+05
max	250.000000	238.000000	2023.000000	9.300000	2.800000e+06

In [60]:

```
top_10_movies = df.sort_values(by='Votes', ascending=False).head(10)

# Add a new column 'Rank' for displaying the rank of each movie
top_10_movies['Rank'] = np.arange(1, len(top_10_movies) + 1)

# Display the top 10 movies with their rank, name, and votes
print(top_10_movies[['Rank', 'Movie_Name', 'Votes']])

# Extract Movie Names and Votes
Movie_Name = top_10_movies['Movie_Name']
Votes = top_10_movies['Votes']
Rank = top_10_movies['Rank']

y_pos = np.arange(len(Movie_Name))
plt.barh(y_pos[::-1], Votes[::-1], align='center')  # Reverse the order
plt.yticks(y_pos[::-1], [f'{r}. {m}' for r, m in zip(Rank[::-1], Movie_Name[::-1])])  # Reverse the order
plt.xlabel('Votes')
plt.title('Top 10 Movies by Votes')

plt.show()
```

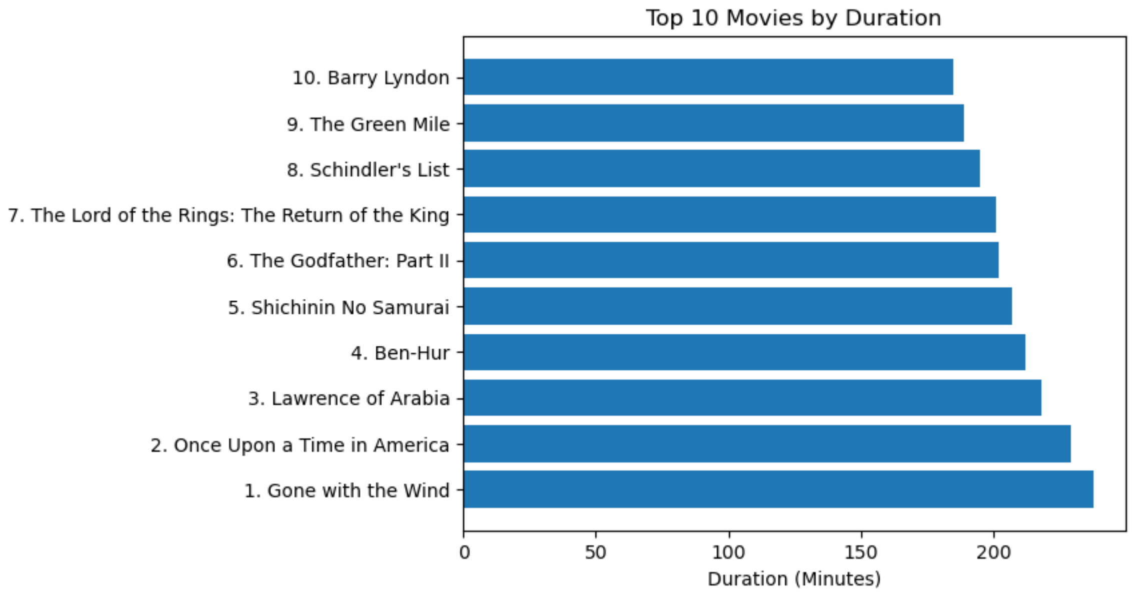
```
    Rank                                      Movie_Name    Votes
0      1                        The Shawshank Redemption  2800000
2      2                                 The Dark Knight  2800000
13     3                                       Inception  2500000
10     4                                    Forrest Gump  2200000
11     5                                      Fight Club  2200000
7      6                                    Pulp Fiction  2200000
22     7                                    Interstellar  2000000
15     8                                      The Matrix  2000000
1      9                                   The Godfather  2000000
8     10  The Lord of the Rings: The Fellowship of the Ring  1900000
```



In [63]:

```python
top_10_duration = df.sort_values(by='Duration(Minutes)', ascending=False).head(10)

# Add a new column 'Rank' for displaying the rank of each movie
top_10_duration['Rank'] = np.arange(1, len(top_10_duration) + 1)

# Display the top 10 movies with their rank, name, and duration
print(top_10_duration[['Rank', 'Movie_Name', 'Duration(Minutes)']])

# Extract Movie Names and Durations
Movie_Name_duration = top_10_duration['Movie_Name']
Duration = top_10_duration['Duration(Minutes)']
Rank_duration = top_10_duration['Rank']

y_pos = np.arange(len(Movie_Name_duration))

# Create the horizontal bar chart
plt.barh(y_pos, Duration, align='center')
plt.yticks(y_pos, [f'{r}. {m}' for r, m in zip(Rank_duration, Movie_Name_duration)])
plt.xlabel('Duration (Minutes)')
plt.title('Top 10 Movies by Duration')

plt.show()
```

```
     Rank                              Movie_Name  Duration(Minutes)
159     1                      Gone with the Wind                238
82      2             Once Upon a Time in America                229
97      3                      Lawrence of Arabia                218
184     4                                 Ben-Hur                212
21      5                     Shichinin No Samurai                207
3       6                  The Godfather: Part II                202
6       7  The Lord of the Rings: The Return of the King         201
5       8                        Schindler's List                195
27      9                          The Green Mile                189
186    10                            Barry Lyndon                185
```



In [66]:

```python
min_yr = df['Released_Year'].min()
print(min_yr)
```

```
1921
```

In [67]:

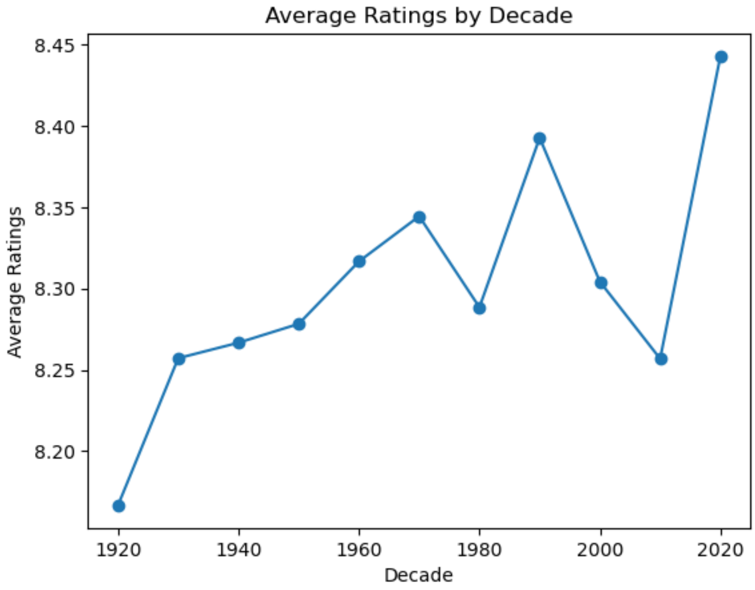
```python
max_yr = df['Released_Year'].max()
print(max_yr)
```

```
2023
```

In [68]:

```python
df['Decade'] = (df['Released_Year'] // 10) * 10
```

```
# Calculate the average ratings for each decade
average_ratings_by_decade = df.groupby('Decade')['Ratings'].mean()

# Plot the average ratings in a line chart
plt.plot(average_ratings_by_decade.index, average_ratings_by_decade.values, marker='o', linestyle='-')

# Set labels and title
plt.xlabel('Decade')
plt.ylabel('Average Ratings')
plt.title('Average Ratings by Decade')

# Show the plot
plt.show()
```

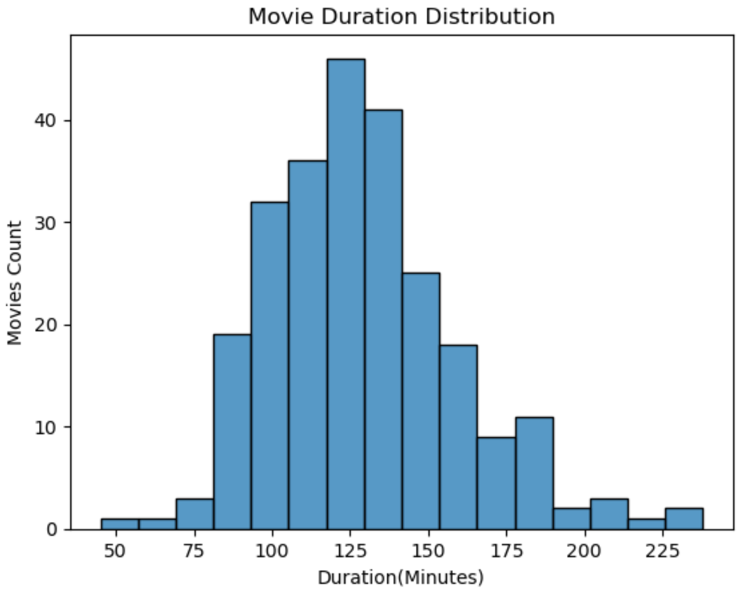


In [69]:

```
import seaborn as sns
sns.histplot(x=df['Duration(Minutes)'])
plt.title('Movie Duration Distribution')
plt.ylabel('Movies Count')
```

Out[69]:

```
Text(0, 0.5, 'Movies Count')
```

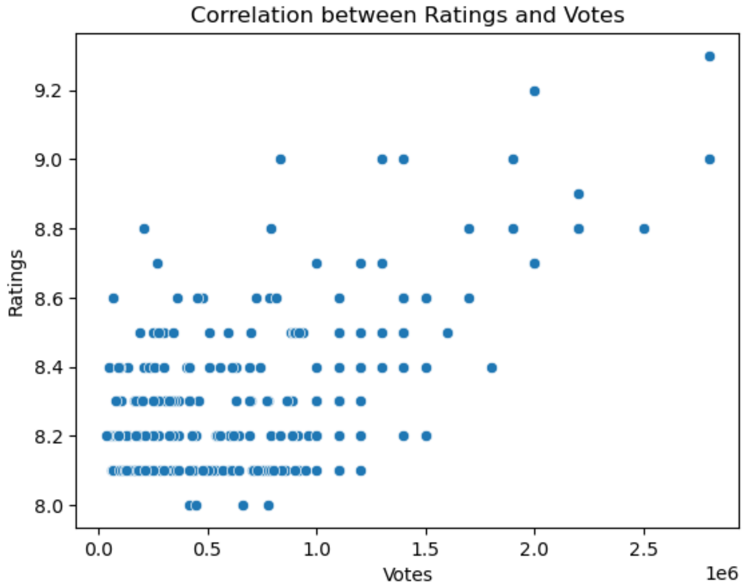


In [73]:

```
from wordcloud import WordCloud
high_rated_movies = df[df['Ratings'] > 8.0]

# Create a string containing movie names and their ratings
movie_ratings_text = ' '.join(f"{name} {rating}" for name, rating in zip(high_rated_movies['Movie_Name'], high_rated_movies['Ratings']))

# Generate the word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(movie_ratings_text)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

```
In [70]: sns.scatterplot(x=df['Votes'],y=df['Ratings'])
         plt.title('Correlation between Ratings and Votes')
```

```
Out[70]: Text(0.5, 1.0, 'Correlation between Ratings and Votes')
```



```
In [75]: Q1 = df['Votes'].quantile(0.25)
         Q3 = df['Votes'].quantile(0.75)
         IQR = Q3 - Q1

         # Define the lower and upper bounds for outliers
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR

         # Find outliers
         outliers = df[(df['Votes'] < lower_bound) | (df['Votes'] > upper_bound)]

         print("Outliers:")
         print(outliers)
```

```
Outliers:
    Rank                Movie_Name  Duration(Minutes)  Released_Year  \
0      1  The Shawshank Redemption                142           1994
2      3           The Dark Knight                152           2008
7      8              Pulp Fiction                154           1994
10    11              Forrest Gump                142           1994
11    12                Fight Club                139           1999
13    14                 Inception                148           2010

   Certification  Ratings    Votes  Decade
0              A      9.3  2800000    1990
2             UA      9.0  2800000    2000
7              A      8.9  2200000    1990
10            UA      8.8  2200000    1990
11             A      8.8  2200000    1990
13            UA      8.8  2500000    2010
```

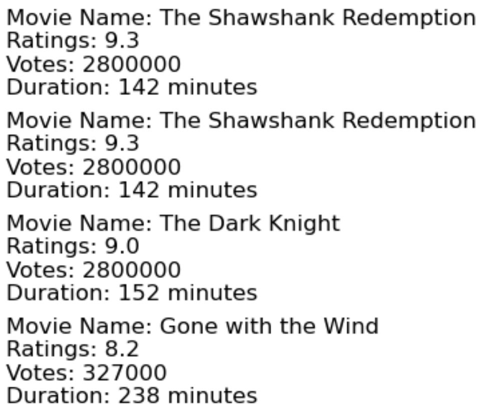
```
In [76]: from sklearn.linear_model import LinearRegression
         X = df[['Votes']]

         # Fit a linear regression model
         model = LinearRegression()
         model.fit(X, df['Votes'])

         # Predict 'Votes' using the linear regression model
         predicted_votes = model.predict(X)

         # Calculate the residuals
         residuals = df['Votes'] - predicted_votes
```

```
# Identify potential outliers (e.g., points with residuals > 2 standard deviations)
outliers = df[np.abs(residuals) > 2 * np.std(residuals)]

# Print the potential outliers
print("Potential Outliers:")
print(outliers)
```

```
Potential Outliers:
Empty DataFrame
Columns: [Rank, Movie_Name, Duration(Minutes), Released_Year, Certification, Ratings, Votes, Decade]
Index: []
```

In [77]:

```
highest_rating_movie = df[df['Ratings'] == df['Ratings'].max()]

# Find the movie with the maximum votes
max_votes_movie = df[df['Votes'] == df['Votes'].max()]

# Find the movie with the maximum duration
max_duration_movie = df[df['Duration(Minutes)'] == df['Duration(Minutes)'].max()]

# Combine all the relevant information
combined_info = pd.concat([highest_rating_movie, max_votes_movie, max_duration_movie], ignore_index=True)

# Create a figure to display the information
fig, ax = plt.subplots(figsize=(10, 5))

# Plot the information
for index, row in combined_info.iterrows():
    ax.text(0.1, 0.9 - index*0.2, f"Movie Name: {row['Movie_Name']}\nRatings: {row['Ratings']}\nVotes: {row['Votes']}\nDuration: {row['Durat
            fontsize=12, ha='left', va='center', transform=ax.transAxes)

# Remove the axes for a cleaner look
ax.axis('off')

plt.show()
```



Movie Name: The Shawshank Redemption
Ratings: 9.3
Votes: 2800000
Duration: 142 minutes

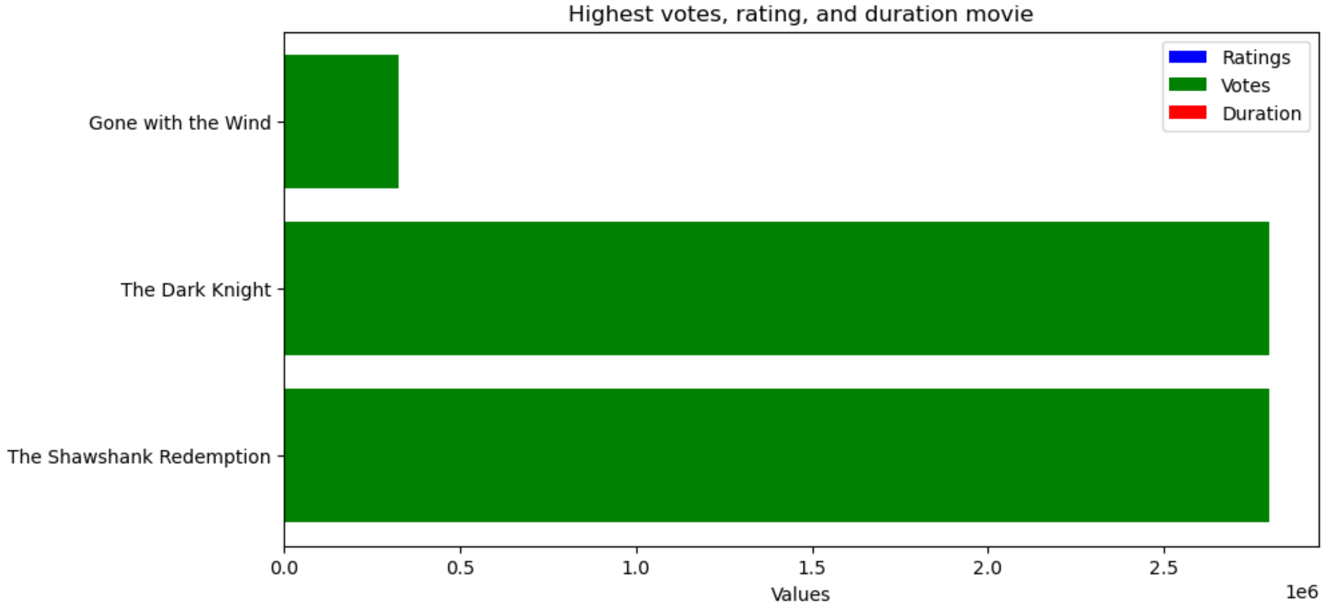
Movie Name: The Shawshank Redemption
Ratings: 9.3
Votes: 2800000
Duration: 142 minutes

Movie Name: The Dark Knight
Ratings: 9.0
Votes: 2800000
Duration: 152 minutes

Movie Name: Gone with the Wind
Ratings: 8.2
Votes: 327000
Duration: 238 minutes

In [79]:

```
highest_rating_movie = df[df['Ratings'] == df['Ratings'].max()]

# Find the movie with the maximum votes
max_votes_movie = df[df['Votes'] == df['Votes'].max()]

# Find the movie with the maximum duration
max_duration_movie = df[df['Duration(Minutes)'] == df['Duration(Minutes)'].max()]

# Combine all the relevant information
combined_info = pd.concat([highest_rating_movie, max_votes_movie, max_duration_movie], ignore_index=True)

# Create a bar chart to display the information
fig, ax = plt.subplots(figsize=(10, 5))

# Plot the information
ax.barh(combined_info['Movie_Name'], combined_info['Ratings'], color='blue', label='Ratings')
ax.barh(combined_info['Movie_Name'], combined_info['Votes'], color='green', label='Votes')
ax.barh(combined_info['Movie_Name'], combined_info['Duration(Minutes)'], color='red', label='Duration')

# Add labels and legend
ax.set_xlabel('Values')
ax.set_title('Highest votes, rating, and duration movie')
ax.legend()

plt.show()
```


Highest votes, rating, and duration movie
Ratings
Votes
Duration
Gone with the Wind
The Dark Knight
The Shawshank Redemption
0.0
0.5
1.0
1.5
2.0
2.5
1e6
Values

/