```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load the dataset
df = pd.read_csv("/content/Highest Holywood Grossing Movies.csv")
```

```
df.head()
```

	Unnamed: 0	Title	Movie Info	Year	Distributor	Budget (in $)	Domestic Opening (in $)	Domestic Sales (in $)
0	0	Avatar	A paraplegic Marine dispatched to the moon Pan...	2009	Twentieth Century Fox	237000000.0	77025481.0	785221649
1	1	Avengers: Endgame	After the devastating events of Avengers: Infi...	2019	Walt Disney Studios Motion Pictures	356000000.0	357115007.0	858373000
2	2	Avatar: The Way of Water	Jake Sully lives with his newfound family form...	2022	20th Century Studios	142022.0	134100226.0	684075767
			A seventeen-year-old		Paramount			

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Unnamed: 0                  1000 non-null   int64
```

```
 1   Title                      1000 non-null   object
 2   Movie Info                 1000 non-null   object
 3   Year                       1000 non-null   int64
 4   Distributor                1000 non-null   object
 5   Budget (in $)              1000 non-null   float64
 6   Domestic Opening (in $)    1000 non-null   float64
 7   Domestic Sales (in $)      1000 non-null   int64
 8   International Sales (in $) 1000 non-null   int64
 9   World Wide Sales (in $)    1000 non-null   int64
 10  Release Date               1000 non-null   object
 11  Genre                      1000 non-null   object
 12  Running Time               1000 non-null   object
 13  License                    1000 non-null   object
dtypes: float64(2), int64(5), object(7)
memory usage: 109.5+ KB
```

```
df.describe()
```

	Unnamed: 0	Year	Budget (in $)	Domestic Opening (in $)	Domestic Sales (in $)	International Sales (in $)
count	1000.000000	1000.000000	1.000000e+03	1.000000e+03	1.000000e+03	1.000000e+03
mean	499.500000	2008.181000	7.666897e+07	4.150137e+07	1.646405e+08	2.640890e+08
std	288.819436	10.585854	6.684609e+07	3.934224e+07	1.197541e+08	2.133847e+08
min	0.000000	1937.000000	1.201700e+04	2.000000e+00	6.752000e+03	2.450000e+07
25%	249.750000	2002.000000	1.800000e+07	1.827249e+07	9.572506e+07	1.321190e+08
50%	499.500000	2010.000000	6.950000e+07	3.161454e+07	1.349169e+08	1.941077e+08
75%	749.250000	2016.000000	1.250000e+08	5.389436e+07	1.983993e+08	3.188993e+08

Start coding or generate with AI.

```
# Clean currency columns
currency_cols = ['Budget (in $)', 'Domestic Opening (in $)']
for col in currency_cols:
    df[col] = df[col].replace('[^0-9]', '', regex=True).replace('', np.nan).astype(float)
```

```
# Fill missing values
df['Distributor'].fillna('Unknown', inplace=True)
df['License'].fillna('Unrated', inplace=True)
df[currency_cols] = df[currency_cols].fillna(df[currency_cols].median())
```

```
<ipython-input-4-303521872>:2: FutureWarning: A value is trying to be set on a copy of a
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col
```

```
  df['Distributor'].fillna('Unknown', inplace=True)
<ipython-input-4-303521872>:3: FutureWarning: A value is trying to be set on a copy of a
The behavior will change in pandas 3.0. This inplace method will never work because the

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col
```

```
  df['License'].fillna('Unrated', inplace=True)
```

```
# Prepare feature set and target
features = df[['Budget (in $)', 'Domestic Opening (in $)', 'Domestic Sales (in $)', 'Interna
target = df['World Wide Sales (in $)']
```

```
# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_
```

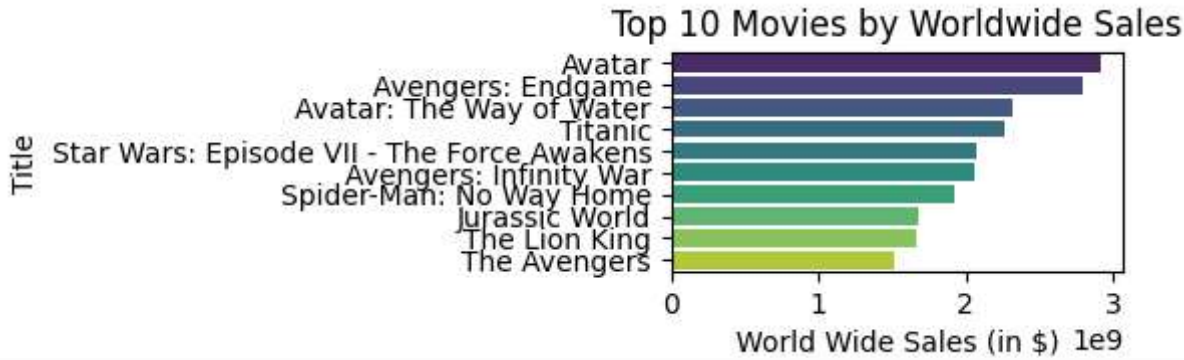
```
# Convert Running Time to minutes
df['Running Time (min)'] = df['Running Time'].str.extract(r'(\d+)\s*hr.*?(\d+)?\s*min').fill
```

```
# Plotting
plt.figure(figsize=(20, 25))
```

```
<Figure size 2000x2500 with 0 Axes>
<Figure size 2000x2500 with 0 Axes>
```

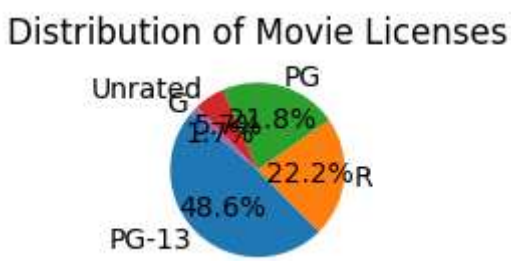
```
# 1. Bar plot - Top 10 movies by worldwide sales
plt.subplot(3, 2, 1)
top10 = df.sort_values(by='World Wide Sales (in $)', ascending=False).head(10)
sns.barplot(data=top10, x='World Wide Sales (in $)', y='Title', palette='viridis')
plt.title('Top 10 Movies by Worldwide Sales')
```

```
<ipython-input-13-48650792>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.barplot(data=top10, x='World Wide Sales (in $)', y='Title', palette='viridis')
Text(0.5, 1.0, 'Top 10 Movies by Worldwide Sales')
```




```
# 2. Pie chart - Distribution by License
plt.subplot(3, 2, 2)
license_counts = df['License'].value_counts()
plt.pie(license_counts, labels=license_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Movie Licenses')
```

```
Text(0.5, 1.0, 'Distribution of Movie Licenses')
```

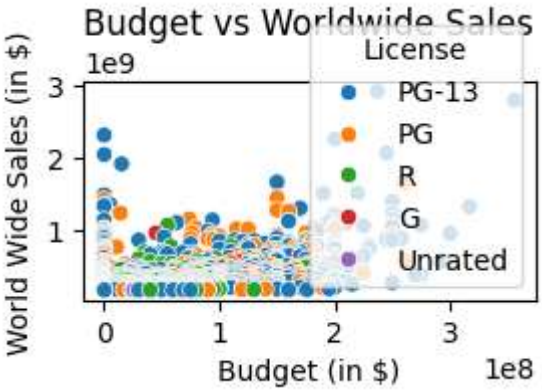



```
# 3. Line plot - Movies released per year
plt.subplot(3, 2, 3)
yearly_count = df['Year'].value_counts().sort_index()
sns.lineplot(x=yearly_count.index, y=yearly_count.values)
plt.title('Movies Released per Year')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
```

```
Text(0, 0.5, 'Number of Movies')
```


```
# 4. Scatter plot - Budget vs Worldwide Sales
plt.subplot(3, 2, 4)
sns.scatterplot(data=df, x='Budget (in $)', y='World Wide Sales (in $)', hue='License')
plt.title('Budget vs Worldwide Sales')
```

```
Text(0.5, 1.0, 'Budget vs Worldwide Sales')
```




```
# 5. Histogram - Running time distribution
plt.subplot(3, 2, 5)
sns.histplot(df['Running Time (min)'], bins=20, kde=True)
plt.title('Distribution of Movie Running Time')

plt.tight_layout()
plt.show()
```