```
import pandas as pd
df = pd.read_csv('tn.movie_budgets.csv')
df.head(10)
```

	id	release_date	movie	production_budget	domestic_gross	worldwide_gro
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,2
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,8
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,3
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,9
			Star Wars			

```
# Convert release_date column to datetime data type
df['release_date'] = pd.to_datetime(df['release_date'])
```

The code turms the date column into uniform date for better data presentation using the df['release date'] function

```
df.head()
```

	id	release_date	movie	production_budget	domestic_gross	worldwide_gro
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625	$2,776,345,2
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,8
			Dark			

```
# Check for duplicated data
print(df.duplicated().sum())
```

```
0
```

```
# Check for null values
print(df.isnull().sum())
```

```
id                 0
release_date       0
movie              0
production_budget  0
domestic_gross     0
worldwide_gross    0
dtype: int64
```

Checking for null values using the .isnull fuctions confirms if there are any null values and for our case there are no null values in all the columns

```
# Remove leading and trailing whitespace from all string columns in a DataFrame
df = df.apply(lambda x: x.str.strip() if x.dtype == 'object' else x)
df.head()
```

	id	release_date	movie	production_budget	domestic_gross	worldwide_gro
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625	$2,776,345,2

By standardizing the format of data in a Pandas DataFrame, we can ensure that the data is consistent and ready for analysis.

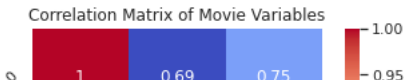
			Caribbean:			
1	2	2011-05-20		$410,600,000	$241,063,875	$1,045,663,8

```
# Remove commas and dollar signs from the budget and gross columns
df['production_budget'] = df['production_budget'].str.replace(',', '').str.replace('$', '').astype(float)
df['domestic_gross'] = df['domestic_gross'].str.replace(',', '').str.replace('$', '').astype(float)
df['worldwide_gross'] = df['worldwide_gross'].str.replace(',', '').str.replace('$', '').astype(float)

# Calculate the correlation matrix
correlation_matrix = df[['production_budget', 'domestic_gross', 'worldwide_gross']].corr()

# Print the correlation matrix

print(correlation_matrix)
```

```
                    production_budget  domestic_gross  worldwide_gross
production_budget           1.000000        0.685682         0.748306
domestic_gross              0.685682        1.000000         0.938853
worldwide_gross             0.748306        0.938853         1.000000
<ipython-input-11-9ae3fad455a8>:2: FutureWarning: The default value of regex will change from True to False in a future version. In
  df['production_budget'] = df['production_budget'].str.replace(',', '').str.replace('$', '').astype(float)
<ipython-input-11-9ae3fad455a8>:3: FutureWarning: The default value of regex will change from True to False in a future version. In
  df['domestic_gross'] = df['domestic_gross'].str.replace(',', '').str.replace('$', '').astype(float)
<ipython-input-11-9ae3fad455a8>:4: FutureWarning: The default value of regex will change from True to False in a future version. In
  df['worldwide_gross'] = df['worldwide_gross'].str.replace(',', '').str.replace('$', '').astype(float)
```

first remove the dollar sign and the commers on the data set using the str.replace() method. Then, it uses the astype() method with the argument float to convert these columns into float values. this enable to run the correlation matrix code that shows us the correlation between 'production_budget', 'domestic_gross', 'worldwide_gross' data

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

# create correlation matrix
corr = [[1.000000, 0.685682, 0.748306],
        [0.685682, 1.000000, 0.938853],
        [0.748306, 0.938853, 1.000000]]

# set variable names
variables = ['Production Budget', 'Domestic Gross', 'Worldwide Gross']

# create heatmap using seaborn
sns.set(style='white')
sns.heatmap(corr, cmap='coolwarm', annot=True, square=True)

# add title and axis labels
plt.title('Correlation Matrix of Movie Variables')
plt.xlabel('Movie Variables')
plt.ylabel('Movie Variables')

# rotate x and y axis labels
plt.xticks(rotation=45)
plt.yticks(rotation=45)

# show plot
plt.show()
```



This code generates a heatmap using the Seaborn library to visualize a correlation matrix. The correlation matrix is a 3x3 matrix that contains the correlation coefficients between three movie variables.

The heatmap shows the correlation values using colors, where darker colors represent higher correlation values. The cmap parameter in the sns.heatmap function sets the color palette to 'coolwarm', which ranges from cool (blue) to warm (red).

The annot parameter adds the correlation values to each cell in the heatmap. The square parameter ensures that the heatmap has square cells.

Finally, the plt.title, plt.xlabel, and plt.ylabel functions add a title and axis labels to the plot. The plt.show function displays the plot.