Step 1: Understand the **Dataset**

```
#Load the Data
import pandas as pd
data = pd.read_csv('/content/hotstar.csv')
```

```
#Check the Structure
data.head()
```

	hotstar_id	title	description	genre	year	age_rating	running_time	type
0	1000087439	Sambha - Aajcha Chawa	A young man sets off on a mission to clean up ...	Action	2012	U/A 16+	141.0	movie
1	1260023113	Cars Toon: Mater And The Ghostlight	Mater is haunted by a mysterious blue light th...	Animation	2006	U	7.0	movie
2	1260103188	Kanmani Rambo Khatija	Unlucky since birth, Rambo finds hope when he ...	Romance	2022	U/A 16+	157.0	movie
3	1260103754	[illegible]	While trying to rescue her sister's kids	[illegible]	2023	U/A 16+	139.0	movie

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

```
#to see the column names, data types, and non-null counts
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6874 entries, 0 to 6873
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   hotstar_id    6874 non-null   int64
 1   title         6874 non-null   object
 2   description   6874 non-null   object
 3   genre         6874 non-null   object
 4   year          6874 non-null   int64
 5   age_rating    6874 non-null   object
 6   running_time  4568 non-null   float64
 7   type          6874 non-null   object
dtypes: float64(1), int64(2), object(5)
memory usage: 429.8+ KB
```

```
#Check the number of rows and columns
data.shape
```

```
(6874, 8)
```

Step 2: Check for Missing **Values**

```
#Check for missing values across each column
data.isnull().sum()
```

	0
hotstar_id	0
title	0
description	0
genre	0
year	0
age_rating	0
running_time	2306
type	0

```
#For numeric data, fill missing values with the mean, median, or a specific value
data['running_time'].fillna(data['running_time'].mean(), inplace=True)
```

```
<ipython-input-7-a8dc4c169b6c>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

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

  data['running_time'].fillna(data['running_time'].mean(), inplace=True)
```

Step 3: Check Data **Types**

```
#Inspect Data Types
data['year'] = pd.to_numeric(data['year'], errors='coerce')
```

```
#For categorical columns, check for inconsistencies in spelling
data['genre'] = data['genre'].str.capitalize()
```

Step 4: Descriptive **Statistics**

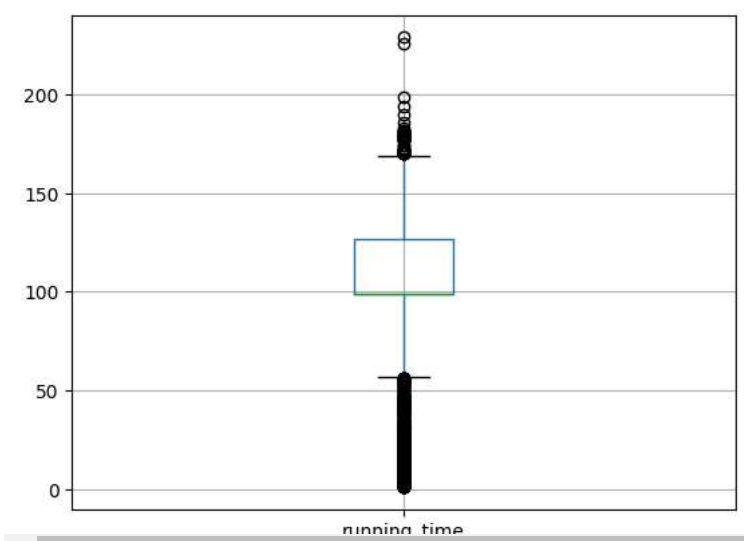
```
#Summary Statistics for Numerical Columns
data.describe()
```

	hotstar_id	year	running_time
count	6.874000e+03	6874.000000	6874.000000
mean	1.059077e+09	2011.718650	98.746716
std	4.812666e+08	11.936894	40.277937
min	3.000000e+00	1928.000000	1.000000
25%	1.000088e+09	2009.000000	98.746716
50%	1.260008e+09	2016.000000	98.746716
75%	1.260099e+09	2019.000000	127.000000
max	1.837059e+09	2023.000000	229.000000

```
#Category Counts
data['genre'].value_counts()
```

genre	count
Drama	2043
Comedy	791
Romance	642
Action	619
Reality	401
Thriller	352
Family	263
Animation	240
Documentary	207
Sport	180
Animals & nature	119
Horror	118
Kids	104
Crime	99
Mythology	81
Talk show	73
Superhero	63
Standup comedy	51
Adventure	49
Biopic	47
Mystery	42
Historical	42
Science fiction	41
Science	34
Teen	31
Awards	28
Lifestyle	24
Food	20
Concert film	18
Musical	16
Fantasy	11
Shorts	10
Travel	7
Docudrama	5
Formula e	1
Football	1
Kabaddi	1

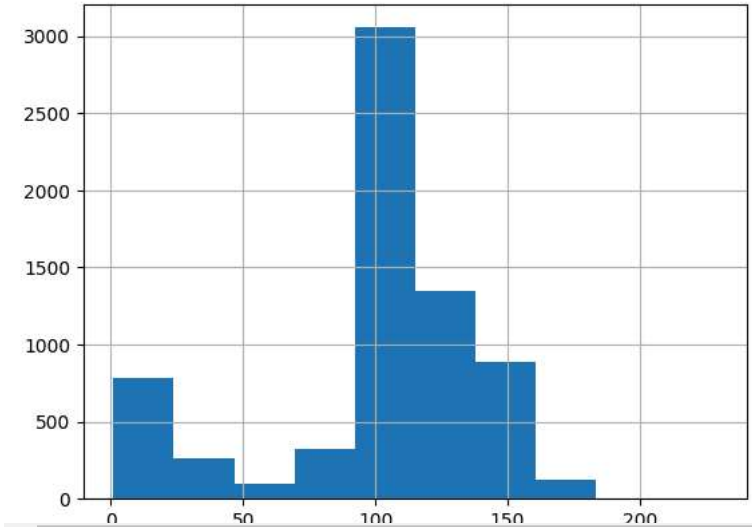
```
#Identify Outliers
import matplotlib.pyplot as plt
data.boxplot(column='running_time')
plt.show()
```



Step 5: Visualize Data **Distributions**

```
#histograms to see the distribution of numeric column
data['running_time'].hist()
```

```
<Axes: >
```

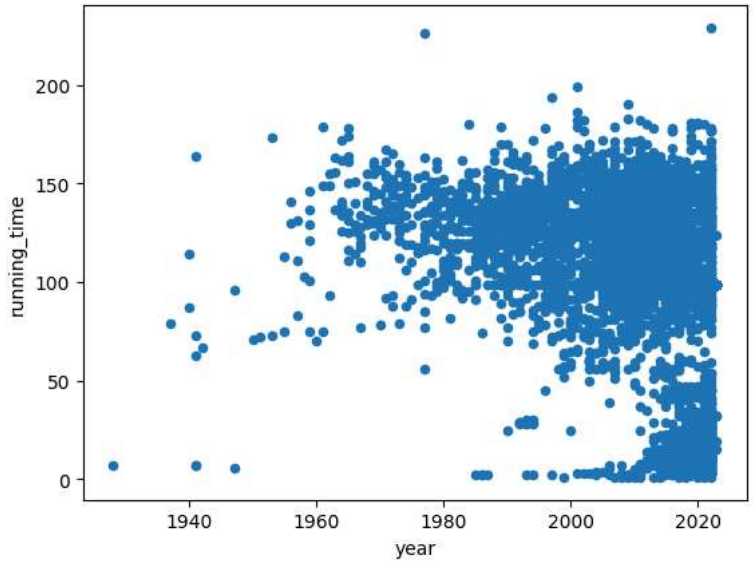



```
#bar charts for categorical data like genre to see the count of each category
data['genre'].value_counts().plot(kind='bar')
```

```
<Axes: xlabel='genre'>
```


```
#scatter plots to explore relationships, such as between running_time and release_year
data.plot.scatter(x='year', y='running_time')
```

```
<Axes: xlabel='year', ylabel='running_time'>
```



Step 6: Correlation **Analysis**

```
# Select only numerical features for correlation calculation.
numerical_data = data.select_dtypes(include=['number'])

# Calculate the correlation matrix.
correlation = numerical_data.corr()

# Print the correlation matrix.
print(correlation)
```

```
              hotstar_id      year  running_time
hotstar_id      1.000000  0.013541     -0.194855
year            0.013541  1.000000     -0.344752
running_time   -0.194855 -0.344752      1.000000
```

```
#Visualize correlations using a heatmap
import seaborn as sns
sns.heatmap(correlation, annot=True, cmap='coolwarm')
```

```
<Axes: >
```