```
# CloudWatch_Traffic_Web_Attack
# Author: Kshitija Agrawal
# Skills: Python (Pandas, numpy, Matplotlib, Seaborn)
```

```
# 1. Import libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# 2. Display settings

pd.set_option('display.max_columns', None)
sns.set_style("whitegrid")
```

```
# 3. Load Dataset

df = pd.read_csv("/content/CloudWatch_Traffic_Web_Attack.csv")
```

```
df.head()
```

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_code	protocol	response.code	dst_port
0	5602	12990	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	147.161.161.82	AE	HTTPS	200	443
1	30912	18186	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.33.6	US	HTTPS	200	443
2	28506	13468	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.212.255	CA	HTTPS	200	443
3	30546	14278	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	136.226.64.114	US	HTTPS	200	443
4	6526	13892	2024-04-25T23:00:00Z	2024-04-25T23:10:00Z	165.225.240.79	NL	HTTPS	200	443

```
# 4. Basic Dataset Information

print("Shape of dataset:", df.shape)
```

```
Shape of dataset: (282, 16)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 282 entries, 0 to 281
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   bytes_in             282 non-null    int64
 1   bytes_out            282 non-null    int64
 2   creation_time        282 non-null    object
 3   end_time             282 non-null    object
 4   src_ip               282 non-null    object
 5   src_ip_country_code  282 non-null    object
 6   protocol             282 non-null    object
 7   response.code        282 non-null    int64
 8   dst_port             282 non-null    int64
 9   dst_ip               282 non-null    object
 10  rule_names           282 non-null    object
 11  observation_name     282 non-null    object
 12  source.meta          282 non-null    object
 13  source.name          282 non-null    object
 14  time                 282 non-null    object
 15  detection_types      282 non-null    object
dtypes: int64(4), object(12)
memory usage: 35.4+ KB
```

```
# 5. Statistical Summary
```

```
df.describe(include='all')
```

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_code	protocol	response.code	d
count	2.820000e+02	2.820000e+02	282	282	282	282	282	282.0	
unique	NaN	NaN	30	30	28	7	1	NaN	
top	NaN	NaN	2024-04-26T09:00:00Z	2024-04-26T09:10:00Z	165.225.209.4	US	HTTPS	NaN	
freq	NaN	NaN	17	17	29	113	282	NaN	
mean	1.199390e+06	8.455429e+04	NaN	NaN	NaN	NaN	NaN	200.0	
std	4.149312e+06	2.549279e+05	NaN	NaN	NaN	NaN	NaN	0.0	
min	4.000000e+01	4.400000e+01	NaN	NaN	NaN	NaN	NaN	200.0	
25%	5.381500e+03	1.114200e+04	NaN	NaN	NaN	NaN	NaN	200.0	
50%	1.318200e+04	1.379950e+04	NaN	NaN	NaN	NaN	NaN	200.0	
75%	3.083300e+04	2.627950e+04	NaN	NaN	NaN	NaN	NaN	200.0	
max	2.520779e+07	1.561220e+06	NaN	NaN	NaN	NaN	NaN	200.0	

```
# 6. Check Missing Values

missing_values = df.isnull().sum()
print(missing_values)
```

```
bytes_in               0
bytes_out              0
creation_time          0
end_time               0
src_ip                 0
src_ip_country_code    0
protocol               0
response.code          0
dst_port               0
dst_ip                 0
rule_names             0
observation_name       0
source.meta            0
source.name            0
time                   0
detection_types        0
dtype: int64
```

```
# 7. Duplicate Records

print("Duplicate rows:", df.duplicated().sum())
```

```
Duplicate rows: 0
```

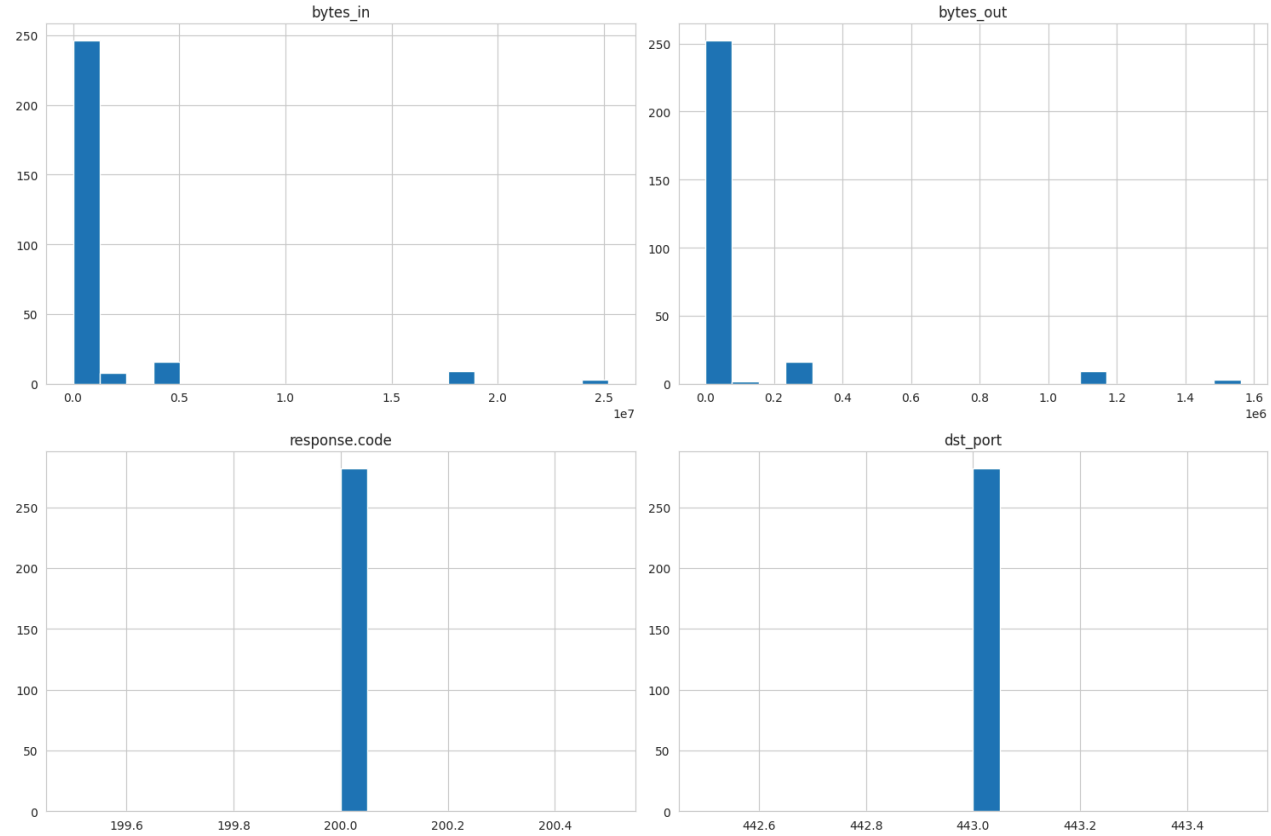
```
# 8. Data Types & Unique values

df.nunique().sort_values(ascending=False)
```

	0
bytes_in	260
bytes_out	239
creation_time	30
end_time	30
time	30
src_ip	28
src_ip_country_code	7
protocol	1
response.code	1
dst_port	1
rule_names	1
dst_ip	1
observation_name	1
source.meta	1
source.name	1
detection_types	1

dtype: int64

```
# ============================
# Exploratory Data Analysis (EDA)
# ============================
```

```
# 9. Univariate Analysis

num_cols = df.select_dtypes(include=np.number).columns

df[num_cols].hist(figsize=(15, 10), bins=20)
plt.tight_layout()
plt.show()
```

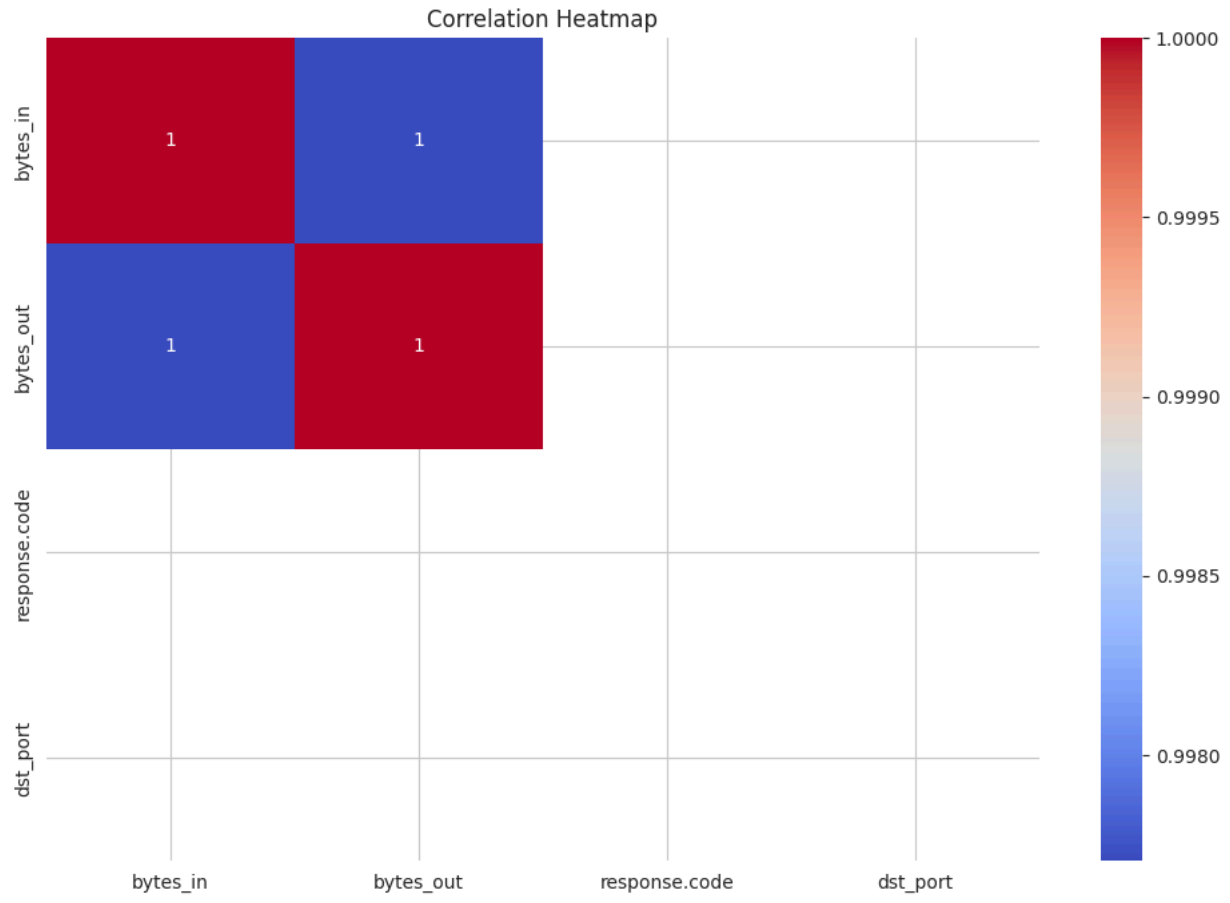



```
cat_cols = df.select_dtypes(include='object').columns

for col in cat_cols:
    print(f"\nValue counts for {col}:")
    print(df[col].value_counts().head())
```

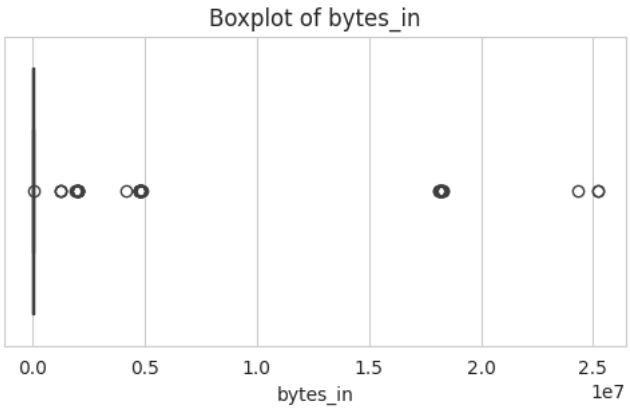
```
Value counts for creation_time:
creation_time
2024-04-26T09:00:00Z    17
2024-04-26T08:50:00Z    13
2024-04-26T09:40:00Z    11
2024-04-26T09:50:00Z    11
2024-04-25T23:10:00Z    11
Name: count, dtype: int64

Value counts for end_time:
end_time
2024-04-26T09:10:00Z    17
2024-04-26T09:00:00Z    13
2024-04-26T09:50:00Z    11
2024-04-26T10:00:00Z    11
2024-04-25T23:20:00Z    11
Name: count, dtype: int64

Value counts for src_ip:
src_ip
165.225.209.4     29
165.225.26.101    28
155.91.45.242     28
136.226.67.101    28
147.161.131.1     21
Name: count, dtype: int64

Value counts for src_ip_country_code:
src_ip_country_code
```

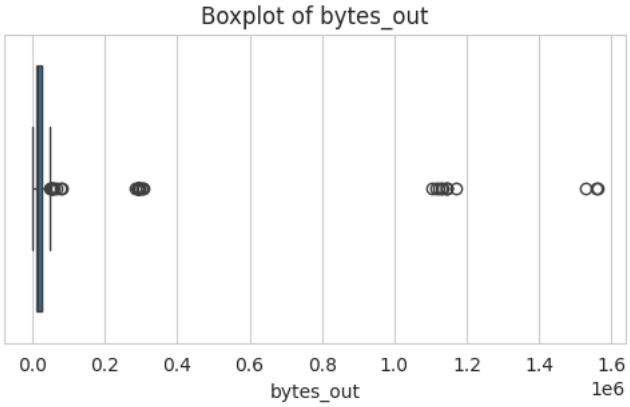
```
US    113
CA     72
DE     28
AT     21
NL     18
Name: count, dtype: int64

Value counts for protocol:
protocol
HTTPS    282
Name: count, dtype: int64

Value counts for dst_ip:
dst_ip
10.138.69.97    282
Name: count, dtype: int64

Value counts for rule_names:
rule_names
Suspicious Web Traffic    282
Name: count, dtype: int64

Value counts for observation_name:
observation_name
Adversary Infrastructure Interaction    282
Name: count, dtype: int64

Value counts for source_meta:
```

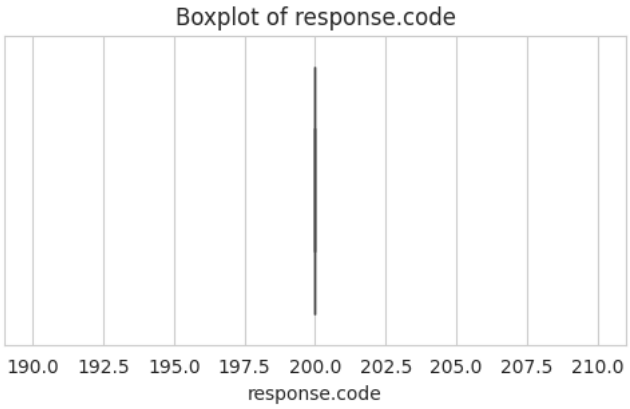
```
# 10. Bivariate Analysis

# ---- 10.1 Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df[num_cols].corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

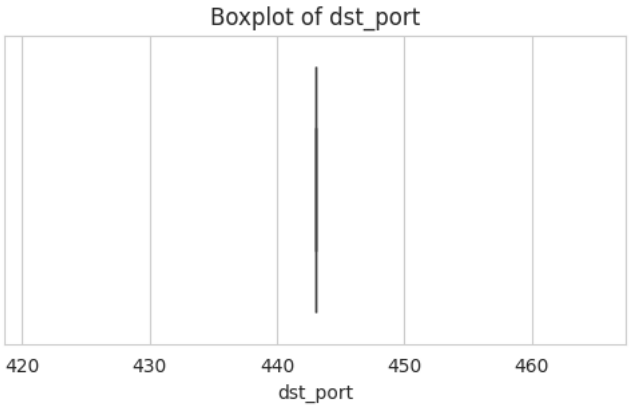



```
# ---- 10.2 Boxplot (Outlier Detection)
for col in num_cols:
    plt.figure(figsize=(6, 3))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

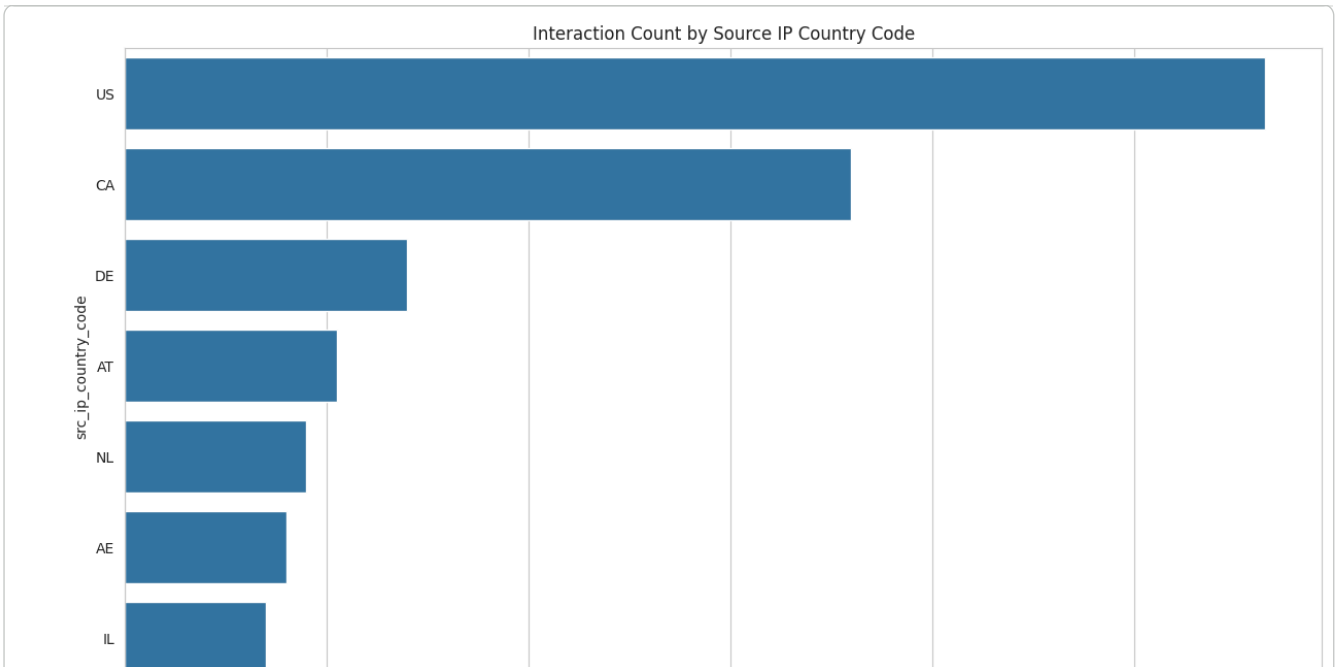









```
# 11. Country-based Interaction Analysis

plt.figure(figsize=(15, 8))
sns.countplot(y='src_ip_country_code', data=df,
order=df['src_ip_country_code'].value_counts().index)
plt.title('Interaction Count by Source IP Country Code')
plt.show()
```


Interaction Count by Source IP Country Code
src_ip_country_code
US
CA
DE
AT
NL
AE
IL