In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Step 2: Load Dataset
```

In [3]:
```
df = pd.read_csv("cybersecurity_data.csv")

df.info()
df.head()
```

```
<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
```

Out[3]:

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

In [4]:
```
# Remove Duplicate Riws
```

In [5]:
```
df_unique = df.drop_duplicates()
```

In [6]:
```
df_unique['bytes_in'].fillna(df_unique['bytes_in'].median(), inplace=True)
df_unique['bytes_out'].fillna(df_unique['bytes_out'].median(), inplace=True)
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_12892\2509335326.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained as
signment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df_unique['bytes_in'].fillna(df_unique['bytes_in'].median(), inplace=True)
C:\Users\krish\AppData\Local\Temp\ipykernel_12892\2509335326.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained as
signment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df_unique['bytes_out'].fillna(df_unique['bytes_out'].median(), inplace=True)
```

In [7]:
```
df_unique['bytes_in'] = df_unique['bytes_in'].fillna(df_unique['bytes_in'].media
df_unique['bytes_out'] = df_unique['bytes_out'].fillna(df_unique['bytes_out'].me
```

In [8]:
```
df_unique.dropna(subset=['src_ip', 'dst_ip'], inplace=True)
```

In [9]:
```
df_unique['creation_time'] = pd.to_datetime(df_unique['creation_time'])
df_unique['end_time'] = pd.to_datetime(df_unique['end_time'])
df_unique['time'] = pd.to_datetime(df_unique['time'])
```

In [10]:
```
df_unique['src_ip_country_code'] = df_unique['src_ip_country_code'].str.upper()
```

In [11]:
```
df_unique.info()
df_unique.head()
```

```
<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    datetime64[ns, UTC]
 3   end_time             282 non-null    datetime64[ns, UTC]
 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    datetime64[ns, UTC]
 15  detection_types      282 non-null    object
dtypes: datetime64[ns, UTC](3), int64(4), object(9)
memory usage: 35.4+ KB
```

Out[11]:

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_co
0	5602	12990	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	147.161.161.82	
1	30912	18186	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.33.6	
2	28506	13468	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.212.255	
3	30546	14278	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	136.226.64.114	
4	6526	13892	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.240.79	

In [12]:
```
visualize the distribution of bytes in and bytes out
```

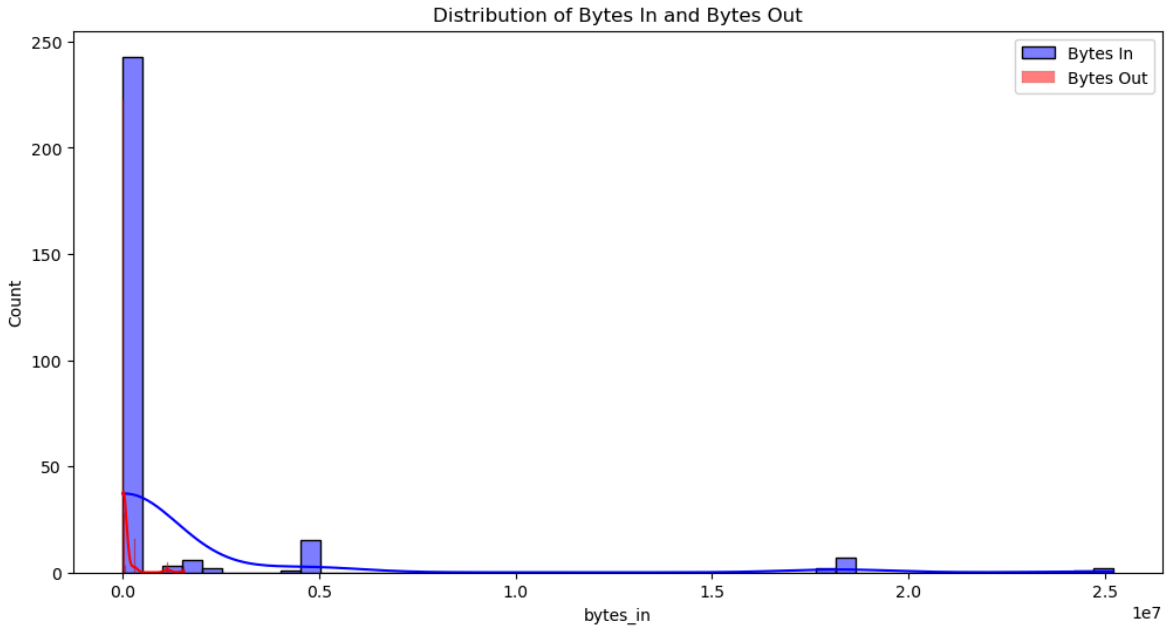
```
  Cell In[12], line 1
    visualize the distribution of bytes in and bytes out
              ^
SyntaxError: invalid syntax
```

In [13]:
```
# visualize the distribution of bytes in and bytes out
```

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

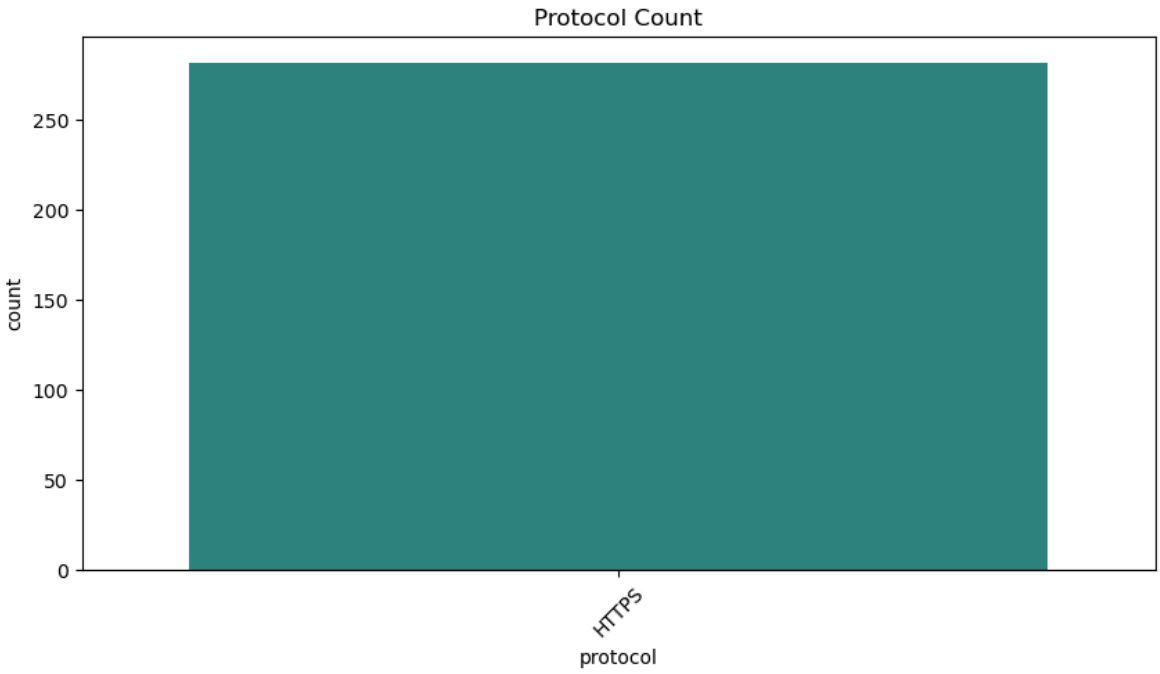
In [15]:
```
# Distribution plot for Bytes In and Bytes Out
```

In [16]:
```
plt.figure(figsize=(12, 6))
sns.histplot(df_unique['bytes_in'], bins=50, color='blue', kde=True, label='Byte
sns.histplot(df_unique['bytes_out'], bins=50, color='red', kde=True, label='Byte
plt.legend()
plt.title('Distribution of Bytes In and Bytes Out')
plt.show()
```

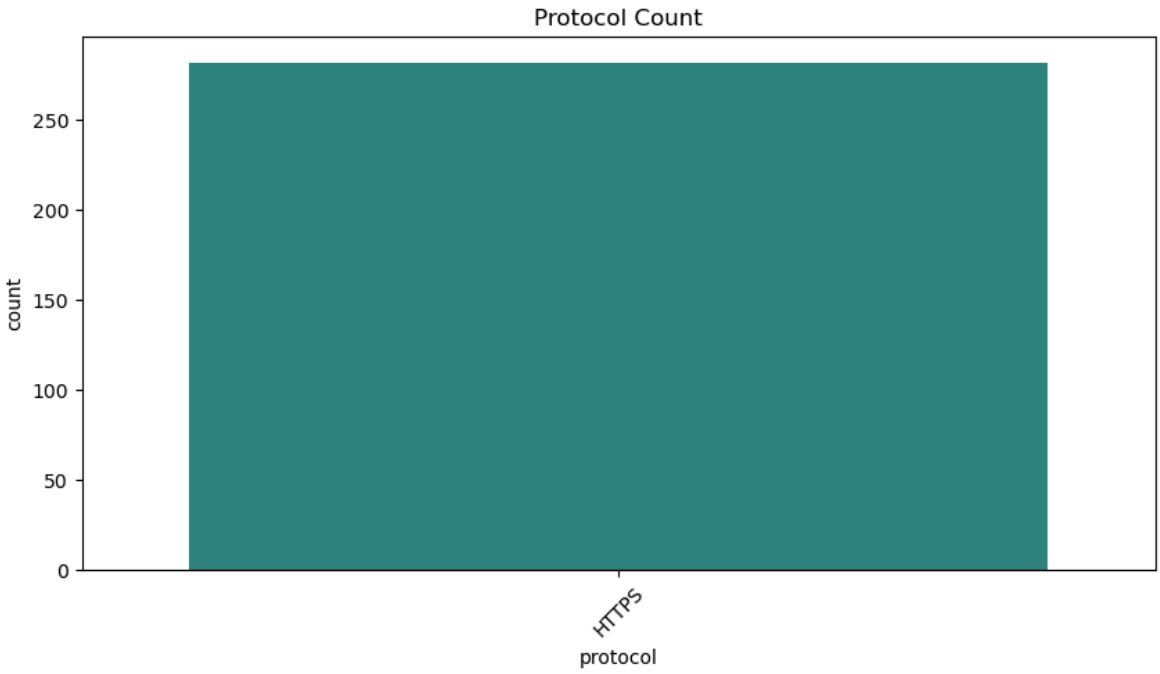


In [17]:
```
plt.figure(figsize=(10, 5))
sns.countplot(x='protocol', data=df_unique, palette='viridis')
plt.title('Protocol Count')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_12892\1922916102.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(x='protocol', data=df_unique, palette='viridis')
```

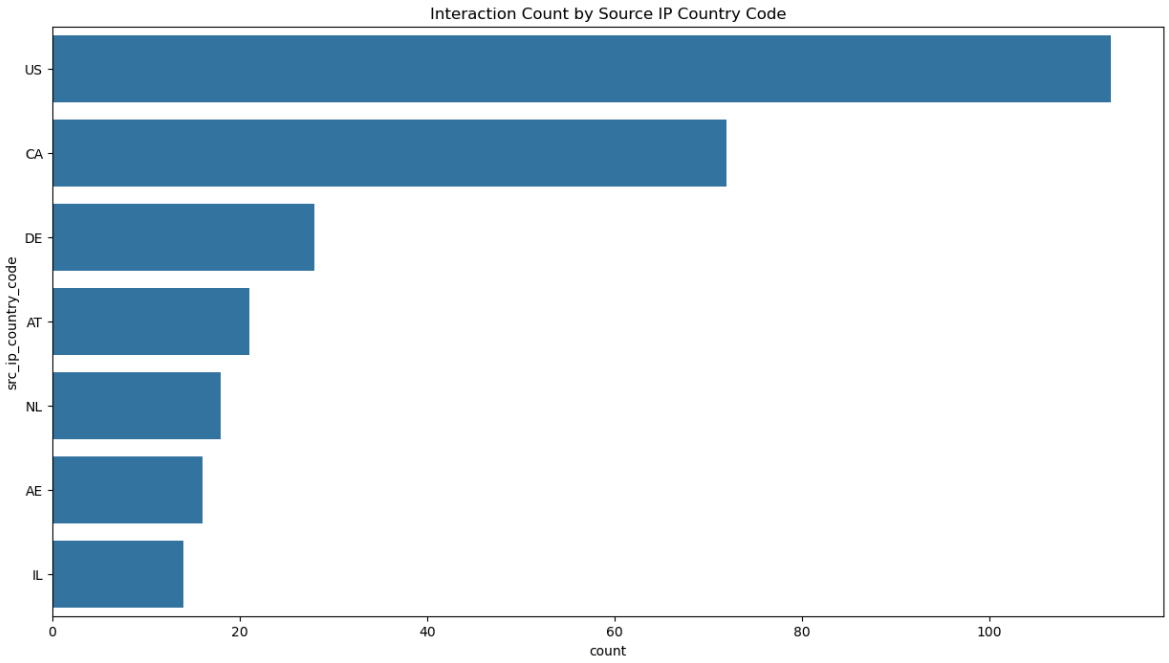


In [18]:
```
plt.figure(figsize=(10, 5))
sns.countplot(x='protocol', hue='protocol', data=df_unique, palette='viridis', l
plt.title('Protocol Count')
plt.xticks(rotation=45)
plt.show()
```



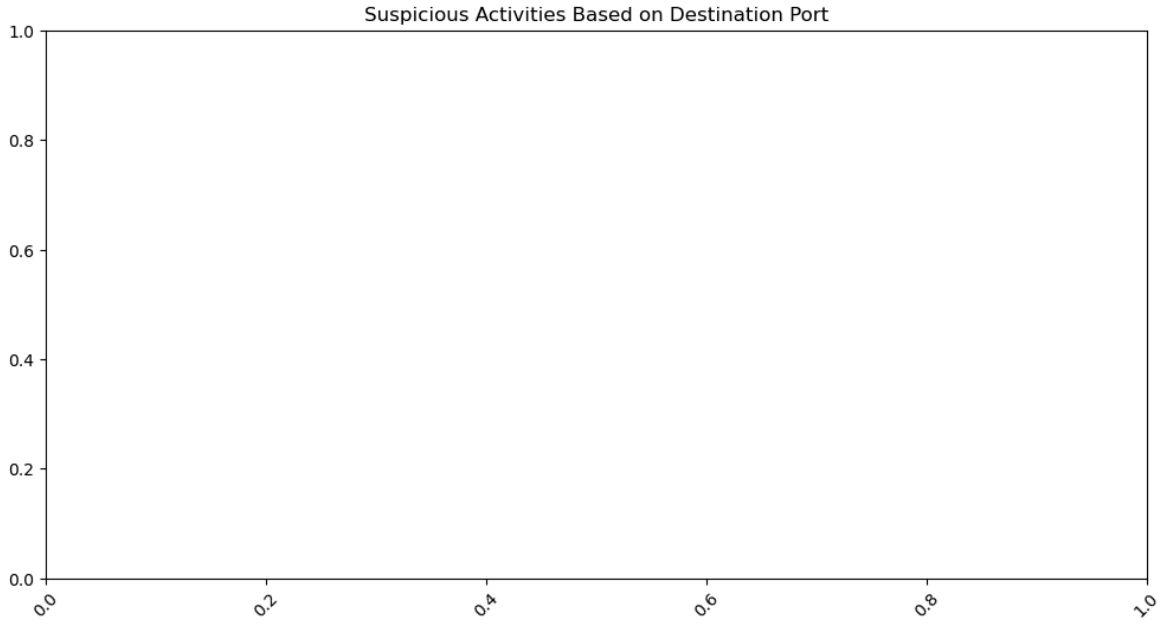
In [19]:
```
# Country-Based Interaction Count
```

In [20]:
```
plt.figure(figsize=(15, 8))
sns.countplot(y='src_ip_country_code', data=df_unique,
              order=df_unique['src_ip_country_code'].value_counts().index)
plt.title('Interaction Count by Source IP Country Code')
plt.show()
```



In [21]:
```
# Suspicious Activities by Destination Port
```

In [22]:
```
plt.figure(figsize=(12, 6))
sns.countplot(x='dst_port',
              data=df_unique[df_unique['detection_types'] == 'Suspicious'],
              palette='coolwarm')
plt.title('Suspicious Activities Based on Destination Port')
plt.xticks(rotation=45)
plt.show()
```

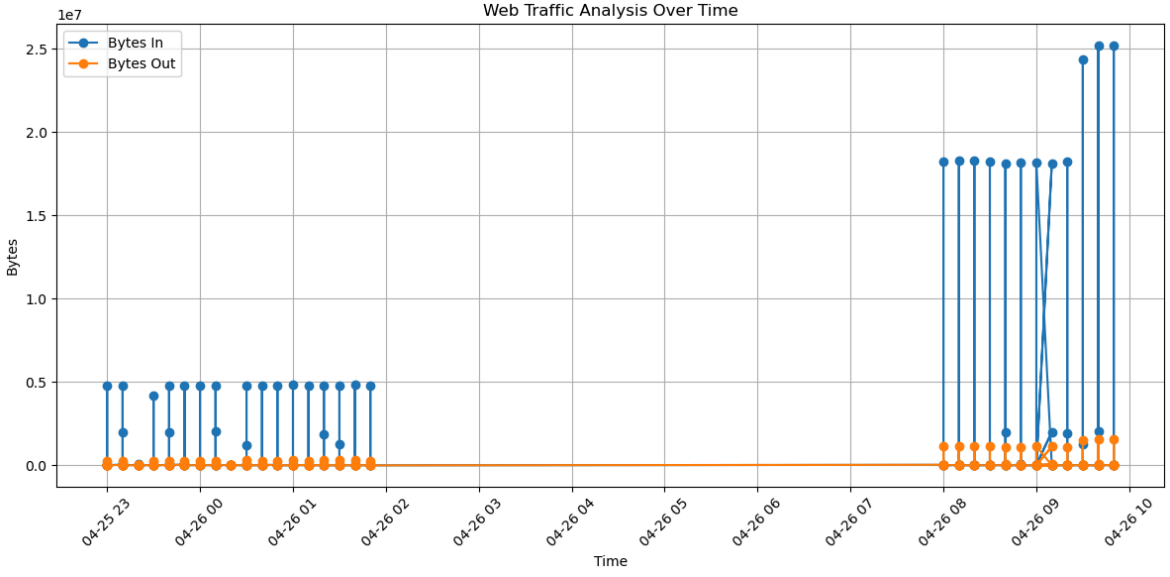


In [23]:
```
df_unique.set_index('cr"eation_time', inplace=False)

plt.figure(figsize=(12, 6))
plt.plot(df_unique['creation_time'], df_unique['bytes_in'], label='Bytes In', ma
plt.plot(df_unique['creation_time'], df_unique['bytes_out'], label='Bytes Out',
plt.title('Web Traffic Analysis Over Time')
plt.xlabel('Time')
plt.ylabel('Bytes')
plt.legend()
```

```
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

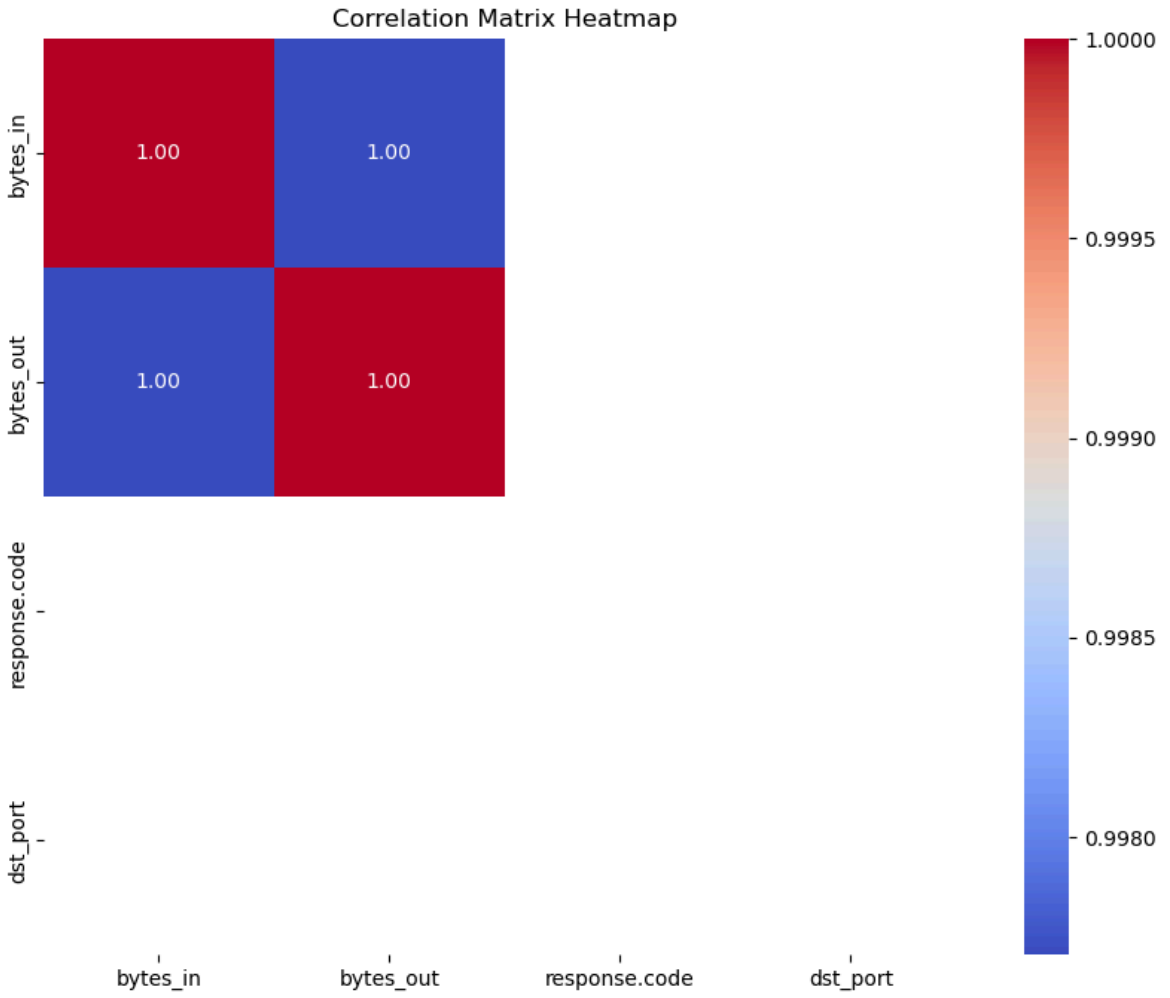
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_12892\3291076860.py in ?()
----> 1 df_unique.set_index('cr"eation_time', inplace=False)
      2
      3 plt.figure(figsize=(12, 6))
      4 plt.plot(df_unique['creation_time'], df_unique['bytes_in'], label='Bytes
In', marker='o')

C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py in ?(self, keys,
drop, append, inplace, verify_integrity)
   6118                     if not found:
   6119                         missing.append(col)
   6120
   6121         if missing:
-> 6122             raise KeyError(f"None of {missing} are in the columns")
   6123
   6124         if inplace:
   6125             frame = self

KeyError: 'None of [\'cr"eation_time\'] are in the columns'
```

In [24]:
```
df_unique['creation_time'] = pd.to_datetime(df_unique['creation_time'])

plt.figure(figsize=(12, 6))
plt.plot(df_unique['creation_time'], df_unique['bytes_in'], label='Bytes In', ma
plt.plot(df_unique['creation_time'], df_unique['bytes_out'], label='Bytes Out',
plt.title('Web Traffic Analysis Over Time')
plt.xlabel('Time')
plt.ylabel('Bytes')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [25]:
```
# Compute correlation matrix for numeric columns
```

In [26]:
```
numeric_df = df_unique.select_dtypes(include=['float64', 'int64'])
correlation_matrix_numeric = numeric_df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix_numeric, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [27]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

In [28]:
```
df_unique['session_duration'] = (df_unique['end_time'] - df_unique['creation_tim
```

In [29]:
```
df_unique['avg_packet_size'] = (df_unique['bytes_in'] + df_unique['bytes_out'])
```

In [30]:
```
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_unique[['bytes_in', 'bytes_out', 'sess
scaled_columns = ['scaled_bytes_in', 'scaled_bytes_out', 'scaled_session_duratio
```

In [31]:
```
scaled_df = pd.DataFrame(scaled_features, columns=scaled_columns, index=df_uniqu

encoder = OneHotEncoder(sparse=False)
encoded_features = encoder.fit_transform(df_unique[['src_ip_country_code']])
encoded_columns = encoder.get_feature_names_out(['src_ip_country_code'])
encoded_df = pd.DataFrame(encoded_features, columns=encoded_columns, index=df_un
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 3
      1 scaled_df = pd.DataFrame(scaled_features, columns=scaled_columns, index=d
f_unique.index)
----> 3 encoder = OneHotEncoder(sparse=False)
      4 encoded_features = encoder.fit_transform(df_unique[['src_ip_country_cod
e']])
      5 encoded_columns = encoder.get_feature_names_out(['src_ip_country_code'])

TypeError: OneHotEncoder.__init__() got an unexpected keyword argument 'sparse'
```

In [32]:
```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse_output=False)
encoded_features = encoder.fit_transform(df_unique[['src_ip_country_code']])
encoded_columns = encoder.get_feature_names_out(['src_ip_country_code'])
encoded_df = pd.DataFrame(encoded_features, columns=encoded_columns, index=df_un
```

In [33]:
```
df_transformed = pd.concat([df_unique, scaled_df, encoded_df], axis=1)
```

In [34]:
```
df_transformed.head()
```

Out[34]:

	bytes_in	bytes_out	creation_time	end_time	src_ip	src_ip_country_co
0	5602	12990	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	147.161.161.82	
1	30912	18186	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.33.6	
2	28506	13468	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.212.255	
3	30546	14278	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	136.226.64.114	
4	6526	13892	2024-04-25 23:00:00+00:00	2024-04-25 23:10:00+00:00	165.225.240.79	

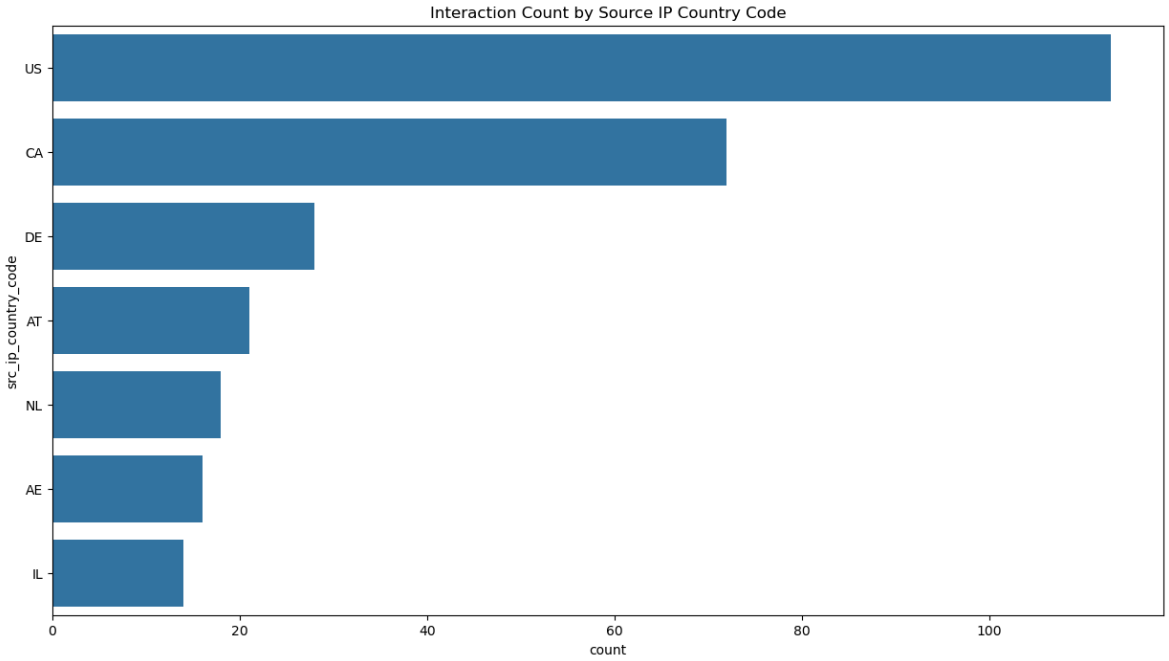
5 rows × 29 columns

In [35]:
```
# visualization
```

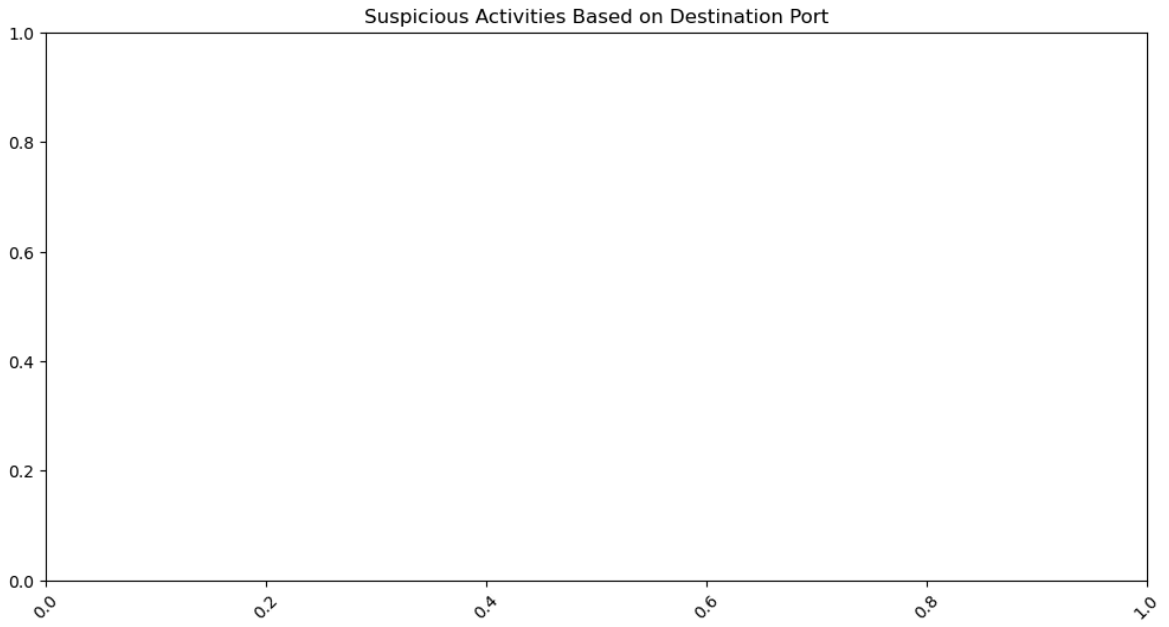
In [36]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [37]:
```
plt.figure(figsize=(15, 8))
sns.countplot(
    y='src_ip_country_code',
```

```
    data=df_transformed,
    order=df_transformed['src_ip_country_code'].value_counts().index
)
plt.title('Interaction Count by Source IP Country Code')
plt.show()
```



In [38]:
```
plt.figure(figsize=(12, 6))
sns.countplot(
    x='dst_port',
    data=df_transformed[df_transformed['detection_types'] == 'Suspicious'],
    palette='coolwarm'
)
plt.title('Suspicious Activities Based on Destination Port')
plt.xticks(rotation=45)
plt.show()
```



In [39]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='bytes_in',
    y='bytes_out',
```

```
    hue='anomaly',
    data=df_transformed,
    palette={'Normal': 'green', 'Suspicious': 'red'}
)
plt.title('Anomalies in Bytes In vs Bytes Out')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[39], line 2
      1 plt.figure(figsize=(10, 6))
----> 2 sns.scatterplot(
      3     x='bytes_in',
      4     y='bytes_out',
      5     hue='anomaly',
      6     data=df_transformed,
      7     palette={'Normal': 'green', 'Suspicious': 'red'}
      8 )
      9 plt.title('Anomalies in Bytes In vs Bytes Out')
     10 plt.show()

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\relational.py:615, in scatterplot(data, x, y, hue, size, style, palette, hue_order, hue_norm, sizes, size_order, size_norm, markers, style_order, legend, ax, **kwargs)
    606 def scatterplot(
    607     data=None, *,
    608     x=None, y=None, hue=None, size=None, style=None,
   (...)
    612     **kwargs
    613 ):
--> 615     p = _ScatterPlotter(
    616         data=data,
    617         variables=dict(x=x, y=y, hue=hue, size=size, style=style),
    618         legend=legend
    619     )
    621     p.map_hue(palette=palette, order=hue_order, norm=hue_norm)
    622     p.map_size(sizes=sizes, order=size_order, norm=size_norm)

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\relational.py:396, in _ScatterPlotter.__init__(self, data, variables, legend)
    387 def __init__(self, *, data=None, variables={}, legend=None):
    388
    389     # TODO this is messy, we want the mapping to be agnostic about
    390     # the kind of plot to draw, but for the time being we need to set
    391     # this information so the SizeMapping can use it
    392     self._default_size_range = (
    393         np.r_[.5, 2] * np.square(mpl.rcParams["lines.markersize"])
    394     )
--> 396     super().__init__(data=data, variables=variables)
    398     self.legend = legend

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_base.py:679, in VectorPlotter.assign_variables(self, data, variables)
    674 else:
```

```
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
    680     frame = plot_data.frame
    681     names = plot_data.names

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_core\data.py:58, in PlotData.__init__(self, data, variables)
     51 def __init__(
     52     self,
     53     data: DataSource,
     54     variables: dict[str, VariableSpec],
     55 ):
     57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
     60     self.frame = frame
     61     self.names = names

File C:\ProgramData\anaconda3\Lib\site-packages\seaborn\_core\data.py:232, in PlotData._assign_variables(self, data, variables)
    230     else:
    231         err += "An entry with this name does not appear in `data`."
--> 232     raise ValueError(err)
    234 else:
    235
    236     # Otherwise, assume the value somehow represents data
    237
    238     # Ignore empty data structures
    239     if isinstance(val, Sized) and len(val) == 0:

ValueError: Could not interpret value `anomaly` for `hue`. An entry with this name does not appear in `data`.
```

```
<Figure size 1000x600 with 0 Axes>
```

In [41]:

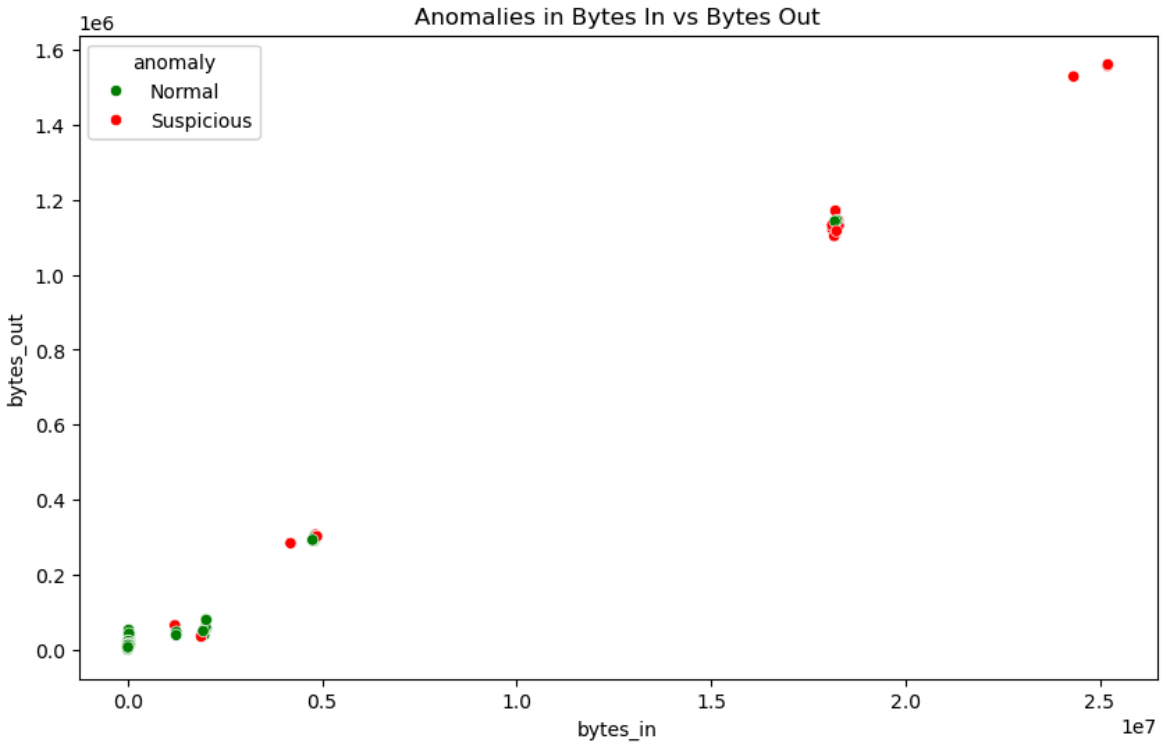
```
from sklearn.ensemble import IsolationForest

features = df_transformed[['bytes_in', 'bytes_out', 'session_duration', 'avg_pac

model = IsolationForest(contamination=0.05, random_state=42)

df_transformed['anomaly'] = model.fit_predict(features)

df_transformed['anomaly'] = df_transformed['anomaly'].apply(lambda x: 'Suspiciou

print(df_transformed['anomaly'].value_counts())
```

```
anomaly
Normal        267
Suspicious     15
Name: count, dtype: int64
```

In [42]:

```
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='bytes_in',
    y='bytes_out',
    hue='anomaly',
    data=df_transformed,
    palette={'Normal': 'green', 'Suspicious': 'red'}
```

```
)
plt.title('Anomalies in Bytes In vs Bytes Out')
plt.show()
```



In [43]:
```
# Green dots → Normal traffic

# Red dots → Suspicious connections detected by Isolation Forest
```

In [ ]:
```

```