In [6]:

```
import pandas as pd
df = pd.read_csv(r"D:\log2.csv")
print("Original DataFrame:")
print(df)
```

```
Original DataFrame:
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
0            57222                53            54587                    53   
1            56258              3389            56258                  3389   
2             6881             50321            43265                 50321   
3            50553              3389            50553                  3389   
4            50002               443            45848                   443   
...            ...               ...              ...                   ...   
65527        63691                80            13237                    80   
65528        50964                80            13485                    80   
65529        54871               445                0                     0   
65530        54870               445                0                     0   
65531        54867               445                0                     0   

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
0      allow      177          94              83        2   
1      allow     4768        1600            3168       19   
2      allow      238         118             120        2   
3      allow     3327        1438            1889       15   
4      allow    25358        6778           18580       31   
...      ...      ...         ...             ...      ...   
65527  allow      314         192             122        6   
65528  allow  4680740       67312         4613428     4675   
65529   drop       70          70               0        1   
65530   drop       70          70               0        1   
65531   drop       70          70               0        1   

       Elapsed Time (sec)  pkts_sent  pkts_received  
0                      30          1              1  
1                      17         10              9  
2                    1199          1              1  
3                      17          8              7  
4                      16         13             18  
...                   ...        ...            ...  
65527                  15          4              2  
65528                  77        985           3690  
65529                   0          1              0  
65530                   0          1              0  
65531                   0          1              0  

[65532 rows x 12 columns]
```

In [7]:

```
q1 = df['Destination Port'].quantile(0.25)
q3 = df['Destination Port'].quantile(0.75)
iqr = q3 - q1
lb = q1 - 1.5 * iqr
ub = q3 + 1.5 * iqr
outliers = df[(df['Destination Port'] < lb) | (df['Destination Port'] > ub)]
print("\nOutliers in 'Destination Port':")
print(outliers)

print("Original DataFrame:")
print(df)

q1 = df['Bytes Sent'].quantile(0.25)
q3 = df['Bytes Sent'].quantile(0.75)
iqr = q3 - q1
```

```python
lb = q1 - 1.5 * iqr
ub = q3 + 1.5 * iqr

outliers = df[(df['Bytes Sent'] < lb) | (df['Bytes Sent'] > ub)]

print("\nOutliers in 'Bytes Sent':")
print(outliers)
```

```
Outliers in 'Destination Port':
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
2             6881             50321            43265                 50321
6            60513             47094            45469                 47094
8            52244             58774             2211                 58774
50           63842             45682            31353                 45682
62           60811             40010            33835                 40010
...            ...               ...              ...                   ...
65495        51688             47961            35098                 47961
65508        36226             60038                0                     0
65518        54013             37965                0                     0
65522        53314             64097                0                     0
65526        51710             43069            65147                 43069

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
2      allow      238         118             120        2
6      allow      320         140             180        6
8      allow       70          70               0        1
50     allow  4687209     3850148          837061     4974
62     allow      316         136             180        6
...      ...      ...         ...             ...      ...
65495  allow       66          66               0        1
65508  allow       66          66               0        1
65518   deny       66          66               0        1
65522   deny       66          66               0        1
65526  allow       70          70               0        2

       Elapsed Time (sec)  pkts_sent  pkts_received
2                    1199          1              1
6                       7          3              3
8                       5          1              0
50                    107       3004           1970
62                      5          3              3
...                   ...        ...            ...
65495                   5          1              0
65508                  91          1              0
65518                   0          1              0
65522                   0          1              0
65526                   8          2              0

[9043 rows x 12 columns]
Original DataFrame:
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
0            57222                53            54587                    53
1            56258              3389            56258                  3389
2             6881             50321            43265                 50321
3            50553              3389            50553                  3389
4            50002               443            45848                   443
...            ...               ...              ...                   ...
65527        63691                80            13237                    80
65528        50964                80            13485                    80
65529        54871               445                0                     0
65530        54870               445                0                     0
65531        54867               445                0                     0

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
0      allow      177          94              83        2
1      allow     4768        1600            3168       19
2      allow      238         118             120        2
3      allow     3327        1438            1889       15
4      allow    25358        6778           18580       31
...      ...      ...         ...             ...      ...
65527  allow      314         192             122        6
65528  allow  4680740       67312         4613428     4675
```

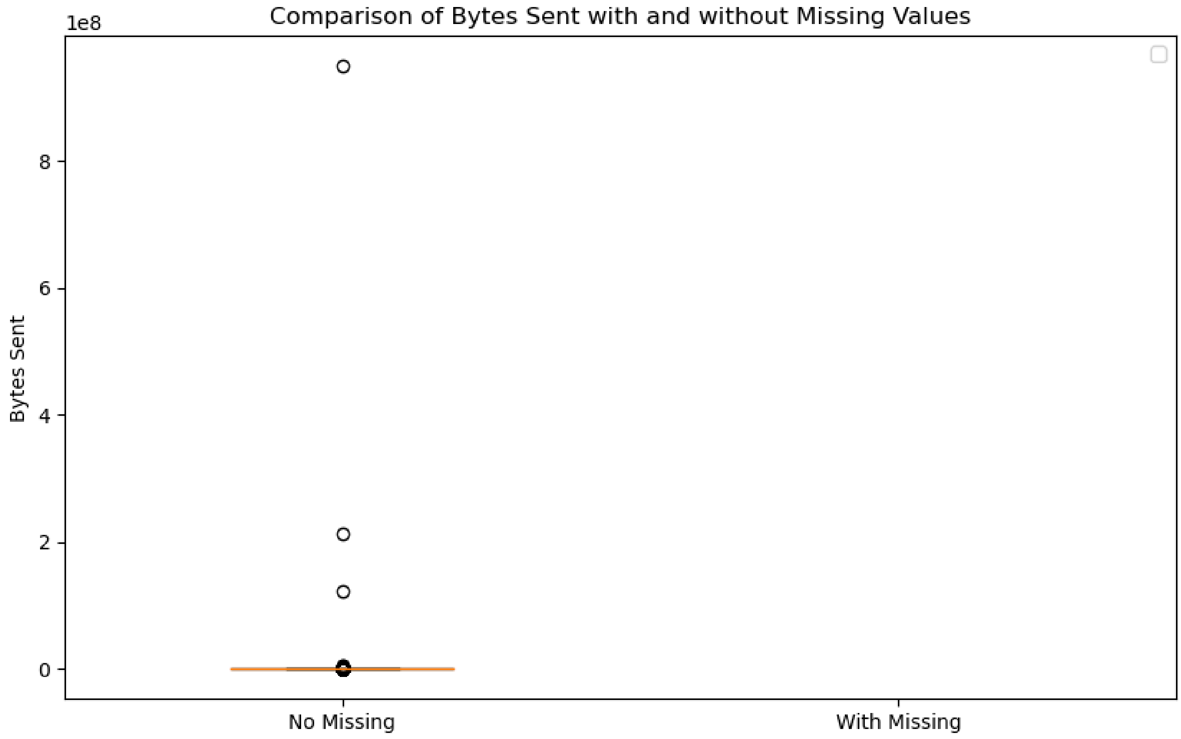
```
65529   drop      70        70             0        1
65530   drop      70        70             0        1
65531   drop      70        70             0        1

       Elapsed Time (sec)  pkts_sent  pkts_received
0                      30          1              1
1                      17         10              9
2                    1199          1              1
3                      17          8              7
4                      16         13             18
...                   ...        ...            ...
65527                  15          4              2
65528                  77        985           3690
65529                   0          1              0
65530                   0          1              0
65531                   0          1              0

[65532 rows x 12 columns]

Outliers in 'Bytes Sent':
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
1            56258              3389            56258                  3389
3            50553              3389            50553                  3389
4            50002               443            45848                   443
5            51465               443            39975                   443
7            50049               443            21285                   443
...            ...               ...              ...                   ...
65499        50343                80            49722                    80
65501        50438              3389            50438                  3389
65505        35608               443            62915                   443
65511        58574               443             3429                   443
65528        50964                80            13485                    80

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
1      allow     4768        1600            3168       19
3      allow     3327        1438            1889       15
4      allow    25358        6778           18580       31
5      allow     3961        1595            2366       21
7      allow     7912        3269            4643       23
...      ...      ...         ...             ...      ...
65499  allow    22233       10123           12110       37
65501  allow     3429        1474            1955       16
65505  allow     5776        1880            3896       19
65511  allow     3447         788            2659       13
65528  allow  4680740       67312         4613428     4675

       Elapsed Time (sec)  pkts_sent  pkts_received
1                      17         10              9
3                      17          8              7
4                      16         13             18
5                      16         12              9
7                      96         12             11
...                   ...        ...            ...
65499                  28         16             21
65501                  16          8              8
65505                 272         11              8
65511                 135          6              7
65528                  77        985           3690

[14701 rows x 12 columns]
```

In [8]:

```python
import matplotlib.pyplot as plt
d_no = df[df['Bytes Sent'].notnull()]
d_with = df[df['Bytes Sent'].isnull()]
```

```
print("\nNumber of rows with missing 'Bytes Sent':", d_with.shape[0])
print("Number of rows without missing 'Bytes Sent':", d_no.shape[0])
plt.figure(figsize=(10, 6))
plt.boxplot(d_no['Bytes Sent'], positions=[1], widths=0.4, patch_artist=True,
boxprops=dict(facecolor='lightblue'),)
plt.boxplot(d_with['Bytes Sent'], positions=[2], widths=0.4, patch_artist=True,
boxprops=dict(facecolor='lightcoral'),)
plt.title('Comparison of Bytes Sent with and without Missing Values')
plt.xticks([1, 2], ['No Missing', 'With Missing'])
plt.ylabel('Bytes Sent')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
Number of rows with missing 'Bytes Sent': 0
Number of rows without missing 'Bytes Sent': 65532
```

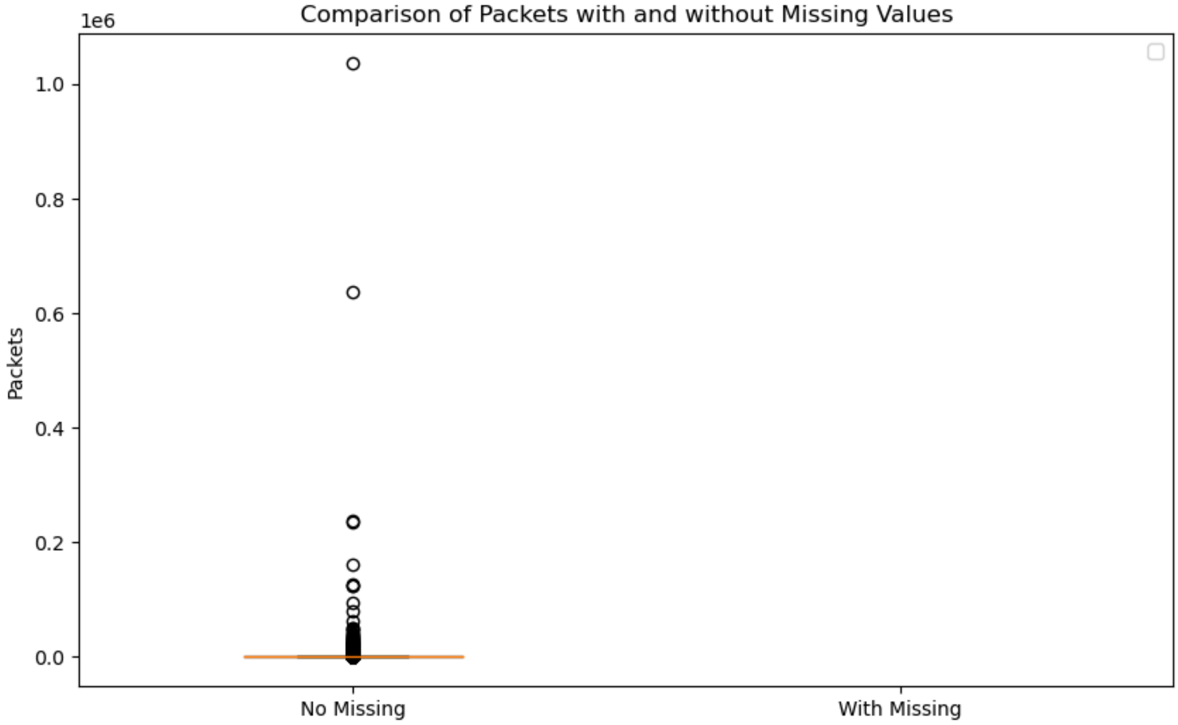


In [9]:

```
d_no = df[df['Packets'].notnull()]
d_with = df[df['Packets'].isnull()]

print("\nNumber of rows with missing 'Packets':", d_with.shape[0])
print("Number of rows without missing 'Packets':", d_no.shape[0])

plt.figure(figsize=(10, 6))

plt.boxplot(d_no['Packets'], positions=[1], widths=0.4, patch_artist=True,
boxprops=dict(facecolor='lightblue'))

plt.boxplot(d_with['Packets'], positions=[2], widths=0.4, patch_artist=True,
boxprops=dict(facecolor='lightcoral'))

plt.title('Comparison of Packets with and without Missing Values')
plt.xticks([1, 2], ['No Missing', 'With Missing'])
plt.ylabel('Packets')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
```

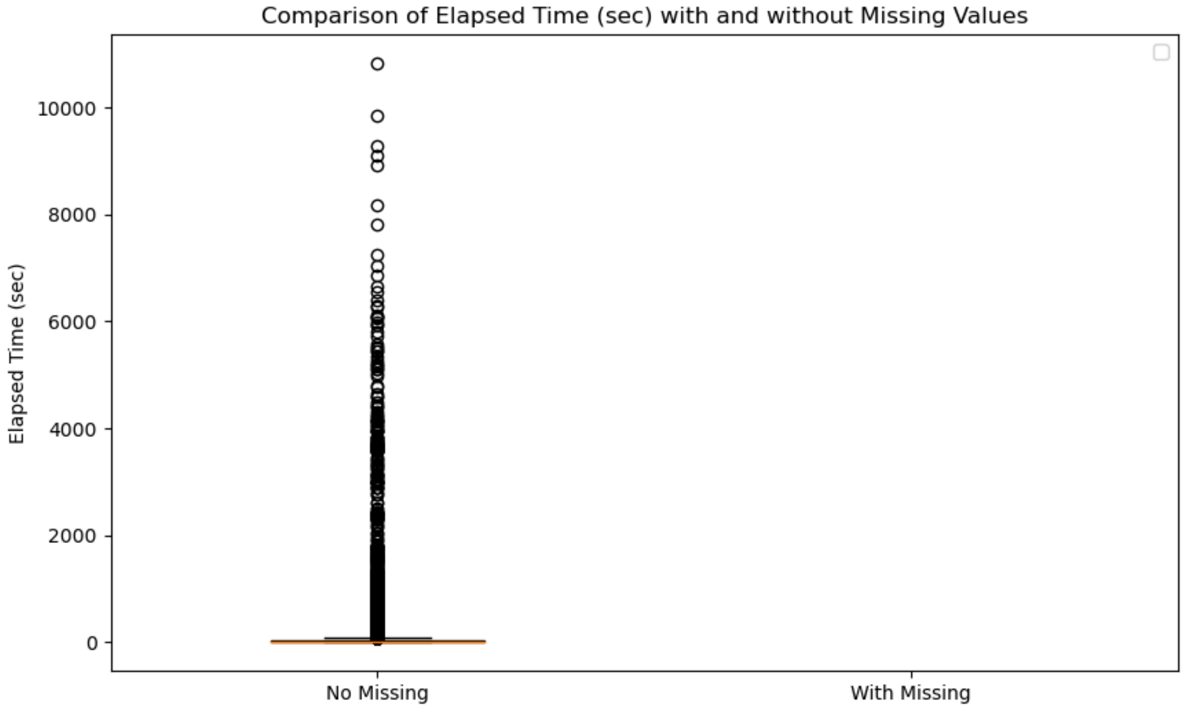
```
Number of rows with missing 'Packets': 0
Number of rows without missing 'Packets': 65532
```



In [10]:

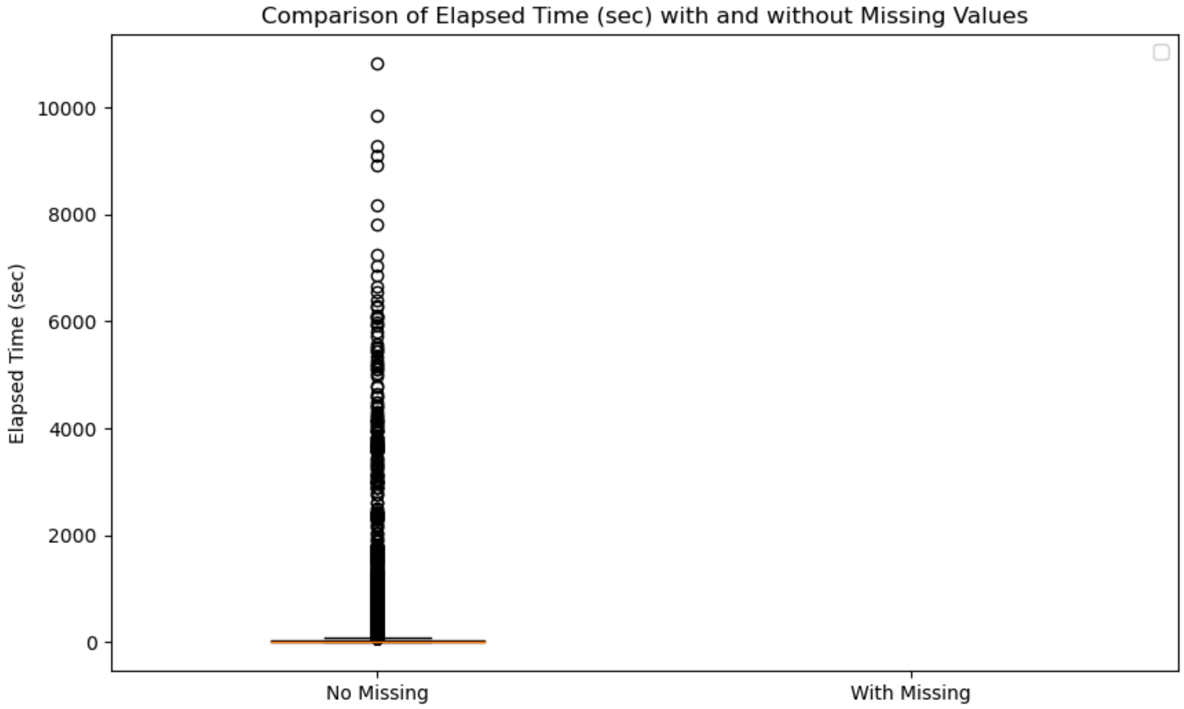
```
d_no = df[df['Elapsed Time (sec)'].notnull()]
d_with = df[df['Elapsed Time (sec)'].isnull()]

print("\nNumber of rows with missing 'Elapsed Time (sec)':", d_with.shape[0])
print("Number of rows without missing 'Elapsed Time (sec)':", d_no.shape[0])

plt.figure(figsize=(10, 6))

plt.boxplot(d_no['Elapsed Time (sec)'], positions=[1], widths=0.4, patch_artist=Tru
boxprops=dict(facecolor='lightblue'), )

plt.boxplot(d_with['Elapsed Time (sec)'], positions=[2], widths=0.4, patch_artist=T
boxprops=dict(facecolor='lightcoral'),)

plt.title('Comparison of Elapsed Time (sec) with and without Missing Values')
plt.xticks([1, 2], ['No Missing', 'With Missing'])
plt.ylabel('Elapsed Time (sec)')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
Number of rows with missing 'Elapsed Time (sec)': 0
Number of rows without missing 'Elapsed Time (sec)': 65532
```



```
In [11]: d_no = df[df['Elapsed Time (sec)'].notnull()]
d_with = df[df['Elapsed Time (sec)'].isnull()]
print("\nNumber of rows with missing 'Elapsed Time (sec)':", d_with.shape[0])
print("Number of rows without missing 'Elapsed Time (sec)':", d_no.shape[0])

plt.figure(figsize=(10, 6))

plt.boxplot(d_no['Elapsed Time (sec)'], positions=[1], widths=0.4, patch_artist=Tru
boxprops=dict(facecolor='lightblue'),  )

plt.boxplot(d_with['Elapsed Time (sec)'], positions=[2], widths=0.4, patch_artist=T
boxprops=dict(facecolor='lightcoral'),  )

plt.title('Comparison of Elapsed Time (sec) with and without Missing Values')
plt.xticks([1, 2], ['No Missing', 'With Missing'])
plt.ylabel('Elapsed Time (sec)')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label star
t with an underscore are ignored when legend() is called with no argument.
Number of rows with missing 'Elapsed Time (sec)': 0
Number of rows without missing 'Elapsed Time (sec)': 65532
```



In [12]:
```
import numpy as np

c = 'Bytes Sent'
if df[c].isnull().sum() == 0:
    print(f"\nNo missing values in '{c}'. Creating null values...")
for i in range(5):
    df.at[i, c] = np.nan
```

```
No missing values in 'Bytes Sent'. Creating null values...
```

In [13]:
```
d = pd.read_csv(r"D:\log2.csv")
```

In [14]:
```
print("\nDataFrame after creating null values:")
print(d[c].isnull().sum())
print(d)
```

```
DataFrame after creating null values:
0
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
0            57222                53            54587                    53   
1            56258              3389            56258                  3389   
2             6881             50321            43265                 50321   
3            50553              3389            50553                  3389   
4            50002               443            45848                   443   
...            ...               ...              ...                   ...   
65527        63691                80            13237                    80   
65528        50964                80            13485                    80   
65529        54871               445                0                     0   
65530        54870               445                0                     0   
65531        54867               445                0                     0   

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
0      allow      177          94              83        2   
1      allow     4768        1600            3168       19   
2      allow      238         118             120        2   
3      allow     3327        1438            1889       15   
4      allow    25358        6778           18580       31   
...      ...      ...         ...             ...      ...   
65527  allow      314         192             122        6   
65528  allow  4680740       67312         4613428     4675   
65529   drop       70          70               0        1   
65530   drop       70          70               0        1   
65531   drop       70          70               0        1   

       Elapsed Time (sec)  pkts_sent  pkts_received  
0                      30          1              1  
1                      17         10              9  
2                    1199          1              1  
3                      17          8              7  
4                      16         13             18  
...                   ...        ...            ...  
65527                  15          4              2  
65528                  77        985           3690  
65529                   0          1              0  
65530                   0          1              0  
65531                   0          1              0  

[65532 rows x 12 columns]
```

In [15]:

```python
c = 'Bytes Received'
if d[c].isnull().sum() == 0:
        print(f"\nNo missing values in '{c}'. Creating null values...")
```

```
No missing values in 'Bytes Received'. Creating null values...
```

In [16]:

```python
for i in range(5):
    d.at[i, c] = np.nan
print("\nDataFrame after creating null values:")
print(d[c].isnull().sum())
print(d)
```

```
DataFrame after creating null values:
5
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
0            57222                53            54587                    53   
1            56258              3389            56258                  3389   
2             6881             50321            43265                 50321   
3            50553              3389            50553                  3389   
4            50002               443            45848                   443   
...            ...               ...              ...                   ...   
65527        63691                80            13237                    80   
65528        50964                80            13485                    80   
65529        54871               445                0                     0   
65530        54870               445                0                     0   
65531        54867               445                0                     0   

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
0      allow      177          94             NaN        2   
1      allow     4768        1600             NaN       19   
2      allow      238         118             NaN        2   
3      allow     3327        1438             NaN       15   
4      allow    25358        6778             NaN       31   
...      ...      ...         ...             ...      ...   
65527  allow      314         192           122.0        6   
65528  allow  4680740       67312       4613428.0     4675   
65529   drop       70          70             0.0        1   
65530   drop       70          70             0.0        1   
65531   drop       70          70             0.0        1   

       Elapsed Time (sec)  pkts_sent  pkts_received  
0                      30          1              1  
1                      17         10              9  
2                    1199          1              1  
3                      17          8              7  
4                      16         13             18  
...                   ...        ...            ...  
65527                  15          4              2  
65528                  77        985           3690  
65529                   0          1              0  
65530                   0          1              0  
65531                   0          1              0  

[65532 rows x 12 columns]
```

In [17]:

```python
c = 'Packets'
if d[c].isnull().sum() == 0:
        print(f"\nNo missing values in '{c}'. Creating null values...")
for i in range(5):
    d.at[i, c] = np.nan
print("\nDataFrame after creating null values:")
```

```
No missing values in 'Packets'. Creating null values...

DataFrame after creating null values:
```

In [18]:

```python
print(d[c].isnull().sum())
print(d)
```

```
5
       Source Port  Destination Port  NAT Source Port  NAT Destination Port  \
0            57222                53            54587                    53   
1            56258              3389            56258                  3389   
2             6881             50321            43265                 50321   
3            50553              3389            50553                  3389   
4            50002               443            45848                   443   
...            ...               ...              ...                   ...   
65527        63691                80            13237                    80   
65528        50964                80            13485                    80   
65529        54871               445                0                     0   
65530        54870               445                0                     0   
65531        54867               445                0                     0   

      Action    Bytes  Bytes Sent  Bytes Received  Packets  \
0      allow      177          94             NaN      NaN   
1      allow     4768        1600             NaN      NaN   
2      allow      238         118             NaN      NaN   
3      allow     3327        1438             NaN      NaN   
4      allow    25358        6778             NaN      NaN   
...      ...      ...         ...             ...      ...   
65527  allow      314         192           122.0      6.0   
65528  allow  4680740       67312       4613428.0   4675.0   
65529   drop       70          70             0.0      1.0   
65530   drop       70          70             0.0      1.0   
65531   drop       70          70             0.0      1.0   

       Elapsed Time (sec)  pkts_sent  pkts_received  
0                      30          1              1  
1                      17         10              9  
2                    1199          1              1  
3                      17          8              7  
4                      16         13             18  
...                   ...        ...            ...  
65527                  15          4              2  
65528                  77        985           3690  
65529                   0          1              0  
65530                   0          1              0  
65531                   0          1              0  

[65532 rows x 12 columns]
```

In [19]:

```
from sklearn.impute import SimpleImputer
c = 'Packets'
imputer = SimpleImputer(strategy='mean')
print(f"\nMissing values in '{c}' before imputation:", d[c].isnull().sum())
d[c] = imputer.fit_transform(d[[c]])
print(f"\nMissing values in '{c}' after imputation:", d[c].isnull().sum())
```

```
Missing values in 'Packets' before imputation: 5

Missing values in 'Packets' after imputation: 0
```

In [20]:

```
print(d.isnull().sum())
```

```
Source Port            0
Destination Port       0
NAT Source Port        0
NAT Destination Port   0
Action                 0
Bytes                  0
Bytes Sent             0
Bytes Received         5
Packets                0
Elapsed Time (sec)     0
pkts_sent              0
pkts_received          0
dtype: int64
```

In [21]:

```
d_deleted = d.dropna()
```

In [22]:

```
print("\nShape of original DataFrame:", d.shape)
print("Shape of DataFrame after deletion:", d_deleted.shape)
```

```
Shape of original DataFrame: (65532, 12)
Shape of DataFrame after deletion: (65527, 12)
```

In [23]:

```
c='Bytes Received'
imputer = SimpleImputer(strategy='mean')
d[c] = imputer.fit_transform(d[[c]])
print(f"\nMissing values in '{c}' after imputation:", d[c].isnull().sum())
```

```
Missing values in 'Bytes Received' after imputation: 0
```

In [24]:

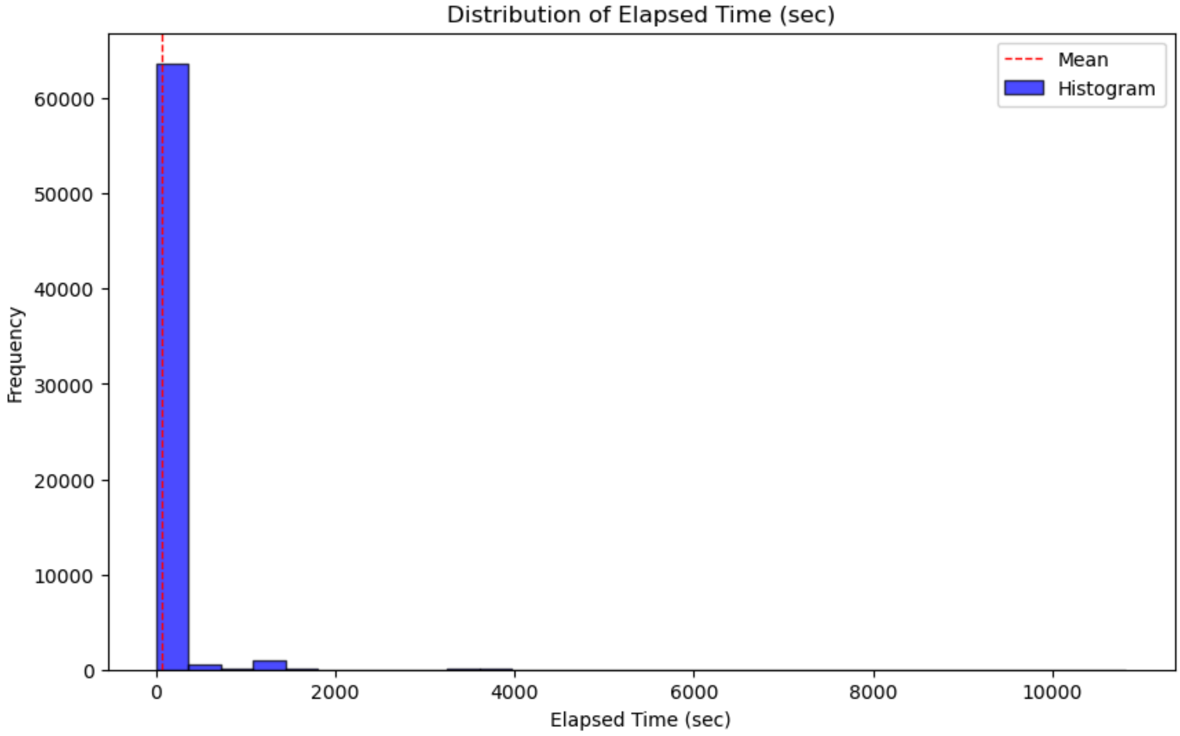
```
print(f"Missing values in '{c}':", d[c].isnull().sum())
```

```
Missing values in 'Bytes Received': 0
```

In [26]:

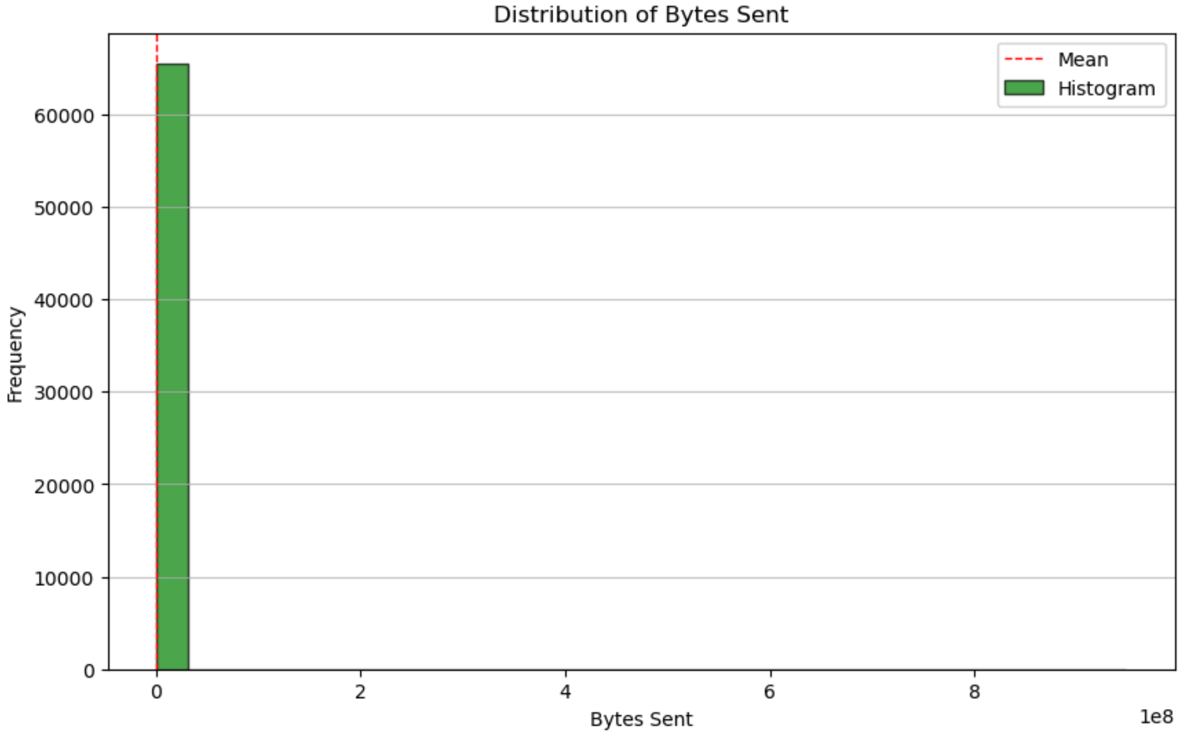
```
c = 'Elapsed Time (sec)'
print(f"Missing values in '{c}':", d[c].isnull().sum())
plt.figure(figsize=(10, 6))
plt.hist(d[c], bins=30, color='blue', edgecolor='black', alpha=0.7)
plt.title(f'Distribution of {c}')
plt.xlabel(c)
plt.ylabel('Frequency')
plt.axvline(x=d[c].mean(), color='red', linestyle='dashed', linewidth=1)
plt.legend(['Mean', 'Histogram'])
plt.show()
```

```
Missing values in 'Elapsed Time (sec)': 0
```



In [33]:
```
c = 'Bytes Sent'
print(f"Missing values in '{c}':", d[c].isnull().sum())
plt.figure(figsize=(10, 6))
plt.hist(d[c], bins=30, color='green', edgecolor='black', alpha=0.7)
plt.title(f'Distribution of {c}')
plt.xlabel(c)
plt.ylabel('Frequency')
plt.axvline(x=d[c].mean(), color='red', linestyle='dashed', linewidth=1)
plt.legend(['Mean', 'Histogram'])
plt.grid(axis='y', alpha=0.75)
```

```
Missing values in 'Bytes Sent': 0
```



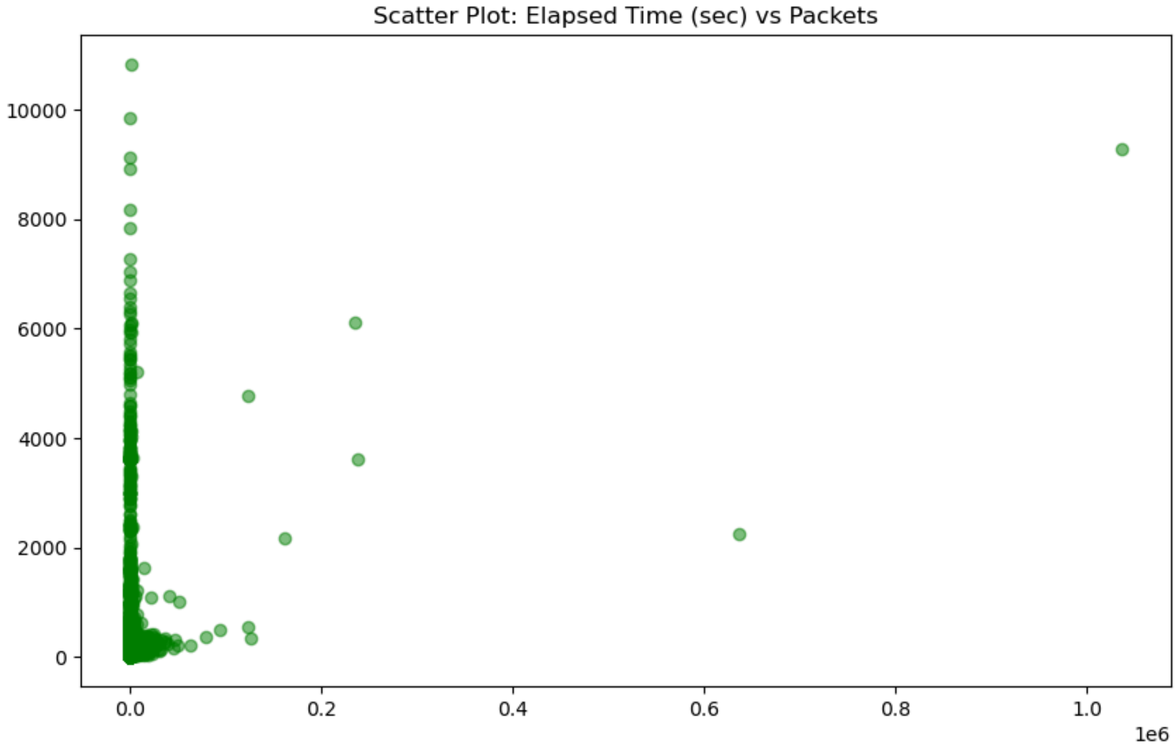
In [34]:
```
x_col = 'Packets'
y_col = 'Elapsed Time (sec)'
print(f"Missing values in '{x_col}':", d[x_col].isnull().sum())
print(f"Missing values in '{y_col}':", d[y_col].isnull().sum())
```

```
plt.figure(figsize=(10, 6))
plt.scatter(d[x_col], d[y_col], color='green', alpha=0.5)
plt.title(f'Scatter Plot: {y_col} vs {x_col}')
```

```
Missing values in 'Packets': 0
Missing values in 'Elapsed Time (sec)': 0
```

Out[34]:

```
Text(0.5, 1.0, 'Scatter Plot: Elapsed Time (sec) vs Packets')
```



In [44]:

```
n_cols = ['B_sent', 'B_received', 'P', 'E_time']
c_col = 'A'
m_scaled = d[n_cols].copy()
for col in n_cols:
    min_v = d[col].min()
    max_v = d[col].max()
    m_scaled[col] = (d[col] - min_v) / (max_v - min_v)
s_scaled = d[n_cols].copy()
for col in n_cols:
    mean_v = d[col].mean()
    std_v = d[col].std()
    s_scaled[col] = (d[col] - mean_v) / std_v
o_encoded = pd.get_dummies(d[c_col], prefix=c_col)
final_data = pd.concat([m_scaled.add_suffix('_minmax'), s_scaled.add_suffix('_stanc
o_encoded], axis=1)
print(final_data)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[44], line 3
      1 n_cols = ['B_sent', 'B_received', 'P', 'E_time']
      2 c_col = 'A'
----> 3 m_scaled = d[n_cols].copy()
      4 for col in n_cols:
      5     min_v = d[col].min()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3813, in Data
Frame.__getitem__(self, key)
   3811     if is_iterator(key):
   3812         key = list(key)
-> 3813     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3815 # take() does not accept boolean indexers
   3816 if getattr(indexer, "dtype", None) == bool:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6070,
in Index._get_indexer_strict(self, key, axis_name)
   6067 else:
   6068     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6070 self._raise_if_missing(keyarr, indexer, axis_name)
   6072 keyarr = self.take(indexer)
   6073 if isinstance(key, Index):
   6074     # GH 42790 - Preserve name from an Index

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6130,
in Index._raise_if_missing(self, key, indexer, axis_name)
   6128     if use_interval_msg:
   6129         key = list(key)
-> 6130     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6132 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
   6133 raise KeyError(f"{not_found} not in index")

KeyError: "None of [Index(['B_sent', 'B_received', 'P', 'E_time'], dtype='objec
t')] are in the [columns]"
```

In [48]:

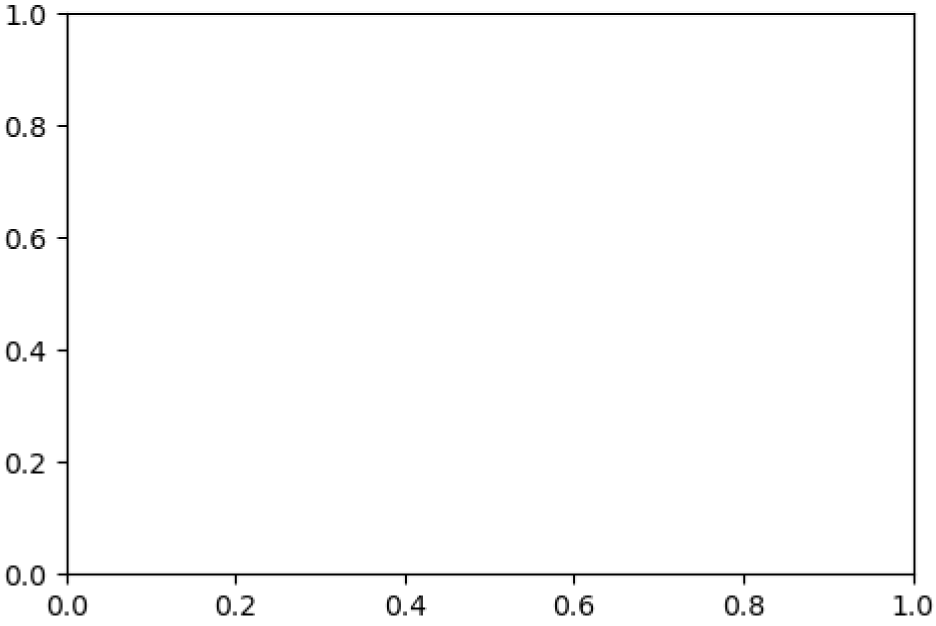
```
n_cols = ['Bytes sent', 'Bytes received', 'Packets', 'Elapsed time']
c_col = 'A'

m_scaled1 = d[n_cols].copy()
for col in n_cols:
    min_v = d[col].min()
    max_v = d[col].max()
    m_scaled[col] = (d[col] - min_v) / (max_v - min_v)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[48], line 4
      1 n_cols = ['Bytes sent', 'Bytes received', 'Packets', 'Elapsed time']
      2 c_col = 'A'
----> 4 m_scaled1 = d[n_cols].copy()
      5 for col in n_cols:
      6     min_v = d[col].min()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3813, in Data
Frame.__getitem__(self, key)
   3811     if is_iterator(key):
   3812         key = list(key)
-> 3813     indexer = self.columns._get_indexer_strict(key, "columns")[1]
   3815 # take() does not accept boolean indexers
   3816 if getattr(indexer, "dtype", None) == bool:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6070,
in Index._get_indexer_strict(self, key, axis_name)
   6067 else:
   6068     keyarr, indexer, new_indexer = self._reindex_non_unique(keyarr)
-> 6070 self._raise_if_missing(keyarr, indexer, axis_name)
   6072 keyarr = self.take(indexer)
   6073 if isinstance(key, Index):
   6074     # GH 42790 - Preserve name from an Index

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:6133,
in Index._raise_if_missing(self, key, indexer, axis_name)
   6130     raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   6132 not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
-> 6133 raise KeyError(f"{not_found} not in index")

KeyError: "['Bytes sent', 'Bytes received', 'Elapsed time'] not in index"
```

In [52]:

```python
n_cols = ['Bytes sent', 'Bytes received']
plt.figure(figsize=(12, 8))
for i, col in enumerate(n_cols):
    plt.subplot(2, len(n_cols), i + 1) # First row
    plt.hist(d[col], bins=20, color='blue', alpha=0.7)
    plt.title(f'Before Normalization: {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802,
in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Bytes sent'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[52], line 5
      3 for i, col in enumerate(n_cols):
      4     plt.subplot(2, len(n_cols), i + 1) # First row
----> 5     plt.hist(d[col], bins=20, color='blue', alpha=0.7)
      6     plt.title(f'Before Normalization: {col}')
      7     plt.xlabel(col)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3807, in DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804,
in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'Bytes sent'
```



In [53]:

```
s_scaled = d[n_cols].copy()
for col in n_cols:
mean_v = d[col].mean()
std_v = d[col].std()
s_scaled[col] = (d[col] - mean_v) / std_v
```

```
  Cell In[53], line 3
    mean_v = d[col].mean()
    ^
IndentationError: expected an indented block after 'for' statement on line 2
```

In [54]:

```

o_encoded = pd.get_dummies(d[c_col], prefix=c_col)

final_data = pd.concat([m_scaled.add_suffix('_minmax'), s_scaled.add_suffix('_stanc
o_encoded], axis=1)

print(final_data.head())
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'A'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[54], line 1
----> 1 o_encoded = pd.get_dummies(d[c_col], prefix=c_col)
      3 final_data = pd.concat([m_scaled.add_suffix('_minmax'), s_scaled.add_suffix('_standardized'),
      4 o_encoded], axis=1)
      6 print(final_data.head())

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3807, in DataFrame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'A'
```

In [55]:

```
f = 'firewall_data.csv'
d = pd.read_csv(f)
n_cols = ['B_sent', 'B_received', 'P']
c_col = 'A'
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[55], line 2
      1 f = 'firewall_data.csv'
----> 2 d = pd.read_csv(f)
      3 n_cols = ['B_sent', 'B_received', 'P']
      4 c_col = 'A'

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\util\_decorators.py:211, in deprecate_kwarg.<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\util\_decorators.py:331, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:950, in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:605, in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1442, in TextFileReader.__init__(self, f, engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\readers.py:1735, in TextFileReader._make_engine(self, f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\common.py:856, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'firewall_data.csv'
```

In [67]:

```
m_scaled = d[n_cols].copy()
for col in n_cols:
min_v = d[col].min()
max_v = d[col].max()
m_scaled[col] = (d[col] - min_v) / (max_v - min_v)
```

```
  Cell In[67], line 3
    min_v = d[col].min()
    ^
IndentationError: expected an indented block after 'for' statement on line 2
```

In [68]:

```
s_scaled = d[n_cols].copy()
for col in n_cols:
mean_v = d[col].mean()
std_v = d[col].std()
s_scaled[col] = (d[col] - mean_v) / std_v
```

```
  Cell In[68], line 3
    mean_v = d[col].mean()
    ^
IndentationError: expected an indented block after 'for' statement on line 2
```

In [69]:

```
o_encoded = pd.get_dummies(d[c_col], prefix=c_col)
final_data = pd.concat([m_scaled.add_suffix('_minmax'), s_scaled.add_suffix('_stanc
o_encoded], axis=1)
print(final_data.head())
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802,
in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in pan
das._libs.index.IndexEngine.get_loc()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in pan
das._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObj
ectHashTable.get_item()

KeyError: 'A'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[69], line 1
----> 1 o_encoded = pd.get_dummies(d[c_col], prefix=c_col)
      2 final_data = pd.concat([m_scaled.add_suffix('_minmax'), s_scaled.add_suffi
x('_standardized'),
      3 o_encoded], axis=1)
      4 print(final_data.head())

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\frame.py:3807, in Data
Frame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804,
in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 'A'
```

In [70]:

```
f = 'firewall_data.csv'
d = pd.read_csv(f)
n_cols = ['B_sent', 'B_received', 'P']
plt.figure(figsize=(12, 8))
for i, col in enumerate(n_cols):
plt.subplot(2, len(n_cols), i + 1) # First row
plt.hist(d[col], bins=20, color='blue', alpha=0.7)
plt.title(f'Before Normalization: {col}')
plt.xlabel(col)
plt.ylabel('Frequency')
```

```
  Cell In[70], line 6
    plt.subplot(2, len(n_cols), i + 1) # First row
    ^
IndentationError: expected an indented block after 'for' statement on line 5
```

In [71]:

```
m_scaled = d[n_cols].copy()
for col in n_cols:
min_v = d[col].min()
max_v = d[col].max()
m_scaled[col] = (d[col] - min_v) / (max_v - min_v)
```

```
  Cell In[71], line 3
    min_v = d[col].min()
    ^
IndentationError: expected an indented block after 'for' statement on line 2
```

In [72]:

```
for i, col in enumerate(n_cols):
plt.subplot(2, len(n_cols), i + 1 + len(n_cols))
plt.hist(m_scaled[col], bins=20, color='green', alpha=0.7)
plt.title(f'After Min-Max Scaling: {col}')
plt.xlabel(col + ' (scaled)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

```
  Cell In[72], line 2
    plt.subplot(2, len(n_cols), i + 1 + len(n_cols))
    ^
IndentationError: expected an indented block after 'for' statement on line 1
```

In [73]:

```
f = 'firewall_data.csv'
d = pd.read_csv(f)
n_cols = ['E_time', 'B_received', 'P']
plt.figure(figsize=(12, 8))
for i, col in enumerate(n_cols):
plt.subplot(2, len(n_cols), i + 1)
plt.hist(d[col], bins=20, color='blue', alpha=0.7)
plt.title(f'Before Normalization: {col}')
plt.xlabel(col)
plt.ylabel('Frequency')
```

```
  Cell In[73], line 6
    plt.subplot(2, len(n_cols), i + 1)
    ^
IndentationError: expected an indented block after 'for' statement on line 5
```

In [74]:

```
m_scaled = d[n_cols].copy()
for col in n_cols:
min_v = d[col].min()
max_v = d[col].max()
m_scaled[col] = (d[col] - min_v) / (max_v - min_v)
```

```
  Cell In[74], line 3
    min_v = d[col].min()
    ^
IndentationError: expected an indented block after 'for' statement on line 2
```

In [75]:

```
for i, col in enumerate(n_cols):
plt.subplot(2, len(n_cols), i + 1 + len(n_cols))
plt.hist(m_scaled[col], bins=20, color='green', alpha=0.7)
plt.title(f'After Min-Max Scaling: {col}')
plt.xlabel(col + ' (scaled)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

```
  Cell In[75], line 2
    plt.subplot(2, len(n_cols), i + 1 + len(n_cols))
    ^
IndentationError: expected an indented block after 'for' statement on line 1
```

In [76]:

```
f = 'firewall_data.csv'
d = pd.read_csv(f)
a = 'E_time'
b = 'B_received'
c = 'P'
plt.figure(figsize=(10, 6))
for i, col in enumerate([a, b, c]):
plt.subplot(1, 3, i + 1)
plt.boxplot(d[col].dropna())
plt.title(f'Box Plot of {col}')
plt.ylabel(col)
plt.tight_layout()
plt.show()
```

```
  Cell In[76], line 8
    plt.subplot(1, 3, i + 1)
    ^
IndentationError: expected an indented block after 'for' statement on line 7
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: