In [55]:
```
import pandas as pd
import numpy as np
import collection as counter
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from wordcloud import WordCloud
```

In [29]:
```
df = pd.read_csv("new_clean_email.csv")
```

In [51]:
```
df.head(2)
```

Out[51]:

	label	urls	hour	day_of_week	sender_username	sender_domain	subject_clean	body_c
0	1	1	23	1	young	iworld.de	never agree to be a loser	buc trou cause s dimen
1	1	1	23	1	s1983	icable.ph	befriend jenna jameson	upg you pleas with t te

In [40]:
```
df.dropna(subset=['subject_clean'], inplace=True)
```

In [42]:
```
df.isnull().sum()
```

Out[42]:
```
label             0
urls              0
hour              0
day_of_week       0
sender_username    0
sender_domain      0
subject_clean      0
body_clean         0
dtype: int64
```

In [44]:
```
df.info()
```

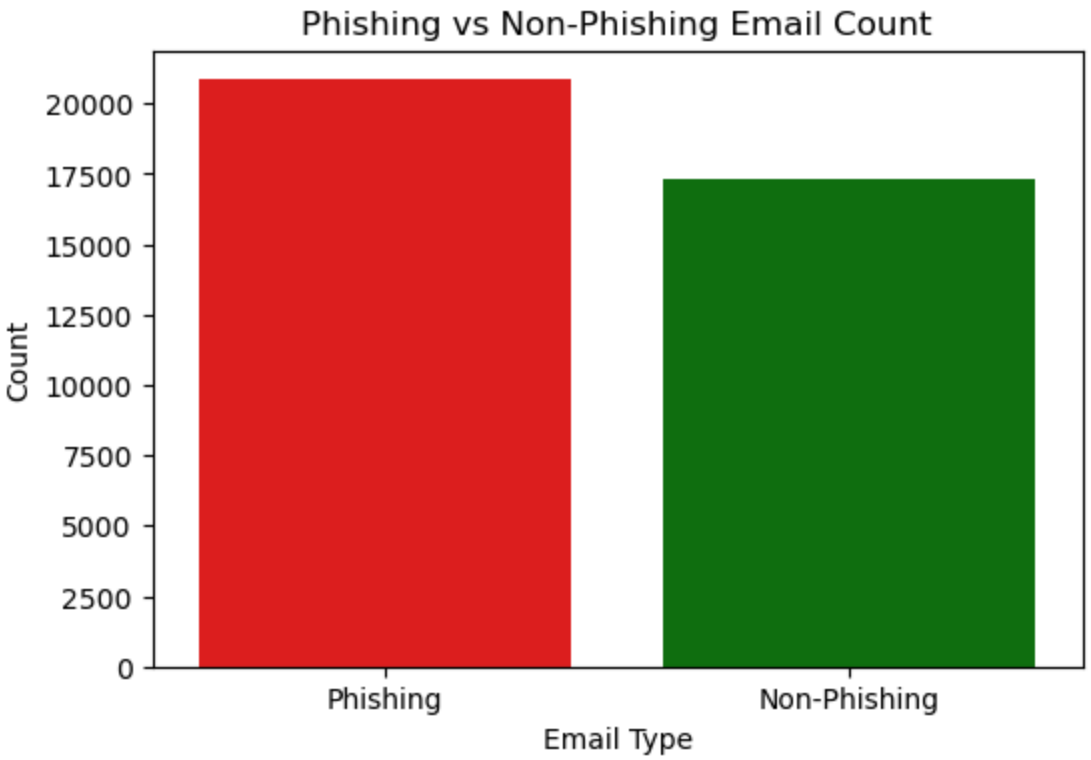
```
<class 'pandas.core.frame.DataFrame'>
Index: 38132 entries, 0 to 38163
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   label            38132 non-null  int64 
 1   urls             38132 non-null  int64 
 2   hour             38132 non-null  int64 
 3   day_of_week      38132 non-null  int64 
 4   sender_username  38132 non-null  object
 5   sender_domain    38132 non-null  object
 6   subject_clean    38132 non-null  object
 7   body_clean       38132 non-null  object
dtypes: int64(4), object(4)
memory usage: 2.6+ MB
```

In [47]:
```
df.describe()
```

Out[47]:

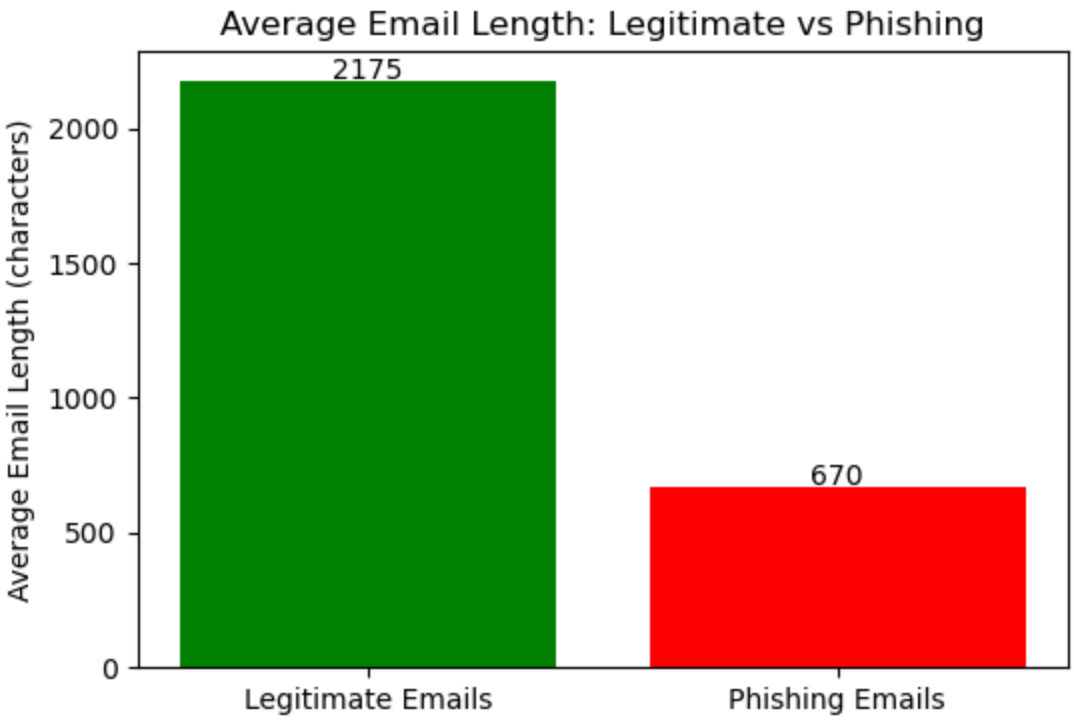
	label	urls	hour	day_of_week
count	38132.000000	38132.000000	38132.000000	38132.000000
mean	0.546549	0.662042	10.362347	2.669412
std	0.497835	0.473020	6.817354	0.884873
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	4.000000	2.000000
50%	1.000000	1.000000	10.000000	3.000000
75%	1.000000	1.000000	16.000000	3.000000
max	1.000000	1.000000	23.000000	6.000000

In [65]:
```
plt.figure(figsize=(6,4))

# Define custom colors
custom_palette = {'Non-Phishing': 'green', 'Phishing': 'red'}

# Use 'label_name' for both x and hue, and disable the legend
sns.countplot(x='label_name', hue='label_name', data=df, palette=custom_palette, le

plt.title('Phishing vs Non-Phishing Email Count')
plt.xlabel('Email Type')
plt.ylabel('Count')

plt.legend([],[], frameon=False)  # Hide legend explicitly (optional)

plt.show()
```



In [190…

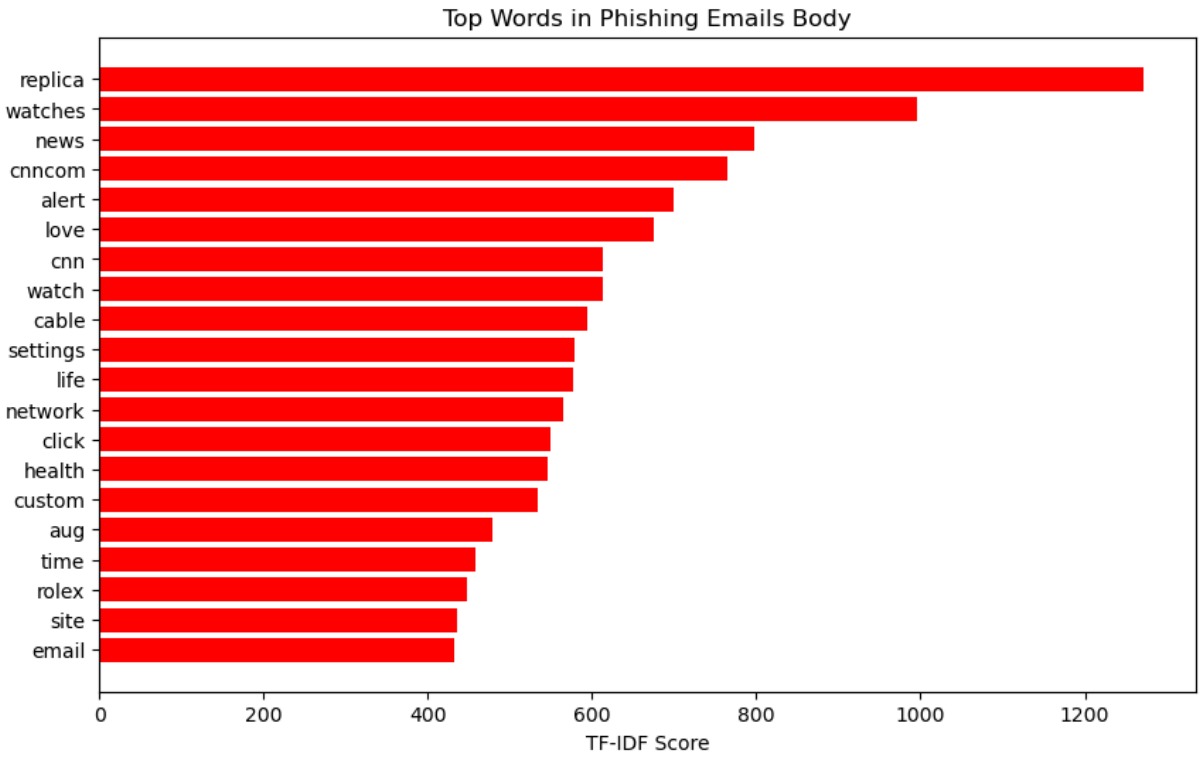
```
df['length'] = df['body_clean'].fillna('').apply(len)
# Calculate average length for each label
avg_length = df.groupby('label')['length'].mean()

# Map label numbers to readable names
labels = ['Legitimate Emails', 'Phishing Emails']

# Plot
plt.figure(figsize=(6, 4))
plt.bar(labels, avg_length, color=['green', 'red'])
plt.ylabel('Average Email Length (characters)')
plt.title('Average Email Length: Legitimate vs Phishing')

# Add value labels on bars
for i, v in enumerate(avg_length):
    plt.text(i, v + 5, f"{v:.0f}", ha='center')

plt.show()
```

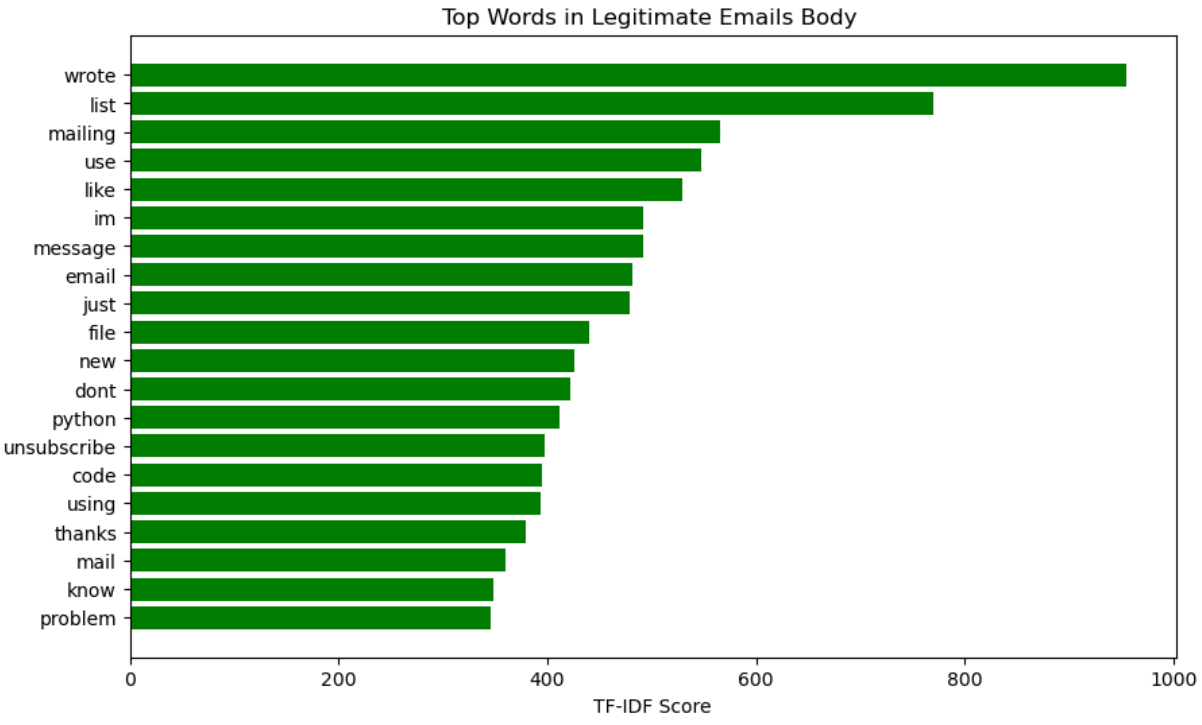


In [169…

```
vectorizer = TfidfVectorizer(
    stop_words='english',
    max_features=1000,
    token_pattern=r'(?u)\b[a-zA-Z]+\b'  # only tokens with letters (no numbers)
)

# Fit vectorizer on full body text (non-null)
body_texts = df['body_clean'].dropna()
vectorizer.fit(body_texts)

# Transform phishing and legitimate separately with same vectorizer
X_phish = vectorizer.transform(df[df['label'] == 1]['body_clean'].dropna())
X_legit = vectorizer.transform(df[df['label'] == 0]['body_clean'].dropna())

# Get feature names and sum TF-IDF scores
words = vectorizer.get_feature_names_out()
phish_scores = X_phish.sum(axis=0).A1
legit_scores = X_legit.sum(axis=0).A1

# Create DataFrames sorted by descending TF-IDF score (top 20)
phish_df = pd.DataFrame({'word': words, 'score': phish_scores}).sort_values(by='sco
legit_df = pd.DataFrame({'word': words, 'score': legit_scores}).sort_values(by='sco

# Plotting function for clarity & reuse
def plot_top_words(df_words, title, color):
    plt.figure(figsize=(10,6))
    plt.barh(df_words['word'], df_words['score'], color=color)
    plt.title(title)
    plt.gca().invert_yaxis()
    plt.xlabel("TF-IDF Score")
    plt.show()

# Plot phishing top words
```

```
plot_top_words(phish_df, "Top Words in Phishing Emails Body", "red")

# Plot legit top words
plot_top_words(legit_df, "Top Words in Legitimate Emails Body ", "green")
```

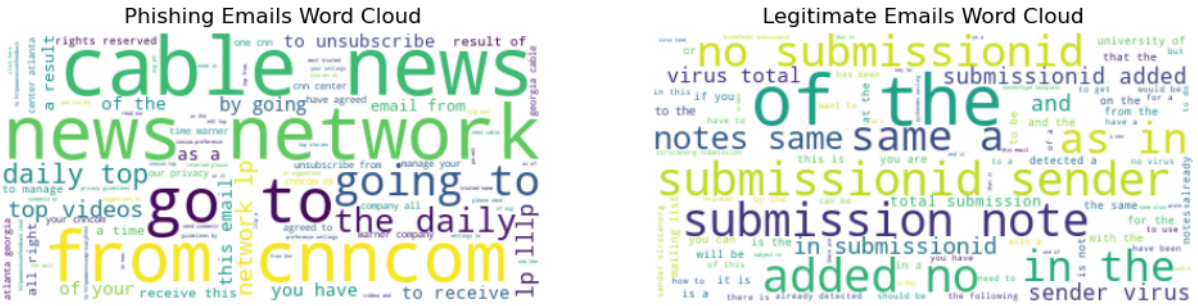




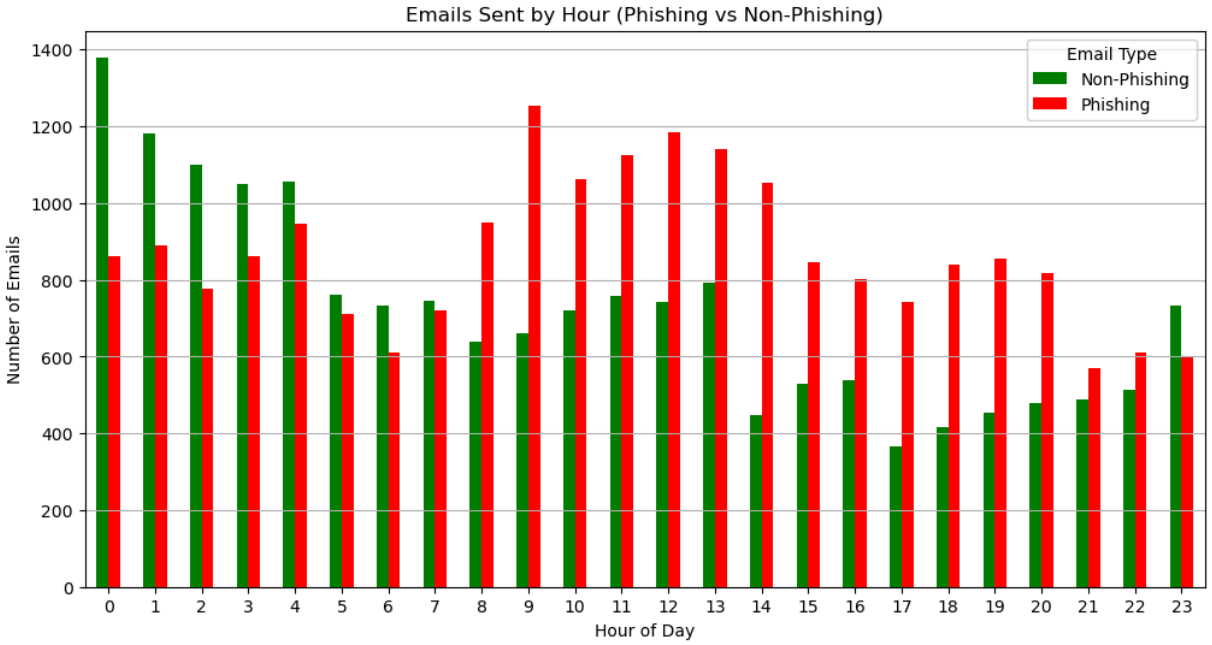
In [178…

```
phish_text = ' '.join(df[df['label'] == 1]['body_clean'].dropna())

# Legitimate emails text
legit_text = ' '.join(df[df['label'] == 0]['body_clean'].dropna())

# Generate word clouds
```

```
phish_wc = WordCloud(stopwords='english', background_color='white', max_words=100).
legit_wc = WordCloud(stopwords='english', background_color='white', max_words=100).

# Plot both with titles
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(phish_wc, interpolation='bilinear')
plt.axis('off')
plt.title('Phishing Emails Word Cloud')

plt.subplot(1, 2, 2)
plt.imshow(legit_wc, interpolation='bilinear')
plt.axis('off')
plt.title('Legitimate Emails Word Cloud')

plt.show()
```

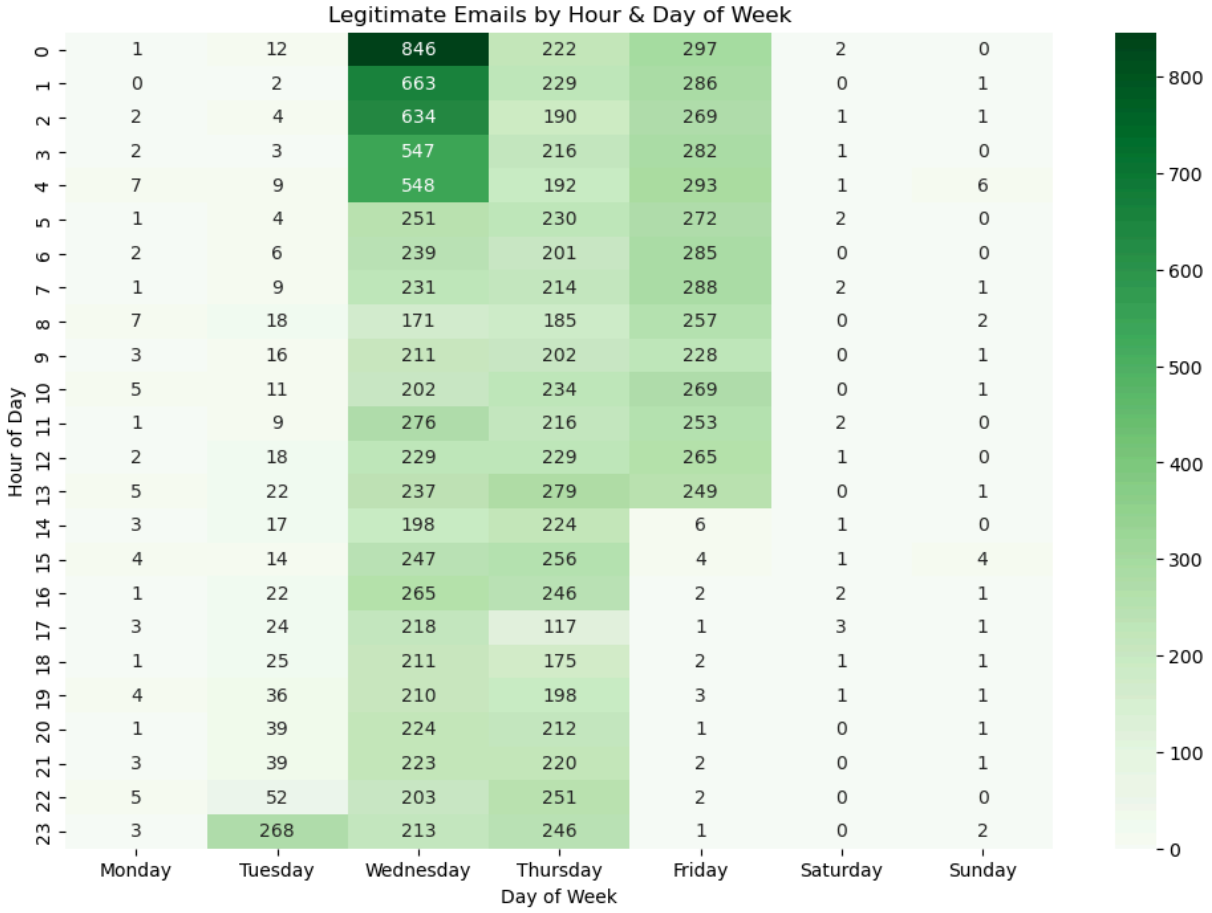


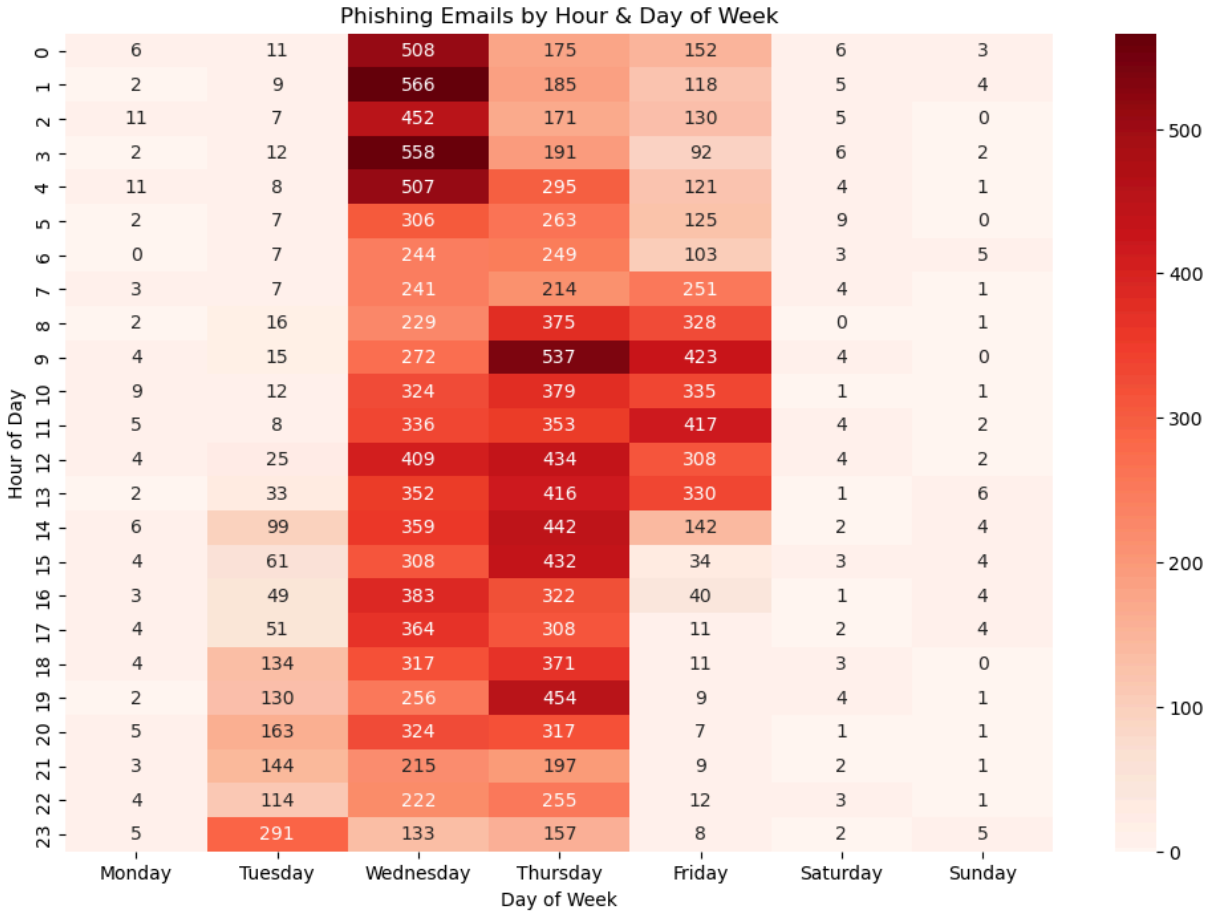
In [82]:
```
hour_label_counts[['Non-Phishing', 'Phishing']].plot(kind='bar', color=['green', 'r

plt.title('Emails Sent by Hour (Phishing vs Non-Phishing)')
plt.xlabel('Hour of Day')
plt.ylabel('Number of Emails')
plt.xticks(rotation=0)
plt.legend(title='Email Type')
plt.grid(axis='y')

plt.show()
```



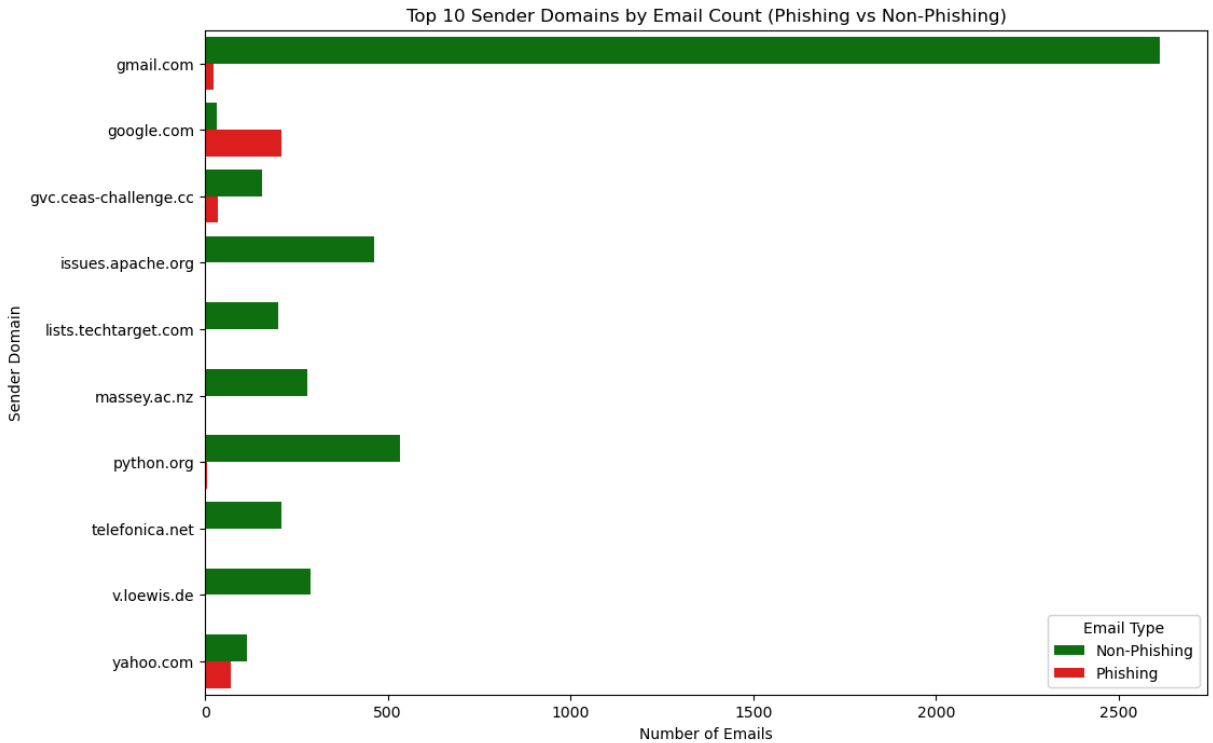
In [100…

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

# Map numeric day_of_week to names if needed
days_map = {0:'Monday',1:'Tuesday',2:'Wednesday',3:'Thursday',4:'Friday',5:'Saturda
df['day_name'] = df['day_of_week'].map(days_map)

day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'S

# Prepare pivot tables
pivot_legit = df[df['label_name'] == 'Non-Phishing'].groupby(['hour', 'day_name']).
pivot_legit = pivot_legit[day_order]

pivot_phishing = df[df['label_name'] == 'Phishing'].groupby(['hour', 'day_name']).s
pivot_phishing = pivot_phishing[day_order]

# Plot Legitimate emails heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_legit, annot=True, fmt='d', cmap='Greens')
plt.title('Legitimate Emails by Hour & Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Hour of Day')
plt.show()

# Plot Phishing emails heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_phishing, annot=True, fmt='d', cmap='Reds')
plt.title('Phishing Emails by Hour & Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Hour of Day')
plt.show()
```


Legitimate Emails by Hour & Day of Week
Hour of Day
Day of Week
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
0 1 12 846 222 297 2 0
1 0 2 663 229 286 0 1
2 2 4 634 190 269 1 1
3 2 3 547 216 282 1 0
4 7 9 548 192 293 1 6
5 1 4 251 230 272 2 0
6 2 6 239 201 285 0 0
7 1 9 231 214 288 2 1
8 7 18 171 185 257 0 2
9 3 16 211 202 228 0 1
10 5 11 202 234 269 0 1
11 1 9 276 216 253 2 0
12 2 18 229 229 265 1 0
13 5 22 237 279 249 0 1
14 3 17 198 224 6 1 0
15 4 14 247 256 4 1 4
16 1 22 265 246 2 2 1
17 3 24 218 117 1 3 1
18 1 25 211 175 2 1 1
19 4 36 210 198 3 1 1
20 1 39 224 212 1 0 1
21 3 39 223 220 2 0 1
22 5 52 203 251 2 0 0
23 3 268 213 246 1 0 2
800
700
600
500
400
300
200
100
0


Phishing Emails by Hour & Day of Week
Hour of Day
Day of Week
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
0 6 11 508 175 152 6 3
1 2 9 566 185 118 5 4
2 11 7 452 171 130 5 0
3 2 12 558 191 92 6 2
4 11 8 507 295 121 4 1
5 2 7 306 263 125 9 0
6 0 7 244 249 103 3 5
7 3 7 241 214 251 4 1
8 2 16 229 375 328 0 1
9 4 15 272 537 423 4 0
10 9 12 324 379 335 1 1
11 5 8 336 353 417 4 2
12 4 25 409 434 308 4 2
13 2 33 352 416 330 1 6
14 6 99 359 442 142 2 4
15 4 61 308 432 34 3 4
16 3 49 383 322 40 1 4
17 4 51 364 308 11 2 4
18 4 134 317 371 11 3 0
19 2 130 256 454 9 4 1
20 5 163 324 317 7 1 1
21 3 144 215 197 9 2 1
22 4 114 222 255 12 3 1
23 5 291 133 157 8 2 5
500
400
300
200
100
0

In [94]:
```
domain_label_counts = df.groupby(['sender_domain', 'label_name']).size().reset_inde

# Get top 10 domains by total emails (phishing + non-phishing)
top_domains = (domain_label_counts.groupby('sender_domain')['count']
               .sum()
               .sort_values(ascending=False)
               .head(10)
               .index)

# Filter data for top domains only
top_domain_data = domain_label_counts[domain_label_counts['sender_domain'].isin(top

plt.figure(figsize=(12,8))

# Use seaborn barplot with hue for label
sns.barplot(
    data=top_domain_data,
    y='sender_domain',
    x='count',
    hue='label_name',
    palette={'Non-Phishing': 'green', 'Phishing': 'red'}
)

plt.title('Top 10 Sender Domains by Email Count (Phishing vs Non-Phishing)')
plt.xlabel('Number of Emails')
plt.ylabel('Sender Domain')
plt.legend(title='Email Type')
plt.show()
```

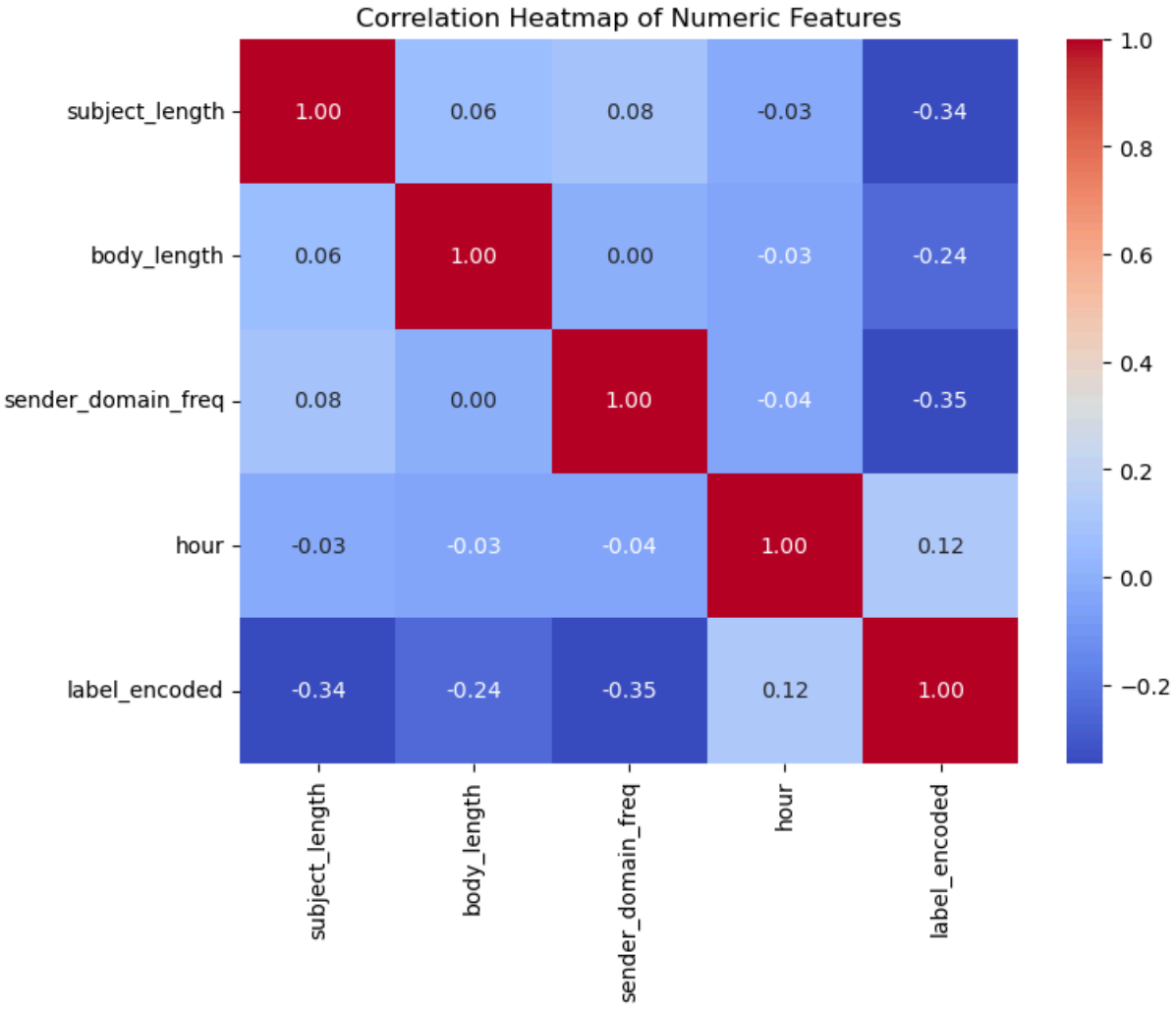


In [102…
```
df['subject_length'] = df['subject_clean'].str.len()
df['body_length'] = df['body_clean'].str.len()
```

```
# Count frequency of sender domains
domain_counts = df['sender_domain'].value_counts()
df['sender_domain_freq'] = df['sender_domain'].map(domain_counts)
```

In [104…

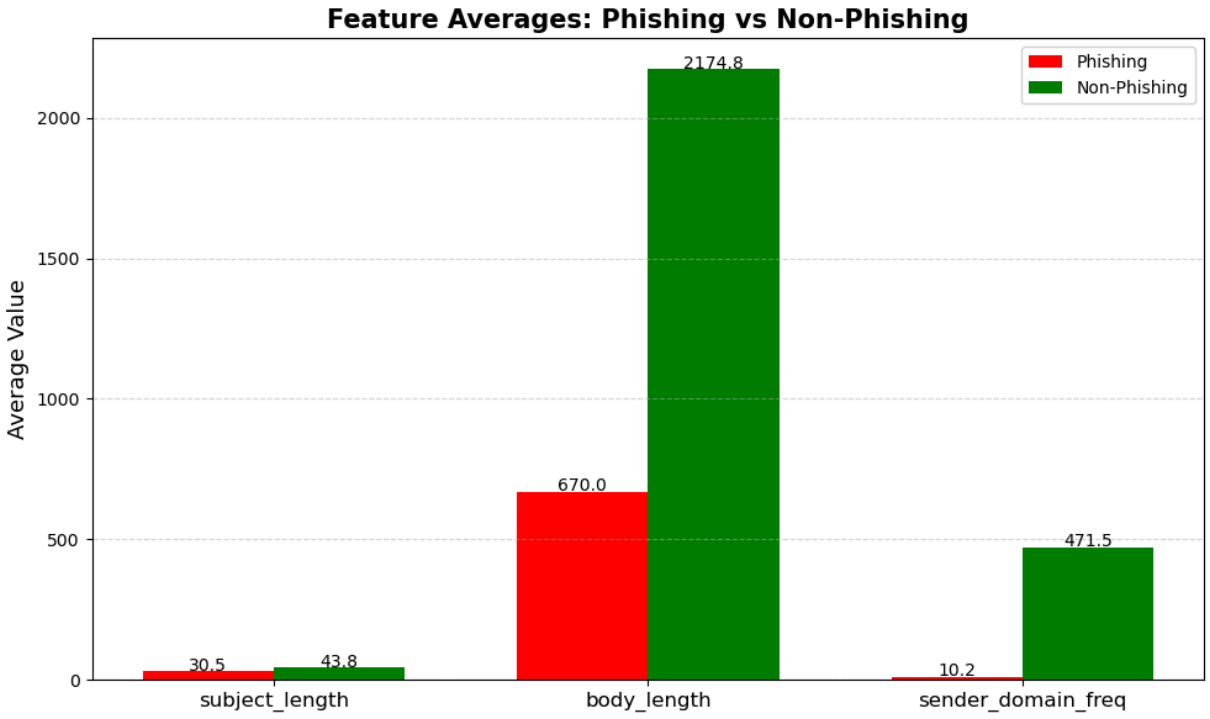
```
df['label_encoded'] = df['label_name'].map({'Non-Phishing': 0, 'Phishing': 1})

# Select numeric columns
numeric_cols = ['subject_length', 'body_length', 'sender_domain_freq', 'hour', 'lab

corr = df[numeric_cols].corr()

plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```



In [116…

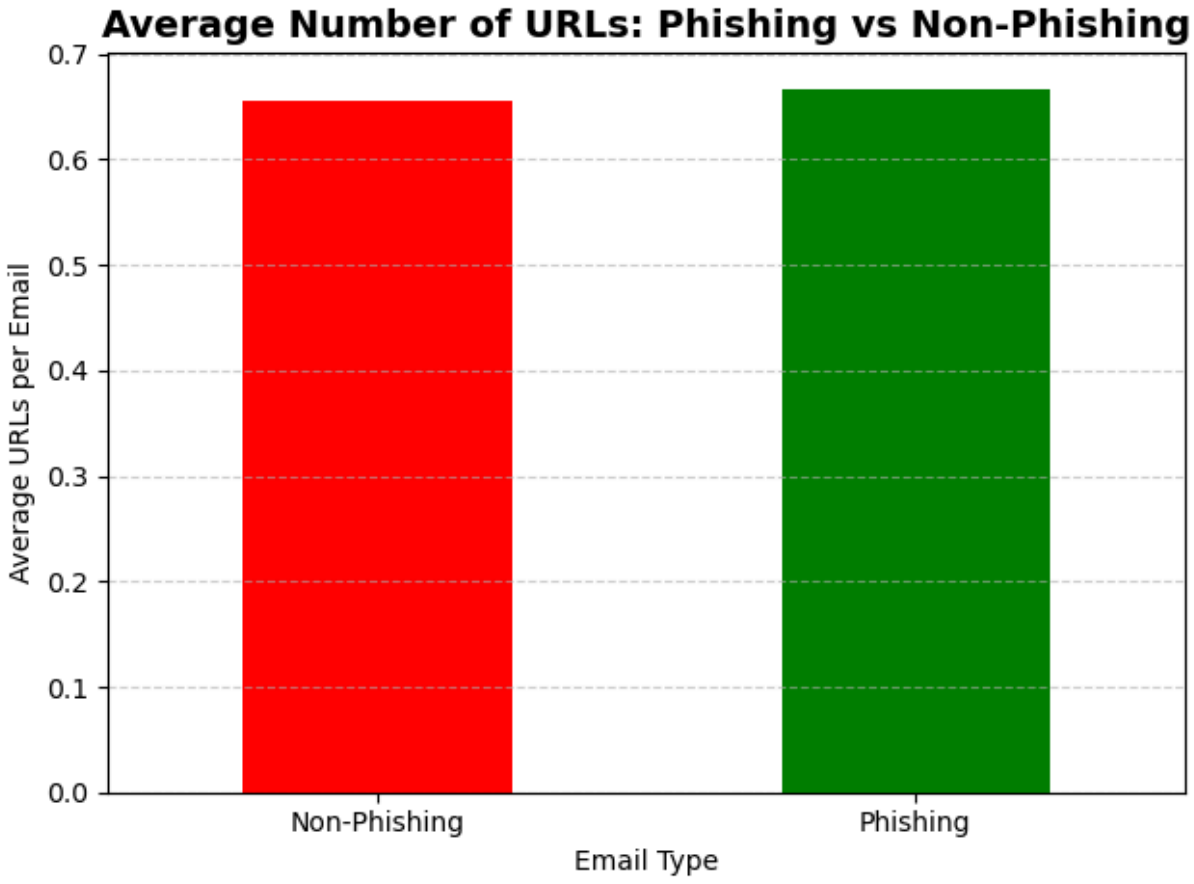
```
grouped_means = df.groupby('label_name')[['subject_length', 'body_length', 'sender_
features = grouped_means.columns
labels = grouped_means.index

x = np.arange(len(features))
bar_width = 0.35

# Plot bars
```

```
plt.figure(figsize=(10, 6))
plt.bar(x - bar_width/2, grouped_means.loc['Phishing'].values, width=bar_width, col
plt.bar(x + bar_width/2, grouped_means.loc['Non-Phishing'].values, width=bar_width,

# Add labels, title, legend
plt.xticks(x, features, fontsize=12)
plt.ylabel('Average Value', fontsize=13)
plt.title('Feature Averages: Phishing vs Non-Phishing', fontsize=15, fontweight='bo
plt.legend()
plt.grid(axis='y', linestyle='--', alpha=0.5)

# Annotate bar values (using .iloc to avoid warnings)
for i in range(len(features)):
    plt.text(x[i] - bar_width/2, grouped_means.loc['Phishing'].iloc[i] + 2,
             f"{grouped_means.loc['Phishing'].iloc[i]:.1f}", ha='center', color='bl
    plt.text(x[i] + bar_width/2, grouped_means.loc['Non-Phishing'].iloc[i] + 2,
             f"{grouped_means.loc['Non-Phishing'].iloc[i]:.1f}", ha='center', color

plt.tight_layout()
plt.show()
```



In [124…

```
# Group by phishing label and calculate mean URLs
df.groupby('label_name')['urls'].mean().plot(kind='bar', color=['red', 'green'])

plt.title('Average Number of URLs: Phishing vs Non-Phishing', fontsize=14, fontweig
plt.ylabel('Average URLs per Email')
plt.xlabel('Email Type')
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```


Average Number of URLs: Phishing vs Non-Phishing
Average URLs per Email
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Non-Phishing
Phishing
Email Type