Advanced Fraud Detection for Credit Card Transactions

Author: Mohan Krishna

Credit card fraud detection aims to identify fraudulent transactions and protect customers and businesses from financial losses. In this project, we use machine learning models to analyze past credit card transactions and predict whether a new transaction is fraudulent.

Credit card fraud is when someone uses another person's credit card or account information to make unauthorized purchases or access funds through cash advances. Credit card fraud doesn’t just happen online; it happens in brick-and-mortar stores, too. As a business owner, you can avoid serious headaches – and unwanted publicity – by recognizing potentially fraudulent use of credit cards in your payment environment.

Problem

The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be a fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

Observation

- Very few transactions are actually fraudulent (less than 1%). The data set is highly skewed, consisting of 492 frauds in a total of 284,807 observations. This resulted in only 0.172% fraud cases. This skewed set is justified by the low number of fraudulent transactions.
- The dataset consists of numerical values from the 28 ‘Principal Component Analysis (PCA)’ transformed features, namely V1 to V28. Furthermore, there is no metadata about the original features provided, so pre-analysis or feature study could not be done.
- The ‘Time’ and ‘Amount’ features are not transformed data.
- There is no missing value in the dataset.

Buisness Questions

Since all features are anonymous, we will focus our analysis on non-anonymized features: Time, Amount

How different is the amount of money used in different transaction classes?

Do fraudulent transactions occur more often during a certain frames?

Importing Libraries

In [33]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.warn('ignore', FutureWarning)
%matplotlib inline
sns.set_style("whitegrid")
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

In [34]:
```
df = pd.read_csv('creditcard.csv')
df.head()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[34]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V
0	0.00	-1.36	-0.07	2.54	1.38	-0.34	0.46	0.24	0.10	0.36	...	-0.02	0.28	-0.11	0.07	0
1	0.00	1.19	0.27	0.17	0.45	0.06	-0.08	-0.08	0.09	-0.26	...	-0.23	-0.64	0.10	-0.34	0
2	1.00	-1.36	-1.34	1.77	0.38	-0.50	1.80	0.79	0.25	-1.51	...	0.25	0.77	0.91	-0.69	-0
3	1.00	-0.97	-0.19	1.79	-0.86	-0.01	1.25	0.24	0.38	-1.39	...	-0.11	0.01	-0.19	-1.18	0
4	2.00	-1.16	0.88	1.55	0.40	-0.41	0.10	0.59	-0.27	0.82	...	-0.01	0.80	-0.14	0.14	-0

5 rows × 31 columns

Exploratory Data Analysis

In [35]:
```
df.info()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [36]:
```
#decreasing decimals in the dataset
pd.set_option("display.float", "{:.2f}".format)
df.describe()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[36]:

	Time	V1	V2	V3	V4	V5	V6	V7
count	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00	284807.00
mean	94813.86	0.00	0.00	-0.00	0.00	-0.00	0.00	-0.00
std	47488.15	1.96	1.65	1.52	1.42	1.38	1.33	1.24
min	0.00	-56.41	-72.72	-48.33	-5.68	-113.74	-26.16	-43.56
25%	54201.50	-0.92	-0.60	-0.89	-0.85	-0.69	-0.77	-0.55
50%	84692.00	0.02	0.07	0.18	-0.02	-0.05	-0.27	0.04
75%	139320.50	1.32	0.80	1.03	0.74	0.61	0.40	0.57
max	172792.00	2.45	22.06	9.38	16.88	34.80	73.30	120.59

8 rows × 31 columns

Check the missing value in dataset

In [37]:
```
df.isnull().sum().sum()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[37]:
```
0
```

In [38]:
```
df.columns
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[38]:
```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

The only non-transformed variables to work with are:

- Time
- Amount
- Class (1: fraud, 0: not_fraud)

In [39]:
```
labels = ["Safe", "Fraud"]

count_classes = pd.value_counts(df['Class'], sort=True)
count_classes.plot(kind='bar', rot=0)
plt.title("Transaction Class Distribution")
plt.xticks(range(2), labels)
plt.xlabel("Class")
plt.ylabel("Frequency")
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[39]:
```
Text(0, 0.5, 'Frequency')
```

Here we can see that, in this dataset very few transactions are actually fraudlent

In [40]:
```
df['Class'].value_counts()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[40]:
```
0    284315
1       492
Name: Class, dtype: int64
```

Notice how imbalanced is our original dataset! Most of the transactions are non-fraud. If we use this dataframe as the base for our predictive models and analysis we might get a lot of errors and our algorithms will probably overfit since it will "assume" that most transactions are not fraud. But we don't want our model to assume, we want our model to detect patterns that give signs of fraud!

Statistical Analtsis

- For dealing with outilers, IQR(Inter Quanrtile Range) in which we will eliminate the outliers those are less than 10th percentile greater than 90th percentile.

In [41]:
```
Q1 = df.quantile(0.25)
Q2 = df.quantile(0.75)
IQR = Q2-Q1
print("IQR of whole dataset: ")
print(IQR)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
IQR of whole dataset:
Time     85119.00
V1           2.24
V2           1.40
V3           1.92
V4           1.59
V5           1.30
V6           1.17
V7           1.12
V8           0.54
V9           1.24
V10          0.99
V11          1.50
V12          1.02
V13          1.31
V14          0.92
V15          1.23
V16          0.99
V17          0.88
V18          1.00
V19          0.92
V20          0.34
V21          0.41
V22          1.07
V23          0.31
V24          0.79
V25          0.67
V26          0.57
V27          0.16
V28          0.13
Amount      71.56
Class        0.00
dtype: float64
```

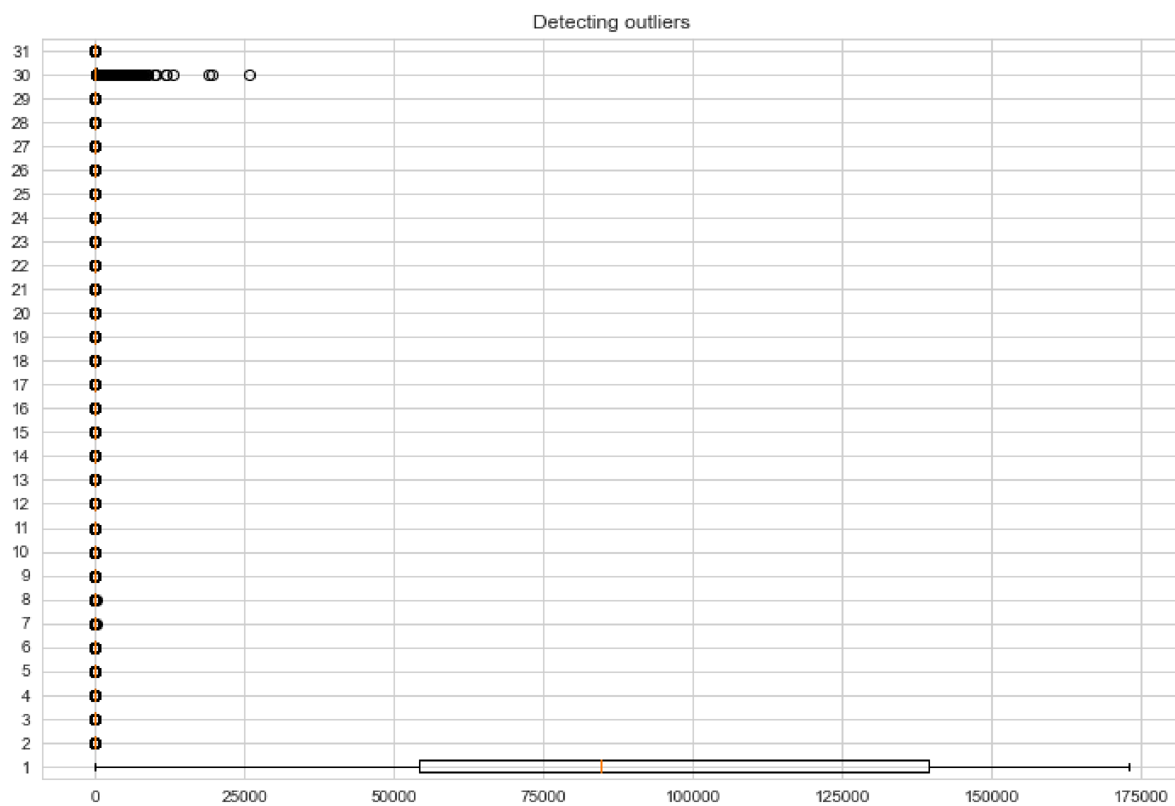
In [42]:
```
print("Skewness of the data: ")
df_skew = df.skew()
print(df_skew)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Skewness of the data:
Time     -0.04
V1       -3.28
V2       -4.62
V3       -2.24
V4        0.68
V5       -2.43
V6        1.83
V7        2.55
V8       -8.52
V9        0.55
V10       1.19
V11       0.36
V12      -2.28
V13       0.07
V14      -2.00
V15      -0.31
V16      -1.10
V17      -3.84
V18      -0.26
V19       0.11
V20      -2.04
V21       3.59
V22      -0.21
V23      -5.88
V24      -0.55
V25      -0.42
V26       0.58
V27      -1.17
V28      11.19
Amount   16.98
Class    24.00
dtype: float64
```

Box Plot of the data

In [43]:
```
print("Detecting Outilers:\n ")
plt.figure(figsize=(12,8))
plt.boxplot(df, vert=False)
plt.title("Detecting outliers")
plt.show()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Detecting Outilers:
```

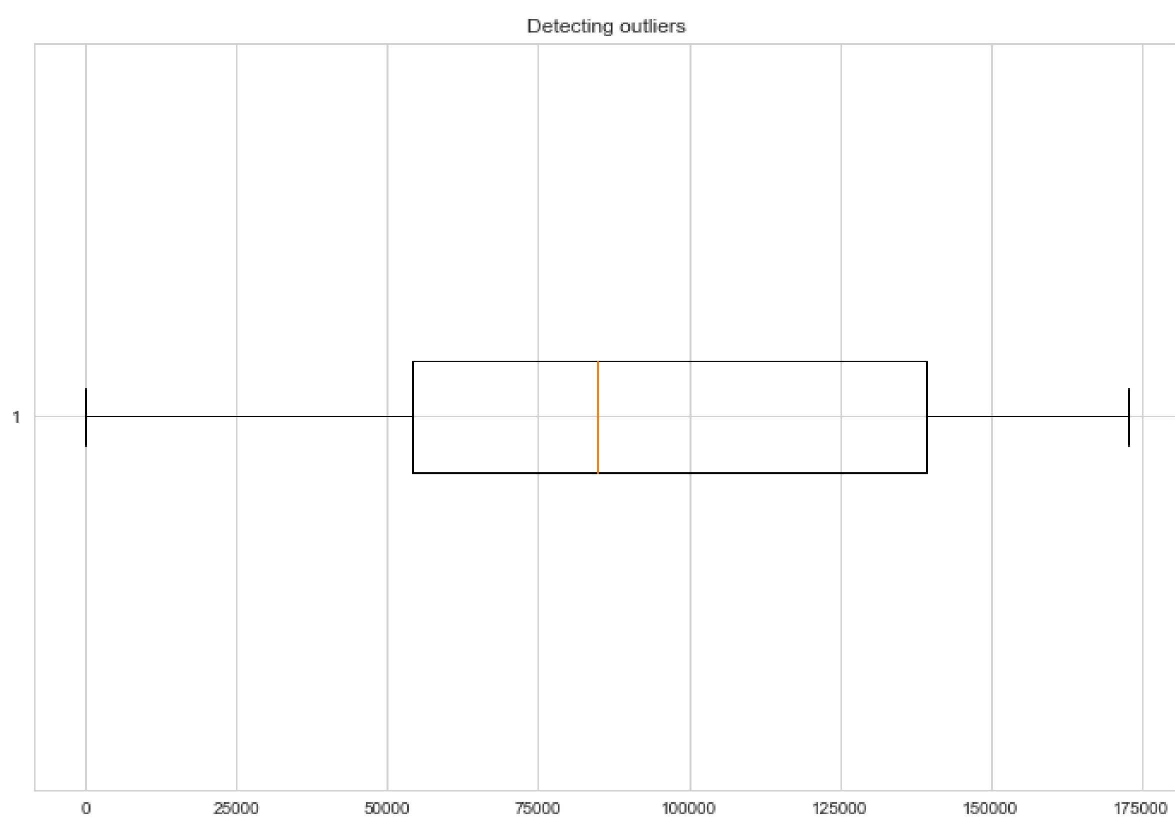


- Time and Amount y-axis displaying outliers, lets check about those.

Note: Columns other than Time, Amount and Class contains numbers within certain range, we

In [44]:
```
def box_out(df):
    print("Detecting Outilers:\n ")
    plt.figure(figsize=(12,8))
    plt.boxplot(df, vert=False)
    plt.title("Detecting outliers")
    plt.show()

print("Time Column")
box_out(df['Time'])
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Time Column
Detecting Outilers:
```



- There is no problem in Time column

Analyzing **Amount** feature for better undersating on Amount

In [45]:
```
df['Amount'].describe()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

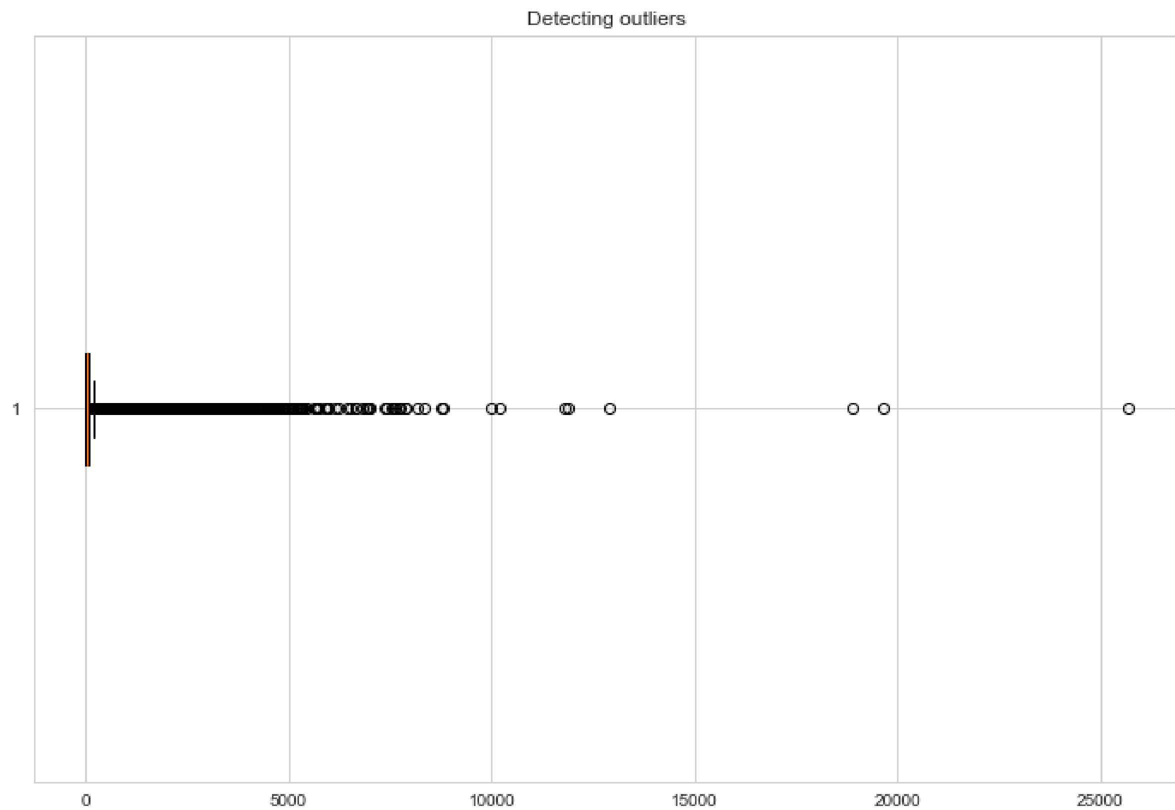
Out[45]:
```
count    284807.00
mean         88.35
std         250.12
min           0.00
25%           5.60
50%          22.00
75%          77.16
max       25691.16
Name: Amount, dtype: float64
```

Above description showed us that m range of maximum and minimum amount of transaction is between 0-25691

Box plot on Amount to identiying the outliers form that column

In [46]:
```
print("Outliers of Amount: :")
box_out(df['Amount'])
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Outliers of Amount: :
Detecting Outilers:
```



- There are good amount of outilers in the Amount column. Amount column is too important in this data.
- Above image showing us that after 900 there are outilers which are distributed till 25000+

Removing outliers

with **Quantile based Flooring and capping**

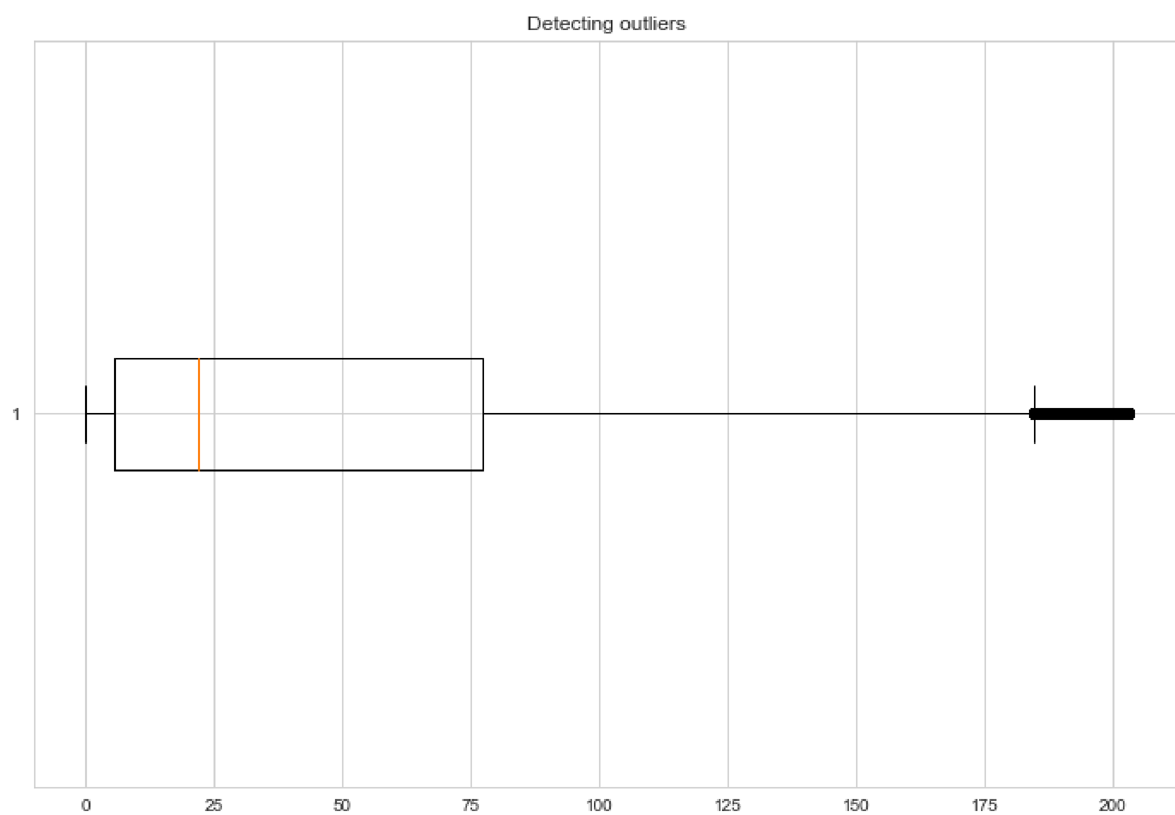
In [47]:
```
# Percentiles
print("10th percentile of Amount: ")
print(df["Amount"].quantile(0.10))
print("90th percentile of price: ")
print(df["Amount"]. quantile(0.90))
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
10th percentile of Amount: 
1.0
90th percentile of price: 
203.0
```

- From above percentiles (10 & 90) we can remove data points those are out of this range but there may not ourliers under 10th percentile.
- So lets remove outliers those are greater than 90th percentile and plot a boxplot so we can see if there are any outliers less than 10th percentile.

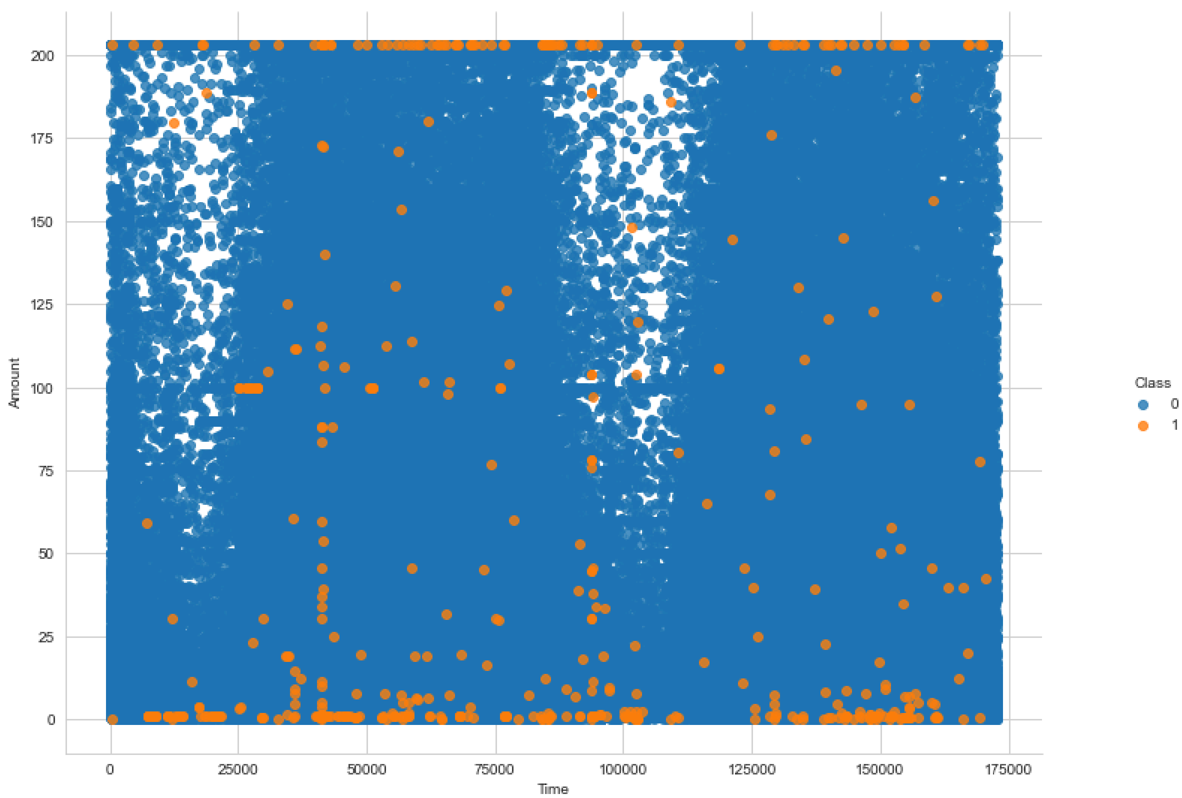
In [48]:
```
df['Amount'] = np.where(df['Amount']>203.0, 203.0, df['Amount'])

print("After removing outliers >90th percentile: :")
box_out(df['Amount'])
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
After removing outliers >90th percentile: :
Detecting Outilers:
```



- We eliminated most of the outlier those are greater than 90th percentile from the data.
- Let's Keep remaining tail, coz that contains good amount of points

In [49]:
```
sns.lmplot('Time', 'Amount', df, hue='Class', fit_reg=False)
fig = plt.gcf()
fig.set_size_inches(12, 8)
plt.show()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```



In [50]:
```
fraud = df[df['Class']==1]
safe = df[df['Class']==0]

print(f"Shape of Fraudlent transactions: {fraud.shape}")
print(f"Shape of Non-Fraudlent transactions: {safe.shape}")
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Shape of Fraudlent transactions: (492, 31)
Shape of Non-Fraudlent transactions: (284315, 31)
```

How different are the amount of money used in different transaction classes?

In [51]:
```
pd.concat([fraud.Amount.describe(), safe.Amount.describe()], axis=1)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[51]:

	Amount	Amount
count	492.00	284315.00
mean	61.82	53.68
std	78.30	65.99
min	0.00	0.00
25%	1.00	5.65
50%	9.25	22.00
75%	105.89	77.05
max	203.00	203.00

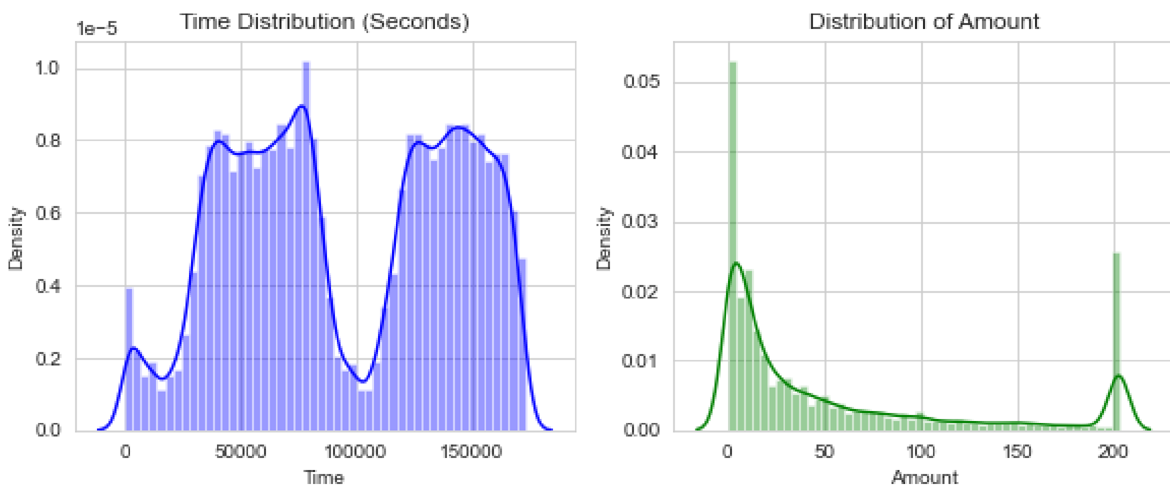
Do fraudulent transactions occur more often during certain time frame ?

In [52]:
```
pd.concat([fraud.Time.describe(), safe.Time.describe()], axis=1)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[52]:

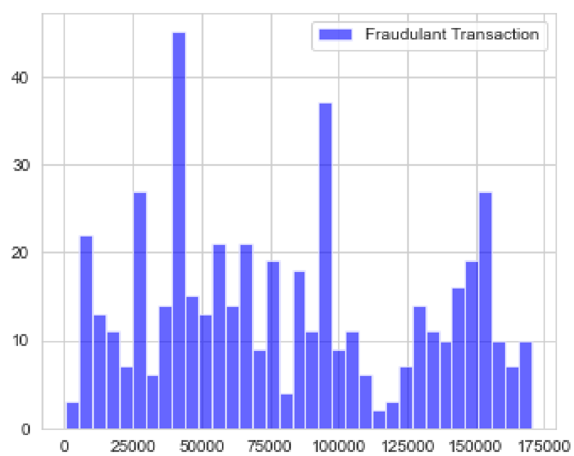
	Time	Time
count	492.00	284315.00
mean	80746.81	94838.20
std	47835.37	47484.02
min	406.00	0.00
25%	41241.50	54230.00
50%	75568.50	84711.00
75%	128483.00	139333.00
max	170348.00	172792.00

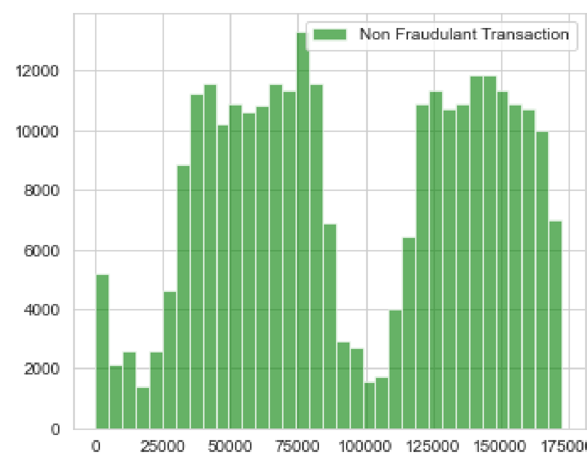
In [53]:
```
plt.figure(figsize=(10,8))

plt.subplot(2, 2, 1)
plt.title('Time Distribution (Seconds)')

sns.distplot(df['Time'], color='blue');

#plot the amount feature
plt.subplot(2, 2, 2)
plt.title('Distribution of Amount')
sns.distplot(df['Amount'],color='green');
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```



In [54]:
```
plt.figure(figsize=(12, 10))

plt.subplot(2, 2, 1)
df[df.Class == 1].Time.hist(bins=35, color='blue', alpha=0.6, label="Fraudulan
plt.legend()

plt.subplot(2, 2, 2)
df[df.Class == 0].Time.hist(bins=35, color='green', alpha=0.6, label="Non Frau
plt.legend()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[54]:
```
<matplotlib.legend.Legend at 0x1d8a8c7d790>
```



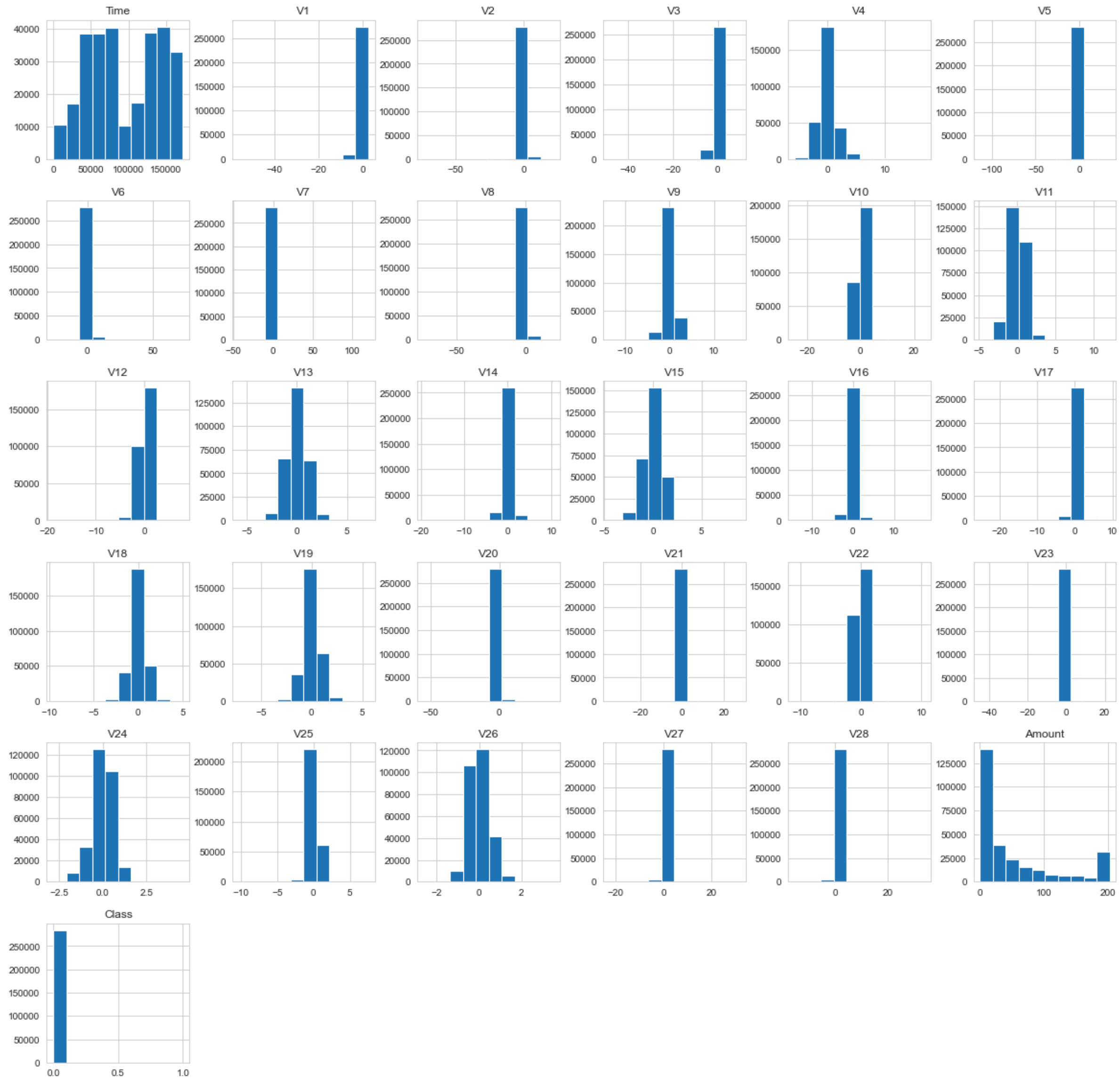


By seeing the distributions we can have an idea how skewed are these features, we can also see further distributions of the other features. There are techniques that can help the distributions be less skewed which will be implemented in this notebook in the future.

Doesn't seem like the time of transaction really matters here as per above observation. Now let us take a sample of the dataset for out modelling and prediction

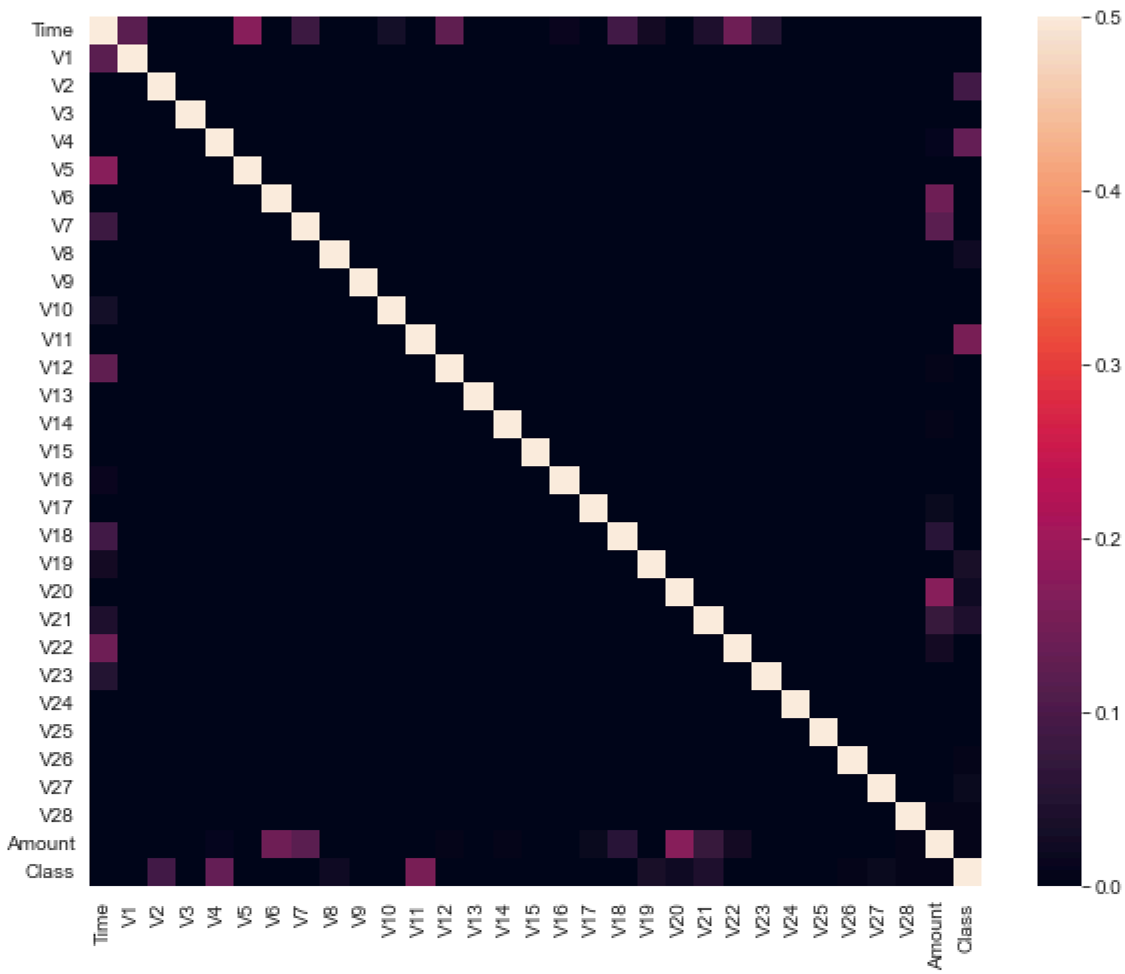
In [55]:
```
df.hist(figsize=(20,20))
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Out[55]:
```
array([[<AxesSubplot:title={'center':'Time'}>,
        <AxesSubplot:title={'center':'V1'}>,
        <AxesSubplot:title={'center':'V2'}>,
        <AxesSubplot:title={'center':'V3'}>,
        <AxesSubplot:title={'center':'V4'}>,
        <AxesSubplot:title={'center':'V5'}>],
       [<AxesSubplot:title={'center':'V6'}>,
        <AxesSubplot:title={'center':'V7'}>,
        <AxesSubplot:title={'center':'V8'}>,
        <AxesSubplot:title={'center':'V9'}>,
        <AxesSubplot:title={'center':'V10'}>,
        <AxesSubplot:title={'center':'V11'}>],
       [<AxesSubplot:title={'center':'V12'}>,
        <AxesSubplot:title={'center':'V13'}>,
        <AxesSubplot:title={'center':'V14'}>,
        <AxesSubplot:title={'center':'V15'}>,
        <AxesSubplot:title={'center':'V16'}>,
        <AxesSubplot:title={'center':'V17'}>],
       [<AxesSubplot:title={'center':'V18'}>,
        <AxesSubplot:title={'center':'V19'}>,
        <AxesSubplot:title={'center':'V20'}>,
        <AxesSubplot:title={'center':'V21'}>,
        <AxesSubplot:title={'center':'V22'}>,
        <AxesSubplot:title={'center':'V23'}>],
       [<AxesSubplot:title={'center':'V24'}>,
        <AxesSubplot:title={'center':'V25'}>,
        <AxesSubplot:title={'center':'V26'}>,
        <AxesSubplot:title={'center':'V27'}>,
        <AxesSubplot:title={'center':'V28'}>,
        <AxesSubplot:title={'center':'Amount'}>],
       [<AxesSubplot:title={'center':'Class'}>, <AxesSubplot:>,
        <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>, <AxesSubplot:>]],
      dtype=object)
```



In [56]:
```
# df.corr()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

In [57]:
```
#Lets find high correlations

plt.figure(figsize=(10,8))
sns.heatmap(data=df.corr(), vmin=0,vmax=0.5, annot=False)
plt.show()
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```



Highest correlations come from:

- Time & V3 (-0.42)
- Amount & V2 (-0.53)

- Amount & V4 (0.4)

- While these correlations are high, I don't expect it to run the risk of multicollinearity.
- The correlation matrix shows also that none of the V1 to V28 PCA components have any correlation to each other however if we observe Class has some form positive and negative correlations with the V components but has no correlation with Time and Amount.

Data Processing

Time and Amount should be scaled as the other columns.

Spliting dataset into train and test

In [58]:
```
from sklearn.model_selection import train_test_split

X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

In [59]:
```
print(X.shape, X_train.shape, X_test.shape)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
(284807, 30) (199364, 30) (85443, 30)
```

Standardization

In [60]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

Training Machine Learning Model

It a supervises ML task with classification problem.

XGBoost

(If model will not train well by above algorithms, then we can train with **ANN**)

In [61]:
```
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

XGBClassifier

In [62]:
```
print("Training with XGBoost Classifier: ")
xgb = XGBClassifier()
xgb.fit(X_train, y_train)

print("\nScore of XGBClassifier: ")
xgb.score(X_train, y_train)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
Training with XGBoost Classifier:

C:\Users\Asus\anaconda3\lib\site-packages\xgboost\sklearn.py:1146: UserWarnin
g: The use of label encoder in XGBClassifier is deprecated and will be remove
d in a future release. To remove this warning, do the following: 1) Pass opti
on use_label_encoder=False when constructing XGBClassifier object; and 2) Enc
ode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_clas
s - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[15:47:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.
4.0/src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation me
tric used with the objective 'binary:logistic' was changed from 'error' to 'l
ogloss'. Explicitly set eval_metric if you'd like to restore the old behavio
r.

Score of XGBClassifier:
```

Out[62]:
```
1.0
```

Evaluating the Model

Score of training

In [63]:
```
from sklearn.metrics import accuracy_score, classification_report, confusion_ma
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

In [65]:
```
train_pred = xgb.predict(X_train)
print(accuracy_score(train_pred, y_train))
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
1.0
```

Score of testing

In [66]:
```
test_pred = xgb.predict(X_test)
print(accuracy_score(test_pred, y_test))
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
0.9996839998595555
```

Classification Report

In [67]:
```
print(classification_report(test_pred, y_test))
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85326
           1       0.83      0.97      0.89       117

    accuracy                           1.00     85443
   macro avg       0.92      0.98      0.95     85443
weighted avg       1.00      1.00      1.00     85443
```

Confusion Matrix

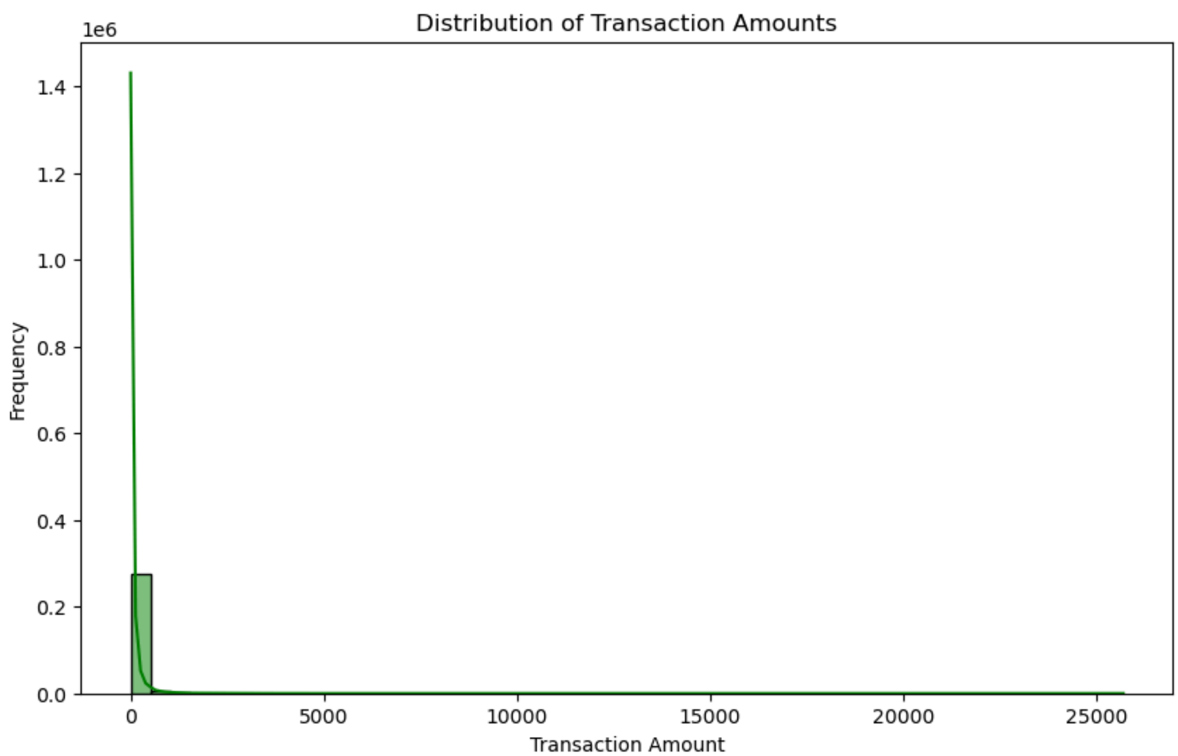
In [71]:
```
cm = confusion_matrix(y_test, test_pred)
print("Confusion Matrix : \n", cm)
# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```

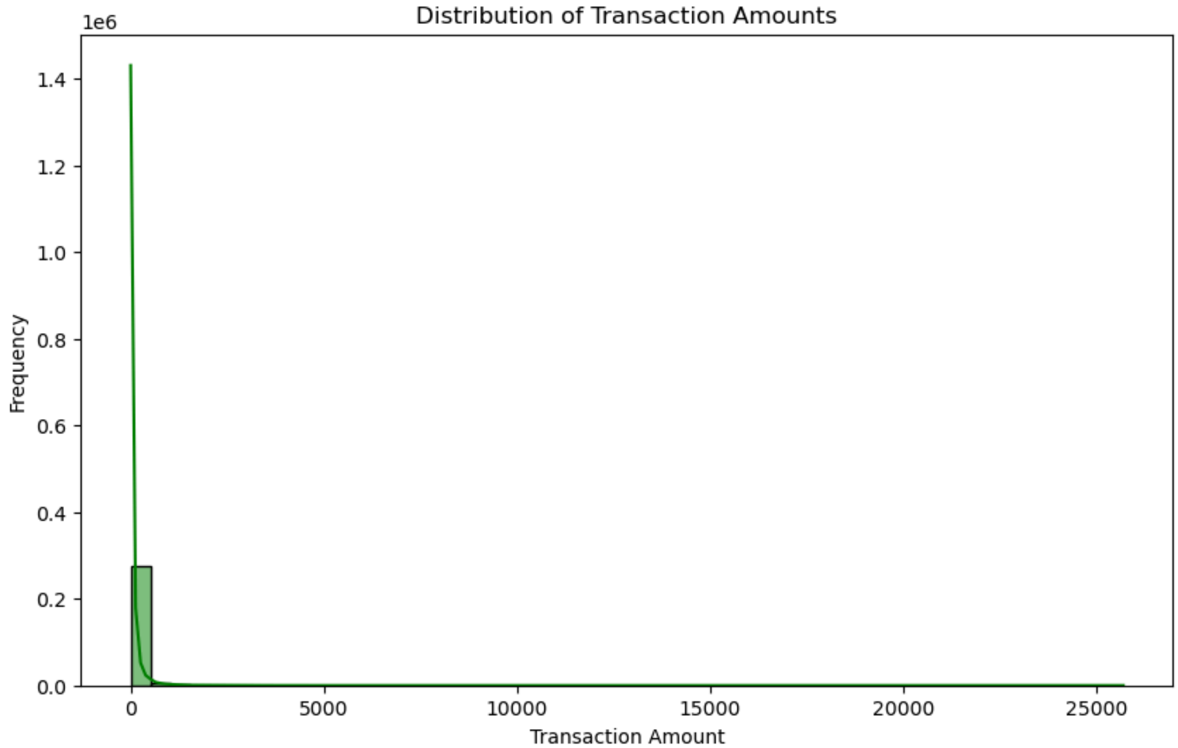
```
Confusion Matrix :
 [[85303     4]
 [   23   113]]
```

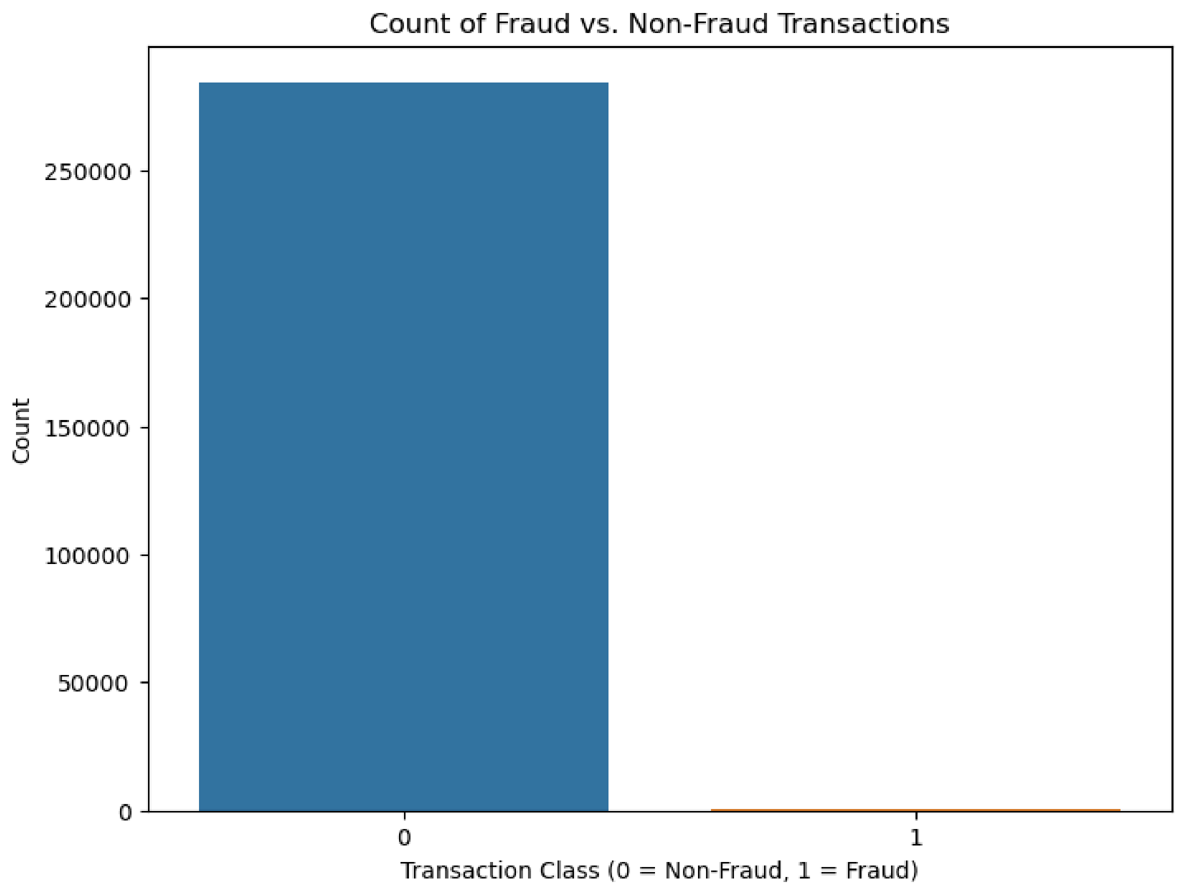
Our model perofmed very well for this problem.

There is no need of tunning.

In [8]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('creditcard.csv')

# Now, you can use the DataFrame for plotting
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Distribution of transaction amounts
plt.figure(figsize=(10,6))
sns.histplot(df['Amount'], bins=50, kde=True, color='green')
plt.title('Distribution of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()

# Count plot for fraud vs. non-fraud transactions
plt.figure(figsize=(8,6))
sns.countplot(x='Class', data=df)
plt.title('Count of Fraud vs. Non-Fraud Transactions')
plt.xlabel('Transaction Class (0 = Non-Fraud, 1 = Fraud)')
plt.ylabel('Count')
plt.show()
```




Distribution of Transaction Amounts
1e6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
0
5000
10000
15000
20000
25000
Transaction Amount


Count of Fraud vs. Non-Fraud Transactions
250000
200000
150000
100000
50000
0
Count
0
1
Transaction Class (0 = Non-Fraud, 1 = Fraud)

In [ ]: