1. Import Libraries and Load Data

In [42]:
```python
import numpy as np

# data processing

import pandas as pd
import numpy as np

# data visualization

import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
```

In [43]:
```python
df=pd.read_csv("creditcard.csv")
```

In [44]:
```python
df.head()
```

Out[44]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8

5 rows × 31 columns

In [45]:
```python
df.tail()
```

Out[45]:

	Time	V1	V2	V3	V4	V5	V6	V7	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.2
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.7
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.4

5 rows × 31 columns

2. Basic Information

In [46]:
```
df.info()
```

```
<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
```

3. Rows and Columns

In [47]:
```
rows, columns = df.shape
print(f'The dataset contains {rows} rows and {columns} columns.')
```

```
The dataset contains 284807 rows and 31 columns.
```

4. Finding null values

In [48]:
```
df.isnull().sum()
```

Out[48]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

5. Display column names and data types

In [49]:
```
print(df.columns)
```

```
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')
```

In [50]:
```
print(df.dtypes)
```

```
Time      float64
V1        float64
V2        float64
V3        float64
V4        float64
V5        float64
V6        float64
V7        float64
V8        float64
V9        float64
V10       float64
V11       float64
V12       float64
V13       float64
V14       float64
V15       float64
V16       float64
V17       float64
V18       float64
V19       float64
V20       float64
V21       float64
V22       float64
V23       float64
V24       float64
V25       float64
V26       float64
V27       float64
V28       float64
Amount    float64
Class       int64
dtype: object
```

6. STATISTICS

In [51]:
```
df.describe()
```

Out[51]:

	Time	V1	V2	V3	V4	V5
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01

8 rows × 31 columns

7. Count of fraudulent and legitimate transactions

In [52]:
```
class_counts =df['Class'].value_counts()
fraudulent = class_counts[1]
legitimate = class_counts[0]
print(f"\nNumber of Fraudulent Transactions: {fraudulent}")
print(f"Number of Legitimate Transactions: {legitimate}")

transaction_counts = df['Class'].value_counts()
print(transaction_counts)
```

```
Number of Fraudulent Transactions: 492
Number of Legitimate Transactions: 284315
0    284315
1       492
Name: Class, dtype: int64
```

8. Percentage of fraudulent transactions

In [53]:
```
fraud_percentage = (transaction_counts[1] / rows) * 100
print(f'Percentage of fraudulent transactions: {fraud_percentage:.2f}%')
```

```
Percentage of fraudulent transactions: 0.17%
```

9. Statistics for the 'Amount' column

In [54]:
```
amount_stats = df['Amount'].describe()
print(amount_stats)

amount_stats = df['Amount'].describe()
print(amount_stats[['min', 'max', 'mean', '50%']])  # 50% is the median
```

```
count    284807.000000
mean         88.349619
std         250.120109
min           0.000000
25%           5.600000
50%          22.000000
75%          77.165000
max       25691.160000
Name: Amount, dtype: float64
min         0.000000
max     25691.160000
mean       88.349619
50%        22.000000
Name: Amount, dtype: float64
```

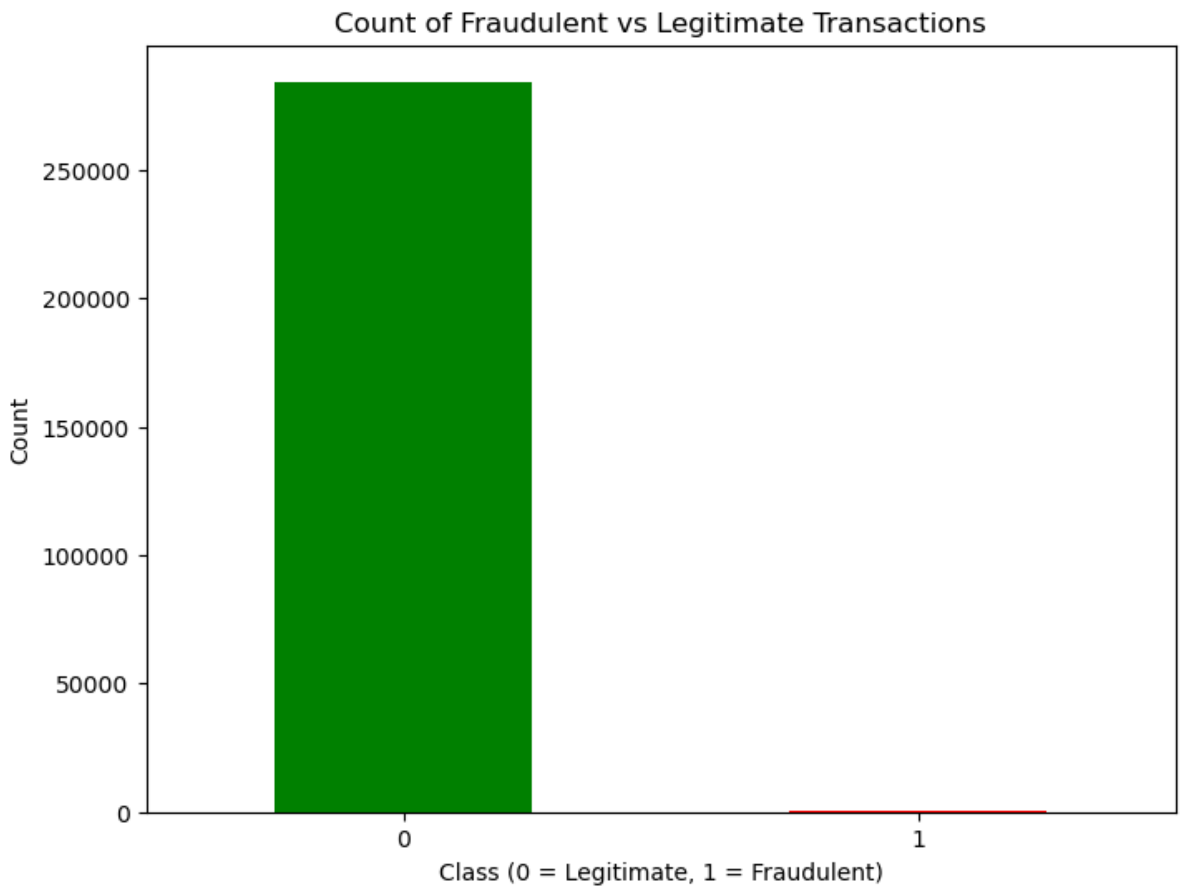
10. Maximum transaction amount and its fraud status

In [55]:
```
max_transaction_idx = df['Amount'].idxmax()
max_transaction_amount = df.loc[max_transaction_idx, 'Amount']
is_fraudulent = df.loc[max_transaction_idx, 'Class']
print(f"\nMaximum Transaction Amount: {max_transaction_amount}")
print(f"Is the Maximum Transaction Fraudulent? {'Yes' if is_fraudulent == 1 else 'N
```

```
Maximum Transaction Amount: 25691.16
Is the Maximum Transaction Fraudulent? No it is Legitimate
```

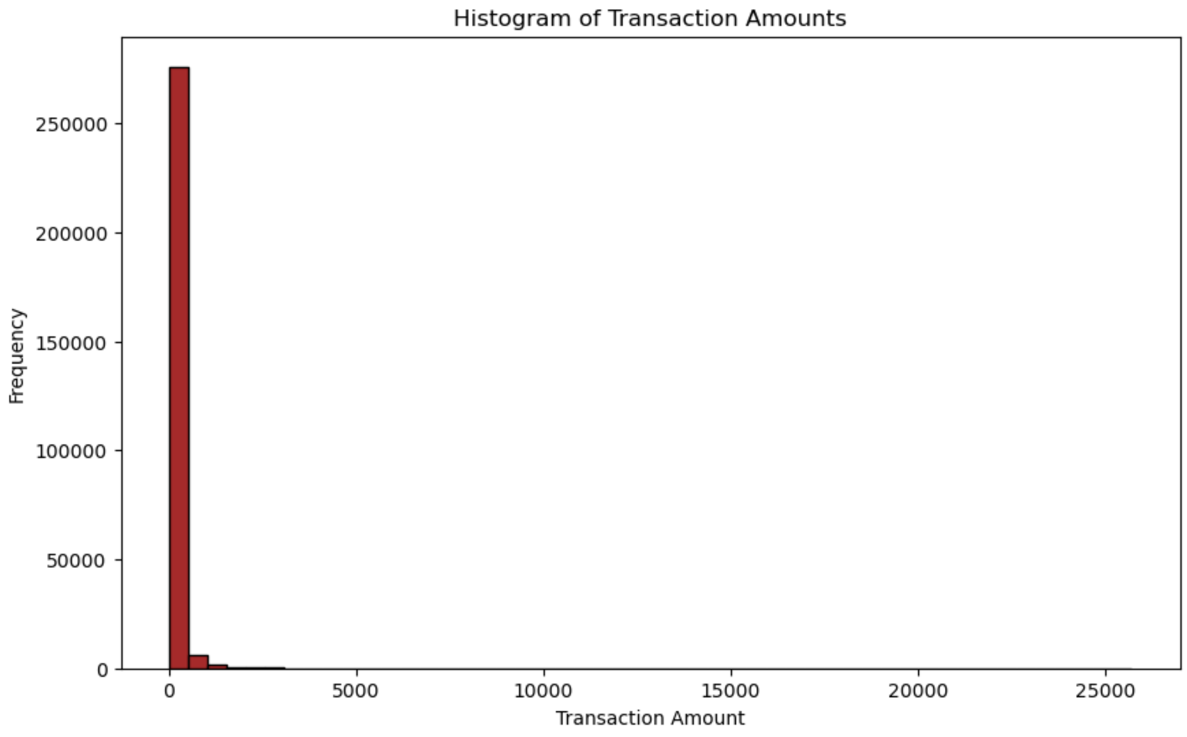
11. Bar chart for fraudulent vs. legitimate transactions

In [56]:
```
plt.figure(figsize=(8, 6))
class_counts.plot(kind='bar', color=['green', 'red'])
plt.title('Count of Fraudulent vs Legitimate Transactions')
plt.xlabel('Class (0 = Legitimate, 1 = Fraudulent)')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```



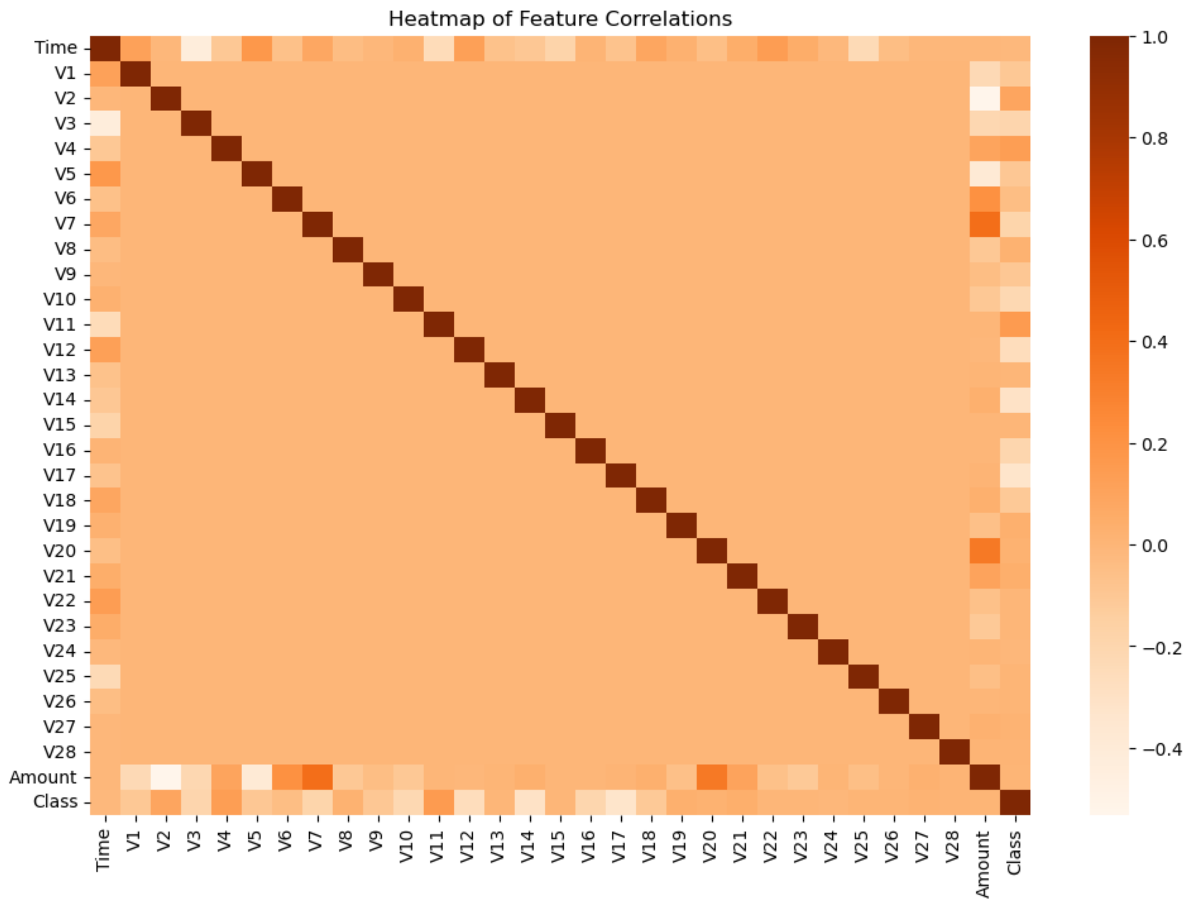
12. Histogram for transaction amounts

In [57]:
```
plt.figure(figsize=(10, 6))
plt.hist(df['Amount'], bins=50, color='brown', edgecolor='black')
plt.title('Histogram of Transaction Amounts')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()
```



13. Heatmap for correlation between numerical features

In [58]:
```
plt.figure(figsize=(12, 8))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, cmap='Oranges', annot=False)
plt.title('Heatmap of Feature Correlations')
plt.show()
```


Heatmap of Feature Correlations
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
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

```
In [ ]:
```