In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [2]:
```
df=pd.read_csv('fraudTrain.csv')
```

In [3]:
```
df
```

Out[3]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	56
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	Su
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 Dal
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	Cou
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	
...	...	...	...	...	...	...	...	...	...	
1296670	1296670	2020-06-21 12:12:08	30263540414123	fraud_Reichel Inc	entertainment	15.56	Erik	Patterson	M	Ro
1296671	1296671	2020-06-21 12:12:19	6011149206456997	fraud_Abernathy and Sons	food_dining	51.70	Jeffrey	White	M	Su
1296672	1296672	2020-06-21 12:12:32	3514865930894695	fraud_Stiedemann Ltd	food_dining	105.93	Christopher	Castaneda	M	Su
1296673	1296673	2020-06-21 12:13:36	2720012583106919	fraud_Reinger, Weissnat and Strosin	food_dining	74.90	Joseph	Murray	M	Unc
1296674	1296674	2020-06-21 12:13:37	4292902571056973207	fraud_Langosh, Wintheiser and Hyatt	food_dining	4.30	Jeffrey	Smith	M	Mou

1296675 rows × 23 columns

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

In [5]:
```
# Separate features and target variable
X = df.drop('is_fraud', axis=1)  # Exclude the target variable
y = df['is_fraud']
```

In [6]:
```
df.drop_duplicates()
```

Out[6]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	56
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	Su
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 Dal
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	Cou
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	
...	...	...	...	...	...	...	...	...	...	
1296670	1296670	2020-06-21 12:12:08	30263540414123	fraud_Reichel Inc	entertainment	15.56	Erik	Patterson	M	Ro
1296671	1296671	2020-06-21 12:12:19	6011149206456997	fraud_Abernathy and Sons	food_dining	51.70	Jeffrey	White	M	Su
1296672	1296672	2020-06-21 12:12:32	3514865930894695	fraud_Stiedemann Ltd	food_dining	105.93	Christopher	Castaneda	M	Su
1296673	1296673	2020-06-21 12:13:36	2720012583106919	fraud_Reinger, Weissnat and Strosin	food_dining	74.90	Joseph	Murray	M	Unc
1296674	1296674	2020-06-21 12:13:37	4292902571056973207	fraud_Langosh, Wintheiser and Hyatt	food_dining	4.30	Jeffrey	Smith	M	Mou

1296675 rows × 23 columns

In [7]:
```
# Drop columns that are not useful for modeling (you may need to adjust this based on your analysis)
columns_to_drop = ['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category', 'first', 'last', 'gender',
X = X.drop(columns=columns_to_drop)
```

In [8]:
```
df
```

Out[8]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	56
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	Su
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 Dal
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	Cou
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	
...	...	...	...	...	...	...	...	...	...	
1296670	1296670	2020-06-21 12:12:08	30263540414123	fraud_Reichel Inc	entertainment	15.56	Erik	Patterson	M	Ro
1296671	1296671	2020-06-21 12:12:19	6011149206456997	fraud_Abernathy and Sons	food_dining	51.70	Jeffrey	White	M	Su
1296672	1296672	2020-06-21 12:12:32	3514865930894695	fraud_Stiedemann Ltd	food_dining	105.93	Christopher	Castaneda	M	Su
1296673	1296673	2020-06-21 12:13:36	2720012583106919	fraud_Reinger, Weissnat and Strosin	food_dining	74.90	Joseph	Murray	M	Unc
1296674	1296674	2020-06-21 12:13:37	4292902571056973207	fraud_Langosh, Wintheiser and Hyatt	food_dining	4.30	Jeffrey	Smith	M	Mou

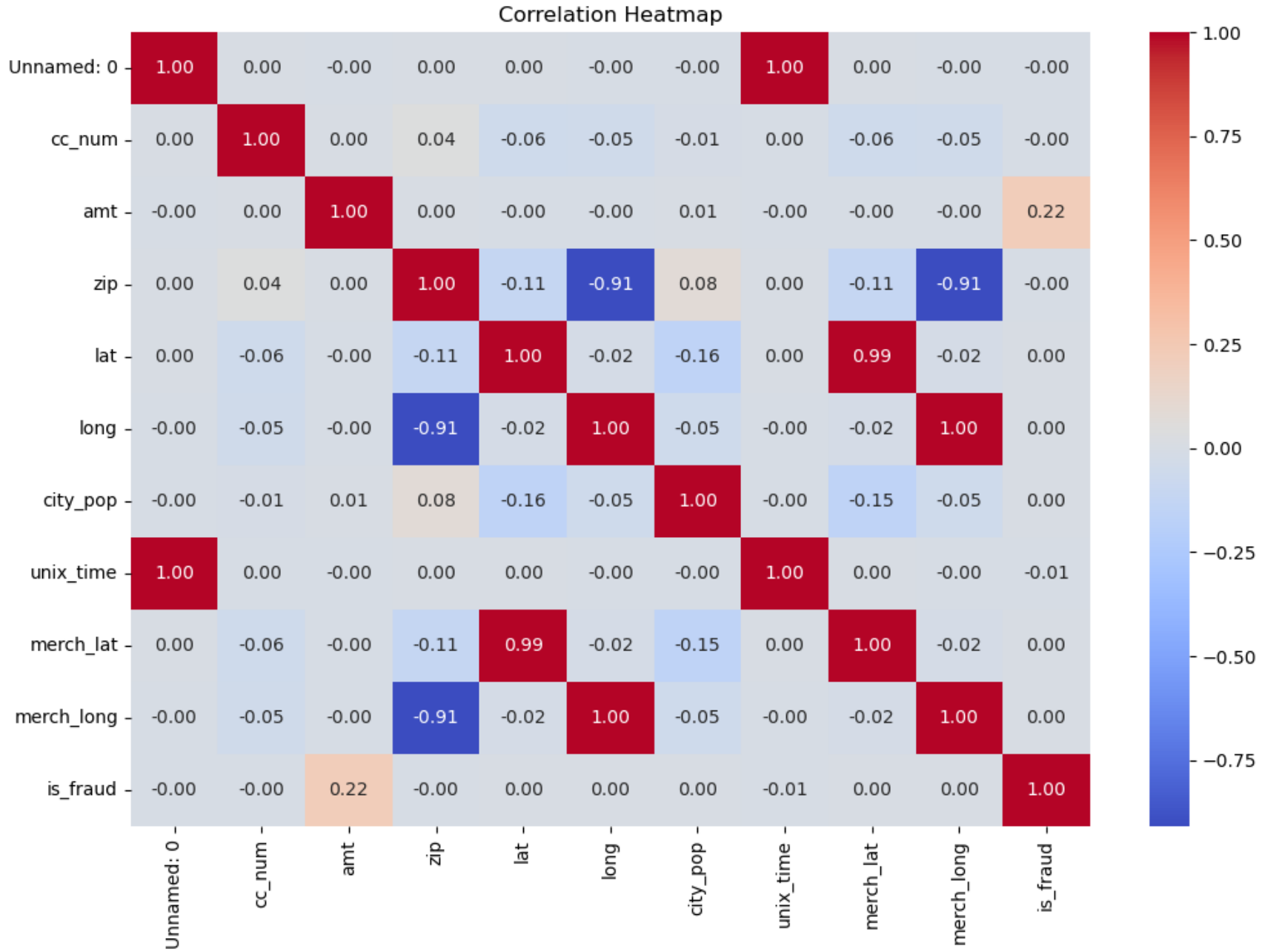
1296675 rows × 23 columns

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

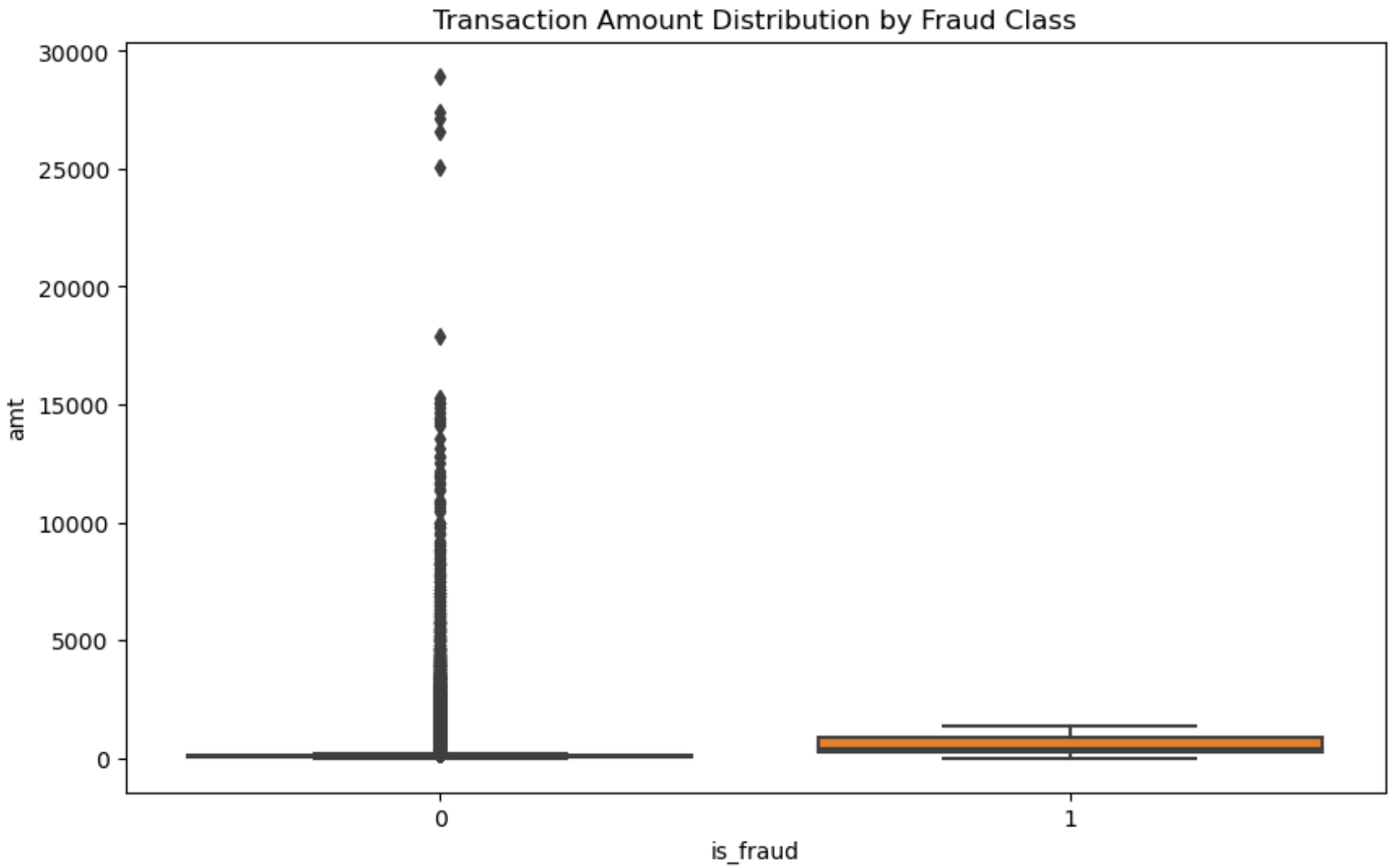
In [10]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [11]:
```
# Visualize the correlation matrix
corr_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```

```
C:\Users\ABHISHEK\AppData\Local\Temp\ipykernel_15952\3038490039.py:2: FutureWarning: The default value of numeric_only
in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify th
e value of numeric_only to silence this warning.
  corr_matrix = df.corr()
```

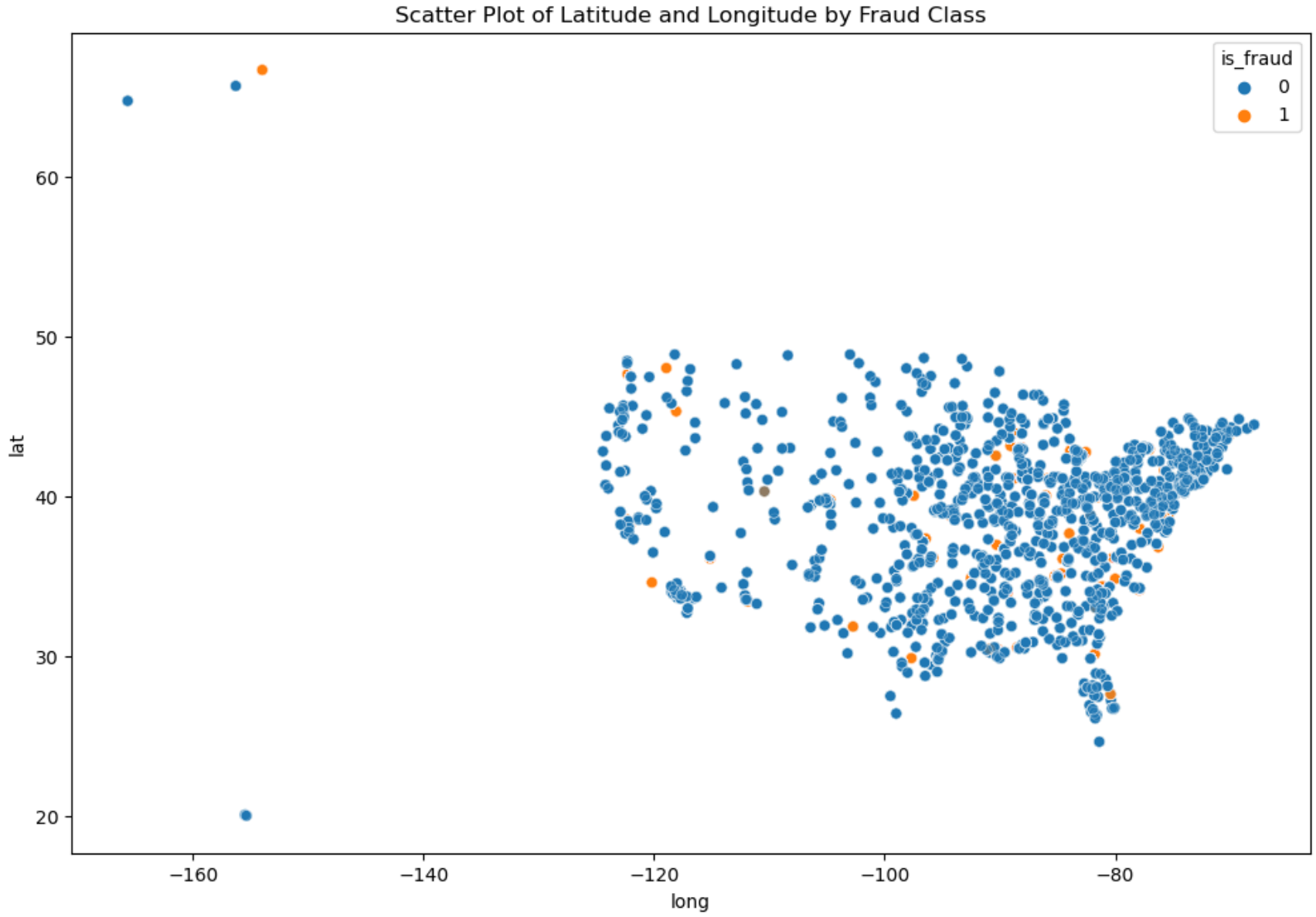


In [12]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='is_fraud', y='amt', data=df)
plt.title('Transaction Amount Distribution by Fraud Class')
plt.show()
```



In [13]:
```
# Violin plot of transaction amount by fraud class
plt.figure(figsize=(10, 6))
sns.violinplot(x='is_fraud', y='amt', data=df)
plt.title('Transaction Amount Distribution by Fraud Class')
plt.show()
```

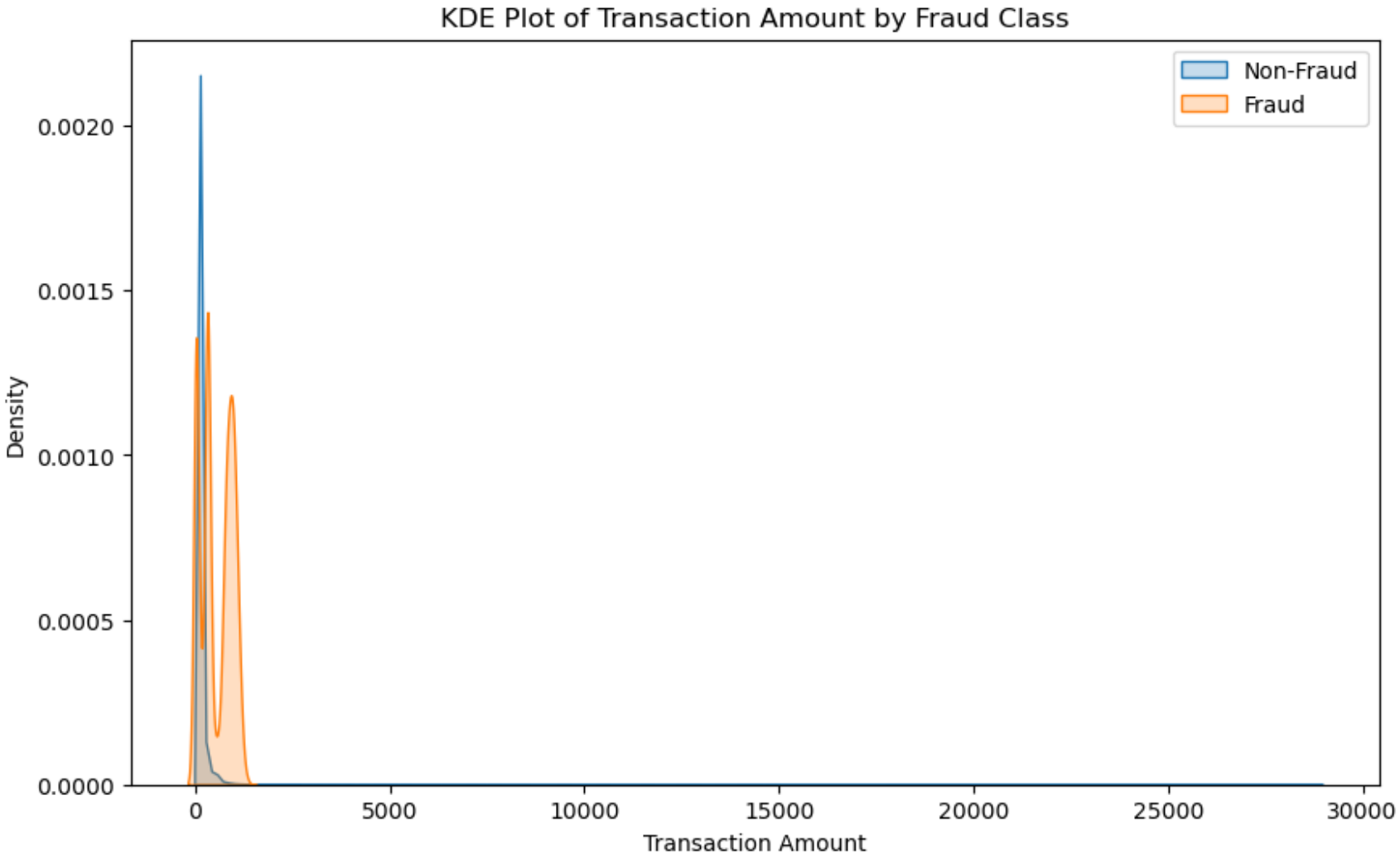
In [14]:
```
# Scatter plot of latitude and longitude for fraud and non-fraud transactions
plt.figure(figsize=(12, 8))
sns.scatterplot(x='long', y='lat', hue='is_fraud', data=df, alpha=0.5)
plt.title('Scatter Plot of Latitude and Longitude by Fraud Class')
plt.show()
```



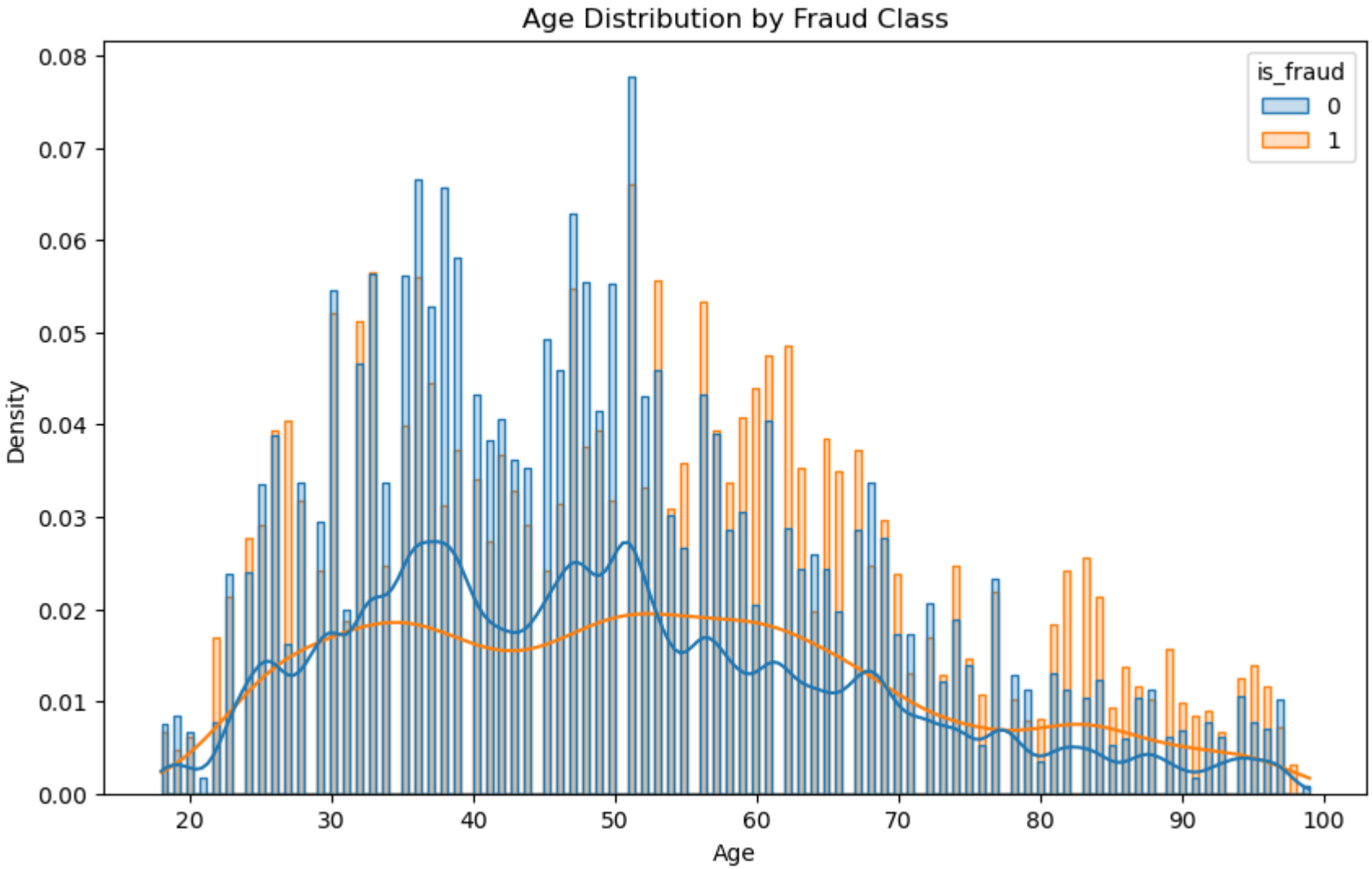
In [15]:
```
# Kernel Density Estimate (KDE) plot for transaction amount by fraud class
plt.figure(figsize=(10, 6))
sns.kdeplot(df[df['is_fraud'] == 0]['amt'], label='Non-Fraud', shade=True)
sns.kdeplot(df[df['is_fraud'] == 1]['amt'], label='Fraud', shade=True)
plt.title('KDE Plot of Transaction Amount by Fraud Class')
plt.xlabel('Transaction Amount')
```

```
plt.legend()
plt.show()
```

```
C:\Users\ABHISHEK\AppData\Local\Temp\ipykernel_15952\3239361695.py:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[df['is_fraud'] == 0]['amt'], label='Non-Fraud', shade=True)
C:\Users\ABHISHEK\AppData\Local\Temp\ipykernel_15952\3239361695.py:4: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(df[df['is_fraud'] == 1]['amt'], label='Fraud', shade=True)
```

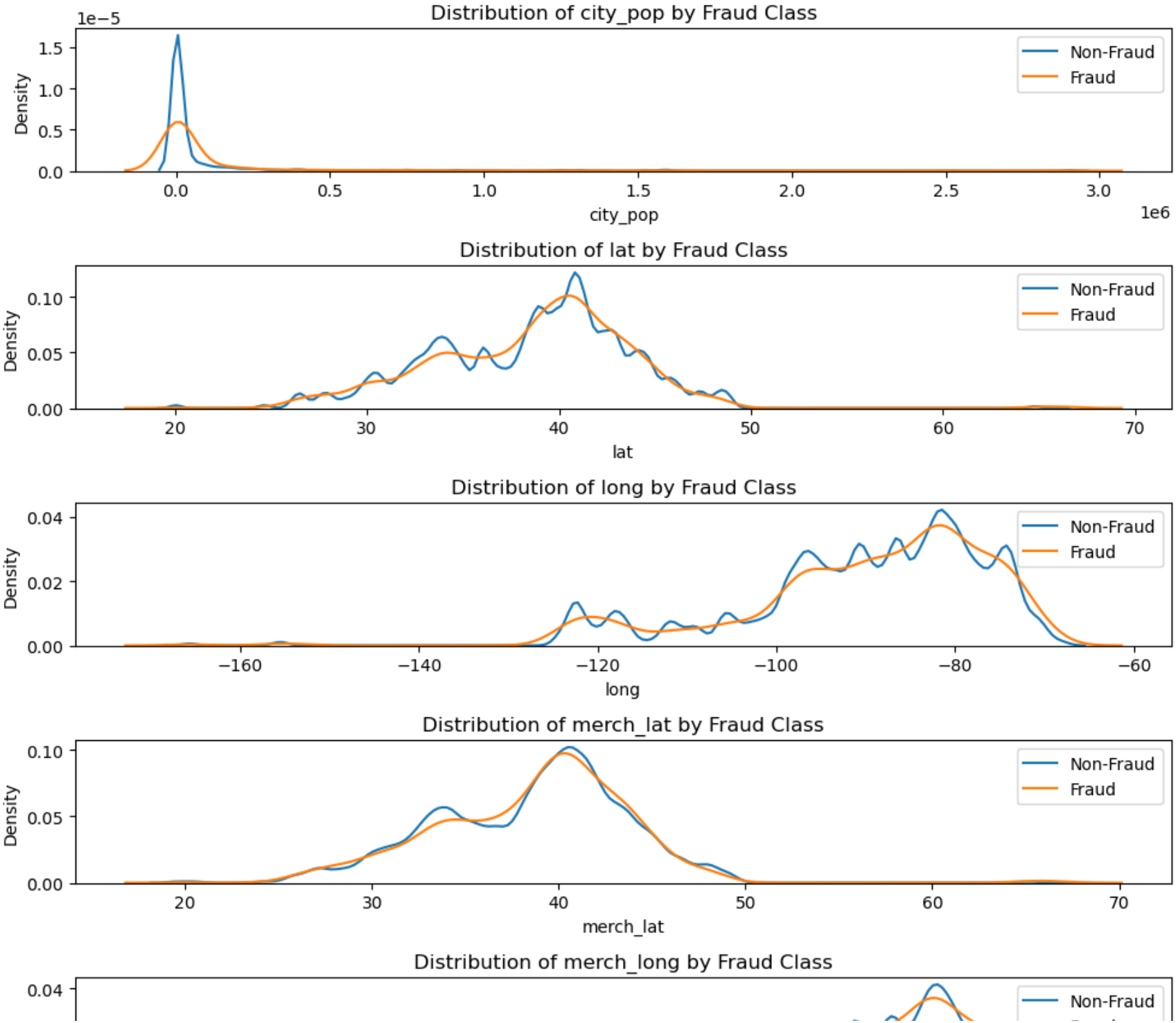


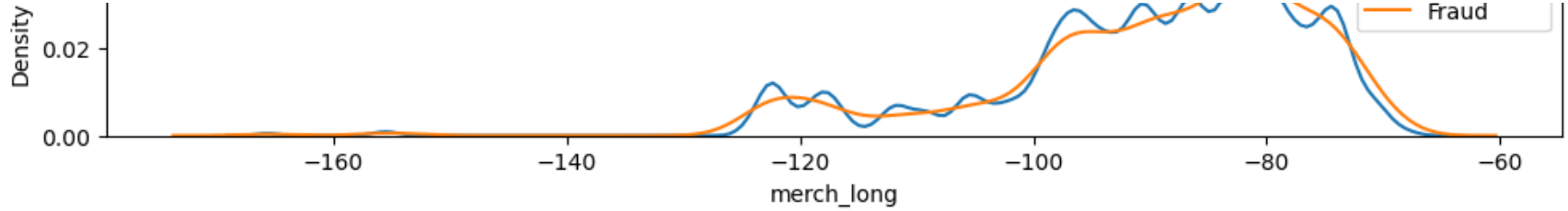
In [16]:
```python
# Convert 'dob' to datetime and calculate age
df['dob'] = pd.to_datetime(df['dob'])
df['age'] = (pd.to_datetime('today') - df['dob']).astype('<m8[Y]')

# Plot the distribution of age by fraud class
plt.figure(figsize=(10, 6))
sns.histplot(df, x='age', hue='is_fraud', element='step', common_norm=False, stat='density', kde=True)
plt.title('Age Distribution by Fraud Class')
plt.xlabel('Age')
plt.show()
```



In [17]:
```
# Plot the distribution of selected features by fraud class
selected_features = ['city_pop', 'lat', 'long', 'merch_lat', 'merch_long']

fig, axes = plt.subplots(nrows=len(selected_features), ncols=1, figsize=(10, 2 * len(selected_features)))

for i, feature in enumerate(selected_features):
    sns.kdeplot(df[df['is_fraud'] == 0][feature], label='Non-Fraud', ax=axes[i])
    sns.kdeplot(df[df['is_fraud'] == 1][feature], label='Fraud', ax=axes[i])
    axes[i].set_title(f'Distribution of {feature} by Fraud Class')
    axes[i].legend()
```

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

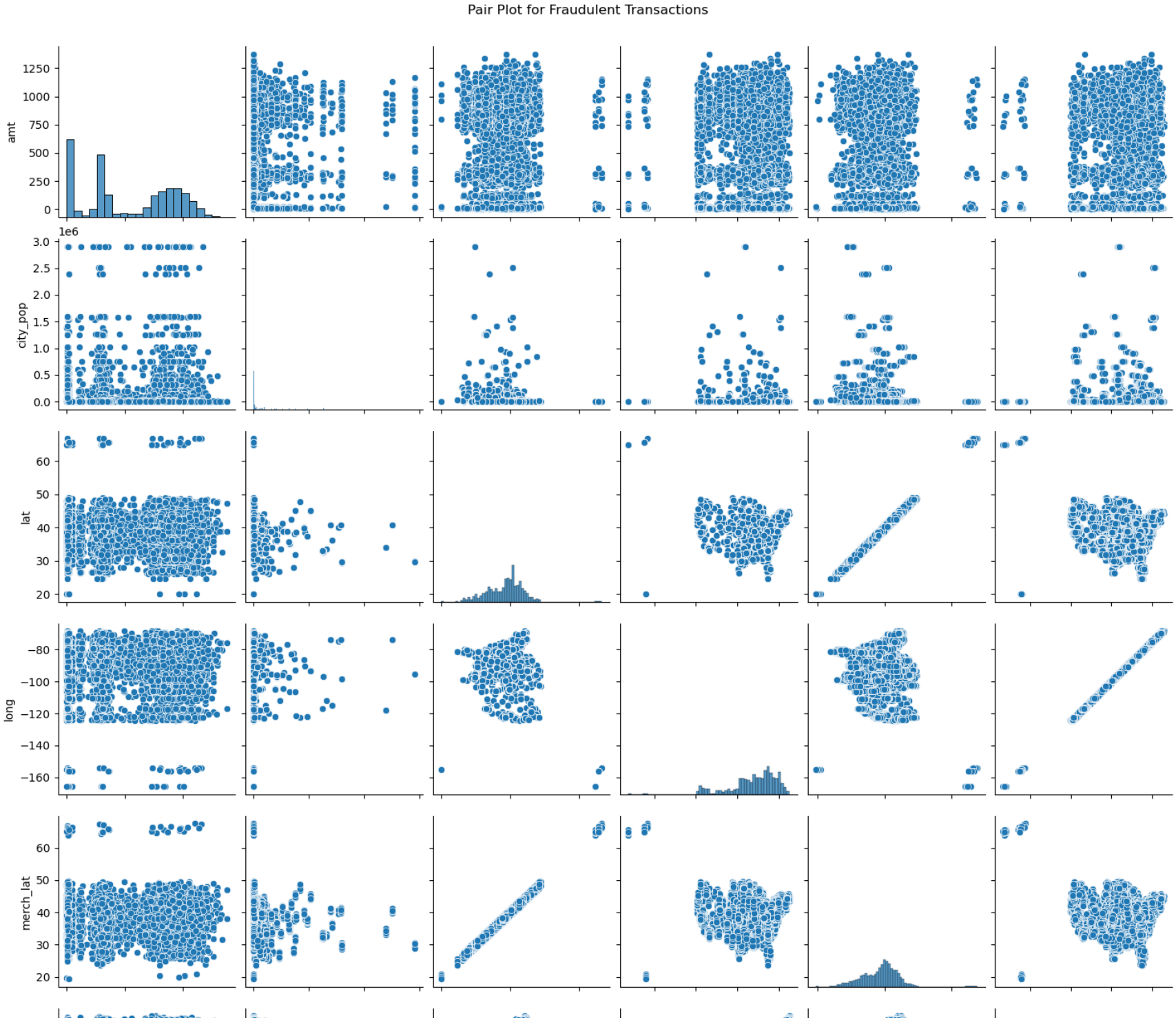

Distribution of city_pop by Fraud Class
1e−5
Density
city_pop
1e6
Non-Fraud
Fraud
Distribution of lat by Fraud Class
Density
lat
Non-Fraud
Fraud
Distribution of long by Fraud Class
Density
long
Non-Fraud
Fraud
Distribution of merch_lat by Fraud Class
Density
merch_lat
Non-Fraud
Fraud
Distribution of merch_long by Fraud Class
Non-Fraud

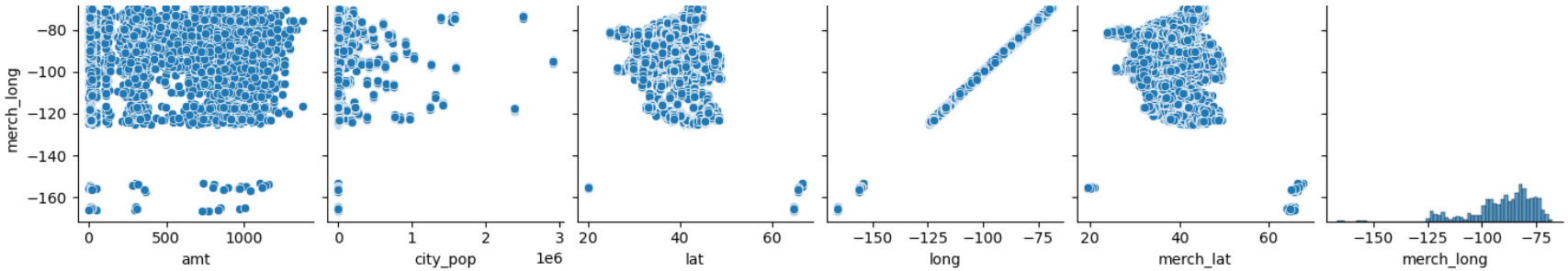


In [18]:
```python
from imblearn.over_sampling import SMOTE
from collections import Counter

# Use SMOTE to resample the data
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check the distribution after resampling
print(f"Original class distribution: {Counter(y)}")
print(f"Resampled class distribution: {Counter(y_resampled)}")
```

```
Original class distribution: Counter({0: 1289169, 1: 7506})
Resampled class distribution: Counter({0: 1289169, 1: 1289169})
```

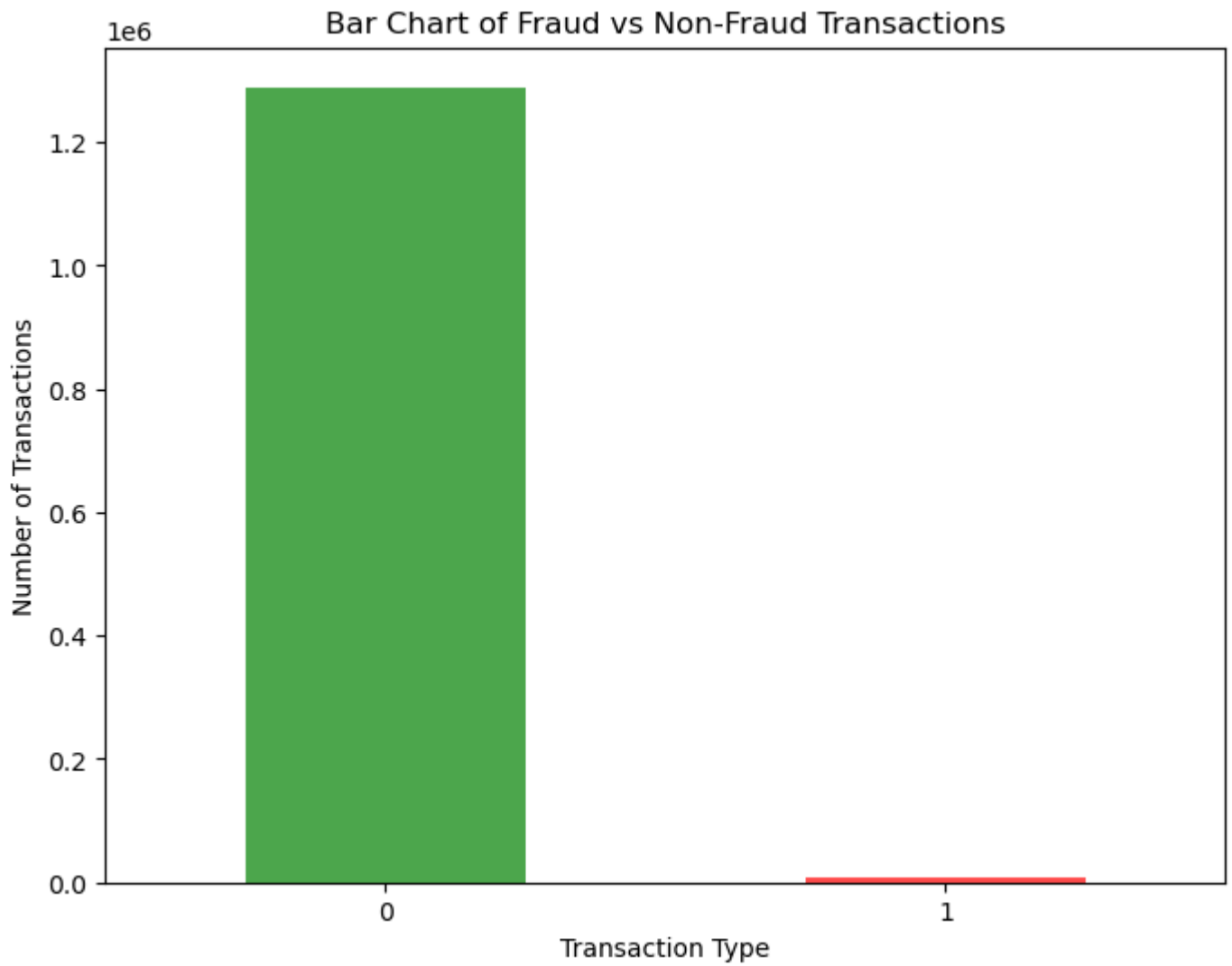
In [19]:
```python
# Create a pair plot for selected features, focusing on fraudulent transactions
fraud_data = df[df['is_fraud'] == 1]
selected_features = ['amt', 'city_pop', 'lat', 'long', 'merch_lat', 'merch_long']

sns.pairplot(fraud_data[selected_features])
plt.suptitle('Pair Plot for Fraudulent Transactions', y=1.02)
plt.show()
```


Pair Plot for Fraudulent Transactions
amt
city_pop
1e6
lat
long
merch_lat



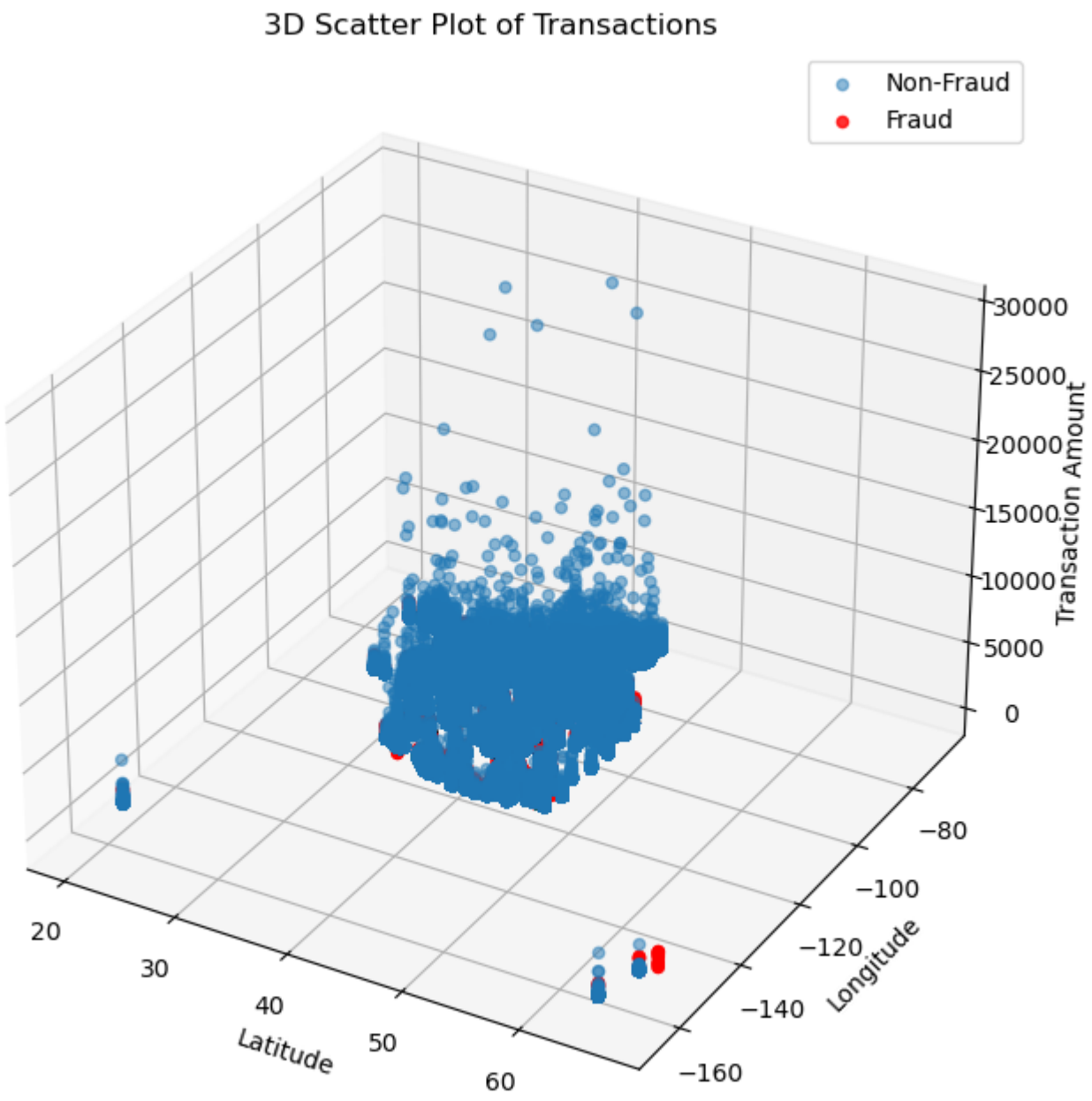
In [20]:

```
# Creating a bar chart for the count of fraud and non-fraud transactions
plt.figure(figsize=(8, 6))
df['is_fraud'].value_counts().plot(kind='bar', color=['green', 'red'], alpha=0.7)
plt.title('Bar Chart of Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('Number of Transactions')
plt.xticks(rotation=0)
plt.show()
```



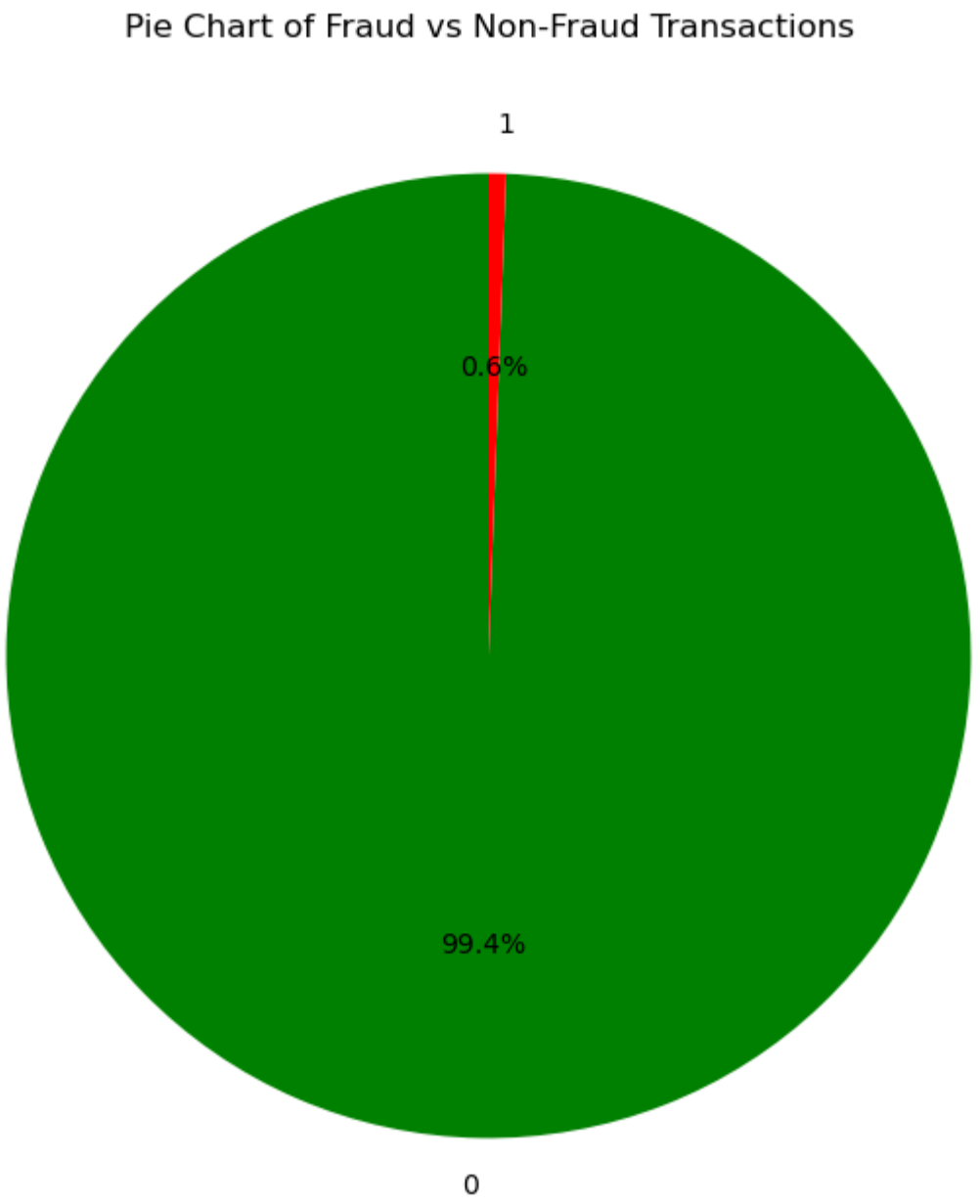
In [21]:
```python
from mpl_toolkits.mplot3d import Axes3D

# Creating a 3D scatter plot for 'lat', 'long', and 'amt'
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

fraud_data = df[df['is_fraud'] == 1]
non_fraud_data = df[df['is_fraud'] == 0]

ax.scatter(non_fraud_data['lat'], non_fraud_data['long'], non_fraud_data['amt'], label='Non-Fraud', alpha=0.5)
```

```
ax.scatter(fraud_data['lat'], fraud_data['long'], fraud_data['amt'], label='Fraud', alpha=0.8, color='red')

ax.set_xlabel('Latitude')
ax.set_ylabel('Longitude')
ax.set_zlabel('Transaction Amount')
ax.set_title('3D Scatter Plot of Transactions')
plt.legend()
plt.show()
```



In [22]:
```
# Creating a pie chart for the distribution of fraud and non-fraud transactions
plt.figure(figsize=(8, 8))
fraud_counts = df['is_fraud'].value_counts()
```

```
plt.pie(fraud_counts, labels=fraud_counts.index, autopct='%1.1f%%', colors=['green', 'red'], startangle=90)
plt.title('Pie Chart of Fraud vs Non-Fraud Transactions')
plt.show()
```



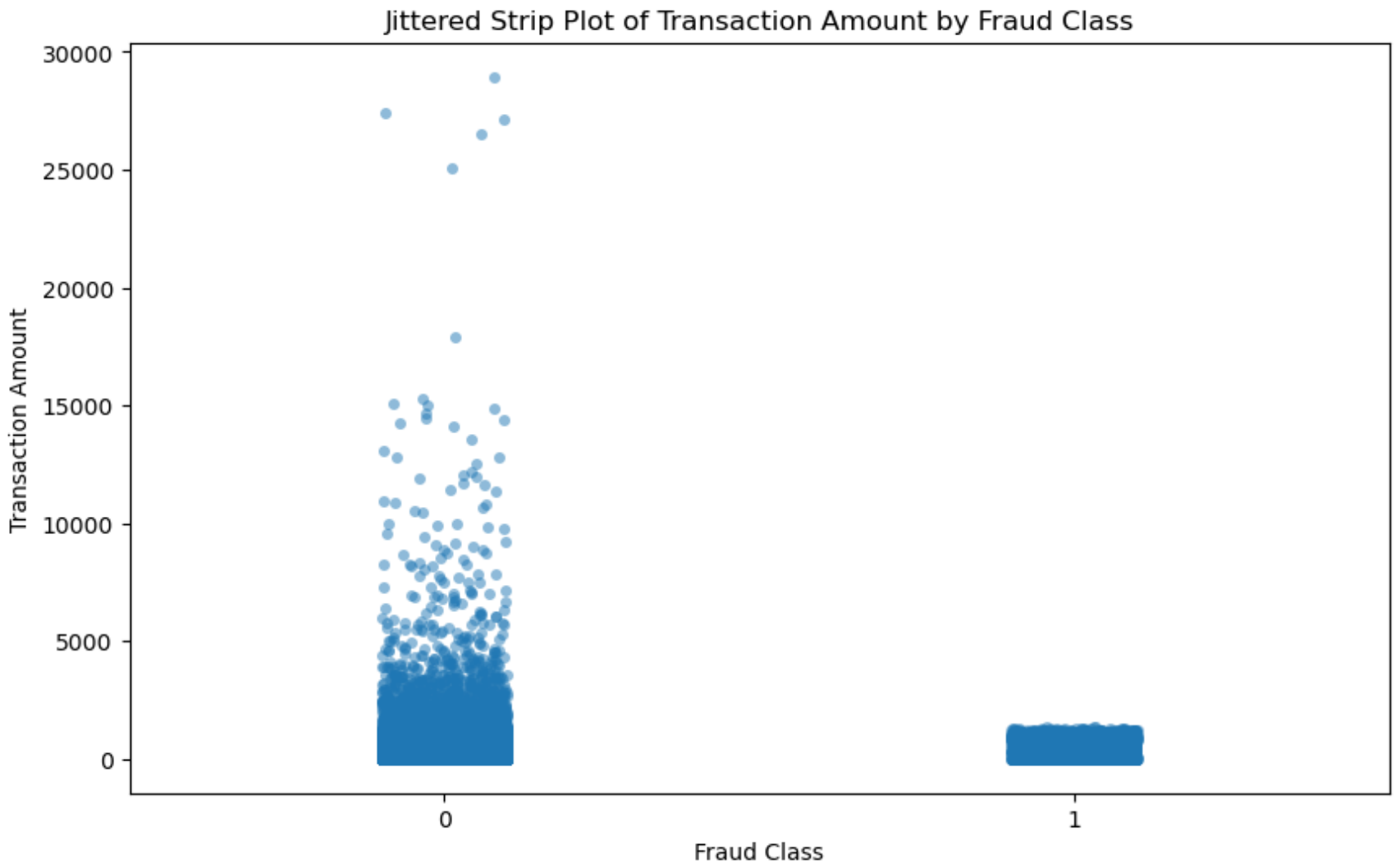
In [23]:

```python
from wordcloud import WordCloud

# Creating a word cloud for the 'merchant' column
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '.join(df['merchant']))

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Merchants')
plt.show()
```

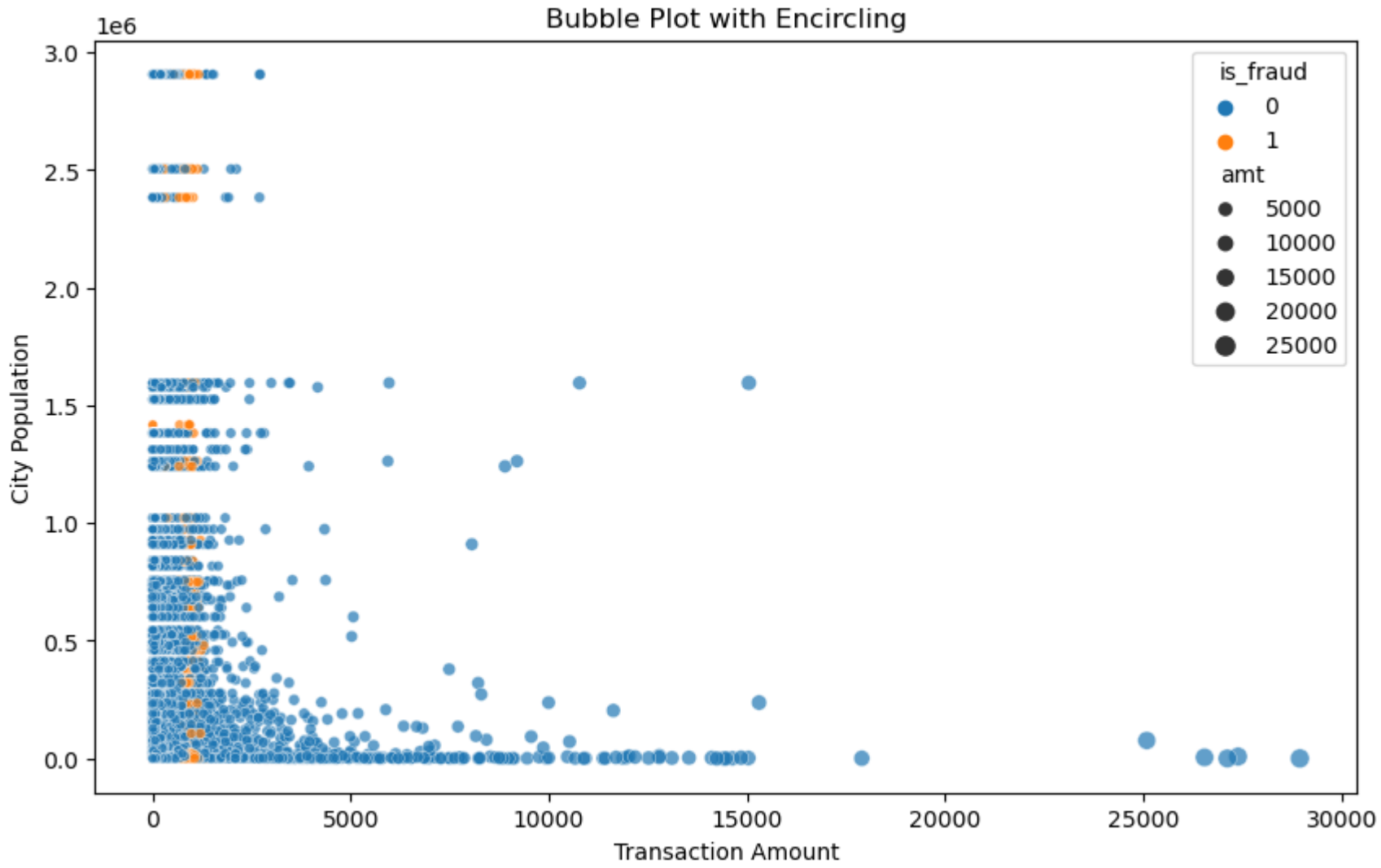
In [24]:

```python
# Jittered strip plot for 'amt' and 'is_fraud'
plt.figure(figsize=(10, 6))
sns.stripplot(x='is_fraud', y='amt', data=df, jitter=True, alpha=0.5)
plt.title('Jittered Strip Plot of Transaction Amount by Fraud Class')
plt.xlabel('Fraud Class')
```

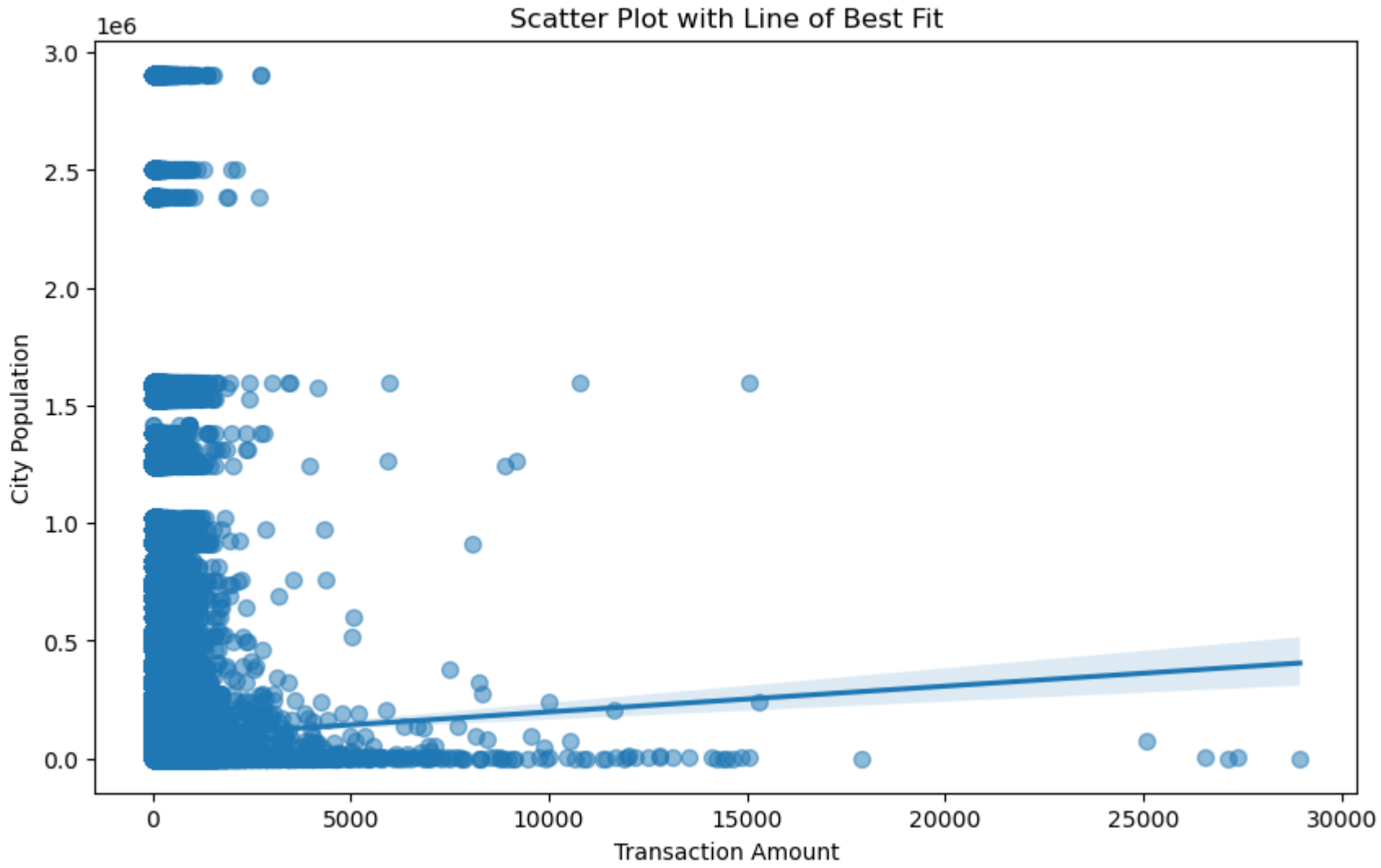
```
plt.ylabel('Transaction Amount')
plt.show()
```



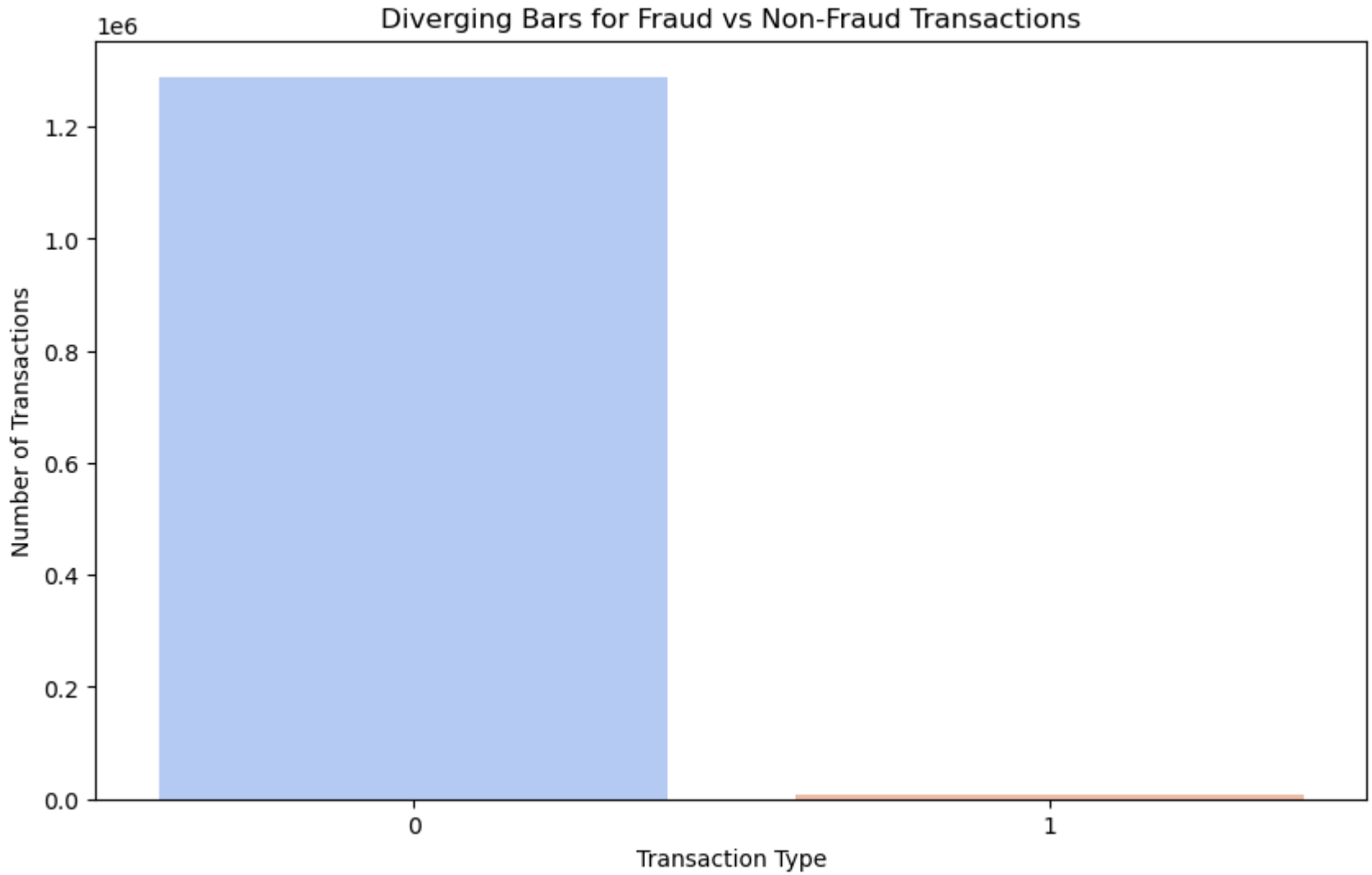
In [25]:
```
# Bubble plot with encircling for 'amt' and 'is_fraud'
plt.figure(figsize=(10, 6))
sns.scatterplot(x='amt', y='city_pop', hue='is_fraud', size='amt', data=df, alpha=0.7)
plt.title('Bubble Plot with Encircling')
plt.xlabel('Transaction Amount')
plt.ylabel('City Population')
plt.show()
```



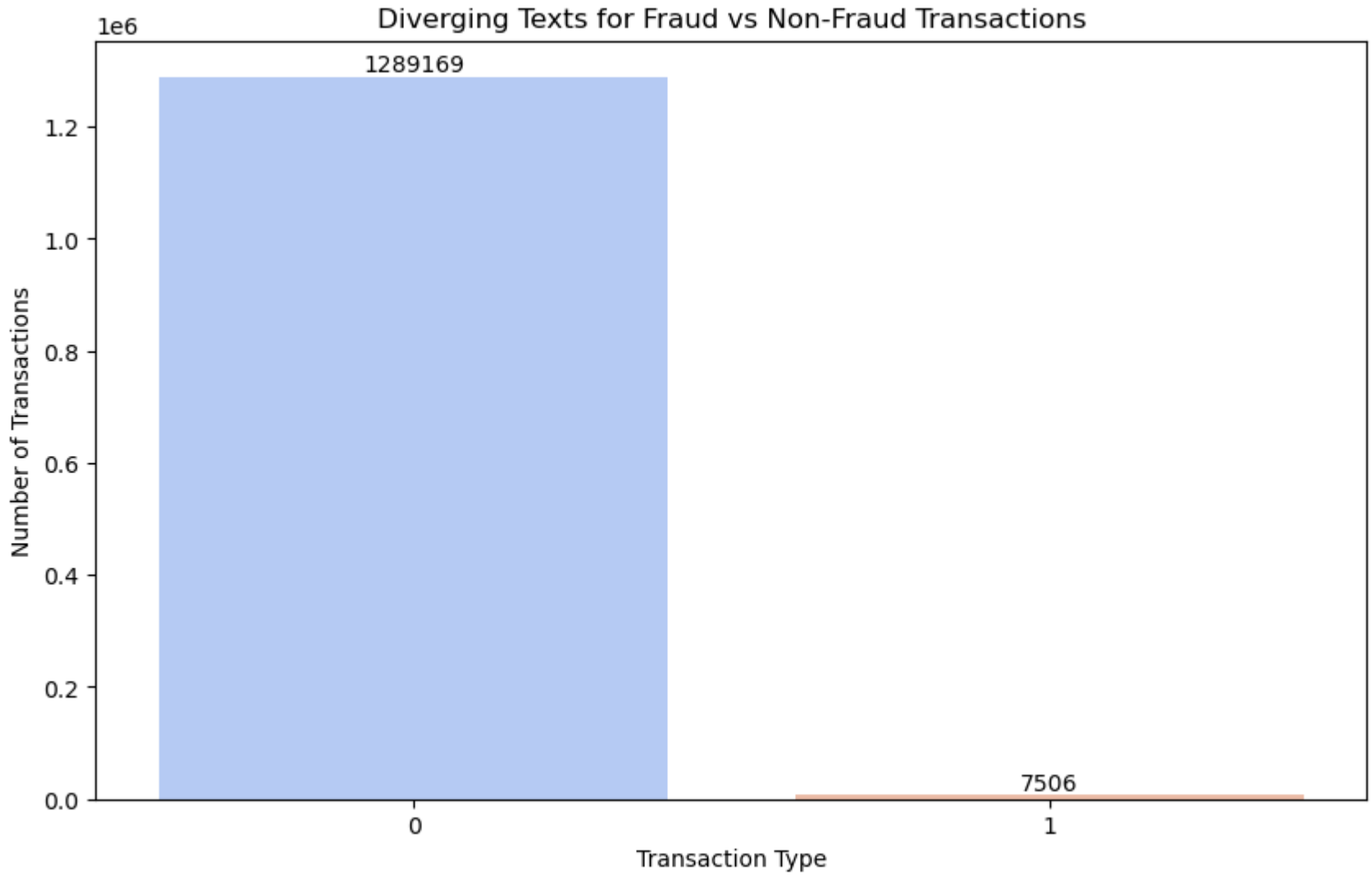
In [26]:
```
# Scatter plot with line of best fit for 'amt' and 'city_pop'
plt.figure(figsize=(10, 6))
sns.regplot(x='amt', y='city_pop', data=df, scatter_kws={'s': 50, 'alpha': 0.5})
plt.title('Scatter Plot with Line of Best Fit')
plt.xlabel('Transaction Amount')
plt.ylabel('City Population')
plt.show()
```



In [27]:
```
# Diverging bars for the count of transactions
plt.figure(figsize=(10, 6))
sns.barplot(x=df['is_fraud'].value_counts().index, y=df['is_fraud'].value_counts(), palette='coolwarm')
plt.title('Diverging Bars for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('Number of Transactions')
plt.show()
```



In [28]:
```
# Diverging texts for the count of transactions
plt.figure(figsize=(10, 6))
sns.barplot(x=df['is_fraud'].value_counts().index, y=df['is_fraud'].value_counts(), palette='coolwarm')
for i, value in enumerate(df['is_fraud'].value_counts()):
    plt.text(i, value + 0.1, str(value), ha='center', va='bottom')
plt.title('Diverging Texts for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('Number of Transactions')
plt.show()
```

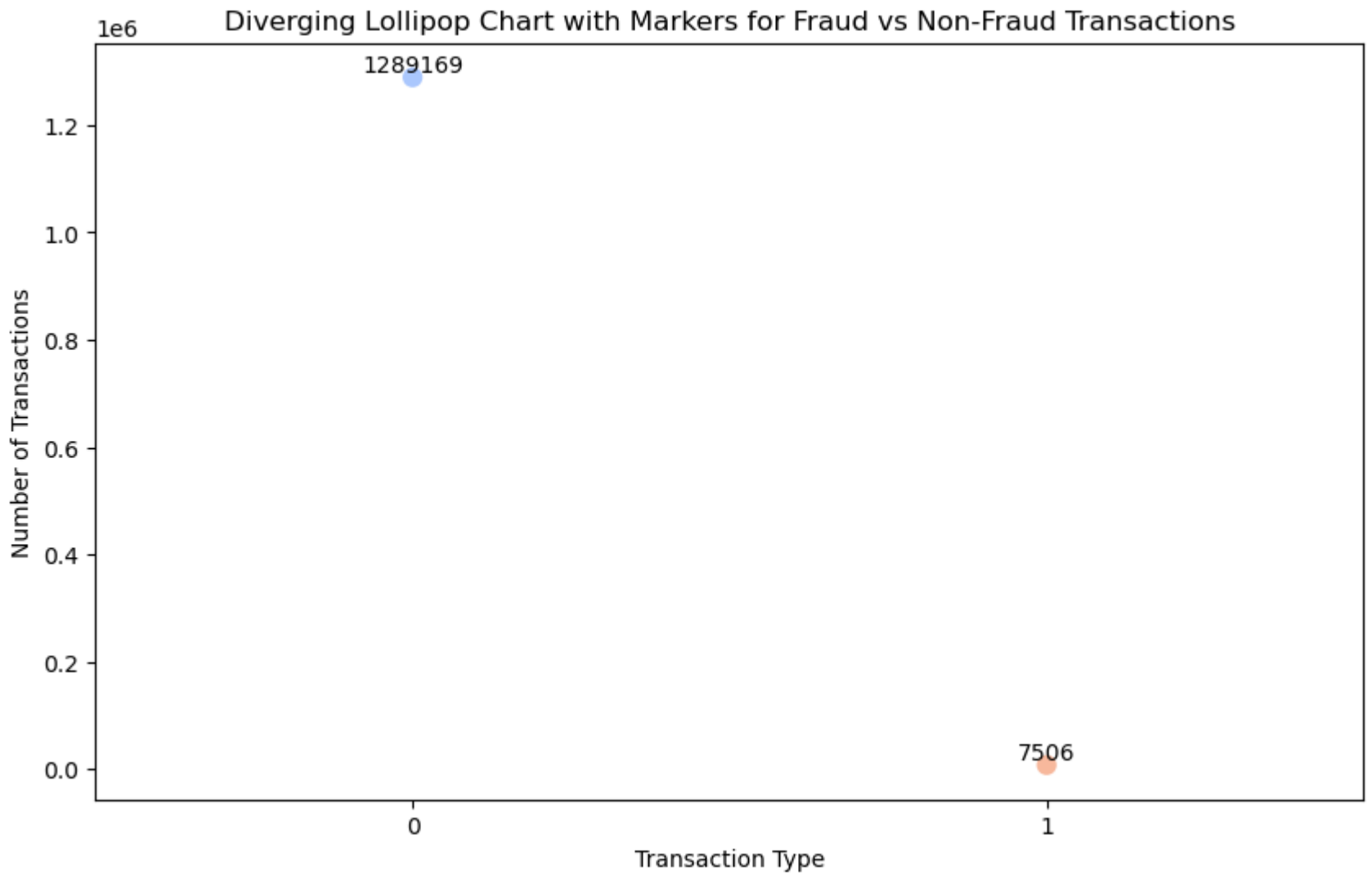


In [29]:
```
# Diverging dot plot for the count of transactions
plt.figure(figsize=(10, 6))
sns.stripplot(x=df['is_fraud'], y=df['is_fraud'], jitter=True, size=10, palette='coolwarm', alpha=0.7)
plt.title('Diverging Dot Plot for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('')
plt.show()
```

```
C:\Users\ABHISHEK\AppData\Local\Temp\ipykernel_15952\2453256856.py:3: FutureWarning: Passing `palette` without assigning `hue` is deprecated.
  sns.stripplot(x=df['is_fraud'], y=df['is_fraud'], jitter=True, size=10, palette='coolwarm', alpha=0.7)
```

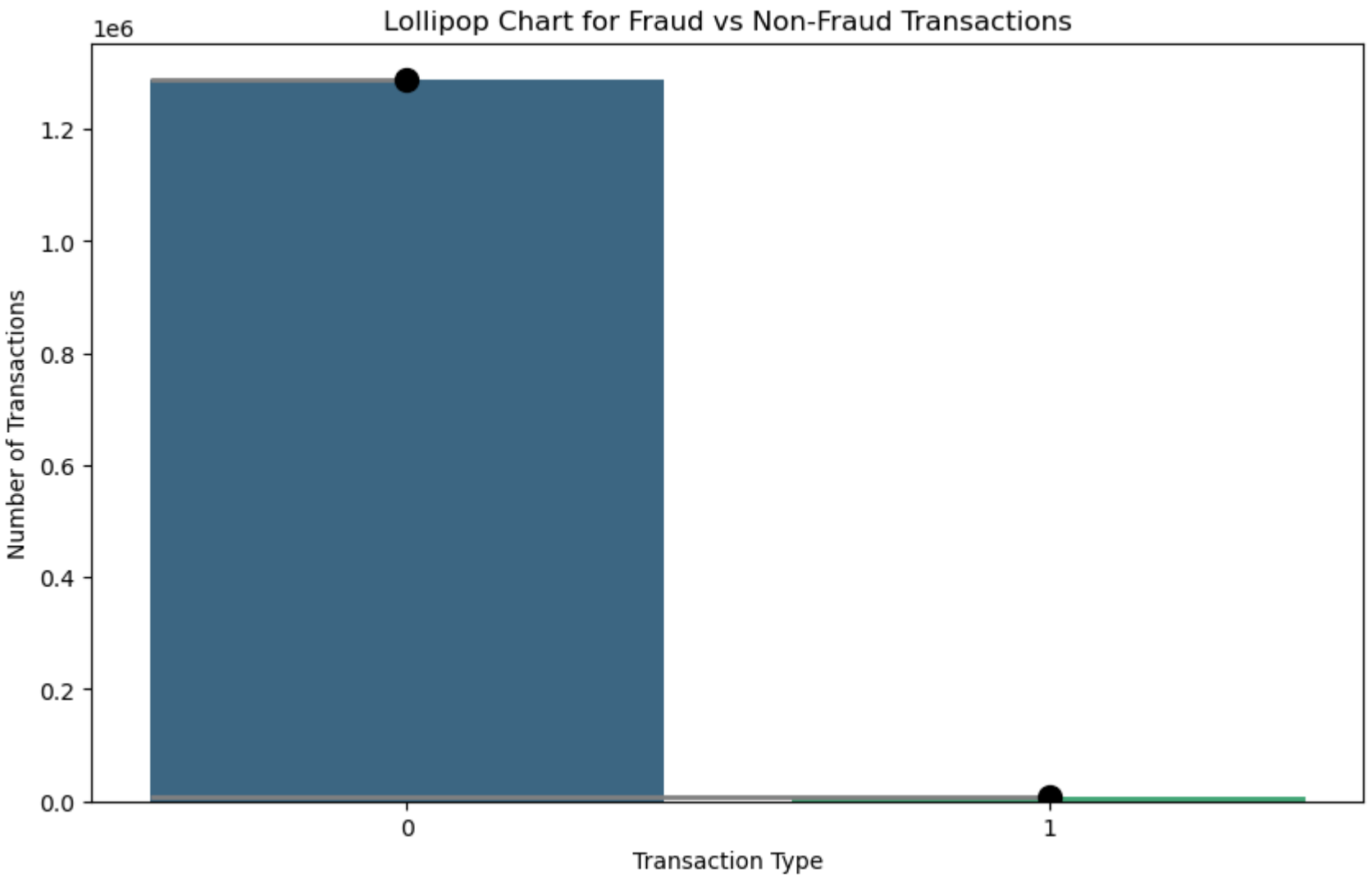
In [30]:
```python
# Diverging lollipop chart with markers for the count of transactions
plt.figure(figsize=(10, 6))
sns.pointplot(x=df['is_fraud'].value_counts().index, y=df['is_fraud'].value_counts(), palette='coolwarm')
for i, value in enumerate(df['is_fraud'].value_counts()):
    plt.text(i, value + 0.1, str(value), ha='center', va='bottom')
plt.title('Diverging Lollipop Chart with Markers for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
```

```
plt.ylabel('Number of Transactions')
plt.show()
```



In [31]:
```
# Lollipop chart for the count of transactions
plt.figure(figsize=(10, 6))
sns.barplot(x=df['is_fraud'].value_counts().index, y=df['is_fraud'].value_counts(), palette='viridis')
for i, value in enumerate(df['is_fraud'].value_counts()):
    plt.hlines(value, -0.4, i, color='gray', linewidth=2)
    plt.scatter(i, value, color='black', s=100, zorder=10)
plt.title('Lollipop Chart for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
```

```
plt.ylabel('Number of Transactions')
plt.show()
```



In [32]:
```
# Dot plot for the count of transactions
plt.figure(figsize=(10, 6))
sns.stripplot(x=df['is_fraud'], y=df['is_fraud'], jitter=True, size=10, palette='viridis', alpha=0.7)
plt.title('Dot Plot for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('')
plt.show()
```

```
C:\Users\ABHISHEK\AppData\Local\Temp\ipykernel_15952\1551883796.py:3: FutureWarning: Passing `palette` without assigning `hue` is deprecated.
  sns.stripplot(x=df['is_fraud'], y=df['is_fraud'], jitter=True, size=10, palette='viridis', alpha=0.7)
```

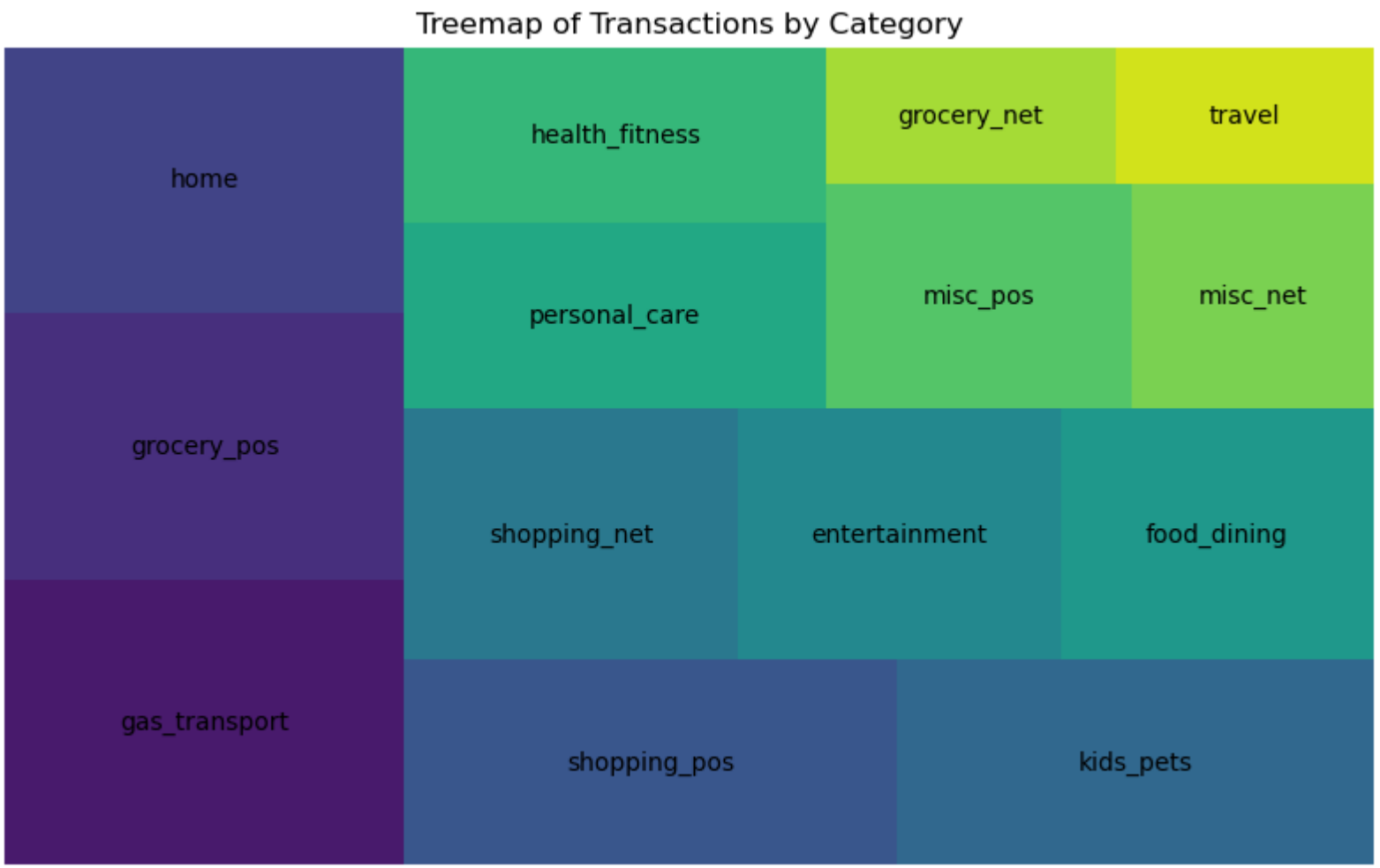
In [33]:
```python
# Slope chart for the count of transactions
fraud_counts = df['is_fraud'].value_counts()

plt.figure(figsize=(10, 6))
plt.plot(fraud_counts.index, fraud_counts, marker='o', color='orange')
plt.title('Slope Chart for Fraud vs Non-Fraud Transactions')
plt.xlabel('Transaction Type')
```

```
plt.ylabel('Number of Transactions')
plt.show()
```

In [ ]:

In [34]:
```
import squarify

# Treemap for the count of transactions by category
plt.figure(figsize=(10, 6))
squarify.plot(sizes=df['category'].value_counts(), label=df['category'].value_counts().index, color=sns.color_palette('
```

```
plt.title('Treemap of Transactions by Category')
plt.axis('off')
plt.show()
```



In [35]:
```
from sklearn.model_selection import train_test_split

# Assuming 'X' is your feature matrix and 'y' is your target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [36]:
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Out[36]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [37]:
```
y_pred = model.predict(X_test)
```

In [38]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9950835791543756
```

In [39]:
```
from sklearn.metrics import classification_report, confusion_matrix

# Print classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    257815
           1       0.66      0.33      0.44      1520

    accuracy                           1.00    259335
   macro avg       0.83      0.66      0.72    259335
weighted avg       0.99      1.00      0.99    259335

[[257564    251]
 [  1024    496]]
```

In [40]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
```

In [41]:
```
from sklearn.svm import SVC

model = SVC()
```

In [42]:
```
model
```

Out[42]:

```
SVC()
```

In [45]:

```
from xgboost import XGBClassifier

# Create and train the XGBoost model
model_xgb = XGBClassifier()
model_xgb.fit(X_train, y_train)
```

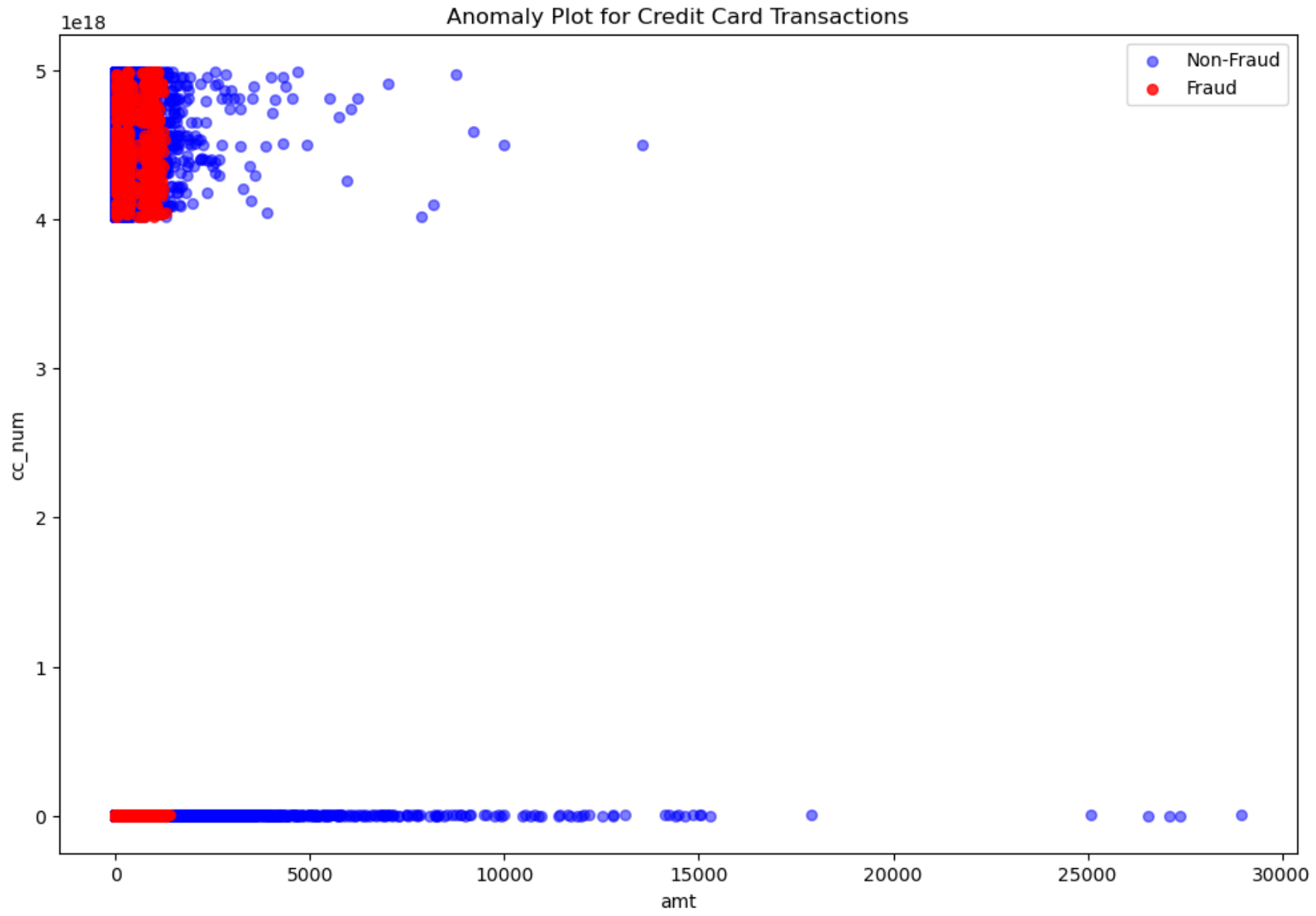
Out[45]:

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

In [50]:

```
features_for_plot = ['amt', 'cc_num']

# Filter the data for fraud and non-fraud transactions
fraud_data = df[df['is_fraud'] == 1]
non_fraud_data = df[df['is_fraud'] == 0]

# Create a scatter plot for non-fraud transactions
plt.figure(figsize=(12, 8))
plt.scatter(non_fraud_data[features_for_plot[0]], non_fraud_data[features_for_plot[1]], label='Non-Fraud', alpha=0.5, s

# Create a scatter plot for fraud transactions (highlighted in red)
plt.scatter(fraud_data[features_for_plot[0]], fraud_data[features_for_plot[1]], label='Fraud', alpha=0.8, s=30, c='red'

# Customize plot
plt.title('Anomaly Plot for Credit Card Transactions')
plt.xlabel(features_for_plot[0])
plt.ylabel(features_for_plot[1])
```

```
plt.legend()
plt.show()
```



In [ ]: