In [2]:
```
## Credit Card Fraud Detection - Data Analysis ##
```

In [4]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

In [8]:
```python
# Load the dataset
df = pd.read_csv('credit_card_fraud_dataset.csv')
df
```

Out[8]:

	TransactionID	TransactionDate	Amount	MerchantID	TransactionType	Location	IsFraud
0	1	2024-04-03 14:15:35.462794	4189.27	688	refund	San Antonio	0
1	2	2024-03-19 13:20:35.462824	2659.71	109	refund	Dallas	0
2	3	2024-01-08 10:08:35.462834	784.00	394	purchase	New York	0
3	4	2024-04-13 23:50:35.462850	3514.40	944	purchase	Philadelphia	0
4	5	2024-07-12 18:51:35.462858	369.07	475	purchase	Phoenix	0
...	...	...	...	...	...	...	...
99995	99996	2024-06-07 00:57:36.027591	1057.29	289	refund	San Antonio	0
99996	99997	2023-10-22 23:12:36.027594	297.25	745	refund	San Antonio	0
99997	99998	2024-05-31 19:27:36.027597	3448.56	690	purchase	San Antonio	0
99998	99999	2024-10-18 09:43:36.027601	3750.79	644	purchase	Philadelphia	0
99999	100000	2024-03-05 19:41:36.027606	1596.79	675	refund	Houston	0

100000 rows × 7 columns

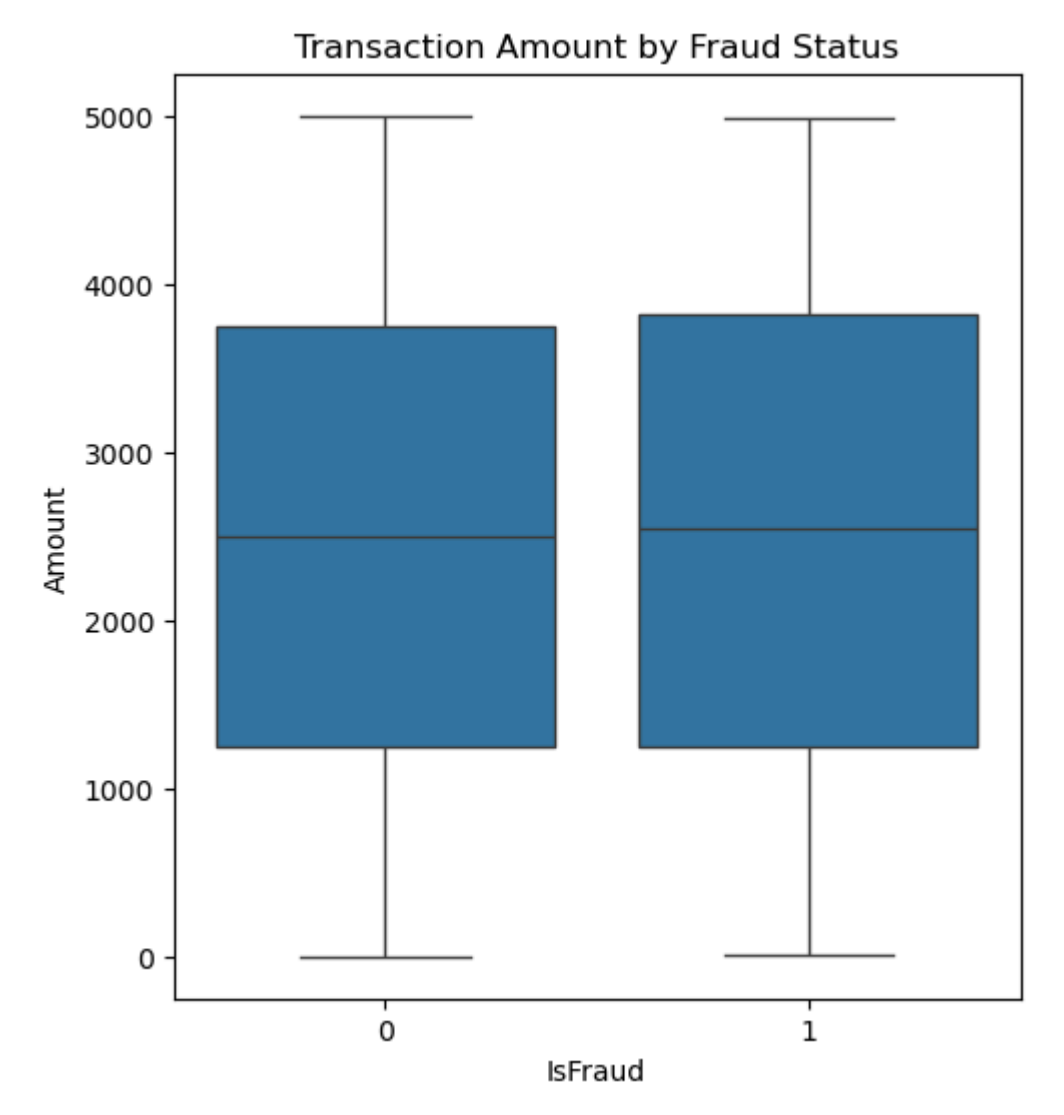
In [10]:
```python
# Basic data exploration
print("Dataset Overview:")
print(f"Number of transactions: {len(df)}")
print(f"Number of fraudulent transactions: {df['IsFraud'].sum()} ({df['IsFraud'].mean()*100:.2f}% of total)")
print("\nFirst 5 rows:")
print(df.head())
```

```
Dataset Overview:
Number of transactions: 100000
Number of fraudulent transactions: 1000 (1.00% of total)

First 5 rows:
   TransactionID             TransactionDate   Amount  MerchantID  \
0              1  2024-04-03 14:15:35.462794  4189.27         688
1              2  2024-03-19 13:20:35.462824  2659.71         109
2              3  2024-01-08 10:08:35.462834   784.00         394
3              4  2024-04-13 23:50:35.462850  3514.40         944
4              5  2024-07-12 18:51:35.462858   369.07         475

  TransactionType      Location  IsFraud
0          refund   San Antonio        0
1          refund        Dallas        0
2        purchase      New York        0
3        purchase  Philadelphia        0
4        purchase       Phoenix        0
```

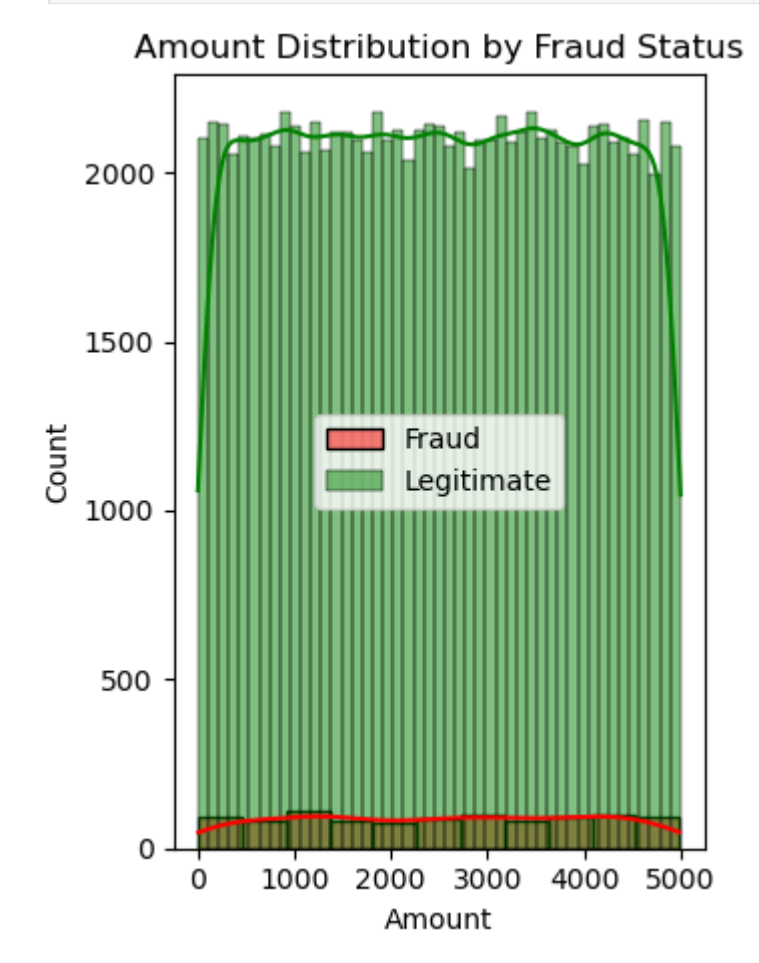
In [12]:
```python
# Data cleaning and preparation
# Convert TransactionDate to datetime
df['TransactionDate'] = pd.to_datetime(df['TransactionDate'])
df['Hour'] = df['TransactionDate'].dt.hour
df['DayOfWeek'] = df['TransactionDate'].dt.dayofweek
df['Month'] = df['TransactionDate'].dt.month
```

In [14]:
```python
# 1. Transaction Amount Analysis
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.boxplot(x='IsFraud', y='Amount', data=df)
plt.title('Transaction Amount by Fraud Status')
```

Out[14]: `Text(0.5, 1.0, 'Transaction Amount by Fraud Status')`

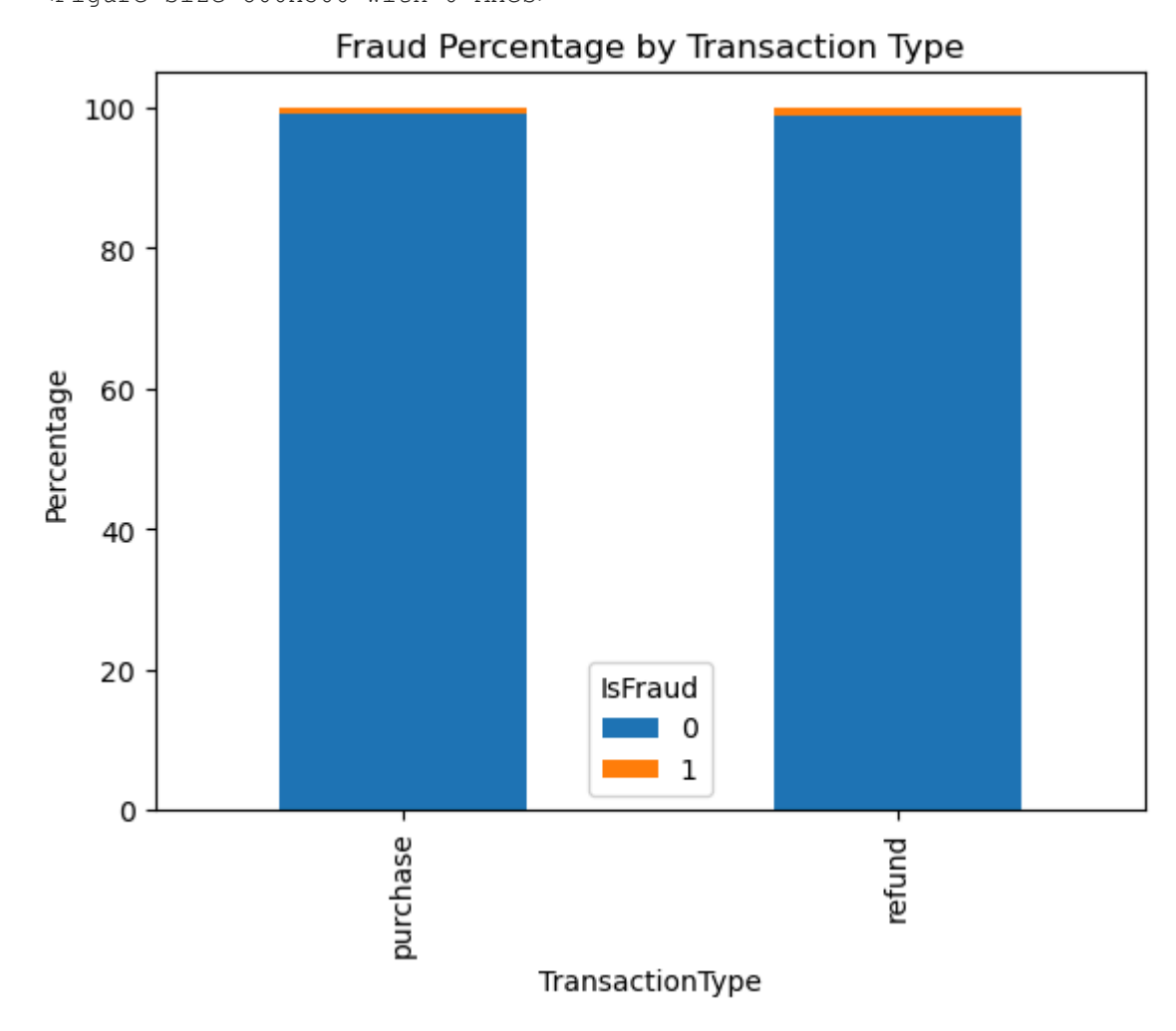


In [16]:
```python
plt.subplot(1, 2, 2)
sns.histplot(df[df['IsFraud'] == 1]['Amount'], kde=True, color='red', label='Fraud')
sns.histplot(df[df['IsFraud'] == 0]['Amount'], kde=True, color='green', label='Legitimate', alpha=0.5)
plt.title('Amount Distribution by Fraud Status')
plt.legend()
plt.tight_layout()
plt.show()
```

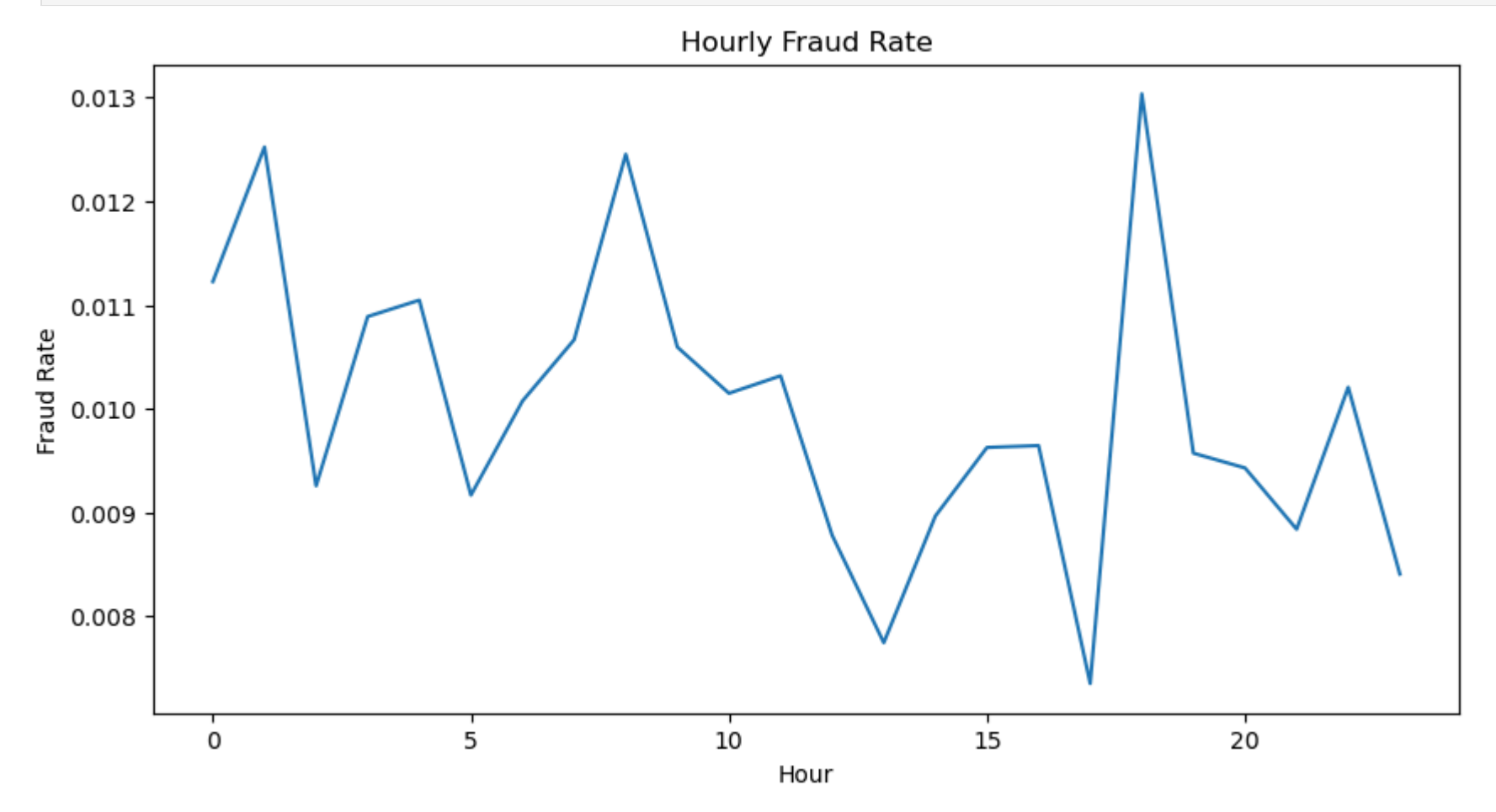


In [18]:
```python
# 2. Transaction Type Analysis
type_fraud = pd.crosstab(df['TransactionType'], df['IsFraud'], normalize='index') * 100
print("\nFraud Percentage by Transaction Type:")
print(type_fraud)

plt.figure(figsize=(8, 5))
type_fraud.plot(kind='bar', stacked=True)
plt.title('Fraud Percentage by Transaction Type')
plt.ylabel('Percentage')
plt.show()
```

```
Fraud Percentage by Transaction Type:
IsFraud                 0        1
TransactionType
purchase         99.01141  0.98859
refund           98.98865  1.01135
<Figure size 800x500 with 0 Axes>
```



In [20]:
```python
# 3. Time Analysis
# Hourly fraud pattern
hourly_fraud = df.groupby('Hour')['IsFraud'].mean().reset_index()
plt.figure(figsize=(10, 5))
sns.lineplot(x='Hour', y='IsFraud', data=hourly_fraud)
plt.title('Hourly Fraud Rate')
plt.ylabel('Fraud Rate')
plt.show()
```



In [22]:
```python
# Monthly fraud pattern
monthly_fraud = df.groupby('Month')['IsFraud'].mean().reset_index()
plt.figure(figsize=(10, 5))
sns.barplot(x='Month', y='IsFraud', data=monthly_fraud)
plt.title('Monthly Fraud Rate')
plt.ylabel('Fraud Rate')
plt.show()
```

In [24]:
```python
# 4. Location Analysis
location_fraud = df.groupby('Location')['IsFraud'].agg(['count', 'sum', 'mean']).sort_values('mean', ascending=False)
print("\nFraud by Location:")
print(location_fraud.head(10))
```

```
Fraud by Location:
              count  sum      mean
Location
New York       9993  116  0.011608
San Diego     10111  115  0.011374
Houston        9991  105  0.010509
Phoenix        9960   99  0.009940
San Antonio   10062   99  0.009839
Dallas        10076   99  0.009825
Los Angeles    9936   95  0.009561
Chicago       10193   95  0.009320
Philadelphia   9873   90  0.009116
San Jose       9805   87  0.008873
```

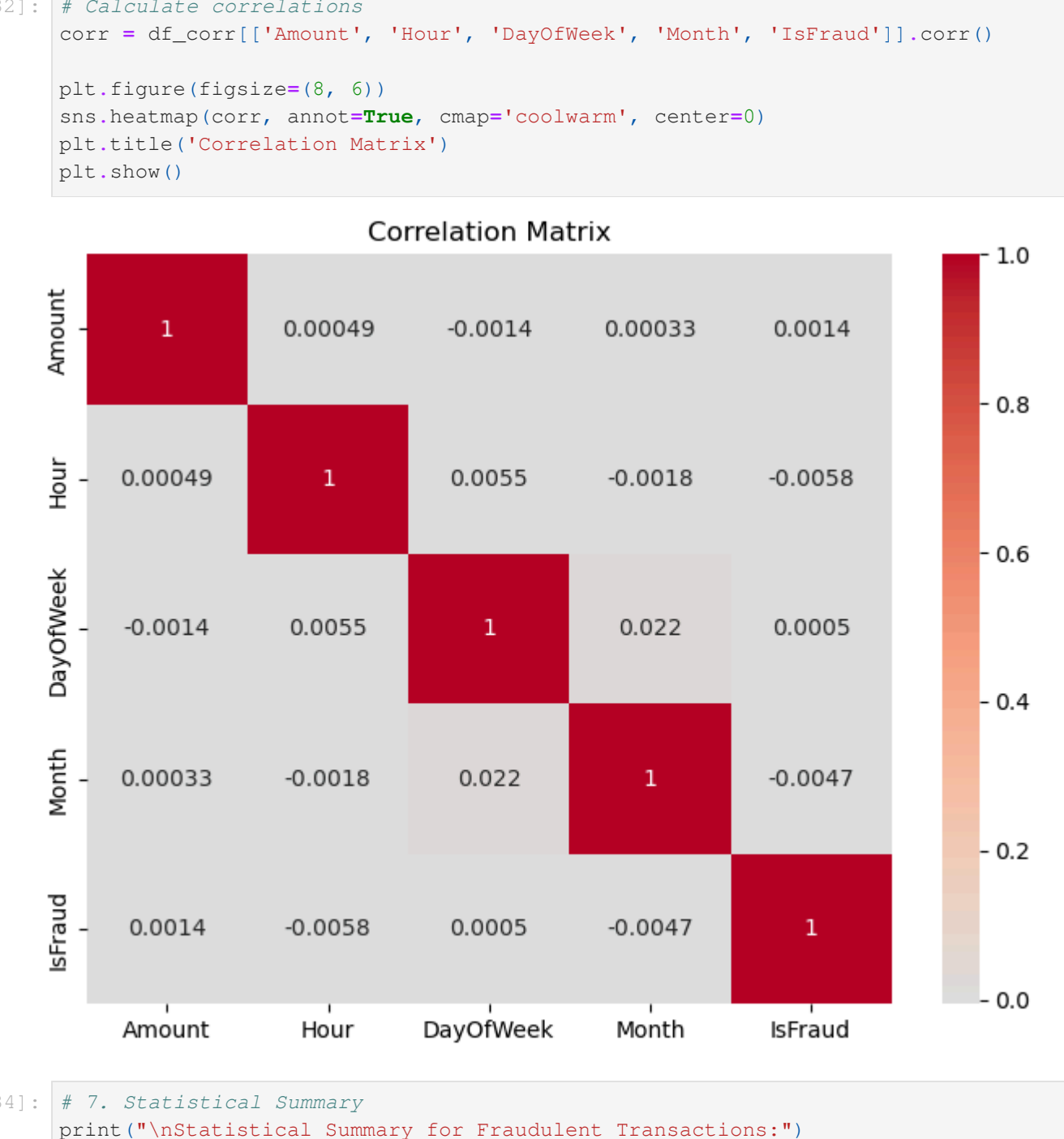
In [26]:
```python
plt.figure(figsize=(12, 6))
sns.countplot(y='Location', hue='IsFraud', data=df, order=df['Location'].value_counts().index[:10])
plt.title('Top 10 Locations by Transaction Count with Fraud Highlight')
plt.show()
```

In [28]:
```python
# 5. Merchant Analysis
merchant_fraud = df.groupby('MerchantID')['IsFraud'].agg(['count', 'sum', 'mean']).sort_values('mean', ascending=False)
print("\nMerchants with Highest Fraud Rates (with at least 5 transactions):")
print(merchant_fraud[merchant_fraud['count'] >= 5].head(10))
```

```
Merchants with Highest Fraud Rates (with at least 5 transactions):
            count  sum      mean
MerchantID
640            86    5  0.058140
156           103    5  0.048544
583           106    5  0.047170
659            88    4  0.045455
939            91    4  0.043956
436            96    4  0.041667
568            97    4  0.041237
968            73    3  0.041096
401            98    4  0.040816
989            99    4  0.040404
```

In [30]:
```python
# 6. Correlation Analysis
# Create a copy of the dataframe for correlation analysis
df_corr = df.copy()
df_corr['IsFraud'] = df_corr['IsFraud'].astype(int)
```

In [32]:
```python
# Calculate correlations
corr = df_corr[['Amount', 'Hour', 'DayOfWeek', 'Month', 'IsFraud']].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix')
plt.show()
```



In [34]:
```python
# 7. Statistical Summary
print("\nStatistical Summary for Fraudulent Transactions:")
print(df[df['IsFraud'] == 1]['Amount'].describe())

print("\nStatistical Summary for Legitimate Transactions:")
print(df[df['IsFraud'] == 0]['Amount'].describe())
```

```
Statistical Summary for Fraudulent Transactions:
count    1000.000000
mean     2517.164440
std      1458.643653
min         5.560000
25%      1249.277500
50%      2543.430000
75%      3813.850000
max      4985.880000
Name: Amount, dtype: float64

Statistical Summary for Legitimate Transactions:
count    99000.000000
mean      2496.889920
std       1442.257175
min          1.050000
25%       1247.955000
50%       2495.610000
75%       3742.855000
max       4999.770000
Name: Amount, dtype: float64
```

In [48]:
```python
# 8. Advanced Analysis - Time between transactions (example for a specific merchant)
# This would require customer IDs which aren't in this dataset, but here's a conceptual example:
# df = df.sort_values(['CustomerID', 'TransactionDate'])
# df['TimeSinceLastTx'] = df.groupby('CustomerID')['TransactionDate'].diff().dt.total_seconds() / 3600
```

In [52]:
```python
# Key Insights from the Analysis:
# Fraud Distribution: The dataset contains a small percentage of fraudulent transactions (exact percentage shown in output).

# Amount Analysis:

# Fraudulent transactions may have different amount distributions compared to legitimate ones

# Boxplots and histograms will show if fraud tends to occur at specific amount ranges

# Transaction Type:

# The analysis shows which transaction types (purchase/refund) are more susceptible to fraud

# Temporal Patterns:

# Hourly patterns may reveal when fraud is more likely to occur

# Monthly trends can show seasonal fraud patterns

# Geographical Patterns:

# Some locations may have higher fraud rates than others

# The location analysis highlights high-risk areas

# Merchant Analysis:

# Identifies merchants with unusually high fraud rates

# Helps pinpoint potential points of compromise

# Correlation Analysis:

# Shows how different features relate to fraud occurrence

# Helps identify potential risk factors
```

In [56]:
```
## END PROJECT  ##
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: