In [30]:
```
import pandas as pd
import plotly.express as px
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [5]:
```
data=pd.read_csv("Fraud_Assignment.csv")
```

In [6]:
```
data.head()
```

Out[6]:

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalance
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	

In [7]:
```
data.describe()
```

Out[7]:

	step	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	2.696617e+01	1.586670e+05	8.740095e+05	8.938089e+05	9.781600e+05	1.114198e+06
std	1.562325e+01	2.649409e+05	2.971751e+06	3.008271e+06	2.296780e+06	2.416593e+06
min	1.000000e+00	1.000000e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	1.500000e+01	1.214907e+04	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
50%	2.000000e+01	7.634333e+04	1.600200e+04	0.000000e+00	1.263772e+05	2.182604e+05
75%	3.900000e+01	2.137619e+05	1.366420e+05	1.746000e+05	9.159235e+05	1.149808e+06
max	9.500000e+01	1.000000e+07	3.890000e+07	3.890000e+07	4.210000e+07	4.220000e+07

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column          Non-Null Count    Dtype  
---  ------          --------------    -----  
 0   step            1048575 non-null  int64  
 1   type            1048575 non-null  object 
 2   amount          1048575 non-null  float64
 3   nameOrig        1048575 non-null  object 
 4   oldbalanceOrg   1048575 non-null  float64
 5   newbalanceOrig  1048575 non-null  float64
 6   nameDest        1048575 non-null  object 
 7   oldbalanceDest  1048575 non-null  float64
 8   newbalanceDest  1048575 non-null  float64
 9   isFraud         1048575 non-null  int64  
 10  isFlaggedFraud  1048575 non-null  int64  
dtypes: float64(5), int64(3), object(3)
memory usage: 88.0+ MB
```

In [9]:
```
data.isna().sum()
```

Out[9]:
```
step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

In [10]:
```
data.duplicated()
```

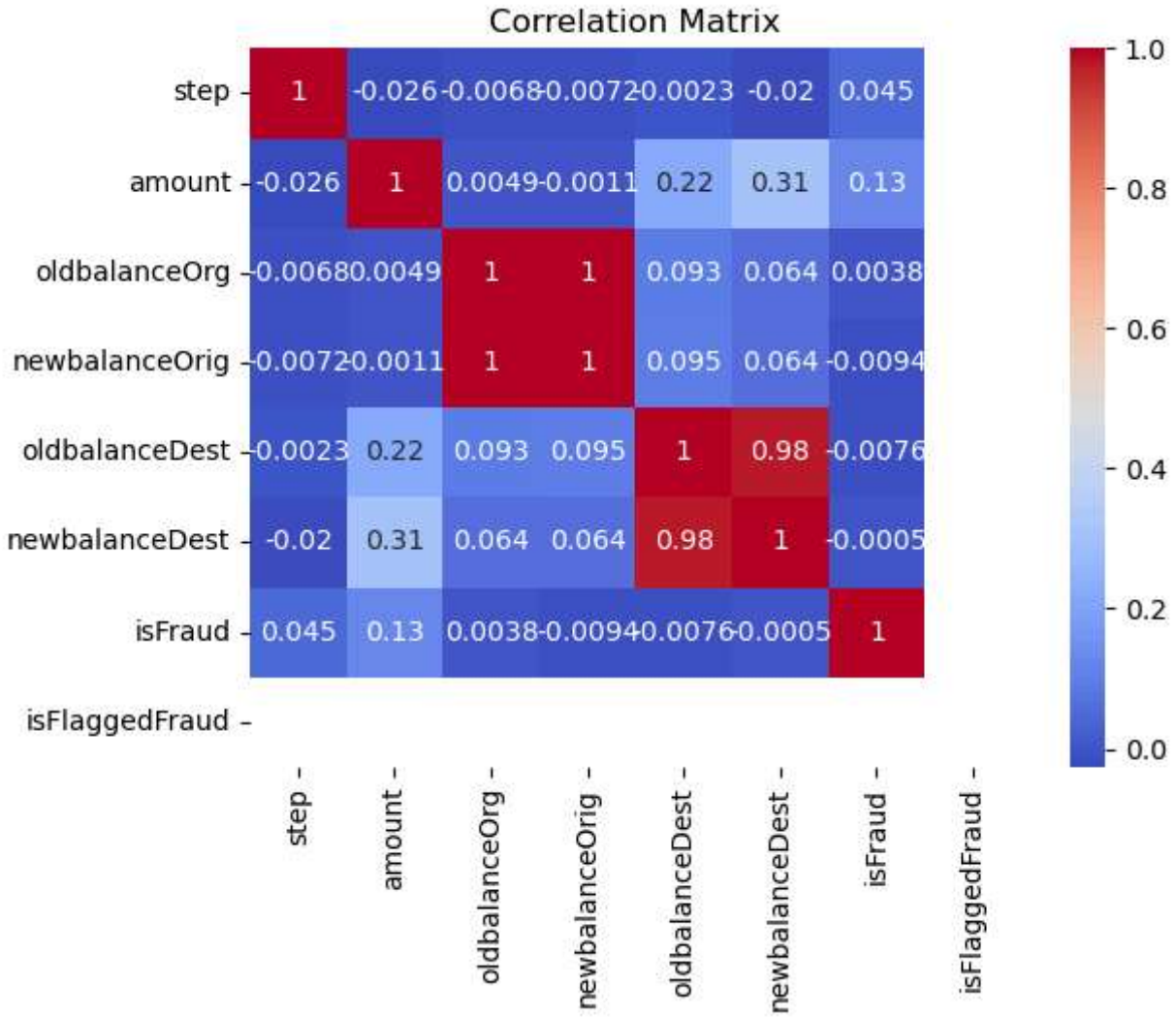
Out[10]:
```
0          False
1          False
2          False
3          False
4          False
           ...
1048570    False
1048571    False
1048572    False
1048573    False
1048574    False
Length: 1048575, dtype: bool
```

In [18]:
```
fig = px.box(data, y="amount",color_discrete_sequence=["#FF5733"] )
fig.show()
```

In [20]:

```
data_numeric = data.select_dtypes(include=['int64', 'float64'])
corr = data_numeric.corr()


# Plot heatmap
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()
```



In [22]:
```
data['difforiginamount'] = data['newbalanceOrig'] - data['oldbalanceOrg']
data['diffdestamount'] = data['newbalanceDest'] - data['oldbalanceDest']
```

In [29]:
```
data.head()
```

Out[29]:

	step	type	amount	nameOrig	nameDest	isFraud	isFlaggedFraud	difforiginamount
0	1	PAYMENT	9839.64	C1231006815	M1979787155	0	0	-9839.64
1	1	PAYMENT	1864.28	C1666544295	M2044282225	0	0	-1864.28
2	1	TRANSFER	181.00	C1305486145	C553264065	1	0	-181.00
3	1	CASH_OUT	181.00	C840083671	C38997010	1	0	-181.00
4	1	PAYMENT	11668.14	C2048537720	M1230701703	0	0	-11668.14

In [ ]:
```
data_new.head()
```

In [31]:
```
# Filter out rows where 'nameDest' starts with 'M'
data_filtered = data[~data['nameDest'].str.startswith('M')]

# Now, proceed with your model preparation and training
X = data_filtered[['difforiginamount', 'diffdestamount', 'amount', 'step']]  # Example
```

```
y = data_filtered['isFraud']  # Target column

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Initialize the RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Fit the model on the training data
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)

# Classification report
cr = classification_report(y_test, y_pred)
print("Classification Report:")
print(cr)
```

```
Accuracy: 99.94%
Confusion Matrix:
[[138690     12]
 [    68    171]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    138702
           1       0.93      0.72      0.81       239

    accuracy                           1.00    138941
   macro avg       0.97      0.86      0.91    138941
weighted avg       1.00      1.00      1.00    138941
```

In [32]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 150, 200, 250],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

# Create a RandomForest model
rf_model = RandomForestClassifier(random_state=42)

# Perform GridSearchCV
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=3, n_jobs=-1,
grid_search.fit(X_train, y_train)
```

```
# Best parameters and score
print(f"Best Parameters: {grid_search.best_params_}")
print(f"Best Accuracy: {grid_search.best_score_ * 100:.2f}%")
```

```
Fitting 3 folds for each of 60 candidates, totalling 180 fits
Best Parameters: {'max_depth': 20, 'min_samples_split': 10, 'n_estimators': 200}
Best Accuracy: 99.94%
```

In [33]:

```
from sklearn.ensemble import RandomForestClassifier

# Create the model with the best parameters
rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=20,
    min_samples_split=10,
    random_state=42
)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on test data
y_pred = rf_model.predict(X_test)

# Evaluate performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9994026241354244
Confusion Matrix:
 [[138692     10]
 [    73    166]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    138702
           1       0.94      0.69      0.80       239

    accuracy                           1.00    138941
   macro avg       0.97      0.85      0.90    138941
weighted avg       1.00      1.00      1.00    138941
```

In [ ]: