In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report
```

In [3]:
```
# Load the dataset
data_path = r'C:\Users\user\fraud_data.xlsx'
data = pd.read_excel(data_path)
```

In [4]:
```
# Preprocessing
X = data.drop(columns=["isFraud", "isFlaggedFraud"])
y = data["isFraud"]
```

In [5]:
```
# Define categorical and numerical features
categorical_features = ["type", "nameOrig", "nameDest"]
numerical_features = ["step", "amount", "oldbalanceOrg", "newbalanceOrig", "oldbalanceDest", "newbalanceDest"
```

In [6]:
```
# Preprocessing for numerical data
numerical_transformer = StandardScaler()
```

In [7]:
```
# Preprocessing for categorical data
categorical_transformer = OneHotEncoder(handle_unknown="ignore")
```

In [8]:
```
# Bundle preprocessing for numerical and categorical data
preprocessor = ColumnTransformer(
    transformers=[
        ("num", numerical_transformer, numerical_features),
        ("cat", categorical_transformer, categorical_features)
    ])
```

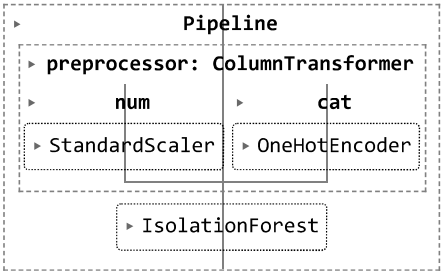
In [9]:
```
# Define the model
model = IsolationForest(random_state=42)
```

In [10]:
```
# Bundle preprocessing and modeling code in a pipeline
from sklearn.pipeline import Pipeline
pipeline = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("model", model)
])
```

In [11]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [12]:
```
# Train the model
pipeline.fit(X_train, y_train)
```

Out[12]:

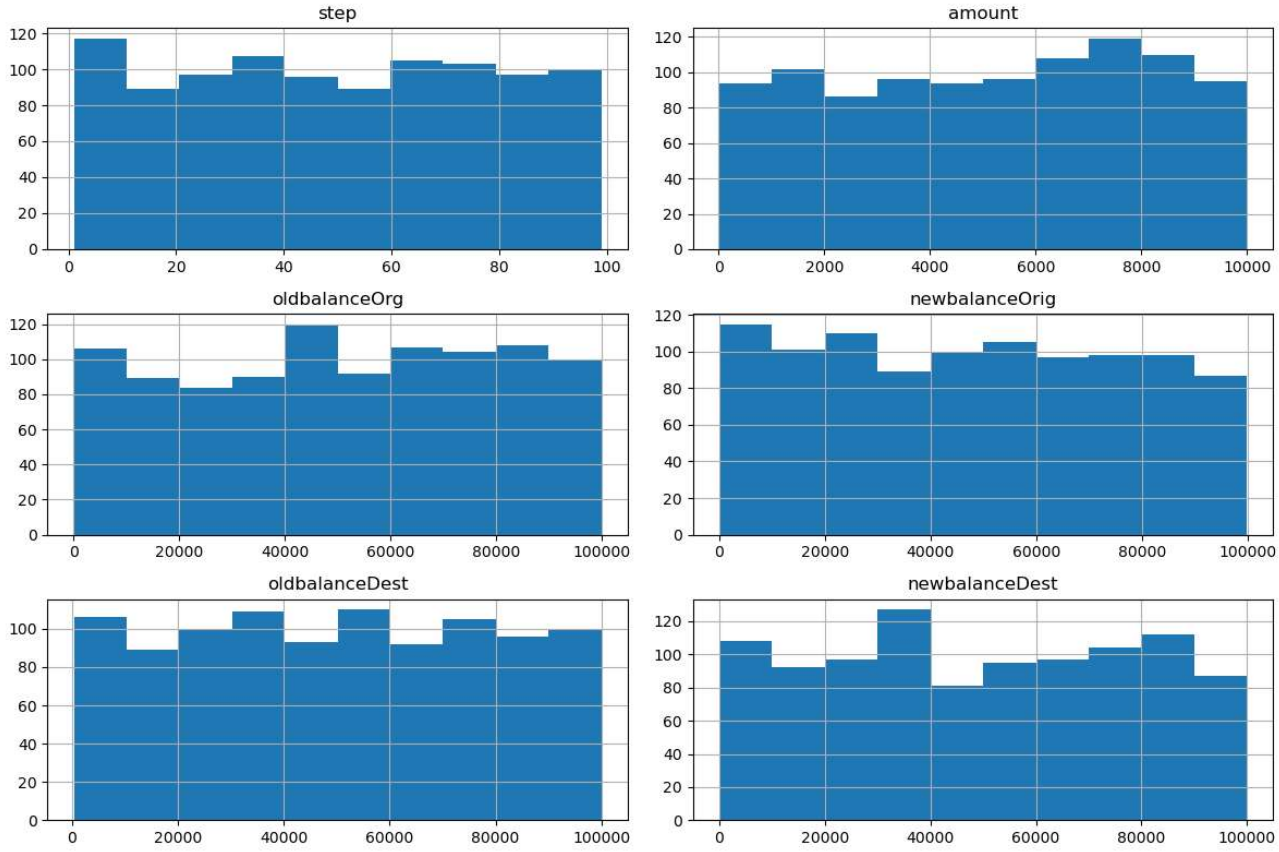


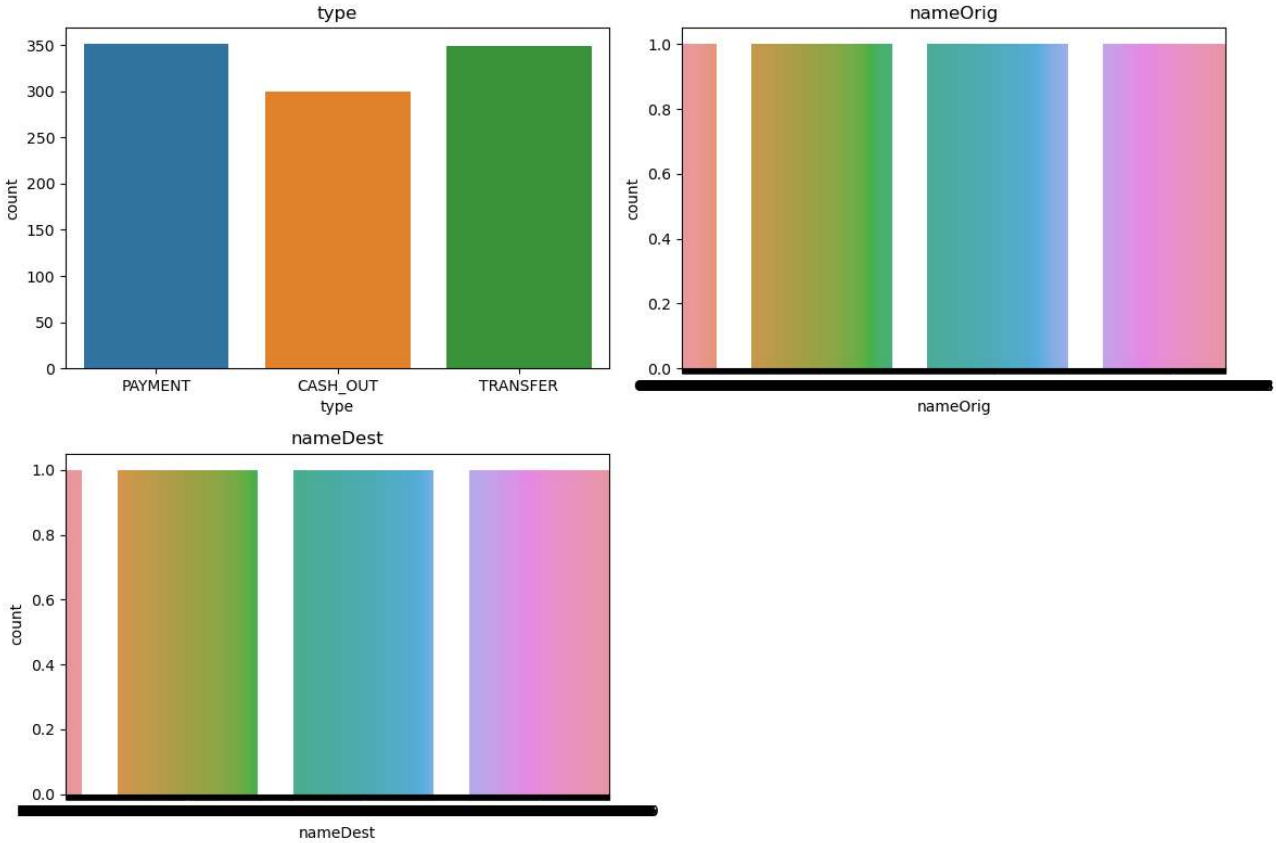
In [13]:
```
# Make predictions
y_pred = pipeline.predict(X_test)
```

In [14]:
```
# Evaluate the model
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       102
           1       0.49      1.00      0.66        98

    accuracy                           0.49       200
   macro avg       0.24      0.50      0.33       200
weighted avg       0.24      0.49      0.32       200


C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histograms of numerical features
data[numerical_features].hist(figsize=(12, 8))
plt.tight_layout()
plt.show()

# Countplots of categorical features
# Countplots of categorical features
plt.figure(figsize=(12, 8))
for i, col in enumerate(categorical_features, 1):
    plt.subplot(2, 2, i)
    sns.countplot(x=col, data=data)
    plt.title(col)
plt.tight_layout()
plt.show()
```





In [17]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```