Data Collection and Preprocessing Phase

Date	5 july 2025
Team ID	SWTID1749620997
Project Title	Online Payments Fraud Detection
Maximum Marks	6 Marks

Data Exploration and Preprocessing

Identified and evaluated diverse data sources to ensure data relevance and consistency. Assessed data quality by detecting issues such as missing values, outliers, and duplicates. Developed and implemented effective data cleaning and preprocessing strategies to ensure accurate, complete, and reliable data for analysis and decision-making.

Data Overview

Overview of the dataset including number of rows and columns, data types, sample records, and descriptive statistics. Mention of any duplicates or missing values.

Source of data

KaggleHub

Dimensions of the Dataset

- Total number of **rows** (observations): 6362620
- Total number of **columns** (features): 11

Column Names and its Datatypes

- `step` (integer): Time steps in hourly increments
- `type` (categorical): Transaction types (e.g., PAYMENT, TRANSFER)
- `amount` (float): Transaction amount
- `nameOrig` (string): Sender account ID
- `oldbalanceOrg` (float): Sender's balance before the transaction
- `newbalanceOrig` (float): Sender's balance after the transaction

- `nameDest` (string): Receiver account ID
- `oldbalanceDest` (float): Recipient's balance before transaction
- `newbalanceDest` (float): Recipient's balance after transaction
- `isFraud` (binary int): Indicator if the transaction is fraudulent (1 = yes, 0 = no)
- `isFlaggedFraud` (binary int): Indicator if the transaction was flagged by the system

Descriptive Statistic

```
Descriptive statistics for numerical columns:
               step        amount  oldbalanceOrg  newbalanceOrig  \
count  6.362620e+06  6.362620e+06   6.362620e+06    6.362620e+06
mean   2.433972e+02  1.798619e+05   8.338831e+05    8.551137e+05
std    1.423320e+02  6.038582e+05   2.888243e+06    2.924049e+06
min    1.000000e+00  0.000000e+00   0.000000e+00    0.000000e+00
25%    1.560000e+02  1.338957e+04   0.000000e+00    0.000000e+00
50%    2.390000e+02  7.487194e+04   1.420800e+04    0.000000e+00
75%    3.350000e+02  2.087215e+05   1.073152e+05    1.442584e+05
max    7.430000e+02  9.244552e+07   5.958504e+07    4.958504e+07

       oldbalanceDest  newbalanceDest       isFraud
count    6.362620e+06    6.362620e+06  6.362620e+06
mean     1.100702e+06    1.224996e+06  1.290820e-03
std      3.399180e+06    3.674129e+06  3.590480e-02
min      0.000000e+00    0.000000e+00  0.000000e+00
25%      0.000000e+00    0.000000e+00  0.000000e+00
50%      1.327057e+05    2.146614e+05  0.000000e+00
75%      9.430367e+05    1.111909e+06  0.000000e+00
max      3.560159e+08    3.561793e+08  1.000000e+00
```

```
Value counts for categorical columns (top 10 for brevity):

--- Column: type ---
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

Sample Record

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Missing Values

```
memory usage: 430.94 MB
step              0
type              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
dtype: int64
```

Unique Values

```
Unique Transaction Types: ['PAYMENT' 'TRANSFER' 'CASH_OUT' 'DEBIT' 'CASH_IN']
type
CASH_OUT    2237500
PAYMENT     2151495
CASH_IN     1399284
TRANSFER     532909
DEBIT         41432
Name: count, dtype: int64
```

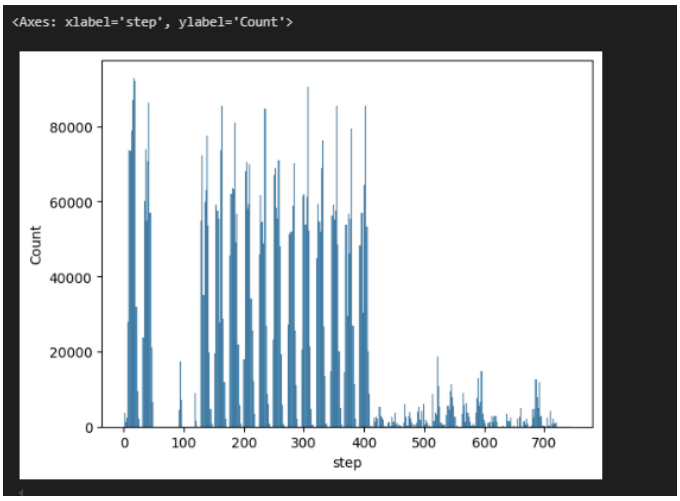
Target Variable

isFraud

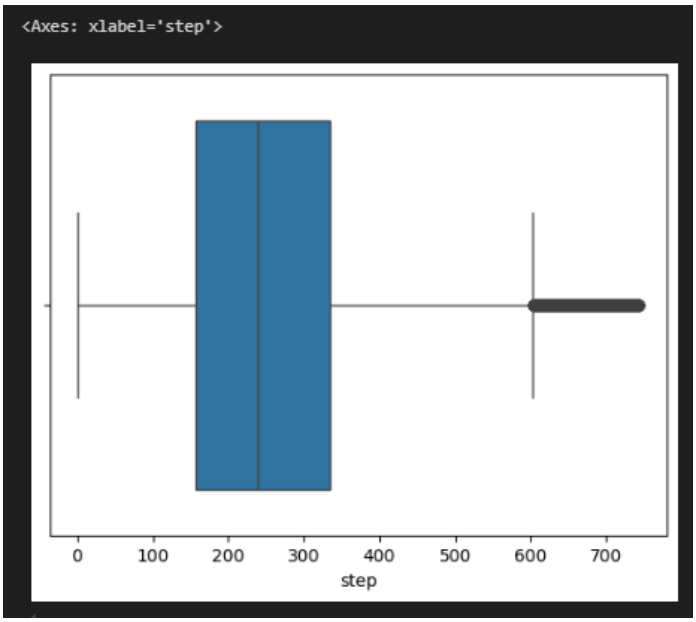
Univariate Analysis

Analyze each variable separately to understand its distribution using mean, median, mode, histograms, box plots, etc. Highlight skewness or kurtosis.

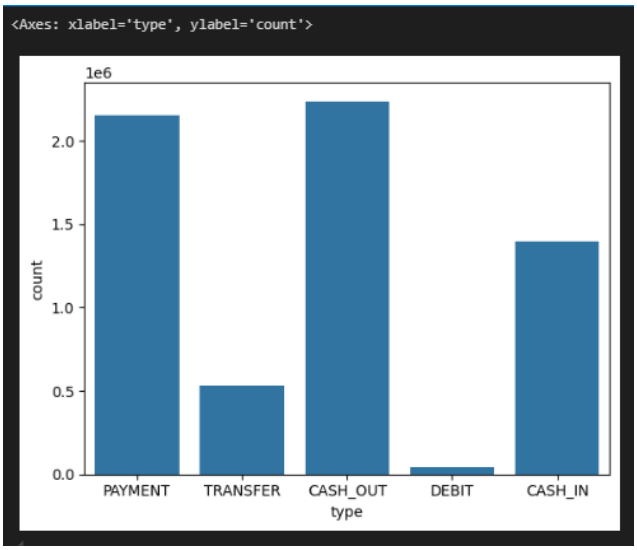
```
1] sns.histplot(data=df,x='step')
```



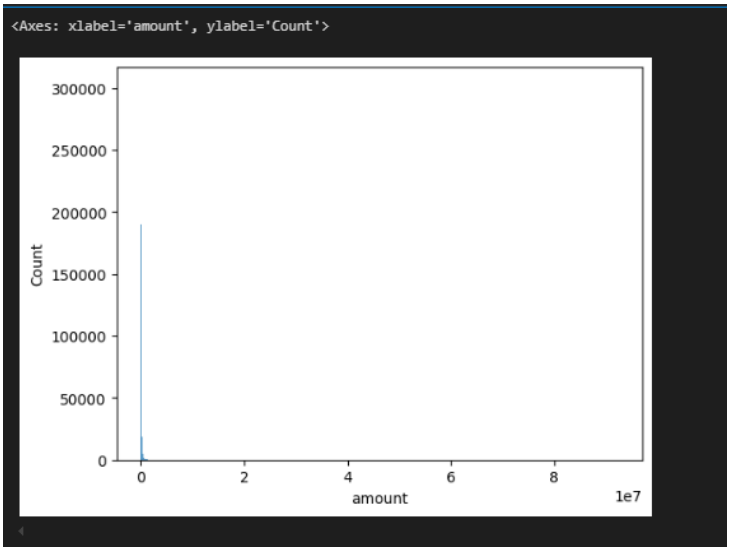

```
2] sns.boxplot(data=df,x='step')
```



```
3] sns.countplot(data=df,x='type')
```

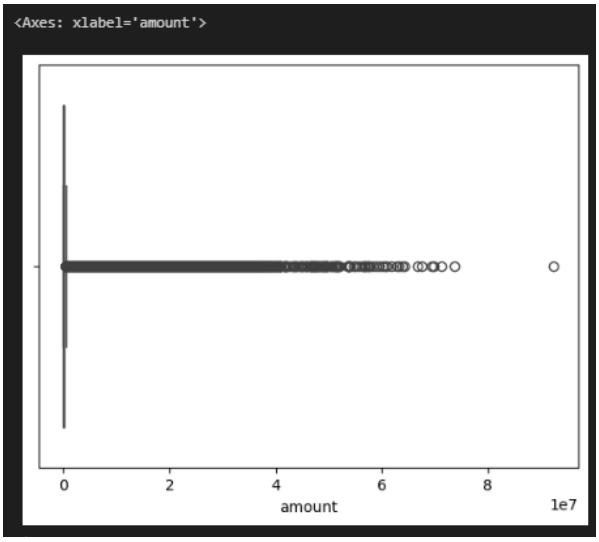



```
4] sns.histplot(data=df,x='amount')
```



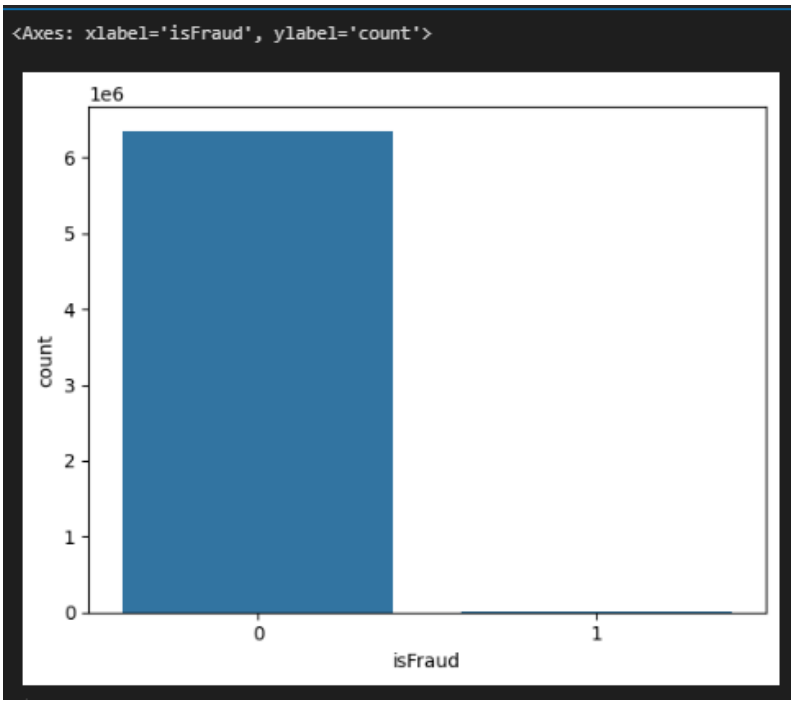
```
5] sns.boxplot(data=df,x='amount')
```

```
<Axes: xlabel='amount'>
```




```
6] sns.countplot(data=df,x='isFraud')
```

```
<Axes: xlabel='isFraud', ylabel='count'>
```



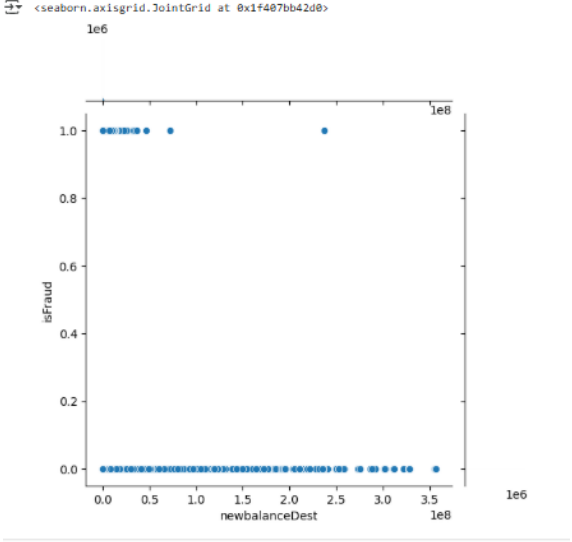
```
isFraud
0    6354407
1       8213
```

Bivariate Analysis

```
plt.figure(figsize=(10, 5))
sns.boxplot(x='isFraud', y='amount', data=df)
plt.yscale('log')  # Amount has high range
plt.title('Transaction Amounts by Fraud Status')
plt.show()
```

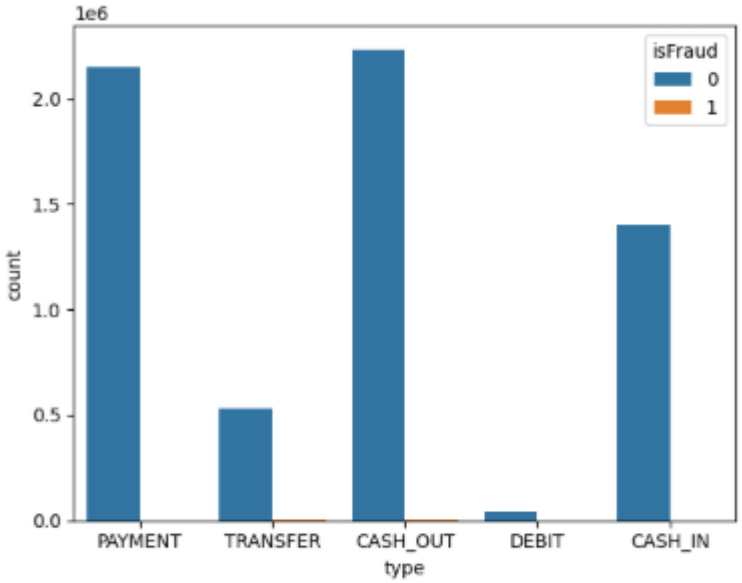

```
sns.jointplot(data=df,x='newbalanceDest',y='isFraud')
```

```
<seaborn.axisgrid.JointGrid at 0x1f407bb42d0>
```



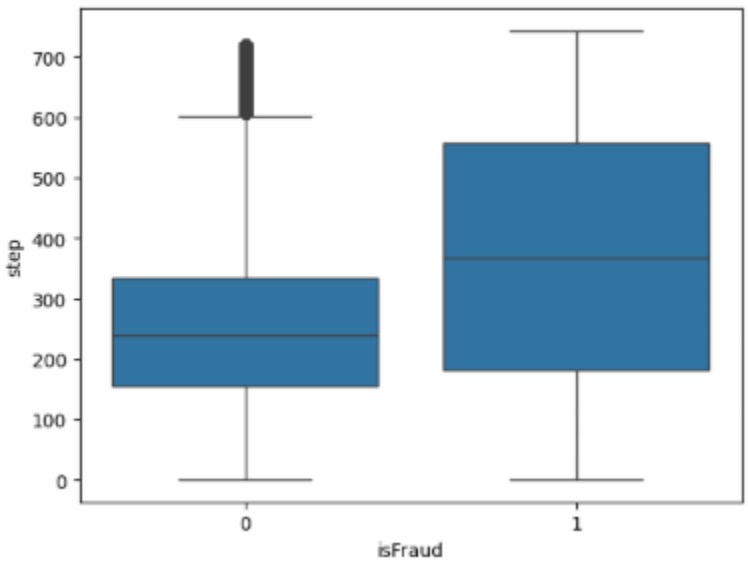
```
sns.countplot(data=df,x='type',hue='isFraud')
```

```
<Axes: xlabel='type', ylabel='count'>
```



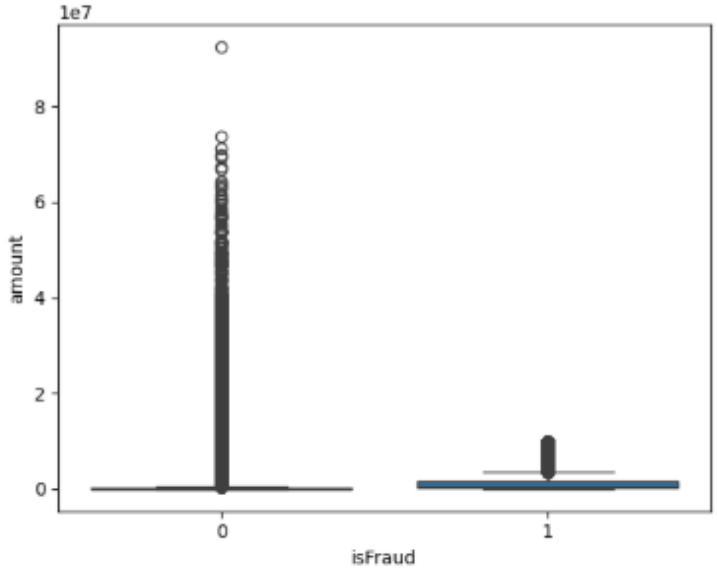

```
sns.boxplot(data=df,x='isFraud',y='step')
```

```
<Axes: xlabel='isFraud', ylabel='step'>
```



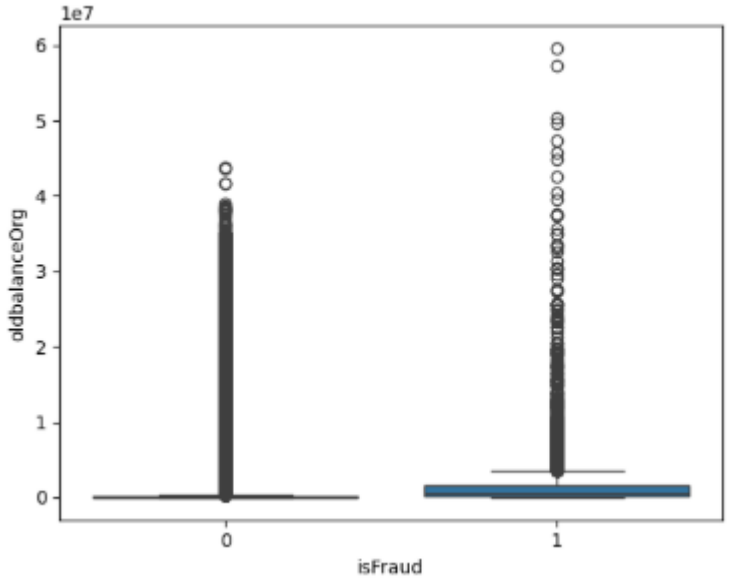
```
sns.boxplot(data=df,x='isFraud',y='amount')
```

```
<Axes: xlabel='isFraud', ylabel='amount'>
```



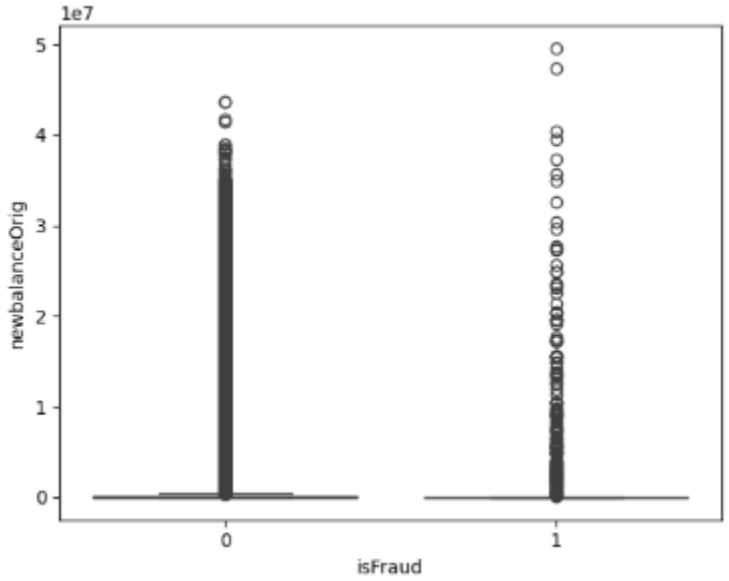

```
sns.boxplot(data=df,x='isFraud',y='oldbalanceOrg')
```

```
<Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>
```



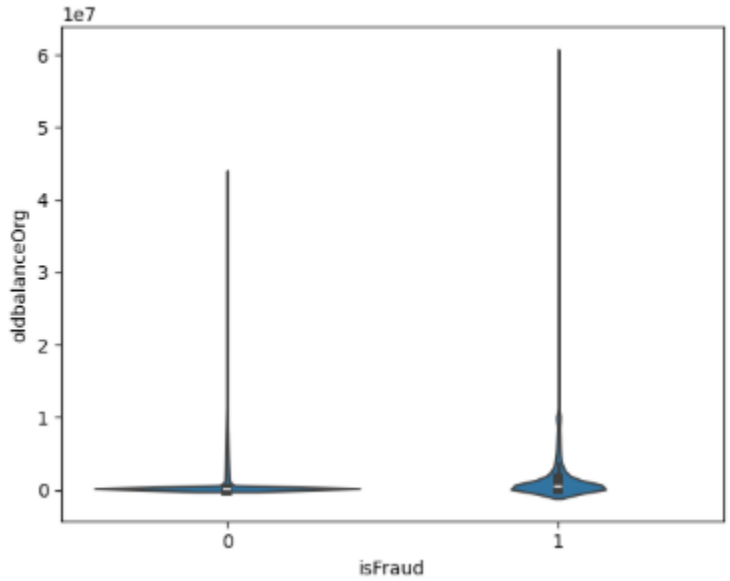
```
sns.boxplot(data=df,x='isFraud',y='newbalanceOrig')
```

```
<Axes: xlabel='isFraud', ylabel='newbalanceOrig'>
```

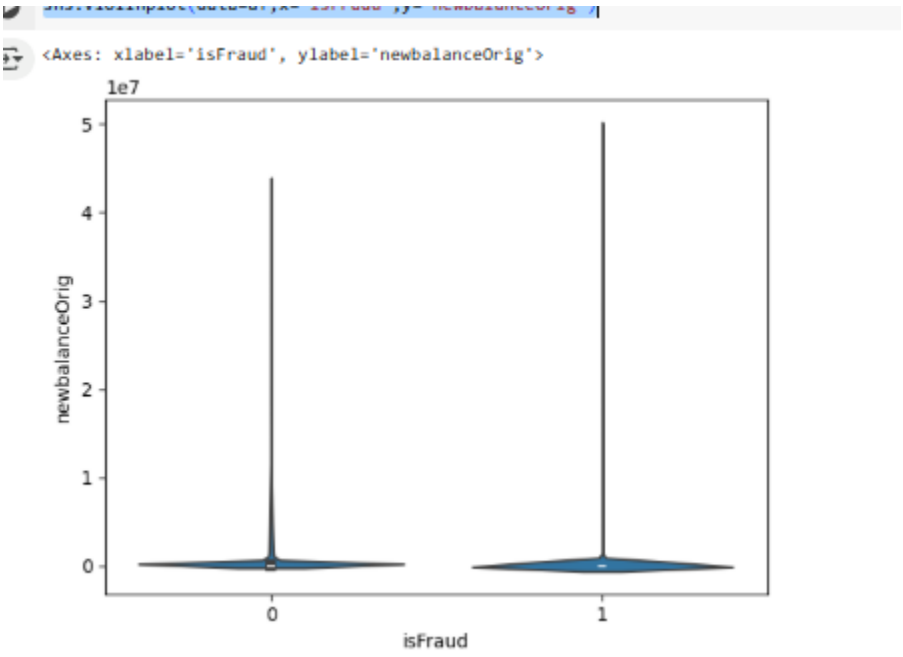



```
sns.violinplot(data=df,x='isFraud',y='oldbalanceOrg')
```

```
<Axes: xlabel='isFraud', ylabel='oldbalanceOrg'>
```

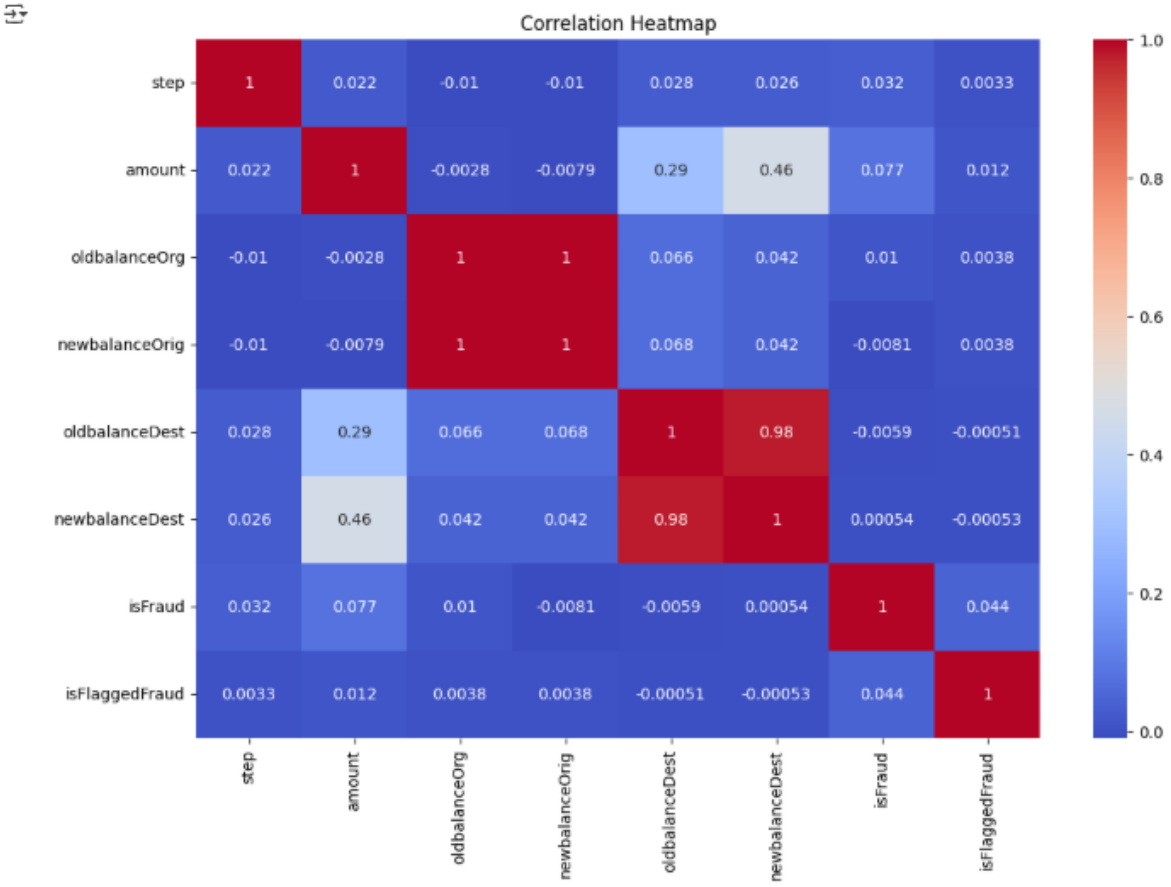


```
sns.violinplot(data=df,x='isFraud',y='newbalanceOrig')
```



Multivariate Analysis

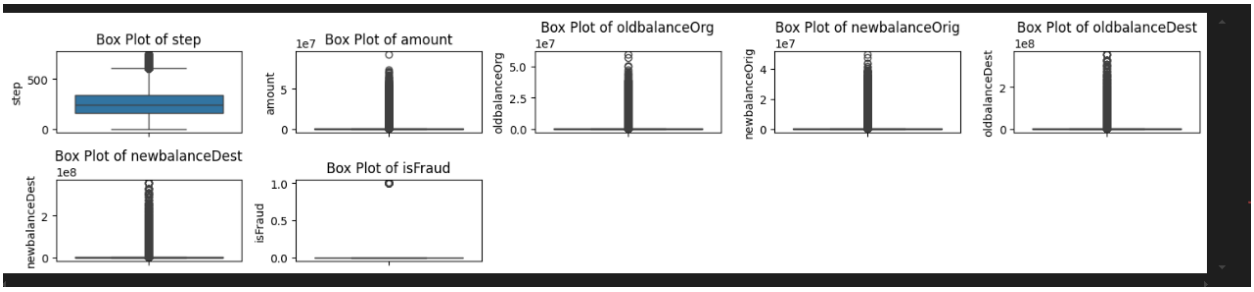
```
correlation_matrix = df.corr(numeric_only=True)
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



Outliners and Anomalies

Box plots were generated for each numerical feature to visually inspect the presence of outliers.

```
print("\n--- Step 4: Outlier Detection (using Box Plots) ---")

plt.figure(figsize=(15, 8))
for i, col in enumerate(numerical_cols):
    plt.subplot(5, 5, i + 1) # Adjust subplot grid based on number of numerical
columns
    sns.boxplot(y=df[col])
    plt.title(f'Box Plot of {col}')
plt.tight_layout()
plt.show()
```

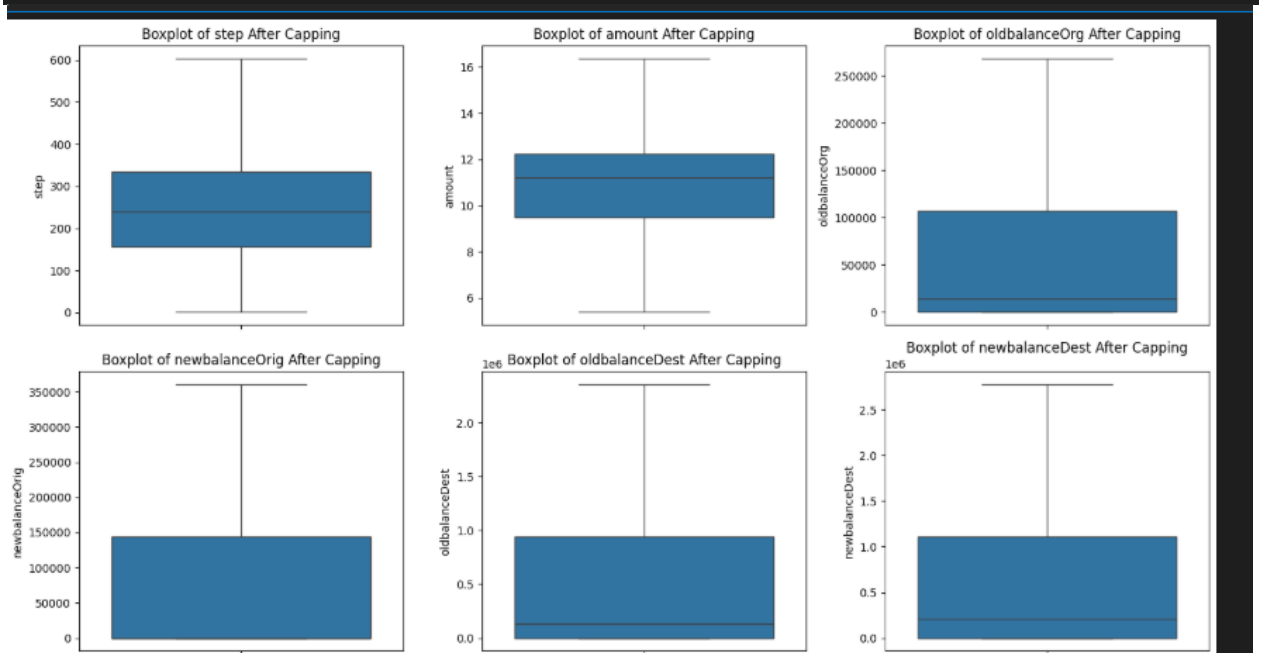


Using the IQR method, outliers were identified in all numerical columns and capped.

```
def cap_outliers_iqr(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Cap values
        df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])
        df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])

        print(f"{col}: capped using IQR method")

# Apply to your numerical columns
numerical_cols = ['step', 'amount', 'oldbalanceOrg', 'newbalanceOrig',
'oldbalanceDest', 'newbalanceDest']
cap_outliers_iqr(df, numerical_cols)
```

Re-plot the boxplot

```
plt.figure(figsize=(15, 8))
for i, col in enumerate(numerical_cols):
    plt.subplot(2, 3, i + 1)
    sns.boxplot(y=df[col])
    plt.title(f'Boxplot of {col} After Capping')
plt.tight_layout()
plt.show()
```



Data Preprocessing Code Screenshots

Loading Data

```
df = pd.read_csv("../data/PS_20174392719_1491204439457_log.csv")
df.drop(['nameOrig', 'nameDest'], axis=1, inplace=True)
```

Handling Missing Data

```
print("\nMissing values per column:")
missing_values = df.isnull().sum()
print(missing_values)
```

✓ 0.3s

```
Missing values per column:
step              0
type              0
amount            0
oldbalanceOrg     0
newbalanceOrig    0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
dtype: int64
```

No missing data

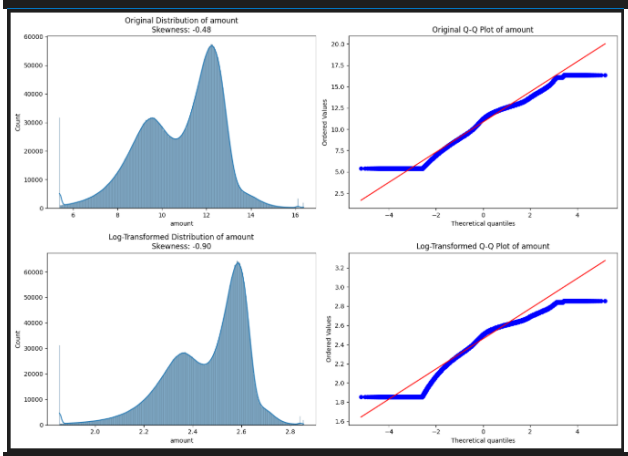
Data Transformation

A custom function was used to visualize both the original and log-transformed distributions of numerical features. The transformation reduced skewness significantly and brought the distributions closer to normality, as seen in the updated histograms and Q-Q plots.

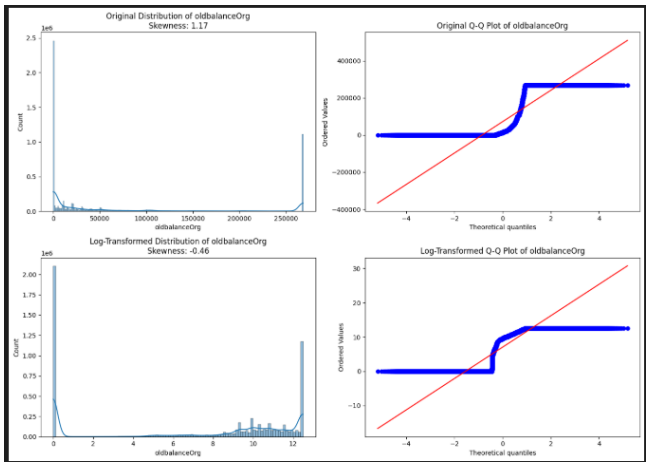
```
def transformationPlot(df, column):
    # Original feature before transformation
    original = df[column]
    transformed = np.log(original + 1)

    fig, axes = plt.subplots(2, 2, figsize=(14, 10))

    # Original Histogram
    sns.histplot(original, kde=True, ax=axes[0, 0])
    axes[0, 0].set_title(f'Original Distribution of {column}\nSkewness:
{skew(original):.2f}')
```

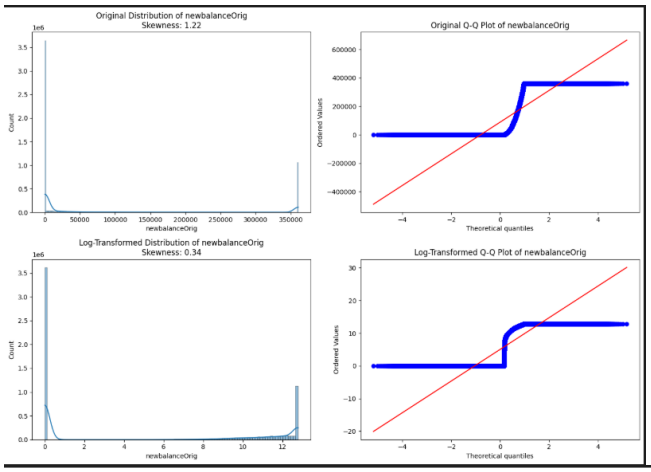
```
    # Original Q-Q Plot
    stats.probplot(original, dist="norm", plot=axes[0, 1])
    axes[0, 1].set_title(f'Original Q-Q Plot of {column}')

    # Transformed Histogram
    sns.histplot(transformed, kde=True, ax=axes[1, 0])
    axes[1, 0].set_title(f'Log-Transformed Distribution of {column}\nSkewness:
{skew(transformed):.2f}')

    # Transformed Q-Q Plot
    stats.probplot(transformed, dist="norm", plot=axes[1, 1])
    axes[1, 1].set_title(f'Log-Transformed Q-Q Plot of {column}')

    plt.tight_layout()
    plt.show()

    df[column] = transformed
```

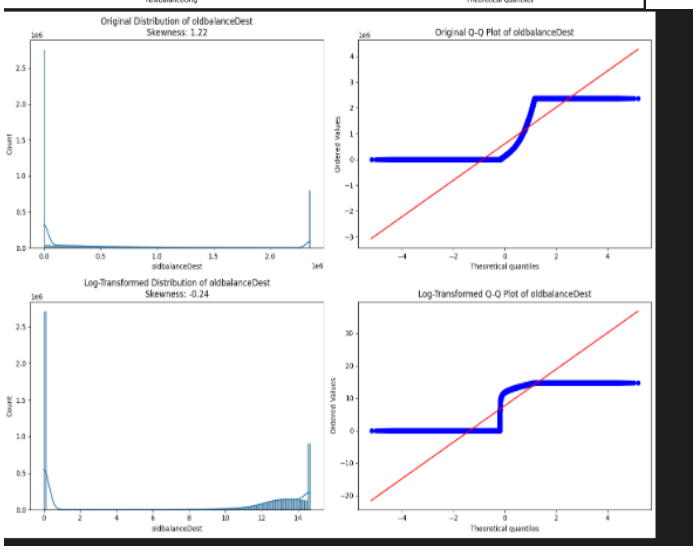
```
transformationPlot(df, 'amount')
```

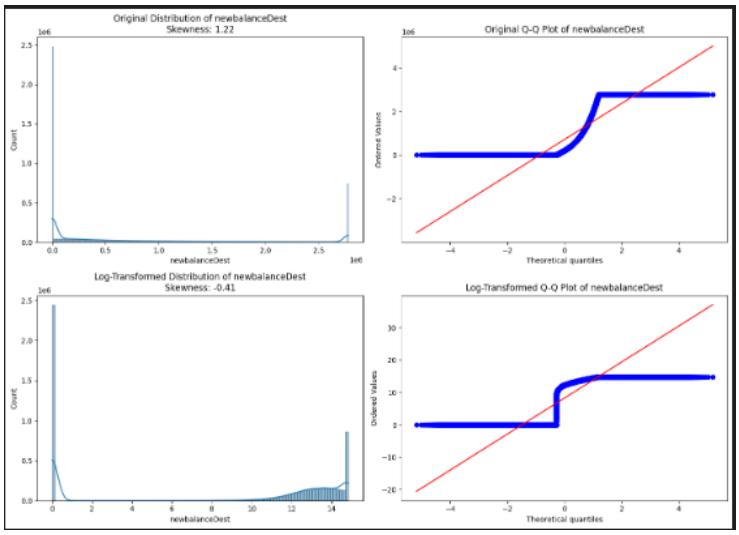



```
for col in ['oldbalanceOrg', 'newbalanceOrig', 'oldbalanceDest',
'newbalanceDest']:
    transformationPlot(df, col)
```


Original Distribution of oldbalanceOrg
Skewness: 1.17
Original Q-Q Plot of oldbalanceOrg
Log-Transformed Distribution of oldbalanceOrg
Skewness: -0.46
Log-Transformed Q-Q Plot of oldbalanceOrg
Count
oldbalanceOrg
Ordered Values
Theoretical quantiles


Original Distribution of newbalanceOrig
Skewness: 1.22
Original Q-Q Plot of newbalanceOrig
Log-Transformed Distribution of newbalanceOrig
Skewness: 0.34
Log-Transformed Q-Q Plot of newbalanceOrig
Count
newbalanceOrig
Ordered Values
Theoretical quantiles


Original Distribution of oldbalanceDest
Skewness: 1.22
Original Q-Q Plot of oldbalanceDest
Log-Transformed Distribution of oldbalanceDest
Skewness: -0.24
Log-Transformed Q-Q Plot of oldbalanceDest
Count
oldbalanceDest
Ordered Values
Theoretical quantiles



Feature Engineering

```
df['sender_balance_diff'] = df['oldbalanceOrg'] - df['newbalanceOrig']
```
✓ 0.8s

```
df['receiver_balance_diff'] = df['newbalanceDest'] - df['oldbalanceDest']
```
✓ 0.0s

```
df['system_balance_change'] = df['receiver_balance_diff'] - df['sender_balance_diff']
```
✓ 0.0s

```
df['sender_was_empty'] = (df['oldbalanceOrg'] == 0).astype(int)
```
✓ 0.1s

```
df['receiver_was_empty'] = (df['oldbalanceDest'] == 0).astype(int)
```
✓ 0.0s

Save Processed data

After completing preprocessing steps including outlier capping, missing value handling, transformation, and feature engineering, the final dataset was saved as a CSV file (`processed_fraud_data.csv`) for future use in model training and evaluation.