# Anomaly Detection in Banking Transactions

## Background

Fraud doesn't just cost money—it costs trust. As a key member of a financial compliance team, your mission is to uncover suspicious transactions that might otherwise slip through the cracks. With the stakes this high, your insights could be the difference between stopping fraud in its tracks or letting it go unnoticed.

In this project, you'll harness the power of IForest from pyod.models to detect anomalies in banking data. Your challenge: flag unusual transactions, summarize your findings, and deliver actionable insights that ensure trust, security, and efficiency in financial operations.

**Hints and Notes**

- Ensure there are no missing values in the output DataFrame.
- Be sure to label the axes and legend clearly on the histogram to make the anomalies easily identifiable.

## Dataset

You will work with a dataset containing information about financial transactions. Below is a summary of the key columns provided:

| Column | Description |
|---|---|
| `TransactionID` | A unique identifier for each transaction. |
| `TransactionAmount` | The amount of money involved in the transaction (in USD). |
| `TransactionDuration` | Duration of the transaction (in seconds). |
| `AccountBalance` | The balance of the account after the transaction was processed (in USD). |

## Instructions

Explore the transactions.csv dataset and use your findings to discover the potential for fraud by focusing on anomalies.

- Compute an anomaly score for each transaction and add it to the transactions DataFrame in a new column named Anomaly_Score.

- Which transactions are flagged as anomalies? Add a boolean column to the transactions DataFrame named Anomaly where True indicates an anomalous transaction.
- Create a summary of anomalous transactions. Save this as a pandas DataFrame called anomalies_summary, containing the following columns:TransactionID,TransactionAmount,TransactionDuration,AccountBalance.
- What is the distribution of TransactionAmount for normal and anomalous transactions? Generate and save a histogram as anomalies_histogram.png that visualizes these two groups with distinct colors. Look for differences in the distribution of transaction amounts between normal and anomalous transactions.

In [9]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pyod.models.iforest import IForest
```

In [10]:
```
# Load the dataset
transactions = pd.read_csv("../data_raw/transactions.csv")
transactions
```

Out[10]:

| | TransactionID | AccountID | TransactionAmount | TransactionDate | TransactionTyp |
|---|---|---|---|---|---|
| **0** | TX000001 | AC00128 | 14.09 | 2023-04-11 16:29:14 | Deb |
| **1** | TX000002 | AC00455 | 376.24 | 2023-06-27 16:44:19 | Deb |
| **2** | TX000003 | AC00019 | 126.29 | 2023-07-10 18:16:08 | Deb |
| **3** | TX000004 | AC00070 | 184.50 | 2023-05-05 16:32:11 | Deb |
| **4** | TX000005 | AC00411 | 13.45 | 2023-10-16 17:51:24 | Cred |
| **...** | ... | ... | ... | ... | . |
| **2507** | TX002508 | AC00297 | 856.21 | 2023-04-26 17:09:36 | Cred |
| **2508** | TX002509 | AC00322 | 251.54 | 2023-03-22 17:36:48 | Deb |
| **2509** | TX002510 | AC00095 | 28.63 | 2023-08-21 17:08:50 | Deb |
| **2510** | TX002511 | AC00118 | 185.97 | 2023-02-24 16:24:46 | Deb |
| **2511** | TX002512 | AC00009 | 243.08 | 2023-02-14 16:21:23 | Cred |

2512 rows × 16 columns

In [11]:

```
# Isolate key columns
columns_to_display = ["TransactionID", "TransactionAmount", "TransactionDura
transactions = transactions[columns_to_display]
transactions
```

Out[11]:

| | TransactionID | TransactionAmount | TransactionDuration | AccountBalance |
|---|---|---|---|---|
| **0** | TX000001 | 14.09 | 81 | 5112.21 |
| **1** | TX000002 | 376.24 | 141 | 13758.91 |
| **2** | TX000003 | 126.29 | 56 | 1122.35 |
| **3** | TX000004 | 184.50 | 25 | 8569.06 |
| **4** | TX000005 | 13.45 | 198 | 7429.40 |
| **...** | ... | ... | ... | ... |
| **2507** | TX002508 | 856.21 | 109 | 12690.79 |
| **2508** | TX002509 | 251.54 | 177 | 254.75 |
| **2509** | TX002510 | 28.63 | 146 | 3382.91 |
| **2510** | TX002511 | 185.97 | 19 | 1776.91 |
| **2511** | TX002512 | 243.08 | 93 | 131.25 |

2512 rows × 4 columns

In [15]:

```
# Load the dataset
transactions = pd.read_csv("../data_raw/transactions.csv")

# Select numerical features for anomaly detection
features = transactions[["TransactionAmount", "TransactionDuration", "Accour

# Train an IForest model with n_estimators parameter
model = IForest(n_estimators=100, contamination=0.05, random_state=42)
model.fit(features)
```

Out[15]:

```
IForest(behaviour='old', bootstrap=False, contamination=0.05,
    max_features=1.0, max_samples='auto', n_estimators=100, n_jobs=1,
    random_state=42, verbose=0)
```

In [ ]:

```
# Add the anomaly scores to the dataset
transactions["Anomaly_Score"] = model.decision_function(features)

# Flag transactions as anomalies based on the model's prediction
transactions["Anomaly"] = (model.predict(features) == 1).astype(int)  # Conv
```

```
/opt/anaconda3/envs/conda_env/lib/python3.12/site-packages/sklearn/base.py:4
86: UserWarning: X has feature names, but IsolationForest was fitted without
feature names
  warnings.warn(
/opt/anaconda3/envs/conda_env/lib/python3.12/site-packages/sklearn/base.py:4
86: UserWarning: X has feature names, but IsolationForest was fitted without
feature names
  warnings.warn(
```

In [18]:

```
# Create a summary of anomalous transactions
anomalies_summary = transactions.loc[transactions["Anomaly"] == 1, ["Transac
```

```
# Plot the distribution of TransactionAmount for normal and anomalous transa
plt.figure(figsize=(8, 6))
transactions[transactions["Anomaly"] == False]["TransactionAmount"].hist(bir
transactions[transactions["Anomaly"] == True]["TransactionAmount"].hist(bins
plt.title("Transaction Amount Distribution")
plt.xlabel("Transaction Amount")
plt.ylabel("Frequency")
plt.legend()
plt.savefig("anomalies_histogram.png")
```