In [3]:
```
import os
import glob

# List all files in the default content folder
all_files = os.listdir("/content")
pkl_files = glob.glob("/content/*.pkl")

print("All Files in /content:", all_files)
print("PKL Files Found:", pkl_files)
```

```
All Files in /content: ['.config', 'sample_data']
PKL Files Found: []
```

In [4]:
```
import pandas as pd
import glob

# Get list of all uploaded .pkl files
pkl_files = glob.glob("/content/*.pkl")

# Load and combine them
all_data = [pd.read_pickle(file) for file in pkl_files]
combined_df = pd.concat(all_data, ignore_index=True)

# Check basic info
print("Combined shape:", combined_df.shape)
print("Columns:\n", combined_df.columns)
combined_df.head()
```

```
Combined shape: (1754155, 9)
Columns:
 Index(['TRANSACTION_ID', 'TX_DATETIME', 'CUSTOMER_ID', 'TERMINAL_ID',
       'TX_AMOUNT', 'TX_TIME_SECONDS', 'TX_TIME_DAYS', 'TX_FRAUD',
       'TX_FRAUD_SCENARIO'],
      dtype='object')
```

Out[4]:

	TRANSACTION_ID	TX_DATETIME	CUSTOMER_ID	TERMINAL_ID	TX_AMOUNT	TX_TIM
0	757803	2018-06-19 00:00:03	484	4651	40.61	
1	757804	2018-06-19 00:01:22	130	9558	11.08	
2	757805	2018-06-19 00:02:13	3211	9226	33.56	
3	757806	2018-06-19 00:02:45	1877	2388	86.33	
4	757807	2018-06-19 00:03:05	1471	4033	17.69	

In [5]:
```
# Check missing values
print("Missing values:\n", combined_df.isnull().sum())

# Basic statistics
print("\nSummary statistics:")
print(combined_df.describe())

# Distribution of fraud
print("\nFraud distribution:")
print(combined_df['TX_FRAUD'].value_counts(normalize=True))
```

```
Missing values:
 TRANSACTION_ID       0
TX_DATETIME           0
CUSTOMER_ID           0
TERMINAL_ID           0
TX_AMOUNT             0
TX_TIME_SECONDS       0
TX_TIME_DAYS          0
TX_FRAUD              0
TX_FRAUD_SCENARIO     0
dtype: int64

Summary statistics:
       TRANSACTION_ID                    TX_DATETIME     TX_AMOUNT  \
count    1.754155e+06                        1754155  1.754155e+06
mean     8.770770e+05  2018-07-01 11:20:33.708571904  5.363230e+01
min      0.000000e+00            2018-04-01 00:00:31  0.000000e+00
25%      4.385385e+05     2018-05-16 14:40:46.500000  2.101000e+01
50%      8.770770e+05            2018-07-01 11:11:10  4.464000e+01
75%      1.315616e+06     2018-08-16 08:01:01.500000  7.695000e+01
max      1.754154e+06            2018-09-30 23:59:57  2.628000e+03
std      5.063811e+05                            NaN  4.232649e+01

           TX_FRAUD  TX_FRAUD_SCENARIO
count  1.754155e+06       1.754155e+06
mean   8.369272e-03       1.882388e-02
min    0.000000e+00       0.000000e+00
25%    0.000000e+00       0.000000e+00
50%    0.000000e+00       0.000000e+00
75%    0.000000e+00       0.000000e+00
max    1.000000e+00       3.000000e+00
std    9.110012e-02       2.113263e-01

Fraud distribution:
TX_FRAUD
0    0.991631
1    0.008369
Name: proportion, dtype: float64
```

In [6]:
```
from sklearn.model_selection import train_test_split

# Step 1: Select input features and target
X = combined_df[['TX_AMOUNT', 'TX_TIME_SECONDS', 'TX_TIME_DAYS']]
y = combined_df['TX_FRAUD']

# Step 2: Train-test split (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

print("Train set size:", X_train.shape)
print("Test set size:", X_test.shape)
print("Class distribution in train:", y_train.value_counts(normalize=True
```

```
Train set size: (1403324, 3)
Test set size: (350831, 3)
Class distribution in train: TX_FRAUD
0    0.991631
1    0.008369
Name: proportion, dtype: float64
```

In [7]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accu

# Step 1: Train the model
model = LogisticRegression(class_weight='balanced', max_iter=1000)
model.fit(X_train, y_train)

# Step 2: Predict on test set
y_pred = model.predict(X_test)

# Step 3: Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred)
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.5420758142809501

Classification Report:
               precision    recall  f1-score   support

           0       0.99      0.54      0.70    347895
           1       0.01      0.60      0.02      2936

    accuracy                           0.54    350831
   macro avg       0.50      0.57      0.36    350831
weighted avg       0.99      0.54      0.70    350831


Confusion Matrix:
 [[188430 159465]
 [  1189   1747]]
```

In [8]:
```
# Save the combined dataset to CSV
combined_df.to_csv("fraud_detection_dataset.csv", index=False)

# Save notebook manually from File > Download > Download .ipynb
```