# AI Credit Card Fraud Detection

## Importing Necessary Libraries

In [ ]:
```
# Importing Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

## Import and read dataset

In [ ]:
```
# Read Data into a Dataframe
raw_data_path = 'creditcard.csv'
df = pd.read_csv(raw_data_path)
df.head(10)
```

Out[ ]:

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | |
|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.2395 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.0788 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.7914 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.2376 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.5929 |
| **5** | 2.0 | -0.425966 | 0.960523 | 1.141109 | -0.168252 | 0.420987 | -0.029728 | 0.4762 |
| **6** | 4.0 | 1.229658 | 0.141004 | 0.045371 | 1.202613 | 0.191881 | 0.272708 | -0.0051 |
| **7** | 7.0 | -0.644269 | 1.417964 | 1.074380 | -0.492199 | 0.948934 | 0.428118 | 1.1206 |
| **8** | 7.0 | -0.894286 | 0.286157 | -0.113192 | -0.271526 | 2.669599 | 3.721818 | 0.3701 |
| **9** | 9.0 | -0.338262 | 1.119593 | 1.044367 | -0.222187 | 0.499361 | -0.246761 | 0.6515 |

10 rows × 31 columns

## Data Cleaning

a. Missing Value

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

b. Duplicate data

In [ ]:
```
df.duplicated().sum()
df = df.drop_duplicates()
df.duplicated().sum()
```

Out[ ]:
```
np.int64(0)
```

## Data Analysis

Question 1: What is the percentage of fraud transactions in the dataset?

In [ ]:
```
# Calculate the percentage of fraud transactions
total_transactions = len(df)
fraud_transactions = df[df['Class'] == 1]
percentage_fraud = (len(fraud_transactions) / total_transactions) * 100
# Print the percentage of fraud transactions
print(f"Percentage of fraud transactions: {percentage_fraud:.2f}%")
```

```
Percentage of fraud transactions: 0.17%
```

Question 2: What is the average transaction amount for fraud transactions?

In [ ]:
```
# Calculate the average transaction amount for fraud transactions
average_fraud_amount = fraud_transactions['Amount'].mean()
```

```
# Print the average transaction amount for fraud transactions
print(f"Average transaction amount for fraud transactions: ${average_frau
```

```
Average transaction amount for fraud transactions: $123.87
```

## Data Visualization

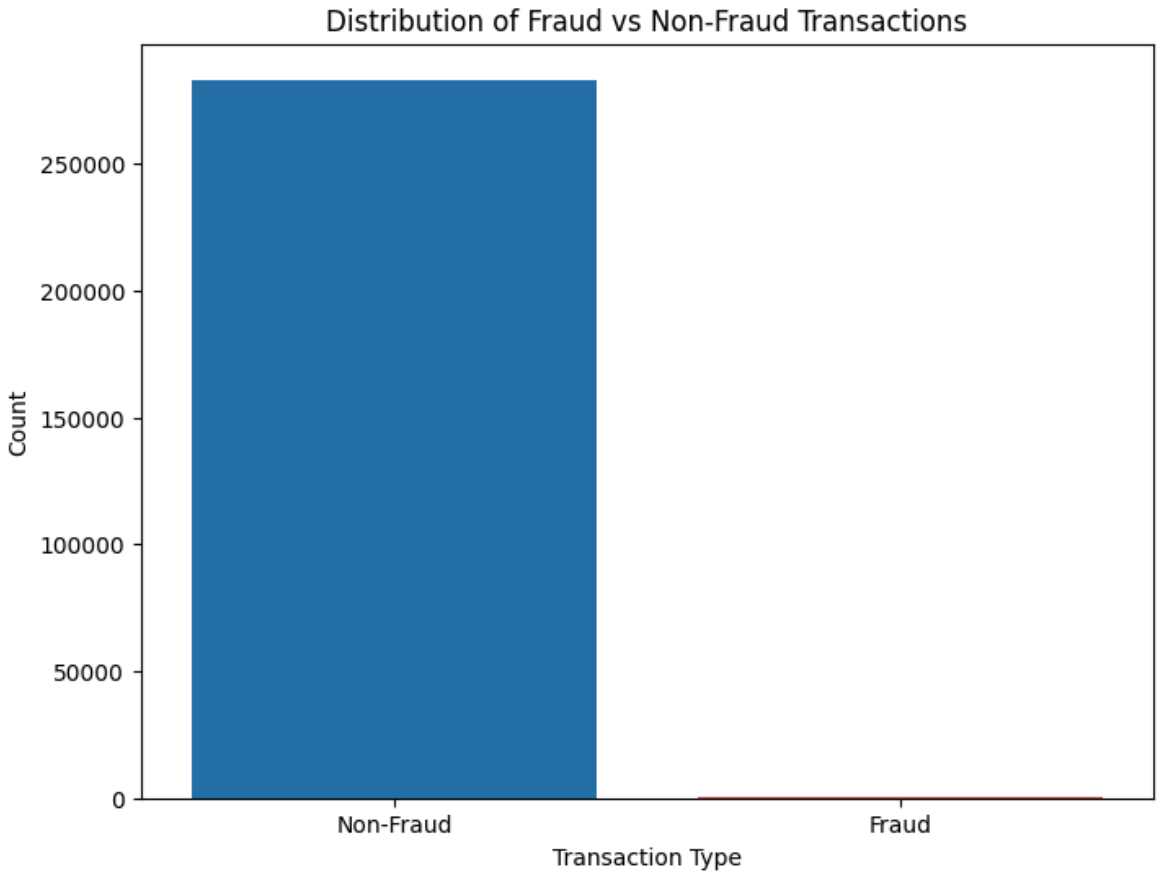
Question 1: How many fraud transactions are there compared to non-fraud transactions? (Using a bar plot)

In [ ]:
```
# Count the number of fraud and non-fraud transactions
fraud_count = df['Class'].value_counts()[1]
non_fraud_count = df['Class'].value_counts()[0]

print(f"Number of fraud transactions: {fraud_count}")
print(f"Number of non-fraud transactions: {non_fraud_count}")

# Plot the distribution of fraud transactions compared to non-fraud trans
plt.figure(figsize=(8, 6))
sns.countplot(x='Class', data=df, palette=['#0D76BF', '#E74C3C'])
plt.title('Distribution of Fraud vs Non-Fraud Transactions')
plt.xticks([0, 1], ['Non-Fraud', 'Fraud'])
plt.xlabel('Transaction Type')
plt.ylabel('Count')
plt.show()
```

```
Number of fraud transactions: 473
Number of non-fraud transactions: 283253
```

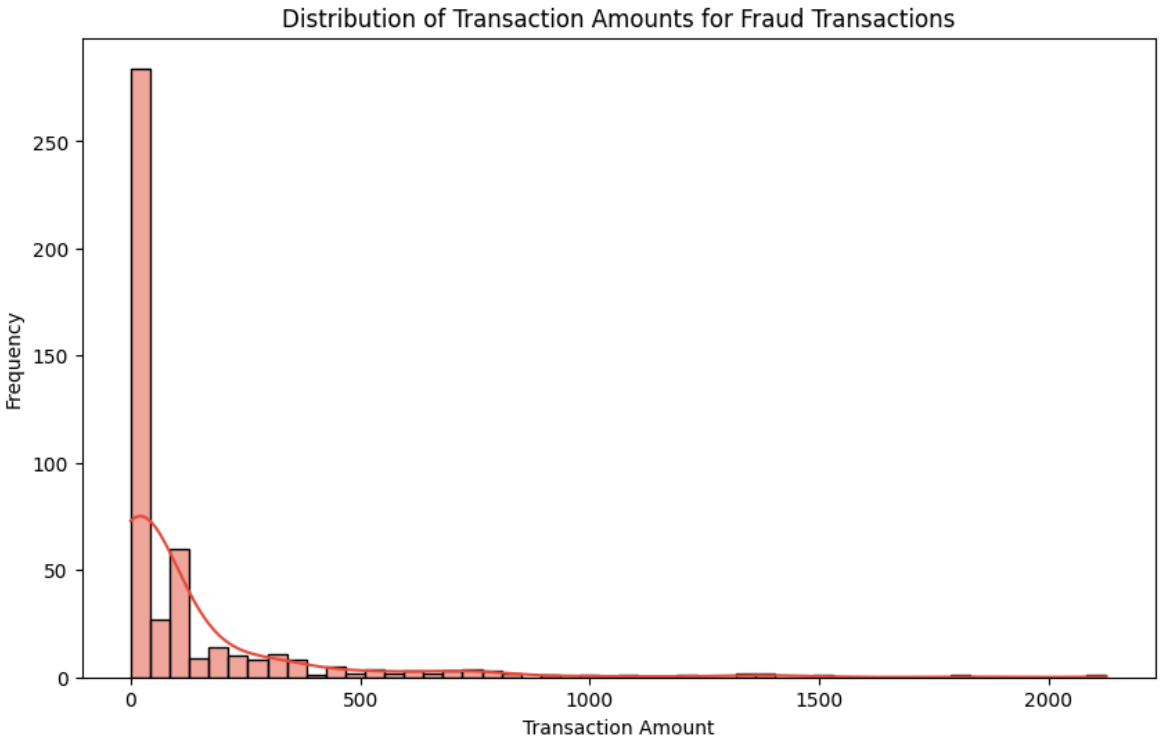



▶ **Click Here for the Hint**

Question 2: What is the distribution of transaction amounts for fraud transactions? (Using a histogram)

In [ ]:
```
# Separate the data for fraud transactions
fraud_transactions = df[df['Class'] == 1]

# Plot the distribution of transaction amounts for fraud transactions
plt.figure(figsize=(10, 6))
sns.histplot(fraud_transactions['Amount'], bins=50, kde=True, color='#E74
plt.title('Distribution of Transaction Amounts for Fraud Transactions')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()
```




## Model Development & Evaluation

## Splitting Dataset

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features and labels
X = df.drop(columns=['Class'])  # Features (all columns except 'Class')
y = df['Class']  # Labels (the 'Class' column)

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

# Initialize the StandardScaler
scaler = StandardScaler()

# Fit the scaler on the training data and transform both the training and
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Display the first 5 rows of the scaled training data
print("First 5 rows of the scaled training data:")
print(X_train_scaled[:5])
```

```
First 5 rows of the scaled training data:
[[ 1.33089570e+00  1.09595846e+00 -1.44314689e+00 -3.91914371e-01
  -1.49160826e+00 -1.15547928e-01  3.72228564e+00 -2.61558811e+00
   1.19953808e+00  3.97403741e-01  1.11088582e+00 -1.13427986e+00
  -2.38395393e-01  2.52393869e-01 -1.77795934e+00 -1.69055815e+00
  -1.01912176e+00  1.06431858e+00  1.10917107e-01  2.91004359e-03
  -5.04118946e-01 -1.02643115e-01  8.68962302e-01  3.89883732e-01
   1.22037203e+00 -6.03450203e-01  9.09711818e-02  3.53688248e-01
  -1.08344937e-01 -3.08399517e-01]
 [-4.17262374e-01 -1.10721302e+00  2.74431474e-01  3.86510616e-01
   8.32013498e-01  1.29495087e-01 -9.06209513e-01 -9.02345201e-02
   6.57230402e-01 -1.31313826e+00 -4.48124475e-01  1.22141501e+00
   1.19926273e+00  2.12134809e-01  1.36771284e+00  2.26894668e-01
   1.85588783e-02  9.19166702e-02  8.89997270e-02  1.15868724e+00
   3.73289697e-02 -9.22487295e-02 -1.19974258e+00 -6.61570665e-02
   8.28187189e-01 -3.79224312e-01 -1.52252557e+00 -6.02506358e-01
  -6.76031791e-01 -3.63961295e-01]
 [-4.00014970e-01 -3.08403784e+00 -3.56238952e+00 -8.05080089e-01
   1.57400566e+00 -6.85673862e-01  1.52660662e-02  8.75070789e-03
   7.75200776e-01 -8.23882950e-01  2.19240825e-01 -1.93520478e+00
  -9.01971246e-01 -6.29355831e-01  8.50402494e-01  1.49685722e+00
  -1.40857556e+00  7.63807518e-01  2.14248435e+00 -1.74631002e-01
   7.22105958e-01 -1.83755555e-01 -1.04244797e+00  7.42916560e-01
  -1.23668402e+00 -4.66150336e-01 -4.28046191e-01  2.33717845e+00
  -3.45254019e+00  1.87639839e+00]
 [ 4.34725667e-01  1.03537689e+00 -3.22816728e-01 -2.77757180e-01
  -8.77021603e-02 -1.35457093e-01  6.93905069e-01 -8.71156388e-01
   2.71031730e-01  9.82594773e-01  1.12422436e-01  5.44983536e-02
   1.07003392e+00  1.04445409e+00 -3.63501252e-01  1.64984823e-01
   1.24940585e+00 -1.50628003e+00  8.47854350e-01  3.74237890e-01
  -1.05311146e-01  2.77270679e-02  2.89211231e-01  3.56928547e-01
  -5.17064161e-01 -8.80459575e-01  1.02938898e+00 -3.64017670e-02
  -1.61771730e-01 -3.59921937e-01]
 [ 5.46949615e-01  7.41663039e-02  4.93287685e-01 -3.93781359e-01
  -3.59307311e-01  8.40217645e-01 -8.42423373e-01  7.79326007e-01
  -1.99485856e-01  1.65068050e-02 -1.03464036e+00 -1.18857437e+00
  -6.56787670e-01 -2.92929319e-01 -8.67450475e-01  6.33065714e-01
   5.01808489e-01  7.73074401e-02  1.07194863e+00 -4.26461853e-01
  -1.63001842e-01  3.02568417e-01  9.38125726e-01 -5.62800378e-01
  -1.21520294e+00  3.56902927e-01 -2.37877764e-01  6.55179610e-02
   4.17191899e-02 -3.44877391e-01]]
```

## Modeling & Evaluation

In [ ]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accu

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

# Train the model on the training data
rf_classifier.fit(X_train_scaled, y_train)
```

```
# Make predictions on the test data
y_pred = rf_classifier.predict(X_test_scaled)
```

In [ ]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, output_dict=True)

# Convert confusion matrix to DataFrame
conf_matrix_df = pd.DataFrame(conf_matrix, index=['Actual Non-Fraud', 'Ac
                              columns=['Predicted Non-Fraud', 'Predicted

# Convert classification report to DataFrame
class_report_df = pd.DataFrame(class_report).transpose()

# Apply styling to the confusion matrix
conf_matrix_styled = conf_matrix_df.style.background_gradient(cmap='Blues

# Apply styling to the classification report
class_report_styled = class_report_df.style.background_gradient(cmap='coo

# Display the results
print(f"Accuracy: {accuracy:.4f}\n")

# Display styled tables
display(conf_matrix_styled)
display(class_report_styled)
```

```
Accuracy: 0.9996
```

Confusion Matrix

| | Predicted Non-Fraud | Predicted Fraud |
|---|---|---|
| **Actual Non-Fraud** | 84971 | 5 |
| **Actual Fraud** | 31 | 111 |

Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **0** | 0.999635 | 0.999941 | 0.999788 | 84976.000000 |
| **1** | 0.956897 | 0.781690 | 0.860465 | 142.000000 |
| **accuracy** | 0.999577 | 0.999577 | 0.999577 | 0.999577 |
| **macro avg** | 0.978266 | 0.890816 | 0.930127 | 85118.000000 |
| **weighted avg** | 0.999564 | 0.999577 | 0.999556 | 85118.000000 |