Importing Necessary Libraries

```
import pandas as pd
import numpy as np

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
```

Load the Datasets

```
transRec = pd.read_csv('transaction_records.csv')
transData = pd.read_csv('transaction_metadata.csv')
amnt = pd.read_csv('amount_data.csv')
fraud = pd.read_csv('fraud_indicators.csv')
sus = pd.read_csv('suspicious_activity.csv')
```

Merge datasets

```
dataset = pd.merge(transRec, transData, on='TransactionID')
dataset = pd.merge(dataset, amnt, on='TransactionID')
dataset = pd.merge(dataset, fraud, on='TransactionID')
dataset = pd.merge(dataset, sus, on='CustomerID')
```

Fill missing values

```
dataset.fillna({'FraudIndicator': 0, 'SuspiciousFlag':0 }, inplace=True)
```

Feature Engineering

```
dataset['Timestamp'] = pd.to_datetime(dataset['Timestamp'])
dataset['Hour'] = dataset['Timestamp'].dt.hour
dataset['DayOfWeek'] = dataset['Timestamp'].dt.dayofweek
```

Features & Target

```
x = dataset[['Amount', 'TransactionAmount', 'Hour', 'DayOfWeek']]
y = dataset['FraudIndicator']
```

Standardize Features

```
sc = StandardScaler()
x = sc.fit_transform(x)
```

Splitting Dataset

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Building Model

```
model = tf.keras.Sequential()
model.add(Dense(64, input_dim=x_train.shape[1], activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Training Model

```
history = model.fit(x_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
20/20 [==============================] - 2s 21ms/step - loss: 0.5703 - accuracy: 0.8422 - val_loss: 0.4544 - val_accuracy: 0.9312
Epoch 2/10
20/20 [==============================] - 0s 5ms/step - loss: 0.3399 - accuracy: 0.9578 - val_loss: 0.2862 - val_accuracy: 0.9312
Epoch 3/10
20/20 [==============================] - 0s 5ms/step - loss: 0.2190 - accuracy: 0.9578 - val_loss: 0.2469 - val_accuracy: 0.9312
Epoch 4/10
20/20 [==============================] - 0s 5ms/step - loss: 0.1928 - accuracy: 0.9578 - val_loss: 0.2539 - val_accuracy: 0.9312
Epoch 5/10
20/20 [==============================] - 0s 5ms/step - loss: 0.1853 - accuracy: 0.9578 - val_loss: 0.2548 - val_accuracy: 0.9312
Epoch 6/10
20/20 [==============================] - 0s 5ms/step - loss: 0.1832 - accuracy: 0.9578 - val_loss: 0.2553 - val_accuracy: 0.9312
Epoch 7/10
20/20 [==============================] - 0s 5ms/step - loss: 0.1794 - accuracy: 0.9578 - val_loss: 0.2545 - val_accuracy: 0.9312
Epoch 8/10
20/20 [==============================] - 0s 6ms/step - loss: 0.1786 - accuracy: 0.9578 - val_loss: 0.2586 - val_accuracy: 0.9312
Epoch 9/10
20/20 [==============================] - 0s 6ms/step - loss: 0.1756 - accuracy: 0.9578 - val_loss: 0.2589 - val_accuracy: 0.9312
Epoch 10/10
20/20 [==============================] - 0s 7ms/step - loss: 0.1753 - accuracy: 0.9578 - val_loss: 0.2578 - val_accuracy: 0.9312
```

Evaluating Model

```
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {accuracy*100:2f}%")
```

```
7/7 [==============================] - 0s 2ms/step - loss: 0.1692 - accuracy: 0.9650
Test Accuracy: 96.499997%
```

Predicting Model

```
predictions = model.predict(x_test)
predicted_classes = (predictions>0.5).astype(int)
```

```
7/7 [==============================] - 0s 2ms/step
```