1. Identifying Potential Fraudulent Activities:

To detect potential fraudulent activities, we can use an unsupervised machine learning technique called Isolation Forest or One-Class SVM. These algorithms are effective in identifying anomalies in a dataset without the need for labeled examples of fraud.

Python

Example code using Isolation Forest

```
import pandas as pd
```

Load the financial transactions dataset

```
data = pd.read_csv('G:\Placement\Stepchange
Assignment/financial_anomaly_data.csv')
data
```

Select relevant features for anomaly detection

```
features = ['Timestamp', 'TransactionID',
'AccountID','Merchant','Amount', 'TransactionType','Location' ]
X = data[features]

X
```

Removing Null value data (Cleaning)

```
X.isnull().sum()

X =X.dropna()

X.isnull().sum()

X
```

To run Isolation forest model for all the data, memory requirement is too high that's why i considered 30% sample data to identify potential fradulent activities:

```
data = X.sample(frac=0.1)
```

Initialize and fit the Isolation Forest model

```
# Assuming you have a DataFrame named 'transactions' with relevant
columns
from sklearn.ensemble import IsolationForest

# Select relevant features for anomaly detection
```

```
features = ['Amount', 'TransactionType', 'AccountID', 'Timestamp',
'Location', 'Merchant']

# Perform one-hot encoding for categorical variables
transactions_encoded = pd.get_dummies(data[features])

# Initialize and fit the Isolation Forest model
model = IsolationForest(contamination=0.01, max_samples= 'auto')  #
Adjust contamination based on your dataset
model.fit(transactions_encoded)

# Predict anomalies
data['IsAnomaly'] = model.predict(transactions_encoded)

# Flag suspicious transactions
suspicious_transactions = data[data['IsAnomaly']==-1]
```

Predict anomalies

```

print(suspicious_transactions)
```

2. I chose the Isolation Forest approach because it's effective for high-dimensional data, it's less sensitive to outliers, and it performs well on imbalanced datasets. Other methods i consider include One-Class SVM.

3. Features to consider for fraud detection is include transaction amount, time of day, transaction type, location. i can improve my model by normalizing or scaling numerical features, encoding categorical variables, and creating new features like transaction frequency or average transaction amount.

4. Predicting Spend for Transaction Types in June:

```
# Example code for predicting spend using linear regression
from sklearn.linear_model import LinearRegression
# Assuming 'Timestamp' is the column containing timestamps in the
DataFrame
data['Timestamp'] = pd.to_datetime(data['Timestamp'], format='%d-%m-%Y
%H:%M')
# Filter data for February
data = data[data['Timestamp'].dt.month == 6]

# Assuming 'TransactionType' is a categorical variable encoded
numerically
X_train = data[data['Timestamp'].dt.month == 6]['TransactionType']
y_train = data[data['Timestamp'].dt.month == 6]['Amount']
# Encode categorical variables if needed
X_train_encoded = pd.get_dummies(X_train, columns=['TransactionType'])
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Predict spend for June
X_june = data[data['Month'] == 'June'][['TransactionType']]
predicted_spend_june = model.predict(X_june)
```

5. Testing Model Effectiveness:

To test the effectiveness of the model on unseen data, we can use a holdout validation set or perform k-fold cross-validation. Evaluate the model's performance using metrics like precision, recall, F1 score, and AUC-ROC. Additionally, consider monitoring the model's performance over time to ensure it adapts to changing patterns in fraudulent activities.