**Steps to collect the list of malicious addresses and the corresponding fraud categories:**

1. Go to the EtherScamDB GitHub repository (<https://github.com/MrLuit/EtherScamDB>).
2. Navigate to the ‘\_data’ folder and download the ‘scams.yaml’ file.
3. Run the ‘Malicious\_addresses\_category\_extractor.py’ code to generate a list of malicious addresses and the corresponding fraud category. The ‘Malicious\_addresses\_category\_extractor.py’ code performs the following operations:
   1. Opens the YAML file and loads the data
   2. Extracts the addresses and corresponding fraud categories
   3. Replace the ‘Scam’ category with ‘Scamming’ (This is because there exists ‘Scam’ and ‘Scamming’ categories in the ‘scams.yaml’ file. However, both of them represent the same fraud category. Consequently, ‘Scamming’ is used to represent both categories.)
   4. Removes the addresses that do not begin with ‘0x’ as they do not represent a valid Ethereum address
   5. Converts the addresses to lowercase to identify duplicates
   6. Removes duplicate addresses
   7. Saves the list of unique malicious addresses and the corresponding categories to ‘malicious\_addresses\_and\_categories.csv’ file

**Steps to extract transactions for malicious addresses:**

1. Run the ‘Malicious\_transactions\_extractor.py’ code to extract the transactions for malicious addresses and save them in a CSV file. The ‘Malicious\_transactions\_extractor.py’ code performs the following operations:
   1. Defines the function to extract transactions for an address using Etherscan API
   2. Reads the malicious addresses from the ‘malicious\_addresses\_and\_categories.csv’ file
   3. Extracts transactions for each address and appends them to a list
   4. Saves the list to a ‘malicious\_transactions.csv’ file

**Steps to label malicious transactions based on the fraud description:**

1. Run the ‘Malicious\_transactions\_labeler.py’ code to label the malicious transactions based on the fraud categories. The ‘Malicious\_transactions\_labeler.py’ code performs the following operations.
   1. Loads addresses and corresponding fraud categories from 'malicious\_addresses\_and\_categories.csv' file
   2. Makes a copy of the 'malicious\_transactions.csv' file to store the labeled dataset. The copied file is named ‘malicious\_transactions\_labeled.csv’
   3. Updates 'malicious\_transactions\_labeled.csv' with the 'Fraud' column to store the fraud category labels
   4. For each transaction, the code checks the sender and receiver addresses. If one of the sender or receiver addresses is present in the 'malicious\_addresses\_and\_categories.csv' file, the fraud category for that address is assigned in the ‘Fraud’ column. If both the addresses are present in the 'malicious\_addresses\_and\_categories.csv' file, then ‘multi-category’ is assigned to the ‘Fraud’ column in the case that category for both addresses is different, else, the category is assigned. If none of the addresses are present, then ‘Null’ is assigned.

**Steps to preprocess labeled malicious:**

1. Run the ‘Malicious\_transactions\_preprocessor.py’ code to preprocess labeled malicious transactions. The ‘Malicious\_transactions\_preprocessor.py’ code performs the following operations.
   1. Loads labeled malicious transactions from ‘malicious\_transactions\_labeled.csv’ file
   2. Removes the transactions where iserror = 1 or value = 0
   3. Saves the preprocessed malicious transactions in ‘malicious\_transactions\_preprocessed\_labeled.csv’ file