```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files Fraud Dete… Dataset.csv
```

- **Fraud Detection Dataset.csv**(text/csv) - 3692523 bytes, last modified: 4/2/2025 - 100% done

Saving Fraud Detection Dataset.csv to Fraud Detection Dataset.csv

```
import pandas as pd
import numpy as np

# Load the CSV file (update filename accordingly)
df = pd.read_csv("Fraud Detection Dataset.csv")

# Display first few rows
print(df.head())

```

```
  Transaction_ID  User_ID  Transaction_Amount Transaction_Type  \
0             T1     4174             1292.76   ATM Withdrawal
1             T2     4507             1554.58   ATM Withdrawal
2             T3     1860             2395.02   ATM Withdrawal
3             T4     2294              100.10     Bill Payment
4             T5     2130             1490.50      POS Payment

   Time_of_Transaction Device_Used       Location  \
0                 16.0      Tablet  San Francisco
1                 13.0      Mobile       New York
2                  NaN      Mobile            NaN
3                 15.0     Desktop        Chicago
4                 19.0      Mobile  San Francisco

   Previous_Fraudulent_Transactions  Account_Age  \
0                                 0          119
1                                 4           79
2                                 3          115
3                                 4            3
4                                 2           57

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0
```

```
# Extract numerical columns as NumPy arrays
transaction_amounts = df["Transaction_Amount"].fillna(0).to_numpy()
fraud_labels = df["Fraudulent"].to_numpy()
account_ages = df["Account_Age"].to_numpy()

```

```
# Pandas Series Object
transaction_series = pd.Series(transaction_amounts, name="Transaction_Amount")
fraud_series = pd.Series(fraud_labels, name="Fraudulent")
```

```
# Pandas DataFrame Object
df_fraud = pd.DataFrame({
    "Transaction_Amount": transaction_series,
    "Fraudulent": fraud_series,
    "Account_Age": account_ages
})
```

```
# Data Indexing and Selecting for Series and DataFrames
indexed_transaction = transaction_series[5]  # Select a specific value
sliced_dataframe = df_fraud.iloc[:10]  # Select first 10 rows
```

```
# Universal Functions for Index Preservation
sqrt_transaction = np.sqrt(df_fraud["Transaction_Amount"])
log_transaction = np.log1p(df_fraud["Transaction_Amount"])
df_fraud["Sqrt_Transaction"] = sqrt_transaction
df_fraud["Log_Transaction"] = log_transaction
```

```
# Index Alignment and Operations between Series and DataFrames
df_fraud["Normalized_Transaction"] = (df_fraud["Transaction_Amount"] - df_fraud["Transaction_Amount"].mean()) / df_fraud["Transactio
```

```
# Handling Missing Data
df_fraud.fillna(0, inplace=True)  # Fill missing values with 0
```

```
df_fraud.dropna(inplace=True)  # Drop rows with missing values
```

```
# Operating on Null Values
missing_values = df_fraud.isnull().sum()  # Count missing values per column
```

```
# Hierarchical Indexing
df_fraud.set_index(["Fraudulent", "Account_Age"], inplace=True)
```

```
# Reset Index
df_fraud.reset_index(inplace=True)
```

```
# Additional Pandas Functions
# Sorting Data
sorted_df = df_fraud.sort_values(by="Transaction_Amount", ascending=False)
```

```
# Grouping Data
grouped_data = df_fraud.groupby("Fraudulent")["Transaction_Amount"].mean()
```

```
# Aggregation Functions
aggregated_data = df_fraud.agg({
    "Transaction_Amount": ["mean", "sum", "min", "max", "std"],
    "Account_Age": ["mean", "count"]
})
```

```
# Applying Functions to Data
scaled_transactions = df_fraud["Transaction_Amount"].apply(lambda x: x / 1000)
df_fraud["Scaled_Transactions"] = scaled_transactions
```

```
# Filtering Data
high_value_transactions = df_fraud[df_fraud["Transaction_Amount"] > 2000]
```

```
# Merging DataFrames
dummy_data = pd.DataFrame({"Fraudulent": [0, 1], "Fraud_Description": ["Legit", "Fraud"]})
merged_df = df_fraud.merge(dummy_data, on="Fraudulent", how="left")
```

```
# Pivot Tables
pivot_table = df_fraud.pivot_table(values="Transaction_Amount", index="Fraudulent", columns="Account_Age", aggfunc="mean")
```

```
# One-Hot Encoding for Categorical Data
if "Transaction_Type" in df.columns:
    df_fraud = pd.get_dummies(df, columns=["Transaction_Type"], drop_first=True)
```

```
# Descriptive Statistics
summary_statistics = df_fraud.describe()
```

```
# Rolling and Expanding Window Functions
df_fraud["Rolling_Mean"] = df_fraud["Transaction_Amount"].rolling(window=5).mean()
df_fraud["Expanding_Mean"] = df_fraud["Transaction_Amount"].expanding().mean()
```

```
# Duplicates Handling
duplicates_count = df_fraud.duplicated().sum()
df_fraud.drop_duplicates(inplace=True)
```

```
# Value Counts
fraud_counts = df_fraud["Fraudulent"].value_counts()
```

```
# Memory Usage Optimization
df_fraud.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51000 entries, 0 to 50999
Data columns (total 19 columns):
```

```
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Transaction_ID                   51000 non-null  object
 1   User_ID                          51000 non-null  int64
 2   Transaction_Amount               48480 non-null  float64
 3   Time_of_Transaction              48448 non-null  float64
 4   Device_Used                      48527 non-null  object
 5   Location                         48453 non-null  object
 6   Previous_Fraudulent_Transactions 51000 non-null  int64
 7   Account_Age                      51000 non-null  int64
 8   Number_of_Transactions_Last_24H  51000 non-null  int64
 9   Payment_Method                   48531 non-null  object
 10  Fraudulent                       51000 non-null  int64
 11  Transaction_Type_Bank Transfer   51000 non-null  bool
 12  Transaction_Type_Bill Payment    51000 non-null  bool
 13  Transaction_Type_Online Purchase 51000 non-null  bool
 14  Transaction_Type_POS Payment     51000 non-null  bool
 15  Sqrt_Transaction                 48480 non-null  float64
 16  Log_Transaction                  48480 non-null  float64
 17  Rolling_Mean                     39623 non-null  float64
 18  Expanding_Mean                   51000 non-null  float64
dtypes: bool(4), float64(6), int64(5), object(4)
memory usage: 16.8 MB
```

```
# Combining Datasets
# Creating a sample DataFrame to demonstrate concatenation
new_data = pd.DataFrame({"Transaction_Amount": [500, 1500], "Fraudulent": [1, 0], "Account_Age": [2, 5]})
```

```
# Concatenation
concatenated_df = pd.concat([df_fraud, new_data], ignore_index=True)

```

```
# Append (deprecated, but similar to concat)
appended_df = pd.concat([df_fraud, new_data], ignore_index=True)

```

```
# Merge with another DataFrame
merged_df_new = df_fraud.merge(dummy_data, on="Fraudulent", how="inner")

```

```
# Joins (Using set_index for demonstration)
df_fraud.set_index("Fraudulent", inplace=True)
joined_df = df_fraud.join(dummy_data.set_index("Fraudulent"), on="Fraudulent")
df_fraud.reset_index(inplace=True)

```

```
# ------------------------- Categorical Data Handling -------------------------
if 'Transaction_Type' in df.columns:
    df['Transaction_Type'] = df['Transaction_Type'].astype('category')
    category_codes = df['Transaction_Type'].cat.codes
    df['Transaction_Type_Code'] = category_codes
print("Category Codes:\n", df[['Transaction_Type', 'Transaction_Type_Code']].head())

```

```
Category Codes:
   Transaction_Type  Transaction_Type_Code
0   ATM Withdrawal                      0
1   ATM Withdrawal                      0
2   ATM Withdrawal                      0
3     Bill Payment                      2
4      POS Payment                      4
```

```
# ------------------------- MultiIndexing -------------------------
if 'Fraudulent' in df.columns and 'Transaction_Amount' in df.columns:
    df.set_index(['Fraudulent', 'Transaction_Amount'], inplace=True)
    print("\nMultiIndex DataFrame:\n", df.head())
    df.reset_index(inplace=True)
```

```
MultiIndex DataFrame:
                               Transaction_ID  User_ID Transaction_Type  \
Fraudulent Transaction_Amount
0          1292.76                         T1     4174   ATM Withdrawal
           1554.58                         T2     4507   ATM Withdrawal
           2395.02                         T3     1860   ATM Withdrawal
           100.10                          T4     2294     Bill Payment
           1490.50                         T5     2130      POS Payment

                               Time_of_Transaction Device_Used       Location  \
Fraudulent Transaction_Amount
0          1292.76                            16.0      Tablet  San Francisco
```

```
           1554.58                          13.0      Mobile       New York
           2395.02                           NaN      Mobile            NaN
           100.10                           15.0     Desktop        Chicago
           1490.50                          19.0      Mobile  San Francisco

                                Previous_Fraudulent_Transactions  Account_Age  \
Fraudulent Transaction_Amount
0          1292.76                                             0          119
           1554.58                                             4           79
           2395.02                                             3          115
           100.10                                              4            3
           1490.50                                             2           57

                                Number_of_Transactions_Last_24H Payment_Method  \
Fraudulent Transaction_Amount
0          1292.76                                           13     Debit Card
           1554.58                                            3    Credit Card
           2395.02                                            9            NaN
           100.10                                             4            UPI
           1490.50                                            7    Credit Card

                                Transaction_Type_Code
Fraudulent Transaction_Amount
0          1292.76                                  0
           1554.58                                  0
           2395.02                                  0
           100.10                                   2
           1490.50                                  4
```

```
# ------------------------- Text Data Handling -------------------------
if 'Transaction_Description' in df.columns:
    df['Text_Lower'] = df['Transaction_Description'].str.lower()
    df['Contains_Fraud'] = df['Transaction_Description'].str.contains('fraud', case=False, na=False)
    df['Replaced_Text'] = df['Transaction_Description'].str.replace('fraud', '***', regex=True)
    print("\nText Data Operations:\n", df[['Transaction_Description', 'Text_Lower', 'Contains_Fraud', 'Replaced_Text']].head())

```

```
# ------------------------- Statistical & Analytical Functions -------------------------
if 'Transaction_Amount' in df.columns:
    median_values = df['Transaction_Amount'].median()
    mode_values = df['Transaction_Amount'].mode().tolist()
    cumsum_values = df['Transaction_Amount'].cumsum().tolist()
    diff_values = df['Transaction_Amount'].diff().tolist()
    print(f"\nMedian: {median_values}, Mode: {mode_values}, Cumulative Sum: {cumsum_values[:5]}, Differences: {diff_values[:5]}")

```

```
Median: 2524.1000000000004, Mode: [49997.8], Cumulative Sum: [1292.76, 2847.34, 5242.360000000001, 5342.460000000001, 6832.960000000
```

```
# ------------------------- Reshaping Data -------------------------
if 'Fraudulent' in df.columns and 'Transaction_Amount' in df.columns:
    melted_df = df.melt(id_vars=['Fraudulent'], value_vars=['Transaction_Amount'])
    print("\nMelted DataFrame:\n", melted_df.head())
    pivot_df = df.pivot_table(index='Fraudulent', columns='Transaction_Amount', aggfunc='count')
    print("\nPivot DataFrame:\n", pivot_df.head())

```

```
Melted DataFrame:
    Fraudulent            variable    value
0           0  Transaction_Amount  1292.76
1           0  Transaction_Amount  1554.58
2           0  Transaction_Amount  2395.02
3           0  Transaction_Amount   100.10
4           0  Transaction_Amount  1490.50

Pivot DataFrame:
                    Account_Age                                             \
Transaction_Amount    5.03     5.04     5.08     5.23     5.47     5.49
Fraudulent
0                        1.0      1.0      1.0      1.0      1.0      1.0
1                        NaN      NaN      NaN      NaN      NaN      NaN

                                                  ...  User_ID           \
Transaction_Amount 5.52     5.55     5.63     5.65      ... 4998.95  4999.09
Fraudulent                                        ...
0                       1.0      1.0      1.0      1.0  ...      1.0      1.0
1                       NaN      NaN      NaN      NaN  ...      NaN      NaN

                                                                        \
Transaction_Amount 4999.12  4999.16  4999.21  4999.38  4999.45  4999.70
Fraudulent
0                       1.0      1.0      1.0      1.0      1.0      1.0
1                       NaN      NaN      NaN      NaN      NaN      NaN
```

```
Transaction_Amount 4999.78  49997.80
Fraudulent
0                      1.0    476.0
1                      NaN     32.0

[2 rows x 493031 columns]
```

```
# ------------------------- Performance Optimization -------------------------
memory_usage = df.memory_usage(deep=True)
if 'Transaction_Amount' in df.columns:
    df['Transaction_Amount'] = df['Transaction_Amount'].astype('float32')
optimized_memory_usage = df.memory_usage(deep=True)
print("\nOriginal Memory Usage:\n", memory_usage)
print("\nOptimized Memory Usage:\n", optimized_memory_usage)

```

```
Original Memory Usage:
 Index                                132
Fraudulent                         408000
Transaction_Amount                 408000
Transaction_ID                    3201668
User_ID                            408000
Transaction_Type                    51522
Time_of_Transaction                408000
Device_Used                       3164372
Location                          3230433
Previous_Fraudulent_Transactions   408000
Account_Age                        408000
Number_of_Transactions_Last_24H    408000
Payment_Method                    3276808
Transaction_Type_Code               51000
dtype: int64

Optimized Memory Usage:
 Index                                132
Fraudulent                         408000
Transaction_Amount                 204000
Transaction_ID                    3201668
User_ID                            408000
Transaction_Type                    51522
Time_of_Transaction                408000
Device_Used                       3164372
Location                          3230433
Previous_Fraudulent_Transactions   408000
Account_Age                        408000
Number_of_Transactions_Last_24H    408000
Payment_Method                    3276808
Transaction_Type_Code               51000
dtype: int64
```

```
# Printing results
print("Indexed Transaction:", indexed_transaction)
print()
print("Sliced DataFrame:\n", sliced_dataframe)
print()
if "Transaction_Amount" in df_fraud.columns:
    df_fraud["Sqrt_Transaction"] = np.sqrt(df_fraud["Transaction_Amount"].clip(lower=0))  # Prevent negative values
    df_fraud["Log_Transaction"] = np.log1p(df_fraud["Transaction_Amount"].clip(lower=0.0001))  # Avoid log(0) error
else:
    print("Error: 'Transaction_Amount' column is missing from the dataset.")
print()
print("Log Transactions:\n", df_fraud["Log_Transaction"].head())
print()

```

```
Indexed Transaction: 2372.04

Sliced DataFrame:
    Transaction_Amount  Fraudulent  Account_Age
0              1292.76           0          119
1              1554.58           0           79
2              2395.02           0          115
3               100.10           0            3
4              1490.50           0           57
5              2372.04           0           96
6               544.81           1            6
7               635.75           0           13
8              2318.87           0          110
9              3656.17           0           66


Log Transactions:
 0    7.165308
```

```
1    7.349604
2    7.781564
3    4.616110
4    7.307538
Name: Log_Transaction, dtype: float64
```

```
print("Hierarchical Indexed DataFrame:\n", df_fraud.head())
print()
print("Sorted DataFrame:\n", sorted_df.head())
print()

```

```
Hierarchical Indexed DataFrame:
   Transaction_ID  User_ID  Transaction_Amount  Time_of_Transaction  \
0             T1     4174             1292.76                 16.0
1             T2     4507             1554.58                 13.0
2             T3     1860             2395.02                  NaN
3             T4     2294              100.10                15.0
4             T5     2130             1490.50                 19.0

  Device_Used       Location  Previous_Fraudulent_Transactions  Account_Age  \
0      Tablet  San Francisco                                 0          119
1      Mobile       New York                                 4           79
2      Mobile            NaN                                 3          115
3     Desktop        Chicago                                 4            3
4      Mobile  San Francisco                                 2           57

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent  \
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0

   Transaction_Type_Bank Transfer  Transaction_Type_Bill Payment  \
0                           False                          False
1                           False                          False
2                           False                          False
3                           False                           True
4                           False                          False

   Transaction_Type_Online Purchase  Transaction_Type_POS Payment  \
0                             False                          False
1                             False                          False
2                             False                          False
3                             False                          False
4                             False                           True

   Sqrt_Transaction  Log_Transaction
0         35.954972         7.165308
1         39.428163         7.349604
2         48.938942         7.781564
3         10.004999         4.616110
4         38.606994         7.307538

Sorted DataFrame:
        Fraudulent  Account_Age  Transaction_Amount  Sqrt_Transaction  \
42536           0          114             49997.8        223.601878
21838           0           36             49997.8        223.601878
42499           0           23             49997.8        223.601878
8588            0            9             49997.8        223.601878
50316           0           74             49997.8        223.601878

       Log_Transaction  Normalized_Transaction
42536        10.819754                9.505153
21838        10.819754                9.505153
42499        10.819754                9.505153
8588         10.819754                9.505153
50316        10.819754                9.505153
```

```
print("Grouped Data (Average Transaction Amount by Fraudulent):\n", grouped_data)
print()
print("Aggregated Data:\n", aggregated_data)
print()
print("Filtered High Value Transactions:\n", high_value_transactions.head())
print()
```

```
Grouped Data (Average Transaction Amount by Fraudulent):
 Fraudulent
0    2842.253736
1    2963.070072
Name: Transaction_Amount, dtype: float64

Aggregated Data:
        Transaction_Amount   Account_Age
mean         2.848200e+03     60.033902
```

```
sum          1.452582e+08           NaN
min          0.000000e+00           NaN
max          4.999780e+04           NaN
std          4.960425e+03           NaN
count                 NaN  51000.000000

Filtered High Value Transactions:
     Fraudulent  Account_Age  Transaction_Amount  Sqrt_Transaction  \
2             0          115             2395.02         48.938942
5             0           96             2372.04         48.703593
8             0          110             2318.87         48.154647
9             0           66             3656.17         60.466272
11            0           68             2733.84         52.286136

    Log_Transaction  Normalized_Transaction  Scaled_Transactions
2          7.781564               -0.091359              2.39502
5          7.771927               -0.095992              2.37204
8          7.749266               -0.106711              2.31887
9          8.204445                0.162883              3.65617
11         7.913828               -0.023054              2.73384
```

```
print("Missing Values Count:\n", missing_values)
print()

print("Merged DataFrame:\n", merged_df.head())
print()

```

```
Missing Values Count:
 Transaction_Amount        0
Fraudulent                 0
Account_Age                0
Sqrt_Transaction           0
Log_Transaction            0
Normalized_Transaction     0
dtype: int64

Merged DataFrame:
    Fraudulent  Account_Age  Transaction_Amount  Sqrt_Transaction  \
0           0          119             1292.76         35.954972
1           0           79             1554.58         39.428163
2           0          115             2395.02         48.938942
3           0            3              100.10         10.004999
4           0           57             1490.50         38.606994

   Log_Transaction  Normalized_Transaction  Scaled_Transactions  \
0         7.165308               -0.313570              1.29276
1         7.349604               -0.260788              1.55458
2         7.781564               -0.091359              2.39502
3         4.616110               -0.554005              0.10010
4         7.307538               -0.273706              1.49050

  Fraud_Description
0             Legit
1             Legit
2             Legit
3             Legit
4             Legit
```

```
print("Pivot Table:\n", pivot_table)
print()
print("Summary Statistics:\n", summary_statistics)
print()
```

```
Pivot Table:
 Account_Age          1            2            3            4            5    \
Fraudulent
0           2727.967783  2728.618032  2866.506854  2487.000644  3199.817831
1           2381.866190  3153.974483  3482.268621  2415.775500  1868.962143

Account_Age          6            7            8            9            10   \
Fraudulent
0           3271.388465  3010.721930  2585.425105  2972.943933  2590.983580
1           3839.535263  4715.352857  2248.062273  2175.485000  2286.867037

Account_Age  ...          110          111          112          113  \
Fraudulent   ...
0            ...  2472.402817  2782.785286  2359.034518  2815.939610
1            ...  2355.346667  4029.612414  2191.338000  2050.957391

Account_Age          114          115          116          117          118  \
Fraudulent
0           2706.844699  3164.928138  2581.058728  2872.109898  2759.841293
1           2634.861538  6021.716154  2989.191250  4996.080000  2968.356800
```

```
Account_Age          119
Fraudulent
0            2765.426563
1            2322.585652

[2 rows x 119 columns]

Summary Statistics:
             User_ID  Transaction_Amount  Time_of_Transaction  \
count  51000.000000        48480.000000         48448.000000
mean    3005.110176         2996.249784            11.488400
std     1153.121107         5043.932555             6.922954
min     1000.000000            5.030000             0.000000
25%     2007.000000         1270.552500             5.000000
50%     2996.000000         2524.100000            12.000000
75%     4006.000000         3787.240000            17.000000
max     4999.000000        49997.800000            23.000000

       Previous_Fraudulent_Transactions   Account_Age  \
count                      51000.000000  51000.000000
mean                           1.995725     60.033902
std                            1.415150     34.384131
min                            0.000000      1.000000
25%                            1.000000     30.000000
50%                            2.000000     60.000000
75%                            3.000000     90.000000
max                            4.000000    119.000000

       Number_of_Transactions_Last_24H    Fraudulent
count                     51000.000000  51000.000000
mean                          7.495588      0.049216
std                           4.020080      0.216320
min                           1.000000      0.000000
25%                           4.000000      0.000000
50%                           7.000000      0.000000
75%                          11.000000      0.000000
max                          14.000000      1.000000
```

```
print("Rolling Mean Transactions:\n", df_fraud["Rolling_Mean"].head())
print("Expanding Mean Transactions:\n", df_fraud["Expanding_Mean"].head())
print("Duplicate Rows Removed Count:\n", duplicates_count)
print("Fraud Count Distribution:\n", fraud_counts)

```

```
Rolling Mean Transactions:
 0         NaN
1         NaN
2         NaN
3         NaN
4    1366.592
Name: Rolling_Mean, dtype: float64
Expanding Mean Transactions:
 0    1292.760000
1    1423.670000
2    1747.453333
3    1335.615000
4    1366.592000
Name: Expanding_Mean, dtype: float64
Duplicate Rows Removed Count:
 0
Fraud Count Distribution:
 Fraudulent
0    48490
1     2510
Name: count, dtype: int64
```

```
print("Concatenated DataFrame:\n", concatenated_df.head())

```

```
Concatenated DataFrame:
   Transaction_ID  User_ID  Transaction_Amount  Time_of_Transaction  \
0              T1   4174.0             1292.76                 16.0
1              T2   4507.0             1554.58                 13.0
2              T3   1860.0             2395.02                  NaN
3              T4   2294.0              100.10                 15.0
4              T5   2130.0             1490.50                 19.0

  Device_Used       Location  Previous_Fraudulent_Transactions  Account_Age  \
0      Tablet  San Francisco                               0.0          119
1      Mobile       New York                               4.0           79
2      Mobile            NaN                               3.0          115
3     Desktop        Chicago                               4.0            3
4      Mobile  San Francisco                               2.0           57

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent  \
0                             13.0     Debit Card           0
1                              3.0    Credit Card           0
2                              9.0            NaN           0
3                              4.0            UPI           0
```

```
4                            7.0    Credit Card           0

  Transaction_Type_Bank Transfer Transaction_Type_Bill Payment  \
0                          False                         False
1                          False                         False
2                          False                         False
3                          False                          True
4                          False                         False

  Transaction_Type_Online Purchase Transaction_Type_POS Payment  \
0                            False                        False
1                            False                        False
2                            False                        False
3                            False                        False
4                            False                         True

   Sqrt_Transaction  Log_Transaction  Rolling_Mean  Expanding_Mean
0         35.954972         7.165308           NaN     1292.760000
1         39.428163         7.349604           NaN     1423.670000
2         48.938942         7.781564           NaN     1747.453333
3         10.004999         4.616110           NaN     1335.615000
4         38.606994         7.307538      1366.592     1366.592000
```

```
print("Merged DataFrame (New):\n", merged_df_new.head())

```

```
Merged DataFrame (New):
   Transaction_ID  User_ID  Transaction_Amount  Time_of_Transaction  \
0             T1     4174             1292.76                 16.0
1             T2     4507             1554.58                 13.0
2             T3     1860             2395.02                  NaN
3             T4     2294              100.10                 15.0
4             T5     2130             1490.50                 19.0

  Device_Used       Location  Previous_Fraudulent_Transactions  Account_Age  \
0      Tablet  San Francisco                                 0          119
1      Mobile       New York                                 4           79
2      Mobile            NaN                                 3          115
3     Desktop        Chicago                                 4            3
4      Mobile  San Francisco                                 2           57

   Number_of_Transactions_Last_24H Payment_Method  Fraudulent  \
0                               13     Debit Card           0
1                                3    Credit Card           0
2                                9            NaN           0
3                                4            UPI           0
4                                7    Credit Card           0

   Transaction_Type_Bank Transfer  Transaction_Type_Bill Payment  \
0                           False                         False
1                           False                         False
2                           False                         False
3                           False                          True
4                           False                         False

   Transaction_Type_Online Purchase  Transaction_Type_POS Payment  \
0                             False                         False
1                             False                         False
2                             False                         False
3                             False                         False
4                             False                          True

   Sqrt_Transaction  Log_Transaction  Rolling_Mean  Expanding_Mean  \
0         35.954972         7.165308           NaN     1292.760000
1         39.428163         7.349604           NaN     1423.670000
2         48.938942         7.781564           NaN     1747.453333
3         10.004999         4.616110           NaN     1335.615000
4         38.606994         7.307538      1366.592     1366.592000

  Fraud_Description
0             Legit
1             Legit
2             Legit
3             Legit
4             Legit
```

```
print("Joined DataFrame:\n", joined_df.head())

```

```
Joined DataFrame:
             Transaction_ID  User_ID  Transaction_Amount  Time_of_Transaction  \
Fraudulent
0                        T1     4174             1292.76                 16.0
0                        T2     4507             1554.58                 13.0
0                        T3     1860             2395.02                  NaN
0                        T4     2294              100.10                 15.0
0                        T5     2130             1490.50                 19.0
```

Fraudulent	Device_Used	Location	Previous_Fraudulent_Transactions	Account_Age	Number_of_Transactions_Last_24H	Payment_Method	Transaction_Type_Bank Transfer	Transaction_Type_Bill Payment	Transaction_Type_Online Purchase	Transaction_Type_POS Payment	Sqrt_Transaction	Log_Transaction	Rolling_Mean	Expanding_Mean	Fraud_Description
0	Tablet	San Francisco	0	119	13	Debit Card	False	False	False	False	35.954972	7.165308	NaN	1292.760000	Legit
0	Mobile	New York	4	79	3	Credit Card	False	False	False	False	39.428163	7.349604	NaN	1423.670000	Legit
0	Mobile	NaN	3	115	9	NaN	False	False	False	False	48.938942	7.781564	NaN	1747.453333	Legit
0	Desktop	Chicago	4	3	4	UPI	False	True	False	False	10.004999	4.616110	NaN	1335.615000	Legit
0	Mobile	San Francisco	2	57	7	Credit Card	False	False	False	True	38.606994	7.307538	1366.592	1366.592000	Legit