```
import kagglehub
path = kagglehub.dataset_download("cristobaltudela/credit-card-transaction-l
```

```
n outdated `kagglehub` version (installed: 0.3.13), please consider upgrading
e.com/api/v1/datasets/download/cristobaltudela/credit-card-transaction-legiti
0<00:00, 82.5MB/s]Extracting files...
```

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

Choose Files fraud_detection.csv

fraud_detection.csv(text/csv) - 324710 bytes, last modified: 1/30/2026 - 100% done

```
Saving fraud_detection.csv to fraud_detection.csv
```

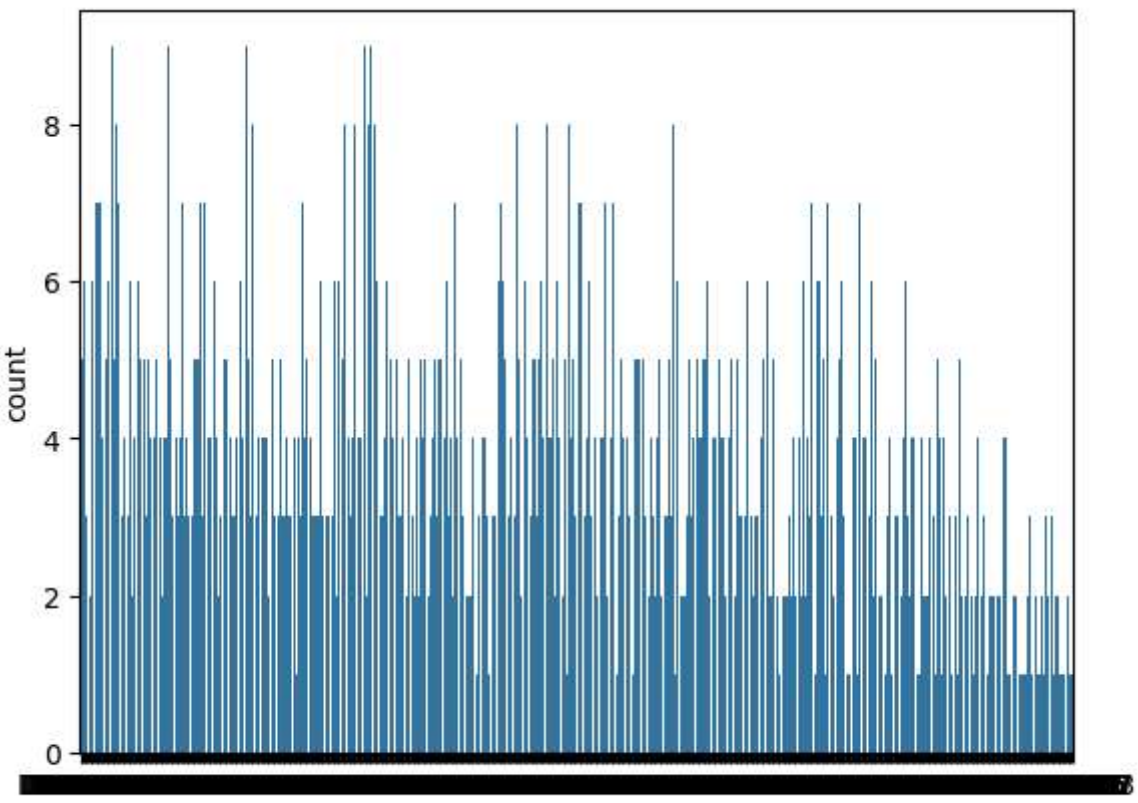
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

New section

```
#Step 2. Load data
df = pd.read_csv("/content/fraud_detection.csv")
df.head()
#Step 3. Basic checks
df.shape
print(df.columns)
#df['TransactionID'].value_counts() # This line caused the error. After insp
```

```
Index(['TransactionID', 'AccountID', 'TransactionAmount', 'TransactionDate',
       'TransactionType', 'Location', 'DeviceID', 'IP Address', 'MerchantID',
       'Channel', 'CustomerAge', 'CustomerOccupation', 'TransactionDuration',
       'LoginAttempts', 'AccountBalance', 'PreviousTransactionDate'],
      dtype='object')
```

```
#Step 4. Data understanding

#Check class imbalance:

sns.countplot(x='DeviceID', data=df) # 'Class' column not found. Please repl
plt.show()
```




```
print(df.columns)
```

```
Index(['TransactionID', 'AccountID', 'TransactionAmount', 'TransactionDate',
       'TransactionType', 'Location', 'DeviceID', 'IP Address', 'MerchantID',
       'Channel', 'CustomerAge', 'CustomerOccupation', 'TransactionDuration',
       'LoginAttempts', 'AccountBalance', 'PreviousTransactionDate'],
      dtype='object')
```

```
# Step-by-step EDA and Outlier Handling on the uploaded dataset
# 1. Basic inspection
shape = df.shape


shape, head
```

```
((2512, 16),
   TransactionID AccountID  TransactionAmount   TransactionDate  \
 0      TX000001   AC00128              14.09   4/11/2023 16:29
 1      TX000002   AC00455             376.24   6/27/2023 16:44
 2      TX000003   AC00019             126.29   7/10/2023 18:16
 3      TX000004   AC00070             184.50    5/5/2023 16:32
 4      TX000005   AC00411              13.45  10/16/2023 17:51

   TransactionType   Location DeviceID      IP Address MerchantID Channel  \
 0           Debit  San Diego  D000380  162.198.218.92       M015     ATM
 1           Debit    Houston  D000051     13.149.61.4       M052     ATM
 2           Debit       Mesa  D000235  215.97.143.157       M009  Online
```

```
 3           Debit    Raleigh  D000187  200.13.225.150       M002  Online
 4          Credit    Atlanta  D000308    65.164.3.100       M091  Online

   CustomerAge CustomerOccupation  TransactionDuration  LoginAttempts  \
 0          70             Doctor                   81              1
 1          68             Doctor                  141              1
 2          19            Student                   56              1
 3          26            Student                   25              1
 4          26            Student                  198              1

   AccountBalance PreviousTransactionDate
 0        5112.21          11/4/2024 8:08
 1       13758.91          11/4/2024 8:09
 2        1122.35          11/4/2024 8:07
 3        8569.06          11/4/2024 8:09
 4        7429.40          11/4/2024 8:06  )
```

```
info = df.info()
head = df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2512 entries, 0 to 2511
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   TransactionID            2512 non-null   object
 1   AccountID                2512 non-null   object
 2   TransactionAmount        2512 non-null   float64
 3   TransactionDate          2512 non-null   object
 4   TransactionType          2512 non-null   object
 5   Location                 2512 non-null   object
 6   DeviceID                 2512 non-null   object
 7   IP Address               2512 non-null   object
 8   MerchantID               2512 non-null   object
 9   Channel                  2512 non-null   object
 10  CustomerAge              2512 non-null   int64
 11  CustomerOccupation       2512 non-null   object
 12  TransactionDuration      2512 non-null   int64
 13  LoginAttempts            2512 non-null   int64
 14  AccountBalance           2512 non-null   float64
 15  PreviousTransactionDate  2512 non-null   object
dtypes: float64(2), int64(3), object(11)
memory usage: 314.1+ KB
```

```
# 2. Descriptive statistics
print("\nDescriptive Statistics:")
print(df.describe(include="all"))
```

```

Descriptive Statistics:
       TransactionID AccountID  TransactionAmount   TransactionDate  \
count           2512      2512        2512.000000              2512
unique          2512       495                 NaN              2405
top         TX002496   AC00460                 NaN  11/20/2023 16:29
```

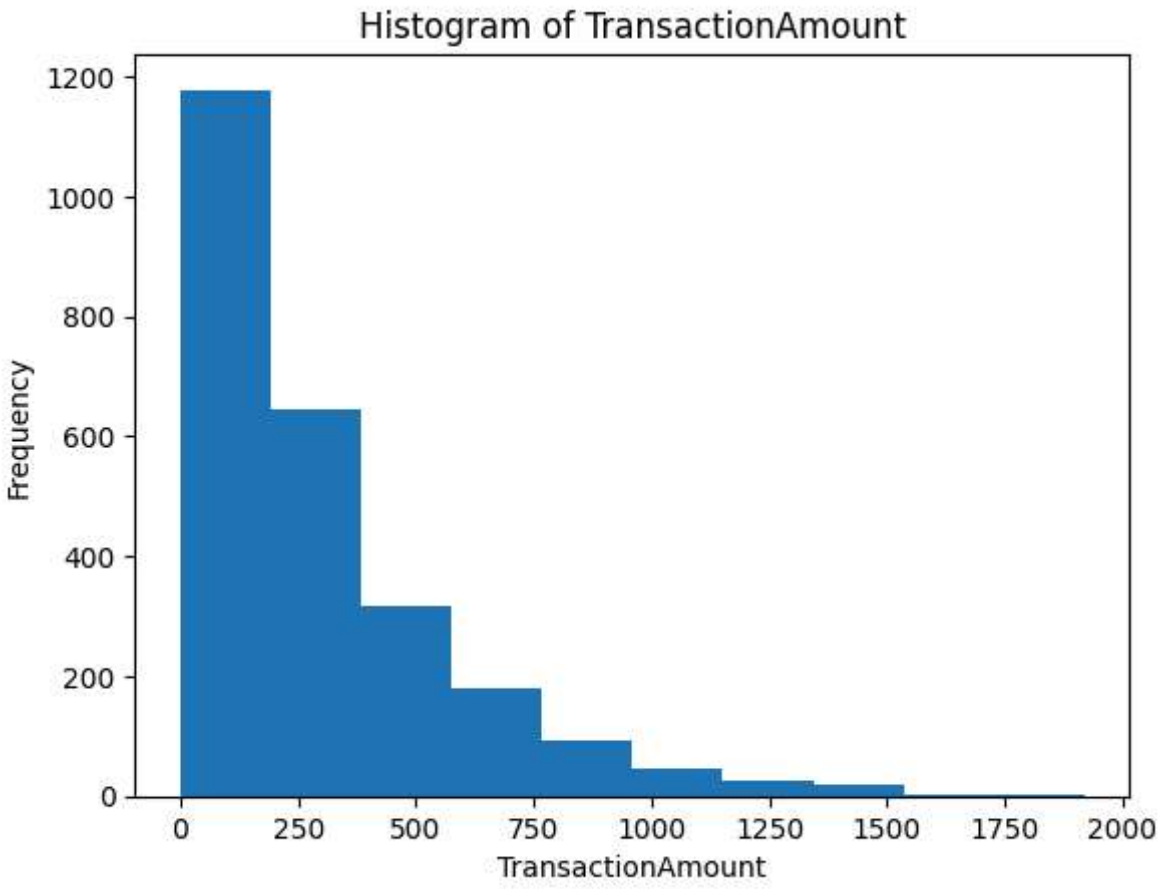
```
freq               1        12                NaN                3
mean             NaN       NaN         297.593778              NaN
std              NaN       NaN         291.946243              NaN
min              NaN       NaN           0.260000              NaN
25%              NaN       NaN          81.885000              NaN
50%              NaN       NaN         211.140000              NaN
75%              NaN       NaN         414.527500              NaN
max              NaN       NaN        1919.110000              NaN

       TransactionType    Location DeviceID      IP Address MerchantID  \
count             2512        2512     2512            2512       2512
unique               2          43      681             592        100
top              Debit  Fort Worth  D000548  200.136.146.93       M026
freq              1944          70        9              13         45
mean               NaN         NaN      NaN             NaN        NaN
std                NaN         NaN      NaN             NaN        NaN
min                NaN         NaN      NaN             NaN        NaN
25%                NaN         NaN      NaN             NaN        NaN
50%                NaN         NaN      NaN             NaN        NaN
75%                NaN         NaN      NaN             NaN        NaN
max                NaN         NaN      NaN             NaN        NaN

       Channel  CustomerAge CustomerOccupation  TransactionDuration  \
count     2512  2512.000000               2512          2512.000000
unique       3          NaN                  4                  NaN
top     Branch          NaN            Student                  NaN
freq       868          NaN                657                  NaN
mean       NaN    44.673965                NaN           119.643312
std        NaN    17.792198                NaN            69.963757
min        NaN    18.000000                NaN            10.000000
25%        NaN    27.000000                NaN            63.000000
50%        NaN    45.000000                NaN           112.500000
75%        NaN    59.000000                NaN           161.000000
max        NaN    80.000000                NaN           300.000000

        LoginAttempts  AccountBalance PreviousTransactionDate
count     2512.000000     2512.000000                    2512
unique            NaN             NaN                       7
top               NaN             NaN         11/4/2024 8:07
freq              NaN             NaN                     435
mean         1.124602     5114.302966                     NaN
std          0.602662     3900.942499                     NaN
min          1.000000      101.250000                     NaN
25%          1.000000     1504.370000                     NaN
50%          1.000000     4735.510000                     NaN
75%          1.000000     7678.820000                     NaN
max          5.000000    14977.990000                     NaN
```

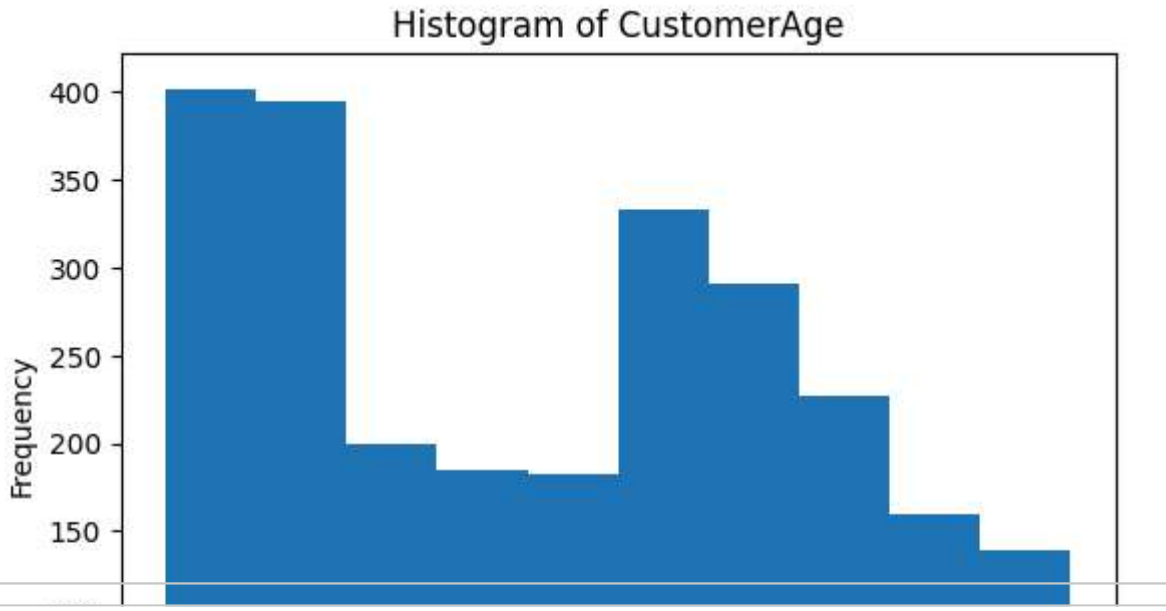
```
# 3. Missing value percentage
missing_pct = df.isnull().mean() * 100
print("\nMissing Value Percentage (%):")
print(missing_pct)
```

```
Missing Value Percentage (%):
```

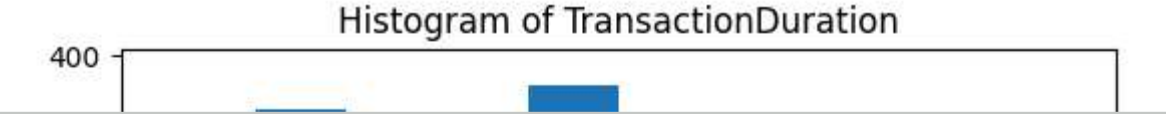
```
TransactionID              0.0
AccountID                  0.0
TransactionAmount          0.0
TransactionDate            0.0
TransactionType            0.0
Location                   0.0
DeviceID                   0.0
IP Address                 0.0
MerchantID                 0.0
Channel                    0.0
CustomerAge                0.0
CustomerOccupation         0.0
TransactionDuration        0.0
LoginAttempts              0.0
AccountBalance             0.0
PreviousTransactionDate    0.0
dtype: float64
```

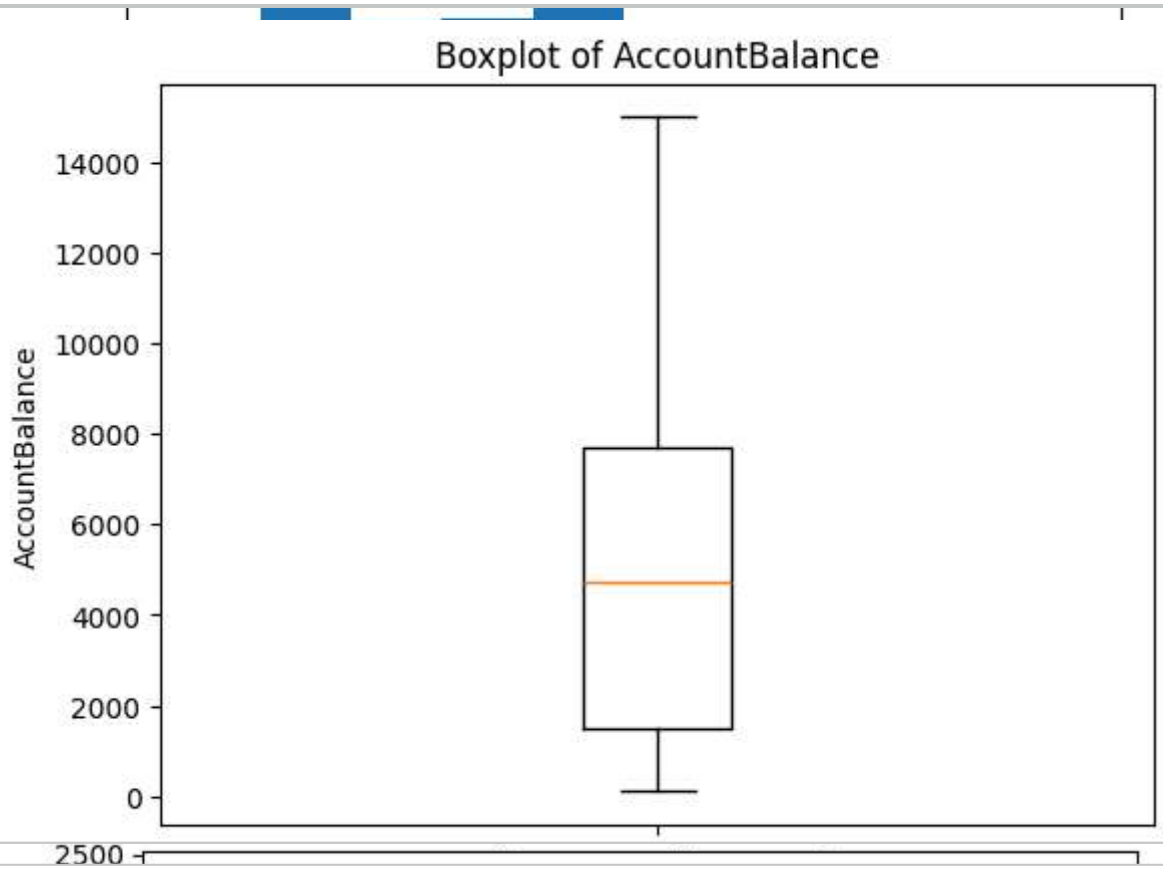
```
# 4. Plot distributions (numeric columns only)
numeric_cols = df.select_dtypes(include=np.number).columns

for col in numeric_cols:
    plt.figure()
    plt.hist(df[col].dropna())
    plt.title(f"Histogram of {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()
```





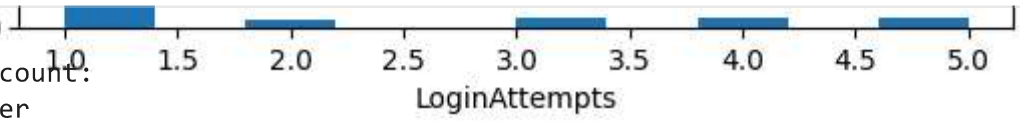

```
plt.figure()
plt.boxplot(df[col].dropna(), vert=True)
plt.title(f"Boxplot of {col}")
plt.ylabel(col)
plt.show()
```



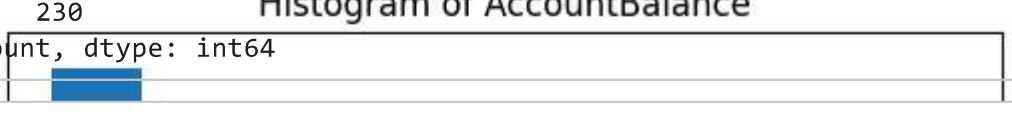



```
# 5. Detect outliers using IQR
outlier_flags = pd.DataFrame(index=df.index)

for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outlier_flags[col + "_outlier"] = ((df[col] < lower) | (df[col] > upper)
```

```
# 6. Create overall outlier flag column
df["is_outlier"] = outlier_flags.any(axis=1)

print("\nOutlier count:")
print(df["is_outlier"].value_counts())
```




```
Outlier count:
is_outlier
False    2282
True      230
Name: count, dtype: int64
```




```
# 7. Handle outliers: capping (winsorization) to preserve data size
df_clean = df.copy()
```

```
for col in numeric_cols:
    Q1 = df_clean[col].quantile(0.25)
    Q3 = df_clean[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    df_clean[col] = np.where(df_clean[col] < lower, lower, df_clean[col])
    df_clean[col] = np.where(df_clean[col] > upper, upper, df_clean[col])
```

```
# 8. Correlation matrix
corr_matrix = df_clean[numeric_cols].corr()
print("\nCorrelation Matrix:")
print(corr_matrix)
```

```
Correlation Matrix:
                    TransactionAmount  CustomerAge  TransactionDuration  \
TransactionAmount            1.000000    -0.021949             0.002129
CustomerAge                 -0.021949     1.000000            -0.017936
TransactionDuration          0.002129    -0.017936             1.000000
LoginAttempts                     NaN          NaN                  NaN
AccountBalance              -0.021299     0.319942             0.005577

                    LoginAttempts  AccountBalance
TransactionAmount             NaN       -0.021299
CustomerAge                   NaN        0.319942
TransactionDuration           NaN        0.005577
LoginAttempts                 NaN             NaN
AccountBalance                NaN        1.000000
```

```
# Top correlations (absolute value)
corr_pairs = (
    corr_matrix.abs()
    .unstack()
    .sort_values(ascending=False)
)

top_corr = corr_pairs[corr_pairs < 1].head(5)
print("\nTop Correlations:")
print(top_corr)
```

```
Top Correlations:
CustomerAge        AccountBalance       0.319942
AccountBalance     CustomerAge          0.319942
CustomerAge        TransactionAmount    0.021949
TransactionAmount  CustomerAge          0.021949
                   AccountBalance       0.021299
dtype: float64
```