Step 1: Import libraries

```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Display plots inline
%matplotlib inline
```

Step 2: Download the dataset

Please note that the dataset that will be downloaded below is a dummy dataset that has been designed for the tutorial. You need to use the actualy dataset provided to you for the analysis.

```
!gdown 1f9ewHqDTGo45XIVH16-benkMWsAHBAZr
```

```
Downloading...
From: https://drive.google.com/uc?id=1f9ewHqDTGo45XIVH16-benkMWsAHBAZr
To: /content/Copy of compiled_risk_data.xlsx
100% 310k/310k [00:00<00:00, 66.9MB/s]
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
print("Setup complete. Imported pandas, seaborn, and matplotlib. Downloaded Compiled risk dataset.")
```

```
Setup complete. Imported pandas, seaborn, and matplotlib. Downloaded Compiled risk dataset.
```

Step 3: Load the Data Section

Now even though we have downloaded the dataset, we still need to load it into our Python environment. For this we will utilize the Pandas library.

```
# Loading the dataset

df = pd.read_excel('Copy of compiled_risk_data.xlsx')

# Display the first five rows of the dataframe
df.head()
```

	project_name	Smart contract address	Blog post link	
0	Data Analytics	384571416209d08623c6ace9422613fc8970475d	https://chainsecurity.com/security-audit/circl...	http
1	Data Analytics	0xAb5801a7D398351b8bE11C439e05C5B3259ae9B	https://stackoverflow.com/questions/75030483/w...	https://studygroup.morali
2	Data Analytics	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db	https://stackoverflow.com/questions/71115106/s...	https://ethereum.stackexchan
3	Data Analytics	0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB	https://stackoverflow.com/questions/75030483/w...	https://studygroup.morali
4	Data Analytics	0x617F2E2fD72FD9D5503197092aC168c91465E7f2	https://stackoverflow.com/questions/69466137/h...	https://ethereum.stackexchan

5 rows × 38 columns

```
# We can then visualize other aspects of the data.
# For example, check for data types and null values

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1093 entries, 0 to 1092
Data columns (total 38 columns):
 #   Column                                   Non-Null Count  Dtype 
---  ------                                   --------------  ----- 
 0   project_name                             1093 non-null   object
 1   Smart contract address                   1093 non-null   object
 2   Blog post link                           1093 non-null   object
 3   Audit website                            1093 non-null   object
 4   Chain                                    1093 non-null   object
 5   Is_closed_source                         1093 non-null   bool  
 6   hidden_owner                             1093 non-null   bool  
 7   anti_whale_modifiable                    1093 non-null   bool  
 8   Is_anti_whale                            1093 non-null   bool  
 9   Is_honeypot                              1093 non-null   bool  
 10  buy_tax                                  1093 non-null   bool  
 11  sell_tax                                 1093 non-null   bool  
 12  slippage_modifiable                      1093 non-null   bool  
 13  Is_blacklisted                           1093 non-null   bool  
 14  can_take_back_ownership                  1093 non-null   bool  
 15  owner_change_balance                     1093 non-null   bool  
 16  is_airdrop_scam                          1093 non-null   bool  
 17  selfdestruct                             1093 non-null   bool  
 18  trust_list                               1093 non-null   bool  
 19  is_whitelisted                           1093 non-null   bool  
 20  is_fake_token                            1093 non-null   bool  
 21  illegal_unicode                          1093 non-null   bool  
 22  exploitation                             1093 non-null   bool  
 23  bad_contract                             1093 non-null   bool  
 24  reusing_state_variable                   1093 non-null   bool  
 25  encode_packed_collision                  1093 non-null   bool  
 26  encode_packed_parameters                 1093 non-null   bool  
 27  centralized_risk_medium                  1093 non-null   bool  
 28  centralized_risk_high                    1093 non-null   bool  
 29  centralized_risk_low                     1093 non-null   bool  
 30  event_setter                             1093 non-null   bool  
 31  external_dependencies                    1093 non-null   bool  
 32  immutable_states                         1093 non-null   bool  
 33  reentrancy_without_eth_transfer          1093 non-null   bool  
 34  incorrect_inheritance_order              1093 non-null   bool  
 35  shadowing_local                          1093 non-null   bool  
 36  events_maths                             1093 non-null   bool  
 37  Summary/rationale of risk tags marked true  701 non-null    object
dtypes: bool(32), object(6)
memory usage: 85.5+ KB
```

Frequency Analysis

```
# Let's now look at the value counts of an individual risk tag: is_airdrop_scam

df['is_airdrop_scam'].value_counts()
```

```
is_airdrop_scam
False    1024
True       69
Name: count, dtype: int64
```

Okay so we see that over 50% of the dataset has True for the column `is_airdrop_scam`. Note that this is a dummy dataset and in real world you won't have that many scams, atleast we can hope that we don't that many scams.

Now, let's define all the risk columns in our dataset so that we can then run the analysis on the same.

```
risk_columns = ['Is_closed_source', 'hidden_owner', 'anti_whale_modifiable',
       'Is_anti_whale', 'Is_honeypot', 'buy_tax', 'sell_tax',
       'slippage_modifiable', 'Is_blacklisted', 'can_take_back_ownership',
       'owner_change_balance', 'is_airdrop_scam', 'selfdestruct', 'trust_list',
       'is_whitelisted', 'is_fake_token', 'illegal_unicode', 'exploitation',
       'bad_contract', 'reusing_state_variable', 'encode_packed_collision',
       'encode_packed_parameters', 'centralized_risk_medium',
       'centralized_risk_high', 'centralized_risk_low', 'event_setter',
       'external_dependencies', 'immutable_states',
       'reentrancy_without_eth_transfer', 'incorrect_inheritance_order',
       'shadowing_local', 'events_maths']
```

Now that we know all the risk columns let's do a full frequency analysis on these columns.

```
# Calculating the frequency of 'True' in each risk tag column
frequencies = df[risk_columns].apply(lambda x: x.value_counts()).loc[True]
frequencies = frequencies.fillna(0)  # Replace NaN with 0 for any column that may not have True values
frequencies
```

```
Is_closed_source                  146
hidden_owner                      164
anti_whale_modifiable             122
Is_anti_whale                     155
Is_honeypot                        94
buy_tax                           128
sell_tax                          126
slippage_modifiable               149
Is_blacklisted                     81
can_take_back_ownership           194
owner_change_balance              222
is_airdrop_scam                    69
selfdestruct                      116
trust_list                        149
is_whitelisted                    109
is_fake_token                      90
illegal_unicode                    62
exploitation                      468
bad_contract                      373
reusing_state_variable            124
encode_packed_collision            81
encode_packed_parameters           87
centralized_risk_medium           283
centralized_risk_high             205
centralized_risk_low              190
event_setter                      149
external_dependencies             316
immutable_states                  154
reentrancy_without_eth_transfer   199
incorrect_inheritance_order       100
shadowing_local                    88
events_maths                      149
Name: True, dtype: int64
```

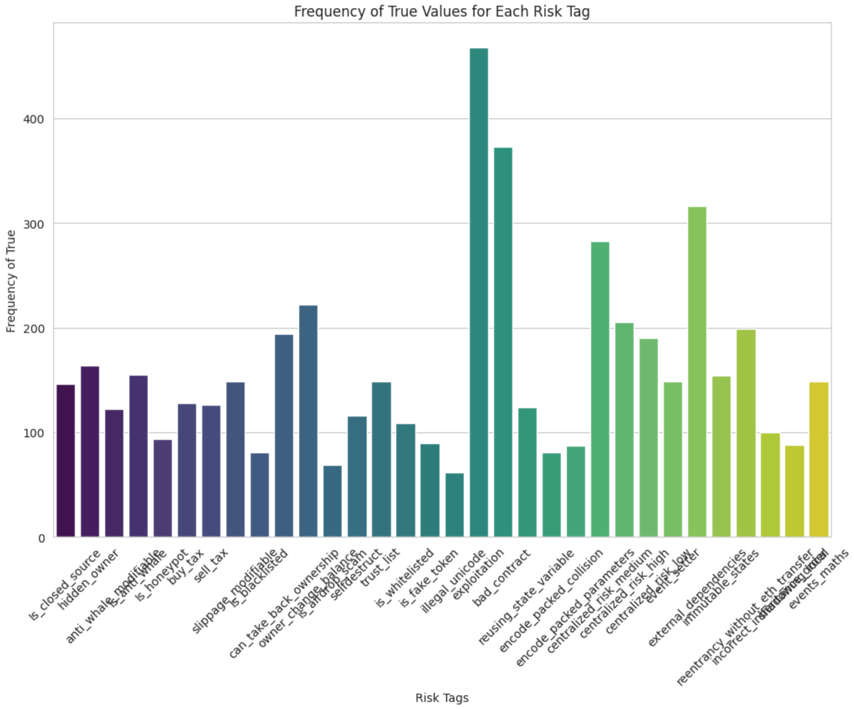
Now that we have the frequencies, we can also visualize these using a barchart

```
# Visualizing the frequencies using a bar chart
sns.set_style("whitegrid")
plt.figure(figsize=(12, 8))
sns.barplot(x=frequencies.index, y=frequencies.values, palette='viridis')
plt.title('Frequency of True Values for Each Risk Tag')
plt.xlabel('Risk Tags')
plt.ylabel('Frequency of True')
plt.xticks(rotation=45)
plt.show()
```

```
<ipython-input-17-4db283aabe4f>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1

  sns.barplot(x=frequencies.index, y=frequencies.values, palette='viridis')
```


Frequency of True Values for Each Risk Tag
Frequency of True
is_closed_source
hidden_owner
anti_whale_modifiable
is_anti_whale
is_honeypot
buy_tax
sell_tax
slippage_modifiable
is_blacklisted
can_take_back_ownership
owner_change_balance
is_airdrop_scam
selfdestruct
trust_list
is_whitelisted
is_fake_token
illegal_unicode
exploitation
bad_contract
reusing_state_variable
encode_packed_collision
encode_packed_parameters
centralized_risk_medium
centralized_risk_high
centralized_risk_low
event_setter
external_dependencies
immutable_states
reentrancy_without_eth_transfer
incorrect_inheritance_order
shadowing_local
events_maths
Risk Tags