```
#!/usr/bin/env python
# coding: utf-8

# In[200]:


import matplotlib.pyplot as plt




# In[282]:


import ssl
import socket
import OpenSSL.crypto
import OpenSSL
import ssl
from datetime import datetime
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from collections import Counter


# In[283]:


# df_https_small.apply(get_cert_data, axis = 1)


# In[284]:


# df_https_small['site'].apply(lambda x: get_cert_data(x))


# In[285]:


df = pd.read_csv('./top_https.csv')
df_https = df[df.https_status == 1]


# In[302]:


df2 = pd.read_csv('./random_https.csv')
df_https_ran = df2[df2.https_status == 1]


# In[303]:


df_https_ran
```

```
# In[289]:


crypto_mapping = {OpenSSL.crypto.TYPE_RSA: 'RSA',\
                  OpenSSL.crypto.TYPE_EC : 'EC', \
                  OpenSSL.crypto.TYPE_DSA: 'DSA', \
                  OpenSSL.crypto.TYPE_DH: 'DH'}


# In[290]:


def get_cert_data(args):
    hostname = args[2]
    print(f"Getting certificate data for {hostname}")

    try:
        context = ssl.create_default_context()
        conn    = context.wrap_socket(socket.socket(), server_hostname=hostname)
        conn.connect((hostname, 443))
        cert     = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_ASN1, \
                                                   conn.getpeercert(True))

        ## Certificate Information
        components = cert.get_issuer().get_components()

        # Q2 Organization Name
        org_name = components[1][1]

        # Q3 Validity range ~ using datetime object
        dt1 = datetime.strptime(cert.get_notBefore().decode(), "%Y%m%d%H%M%SZ")
        dt2 = datetime.strptime(cert.get_notAfter().decode(), "%Y%m%d%H%M%SZ")
        validity = dt2-dt1

        # Q4 Country
        country = components[0][1]

        # Q5 Cryptographic algorithm (convert to rsa etc )
        crypt_type = crypto_mapping[cert.get_pubkey().type()]

        # Q6 Public Key length
        key_len = cert.get_pubkey().bits() # or .key_size

        # Q7 Exponent (incorrect, check how it changes for diff websites)
        try:
            exponent = cert.get_pubkey().to_cryptography_key().public_numbers().e
        except Exception as e:
            exponent = np.nan

        # Q8 Signature algorithm used
        sign_algo = cert.get_signature_algorithm()

        return pd.Series({'org_name': org_name, 'validity': validity, 'country':
country,\
                          'crypt_type': crypt_type, 'key_len': key_len, 'exponent':
exponent,\
                          'sign_algo': sign_algo})

    except Exception as e:
        print(f"FAILED with error {e}")
        dic = {}
        for para in parameter_lst:
            dic[para] = np.nan
        return pd.Series(dic)

#     print(org_name)
#     print(validity)
#     print(country)
#     print(key_len)
#     print(exponent)
#     print(sign_algo)



# In[292]:


df_https_small = df_https[:4]
parameter_lst = ['org_name', 'validity', 'country', 'crypt_type', 'key_len',
'exponent', 'sign_algo']
df_https_small[parameter_lst] = df_https_small.apply(get_cert_data, axis=1)


# In[354]:


np.unique(df_https.sign_algo)


# In[294]:


df_https[parameter_lst] = df_https.apply(get_cert_data, axis=1)


# In[304]:


df_https_ran[parameter_lst] = df_https_ran.apply(get_cert_data, axis=1)


###################################
###################################
###################################
###################################
# NOTEBOOK STUFF ~ Plotting etc.
###################################
###################################
###################################
###################################

# In[189]:


df_https.org_name
```

```
[x.decode() for (x,y) in Counter(df_https.org_name).most_common() if type(x)!=
float]


# In[311]:


[x.decode() for (x,y) in Counter(df_https_ran.org_name).most_common() if type(x)!=
float]


# In[ ]:


df_https.crypt_type


# In[344]:


len(df_https.crypt_type)


# In[362]:


lst = [(x,y) for (x,y) in Counter(df_https.sign_algo).most_common() if type(x)!=
float]
x = [k[0] for k in lst]
y = [k[1] for k in lst]

plt.plot(x,y)
plt.xlabel('Signature Algorithm')

# plt.ylabel('Number of websites')
plt.ylabel('Number of websites')
plt.title('Top Sites')
# plt.title('Top Sites')

plt.xticks(rotation=90);
plt.grid()


# In[356]:


Counter(df_https.sign_algo).most_common()


# In[301]:


sorted([str(name) for (name, count) in Counter(df_https.org_name).most_common()])


# In[ ]:
```

```
# In[312]:


# Q2 Who are the most common CAs, and does the usage of particular CAs
#     seem to vary with the popularity of the site?
c2 = Counter(df_https.org_name).most_common()

# Q3
c3 = Counter(df_https_ran.validity).most_common()


# In[313]:


c3[:3]


# In[314]:


times = [x[0] for x in c3]


# In[315]:


times[0].days


# In[317]:


min(times)


# In[345]:


df_https.crypt_type


# In[350]:


np.unique(df_https_ran[df_https_ran.crypt_type == 'EC']['key_len'], return_counts=
True)


# In[348]:


np.unique(df_https[df_https.crypt_type == 'EC']['key_len'], return_counts= True)


# In[328]:
```

```
df_https_no_na.groupby(df_https.crypt_type).agg({'validity': ['mean', 'count']})


# In[319]:


df_https_ran_no_na = df_https_ran.dropna()


# In[326]:


df_https_ran_no_na.groupby(df_https_ran_no_na.org_name).agg({'validity': ['mean',
'count']})


# In[239]:


# take average validity for each org_name (remove nans before that) to find a
pattern
df_https_no_na = df_https.dropna()
# df_https_no_na.groupby(df_https.org_name).agg({'validity': 'mean'})


# In[333]:


lst = [(x,y) for (x,y) in Counter(df_https.crypt_type).most_common() if
type(x)==bytes and not str(x).startswith('Apple')]
# lst.pop(5)
# lst.pop(6)
x = [k[0] for k in lst]
y = [k[1] for k in lst]

plt.plot(x,y)
plt.xlabel('Country')
plt.ylabel('Number of websites')
plt.title('Top sites ~ Average number of certificates issued')
plt.xticks(rotation=90);
plt.grid()


# In[255]:


Counter(df_https.key_len).most_common()


# In[258]:


# lst = [(x,y) for (x,y) in Counter(df_https.key_len).most_common() if x!
=float(np.nan)]


# x = [k[0] for k in lst]
# y = [k[1] for k in lst]
# plt.plot(x,y)
# plt.xlabel('Country')
# plt.ylabel('Number of websites')
# plt.title('Top Sites ~ Average length of certificate issued')
# plt.xticks(rotation=90);
# plt.grid()


# In[253]:


lst


# In[155]:


type(float(np.nan))


# In[351]:


np.unique(df_https.exponent, return_counts=True)


# In[352]:


np.unique(df_https_ran.exponent, return_counts=True)
```