HTML Extraction Code:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

from bs4 import BeautifulSoup as soup
from urllib.request import urlopen as uReq

my_url = 'https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks/pageLandslideIncidents?_adf.ctrl-
state=ogl93k08h_1&_afrLoop=22162799097188376#!%40%40%3F_afrLoop%3D22162799097188376%26_adf.ctrl-
state%3Dogl93k08h_5'

print('HTTP GET: %s', my_url)
response = requests.get(my_url)

content = soup(response.text, 'lxml')

all_urls = content.find_all('a')

list_url = []
for my_url in all_urls:
  # try URLs containing 'href' attribute
  try:
    # pick up only those URLs containing 'landslide incidence <year>'
    # within 'href' attribute
    if 'https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks' in my_url['href']:
        list_url.append(my_url['href'])

  except:
    pass

list_url

list_ = []
for url in list_url:
  uClient = uReq(url)
  page_html = uClient.read()
  uClient.close()
  page_soup = soup(page_html,"html.parser")
  start = page_soup.findAll('li')
  facts = start[272:500]
  for tag in facts:
    fact = tag.text
    list_.append(fact)
    continue

list_

type(list_)

import pandas as pd
df = pd.DataFrame(list_)
df

df.columns = ['Description']
df.head(5)

df.to_csv('2016-2020 scrapped data.csv')
```

PDF or Human Annotated Extraction Code:

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

from bs4 import BeautifulSoup as soup
from urllib.request import urlopen as uReq

my_url = 'https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks/pageLandslideIncidents?_adf.ctrl-
state=ogl93k08h_1&_afrLoop=22162799097188376#!%40%40%3F_afrLoop%3D22162799097188376%26_adf.ctrl-
state%3Dogl93k08h_5'

print('HTTP GET: %s', my_url)
response = requests.get(my_url)

content = soup(response.text, 'lxml')

all_urls = content.find_all('a')

list_url = []
for my_url in all_urls:
    # try URLs containing 'href' attribute
    try:
        # pick up only those URLs containing 'landslide incidence <year>'
        # within 'href' attribute
        if 'https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks' in my_url['href']:
                list_url.append(my_url['href'])

    except:
        pass

list_url

pdf_url = []
for my_url in all_urls:
    try:
        if 'https://employee.gsi.gov.in/cs/groups/public/' in my_url['href']:
                pdf_url.append(my_url['href'])

    except:
        pass

pdf_url

pip install tabula-py

pip install PyPDF2

from PyPDF2 import PdfFileReader,PdfFileWriter
import requests
import io
from bs4 import BeautifulSoup

desc = []

url=requests.get('https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks/pageLandslideIncidents?_adf.ctrl-
state=ogl93k08h_1&_afrLoop=22162799097188376#!%40%40%3F_afrLoop%3D22162799097188376%26_adf.ctrl-
state%3Dogl93k08h_5')
soup = BeautifulSoup(url.content,"lxml")

for a in soup.find_all('a', href=True):
    if 'https://employee.gsi.gov.in/cs/groups/public/' in a['href']:
        print ("url with pdf final:", a['href'])
        urlpdf = a['href']
        response = requests.get(urlpdf)
        with io.BytesIO(response.content) as f:
            pdf = PdfFileReader(f)
            information = pdf.getDocumentInfo()
            number_of_pages = pdf.getNumPages()
            txt = f"""
            Author: {information.author}
            Creator: {information.creator}
            Producer: {information.producer}
            Subject: {information.subject}
            Title: {information.title}
            Number of pages: {number_of_pages}
            """

            print(txt)
            for i in range(number_of_pages):
              page = pdf.getPage(i)
              output = page.extractText()
              desc.append(output)



desc

len(desc)

pdf_desc = []
for x in range (len(desc)):
  l = desc[x].split('\n \n\n  \n')
  pdf_desc.append(l)


pdf_desc

sdf = pd.DataFrame()
temp =[]
for myList in pdf_desc:
  for i in range(len(myList)):
    temp.append(myList[i])

df["Description"]=temp

df
```

```
df.to_csv('2009-2015.csv')

# loop over all URLs
pdf_desc = []
for my_desc in all_urls:
    # try URLs containing 'href' attribute
    try:
        # pick up only those URLs containing 'pdf'
        # within 'href' attribute
        if 'https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks' in my_desc['href']:
            pdf_desc.append((my_desc['href']))

    except:
        pass

pdf_desc

my_desc = "https://www.gsi.gov.in/webcenter/portal/OCBIS/pageQuickLinks/pageLandslideIncidents2020"
page=requests.get(my_desc)
soup=BeautifulSoup(page.text,'html.parser')
#print(soup.prettify())

# loop over all URLs
pdf_link = []
url = soup.find_all('a')
print(url)
for pdf_url in url:
    # try URLs containing 'href' attribute
    try:
        # pick up only those URLs containing 'pdf'
        # within 'href' attribute
        if 'http://employee.gsi.gov.in/cs/groups/public/documents/document/b3zp/odi0/~edisp/dcport1gsigovi8' in pdf_url['href']:
                pdf_link.append(pdf_url['href'])
        scrapping
    except:
        pass

pdf_link

import tabula

pip install tabula-py

from tabula.io import read_pdf

pdf = "http://employee.gsi.gov.in/cs/groups/public/documents/document/b3zp/odi0/~edisp/dcport1gsigovi824861.pdf"
table_1 = tabula.read_pdf(pdf,pages="all")
table_1
```

Final Data/ Fusing:

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib as plt

# Landslide Description dataset from the year 2009-2015 (which was in PDF format)
```

```
df1 = pd.read_csv('2009-2015.csv')
df1

#Landslide Description dataset from the year 2009-2015 (which was in PDF format)
df2 = pd.read_csv('2016-2020 scrapped data.csv')
df2

#Merging both the dataset
final_data = pd.concat([df1,df2],ignore_index=True)
final_data

final_data.to_csv('Merged_Description.csv')
final_data

del final_data['Unnamed: 0']

final_data

final_data.to_csv('Description:2009-2020')


# for manipulating dataframes
import pandas as pd
# for natural language processing: named entity recognition
import spacy
!python -m spacy download en
from collections import Counter
import en_core_web_sm
nlp = en_core_web_sm.load()
# for visualizations
%matplotlib inline

tokens = nlp(''.join(str(final_data.Description.tolist())))

items = [x.text for x in tokens.ents]
Counter(items)

location_list = []
for ent in tokens.ents:
    if ent.label_ == 'GPE':
        location_list.append(ent.text)

location_counts = Counter(location_list).most_common(20)
df_location = pd.DataFrame(location_counts, columns =['Location', 'count'])

df_location.plot.barh(x='Location', y='count', title="Landslide Prone Areas", figsize=(10,8)).invert_yaxis()

pip install geopandas

pip install geotext

!pip install geopy

!pip install Descartes

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import pandas as pd
import geopandas as gpd
from urllib import request
from geotext import GeoText
from geopy.geocoders import Nominatim
from geopy.exc import GeocoderTimedOut
from shapely.geometry import Point, Polygon
import Descartes

list_ = final_data.values.tolist()
list_

cities = []
for x in list_:
  places = GeoText(str(x))
  city = list(places.cities)
  if city == []:
    cities.append('.')
  else:
    cities.append(city)

cities

df_city = pd.DataFrame(cities)
df_city

geolocator = Nominatim(user_agent='GSI')
lat_lon = []
for city in cities:
        location = geolocator.geocode(city)
        if location:
            print(location.latitude, location.longitude)
            lat_lon.append(location)
        else:
          lat_lon.append(['.','.'])
lat_lon


geolocator = Nominatim(user_agent='GSI')
lat = []
lon = []
for city in cities:
        location = geolocator.geocode(city)
        if location:
            print(location.latitude, location.longitude)
            lat.append(location.latitude)
            lon.append(location.longitude)
        else:
          lat.append(0)
          lon.append(0)

df_lat = pd.DataFrame(lat)
df_lat.columns = ['Latitude']
df_lat

df_lon = pd.DataFrame(lon)
df_lon.columns = ['Longitude']
df_lon
```

```
df_latlong = pd.DataFrame(lat_lon)
df_latlong

df_latlong.columns = ['Address', 'Latitude,Longitude']

df_latlong

df_latlong['Pin code'] = df_latlong['Address'].str.extract(r"\b(\d{6})\b")
df_latlong

type(list_)

!pip install lexnlp
import lexnlp.extract.en.dates
date_list = []
for x in list_:
  date = lexnlp.extract.en.dates.get_dates(str(x))
  dt = list(date)
  date_list.append(dt)

date_list

df_date = pd.DataFrame(date_list)
df_date

for x in range (1,6):
  del df_date[x]

df_date

df_date.columns = ['Date']
df_date

result = pd.concat([final_data,df_date,df_latlong, df_lat, df_lon], axis=1)

result

result.to_csv('Landslideincidence_2009-2020.csv')
```

Visualization:

```
# for manipulating dataframes
import pandas as pd
import matplotlib.pyplot as plt
import spacy
!python -m spacy download en
from collections import Counter
```

```
import en_core_web_sm
nlp = en_core_web_sm.load()
%matplotlib inline
import seaborn as sns

df = pd.read_csv('Merged_Description.csv')
df

tokens = nlp(''.join(str(df.Description.tolist())))

items = [x.text for x in tokens.ents]
Counter(items)

location_list = []
for ent in tokens.ents:
    if ent.label_ == 'GPE':
        location_list.append(ent.text)

location_counts = Counter(location_list).most_common(20)
df_location = pd.DataFrame(location_counts, columns =['Location', 'count'])

df_location.plot.barh(x='Location', y='count', title="Landslide Prone Areas", figsize=(25,17)).invert_yaxis()
# Visualizing at district/village levels.

locations = df[['Latitude', 'Longitude']]
locationlist = locations.values.tolist()
len(locationlist)

import folium
map = folium.Map(location=[20.5937,78.9629], zoom_start=5)
for point in range(0, len(locationlist)):
    folium.Marker(locationlist[point], popup = df['Address'][point]).add_to(map)
map

import datetime
df['Year'] = pd.DatetimeIndex(df['Date']).year
df

count_year = df['Year'].value_counts(dropna=False)

data= df['Year']
data

data = df.groupby(["Year"])["Year"].count().reset_index(name="count")
data

df['Year'].value_counts().plot(kind='pie', figsize=(20,10))

plt.figure(figsize=(15,10))
sns.barplot(x=data['Year'], y=data['count'])
plt.xlabel('Year')
plt.ylabel('Number of Landslides')
plt.xticks(rotation=45)
plt.title('No. of landslides per year')
plt.show()

df['Month'] = pd.DatetimeIndex(df['Date']).month
df
```

```
data = df.groupby(["Month"])["Month"].count().reset_index(name="count")
data

import calendar
data['Month_Name']=[calendar.month_abbr[int(i)] if pd.notna(i) else i for i in data['Month']]
data

plt.figure(figsize=(15,10))
sns.barplot(x=data['Month_Name'], y=data['count'])
plt.xlabel('Month')
plt.ylabel('Number of Landslides')
plt.xticks(rotation=45)
plt.title('No. of landslides each Month')
plt.show()

pip install geopandas

import geopandas
gdf = geopandas.GeoDataFrame(df, geometry=geopandas.points_from_xy(df.Latitude, df.Longitude))
gdf

gdf.plot(markersize = 4.5, figsize = (5,5))

type(gdf)ss
```