TENDER PROJECT REPORT

Introduction:

Tendering is the process of making an offer, bid or proposal, or expressing interest in response to an invitation or request for tender. Organization will seek other businesses to respond to a particular need, such as the supply of goods and services, and will select an offer or tender that meets their needs and provides the best value for money.

Tender request documents; also referred to as invitation to tender, Request for Tender (RTF), Request for Proposal (RFP) etc. Outline what is required, that is what the requesting organization's needs are.

The tendering process is generally utilised for e-procurements or contracts involving substantial amounts of money. Tendering is utilised by:

- Government departments, offices and agencies
- Private sector companies and businesses
- Non-Government Organizations
- Overseas markets and businesses

Scraping the tender data from e-procurement government site using python and scraped data is inserted into oracle database table.

Working Steps:

STEP 1:

Open jupyter Notebook and connect the database with python

Code:

```
#this library is for python connection with oracle database using cx_oracle

import db_config

con = cx_Oracle.connect(db_config.user, db_config.pw, db_config.dsn)
```

STEP 2:

Import the libraries and load the tender link data with State wise

Code:

```
# Popular Python-based data analysis toolkit (pandas)

Import pandas as pd
```

```
# Beautiful Soup is a library that makes it easy to scrape
information from web pages
From bs4 import BeautifulSoup
#the selenium package is used to automate web browser
interaction from Python.
From selenium import webdriver
From selenium.webdriver.common.keys import Keys
#this can be with the help of the Chrome Options and the Desired capabilities class
From selenium.webdriver.chrome.options import Options
#for requesting the url
Import requests
#load the data of tender links with state wise
link = pd.read_excel (r'C:\Users\vishal.lote\loan_project\Tender_Links.xlsx')
urls = list (link['STATE_URL'].str.split("?", n= 1, expand=False).str[0])
```

STEP 3:

Used the for loop condition on all state urls column and try the all url to scrapped them

Code:

```
#using for loop for scraping all urls data from data frame
For url in urls:
    try:
        …… (Enter here all below code from step 4 to step 7)
    except:
        continue
```

STEP 4:

Request the url using webdriver.Chrome and fetch the all links of tender details from table data

Code:

```
options = Options()

options.headless = False

options.add_argument('--headless')

#load the chromedriver.exe path for requesting the url

driver = webdriver.Chrome
(executable_path=r'C:\Users\vishal.lote\Downloads/chromedriver.exe',options=op
tions)

links = [ ]

driver.get(url + "?page=FrontEndTendersByOrganisation&service=page")

#it is used retrieve the page source of the webpage the user is currently accessing

src = driver.page_source

#it is used for parsed the html

soup = BeautifulSoup(src, 'lxml')

#it is used for find the class of the table

name_div = soup.find('table', {'id':"table",'class': 'list_table'})

gdp_table_data = name_div.findAll("td")

#then all table class ‘td’ data in for loop

for tr in gdp_table_data:

    cols = tr.find_all('a')

    cols = [ele.get('href').strip() for ele in cols]

    links.append([ele for ele in cols if ele])

# All links data to the dataframe

data = pd.DataFrame(links)

#then cleaned the link data

data.dropna(inplace = True)
```

```
data.reset_index(drop = True, inplace = True)
link1 = pd.DataFrame(data[0].str.split("?", n= 1, expand=False).str[1])
```

STEP 5:

Scrapped the one by one URL links of tender from the table data

Code:

```
all_data = []
#used the for loop in link1 data
for x in range(0, len(link1[0].index)):
# Request the url using webdriver
driver.get(url +'?'+ link1[0][x])
src = driver.page_source
soup1 = BeautifulSoup(src, 'lxml')
#then find the class of the table
name_div = soup1.find('table', {'id':"table",'class': 'list_table'})
table = soup1.find("table",{"class":"list_table"})
#it is used for find all table heading with data
columns = [i.get_text(strip=True) for i in table.find_all("th")]
data2 = []
data3 = []
#then we are used for loop of ‘tr’ class and append the data in list
for tr in table.find("tbody").find_all("tr"):
    data2.append([td.get_text(strip=True) for td in tr.find_all("td")])
#then convert the list data into dataframe
data2 = pd.DataFrame(data2)
data2 = data2.iloc[1:].reset_index()
#then we have to find the list_table class for scraping the tender links
```

```
name_div = soup1.find('table', {'id':"table",'class': 'list_table'})
gdp_table_data = name_div.findAll("td")
#we are using for loop for the links of tender data
for tr in gdp_table_data:
    cols = tr.find_all('a')
    cols = [ele.get('href').strip() for ele in cols]
    data3.append([ele for ele in cols if ele])
#then convert the list data into dataframe
data3 = pd.DataFrame(data3)
#then cleaning the data
data3.dropna(inplace = True)
data3.reset_index(drop = True, inplace = True)
link3 = pd.DataFrame(data3[0].str.split("?", n= 1, expand=False).str[1])
link3.columns = ['Link']
link3['Link'] = url #+ '?'+ link3['Link'].astype(str)
#then concat the tender data and link data
data = pd.concat([data2, link3], axis = 1)
# append the data to all_data
all_data.append(data)
```

STEP 6:

Data cleaning of the all scrapped data from tender links.

Code:

```
#it is used for display the 500 row and columns
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 500)
#then convert the list data into dataframe
df_temp3 = list(pd.DataFrame(all_data)[0])
```

#then concat the data with ignore index

```
tender_data = pd.concat(df_temp3, axis=0, ignore_index=True)
```

#then clean the data frame data

```
tender_data[['A', 'B','C']] = tender_data[4].str.split(']', 2, expand=True)

tender_data.dropna(inplace = True)

tender_data = tender_data.drop(['index'],axis=1)
```

#rename the columns according to columns name

```
tender_data = tender_data.rename({0: 'S.No', 1: 'e-Published Date', 2: 'Closing Date', 3: 'Opening Date', 4: 'Title and Ref.No./Tender ID',5:'Organisation Chain','A': 'Title','B': 'Ref.No.','C': 'Tender ID'}, axis=1)

tender_data = tender_data.rename({'S.No':'IDS','e-Published Date':'e_published_date', 'Closing Date':'closing_date','Opening Date':'opening_date','Organisation Chain':'organisation_chain','Title':'tender_title','Ref.No.':'ref_no','Tender ID':'tender_id','Link':'tender_url'}, axis=1)
```

#sequence the data according tender data

```
tender_data = tender_data[['IDS','e_published_date','closing_date','opening_date','ref_no', 'tender_id', 'organisation_chain', 'tender_url', 'tender_title']]
```

#then clean the dataframe column

```
tender_data["ref_no"] = tender_data["ref_no"].str.replace("[", "", regex=True)

tender_data["tender_id"] = tender_data["tender_id"].str.replace("[", "").str.replace("]", "", regex=True)

tender_data["tender_title"] = tender_data["tender_title"].str.replace("[", "").str.replace("]", "", regex=True).str.upper()
```

#then rename the dataframe column in upper case

```
tender_data = tender_data.rename({'IDS':'IDS','e_published_date':'E_PUBLISHED_DATE', 'closing_date':'CLOSING_DATE','opening_date':'OPENING_DATE','ref_no':'REF_NO','tender_id':'TENDER_ID','organisation_chain':'ORGANISATION_CHAIN','tender_url':'TENDER_URL','tender_title':'TENDER_TITLE'}, axis=1)
```

#then tender data column IDS with indexes and 'E_PUBLISHED_DATE', 'CLOSING_DATE'&'OPENING_DATE' is convert into pd.datetime

```
tender_data['IDS'] = tender_data.index

tender_data['E_PUBLISHED_DATE'] = pd.to_datetime(tender_data['E_PUBLISHED_DATE'])

tender_data['CLOSING_DATE'] = pd.to_datetime(tender_data['CLOSING_DATE'])

tender_data['OPENING_DATE'] = pd.to_datetime(tender_data['OPENING_DATE'])

tender_data = tender_data.fillna(0)
```

STEP 7:

Then connect the database with python and insert the scraping dataframe to the per_day_tender_data table in loop.

Code:

```
#connect the database with python

con = cx_Oracle.connect('py/py@192.168.1.42/orcl')

cursor = con.cursor()

#Then tender data into list

df_list = tender_data.values.tolist()

#Using for loop insert the data per_day_tender_data

for i in range(len(df_list)):

    cursor.prepare('INSERT INTO py.per_day_tender_data VALUES(:1,:2,:3,:4,:5,:6,:7,:8,:9)')

    cursor.executemany(None,([df_list[i]]))

con.commit()

#print url which urls is scrapped

print(url)
```

STEP 8:

Then write a procedure for update, insert and delete the column from database table, such as now we have 3 tables in database per_day_tender_data, test_tender_model and history_tender_data. If we have data in test_tender_model then update the data in update_date columns and if we have not data in test_tender_model then insert the data from per_day_tender_data according to tender_id. Then insert the data into history_tender_data from test_tender_model which closed_date column is less than sysdate and deleted this data from test_tender_model.

Code:

```
#connect the database with python
con = cx_Oracle.connect('py/py@192.168.1.42/orcl')
cursor = con.cursor()
#execute the oracle procedure in python
cursor.execute('Begin pr_insert_tender_data;commit;end;')
#at the end truncate the per_day_table_data
cursor.execute('TRUNCATE TABLE per_day_tender_data')
con.commit()
cursor.close()
con.close()
```