# 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.

## Objective: Extract the Data of Government e-procurement tender using python web scraping.

## SUMMARY:

1. Connect the oracle database with python
2. Import the required library
3. Create the excel of e-procurements government website link for all states
4. Then send the request to one by one urls using beautiful soup library for scrapped the data
5. Fetch the data using class and tag attributes of the html page
6. Then all data is append in empty list and converted into data frame using pandas library
7. And scraped data frame is inserted into oracle database table using python sql query

Algorithm

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=options)
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','t
ender_id':'TENDER_ID','organisation_chain':'ORGANISATION_CHAIN','tender_url':'TE
NDER_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()
```

## API Documentation

### Summary

- First We create the project using django command
- Inside of project we create application using django command
- Then in Project folder setting.py file is use for Database integration and Install our application
- Then We create the table using model.py file and then Migrate using django command
- Then we create the form for frontend in form.py file for user interface inside the application folder
- Then we create the HTML and CSS inside the application Folder in templates folder for Frontend.
- After that we import required files in view.py for  business logic
- Create the route URL in url.py for application in both application folder and project folder
- And at the end we run the command for API start

# Open the terminal for Django forlder creation

# write the command for django application create the folder stracture

- django-admin startproject project_name

# Go inside the project

- cd project_name

#Folder stracture in project directory

- project_name/
- manage.py
- project_name /

- __init__.py
- settings.py
- urls.py
- asgi.py
- wsgi.py

# Create the app

- django manage.py startapp app_name

# Inside the app_name project stractuer

- app_name /
- __init__.py
- admin.py
- apps.py
- migrations/
- __init__.py
- models.py
- tests.py
- views.py
- templates/
- dashboard.html

## File Name: setting.py

# Provide the your system ip for run the application on your ip server

ALLOWED_HOSTS = ['192.168.1.76']

Run the Application write the command on default port ie. 8000

- Python manage.py runserver 192.168.1.76:8000

# Add the app name in the setting.py file In Installation variable

INSTALLED_APPS = [

```
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app_name',
]
   # Add the Database connection in setting.py file
DATABASES = {
    'default': {
    'ENGINE':   'django.db.backends.oracle',
    'NAME':     '192.168.1.42/orcl',
    'USER':           'py',
    'PASSWORD': 'py',
    #'PORT':          '1521'
     }
}
```

## File Name: model.py

```
# Create the model.py file for table creation in django for creating the table in database
# Create your models here.
from django.db import models
from datetime import datetime
```

```
# Create your models here for database table.
# Create the table test_tender_model
class test_tender_model(models.Model):
    IDs = models.AutoField(primary_key=True)
    E_Published_Date = models.DateField()
    Closing_Date = models.DateField()
    Opening_Date= models.DateField()
    Update_Date=models.DateField(default=datetime.today().strftime('%Y-%m-%d'))
    Organisation_Chain= models.CharField(max_length=1000)
    Tender_Title= models.CharField(max_length=1000)
    Ref_NO= models.CharField(max_length=100)
    Tender_ID= models.CharField(max_length=50)
    Tender_URL=models.CharField(max_length = 1000)

    class Meta:
      db_table = "test_tender_model"


class tender_keword_models(models.Model):
    keywords= models.CharField(max_length=50)
    class Meta:
        db_table='test_tender_kewords'


# creating the table in database using following command
```

- Python manage.py makemigrations
- Python manage.py migrate

## File Name: form.py

```
# In form.py file create the front end form

# Get the data from the dastabase tender_keyword_models table for search keyword in database tender table

# Import form from django

from django import forms

# Import model from model.py file for form creation

from app_test_tender.models import tender_keword_models

import pandas as pd

# Create DataFrame using object relation Maping

read_data = pd.DataFrame(list(tender_keword_models.objects.all().values()))

OPTIONS=[ ]

for i in range(len(read_data)):

    a=(tuple((read_data['keywords'][i],read_data['keywords'][i])))

    OPTIONS.append(a)

OPTIONS=tuple(OPTIONS)


class CountryForm(forms.Form):

    # Create the keyword form for selection

KETWORDS = forms.MultipleChoiceField(widget=forms.CheckboxSelectMultiple,required=True,choices=OPTIONS)
```

## File Name: view.py

```
# Creating the view.py file for data processing and performing business operation  logic

# Import all the library and model and form in application

from django.http.response import HttpResponse

from django.shortcuts import render,redirect
```

```
from numpy import frombuffer
from app_test_tender.models import test_tender_model
from app_test_tender.forms import CountryForm
from django.template import RequestContext
import pandas as pd
from django.db.models import Q
from django.db import connection

# Create the function for GET & POST request
def show(request):
    form = CountryForm(request.POST )
# If form received the post request
  if request.method == 'POST':
        if form.is_valid():
            form = CountryForm()


    # The "display_type" key is taking from "displaybox" or "locationbox"
        fruits_ = request.POST.getlist("KETWORDS")

   # Convet it into labmdata function for list creation
        fruits_ = list(map(lambda x:x.upper(), fruits_))

  # join the list using “ | “ function
        pattern = '|'.join(fruits_)
        print(pattern)
```

```
 # Get the data from the query send in Database

        dfs_query="SELECT * FROM TEST_TENDER_MODEL WHERE regexp_like
(Tender_Title, '{}') ".format(pattern)

        tenders_=pd.read_sql_query(dfs_query,connection)

 # If the length of tenders_ data is greater than zero

        if int(len(tenders_))>0:

            tenders_=tenders_.to_dict(orient='records')

            # print(tenders_)

            return
render(request,r'D:\Python_Project\py\Clinet\test_tender\app_test_tender\templates\das
hboard.html', {'student': tenders_,'form': form})

        else:

            return
render(request,r'D:\Python_Project\py\Clinet\test_tender\app_test_tender\templates\das
hboard.html',{'form': form})#, {'student': tenders_}


# render on same page if provide get request


    if request.method == 'GET':


        return
render(request,r'D:\Python_Project\py\Clinet\test_tender\app_test_tender\templates\das
hboard.html',{'form': form})#, {'student': tenders_})




# Create the HTML page inside the template folder

# Create the the HTML inside the Application folder

Path = r“app_test_tender\templates\dashboard.html”
```

#dashboard.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta http-equiv="x-ua-compatible" content="ie=edge">
  <title>Fluent Design from MDB - Shadows</title>
  <!-- Bootstrap core CSS -->

  <style type="text/css">
   * {
          margin:0px;
          padding:0px;
          border:none;
          outline:none;
}
body {
          font-family: 'Montserrat', sans-serif;
          font-size:13px;
          color:#303030;
          line-height:1.7em;
          font-weight:400;
          background:#ffffff;
          -webkit-font-smoothing: antialiased;
          -moz-font-smoothing: antialiased;
}
```

```
.btn-secondary{
    background: #0a5c95;
}
/* nav */
row{
    width: 100%;
}
nav {
        position: relative;
        margin: 50px;
        width: 360px;
}
nav ul {
        list-style: none;
        margin: 0;
        padding: 0;
}
nav ul li {
  /* Sub Menu */
}
nav ul li a {
        display: block;
        background: #ebebeb;
        padding: 10px 15px;
        color: #333;
        text-decoration: none;
        -webkit-transition: 0.2s linear;
        -moz-transition: 0.2s linear;
```

```
        -ms-transition: 0.2s linear;
        -o-transition: 0.2s linear;
        transition: 0.2s linear;
}
nav ul li a:hover {
        background: #f8f8f8;
        color: #515151;
}
nav ul li a .fa {
        width: 16px;
        text-align: center;
        margin-right: 5px;
        float:right;
}
nav ul ul {
        background-color:#ebebeb;
}
nav ul li ul li a {
        background: #f8f8f8;
        border-left: 4px solid transparent;
        padding: 10px 20px;
}
nav ul li ul li a:hover {
        background: #ebebeb;
        border-left: 4px solid #3498db;
}
.keywords_title a{  padding-left: 12%;
    font-size: 18px;
```

```
    color:#0a5c95;}

 .fa-caret-down, .fa-caret-up{
    position: absolute;
    right:79%;
 }

/* nav end */
.top_header{
background: #fff;
box-shadow: 5px 2px 2px #eee;
padding: 10px 0;
}
.logo_left{ margin-left: 35px; }
.logo{
    display: block;
    float: left;
}
.top_right_side{
    display: block;
    float: right;
    right: 0;
    width: 12%;
    position: absolute;
}
.mr_right{
    margin-top: 20px;
    right: 0;
```

```
    position: absolute;
}
.logout_name{
    margin-left:10px;
    font-size: 18px;
    font-weight: 600;
    color:#222;
}
.top_right_side img{
    width: 30px;
}
.logout_name:hover{
    color:#03459e;
}
 a:hover{
    text-decoration: none;
}
.list{
    padding: 0 15px;
}
.keywords_title{
    color: #222;
    padding: 10px 5px;
    width: 100%;
}
.left_side_item{
background: #fff;
box-shadow: 5px 2px 2px #eee;
```

```
height: 100vh;
width:25%;
overflow: scroll;
top: 0px;
padding-top: 10px;
position: fixed;
}
.list_side_item  li{
    list-style: none;
}
.list_side_item label{
    display: flex;
}
.middle_box{
    display: block;
    float: left;
    padding: 23px 10px 0px 90px;
    width: 415px;
}
.list{
    padding-left: 35px;
    padding-top: 15px;
}
.submit_fixed{
    width: 24.6%;
    position: fixed;
    bottom: 0;
    left: 0;
```

```
    background: #dcedfb;

}
.submit_btn , .clear_btn{
    padding: 10px 40px;
    background: #0a5c95;
    border-radius: 5px;
    color:#fff;
    margin: 20px 10px;
    display: block;
    float: left;
    border:none;

}
.group_btn{
   text-align: center;
   display: table;
   margin: auto;
}
input[type=checkbox]
{
    margin-right: 10px;
    margin-top: 8px;
    min-width: 20px;
}
.main_container{
    width: 90%;
    margin: 30px;
```

```
    position: relative;
}
.table_box{
    margin-bottom: 10px;
}
.table .thead-dark th{
    background: #0a5c95;
    color : #fff;
    padding: 4px;
}
.table-responsive{
    display: inline-table;
}
.hidden-bar-wrapper{
overflow-x: hidden;
}
td th{
    padding: 10px;
}
.table-responsive{
width: 100%;
margin-bottom: 20px;
}
.table_bg{
    position: relative;
    padding-bottom: 20px;
    border-bottom:1px solid rgb(170, 170, 170);
}
```

```
@media only screen and (max-width: 767px) {
    .logo img{
        width: 200px;
    }
    .top_header, .left_side_item{
        width: 100%;
    }
    .top_right_side{
        width: 31%;
        margin-top: 25px;
    }
    .top_header{
        box-shadow: none;
    }
    .middle_box{
        padding: 13px 20px 10px 38px;
    }
    .left_side_item{
        height: 450px;
        overflow: scroll;
    }
.table_responsive{
    overflow: scroll;
}
.main_container{
    width: 90%;
    background: #fff;
```

```
    z-index: 1;
    margin: 25px;
}
.submit_fixed{
    width: 100%;
    position: fixed;
    bottom: 60px;
}
.table-responsive{
    display: block;
}
.sh2 {
text-align: left;
}

}
  </style>
</head>
<body class="hidden-bar-wrapper">

  <main>
    <div class="container-fluild">
      <div class="row">
          <div class="left_side_item">
            <ul class="list_side_item">
              <form action="" method="post">
                # From form.py it will create frontend form in html.
                {% csrf_token %}
```

```
            {# This will display the radio button HTML for you #}
            {{ form.as_p }}

           # You'll need a submit button or similar here to actually send the form
            <!-- <input type="submit" value="Submit"> -->
            <div class="submit_fixed">
               <div class="group_btn">
                  <input type="submit"  class="submit_btn" value="Submit">
                </div>
              </div>
           </form>
         </ul>
       </div>
        <div class="main_container">
        <div style = "position:absolute;left:310px">
        <div style="margin-left:5px;display:flex;padding:5px">
        <div>
    {% for stud in student %}
        <table width="100%" border="0" align="center" cellpadding="0" cellspacing="1"
class="table_bg">
           <tbody><tr>
              <td class="td_caption"><b>Organisation Chain</b></td>
              <td class="td_field" colspan="5">{{stud.ORGANISATION_CHAIN}}</td>
           </tr>
             <tr>
              <td class="td_caption"><b>Tender Reference Number</b></td>
              <td colspan="3" class="td_field">{{stud.REF_NO}}</td>
           </tr>
```

```
            <tr>
               <td class="td_caption"><b>Tender Title<b></b></td>
               <td colspan="3" class="td_field">{{stud.TENDER_TITLE}}</td>
            </tr>
            <tr>
               <td class="td_caption" width="25%"><b>Tender Url</b></td>
               <td class="td_field" width="25%"><a href
={{stud.TENDER_URL}}>{{stud.TENDER_URL}}</a></td>
               <td class="td_caption" width="25%"><b>Tender Id</b></td>
               <td class="td_field" width="25%">{{stud.TENDER_ID}}</td>
            </tr>
            <tr>
               <td class="td_caption"><b>E Published Date</b></td>
               <td class="td_field">{{stud.E_PUBLISHED_DATE}}</td>
               <td class="td_caption" width="20%"><b>Opening Date</b></td>
               <td class="td_field" width="20%">{{stud.OPENING_DATE}}</td>
            </tr>
            <tr>
               <td class="td_caption"><b>Closing Date</b></td>
               <td class="td_field">{{stud.CLOSING_DATE}}</td>
               <td class="td_caption"><b>ID</b></td>
               <td class="td_field">{{stud.IDS}}</td>
            </tr>
         </tbody>
         </table>

         <br>
      {% endfor %}
```

```
      </div>
      </div>
    </div>
    </div>
    </main>
  <!-- Main layout -->
</body>
  <!-- Footer -->
  <footer id="page-footer">
</footer>
</body>
</html>
```

## File Name: url.py

```
# find the url.py file in project name directory
# test_tender URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
```

```
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
from django.contrib import admin
from django.urls import path
from app_test_tender import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path('show/',views.show),

]

# Run command
```

- Python manage.py runserver system_ip:default_post