Take Home Assignment - CivicDataLab

1) Data Scraping
2) Data Cleaning and Transformation
3) Data Loading
4) Data checkup
4) Airflow Pipeline Setup

Data Scraping :

To scrap the data from a website we have python inbuilt libraries like BeautifulSoup , Scrapy and few others. For using BeautifulSoup we need additional packages like requests and parsers like html and lxml . we can parse only html or XML pages in it. Also , It takes more time to fetch when the data is so huge . So , In this case I have chosen scrapy to fetch the data as using it can scale even with the high data coming in , also we can use it even with the dynamic web pages.

Scrapy Setup :

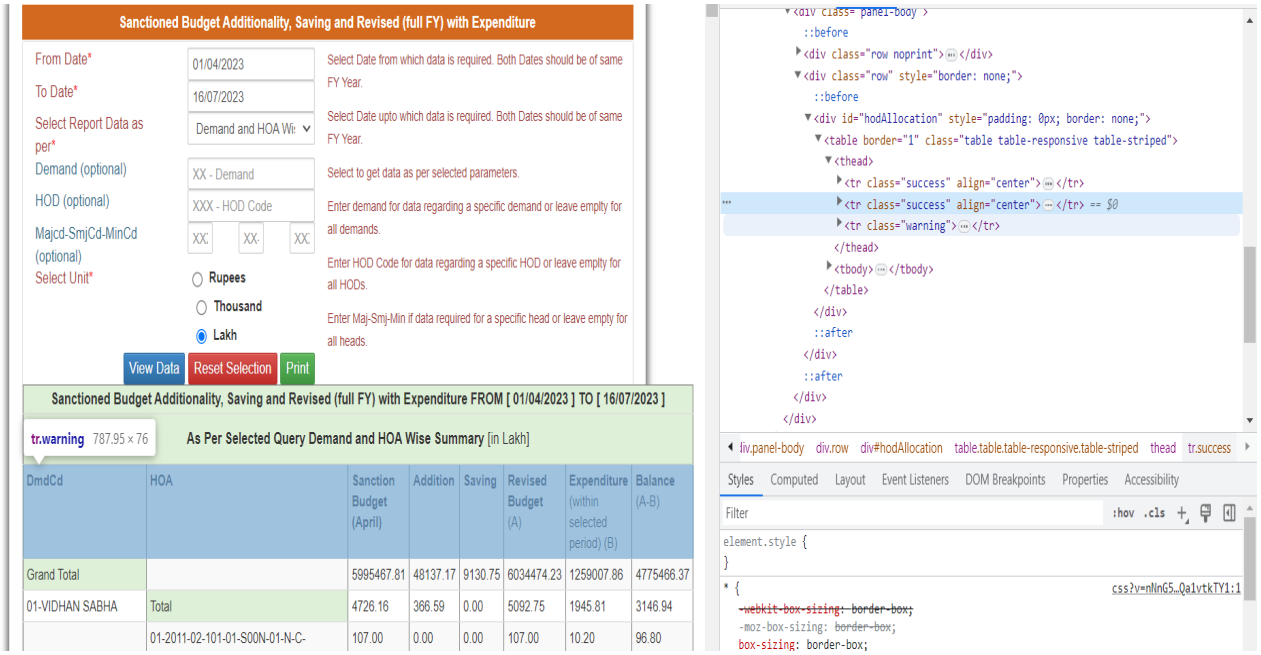
⇒ Created a new project using pycharm as it creates a **virtual environment** on that path
⇒ Then install all the required python libraries that will be required
⇒ Now , we need to start a scrapy project using **scrapy startproject budget_dataset**
⇒ Then we need to move to the budget_dataset path from current path using the terminal **cd budget_dataset**
⇒ Now we need to set up a budget spider using **scrapy genspider budget x**
⇒ **Replace the value of "x"** with the actual website in the start_urls list
⇒ Now we will be having a parse function where it will be called once the response is available for the given website
⇒ In our case we need to fetch the variables like `__VIEWSTATE ,__VIEWSTATEGENERATOR.` In our . But in scrapy we do have a inbuilt function `FormRequest` which fetches all those values .
⇒ Additionally , we need to pass all the other params which we require from the website Like the **start date , end date , unit and type of da**ta we want .

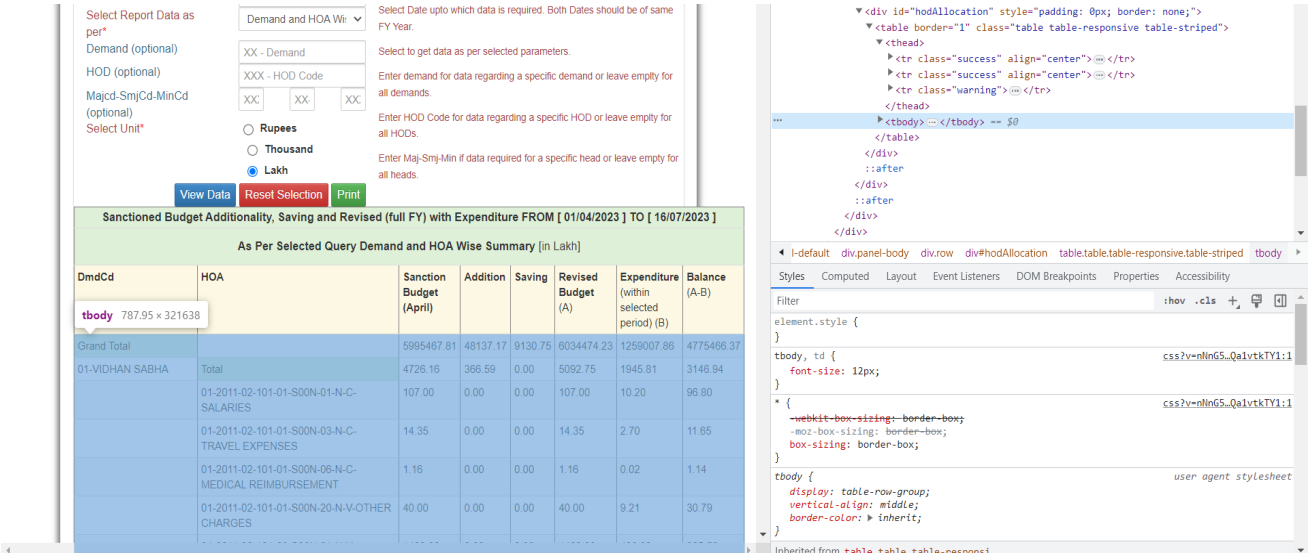
```
data = {
    "ctl00$MainContent$txtFromDate": "01/04/2018",
    "ctl00$MainContent$txtQueryDate": "31/03/2022",
    "ctl00$MainContent$ddlQuery": "DmdCd,HOA",
    "ctl00$MainContent$rbtUnit": "0",
    "ctl00$MainContent$btnGetdata": "View Data"
}

yield FormRequest.from_response(response, formdata=data, callback=self.parse_table)
```

⇒ Now , this will trigger the url with the params we passed and sends the response back to **parse_table** function .

⇒ In this we have loaded the table from the website directly to dataframe using pandas Function **read_html** which fetches table data from the html page .

⇒ If we don't wish to use the inbuilt function we can still scrap the table based on the css selector on that which we had like table "thead -> tr.warning for columns in table class and tbody for rows "





⇒ Then we pass the dataframe to `clean_dataframe` dataframe where we do all the cleaning and transformation of data

Data Cleaning and Transformation :

⇒ First we clean all the column names by **removing all the characters** apart from alphabets in it by using regex sub function which replaces non alpha characters to empty
⇒ Then, we are dropping all the rows where **'DmdCd'** column has `'Grand Total'` by fetching their indexes
⇒ After that , we will all the rows where **'HOA'** columns has **'Total'** value to a new dataframe
⇒ From the new dataframe we take the **'DmdCd' col values** and store it in a list and sorting it as it has a prefix of DemandCode which we use to map the prefix from 'HOA'
⇒ Now , we will drop all the rows where 'HOA' columns has 'Total' value
⇒ As 'DmdCd' column will have null values for the rows we will be mapping every row with the list we created above by matching with 'HOA' prefix
⇒ Now , we initialise all the new column names that we require for **splitting "HOA" and "DmdCd" columns**
⇒ Then we split those columns and assign them to those lists .

Data Loading :

⇒ Once after dataframe is cleaned and transformed we load it into a csv file in *files* path using pandas **to_csv** function
⇒ Then , we run **loading_to_sqllite** python file where it has `df_to_sqllite` function
⇒ Inside this , It will try to connect to database present in database path .If database is not present it will create with the name we have specified.
⇒ Then , it will create `TREASURY_EXPENDITURE` table in the db after checking if its present or not.
⇒ Now , it will read the csv file from *files* path and load it to dataframe using pandas **read_csv.**
⇒ Then we will load the data frame to the `TREASURY_EXPENDITURE` table using pandas **to_sql** function.

Data Checking :

⇒ Once after loading the data to the sqlite table .We need to verify the data is matching with web UI values .
⇒ So , I have checked the group total value and each group value of the ‘Dmdcd’ column.

UI Data :

DmdCd	HOA	Sanction Budget (April)	Addition	Saving	Revised Budget (A)	Expenditure (within selected period) (B)	Balance (A-B)
Grand Total		567779223000.00	60822652816.00	79446508115.00	549155367701.00	354729782693.00	194425585008.00
01-VIDHAN SABHA	Total	459262000.00	80751000.00	4115000.00	535898000.00	494356220.00	41541780.00

DB Data :

```
select sum(SanctionBudgetApril),sum(Addition),sum(Saving),sum(RevisedBudgetA),sum(ExpenditurewithinselectedperiodB),sum(BalanceAB) from TREASURY_EXPENDITURE
```

Grid view | Form view

Total rows loaded: 1

	sum(SanctionBud	sum(Addition)	sum(Saving)	sum(RevisedBudg	sum(Expenditure	sum(BalanceAB)
1	567779223000	60822652816	79446508115	549155367701	354729782693	194425585008

```
select DmdCd , sum(SanctionBudgetApril),sum(Addition),sum(Saving),sum(RevisedBudgetA),sum(ExpenditurewithinselectedperiodB),sum(BalanceAB) from TREASURY_EXPENDITURE group by 1
```

Grid view | Form view

Total rows loaded: 32

	DmdCd	sum(SanctionBud	sum(Addition)	sum(Saving)	sum(RevisedBudg	sum(Expenditure	sum(BalanceAB)
1	01-VIDHAN SABHA	459262000	80751000	4115000	535898000	494356220	41541780
2	02-GOVERNOR AND COUNCIL OF MINISTERS	252364000	20264333	6171000	266457333	233979691	32477642
3	03-ADMINISTRATION OF JUSTICE	2518431000	521381068	334147538	2705664530	2303046142	402618388
4	04-GENERAL ADMINISTRATION	2841977000	408196723	108590043	3141583680	2434424894	707158786
5	05-LAND REVENUE AND DISTRICT ADMINISTRATION	17663847000	5877329583	4956563730	18584612853	11241692618	7342920235
6	06-EXCISE AND TAXATION	1023514000	178254720	15602278	1186166442	1062920454	123245988
7	07-POLICE AND ALLIED ORGANISATIONS	16183672000	730445985	2158038978	14756079007	13865626598	890452409
8	08-EDUCATION	71788332000	2501934021	8185769636	66104496385	62625575556	3478920829
9	09-HEALTH AND FAMILY WELFARE	25085460000	4592829224	1167676077	28510613147	24901613610	3608999537
10	10-PUBLIC WORKS - ROADS, BRIDGES AND BUILDINGS	65305413000	3116077051	11576331323	56845158728	16539665389	40305493339
11	11-AGRICULTURE	5152531000	316180000	690498175	4778212825	4253786128	524426697
12	12-HORTICULTURE	3980724000	1372810391	710007000	4643527391	4545808180	97719211

Airflow Pipeline Setup :

⇒ To setup the dag in airflow , we need to place the “budget_dataset” project in the dags folder
⇒ The airflow dag code for our project is present in airflow_dag directory
⇒ Inside the dag we will be having two tasks one is a bash task where it goes to project path present in the dags folder and then ,it runs scrapy crawl budget command .This will be responsible for loading the data to the csv file.
⇒ Then we have a python operator which loads the data from csv to sqlite db.

Setting Up:

⇒ Add the **budget_dataset** directory in the **airflow dags directory**.
⇒ Changing the **`project_path`** in web_to_sqldb_pipeline.py python file to dags path adding project_name in the path
⇒ Also , we need to provide the same path for **`project_path`** variable in budget.py file in spiders directory