# 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.

  But in scrapy we do have an 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 .

v<alv class= panel-body .</pre>

::before

▶ <div class="row noprint"> ... </div>

▼ <div class="row" style="border: none;">

- ⇒ 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 .
- ⇒ The other way of fetching the data from the table is filtering rows data and columns using the selectors. As shown below for we have tr.warning and tbody for rows.
- ⇒ With change in any tags in future we might again need to change the code.So , As a generalised onel used the **read\_html** function.

Select Date from which data is required. Both Dates should be of same

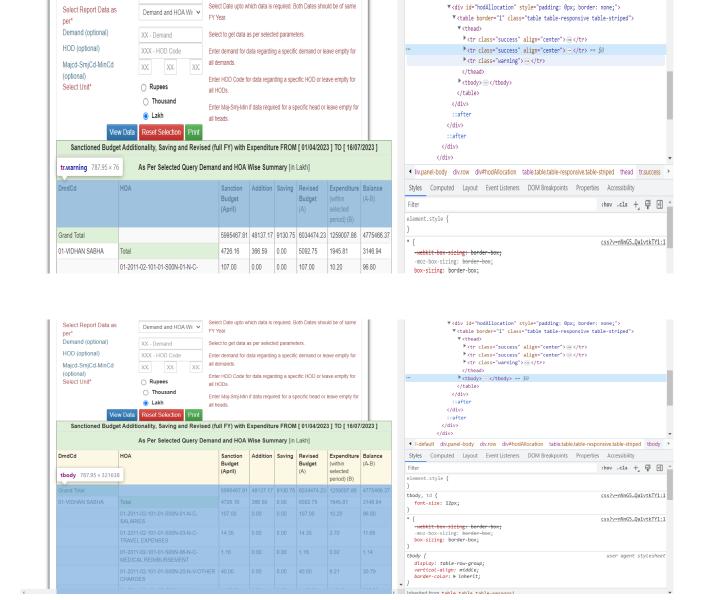
Sanctioned Budget Additionality, Saving and Revised (full FY) with Expenditure

16/07/2023

FY Year.

From Date\*

To Date\*



⇒ Then we pass the data frame to clean\_dataframe function 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 fetch 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 , we read the data from 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.

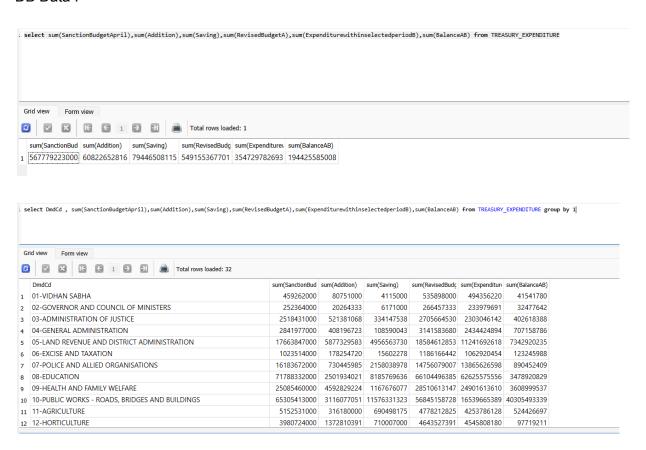
## Data Checking:

- ⇒ 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	ноа	Sanction Budget (April)	Addition	Saving	Revised Budget (A)	'	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:



# **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**.
- ⇒We need to change the project\_path in 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