

# Carbon Accounting and Sustainable Designs in Product Lifecycle Management

Prof. Deepu Philip

Department of Management Sciences

Prof. Amandeep Singh Oberoi

Imagineering Laboratory

Dr. Prabal Pratap Singh

Department of Management Sciences

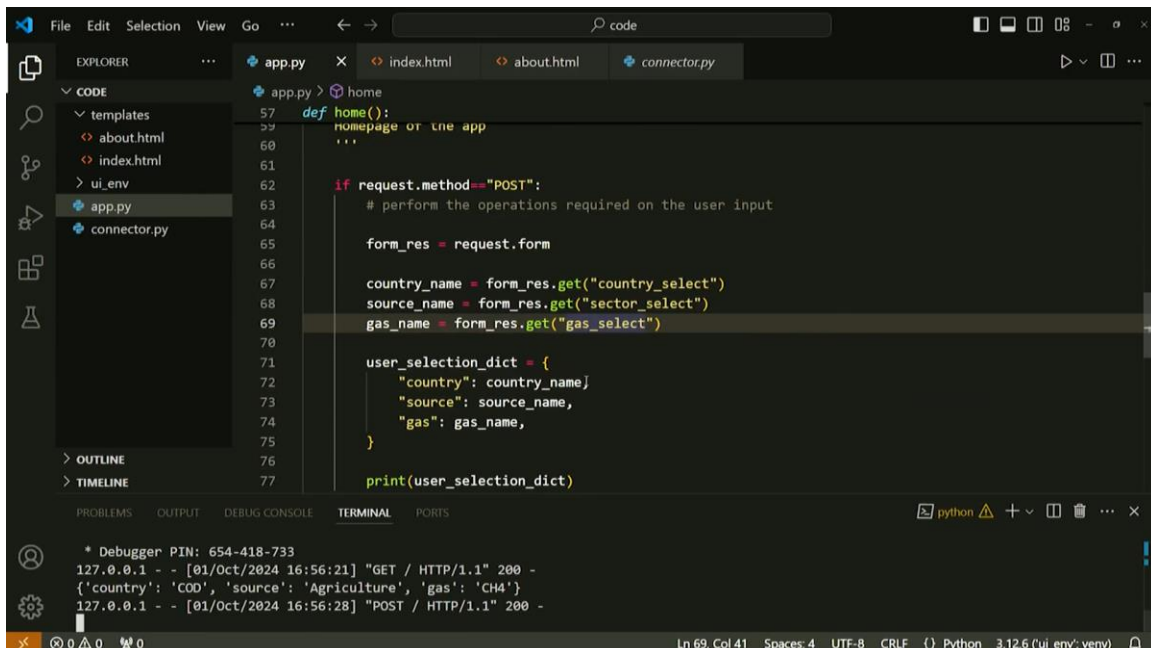
Indian Institute of Technology, Kanpur

Week 12

Lecture 54

## Carbon Accounting User Interface (Part-3)

So continuing forward let us now create the user queries.



```
def home():
    """homepage of the app"""
    ...

    if request.method=="POST":
        # perform the operations required on the user input

        form_res = request.form

        country_name = form_res.get("country_select")
        source_name = form_res.get("sector_select")
        gas_name = form_res.get("gas_select")

        user_selection_dict = {
            "country": country_name,
            "source": source_name,
            "gas": gas_name,
        }

        print(user_selection_dict)
```

Terminal output:

```
* Debugger PIN: 654-418-733
127.0.0.1 - - [01/Oct/2024 16:56:21] "GET / HTTP/1.1" 200 -
{'country': 'COO', 'source': 'Agriculture', 'gas': 'CH4'}
127.0.0.1 - - [01/Oct/2024 16:56:28] "POST / HTTP/1.1" 200 -
```

So with the user interface what the user is saying that he or she needs to find the information let's say about afghanistan. For the sector let's say agriculture and he or she is

trying to capture the details regarding co2. So these three are the conditions that needs to filter the database for the particular information that are available in the mariadb database right. So with this background we will try to create our query.

So let us store our query in a new variable with the name user query and this should be a string and it will the query will be select star from the name of the table that is historical data. And we will utilize the where clause where country equals the country name variable. Here we are trying to utilize this country name variable that has been provided through the user interface form and we have multiple conditions. We need to use the end statement and the next thing we need to choose from the database is sector. So this is source name source underscore name and guess GAS equal to guess underscore name.

Now we need to forward this query to the function that we created earlier to connect with our database, right. So we can write db underscore query. This is the function that we created and we have also imported this function in this script as well. And we can also utilize this function call directly to save some time. So the username password hostname port and the name of the database remains the same because we are still working on the same database.

But the query is now different so the query is user query and this user query is this string that we have defined here okay. Now what this function call will do is it will try to search this function in the current script and this function is defined here. So using this user query it will create a connection and will try to execute this query and whatever the rows that follows or that are available in the database based on the conditions available in this user query. It will try to fetch all those rows and try to print rows in the terminal. So let us try to check whether this much of our application is working or not.

So we can just go to our application and change this to something else try to submit. So now you can see that the terminal is saying that the connection is successful. Why? Because this function of connecting it is we have already told here that print connection successful if we are successfully connecting it to the server. The next thing is the query.

So we have selected CAN as the country, energy as the sector and all GHG emissions. And these are all the data from 1990 to 2021 that has been provided in this query. So there is this one row that is available in the data set, Okay. So now this is something, this is the retrieval aspect of our application. To move forward, we can also perform different kinds of complex modeling on the energy systems.

And instead of doing the complex modeling, we are trying to show here a very basic example of how we can manipulate the data. In the data set by first fetching it and then trying to find out various statistics based on the data available in the query. So, let us now edit our DB query function in this script and try to see how we can calculate very basic statistics like minimum value of emission, maximum value of emission and mean value of emission. So, to do that we will again try to edit our query function. And instead of printing these rows in the terminal, we will try to utilize whatever we are retrieving from the database and try to manipulate it.

So we can fetch in this function until now what we have done is we have created a cursor based on the successful connection. Then we have executed the user query and using that cursor and in the rows variable we have all the fetched values, right. Now let us try to fetch the column names. So to do that we can store all the column names in the variable column names and we can write a for statement for i in description in cursor dot discription. With each iteration what we need to do is we will capture the 0th position of I. So this will hold all the column names.

And we have already fetched all the rows in the rows variable. Rows is equal to cursor.fetchAll. Next is we can manipulate it in different ways. Currently, we will try to use the pandas package which we have installed. So, we have imported it using the pandas as pd.

So, whatever function we require from the pandas package, we will use the pd alias, right. So, let us convert the fetched values to convert to pandas data frame. So this is pd.dataframe. This is a function available inside the pandas package. And we will utilize all the rows that we captured by using this statement.fetchall.

And the name of the columns for this data frame will be the column names that we have received by running the statement using the for loop okay. And store this data frame into a different variable name as tf now we have a data frame and we can either print the values or we can store them into different variables. So different statistics. So maximum value of the emission for the required information that has been asked by the user. We have a data frame of all those rows, right.

In this DF variable. So we need to assess this DF variable and try to find out what is the maximum value out of all the emissions value provided since 1990 until 2021. So, to do that we can write df dot ilock and three dot values. Now what this statement will do on the data frame is it will try to fetch all the columns after the first three columns. why?

Because if you remember in our output the first three columns have strings that provides the information about whose emissions data we have retrieved from the database.

So all the information after these three columns are useful for our numerical analysis, right. So this statement will try to get all the numerical values into a new variable and out of all those values we need to find the maximum value. So we can simply write np dot max. So here we are utilizing the numpy package with the alias np. Similarly we can write min well or the minimum value copy the same thing and just change the function from the numpy package also we can calculate the mean well.

Now these values we can return to the basic function call of this DB query. This will be utilized and will be helpful to showcase this on the actual user interface of the system. So we can write here minwell comma max comma min and the actual data frame without the columns. So now this dbQuery function will not only fetch the data from the database it will also try to calculate these three variables. And will return wherever this function has been called it will return these four values.

So we need to capture these four values as well whenever we are trying to call this function. So we are calling this function in the home function and here is our call. So we can again write the same variable names for the function call. And since we are only capturing the numerical values from the data frame that we have fetched from the database. We can write the final data set as erd data.

This will let us know that the fourth value contains only the numerical data. Now this is this also we can store in a new dictionary stats underscore dict. And this will have a min storing the min well max value in the next one, okay. So now what we have accomplished is we fetched the data based on the user input and then also we calculated few basic statistics based on the data that we have fetched, right. Now we need to transfer these variables to the user interface so that we can show the user whatever that has been fetched or that has been calculated are available to the user for his or her quick verification.

So the first thing we want to show is what are the user selections. So we can create new variable user selection. This is a new argument and we can store the user selection dictionary data so now this render template function will send the complete dictionary that is available in the user selection deck. And this dictionary will store the country name that was selected by the user, the source name and the guest name, right. And the keys is country, source and guest.

So similarly, we have some statistics that we have calculated and this is under statistics. Now what we are telling this home function of the flask package is if the request method is post then do all of these things and render the template in this way right. But if we have not mentioned what to do if the request is not post method. So to do that we can write else return. So there are multiple return statements because we are we are classifying the working of this function based on the if else of the request that has been provided by the user.

So this return function will render the template of name index dot html. And it will have a basic user selection toggle. So I will explain what this variable will do, okay. Now since this user selection toggle should also be there in this function in this return statement with a different value. So these arguments will transfer the data that has been available in these variables to the user interface.

Now let us switch back to the user interface. So, we have created the two columns in our user interface. If you see here, this is choose emission sources and user selection. So this is the input area for the user. And this is the output that has been provided by the user.

And we can show whatever statistics that we have calculated in this column. So to do that, we need to modify this column. And this column is available in under this column two, right. And this this template is following the jinja templating engine. So we can utilize the if else statements using the jinja template syntax so all of this has been covered in the previous course.

And you can refer to those those lectures in that course to understand how these jinja templating index works. So now first of all we need to check whether the user has selected something or not. So to do that programmatically we need a variable that gets changed whenever a user selects something right. That is why we have created this variable user selection toggle. If this user selection toggle is zero that means the web application has been loaded with the get method only then it will only provide this index.html with user selection toggle as zero.

Otherwise, the value of this will be 1. And if this is 1, then it indicates that we have fetched the data from the database and calculated the statistics. Let us now create this if statement based on this templating system. So we can check if user selection underscore toggle if this is one then what we need to do we need to create paragraph. And first of all show the user what he or she has selected.

So we can write user selection that so whatever the user has selected we have stored it in the user selection dictionary and we are fetching the data from that only. The country this is the key of the dictionary. Similarly we can show other two things as well this is the source source of emissions and this is also the key name is also source, right. Next is the guess so the name of the guess the next thing we can show to the user is the emission statistics. So there are three different things that we have calculated that is the minimum emission.

And it is available in stats dict that we are transferring to the user interface and the key for this is min we can copy this. The next is maximum the third one is the average average emissions okay. Now this is the if statement which will run when the user selection toggle will have one value but what if the user selection toggle is still zero. Then we need to tell the user interface that if there are no user selections till now. Then we can write as and provide a warning to the user that select the data fields using the form available and we need to close this if statement.

Now let us check. What has changed. In our user interface. So we have not used the codes here. That is why it is not identifying that index.html is a template available.

So we can again do this. Okay. Now there is nothing selected by the user and these are all the default values that has been shown. So let us just select randomly submit. So this is the stats tick variable and we have mistyped here somewhere these are stats ticked. So user has not selected anything and let us randomly choose few inputs.

And now you can see that the user interface showing that user has selected country as ARG, source as bunker fuels and gas as CH4. And the emission statistics for this is minimum is this much, maximum and average. You can change this also. Let's say agriculture and now the data has changed. So this user interface is trying to fetch the data from the MariaDB database and then showing you all these statistics.

Further, we can also create a chart based on all the information and try to see the trend of all the data that has been provided in the dataset. So let us now further modify our script and try to create a chart based on the data set that we are utilizing. So if you remember we have included this package named as matplotlib pyplot. So this is a visualization library and we will try to create a line chart. So to do that, we need to first modify, we need to assess all the data that we are fetching from the database in the yearly data variable.

So we define this variable here, yearly data, but we have not used it. So now let us try to utilize this data. So instead of creating the chart in this home function itself, we can create a different function for visualization. So let us create a new function with the name def build\_underscore\_chart and it will take an argument as the yearly\_underscore\_data . So detaching this function will create a chart based on the data fetched from the database, okay.

And first let us initialize a figure with axis using the plt that is the pi plot dot subplots and let us create the plot on this axis ax dot plot. So with plot function you can create a line chart. And what we are trying to plot the x-axis will have the data from 1990 to 2021. So we can write list all the values between the range 1990 to 2021 and what we need to plot on these x-axis the yearly data that we have fetched from the database okay. Now let us provide the title of the x.title as annual emissions.

And x.setlabel as year and X dot set Y label as emissions. Now we need to save this figure. So we are storing this figure in this fig variable and we can utilize the matplotlib function to save this figure in our local directory and where this will be saved. So currently our app.py is in this directory and to show any kind of static data on the web application we need to create a new folder with the name static right. And whenever we fetch the data the this image will get dynamically created on the local file system inside this static directory.

So let us now provide the path for this storage of this dynamically generated figure and we can write the name of the figure as annual\_emissions\_dot\_png. And there are some more arguments that can be provided like bbox\_underscore which is equals true or equals tight and dpi equals 300. And we should return this figure. So this function will take yearly data as input and try to create this figure okay. Now we can call this function here only where we are getting this yearly data here so we can develop this figure.

We can call this build\_chart function that we have created and provide the yearly data as an argument to this function. And this will return a figure variable which we can capture in the fig variable inside this home function. Now again we can provide one more variable. That is chart\_underscore\_generator and equals 1. So whenever a chart gets generated then only this variable will be useful.

So we can create one more section in the UI of the application. Here, we can write div class of this division is container fluid this is from the bootstrap CSS. Then again we can create a div.class the class should be row and again we need to create div.class should be

col-md-12. So we are utilizing the whole horizontal space here now and the next division is div.class it should have p3 border with light background. Now we have generated our image and stored dynamically in this static folder there is nothing right now.

But whenever the user will select something the image will be generated here so to show that image into this section we need to first center this image and utilize the image tag of the html. So we can write image the source is so we are again using the ginger templating syntax. And we will be utilizing the url of the local file system. Url for static comma the file name which is the annual sub missions dot PNG. The alternative string could be chart and height should be auto but width should be 70 percent.

Now, when this chart should be displayed on the user interface? Only when the template is getting the chart gen as 1. Otherwise, we should not show this chart. So, to do that, we can again check if chart underscore gen is 1. Then only show this and now let us try to run our application again and let us choose afghanistan agriculture all ghc, okay.

So we should write 2022 here because the last value which we have is 2021. So due to python indexing system the last value is excluded that is why we are getting error so now let us try to run this again text object is not calling okay. There is one more error here while setting the title it should be set underscore title. So we have wrong name for the file missions. Now you can see that when user is selecting these three variables as the country is afghanistan energy sector and all ghg it is showing the emission statistics and the curve since 1990 until 2021, right.

Now you can check whether it is changing for different inputs or not so let's say you can use india. And let's say agriculture all ghg emissions. So now you can see that the curve has changed dynamically and these images are dynamically created in under the static folder, right. So with this, I would like to stop here and the complete code will be available as a GitHub repository for the development of the user interface. And we can further customize this user interface to create any kind and different kinds of complex energy systems modeling.

And we can utilize these kinds of user interface so that we can create a complete end to end carbon accounting system.



### Choose emission sources

Select a country

AFG

Select a sector

Energy

Select type of gas

All GHG

Submit

### User Selection

Country: AFG

Source: Energy

Gas: All GHG

Emission Statistics

Minimum: 1.158729157

Maximum: 14.93444349

Average: 6.008185752656249

### Annual Emissions



- Final Output Program

Thank you.