**Data Upload and Excel Merging using Flask Application**

Data storage and data usage is playing a vital role in present business world. Especially, when working with Microsoft Excel. The use of excel files has become common. They are used for aligning the data or records in an organized way. As the data is getting noted, they need to be updated and saved at a single place or in a single file. It is a lot easier to process data in a single file instead of switching between numerous sources [1].

Practically sometimes during gathering data or records from different resources, we may face certain difficulties, or it may become hectic. We may lose some of the information also while combining the files in the phenomena of storing all the records into one. We may find that the data has become a little hard to follow, with data sets spread across separate sheets, pivot tables, and more. We don’t always need to use multiple worksheets or Excel files to work on the data, however, especially if working as a team [2].

In order to overcome few of the hurdles during merging the files, we can plan a too or website where a user can upload their first file and save it into the database. When the user uploads the second file, this gets appended to the first file and this merged file gets saved to the database again. The user also can download this merge file into their local drive.

In order to facilitate the excel file merging, there are various ways available in the technology. One of the below techniques may be used to combine multiple excel files data into one [3].

1. Copy the cell ranges  
2. Manually copy worksheets  
3. Use the indirect formula  
4. Using simple VBA macro  
5. Automatically merge the workbooks 6. Use the Get & Transform tools

As mentioned above, these techniques can be applied to combine worksheets in an excel file and also few are applied to combine the excel files as well. These are the processes which are to be performed manually by the user. But in this project, user needs to just upload the file directly which gets saved to the database as a table.

**Platform**

In this project, python is used. In addition, the following applications are used for front end and the backend [Figure 1]. They are namely:

1. **Front end Technology:**

**A webpage is developed using HTML, CSS and JavaScript where user will be able to upload the**

**files and get the information about the page.**

1. **Backend Technology: Flask Application:**

Flask is a small and lightweight Python web framework that provides useful tools and features for building web applications in Python [4]. Because you can create a web application quickly using only a single Python file, it allows developers more flexibility and is a more accessible framework for new developers. Flask is also extensible, as it doesn't require or force a specific directory structure.

**3. Database technologies:  
a) SQLAlchemy:**SQLAlchemy is a well-known Python database toolkit and object-relational mapper (ORM) implementation. SQLAlchemy is a generalized interface for writing and running database-agnostic code without writing SQL statements. Flask SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to the application [5].

**b) PostgreSQL/ pgAdmin:**

PostgreSQL is an open-source object relational database management system with features designed to accommodate a wide range of workloads, from single machines to data warehouses to web services with multiple concurrent users [6]. PostgreSQL is a relational database management system that uses and extends SQL (hence the name) and is extensible to a wide range of use cases beyond transactional data.

PostgreSQL, like other RDMS is used for accessing, storing, handling the data in the form of database tables. It supports modern application features like JSON, XML, etc.

pgAdmin is a popular open-source administrator and development platform for PostgreSQL.

**Figure 1:** Technologies used for the application.

**Project**

As per the requirements, the workflow for the project should fulfil CRUD operations i.e., creating, reading, updating, deleting the table in the database [Figure 2].

Python scripting is used in order to develop this webpage. Mainly Flask application is used. It recognizes or accepts the webpages or HTML files when they are placed in a “templates” folder. The below html and other python files are created and explained below.

**Index.html:**

In order to make user upload their files, a webpage with HTML forms, JavaScript and CSS is created. In this html file, a form to submit the files. JavaScript is used to display the uploaded files. The “action” is the main attribute which plays a key role to connect to the python application in the backend. [Figure 3a, 3b]

page3image66935808page3image66939504

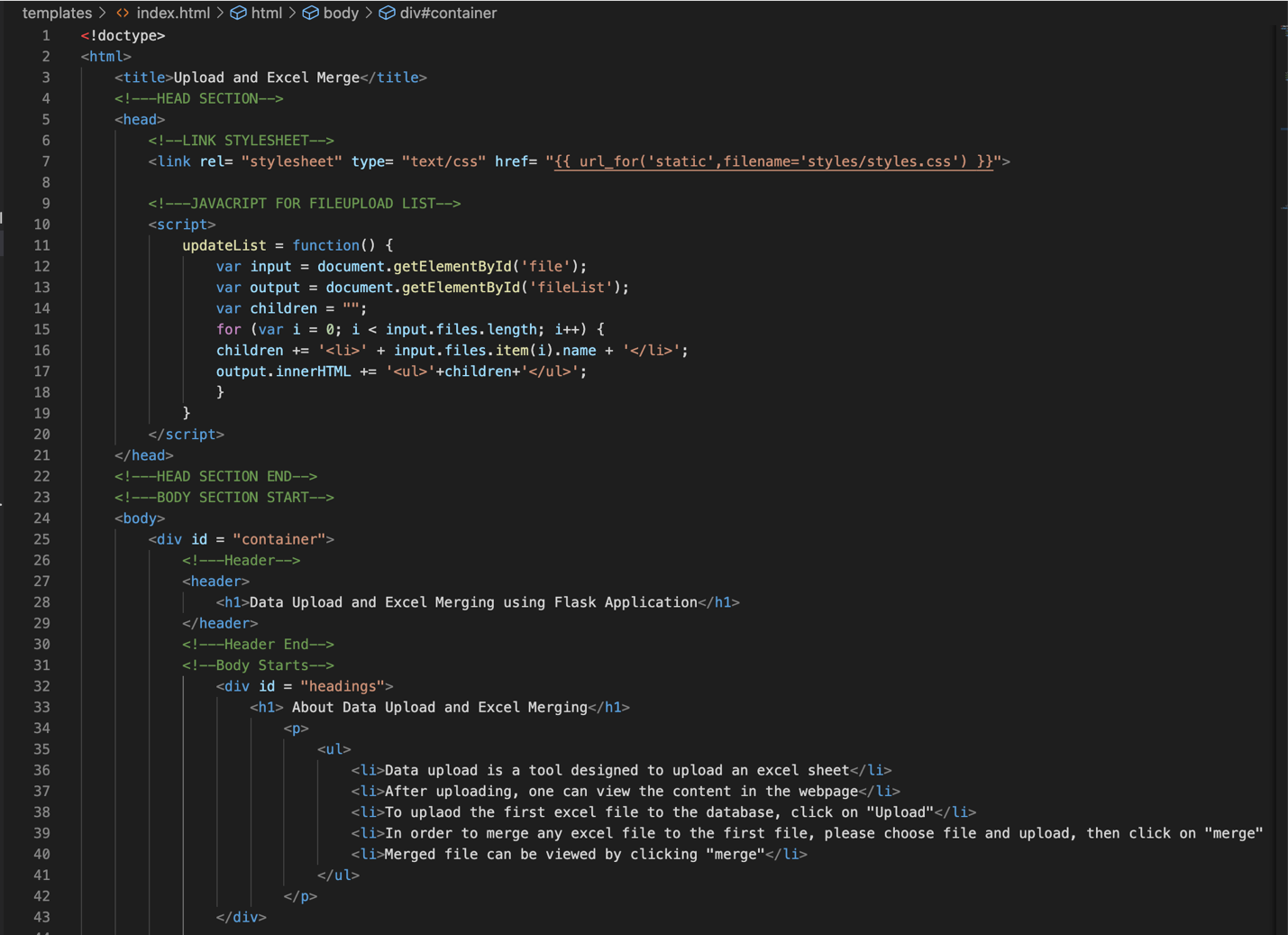


Figure 3a: Explaining the index.html first part.

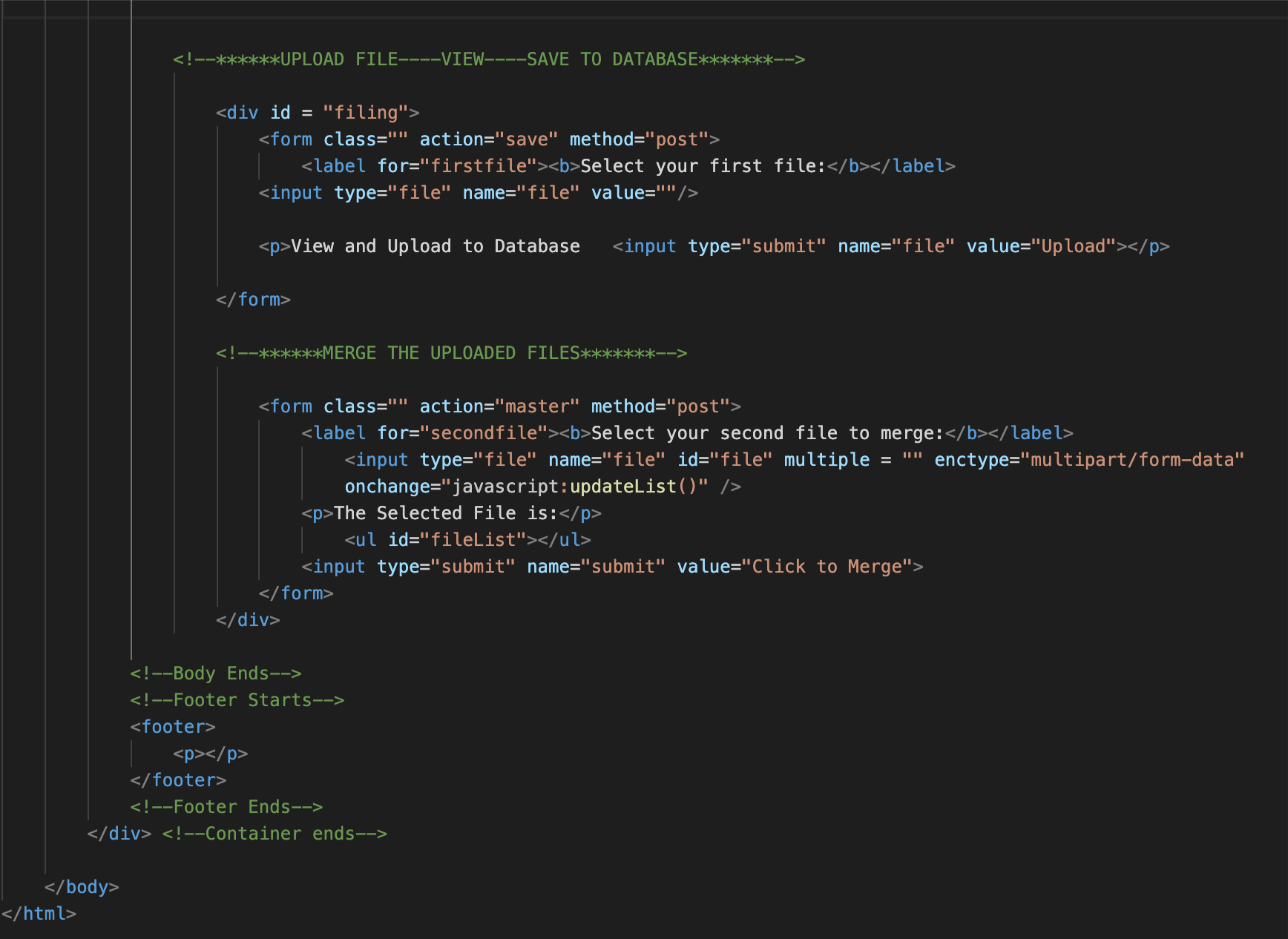


Figure 3b: Explaining the index.html second part

**styles.css:**

Cascade style sheet (CSS) is used to add color and style to the webpage created. It is applied to the table also [Figure 4].

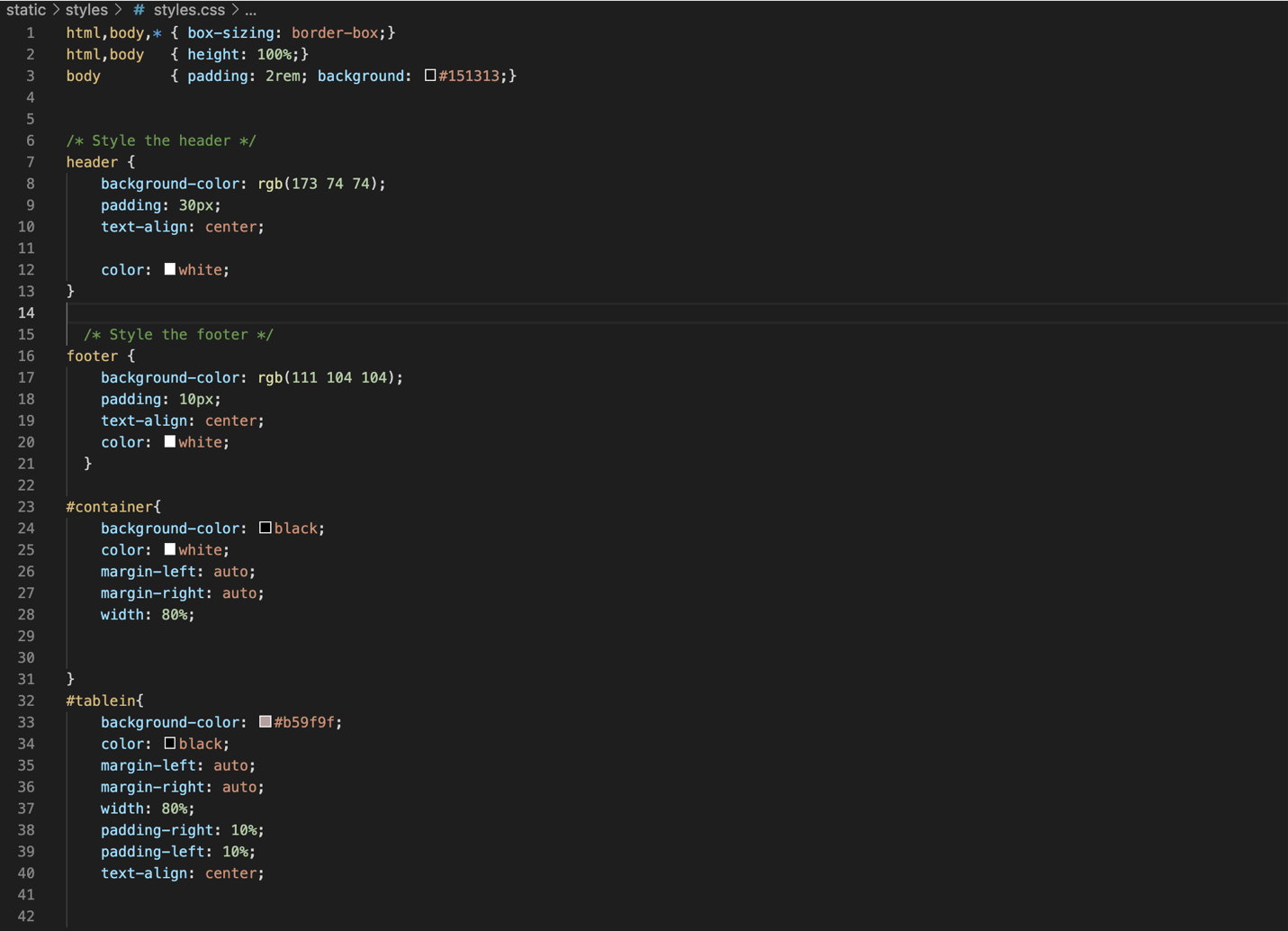


Figure 4: Explaining the CSS file

**app.py:**

Initially for the python flask application, we need to install certain libraries. All these are imported and then flask app is initialized. This file is named as “app.py”. Flask application user handlers to redirect to different html pages when mentioned with “return template” keyword. [Figure 5]

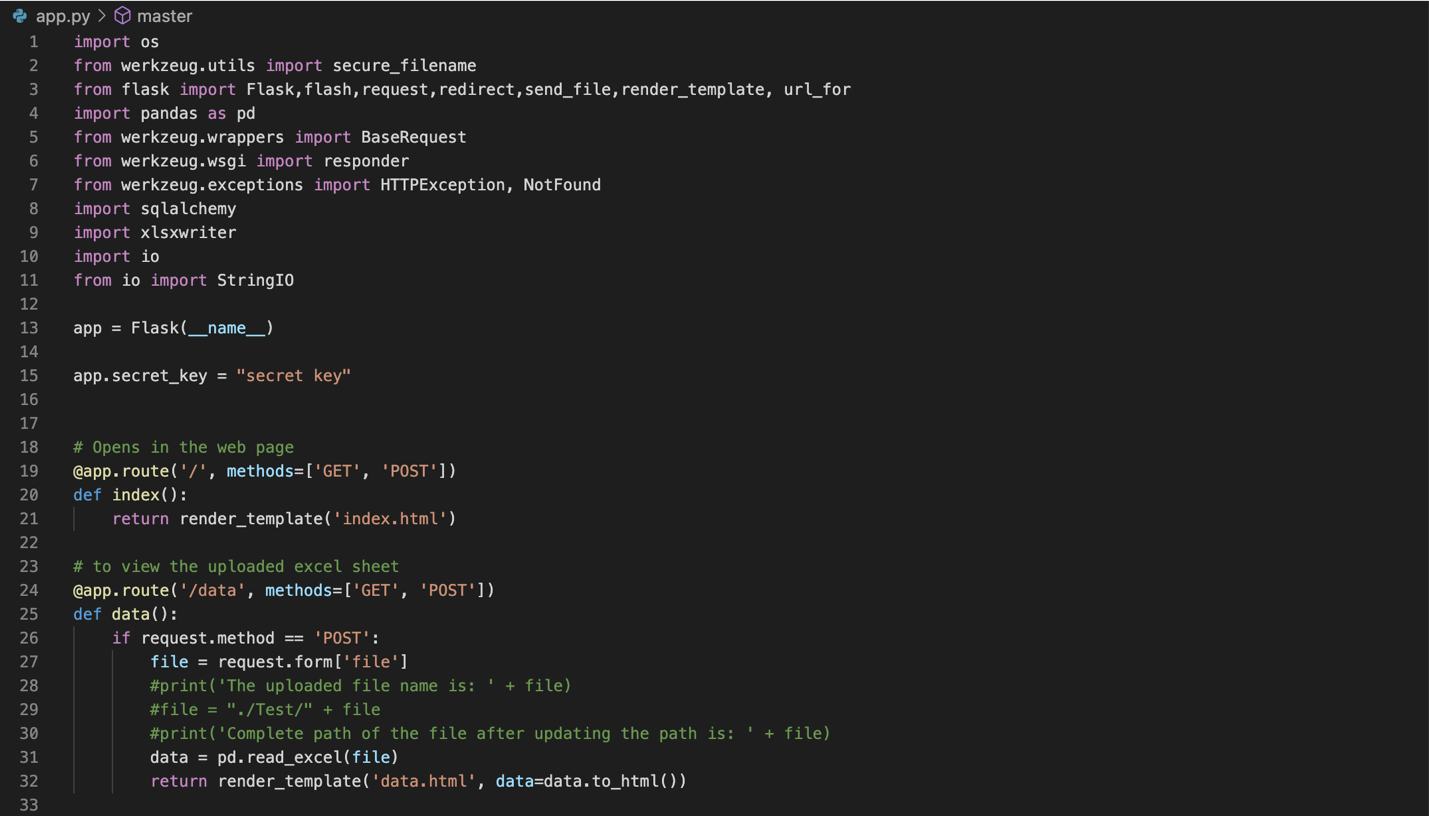


Figure 5: Showing app.py with libraries import

After creating webpages and importing libraries into the flask app, next challenge is to connect to the database. The connection to database involves importing SQLAlchemy, psycopg2 and pandas from libraries.

As soon as the user submits the file from the form, the flask app takes the file as an object. Since the file submitted is an excel, it should be read into data frame by pandas to make any processing in the python. This can be operated by the below command.  
df = pd.read\_excel(file)

After converting the file, we should connect to the database using SQLAlchemy and connecting to it from the below line.

Next step is to create the table in the database so that the records in the excel file will get saved in the table. If the table already exists, it gets replaced like the below.

We will close the connection to database by using

con.close()

|  |
| --- |
| engine = |
| sqlalchemy.create\_engine("postgresql://postgres:admin@localhost/m |
| erge")  con = engine.connect() |

page7image35370496

table\_name = 'mergetable'  
df.to\_sql(table\_name, con, if\_exists='replace')

**data.html:**

When the file is submitted and saved to PostgreSQL, the page redirects from index.html to data.html [Figure 6] where the excel file is viewed.  
return render\_template('data.html', data=df.to\_html())

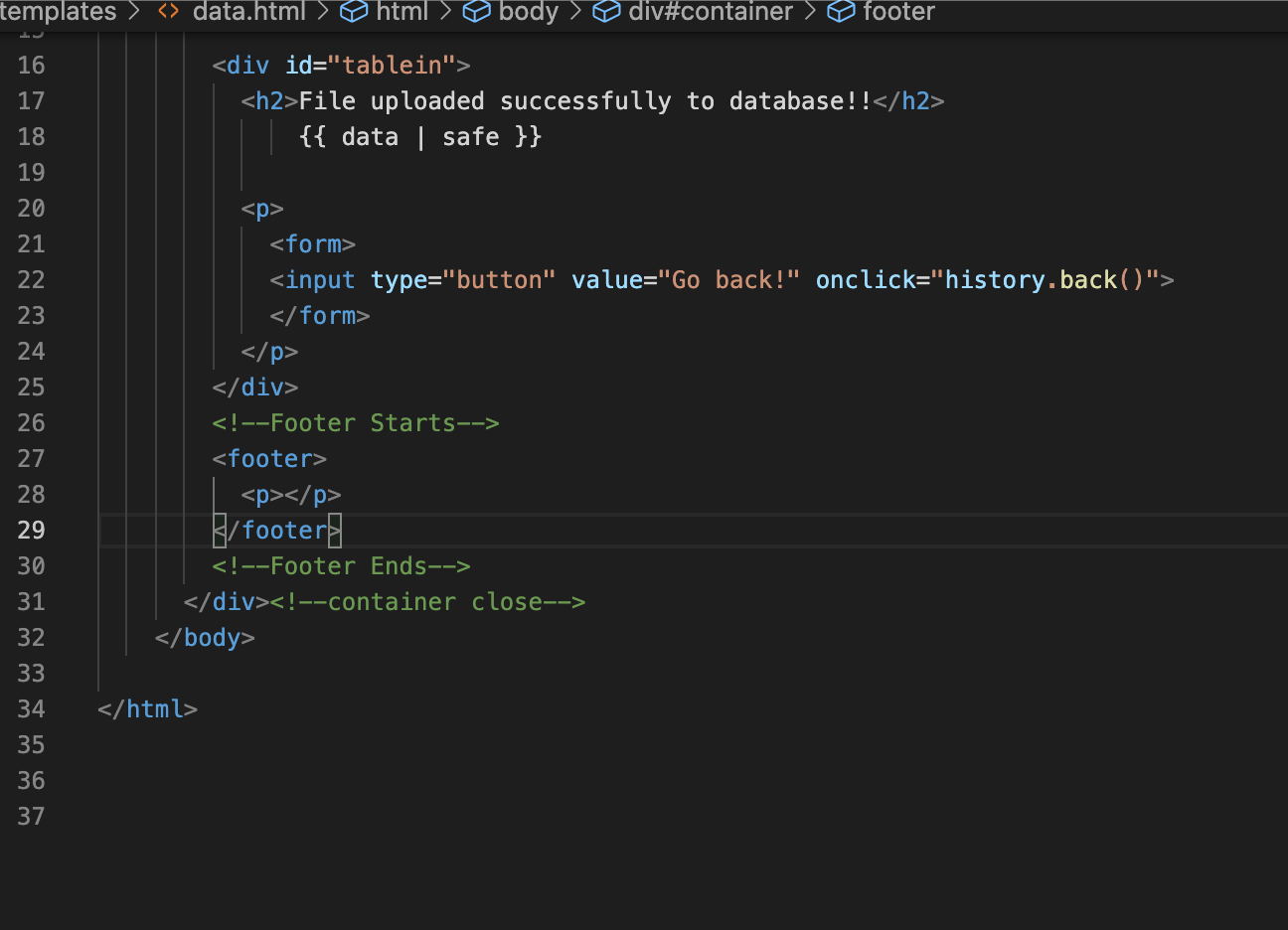


Figure 6: Showing the uploaded table in data.html

**Merge.html**

Similarly, when the second file is uploaded, it gets appended into the existing table and the resulted merged file is displayed in the merge.html [Figure 7].

table\_name = 'mergetable'  
df1.to\_sql(table\_name, con, if\_exists='append')  
result = pd.read\_sql("select \* from \"mergetable\"", con) con.close()

return render\_template('merge.html', data=result.to\_html())

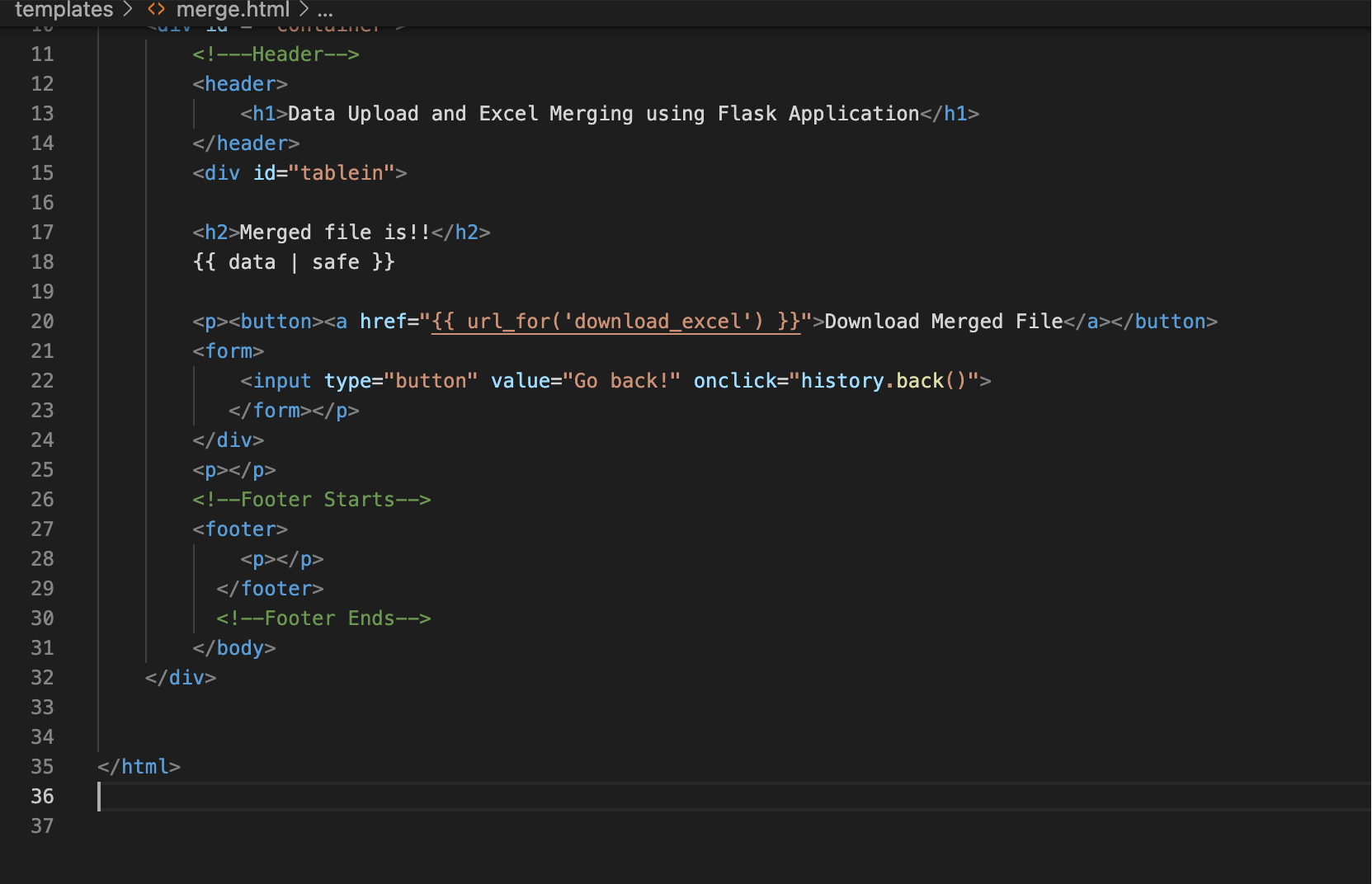


Figure 7: Showing the merged table

After saving the merged file, user should be able to download that in the form of excel sheet. pd.ExcelWriter is a library imported feature which reads the records from table in database to excel file. This can be depicted as below:

def download\_excel():  
engine =

sqlalchemy.create\_engine("postgresql://postgres:admin@localhost/test") con = engine.connect()  
result = pd.read\_sql("select \* from \"sleepdata\"", con)  
strIO = io.BytesIO()

excel\_writer = pd.ExcelWriter(strIO, engine="xlsxwriter") result.to\_excel(excel\_writer, sheet\_name="sheet1") excel\_writer.save()  
excel\_data = strIO.getvalue()

strIO.seek(0)

con.close()

return send\_file(strIO,

attachment\_filename='master.xlsx',

as\_attachment=True)

Thus, when the app.py file is run in the terminal, the webpage gets connected and displays with the localhost link. The command line for running flask application is

FLASK\_APP=app.py FLASK\_DEBUG=True flask run

It gets opened in the below link:

http://127.0.0.1:5000/

**Result**

The below functionalities are fulfilled in the project.

1. **Upload file**:  
As the file gets uploaded from the frontend, this file should get submitted to the database. This step is done by the JavaScript [Figure 8]

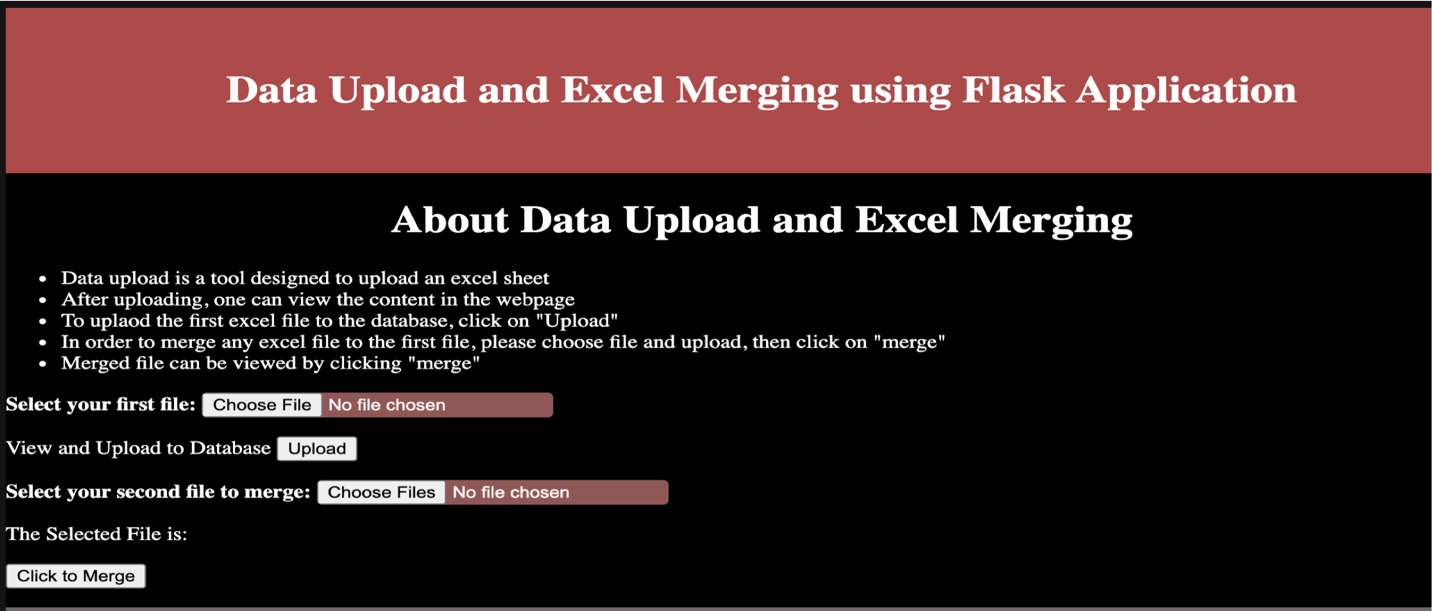


Figure 8: The webpage allowing user to upload first file Once first excel file is uploaded and submitted, the table is displayed [Figure 9].

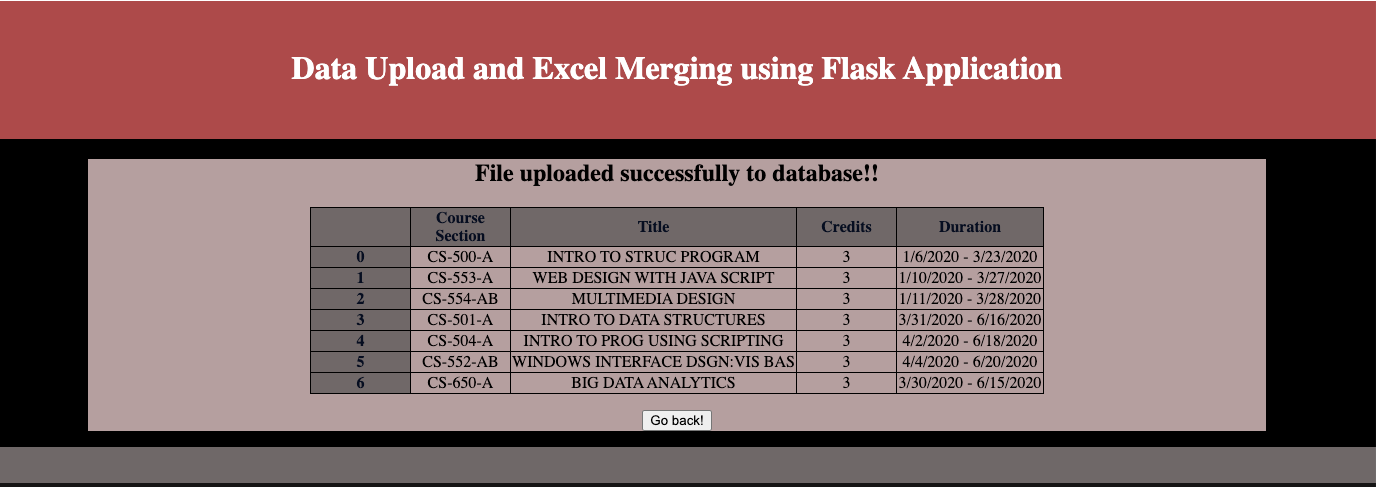


Figure 9: The webpage showing the table which is uploaded by user

**2. Save to PostgreSQL:**

As soon as the user clicks on “upload” button, a new table is created in the database. If a new user uploads the file for first time, new table is created replacing the existing one. The excel file is saved as a table in the Postgres database [Figure 10].

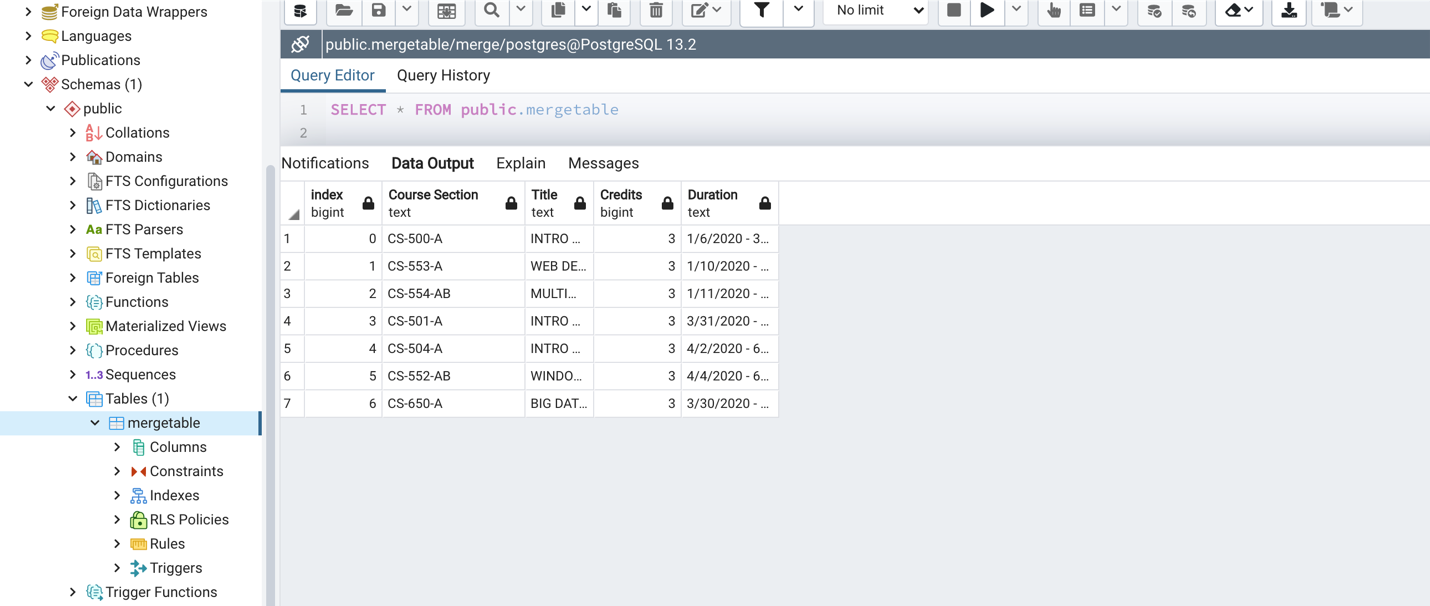


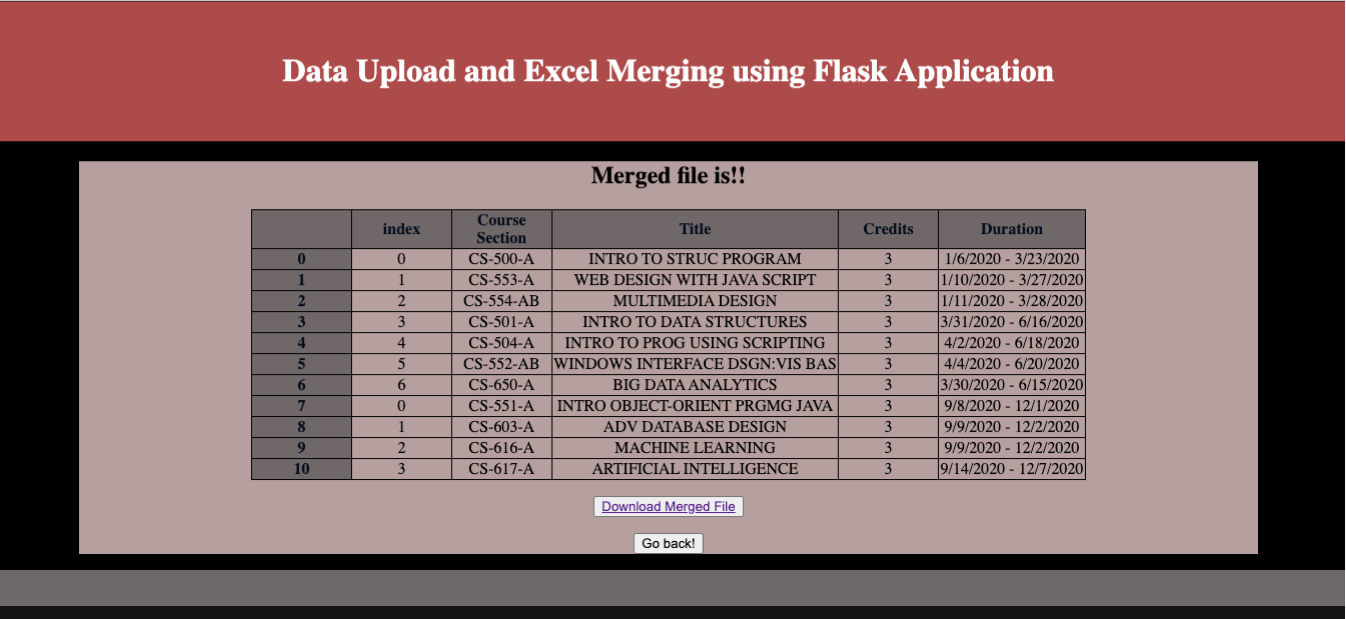
Figure 10: The table gets saved in postgreSQL (shown in pgAdmin4)

1. **Upload second file:**

**If the user wants to merge any other excel file with the same columns into the first submitted excel file, they can upload it in the next section.**

1. **Append to first file:**

**As soon as the second file is submitted, it gets appended to the first submitted excel file in the table in the database. This is saved in the database.**



**5. Save Merged file:**

Figure 11: The webpage showing the merged table

The merged file gets saved as the table in the database [Figure 12].

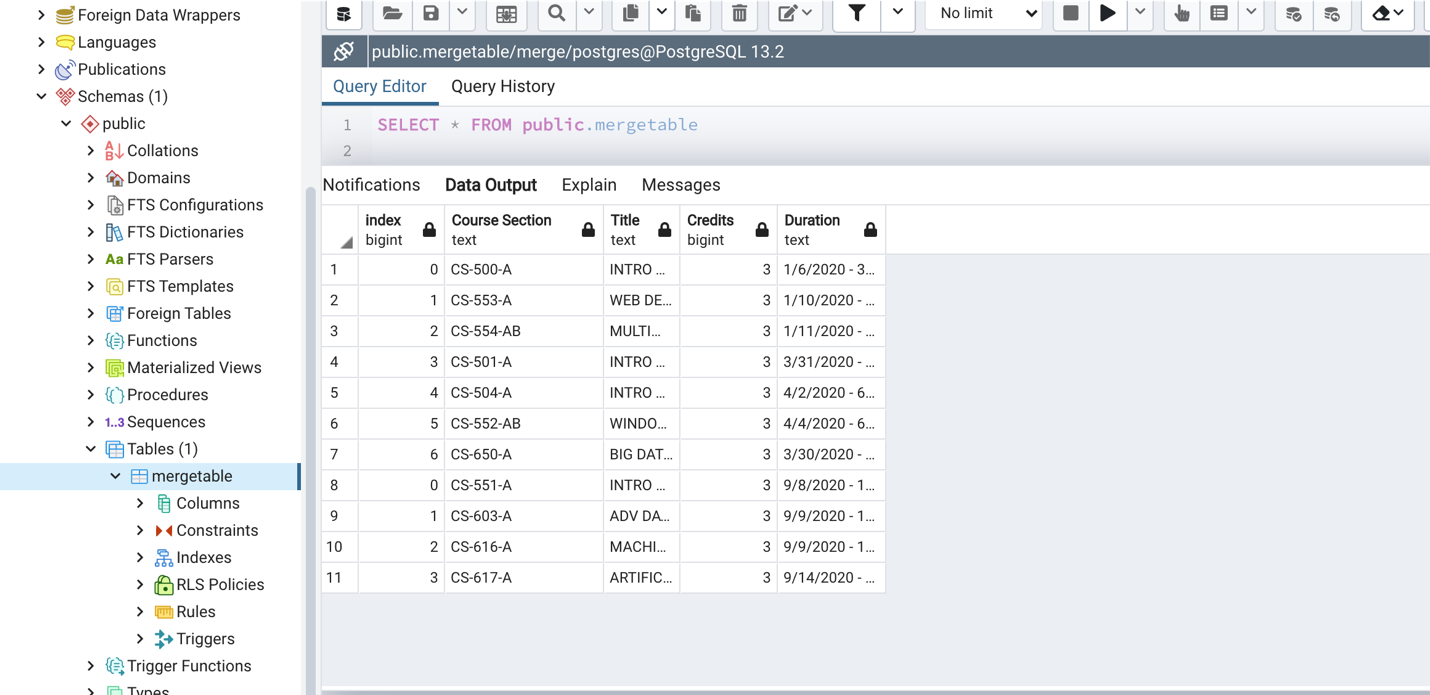


Figure 12: The database showing merged file

**6. Download Merged file:**

The merged file can be saved to the local drive in the excel format [Figure 13].

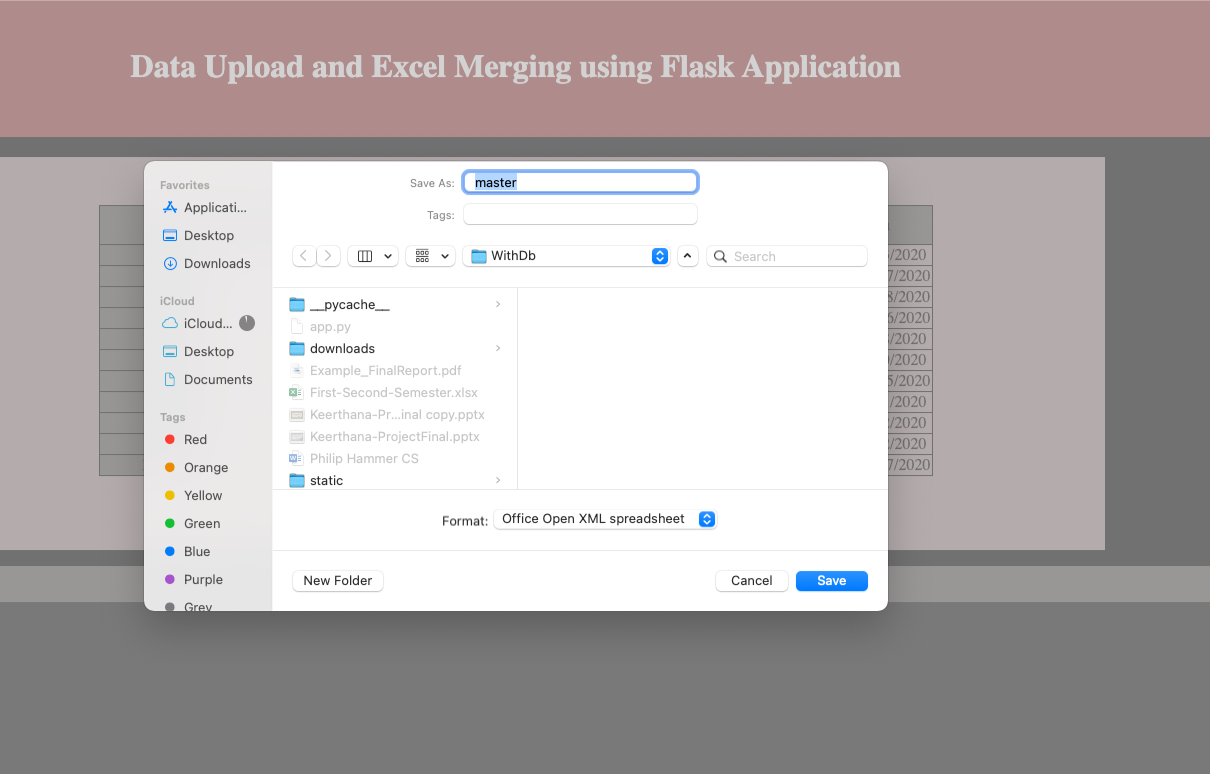


Figure 13: The webpage allowing user to download the merged file

**Conclusion**

On the whole the project was good. Initially, the plan was to develop a website where multiple files are to be given as input and result in a merged file. The plan also included that the website should accept any type of files. Usage of flask application which is a recent technology was wonderful. Python scripts and applying different libraries, databases made me gain much knowledge. The connection to the Postgres database has been challenge. But it was very easy and interesting after connecting the front-end submission to the database table creation. With this application, learning the CRUD operations was made possible. The libraries like panda, excel read and write are very advanced and makes the programming easy. Installing and importing various other libraries made to understand that python made the programming a bit easy with its in built functions additionally.

The project can be improved to develop a website where multiple types of files can be submitted to give a merged file as an output. Also, to include a search bar so that user can download only specific columns from the merged file.

Application can be used in various organizations, schools or offices to update employee records, research updates, student admission records etc.

**References**

1. Svetlana Cheusheva, How to merge multiple Excel files into one.  
2. Ben Stockton, How To Merge Data In Multiple Excel Files.  
3. Henrik Schiffner, Merge Excel Files: How to Combine Workbooks into One File. 4. Abdelhadi Dyouri, How To Make a Web Application Using Flask in Python 3.  
5. Flask-SQLAlchemy 2.5.1  
6. John Hammink, An introduction to PostgreSQL