**Data Analysis & Visualization**

The project is made by 7 python files named:

* final\_dashboard.py
* table\_layout.py
* dashboard\_layout.py
* global\_file.py
* data.py
* stats.py
* compare\_layout.py

**FINAL\_DASHBOARD.PY FILE:**

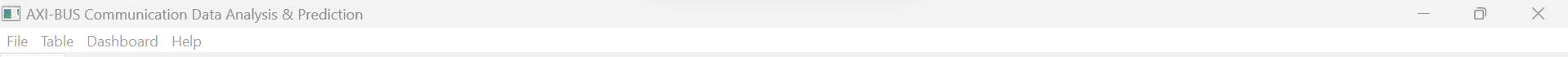
****

Figure .1

The final\_dashboard.py file makes a menu bar on the main window with contents “File”, “Table”, “Dashboard” and “Help” as shown in Figure 1.1. The file has functions in the syntax of “{function-name}\_action\_triggered” which are associated with the actions of the menu bar.

By default: **stacked\_layout.setCurrentWidget(global\_file.table\_obj)** sets the layout designed in the table\_layout.py file in the main window below the menu bar.

**TABLE\_LAYOUT.PY FILE:**

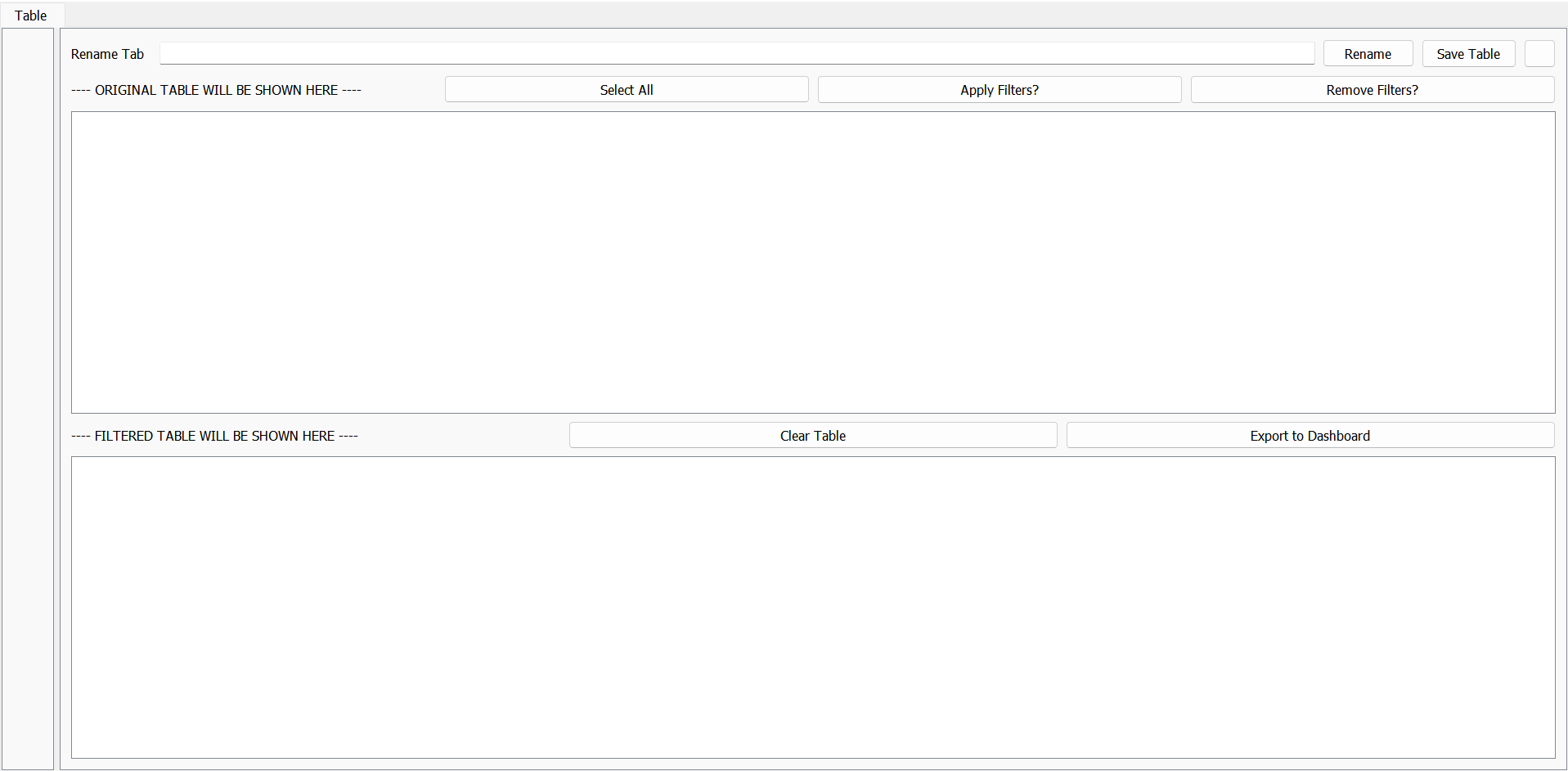
****

Figure 1.2

Figure 1.2 is the layout made in the table\_layout.py file specifically in the **“add\_tabs”** method. It has a couple of methods defined initially, all necessary to carry out the functions from the buttons present on the UI. One of the most important methods in the file is: **“highlight\_cells”**, a method which filters the table according to your preferences chosen (demonstration in Figure 1.3). A new class is also defined **“Filter\_popupBox”** to design the layout of the filter popup box as shown in Figure 1.4.

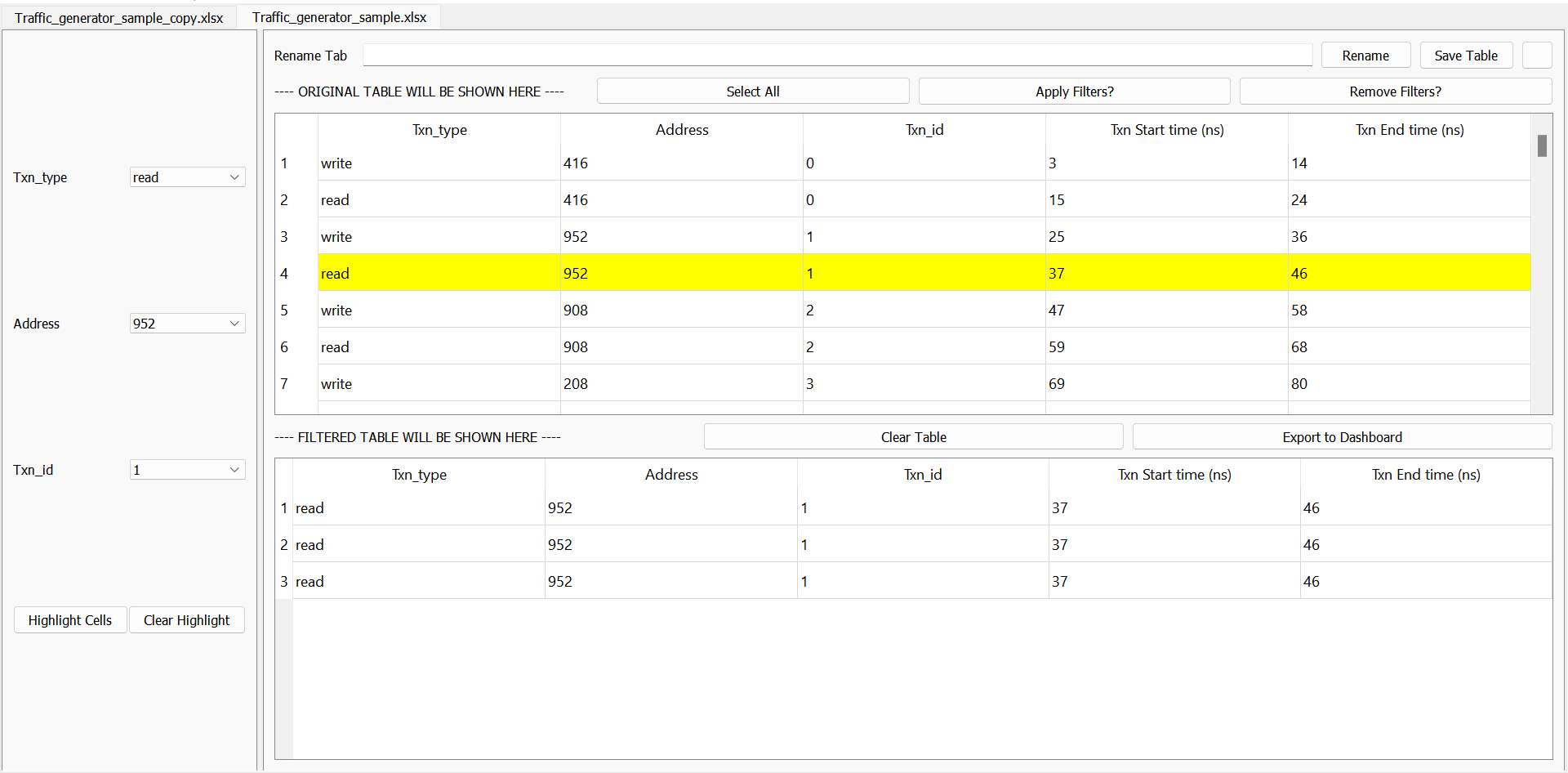
****

Figure .3

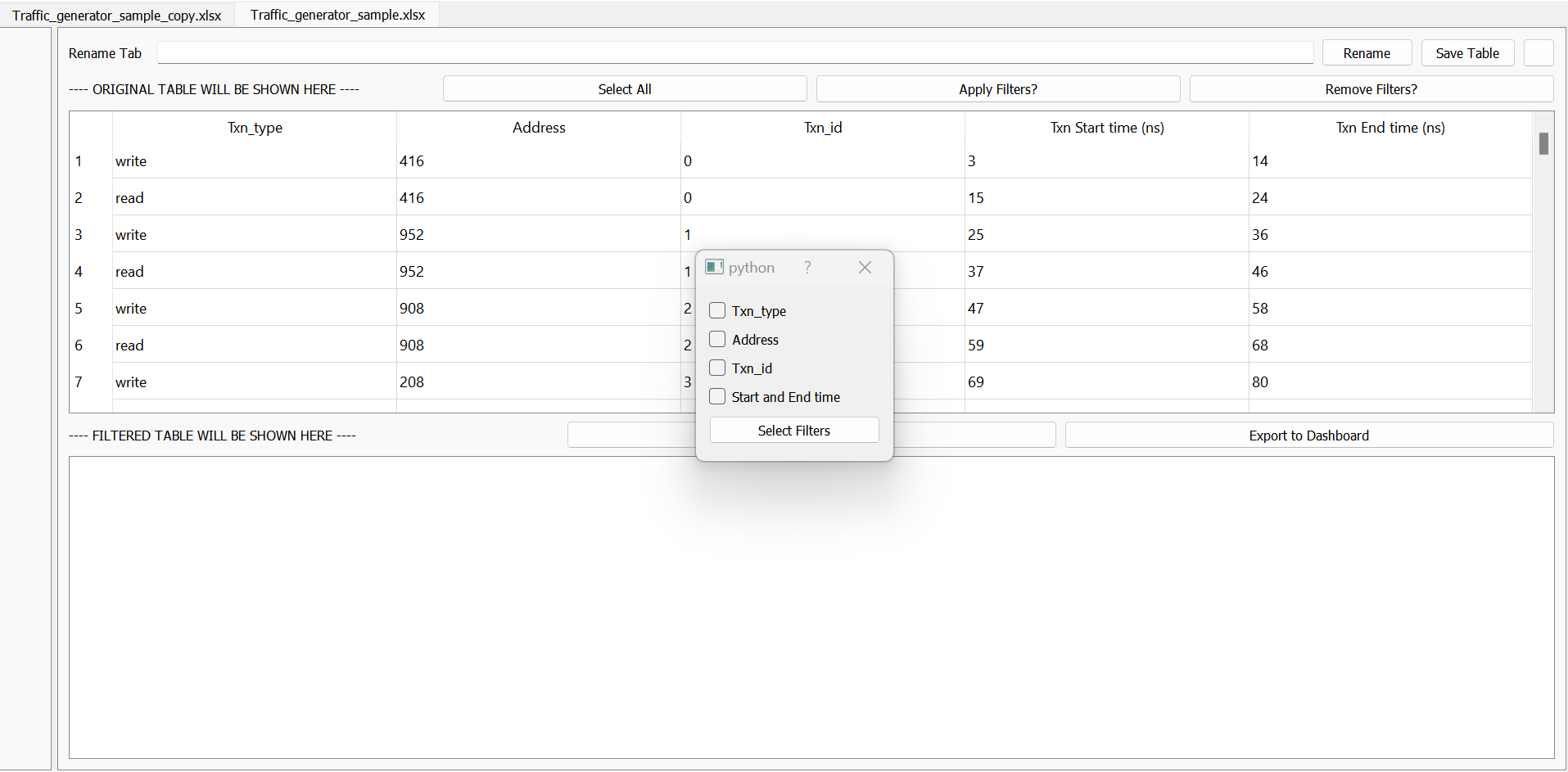


Figure .4

The button “Select Filters” seen in the popup box of Figure 1.4 is, when clicked, calls a method **“select\_filters”**. This calls another method **“filter\_visible”** which sets a new layout on the left side of the window as seen in Figure 1.5.

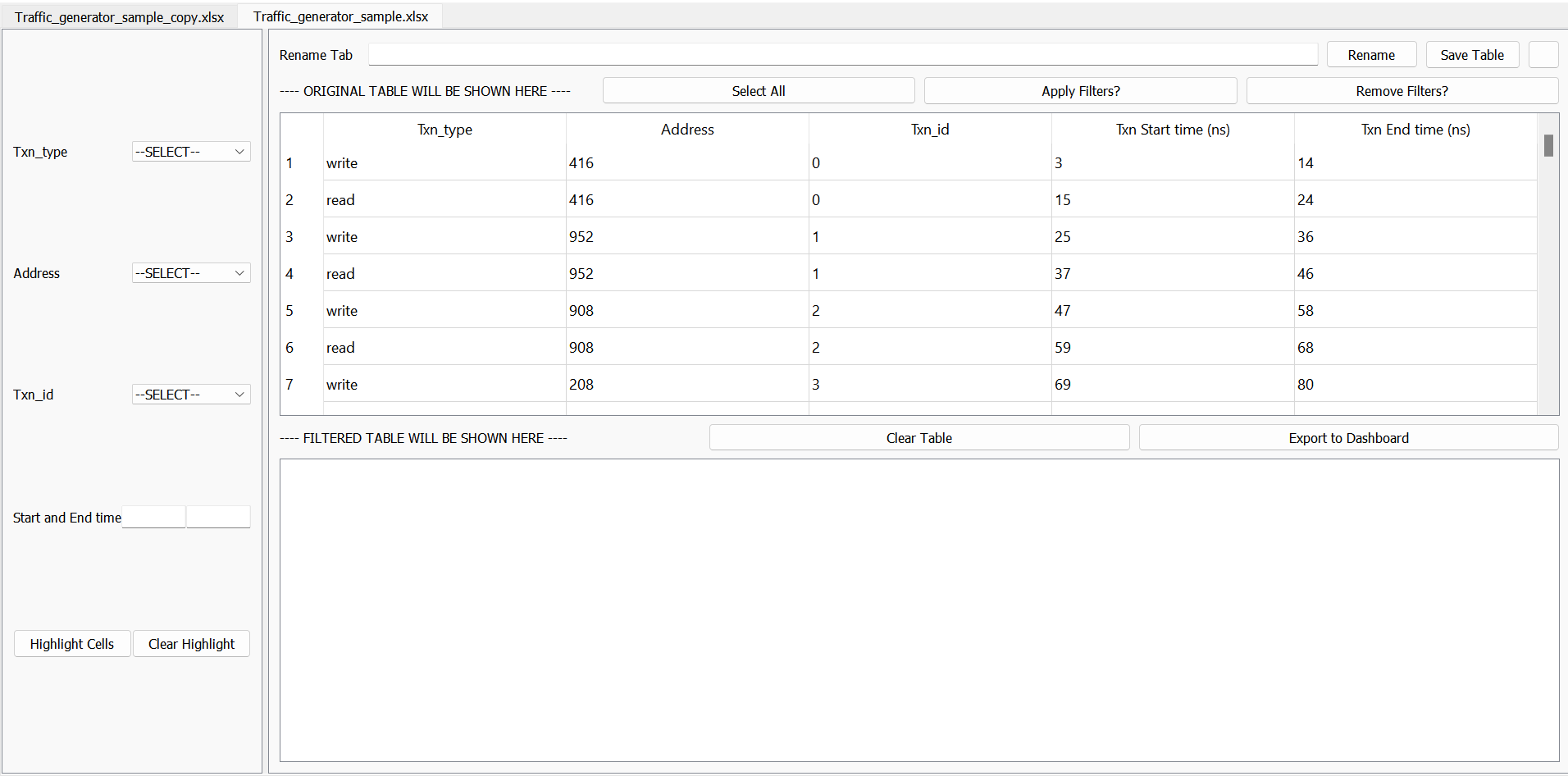


Figure .5

**DASHBOARD\_LAYOUT.PY FILE:**

The initial design of this file is shown in Figure 2.1.



Figure 2.1

Once you export any table with “Export to Dashboard” button in Figure 1.5, you can see the exported table in the dashboard UI as shown in Figure 2.1. This file is meant to perform all the data analytics part of the project, where you can select any filtered and imported table (from the table\_layout) and either read the statistics or even choose a graph to plot graphically.

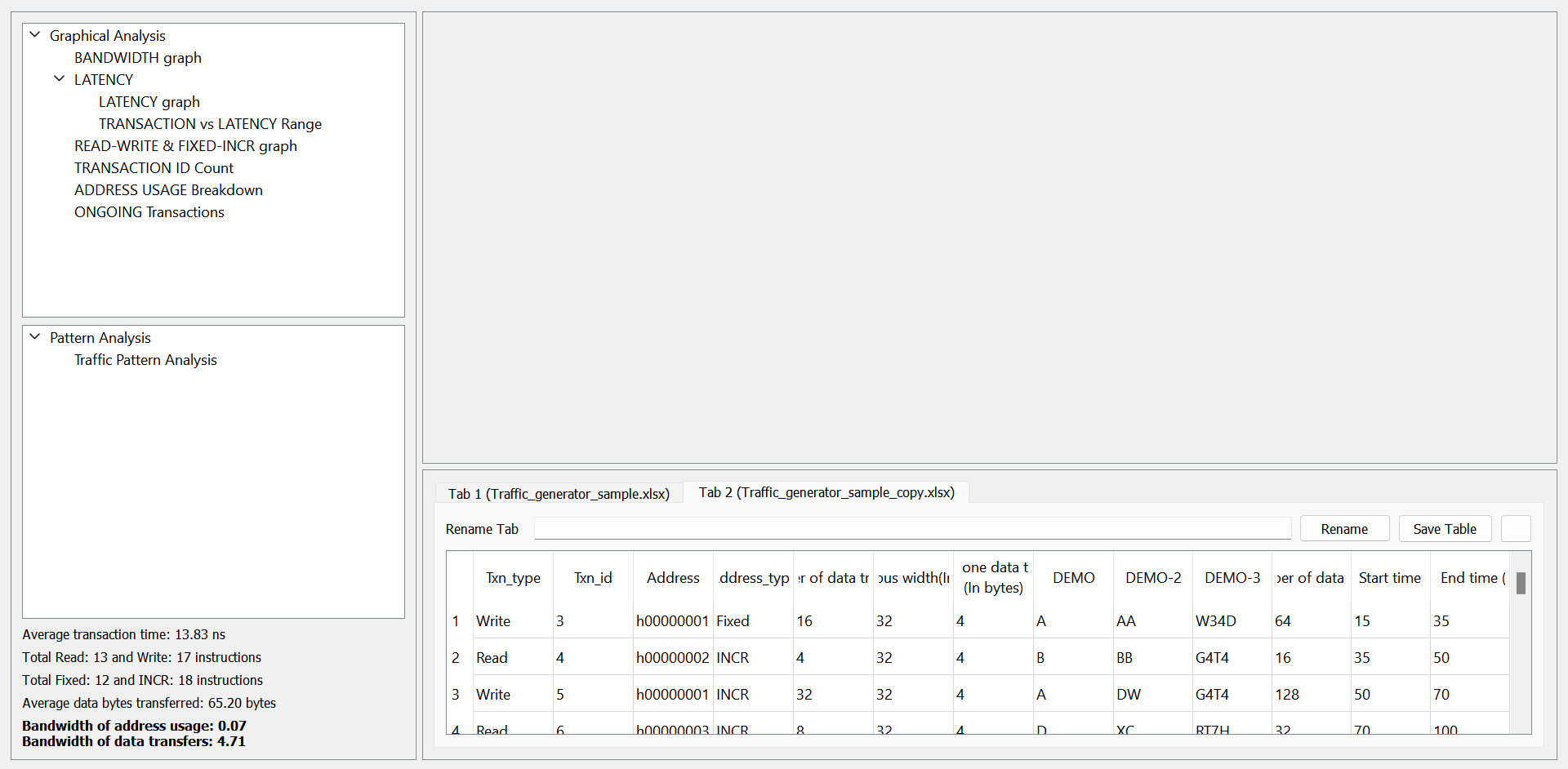


Figure 2.2

**self.dynamic\_graph\_pane** is the main (top-right) frame of the window, where all the plots will be visible. **self.dynamic\_graph\_pane** is a QVBox which contains 2 layouts: “**self.pattern\_diff”** and **“self.dynamic\_pattern\_pane”**. During Graphical Analysis, as shown in Figure 2.3, self.pattern\_diff is null/empty. And the plot is set on self.dynamic\_pattern\_pane. In plots where we need to input, for example: interval length as shown in Figure 2.4, self.pattern\_diff contains the Label, Text-field and the Buttons and self.dynamic\_pattern\_pane will still contain the plot.

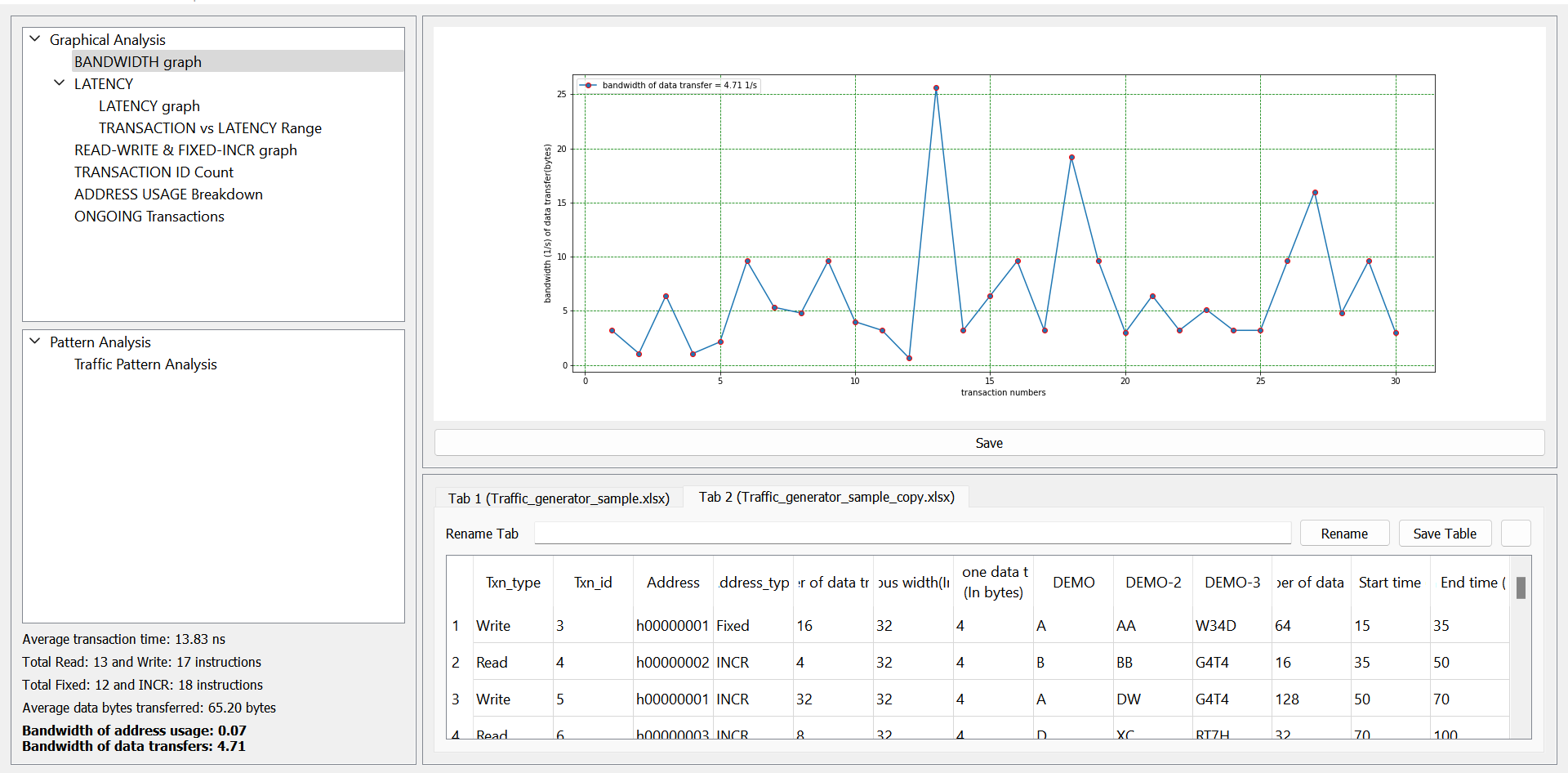


Figure 2.3

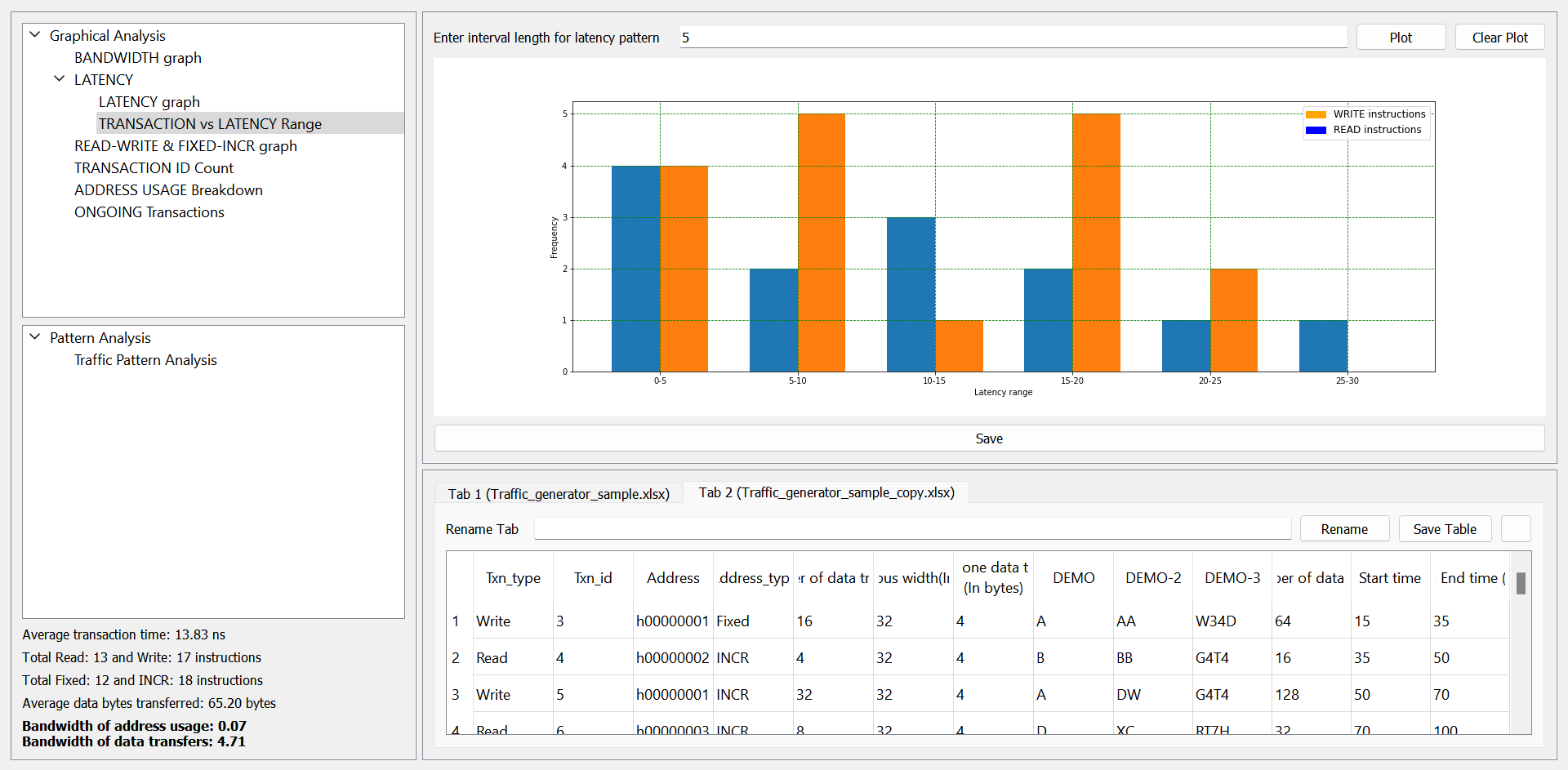


Figure 2.4

**GLOBAL\_FILE.PY**

This file instantiates object of table\_layout, dashboard\_layout & compare\_layout file classes. I used “import global\_file” in different files to access these objects. These class objects cannot be instantiated in the middle of a file because when any {}\_layout file reruns to show the UI again on the screen, a new object will be made, and all the data will be lost. To prevent this error, everytime the python interpreter will access the same object (instantiated only once) in the global\_file.

**DATA.PY**

This file contains all the methods which uses **“matplotlib”** to plot different line graphs, bar graphs or pie charts. All the functions return 4 parameters -> (FigureCanvas, list, list, list). FigureCanvas sends the plot to display it on the dashboard\_layout. The 3 list (array) contains the values of the x\_axis and y\_axis to successfully implement the functionality of save image in the form of “.txt” file.

def {function\_name}(self,data) -> FigureCanvas, list1, list2, list3:

list1, list2, list3 contains the values on x\_axis, and y\_axis (which is in some cases divided in “read” and “write” instructions):

list1 returns the values of the x-axis.

list2 and list3 returns the values of y-axis.

If y-axis is bifurcated in 2 parts, for example: read & write instructions, then list2 contains “read” and list3 contains “write”.

If there is only 1 value plotted on the y\_axis, then list2 contains that values and list3 will be null.

**STATS.PY**

This file returns all the statistical data of a table. The stats which can be seen on the bottom left side of the dashboard\_layout as seen in Figure 2.4 is calculated in “stats.py” file.

**COMPARE\_LAYOUT.PY**

It is a template for the people who will be working on this project. It does not contain any layout yet.