For my second assignment for Sound Design and Sonification, I decided to sonify 2 datasets at the same time running on a time axis. I chose data representing the number of registered Facebook users there were in the world over a period of 10 years, contrasting against the number of US colleges that were not represented on Facebook over this same period of time. I initially decided on these two datasets as I thought that the data alone told a clear story. Wanting a dataset that had clear contrasting values to further create an abstract sonification, I decided to use these two datasets. I also wanted to experiment in creating scale and exaggeration with the datasets, and so these fit in perfectly.

The purpose of my sonification patch is to create an abstract representation of the two datasets I chose, through sound. The patch takes the datasets as inputs and manipulates that data in such a way that a sonification of the data is outputted. The purpose of the sonification patch is also to allow users and viewers a level of interaction with the patch, allowing further creative use and experimentation with the different options available for use and manipulation, or using different datasets altogether.

My completed patch is set out in a way that is user-friendly and follows a linear, progressive structure that aids the user's understanding and use of the patch. I have labelled every object and left a comment on every part of the patch to further assist in the correct usage of the patch, as the patch does at first glance look quite complex.

In regards to operating the patch, firstly the dataset that I have provided in the submission has to be
loaded into the patch. The dataset is in a text file and this is loaded into the coll object which is
found 3 objects down from the toggle with the comment “Select to Start”. Having this coll object
also allows the user to load their own datasets in the future if they wish.

Once the dataset is loaded into the patch, the user then only has to turn up the volume slider located
at the bottom of the patch and then tick the “Select to Start” toggle located at the top left of the
patch. The patch will then cycle through the data rows at .35 second intervals. This interval can be
reduced or extended at the user's discretion by adjusting the number object connected to and located
above the metro object. A Bang button is present underneath the metro object to allow for manual
cycling through data rows, otherwise giving off a flash every time a new row of data is read into the
patch. Underneath the coll object is an unpack object which is required to 'unpack' the data that is
inputted from the coll object into distinct columns, discriminating between the different types of
data being read, in my submission case 3 columns of data representing the year, registered users in
millions and unrepresented colleges in hundreds.

As the year is not as fluctuating a variable as the other two data attributes, I decided to keep the year
values impartial to the sonification process and therefore kept the year as a numeric value in plain
view for the user to see at any point during use of the patch.

The two other values however are sent through a gauntlet of objects manipulating the raw data
being imported. In general this process is used to transform the raw data from integers into signals,
or numbers into sounds. Since this entire patch is one of sonification, this is a required process in
order to create any audio elements at the end of the patch. The two data attributes are sent through
identical streams of conversion. I did this to keep the end product as understandable as possible. I
did experiment with two completely different transformation processes for each of the two data
types but it became difficult to tell what was happening to what data type, and I wanted to spare any
needless confusion. This way as well, it creates a seamless transition into sound, which then has to
be manipulated after the transformation process is complete.

Each data type is multiplied by a factor of 150, effectively to raise the values to a usable or audible
frequency range when generating the sine wave that comes next in the form of the cycle~ object.

After the numeric values are converted into signals via sine wave generations, these newly create
signals are then multiplied with each other to create a single, mixed wave signal.

Before this signal is then outputted as audio, there is another subroutine running alongside this main
process. I implemented a function graph that would allow the user to plot points on the graph to
determine the amplitude of the audio output. I found this by experimenting with many different
objects and examining their help files, but many other objects I could not figure out how to
effectively use, and even when I did they did not always better the end result.

A bang signal is emitted every 0.35 seconds to match the intervals for the data values. This can of
course be changed to create more complex and abstract sounds, but for purposes of demonstration
and recording, I decided to keep the intervals equal. The plots I chose for the function graph were to
resemble a sort of blunt keystroke sound. So the points are plotted in a way that there is a sudden
plunge in amplitude and no recovery to create a dull, short sound to represent the data. The values
created from the function graph are then processed through a line~ object which transforms these
numeric values into a signal or sound. I attached a number object/ signal monitor to the line~ object
just to let the user know what sort of values are being created from this function graph and line~
object relationship.

This signal is then sent to the initial process and is multiplied with the result of the multiplication
between the two transformed data input signals, which in effect adds on the manipulations from the
function graph into the main signal. This final output is then attached to slider to control volume
and finally and ezdac~ object to control playback. I also implemented a sound record system that I
discovered from the previous year's submissions. This is a simple system that allows users to record
their experiments they have created from my patch.

I also added in a Spectrogram to further demonstrate the dual pitches that can be heard when the
patch is run, and to clarify how the sine waves formed in the main process interact with each other.
The results of my patch are encouraging. The data sets are clearly discernible and the sonification
paints quite a clear story to how the data relates to each other. Having the signal manipulations throughout the patch creates a hollow yet full sound each time a row of data is read. Hollow in the sense that the amplitude drops off so sharply and quickly, but full in the sense that dual pitches are heard in each note, representing the two data attributes.

I think the patch is a success in the fact that it is clear and anyone can learn how to use it without much difficulty. The patch is designed in a way that allows for heavy customisation and customised input in the form of changing variables, manipulating different objects and the data, as well as using completely different data sets.

The resulting audio is clear and offers a very abstract sonification into the dataset I nominated for this submission, and it also offers a platform to retain that quality no matter what the data set.