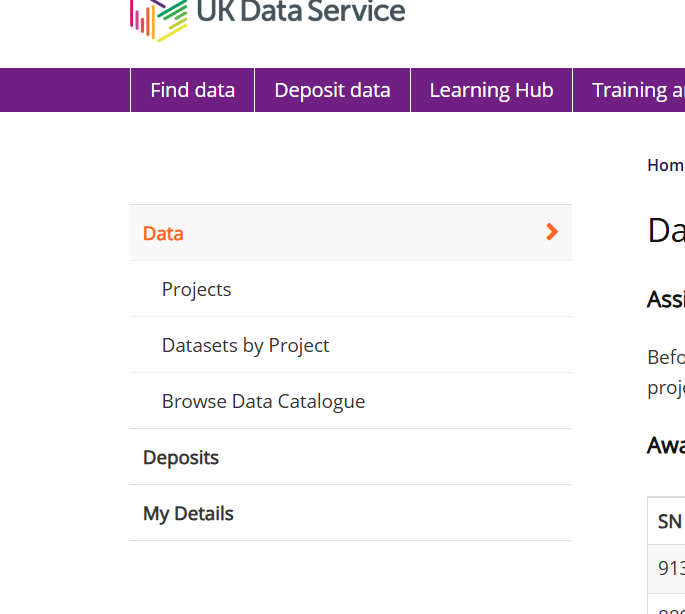
# Step 1: Necessary Downloads

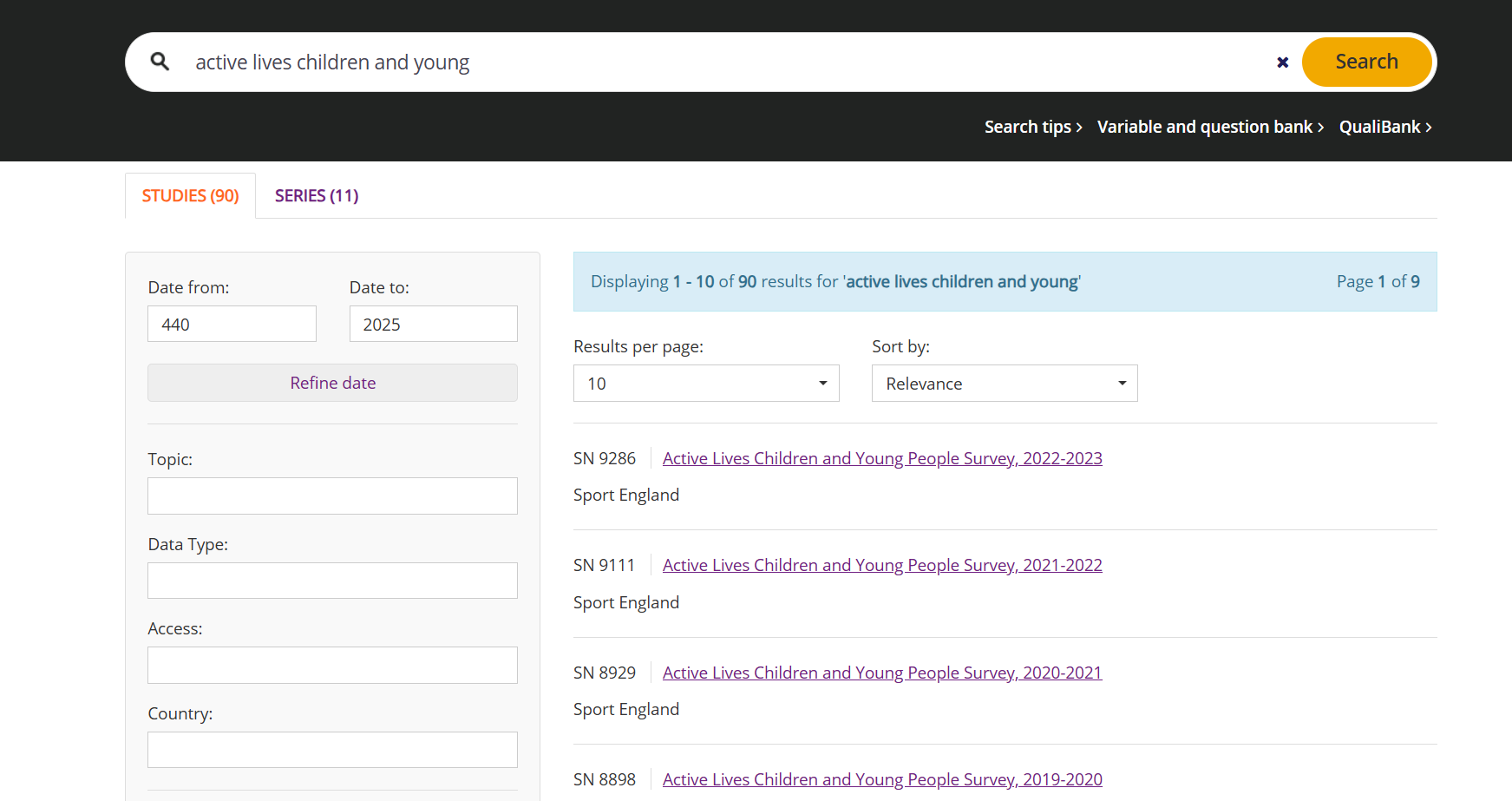
## UK Data Service Accounts

Create an account on UK Data Service, then create a new project to assign the dataset. Any dataset that we browse on UK Data Service will need to be assigned to a project. Hence this step. Once done, select the browse data catalogue option and search for “ Active Lives Survey”.

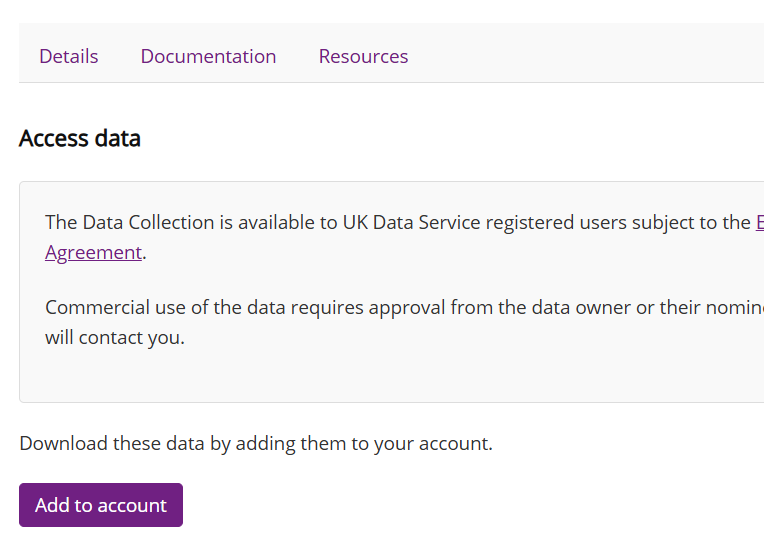


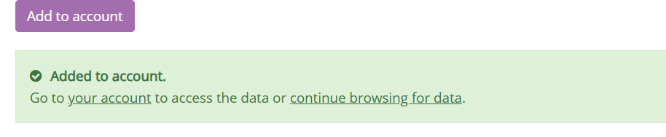
You will be provided with various datasets.

 Select the one that you need, in the next screen click on Access Data (shown below)

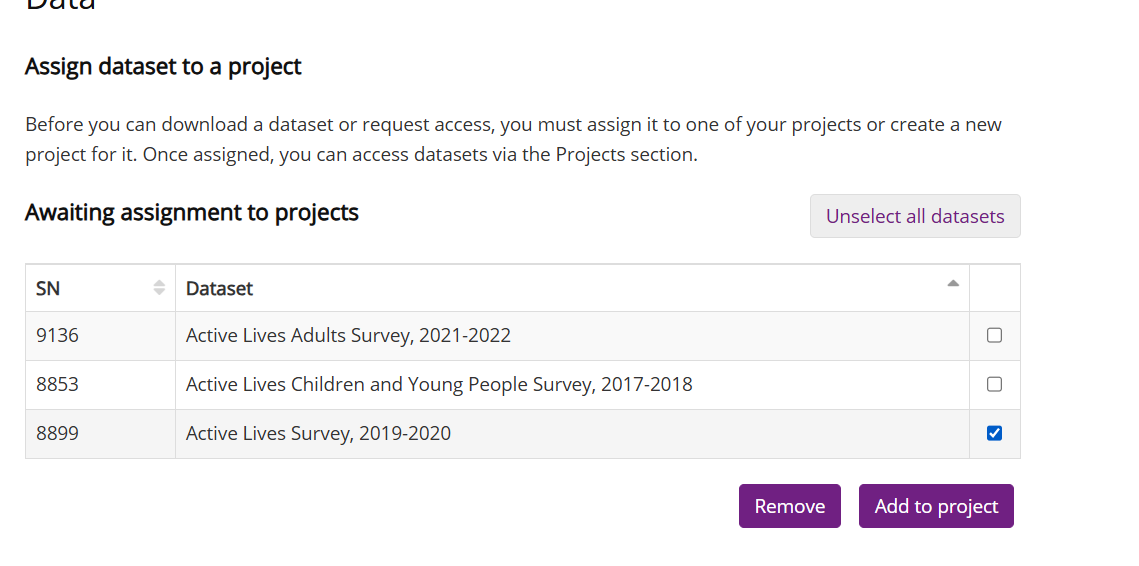
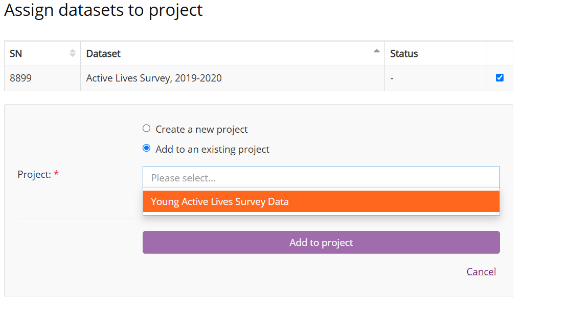


Then, click Add to Account

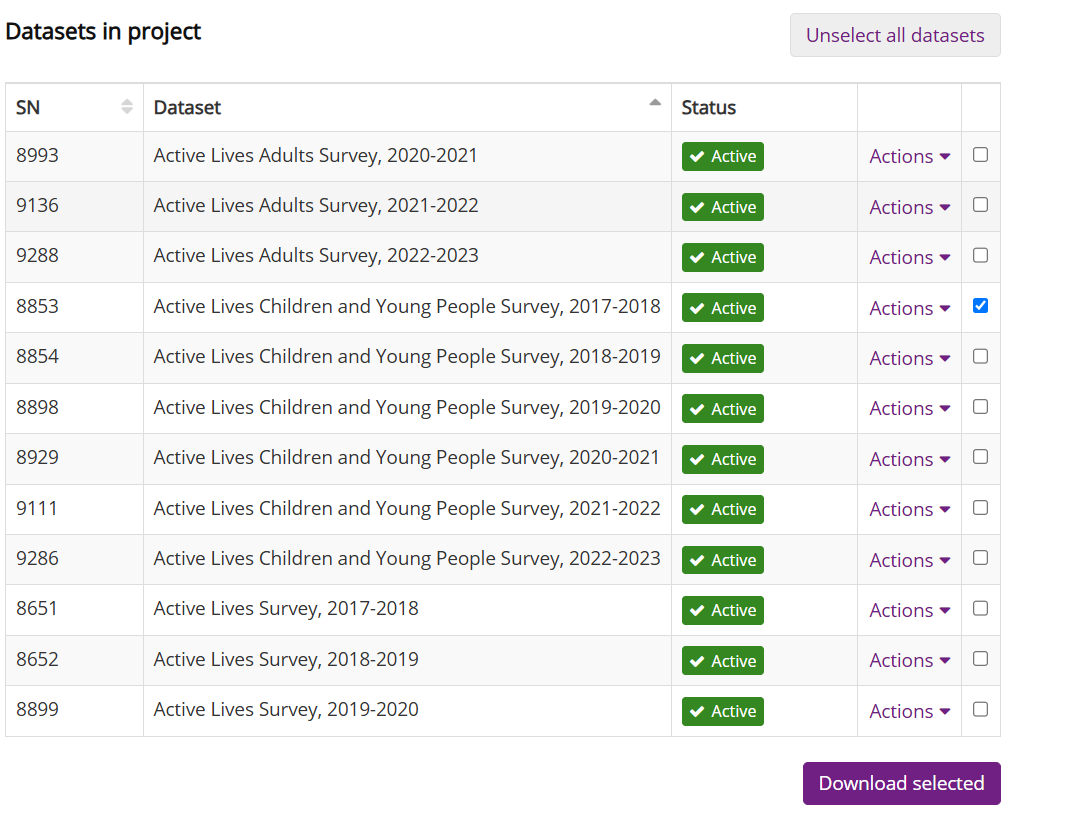
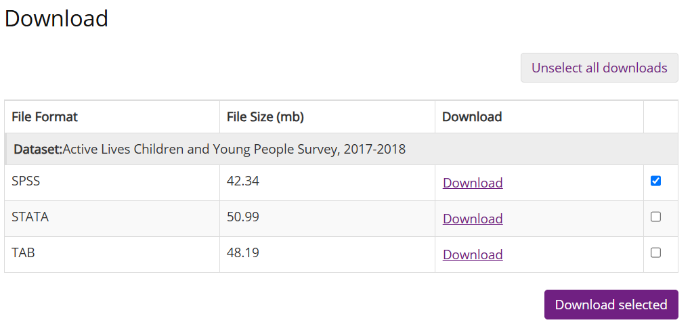




The go to your account



Select the datasets and then click on “Add to Project” (as shown above) and Select the project name.



Select the dataset and then go on to select the type, and press on download selected (as shown above)

Once, you have downloaded the zip file, go to the file location and then unzip the file, keep clicking the relevant files till you reach the .sav file.

## Download Dynamic Code

In the ALS report, a link to a Github account of Atharva Atul Joshi is provided. Upon selecting it, you will see “.ipynb” file (which we will be referring to as notebook from here on). Download that file and save it in your computer.

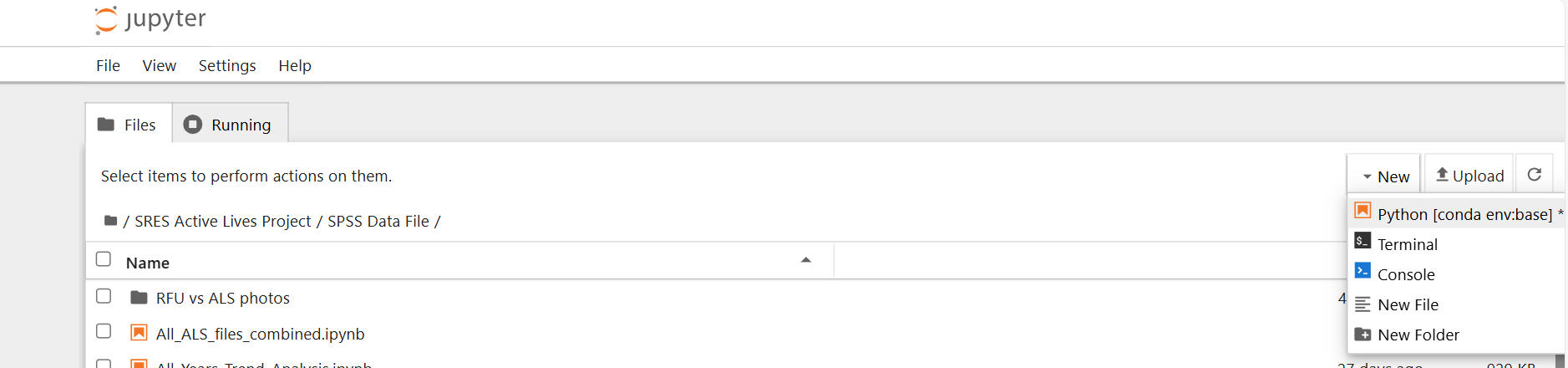
## Platform to execute the dynamic code:

**Step 1: Download and Install Anaconda**

1. Visit the official Anaconda website:  
   https://www.anaconda.com/products/distribution
2. Choose the correct installer for your operating system (Windows, macOS, Linux).
3. Download the Python 3.x version (preferably the latest stable version).
4. Run the installer and follow the instructions:
   * Choose “Add Anaconda to my PATH environment variable” if prompted (optional but helpful).
   * Proceed with the default settings unless you have a specific preference.

**Step 2: Launch Jupyter Notebook**

1. Open the **Anaconda Navigator** from your Start Menu (Windows) or Applications folder (Mac).
2. In Navigator, click **Launch** under the **Jupyter Notebook** tile.
3. A browser window will open showing the Jupyter Notebook interface.
4. You can now create a new Python notebook from the **New > Python 3** dropdown.



# 

# Step 2: Executing the Dynamic Code

While instructions and comments on how to execute the code are given in the notebook, I am writing down some fundamentals that any ALS/ RFU researchers need to keep in mind.

1. The notebook consists of various functions. So, the part where you will see the words “def” followed by some phrase, is called as creation of that function. All the functions are already created in the notebook and you as a researcher will just be implementing the functions.
2. Every function has some parameters that are defined while creating the function. These parameters are the inputs that are needed for that function to work. While implementing any function, once you call the function (write the name of the function, like, *function\_1(),* we need to pass these parameters inside the round brackets, ***function\_1(input\_parameter\_1= , input\_parameter\_2= ).***
3. A trick to see what input parameters are there inside a function, *call the function, place the cursor inside the curly bracket and press “****shift+tab****”*
4. The code is executed cell by cell, you can place your cursor on the cell and press “shift+enter”.
5. An addition done for better execution, is extracting the Active Lives Data in batches of 3. We have done this for the first 6 functions. You can use hash and unhash technique to avoid confusion. If you are dealing with first 3 years, select the cell containing the next 3 years (ctrl+A) and then (ctrl+?). This will comment out that cell. When you will be using that cell, ctrl+A and ctrl+?.

# STEP 3: User Inputs

There are some codes where the researchers will have to do some selections and assign a variable to those selections. It is important to do this as these variables are also used as input parameters to some functions. The places where they will have to do this is mentioned in the code book/ this file.

# Function 1:

The load\_file function is a versatile utility that loads data from various file types, including SPSS (.sav), CSV (.csv), and Excel (.xls, .xlsx). It checks if the specified file exists and then reads the file based on its extension. For Excel files, it supports loading a specific sheet by name, multiple sheets by their indices, or all sheets as a dictionary. Starting from 0, sheet indices can be specified in form of a list (inside square brackets) when implementing the function.

Just replace the path of the location where you saved the dataset inside the inverted commas.

# Function 2: Finding Columns Using Regular Expressions (RegEx)

Once we have loaded the dataset, we need to understand that ALS file has plenty of variables. So we need to narrow it down according to the sport and some generic variables. That is why we will RegEx. Regular expression has been used to find relevant variables defined by the user. We need to give a pattern for Regex, which is show below in the first 4 lines. Next we will call the next function ***find\_columns****()* and store it in respective variables.



When you print the above variables, you will get the respective variables in front of you to select from.

# User Input:

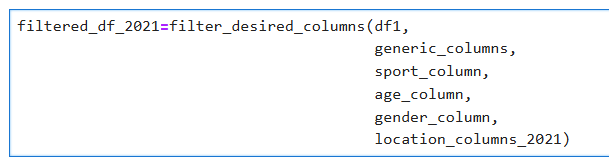
There will be some generic variables that are present in the dataset and mostly they are in the beginning columns itself. Add any column that you as a researcher deem fit. Select the one you find the best and replace that variable name in the quotes below and name it accordingly as follows



# Function 3: filter\_desired\_column()

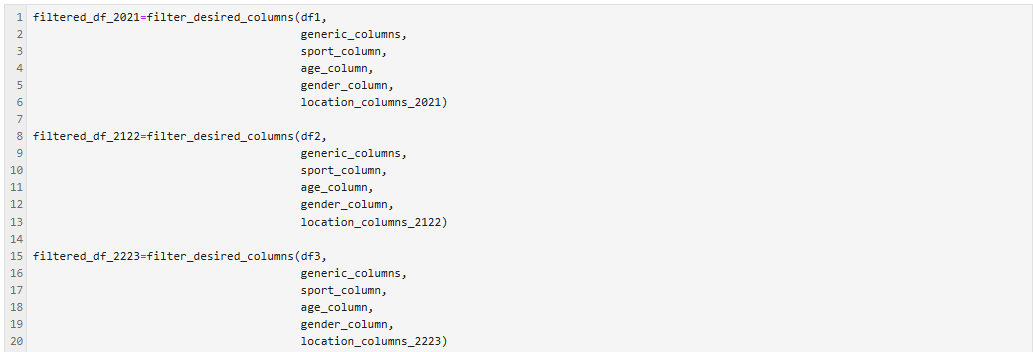
All the previously defined columns will get filtered using this function. The function is called and implemented as follows:

Desired dataset



User defined variables

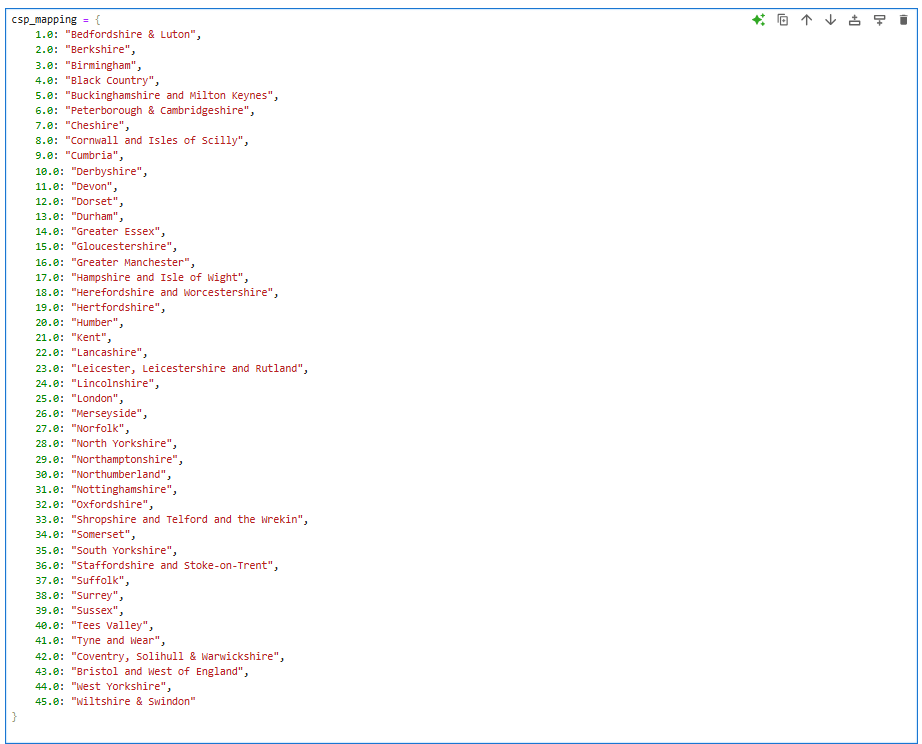
Point to note: We will be working in batches of 3 datasets till Function 7. Hence, the cell block will look like this:



# Function 4: map\_numerical\_data\_to\_labels()

Function 4: Map numerical labels to actual names. Usually in datasets, some categorical variables will be represented using numbers. We need to convert it to category usingmapping function of Python.

Step 1: Annotate the numerical value (which is there in the dataset) as key and,the actual name (which you can find in data dictionaries) as a value in dictionary format (key:value), as below



Step 2: Replace the names of variables in the below function

New column name

# 

Value: Label format

(can be done via CHATgpt as well)

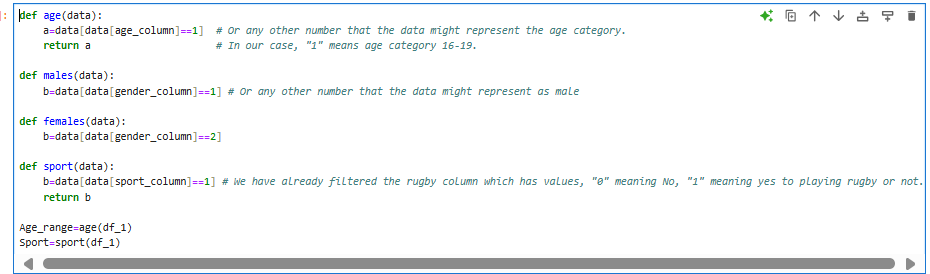
Existing column name

# Seperating the age column (Optional Step)

The variable in ALS datasets that represents age is ***age\_11***, which has values from age 5 to 16. So this code cell just converts these values from one row to each age group having separate columns. This step is done to just align the dataset like that of RFU.

# Function 5: Preprocessing Functions

These are the common functions for filtering values in a column. You can ignore if it doesn't apply to you or your dataset.



* For ALS for Children and Young People, only use the ***Sport*** Function.
* If you are dealing with ALS for Adults, you will need to use the ***age*** function.
* ***Males*** and ***females*** functions are created if you want to use or just segregate the males and females in this stage itself. The study does the segregation a little while later when all the datasets have been extracted, cleaned, and have been adjusted using the weight.

# Function 6: convert to desired format

If you have many datasets for different years you will need to save your filtered datasets properly, which will be case for ALS datasets. We will use the first function again to extract the desired files for further analysis and preprocessing. If it doesn't apply to you, you can skip to the next step.



The acceptable file formats are csv, xlsx, as there is a seamless experience while working with these two file formats in python with pandas (data science library).

***Now the above 6 functions will be used for all the ALS files that we want to analyse and by the virtue of function 6, it gets stored into the desired format. The next functions will be analysing all the filtered and stored ALS files. If you have an RFU dataset, start from the next function.***

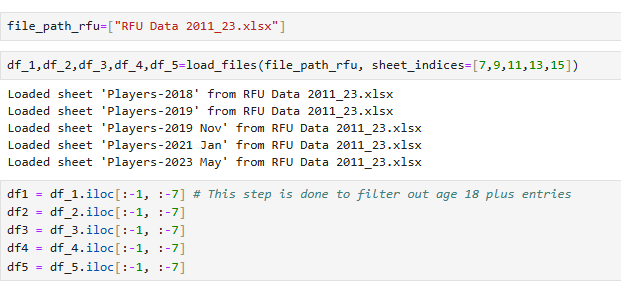
# Function 7 = Function 1.

As we have stored the datasets, we will use the function 1 to import those into our code. The words in red are the files names (file paths) which will be changed by you as a user.

“#” ensures that line is not included while the code is been run. You can unhash the next lines according to the number of files you will be importing.

## Extracting RFU

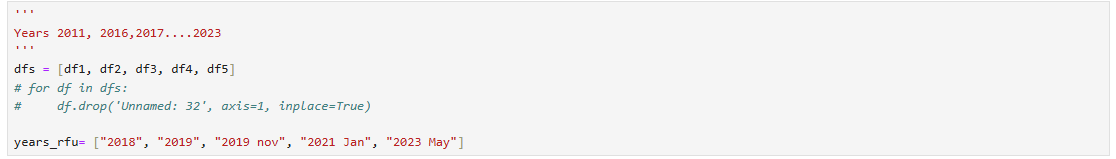
Name of the file



Starting from zero, write the indices that you want to extract if there are sheets present.

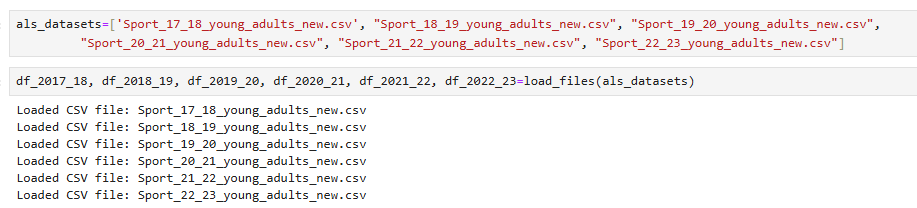
-1 means, all the rows except last 1 row and -7 means all the columns except last 7

You will also need to add the new variable i.e., df6 and then add the year according to the name of the sheet.



## Extracting ALS

This is similarly done has the first function, except that we will use the function ***“load\_files”***



Calling (implementing) the function.

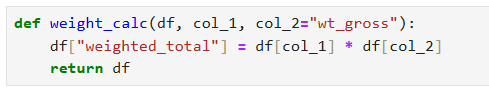
Storing all the filtered datasets in one list***(als\_datasets***). If you later get a new filtered ALS dataset, just put a comma and add it in the ***als\_datasets.***

Make sure you name you datasets correctly according to the year or whatever nomenclature you want to follow.

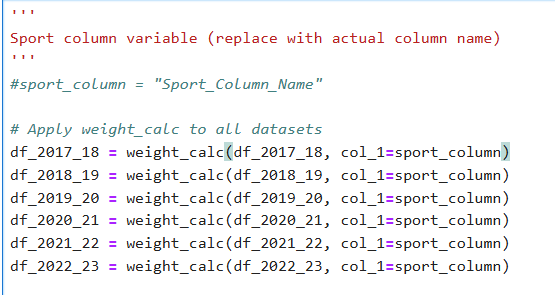
Point to note: If you have already saved your filtered ALS files and want to work on it, you can do that, just make sure some of the variables (sport\_column, age\_column, etc.) that we define in the earlier stages get defined here as well. For ease of code execution, I will be attaching a cell block here as well.

# 

# Function 8: weight\_calc() [Only for ALS Data],[Skip for RFU]



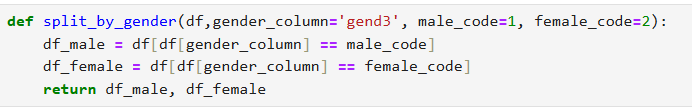
The whole ALS survey is based on the sample and hence we multiply the respective respondents with the weight column given in the ALS data.



You need to define the sport column beforehand. Unhash it and define.

# Function 9: split\_by\_gender()

Function to create male & female subsets. Just know you have to give the male code, female code according to your dataset. Every years’ dataset has combination of males and females, that’s why we call the function in this way. The”gend” column depicts males as 1.0 and females as 2.0, the function accordingly uses the same for splitting according to the gender.

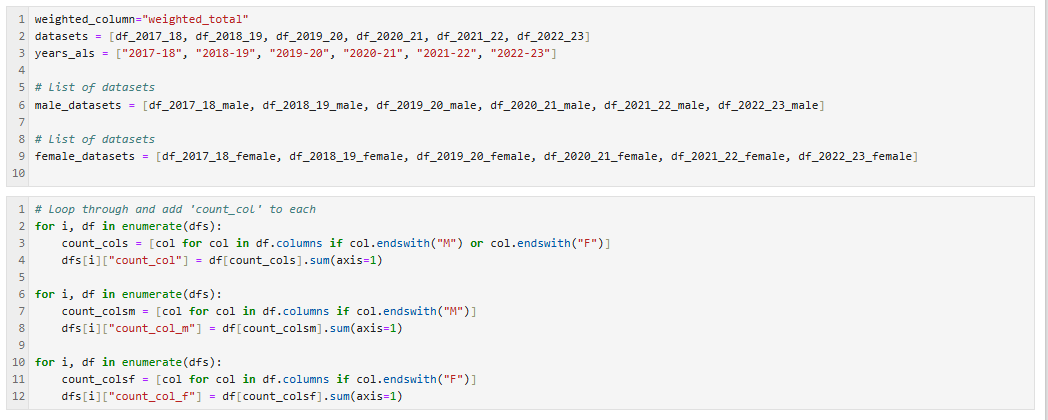


# 

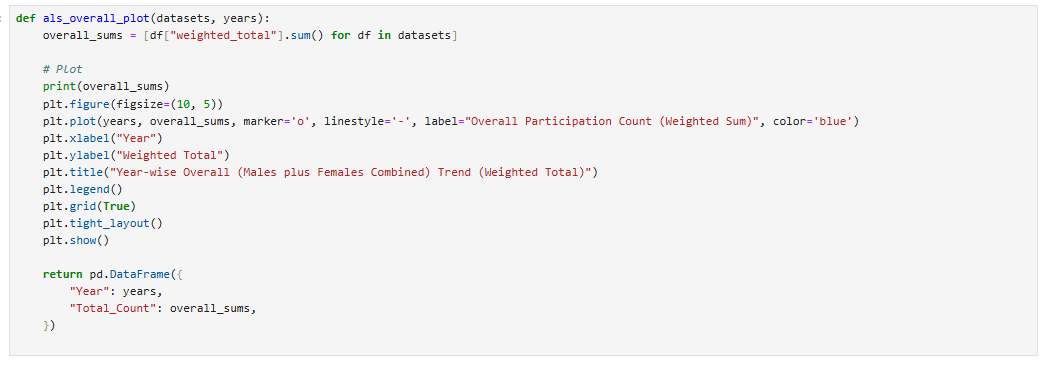
Original datasets

Ensure you have two variables this time for male and female

# Labelling and cleaning ALS and RFU data

* Block 1 is for ALS data, it stores all the overall combined datasets and male, female datasets into lists
* Block 2 stores overall, male and female counts which will be used later on.

# Function 10: Plot year-on-year trend for ALS data





# Function 11: als\_genderwise\_plot()

Same function as above but separated for males and females.





# Function 12: rfu\_overall\_plot()

Same as ALS, but this function takes into account the RFU data. Gives the trend for overall participation till the age of 17.





# Function 13: rfu\_gender plots

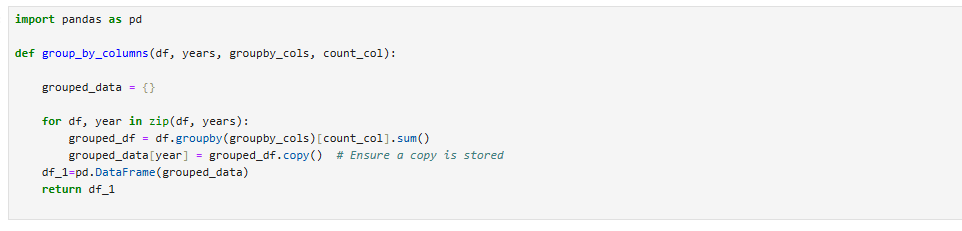
Same as ALS, but this function takes into account the RFU data for genderwise y-o-y trend

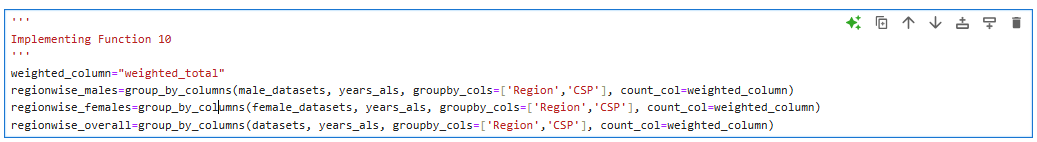




# Function 14: group\_by\_columns()

A very important function that helps the user group across categories like age, region





This should be a numerical/categorical value depending on the aggregating function that we use.

This should be a category

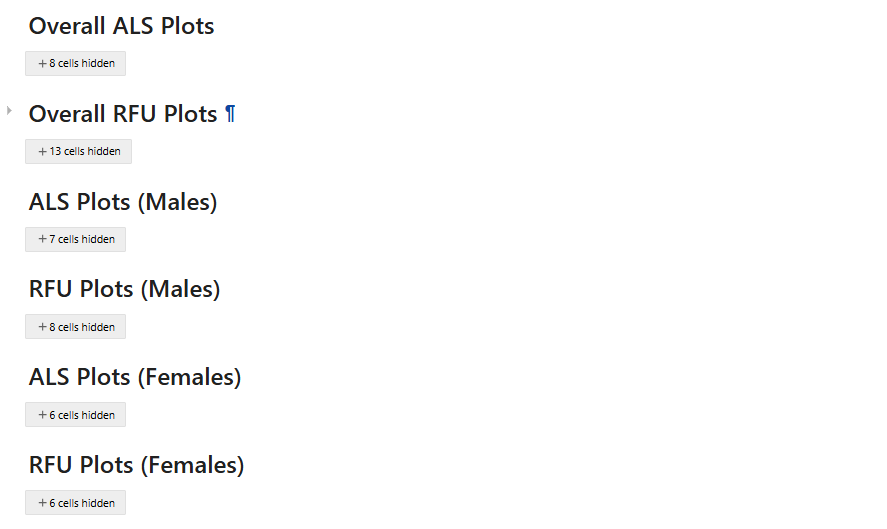
3 datasets (overall, male and female)

# Function 15: Plot Category trends

After applying the above function, we have essentially grouped the the data according to the region or the age and now we will be looking at some insightful trends with the help of this function. These insights can be the largest counties playing Rugby or the counties with best percentage growth across the years and vice versa.



This function will be used for all the sections that are shown in the picture below



This is the way to call the function:

****

The first and last columns for ALS and RFU data will be different. Mention accordingly

Largest, Smallest, Percent growth, etc

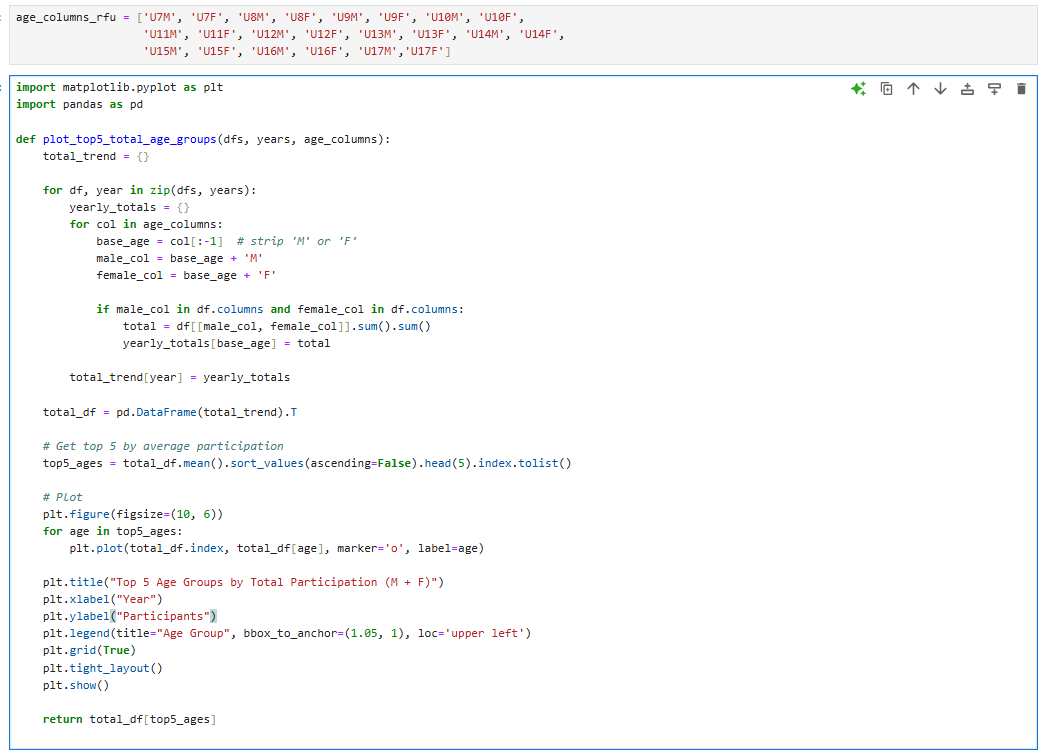
The region-wise grouped column.

# Function 16,17,18: For agewise trends in ALS and RFU datasets

The next two sections shown below will be used to plot agewise analysis. The same group by and plot participation trend function.

* For ALS, function 15 can be used, by changing the the group-by column to age and then using the above function that we used for plotting.

## Function to group-by age for RFU data

1. We will need to give the age\_columns that are mentioned in the RFU.
2. The code separates male and female columns and adds the total number for a combined column analysis
3. The next two functions below, will be used to show the age-wise trend according to the gender.

