# Homework assignment week 4

# GitHub Instructions

1. Use the same repository as in week 2 (ds5\_assignment\_ group#), make sure all the group members and teachers are still collaborators.
2. Create one main.ipynb file that contains the answers to all the exercises below! However, you can call other files in your main file.
3. Make sure that you push and commit all your changes to the file before the deadline. If changes are made after the deadline, it will not be taken into account for the bonus point.

# Exercise 1: Hotel Bookings

You have been given a file about hotel bookings, you can find these data in the file *‘hotelBookings.xlsx’*. These data contain the number of bookings of the hotel ‘Resort Hotel’, which is spread over different countries. This file contains a lot of errors, and needs to be cleaned. For this exercise, you need to read the Excel file, analyse the data, find as many mistakes as possible and correct them. In particular,

1. Read the excel file using the pandas library
2. Inspect the data and find as many mistakes you can.
3. Clean the data, which decisions did you make to solve the errors?

Create one main function (which must depend on other functions) that performs all the data cleaning. Like last week:

1. Work in Github from the start
2. Break down the code into smaller functions. Give each person a task. In the main.ipynb file, clearly describe the tasks and who is responsible for that task.
3. Add descriptive docstrings to each function and type hints for your functions. Add the name of the person who worked on the function in the docstrings.

# Exercise 2: Retail 2.0

You have been given the file ‘*detailedRetail.xlsx’*, which contains the sales data for a company including the responsible sales manager. For this exercise, you need to read the Excel file, analyse the data and generate a sales report with key metrics. In particular, perform the following steps:

1. Read the excel file using the pandas library
2. Perform data analysis on the sales data to calculate the following metrics:

* Total sales for each category
* Percentage that a category contributes to the total sales
* Sales for each month
* Percentage that the month contributes to the total
* Sales for each sales manager
* Percentage that the sales manager contributes to the total

1. Generate a report (dataframe) that displays the metrics calculated in step 2.
2. Store the report into an excel file called ‘reportRetail.xlsx’
3. After applying these techniques, structure your code by adding docstrings and breaking down your code.

Again, create one main function (which must depend on other functions) that performs all the data cleaning and of course:

1. Work in Github from the start
2. Break down the code into smaller functions. Give each person a task. In the main.ipynb file, clearly describe the tasks and who is responsible for that task.
3. Add descriptive docstrings to each function and type hints for your functions. Add the name of the person who worked on the function in the docstrings.

# Exercise 3: Post-processing of tweets

Everyone knows the social media platform Twitter (it now has a different name, but who cares), a place where people can post Tweets about their life, politics, interests and so on. As Twitter is very popular these days, the company wants to analyse the sentiment of all the tweets. To be specific, they want to know whether it is positive, negative or neutral. A part of the tweets can be found in ‘*tweets.xlsx’*. However, not all these tweets are in English, which means that we have to post-process this file to make it useable for Twitter.

# Exercise 3.1: Language detection

To find out the sentiment of a tweet, we first have to find out in which language this tweet is written. In this exercise, detect the language of a tweet, and store it in the column ‘language’. There are already multiple packages available that you can use. One of the most famous packages is the **langdetect** package in Python. An example of how your output can look like is as follows:

|  |  |
| --- | --- |
| Tweet | Language |
| Hello Wolrd | en |
| Bonjour! | fr |
| Hola! | es |
| Ciao! | it |
| привет! | ru |

Handle any exceptions by assigning ‘Unknown’ to the language column and install any missing libraries using pip (or conda) install. Add a requirements.txt file in your folder to let the user know which (unusual) libraries you have used.

Again, create one main function (which must depend on other functions) that performs all the data cleaning and of course:

1. Work in Github from the start
2. Break down the code into smaller functions. Give each person a task. In the main.ipynb file, clearly describe the tasks and who is responsible for that task.
3. Add descriptive docstrings to each function and type hints for your functions. Add the name of the person who worked on the function in the docstrings.

# Exercise 3.2: Sentiment detection

In the previous exercise, we have established the language of the tweet. In this exercise, you will be asked to define the sentiment of a tweet based on the language. As this is not a straightforward approach, we have defined a few steps that should be taken:

1. Create a function analyze\_sentiment\_english(tweet) that takes a tweet as input and performs sentiment analysis using the TextBlob library. The function should return 'positive' if the polarity score is greater than 0, 'negative' if the polarity score is less than 0, and 'neutral' otherwise. Hint: Use the TextBlob library to create a TextBlob object from the tweet text and access its sentiment.polarity property.
2. Define a function analyze\_sentiment\_other(tweet) that takes a tweet as input and performs sentiment analysis using the SentimentIntensityAnalyzer from nltk. The function should return 'positive' if the compound score is greater than or equal to 0.05, 'negative' if the compound score is less than or equal to -0.05, and 'neutral' otherwise. Hint: Use the SentimentIntensityAnalyzer from nltk.sentiment to create a SentimentIntensityAnalyzer object and calculate the sentiment scores using its polarity\_scores() method.
3. Apply language-specific sentiment analysis to each tweet in the DataFrame. Iterate over each tweet and use the appropriate sentiment analysis function based on the identified language. Store the sentiment result in a new column 'sentiment' in the DataFrame.

An example of how your output can look like is as follows:

|  |  |  |
| --- | --- | --- |
| Tweet | Language | Sentiment |
| Hello Wolrd | en | positive |
| Bonjour! | fr | positive |
| Hola! | es | positive |
| Ciao! | it | positive |
| привет! | ru | positive |

Again, create one main function (which must depend on other functions) that performs all the data cleaning and of course:

1. Work in Github from the start
2. Give each person a task. In the main.ipynb file, clearly describe the tasks and who is responsible for that task.
3. Add descriptive docstrings to each function and type hints for your functions. Add the name of the person who worked on the function in the docstrings.