# Data Tech Test Solution

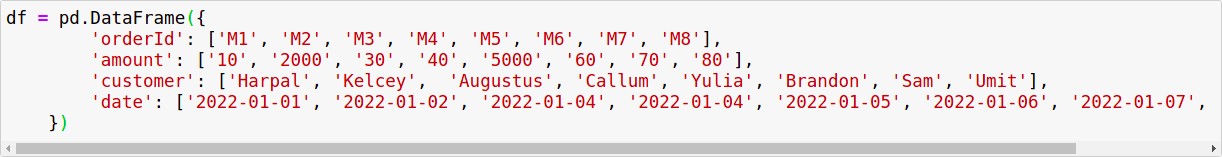
## Task I

As part of this task, we have to populate the pandas table with the given column name and values.

In order to accomplish this task, we will rely on the fact that with pandas, we can work using vectorized notations.

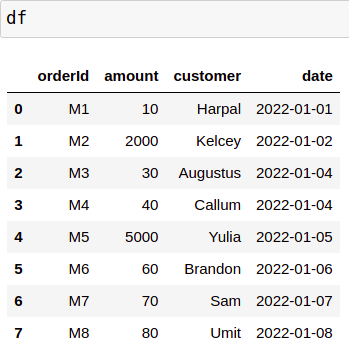
Hence, I have used the column name as the **key** for pandas and the value provided (either in the form of a string or in the form of a python list) will be set as the value for the given key.

Please refer to the screenshots below for a **test data** that I have populated in pandas (the test data is only shown here for demonstration purposes) -



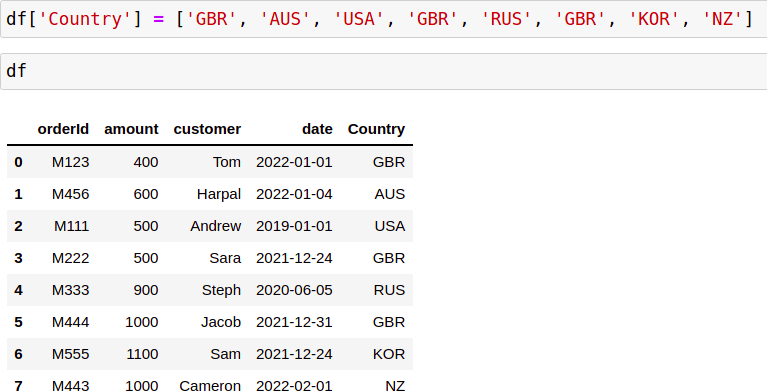
**Figure 1:** Test dataframe created in Pandas

The dataframe that got created using the test data above can be found below -



**Figure 2:** Dataframe created using test data

Let’s consider the column, that we need to add, by the name “*Country*” and the value that needs to be set is ['GBR', 'AUS', 'USA', 'GBR', 'RUS', 'GBR', 'KOR', 'NZ']. We can successfully set the value as follows (the output in the second cell verifies that we’ve successfully populated the pandas dataframe)



**Figure 3:** Updated pandas dataframe with country values for a newly added column (*Country*)

As can be seen above, the data frame is enriched with the county names as expected.

The same logic is applied in the *transformer.py* to enrich the data as required for Task I.

## Task II

In this task, we can again use the vectorized notation that pandas supports in order to obtain the mask for the customers who spend less than and greater than a given threshold amount.

The conditions can be written as -



**Figure 4:** Condition to obtain rows where the customers spent less than a threshold amount



**Figure 5:** Condition to obtain rows where the customers spent more than a threshold amount

Note that we’ve made an assumption that the **amount** which is equal to

*threshold* is considered in the lesser spend

Once we’ve obtained the mask, we can then extract the customers based on this mask and can return both the high spending customers and low spending customers in the form of a tuple.

## Additional

Usage of seaborn package to plot graphs

In order to determine what would be an appropriate threshold to split the customers, we can plot the amount spent by each of the customers with a line plot. Based on that, we can then decide that a threshold amount (like 700 in the example below for orders.csv) would be appropriate. This will split the customers in half based on their amount spent. The graph can be found below -



**Figure 6:** Plotting the amount spent by each customer (and the split threshold) using seaborn package.

# Bonus Tasks

**Note:** All the bonus tasks are implemented in the class by the name “Bonus” in

*transformer.py*

***Task 1:*** Which customer placed the highest order amount?

In order to find the name of the customer who placed the highest order amount, we need to do the following steps

1. Extract the column corresponding to amount from the orders dataframe
2. Convert the amounts into integer as we need to perform integer comparison
3. Find the index with the maximum value using “*argmax*”
4. Find the row corresponding to the index obtained in step 3
5. Return the name of the customer from the row obtained in step 4

The implementation for this bonus task can be found by the name:

### find\_customer\_with\_highest\_order\_amount

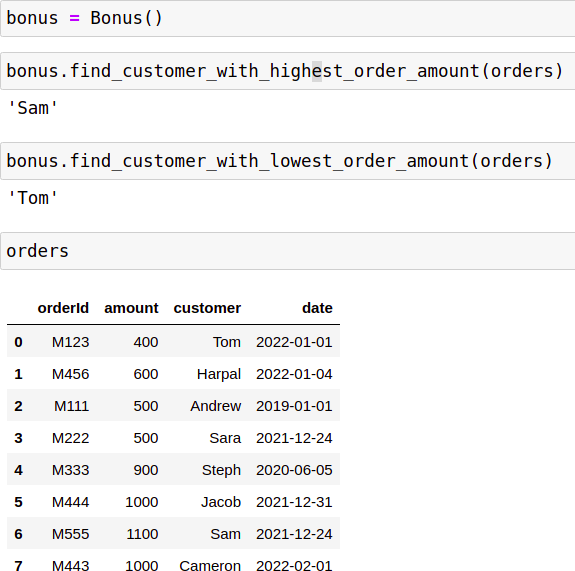
***Task 2***: Which customer placed the lowest order amount?

The steps for this task are very similar to Task 1, however instead of obtaining the index corresponding to maximum value, we’ll obtain the index for the minimum value.

The implementation for this bonus task can be found by the name:

### find\_customer\_with\_lowest\_order\_amount

The figure below shows the output obtained for orders.csv for find customer with highest and lowest order amount



**Figure 7:** Output obtained for customer with highest and lowest order amount (can be verified manually as well from the dataframe values)

***Task 3:*** What was the average order amount across all customers?

In order to calculate the average order amount across all the customers, we can extract the column corresponding to the amount from the dataframe and then use the “mean” method that is available for pandas dataframe.

The implementation for this bonus task can be found by the name:

### find\_average\_order\_amount

The output obtained after doing so is as shown below -



**Figure 8:** Output obtained for average amount across all customers

***Task 4:*** Which customer placed the earliest order?

In order to calculate this, we can do the following steps

1. Extract the column corresponding to dates from the orders dataframe
2. Convert the date to datetime using pd.to\_datetime() method
3. Extract the index corresponding to earliest date (or minimum date) using argmin
4. Obtain the customer corresponding to the index obtained in the above step

The implementation for this bonus task can be found by the name:

### find\_customer\_with\_earliest\_order

The output obtained after calling this method is as shown below -



**Figure 9:** Output obtained for the customer with earliest order

***Task 5:*** In which month did most of the orders happen (the year can be ignored)?

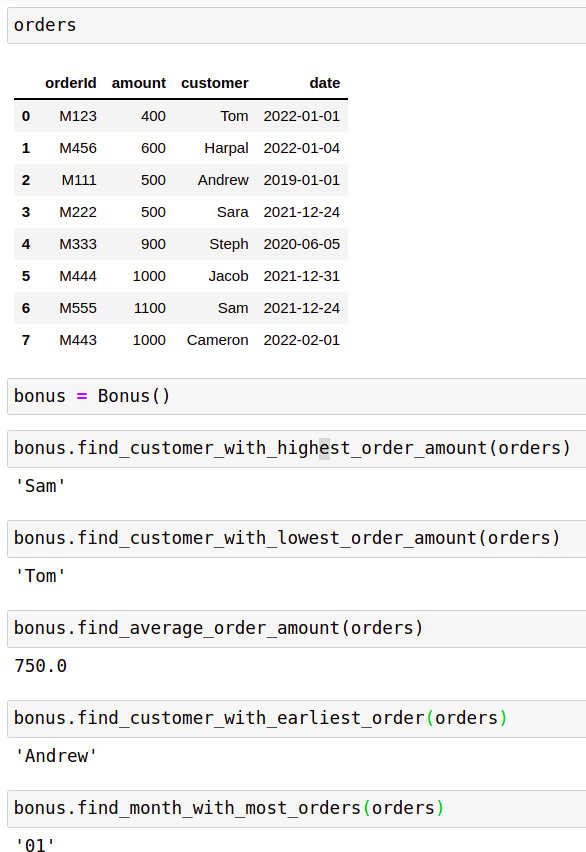
In order to accomplish this task, we first obtain all the dates from the date column. Note that each of the dates is present in the format: ‘YYYY-MM-DD’. Hence we can split the date using the delimiter ‘-’ and extract the value corresponding to the field with index = 1.

Once we’ve extracted the month, we can then check which months occurs the most and this is the answer we are looking for.

The output obtained is as shown below -



**Figure 10:** Output obtained for finding the month with the most orders As can be seen, the month when most orders happened is ‘01’, i.e. January.

Screenshot of all the bonus tasks along with the orders dataframe taken under consideration -