Please complete this Task as soon as possible. The goal of this exercise is to test your understanding of data processing. Complete the task on section A if programming in C# and Section B if programming in Python. Complete as much as you can.

SECTION A - C# Programming

- You can use C# to complete this task in .NET Core.
- Your code should use SOLID principle, LINQ, lambda expressions.
- You can also load your data on table or memory, whichever is convenient.
- Write Unit Test to test all functionalities.
- Make use of state management.

Task 1: Extract rates from 1st October 2019 and 31st October 2019.

The goal of this task is to extract the rates for the Euro i.e. (EUR) base currency between 1st October 2019 and 31st October 2019 using an API, in order to complete this task, you would need to carry out the following.

1. Sing up for Fixer's API, which can be found in the following link. (https://fixer.io/signup/free)

2. After signing up, go to the following link (https://fixer.io/signup/free) to read more about the API.

3. The API that will be used in this task is http://data.fixer.io/api/. Please note you are limited to only 1000 API calls for the free license so make your request carefully. This is to test how you can avoid making unnecessary repetition of API calls

4. Extract the rates for all the dates from **2019-10-01** to **2019-10-31** where **'EUR'** is set as the base currency.

5. Store the following information in a table or memory.

- **date**
- **quote**
- **base**
- **rates**

Task 2: Load data files and merge into one data.

The goal of this task is to merge the three files in the **'data'** folder into one.

The loaded data has the following columns.

- **date** i.e. (The date on which the transaction took place).
- **country** i.e. (The country in which the transaction was made).
- **currency** i.e. (The currency in which transaction amount is denominated in).
- **amount** i.e. (The amount of the transaction).

Task 3: Merge the data with the rates dataset.

The goal of this task is to merge the two data acquired in Task-1 and Task-2 and store the result in a new table or memory.

Think about what keys you would need to join the two data and how you will join them. i.e. (**inner join**, **left join**, **right join** or an **outer join**). You are free to use LINQ expression for the joins.

Task 4: Add a new column or property called amount_eur, with the values in the column amount converted to Euros.

The Goal of this task is to create a new property or column called **'amount_eur'** in the data derived from **Task-3**.

amount_eur can be derived by **dividing** the **'amount'** by **'rates'**.

Task 5: Group Countries into the specified Categories.

The goal of this task is to put the countries from Task-4 in appropriate groups with the title **country_group**.

The mapping of each country to a country group is as follows.

1. **Austria**, **Italy**, **Belgium** and **Latvia** are mapped to the country group '**EU**'
2. **Chile**, **Qatar**, **United Arab Emirates** and **United States of America** are mapped to the country group '**ROW**'
3. **United Kingdom**, **Australia** and **South Africa** should be mapped to themselves i.e. (United Kingdom should be mapped to United Kingdom)

Task 6: Calculate the total amount in Euro for each country group.

The goal of this task is to calculate the total amount in 'EUR' i.e. (using the column **amount_eur**) for each of the country group classified in the previous task and the result should be ordered in descending order of total amount in Euros.

Task 7: Display on web view as a formatted HTML table.

- Output from Task 6.
- Output from Task 5.
    - All the countries relating to the category to display in a sub table when the user clicks on a category on the main table.

SECTION B - Python Programming

Task 1: Extract rates from 1st October 2019 and 31st October 2019.

The goal of this task is to extract the rates for the Euro i.e. (EUR) base currency between 1st October 2019 and 31st October 2019 using an API, in order to complete this task, you would need to carry out the following.

1. Sing up for Fixer's API, which can be found in the following link. (https://fixer.io/signup/free)

2. After signing up, go to the following link (https://fixer.io/signup/free) to read more about the API.

3. The API that will be used in this task is http://data.fixer.io/api/. Please note you are limited to only 1000 API calls for the free license so make your request carefully. This is to test how you can avoid making unnecessary repetition of API calls

4. Extract the rates for all the dates from **2019-10-01** to **2019-10-31** where **'EUR'** is set as the base currency.

5. Store the following information in a Table or if you are using Python, then store it in a Pandas DataFrame.

- **date**
- **quote**
- **base**
- **rates**

Task 2: Load data into one table.

The goal of this task is to load the three files in the **'data'** folder into a single table or a Pandas DataFrame, if you are using python.

The loaded data has the following columns.

- **date** i.e. (The date on which the transaction took place).
- **country** i.e. (The country in which the transaction was made).
- **currency** i.e. (The currency in which transaction amount is denominated in).
- **amount** i.e. (The amount of the transaction).

Task 3: Merge the data with the rates table.

The goal of this task is to merge the two tables acquired in Task-1 and in Task-2 and store the result in a new table/ Pandas DataFrame.

Think about what columns you would need to join the two tables on and how you would join them. i.e. (**inner join**, **left join**, **right join** or an **outer join**).

Task 4: Create a new column called amount_eur, with the values in the column amount converted to Euros.

The Goal of this task is to create a new column called **'amount_eur'** in the table derived from **Task-3**.

This column can be derived by **dividing** the values in the column **'amount'**, by the values in the column **'rates'**.

Task 5: Group Countries into the specified Categories.

The Goal of this task is to create a new column called **country_group**, in the resulting table from Task-4, which contains the country group value each country belongs to.

The mapping of each country to a country group is as follows.

4. **Austria**, **Italy**, **Belgium** and **Latvia** are mapped to the country group '**EU**'
5. **Chile**, **Qatar**, **United Arab Emirates** and **United States of America** are mapped to the country group '**ROW**'
6. **United Kingdom**, **Australia** and **South Africa** should be mapped to themselves i.e. (United Kingdom should be mapped to United Kingdom)

Task 6: Calculate the total amount in Euro for each country group.

The goal of this task is to calculate the total amount in 'EUR' i.e. (using the column **amount_eur**) for each of the country group classified in the previous task and the result should be ordered in descending order of total amount in Euros.

Finally, you would need to write the results obtained to a CSV file as given in the file output.csv.