UNIVERSITÉ DE NEUCHÂTEL

EXERCISE 8

DATA MANAGEMENT

Group 11

Alexandre Ducommun
Rabeb Ben Ramdhane
Abdallah Shukor

Professor

Iulian Ciorascu

25 November 2019

Project Environment

We work on different python script.

Scripts:

- project.py (For testing and all steps)
- getregions.py (Q1 and Q2)
- download.py (Q3 and Q4)
- transformregions.py (Q5)
- loadregions.py (Q6)
- tools.py (Functions to retrieves file path, change date format, etc…)

Question 1

We used the code below to return à dictionary that contain all the regions with attributes country, region, city, the publication date of data and the URL to the source (…listings.csv). To retrieve the regions, we used web scraping. The modules used for this task are BeautifulSoup, requests and tools. During this task we change the date format to yyyy-mm-dd.

*We created **tools** to put functions that manage file path, date format or the currency.*

getregions.py

```
21  def get_all_regions(URL):
22
23      # Create the parse tree
24      soup = BeautifulSoup(requests.get(URL).content, 'html5lib')
25      table = soup.find('div', attrs = {'class':'contentContainer'})
26      regions = {}
27
28      # Put all the regions in the dictionary with the city as key
29      for row in table.findAll('h2'):
30          area = row.text.rsplit(',', 2)
31          city = area[0]
32          region = area[1]
33          country = area[2]
34
35          regions[city] = [country[1:], region[1:], city]
36
37      # Add metadata for each region of the dictionary
38      for t in table.findAll('table'):
39          date = t.findAll('tr')[4].findAll('td')[0].text
40          date = tools.format_date(date)
41          city = t.findAll('tr')[4].findAll('td')[1].text
42          link = t.findAll('tr')[4].findAll('td')[2].find('a').get('href')
43          metadata = [date, link]
44
45          if city in regions:
46              regions[city].extend(metadata)
47
48      return regions
```

Question 2

We didn't implement this part. The web page has a table for each region with the most recent files and if we need all the historical files, we should click on a link (eventLIstener) to have a complete table with all the historical files. So, during the web scraping we must manage this eventLIstener to extend the table for each region and then loop for each table row that has the bracket <a> with the file name 'listings.csv' to retrieve the href attribute.

Question 3 and 4

For this task we had to download the listings.csv file from a given region and to save it locally in a folder with a specific structure. With the previous task we got the regions with their latest version of the file listings.csv.

This function takes to arguments: given region (city) and force (Question 4). We used the given region as a key to retrieve the link from the dictionary containing all regions. With the data we construct the directory structure (country/region/city/date/listings.csv)

Before downloading the file, we check if the file path we build already exists. If not, the folder structure is built and the listings.csv file downloaded. In the case of the file already exists, we use the second argument of the function. If it is set to True, the file will be downloaded again. If it is set to false, nothing happened.

download.py

```
8  def download_region(city, force):
9      regions = getregions.get_all_regions("http://insideairbnb.com/get-the-data.html")
10     region  = regions[city]
11     link    = region[4]
12     dir     = tools.get_file_dir(region)
13     filename = 'listings.csv'
14     file_path = dir + filename
15
16     if not os.path.exists(file_path):
17         os.makedirs(dir)
18         wget.download(link, file_path)
19         print('file downloaded')
20     elif os.path.exists(dir):
21         if force == True:
22             wget.download(link, file_path)
23             print('File overrided')
24         else:
25             print('file already exist')
```

Question 5

For the given region we open the listings.csv file and add the columns we suppose to be useful to have and save the result in a new csv file.

The columns:

- **City**: The city of the given region
- **Region**: The region of the given region
- **Country**: The country of the given region
- **Currency**: We choose to put the currency of the country of the given region from another data source containing all the countries with their currency.
- **CSV_FILE_PATH**: The file path of the csv file
- **Download_date**: The date of the listings.csv
- **Link**: The link to download the original file

transformregions.py

```
def transform_region(city_key):
    regions = getregions.get_all_regions("http://insideairbnb.com/get-the-data.html")
    region = regions[city_key]
    file = tools.get_file_dir(region)
    trname = file + city_key + ' ' + region[3] + ' TR.csv'

    f = pd.read_csv(file + 'listings.csv')

    # add "CITY"
    f.insert(16, 'city', region[2])
    f.to_csv(trname, index=False)

    # add "REGION"
    f.insert(17, 'region', region[1])
    f.to_csv(trname, index=False)

    # add "COUNTRY"
    f.insert(18, 'country', region[0])
    f.to_csv(trname, index=False)

    # add "CURRENCY"
    f.insert(19, 'Currency', tools.get_currency(region[0]))
    f.to_csv(trname, index=False)

    # add "CSV_FILE_PATH"
    f.insert(20, 'csv_file_path', trname)
    f.to_csv(trname, index=False)

    # add "DOWNLOAD_DATE"
    f.insert(21, 'download_date', region[3])
    f.to_csv(trname, index=False)

    # add "LINK"
    f.insert(22, 'link', region[4])
    f.to_csv(trname, index=False)

    return trname
```

Question 6

Finally, we load the data from a given csv file to a database table. We choose SQLite to store the data. We begin to create and connect to the database then we execute the first sql statement to create the table LISTINGS where we will store our data.

Then we insert the data from the given csv file to the table LISTINGS. We commit the transaction and close the connection.

loadregions.py

```python
# CREATE or REPLACE table LISTINGS to store data from csv files
def csv_to_sqlite(csv_file_path):
    try:
        conn = sqlite3.connect('PROJECT-DM-G11.db')
        c = conn.cursor()

        # Create table LISTINGS if not exists
        c.execute(
            """
            CREATE TABLE IF NOT EXISTS LISTINGS (
                [id] INTEGER PRIMARY KEY,
                [country] TEXT,
                [region] TEXT,
                [city] TEXT,
                [name] TEXT,
                [host_id] INTEGER,
                [host_name] TEXT,
                [neighbourhood_group] TEXT,
                [neighbourhood] TEXT,
                [latitude] TEXT,
                [longitude] TEXT,
                [room_type] TEXT,
                [price] INTEGER,
                [currency] TEXT,
                [minimum_nights] INTEGER,
                [number_of_reviews] INTEGER,
                [last_review] DATE,
                [reviews_per_month] FLOAT,
                [calculated_host_listings_count] INTEGER,
                [availability_365] INTEGER,
                [csv_file_path] TEXT,
                [download_date] TEXT,
                [link] TEXT
            )
            """
        )

        # Insert data from csv to sqlite
        reader = pd.read_csv(csv_file_path)
        reader.to_sql('LISTINGS', conn, if_exists='append', index=False)

        conn.commit()
        conn.close()
    except:
        conn.close()
        pass
```