Лабораторна робота №5

Добування бажаних бізнес оглядів

В попередньому зошиті ми досягли двох речей. По-перше, ми визначили категорії бізнесу (italian / Pizza), на які буде спрямована наша кампанія. По-друге, ми визначили ідентифікатори підприємств (business ID), пов'язані з цими категоріями. Знання business ID дозволяє вибрати з файлу відгуків лише ті відгуки, які нас цікавлять. Перш ніж поспішати читати дані оглядів в один великий фрейм даних, щоб потім відфільтрувати лише ті ідентифікатори, які вас цікавлять, ви, як Data Scientist, спочатку повинні отримати уявлення про розмір файлу. Файл оглядів досить великий (майже шість мільйонів рядків). Якщо у вас дуже хороший комп'ютер з великим обсягом оперативної пам'яті, або ви чомусь хочете запустити монітор системи / пам'яті і спостерігати, як вільна пам'ять падає все далі і далі, поки комп'ютер не захопить її всю, вам варто замислитися про те, як читати лише ті рядки, які нас дійсно цікавлять. Таким чином, ми будемо споживати лише мінімум необхідної оперативної пам'яті. Звичайно, загалом, навіть у цьому випадку нам слід робити якийсь розрахунок, чи відповідає файл можливостям комп'ютера. У цьому випадку слід мати подвійний запас ресурсів для більшості сучасних комп'ютерів. Завдання цього зошита - проаналізувати файл даних з оглядами та зберегти (набагато меншу) підмножину даних, які нас власне цікавлять.

In [1]:

```
# import the required libraries here
import pandas as pd
```

Список необхідних business IDs

Спочатку нам потрібно прочитати список business ID, який ми отримали раніше.

In [2]:

```
# task: read in our previously created file that contains the business IDs of interes
# one line
businesses = pd.read_csv('out/04_lab4_italian_pizza.csv')
```

Важливо спочатку зробити швидку перевірку зчитаних даних.

In [3]:

```
businesses.head()
```

Out[3]:

	business_id	review_count	stars	state
0	fweCYi8FmbJXHCqLnwuk8w	16	4.0	OH
1	PZ-LZzSlhSe9utkQYU8pFg	40	4.0	NV
2	BvYU3jvGd0TJ7IyZdfiN2Q	16	3.5	NC
3	PkDghu4aan2_wxrhXjTEgg	16	3.5	AB
4	RFbMVekR8lU9tPJ8sWrwHg	18	4.0	ON

Ми хочемо відобразити лише колонку business_id як список. Варто також зробити подвійну перевірку, чи відповідає він стовпцю business_id, наведеному вище.

In [4]:

```
business_ids = businesses['business_id'].values
print(business_ids[:5])
```

```
['fweCYi8FmbJXHCqLnwuk8w' 'PZ-LZzSlhSe9utkQYU8pFg'
 'BvYU3jvGd0TJ7IyZdfiN2Q' 'PkDghu4aan2_wxrhXjTEgg'
 'RFbMVekR8lU9tPJ8sWrwHg']
```

Фільтрування великого файлу даних фрагментами

Файл даних з оглядами містить близько 6 мільйонів рядків. Нам немає потреби використовувати всі ці рядки. Наш загальний підхід полягає в тому, щоб прочитати файл і перевірити business_id у кожному рядку. Якщо цей ідентифікатор міститься в нашому списку розшукуваних ідентифікаторів, ми збережемо цей рядок. Це буде найшвидше, але найдорожче для пам'яті, за один раз прочитати весь файл з диску, а потім фільтрувати рядки в пам'яті. З іншого боку, найефективніше з точки зору використання пам'яті читати по одному запису за один раз, зразу ж перевіряти, чи потрібний нам цей рядок (і відкинути, якщо ні). Такий підхід буде найповільнішим. Ми можемо досягти балансу, коли будемо вибирати достатньо великий фрагмент, який все ще легко вписується в пам'ять, але при цьому нам не потрібно читати занадто багато фрагментів.

In [5]:

```
# For convenience here, again we are accessing the data in the working directory that
# Normal best practise is to keep your data separate, but this keeps things simple he
# If you're comfortable specifying a filepath to files outside of this working direct
# task: create a reader object for the review json file
# Hint: use lines=True as before but add the chunksize=100000 parameter
# one line of code here
review_reader = pd.read_json('JSON/review.json', chunksize=100000, lines=True)
```

Ми використовуємо магію часу для наступної комірки, не цікавлячись тим, скільки часу це потребуватиме. Якщо у вас є час і вам цікаво, ви можете поекспериментувати з розміром фрагменту і контролювати використання системної пам'яті та час, який займає робота цієї комірки. Я пропоную вам зберегти свій зошит перед запуском цієї комірки, про всяк випадок.

In [6]:

```
%%time
# task: process the file one chunk at a time,
# filter that chunk for rows with a business_id in business_ids
# You can either do this within in a loop, having initialized an empty list,
# or using a more pythonic list comprehension
reviews = pd.concat([chunk.loc[chunk.business_id.isin(business_ids)] for chunk in revi
# (this took some 24 minutes on my old i7)
```

```
CPU times: user 1min 17s, sys: 9.25 s, total: 1min 26s
Wall time: 1min 26s
```

Після успішного виконання коду у вас є прочитані бажані огляди. Але ми ще не маємо їх у зручній DataFrame:

In [7]:

```
type(reviews)
```

Out[7]:

```
pandas.core.frame.DataFrame
```

In [8]:

```
len(reviews)
```

Out[8]:

```
594862
```

Перетворіть свої огляди в структуру DataFrame. Підказка: тут ви можете скористатися методом concat бібліотеки pandas.

In [9]:

```
# task convert your reviews into a DataFrame
# one line of code here
reviews
```

Out[9]:

	review_id	user_id	business_id	stars	useful
5	fdiNeiN_hoCxCMy2wTRW9g	w31MKYsNFMrjhWxxAb5wIw	eU_713ec6fTGNO4BegRaww	4	0
12	Z7wgXp98wYB57QdRY3HQ3w	GYNnVehQeXjty0xH7-6Fhw	FxLfqxdYPA6Z85PFKaqLrg	4	0
15	svK3nBU7Rk8VfGorlrN52A	NJlxGtouq06hhC7sS2ECYw	YvrylyuWgbP90RgMqZQVnQ	5	0
19	4bUyL7lzoWzDZaJETAKREg	_N7Ndn29bpll_961oPeEfw	y-Iw6dZflNix4BdwIyTNGA	3	0
20	Amo5gZBvCuPc_tZNpHwtsA	DzZ7piLBF-WsJxqosfJgtA	qx6WhZ42eDKmBchZDax4dQ	5	1
...	...	...	...	...	...
6685880	FVXOrfpHgge6oleGkS9kNA	CwkEpm0lCai2cJZIp_MMiQ	hlrZHM4D48XiQtXh6cRg_w	3	2
6685886	7bh0zTi1gEJwRCESsxH2Wg	TdbTpjfhOUeTCVBX9evLkw	FvXZcRB8bocNMDvFUnoWhg	5	2
6685890	yZbleOxC4_k8n4FclEBIRw	pSvD8Djf3g8eR9VmgaRKhA	3kdSl5mo9dWC4clrQjEDGg	5	0
6685891	bzQFrsMDtGIVOeXp5gbNeg	g5K5E9QrxObn6ZC1q8Mb9g	CRVtzesMuwHK-phmS_ojaA	4	0
6685896	jPCXuSBbI_cPocgoNBhdkg	V6BjjQICDO4q7TT3ZhaPWw	yA6dKNm_zl1ucZCnwW8ZCg	1	13

594862 rows × 9 columns

In [10]:

```
reviews.shape
```

Out[10]:

```
(594862, 9)
```

In [11]:

```
reviews.head()
```

Out[11]:

	review_id	user_id	business_id	stars	useful	funny
5	fdiNeiN_hoCxCMy2wTRW9g	w31MKYsNFMrjhWxxAb5wIw	eU_713ec6fTGNO4BegRaww	4	0	0
12	Z7wgXp98wYB57QdRY3HQ3w	GYNnVehQeXjty0xH7-6Fhw	FxLfqxdYPA6Z85PFKaqLrg	4	0	0
15	svK3nBU7Rk8VfGorlrN52A	NJlxGtouq06hhC7sS2ECYw	YvrylyuWgbP90RgMqZQVnQ	5	0	0
19	4bUyL7lzoWzDZaJETAKREg	_N7Ndn29bpll_961oPeEfw	y-Iw6dZflNix4BdwIyTNGA	3	0	0
20	Amo5gZBvCuPc_tZNpHwtsA	DzZ7piLBF-WsJxqosfJgtA	qx6WhZ42eDKmBchZDax4dQ	5	1	0

Збереження даних

Зробивши всю цю важку роботу, фільтруючи файл оглядів і задокументувавши процес у цьому зошиті, нам потрібно зберегти дані. Збережемо огляди структури DataFrame на csv, який назвемо "reviews_filtered.csv".

In [ ]:

```
# task: save the DataFrame to the specified file now.
# don't forget to use index=False
reviews.to_csv('out/05_lab5_reviews_filtered.csv', index=False)
```

Якщо поглянути на розмір оригінального файлу json та наш новий файл csv, побачимо, що ми перейшли від 4,4 ГБ до 325 МБ. Це набагато зручніше!

Підсумок

У цьому проекті ви мали навчитися, як вирішити проблему та визначити відповідні дані, використовувати дані, щоб отримати уявлення про проблему та прийняти рішення, а потім використовувати ці знання для отримання потрібної підмножини даних з надзвичайно великого файлу. У наступному зошиті ми зануримось у ці дані.