pyensae_flat2db3

January 17, 2022

1 Import a flat file into a SQLite database

Importing a flatfile can be done with *pandas*. *pyensae* proposes a function which does so by guessing the schema over the first lines.

[1]:
```
import pyensae
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.0.1 Mix SQLite and DataFrame

When a dataset is huge (~3Gb), it takes some time to load it into a DataFrame. It is difficult to look at it in any tool (Python, Excel, ...) One option I usually do is to load it a SQL server if you have one. If you do not, then SQLite is the best option. Let's see how it works with a custom datasets.

[2]:
```
import pyensae
import pyensae.datasource
pyensae.datasource.download_data("velib_vanves.zip", website = "xd")
```

```
    downloading of
http://www.xavierdupre.fr/enseignement/complements/velib_vanves.zip  to
velib_vanves.zip
    unzipped  velib_vanves.txt  to  .\velib_vanves.txt
```

[2]:
```
['.\\velib_vanves.txt']
```

As this file is small (just an example), let's see how it looks like with a DataFrame.

[3]:
```
import pandas
df = pandas.read_csv("velib_vanves.txt",sep="\t")
df.head(n=2)
```

[3]:

	address	available_bike_stands	available_bikes	banking	bike_stands	bonus	contract_name
0	112 RUE VERCINGETORIX - 75014 PARIS	65	2	0	67	0	Paris
1	112 RUE VERCINGETORIX - 75014 PARIS	65	2	0	67	0	Paris

```
        last_update          lat          lng                            name  \
0  15/07/2013 15:00  48,83425925  2,313391647  14029 - GERGOVIE VERCINGETORIX
1  15/07/2013 15:05  48,83425925  2,313391647  14029 - GERGOVIE VERCINGETORIX

   number status   idr
0   14029   OPEN   669
1   14029   OPEN  1898

[2 rows x 14 columns]
```

Then we import it into a SQLite3 database. The following function automatically guesses the table schema.

[4]:
```
from pyensae.sql import import_flatfile_into_database
import_flatfile_into_database("velib_vanves.db3", "velib_vanves.txt", add_key="key")
```

```
processing file  velib_vanves.txt
  TextFile: opening file  velib_vanves.txt
  TextFile: closing file  velib_vanves.txt
   swith to str  ('address', (<class 'str'>, 70))  value  112 RUE VERCINGETORIX
- 75014 PARIS
   swith to str  ('contract_name', (<class 'str'>, 10))  value  Paris
   swith to str  ('last_update', (<class 'str'>, 32))  value  15/07/2013 15:00
   swith to str  ('lat', (<class 'str'>, 22))  value  48,83425925
   swith to str  ('lng', (<class 'str'>, 22))  value  2,313391647
   swith to str  ('name', (<class 'str'>, 60))  value  14029 - GERGOVIE
VERCINGETORIX
   swith to str  ('status', (<class 'str'>, 8))  value  OPEN
   guess {0: ('address', (<class 'str'>, 70)), 1: ('available_bike_stands',
<class 'int'>), 2: ('available_bikes', <class 'int'>), 3: ('banking', <class
'int'>), 4: ('bike_stands', <class 'int'>), 5: ('bonus', <class 'int'>), 6:
('contract_name', (<class 'str'>, 10)), 7: ('last_update', (<class 'str'>, 32)),
8: ('lat', (<class 'str'>, 22)), 9: ('lng', (<class 'str'>, 22)), 10: ('name',
(<class 'str'>, 60)), 11: ('number', <class 'int'>), 12: ('status', (<class
'str'>, 8)), 13: ('idr', <class 'int'>)}
columns  {0: ('address', (<class 'str'>, 70)), 1: ('available_bike_stands',
<class 'int'>), 2: ('available_bikes', <class 'int'>), 3: ('banking', <class
'int'>), 4: ('bike_stands', <class 'int'>), 5: ('bonus', <class 'int'>), 6:
('contract_name', (<class 'str'>, 10)), 7: ('last_update', (<class 'str'>, 32)),
8: ('lat', (<class 'str'>, 22)), 9: ('lng', (<class 'str'>, 22)), 10: ('name',
(<class 'str'>, 60)), 11: ('number', <class 'int'>), 12: ('status', (<class
'str'>, 8)), 13: ('idr', <class 'int'>), 14: ('key', <class 'int'>,
'PRIMARYKEY', 'AUTOINCREMENT')}
SQL  'CREATE TABLE velib_vanves(address TEXT,'
'       available_bike_stands INTEGER,'
'       available_bikes INTEGER,'
'       banking INTEGER,'
'       bike_stands INTEGER,'
'       bonus INTEGER,'
'       contract_name TEXT,'
'       last_update TEXT,'
'       lat TEXT,'
'       lng TEXT,'
'       name TEXT,'
'       number INTEGER,'
```

```
'       status TEXT,'
'       idr INTEGER,'
'       key INTEGER PRIMARY KEY AUTOINCREMENT);'
  TextFile: opening file  velib_vanves.txt
  TextFile: closing file  velib_vanves.txt
9461  lines imported
```

We check the database exists:

[5]:
```
import os
os.listdir(".")
```

[5]:
```
['pyensae_flat2db3.ipynb',
 'pyensae_StockPrices.ipynb',
 'pyensae_velib.ipynb',
 'velib_vanves.db3',
 'velib_vanves.txt',
 'velib_vanves.zip']
```

On Windows, you can use SQLiteSpy to visualize the created table. We use pymysintall to download it.

[6]:
```
try:
    from pymyinstall.installcustom import install_sqlitespy
    exe = install_sqlitespy()
except:
    # we skip an exception
    # the website can be down...
    exe = None
exe
```

```
SQLiteSpy, version  1.9.7
download  http://www.yunqa.de/delphi/lib/exe/fetch.php?hash=938481&media=http
%2F%2Fwww.yunqa.de%2Fdelphi%2Fdownloads%2FSQLiteSpy_1.9.7.zip
len  1828282
    unzipped  SQLiteSpy.exe  to  .\SQLiteSpy.exe
    unzipped  World.db3  to  .\World.db3
```

[6]:
```
'.\\SQLiteSpy.exe'
```

We just need to run it (see run_cmd).

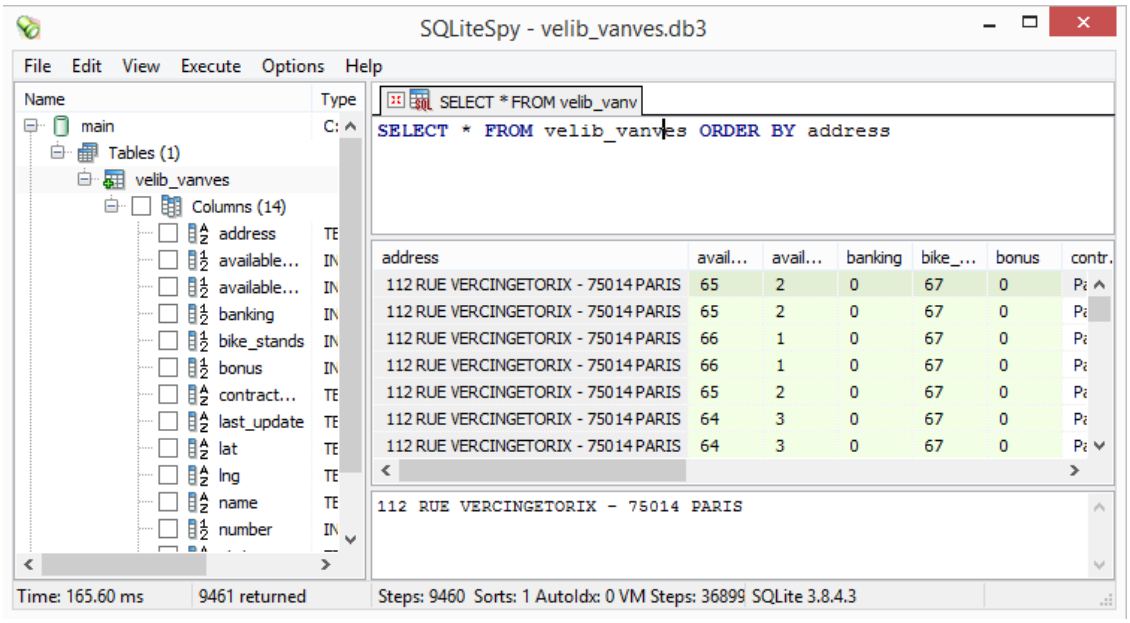
[7]:
```
if exe:
    from pyquickhelper import run_cmd
    run_cmd("SQLiteSpy.exe velib_vanves.db3")
```

[7]:
```
('', '')
```

You should be able to see something like (on Windows):

[8]:
```
from pyquickhelper.helpgen import NbImage
NbImage('img_nb_sqlitespy.png')
```

[8]:



It is easier to use that tool to extract a sample of the data. Once it is ready, you can execute the SQL query in Python and converts the results into a DataFrame. The following code extracts a random sample from the original sets.

[9]:
```
sql = """SELECT * FROM velib_vanves WHERE key IN ({0})"""

import random
from pyquickhelper.loghelper import noLOG
from pyensae.sql import Database
db = Database("velib_vanves.db3", LOG = noLOG)
db.connect()
mx = db.execute_view("SELECT MAX(key) FROM velib_vanves")[0][0]
rnd_ids = [ random.randint(1,mx) for i in range(0,100) ]  # liste de 100 id aléatoires
strids  = ",".join( str(_) for _ in rnd_ids )
res = db.execute_view(sql.format (strids))
df = db.to_df(sql.format (strids))
db.close()
df.head()[["key","last_update","available_bike_stands","available_bikes"]]
```

[9]:

	key	last_update	available_bike_stands	available_bikes
0	24	15/07/2013 16:55	66	1
1	26	15/07/2013 17:05	66	1
2	178	16/07/2013 05:45	49	18
3	220	16/07/2013 09:15	63	4
4	342	16/07/2013 19:25	61	6

[5 rows x 4 columns]

Memory Dump

Once you have a big dataset available in text format, it takes some time to load into memory and you need to do that every time you need it again after you closed your python instance.

[10]:
```
with open("temp_big_file.txt","w") as f :
    f.write("c1\tc2\tc3\n")
    for i in range(0,10000000):
        x = [ i, random.random(), random.random() ]
        s = [ str(_) for _ in x ]
        f.write( "\t".join(s) + "\n" )
os.stat("temp_big_file.txt").st_size
```

[10]:
```
474285221
```

[11]:
```
import pandas,time
t = time.perf_counter()
df = pandas.read_csv("temp_big_file.txt",sep="\t")
print("duration (s)",time.perf_counter()-t)
```

```
duration (s) 8.405750774129501
```

It is slow considering that many datasets contain many more features. But we can speed it up by doing a kind of memory dump with to_pickle.

[12]:
```
t = time.perf_counter()
df.to_pickle("temp_big_file.bin")
print("duration (s)",time.perf_counter()-t)
```

```
duration (s) 2.2846547239112738
```

And we reload it with read_pickle:

[13]:
```
t = time.perf_counter()
df = pandas.read_pickle("temp_big_file.bin")
print("duration (s)",time.perf_counter()-t)
```

```
duration (s) 0.7951709542244885
```

It is 10 times faster and usually smaller on the disk.

[14]:
```
```