seance5_sql_multidimensionnelle_correction

July 2, 2023

1 Données multidimensionnelles SQL - correction

Correction de la séance sur l'utilisation du SQL depuis un notebook.

[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('ggplot')
from pyquickhelper.helpgen import NbImage
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

```
Populating the interactive namespace from numpy and matplotlib
```

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

1.1 Exercice 1 : filtre

On veut comparer les espérances de vie pour deux pays et deux années.

[2]:
```
from actuariat_python.data import table_mortalite_euro_stat
table_mortalite_euro_stat()
import pandas
df = pandas.read_csv("mortalite.txt", sep="\t", encoding="utf8", low_memory=False)
# ...
```

[3]:
```
import os
if not os.path.exists('mortalite.db3'):
    import sqlite3
    from pandas.io import sql
    cnx = sqlite3.connect('mortalite.db3')
    df.to_sql(name='mortalite', con=cnx)
    cnx.close()
```

1.2 Exercice 2 : échantillon aléatoire

[4]:
```
import random  #loi uniforme
def echantillon(proportion):
    return 1 if random.random() < proportion else 0
```

[5]:
```
import sqlite3
from pandas.io import sql
cnx = sqlite3.connect('mortalite.db3')
```

[6]:
```
cnx.create_function('echantillon', 1, echantillon)
```

Que faut-il écrire ici pour récupérer 1% de la table ?

[7]:
```
import pandas
#example = pandas.read_sql(' ???  ', cnx)
#example
```

[8]:
```
cnx.close()
```

1.3 Exercice 3 : reducer SQL

[9]:
```
import sqlite3, pandas
from pandas.io import sql
cnx = sqlite3.connect('mortalite.db3')
```

Il faut compléter le programme suivant.

[10]:
```
class ReducerMediane:
    def __init__(self):
        self.indicateur = []
    def step(self, value):
        if value >= 0:
            self.indicateur.append(value)
    def finalize(self):
        self.indicateur.sort()
        return self.indicateur[len(self.indicateur)//2]

cnx.create_aggregate("ReducerMediane", 1, ReducerMediane)
```

[11]:
```
query = """SELECT annee,age,age_num, ReducerMediane(valeur) AS mediane FROM mortalite
                WHERE indicateur=="LIFEXP" AND genre=="F"
                GROUP BY annee,age,age_num"""
df = pandas.read_sql(query, cnx)
```

[12]:
```
df.head()
```

[12]:

	annee	age	age_num	mediane
0	1960	None	NaN	66.7
1	1960	Y01	1	73.7
2	1960	Y02	2	72.8
3	1960	Y03	3	71.9
4	1960	Y04	4	71.0

Un reducer à deux entrées même si cela n'a pas beaucoup de sens ici :

[13]:
```
class ReducerMediane2:
    def __init__(self):
        self.indicateur = []
    def step(self, value, value2):
        if value >= 0:
```

```
            self.indicateur.append(value)
        if value2 >= 0:
            self.indicateur.append(value2)
    def finalize(self):
        self.indicateur.sort()
        return self.indicateur[len(self.indicateur)//2]

cnx.create_aggregate("ReducerMediane2", 2, ReducerMediane2)
```

[14]:
```
query = """SELECT annee,age,age_num, ReducerMediane2(valeur, valeur+1) AS mediane2␣
 ↪FROM mortalite
                WHERE indicateur=="LIFEXP" AND genre=="F"
                GROUP BY annee,age,age_num"""
df = pandas.read_sql(query, cnx)
df.head()
```

[14]:

	annee	age	age_num	mediane2
0	1960	None	NaN	66.7
1	1960	Y01	1	74.0
2	1960	Y02	2	73.2
3	1960	Y03	3	72.3
4	1960	Y04	4	71.3

Il n'est apparemment pas possible de retourner deux résultats mais on peut utiliser une ruse qui consise à les concaténer dans une chaîne de caracères.

[15]:
```
class ReducerQuantile:
    def __init__(self):
        self.indicateur = []
    def step(self, value):
        if value >= 0:
            self.indicateur.append(value)
    def finalize(self):
        self.indicateur.sort()
        q1 = self.indicateur[len(self.indicateur)//4]
        q2 = self.indicateur[3*len(self.indicateur)//4]
        n = len(self.indicateur)
        return "%f;%f;%s" % (q1,q2,n)

cnx.create_aggregate("ReducerQuantile", 1, ReducerQuantile)
```

[16]:
```
query = """SELECT annee,age,age_num, ReducerQuantile(valeur) AS quantiles FROM␣
 ↪mortalite
                WHERE indicateur=="LIFEXP" AND genre=="F"
                GROUP BY annee,age,age_num"""
df = pandas.read_sql(query, cnx)
df.head()
```

[16]:

	annee	age	age_num	quantiles
0	1960	None	NaN	4.400000;72.800000;20
1	1960	Y01	1	73.000000;74.000000;10
2	1960	Y02	2	72.100000;73.200000;10
3	1960	Y03	3	71.200000;72.300000;10
4	1960	Y04	4	70.300000;71.300000;10

[17]:
```
cnx.close()
```

[18]:
```

```