ÉCOLE CENTRALE LYON

Rapport de Bureau d'Etude

INF-tc2

BE n°3 : Programmation Orientée Objet, SQL et Matplotlib

Groupe Ab2

Marijan SORIC
Rémi LAURENS-BERGE

Encadrant

Stéphane DERRODE

Année 2021 - 2022 mars 11

Table des matières

1 La base de donnée Hotellerie.db **1**
1.1 Présentation de la base de donnée 1
1.2 La classe HotelDB 1

2 Requêtes libres **2**
2.1 Requête 1 3
2.2 Requête 2 5

1 La base de donnée Hotellerie.db

1.1 Présentation de la base de donnée

L'objectif de ce BE est de travailler sur une base de donnée appelée "*hotellerie.db*". Elle possède 5 tables : $hotel, client, chambre, occupation$ et $rservation$. Nous pouvons voir en détail ces tables à l'aide du logiciel DB Browser For SQLite, et on peut également écrire des requêtes SQL et les éxecuter. Cependant, l'objectif de ce travail est de travailler sur la base de donnée depuis Python.

1.2 La classe HotelDB

Nous souhaitons créer une classe sur Python pour réaliser des requêtes SQL sur la base Hotellerie.db directement depuis Python.

Dans un premier temps, nous voulons créer une fonction que l'on appelle get_name_hotel_etoile(n) et qui permet d'afficher le nom de tous les hotêls possédant n étoiles. On utilise des variables privés car elles n'ont pas vocation à être utilisées en dehors de la classe. Le code de cette fonction est le suivant :

```
1  def get_name_hotel_etoile(self,n):
2          curseur = self.conn__.cursor()
3          liste = []
4          try:
5              if  n <= 0:
6                  raise ValueError("Le nombre d' toiles  doit  tre
       strictement positif")
7              curseur.execute("SELECT nom FROM hotel WHERE etoiles = {};".
       format(n))
8          except sqlite3.OperationalError as sqlerr:
9              print('{} in {}'.format(sqlerr, self.__DBname))
10         except TypeError as typerr:
11             print('TypeError: ', str(typerr))
12         except ValueError as valerr:
13             print('ValueError: ', str(valerr))
14         else:
15             liste = curseur.fetchall()
16
17         return liste
```

Dans cette fonction, on s'est rassuré que les rappels erronés renvoient une liste vide à l'aide de l'instruction try et la clause except et le programme nous informe aussi de quel type d'erreur il s'agit. L'appel de get_name_hotel_etoile(-1) informera d'une erreur *ValueError* et l'appel de get_name_hotel_etoile('Hello') informera d'une erreur de TypeError.

Dans un deuxième temps, nous allons ajouté à la classe une fonction qui permet d'ajouter un nouveau client dans la base de donnée *Hotellerie.db*. Le code de cette fonction est le suivant :

```
def ajout(self,nomclient,prenomclient):
        curseur = self.conn__.cursor()
        curseur.execute("SELECT numclient FROM client WHERE nom='{}' AND
    prenom ='{}'".format(nomclient, prenomclient))
        ligneAll = curseur.fetchall()
        if ligneAll == [] :
            curseur.execute("SELECT numclient FROM client")
            tout = (curseur.fetchall())
            liste = []
            for e in tout:
                liste.append(e[0])
            i=1
            while True:
                if i not in liste:
                    textesql="'"+str(i)+"','"+nomclient+"','"+
   prenomclient+"'"
                    curseur.execute("INSERT INTO client VALUES ("+
   textesql+")")
                    print("Nouveau client ajout : id : "+str(i))
                    self.conn__.commit()
                    break
                i+=1
        else:
            print('Le client existe deja!')
```

La fonction ajout permet grâce au if de vérifier si le client existe déjà dans la base de donnée et afiche dans ce cas le message 'Le client existe déjà!'. Si le client n'est pas déjà dans la base de donnée, la fonction affichera le message 'Nouveau client ajouté ' suivi de l'id qui représentera le client.

2 Requêtes libres

On choisit dans cette partie de travailler avec une nouvelle base de donnée de films ; elle se nomme *base_stanford*. Elle comporte trois tables : *'Student'* (étudiants, *'College'* (écoles dans lesquelles ils postulent) et *'Apply'* (résultats des étudiants selon les majeurs et les écoles).

2.1 Requête 1

On se propose une première requête visant à obtenir les proportions des majeurs demandées pour chaque "States". Ainsi, on aura accès aux majeurs les plus plébicitées par les étudiants et donc une carte des "spacialité" en fonction de la géographique. En effet, on regroupe les universitées localisées dans un même état pour obtenir ces statistiques. Les résultats obtenus permetteraient, par exemple pour les états, de connaître dans quel secteur investir ils pourraient investir pour dynamiser ou redynamiser la région. Par exemple, le code SQL écrit pour avoir le graphique de la Californie sera :

```
SELECT count(Apply.major), Apply.major
FROM Apply JOIN College ON Apply.cName = College.cName
WHERE College.state = "CA"
GROUP BY Apply.major
```

On joint les noms des écoles de la table *Apply* et ceux de la table *College* pour obtenir un tableau à deux colonnes avec le nombre de fois que la majeur ait été demandée et le nom de la majeur. Le GROUP BY permet de classer ce tableau par majeur pour n'avoir qu'une seule ligne pour chaque majeur. Il nous reste ensuite à afficher le résultat avec *Matplotlib* sous forme d'un diagramme en camembert. Le code est donc :

```
1  import sqlite3
2  import matplotlib.pyplot as plt
3
4  class Ouvrir:
5
6      def __init__(self, nom):
7          self.__DBname = nom
8          self.__conn = sqlite3.connect(self.__DBname)
9
10     def __str__(self):
11         return 'Base de donn es :'+self.__DBname
12
13     def requete_1(self,name_state):
14         curseur = self.__conn.cursor()
15         liste = []
16         try:
17             S = "SELECT count(Apply.major), Apply.major \
18             FROM Apply JOIN College ON Apply.cName = College.cName WHERE
       College.state = '{}' GROUP BY Apply.major".format(name_state)
19             curseur.execute(S)
20         except sqlite3.OperationalError as sqlerr:  #interception d'
       erreur sql
21             print('{} in {}'.format(str(sqlerr), self.__DBname))
22         except TypeError as typerr: #erreur de type
23             print('TypeError : ',str(typerr))
24         else:
25             liste = curseur.fetchall()
26         return liste
27
28     def __del__(self):
29         self.__conn.close()
```

```
30
31
32
33
34  if __name__ == '__main__':
35      aBase = Ouvrir('base-stanford.db')
36      state='CA'
37      req1 = aBase.requete_1(state)
38      print(req1)
39
40      x=[req1[i][0] for i in range(len(req1))]
41      y=[req1[i][1] for i in range(len(req1))]
42      plt.pie(x, labels = y, normalize = True, autopct = lambda x: str(
        round(x, 2)) + '\%')
43      plt.title('Proportion des majors demnad s dans les  coles  de
        Californie')
```

Le programme principal du code précédent permet d'afficher le diagramme pour la Californie. En exécutant le programme pour New York et le Massachusetts, on obtient :

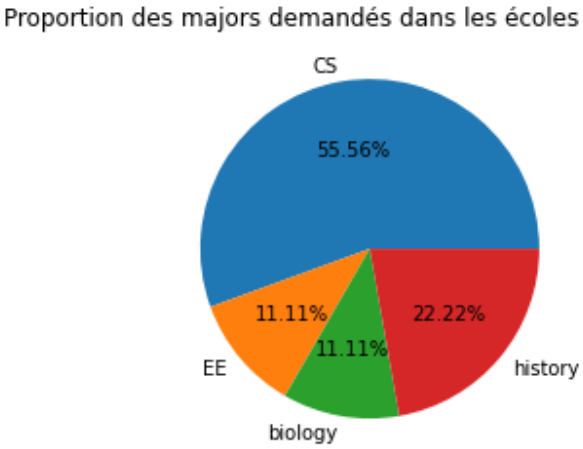

FIGURE 1 – Majeurs demandées dans les états de Californie

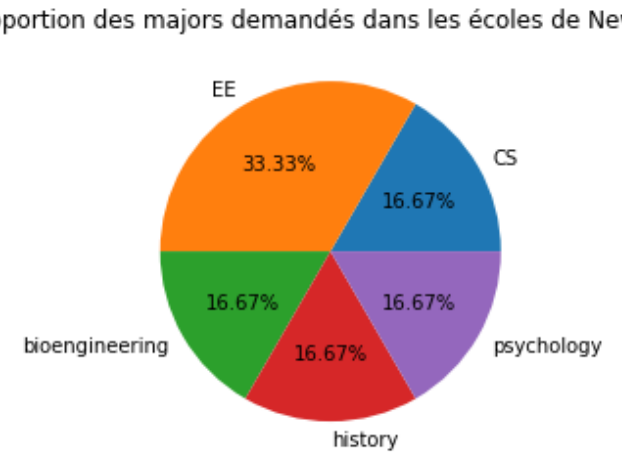

FIGURE 2 – Majeurs demandées dans les états de New-York

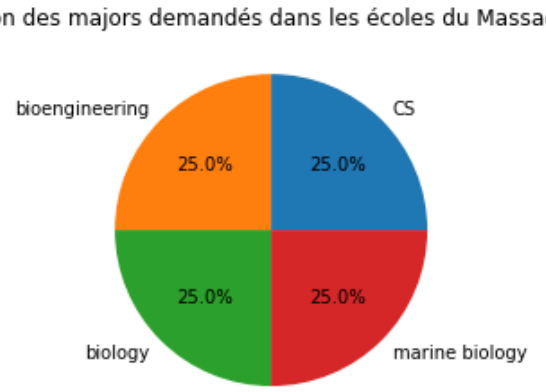

FIGURE 3 – Majeurs demandées dans les états du Massachusetts

Après avoir vérifié grâce a DB browser on a bien le bon résultat.

2.2 Requête 2

On se propose à l'étude d'une seconde requête permettant d'obtenir les taux d'acceptation des universités de la base de donnée, c'est-à-dire la mesure de la sélectivité de chacune des universités, indépendamment des fillières choisies par les étudiants. Pour ce faire, on utilise le simple rapport :

$$Selectivite = \frac{\sum [Decision = Y]}{\sum Demandes}$$

Où au numérateur on retrouve le nombre de réponses positives donné par l'université aux demandes d'intégration, et au dénominateur : le nombre de demande total reçu par l'université. Malgré l'échantillon restraint, cette requête permet d'avoir accès à la "cote de popularité" des universités, un indicateur intéressant pour les étudiants (dans leur

choix) mais aussi pour les recruteurs du monde du travail. Le niveau de sélectivité est un gage de niveau à l'entré. La requête SQL est la suivante :

```
SELECT cName,decision, count(*)
FROM Student JOIN Apply  ON Apply.sID = Student.sID
GROUP BY cName, decision
```

Qui revoit pour chaque université, le nombre d'acceptation et de refus délivré.

```
import sqlite3
import matplotlib.pyplot as plt
import numpy             as np

class Ouvrir:
    def __init__(self, nom):
        self.__DBname = nom
        self.__conn = sqlite3.connect(self.__DBname)

    def __str__(self):
        return 'Base de donn es :'+self.__DBname

    def requete_2(self):
        curseur = self.__conn.cursor()
        liste = []
        try:
            S = "SELECT cName,decision, count(*) \
                FROM Student JOIN Apply  ON Apply.sID = Student.sID \
                GROUP BY cName, decision"
            curseur.execute(S)
        except sqlite3.OperationalError as sqlerr:
            print('{} in {}'.format(str(sqlerr), self.__DBname))
        else:
            liste = curseur.fetchall()
        return liste

    def __del__(self):
        self.__conn.close()


if __name__ == '__main__':

    aBase = Ouvrir('base-stanford.db')
    req2 = aBase.requete_1()
    print(req2)

    TauxAcceptation = []
    College = []
    for tup in req2: #on remplit la liste des College (universit )
        if tup[0] not in College:
            College.append(tup[0])

    Demande = [0 for i in range(len(College))] #liste des nombres de
   demande pour chaque univesit  (class  dans le m me ordre que les
```

```
    colleges

    for j in range(len(College)):
        for i in range(len(req2)):
            if req2[i][0] == College[j]:
                Demande[j] += req2[i][2]

    for cName in College: #On remplit TauxAcceptation par le nombre de
d cision favorable d livr  par les colleges
        for tup in req2:
            if tup[0] == cName and tup[1] == 'Y':
                TauxAcceptation.append(tup[2])

    for i in range(len(TauxAcceptation)): #Enfin on obtient le Taux en
faisant le rapport : avis favorable/demande
        TauxAcceptation[i] = TauxAcceptation[i]/Demande[i]*100

    print(TauxAcceptation)

    #Trac  de l'histogram
    labels = College
    men_means = TauxAcceptation
    width = 0.5
    fig, ax = plt.subplots()
    ax.bar(labels, men_means, width, color = ['blue', 'purple', 'black',
    'red'])
    ax.set_ylabel('Taux d\'acceptation (en %)')
    ax.set_xlabel('Universit s')
    ax.set_title('Taux d\'accepation pour l\'  chantillon   observ ')
    ax.legend()
    plt.show()
```

Graphiquement, on obtient :

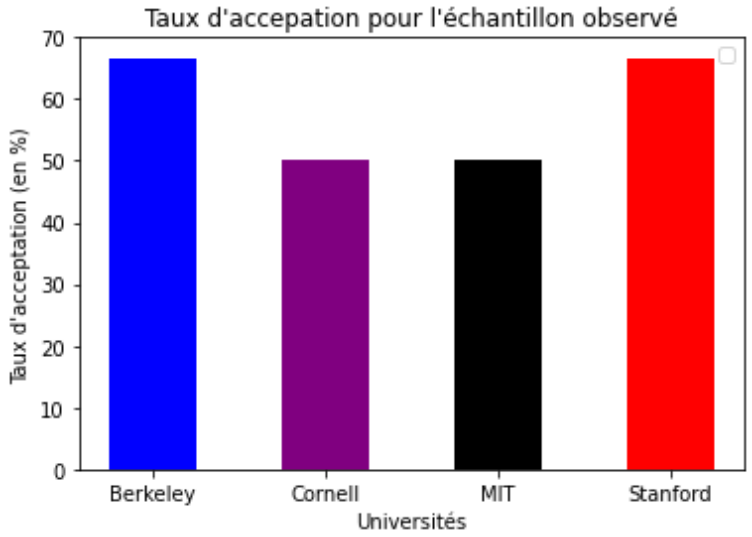

FIGURE 4 – Taux d'accepation par université

Table des figures

1 Majeurs demandées dans les états de Californie 4
2 Majeurs demandées dans les états de New-York 5
3 Majeurs demandées dans les états du Massachusetts 5
4 Taux d'accepation par université 7