ÉCOLE CENTRALE LYON

Informatique TC2 Rapport BE3

Groupe A2b

Utilisation de bases de données avec Python dans le cadre des élections présidentielles de 2017

Auteur :
M. Valentin VERDON

Encadrant :
M. Stéphane DERRODE

Version du
10 mars 2022

Table des matières

1 Introduction **1**

2 TD HotelDB **2**
2.1 Requête : get$_{n}ame_{h}otel_{e}toile$ 2
2.2 Requête : ajout d'un client 3

3 Etude de l'élection présidentielle de 2017 **4**
3.1 Première requête : l'abstention dans nos régions 4
3.2 Seconde requête : Différence de point entre les deux candidats dans une liste de région demandée 5

4 Conclusion **7**

1 Introduction

La première étape du BE3 a été la prise en main de la gestion de donnée sur Python à l'aide de la base de donnée *HotelDB*.

Les fonctions et leurs implémentations étant déjà étudiées en cours, nous reviendrons dessus brièvement (des commentaires ont été mis sur le fichier python pour expliquer le code).

Le sujet qui a été choisi est l'élection présidentielle francaise de 2017. Plus pariculièrement, les statistiques régionales relatives au second tour de cette élection.

Dans un premier temps, nous verrons une requête concernant les taux d'abstentions dans les régions puis, dans un second temps, nous étudierons une requête relative à la différence de voix entre les deux candidats dans les régions souhaitées.

2 TD HotelDB

2.1 Requête : $get_name_hotel_etoile$

La première requête consiste à renvoyer la liste des noms des hôtels ayant le nombre d'étoiles souhaité. Voici le code utilisé :

```
#l'objectif de cette requête est d'avoir la liste des noms des hotels ayant le nombre d'étoil
    def get_name_hotel_etoile(self,nbetoiles):
        curseur = self.__conn.cursor()
        liste=[]
        try:
            #vérification de la valeur insérée
            if nbetoiles<=0:
                raise ValueError("Le nombre d'étoile doit être strictement positif")

            #commande sql exécutée dans la bdd
            curseur.execute("SELECT nom FROM hotel WHERE Etoiles = " + str(nbEtoiles) +";")

        #erreurs renvoyées en cas de non conformité
        except TypeError as typerr:
            print('TypeError:',str(typerr))
        except ValueError as valerr:
            print('ValueError:',str(valerr))
        else:
            liste=curseur.fetchall()
        return liste
```

FIGURE 1 – requête recherche d'hôtel étoilé

On remarquera que les fonctions *try*, *except* et *raise* ont été utilisé afin de gérer des cas d'erreur qui pourrait être rencontrées lors de l'utilisation de la fonction :

```
#exemple d'erreur
nbEtoiles="a"
resultat = aHotelDB.get_name_hotel_etoile(nbEtoiles)
nbEtoiles=-1
resultat = aHotelDB.get_name_hotel_etoile(nbEtoiles)
```

(a) exemple de commandes erronées

```
In [1]: runfile('C:/Users/valen/Documents/Centrale Lyon/1A/
cours/INFO/python/BE3/HotelDB.py', wdir='C:/Users/valen/
Documents/Centrale Lyon/1A/cours/INFO/python/BE3')
TypeError: '<=' not supported between instances of 'str' and
'int'
ValueError: Le nombre d'étoile doit être strictement positif
```

(b) réponse générée

FIGURE 2 – Exemple d'erreurs possibles

Lorsque celle-ci est correctement utilisée, nous obenons ceci :

```
#test avec une bonne valeur
nbEtoiles = 2
resultat = aHotelDB.get_name_hotel_etoile(nbEtoiles)
print("Liste des noms d'hotel", nbEtoiles, "étoiles : ", resultat)
```

(a) exemple de commande

```
Liste des noms d'hotel 2 étoiles :  [('La nuit noire',), ('Hotel
chez soi',), ('Chez Philippe',)]
```

(b) réponse générée

FIGURE 3 – Exemple d'utilisation de la fonction

Ici, on souhaite ajouter un client. On va donc dans un premier temps vérifier s'il n'existe pas déjà dans la base de données puis si c'est le cas, on l'ajoute à l'aide de la commande SQL : *INSERT INTO*, ce qui nous donne le code suivant :

```
def ajout(self,nom,prenom):
    curseur=self.__conn.cursor()
    # on vérifie si le client n'existe pas déjà
    try:
        curseur.execute('SELECT numclient FROM client WHERE nom="'+nom+'" AND prenom="'+prenom+'";')
    except TypeError:
        print ("Les noms et prénoms sont des chaînes de caractères")
        return None
    liste=curseur.fetchall()
    if len(liste) !=0:
        print('Le client existe déjà')
        return liste[0][0]
    # on ajoute le client
    try:
        curseur.execute("INSERT INTO client(nom,prenom) VALUES ('{}','{}');".format(nom,prenom))
        self.__conn.commit()
    except TypeError:
        print ("Les noms et prénoms sont des chaînes de caractères")
        return None
    return curseur.lastrowid
```

FIGURE 4 – requête d'ajout d'un client

2.2 Requête : ajout d'un client

Comme dans la première fonction, on traite quelques cas d'erreur possible, voici ce que cela nous donne pour des erreurs et une bonne valeur :

```
#exemple d'erreur
res=aHotelDB.ajout(1,"bernard")

#test avec doublons de noms et prénoms
res=aHotelDB.ajout("Dupont","Marcel")

#ajout d'un client
res=aHotelDB.ajout("Tintin","Milou")
print(res)
```

(a) exemples de commandes

```
Les noms et prénoms sont des chaînes de caractères
Le client existe déjà
181
```

(b) réponses générées

FIGURE 5 – Exemple d'utilisation de la fonction

3 Etude de l'élection présidentielle de 2017

3.1 Première requête : l'abstention dans nos régions

La base de données sélectionnée contient les données régionales relatives au second tour des élections présidentielles de 2017. Ainsi, dans cette première requête, on souhaite avoir la liste des noms des régions ainsi que leur taux d'abstention avec un critère de minoration et de majoration sur ce taux d'abstention.

Cela donne le code suivant :

```
class election:
    def __init__(self,bdd):
        self.__conn = sqlite3.connect(bdd)

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

    def selection_region(self,pourcentage_minimum_abst,pourcentage_max):
        curseur = self.__conn.cursor()
        try:
            curseur.execute('SELECT Libellédelarégion, "%Abs/Ins" FROM Presidentielle WHERE "%Abs/Ins" >='+str(int(pourcentage

        #Ici, on rappele que les bornes sont des nombres entre 0 et 100

        except ValueError:
            print("Les pourcentages d'abstentions doivent être des nombres entre 0 et 100")
            return None

        return curseur.fetchall()
```

FIGURE 6 – requête recherche des régions par rapport au taux d'abstention

Comme ce qui as été réalisé en td, on a pris en compte certain cas d'exception sur les valeurs des encadrements du taux d'abstention.

NB : On remarquera que certaines lignes ont été tronquées car elles dépassaient du cadre de l'écran et empêchaient dans le cas contraire une bonne visibilité du code. Vous pouvez, si besoin, retrouver le code dans le fichier python joint avec ce rapport.

Voici des exemples d'utilisation et les réponses renvoyées :

```
#un exemple d'erreur
resultat=ab.selection_region(10,"cinquante")

#exemple de résultat
resultat=ab.selection_region(10,50)
print(resultat)
```

(a) exemple de commande

```
Les pourcentages d'abstentions doivent être des nombres entre 0 et
100
[('Bretagne', '20,31'), ('Pays de la Loire', '21,45'), ('Nouvelle-
Aquitaine', '22,15'), ('Bourgogne-Franche-Comté', '22,65'),
('Normandie', '22,81'), ('Centre-Val de Loire', '23,3'),
('Occitanie', '23,3'), ('Auvergne et Rhône-Alpes', '23,51'), ('Grand
Est', '24,69'), ('Hauts-de-France', '25'), ('Île-de-France', '25,3'),
("Provence-Alpes-Côte d'Azur", '25,61'), ('Corse', '36,02'), ('La
Réunion', '37,09'), ('Martinique', '49,74')]
```

(b) réponse générée

FIGURE 7 – Exemple d'utilisation de la fonction

A l'aide de ces résultats, on peut savoir la proportion de région ayant un taux d'abstention contenue dans l'intervalle étudié. Nous avons choisi de représenter cette proportion de région sous la forme d'un diagramme circulaire :

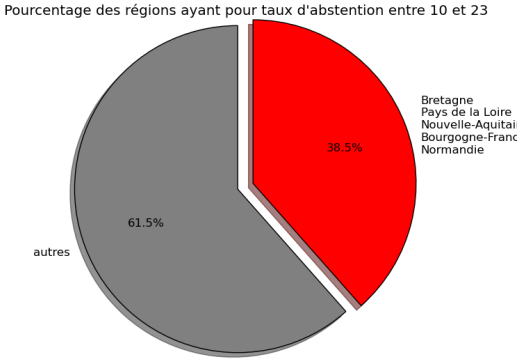

FIGURE 8 – Pourcentage des régions ayant entre 10 et 23 % d'abstention

On remarque que les taux d'abstentions dans 5 régions se situent entre 10 et 23% ce qui n'est pas négligeable ! On pourrai continuer notre étude en prenant des invtervalles pour des taux plus ou moins grand en fonction de l'objectif de l'étude.

3.2 Seconde requête : Différence de point entre les deux candidats dans une liste de région demandée

Dans cette seconde requête, nous nous intéressons à l'écart de point entre les deux candidats dans une sélection de région qui sera donnée sous forme d'une liste. Cette étude est intéressante car elle permet d'avoir une idée plus précise de l'avie de la population dans les différente région, savoir si l'avis était unanime

ou bien si la victoire dans la région était serrée.

Voici la requête :

```
def diffe_voix(self,liste_région):
    curseur = self.__conn.cursor()
    try:
        res=[]
        for i in liste_région:
            aux=curseur.execute('SELECT Libellédelarégion FROM Presidentielle WHERE Libellédelarégion = "'+i+'" ;')
            aux=curseur.fetchall()

            #test pour vérifier si la région existe
            if aux==[]:
                print(i+" n'est pas une région francaise")

            #réalisation de la requête
            else:
                aux=curseur.execute('SELECT Libellédelarégion, "%Voix/Ins1" - "%Voix/Ins2" FROM Presidentielle WHERE Libellédelarégion="'+i+'";')
                res.append(curseur.fetchall())

    #gestion de l'erreur de typage
    except TypeError:
        print("Les noms de région sont des chaînes de caractères")
    return res
```

FIGURE 9 – requête recherche des écarts de points entre les candidats dans une liste de régions donnée

Voici des cas d'utilisation de cette fonction et les réponses générées :

```
#exemple d'erreur
liste=ab.diffe_voix(["Occcid"])
liste=ab.diffe_voix([1])

#exemple de résultat
liste=ab.diffe_voix(["Île-de-France","Occitanie"])
print(liste)
```

(a) exemple de commande

```
Occcid n'est pas une région francaise
Les noms de région sont des chaînes de caractères
[[('Île-de-France', 39)], [('Occitanie', 17)]]
```

(b) réponse générée

FIGURE 10 – Exemple d'utilisation de la fonction

Il est intéressant d'avoir une vue globale des écarts entre les deux candidats sur l'ensemble des régions. Ainsi, avec notre fonction, on peut en déduire cet histogramme :

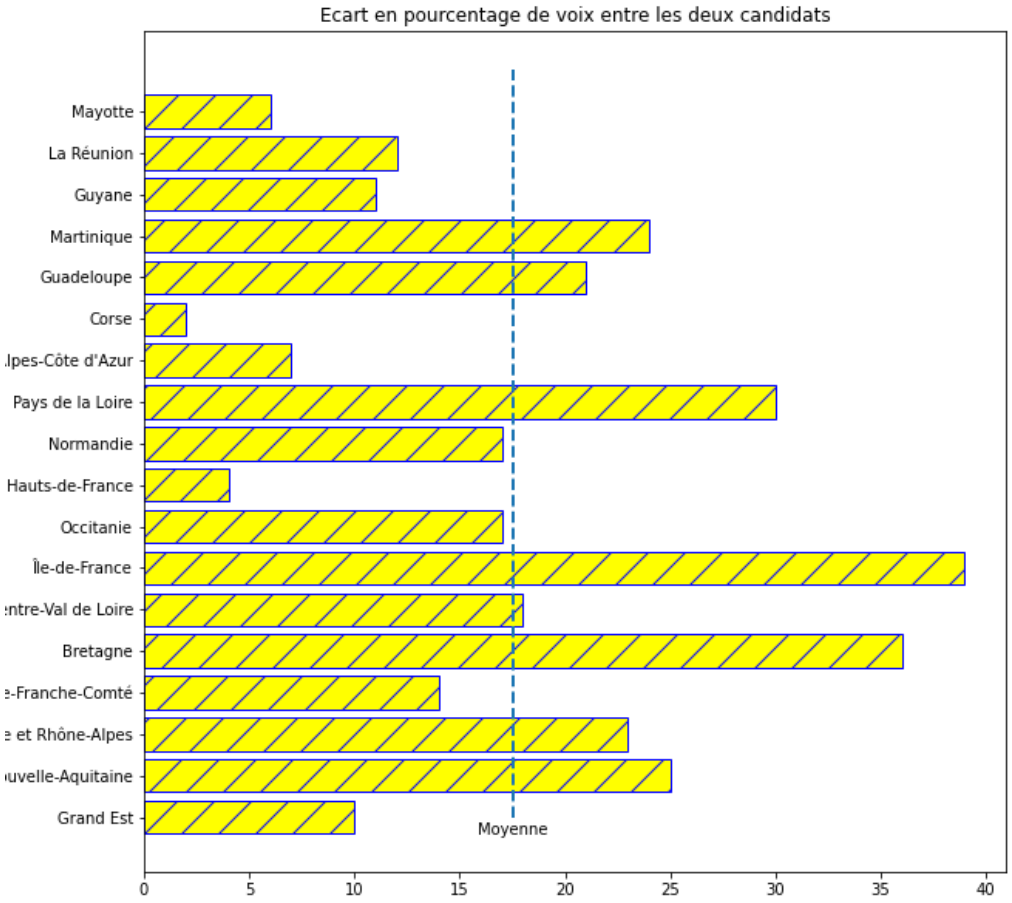

FIGURE 11 – Histogramme des écarts de points entre les deux candidats

Il est intéressant de voir que les écarts sont plus ou moins importants en fonction des régions. Par exemple, l'écart est plus serré dans les régions d'outre-mer comparé aux régions du territoire métropolitain.

4 Conclusion

Pour conclure, nous avons vu comment utiliser des requêtes SQL à l'aide de python. Nous avons de plus gérer les exceptions afin de rendre nos programmes

robutes. Finalement, nous avons utilisé ces connaissances dans deux cas de figures : la gestion d'un hotele et l'étude des résultats d'une élection présientielle.