

PROJET -- GESTION DES ETUDIANTS

Le but de ce projet est de vous exercer de façon « ludique » à la conception d'un élément du système d'information qui gère les étudiants de l'université, et de faire une étude statistique.

Les objectifs pédagogiques que vous devez atteindre sont :

- Le schéma relationnel d'une base de données
- L'utilisation du langage SQL à travers sqlite3
- L'utilisation du langage de python afin de développer une couche d'accès à une base de données
- Une étude statistique et exploitation des données représentées dans un dashboard

Méthode utilisée :

- SQL et Sqlite
- Python
- Les notions statistiques

Cours : Utilisation de Sqlite dans python

Le module permettant d'intégrer un SGBD à un environnement Python s'appelle sqlite3.

Voici quelques informations (très) basiques permettant l'utilisation de SQLite3 (et donc d'une base de données) directement dans du code python. De nombreux sites internet permettent d'approfondir ce lien (la documentation officielle est sur le site : <http://docs.python.org/2/library/sqlite3.html>).

La première chose à faire c'est d'importer le bon module ; dans python :

```
import sqlite3
```

Comme dans le cas de la ligne de commande, il faut ensuite se connecter à une base (si elle n'existe pas, elle sera créée)

```
connexion = sqlite3.connect('ma_base.db')
```

A la fin de votre programme, si vous voulez que les changements apportés à la base soient enregistrés :

```
connexion.commit()
```

Et puis n'oubliez pas de fermer proprement

```
connexion.close()
```

Une fois sqlite3 est importé, on se connecte à une base de données par l'intermédiaire de la fonction **connect**, en précisant en paramètre un chemin d'accès vers la base de données. Par exemple, pour utiliser la base de données **student.sqlite** on écrira :

```
import sqlite3 as sql
conn=sql.connect("student.sqlite") #Changer le chemin
```

L'objet **conn** est désormais en place, et vous allez pouvoir dialoguer avec lui à l'aide du langage SQL. On va d'abord mettre en place ce que l'on appelle un curseur. Il s'agit d'une sorte de tampon mémoire intermédiaire, destiné à mémoriser temporairement les données en cours de traitement, ainsi que les opérations que vous effectuez sur elles, avant leur transfert définitif dans la base de données. Cette technique permet donc d'annuler si nécessaire une ou plusieurs opérations qui se seraient révélées inadéquates (dans le cas où on modifie la base de données), et de revenir en arrière dans le traitement, sans que la base de données n'en soit affectée.

```
cur=conn.cursor()
```

Une fois le curseur créé, la méthode **execute** du curseur permet de transmettre des requêtes rédigées en SQL sous forme de chaîne de caractères :

```
cur.execute("SELECT * FROM communes WHERE dep_id=6 ORDER BY population DESC")
```

Dans le cas où l'on effectue des requêtes d'extraction de données dans la base (SELECT ...), on peut parcourir le curseur comme un itérable (et par exemple afficher les résultats)

```
for resultat in cur:
    print(resultat)
```

Ceci va vider le curseur, et l'on peut récupérer les résultats de la requête. Si l'on avait effectué plusieurs requêtes avant de « vider » le curseur, les résultats se seraient « enfilés » dans le curseur.

Enfin, si des modifications ont été effectuées sur la BDD (ce qu'on verra plus tard), il faut appliquer la méthode **commit** à la connexion créée pour qu'elles deviennent définitives. On peut ensuite refermer le curseur et la connexion :

```
conn.commit() #utile si une modification a eu lieu
cur.close()
conn.close()
```

Travaille à faire en binôme (2 max)

A rendre :

a-fichier doc contenant les captures écrans de l'exécution de toutes les requêtes et résultats de la requête

b-des fichiers text contenant tous les codes sources python, sql...avec commentaires

d- un rapport word expliquant les résultats et la méthode de travaille

Travail à faire

Nous allons créer et utiliser une base de données nommée ***student*** qui a pour schéma relationnel suivant :

Auteur (Nauteur ,nomA, prenomA, nationaliteA)

Livre (Nlivre, num_ISBN,, titre, nbPages, annéeS, prix)

Possede (Nlivre, Nauteur)

Pret (Npret, num_etu, Nlivre, datePret, dateRetour, DateRetourPrevue).

Etudiant (num_etu, nomE, prenomE, date_naissance, ville,
dateInscripBU,dateAbs,numClasse)

Class(numClass,nomclass)

Cours (num_cours, nomC, nb_heures, num_ens)

Enseignant (num_ens, nomP, prénomP, specialite, département)

Resultat (num_etu, num_cours, note)

Charge (num_cours, num_ens, nbH)

Inscrit(NumEtudiant,_num_cours, dateInsC)

Quelque explication de colonnes :

annéeS : année sortie livre

num_etu : numero étudiant,

Nlivre : numero livre

Npret : numero dupret

date_naissance : date,

ville : caractère variable de longueur 10,

dateInscripBU : date inscription à la BU

num_ens : numero enseignant

spécialité : specialite du prof

num_ens : représente numéro de l'enseignant responsable d'un cours

nbH : nombre d'heure (en heure et minute) effectué par le prof dans un cours

dateInsC : date inscription dans un cours

dateAbs : date d'absence de l'étudiant

numClass : numéro de la classe exemple GI3

Partie 1 : création de la base de données

A)

1-Dans un terminal, lancer sqlite3 de façon à créer la base de données ***student.sqlite***:

2-Après avoir ouvert une connexion ***conn*** et un curseur ***cur***, Créer les tables et les clés (primaire et étrangères) du schéma relations ci-dessus dans cette base de données.

Voici comment créer une table par exemple de nom ***cours_suivi*** :

```
cur.execute("CREATE TABLE IF NOT EXISTS cours_suivi (id_etu INTEGER, id_cours  
INTEGER,  
PRIMARY KEY (id_etu,id_cours), FOREIGN KEY(id_etu) REFERENCES etudiant(id),  
FOREIGN KEY(id_cours) REFERENCES cours(id))")
```

3-insérez un jeu de données (de votre choix ; contenant une cinquantaine de lignes pour les tables résultats et étudiant) dans les tables créées ; en utilisant *insert*

4- Sauvegarder la base et sortir de SQLite3.

5-Relancer sqlite3 avec la base **student** Vérifier que tout a bien fonctionné (les tables sont toujours là ? Les données aussi ?).

B) modification de la structure de la base

1-Augmenter la taille de la colonne nom de la table étudiant.

2-Ajouter le champ email à la table étudiant.

3-Ajouter la colonne adresse à la table adresse.

4-Supprimer les contraintes de la table pret.

5-Modifier la table pret en y ajoutant les contraintes suivantes :

a- Npret est clé primaire

a. num_etu est une clé étrangère

b. Nlivre est une clé étrangère fais référence à la table livre.

6-Modifier la table livre en y ajoutant la colonne Nauteur et la contrainte indiquant que cette colonne est une clé étrangère.

7-Supprimer la table possede.

8-Ajouter une contrainte à la table livre qui assure que titre de livre est une valeur unique.

9-Ajoute les champs langue et NbreExemplaires à la table livre.

10-Ajouter la contrainte qui assure l'unicité du numISBN.

11-Ajouter la contrainte qui spécifie que date_retour est >= date_pret.

Partie 2 : Python et sql

Écrire les fonctions en python :

1- de nom **insBU(*nomE*)** prenant en entrée ***nomE*** et renvoyant date inscription de l étudiant dans la bibliothèque, (On affichera une erreur si étudiant n'existe pas dans la base).

Exemple de sortie ecran :

```
>>> insBU( "Sofia")
```

```
2020-01-21
```

```
>>> insBU( "Atlas")
```

2020-01-25

2- de nom ***insCour(num_cours)*** prenant en entrée ***num_cours*** et renvoyant date la liste (Les noms et prenoms) des étudiants inscrits dans cours ***num_cours***

3- de nom ***ResuEtu(num_etu)*** prenant en entrée ***num_etu*** et renvoyant les résultats de l étudiant (nom, prénom, note obtenue, nom module ; moyenne note de la classe dans le moule, note superieure dans le module, note inferieure dans le module) obtenus dans les différents modules. A la fin le programme affiche aussi la moyenne générale de l étudiant.

<i>Nom étudiant</i>	Moyennes			
	Élève	Classe	M-	M+
Module				
Base des données	18,00	14,31	7,50	19,00
Programmation 1	19,00	17,48	13,00	19,00
Algebre	18,50	13,70	6,50	20,00

4-de nom ***resultEchec()*** donnant la liste des étudiants (nom, prenom, note ; la moyene classe) ayant une note inferieur à 10 et groupés par nom module

5- de nom ***insr()*** donnant les noms des étudiants inscrits dans tous les modules

6- de nom ***empLiv(Nlivre)*** prenant en entrée ***Nlivre*** et renvoyant les Noms des étudiants avec date retour , qui ont empruntés le livre pour code ***Nlivre***?

7- de nom ***retard()*** donnant les étudiants n ayant pas encore rendus au moins un livre

8- de nom ***noEmp()*** Donne les noms des livres empruntés par personne

9- de nom ***ResultTot()*** qui renvoie pour chaque classe le nom de la classe et la moyenne des notes obtenues par cours obtenue dans la classe.

4. Ecrire une requête en langage SQL qui détermine le nom du professeur de

Modification des tables

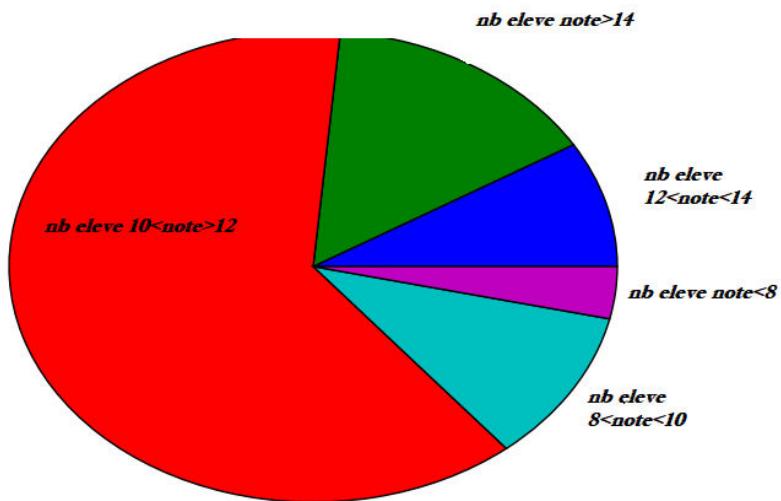
Donner des fonctions en python permettant de

[Q1]. Changer le nom du cours; en prenant num_cours en entrée

[Q2]. Supprimer un cours en prenant ***num_cours*** en entrée

Partie 3 étude statistique

1-Camembert. La fonction pie de matplotlib.pyplot permet le tracé d'un diagramme circulaire (« camembert» en français, « tarte » en anglais). Consulter l'aide (help(plt.pie) avec matplotlib.pyplot importé sous le nom plt) associée à cette fonction, puis tracer un diagramme circulaire dans lequel seront représentés ; les notes élèves (moyen générale) par catégorie de notes (voir exemple)

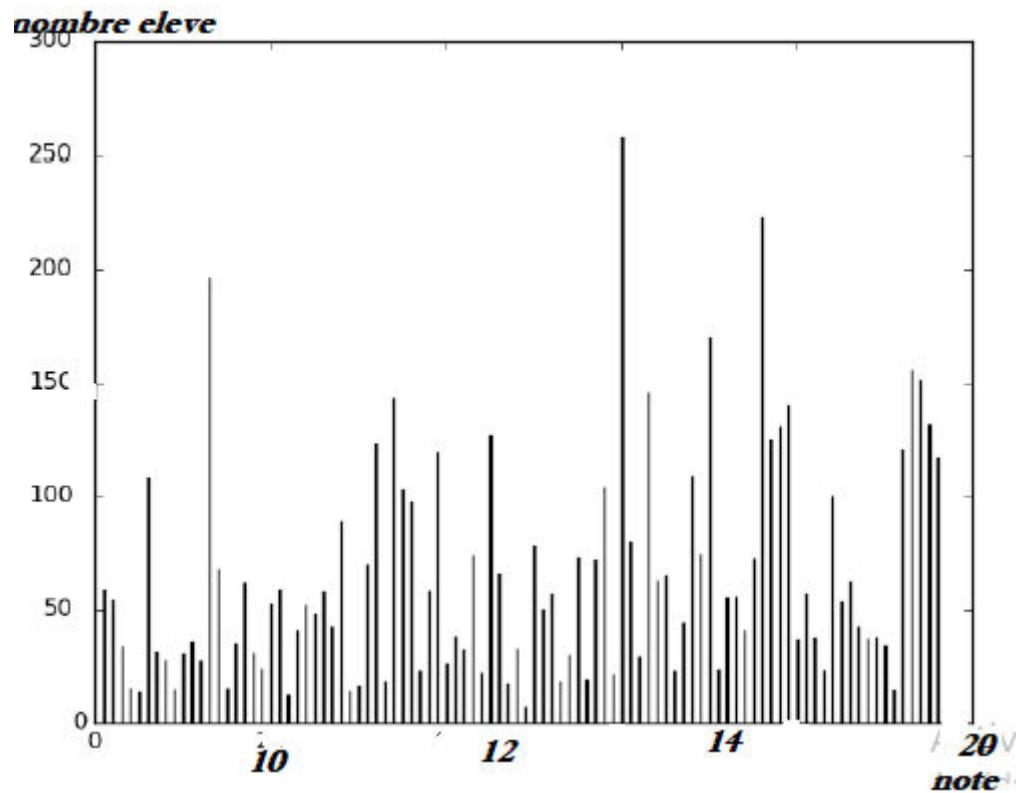


Exemple de camembert

2- Histogramme. De même, la fonction plt.bar permet de réaliser un histogramme (diagramme en « rectangles»). Il s'utilise par exemple comme suit :

```
x = [1,2,3,4,5,6,7,8,9,10]
hauteurs_barres = [8,12,8,5,4,3,2,1,0,0]
largeur_barres = 0.1
plt.bar(x, hauteurs_barres, largeur_barres)
plt.show()
```

On peut convertir le résultat d'une requête en liste (cur.execute(requete) puis list(cur)). Le résultat est une liste de tuples, chaque tuple étant associé à une ligne du résultat de la requête. Les types des éléments du tuple sont les mêmes que les attributs de la table associée au résultat de la requête. En exécutant une seule requête, réalisez un histogramme représentant le nombre d'étudiants par catégorie de note (moyens générales). Voir exemple ci dessous



Partie 4 : Analyse des données

Pour Obtenir des données plus précises sur les performances de notre établissement, nous allons présenter les données sous format de dashboard (tableau de bord voir figure A), pour cela ; nous allons utiliser **Power BI Desktop**.

Power BI Desktop est une application gratuite qui s'installe sur un ordinateur local et permet de se connecter à des données, de les transformer et de les visualiser. Avec **Power BI Desktop**, vous pouvez vous connecter à différentes sources de données et les associer (selon processus couramment nommé modélisation) dans un modèle de données permettant de créer des visuels et des collections de visuels, et de les partager sous forme de rapports avec d'autres personnes de votre organisation.

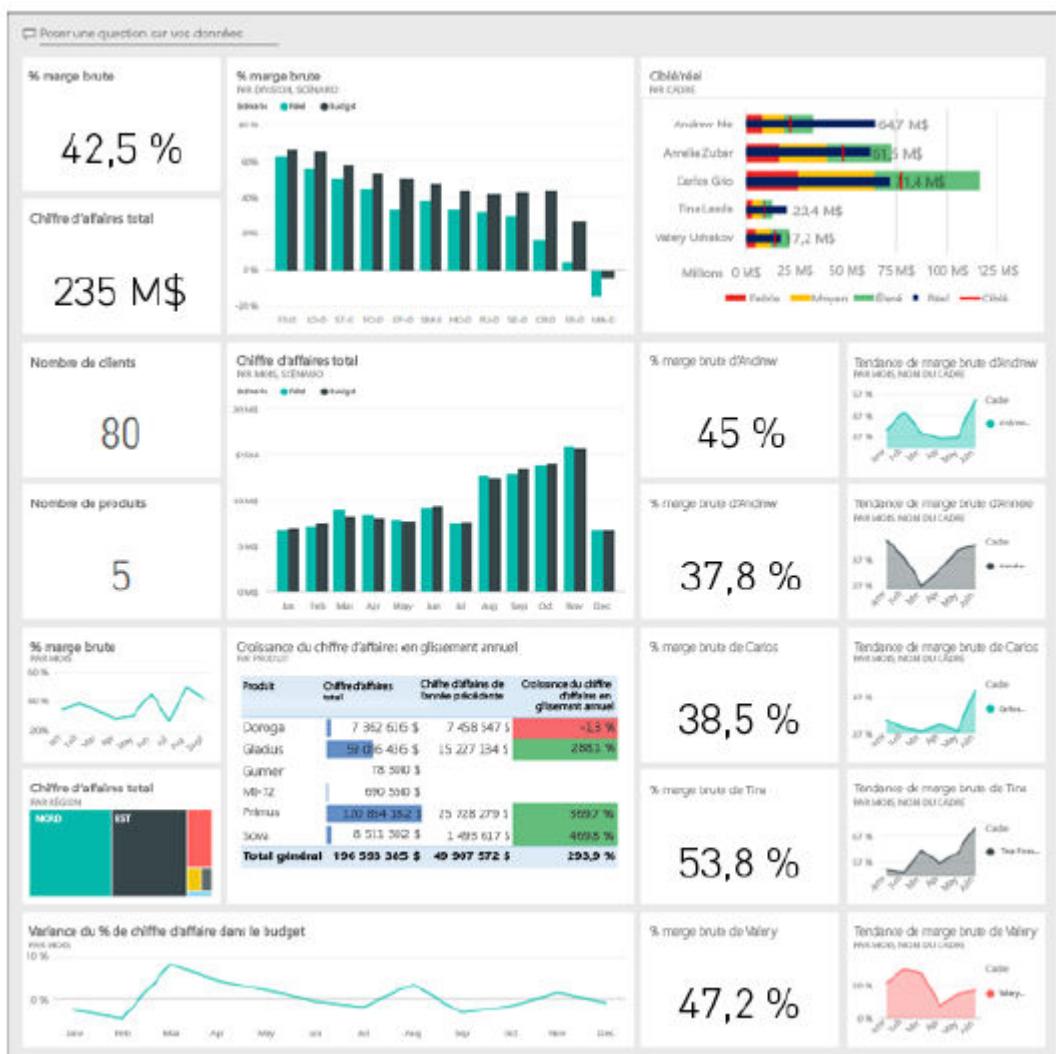


Figure : exemple de tableau de bord

Créer un tableau de bord montrant les différentes données importantes en utilisant le logiciel powerBI, et exploitant la base de données déjà créée.

Les données à présenter :

- le nombre des étudiants étrangers inscrit dans l'établissement sous forme de cercle, Les tailles des cercles sur la carte reflètent le nombre des étudiants étrangers ; à présenter dans une mapp (voir fig1)
- le nombre d'étudiant en échec par cours (note <10), à présenter sous forme de diagramme en bâtonnet (ex voir fig2)
- le nombre d'absences par cours, à présenter sous forme de diagramme en bâtonnet (ex voir fig2)
- le nombre d'auteurs par nationalité à présenter sous forme de diagramme circulaire (ex voir fig3)
- certains chiffres clés à présenter comme figure 4, comme nombre total d'inscrits, nombre total échec....

L ensemble de ces résultats doit être représenter dans un tableau de bord.



Fig 1

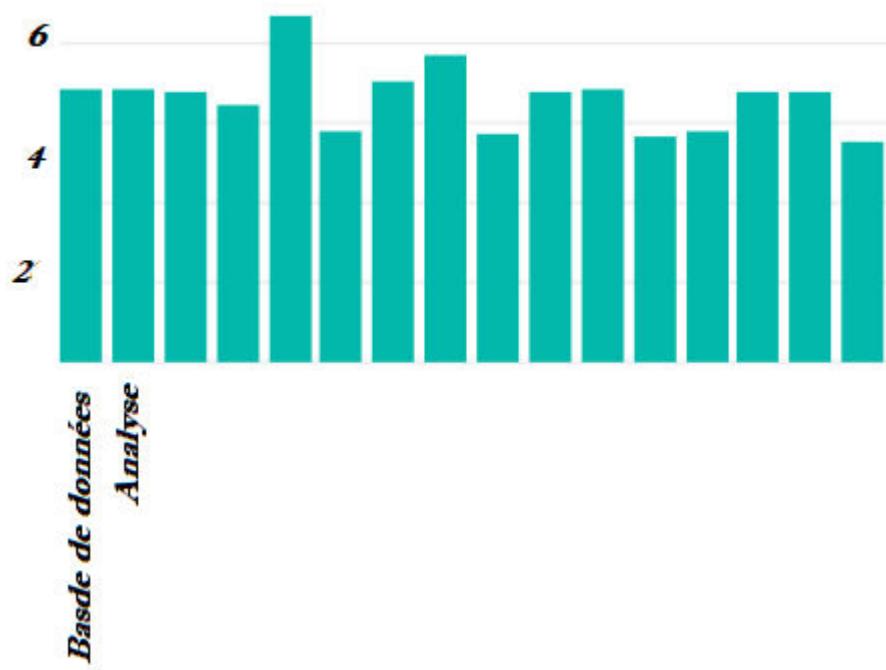


Figure 2

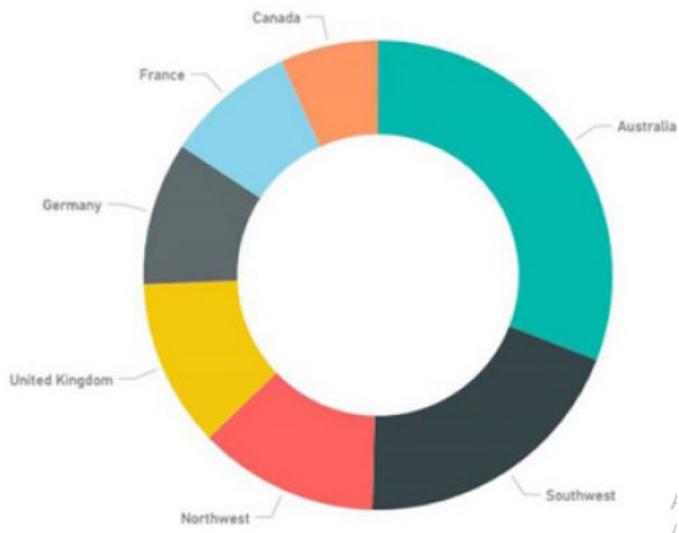


Figure 3



Figure 4