```
#Script - création d'une BD (Cours,SallesCours,Etudiants,Notes) et interrogation de la base
#Auteur: Leila Ben Othman
#Date:Février 2017
import sqlite3 as sq

"""Dans toutes les fonctions ci-dessous, je considère que curseur est une varibale globale
définie au niveau de mon script principal
On aurait pu faire un autre choix (conceptuel) en considérant curseur comme paramètre qu'on
passe à toutes les fonctions."""

def connexion_base(x):
    return (sq.connect(x))

def creation_curseur(x):
    return(x.cursor())

def validation(x):
    x.commit()

def fermeture(x):
    curseur.close()
    x.close()

#Création des tables
def creation_tables():
    curseur.execute("create table if not exists Cours(IdCours text,Jour text,Heure text, primary
key(IdCours,Jour))")
    curseur.execute("create table if not exists SallesCours(IdCours text,IdSalle text, primary key
(IdCours,IdSalle))")
    curseur.execute("create table if not exists Etudiants(IdEtudiant integer primary key,Nom
text,Prenom text,Adresse text)")
    curseur.execute("""create table if not exists Notes(IdCours text,IdEtudiant integer,Note text,
primary key(IdCours,IdEtudiant),
                    foreign key (IdCours) references Cours(IdCours), foreign key (IdEtudiant)
references Etudiant(IdEtudiant)) """)


#Insertion d'un étudiant passé sous forme d'un tuple
def insertion_Etudiant(T):
    curseur.execute('insert into Etudiants VALUES'+str(T))

#Insertion des Etudiants ligne par ligne à partir d'une liste de tuples
def insertion_Etudiants(L):
```

```
    for i in L:
        insertion_Etudiant(i)

#Insertion des Cours à partir d'une liste
def insertion_Cours(L):
    curseur.executemany("insert into Cours(IdCours,Jour,Heure) values (?,?,?)",L)

#Insertion des Salles de Cours à partir d'une liste
def insertion_SallesCours(L):
    curseur.executemany("insert into SallesCours(IdCours,IdSalle) values (?,?)",L)

#Insertion des Notes à partir d'une liste
def insertion_Notes(L):
    curseur.executemany("insert into Notes(IdCours,IdEtudiant,Note) values (?,?,?)",Lnotes)

#Fonction qui execute une requete (Recherche)
def executer_requete(requete):
    curseur.execute(requete)

#Fonctions qui affichent un curseur (contenant le resultat d'une recherche)
#verion1
def afficher_curseurV1():
    for i in curseur:
        print(i)
#version2
def afficher_curseurV2():
    while True:
        T=curseur.fetchone()

        if T!=None:
            print(T)
        else:
            break
#version3
def afficher_curseurV3():
    L=curseur.fetchall()
    for i in L:
        print(i)

"""On pourra également écrire des fonctions qui retournent la première ou la dernière ligne
d'une requête
le nombre de lignes retournés par la requête, un attribut particulier d'une lignge particulière etc,
On n'oubliera pas également les insctructions SQL qui permettent de faire de la mise à jour des
```

```
données: update, delete etc"""

#-------------------------Script principal

ma_base=connexion_base("BASE_Cours_Etud_Notes.sqlite")
curseur=creation_curseur(ma_base)
creation_tables()


E1=(100,'Toto','prenomToto', 'Tunis')
E2=(200,'Tata','prenomTata', 'La Manouba')
E3=(300,'Titi','prenomTiti', 'Kairouan')
L_etud=[]
L_etud.append(E1)
L_etud.append(E2)
L_etud.append(E3)
insertion_Etudiants(L_etud)

C1=("BD","Lundi","9H")
C2=("Algo","mardi","9H")
C3=("Algo","vendredi","9H")
C4=("Sys","mardi","14H")
L_cours=[]
L_cours.append(C1)
L_cours.append(C2)
L_cours.append(C3)
L_cours.append(C4)
insertion_Cours(L_cours)

S1=("BD","S1")
S2=("Algo","S2")
S3=("Sys","S1")
Lsalles=[]
Lsalles.append(S1)
Lsalles.append(S2)
Lsalles.append(S3)
insertion_SallesCours(Lsalles)

N1=('BD',100,'A')
N2=('BD',300,'A')
N3=('Sys',100,'B')
N4=('Sys',200,'A')
N5=('Sys',300,'B')
```

```
N6=('Algo',100,'C')
N7=('Algo',200,'A')

Lnotes=[]
Lnotes.append(N1)
Lnotes.append(N2)
Lnotes.append(N3)
Lnotes.append(N4)
Lnotes.append(N5)
Lnotes.append(N6)
Lnotes.append(N7)
insertion_Notes(Lnotes)

#on valide toutes les opérations
validation(ma_base)


#On effectue une interrogation des données

#R1 Les IdCours de tous les cours enseignés.
R1='select distinct IdCours from Cours'

#R2 Les IdEtudiant de tous les étudiants.
R2='select IdEtudiant from Etudiants'

#R3 Toutes informations relatives aux notes des étudiants qui suivent le cours 'Algo'.
R3='select * from  Notes where IdCours="Algo"'

#R4 Le jour, l'heure et la salle de chaque Cours.
R41="select * from Cours join SallesCours on Cours.IdCours=SallesCours.IdCours"
R42="select * from Cours,SallesCours where Cours.IdCours=SallesCours.IdCours"

#R5 Les inscriptions réelles
R5='select idEtudiant,IdCours from Notes'

#R6 Les étudiants inscrits à tous les cours
"""on ne peut pas la traduire directement car l'opérateur division n'existe pas en SQL
on va la retouver grâce à la partie 2 (voir R11)"""


#R7 les inscriptions possibles
R7='select idEtudiant,IdCours from Cours,Etudiants'
```

#R8 Les inscriptions manquantes: les inscriptions possibles - les inscriptions réelles
R8="select idEtudiant,IdCours from Cours,Etudiants except select idEtudiant,IdCours from Notes"

#R9 les étudiants inscrits à certains cours : les étudiants inscrits à au moins un cours
R9='select distinct idEtudiant from Notes';

#R10: les étudiants qui ne sont pas inscrits à certains cours = les étudiants qui ne sont pas inscrits
#à au moins un cours. Projection sur l'attribut idEtudiant de la relation R8
R10='select idEtudiant from (select idEtudiant,IdCours from Cours,Etudiants except select idEtudiant,IdCours from Notes)'

#R11: Les étudiants inscrits à tous les cours (R9-R10) verison1
R111="""select idEtudiant
from Notes
except
select idEtudiant from
(select idEtudiant,IdCours from Cours,Etudiants except select idEtudiant,IdCours from Notes)"""

#une deuxième façon d'exprimer la requête : les étudiants inscrits à tous les cours est la suivante:
#On sélectionne l'étudiant tel qu'il n'existe pas un cours pour lequel l'étudiant n'est pas inscrit
--version2
R112="""select idEtudiant from Etudiants
where not exists
(select * from Cours
where not exists
(select * from Notes where Notes.idCours=Cours.idCours and
Etudiants.idEtudiant=Notes.idetudiant))"""

#Les noms des étudiants qui suivent le cours ‘Algo’
R12="select Nom from Etudiants join Notes on Etudiants.IdEtudiant=Notes.IdEtudiant where IdCours='Algo'"

#Les notes en ‘BD’ des étudiants dont le nom est ‘TiTi'.
R13="select Note from Notes join Etudiants on Notes.IdEtudiant=Etudiants.IdEtudiant where idCours='BD' and nom='Titi'"

#Les couples (jour, heure) pour lesquels la salle ‘S1’ est occupée par un cours.
R14="select Jour, Heure from SallesCours join Cours on SallesCours.IdCours=Cours.IdCours where idSalle='S1'"

```
#Les identifiants des étudiants qui n’ont que des notes ‘A'
R15="select distinct idEtudiant from Notes except select distinct idEtudiant from Notes where
Note!='A'"

#Testons nos requêtes
print('-----------------resultat requete 41')
executer_requete(R41)
afficher_curseurV1()

print('----------------resultat requete 111')
executer_requete(R111)
afficher_curseurV1()

print('----------------esultat requete 112')
executer_requete(R112)
afficher_curseurV2()

print('------------resultat requete 12')
executer_requete(R12)
afficher_curseurV3()

#On ferme la base (déconnexion)
fermeture(ma_base)
```