**EXERCICE SQL\_00**

Table des matières

[Chapitre 1 QUESTION 1 : 2](#_Toc93151455)

[Chapitre 2 QUESTION 2 : 3](#_Toc93151456)

[Chapitre 3 : QUESTION 3 : 5](#_Toc93151457)

[Chapitre 4 : QUESTION 4 : 5](#_Toc93151458)

[Chapitre 5 : Question 5 : 5](#_Toc93151459)

[Chapitre 6 : QUESTION 6 : 6](#_Toc93151460)

[Chapitre 7 : QUESTION 7 : 6](#_Toc93151461)

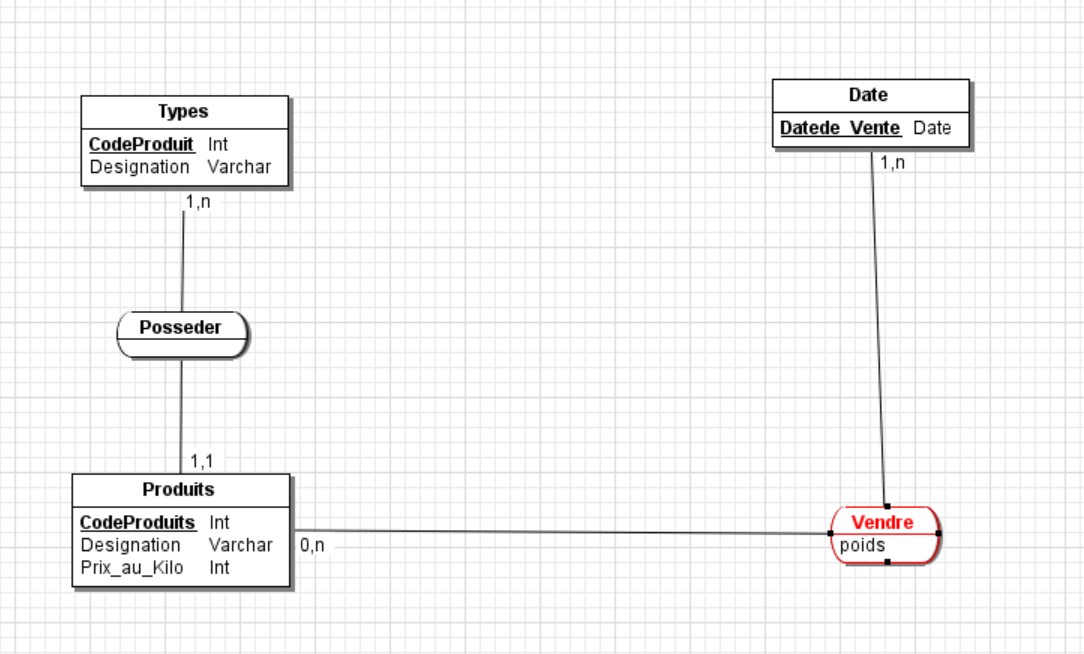
[Chapitre 8 : QUESTION 8 : 6](#_Toc93151462)

[Chapitre 9 : QUESTION 9 : 7](#_Toc93151463)

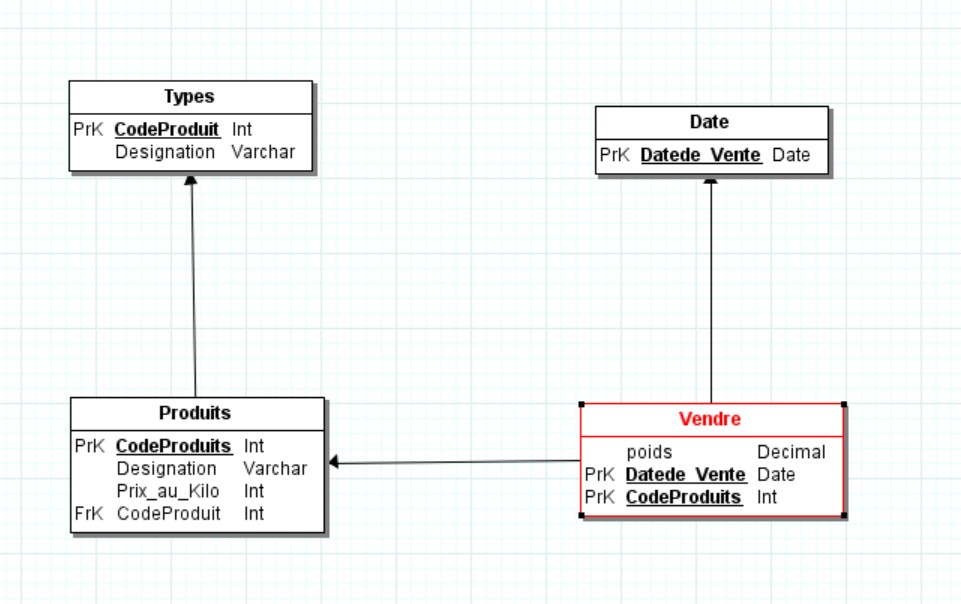
# 

# QUESTION 1 :

Avec Jmerise on met en place le modèle conceptuel (MCD) suivant :



A partir du modèle MCD on obtient le modèle Logique (MLD) suivant :



L’objectif donc dans Sqlite Spy est de créer physiquement (MPD)cette base de données relationnelles :

Créer 4 tables :

3 tables de référence : Date, Types et Produits avec chacune sa clé primaire (PRk)

1 table de fait : Vendre qui contient 2 clés étrangères qui sont des clés primaires venues des tables qu’elle a reliées (Date, Produits)

Donc on s’en servira du langage SQL pour écrire le code qui sert à intégrer ce modèle dans le SQLite :



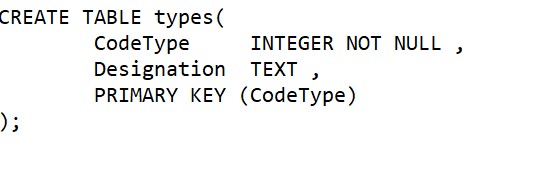
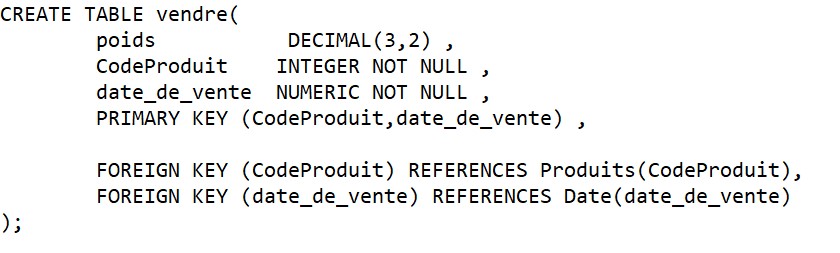


Fig 1 fig 2



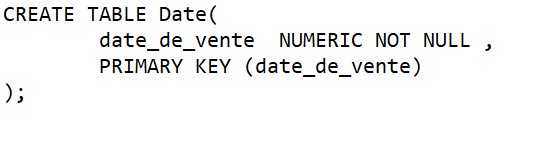


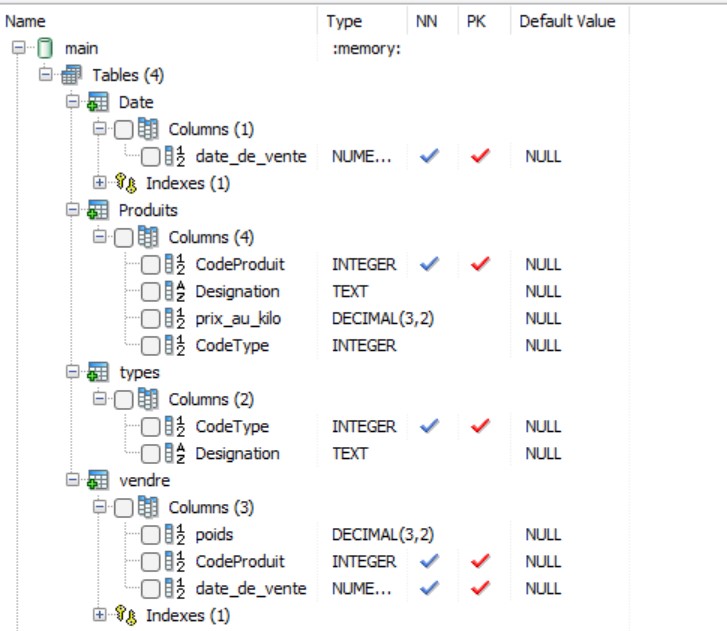
Fig 3 fig 4

Les figures 1, 2, 3 et 4 nous indiquent le code qu’il faut utiliser pour la création de chaque table dans SQLITE Spy.

On met en évidence la présence de clés étrangères dans les tables Types et Vendre, pour activer ces Clé étrangères il faut mettre en place le code suivant avant la création des tables :



Le résultat de la création est sur la figure suivante :



# QUESTION 2 :

Pour alimenter les différentes tables il faut rentrer les différentes valeurs de chaque table comme mentionné dans les figures suivantes



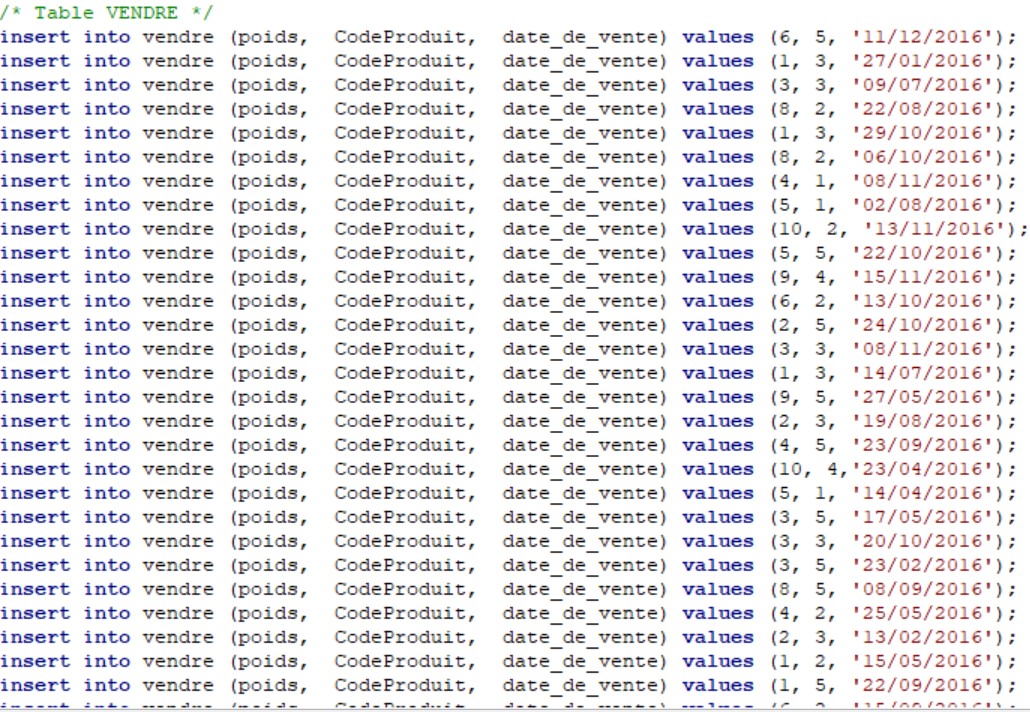
1. La table « Types » :
2. La table « Produits » :



1. La table Date :



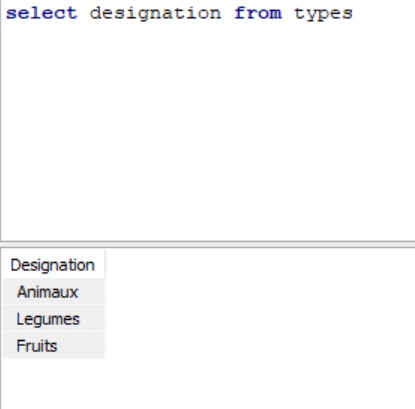
1. Table Vendre :



Il faut noter que l’ordre de création de ces tables est un ordre logique en tenant en compte des dépendances fonctionnelles du modèle MLD par exemple la table Venre on la crée la dernière car c’est une table de fait.

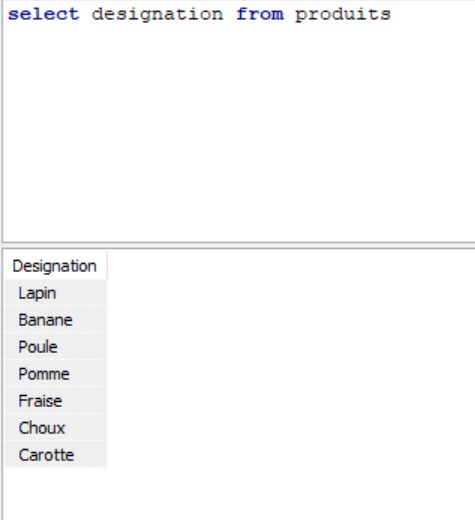
# : QUESTION 3 :

Afficher la colonne Designation dans la table Types :



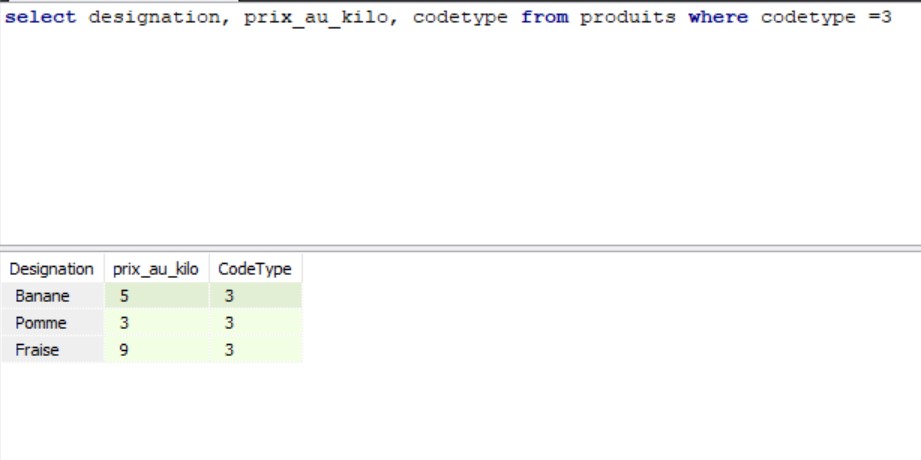
# : QUESTION 4 :

La liste de la colonne Designation dans la table Produits



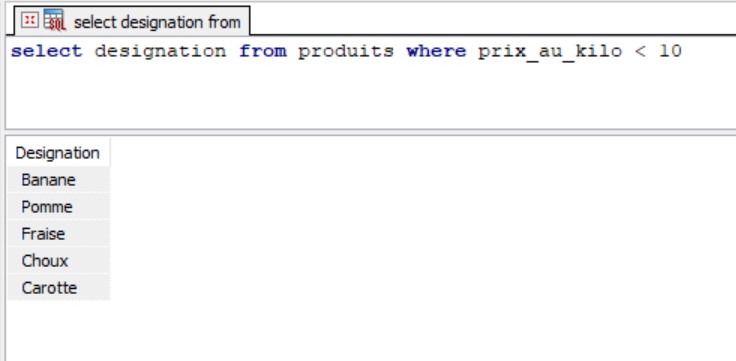
# : Question 5 :

Afficher quand la colonne codetype = 3 toutes les colonnes (sauf codeproduit)de la table produits



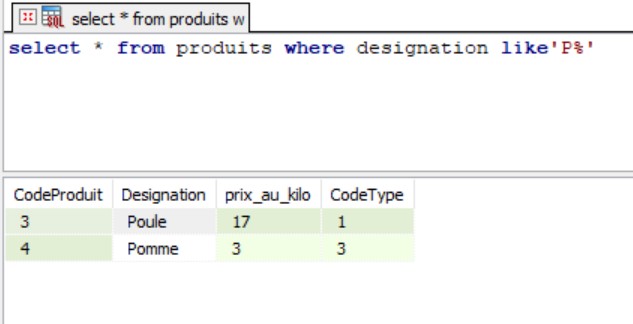
# : QUESTION 6 :

Afficher les valeurs de la colonne designation de la table produite dont quand les valeurs de la colonne prix\_au\_kilo < 10



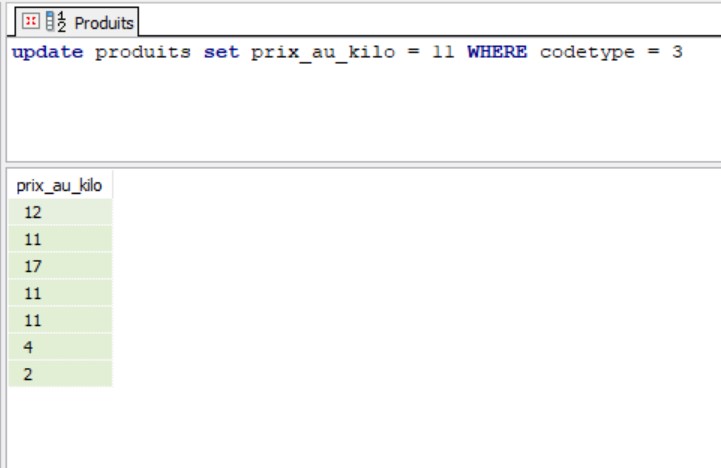
# : QUESTION 7 :

Afficher la colonne designation dans la table produits si les noms de la colonnes commencent par « P »



# : QUESTION 8 :

Changer dans la colonne prix\_au\_kilo la valeur quand la colonne codetype=3 (passera de 9 à 11=



# : QUESTION 9 :

Suppression dans la table produit toutes les lignes correspondant à des valeurs 1 et 2 de la colonne Codetype :

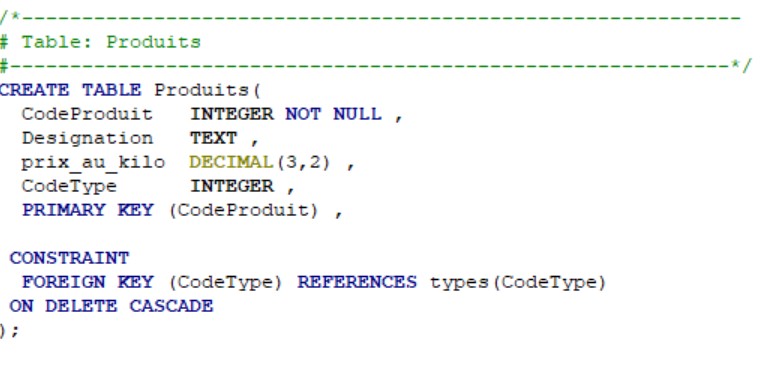
En fait pour pouvoir supprimer ces deux lignes il faut d’abord supprimer les contraintes de clés étrangères dans la tables produit, car la colonne CodeType est une clé étrangère sinon le SGBD ne nous laissera pas y parvenir car ces contraintes protègent l’intégrité des données.

Pour ce faire il faut désactiver PRAGMA foreign\_keys = OF

Puis supprimer l’effet cascade comme dans les figues :

J’ai réussi à faire ça en supprimant complètement la base de données et recréé une nouvelle avec un code différent concernant la table produits :





Puis on applique le code suivant pour supprimer les deux lignes :

