Exercice 2 (6 points)

Cet exercice porte sur les bases de données et la programmation Python et la gestion de bugs.

Dans cet exercice, on pourra utiliser les clauses du langage SQL pour :

- construire des requêtes d'interrogation à l'aide de `SELECT`, `FROM`, `WHERE` (avec les opérateurs logiques `AND`, `OR`) et `JOIN ... ON` ;
- construire des requêtes d'insertion et de mise à jour à l'aide de `UPDATE`, `INSERT` et `DELETE` ;
- affiner les recherches à l'aide de `DISTINCT` et `ORDER BY`.

Un magasin de bricolage utilise une base de données constituée de quatre tables dont voici des *extraits*.

client				
ref_client	nom	prenom	email	telephone
25123	Renaud	Martine	renaudm@tmail.com	0601020304
25137	Dupont	Jacques	dj@mail.fr	0604030201
25145	Pasteur	Emile	pasteur0@dmail.fr	0611121314
25149	Eiffel	Franck	eiffel95@popmail.fr	0614131211
25189	Kanek	Elise	ekanek@mail.fr	0600112233
25322	Shar	Sofia	shs@fmail.fr	0644332211
…	…	…	…	…

produit			
ref_produit	designation	type	prix_unitaire
85235	Marteau TAP	Outillage	15.89
86782	Rouleau peinture	Outillage	9.55
89363	Niveau à bulle	Outillage	8.2
89552	Clous inox	Visserie	4.5
89588	Sac sable	Materiau	11.6
…	…	…	…

remise				
ref_remise	designation	valeur	date_debut	date_fin
289	Client en or	25.0	2025/01/01	2025/12/31
326	Fin de serie	40.0	2025/01/01	2025/12/31
275	Jour fou	30.0	2025/03/17	2025/03/19
263	Soldes hiver	20.0	2025/01/01	2025/02/01
…	…	…	…	…

vente					
ref_vente	date	ref_produit	ref_client	quantité	ref_remise
25631	2025/03/16	86782	25123	2	289
25632	2025/03/16	89363	25123	1	289
25633	2025/03/17	85235	25149	1	326
25634	2025/03/18	89588	25145	5	275
…	…	…	…	…	…

- L'attribut `ref_client` est une clé primaire de la table `client` ;
- l'attribut `ref_produit` est une clé primaire de la table `produit` ;
- l'attribut `ref_remise` est une clé primaire de la table `remise` ;
- l'attribut `ref_vente` est une clé primaire de la table `vente` ;
- dans la table `remise`, l'attribut `valeur` correspond au taux de remise appliqué, exprimé en pourcentage.

1. Expliquer le rôle d'une clé primaire et rappeler la contrainte dans le choix de celle-ci.

Dans la table `vente`, les attributs `ref_produit` et `ref_client` sont des clés étrangères qui référencent respectivement les attributs `ref_produit` de la table `produit` et `ref_client` de la table `client`.

2. À l'aide des extraits de tables, détailler un achat d'Emile Pasteur en précisant :
   - la date de son achat ;
   - le ou les article(s) acheté(s) ;
   - si c'est le cas, le taux de remise dont il a bénéficié.

Un nouveau client doit être entré dans la base, il s'agit de Gilles Bertaut, son email est `gbertaut@fmail.fr` et son numéro de téléphone est `0641424344`. Son identifiant (`ref_client`) dans la base sera `25345`.

3. Écrire une requête `SQL` permettant cet ajout.

Une correction est à apporter dans la base, l'email de la cliente Shar Sofia n'a pas été correctement saisi, voici son email correct : 'shars@fmail.fr'.

4. Écrire une requête `SQL` permettant cette mise à jour.

Dans la base de données, toutes les dates sont de type chaîne de caractères au format `'aaaa/mm/jj'`. Cela permet de comparer des dates entre elles : par exemple l'opération `'2025/04/12' < '2025/05/03'` est vraie puisque la date du 12 avril 2025 est antérieure à celle du 3 mai 2025.

5. Écrire une requête qui permet d'afficher les attributs `ref_client` de tous les clients ayant fait un achat à partir du 1er janvier 2025. On souhaite qu'un même client n'apparaisse qu'une seule fois dans cet affichage.
6. La tondeuse de référence 90222 vendue dans le magasin a un défaut de fabrication. Le responsable doit rappeler tous les clients qui ont acheté cette tondeuse depuis le 15 septembre 2024, date de mise en stock des tondeuses défectueuses. Écrire une requête qui permet d'obtenir le nom et le numéro de téléphone des clients à rappeler.

Dans cette partie on suppose que du code Python associé aux requêtes `SQL` est exécuté pour réaliser l'objectif souhaité. Pour cela, à chaque table de la base de données est associé un dictionnaire Python qui porte le nom de la table.

Le dictionnaire contient les éléments suivants : `cle_primaire : [attribut_1, attribut_2, ...]`. L'ordre des attributs est identique à celui des extraits de table présentés en début d'exercice.

Par exemple, pour la table `client`, le dictionnaire associé est le suivant :

```
Client={25123:['Renaud','Martine','renaudm@tmail.com','0601020304'],
        25137:['Dupont', 'Jacques', 'dj@mail.fr','0604030201'],
        25145:['Pasteur','Emile','pasteur0@dmail.fr','0611121314'],
        25149:['Eiffel','Franck','eiffel95@popmail.fr'] }
```

```
vente = {25631 : ['2025/03/16', 86782, 25123, 2, 289],
         25632 : ['2025/03/16', 89363, 25123, 1, 289],
         25633 : ['2025/03/17', 85235, 25149, 1, 326],
         25634 : ['2025/03/18', 89588, 25145, 5, 275]}
```

On considère la fonction Python `select_tel` ci-dessous. Cette fonction est associée à une requête SQL demandant le numéro de téléphone d'un client connaissant son attribut `ref_client`. Elle prend en paramètres :

- `client`, un dictionnaire associé à la table `client` ;
- `ref_client`, un entier correspondant à l'attribut `ref_client` d'un client.

Ainsi l'instruction Python `select_tel(client, 25137)` est associée à la requête SQL : `SELECT telephone FROM client WHERE ref_client = 25137;`

```
1 def select_tel(client, ref_client):
2     return client[...][...]
```

7. Recopier et compléter la ligne 2 du code de la fonction `select_tel`.

8. On considère le code de la fonction `select_tel` complétée. Ainsi l'exécution de l'instruction `select_tel(client, 25137)` renvoie le bon résultat `'0604030201'`. Mais l'exécution de l'instruction `select_tel(client, 1234)` provoque l'erreur ci-dessous :

   ```
   KeyError: 1234
   ```

   Expliquer ce qui, concernant le dictionnaire `client`, est à l'origine de cette erreur et à quelle situation pour le magasin cela correspond.

9. Recopier le code de la fonction `select_tel` en proposant une modification pour que la fonction renvoie `None` plutôt que de provoquer une `KeyError` dans le cas où la situation précédente se présente.

On souhaite écrire le code de la fonction `nb_produits` qui prend comme paramètres :

- `vente`, un dictionnaire associé à la table `vente` ;
- `ref_produit`, un entier associé à l'attribut `ref_produit` du dictionnaire à la table `vente`.

Cette fonction renvoie alors le nombre total de produits de cette référence vendus.

Par exemple pour connaître le nombre de « Marteau TAP » vendu, on écrit l'instruction : `nb_produits (vente, 85235)` où 85235 est le numéro identifiant le produit « Marteau TAP » et `vente` est le dictionnaire associé à la table `vente`.

10. Écrire le code en Python de la fonction `nb_produits`.

11. Dans cette question on considère que les tables de la base de données contiennent **exactement** ce qui est présenté dans les extraits en début d'exercice. On exécute la requête SQL : `DELETE FROM produit WHERE ref_produit = 89363;`. Celle-ci n'est pas exécutée et on obtient le message d'erreur `foreign key constraint failed`. Expliquer pourquoi il est nécessaire que la requête demandée ne soit pas exécutée et qu'elle affiche ce message d'erreur.

On considère le code Python de la fonction associée `delete_prod` qui permet de supprimer un produit dont la référence est précisée. Cette fonction prend en paramètres :

- `produit`, un dictionnaire associé à la table `produit` ;
- `ref_produit`, un entier correspondant à un attribut `ref_produit` de la table `produit`.

```
1 def delete_prod(produit, vente, ref_produit):
2     del produit[ref_produit]
```

12. Réécrire le code de la fonction `delete_prod` en faisant toutes les modifications et ajouts nécessaires pour qu'à son appel, elle refuse la suppression du produit lorsqu'on rencontre la situation présentée à la question précédente.