

## Informatique S4 : Bases de Données

### Projet pratique N°2

#### ÉTAPE 1 : Structuration des données, lien avec Python

- i. Téléchargement d'un data set Kaggle
- ii. Rétro conception

1. La technique de reconnaissance faciale triplet est une méthode de reconnaissance faciale qui utilise un réseau de neurones pour apprendre à représenter les visages en utilisant des vecteurs de caractéristiques. Le réseau est entraîné à minimiser la distance entre les vecteurs de caractéristiques de visages similaires et à maximiser la distance entre les vecteurs de caractéristiques de visages différents. Pour construire une base de données de reconnaissance faciale en utilisant la technique triplet, nous devons suivre les étapes suivantes :

Collecte de données (1) : Il faut collecter des données de visages pour former notre modèle de reconnaissance faciale. Les données doivent être variées et représentatives de la population cible. Il est possible d'utiliser des sources de données publiques ou de collecter des données à partir d'une caméra.

Prétraitement des données (2) : Les images de visages doivent être prétraitées avant d'être utilisées pour entraîner le modèle. Cela peut inclure le redimensionnement, le recadrage, la normalisation des couleurs, la suppression des arrière-plans et la correction des distorsions.

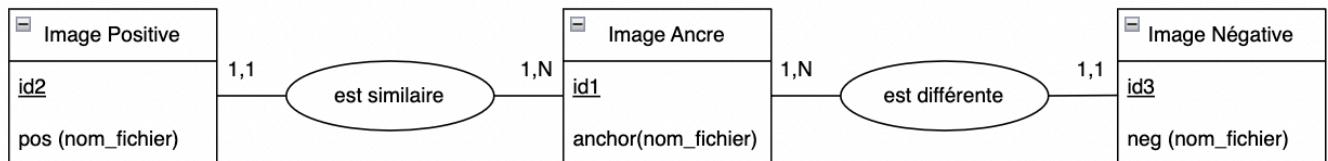
Extraction de caractéristiques (3) : Il faut extraire les caractéristiques des images de visages en utilisant un réseau de neurones pré-entraîné. Le réseau est utilisé pour extraire un vecteur de caractéristiques pour chaque image de visage. Les vecteurs de caractéristiques sont des représentations numériques des images de visages.

Entraînement du modèle (4) : Il faut entraîner le modèle à partir des triplets de visages. Chaque triplet se compose de trois images : une image ancre, une image positive et une image négative. L'image ancre est une image de visage pour laquelle il faut trouver une correspondance. L'image positive est une image de visage similaire à l'image ancre. L'image négative est une image de visage différente de l'image ancre. Le réseau de neurones doit apprendre à minimiser la distance entre les vecteurs de caractéristiques de l'image ancre et de l'image positive, tout en maximisant la distance entre les vecteurs de caractéristiques de l'image ancre et de l'image négative.

Validation du modèle (5) : Pour finir, il faut tester le modèle en utilisant des données de test pour évaluer sa précision et son efficacité.

Une fois que le modèle de reconnaissance faciale a été entraîné en utilisant la technique triplet, il est possible de l'utiliser pour reconnaître des visages en temps réel. Il est également possible d'intégrer le modèle dans une application ou un système de sécurité pour l'identification de personnes ou la surveillance de zones à accès restreint.

On en déduit le modèle EA suivant :



La cardinalité entre les entités est déterminée par la nature des relations entre elles dans le modèle. Dans ce cas, chaque triplet se compose d'une image ancre, d'une image positive et d'une image négative, et ces images sont liées par des relations spécifiques :

- Chaque triplet doit inclure une image ancre pour trouver une correspondance.
- Chaque triplet doit inclure une image positive, qui est une image similaire à l'image ancre.
- Chaque triplet doit inclure une image négative, qui est une image différente de l'image ancre.

En effet, il est possible d'associer plusieurs images positives et négatives à une seule image ancre, car plusieurs images similaires ou différentes peuvent être utilisées pour trouver une correspondance. En revanche, chaque image positive ou négative ne peut être associée qu'à une seule image ancre, car chaque image ancre a une seule référence correspondante pour trouver une correspondance. Ainsi, on a une relation "un-à-plusieurs" entre l'image ancre et les images positives et négatives, et une relation "un-à-un" entre chaque image positive ou négative et son image ancre correspondante.

## 2. Relations :

Ancre (id1, anchor)

Positive (id2, pos)

Negative (id3, neg)

Schéma relationnel :

Ancre (id1, anchor)

Positive (id2, #id1, pos)

Negative (id3, #id1, neg)

3. Oui, ce modèle est en 3FN (troisième forme normale). La troisième forme normale (3FN) est également respectée car il n'y a pas de dépendances transitives entre les attributs. Chaque attribut dépend directement de la clé primaire de sa relation respective, et il n'y a pas d'attributs non-clés qui dépendent d'autres attributs non-clés. Donc, le modèle est en 3FN et satisfait les exigences de la modélisation relationnelle des bases de données.
4. Voici le code SQL (SQLite sur VSC) qui nous a permis de créer le modèle physique :

The screenshot shows the Visual Studio Code interface with the SQLite Explorer and SQL Editor panes open. The SQL Editor pane contains the following SQL code:

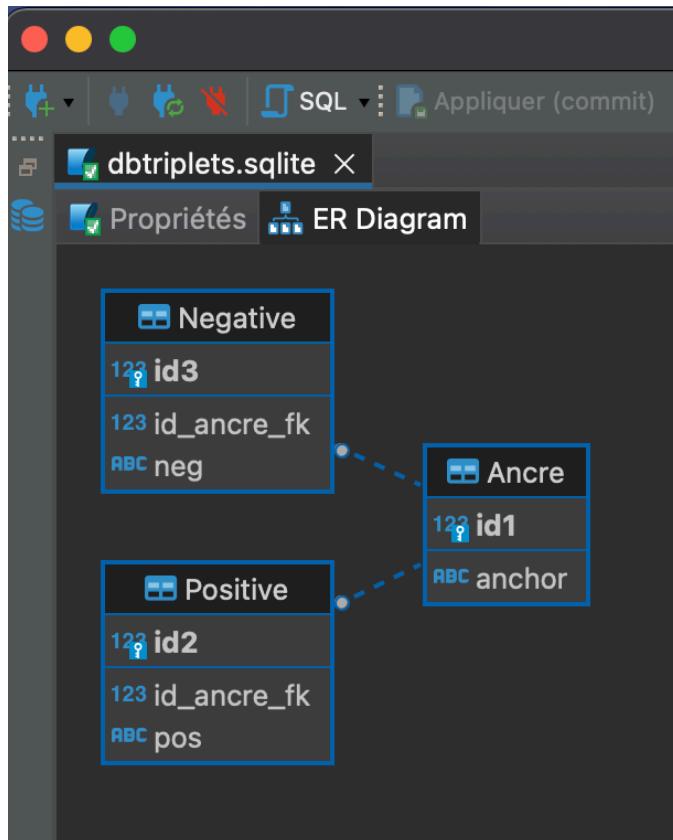
```

-- SQLite Untitled-1
1 -- SQLite
2 CREATE TABLE Ancre (
3     id1 INTEGER PRIMARY KEY,
4     anchor VARCHAR(255) NOT NULL
5 );
6
7 CREATE TABLE Positive (
8     id2 INTEGER PRIMARY KEY,
9     id_ancre_fk INTEGER NOT NULL,
10    pos VARCHAR(255) NOT NULL,
11    FOREIGN KEY (id_ancre_fk) REFERENCES Ancre(id1)
12 );
13
14 CREATE TABLE Negative (
15     id3 INTEGER PRIMARY KEY,
16     id_ancre_fk INTEGER NOT NULL,
17     neg VARCHAR(255) NOT NULL,
18     FOREIGN KEY (id_ancre_fk) REFERENCES Ancre(id1)
19 );
20
21

```

The SQLite Explorer pane shows the database structure with three tables: Ancre, Positive, and Negative. The Ancre table has columns id1 (integer, primary key) and anchor (varchar(255), not null). The Positive table has columns id2 (integer, primary key), id\_ancre\_fk (integer, foreign key referencing Ancre.id1), and pos (varchar(255), not null). The Negative table has columns id3 (integer, primary key), id\_ancre\_fk (integer, foreign key referencing Ancre.id1), and neg (varchar(255), not null).

Voici le modèle physique :



### iii. Data Analyse : Faites un lien avec EXCEL

1. Nous avons traité les données avec la fonction NB.SI, fonction statistique qui permet de compter le nombre de cellules qui correspondent à un critère à déterminer.
  - Unicité Id3 : Est-ce que l'id3 de la cellule concernée est unique ou non ?
  - Double Id3 : Est-ce que l'id3 de la cellule concernée est en double ou non ?
  - Nombre d'occurrence Id3 : Comment de fois apparaît l'identifiant dans la table ?

	A	B	C	D	E	F	G	H	I	J
1	anchor	id1	pos	id2	neg	id3		Unicité Id3	Doublon Id3	Nombre d'occurrence Id3
2	056279.jpg	1	108998.jpg	1	030848.jpg	496		FAUX	VRAI	2
3	024091.jpg	1	000023.jpg	1	093653.jpg	9313		VRAI	FAUX	1
4	122082.jpg	3	045833.jpg	3	188283.jpg	7200		VRAI	FAUX	1
5	110393.jpg	3	021233.jpg	3	178433.jpg	4643		VRAI	FAUX	1
6	101388.jpg	4	056784.jpg	4	105432.jpg	2988		VRAI	FAUX	1
7	143743.jpg	4	107918.jpg	4	079773.jpg	5029		VRAI	FAUX	1
8	093187.jpg	5	101049.jpg	5	141930.jpg	52		VRAI	FAUX	1
9	018771.jpg	5	093951.jpg	5	151909.jpg	6247		VRAI	FAUX	1
10	122677.jpg	6	003289.jpg	6	165740.jpg	4441		VRAI	FAUX	1
11	147869.jpg	6	122930.jpg	6	103819.jpg	6147		FAUX	VRAI	2
12	140464.jpg	7	091155.jpg	7	168229.jpg	620		VRAI	FAUX	1
13	020284.jpg	7	003099.jpg	7	111670.jpg	2306		VRAI	FAUX	1
14	108789.jpg	8	100773.jpg	8	073767.jpg	9544		FAUX	VRAI	2
15	094456.jpg	8	137740.jpg	8	094731.jpg	2060		VRAI	FAUX	1
16	038345.jpg	10	010990.jpg	10	073824.jpg	6118		VRAI	FAUX	1
17	090423.jpg	10	150519.jpg	10	117513.jpg	1035		FAUX	VRAI	3
18	028512.jpg	12	094568.jpg	12	189375.jpg	6814		VRAI	FAUX	1
19	046666.jpg	12	028858.jpg	12	023205.jpg	1705		VRAI	FAUX	1

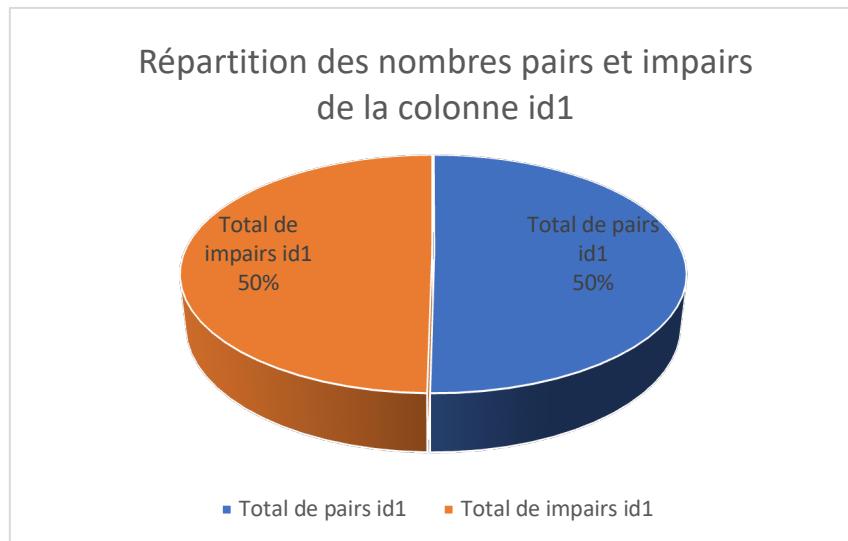
- La fonction SOMMEPROD, fonction matricielle utilise des cellules pour effectuer une multiplication matricielle, puis additionne le résultat pour donner une seule valeur. Ici nous avons cherché à avoir la somme des nombres pairs et impairs parmi les identifiants (id1) de notre Dataset.

Total de pairs id1	Total de impairs id1
394	390

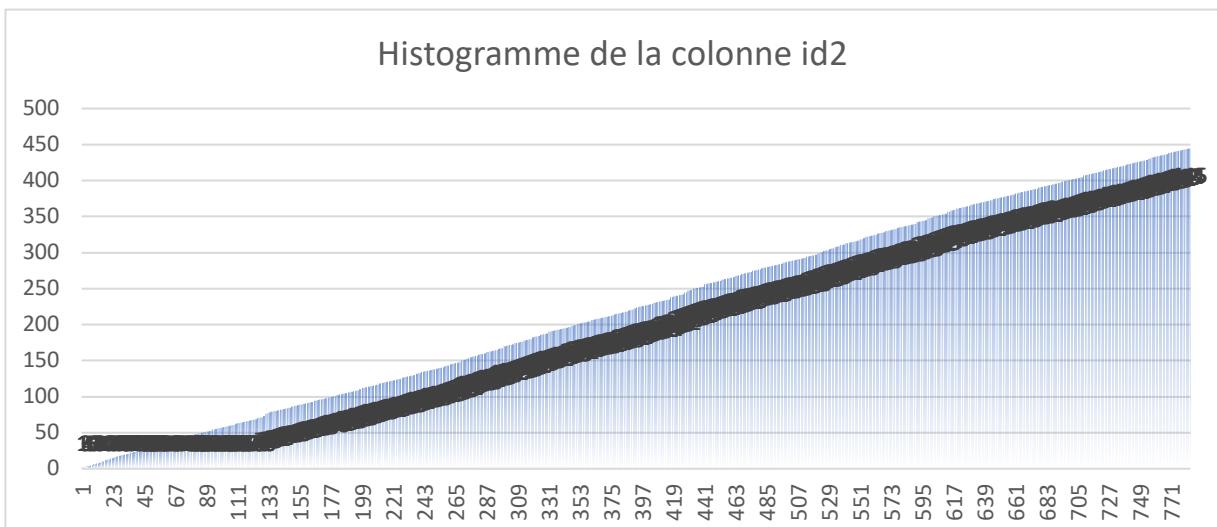
- Le tableau croisé dynamique ci-dessous compte le nombre d'enregistrement parmi les attributs pos, neg et anchor, on constate donc qu'aucune valeur est nulle parmi les fichiers images de notre Dataset.

Tableaux croisés dynamiques		
Nombre de anchor	Nombre de pos	Nombre de neg
784	784	784

2. Voici un camembert :



Voici un histogramme :



## iv. Faites un lien avec Python :

1. Voici le programme python qui nous a permis de créer une 4ème table et de dans notre DataBase Revonnaissance\_faciale.

```

import mysql.connector

# Connexion à la base de données
mydb = mysql.connector.connect(
  host="localhost",
  user="hadi",
  password="password",
  database="Reconnaissance_faciale"
)

# Création de la première table
mycursor = mydb.cursor()
mycursor.execute("CREATE TABLE Image_ancre (id INT AUTO_INCREMENT PRIMARY KEY, nom_du_fichier VARCHAR(255))")

# Insertion des données dans la première table
mycursor.execute("INSERT INTO Image_ancre (nom_du_fichier) VALUES ('image_ancre1.jpg')")
mycursor.execute("INSERT INTO Image_ancre (nom_du_fichier) VALUES ('image_ancre2.jpg')")

# Création de la deuxième table
mycursor.execute("CREATE TABLE Image_positive (id INT AUTO_INCREMENT PRIMARY KEY, nom_du_fichier VARCHAR(255))")

# Insertion des données dans la deuxième table
mycursor.execute("INSERT INTO Image_positive (nom_du_fichier) VALUES ('image_positive1.jpg')")
mycursor.execute("INSERT INTO Image_positive (nom_du_fichier) VALUES ('image_positive2.jpg')")

# Création de la troisième table
mycursor.execute("CREATE TABLE Image_negative (id INT AUTO_INCREMENT PRIMARY KEY, nom_du_fichier VARCHAR(255))")

# Insertion des données dans la troisième table
mycursor.execute("INSERT INTO Image_negative (nom_du_fichier) VALUES ('image_negative1.jpg')")
mycursor.execute("INSERT INTO Image_negative (nom_du_fichier) VALUES ('image_negative2.jpg')")

# Création de la quatrième table
mycursor.execute("CREATE TABLE Nouvelle_table (id INT AUTO_INCREMENT PRIMARY KEY, nom_du_fichier VARCHAR(255), type VARCHAR(10))")

# Insertion des données dans la quatrième table
mycursor.execute("SELECT nom_du_fichier FROM Image_ancre")
rows = mycursor.fetchall()
for row in rows:
    nom_du_fichier = row[0]
    mycursor.execute("INSERT INTO Nouvelle_table (nom_du_fichier, type) VALUES (%s, %s)", (nom_du_fichier, "ancre"))

mycursor.execute("SELECT nom_du_fichier FROM Image_positive")
rows = mycursor.fetchall()
for row in rows:
    nom_du_fichier = row[0]
    mycursor.execute("INSERT INTO Nouvelle_table (nom_du_fichier, type) VALUES (%s, %s)", (nom_du_fichier, "positive"))

mycursor.execute("SELECT nom_du_fichier FROM Image_negative")
rows = mycursor.fetchall()
for row in rows:
    nom_du_fichier = row[0]
    mycursor.execute("INSERT INTO Nouvelle_table (nom_du_fichier, type) VALUES (%s, %s)", (nom_du_fichier, "negative"))

# Valider les changements
mydb.commit()

```

Ln: 5 Col: 0

Nous avons utilisé la fonction commit vu durant les exposés des projets précédents.

Voici le résultat de ce code :

```

hadi — mysql -u hadi -p — 73x21
MariaDB [Reconnaissance_faciale]> SHOW TABLES;
+-----+
| Tables_in_Reconnaissance_faciale |
+-----+
| Image_ancre
| Image_negative
| Image_positive
| Nouvelle_table
+-----+
[4 rows in set (0,020 sec)]]

MariaDB [Reconnaissance_faciale]> SELECT * FROM Image_ancre;
+----+-----+
| id | nom_du_fichier |
+----+-----+
| 1  | image_ancre1.jpg |
| 2  | image_ancre2.jpg |
+----+-----+
2 rows in set (0,034 sec)

MariaDB [Reconnaissance_faciale]>

```

2. Voici le code qui nous a permis de supprimer un enregistrement :

```

supprimeer.py - /Users/hadi/supprimeer.py (3.11.3)
import mysql.connector

# Connexion à la base de données
mydb = mysql.connector.connect(
    host="localhost",
    user="hadi",
    password="password",
    database="Reconnaissance_faciale"
)

# Créer un curseur
mycursor = mydb.cursor()

# Supprimer un enregistrement
sql = "DELETE FROM Nouvelle_table WHERE nom_du_fichier = %s"
val = ("image_ancre1.jpg",) # spécifier la valeur à supprimer
mycursor.execute(sql, val)

# Valider les changements
mydb.commit()

# Afficher le nombre de lignes supprimées
print(mycursor.rowcount, "enregistrement(s) supprimé(s)")

# Fermer la connexion
mydb.close()

```

Ln: 23 Col: 51

Voici le résultat :

```

>>> 
=====
1 enregistrement(s) supprimé(s)
| 
Ln: 35 Col: 0

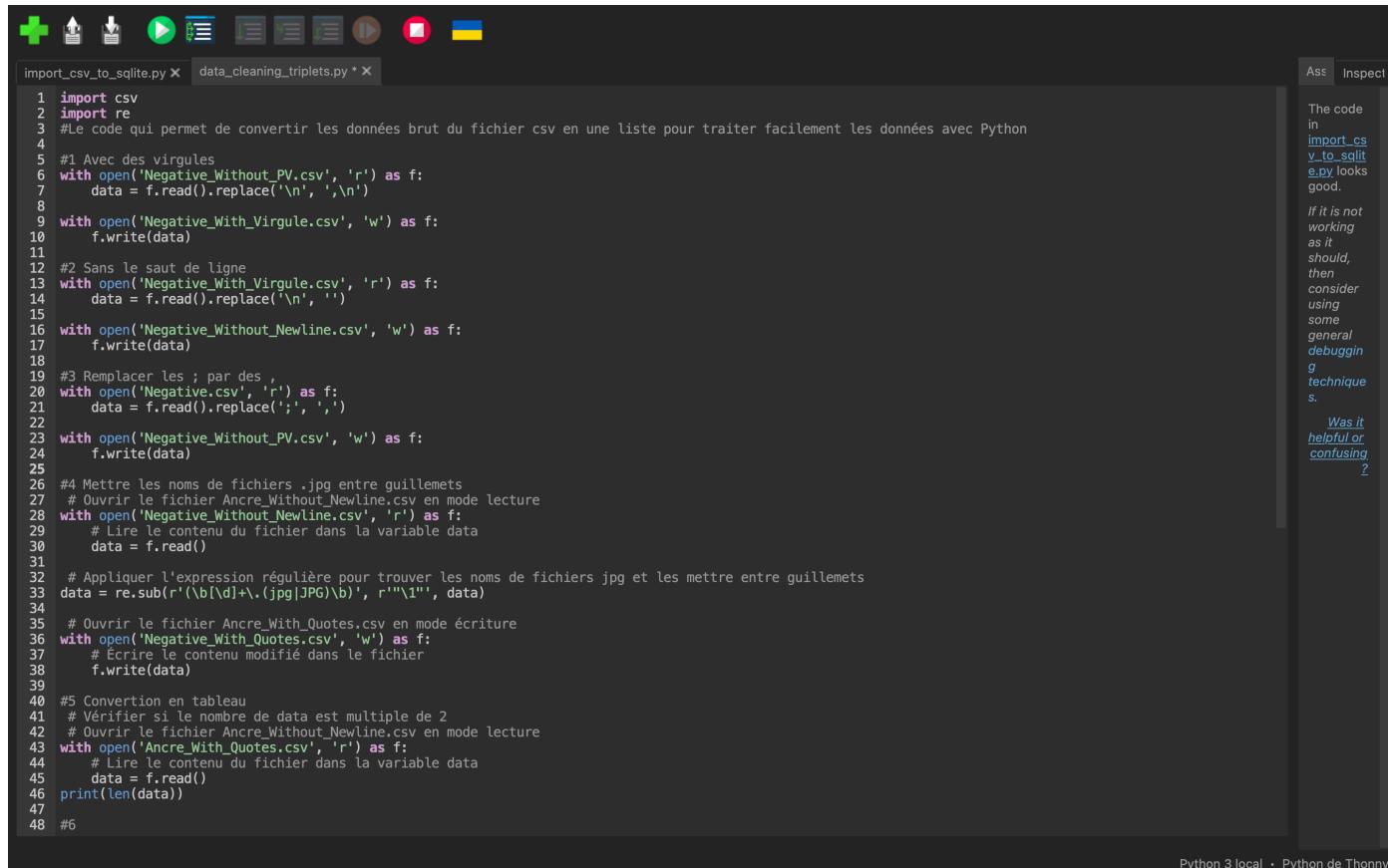
```

3. Afin de garantir les contraintes de la base de données, on utilise les contraintes NOT NULL, et UNIQUE.

## ÉTAPE 2 : SQL

1,2 et 3. Ajout les données Kaggle dans la base.

Nous avons effectué un long travail de DataCleaning grâce à Python pour importer les données Kaggle dans la base.

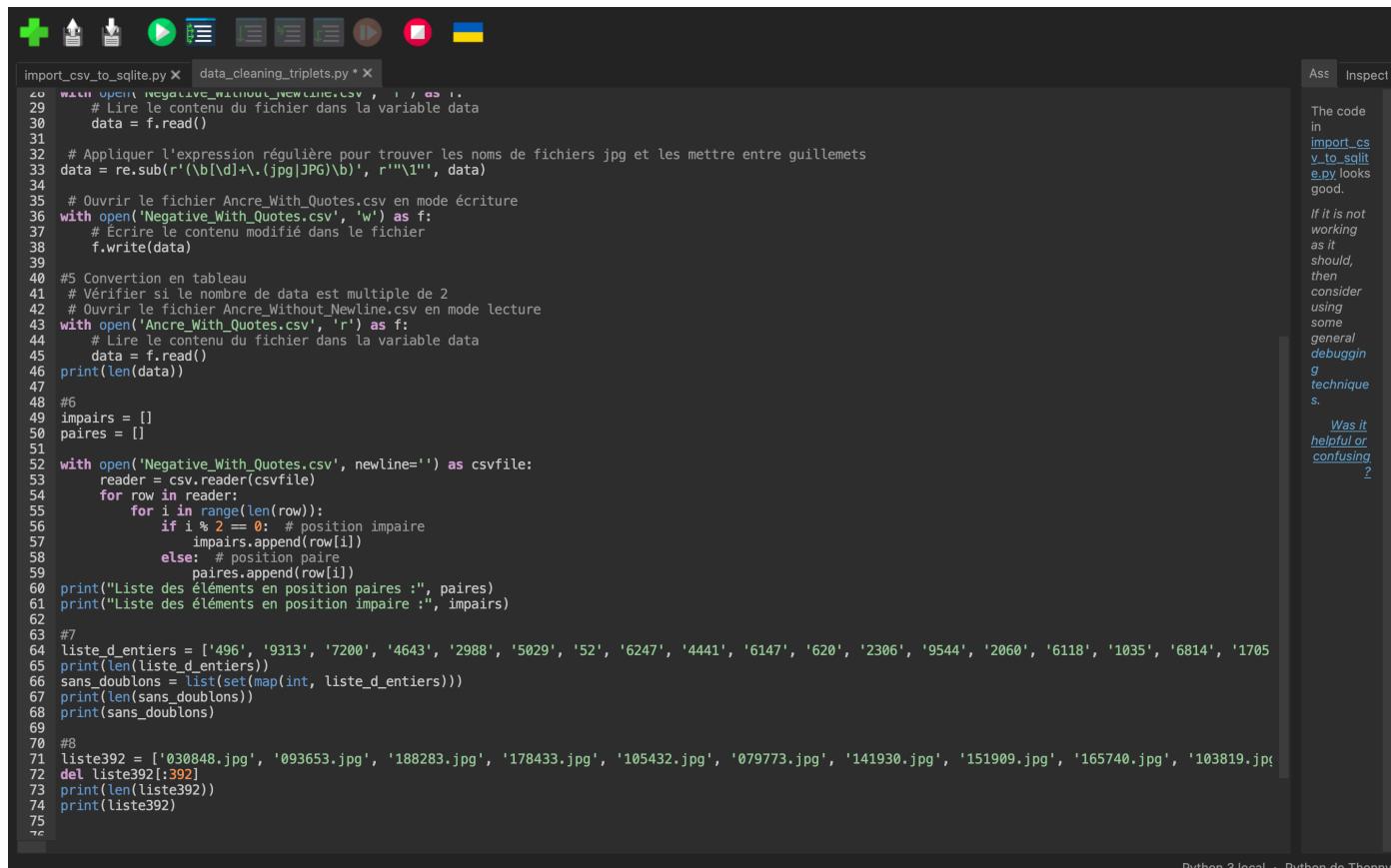


```

import_csv_to_sqlite.py *  data_cleaning_triplets.py * 
1 import csv
2 import re
3 #Le code qui permet de convertir les données brut du fichier csv en une liste pour traiter facilement les données avec Python
4
5 #1 Avec des virgules
6 with open('Negative_Without_PV.csv', 'r') as f:
7     data = f.read().replace('\n', ',\n')
8
9 with open('Negative_With_Virgule.csv', 'w') as f:
10    f.write(data)
11
12 #2 Sans le saut de ligne
13 with open('Negative_With_Virgule.csv', 'r') as f:
14     data = f.read().replace('\n', '')
15
16 with open('Negative_Without_Newline.csv', 'w') as f:
17    f.write(data)
18
19 #3 Remplacer les ; par des ,
20 with open('Negative.csv', 'r') as f:
21     data = f.read().replace(';', ',')
22
23 with open('Negative_Without_PV.csv', 'w') as f:
24    f.write(data)
25
26 #4 Mettre les noms de fichiers .jpg entre guillemets
27 # Ouvrir le fichier Ancre_Without_Newline.csv en mode lecture
28 with open('Negative_Without_Newline.csv', 'r') as f:
29     # Lire le contenu du fichier dans la variable data
30     data = f.read()
31
32     # Appliquer l'expression régulière pour trouver les noms de fichiers jpg et les mettre entre guillemets
33 data = re.sub(r'(\b[d]+\.(jpg|JPG)\b)', r'"\1"', data)
34
35 # Ouvrir le fichier Ancre_With_Qoutes.csv en mode écriture
36 with open('Negative_With_Qoutes.csv', 'w') as f:
37     # Écrire le contenu modifié dans le fichier
38     f.write(data)
39
40 #5 Conversion en tableau
41 # Vérifier si le nombre de data est multiple de 2
42 # Ouvrir le fichier Ancre_Without_Newline.csv en mode lecture
43 with open('Ancre_With_Qoutes.csv', 'r') as f:
44     # Lire le contenu du fichier dans la variable data
45     data = f.read()
46 print(len(data))
47
48 #6

```

Python 3 local • Python de Thonny



```

import_csv_to_sqlite.py *  data_cleaning_triplets.py * 
49
50 with open('NEGATIVE_WITHOUT_NEWLINE.csv', 'r') as f:
51     # Lire le contenu du fichier dans la variable data
52     data = f.read()
53
54     # Appliquer l'expression régulière pour trouver les noms de fichiers jpg et les mettre entre guillemets
55 data = re.sub(r'(\b[d]+\.(jpg|JPG)\b)', r'"\1"', data)
56
57     # Ouvrir le fichier Ancre_With_Qoutes.csv en mode écriture
58 with open('Negative_With_Qoutes.csv', 'w') as f:
59     # Écrire le contenu modifié dans le fichier
60     f.write(data)
61
62 #5 Conversion en tableau
63 # Vérifier si le nombre de data est multiple de 2
64 # Ouvrir le fichier Ancre_Without_Newline.csv en mode lecture
65 with open('Ancre_With_Qoutes.csv', 'r') as f:
66     # Lire le contenu du fichier dans la variable data
67     data = f.read()
68 print(len(data))
69
70 #6
71 impairs = []
72 paires = []
73
74 with open('Negative_With_Qoutes.csv', newline='') as csvfile:
75     reader = csv.reader(csvfile)
76     for row in reader:
77         for i in range(len(row)):
78             if i % 2 == 0: # position impaire
79                 impairs.append(row[i])
80             else: # position paire
81                 paires.append(row[i])
82
83 print("Liste des éléments en position paire :", paires)
84 print("Liste des éléments en position impaire :", impairs)
85
86 #7
87 liste_d_entiers = ['496', '9313', '7200', '4643', '2988', '5029', '52', '6247', '4441', '6147', '620', '2306', '9544', '2060', '6118', '1035', '6814', '1705
88 print(len(liste_d_entiers))
89 sans_doublons = list(set(map(int, liste_d_entiers)))
90 print(len(sans_doublons))
91 print(sans_doublons)
92
93 #8
94 liste392 = ['030848.jpg', '093653.jpg', '188283.jpg', '178433.jpg', '105432.jpg', '079773.jpg', '141930.jpg', '151909.jpg', '165740.jpg', '103819.jpg'
95 del liste392[:392]
96 print(len(liste392))
97 print(liste392)
98
99

```

Python 3 local • Python de Thonny

Voici le code Python qui nous a permis d'importer les données Kaggle dans notre base de données « dbtriplets.sqlite » pour la **table Ancre** :

The screenshot shows a Jupyter Notebook environment. At the top, there are two tabs: 'import\_csv\_to\_sqlite.py' and 'data\_cleaning\_triplets.py'. The code in 'import\_csv\_to\_sqlite.py' is as follows:

```

1 import sqlite3
2
3 # connexion à la base de données
4 conn = sqlite3.connect('dbtriplets.sqlite')
5 cur = conn.cursor()
6
7 # liste de données
8 id1Ancre = [1, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41]
9 AnchorAncre = ['144692.jpg', '080830.jpg', '132686.jpg', '119327.jpg', '108120.jpg', '065145.jpg', '051068.jpg', '021977.jpg', '173157.jpg', '119327.jpg']
10
11 # Insertion des données de la première liste
12 for element1, element2 in zip(id1Ancre, AnchorAncre):
13     cur.execute('INSERT INTO Ancre (id1, Anchor) VALUES (?, ?)', (element1, element2))
14
15 # Validation de la transaction et fermeture de la connexion
16 conn.commit()
17 conn.close()
18

```

In the 'Assistant' panel on the right, it says: 'The code in [import\\_csv\\_to\\_sqlite.py](#) looks good.' and 'If it is not working as it should, then consider using some general debugging techniques.'

Below the code editor is a 'Console' pane with the following output:

```

>>> %Run import_csv_to_sqlite.py
>>>

```

At the bottom right, it says 'Python 3 local • Python de Thonny'.

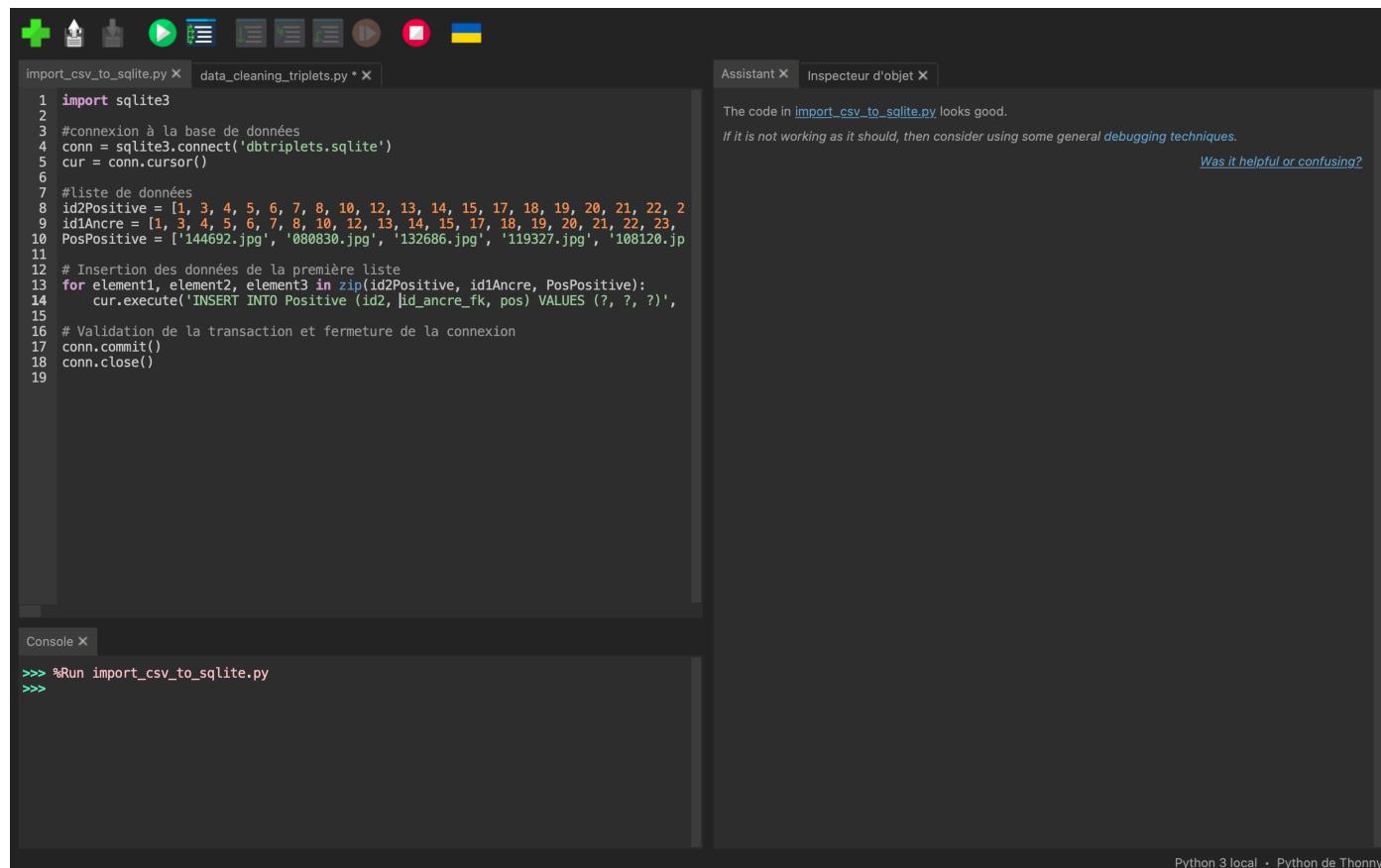
Voici le résultat sur SQLite pour la **table Ancre** :

The screenshot shows the SQLite Explorer interface. On the left, the 'EXPLORATEUR' sidebar shows a project structure with files like 'Ancre.csv', 'Ancre\_Anchor.csv', 'Ancre\_Id1.csv', etc. In the center, there are two SQLite databases: 'Untitled-1' and 'triplets.sqlite'. The 'Untitled-1' database has a single table 'Ancre' with the following data:

id1	anchor
1	144692.jpg
3	080830.jpg
4	132686.jpg
5	119327.jpg
6	108120.jpg
7	065145.jpg
8	051068.jpg
10	021977.jpg
12	173157.jpg
13	167985.jpg
14	173804.jpg
15	179904.jpg
17	065911.jpg
18	023811.jpg
19	093994.jpg
20	123963.jpg
21	169184.jpg
22	174913.jpg
23	096308.jpg
24	139112.jpg
25	117616.jpg
27	041281.jpg
28	162941.jpg
29	171940.jpg
30	068739.jpg
31	025637.jpg

The 'triplets.sqlite' database also contains the 'Ancre' table with the same data. A yellow circle highlights the page navigation controls at the bottom of the central pane, showing '1 / 8' and '1 - 50 of 392'.

Voici le code Python qui nous a permis d'importer les données Kaggle dans notre base de données « dbtriplets.sqlite » pour la **table Positive** :



```

import_csv_to_sqlite.py *  data_cleaning_triplets.py * 
1 import sqlite3
2
3 #connexion à la base de données
4 conn = sqlite3.connect('dbtriplets.sqlite')
5 cur = conn.cursor()
6
7 #liste de données
8 idPositive = [1, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 2
9 idAncre = [1, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23,
10 PosPositive = ['144692.jpg', '080830.jpg', '132686.jpg', '119327.jpg', '108120.jp
11
12 # Insertion des données de la première liste
13 for element1, element2, element3 in zip(idPositive, idAncre, PosPositive):
14     cur.execute('INSERT INTO Positive (id2, id_ancre_fk, pos) VALUES (?, ?, ?)', [
15
16 # Validation de la transaction et fermeture de la connexion
17 conn.commit()
18 conn.close()
19

```

The code in `import_csv_to_sqlite.py` looks good.  
If it is not working as it should, then consider using some general debugging techniques.  
[Was it helpful or confusing?](#)

Console X

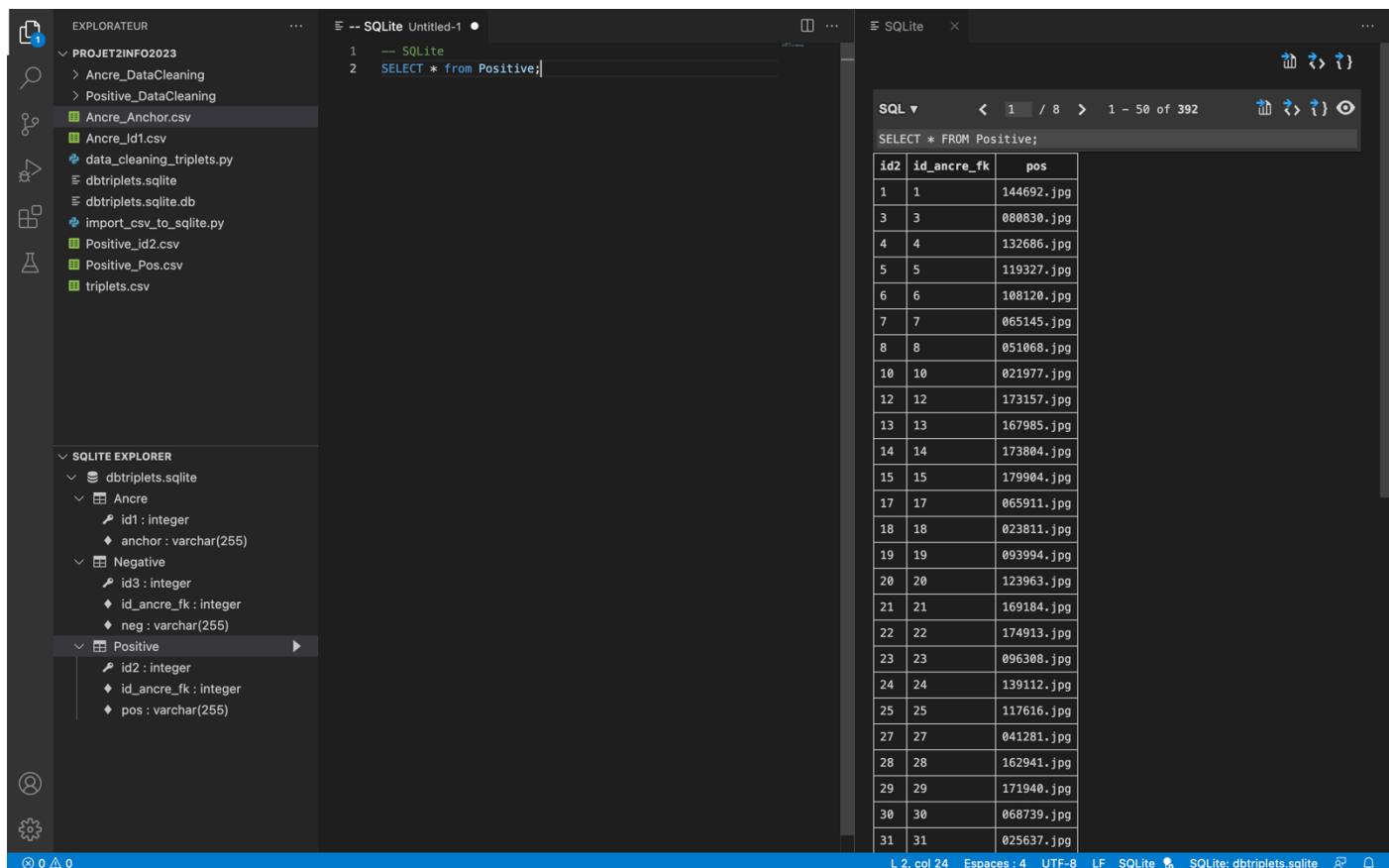
```

>>> %Run import_csv_to_sqlite.py
>>>

```

Python 3 local • Python de Thoriny

Voici le résultat sur SQLite pour la **table Positive** :



EXPLORATEUR

- PROJET2INFO2023
  - Ancre\_DataCleaning
  - Positive\_DataCleaning
  - Ancre\_Anchor.csv
  - Ancre\_Id1.csv
  - data\_cleaning\_triplets.py
  - dbtriplets.sqlite
  - dbtriplets.sqlite.db
  - import\_csv\_to\_sqlite.py
  - Positive\_id2.csv
  - Positive\_Pos.csv
  - triplets.csv

SQLITE EXPLORER

- dbtriplets.sqlite
  - Ancre
  - Negative
  - Positive

-- SQLite Untitled-1 ●

```

1 -- SQLite
2 SELECT * from Positive;

```

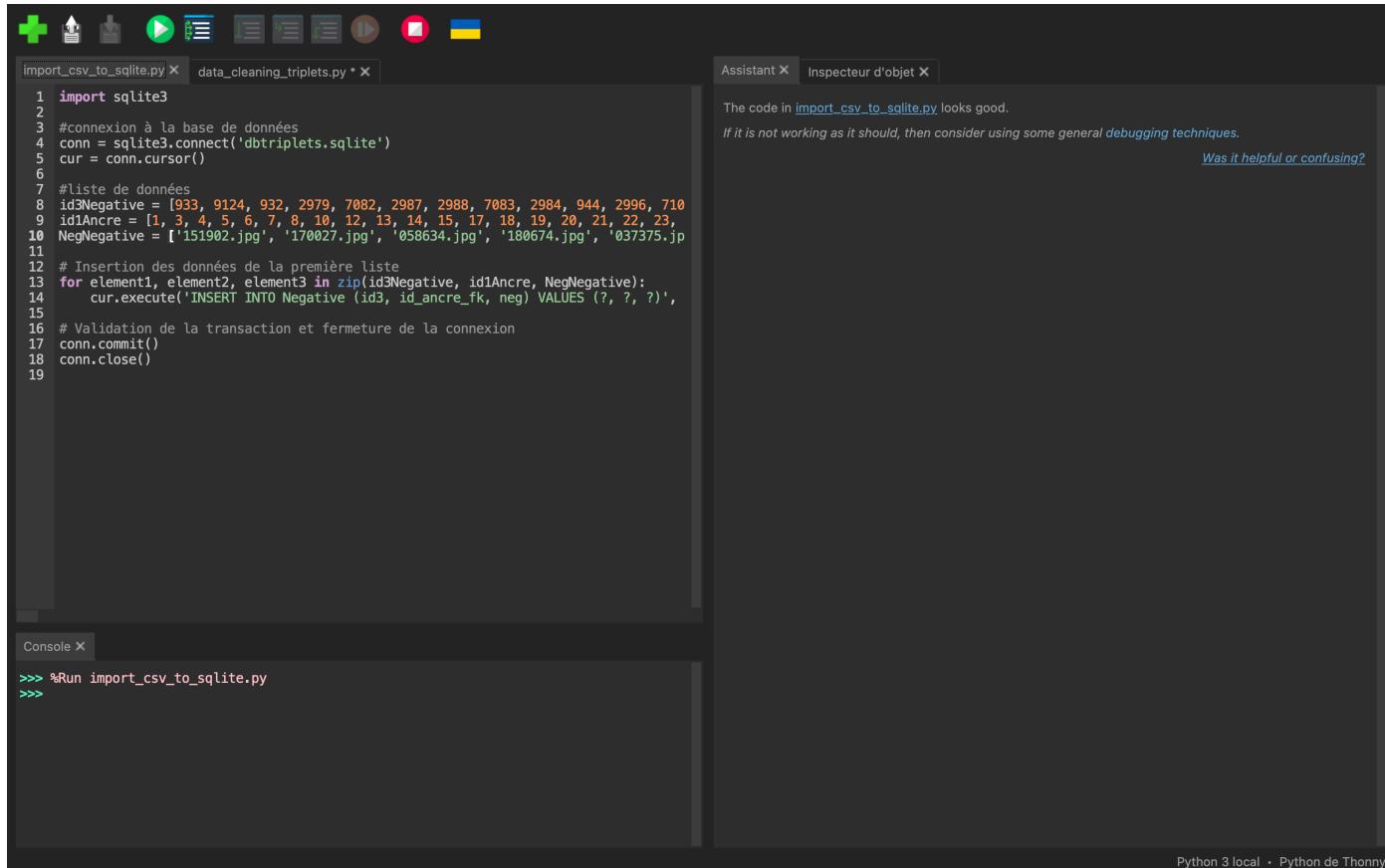
SQLite

SELECT \* FROM Positive;

id2	id_ancre_fk	pos
1	1	144692.jpg
3	3	080830.jpg
4	4	132686.jpg
5	5	119327.jpg
6	6	108120.jpg
7	7	065145.jpg
8	8	051068.jpg
10	10	021977.jpg
12	12	173157.jpg
13	13	167985.jpg
14	14	173804.jpg
15	15	179904.jpg
17	17	065911.jpg
18	18	023811.jpg
19	19	093994.jpg
20	20	123963.jpg
21	21	169184.jpg
22	22	174913.jpg
23	23	096308.jpg
24	24	139112.jpg
25	25	117616.jpg
27	27	041281.jpg
28	28	162941.jpg
29	29	171940.jpg
30	30	068739.jpg
31	31	025637.jpg

L 2, col 24 Espaces : 4 UTF-8 LF SQLite % SQLite: dbtriplets.sqlite

Voici le code Python qui nous a permis d'importer les données Kaggle dans notre base de données « dbtriplets.sqlite » pour la **table Negative** :



```

import_csv_to_sqlite.py *  data_cleaning_triplets.py *
1 import sqlite3
2
3 #connexion à la base de données
4 conn = sqlite3.connect('dbtriplets.sqlite')
5 cur = conn.cursor()
6
7 #liste de données
8 id3Negative = [933, 9124, 932, 2979, 7082, 2987, 2988, 7083, 2984, 944, 2996, 710
9 id1Ancre = [1, 3, 4, 5, 6, 7, 8, 10, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23,
NegNegative = ['151902.jpg', '170027.jpg', '058634.jpg', '180674.jpg', '037375.jp
11
12 # Insertion des données de la première liste
13 for element1, element2, element3 in zip(id3Negative, id1Ancre, NegNegative):
14     cur.execute('INSERT INTO Negative (id3, id_ancre_fk, neg) VALUES (?, ?, ?)', (
15
16 # Validation de la transaction et fermeture de la connexion
17 conn.commit()
18 conn.close()
19

```

Console X

```

>>> %Run import_csv_to_sqlite.py
>>>

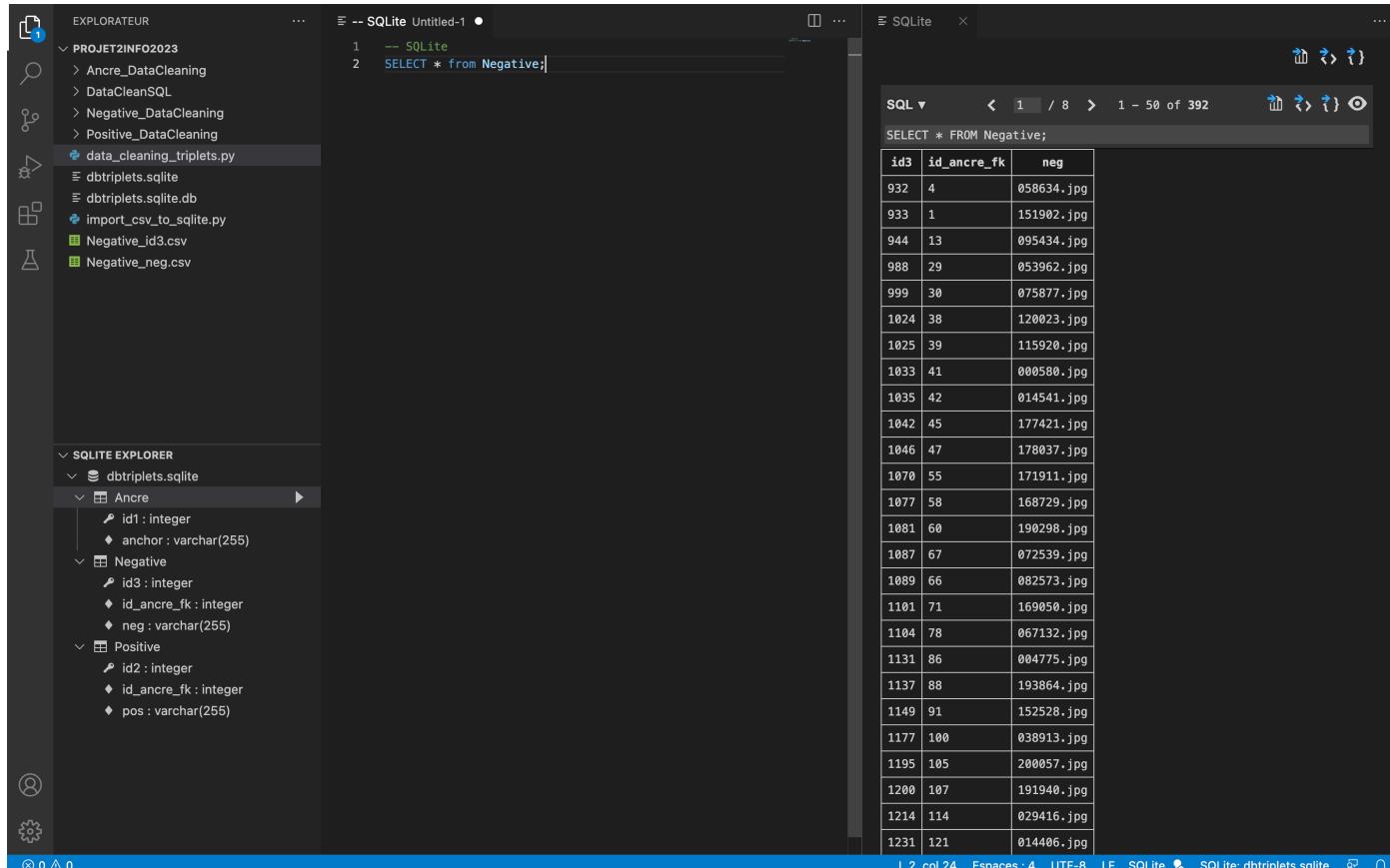
```

Assistant X Inspecteur d'objet X

The code in [import\\_csv\\_to\\_sqlite.py](#) looks good.  
If it is not working as it should, then consider using some general [debugging techniques](#).  
[Was it helpful or confusing?](#)

Python 3 local • Python de Thonny

Voici le résultat sur SQLite pour la **table Negative** :



EXPLORATEUR

- PROJET2INFO2023
  - Ancre\_DataCleaning
  - DataCleanSQL
  - Negative\_DataCleaning
  - Positive\_DataCleaning
  - data\_cleaning\_triplets.py
  - dbtriplets.sqlite
  - dbtriplets.sqlite.db
  - import\_csv\_to\_sqlite.py
  - Negative\_id3.csv
  - Negative\_neg.csv

-- SQLite Untitled-1 ●

```

1 -- SQLite
2 SELECT * from Negative;

```

SQLITE EXPLORER

- dbtriplets.sqlite
  - Ancre
  - Negative
  - Positive

SQLite

id3	id_ancre_fk	neg
932	4	058634.jpg
933	1	151902.jpg
944	13	095434.jpg
988	29	053962.jpg
999	30	075877.jpg
1024	38	120023.jpg
1025	39	115920.jpg
1033	41	000580.jpg
1035	42	014541.jpg
1042	45	177421.jpg
1046	47	178037.jpg
1070	55	171911.jpg
1077	58	168729.jpg
1081	60	190298.jpg
1087	67	072539.jpg
1089	66	082573.jpg
1101	71	169050.jpg
1104	78	067132.jpg
1131	86	004775.jpg
1137	88	193864.jpg
1149	91	152528.jpg
1177	100	038913.jpg
1195	105	200057.jpg
1200	107	191940.jpg
1214	114	029416.jpg
1231	121	014406.jpg

#### 4. Introduction des requêtes SQL

- Cette requête SQL sélectionne toutes les lignes de la table "Positive" où la valeur de la colonne "id2" est comprise entre 10 et 20 inclus.

The screenshot shows the SQLite Manager interface. On the left, the SQL tab contains the following code:

```
1 -- SQLite Untitled-1
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
```

On the right, the Results tab displays the following table:

id2	id_ancre_fk	pos
10	10	021977.jpg
12	12	173157.jpg
13	13	167985.jpg
14	14	173804.jpg
15	15	179904.jpg
17	17	065911.jpg
18	18	023811.jpg
19	19	093994.jpg
20	20	123963.jpg

Cette requête SQL sélectionne toutes les valeurs distinctes de la colonne "id1" de la table "Ancre". Cette requête joint les trois tables ensemble en utilisant les identifiants correspondants et sélectionne les colonnes anchor, pos et neg pour chaque ligne.

The screenshot shows the SQLite Manager interface. On the left, the SQL tab contains the following code:

```
1 -- SQLite Untitled-1
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
3 --SELECT DISTINCT id1 FROM Ancre;
4 SELECT a.anchor, p.pos, n.neg
5 FROM Ancre a
6 INNER JOIN Positive p ON a.id1 = p.id2
7 INNER JOIN Negative n ON a.id1 = n.id3
8 |
```

On the right, the Results tab displays the following table:

id1	anchor	pos	neg
10	10	021977.jpg	00000000000000000000000000000000
12	12	173157.jpg	00000000000000000000000000000000
13	13	167985.jpg	00000000000000000000000000000000
14	14	173804.jpg	00000000000000000000000000000000
15	15	179904.jpg	00000000000000000000000000000000
17	17	065911.jpg	00000000000000000000000000000000
18	18	023811.jpg	00000000000000000000000000000000
19	19	093994.jpg	00000000000000000000000000000000
20	20	123963.jpg	00000000000000000000000000000000

Cette requête sélectionne toutes les lignes dans la table Ancre où la colonne anchor contient les nombres "123" dans le nom de fichier jpg.

The screenshot shows the SQLite Manager interface. On the left, the SQL tab contains the following code:

```
1 -- SQLite Untitled-1
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
3 --SELECT DISTINCT id1 FROM Ancre;
4
5 /* SELECT a.anchor, p.pos, n.neg
6 FROM Ancre a
7 INNER JOIN Positive p ON a.id1 = p.id2
8 INNER JOIN Negative n ON a.id1 = n.id3 */
9
10
11 SELECT * FROM Ancre
12 WHERE anchor LIKE '%123%'
```

On the right, the Results tab displays the following table:

id1	anchor
20	123963.jpg
46	123916.jpg
55	123764.jpg
228	123790.jpg
410	123544.jpg
434	071236.jpg

2. Cette requête sélectionne toutes les données dans la table Ancre qui ont au moins une occurrence positive mais pas de négative.

The screenshot shows two panes in the SQLite Manager interface. The left pane contains the following SQL code:

```

1 -- SQLite
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
3
4 --SELECT DISTINCT id1 FROM Ancre;
5
6 /* SELECT a.anchor, p.pos, n.neg
7  FROM Ancre a
8  INNER JOIN Positive p ON a.id1 = p.id2
9  INNER JOIN Negative n ON a.id1 = n.id3 */
10
11 /*SELECT * FROM Ancre
12 WHERE anchor LIKE '%123%'*/
13
14 SELECT *
15 FROM Ancre
16 WHERE id1 IN (
17   SELECT id2
18   FROM Positive
19 )
20 AND id1 NOT IN (
21   SELECT id3
22   FROM Negative
23 )
24
25
26
27
28
29
30
31
32

```

The right pane displays the results of the query, showing a table with columns 'id1' and 'anchor'. The data includes:

id1	anchor
1	144692.jpg
3	080830.jpg
4	132686.jpg
5	119327.jpg
6	188120.jpg
7	065145.jpg
8	051068.jpg
10	021977.jpg
12	173157.jpg
13	167985.jpg
14	173804.jpg
15	179904.jpg
17	065911.jpg

Cette requête toutes les lignes de la table "Ancre" où la valeur de la colonne "id1" est présente dans la sous-requête qui sélectionne toutes les valeurs de la colonne "id2" de la table "Positive". De plus, elle filtre les résultats en ne sélectionnant que les lignes où la valeur de la colonne "anchor" contient le chiffre "75" n'importe où dans la chaîne de caractères.

The screenshot shows two panes in the SQLite Manager interface. The left pane contains the following SQL code:

```

1 -- SQLite
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
3
4 --SELECT DISTINCT id1 FROM Ancre;
5
6 /* SELECT a.anchor, p.pos, n.neg
7  FROM Ancre a
8  INNER JOIN Positive p ON a.id1 = p.id2
9  INNER JOIN Negative n ON a.id1 = n.id3 */
10
11 /*SELECT * FROM Ancre
12 WHERE anchor LIKE '%123%'*/
13
14 /*SELECT *
15  FROM Ancre
16  WHERE id1 IN (
17    SELECT id2
18    FROM Positive
19  )
20  AND id1 NOT IN (
21    SELECT id3
22    FROM Negative
23  )*/
24
25 SELECT *
26 FROM Ancre
27 WHERE id1 IN (
28   SELECT id2
29   FROM Positive
30 )
31 AND anchor LIKE '%75%';
32

```

The right pane displays the results of the query, showing a table with columns 'id1' and 'anchor'. The data includes:

id1	anchor
120	175486.jpg
148	129775.jpg
157	095757.jpg
187	075254.jpg
212	080756.jpg
234	118275.jpg
259	133675.jpg
349	052875.jpg
351	075619.jpg
375	075477.jpg
383	015975.jpg
387	009675.jpg
394	047560.jpg
440	075877.jpg

3. La requête permet ainsi d'obtenir une vue d'ensemble des occurrences des identifiants dans la table Negative.

The screenshot shows two panes in the SQLite Manager interface. The left pane contains the following SQL code:

```

1 -- SQLite
2 --SELECT * FROM Positive WHERE id2 BETWEEN 10 AND 20;
3
4 --SELECT DISTINCT id1 FROM Ancre;
5
6 /* SELECT a.anchor, p.pos, n.neg
7  FROM Ancre a
8  INNER JOIN Positive p ON a.id1 = p.id2
9  INNER JOIN Negative n ON a.id1 = n.id3 */
10
11 /*SELECT * FROM Ancre
12 WHERE anchor LIKE '%123%'*/
13
14 /*SELECT *
15  FROM Ancre
16  WHERE id1 IN (
17    SELECT id2
18    FROM Positive
19  )
20  AND id1 NOT IN (
21    SELECT id3
22    FROM Negative
23  )*/
24
25 /* SELECT *
26  FROM Ancre
27  WHERE id1 IN (
28    SELECT id2
29    FROM Positive
30
31    AND anchor LIKE '%75%' */
32
33 SELECT id3, COUNT(*) AS count_negative
34 FROM Negative
35 GROUP BY id3;
36
37
38
39

```

The right pane displays the results of the query, showing a table with columns 'id3' and 'count\_negative'. The data includes:

id3	count_negative
932	1
933	1
944	1
988	1
999	1
1024	1
1025	1
1033	1
1035	1
1042	1
1046	1
1070	1
1077	1
1081	1
1087	1
1089	1
1101	1
1104	1
1131	1
1137	1
1149	1
1177	1
1195	1
1200	1
1214	1
1231	1
1245	1

Cette requête qui regroupe les enregistrements de chaque table selon l'ID et affiche l'ancrage, la valeur positive et la valeur négative correspondants.

```

-- -- SQLite Untitled-1 ●
18 |     FROM Positive
19 | )
20 | AND id1 NOT IN (
21 |     SELECT id3
22 |     FROM Negative
23 | )*/
24 |
25 /* SELECT *
26 FROM Ancre
27 WHERE id1 IN (
28 |     SELECT id2
29 |     FROM Positive
30 | )
31 AND anchor LIKE '%75%' */
32
33 /*SELECT id3, COUNT(*) AS count_negative
FROM Negative
GROUP BY id3;*/
36
37 SELECT Ancre.anchor, Positive.pos, Negative.neg
FROM Ancre
INNER JOIN Positive ON Ancre.id1 = Positive.id2
INNER JOIN Negative ON Ancre.id1 = Negative.id3
GROUP BY Ancre.id1
41
42
43
44
45

```

4. Cette requête retourne le nombre d'occurrences de chaque valeur de anchor dans la table Ancre, en filtrant les résultats pour ne renvoyer que les ancrées ayant plus d'occurrences positives que négatives dans les tables Positive et Negative.

```

-- -- SQLite Untitled-1 ●
18 |     FROM Positive
19 | )
20 | AND id1 NOT IN (
21 |     SELECT id3
22 |     FROM Negative
23 | )*/
24 |
25 /* SELECT *
26 FROM Ancre
27 WHERE id1 IN (
28 |     SELECT id2
29 |     FROM Positive
30 | )
31 AND anchor LIKE '%75%' */
32
33 /*SELECT id3, COUNT(*) AS count_negative
FROM Negative
GROUP BY id3;*/
36
37 /*SELECT Ancre.anchor, Positive.pos, Negative.neg
FROM Ancre
INNER JOIN Positive ON Ancre.id1 = Positive.id2
INNER JOIN Negative ON Ancre.id1 = Negative.id3
GROUP BY Ancre.id1 */
41
42
43 SELECT Ancre.anchor, COUNT(*)
FROM Ancre
LEFT JOIN Positive ON Ancre.id1 = Positive.id2
LEFT JOIN Negative ON Ancre.id1 = Negative.id3
GROUP BY Ancre.anchor
48 HAVING COUNT(Positive.id2) > COUNT(Negative.id3)
49
50
51
52
53
54

```

anchor	COUNT(*)
000074.jpg	1
001013.jpg	1
003263.jpg	1
003637.jpg	1
004159.jpg	1
005259.jpg	1
005461.jpg	1
005789.jpg	1
006140.jpg	1
006921.jpg	1
008567.jpg	1
008613.jpg	1
009207.jpg	1
009675.jpg	1
009926.jpg	1
010373.jpg	1
010866.jpg	1
011064.jpg	1
011289.jpg	1
011493.jpg	1
012770.jpg	1
013061.jpg	1
013318.jpg	1
013560.jpg	1
013561.jpg	1
014429.jpg	1
015102.jpg	1

## 5. Différence et Union :

```

-- SQLite Untitled-1 •
41 GROUP BY Ancre.id1 */
42
43 /* SELECT Ancre.anchor, COUNT(*)
44 FROM Ancre
45 LEFT JOIN Positive ON Ancre.id1 = Positive.id2
46 LEFT JOIN Negative ON Ancre.id1 = Negative.id3
47 GROUP BY Ancre.anchor
48 HAVING COUNT(Positive.id2) > COUNT(Negative.id3) */
49
50 SELECT anchor FROM Ancre
51 WHERE id1 NOT IN (
52 | SELECT id2 FROM Positive
53 | UNION
54 | SELECT id3 FROM Negative
55 );
56
57

```

SQL ▼

1 - 0 of 0

anchor

Union :

```

-- SQLite Untitled-1 •
41 GROUP BY Ancre.id1 */
42
43 /* SELECT Ancre.anchor, COUNT(*)
44 FROM Ancre
45 LEFT JOIN Positive ON Ancre.id1 = Positive.id2
46 LEFT JOIN Negative ON Ancre.id1 = Negative.id3
47 GROUP BY Ancre.anchor
48 HAVING COUNT(Positive.id2) > COUNT(Negative.id3) */
49
50 SELECT anchor FROM Ancre
51 WHERE id1 IN (
52 | SELECT id2 FROM Positive
53 | UNION
54 | SELECT id3 FROM Negative
55 );
56
57

```

SQL ▼

1 - 50 of 392

anchor
144692.jpg
080830.jpg
132686.jpg
119327.jpg
108120.jpg
065145.jpg
051068.jpg

Intersection :

```

-- SQLite Untitled-1 •
41 GROUP BY Ancre.id1 */
42
43 /* SELECT Ancre.anchor, COUNT(*)
44 FROM Ancre
45 LEFT JOIN Positive ON Ancre.id1 = Positive.id2
46 LEFT JOIN Negative ON Ancre.id1 = Negative.id3
47 GROUP BY Ancre.anchor
48 HAVING COUNT(Positive.id2) > COUNT(Negative.id3) */
49
50 /* SELECT anchor FROM Ancre
51 WHERE id1 IN (
52 | SELECT id2 FROM Positive
53 | UNION
54 | SELECT id3 FROM Negative
55 ); */
56
57 SELECT anchor FROM Ancre
58 WHERE id1 IN (
59 | SELECT id2 FROM Positive
60 ) AND id1 IN (
61 | SELECT id3 FROM Negative
62 );
63

```

SQL ▼

1 - 0 of 0

anchor

## Division :

```
-- SQLite Untitled-1 ●
58 WHERE id1 IN (
59 | SELECT id2 FROM Positive
60 ) AND id1 IN (
61 | SELECT id3 FROM Negative
62 ); */
63
64 SELECT anchor FROM Ancre
65 WHERE NOT EXISTS (
66 | SELECT * FROM Positive
67 WHERE NOT EXISTS (
68 | SELECT * FROM Negative
69 WHERE Negative.id3 = Positive.id2
70 AND Ancre.id1 = Positive.id2
71 )
72 );
73
```

SQL ▾ < 1 / 1 > 1 - 0 of 0

## ALL :

```
-- SQLite Untitled-1 ●
65 WHERE NOT EXISTS (
66 | SELECT * FROM Positive
67 WHERE NOT EXISTS (
68 | SELECT * FROM Negative
69 WHERE Negative.id3 = Positive.id2
70 AND Ancre.id1 = Positive.id2
71 )
72 );*/
73
74 SELECT anchor FROM Ancre
75 WHERE id1 <= ALL (SELECT id2 FROM Positive);
76
77
```

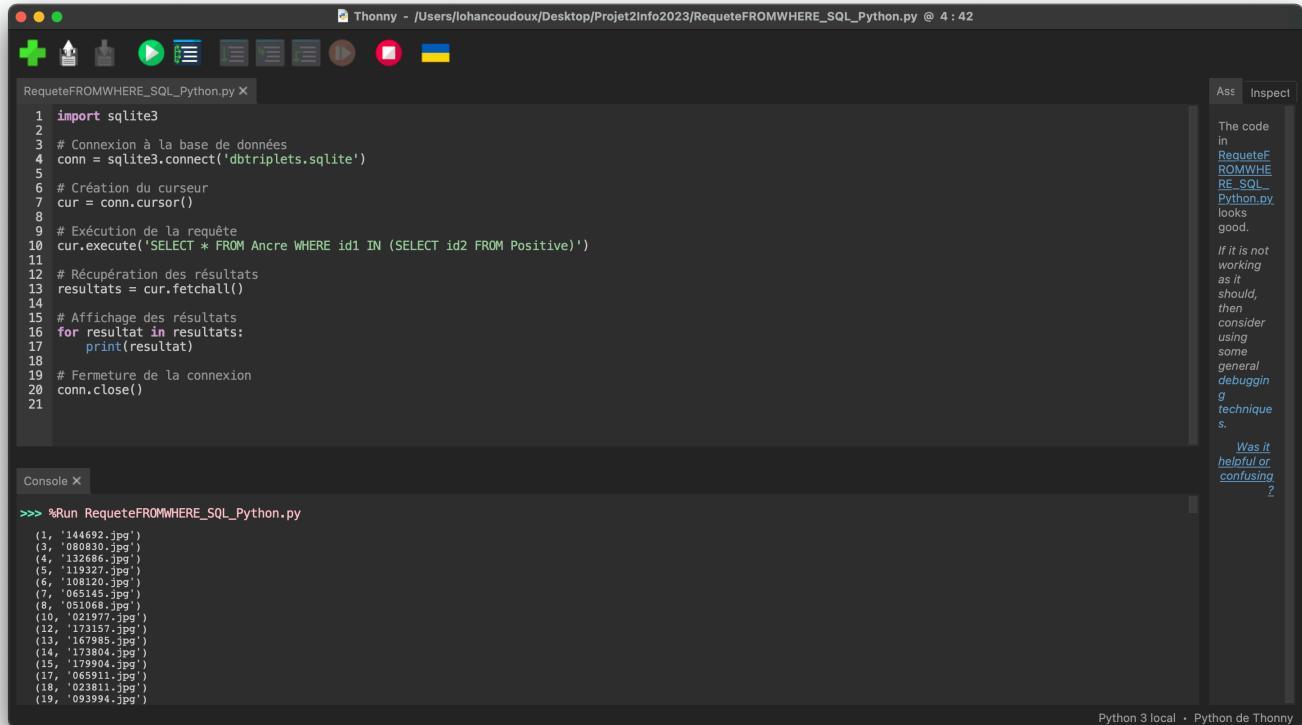
SQL ▾ < 1 / 1 > 1 - 0 of 0

## ANY :

```
-- SQLite Untitled-1 ●
66 | SELECT * FROM Positive
67 WHERE NOT EXISTS (
68 | SELECT * FROM Negative
69 WHERE Negative.id3 = Positive.id2
70 AND Ancre.id1 = Positive.id2
71 )
72 );*/
73
74 /*SELECT anchor FROM Ancre
75 WHERE id1 <= ANY (SELECT id2 FROM Positive);*/
76
77 SELECT anchor FROM Ancre
78 WHERE id1 = ANY (
79 | SELECT id3 FROM Negative
80 );
81
```

SQL ▾ < 1 / 1 > 1 - 0 of 0

## ii. Introduction de requêtes SQL dans PYTHON



```

import sqlite3
# Connexion à la base de données
conn = sqlite3.connect('dbtriplets.sqlite')
# Création du curseur
cur = conn.cursor()
# Exécution de la requête
cur.execute('SELECT * FROM Ancre WHERE id1 IN (SELECT id2 FROM Positive)')
# Récupération des résultats
resultats = cur.fetchall()
# Affichage des résultats
for resultat in resultats:
    print(resultat)
# Fermeture de la connexion
conn.close()

```

>>> %Run RequeteFROMWHERE\_SQL\_Python.py

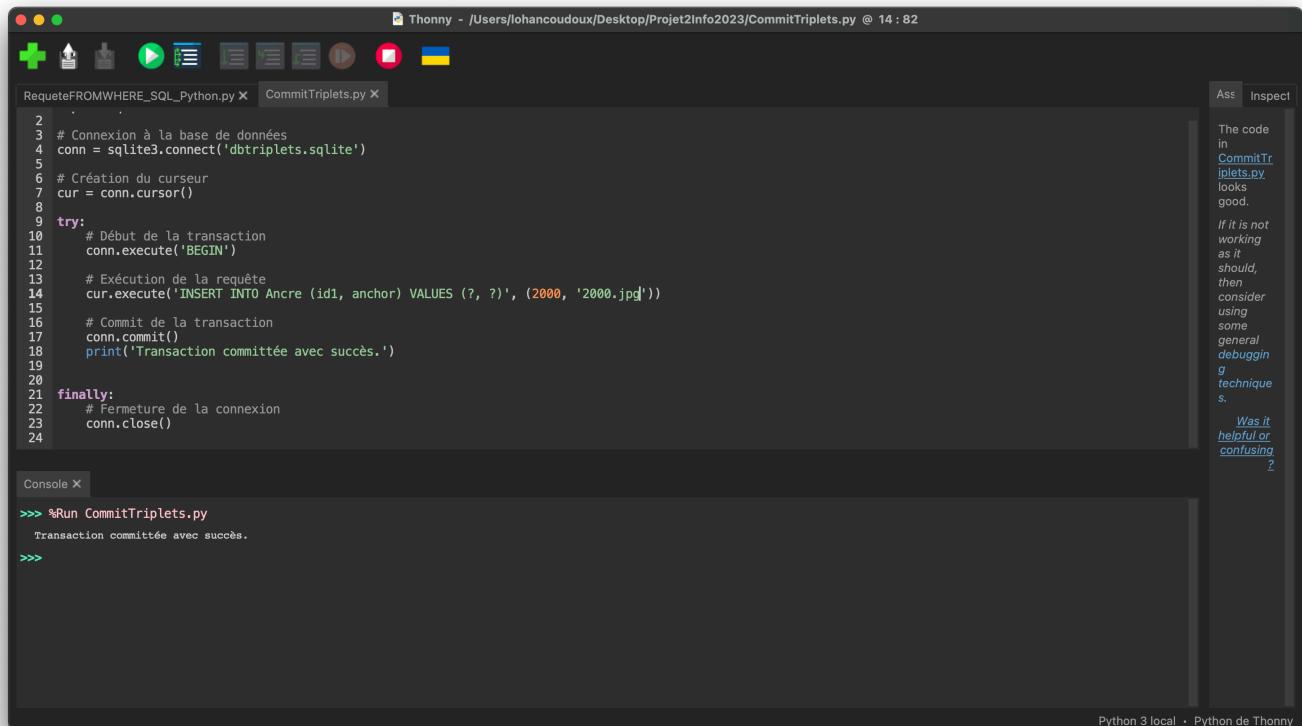
```

(1, '144692.jpg')
(3, '080830.jpg')
(4, '132686.jpg')
(5, '119327.jpg')
(6, '108120.jpg')
(7, '098111.jpg')
(8, '051068.jpg')
(9, '021977.jpg')
(10, '173157.jpg')
(11, '167985.jpg')
(12, '173804.jpg')
(13, '173991.jpg')
(14, '062811.jpg')
(15, '023811.jpg')
(16, '093994.jpg')
(17, '023811.jpg')
(18, '093994.jpg')
(19, '093994.jpg')

```

Python 3 local • Python de Thonny

### 1. Implémentation dans notre programme Python d'un commit :



```

# Connexion à la base de données
conn = sqlite3.connect('dbtriplets.sqlite')
# Création du curseur
cur = conn.cursor()
try:
    # Début de la transaction
    conn.execute('BEGIN')
    # Exécution de la requête
    cur.execute('INSERT INTO Ancre (id1, anchor) VALUES (?, ?)', (2000, '2000.jpg'))
    # Commit de la transaction
    conn.commit()
    print('Transaction committée avec succès.')
finally:
    # Fermeture de la connexion
    conn.close()

```

>>> %Run CommitTriples.py

```

Transaction committée avec succès.
>>>

```

Python 3 local • Python de Thonny

## 2. Implémentation dans notre programme Python d'un roll back :

The screenshot shows the Thonny Python IDE interface. The code editor tab is titled 'RollBackTriplets.py' and contains the following Python script:

```
5 # Cr ation du curseur
6 cur = conn.cursor()
7
8 try:
9     # D but de la transaction
10    conn.execute('BEGIN')
11
12    # Ex cution de la requ te
13    cur.execute('INSERT INTO Ancre (id1, anchor) VALUES (?, ?)', (2000, '2000.jpg'))
14
15    # Commit de la transaction
16    conn.commit()
17    print('Transaction committ e avec succ s.')
18
19 |
20
21 # Rollback de la transaction
22 conn.rollback()
23 print('Transaction annul e avec succ s.')
24
25 # Fermeture de la connexion
26 conn.close()
27
```

The console tab shows the output of running the script:

```
>>> %Run RollBackTriplets.py
UNIQUE constraint failed: Ancre.id1
Transaction annul e avec succ s.
>>>
```

A tooltip on the right side of the interface provides a brief explanation of the code's purpose:

The code in [RollBackTriplets.py](#) looks good.  
If it is not working as it should, then consider using some general debugging techniques.  
Was it helpful or confusing?