Mai Nguyen

TD03 G7

**Informatique S4 : Bases de Données**

**Projet pratique N°2**

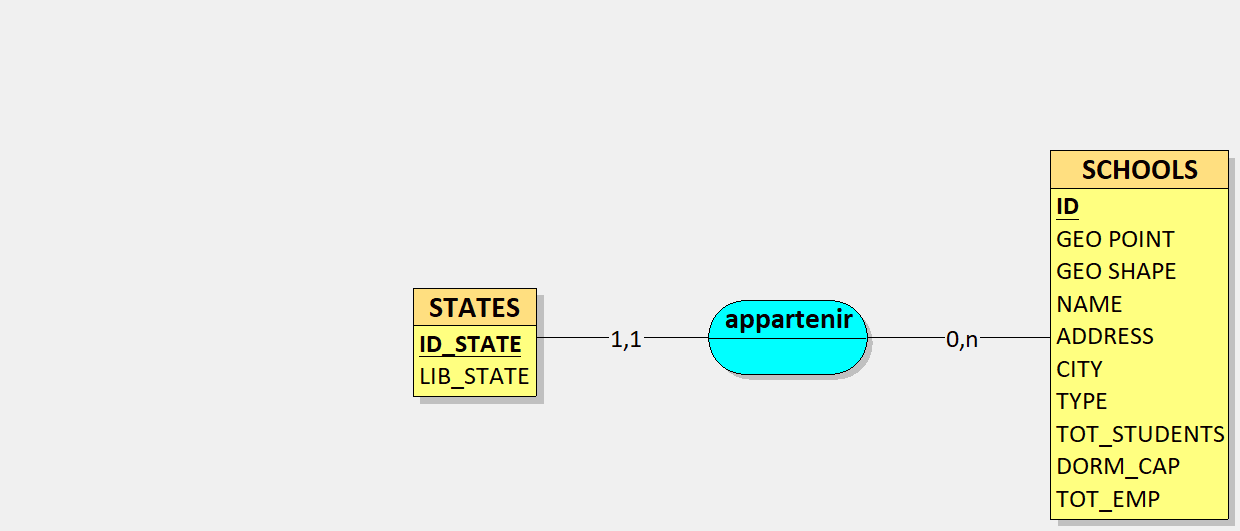
**US Colleges and Universities**

**ETAPE 1 : Structuration des données, lien avec Python**

i. Téléchargement du Data set *us-colleges-and-universities.csv* sur kaggle.com et nettoyage (*nettoyer-us-colleges-and-universities.csv*)

ii. Rétro conception

1. **Le modèle Entité/Association**



Réalisé avec Looping

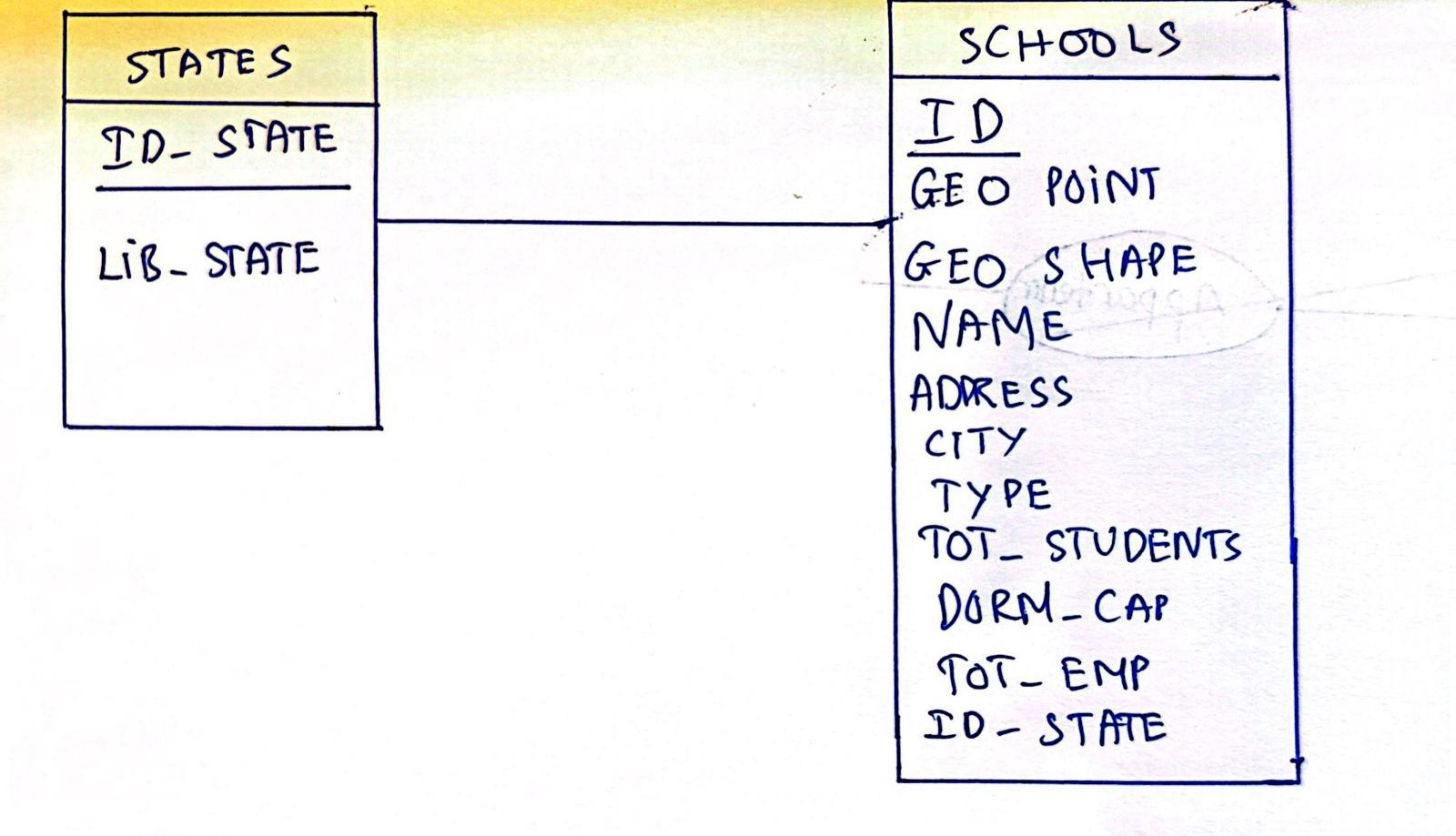
Lecture : 0 à N écoles appartiennent à 1 état et 1 état possède 0 à N écoles.

1. **Les relations et les schémas relationnels**

SCHOOLS (ID, GEO POINT, GEO SHAPE, NAME, ADDRESS, CITY, TYPE, TOT\_STUDENTS, DORM\_CAP, TOT\_EMP, #ID\_STATE)

STATES (ID\_STATE, LIB\_STATE)

1. Les schémas relationnels précédents sont normalisés en 3FN car tout attribut dépend fonctionnellement de la clé primaire (1FN), tout attribut dépend de toute la clé (2FN) et il n’existe aucune dépendance fonctionnelle entre les attributs non-clé.
2. **Le modèle physique :**

****

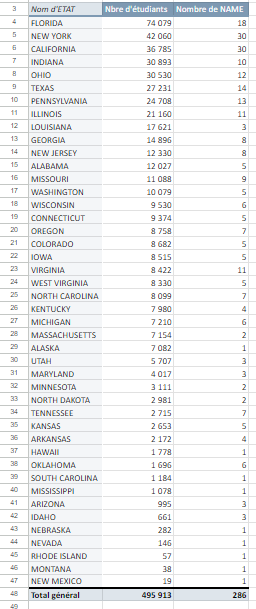
iii. **Data Analyse, lien avec EXCEL**

1. Utilisation des fonctions mathématiques simples d’EXCEL

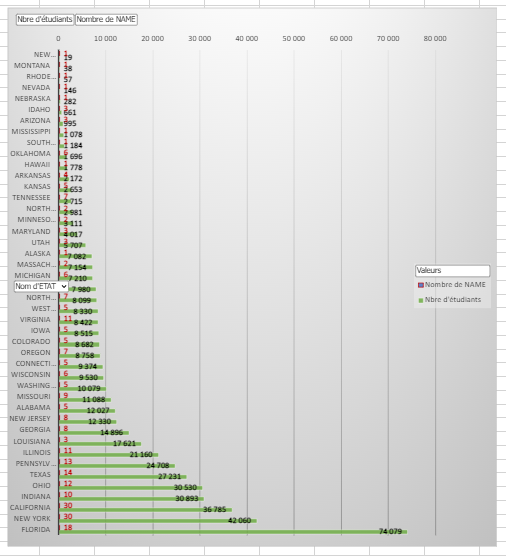
Exemple sur la colonne J “TOT\_STUDENTS”

|  |  |
| --- | --- |
| Fonction | Résultat |
| =MOYENNE(J2:J287) | 1733,96153846154 |
| =MEDIANE(J2:J287) | 492,5 |
| =MIN(J2:J287) | 2 |
| =MAX(J2:J287) | 23980 |
| =SOMME(J2:J287) | 495913 |
| =ECARTYPE(J2:J287) | 3117,35154380098 |
| =VAR(J2:J287) | 9717880,64763834 |

**Tableau croisé dynamique avec le nombre d’étudiants et d’écoles par état**



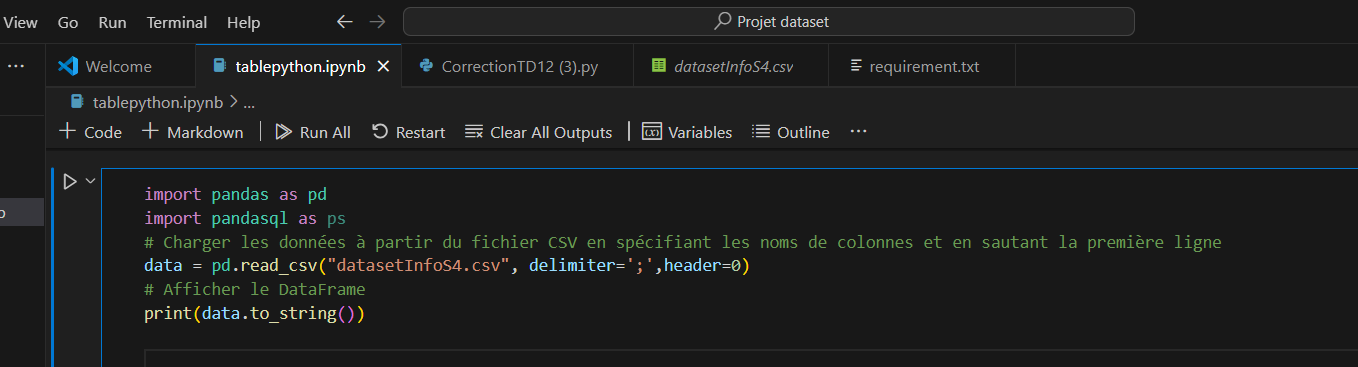
1. Histogramme avec le nombre d’étudiants et d’écoles par état

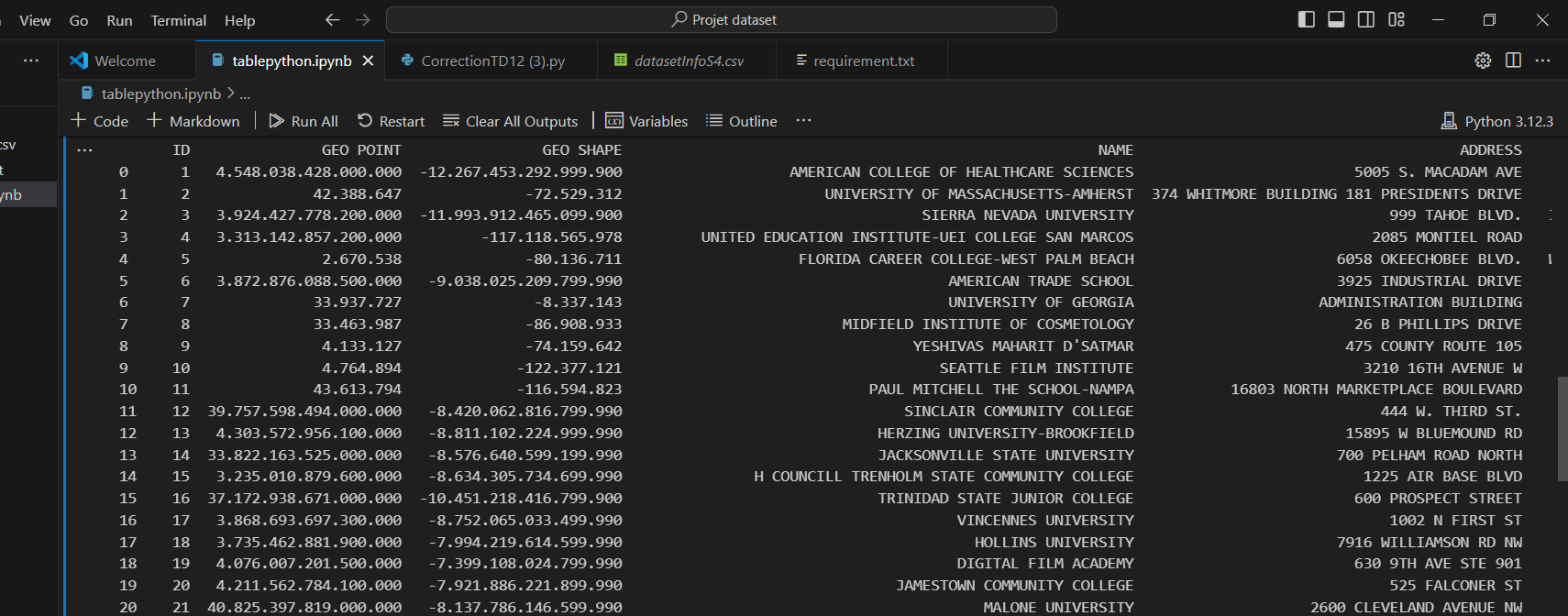


iv. **Lien avec Python**

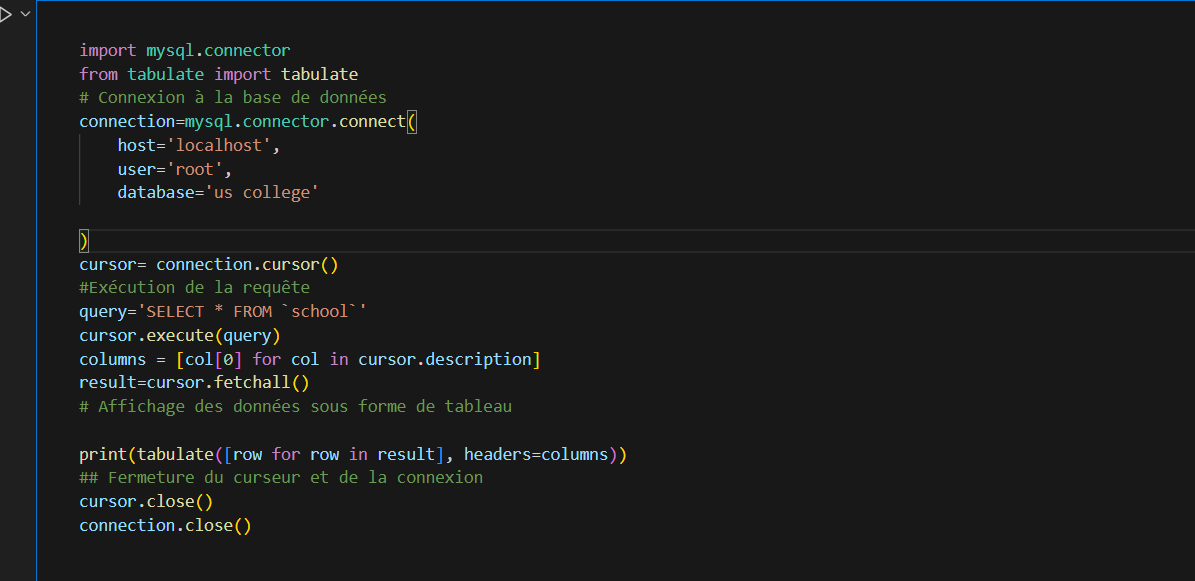
1. Création d’une 4ème table et alimentation à partir d’un programme Python :

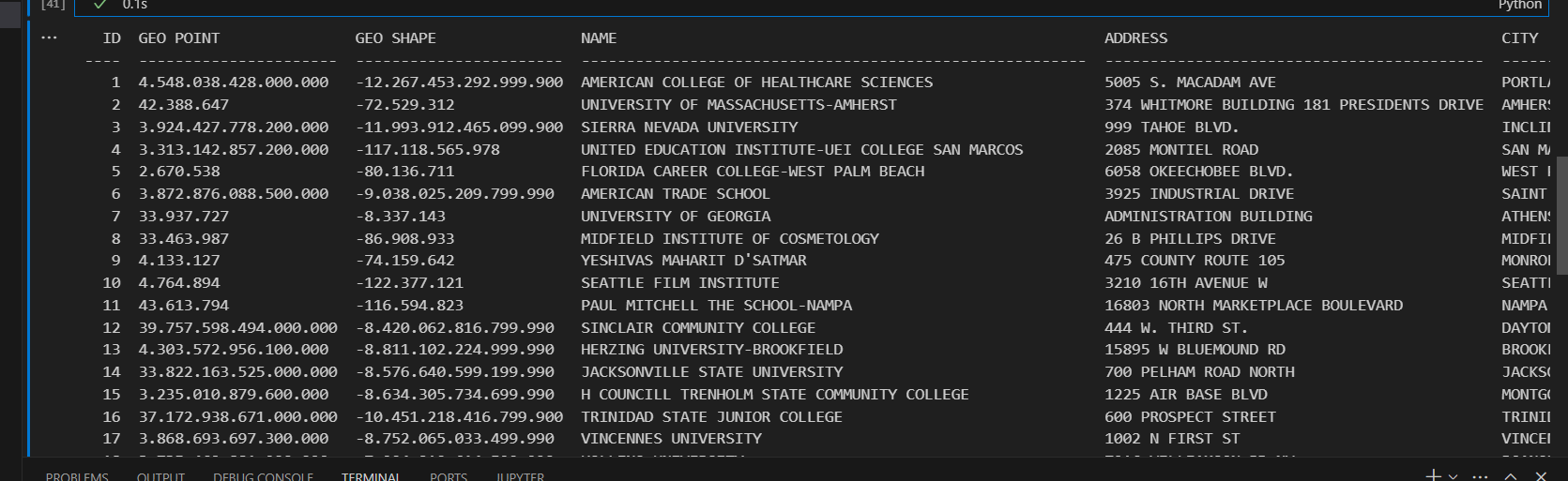
Première méthode en utilisant pandas :





Deuxième méthode en utilisant mysql.connector :

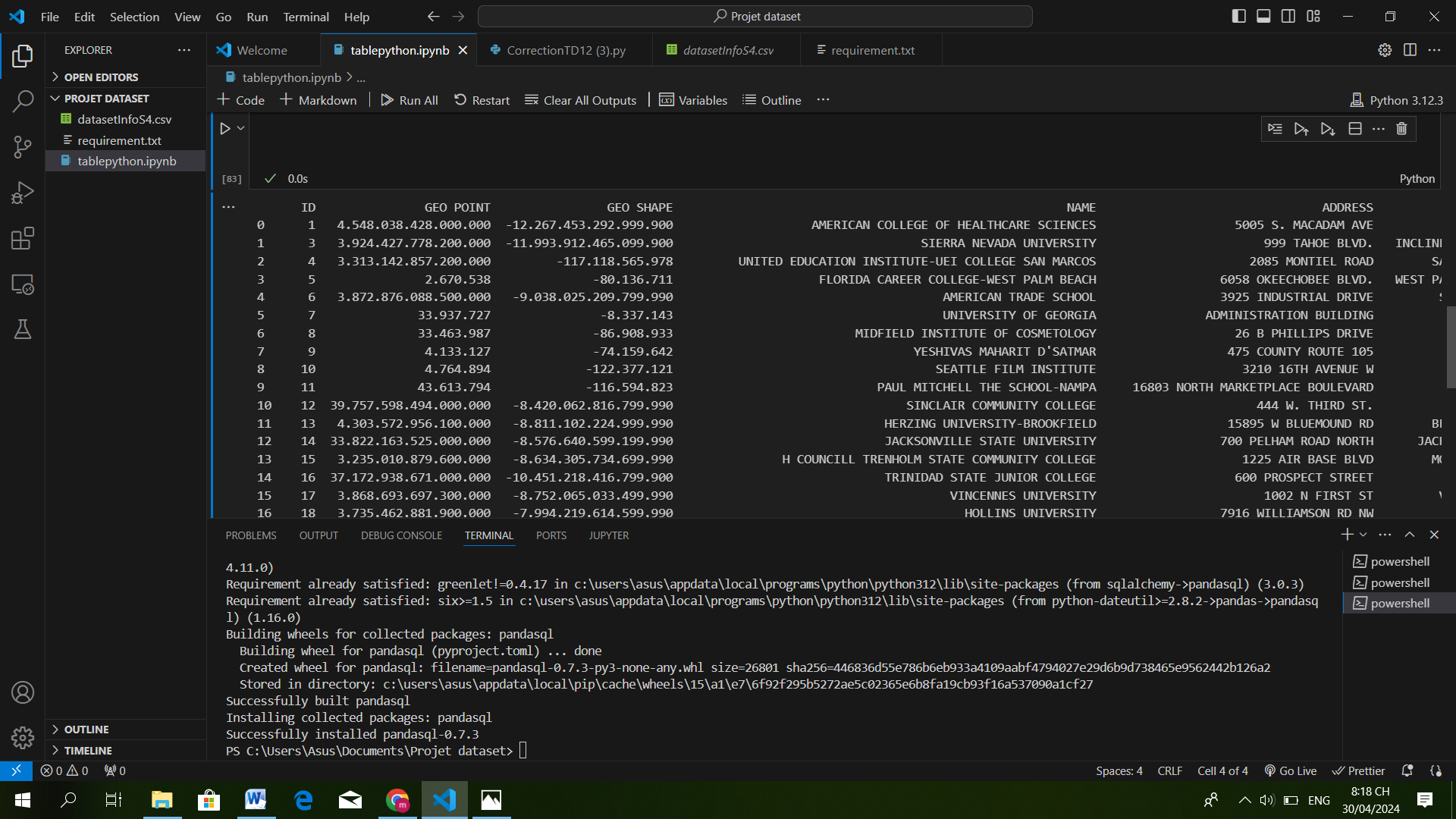




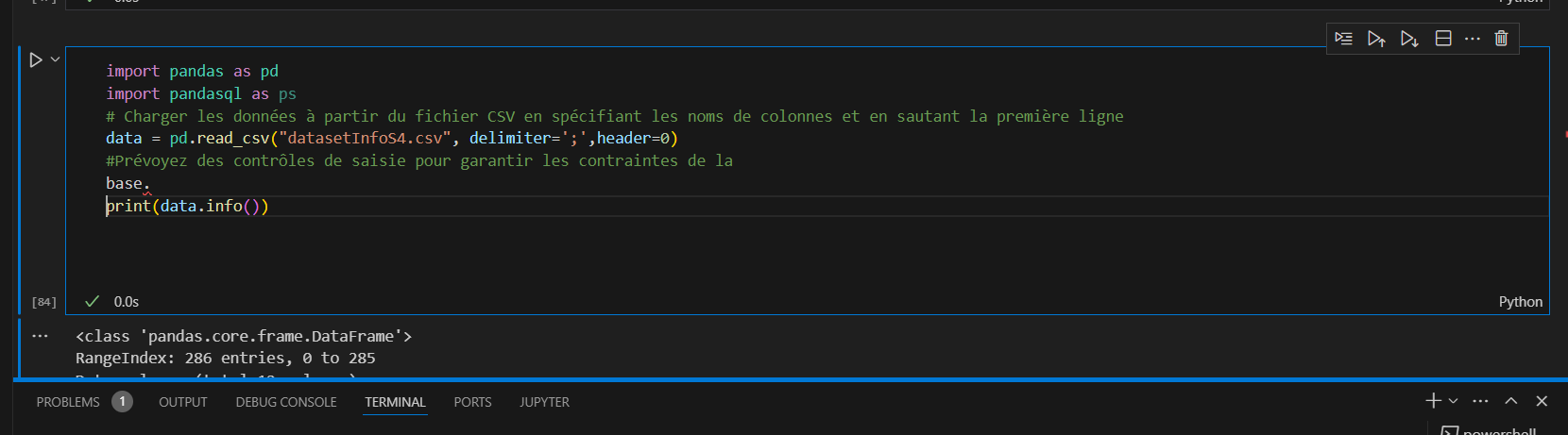
2. Suppression d’un enregistrement à partir du programme Python :

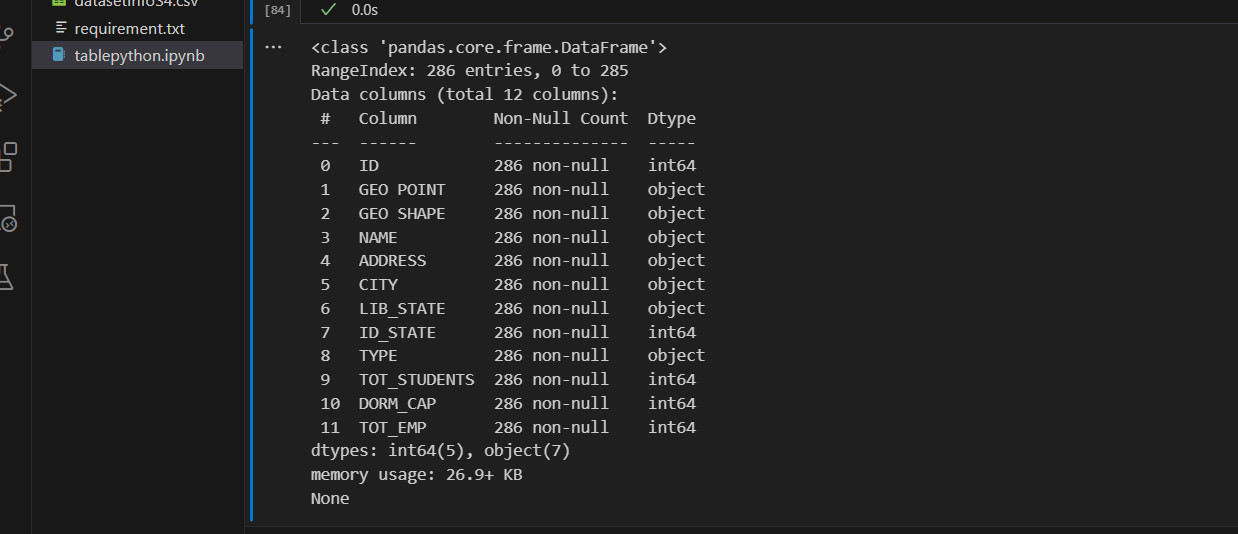
Ici on supprime l’enregistrement d’identifiant ID = 2.





**3. Contrôles de saisie pour garantir les contraintes de la base :**

****

****

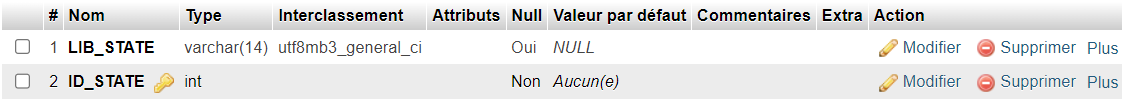
**ETAPE 2 : SQL**

i.

1. Importation des données du dataset avec phpMyAdmin dans la base de données MySQL “projet2”

**Table “state”**

**Structure :**

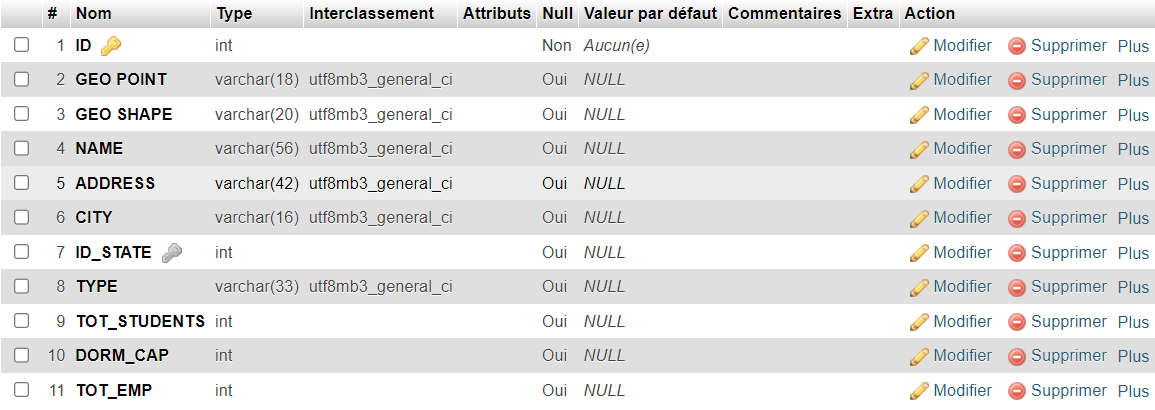


**Quelques enregistrements :**



**Table “school”**

**Structure :**



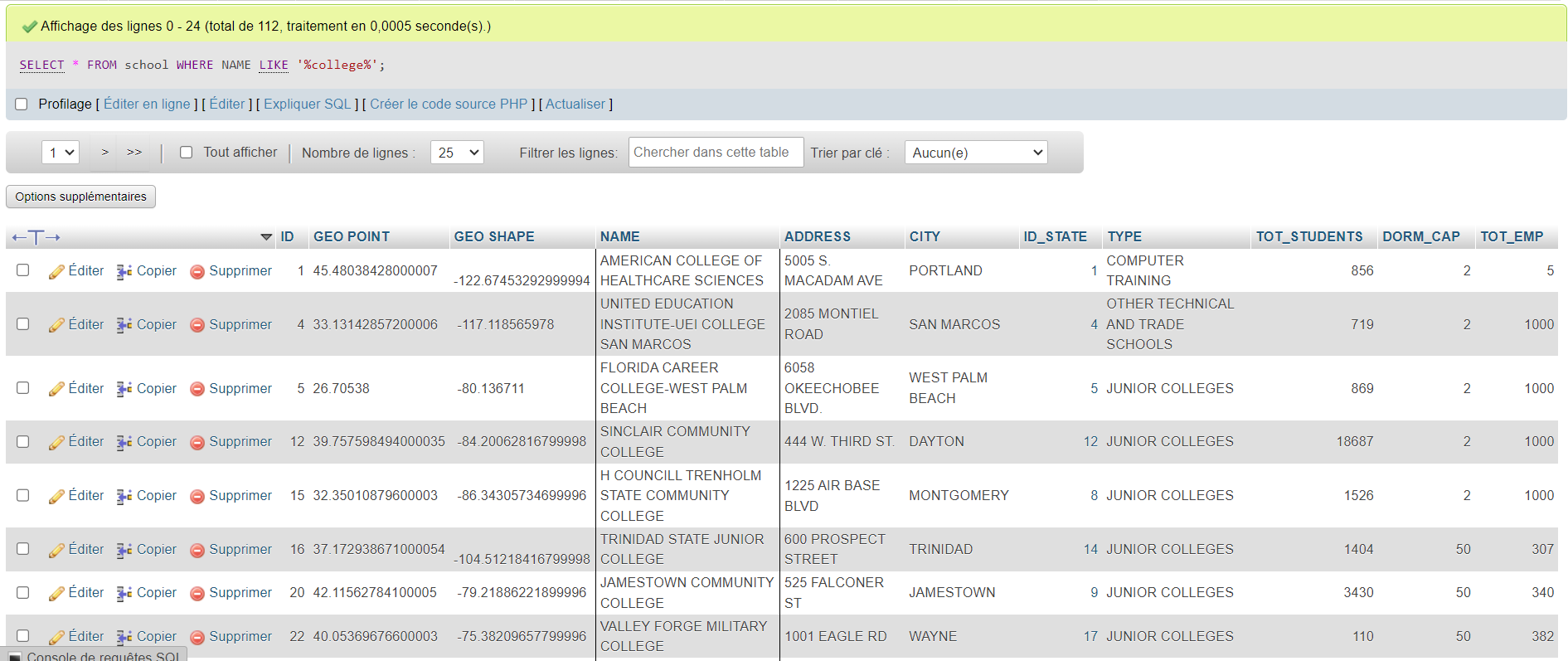
**Quelques enregistrements :**

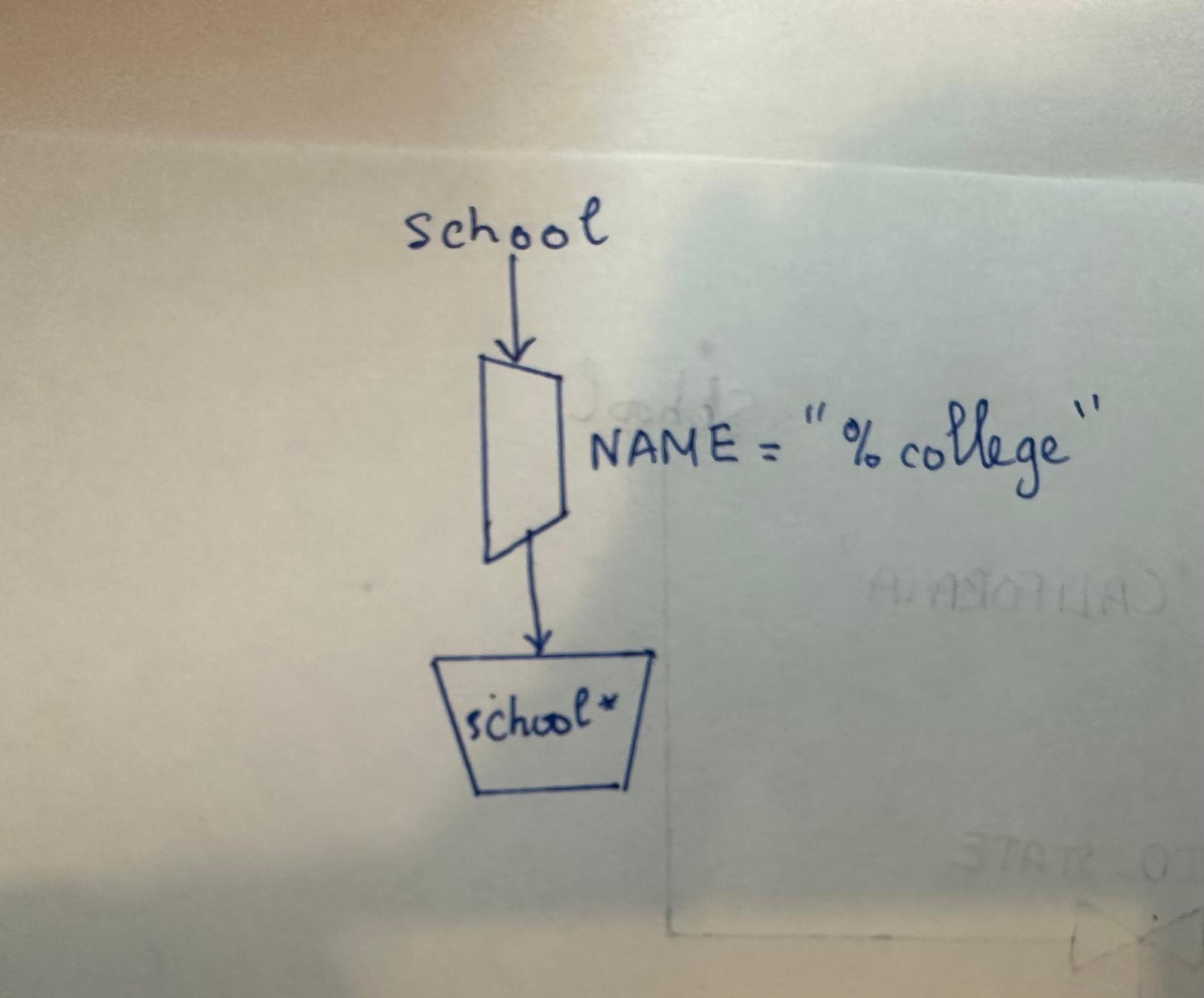


1. Introduction des requêtes SQL :
2. 3 requêtes SQL avec un “SELECT FROM WHERE” et des conditions de types différents

**Condition de type “like” :** SELECT \* FROM school WHERE NAME LIKE ‘%college%’ ;

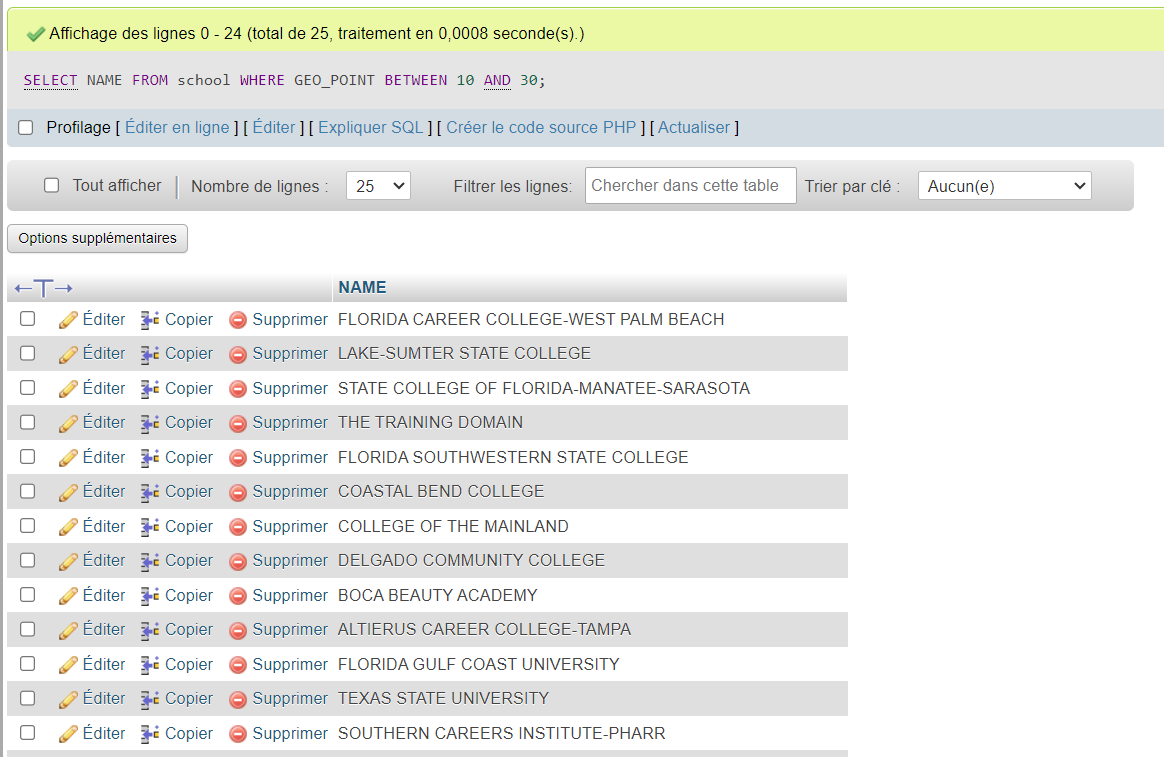
Cette requête renvoie toutes les écoles dont le nom contient “college”.

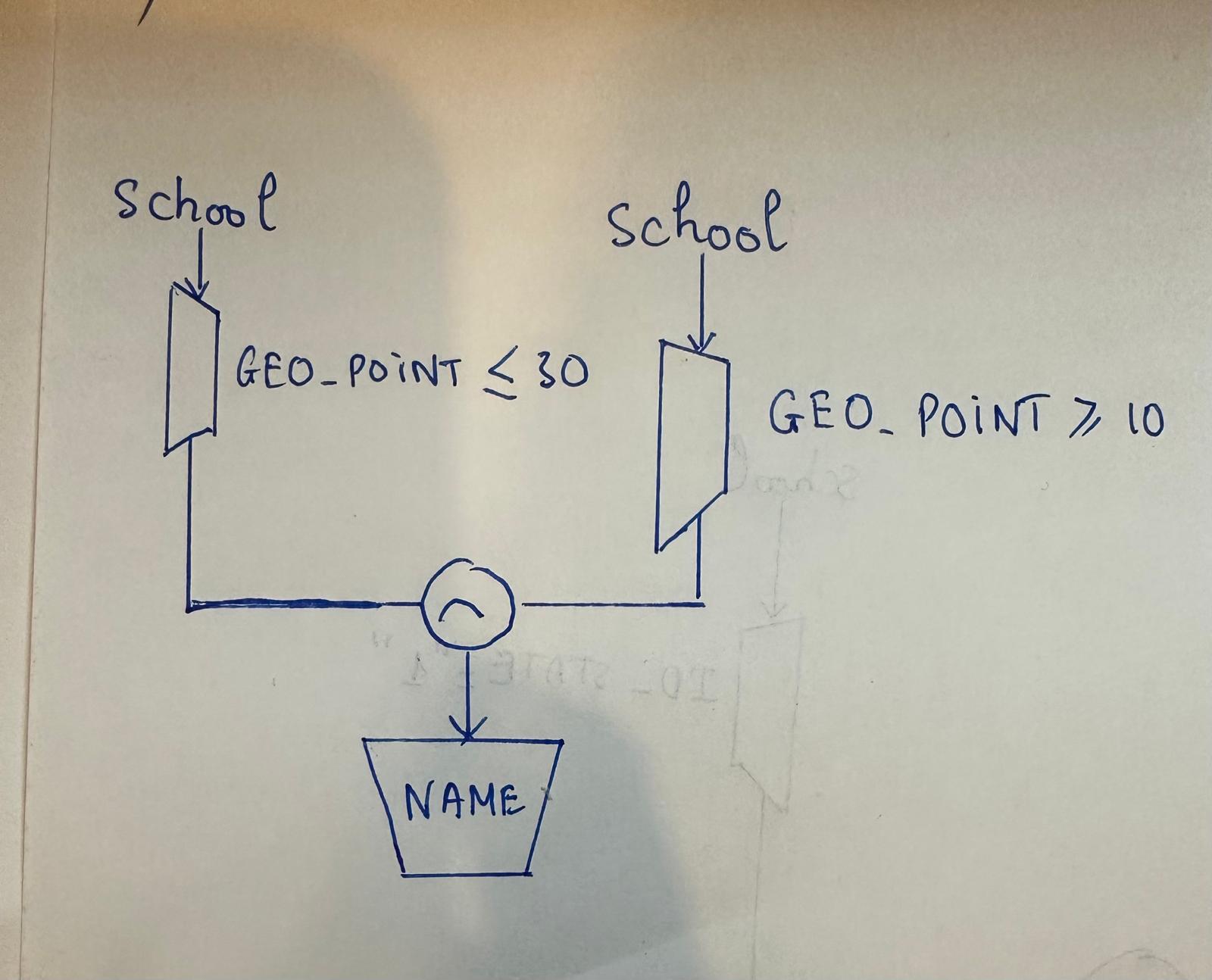




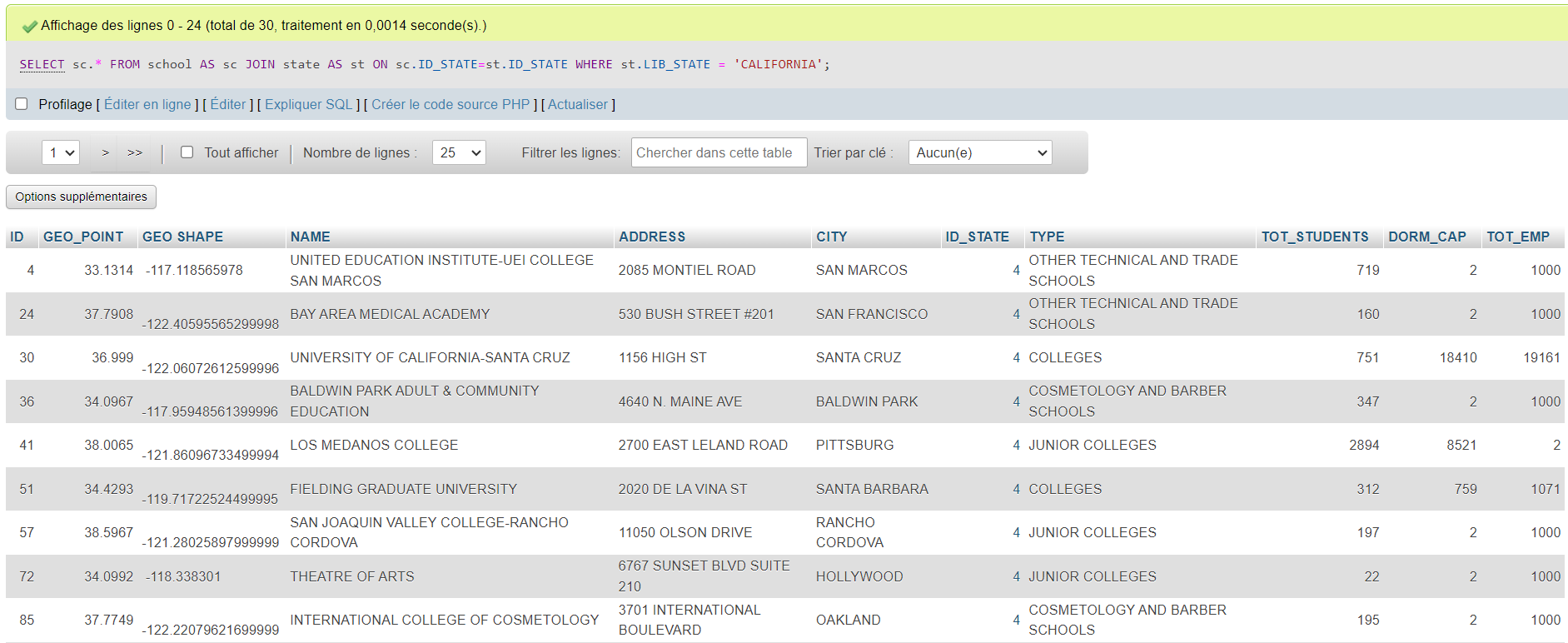
**Condition de type “between” :** SELECT NAME FROM school WHERE GEO\_POINT BETWEEN 10 AND 30 ;

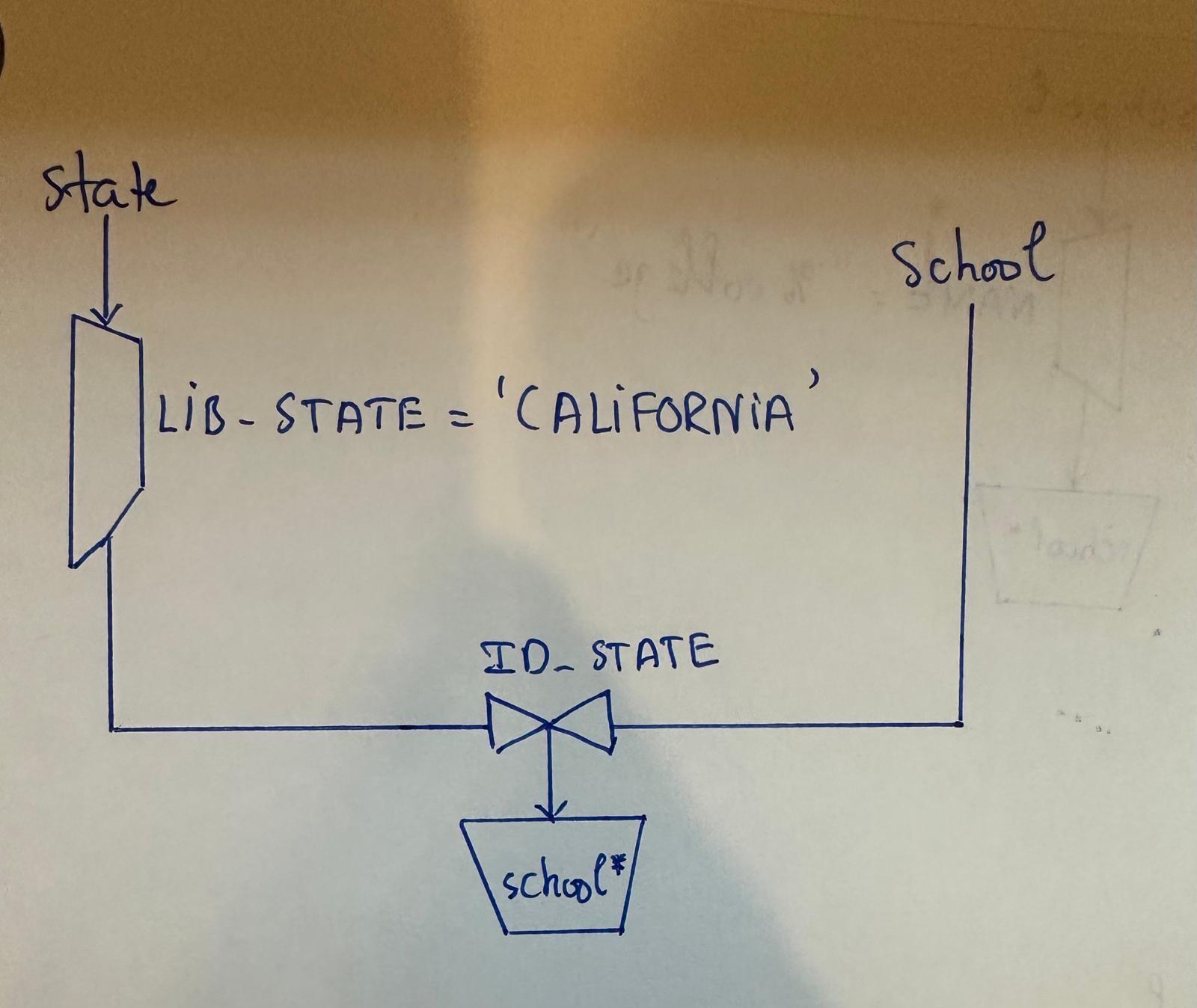
Cette requête renvoie le nom des écoles dont la coordonnée géographique est comprise dans la fourchette 10-30.





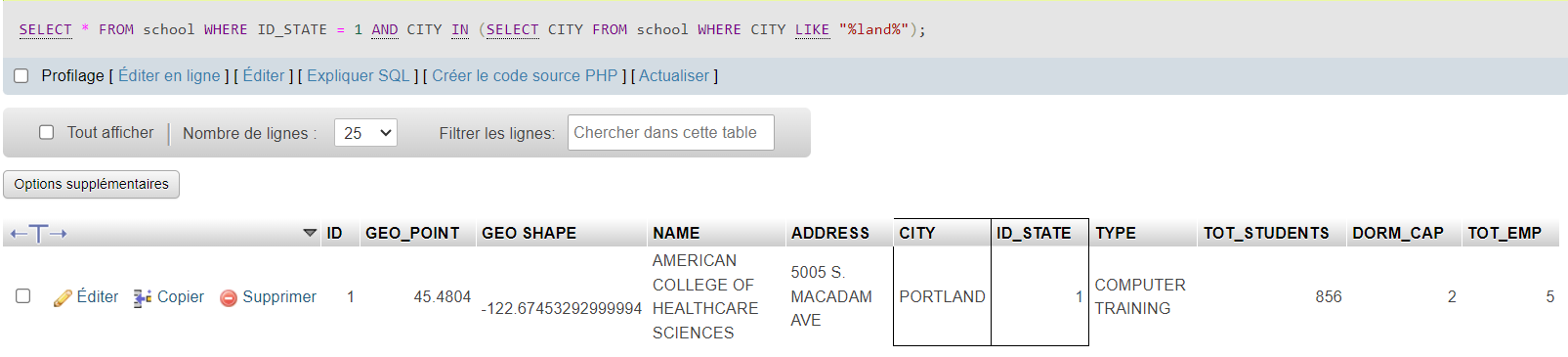
**Condition avec une jointure :** SELECT sc.\* FROM school AS sc JOIN state AS st ON sc.ID\_STATE = st.ID\_STATE WHERE st.LIB\_STATE = ‘CALIFORNIA’ ;

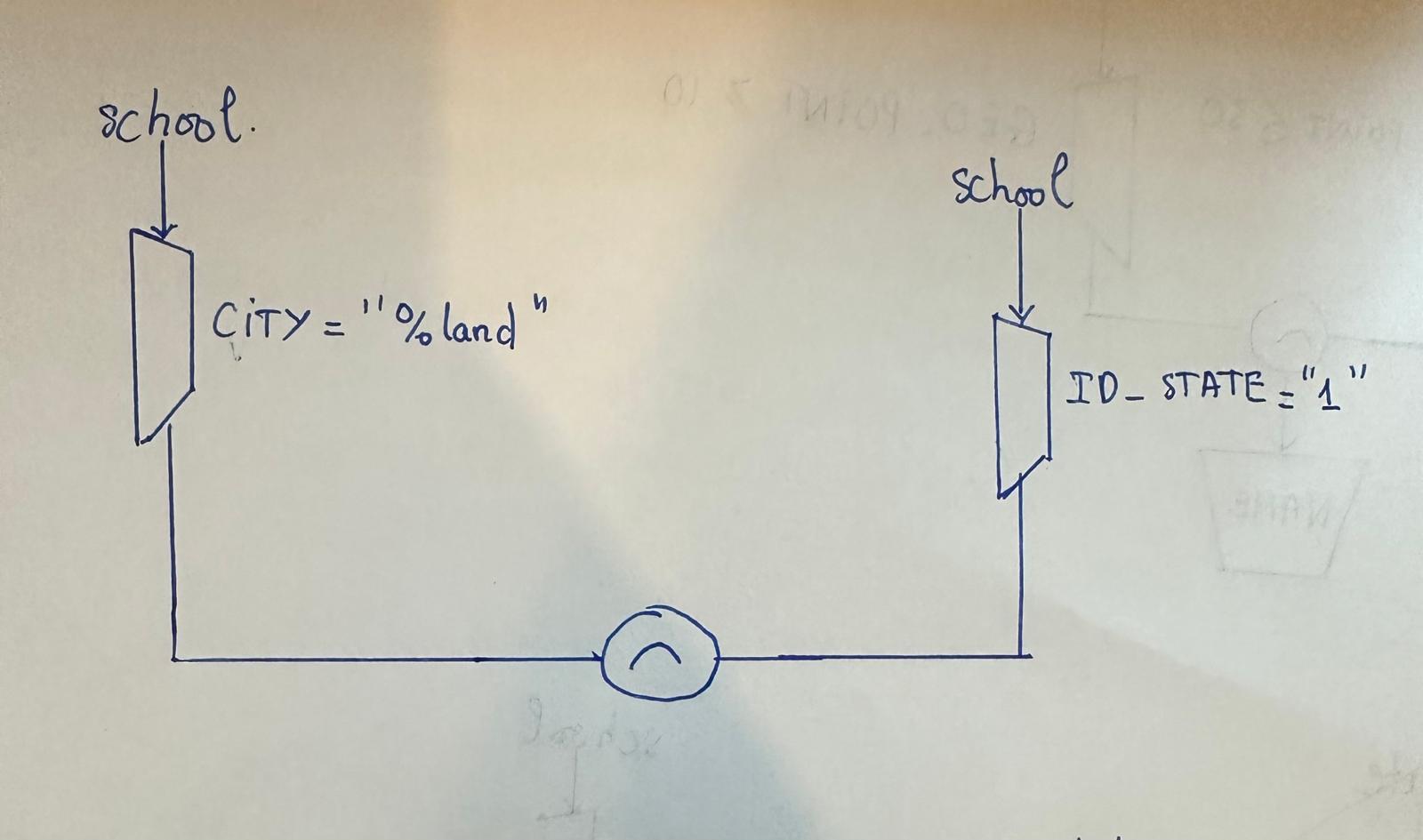




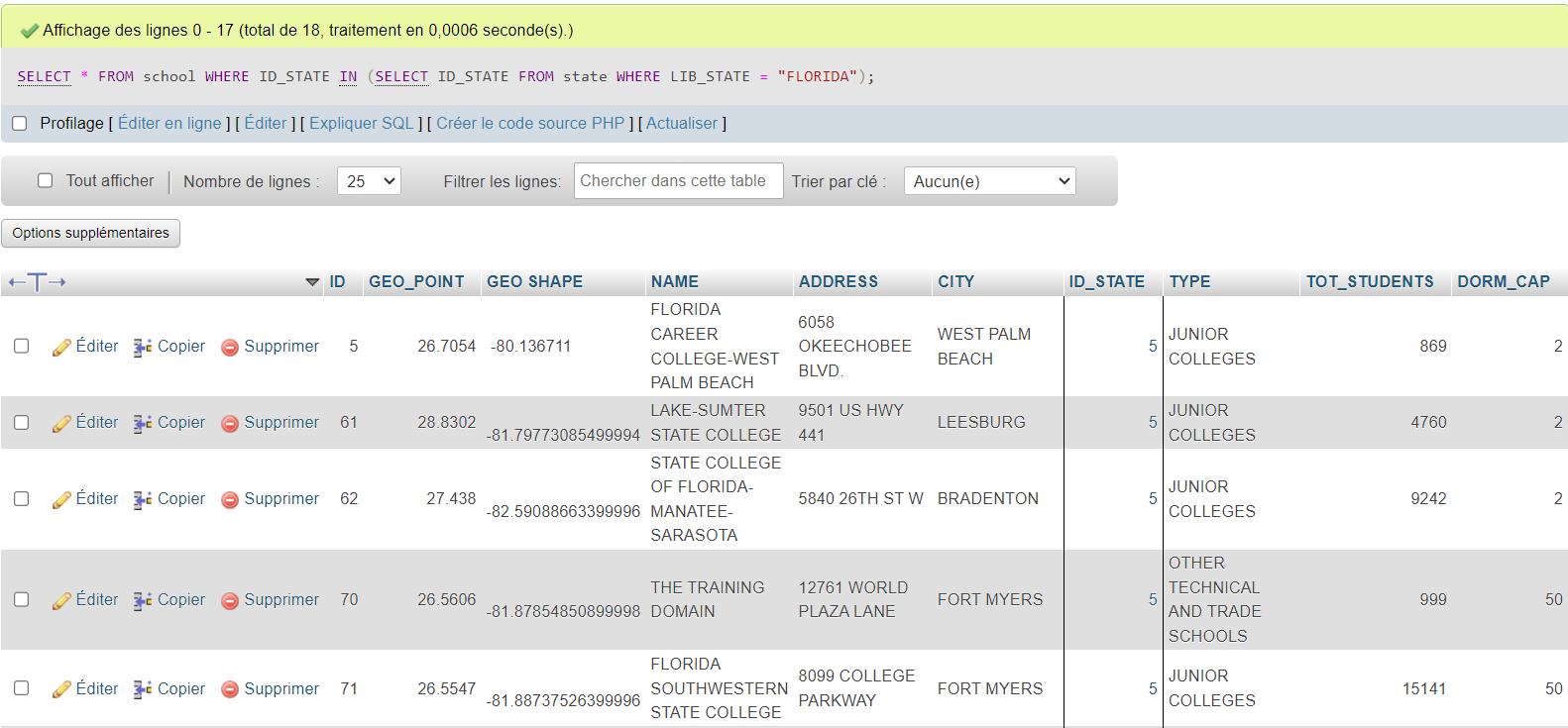
1. 2 requêtes SQL avec 2 “SELECT FROM WHERE” imbriqués utilisant des conditions de types différents

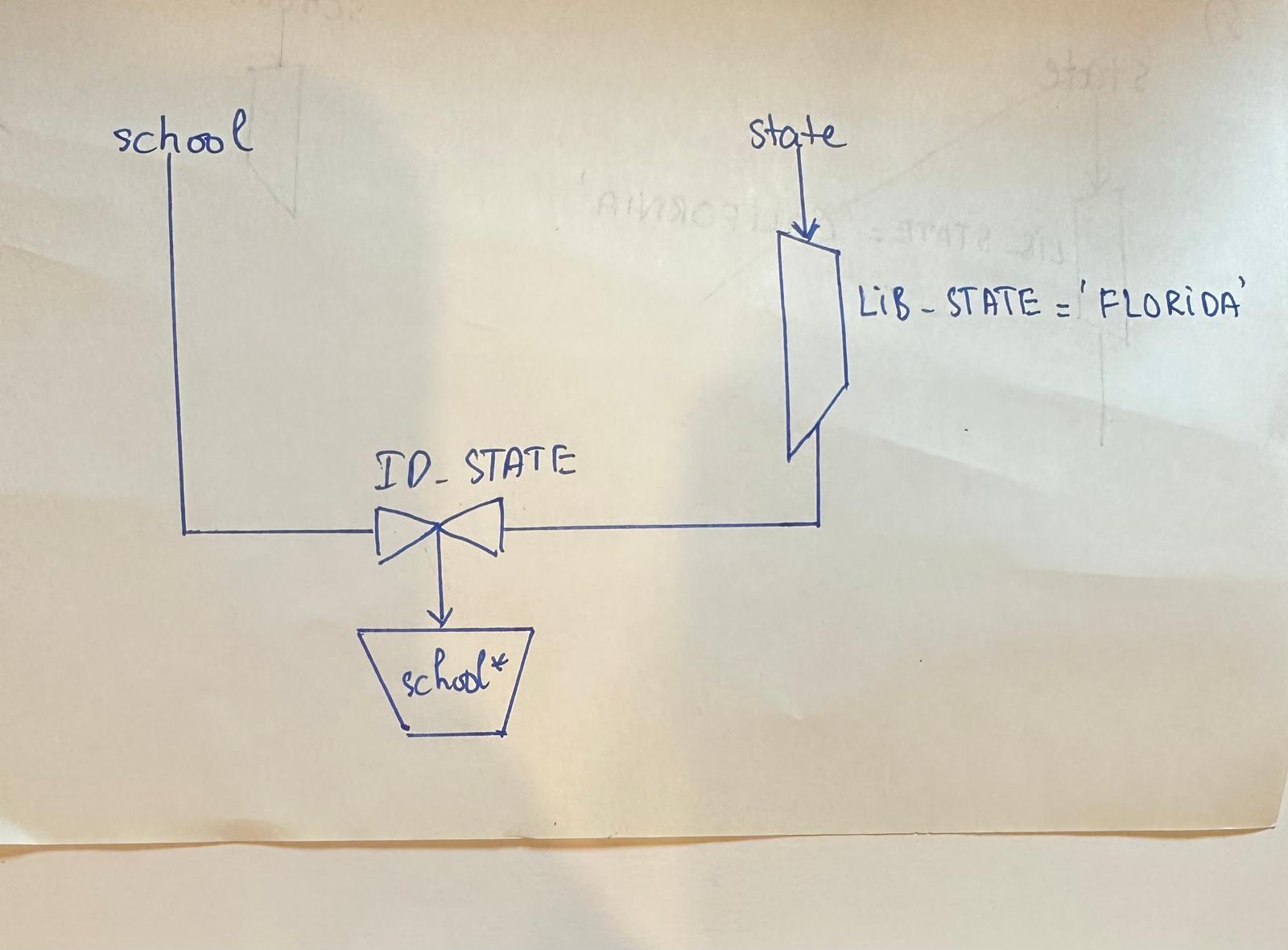
**Conditions de type |N et “like” :** SELECT \* FROM school WHERE ID\_STATE = 1 AND CITY IN (SELECT CITY FROM school WHERE CITY LIKE “%land%”) ;





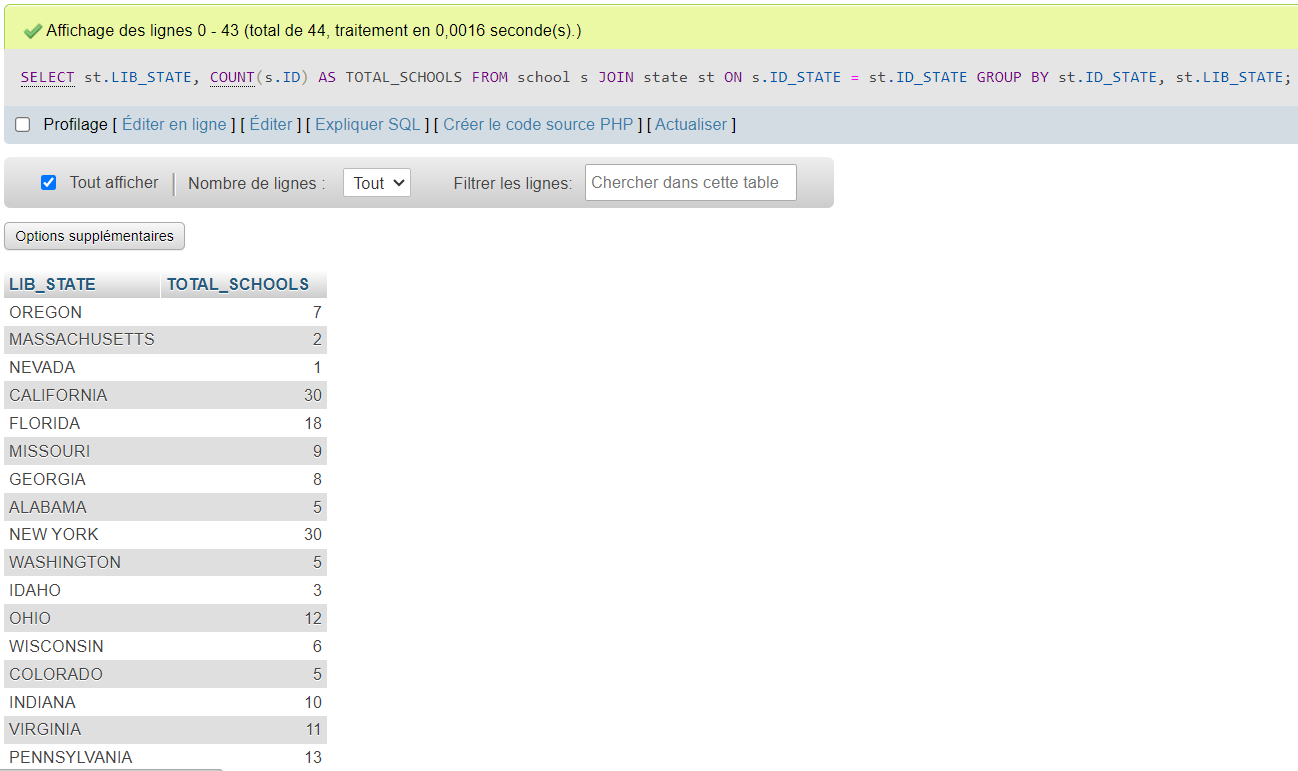
**Condition de type |N :** SELECT \* FROM school WHERE ID\_STATE IN (SELECT ID\_STATE FROM state WHERE LIB\_STATE = “FLORIDA”) ;





1. **Requête SQL avec un “GROUP BY” :**

SELECT st.LIB\_STATE, COUNT(s.ID) AS TOTAL\_SCHOOLS FROM school s JOIN state st ON s.ID\_STATE = st.ID\_STATE GROUP BY st.ID\_STATE, st.LIB\_STATE ;



1. **Requête SQL avec un “GROUP BY HAVING COUNT” :**

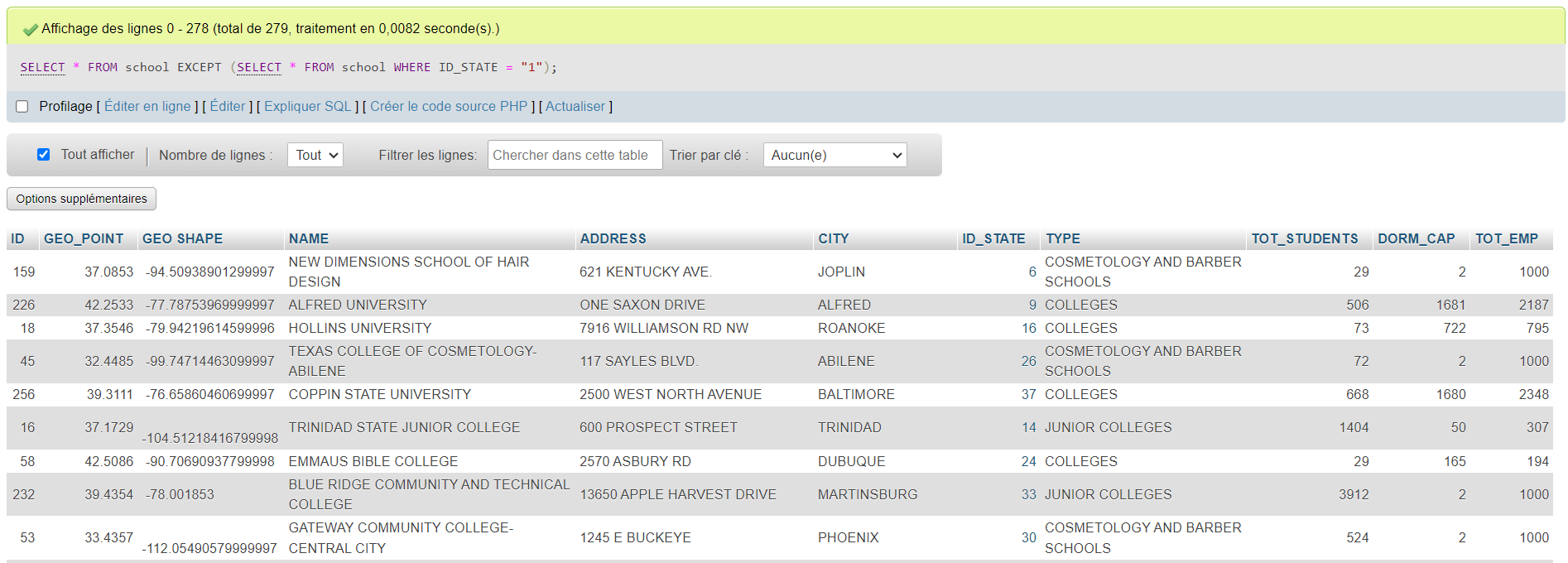
SELECT s.ID\_STATE, ST.LIB\_STATE, COUNT(s.ID) AS TOTAL\_SCHOOLS FROM school s INNER JOIN state st ON s.ID\_STATE = st.ID\_STATE GROUP BY S.ID\_STATE, ST.LIB\_STATE HAVING COUNT(s.ID) > 10 ;

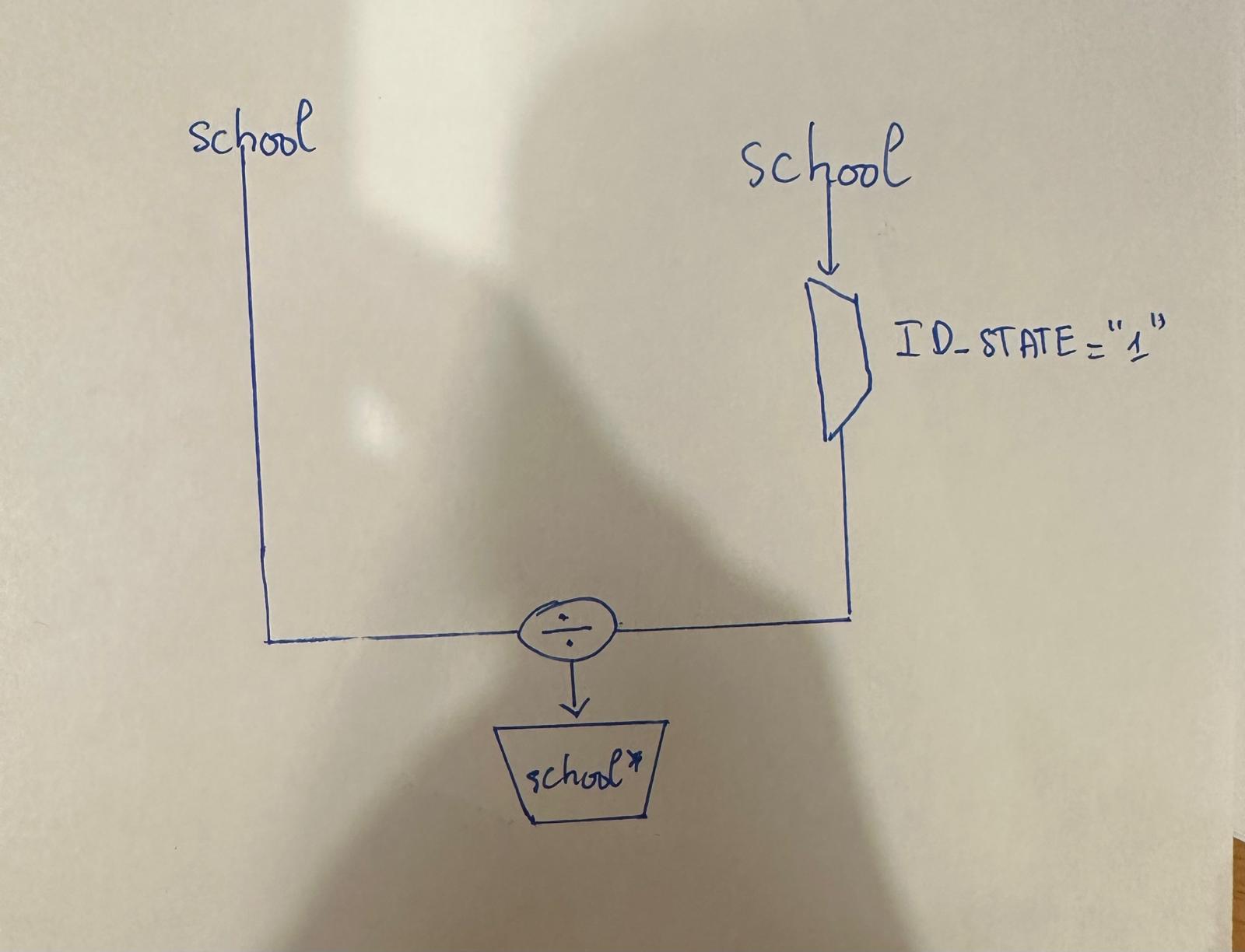


1. **Requête réalisant une différence :**

SELECT \* FROM school EXCEPT (SELECT \* FROM school WHERE ID\_STATE = “1”) ;

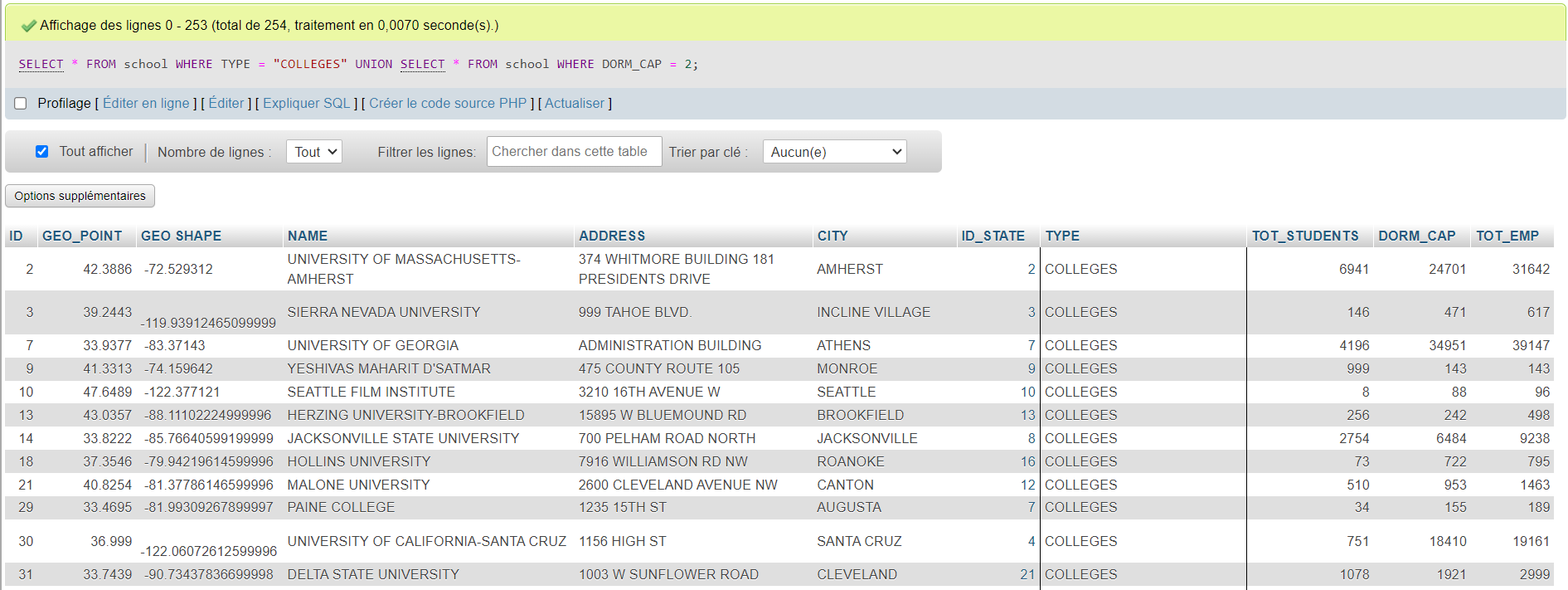
Cette requête renvoie toutes les écoles qui ne sont pas dans l’état d’identifiant 1 (OREGON).

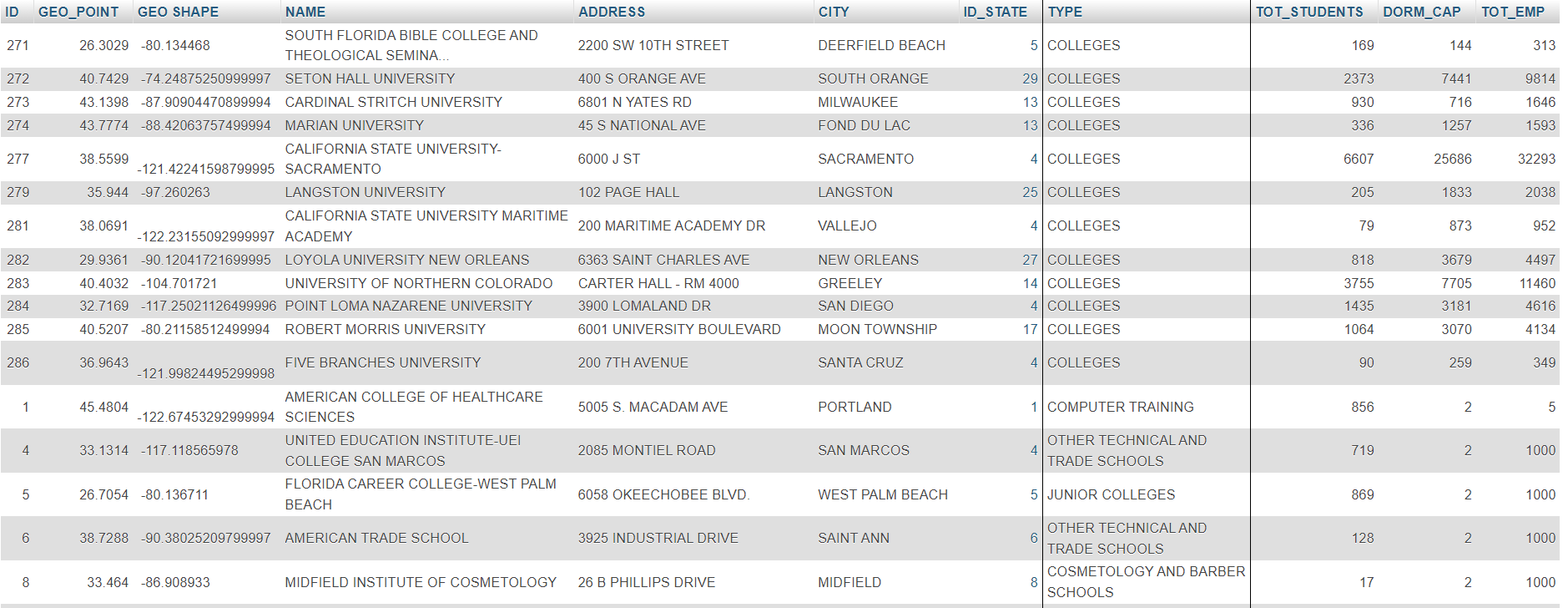


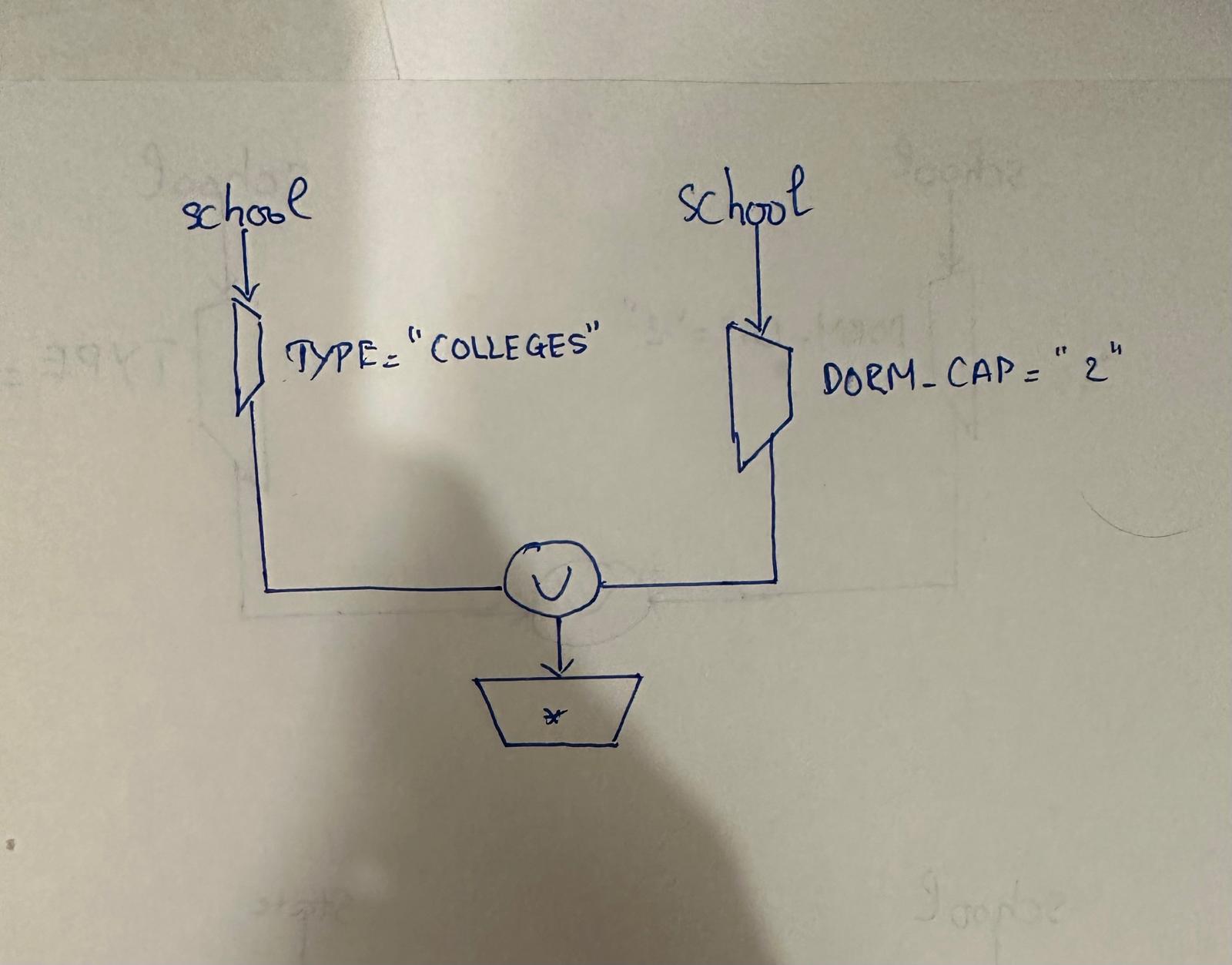


**Requête réalisant une union :**

SELECT \* FROM school WHERE TYPE = ‘COLLEGES’ UNION SELECT \* FROM school WHERE DORM\_CAP = 2 ;

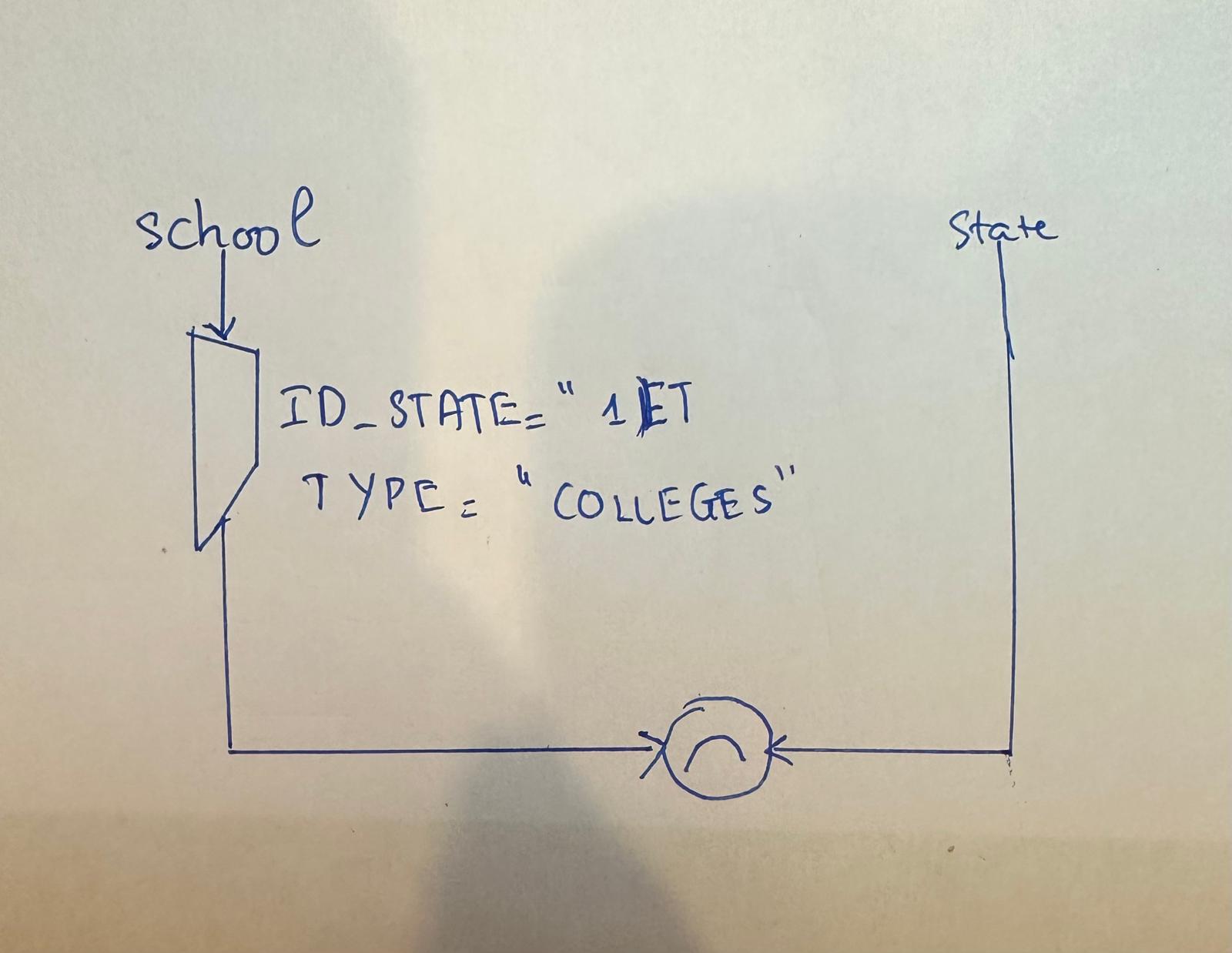






**Requête réalisant une intersection :**

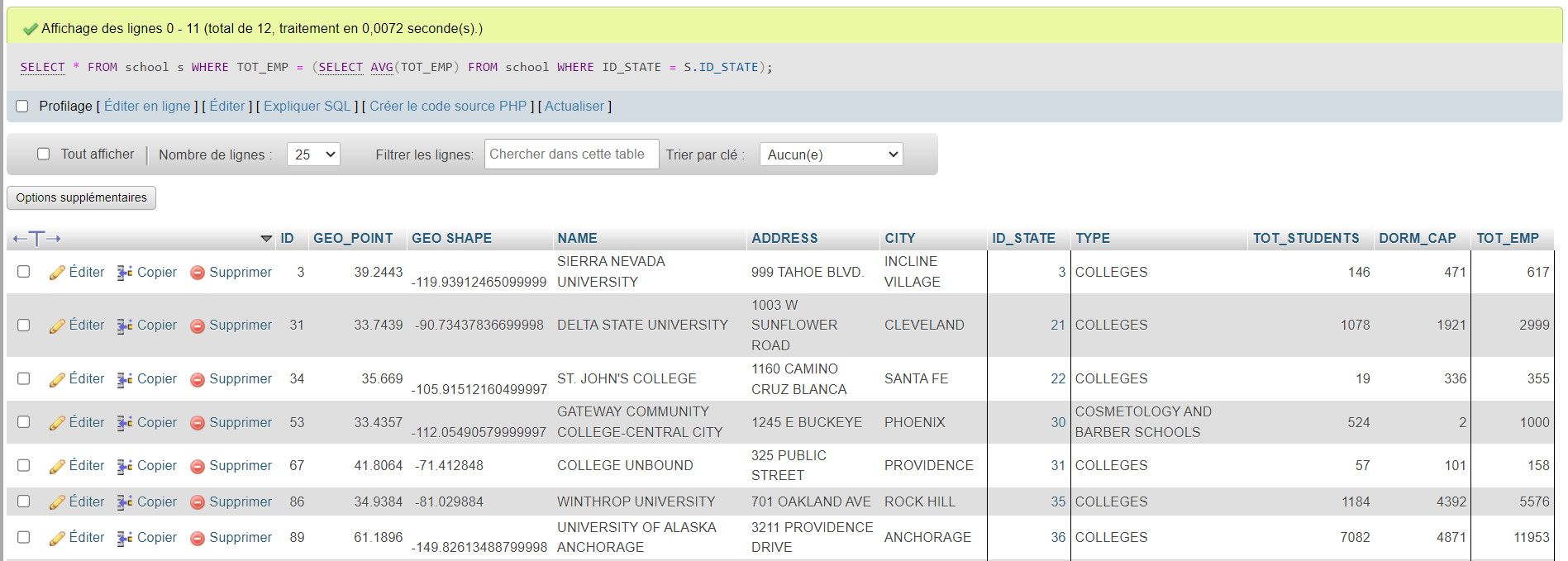
La requête suivante ne fonctionne pas car “INTERSECTION” n’est pas un opérateur standard en SQL : SELECT \* FROM school WHERE ID\_STATE = 1 INTERSECTION SELECT \* FROM school WHERE TYPE = “COLLEGES” ;



**Requête réalisant une division :**

Cette requête renvoie les écoles qui ont un nombre total d’employés égal à la moyenne par état.

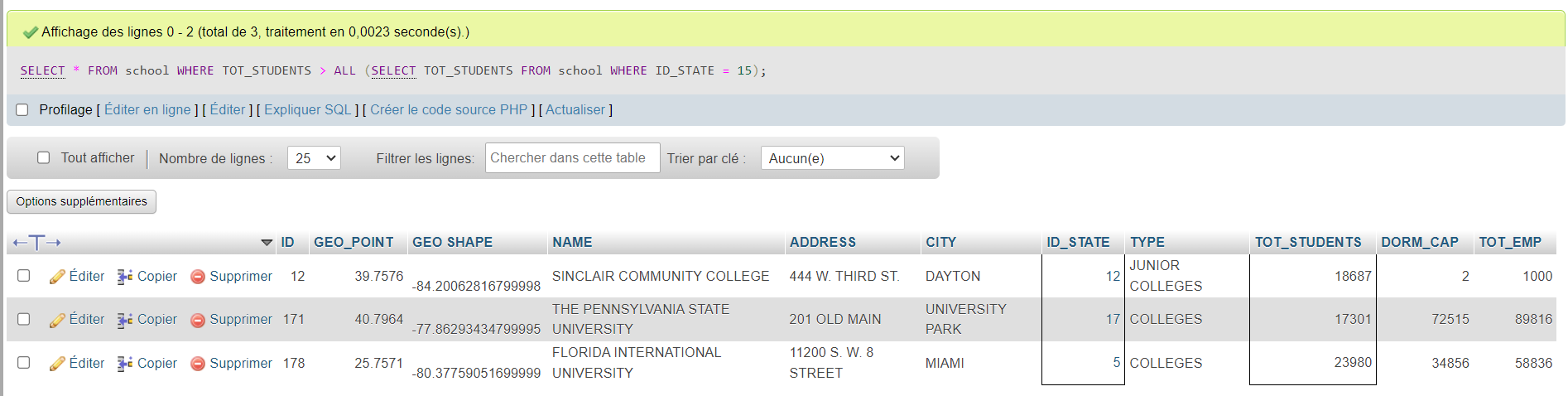
SELECT \* FROM school s WHERE TOT\_EMP = (SELECT AVG(TOT\_EMP) FROM school WHERE ID\_STATE = S.ID\_STATE)



**Requête utilisant l’opérateur “all” :**

Cette requête renvoie les écoles ayant un nombre total d’étudiants supérieur à tous les autres dans l’état d’identifiant 15 (INDIANA).

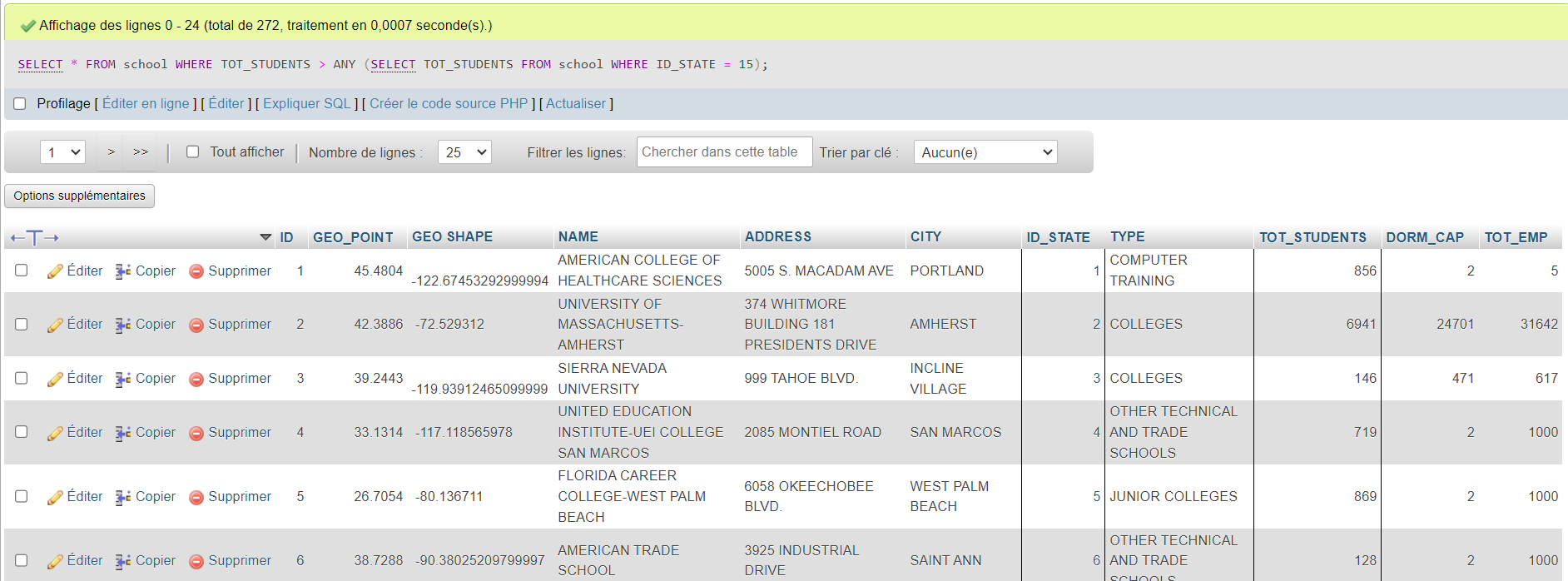
SELECT \* FROM school WHERE TOT\_STUDENTS > ALL (SELECT TOT\_STUDENTS FROM school WHERE ID\_STATE = 15) ;



**Requête utilisant l’opérateur “any” :**

Cette requête renvoie les écoles ayant un nombre total d’étudiants supérieur à au moins une autre école dans l’état d’identifiant 15 (INDIANA).

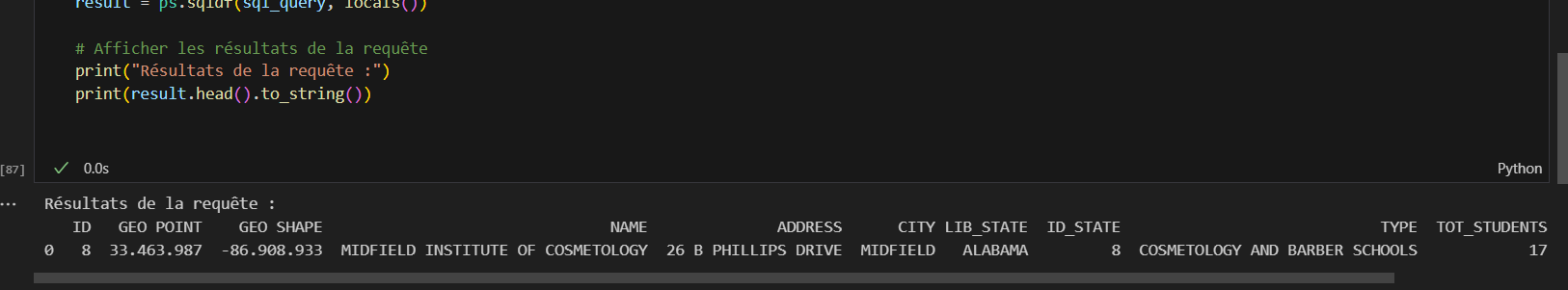
SELECT \* FROM school WHERE TOT\_STUDENTS > ANY (SELECT TOT\_STUDENTS FROM school WHERE ID\_STATE = 15) ;



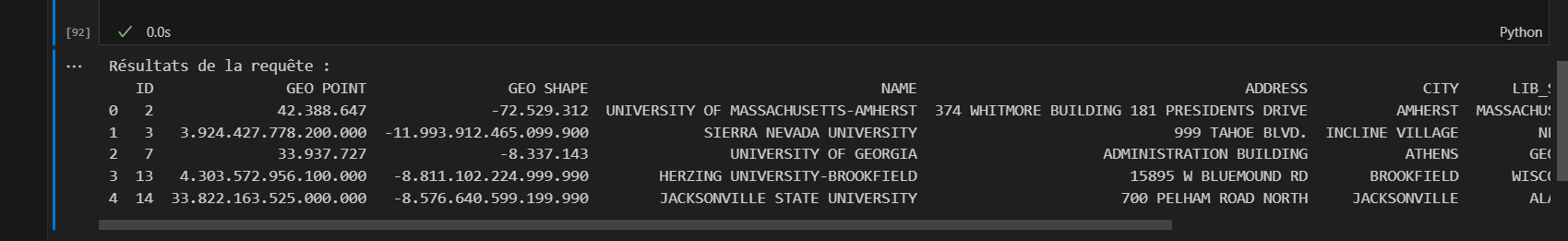
ii. **Introduction de requêtes SQL dans PYTHON :**

1. **Introduction dans le code Python de l’étape 1 d’une requête SQL du type “SELECT FROM WHERE”**

****

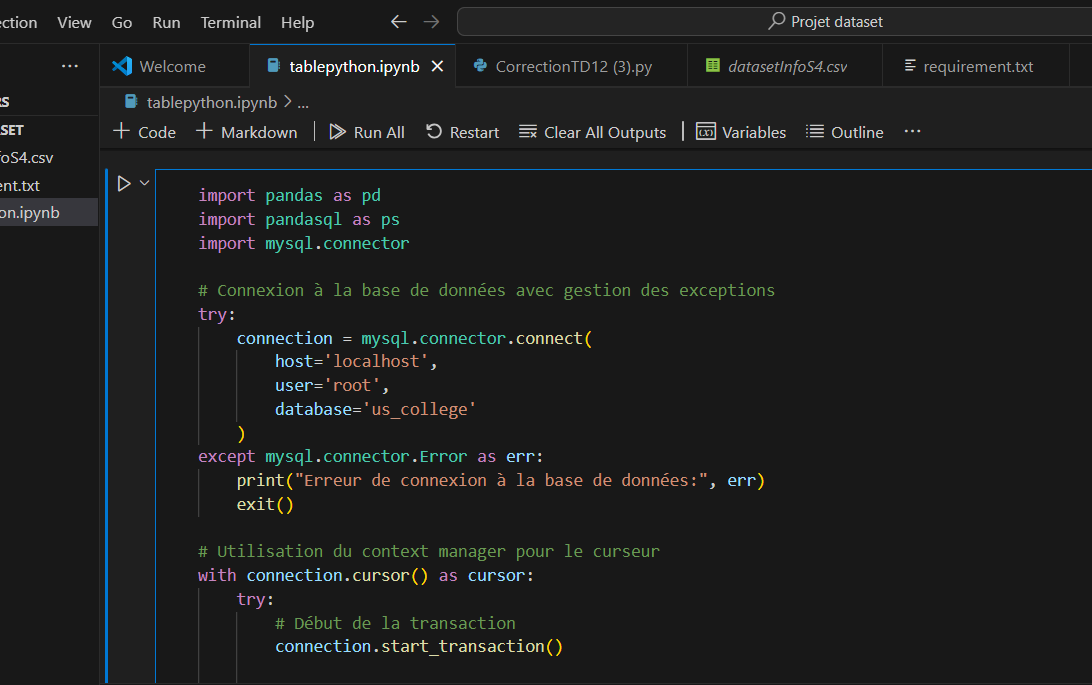
****

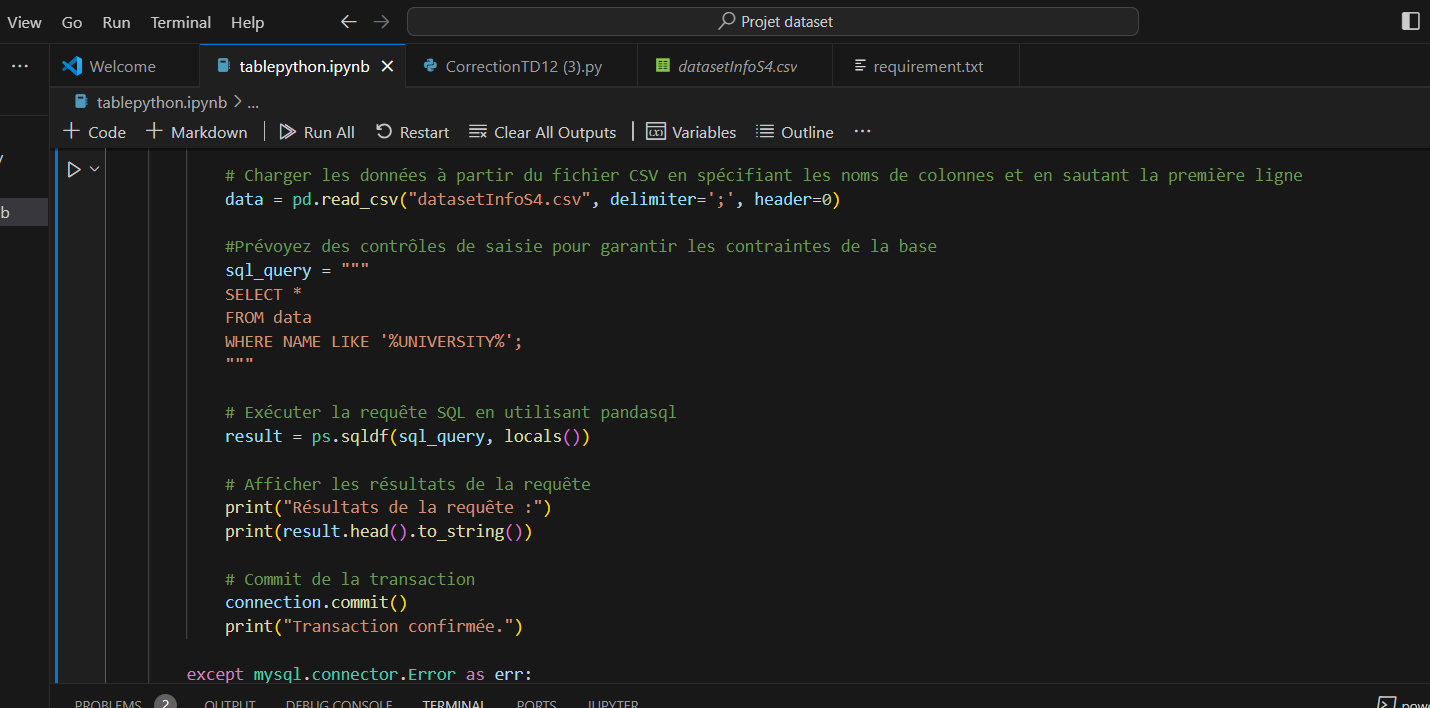
****

****

iii. **Implémentation des propriétés ACID de la base : transaction**

**1. et 2. Implémentation dans le programme Python d’un commit et d’un roll back**

****

****

