Plaksha SQL assignment

Submission details:

Please submit this as a Jupyter Notebook and a PDF of your results (both should show output). Also push your solutions to Github.

For the submision create a local database with `sqlite3` or `sqlalchemy` in a Jupyter notebook and make the queries either with a cursor object (and then print the results) or by using pandas `pd.read_sql_query()`.

When completing this homework you can experiment with SQL commands by utilizing this great online editor:

https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

There are already some tables in the online Database, namely:

`Categories, Employees, OrderDetails, Orders, Products, Shippers, and Suppliers.`

If you want you can drop them by running `DROP TABLE [table-name];` (or just keep them).

Exercises:

First create a table called students. It has the columns: 'student_id', 'name', 'major', 'gpa' and 'enrollment_date' We will use a new form of `CREATE TABLE` expression to produce this table.

Note that you can improve this and are welcome to do so -- e.g. by specifying for example a PRIMARY KEY and a FOREIGN KEY in Q2 :)

```
CREATE TABLE students AS
    SELECT 1 AS student_id, "John" AS name, "Computer Science" AS
major, 3.5 AS gpa, "01-01-2022" AS enrollment_date UNION
    SELECT 2, "Jane", "Physics", 3.8, "01-02-2022" UNION
    SELECT 3, "Bob", "Engineering", 3.0, "01-03-2022" UNION
    SELECT 4, "Samantha", "Physics", 3.9, "01-04-2022" UNION
    SELECT 5, "James", "Engineering", 3.7, "01-05-2022" UNION
    SELECT 6, "Emily", "Computer Science", 3.6, "01-06-2022" UNION
    SELECT 7, "Michael", "Computer Science", 3.2, "01-07-2022" UNION
    SELECT 8, "Jessica", "Engineering", 3.8, "01-08-2022" UNION
```

```
    SELECT 9, "Jacob", "Physics", 3.4, "01-09-2022" UNION
    SELECT 10, "Ashley", "Physics", 3.9, "01-10-2022";
```

Q1 Simple SELECTS (on the students table)

1. SELECT all records in the table.
2. SELECT students whose major is "Computer Science".
3. SELECT all unique majors (use SELECT DISTINCT) and order them by name, descending order (i.e. Physics first).
4. SELECT all students that have an 'e' in their name and order them by gpa in ascending order.

```
# Import libraries
import sqlite3
import pandas as pd

# Connect to the database
conn = sqlite3.connect("student_db.db")
cursor = conn.cursor()

# Create students table
cursor.execute("""
CREATE TABLE students (
    student_id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    major TEXT NOT NULL,
    gpa REAL NOT NULL,
    enrollment_date TEXT NOT NULL
);
""")
```

```
<sqlite3.Cursor at 0x7fb812e95f80>
```

```
# Insert data into students table
cursor.execute("""
INSERT INTO students (student_id, name, major, gpa, enrollment_date)
VALUES
    (1, 'John', 'Computer Science', 3.5, '01-01-2022'),
    (2, 'Jane', 'Physics', 3.8, '01-02-2022'),
    (3, 'Bob', 'Engineering', 3.0, '01-03-2022'),
    (4, 'Samantha', 'Physics', 3.9, '01-04-2022'),
    (5, 'James', 'Engineering', 3.7, '01-05-2022'),
    (6, 'Emily', 'Computer Science', 3.6, '01-06-2022'),
    (7, 'Michael', 'Computer Science', 3.2, '01-07-2022'),
    (8, 'Jessica', 'Engineering', 3.8, '01-08-2022'),
    (9, 'Jacob', 'Physics', 3.4, '01-09-2022'),
    (10, 'Ashley', 'Physics', 3.9, '01-10-2022');
""")
```

```
<sqlite3.Cursor at 0x7fb812e95f80>
```

```
# committing the current transaction.
conn.commit()
```

1. SELECT all records in the table.

```
print("1. SELECT all records in the table:")
cursor.execute("SELECT * FROM students;")
print(cursor.fetchall())
```

```
1. SELECT all records in the table:
[(1, 'John', 'Computer Science', 3.5, '01-01-2022'), (2, 'Jane',
'Physics', 3.8, '01-02-2022'), (3, 'Bob', 'Engineering', 3.0, '01-03-
2022'), (4, 'Samantha', 'Physics', 3.9, '01-04-2022'), (5, 'James',
'Engineering', 3.7, '01-05-2022'), (6, 'Emily', 'Computer Science',
3.6, '01-06-2022'), (7, 'Michael', 'Computer Science', 3.2, '01-07-
2022'), (8, 'Jessica', 'Engineering', 3.8, '01-08-2022'), (9, 'Jacob',
'Physics', 3.4, '01-09-2022'), (10, 'Ashley', 'Physics', 3.9, '01-10-
2022')]
```

2. SELECT students whose major is "Computer Science".

```
print("2. SELECT students whose major is 'Computer Science':")
cursor.execute("SELECT * FROM students WHERE major = 'Computer
Science';")
print(cursor.fetchall())
```

```
2. SELECT students whose major is 'Computer Science':
[(1, 'John', 'Computer Science', 3.5, '01-01-2022'), (6, 'Emily',
'Computer Science', 3.6, '01-06-2022'), (7, 'Michael', 'Computer
Science', 3.2, '01-07-2022')]
```

3. SELECT all unique majors and order them by name, descending order

```
print("3. SELECT all unique majors and order them by name, descending
order:")
cursor.execute("""
SELECT DISTINCT major
FROM students
ORDER BY major DESC;
""")
print(cursor.fetchall())
```

```
3. SELECT all unique majors and order them by name, descending order:
[('Physics',), ('Engineering',), ('Computer Science',)]
```

4. SELECT all students that have an 'e' in their name and order them by gpa in ascending order

```
print("4. SELECT all students that have an 'e' in their name and order
them by gpa in ascending order:")
cursor.execute("""
SELECT *
FROM students
WHERE name LIKE '%e%'
ORDER BY gpa;
```

```
""")
print(cursor.fetchall())
```

```
4. SELECT all students that have an 'e' in their name and order them
by gpa in ascending order:
[(7, 'Michael', 'Computer Science', 3.2, '01-07-2022'), (6, 'Emily',
'Computer Science', 3.6, '01-06-2022'), (5, 'James', 'Engineering',
3.7, '01-05-2022'), (2, 'Jane', 'Physics', 3.8, '01-02-2022'), (8,
'Jessica', 'Engineering', 3.8, '01-08-2022'), (10, 'Ashley',
'Physics', 3.9, '01-10-2022')]
```

Q2 Joins

Create a new table called courses, which indicates the courses taken by the students.

Create the table by running:

```
CREATE TABLE courses AS
    SELECT 1 AS course_id, "Python programming" AS course_name, 1 AS
student_id, "A" AS grade UNION
    SELECT 2, "Data Structures", 2, "B" UNION
    SELECT 3, "Database Systems", 3, "B" UNION
    SELECT 1, "Python programming", 4, "A" UNION
    SELECT 4, "Quantum Mechanics", 5, "C" UNION
    SELECT 1, "Python programming", 6, "F" UNION
    SELECT 2, "Data Structures", 7, "C" UNION
    SELECT 3, "Database Systems", 8, "A" UNION
    SELECT 4, "Quantum Mechanics", 9, "A" UNION
    SELECT 2, "Data Structures", 10, "F";
```

1. COUNT the number of unique courses.
2. JOIN the tables students and courses and COUNT the number of students with the major Computer Science taking the course Python programming.
3. JOIN the tables students and courses and select the students who have grades higher than "C", only show their name, major, gpa, course_name and grade.

```
# Create the courses table
conn.execute('''
CREATE TABLE courses AS
    SELECT 1 AS course_id, "Python programming" AS course_name, 1 AS
student_id, "A" AS grade UNION
    SELECT 2, "Data Structures", 2, "B" UNION
    SELECT 3, "Database Systems", 3, "B" UNION
    SELECT 1, "Python programming", 4, "A" UNION
    SELECT 4, "Quantum Mechanics", 5, "C" UNION
    SELECT 1, "Python programming", 6, "F" UNION
    SELECT 2, "Data Structures", 7, "C" UNION
    SELECT 3, "Database Systems", 8, "A" UNION
    SELECT 4, "Quantum Mechanics", 9, "A" UNION
    SELECT 2, "Data Structures", 10, "F";
''')
```

```
<sqlite3.Cursor at 0x7fb812e34ab0>
```

```
# committing the current transaction.
conn.commit()
```

1. COUNT the number of unique courses

```
cursor = conn.execute('SELECT COUNT(DISTINCT course_name) FROM
courses;')
print("Number of unique courses:", cursor.fetchone()[0])
```

```
Number of unique courses: 4
```

2. JOIN the tables students and courses and COUNT the number of students with the major Computer Science taking the course Python programming

```
cursor = conn.execute('''
SELECT COUNT(*)
FROM students
JOIN courses
ON students.student_id = courses.student_id
WHERE students.major = "Computer Science" AND courses.course_name =
"Python programming";
''')
print("Number of Computer Science students taking Python
programming:", cursor.fetchone()[0])
```

```
Number of Computer Science students taking Python programming: 2
```

3. JOIN the tables students and courses and select the students who have grades higher than "C", only show their name, major, gpa, course_name and grade

```
cursor = conn.execute('''
SELECT name, major, gpa, course_name, grade
FROM students
JOIN courses
ON students.student_id = courses.student_id
WHERE grade < "C";
''')
print("\nStudents with grades higher than C:")

# fetch and print the results
results = cursor.fetchall()
for row in results:
    print(row)
```

```
Students with grades higher than C:
('John', 'Computer Science', 3.5, 'Python programming', 'A')
('Samantha', 'Physics', 3.9, 'Python programming', 'A')
('Jane', 'Physics', 3.8, 'Data Structures', 'B')
('Bob', 'Engineering', 3.0, 'Database Systems', 'B')
('Jessica', 'Engineering', 3.8, 'Database Systems', 'A')
('Jacob', 'Physics', 3.4, 'Quantum Mechanics', 'A')
```

Q3 Aggregate functions, numerical logic and grouping

1. Find the average gpa of all students.
2. SELECT the student with the maximum gpa, display only their student_id, major and gpa
3. SELECT the student with the minimum gpa, display only their student_id, major and gpa
4. SELECT the students with a gpa greater than 3.6 in the majors of "Physics" and "Engineering", display only their student_id, major and gpa
5. Group the students by their major and retrieve the average grade of each major.
6. SELECT the top 2 students with the highest GPA in each major and order the results by major in ascending order, then by GPA in descending order

1. Find the average gpa of all students

```
cursor.execute("SELECT AVG(gpa) FROM students")
print("Average GPA is:", cursor.fetchone()[0])
```

```
Average GPA is: 3.5800000000000005
```

2. SELECT the student with the maximum gpa, display only their student_id, major and gpa

```
cursor.execute("""
SELECT student_id, major, gpa
FROM students
WHERE gpa = (SELECT MAX(gpa) FROM students);
""")
result = cursor.fetchone()
print(f"The student with the maximum GPA is: ID:{result[0]}, Major:
{result[1]}, and GPA:{result[2]}")
```

```
The student with the maximum GPA is: ID:4, Major:Physics, and GPA:3.9
```

3. SELECT the student with the minimum gpa, display only their student_id, major and gpa

```
cursor.execute("""
SELECT student_id, major, gpa FROM students WHERE gpa = (SELECT
MIN(gpa) FROM students);
""")
result = cursor.fetchone()
print(f"The student with the minimum GPA has ID: {result[0]}, Major:
{result[1]} and GPA: {result[2]}")
```

```
The student with the minimum GPA has ID: 3, Major: Engineering and
GPA: 3.0
```

4. SELECT the students with a gpa greater than 3.6 in the majors of "Physics" and "Engineering", display only their student_id, major and gpa

```
cursor.execute("""
SELECT student_id, major, gpa FROM students WHERE (major = 'Physics'
OR major = 'Engineering') AND gpa > 3.6;
""")
result = cursor.fetchall()
```

```
for i in result:
    print(f"Student ID: {i[0]}, Major: {i[1]}, and GPA: {i[2]}")
```

```
Student ID: 2, Major: Physics, and GPA: 3.8
Student ID: 4, Major: Physics, and GPA: 3.9
Student ID: 5, Major: Engineering, and GPA: 3.7
Student ID: 8, Major: Engineering, and GPA: 3.8
Student ID: 10, Major: Physics, and GPA: 3.9
```

5. Group the students by their major and retrieve the average grade of each major.

```
cursor.execute("""
SELECT major, AVG(gpa) as average_gpa FROM students GROUP BY major;
""")
result = cursor.fetchall()
for i in result:
    print(f"Major: {i[0]} and Average GPA is: {i[1]}")
```

```
Major: Computer Science and Average GPA is: 3.4333333333333336
Major: Engineering and Average GPA is: 3.5
Major: Physics and Average GPA is: 3.75
```

6. SELECT the top 2 students with the highest GPA in each major and order the results by major in ascending order, then by GPA in descending order

```
cursor.execute("""
WITH hgpa AS (
  SELECT student_id, major, gpa,
    ROW_NUMBER() OVER (PARTITION BY major ORDER BY gpa DESC) AS rank
  FROM students
)
SELECT student_id, major, gpa
FROM hgpa
WHERE rank <= 2
ORDER BY major, gpa DESC;
""")
result = cursor.fetchall()
for i in result:
    print(f"Student ID: {i[0]}, Major: {i[1]}, and GPA: {i[2]}")
```

```
Student ID: 6, Major: Computer Science, and GPA: 3.6
Student ID: 1, Major: Computer Science, and GPA: 3.5
Student ID: 8, Major: Engineering, and GPA: 3.8
Student ID: 5, Major: Engineering, and GPA: 3.7
Student ID: 4, Major: Physics, and GPA: 3.9
Student ID: 10, Major: Physics, and GPA: 3.9
```

```
# close the connection
conn.close()
```