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 (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.

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.

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

Your solution

In [1]:

```
import sqlite3
```

In [34]:

```
# Connect to the database file 'company.db' using the sqlite3 library
connection = sqlite3.connect('company.db')
```

In [35]:

```
# Create a cursor object to execute SQL commands
cursor = connection.cursor()
```

Question 1

In [3]:

```
sql_command = """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";"""
```

In [5]:

```
cursor.execute(sql_command) # create students table by executing the query using cursor object
```

1. SELECT all records in the table.

In [36]:

```
result = cursor.execute('SELECT * FROM students;')
```

In [37]:

```
for row in result.fetchall():
    print(row)
```

```
(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".

In [39]:

```
result = cursor.execute('SELECT * FROM students WHERE major = "Computer Science";')
```

In [40]:

```
for row in result.fetchall():
    print(row)
```

```
(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 (use SELECT DISTINCT) and order them by name, descending order (i.e. Physics first).

In [41]:

```
result = cursor.execute('SELECT DISTINCT(major) FROM students ORDER BY major DESC;')
```

In [42]:

```
for row in result.fetchall():
    print(row)
```

```
('Physics',)
('Engineering',)
('Computer Science',)
```

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

In [43]:

```
result = cursor.execute('SELECT name FROM students WHERE name LIKE "%e%" ORDER BY gpa ASC;')
```

In [44]:

```
for row in result.fetchall():
    print(row)
```

```
('Michael',)
('Emily',)
('James',)
('Jane',)
('Jessica',)
('Ashley',)
```

Question 2

In [46]:

```
sql_command = """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";"""
```

In [ ]:

```
cursor.execute(sql_command) # create courses table by executing the query using cursor object
```

1. COUNT the number of unique courses.

In [47]:

```
result = cursor.execute('SELECT COUNT(DISTINCT(course_name)) FROM courses;')
```

In [48]:

```
for row in result.fetchall():
    print(row)
```

```
(4,)
```

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

In [49]:

```
result = cursor.execute('SELECT COUNT(DISTINCT(name)) FROM students AS s INNER JOIN courses AS c USING(student_id) WHERE s.major = "Comput
```

In [50]:

```
for row in result.fetchall():
    print(row)
```

```
(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.

In [51]:

```
'SELECT s.name, s.major, s.gpa, c.course_name, c.grade FROM students AS s INNER JOIN courses AS c USING(student_id) WHERE c.grade < "C";')
```

In [52]:

```
for row in result.fetchall():
    print(row)
```

```
('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')
```

Question 3

1. Find the average gpa of all students.

In [53]:

```
result = cursor.execute('SELECT AVG(gpa) FROM students;')
```

In [54]:

```
for row in result.fetchall():
    print(row)
```

```
(3.5800000000000005,)
```

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

In [55]:

```
result = cursor.execute('SELECT student_id, major, MAX(gpa) FROM students;')
```

In [56]:

```
for row in result.fetchall():
    print(row)
```

```
(4, 'Physics', 3.9)
```

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

In [57]:

```
result = cursor.execute('SELECT student_id, major, MIN(gpa) FROM students;')
```

In [58]:

```
for row in result.fetchall():
    print(row)
```

```
(3, 'Engineering', 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

In [59]:

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

In [60]:

```
for row in result.fetchall():
    print(row)
```

```
(2, 'Physics', 3.8)
(4, 'Physics', 3.9)
(5, 'Engineering', 3.7)
(8, 'Engineering', 3.8)
(10, 'Physics', 3.9)
```

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

In [61]:

```
result = cursor.execute('SELECT major, AVG(gpa) FROM students GROUP BY major;')
```

In [62]:

```
for row in result.fetchall():
    print(row)
```

```
('Computer Science', 3.4333333333333336)
('Engineering', 3.5)
('Physics', 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

In [63]:

```
result = cursor.execute("""
WITH subquery AS (SELECT name, major, gpa, row_number() OVER (PARTITION BY major ORDER BY gpa DESC) AS rank FROM students)
SELECT name, major, gpa FROM subquery WHERE rank <= 2 ORDER BY major, gpa DESC;
""")
```

In [64]:

```
for row in result.fetchall():
    print(row)
```

```
('Emily', 'Computer Science', 3.6)
('John', 'Computer Science', 3.5)
('Jessica', 'Engineering', 3.8)
('James', 'Engineering', 3.7)
('Samantha', 'Physics', 3.9)
('Ashley', 'Physics', 3.9)
```

In [ ]: