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

First We will create the table using local database created by sqlite3 and then apply the functions given in the exercise

Create a student table having column names as student_id, name, major, gpa and enrollment_date and enter the data given.

In [1]:

```
import sqlite3
import pandas as pd
# Connect to the local database or create one
conn = sqlite3.connect('records.db')

# Create a cursor object
cursor = conn.cursor()

# Create the students table
cursor.execute('''
CREATE TABLE students (
student_id INT,
name TEXT,
major TEXT,
gpa REAL,
enrollment_date TEXT
);
''')

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

# Commit changes and close the connection
conn.commit()
conn.close()
```

Simple SELECTS (on the students table)

In [2]:

```
# Connect to the database
conn = sqlite3.connect('records.db')

# Create a cursor object
cursor = conn.cursor()
```

1. SELECT all records in the table.

In [3]:

```
selectall_qr = '''SELECT *
                  FROM students;
               '''
selectall = pd.read_sql_query(selectall_qr, conn)
print('Select all records in the table \n',selectall)
```

```
Select all records in the table
    student_id      name             major  gpa enrollment_date
0           1      John  Computer Science  3.5      2022-01-01
1           2      Jane           Physics  3.8      2022-01-02
2           3       Bob       Engineering  3.0      2022-01-03
3           4  Samantha           Physics  3.9      2022-01-04
4           5     James       Engineering  3.7      2022-01-05
5           6     Emily  Computer Science  3.6      2022-01-06
6           7   Michael  Computer Science  3.2      2022-01-07
7           8   Jessica       Engineering  3.8      2022-01-08
8           9     Jacob           Physics  3.4      2022-01-09
9          10    Ashley           Physics  3.9      2022-01-10
```

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

In [4]:

```
selectcs_qr = '''SELECT *
                 FROM students
                 WHERE major="Computer Science";
              '''
selectcs = pd.read_sql_query(selectcs_qr, conn)
print('Select students whose major is "Computer Science" \n',selectcs)
```

```
Select students whose major is "Computer Science"
    student_id     name             major  gpa enrollment_date
0           1     John  Computer Science  3.5      2022-01-01
1           6    Emily  Computer Science  3.6      2022-01-06
2           7  Michael  Computer Science  3.2      2022-01-07
```

3. SELECT all unique majors (use SELECT DISTINCT) and order them by name, descending order (i.e. Physics first).

In [5]:

```
selectmaj_qr = '''SELECT DISTINCT major
                  FROM students
                  ORDER BY major DESC;
               '''
select_maj = pd.read_sql_query(selectmaj_qr, conn)
print('Select all unique majors and order them by name, descending order \n',select_maj)
```

```
Select all unique majors and order them by name, descending order
              major
0           Physics
1       Engineering
2  Computer Science
```

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

In [6]:

```
selecte_qr = '''SELECT *
                FROM students
                WHERE name LIKE '%e%'
                ORDER BY gpa ASC;
             '''
selecte = pd.read_sql_query(selecte_qr, conn)
print('Select all students that have an "e" in their name and order them by gpa in ascending order \n',selecte)
```

```
Select all students that have an "e" in their name and order them by gpa in ascending order
    student_id     name             major  gpa enrollment_date
0           7  Michael  Computer Science  3.2      2022-01-07
1           6    Emily  Computer Science  3.6      2022-01-06
2           5    James       Engineering  3.7      2022-01-05
3           2     Jane           Physics  3.8      2022-01-02
4           8  Jessica       Engineering  3.8      2022-01-08
5          10   Ashley           Physics  3.9      2022-01-10
```

In [7]:

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

Create new table courses and perform the given tasks

In [8]:

```
import sqlite3
import pandas as pd
# Connect to the local database or create one
conn = sqlite3.connect('records.db')

# Create a cursor object
cursor = conn.cursor()

# Create courses table
cursor.execute('''
CREATE TABLE courses (
course_id INT,
course_name TEXT,
student_id INT,
grade TEXT
);
''')

# Insert data into the table
cursor.execute('''
INSERT INTO courses (course_id, course_name, student_id, grade)
VALUES
(1, "Python programming", 1, "A"),
(2, "Data Structures", 2, "B"),
(3, "Database Systems", 3, "B"),
(1, "Python programming", 4, "A"),
(4, "Quantum Mechanics", 5, "C"),
(1, "Python programming", 6, "F"),
(2, "Data Structures", 7, "C"),
(3, "Database Systems", 8, "A"),
(4, "Quantum Mechanics", 9, "A"),
(2, "Data Structures", 10, "F");
''')

# Commit changes and close the connection
conn.commit()
conn.close()
```

Joins

In [9]:

```
# Connect to the database
conn = sqlite3.connect('records.db')

# Create cursor object
cursor = conn.cursor()
```

1. COUNT the number of unique courses.

In [10]:

```
selectco_qr = '''SELECT COUNT( DISTINCT course_name) as course_name
                 FROM courses;
               '''
select_co = pd.read_sql_query(selectco_qr, conn)
print('Count the number of unique courses.\n',select_co)
```

```
Count the number of unique courses.
    course_name
0            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 [11]:

```
selectcspy_qr = '''SELECT COUNT(*)
                   FROM students
                   JOIN courses ON students.student_id = courses.student_id
                   WHERE major = "Computer Science"
                   AND course_name = "Python programming";
                  '''
select_cspy = pd.read_sql_query(selectcspy_qr, conn)
print('Students with the major Computer Science taking the course Python.\n',select_cspy)
```

```
Students with the major Computer Science taking the course Python.
    COUNT(*)
0         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 [12]:

```
selecthc_qr = '''SELECT name, major, gpa, course_name, grade
                    FROM students
                    JOIN courses
                    ON students.student_id = courses.student_id
                    WHERE grade < "C";
                 '''
select_hc = pd.read_sql_query(selecthc_qr, conn)
print('Students who have grades higher than "C"\n',select_hc)
```

```
Students who have grades higher than "C"
        name             major  gpa         course_name grade
0      John  Computer Science  3.5  Python programming     A
1      Jane           Physics  3.8     Data Structures     B
2       Bob       Engineering  3.0    Database Systems     B
3  Samantha           Physics  3.9  Python programming     A
4   Jessica       Engineering  3.8    Database Systems     A
5     Jacob           Physics  3.4   Quantum Mechanics     A
```

In [13]:

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

Q3 Aggregate functions, numerical logic and grouping

In [14]:

```
# Connect to the database
conn = sqlite3.connect('records.db')

# Create a cursor object
cursor = conn.cursor()
```

1. Find the average gpa of all students.

In [15]:

```
avg_qr = '''SELECT AVG(gpa)
               FROM students;
            '''
avg_gpa = pd.read_sql_query(avg_qr, conn)
print('Average gpa of all students.\n',avg_gpa)
```

```
Average gpa of all students.
    AVG(gpa)
0      3.58
```

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

In [16]:

```
maxgpa_qr = '''SELECT student_id, major, gpa
               FROM students
               WHERE gpa = (SELECT MAX(gpa) FROM students);
            '''
max_gpa = pd.read_sql_query(maxgpa_qr, conn)
print('Student with the maximum gpa.\n',max_gpa)
```

```
Student with the maximum gpa.
    student_id    major  gpa
0           4  Physics  3.9
1          10  Physics  3.9
```

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

In [17]:

```
mingpa_qr = '''SELECT student_id, major, gpa
               FROM students
               WHERE gpa = (SELECT MIN(gpa) FROM students);
            '''
min_gpa = pd.read_sql_query(mingpa_qr, conn)
print('Student with the minimum gpa.\n',min_gpa)
```

```
Student with the minimum gpa.
    student_id        major  gpa
0           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 [18]:

```
gr_qr = '''SELECT student_id, major, gpa
               FROM students
               WHERE major IN ('Physics', 'Engineering') AND gpa > 3.6;
            '''
gr_gpa = pd.read_sql_query(gr_qr, conn)
print('Students with a gpa greater than 3.6 in the majors of "Physics" and "Engineering".\n',gr_gpa)
```

```
Students with a gpa greater than 3.6 in the majors of "Physics" and "Engineering".
    student_id        major  gpa
0           2      Physics  3.8
1           4      Physics  3.9
2           5  Engineering  3.7
3           8  Engineering  3.8
4          10      Physics  3.9
```

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

In [19]:

```
avg_maj_qr = '''SELECT major, AVG(gpa)
               FROM students
               GROUP BY major;
            '''
avg_maj = pd.read_sql_query(avg_maj_qr, conn)
print('Average grade of each major.\n',avg_maj)
```

```
Average grade of each major.
               major  AVG(gpa)
0  Computer Science  3.433333
1       Engineering  3.500000
2           Physics  3.750000
```

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 [20]:

```
hgpa_qr = '''WITH students_rank 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 students_rank
                WHERE rank <= 2
                ORDER BY major, gpa DESC;
            '''
h_gpa = pd.read_sql_query(hgpa_qr, conn)
print('Average gpa of all students.\n',h_gpa)
```

```
Average gpa of all students.
    student_id             major  gpa
0           6  Computer Science  3.6
1           1  Computer Science  3.5
2           8       Engineering  3.8
3           5       Engineering  3.7
4           4           Physics  3.9
5          10           Physics  3.9
```

In [21]:

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