Data X Berkeley

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 gp
a, "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]:

```
# connecting database
import sqlite3
connection = sqlite3.connect('students.db')
cursor = connection.cursor()
```

In [4]:

```
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";
"""
cursor.execute(sql_command)
```

Q1 Simple SELECTS (on the students table)

In [2]:

```
#  SELECT all records in the table
cursor.execute('SELECT * FROM students;').fetchall()
```

Out[2]:

```
[(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')]
```

In [3]:

```
# SELECT students whose major is "Computer Science"
cursor.execute('SELECT name FROM students WHERE major IS "Computer Science";').fetchall()
```

Out[3]:

```
[('John',), ('Emily',), ('Michael',)]
```

In [4]:

```
# SELECT all unique majors (use SELECT DISTINCT) and order them by name, descending order (i.e. Physics first)
cursor.execute('SELECT DISTINCT major FROM students ORDER BY major DESC;').fetchall()
```

Out[4]:

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

In [5]:

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

Out[5]:

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

In [ ]:

```

```

In [ ]:

```

```

Q2 Joins

In [6]:

```
connection = sqlite3.connect('courses.db')
cursor = connection.cursor()
```

In [7]:

```
# We can define long SQL commands within three quotes

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";
"""
cursor.execute(sql_command)
```

In [11]:

```
cursor.execute('SELECT * FROM courses;').fetchall()
```

Out[11]:

```
[(1, 'Python programming', 1, 'A'),
 (1, 'Python programming', 4, 'A'),
 (1, 'Python programming', 6, 'F'),
 (2, 'Data Structures', 2, 'B'),
 (2, 'Data Structures', 7, 'C'),
 (2, 'Data Structures', 10, 'F'),
 (3, 'Database Systems', 3, 'B'),
 (3, 'Database Systems', 8, 'A'),
 (4, 'Quantum Mechanics', 5, 'C'),
 (4, 'Quantum Mechanics', 9, 'A')]
```

In [12]:

```
# COUNT the number of unique courses
cursor.execute('SELECT COUNT(DISTINCT course_name) FROM courses;').fetchall()
```

Out[12]:

```
[(4,)]
```

In [7]:

```
# JOIN the tables students and courses and COUNT the number of students with the major co
mputer Science taking the course Python programming.
cursor.execute('SELECT COUNT(students.name) FROM students INNER JOIN courses ON students.
student_id = courses.student_id WHERE major IS "Computer Science" AND course_name IS "Pyt
hon programming";').fetchall()
```

Out[7]:

```
[(2,)]
```

In [10]:

```
# JOIN the tables students and courses and select the students who have grades higher tha
n "C", only show their name, major, gpa, course_name and grade.
cursor.execute('SELECT name, major, gpa, course_name, grade FROM students INNER JOIN cour
ses ON students.student_id = courses.student_id WHERE grade < "C";').fetchall()
```

Out[10]:

```
[('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')]
```

In [ ]:

```
```

In [ ]:

```
```

Q3 Aggregate functions, numerical logic and grouping

In [ ]:

```
# Find the average gpa of all students
cursor.execute('SELECT AVG(gpa) FROM students;').fetchall()
```

```
[(3.5800000000000005,)]
```

In [24]:

```
# SELECT the student with the maximum gpa, display only their student_id, major and gpa
cursor.execute('SELECT student_id, major, MAX(gpa) FROM students;').fetchall()
```

Out[24]:

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

In [25]:

```
# SELECT the student with the minimum gpa, display only their student_id, major and gpa
cursor.execute('SELECT student_id, major, MIN(gpa) FROM students;').fetchall()
```

Out[25]:

```
[(3, 'Engineering', 3.0)]
```

In [11]:

```
# SELECT the students with a gpa greater than 3.6 in the majors of "Physics" and "Enginee
ring", display only their student_id, major and gpa
cursor.execute('SELECT student_id, major, gpa FROM students WHERE (major IS "Physics" OR
major IS "Engineering") AND gpa > 3.6;').fetchall()
```

Out[11]:

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

In [16]:

```
# Group the students by their major and retrieve the average grade of each major.
cursor.execute('SELECT AVG(gpa),major FROM courses INNER JOIN students ON courses.student
_id = students.student_id GROUP BY major;').fetchall()
```

Out[16]:

```
[(3.4333333333333336, 'Computer Science'),
 (3.5, 'Engineering'),
 (3.75, 'Physics')]
```

In [21]:

```
# 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('SELECT * FROM (SELECT major, gpa, row_number() OVER (PARTITION BY major ORDER BY gpa DESC) AS c_rank FROM students) ranks WHERE c_rank <=2;').fetchall()
```

Out[21]:

```
[('Computer Science', 3.6, 1),
 ('Computer Science', 3.5, 2),
 ('Engineering', 3.8, 1),
 ('Engineering', 3.7, 2),
 ('Physics', 3.9, 1),
 ('Physics', 3.9, 2)]
```

In [ ]:

In [ ]: