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 (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]:

```
# sqlite3 package comes with the Python installation
import sqlite3
```

In [2]:

```
connection = sqlite3.connect('company.db')
cursor = connection.cursor()
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).fetchall()
```

Out[2]:

```
[]
```

Question 1

In [3]:

```
sql_command= '''
SELECT *
FROM students
'''
cursor.execute(sql_command).fetchall()
```

Out[3]:

```
[(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 [4]:

```
sql_command='''
SELECT *
FROM students
WHERE major='Computer Science'
'''
cursor.execute(sql_command).fetchall()
```

Out[4]:

```
[(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')]
```

In [5]:

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

Out[5]:

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

In [6]:

```
sql_command='''
SELECT *
FROM students
WHERE name LIKE '%e%'
ORDER BY gpa;
'''
cursor.execute(sql_command).fetchall()
```

Out[6]:

```
[(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')]
```

Question 2

In [7]:

```
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).fetchall()
```

Out[7]:

```
[]
```

In [8]:

```
sql_command='''
SELECT COUNT (DISTINCT course_name) AS no_of_distinct_courses
FROM courses;
'''
print("The number of unique courses is",cursor.execute(sql_command).fetchall())
```

```
The number of unique courses is [(4,)]
```

In [9]:

```
sql_command='''
SELECT COUNT(*) AS comp_sci_major_with_python
FROM students
INNER JOIN courses
USING (student_id)
WHERE major = 'Computer Science' AND course_name='Python programming';
'''
cursor.execute(sql_command).fetchall()
```

Out[9]:

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

In [10]:

```
sql_command='''
SELECT *
FROM students
INNER JOIN courses
USING (student_id)
WHERE grade<'C';
'''
cursor.execute(sql_command).fetchall()
```

Out[10]:

```
[(1,
  'John',
  'Computer Science',
  3.5,
  '01-01-2022',
  1,
  'Python programming',
  'A'),
 (4, 'Samantha', 'Physics', 3.9, '01-04-2022', 1, 'Python programming', 'A'),
 (2, 'Jane', 'Physics', 3.8, '01-02-2022', 2, 'Data Structures', 'B'),
 (3, 'Bob', 'Engineering', 3.0, '01-03-2022', 3, 'Database Systems', 'B'),
 (8, 'Jessica', 'Engineering', 3.8, '01-08-2022', 3, 'Database Systems', 'A'),
 (9, 'Jacob', 'Physics', 3.4, '01-09-2022', 4, 'Quantum Mechanics', 'A')]
```

Question 3

In [11]:

```
sql_command='''
SELECT AVG(gpa) AS avg_gpa
FROM students
'''
cursor.execute(sql_command).fetchall()
```

Out[11]:

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

In [12]:

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

Out[12]:

```
[(4, 'Physics', 3.9), (10, 'Physics', 3.9)]
```

In [13]:

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

Out[13]:

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

In [14]:

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

Out[14]:

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

In [17]:

```
#Group the students by their major and retrieve the average grade of each major
sql_command='''
SELECT major,AVG(gpa)
FROM students
GROUP BY major;
'''
cursor.execute(sql_command).fetchall()
```

Out[17]:

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

In [19]:

```
#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 desce
sql_command='''
SELECT student_id,name,major,gpa,enrollment_date
FROM (SELECT student_id,name,major,gpa,enrollment_date,
     ROW_NUMBER() OVER (PARTITION BY major ORDER BY gpa DESC) AS rank_within_major
     FROM students
     )
WHERE rank_within_major<3
ORDER BY major,gpa DESC;
'''
cursor.execute(sql_command).fetchall()
```

Out[19]:

```
[(6, 'Emily', 'Computer Science', 3.6, '01-06-2022'),
 (1, 'John', 'Computer Science', 3.5, '01-01-2022'),
 (8, 'Jessica', 'Engineering', 3.8, '01-08-2022'),
 (5, 'James', 'Engineering', 3.7, '01-05-2022'),
 (4, 'Samantha', 'Physics', 3.9, '01-04-2022'),
 (10, 'Ashley', 'Physics', 3.9, '01-10-2022')]
```

In [ ]:

In [ ]: