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).

Importing Libraries

In [1]:

```
# sqlite3 package comes with the Python installation
import sqlite3
import pandas as pd
```

'students' Table Creation

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 :)

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";
```

In [2]:

```
# open connnection to a db file stored locally on disk if file doesn't exist it is
created
connection = sqlite3.connect('students.db')
# In order to run SQL commands with sqlite 3 we must create a cursor object that
traverses the database
cursor = connection.cursor()
# Check that we are working with an empty db
cursor.execute("DROP TABLE IF EXISTS employee;")
```

Out[2]:

```
<sqlite3.Cursor at 0x7fd7735c3ce0>
```

In [3]:

```
sql_command='''
CREATE TABLE students AS
        SELECT 1 AS student_id, "John" AS name, "Computer Science" AS major, 3.5 A
S 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 order to run SQL command on the databse file
# we have to execute them with the cursor
cursor.execute(sql_command)
```

Out[3]:

```
<sqlite3.Cursor at 0x7fd7735c3ce0>
```

Q1 Simple SELECTS (on the students table)

1. **SELECT all records in the table.**

In [9]:

```
sql_command = '''
SELECT *
FROM students;
'''
pd.read_sql_query(sql_command,connection)
```

Out[9]:

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

1. **SELECT students whose major is "Computer Science".**

In [10]:

```
sql_command = '''
SELECT *
FROM students
WHERE major='Computer Science';
'''
pd.read_sql_query(sql_command,connection)
```

Out[10]:

	student_id	name	major	gpa	enrollment_date
0	1	John	Computer Science	3.5	01-01-2022
1	6	Emily	Computer Science	3.6	01-06-2022
2	7	Michael	Computer Science	3.2	01-07-2022

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

In [11]:

```
sql_command = '''
SELECT DISTINCT major
FROM students
```

```
ORDER BY major DESC;
'''
pd.read_sql_query(sql_command,connection)
```

Out[11]:

	major
0	Physics
1	Engineering
2	Computer Science

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

In [12]:

```
sql_command = '''
SELECT *
FROM students
WHERE name LIKE '%e%'
'''
pd.read_sql_query(sql_command,connection)
```

Out[12]:

	student_id	name	major	gpa	enrollment_date
0	2	Jane	Physics	3.8	01-02-2022
1	5	James	Engineering	3.7	01-05-2022
2	6	Emily	Computer Science	3.6	01-06-2022
3	7	Michael	Computer Science	3.2	01-07-2022
4	8	Jessica	Engineering	3.8	01-08-2022
5	10	Ashley	Physics	3.9	01-10-2022

Q2 Joins

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

'courses' Table Creation

Create the table by running:

```
CREATE TABLE courses AS
    SELECT 1 AS course_id, "Python programming" AS course_name, 1 AS studen
t_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 [13]:

```
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)
```

Out[13]:

```
<sqlite3.Cursor at 0x7fc60d91e3b0>
```

1. **COUNT the number of unique courses.**

In [14]:

```
sql_command = '''
SELECT COUNT(DISTINCT course_name) AS unique_course_count
FROM courses
'''
pd.read_sql_query(sql_command,connection)
```

Out[14]:

	unique_course_count
0	**4**

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

In [15]:

```
sql_command = '''
SELECT COUNT(*) AS student_count_cs_pp
FROM students AS s
INNER JOIN courses AS c
ON s.student_id=c.student_id
WHERE s.major = 'Computer Science'
AND c.course_name='Python programming';
'''
pd.read_sql_query(sql_command,connection)
```

Out[15]:

	student_count_cs_pp
0	2

1. **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 [16]:

```
sql_command = '''
SELECT s.name,s.major,s.gpa,c.course_name,c.grade
FROM students AS s
INNER JOIN courses as c
ON s.student_id=c.student_id
WHERE c.grade < 'C';
'''
pd.read_sql_query(sql_command,connection)
```

Out[16]:

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

Q3 Aggregate functions, numerical logic and grouping

1. **Find the average gpa of all students.**

In [17]:

```
sql_command = '''
SELECT ROUND(AVG(gpa),2) AS average_gpa
FROM students;
'''
pd.read_sql_query(sql_command,connection)
```

Out[17]:

	average_gpa
0	3.58

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

In [4]:

```
sql_command = '''
SELECT student_id,major,MAX(gpa)
```

```
FROM students;
'''
pd.read_sql_query(sql_command,connection)
```

Out[4]:

	student_id	major	MAX(gpa)
0	4	Physics	3.9

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

In [5]:

```
sql_command = '''
SELECT student_id,major,MIN(gpa)
FROM students;
'''
pd.read_sql_query(sql_command,connection)
```

Out[5]:

	student_id	major	MIN(gpa)
0	3	Engineering	3.0

1. **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 [6]:

```
sql_command = '''
SELECT student_id,major,gpa
FROM students
WHERE gpa>3.6
AND major='Physics'
OR major='Engineering';
'''
pd.read_sql_query(sql_command,connection)
```

Out[6]:

	student_id	major	gpa
0	2	Physics	3.8
1	3	Engineering	3.0
2	4	Physics	3.9
3	5	Engineering	3.7
4	8	Engineering	3.8
5	10	Physics	3.9

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

In [7]:

```
sql_command = '''
```

```
SELECT ROUND(AVG(gpa),2),major
FROM students
GROUP BY major;
'''
pd.read_sql_query(sql_command,connection)
```

Out[7]:

	ROUND(AVG(gpa),2)	major
0	3.43	Computer Science
1	3.50	Engineering
2	3.75	Physics

1. **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 [9]:

```
sql_command = '''
SELECT name,major,gpa,enrollment_date
FROM (SELECT 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<=2
ORDER BY major,gpa DESC;
'''
pd.read_sql_query(sql_command,connection)
```

Out[9]:

	name	major	gpa	enrollment_date
0	Emily	Computer Science	3.6	01-06-2022
1	John	Computer Science	3.5	01-01-2022
2	Jessica	Engineering	3.8	01-08-2022
3	James	Engineering	3.7	01-05-2022
4	Samantha	Physics	3.9	01-04-2022
5	Ashley	Physics	3.9	01-10-2022