Author - Vysakh R K

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 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

MY SOLUTION

Q1 Simple SELECTS (on the students table)

Importing the required packages

In [201]:
```
import sqlite3
import pandas as pd

# No warnings
import warnings
warnings.filterwarnings("ignore")  # Filter out warnings
```

Initial activities of creation and inserting values using sqlite

In [202]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
cursor.execute('DROP TABLE IF EXISTS students')
```

Out[202]:
```
<sqlite3.Cursor at 0x1b38bc94650>
```

In [203]:
```
table=  '''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 [204]:
```
cursor.execute(table)
```

Out[204]:
```
<sqlite3.Cursor at 0x1b38bc94650>
```

In [205]:
```
connection.commit()
connection.close()
```

In [206]:
```
#See if the above steps have worked well
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
A=cursor.execute('SELECT * FROM students')

for row in A.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')
```

Q1 Simple SELECTS (on the students table)

1. SELECT all records in the table.

In [207]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
pd.read_sql("SELECT * FROM students",con = connection)
```

Out[207]:

	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

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

In [208]:
```
pd.read_sql('SELECT * FROM students WHERE [major]="Computer Science"',con=conn
ection)
```

Out[208]:

	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

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

In [209]:
```
pd.read_sql('SELECT DISTINCT [major] FROM students ORDER BY [name] DESC',con=connection)
```

Out[209]:

	major
0	Computer Science
1	Physics
2	Engineering

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

In [210]:
```
pd.read_sql('SELECT * FROM students WHERE [name] LIKE "%e%" ORDER BY [gpa]' ,con=connection)
```

Out[210]:

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

In [211]:
```
connection.close()
```

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";
```

In [212]:
```
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
cursor.execute('DROP TABLE IF EXISTS courses')
```

Out[212]:
```
<sqlite3.Cursor at 0x1b38bc94c00>
```

In [213]:
```
query = '''CREATE TABLE courses AS
        SELECT 1 AS course_id, "Python programming" AS course_name, 1 AS stude
nt_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 [214]:
```
cursor.execute(query)
connection.commit()
connection.close()
```

In [215]:
```
#Checking if the connection is established and table is set
connection=sqlite3.connect('database.db')
cursor=connection.cursor()
pd.read_sql('SELECT * FROM courses',con=connection)
```

Out[215]:

	course_id	course_name	student_id	grade
0	1	Python programming	1	A
1	1	Python programming	4	A
2	1	Python programming	6	F
3	2	Data Structures	2	B
4	2	Data Structures	7	C
5	2	Data Structures	10	F
6	3	Database Systems	3	B
7	3	Database Systems	8	A
8	4	Quantum Mechanics	5	C
9	4	Quantum Mechanics	9	A

1. COUNT the number of unique courses.

In [216]:
```
pd.read_sql('SELECT COUNT ( DISTINCT course_name ) AS "Number of unique course
s" FROM courses',con=connection)
```

Out[216]:

	Number of unique courses
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 [217]:
```
pd.read_sql('SELECT COUNT(*) AS "Number of Students with major CS and course P
ython"\
            FROM students A \
            INNER JOIN courses B\
            ON A.student_id=B.student_id\
            AND A.major="Computer Science"\
            AND B.course_name="Python programming" ',con=connection)
```

Out[217]:

	Number of Students with major CS and course Python
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 [218]:
```
pd.read_sql('SELECT DISTINCT A.name, A.major, A.gpa, B.course_name, B.grade\
             FROM students A \
             INNER JOIN courses B\
             ON A.student_id=B.student_id\
             WHERE B.grade < "C" ORDER BY B.grade',con=connection)
```

Out[218]:

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

Q3 Aggregate functions, numerical logic and grouping

1. Find the average gpa of all students.

In [219]:
```
pd.read_sql('SELECT AVG([gpa]) AS "Average GPA"\
             FROM students ',con = connection)
```

Out[219]:

	Average GPA
0	3.58

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

In [220]:
```
pd.read_sql('SELECT [student_id],[major],MAX([gpa]) AS "Maximum GPA"\
             FROM students',con=connection)
```

Out[220]:

	student_id	major	Maximum GPA
0	4	Physics	3.9

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

In [221]:
```
pd.read_sql('SELECT [student_id],[major],MIN([gpa]) AS "Minimum GPA"\
            FROM students',con=connection)
```

Out[221]:

	student_id	major	Minimum 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 [222]:
```
pd.read_sql('SELECT [student_id],[major],[gpa]FROM students \
            WHERE [major] IN ("Physics", "Engineering") AND gpa > 3.6 ',con=co
nnection)
```

Out[222]:

	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 [223]:
```
pd.read_sql('SELECT [major],AVG([gpa]) AS "Average Grade of each major"\
            FROM students \
            GROUP BY [major]',con = connection)
```

Out[223]:

	major	Average Grade of each major
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 [224]:
```
#create a partition for each MAJOR using the row_number() concept
pd.read_sql('SELECT [student_id], [name], [major], [gpa], \
             row_number() OVER (PARTITION BY [major] order by [gpa] DESC) AS
"Major Rank" \
              from students ',con = connection)
```

Out[224]:

	student_id	name	major	gpa	Major Rank
0	6	Emily	Computer Science	3.6	1
1	1	John	Computer Science	3.5	2
2	7	Michael	Computer Science	3.2	3
3	8	Jessica	Engineering	3.8	1
4	5	James	Engineering	3.7	2
5	3	Bob	Engineering	3.0	3
6	4	Samantha	Physics	3.9	1
7	10	Ashley	Physics	3.9	2
8	2	Jane	Physics	3.8	3
9	9	Jacob	Physics	3.4	4

Using the above query we got the partition of the majors. Now using the saem logic, keeping them in a temporary space, making use of the temp query - we can query out the top 2 students in each major

In [225]:
```
#Selecting the top 2 candidates of each major
pd.read_sql('SELECT * FROM (SELECT [student_id] AS "Student ID", [name] AS "Na
me", [major] AS "Major", [gpa] AS "GPA", \
             row_number() OVER (PARTITION BY [major] order by [gpa] DESC) AS
"Major Rank" \
              from students) top_students WHERE [Major Rank] <= 2 ',con = conn
ection)
```

Out[225]:

	Student ID	Name	Major	GPA	Major Rank
0	6	Emily	Computer Science	3.6	1
1	1	John	Computer Science	3.5	2
2	8	Jessica	Engineering	3.8	1
3	5	James	Engineering	3.7	2
4	4	Samantha	Physics	3.9	1
5	10	Ashley	Physics	3.9	2

In [226]:
```
#Closing the connection (BEST PRACTICES)
connection.commit()
connection.close()
```

Thank You