

# 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 submission 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

In [1]:

```
import sqlite3
!ls
```

sample\_data

In [3]:

```
connect = sqlite3.connect('students.db')
!ls
```

sample\_data students.db

In [4]:

```
cursor = connect.cursor()
```

# Q1 Answer

In [6]:

```
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";
"""
```

In [7]:

```
cursor.execute(command)
```

Out[7]:

```
<sqlite3.Cursor at 0x7f2300aab110>
```

In [8]:

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

Out[8]:

```
[(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 [9]:

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

Out[9]:

```
[(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 [10]:

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

Out[10]:

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

In [11]:

```
cursor.execute("SELECT name, gpa FROM students WHERE name LIKE '%e%' OR name LIKE '%E%'
ORDER BY gpa ;").fetchall()
```

Out[11]:

```
[('Michael', 3.2),
 ('Emily', 3.6),
```

```
('James', 3.7),  
('Jane', 3.8),  
('Jessica', 3.8),  
('Ashley', 3.9)]
```

## Q2 Answer

In [12]:

```
command2 = """  
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 [13]:

```
cursor.execute(command2)
```

Out[13]:

```
<sqlite3.Cursor at 0x7f2300aab110>
```

In [14]:

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

Out[14]:

```
[(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 [15]:

```
cursor.execute('SELECT COUNT(DISTINCT course_name) FROM courses;').fetchone()
```

Out[15]:

```
(4,)
```

In [16]:

```
cursor.execute('SELECT COUNT(students.major) FROM students INNER JOIN courses ON students.  
.student_id=courses.student_id WHERE students.major=="Computer Science" AND courses.course  
_name=="Python programming" ;').fetchall()
```

Out[16]:

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

In [17]:

```
cursor.execute('SELECT students.name, students.major, students.gpa, courses.course_name,  
courses.grade FROM students INNER JOIN courses ON students.student_id=courses.student_id
```

```
WHERE courses.grade<"C" ;').fetchall()
```

Out[17]:

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

## Q3 Answer

In [19]:

```
cursor.execute("SELECT AVG(gpa) FROM students").fetchall()
```

Out[19]:

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

In [20]:

```
cursor.execute("SELECT student_id, major, gpa FROM students WHERE gpa = (SELECT MAX(gpa)  
FROM students)").fetchall()
```

Out[20]:

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

In [21]:

```
cursor.execute("SELECT student_id, major, gpa FROM students WHERE gpa = (SELECT MIN(gpa)  
FROM students)").fetchall()
```

Out[21]:

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

In [22]:

```
cursor.execute("SELECT student_id, major, gpa FROM students WHERE major IN ('Physics', 'E  
ngineering') AND gpa > 3.6").fetchall()
```

Out[22]:

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

In [23]:

```
cursor.execute("SELECT major, AVG(gpa) FROM students GROUP BY major").fetchall()
```

Out[23]:

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

In [24]:

```
cursor.execute('SELECT major, name, gpa FROM'  
               '(SELECT major, name, gpa, ROW_NUMBER() OVER (PARTITION BY major ORDER B  
Y gpa DESC) AS rank FROM students)'  
               'WHERE rank<=2 ORDER BY major ASC, gpa DESC'  
) .fetchall()
```

Out[24]:

```
[('Computer Science', 'Emilv', 3.6),
```

```
('Computer Science', 'John', 3.5),  
('Engineering', 'Jessica', 3.8),  
('Engineering', 'James', 3.7),  
('Physics', 'Samantha', 3.9),  
('Physics', 'Ashley', 3.9)]
```

In [ ]: