Implementation/SQL

Introduction:

An assignment to make sure you get Postgres installed and can execute basic statements. The subject of the table you will work with is books. (If you know a little Internet history, you might recall that Jeff Bezos started Amazon as a business to sell books online.) We and modify this in the weeks to come.

The assignment:

1. Download and install Postgres.
2. Create a database or use the default database called "postgres"
3. Create a script "create.sql" to define the table "book".
    1. Remember that SQL is code, even if the syntax is new for you. Do not use Word or another word processor to edit it. The Query Tool feature of pgAdmin has syntax highlighting though it is not the smoothest UI to work with.
        1. When saving your files, remember to add ".sql" at the end – the pgAdmin editor does not do this automatically ☹
        2. Execute your scripts to test them.
    2. To distinguish assignments for grading purposes, create a schema named using your last name (up to 6 letters), appending your first initial, so my schema would be `yangd`
        1. You can find examples at: PostgreSQL: Documentation: 13: CREATE SCHEMA
        2. This general section of the Postgres documentation has all the SQL syntax, so it's a decent bookmark to have for this semester
    3. Define the table within the schema – for me, this would be `yangd.book`
    4. The fields are:
        1. ASIN : a 10-character string for Amazon products
            - Note that SQL uses parentheses to indicate the number of characters, so something like VARCHAR(100)
            - You will not find a value for this for many books
            - For now, just assign the type of the field – we will modify the design in hw2 to support uniqueness
        2. title : a string for the name of the book, like 'Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought'.
        3. name : a string for the name of the author, like 'Toni Morrison'
        4. publisher : a string for the name of the publisher
        5. pubdate: date on which the book was first published
        6. pages : an int for the number of pages in the book
2. Create a script "insert_ok.sql" to insert acceptable data. Use INSERT statements to insert records into the book table
    1. Insert enough records so that any queries beyond the first one will return at least one, but not all records
    2. You may find PostgreSQL: Documentation: 13: 9.8. Data Type Formatting Functions a useful resource for the pubdate attribute – just call the function instead of a literal value
        1. SQL itself does not have the kind of variables like Python or C++ does, so do not try to separate this into multiple statements
        2. [Databases like Postgres do support *stored procedures* for more complicated operations, but that is a separate language]
3. Create a script "query.sql" that includes semicolon-separated queries to do the following (terminate each query with a semicolon so the grader can run the whole script at once) :
    1. List names of all publishers
    2. List titles and number of pages of all available books by a particular author (pick any author)
    3. List names of authors who have published a book in 2021. Since the date includes month and day as well, use the EXTRACT() function to get the year out of pubdate. See PostgreSQL: Documentation: 13: 9.9. Date/Time Functions and Operators for ex like: extract(year from pubdate)]
4. Create a script "insert_bad.sql" that uses INSERT statements to insert data that should not be allowed, "violating" the rules below. We will eventually cover the various features of SQL that allow you to block these inserts, but for now, they should be allowed
    1. ASIN should be unique
    2. Pages should be at least 1
    3. Title should not be null
5. Submit all files to Blackboard

Grading:

20% create.sql
15% insert_ok.sql
30% query.sql, 10% each query
15% insert_bad.sql, 5% each insert
10% comments to explain your statements – this is primarily for the queries and inserting bad data. Single line comments for each query and each bad insert is what I am looking for. Use a double dash (--) like you would use # in Python or // in C++ or Java
10% naming – create and name the schema, table and attributes appropriately