EECS 484 W18 Homework #2

EECS 484 W18 Staff

Due: February 2nd, 2018 at 11:55pm

There are three parts to this homework assignment, all of which must be appropriately submitted prior to the assignment deadline. For Part 3, you may submit either a computer-generated PDF or a PDF conversion of a hand-written solution. If you opt to write your solution by hand, make sure that your writing is clear, legible, and unambiguous in all circumstances.

This assignment is to be completed ***individually***; you may not share answers with other students actively enrolled in the course, nor may you consult with students who took the course in previous semesters. You are, however, allowed to discuss general approaches and class concepts with other students, and you are also permitted (and encouraged!) to post questions on Piazza. As always, you are also encouraged to come to Office Hours with questions.

The University of Michigan College of Engineering Honor Code strictly applies to this assignment, and we will be thoroughly checking to ensure that all submissions adhere to the Honor Code guidelines. Students whose submissions are found to be in violation of the Honor Code will be reported directly to the Honor Council.

Late submissions for this assignment will not be accepted. Both Gradescope and the Autograder enforce a strict cut-off at the specified deadline, so be sure that your submission is made prior to that time. There is no limit to the number of submissions you can make; your final submission will be accepted for grading.

Part 1: University SQL (40 points)

Consider a database consisting of the following five tables. Attributes that are underlined have been designated the primary key of their respective tables, and fields with identical names in different tables can be safely assumed to be foreign keys.

`Students(SID, Name, Major)`

`Projects(PID, Project_Name)`

`Courses(CID, Course_Name)`

`Members(PID, SID)`

`Enrollments(CID, SID)`

The `Project_Name` and `Course_Name` fields are specified as `UNIQUE`, and all fields are required *except* for `Major`, which may be `NULL` (to represent undeclared).

Write each of the following queries in a separate file titled `Query[N].sql`, where `[N]` is the number of the query as enumerated. Each query is worth 10 points. Be sure to follow the direction, as there are restrictions on the constructs that you can use for some of the queries. We have not provided you with any sample data for this part of the assignment, so it is suggested that you devise a way to test your scripts to ensure that they are correct. In all instances, do not duplicate rows in your results.

1. Write a query that finds the list of the IDs of all students who are enrolled in (EECS442 and EECS445 and EECS492) or (EECS482 and EECS486) or (EECS281). The results should be sorted in ascending order. Note that "EECS442" is a course title and that there is no space between the department abbreviation and the course number. **This query should be completed without views or nested queries; correct implementations of this query that utilize views or nested queries will receive half credit.**

2. Write a query that finds the IDs and names of all CS majors who are enrolled in at least one CS-heavy course. A CS-heavy course is defined as one in which strictly fewer than 50 non-CS majors are enrolled; this includes 0 non-CS majors. The results should be sorted in descending order by ID. A student whose major is CS will have the `VARCHAR2` value "CS" for their `Major` field, while a non-CS student will have something else. Remember that the `Major` field *can* be `NULL`.

3. Write a query that finds the IDs and names of all students with at least one project partner who is enrolled in (EECS482 or EECS483) and (EECS484 or EECS485) and (EECS280). The results should be sorted in descending order by the students' names. **This query should be completed without views or set operations (`MINUS`, `UNION`, `INTERSECT`); correct implementations of this query that utilize views or set operations will receive half credit.**

4. Create a view called `StudentPairs` with two columns, `SID1` and `SID2`. The contents of this view should be all pairs of IDs of two students who are enrolled in at least one common class but are not already partners on any project. The contents of the view do not need to be sorted.

Part 2: Booktown SQL (56 points)

A new bookstore called Booktown has opened in your suburb, and they've hired you to find some information about their current inventory. They've provided you with a database of their store's contents in `BuildBooktown.sql`, which defines the schema of the data tables as well as the data they contain. You should familiarize yourself with the schemas contained in this file, as they may contain information that is helpful in completing your queries. You can use the script file `DropBooktown.sql` to drop the database once you've finished using it.

Booktown wants to emphasize a few things about the schema of its database that you may accidentally overlook. Pay close attention to these points:

- There is no requirement that the pair `(First_Name, Last_Name)` for an author is unique; the same pair with a different `Author_ID` represents a different author
- Despite consisting only of numerals, the `ISBN` field of an edition is a string
- The `Publication_Date` field is a string that has the format `YYYY-MM-DD`, with the month and day being padded with a leading 0 where necessary; these values can be compared using standard comparators

Write each of the following queries in a separate file titled `BooktownQuery[N].sql`, where `[N]` is the number of the query as enumerated. Each query is worth 7 points. There are no restrictions on the methods you use to complete the query: nested queries, views, and set operations are all legal. Your queries must be correct *for any data set* and not just the one that we have provided; there is no guarantee that we will run your scripts against the exact data you have access to.

Be sure not to repeat results in the output of any of your queries. Be careful, though: just because the values of the fields you returned are the same in more than one row does not necessarily mean that they are the same result. This might happen if you are selecting only fields that are not guaranteed to be unique, as repeated tuples may then represent disparate entities.

1. Write a query that finds the ISBNs of all editions of books written by Agatha Christie. The results should be sorted in descending order.
2. Write a query that finds the IDs, first names, and last names of all authors who have written at least one book in every subject for which J. K. Rowling has written at least one book. The results should be sorted in ascending order by the author's last name, with ties being broken in favor of the larger ID.
3. Write a query that finds the first and last names of all authors who have written at least one children's/young adult book (subject: "Children/YA") and at least one book of fiction (subject: "Fiction"). The results should be sorted first in ascending order by first name and then in ascending order by last name.
4. Write a query that finds titles, publication dates, author IDs, author first names, and author last names of all editions of books written by an author who wrote at least one book with at least one edition published between the dates of 2003-01-01 and 2008-12-31; both dates should be included in the range. The results should be sorted in ascending order by the author's ID, then in ascending order by the book title, then in descending order by the date of publication.

5. Write a query that finds the titles of books and the cumulative total number of pages of all editions of each respective book. The column containing the cumulative total should be named `Total_Pages`. Only books that have editions listed need to be included in the results, which should be sorted in descending order by the cumulative total number of pages across all editions.

6. Write a query that finds the subjects for which no book has been written by any author. The results should be sorted in ascending order.

7. Write a query that finds the IDs and names of publishing companies that have published at least one edition of a book written by any author who has written exactly 3 books. The results should be sorted in descending order by the publisher's ID.

8. Write a query that finds the IDs of all authors who have written exactly 1 book. The results should be sorted in ascending order.

Please make sure that all of your queries execute in SQL*PLUS without syntax errors or undesired side effects. If you ever create an intermediate view, be sure that you drop in at the end of the file in which it was created. Under no circumstances should you alias the names of the columns *except* in `Query 5` as specified. Do not include any column formatting, SQL prompts, or other contents in your scripts that might change the format of your output. Do not tamper with the autocommit feature either.

Part 3: Relational Algebra (20 points)

Write relational algebra expressions equivalent to queries 1, 2, 3, and 4 listed above. You do not need to represent the sorted order of the results of the queries in your relational algebra expressions. Each expression is worth 5 points, and there may be a possibility for partial credit.

Submitting

Parts 1 and 2 will be submitted to the online Autograder, which should be open. You will have to create a team in order to submit: simply click on the "Create Team" button from the assignment home page and you will be set. Unlike for Project 1, you will not have the opportunity to invite classmates to join your team, as the assignment is to be completed individually. All 12 SQL script files should be placed in a single tarball named `homework2.tar.gz` and uploaded to the system. To make this tarball, place all of your SQL scripts in a the same directory with no other SQL files and run the following command:

```
% tar -zcf homework2.tar.gz *.sql
```

You will not be getting feedback from the Autograder for this assignment, other than a notification that your submission either did not contain the correct files (or was improperly named) or that your submission was accepted and graded.

If, for some reason, you are unable to complete one or more of the queries for Part 1 and/or Part 2, you still must submit a file with the appropriate name (but it can just be a blank file). If

you are missing any of the required files, your submission will *not* be counted in any capacity.

Part 3 will be submitted to Gradescope. The PDF of your relational algebra expressions can be named whatever you would like. Ensure that your uniqname is clearly visible on the PDF, which can either be fully computer-generated or a scan of something hand-written. The staff recommends formatting software such as LaTeX or Microsoft Word's formulas feature. If you choose to hand-write your solutions, give yourself plenty of space and write legibly.