CSCI 585 Spring 2021 Midterm Rubrics

Q1. In joining four tables A, B, C, D (where we want to list all the columns from all the tables), **what** can you say, about the join order (ie order matters, etc)? Explain, with simple illustrations.
A1. Join order will NOT matter, for **inner** joins - the operations are commutative, associative.

Join order DOES matter, for **left and right** outer joins (not commutative); order does NOT matter, for **full** outer joins!

No need to mention commutation/association. But, need to mention the above three cases.

+2 Correct answer for inner join case (+1 if partially correct and/or is close to correct answer)
+1 Correct answer for left join case (+0.5 if partially correct and/or is close to correct answer)
+1 Correct answer for right join case (+0.5 if partially correct and/or is close to correct answer)
+2 Correct answer for full outer join case (+1 if partially correct and/or is close to correct answer)

Q2. **What** is a new table operation can you think of, that won't break closure?

What about a new operation that does break closure?

Your operations don't need to be useful in practice, but, they could be :)

Note - changing column names/types, adding or deleting columns or rows are not considered table operations, as you know [so those can't be your answers :)] - other than that, you can come up with ANY operations!
A2.
Examples of a new operation that will not break closure: uppercasing all strings, filling in null values with defaults, removing (filtering) rows with ‘too many’ nulls, filtering out rows that look ‘too similar’ to other rows..

Examples of an operation that will break closure: std_deviation() of a column, or variance(), finding geom mean, harmonic mean…

The above are not the only answers, more might be possible.

Note: Operation preserving closure is the one, which when performed gives the same type of dataset as the one it is acting upon allowing to chain multiple operations together, one over another.
+3 for table operation that doesn’t break closure
+3 for table operation that does break closure

Q3. In the 2PL algorithm we considered, a transaction can't start until it has acquired all the locks it needs (we call this, Conservative 2PL).

There is a different scheme possible, where a transaction does NOT need to wait for all its locks - it can start its transactions before all the locks have been acquired.

What would be good about such a scheme, and, **what would be bad**? Do feel free to illustrate with a diagram.
A3.
A transaction that can start even while locks are being acquired, will lead to **higher throughput** (less waiting). On the flip side, the problem is, such a transaction, after starting, **could hang - on account of deadlock** when waiting for the remainder of its locks - so it would need to be aborted and restarted.

+3 correct answer for good point about scheme (+1.5 if partially correct and/or is close to correct answer)
+3 correct answer for bad point about scheme (+1.5 if partially correct and/or is close to correct answer)

Q4. When a webserver is set up to serve data (in addition to documents and hypermedia, ie ‘usual’ HTTP content), the stereotypical way it does so, is by fetching data from a DB (eg. via SQL, from a relational DB).

What are two other ways (sources) using which the webserver can send data to its clients?

Discuss each, using a sentence or two - be sure to maintain their utility (ie. why they would be useful).
A4.
The webserver can **compute data** to send - random number distributions, synthetic data for ML augmentation, geometric models using submitted/default parameters, etc. The webserver can also **measure or collect data** to send, eg. temperature, a picture of the night sky via a telescope… Or a webserver can use services to **request data** from other servers. All these are non-DB lookups, there might be more answers.

Different data sources must be mentioned NOT other protocols to send data like FTP, SMTP etc.
+3 for mentioning two different data sources (1.5 x 2)
+3 for explanation for each of the above two data sources (1.5 x 2)

Q5. A teacher would like to track the status of various tests (eg. in English, Math, Physics, Design...) each student in her class needs to complete, over a period of a month. A test has multiple steps, the number of steps can be different for different subjects (eg English has 3 steps, Design has 5). A student can do the steps in any order, eg. 1,3,2 for the English test. When the student does a step, the teacher records the subject, step#, completion date, in a table like so (a separate table exists for each student):

![p1.png](https://usercontent.crowdmark.com/fd3f8a55-25c8-4525-8d6b-0b95834537c1.png)

At the end of the testing period, the teacher runs this query (for each student):

![p2.png](https://usercontent.crowdmark.com/bf7fce4b-442c-43c4-ad81-787e7ed77a3f.png)

What does the query produce? In a few sentences, explain what it does (ie. how it does what it does).
A5.
The query outputs **completed tests** - for which all steps have been done by the student and therefore entered into the table by the teacher.

It works by **outputting the complement** (via NOT EXISTS) of entries where there is a NULL completion date [the triplets stored being subject,step,date-completed], ie, all fully completed tests. And, it **eliminates duplicates** (which would exist because of the multiple steps), via SELECT DISTINCT.

+2 for correct answer of outputting completed tests
+2 for explaining the logic of NOT EXISTS
+2 for explaining duplicate elimination with SELECT DISTINCT

Q6. In addition to the fact that a spreadsheet being used as a ‘database’ looks plain UGLY (!) on account of repeating groups [blocks of empty cells], what are **five** ‘real’ problems/issues with this?
A6.
Insertion anomaly - a new entry might need to be made, just to put in partial data (eg a new project that hasn’t formally started yet)

Deletion anomaly - deleting a row would delete valuable data not stored elsewhere.

Modification (update) anomaly - possible to introduce a typo, or omit to update an entry while updating all others…

Cumbersome to extend, in the future.

Unnecessary disk usage.

Unnecessary memory usage, bandwidth usage.

…

Total 5 points with breakdown as follows:
+3 for correct answer of 3 anomalies (insert, delete, update) (need not specify exact term, similar explanations also fetch full marks)
+2 for two more problems/issues that are logically valid and make sense