MLDS-413 Introduction to Databases and Information Retrieval

Homework 8: Triggers, Integrity Constraints, Transactions, Views, and Window Functions

Name 1: _____

NetID 1: _____

Name 2: _____

NetID 2: _____

Instructions

You should submit this homework assignment via Canvas. Acceptable formats are word files, text files, and pdf files. Paper submissions are not allowed and they will receive an automatic zero.

As explained during lecture and in the syllabus, assignments are done in groups. The groups have been created and assigned. Each group needs to submit only one assignment (i.e., there is no need for both partners to submit individually the same homework assignment).

Each group can submit solutions multiple times (for example, you may discover an error in your earlier submission and choose to submit a new solution set). We will grade only the last submission and ignore earlier ones.

Make sure you submit your solutions before the deadline. The policies governing academic integrity, tardiness and penalties are detailed in the syllabus.

SchoolScheduling.sqlite Database (30 points)

1. **(10 points)** Write the SQL statements that perform the following operations.
   a. **(1 point)** First, in preparation for this section of the homework, write and execute a query that inserts a new class status with ID 4 and description "Failed".

   b. **(3 points)** Start a new transaction.

   c. **(3 points)** Update the class status in student 1001's schedule for class 4180 to "Completed".

   d. **(3 points)** The problem now is that if the administrator forgets to set 1001's grade for class 4180, the student will have a class marked as completed with grade 0. You want to enforce a rule that a class cannot be marked completed unless the student has already received a passing grade, i.e., a grade of at least 60.0. You want to implement your integrity rules first and then do the data updates. Abort the transaction you started in part (b) in order to undo the changes in part (c). Do not undo the changes of part (a), though.

2. **(10 points)** Write a query that enforces the data integrity constraint described in question Q1.d.

3. **(10 points)** Write a query that enforces the following data integrity constraint: when a grade changes from a non-passing grade to a passing grade, automatically set the status of the class to completed. If the grade changes to a non-passing grade, set the class status to failed.

Homework 5 Question 6 Solution Database (10 points)

4. **(10 points)** Sometimes you want to only check integrity constraints, not enforce them. One way to do that is to create a view that you examine whenever you want to verify data integrity. One such example are the constraints (e), (g), (i), (k), and (n) in Homework 5 Question 6.

   For this assignment, you will use the Homework 5 Question 6 solution database. Your goal is to create a view named `check_db` that checks if the database violates any of the constraints (e), (g), (i), (k), and (n). The view should return a table that lists all the violated constraints, or an empty row if there are no violations. For example, if constraints e, g, i, k, and n are all violated, the view will be the table below (do not worry about the row order):

	ERRORS_FOUND
1	e
2	g
3	i
4	k
5	n

   Note that it is OK if some of the rows of your result are empty rows. Similarly, if no constraints are violated, the view could simply return a table with an empty row:

	ERRORS_FOUND
1	

To check that your view works properly, you can execute the following deletions on the Homework 5 Question 6 solution database and check the output of your query after each deletion set, as the comments and the SQL queries below show. The queries should be executed in the exact order below to achieve each of the stated results.

Suggestion: use transactions when you are experimenting in this question. This way, when things don't work, you can simply rollback the changes. Note that if you rollback once you will need to start a new transaction again to be able to rollback your changes a second time (`ROLLBACK` will undo your changes AND terminate the transaction). So, it is better to use `SAVEPOINT X` and `ROLLBACK TO X`. This way you can issue `ROLLBACK TO X` as many times as you want without taking a new savepoint (`ROLLBACK TO X` will undo the changes but the transaction remains active, so there is no need to remember to start a new one each time).

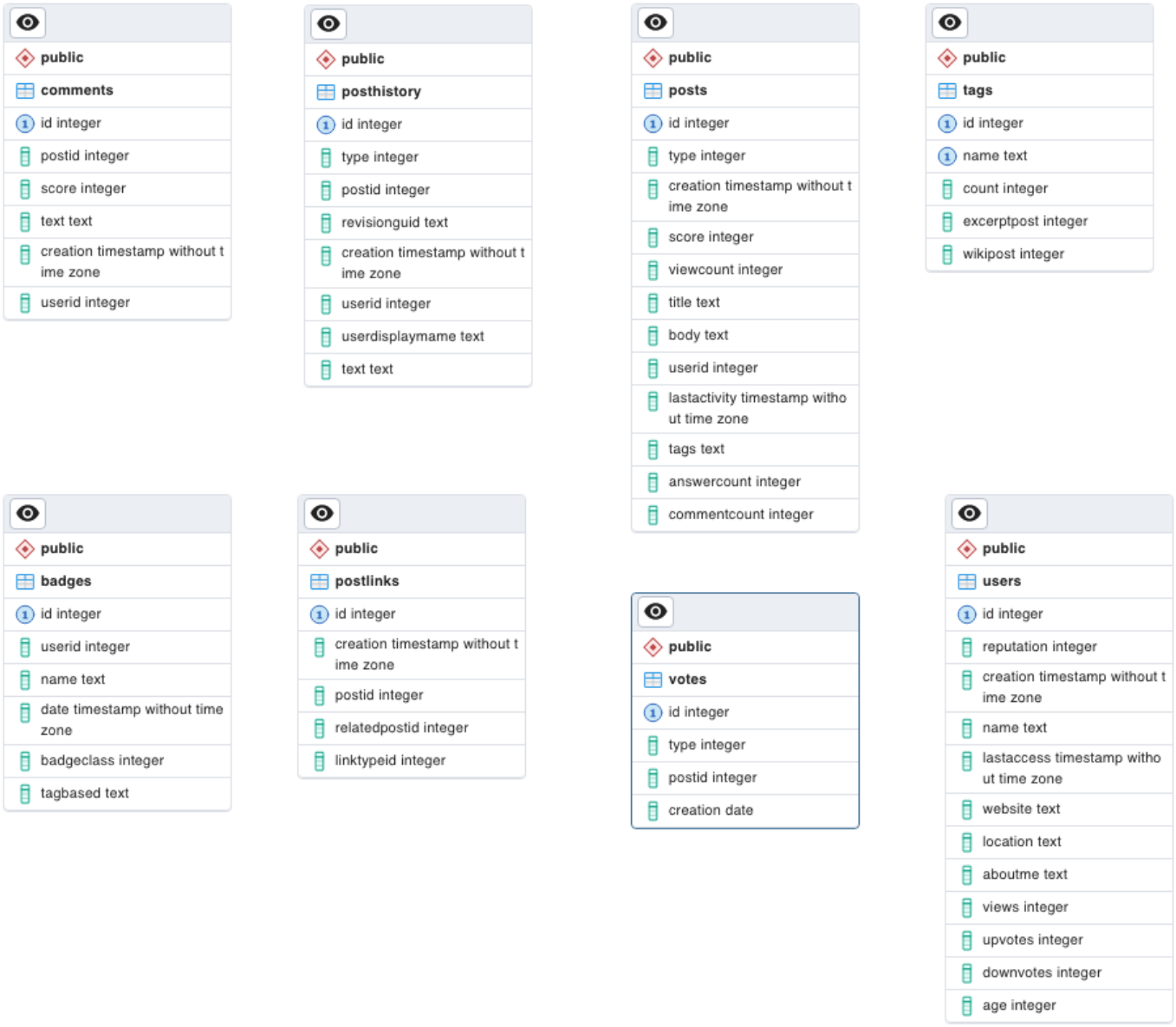
```
-- check that your view works by examining it on the HW5 Q6 solution database before any deletions
-- your view should be just an empty row (i.e., there are no violations)
SELECT * FROM check_db;

-- performing the following deletions would violate the following constraint
-- k. Each invoice has at least one invoice item
-- your view should contain a row for k
DELETE FROM invoice_items WHERE invoiceId IN (2, 3);
SELECT * FROM check_db;

-- performing the following additional deletions would violate the following additional constraint
-- e. Each album has at least one track
-- your view should contain rows for e, k
DELETE FROM tracks WHERE albumId=2;
SELECT * FROM check_db;

-- performing the following additional deletions would violate the following additional constraint
-- g. Each genre is represented by at least one track
-- your view should contain rows e, g, k
DELETE FROM tracks WHERE trackId=3451;
SELECT * FROM check_db;

-- performing the following additional deletions would violate the following additional constraint
-- i. Each media type is used by at least one track
-- your view should contain rows for e, g, i, k
DELETE FROM invoice_items WHERE trackId IN (SELECT trackId FROM tracks WHERE mediaTypeId=4);
DELETE FROM tracks WHERE mediaTypeId=4;
SELECT * FROM check_db;

-- performing the following additional deletions would violate the following additional constraint
-- n. Each customer has been issued at least one invoice
-- your view should contain rows for e, g, i, k, n
DELETE FROM invoice_items WHERE invoiceId IN (SELECT invoiceId FROM invoices WHERE customerId=20);
DELETE FROM invoices WHERE customerId=20;
SELECT * FROM check_db;
```

SalesOrders.sqlite Database (10 points)

5. **(10 points)** Monthly revenue growth is defined as the percent of revenue change of a month relative to the previous month, i.e., $(M_i - M_{i-1}) / M_{i-1}$. Write a query that will return the revenue growth of the sales in the SalesOrders database and provide your query's output. This should be a single query (CTE, windowing allowed).

Stackoverflow Database (50 points)

Please follow the instructions from Homework 6 to connect to the Stackoverflow (so) database on MLDS's Postgres server. The database schema is provided below:



Please note that the Stackoverflow database does not give any information about the relationship between different entities. You need to analyze each table and sample some data to **infer yourself** the relationships between tables. Unfortunately, the real world is often messy.

You will use this database to answer the following questions. Please make sure that your queries in this homework are read-only.

Unless otherwise noted, for each question please provide:

- The query you constructed
- The output of that query
- Any other information requested by the question

6. **(10 points)** How many posts are there that have never been edited after creation. Please provide **two different solutions** for this question. Hint: You can use many operations such as LEFT JOIN, EXCEPT, and EXIST.

7. **(10 points)** Write a SQL query to count the number of posts that were created on Christmas Day (December 25th) for each year. Present the results in ascending years.

8. **(10 points)** Rank users by their reputation and assign a percentile rank. Print the id, name, reputation, and the percentile rank of user 19787814 (user id).

9. **(10 points)** For the post with ID 7518463 in postlinks (i.e., postlinks.postid = 7518463), find the related post (directly or **indirectly**) with the highest number of answers. In this question, you should only consider the postlink with linktypeid = 1. Hint: You need to use recursive query in this question. The directly related posts can be found in the table postlinks. Note: This is a prime example where real-world data are messy! There is a postlinks.postid = 7518463 but not a posts.id = 7518463. Most likely the user removed the post before the database dump, and now we have dangling references in the linking table, because the database designer did not enforce integrity constraints. It seems the database designer didn't take MLDS-413!

10. **(10 points)** Find the month-over-month percentage growth in new posts in the year of 2022. Hint1: You may need to create some CTEs first. Hint2: You need to protect against the "divide by zero" error; divide only when it is safe to do so, otherwise set the corresponding percentage growth to NULL.