Homework Assignment 5

Maximum earnable: 51 pt.
Due: 11:59PM, June 5, 2025

- Read the assignment carefully. *You will need to* ***write and execute several SQL queries****; and* ***submit the results of your queries***.
- You are **allowed to re-use any of the queries from the lecture slides** while developing solutions to the problems.
- This is an individual work; Please be clear with HGU CSEE Standard:
  - Submitting assignments or program codes written by others or acquired from the internet without explicit approval of the professor is regarded as cheating.
  - Showing or lending one's own homework to other student is also considered cheating that disturbs fair evaluation and hinders the academic achievement of the other student.
  - It is regarded as cheating if two or more students conduct their homework together and submit it individually when the homework is not a group assignment.
- **Use of ChatGPT or similar AI tools**: Students are prohibited from using ChatGPT or similar AI platforms to directly obtain solutions for this assignment. The intent of the assignment is to exercise your understanding and application of the course material. Leveraging AI tools to bypass this learning process is considered a breach of academic integrity. Any evidence of such behavior will result in penalties.
- When finished, submit your work to *LMS*.

1. (1 pt. each) Read the textbook Chapters 5 and 17. Fill in the blanks with the correct answers.

(a) The (   ) function gives the same rank to all tuples that are equal on the ORDER BY attributes; whereas the (DENSE_RANK()) function does not create gaps in the ordering.

(b) The use of the keyword (   ) in place of **ROWS** allows the windowing query to cover all tuples with a particular value rather than covering a specific number of typles. Thus, **ROWS CURRENT ROW** refers to eactly one tuple, while **RANGE CURRENT ROW** refers to all tuples whose value for the sort attribute is the same as that of the current tuple.

(c) (   ) refers to a collection of operations that form a single logical unit of work.

2. (3 pt.) Match each of the following key types to the corresponding definition.

Primary key	●	●	Unique identifier attribute(s) chosen in a relation
Foreign key	●	●	Minimal subset of uniquely identifiable attribute(s)
Composite key	●	●	Attribute(s) that define(s) the relationship between relations
Alternate key	●	●	Any uniquely identifiable attribute(s)
Candidate key	●	●	Any key with more than one attribute
Super key	●	●	Minimal subset of uniquely identifiable attribute(s) that are NOT chosen to represent rows in a relation

3. (3 pt. each) Answer the following questions.

(a) According to the textbook description from Section 4.2, what is the main difference between views and named subqueries defined by **WITH**?

(b) List the conditions that an SQL view is said to be updatable.

(c) According to the description from Section 17.1, explain what the ACID properties of a database are.

(d) Discuss the differences between the following two queries in terms of the computational efficiency.

```
-   SELECT ID, RANK() OVER (ORDER BY GPA DESC) AS s_rank
    FROM student_grades;

-   SELECT ID, (1 + (
        SELECT COUNT(*)
        FROM student_grades B
        WHERE B.GPA > A.GPA)) AS s_rank
    FROM student_grades A
    ORDER BY s_rank;
```

4. Answer the following questions that are from the textbook exercise problem sets. You may refer to the Internet as well as the textbook for assistance; however, <u>your solution should contain your own ideas in your own language</u>.

(a) (Exercise 17.8) The *lost update* anomaly is said to occur if a transaction T_j reads a data item, then another transaction T_k writes the data item (possibly based on a previous read), after which T_j writes the data item. The update performed by T_k has been lost, since the update done by T_j ignored the value written by T_k.

i. (2 pt.) Give an example of a schedule shown the lost update anomaly.

ii. (3 pt.) Give an example schedule to show that the lost update anomaly is possible with the **read committed** isolation level.

iii. (3 pt.) Explain why the lost update anomaly is not possible with the **repeatable read** isolation level.

(b) (3 pt.; Exercise 5.9) Given a relation *nyse*(*year, month, day, shares_traded, dollar_volume*) with trading data from the New York Stock Exchange, write a query that lists each trading day in order of number of shared traded, and show each day's rank.

(c) (3 pt.; Exercise 5.23) Consider the *nyse* relation from Problem 4(b). For each month of each year, write a query that shows the total monthly dollar volume and the average monthly dollar volume for that month and the two prior months.
(You may want to use the hint suggested by the textbook.)

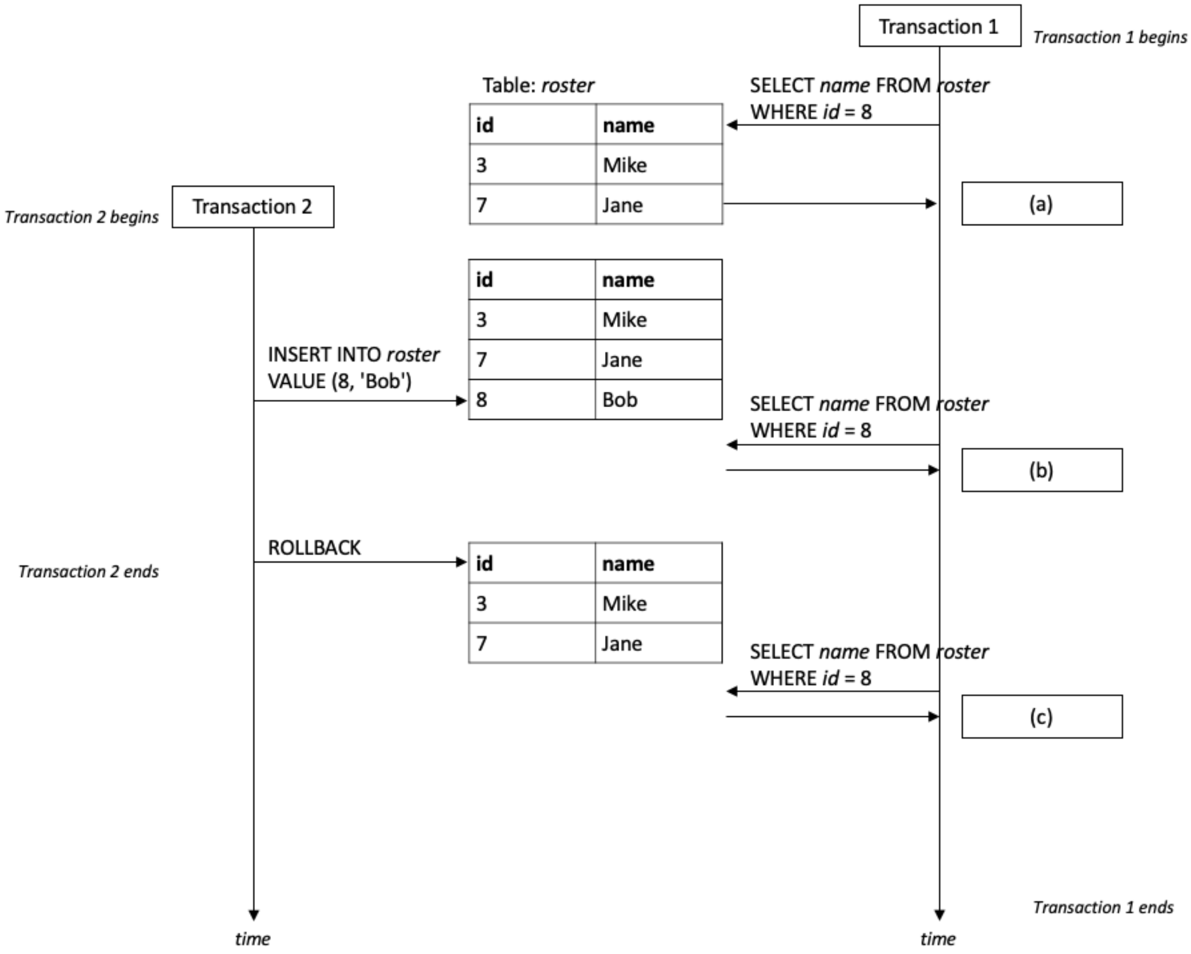
(d) (3 pt.; Exercise 5.8) Given a relation *S*(*student, subject, marks*), write a query to find the top 10 students by total marks, by using SQL ranking. Include all students tied for the final spot in the ranking, even if that results in more than 10 total students.

5. (3 pt. each) More query exercises. Launch and access the MySQL databases distributed with the class virtual machine. Below uses the "*sakila*" database (DVD rental database), which consists of 16 tables regarding movie inventory, actors, customers, rental history, payment information, *etc*. **For each of the following queries, evaluate and report the results. You may type in and execute each query to find the solution. Make sure you understand why you obtained the reported results.**

```
(a)  SELECT inventory_id,
         ROW_NUMBER() OVER (PARTITION BY inventory_id ORDER BY rental_date) AS ROW_NO,
         rental_id, customer_id, rental_date,
         LAG(rental_date, 1) OVER (PARTITION BY inventory_id ORDER BY rental_date) AS PREV_RENTAL
     FROM rental
     WHERE YEAR(rental_date) = 2005 AND MONTH(rental_date) = 8 AND inventory_id <= 5;
```

```
(b)  SELECT payment_id, customer_id, rental_id, payment_date, amount,
             SUM(amount) OVER (ORDER BY payment_date
             RANGE BETWEEN INTERVAL 0.5 DAY PRECEDING
                   AND INTERVAL 0.5 DAY FOLLOWING) AS DAILY_SUM
     FROM payment
     WHERE YEAR(payment_date) = 2005 AND MONTH(payment_date) = 5
         AND customer_id <= 5;
```

6. (10 pt.) Consider the following timelines where two transactions are intervening each other. The two vertical downward arrows represent the progression of time. The horizontal arrows represent the dataflow between transaction and storage.



SELECT *name* FROM *roster*
WHERE *id* = 8

id	name
3	Mike
7	Jane

INSERT INTO *roster*
VALUE (8, 'Bob')

id	name
3	Mike
7	Jane
8	Bob

SELECT *name* FROM *roster*
WHERE *id* = 8

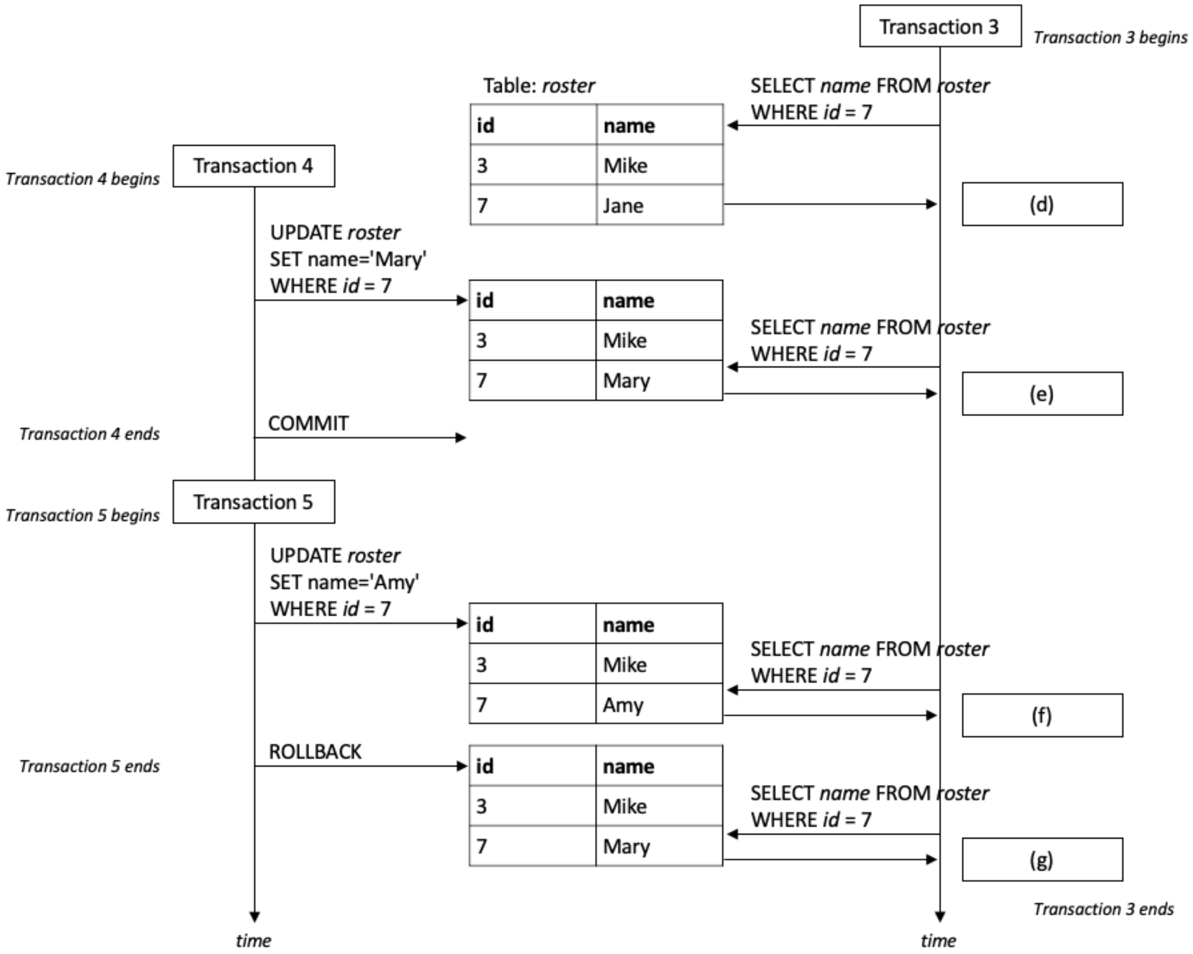
ROLLBACK

id	name
3	Mike
7	Jane

SELECT *name* FROM *roster*
WHERE *id* = 8

Assuming three isolation levels, REPEATABLE READ, READ COMMITTED, and READ UNCOMMITTED, what name would be returned after executing "SELECT name FROM roaster WHERE id = 8;"?

	REPEATABLE READ	READ COMMITTED	READ UNCOMMITTED
(a)			
(b)			
(c)			



Assuming three isolation levels, REPEATABLE READ, READ COMMITTED, and READ UNCOMMITTED, what name would be returned after executing "SELECT name FROM roaster WHERE id = 7;"?

	REPEATABLE READ	READ COMMITTED	READ UNCOMMITTED
(d)			
(e)			
(f)			
(g)			