COMP 5120/6120

Database Systems I
Spring 2023
Midterm Exam

Name: Caleb St.Germain
Student ID: 904068255

	Points	Received
Problem 1	20	
Problem 2	28	
Problem 3	28	
Problem 4	24	
Total	100	

Exam Rules:

1. Closed book, closed notes, **50 minutes**.
2. Please write down your name and student ID number.
3. Please wait until being told to start reading and working on the exam.

Problem 1 Database Concepts (20 points, 2 points each)

(1) A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, which is called the identifying owner.

[✓ **True**] **False**

(2) For a given view, an INSERT or UPDATE may change the underlying base table so that the resulting (i.e., inserted or modified) row is not in the view.

[✓ **True**] **False**

(3) The natural join of two relations *R*(A, B) and *S*(C, D), which have no common attribute, is equivalent to their full outer join.

True [✓ **False**]

(4) In a table, there is exactly one candidate key, but there can be multiple superkeys.

True [✓ **False**]

(5) Every relationship in an ER diagram must translate to an individual relation in the relational model.

True [✓ **False**]

(6) Two relation instances are said to be union-compatible on the only condition that they have the same number of fields.

[✓ **True**] **False**

(7) SQL supports the creation of assertions, which are constraints not associated with any single table.

[✓ **True**] **False**

(8) Only columns that appear in the GROUP BY clause can appear in the HAVING clause, unless they appear as arguments to an aggregate operator in the HAVING clause.

[✓ **True**] **False**

(9) The expression, COUNT(*), handles null values just like other values; that is, they get counted.

True [✓ **False**]

(10) Once the schema of a database is set, it cannot be changed.

True [✓ **False**]

Problem 2 Relational Algebra (28 points)

Consider the following relations containing airline flight information:

Flights(*flno*: integer, *from*: string, *to*: string, *distance*: integer, *departs*: time, *arrives*: time)
Aircraft(*aid*: integer, *model*: string, *cruising-range*: integer)
Certified(*eid*: integer, *aid*: integer)
Employees(*eid*: integer, *ename*: string, *salary*: integer)

The Employees relation describes pilots and other kinds of employees as well. Every pilot is certified for some aircraft (otherwise, he or she would not qualify as a pilot), and only pilots are certified to fly. Write the following queries in ***relational algebra***.

(1) Find the *names* of pilots certified for some Boeing aircraft. (9 points)

Relational Algebra:

$$\pi_{ename}(\sigma_{aname}=\text{‘Boeing'}\ (\text{Aircraft} \bowtie \text{Certified} \bowtie \text{Employees}))$$

(2) Identify the flights that can be piloted by every pilot whose salary is more than \$100,000. (9 points)

$$\pi_{flno}(\sigma_{distance<cruisingrange\ \wedge\ salary\ >\ 100{,}000}(\text{Flights} \bowtie \text{Aircraft} \bowtie \text{Certified} \bowtie \text{Employees})))$$

(3) Find the *aids* of all aircraft that can be used on non-stop flights from Atlanta to Los Angeles. (10 points)

$$\rho(AtlantaToLosAngeles,\ \sigma_{from=\text{'Atlanta'} \wedge to=\text{LosAngeles'}}\ (Flights))$$
$$\pi_{aid}(\sigma_{cruisingrange>distance}(Aircraft\ x\ AtlantaToLosAngeles\ ))$$

Problem 3 SQL (28 points)
Consider the following schemas:

Employee (eid, name, office, age)
Books (isbn, title, authors, publisher)
Loan (eid, isbn, date)

Write the following queries in **SQL**.

(1) Print the names of employees who have borrowed any book published by Auburn University Press. (8 points)

```
SELECT DISTINCT e.name
FROM Employee e
JOIN Loan l ON e.eid = l.eid
JOIN Books b ON l.isbn = b.isbn
WHERE b.publisher = 'Auburn University Press'
```

(2) Print the names of employees who have borrowed all books published by Auburn University Press. (10 points)

```
WITH auburn_books AS (
   SELECT isbn
   FROM Books
   WHERE publisher = 'Auburn University Press'
GROUP BY isbn
   HAVING COUNT(*) = (
   SELECT COUNT(*)
   FROM Books
   WHERE publisher = 'AuburnUniversity Press'
    )
)
SELECT e.name
FROM Employee e
JOIN Loan l ON e.eid = l.eid
JOIN auburn_books ab ON l.isbn = ab.isbn
GROUP BY e.name
HAVING COUNT(DISTINCT l.isbn) = (SELECT COUNT(*) FROM auburn_books)
```

(3) For each publisher, print the names of employees who have borrowed more than two books of that publisher. (10 points)

```
SELECT b.publisher, e.name
FROM Employee e
JOIN Loan l ON e.eid = l.eid
JOIN Books b ON l.isbn = b.isbn
GROUP BY b.publisher, e.name
HAVING COUNT(*) > 2
```

<u>Problem 4</u> SQL (24 points, 8 points each)
The following three tables contain the information of students, departments, and student-department relationships. The primary key of each table is underlined. For each query, <u>write the **output**</u> *and* <u>provide **a brief description** of the query</u>.

Student

<u>**sid**</u>	**sname**	**gpa**	**age**
9001	John	3.7	22
9003	Jason	3.3	27
9004	Brian	2.8	26
9005	Amanda	3.0	25
9006	Lauren	3.0	23
9007	Chris	3.4	24

Student_Department

<u>**sid**</u>	<u>**did**</u>	**startdate**
9001	D001	8/16/2021
9003	D002	1/10/2020
9004	D002	8/16/2022
9005	D001	8/16/2019
9006	D003	1/10/2021
9007	D004	8/16/2021

Department

<u>**did**</u>	**dname**
D001	Computer Science
D002	Electrical Engineering
D003	Civil Engineering
D004	Mechanical Engineering

```
(1) SELECT s.sname, d.dname, sd.startdate
    FROM Student s, Student_Department sd, Department d
    WHERE s.sid = sd.sid and d.did = sd.did
```

The output of the first query would be a table with three columns: sname, dname, and startdate.

The first query joins the Student, Student_Department, and Department tables and uses the WHERE clause to link them based on the sid and did columns. This allows it to retrieve the student`s name, department's name, and start date of the student's relationship with the department.

Output:

```
sname     dname                     startdate
John      Computer Science          8/16/2021
Jason     Electrical Engineering    1/10/2020
Brian     Electrical Engineering    8/16/2022
Amanda    Computer Science          8/16/2019
Lauren    Civil Engineering         1/10/2021
Chris     Mechanical Engineering    8/16/2021
```

```
(2) SELECT did, dname
    FROM Department
    WHERE did NOT IN (SELECT did FROM Student_Department GROUP BY did
    HAVING COUNT(*) > 1)
```

The second query selects the did and dname columns from the Department table. It uses a subquery to check for departments that have more than one student and excludes them from the result using the NOT IN clause. This effectively retrieves departments with only one student.

Output:

```
did     dname
D001    Computer Science
D002    Electrical Engineering
```

```
(3) SELECT s.sname, s.gpa, d.dname
    FROM Student s, Student_Department sd, Department d
    WHERE s.sid = sd.sid AND d.did = sd.did AND s.gpa = (SELECT MIN (gpa)
    FROM Student)
```

The third query selects the student name, gpa, and department name from the Student, Student_Department, and Department tables using a WHERE clause to join them based on their common columns, sid and did. It adds an additional condition to filter the result based on the student with the lowest GPA using a subquery to find the minimum GPA in the Student table, and then comparing it to the gpa column in the main query. The output shows the student with the lowest GPA and their corresponding department name.

Output:

```
sname       gpa    dname
Brian       2.8    Electrical Engineering
Lauren      3.0    Civil Engineering
Amanda      3.0    Computer Science
```