CSE 182 Midterm Exam, Spring 2023, Shel Finkelstein

ANSWERS

Student Name: _____

Student ID: _____

UCSC Email: _____

Midterm Points:

Part	Max Points
I	30
II	20
III	24
IV	27
Total	101

Closed book, but it's okay to bring a single two-sided 8.5" x 11" sheet of paper with as much info written or printed on it as you can read unassisted. Please hand in Midterm (but **not** your sheet, and **not** the Relation Instances sheet you've been given) when you finish the Midterm. You must also show your UCSC ID when you hand in the Midterm.

Be sure to answer each question readably in the space provided for that question.

Part I: (30 points, 6 each):

Question 1: Assume that R1 is a relation that has 6 tuples in it, R2 is a relation that has 3 tuples in it, and R3 is a relation that has 10 tuples in it. How many tuples are there in the result of the following SQL query?

```
SELECT *
FROM R1, R2, R3;
```

Just supply your answer; no part credit.

Answer 1: 180 = 6 * 3* 10

Question 2: Let R(A,B,C) be a relation, where (A, B) is the Primary Key for that relation, and C cannot be NULL.

Assume that A's domain has 5 different values, B's domain has 7 different values, and C's domain has 12 different values.

What is the maximum number of different tuples that can be in R?

Just supply your answer; no part credit.

Answer 2: 35 = 5 * 7

Question 3: What does each of the following statements do? Answer both parts of this question clearly and precisely in the spaces provided.

3a): DELETE FROM Students;
Answer 3a):

The statement deletes all of the tuples that are in the Students table, but the Students table still exists.

3b): DROP TABLE Students;
Answer 3b):

The Students table is deleted, including all of the tuples/rows that were in that table.

Question 4 We discussed the ACID properties for transactions. The letter "D" in ACID stands for Durability. Briefly explain what Durability means, and also say why Durability is important.

Answer 4:

Durability for a transaction means that if the commit succeeds for a transaction, its data changes are entered permanently in the database, even if there are failures of components such as computers, memory and storage devices. However, subsequent modifications may change that data.

Durability is important because once an application or user is told that a transaction has committed (e.g., transfer of money between bank accounts), the effects of that transaction should really have happened, and should not go away.

Question 5: SQL uses 3-valued logic. Give the SQL truth value of each of the following conditions if the value of salary1 is NULL and the value of salary2 is 2000.

5a): salary1 > 500 OR salary2 > 500

Answer 5a): TRUE

5b): salary1 > 500 OR salary1 <= 500

Answer 5b): UNKNOWN

5c): NOT (salary1 = salary2)

Answer 5c): UNKNOWN

Part II (20 points, 4 each): The questions in Part II are about an Employees table that was created as follows:

```
CREATE TABLE Employees (
        name        CHAR(30) PRIMARY KEY,
        age         INTEGER NOT NULL DEFAULT(21),
        salary      INTEGER,
        department  CHAR(20) NOT NULL
    );
```

Answer **TRUE** or **FALSE** to each of the following questions, writing out the full word.

Question 6: Attributes age and department can't be NULL, but the other attributes of Employees can be NULL.

Answer 6: FALSE (since the Primary Key, name, also can't be NULL)

Question 7: The result of the query:

```
SELECT COUNT(age)
FROM Employees;
```

can sometimes be greater than the result of the query:

```
SELECT COUNT(DISTINCT age)
FROM Employees;
```

Answer 7: TRUE (since there might be duplicate values of age)

Question 8: The following query is a legal SQL query:

```
SELECT salary, MIN(age), MAX(age)
FROM Employees
GROUP BY department
HAVING salary > 9000;
```

Answer 8: FALSE (since salary is not a GROUP BY attribute)

```
CREATE TABLE Employees (
        name        CHAR(30) PRIMARY KEY,
        age         INTEGER NOT NULL DEFAULT(21),
        salary      INTEGER,
        department  CHAR(20) NOT NULL
    );
```

Question 9: The following two SQL queries are equivalent:

```
SELECT e.name
FROM Employees e
WHERE e.age > ANY ( SELECT e2.age
                    FROM Employees e2 );

SELECT e.name
FROM Employees e
WHERE EXISTS ( SELECT *
               FROM Employees e2
               WHERE e.age > e2.age );
```

Answer 9: TRUE

Question 10: To require that two employees who are in the same department can't have the same salary, we would specify that department is UNIQUE, and that salary is UNIQUE.

Answer 10: FALSE (since the correct constraint wouldn't be UNIQUE(department) and UNIQUE(salary), it would be UNIQUE(department, salary))

Part III: (24 points, 6 each):

Questions 11-14 in this Midterm Part ask for the results of SQL queries on the instances of the tables Customers, Slopes and Activities that are on the separate sheet of paper that you've been given.

Do not turn in that page at the end of the exam.

Show attribute names at the top of all SQL outputs for all questions in Part III.

Question 11: What is the result of the following SQL query?

```
SELECT DISTINCT color
FROM Slopes s
WHERE s.slopeid IN ( SELECT a.slopeid
                     FROM Activities a );
```

Answer 11:

```
color
---------
Blue
Black
```

Tuples may appear in any order.
Okay to have tuple variable appearing with name of attribute at top of result.

Question 12: What is the result of the following SQL query?

```
SELECT type, MAX(age) AS oldest
FROM Customers
GROUP BY type
ORDER BY type;
```

Answer 12:

type	oldest
ski	33
snowboard	25

Tuples must appear in this order.
Okay to have tuple variable appearing with name of attribute at top of result.

Question 13: What is the result of the following SQL query?

```
SELECT DISTINCT a.cid, a.day
FROM Activities a, Slopes s
WHERE a.slopeid = s.slopeid
      AND s.name = 'Mountain Run';
```

Answer 13:

cid	day
36	1/06/09
36	1/07/09
87	1/07/09

Tuples may appear in any order.
Okay to have tuple variable appearing with name of attribute at top of result.

Question 14: What is the result of the following SQL query?

```
SELECT c.cname, s.name
FROM Activities a, Customers c, Slopes s
WHERE a.cid = c.cid
      AND a.slopeid = s.slopeid
      AND a.day <= DATE '01/06/09';
```

Answer 14:

cname	name
Cho	Magic Carpet
Cho	Mountain Run
Luke	Olympic Lady

Tuple may appear in any order.
Okay to have tuple variable with name of attribute at top of result.

Part IV (27 points, 9 each):

The questions in Part IV ask you to write SQL statements using the tables shown below, which are 4 of the tables in our Lab Assignments. (The CandidatesForOffice tables has been simplified, omitting some unneeded attributes.)

The Primary Key in each table is shown underlined. Assume that there aren't any NOT NULL or UNIQUE constraints specified for these tables. Data types aren't shown to keep things simple. There aren't any trick questions about data types.

Persons(personID, personName, city, state, occupation, isFelon)

ElectedOffices(officeID, officeName, city, state, salary)

Elections(officeID, electionDate, officeStartDate, officeEndDate)

CandidatesForOffice(candidateID, officeID, electionDate, party, votes)

You should assume the following Foreign Keys:

- Every officeID in Elections appears as an officeID in ElectedOffices.
- Every candidateID in CandidatesForOffice appears as a personID in Persons.
- Every (officeID, electionDate) in CandidatesForOffice appears as an (officeID, electionDate) in Elections.

Tables and attributes are repeated at the top of each question, with Primary Keys underlined.

Be sure to answer each question readably in the space provided below that question.

Persons(personID, personName, city, state, occupation, isFelon)
ElectedOffices(officeID, officeName, city, state, salary)
Elections(officeID, electionDate, officeStartDate, officeEndDate)
CandidatesForOffice(candidateID, officeID, electionDate, party, votes)

Question 15: An election is identified by (officeID, electionDate), where officeID is the elected office for that election.

Write a SQL query which finds all the elections for which:

- the electionDate is March 19, 2022 or later,
- the city for that election's elected office starts with 'Santa' (with that capitalization), and
- the salary for that election's elected office isn't NULL.

The attributes in your result should appear as theOfficeID, theElectionDate, theCity and theSalary.

No duplicates should appear in your result.

Answer 15:

Use of DISTINCT is not necessary in this query, since there can't be duplicates. But no deduction in CSE 182 if you use DISTINCT.

```
SELECT eo.officeID AS theOfficeID, e.electionDate AS theElectionDate,
                    eo.city AS theCity, eo.salary AS theSalary
FROM Elections e, ElectedOffices eo
WHERE e.officeID = eo.officeID
    AND e.electionDate >= DATE '19-03-23'
    AND eo.city LIKE 'Santa%'
    AND eo.salary IS NOT NULL;
```

- Spacing doesn't matter.
- Okay to use other tuple variables, or use table names instead of tuple variables, or omit table names completely if attributes aren't ambiguous.
- Order of tables in the FROM clause doesn't matter.
- Okay to omit AS in the SELECT clause and include AS in the FROM clause.
- Order of AND'ed conditions doesn't matter.
- Okay to write date constant as DATE '03/19/23' or DATE '3/19/23'.

Persons(personID, personName, city, state, occupation, isFelon)
ElectedOffices(officeID, officeName, city, state, salary)
Elections(officeID, electionDate, officeStartDate, officeEndDate)
CandidatesForOffice(candidateID, officeID, electionDate, party, votes)

Question 16: party is an attribute in the CandidatesForOffice table that identifies the political party of the candidate for office in an election. For example, party is 'Gold' for a candidate who's running for office in the Gold party.

Write a SQL query that finds personName and occupation for each person who never was a candidate running for office in the Gold party.

The tuples in your result should be in reverse alphabetical order based on personName. No duplicates should appear in your result.

Answer 16:

Here are several correct solutions. Using DISTINCT is necessary, since the query only asks for personName and occupation, and there could be multiple persons who have the same personName and occupation who satisfy the query.

```
SELECT p.personName, p.occupation
FROM Persons p
WHERE NOT EXISTS ( SELECT *
                   FROM CandidatesForOffice c
                   WHERE p.personID = c.candidateID
                      AND c.party = 'Gold' )
ORDER BY p.personName DESC;
```

```
SELECT p.personName, p.occupation
FROM Persons p
WHERE p.personID NOT IN ( SELECT c.candidateID
                          FROM CandidatesForOffice c
                          WHERE c.party = 'Gold' )
ORDER BY p.personName DESC;
```

```
SELECT p.personName, p.occupation
FROM Persons p
WHERE p.personID != ALL ( SELECT c.candidateID
                          FROM CandidatesForOffice c
                          WHERE c.party = 'Gold' )
ORDER BY p.personName DESC;
```

There are more complicated correct solutions using LEFT OUTER JOIN and using EXCEPT, which I don't expect many students to provide.

- Spacing doesn't matter.
- Okay to use other tuple variables, or use table names instead of tuple variables, or omit table names completely if attributes aren't ambiguous.
- Order of tables in the FROM clause doesn't matter.
- Okay to omit AS in the SELECT clause and include AS in the FROM clause.
- Order of AND'ed conditions doesn't matter.

Persons(personID, personName, city, state, occupation, isFelon)
ElectedOffices(officeID, officeName, city, state, salary)
Elections(officeID, electionDate, officeStartDate, officeEndDate)
CandidatesForOffice(candidateID, officeID, electionDate, party, votes)

Question 17: As the previous question told you, party is an attribute in the CandidatesForOffice table that identifies the political party of the candidate for office in an election. For example, party is 'Gold' for a candidate who's running for office in the Gold party.

We'll say that a person is a Programmer if the occupation for that person is 'Programmer'. Write a SQL query which finds all the parties for which at least 6 candidates for office are Programmers. But do not include a tuple for the Gold party in your result.

The attributes in your result should appear as party and numProgrammers, where numProgrammers is the number of candidates for office from that party who are Programmers.

No duplicates should appear in your result.

Answer 17:

Using DISTINCT isn't necessary, but for CSE 182, it's okay to use DISTINCT in any of these solutions, since you're asked that no duplicates appear in result.

```
SELECT c.party, COUNT(*) AS numProgrammers
FROM Persons p, CandidatesForOffice c
WHERE p.personID = c.candidateID
    AND p.occupation = 'Programmer'
    AND c.party != 'Gold'
GROUP BY c.party
HAVING COUNT(*) >= 6;
```

It's also okay, to exclude the Gold party from the result by using the HAVING clause.

```
SELECT c.party, COUNT(*) AS numProgrammers
FROM Persons p, CandidatesForOffice c
WHERE p.personID = c.candidateID
    AND p.occupation = 'Programmer'
GROUP BY c.party
HAVING COUNT(*) >= 6 AND c.party != 'Gold';
```

- Spacing doesn't matter.
- Okay to use other tuple variables, or use table names instead of tuple variables, or omit table names completely if attributes aren't ambiguous.
- Order of tables in the FROM clause doesn't matter.
- Okay to omit AS in the SELECT clause and include AS in the FROM clause.
- Order of AND'ed conditions doesn't matter.
- Can use COUNT(a), where a is any attribute that can't be NULL, instead of COUNT(*). Attributes in the Primary Keys of FROM clause tables can't be NULL, and attributes compared in AND'ed conditions in the WHERE clause can't be NULL.