CSE 180, Midterm Exam, Fall 2021, Shel Finkelstein
ANSWERS

Student Name: ______________________________

Student ID: ______________________________

UCSC Email: ______________________________

Midterm Points

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

Please show your **UCSC ID** when you turn in your exam.

This exam is single sided, and you may write an answer on the blank side of a page, if you need more space. But if you do that, clearly identify question that you've answered.

Note: For questions that ask you to write SQL statement, points might be deducted if you write complicated queries, even if they are correct. You can lose credit if you use DISTINCT but it's not needed, and you will lose credit if you don't use DISTINCT and it is needed.

Part I: (30 points, 6 points each):

Question 1: R(A, B, C) is a relation instance. If R has 5 tuples in it, then how many tuples are there in the result of the following SQL query?

```
SELECT *
FROM R r1, R r2;
```

Answer 1: _____25____

Question 2: Let R(A,B,C,D) be a relation, where (A, B) is the Primary Key for R, attribute C can be NULL, and attribute D cannot be NULL. Assume that attribute A’s domain has 5 different values, B’s domain has 3 different values, attribute C’s domain has 2 different values and attribute D’s domain has 4 different values.

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

Answer 2: _____15____

Question 3: In our database, we have a table Movies in which (movieTitle, movieYear) is the Primary Key:

Movies (movieTitle , movieYear, length , genre , studioName , producerC#)

and a view PixarMovies that was created as follows:

```
CREATE VIEW PixarMovies AS
    SELECT movieTitle, movieYear
    FROM Movies
    WHERE studioName = 'Pixar';
```

Assume that ('Toy Story', 1995, 81, TRUE, 'Pixar', 7358) is a tuple in the Movies table, and that it is the only tuple in the Movies table whose studioName is 'Pixar'. (Yes, genre is Boolean; there's no trick here.)

3a): What is the result of the query? Show attribute names at the top of your answer.

```
SELECT * FROM PixarMovies;
```

Answer 3a):

```
movieTitle          movieYear
---------------------------------------------
Toy Story           1995
```

Okay if you put quotes around 'Toy Story'

3b): Suppose that we execute:

DELETE FROM Movies WHERE **movieTitle** = 'Toy Story' AND **movieYear** = 1995;

After this DELETE, what is the result of that same SELECT query? Show attribute names at the top of your answer.

```
SELECT * FROM PixarMovies;
```

Answer 3b):)

```
movieTitle              movieYear
---------------------------------------------
```

Okay if you wrote "Empty Set" below the attribute names, but attributes should be shown.

WHERE clause had originally had title and year instead of movieTitle and movieYear, so we'll give full credit to answers that say that the result is the same as for 3a).

3c): Now let's assume that there are 50 Pixar movies in the Movies table. Explain what happens to the tuples in the Movies table after we execute:

```
DROP VIEW PixarMovies;
```

Answer 3c):

The tuples in the Movies table are unchanged; nothing happens to them when the PixarMovies view is dropped.

Question 4: If a tuple appears 20 times in the result for Q1 and that exact same tuple appears 4 times in the result for Q2, then:

4a): How many times would that tuple appear in Q1 UNION ALL Q2?

Answer 4a): ____24_____

4b): How many times would that tuple appear in Q1 INTERSECT ALL Q2?

Answer 4b): _____4_____

4c): How many times would that tuple appear in Q1 EXCEPT ALL Q2?

Answer 4c): ____16_____

Question 5: Your answers to the following questions should provide a brief explanation, not just some buzzwords. Give just one answer for each.

5a): The SQL language is described as Declarative (or Non-Procedural). Explain why SQL is described as Declarative.

Answer 5a):

SQL queries are described as Declarative because they indicate what the result should be done, not how it the result should be computed.

5b): The D in database ACID properties stands for Durability. Explain what Durability means.

Answer 5b):

Durability means that when database transactions commit, their effects don't go away, even if there are software or hardware failures.

Part II: (24 points, 8 points each):

Questions 6-8 are about the Scores(Team, Day, Opponent, Runs) relation instance that appears below. (Team, Day) is the Primary Key for Scores.

Be sure to show attribute names at the top for all your SQL query results!

Scores(Team, Day, Opponent, Runs)

Team	Day	Opponent	Runs
Dragons	Sunday	Swallows	3
Tigers	Sunday	Bay Stars	10
Carp	Sunday	Giants	2
Swallows	Sunday	Dragons	10
Bay Stars	Sunday	Tigers	1
Giants	Sunday	Carp	4
Dragons	Monday	Carp	9
Tigers	Monday	Bay Stars	9
Carp	Monday	Dragons	5
Swallows	Monday	Giants	0
Bay Stars	Monday	Tigers	6
Giants	Monday	Swallows	4

Question 6: For this instance of Scores, what is the result of the following SQL query?

```
SELECT Team, MIN(Runs) AS theMin
FROM Scores
GROUP BY Team;
```

Answer 6:

```
Team            theMin
--------------------------------
Dragons         3
Tigers          9
Carp            2
Swallows        0
Bay Stars       1
Giants          4
```

The result tuples may appear in any order.

Question 7: For this instance of Scores, what is the result of the following SQL query?

```
SELECT S1.Day, S1.Team
FROM Scores S1, Scores S2
WHERE S1.Day = S2.Day
      AND S1.Team = S2.Opponent
       AND S1.Runs > S2.Runs
ORDER BY S1.Team;
```

Answer 7:

```
Day             Team
----------------------------------
Monday          Dragons
Sunday          Giants
Monday          Giants
Sunday          Swallows
Sunday          Tigers
Monday          Tigers
```

Okay if the attribute names are S1.Day and S1.Team.
The ordering is by Team.
The two tuples for Giants may appear in either order.
The two tuples for Tigers may appear in either order.

(The query finds the winning teams on each day.)

Question 8: For this instance of Scores, what is the result of the following SQL query?

```
SELECT S1.Day, S1.Team
FROM Scores S1
WHERE S1.Runs =
   ( SELECT MAX(S2.Runs)
     FROM Scores S2
     WHERE S2.Day = S1.Day );
```

Answer 8:

Day	Team
Sunday	Tigers
Sunday	Swallows
Monday	Dragons
Monday	Tigers

The query is always legal SQL since MAX(Runs) always returns one value.
Okay if the attribute names are S1.Day and S1.Team.
The result tuples may appear in any order.

Part III: (20 points, 4 points each)

The following **TRUE** or **FALSE** questions all refer to the Customers table that is created by the following statement:

```
CREATE TABLE Customers (
        cid             INTEGER,
        name            VARCHAR(20) UNIQUE,
        type            VARCHAR(20),
        level           VARCHAR(20) DEFAULT ‘Advanced’ NOT NULL,
        age             INTEGER,
        PRIMARY KEY (cid)
        );
```

Answer each of these questions with **TRUE** or **FALSE**. No explanation is required, and no part credit will be given.

Question 9: TRUE or **FALSE:** If there are no tuples in the Customers table, and we execute the statement:

```
      INSERT INTO Customers(cid, name)
        VALUES (41, ‘Chou’);
```

then the database system will return an error.

Answer 9: ___FALSE_____

Question 10: TRUE or **FALSE:** The following is a legal SQL query.

```
SELECT level, type, COUNT(*), MIN(age)
FROM Customers
WHERE type != ‘snowboard’
GROUP BY level;
```

Answer 10: ___FALSE_____

Question 11: TRUE or **FALSE:** The following two SQL statements are equivalent.

```
DELETE *                    DROP TABLE Customers;
FROM Customers;
```

Answer 11: ____**FALSE** _____

Question 12: TRUE or **FALSE:** The following two SQL queries are equivalent.

```
SELECT COUNT(*)             SELECT COUNT(level)
FROM Customers;             FROM Customers;
```

Answer 12: ____**TRUE**_____

Question 13: TRUE or **FALSE**: The following two queries are equivalent.

```
SELECT c.cid
FROM Customers c
WHERE c.level != ANY ( SELECT c2.level
                       FROM Customers c2
                       WHERE c2.type = 'snowboard' );
```

```
SELECT c.cid
FROM Customers c
WHERE c.level NOT IN ( SELECT c2.level
                       FROM Customers c2
                       WHERE c2.type = 'snowboard' );
```

Answer 13: ___**FALSE**_____

Part IV: (27 points, 9 points each): The questions in Part IV ask you to write SQL statements using the tables shown below, which are 3 of the tables in our Lab Assignments, with some of the attributes removed.

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 thing simple. There aren't any trick questions about data types.

Persons(personID, lastName, firstName, isStudent)

Conferences(conferenceName, year, conferenceDate, regularAttendeeCost, studentAttendeeCost, submissionDueDate)

Attendees(attendeeID, conferenceName, year)

You should assume that the following Foreign Key relationships hold.

- Every attendeeID in Attendees appears as a personID in persons.
- Every (conferenceName, year) in Attendees appears as a (conferenceName, year) in Conferences.

Points might be deducted if you write complicated queries, even if they are correct. You can lose credit if you use DISTINCT but it's not needed, and you will lose credit if you don't use DISTINCT and it is needed.

You don't have to use views to answer any of these questions, but you may use views if you want, as long as you create them correctly before you use them.

Question 14: Write a SQL statement that finds the personID, lastName and firstName for each person whose first name has lowercase 't' as its second letter, and who attended a conference whose conferenceDate attribute isn't NULL.

The attributes in your result should appear as thePerson, theLastName and theFirstName.

No duplicates should appear in your result.

Answer 14:

```
SELECT DISTINCT p.personID AS thePerson, p.lastName AS theLastName,
                p.firstName AS theFirstName
FROM Persons p, Conferences c, Attendees a
WHERE p.personID = a.attendeeID
      AND c.conferenceName = a.conferenceName
      AND c.year = a.year
      AND p.firstName LIKE '_t%'
      AND c.conferenceDate IS NOT NULL;
```

- DISTINCT is needed in this solution, since a person may have attended many such conferences.
- Okay to leave out the appearances of the keyword AS in the SELECT clause.
- Okay to use other tuple variables, or use table names instead, or use neither if the attribute is unambiguous.

Here's another correct answer using a subquery; there are others.

```
SELECT p.personID AS thePerson, p.lastName AS theLastName,
                p.firstName AS theFirstName
FROM Persons p
WHERE p.firstName LIKE '_t%'
      AND EXISTS ( SELECT *
                        FROM Conferences c, Attendees a
                        WHERE p.personID = a.attendeeID
                              AND c.conferenceName = a.conferenceName
                              AND c.year = a.year
                              AND c.conferenceDate IS NOT NULL );
```

- DISTINCT is not needed in this solution, since no person is considered more than once, and personID is the Primary Key of Persons.
- Okay to leave out the appearances of the keyword AS in the SELECT clause.
- kay to use other tuple variables, or use table names instead, or use neither if the attribute is unambiguous.
- For an EXISTS subquery, it doesn't matter what you SELECT, as long as it's legal SQL.

Question 15: Write a SQL statement that finds the conferenceName, year, regularAttendeeCost and studentAttendeeCost of each conference for which

- studentAttendeeCost is more than regularAttendeeCost, and
- there are <u>no attendees</u> for that conference.

Tuples in the result should appear in alphabetical order based on conferenceName. If result tuples have the same conferenceName, then tuples which have later years should appear before tuples which have earlier years.

No duplicates should appear in your result.

Answer 15:

```
SELECT c.conferenceName, c.year, c.regularAttendeeCost, c.studentAttendeeCost
FROM Conferences c
WHERE c.studentAttendeeCost > c.regularAttendeeCost
    AND NOT EXISTS ( SELECT *
                     FROM Attendees a
                     WHERE a.conferenceName = c.conferenceName
                         AND a.year = c.year )
ORDER BY c.conferenceName, c.year DESC;
```

- DISTINCT <u>is not needed</u> in this solution, since no conference is considered more than once, and the SELECT clause includes the Primary Key for Conferences.
- Okay to write "c.conferenceName ASC" in the ORDER BY clause.
- Okay to use other tuple variables, or use table names instead, or use neither if the attribute is unambiguous.
- For a NOT EXISTS subquery, it doesn't matter what you SELECT, as long as it's legal SQL.

Here’s another correct way to write this.

```
SELECT c.conferenceName, c.year, c.regularAttendeeCost, c.studentAttendeeCost
FROM Conferences C
WHERE c.studentAttendeeCost > c.regularAttendeeCost
    AND (c.conferenceName, c.year) NOT IN
                ( SELECT a.conferenceName, a.year)
                  FROM Attendees a )
ORDER BY c.conferenceName, c.year DESC;
```

- DISTINCT is not needed in this solution, since no conference is considered more than once, and the SELECT clause includes the Primary Key for Conferences.
- Okay to write “c.conferenceName ASC” in the ORDER BY clause.
- Okay to use other tuple variables, or use table names instead, or use neither if the attribute is unambiguous.

Question 16: Persons has a Boolean attribute **isStudent** that indicates whether the person is a student.

For each conference whose submissionDueDate is before December 5, 2020, find the number of students who attended that conference. But you should only include a tuple for a conference in your result if the number of students who attended that conference is more than 25.

The 3 attributes in your result should be conferenceName, year, and the number of students who attended the conference, which should appear as numStudentAttendees.

No duplicates should appear in your result.

Answer 16:

```
SELECT c.conferenceName c.year, COUNT(*) AS numStudentAttendees
FROM Persons p, Conferences c, Attendees a
WHERE p.personID = a.attendeeID
    AND c.conferenceName = a.conferenceName
   AND c.year = a.year
   AND p.isStudent
   AND c.submissionDate < DATE '2020-12-05'
GROUP BY c.conferenceName, c.year
HAVING COUNT(*) > 25;
```

- DISTINCT is not needed in this solution, since each group only show up once, and conferenceName and year are the GROUP BY attributes.
- Okay to use other tuple variables, or use table names instead, or use neither if the attribute is unambiguous.
- Okay to write"p.isStudent = TRUE" or even "p.isStudent IS TRUE" (which we didn't discuss in class), instead of "p.isStudent". You can also write T instead of TRUE.
- Okay to write DATE '12/05/20' instead of DATE '2020-12-05'.
- Okay to have COUNT of some attribute in the SELECT and HAVING clauses, as long as that attribute can't be NULL. (Primary Key attributes and the isStudent can't be NULL, since the WHERE clause won't be true if isStudent is NULL.)
- Can't use numStudentAttendees in the HAVING clause because it's defined in the SELECT clause. (Yes, this may work in some implementations, but not in the SQL standard.)