CSE 180, Midterm Exam, Fall 2019, Shel Finkelstein

Student Name: _____

Student ID: _____

UCSC Email: _____

Midterm Points

Part	Max Points	Points
I	24	
II	24	
III	24	
IV	30	
Total	102	

You may use a double-sided 8.5 x 11 sheet with anything that you like written or printed on it during the Midterm. Sheets may not be shared during the Midterm. No devices may be used during the Midterm.

Please show your **UCSC ID** at the end of the exam.

This exam is single sided, and you may write answers on the blank side of a page, if you need more space. But if you do that, clearly identify the 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 might also lose half a point if you use DISTINCT but it's not needed. But you will definitely lose credit if you don't use DISTINCT and it is needed.

Part I: (24 points, 6 each):

Question 1: If R is a relation instance that has 5 tuples in it, S is a relation instance that has 8 tuples in it, and T is a relation instance that has 3 tuples in it, then how many tuples are there in the Cartesian Product (R x S) x T?

Answer 1: ____**120**__

Question 2: If R(A,B) is a relation where A's domain is (a1, a2, a3, a4) and B's domain is (b1, b2, b3), what the maximum number of <u>different</u> tuples that can be in an instance of R, assuming that A <u>can</u> also be **NULL**, but B <u>can't</u> be **NULL**?

Answer 2: ____**15**___

Question 3: If a tuple appears 20 times in the answer to Q1, and that same tuple also appears 6 times in the answer to Q2, then:

3a): How many times would that tuple appear in Q1 **EXCEPT** Q2?

Answer 3a): ____**0**____

3b): How many times would that tuple appear in Q1 **EXCEPT ALL** Q2?

Answer 3b): ____**14**___

Question 4: SQL uses 3-valued logic, with truth values TRUE, FALSE and UNKNOWN.

If the value of *age1* is **NULL** and the value of *age2* is 18, give the truth value for each of the following:

4a): *age1* = *age2*

Answer 4a): **__UNKNOWN____**

4b): *age1* **IS NOT NULL**

Answer 4b): **____FALSE____**

4c): *age1* > 20 **OR** *age2* <= 20

Answer 4c): **____TRUE____**

Part II: (24 points, 6 each):

Questions 5-8 are about the instance of the table Scores that is on the page at the back of the test. You may tear off that page to do these questions. You do not need to turn in that page after the exam.

What is the result of each of the following SQL queries? Be sure to show attributes at the top of all of your SQL results.

Question 5:

SELECT DISTINCT *Team*
FROM *Scores*
WHERE *Team* **LIKE** '%n%';

Answer 5:

Team

Dragons
Giants

Tuples may appear in any order.

Question 6:

SELECT *Opponent*, *Day*
FROM *Scores*
WHERE *Runs* >= 6
ORDER BY *Day*, *Opponent* **DESC**;

Answer 6:

Opponent	Day
Tigers	**Monday**
Carp	**Monday**
Dragons	**Sunday**
Bay Stars	**Sunday**

Tuples must appear in this precise order.

Question 7:

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

Answer 7:

Team	Day
Carp	Sunday
Bay Stars	Sundays
Swallows	Monday

Tuples may appear in any order.
Okay to have S1.Team and S1.Day as the attributes.

Question 8:

```
SELECT Team, MIN(Runs) AS MinRuns, MAX(Runs) AS MaxRuns
FROM Scores
WHERE Opponent IN ('Swallows', 'Bay Stars', 'Giants')
GROUP BY Team;
```

Answer 8:

Team	MinRuns	MaxRuns
Dragons	4	4
Tigers	5	9
Carp	2	2
Swallows	0	0
Giants	5	5

Tuples may appear in any order.

Part III: (24 points, 4 each)

The following **TRUE** or **FALSE** questions refer to the *customer* table created by the following statement:

CREATE TABLE *customer* (

cid	**INTEGER PRIMARY KEY**,
name	**VARCHAR**(20),
type	**VARCHAR**(20) **DEFAULT**('snowboard') **NOT NULL**,
level	**VARCHAR** (20),
age	**INTEGER NOT NULL**);

Answer each question TRUE or FALSE. No explanation required, no part credit.

Question 9: TRUE or **FALSE:** The only attributes of the *customer* table that can have the value **NULL** are *name* and *level*.

Answer 9: ___**TRUE**___

Question 10: TRUE or **FALSE**:

The following statement deletes the *customer* table, as well as all the tuples in that table.

DELETE FROM *customer*;

Answer 10: ___**FALSE**___

Question 11: TRUE or **FALSE:** The following two SQL queries are equivalent, meaning that they always have the same answer on any instance of the *customer* table that corresponds to the above **CREATE** statement.

SELECT COUNT(*)	**SELECT COUNT**(**DISTINCT** *age*)
FROM *customer*;	**FROM** *customer*;

Answer 11: ___**FALSE**___

Question 12: TRUE or **FALSE:** The following is a legal SQL query.

```
SELECT level, MIN(age), type, MAX(age)
FROM customer
WHERE age > 21
GROUP BY level, type
ORDER BY type;
```

Answer 12: ___**TRUE**___

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

```
SELECT C.name
FROM customer C
WHERE C.age <> ( SELECT C2.age
                 FROM customer C2
                 WHERE C2.level = 'Advanced');
```

```
SELECT C.name
FROM customer C
WHERE C.age NOT IN ( SELECT C2.age
                     FROM customer C2
                     WHERE C2.level = 'Advanced');
```

Answer 13: ___**FALSE**___

Question 14: TRUE or **FALSE**: Suppose that another table called *activity* has been created in the same database as *customer* by the following statement

```
CREATE TABLE activity (
        cid             INTEGER,
        slopeid         INTEGER
        day             DATE,
        PRIMARY KEY (cid, slopeid, day),
        FOREIGN KEY (cid) REFERENCES customer   );
```

Then for any *cid* value that appears in a tuple of *customer*, there must be a tuple in *activity* that has the same *cid* value.

Answer 14: ___**FALSE**___

Part IV: (30 points, 10 each): The questions in Part IV ask you to write SQL statement using the tables shown below, which are simplified versions of the tables in our Lab Assignments. There are just 4 tables, and they have fewer 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. You may assume the same Referential Integrity constraints that we had in our Lab Assignment tables. Data types aren't shown to keep thing simple. There aren't any trick questions about data types.

customer(custID, name, address, joinDate)
menuItem(menuItemID, name, price)
visit(visitID, custID, cost)
billEntry(visitID, menuItemID, quantity)

- There could be other correct ways of writing these queries. There may be a deduction if you write complicated queries, even if they are correct.
- Capitalization and indenting don't affect correctness of answers, except for the lowercase 'e' in Q15.
- Choice of tuple variables (or relation names) doesn't affect correctness of answers.
- Okay to use AS or not use AS with tuple variables in the FROM clause and when giving aliases for attributes in the SELECT clause.

Question 15: Write a SQL statement that gives just the names and addresses of customers whose addresses have the letter 'e' (lowercase 'e') as their second character.

The tuples in your result should be alphabetized by name. No duplicates should appear in your result.

Answer 15:

- DISTINCT is needed, since there could be duplicate name/address values.
- Okay to use other tuples variable or not use tuple variable.

```
SELECT DISTINCT C.name, C.address
FROM Customer C
WHERE C.address LIKE '_e%'
ORDER BY C.name;
```

customer(custID, name, address, joinDate)
menuItem(menuItemID, name, price)
visit(visitID, custID, cost)
billEntry(visitID, menuItemID, quantity)

Question 16: Write a SQL statement that outputs the menuItemID, name and price for each menuItem that DOES NOT appear in any billEntry tuple. Order your result so that highest prices come first and lowest prices comes last. No duplicates should appear in your result.

Answer 16:

- Distinct is not needed in these solution since there can't be multiple menuItem tuples that have the same menuItemID.
- Okay to use other tuples variables, or not use tuple variables where there is no attribute ambiguity. Could also use relation names instead of tuple variables.

```
SELECT M.menuItemID, M.name, M.price
FROM menuItem M
WHERE M.menuItemID NOT IN (
                    SELECT B.menuItemID
                    FROM billEntry B)
ORDER BY M.price DESC;
```

```
SELECT M.menuItemID, M.name, M.price
FROM menuItem M
WHERE M.menuItemID != ALL (
                    SELECT B.menuItemID
                    FROM billEntry B )
ORDER BY M.price DESC;
```

Could use <> instead of != in this solution.

```
SELECT M.menuItemID, M.name, M.price
FROM menuItem M
WHERE NOT EXISTS ( SELECT B.menuItemID
                   FROM billEntry B
                   WHERE B.menuItemID = M.menuItemID )
ORDER BY M.price DESC;
```

Need tuple variables (or relation names) in this solution to distinguish menuItem attributes.

customer(custID, name, address, joinDate)
menuItem(menuItemID, name, price)
visit(visitID, custID, cost)
billEntry(visitID, menuItemID, quantity)

Question 17: A "New Customer" is a customer whose joinDate was June 1, 2018 or later.. Write a SQL statement that for each New Customer outputs the average cost of visits by that customer. But DON'T include a New Customer in your output unless that New Customer had at least one visit whose cost was greater than 80.

Output the custID, name and average cost. Your result should have attributes cID, cName and averageCost. No duplicates should appear in your result.

Answer 17:

Need tuple variable (or relation name) to distinguish custID attributes in customer and visit. Okay to use other tuple variables, or use relation names.

DISTINCT is not needed in the following solutions, since there will only be one tuple in the result for each group, and each such tuple will have a different custID value.

It's okay to have GROUP BY with just custID (instead of having GROUP BY with both custID and name) because custID is the key of the customer table, so custID determines name, per slide 69 of Lecture 4.

```
SELECT C.custID AS cID, C.name AS cName, AVG(V.cost) AS averageCost
FROM customer C, visit V
WHERE C.custID = V.custID
   AND C.joinDate >= DATE '2018-06-01'
GROUP BY C.custID, C.name
HAVING SOME ( V.cost > 80 );
```

```
SELECT C.custID AS cID, C.name AS cName, AVG(V.cost) AS averageCost
FROM customer C, visit V
WHERE C.custID = V.custID
   AND C.joinDate >= DATE '2018-06-01'
   AND C.custID IN ( SELECT V2.custID
                     FROM visit V2
                     WHERE V2.cost > 80 )
GROUP BY C.custID, C.name;
```

Okay not to have tuple variable V2 in the subquery following IN.

This instance of the Scores table is referenced in Part II (Questions 5-8 of the Midterm). You may tear off this page to do these questions. You don't have to turn in this page after the exam.

Here are the Scores from the Japanese Baseball League. (Team, Day) is the primary key of the Scores table.

Scores

Team	Day	Opponent	Runs
Dragons	Sunday	Swallows	4
Tigers	Sunday	Bay Stars	9
Carp	Sunday	Giants	2
Swallows	Sunday	Dragons	7
Bay Stars	Sunday	Tigers	2
Giants	Sunday	Carp	4
Dragons	Monday	Carp	6
Tigers	Monday	Bay Stars	5
Carp	Monday	Dragons	3
Swallows	Monday	Giants	0
Bay Stars	Monday	Tigers	7
Giants	Monday	Swallows	5