CSE 180, Final Exam, Fall 2019, Shel Finkelstein

Student Name: ______________________________

Student ID: ______________________________

UCSC Email: ______________________________

Final Points

Part	Max Points	Points
I	42	
II	24	
III	36	
Total	102	

The first Section (Part I) of the Fall 2019 CSE 180 Final is multiple choice and is double-sided. Answer all multiple choice questions on your Scantron sheet. You do not have to hand in the first Section of the Exam, but you must hand in the Scantron sheet, with your name, email and student id on that Scantron sheet. Please be sure to use a #2 pencil to mark your choices on this Section of the Final.

This separate second Section (Parts II and III) of the Final is not multiple choice and is single-sided, so that you have extra space to write your answers. If you use that extra space, please be sure to write the number of the problem that you're solving next to your answer. Please write your name, email and student id on this second Section of the Exam, which you must hand in. You may use any writing implement on this Section of the Exam.

At the end of the Final, please be sure to hand in your **Scantron sheet** for the first Section of the Exam and also **this second Section of the Exam**, and show your **UCSC id** when you hand them in. Also hand in your **8.5 x 11 "cheat sheet"**, with your name written in the upper right corner of the sheet.

Part II: (24 points, 6 points each)

Question 22: Assume that we already have:

- a *person* table, whose Primary Key is *SSN*, and
- a *house* table, whose Primary Key is *houseID*.

Write a Create statement for another table:

property(*owner, houseID, datePurchased*)

A *property* tuple indicates that a person (*owner*) purchased a house (*houseID*) on the specified datePurchased. *owner* and *SSN* are integers, and *houseID* is an integer in both *house* and *property*; *datePurchased* is a date.

The Primary Key of *property* is (*owner, houseID*). In the *property* table that you create, *owner* should be a Foreign Key corresponding to the Primary Key of *person* (*SSN*), and *houseID* should be a Foreign Key corresponding to the Primary Key of *house* (also called houseID).

Moreover, when a *person* is deleted, tuples in *property* which that person owns should be deleted. But deletion of a tuple in *house* is not permitted if there is a tuple in *property* for that *houseID*. (You don't have to consider updates in your Create.)

Answer 22:

Question 23:

23a): Explain the meaning of Physical Independence for Relational Database.

Answer 23a):

23b): Why are Relational Database queries described as Declarative?

Answer 23b):

23c): What's the connection between Physical Independence and Declarative queries for Relational Database?

Answer 23c):

Question 24: We have the following relations:

Sailors(sid, sname, rating, age) // sailor id, sailor name, rating, age
Boats(bid, bname, color) // boat id, boat name, color of boat
Reserves(sid, bid, day) // sailor id, boat id, date that sailor sid reserved boat bid.

Codd's relational algebra for sets included only the 5 operators Selection(σ), Projection(π), Product(x), Union ($\cup$) and Difference (-). Using <u>only those 5 operators</u>, write a Relational Algebra query that finds the names of sailors whose age is more than 20, but who <u>have not reserved</u> any boat whose color is red.

If you'd like, you may also use Rename (ρ) and Assignment ($\leftarrow$) in your answer.

Very Big Hint: In SQL, you can use NOT EXIST or NOT IN or COUNT, but you can't use any of those in Relational Algebra.

But you <u>can</u> use Difference. So first, find the names of sailors whose age is more than 20 and who <u>have reserved</u> a boat whose color is red. Then use Difference to find the answer.

Answer 24:

Question 25: Suppose that you have a relation:

BookInfo(Author, Citizenship, Title, PageCount)

with the following Functional Dependencies:

Author $\rightarrow$ Citizenship

Author, Title $\rightarrow$ PageCount

25a) Is {Author, Title} a key for BookInfo? Justify your answer fully and clearly.

Answer 25a):

25b) Is BookInfo in BCNF? Justify your answer fully and clearly.

Answer 25b):

25c) Is BookInfo in 3NF? Justify your answer fully and clearly.

Answer 25c):

Part III: (36 points, 9 points each)

Some familiar tables appear below, with Primary Keys underlined. **These tables also appear on the last page of the Final, which you can tear off to help you do questions in Part III of the Final.** You don't have to turn in that last page at the end of the Exam.

customer(<u>custID</u>, name, address, joinDate, status)

menuItem(<u>menuItemID</u>, name, description, price)

dinnerTable(<u>dinnerTableID</u>, numSeats, InUse)

server(<u>serverID</u>, name, level, salary)

visit(<u>visitID</u>, custID, dinnerTableID, serverID, numPeople, cost, custArrive, custDepart)

billEntry(<u>visitID, menuItemID</u>, quantity)

Assume that no attributes can be NULL, and that there are no UNIQUE constraints.

You may assume Referential Integrity as follows:

- Each custID in visit appears as a Primary Key in customer.
- Each dinnerTableID in visit appears as a Primary Key in dinnerTable.
- Each serverID in visit appears as a Primary Key in server.
- Each visitID in billEntry appears as a Primary Key in visit.
- Each menuItemID in billEntry appears as a Primary Key in menuItem.

Write legal SQL queries for Questions 26-29. If you want to create and then use views to answer these questions, that's okay, but views are not required unless the question asks for them.

Don't use DISTINCT in your queries unless it's necessary, 1 point will be deducted if you use DISTINCT when you don't have to do so. And some points may be deducted for queries that are very complicated, even if they are correct.

Question 26: Find the menuItemID, name and price for every menuItem that has the highest price of any menuItem. Note that there may be more than one menuItem that has that highest price. Your result should appear in reverse alphabetical order based on name. No duplicates should appear in your result

Answer 26:

Question 27: Find the ID and name of each server whose level is ‘B’ and who was the server for a visit in which ‘Apple Pie’ was one of the items on the bill. No duplicates should appear in your result.

Answer 27:

Question 28: For each visit, there may be billEntry tuples for that visit. Let's define a *billedVisit* to be a visit for which there is at least one billEntry tuple.

Each billEntry tuple has a quantity, and each billEntry tuple has a menuItemID that identifies a menuItem tuple that has a price. So the calculatedCost of a *billedVisit* can be computed by adding up price*quantity for all of the billEntry tuples for that *billedVisit*.

The visit table has a cost attribute. Find all the *billedVisits* for which the cost of that visit is not equal to the calculated cost of that visit. In your result, the attributes should be visitID, cost and calculatedCost. No duplicates should appear in your result.

Answer 28:

Question 29: This question has two parts; be sure to answer both.

29a): Create a SQL view called custVisitCount. For each customer who joined before December 25, 2018, the custVisitCount view should give the custID, address and the number of visits for that customer. In your view, the attributes should be called theCustomer, theAddress and theVisitCount.

[Hint: There could be customers who have no visits, and theVisitCount for them should be 0. If your view works correctly just for customers who do have visits, you will only lose 1 point for that.]

29b): <u>Use the view custVisitCount</u> to find the address of each customer such that:

- the customer visited 8 or more times, and also
- no other customer in the customer table has the same address.

[Yes, you'll also have to use the customer table to do this.]

The only attribute in your result should be theAddress. No duplicates should appear in your result.

Answers 29a) and 29b):

This is extra blank space, in case you need more paper to answer questions. Write Question number clearly if you use blank pages.

You may tear off this page to help you do Part III of the Final.

Some familiar tables appear below, with Primary Keys underlined. **These tables also appear on the last page of the Final, which you can tear off to help you do questions in Part III of the Final.** You don't have to turn in that last page at the end of the Exam.

customer(custID, name, address, joinDate, status)

menuItem(menuItemID, name, description, price)

dinnerTable(dinnerTableID, numSeats, InUse)

server(serverID, name, level, salary)

visit(visitID, custID, dinnerTableID, serverID, numPeople, cost, custArrive, custDepart)

billEntry(visitID, menuItemID, quantity)

Assume that no attributes can be NULL, and that there are no UNIQUE constraints.

You may assume Referential Integrity as follows:
- Each custID in visit appears as a Primary Key in customer.
- Each dinnerTableID in visit appears as a Primary Key in dinnerTable.
- Each serverID in visit appears as a Primary Key in server.
- Each visitID in billEntry appears as a Primary Key in visit.
- Each menuItemID in billEntry appears as a Primary Key in menuItem.

Write legal SQL queries for Questions 26-29. If you want to create and then use views to answer these questions, that's okay, but views are not required unless the question asks for them.

Don't use DISTINCT in your queries unless it's necessary, 1 point will be deducted if you use DISTINCT when you don't have to do so. And some points may be deducted for queries that are very complicated, even if they are correct.