CMPS 182, Final Exam, Spring 2019, Shel Finkelstein

Student Name: ______________________________

Student ID: ______________________________

UCSC Email: ______________________________

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

The first Section (Part I) of the Spring 2019 CMPS 182 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: Here are statements creating two relations R and S:

```
CREATE TABLE S (
   c INT PRIMARY KEY,
   d INT
    );
```

```
CREATE TABLE R (
   a INT PRIMARY KEY,
   b INT,
   FOREIGN KEY(b)
         REFERENCES S(c)
            ON UPDATE CASCADE
            ON DELETE SET NULL
    );
```

Relation R(a,b) currently contains the four tuples, (0,4), (1,5), (2,4), and (3,5).
Relation S(c,d) currently contains the four tuples, (2,80), (3,81), (4,82), and (5,83).

Indicate all changes that happen to both R and S when the following SQL statements are executed upon the R and S instances that are shown above. That is, you should **ignore changes made by earlier parts** of this question when you answer later parts.

22a): UPDATE S set c=7 WHERE d=82;

Answer 22a):
Changes made to the original R:

Changes made to the original S:

22b): DELETE FROM R WHERE a =2;

Answer 22b):
Changes made to the original R:

Changes made to the original S:

22c): DELETE FROM S WHERE c=5;

Answer 22c):
Changes made to the original R:

Changes made to the original S:

Question 23a): What are advantages of having Stored Procedures in a database? Give <u>two different</u> clear advantages. If you give more than two, only first two will be graded.

Answer 23a):
<u>Advantage 1</u>:

<u>Advantage 2</u>:

Question 23b): Since Stored Procedures have the advantages that you listed in part a), why are there also other approaches that combine programming language constructs with SQL?

Answer 23b):

Question 24: In the dashed lines, write **YES** if the statement is correct, and **NO** if the statement is incorrect.

24a) If the value of the attribute salary1 is NULL, then the truth value for the predicate "salary1 = 1000" will be UNKNOWN.

Answer a) ________________

24b) If the value of the attribute salary2 is NULL, then the truth value for the predicate "salary2 = NULL" will be UNKNOWN.

Answer b) ________________

24c) If the value of the attribute salary1 is 1000, and the value of the attribute salary2 is NULL, then the truth value for the predicate "salary1 = 1000 AND salary2 = 1000" will be UNKNOWN.

Answer c) ________________

24d) A tuple contributes to the result of a query only if its WHERE clause evaluates to TRUE, not if its WHERE evaluates to FALSE or UNKNOWN.

Answer d) ________________

25e) A CHECK constraint is violated if it evaluates to FALSE or UNKNOWN.

Answer e) ________________

24f) Assume that our database has a table Customers(cid, cname, level, type, age). If there are no customers whose type is 'ski', then the result of the following query will always be the empty set.

```
SELECT c.name
FROM Customers c
WHERE c.age > ALL  ( SELECT c2.age
                     FROM Customers c2
                     WHERE c2.type = 'ski' );
```

Answer f) ________________

Question 25: You have a relation Company_Info(Emp, Dept, Manager), with the following non-trivial Functional Dependencies:

$$\begin{aligned}&\text{Emp} \rightarrow \text{Dept}\\&\text{Emp} \rightarrow \text{Manager}\\&\text{Dept} \rightarrow \text{Manager}\end{aligned}$$

25a): Is Company_Info in 3NF? Justify your answer clearly.

Answer 25a)

25b): In Design Theory, what is the Update Anomaly? Explain why it is a problem by giving a clear example.

Answer 25b)

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 hand in that last page at the end of the Exam.

Products(productID, productName, manuf, normalPrice, discount)

Customers(customerID, custName, address, joinDate, amountOwed, lastPaidDate, status)

Stores(storeID, storeName, region, address, manager)

Days(dayDate, category)

Sales(productID, customerID, storeID, dayDate, paidPrice, quantity)

Payments(customerID, paidDate, amountPaid, cleared)

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

Assume Referential Integrity as follows:
- Each productID in Sales appears as a productID in Products.
- Each customerID in Sales appears as a customerID in Customers.
- Each storeID in Sales appears as a storeID in Stores.
- Each dayDate in Sales appears as a dayDate in Days.
- Each customerID in Payments appears as a customerID in Customers.

Write legal SQL queries for Questions 25-28. 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 do. Some points will also be deducted for unnecessarily complex queries, including use of GROUP BY when it's not needed.

Question 26: Find the ID, name and address for each customer whose name begins with ‘Prof’ and who has no payments that have cleared. (cleared is a Boolean attribute in Payments.) Order your result by name in reverse alphabetical order. Duplicates should not appear in your result.

Answer 26:

Question 27: Customers whose status is ‘H’ are high-status customers. For each high-status customer, find the number of different products that they purchased. The two attributes appearing in your result should be called customerID and numDiffProducts.

But only include a customer in your result if they purchased at least 4 different products. No duplicates should appear in your result.

Answer 27:

Question 28: paidPrice and quantity are attributes in the Sales table. The cost of a sale in Sales equals paidPrice times quantity.

For each store in Stores, give the storeID, storeName, the number of sales for that store and the total of the costs of the Sales for that Store. The last two attributes should appear in your result as numSales and totalCost. No duplicates should appear in your result.

Note that if there are no sales in Sales for a particular store, there your result should include a tuple for that store with storeID, storeName, and with numSales and totalCost both 0. If your result does not provide this, then you'll lose 2 points.

Answer 28:

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

29a): In Products, there are attributes normalPrice and discount. The discounted price of a product equals normalPrice – normalPrice*discount/100.00

Create a view named ProductsDiscounted. For each product, this view should provide productID, manufacturer (manuf) and the discounted price of the product. The attributes in your view should appear as productID, manuf and discountedPrice. No duplicates should appear in your result.

29b): Products from the same manufacturer might have different discounted prices, but there also could be multiple products that have the same discounted price.

Write a query using the ProductsDiscounted view that gives the productID, manufacturer and discountedPrice for each product that has the highest discountedPrice of any product made by that product's manufacturer. If there are multiple products from a manufacturer that have the same highest discountedPrice, then there should be a tuple in your result for each of those products.

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 hand in that last page at the end of the Exam.

Products(productID, productName, manuf, normalPrice, discount)

Customers(customerID, custName, address, joinDate, amountOwed, lastPaidDate, status)

Stores(storeID, storeName, region, address, manager)

Days(dayDate, category)

Sales(productID, customerID, storeID, dayDate, paidPrice, quantity)

Payments(customerID, paidDate, amountPaid, cleared)

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

Assume Referential Integrity as follows:
- Each productID in Sales appears as a productID in Products.
- Each customerID in Sales appears as a customerID in Customers.
- Each storeID in Sales appears as a storeID in Stores.
- Each dayDate in Sales appears as a dayDate in Days.
- Each customerID in Payments appears as a customerID in Customers.

Write legal SQL queries for Questions 25-28. 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 do. Some points will also be deducted for unnecessarily complex queries, including use of GROUP BY when it's not needed.