Question 1. [6 MARKS]

Consider the following schema. SIN stands for Social Insurance Number.

Relations

- Person(SIN, name)
- Doctor(SIN, speciality)
- Caresfor(doctor, patient)

Integrity constraints

- Doctor[SIN] $\subseteq$ Person[SIN]
- Caresfor[doctor] $\subseteq$ Doctor[SIN]
- Caresfor[patient] $\subseteq$ Person[SIN]
- $(\sigma_{doctor=patient}$ Caresfor$) = \emptyset$

Part (a) [3 MARKS]

Suppose relation Caresfor has 4 tuples. What can you say about the minimum number of tuples in relation Person?

Solution:

There must be 3 tuples in Person. We can make a valid instance with that few because the schema does not prevent two people from caring for each other. Also, we can chain together CaresFor relationships, for example A cares for B and B cares for C.

Give a valid instance of the database that demonstrates your answer.

Person:

SIN	name

Doctor:

SIN	specialty

Caresfor:

doctor	patient

Solution:

Person:

SIN	name
A	Barney
B	Wilma
C	BamBam

Doctor:

SIN	specialty
A	pediatrics
B	pediatrics
C	pediatrics

Caresfor:

doctor	patient
A	B
B	A
B	C
C	B

Part (b) [3 MARKS]

Use the notation $R = \emptyset$ to express the following new constraint: A doctor's patient can't be his or her doctor.

Solution:

$$[\sigma_{c1.doctor=c2.patient \wedge c2.patient=c1.doctor}(\rho_{c1}CaresFor \times \rho_{c2}CaresFor)] = \emptyset$$

Question 2. [14 MARKS]

Here is part of the schema from assignment 1:

Relations

- Object(<u>CN</u>, date, name, description, type, length, width, height, who)
- Donor(<u>DID</u>, surname, firstname, address, email)
- Donation(<u>NID</u>, date, DID)
- Contains(<u>NID, CN</u>)
- Staff(<u>SID</u>, surname, firstname, address, email, type, date)

Integrity constraints

- Object[who] $\subseteq$ Staff[SID]
- Contains[NID] $\subseteq$ Donation[NID]
- Contains[CN] $\subseteq$ Object[CN]
- Donation[DID] $\subseteq$ Donor[DID]

Answer the following questions in relational algebra, using only the basic operators $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$.

Part (a) [7 MARKS]

For each staff member who has catalogued at least one object, find the first object that he or she catalogued. If there are ties, report them all. Report the SID and CN, as well as the DID of the donor who donated the item.

Solution:

$$Catalogued(SID, CN, date) := \Pi_{who,CN,date} Object$$
$$NotFirst(SID, CN, date) := \Pi_{c1.SID,c1.CN,c1.date} \sigma_{c1.SID=c2.SID \wedge c1.date>c2.date} (\rho_{c1} Catalogued \times \rho_{c2} Catalogued)$$
$$First(SID, CN, date) := Catalogued - NotFirst$$
$$Answer(SID, CN, DID) := \Pi_{SID,CN,DID} (\sigma_{Contains.NID=Donation.NID} (First \bowtie Contains) \times Donation)$$

Part (b) [7 MARKS]

Find donations that meet the following conditions: they contain no items whose type is "couch", and no items that were catalogued by a temp (i.e., a staff member whose type is "temp"). Report the donation's NID and the donor's email address.

Solution:

$HasCouch(NID) := \sigma_{type="couch"}(Contains \bowtie Object)$

$HasTempCataloguedItem(NID) :=$

$\quad \Pi_{NID}\sigma_{who=SID \wedge Staff.type="temp"}((Contains \bowtie Object) \times Staff)$

$BareAnswer(NID) := \Pi_{NID}((Donation - HasCouch) - HasTempCataloguedItem)$

$Answer(NID, email) := \Pi_{NID,email}(BareAnswer \bowtie Donation \bowtie Donor)$

Question 3. [11 MARKS]

Here is part of a schema you used for one of your Lecture Prep exercises.

Relations

- Employee(eid, name, salary, dept)
- Department(did, name, division)
- Sales(eid, day, amount)

Integrity constraints

- Employee[dept] ⊆ Department[did]
- Sales[eid] ⊆ Employee[eid]

Part (a) [2 MARKS]

Suppose table Sales has this content:

eid	day	amount
4	2013-11-02	9
4	2013-11-03	10
4	2013-11-05	25
4	2013-11-06	129
5	2013-11-01	12
5	2013-11-02	9
6	2013-11-06	129
6	2013-11-07	18
7	2013-11-01	18
7	2013-11-02	8
8	2013-11-01	28
8	2013-11-02	129

Below, show the output of the following query.

```
SELECT eid
FROM Sales s1
WHERE EXISTS
  (SELECT eid
  FROM Sales s2
  WHERE s1.eid <> s2.eid AND s1.day = s2.day AND s1.amount = s2.amount);
```

Solution:

eid
4
4
5
6

(4 rows)

Part (b) [2 MARKS]

Complete each of the following two queries so that they will run without error:

```
SELECT eid, min(amount), count(day)
FROM Sales
```

```
SELECT count(eid), day, sum(amount)
FROM Sales
```

Solution:

```
SELECT eid, min(amount), count(day)
FROM Sales
GROUP BY eid;
```

groups / sorts by eid

```
 eid | min | count
-----+-----+-------
   8 |  28 |     2
   4 |   9 |     4
   5 |   9 |     2
   6 |  18 |     2
   7 |   8 |     2
(5 rows)
```

```
SELECT count(eid), day, sum(amount)
FROM Sales
GROUP BY day;
```

groups / sorts by day

```
 count |    day     | sum
-------+------------+-----
     4 | 2013-11-02 | 155
     1 | 2013-11-07 |  18
     1 | 2013-11-03 |  10
     3 | 2013-11-01 |  58
     1 | 2013-11-05 |  25
     2 | 2013-11-06 | 258
(6 rows)
```

Part (c) [2 MARKS]

Complete this query so that it will report the eid of employees whose total sales in the Sales table exceeds 1,000. Do not use subqueries.

```
SELECT eid
FROM Sales
```

Solution:

```
SELECT eid
FROM Sales
GROUP BY eid
HAVING sum(amount) > 1000;
```

selects from sales table

groups / sorts by eid

gets sum > 1000

Part (d) [5 MARKS]

Write a query in SQL to find the eid of employees whose department name is "Widgets" and who have never made a sale.

Solution:

```
(SELECT eid FROM Employee, Department WHERE did=dept AND department.name='Widgets')
except
(SELECT eid FROM Sales);
```