Question 1. [7 MARKS]

Suppose we have these relations: M(a, b), N(c, d, e), and O(m, g) and these constraints:

1. N[c] ⊆ M[a]
2. N[e] ⊆ O[g]

Part (a) [3 MARKS]

Draw a valid instance of this schema, with three tuples in each relation. Draw the relations in table form and include column headings.

Solution:

M

a	b
1	2
3	4
5	6

N

c	d	e
5	8	17
3	11	17
3	14	17

O

m	g
16	17
18	19
20	21

Part (b) [1 MARK]

Name one attribute (from any relation) that is a foreign key:

Solution:
The *only* attribute in the schema that is a foreign key is c.

Part (c) [3 MARKS]

Express the following constraint using the $R = \emptyset$ notation we used on the assignment: No value can occur as a `b` in relation M and as an `e` in relation N and as a `g` in relation O. In other words, no value can be in all three columns.

Solution:
$(\sigma_{b=e=g} M \times N \times O) = \emptyset$

In fact, it is also correct to express this constraint with a simpler condition: $(\sigma_{b=e=g} M \times N \times O) = \emptyset$ Knowing that $b = e$ is sufficient to guarantee a violation because the integrity constraint N[e] ⊆ O[g] guarantees that every value for attribute `e` in N occurs as a value for attribute `g` in O.

Question 2. [6 MARKS]

We used the following schema many times in lecture:

Relations

- Students(SID, surName, campus)
- Courses(CID, cName, WR)
- Offerings(OID, CID, term, instructor)
- Took(SID, OID, grade)

Integrity constraints

- Offerings[CID] $\subseteq$ Courses[CID]
- Took[SID] $\subseteq$ Students[SID]
- Took[OID] $\subseteq$ Offerings[OID]

Part (a) [6 MARKS]

Write a query in relational algebra to report the instructors who have taught all the writing requirement courses that Pitt has taught.

Use only the basic operators we used in class and on the assignment: $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$.

Solution:

– This course is a WR course taught by Pitt.

$PWR(CID) := \Pi_{CID}\sigma_{WR \wedge instructor = \text{“}Pitt\text{”}}(Courses \bowtie Offerings)$

– This instructor should have taught this course (in any offering) if they are to be in the final result.

$ShouldHaveTaught(instructor, CID) := (\Pi_{instructor}\, Offerings) \times (PWR)$

– This instructor did teach this course.

$DidTeach(instructor, CID) := \Pi_{instructor,CID}\, Offerings$

– This instructor failed to teach this course that is one of the WR courses taught by Pitt.

$MissedPWR(instructor, CID) := ShouldHaveTaught - DidTeach$

– This instructor did not miss any WR course taught by Pitt. In other words, he or she taught every one of them.

$Answer(instructor) := (\Pi_{instructor}\, Offerings) - (\Pi_{instructor}\, MissedPWR)$

Question 3. [4 MARKS]

This question uses the Offerings table from the courses schema of question 2.

Suppose we were interested in who were the first people to teach courses. The following query is intended to find the instructor or instructors who have the lowest value for term. (There are very likely ties.) The query is syntactically well-formed, but does not produce the correct answer in all cases.

Note: The query was supposed to simply "find the instructor or instructors who have the lowest value for term". Some students thought that the query should find the lowest term *for each course*. Those who answered correctly under this assumption got full credit.

$$NotMin(instructor) := \Pi_{O1.instructor}\sigma_{O1.term>O2.term}(\rho_{O1}Offerings \times \rho_{O2}Offerings)$$
$$Answer(instructor) := (\Pi_{instructor}Score) - NotMin$$

Below you will show that the query is incorrect by giving a valid instance of the table Offerings, with at most five tuples, on which the query would give the wrong result.

1. Valid instance of the table Offerings (with at most 5 tuples):

   Solution:

OID	CID	term	instructor
O1	csc111	2	A
O2	csc222	3	A
O3	csc333	8	B
O4	csc444	2	C

2. The correct answer for this instance:

   Solution:

instructor
A
C

3. The result that the query given above would produce for this instance:

   Solution:

instructor
C

Question 4. [4 MARKS]

Suppose the tables One and Two have the following contents:

One

x	y
2	5
1	6
1	6
1	6
4	8
4	8

Two

x	y
2	55
1	1
4	4
1	6
4	8
4	8
9	9
1	6

For each query below, draw the table that results. If the query would give an error, write "Error" instead, and explain what the error is.

```
(SELECT * FROM One)
    EXCEPT
(SELECT * FROM Two);
```

Solution:

```
 x | y
---+---
 2 | 5
(1 row)
```

```
(SELECT * FROM One)
    EXCEPT ALL
(SELECT * FROM Two);
```

Solution:

```
 x | y
---+---
 1 | 6
 2 | 5
(2 rows)
```

```
(SELECT * FROM One)
    INTERSECT
(SELECT * FROM Two);
```

Solution:

```
 x | y
---+---
 4 | 8
 1 | 6
(2 rows)
```

```
(SELECT * FROM One)
    INTERSECT ALL
(SELECT * FROM Two);
```

Solution:

```
 x | y
---+---
 4 | 8
 4 | 8
 1 | 6
 1 | 6
(4 rows)
```

Question 5. [5 MARKS]

Here's a portion of the schema for the IMDb database we have used extensively in class:

```
CREATE TABLE people (
  person_id integer primary key,
  name varchar NOT NULL UNIQUE
) ;

CREATE TABLE movies (
  movie_id integer primary key,
  title varchar NOT NULL,
  year integer NOT NULL,
  rating float NOT NULL,
  UNIQUE (title,year)
) ;

CREATE TABLE roles (
  person_id integer NOT NULL references people,
  movie_id integer NOT NULL references movies,
  role varchar NOT NULL,
  PRIMARY KEY (person_id,movie_id,role)
) ;
```

Complete the WHERE clause in the following query so that you find all people who've had a role in a Star Wars movie. Use `LIKE '%Star Wars%'` to identify the Star Wars movies.

```
SELECT *
FROM people
WHERE .  .  .
```

Solution:

```
SELECT *
FROM people
WHERE person_id IN (
   SELECT person_id
   FROM roles JOIN movies ON roles.movie_id = movies.movie_id
   WHERE title LIKE '%War%'
);
```