2. (10 points) **SQL.** *Some Quidditch League!* [Exercise]

Consider the Movie database with the schema in Figure 2 on page 15 for the questions below.

---

a. [5pt] Write an SQL query for the following.

List each actor by name, gender, role, and the minutes they appear on screen in that role in a given movie by title, studio, and year such that "Hampton Fancher" was an author of the screenplay of the movie and the movie's genre is "SciFi".

```
select P.name, P.gender, C.role, C.minutes, M.title, M.studio, M.year
from Person P, Cast C, Movie M, Authored A, Person W
where C.actor = P.p#
  and C.title = M.title and C.studio = M.studio and C.year = M.year
  and M.genre = 'SciFi'
  and A.title = M.title and A.year = M.year
  and A.writer = W.p#
  and W.name = 'Hampton Fancher';
```

marking guide

+2 correct sources and schema
+2 $\bowtie$'s correct
+1 σ's correct
  * Note that 'Hampton Fancher' is a Person's *name*, *and* not the value of p# in Writer.

b. [5pt] For each movie, report how many male actors and how many female actresses—gender = 'M' for *male* and gender = 'F' for *female*—were *cast* in the movie in columns male and female, respectively.

Only count a given actor or actress *once* per movie; that is, that a person may have played several *characters* (*roles*) in the movie does *not* count multiple times.

For full credit, the column male or female should report '0' (zero) if there were no actors or actresses, respectively, in the movie.

```
with
    Counts (title, studio, year, male, female) as (
        select title, studio, year, 0, 0
            from Movie
        union
        select C.title, C.studio, C.year, count(distinct C.actor), 0
            from Cast C, Person P
            where C.actor = P.p#
              and P.gender = 'M'
        union
        select C.title, C.studio, C.year, 0, count(distinct C.actor)
            from Cast C, Person P
            where C.actor = P.p#
              and P.gender = 'F'
    )
select C.title, C.studio, C.year,
       max(C.male) as male,
       max(C.female) as female
    from Counts C
    group by C.title, C.studio, C.year;
```

marking guide

+2 counts per gender are done correctly
+1 accounts for the 0's
+1 right logic to get the correct numbers
+1 correct schema, ⋈'s, structure of query

REFERENCE *(Detach this page for convenience, if you want.)*

Schema for the Colours Database.

Customer	
cust#	PK
cname	
fav_colour	
phone#	

Item	
item#	PK
prod#	FK to **Product**
cust#	FK to **Customer**
colour	
date_sold	

Product	
prod#	PK
pname	
cost	
maker	FK to **Company**

Avail_Colours	
prod#	PK, FK to **Product**
colour	PK

Figure 1: Colours Schema.

Schema for the Movie Database.

Person(p#, *name*, birthdate, *nationality*, *gender*)
Actor(p#, *aguild#*)
  FK (p#) refs **Person**
Director(p#, *dguild#*)
  FK (p#) refs **Person**
Writer(p#, *wguild#*)
  FK (p#) refs **Person**
Studio(name)
ScreenPlay(title, year)
Authored(title, year, writer)
  FK (title, year) refs **ScreenPlay**
  FK (writer) refs **Writer** (p#)
Movie(title, studio, year, *genre*, *director*, *length*)
  FK (studio) refs **Studio** (name)
  FK (title, year) refs **ScreenPlay**
  FK (director) refs **Director** (p#)
Cast(title, studio, year, role, actor, *minutes*)
  FK (title, studio, year) refs **Movie**
  FK (actor) refs **Actor** (p#)
Affiliated(director, studio)
  FK (director) refs **Director** (p#)
  FK (studio) refs **Studio** (name)

Figure 2: Movie Schema.

```
create table A(a1 int not null primary key,
               a2 int);
create table B(b1 int not null primary key,
               b2 int references A
                   on delete cascade on update no action);
create table C(c1 int not null primary key,
               c2 int references B
                   on delete cascade on update no action);
insert into A values (1,1),(2,1),(3,1),(4,2),(5,2),(6,3);
insert into B values (1,1),(2,1),(3,1),(4,2),(5,2),(6,4);
insert into C values (1,1),(2,1),(3,2),(4,3),(5,4),(6,6);
```

For Questions 4i and 4j, assume the tables **A**, **B**, and **C** were created and populated with data as above, followed by the statement in the question.

Choose the answer that represents what is in table **C** afterwards.

---

i. `delete from A;`

A. `(1,1),(2,1),(3,2),(4,3),(5,4),(6,6)`
B. `(2,1),(3,2),(4,3),(5,4),(6,6)`
C. `(4,3),(5,4),(6,6)`
D. `(5,4),(6,6)`
E. *no tuples*

---

j. `delete from A where a1 = 1;`

A. `(1,1),(2,1),(3,2),(4,3),(5,4),(6,6)`
B. `(2,1),(3,2),(4,3),(5,4),(6,6)`
C. `(4,3),(5,4),(6,6)`
D. `(5,4),(6,6)`
E. *no tuples*

---

4. (10 points) **Normal Forms.** *This just isn't normal!* [Analysis]

Consider table **R** with attributes A, B, C, D, E, F, & G, and the set of functional dependencies

$$\begin{array}{ll} \mathsf{BG} \mapsto \mathsf{AC} & \mathsf{A} \mapsto \mathsf{G} \\ \mathsf{ABG} \mapsto \mathsf{DF} & \mathsf{D} \mapsto \mathsf{E} \\ & \mathsf{E} \mapsto \mathsf{C} \end{array}$$

a. (3 points) Is AD a candidate key? Prove your answer.

The attribute closure of AD is ACDEG, which does not include all attributes. Therefore AD is not a key.

b. (2 points) Is BDG a superkey? Prove your answer.

The attribute closure of BG consists of all the attributes; hence, BG is key and BDG is superkey.

c. (2 points) Does ADC $\mapsto$ F logically follow from the set of functional dependencies? Prove your answer.

The attribute closure of ADC is ADCGE. This does not contain F. Therefore the answer is no.

1. **Query Semantics.** *Say what?* (10 points) [Short Answer]

Customer	
cust#	PK
cname	
fav_colour	
phone#	

Item	
item#	PK
prod#	FK to **Product**
cust#	FK to **Customer**
colour	
date_sold	

Product	
prod#	PK
pname	
cost	
maker	FK to **Company**

Avail_Colours	
prod#	PK, FK to **Product**
colour	PK

Figure 1: **Colour** Schema.

Consider the schema in Figure 1 as we have used in class.

For each of the following, say briefly in *plain* English—as the descriptions for Question 2a–2d—what the query means, as though you are explaining it to a non-technical person.

Note that simply paraphrasing the query—e.g., "This query finds, for all cust#'s such that there exists a prod#, for all. . ."—is *not* adequate!

---

a. (2 points)

```
select C.cust#, C.cname, P.prod#, P.pname
    from Customer C, Item I, Product P
    where C.cust# = I.cust# and I.prod# = P.prod#
        and P.cost > 1000
        and I.date_sold >= '1 May 2008';
```

Report customers by name and items they have bought on and after 1 May 2008 by product name that cost more than one thousand.

b. (3 points)

```
select P.prod#, P.pname, count(*) as number, sum(cost) as amount
    from Item I, Product P
    where I.prod# = P.prod#
    group by P.prod#, P.pname;
```

List each distinct product by name that has ever been sold and report how many items of it have been sold and the total (cost) spent for it.

c. (2 points)

$\pi_{\mathsf{cname}}(\sigma_{\mathsf{colour}=\mathsf{fav\_colour}}(\textbf{Customer} \bowtie \textbf{Item}))$

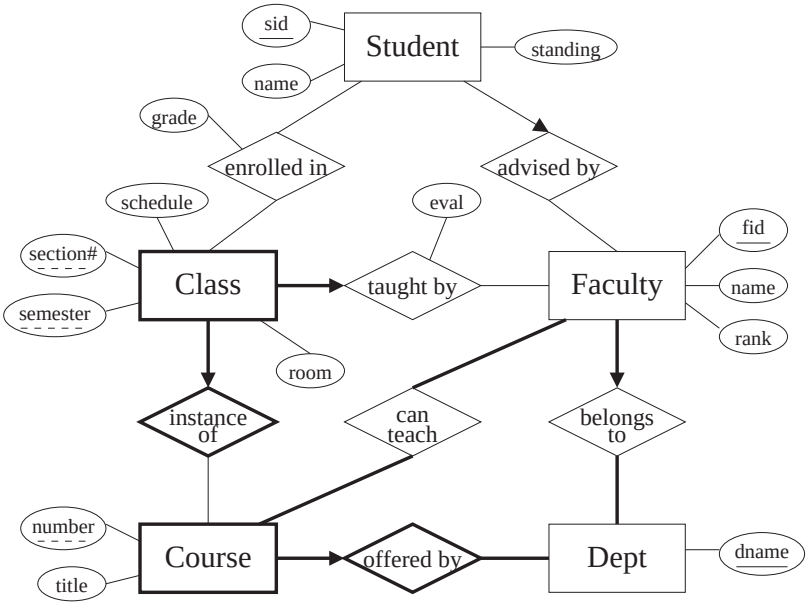
List customers by name who have ever bought an item in his or her favourite colour. (Note that cust# is not returned; this means if there are two customers of the same name each of whom qualifies, the name will be reported only once. This could lead to confusion.)

d. (3 points)

```
select C.cust#, C.cname, P.prod#, P.pname
    from Customer C, Product P
    where not exists (
            select A.colour
                from Avail_Colour A
                where P.prod# = A.prod#
            except
            select I.colour
                from Item I
                where C.cust# = I.cust# and P.prod# = I.prod#
        );
```

List customers by name with each product for which they own in all its available colours.

3. (10 points) **E-R to Schema.** *A Classy E-R.* (Exercise)



a. (7 points) Write a relational schema for the E-R diagram above. You may use the shorthand style used in exercises and in Question 1. Do not introduce tables unless they are completely necessary.

Dept(dname)
Faculty(fid, name, rank, *dname*)
FK (dname) refs **Dept**
Student(sid, name, standing, advisor)
FK (advisor) refs **Faculty** (fid)
Course(dname, number, title)
FK (dname) refs **Dept**
Class(dname, number, section#, semester, room, eval, *fid*)
FK (dname, number) refs **Course**
FK (fid) refs **Faculty**
can_teach(dname, number, fid)
FK (dname, number) refs **Course**
FK (fid) refs **Faculty**
enrolled_in(dname, number, section#, semester, sid, grade)
FK (dname, number, section#, semester) refs **Class**
FK (sid) refs **Student**

2. In each of the following, **R** stands for *read*, **W** for *write*, **C** for *commit*, and **A** for *abort*.

   a. Consider transactions $\mathbf{T}_1$ and $\mathbf{T}_2$, both of which read and write database values A and B. Which of the following schedules is serializable (that is, is equivalent to a serial schedule)?

$\mathbf{T}_1$	**R**(A)			**W**(A)			**R**(B)			**W**(B)		**C**
$\mathbf{T}_2$		**R**(A)			**W**(A)			**R**(B)			**W**(B)	**C**
$\mathbf{T}_1$	**R**(A)	**W**(A)	**R**(B)	**W**(B)								**A**
$\mathbf{T}_2$							**R**(A)	**W**(A)	**R**(B)	**W**(B)		**C**
$\mathbf{T}_1$	**R**(A)	**W**(A)					**R**(B)	**W**(B)				**C**
$\mathbf{T}_2$				**R**(A)	**W**(A)				**R**(B)	**W**(B)		**C**
$\mathbf{T}_1$	**R**(A)	**W**(A)	**R**(B)								**W**(B)	**C**
$\mathbf{T}_2$					**R**(A)	**W**(A)	**R**(B)	**W**(B)				**C**
$\mathbf{T}_1$	**R**(A)	**W**(A)							**R**(B)	**W**(B)	**C**	
$\mathbf{T}_2$			**R**(A)	**W**(A)	**R**(B)	**W**(B)						**C**

---

d. (3 points) Consider the following schedule.

T_1	R(X)	W(X)	C
T_2		W(X)	C

Which of the following is true about this schedule?

A. It is serializable, but it is not conflict serializable.
B. It is avoids cascading aborts, but it is not recoverable.
C. It is recoverable, but it is not serializable.
D. It is not recoverable, and it is not serializable.
E. It is a strict schedule.

---

e. (3 points) Each transaction should be (logically) processed as though it were independent of all other transactions. This is which ACID property?

A. Atomicity
B. Consistency
C. Isolation
D. Durability
