In the SQL questions, you are welcome to use views. Comments are not required, although they may help us mark your answers.

The last page of this exam is a reference page, containing schemas and a dataset.

There is a blank page at the end for rough work. If you want any of it marked, indicate that clearly there, as well as in the question itself.

A mark of at least 40 out of 100 on this exam is required in order to pass the course.
Question 1. [6 marks]
Tear off the last page of the exam, which contains a reference sheet.

Consider the Company schema given on the reference sheet. Answer the following questions about this schema, assuming the pure relational model: no duplicate tuples or NULLs allowed.

Part (a) [2 marks]
According to this schema, can one employee belong to more than one department?

Circle one: Yes No

Explain your answer in the space below.

Part (b) [2 marks]
Write an integrity constraint to enforce the following condition: for all pairs of employees A and B, if employee A manages employee B, then employee B cannot also manage employee A.

Part (c) [2 marks]
What would be the real-world implication of making \{junior\} a key of the Manages relation instead of \{manager, junior\}?
Question 2. [12 marks]
Write relational algebra expressions to make the following queries, using only the basic operators we used in class: Π, σ, △◁, ×, ∩, ∪, −, ρ, and assignment.

Part (a) [5 marks]
For each department, report the highest salary of any employee in that department. You may assume every department has at least one employee.

Your output relation should have two attributes called dID and highest. You should have one tuple per department.
Part (b) [7 marks]

We say that employee $a$ dominates employee $b$ if they are different employees and for every day in which $b$ had a sales amount, $a$ had a sales amount that was higher than $b$’s sale amount. An employee who did not sell on any day is dominated by every other employee.

Report all pairs of employees where one dominated the other. Your output relation should have two attributes called $eID_1$ and $eID_2$, where $eID_1$ is the ID of the employee who dominates, and $eID_2$ is the ID of the employee who is dominated.

**Hint:** A good starting point is constructing a relation with all pairs of eIDs that are not the same, and another relation DominateDays($eID_1$, $eID_2$, day) where $eID_1$ sold a higher amount than $eID_2$ on that day.
Now answer some questions on the same schema, but in SQL.

**Question 3. [6 marks]**

Write a query in SQL that does the following. For each manager, find their junior with the greatest total sales. If there is a tie, report them all. Produce output with the form below; `manager` is the ID of the manager, and `besttotal` is the greatest total sales.

```
manager | besttotal
---------+-----------
          +----------
```
Question 4. [13 marks]

Important: You may use the view defined in any subquestion (even if you didn’t answer it) when solving other subquestions. It may help to define additional views.

Part (a) [5 marks]
For each department, find the number of employees with an unknown salary. Include departments in which no one’s salary is unknown. (For them, the number of unknown salaries is, of course, zero.) Do not include departments which have no employees. Create a view for this query called DeptUnknowns, with the form below.

Hint: You can use a constant, such as 0, in place of a column name in a SELECT statement. For example, this is legal: SELECT 1234 AS amount ... FROM Sales ...

<table>
<thead>
<tr>
<th>did</th>
<th>unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part (b)  [4 MARKS]
For each department, find the employee with the minimum salary (the “poorest”). Report the department ID, the employee ID of the poorest employee from that department, and their salary. Note that an employee’s salary may be unknown (that is, it may be recorded as NULL). If everyone in a department has a NULL salary, or if a department has no employees, that department will not appear in your result.

Create a view for this query called *DeptPoorest*, with the form below.

<table>
<thead>
<tr>
<th>did</th>
<th>poorest</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part (c)  [4 MARKS]
Write a query to report, for each department in the *Departments* table, the department ID, the ID and salary of the employee with the lowest salary, and the number of employees with an unknown salary. Don’t leave any departments out! Your result must have the form below.

<table>
<thead>
<tr>
<th>department</th>
<th>poorest</th>
<th>salary</th>
<th>unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>did</th>
<th>poorest</th>
<th>salary</th>
<th>unknowns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 5. [8 marks]
Suppose I have a file called `nonsense.ddl` containing this:

```
DROP SCHEMA IF EXISTS rp CASCADE;
CREATE SCHEMA rp;
SET SEARCH_PATH TO rp;

CREATE TABLE Things (  
    A INT PRIMARY KEY,
    B INT,
    C INT UNIQUE
);

CREATE TABLE Junk (  
    G INT PRIMARY KEY,
    H INT,
    I INT,
    FOREIGN KEY (I) REFERENCES Things(A) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE Stuff (  
    D INT,
    E INT,
    F INT PRIMARY KEY,
    FOREIGN KEY (E) REFERENCES Things(C) ON UPDATE RESTRICT ON DELETE SET NULL,
    FOREIGN KEY (E) REFERENCES Junk(G) ON UPDATE SET NULL ON DELETE CASCADE
);
```

Part (a) [2 marks]
Suppose I imported this file into PostgreSQL using the command `\i nonsense.ddl` and then a few weeks later the following happened when I tried to access table `Junk`.

```
dbsrv1% psql csc343h-dianeh
psql (9.1.15, server 9.1.14)
Type "help" for help.

csc343h-dianeh=> SELECT * FROM Junk;
ERROR: relation "junk" does not exist
LINE 1: SELECT * FROM Junk;
```

Modify my interaction above so that the `SELECT` statement works.
Part (b) [2 marks]
What is the most important thing that is the same about PRIMARY KEY and UNIQUE?

What is one important difference between PRIMARY KEY and UNIQUE?

Part (c) [2 marks]
Suppose the tables have been populated as shown below. Modify the data to show the contents of the three tables after this command is executed:

UPDATE Things SET C = 20 WHERE A = 8;

<table>
<thead>
<tr>
<th>Things:</th>
<th>Stuff:</th>
<th>Junk:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>

Part (d) [2 marks]
Suppose we began with the same original tables, shown below, but ran a different command. Modify the data to show the contents of the three tables after this command is executed:

DELETE FROM Things WHERE C = 3;

<table>
<thead>
<tr>
<th>Things:</th>
<th>Stuff:</th>
<th>Junk:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9</td>
</tr>
</tbody>
</table>
Quiz DTD
This is a DTD representing questions on a quiz, and student responses for that quiz.

```xml
<!ELEMENT quiz (questions, class-responses)>  
<!ELEMENT questions ((mc-question | tf-question)+)>  
<!ELEMENT mc-question (question, option+)>  
<!ATTLIST mc-question qid ID #REQUIRED>  
<!ATTLIST mc-question solution CDATA #REQUIRED>  
<!ELEMENT question (#PCDATA)>  
<!ELEMENT option (#PCDATA)>  
<!ELEMENT tf-question (question)>  
<!ATTLIST tf-question qid ID #REQUIRED>  
<!ATTLIST tf-question solution (True|False) #REQUIRED>  
<!ELEMENT class-responses (student+)>  
<!ELEMENT student (response+)>  
<!ATTLIST student sid ID #REQUIRED>  
<!ELEMENT response EMPTY>  
<!ATTLIST response qid IDREF #REQUIRED>  
<!ATTLIST response answer CDATA #REQUIRED>
```
Question 6.  [6 marks]
Answer the following questions about the quiz DTD on the previous page.

Part (a)  [2 marks]
Does this DTD enforce the constraint that every quiz contains at least one multiple-choice question and at least one true-false question?

Circle one:  Yes  No

Explain your answer.

Part (b)  [2 marks]
What restrictions, if any, does the DTD enforce about the number of responses each student could have for the quiz?

Part (c)  [2 marks]
Three attributes are listed in this DTD with ID data.
What is the purpose of using ID?

What makes IDs less powerful than keys in SQL?
Question 7. [15 marks]

Here is a standalone XML file called data.xml. It also appears on the reference page at the end of this exam.

```xml
<?xml version="1.0" standalone="yes"?>
<foo a="1">
    <bar b="hi" c="5">
        <baz a="15" />
        <qux>Data</qux>
    </bar>
    <bar b="good" c="4">
    </bar>
    <bar b="bye">
        <baz a="19" />
        <baz a="10" />
        <baz a="0" />
    </bar>
    <bar b="greetings" c="1">
        <qux>Dream</qux>
        <qux>Electric</qux>
        <baz a="16" />
        <qux>Sheep</qux>
    </bar>
</foo>
```

Each of the queries on the next two pages runs without errors. Show the output. We will not be grading the whitespace in your answer, but please use newlines and indentation appropriately to make the output easier to read.

Note: the queries are NOT meant to “make sense.” If you think a query outputs nothing, say “outputs nothing”. If you think it outputs an empty sequence, simply write (). At least one of the queries doesn’t successfully do what it is trying to do.
Part (a)  [3 MARKS]
let $d := doc("data.xml")
return ($d/foo/bar[@b="bye"], $d//qux[2])

Part (b)  [3 MARKS]
<docs>{
    for $bar in doc("data.xml")/*/*
        where $bar/baz[@a < 16]
        return $bar/qux
    }
}</docs>

Part (c)  [3 MARKS]
for $e in doc("data.xml")/*/*
return
    <elem>data($e/@b)</elem>
Part (d)  [6 MARKS]

let $z :=$
\[
<r>
\{
  for $q$ in doc("data.xml")//qux
  return
    if ($q$/parent::bar/@c < 3)
      then <answer y="{data($q)}" />
    else <answer y="none" />
\}
\</r>
\let $f := doc("data.xml")//baz[@a >= 16]
\for $n$ in $z//@y$
\for $m$ in $f//@a$
\return <x>{($n,$m)}</x>
Question 8. [6 marks]
Consider relation $R(A, B, C, D, E, F)$ with functional dependencies $S$.

$$S = \{ CD \rightarrow A, \quad B \rightarrow EF, \quad A \rightarrow BC, \quad F \rightarrow D \}$$

Part (a) [1 mark]
Above, circle all FDs that violate BCNF.

Part (b) [2 marks]
Add a second row to this instance of $R$ that satisfies all its functional dependencies and has redundant data. Circle every value that could be erased and yet its value could be inferred.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Part (c) [2 marks]
Show below what attributes would go into the two relations resulting from the first step of BCNF decomposition on $R$. (There may be several correct answers.) Do not project the functional dependencies onto the relations, just show the relations.

Rough work:
One new relation would have attributes:

The other new relation would have attributes:

Part (d) [1 mark]
What important property does BCNF decomposition not guarantee?

Question 9. [4 marks]
Suppose we are employing the 3NF synthesis algorithm on a relation $R(A, B, C, D, E, F)$, and we already have the following minimal basis:

$$S = \{ A \rightarrow C, \quad BF \rightarrow D, \quad DF \rightarrow C, \quad EF \rightarrow D \}$$

I therefore propose that the following schema is the result of the 3NF synthesis algorithm: $R1(A, C)$, $R2(B, F, D)$, $R3(D, F, C)$, and $R4(E, F, D)$. But I am wrong. Why am I wrong? Justify all points in your answer.

Correct my proposed schema, according to the 3NF synthesis algorithm. Explain all steps in your answer.
Question 10. [6 marks]
Consider relation $R(A, B, C, D, E, F)$ with functional dependencies $S$.

$$S = \{ AC \rightarrow B, \ B \rightarrow ACE, \ BC \rightarrow E, \ BD \rightarrow AF, \}$$

Compute a minimal basis for $S$. Show your rough work, and put your final answer where shown on the next page. There will be no marks for a correct answer without the rough work.

Rough work:
A minimal basis is:
Question 11. [6 marks]

Suppose we are employing the 3NF synthesis algorithm on a relation $R(A, B, C, D, E)$, and we already have the following minimal basis:

$$S = \{A \rightarrow DE \quad C \rightarrow A, \quad E \rightarrow A\}$$

Part (a) [3 marks]
List all the keys for relation $R$:

How do you know that nothing else is a key?

Part (b) [2 marks]
Show the final schema produced by the 3NF algorithm. Explain your answer in terms of the steps of the algorithm. Do not project the functional dependencies onto the relations, just show the relations.

Part (c) [1 mark]
What important property does 3NF synthesis not guarantee?
Question 12. [6 marks]

Part (a) [2 marks]
Consider the table Parent(pID, cID, ageAtBirth). A tuple in this relation records the fact that the person pID is a parent of the person cID, and the parent’s age on the day their child was born is ageAtBirth years.

For each query below, circle Yes or No to indicate whether it can be expressed in the relational algebra we have been using in this course, i.e., with operators Π, σ, ⊘◁, ×, ∩, ∪, -, ρ, and assignment.

1. Find the pID of the parent with the second-highest value for ageAtBirth.
   Yes  No

2. All pairs x, y where y is a descendent of x (any number of generations away).
   Yes  No

3. Find the ID of the person with the most children. There may be ties; all such parents should be included.
   Yes  No

4. Find the IDs of people with fewer than 6 children.
   Yes  No

Part (b) [2 marks]
An error occurs if we attempt to create the following two tables:

```
CREATE TABLE Person (  CREATE TABLE Contributions (  
   person_id PRIMARY KEY INT,  who TEXT,  
   name TEXT   organization TEXT,  
 );  hours INT,  FOREIGN KEY (who) REFERENCES Person(name)  );
```

All keywords and the statement syntax are correct, but we are asking to do something improper. Explain.

Part (c) [2 marks]
Can a relation with exactly three attributes violate BCNF? Circle one:  Yes  No

If yes, give an example (a relation schema and its FDs) that violates BCNF. If no, explain why not.
Question 13. [4 marks]

Below is an Entity-Relationship diagram about car dealerships. It may or may not represent the domain well. Answer the questions below.

1. A car sale cannot involve more than one salesperson.
   True    False

2. There can be two cars with the same VIN as long as the model and year are different.
   True    False

3. A salesperson can work at any number of dealerships.
   True    False

4. There can’t be more salespeople than dealerships.
   True    False

5. There can be multiple sales on the same date.
   True    False

6. Two salespeople can have the same sID as long as they work at different dealerships.
   True    False

7. This model contains a weak entity set.
   True    False

8. The works at relationship is a one-to-many relationship.
   True    False
Question 14. [6 marks]
Translate this Entity-Relationship diagram into a relational schema. For each relation, provide its name, attributes and keys. To indicate a key, underline all attributes that are part of the key using a single line. Also include all referential integrity constraints, using relational notation, not SQL notation.
[Use the space below for rough work. This page will not be marked, unless you clearly indicate the part of your work that you want us to mark.]
Company Schema

This is a schema representing the organization of employees at a company.

Relations

- **Department** (dID, name, division)
  Each tuple in this relation represents information about a different department in the company: its name, and which division it belongs to.

- **Employee** (eID, name, salary, dept)
  Each tuple in this relation represents information about a different employee in the company: their name, salary (amount of money earned per year), and which department the employee belongs to.

- **Manages** (manager, junior)
  Each tuple in this relation represents a relationship between two employee: manager is an employee who manages junior.

- **Sales** (eID, day, amount)
  Each tuple in this relation represents the total amount of sales made by an employee on a particular day.

Integrity constraints

- Employee[dept] ⊆ Department[dID]
- Manages[manager] ⊆ Employee[eID]
- Manages[junior] ⊆ Employee[eID]
- Sales[eID] ⊆ Employee[eID]
- \( \sigma_{amount<0} Sales = \emptyset \)
- \( \sigma_{salary<0} Employee = \emptyset \)
Company schema in DDL

```sql
CREATE TABLE Department(
    dID INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    division INTEGER);

CREATE TABLE Employee(
    eID INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    salary FLOAT,
    dept INTEGER REFERENCES Department(dID),
    CHECK(salary >= 0));

CREATE TABLE Manages(
    manager INTEGER REFERENCES Employee(eID),
    junior INTEGER REFERENCES Employee(eID),
    PRIMARY KEY(manager, junior));

CREATE TABLE Sales(
    eID INTEGER REFERENCES Employee(eID),
    day DATE,
    amount INTEGER,
    PRIMARY KEY(eID, day),
    CHECK(amount >= 0));
```

An XML dataset: data.xml

```xml
<?xml version="1.0" standalone="yes"?>
<foo a="1">
    <bar b="hi" c="5">
        <baz a="15" />
        <qux>Data</qux>
    </bar>
    <bar b="good" c="4">
    </bar>
    <bar b="bye">
        <baz a="19" />
        <baz a="10" />
        <baz a="0" />
    </bar>
    <bar b="greetings" c="1">
        <qux>Dream</qux>
        <qux>Electric</qux>
        <baz a="16" />
        <qux>Sheep</qux>
    </bar>
</foo>
```