Q1: Recommending Collaborators [20 points]
Consider ResearchBook, an online social network for researchers, which also helps one find possible collaborators. We are given a simple table with schema:

\[
\text{ResInt \ (pid, topic)} - \text{Person with pid has topic as a research interest}
\]

Clearly a person can have multiple research interests (like ‘Databases’ and ‘Artificial Intelligence’)---so there may be multiple tuples for each pid in the relation. The goal of this question is to write a SQL query that gives us pairs of researchers with exactly the same research interests (as we can recommend them as collaborators). This query is tricky---so let’s try to solve it by the following steps.

Q1.1. (10 points) Write a SQL query to find all person-person-interest triplets \((\text{pid1}, \text{pid2}, \text{topic})\) such that person \text{pid1} has \text{topic} as a research interest, but person \text{pid2} \text{doesn’t}. Call this the PPT view.

Solution:
create view PPT as
select A.pid as pid1, B.pid as pid2, A.topic
from ResInt as A, ResInt as B
where A.pid <> B.pid and A.topic not in (select topic from ResInt where pid = B.pid);

Q1.2. (10 points) Given the earlier view PPT and the original table ResInt, write the full SQL query to find all pairs of people with the exact same set of research interests. Remove mirror pairs and self-pairs.

Solution:
select C.pid as pid1, D.pid as pid2
from ResInt as C, ResInt as D
where C.topic = D.topic and C.pid < D.pid except
select pid1, pid2
from PPT
except
select pid2, pid1
from PPT;

Q2. Where Art thou? [15 points]
Consider the situation in Exercise 2.8 in your textbook. We repeat it here for your convenience. Although you always wanted to be an artist, you ended up being an expert on databases because you love to cook data and you somehow confused “database” with “data baste”. Your old love is still there, however, so you set up a database company, ArtBase that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still-lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person’s unique name, address, total amount of dollars spent in gallery (very important!), and the artists and groups of art that the customer tends to like.

The ER model your DB engineer designed was lost, and he has resigned. So you need to step in:

Q2.1. (10 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.
Solution:

create table artists (  
   name character varying(40) primary key,  
   birthplace character varying(60), age int,  
   style character varying (40) );

create table customers (  
   name
);
name character varying(40) primary key,
address character varying(100), spent numeric);

create table likeartist(
aname character varying(40) references artists (name),
cname character varying(40) references customers (name),
primary key (aname, cname));

create table artGroup( name character varying(40) primary key );

create table likegroup(
cname character varying(40) references customers(name),
gname character varying(40) references artGroup (name),
primary key (gname, cname));

create table artofwork(
title character varying(40) primary key,
year integer, type character varying(20),
price integer, artist character varying(40) references artists (name));

create table belongto(
title character varying(40) references artofwork (title),
gname character varying(40) references artGroup (name),
primary key (title, gname));
Q3. Hospital DB [20 points]
We want to design a database schema for a hospital. The patient is admitted to a hospital with a medical condition. The hospital maintains patients’ information. Name, age, sex, DOB, and address are recorded. Hospital identifies each patient by a unique id and creates a patient admission record. Each admission record has an admission number, admission date, and discharge date information. Hospital assigns a doctor to treat a patient. Hospital stores doctor’s name, id, specialty and year of experience. Each doctor has multiple patients. A patient is admitted to a ward. Ward is identified by ward number, name, and type (e.g., medical/surgical). Each ward contains multiple beds. Bed has number and type (e.g., side room bed/ open ward bed) as well. If patient needs a surgery than hospital schedule an operation. Operation number, date, time, patient id and doctors’ id are required to schedule an operation. We assume multiple doctors participated in an operation. Draw an ER diagram based on above scenario. List all of your assumptions you make for your drawing.

Solution:

Q4. Real-estate Database [25 points]
We are asked to design a database management system for all information related to a real-estate company which has several branches throughout the United States. The first
step is to organize the information given about company. We have collected the following data:

- Each branch has a unique branch number. It allocated staff, which includes one Manager. Also, each branch has a list of available houses to rent/sell.
- The manager is responsible for the day-to-day running of a given branch.
- Each staff has a unique staff number, name, position, salary, and branch number.
- Each house has a unique house number, address, rent cost, sell value, status, branch number.
- The status of a house indicates whether it is available for rent/sell. The branch number indicates which branch of the real-estate company can rent/sell the house.
- A Customer has SSN, name, contract number, house number.
- Each Contract has a unique contract number, type, date, branch number, and customer number.
- If the type of a contract is rent, it will have a deposit, and rent cost, and lengths of contract. The length of a rent contract can be six months, one year or 2 years but customers can extend their contract by signing a new contract.
- Each contract extension has a unique extension number, the original contract number, and discount value.
- If the contract type is "sell", it will have sell value.
- Customers can issue a contract termination request. Each termination request has a unique request number, customer number, and termination cost.

Please answer the following questions:

Q4.1. (15 points) Draw an ER diagram for this database. Make sure to indicate primary keys, cardinality constraints, weak entities (if any), and participation constraints. List any assumptions you make in the process.

*Hint:* You may need an ISA hierarchy somewhere.

*Answer:*
Q4.2. (10 points) Translate the ER diagram in Q4.1 into relational database tables (i.e. give the SQL DDL statements). Make sure that the translation captures key constraints (primary keys and foreign keys if applicable) and participation constraints in the ER diagram. Identify constraints, if any, that you are not able to capture.

**Answer:**

create table Branch(Branch_no integer primary key);

create table Staff(
    Staff_no int primary key,
    Name char(100),
    Position char(100),
    Salary float);

create table House(
    House_no int primary key,
    Address char(100),
...
Rent_cost float,  
Sell_value float,  
Status char(30)  
check Status in (‘available’, ‘unavailable’));

create table Customer(
SSN int primary key,  
Name char(100));

create table Contract(
Contract_no int primary key,  
Date date);

create table Sell(
Contract_no primary key references Contract(Contract_no),  
Sell value float);  

create table Rent(
Contract_no primary key references Contract(Contract_no),  
Deposit float,  
Rent_cost float,  
Length char(30)  
check Length in (‘6 months’, ‘1 year’, ‘2 years’));

create table Manage(
foreign key Branch_no references Branch,  
foreign key Staff_no references Staff,  
Primary key(Branch_no, Staff_no));

create table Work(
foreign key Branch_no references Branch,  
foreign key Staff_no references Staff,  
Primary key(Branch_no, Staff_no));

create table sign(
foreign key Branch_no references Branch,  
foreign key House_no references House,  
foreign key Contract_no references Contract,  
foreign key SSN references Customer,  
Primary key(Branch_no, House_no, Contract_no, SSN));

create table Termination(
Req_no int primary key,  
Cost float,  
foreign key Contract_no references Contract,  
foreign key SSN references Customer);

create table Extension(
Ext_no int primary key,
Discount float,
foreign key SSN references Customer,
foreign key Contract_no references Contract);

create table control(
foreign key Branch_no references Branch,
foreign key House_no references House,
Primary key(Branch_no, House_no));

Q5. Subject Code Formatting [20 points]
Suppose in a university database there is a table called “Subjects” with a single unique “code” column. Assume you are given the following SQL script:

```sql
CREATE TABLE Subjects
    (code VARCHAR(15) NOT NULL PRIMARY KEY);
INSERT INTO Subjects
VALUES ('CS'), ('ECE'), ('EDCO'),
    ('ART'), ('SOC'), ('MKTG'),
    ('STAT'), ('BC'), ('PHYS'),
    ('FST'), ('FIN'), ('SPAN'),
    ('ARCH');
```

Now the “SELECT code FROM Subjects ORDER BY code” query will return the subject codes in alphabetic order. However, we would like to organize the list in two columns. This is tricky. Let’s first answer a simpler query instead.

For the questions below, make sure both your SQL queries can run “as-is” in SQLite. Use “.header on” command to see column header(s) in SQLite.

**Q5.1.** (10 points) Write a SQL query to create a VIEW called “ALTSKIP” which displays a version of the Subjects table by skipping each next entry. Arrange your list in alphabetic order. For example, on our dataset, your output should look like following:

```
<table>
<thead>
<tr>
<th>FirstC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCH</td>
</tr>
<tr>
<td>BC</td>
</tr>
<tr>
<td>ECE</td>
</tr>
<tr>
<td>FIN</td>
</tr>
<tr>
<td>MKTG</td>
</tr>
<tr>
<td>SOC</td>
</tr>
</tbody>
</table>
```
Hints:
1. Of course your query should be general, and be able to run on any instance of the Subjects table, not just the one shown above.
2. You maybe need a self join and a Group by and Having.
3. The modulus operator in SQLite is % (other systems have other syntax): x % y will give you the remainder after dividing x with y. You may find the following SQL fragment useful: SELECT (COUNT(code)% 2) FROM Subjects will give you 1 if the total number of subjects is odd; 0 otherwise.

Solution:
CREATE VIEW ALTSKIP
AS
SELECT a1.code AS FirstC
FROM Subjects AS a1
    INNER JOIN
    Subjects AS a2
    ON a1.code <= a2.code
GROUP BY a1.code
HAVING (COUNT(a2.code)% 2) =
    (SELECT (COUNT(*)% 2) FROM Subjects);

Q5.2. (10 points) Using the view you made in Q5.1 (“ALTSKIP”), write a SQL query that will display subject codes across in two columns like the following:

<table>
<thead>
<tr>
<th>FirstC</th>
<th>SecondC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCH</td>
<td>ART</td>
</tr>
<tr>
<td>BC</td>
<td>CS</td>
</tr>
<tr>
<td>ECE</td>
<td>EDCO</td>
</tr>
<tr>
<td>FIN</td>
<td>FST</td>
</tr>
<tr>
<td>MKTG</td>
<td>PHYS</td>
</tr>
<tr>
<td>SOC</td>
<td>SPAN</td>
</tr>
<tr>
<td>STAT</td>
<td></td>
</tr>
</tbody>
</table>

Hint: You may need a left outer join. Again, your query should be general and be able to run on any instance of the Subjects table, not just the one shown above. The empty cell can be NULL as well.
Solution:
SELECT a1.code AS FirstC, MIN(a2.code) AS SecondC
    FROM Subjects AS a1
    LEFT OUTER JOIN
    Subjects AS a2
    ON a1.code < a2.code
WHERE a1.code
  IN (SELECT * FROM ALTSKIP)
GROUP BY a1.code
ORDER BY a1.code;