For the first two problems of this exam, use the following relations for a medium-sized library that has about 20 branches:

Branch(branchNo, address, phoneNo, manager): The value of the manager field is a staffNo
Staff(staffNo, branchNo, firstname, lastname, position, salary)
Catalog(catalogNo, publisher, title, author, genre)
Book(bookNo, catalogNo, loanStatus, branchNo)
Member(memberNo, branchNo, firstname, lastname, address, dateRegistered)
Loan(loanNo, memberNo, bookNo, dateLoaned, dateDue)
Request(memberNo, catalogNo, dateRequested, status, branchNo)

1. SQL (24 points CS465, 30 points CS565): Write the following SQL queries:

a. List the firstname, lastname, salary, and position of any staff who make more than 50,000 dollars per year and whose position is either ‘checkout clerk’ or ‘reference librarian’.

   SELECT firstname, lastname, position from Staff
   WHERE salary > 50000
   and position in ('checkout clerk', 'reference librarian');

b. List the firstname, lastname, salary, and branchNo of all managers.

   SELECT s.firstname, s.lastname, s.salary, s.branchNo
   FROM Staff s, Branch b
   WHERE s.staffNo = b.manager;

c. List details of the books checked out by member ‘Smiley Smith’ (firstname ‘Smiley’, lastname ‘Smith’). The information you should print for each book should be book title, genre, dateLoaned, and dateDue. Do not print out the member’s number or name.

   SELECT c.title, c.genre, l.dateLoaned, l.dateDue
   FROM Catalog c, Book b, Loan l
   WHERE c.catalogNo = b.catalogNo and b.loanStatus = 'loaned'
   and b.bookNo = l.bookNo
   and l.memberNo = (SELECT memberNo From Member
                   WHERE firstname='Smiley' and
                   lastname = 'Smith');

   i) You can do a join of l.memberNo with Member.memberNo to avoid the subquery that I used.

   ii) Rather than checking a book’s loan status, you can check whether CURDATE is between dateLoaned and dateDue

d. List the number of requests for each member in the Request relation. For each member, list the member’s number and the number of requests as a column labeled numRequests (you can do this without using a view).
SELECT memberNo, count(*) as numRequests From Request 
Group By memberNo;

e. (CS565 students only) Write an assertion constraint that no member may have more than 20 books checked out (i.e., on loan). Assume the constraint is listed outside of any relation.

CONTRAINT NoLargeLoans
CHECK (NOT EXISTS (select memberNo from Loan
where CURDATE between dateLoaned and dateDue
group by memberNo having count(*) > 20));

2. Relational Algebra (20 points CS465, 24 points CS565)
a. Give a three sentence or less English-language description of the following relational algebra query that a non-computer scientist could understand:

\[ \Pi_{\text{Distinct Member.firstname, Member.lastname}} (\text{Member} \bowtie \text{Member.memberNo} = \text{Loan.memberNo}) \]

List the names of members who currently have a book on loan

b. Using relational algebra notation, write a query that prints the first and last names of all members who belong to the branch at ‘1678 Cardiff Rd.’

\[ \Pi_{\text{Member.firstname, Member.lastname}} (\text{Member} \bowtie \text{Member.branchNo} = \text{Branch.branchNo}) \]

\( (\text{address} = '1678 \text{ Cardiff Rd}' \ (\text{Branch})) \)

c. (CS565 students only) Give a three sentence or less English-language description of the following relational algebra query that a non-computer scientist could understand:

\[ \Pi_{\text{Distinct Catalog.title, Catalog.author}} (\text{Catalog} \bowtie \text{Catalog.catalogNo} = \text{Book.catalogNo} \bowtie \text{Book.bookNo} \bowtie \text{Loan.bookNo} \bowtie \text{Loan.dateLoaned} \bowtie \text{Loan.dateDue}) \]

List the book title and author of all books currently on loan from Branch B003.

3. Physical Design and Normalization (28 points):

a. Examples of anamolies:
   i. insert: If we want to insert a new hotel, there is no way to do so unless we insert NULLs for guestNo and dateFrom, which is a primary key
   ii. insert: if we want to insert a new guest, there is no way to do so unless the guest also makes a reservation
   iii. update: if we change the address of a hotel, we must change it everywhere
iv. update: if we change the address of a guest, we must change it everywhere
v. delete: if we delete all entries related to a guest, we lose the guest’s address
vi. delete: if we delete all reservations at a hotel, we lose the hotel’s information

b. The functional dependencies can be derived directly from the 5 assumptions you were given:
   i. HotelNo \rightarrow HotelName HotelZip
   ii. GuestNo \rightarrow GuestName GuestZip
   iii. RoomNo, HotelNo \rightarrow RoomType RoomPrice
   iv. GuestNo, DateFrom \rightarrow HotelNo, DateTo, RoomNo
   v. GuestNo, DateTo \rightarrow HotelNo, DateFrom, RoomNo
   vi. HotelZip \rightarrow HotelCity
   vii. GuestZip \rightarrow GuestCity

If you have a single functional dependency that maps a zip code to a city, rather than two dependencies as I have here, that is fine, and even desirable.

c. The primary key for the relation is GuestNo and DateFrom: All other attributes can be derived from these two attributes

d. To go from 1st to 2nd normal forms we use functional dependencies that are partial dependencies. There is only one such dependency:

   GuestNo \rightarrow GuestName GuestZip

   and then there is a transitive dependency of GuestZip \rightarrow GuestCity that forces GuestCity to accompany GuestZip to any new relation, so we decompose our relation into two relations as follows:

   GuestInfo(GuestNo, GuestName, GuestZip, GuestCity)
   Booking(GuestNo, DateFrom, DateTo, HotelNo, HotelName, HotelCity, HotelZip, RoomNo, RoomType, RoomPrice)

e. To go from 2nd to 3rd normal forms we use the following transitive dependencies:
   i. HotelNo \rightarrow HotelName HotelZip
   ii. RoomNo \rightarrow RoomType RoomPrice
   iii. HotelZip \rightarrow HotelCity
   iv. GuestZip \rightarrow GuestCity

Because the two zipcode functional dependencies really identify the same relationship of a zip code determining a city, I created just one relation from these two dependencies. If you did not realize that these two dependencies were expressing the same relationship then you would create two different relations. Note however that you are creating redundant information when you do so, which could lead to inconsistency in your database:

   GuestInfo(GuestNo, GuestName, GuestZip)
Note that we did nothing with dependency 5, which represents an alternative candidate key for the Booking relation. However, note that the integrity constraint represented by this functional dependency has been preserved in the final Booking relation, because all five attributes in that functional dependency are present in the Booking relation.

4. **Database Design (28 points):** Consider the following description of an investment firm:
   a. The firm has a number of offices. Each office has a number of investment advisors who service clients. Clients own stocks that pay dividends.
   b. An investor has a name, a unique investor number, a single investment advisor and own zero or more stocks.
   c. Each investor's holding of a stock contains information about the stock's ticker symbol (a 3 or 4 character code that uniquely identifies the stock, such as IBM or APPL), the stock's name (e.g., International Business Machines), the number of shares of stock owned by the investor, and the dividend per share paid by the stock. We only keep track of an investor's cumulative holding—we do not care about the individual transactions that led to this holding.
   d. An investment advisor has a name, a unique advisor number, one or more clients that the advisor services, and an office to which the advisor belongs.
   e. An office has an office number and an address.
   f. A broker is associated with only one office
   g. An office may have multiple advisors,
   h. The dividend per share is determined by the stock (e.g., all holdings of IBM pay the same dividend, such as $.05 dividend per share).

Answer the following questions:
   i. Define the entities and attributes for each entity.

   Office(OfficeNo, address)
   InvestorInfo(InvestorNo, name, advisorNo)
   Advisor(advisorNo, name, OfficeNo)
   StockInfo(Ticker, Name, Dividend)
   Holdings(Ticker, InvestorNo, Quantity)—you may not have this one as it is represented as a relationship in the ER diagram

   ii. Identify the primary and foreign keys for each entity. Please don’t underline the attributes from the previous answer. Instead follow the format of the following example answer:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Primary Key</th>
<th>Foreign Key(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>OfficeNo</td>
<td>----</td>
</tr>
<tr>
<td>InvestorInfo</td>
<td>InvestorNo</td>
<td>advisorNo</td>
</tr>
<tr>
<td>Advisor</td>
<td>AdvisorNo</td>
<td>officeNo</td>
</tr>
</tbody>
</table>
iii. Draw an ER diagram for this investment firm that shows entities, relationships, and multiplicity of relationships.