1. **[20 points]** Consider the following relational schema. Primary keys are underlined and foreign keys are in *italics* (you can see an ER diagram for this on the last page).

```
PERSON: ssn, fname, lname, address, phone, bdate
EMPLOYEE: ssn, job_title, date_hired
SAILOR: ssn, rating
BOATMODEL: model#, model_name, manufacturer
BOAT: reg#, bname, color, model#
RESERVATION: sailor_ssn, date_reserved_for, boat_reg#, emp_ssn
```

Provide SQL answering the following queries:

**a)** List the name (first and last) and job title of all employees that have handled no reservations.

```
SELECT fname, lname, job_title
FROM PERSON JOIN EMPLOYEE USING (ssn)
WHERE ssn NOT IN (SELECT emp_ssn FROM RESERVATION);
```

**b)** Count the number of boats made by each manufacturer. List them by manufacturer name, in decreasing order of the number of boats.

```
SELECT manufacturer, count(*) as total
FROM boat join boatmodel using (model#)
GROUP BY manufacturer
ORDER BY total DESC;
```

**c)** List the first and last name of all sailors and the model name and name of the boat they have reserved, for all reservations in October 2018.

```
SELECT p.fname, p.lname, b.color, m.model_name
FROM person p JOIN sailor s using(ssn)
JOIN reservation r on s.ssn=r.sailor_ssn
JOIN boat b using(reg#)
JOIN boatmodel m using(model#)
WHERE MONTHNAME(date_reserved_for)=October AND
YEAR(date_reserved_for)=2018;
```
d) List the names of all sailors who have rented any of the two most popular boats (a boat is considered popular by counting the number of reservations for it). You may assume that all boats have different numbers of reservations (so ties not a concern).

```
SELECT DISTINCT p.fname, p.lname
FROM person p JOIN sailor s using(ssn)
JOIN reservation r ON s.ssn=r.sailor_ssn
JOIN (SELECT boat_reg#, COUNT(*) total
     FROM reservation
     GROUP BY boat_reg#
     ORDER BY total DESC
     LIMIT 2) n ON r.boat_reg#=n.boat_reg#
```

2) Give an ER diagram for a library as described below. Show relevant constraints. Use Chen notation (as in the text) or Crowsfoot notation (as in MySQLWorkbench).

- Each **employee** has an ssn, fname, lname, and address.
- An employee is classified into one of three categories: **managerial**, **research**, and **floor**. Floor employees are paid by the hour and have an hourly wage rate. The other two categories have a salary. Research workers have a specialty, while managerial workers have a job title.
- **Customers** are identified by their card number, and also have a fname, lname, and address.
- Each **book** is identified by its LCN (Library of Congress Number). It has a title, and one or more **authors**.
- An **author** has as a key an author code, since (fname, lname) does not suffice. We also keep track of their birth date and date of death.
- Customers may check out books. We keep track of the date it was checked out, as well as the date of return, if it has been returned.
- Each time a book is checked out, we want to track which employee was involved in that transaction. Checking out a book can be handled by floor or research staff, **but not managerial staff**.
- Each member of the floor staff has exactly one member of the managerial staff as a supervisor.
3) Derive a relational schema based on the attached ER diagram. Indicate all foreign keys and NOT NULL constraints.

**branch:** branch_id, branch_address

**securityBox:** sec_box_num, branch_id, cust_num
FK: branch_id refers to branch not null
FK: cust_num refers to premiumCustomer

**account:** account_num, account_type, primary_cust_num, second_cust_num
FK: (primary_cust_num) refers to customer(cust_num) not null
FK: (second_cust_num) refers to customer(cust_num)

customer: cust_num, cust_name, cust_address

premiumCustomer: cust_num, prem_status
FK: cust_num refers to customer(cust_num) not null

check: check_num, account_num, check_amount
FK: (account_num) refers to account not null

transfer: acct_num_from, acct_num_to, timestamp, amount
FK: (account_num_from) refers to account not null
FK: (account_num_to) refers to account not null