Q 1: (2 points) Write an SQL query using the university schema to find the ID of each student who has never taken a course at the university. Do this using no subqueries and no set operations (use an outer join).

select s.ID, s.name
from student s
full outer join takes t on s.ID = t.ID
where t.ID is NULL

Q 2: (2 points) Express the following query in SQL using no subqueries and no set operations.
select ID
from student
except
select s_id
from advisor
where i_ID is not null

select s.ID
from student s
full outer join advisor ss on s.ID = ss.s_id
where ss.i_id is NULL

Q 3 (BONUS): (2 points) Consider the following relation definition:

create table manager(
emp_id char(20),
manager_id char(20),
primary key emp_id,
foreign key (manager_id) references manager(emp_id) on delete cascade)

The foreign key constraint means that every manager has to be an employee. Explain what is going to happen when a manager is deleted.

If an employee is deleted, that's all that happens. If a manager is deleted, then the corresponding employee id is also deleted. This is due to the delete cascade. In sql cascade delete means that records in the child table are automatically deleted when the corresponding parent table is deleted.

Q 4: (5 points) Consider the following schema:

employee(ID, person name, street, city
works(ID, company name, salary)
company(company name, city)
manages(ID, manager id)

Write a query to find the ID of each employee with no manager:

(a) Using outer join. (2 marks)

```
select e.ID
from employee e
full outer join manages m on e.ID = m.ID
where m.manager_id = null
```

(b) Without using outer join. (3 marks)

```
select ID
from employee
except
select ID
from manages
where manager_id is not null
```

Q 5: (12 points) Write a Java program that finds all prerequisites for a given course using JDBC. The program should:

- Takes a course id value as input using keyboard.
- Finds the prerequisites of this course through a SQL query.
- For each course returned, repeats the previous step until no new prerequisites can be found.
- Prints the results.

Don't forget to handle the case for cyclic prerequisites. For example, if course A is prerequisite to course B, course B is prerequisite to course C, and course C is prerequisite to course A, do not infinite loop.

Q 6: (5 points) Consider the following schema:
employee(emp name, street, city)
works(emp name, company name, salary)
Write a function avg sal that takes a company name as input and finds the average salary of employees in the company. Then, write a SQL query that uses this function to find companies whose employees earn (on average) higher salary than the company "Losers Inc.".

```
Create function avg_sal(comp_name varchar(20))
        returns numeric(12, 2)
        Deterministic no external action
        contains sql
        begin
        declare result numeric(12, 2) default  0.0
        declare count int default 0
        declare n int default 0
        if comp_name is null then
        return null
        end if
        for c as
```

```
        select salary from works
    do
        set n = n + c.salary
        set count = count + 1
    end for
    set result = n / count
    return result
end
```

```
Select c.company_name
from works c
where (avg_sal(c) > avg_sal('Losers Inc.'))
```

Q 7 (BONUS): (3 points) Consider the following schema:
branch(branch name, branch city, assets)
customer (customer name, customer street, customer city)
loan (loan number, branch name, amount)
borrower (customer name, loan number)
account (account number, branch name, balance)
depositor (customer name, account number)
Write a trigger that enforces the following: On delete of an account, for each owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the depositor relation.

```
Create trigger del_ala
After delete on account
referencing old row as oldrow
for each row
delete from depositor
where depositor.customer_name not in
    (select customer_name
    from depositor
    where account_number <> oldrow.account_number)
end
```