Question 1. [8 MARKS]

Consider the following schema, slightly adapted from an earlier midterm:

Team(tID, school, city). Each tuple records the team ID, school name and city name of a team.
Member(player, team). Each tuple indicates that a given player is a member of a given team.
Score(tID, score, when). Each tuple indicates that a given team earned a given score on a given date.
Member[team] $\subseteq$ Team[tID]
Score[tID] $\subseteq$ Team[tID]

Here is an instance of this schema:

Team:

tID	school	city
13	MGCI	Toronto
99	Humberside	Toronto
14	Woodlands	Mississauga
18	Jarvis	Toronto

Member:

player	team
Ben	14
Tim	99
Macie	14
Juan	13
Liam	99
Javier	18
Macie	99

Score:

tID	score	when
13	100	3-Feb-11
14	237	9-Feb-11
13	88	9-Feb-11
99	150	24-Jan-11
14	110	18-Oct-10
14	200	24-Jan-11

Compute the result of the following valid queries.

Part (a) [2 MARKS]

```
SELECT *
FROM Member
WHERE player <> ALL (
        SELECT player FROM Member
);
```

Solution:

```
 player | team
--------+------
(0 rows)
```

Part (b) [2 MARKS]

```
SELECT *
FROM Member
WHERE player <> ANY (
        SELECT player FROM Member
);
```

Solution:

```
 player | team
--------+------
 Ben    |   14
 Tim    |   99
 Macie  |   14
 Juan   |   13
 Liam   |   99
 Javier |   18
 Macie  |   99
```

Part (c) [2 MARKS]

```
SELECT tID
FROM Team FULL OUTER JOIN Member ON tID=team
WHERE EXISTS (
        SELECT *
        FROM Member M2
        WHERE M2.player <> Member.player AND M2.team = Member.team
);
```

Solution:

```
 tid
-----
  14
  99
  14
  99
  99
```

Part (d) [2 MARKS]

```
SELECT school
FROM (SELECT tid FROM Score GROUP BY tID HAVING COUNT(*) > 1) AS Rep NATURAL JOIN Team;
```

Solution:

```
  school
-----------
 MGCI
 Woodlands
```

Question 2. [12 MARKS]

Here is a schema for the Twitter data you worked with on Assignment 2:

```
CREATE TABLE Profile (
        ID VARCHAR(50),
        name VARCHAR(50),
        location VARCHAR(50),
        url VARCHAR(150),
        bio VARCHAR(500),
        PRIMARY KEY (ID)
);

-- Follows(a,b) indicates that a follows b on Twitter.
CREATE TABLE Follows (
        a VARCHAR(50),
        b VARCHAR(50),
        PRIMARY KEY(a, b),
        FOREIGN KEY (a) REFERENCES Profile(ID)
);
```

1. Write an SQL query to find, for each location, the number of members who (a) have a url that has the empty string as its value and (b) follow no one. Locations will be considered the same only if they are exactly the same string. Don't try to improve that.

   Solution:

   Keep in mind that there are many correct ways to write these queries.

   ```
   create view inactive as
       select distinct id
       from profile
       where url='' and not exists(
           select * from follows where profile.id=follows.a
           );

   select location, count(*)
   from inactive, profile
   where inactive.id=profile.id
   group by location;
   ```

2. Write an SQL query to find, for each person p on the profiles list, the number of people who follow p and who have no followers themselves. Be sure to include p even if there are zero people who follow p and who have no followers themselves.

   Solution:

```
create view noFollowers as
    select profile.id from profile where not exists(
        select * from follows where profile.id = follows.b
        );

create view haveSome as
    select profile.id, count(follows.b)
    from profile, follows, noFollowers
    where profile.id = follows.b and follows.a = nofollowers.id
    group by profile.id;

create view haveNone as
    select profile.id, 0
    from profile
    where not exists(
        select *
        from follows, noFollowers
        where profile.id = follows.b and follows.a = nofollowers.id
    );

(select * from haveSome) union (select * from haveNone);
```

Question 3. [2 MARKS]

In plain English, describe which schools are included in the result of the query from Question 1(d).

Solution:

Schools with a team that has more than one tuple in the score table.

Empty space you can use for rough work. This will not be marked unless you clearly indicate that a solution is written here.

Question 4. [5 MARKS]

The XML document below is valid. Change the DTD so that the root element is a MemberList consisting of one or more Member elements. Define an optional attribute for MemberList that is a string called City. Change the XML data to match the new schema, so that the document is still valid. Use the city value Iqaluit for this very small list of one member.

White space doesn't matter in XML, so don't worry about trying to maintain perfect indentation.

```
<?xml version="1.0" standalone="no" ?>


<!DOCTYPE Member [

    <!ELEMENT Member (Name, Account)>

    <!ELEMENT Name (First, Last)>

    <!ATTLIST Name title CDATA #REQUIRED>

    <!ELEMENT First (#PCDATA)>

    <!ELEMENT Last (#PCDATA)>

    <!ELEMENT Account (#PCDATA)>

]>


<Member>

    <Name title = "Dr">

        <First>Greg</First>

        <Last>Wilson</Last>

    </Name>

    <Account>gvwilson</Account>

</Member>
```

Solution:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE MemberList [
    <!ELEMENT MemberList (Member+)>
    <!ATTLIST MemberList city CDATA #IMPLIED>
    <!ELEMENT Member (Name, Account)>
    <!ELEMENT Name (First, Last)>
    <!ATTLIST Name title CDATA #REQUIRED>
    <!ELEMENT First (#PCDATA)>
    <!ELEMENT Last (#PCDATA)>
    <!ELEMENT Account (#PCDATA)>
]>
<MemberList city="Iqaluit">
    <Member>
        <Name title = "Dr">
            <First>Greg</First>
            <Last>Wilson</Last>
        </Name>
        <Account>gvwilson</Account>
    </Member>
</MemberList>
```