Question 1. [8 MARKS]

Consider the following schema (slightly adapted from one we used in lectures) and the instance of it below.

Students(sID, surName)

Offerings(oID, cID, term, instructor)

Took(sID, oID, grade)

Took[sID] $\subseteq$ Students[sID]

Took[oID] $\subseteq$ Offerings[oID]

Students:

sID	surName
12345	Mielle
99999	Cole
55555	Xiaoyuan
21111	Avery
31111	Nidhi
41111	Rolisha

Offerings:

oID	cID	term	instructor
1	CSC108	20109	Horton
2	CSC108	20109	Engels
3	ENG400	20091	Atwood
4	CSC369	20109	Reid
5	CSC300	20109	Heap
6	CSC343	20111	Horton

Took:

sID	oID	grade
12345	1	77
12345	3	38
12345	4	82
99999	3	80
21111	3	87
21111	1	77
41111	1	82
31111	1	100
31111	3	77
31111	5	100
31111	6	100
55555	1	100
55555	6	100
55555	5	79
55555	4	88

Compute the result of the following valid queries.

Part (a) [2 MARKS]

```
SELECT *
FROM Students
WHERE sID > ANY (
        SELECT sID
        FROM Students
);
```

Solution:

```
  sid  | surname  | campus
-------+----------+--------
 99999 | Cole     | StG
 55555 | Xioayuan | UTM
 21111 | Avery    | StG
 31111 | Nidhi    | UTSC
 41111 | Rolisha  | StG
```

Part (b) [2 MARKS]

```
(SELECT sID FROM Took)
   EXCEPT ALL
(SELECT sID
FROM Took NATURAL JOIN Offerings
WHERE cID LIKE 'CSC3__');
```

Solution:

```
  sid
-------
 12345
 12345
 21111
 21111
 31111
 31111
 41111
 55555
 99999
```

Part (c) [2 MARKS]

```
SELECT count(sID), grade
FROM Took
GROUP BY grade
HAVING max(sID) < 55555;
```

Solution:

```
 count | grade
-------+-------
     2 |    82
     1 |    38
     3 |    77
     1 |    87
```

Part (d) [2 MARKS]

```
CREATE VIEW Pop as (
   SELECT oID
   FROM Took NATURAL JOIN Offerings
   GROUP BY oID
   HAVING count(sID) >= 3
);

(SELECT * FROM (SELECT sID FROM Students) as Them, Pop)
   EXCEPT
(SELECT sID, oID FROM Took);
```

Solution:

```
  sid  | oid
-------+-----
 41111 |   3
 55555 |   3
 99999 |   1
```

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)
);

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 the name and bio of everyone with at least 10 followers, that is, everyone who has at least 10 members who follow them.

   Solution:

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

   ```
   create view popular as
       select b as id from
       follows
       group by b
       having count(*) >= 10;

   select name, bio
   from popular, profile
   where popular.id = profile.id;
   ```

2. Write an SQL query to find the number of different ways that the member with id 'oprah' is connected to the member with id 'barackobama' through 1, 2, or 3 degrees of separation, as defined in assignment two. Remember that

   - a is connected to b with 1 degree of separation if a follows b,
   - a is connected to b with 2 degrees of separation if a follows someone who follows b,

   and so on. Your answer table will have one row and one column.

   Solution:

```
create view oneDegree as
    select count(*)
    from follows
    where a='oprah' and b='barackobama';

create view twoDegrees as
    select count(*)
    from follows f1, follows f2
    where f1.a = 'oprah' and f1.b=f2.a and f2.b = 'barackobama';

create view threeDegrees as
    select count(*)
    from follows f1, follows f2, follows f3
    where f1.a = 'oprah' and f1.b=f2.a and f2.b=f3.a and f3.b = 'barackobama';

create view combo as
    (select * from oneDegree) union all
    (select * from twoDegrees) union all
    (select * from threeDegrees);

select sum(count)
from combo;
```

Question 3. [2 MARKS]

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

Solution:

Students who did not take all of the courses that had at least 3 students enrolled (and such courses that they failed to take).

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 valid XML document below stores data about a Twitter member. Add to the DTD a new element for Member called Tweets, which must come after Name and Account. Define it so that it consists of zero or more elements called Tweet, each of which is just a string. Add to the XML data a single tweet from Greg Wilson: "Playing with Maddie". (Tweets are messages that people post on Twitter.)

```
<?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 Member [
    <!ELEMENT Member (Name, Tweets)>
    <!ATTLIST Member account CDATA #REQUIRED>
    <!ELEMENT Name (First, Last)>
    <!ATTLIST Name title CDATA #REQUIRED>
    <!ELEMENT First (#PCDATA)>
    <!ELEMENT Last (#PCDATA)>
    <!ELEMENT Tweets (Tweet*)>
    <!ELEMENT Tweet (#PCDATA)>
]>
<Member account = "gvwilson">
    <Name title = "Dr">
        <First>Greg</First>
        <Last>Wilson</Last>
    </Name>
        <Tweets>
            <Tweet>Playing with Maddie</Tweet>
        </Tweets>
</Member>
```