Question 1. [8 MARKS]

Consider the following database.

R:

A	B
1	2
3	4
1	3

S:

B	C
1	3
2	4

T:

B	C
2	5
2	3
2	4

Give the results (schema and data) produced by each of the following queries.

Part (a) [2 MARKS]

```
(select R1.A from R R1, R R2 where R1.B <= R2.B)
   intersect all
(select A from R, T where R.B <> 3);
```

Solution:

A
1
1
1
3

Part (b) [2 MARKS]

```
select *
from R, S
where C = any (select C from S natural join T);
```

Solution:

A	B	B	C
1	2	2	4
3	4	2	4
1	3	2	4

Part (c) [2 MARKS]

```
select S.B, avg(T.C)
from (S full outer join T on S.B = T.B)
group by S.B;
```

Solution:

B	avg
2	4.0 (4 is fine)
1	null (shows as blank in psql)

Part (d) [2 MARKS]

```
select R.A
from R, S
where not exists
    (select * from T where T.B = S.B);
```

Solution:

A
1
3
1

Question 2. [4 MARKS]

Consider the following relation instance for Results.

Name	Course	Grade
Johan	CSC309	85
Johan	CSC454	95
Ari	CSC454	67
Vince	CSC108	78
Jiaqi	CSC309	85
Zoya	CSC309	72
Fred	CSC148	82
Ming	CSC207	70
Elaine	CSC309	73
Fred	CSC207	68
Vince	CSC207	68

Part (a) [2 MARKS]

Compute the result of this query on the instance above. Make sure it is clear what is your final answer.

```
select Name from
   (select Course, min(Grade) as X from Results group by Course) as Temp
   natural join Results
where Results.Grade = Temp.X
group by Name
having count(*) > 1;
```

Solution:

Name
Fred
Vince

Part (b) [2 MARKS]

Compute the result of this query on the instance above. Make sure it is clear what is your final answer.

```
create view Enrolments as
   (select Course, count(Crade)
    from Results
    group by Course);

select *
from Enrolments
where Enrolments.count >
   (select avg(Count) from Enrolments);
```

Solution:

Course	count
CSC309	4
CSC207	3

Question 3. [10 MARKS]

Write the queries below in SQL, for the following relational schema. (It is slightly modified from what we saw on Test 1: appointments have a date, but no time.) Keys are underlined and the following inclusion dependencies hold: Appointments[CID] ⊆ Clients[CID], and Appointments[SID] ⊆ Staff[SID].

Clients(CID, name, phone).

Staff(SID, name).

Appointments(CID, date, service, SID)

Part (a) [5 MARKS]

Find the CID of the client who had staff member Giuliano's first appointment, that is, the appointment with the lowest value for date. For simplicity, assume that date has been defined with domain `int`.

Solution:

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

```
select CID
from Appointments natural join Staff
where name = 'Giuliano' and
      date = (select min(date)
              from Appointments natural join Staff
              where name = 'Giuliano');
```

Part (b) [5 MARKS]

For every client, return their CID and the number of appointments they have had with Giuliano. Be sure to include even client's who have had no appointments with Giuliano.

Solution:

```
(select CID, count(*) as numAppts
from appointments, staff
where appointments.SID = staff.SID and
      staff.name = 'Giuliano'
group by CID)
   union
(select CID, 0 as numAppts
from Clients
where CID not in
   (select CID
   from appointments, staff
   where appointments.SID = staff.SID and
         staff.name = 'Giuliano'
   ));
```

Question 4. [6 MARKS]

Indicate if the statements below are correct (true) or incorrect (false). Circle at most one answer for each. **Do not guess.** One point for each correct answer, -1 for each incorrect answer, 0 points if you leave the answer blank. The questions refer to the following tables:

```
create table R (A int, B int);

create table S (B int, C int);

create table T (B int, C int);
```

Part (a) [1 MARK]

The following is an illegal SQL query:

```
SELECT avg(A), B
FROM R, S
WHERE S.C = 97
```

Solution: True. First, we cannot refer to B without specifying whether it comes from relation R or S, since it occurs in both. Second, since we have aggregation, we every attribute in the select must either be aggregated or be included in a group-by clause; B is neither.

Part (b) [1 MARK]

The following queries are equivalent (meaning for all instances of S they return the same results):

```
SELECT COUNT(*) FROM S WHERE B > C
SELECT COUNT(B) FROM S WHERE B > C
```

Solution: True. At first it seems like these queries are not equivalent, because COUNT(B) will not count tuples whose B value is null, but COUNT(*) will, as long as not *all* attributes are null. But consider the WHERE condition. If B is null, the condition will evaluate to *unknown* and that tuple will not be included in the result. So for both queries, tuples with a null value for B are eliminated before the COUNT is performed.

Part (c) [1 MARK]

If a tuple of R has a B value that matches some but not all B values in S, it is included in the result of the following query:

```
SELECT * FROM R WHERE B <> ALL (SELECT B FROM S)
```

Solution: False. In order for a tuple of R to be included, its B value must satisfy the <> comparison for every tuple in S.

Part (d) [1 MARK]

The following queries are not equivalent (meaning for at least one instance of R and S they return different results):

```
SELECT S.C FROM R, S WHERE R.B = S.B
SELECT S.C FROM S WHERE B in (SELECT B FROM R)
```

Solution: True. One will give duplicates while the other will not.

Part (e) [1 MARK]

The following is a legal SQL query:

```
(R) UNION (SELECT B A, C B FROM S)
```

Solution: False: We cannot simply name a relation when giving the operand of a set operator; we must give an actual query. Also, `SELECT B A, C B` needs to have `as` for each: `SELECT B AS A, C AS B`

Part (f) [1 MARK]

Suppose $Q1$ and $Q2$ are subqueries. The number of tuples in $Q1$ `union all` $Q2$ is the number of tuples in the result of $Q1$ plus the number of tuples in $Q2$.

Solution: True. It would not be true if the operation were simply `union`.

Question 5. [8 MARKS]

Consider the following `CREATE TABLE` declarations:

```
create table MovieExec (
   Name char(30),
   ID int primary key );
create table MovieStudio (
   Name char(30) primary key,
   President int references MovieExec(ID)
      on delete set null
      on update cascade );
```

For each of the following questions, assume that
MovieExec = { ('John', 32), ('Arnold', 11), ('Sara', 98) } and
MovieStudio = { ('Disney', 98), ('MGM', 32) }
when the SQL statement is executed. Show the contents of the two tables afterwards. If the statement gives an error, explain.

Part (a) [2 MARKS]

```
insert into MovieStudio values ('WLM Productions', 50);
```

Solution: This gives an error because it violates the foreign key constraint for MovieStudio.President.

Part (b) [2 MARKS]

```
delete from MovieExec where name = 'Sara';
```

Solution:
MovieExec = { ('John', 32), ('Arnold', 11) } and
MovieStudio = { ('Disney', null), ('MGM', 32) }
It was fine to draw this result in the form of a table.

Part (c) [2 MARKS]

```
update MovieExec set ID = 0 where name = 'Sara';
```

Solution:
MovieExec = { ('John', 32), ('Arnold', 11), ('Sara', 0) } and
MovieStudio = { ('Disney', 0), ('MGM', 32) }
It was fine to draw this result in the form of a table.

Part (d) [2 MARKS]

```
insert into MovieExec values ('William', 11);
```

Solution: This gives an error because 11 already occurs as a value for MovieExec.ID, which is a key.