Question #: 1

Consider the following schemas:
students(matric, sname)
workings(pid, matric, since)
projects(pid, pname)
category(pid, cname)
The underlined attributes are the primary key of the schema. You may assume that there can never be any NULL values in the instance. No other constraints can be assumed.
[2 marks] Complete the relational algebra expression below to find the pair of different students' matric number (m1,m2) working on the same project, excluding students not working on any project.
$\pi([\_\_A\_\_], \sigma(\_\_B\_\_ \wedge \_\_C\_\_, (\rho(w1(p1,m1,s1), workings) \times \rho(w2(p2,m2,s2), workings))))$

__A__ = __1__
__B__ = __2__ (*must only be relational operations*)
__C__ = __3__ (*must only be relational operations*)

Question #: 2

Consider the following schemas:
students(<u>matric</u>, sname)
workings(<u>pid, matric</u>, since)
projects(<u>pid</u>, pname)
category(<u>pid, cname</u>)
The underlined attributes are the primary key of the schema. You may assume that there can never be any NULL values in the instance. No other constraints can be assumed.
[2 marks] Complete the relational algebra expression below to find the oldest projects pid in the database. The oldest project is the project with the smallest value of since attribute. You may compare the since attributes with standard relational operators (*e.g.*, <, >, <=, >=, ==, !=).

$\pi([\_\_A\_\_], (\pi([\_\_B\_\_, \_\_C\_\_], workings) - \pi([p2, s2], \sigma(\_\_D\_\_ > \_\_E\_\_, (\rho(w1(p1, m1, s1), workings) \times \rho(w2(p2, m2, s2), workings)))))))$

All answers must be single attributes
__A__ = 1
__B__ = 2
__C__ = 3
__D__ = 4
__E__ = 5

Question #: 3

Consider the following schemas:
students(matric, sname)
workings(pid, matric, since)
projects(pid, pname)
category(pid, cname)
The underlined attributes are the primary key of the schema. You may assume that there can never be any NULL values in the instance. No other constraints can be assumed.
[3 marks] Complete the SQL create table statement below to create the schema above. The first table (i.e., students table) has been filled in for you.

```
CREATE TABLE students (
 matric  VARCHAR(9) PRIMARY KEY,
 sname   VARCHAR(50)
);
CREATE TABLE projects (
 ____1____   VARCHAR(9) ____2____,
 ____3____   VARCHAR(50)
);
CREATE TABLE workings (
 ____4____   VARCHAR(9) REFERENCES ____5____,
 ____6____   VARCHAR(9) REFERENCES ____7____,
 ____8____   DATE,
 PRIMARY KEY(pid, matric)
);
CREATE TABLE category (
 ____9____   VARCHAR(9) REFERENCES ____10____,
 ____11____   VARCHAR(9),
 PRIMARY KEY(pid, cname)
);
```

Question #: 4

Consider the following schemas:
students(matric, sname)
workings(pid, matric, since)
projects(pid, pname)
category(pid, cname)
The underlined attributes are the primary key of the schema. You may assume that there can never be any NULL values in the instance. No other constraints can be assumed.
[3 marks] Complete the SQL query below to find all pair of distinct projects' pid (p1, p2) such that the two projects have exactly the same set of categories.

```
SELECT P1.pid, P2.pid
 FROM projects P1, projects P2
 WHERE __D__
  AND ( SELECT COUNT(*)
          FROM ( SELECT __A__ FROM category C0 WHERE __B__
                 __E__
                 SELECT __A__ FROM category C0 WHERE __C__ ) AS T1 )
      =
      ( SELECT COUNT(*)
          FROM ( SELECT __A__ FROM category C0 WHERE __B__
                 __F__
                 SELECT __A__ FROM category C0 WHERE __C__ ) AS T2 )
```

All blanks must be short answers, they cannot be nested queries.
__A__ = 1
__B__ = 2 (*cannot add additional clauses such as GROUP BY, etc as well as AND and OR conditions*)
__C__ = 3 (*cannot add additional clauses such as GROUP BY, etc as well as AND and OR conditions*)
__D__ = 4
__E__ = 5
__F__ = 6

Question #: 5

Consider the following schemas:
students(<u>matric</u>, sname)
workings(<u>pid, matric</u>, since)
projects(<u>pid</u>, pname)
category(<u>pid, cname</u>)
The underlined attributes are the primary key of the schema. You may assume that there can never be any NULL values in the instance. No other constraints can be assumed.
[3 marks] Consider the following result for the following SQL query

```
SELECT *
 FROM students NATURAL JOIN workings
               NATURAL JOIN projects
               NATURAL JOIN category;
 matric | sname | pid | since | cname
--------+-------+-----+-------+-------
 A0001  | AA    | P01 |  2002 | CA
 A0001  | AA    | P01 |  2002 | CB
 A0001  | AA    | P02 |  2004 | CB
 A0002  | BB    | P01 |  2003 | CA
 A0002  | BB    | P01 |  2003 | CB
 A0003  | CC    | P03 |  2004 | CA
 A0003  | CC    | P03 |  2004 | CC
 A0003  | CC    | P03 |  2004 | CD
 A0004  | AA    | P03 |  2004 | CA
 A0004  | AA    | P03 |  2004 | CC
 A0004  | AA    | P03 |  2004 | CD
```

You are further told that the result of running the SQL query on Question 4 is

```
 pid | pid
-----+-----
 P01 | P04
 P04 | P01
 P03 | P05
 P05 | P03
```

What is the result of the following SQL query?

```
SELECT PID FROM category
EXCEPT ALL
SELECT DISTINCT pid FROM category WHERE cname != 'CA';
```

If you have fewer than 6 rows, write the answer as -. *Leaving your answer as blank constitutes not answering the question.*
Exclude quote symbols in your answer. For instance, write CS2102 insted of 'CS2102' if that is your answer.

 pid
-----
1 ____
2 ____
3 ____
4 ____
5 ____
6 ____

Question #: 6

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[2 marks] Write four completely non-trivial functional dependencies that are *implied* by F but not in F. If you have fewer than 4 functional dependencies, write the answer as -. *Leaving your answer as blank constitutes not answering the question*.
FD1 = ___1___
FD2 = ___2___
FD3 = ___3___
FD4 = ___4___

Question #: 7

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[2 marks] Write at most four superkeys of R with at most 3 attributes. If you have fewer than 4 superkeys, write the answer as -. *Leaving your answer as blank constitutes not answering the question*.
Superkey 1 = __1__
Superkey 2 = __2__
Superkey 3 = __3__
Superkey 4 = __4__

Question #: 8

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[3 marks] Write all keys of R with at most 3 attributes. If you have fewer than 4 keys, write the answer as **-**. *Leaving your answer as blank constitutes not answering the question*.

Key 1 = __1__
Key 2 = __2__
Key 3 = __3__
Key 4 = __4__

Question #: 9

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[5 marks] Write one possible lossless-join valid BCNF decomposition of R into exactly three fragments such that each fragment has exactly three attributes.
Each blank is a single attribute in uppercase.
R1(__1__, __2__, __3__)
R2(__4__, __5__, __6__)
R3(__7__, __8__, __9__)

Question #: 10

[1 mark] Is your decomposition in Question 9 a dependency-preserving decomposition?

A. True
B. False

Question #: 11

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[3 marks] Write one possible minimal cover of F. Write your answer in the same comma-separated format as above, but without the curly brackets.
Minimal Cover = {__1__}

Question #: 12

Consider the schema R(A, B, C, D, E) with F = {BDE -> AE, AD -> C, CD -> E, BE -> AD}.
[5 marks] Write one possible lossless-join dependency-preserving valid decomposition of R that has exactly three fragments. Write the attribute of each fragment as comma-separated value as above.
Each blank is a <u>*comma-separated*</u> *single attribute in uppercase.*
R1(__1__)
R2(__2__)
R3(__3__)

Question #: 13

[4 marks] We know that Armstrong's axioms are both sound and complete. Consider the augmentation rule in Armstrong's axioms:

if `A -> B` then `AC -> BC`

Our aim in this question is to show that the "reverse" of the augmentation rule is not sound. By the "reverse", we meant the following rule:

if `AC -> BC` then `A -> B`

To show that it is unsound, we want to imply the following:

```
{} |= A -> B
```

In other words, any functional dependencies are implied if we allow "reverse" augmentation rule. Note that you are not allowed to use the decomposition and union rule from extended Armstrong's axioms. The following example shows the use of Armstrong's axioms in a proof:

```
A  -> B  [Given]
A  -> C  [Given]|
AB -> A  [Reflexivity]
A  -> AB [Augmentation (1) with A]
AB -> BC [Augmentation (2) with B]
A  -> BC [Transitivity (4) and (5)]
```

To use "reverse" augmentation:

```
AC -> BC [Given]
A  -> B  [Remove C from (1)]
```

Your proof should not exceed 3 lines. If you have fewer than 3 lines, write the answer as -. *Leaving your answer as blank constitutes not answering the question.*

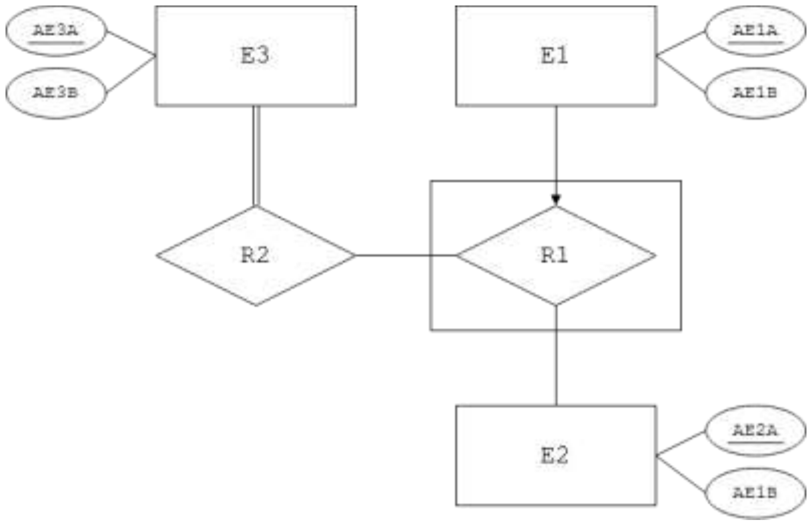
Blank number 1, 2, 4, 5, 7 and 8 are single/multi attributes in uppercase not separated by commas.

Blank number 3, 6 and 9 are reasons.

__1__ -> __2__ [__3__]
__4__ -> __5__ [__6__]
__7__ -> __8__ [__9__]

Question #: 14

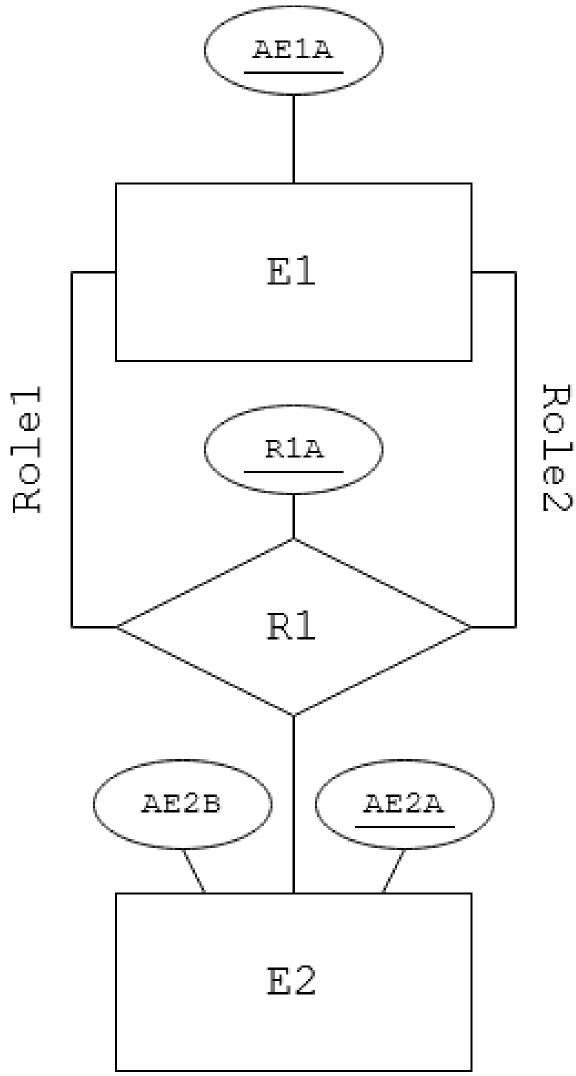
[1 mark] Find the mistake in the ER diagram below. If the mistake is on a line, make sure your pin is on the line outside of any box/diamond/oval. If the mistake is in the arrow make sure your pin is on the arrow. If the mistake is on the box/diamond/oval, make sure your pin is inside them without touching any other lines.



ER1.png

Question #: 15

[1 mark] Find the mistake in the ER diagram below. If the mistake is on a line, make sure your pin is on the line outside of any box/diamond/oval. If the mistake is in the arrow make sure your pin is on the arrow. If the mistake is on the box/diamond/oval, make sure your pin is inside them without touching any other lines.



ER2.png