1. Consider a relation \( R(\overline{A}, \overline{B}, \overline{C}) \) containing the following tuples

\[
\begin{array}{ccc}
A & B & C \\
1 & 2 & 4 \\
1 & 2 & 3 \\
3 & 3 & 1
\end{array}
\]

How many tuples are in the relation

\[\pi_{\overline{A}, \overline{B}}(R) \times \pi_{\overline{A}, \overline{C}}(R)\]

(a) 3
(b) 5
(c) 6 ***
(d) 9

2. How many tuples are in the relation

\[\pi_{\overline{A}, \overline{B}}(R) \bowtie \pi_{\overline{B}, \overline{C}}(R)\]

where \( R \) is the same relation as in Question 1.

(a) 3 ***
(b) 5
(c) 6
(d) 9

---

SEE OVER
3. Suppose that the relation $R$ from Question 1 was in a MySQL table, then how many rows would the following query produce?

```sql
SELECT * FROM
(SELECT A,B FROM R) AS T1
NATURAL JOIN
(SELECT B,C FROM R) AS T2;
```

(a) 3  
(b) *** 5  
(c) 6  
(d) 9

4. Consider the following relation `Marks` that stores students’ average CS and Maths marks and has the following contents:

<table>
<thead>
<tr>
<th>Student</th>
<th>csAvg</th>
<th>mathAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>45</td>
<td>NULL</td>
</tr>
<tr>
<td>Bill</td>
<td>NULL</td>
<td>90</td>
</tr>
<tr>
<td>Christine</td>
<td>100</td>
<td>80</td>
</tr>
</tbody>
</table>

Which rows will be returned by the statement

```sql
SELECT Student
FROM Marks
WHERE (csAvg > mathAvg AND mathAvg > 75 AND csAvg > 90)
    OR (csAvg < 50);
```

(a) Bill and Christine only  
(b) Amy only  
(c) Amy, Bill and Christine  
(d) *** Amy and Christine only
5. Consider a relation $R(A, B, C, D, E)$ with functional dependencies

$$A \rightarrow B, AB \rightarrow CD, D \rightarrow ABC$$

and consider three possible sets of attributes, namely $A$, $AB$ and $CD$.

Which of the three are keys (not necessarily minimal) for $R$?

(a) $AB$ only
(b) $A$ and $AB$ only
(c) $AB$ and $CD$ only
(d) *** $A$, $AB$ and $CD$

6. Suppose we know that $R(A, B, C, D)$ is in Boyce-Codd normal form (BCNF) and that three out of the four FDs listed hold in $R$. Choose the one that does NOT hold in $R$.

(a) $A \rightarrow BCD$
(b) $BC \rightarrow A$
(c) $CD \rightarrow B$
(d) *** $D \rightarrow C$

**Solution:** In all other cases, the three remaining FDs show that the left hand side of the FD is actually a super key

SEE OVER
7. Consider the relations $R(A, B)$ and $S(B, C)$ which currently have the following data

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th></th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

How many rows will be returned from the following MySQL query?

```
SELECT * 
FROM R NATURAL LEFT OUTER JOIN S;
```

(a) 2  
(b) *** 3  
(c) 4  
(d) 9

8. Suppose that an SQL table `data(A,B)` contains $n$ tuples (not necessarily distinct), and then consider the following SQL query

```
FROM data D1, data D2 
```

Which is the strongest correct statement that can be made about the number $m$ of tuples (again not necessarily distinct), returned by this query.

(a) $m = n$  
(b) $n \leq m \leq 2n$  
(c) *** $n \leq m \leq n^2$  
(d) $0 \leq m \leq n$
9. Consider a database used by a home loan provider that contains two relations:

\[
\text{Customer}(\text{name}, \text{mortgage\_id})
\]
\[
\text{Mortgage}(\text{mortgage\_id}, \text{amount})
\]

and a view is defined as

\[
\text{CREATE VIEW mortgage\_info AS}
\]
\[
\text{SELECT name, amount}
\]
\[
\text{FROM Customer JOIN Mortgage USING (mortgage\_id)};
\]

Which of the following sequences of operations creates a new tuple accessible by using the view?

I.
\[
\text{INSERT INTO Customer VALUES(“Smith”, 10118)};
\]
\[
\text{INSERT INTO Mortgage VALUES(10118, 500000)};
\]

II.
\[
\text{INSERT INTO Customer VALUES(“Smith”, NULL)};
\]
\[
\text{INSERT INTO Mortgage VALUES(NULL, 500000)};
\]

(a) *** Just I
(b) Just II
(c) Both I and II
(d) Neither I nor II
10. Consider three transactions running at approximately the same time on a table R(x) where x is of type INT.

```
SELECT SUM(x) FROM R;
COMMIT;
```

```
INSERT INTO R VALUES(1);
INSERT INTO R VALUES(2);
INSERT INTO R VALUES(3);
COMMIT;
```

```
DELETE FROM R WHERE x=3;
DELETE FROM R WHERE x=2;
COMMIT;
```

If the database originally contained integers summing to 50, none of which are 1, 2, or 3, then which of the following values can be returned by the first transaction if all three transactions run with isolation level READ COMMITTED?

(a) 52  
(b) 45  
(c) 53  
(d) *** 54  

SEE OVER
11. Suppose that a database scheme has two tables \( R(A, B) \) and \( S(B, C) \) where all values are integers. Consider the three relational expressions

I \( \pi_{A,C} (R \bowtie \sigma_{B=1} S) \)

II \( \pi_A(\sigma_{B=1}(R)) \times \pi_C(\sigma_{B=1}(S)) \)

III \( \pi_{A,C}(\pi_A(R) \times \sigma_{B=1}(S)) \)

Which of the following statements are true for all instances of \( R \) and \( S \).

(a) I, II and III are always equal
(b) *** I and II are always equal, but III may be different
(c) I and III are always equal, but II may be different
(d) II and III are always equal, but I may be different

12. Suppose we have a database with the following schema

\begin{verbatim}
Student(sid:integer, name:text, wam:real)
Unit(ucode:string, name:text)
Enrolled(ucode:string, sid:integer, mark:integer)
\end{verbatim}

and consider the query:

\begin{verbatim}
SELECT *
FROM Student JOIN Enrolled USING (sid) JOIN Unit USING (ucode)
WHERE Student.wam < 50 AND Unit.ucode = "CITS1402";
\end{verbatim}

If you put two tree-based indexes on these tables, then which would be most useful in speeding up this query (which is assumed to be a very common form of query)?

(a) Indexes on \texttt{Student.sid} and \texttt{Student.wam}
(b) *** Indexes on \texttt{Student.sid} and \texttt{Unit.ucode}
(c) Indexes on \texttt{Enrolled.ucode} and \texttt{Unit.ucode}
(d) Indexes on \texttt{Enrolled.sid} and \texttt{Student.wam}
13. Which SQL keyword is *most similar* to the relational algebra *rename* operator \( \rho \)?

(a) ***\( \text{AS} \)***
(b) \( \text{USING} \)
(c) \( \text{LIKE} \)
(d) \( \text{RENAME} \)

14. Consider three transactions running at approximately the same time on a table \( R(x) \), where \( x \) is of type \( \text{INT} \).

```sql
SELECT SUM(x) FROM R;
COMMIT;

INSERT INTO R VALUES(1);
INSERT INTO R VALUES(2);
INSERT INTO R VALUES(3);
COMMIT;

DELETE FROM R WHERE x=3;
DELETE FROM R WHERE x=2;
COMMIT;
```

If the database originally contained integers summing to 50, none of which are 1, 2, or 3, then which of the following values *cannot* be returned by the first transaction if all three transactions run with isolation level \( \text{SERIALIZABLE} \)?

(a) 51
(b) 50
(c) ***53***
(d) 56
15. Which of the following SQL statements deletes all the data from a table \texttt{Student} (but does not delete the table itself).

(a) \texttt{DELETE * FROM Student;}
(b) ***\texttt{DELETE FROM Student;}
(c) \texttt{DROP TABLE Student;}
(d) \texttt{DELETE Student;}

16. Suppose tables \textit{R} and \textit{S} are created with the following SQL commands

\begin{verbatim}
CREATE TABLE R(c INT PRIMARY KEY, d INT);
CREATE TABLE S(a INT PRIMARY KEY, b INT REFERENCES R(c));
\end{verbatim}

Suppose that \textit{R} contains the tuples (1,2), (2,10), (3,4) and (4,11), while \textit{S} contains the tuples (1,4), (2,3), (3,1) and (4,1).

Which of the following four SQL statements is the only one that will succeed without an error?

(a) \texttt{DELETE FROM R WHERE c = 1;}
(b) \texttt{INSERT INTO S VALUES (1,1);}
(c) ***\texttt{INSERT INTO R VALUES (5,1);}
(d) \texttt{INSERT INTO S VALUES (5,5);}

SEE OVER
17. In a database there is a trade-off between isolation levels and performance, because stronger isolation reduces the ability of the system to interleave statements from different transactions. Which of the following lists the possible isolation levels in *increasing* isolation strength (and thus *decreasing* performance potential)?

(a) **READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE**

(b) **REPEATABLE READ, READ COMMITTED, READ UNCOMMITTED, SERIALIZABLE**

(c) **SERIALIZABLE, READ COMMITTED, REPEATABLE READ, READ UNCOMMITTED**

(d) **REPEATABLE READ, READ UNCOMMITTED, READ COMMITTED, SERIALIZABLE**

18. Suppose a table $T(A, B, C)$ has the following tuples: $(1, 1, 3), (1, 2, 3), (2, 1, 4), (2, 3, 5), (2, 4, 1), (3, 2, 4)$, and $(3, 3, 6)$ and consider the following view definition:

```sql
CREATE VIEW V AS
SELECT A+B AS D, C
FROM T;
```

Consider the following query over the view $V$

```sql
SELECT D, SUM(C)
FROM V
GROUP BY D
HAVING COUNT(*) <> 1
```

Which of the following tuples is in the output?

(a) $(2, 3)$

(b) $(1, 8)$

(c) **$(6, 7)$**

(d) $(3, 5)$
19. Consider the following UML diagram with two classes $A$ and $B$, and one association class $R$

If $|A|$, $|B|$ denote the size (i.e. number of tuples) of $A$ and $B$, then which of the following inequalities must hold according to the constraints shown in the UML diagram.

(a) $|A| = |B|$
(b) $|A| \geq |B|$
(c) *** $|A| \leq |B|$
(d) None of the above

20. In a UML diagram, aggregation is represented by an arrow ending with an open diamond.

Aggregation implies a cardinality constraint, that would otherwise need to be marked in the diagram (in the position marked by the “?”). Which of the following is the cardinality constraint that is implied by the open diamond?

(a) 1..1
(b) *** 0..1
(c) 0..*
(d) 1..*
21. A company assigns its employees (uniquely determined by their employee ID) to various client projects (uniquely determined by a project ID) and keeps track of the percentage of each employee’s time that is allocated to that project. (The percentages do not need to add up to 100% because any time not spent on named projects is spent in the central office). Ultimately project clients are billed weekly according to the level and percentage of each of the employees allocated to their project.

Currently it keeps the information for each week in a flat file format as shown below:

<table>
<thead>
<tr>
<th>ProjId</th>
<th>ProjName</th>
<th>ProjClient</th>
<th>EmpId</th>
<th>EmpName</th>
<th>EmpLevel</th>
<th>Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Evergreen</td>
<td>BHP</td>
<td>101</td>
<td>John Barnes</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>Evergreen</td>
<td>BHP</td>
<td>105</td>
<td>Jill Bates</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>15</td>
<td>Evergreen</td>
<td>BHP</td>
<td>110</td>
<td>James Callan</td>
<td>5</td>
<td>90</td>
</tr>
<tr>
<td>20</td>
<td>SilverMoon</td>
<td>Shell</td>
<td>110</td>
<td>James Callan</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>SilverMoon</td>
<td>Shell</td>
<td>115</td>
<td>Alice Tan</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) Identify the functional dependencies apparent in this system, using reasonable assumptions where necessary.

**Solution:** We have

\[ \text{ProjId} \rightarrow \text{ProjName, ProjClient} \]

\[ \text{EmpID} \rightarrow \text{EmpName, EmpLevel} \]

\[ \text{ProjId, EmpID} \rightarrow \text{Pct} \] as the only obvious FDs.

Marking guide: 3 marks for these three FDs, then 2 marks for either 2-out-of-3 correct or if an extra incorrect FD has been added.

Below this, give 1 mark if the answer shows some idea of what an FD is.

[3 marks]
21 (Continued)

(b) Carefully explain why this database schema is not in BCNF.

Solution:

This schema is not in BCNF because there are non-trivial FDS such as

\[ \text{ProjId} \rightarrow \text{ProjName}, \text{ProjClient} \]

where the left-hand side of the FD is not a super key.

Marking guide: 3 marks if the definition of a BCNF violation is correct and one of the FDS from the first part is demonstrated as an example of an FD. Otherwise, 2 marks for the definition or some explanation of a BCNF violation. Below this, one mark if something sensible is said.

[3 marks]
(c) Using the theory of database normalisation, or otherwise, design a well structured schema for this data, and describe it using a simple UML diagram, just indicating the main classes (including association classes) along with their keys. (You do not need to include participation constraints.)

Solution:

The most appropriate design is to have two classes

Employee(EmpId, EmpName, EmpLevel)
Project(ProjId, ProjName, ProjClient)

and a single association class

Works(EmpId, ProjId, Pct)

The UML diagram follows immediately from this.

[4 marks]
22. Consider the following schema

```
Suppliers(sid:integer, sname:string)
Parts(pid:integer, pname:string, colour:string)
Catalogue(sid:integer, pid:integer, price:real)
```

where the fields Catalogue.sid and Catalogue.pid are foreign keys to the fields of the same name in Suppliers and Parts.

The Catalogue relation lists the prices charged for various parts by the various suppliers (more than one supplier can supply a part).

Write MySQL queries for the following tasks, avoiding unnecessary joins.

(a) List the names of the parts that come in red

Solution:
```
SELECT pname
FROM Parts
WHERE colour = "red";
```

Two marks for correct answer.

(b) You need part #102 (i.e. the part with ID 102). List the names of the suppliers who can supply this part, the price they charge, and present the list in increasing order of price.

Solution:
```
SELECT sname, price
FROM Suppliers JOIN Catalogue USING (pid)
WHERE pid = 102
ORDER BY price ASC;
```

Here, look for the following three things — the join clause, the selection of the part id, and the ordering — which are worth 1 mark each.

If there are then other things that are incorrect, say using an extra table, or using the wrong syntax for something, then let the first such mistake
go, but then mentally start deducting marks if there are 2 or more things incorrect.

[3 marks]
22 (Continued)

(c) List the total number of parts of each colour supplied by supplier #199 (i.e. the supplier with ID 199)

Solution:

```
SELECT colour, COUNT(*)
FROM Catalogue JOIN Parts USING (pid)
WHERE sid = 199
GROUP BY colour;
```

[2 marks]

(d) List the names of parts that come only in red (you may assume that all parts with the same name are just colour variations on the same part).

Solution:

```
SELECT name
FROM parts
WHERE colour = "red"
AND name NOT IN (SELECT name FROM parts WHERE colour <> "red");
OR
SELECT P.pname
FROM Parts P
WHERE P.colour = "red"
AND NOT EXISTS (SELECT * FROM Parts P2 WHERE P2.pname = P.pname AND P2.colour <> "red");
```

[3 marks]
23. Consider the following schema which represents students, units and enrolments.

\[
\begin{align*}
\text{Student}(\text{snum}: \text{integer}, \text{sname}: \text{string}) \\
\text{Unit}(\text{uid}: \text{integer}, \text{ucode}: \text{string}, \text{year}: \text{int}, \text{uname}: \text{string}) \\
\text{Enrolled}(\text{snum}: \text{integer}, \text{uid}: \text{integer}, \text{mark}: \text{integer})
\end{align*}
\]

The meanings of the fields are as follows:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student.snum</td>
<td>Student Number</td>
<td>10212345</td>
</tr>
<tr>
<td>Student.sname</td>
<td>Name</td>
<td>“Gillian Tan”</td>
</tr>
<tr>
<td>Unit.uid</td>
<td>An artificial key</td>
<td>57</td>
</tr>
<tr>
<td>Unit.ucode</td>
<td>Unit code</td>
<td>CITS1402</td>
</tr>
<tr>
<td>Unit.year</td>
<td>Year</td>
<td>2014</td>
</tr>
<tr>
<td>Unit.uname</td>
<td>Unit Name</td>
<td>Databases</td>
</tr>
<tr>
<td>Enrolled.snum</td>
<td>Foreign key to Student.snum</td>
<td></td>
</tr>
<tr>
<td>Enrolled.uid</td>
<td>Foreign key to Unit.uid</td>
<td></td>
</tr>
<tr>
<td>Enrolled.mark</td>
<td>Mark attained</td>
<td>67</td>
</tr>
</tbody>
</table>

(Notice particularly that each unit has a different \text{uid} for each year.)

Write single MySQL queries for the following tasks:

(a) List the student numbers of all the students who passed CITS2211 in 2013 (a pass is 50 or more).

Solution:

The solution needs the tables Unit (to get the ID of CITS2211 in 2013) and Enrolled (to get the marks and student numbers). The table Student is not needed here.

```
SELECT snum
FROM Enrolled JOIN Unit USING (uid)
WHERE ucode = "CITS2211" AND year = 2013 AND mark >= 50;
```

[3 marks]
23 (Continued)

(b) List the unit id and enrolment for all units that had fewer than 10 students enrolled. Name the columns unitID and enrolment.

Solution:

```
SELECT uid AS unitID, COUNT(*) AS enrolment
FROM Enrolled
GROUP BY uid
HAVING enrolment < 10;
```

[3 marks]

(c) Find the student number(s) of the student(s) who got the top mark in CITS2211 in 2013.

Solution:

```
SELECT snum FROM
Enrolled JOIN Unit USING (uid)
WHERE ucode = "CITS2211" AND year = "2013"
AND mark =
(SELECT MAX(mark) FROM Enrolled JOIN Unit USING (uid)
WHERE ucode = "CITS2211" and year = "2013");
```

[4 marks]
24. Explain what a database transaction is, and briefly outline the desirable ACID properties of database transactions — give the name of the property that each of the letters refers to, and give a 1–2 sentence description of what the property means.

Solution:

A database transaction is a collection of one or more statements that accomplishes a single logical task,

Atomicity

Atomicity means that either the whole transaction completes and commits or none of the statements are executed; it can be treated by the user as a single indivisible statement.

Consistency

The database should always be in a consistent state, so while a transaction may temporarily leave the database inconsistent, it must restore it to a consistent state when it commits.

Isolation

The effect of one user’s transactions should be isolated from the effect of those of another user.

Durability

Once a transaction has committed, then it is permanently reflected in the database.

[10 marks]
25. This question refers to the following table-creation statements:

CREATE TABLE Product (  
    pID INT PRIMARY KEY,  
    pname TEXT);  

CREATE TABLE Customer (  
    cID INT PRIMARY KEY,  
    cname TEXT);  

CREATE TABLE ProductOrder (  
    cid INT,  
    pid INT,  
    amount INT,  
    FOREIGN KEY (cid) REFERENCES Customer (cid)  
        ON UPDATE CASCADE ON DELETE CASCADE,  
    FOREIGN KEY (pid) REFERENCES Product (pid)  
        ON UPDATE CASCADE ON DELETE CASCADE );

(a) What is meant by referential integrity in a database? (No more than two sentences.)

Solution:
Referential integrity means that if an attribute in one table, say $R$ refers to a tuple in a second table $S$ then the database should never be allowed to be in a (committed) state where there is no row of $S$ matching the value of the attribute in $R$. 

[3 marks]
(b) In what way do the statements ON UPDATE CASCADE and ON DELETE CASCADE help enforce referential integrity?

Solution:

If the Product or Customer tables are changed by either deleting or altering Product.pid or Customer.cid, then the corresponding rows in ProductOrder are either deleted, or altered to have the new value of pid or cid.

[4 marks]

(c) What SQL statement(s) would you add to the definition(s) of the tables above to enforce the rule that ProductOrder.amount cannot be negative?

Solution:

We would add a CHECK statement, like

CHECK (amount >= 0)

at the end of the definition.

[3 marks]
BLANK PAGE FOR WORKING