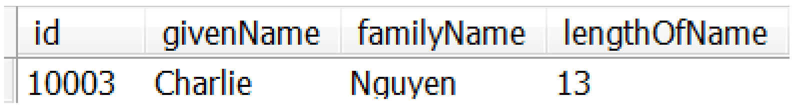
Sample Answers

1. **What is the longest student name? (The length of a student's name is the sum of the lengths of their given and family names) (1)**

```
SELECT *, length(givenName) + length(familyName) as lengthOfName
FROM Student
ORDER BY lengthOfName DESC
LIMIT 1;
```

Using CONCAT is fine. The alternative way of finding a maximum is (full marks) …

```
SELECT * FROM Student
WHERE length(givenName) + length(familyName) =
     (SELECT MAX(length(givenName) + length(familyName))
     FROM Student);
```



id	givenName	familyName	lengthOfName
10003	Charlie	Nguyen	13

1 row returned

2. **List the names of students who have not yet entered any free times (1)**

The simplest:

```
SELECT givenName, familyName FROM Student s
WHERE id NOT IN
     (SELECT student FROM Availability);
```

Also good:

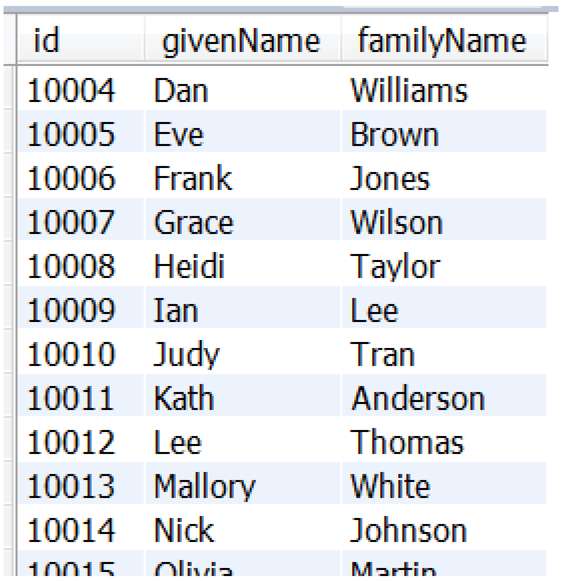
```
SELECT givenName, familyName
FROM Student
LEFT JOIN Availability ON id = student
WHERE Student IS NULL;
```

Not as simple, but gets full marks:

```
SELECT givenName, familyName FROM Student s
WHERE NOT EXISTS
     (SELECT * FROM Availability a
     WHERE a.student = s.id);
```

The following gets HALF marks as it is complex:

```
SELECT * FROM Student
WHERE id in
     (SELECT s.id
     FROM Student s LEFT JOIN Availability a
     ON s.id = a.student
     GROUP BY id
     HAVING COUNT(student) = 0);
```



id	givenName	familyName
10004	Dan	Williams
10005	Eve	Brown
10006	Frank	Jones
10007	Grace	Wilson
10008	Heidi	Taylor
10009	Ian	Lee
10010	Judy	Tran
10011	Kath	Anderson
10012	Lee	Thomas
10013	Mallory	White
10014	Nick	Johnson
10015	Olivia	Martin

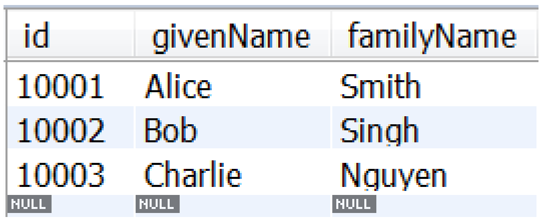
17 rows returned

3. **Which students are free on Wednesday at 10am? (show id and name) (2)**

```
SELECT * FROM Student
WHERE id IN
     (SELECT student FROM Availability
     WHERE day = 'Wed' and hour = 10);
```

Or:

```
SELECT *
FROM Student INNER JOIN Availability
ON id = Student
WHERE day = 'Wed' AND hour = 10;
```

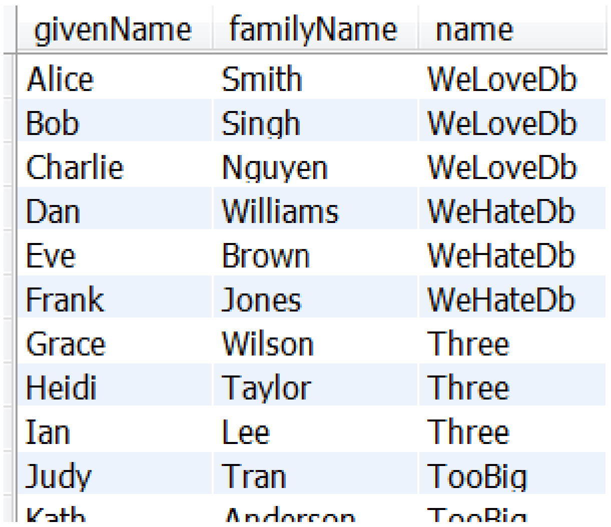


id	givenName	familyName
10001	Alice	Smith
10002	Bob	Singh
10003	Charlie	Nguyen
NULL	NULL	NULL

3 rows returned

4. **List each student's name. For those who are in a group, list also the name of their group. (2)**

```
SELECT givenName, familyName, name FROM Student s
LEFT JOIN StudentInGroup sg ON s.id = sg.studentid
LEFT JOIN Groups g ON sg.groupid = g.id
ORDER BY s.id;
```



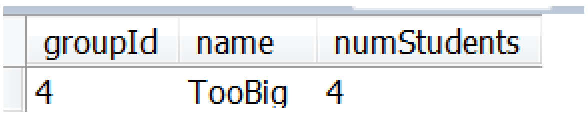
givenName	familyName	name
Alice	Smith	WeLoveDb
Bob	Singh	WeLoveDb
Charlie	Nguyen	WeLoveDb
Dan	Williams	WeHateDb
Eve	Brown	WeHateDb
Frank	Jones	WeHateDb
Grace	Wilson	Three
Heidi	Taylor	Three
Ian	Lee	Three
Judy	Tran	TooBig
Kath	Anderson	TooBig

20 rows returned

(common problem – didn't LEFT join)

5. **For any groups that have more than 3 students, list the group's id, name and number of students. (3)**

```
    SELECT groupId, name, COUNT(*) as numStudents
    FROM StudentInGroup s JOIN Groups g
    ON s.groupId = g.id
    GROUP BY s.groupId
    HAVING numStudents > 3;
```

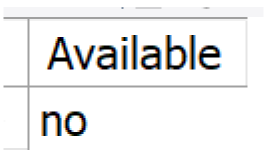


groupId	name	numStudents
4	TooBig	4

1 row returned

6. **Is student "Alice Smith" free at lunch on Wednesdays? (3)**

```
SELECT CASE COUNT(*)
  WHEN 1 THEN 'yes'
  ELSE 'no'
END AS 'Available'
FROM Student s JOIN Availability a ON s.id = a.student
JOIN Calendar c ON a.day = c.day AND a.hour = c.hour
WHERE s.givenName = 'Alice' AND s.familyName = 'Smith'
AND c.day = 'Wed' and c.description = 'lunch';
```



Available
no

1 row returned

A common error is not searching for the word 'lunch' – for this error, subtract 2.

Note also that this question can be interpreted in a way that requires more code to answer:
If we allow that "lunch" can refer to *multiple* hours, then the above code only finds whether Alice is available at "at least one of" the lunch-hours. To see if Alice is available at ALL lunch-hours on Wednesday requires something like the following. This longer version is fine.

```
SELECT
     (SELECT COUNT(*) FROM Student s
     JOIN Availability a ON s.id = a.student
     JOIN Calendar c ON a.day = c.day AND a.hour = c.hour
     WHERE s.givenName = 'Alice' AND s.familyName = 'Smith'
     AND c.day = 'Wed' and c.description = 'lunch')
     =
     (SELECT COUNT(*) FROM Calendar c
     WHERE c.day = 'Wed' and c.description = 'lunch')
;
```

7. **List all times when students 10001 and 10002 are both free. (4)**

```
SELECT * FROM Calendar c
WHERE EXISTS
     (SELECT * FROM Availability a
     WHERE a.day = c.day AND a.hour = c.hour and student = '10001')
     AND EXISTS
          (SELECT * FROM Availability a
          WHERE a.day = c.day AND a.hour = c.hour and student = 
'10002');
```

or

```
SELECT A1.day, A2.hour
FROM Availability A1 JOIN Availability A2
WHERE A1.student = 10001 AND A2.student = 10002
AND A1.day = A2.day AND A1.hour = A2.hour;
```

or

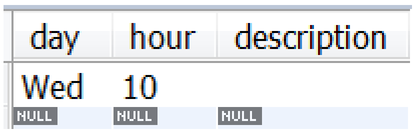
```
SELECT Availability.day, Availability.hour
FROM Availability
WHERE Availability.student IN (10002,10001)
GROUP BY Availability.day, Availability.hour
HAVING COUNT(*) = 2;
```

The following is less elegant but is a reasonable correlated subquery.

```
SELECT * FROM Availability A1
WHERE A1.student = 10001 AND A1.day =
    (SELECT day FROM Availability A2
    WHERE A2.student = 10002 AND A2.hour = A1.hour);
```

However, the following doesn't work, as it checks the day and hour separately:

```
SELECT day, hour FROM Student INNER JOIN Availability
ON Student.Id = Availability.Student
WHERE Student.id = 10001
AND day IN
     (SELECT Availability.day
     FROM Student INNER JOIN Availability
     ON Student.Id = Availability.Student
     WHERE Student.id = 10002 )
AND hour IN
     (SELECT Availability.hour
     FROM Student INNER JOIN Availability
     ON Student.Id = Availability.Student
     WHERE Student.id = 10002 );
```



```
day   hour  description
Wed   10
NULL  NULL  NULL
```

1 row returned

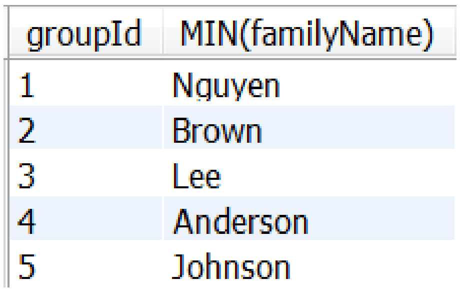
A common error is to check "OR" rather than "AND". This is similar to the red-text example above.

8. **For each group, list the group id and name of the student whose family name is alphabetically first in the group. (4)**

```
SELECT groupId, MIN(familyName)
FROM Groups g
JOIN StudentInGroup sg ON sg.groupid = g.id
JOIN Student s ON s.id = sg.studentid
GROUP BY groupId;
```

Or

```
SELECT Groups.id AS groupId,
min( concat(Student.familyName,', ',Student.givenName) ) AS student
FROM Groups JOIN StudentInGroup
on Groups.id = StudentInGroup.groupId
JOIN Student on Student.id = StudentInGroup.studentId
GROUP BY Groups.id;
```



groupId	MIN(familyName)
1	Nguyen
2	Brown
3	Lee
4	Anderson
5	Johnson

5 rows returned

Some students have written LESS complex SQL, because joining to table Groups is strictly not required – this is ok.

Some students have written MORE complex SQL whose purpose is to show both first and last name - this is ok too.

9. **Which students are free on Wednesdays between 10am and 12 noon? Show their ids and names.** **(5)**

```
SELECT * FROM Student s
WHERE NOT EXISTS
     (SELECT * FROM Calendar c
      WHERE day = 'Wed' and hour >= 10 and hour < 12
      AND NOT EXISTS
            (SELECT * FROM Availability a
             WHERE a.student = s.id
             AND a.day = c.day
             AND a.hour = c.hour));
```

OR

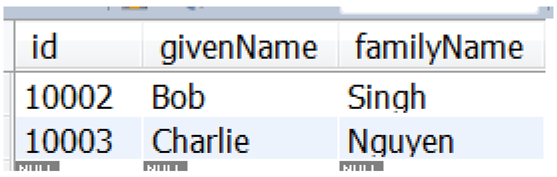
```
SELECT givenName, familyName
FROM Availability JOIN Student ON student = id
WHERE day = 'Wed' AND hour = 10
AND student IN
     (SELECT student
     FROM Availability
     WHERE day = 'Wed' AND hour = 11);
```

OR

```
SELECT * FROM Student
INNER JOIN Availability A1 ON id = A1.student
INNER JOIN Availability A2 ON id = A2.student
WHERE A1.day = 'Wed' AND A1.hour = 10
  AND A2.day = 'Wed' AND A2.hour = 11;
```

OR

```
SELECT * FROM Student
WHERE id IN (
     SELECT student
     FROM Availability
     WHERE day = 'Wed'
     AND hour in (10,11)
     GROUP BY student
     HAVING COUNT(hour) = 2
);
```



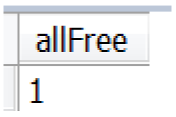
```
id      givenName  familyName
10002   Bob        Singh
10003   Charlie    Nguyen
```

2 rows returned

A common error here is to list students who are free at 10 *OR* 11, rather than 10 *AND* 11. This gets a penalty.

10. **Are the members of 'WeLoveDb' all free on Wednesday at 10am? (5)**

```
     SELECT
          (SELECT COUNT(*) FROM Availability
          WHERE day = 'Wed' AND hour = 10 AND student IN
               (SELECT studentId FROM StudentInGroup
               WHERE groupId in
                    (SELECT id from Groups
                     WHERE name = 'WeLoveDb')))
           =
          (SELECT COUNT(*) FROM StudentInGroup
          WHERE groupId IN
               (SELECT id from Groups
                WHERE name = 'WeLoveDb'))
     as allFree;
```



```
allFree
1
```

1 row returned

Or

```
SELECT (
     SELECT COUNT(G.id)
     FROM StudentInGroup SG LEFT JOIN Groups G ON groupId = id
     LEFT JOIN Availability A
on SG.studentId = A.student AND A.day = 'Wed' AND A.hour = '10'
     WHERE G.name = 'WeLoveDb'
     GROUP BY G.id
     HAVING count(distinct SG.StudentId) = count(distinct
A.student)
) > 0;
```

A common error is to check ANY rather than ALL