Question 6: From the May 2005 Exam

d. List all courses and their average mark for 2003 and for 2004 in a single query.

It is easy enough to calculate just the average mark for 2003:

```sql
SELECT bc, AVG(mark)
FROM allmarks03
GROUP BY bc
```

and we can write a similar query for 2004.

To produce a single table with both sets of results, we use the above as a nested query. That gives a table of banner codes and average marks in 2003. Call it `averages03`. Use a similar query to produce `averages04`. We can now join these two tables in a yet another `SELECT` query.

```sql
SELECT averages03.bc AS "Banner Code",
       averages03.average AS "Average mark in 2003",
       averages04.average AS "Average mark in 2004"
FROM (SELECT bc, AVG(mark) AS average
       FROM allmarks03
       GROUP BY bc) AS averages03,
       (SELECT bc, AVG(mark) AS average
       FROM allmarks04
       GROUP BY bc) AS averages04
WHERE averages03.bc = averages04.bc;
```

First few lines of the result from database:

```
<table>
<thead>
<tr>
<th>bc</th>
<th>avg</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>06</td>
<td>60.401574803149606</td>
<td>60.842975206611570</td>
</tr>
<tr>
<td>06</td>
<td>51.592592592592593</td>
<td>59.386363636363636</td>
</tr>
<tr>
<td>06</td>
<td>58.978571428571429</td>
<td>67.434108527131783</td>
</tr>
<tr>
<td>06</td>
<td>61.027777777777778</td>
<td>59.296296296296296</td>
</tr>
</tbody>
</table>
```

Another solution that appears to work:

```sql
SELECT allmarks03.bc, AVG(allmarks03.mark), AVG(allmarks04.mark)
FROM allmarks03, allmarks04
WHERE allmarks03.bc = allmarks04.bc
```

is actually flawed, but happens to work accidentally. We are joining `allmarks03` and `allmarks04` by matching the `bc` column. If there are 100 records for a particular banner code in 2003, and another different 100 records for it in 2004, the join would involve 10,000 records by considering all possible combinations. Each student mark of 2003 would appear 100 times among these combinations. So, when we calculate the average, this mark would be counted 100 times. That is not a very good way to calculate the average.

The results are still ok in this instance because each mark has exactly the same number of copies. So, the average mark is still the same. But that is a mathematical accident. If we used the same technique for some other calculation, e.g., counting the first class marks, then this form of query would produce the wrong results!
e. Adapt your query from the previous question so that it only lists those courses where in both years there were at least 100 students taking the exam.

```
SELECT averages03.bc AS "Banner Code",
       averages03.average AS "Average mark in 2003",
       averages04.average AS "Average mark in 2004"
FROM (SELECT bc, AVG(mark) AS average
       FROM allmarks03
       GROUP BY bc
       HAVING COUNT(*) >= 100) averages03,
       (SELECT bc, AVG(mark) AS average
       FROM allmarks04
       GROUP BY bc
       HAVING COUNT(*) >= 100) averages04
WHERE averages03.bc = averages04.bc;
```

Only 24 courses satisfy the extra condition in both 2003 and 2004.

Question 7: From the SQL Handout, paragraph 30

a. Write a query which shows for every course the following information: name of course, number of students sitting the examination, average mark, highest mark, lowest mark, and standard deviation.

```
SELECT courses.name,
       count(DISTINCT student) AS students,
       (avg(mark)) AS average,
       max(mark) AS highest,
       min(mark) AS lowest,
       (stddev(mark)) AS "std dev"
FROM courses, allmarks04
WHERE courses.bc = allmarks04.bc
GROUP BY courses.bc, courses.name
ORDER BY courses.name;
```

b. Write a query to list all courses where enrolment has increased between 2002 and 2003.

```
SELECT courses.name,
       lect1.numbers as "2002 enrollment",
       lect2.numbers as "2003 enrollment"
FROM courses, lecturing AS lect1, lecturing AS lect2
WHERE courses.cid = lect1.cid AND courses.cid = lect2.cid
AND lect1.year = 2002 and lect2.year = 2003
AND lect1.numbers < lect2.numbers;
```

We are using two copies of the “lecturing” table, one for selecting the 2002 data and the other for selecting the 2003 data.

c. Write a query which shows for every course the average mark plus the average mark of those students who achieved a pass mark (40% or more).

```
SELECT AllResults.bc AS "banner code",
       AllResults.avgmark AS "average mark",
       PassResults.avgmark AS "average pass mark"
FROM (SELECT bc, avg(mark) AS avgmark
       FROM allmarks04
       GROUP BY bc) AS AllResults,
       (SELECT bc, avg(mark) AS avgmark
       FROM allmarks04
       WHERE mark >= 40
       GROUP BY bc) AS PassResults
WHERE AllResults.bc = PassResults.bc;
```
This solution is not fully right. It does not work correctly if there are courses in which no student passed the exam. The average pass mark is undefined and it is reasonable to omit it. But this query omits the overall average mark as well. (Why?) A better solution can be written using “OUTER JOIN” which we study later in the course.

d. For each course, what was the failure rate in the examination according to “allmarks04”? This question is similar to 6(c), but we need to group results by individual courses.

```
SELECT AllResults.bc, Failures.count * 100 / AllResults.count || '%'
  AS "Percentage of failures"
FROM (SELECT bc, COUNT(*) AS count
      FROM allmarks03
      GROUP BY bc) AS AllResults,
      (SELECT bc, COUNT(*) AS count
      FROM allmarks03
      WHERE mark < 40
      GROUP BY bc) AS Failures
WHERE AllResults.bc = Failures.bc;
```

e. Which staff members are sharing an office with someone else?

```
SELECT DISTINCT staff.firstname, staff.lastname, staff.office
FROM staff, staff AS other
WHERE staff.sid <> other.sid AND staff.office = other.office
ORDER BY staff.lastname;
```

We are using two copies of the “staff” table to obtain data about two staff members respectively.

f. Who taught the course(s) with the largest number of students in each of the years covered by the “lecturing” table?

```
SELECT staff.firstname, staff.lastname, lecturing.year, lecturing.numbers
FROM staff, lecturing
WHERE staff.sid = lecturing.sid
  AND (lecturing.year, lecturing.numbers) IN
    (SELECT year, max(numbers)
     FROM lecturing
     GROUP BY year)
ORDER BY year;
```

The solution uses the maximum pattern discussed in the Handout.

**Question 8: From the May 2007 exam**

a. List the names of all companies whose share price was above 500p at least once during 2006.

```
SELECT DISTINCT name
FROM companies, shares
WHERE companies.cid = shares.cid AND shares.price > 500;
```

Notice that the words “at least once” in the question do not need any special treatment in this form of the query. Since `shares` has prices for individual dates, if a record is found with a price greater than 500, we include it among the results.

It is also possible to code “at least once” using an `EXISTS` operator:

```
SELECT name
FROM companies
WHERE EXISTS (SELECT price
              FROM shares
              WHERE shares.cid = companies.cid
                    AND shares.price > 500);
```
b. List the names of all companies together with their highest share price in 2006.

```sql
SELECT companies.name, MAX(shares.price)
FROM companies, shares
WHERE companies.cid = shares.cid
GROUP BY companies.cid, companies.name;
```

c. List the names of all companies together with the date when their highest share price was first achieved.

```sql
SELECT companies.name, MIN(shares.date)
FROM companies, shares,
(SELECT MAX(price) AS price, cid
 FROM shares
 GROUP BY cid) AS highest,
WHERE companies.cid = highest.cid
AND max.cid = shares.cid
AND highest.price = shares.price
GROUP BY companies.cid, companies.name
```

This shows an elaborate version of the maximum pattern discussed in the Handout. We calculate the maximum prices of all companies using a nested query and then search through the shares table again in the outer query trying to find the record with the maximum price for each company.

d. List the names of all companies whose share price had a percentage increase greater than 20% from 1 January 2006 to 31 December 2006. A solution using nested queries:

```sql
SELECT companies.name
FROM companies,
(SELECT cid, price
 FROM shares
 WHERE date = ‘1 January 2006’) AS jps,
(SELECT cid, price
 FROM shares
 WHERE date = ‘31 December 2006’) AS dps
WHERE jps.cid = dps.cid AND dps.cid = companies.cid AND
dps.price > 1.2 * jps.price;
```

A solution using two copies of the shares table in the FROM clause:

```sql
SELECT companies.name
FROM companies,
 shares AS s1,
 shares AS s2
WHERE companies.cid = s1.cid AND s1.cid = s2.cid AND
 s1.date = ‘1 January 2006’ AND
 s2.date = ‘31 December 2006’ AND
 s2.price > 1.2*s1.price;
```