YellowYellowYellowYellowYellowYellowYellow
There's a table, post_likes. It shows how many likes a particular post has received on a particular day and contains the following fields:
ds (Date)
post_id (int)
num_likes(int)

A sample table may look something like:

```
+--------------+------------+------------+
| date         | post_id    | num_likes  |
+--------------+------------+------------+
| 2014-01-01   | 101        | 2          |
| 2014-01-01   | 102        | 7          |
| 2014-01-01   | 103        | 0          |
| 2014-01-01   | 104        | 9          |
| 2014-01-02   | 102        | 11         |
| 2014-01-02   | 103        | 2          |
| 2014-01-02   | 104        | 1          |
| 2014-01-02   | 105        | 7          |
+--------------+------------+-------------
```

Question 1: Write a query to return the dates that have > 20 likes.

This is a textbook use of HAVING

```
 SELECT ds
 FROM post_likes
 GROUP BY ds
 HAVING SUM(num_likes) > 20;
```

If you don't know how to do this, you can also do it with subqueries. First, get the number of likes per date:

```
SELECT ds, SUM(num_likes) as total_likes
FROM post_likes
GROUP BY ds
```

Then, pick only those dates which have >20 likes

```
SELECT ds
FROM (
SELECT ds, SUM(num_likes) as total_likes
FROM post_likes
GROUP BY ds
) a
WHERE total_likes > 20
```

Question 2: Write a single query to get the count of the following for a ds
*** total number of posts**

*** total number of posts with > 0 likes**

People will probably ask questions about uniqueness--are we guaranteed that each post_id will appear once on each day? Ask if they can do it without that? If not, then say yes; this makes the construction somewhat easier.

For the harder variant, one solution looks like:

```
 SELECT
     ds,
     COUNT(DISTINCT post_id) AS cnt_unique_posts,
     COUNT(DISTINCT CASE WHEN num_likes>0 THEN post_id ELSE NULL END) AS cnt_posts_likes
     --COUNT(DISTINCT IF(num_likes>0, post_id, NULL)) AS cnt_posts_likes
 FROM post_likes
 GROUP BY ds
```

If you don't know DISTINCT, you might have to do something like:

```
SELECT ds,
COUNT(*) as total_likes
SUM(CASE WHEN total_likes > 0 THEN 1 ELSE 0 END) as cnt_post_likes
from (
    SELECT ds, post_id, sum(num_likes) as total_likes
    FROM post_likes
    GROUP BY 1, 2
     ) x
GROUP BY 1
```

Question 3: Write a query to select the post_ids that had 0 likes on 2014-01-01 and > 0 likes on 2014-01-02.

This requires some sort of self-join. Many constructions will work; a simple one is:

```
 SELECT a.post_id
 FROM post_likes a
 JOIN post_likes b
 ON a.post_id = b.post_id
   AND a.ds = '2014-01-01'
   AND b.ds = '2014-01-02'
   AND a.num_likes = 0
   AND b.num_likes > 0
```

Question 4: Write a query to check that there is a unique post_id for each ds (i.e. no duplicate post_ids on a given date). Return 1 if the test passes; 0 if the test fails.

There are multiple solutions here; the simplest comes from realizing that COUNT(*) = COUNT(DISTINCT post_id) implies that the test is correct.

```
 SELECT
   ds,
   IF(COUNT(*) = COUNT(DISTINCT post_id), 1, 0) AS test_result
 FROM post_likes
 GROUP BY ds
```

Because SQL gives us a lot of ways to do the same thing, a less efficient but still correct solution might look like:

```
SELECT ds, CASE WHEN max(num_post_ds) = 1 THEN 1 ELSE 0 END as tvar
FROM (
    SELECT
    ds, post_id, count(*) as num_post_ds
    FROM post_likes
    GROUP BY 1, 2
    ) x
GROUP BY 1
```