**Question: 1**
Given the following tables:

```
sql> SELECT * FROM runners;
+----+--------------+
| id | name         |
+----+--------------+
|  1 | John Doe     |
|  2 | Jane Doe     |
|  3 | Alice Jones  |
|  4 | Bobby Louis  |
|  5 | Lisa Romero  |
+----+--------------+

sql> SELECT * FROM races;
+----+----------------+-----------+
| id | event          | winner_id |
+----+----------------+-----------+
|  1 | 100 meter dash |  2        |
|  2 | 500 meter dash |  3        |
|  3 | cross-country  |  2        |
|  4 | triathalon     |  NULL     |
+----+----------------+-----------+
```

What will be the result of the query below?

**SELECT** * **FROM** runners **WHERE** id **NOT IN** (**SELECT** winner_id **FROM** races)

Explain your answer and also provide an alternative version of this query that will avoid the issue that it exposes.

**Answer.** The query provided:

**SELECT** * **FROM** runners **WHERE** id **NOT IN** (**SELECT** winner_id **FROM** races)

attempts to select all runners who have not won any race. However, there's a potential issue with this query related to the presence of `NULL` values in the `winner_id` column of the `races` table. When the subquery `(SELECT winner_id FROM races)` returns `NULL`, the behavior of the `NOT IN` clause can lead to an empty result set. This happens because the SQL standard specifies that comparisons with `NULL` using the `=` or `<>` operators yield `NULL`, and when `NULL` is part of the `NOT IN` list, the entire `NOT IN` expression evaluates to `FALSE` for all rows, so no records from `runners` will be selected.

As for the result of the query with the given tables, since there is a `NULL` in the `winner_id` column, the result of this query will likely be an empty set, which is probably not the intended outcome.

An **alternative way** to write this query that avoids the issue with `NULL` values is to use a `LEFT JOIN` and look for `NULL` in the joined table, like this:

**SELECT runners.* FROM runners**

**LEFT JOIN races ON runners.id = races.winner_id**

**WHERE races.winner_id IS NULL;**

### Question 2

Given two tables created as follows

```
create table test_a(id numeric);

create table test_b(id numeric);

insert into test_a(id) values
  (10),
  (20),
  (30),
  (40),
  (50);

insert into test_b(id) values
  (10),
  (30),
  (50);
```

Write a query to fetch values in table test_a that are and not in test_b without using the NOT keyword.

**Answer**

**SELECT** test_a.id FROM test_a

**LEFT JOIN** test_b ON test_a.id = test_b.id

**WHERE** test_b.id IS NULL;

**Question 3.**

Given the following tables**:**

```
SELECT * FROM users;

user_id  username
1        John Doe
2        Jane Don
3        Alice Jones
4        Lisa Romero

SELECT * FROM training_details;

user_training_id  user_id  training_id  training_date
1                 1        1            "2015-08-02"
2                 2        1            "2015-08-03"
3                 3        2            "2015-08-02"
4                 4        2            "2015-08-04"
5                 2        2            "2015-08-03"
6                 1        1            "2015-08-02"
7                 3        2            "2015-08-04"
8                 4        3            "2015-08-03"
9                 1        4            "2015-08-03"
10                3        1            "2015-08-02"
11                4        2            "2015-08-04"
12                3        2            "2015-08-02"
13                1        1            "2015-08-02"
14                4        3            "2015-08-03"
```

Write a query to to get the list of users who took the a training lesson more than once in the same day, grouped by user and training lesson, each ordered from the most recent lesson date to oldest date.

**Answer:**

```
SELECT user_id, training_id, training_date, COUNT(*) as times_taken

FROM training_details

GROUP BY user_id, training_id, training_date

HAVING COUNT(*) > 1

ORDER BY training_date DESC, user_id, training_id;
```

### Question: 4

Consider the Employee table below.

| Emp_Id | Emp_name | Salary | Manager_Id |
|---|---|---|---|
| 10 | Anil | 50000 | 18 |
| 11 | Vikas | 75000 | 16 |
| 12 | Nisha | 40000 | 18 |
| 13 | Nidhi | 60000 | 17 |
| 14 | Priya | 80000 | 18 |
| 15 | Mohit | 45000 | 18 |
| 16 | Rajesh | 90000 | – |
| 17 | Raman | 55000 | 16 |
| 18 | Santosh | 65000 | 17 |

Write a query to generate below output:

| Manager_Id | Manager | Average_Salary_Under_Manager |
|---|---|---|
| 16 | Rajesh | 65000 |
| 17 | Raman | 62500 |
| 18 | Santosh | 53750 |

**Answer:**

```
SELECT
    m.Emp_Id AS Manager_Id,
    m.Emp_name AS Manager,
    AVG(e.Salary) AS Average_Salary_Under_Manager
FROM
    employees e
INNER JOIN
    employees m ON e.Manager_Id = m.Emp_Id
GROUP BY
    m.Emp_Id, m.Emp_name
ORDER BY
    m.Emp_Id;
```