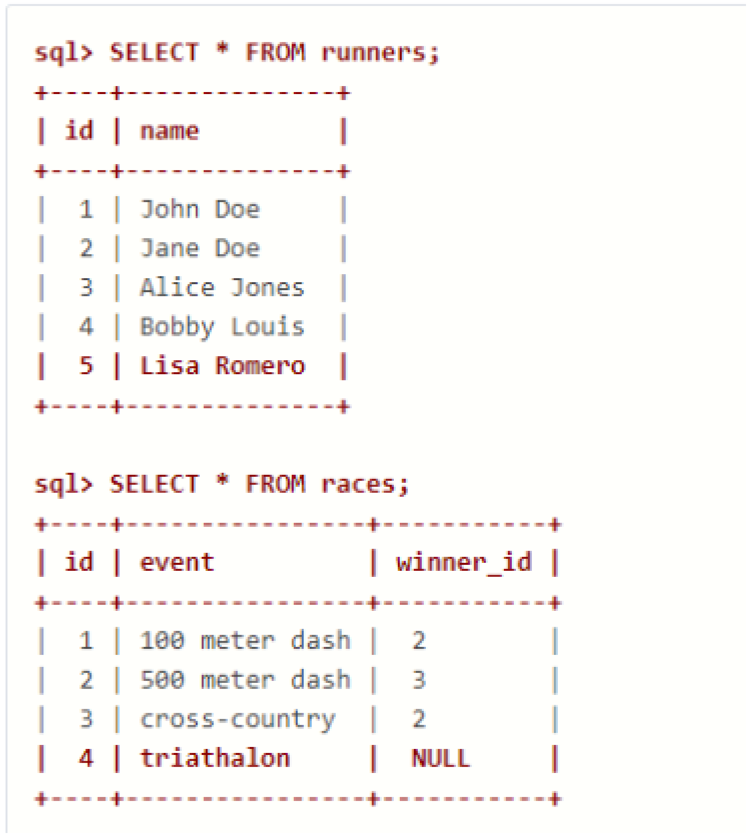
Question: 1



```
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.

Ans:

The query you provided aims to retrieve all the runners who have not won any races. It selects all records from the "runners" table where the "id" column does not exist in the list of winner IDs from the "races" table.

However, this query may not yield the correct result if there are NULL values in the "winner_id" column of the "races" table. This is because the comparison `id NOT IN (SELECT winner_id FROM races)` would not evaluate as expected when there are NULL values involved. In SQL, comparisons with NULL values typically result in an unknown or NULL outcome, which can affect the logic of the query.

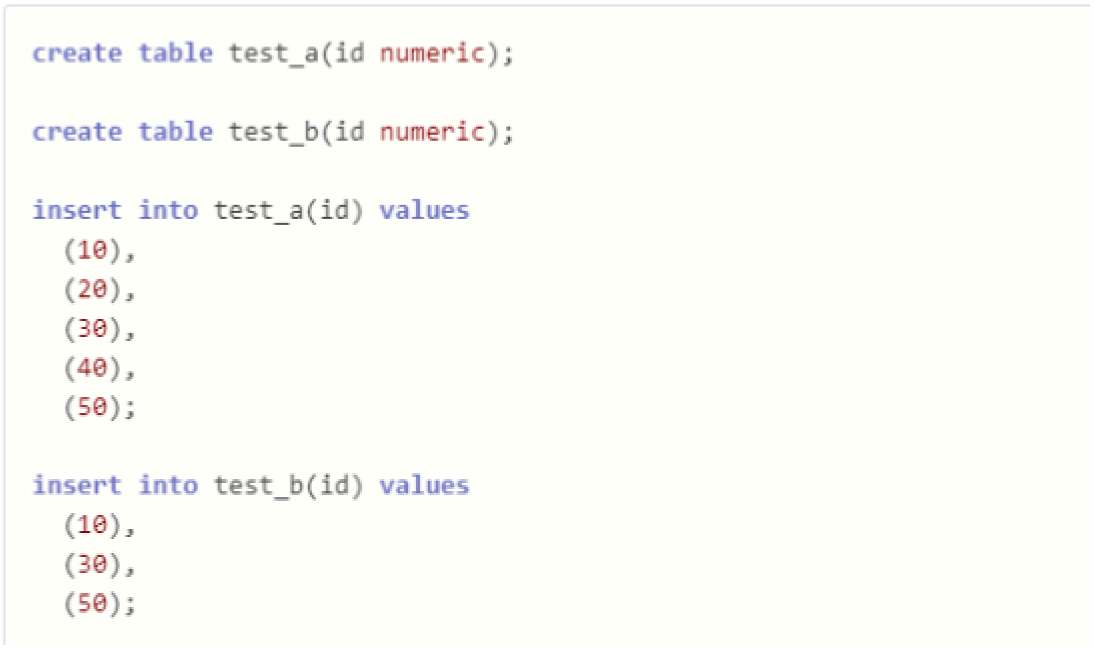
To avoid this issue, you can rewrite the query using a LEFT JOIN instead of a subquery:

```
SELECT runners.*

FROM runners

LEFT JOIN races ON runners.id = races.winner_id

WHERE races.winner_id IS NULL;
```

This query joins the "runners" table with the "races" table based on the "id" and "winner_id" columns, respectively. It selects all records from the "runners" table where there is no matching entry in the "races" table, thereby ensuring that NULL values are handled correctly.

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.

Ans:

SELECT test_a.*

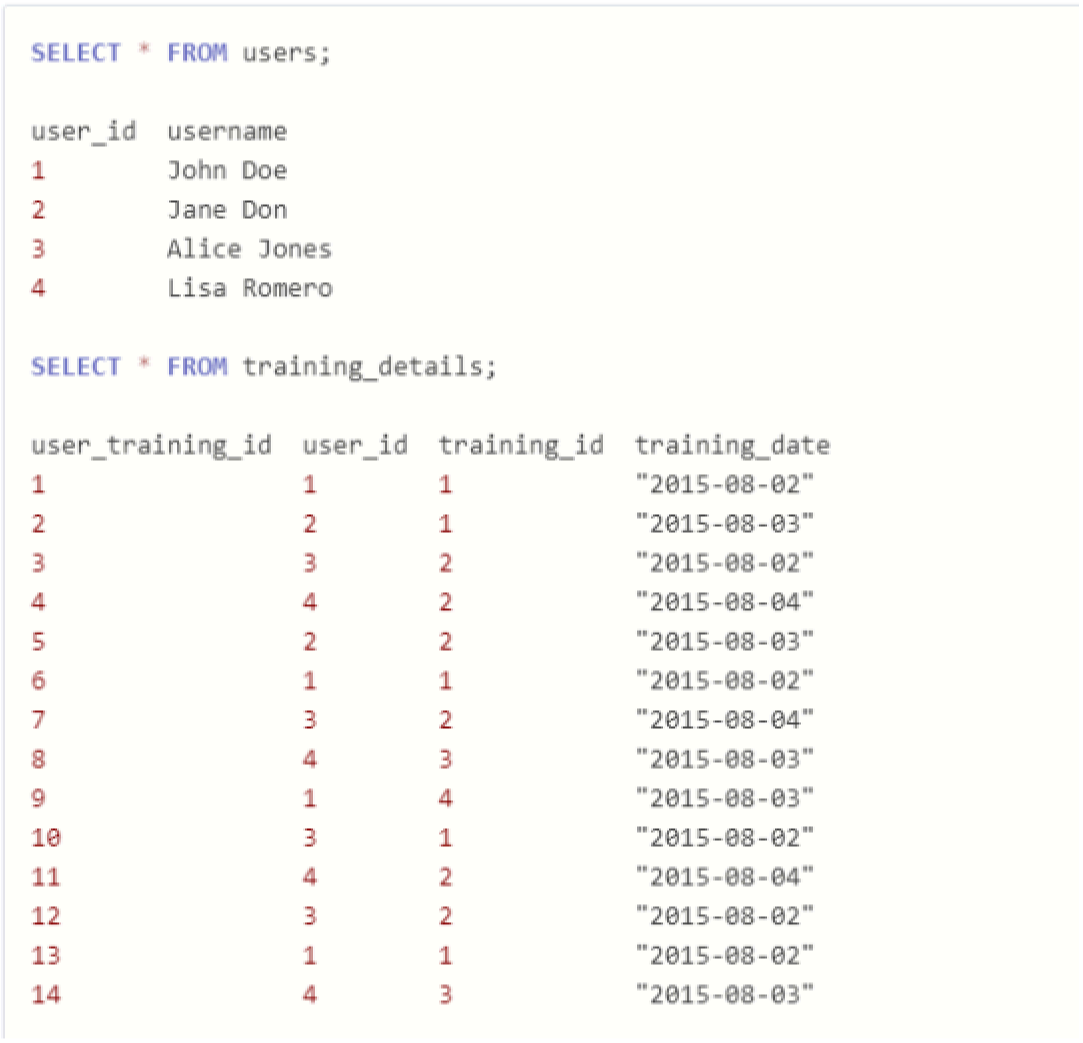
FROM test_a

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"
```

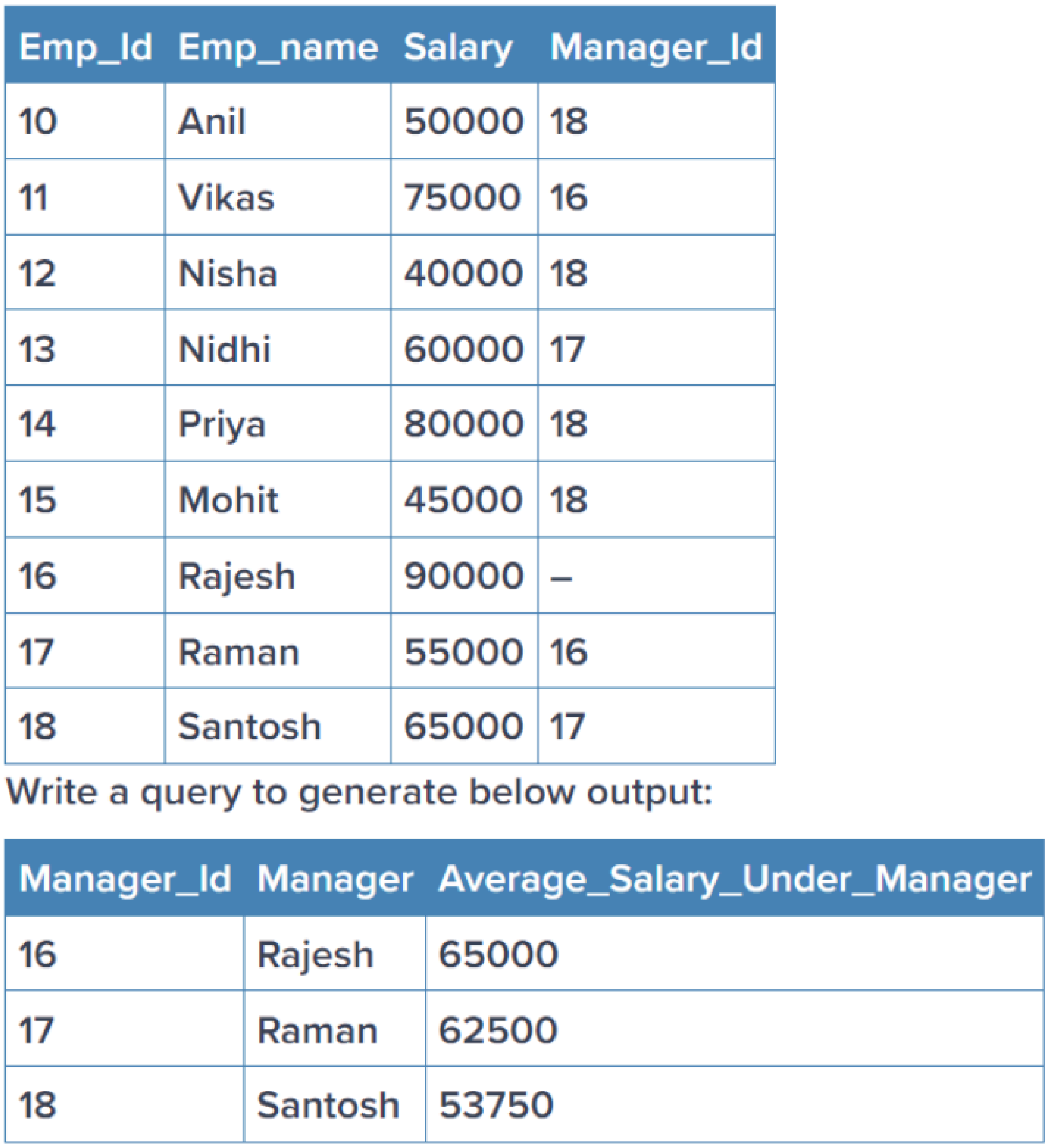
Write a query to to get the list of users who took the a training lesson more than once in the same day, grouped by user and training lesson, each ordered from the most recent lesson date to oldest date.

Ans:

```
SELECT u.User_id, u.User_name, td.User_training_id, td.Training_id, td.Training_date

FROM Users u

JOIN Training_details td ON u.User_id = td.User_id

GROUP BY u.User_id, td.Training_id, td.Training_date

HAVING COUNT(*) > 1

ORDER BY td.Training_date DESC;
```

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

Ans:

```
SELECT e.Manager_ID, MAX(m.Emp_name) AS Manager_Name, AVG(e.Salary) AS
Avg_Salary_under_manager

FROM employee e

JOIN employee m ON e.Manager_ID = m.Emp_id

GROUP BY e.Manager_ID

ORDER BY e.Manager_ID;
```