SQL

Questions 1:
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:
This query selects all rows from the runners table where the id column is not present in the result of the subquery, which selects the winner_id column from the races table.

The potential issue with this query arises if the subquery (SELECT winner_id FROM races) returns any NULL values. If any winner_id in the races table is NULL, the entire subquery will return NULL. In SQL, comparing a value to NULL using NOT IN will always return NULL, not true. So, if the subquery returns any NULL values, the outer query will not return any rows at all. This could lead to unexpected results.

To avoid this issue, you can rewrite the query using a LEFT JOIN with a NULL check in the WHERE clause. Here's the alternative version of the query:

```
SELECT runners.*
FROM runners
LEFT JOIN races ON runners.id = races.winner_id
WHERE races.winner_id IS NULL;
```

This query performs a LEFT JOIN between the runners table and the races table on the id and winner_id columns, respectively. Then, it filters out the rows where there is no matching winner_id in the races table (i.e., where races.winner_id is NULL). This approach ensures that rows with NULL winner_id values are properly handled and included in the result set.

Questions 2:
Given two tables created as follows
Write a query to fetch values in table test_a that are and not in test_b without using the NOT keyword.

Answer:
create table test_a(id integer); insert into test_a(id) values(10),(20),(30),(40),(50); select * from test_a
create table test_b(id integer); insert into test_b(id) values(10),(20),(30); select* from test_b
SELECT test_a.id FROM test_a LEFT JOIN test_b ON test_a.id = test_b.id WHERE test_b.id IS NULL;
Questions 3:

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, user_training_id, training_date from(
Select user_id, user_training_id, training_date, count(*) as  lesson_count
From training_details
Group by user_id, user_training_id, training_date
Having count(*)>1
)
As duplicates
Order by user_id, user_training_id, training_date DESC;
```

Questions 4:

Answer

```
SELECT Manager_id, Manager, AVG(Salary) AS Average_Salary, COUNT(*) AS
Under_Manager

FROM (

   SELECT Manager_id, Emp_name AS Manager, Salary

   FROM employee

   WHERE Manager_id IS NOT NULL

) AS managers

GROUP BY Manager_id;
```