**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;