SQL

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     |
+----+----------------+------------+
```

Solution:

SELECT * FROM runners WHERE id <> 2 and id <> 3 and id <> null;

`id <> null` is unknown by definition, so the whole `where` clause evaluates to either `false` (for ids: 2, 3) or unknown (for ids: 1, 4, 5) because of the `and` operators, thus no result returned.

If we want to run this query disregarding the nulls in `races`, you can use:

Correct Code :

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

Question: 2

Given two tables created as follows

```sql
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.

Solution:

SELECT * FROM test_a EXCEPT SELECT * FROM test_b;

EXCEPT is a operator, which is specifically designed to find rows present in one table but not in another.

Question: 3

Given the following tables:

```sql
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.

Solution:

SELECT u.user_id, td.training_id, COUNT(*) AS lesson_count, MAX(td.training_date) AS most_recent_date FROM users u INNER JOIN training_details td ON u.user_id = td.user_id GROUP BY u.user_id, td.training_id HAVING lesson_count > 1 -- Filter for users with more than 1 lesson in a day ORDER BY u.user_id, most_recent_date DESC;

where,
u= users
td=training_details

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

Solutions:

SELECT manager_name, AVG(salary) AS average_salary, COUNT(*) AS employee_count
FROM Employee e INNER JOIN Employee m ON e.Manager_Id = m.Emp_id WHERE m.Manager_Id IS NULL GROUP BY manager_nameORDER BY manager_name;