**PPM Technology Team Exam**

**SQL:**

1. Given the following tables:

|  |  |
| --- | --- |
| Name | ID |
| Adam Cohen | 1 |
| Noa Sabath | 2 |
| Sharon Finish | 3 |
| Lee Ronen | 4 |
| Michal Shalom | 5 |

* 1. SELECT \* FROM Employees;

|  |  |  |
| --- | --- | --- |
| Manager\_ID | Name | ID |
| 3 | Sales | 1 |
| 5 | Technology | 2 |
| 2 | Customer Support | 3 |
| NULL | Finance | 4 |
| 1 | HR | 5 |

* 1. SELECT \* FROM Departments;

What will be the result of the query below:

SELECT \* FROM Employees WHERE id NOT IN (SELECT manager\_id FROM Departments)

Answer: nothing

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

In the case a result of a nested query contains null, “not in” condition will give always false

There are two right solutions:

***SELECT \* FROM Employees WHERE not exists (SELECT \* FROM Departments where manager\_id = Employees.id)***

***select \* from employees left join departments on employees.id=manager\_id where manager\_id is null***

The first solution will be more effective if the field manager\_id is an index

1. 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.

SELECT a.id FROM test\_a a left join test\_b b on a.id=b.id where b.id is NULL

1. Given the following tables:

|  |  |
| --- | --- |
| Name | ID |
| Adi Shemesh | 1 |
| Shira Levy | 2 |
| Aharon Ferra | 3 |
| Gal Talmon | 4 |
| Erez Yosef | 5 |

SELECT \* FROM users;

SELECT \* FROM training\_details;

|  |  |  |  |
| --- | --- | --- | --- |
| training\_date | training\_id | user\_id | user\_training\_id |
| 01/08/2019 | 2 | 1 | 1 |
| 07/08/2019 | 4 | 2 | 2 |
| 03/08/2019 | 3 | 3 | 3 |
| 09/08/2019 | 1 | 4 | 4 |
| 05/08/2019 | 2 | 5 | 5 |
| 06/08/2019 | 2 | 4 | 6 |
| 07/08/2019 | 5 | 2 | 7 |
| 05/08/2019 | 3 | 5 | 8 |
| 09/08/2019 | 4 | 3 | 9 |
| 10/08/2019 | 1 | 2 | 10 |
| 05/08/2019 | 4 | 5 | 11 |
| 12/08/2019 | 3 | 2 | 12 |
| 09/08/2019 | 2 | 4 | 13 |
| 01/08/2019 | 1 | 1 | 14 |

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

SELECT u.name FROM users u join training\_details on u.id = user\_id group by u.name, training\_date having count(\*) > 1 order by training\_date

1. Write a query to get the last id from specific table without using the max function?

Select id from A order by id desc limit 1

1. Given the following table named A:

|  |
| --- |
| X |
| 5 |
| -7 |
| 3 |
| -1 |
| -2 |
| 0 |
| 9 |

Write a **single** query to calculate the sum of all positive values of x and the sum of all negative values of x.

select sum(negative.num) as negative\_sum, sum(positive.num) as positive\_sum from (select \* from table\_a where num < 0) as negative

full join (select \* from table\_a where num > 0) as positive on negative.id=positive.id

1. Write a SQL statement to find the highest purchase amount ordered by the each customer on a particular date with their ID, order date and highest purchase amount

SELECT customer\_id, order\_date, max(purchase\_amount) highest\_purchase\_amount FROM orders group by cutomer\_id, order\_date order by customer\_id

Sample table: orders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| salesman\_id | customer\_id | order\_date | purchase\_amount | order\_no |
| 4001 | 6002 | 02/05/2020 | 150.9 | 8002 |
| 4003 | 6005 | 03/05/2020 | 45.2 | 8003 |
| 4004 | 6002 | 04/05/2020 | 99 | 8004 |
| 4002 | 6003 | 06/05/2020 | 1100 | 8005 |
| 4001 | 6005 | 05/05/2020 | 850 | 8006 |
| 4003 | 6002 | 03/05/2020 | 150 | 8007 |
| 4002 | 6005 | 02/05/2020 | 280 | 8008 |
| 4001 | 6004 | 05/05/2020 | 360 | 8009 |
| 4003 | 6002 | 03/05/2020 | 450 | 8010 |
| 4002 | 6003 | 04/05/2020 | 620 | 8011 |
| 4001 | 6005 | 06/05/2020 | 102.5 | 8012 |
| 4002 | 6002 | 02/05/2020 | 745 | 8013 |
| 4004 | 6004 | 03/05/2020 | 628 | 8014 |

**Good Luck!**