1. With SQL, how can you insert a new record into the "Persons" table?

- INSERT INTO Persons VALUES ('Jimmy', 'Jackson')
- INSERT ('Jimmy', 'Jackson') INTO Persons
- INSERT VALUES ('Jimmy', 'Jackson') INTO Persons

2. With SQL, how can you insert "Olsen" as the "LastName" in the "Persons" table?

- INSERT INTO Persons ('Olsen') INTO LastName
- INSERT INTO Persons (LastName) VALUES ('Olsen')
- INSERT ('Olsen') INTO Persons (LastName)

3 How can you change "Hansen" into "Nilsen" in the "LastName" column in the Persons table?

- MODIFY Persons SET LastName='Nilsen' WHERE LastName='Hansen'
- MODIFY Persons SET LastName='Hansen' INTO LastName='Nilsen
- UPDATE Persons SET LastName='Nilsen' WHERE LastName='Hansen'
- UPDATE Persons SET LastName='Hansen' INTO LastName='Nilsen'

4. With SQL, how can you delete the records where the "FirstName" is "Peter" in the Persons Table?

- DELETE FirstName='Peter' FROM Persons
- DELETE ROW FirstName='Peter' FROM Persons
- DELETE FROM Persons WHERE FirstName = 'Peter'

5. With SQL, how can you return the number of records in the "Persons" table?

- SELECT LEN(*) FROM Persons
- SELECT COLUMNS(*) FROM Persons
- SELECT COUNT(*) FROM Persons
- SELECT NO(*) FROM Persons

6. Which operator is used to select values within a range?

- WITHIN
- RANGE
- BETWEEN

7. The NOT NULL constraint enforces a column to not accept null values.

- True
- False

8. Which operator is used to search for a specified pattern in a column?

- From
- Like
- Get

9. Select the correct example of JOINing three tables

- SELECT * FROM actor JOIN casting BY actor.id = actor.id JOIN movie BY movie.id = movie.id
- SELECT * FROM actor JOIN casting JOIN movie ON actor.id = actor.id AND movie.id = movie.id

- SELECT * FROM actor JOIN casting ON actor.id = actor.id AND movie ON movie.id = movie.id

- SELECT * FROM actor JOIN casting ON actor.id = actor.id JOIN movie ON movie.id = movie.id

10. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in table **Employee** having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee_id.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

```
SELECT name FROM Employee WHERE salary > 2000 AND months < 10 ORDER BY employee_id;
```

11. We define an employee's *total earnings* to be their monthly *Salary X Months* worked, and the *maximum total earnings* to be the maximum total earnings for any employee in the **Employee** table. Write a query to find the *maximum total earnings* for all employees as well as the total number of employees who have maximum total earnings.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

For example, if the maximum *earnings* value is 69952. The only employee with *earnings* = 69952 is *Kimberly*, so we print the maximum *earnings* value (69952) and a count of the number of employees who have earned (which is 1).

*Hint: use (months * salary) as a segmentation variable, count number of records in each segment, order and limit 1 to get the final result*

Original Thought: Use (month*salary) to make a segmentation variable first, then we use order by desc and group by to make a column of maximized salary ranking. After that we can use it as a subquery and use count(*) to find the number of employees who are receiving the maximum earnings.

Best Answer:
SELECT month*salary, count(*) FROM Employee GROUP BY 1 ORDER BY 1 DESC LIMIT 1;

In this way, we can limit the limit the column to be only displaying the "max earning" category and then we count the number of employees.