**Question 101. Write an SQL query to show the second most recent activity of each user. If the user only has one activity, return that one. A user cannot perform more than one activity at the same time. Return the result table in any order.**

|  |
| --- |
| (SELECT \*  FROM UserActivity  GROUP BY username  HAVING COUNT(Activity) = 1)  UNION  (SELECT tbl1.\*  FROM UserActivity tbl1  INNER JOIN UserActivity tbl2  ON tbl1.username = tbl2.username AND tbl1.endDate < tbl2.endDate  GROUP BY tbl1.username, tbl1.endDate  HAVING COUNT(tbl2.endDate) = 1)  ORDER BY username; |

**Question 102. Write an SQL query to show the second most recent activity of each user. If the user only has one activity, return that one. A user cannot perform more than one activity at the same time. Return the result table in any order.**

*Duplicate*

**Question 103. Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.**

|  |
| --- |
| SELECT name  FROM students  WHERE marks > 75  ORDER BY RIGHT(name, 3), id ASC; |

**Question 104. Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.**

|  |
| --- |
| SELECT name  FROM employee  ORDER BY name; |

**Question 105. Write a query that prints a list of employee names (i.e.: the name attribute) for employees in 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.**

|  |
| --- |
| SELECT name  FROM employee  WHERE salary > 2000  AND months < 10  ORDER BY employee\_id; |

**Question 106. Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:**

* **Equilateral: It's a triangle with sides of equal length.**
* **Isosceles: It's a triangle with sides of equal length.**
* **Scalene: It's a triangle with sides of differing lengths.**
* **Not A Triangle: The given values of A, B, and C don't form a triangle.**

|  |
| --- |
| SELECT CASE  WHEN A + B <= C OR A + C <= B OR B + C <= A THEN 'Not A Triangle'  WHEN A = B AND B = C THEN 'Equilateral'  WHEN A = B OR B = C OR A = C THEN 'Isosceles'  ELSE 'Scalene'  END AS Triangle\_Type  FROM TRIANGLES; |

**Question 107. Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realise her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary. Write a query calculating the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.**

|  |
| --- |
| SELECT CEIL(AVG(salary) - AVG(REPLACE(salary, '0', '')))  FROM employees; |

**Question 108. We define an employee's total earnings to be their monthly salary \* 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. Then print these values as 2 space-separated integers.**

|  |
| --- |
| SELECT CONCAT(months\*salary,' ',COUNT(employee\_id)) AS Max\_Total\_Earning\_And\_No\_of\_Employees  FROM employee  GROUP BY months\*salary  ORDER BY months\*salary DESC  LIMIT 1; |

**Question 109. Generate the following two result sets:**

1. **Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).**
2. **Query the number of occurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format: Level - Medium There are a total of [occupation\_count] [occupation]s.where [occupation\_count] is the number of occurrences of an occupation in OCCUPATIONS and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation\_count], they should be ordered alphabetically**

|  |
| --- |
| -- 1  SELECT CONCAT(name, '(', SUBSTRING(occupation, 1, 1), ')') AS name\_occupation  FROM occupations  ORDER BY name\_occupation;  -- 2  SELECT CONCAT('There are a total of ', COUNT(occupation),' ', LOWER(occupation), (IF (COUNT(occupation) > 1, 's','')), '.') AS name\_occupation  FROM occupations  GROUP BY occupation  ORDER BY COUNT(occupation) ASC; |

**Question 110. Pivot the Occupation column in OCCUPATIONS so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively.**

**Note: Print NULL when there are no more names corresponding to an occupation.**

**Question 111. You are given a table, BST, containing two columns: N and P, where N represents the value of a node in Binary Tree, and P is the parent of N. Write a query to find the node type of Binary Tree ordered by the value of the node. Output one of the following for each node:**

* **Root: If node is root node.**
* **Leaf: If node is leaf node.**
* **Inner: If node is neither root nor leaf node.**

|  |
| --- |
| SELECT N,  CASE  WHEN N NOT IN (SELECT DISTINCT P FROM BST WHERE P IS NOT NULL) THEN 'Leaf'  WHEN P IS NULL THEN 'Root'  ELSE 'Inner'  END AS node\_type  FROM BST  ORDER BY N; |

**Question 112. Amber's conglomerate corporation just acquired some new companies.Each of the companies follows this hierarchy:**

**Given the table schemas below, write a query to print the company\_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company\_code.**

**Level – Medium**

**Note:**

* **The tables may contain duplicate records.**
* **The company\_code is string, so the sorting should not be numeric. For example, if the company\_codes are C\_1, C\_2, and C\_10, then the ascending company\_codes will be C\_1, C\_10, and C\_2.**