**Question 51. Each row of this table gives information about the name of a country, the continent to which it belongs, its area, the population, and its GDP value.**

**A country is big if:**

* **it has an area of at least three million (i.e., 3000000 km2), or**
* **it has a population of at least twenty-five million (i.e., 25000000).**

**Write an SQL query to report the name, population, and area of the big countries. Return the result table in any order**

|  |
| --- |
| SELECT name,  population,  area  FROM world  WHERE area >= 3000000  OR population >= 25000000; |

**Question 52. Write an SQL query to report the names of the customer that are not referred by the customer with id = 2. Return the result table in any order.**

|  |
| --- |
| SELECT name  FROM customer  WHERE referee\_id <> 2  OR referee\_id IS NULL; |

**Question 53. Write an SQL query to report all customers who never order anything. Return the result table in any order.**

|  |
| --- |
| SELECT name AS Customers  FROM customers  WHERE id NOT IN  (SELECT customerid  FROM orders); |

**Question 54. Write an SQL query to find the team size of each of the employees. Return result table in any order.**

|  |
| --- |
| SELECT e1.employee\_id,  COUNT(1) AS team\_size  FROM employee e1  LEFT JOIN employee e2  ON e1.team\_id = e2.team\_id  GROUP BY e1.employee\_id; |

**Question 55. A telecommunications company wants to invest in new countries. The company intends to invest in the countries where the average call duration of the calls in this country is strictly greater than the global average call duration.**

**Write an SQL query to find the countries where this company can invest. Return the result table in any order.**

**Question 56. Write an SQL query to report the device that is first logged in for each player. Return the result table in any order.** (Duplicate. Question 25, Set 1)

|  |
| --- |
| SELECT player\_id,  device\_id  FROM Activity  WHERE (player\_id, event\_date) IN  (select player\_id, MIN(event\_date)  FROM Activity  GROUP BY player\_id); |

**Question 57. Write an SQL query to find the customer\_number for the customer who has placed the largest number of orders.**

|  |
| --- |
| SELECT customer\_number  FROM orders  GROUP BY customer\_number  ORDER BY COUNT(order\_number) DESC  LIMIT 1; |

**Question 58. Write an SQL query to report all the consecutive available seats in the cinema. Return the result table ordered by seat\_id in ascending order.**

|  |
| --- |
| SELECT seat\_id  FROM  (SELECT current.seat\_id,  CASE WHEN EXISTS  (SELECT 1  FROM cinema previous  WHERE previous.seat\_id = current.seat\_id - 1  AND previous.free = 1)  AND current.free = 1 THEN current.seat\_id - 1  ELSE null  END previous,  CASE WHEN EXISTS  (SELECT 1  FROM cinema next  WHERE next.seat\_id = current.seat\_id + 1  AND next.free = 1)  AND current.free = 1 THEN current.seat\_id + 1  ELSE null  END next  FROM cinema current) seats  WHERE seats.previous IS NOT NULL  OR seats.next IS NOT NULL  ORDER BY seat\_id ASC; |

**Question 59. Write an SQL query to report the names of all the salespersons who did not have any orders related to the company with the name "RED". Return the result table in any order.**

|  |
| --- |
| SELECT s.name  FROM salesperson s  WHERE s.sales\_id NOT IN  (SELECT o.sales\_id  FROM orders o  LEFT JOIN company c  ON o.com\_id = c.com\_id  WHERE c.name = 'RED'); |

**Question 60. Write an SQL query to report for every three line segments whether they can form a triangle. Return the result table in any order.**

|  |
| --- |
| SELECT x,  y,  z,  CASE WHEN x + y <= z OR x + z <= y OR y + z <= x THEN 'No'  ELSE 'YES'  END AS triangle  FROM triangle; |

**Question 61. Write an SQL query to report the shortest distance between any two points from the Point table.**

|  |
| --- |
| SELECT MIN(ABS(p1.x - p2.x)) AS shortest  FROM point p1, point p2  WHERE p1.x <> p2.x; |

**Question 62. Write a SQL query for a report that provides the pairs (actor\_id, director\_id) where the actor has cooperated with the director at least three times. Return the result table in any order.**

|  |
| --- |
| SELECT actor\_id, director\_id  FROM ActorDirector  GROUP BY actor\_id, director\_id  HAVING COUNT(1) >= 3 |

**Question 63. Write an SQL query that reports the product\_name, year, and price for each sale\_id in the Sales table. Return the resulting table in any order.**

|  |
| --- |
| SELECT p.product\_name,  s.year,  s.price  FROM product p  INNER JOIN sales s  ON s.product\_id = p.product\_id; |

**Question 64. Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits. Return the result table in any order.**

|  |
| --- |
| SELECT project\_id,  ROUND(AVG(experience\_years), 2) AS average\_years  FROM project p  INNER JOIN employee e  ON p.employee\_id = e.employee\_id  GROUP BY project\_id; |

**Question 65. Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all. Return the result table in any order.**

|  |
| --- |
| SELECT seller\_id  FROM sales  GROUP BY seller\_id  HAVING SUM(price) = (  SELECT SUM(price)  FROM sales  GROUP BY seller\_id  ORDER BY SUM(price) DESC  LIMIT 1); |

**Question 66. Write an SQL query that reports the buyers who have bought S8 but not iPhone. Note that S8 and iPhone are products present in the Product table. Return the result table in any order.**

|  |
| --- |
| SELECT DISTINCT buyer\_id  FROM sales s  INNER JOIN product p  ON p.product\_id = s.product\_id  WHERE p.product\_name = 'S8'  AND buyer\_id NOT IN (  SELECT buyer\_id  FROM sales s  INNER JOIN product p  ON p.product\_id = s.product\_id  WHERE p.product\_name = 'iPhone'  ); |

**Question 67. Write an SQL query to compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average\_amount should be rounded to two decimal places. Return result table ordered by visited\_on in ascending order.**

|  |
| --- |
| SELECT t1.visited\_on,  SUM(t2.amount) AS amount,  ROUND(AVG(t2.amount), 2) AS average\_amount  FROM (  SELECT visited\_on, SUM(amount) AS amount  FROM customer  GROUP BY visited\_on  ) t1  INNER JOIN  (  SELECT visited\_on, SUM(amount) AS amount  FROM customer  GROUP BY visited\_on  ) t2  ON t2.visited\_on BETWEEN DATE\_SUB(t1.visited\_on, INTERVAL 6 DAY)  AND t1.visited\_on  GROUP BY t1.visited\_on  HAVING COUNT(1) = 7  ORDER BY t1.visited\_on; |

**Question 68. Write an SQL query to find the total score for each gender on each day. Return the result table ordered by gender and day in ascending order.**

|  |
| --- |
| SELECT s1.gender,  s1.day,  SUM(s2.score\_points) AS total  FROM scores s1, scores s2  WHERE s1.gender = s2.gender  AND s1.day >= s2.day  GROUP BY s1.gender, s1.day  ORDER BY s1.gender, s1.day; |

**Question 69. Write an SQL query to find the start and end number of continuous ranges in the table Logs. Return the result table ordered by start\_id.**

|  |
| --- |
| SELECT log\_start.log\_id AS start\_id,  MIN(log\_end.log\_id) AS end\_id  FROM  (  SELECT log\_id  FROM logs  WHERE log\_id - 1 NOT IN (  SELECT \*  FROM logs)  ) log\_start,  (  SELECT log\_id  FROM logs  WHERE log\_id + 1 NOT IN (  SELECT \*  FROM logs)  ) log\_end  WHERE log\_start.log\_id <= log\_end.log\_id  GROUP BY log\_start.log\_id  ORDER BY start\_id; |

**Question 70. Write an SQL query to find the number of times each student attended each exam. Return the result table ordered by student\_id and subject\_name.**

|  |
| --- |
| SELECT stud.student\_id,  stud.student\_name,  sub.subject\_name,  COUNT(exam.subject\_name) AS attended\_exams  FROM students stud  INNER JOIN subjects sub  LEFT JOIN examinations exam  ON stud.student\_id = exam.student\_id  AND sub.subject\_name = exam.subject\_name  GROUP BY stud.student\_id, sub.subject\_name  ORDER BY stud.student\_id, sub.subject\_name; |