1. **Q.1**

Table: Employee

+-------------+---------+

| Column Name | Type    |

+-------------+---------+

| empId       | int     |

| name        | varchar |

| supervisor  | int     |

| salary      | int     |

+-------------+---------+

empId is the primary key column for this table.

Each row of this table indicates the name and the ID of an employee in addition to their salary and the id of their manager.

Table: Bonus

+-------------+------+

| Column Name | Type |

+-------------+------+

| empId       | int  |

| bonus       | int  |

+-------------+------+

empId is the primary key column for this table.

empId is a foreign key to empId from the Employee table.

Each row of this table contains the id of an employee and their respective bonus.

Write an SQL query to report the name and bonus amount of each employee with a bonus **less than** 1000.

Return the result table in **any order**.

The query result format is in the following example.

Solution:

* SELECT name, bonus

FROM Employee

LEFT JOIN Bonus on Employee.empId=Bonus.empId

WHERE bonus <1000;

**Output:**

+------+-------+

| name | bonus |

+------+-------+

| Brad | null  |

| John | null  |

| Dan  | 500   |

+------+-------+

1. **Q.2**

**Table: Customer**

**+-------------+---------+**

**| Column Name | Type    |**

**+-------------+---------+**

**| id          | int     |**

**| name        | varchar |**

**| referee\_id  | int     |**

**+-------------+---------+**

**id is the primary key column for this table.**

**Each row of this table indicates the id of a customer, their name, and the id of the customer who referred them.**

**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.**

**The query result format is in the following example.**

* **SELECT name FROM Customer  
  WHERE referee\_id IS NULL OR referee\_id!=2 ;**

**Example 1:**

**Input:**

**Customer table:**

**+----+------+------------+**

**| id | name | referee\_id |**

**+----+------+------------+**

**| 1  | Will | null       |**

**| 2  | Jane | null       |**

**| 3  | Alex | 2          |**

**| 4  | Bill | null       |**

**| 5  | Zack | 1          |**

**| 6  | Mark | 2          |**

**+----+------+------------+**

**Output:**

**+------+**

**| name |**

**+------+**

**| Will |**

**| Jane |**

**| Bill |**

**| Zack |**

**+------+**

1. **Q.3**

**+-------------+---------+**

**| Column Name | Type    |**

**+-------------+---------+**

**| student     | varchar |**

**| class       | varchar |**

**+-------------+---------+**

**(student, class) is the primary key column for this table.**

**Each row of this table indicates the name of a student and the class in which they are enrolled.**

**Write an SQL query to report all the classes that have at least five students.**

**Return the result table in any order.**

**The query result format is in the following example.**

**🡪** **SELECT class FROM Student   
GROUP BY class  
WHERE COUNT (student)>=5;**

**Example 1:**

**Input:**

**Courses table:**

**+---------+----------+**

**| student | class    |**

**+---------+----------+**

**| A       | Math     |**

**| B       | English  |**

**| C       | Math     |**

**| D       | Biology  |**

**| E       | Math     |**

**| F       | Computer |**

**| G       | Math     |**

**| H       | Math     |**

**| I       | Math     |**

**+---------+----------+**

**Output:**

**+---------+**

**| class   |**

**+---------+**

**| Math    |**

**+---------+**

**Explanation:**

**- Math has 6 students, so we include it.**

**- English has 1 student, so we do not include it.**

**- Biology has 1 student, so we do not include it.**

**- Computer has 1 student, so we do not include it.**

1. **Qsn.4**

**Table: SalesPerson**

**+-----------------+---------+**

**| Column Name     | Type    |**

**+-----------------+---------+**

**| sales\_id        | int     |**

**| name            | varchar |**

**| salary          | int     |**

**| commission\_rate | int     |**

**| hire\_date       | date    |**

**+-----------------+---------+**

**sales\_id is the primary key column for this table.**

**Each row of this table indicates the name and the ID of a salesperson alongside their salary, commission rate, and hire date.**

**Table: Company**

**+-------------+---------+**

**| Column Name | Type    |**

**+-------------+---------+**

**| com\_id      | int     |**

**| name        | varchar |**

**| city        | varchar |**

**+-------------+---------+**

**com\_id is the primary key column for this table.**

**Each row of this table indicates the name and the ID of a company and the city in which the company is located.**

**Table: Orders**

**+-------------+------+**

**| Column Name | Type |**

**+-------------+------+**

**| order\_id    | int  |**

**| order\_date  | date |**

**| com\_id      | int  |**

**| sales\_id    | int  |**

**| amount      | int  |**

**+-------------+------+**

**order\_id is the primary key column for this table.**

**com\_id is a foreign key to com\_id from the Company table.**

**sales\_id is a foreign key to sales\_id from the SalesPerson table.**

**Each row of this table contains information about one order. This includes the ID of the company, the ID of the salesperson, the date of the order, and the amount paid.**

**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.**

**The query result format is in the following example.**

**🡪 SELECT name FROM SalesPerson  
LEFT JOIN (SELECT Order.sales\_id FROM Orders  
JOIN Company ON Orders.com\_id=Company.com\_id  
WHERE Company.name='RED')  AS t1 ON SalesPerson.sales\_id=t1.sales\_id  
WHERE t1.sales\_id is NULL;**

**Example 1:**

**Input:**

**SalesPerson table:**

**+----------+------+--------+-----------------+------------+**

**| sales\_id | name | salary | commission\_rate | hire\_date  |**

**+----------+------+--------+-----------------+------------+**

**| 1        | John | 100000 | 6               | 4/1/2006   |**

**| 2        | Amy  | 12000  | 5               | 5/1/2010   |**

**| 3        | Mark | 65000  | 12              | 12/25/2008 |**

**| 4        | Pam  | 25000  | 25              | 1/1/2005   |**

**| 5        | Alex | 5000   | 10              | 2/3/2007   |**

**+----------+------+--------+-----------------+------------+**

**Company table:**

**+--------+--------+----------+**

**| com\_id | name   | city     |**

**+--------+--------+----------+**

**| 1      | RED    | Boston   |**

**| 2      | ORANGE | New York |**

**| 3      | YELLOW | Boston   |**

**| 4      | GREEN  | Austin   |**

**+--------+--------+----------+**

**Orders table:**

**+----------+------------+--------+----------+--------+**

**| order\_id | order\_date | com\_id | sales\_id | amount |**

**+----------+------------+--------+----------+--------+**

**| 1        | 1/1/2014   | 3      | 4        | 10000  |**

**| 2        | 2/1/2014   | 4      | 5        | 5000   |**

**| 3        | 3/1/2014   | 1      | 1        | 50000  |**

**| 4        | 4/1/2014   | 1      | 4        | 25000  |**

**+----------+------------+--------+----------+--------+**

**Output:**

**+------+**

**| name |**

**+------+**

**| Amy  |**

**| Mark |**

**| Alex |**

1. **Qsn.5**

**Table: Cinema**

**+----------------+----------+**

**| Column Name    | Type     |**

**+----------------+----------+**

**| id             | int      |**

**| movie          | varchar  |**

**| description    | varchar  |**

**| rating         | float    |**

**+----------------+----------+**

**id is the primary key for this table.**

**Each row contains information about the name of a movie, its genre, and its rating.**

**rating is a 2 decimal places float in the range [0, 10]**

**Write an SQL query to report the movies with an odd-numbered ID and a description that is not "boring".**

**Return the result table ordered by rating in descending order.**

**The query result format is in the following example.**

**🡪SELECT movie**

**FROM Cinema**

**WHERE id % 2=1 ODD AND description! = ‘boring’**

**ORDER BY rating DESC**

**Example 1:**

**Input:**

**Cinema table:**

**+----+------------+-------------+--------+**

**| id | movie      | description | rating |**

**+----+------------+-------------+--------+**

**| 1  | War        | great 3D    | 8.9    |**

**| 2  | Science    | fiction     | 8.5    |**

**| 3  | irish      | boring      | 6.2    |**

**| 4  | Ice song   | Fantacy     | 8.6    |**

**| 5  | House card | Interesting | 9.1    |**

**+----+------------+-------------+--------+**

**Output:**

**+----+------------+-------------+--------+**

**| id | movie      | description | rating |**

**+----+------------+-------------+--------+**

**| 5  | House card | Interesting | 9.1    |**

**| 1  | War        | great 3D    | 8.9    |**

**+----+------------+-------------+--------+**

**Explanation:**

**We have three movies with odd-numbered IDs: 1, 3, and 5. The movie with ID = 3 is boring so we do not include it in the answer.**

1. **Qsn.6**

**ActorDirector**

**+-------------+---------+**

**| Column Name | Type    |**

**+-------------+---------+**

**| actor\_id    | int     |**

**| director\_id | int     |**

**| timestamp   | int     |**

**+-------------+---------+**

**timestamp is the primary key column for this table.**

**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.**

**The query result format is in the following example.**

* **SELECT actor\_id, director\_id**

**FROM ActorDirector**

**GROUP BY actor\_id, director\_id**

**HAVING COUNT (actor\_id= director\_id) >=3;**

**Example 1:**

**Input:**

**ActorDirector table:**

**+-------------+-------------+-------------+**

**| actor\_id    | director\_id | timestamp   |**

**+-------------+-------------+-------------+**

**| 1           | 1           | 0           |**

**| 1           | 1           | 1           |**

**| 1           | 1           | 2           |**

**| 1           | 2           | 3           |**

**| 1           | 2           | 4           |**

**| 2           | 1           | 5           |**

**| 2           | 1           | 6           |**

**+-------------+-------------+-------------+**

**Output:**

**+-------------+-------------+**

**| actor\_id    | director\_id |**

**+-------------+-------------+**

**| 1           | 1           |**

**+-------------+-------------+**

**Explanation: The only pair is (1, 1) where they cooperated exactly 3 times.**