Write a solution to find the number of times each student attended each exam.

Return the result table ordered by student\_id and subject\_name.

The result format is in the following example.

**Example 1:**

**Input:**

Students table:

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

| student\_id | student\_name |

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

| 1 | Alice |

| 2 | Bob |

| 13 | John |

| 6 | Alex |

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

Subjects table:

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

| subject\_name |

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

| Math |

| Physics |

| Programming |

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

Examinations table:

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

| student\_id | subject\_name |

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

| 1 | Math |

| 1 | Physics |

| 1 | Programming |

| 2 | Programming |

| 1 | Physics |

| 1 | Math |

| 13 | Math |

| 13 | Programming |

| 13 | Physics |

| 2 | Math |

| 1 | Math |

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

**Output:**

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

| student\_id | student\_name | subject\_name | attended\_exams |

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

| 1 | Alice | Math | 3 |

| 1 | Alice | Physics | 2 |

| 1 | Alice | Programming | 1 |

| 2 | Bob | Math | 1 |

| 2 | Bob | Physics | 0 |

| 2 | Bob | Programming | 1 |

| 6 | Alex | Math | 0 |

| 6 | Alex | Physics | 0 |

| 6 | Alex | Programming | 0 |

| 13 | John | Math | 1 |

| 13 | John | Physics | 1 |

| 13 | John | Programming | 1 |

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

**Explanation:**

The result table should contain all students and all subjects.

Alice attended the Math exam 3 times, the Physics exam 2 times, and the Programming exam 1 time.

Bob attended the Math exam 1 time, the Programming exam 1 time, and did not attend the Physics exam.

Alex did not attend any exams.

John attended the Math exam 1 time, the Physics exam 1 time, and the Programming exam 1 time.

select student\_id ,student\_name ,subject\_name ,attended\_exams

from  Students s join

table t on(s.student\_name =t.subject\_name ) join Examinations e

on(s.student\_id =e.student\_id )

group by student\_name,subject\_name ,student\_id

having count(\*)>1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1 | | | |  |  |  |  | Table 2 | | | | | |
| day | seller\_id | region | sales |  |  |  |  | seller\_id | seller\_name | | | start\_date | end\_date |
| 5/1/2023 | 101 | South | 20000 |  |  |  |  | 101 | ABC | | | 5/1/2023 | 6/30/2023 |
| 5/1/2023 | 102 | North | 12784 |  |  |  |  | 102 | DEF | | | 5/1/2023 | 5/30/2023 |
| 5/1/2023 | 103 | East | 93054 |  |  |  |  | 103 | GHI | | | 6/1/2023 | 6/30/2023 |
| 5/1/2023 | 104 | West | 23749 |  |  |  |  | 104 | JKL | | | 5/1/2023 | 6/30/2023 |
| 5/1/2023 | 105 | South | 12348 |  |  |  |  | 105 | KJI | | | 5/1/2023 | 5/30/2023 |
| 5/1/2023 | 106 | North | 9434 |  |  |  |  | 106 | MNB | | | 6/1/2023 | 6/30/2023 |
| 5/1/2023 | 107 | East | 38745 |  |  |  |  | 107 | JIO | | | 5/1/2023 | 6/30/2023 |
| 5/1/2023 | 108 | West | 12348 |  |  |  |  | 108 | KJL | | | 5/1/2023 | 5/30/2023 |
| 5/1/2023 | 109 | South | 9434 |  |  |  |  | 109 | POI | | | 6/1/2023 | 6/30/2023 |
| 5/2/2023 | 101 | South | 20000 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 102 | North | 12784 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 103 | East | 93054 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 104 | West | 23749 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 105 | South | 12348 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 106 | North | 9434 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 107 | East | 38745 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 108 | West | 12348 |  |  |  |  |  |  | | |  |  |
| 5/2/2023 | 109 | South | 9434 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 101 | South | 20000 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 102 | North | 12784 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 103 | East | 93054 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 104 | West | 23749 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 105 | South | 12348 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 106 | North | 9434 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 107 | East | 38745 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 108 | West | 12348 |  |  |  |  |  |  | | |  |  |
| 5/3/2023 | 109 | South | 9434 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 101 | South | 20000 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 102 | North | 12784 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 103 | East | 93054 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 104 | West | 23749 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 105 | South | 12348 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 106 | North | 9434 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 107 | East | 38745 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 108 | West | 12348 |  |  |  |  |  |  | | |  |  |
| 6/1/2023 | 109 | South | 9434 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 101 | South | 20000 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 102 | North | 12784 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 103 | East | 93054 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 104 | West | 23749 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 105 | South | 12348 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 106 | North | 9434 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 107 | East | 38745 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 108 | West | 12348 |  |  |  |  |  |  | | |  |  |
| 6/2/2023 | 109 | South | 9434 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 101 | South | 20000 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 102 | North | 12784 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 103 | East | 93054 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 104 | West | 23749 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 105 | South | 12348 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 106 | North | 9434 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 107 | East | 38745 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 108 | West | 12348 |  |  |  |  |  |  | | |  |  |
| 6/3/2023 | 109 | South | 9434 |  |  |  |  |  |  | | |  |  |
|  |  |  |  |  |  |  |  |  |  | | |  |  |
| Q1. Using Table1, Write a SQL Code to fetch Sum of Sales for every Region | | | | | | | | | | |
| Columns expected: Region, Sales | | | | | | | | | |  |
| SELECT region, SUM(sales) AS Sales  FROM Table1  GROUP BY region;   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q2. Using Table1, Write a SQL Code to fetch Top 3 Seller IDs based on Total Sales | | | | | | | | | | |
| Columns expected: Seller ID, Sales | | | | | | | | | |  |
| SELECT seller\_id, SUM(sales) AS Sales  FROM Table1  GROUP BY seller\_id  ORDER BY Sales DESC  LIMIT 3;   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q3. Using Table1, Write a SQL Code to fetch Top most Seller ID based on Sales in each Region | | | | | | | | | | |
| Columns expected: Region, Seller ID, Sales | | | | | | | | | |  |
| WITH RegionSales AS (  SELECT  region,  seller\_id,  SUM(sales) AS Sales  FROM Table1  GROUP BY region, seller\_id  ),  TopSeller AS (  SELECT  region,  seller\_id,  total\_sales,  RANK() OVER (PARTITION BY region ORDER BY Sales DESC) AS sales\_rank  FROM RegionSales  )  SELECT  region,  seller\_id,  Sales  FROM TopSeller  WHERE sales\_rank = 1;   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q4. Using Table1, Write a SQL Code to fetch month wise Total Sales | | | | | | | | | | |
| Columns expected: Month, Sales | | | | | | | | | |  |
| SELECT  DATE\_FORMAT(day, '%Y-%m') AS month,  SUM(sales) AS Sales  FROM  Table1  GROUP BY  DATE\_FORMAT(day, '%Y-%m')  ORDER BY  month;   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q5. Using Table2, Write a SQL Code to fetch Seller IDs which were active only during the month of May | | | | | | | | | | |
| Columns expected: Seller ID, Start date, End date | | | | | | | | | |  |
| SELECT  (seller\_id) AS Seller ID,  (start\_date) AS Start date ,  (end\_date) AS End date  FROM  Table2  WHERE  start\_date >= '2023-05-01' AND start\_date <= '2023-05-31'  AND end\_date >= '2023-05-01' AND end\_date <= '2023-05-31';   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q6. Using Table2, Write a SQL Code to fetch Seller IDs which were active only during the month of June | | | | | | | | | | |
| Columns expected: Seller ID, Start date, End date | | | | | | | | | |  |
| SELECT  (seller\_id) AS Seller ID,  (start\_date) AS Start date ,  (end\_date) AS End date  FROM  Table2  WHERE  start\_date >= '2023-06-01' AND start\_date <= '2023-06-30'  AND end\_date >= '2023-06-01' AND end\_date <= '2023-06-30';   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q7. Using both Table1 & Table2, Write a SQL Code to fetch Seller ID, Seller Name and sum of sales | | | | | | | | | | |
| Columns expected: Seller ID, Seller Name, Start date, End date, Sales | | | | | | | | | | |
| SELECT  (t2.seller\_id) AS Seller ID,  (t2.seller\_name) AS Seller Name,  (t2.start\_date) AS Start date,  (t2.end\_date) AS End date,  SUM(t1.sales) AS Sales  FROM  Table1 t1  JOIN  Table2 t2 ON t1.seller\_id = t2.seller\_id  GROUP BY  t2.seller\_id,  t2.seller\_name,  t2.start\_date,  t2.end\_date  ORDER BY  t2.seller\_id;   |  | | --- | |  | | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
| Q8. You are given two tables, Orders and Customers. The Orders table contains details about the orders placed by customers, including the order amount and the date of the order. The Customers table contains customer information.  Table: Orders   | OrderID | CustomerID | OrderDate | OrderAmount | | --- | --- | --- | --- | | 1 | 101 | 2024-01-05 | 500 | | 2 | 101 | 2024-02-10 | 200 | | 3 | 102 | 2024-01-15 | 300 | | 4 | 103 | 2024-03-01 | 700 | | 5 | 101 | 2024-03-05 | 100 | | 6 | 102 | 2024-02-20 | 150 | | 7 | 103 | 2024-03-10 | 250 |   Table: Customers   | CustomerID | CustomerName | Region | | --- | --- | --- | | 101 | Alice | North | | 102 | Bob | South | | 103 | Charlie | East |   Task:  Write an SQL query to achieve the following:   1. For each customer, retrieve the most recent three orders based on OrderDate. 2. Include the CustomerName, OrderID, OrderDate, and OrderAmount in the result. 3. The result should be ordered by CustomerName, and within each customer, the orders should be ordered by OrderDate in descending order. 4. If a customer has fewer than three orders, display all of their orders.   Expected Output:   | CustomerName | OrderID | OrderDate | OrderAmount | | --- | --- | --- | --- | | Alice | 5 | 2024-03-05 | 100 | | Alice | 2 | 2024-02-10 | 200 | | Alice | 1 | 2024-01-05 | 500 | | Bob | 6 | 2024-02-20 | 150 | | Bob | 3 | 2024-01-15 | 300 | | Charlie | 7 | 2024-03-10 | 250 | | Charlie | 4 | 2024-03-01 | 700 |   Write an SQL query to achieve the following:   1. For each customer, retrieve the most recent three orders based on OrderDate. 2. Include the CustomerName, OrderID, OrderDate, and OrderAmount in the result. 3. The result should be ordered by CustomerName, and within each customer, the orders should be ordered by OrderDate in descending order. 4. If a customer has fewer than three orders, display all of their orders.   Expected Output:   | CustomerName | OrderID | OrderDate | OrderAmount | | --- | --- | --- | --- | | Alice | 5 | 2024-03-05 | 100 | | Alice | 2 | 2024-02-10 | 200 | | Alice | 1 | 2024-01-05 | 500 | | Bob | 6 | 2024-02-20 | 150 | | Bob | 3 | 2024-01-15 | 300 | | Charlie | 7 | 2024-03-10 | 250 | | Charlie | 4 | 2024-03-01 | 700 |   WITH CTE as  (Select c.customerName,o.OrderId,o.OrderDate,o.orderAmount, dense\_RANK() OVER(PARTITION BY O.CustomerID order by o.OrderDate DESC ) as dnk  From Customers C join orders o on(C. CustomerID=o. CustomerID)  Group by c.customerName,o.OrderId,o.OrderDate,o.orderAmount  Order by OrderDate DESC)  Select  customerName, OrderId, OrderDate, orderAmount  from CTE where dnk<=3 |
|  | | | | | | | | | |  |
| [Q1] There are two SQL tables: authors and books.  Table : authors  The authors dataset has 1M+ rows; here’s the first six rows:  author\_name book\_name  author\_1 book\_1  author\_1 book\_2  author\_2 book\_3  author\_2 book\_4  author\_2 book\_5  author\_3 book\_6  … …  The books dataset also has 1M+ rows and here’s the first six:  Table : books  book\_name sold\_copies  book\_1 1000  book\_2 1500  book\_3 34000  book\_4 29000  book\_5 40000  book\_6 4400  … …  Create an SQL query that shows the TOP 3 authors who sold the most books in total!  Solution:  select author\_name, sum(sold\_copies) as total\_books\_sold  from authors a  join books b on a.book\_name = b.book\_name  group by author\_name order by total\_books\_sold desc limit 3  [Q2]  Table : users  user\_id username  1 John Doe  2 Jane Don  3 Alice Jones  4 Lisa Romero  Table: training\_details;  user\_training\_id user\_id training\_id training\_date  1 1 1 "2015-08-02"  2 2 1 "2015-08-03"  3 3 2 "2015-08-02"  4 4 2 "2015-08-04"  5 2 2 "2015-08-03"  6 1 1 "2015-08-02"  7 3 2 "2015-08-04"  8 4 3 "2015-08-03"  9 1 4 "2015-08-03"  10 3 1 "2015-08-02"  11 4 2 "2015-08-04"  12 3 2 "2015-08-02"  13 1 1 "2015-08-02"  14 4 3 "2015-08-03"  Write a query to get the list of names of users who took the same training lesson more than once in the same day  Expected Output:  username training\_id training\_date total  John Doe 1 "2015-08-02" 3  Alice Jones 2 "2015-08-02" 2  Lisa Romero 2 "2015-08-04" 2  Lisa Romero 3 "2015-08-03" 2  Solution:  select u.user\_name, t.training\_id, t.training\_date, count(t.training\_id) as total  from user u  join trainingdetails t on u.user\_id = t.user\_id  group by u.user\_name, t.training\_id, t.training\_date having count(t.training\_id) > 1;  SELECT  u.username,  td.training\_id,  td.training\_date,  COUNT(\*) AS total  FROM  users u  JOIN  training\_details td ON u.user\_id = td.user\_id  GROUP BY  u.username, td.training\_id, td.training\_date  HAVING  COUNT(\*) > 1;  ................................  Playback Table (table name - “streams”): streaming data of Prime Video customers. It contains information at a stream level, with data such as minutes streamed, buffers encountered while streaming, location, device used, and the customer ID.    playback\_date, stream\_id, device\_id, device\_category, customer\_id, minutes\_streamed, buffer\_count, customer\_country  Primary key - Stream\_id    playback\_date stream\_id device\_id device\_category customer\_id minutes\_streamed buffer\_count customer\_country  2017-01-01 11from2344 108 Television YYTGBCSDFG 12 0 US  2017-01-04 927612192 112 Television HIGDSGKJAF 34 10 UK  2017-04-03 812618726 179 Web Browser HJFGJLFFHJK 80 95 IN  2017-06-01 921629712 401 Web Browser TEEWRJGHKJKG 238 1 UK  2017-12-05 4982472376 105 Mobile KHGHJFDKLJHFF 10 5 US      Purchases table (table name - "transactions"): contains information on transactions conducted by customers on Prime Video.  purchase\_id purchase\_date customer\_id  716219621 2017-09-26 ASKGASKAJSA  918261982 2017-03-30 UKHJDLKGSDG  984673876 2017-12-16 UPFASFJSHDV  836453323 2018-09-24 AWETUYOIUBV  Q3.1. List the device category with at least 100,000 streams in the last 7 days?  select device\_category  from streams where DATEDIFF(current(date) , playback\_date) <= 7  group by device\_category having count(stream\_id ) > 10000  Q3.2. Find the number of customers who have streamed and purchased during the last 30 days?  select customer\_id  from streams where DATEDIFF(current(date) , playback\_date) <= 30  and customer\_id in(  select customer\_id from transactions where DATEDIFF(current(date) ,playback\_date) <=30 )  [Q4] Write a Query to print whether a player scored Highest Score in a particular match  Table: Match  Player Match\_id Series\_id Score  Dhoni 1 1 70  Kohli 1 1 50  Dhoni 2 1 10  Kohli 2 1 15  Dhoni 1 2 27  Kohli 1 2 45  Dhoni 2 2 69  Kohli 2 2 38  Expected Output:  Player Match\_id Series\_id HS/Not  Dhoni 1 1 Y  Dhoni 2 1 N  Dhoni 1 2 N  Dhoni 2 2 Y  Kohli 1 1 N  Kohli 2 1 Y  Kohli 1 2 Y  Kohli 2 2 N  with cte as  (select m.\*,  Max(Score) over(partition by Match\_id, Series\_id ) as max\_score  from match m)  select Player,Match\_id,Series\_id ,  case when max\_score > score then 'N'  else 'Y' end as HS/Not  from cte  ------SELECT  m.Player,  m.Match\_id,  m.Series\_id,  CASE  WHEN m.Score = max\_scores.max\_score THEN 'Y'  ELSE 'N'  END AS "HS/Not"  FROM  Match m  JOIN  (SELECT  Match\_id,  Series\_id,  MAX(Score) AS max\_score  FROM  Match  GROUP BY  Match\_id,  Series\_id) max\_scores  ON  m.Match\_id = max\_scores.Match\_id  AND m.Series\_id = max\_scores.Series\_id  ORDER BY  m.Player,  m.Match\_id,  m.Series\_id;  [Q5] Write a Query to print whether an item had increased discount % on consecutive days  Table: POS  Date Product Discount  01-Jan-20 A 0%  01-Jan-20 B 5%  01-Jan-20 C 10%  02-Jan-20 A 10%  02-Jan-20 B 5%  02-Jan-20 C 5%  03-Jan-20 A 15%  03-Jan-20 B 10%  03-Jan-20 C 10%  04-Jan-20 A 0%  04-Jan-20 B 15%  04-Jan-20 C 10%  Expected Output:  Date Product  02-Jan-20 A  03-Jan-20 A  03-Jan-20 B  03-Jan-20 C  04-Jan-20 B  --------------  with cte as(select p.\*,  lag(Discount) over(partition by product order by Date) as previous\_day\_discount  from POS p)  select Date, Product    from cte where discount::decimal > previous\_day\_discount::decimal  -----WITH PosWithRank AS (  SELECT  Date,  Product,  Discount,  LAG(Discount) OVER (PARTITION BY Product ORDER BY Date) AS Previous\_Discount  FROM  POS  )  SELECT  Date,  Product  FROM  PosWithRank  WHERE  Discount > Previous\_Discount  ORDER BY  Date,  Product; | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |
|  | | | | | | | | | |  |