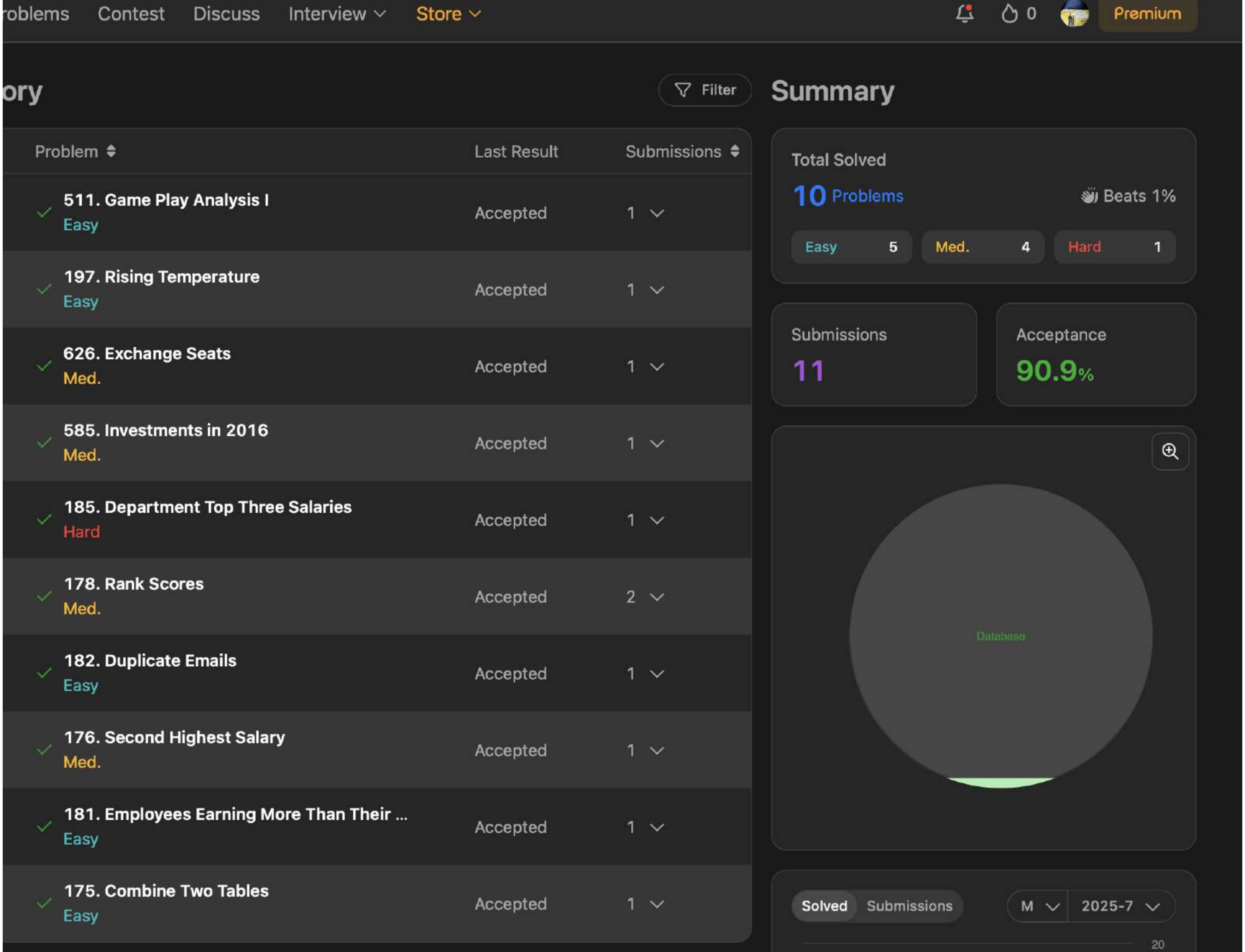


ory

Problem	Last Result	Submissions
✓ 511. Game Play Analysis I Easy	Accepted	1
✓ 197. Rising Temperature Easy	Accepted	1
✓ 626. Exchange Seats Med.	Accepted	1
✓ 585. Investments in 2016 Med.	Accepted	1
✓ 185. Department Top Three Salaries Hard	Accepted	1
✓ 178. Rank Scores Med.	Accepted	2
✓ 182. Duplicate Emails Easy	Accepted	1
✓ 176. Second Highest Salary Med.	Accepted	1
✓ 181. Employees Earning More Than Their ... Easy	Accepted	1
✓ 175. Combine Two Tables Easy	Accepted	1

Summary

Total Solved

10 Problems

Beats 1%

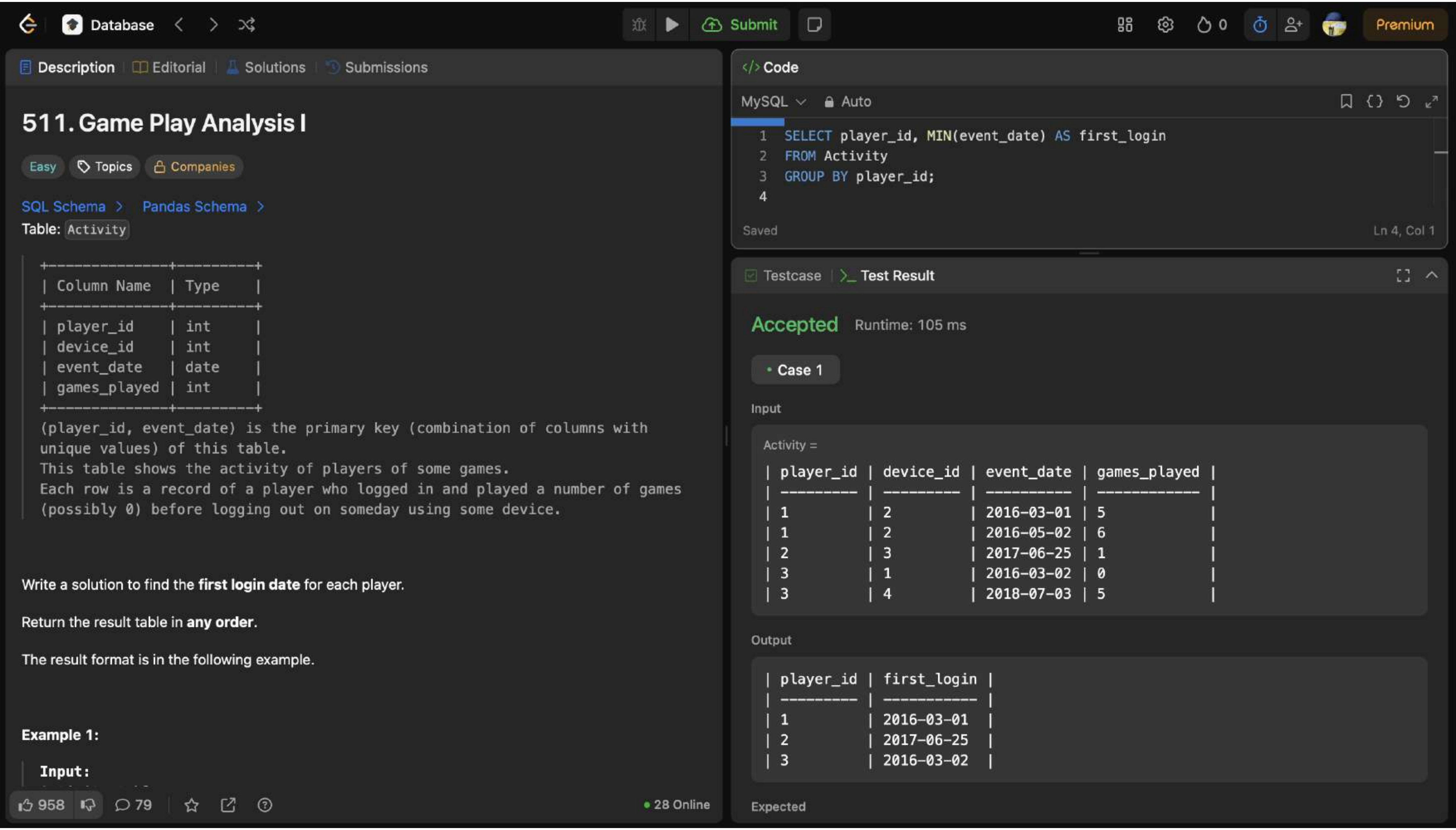
Easy	Med.	Hard
5	4	1

Submissions

11

Acceptance

90.9%



511. Game Play Analysis I

Easy · Topics · Companies

SQL Schema > Pandas Schema >

Table: `Activity`

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| player_id    | int     |
| device_id    | int     |
| event_date   | date    |
| games_played | int     |
+--------------+---------+
(player_id, event_date) is the primary key (combination of columns with
unique values) of this table.
This table shows the activity of players of some games.
Each row is a record of a player who logged in and played a number of games
(possibly 0) before logging out on someday using some device.
```

Write a solution to find the **first login date** for each player.

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

The result format is in the following example.

Example 1:

Input:

```sql
SELECT player_id, MIN(event_date) AS first_login
FROM Activity
GROUP BY player_id;
```

Accepted Runtime: 105 ms

Case 1

Input

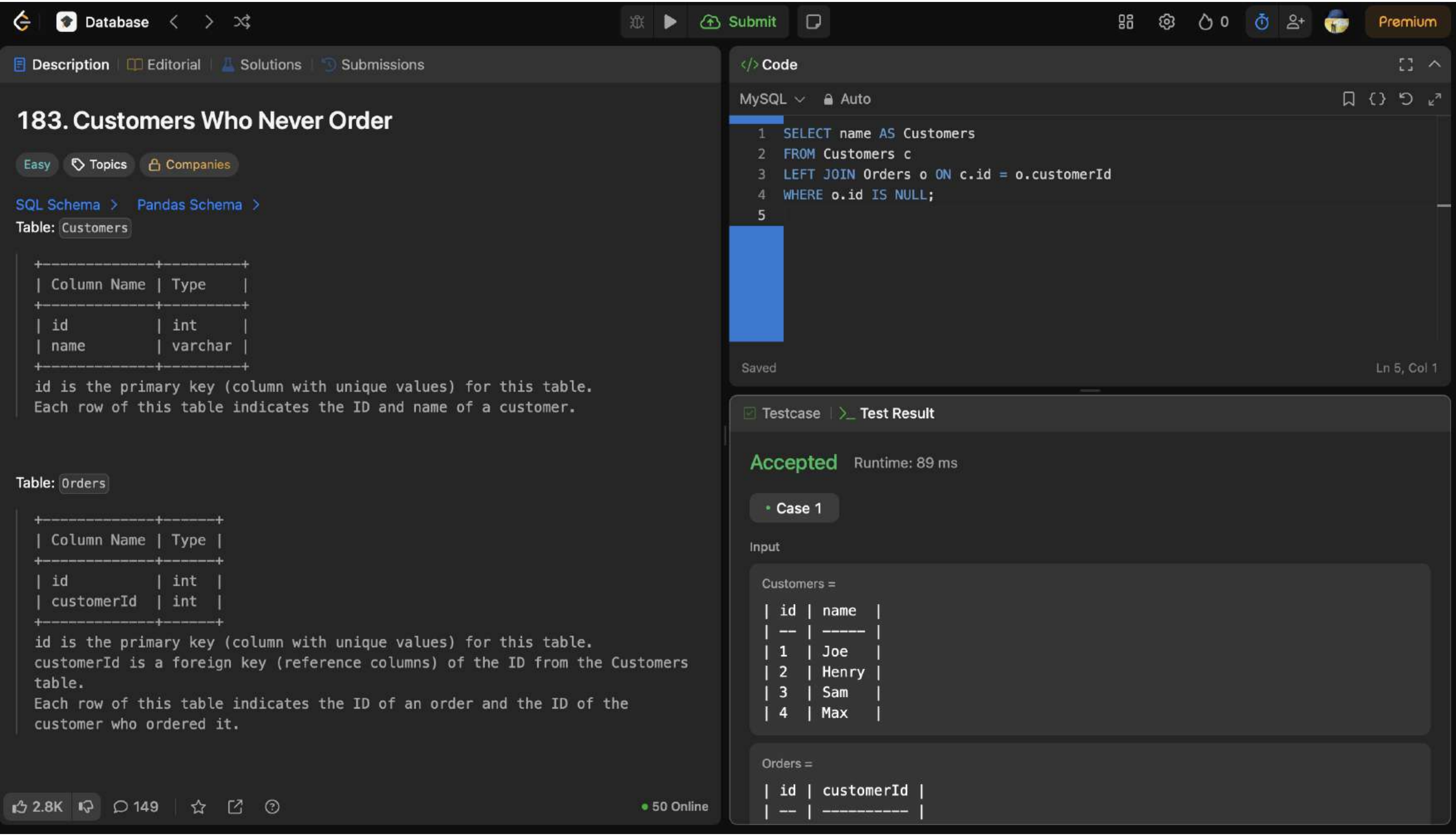
```
Activity =

| player_id | device_id | event_date | games_played |
| --------- | --------- | ---------- | ------------ |
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-05-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
```

Output

```
| player_id | first_login |
| --------- | ----------- |
| 1         | 2016-03-01  |
| 2         | 2017-06-25  |
| 3         | 2016-03-02  |
```

Expected



183. Customers Who Never Order

Table: `Customers`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table indicates the ID and name of a customer.
```

Table: `Orders`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| customerId  | int  |
+-------------+------+
id is the primary key (column with unique values) for this table.
customerId is a foreign key (reference columns) of the ID from the Customers
table.
Each row of this table indicates the ID of an order and the ID of the
customer who ordered it.
```

```sql
SELECT name AS Customers
FROM Customers c
LEFT JOIN Orders o ON c.id = o.customerId
WHERE o.id IS NULL;
```

Accepted Runtime: 89 ms

Case 1

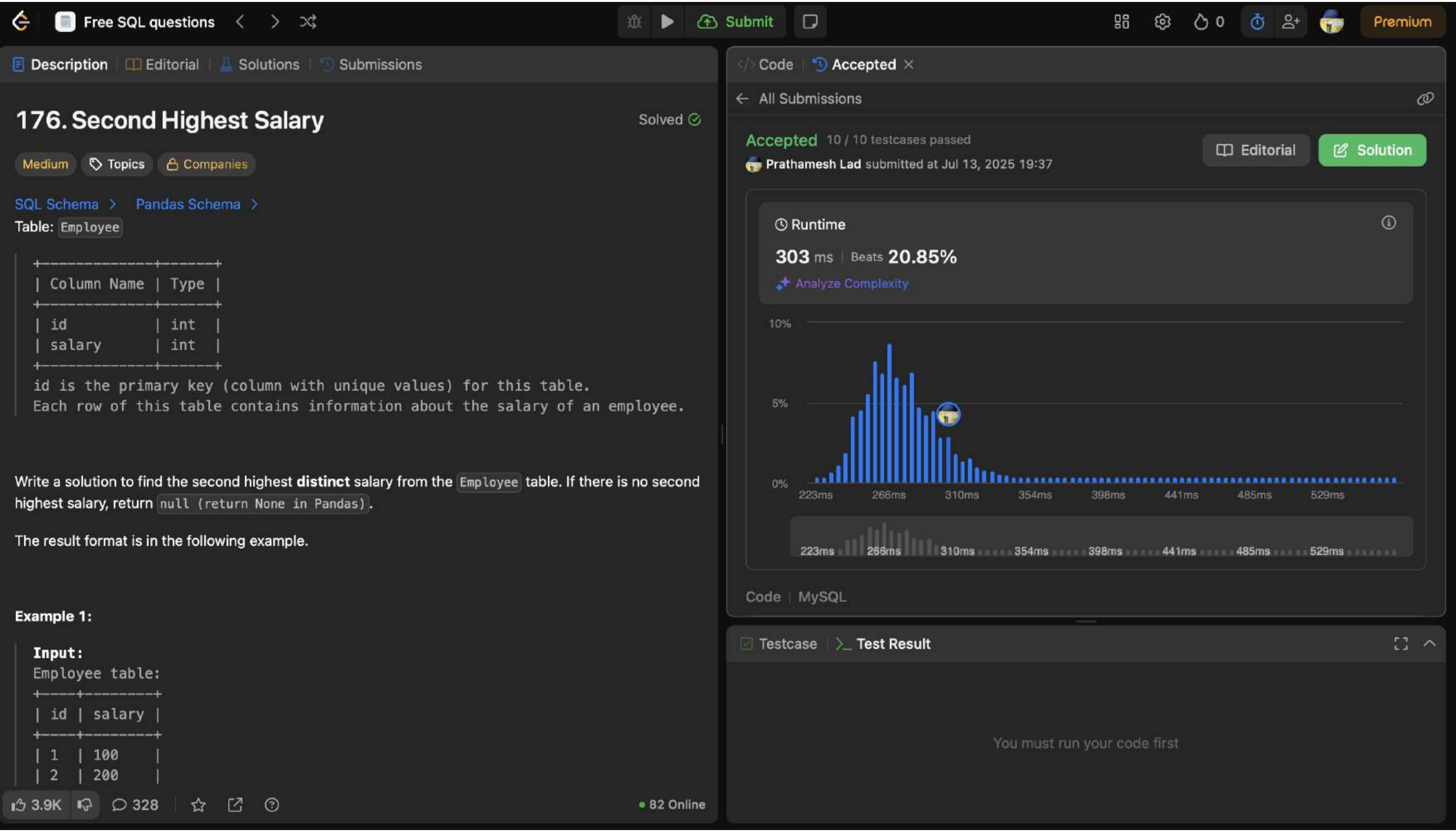
Input

Customers =

```
| id | name  |
| -- | ----- |
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
```

Orders =

```
| id | customerId |
| -- | ---------- |
```



176. Second Highest Salary

Solved ✓

Medium | Topics | Companies

SQL Schema > Pandas Schema >

Table: `Employee`

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| salary      | int  |
+-------------+------+
id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an employee.
```

Write a solution to find the second highest **distinct** salary from the `Employee` table. If there is no second highest salary, return `null (return None in Pandas)`.

The result format is in the following example.

Example 1:

```
Input:
Employee table:
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
```

Accepted 10 / 10 testcases passed

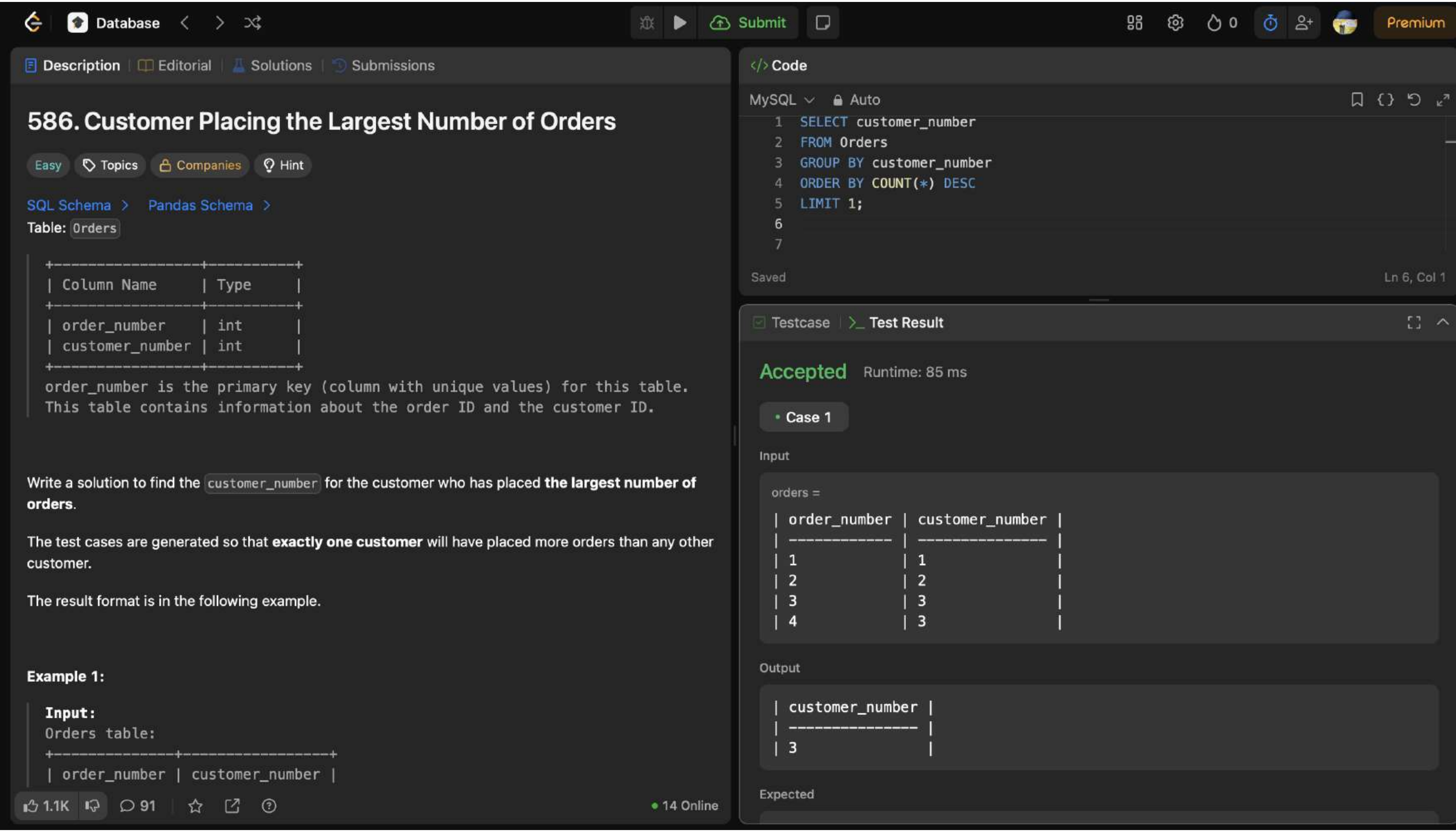
Prathamesh Lad submitted at Jul 13, 2025 19:37

Runtime

303 ms | Beats 20.85%

Code | MySQL

You must run your code first



586. Customer Placing the Largest Number of Orders

SQL Schema > Pandas Schema >

Table: `Orders`

```
+-----------------+----------+
| Column Name     | Type     |
+-----------------+----------+
| order_number    | int      |
| customer_number | int      |
+-----------------+----------+
order_number is the primary key (column with unique values) for this table.
This table contains information about the order ID and the customer ID.
```

Write a solution to find the `customer_number` for the customer who has placed **the largest number of orders**.

The test cases are generated so that **exactly one customer** will have placed more orders than any other customer.

The result format is in the following example.

Example 1:

```
Input:
Orders table:
+--------------+-----------------+
| order_number | customer_number |
```

```sql
SELECT customer_number
FROM Orders
GROUP BY customer_number
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Accepted Runtime: 85 ms

Case 1

Input

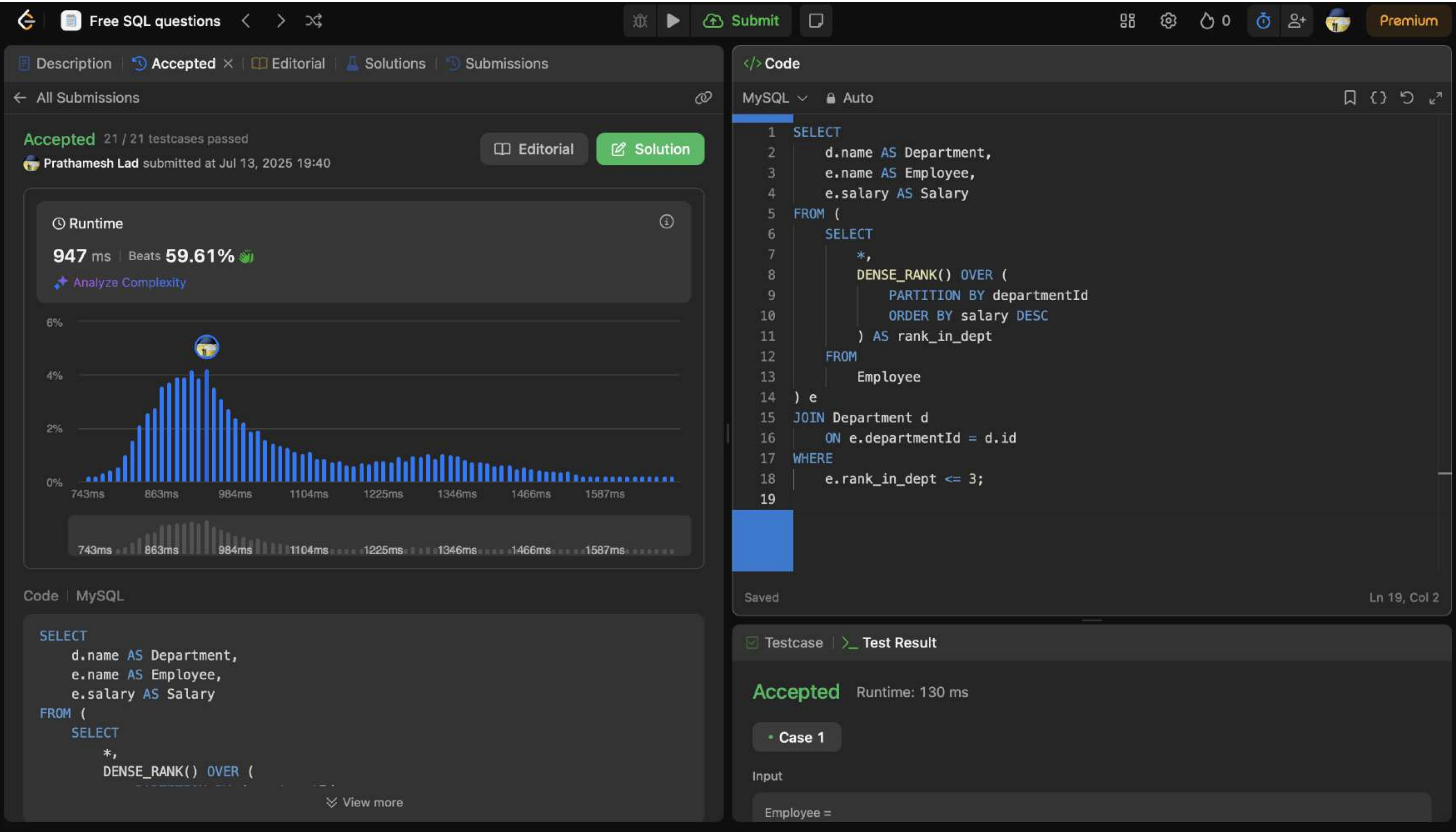
orders =

```
| order_number | customer_number |
| ------------ | --------------- |
| 1            | 1               |
| 2            | 2               |
| 3            | 3               |
| 4            | 3               |
```

Output

```
| customer_number |
| --------------- |
| 3               |
```

Expected



Accepted 21 / 21 testcases passed

Prathamesh Lad submitted at Jul 13, 2025 19:40

947 ms | Beats 59.61%

Code | MySQL

```
SELECT
    d.name AS Department,
    e.name AS Employee,
    e.salary AS Salary
FROM (
    SELECT
        *,
        DENSE_RANK() OVER (
```

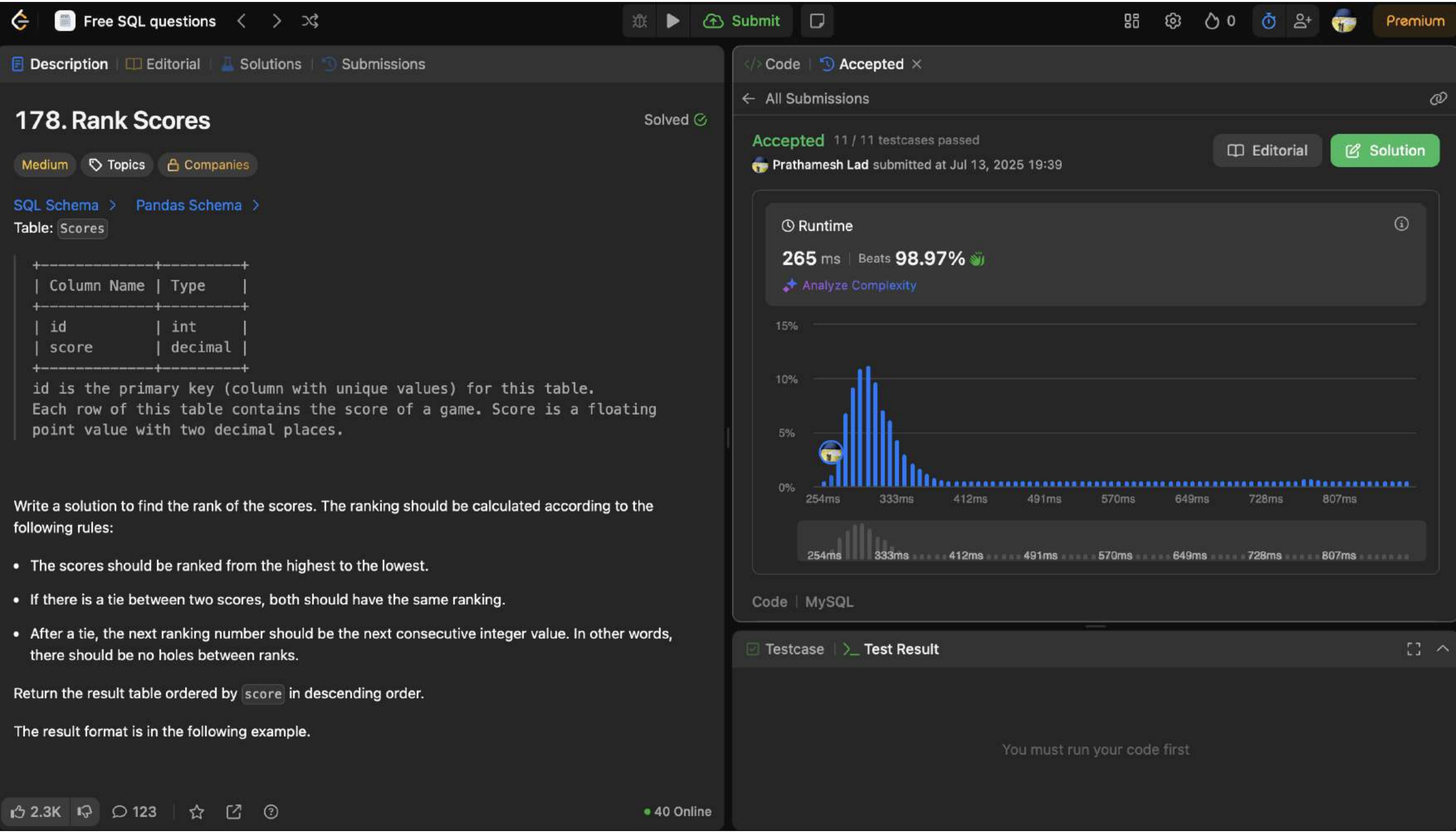
```
SELECT
    d.name AS Department,
    e.name AS Employee,
    e.salary AS Salary
FROM (
    SELECT
        *,
        DENSE_RANK() OVER (
            PARTITION BY departmentId
            ORDER BY salary DESC
        ) AS rank_in_dept
    FROM
        Employee
) e
JOIN Department d
    ON e.departmentId = d.id
WHERE
    e.rank_in_dept <= 3;
```

Accepted Runtime: 130 ms

• Case 1

Input

Employee =



178. Rank Scores

Solved

Medium · Topics · Companies

SQL Schema > Pandas Schema >

Table: `Scores`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| score       | decimal |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table contains the score of a game. Score is a floating
point value with two decimal places.
```

Write a solution to find the rank of the scores. The ranking should be calculated according to the following rules:

- The scores should be ranked from the highest to the lowest.
- If there is a tie between two scores, both should have the same ranking.
- After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by `score` in descending order.

The result format is in the following example.

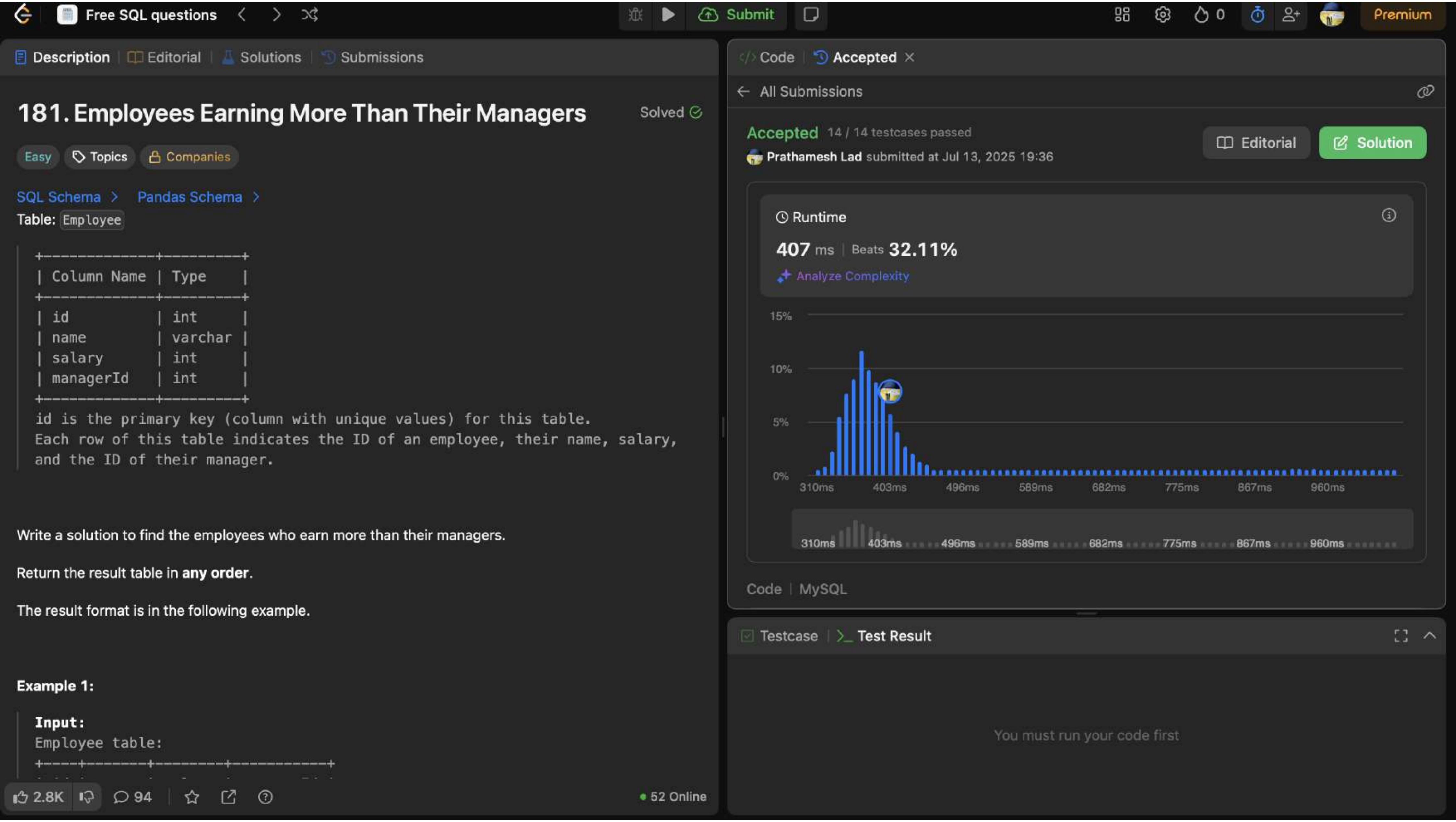
Accepted 11 / 11 testcases passed

Prathamesh Lad submitted at Jul 13, 2025 19:39

Runtime

265 ms | Beats 98.97%

You must run your code first



181. Employees Earning More Than Their Managers

Solved

Easy Topics Companies

SQL Schema > Pandas Schema >

Table: `Employee`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| managerId   | int     |
+-------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table indicates the ID of an employee, their name, salary,
and the ID of their manager.
```

Write a solution to find the employees who earn more than their managers.

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

The result format is in the following example.

Example 1:

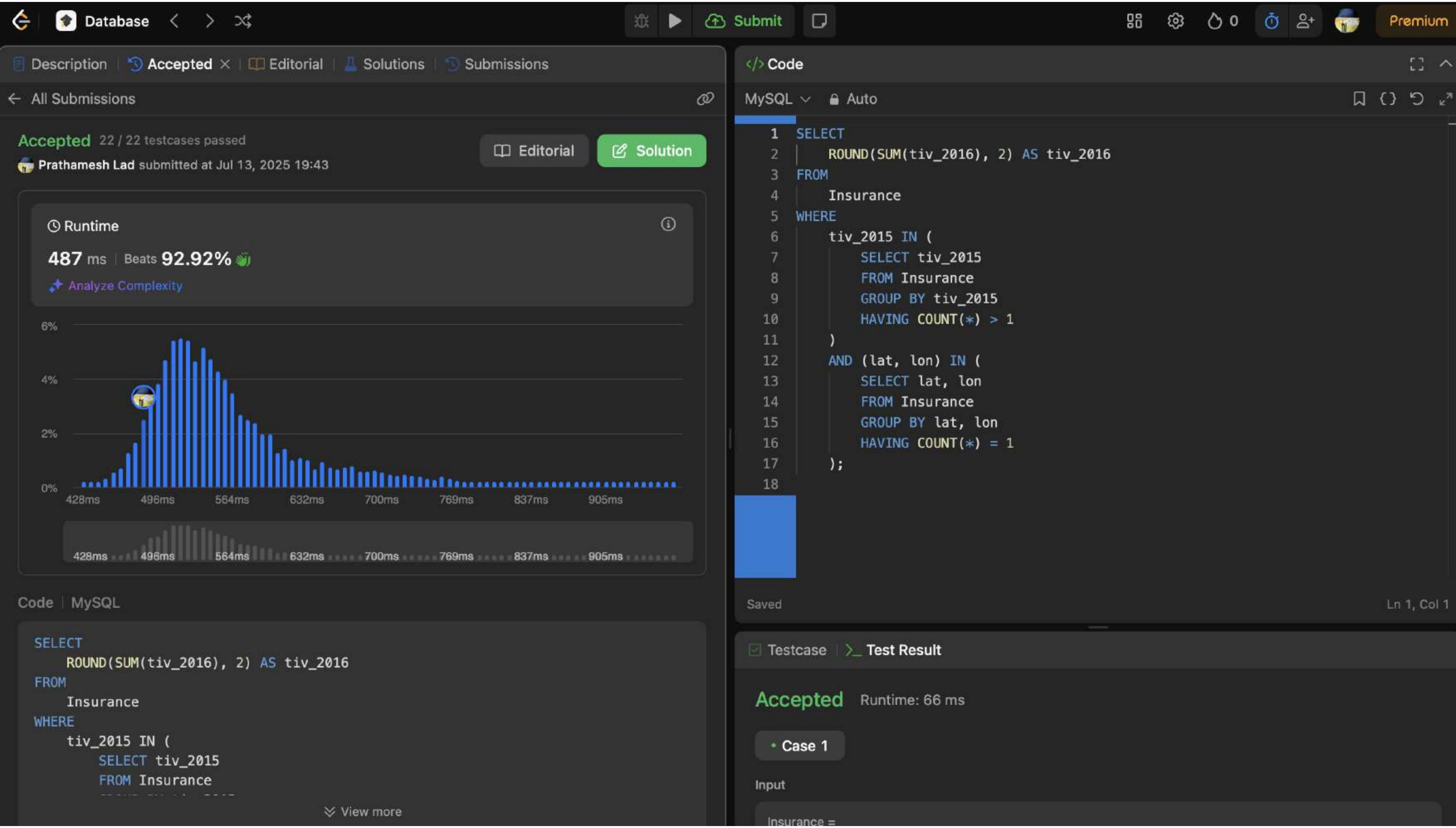
```
Input:
Employee table:
+----+-------+--------+-----------+
```

Accepted 14 / 14 testcases passed

Prathamesh Lad submitted at Jul 13, 2025 19:36

407 ms | Beats 32.11%

You must run your code first



Accepted 22 / 22 testcases passed

Prathamesh Lad submitted at Jul 13, 2025 19:43

Runtime

487 ms | Beats 92.92%

Code | MySQL

```
SELECT
    ROUND(SUM(tiv_2016), 2) AS tiv_2016
FROM
    Insurance
WHERE
    tiv_2015 IN (
        SELECT tiv_2015
        FROM Insurance
```

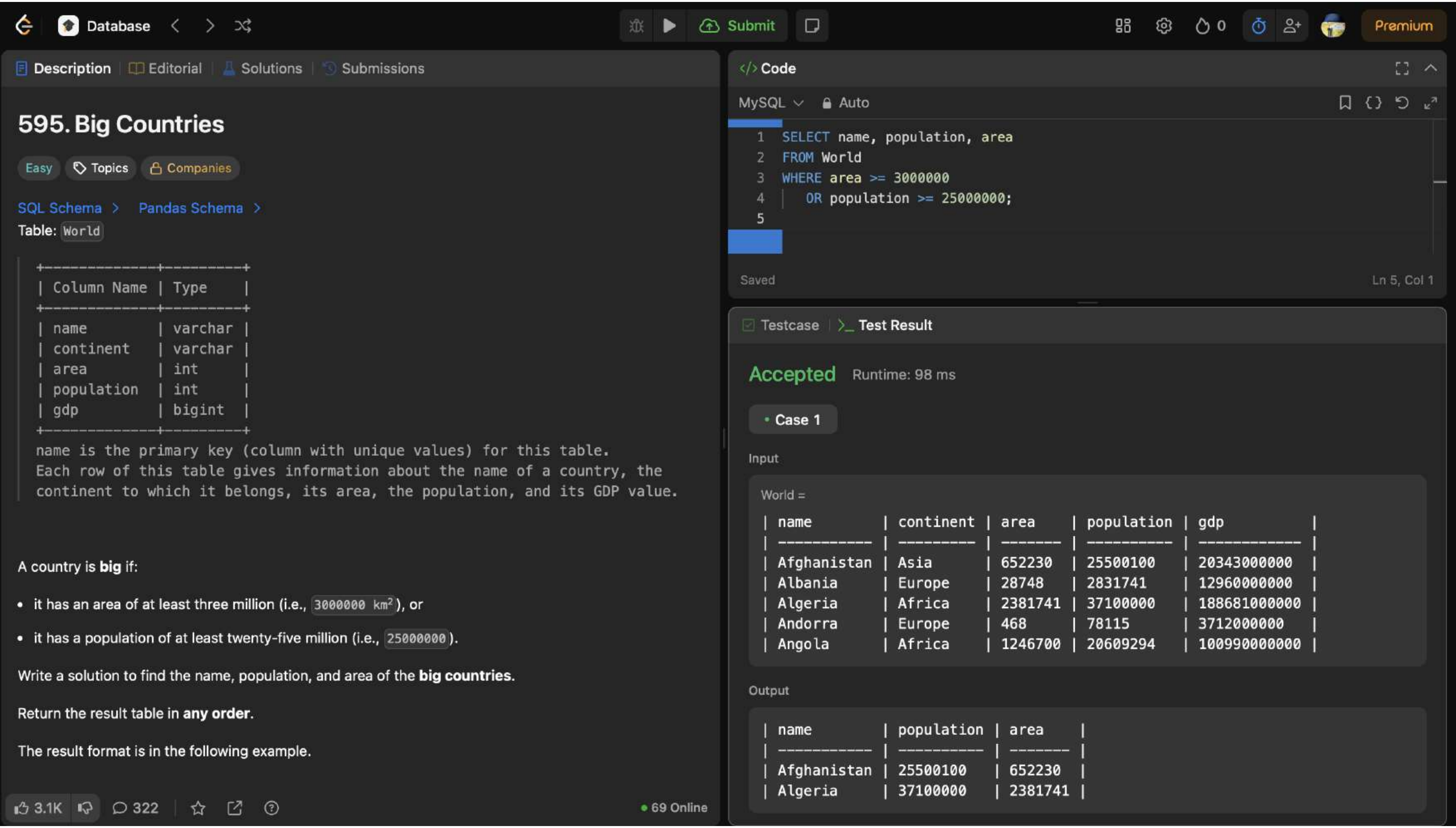
```
SELECT
    ROUND(SUM(tiv_2016), 2) AS tiv_2016
FROM
    Insurance
WHERE
    tiv_2015 IN (
        SELECT tiv_2015
        FROM Insurance
        GROUP BY tiv_2015
        HAVING COUNT(*) > 1
    )
    AND (lat, lon) IN (
        SELECT lat, lon
        FROM Insurance
        GROUP BY lat, lon
        HAVING COUNT(*) = 1
    );
```

Accepted Runtime: 66 ms

Case 1

Input

Insurance =



595. Big Countries

SQL Schema > Pandas Schema >

Table: `World`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| name        | varchar |
| continent   | varchar |
| area        | int     |
| population  | int     |
| gdp         | bigint  |
+-------------+---------+
name is the primary key (column with unique values) for this table.
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 a solution to find the name, population, and area of the **big countries**.

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

The result format is in the following example.

```sql
SELECT name, population, area
FROM World
WHERE area >= 3000000
   OR population >= 25000000;

```

Saved

Ln 5, Col 1

Accepted Runtime: 98 ms

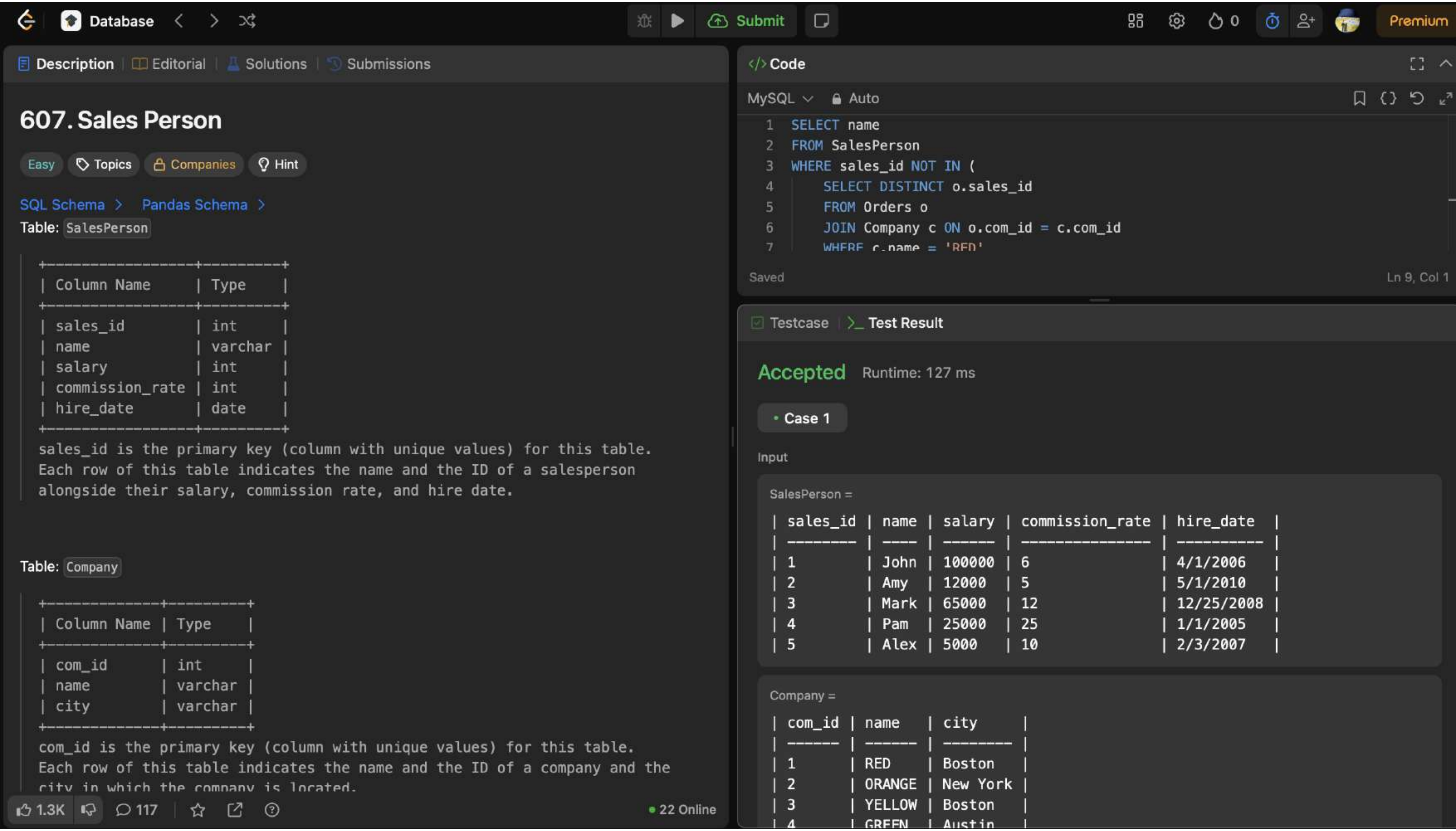
Case 1

Input

```
World =
| name        | continent | area    | population | gdp          |
| ----------- | --------- | ------- | ---------- | ------------ |
| Afghanistan | Asia      | 652230  | 25500100   | 20343000000  |
| Albania     | Europe    | 28748   | 2831741    | 12960000000  |
| Algeria     | Africa    | 2381741 | 37100000   | 188681000000 |
| Andorra     | Europe    | 468     | 78115      | 3712000000   |
| Angola      | Africa    | 1246700 | 20609294   | 100990000000 |
```

Output

```
| name        | population | area    |
| ----------- | ---------- | ------- |
| Afghanistan | 25500100   | 652230  |
| Algeria     | 37100000   | 2381741 |
```



607. Sales Person

SQL Schema > Pandas Schema >

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 with unique values) 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 with unique values) 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.
```

```sql
SELECT name
FROM SalesPerson
WHERE sales_id NOT IN (
    SELECT DISTINCT o.sales_id
    FROM Orders o
    JOIN Company c ON o.com_id = c.com_id
    WHERE c.name = 'RED'
```

Accepted Runtime: 127 ms

Case 1

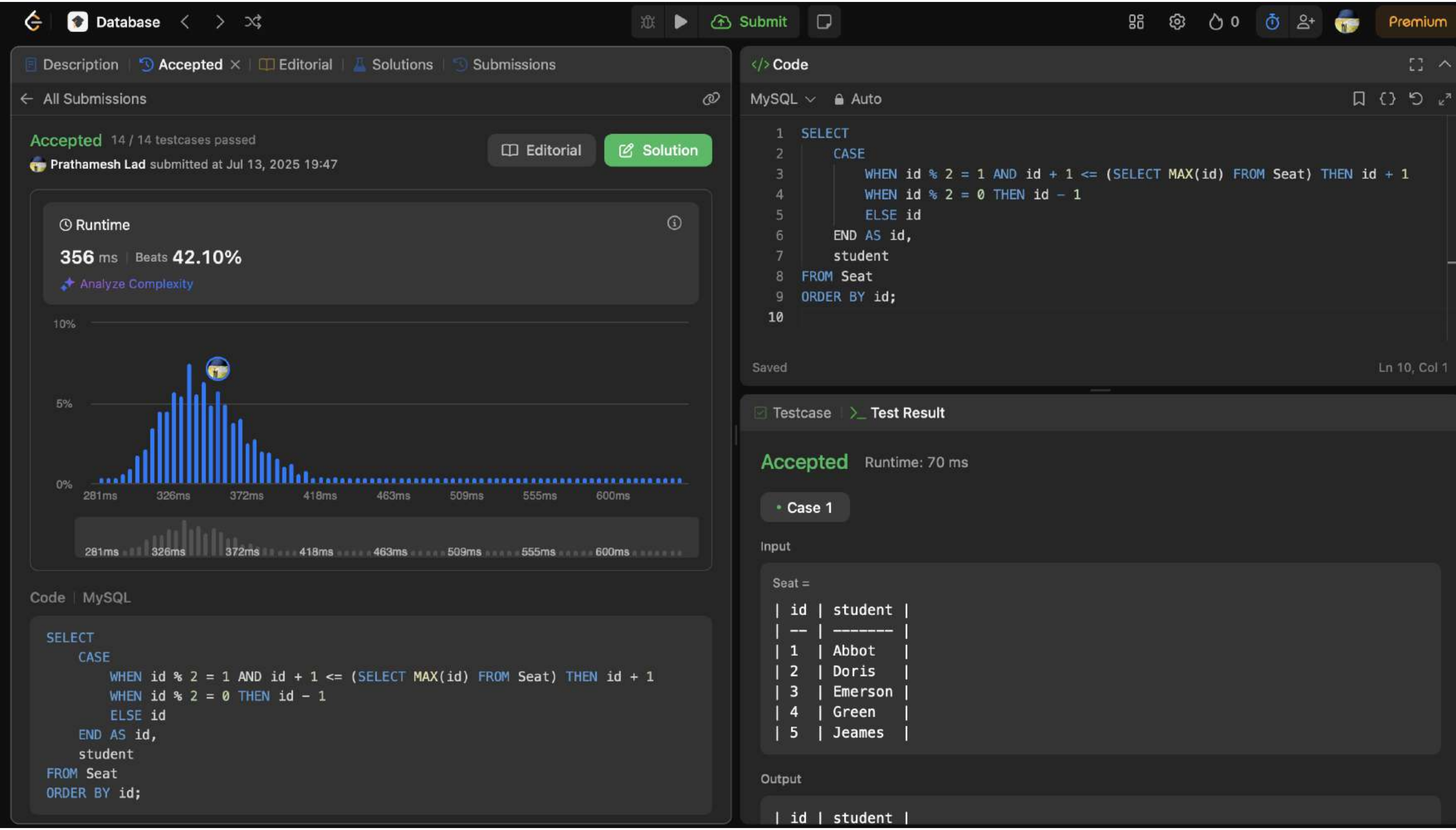
Input

SalesPerson =

```
| 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 =

```
| com_id | name   | city     |
| ------ | ------ | -------- |
| 1      | RED    | Boston   |
| 2      | ORANGE | New York |
| 3      | YELLOW | Boston   |
| 4      | GREEN  | Austin   |
```



Accepted 14 / 14 testcases passed

Prathamesh Lad submitted at Jul 13, 2025 19:47

356 ms | Beats 42.10%

Code | MySQL

```sql
SELECT
    CASE
        WHEN id % 2 = 1 AND id + 1 <= (SELECT MAX(id) FROM Seat) THEN id + 1
        WHEN id % 2 = 0 THEN id - 1
        ELSE id
    END AS id,
    student
FROM Seat
ORDER BY id;
```

Accepted Runtime: 70 ms

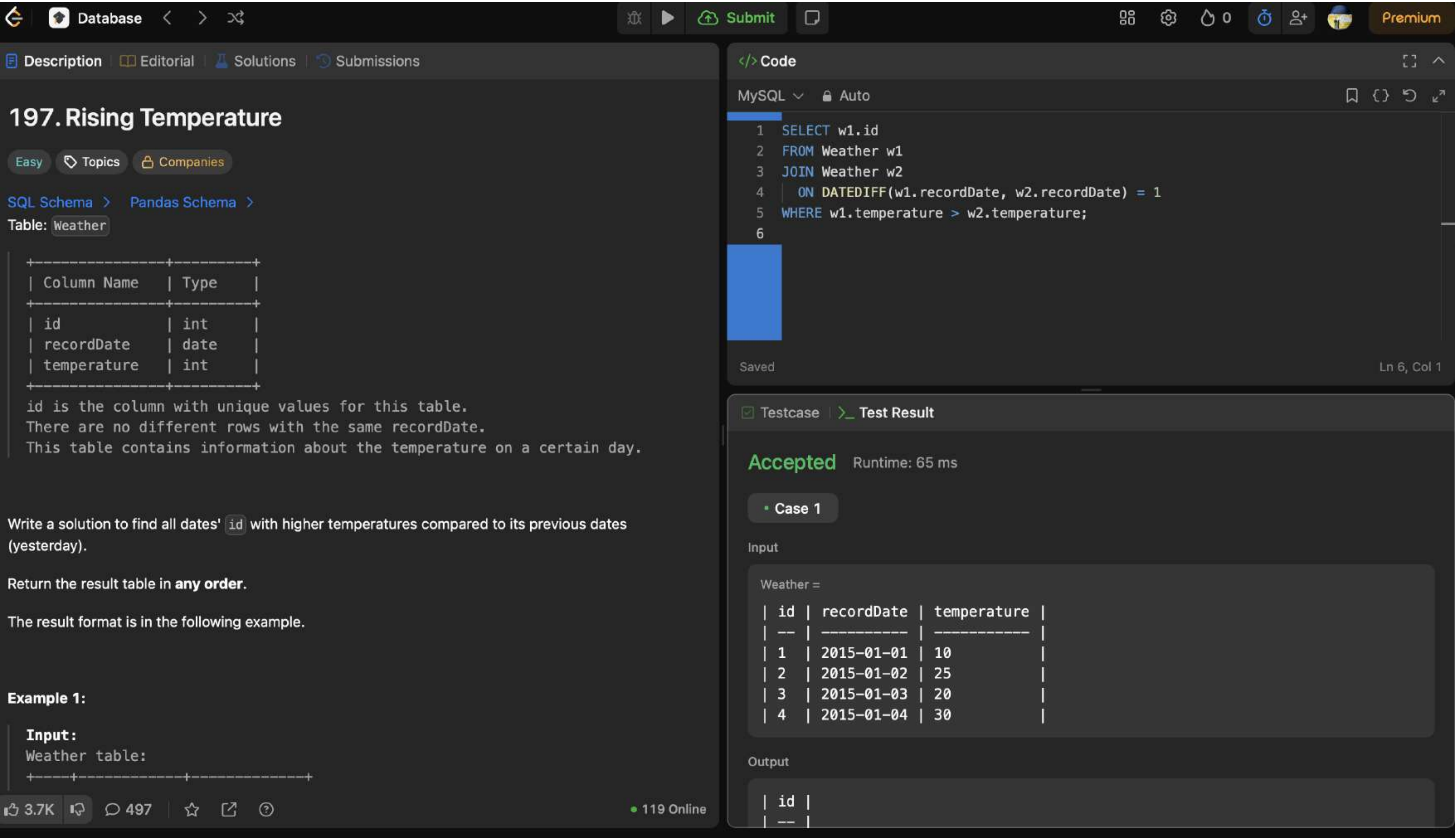
Case 1

Input

```
Seat =
| id | student |
| -- | ------- |
| 1  | Abbot   |
| 2  | Doris   |
| 3  | Emerson |
| 4  | Green   |
| 5  | Jeames  |
```

Output

```
| id | student |
```



197. Rising Temperature

Table: `Weather`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| recordDate    | date    |
| temperature   | int     |
+---------------+---------+
id is the column with unique values for this table.
There are no different rows with the same recordDate.
This table contains information about the temperature on a certain day.
```

Write a solution to find all dates' `id` with higher temperatures compared to its previous dates (yesterday).

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

The result format is in the following example.

Example 1:

```
Input:
Weather table:
+-----+-------------+-------------+
```

```sql
SELECT w1.id
FROM Weather w1
JOIN Weather w2
  ON DATEDIFF(w1.recordDate, w2.recordDate) = 1
WHERE w1.temperature > w2.temperature;
```

Accepted Runtime: 65 ms

Input

```
Weather =
| id | recordDate | temperature |
| -- | ---------- | ----------- |
| 1  | 2015-01-01 | 10          |
| 2  | 2015-01-02 | 25          |
| 3  | 2015-01-03 | 20          |
| 4  | 2015-01-04 | 30          |
```

Output

```
| id |
| -- |
```