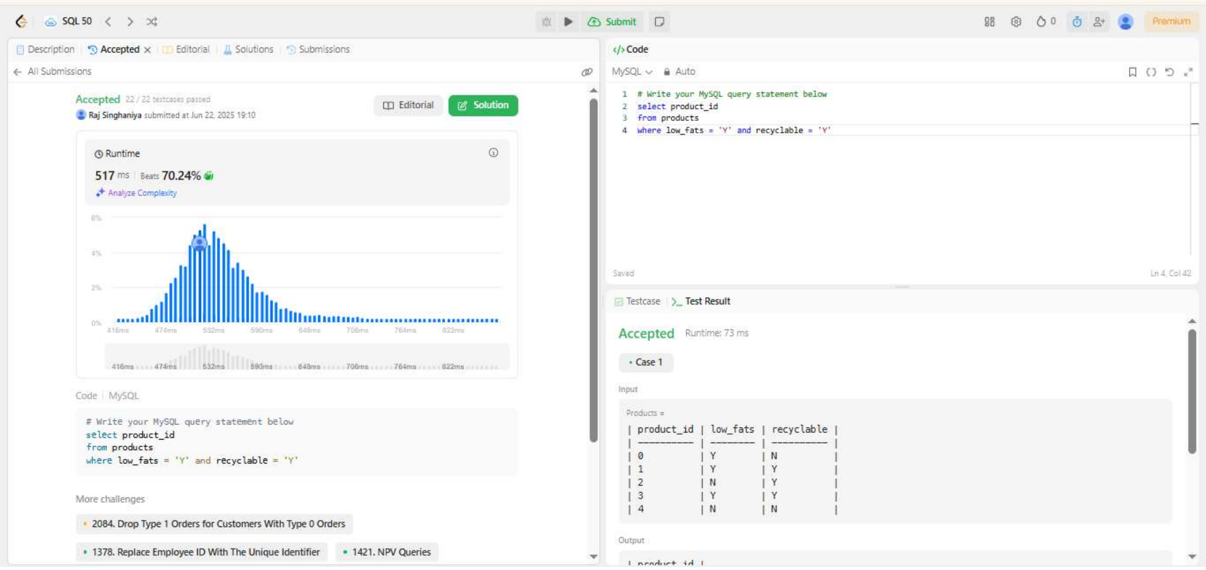
0112cs221101@gmail.com
Raj Singhaniya

CTE ASSIGNMENT

Subject Change Request Problem

```
CREATE PROCEDURE sp_UpdateSubjectAllotment
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @StudentID VARCHAR(50)
    DECLARE @RequestedSubjectID VARCHAR(50)
    DECLARE @CurrentSubjectID VARCHAR(50)

    DECLARE request_cursor CURSOR FOR
        SELECT StudentId, SubjectId FROM SubjectRequest;

    OPEN request_cursor;
    FETCH NEXT FROM request_cursor INTO @StudentID, @RequestedSubjectID;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        SELECT @CurrentSubjectID = SubjectId
        FROM SubjectAllotments
        WHERE StudentId = @StudentID AND Is_Valid = 1;

        IF @CurrentSubjectID IS NULL
        BEGIN
            INSERT INTO SubjectAllotments (StudentId, SubjectId, Is_Valid)
            VALUES (@StudentID, @RequestedSubjectID, 1);
        END
        ELSE
        BEGIN

            IF @RequestedSubjectID <> @CurrentSubjectID
            BEGIN
                UPDATE SubjectAllotments
                SET Is_Valid = 0
                WHERE StudentId = @StudentID AND Is_Valid = 1;

                INSERT INTO SubjectAllotments (StudentId, SubjectId, Is_Valid)
                VALUES (@StudentID, @RequestedSubjectID, 1);
            END
        END
        FETCH NEXT FROM request_cursor INTO @StudentID, @RequestedSubjectID;
    END

    CLOSE request_cursor;
    DEALLOCATE request_cursor;
    DELETE FROM SubjectRequest;
END
```

LEETCODE



Accepted 22 / 22 testcases passed

Raj Singhaniya submitted at Jun 22, 2025 19:10

Runtime

517 ms | Beats 70.24%

Analyze Complexity

Code | MySQL

```
# Write your MySQL query statement below
select product_id
from products
where low_fats = 'Y' and recyclable = 'Y'
```

More challenges

- 2084. Drop Type 1 Orders for Customers With Type 0 Orders
- 1378. Replace Employee ID With The Unique Identifier
- 1421. NPV Queries

Accepted Runtime: 73 ms

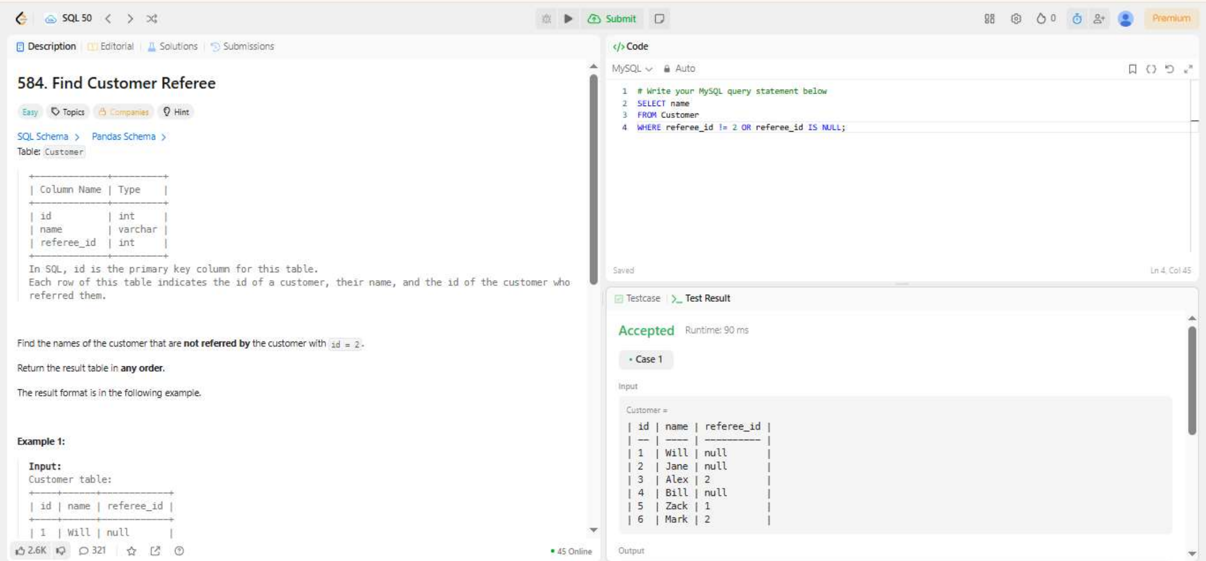
- Case 1

Input

```
Products =
| product_id | low_fats | recyclable |
| ---------- | -------- | ---------- |
| 0          | Y        | N          |
| 1          | Y        | Y          |
| 2          | N        | Y          |
| 3          | Y        | Y          |
| 4          | N        | N          |
```

Output

```
| product_id |
```



584. Find Customer Referee

Table: `Customer`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| referee_id  | int     |
+-------------+---------+
In SQL, 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.
```

Find the names of the customer that are **not referred by** the customer with `id = 2`.

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

The result format is in the following example.

Example 1:

```
Input:
Customer table:
+----+------+------------+
| id | name | referee_id |
+----+------+------------+
| 1  | Will | null       |
```

```
# Write your MySQL query statement below
SELECT name
FROM Customer
WHERE referee_id != 2 OR referee_id IS NULL;
```

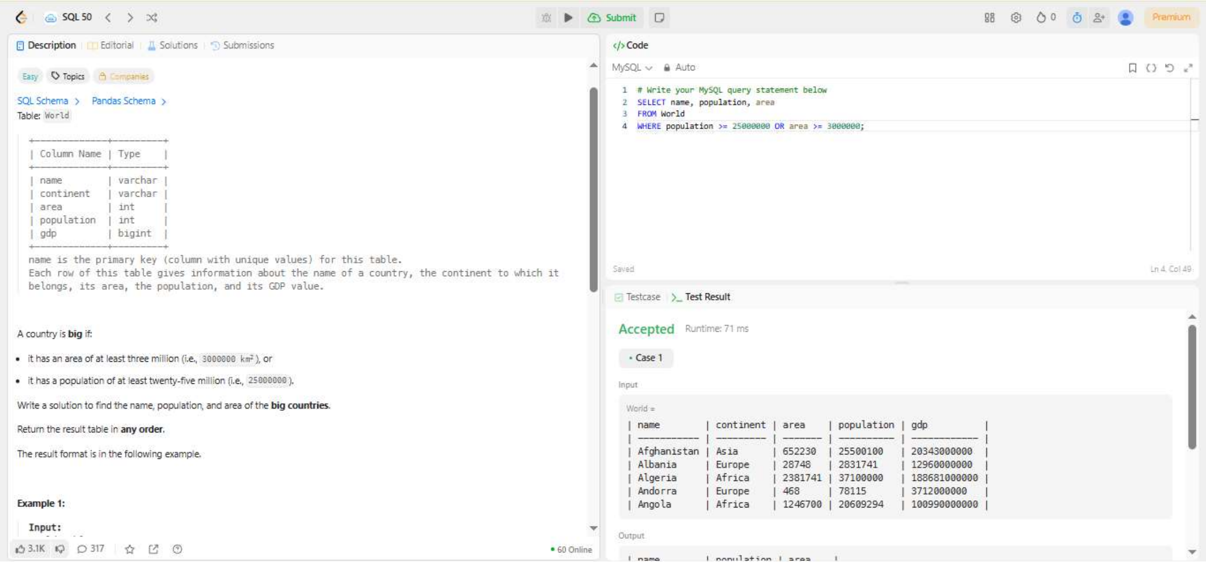
Accepted Runtime: 90 ms

- Case 1

Input

```
Customer =
| id | name | referee_id |
| -- | ---- | ---------- |
| 1  | Will | null       |
| 2  | Jane | null       |
| 3  | Alex | 2          |
| 4  | Bill | null       |
| 5  | Zack | 1          |
| 6  | Mark | 2          |
```

Output



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.

Example 1:

Input:

```sql
# Write your MySQL query statement below
SELECT name, population, area
FROM World
WHERE population >= 25000000 OR area >= 3000000;
```

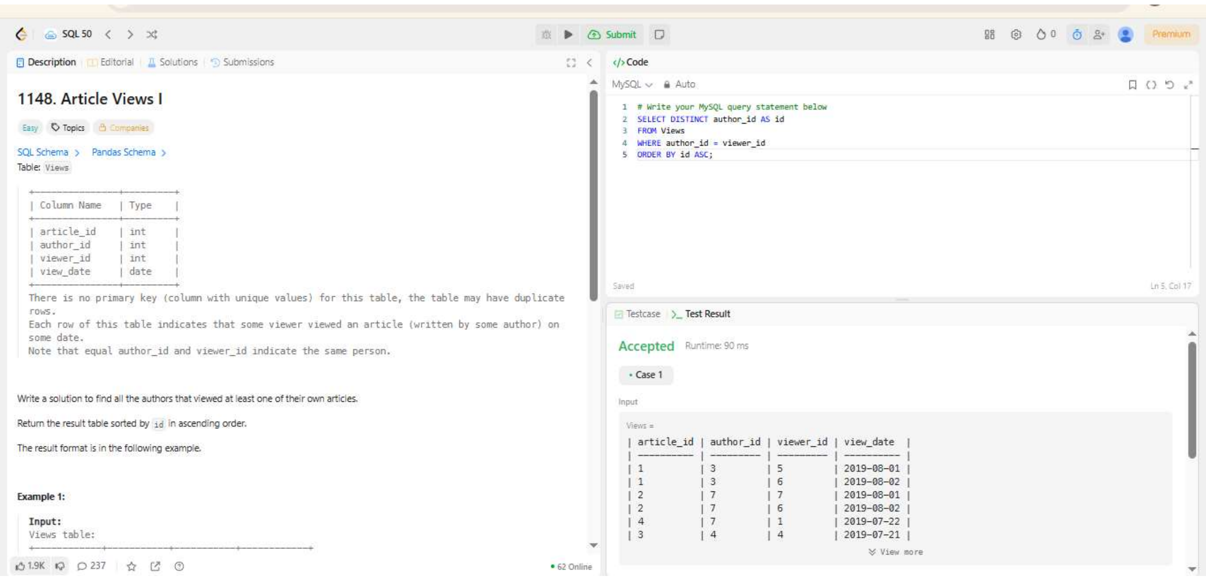
Accepted Runtime: 71 ms

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    |
```



1148. Article Views I

SQL Schema > Pandas Schema >

Table: `Views`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| article_id    | int     |
| author_id     | int     |
| viewer_id     | int     |
| view_date     | date    |
+---------------+---------+
There is no primary key (column with unique values) for this table, the table may have duplicate
rows.
Each row of this table indicates that some viewer viewed an article (written by some author) on
some date.
Note that equal author_id and viewer_id indicate the same person.
```

Write a solution to find all the authors that viewed at least one of their own articles.

Return the result table sorted by `id` in ascending order.

The result format is in the following example.

Example 1:

Input:

Views table:

```
+------------+-----------+-----------+------------+
```

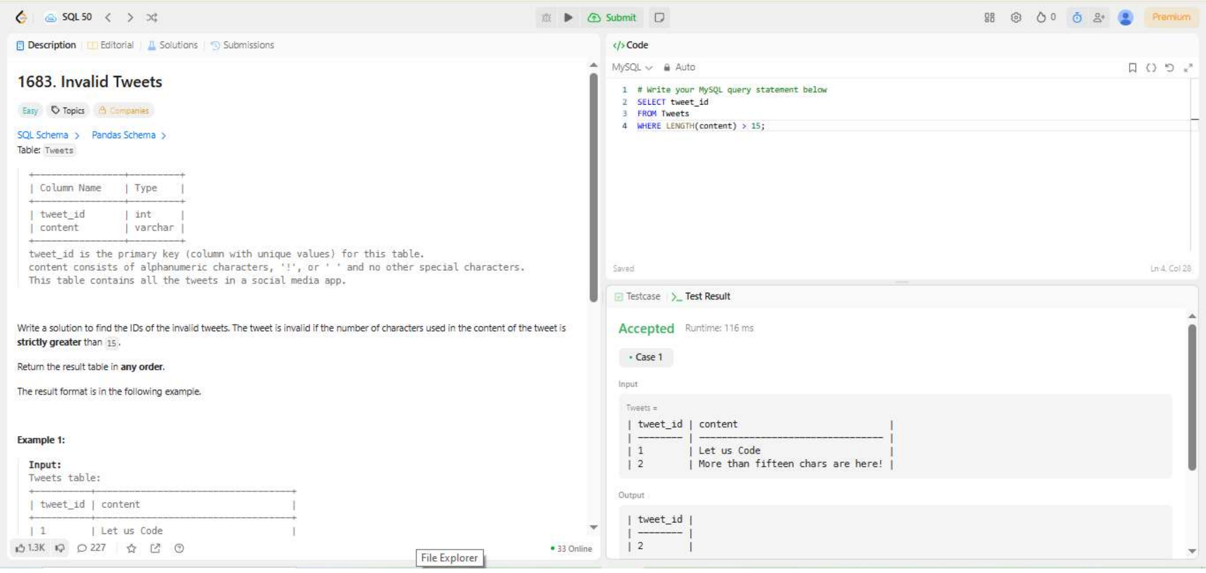
```sql
# Write your MySQL query statement below
SELECT DISTINCT author_id AS id
FROM Views
WHERE author_id = viewer_id
ORDER BY id ASC;
```

Accepted Runtime: 90 ms

Input

```
Views =
| article_id | author_id | viewer_id | view_date  |
| ---------- | --------- | --------- | ---------- |
| 1          | 3         | 5         | 2019-08-01 |
| 1          | 3         | 6         | 2019-08-02 |
| 2          | 7         | 7         | 2019-08-01 |
| 2          | 7         | 6         | 2019-08-02 |
| 4          | 7         | 1         | 2019-07-22 |
| 3          | 4         | 4         | 2019-07-21 |
```

View more



1683. Invalid Tweets

SQL Schema > Pandas Schema >

Table: `Tweets`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| tweet_id       | int     |
| content        | varchar |
+----------------+---------+
tweet_id is the primary key (column with unique values) for this table.
content consists of alphanumeric characters, '!', or ' ' and no other special characters.
This table contains all the tweets in a social media app.
```

Write a solution to find the IDs of the invalid tweets. The tweet is invalid if the number of characters used in the content of the tweet is **strictly greater** than `15`.

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

The result format is in the following example.

Example 1:

Input:

Tweets table:

```
+----------+-----------------------------------+
| tweet_id | content                           |
+----------+-----------------------------------+
| 1        | Let us Code                       |
```

```sql
# Write your MySQL query statement below
SELECT tweet_id
FROM Tweets
WHERE LENGTH(content) > 15;
```

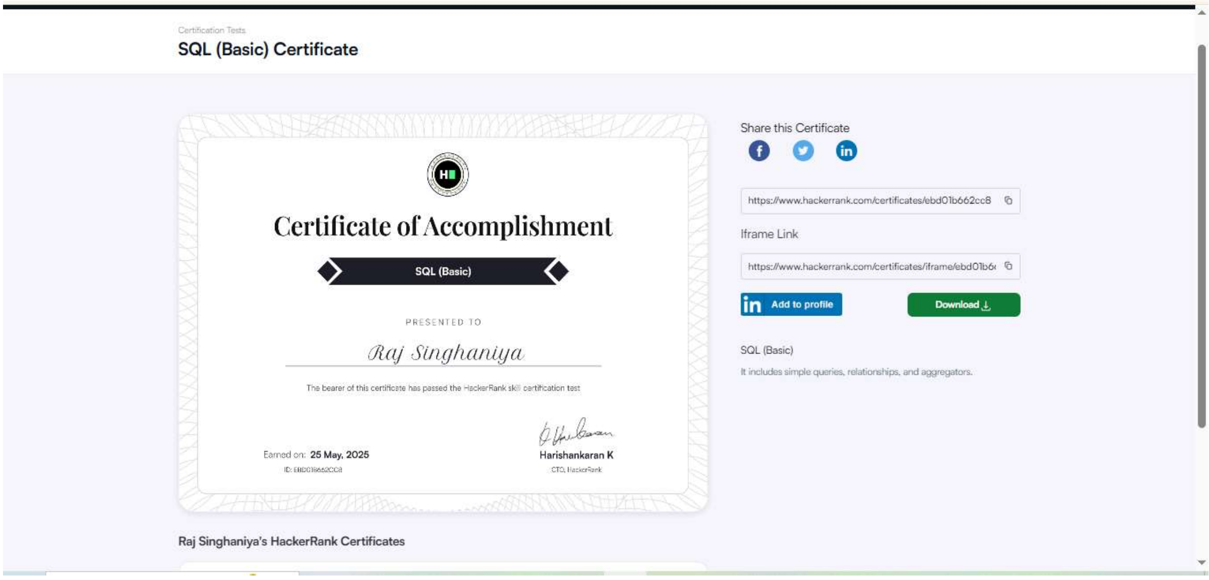
Accepted Runtime: 116 ms

Input

```
Tweets =
| tweet_id | content                          |
| -------- | -------------------------------- |
| 1        | Let us Code                      |
| 2        | More than fifteen chars are here! |
```

Output

```
| tweet_id |
| -------- |
| 2        |
```



Certification Tests

SQL (Basic) Certificate

Certificate of Accomplishment

SQL (Basic)

PRESENTED TO

Raj Singhaniya

The bearer of this certificate has passed the HackerRank skill certification test

Earned on: **25 May, 2025**

ID: EBD01B662CC8

Harishankaran K

CTO, HackerRank

Share this Certificate

https://www.hackerrank.com/certificates/ebd01b662cc8

Iframe Link

https://www.hackerrank.com/certificates/iframe/ebd01b6

SQL (Basic)

It includes simple queries, relationships, and aggregators.

Raj Singhaniya's HackerRank Certificates