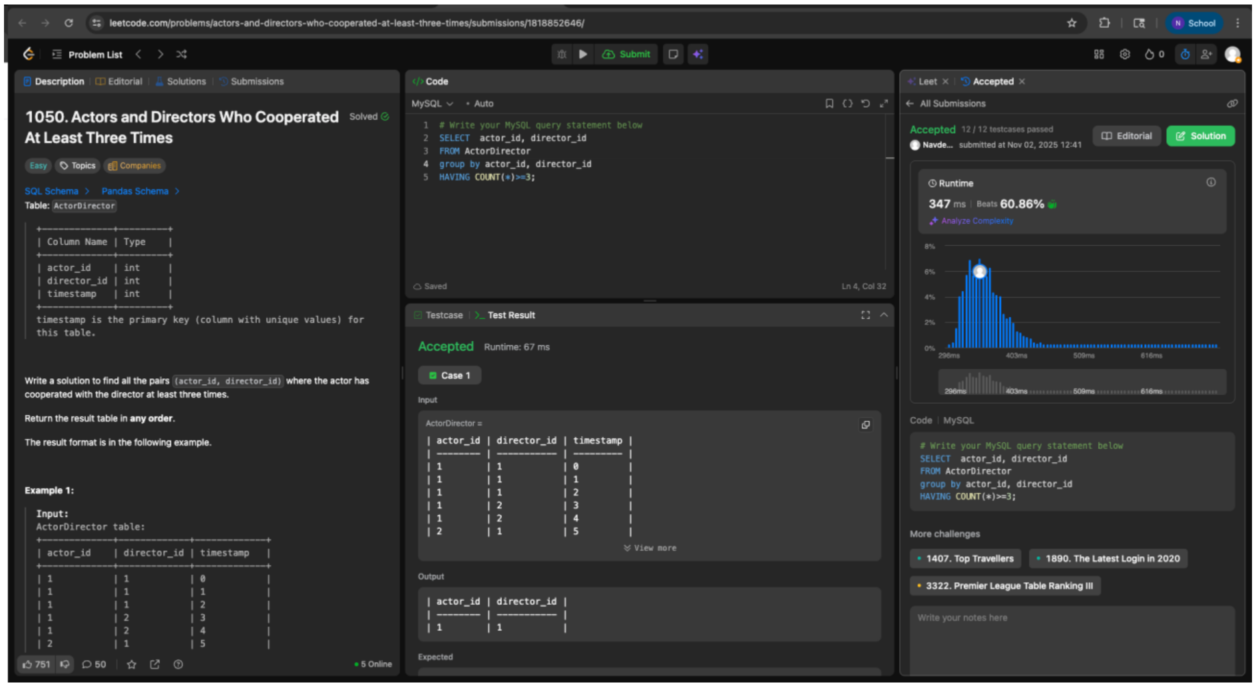
Data Science Bootcamp

Week 5 – SQL

Navdeep Mugathihalli Kumaregowda – nm4686

Q1)1050. Actors and Directors Who Cooperated At Least Three Times

SQL Solution:


leetcode.com/problems/actors-and-directors-who-cooperated-at-least-three-times/submissions/1818852646/
Problem List
Submit
Description | Editorial | Solutions | Submissions
Code
Leet | Accepted
School

1050. Actors and Directors Who Cooperated At Least Three Times Solved

Easy · Topics · Companies

SQL Schema > Pandas Schema >

Table: `ActorDirector`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| actor_id    | int     |
| director_id | int     |
| timestamp   | int     |
+-------------+---------+
timestamp is the primary key (column with unique values) for
this table.
```

Write a solution to find all 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 result format is in the following example.

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           |
```

751 · 50 · 5 Online

MySQL · Auto

```sql
# Write your MySQL query statement below
SELECT  actor_id, director_id
FROM ActorDirector
group by actor_id, director_id
HAVING COUNT(*)>=3;
```

Saved · Ln 4, Col 32

Testcase | Test Result

Accepted Runtime: 67 ms

Case 1

Input

ActorDirector =

```
| actor_id | director_id | timestamp |
| -------- | ----------- | --------- |
| 1        | 1           | 0         |
| 1        | 1           | 1         |
| 1        | 1           | 2         |
| 1        | 2           | 3         |
| 1        | 2           | 4         |
| 2        | 1           | 5         |
```

View more

Output

```
| actor_id | director_id |
| -------- | ----------- |
| 1        | 1           |
```

Expected

All Submissions

Accepted 12 / 12 testcases passed

Navde... submitted at Nov 02, 2025 12:41

Editorial · Solution

Runtime

347 ms | Beats 60.86%

Analyze Complexity

Code | MySQL

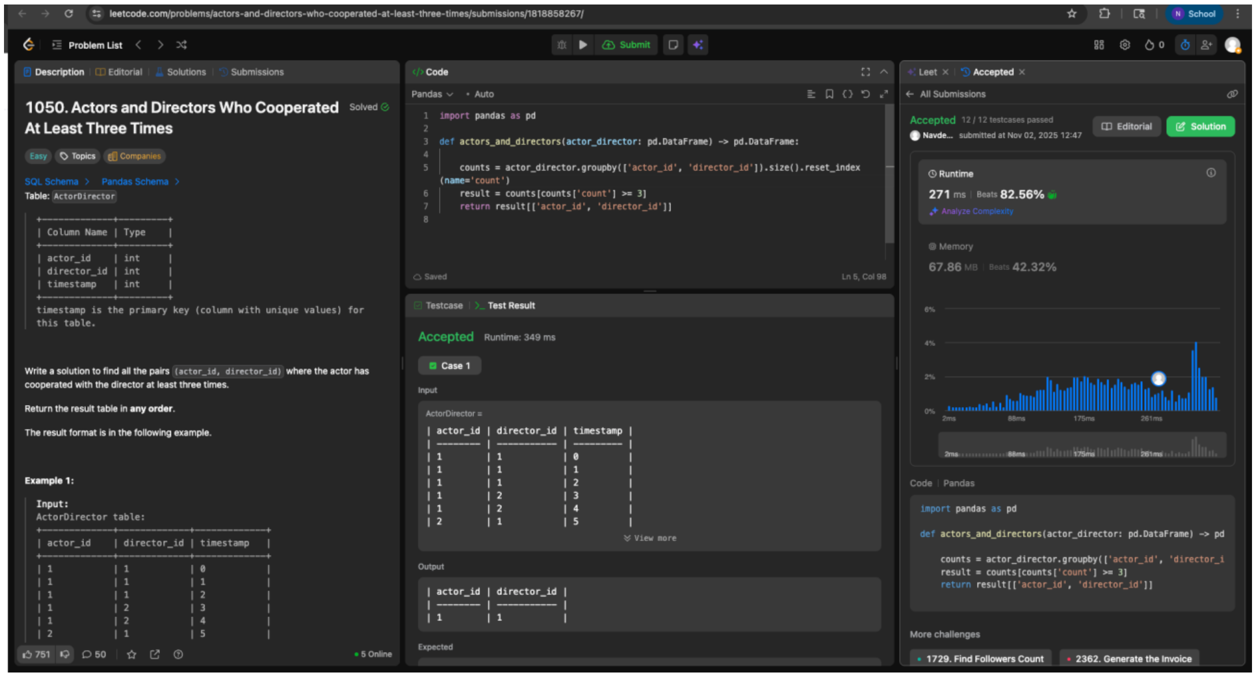
```sql
# Write your MySQL query statement below
SELECT  actor_id, director_id
FROM ActorDirector
group by actor_id, director_id
HAVING COUNT(*)>=3;
```

More challenges

- 1407. Top Travellers
- 1890. The Latest Login in 2020
- 3322. Premier League Table Ranking III

Write your notes here

Pandas Solution:



leetcode.com/problems/actors-and-directors-who-cooperated-at-least-three-times/submissions/1818858267/

1050. Actors and Directors Who Cooperated At Least Three Times — Solved

Easy · Topics · Companies

SQL Schema > Pandas Schema >

Table: `ActorDirector`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| actor_id    | int     |
| director_id | int     |
| timestamp   | int     |
+-------------+---------+
timestamp is the primary key (column with unique values) for
this table.
```

Write a solution to find all 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 result format is in the following example.

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           |
```

751 · 50 · 5 Online

Code — Pandas · Auto

```python
import pandas as pd

def actors_and_directors(actor_director: pd.DataFrame) -> pd.DataFrame:

    counts = actor_director.groupby(['actor_id', 'director_id']).size().reset_index
(name='count')
    result = counts[counts['count'] >= 3]
    return result[['actor_id', 'director_id']]

```

Saved · Ln 5, Col 98

Testcase | Test Result

Accepted · Runtime: 349 ms

Case 1

Input

```
ActorDirector =
| actor_id | director_id | timestamp |
| -------- | ----------- | --------- |
| 1        | 1           | 0         |
| 1        | 1           | 1         |
| 1        | 1           | 2         |
| 1        | 2           | 3         |
| 1        | 2           | 4         |
| 2        | 1           | 5         |
```

View more

Output

```
| actor_id | director_id |
| -------- | ----------- |
| 1        | 1           |
```

Expected

All Submissions

Accepted 12 / 12 testcases passed
Navde... submitted at Nov 02, 2025 12:47

Runtime 271 ms | Beats 82.56%

Memory 67.86 MB | Beats 42.32%

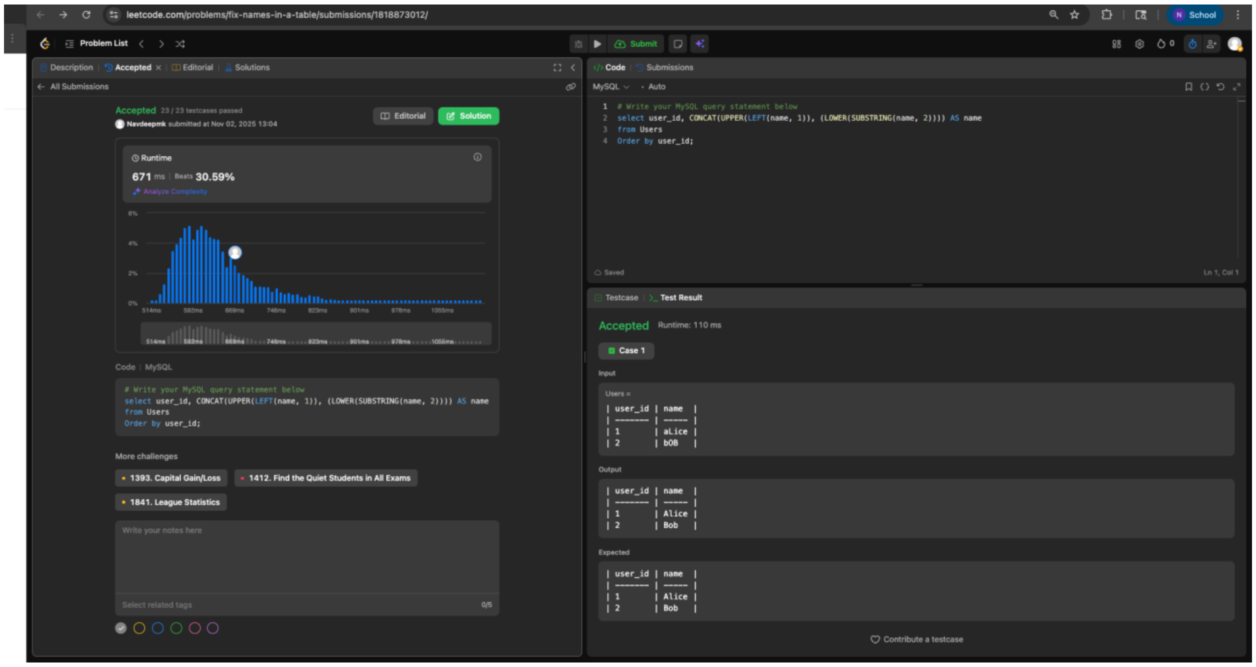
Code | Pandas

```python
import pandas as pd

def actors_and_directors(actor_director: pd.DataFrame) -> pd.DataFrame

    counts = actor_director.groupby(['actor_id', 'director_i
    result = counts[counts['count'] >= 3]
    return result[['actor_id', 'director_id']]
```

More challenges: 1729. Find Followers Count · 2362. Generate the Invoice

Q2)**1667. Fix Names in a Table**

SQL Solution:



leetcode.com/problems/fix-names-in-a-table/submissions/1818873012/

All Submissions

Accepted 23 / 23 testcases passed
Navdeepmk submitted at Nov 02, 2025 13:04

Runtime 671 ms | Beats 30.59%

Code | MySQL

```sql
# Write your MySQL query statement below
select user_id, CONCAT(UPPER(LEFT(name, 1)), (LOWER(SUBSTRING(name, 2)))) AS name
from Users
Order by user_id;
```

More challenges: 1393. Capital Gain/Loss · 1412. Find the Quiet Students in All Exams · 1841. League Statistics

Code — MySQL · Auto

```sql
# Write your MySQL query statement below
select user_id, CONCAT(UPPER(LEFT(name, 1)), (LOWER(SUBSTRING(name, 2)))) AS name
from Users
Order by user_id;
```

Saved · Ln 1, Col 1

Testcase | Test Result

Accepted · Runtime: 110 ms

Case 1

Input

```
Users =
| user_id | name  |
| ------- | ----- |
| 1       | aLice |
| 2       | bOB   |
```

Output

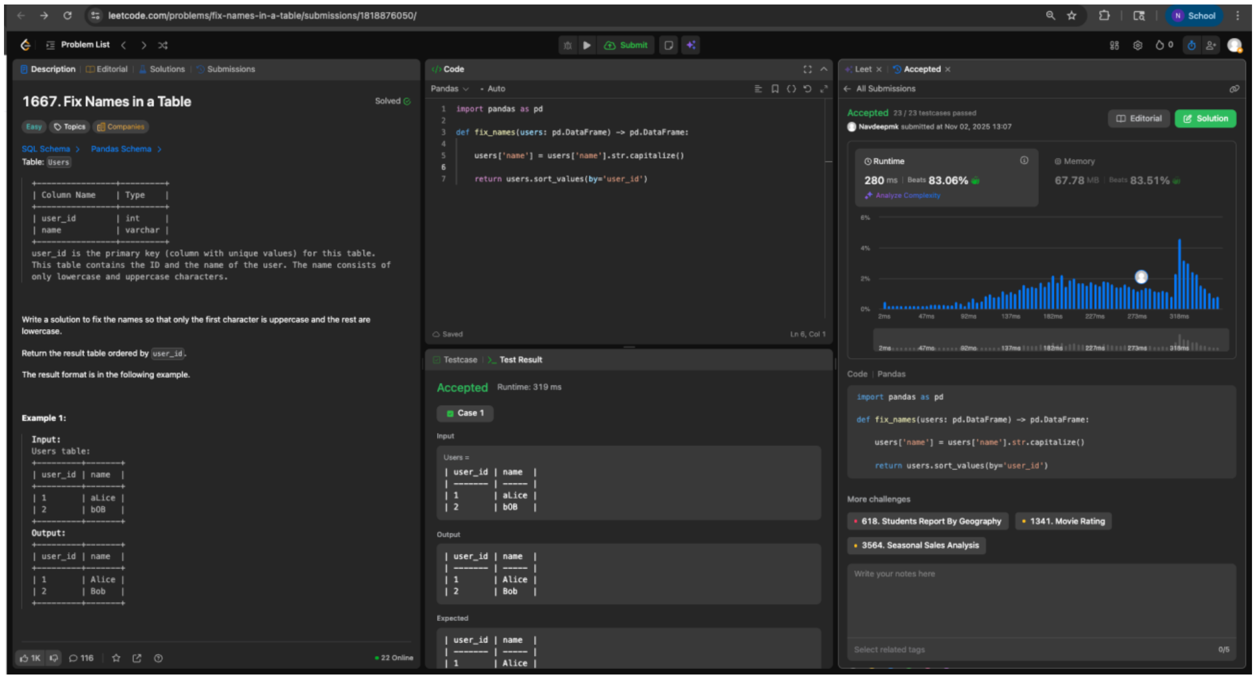
```
| user_id | name  |
| ------- | ----- |
| 1       | Alice |
| 2       | Bob   |
```

Expected

```
| user_id | name  |
| ------- | ----- |
| 1       | Alice |
| 2       | Bob   |
```

Contribute a testcase

Pandas Solution:



1667. Fix Names in a Table Solved

Easy Topics Companies

SQL Schema > Pandas Schema >

Table: `Users`

```
+----------------+---------+
| Column Name    | Type    |
+----------------+---------+
| user_id        | int     |
| name           | varchar |
+----------------+---------+
user_id is the primary key (column with unique values) for this table.
This table contains the ID and the name of the user. The name consists of
only lowercase and uppercase characters.
```

Write a solution to fix the names so that only the first character is uppercase and the rest are lowercase.

Return the result table ordered by `user_id`.

The result format is in the following example.

Example 1:

```
Input:
Users table:
+---------+-------+
| user_id | name  |
+---------+-------+
| 1       | aLice |
| 2       | bOB   |
+---------+-------+
Output:
+---------+-------+
| user_id | name  |
+---------+-------+
| 1       | Alice |
| 2       | Bob   |
+---------+-------+
```

```python
import pandas as pd

def fix_names(users: pd.DataFrame) -> pd.DataFrame:

    users['name'] = users['name'].str.capitalize()

    return users.sort_values(by='user_id')
```

Accepted Runtime: 319 ms

Case 1

Input

```
Users =
| user_id | name  |
| ------- | ----- |
| 1       | aLice |
| 2       | bOB   |
```

Output

```
| user_id | name  |
| ------- | ----- |
| 1       | Alice |
| 2       | Bob   |
```

Expected

```
| user_id | name  |
| ------- | ----- |
| 1       | Alice |
```

Accepted 23 / 23 testcases passed

Navdeepmk submitted at Nov 02, 2025 13:07

Runtime 280 ms | Beats 83.06% Analyze Complexity

Memory 67.78 MB | Beats 83.51%

Code | Pandas

```python
import pandas as pd

def fix_names(users: pd.DataFrame) -> pd.DataFrame:

    users['name'] = users['name'].str.capitalize()

    return users.sort_values(by='user_id')
```

More challenges

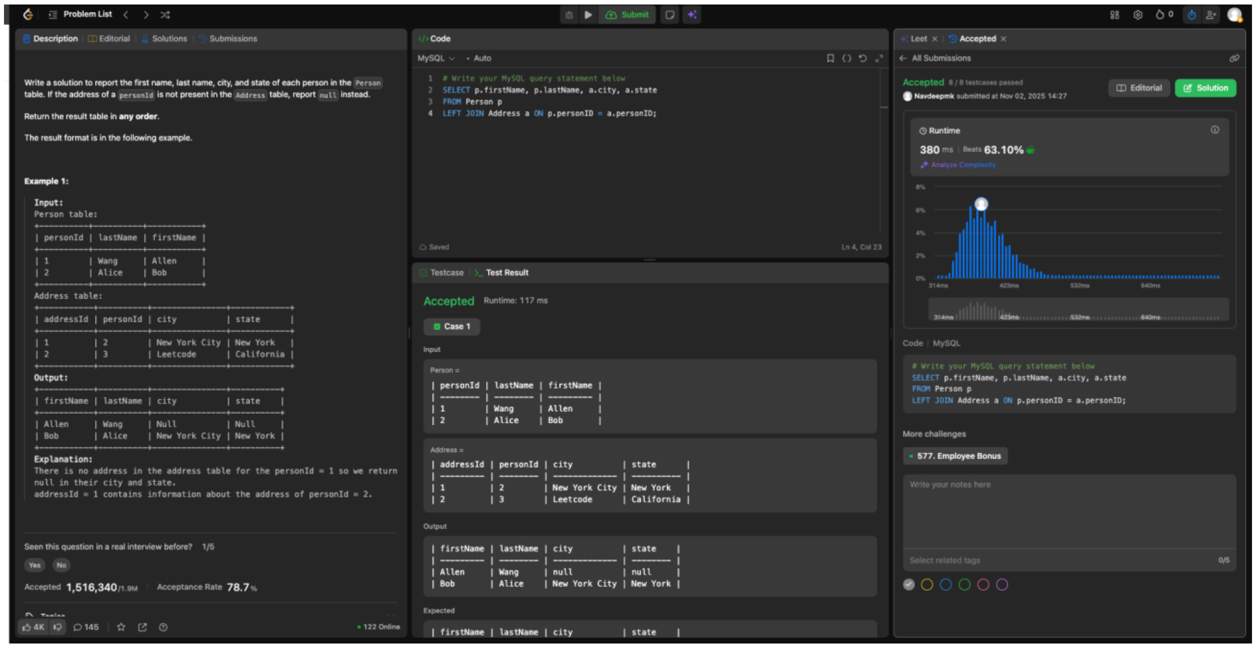
618. Students Report By Geography 1341. Movie Rating 3564. Seasonal Sales Analysis

Write your notes here

Select related tags 0/5

Q3)**175. Combine Two Tables**

SQL Solution:



Write a solution to report the first name, last name, city, and state of each person in the `Person` table. If the address of a `personId` is not present in the `Address` table, report `null` instead.

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

The result format is in the following example.

Example 1:

```
Input:
Person table:
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
Address table:
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
Output:
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
Explanation:
There is no address in the address table for the personId = 1 so we return
null in their city and state.
addressId = 1 contains information about the address of personId = 2.
```

Seen this question in a real interview before? 1/5

Accepted 1,516,340/1.9M | Acceptance Rate 78.7%

```sql
# Write your MySQL query statement below
SELECT p.firstName, p.lastName, a.city, a.state
FROM Person p
LEFT JOIN Address a ON p.personID = a.personID;
```

Accepted Runtime: 117 ms

Case 1

Input

```
Person =
| personId | lastName | firstName |
| -------- | -------- | --------- |
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
```

```
Address =
| addressId | personId | city          | state      |
| --------- | -------- | ------------- | ---------- |
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
```

Output

```
| firstName | lastName | city          | state    |
| --------- | -------- | ------------- | -------- |
| Allen     | Wang     | null          | null     |
| Bob       | Alice    | New York City | New York |
```

Expected

```
| firstName | lastName | city          | state    |
```

Accepted 8 / 8 testcases passed

Navdeepmk submitted at Nov 02, 2025 14:27

Runtime 380 ms | Beats 63.10% Analyze Complexity

Code | MySQL

```sql
# Write your MySQL query statement below
SELECT p.firstName, p.lastName, a.city, a.state
FROM Person p
LEFT JOIN Address a ON p.personID = a.personID;
```

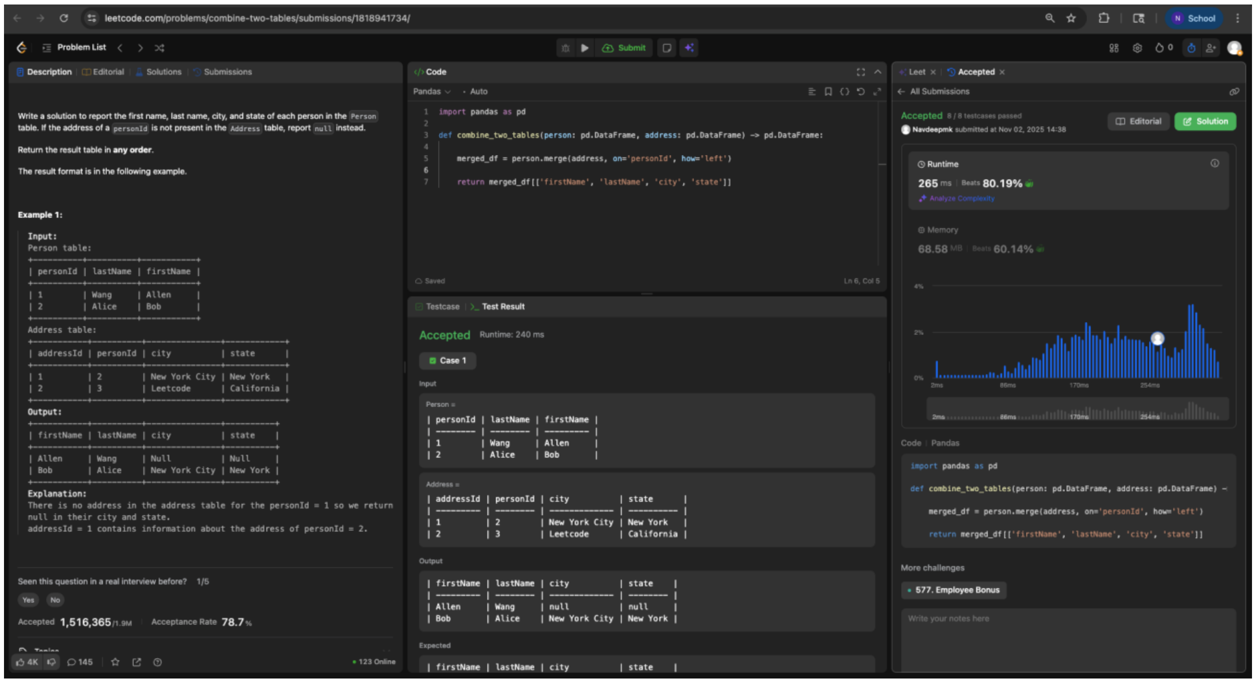
More challenges

577. Employee Bonus

Write your notes here

Select related tags 0/5

Pandas Solution:



leetcode.com/problems/combine-two-tables/submissions/1818941734/

Write a solution to report the first name, last name, city, and state of each person in the `Person` table. If the address of a `personId` is not present in the `Address` table, report `null` instead.

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

The result format is in the following example.

Example 1:

```
Input:
Person table:
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
Address table:
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
Output:
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
Explanation:
There is no address in the address table for the personId = 1 so we return
null in their city and state.
addressId = 1 contains information about the address of personId = 2.
```

Seen this question in a real interview before? 1/5

Accepted 1,516,365/1.9M | Acceptance Rate 78.7%

Code — Pandas

```python
import pandas as pd

def combine_two_tables(person: pd.DataFrame, address: pd.DataFrame) -> pd.DataFrame:

    merged_df = person.merge(address, on='personId', how='left')

    return merged_df[['firstName', 'lastName', 'city', 'state']]
```

Test Result

Accepted Runtime: 240 ms

Case 1

Input

```
Person =
| personId | lastName | firstName |
| -------- | -------- | --------- |
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |

Address =
| addressId | personId | city          | state      |
| --------- | -------- | ------------- | ---------- |
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
```

Output

```
| firstName | lastName | city          | state    |
| --------- | -------- | ------------- | -------- |
| Allen     | Wang     | null          | null     |
| Bob       | Alice    | New York City | New York |
```

Expected

```
| firstName | lastName | city          | state    |
```

Accepted 8 / 8 testcases passed
Navdeepmk submitted at Nov 02, 2025 14:38

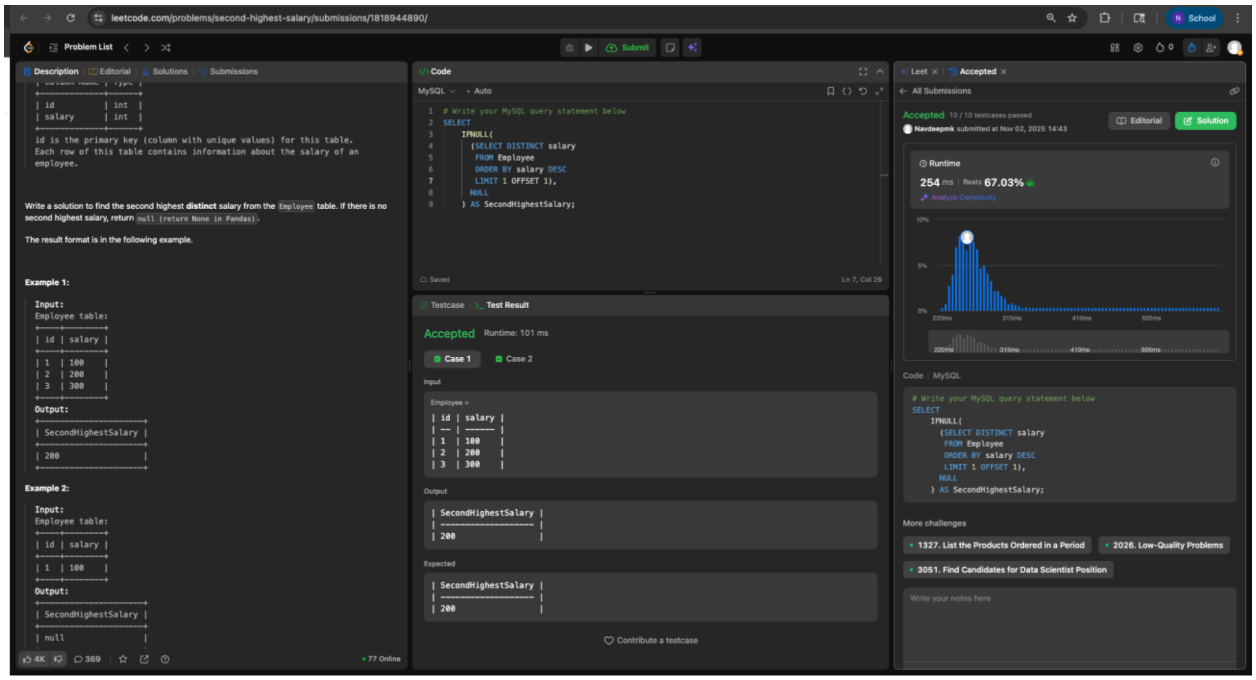
Runtime 265 ms | Beats 80.19%

Memory 68.58 MB | Beats 60.14%

More challenges: 577. Employee Bonus

Q4)176. Second Highest Salary

SQL Solution:



leetcode.com/problems/second-highest-salary/submissions/1818944890/

```
+-------------+------+
| 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    |
| 3  | 300    |
+----+--------+
Output:
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

Example 2:

```
Input:
Employee table:
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
+----+--------+
Output:
+---------------------+
| SecondHighestSalary |
+---------------------+
| null                |
```

Code — MySQL

```sql
# Write your MySQL query statement below
SELECT
    IFNULL(
      (SELECT DISTINCT salary
       FROM Employee
       ORDER BY salary DESC
       LIMIT 1 OFFSET 1),
      NULL
    ) AS SecondHighestSalary;
```

Test Result

Accepted Runtime: 101 ms

Case 1 | Case 2

Input

```
Employee =
| id | salary |
| -- | ------ |
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
```

Output

```
| SecondHighestSalary |
| ------------------- |
| 200                 |
```

Expected

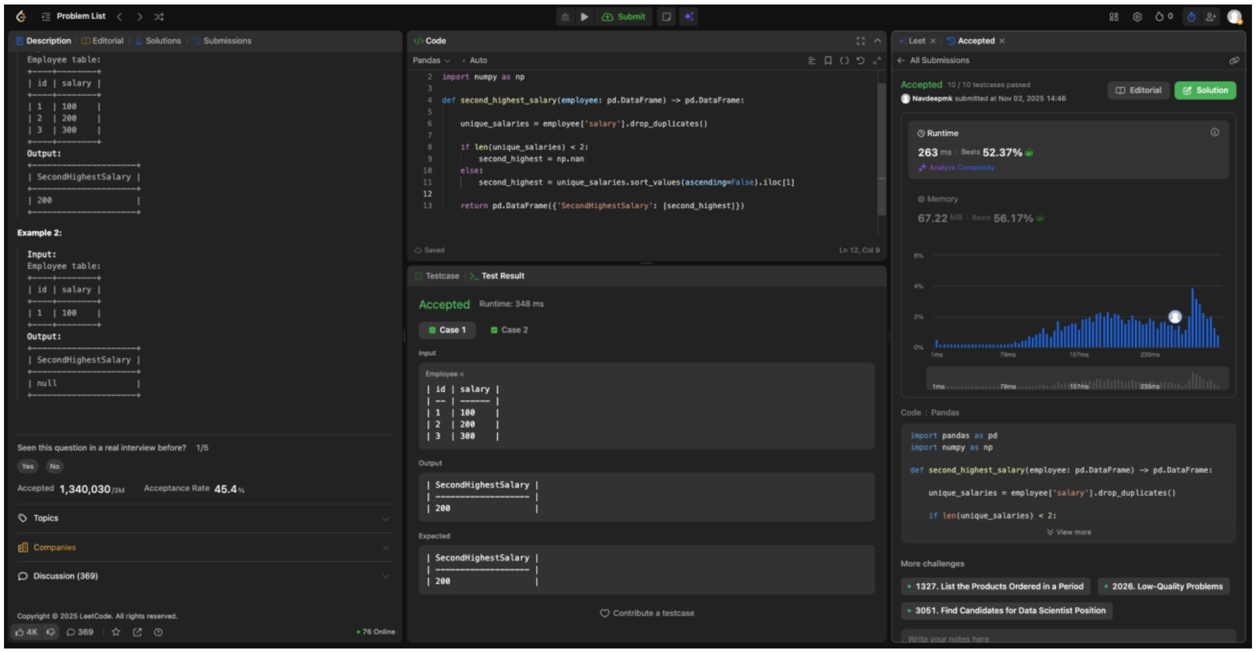
```
| SecondHighestSalary |
| ------------------- |
| 200                 |
```

Accepted 10 / 10 testcases passed
Navdeepmk submitted at Nov 02, 2025 14:43

Runtime 254 ms | Beats 67.03%

More challenges: 1327. List the Products Ordered in a Period · 2026. Low-Quality Problems · 3051. Find Candidates for Data Scientist Position

Pandas Solution:



```
Employee table:
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
Output:
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

Example 2:

```
Input:
Employee table:
+----+--------+
| id | salary |
+----+--------+
| 1  | 100    |
+----+--------+
Output:
+---------------------+
| SecondHighestSalary |
+---------------------+
| null                |
+---------------------+
```

Seen this question in a real interview before? 1/5

Yes No

Accepted 1,340,030/3M | Acceptance Rate 45.4%

Copyright © 2025 LeetCode. All rights reserved.

```python
import numpy as np

def second_highest_salary(employee: pd.DataFrame) -> pd.DataFrame:

    unique_salaries = employee['salary'].drop_duplicates()

    if len(unique_salaries) < 2:
        second_highest = np.nan
    else:
        second_highest = unique_salaries.sort_values(ascending=False).iloc[1]

    return pd.DataFrame({'SecondHighestSalary': [second_highest]})
```

Accepted Runtime: 348 ms

Case 1 Case 2

Input

```
Employee =
| id | salary |
| -- | ------ |
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
```

Output

```
| SecondHighestSalary |
| ------------------- |
| 200                 |
```

Expected

```
| SecondHighestSalary |
| ------------------- |
| 200                 |
```

Accepted 10 / 10 testcases passed

Navdeepmk submitted at Nov 02, 2025 14:46

Runtime

263 ms | Beats 52.37%

Memory

67.22 MB | Beats 56.17%

Code | Pandas

```python
import pandas as pd
import numpy as np

def second_highest_salary(employee: pd.DataFrame) -> pd.DataFrame:

    unique_salaries = employee['salary'].drop_duplicates()

    if len(unique_salaries) < 2:
```

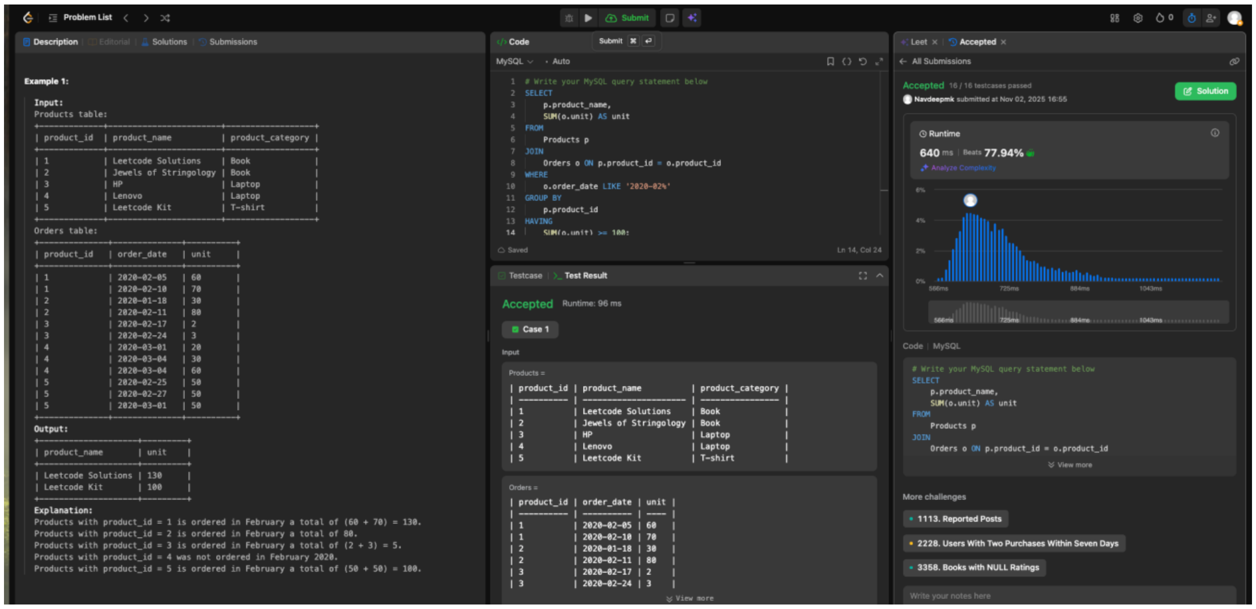
View more

More challenges

- 1327. List the Products Ordered in a Period
- 2026. Low-Quality Problems
- 3051. Find Candidates for Data Scientist Position

Q5)**1327. List the Products Ordered in a Period**

SQL Solution:



Example 1:

```
Input:
Products table:
+-------------+-----------------------+------------------+
| product_id  | product_name          | product_category |
+-------------+-----------------------+------------------+
| 1           | Leetcode Solutions    | Book             |
| 2           | Jewels of Stringology | Book             |
| 3           | HP                    | Laptop           |
| 4           | Lenovo                | Laptop           |
| 5           | Leetcode Kit          | T-shirt          |
+-------------+-----------------------+------------------+
Orders table:
+--------------+--------------+----------+
| product_id   | order_date   | unit     |
+--------------+--------------+----------+
| 1            | 2020-02-05   | 60       |
| 1            | 2020-02-10   | 70       |
| 2            | 2020-01-18   | 30       |
| 2            | 2020-02-11   | 80       |
| 3            | 2020-02-17   | 2        |
| 3            | 2020-02-24   | 3        |
| 4            | 2020-03-01   | 20       |
| 4            | 2020-03-04   | 30       |
| 4            | 2020-03-04   | 60       |
| 5            | 2020-02-25   | 50       |
| 5            | 2020-02-27   | 50       |
| 5            | 2020-03-01   | 50       |
+--------------+--------------+----------+
Output:
+--------------------+---------+
| product_name       | unit    |
+--------------------+---------+
| Leetcode Solutions | 130     |
| Leetcode Kit       | 100     |
+--------------------+---------+
Explanation:
Products with product_id = 1 is ordered in February a total of (60 + 70) = 130.
Products with product_id = 2 is ordered in February a total of 80.
Products with product_id = 3 is ordered in February a total of (2 + 3) = 5.
Products with product_id = 4 was not ordered in February 2020.
Products with product_id = 5 is ordered in February a total of (50 + 50) = 100.
```

```sql
# Write your MySQL query statement below
SELECT
    p.product_name,
    SUM(o.unit) AS unit
FROM
    Products p
JOIN
    Orders o ON p.product_id = o.product_id
WHERE
    o.order_date LIKE '2020-02%'
GROUP BY
    p.product_id
HAVING
    SUM(o.unit) >= 100;
```

Accepted Runtime: 96 ms

Case 1

Input

```
Products =
| product_id | product_name          | product_category |
| ---------- | --------------------- | ---------------- |
| 1          | Leetcode Solutions    | Book             |
| 2          | Jewels of Stringology | Book             |
| 3          | HP                    | Laptop           |
| 4          | Lenovo                | Laptop           |
| 5          | Leetcode Kit          | T-shirt          |
```

```
Orders =
| product_id | order_date | unit |
| ---------- | ---------- | ---- |
| 1          | 2020-02-05 | 60   |
| 1          | 2020-02-10 | 70   |
| 2          | 2020-01-18 | 30   |
| 2          | 2020-02-11 | 80   |
| 3          | 2020-02-17 | 2    |
| 3          | 2020-02-24 | 3    |
```

Accepted 16 / 16 testcases passed

Navdeepmk submitted at Nov 02, 2025 16:55

Runtime

640 ms | Beats 77.94%

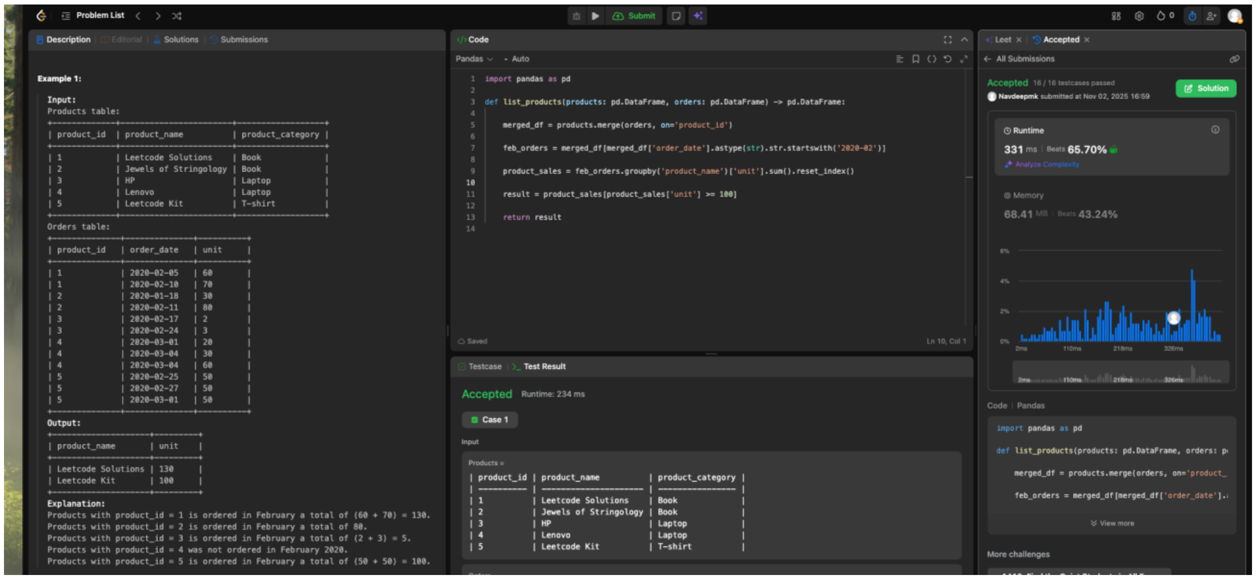
Code | MySQL

```sql
# Write your MySQL query statement below
SELECT
    p.product_name,
    SUM(o.unit) AS unit
FROM
    Products p
JOIN
    Orders o ON p.product_id = o.product_id
```

More challenges

- 1113. Reported Posts
- 2228. Users With Two Purchases Within Seven Days
- 3358. Books with NULL Ratings

Pandas Solution:



Problem List

Description | Editorial | Solutions | Submissions

Example 1:

```
Input:
Products table:
+-------------+-----------------------+------------------+
| product_id  | product_name          | product_category |
+-------------+-----------------------+------------------+
| 1           | Leetcode Solutions    | Book             |
| 2           | Jewels of Stringology | Book             |
| 3           | HP                    | Laptop           |
| 4           | Lenovo                | Laptop           |
| 5           | Leetcode Kit          | T-shirt          |
+-------------+-----------------------+------------------+
Orders table:
+--------------+--------------+----------+
| product_id   | order_date   | unit     |
+--------------+--------------+----------+
| 1            | 2020-02-05   | 60       |
| 1            | 2020-02-10   | 70       |
| 2            | 2020-01-18   | 30       |
| 2            | 2020-02-11   | 80       |
| 3            | 2020-02-17   | 2        |
| 3            | 2020-02-24   | 3        |
| 4            | 2020-03-01   | 20       |
| 4            | 2020-03-04   | 30       |
| 4            | 2020-03-04   | 60       |
| 5            | 2020-02-25   | 50       |
| 5            | 2020-02-27   | 50       |
| 5            | 2020-03-01   | 50       |
+--------------+--------------+----------+
Output:
+--------------------+---------+
| product_name       | unit    |
+--------------------+---------+
| Leetcode Solutions | 130     |
| Leetcode Kit       | 100     |
+--------------------+---------+
Explanation:
Products with product_id = 1 is ordered in February a total of (60 + 70) = 130.
Products with product_id = 2 is ordered in February a total of 80.
Products with product_id = 3 is ordered in February a total of (2 + 3) = 5.
Products with product_id = 4 was not ordered in February 2020.
Products with product_id = 5 is ordered in February a total of (50 + 50) = 100.
```

Code — Pandas · Auto

```python
import pandas as pd

def list_products(products: pd.DataFrame, orders: pd.DataFrame) -> pd.DataFrame:

    merged_df = products.merge(orders, on='product_id')

    feb_orders = merged_df[merged_df['order_date'].astype(str).str.startswith('2020-02')]

    product_sales = feb_orders.groupby('product_name')['unit'].sum().reset_index()

    result = product_sales[product_sales['unit'] >= 100]

    return result

```

Saved — Ln 10, Col 1

Testcase | Test Result

Accepted Runtime: 234 ms

Case 1

Input

```
Products =
| product_id | product_name          | product_category |
| ---------- | --------------------- | ---------------- |
| 1          | Leetcode Solutions    | Book             |
| 2          | Jewels of Stringology | Book             |
| 3          | HP                    | Laptop           |
| 4          | Lenovo                | Laptop           |
| 5          | Leetcode Kit          | T-shirt          |
```

Leet | Accepted

All Submissions

Accepted 16 / 16 testcases passed

Navdeepmk submitted at Nov 02, 2025 16:59

Runtime 331 ms | Beats 65.70%

Analyze Complexity

Memory 68.41 MB | Beats 43.24%

Code | Pandas

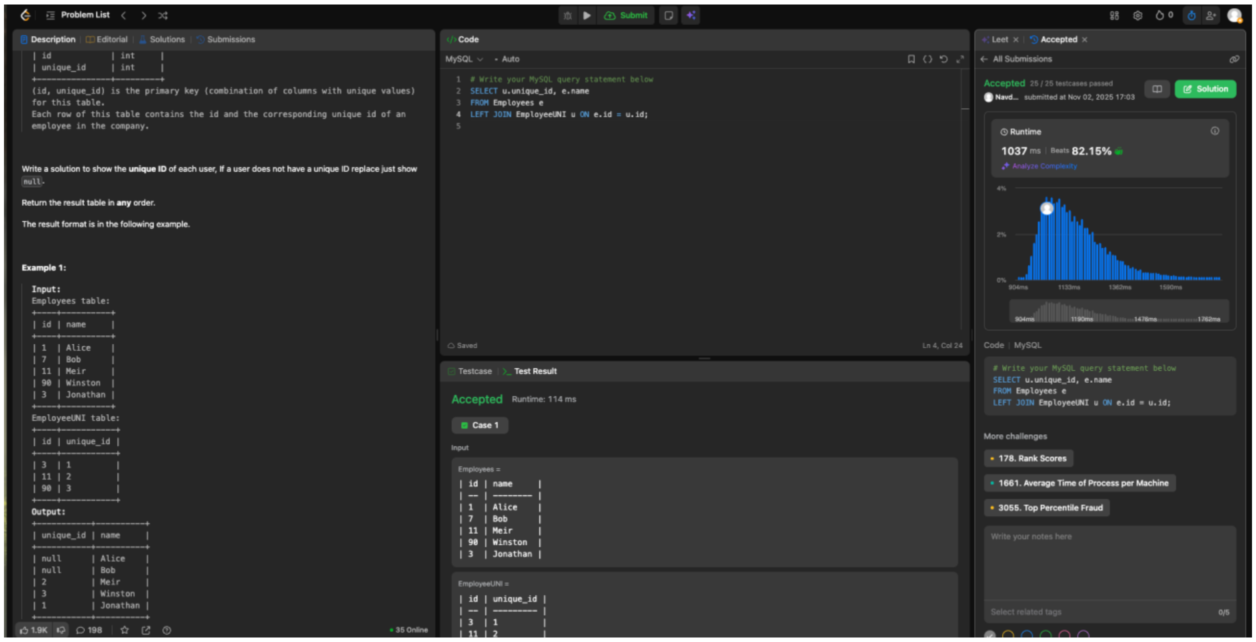
```python
import pandas as pd

def list_products(products: pd.DataFrame, orders: p

    merged_df = products.merge(orders, on='product_

    feb_orders = merged_df[merged_df['order_date'].
```

View more

More challenges

Q6)**1378. Replace Employee ID With The Unique Identifier**

SQL Solution:



Problem List

Description | Editorial | Solutions | Submissions

```
| id            | int      |
| unique_id     | int      |
+---------------+----------+
(id, unique_id) is the primary key (combination of columns with unique values)
for this table.
Each row of this table contains the id and the corresponding unique id of an
employee in the company.
```

Write a solution to show the **unique ID** of each user, If a user does not have a unique ID replace just show `null`.

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

The result format is in the following example.

Example 1:

```
Input:
Employees table:
+----+----------+
| id | name     |
+----+----------+
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |
+----+----------+
EmployeeUNI table:
+----+-----------+
| id | unique_id |
+----+-----------+
| 3  | 1         |
| 11 | 2         |
| 90 | 3         |
+----+-----------+
Output:
+-----------+----------+
| unique_id | name     |
+-----------+----------+
| null      | Alice    |
| null      | Bob      |
| 2         | Meir     |
| 3         | Winston  |
| 1         | Jonathan |
+-----------+----------+
```

1.9K 198 • 35 Online

Code — MySQL · Auto

```sql
# Write your MySQL query statement below
SELECT u.unique_id, e.name
FROM Employees e
LEFT JOIN EmployeeUNI u ON e.id = u.id;

```

Saved — Ln 4, Col 24

Testcase | Test Result

Accepted Runtime: 114 ms

Case 1

Input

```
Employees =
| id | name     |
| -- | -------- |
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |

EmployeeUNI =
| id | unique_id |
| -- | --------- |
| 3  | 1         |
| 11 | 2         |
```

Leet | Accepted

All Submissions

Accepted 25 / 25 testcases passed

Navd... submitted at Nov 02, 2025 17:03

Runtime 1037 ms | Beats 82.15%

Analyze Complexity

Code | MySQL

```sql
# Write your MySQL query statement below
SELECT u.unique_id, e.name
FROM Employees e
LEFT JOIN EmployeeUNI u ON e.id = u.id;
```

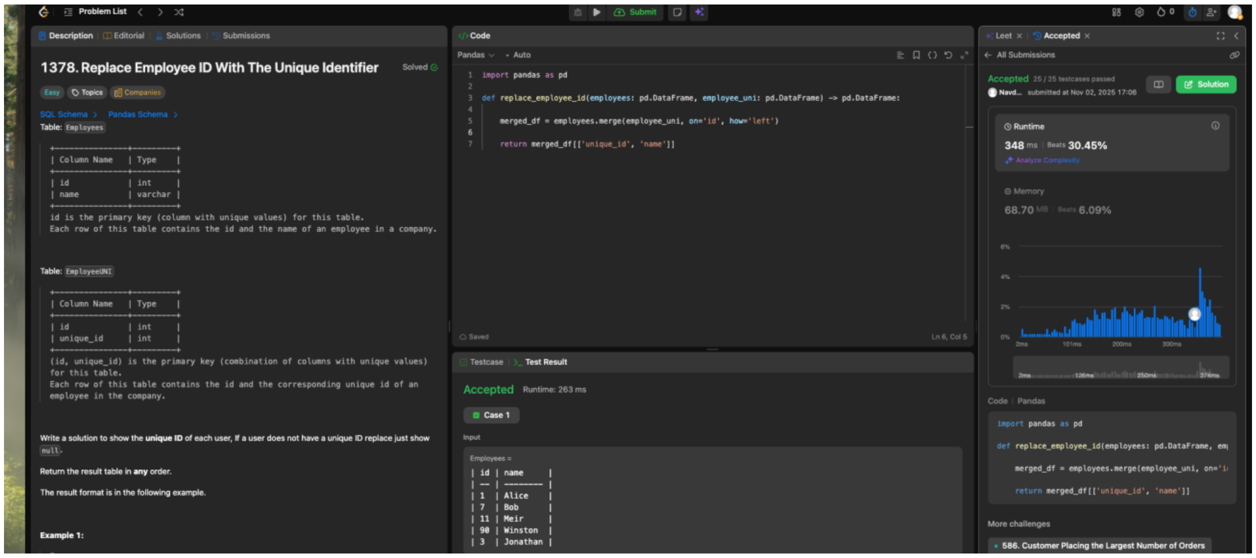
More challenges

- 178. Rank Scores
- 1661. Average Time of Process per Machine
- 3055. Top Percentile Fraud

Write your notes here

Select related tags 0/5

Pandas Solution:



1378. Replace Employee ID With The Unique Identifier

Table: `Employees`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| name          | varchar |
+---------------+---------+
id is the primary key (column with unique values) for this table.
Each row of this table contains the id and the name of an employee in a company.
```

Table: `EmployeeUNI`

```
+---------------+---------+
| Column Name   | Type    |
+---------------+---------+
| id            | int     |
| unique_id     | int     |
+---------------+---------+
(id, unique_id) is the primary key (combination of columns with unique values)
for this table.
Each row of this table contains the id and the corresponding unique id of an
employee in the company.
```

Write a solution to show the **unique ID** of each user, If a user does not have a unique ID replace just show `null`.

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

The result format is in the following example.

Example 1:

```python
import pandas as pd

def replace_employee_id(employees: pd.DataFrame, employee_uni: pd.DataFrame) -> pd.DataFrame:

    merged_df = employees.merge(employee_uni, on='id', how='left')

    return merged_df[['unique_id', 'name']]
```

Accepted Runtime: 263 ms

Case 1

Input

```
Employees =
| id | name     |
| -- | -------- |
| 1  | Alice    |
| 7  | Bob      |
| 11 | Meir     |
| 90 | Winston  |
| 3  | Jonathan |
```

Accepted 25 / 25 testcases passed
Navd... submitted at Nov 02, 2025 17:06

Runtime
348 ms | Beats 30.45%

Memory
68.70 MB | Beats 6.09%

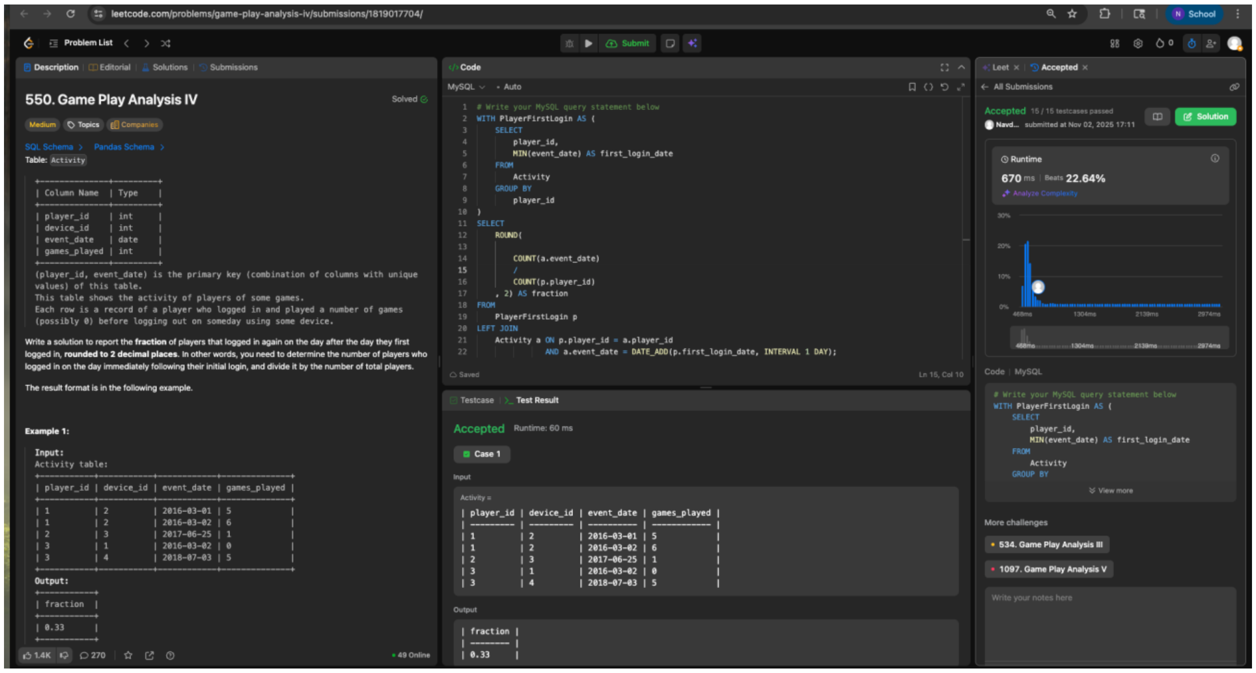
Code | Pandas

```python
import pandas as pd

def replace_employee_id(employees: pd.DataFrame, em

    merged_df = employees.merge(employee_uni, on='i

    return merged_df[['unique_id', 'name']]
```

586. Customer Placing the Largest Number of Orders

Q7)**550. Game Play Analysis IV**

SQL Solution:



leetcode.com/problems/game-play-analysis-iv/submissions/1819017704/

550. Game Play Analysis IV

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 report the **fraction** of players that logged in again on the day after the day they first logged in, **rounded to 2 decimal places**. In other words, you need to determine the number of players who logged in on the day immediately following their initial login, and divide it by the number of total players.

The result format is in the following example.

Example 1:

```
Input:
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-03-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+
Output:
+-----------+
| fraction  |
+-----------+
| 0.33      |
+-----------+
```

```sql
# Write your MySQL query statement below
WITH PlayerFirstLogin AS (
    SELECT
        player_id,
        MIN(event_date) AS first_login_date
    FROM
        Activity
    GROUP BY
        player_id
)
SELECT
    ROUND(

        COUNT(a.event_date)
        /
        COUNT(p.player_id)
    , 2) AS fraction
FROM
    PlayerFirstLogin p
LEFT JOIN
    Activity a ON p.player_id = a.player_id
            AND a.event_date = DATE_ADD(p.first_login_date, INTERVAL 1 DAY);
```

Accepted Runtime: 60 ms

Case 1

Input

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

Output

```
| fraction |
| -------- |
| 0.33     |
```

Accepted 15 / 15 testcases passed
Navd... submitted at Nov 02, 2025 17:11

Runtime
670 ms | Beats 22.64%

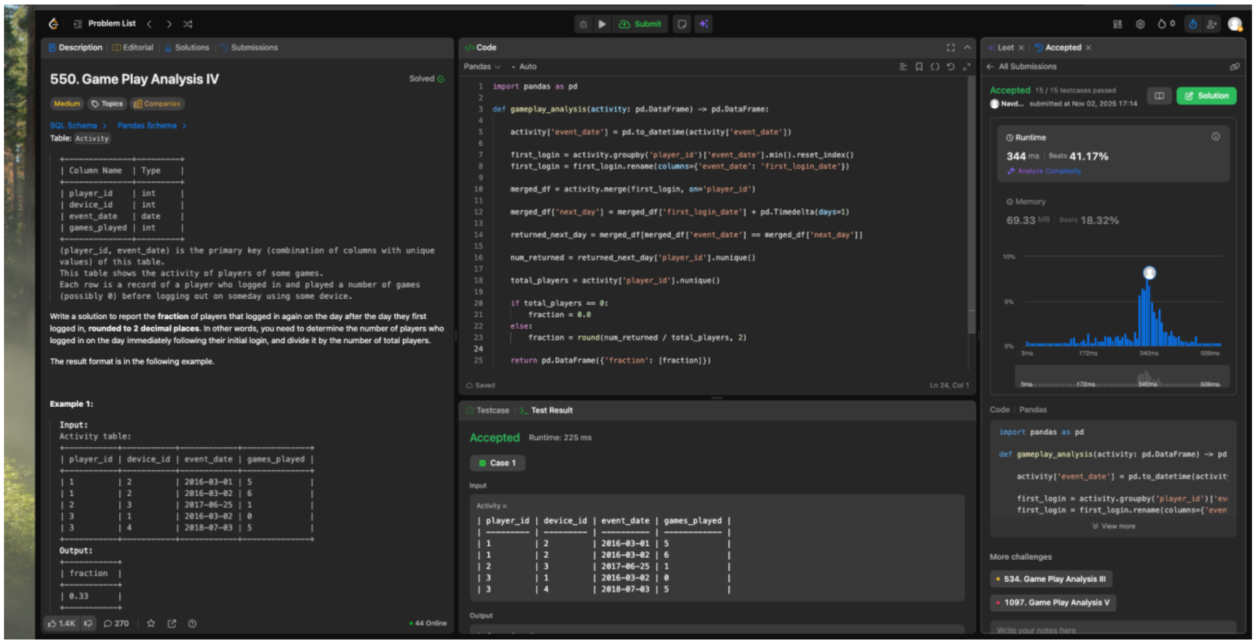
Code | MySQL

```sql
# Write your MySQL query statement below
WITH PlayerFirstLogin AS (
    SELECT
        player_id,
        MIN(event_date) AS first_login_date
    FROM
        Activity
    GROUP BY
```

534. Game Play Analysis III

1097. Game Play Analysis V

Pandas Solution:



550. Game Play Analysis IV

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 report the **fraction** of players that logged in again on the day after the day they first logged in, **rounded to 2 decimal places**. In other words, you need to determine the number of players who logged in on the day immediately following their initial login, and divide it by the number of total players.

The result format is in the following example.

Example 1:

```
Input:
Activity table:
+-----------+-----------+------------+--------------+
| player_id | device_id | event_date | games_played |
+-----------+-----------+------------+--------------+
| 1         | 2         | 2016-03-01 | 5            |
| 1         | 2         | 2016-03-02 | 6            |
| 2         | 3         | 2017-06-25 | 1            |
| 3         | 1         | 2016-03-02 | 0            |
| 3         | 4         | 2018-07-03 | 5            |
+-----------+-----------+------------+--------------+
Output:
+----------+
| fraction |
+----------+
| 0.33     |
+----------+
```

```python
import pandas as pd

def gameplay_analysis(activity: pd.DataFrame) -> pd.DataFrame:

    activity['event_date'] = pd.to_datetime(activity['event_date'])

    first_login = activity.groupby('player_id')['event_date'].min().reset_index()
    first_login = first_login.rename(columns={'event_date': 'first_login_date'})

    merged_df = activity.merge(first_login, on='player_id')

    merged_df['next_day'] = merged_df['first_login_date'] + pd.Timedelta(days=1)

    returned_next_day = merged_df[merged_df['event_date'] == merged_df['next_day']]

    num_returned = returned_next_day['player_id'].nunique()

    total_players = activity['player_id'].nunique()

    if total_players == 0:
        fraction = 0.0
    else:
        fraction = round(num_returned / total_players, 2)

    return pd.DataFrame({'fraction': [fraction]})
```

Accepted Runtime: 225 ms

Case 1

Input

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

Output

Accepted 15 / 15 testcases passed

Navd... submitted at Nov 02, 2025 17:14

Runtime 344 ms | Beats 41.17%

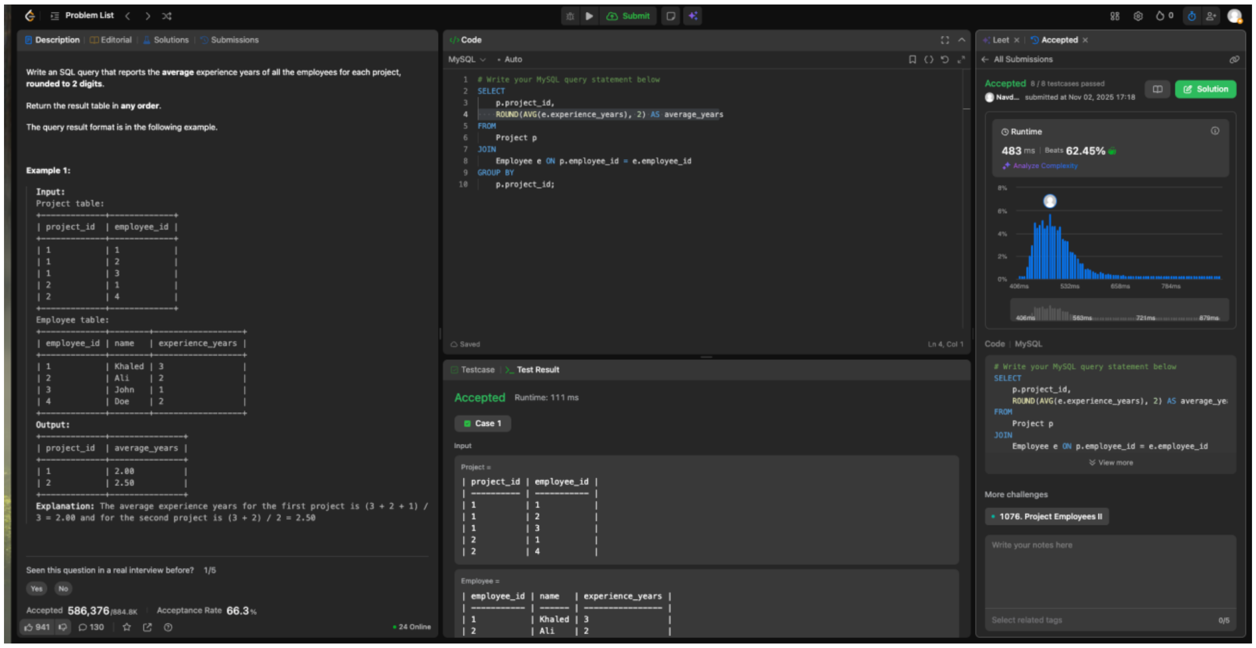
Memory 69.33 MB | Beats 18.32%

Code | Pandas

```python
import pandas as pd

def gameplay_analysis(activity: pd.DataFrame) -> pd

    activity['event_date'] = pd.to_datetime(activit

    first_login = activity.groupby('player_id')['ev
    first_login = first_login.rename(columns={'even
```

Q8)**1075. Project Employees I**

SQL Solution:



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**.

The query result format is in the following example.

Example 1:

```
Input:
Project table:
+-------------+-------------+
| project_id  | employee_id |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 1           |
| 2           | 4           |
+-------------+-------------+
Employee table:
+-------------+--------+------------------+
| employee_id | name   | experience_years |
+-------------+--------+------------------+
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
| 3           | John   | 1                |
| 4           | Doe    | 2                |
+-------------+--------+------------------+
Output:
+-------------+---------------+
| project_id  | average_years |
+-------------+---------------+
| 1           | 2.00          |
| 2           | 2.50          |
+-------------+---------------+
Explanation: The average experience years for the first project is (3 + 2 + 1) /
3 = 2.00 and for the second project is (3 + 2) / 2 = 2.50
```

Seen this question in a real interview before? 1/5

Accepted 586,376/884.8K | Acceptance Rate 66.3%

```sql
# Write your MySQL query statement below
SELECT
    p.project_id,
    ROUND(AVG(e.experience_years), 2) AS average_years
FROM
    Project p
JOIN
    Employee e ON p.employee_id = e.employee_id
GROUP BY
    p.project_id;
```

Accepted Runtime: 111 ms

Case 1

Input

```
Project =
| project_id | employee_id |
| ---------- | ----------- |
| 1          | 1           |
| 1          | 2           |
| 1          | 3           |
| 2          | 1           |
| 2          | 4           |

Employee =
| employee_id | name   | experience_years |
| ----------- | ------ | ---------------- |
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
```

Accepted 8 / 8 testcases passed

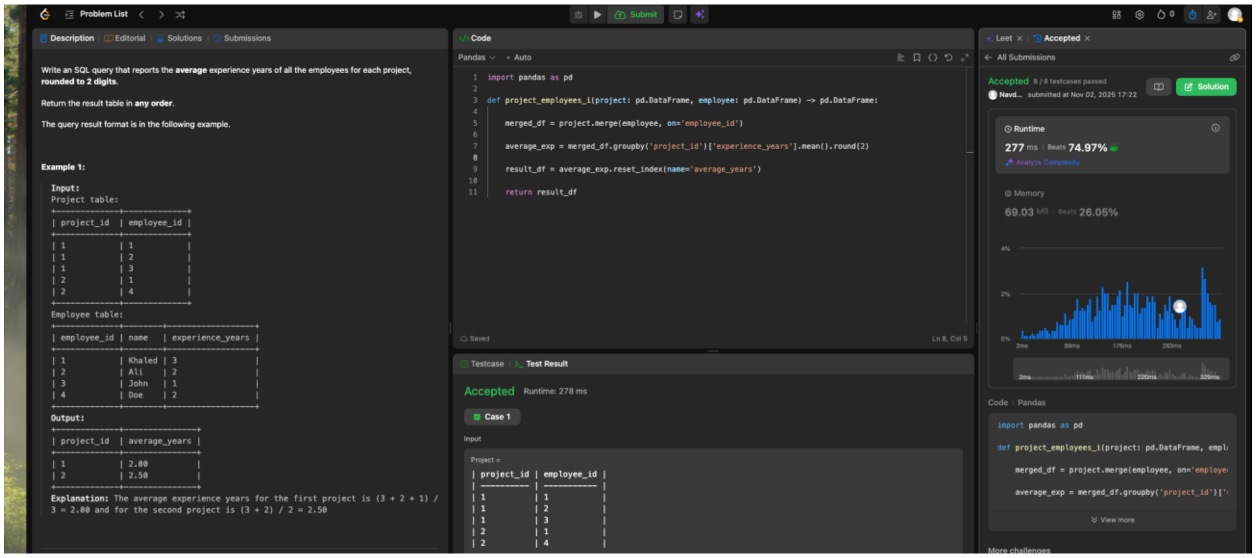
Navd... submitted at Nov 02, 2025 17:18

Runtime 483 ms | Beats 62.45%

Code | MySQL

```sql
# Write your MySQL query statement below
SELECT
    p.project_id,
    ROUND(AVG(e.experience_years), 2) AS average_ye
FROM
    Project p
JOIN
    Employee e ON p.employee_id = e.employee_id
```

Pandas Solution:



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**.

The query result format is in the following example.

Example 1:

```
Input:
Project table:
+-------------+-------------+
| project_id  | employee_id |
+-------------+-------------+
| 1           | 1           |
| 1           | 2           |
| 1           | 3           |
| 2           | 1           |
| 2           | 4           |
+-------------+-------------+
Employee table:
+-------------+--------+------------------+
| employee_id | name   | experience_years |
+-------------+--------+------------------+
| 1           | Khaled | 3                |
| 2           | Ali    | 2                |
| 3           | John   | 1                |
| 4           | Doe    | 2                |
+-------------+--------+------------------+
Output:
+-------------+---------------+
| project_id  | average_years |
+-------------+---------------+
| 1           | 2.00          |
| 2           | 2.50          |
+-------------+---------------+
Explanation: The average experience years for the first project is (3 + 2 + 1) /
3 = 2.00 and for the second project is (3 + 2) / 2 = 2.50
```

```python
import pandas as pd

def project_employees_i(project: pd.DataFrame, employee: pd.DataFrame) -> pd.DataFrame:

    merged_df = project.merge(employee, on='employee_id')

    average_exp = merged_df.groupby('project_id')['experience_years'].mean().round(2)

    result_df = average_exp.reset_index(name='average_years')

    return result_df
```

Accepted Runtime: 278 ms

Case 1

Input

Project =

```
| project_id | employee_id |
| ---------- | ----------- |
| 1          | 1           |
| 1          | 2           |
| 1          | 3           |
| 2          | 1           |
| 2          | 4           |
```

Accepted 8 / 8 testcases passed

Navd... submitted at Nov 02, 2025 17:22

Runtime

277 ms | Beats 74.97%

Memory

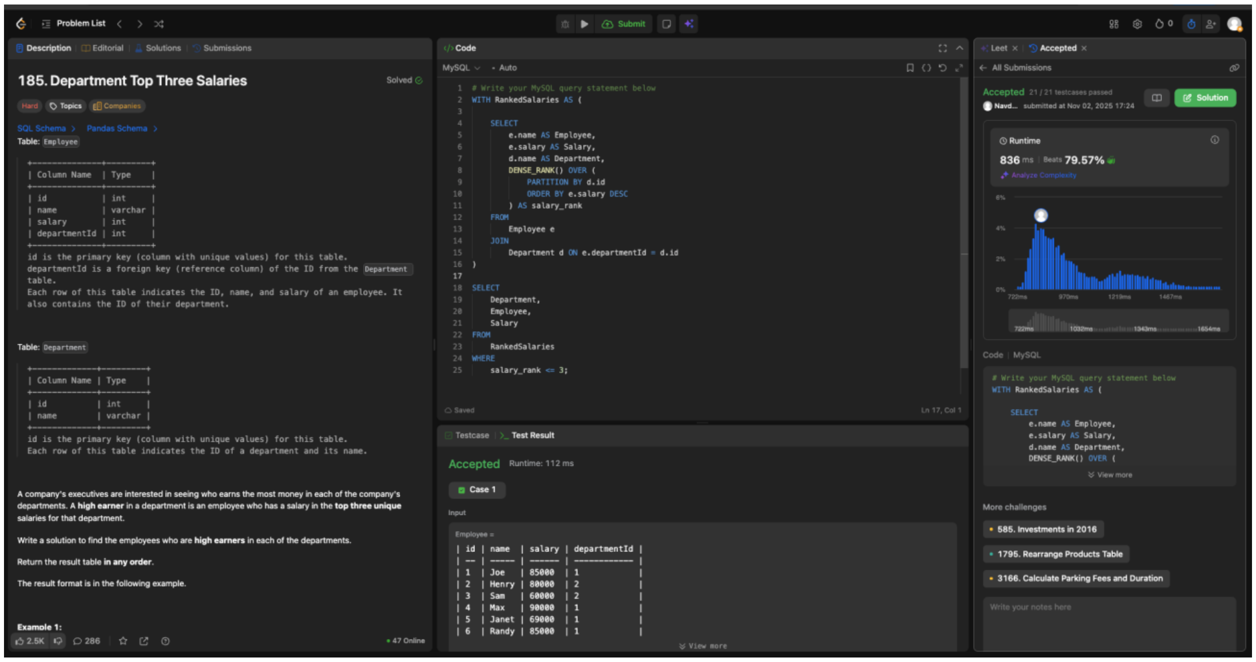
69.03 MB | Beats 26.05%

Code | Pandas

```python
import pandas as pd

def project_employees_i(project: pd.DataFrame, empl
    merged_df = project.merge(employee, on='employe
    average_exp = merged_df.groupby('project_id')['
```

Q9)**185. Department Top Three Salaries**

SQL Solution:



185. Department Top Three Salaries

Table: Employee

```
+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| id           | int     |
| name         | varchar |
| salary       | int     |
| departmentId | int     |
+--------------+---------+
id is the primary key (column with unique values) for this table.
departmentId is a foreign key (reference column) of the ID from the Department
table.
Each row of this table indicates the ID, name, and salary of an employee. It
also contains the ID of their department.
```

Table: Department

```
+-------------+---------+
| 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 of a department and its name.
```

A company's executives are interested in seeing who earns the most money in each of the company's departments. A **high earner** in a department is an employee who has a salary in the **top three unique** salaries for that department.

Write a solution to find the employees who are **high earners** in each of the departments.

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

The result format is in the following example.

Example 1:

```sql
# Write your MySQL query statement below
WITH RankedSalaries AS (

    SELECT
        e.name AS Employee,
        e.salary AS Salary,
        d.name AS Department,
        DENSE_RANK() OVER (
            PARTITION BY d.id
            ORDER BY e.salary DESC
        ) AS salary_rank
    FROM
        Employee e
    JOIN
        Department d ON e.departmentId = d.id
)

SELECT
    Department,
    Employee,
    Salary
FROM
    RankedSalaries
WHERE
    salary_rank <= 3;
```

Accepted Runtime: 112 ms

Case 1

Input

Employee =

```
| id | name  | salary | departmentId |
| -- | ----- | ------ | ------------ |
| 1  | Joe   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
```

Accepted 21 / 21 testcases passed

Navd... submitted at Nov 02, 2025 17:24

Runtime

836 ms | Beats 79.57%

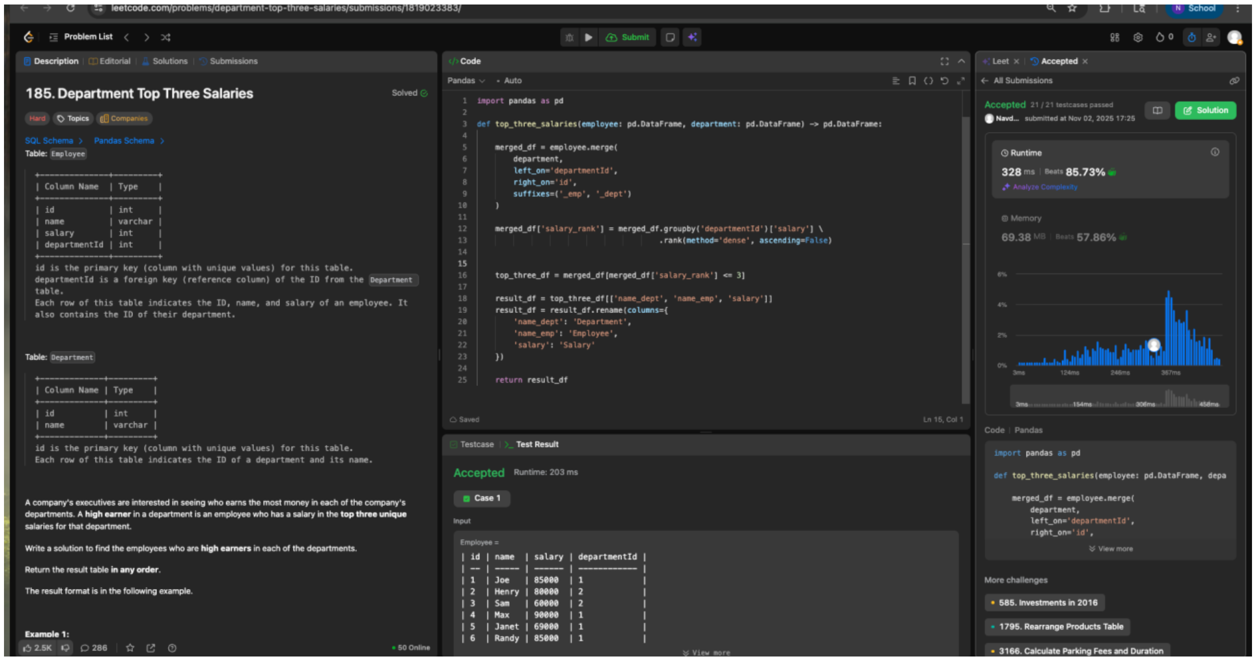
Code | MySQL

```sql
# Write your MySQL query statement below
WITH RankedSalaries AS (

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

More challenges

- 585. Investments in 2016
- 1795. Rearrange Products Table
- 3166. Calculate Parking Fees and Duration

Pandas Solution:



185. Department Top Three Salaries

Table: `Employee`

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| departmentId | int    |
+-------------+---------+
id is the primary key (column with unique values) for this table.
departmentId is a foreign key (reference column) of the ID from the Department
table.
Each row of this table indicates the ID, name, and salary of an employee. It
also contains the ID of their department.
```

Table: `Department`

```
+-------------+---------+
| 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 of a department and its name.
```

A company's executives are interested in seeing who earns the most money in each of the company's departments. A **high earner** in a department is an employee who has a salary in the **top three unique** salaries for that department.

Write a solution to find the employees who are **high earners** in each of the departments.

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

The result format is in the following example.

Example 1:

```python
import pandas as pd

def top_three_salaries(employee: pd.DataFrame, department: pd.DataFrame) -> pd.DataFrame:

    merged_df = employee.merge(
        department,
        left_on='departmentId',
        right_on='id',
        suffixes=('_emp', '_dept')
    )

    merged_df['salary_rank'] = merged_df.groupby('departmentId')['salary'] \
                                        .rank(method='dense', ascending=False)


    top_three_df = merged_df[merged_df['salary_rank'] <= 3]

    result_df = top_three_df[['name_dept', 'name_emp', 'salary']]
    result_df = result_df.rename(columns={
        'name_dept': 'Department',
        'name_emp': 'Employee',
        'salary': 'Salary'
    })

    return result_df
```

Accepted Runtime: 203 ms

Input

```
Employee =
| id | name  | salary | departmentId |
| -- | ----- | ------ | ------------ |
| 1  | Joe   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
```

Accepted 21 / 21 testcases passed
Navd... submitted at Nov 02, 2025 17:25

Runtime
328 ms | Beats 85.73%

Memory
69.38 MB | Beats 57.86%

Code | Pandas

```python
import pandas as pd

def top_three_salaries(employee: pd.DataFrame, depa

    merged_df = employee.merge(
        department,
        left_on='departmentId',
        right_on='id',
```

More challenges

- 585. Investments in 2016
- 1795. Rearrange Products Table
- 3166. Calculate Parking Fees and Duration