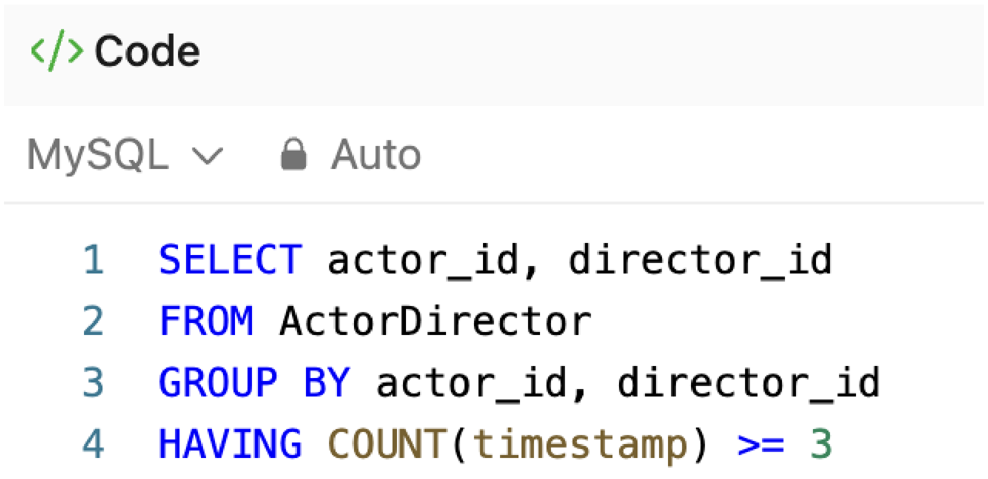
Mariia Onokhina

1050. Actors and Directors Who Cooperated At Least Three Times

[Actors and Directors Who Cooperated At Least Three Times - LeetCode]()

MySQL:

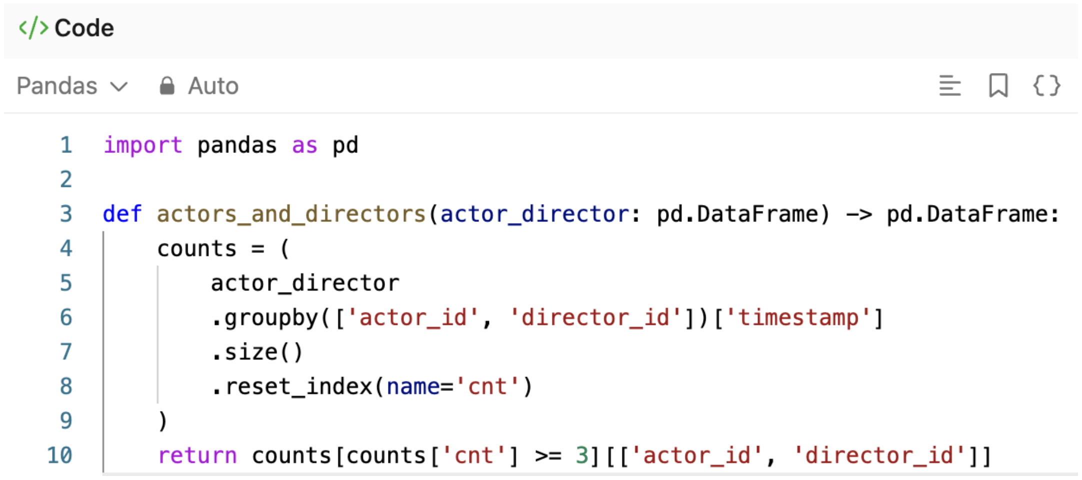
https://leetcode.com/problems/actors-and-directors-who-cooperated-at-least-three-times/submissions/1828996050



```sql
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id, director_id
HAVING COUNT(timestamp) >= 3
```

Pandas:

https://leetcode.com/problems/actors-and-directors-who-cooperated-at-least-three-times/submissions/1829005045



```python
import pandas as pd

def actors_and_directors(actor_director: pd.DataFrame) -> pd.DataFrame:
    counts = (
        actor_director
        .groupby(['actor_id', 'director_id'])['timestamp']
        .size()
        .reset_index(name='cnt')
    )
    return counts[counts['cnt'] >= 3][['actor_id', 'director_id']]
```

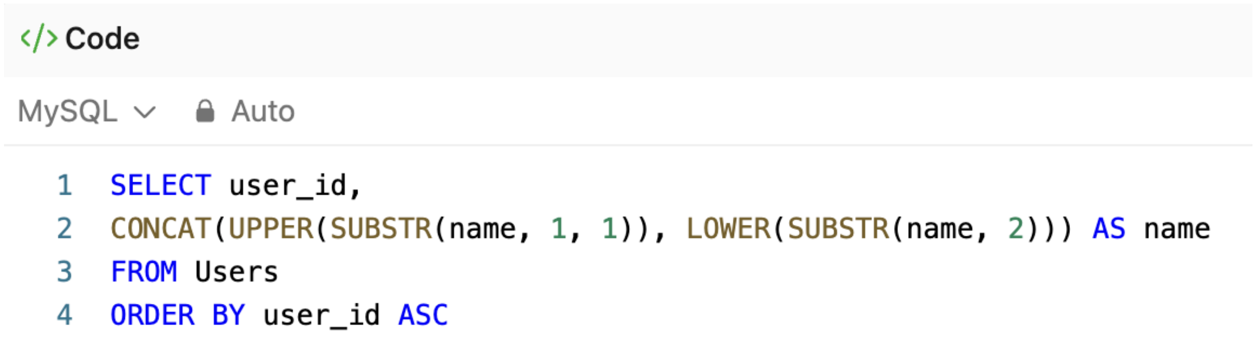
Mariia Onokhina

1667. Fix Names in a Table

[Fix Names in a Table - LeetCode]

MySQL:

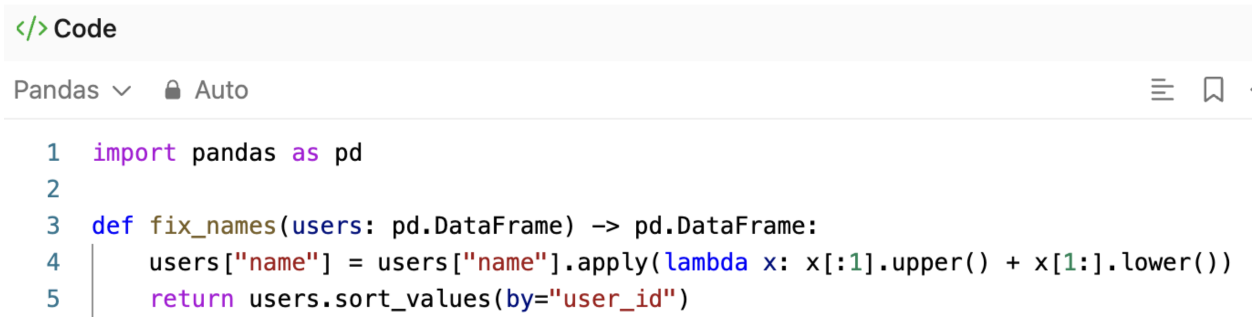
https://leetcode.com/problems/fix-names-in-a-table/submissions/1829009997



```sql
SELECT user_id,
CONCAT(UPPER(SUBSTR(name, 1, 1)), LOWER(SUBSTR(name, 2))) AS name
FROM Users
ORDER BY user_id ASC
```

Pandas:

https://leetcode.com/problems/fix-names-in-a-table/submissions/1829013204



```python
import pandas as pd

def fix_names(users: pd.DataFrame) -> pd.DataFrame:
    users["name"] = users["name"].apply(lambda x: x[:1].upper() + x[1:].lower())
    return users.sort_values(by="user_id")
```

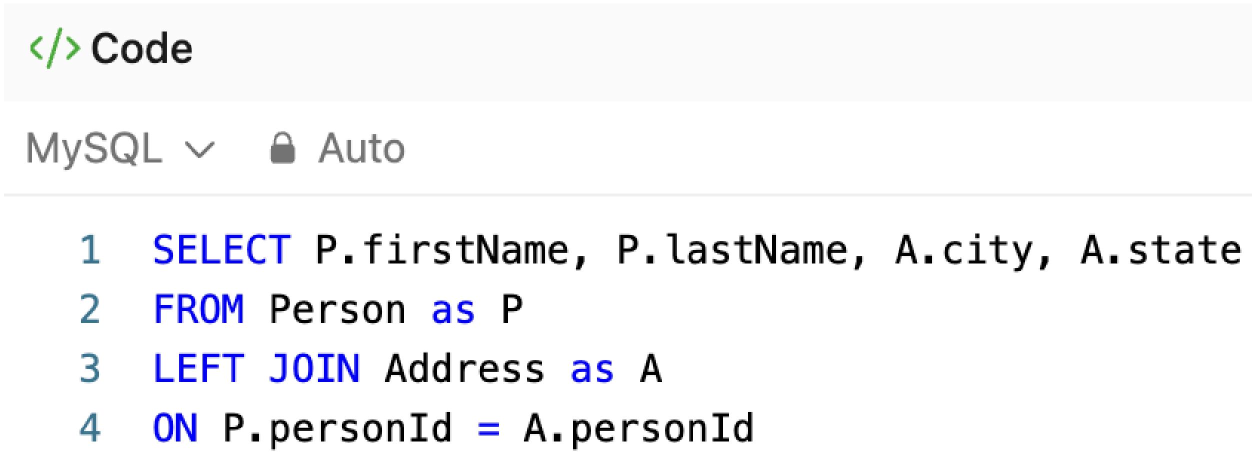
Mariia Onokhina

175. Combine Two Tables

Combine Two Tables - LeetCode

MySQL:

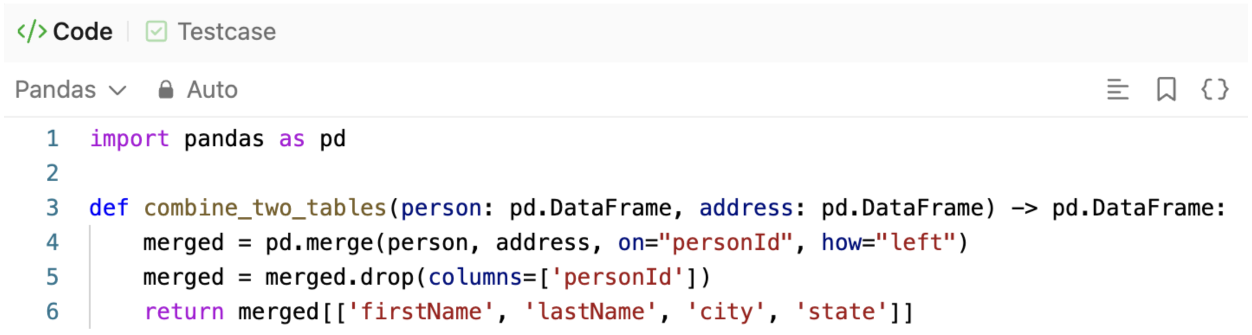
https://leetcode.com/problems/combine-two-tables/submissions/1829016234



```sql
SELECT P.firstName, P.lastName, A.city, A.state
FROM Person as P
LEFT JOIN Address as A
ON P.personId = A.personId
```

Pandas:

https://leetcode.com/problems/combine-two-tables/submissions/1829022017



```python
import pandas as pd

def combine_two_tables(person: pd.DataFrame, address: pd.DataFrame) -> pd.DataFrame:
    merged = pd.merge(person, address, on="personId", how="left")
    merged = merged.drop(columns=['personId'])
    return merged[['firstName', 'lastName', 'city', 'state']]
```

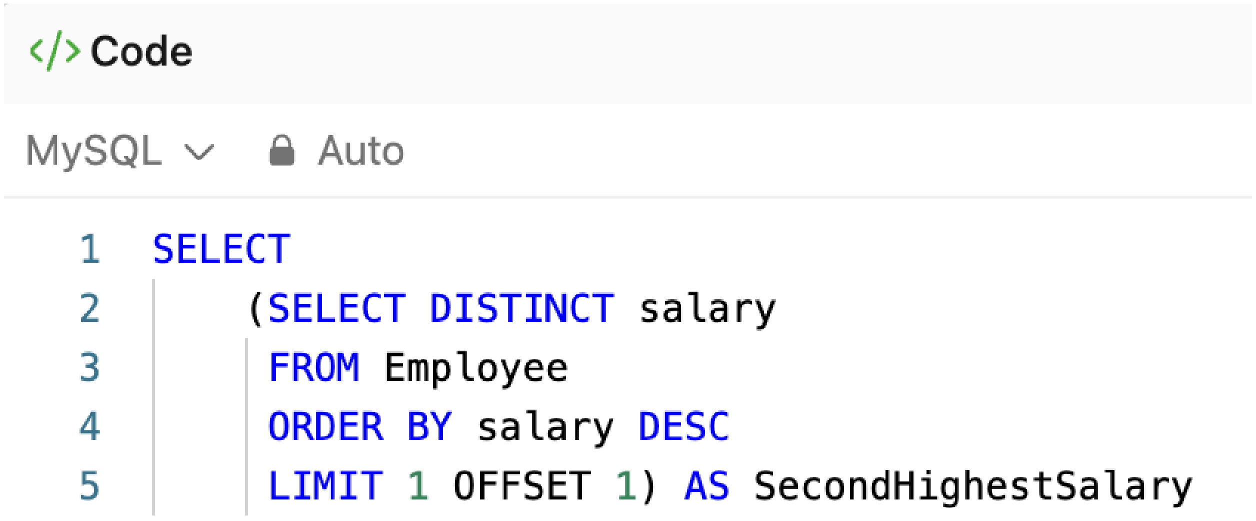
Mariia Onokhina

176. Second Highest Salary

[Second Highest Salary - LeetCode]

MySQL:

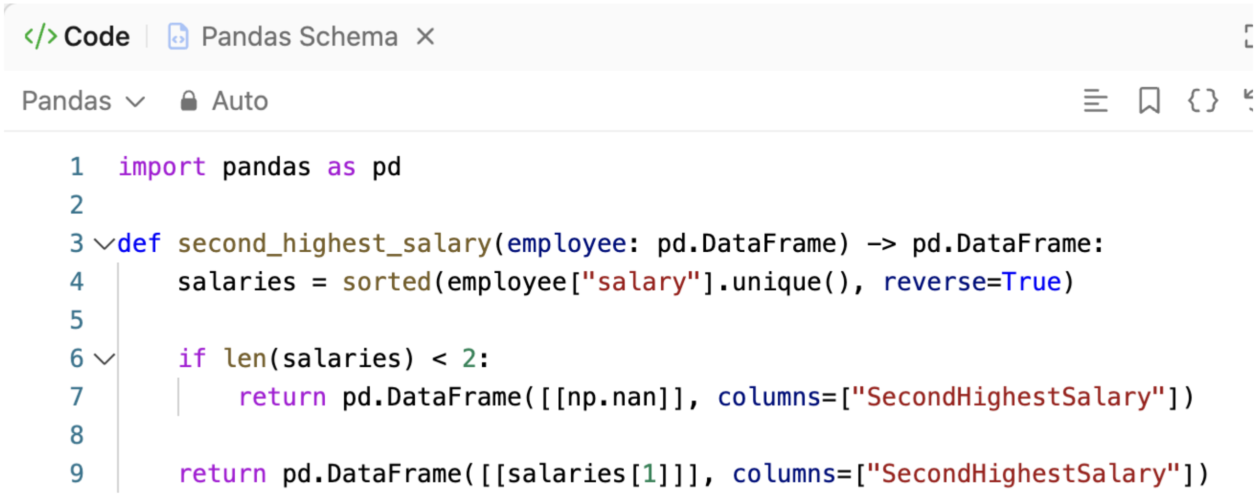
https://leetcode.com/problems/second-highest-salary/submissions/1829025158



```sql
SELECT
    (SELECT DISTINCT salary
     FROM Employee
     ORDER BY salary DESC
     LIMIT 1 OFFSET 1) AS SecondHighestSalary
```

Pandas:

https://leetcode.com/problems/second-highest-salary/submissions/1829028435



```python
import pandas as pd

def second_highest_salary(employee: pd.DataFrame) -> pd.DataFrame:
    salaries = sorted(employee["salary"].unique(), reverse=True)

    if len(salaries) < 2:
        return pd.DataFrame([[np.nan]], columns=["SecondHighestSalary"])

    return pd.DataFrame([[salaries[1]]], columns=["SecondHighestSalary"])
```

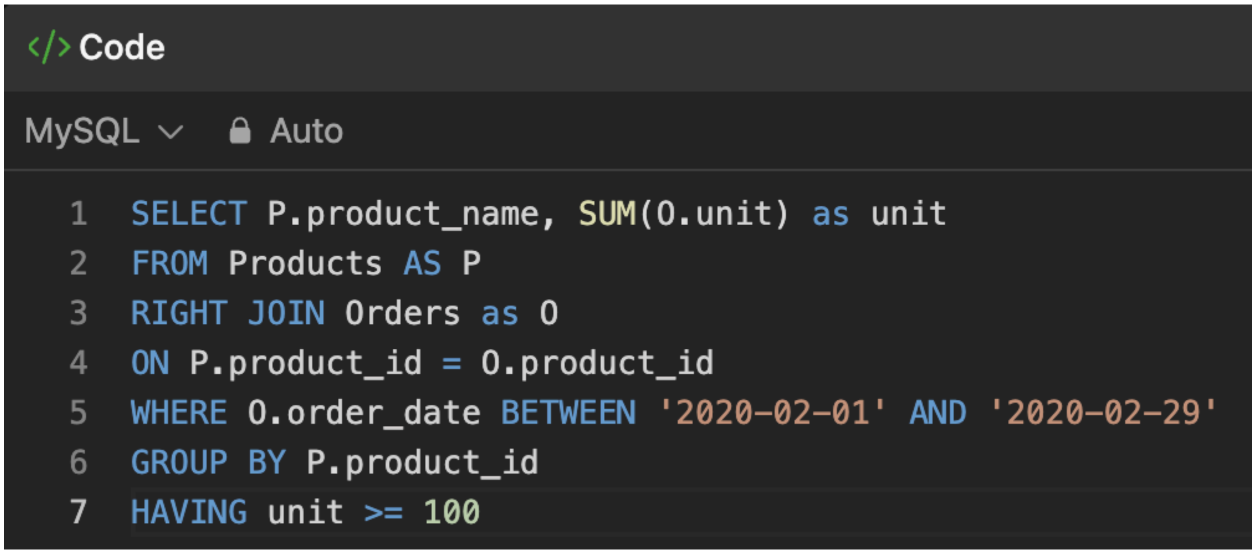
Mariia Onokhina

1327. List the Products Ordered in a Period

https://leetcode.com/problems/list-the-products-ordered-in-a-period/

MySQL:

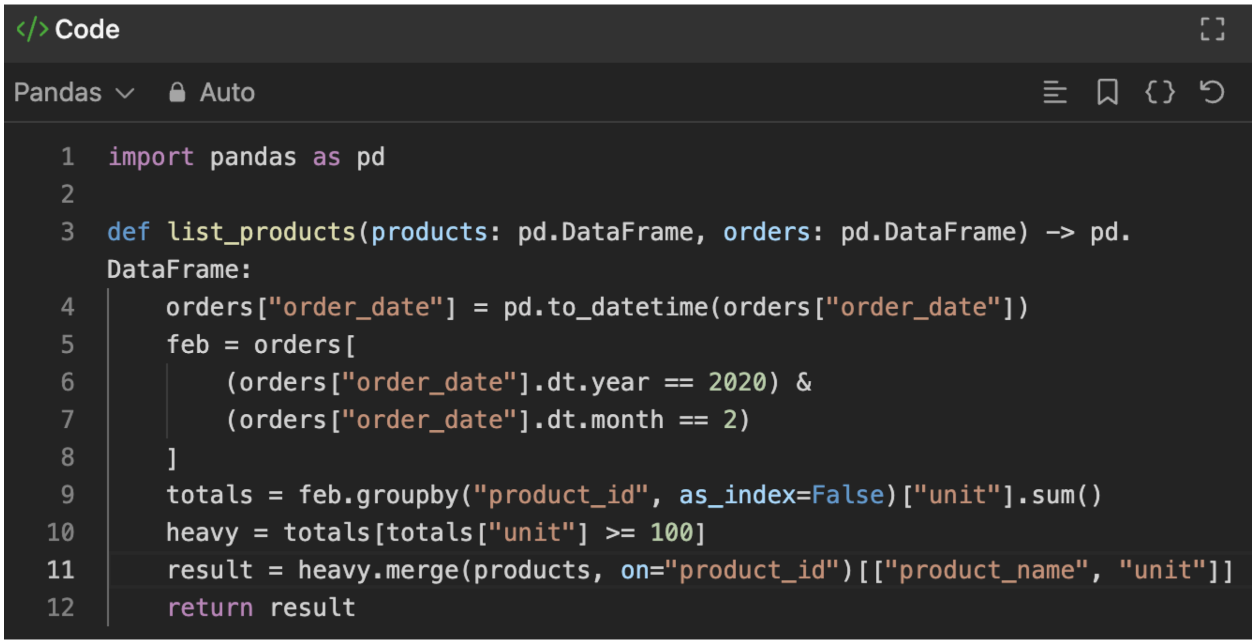
https://leetcode.com/problems/list-the-products-ordered-in-a-period/submissions/1829152506



```sql
SELECT P.product_name, SUM(O.unit) as unit
FROM Products AS P
RIGHT JOIN Orders as O
ON P.product_id = O.product_id
WHERE O.order_date BETWEEN '2020-02-01' AND '2020-02-29'
GROUP BY P.product_id
HAVING unit >= 100
```

Pandas:

https://leetcode.com/problems/list-the-products-ordered-in-a-period/submissions/1829282672



```python
import pandas as pd

def list_products(products: pd.DataFrame, orders: pd.DataFrame) -> pd.DataFrame:
    orders["order_date"] = pd.to_datetime(orders["order_date"])
    feb = orders[
        (orders["order_date"].dt.year == 2020) &
        (orders["order_date"].dt.month == 2)
    ]
    totals = feb.groupby("product_id", as_index=False)["unit"].sum()
    heavy = totals[totals["unit"] >= 100]
    result = heavy.merge(products, on="product_id")[["product_name", "unit"]]
    return result
```

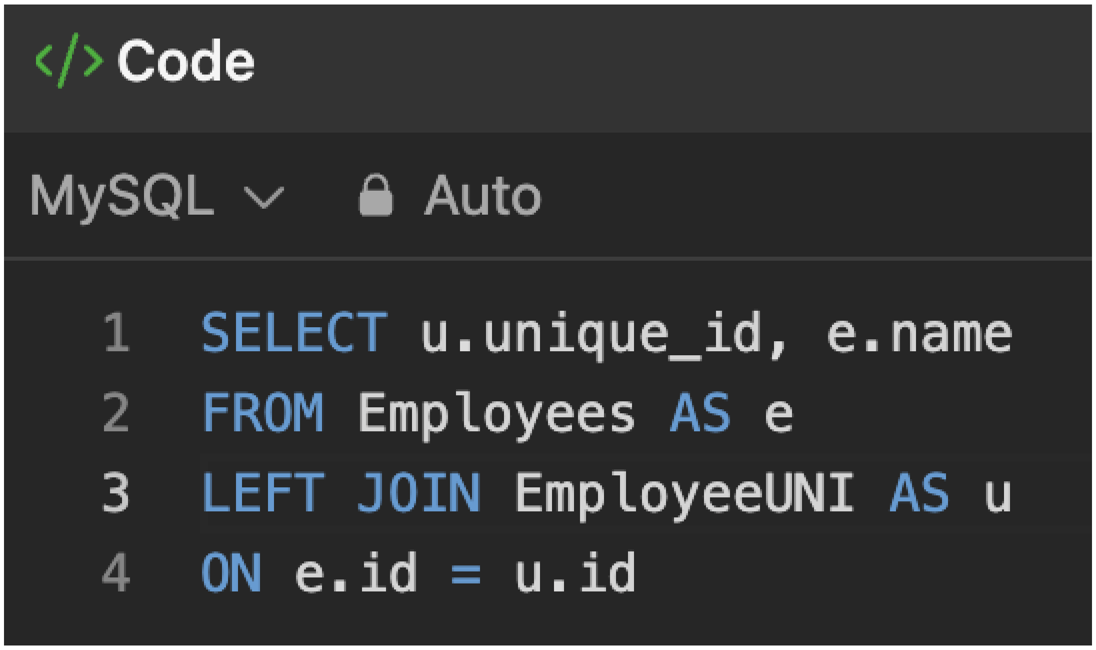
Mariia Onokhina

1378. Replace Employee ID With The Unique Identifier

https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/

MySQL:

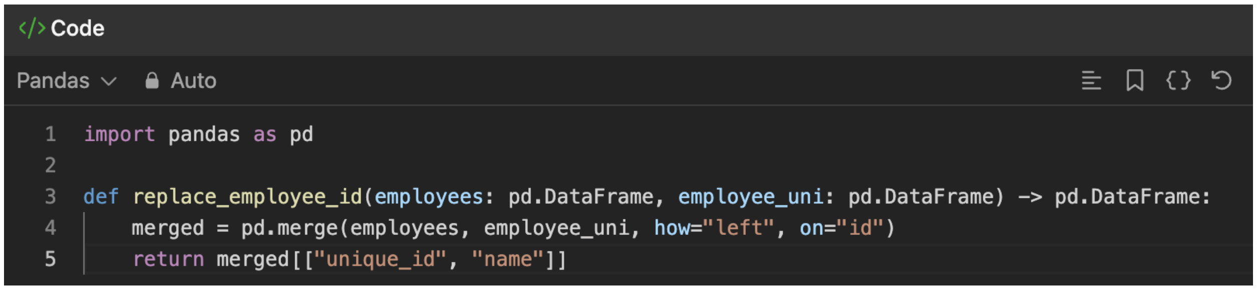
https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/submissions/1829284645



```sql
SELECT u.unique_id, e.name
FROM Employees AS e
LEFT JOIN EmployeeUNI AS u
ON e.id = u.id
```

Pandas:

https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/submissions/1829285997



```python
import pandas as pd

def replace_employee_id(employees: pd.DataFrame, employee_uni: pd.DataFrame) -> pd.DataFrame:
    merged = pd.merge(employees, employee_uni, how="left", on="id")
    return merged[["unique_id", "name"]]
```

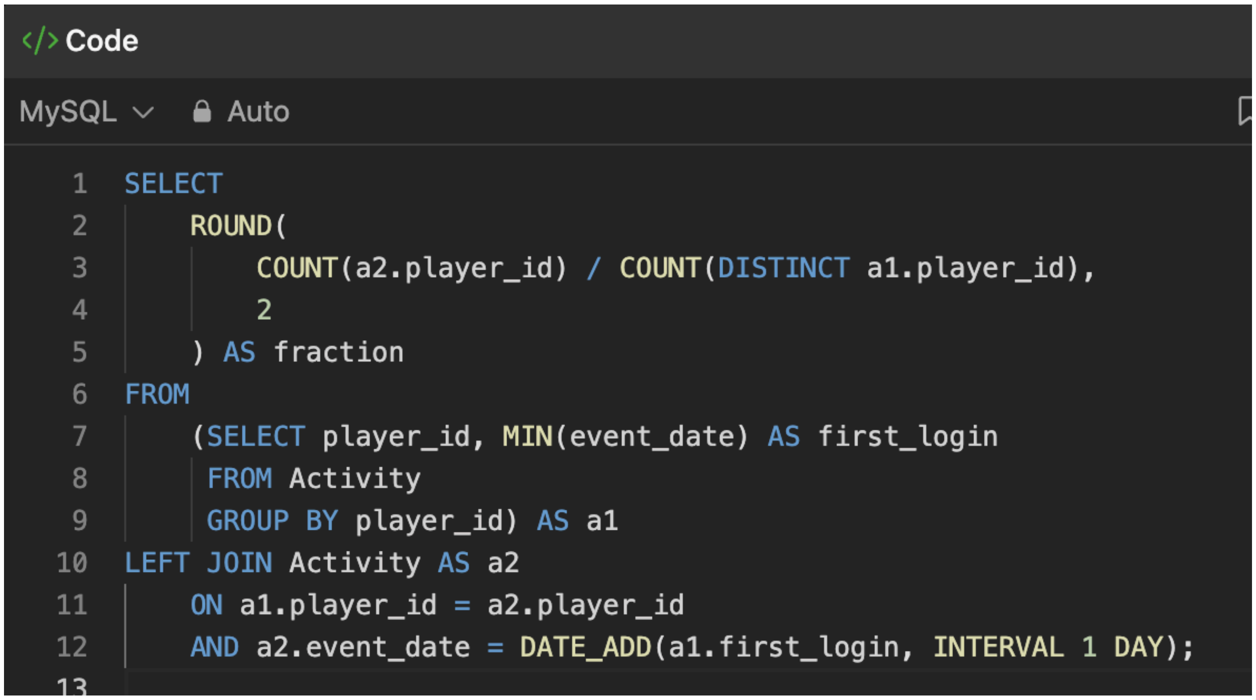
Mariia Onokhina

550. Game Play Analysis IV

https://leetcode.com/problems/game-play-analysis-iv/

MySQL:

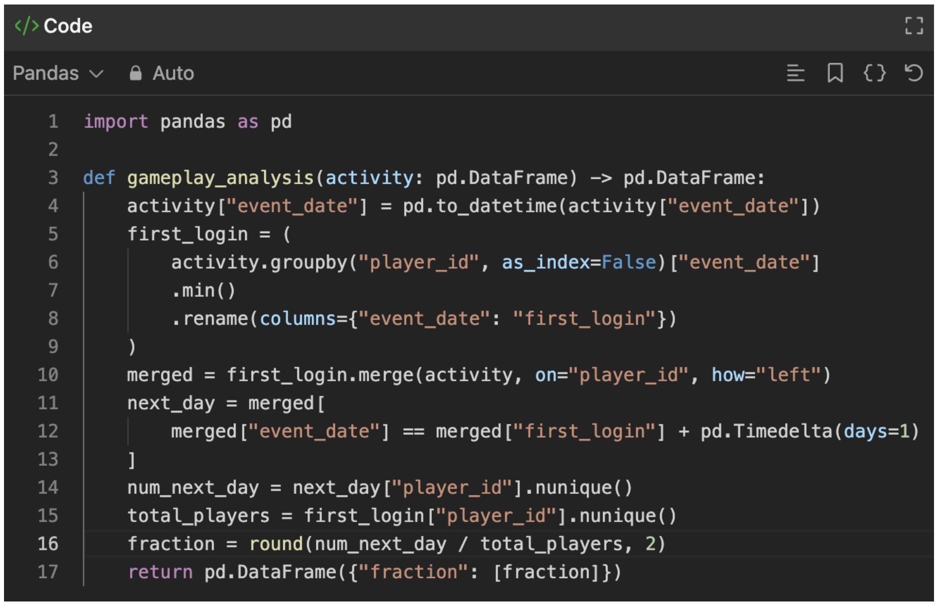
https://leetcode.com/problems/game-play-analysis-iv/submissions/1829287170



```sql
SELECT
    ROUND(
        COUNT(a2.player_id) / COUNT(DISTINCT a1.player_id),
        2
    ) AS fraction
FROM
    (SELECT player_id, MIN(event_date) AS first_login
     FROM Activity
     GROUP BY player_id) AS a1
LEFT JOIN Activity AS a2
    ON a1.player_id = a2.player_id
    AND a2.event_date = DATE_ADD(a1.first_login, INTERVAL 1 DAY);
```

Pandas:

https://leetcode.com/problems/game-play-analysis-iv/submissions/1829287977



```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", as_index=False)["event_date"]
        .min()
        .rename(columns={"event_date": "first_login"})
    )
    merged = first_login.merge(activity, on="player_id", how="left")
    next_day = merged[
        merged["event_date"] == merged["first_login"] + pd.Timedelta(days=1)
    ]
    num_next_day = next_day["player_id"].nunique()
    total_players = first_login["player_id"].nunique()
    fraction = round(num_next_day / total_players, 2)
    return pd.DataFrame({"fraction": [fraction]})
```

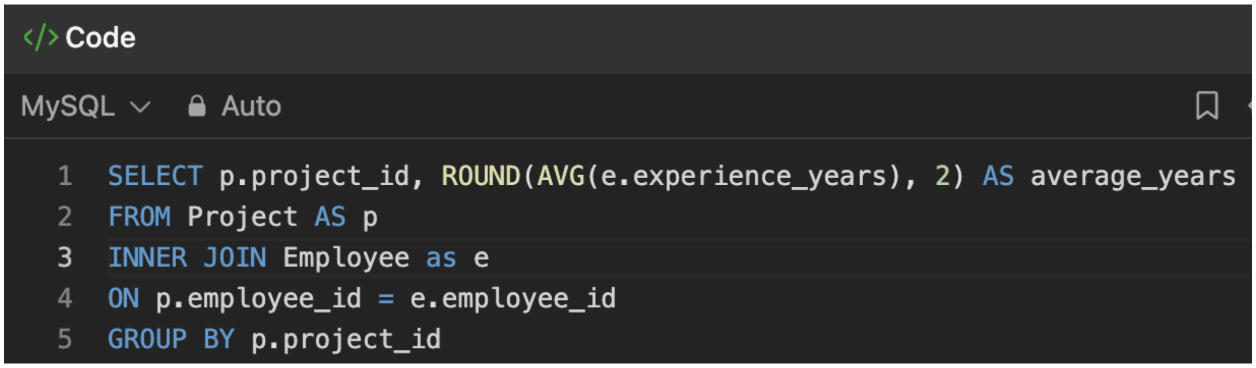
Mariia Onokhina

1075. Project Employees I

https://leetcode.com/problems/project-employees-i/

MySQL:

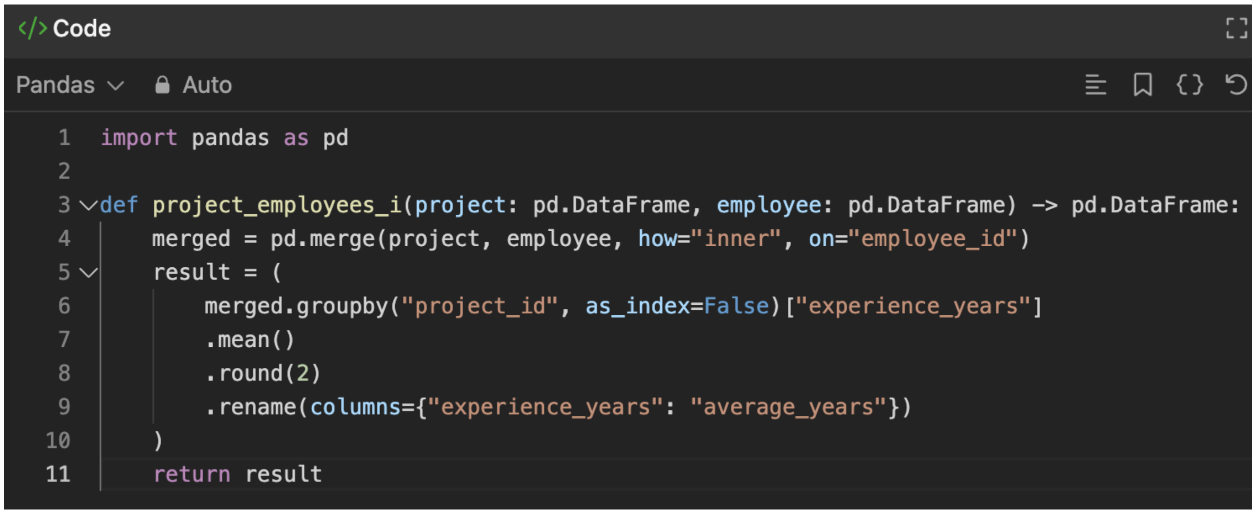
https://leetcode.com/problems/project-employees-i/submissions/1829291158



```sql
SELECT p.project_id, ROUND(AVG(e.experience_years), 2) AS average_years
FROM Project AS p
INNER JOIN Employee as e
ON p.employee_id = e.employee_id
GROUP BY p.project_id
```

Pandas:

https://leetcode.com/problems/project-employees-i/submissions/1829293005



```python
import pandas as pd

def project_employees_i(project: pd.DataFrame, employee: pd.DataFrame) -> pd.DataFrame:
    merged = pd.merge(project, employee, how="inner", on="employee_id")
    result = (
        merged.groupby("project_id", as_index=False)["experience_years"]
        .mean()
        .round(2)
        .rename(columns={"experience_years": "average_years"})
    )
    return result
```

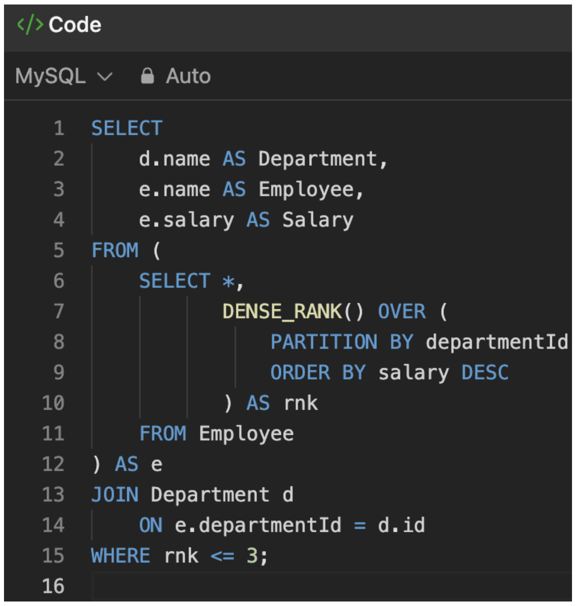
Mariia Onokhina

185. Department Top Three Salaries

https://leetcode.com/problems/department-top-three-salaries/

MySQL:

https://leetcode.com/problems/department-top-three-salaries/submissions/1829297370

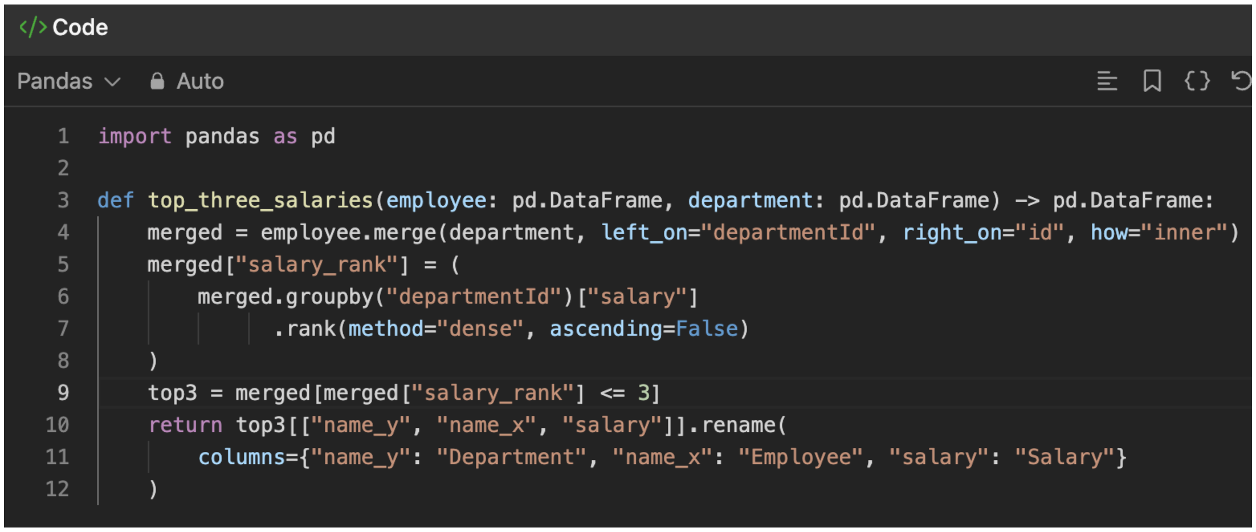


```sql
SELECT
    d.name AS Department,
    e.name AS Employee,
    e.salary AS Salary
FROM (
    SELECT *,
           DENSE_RANK() OVER (
               PARTITION BY departmentId
               ORDER BY salary DESC
           ) AS rnk
    FROM Employee
) AS e
JOIN Department d
    ON e.departmentId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

```

Pandas:

https://leetcode.com/problems/department-top-three-salaries/submissions/1829297921



```python
import pandas as pd

def top_three_salaries(employee: pd.DataFrame, department: pd.DataFrame) -> pd.DataFrame:
    merged = employee.merge(department, left_on="departmentId", right_on="id", how="inner")
    merged["salary_rank"] = (
        merged.groupby("departmentId")["salary"]
              .rank(method="dense", ascending=False)
    )
    top3 = merged[merged["salary_rank"] <= 3]
    return top3[["name_y", "name_x", "salary"]].rename(
        columns={"name_y": "Department", "name_x": "Employee", "salary": "Salary"}
    )
```