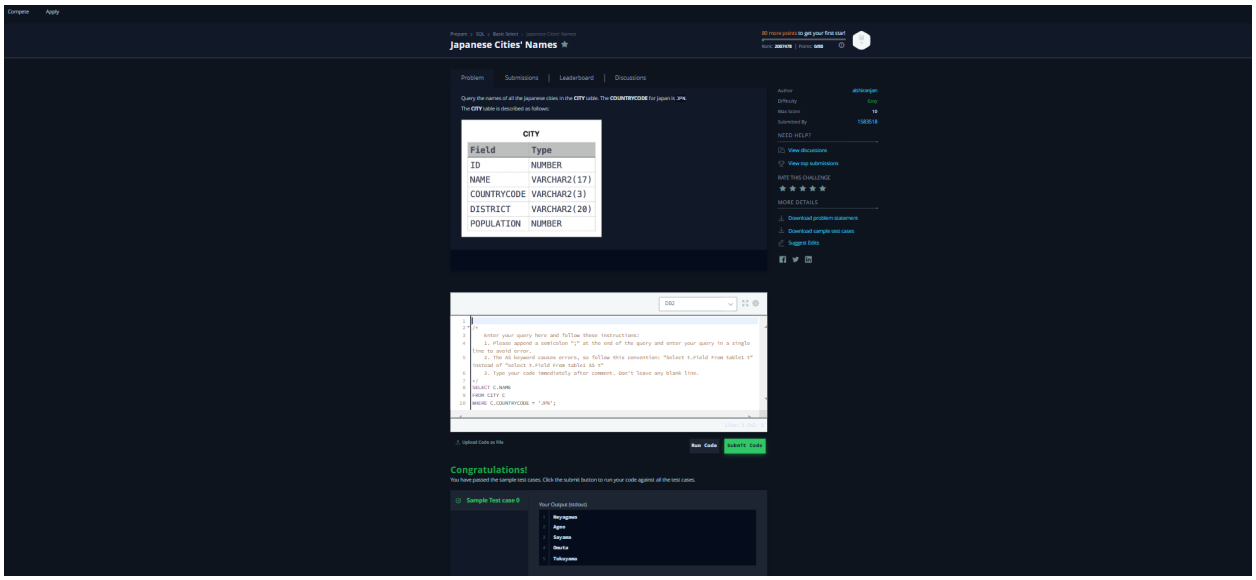
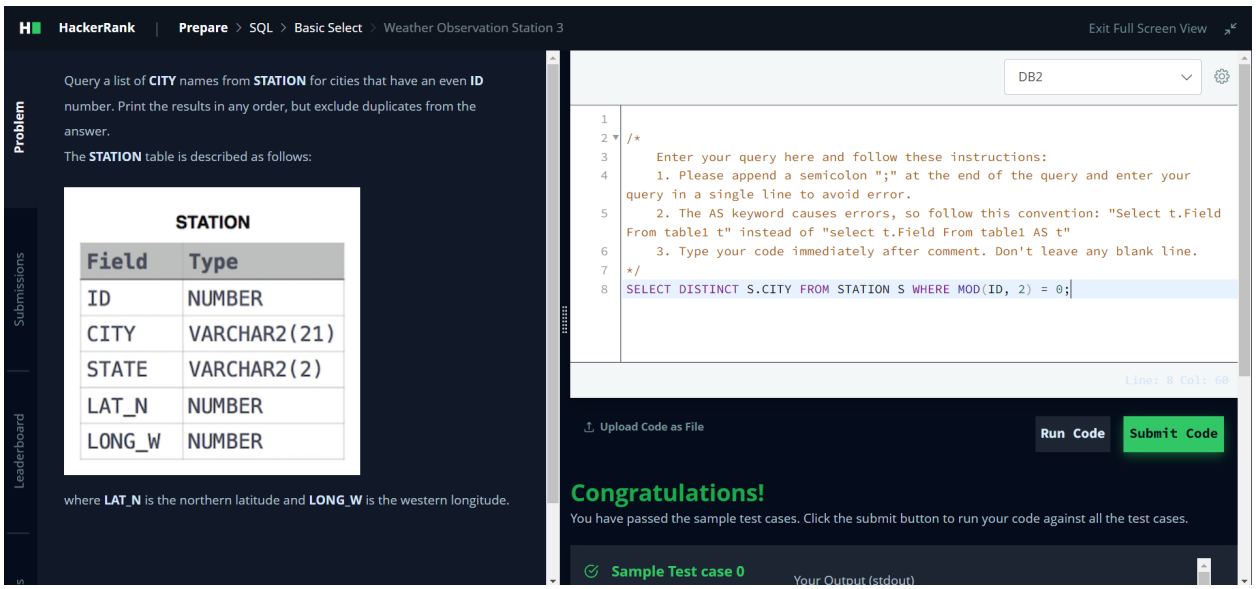
1. [Japanese Cities' Names]

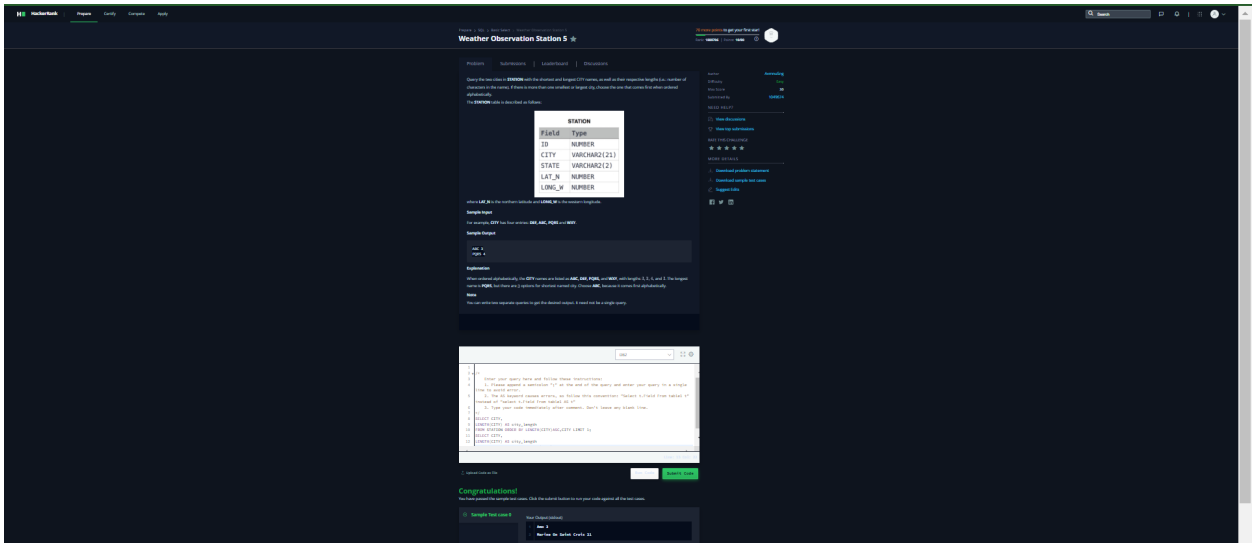
QUERY:
SELECT C.NAME FROM CITY C WHERE C.COUNTRYCODE = ‘JPN’;



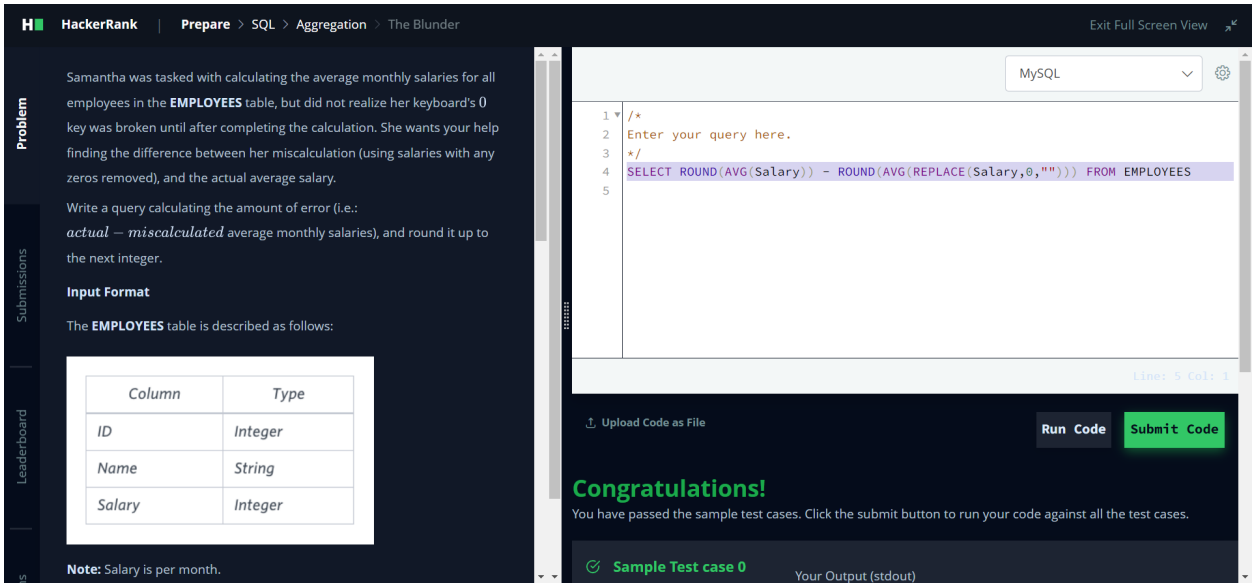
2. [Weather Observation Station 3]
QUERY:
SELECT DISTINCT S.CITY FROM STATION S WHERE MOD(ID,2) = 0;



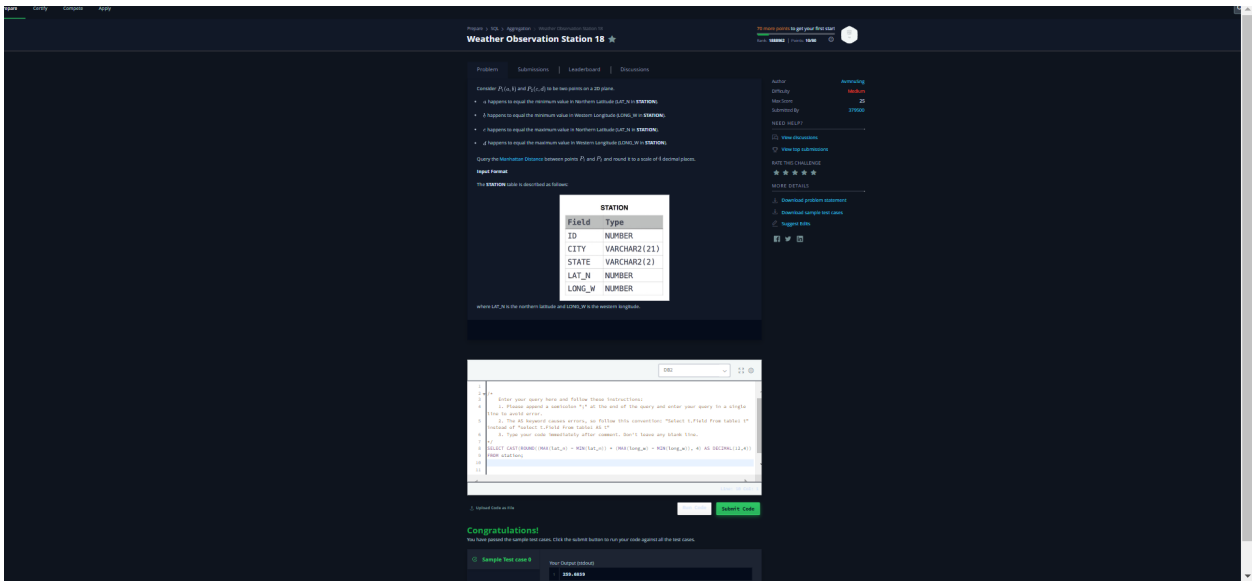
3. Weather Observation Station 5
   QUERY:
   SELECT CITY,
   LENGTH(CITY) AS city_length
   FROM STATION ORDER BY LENGTH(CITY)ASC,CITY LIMIT 1;
   SELECT CITY,
   LENGTH(CITY) AS city_length
   FROM STATION ORDER BY LENGTH(CITY)DESC, CITY LIMIT 1;



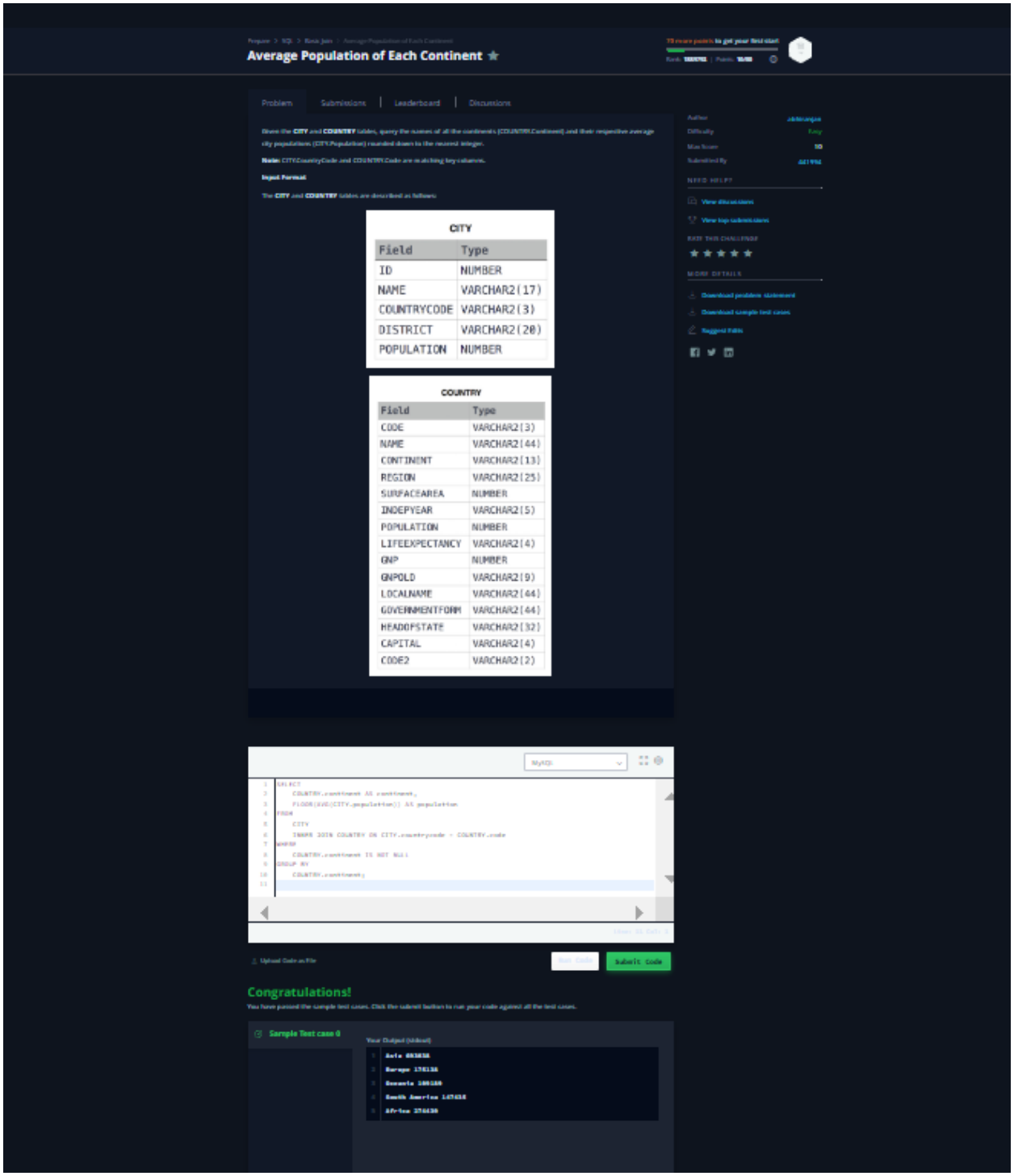
4. The Blunder
   QUERY:
   SELECT ROUND(AVG(Salary)) - ROUND(AVG(REPLACE(Salary,0,""))) FROM EMPLOYEES



5. [Weather Observation Station 1]
   QUERY:
   SELECT CAST(ROUND((MAX(lat_n) - MIN(lat_n)) + (MAX(long_w) - MIN(long_w)), 4)
   AS DECIMAL(12,4))
   FROM station;



6. [Average Population of Each Continent]
   QUERY:
   SELECT
   COUNTRY.continent AS continent,
   FLOOR(AVG(CITY.population)) AS population
   FROM
   CITY
   INNER JOIN COUNTRY ON CITY.countrycode = COUNTRY.code
   WHERE
   COUNTRY.continent IS NOT NULL
   GROUP BY
   COUNTRY.continent;


Average Population of Each Continent
Problem
Submissions
Leaderboard
Discussions
CITY
Field Type
ID NUMBER
NAME VARCHAR2(17)
COUNTRYCODE VARCHAR2(3)
DISTRICT VARCHAR2(20)
POPULATION NUMBER
COUNTRY
Field Type
CODE VARCHAR2(3)
NAME VARCHAR2(44)
CONTINENT VARCHAR2(13)
REGION VARCHAR2(25)
SURFACEAREA NUMBER
INDEPYEAR VARCHAR2(5)
POPULATION NUMBER
LIFEEXPECTANCY VARCHAR2(4)
GNP NUMBER
GNPOLD VARCHAR2(9)
LOCALNAME VARCHAR2(44)
GOVERNMENTFORM VARCHAR2(44)
HEADOFSTATE VARCHAR2(32)
CAPITAL VARCHAR2(4)
CODE2 VARCHAR2(2)
Submit Code
Congratulations!
Sample Test case 0

7. The PADS

QUERY:

```
SELECT
   CONCAT(NAME, '(', LEFT(OCCUPATION, 1), ')') AS formatted_name
FROM
   OCCUPATIONS
ORDER BY
   NAME;


SELECT
   CONCAT('There are a total of ', COUNT(*), ' ', LOWER(OCCUPATION), 's.') AS
formatted_name
FROM
   OCCUPATIONS
GROUP BY
   OCCUPATION
ORDER BY
   COUNT(*), OCCUPATION;
```

The **OCCUPATIONS** table is described as follows:

Column	Type
Name	*String*
Occupation	*String*

Occupation will only contain one of the following values: **Doctor**, **Professor**, **Singer** or **Actor**.

Sample Input

An **OCCUPATIONS** table that contains the following records:

Name	Occupation
Samantha	Doctor
Julia	Actor
Maria	Actor
Meera	Singer
Ashely	Professor
Ketty	Professor
Christeen	Professor
Jane	Actor
Jenny	Doctor
Priya	Singer

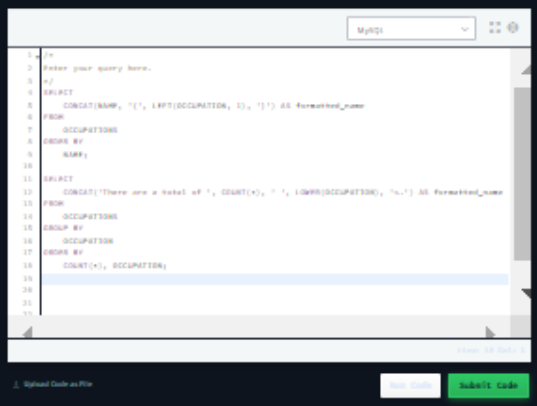
Sample Output

```
Ashely(P)
Christeen(P)
Jane(A)
Jenny(D)
Julia(A)
Ketty(P)
Maria(A)
Meera(S)
Priya(S)
Samantha(D)
There are a total of 3 doctors.
There are a total of 3 singers.
There are a total of 3 actors.
There are a total of 3 professors.
```

Explanation

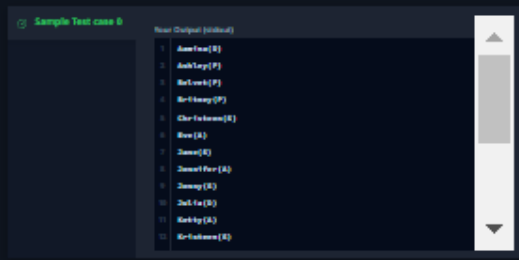
The results of the first query are formatted to the problem description's specifications.

The results of the second query are ascendingly ordered first by number of names corresponding to each profession ($3 \leq 3 < 4 < 5$), and then alphabetically by profession ($doctor < singer < actor < professor$).



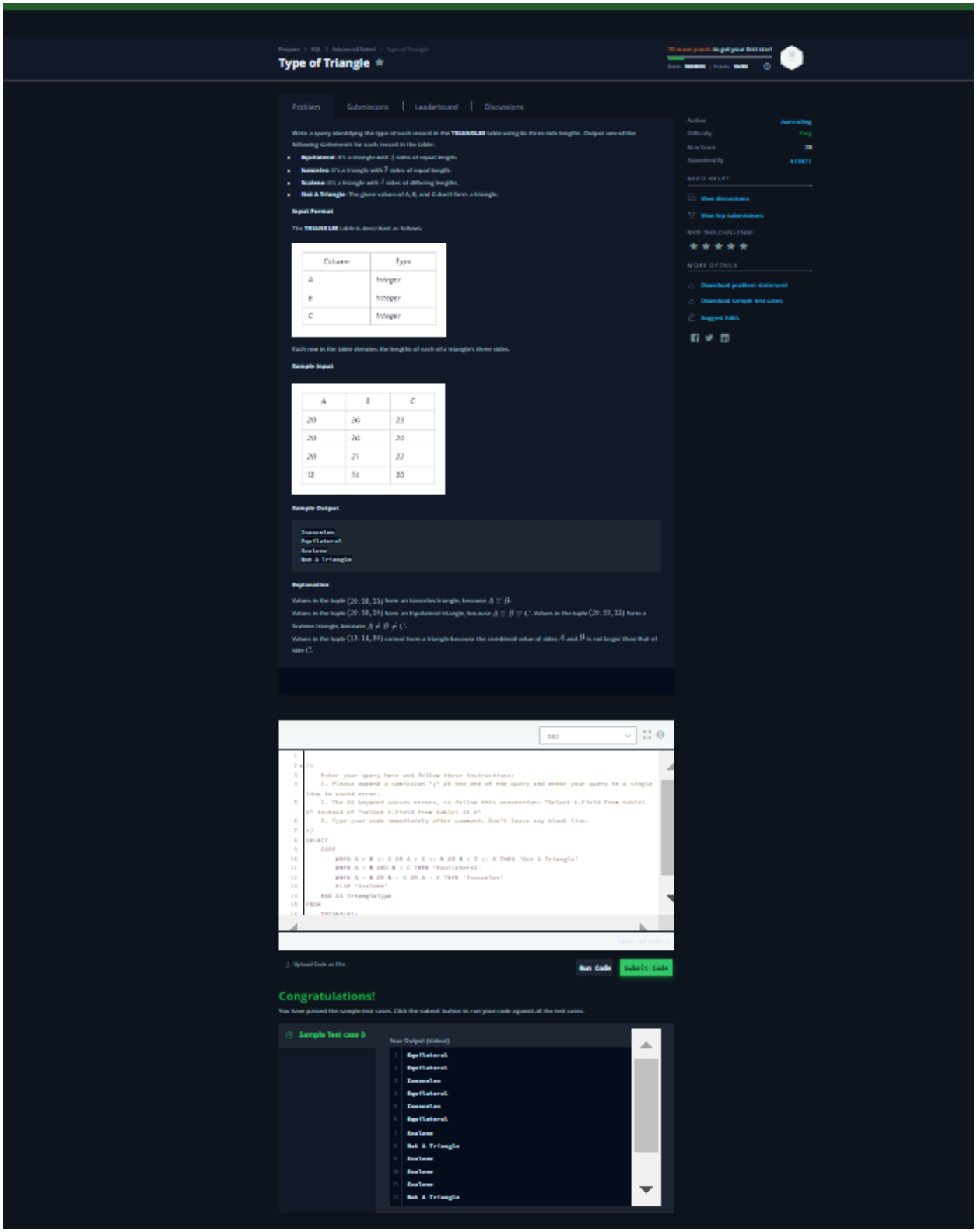
Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.



8. Type of Triangle (Use Case statement)
   QUERY:

```
SELECT
   CASE
      WHEN A + B <= C OR A + C <= B OR B + C <= A THEN 'Not A Triangle'
      WHEN A = B AND B = C THEN 'Equilateral'
      WHEN A = B OR B = C OR A = C THEN 'Isosceles'
      ELSE 'Scalene'
   END AS TriangleType
FROM
   TRIANGLES;
```



Prepare > SQL > Advanced Select > Type of Triangle

Type of Triangle

Problem | Submissions | Leaderboard | Discussions

Write a query identifying the type of each record in the **TRIANGLES** table using its three side lengths. Output one of the following statements for each record in the table:

- **Equilateral**: It's a triangle with 3 sides of equal length.
- **Isosceles**: It's a triangle with 2 sides of equal length.
- **Scalene**: It's a triangle with 3 sides of differing lengths.
- **Not A Triangle**: The given values of A, B, and C don't form a triangle.

Input Format

The **TRIANGLES** table is described as follows:

Column	Type
A	Integer
B	Integer
C	Integer

Each row in the table denotes the lengths of each of a triangle's three sides.

Sample Input

A	B	C
20	20	23
20	20	20
20	21	22
13	14	30

Sample Output

```
Isosceles
Equilateral
Scalene
Not A Triangle
```

Explanation

Values in the tuple $(20, 20, 23)$ form an Isosceles triangle, because $A = B$.
Values in the tuple $(20, 20, 20)$ form an Equilateral triangle, because $A = B = C$. Values in the tuple $(20, 21, 22)$ form a Scalene triangle, because $A \neq B \neq C$.
Values in the tuple $(13, 14, 30)$ cannot form a triangle because the combined value of sides A and B is not larger than that of side C.

MySQL

```
# Write your query here and follow these instructions:
#   1. Please append a semicolon ";" at the end of the query and enter your query in a single line to avoid error.
#   2. The AS keyword causes errors, so follow this convention: "SelCount b.Field From tabSel AS b" instead of "select b.Field from tabSel AS b"
#   3. Type your code immediately after comment. Don't leave any blank line.
*/
SELECT
    CASE
        WHEN A + B <= C OR A + C <= B OR B + C <= A THEN 'Not A Triangle'
        WHEN A = B AND B = C THEN 'Equilateral'
        WHEN A = B OR B = C OR A = C THEN 'Isosceles'
        ELSE 'Scalene'
    END AS TriangleType
FROM
    TRIANGLES;
```

Upload Code as File | Run Code | Submit Code

Congratulations!

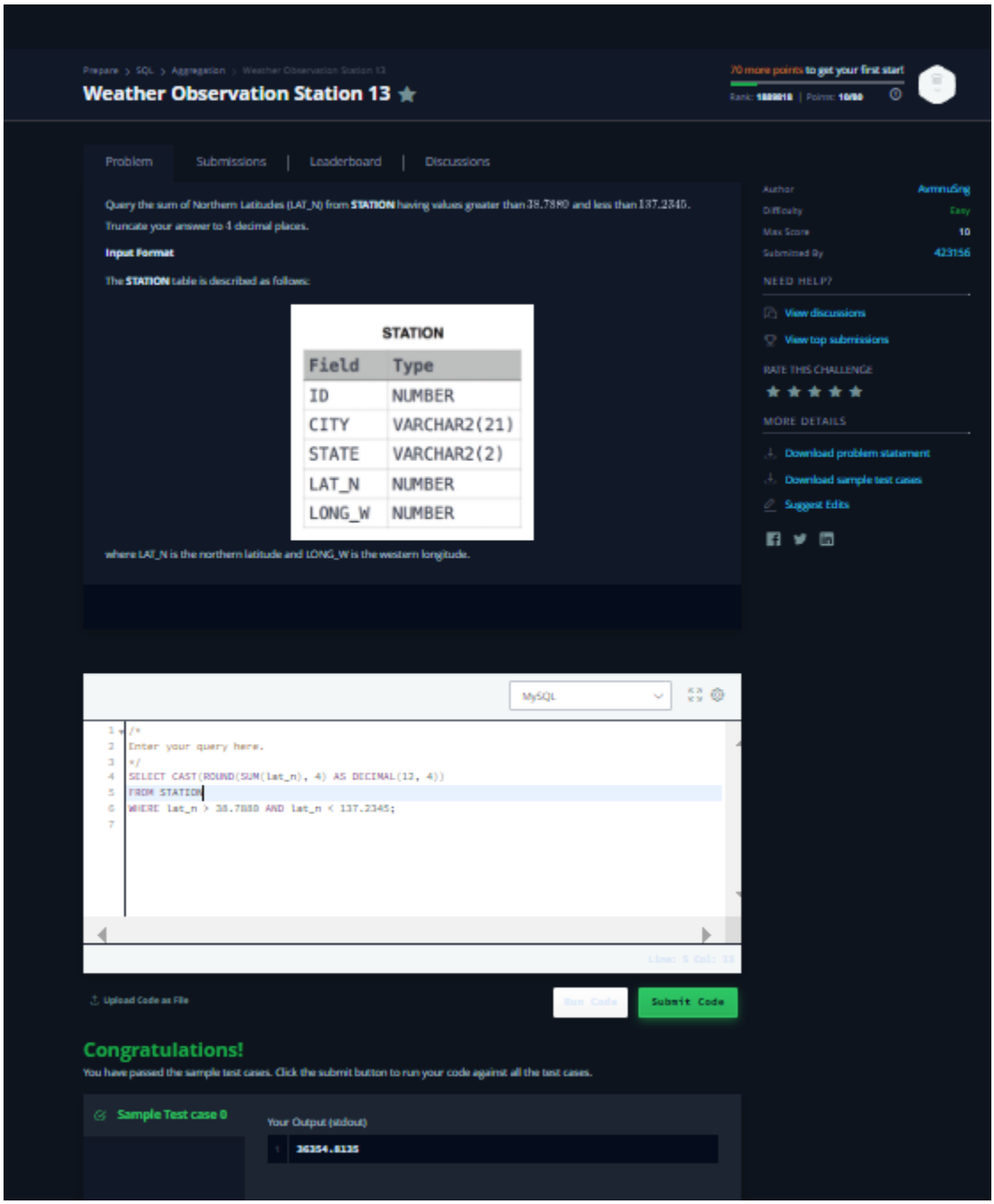
You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Your Output (stdout)

```
Equilateral
Equilateral
Isosceles
Equilateral
Isosceles
Equilateral
Scalene
Not A Triangle
Scalene
Scalene
Scalene
Not A Triangle
```

9. [Weather Observation Station 13]
   QUERY:
   SELECT CAST(ROUND(SUM(lat_n), 4) AS DECIMAL(12, 4))
   FROM STATION
   WHERE lat_n > 38.7880 AND lat_n < 137.2345;



10. The Report

    QUERY:

```
 SELECT IF(G.Grade >7 ,S.Name, NULL), G.Grade, S.Marks
 FROM Students AS S
JOIN Grades AS G ON S.Marks >=G.Min_Mark AND S.Marks <=G.Max_Mark
ORDER BY 2 DESC, S.Name
```

Name	String
Marks	Integer

Grades contains the following data:

Grade	Min_Mark	Max_Mark
1	0	9
2	10	19
3	20	29
4	30	39
5	40	49
6	50	59
7	60	69
8	70	79
9	80	89
10	90	100

Kelly gives Fae a task to generate a report containing three columns: Name, Grade and Mark. Kelly doesn't want the NAMES of those students who received a grade lower than 8. The report must be in descending order by grade -- i.e. higher grades are entered first. If there is more than one student with the same grade (8-10) assigned to them, order those particular students by their name alphabetically. Finally, if the grade is lower than 8, use "NULL" as their name and list them by their grades in descending order. If there is more than one student with the same grade (1-7) assigned to them, order those particular students by their marks in ascending order.

Write a query to help Fae.

Sample Input

ID	Name	Marks
1	Julia	88
2	Samantha	68
3	Maria	99
4	Scarlet	78
5	Ashley	63
6	Jane	81

Sample Output

```
Maria 10 99
Jane 9 81
Julia 9 88
Scarlet 8 78
NULL 7 63
NULL 7 68
```

Note

Print "NULL" as the name if the grade is less than 8.

Explanation

Consider the following table with the grades assigned to the students:

ID	Name	Marks	Grade
1	Julia	88	9
2	Samantha	68	7
3	Maria	99	10
4	Scarlet	78	8
5	Ashley	63	7
6	Jane	81	9

So, the following students got 8, 9 or 10 grades:

- Maria (grade 10)
- Jane (grade 9)
- Julia (grade 9)
- Scarlet (grade 8)

View discussions

View top submissions

RATE THIS CHALLENGE

MORE DETAILS

Download problem statement

Download sample test cases

Suggest hints

Upload Code as File

Test Code

Submit Code

Congratulations!