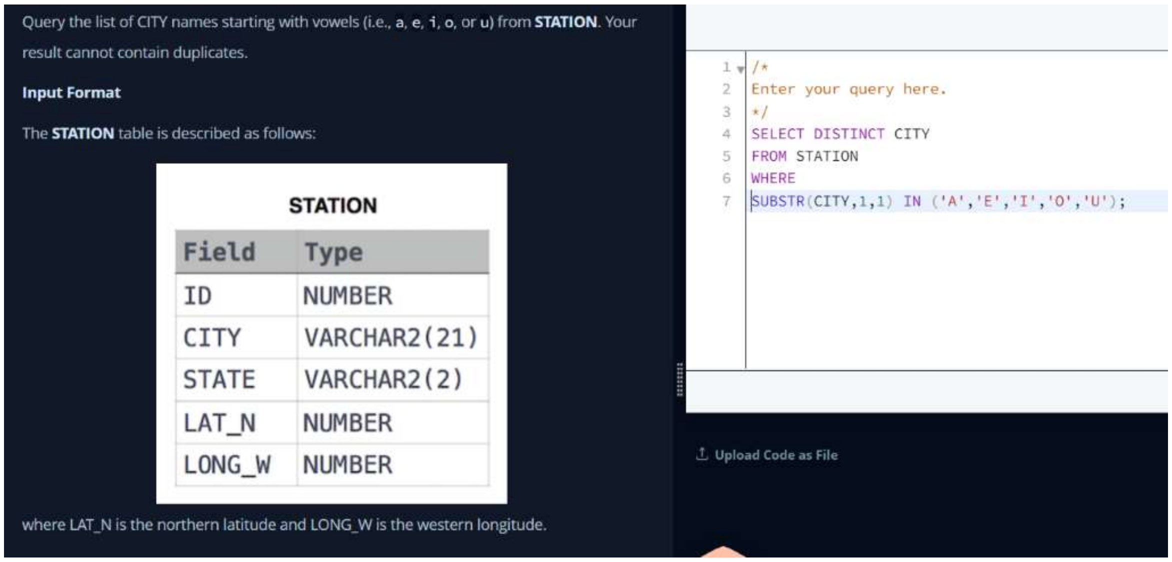
EASY

Sunday, November 6, 2022 4:35 PM

1) COLUMN VALUES START WITH CERTAIN CHARACTERS GIVEN


Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result cannot contain duplicates.

Input Format

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

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

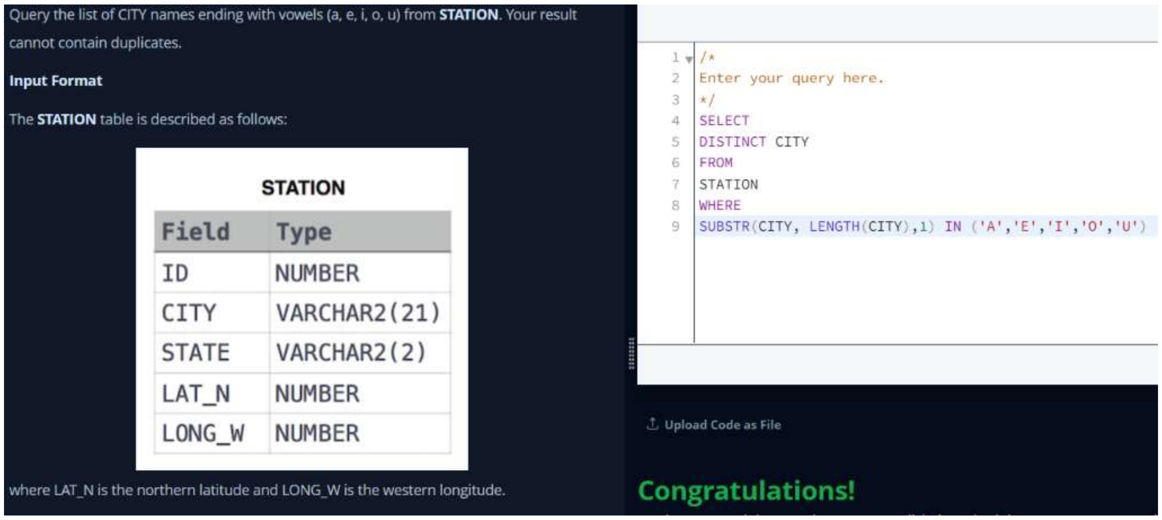
where LAT_N is the northern latitude and LONG_W is the western longitude.

```
/*
Enter your query here.
*/
SELECT DISTINCT CITY
FROM STATION
WHERE
SUBSTR(CITY,1,1) IN ('A','E','I','O','U');
```

BONUS – CAN USE 1) LOWER(CITY) IN (a,e,I,o,u) 2) WHERE LEFT(CITY,1) IN ('A','E','I','O','U')

SUBSTR(COLUMN, 5 , 3) Start at 5 th position, extract 3 characters

2. COLUMN VALUES END WITH CERTAIN CHARACTERS


Query the list of CITY names ending with vowels (a, e, i, o, u) from **STATION**. Your result cannot contain duplicates.

Input Format

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

STATION

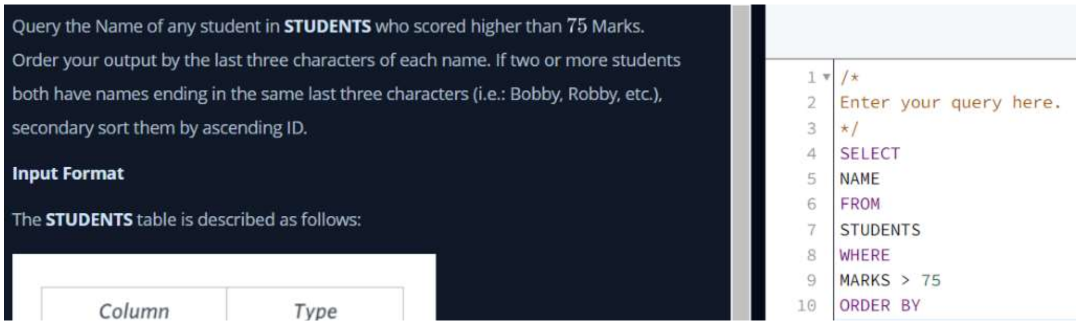
Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where LAT_N is the northern latitude and LONG_W is the western longitude.

```
/*
Enter your query here.
*/
SELECT
DISTINCT CITY
FROM
STATION
WHERE
SUBSTR(CITY, LENGTH(CITY),1) IN ('A','E','I','O','U')
```

BONUS - RIGHT(CITY,1) IN ('A','E','I','O','U')

3.

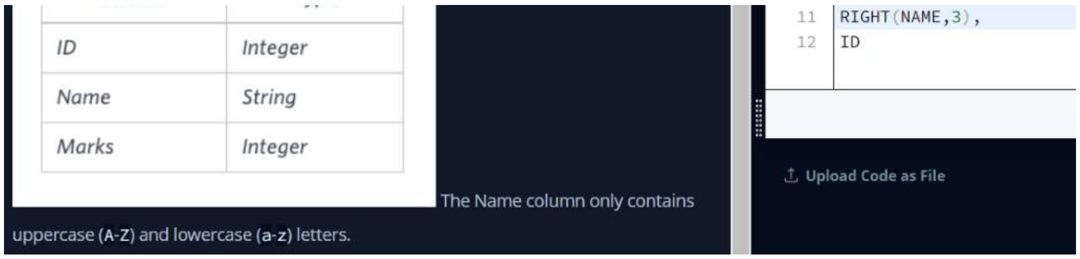

Query the Name of any student in **STUDENTS** who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

Input Format

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

Column	Type

```
/*
Enter your query here.
*/
SELECT
NAME
FROM
STUDENTS
WHERE
MARKS > 75
ORDER BY
```

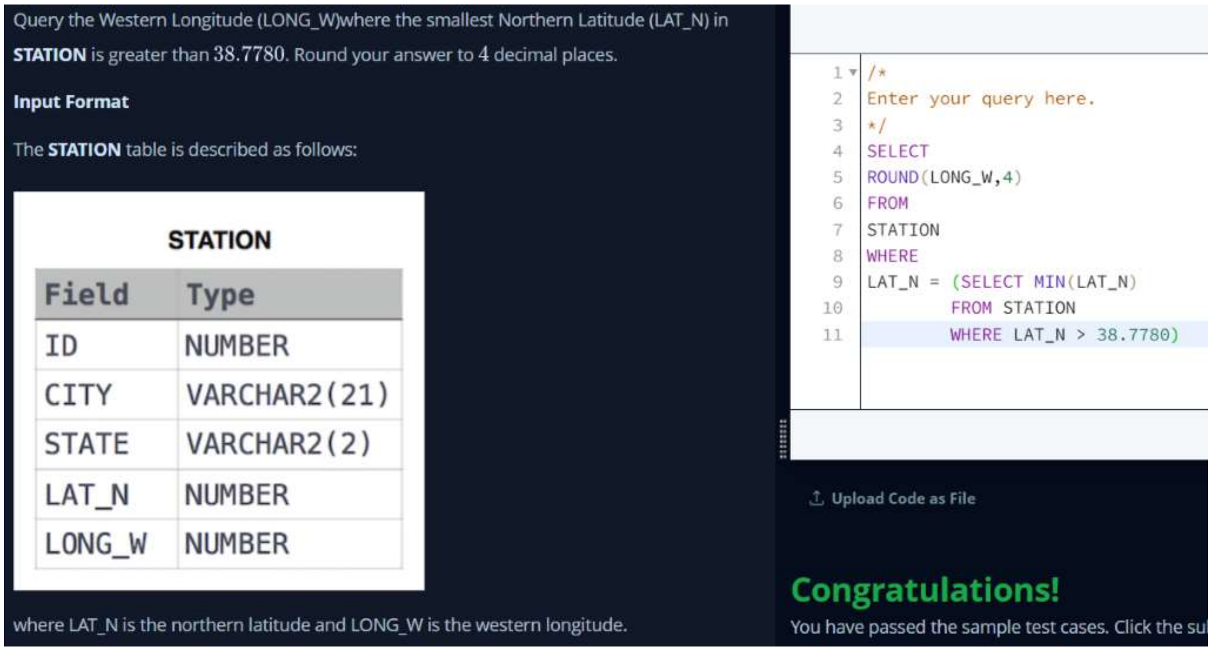


ID	Integer
Name	String
Marks	Integer

The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.

```
RIGHT(NAME,3),
ID
```

4.



Query the Western Longitude (LONG_W)where the smallest Northern Latitude (LAT_N) in **STATION** is greater than 38.7780. Round your answer to 4 decimal places.

Input Format

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

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

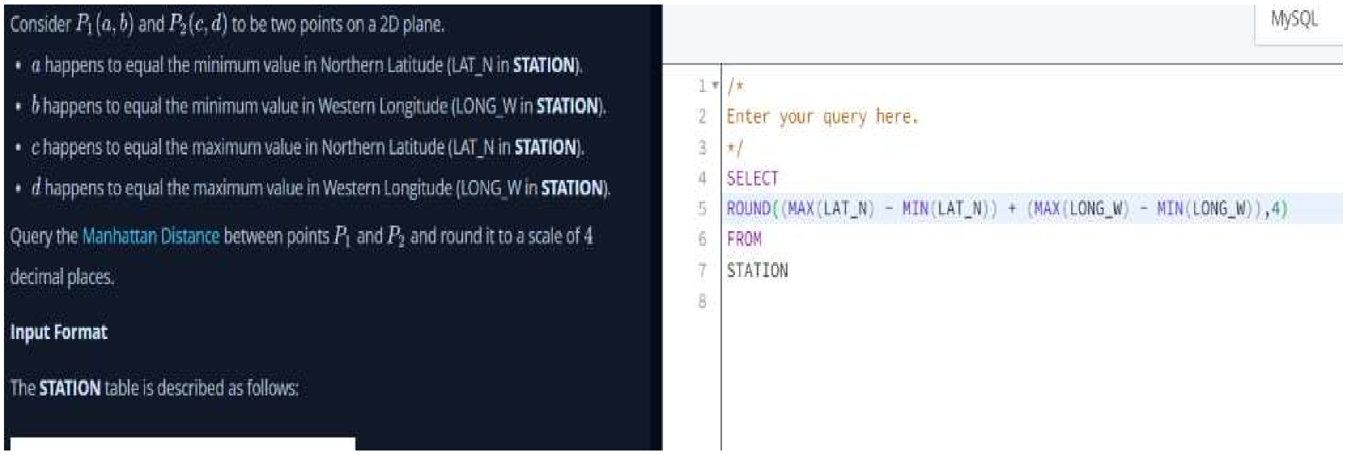
where LAT_N is the northern latitude and LONG_W is the western longitude.

```
/*
Enter your query here.
*/
SELECT
ROUND(LONG_W,4)
FROM
STATION
WHERE
LAT_N = (SELECT MIN(LAT_N)
        FROM STATION
        WHERE LAT_N > 38.7780)
```

Congratulations!

You have passed the sample test cases. Click the su

5.



Consider $P_1(a,b)$ and $P_2(c,d)$ to be two points on a 2D plane.

- a happens to equal the minimum value in Northern Latitude (LAT_N in **STATION**).
- b happens to equal the minimum value in Western Longitude (LONG_W in **STATION**).
- c happens to equal the maximum value in Northern Latitude (LAT_N in **STATION**).
- d happens to equal the maximum value in Western Longitude (LONG_W in **STATION**).

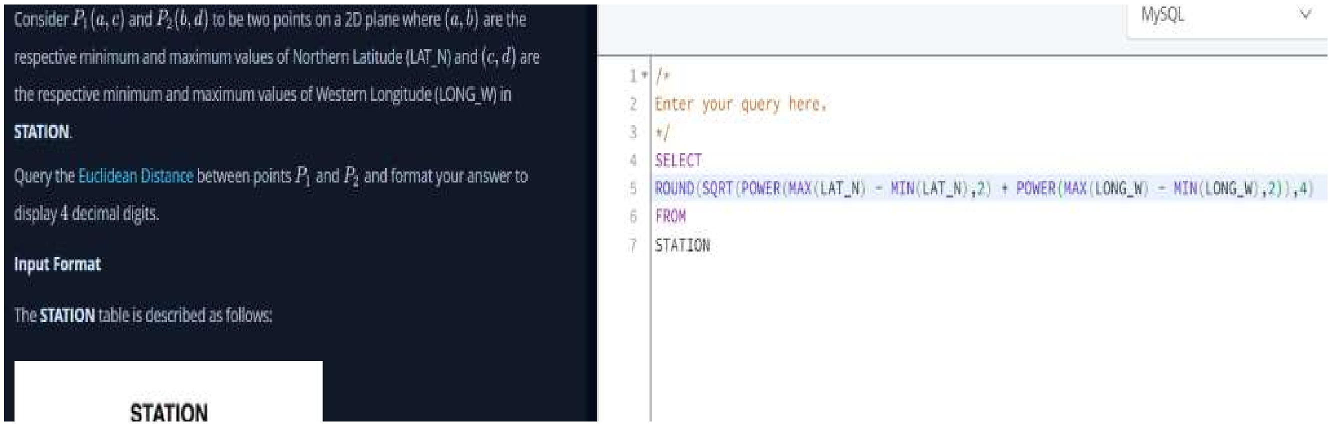
Query the Manhattan Distance between points P_1 and P_2 and round it to a scale of 4 decimal places.

Input Format

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

```
/*
Enter your query here.
*/
SELECT
ROUND((MAX(LAT_N) - MIN(LAT_N)) + (MAX(LONG_W) - MIN(LONG_W)),4)
FROM
STATION
```

6.



Consider $P_1(a,c)$ and $P_2(b,d)$ to be two points on a 2D plane where (a,b) are the respective minimum and maximum values of Northern Latitude (LAT_N) and (c,d) are the respective minimum and maximum values of Western Longitude (LONG_W) in **STATION**.

Query the Euclidean Distance between points P_1 and P_2 and format your answer to display 4 decimal digits.

Input Format

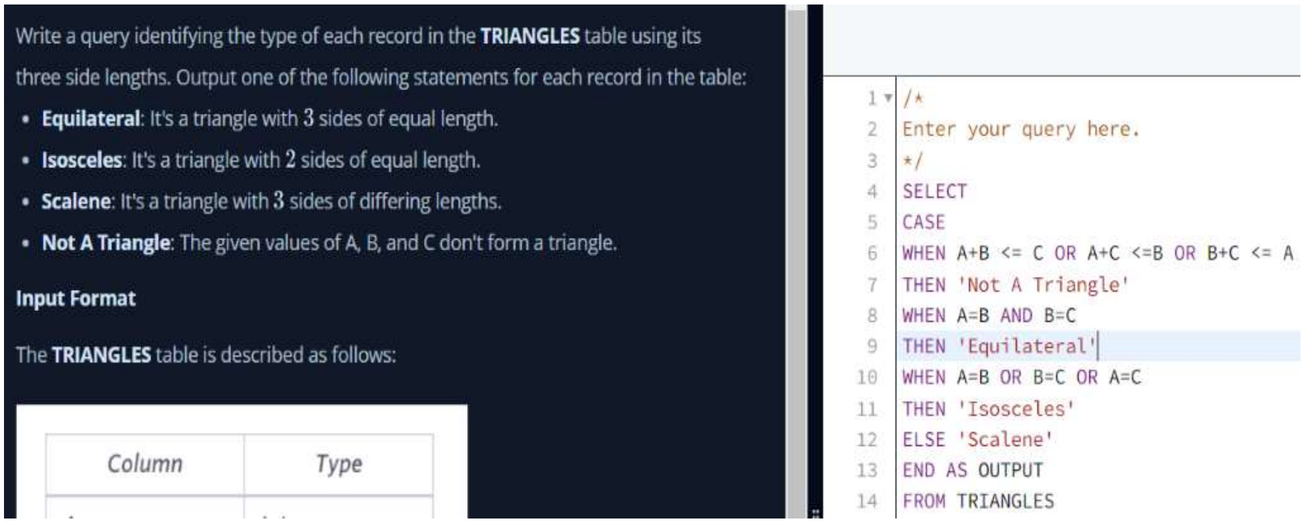
The **STATION** table is described as follows:

STATION

```
/*
Enter your query here.
*/
SELECT
ROUND(SQRT(POWER(MAX(LAT_N) - MIN(LAT_N),2) + POWER(MAX(LONG_W) - MIN(LONG_W),2)),4)
FROM
STATION
```

You can try and use SQUARE function , it wasn't working for this problem

7.



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.

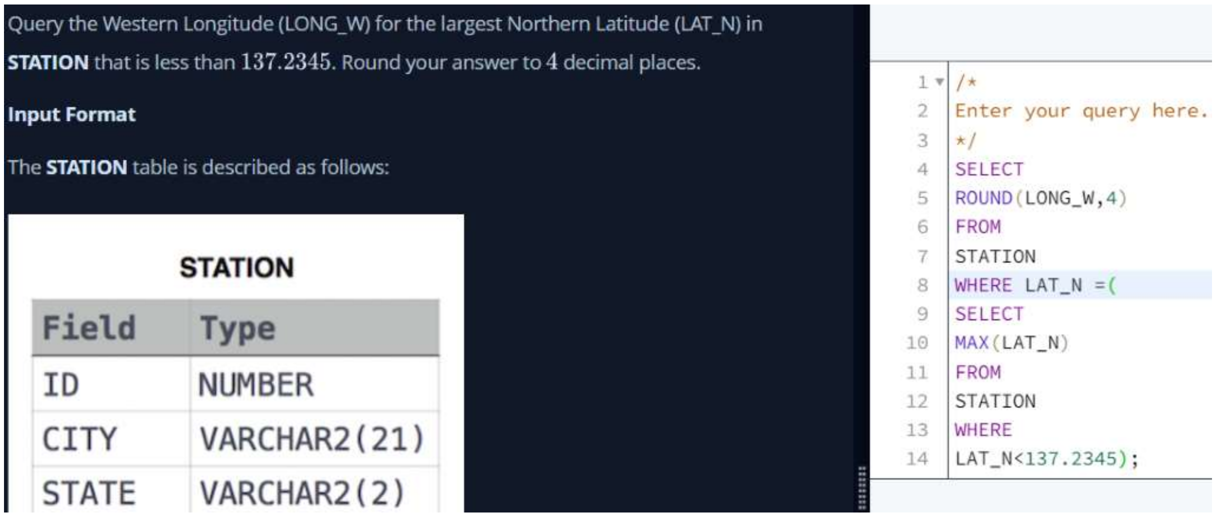
Input Format

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

Column	Type

```
/*
Enter your query here.
*/
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 OUTPUT
FROM TRIANGLES
```

8.



Query the Western Longitude (LONG_W) for the largest Northern Latitude (LAT_N) in **STATION** that is less than 137.2345. Round your answer to 4 decimal places.

Input Format

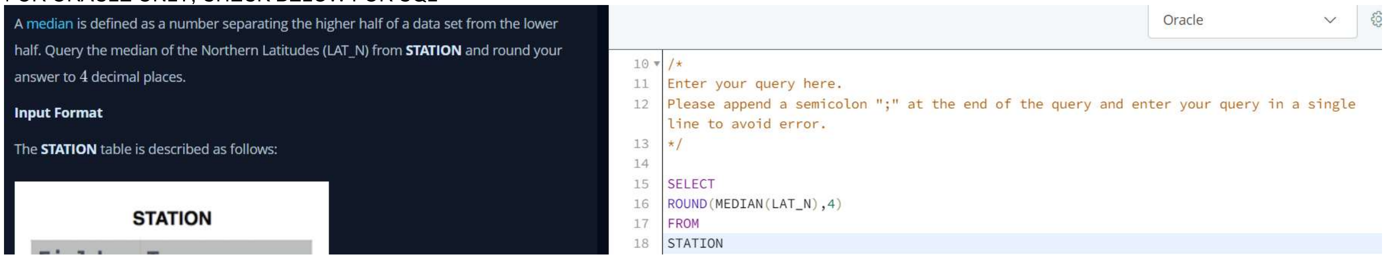
The **STATION** table is described as follows:

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)

```
/*
Enter your query here.
*/
SELECT
ROUND(LONG_W,4)
FROM
STATION
WHERE LAT_N =(
SELECT
MAX(LAT_N)
FROM
STATION
WHERE
LAT_N<137.2345);
```

9.
FOR ORACLE ONLY, CHECK BELOW FOR SQL



A median is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern Latitudes (LAT_N) from **STATION** and round your answer to 4 decimal places.

Input Format

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

STATION

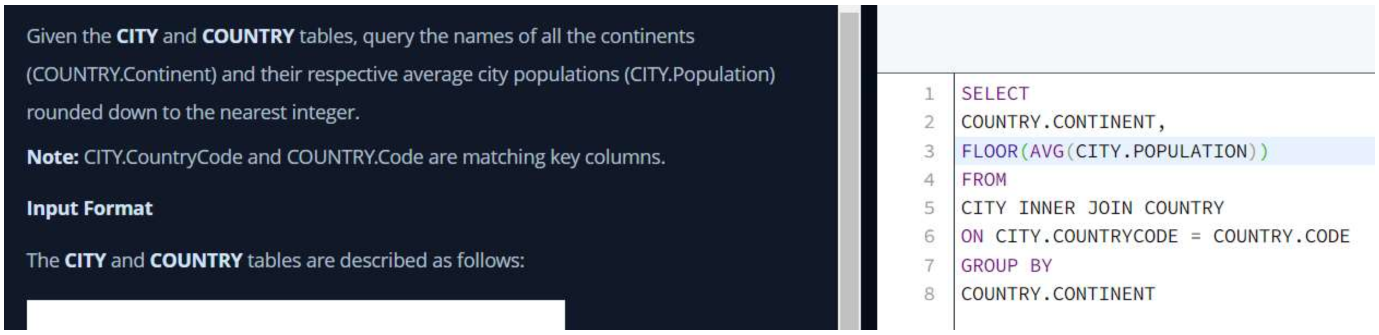
Oracle

```
/*
Enter your query here.
Please append a semicolon ";" at the end of the query and enter your query in a single line to avoid error.
*/

SELECT
ROUND(MEDIAN(LAT_N),4)
FROM
STATION
```

UNDERSTAND MEDIAN FOR SQL (LEFT)

10.



Given the **CITY** and **COUNTRY** tables, query the names of all the continents (COUNTRY.Continent) and their respective average city populations (CITY.Population) rounded down to the nearest integer.

Note: CITY.CountryCode and COUNTRY.Code are matching key columns.

Input Format

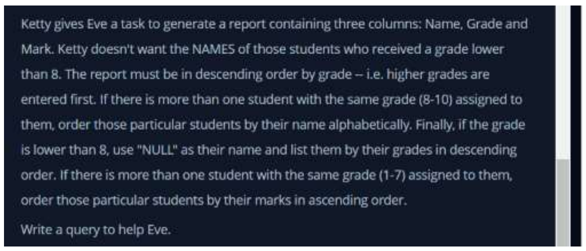
The **CITY** and **COUNTRY** tables are described as follows:

```
SELECT
COUNTRY.CONTINENT,
FLOOR(AVG(CITY.POPULATION))
FROM
CITY INNER JOIN COUNTRY
ON CITY.COUNTRYCODE = COUNTRY.CODE
GROUP BY
COUNTRY.CONTINENT
```

CITY

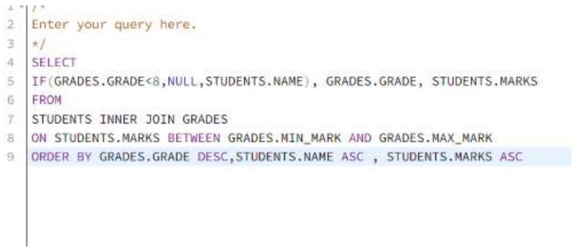
Field	Type

11.

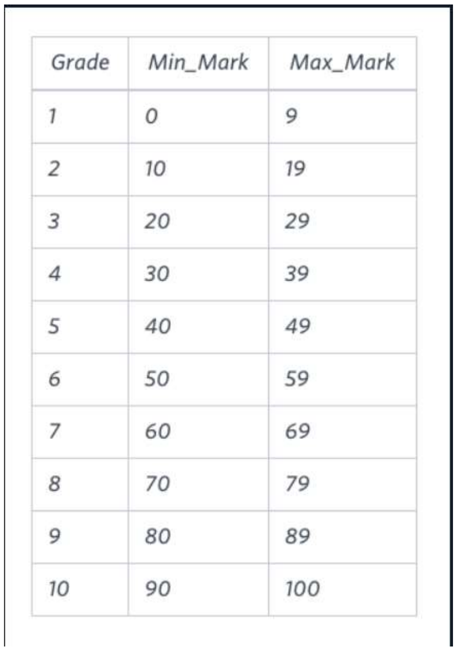


Ketty gives Eve a task to generate a report containing three columns: Name, Grade and Mark. Ketty doesn't want the NAMES of those students who received a grade lower than 8. The report must be in descending order by grade -- i.e. higher grades are entered first. If there is more than one student with 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.

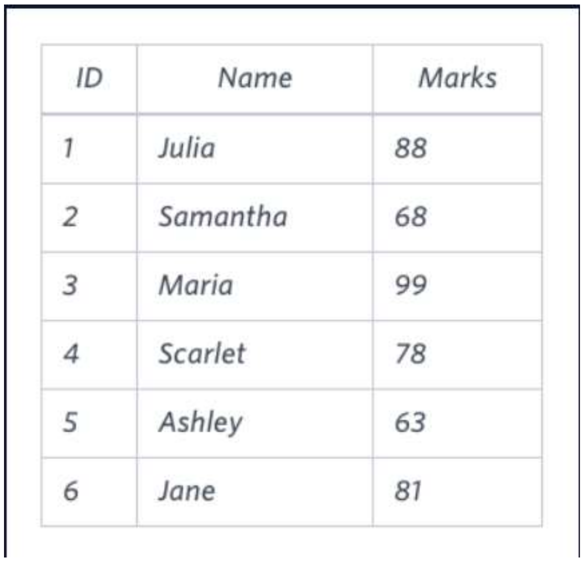
Write a query to help Eve.



```sql
/*
Enter your query here.
*/
SELECT
IF(GRADES.GRADE<8,NULL,STUDENTS.NAME), GRADES.GRADE, STUDENTS.MARKS
FROM
STUDENTS INNER JOIN GRADES
ON STUDENTS.MARKS BETWEEN GRADES.MIN_MARK AND GRADES.MAX_MARK
ORDER BY GRADES.GRADE DESC,STUDENTS.NAME ASC , STUDENTS.MARKS ASC
```

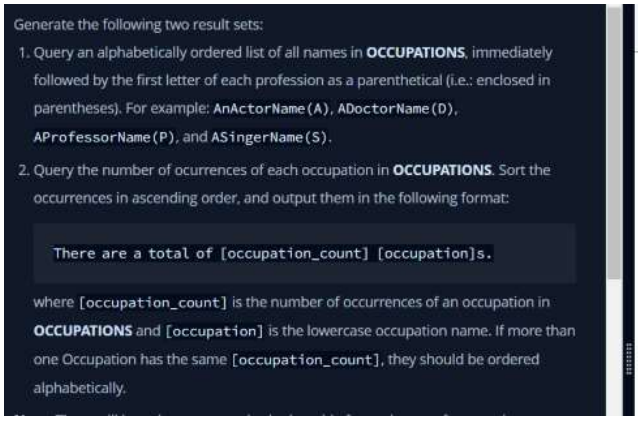


Grade	Min_Mark	Max_Mark
1	0	9
2	10	19
3	20	29
4	30	39
5	40	49
6	50	59
7	60	69
8	70	79
9	80	89
10	90	100



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

12.

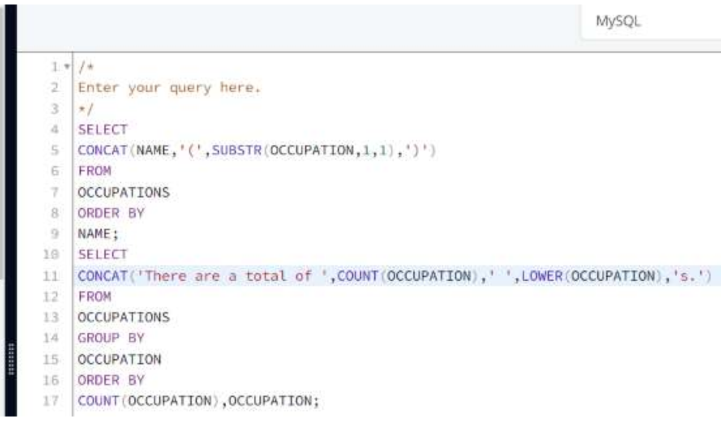


Generate the following two result sets:

1. Query an alphabetically ordered list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: `AnActorName(A)`, `ADoctorName(D)`, `AProfessorName(P)`, and `ASingerName(S)`.
2. Query the number of ocurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in ascending order, and output them in the following format:

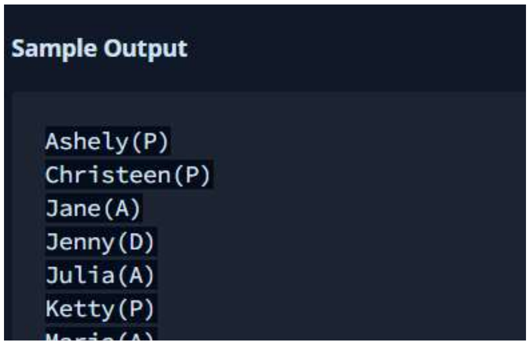
```
There are a total of [occupation_count] [occupation]s.
```

where `[occupation_count]` is the number of occurrences of an occupation in **OCCUPATIONS** and `[occupation]` is the lowercase occupation name. If more than one Occupation has the same `[occupation_count]`, they should be ordered alphabetically.



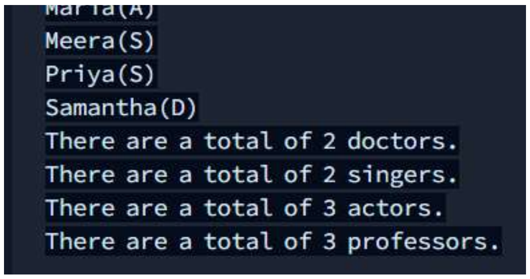
MySQL

```sql
/*
Enter your query here.
*/
SELECT
CONCAT(NAME,'(',SUBSTR(OCCUPATION,1,1),')')
FROM
OCCUPATIONS
ORDER BY
NAME;
SELECT
CONCAT('There are a total of ',COUNT(OCCUPATION),' ',LOWER(OCCUPATION),'s.')
FROM
OCCUPATIONS
GROUP BY
OCCUPATION
ORDER BY
COUNT(OCCUPATION),OCCUPATION;
```



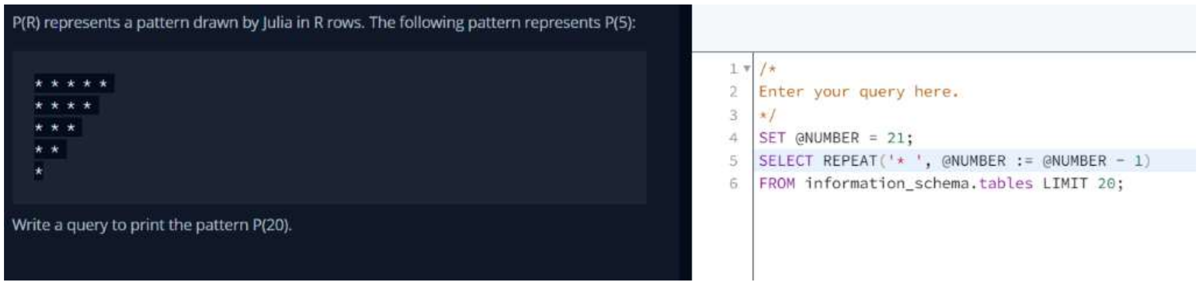
Sample Output

```
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 2 doctors.
There are a total of 2 singers.
There are a total of 3 actors.
There are a total of 3 professors.
```

13.



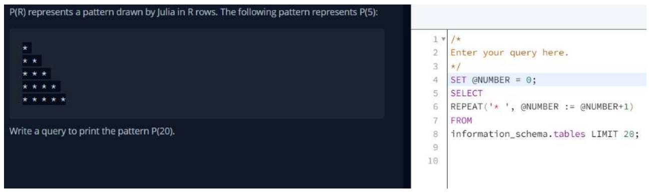
P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
* * * * *
* * * *
* * *
* *
*
```

Write a query to print the pattern P(20).

```sql
/*
Enter your query here.
*/
SET @NUMBER = 21;
SELECT REPEAT('* ', @NUMBER := @NUMBER - 1)
FROM information_schema.tables LIMIT 20;
```

14



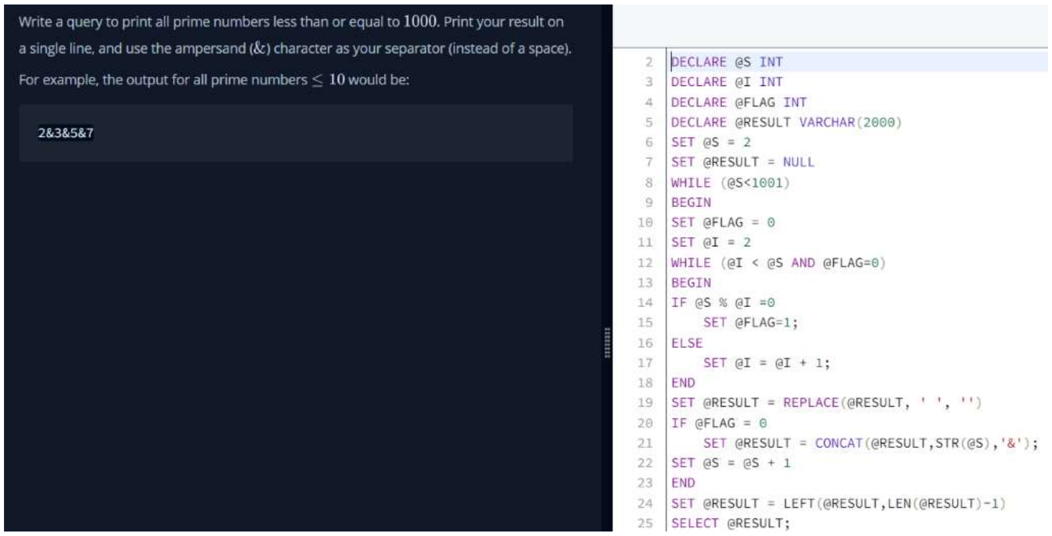
P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
*
* *
* * *
* * * *
* * * * *
```

Write a query to print the pattern P(20).

```sql
/*
Enter your query here.
*/
SET @NUMBER = 0;
SELECT
REPEAT('* ', @NUMBER := @NUMBER+1)
FROM
information_schema.tables LIMIT 20;
```

15.



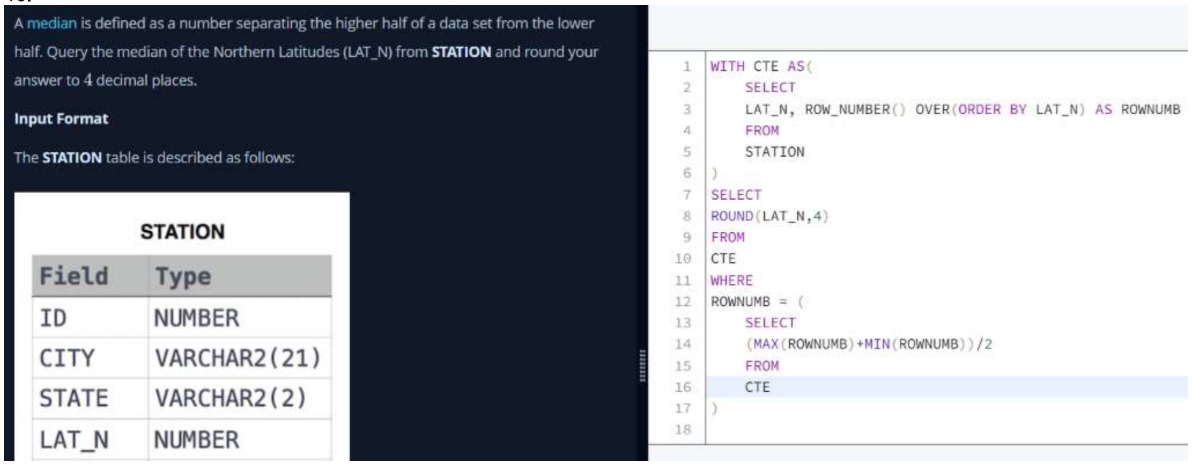
Write a query to print all prime numbers less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

For example, the output for all prime numbers ≤ 10 would be:

```
2&3&5&7
```

```sql
DECLARE @S INT
DECLARE @I INT
DECLARE @FLAG INT
DECLARE @RESULT VARCHAR(2000)
SET @S = 2
SET @RESULT = NULL
WHILE (@S<1001)
BEGIN
SET @FLAG = 0
SET @I = 2
WHILE (@I < @S AND @FLAG=0)
BEGIN
IF @S % @I =0
    SET @FLAG=1;
ELSE
    SET @I = @I + 1;
END
SET @RESULT = REPLACE(@RESULT, ' ', '')
IF @FLAG = 0
    SET @RESULT = CONCAT(@RESULT,STR(@S),'&');
SET @S = @S + 1
END
SET @RESULT = LEFT(@RESULT,LEN(@RESULT)-1)
SELECT @RESULT;
```

16.



A median is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern Latitudes (LAT_N) from **STATION** and round your answer to 4 decimal places.

Input Format

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

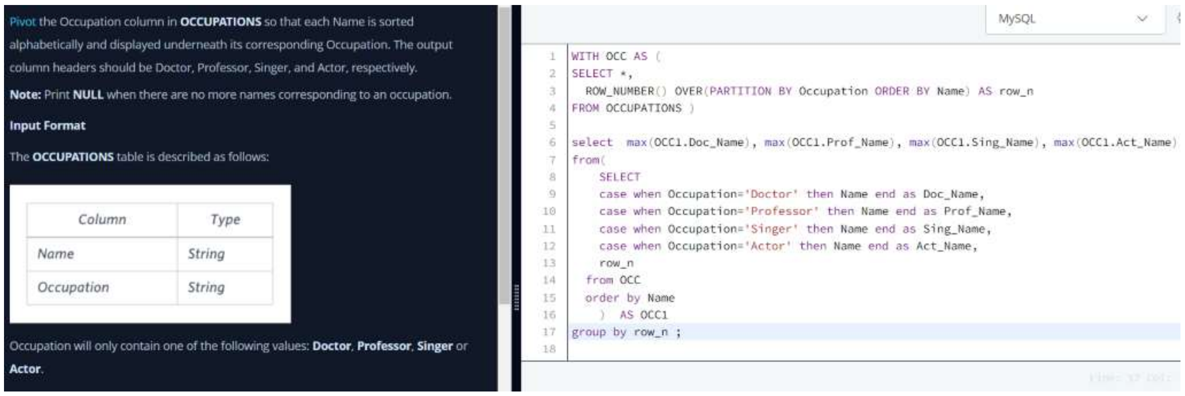
STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER

```sql
WITH CTE AS(
    SELECT
    LAT_N, ROW_NUMBER() OVER(ORDER BY LAT_N) AS ROWNUMB
    FROM
    STATION
)
SELECT
ROUND(LAT_N,4)
FROM
CTE
WHERE
ROWNUMB = (
    SELECT
    (MAX(ROWNUMB)+MIN(ROWNUMB))/2
    FROM
    CTE
)
```

LONG_W	NUMBER

17.



Pivot the Occupation column in **OCCUPATIONS** so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively.

Note: Print **NULL** when there are no more names corresponding to an occupation.

Input Format

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**.

MySQL

```sql
WITH OCC AS (
SELECT *,
  ROW_NUMBER() OVER(PARTITION BY Occupation ORDER BY Name) AS row_n
FROM OCCUPATIONS )

select  max(OCC1.Doc_Name), max(OCC1.Prof_Name), max(OCC1.Sing_Name), max(OCC1.Act_Name)
from(
    SELECT
    case when Occupation='Doctor' then Name end as Doc_Name,
    case when Occupation='Professor' then Name end as Prof_Name,
    case when Occupation='Singer' then Name end as Sing_Name,
    case when Occupation='Actor' then Name end as Act_Name,
    row_n
  from OCC
  order by Name
    )  AS OCC1
group by row_n ;
```