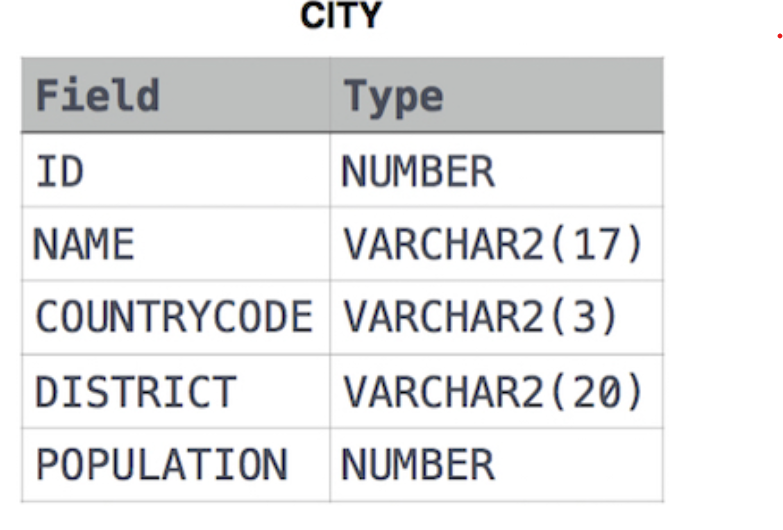
17) Query all columns for a city in **CITY** with the *ID* 1661.

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



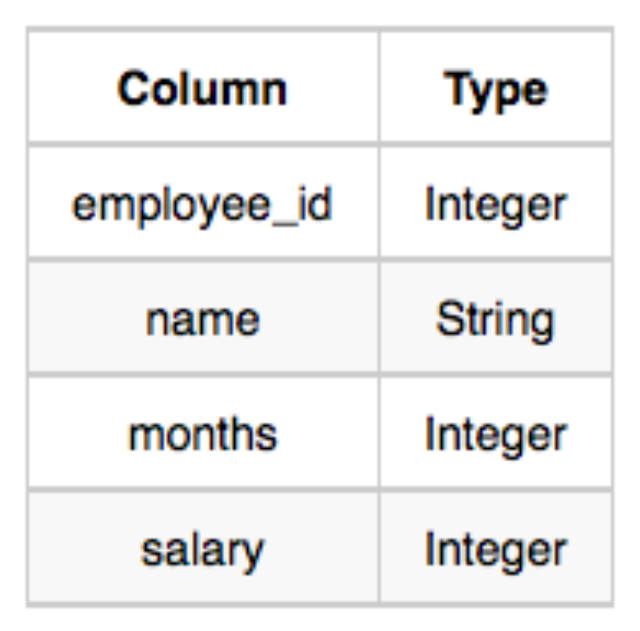
Solution:

SELECT \* FROM CITY WHERE ID= 1661;

18) Write a query that prints a list of employee names (i.e.: the *name* attribute) from the **Employee** table in alphabetical order.

**Input Format**

The **Employee** table containing employee data for a company is described as follows:

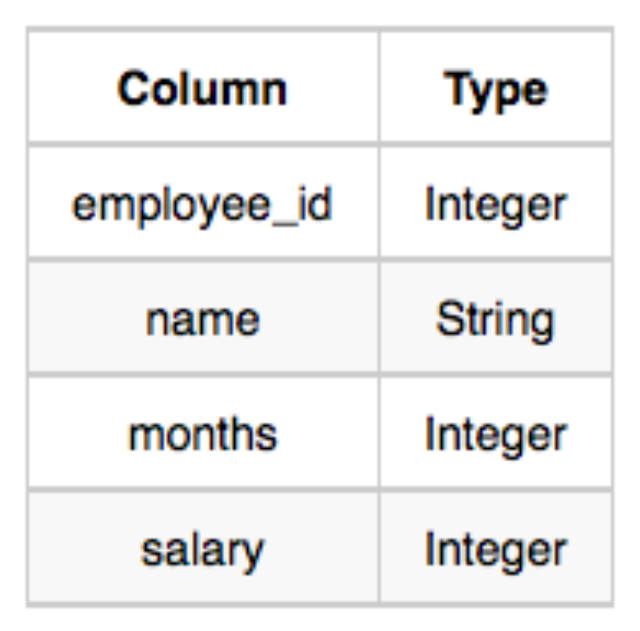


Solution:

SELECT NAME FROM EMPLOYEE ORDER BY NAME

19) Write a query that prints a list of employee names (i.e.: the *name* attribute) for employees in **Employee** having a salary greater than  per month who have been employees for less than  months. Sort your result by ascending *employee\_id*.

**Employee**



Solution:

SELECT NAME FROM EMPLOYEE WHERE SALARY > 2000 AND MONTHS < 10 ORDER BY EMPLOYEE\_ID;

20) 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:

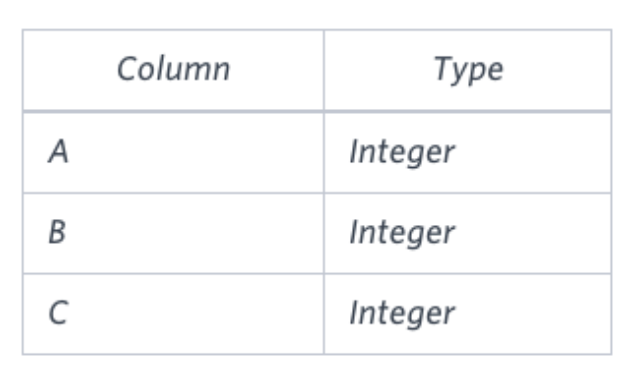
**Equilateral**: It's a triangle with  sides of equal length.

**Isosceles**: It's a triangle with  sides of equal length.

**Scalene**: It's a triangle with  sides of differing lengths.

**Not A Triangle**: The given values of *A*, *B*, and *C* don't form a triangle.

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



Solution: 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'

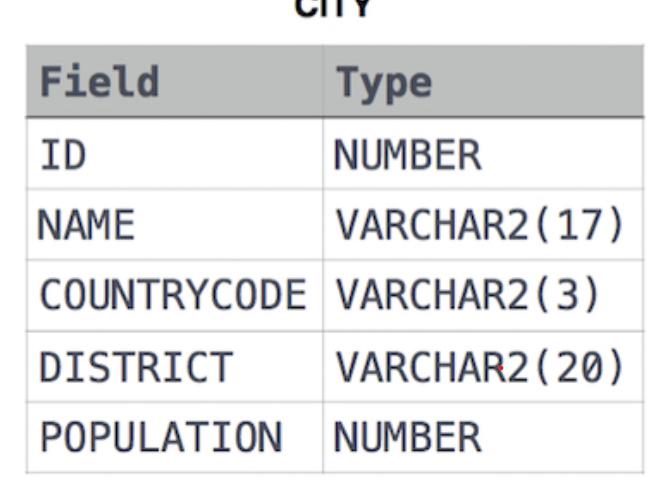
ELSE 'Scalene'

END AS TriangleType

FROM TRIANGLES;

21) Query a *count* of the number of cities in **CITY** having a *Population* larger than .

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

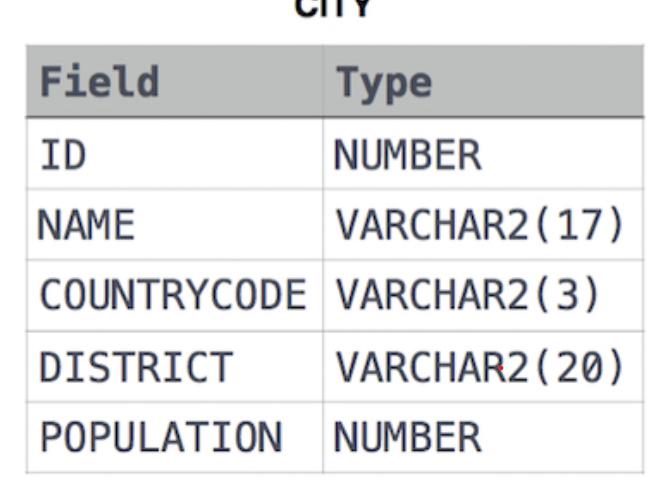


Solution:

SELECT COUNT(NAME) FROM CITY WHERE POPULATION > 100000

22) Query the total population of all cities in **CITY** where *District* is **California**.

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



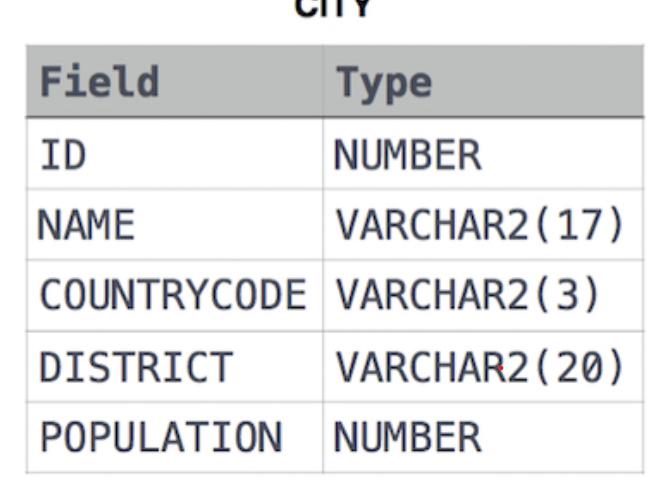
Solution:

SELECT SUM(POPULATION) FROM CITY WHERE DISTRICT= 'California'

23) Query the average population of all cities in **CITY** where *District* is **California**.

**Input Format**

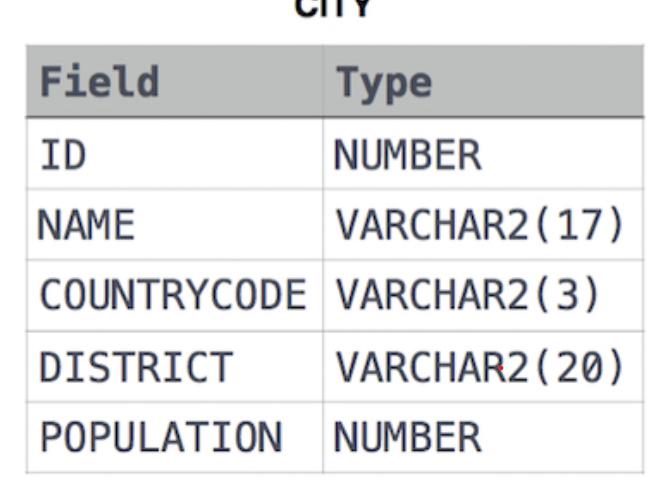
The **CITY** table is described as follows:



Solution:

SELECT AVG(POPULATION) FROM CITY WHERE DISTRICT='California'

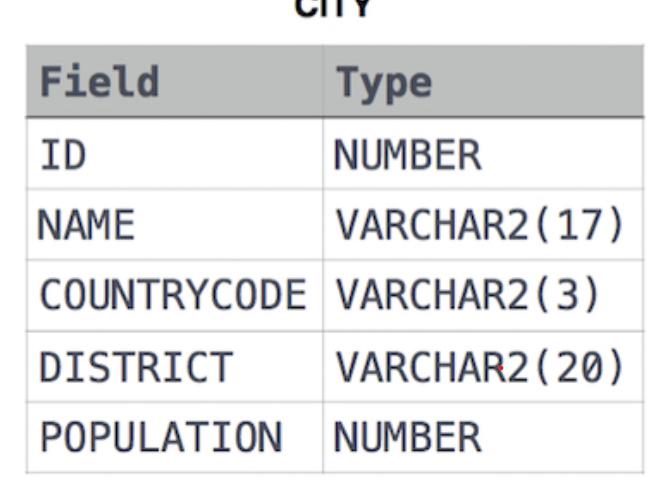
24) Query the average population for all cities in **CITY**, rounded *down* to the nearest integer.



Solution:

SELECT FLOOR(AVG(POPULATION)) FROM CITY ROUND

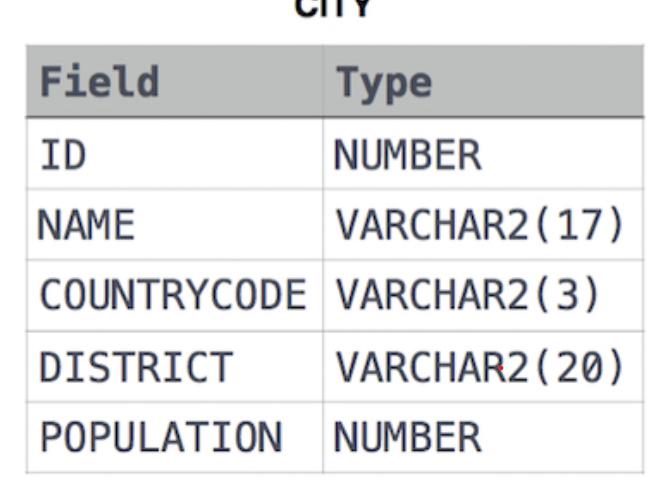
25) Query the sum of the populations for all Japanese cities in **CITY**. The *COUNTRYCODE* for Japan is **JPN**.



Solution:

SELECT SUM(POPULATION) FROM CITY WHERE COUNTRYCODE= 'JPN'

26) Query the difference between the maximum and minimum populations in **CITY**.



Solution:

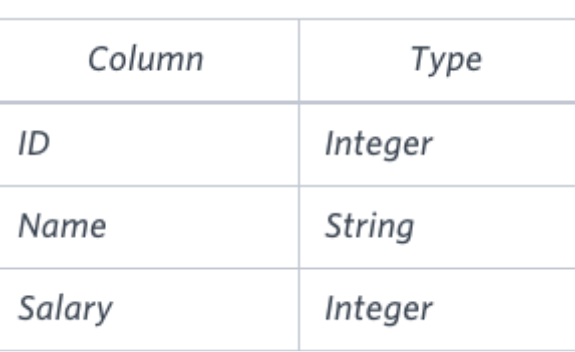
SELECT MAX(POPULATION)- MIN(POPULATION) FROM CITY

27) Samantha was tasked with calculating the average monthly salaries for all employees in the **EMPLOYEES** table, but did not realize her keyboard's  key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary.

Write a query calculating the amount of error (i.e.:  average monthly salaries), and round it up to the next integer.

**Input Format**

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



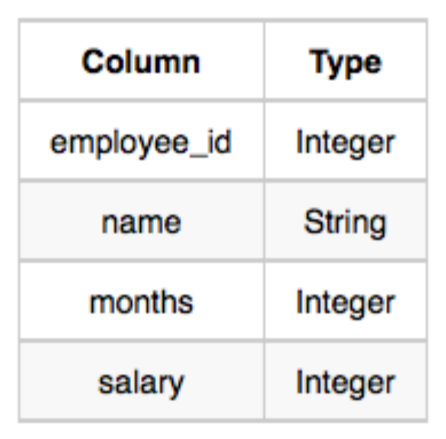
Solution:

SELECT CEIL(AVG(SALARY) - AVG(REPLACE(SALARY, '0', ''))) FROM EMPLOYEES;

28) We define an employee's *total earnings* to be their monthly  worked, and the *maximum total earnings* to be the maximum total earnings for any employee in the **Employee** table. Write a query to find the *maximum total earnings* for all employees as well as the total number of employees who have maximum total earnings. Then print these values as  space-separated integers.

**Input Format**

The **Employee** table containing employee data for a company is described as follows:

Solution:

SELECT salary \* months, count(\*) from Employee

group by salary \* months

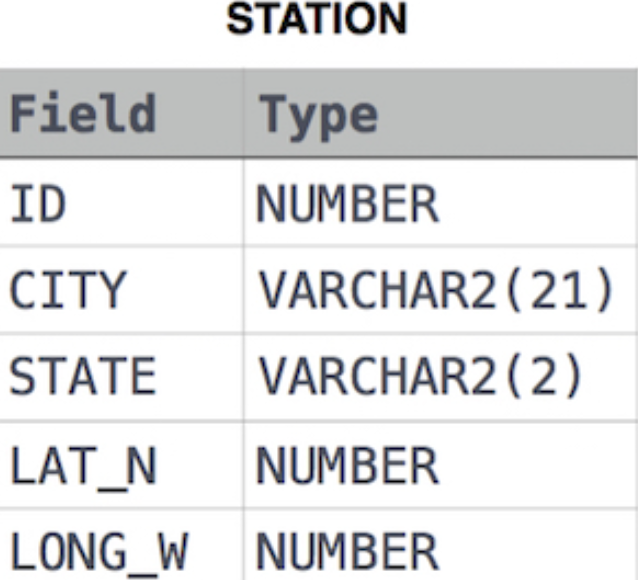
order by salary \* months DESC Limit 1;

29) Query the following two values from the **STATION** table:

1. The sum of all values in *LAT\_N* rounded to a scale of  decimal places.
2. The sum of all values in *LONG\_W* rounded to a scale of  decimal places.

**Input Format**

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

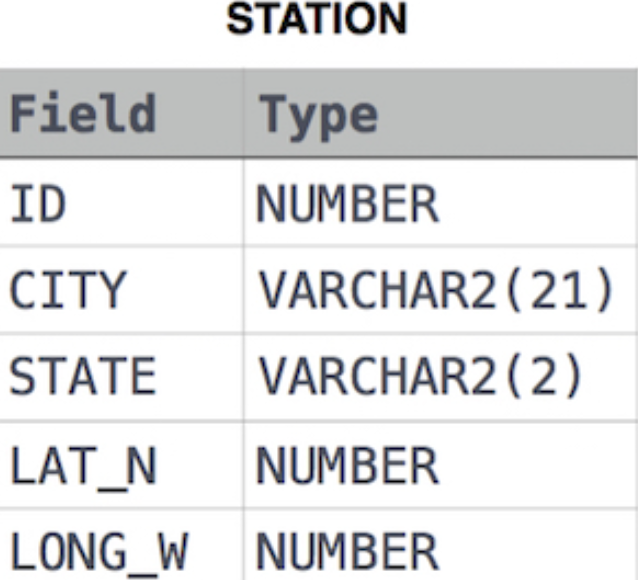


Solution:

SELECT ROUND(SUM(LAT\_N),2), ROUND(SUM(LONG\_W),2) from STATION;

30) Query the sum of *Northern Latitudes* (*LAT\_N*) from **STATION** having values greater than  and less than . Truncate your answer to  decimal places.

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



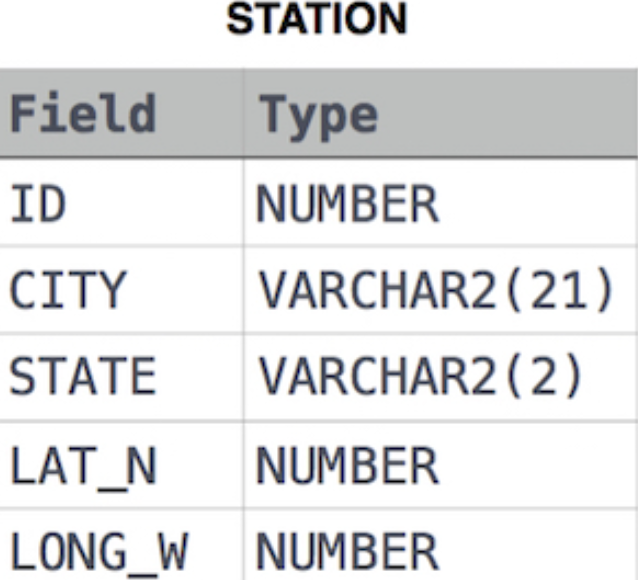
Solution:

SELECT ROUND(SUM(LAT\_N),4) FROM STATION WHERE LAT\_N > 38.7880 AND LAT\_N < 137.2345;

31) Query the greatest value of the *Northern Latitudes* (*LAT\_N*) from **STATION** that is less than . Truncate your answer to  decimal places.

**Input Format**

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



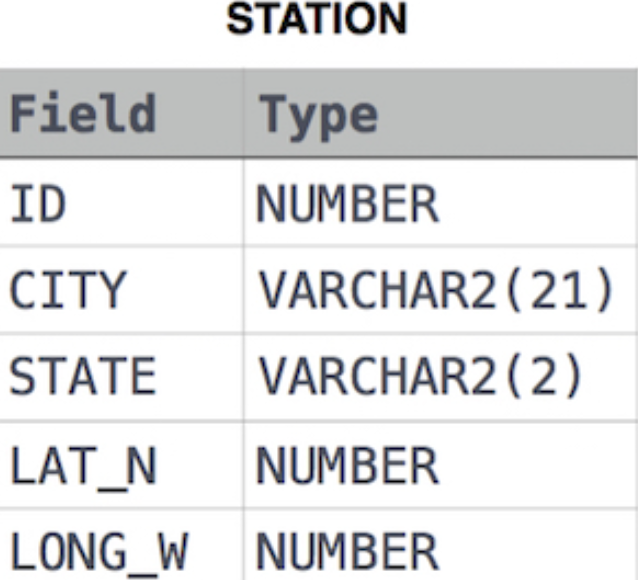
Solution:

SELECT ROUND(MAX(LAT\_N),4) FROM STATION WHERE LAT\_N < 137.2345

32) Query the *Western Longitude* (*LONG\_W*) for the largest *Northern Latitude* (*LAT\_N*) in **STATION** that is less than . Round your answer to  decimal places.

**Input Format**

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



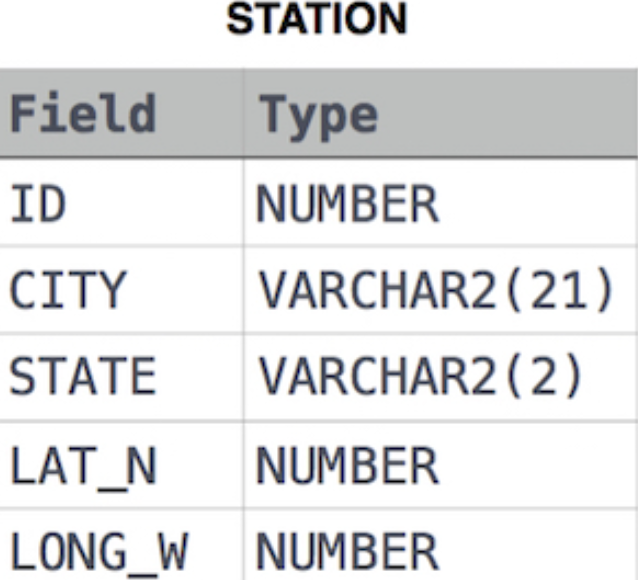
Solution:

SELECT ROUND(LONG\_W,4) FROM STATION WHERE LAT\_N < 137.2345 ORDER BY LAT\_N DESC LIMIT 1

33) Query the smallest *Northern Latitude* (*LAT\_N*) from **STATION** that is greater than . Round your answer to  decimal places.

**Input Format**

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



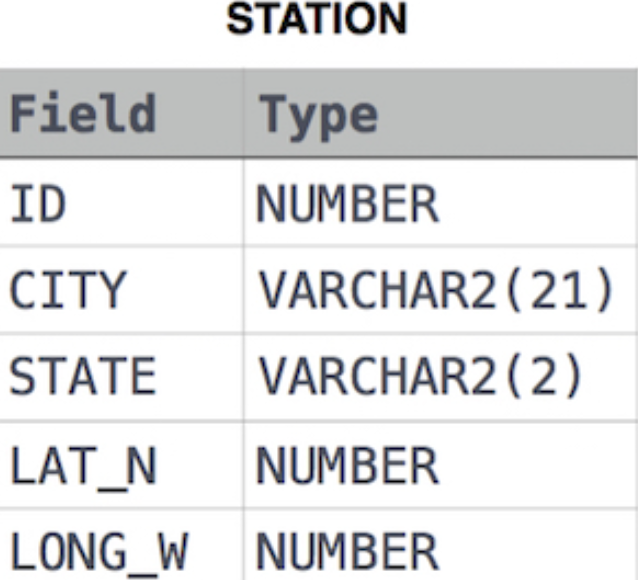
Solution:

SELECT ROUND(MIN(LAT\_N),4) FROM STATION WHERE LAT\_N > 38.7780

34) Query the *Western Longitude* (*LONG\_W*)where the smallest *Northern Latitude* (*LAT\_N*) in **STATION** is greater than . Round your answer to  decimal places.

**Input Format**

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



Solution:

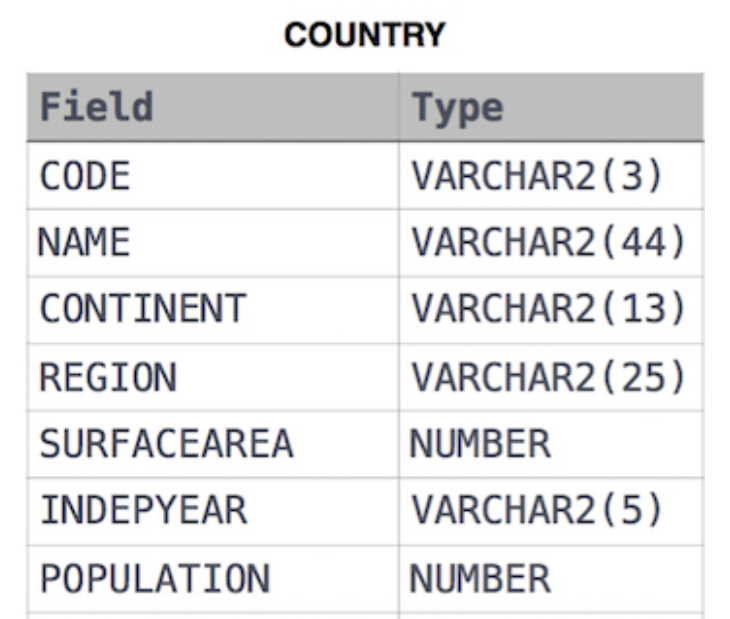
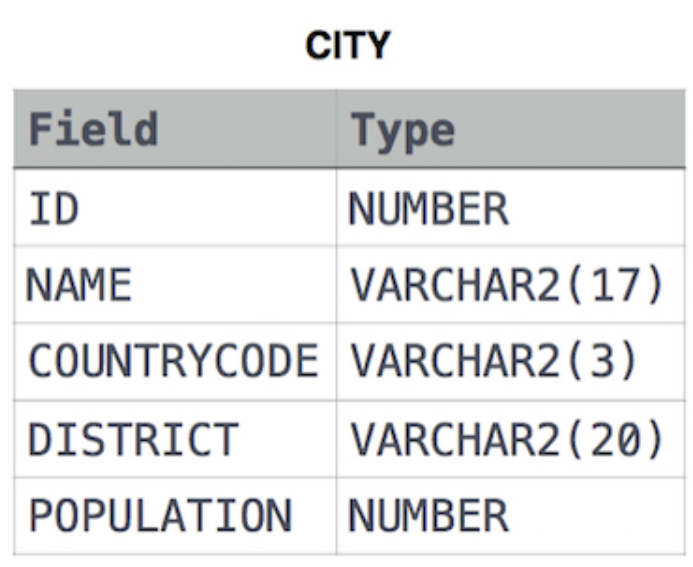
SELECT ROUND(LONG\_W,4) FROM STATION WHERE LAT\_N > 38.7780 ORDER BY LAT\_N LIMIT 1

35) Given the **CITY** and **COUNTRY** tables, query the sum of the populations of all cities where the *CONTINENT* is *'Asia'*.

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

**Input Format**

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



Solution:

SELECT SUM(CITY.Population)

FROM CITY

INNER JOIN COUNTRY ON CITY.CountryCode = COUNTRY.Code

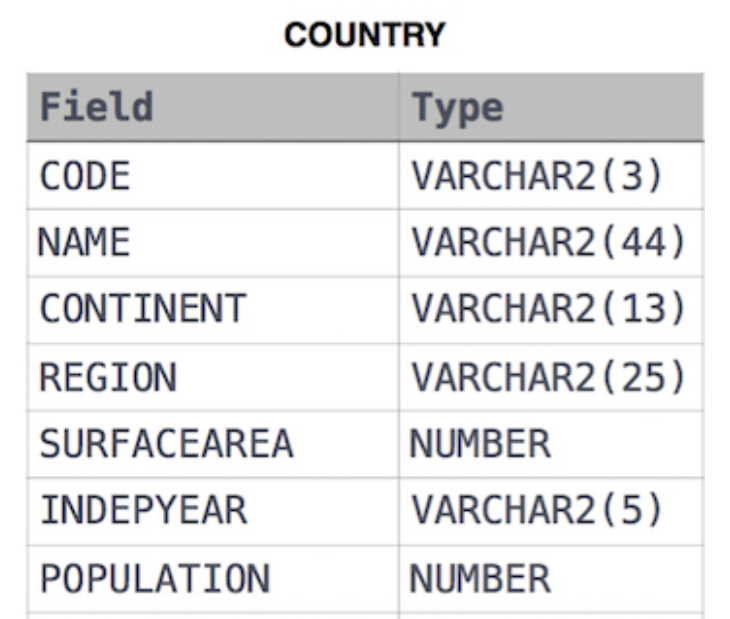
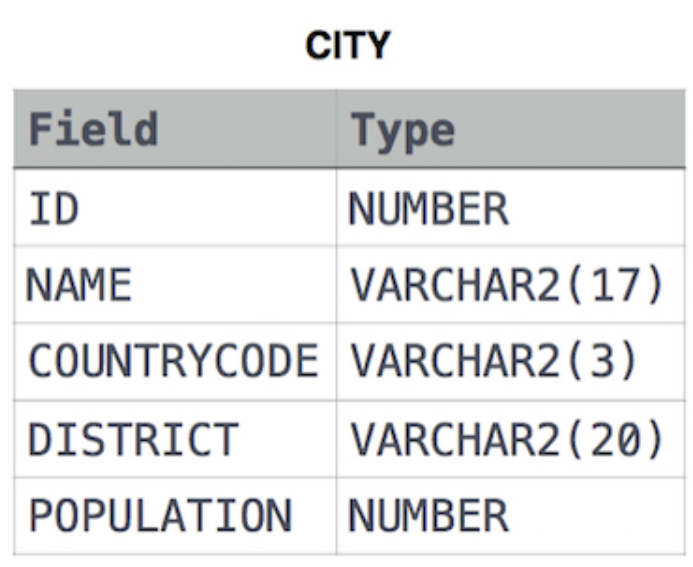
WHERE COUNTRY.Continent = 'Asia';

36) Given the **CITY** and **COUNTRY** tables, query the names of all cities where the *CONTINENT* is *'Africa'*.

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

**Input Format**

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



Solution:

Select CITY.Name From CITY

Inner Join COUNTRY

On CITY.Countrycode = COUNTRY.Code

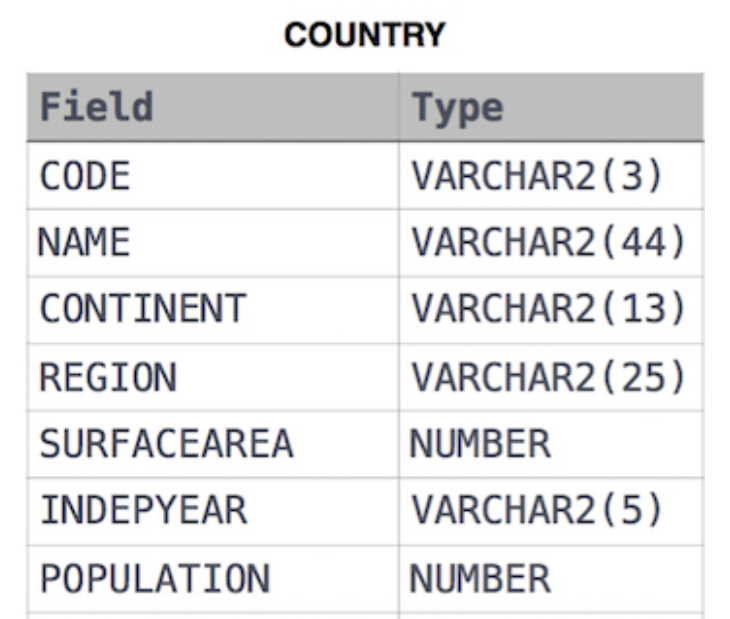
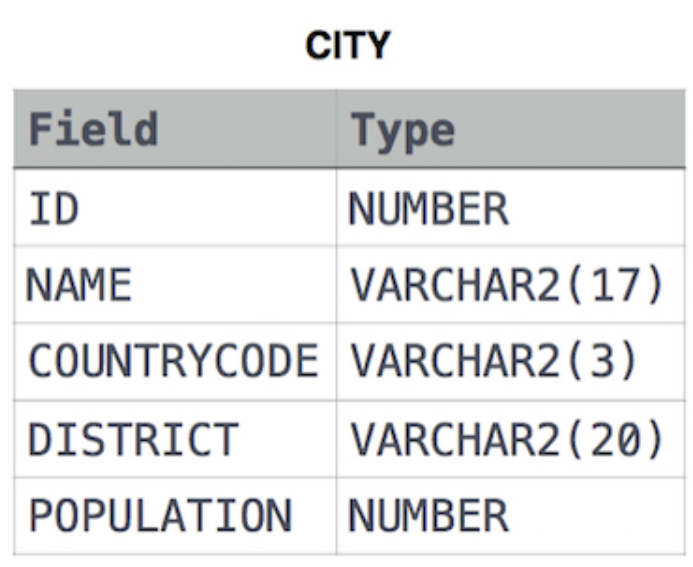
Where COUNTRY.Continent = 'Africa';

37) 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:



Solution:

SELECT COUNTRY.Continent,

FLOOR(AVG(CITY.Population))

FROM COUNTRY

JOIN CITY ON COUNTRY.Code = CITY.CountryCode

GROUP BY COUNTRY.Continent;