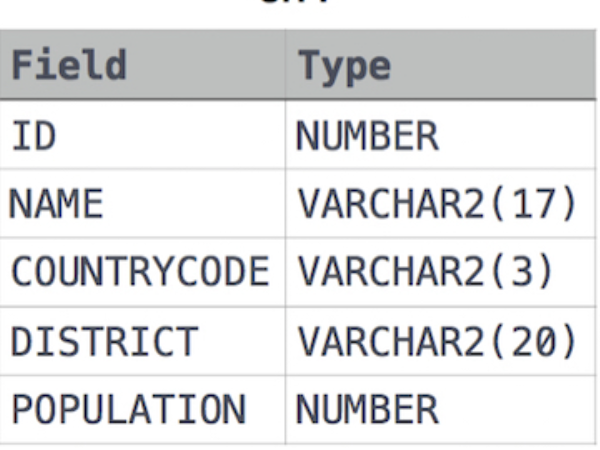
1)Query all columns for all American cities in the **CITY** table with populations larger than 100000. The **CountryCode** for America is USA.

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

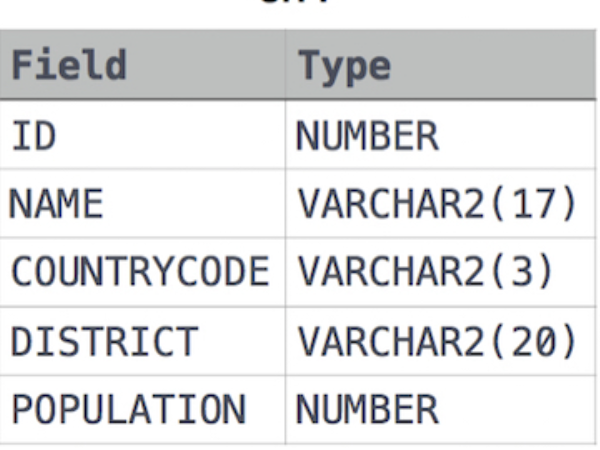


Solution:

SELECT \* FROM CITY WHERE POPULATION > 100000 AND COUNTRYCODE ='USA';

2) Query the **NAME** field for all American cities in the **CITY** table with populations larger than 120000. The **CountryCode** for America is USA.

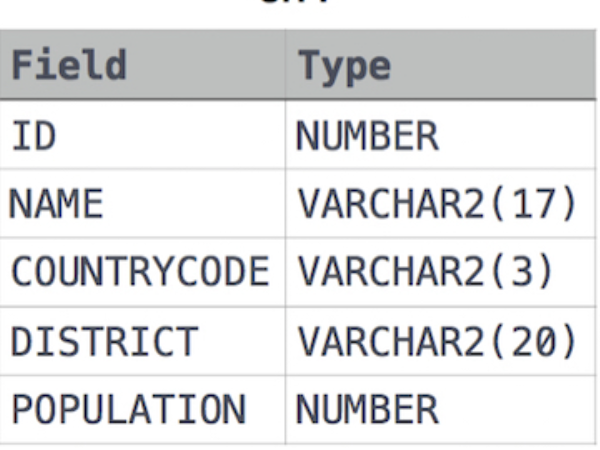
The **CITY** table is described as follows:



SELECT NAME FROM CITY

WHERE COUNTRYCODE ='USA' AND POPULATION > 120000;

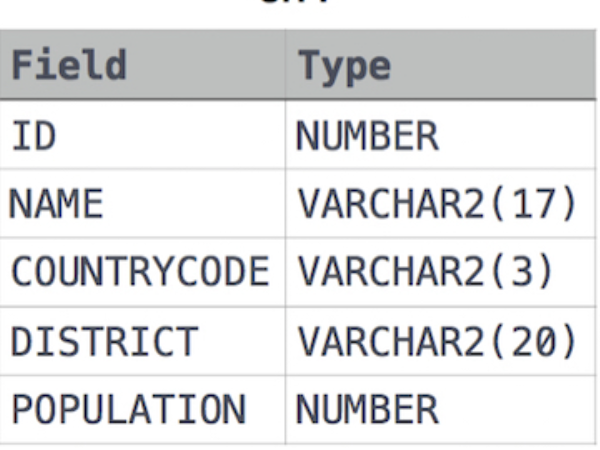
3) Query all columns (attributes) for every row in the **CITY** table.



Solution :

SELECT \* FROM CITY;

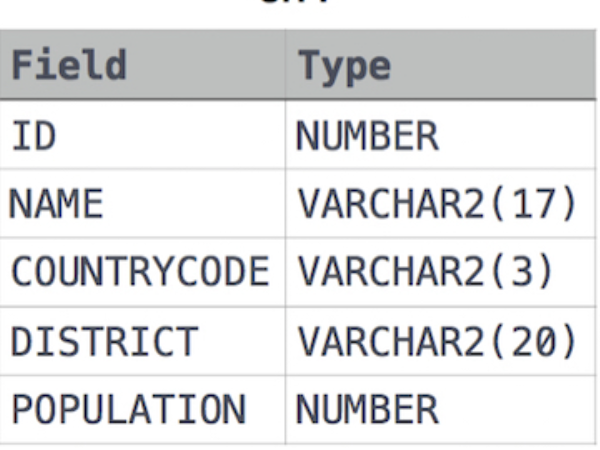
4) Query all attributes of every Japanese city in the **CITY** table. The **COUNTRYCODE** for Japan is JPN.



Solution:

SELECT \* FROM CITY WHERE COUNTRYCODE ='JPN';

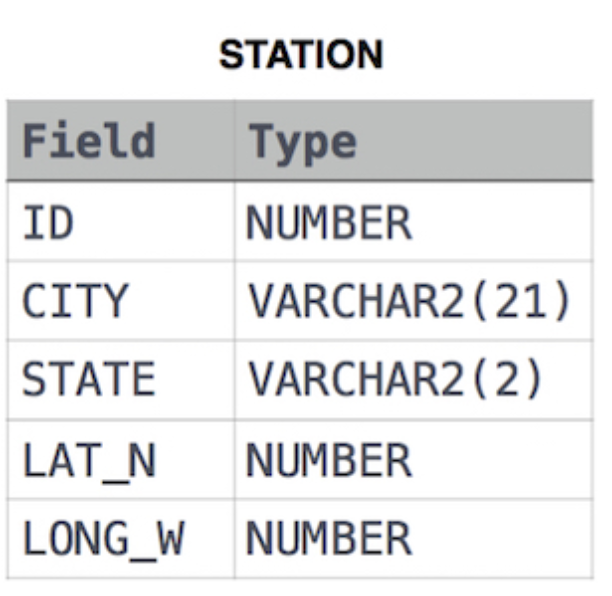
5) Query the names of all the Japanese cities in the **CITY** table. The **COUNTRYCODE** for Japan is JPN.



Solution:

SELECT NAME FROM CITY WHERE COUNTRYCODE ='JPN';

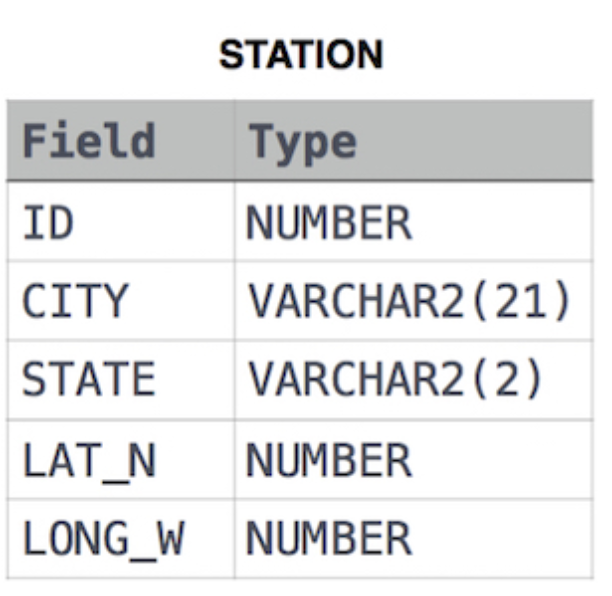
6) Query a list of **CITY** and **STATE** from the **STATION** table.  
The **STATION** table is described as follows:



Solution:

SELECT STATION.CITY, STATION.STATE FROM STATION;

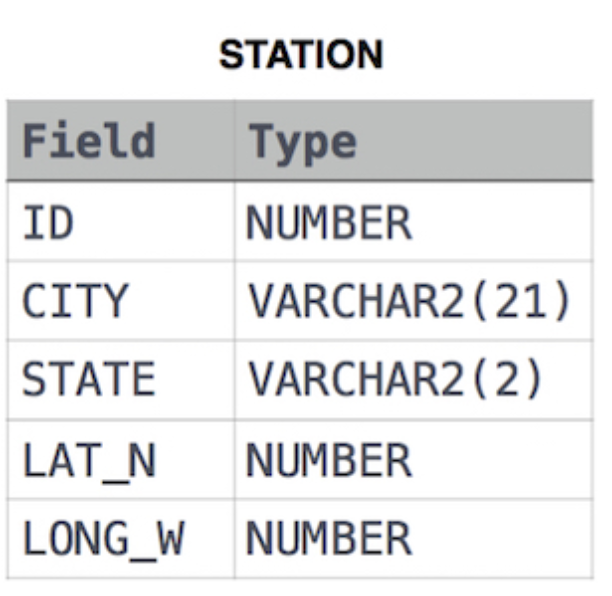
7) Query a list of **CITY** names from **STATION** for cities that have an even **ID** number. Print the results in any order, but exclude duplicates from the answer.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE ID % 2 = 0;

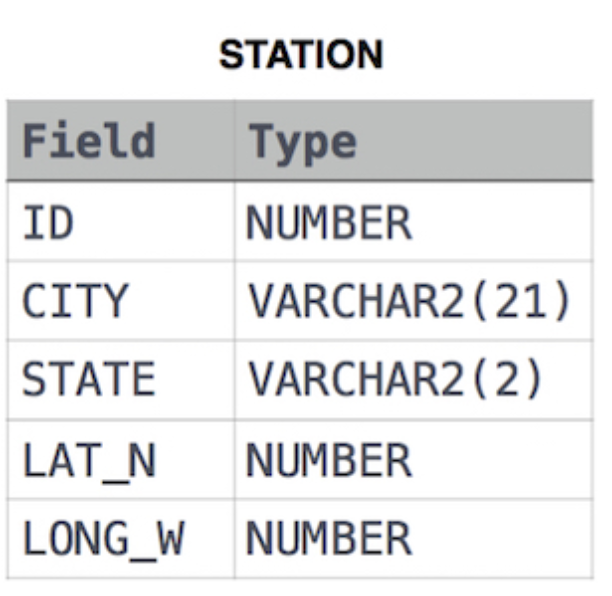
8) Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.



Solution:

SELECT (COUNT(CITY) – COUNT(DISTINCT CITY)) FROM STATION;

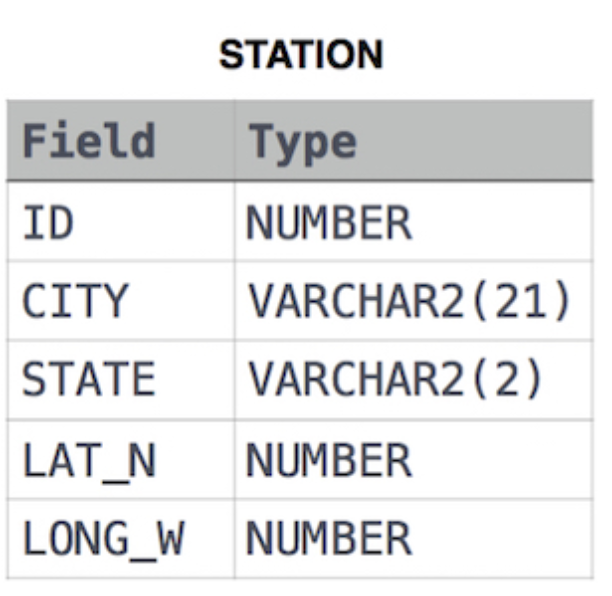
9) Query the list of *CITY* names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result *cannot* contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP '^[aeiouAEIOU]';

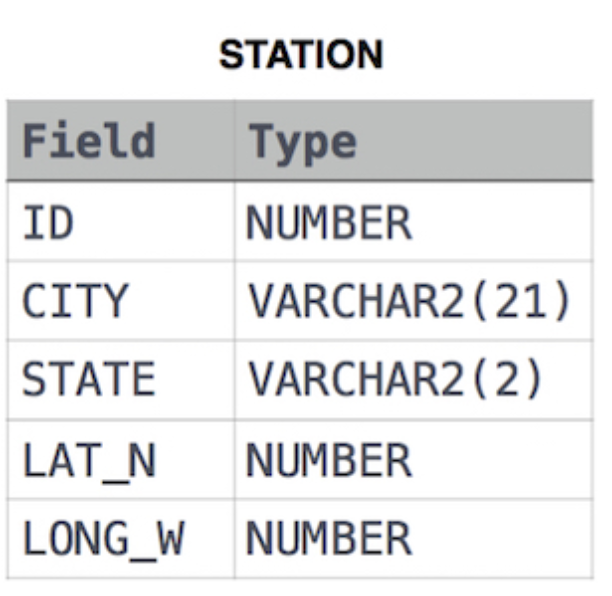
10) Query the list of *CITY* names ending with vowels (a, e, i, o, u) from **STATION**. Your result *cannot* contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP ‘[aeiouAEIOU]$’;

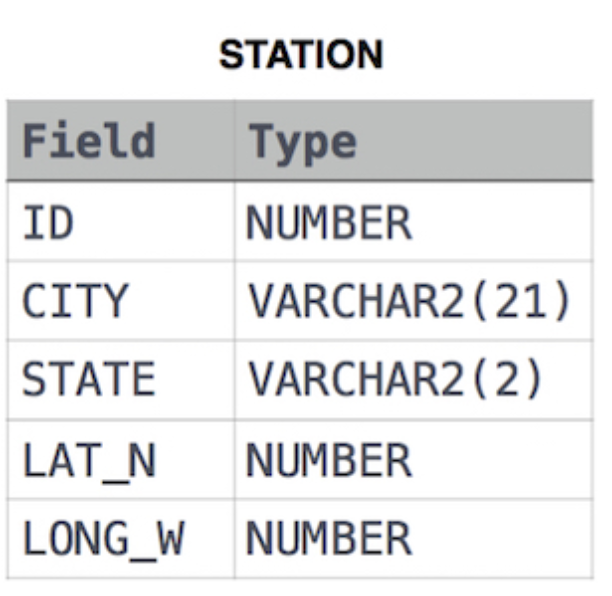
11) Query the list of *CITY* names from **STATION** which have vowels (i.e., *a*, *e*, *i*, *o*, and *u*) as both their first *and* last characters. Your result cannot contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY REGEXP '^[aeiouAEIOU]' AND CITY REGEXP '[aeiouAEIOU]$';

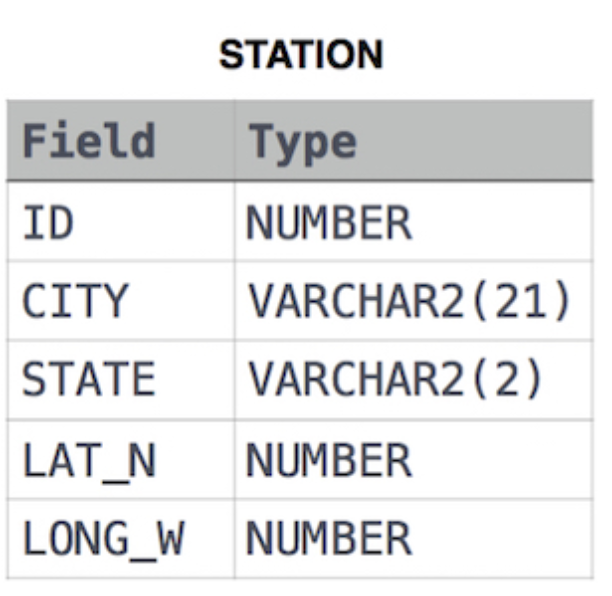
12) Query the list of *CITY* names from **STATION** that *do not start* with vowels. Your result cannot contain duplicates.



Solution :

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT REGEXP ‘^[aeiouAEIOU]’;

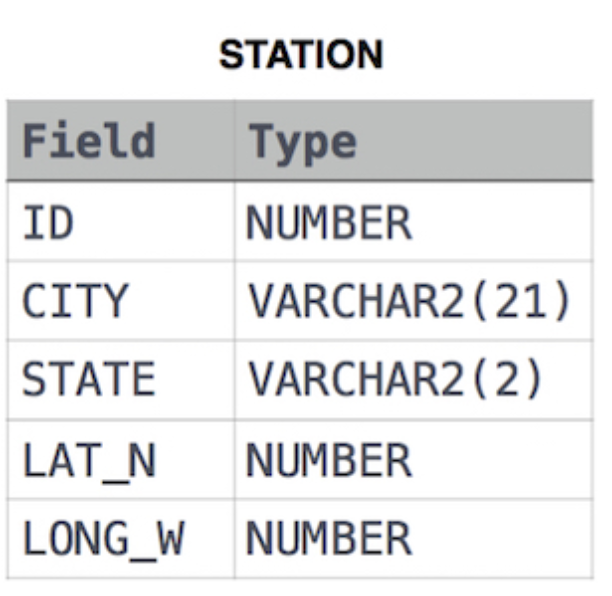
13) Query the list of *CITY* names from **STATION** that *do not end* with vowels. Your result cannot contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT REGEXP ‘[aeiouAEIOU]$’;

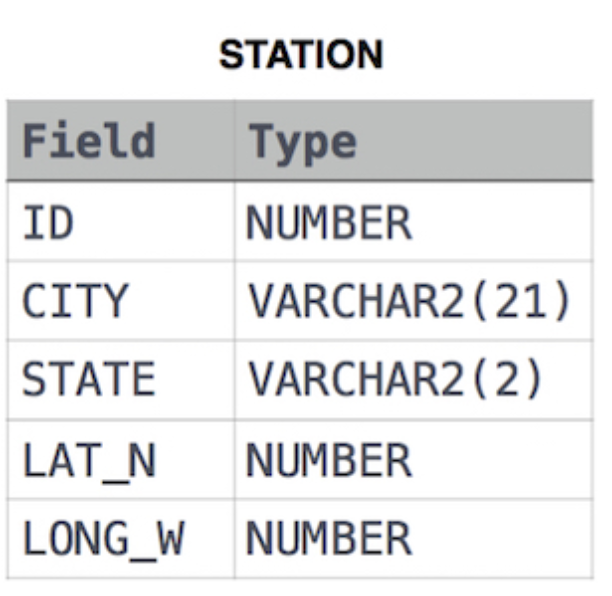
14) Query the list of *CITY* names from **STATION** that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT REGEXP '^[aeiouAEIOU]' OR CITY NOT REGEXP '[aeiouAEIOU]$';

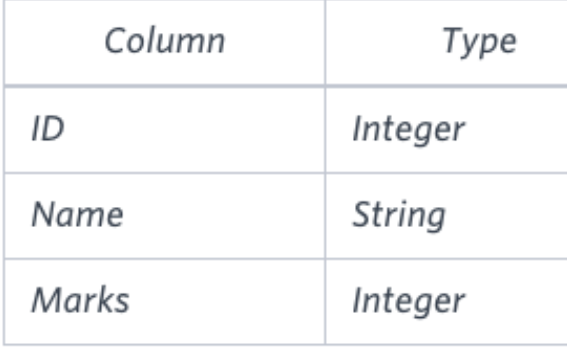
15) Query the list of *CITY* names from **STATION** that *do not start* with vowels and *do not end* with vowels. Your result cannot contain duplicates.



Solution:

SELECT DISTINCT CITY FROM STATION WHERE CITY NOT REGEXP '^[aeiouAEIOU]' AND CITY NOT REGEXP '[aeiouAEIOU]$';

16) Query the *Name* of any student in **STUDENTS** who scored higher than  *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*.

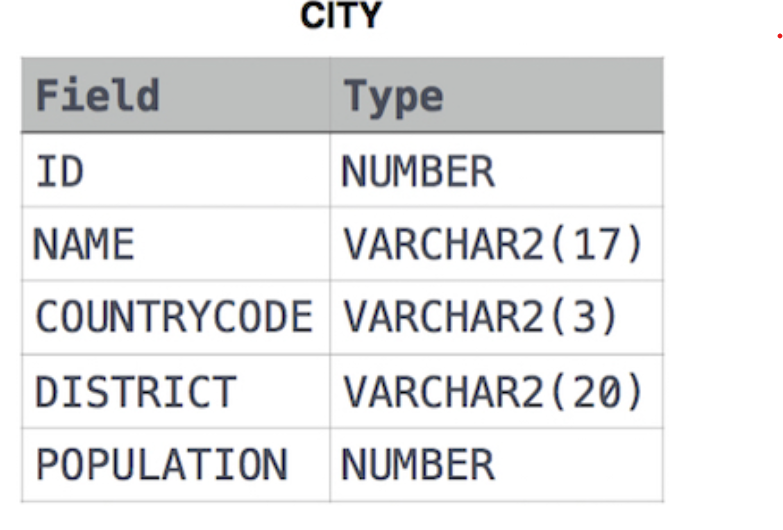


Solution:

SELECT NAME FROM STUDENTS WHERE MARKS > 75 ORDER BY SUBSTR(NAME, -3),ID;

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

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



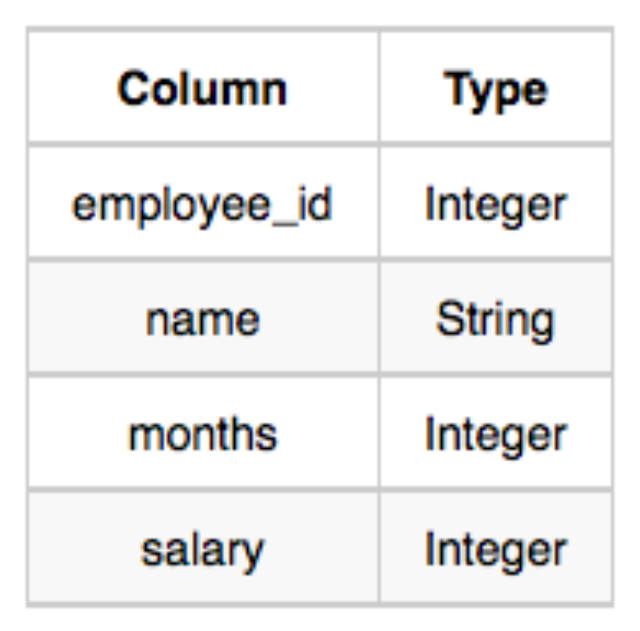
Solution:

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:

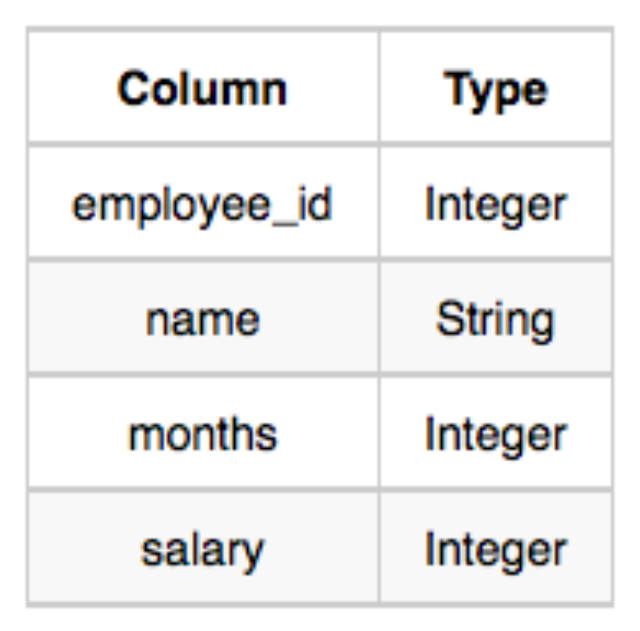


Solution:

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:

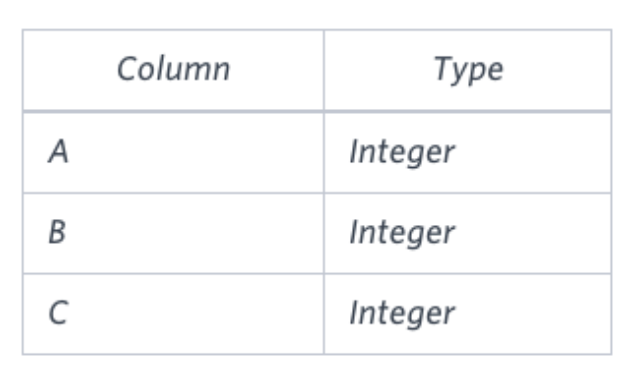
**Equilateral**: It's a triangle with  sides of equal length.

**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;