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

The **CITY** table is described as follows:  
A screenshot of a computer

AI-generated content may be incorrect.

**QUERY:**

**SELECT \* FROM** CITY**;**

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:

A screenshot of a computer

AI-generated content may be incorrect.

**QUERY:**

**SELECT \*FROM** CITY

**WHERE** CITY **= ‘**USA’

**AND** POPULATION **>** 100000**;**

1. **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 followsA screenshot of a computer

AI-generated content may be incorrect.**

**QUERY:**

**SELECT** NAME

**FROM** CITY

**WHERE** COUNTRYCODE **= ‘**USA’

**AND** POPULATION **>120000;**

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

**The CITY table is described as follows:  
A screenshot of a computer

AI-generated content may be incorrect.**

**QUERY:**

**SELECT \* FROM** CITY

**WHERE** ID **=** 1661**;**

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

**The CITY table is described as follows:  
A screenshot of a computer

AI-generated content may be incorrect.**

**QUERY:**

**SELECT \* FROM** CITY

**WHERE** COUNTRYCODE **= ‘**JPN’**;**

1. **Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.  
   The CITY table is described as follows:  
   **

**QUERY:**

**SELECT** NAME

**FROM** CITY

**WHERE** COUNTRYCODE = ‘JPN’;

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

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**QUERY:**

**SELECT** CITY, STATE

**FROM STATION;**

1. **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.  
   The STATION table is described as follows:**

****

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**QUERY:**

**SELECT** CITY

**FROM** STATION

**WHERE MOD(**ID,2**)** = 0**;**

**OR**

**SELECT** CITY

**FROM** STATION

**WHERE** ID%2 =0;

1. Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.  
   The **STATION** table is described as follows:



**QUERY:**

**SELECT COUNT(**CITY**) – COUNT(DISTINCT** CITY**)**

**FROM** STATION**;**

1. **Query the two cities in STATION with the shortest and longest *CITY* names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.  
   The STATION table is described as follows:**

****

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**Sample Input**

**For example, CITY has four entries: DEF, ABC, PQRS and WXY.**

**Sample Output**

**ABC 3**

**PQRS 4**

**Explanation**

**When ordered alphabetically, the CITY names are listed as ABC, DEF, PQRS, and WXY, with lengths  and . The longest name is PQRS, but there are  options for shortest named city. Choose ABC, because it comes first alphabetically.**

**Note  
You can write two separate queries to get the desired output. It need not be a single query.**

**QUERY:**

**(SELECT** CITY, **MAX**(**LENGTH**(CITY))

**FROM** STATION

**GROUP BY** CITY

**ORDER BY MAX**(**LENGTH**(CITY)) **DESC**, CITY **ASC**

**LIMIT 1)**

**UNION**

**(SELECT** CITY, **MIN**(**LENGTH**(CITY))

**FROM** STATION

**GROUP BY** CITY

**ORDER BY MIN**(**LENGTH**(CITY)) **ASC**, CITY **ASC**

**LIMIT 1);**

1. 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:

A table with text and numbers

AI-generated content may be incorrect.

where LAT\_N is the northern latitude and LONG\_W is the western longitude.

**QUERY:**

**SELECT** CITY

**FROM** STATION

**WHERE** CITY **REGEXP ‘^[**AEIOUaeiou**]’;**

1. 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:

A table with text and numbers

AI-generated content may be incorrect.

where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.

**QUERY:**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **REGEXP ‘[**AEIOUaeiou**]$’;**

1. **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.**

**Input Format**

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

****

**where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.**

**QUERY:**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **REGEXP ‘^[**AEIOUaeiou]**.\*[**AEIOUaeiou]**’;**

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

**Input Format**

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

**A table with text and numbers

AI-generated content may be incorrect.**

**where *LAT\_N* is the northern latitude and *LONG\_W* is the western longitude.**

**QUERY:**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **NOT REGEXP ‘^[**AEIOUaeiou]**’;**

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

**Input Format**

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

A table with text and numbers

AI-generated content may be incorrect.

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**QUERY:**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **NOT REGEXP ‘[**AEIOUaeiou**]$’;**

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

**Input Format**

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

****

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**QUERY:**

**SELECT DISITINCT** CITY

**FROM** STATION

**WHERE RIGHT(**CITY**,**1**) NOT IN (**A’, ‘E’, ‘I’, ‘O’, ‘U’, ‘a’, ’e’, ’i’, ’o’, ‘u’);

**OR**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **REGEXP ‘[**AEIOUaeiou**]$’;**

1. **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.**

**Input Format**

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

**A table with text and numbers

AI-generated content may be incorrect.**

**where LAT\_N is the northern latitude and LONG\_W is the western longitude.**

**QUERY:**

**SELECT DISTINCT** CITY

**FROM STATION**

**WHERE** CITY **NOT REGEXP ‘^[**AEIOUaeiou**]’**

**OR** CITY **NOT REGEXP ‘[**AEIOUaeiou**]$’;**

1. 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.

**Input Format**

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



where LAT\_N is the northern latitude and LONG\_W is the western longitude.

**QUERY:**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **NOT REGEXP ‘^[**AEIOUaeiou**].\*[**AEIOUaeiou**]$’;**

* **(Does it meet your requirement? No,** because your requirement is to exclude city names that start or end with vowels. **Your query only excludes city names that start and end with vowels. So, it will include city names that:**
  + **start with a vowel but do not end with a vowel,**
  + **or end with a vowel but do not start with a vowel,**
  + **or neither start nor end with a vowel.)**

**SELECT DISTINCT** CITY

**FROM** STATION

**WHERE** CITY **NOT REGEXP ‘^[**AEIOUaeiou**]’**

**AND** CITY **NOT REGEXP ‘[**AEIOUaeiou**]$’;**

1. 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*.

**Input Format**

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

A table of numbers with text

AI-generated content may be incorrect.

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

**Sample Input**

A screenshot of a computer

AI-generated content may be incorrect.

**Sample Output**

Ashley

Julia

Belvet

**Explanation**

Only Ashley, Julia, and Belvet have *Marks* > . If you look at the last three characters of each of their names, there are no duplicates and 'ley' < 'lia' < 'vet'.

**QUERY:**

**SELECT** NAME

**FROM** STUDENTS

**WHERE MARKS >** 75

**ORDER BY RIGHT(**NAME, 3), **ID ASC;**

1. 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:



where employee\_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is their monthly salary.

**Sample Input**



**Sample Output**

Angela

Bonnie

Frank

Joe

Kimberly

Lisa

Michael

Patrick

Rose

Todd

**QUERY:**

**SELECT** NAME

**FROM** EMPLOYEE

**ORDER BY** NAME;

1. 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.

**Input Format**

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

A table of numbers with black text

AI-generated content may be incorrect.

where employee\_id is an employee's ID number, name is their name, months is the total number of months they've been working for the company, and salary is the their monthly salary.

**Sample Input**



**Sample Output**

Angela

Michael

Todd

Joe

**Explanation**

Angela has been an employee for  month and earns  per month.

Michael has been an employee for  months and earns  per month.

Todd has been an employee for  months and earns  per month.

Joe has been an employee for  months and earns  per month.

We order our output by ascending employee\_id.

**QUERY:**

**SELECT** NAME

**FROM** EMPLOYEE

**WHERE** SALARY **>** 2000

**AND** MONTHS **>** 10

**ORDER BY** EMPLOYEE\_ID**;**