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:



**Solution:**

select name from city where population>=120000 and countrycode='USA'

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

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



**Solution:**

**select \* from city**

**4.** 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**

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

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



**Solution:**

**select \* from city where Countrycode='JPN'**

**6.** 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:



**Solution:**

**select name from city where countrycode='JPN'**

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

****

**Solution:**

**select city,state from station**

**8.**

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:

****

**Solution:**

**select distinct city from station**

**where id % 2 = 0**

**9.**

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:

****

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

For example, if there are three records in the table with **CITY** values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns , because

# Example:Total number of records – number of unique city names =3-2 =1

**Solution:**

**select count(city)-count(distinct city) from station**

**10.**

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 3,3,4  and 3. The longest name is **PQRS**, but there are 3 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.

**Solution:**

**SELECT TOP 1 CITY, LEN(CITY) FROM STATION ORDER BY LEN(CITY),CITY**

**SELECT TOP 1 CITY, LEN(CITY) FROM STATION ORDER BY LEN(CITY) DESC,CITY**

**(there are two queries)**

**11.** 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:

****

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

Solution:

**select distinct city from station**

**where city like "[aeiou]%"**

**12.** 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:

****

Solution:

**select distinct city from station**

**where city like "%[aeiou]"**

**13.** 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.

**Solution:**

**select distinct city from station**

**where city like "[aeiou]%[aeiou]"**

**14.**

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:

****

**Solution:**

**select distinct city from station**

**where city not like "[aeiou]%"**

**15.** 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.

Solution:

**select distinct city from station**

**where city not like "%[aeiou]"**

**16.** 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:

****

**Solution:**

**select distinct city from station**

**where city not like "[aeiou]%" OR city not like "%[aeiou]"**

17.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.

Solution:

**select distinct city from station**

**where city not like "[aeiou]%" AND city not like "%[aeiou]"**

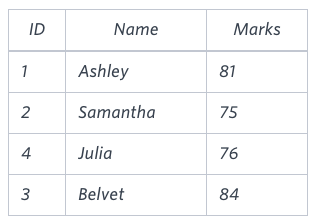
**18.**

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

**Sample Input**



**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'.

**Solution:**

**select Name from students**

**where marks>75**

## ORDER BY RIGHT(NAME, 3), ID ASC

# Right function is use to slice string Name from last

**19.**

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**



**Solution:**

**select name from Employee**

**order by name asc**

**20.**

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:



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 $3445  per month.

Michael has been an employee for  months and earns  $2017 per month.

Todd has been an employee for  months and earns $3396 per month.

Joe has been an employee for  months and earns $3573 per month.

We order our output by ascending employee\_id.

**Solution**:

**select name**

**from employee**

**where (salary >2000 AND months<10) order by employee\_id asc**