---[**Revising the Select Query-1**](https://www.hackerrank.com/challenges/revising-the-select-query)---

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 COUNTRYCODE = 'USA' AND POPULATION > 100000;

###[**Revising the Select Query-2**](https://www.hackerrank.com/challenges/revising-the-select-query-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 COUNTRYCODE = 'USA' AND POPULATION > 120000;

---[**Select All**](https://www.hackerrank.com/challenges/select-all-sql)---Query all columns (attributes) for every row in the **CITY** table. The **CITY** table is described as follows:  


**Solution**

SELECT \* FROM CITY;

---[**Select by ID**](https://www.hackerrank.com/challenges/select-by-id)---

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;

---[**Japanese Cities' Detail**](https://www.hackerrank.com/challenges/japanese-cities-detail)---

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';

---[**Japanese Cities' Name**](https://www.hackerrank.com/challenges/japanese-cities-name)---

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';

---[**Weather Observation Station 1**](https://www.hackerrank.com/challenges/weather-observation-station-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.

**Solution**

SELECT CITY,STATE FROM STATION;

---[**Weather Observation Station 3**](https://www.hackerrank.com/challenges/weather-observation-station-3)---

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.

**Solution**

SELECT DISTINCT CITY FROM STATION WHERE MOD(ID,2)=0 ORDER BY CITY ASC;

---[**Weather Observation Station 4**](https://www.hackerrank.com/challenges/weather-observation-station-4)---

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 .

**Solution**

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

---[**Weather Observation Station 5**](https://www.hackerrank.com/challenges/weather-observation-station-5)---

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.

**Solution**

select city, length(city) from station order by length(city) DESC,city ASC fetch first row only;

select city, length(city) from station order by length(city) asc ,city asc fetch first row only;

---[**Weather Observation Station 6**](https://www.hackerrank.com/challenges/weather-observation-station-6)---

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 'A%' OR CITY LIKE 'E%' OR CITY LIKE 'I%' OR CITY LIKE 'O%'

OR CITY LIKE 'U%' ORDER BY CITY ASC;

---[**Weather Observation Station 7**](https://www.hackerrank.com/challenges/weather-observation-station-7)---

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:



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

**Solution**

SELECT DISTINCT(CITY) FROM STATION WHERE CITY LIKE '%a' OR CITY LIKE '%e' OR CITY LIKE '%i' OR CITY LIKE '%o'

OR CITY LIKE '%u';

---[**Weather Observation Station 8**](https://www.hackerrank.com/challenges/weather-observation-station-8/problem)---

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 'A%' OR CITY LIKE 'E%' OR CITY LIKE 'I%' OR CITY LIKE 'O%' OR CITY LIKE 'U%') AND (CITY LIKE '%a' OR CITY LIKE '%e' OR CITY LIKE '%i' OR CITY LIKE '%o' OR CITY LIKE '%u') order by city;

---[**Weather Observation Station 9**](https://www.hackerrank.com/challenges/weather-observation-station-9/problem)---

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:



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

**Solution**

SELECT DISTINCT CITY FROM STATION WHERE upper(SUBSTR(CITY,1,1)) NOT IN ('A','E','I','O','U') AND lower(SUBSTR(CITY,1,1)) NOT IN

('a','e','i','o','u');

---[**Weather Observation Station 10**](https://www.hackerrank.com/challenges/weather-observation-station-10/problem)---

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 UPPER(SUBSTR(CITY, LENGTH(CITY), 1)) NOT IN ('A','E','I','O','U') AND LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

---[**Weather Observation Station 11**](https://www.hackerrank.com/challenges/weather-observation-station-11/problem)---

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:



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

**Solution**

SELECT DISTINCT CITY FROM STATION WHERE LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') OR LOWER(SUBSTR(CITY, LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

---[**Weather Observation Station 12**](https://www.hackerrank.com/challenges/weather-observation-station-12/problem)---

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 LOWER(SUBSTR(CITY,1,1)) NOT IN ('a','e','i','o','u') AND LOWER(SUBSTR(CITY,LENGTH(CITY),1)) NOT IN ('a','e','i','o','u');

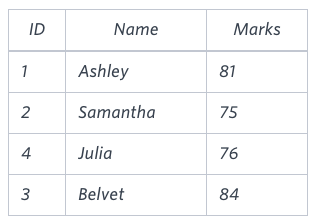
---[**Higher Than 75 marks**](https://www.hackerrank.com/challenges/more-than-75-marks/problem)---

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 SUBSTR(NAME, LENGTH(NAME)-2, 3), ID;

---[**Employee Names**](https://www.hackerrank.com/challenges/name-of-employees/problem)---

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

**Solution**

SELECT NAME FROM EMPLOYEE ORDER BY NAME;

---[**Employee Salaries**](https://www.hackerrank.com/challenges/salary-of-employees/problem)---

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  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*.

**Solution**

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