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

Query all columns for all American cities in CITY with populations larger than 100,000. The CountryCode for America is USA.

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**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 names of all American cities in CITY with populations larger than 120,000. The CountryCode for America is USA.

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**Solution**

SELECT NAME FROM CITY WHERE COUNTRYCODE = 'USA' AND POPULATION > 120000;

###[**Select All**](https://www.hackerrank.com/challenges/select-all-sql)

Query all columns for every row in the CITY table.

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**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.

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**Solution**

SELECT \* FROM CITY WHERE ID = 1661;

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

Query the details for all the Japanese cities in CITY. The COUNTRYCODE for Japan is JPN.

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**Solution**

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

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

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

Input Format

The CITY table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| NAME | VARCHAR2(17) |
| COUNTRY CODE | VARCHAR2(3) |
| DISTRICT | VARCHAR2(20) |
| POPULATION | NUMBER |

**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 STATION.

Input Format

The STATION table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

**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 with even ID numbers only. You may print the results in any order, but must exclude duplicates from your answer.

Input Format

The STATION table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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)

Let NUM be the number of CITY entries in STATION, and NUMunique be the number of unique cities. Query the value of NUM−NUMunique from STATION.

In other words, query the number of non-unique CITY names in STATION by subtracting the number of unique CITY entries in the table from the total number of CITY entries in the table.

Input Format

The STATION table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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

**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.

Input Format

The STATION table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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

*Sample Input*

Let's say that CITY only 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 the respective lengths 3,3,4,3,3,4, and 33. The longest-named city is obviously PQRS, but there are 33 options for shortest-named city; we choose ABC, because it comes first alphabetically.

**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 (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

Input Format

The STATION table is described as follows:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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:

| **Field** | **Type** |
| --- | --- |
| ID | NUMBER |
| CITY | VARCHAR2(21) |
| STATE | VARCHAR2(2) |
| LAT\_N | NUMBER |
| LONG\_W | NUMBER |

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 75 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:

| **Column** | **Type** |
| --- | --- |
| ID | INTEGER |
| NAME | STRING |
| MARKS | INTEGER |

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

Sample Input

| **ID** | **NAME** | **MARKS** |
| --- | --- | --- |
| 1 | ASHLEY | 81 |
| 2 | SAMANTHA | 75 |
| 4 | JULIA | 76 |
| 3 | JULIA | 84 |

Sample Output

Ashley Julia Belvet

Explanation

Only Ashley, Julia, and Belvet have Marks > 75. 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:

| **Column** | **Type** |
| --- | --- |
| employee\_id | INTEGER |
| name | STRING |
| months | INTEGER |
| salary | INTEGER |

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

| **employee\_id** | **name** | **marks** | **salary** |
| --- | --- | --- | --- |
| 12228 | Rose | 15 | 1968 |
| 33645 | Angela | 1 | 3443 |
| 45692 | Frank | 17 | 1608 |
| 56118 | Patrick | 7 | 1345 |
| 59725 | Lisa | 11 | 2330 |
| 74197 | Kimberly | 16 | 4372 |
| 78454 | Bonnie | 8 | 1771 |
| 83565 | Michael | 6 | 2017 |
| 98607 | Todd | 5 | 3396 |
| 99989 | Joe | 9 | 3573 |

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 $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id.

Input Format

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

| **Column** | **Type** |
| --- | --- |
| employee\_id | INTEGER |
| name | STRING |
| months | INTEGER |
| salary | INTEGER |

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

| **employee\_id** | **name** | **marks** | **salary** |
| --- | --- | --- | --- |
| 12228 | Rose | 15 | 1968 |
| 33645 | Angela | 1 | 3443 |
| 45692 | Frank | 17 | 1608 |
| 56118 | Patrick | 7 | 1345 |
| 59725 | Lisa | 11 | 2330 |
| 74197 | Kimberly | 16 | 4372 |
| 78454 | Bonnie | 8 | 1771 |
| 83565 | Michael | 6 | 2017 |
| 98607 | Todd | 5 | 3396 |
| 99989 | Joe | 9 | 3573 |

Sample Output

Angela Michael Todd Joe

Explanation

Angela has been an employee for 1 month and earns $3443 per month. Michael has been an employee for 6 months and earns $2017 per month. Todd has been an employee for 5 months and earns $3396 per month. Joe has been an employee for 9 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;