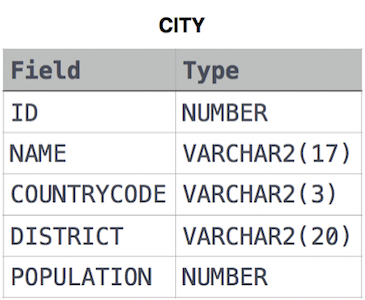
**WEEK 4 ASSIGNMENT**

**TO PERFORM SQL QUERIES ON HACKER RANK**

* **Questions related to SELECT command**

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

****

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

**Solution:**

select Id,Name,CountryCode,District,Population

from CITY

where CountryCode="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**

**Solution:**

select NAME from CITY

where POPULATION>120000 and COUNTRYCODE="USA"

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

**Solution:**

select \*from CITY

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

**Solution:**

select \* from CITY where ID=1661

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

**Solution:**

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.**

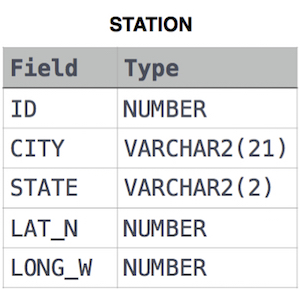
**Solution:**

select name from CITY

where COUNTRYCODE = "JPN"

* **Weather Observation Station Problems**

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

****

1. **Query a list of CITY and STATE from the STATION table.**

**Solution:**

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.**

**Solution:**

select distinct 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.**

**Solution:**

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.**

**Solution** :

select city,length(city)from station order by length(city),city limit 1;

select city,length(city)from station order by length(city) desc 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.**

**Solution 1:**

select distinct city from station

where city REGEXP '^a|^e|^i|^o|^u';

**Solution 2:**

select city from station

where city like 'a%' or

city like 'e%' or

city like 'i%' or

city like 'o%' or

city like 'u%';

1. **Query the list of *CITY* names ending with vowels (a, e, i, o, u) from**

**STATION. Your result *cannot* contain duplicates.**

**Solution 1:**

select distinct city from station

where city REGEXP 'a$|e$|i$|o$|u$';

**Solution 2:**

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';

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.**

**Solution1:**

select distinct city from station

where (city REGEXP '^a|^e|^i|^o|^u')and(city REGEXP 'a$|e$|i$|o$|u$');

**Solution 2 :**

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%');

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

**Solution 1:**

select distinct city from station

where (city not REGEXP '^a|^e|^i|^o|^u')

**Solution 2 :**

select distinct city from station

where city not like 'a%' and

city not like 'e%' and

city not like 'i%' and

city not like 'o%' and

city not like 'u%';

**15.Query the list of *CITY* names from STATION that *do not end* with vowels.**

**Your result cannot contain duplicates.**

**Solution 1:**

select distinct city from station

where (city not REGEXP 'a$|e$|i$|o$|u$')

**Solution 2 :**

select distinct city from station

where city not like '%a' and

city not like '%e' and

city not like '%i' and

city not like '%o' and

city not like '%u';

**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.**

**Solution 1:**

select distinct city from station

where (city not REGEXP '^a|^e|^i|^o|^u')or(city not REGEXP 'a$|e$|i$|o$|u$')

**Solution 2 :**

select distinct city from station

where (city not like 'a%' and

city not like 'e%' and

city not like 'i%' and

city not like 'o%' and

city not like 'u%')

or

(city not like '%a' and

city not like '%e' and

city not like '%i' and

city not like '%o' and

city not like '%u');

**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.**

**Solution 1:**

select distinct city from station

where (city not REGEXP '^a|^e|^i|^o|^u')and(city not REGEXP 'a$|e$|i$|o$|u$')

**Solution 2 :**

select distinct city from station

where (city not like 'a%' and

city not like 'e%' and

city not like 'i%' and

city not like 'o%' and

city not like 'u%')

and

(city not like '%a' and

city not like '%e' and

city not like '%i' and

city not like '%o' and

city not like '%u');

**18. Query the following two values from the STATION table:**

* 1. **The sum of all values in *LAT\_N* rounded to a scale of**

**decimal places.**

**2. The sum of all values in *LONG\_W* rounded to a scale of**

**decimal places.**

**Solution :**

select round(sum(LAT\_N),2),round(sum(LONG\_W),2)

from STATION;

**19. Query the sum of *Northern Latitudes* (*LAT\_N*) from STATION having values greater than 38.7880 and less than 137.2345. Truncate your answer to decimal places.**

**Solution :**

select round(sum(LAT\_N),4)

from STATION

where LAT\_N >38.7880 and LAT\_N <137.2345;

**20. Query the greatest value of the *Northern Latitudes* (*LAT\_N*) from STATION that is less than 137.2345 .Truncate your answer to decimal places.**

**Solution :**

select round(max(LAT\_N),4)from STATION

where LAT\_N<137.2345;

**21. Query the *Western Longitude* (*LONG\_W*) for the largest *Northern Latitude* (*LAT\_N*) in STATION that is less than 137.2345. Round your answer to decimal places.**

**Solution :**

select round(LONG\_W,4)

from STATION where LAT\_N =(select max(LAT\_N)from STATION

where LAT\_N <137.2345);

**22. Query the smallest *Northern Latitude* (*LAT\_N*) from STATION that is greater than 38.7780. Round your answer to decimal places.**

**Solution:**

select round(min(LAT\_N),4)

from STATION

where LAT\_N>38.7780;

**23. Query the *Western Longitude* (*LONG\_W*)where the smallest *Northern Latitude* (*LAT\_N*) in STATION is greater than 38.7780 . Round your answer to 4 decimal places.**

**Solution :**

select round(LONG\_W,4)

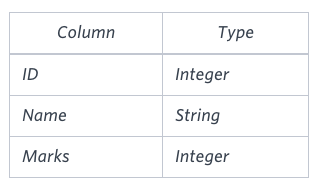
from STATION

where LAT\_N= (select min(LAT\_N)

from STATION

where LAT\_N>38.7780);

**24. 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*.**

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

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

**Solution :**

select Name from students

where Marks >75

order by substr(name,length(name)-2,3),id;

**25. 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.**

**Solution :**

select name from Employee

order by name

**26. 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:**

****

**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.**

**Solution:**

select name from Employee

where salary>2000 and months<10

order by employee\_id

**27. Query a *count* of the number of cities in CITY having a *Population* larger than 100,000.**

**.Input Format**

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

**Solution:**

select count(name) from CITY

where Population>100000;

**28. Query the total population of all cities in CITY where *District* is California.**

**Solution:**

select sum(Population) from CITY

where District="California";

**29. Query the average population of all cities in CITY where *District* is California.**

**Solution:**

select avg(Population) from CITY

where District="California";

**30. Query the average population for all cities in CITY, rounded *down* to the nearest integer.**

**Solution:**

select round(avg(Population),0)

from CITY;

**31. Query the sum of the populations for all Japanese cities in CITY. The *COUNTRYCODE* for Japan is JPN.**

**Solution:**

select sum(Population) from CITY

where COUNTRYCODE="JPN";

**32. Query the difference between the maximum and minimum populations in CITY.**

**Solution:**

select max(Population)-min(Population)

from CITY;