SQL

# 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.

Ans:

select distinct city from station where mod(id,2)=0;

**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.

Ans:

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

**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.

Ans:

select \* from (select city,length(city) from station

order by length(city),city)

where rownum=1

union

select \* from (select city,length(city) from station

order by length(city) DESC,city)

where rownum=1;

**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.

Ans:

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 9

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

Ans:

select distinct city from station where not (city like 'A%' or city like 'E%' or city like 'I%' or city like 'O%' or city like 'U%');

# Weather Observation Station 8

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.

Ans :

select distinct city from station where

regexp\_like(city,'^[AEIOU]','i') and

regexp\_like(city,'[aeiou]$','i');

ORACLE REGEXP\_LIKE SIMILAR TO LIKE

* The following statement returns names that contain the letter ‘c’: REGEXP\_LIKE( first\_name, 'c' )
* The caret (^) operator matches the beginning of the line. The following query returns employees whose first names start with the letter A: REGEXP\_LIKE( last\_name, '^a', 'i' ); Here ‘i’ means case insensitive whereas ‘c’ would mean case sensitive.
* Matching the end of the line : REGEXP\_LIKE( last\_name, '^a', 'i' );
* Matching the end of the line : REGEXP\_LIKE( first\_name, 'y$', 'i' )
* Matching either a or b : first\_name,'^m|^n','i'
* To match a preceding character exactly n times, you use the char{n} pattern. The following example returns the first names that contain exactly two letters L or 'l': REGEXP\_LIKE( first\_name, 'l{2}', 'i' )
* This REGEXP\_LIKE example would retrieve all names that contain the letters ‘b’ or ‘z’ or ‘E’ (case sensitive search) : WHERE regexp\_like (name , '[bzE]') ; and for case sensitive WHERE regexp\_like (name , '[bzE]' , 'i') ;
* And instead of specifying the letters ‘a’, ‘b’ and ‘c’ separately, we can specify a range : WHERE regexp\_like (name , '[a-c]') ;
* The next Oracle REGEP\_LIKE example would retrieve all names that contain a letter in the range of ‘d’ and ‘g’, followed by the letter ‘a’ :WHERE regexp\_like (name , '[d-g]a') ;
* The period (.) operator matches any character except NULL, the next Oracle REGEXP\_LIKE example would retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’. WHERE regexp\_like (name , '[b-g].[a]') ;
* We can use the Period Operator to represent more than one character, the next Oracle REGEXP\_LIKE example would retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any two characters, followed by the letter ‘a’. WHERE regexp\_like (name , '[b-g]..[a]') ;
* .\* means more than one chars. Ex above q can be written as select distinct city from station where regexp\_like(city,'^[AEIOU].\*[aeiou]$');

# Weather Observation Station 10

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

Ans:

select distinct city from station where not regexp\_like(city,'[aeiou]$');

or

select distinct city from station where not regexp\_like(city,'.\*[aeiou]$');

# Weather Observation Station 11

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.

ANS:

select distinct city from station where not (regexp\_like(city,'^[AEIOU].\*[aeiou]$'));

or

select distinct city from station where not(regexp\_like(city,'^[AEIOU]')) or not (regexp\_like(city,'[aeiou]$'));

# Weather Observation Station 12

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.

ANS:

select distinct city from station where not(regexp\_like(city,'^[AEIOU]')) and not (regexp\_like(city,'[aeiou]$'));

# Higher Than 75 Marks

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.

Ans:

select name from students where marks>75 order by substr(name,-3,3) ASC,id ASC;

ex-> name= “abcdefg”;

also substr(name,3,4) = cdef and substr(name,-4,2)=de

# Employee Names

Write a query that prints a list of employee names (i.e.: the name attribute) from the **Employee** table in alphabetical order.

select name from employee order by name;

# Type of Triangle

Write a query identifying the type of each record in the **TRIANGLES** table using its three side lengths. Output one of the following statements for each record in the table:

* **Equilateral**: It's a triangle with 3  sides of equal length.
* **Isosceles**: It's a triangle with 2 sides of equal length.
* **Scalene**: It's a triangle with  all sides of differing lengths.
* **Not A Triangle**: The given values of A, B, and C don't form a triangle.

select case

when

a+b<=c or b+c<=a or a+c<=b

then 'Not A Triangle'

when

(a=c and a=b)

then 'Equilateral'

when

(a=c or a=b or b=c)

then 'Isosceles'

else

'Scalene'

end

from triangles;

# Revising Aggregations - The Count Function

Query a count of the number of cities in **CITY** having a Population larger than .

select count(id) from city where population>100000;

# Revising Aggregations - The Sum Function

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

select sum(population) from city where District='California';

in case of avg population

select avg(population) from city where District='California';

# Average Population

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

select round(avg(population)) from city;

here it’s rounded down so

select floor(avg(population)) from city; can also be used

# Population Density Difference

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

select (max(population)-min(population)) from city;