# BASIC SELECT

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

SELECT DISTINCT CITY FROM STATION WHERE LOWER(SUBSTRING(CITY,1,1)) NOT IN (**'a'**,**'e'**,**'i'**,**'o'**,**'u'**)

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

SELECT DISTINCT CITY FROM STATION WHERE LOWER (SUBSTRING (CITY, LEN(CITY),1)) NOT IN (**'a'**,**'e'**,**'i'**,**'o'**,**'u'**)

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.

SELECT DISTINCT CITY FROM STATION WHERE LOWER (SUBSTRING (CITY, LEN(CITY),1)) NOT IN (**'a'**,**'e'**,**'i'**,**'o'**,**'u'**) OR LOWER (SUBSTRING (CITY, 1,1)) NOT IN (**'a'**,**'e'**,**'i'**,**'o'**,**'u'**)

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.

SELECT name FROM STUDENTS   
where marks>75   
ORDER BY SUBSTRING (name, LEN(name)-2,3), id

# ADVANCED SELECT

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 3 sides of differing lengths.
* **Not A Triangle**: The given values of *A*, *B*, and *C* don't form a triangle.



Note: A, B, C are column names

SELECT  
 CASE  
 WHEN (A + B <= C) OR (B+C <= A) OR (A+C <= B) THEN **"Not A Triangle"** WHEN (A=B) AND (B=C) THEN **"Equilateral"** WHEN (A=B) OR (C=A) OR (B=C) THEN **"Isosceles"** ELSE **"Scalene"** END  
FROM TRIANGLES

**Output**

Isosceles

Equilateral

Scalene

Not A Triangle

Generate the following two result sets:

1. Query an *alphabetically ordered* list of all names in **OCCUPATIONS**, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).
2. Query the number of ocurrences of each occupation in **OCCUPATIONS**. Sort the occurrences in *ascending order*, and output them in the following format:
3. There are a total of [occupation\_count] [occupation]s.

where [occupation\_count] is the number of occurrences of an occupation in **OCCUPATIONS** and [occupation] is the *lowercase* occupation name. If more than one *Occupation* has the same [occupation\_count], they should be ordered alphabetically.

**Note:** There will be at least two entries in the table for each type of occupation.

SELECT CONCAT (name, **'('**, SUBSTRING(occupation, 1, 1), **')'**)  
 FROM occupations  
 ORDER BY name **ASC** ;  
  
SELECT CONCAT (**'There are a total of '**, count(\*), **' '**, LOWER(occupation), **'s.'**)  
 FROM occupations  
 GROUP BY occupation  
 ORDER BY COUNT(\*)

**Sample Output**

Ashely(P)

Christeen(P)

Jane(A)

Jenny(D)

Julia(A)

Ketty(P)

Maria(A)

Meera(S)

Priya(S)

Samantha(D)

There are a total of 2 doctors.

There are a total of 2 singers.

There are a total of 3 actors.

There are a total of 3 professors.