|  |  |
| --- | --- |
| \* | Zero or more instances of string preceding it |
| + | One or more instances of strings preceding it |
| . | Any single character |
| ? | Match zero or one instances of the strings preceding it. |
| ^ | caret(^) matches Beginning of string |
| $ | End of string |
| [abc] | Any character listed between the square brackets |
| [^abc] | Any character not listed between the square brackets |
| [A-Z] | match any upper case letter. |
| [a-z] | match any lower case letter |
| [0-9] | match any digit from 0 through to 9. |
| [[:<:]] | matches the beginning of words. |
| [[:>:]] | matches the end of words. |
| [:class:] | matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters. |
| p1|p2|p3 | Alternation; matches any of the patterns p1, p2, or p3 |
| {n} | n instances of preceding element |
| {m,n} | m through n instances of preceding element |

**Examples with explanation :**

* **Match beginning of string(^):**  
  Gives all the names starting with ‘sa’.Example- sam,samarth.
* SELECT name FROM student\_tbl WHERE name REGEXP '^sa';
* **Match the end of a string($):**  
  Gives all the names ending with ‘on’.Example – norton,merton.

SELECT name FROM student\_tbl WHERE name REGEXP 'on$';

* **Match zero or one instance of the strings preceding it(?):**  
  Gives all the titles containing ‘com’.Example – comedy , romantic comedy.
* SELECT title FROM movies\_tbl WHERE title REGEXP 'com?';
* **matches any of the patterns p1, p2, or p3(p1|p2|p3):**  
  Gives all the names containing ‘be’ or ‘ae’.Example – Abel, Baer.
* SELECT name FROM student\_tbl WHERE name REGEXP 'be|ae' ;
* **Matches any character listed between the square brackets([abc]):**  
  Gives all the names containing ‘j’ or ‘z’.Example – Lorentz, Rajs.
* SELECT name FROM student\_tbl WHERE name REGEXP '[jz]' ;
* **Matches any lower case letter between ‘a’ to ‘z’- ([a-z]) ([a-z] and (.)):**  
  Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’.Example – Tobias, sewall.

Matches any single character(.)

SELECT name FROM student\_tbl WHERE name REGEXP '[b-g].[a]' ;

* **Matches any character not listed between the square brackets.([^abc]):**  
  Gives all the names not containing ‘j’ or ‘z’. Example – nerton, sewall.
* SELECT name FROM student\_tbl WHERE name REGEXP '[^jz]' ;
* **Matches the end of words[[:>:]]:**  
  Gives all the titles ending with character “ack”. Example – Black.
* SELECT title FROM movies\_tbl WHERE REGEXP 'ack[[:>:]]';
* **Matches the beginning of words[[:<:]]:**  
  Gives all the titles starting with character “for”. Example – Forgetting Sarah Marshal.
* SELECT title FROM movies\_tbl WHERE title REGEXP '[[:<:]]for';
* **Matches a character class[:class:]:**  
  i.e [:lower:]- lowercase character ,[:digit:] – digit characters etc.  
  Gives all the titles containing alphabetic character only. Example – stranger things, Avengers.
* SELECT title FROM movies\_tbl WHERE REGEXP '[:alpha:]' ;

===================================================================

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.

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

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

# Higher Than 75 Marks

90 more points to get your next star!

Rank: **275546**|Points: **210/300**

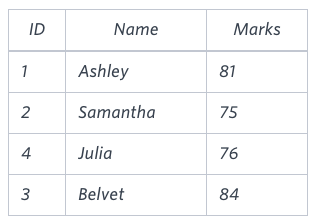
Sql

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'.

**select** name **from** students **where** marks>75 **order** **by** **RIGHT**(name,3),id;

# Employee Names

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.

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

**Input Format**

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



Each row in the table denotes the lengths of each of a triangle's three sides.

**Sample Input**



**Sample Output**

Isosceles

Equilateral

Scalene

Not A Triangle

# Revising Aggregations - The Count Function

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

**Input Format**

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

# 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';

# Revising Aggregations - Averages

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

**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;

# The Blunder

Samantha was tasked with calculating the average monthly salaries for all employees in the **EMPLOYEES** table, but did not realize her keyboard's 0  key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeroes removed), and the actual average salary.

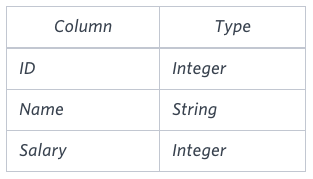
Write a query calculating the amount of error (i.e.:actual-miscalculated   average monthly salaries), and round it up to the next integer.

**Input Format**

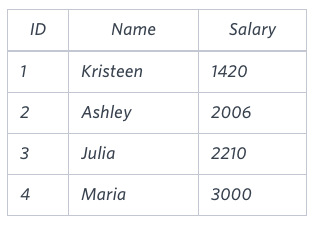
The **EMPLOYEES** table is described as follows:

**Input Format**

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



**Sample Input**

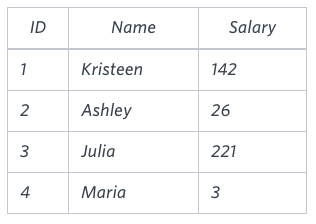


**Sample Output**

2061

**Explanation**

The table below shows the salaries *without zeroes* as they were entered by Samantha:



**select** ceil(avg(salary)-avg(replace(salary,'0',''))) **from** employees;

# Top Earners

We define an employee's total earnings to be their monthly salary\*monthly  worked, and the maximum total earnings to be the maximum total earnings for any employee in the **Employee** table. Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings. Then print these values as 2 space-separated integers.

**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.

**SELECT** (months\*salary) **as** earnings, COUNT(\*) **FROM** Employee **GROUP** **BY** earnings **ORDER** **BY** earnings **DESC** **LIMIT** 1;