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



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

**select** \* **from** city **where** CountryCode='USA' **and** population>100000

Query the **NAME** field for all American cities in the **CITY** table with populations larger than 120000. The CountryCode for America is USA.

**select** name **from** city **where** countryCode='USA' **and** population>120000

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

**select** \*from city;

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

**select** \*from city **where** id=1661

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

**select** \* **from** city **where** CountryCode='JPN'

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

**select** name **from** city **where** CountryCode='JPN'

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


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

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

For example, if there are three records in the table with **CITY** values 'New York', 'New York', 'Bengalaru', there are 2 different city names: 'New York' and 'Bengalaru'. The query returns , because

**select** city, state **from** station;

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.

**select** **distinct**(city) **from** station **where** id%2=0

Find the difference between the total number of **CITY** entries in the table and the number of distinct **CITY** entries in the table.

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

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

**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;

Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result cannot contain duplicates.

**select** **distinct**(city) **from** station **where** city REGEXP '^[aeiou]';

Query the list of CITY names ending with vowels (a, e, i, o, u) from **STATION**. Your result cannot contain duplicates.

**select** **distinct**(city) **from** station **where** city REGEXP '[aeiou]$';

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.

**select** **distinct**(city) **from** station **where** city REGEXP '^[aeiou]' **and** city REGEXP '[aeiou]$';

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** city REGEXP '^[^aeiou]';

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** city REGEXP '[^aeiou]$';

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** city REGEXP '^[^aeiou]' **or** city REGEXP '[^aeiou]$';

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.

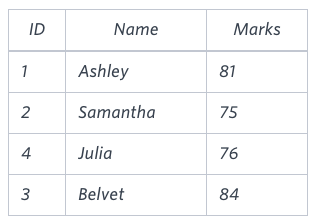
**select** **distinct**(city) **from** station **where** city REGEXP '^[^aeiouAEIOU]' **and** city REGEXP '[^aeiouAEIOU]$';

# Higher Than 75 Marks

**Input Format**

The **STUDENTS** table is described as follows:  The Name column only contains uppercase (A-Z) and lowercase (a-z) letters.

**Sample Input**



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

**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;

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.

**select** name **from** employee **where** salary>2000 **and** months<10 **order** **by** employee\_id;

# 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.

**select** **if**(a+b>c **and** b+c>a **and** a+c>b,

**if**(a=b **and** b=c,

'Equilateral',

**if**(a=b **or** b=c **or** c=a,

'Isosceles',

'Scalene')),

'Not A Triangle')

**from** TRIANGLES;

# Revising Aggregations - The Count Function

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

**Input Format**

# Average Population

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

**select** count(\*) **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';

# 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;

# Japan Population

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

select sum(population) from city where countrycode='JPN';

# Population Density Difference

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

select max(population)-min(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.

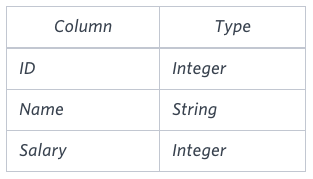
Write a query calculating the amount of error (i.e.:actual-miscalculated   average monthly salaries), and round it up to the next integer.

**Input Format**

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

**Input Format**

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



**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;