Hacker Rank Functions in SQL problem solution

1. 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.

Input Format

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

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where *LAT_N* is the northern latitude and *LONG_W* is the western longitude.

ANSWER:

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%’) or

Not(city like ‘%A’ or city like ‘%E’ or city like ‘%I’ or city like ‘%O’ or city like ‘%U’);

2. 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.

Input Format

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

STATION

Field	Type
ID	NUMBER
CITY	VARCHAR2(21)
STATE	VARCHAR2(2)
LAT_N	NUMBER
LONG_W	NUMBER

where *LAT_N* is the northern latitude and *LONG_W* is the western longitude.

ANSWER:

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%’) and

Not(city like ‘%A’ or city like ‘%E’ or city like ‘%I’ or city like ‘%O’ or city like ‘%U’);

3 . 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:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

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.

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771

employee_id	name	months	salary
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

ANSWER:

SELECT name

FROM employee

ORDER BY name;

4 . 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:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

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.

Explanation

Angela has been an employee for **1** month and earns **$3443** per month.
Michael has been an employee for **6** months and earns **$2017** per month.
Todd has been an employee for **5** months and earns **$3396** per month.
Joe has been an employee for **9** months and earns **$3573** per month.
We order our output by ascending *employee_id*.

ANSWER:

select name **from** employee

where salary > 2000 and months <10

order By employee_id;

5 . 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:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

SELECT COUNT(*)

FROM CITY

WHERE POPULATION > 100000;

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

Input Format

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

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)

Field	Type
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

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

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

Input Format

The **CITY** is described as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

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

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

Input Format

The **CITY** is described as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

```
select round(avg(population),0) from city;
```

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

Input Format

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

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

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

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

Input Format

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

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

ANSWER:

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

11. Given the **CITY** and **COUNTRY** tables, query the sum of the populations of all cities where the *CONTINENT* is *'Asia'*.

Note: *CITY.CountryCode* and *COUNTRY.Code* are matching key columns.

Input Format

The CITY and COUNTRY tables are described as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)

Field	Type
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

CITY

Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)
HEADOFSTATE	VARCHAR2(32)
CAPITAL	VARCHAR2(4)

Field	Type
CODE2	VARCHAR2(2)

ANSWER:

select sum(city.population)

from country **Inner join** city

on country.code = city.countrycode

where country.continent = 'Asia';

12. 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 zeros 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:

Column	Type
ID	*Integer*
Name	*String*
Salary	*Integer*

Note: *Salary* is per month.

Constraints

- **1000 < Salary < 10^5**

Sample Input

Id	Name	Salary
1	*Kristeen*	*1420*
2	*Ashley*	*2006*
3	*Julia*	*2210*
4	*Maria*	*3000*

Sample Output

2061

Explanation

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

Id	Name	Salary
1	*Kristeen*	*142*
2	*Ashley*	*26*
3	*Julia*	*221*
4	*Maria*	*3*

Samantha computes an average salary of **98.00**. The *actual* average salary is **2159.00**.

The resulting error between the two calculations is **2159.00 – 98.00 = 2061.00**. Since it is equal to the integer **2061**, it does not get rounded up.

ANSWER:

```
SELECT CEIL(AVG(Salary)-AVG(REPLACE(Salary,'0','')))
FROM EMPLOYEES;
```

13. We define an employee's *total earnings* to be their monthly ***salary x months*** 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:

Column	Type
employee_id	Integer
name	String
months	Integer
salary	Integer

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.

Sample Input

employee_id	name	months	salary
12228	Rose	15	1968
33645	Angela	1	3443
45692	Frank	17	1608
56118	Patrick	7	1345
59725	Lisa	11	2330
74197	Kimberly	16	4372
78454	Bonnie	8	1771

employee_id	name	months	salary
83565	Michael	6	2017
98607	Todd	5	3396
99989	Joe	9	3573

Sample Output

69952 1

Explanation

The table and earnings data is depicted in the following diagram:

employee_id	name	months	salary	earnings
12228	Rose	15	1968	29520
33645	Angela	1	3443	3443
45692	Frank	17	1608	27336
56118	Patrick	7	1345	9415
59725	Lisa	11	2330	25630
74197	Kimberly	16	4372	69952
78454	Bonnie	8	1771	14168
83565	Michael	6	2017	12102
98607	Todd	5	3396	16980
99989	Joe	9	3573	32157

The maximum *earnings* value is **69952**. The only employee with *earnings*=**69952** is *Kimberly*, so we print the maximum *earnings* value (**69952**) and a count of the number of employees who have earned **$69952** (which is **1**) as two space-separated values.

ANSWER:

select max(months * salary), count(*)

from Employee

where (months * salary) = (**select** max(months * salary) **from** Employee);

14. Given the **CITY** and **COUNTRY** tables, query the names of all cities where the *CONTINENT* is *'Africa'*.

Note: *CITY.CountryCode* and *COUNTRY.Code* are matching key columns.

Input Format

The **CITY** and **COUNTRY** tables are described as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

CITY

Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)

Field	Type
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)
HEADOFSTATE	VARCHAR2(32)
CAPITAL	VARCHAR2(4)
CODE2	VARCHAR2(2)

ANSWER:

```
select city.name
from city inner join country
on city.countrycode = country.code
where country.continent='Africa';
```

15. Given the **CITY** and **COUNTRY** tables, query the names of all the continents (*COUNTRY.Continent*) and their respective average city populations (*CITY.Population*) rounded *down* to the nearest integer.

Note: *CITY.CountryCode* and *COUNTRY.Code* are matching key columns.

Input Format

The **CITY** and **COUNTRY** tables are described as follows:

Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

CITY

Field	Type
CODE	VARCHAR2(3)
NAME	VARCHAR2(44)
CONTINENT	VARCHAR2(13)
REGION	VARCHAR2(25)
SURFACEAREA	NUMBER
INDEPYEAR	VARCHAR2(5)
POPULATION	NUMBER
LIFEEXPECTANCY	VARCHAR2(4)

Field	Type
GNP	NUMBER
GNPOLD	VARCHAR2(9)
LOCALNAME	VARCHAR2(44)
GOVERNMENTFORM	VARCHAR2(44)
HEADOFSTATE	VARCHAR2(32)
CAPITAL	VARCHAR2(4)
CODE2	VARCHAR2(2)

ANSWER:

select country.continent, floor(avg(city.population))

from country **inner join** city

on city.countrycode = country.code

group by country.continent;

16. 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.

Input Format

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

Column	Type
A	*Integer*
B	*Integer*
C	*Integer*

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

Sample Input

A	***B***	***C***
20	20	*23*
20	20	*20*
20	21	*22*
13	*14*	*30*

Sample Output

Isosceles

Equilateral

Scalene

Not A Triangle

Explanation

Values in the tuple **(20, 20, 30)** form an Isosceles triangle, because ***A*** == ***B***. Values in the tuple **(20, 20, 20)** form an Equilateral triangle, because ***A*** == ***B*** == ***C***. Values in the tuple **(20, 21, 22)** form a Scalene triangle, because ***A*** =! ***B*** =! ***C***. Values in the tuple **(13, 14, 30)** cannot form a triangle because the combined value of sides ***A*** and ***B*** is not larger than that of side ***C***.

ANSWER:

Select case

When a+b<=c or a+c<=b or b+c<=a then ‘Not A Triangle’

When a=b and b=c then ‘Equilateral’

When a=b or b=c or c=a then ‘Isosceles’

Else ‘Scalene’

end

from triangles;