Revising Aggregations - The Count Function

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:

CITY

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

SOLUTION –

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**.
Input Format
The **CITY** table is described as follows:

CITY

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

SOLUTION –

SELECT SUM(POPULATION) FROM CITY WHERE DISTRICT = 'California';

Revising Aggregations – Averages

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

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

CITY

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

SOLUTION –

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.

Input Format

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

CITY

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

SOLUTION –

SELECT FLOOR(AVG(POPULATION)) FROM CITY ORDER BY NAME;

SELECT ROUND(AVG(POPULATION)) FROM CITY ORDER BY NAME;

Japan Population

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:

CITY

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

SOLUTION –

SELECT SUM(POPULATION) FROM CITY WHERE COUNTRYCODE = 'JPN';

Population Density Difference

Query the difference between the maximum and minimum populations in **CITY**.
Input Format
The **CITY** table is described as follows:

CITY

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

SOLUTION –

SELECT MAX(POPULATION) - MIN(POPULATION) FROM CITY;

Weather Observation Station 2

Query the following two values from the **STATION** table:

1. The sum of all values in *LAT_N* rounded to a scale of *2* decimal places.
2. The sum of all values in *LONG_W* rounded to a scale of *2* decimal places.

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.

Output Format

Your results must be in the form:
lat lon
where *lat* is the sum of all values in *LAT_N* and *lon* is the sum of all values in *LONG_W*. Both results must be rounded to a scale of *2* decimal places.

SOLUTION –

SELECT ROUND(SUM(LAT_N),2) AS lat, ROUND(SUM(LONG_W),2) AS lon FROM STATION;

Weather Observation Station 13

Query the sum of *Northern Latitudes* (*LAT_N*) from **STATION** having values greater than *38.7880* and less than *137.2345*. Truncate your answer to *4* decimal places.

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.

SOLUTION –

SELECT ROUND(SUM(LAT_N),4) FROM STATION WHERE LAT_N BETWEEN 38.7880 AND 137.2345;

SELECT ROUND(SUM(LAT_N),4) FROM STATION WHERE LAT_N > 38.7880 AND LAT_N < 137.2345;

Weather Observation Station 14

Query the greatest value of the *Northern Latitudes* (*LAT_N*) from **STATION** that is less than *137.2345*. Truncate your answer to *4* decimal places.

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.

SOLUTION –

SELECT ROUND(MAX(LAT_N),4) FROM STATION WHERE LAT_N < 137.2345;

Weather Observation Station 15

Query the *Western Longitude* (*LONG_W*) for the largest *Northern Latitude* (*LAT_N*) in **STATION** that is less than *137.2345*. Round your answer to *4* decimal places.

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.

SOLUTION –

SELECT ROUND(LONG_W,4) FROM STATION WHERE LAT_N = (SELECT MAX(LAT_N) FROM STATION WHERE LAT_N < 137.2345);

Weather Observation Station 16

Query the smallest *Northern Latitude* (*LAT_N*) from **STATION** that is greater than *38.7780*. Round your answer to *4* decimal places.

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.

SOLUTION –

SELECT ROUND(MIN(LAT_N),4) FROM STATION WHERE LAT_N > 38.7780;

Weather Observation Station 17

Query the *Western Longitude* (*LONG_W*) where the smallest *Northern Latitude* (*LAT_N*) in **STATION** is greater than *38.7780*. Round your answer to *4* decimal places.

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.

SOLUTION –

SELECT ROUND(LONG_W,4) FROM STATION WHERE LAT_N = (SELECT MIN(LAT_N) FROM STATION WHERE LAT_N > 38.7780);

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 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.

Top Earners

We define an employee's *total earnings* to be their monthly *salary * 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
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 . The only employee with *earnings* is *Kimberly*, so we print the maximum *earnings* value () and a count of the number of employees who have earned (which is) as two space-separated values.