SQL Practice Questions

Q.1: Query all columns for all American cities in **CITY** with populations larger than 100000. The *CountryCode* for America is USA. **(Mode of Difficulty: Easy)**

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

Q.2. Query the names of all American cities in **CITY** with populations larger than 120000. The *CountryCode* for America is USA. **(Mode of Difficulty:Easy)**

Q.3. Query all columns (attributes) for every row in the **CITY** table. **(Mode of Difficulty: Easy)**

Q.4. Query all columns for a city in **CITY** with the *ID* 1661. **(Mode of Difficulty: Easy)**

Q.5. Query all attributes of every Japanese city in the **CITY** table. The *COUNTRYCODE* for Japan is JPN. **(Mode of Difficulty: Easy)**

Q.6. Query the names of all the Japanese cities in the **CITY** table. The *COUNTRYCODE* for Japan is JPN. **(Mode of Difficulty: Easy)**

Q.7. Query a list of *CITY* and *STATE* from the **STATION** table.
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. **(Mode of Difficulty: Easy)**

Q.8. Query a list of *CITY* names from **STATION** with even *ID* numbers only. You may print the results in any order, but must exclude duplicates from your answer. **(Mode of Difficulty: Easy)**

Q.9. Let N be the number of *CITY* entries in **STATION**, and let N' be the number of distinct *CITY* names in **STATION**; query the value of N-N' from **STATION**. In other words, find the difference between the total number of *CITY* entries in the table and the number of distinct *CITY* entries in the table. **(Mode of Difficulty: Easy)**

Q.10. 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. **(Mode of Difficulty: Easy)**

Q.11. Query the list of *CITY* names starting with vowels (i.e., a, e, i, o, or u) from **STATION**. Your result *cannot* contain duplicates. **(Mode of Difficulty: Easy)**

Q.12. Query the list of *CITY* names ending with vowels (a, e, i, o, u) from **STATION**. Your result *cannot* contain duplicates. **(Mode of Difficulty: Easy)**

Q.13 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. **(Mode of Difficulty: Easy)**

Q.14. Query the list of *CITY* names from **STATION** that *do not start* with vowels. Your result cannot contain duplicates. **(Mode of Difficulty: Easy)**

Q.15. Query the list of *CITY* names from **STATION** that *do not end* with vowels. Your result cannot contain duplicates. **(Mode of Difficulty: Easy)**

Q.16. 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. **(Mode of Difficulty: Easy)**

Q.17. 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. **(Mode of Difficulty: Easy)**

Q.18. 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*. **(Mode of Difficulty: Easy)**

Input Format
The **STUDENTS** table is described as follows:

Column	*Type*
ID	*Integer*
Name	*String*
Marks	*Integer*

Q.19. Write a query that prints a list of employee names (i.e.: the *name* attribute) from the **Employee** table in alphabetical order. **(Mode of Difficulty: Easy)**

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.

Q.20. 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*. **(Mode of Difficulty: Easy)**

Q.21. Query a *count* of the number of cities in **CITY** having a *Population* larger than 100000. **(Mode of Difficulty: Easy)**

Q.22. Query the total population of all cities in CITY where District is California. **(Mode of Difficulty: Easy)**

Q.23. Query the average population of all cities in **CITY** where *District* is **California**. **(Mode of Difficulty: Easy)**

Q.24. Query the average population for all cities in **CITY**, rounded *down* to the nearest integer. **(Mode of Difficulty: Easy)**

Q.25. Query the sum of the populations for all Japanese cities in **CITY**. The *COUNTRYCODE* for Japan is **JPN**. **(Mode of Difficulty: Easy)**

Q.26. Query the difference between the maximum and minimum populations in CITY. **(Mode of Difficulty: Easy)**

Q.27. 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. **(Mode of Difficulty: Easy)**

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

Column	Type
ID	Integer
Name	String
Salary	Integer

Note: *Salary* is measured in dollars per month and its value is < 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 zeroes* 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 which, when rounded to the next integer, is 2061. **(Mode of Difficulty: Easy)**

Q.28. 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. **(Mode of Difficulty: Easy)**

Q.29. Query the following two values from the STATION table:
The sum of all values in LAT_N rounded to a scale of 2 decimal places.
The sum of all values in LONG_W rounded to a scale of 2 decimal places. **(Mode of Difficulty: Easy)**

Q.30. 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. **(Mode of Difficulty: Easy)**

Q.31. 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. **(Mode of Difficulty: Easy)**

Q.32. 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. **(Mode of Difficulty: Easy)**

Q.33. Query the smallest Northern Latitude (LAT_N) from STATION that is greater than 38.7780. Round your answer to 4 decimal places. **(Mode of Difficulty: Easy)**

Q.34. 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. **(Mode of Difficulty: Easy)**

Q.35. Consider P1(a,b) and P2(c,d) to be two points on a 2D plane.
A happens to equal the minimum value in Northern Latitude (LAT_N in STATION).
B happens to equal the minimum value in Western Longitude (LONG_W in STATION).
C happens to equal the maximum value in Northern Latitude (LAT_N in STATION).
D happens to equal the maximum value in Western Longitude (LONG_W in STATION).
Query the Manhattan Distance between points P1 and P2 and round it to a scale of 4 decimal places.
(Manhattan Distance:- |x1 – x2| + |y1 – y2|). **(Mode of Difficulty: Easy)**

Q.36. Consider P1(a,b) and P2(c,d) to be two points on a 2D plane where (a,b) are the respective minimum and maximum values of Northern Latitude (LAT_N) and (c,d) are the respective minimum and maximum values of Western Longitude (LONG_W) in STATION. Query the Euclidean Distance between points and and format your answer to display 4 decimal digits. **(Mode of Difficulty: Easy)**

Q.37. A median is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern Latitudes (LAT_N) from STATION and round your answer to 4 decimal places. **(Mode of Difficulty: Easy)**

Q.38. Pivot the Occupation column in OCCUPATIONS so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively. Note: Print NULL when there are no more names correspond. **(Mode of Difficulty: Easy)**

Q.39. New_Companies (Mode of Difficulty: Medium)
Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:
FOUNDER => LEAD MANAGER => SENIOR MANAGER => MANAGER => EMPLOYEE
Given the table schemas below, write a query to print the company_code, founder name, total number of lead managers, total number of senior managers,
total number of managers, and total number of employees. Order your output by ascending company_code.

Q.40. Write a query to print all *prime numbers* less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).

For example, the output for all prime numbers <= 10 would be: 2&3&5&7. **(Mode of Difficulty: Medium)**

Q.41. Type of Triangle (Mode of Difficulty: Easy)

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:

Column	Type
A	Integer
B	Integer
C	Integer

Sample Input

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

Explanation

Values in the tuple(20,20,23) 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 .

Q.42. The PADS (Mode of Difficulty: Medium)

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:

   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.

Input Format

Column	Type
Name	String
Occupation	String

The **OCCUPATIONS** table is described as follows: *Occupation* will only contain one of the following values: **Doctor**, **Professor**, **Singer** or **Actor**.

Explanation

The results of the first query are formatted to the problem description's specifications.

The results of the second query are ascendingly ordered first by number of names corresponding to each profession (2 **≤2 ≤3 ≤3**), and then alphabetically by profession (docter **≤**singer , and actor **≤** professor)

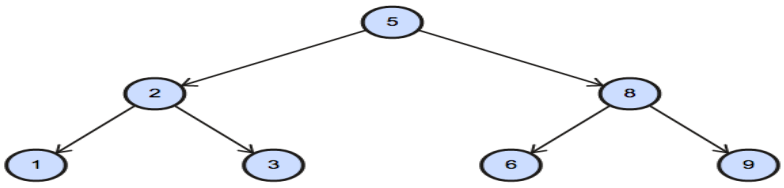
Q.43. Binary Tree Nodes (Mode of Difficulty: Medium)

You are given a table, *BST*, containing two columns: *N* and *P,* where *N* represents the value of a node in *Binary Tree*, and *P* is the parent of *N*.

Column	Type
N	Integer
P	Integer

Write a query to find the node type of *Binary Tree* ordered by the value of the node. Output one of the following for each node:

- *Root*: If node is root node.
- *Leaf*: If node is leaf node.
- *Inner*: If node is neither root nor leaf node.
- **Explanation**
- The *Binary Tree* below illustrates the sample:



Q.44. The Report (Mode of Difficulty: Medium)

You are given two tables: *Students* and *Grades*. *Students* contains three columns *ID*, *Name* and *Marks*.

Column	Type
ID	Integer
Name	String
Marks	Integer

Grades contains the following data:

Grade	Min_Mark	Max_Mark
1	0	9
2	10	19
3	20	29
4	30	39
5	40	49
6	50	59
7	60	69
8	70	79
9	80	89
10	90	100

Ketty gives *Eve* a task to generate a report containing three columns: *Name*, *Grade* and *Mark*. *Ketty* doesn't want the NAMES of those students who received a grade lower than *8*. The report must be in descending order by grade -- i.e. higher grades are entered first. If there is more than one student with the same grade (8-10) assigned to them, order those particular students by their name alphabetically. Finally, if the grade is lower than 8, use "NULL" as their name and list them by their grades in descending order. If there is more than one student with the same grade (1-7) assigned to them, order those particular students by their marks in ascending order.

Write a query to help Eve.

Note

Print "NULL" as the name if the grade is less than 8.

Explanation

Consider the following table with the grades assigned to the students:

ID	Name	Marks	Grade
1	Julia	88	9
2	Samantha	68	7
3	Maria	99	10
4	Scarlet	78	8
5	Ashley	63	7
6	Jane	81	9

So, the following students got *8*, *9* or *10* grades:

- *Maria (grade 10)*
- *Jane (grade 9)*

- *Julia (grade 9)*
- *Scarlet (grade 8)*

Current Buffer (saved locally, editable)

Q.45. Top Competitors (Mode of Difficulty: Medium)

Julia just finished conducting a coding contest, and she needs your help assembling the leaderboard! Write a query to print the respective *hacker_id* and *name* of hackers who achieved full scores for *more than one* challenge. Order your output in descending order by the total number of challenges in which the hacker earned a full score. If more than one hacker received full scores in same number of challenges, then sort them by ascending *hacker_id*.

Input Format

The following tables contain contest data:

- *Hackers:* The *hacker_id* is the id of the hacker, and *name* is the name of the hacker.

Column	Type
hacker_id	Integer
name	String

- *Difficulty:* The *difficult_level* is the level of difficulty of the challenge, and *score* is the score of the challenge for the difficulty level.

Column	Type
difficulty_level	Integer
score	Integer

- *Challenges:* The *challenge_id* is the id of the challenge, the *hacker_id* is the id of the hacker who created the challenge, and *difficulty_level* is the level of difficulty of the challenge.

Column	Type
challenge_id	Integer
hacker_id	Integer
difficulty_level	Integer

- *Submissions:* The *submission_id* is the id of the submission, *hacker_id* is the id of the hacker who made the submission, *challenge_id* is the id of the challenge that the

submission belongs to, and *score* is the score of the submission.

Column	Type
submission_id	Integer
hacker_id	Integer
challenge_id	Integer
score	Integer

Sample Output

90411 Joe

Explanation

- Hacker *86870* got a score of *30* for challenge *71055* with a difficulty level of *2*, so *86870* earned a full score for this challenge.
- Hacker *90411* got a score of *30* for challenge *71055* with a difficulty level of *2*, so *90411* earned a full score for this challenge.
- Hacker *90411* got a score of *100* for challenge *66730* with a difficulty level of *6*, so *90411* earned a full score for this challenge.
- Only hacker *90411* managed to earn a full score for more than one challenge, so we print the their *hacker_id* and *name* as 2 space-separated values.

Q.46.Asian Population (Mode of Difficulty: Easy)

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

Q.47.African Cities (Mode of Difficulty: Easy)

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

Q.48.Average Population of Each Continent ((Mode of Difficulty: Easy)

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.

Q.49. Contest Leaderboard (Mode of Difficulty: Medium)

You did such a great job helping Julia with her last coding contest challenge that she wants you to work on this one, too! The total score of a hacker is the sum of their maximum scores for all of the challenges. Write a query to print the *hacker_id*, *name*, and total score of the

hackers ordered by the descending score. If more than one hacker achieved the same total score, then sort the result by ascending *hacker_id*. Exclude all hackers with a total score of 0 from your result.(Tables are same as given in Top Competitors)

Explanation

Hacker *4071* submitted solutions for challenges *19797* and *49593*, so the total score =95+max(43,96)=191.

Hacker *74842* submitted solutions for challenges *19797* and *63132*, so the total score=max(98,5)+76=174

Hacker *84072* submitted solutions for challenges *49593* and *63132*, so the total score=100+0=100 .

The total scores for hackers *4806*, *26071*, *80305*, and *49438* can be similarly calculated.

Q.50. Challenges (Mode of Difficulty: Medium)

Julia asked her students to create some coding challenges. Write a query to print the *hacker_id*, *name*, and the total number of challenges created by each student. Sort your results by the total number of challenges in descending order. If more than one student created the same number of challenges, then sort the result by *hacker_id*. If more than one student created the same number of challenges and the count is less than the maximum number of challenges created, then exclude those students from the result.

Input Format

The following tables contain challenge data:

- *Hackers:* The *hacker_id* is the id of the hacker, and *name* is the name of the hacker.

Column	Type
hacker_id	Integer
name	String

- *Challenges:* The *challenge_id* is the id of the challenge, and *hacker_id* is the id of the student who created the challenge.

Column	Type
challenge_id	Integer
hacker_id	Integer

- **Explanation**

- For *Sample Case 0*, we can get the following details:

hacker_id	name	challenges_created
21283	Angela	6
88255	Patrick	5
5077	Rose	4
62743	Frank	4
96196	Lisa	1

Students 5077 and 62743 both created 4 challenges, but the maximum number of challenges created is 6 so these students are excluded from the result.

- For *Sample Case 1*, we can get the following details:

hacker_id	name	challenges_created
12299	Rose	6
34856	Angela	6
79345	Frank	4
80491	Patrick	3
81041	Lisa	1

Students 12299 and 34856 both created 6 challenges. Because 6 is the maximum number of challenges created, these students are included in the result.

Q.51. Draw The Triangle 1 (Mode of Difficulty: Easy)

P(R) represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

```
* * * * *
* * * *
* * *
* *
*
```

Write a query to print the pattern *P(20)*.

Q.52. Draw The Triangle 2 (Mode of Difficulty: Easy)

P(R) represents a pattern drawn by Julia in *R* rows. The following pattern represents *P(5)*:

```
*
* *
* * *
* * * *
* * * * *
```

Write a query to print the pattern *P(20)*.

Q.53. PLACEMENTS (Mode of Difficulty: Medium)

You are given three tables: *Students*, *Friends* and *Packages. Students* contains two columns: *ID* and *Name*. *Friends* contains two columns: *ID* and *Friend_ID* (*ID* of the ONLY best friend). *Packages* contains two columns: *ID* and *Salary* (offered salary in $ thousands per month).

Column	Type
ID	Integer
Name	String

Students

Column	Type
ID	Integer
Friend_ID	Integer

Friends

Column	Type
ID	Integer
Salary	Float

Packages

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

Sample Input

ID	Friend_ID
1	2
2	3
3	4
4	1

Friends

ID	Name
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Students

ID	Salary
1	15.20
2	10.06
3	11.55
4	12.12

Packages

Sample Output
Samantha
Julia
Scarlet

Explanation
See the following table:

ID	1	2	3	4
Name	Ashley	Samantha	Julia	Scarlet
Salary	15.20	10.06	11.55	12.12
Friend ID	2	3	4	1
Friend Salary	10.06	11.55	12.12	15.20

Now,

- *Samantha's* best friend got offered a higher salary than her at 11.55
- *Julia's* best friend got offered a higher salary than her at 12.12
- *Scarlet's* best friend got offered a higher salary than her at 15.2
- *Ashley's* best friend did NOT get offered a higher salary than her

The name output, when ordered by the salary offered to their friends, will be:

- *Samantha*
- *Julia*
- *Scarlet*

Q.54. PROJECT_PLANNING (Mode of Difficulty: Medium)

You are given a table, *Projects*, containing three columns: *Task_ID*, *Start_Date* and *End_Date*. It is guaranteed that the difference between the *End_Date* and the *Start_Date* is equal to *1* day for each row in the table.

Column	Type
Task_ID	Integer
Start_Date	Date
End_Date	Date

If the *End_Date* of the tasks are consecutive, then they are part of the same project. Samantha is interested in finding the total number of different projects completed. Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order. If there is more than one project that have the same number of completion days, then order by the start date of the project.

Sample Input

Task_ID	Start_Date	End_Date
1	2015-10-01	2015-10-02
2	2015-10-02	2015-10-03
3	2015-10-03	2015-10-04
4	2015-10-13	2015-10-14
5	2015-10-14	2015-10-15
6	2015-10-28	2015-10-29
7	2015-10-30	2015-10-31

Sample Output
2015-10-28 2015-10-29
2015-10-30 2015-10-31
2015-10-13 2015-10-15
2015-10-01 2015-10-04

Explanation
The example describes following *four* projects:

- ***Project 1*: Tasks *1*, *2* and *3* are completed on consecutive days, so these are part of the project. Thus start date of project is *2015-10-01* and end date is *2015-10-04*, so it took *3 days* to complete the project.**
- ***Project 2*: Tasks *4* and *5* are completed on consecutive days, so these are part of the project. Thus, the start date of project is *2015-10-13* and end date is *2015-10-15*, so it took *2 days* to complete the project.**
- ***Project 3*: Only task *6* is part of the project. Thus, the start date of project is *2015-10-28* and end date is *2015-10-29*, so it took *1 day* to complete the project.**
- ***Project 4*: Only task *7* is part of the project. Thus, the start date of project is *2015-10-30* and end date is *2015-10-31*, so it took *1 day* to complete the project.**

Q.55. OLLIVANDER'S INVENTORY (Mode of Difficulty: Medium) (doubt)

Harry Potter and his friends are at Ollivander's with Ron, finally replacing Charlie's old broken wand.

Hermione decides the best way to choose is by determining the minimum number of gold galleons needed to buy each *non-evil* wand of high power and age. Write a query to print the *id*, *age*, *coins_needed*, and *power* of the wands that Ron's interested in, sorted in order of descending *power*. If more than one wand has same power, sort the result in order of descending *age*.

Input Format

The following tables contain data on the wands in Ollivander's inventory:

- ***Wands:* The *id* is the id of the wand, *code* is the code of the wand, *coins_needed* is the total number of gold galleons needed to buy the wand, and *power* denotes the quality of the wand (the higher the power, the better the wand is).**

Column	Type
id	Integer
code	Integer
coins_needed	Integer
power	Integer

- ***Wands_Property:* The *code* is the code of the wand, *age* is the age of the wand, and *is_evil* denotes whether the wand is good for the dark arts. If the value of *is_evil* is *0*, it means that the wand is not evil. The mapping between *code* and *age* is one-one, meaning that if there are two pairs,(code1,age1) and (code2,age2), then code1 != code2 and age1 != age2.**

Column	Type
code	Integer
age	Integer
is_evil	Integer

Sample Input

Wands Table:-

id	code	coins_needed	power
1	4	3688	8
2	3	9365	3
3	3	7187	10
4	3	734	8
5	1	6020	2
6	2	6773	7
7	3	9873	9
8	3	7721	7
9	1	1647	10
10	4	504	5
11	2	7587	5
12	5	9897	10
13	3	4651	8
14	2	5408	1
15	2	6018	7
16	4	7710	5
17	2	8798	7
18	2	3312	3
19	4	7651	6
20	5	5689	3

Wands_Property **Table:**

code	age	is_evil
1	45	0
2	40	0
3	4	1
4	20	0
5	17	0

Sample Output
9 45 1647 10
12 17 9897 10
1 20 3688 8
15 40 6018 7
19 20 7651 6
11 40 7587 5
10 20 504 5
18 40 3312 3
20 17 5689 3
5 45 6020 2
14 40 5408 1
Explanation
The data for wands of *age 45* (code 1):

id	age	coins_needed	power
5	45	6020	2
9	45	1647	10

- **The minimum number of galleons needed for wand(age = 45, power = 2) = 6020**
- **The minimum number of galleons needed for wand(age = 45, power = 10) = 1647**

The data for wands of *age 40* (code 2):

id	age	coins_needed	power
14	40	5408	1
18	40	3312	3
11	40	7587	5
15	40	6018	7
17	40	8798	7
6	40	6773	7

- **The minimum number of galleons needed for wand(age = 40, power = 1) = 5408**
- **The minimum number of galleons needed for wand(age = 40, power = 3) = 3312**
- **The minimum number of galleons needed for wand(age = 40, power = 5) = 7587**
- **The minimum number of galleons needed for wand(age = 40, power = 7) = 6018**

The data for wands of *age 20* (code 4):

id	age	coins_needed	power
10	20	504	5
16	20	7710	5
19	20	7651	6
1	20	3688	8

- **The minimum number of galleons needed for wand(age = 20, power = 5) = 504**
- **The minimum number of galleons needed for wand(age = 20, power = 6) = 7651**
- **The minimum number of galleons needed for wand(age = 0, power = 8) = 3688**

The data for wands of *age 17* (code 5):

id	age	coins_needed	power
20	17	5689	3
12	17	9897	10

- **The minimum number of galleons needed for wand(age = 17, power = 3) = 5689**
- **The minimum number of galleons needed for wand(age = 17, power = 10) = 9897**

Q.56. SYMMETRIC PAIRS (Mode of Difficulty: Medium) (Doubt)
You are given a table, *Functions*, containing two columns: *X* and *Y*.

Column	Type
X	Integer
Y	Integer

Two pairs (X_1, Y_1) and (X_2, Y_2) are said to be *symmetric pairs* if $X_1 = Y_2$ and $X_2 = Y_1$.
Write a query to output all such *symmetric pairs* in ascending order by the value of *X*.
Sample Input

X	Y
20	20
20	20
20	21
23	22
22	23
21	20

Sample Output
20 20
20 21
22 23

Q.57. INTERVIEWS (Mode of Difficulty: Hard) (Doubt)

Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the *contest_id*, *hacker_id*, *name*, and the sums of *total_submissions*, *total_accepted_submissions*, *total_views*, and *total_unique_views* for each contest sorted by *contest_id*. Exclude the contest from the result if all four sums are .

Note: A specific contest can be used to screen candidates at more than one college, but each college only holds screening contest.

Input Format

The following tables hold interview data:

- ***Contests:*** **The *contest_id* is the id of the contest, *hacker_id* is the id of the hacker who created the contest, and *name* is the name of the hacker.**

Column	Type
contest_id	Integer
hacker_id	Integer
name	String

- ***Colleges:*** **The *college_id* is the id of the college, and *contest_id* is the id of the contest that Samantha used to screen the candidates.**

Column	Type
college_id	Integer
contest_id	Integer

- ***Challenges:*** **The *challenge_id* is the id of the challenge that belongs to one of the contests whose contest_id Samantha forgot, and *college_id* is the id of the college where the challenge was given to candidates.**

Column	Type
challenge_id	Integer
college_id	Integer

- ***View_Stats:*** **The *challenge_id* is the id of the challenge, *total_views* is the number of times the challenge was viewed by candidates, and *total_unique_views* is the number of times the challenge was viewed by unique candidates.**

Column	Type
challenge_id	Integer
total_views	Integer
total_unique_views	Integer

- ***Submission_Stats:*** **The *challenge_id* is the id of the challenge, *total_submissions* is the number of submissions for the challenge, and *total_accepted_submission* is the number of submissions that achieved full scores.**

Column	Type
challenge_id	Integer
total_submissions	Integer
total_accepted_submissions	Integer

Sample Input
***Contests* Table:**

contest_id	hacker_id	name
66406	17973	Rose
66556	79153	Angela
94828	80275	Frank

***Colleges* Table:**

college_id	contest_id
11219	66406
32473	66556
56685	94828

***Challenges* Table:**

challenge_id	college_id
18765	11219
47127	11219
60292	32473
72974	56685

***View_Stats* Table:**

challenge_id	total_views	total_unique_views
47127	26	19
47127	15	14
18765	43	10
18765	72	13
75516	35	17
60292	11	10
72974	41	15
75516	75	11

***Submission_Stats* Table:**

challenge_id	total_submissions	total_accepted_submissions
75516	34	12
47127	27	10
47127	56	18
75516	74	12
75516	83	8
72974	68	24
72974	82	14
47127	28	11

Sample Output
66406 17973 Rose 111 39 156 56
66556 79153 Angela 0 0 11 10
94828 80275 Frank 150 38 41 15
Explanation
The contest 66406 is used in the college 11219. In this college , 11219 challenges 18765 and 47127 are asked, so from the *view* and *submission* stats:

- **Sum of total submissions = 27 + 56 + 28 = 111**
- **Sum of total accepted submissions = 10 + 18 + 11 = 39**
- **Sum of total views = 43 + 72 + 26 + 15 =156**
- **Sum of total unique views = 10 + 13 + 19 + 14 = 56**

Similarly, we can find the sums for contests 66556 and 94828.

Q.58. 15 DAYS OF LEARNING SQL (Mode of Difficulty: Hard) (Doubt)

Julia conducted a 15 days of learning SQL contest. The start date of the contest was *March 01, 2016* and the end date was *March 15, 2016*.

Write a query to print total number of unique hackers who made at least 1 submission each day (starting on the first day of the contest), and find the *hacker_id* and *name* of the hacker who made maximum number of submissions each day. If more than one such hacker has a

maximum number of submissions, print the lowest *hacker_id*. The query should print this information for each day of the contest, sorted by the date.

Input Format

The following tables hold contest data:

- *Hackers:* The *hacker_id* is the id of the hacker, and *name* is the name of the hacker.

Column	Type
hacker_id	Integer
name	String

- *Submissions:* The *submission_date* is the date of the submission, *submission_id* is the id of the submission, *hacker_id* is the id of the hacker who made the submission, and *score* is the score of the submission.

Column	Type
submission_date	Date
submission_id	Integer
hacker_id	Integer
score	Integer

Sample Input

For the following sample input, assume that the end date of the contest was *March 06, 2016*.

Hackers Table:

hacker_id	name
15758	Rose
20703	Angela
36396	Frank
38289	Patrick
44065	Lisa
53473	Kimberly
62529	Bonnie
79722	Michael

Submissions Table:

submission_date	submission_id	hacker_id	score
2016-03-01	8494	20703	0
2016-03-01	22403	53473	15
2016-03-01	23965	79722	60
2016-03-01	30173	36396	70
2016-03-02	34928	20703	0
2016-03-02	38740	15758	60
2016-03-02	42769	79722	25
2016-03-02	44364	79722	60
2016-03-03	45440	20703	0
2016-03-03	49050	36396	70
2016-03-03	50273	79722	5
2016-03-04	50344	20703	0
2016-03-04	51360	44065	90
2016-03-04	54404	53473	65
2016-03-04	61533	79722	45
2016-03-05	72852	20703	0
2016-03-05	74546	38289	0
2016-03-05	76487	62529	0
2016-03-05	82439	36396	10
2016-03-05	90006	36396	40
2016-03-06	90404	20703	0

Sample Output

2016-03-01 4 20703 Angela
2016-03-02 2 79722 Michael
2016-03-03 2 20703 Angela
2016-03-04 2 20703 Angela
2016-03-05 1 36396 Frank
2016-03-06 1 20703 Angela

Explanation

On *March 01, 2016* hackers 20703,36396,53473,79722 and made submissions. There are 4 unique hackers who made at least one submission each day. As each hacker made one submission, 20703 is considered to be the hacker who made maximum number of submissions on this day. The name of the hacker is *Angela*.

On *March 02, 2016* hackers 15758,20703 and 79722 made submissions. Now 20703 and 79722 were the only ones to submit every day, so there are 2 unique hackers who made at least one submission each day. 79722 made 2 submissions, and name of the hacker is *Michael*.

On *March 03, 2016* hackers 20703,36396 and 79722 made submissions. Now 20703 and 79722 were the only ones, so there are 2 unique hackers who made at least one submission each day. As each hacker made one submission so 20703 is considered to be the hacker who made maximum number of submissions on this day. The name of the hacker is *Angela*.

On *March 04, 2016* hackers 20703, 44065 , 53473 and 79722 made submissions. Now 20703 and 79722 only submitted each day, so there are 2 unique hackers who made at least one submission each day. As each hacker made one submission so 20703 is

considered to be the hacker who made maximum number of submissions on this day. The name of the hacker is *Angela*.

On *March 05, 2016* hackers 20703 , 36396 , 38289 and 62529 made submissions.

Now 20703 only submitted each day, so there is only 1 unique hacker who made at least one submission each day. 36396 made 2 submissions and name of the hacker is *Frank*.

On *March 06, 2016* only 20703 made submission, so there is only 1 unique hacker who made at least one submission each day. 20703 made 1 submission and name of the hacker is *Angela*.

ALL THE BEST