Basic Select

1. SQL Project Planning

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.

Link of question Markdown Live Preview.

```
Query : SELECT s.Proj_Start_Date, min(e.Proj_End_Date) as Real_Proj_End_Date
FROM
(SELECT Start_Date as Proj_Start_Date FROM Projects WHERE Start_Date NOT IN
(SELECT End_Date FROM Projects)) s,
(SELECT End_Date as Proj_End_Date FROM Projects WHERE End_Date NOT IN (SELECT
Start_Date FROM Projects)) e
WHERE s.Proj_Start_Date < e.Proj_End_Date
GROUP BY s.Proj_Start_Date
ORDER BY DATEDIFF(min(e.Proj_End_Date), s.Proj_Start_Date) ASC, s.Proj_Start_Date
ASC;
```

2. Placements

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	Name
1	Ashley
2	Samantha
3	Julia
4	Scarlet

Students

ID	Friend_ID
1	2
2	3
3	4
4	1

Friends

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

Link of question Markdown Live Preview.

```
Query : SELECT S.NAME
FROM STUDENTS S,
     FRIENDS F,
     PACKAGES P1,
     PACKAGES P2
WHERE S.ID = F.ID
  AND F.FRIEND_ID = P2.ID
  AND S.ID = P1.ID
  AND P1.SALARY < P2.SALARY
ORDER BY P2.SALARY;
```

3. Symmetric Pairs

You are given a table, Functions, containing two columns: X and Y.

Column	*Type*
X	*Integer*
Y	*Integer*

Two pairs (X1, Y1) and (X2, Y2) are said to be symmetric pairs if X1 = Y2 and X2 = Y1.

Write a query to output all such symmetric pairs in ascending order by the value of X. List the rows such that X1 ≤ Y1.

Sample Input

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

Sample Output

20 20 20 21 22 23

Link of question Markdown Live Preview.

```
Query : SELECT X,
       Y
FROM FUNCTIONS F1
WHERE EXISTS
    (SELECT *
     FROM FUNCTIONS F2
     WHERE F2.Y = F1.X
       AND F2.X = F1.Y
       AND F2.X > F1.X)
  AND (X != Y)
UNION
SELECT X,
       Y
FROM FUNCTIONS F1
WHERE X = Y
  AND (
         (SELECT COUNT(*)
          FROM FUNCTIONS
          WHERE X = F1.X
            AND Y = F1.X) > 1)
ORDER BY X;
```

4. Interviews

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 0.

Note: A specific contest can be used to screen candidates at more than one college, but each college only holds 1 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

- 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

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

- 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.

Link of question Markdown Live Preview.

```
Query : SELECT CON.CONTEST_ID,
       CON.HACKER_ID,
       CON.NAME,
       SUM(TOTAL_SUBMISSIONS),
       SUM(TOTAL_ACCEPTED_SUBMISSIONS),
       SUM(TOTAL_VIEWS),
       SUM(TOTAL_UNIQUE_VIEWS)
FROM CONTESTS CON
JOIN COLLEGES COL ON CON.CONTEST_ID = COL.CONTEST_ID
JOIN CHALLENGES CHA ON COL.COLLEGE_ID = CHA.COLLEGE_ID
LEFT JOIN
  (SELECT CHALLENGE_ID,
          SUM(TOTAL_VIEWS) AS TOTAL_VIEWS,
          SUM(TOTAL_UNIQUE_VIEWS) AS TOTAL_UNIQUE_VIEWS
   FROM VIEW_STATS
   GROUP BY CHALLENGE_ID) VS ON CHA.CHALLENGE_ID = VS.CHALLENGE_ID
LEFT JOIN
  (SELECT CHALLENGE_ID,
          SUM(TOTAL_SUBMISSIONS) AS TOTAL_SUBMISSIONS,
          SUM(TOTAL_ACCEPTED_SUBMISSIONS) AS TOTAL_ACCEPTED_SUBMISSIONS
   FROM SUBMISSION_STATS
   GROUP BY CHALLENGE_ID) SS ON CHA.CHALLENGE_ID = SS.CHALLENGE_ID
GROUP BY CON.CONTEST_ID,
         CON.HACKER_ID,
         CON.NAME
HAVING SUM(TOTAL_SUBMISSIONS) != 0
OR SUM(TOTAL_ACCEPTED_SUBMISSIONS) != 0
OR SUM(TOTAL_VIEWS) != 0
OR SUM(TOTAL_UNIQUE_VIEWS) != 0
ORDER BY CONTEST_ID;
```

5. 15 Days of learning SQL

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.

![This is a Table image.](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. "This is a sample image.")

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, 54373, and 79722 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, 36396, 38289, and 62529 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.

Link of question Markdown Live Preview.

```
Query : WITH R(SUBMISSION_DATE, HACKER_ID) AS
  (SELECT DISTINCT SUBMISSION_DATE,
                   HACKER_ID
   FROM SUBMISSIONS
   WHERE SUBMISSION_DATE = TO_DATE('2016-03-01')
   UNION ALL SELECT CHILD.SUBMISSION_DATE,
                    CHILD.HACKER_ID
   FROM R PARENT,
        SUBMISSIONS CHILD
   WHERE PARENT.SUBMISSION_DATE + 1 = CHILD.SUBMISSION_DATE
     AND PARENT.HACKER_ID = CHILD.HACKER_ID),
     TOTAL AS
  (SELECT SUBMISSION_DATE,
          COUNT(DISTINCT HACKER_ID) AS TOTAL
   FROM R
   GROUP BY SUBMISSION_DATE),
     COUNTER AS
  (SELECT SUBMISSION_DATE,
          HACKER_ID,
          COUNT(HACKER_ID) AS N
   FROM SUBMISSIONS
   GROUP BY SUBMISSION_DATE,
            HACKER_ID),
     MAXPERDAY AS
  (SELECT C.SUBMISSION_DATE,
          MIN(C.HACKER_ID) AS HACKER_ID
   FROM COUNTER C
   WHERE C.N =
       (SELECT MAX(K.N)
        FROM COUNTER K
        WHERE C.SUBMISSION_DATE = K.SUBMISSION_DATE)
```

```
   GROUP BY SUBMISSION_DATE)
SELECT TOTAL.SUBMISSION_DATE,
       TOTAL.TOTAL,
       HACKERS.HACKER_ID,
       HACKERS.NAME
FROM TOTAL
JOIN MAXPERDAY ON TOTAL.SUBMISSION_DATE = MAXPERDAY.SUBMISSION_DATE
JOIN HACKERS ON MAXPERDAY.HACKER_ID = HACKERS.HACKER_ID
ORDER BY TOTAL.SUBMISSION_DATE;
```