SQL praticse and answer

Jane Liu

July 23,2019

603. Consecutive Available Seats

NOTE: Several friends at a cinema ticket office would like to reserve consecutive available seats. Can you help to query all the consecutive available seats order by the seat_id using the following cinematable?

The seat_id is an auto increment int, and free is bool ('1' means free, and '0' means occupied.). Consecutive available seats are more than 2(inclusive) seats consecutively available.

Answer: CREATE TABLE cinema (seat_id INT, free INT);

INSERT INTO cinema VALUES (1,1); INSERT INTO cinema VALUES (2,0); INSERT INTO cinema VALUES (3,1); INSERT INTO cinema VALUES (4,1); INSERT INTO cinema VALUES (5,1);

SELECT DISTINCT c1.seat_id FROM cinema c1 JOIN cinema c2 ON ABS(c1.seat_id- c2.seat_id)=1 WHERE c1.free = 1 AND c2.free =1 ORDER BY c1.seat_id

[explain]https://leetcode.com/articles/consecutive-available-seats/

idea: join two table and choose the consecutive id and here ABS is help to get the 34 and 45 and keep mind the double 3 4 4 5, so we need add distinct

626. Exchange Seats

NOTE: Mary is a teacher in a middle school and she has a table seat storing students' names and their corresponding seat ids.

The column id is continuous increment.

Mary wants to change seats for the adjacent students.

Can you write a SQL query to output the result for Mary?

ANSWER:

CREATE TABLE seat (id INT, student VARCHAR(50));

INSERT INTO seat VALUES (1, 'Abbot'), (2, 'Doris'), (3, 'Emerson'), (4, 'Green'), (5, 'Jeames');

SELECT (CASE WHEN MOD(id, 2) != 0 AND counts != id THEN id + 1 WHEN MOD(id, 2) != 0 AND counts = id THEN id ELSE id - 1 END) AS id, student FROM seat, (SELECT COUNT(*) AS counts FROM seat) AS seat_counts ORDER BY id ASC;* idea:* Subgroup the student into three groups: the odd id and the last one keep original, the odd id +1 and even id -1, here MOD(id, 2) could be replace with id%2 = 0 or 1.

569. Median Employee Salary

Note The Employee table holds all employees. The employee table has three columns: Employee Id, Company Name, and Salary.

CREATE TABLE Employee (Id INT, Company VARCHAR(50), Salary INT);

INSERT INTO Employee VALUES (1, 'A', 2341), (2, 'A', 341), (3, 'A', 15), (4, 'A', 15314), (5, 'A', 451), (6, 'A', 513), (7, 'B', 15), (8, 'B', 13), (9, 'B', 1154), (10, 'B', 1314), (11, 'B', 1221), (12, 'B', 234), (13, 'C', 2345), (14, 'C', 2645), (15, 'C', 2645), (16,'C', 2652), (17, 'C', 65);

SELECT Employee.Id, Employee.Company, Employee.Salary FROM Employee, Employee alias WHERE Employee.Company = alias.Company GROUP BY Employee.Company , Employee.Salary HAVING SUM(CASE WHEN Employee.Salary = alias.Salary THEN 1 ELSE 0 END) >= ABS(SUM(SIGN(Employee.Salary - alias.Salary))) ORDER BY Employee.Id ;

*explain*https://leetcode.com/articles/median-employee-salary/

message

CREATE TABLE table1 (date DATE, u1 INT, u2 INT, n_msg INT); DELETE FROM table1; INSERT INTO table1 VALUES ('19-07-01', 1, 2, 3), ('19-07-01', 1, 3, 3), ('19-07-02', 4, 2, 5), ('19-07-02', 1, 6, 3), ('19-07-03', 2, 5, 3), ('19-07-03', 6, 2, 5), ('19-07-04', 1, 2, 3), ('19-07-05', 4, 2, 3);

SELECT cn, COUNT(cn) ccn FROM(SELECT date, u1, COUNT(DISTINCT u2) cn FROM table1 GROUP BY date, u1) a GROUP BY cn

sum(n_msg_with_top_partners) / sum(n_msg_with_all_contacts):

SELECT msm/(SELECT SUM(n_msg) FROM table1) FROM(

SELECT u1, MAX(sm) msm FROM(SELECT u1,u2, SUM(n_msg) sm FROM table1 GROUP BY u1, u2) a GROUP BY u1) b

Friends

CREATE TABLE Friends (date DATE, action_id INT, target_id INT, action_type VARCHAR(50)); DELETE FROM Friends; INSERT INTO Friends VALUES ('19-07-01', 1, 2, 'accept'), ('19-07-01', 2, 1, 'request'), ('19-07-02', 4, 2, 'unfriend'), ('19-07-02', 1, 6, 'accept'), ('19-07-03', 7, 1, 'request'), ('19-07-03', 6, 1, 'request'), ('19-07-04', 1, 5, 'accept'), ('19-07-05', 5, 1, 'request');

SELECT sa/ sr FROM

(SELECT SUM(a.accept) sa FROM (SELECT a.action_id, COUNT(a.target_id) accept FROM Friends a, Friends b WHERE a.target_id = b.action_id AND a.action_id = b. target_id AND a.action_type ='accept' AND b.action_type ='request' GROUP BY a.action_id)a) aa

, (SELECT SUM(request) sr FROM (SELECT action_id, COUNT(target_id) request FROM Friends WHERE action_type = 'request' GROUP BY action_id)b) bb

The person who has the most friends SELECT user_id, SUM(cnt) AS n_friend FROM ((SELECT action_id AS user_id, SUM(CASE WHEN action_type = 'accept' THEN 1 WHEN action_type = 'unfriend' THEN -1 ELSE 0 END) AS cnt FROM Friends GROUP BY action_id) UNION ALL (SELECT target_id AS user_id, SUM(CASE WHEN action_type = 'accept' THEN 1 WHEN action_type = 'unfriend' THEN -1 ELSE 0 END) AS cnt FROM Friends GROUP BY target_id)) a GROUP BY user_id ORDER BY n_friend DESC LIMIT 1

PostgreSQL

CREATE TABLE Friends (date DATE, action_id INT, target_id INT, action_type VARCHAR(50)); INSERT INTO Friends VALUES ('2019-07-01', 1, 2, 'accept'), ('2019-07-01', 2, 1, 'request'), ('2019-07-02', 4, 2, 'unfriend'), ('2019-07-02', 1, 6, 'accept'), ('2019-07-03', 7, 1, 'request'), ('2019-07-03', 6, 1, 'request'), ('2019-07-04', 1, 5, 'accept'), ('2019-07-05', 5, 1, 'request');

SELECT ROUND(SUM(accept)/ SUM(request), 2) AS Accept_Rate FROM (SELECT a.action_id, COUNT(a.target_id) AS accept FROM Friends a, Friends b WHERE a.target_id = b.action_id AND a.action_id = b. target_id AND a.action_type ='accept' AND b.action_type ='request' GROUP BY a.action_id) AS temp_1

FULL JOIN

(SELECT action_id, COUNT(target_id) AS request FROM Friends WHERE action_type = ‘request’ GROUP BY action_id) AS temp_2

USING(action_id)

CTR

CREATE TABLE table2 (time DATE, user_id INT, app INT, type VARCHAR(50)); DELETE FROM table2; INSERT INTO table2 VALUES (‘19-07-01’, 1, 2, ‘impression’), (‘19-07-01’, 2, 1, ‘impression’), (‘19-07-02’, 4, 2, ‘impression’), (‘19-07-02’, 1, 6, ‘click’), (‘19-07-03’, 7, 1, ‘click’), (‘19-07-03’, 6, 1, ‘click’), (‘19-07-04’, 1, 5, NULL), (‘19-07-05’, 5, 1, NULL);

SELECT SUM(CASE WHEN type = ‘click’ THEN 1 ELSE 0 END)/ SUM(CASE WHEN type = ‘click’ OR type = ‘impression’ THEN 1 ELSE 0 END) AS CTR FROM table2

CREATE TABLE table2 (time DATE, user_id INT, app INT, type VARCHAR(50)); DELETE FROM table2; INSERT INTO table2 VALUES (‘19-07-01’, 1, 2, ‘impression’), (‘19-07-01’, 2, 1, ‘impression’), (‘19-07-02’, 4, 2, ‘impression’), (‘19-07-02’, 1, 1, ‘click’), (‘19-07-03’, 7, 1, ‘click’), (‘19-07-03’, 6, 1, ‘click’), (‘19-07-04’, 1, 2, NULL), (‘19-07-05’, 5, 1, NULL);

SELECT cn/tn

FROM (SELECT app, COUNT (DISTINCT user_id) cn FROM table2 WHERE type = ‘click’ GROUP BY app) c, (SELECT app, COUNT (DISTINCT user_id) tn FROM table2 WHERE type = ‘click’ OR type = ‘impression’ GROUP BY app) t WHERE c.app = t.app

State

CREATE TABLE Tracking (

userid INT,

state VARCHAR(50)); INSERT INTO Tracking VALUES (1, ‘churned’), (2, ‘revived’), (3, ‘new’), (4, NULL);

CREATE TABLE Days (

userid INT); INSERT INTO Days VALUES (2), (3), (4);

SELECT t.*, d.*, CASE WHEN state IS NULL AND d.userid IS NOT NULL THEN ‘new’ WHEN state= ‘churned’ AND d.userid IS NULL THEN ‘churn’ WHEN state= ‘churned’ AND d.userid IS NOT NULL THEN ‘revived’ WHEN state IN (‘revived’, ‘new’) AND d.userid IS NOT NULL THEN ‘stayed’

```
END AS state_new
FROM Tracking t
FULL JOIN Days d
ON t.userid = d.userid
ORDER BY t.userid, d.userid
```

renew notification

CREATE TABLE table1 (

userid INT,

action_type VARCHAR(50)); INSERT INTO table1 VALUES (1, ‘on’), (2, ‘off’), (3, ‘on’), (4, ‘off’);

CREATE TABLE table2 (

userid INT, action_type VARCHAR(50)); INSERT INTO table2 VALUES (2, ‘on’), (3, ‘off’), (4, ‘on’);

SELECT CASE WHEN t2.userid IS NULL THEN t1.userid ELSE t2.userid END AS user, CASE WHEN t2.action_type != t1.action_type THEN t2.action_type ELSE t1.action_type END AS type FROM table1 t1 FULL JOIN table2 t2 ON t1.userid= t2.userid

***Recoomendation

CREATE TABLE table1 (

user_id INT,

friend_id INT); INSERT INTO table1 VALUES (1, 5), (2, 6), (3, 7), (4, 8);

CREATE TABLE table2 (

user_id INT, post_id INT); INSERT INTO table2 VALUES (1, 15), (2, 14), (3, 25), (4, 32), (5, 15), (6, 35), (7, 45), (8, 55) ;

SELECT t1.user_id , post_id FROM table1 t1 JOIN table2 t2 ON t1.friend_id = t2.user_id WHERE(t1.user_id, post_id) NOT IN (SELECT user_id, post_id FROM table2)

Phone logins

CREATE TABLE table1 (

dates date, country VARCHAR(50), carrier VARCHAR(50), phone_no INT, type VARCHAR(50));

INSERT INTO table1 VALUES (‘2018/1/1’, ‘US’,‘mobile’, 1000, ‘login_confirmation’), (‘2018/1/1’, ‘US’,‘desktop’, 100, ‘notification’), (‘2018/1/1’, ‘US’,‘mobile’, 1000, ‘login_confirmation’), (‘2019/07/24’, ‘UK’,‘mobile’, 100, ‘login_confirmation’), (‘2018/1/1’, ‘CA’,‘desktop’, 1000, ‘login_confirmation’), (‘2018/1/2’, ‘UK’,‘desktop’, 100, ‘login_confirmation’), (‘2018/1/3’, ‘US’,‘mobile’, 10000, ‘login_confirmation’);

CREATE TABLE table2 (

dates date,

phone_no INT);

INSERT INTO table2 VALUES (‘2018/1/1’, 100), (‘2018/1/2’, 1000), (‘2018/1/3’, 10000), (‘2018/1/4’, 100000);

SELECT country, carrier, COUNT(phone_no) FROM table1 WHERE type = ‘login_confirmation’ AND dates = curdate()-1 GROUP BY country, carrier;

CREATE TABLE table1 (

dates date, country VARCHAR(50), carrier VARCHAR(50), phone_no INT, type VARCHAR(50));

INSERT INTO table1 VALUES (‘2018/1/1’, ‘US’,‘mobile’, 1000, ‘login_confirmation’), (‘2018/1/1’, ‘US’,‘desktop’, 1002, ‘login_confirmation’), (‘2018/1/1’, ‘US’,‘mobile’, 10001, ‘login_confirmation’), (‘2019/07/24’, ‘UK’,‘mobile’, 100, ‘login_confirmation’), (‘2018/1/1’, ‘CA’,‘desktop’, 1000, ‘login_confirmation’), (‘2018/1/2’, ‘UK’,‘desktop’, 1003, ‘login_confirmation’), (‘2018/1/3’, ‘US’,‘mobile’, 100004, ‘login_confirmation’);

CREATE TABLE table2 (

dates date,

phone_no INT);

INSERT INTO table2 VALUES (‘2018/1/1’, 1000), (‘2018/1/2’, 1002), (‘2018/1/3’, 10000), (‘2018/1/4’, 100000);

SELECT x.dates, x.country, x.carrier, ROUND(y.ln/x.tn, 2) AS lr FROM

(SELECT dates, country, carrier, COUNT(DISTINCT phone_no) AS tn FROM table1 WHERE type = 'login_confirmation' GROUP BY dates, country, carrier) x JOIN (SELECT a.dates, a.country, a.carrier, COUNT(DISTINCT a.phone_no) AS ln FROM table1 a JOIN table2 b ON a.dates = b.dates AND a.phone_no = b.phone_no WHERE a.type = 'login_confirmation' GROUP BY a.dates, a.country, a.carrier) y ON x.dates = y.dates AND x.country = y.country AND x.carrier= y.carrier

article

CREATE TABLE article_views (date timestamp, viewer_id int, article_id int, author_id int);

INSERT INTO article_views VALUES ('2017-08-01',123, 456 ,789); INSERT INTO article_views VALUES ('2017-08-02',432 ,543, 654); INSERT INTO article_views VALUES ('2017-08-01',789, 456 ,789); INSERT INTO article_views VALUES ('2017-08-03',567, 780, 432); INSERT INTO article_views VALUES ('2017-08-01',789, 457 ,789);

How many article authors have never viewed their own article?

SELECT COUNT(DISTINCT author_id) FROM article_views

WHERE author_id NOT IN (SELECT author_id FROM article_views WHERE viewer_id = author_id)

Below is wrong for the author view the article in another row hasn't been exclusived

SELECT COUNT(a1.author_id) FROM article_views a1 JOIN article_views a2 ON a1.article_id =a2.article_id AND a1.date = a2.date WHERE a1.viewer_id != a2.author_id

How many members viewed more than one articles on 2017-08-01

SELECT viewer_id FROM article_views WHERE date ='2017-08-01' GROUP BY viewer_id HAVING COUNT(DISTINCT article_id)>1

Companies

CREATE TABLE companies (member_id int, company_name varchar(80), year_start int);

INSERT INTO companies VALUES (1,'Google', 1990); INSERT INTO companies VALUES (1,'Microsoft', 2000); INSERT INTO companies VALUES (2,'Microsoft', 2000); INSERT INTO companies VALUES (2,'Google', 2001); INSERT INTO companies VALUES (3,'Microsoft', 1997); INSERT INTO companies VALUES (3,'Google', 1998); INSERT INTO companies VALUES (4,'Microsoft', 1997); INSERT INTO companies VALUES (4,'LinkedIn', 1998); INSERT INTO companies VALUES (4,'Google', 2000);

*count members who ever moved from Microsoft to Google

SELECT COUNT(DISTINCT c1.member_id) FROM companies c1 JOIN companies c2 ON c1.member_id = c2.member_id WHERE c1.company_name = 'Microsoft' AND c2.company_name= 'Google' AND c1.year_start < c2. year_start

***count members who directly moved from Microsoft to Google? (Microsoft – Linkedin – Google doesn't count)**

SELECT COUNT(DISTINCT member_id)FROM(

```
SELECT member_id, company_name AS c_name,
LAG(company_name) OVER(PARTITION BY member_id ORDER BY year_start) AS p_name
FROM companies )  t
WHERE t.c_name = 'Google' AND t.p_name = 'Microsoft'
```

continent

***find the country with largest population in each continent, with strictly output: continent, country, population.**

SELECT a.continent, a.country, c.population FROM (SELECT continent, country, DENSE_RANK() OVER(PARTITION BY continent ORDER BY population DESC) AS rank FROM continents)a JOIN continents c ON c.country = a.country WHERE rank =1

Becasue window fucntion could not be used in where so we need a outside query, and here dense_rank give the value which are the same

now for each continent, find the country with largest % of population in given continent. write SQL, then write Python

SELECT c.continent, c.country, ROUND(c.population/d.sp*100) FROM continents c JOIN (SELECT continent,country, DENSE_RANK() OVER(PARTITION BY continent ORDER BY population DESC) AS rank FROM continents) a ON a.country = c.country JOIN (SELECT continent, CAST(SUM(population) AS FLOAT) AS sp FROM continents GROUP BY continent) d

ON d.continent = a.continent

WHERE rank = 1

Because here in PostgreSQL ROUND() could not add 2 in it

student attendence

CREATE TABLE attendance (date timestamp, student_id int, attendance int – 0, 1);

INSERT INTO attendance VALUES ('2018/01/02', 2,1); INSERT INTO attendance VALUES ('2018/01/02', 1,1); INSERT INTO attendance VALUES ('2018/01/02', 4,1); INSERT INTO attendance VALUES ('2018/02/02', 5,1); INSERT INTO attendance VALUES ('2018/02/22', 1,1); INSERT INTO attendance VALUES ('2018/02/22', 4,1); INSERT INTO attendance VALUES ('2018/02/22', 5,1);

CREATE TABLE student (student_id int, school_id int, grade_level int, date_of_birth timestamp, hometown varchar(80)
);

What was the overall attendance rate for the school district yesterday?

SELECT COUNT(DISTINCT a.student_id)/ COUNT(DISTINCT s.student_id) FROM attendance a JOIN student s ON a.student_id = s.student_id WHERE date = '2018/01/02'

Which grade level currently has the most students in this school district?

```
SELECT grade_level, COUNT(s.student_id) FROM student s GROUP BY grade_level ORDER BY
COUNT(s.student_id) DESC LIMIT 1
```

Which school had the average highest attendance rate? the lowest?

```
SELECT school_id, AVG(ar) FROM (

SELECT date, t1.school_id, ROUND(an/tn,2) AS ar FROM

(SELECT date, school_id, COUNT(a.student_id) As an FROM attendance a JOIN student s ON a.student_id
= s.student_id GROUP BY date, school_id) t1 JOIN (SELECT school_id, COUNT(student_id) AS tn
FROM student GROUP BY school_id) t2 ON t1.school_id= t2.school_id) t3 GROUP BY school_id
ORDER BY AVG(ar) DESC LIMIT 1
```