**----Combine Two Tables--**

**CREATE** **TABLE** Person (

PersonId serial **PRIMARY** **KEY**,

FirstName **varchar**(255),

LastName **varchar**(255)

);

-- Create the "Address" table

**CREATE** **TABLE** Address (

AddressId serial **PRIMARY** **KEY**,

PersonId **int** **REFERENCES** Person(PersonId),

City **varchar**(255),

State **varchar**(255)

);

**SELECT**

p.firstname,

p.lastname,

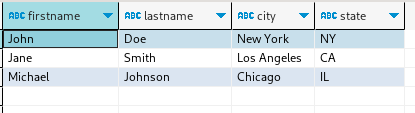
a.city,

a.state

**FROM**

person p

**JOIN** address a **on** p.personid = a.personid;



**----176. Second Highest Salary--**

-- Create the "Employee" table

**CREATE** **TABLE** Employee (

Id **int** **PRIMARY** **KEY**,

Salary **int**

);

-- Insert data into the "Employee" table

**INSERT** **INTO** Employee (Id, Salary)

**VALUES**

(1, 100),

(2, 200),

(3, 300);

**SELECT**

**MAX**(salary) **AS** SecondHighestSalary

**FROM**

employee

**WHERE**

salary < (

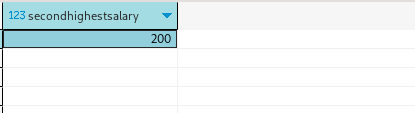
**SELECT**

**MAX**(salary)

**FROM**

employee

);



**-----Nth highest salary------**

**CREATE** **OR** **REPLACE** **FUNCTION** get\_nth\_highest\_salary(n **INT**) **RETURNS** **INT** **AS** **$$**

**DECLARE**

nth\_salary **INT**;

**BEGIN**

n := n - 1;

**SELECT** **DISTINCT** Salary

**INTO** nth\_salary

**FROM** (

**SELECT** Salary, **RANK**() **OVER** (**ORDER** **BY** Salary **DESC**) **AS** **rank**

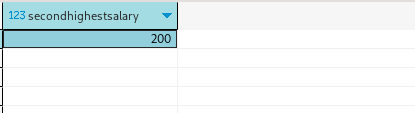
**FROM** Employee

) ranked\_salaries

**WHERE** **rank** = n + 1;

**RETURN** nth\_salary;

**END**;



**---------178. Rank Scores**

-- Create the "Scores" table

**CREATE** **TABLE** Scores (

Id **int** **PRIMARY** **KEY**,

Score **numeric**(4, 2)

);

-- Insert data into the "Scores" table

**INSERT** **INTO** Scores (Id, Score)

**VALUES**

(1, 3.50),

(2, 3.65),

(3, 4.00),

(4, 3.85),

(5, 4.00),

(6, 3.65);

**SELECT**

score,

**DENSE\_RANK**() **OVER**(

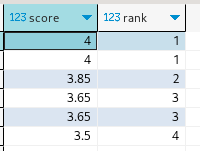
**ORDER** **BY**

score **DESC**

) **AS** **rank**

**FROM**

Scores;



**--------180. Consecutive Numbers**

-- Create the "Numbers" table

**CREATE** **TABLE** Numbers (

Id **int** **PRIMARY** **KEY**,

Num **int**

);

-- Insert data into the "Numbers" table

**INSERT** **INTO** Numbers (Id, Num)

**VALUES**

(1, 1),

(2, 1),

(3, 1),

(4, 2),

(5, 1),

(6, 2),

(7, 2);

**select**

**distinct** l1.num **as** ConsecutiveNums

**from**

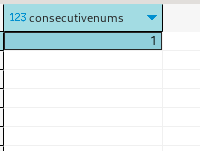
numbers l1

**join** numbers l2 **on** l2.id = l1.id + 1

**and** l2.num = l1.num

**join** numbers l3 **on** l3.id = l1.id + 2

**and** l3.num = l1.num



**----------181. Employees Earning More Than Their Managers**

-- Create the "Employees" table

**CREATE** **TABLE** Employees (

Id **int** **PRIMARY** **KEY**,

**Name** **varchar**(255),

Salary **numeric**(10, 2),

ManagerId **int**

);

-- Insert data into the "Employees" table

**INSERT** **INTO** Employees (Id, **Name**, Salary, ManagerId)

**VALUES**

(1, 'Joe', 70000.00, 3),

(2, 'Henry', 80000.00, 4),

(3, 'Sam', 60000.00, **NULL**),

(4, 'Max', 90000.00, **NULL**);

**select**

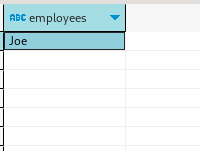
e1.**name** **as** Employees

**from**

Employees e1

**join** Employees e2 **on** e1.managerId = e2.id

**and** e1.salary > e2.salary;



**CREATE** **TABLE** Persons (

Id **int** **PRIMARY** **KEY**,

Email **varchar**(255)

);

-- Insert data into the "Person" table

**INSERT** **INTO** Persons (Id, Email)

**VALUES**

(1, 'a@b.com'),

(2, 'c@d.com'),

(3, 'a@b.com');

**select**

email **as** Email

**from**

(

**select**

email,

**count**(\*) **as** c

**from**

Persons

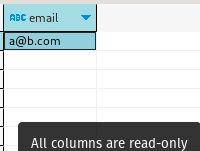
**group** **by**

email

) **temp**

**where**

**temp**.c > 1;



**--------------183. Customers Who Never Order**

-- Create the "Customers" table

**CREATE** **TABLE** Customers (

Id **int** **PRIMARY** **KEY**,

**Name** **varchar**(255)

);

-- Insert data into the "Customers" table

**INSERT** **INTO** Customers (Id, **Name**)

**VALUES**

(1, 'Joe'),

(2, 'Henry'),

(3, 'Sam'),

(4, 'Max');

-- Create the "Orders" table

**CREATE** **TABLE** Orders (

Id **int** **PRIMARY** **KEY**,

CustomerId **int** **REFERENCES** Customers(Id)

);

-- Insert data into the "Orders" table

**INSERT** **INTO** Orders (Id, CustomerId)

**VALUES**

(1, 3),

(2, 1);

**SELECT**

**name** **AS** Customers

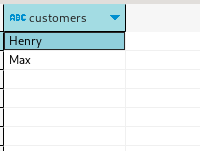
**FROM**

customers c

**LEFT** **JOIN** orders o **ON** c.id = o.customerid

**WHERE**

o.id **IS** **NULL**;



**-------------196. Delete Duplicate Emails**

-- Create the "Person" table

**CREATE** **TABLE** Persons (

Id **int** **PRIMARY** **KEY**,

Email **varchar**(255)

);

-- Insert data into the "Person" table

**INSERT** **INTO** Persons (Id, Email)

**VALUES**

(1, 'john@example.com'),

(2, 'bob@example.com'),

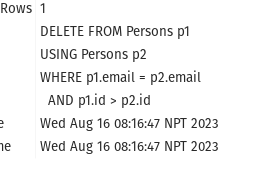
(3, 'john@example.com');

**DELETE** **FROM** Persons p1

**USING** Persons p2

**WHERE** p1.email = p2.email

**AND** p1.id > p2.id;



**-------------197. Rising Temperature**

**CREATE** **TABLE** Weather (

Id **int** **PRIMARY** **KEY**,

RecordDate **date**,

Temperature **int**

);

**INSERT** **INTO** Weather (Id, RecordDate, Temperature)

**VALUES**

(1, '2015-01-01', 10),

(2, '2015-01-02', 25),

(3, '2015-01-03', 20),

(4, '2015-01-04', 30);

**SELECT**

w1.id **AS** Id

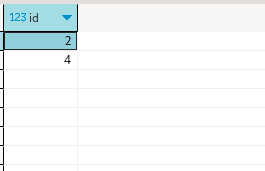
**FROM**

Weather w1

**JOIN**

Weather w2 **ON** w1.recordDate = w2.recordDate + **INTERVAL** '1 DAY'

**AND** w1.temperature > w2.temperature;



**--------------511. Game Play Analysis I**

**CREATE** **TABLE** Activity (

player\_id **int**,

device\_id **int**,

event\_date **date**,

games\_played **int**,

**PRIMARY** **KEY** (player\_id, event\_date)

);

-- Insert data into the "Activity" table

**INSERT** **INTO** Activity (player\_id, device\_id, event\_date, games\_played)

**VALUES**

(1, 2, '2016-03-01', 5),

(1, 2, '2016-05-02', 6),

(2, 3, '2017-06-25', 1),

(3, 1, '2016-03-02', 0),

(3, 4, '2018-07-03', 5);

**with** cte **as** (

**select**

player\_id,

event\_date,

**count**(\*) **over**(

**partition** **by** player\_id

**order** **by**

event\_date

) c

**from**

Activity

)

**select**

player\_id,

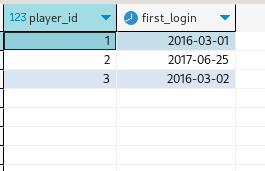
event\_date first\_login

**from**

cte

**where**

cte.c = 1;



**--------------512. Game Play Analysis II**

**with** cte **as** (

**select**

\*,

**dense\_rank**() **over**(

**partition** **by** player\_id

**order** **by**

a.event\_date

) **rank**

**from**

activity a

)

**select**

player\_id,

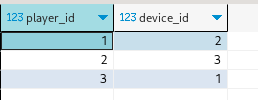
device\_id

**from**

cte

**where**

cte.**rank** = 1;



**--------------534. Game Play Analysis III**

**select**

player\_id,

event\_date,

**sum**(games\_played) **over**(

**partition** **by** player\_id

**order** **by**

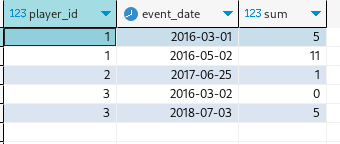
player\_id **rows** **between** **unbounded** **preceding**

**and** **current** **row**

)

**from**

activity a;



**--------------570. Managers with at Least 5 Direct Reports**

-- Create the "Employee" table

**CREATE** **TABLE** Employee (

Id **int** **PRIMARY** **KEY**,

**Name** **varchar**(255),

Department **varchar**(255),

ManagerId **int**

);

-- Insert data into the "Employee" table

**INSERT** **INTO** Employee (Id, **Name**, Department, ManagerId)

**VALUES**

(101, 'John', 'A', **NULL**),

(102, 'Dan', 'A', 101),

(103, 'James', 'A', 101),

(104, 'Amy', 'A', 101),

(105, 'Anne', 'A', 101),

(106, 'Ron', 'B', 101);

**with** cte **as** (

**select**

managerId,

**count**(managerId) c

**from**

employee

**group** **by**

managerId

)

**select**

**name**

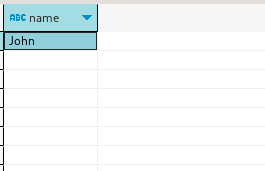
**from**

employee e

**join** cte **on** e.id = cte.managerId

**where**

cte.c >= 5;



**----------------577. Employee Bonus**

-- Create the "Employee" table

**CREATE** **TABLE** Employee (

empId **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

supervisor **int**,

salary **int**

);

-- Insert data into the "Employee" table

**INSERT** **INTO** Employee (empId, **name**, supervisor, salary)

**VALUES**

(1, 'John', 3, 1000),

(2, 'Dan', 3, 2000),

(3, 'Brad', **NULL**, 4000),

(4, 'Thomas', 3, 4000);

-- Create the "Bonus" table

**CREATE** **TABLE** Bonus (

empId **int** **PRIMARY** **KEY**,

bonus **int**

);

-- Insert data into the "Bonus" table

**INSERT** **INTO** Bonus (empId, bonus)

**VALUES**

(2, 500),

(4, 2000);

**select**

**name**,

bonus

**from**

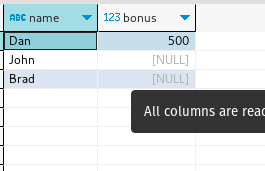
employee e

**left** **join** bonus b **on** e.empid = b.empid

**where**

b.bonus < 1000

**or** b.bonus **is** **null**;



**----------------584. Find Customer Referee**

**CREATE** **TABLE** Customer (

id **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

referee\_id **int**

);

-- Insert data into the "Customer" table

**INSERT** **INTO** Customer (id, **name**, referee\_id)

**VALUES**

(1, 'Will', **NULL**),

(2, 'Jane', **NULL**),

(3, 'Alex', 2),

(4, 'Bill', **NULL**),

(5, 'Zack', 1),

(6, 'Mark', 2);

**select**

**name**

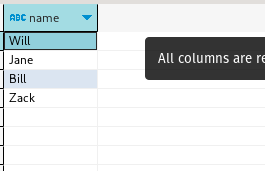
**from**

customer

**where**

referee\_id != 2

**or** referee\_id **is** **null**;



**----------------586. Customer Placing the Largest Number of Orders**

-- Create the "Orders" table

**CREATE** **TABLE** Orders (

order\_number **int** **PRIMARY** **KEY**,

customer\_number **int**,

order\_date **date**,

required\_date **date**,

shipped\_date **date**,

status **char**(15)

);

-- Insert data into the "Orders" table

**INSERT** **INTO** Orders (order\_number, customer\_number, order\_date, required\_date, shipped\_date, status)

**VALUES**

(1, 1, '2017-04-09', '2017-04-13', '2017-04-12', 'Closed'),

(2, 2, '2017-04-15', '2017-04-20', '2017-04-18', 'Closed'),

(3, 3, '2017-04-16', '2017-04-25', '2017-04-20', 'Closed'),

(4, 3, '2017-04-18', '2017-04-28', '2017-04-25', 'Closed');

**select**

customer\_number

**from**

(

**select**

customer\_number,

**count**(\*) **as** c

**from**

Orders

**group** **by**

customer\_number

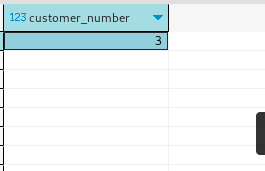
) t

**order** **by**

c **desc**

**limit**

1;



**----------------595. Big Countries**

-- Create the "World" table

**CREATE** **TABLE** World (

**name** **varchar**(255),

continent **varchar**(255),

**area** **int**,

population **int**,

gdp **int**

);

-- Insert data into the "World" table

**INSERT** **INTO** World (**name**, continent, **area**, population, gdp)

**VALUES**

('Afghanistan', 'Asia', 652230, 25500100, 20343000),

('Albania', 'Europe', 28748, 2831741, 12960000),

('Algeria', 'Africa', 2381741, 37100000, 188681000),

('Andorra', 'Europe', 468, 78115, 3712000),

('Angola', 'Africa', 1246700, 20609294, 100990000);

**select**

**name**,

population,

w.**area**

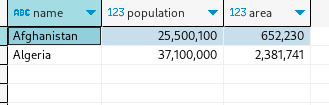
**from**

World w

**where**

w.**area** >= 3000000

**or** w.population >= 25000000



**----------------596. Classes More Than 5 Students**

-- Create the "course" table

**CREATE** **TABLE** courses (

student **varchar**(255),

**class** **varchar**(255)

);

-- Insert data into the "course" table

**INSERT** **INTO** courses (student, **class**)

**VALUES**

('A', 'Math'),

('B', 'English'),

('C', 'Math'),

('D', 'Biology'),

('E', 'Math'),

('F', 'Computer'),

('G', 'Math'),

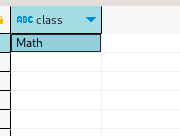
('H', 'Math'),

('I', 'Math');

**with** cte **as**

(**select** **class**, **count**(\*) c **from** courses **group** **by** **class**)

**select** **class** **from** cte **where** c>=5;



**----------------603. Consecutive Available Seats**

**CREATE** **TABLE** cinema (

seat\_id Serial **primary** **key**,

**free** **INT**

);

**INSERT** **INTO** cinema (seat\_id, **free**) **VALUES**

(1, 1),

(2, 0),

(3, 1),

(4, 1),

(5, 1);

**WITH** cte **AS** (

**SELECT**

\*,

**LAG**(**free**) **OVER** (

**ORDER** **BY**

seat\_id

) **AS** prev\_free,

**LEAD**(**free**) **OVER** (

**ORDER** **BY**

seat\_id

) **AS** next\_free

**FROM**

cinema

)

**select**

seat\_id

**from**

cte

**where**

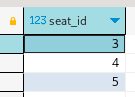
**free** = 1

**and** (

prev\_free = 1

**or** next\_free = 1

);



**-----------------607. Sales Person**

-- Create the "salesperson" table

**CREATE** **TABLE** salesperson (

sales\_id **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

salary **int**,

commission\_rate **int**,

hire\_date **date**

);

-- Insert data into the "salesperson" table

**INSERT** **INTO** salesperson (sales\_id, **name**, salary, commission\_rate, hire\_date)

**VALUES**

(1, 'John', 100000, 6, '2006-04-01'),

(2, 'Amy', 120000, 5, '2010-05-01'),

(3, 'Mark', 65000, 12, '2008-12-25'),

(4, 'Pam', 25000, 25, '2005-01-01'),

(5, 'Alex', 50000, 10, '2007-02-03');

-- Create the "company" table

**CREATE** **TABLE** company (

com\_id **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

city **varchar**(255)

);

-- Insert data into the "company" table

**INSERT** **INTO** company (com\_id, **name**, city)

**VALUES**

(1, 'RED', 'Boston'),

(2, 'ORANGE', 'New York'),

(3, 'YELLOW', 'Boston'),

(4, 'GREEN', 'Austin');

-- Create the "orders" table

**CREATE** **TABLE** orders (

order\_id **int** **PRIMARY** **KEY**,

order\_date **date**,

com\_id **int**,

sales\_id **int**,

amount **int**

);

-- Insert data into the "orders" table

**INSERT** **INTO** orders (order\_id, order\_date, com\_id, sales\_id, amount)

**VALUES**

(1, '2014-01-01', 3, 4, 100000),

(2, '2014-02-01', 4, 5, 5000),

(3, '2014-03-01', 1, 1, 50000),

(4, '2014-04-01', 1, 4, 25000);

**select**

**name**

**from**

SalesPerson

**where**

**name** **not** **in** (

**select**

s.**name**

**from**

SalesPerson s

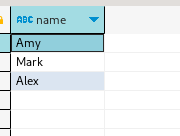
**left** **join** Orders o **on** s.sales\_id = o.sales\_id

**left** **join** Company c **on** o.com\_id = c.com\_id

**where**

c.**name** = 'RED'

);



**-----------------608. Tree Node**

-- Create the "tree" table

**CREATE** **TABLE** tree (

id **int** **PRIMARY** **KEY**,

p\_id **int**

);

-- Insert data into the "tree" table

**INSERT** **INTO** tree (id, p\_id)

**VALUES**

(1, **NULL**),

(2, 1),

(3, 1),

(4, 2),

(5, 2);

**select**

id,

**case** **when** id **in** (

**select**

p\_id

**from**

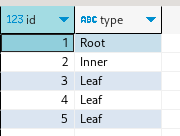
Tree

)

**and** p\_id **is** **not** **null** **then** 'Inner' **when** p\_id **is** **null** **then** 'Root' **else** 'Leaf' **end** **as** **type**

**from**

Tree



**-----------------610. Triangle Judgement**

-- Create the "triangle" table

**CREATE** **TABLE** triangle (

x **int**,

y **int**,

z **int**

);

-- Insert data into the "triangle" table

**INSERT** **INTO** triangle (x, y, z)

**VALUES**

(13, 15, 30),

(10, 20, 15);

**select**

\*,

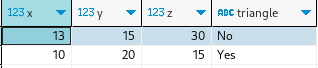
**case** **when** x + y > z

**and** x + z > y

**and** y + z > x **then** 'Yes' **else** 'No' **end** **as** triangle

**from**

triangle



**-----------------612. Shortest Distance in a Plane**

**CREATE** **TABLE** plane2d (

x **INT**,

y **INT**

);

**INSERT** **INTO** plane2d (x, y) **VALUES**

(-1, -1),

(0, 0),

(-1, -2);

**WITH** cte **AS** (

**SELECT**

p1.x **AS** x1,

p1.y **AS** y1,

p2.x **AS** x2,

p2.y **AS** y2,

**SQRT**(**POWER**(p2.x - p1.x, 2) + **POWER**(p2.y - p1.y, 2)) **AS** distance

**FROM**

plane2d p1

**CROSS** **JOIN**

plane2d p2

**WHERE**

p1.x != p2.x **OR** p1.y != p2.y

)

**SELECT** **MIN**(distance) **AS** shortest **FROM** cte;



**-----------------613. Shortest Distance in a Line**

**CREATE** **TABLE** **line** (

x **INT**

);

**INSERT** **INTO** **line** (x) **VALUES**

(-1),

(0),

(2);

**with** cte **as** (

**select**

\*,

**abs**(l1.x - l2.x) dist

**from**

**line** l1 **cross**

**join** **line** l2

**where**

l1.x != l2.x

)

**select**

**min**(dist) shortest

**from**

cte;



**-----------------619. Biggest Single Number**

-- Create the "mynumber" table

**CREATE** **TABLE** mynumbers (

num **int**

);

-- Insert data into the "mynumber" table

**INSERT** **INTO** mynumbers (num)

**VALUES**

(8),

(8),

(3),

(3),

(1),

(4),

(5),

(6);

**with** cte **as** (

**select**

\*,

**count**(\*) **over**(**partition** **by** num) c

**from**

mynumbers

)

**select**

**max**(num) **as** num

**from**

cte

**where**

cte.c = 1;



**-----------------620. Not Boring Movies**

-- Create the "cinema" table

**CREATE** **TABLE** cinema (

id **int** **PRIMARY** **KEY**,

movie **varchar**(255),

description **varchar**(255),

rating **float**

);

-- Insert data into the "cinema" table

**INSERT** **INTO** cinema (id, movie, description, rating)

**VALUES**

(1, 'War', 'great 3D', 8.9),

(2, 'Science fiction', **NULL**, 8.5),

(3, 'Irish', 'boring', 6.2),

(4, 'Ice song', 'Fantasy', 8.6),

(5, 'House card', 'Interesting', 9.1);

**select** \*

**from**

cinema

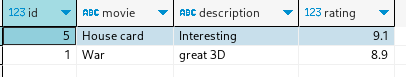
**where**

id % 2 != 0

**and** description != 'boring'

**order** **by**

rating **desc**;



**-----------------626. Exchange Seats**

**CREATE** **TABLE** seats (

id SERIAL **PRIMARY** **KEY**,

student **VARCHAR**(255)

);

**INSERT** **INTO** seats (student) **VALUES**

('Abbot'),

('Doris'),

('Emerson'),

('Green'),

('Jeames');

**with** cte **as** (

**select**

id cid,

student,

**case** **when** id % 2 = 0 **then** id - 1 **else** id + 1 **end** **as** swap

**from**

seats

),

cte2 **as** (

**select**

**max**(id) tot

**from**

seats

)

**select**

**case** **when** swap > tot **then** tot **else** swap **end** **as** id,

student

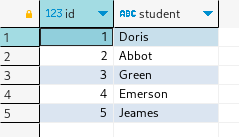
**from**

cte,

cte2

**order** **by**

Id;



**-----------------627. Swap Salary**

-- Create the "salary" table

**CREATE** **TABLE** salary (

id **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

sex **char**(1),

salary **int**

);

-- Insert data into the "salary" table

**INSERT** **INTO** salary (id, **name**, sex, salary)

**VALUES**

(1, 'A', 'm', 2500),

(2, 'B', 'f', 1500),

(3, 'C', 'm', 5500),

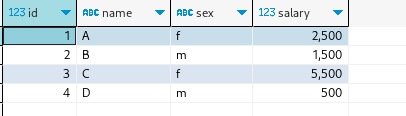
(4, 'D', 'f', 500);

**update**

salary

**set**

sex = **case** **when** sex = 'f' **then** 'm' **else** 'f' **end**



**-----------------1045. Customers Who Bought All Products**

-- Create the "Product" table

**CREATE** **TABLE** Product (

product\_key **int** **PRIMARY** **KEY**

);

-- Insert data into the "Product" table

**INSERT** **INTO** Product (product\_key)

**VALUES**

(5),

(6);

-- Create the "Customer" table

**CREATE** **TABLE** Customer (

customer\_id **int**,

product\_key **int**,

**FOREIGN** **KEY** (product\_key) **REFERENCES** Product(product\_key)

);

-- Insert data into the "Customer" table

**INSERT** **INTO** Customer (customer\_id, product\_key)

**VALUES**

(1, 5),

(2, 6),

(3, 5),

(3, 6),

(1, 6);

**SELECT**

customer\_id

**FROM**

Customer

**GROUP** **BY**

customer\_id

**HAVING**

**COUNT**(**DISTINCT** product\_key) = (

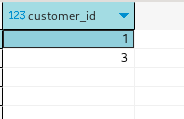
**SELECT**

**COUNT**(\*)

**FROM**

Product

)



**-----------------1050. Actors and Directors Who Cooperated At Least Three Times**

-- Create the "ActorDirector" table

**CREATE** **TABLE** ActorDirector (

actor\_id **int**,

director\_id **int**,

**timestamp** **int** **PRIMARY** **KEY**

);

-- Insert data into the "ActorDirector" table

**INSERT** **INTO** ActorDirector (actor\_id, director\_id, **timestamp**)

**VALUES**

(1, 1, 0),

(1, 1, 1),

(1, 1, 2),

(1, 2, 3),

(1, 2, 4),

(2, 1, 5),

(2, 1, 6);

**with** cte **as** (

**select**

actor\_id, director\_id ,

**count**(\*) pair

**from**

actordirector

**group** **by**

actor\_id,

director\_id

)

**select**

actor\_id,

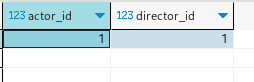
director\_id

**from**

cte

**where**

cte.pair >= 3;



**-----------------1068. Product Sales Analysis I**

**drop** **table** products;

-- Create the "Product" table

**CREATE** **TABLE** Products (

product\_id **int** **PRIMARY** **KEY**,

product\_name **varchar**(255)

);

-- Insert data into the "Product" table

**INSERT** **INTO** Products (product\_id, product\_name)

**VALUES**

(100, 'Nokia'),

(200, 'Apple'),

(300, 'Samsung');

-- Create the "Sales" table

**drop** **table** sales;

**CREATE** **TABLE** Sales (

sale\_id **int**,

product\_id **int**,

**year** **int**,

quantity **int**,

price **int**,

**PRIMARY** **KEY** (sale\_id, **year**),

**FOREIGN** **KEY** (product\_id) **REFERENCES** Products(product\_id)

);

-- Insert data into the "Sales" table

**INSERT** **INTO** Sales (sale\_id, product\_id, **year**, quantity, price)

**VALUES**

(1, 100, 2008, 10, 5000),

(2, 100, 2009, 12, 5000),

(7, 200, 2011, 15, 9000);

**select**

product\_name,

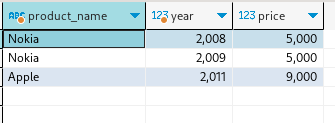
**year**,

price

**from**

sales s

**join** products p **on** s.product\_id = p.product\_id;



**-----------------1069. Product Sales Analysis II**

**select**

**distinct** product\_id,

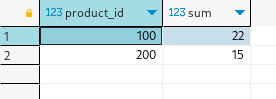
**sum**(quantity) **over**(**partition** **by** product\_id)

**from**

sales

**order** **by**

Product\_id;



**-----------------1070. Product Sales Analysis III**

-- Create the "Product" table

**drop** **table** products;

**CREATE** **TABLE** Products (

product\_id **int** **PRIMARY** **KEY**,

product\_name **varchar**(255)

);

-- Insert data into the "Product" table

**INSERT** **INTO** Products (product\_id, product\_name)

**VALUES**

(100, 'Nokia'),

(200, 'Apple'),

(300, 'Samsung');

-- Create the "Sales" table

**drop** **table** sales;

**CREATE** **TABLE** Sales (

sale\_id **int**,

product\_id **int**,

**year** **int**,

quantity **int**,

price **int**

);

-- Insert data into the "Sales" table

**INSERT** **INTO** Sales (sale\_id, product\_id, **year**, quantity, price)

**VALUES**

(1, 100, 2008, 10, 5000),

(2, 100, 2009, 12, 5000),

(7, 200, 2011, 15, 9000);

**with** cte **as** (

**select**

\*,

**min**(**year**) **over**(**partition** **by** product\_id) early\_year

**from**

sales

)

**select**

p.product\_id,

**year** first\_year,

quantity,

price

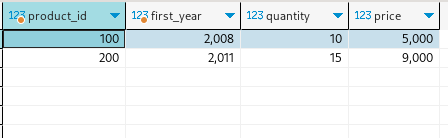
**from**

products p

**join** cte **on** p.product\_id = cte.product\_id

**where**

**year** = early\_year;



**-----------------1075. Project Employees I**

**drop** **table** employee;

-- Create the "Employee" table

**CREATE** **TABLE** Employee (

employee\_id **int** **PRIMARY** **KEY**,

**name** **varchar**(255),

experience\_years **int**

);

-- Insert data into the "Employee" table

**INSERT** **INTO** Employee (employee\_id, **name**, experience\_years)

**VALUES**

(1, 'Khaled', 3),

(2, 'Ali', 2),

(3, 'John', 1),

(4, 'Doe', 2);

-- Create the "Project" table

**CREATE** **TABLE** Project (

project\_id **int**,

employee\_id **int**,

**PRIMARY** **KEY** (project\_id, employee\_id),

**FOREIGN** **KEY** (employee\_id) **REFERENCES** Employee(employee\_id)

);

-- Insert data into the "Project" table

**INSERT** **INTO** Project (project\_id, employee\_id)

**VALUES**

(1, 1),

(1, 2),

(1, 3),

(2, 1),

(2, 4);

**select**

project\_id,

**round**(

**avg**(experience\_years),

2

) average\_years

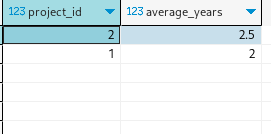
**from**

project p

**join** employee e **on** p.employee\_id = e.employee\_id

**group** **by**

project\_id;



**-----------------1076. Project Employees II**

**with** cte

**as** (**select** project\_id,

**count**(\*) c

**from** project p

**group** **by** project\_id

)

**select** project\_id

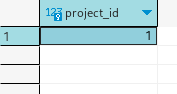
**from** cte

**where** c =

(

**select** **max**(c) **from** cte

);



**-----------------1077. Project Employees III**

**with** cte

**as** (**select** P.project\_id,

P.employee\_id,

**dense\_rank**() **over** (**partition** **by** P.project\_id **order** **by** E.experience\_years **desc**) **as** experience\_RANK

**from** Project P

**join** Employee E

**on** P.employee\_id = E.employee\_id

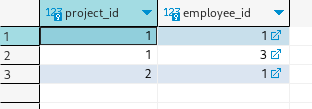
)

**select** project\_id,

employee\_id

**from** cte

**where** experience\_RANK = 1;



**-----------------1082. Sales Analysis I**

**with** cte

**as** (**select** **distinct**

seller\_id,

**sum**(price) **over** (**partition** **by** seller\_id) tot

**from** sales

)

**select** seller\_id

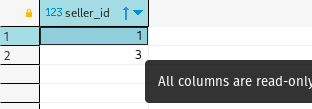
**from** cte

**where** cte.tot =

(

**select** **max**(tot) **from** cte

);



**-----------------1083. Sales Analysis II**

**SELECT** S.buyer\_id

**FROM** Product P

**INNER** **JOIN** Sales S

**ON** P.product\_id = S.product\_id

**GROUP** **BY** S.buyer\_id

**HAVING** **SUM**( **CASE**

**WHEN** P.product\_name = 'S8' **THEN**

1

**ELSE**

0

**END**

) > 0

**AND** **SUM**( **CASE**

**WHEN** P.product\_name = 'iPhone' **THEN**

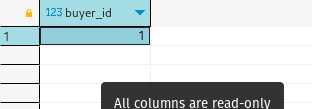
1

**ELSE**

0

**END**

) = 0;



**-----------------1084. Sales Analysis III**

-- Create the "Product" table

**drop** **table** sales;

**drop** **table** products;

**CREATE** **TABLE** Products (

product\_id **int** **PRIMARY** **KEY**,

product\_name **varchar**(255),

unit\_price **int**

);

-- Insert data into the "Product" table

**INSERT** **INTO** Products (product\_id, product\_name, unit\_price)

**VALUES**

(1, 'S8', 1000),

(2, 'G4', 800),

(3, 'iPhone', 1400);

-- Create the "Sales" table

**CREATE** **TABLE** Sales (

seller\_id **int**,

product\_id **int**,

buyer\_id **int**,

sale\_date **date**,

quantity **int**,

price **int**,

**FOREIGN** **KEY** (product\_id) **REFERENCES** Products(product\_id)

);

-- Insert data into the "Sales" table

**INSERT** **INTO** Sales (seller\_id, product\_id, buyer\_id, sale\_date, quantity, price)

**VALUES**

(1, 1, 1, '2019-01-21', 2, 2000),

(1, 2, 2, '2019-02-17', 1, 800),

(2, 2, 3, '2019-06-02', 1, 800),

(3, 3, 4, '2019-05-13', 2, 2800);

**SELECT**

s.product\_id,

**MAX**(p.product\_name) **AS** product\_name

**FROM**

Sales s

**LEFT** **JOIN** Products p **ON** s.product\_id = p.product\_id

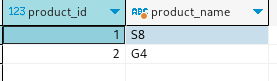
**WHERE**

sale\_date >= '2019-01-01'::**DATE**

**AND** sale\_date <= '2019-03-31'::**DATE**

**GROUP** **BY**

s.product\_id;



**-----------------1112. Highest Grade For Each Student**

**CREATE** **TABLE** enrollments (

student\_id **INT**,

course\_id **INT**,

grade **INT**,

**PRIMARY** **KEY** (student\_id, course\_id)

);

**INSERT** **INTO** Enrollments (student\_id, course\_id, grade) **VALUES**

(2, 2, 95),

(2, 3, 95),

(1, 1, 90),

(1, 2, 99),

(3, 1, 80),

(3, 2, 75),

(3, 3, 82);

**with** cte

**as** (**select** student\_id,

course\_id,

grade,

**rank**() **over** (**partition** **by** student\_id **order** **by** grade **desc**, course\_id **asc**) **as** **rank**

**from** enrollments

)

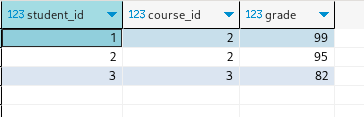
**select** student\_id,

course\_id,

grade

**from** cte

**where** **rank** = 1;



**-----------------1113. Reported Posts**

**CREATE** **TYPE** action\_enum **AS** **ENUM** ('view', 'like', 'reaction', 'comment', 'report', 'share');

**CREATE** **TABLE** Actions (

user\_id **INT**,

post\_id **INT**,

action\_date **DATE**,

**action** action\_enum,

extra **VARCHAR**(255)

);

**INSERT** **INTO** Actions (user\_id, post\_id, action\_date, **action**, extra) **VALUES**

(1, 1, '2019-07-01', 'view', **NULL**),

(1, 1, '2019-07-01', 'like', **NULL**),

(1, 1, '2019-07-01', 'share', **NULL**),

(2, 4, '2019-07-04', 'view', **NULL**),

(2, 4, '2019-07-04', 'report', 'spam'),

(3, 4, '2019-07-04', 'view', **NULL**),

(3, 4, '2019-07-04', 'report', 'spam'),

(4, 3, '2019-07-02', 'view', **NULL**),

(4, 3, '2019-07-02', 'report', 'spam'),

(5, 2, '2019-07-04', 'view', **NULL**),

(5, 2, '2019-07-04', 'report', 'racism'),

(5, 5, '2019-07-04', 'view', **NULL**),

(5, 5, '2019-07-04', 'report', 'racism');

**select** extra **as** report\_reason,

**count**(**distinct** post\_id) **as** report\_count

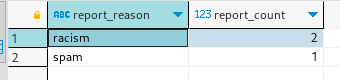
**from** Actions

**where** **action** = 'report'

**and** action\_date = '2019-07-04'

**group** **by** extra

**order** **by** report\_reason;



**-----------------1126. Active Businesses**

**CREATE** **TABLE** Events (

business\_id **INT**,

event\_type **VARCHAR**(255),

occurrences **INT**,

**PRIMARY** **KEY** (business\_id, event\_type)

);

**INSERT** **INTO** Events (business\_id, event\_type, occurrences) **VALUES**

(1, 'reviews', 7),

(3, 'reviews', 3),

(1, 'ads', 11),

(2, 'ads', 7),

(3, 'ads', 6),

(1, 'page views', 3),

(2, 'page views', 12);

**with** cte

**as** (**select** event\_type,

**avg**(occurrences) mean

**from** events

**group** **by** event\_type

)

**select** business\_id

**from**

(

**select** business\_id,

**count**(\*) cou

**from** events e

**join** cte c

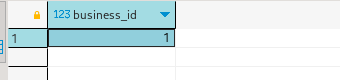
**on** e.event\_type = c.event\_type

**where** e.occurrences > c.mean

**group** **by** business\_id

) bc

**where** bc.cou >= 2;



**-----------------1141. User Activity for the Past 30 Days I**

-- Create the "Activity" table

**drop** **table** activity ;

**CREATE** **TABLE** Activity (

user\_id **int**,

session\_id **int**,

activity\_date **date**,

activity\_type **varchar**(255)

);

-- Insert data into the "Activity" table

**INSERT** **INTO** Activity (user\_id, session\_id, activity\_date, activity\_type)

**VALUES**

(1, 1, '2019-07-20', 'open\_session'),

(1, 1, '2019-07-20', 'scroll\_down'),

(1, 1, '2019-07-20', 'end\_session'),

(2, 4, '2019-07-20', 'open\_session'),

(2, 4, '2019-07-21', 'send\_message'),

(2, 4, '2019-07-21', 'end\_session'),

(3, 2, '2019-07-21', 'open\_session'),

(3, 2, '2019-07-21', 'send\_message'),

(3, 2, '2019-07-21', 'end\_session'),

(4, 3, '2019-06-25', 'open\_session'),

(4, 3, '2019-06-25', 'end\_session');

**SELECT**

activity\_date **AS** **day**,

**count**(**DISTINCT** user\_id) **AS** active\_users

**FROM**

Activity

**WHERE**

(

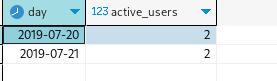
activity\_date > '2019-06-27'

**AND** activity\_date <= '2019-07-27'

)

**GROUP** **BY**

activity\_date;



**-----------------1142. User Activity for the Past 30 Days II**

**SELECT** **COALESCE**(

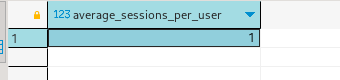
**ROUND**(**COUNT**(**DISTINCT** session\_id)::**NUMERIC** / **NULLIF**(**COUNT**(**DISTINCT** user\_id), 0.00), 2),

0.00

) **AS** average\_sessions\_per\_user

**FROM** Activity

**WHERE** activity\_date **BETWEEN** '2019-06-28' **AND** '2019-07-27';



**-----------------1148. Article Views I**

-- Create the "Views" table

**CREATE** **TABLE** **Views** (

article\_id **int**,

author\_id **int**,

viewer\_id **int**,

view\_date **date**

);

-- Insert data into the "Views" table

**INSERT** **INTO** **Views** (article\_id, author\_id, viewer\_id, view\_date)

**VALUES**

(1, 3, 5, '2019-08-01'),

(1, 3, 6, '2019-08-02'),

(2, 7, 7, '2019-08-01'),

(2, 7, 6, '2019-08-02'),

(4, 7, 1, '2019-07-22'),

(3, 4, 4, '2019-07-21'),

(3, 4, 4, '2019-07-21');

**select**

**distinct** author\_id id

**from**

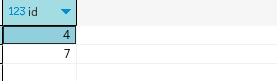
**views**

**where**

author\_id = viewer\_id

**order** **by**

author\_id **asc**;



**-----------------1164. Product Price at a Given Date**

-- Create the "Products" table

**drop** **table** products ;

**drop** **table** sales ;

**CREATE** **TABLE** Products (

product\_id **int**,

new\_price **int**,

change\_date **date**

);

-- Insert data into the "Products" table

**INSERT** **INTO** Products (product\_id, new\_price, change\_date)

**VALUES**

(1, 20, '2019-08-14'),

(2, 50, '2019-08-14'),

(1, 30, '2019-08-15'),

(1, 35, '2019-08-16'),

(2, 65, '2019-08-17'),

(3, 20, '2019-08-18');

**WITH** cte **AS** (

**SELECT**

\*,

**RANK**() **OVER** (

**PARTITION** **BY** product\_id

**ORDER** **BY**

change\_date **DESC**

) **AS** r

**FROM**

Products

**WHERE**

change\_date <= '2019-08-16'

)

**SELECT**

product\_id,

new\_price **AS** price

**FROM**

cte

**WHERE**

r = 1

**UNION**

**SELECT**

product\_id,

10 **AS** price

**FROM**

Products

**WHERE**

product\_id **NOT** **IN** (

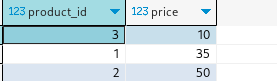
**SELECT**

product\_id

**FROM**

cte

);



**-----------------1173. Immediate Food Delivery I**

-- Create the "Delivery" table

**CREATE** **TABLE** Delivery (

delivery\_id **int**,

customer\_id **int**,

order\_date **date**,

customer\_pref\_delivery\_date **date**

);

-- Insert data into the "Delivery" table

**INSERT** **INTO** Delivery (delivery\_id, customer\_id, order\_date, customer\_pref\_delivery\_date)

**VALUES**

(1, 1, '2019-08-01', '2019-08-02'),

(2, 5, '2019-08-02', '2019-08-02'),

(3, 1, '2019-08-11', '2019-08-11'),

(4, 3, '2019-08-24', '2019-08-26'),

(5, 4, '2019-08-21', '2019-08-22'),

(6, 2, '2019-08-11', '2019-08-13');

**WITH** cte1 **AS** (

**SELECT** **COUNT**(\*) **AS** c **FROM** delivery

),

cte2 **AS** (

**SELECT** **COUNT**(\*) **AS** c

**FROM** delivery

**WHERE** order\_date = customer\_pref\_delivery\_date

)

**SELECT** **ROUND**((cte2.c::**numeric** / cte1.c) \* 100, 2) **AS** immediate\_percentage

FROM cte1, cte2;



**-----------------1174. Immediate Food Delivery II**

**drop** **table** delivery;

**CREATE** **TABLE** Delivery (

delivery\_id serial **PRIMARY** **KEY**,

customer\_id **int**,

order\_date **date**,

customer\_pref\_delivery\_date **date**

);

**INSERT** **INTO** Delivery (customer\_id, order\_date, customer\_pref\_delivery\_date)

**VALUES**

(1, '2019-08-01', '2019-08-02'),

(2, '2019-08-02', '2019-08-02'),

(1, '2019-08-11', '2019-08-12'),

(3, '2019-08-24', '2019-08-24'),

(3, '2019-08-21', '2019-08-22'),

(2, '2019-08-11', '2019-08-13'),

(4, '2019-08-09', '2019-08-09');

**SELECT**

**ROUND**(

(**SUM**(**CASE** **WHEN** order\_date = customer\_pref\_delivery\_date **THEN** 1 **ELSE** 0 **END**) \* 100.0) / **COUNT**(\*),

2

) **AS** immediate\_percentage

**FROM** Delivery

**WHERE** (customer\_id, order\_date) **IN** (

**SELECT** customer\_id, **MIN**(order\_date)

**FROM** Delivery

**GROUP** **BY** customer\_id

);



**-----------------1179. Reformat Department TABLE**

**CREATE** TABLE Department (

id **int**,

revenue **int**,

**month** **varchar**

);

**INSERT** INTO Department (id, revenue, **month**)

**VALUES**

(1, 8000, 'Jan'),

(2, 9000, 'Jan'),

(3, 10000, 'Feb'),

(1, 7000, 'Feb'),

(1, 6000, 'Mar');

**SELECT**

id,

**MAX**(**CASE** **WHEN** **month** = 'Jan' **THEN** revenue **ELSE** **NULL** **END**) **AS** Jan,

**MAX**(**CASE** **WHEN** **month** = 'Feb' **THEN** revenue **ELSE** **NULL** **END**) **AS** Feb,

**MAX**(**CASE** **WHEN** month = 'Mar' **THEN** revenue **ELSE** NULL END) **AS** Mar,

**MAX**(**CASE** WHEN **month** = 'Apr' **THEN** revenue **ELSE** **NULL** **END**) **AS** Apr,

**MAX**(**CASE** **WHEN** **month** = 'May' THEN revenue **ELSE** **NULL** END) **AS** May,

**MAX**(**CASE** **WHEN** **month** = 'Jun' **THEN** revenue **ELSE** **NULL** **END**) **AS** Jun,

**MAX**(**CASE** **WHEN** **month** = 'Jul' **THEN** revenue **ELSE** **NULL** **END**) AS Jul,

**MAX**(**CASE** **WHEN** **month** = 'Aug' **THEN** revenue ELSE **NULL** **END**) AS Aug,

**MAX**(**CASE** **WHEN** **month** = 'Sep' **THEN** revenue **ELSE** **NULL** END) **AS** Sep,

**MAX**(**CASE** **WHEN** **month** = 'Oct' **THEN** revenue **ELSE** **NULL** **END**) **AS** Oct,

**MAX**(**CASE** **WHEN** **month** = 'Nov' **THEN** revenue **ELSE** **NULL** **END**) **AS** Nov,

**MAX**(**CASE** **WHEN** **month** = 'Dec' THEN revenue **ELSE** **NULL** **END**) **AS** **Dec**

**FROM** Department

**GROUP** **BY** id;

**-----------------1193. Monthly Transactions I**

**CREATE** **TABLE** Transactions (

id **int**,

country **varchar**,

state **varchar**,

amount **int**,

trans\_date **date**

);

**INSERT** **INTO** Transactions (id, country, state, amount, trans\_date)

**VALUES**

(121, 'US', 'approved', 1000, '2018-12-18'),

(122, 'US', 'declined', 2000, '2018-12-19'),

(123, 'US', 'approved', 2000, '2019-01-01'),

(124, 'DE', 'approved', 2000, '2019-01-07');

**SELECT**

**TO\_CHAR**(trans\_date, 'YYYY-MM') **AS** **month**,

country,

**COUNT**(\*) **AS** trans\_count,

**SUM**(**CASE** **WHEN** state = 'approved' **THEN** 1 **ELSE** 0 **END**) **AS** approved\_count,

**SUM**(amount) **AS** trans\_total\_amount,

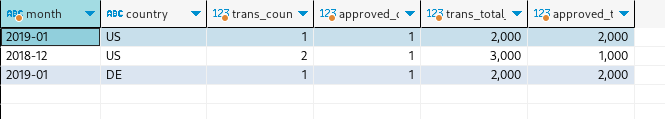
**SUM**(**CASE** **WHEN** state = 'approved' **THEN** amount **ELSE** 0 **END**) **AS** approved\_total\_amount

**FROM**

Transactions

**GROUP** **BY**

country,

**TO\_CHAR**(trans\_date, 'YYYY-MM');

**-----------------1204. Last Person to Fit in the Elevator**

**CREATE** **TABLE** Queue (

person\_id **int**,

person\_name **varchar**,

weight **int**,

turn **int**

);

**INSERT** **INTO** Queue (person\_id, person\_name, weight, turn)

**VALUES**

(5, 'George Washington', 250, 1),

(3, 'John Adams', 350, 2),

(6, 'Thomas Jefferson', 400, 3),

(2, 'Will Johnliams', 200, 4),

(4, 'Thomas Jefferson', 175, 5),

(1, 'James Elephant', 500, 6);

**with** cte **as** (**SELECT** person\_name,weight,turn,**SUM**(weight) **over** (**order** **by** turn) **as** total **from** Queue)

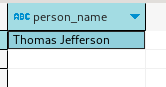
**SELECT** person\_name

**FROM** cte

**where** total <= 1000

**order** **by** total **DESC**

**LIMIT** 1;



**-----------------1211. Queries Quality and Percentage**

**CREATE** **TABLE** Queries (

query\_name **varchar**,

**result** **varchar**,

**position** **int**,

rating **int**

);

**INSERT** **INTO** Queries (query\_name, **result**, **position**, rating)

**VALUES**

('Dog', 'Golden Retriever', 1, 5),

('Dog', 'German Shepherd', 2, 5),

('Dog', 'Mule', 200, 1),

('Cat', 'Shirazi', 5, 2),

('Cat', 'Siamese', 3, 3),

('Cat', 'Sphynx', 7, 4);

**SELECT**

query\_name,

**ROUND**(

**AVG**(rating / **position**),

2

) **AS** quality,

**ROUND**(

100 \* **AVG**(

**CASE** **WHEN** rating < 3 **THEN** 1 **ELSE** 0 **END**

),

2

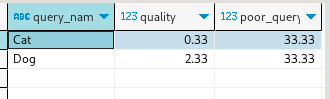
) **AS** poor\_query\_percentage

**FROM**

Queries

**GROUP** **BY**

query\_name;



**-----------------1212. Team Scores in Football Tournament**

**CREATE** **TABLE** Teams (

team\_id **int** **PRIMARY** **KEY**,

team\_name **varchar**

);

**INSERT** **INTO** Teams (team\_id, team\_name)

**VALUES**

(10, 'Leetcode FC'),

(20, 'NewYork FC'),

(30, 'Atlanta FC'),

(40, 'Chicago FC'),

(50, 'Toronto FC');

**CREATE** **TABLE** Matches (

match\_id **int** **PRIMARY** **KEY**,

host\_team **int**,

guest\_team **int**,

host\_goals **int**,

guest\_goals **int**,

**FOREIGN** **KEY** (host\_team) **REFERENCES** Teams(team\_id),

**FOREIGN** **KEY** (guest\_team) **REFERENCES** Teams(team\_id)

);

**INSERT** **INTO** Matches (match\_id, host\_team, guest\_team, host\_goals, guest\_goals)

**VALUES**

(1, 10, 20, 3, 0),

(2, 30, 10, 2, 2),

(3, 10, 50, 5, 1),

(4, 20, 30, 1, 0),

(5, 50, 30, 1, 0);

**SELECT**

team\_id,

team\_name,

**SUM**(

**CASE**

**WHEN** team\_id = host\_team **AND** host\_goals > guest\_goals **THEN** 3

**WHEN** team\_id = guest\_team **AND** guest\_goals > host\_goals **THEN** 3

**WHEN** host\_goals = guest\_goals **THEN** 1

**ELSE** 0

**END**

) **AS** num\_points

**FROM**

Teams

**LEFT** **JOIN**

Matches **ON** team\_id = host\_team **OR** team\_id = guest\_team

**GROUP** **BY**

team\_id, team\_name

**ORDER** **BY**

num\_points **DESC**, team\_id;



**-----------------1225. Report Contiguous Dates**

**CREATE** **TABLE** Failed (

fail\_date **DATE** **PRIMARY** **KEY**

);

**INSERT** **INTO** Failed (fail\_date) **VALUES**

('2018-12-28'),

('2018-12-29'),

('2019-01-04'),

('2019-01-05');

**CREATE** **TABLE** Succeeded (

success\_date **DATE** **PRIMARY** **KEY**

);

**INSERT** **INTO** Succeeded (success\_date) **VALUES**

('2018-12-30'),

('2018-12-31'),

('2019-01-01'),

('2019-01-02'),

('2019-01-03'),

('2019-01-06');

**-----------------1241. Number of Comments per Post**

**CREATE** **TABLE** Submissions (

sub\_id **int**,

parent\_id **int**

);

**INSERT** **INTO** Submissions (sub\_id, parent\_id)

**VALUES**

(1, **Null**),

(2, **Null**),

(1, **Null**),

(12, **Null**),

(3, 1),

(5, 2),

(3, 1),

(4, 1),

(9, 1),

(10, 2),

(6, 7);

**with** cte1 **as** (

**select**

**distinct** sub\_id

**from**

submissions

**where**

parent\_id **is** **null**

),

cte **as** (

**select**

**distinct** sub\_id,

parent\_id

**from**

submissions

**where**

parent\_id **is** **not** **null**

)

**select**

cte1.sub\_id,

**coalesce** (tempo.c, 0)

**from**

cte1

**left** **join** (

**select**

parent\_id,

**count**(\*) c

**from**

cte

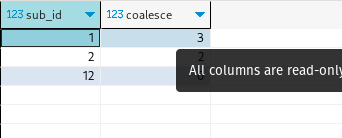
**group** **by**

parent\_id

) tempo **on** tempo.parent\_id = cte1.sub\_id

**order** **by**

cte1.sub\_id;



**-----------------1251. Average Selling Price**

**CREATE** **TABLE** Prices (

product\_id **int**,

start\_date **date**,

end\_date **date**,

price **int**

);

**CREATE** **TABLE** UnitsSold (

product\_id **int**,

purchase\_date **date**,

units **int**

);

-- Insert data into Prices table

**INSERT** **INTO** Prices (product\_id, start\_date, end\_date, price)

**VALUES**

(1, '2019-02-17', '2019-02-28', 5),

(1, '2019-03-01', '2019-03-22', 20),

(2, '2019-02-01', '2019-02-20', 15),

(2, '2019-02-21', '2019-03-31', 30);

-- Insert data into UnitsSold table

**INSERT** **INTO** UnitsSold (product\_id, purchase\_date, units)

**VALUES**

(1, '2019-02-25', 100),

(1, '2019-03-01', 15),

(2, '2019-02-10', 200),

(2, '2019-03-22', 30);

**with** cte **as** (

**select**

p.product\_id,

p.price \* u.units tot,

u.units

**from**

prices p

**join** unitssold u **on** purchase\_date **between** start\_date

**and** end\_date

**and** p.product\_id = u.product\_id

)

**select**

product\_id,

**round**(

**sum**(tot)/ **sum**(units):: **decimal**,

2

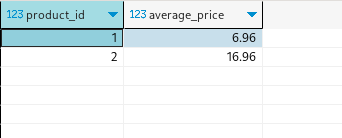
) average\_price

**from**

cte

**group** **by**

product\_id;



**-----------------1264. Page Recommendations**

**CREATE** **TABLE** Friendship (

user1\_id **int**,

user2\_id **int**

);

**CREATE** **TABLE** Likes (

user\_id **int**,

page\_id **int**

);

-- Insert data into Friendship table

**INSERT** **INTO** Friendship (user1\_id, user2\_id) **VALUES**

(1, 2),

(1, 3),

(1, 4),

(2, 3),

(2, 4),

(2, 5),

(6, 1);

-- Insert data into Likes table

**INSERT** **INTO** Likes (user\_id, page\_id) **VALUES**

(1, 88),

(2, 23),

(3, 24),

(4, 56),

(5, 11),

(6, 33),

(2, 77),

(3, 77),

(6, 88);

**with** cte **as** (

**select**

**case** **when** user1\_id = 1 **then** user2\_id **when** user2\_id = 1 **then** user1\_id **else** 0 **end** **as** friend

**from**

friendship

)

**select**

**distinct** page\_id recommended\_page

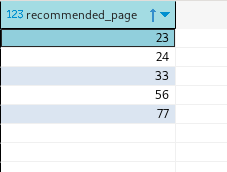
**from**

cte

**join** likes **on** cte.friend = likes.user\_id

**where**

likes.page\_id != 88;



**-----------------1280. Students and Examinations**

**CREATE** **TABLE** Students (

student\_id **int** **PRIMARY** **KEY**,

student\_name **varchar**

);

**CREATE** **TABLE** Subjects (

subject\_name **varchar** **PRIMARY** **KEY**

);

**CREATE** **TABLE** Examinations (

student\_id **int**,

subject\_name **varchar**,

**FOREIGN** **KEY** (student\_id) **REFERENCES** Students(student\_id),

**FOREIGN** **KEY** (subject\_name) **REFERENCES** Subjects(subject\_name)

);

-- Insert data into Students table

**INSERT** **INTO** Students (student\_id, student\_name)

**VALUES**

(1, 'Alice'),

(2, 'Bob'),

(13, 'John'),

(6, 'Alex');

-- Insert data into Subjects table

**INSERT** **INTO** Subjects (subject\_name)

**VALUES**

('Math'),

('Physics'),

('Programming');

-- Insert data into Examinations table

**INSERT** **INTO** Examinations (student\_id, subject\_name)

**VALUES**

(1, 'Math'),

(1, 'Physics'),

(1, 'Programming'),

(2, 'Programming'),

(1, 'Physics'),

(1, 'Math'),

(13, 'Math'),

(13, 'Programming'),

(13, 'Physics'),

(2, 'Math'),

(1, 'Math');

**SELECT**

s.student\_id,

s.student\_name,

sub.subject\_name,

**COUNT**(e.subject\_name) **AS** attended\_exams

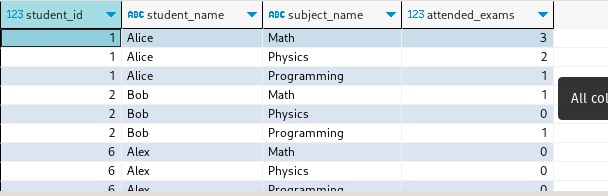
**FROM** Students s

**CROSS** **JOIN** Subjects sub

**LEFT** **JOIN** Examinations e **ON** s.student\_id = e.student\_id **AND** sub.subject\_name = e.subject\_name

**GROUP** **BY** s.student\_id, s.student\_name, sub.subject\_name

**ORDER** **BY** s.student\_id, sub.subject\_name;



**-----------------1285. Find the Start and End Number of Continuous Ranges**

**CREATE** **TABLE** Logs (

log\_id **int** **PRIMARY** **KEY**

);

**INSERT** **INTO** Logs (log\_id) **VALUES**

(1),

(2),

(3),

(7),

(8),

(10);

**select** **min**(log\_id) **as** start\_id, **max**(log\_id) **as** end\_id

**from** (**select** l.log\_id, (l.log\_id - l.row\_num) **as** diff

**from** (**select** log\_id, **row\_number**() **over**() **as** row\_num **from** Logs) l

) l2

**group** **by** diff **order** **by** start\_id;



**-----------------1294. Weather Type in Each Country**

**CREATE** **TABLE** Countries (

country\_id **int** **PRIMARY** **KEY**,

country\_name **varchar**

);

**drop** **table** weather;

**CREATE** **TABLE** Weather (

country\_id **int**,

weather\_state **varchar**,

**day** **date**,

**PRIMARY** **KEY** (country\_id, **day**)

);

-- Inserting data into Countries table

**INSERT** **INTO** Countries (country\_id, country\_name)

**VALUES**

(2, 'USA'),

(3, 'Australia'),

(7, 'Peru'),

(5, 'China'),

(8, 'Morocco'),

(9, 'Spain');

-- Inserting data into Weather table

**INSERT** **INTO** Weather (country\_id, weather\_state, **day**)

**VALUES**

(2, '15', '2019-11-01'),

(2, '12', '2019-10-28'),

(2, '12', '2019-10-27'),

(3, '-2', '2019-11-10'),

(3, '0', '2019-11-11'),

(3, '3', '2019-11-12'),

(5, '16', '2019-11-07'),

(5, '18', '2019-11-09'),

(5, '21', '2019-11-23'),

(7, '25', '2019-11-28'),

(7, '22', '2019-12-01'),

(7, '20', '2019-12-02'),

(8, '25', '2019-11-05'),

(8, '27', '2019-11-15'),

(8, '31', '2019-11-25'),

(9, '7', '2019-10-23'),

(9, '3', '2019-12-23');

**with** cte **as** (

**select**

c.country\_name,

**round**(

**avg**(w.weather\_state :: **int**),

2

) avg\_temp

**from**

countries c

**join** weather w **on** c.country\_id = w.country\_id

**group** **by**

c.country\_name

)

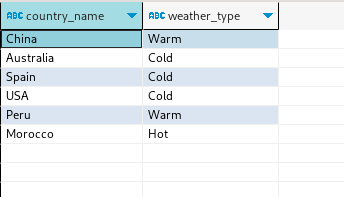
**select**

country\_name,

**case** **when** avg\_temp <= 15 **then** 'Cold' **when** avg\_temp >= 25 **then** 'Hot' **else** 'Warm' **end** **as** weather\_type

**from**

cte;



**-----------------1303. Find the Team Size**

**CREATE** **TABLE** Employees (

employee\_id **INT** **PRIMARY** **KEY**,

team\_id **INT**

);

**INSERT** **INTO** Employees (employee\_id, team\_id)

**VALUES**

(1, 8),

(2, 8),

(3, 8),

(4, 7),

(5, 9),

(6, 9);

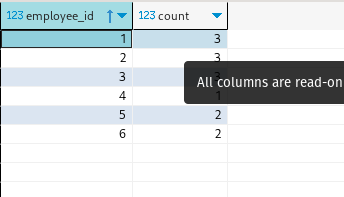
**select**

employee\_id,

**count**(team\_id) **over**(**partition** **by** team\_id)

**from**

employees e;



**-----------------1308. Running Total for Different Genders**

**CREATE** **TABLE** Scores (

player\_name **varchar**,

gender **varchar**,

**day** **date**,

score\_points **int**

);

**INSERT** **INTO** Scores (player\_name, gender, **day**, score\_points)

**VALUES**

('Aron', 'F', '2020-01-01', 17),

('Alice', 'F', '2020-01-07', 23),

('Bajrang', 'M', '2020-01-07', 7),

('Khali', 'M', '2019-12-25', 11),

('Slaman', 'M', '2019-12-30', 13),

('Joe', 'M', '2019-12-31', 3),

('Jose', 'M', '2019-12-18', 2),

('Priya', 'F', '2019-12-31', 23),

('Priyanka', 'F', '2019-12-30', 17);

**select**

s.gender,

s.**day**,

**sum**(s.score\_points) **over**(

**partition** **by** (s.gender)

**order** **by**

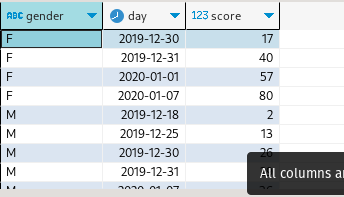
s.**day** **rows** **between** **unbounded** **preceding**

**and** **current** **row**

) score

**from**

scores s;



**-----------------1321. Restaurant Growth**

**CREATE** **TABLE** Customer (

customer\_id **int**,

**name** **varchar**,

visited\_on **date**,

amount **int**,

**PRIMARY** **KEY** (customer\_id, visited\_on)

);

**INSERT** **INTO** Customer (customer\_id, **name**, visited\_on, amount)

**VALUES**

(1, 'Jhon', '2019-01-01', 100),

(2, 'Daniel', '2019-01-02', 110),

(3, 'Jade', '2019-01-03', 120),

(4, 'Khaled', '2019-01-04', 130),

(5, 'Winston', '2019-01-05', 110),

(6, 'Elvis', '2019-01-06', 140),

(7, 'Anna', '2019-01-07', 150),

(8, 'Maria', '2019-01-08', 80),

(9, 'Jaze', '2019-01-09', 110),

(1, 'Jhon', '2019-01-10', 130),

(3, 'Jade', '2019-01-10', 150);

**SELECT** visited\_on, amount, **ROUND**(average\_amount, 2) **AS** average\_amount

**FROM** (

**SELECT**

visited\_on,

**SUM**(amount) **OVER** (**ORDER** **BY** visited\_on **ROWS** **BETWEEN** 6 **PRECEDING** **AND** **CURRENT** **ROW**) **AS** amount,

**AVG**(amount) **OVER** (**ORDER** **BY** visited\_on **ROWS** **BETWEEN** 6 **PRECEDING** **AND** **CURRENT** **ROW**) **AS** average\_amount

**FROM** (

**SELECT** visited\_on, **SUM**(amount) **AS** amount

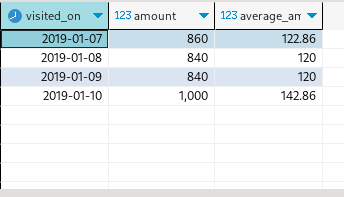
**FROM** Customer

**GROUP** **BY** visited\_on

) a

) b

**WHERE** **EXTRACT**(**DAY** **FROM** visited\_on) - **EXTRACT**(**DAY** **FROM** (**SELECT** **MIN**(visited\_on) **FROM** Customer)) >= 6;



**-----------------1322. Ads Performance**

**drop** **table** ads;

**CREATE** **TABLE** Ads (

ad\_id **int**,

user\_id **int**,

**action** **varchar**(10),

**primary** **key** (ad\_id, user\_id)

);

**INSERT** **INTO** Ads (ad\_id, user\_id, **action**) **VALUES**

(1, 1, 'Clicked'),

(2, 2, 'Clicked'),

(3, 3, 'Viewed'),

(5, 5, 'Ignored'),

(1, 7, 'Ignored'),

(2, 7, 'Viewed'),

(3, 5, 'Clicked'),

(1, 4, 'Viewed'),

(2, 11, 'Viewed'),

(1, 2, 'Clicked');

**SELECT** ad\_id,

**CASE**

**WHEN** clicks + **views** = 0 **THEN** 0.00

**ELSE** **ROUND**(100.00 \* clicks / (clicks + **views**), 2)

**END** ctr

**FROM**

(**SELECT** ad\_id,

**SUM**(**CASE**

**WHEN** **action** ='Viewed' **THEN** 1

**ELSE** 0

**END**) **views**,

**SUM**(**CASE**

**WHEN** **action** = 'Clicked' **THEN** 1

**ELSE** 0

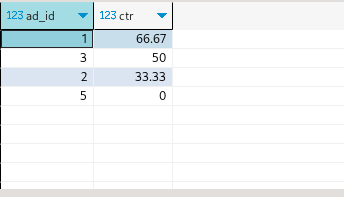
**END**) clicks

**FROM** Ads

**GROUP** **BY** ad\_id) a

**ORDER** **BY** ctr **DESC**,

ad\_id **ASC**;



**-----------------1327. List the Products Ordered in a Period**

-- Create Products table

**CREATE** **TABLE** Products (

product\_id **int** **PRIMARY** **KEY**,

product\_name **varchar**,

product\_category **varchar**

);

-- Create Orders table

**CREATE** **TABLE** Orders (

product\_id **int**,

order\_date **date**,

unit **int**

);

-- Insert data into Products table

**INSERT** **INTO** Products (product\_id, product\_name, product\_category)

**VALUES**

(1, 'Leetcode Solutions', 'Book'),

(2, 'Jewels of Stringology', 'Book'),

(3, 'HP', 'Laptop'),

(4, 'Lenovo', 'Laptop'),

(5, 'Leetcode Kit', 'T-shirt');

-- Insert data into Orders table

**INSERT** **INTO** Orders (product\_id, order\_date, unit)

**VALUES**

(1, '2020-02-05', 60),

(1, '2020-02-10', 70),

(2, '2020-01-18', 30),

(2, '2020-02-11', 80),

(3, '2020-02-17', 2),

(3, '2020-02-24', 3),

(4, '2020-03-01', 20),

(4, '2020-03-04', 30),

(4, '2020-03-04', 60),

(5, '2020-02-25', 50),

(5, '2020-02-27', 50),

(5, '2020-03-01', 50);

**select**

p.product\_name,

**sum**(o.unit) **as** unit

**from**

products p

**join** orders o **on** p.product\_id = o.product\_id

**where**

**extract**(

**month**

**from**

o.order\_date

) = '02'

**and** **extract**(

**year**

**from**

o.order\_date

) = '2020'

**group** **by**

p.product\_name

**having**

**sum**(o.unit)>= 100;

