use sql1;

#19

CREATE TABLE delivery (

delivery\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

customer\_pref\_delivery\_date DATE

);

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');

select

round(count

(

case

when order\_date = customer\_pref\_delivery\_date then 1

end

)/count(\*) \* 100,2) as immediate\_percent

from delivery;

#Q20

CREATE TABLE ad\_action (

ad\_id INT,

user\_id INT,

action ENUM('clicked', 'viewed', 'liked'),

PRIMARY KEY (ad\_id, user\_id)

);

ALTER TABLE ad\_action

MODIFY COLUMN action ENUM('Clicked', 'Viewed', 'Ignored');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (1, 1, 'clicked');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (2, 2, 'clicked');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (3, 3, 'viewed');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (5, 5, 'ignored');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (1, 7, 'ignored');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (2, 7, 'viewed');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (3, 5, 'clicked');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (1, 4, 'viewed');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (2, 11, 'viewed');

INSERT INTO ad\_action (ad\_id, user\_id, action) VALUES (1, 2, 'clicked');

select \* from ad\_action;

select ad\_id,

case

when total\_clicks+total\_views= 0 then 0

else round((total\_clicks / (total\_clicks + total\_views))\*100,2)

end as ctr

from

(

select

ad\_id,

sum(CASE WHEN action='Clicked' THEN 1 END) as total\_clicks,

sum(CASE WHEN action='Viewed' THEN 1 END) as total\_views

from ad\_action

group by ad\_id

) as ad\_stats

order by

ctr desc,

ad\_id asc;

#21

create table employee\_team (

employee\_id int primary key,

team\_id int

);

insert into employee\_team (employee\_id,team\_id)

values

(1,8),

(2,8),

(3,8),

(4,7),

(5,9),

(6,9);

show tables;

select \* from employee\_team;

select a.employee\_id,b.team\_size

from

employee\_team a

join

(

select team\_id,count(\*) as team\_size

from

employee\_team

group by team\_id

) b

on a.team\_id = b.team\_id;

#Q22

create table countries (

country\_id int primary key,

country\_name varchar(255)

);

create table weather\_forecast (

country\_id int,

weather\_state int,

day date,

primary key (country\_id,day)

);

INSERT INTO countries (country\_id, country\_name)

VALUES

(2, 'USA'),

(3, 'Australia'),

(7, 'Peru'),

(5, 'China'),

(8, 'Morocco'),

(9, 'Spain');

INSERT INTO weather\_forecast (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');

select

c.country\_name,

CASE

WHEN avg(w.weather\_state) <= 15 THEN 'COLD'

WHEN avg(w.weather\_state) >= 15 THEN 'HOT'

ELSE 'WARM'

END as weather\_type

FROM

countries c

inner join

weather\_forecast w on c.country\_id = w.country\_id

where

w.day >= '2019-11-01' AND w.day <= '2019-11-30'

group by country\_name;

show tables;

select \* from countries;

select \* from weather\_forecast;

select a.\*,b.\*

from countries a

inner join

(

select

country\_id,

case

when ws >=25 then 'HOT'

when ws<=15 then 'COLD'

else 'WARM'

end as ws\_verdict

from

(

select

country\_id,sum(weather\_state)/count(\*) as ws

from

weather\_forecast

group by country\_id

) as t

) b

on a.country\_id = b.country\_id;

#Q23

create table product\_pricing

(

product\_id int,

start\_date date,

end\_date date,

price int,

primary key (product\_id,start\_date,end\_date)

);

CREATE TABLE product\_purchase (

product\_id INT,

purchase\_date DATE,

units INT

);

alter table product\_purchase

rename to units\_sold;

alter table product\_pricing

rename to prices;

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 INTO units\_sold (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);

show tables;

select \* from prices;

select \* from units\_sold;

select p.\*,u.\*

from prices p

inner join

units\_sold u

on p.product\_id = u.product\_id and u.purchase\_date between p.start\_date and p.end\_date;

select p.product\_id,round(sum(p.price \* u.units)/sum(units),2)

from prices p

inner join

units\_sold u

on p.product\_id = u.product\_id and u.purchase\_date between p.start\_date and p.end\_date

group by product\_id;

#24

create table activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int

);

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);

select \* from activity;

with rankeddevice as(

select player\_id,

event\_date,

RANK() OVER (partition by player\_id order by event\_date asc) as devicerank

from activity

)

select player\_id,event\_date

from rankeddevice

where devicerank = 1;

#25

select \* from activity;

with rankeddevice as(

select player\_id,

device\_id,

RANK() OVER (partition by player\_id order by event\_date asc) as devicerank

from activity

)

select player\_id,device\_id

from rankeddevice

where devicerank = 1;

#26

create table products (

product\_id int primary key,

product\_name varchar(255),

product\_category varchar(255)

);

create table orders (

product\_id int,

order\_date date,

unit int,

foreign key (product\_id) references Products(product\_id)

);

INSERT INTO products (product\_id, product\_name, product\_category) VALUES

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

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

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

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

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

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 \* from products;

select \* from orders;

select pt.product\_id,pt.product\_name,sum(o.unit) as sales

from

products pt

inner join

orders o

on pt.product\_id = o.product\_id

where o.order\_date between '2020-02-01' and '2020-02-29'

group by pt.product\_id,pt.product\_name

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

#27

create table users (

user\_id int primary key,

name varchar(255),

mail varchar(255)

);

insert into users (user\_id,name,mail) VALUES

(1, 'Winston', 'winston@leetcode.com'),

(2, 'Jonathan', 'jonathanisgreat'),

(3, 'Annabelle', 'bella-@leetcode.com'),

(4, 'Sally', 'sally.come@leetcode.com'),

(5, 'Marwan', 'quarz#2020@leetcode.com'),

(6, 'David', 'david69@gmail.com'),

(7, 'Shapiro', '.shapo@leetcode.com');

select \* from users;

select \* from users where mail regexp '^[a-zA-z][a-zA-z0-9\_.-]\*@leetcode.com$';

#28

create table customer (

customer\_id int primary key,

name varchar(255),

country varchar(255)

);

insert into customer (customer\_id,name,country) values

(1, 'Winston', 'USA'),

(2, 'Jonathan', 'Peru'),

(3, 'Moustafa', 'Egypt');

CREATE TABLE products\_28 (

product\_id INT,

description VARCHAR(255),

price INT

);

INSERT INTO products\_28 (product\_id, description, price) VALUES

(10, 'LC Phone', 300),

(20, 'LC T-Shirt', 10),

(30, 'LC Book', 45),

(40, 'LC Keychain', 2);

CREATE TABLE orders\_28 (

order\_id INT PRIMARY KEY,

customer\_id INT,

product\_id INT,

order\_date DATE,

quantity INT

);

INSERT INTO orders\_28 (order\_id, customer\_id, product\_id, order\_date, quantity) VALUES

(1, 1, 10, '2020-06-10', 1),

(2, 1, 20, '2020-07-01', 1),

(3, 1, 30, '2020-07-08', 2),

(4, 2, 10, '2020-06-15', 2),

(5, 2, 40, '2020-07-01', 10),

(6, 3, 20, '2020-06-24', 2),

(7, 3, 30, '2020-06-25', 2),

(9, 3, 30, '2020-05-08', 3);

select \* from customer;

select \* from products\_28;

select \* from orders\_28;

#This is june and july combined

select c.\*,sum(o.quantity \* p.price) as spent

from

customer c

inner join

(

select \* from orders\_28

where order\_date between '2020-06-01' and '2020-07-31'

) as o

on

c.customer\_id = o.customer\_id

left join

products\_28 p

on o.product\_id = p.product\_id

group by c.customer\_id

having sum(o.quantity \* p.price) >100;

#This is june and july separate(CORRECT)

#step 1

SELECT o.\*,p.\*

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

#step 2

SELECT o.customer\_id, EXTRACT(MONTH FROM order\_date),o.quantity,p.price

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

#step 3

SELECT o.customer\_id, EXTRACT(MONTH FROM order\_date)

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

GROUP BY customer\_id, EXTRACT(MONTH FROM order\_date);

#step 4

SELECT customer\_id, EXTRACT(MONTH FROM order\_date) AS month, SUM(quantity \* price) AS spent

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

GROUP BY customer\_id, EXTRACT(MONTH FROM order\_date)

HAVING SUM(quantity \* price) >= 100;

#step 5

SELECT c.customer\_id, c.name

FROM customer c

JOIN (

SELECT customer\_id, EXTRACT(MONTH FROM order\_date) AS month, SUM(quantity \* price) AS spent

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

GROUP BY customer\_id, EXTRACT(MONTH FROM order\_date)

HAVING SUM(quantity \* price) >= 100

) as monthly\_spending ON c.customer\_id = monthly\_spending.customer\_id

#step 6

SELECT c.customer\_id, c.name

FROM customer c

JOIN (

SELECT customer\_id, EXTRACT(MONTH FROM order\_date) AS month, SUM(quantity \* price) AS spent

FROM orders\_28 o

JOIN products\_28 p ON o.product\_id = p.product\_id

WHERE order\_date BETWEEN '2020-06-01' AND '2020-07-31'

GROUP BY customer\_id, EXTRACT(MONTH FROM order\_date)

HAVING SUM(quantity \* price) >= 100

) as monthly\_spending ON c.customer\_id = monthly\_spending.customer\_id

GROUP BY c.customer\_id, c.name

HAVING COUNT(DISTINCT monthly\_spending.month) = 2;

#29

create table tvprogram (

program\_date date,

content\_id int,

channel varchar(255),

primary key (program\_date,content\_id)

);

create table content (

content\_id varchar(255) primary key,

title varchar(255),

kids\_content enum('Y','N'),

content\_type varchar(255)

);

INSERT INTO tvprogram (program\_date, content\_id, channel) VALUES

('2020-06-10 08:00', 1, 'LC-Channel'),

('2020-05-11 12:00', 2, 'LC-Channel'),

('2020-05-12 12:00', 3, 'LC-Channel'),

('2020-05-13 14:00', 4, 'Disney Ch'),

('2020-06-18 14:00', 4, 'Disney Ch'),

('2020-07-15 16:00', 5, 'Disney Ch');

INSERT INTO content (content\_id, title, Kids\_content, content\_type) VALUES

('1', 'Leetcode Movie', 'N', 'Movies'),

('2', 'Alg. for Kids', 'Y', 'Series'),

('3', 'Database Sols', 'N', 'Series'),

('4', 'Aladdin', 'Y', 'Movies'),

('5', 'Cinderella', 'Y', 'Movies');

select \* from tvprogram;

select \* from content;

select c.title

from

content c

inner join

(

select \* from tvprogram where extract(month from program\_date) = 6

)as junetv

on

c.content\_id = junetv.content\_id

where c.title='Aladdin';

#30

create table npv (

id int,

year int,

npv int,

primary key (id,year)

);

create table queries (

id int,

year int,

primary key (id,year)

);

INSERT INTO npv (id, year, npv) VALUES

(1, 2018, 100),

(7, 2020, 30),

(13, 2019, 40),

(1, 2019, 113),

(2, 2008, 121),

(3, 2009, 12),

(11, 2020, 99),

(7, 2019, 0);

INSERT INTO queries (id, year) VALUES

(1, 2019),

(2, 2008),

(3, 2009),

(7, 2018),

(7, 2019),

(7, 2020),

(13, 2019);

select \* from npv;

select \* from queries;

SELECT q.\*, COALESCE(n.npv, 0) AS npv

FROM queries q

LEFT JOIN npv n

ON q.id = n.id AND q.year = n.year;

#31 is same as 30

#32

create table employees\_32 (

id int primary key,

name varchar(255)

);

create table employeeUNI (

id int,

unique\_id int,

primary key (id,unique\_id)

);

INSERT INTO employees\_32 (id, name) VALUES

(1, 'Alice'),

(7, 'Bob'),

(11, 'Meir'),

(90, 'Winston'),

(3, 'Jonathan');

INSERT INTO employeeUNI (id, unique\_id) VALUES

(3, 1),

(11, 2),

(90, 3);

select euni.unique\_id,e32.name

from

employees\_32 e32

left join

employeeUNI euni

on

e32.id =euni.id

#33

create table users (

id int primary key,

name varchar(255)

);

create table rides (

id int primary key,

user\_id int,

distance int

);

INSERT INTO users (id, name) VALUES

(1, 'Alice'),

(2, 'Bob'),

(3, 'Alex'),

(4, 'Donald'),

(7, 'Lee'),

(13, 'Jonathan'),

(19, 'Elvis');

INSERT INTO rides (id, user\_id, distance) VALUES

(1, 1, 120),

(2, 2, 317),

(3, 3, 222),

(4, 7, 100),

(5, 13, 312),

(6, 19, 50),

(7, 7, 120),

(8, 19, 400),

(9, 7, 230);

select \* from users;

select \* from rides;

SELECT u.id,u.name,coalesce(sum(r.distance),0)

FROM

users u

left join

rides r

on u.id = r.user\_id

group by u.id

order by sum(r.distance) desc ;

#34 is same as 26

#35

CREATE TABLE movies\_35 (

movie\_id int PRIMARY KEY,

title varchar(255)

);

CREATE TABLE users\_35 (

user\_id int PRIMARY KEY,

name varchar(255)

);

CREATE TABLE movierating\_35 (

movie\_id int,

user\_id int,

rating int,

created\_at date,

PRIMARY KEY (movie\_id, user\_id)

);

INSERT INTO movies\_35 (movie\_id, title) VALUES

(1, 'Avengers'),

(2, 'Frozen 2'),

(3, 'Joker');

INSERT INTO users\_35 (user\_id, name) VALUES

(1, 'Daniel'),

(2, 'Monica'),

(3, 'Maria'),

(4, 'James');

INSERT INTO movierating\_35 (movie\_id, user\_id, rating, created\_at) VALUES

(1, 1, 3, '2020-01-12'),

(1, 2, 4, '2020-02-11'),

(1, 3, 2, '2020-02-12'),

(1, 4, 1, '2020-01-01'),

(2, 1, 5, '2020-02-17'),

(2, 2, 2, '2020-02-01'),

(2, 3, 2, '2020-03-01'),

(3, 1, 3, '2020-02-22'),

(3, 2, 4, '2020-02-25');

select \* from movies\_35;

select \* from users\_35;

select \* from movierating\_35;

#part 1

select u35.name

from

users\_35 u35

inner join

(

select user\_id,count(\*) as num\_ratings

from

movierating\_35

group by user\_id

order by num\_ratings desc

limit 2

) as grrating

on u35.user\_id = grrating.user\_id

order by length(u35.name)

limit 1

#part 2

select m35.title

from

movies\_35 m35

inner join

(

select movie\_id , avg(rating) as ar

from

movierating\_35

where created\_at between '2020-02-01' and '2020-02-29'

group by movie\_id

order by ar desc

limit 2

) as mrating\_35

on m35.movie\_id = mrating\_35.movie\_id

order by title ASC

limit 1

select movie\_id , avg(rating) as ar

from

movierating\_35

where created\_at between '2020-02-01' and '2020-02-29'

group by movie\_id

order by ar desc

#cross check 35 once

#36 is same as 33

#37 is same as 32

#38

create table departments (

id int primary key,

name varchar(255)

);

create table students (

id int primary key,

name varchar(255),

department\_id INT

);

INSERT INTO departments (id, name) VALUES

(1, 'Electrical Engineering'),

(7, 'Computer Engineering'),

(13, 'Business Administration');

INSERT INTO students (id, name, department\_id) VALUES

(23, 'Alice', 1),

(1, 'Bob', 7),

(5, 'Jennifer', 13),

(2, 'John', 14),

(4, 'Jasmine', 77),

(3, 'Steve', 74),

(6, 'Luis', 1),

(8, 'Jonathan', 7),

(7, 'Daiana', 33),

(11, 'Madelynn', 1);

select \* from departments;

select \* from students;

select id,name from students

where students.department\_id not in

(

select id from departments

);

#39  
  
CREATE TABLE calls (

from\_id int,

to\_id int,

duration int

);

INSERT INTO calls (from\_id, to\_id, duration) VALUES

(1, 2, 59),

(2, 1, 11),

(1, 3, 20),

(3, 4, 100),

(3, 4, 200),

(3, 4, 200),

(4, 3, 499);

select \* from calls

select

least(c.from\_id,c.to\_id) as person1,

greatest(c.from\_id,c.to\_id) as person2,

count(\*),

sum(duration)

from

calls c

group by person1,person2

#40

CREATE TABLE prices\_40 (

product\_id int,

start\_date date,

end\_date date,

price int,

PRIMARY KEY (product\_id, start\_date, end\_date)

);

CREATE TABLE unitssold\_40 (

product\_id int,

purchase\_date date,

units int

);

INSERT INTO prices\_40 (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 INTO unitssold\_40 (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);

select \* from prices\_40

select \* from unitssold\_40

select p40.product\_id,sum(p40.price\*u40.units)/sum(u40.units)

from

unitssold\_40 u40

inner join

prices\_40 p40

on p40.product\_id = u40.product\_id

where u40.purchase\_date between p40.start\_date and p40.end\_date

group by p40.product\_id

#watch Q40 again

#A spl question  
  
Question  
<https://leetcode.com/problems/customers-who-bought-all-products/>  
  
Solution  
WITH ProductCount AS (

SELECT COUNT(\*) AS total\_products

FROM Product

),

CustomerProductCount AS (

SELECT customer\_id, COUNT(DISTINCT product\_key) AS product\_count

FROM Customer

GROUP BY customer\_id

)

select cpc.customer\_id

from

CustomerProductCount cpc

join

ProductCount pc

on

cpc.product\_count = pc.total\_products

#41

CREATE TABLE warehouse\_41 (

name VARCHAR(255),

product\_id INT,

units INT,

PRIMARY KEY (name, product\_id)

);

CREATE TABLE products\_41 (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(255),

Width INT,

Length INT,

Height INT

);

INSERT INTO warehouse\_41 (name, product\_id, units) VALUES

('LCHouse1', 1, 1),

('LCHouse1', 2, 10),

('LCHouse1', 3, 5),

('LCHouse2', 1, 2),

('LCHouse2', 2, 2),

('LCHouse3', 4, 1);

INSERT INTO products\_41 (product\_id, product\_name, Width, Length, Height) VALUES

(1, 'LC-TV', 5, 50, 40),

(2, 'LC-KeyChain', 5, 5, 5),

(3, 'LC-Phone', 2, 10, 10),

(4, 'LC-T-Shirt', 4, 10, 20);

select w.name,sum((u.width\*u.length\*u.height)\*w.units)

from

warehouse\_41 w

join

products\_41 u

on

w.product\_id = u.product\_id

group by w.name

#42

create table sales (

sale\_date date,

fruit enum('apples','oranges') not null,

sold\_num int,

primary key (sale\_date,fruit)

);

INSERT INTO sales (sale\_date, fruit, sold\_num) VALUES

('2020-05-01', 'apples', 10),

('2020-05-01', 'oranges', 8),

('2020-05-02', 'apples', 15),

('2020-05-02', 'oranges', 15),

('2020-05-03', 'apples', 20),

('2020-05-03', 'oranges', 0),

('2020-05-04', 'apples', 15),

('2020-05-04', 'oranges', 16);

with applecount as (

select sale\_date,sum(sold\_num) as appsold

from sales

where fruit = 'apples'

group by sale\_date

),

orangecount as (

select sale\_date,sum(sold\_num) as orsold

from sales

where fruit = 'oranges'

group by sale\_date

)

select ac.sale\_date, appsold-orsold

from

applecount ac

join

orangecount oc

on

ac.sale\_date = oc.sale\_date

#43

CREATE TABLE activity\_43 (

player\_id INT,

device\_id INT,

event\_date DATE,

games\_played INT,

PRIMARY KEY (player\_id, event\_date)

);

INSERT INTO activity\_43 (player\_id, device\_id, event\_date, games\_played) VALUES

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

(1, 2, '2016-03-02', 6),

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

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

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

with firstlogin as (

select player\_id,min(event\_date) as firstlogin

from activity\_43

group by player\_id

),

consecutivelogin as (

select f.player\_id,min(event\_date)

from firstlogin f

inner join

activity\_43 a

on f.player\_id = a.player\_id and a.event\_date = DATE\_ADD(f.firstlogin,INTERVAL 1 DAY)

group by f.player\_id

),

totalplayers as (

select count(distinct player\_id) as tp

from

activity\_43

),

consecutiveloginplayers as (

select count(distinct player\_id) as cp

from

consecutivelogin

)

select round(cp\*1/tp,2)

from

totalplayers,consecutiveloginplayers

#44

CREATE TABLE Employee (

id INT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

department VARCHAR(255) NOT NULL,

managerId INT,

FOREIGN KEY (managerId) REFERENCES Employee(id),

CHECK (id <> managerId)

);

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);

select \* from Employee

with numreportees as (

select managerId,count(\*) as reportees

from

Employee

group by managerId

)

select e.name

from

Employee e

join

numreportees nr

on

e.id = nr.managerId

WHERE nr.reportees >= 5;

#45

create table department (

dept\_id int primary key,

dept\_name varchar(255)

)

CREATE TABLE student (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(255) NOT NULL,

gender VARCHAR(10) NOT NULL,

dept\_id INT,

foreign key (dept\_id) references department(dept\_id)

);

INSERT INTO Department (dept\_id, dept\_name) VALUES

(1, 'Engineering'),

(2, 'Science'),

(3, 'Law');

INSERT INTO student (student\_id, student\_name, gender, dept\_id) VALUES

(1, 'Jack', 'M', 1),

(2, 'Jane', 'F', 1),

(3, 'Mark', 'M', 2);

select \* from student;

select \* from Department;

select d.dept\_name,count(s.student\_id)

from

student s

right join

Department d

on s.dept\_id =d.dept\_id

group by d.dept\_name

#46

create table product\_46 (

product\_key int primary key

)

create table customer\_46 (

customer\_id int,

product\_key int,

foreign key (product\_key) references product\_46(product\_key)

);

INSERT INTO product\_46 (product\_key) VALUES

(5),

(6);

INSERT INTO customer\_46 (customer\_id, product\_key) VALUES

(1, 5),

(2, 6),

(3, 5),

(3, 6),

(1, 6);

select \* from product\_46

select \* from customer\_46

with p as (

select count(product\_key) as prod\_count from product\_46

),

c as (

select customer\_id,count(product\_key) as cbought

from

customer\_46

group by customer\_id

)

select c.customer\_id

from

p

inner join

c

on

p.prod\_count = c.cbought

select customer\_id,count(product\_key) as cbought

from

customer\_46

group by customer\_id

#47

CREATE TABLE employee\_47 (

employee\_id INT PRIMARY KEY,

name VARCHAR(255),

experience\_years INT

);

CREATE TABLE project\_47 (

project\_id INT,

employee\_id INT,

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

FOREIGN KEY (employee\_id) REFERENCES employee\_47(employee\_id)

);

INSERT INTO employee\_47 (employee\_id, name, experience\_years) VALUES

(1, 'Khaled', 3),

(2, 'Ali', 2),

(3, 'John', 3),

(4, 'Doe', 2);

INSERT INTO project\_47 (project\_id, employee\_id) VALUES

(1, 1),

(1, 2),

(1, 3),

(2, 1),

(2, 4);

select \* from employee\_47

select \* from project\_47

with rankedmembers\_s1 as (

select p.project\_id,e.\*

from

project\_47 p

inner join

employee\_47 e

on

p.employee\_id = e.employee\_id

),

mostop as (

select project\_id,employee\_id,

dense\_rank() over (partition by project\_id order by experience\_years desc) as most\_op\_id

from

rankedmembers\_s1

)

select project\_id,employee\_id from

mostop

where most\_op\_id =1

**#48 is incomplete question**

#49

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);

select \* from enrollments;

with gradesfilter as(

select

student\_id,course\_id,grade,

row\_number() over(partition by student\_id order by grade desc,course\_id asc) as r1

from

enrollments

)

select student\_id,course\_id,grade

from gradesfilter

where r1 = 1

order by

student\_id asc

**#50 is a incomplete question**