**create database SQL\_Assignment;**

**use SQL\_Assignment;**

--- Q1

**Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.**

create table city

(

ID int unique,

NAME varchar(17),

COUNTRYCODE varchar(3),

DISTRICT varchar (20),

POPULATION int

);

insert into city values (6,'Rotterdam','NLD','Zuid-Holland',593321),

(3878,'Scottsdale','USA','Arizona',202705),

(3965,'Corona','USA','California',124966),

(3973,'Concord','USA','California',121780),

(3977,'Cedar Rapids','USA','Iowa',120758),

(3982,'Coral Springs','USA','Florida',117549),

(4054,'Fairfield','USA','California',92256),

(4058,'Boulder','USA','Colorado',91238),

(4061,'Fall River','USA','Massachusetts',90555);

select \*

from city

where COUNTRYCODE = 'USA' and POPULATION > 100000;

--- Q2

**Query the NAME field for all American cities in the CITY table with populations larger than 120000.**

select NAME as USA\_cities

from city

where COUNTRYCODE = 'USA' and POPULATION > 120000;

--- Q3

**Query all columns (attributes) for every row in the CITY table.**

select \*

from city;

--- Q4

**Query all columns for a city in CITY with the ID 1661.**

update city

set ID = 1661

where ID = 6;

select \*

from city

where ID = 1661;

--- Q5

**Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.**

insert into city values (5325,'Htachi','JPN','Ibaraki',560743),

(6525,'Hokkaido','JPN','Chitose',555352),

(6542,'Fukui','JPN','Takefu',481493),

(8452,'Mie','JPN','Suzuka',470277);

select \*

from city

where COUNTRYCODE = 'JPN';

--- Q6

**Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.**

select NAME as Japanese\_city

from city

where COUNTRYCODE = 'JPN';

--- Q7

**Query a list of CITY and STATE from the STATION table.**

create table STATION

(

ID int,

CITY varchar(21),

STATE varchar(2),

LAT\_N int,

LONG\_W int

);

insert into STATION values (794,'Kissee Mills','MO',139,73),

(824,'Loma Mar','CA',48,130),

(603,'Sandy Hook','CT',72,148),

(478,'Tipton','IN',33,97),

(619,'Arlington','CO',75,92),

(711,'Turner','AR',50,101),

(839,'Slidell','LA',85,151),

(411,'Negreet','LA',98,105),

(588,'Glencoe','KY',46,136),

(665,'Chelsea','IA',98,59),

(342,'Chignik Lagoon','AK',103,153),

(733,'Pelahatchie','MS',38,28),

(441,'Hanna Cit','IL',50,136),

(811,'Dorranc','KS',102,121),

(698,'Albany','CA',49,80),

(325,'Monument','KS',70,141),

(414,'Manchester','MD',73,37),

(113,'Prescott','IA',9,65),

(971,'Graettinger','IA',94,150),

(266,'Cahone','CO',116,127);

select CITY, STATE

from STATION;

--- Q8

**Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.**

select distinct CITY

from STATION

where mod(ID,2)=0;

--- Q9

**Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.**

select count(CITY) - count(distinct city)as diff\_btw\_city\_dist\_city

from STATION;

--- Q10

**Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.**

select CITY as Longest\_city\_name,

length(city) as length

from station order by length(City) desc, city limit 1;

select CITY as Shortest\_city\_name,

length(city) as length

from station order by length(City), city limit 1;

--- Q11

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

SELECT DISTINCT city

FROM station

where city like 'a%' or city like 'e%' or city like 'i%' or city like 'o%' or city like 'u%';

--- Q12

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

select distinct city

from station

where city like '%a' or city like '%e' or city like '%i' or city like '%o' or city like '%u';

--- Q13

**Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates**

select distinct city

from station

where city not like 'a%' and city not like 'e%' and city not like 'i%' and city not like 'o%' and city not like 'u%';

--- Q14

**Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.**

select distinct city

from station

where city not like '%a' and city not like '%e' and city not like '%i' and city not like '%o' and city not like '%u';

--- Q15

**Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.**

select distinct city

from station

where city like 'a%a' or city like 'a%e' or city like 'a%i' or city like 'a%o' or city like 'a%u' or

city like 'e%a' or city like 'e%e' or city like 'e%i' or city like 'e%o' or city like 'e%u' or

city like 'i%a' or city like 'i%e' or city like 'i%i' or city like 'i%o' or city like 'i%u' or

city like 'o%a' or city like 'o%e' or city like 'o%i' or city like 'o%o' or city like 'o%u' or

city like 'u%a' or city like 'u%e' or city like 'u%i' or city like 'u%o' or city like 'u%u';

--- Q16

**Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.**

select distinct city

from station

where city not like 'a%' and city not like 'e%' and city not like 'i%' and city not like 'o%' and city not like 'u%' and

city not like '%a' and city not like '%e' and city not like '%i' and city not like '%o' and city not like '%u';

--- Q17

**Write an SQL query that reports the products that were only sold in the first quarter of 2019. That is, between 2019-01-01 and 2019-03-31 inclusive.**

create table Product

(

product\_id int,

product\_name varchar(50),

unit\_price int,

primary key (product\_id)

);

create table Sales

(

seller\_id int,

product\_id int,

buyer\_id int,

sale\_date date,

quantity int,

price int

);

insert into Product values (1,'S8',1000),

(2,'G4',800),

(3,'iPhone',1400);

insert into Sales 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 p.product\_id, p.product\_name

from product as p inner join sales as s on p.product\_id = s.product\_id

where s.sale\_date not between '2019-03-31' and '2019-12-31' and

s.sale\_date between '2019-01-01' and '2019-03-31';

--- Q18

**Write an SQL query to find all the authors that viewed at least one of their own articles. Return the result table sorted by id in ascending order**

create table Views

(

article\_id int,

author\_id int,

viewer\_id int,

view\_date date

);

insert into views 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 as Id

from views

where author\_id = viewer\_id

order by author\_id;

--- Q19

**Write an SQL query to find the percentage of immediate orders in the table, rounded to 2 decimal places.**

create table Delivery

(

delivery\_id int,

customer\_id int,

order\_date date,

customer\_pref\_delivery\_date date

);

insert into delivery 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 count(order\_date)

from delivery

where order\_date = customer\_pref\_delivery\_date;

select (2/ count(order\_date) \* 100) as immediate\_percentage

from delivery;

--- Q20

**Write an SQL query to find the ctr of each Ad. Round ctr to two decimal points. Return the result table ordered by ctr in descending order and by ad\_id in ascending order in case of a tie.**

create table Ads

(

ad\_id int,

user\_id int,

action enum('Clicked', 'Viewed', 'Ignored'),

primary key (ad\_id,user\_id)

);

insert into Ads 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 distinct ad\_id, (count(action = 'clicked')/(count(action = 'Clicked') + count(action = 'Viewed')) \* 100,2) as ctr

from Ads

group by ad\_id;

select ad\_id,

ifnull(

round(

avg(

case

when action = 'Clicked' then 1

when action = 'viewed' then 0

else null

end

) \* 100,

2),

0)

as ctr

from Ads

group by ad\_id;

--- Q21

**Write an SQL query to find the team size of each of the employees.**

create table if not exists Employee

(

employee\_id int,

team\_id int,

primary key (employee\_id)

);

insert into Employee values (1, 8),

(2, 8),

(3, 8),

(4, 7),

(5, 9),

(6, 9);

select e1.employee\_id,

count(\*) as team\_size

from employee e1 left join employee e2 on e1.team\_id = e2.team\_id

group by e1.employee\_id;

--- Q22

**Write an SQL query to find the type of weather in each country for November 2019. The type of weather is: ● Cold if the average weather\_state is less than or equal 15, ● Hot if the average weather\_state is greater than or equal to 25, and ● Warm otherwise.**

create table countries

(

country\_id int,

country\_name varchar(30),

primary key(country\_id)

);

create table weather

(

country\_id int,

weather\_state int,

day date,

primary key(country\_id, day)

);

insert into countries values (2, 'USA'),

(3, 'Australia'),

(7, 'Peru'),

(5, 'China'),

(8, 'Morocco'),

(9, 'Spain');

insert into weather 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 distinct c.country\_name,

case

when w.weather\_state <= 15 then 'Cold'

when w.weather\_state >= 25 then 'Hot'

else 'Warm'

end as weather\_type

from countries as c inner join weather as w on c.country\_id = w.country\_id

where w.day between '2019-11-01' and '2019-11-30';

--- Q23

**Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places.**

create table Prices

(

product\_id int,

start\_date date,

end\_date date,

price int,

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

);

create table UnitsSold

(

product\_id int,

purchase\_date date,

units int

);

insert into Prices 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 values (1, '2019-02-25', 100),

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

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

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

select product\_id,

ifnull

(round

(sum(prices\_sum)/ sum(units),

2),

0) as average\_price

from (select p.product\_id as product\_id, units,

price \* units as prices\_sum

from Prices p inner join UnitsSold u on p.product\_id = u.product\_id and

purchase\_date between start\_date and end\_date) as temp

group by product\_id;

--- Q24

**Write an SQL query to report the first login date for each player.**

create table Activity

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

primary key(player\_id, event\_date)

);

insert into Activity 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 player\_id, min(event\_date) as first\_login

from activity

group by player\_id

order by first\_login and player\_id;

--- Q25

**Write an SQL query to report the device that is first logged in for each player.**

select player\_id, device\_id

from activity

where (player\_id, event\_date) in (select player\_id, min(event\_date)

from activity

group by player\_id);

--- Q26

**Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount**

create table Products

(

product\_id int,

product\_name varchar(30),

product\_category varchar(30),

primary key(product\_id)

);

create table if not exists Orders

(

product\_id int,

order\_date date,

unit int

);

insert into Products values (1, 'Leetcode Solutions', 'Book'),

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

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

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

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

insert into Orders 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 product\_name,

unit

from Products inner join

(select product\_id, sum(unit) as unit

from orders

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

group by product\_id

having sum(unit) >= 100) as temp

on products.product\_id = temp.product\_id;

--- Q27

**Write an SQL query to find the users who have valid emails. A valid e-mail has a prefix name and a domain where: ● The prefix name is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'. The prefix name must start with a letter. ● The domain is '@leetcode.com'.**

create table if not exists Users

(

user\_id int,

name varchar(30),

mail varchar(50),

primary key(user\_id)

);

insert into users 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

where mail regexp '^[a-zA-Z]+[a-zA-Z0-9\_\\./\\-]{0,}@leetcode.com$'

order by user\_id;

--- Q28

**Write an SQL query to report the customer\_id and customer\_name of customers who have spent at least $100 in each month of June and July 2020.**

create table if not exists Customers

(

customer\_id int,

name varchar(30),

country varchar(30),

primary key(customer\_id)

);

create table if not exists Product\_1

(

product\_id int,

description varchar(30),

price int,

primary key(product\_id)

);

create table if not exists orders\_1

(

order\_id int,

customer\_id int,

product\_id int,

order\_date date,

quantity int

);

insert into customers values (1, 'Winston', 'USA'),

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

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

insert into product\_1 values (10, 'LC Phone', 300),

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

(30, 'LC Book', 45),

(40, 'LC Keychain', 2);

insert into orders\_1 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 c.customer\_id,

c.name

from customers c inner join orders\_1 o on c.customer\_id = o.customer\_id

inner join product\_1 p on o.product\_id = p.product\_id

group by c.customer\_id, c.name

having sum(case

when left (o.order\_date,7) = '2020-06' then p.price \* o.quantity

else 0

end) >= 100

and

sum(case

when left (o.order\_date,7) = '2020-07' then p.price \* o.quantity

else 0

end) >= 100;

--- Q29

**Write an SQL query to report the distinct titles of the kid-friendly movies streamed in June 2020. Return the result table in any order.**

create table TVProgram

(

program\_date datetime,

content\_id int,

channel varchar(30),

primary key (program\_date, content\_id)

);

create table Content

(

content\_id int,

title varchar(30),

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

content\_type varchar(30),

primary key (content\_id)

);

insert into TVProgram 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 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 distinct c.title

from TVProgram t inner join content c on t.content\_id = c.content\_id

where t.program\_date between '2020-06-01 00:00' and '2020-06-30 23:59' and c.Kids\_content = 'Y';

--- Q30

**Write an SQL query to find the npv of each query of the Queries table.**

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 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 values (1, 2019),

(2, 2008),

(3, 2009),

(7, 2018),

(7, 2019),

(7, 2020),

(13, 2019);

select id, year, ifnull(NPV,0)

from(

select q.id as id, q.year as year, n.NPV as NPV

from queries q left join NPV n on q.id = n.id and q.year = n.year)as tmp;

--- Q31

--- same as question no. 30

--- Q32

**Write an SQL query to show the unique ID of each user, If a user does not have a unique ID replace just show null.**

create table if not exists employees

(

id int,

name varchar(50),

primary key(id)

);

create table EmployeeUNI

(

id int,

unique\_id int,

primary key(id, unique\_id)

);

insert into employees values (1, 'Alice'),

(7, 'Bob'),

(11, 'Meir'),

(90, 'Winston'),

(3, 'Jonathan');

insert into EmployeeUNI values (3, 1),

(11, 2),

(90, 3);

select u.unique\_id as unique\_id, e.name as name

from employees e left join EmployeeUNI u on e.id = u.id;

--- Q33

**Write an SQL query to report the distance travelled by each user. Return the result table ordered by travelled\_distance in descending order, if two or more users travelled the same distance, order them by their name in ascending order.**

create table if not exists users\_1

(

id int,

name varchar(50),

primary key(id)

);

create table if not exists Rides

(

id int,

user\_id int,

distance int,

primary key(id)

);

insert into users\_1 values (1, 'Alice'),

(2, 'Bob'),

(3, 'Alex'),

(4, 'Donald'),

(7, 'Lee'),

(13, 'Jonathan'),

(19, 'Elvis');

insert into rides 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 name,

distance

from

(select u.name as name, ifnull(sum(r.distance),0) as distance from users\_1 u left join rides r on u.id = r.user\_id

group by u.name

order by name) as temp

order by distance desc;

--- Q34

--- same as question no 26

--- Q35

**Write an SQL query to: ● Find the name of the user who has rated the greatest number of movies. In case of a tie, return the lexicographically smaller user name. ● Find the movie name with the highest average rating in February 2020. In case of a tie, return the lexicographically smaller movie name.**

create table Movies

(

movie\_id int,

title varchar(50),

primary key(movie\_id)

);

create table users\_2

(

user\_id int,

name varchar(50),

primary key(user\_id)

);

create table MovieRating

(

movie\_id int,

user\_id int,

rating int,

created\_at date,

primary key(movie\_id, user\_id)

);

insert into Movies values (1, 'Avengers'),

(2, 'Frozen 2'),

(3, 'Joker');

insert into users\_2 values (1, 'Daniel'),

(2, 'Monica'),

(3, 'Maria'),

(4, 'James');

insert into MovieRating 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 user\_name as results

from

(select u.name as user\_name, count(r.rating) as total\_rating

from users\_2 u inner join MovieRating r on u.user\_id = r.user\_id

group by r.user\_id

order by total\_rating desc, user\_name asc limit 1)first\_temp

union

select movie\_name as results

from

(select m.movie\_id, m.title as movie\_name, avg(r.rating) as rating

from movies m inner join MovieRating r on m.movie\_id = r.movie\_id

where left(r.created\_at,7) = '2020-02'

group by m.movie\_id

order by rating desc, movie\_name asc limit 1 )second\_temp;

--- Q36

--- same as question no 33

--- Q37

--- same as question no 32

--- Q38

**Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.**

create table Departments

(

id int,

name varchar(50),

primary key(id)

);

create table Students

(

id int,

name varchar(50),

department\_id int,

primary key(id)

);

insert into departments values (1, 'Electrical Engineering'),

(7, 'Computer Engineering'),

(13, 'Business Administration');

insert into students 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 id, name

from students

where department\_id not in (

select id

from departments

);

--- Q39

**Write an SQL query to report the number of calls and the total call duration between each pair of distinct persons (person1, person2) where person1 < person2.**

create table if not exists Calls

(

from\_id int,

to\_id int,

duration int

);

insert into calls values (1, 2, 59),

(2, 1, 11),

(1, 3, 20),

(3, 4, 100),

(3, 4, 200),

(3, 4, 200),

(4, 3, 499);

with

both\_caller as (

select from\_id as person1, to\_id as person2, duration

from calls

UNION ALL

select to\_id as person1, from\_id as person2, duration

from calls

),

caller as (

select person1, person2, duration

from both\_caller

where person1 < person2

)

select person1, person2, count(\*) as call\_count, sum(duration)as total\_duration

from caller

group by person1, person2;

--- &

select

case

when from\_id < to\_id then from\_id

else to\_id

end as person1,

case

when from\_id < to\_id then to\_id

else from\_id

end as person2,

count(\*) as call\_count,

sum(duration) as total\_duration

from calls

group by person1, person2;

--- Q40

--- same as question no 23

--- Q41

**Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.**

create table warehouse

(

name varchar(30),

product\_id int,

units int,

primary key(name, product\_id)

);

create table products\_1

(

product\_id int,

product\_name varchar(50),

Width int,

Length int,

Height int,

primary key(product\_id)

);

insert into warehouse values ('LCHouse1', 1, 1),

('LCHouse1', 2, 10),

('LCHouse1', 3, 5),

('LCHouse2', 1, 2),

('LCHouse2', 2, 2),

('LCHouse3', 4, 1);

insert into products\_1 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 name as warehouse\_name,

sum((Width \* Length \* Height)\*units) as volume

from warehouse w inner join products\_1 p on w.product\_id = p.product\_id

group by name;

--- Q42

**Write an SQL query to report the difference between the number of apples and oranges sold each day. Return the result table ordered by sale\_date**

create table sales\_1

(

sale\_date date,

fruit enum ('apples', 'oranges'),

sold\_num int,

primary key (sale\_date, fruit)

);

insert into sales\_1 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);

select sale\_date,

sum(if(fruit = 'apples', sold\_num, -sold\_num)) as diff

from sales\_1

group by sale\_date

order by sale\_date;

--- Q43

**Write an SQL query to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.**

create table activity\_1

(

player\_id int,

device\_id int,

event\_date date,

games\_played int,

primary key(player\_id, event\_date)

);

insert into activity\_1 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);

select round(

ifnull(

(

select count(distinct a.player\_id)

from activity\_1 a join activity\_1 b

on a.player\_id = b.player\_id

and datediff(b.event\_date, a.event\_date) = 1

where a.event\_date = (

select min(event\_date)

from activity\_1

where player\_id = a.player\_id

)

)/(

select count(distinct player\_id)

from activity\_1),

0),

2) as fraction;

--- Q44

**Write an SQL query to report the managers with at least five direct reports.**

create table employee\_5

(

id int,

name varchar(50),

department varchar(30),

managerId int,

primary key (id)

);

insert into employee\_5 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 name

from employee\_5 e join (

select managerId

from employee\_5

group by managerId

having count(managerId) >= 5) m

on e.id = m.managerId;

--- Q45

**Write an SQL query to report the respective department name and number of students majoring in each department for all departments in the Department table (even ones with no current students). Return the result table ordered by student\_number in descending order. In case of a tie, order them by dept\_name alphabetically.**

create table if not exists student

(

student\_id int,

student\_name varchar(50),

gender varchar(20),

dept\_id int,

primary key (student\_id)

);

create table department

(

dept\_id int,

dept\_name varchar(30),

primary key(dept\_id)

);

insert into student values (1, 'Jack', 'M', 1),

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

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

insert into department values (1, 'Engineering'),

(2, 'Science'),

(3, 'Law');

select d.dept\_name,

count(s.dept\_id) as student\_number

from department d left join student s on d.dept\_id = s.dept\_id

group by d.dept\_id;

--- Q46

**Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.**

create table customer

(

customer\_id int,

product\_key int

);

create table product\_2

(

product\_key int,

primary key(product\_key)

);

insert into customer values (1, 5),

(2, 6),

(3, 5),

(3, 6),

(1, 6);

insert into product\_2 values (5), (6);

select customer\_id

from(select c.customer\_id, c.product\_key

from customer c inner join product\_2 p on c.product\_key = p.product\_key

group by c.customer\_id

having c.product\_key = 5 and 6)as temp;

--- Q47

**Write an SQL query that reports the most experienced employees in each project. In case of a tie, report all employees with the maximum number of experience years.**

create table project

(

project\_id int,

employee\_id int,

primary key(project\_id, employee\_id)

);

create table employee\_1

(

employee\_id int,

name varchar(50),

experience\_years int,

primary key(employee\_id)

);

insert into project values (1, 1),

(1, 2),

(1, 3),

(2, 1),

(2, 4);

insert into employee\_1 values (1, 'Khaled', 3),

(2, 'Ali', 2),

(3, 'John', 3),

(4, 'Doe', 2);

select p.project\_id,

e.employee\_id

from project p left join employee\_1 e on p.employee\_id = e.employee\_id

where (p.project\_id, e.experience\_years) in

( select p.project\_id, max(e.experience\_years)

from project p left join employee\_1 e on p.employee\_id = e.employee\_id

group by project\_id);

--- Q48

**Write an SQL query that reports the books that have sold less than 10 copies in the last year, excluding books that have been available for less than one month from today. Assume today is 2019-06-23.**

create table if not exists books

(

book\_id int,

name varchar(50),

available\_from date,

primary key(book\_id)

);

create table if not exists orders\_3

(

order\_id int,

book\_id int,

quantity int,

dispatch\_date date,

primary key(order\_id)

);

insert into books values (1, "Kalila And Demna", '2010-01-01'),

(2, "28 Letters", '2012-05-12'),

(3, "The Hobbit", '2019-06-10'),

(4, "13 Reasons Why", '2019-06-01'),

(5, "The Hunger Games", '2008-09-21');

--- Q49

**Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course\_id.**

create table if not exists Enrollments

(

student\_id int,

course\_id int,

grade int,

primary key(student\_id, course\_id)

);

insert into Enrollments values (2, 2, 95),

(2, 3, 95),

(1, 1, 90),

(1, 2, 99),

(3, 1, 80),

(3, 2, 75),

(3, 3, 82);

select student\_id, min(course\_id) as course\_id, grade

from Enrollments

where (student\_id, grade) in

(select student\_id, max(grade)

from Enrollments

group by student\_id)

group by student\_id;

--- Q50

**Write an SQL query to find the winner in each group.**

create table players

(

player\_id int,

group\_id int,

primary key(player\_id)

);

insert into players values (15, 1),

(25, 1),

(30, 1),

(45, 1),

(10, 2),

(35, 2),

(50, 2),

(20, 3),

(40, 3);

create table matches

(

match\_id int,

first\_player int,

second\_player int,

first\_score int,

second\_score int,

primary key (match\_id)

);

insert into matches values (1,15,45,3,0),

(2,30,25,1,2),

(3,30,15,2,0),

(4,40,20,5,2),

(5,35,50,1,1);

select group\_id,

player\_id

from (

select p.group\_id, temp.player\_id, sum(temp.score) as score

from players p,

(

select first\_player as player\_id, first\_score as score

from matches

UNION ALL

select second\_player, second\_score

from matches

) temp

where p.player\_id = temp.player\_id

group by temp.player\_id

order by group\_id, score desc, player\_id

) top\_scores

group by group\_id;