/\* Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.

The CountryCode for America is USA. \*/

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.

The CountryCode for America is USA.\*/

select CNAME AS NAME 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.\*/

select \* from city where CNAME in (select CNAME from city where ID = 1661)

/\*Q5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is

JPN.\*/

select \* from city where cname in (select CNAME 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 CNAME from city where COUNTRYCODE ='JPN';

/\*Q7. Query a list of CITY and STATE from the STATION table.\*/

select distinct 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(1) - count(distinct city) as total\_cities\_minus\_dist\_cities 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.\*/

-- minimum length city

select \*,length(city) as city\_length from station order by length(city),city limit 1

-- maximum length city

select \*,length(city) as city\_length from station order by length(city) desc,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 left(city,1) in('a','e','i','o','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 right(city,1) in('a','e','i','o','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 left(city,1) not in('a','e','i','o','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 right(city,1) not in('a','e','i','o','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 right(city,1) not in('a','e','i','o','u') and left(city,1) not in('a','e','i','o','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 right(city,1) not in('a','e','i','o','u') and left(city,1) not in('a','e','i','o','u');

/\*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.

Return the result table in any order.\*/

select \* from product

where product\_id in (select product\_id from Sales

where date(sale\_date)>= '2019-01-01' and date(sale\_date)<='2019-03-31')

and product\_id not in (select product\_id from sales where sale\_date not between '2019-01-01' and '2019-03-31')

/\*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.\*/

select distinct v1.author\_id

from (select article\_id,author\_id from views) v1

inner join (select article\_id,viewer\_id from views) v2 on v1.article\_id=v2.article\_id and v1.author\_id=v2.viewer\_id

order by v1.author\_id

/\*Q19. If the customer's preferred delivery date is the same as the order date, then the order is called

immediately; otherwise, it is called scheduled.

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

select round(sum(convert(float,case when date(order\_date)=date(customer\_pref\_delivery\_date) then 1 else 0 end))/count(\*)\*100,2) as Percentage\_immediate 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.\*/

select ad\_id,round(sum(convert(float,case when Clicked+Viewed>0 then (Clicked/(Clicked+Viewed))\*100 else 0 end)),2) as ctr from (

select \*, case when action = 'Clicked' then 1 else 0 end as Clicked,case when action = 'Viewed' then 1 else 0 end as Viewed from Ads) as x

group by ad\_id order by 2 desc,1

/\*Q21. Write an SQL query to find the team size of each of the employees.

Return result table in any order.\*/

select employee\_id, count(employee\_id) over(partition by team\_id) as team\_size from employee;

/\*Q3. Write an SQL query to find the average selling price for each product. average\_price should be

rounded to 2 decimal places. Return the result table in any order.\*/

select s.product\_id, round(sum(s.units\*p.price)/sum(s.units),2) as avg\_price

from UnitsSold s

JOIN prices p on p.product\_id = s.product\_id and date(s.purchase\_date) between date(p.start\_date) and date(p.end\_date)

group by s.product\_id

/\*Q24 Write an SQL query to report the first login date for each player.

Return the result table in any order.\*/

select player\_id, min(event\_date) as first\_login\_date

from Activity

group by player\_id;

/\*Q25 .Write an SQL query to report the device that is first logged in for each player.

Return the result table in any order.\*/

select player\_id, device\_id from (

select \*, row\_number() over(partition by player\_id order by event\_date) as first\_event

from Activity) as x where first\_event=1

/\*Q26. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.

Return result table in any order.\*/

select p.product\_name,sum(O.unit) as units

from Orders O

inner join Products P

on P.product\_id = O.product\_id

where year(order\_date)=2020 and month(order\_date)=2

group by p.product\_name

having sum(O.unit)>=100

#note: amount column is not mentioned in the table structure

/\*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'.

Return the result table in any order.\*/

select \*

from users

where mail regexp '[a-z0-9\_.-]@leetcode.com';

-- in progress -- get back

/\*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.

Return the result table in any order.\*/

select customer\_id, name from (

select c.customer\_id, c.name, sum(price\*quantity) as amount

from Orders O

inner join Product P on P.product\_id = O.product\_id

inner join Customers C on C.customer\_id = O.customer\_id

where year(order\_date)=2020 and month(order\_date) in (6,7)

group by c.customer\_id, c.name) as x

where amount >=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.\*/

select distinct c.title

from TVProgram tv

inner join content c on c.content\_id = tv.content\_id

where year(program\_date)=2020 and month(program\_date)=6

and c.kids\_content='Y'

/\*Q30 Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order.\*/

select q.\*, NPV.npv

from Queries q

left join NPV on NPV.id = q.id and NPV.year=q.year

/\*Q31 Write an SQL query to find the npv of each query of the Queries table.

Return the result table in any order.\*/

select q.\*, NPV.npv

from Queries q

left join NPV on NPV.id = q.id and NPV.year=q.year

/\*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.

Return the result table in any order.\*/

select eu.unique\_id, e.name

from Employees E

left join EmployeeUNI eu on eu.id = e.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.\*/

select u.name, sum(distance) as trs\_dis

from Users u

left join Riders r on r.user\_id = u.id

group by u.name

order by 2 desc, 1;

/\*Q34. Write an SQL query to get the names of products that have at least 100 units ordered in February 2020 and their amount.

Return result table in any order.\*/

select p.product\_name, sum(o.unit) as ordered\_units

from products p

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

where year(order\_date)=2020 and month(order\_date)=2

group by p.product\_name

having sum(o.unit)>=100

/\*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.\*/

select u.name, count(mr.movie\_id) as no\_of\_movies\_rated

from MovieRating mr

inner join Users u on u.user\_id = mr.user\_id

group by u.name

order by count(mr\_movie\_id) desc, u.name limit 1

select m.name, avg(mr.rating) as avg\_rating

from MovieRating mr

inner join Movies m on m.movie\_id = mr.movie\_id

where year(created\_at)=2020 and month(created\_at)=2

group by m.name

order avg(mr.rating) desc, m.name limit 1

/\*Q36. 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.\*/

select u.name, sum(r.distance) as travelled\_dis

from Users u

left join Riders r on r.user\_id = u.id

group by u.name

order by 2 desc, 1;

/\*Q37. 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.

Return the result table in any order.\*/

select unique\_id, e.name

from Employees e

left join EmployeeUNI eu on eu.unique\_id = e.id

/\*Q38. Write an SQL query to find the id and the name of all students who are enrolled in departments that no longer exist.

Return the result table in any order.\*/

select s.id, s.name

from Students s

left join Departments d on d.id = s.department\_id

where d.id is null

/\*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.

Return the result table in any order.\*/

select from\_id, to\_id, count(distinct from\_id, to\_id) as call\_count, sum(duration) as total\_duration

from Calls

group by from\_id, to\_id;

/\*Q40. Write an SQL query to find the average selling price for each product. average\_price should be rounded to 2 decimal places.

Return the result table in any order.\*/

select s.product\_id, round(sum(s.units\*p.price)/sum(s.units),2) as avg\_price

from UnitsSold s

JOIN prices p on p.product\_id = s.product\_id and date(s.purchase\_date) between date(p.start\_date) and date(p.end\_date)

group by s.product\_id

/\*Q41. Write an SQL query to report the number of cubic feet of volume the inventory occupies in each warehouse.

Return the result table in any order.\*/

select w.name, sum((p.width\*p.length\*p.height)\*w.unit) as volume

from Warehouse w

inner join Products p on p.product\_id = w.product\_id

group by u.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.\*/

select sale\_date, sum(case when fruit='apples' then sold\_num else 0 end - case when fruit = 'oranges' then sold\_num else 0 end) as diff

from Sales

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.\*/

select round(count(b.player\_id)/count(distinct a.player\_id),2)

from Activity a

left join (select player\_id, min(event\_date) as first\_date from Activity group by player\_id) b

on a.player\_id = b.player\_id and datediff(a.event\_date, b.first\_date)=1

/\*Q44. Write an SQL query to report the managers with at least five direct reports.

Return the result table in any order.\*/

select name from employee

where Id in (

select managerId

from employee

group by managerId

having count(distinct id)>=5);

/\*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.\*/

select dept\_name, count(student\_id) as student\_number

from department d

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

group by dept\_name

order by count(student\_id) desc, dept\_name;

/\*Q46. Write an SQL query to report the customer ids from the Customer table that bought all the products in the Product table.

Return the result table in any order.\*/

select customer\_id

from customer

group by curstomer\_id

having count(distinct product\_key) = (select count(\*) from product);

/\*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.

Return the result table in any order. \*/

select project\_id, employee\_id from (

select p.\*, dense\_rank() over(partition by p.project\_id order by e.experience\_years desc) as drnk

from project p

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

) as x where drnk=1

/\*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.

Return the result table in any order.\*/

select b.name

from books b

left join (select book\_Id,sum(quantity) as quantity

from Orders where dispatch\_date > '2018-06-23'

Group by book\_Id) o on o.book\_id = b.book\_id

where quantity < 10 and available\_from < '2019-05-23'

/\*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.

Return the result table ordered by student\_id in ascending order.\*/

select e1.student\_id, min(e1.course\_id) as course\_id, e1.grade

from Enrollments e1

where e1.grade =

(select max(grade) as max\_grade

from Enrollments e2

where e1.student\_id = e2.student\_id)

group by e1.student\_id

order by e1.student\_id

/\*Q50. The winner in each group is the player who scored the maximum total points within the group. In the case of a tie, the lowest player\_id wins.\*/

/\*Q51. Write an SQL query to report the name, population, and area of the big countries.

Return the result table in any order.\*/

select name, population, area

from world

where (area >= 3000000 or population >= 25000000);

/\*Q52. Write an SQL query to report the names of the customer that are not referred by the customer with id = 2.\*/

select name

from Customer

where referee\_id <> 2;

/\*Q53. Write an SQL query to report all customers who never order anything.

Return the result table in any order.\*/

select c.name

from Customer c

left join Orders O on O.customerId = c.id

where O.id is null;

/\*Q54. Write an SQL query to find the team size of each of the employees.

Return result table in any order.\*/

select employee\_id, count(\*) over(partition by team\_id) as team\_size

from Employee;

/\*Q55. A telecommunications company wants to invest in new countries. The company intends to invest in

the countries where the average call duration of the calls in this country is strictly greater than the

global average call duration.

Write an SQL query to find the countries where this company can invest.

Return the result table in any order.\*/

select c.name from (

select c.\*, avg(ca.duration) over(partition by p.id) as per\_avg, avg(duration) over() glob\_avg

from person p

inner join country c on substring(p.phone\_number,1,3)=c.country\_code

inner join calls ca on p.id in (c.caller\_id, c.callee\_id)

) as x where per\_avg > glob\_avg;

/\*Q56. Write an SQL query to report the device that is first logged in for each player.

Return the result table in any order.\*/

select player\_id, device\_id from (

select \*, row\_number() over(partition by player\_id order by event\_date) as first\_event

from Activity) as x where first\_event=1

/\*Q57. Write an SQL query to find the customer\_number for the customer who has placed the largest number of orders.

The test cases are generated so that exactly one customer will have placed more orders than any other customer.\*/

select customer\_number

from orders

group by customer\_number

order by count(\*) desc

limit 1;

select customer\_number from (

select customer\_number, dense\_rank() over(partition by customer\_number order by customer\_number) as drnk

from orders ) as x where drnk=1;

/\*Q58. Write an SQL query to report all the consecutive available seats in the cinema.

Return the result table ordered by seat\_id in ascending order.\*/

select a.seat\_id

from cinema x

inner join cinema y on (x.seat\_id - y.seat\_id)=1 and x.free=1 and y.free=1

order by a.seat\_id;

/\*Q59. Write an SQL query to report the names of all the salespersons who did not have any orders related to the company with the name "RED".

Return the result table in any order.\*/

select s.name

from salesperson s

left join (select sales\_id from orders o

join company c on c.com\_id = s.com\_id and c.name = 'RED') excl on excl.sales\_id = s.sales\_id

where excl.sales\_id is null;

/\*Q60 Write an SQL query to report for every three line segments whether they can form a triangle.

Return the result table in any order.\*/

select \*, case when x+y>z and y+z>x and z+x>y then 'Yes' else 'No' end as triangle\_y\_n

from Triangle

/\*Q61. Write an SQL query to report the shortest distance between any two points from the Point table.\*/

select min(abs(b.x-a.b)) shortest\_distance

from Point a

inner join Point b on a.x!=b.x;

/\*Follow up: How could you optimise your query if the Point table is ordered in ascending order?\*/

select min(minval) shortest\_dis from (

select (lead(x,1) over(order by x)-x) as minval

from point) as x;

/\*Q62. Write a SQL query for a report that provides the pairs (actor\_id, director\_id) where the actor has

cooperated with the director at least three times.\*/

select actor\_id, director\_id

from ActorDirector

group by actor\_id, director\_id

having count(\*)> 2;

/\*Q63. Write an SQL query that reports the product\_name, year, and price for each sale\_id in the Sales table.

Return the resulting table in any order.\*/

select p.product\_name, s.year, s.price

from Sales s

inner join Product p on p.product\_id = s.product\_id

/\*Q64. Write an SQL query that reports the average experience years of all the employees for each project,

rounded to 2 digits.\*/

select p.project\_id, round(avg(experience\_years), 2) as avg\_years

from project p

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

group by p.project\_id;