1)Write an SQL query to report the first name, last name, city, and state of each person in the Person

table. If the address of a personId is not present in the Address table, report null instead.

select firstname,lastname,city,state from person left join address on person.personId=Address.personid

2)Write an SQL query to report the second highest salary from the Employee table. If there is no second highest salary, the query should report null.

Select max(salary) as SecondHighestSalary from Employee where salary not in (select max(salary) from Employee)

SELECT MAX(Salary) From Employee e WHERE Salary < ( SELECT Max(Salary) FROM Employee);

3)Write an SQL query to report the nth highest salary from the Employee table. If there is no nth highest salary, the query should report null.

SELECT DISTINCT salary AS nth\_highest\_salary FROM Employee ORDER BY salary DESC LIMIT 1 OFFSET (N - 1);

Eg:

SELECT DISTINCT salary

FROM Employee

ORDER BY salary DESC

LIMIT 1 OFFSET 4;

4)Write an SQL query to rank the scores. The ranking should be calculated according to the

following rules1

\_x0019\_ The scores should be ranked from the highest to the lowest+

\_x0019\_ If there is a tie between two scores, both should have the same ranking+

\_x0019\_ After a tie, the next ranking number should be the next consecutive integer value. In other

words, there should be no holes between ranks.

select Score, dense\_rank() over(order by Score desc) as 'Rank' from Scores

5)Write an SQL query to find all numbers that appear at least three times consecutively.

SELECT DISTINCT t1.num

FROM logs t1

JOIN logs t2 ON t1.num = t2.num

JOIN logs t3 ON t2.num = t3.num

JOIN logs t4 ON t3.num = t4.num

WHERE t1.id = t2.id - 1 AND t2.id = t3.id - 1 AND t3.id= t4.id-1

6)Write an SQL query to find the employees who earn more than their managers.

Select e.Name as Employee from Employee e inner join Employee m

on e.ManagerId = m.Id where e.Salary > m.Salary

7)Write an SQL query to report all the duplicate emails. Note that it's guaranteed that the email

field is not NULL.

select Email

from Person

group by Email

having count(Email) > 1;

8)Write an SQL query to report all customers who never order anything.

select

Name as Customers

from

Customers

where

Id

not in(

select CustomerId from Orders

);

9)Write an SQL query to find employees who have the highest salary in each of the departments.

SELECT d.name AS Department, e.name AS Employee, e.Salary

FROM Employee e

INNER JOIN Department d ON e.departmentId = d.id

WHERE (e.departmentId, e.salary) IN (

SELECT departmentId, MAX(salary)

FROM Employee

GROUP BY departmentId

);

SELECT d.Name as Department, e.Name as Employee, e.Salary FROM Employee e

    left join Department d on e.DepartmentId=d.Id

    GROUP BY d.Name

    order by e.Salary desc limit 1

10)Department Top Three Salaries

Question. 10

A company's executives are interested in seeing who earns the most money in each of the

company's departments. A high earner in a department is an employee who has a salary in the

top three unique salaries for that department.

Write an SQL query to find the employees who are high earners in each of the departments.

SELECT

d.Name AS 'Department', e1.Name AS 'Employee', e1.Salary

FROM

Employee e1

JOIN

Department d ON e1.DepartmentId = d.Id

WHERE

3 > (SELECT

COUNT(DISTINCT e2.Salary)

FROM

Employee e2

WHERE

e2.Salary > e1.Salary

AND e1.DepartmentId = e2.DepartmentId

)

11)Write an SQL query to delete all the duplicate emails, keeping only one unique email with the

smallest id. Note that you are supposed to write a DELETE statement and not a SELECT one.

After running your script, the answer shown is the Person table. The driver will first compile and

run your piece of code and then show the Person table. The final order of the Person table does

not matter.

delete A from Person A, Person B where A.id > B.id and A.email=B.email;

12)Write an SQL query to find all dates' Id with higher temperatures compared to its previous dates

(yesterday).

SELECT

weather.id AS 'Id'

FROM

weather

JOIN

weather w ON DATEDIFF(weather.recordDate, w.recordDate) = 1

AND weather.Temperature > w.Temperature

13)The cancellation rate is computed by dividing the number of canceled (by client or driver)

requests with unbanned users by the total number of requests with unbanned users on that

day.

Write a SQL query to find the cancellation rate of requests with unbanned users (both client and

driver must not be banned) each day between "2013-10-01" and "2013-10-03". Round

Cancellation Rate to two decimal points.

SELECT t.request\_at AS 'Day', ROUND(SUM(

IF(t.status LIKE 'ca%', 1, 0)

)/COUNT(\*),2) AS "Cancellation Rate"

FROM Trips t

JOIN Users clients

ON t.client\_id = clients.users\_id

AND clients.banned = 'No'

JOIN Users drivers

ON t.driver\_id = drivers.users\_id

AND drivers.banned = 'No'

WHERE t.request\_at

BETWEEN '2013-10-01'

AND '2013-10-03'

GROUP BY t.request\_at;

14)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 or referee\_id is null

15)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

(

select customer\_number, count(\*) as cnt from orders group by customer\_number

) as e

order by e.cnt desc

limit 1

16)A country is big if,

\* it has an area of at least three million (i.e., 3000000 km2), o6

\* it has a population of at least twenty-five million (i.e., 25000000).

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

select name,population,area from world where area>=3000000 or population>=25000000

17)Write an SQL query to report all the classes that have at least five students.

select class from Courses group by class having count(class)>= 5

18)Write an SQL query to display the records with three or more rows with consecutive id's, and the

number of people is greater than or equal to 100 for each.

select distinct t1.\*

from stadium t1, stadium t2, stadium t3

where t1.people >= 100 and t2.people >= 100 and t3.people >= 100

and

(

(t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1) -- t1, t2, t3

or

(t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3

or

(t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1

)

order by t1.id

19)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".

SELECT s.name

FROM orders o

JOIN company c ON o.com\_id = c.com\_id AND c.name = 'RED'

RIGHT JOIN salesperson s ON o.sales\_id = s.sales\_id

WHERE o.sales\_id IS NULL;

20)Each node in the tree can be one of three types#

& "Leaf": if the node is a leaf node

& "Root": if the node is the root of the tree

& "Inner": If the node is neither a leaf node nor a root node.

Write an SQL query to report the type of each node in the tree.

SELECT Id,

CASE

WHEN p\_id IS NULL THEN "Root"

WHEN (p\_id IS NOT NULL AND id IN (SELECT p\_id FROM Tree)) THEN "Inner"

ELSE "Leaf"

END AS Type

FROM tree;

21)Write an SQL query to report the movies with an odd-numbered ID and a description

that is not "boring".

SELECT \*

FROM cinema

WHERE id % 2 = 1 AND description != "boring"

ORDER BY rating DESC;

22)Write an SQL query to swap the seat id of every two consecutive students. If the number of

students is odd, the id of the last student is not swapped.

SELECT

(CASE

WHEN MOD(id, 2) != 0 AND counts != id THEN id + 1

WHEN MOD(id, 2) != 0 AND counts = id THEN id

ELSE id - 1

END) AS id,

student

FROM

seat,

(SELECT

COUNT(\*) AS counts

FROM

seat) AS seat\_counts

ORDER BY id ASC

23)Write an SQL query to swap all 'f' and 'm' values (i.e., change all 'f' values to 'm' and vice versa) with a

single update statement and no intermediate temporary tables.

UPDATE salary

SET sex=CASE

WHEN sex='f' THEN 'm'

WHEN sex='m' THEN 'f'

END

WHERE sex IN ('m', 'f')

24)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(\*) >= 3

25)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.

SELECT product\_id,

product\_name

FROM product

WHERE product\_id NOT IN (SELECT product\_id

FROM sales

WHERE sale\_date NOT BETWEEN

'2019-01-01' AND '2019-03-31');

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

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

from Activity

group by player\_id;

27)Write an SQL query to find the daily active user count for a period of 30 days ending

2019-07-27 inclusively. A user was active on someday if they made at least one

activity on that day.

select activity\_date as day, count(distinct(user\_id)) as active\_users from Activity

where activity\_date between "2019-06-28" and "2019-07-27"

group by activity\_date;

28)Write an SQL query to find all the authors that viewed at least one of their own articles.

select author\_id as id

from Views

where author\_id = viewer\_id

group by author\_id

order by author\_id asc

29)Write an SQL query to find for each user, the join date and the number of orders they

made as a buyer in 2019.

30)Write an SQL query to reformat the table such that there is a department id column

and a revenue column for each month.

select id,  
max(if(month = 'Jan', revenue, null)) as Jan\_Revenue,  
max(if(month = 'Feb', revenue, null)) as Feb\_Revenue,  
max(if(month = 'Mar', revenue, null)) as Mar\_Revenue,  
max(if(month = 'Apr', revenue, null)) as Apr\_Revenue,  
max(if(month = 'May', revenue, null)) as May\_Revenue,  
max(if(month = 'Jun', revenue, null)) as Jun\_Revenue,  
max(if(month = 'Jul', revenue, null)) as Jul\_Revenue,  
max(if(month = 'Aug', revenue, null)) as Aug\_Revenue,  
max(if(month = 'Sep', revenue, null)) as Sep\_Revenue,  
max(if(month = 'Oct', revenue, null)) as Oct\_Revenue,  
max(if(month = 'Nov', revenue, null)) as Nov\_Revenue,  
max(if(month = 'Dec', revenue, null)) as Dec\_Revenue  
from Department  
group by id

31)Write an SQL query to report the Capital gain/loss for each stock.

The Capital gain/loss of a stock is the total gain or loss after buying and selling the

stock one or many times.

select stock\_name, sum(if(operation='sell', price,-price)) as capital\_gain\_loss

from stocks

group by stock\_name

32)Write an SQL query to report the distance traveled by each user.

SELECT Users.name, IFNULL(SUM(Rides.distance), 0) AS travelled\_distance  
FROM Users LEFT JOIN Rides  
ON Users.id = Rides.user\_id  
GROUP BY user\_id  
ORDER BY travelled\_distance DESC,  
name ASC;

33)Write an SQL query to find for each date the number of different products sold and

their names.

The sold products names for each date should be sorted lexicographically.

select sell\_date, count(distinct product) as num\_sold,  
group\_concat(distinct product order by product asc) as products  
from Activities  
group by sell\_date  
order by sell\_date

34)Write an SQL query to report the patient\_id, patient\_name and conditions of the

patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix.

SELECT \* FROM Patients

WHERE conditions like '% DIAB1%' OR conditions like'DIAB1%'

35)Write a SQL query to find the IDs of the users who visited without making any

transactions and the number of times they made these types of visits.

SELECT V.customer\_id, COUNT(V.visit\_id) AS count\_no\_trans  
FROM Visits V  
LEFT JOIN Transactions T ON V.visit\_id = T.visit\_id  
WHERE T.transaction\_id IS NULL  
GROUP BY V.customer\_id;

36)Write an SQL query to report the name and balance of users with a balance higher

than 10000. The balance of an account is equal to the sum of the amounts of all

transactions involving that account.

SELECT u.name AS NAME,SUM(t.amount) AS BALANCE

FROM Transactions t LEFT JOIN Users u

ON u.account = t.account

GROUP BY u.account

HAVING SUM(t.amount)>10000;

37)Write an SQL query to fix the names so that only the first character is uppercase and the rest

are lowercase.

select user\_id,  
concat(upper(left(name, 1)), lower(right(name, length(name) - 1))) as name  
from Users  
group by 1  
order by 1;

38)

39)Write an SQL query that will, for each user, return the number of followers.

Return the result table ordered by user\_id in ascending order.

select user\_id, count(distinct follower\_id) followers\_count

from followers

group by user\_id

order by user\_id

40)Write an SQL query to calculate the total time in minutes spent by each employee on

each day at the office. Note that within one day, an employee can enter and leave

more than once. The time spent in the office for a single entry is out\_time - in\_time.

SELECT event\_day AS day, emp\_id, SUM(out\_time - in\_time) AS total\_time

FROM Employees

GROUP BY event\_day, emp\_id;

41)Write an SQL query to find the ids of products that are both low fat and recyclable.

SELECT product\_id   
FROM Products   
WHERE low\_fats='Y' and recyclable='Y';

42)Write an SQL query to rearrange the Products table so that each row has (product\_id, store, price). If

a product is not available in a store, do not include a row with that product\_id and store

combination in the result table.

SELECT product\_id, 'store1' store, store1 price

FROM Products

where store1 is not null

union

SELECT product\_id, 'store2' store, store2 price

FROM Products

where store2 is not null

union

SELECT product\_id, 'store3' store, store3 price

FROM Products

where store3 is not null

43)Write an SQL query to calculate the bonus of each employee. The bonus of an

employee is 100% of their salary if the ID of the employee is an odd number and the

employee name does not start with the character 'M'. The bonus of an employee is 0

otherwise

44)Write an SQL query to report the latest login for all users in the year 2020. Do not

include the users who did not login in 2020.

SELECT user\_id,

MAX(time\_stamp) AS last\_stamp

FROM Logins

WHERE YEAR(time\_stamp) = 2020

GROUP BY user\_id

ORDER BY NULL;

45)Write an SQL query to report the IDs of all the employees with missing information. The

information of an employee is missing if6

\* The employee's name is missing, oA

\* The employee's salary is missing.

select e.employee\_id

from Employees e

left join Salaries s

ON s.employee\_id = e.employee\_id

where s.salary is null

union all

select s.employee\_id

from Employees e

right join Salaries s

ON s.employee\_id = e.employee\_id

where e.name is null

order by employee\_id