SQL CHALLENGE

[Suru Shashank]

Q1. Query all columns for all American cities in the CITY table with populations larger than 100000.
The CountryCode for America is USA.
The CITY table is described as follows:

ANS:

```
#QUERY-1#
Select * from cities where country_code='USA' and population > 100000;
```

Result Grid | Filter Rows: | Export:

id	name	country_code	district	population
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

Q2. Query the NAME field for all American cities in the CITY table with populations larger than 120000.
The CountryCode for America is USA.
The CITY table is described as follows:

ANS:

```
#QUERY-2#
Select name from cities where country_code='USA' and population > 120000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

name
Scottsdale
Corona
Concord
Cedar Rapids

Q3. Query all columns (attributes) for every row in the CITY table.
The CITY table is described as follows:

ANS:

```
#QUERY-3#
select * from cities;
```

esult Grid | Filter Rows: | Export: | Wrap

id	name	country_code	district	population
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

Q4. Query all columns for a city in CITY with the ID 1661.
The CITY table is described as follows:

ANS:

```
#QUERY-4#
select * from cities where id='1661'
```

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

ANS:

```
#QUERY-5#
select * from cities where country_code='JPN'
```

Q6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

ANS:

```
#QUERY-6#
select name from cities where country_code='JPN'
```

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

ANS:

```
#QUERY-7#
select city,state from station;
```

esult Grid | Filter Rows:

city	state
Kissee Mills	MO
Loma Mar	CA
Sandy Hook	CT
Tipton	IN
Arlington	CO
Turner	AR
Slidell	LA
Negreet	LA
Glencoe	KY
Chelsea	IA
Chignik Lag...	AK

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.

ANS:

```
#QUERY-8#
SELECT DISTINCT CITY FROM STATION WHERE MOD(STATION.ID,2)=0 ORDER BY CITY;

#QUERY-7#
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CITY
Albany
Cahone
Chignik Lagoon
Glencoe
Kissee Mills
Loma Mar
Manchester
Tipton

Q9. Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.

ANS:

```
#QUERY-9#
select(count(city)- count(distinct city))  as difference from station;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

difference
0

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.

ANS:

Longest characters:

```
#QUERY-10#
select city, length(city) from station order by length(city) desc limit 1;
#longest city name
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

city	length(city)
Chignik Lagoon	14

Shortest characters:

```
select city, length(city) as no_of_chars_in_cityname from station order by length(city) asc limit 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

city	no_of_chars_in_cityname
Tipton	6

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.

ANS:

```
#QUERY-11#
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%");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Arlington
Albany

Q12. Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.

ANS:

```
#QUERY-12#
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");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Glencoe
Chelsea
Pelahatchie
Dorrance
Cahone

Q13. Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.

ANS:

```
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%");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Kissee Mills
Loma Mar
Sandy Hook
Tipton
Turner
Slidell
Negreet
Glencoe
Chelsea
Chignik Lagoon

Q14. Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.

ANS:

```
#QUERY-14#
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");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Kissee Mills
Loma Mar
Sandy Hook
Tipton
Arlington
Turner
Slidell
Negreet
Chignik Lagoon
Hanna City
Albany

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.

ANS:

```
#QUERY-15#
select distinct city
from station
where (city not in (select distinct city from station where city like '%a' and city like '%e' and city like '%i' and city like '%o' and city l
or
(city not in (select city from station where city like 'a%' and city like 'e%' and city like 'i%' and city like 'o%' and city like 'u%'));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Kissee Mills
Loma Mar
Sandy Hook
Tipton
Arlington
Turner
Slidell
Negreet
Glencoe
Chelsea
Chignik Lag...

Result Grid | Form Editor | Field Types

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.

ANS:

```
#QUERY-16#
select distinct city
from station
where (city not in (select distinct city from station where city like '%a' or city like '%e' or city like '%i' or city like '%o' or city like
or
(city not in (select city from station where city like 'a%' or city like 'e%' or city like 'i%' or city like 'o%' or city like 'u%'));
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

city
Kissee Mills
Loma Mar
Sandy Hook
Tipton
Arlington
Turner
Slidell
Negreet
Glencoe

17)

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.

ANS:

```
#QUERY-17#
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');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

product_id	product_name
1	S8
NULL	NULL

18)

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.

ANS:

```sql
#QUERY-18#
select distinct author_id as id
from Views
where author_id = viewer_id
order by author_id asc
```

Result Grid | Filter Rows: | Exp

id
4
7

19)

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

ANS:

```sql
select round(100*del2.immediate_orders/count(del1.delivery_id), 2) as immediate_delivery_percent
from delivery as del1,
    (select count(order_date) as immediate_orders
    from delivery
    where (order_date = customer_pref_delivery_date)) del2
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

immediate_delivery_percent
33.33

20)

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

ANS:

```
select
    ad_id,
    round(sum(case when action='clicked' then 1 else 0 end) * 100 /
    (sum(case when action='clicked' then 1 else 0 end) +
    sum(case when action='viewed' then 1 else 0 end)), 2) as ctr
from adss
group by 1
order by 2 desc, 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ad_id	ctr
1	66.67
3	50.00
2	33.33
5	NULL

21)

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

ANS:

```
#QUERY-21#
select employee_id, count(team_id) over (partition by team_id) team_size
from employee
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_id	team_size
4	1
1	3
2	3
3	3
5	2
6	2

22)

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.

ANS:

```
#QUERY-22#
select c.country_name,
        case
            when avg(w.weather_state) <= 15.0 then 'Cold'
            when avg(w.weather_state) >= 25.0 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'
group by c.country_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

country_name	weather_type
USA	Cold
Australia	Cold
China	Warm
Peru	Hot
Morocco	Hot

23)

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.

```
#QUERY-23#
select units_sold.product_id, round(sum(units*price)/sum(units), 2) as average_price
from units_sold inner join prices
on units_sold.product_id = prices.product_id
and units_sold.purchase_date between prices.start_date and prices.end_date
group by units_sold.product_id
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_id	average_price
1	6.96
2	16.96

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

ANS:

```
#QUERY-24#
select player_id, min(event_date) as first_login
from activity
group by player_id
```

Result Grid | Filter Rows: | Export: | Wrap Ce

player_id	first_login
1	2016-03-01
2	2017-06-25
3	2016-03-02

25)

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

ANS:

```
#QUERY-25#
select player_id, min(device_id)
from activity
group by player_id
```

Result Grid | Filter Rows:

player_id	min(device_id)
1	2
2	3
3	1

26)

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

ANS:

```
#QUERY-26#
select p.product_name as product_name, o.sum_unit as unit from products p
join

(select product_id, sum(unit) as sum_unit from orders where order_date >= '2020-02-01' and order_date < '2020-03-01
group by product_id) o

on p.product_id = o.product_id
where o.sum_unit >= 100
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	unit
Leetcode Solutions	130
Leetcode Kit	100

27)

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'.

ANS:

```
#QUERY-27#
SELECT *
FROM Users
WHERE REGEXP_LIKE(mail, '^[a-zA-Z][a-zA-Z0-9\_\.\-]*@leetcode.com')
```

28)

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.

ANS:

```
#QUERY-28#
select o.customer_id, c.name
from customers as c, product p, orders o
where c.customer_id = o.customer_id and p.product_id = o.product_id
group by o.customer_id
having
(
    sum(case when o.order_date like '2020-06%' then o.quantity*p.price else 0 end) >= 100
    and
    sum(case when o.order_date like '2020-07%' then o.quantity*p.price else 0 end) >= 100
)
```

29)

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.

ANS:

```
#QUERY-30#
select distinct title
from content
join tv_program using(content_id)
where kids_content = 'Y'
    and content_type = 'Movies'
    and (month(program_date)=6 and year(program_date) =  2020
```

30)

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

ANS:

```
select q.id, q.year
from querie as q
left join npv as n
on (q.id, q.year) = (n.id, n.year)
```

Result Grid | Filter Rows: | Exp

id	year
1	2019
2	2008
3	2009
7	2018
7	2019

31) REPEATED previously

32)

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.

ANS:

```
#QUERY-32#
select unique_id, name
from employees1 left join employees2
on employees1.id = employees2.id
order by name asc;
```

Result Grid | Filter Rows: | Expor

unique_id	name
NULL	Alice
NULL	Bob
1	Jonathan
2	Meir
3	Winston

33)

Write an SQL query to report the distance travelled by each user.

ANS:

```
#QUERY-33#
select name, sum(distance) as travelled_distance
from rides r
right join users u
on r.user_id = u.id
group by name
order by 2 desc,1 asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

name	travelled_distance
Elvis	450
Lee	450
Bob	317
Jonathan	312
Alex	222

34)

REPEATED PREVIOUSLY

35)

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.

ANS:

```
(
    select name results
    from movie_rating natural join users1
    group by users1.user_id
    order by count(*) desc, name asc
    limit 1
)
union
(
    select movies.title results
    from movie_rating natural join movies
    where month(created_at)='2'
    group by movies.movie_id
    order by avg(rating) desc,title asc
);
```

36) REPEATED PREVIOUSLY

37) REPEATED PREVIOUSLY

38)

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.

ANS:

```
#QUERY-38#
select id, name
from students
where department_id not in (select id from departments)
```

Result Grid | Filter Rows: | Edit: | Export/Im

id	name
2	John
3	Steve
4	Jasmine
7	Daiana
NULL	NULL

39)

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.

ANS:

```
select from_id as person1,to_id as person2,
    count(duration) as call_count, sum(duration) as total_duration
from (select * from calls
union all
select to_id, from_id, duration
from calls) as t1
where from_id < to_id
group by person1, person2

drop table calls;

select * from calls;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person1	person2	call_count	total_duration
1	2	2	70
1	3	1	20
3	4	4	999

40)

REPEATED AGAIN-Q23

41)

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

ANS:

```
select warehouse_name, sum(volume) as volume from (
    select w.name as warehouse_name
```

42)

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.

ANS:

```
#QUERY-42#
select a.sale_date, a.sold_num - b.sold_num as difference
from sale1 a left join sale1 b
on a.sale_date = b.sale_date
where a.fruit = 'apples' and b.fruit = 'oranges'
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sale_date	difference
2020-05-01	2
2020-05-02	0
2020-05-03	20
2020-05-04	-1

43) pending

44)

Write an SQL query to report the managers with at least five direct reports. Return the result table in any order.

ANS:

```
select e2.name
from employee2 e1
inner join employee2 e2 on e1.managerid = e2.id
group by e1.managerid
having count(e1.id) >= 5;
```

Result Grid | Filter Rows: | Export: | Wrap

name
John

45)

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
    a.dept_name,
    count(student_id) student_count
from
    department2 a
left join
    student2 b
on
    (a.dept_id = b.dept_id)
group by a.dept_name
order by student_count desc, a.dept_name asc;
```

Result Grid | Filter Rows: | Export: | Wrap

dept_name	student_count
Engineering	2
Science	1
Law	0

46)

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

ANS:

```
#QUERY-46#
select a.customer_id from
(select customer_id, count(distinct product_key) as num
from customer4
group by customer_id) a
where a.num = (select count(distinct product_key) from product4);
```

47)

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.

ANS:

```
select project_id, project47.employee_id
from project47 inner join Employee
on project47.employee_id = employee47.employee_id
where (project_id, experience_years) in
    (select project_id, max(experience_years)
```

48) 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.

```
#QUERY-48#
select book_id, name
from Books
where book_id not in (
    select book_id
    from Orders
    where dispatch_date >= '2018-06-23' and dispatch_date <= '2019-06-22'
    group by book_id
    having sum(quantity) >= 10)
and available_from < '2019-05-23'
```

49)

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.

ANS:

```
#QUERY-49#
select student_id, min(course_id) as course_id, grade from enrollments48
    where (student_id, grade) in (
        select student_id, max(grade) from enrollments48 group by student_id
    )
    group by student_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

student_id	course_id	grade
1	2	99
2	2	95
3	3	82

50) QUESTION is wrong because the given input tables to create say "teams" and matches but the expected output shows "players". Confusing…

51) Write an SQL query to report the name, population, and area of the big countries. Return the result table in any order.

ANS:

```
#QUERY-51#
select name,population,area
from world1
where area > 3000000 or population > 25000000;
```

Result Grid | Filter Rows: | Edit:

name	population	area
Afghanistan	25500100	652230
Algeria	37100000	2381741
NULL	NULL	NULL

52)

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

```
#QUERY-52#
select name
from customer52
where referee_id <> 2 or referee_id is NULL
```

Result Grid | Filter Rows: | Export: | W

name
Will
Jane
Bill
Zack

ANS:

53) Write an SQL query to report all customers who never order anything. Return the result table in any order.

ANS:

```
select c.name as customers
from customers53 as c
where c.id not in (
select o.customerId from orders53 as o
);
```

54)

Write an SQL query to find the team size of each of the employees.
Return result table in anv order.

ANS:

```
select e1.employee_id, count(*) as team_size
    from employee54 e1 left join employee54 e2
    on e1.team_id = e2.team_id
    group by e1.employee_id;
```

Result Grid | Filter Rows: | Export: | Wrap

employee_id	team_size
1	3
2	3
3	3
4	1
5	2
6	2

55)

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

ANS:

```
#QUERY-55#
select c.name as country
from person55 p
inner join country55 c
on left (p.phone_number,3) = c.country_code
inner join (select caller_id as id, duration
            from calls55
union
select callee_id as id, duration
            from calls55) phn
on p.id = phn.id
group by country
having avg(duration) > (select avg(duration) from calls55);
```

56)

ANS: REPEATED PREVIOUSLY→ANS->25

57)

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.

ANS:

```
SELECT
    customer_number
FROM orders57
GROUP BY 1
ORDER BY COUNT(*) DESC
LIMIT 1
```

Result Grid | Filter Rows:

customer_number
3

58) 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.

ANS:

```
#QUERY-58#
SELECT c1.seat_id
FROM   Cinema c1,
       Cinema c2
WHERE  ( ( c1.seat_id = c2.seat_id + 1 )
         OR ( c1.seat_id = c2.seat_id - 1 ) )
```

59)

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 name
from sales_person
where name not in
    (select distinct sales_person.name
    from sales_person, orders59, company59
    where company59.name = 'red'
    and sales_person.sales_id = orders59.sales_id
    and orders59.com_id = company59.com_id)
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

name
Amy
Mark
Alex

ANS:

60)

Write an SQL query to report for every three line segments whether they can form a triangle. Return the result table in any order.

ANS:

```sql
#QUERY-60#
select x, y, z,
    case when x + y > z and y + z > x then 'Yes'
    when y + z > x and z + x > y then 'No'
    else 'wrong'
    end as triangle
from triangle;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

x	y	z	triangle
10	20	15	Yes
13	15	30	No

61)

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

ANS:

```sql
select min(p2.x - p1.x) as least
from point as p1 inner join point as p2
on p1.x <> p2.x;
```

62)

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.

ANS:

```sql
#QUERY-62#
select actor_id, director_id from actordirector
    group by actor_id, director_id
    having count(actor_id) >= 3;
```

Result Grid | Filter Rows: | Export: | Wrap C

actor_id	director_id
1	1

63)

Write an SQL query that reports the product_name, year, and price for each sale_id in the Sales table.

ANS:

```
#QUERY-63#
select product_name, year, price from sales63
left join product63
on sales63.product_id = product63.product_id;
```

64)

Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits.

```
#QUERY-64#
select project_id, round(avg(experience_years), 2) as avg_years
from project64 inner join employee64
on project64.employee_id = employee64.employee_id
group by project_id
```

65)

Write an SQL query that reports the best seller by total sales price, If there is a tie, report them all.

ANS:

```
select a.seller_id
from
(select seller_id, sum(price) as sum
from sales65
group by seller_id) as a1
where a.sum = (select max(b.sum)
from(select seller_id, sum(price) as sum
from sales65
group by seller_id) as b1 )
```

66)

67)

Write an SQL query to compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average_amount should be rounded to two decimal places.

ANS:

```
select
    visited_on,
    sum(amount) over(order by visited_on rows between 6 preceding and current row) as amount,
    round(avg(amount) over(order by visited_on rows between 6 preceding and current row),2) as average_amount,
    dense_rank() over(order by visited_on) as rnk
from result
)
```

68)

Write an SQL query to find the total score for each gender on each day.
Return the result table ordered by gender and day in ascending order.

ANS:

```
#QUERY-68#
select s.gender, s.day,
(select sum(score_points) from scores68
where gender = s.gender and day <= s.day) as total
    from scores68 as s
    group by gender, day
    order by gender, day;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

gender	day	total
F	2019-12-30	17
F	2019-12-31	40
F	2020-01-01	57
F	2020-01-07	80
M	2019-12-18	2
M	2019-12-25	13
M	2019-12-30	26

69)

Write an SQL query to find the start and end number of continuous ranges in the table Logs. Return the result table ordered by start_id.

ANS:

with cte as(

select log_id,

log_id

70)

ANS:

71)

Write an SQL query to find employee_id of all employees that directly or indirectly report their work to the head of the company.

ANS:

```
#QUERY-71#
select t1.employee_id
from Employees as t1 inner join employees71 as t2
on t1.manager_id = t2.employee_id
inner join employees71 as t3
on t2.manager_id = t3.employee_id
where t3.manager_id = 1 and t1.employee_id <> 1
```

72)

Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

ANS:

```
#QUERY-72#
select trans_date as month, country,
    count(id) 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 transactions72
group by month, country;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	country	trans_count	approved_count	trans_total_amount	approved_total_amount
2018-12-18	US	1	1	1000	1000
2018-12-19	US	1	0	2000	0
2019-01-01	US	1	1	2000	2000
2019-01-07	DE	1	1	2000	2000

73)

Write an SQL query to find the average daily percentage of posts that got removed after being reported as spam, rounded to 2 decimal places.

ANS:

```
#QUERY-73#
(select
        action_date,
        count(distinct a.post_id) as total,
        count(distinct r.post_id) as del
    from
        actions as a left join removals as r using (post_id)
    where
        action = 'report' and extra = 'spam'
```

74) and 75) repeated previously/

76)

Write an SQL query to find the salaries of the employees after applying taxes. Round the salary to the nearest integer.
The tax rate is calculated for each company based on the following criteria:

- 0% If the max salary of any employee in the company is less than $1000.
- 24% If the max salary of any employee in the company is in the range [1000, 10000] inclusive.
- 49% If the max salary of any employee in the company is greater than $10000.

ANS:

```
#QUERY-76#
select salaries76.company_id, salaries76.employee_id, salaries76.employee_name,
    round(case when salary_max<1000 then salaries76.salary
               when salary_max>=1000 and salary_max<=10000 then salaries76.salary * 0.76
               else salaries76.salary * 0.51 end, 0) as salary
from Salaries
```

77)

Write an SQL query to evaluate the boolean expressions in Expressions table.
Return the result table in any order.

ANS:

```
#QUERY-77#
select e.left_operand, e.operator, e.right_operand,
       case when
       e.operator = '<' then if(l.value < r.value, 'true', 'false')
       when e.operator = '>' then if(l.value > r.value, 'true', 'false')
       else if(l.value = r.value, 'true', 'false') end
       as value
from expressions77 e
```

78) Repeated previously

79)

Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.

ANS:

```
select name from employee73 order by name;
```

84)

Write a query to update the Facebook advertiser's status using the daily_pay table. Advertiser is a two-column table containing the user id and their payment status based on the last payment and daily_pay table has current information about their payment. Only advertisers who paid will show up in this table.

ANS:

```
#QUERY-84#
select advertiser.user_id, advertiser.status,payment.paid
from advertiser
left join daily_pay as pay
on advertiser.user_id = pay.user_id
union
select pay.user_id,advertiser.status,pay.paid
from daily_pay as pay
left join advertiser
  on advertiser.user_id = pay.user_id
)
```

88) Previously Repeated

89) Previously Repeated

91)

ANS:

91)

Write an SQL query to report the comparison result (higher/lower/same) of the average salary of employees in a department to the company's average salary.

ANS:

```
#QUERY-91#
select department_salary.pay_month, department_id,
    case
        when department_avg > company_avg then 'higher'
        when department_avg < company_avg then 'lower'
        else 'same'
    end as comparison
    from (
        select department_id, avg(amount) as department_avg,
        from salary
        join employee91 on salary.employee_id = employee91.employee_id
            group by department_id,
    ) as department_salary
```

92)

Write an SQL query to report for each install date, the number of players that installed the game on that day, and the day one retention.

ANS:

```
    #QUERY-92#
select t1.install_date as install_dt, count(t1.install_date) as installs,
count(t2.event_date) / count(*) as first_retention
from (
    select player_id, min(event_date) as install_date
    from activity92
    group by 1
) as team1
left join activity92 as team2
on t1.install_date = t2.event_date
    and t1.player_id = t2.player_id
group by 1
order by 1;
```

93)

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

ANS:

```
select group_id, player_id from (
select p.group_id, ps.player_id, sum(ps.score) as scores
from players93 as p,
    (
    select first_player as player_id, first_score as scores
    from matches93
    union
    select second_player, second_score
    from matches
    ) as pls
    where p.player_id = pls.player_id
    group by ps.player_id
    order by group_id, score desc, player_id
) as highest_score
group by group_id;
```

94)

95) Repeated, same as 94)

96)

You're given two tables on Spotify users' streaming data. songs_history table contains the historical streaming data and songs_weekly table contains the current week's streaming data.
Write a query to output the user id, song id, and cumulative count of song plays as of 4 August 2022 sorted in descending order.

ANS:

```
#QUERY-96#
select user_id, song_id, sum(song_plays) as total_songs
from (
select user_id, song_id, song_plays
from songs_history
union all
select user_id, song_id, count(song_id) as songs_played
from songs_weekly
where listen_time <= '08/04/2022 23:59:59'
group by user_id, song_id
) as full
group by user_id, song_id
order by total_songs desc;
```

99)

Assume you are given the tables below containing information on Snapchat users, their ages, and their time spent sending and opening snaps. Write a query to obtain a breakdown of the time spent sending vs. opening snaps (as a percentage of total time spent on these activities) for each age group.

ANS:

```
select
age.age_bucket,
sum(case when activities99.activity_type = 'sending_snaps'
    then activities99.time_spent else 0 end) as sending_time,
sum(case when activities99.activity_type = 'opening_snaps'
    then activities99.time_spent else 0 end) as opening_time,
sum(activities99.time_spent) as total_time
  from activities99
inner join age_breakdown as age_br
    on activities99.user_id = age_br.user_id
  where activities99.activity_type in ('sending_snaps', 'opening_snaps')
  group by age_br.age_bucket)
```