```SQL LEETCODE 50-QUESTIONS```

BASIC TO INTERMEDIATE LEVEL QUESTIONS

SOLUTIONS BY - AAKASHDEEP SEDHA
```

SELECT

1. [Recyclable and Low Fat Products - LeetCode]

```
select p.product_id
from Products p
where low_fats = 'Y' and recyclable = 'Y'
group by p.product_id
```

2. [Find Customer Referee - LeetCode]

   select name from Customer where coalesce(referee_id,0) <> 2

3. [Big Countries - LeetCode]

   SELECT name,population,area from World where(population >= 25000000) or (area >= 3000000)

4. [Article Views I - LeetCode]

Select DISTINCT author_id as id
from Views
where author_id = viewer_id
order by author_id asc

5.[Invalid Tweets - LeetCode]

```
Select tweet_id
from Tweets
where len(content)>15
```

Basic Joins

BASIC JOINS

6.[Replace Employee ID With The Unique Identifier - LeetCode]

```
select eu.unique_id , e.name
from Employees e
left join EmployeeUNI  eu
on e.id = eu.id
```

7.[Product Sales Analysis I - LeetCode]

```
select product.product_name,sales.year,sales.price
from sales,product
where sales.product_id=product.product_id;
```

8.[Customer Who Visited but Did Not Make Any Transactions - LeetCode]

```
select v.customer_id , count(1) 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
```

9.[Rising Temperature - LeetCode]

```
with cte as(
SELECT *,
      LAG(temperature, 1, 999) OVER (ORDER BY
recordDate ASC) AS prev_temp,
      LAG(recordDate, 1, NULL) OVER (ORDER BY
recordDate ASC) AS prev_day
FROM Weather

)
select id
from cte
where temperature>prev_temp and
dateadd(day,1,prev_day) = recordDate
```

10 Average Time of Process per Machine - LeetCode

```
select a1.machine_id
, ROUND(AVG(a2.timestamp - a1.timestamp),3) as
processing_time
from Activity a1
join Activity a2
on a1.process_id=a2.process_id
and a1.machine_id=a2.machine_id
and a1.timestamp<a2.timestamp
group by a1.machine_id;
```

11Employee Bonus - LeetCode

```
SELECT e.name, b.bonus
FROM Employee e
LEFT JOIN Bonus b ON e.empId = b.empId
WHERE b.bonus < 1000 OR b.bonus IS NULL;
```

12[Managers with at Least 5 Direct Reports - LeetCode]

```
with cte as(
SELECT e.managerId , count(managerId) as cnt
FROM Employee e
group by e.managerId)

Select e.name
from cte
inner join Employee e
on e.id = cte.managerId
where cnt>=5
```

13 [Students and Examinations - LeetCode]

```
---catch was to also add enteries with 0 attempts

--generate all possible pairs
with cte as(
select *
from students
cross join
subjects
)

--compute attempts using window func
,cte2 as
(
```

```
    select
    Distinct
    e.student_id , s.student_name  ,e.subject_name
    , count(1)over(partition by
e.student_id,e.subject_name) as attended_exams
    from examinations e
    left join
    Students s
    on s.student_id = e.student_id

)

--handled 0 attempt case
select
cte.*,
   ISNULL(cte2.attended_exams,0) attended_exams
from cte
left join cte2
on cte.student_id = cte2.student_id and
cte.subject_name = cte2.subject_name
order by cte.student_id , cte.subject_name
```

14.Confirmation Rate - LeetCode

```
with cte as(
SELECT
c.user_id
,sum(case when c.action = 'confirmed' then 1 else 0
end) as
confirmed_requests
,count(1) as total_requests
FROM Confirmations c
group by c.user_id
)
SELECT
s.user_id ,
ISNULL(ROUND(confirmed_requests*1.0/total_reques
ts  , 2),0) as confirmation_rate
FROM
Signups s
left join
cte on s.user_id = cte.user_id
```

BASIC AGGREGATE FUNCTIONS

15.Not Boring Movies - LeetCode

```
select *
from Cinema c1
where c1.description not in ('boring') and c1.id%2!=0
order by rating desc
```

16 Average Selling Price - LeetCode

```
SELECT p.product_id,
ISNULL(round(SUM(p.price*u.units)*1.0/sum(u.units),
2),0) as average_price
FROM Prices p
LEFT JOIN UnitsSold u
ON p.product_id = u.product_id AND
u.purchase_date BETWEEN p.Start_date and
p.end_date
GROUP BY p.product_id
```

17 Project Employees I - LeetCode

```
SELECT p.project_id ,
ROUND(sum(e.experience_years)*1.0/count(1) , 2)
average_years
from Project p
left join  Employee e
on p.employee_id = e.employee_id
group by p.project_id
```

18 Percentage of Users Attended a Contest - LeetCode

```
Select contest_id ,
ROUND((count(1)*1.0*100 /  (SELECT COUNT(user_id)
FROM USERS) ) , 2 )  as percentage
from
Register
group by contest_id
order by ROUND((count(1)*1.0*100 /  (SELECT
COUNT(user_id)  FROM USERS) ) , 2 )  desc
,contest_id asc
```

19 Queries Quality and Percentage - LeetCode

```
WITH PoorQueryCounts AS (
    SELECT
        query_name,
        COUNT(*) AS poor_cnt
    FROM
        Queries
    WHERE rating < 3
    GROUP BY query_name
),
QueryQuality AS (
    SELECT
        q.query_name,
        ROUND(SUM(q.rating * 1.0 / q.position) / COUNT(*), 2) AS
quality,
        ROUND(COALESCE(pq.poor_cnt, 0) * 100.0 / COUNT(*), 2)
AS poor_query_percentage
    FROM
        Queries q
    LEFT JOIN PoorQueryCounts pq ON q.query_name =
pq.query_name
    GROUP BY q.query_name, pq.poor_cnt
)

SELECT
    query_name,
    quality,
    poor_query_percentage
FROM QueryQuality;
```

APPROACH 2:

```
/* Write your T-SQL query statement below */
SELECT query_name,
ROUND(SUM(rating*1.0/position)/count(*),2) as quality,
ROUND(SUM(case WHEN rating <3 then 1 else 0 end )*100.0/
COUNT(*),2)  as poor_query_percentage
FROM Queries
GROUP BY query_name
```

20 Monthly Transactions I - LeetCode

```
SELECT
format(t.trans_date,'yyyy-MM') as month
,t.country
,count(1) as trans_count
,sum(case when t.state = 'approved' then 1 else 0
end) as approved_count
,sum(case when t.state = 'approved' then amount else
0 end) as approved_total_amount
,sum(amount) as trans_total_amount

FROM
   Transactions t
group by country , format(t.trans_date,'yyyy-MM')
```

21 Immediate Food Delivery II - LeetCode

```
WITH cte as (
SELECT
*, row_number() over(partition by customer_id order
by order_date) rn
FROM Delivery)

Select
ROUND(sum(case when cte.order_date =
cte.customer_pref_delivery_date then 1 else 0 end) *
100.0/count(1) , 2) as immediate_percentage
 from cte
where cte.rn = 1
```

22 Game Play Analysis IV - LeetCode**

APPROACH_1:RANK(),JOIN()

```
with cte as (
select *, rank() over (partition by player_id order by event_date) r
from activity)
select
round(cast(count(case when a.event_date=dateadd(day, 1, c.event_date)
then c.player_id end) as float)/
count(distinct a.player_id),2) as fraction
from activity a
 left join cte c on a.player_id=c.player_id and c.r=1
```

APPROACH_2:

```
WITH CTE AS(
SELECT a.player_id , min(a.event_date) first_visit
FROM
    Activity a
GROUP BY a.player_id
)

SELECT
ROUND(sum(case when
DATEDIFF(day,cte.first_visit,a.event_date)=1 then 1 else 0
end)*1.0/(Select count(player_id) from cte),2) as fraction
FROM cte
INNER JOIN  Activity a
on cte.player_id  = a.player_id
```

SORTING & GROUPING

23[User Activity for the Past 30 Days I - LeetCode]

```
SELECT activity_date as day , COUNT(DISTINCT user_id) AS
active_users
FROM Activity
WHERE activity_date > DATEADD(DAY, -30, '2019-07-27')
AND activity_date <= '2019-07-27'
GROUP BY activity_date
```

24[Number of Unique Subjects Taught by Each Teacher - LeetCode]

```
SELECT
t.teacher_id,
count(DISTINCT subject_id) as cnt
FROM Teacher t
group by teacher_id
```

25 Product Sales Analysis III - LeetCode

```
WITH MinYearPerProduct AS (
    SELECT s.product_id, MIN(s.year) AS first_year
    FROM Sales s
    GROUP BY s.product_id
)
SELECT s.product_id, m.first_year, s.quantity, s.price
FROM Sales s
INNER JOIN MinYearPerProduct m ON s.product_id =
m.product_id AND s.year = m.first_year;
```

26 Classes More Than 5 Students - LeetCode

```
SELECT
class
FROM Courses s
group by class
having Count(distinct student) >= 5
```

27 Find Followers Count - LeetCode

```
Select user_id
, count(DISTINCT follower_id) followers_count
from Followers
group by user_id
```

28Biggest Single Number - LeetCode

```
ith cte as(
select  DISTINCT num as num
from MYNumbers
group by num
having(count(num)) = 1

)
Select max(num)AS num from cte
```

29 Customers Who Bought All Products - LeetCode

```
SELECT
customer_id
FROM
Customer
group by customer_id
having count(Distinct product_key) = (Select count(distinct
product_key) from Product )
```

30 The Number of Employees Which Report to Each Employee - LeetCode

```
/* Write your T-SQL query statement below */
WITH cte AS (
    SELECT
        reports_to,
        COUNT(reports_to) AS reports_count,
        ROUND(SUM(age * 1.0) / COUNT(age), 0) AS average_age
    FROM Employees
    WHERE reports_to IS NOT NULL
    GROUP BY reports_to
)

SELECT e.employee_id, e.name, cte.reports_count, cte.average_age
FROM Employees e
JOIN cte ON cte.reports_to = e.employee_id;
```

31 Primary Department for Each Employee - LeetCode

```
WITH cte as (
SELECT DISTINCT
employee_id
,count(1) as cnt
FROM Employee
group by employee_id
)

Select DISTINCT cte.employee_id , e.department_id
from cte
left join Employee e
on cte.employee_id = e.employee_id
where primary_flag = 'Y' or cnt = 1
```

32 Triangle Judgement - LeetCode

```
SELECT
x,y,z,
CASE when x+y>z and y+z>x and z+x> y then 'Yes' else 'No' end as
triangle
from triangle
```

33 Consecutive Numbers - LeetCode

```
SELECT DISTINCT num as 'ConsecutiveNums'
,COUNT(1) as cnt
        FROM (
                SELECT id, num,
                                ROW_NUMBER() OVER (ORDER BY id)
                                - ROW_NUMBER() OVER (PARTITION BY num
ORDER BY id) as 'ConsecutiveGroup'
                FROM Logs
        )T
        GROUP BY num, ConsecutiveGroup
        HAVING COUNT(1) >= 3
```

APPROACH_2:

```
WITH LogDetails AS (
   SELECT LAG(num) OVER (ORDER BY id) AS PrevNum
      ,num AS CurrentNum
      ,LEAD(num) OVER (ORDER BY id) AS NextNum
   FROM Logs
)

SELECT DISTINCT CurrentNum AS ConsecutiveNums
FROM LogDetails
WHERE (PrevNum = CurrentNum
     AND CurrentNum = NextNum)
```

34 Count Salary Categories - LeetCode

```
select
'Low Salary' as category
, sum(iif(income < 20000, 1, 0)) as accounts_count
from Accounts
union all
select
'Average Salary' as category
, sum(iif(income >= 20000 and income <= 50000, 1, 0)) as
accounts_count
from Accounts
union all
select
'High Salary' as category
, sum(iif(income > 50000, 1, 0)) as accounts_count
from Accounts
```

35[Last Person to Fit in the Bus - LeetCode]

```
With cte as(
SELECT *  , sum(weight) over(order by turn) as r_weight
from
Queue)
Select top 1 person_name from cte where r_weight <= 1000
order by r_weight DESC
```

36

[Investments in 2016 - LeetCode]

```
WITH CTE AS (
        SELECT
            *,
            COUNT(lat) OVER(PARTITION BY lat,lon)
CountLatLon,
            COUNT(tiv_2015) OVER(PARTITION BY tiv_2015)
CountT_2015
        FROM
            Insurance T1
)
SELECT
    ROUND(SUM(tiv_2016),2) tiv_2016
FROM
    CTE
WHERE
    CountLatLon = 1
AND
    CountT_2015 > 1
```

SUBQUERY

37 Employees Whose Manager Left the Company - LeetCode

```
SELECT employee_id from
(SELECT
* , case when manager_id in (Select distinct employee_id from
Employees) or manager_id is null then 'Yes' else 'No' end as flag
FROM
Employees e
where salary < 30000
) a
where flag = 'No'
order by employee_id
```

38Exchange Seats - LeetCode

```
SELECT CASE
```

```
        WHEN ID % 2 = 1 THEN LEAD(ID, 1, ID) OVER (ORDER BY ID
ASC)
        WHEN ID % 2 = 0 THEN ID - 1
    END AS ID, STUDENT
FROM SEAT
ORDER BY ID
```

39 Movie Rating - LeetCode

```
--2 results

--1st result
-- 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
with userid_rating as (
Select
user_id , count(1) as cnt
from MovieRating
group by user_id
)
, res1 as(
Select top 1 u.name as name , ur.user_id as id
from userid_rating ur left join Users u
on ur.user_id = u.user_id
order by ur.cnt desc , name asc
)



--2nd result Find the movie name with the highest average rating in February 2020. In case of
a tie, return the lexicographically smaller movie name.
---choose movie whch has been rated by person in res 1

-- avg ratings
, avg_ratings as (
SELECT TOP 1 Movies.title
, a.avg_rating
FROM Movies
LEFT JOIN (
   SELECT movie_id,
        ROUND(SUM(rating)*1.0/COUNT(movie_id), 2) AS avg_rating
   FROM MovieRating
   WHERE created_at BETWEEN '2020-02-01' AND '2020-02-28'
   GROUP BY movie_id
) AS a ON Movies.movie_id = a.movie_id
```

```
order by avg_rating desc , title asc
)

Select name as results from res1
UNION ALL
select avg_ratings.title as results  from avg_ratings
```

40Friend Requests II: Who Has the Most Friends - LeetCode

/* Write your T-SQL query statement below */

```
--request count
with request_count as (
Select  requester_id  , count( requester_id ) as cnt1
from RequestAccepted rq
group by requester_id
)


---accept_count
,accept_count as(
Select  accepter_id  , count( accepter_id ) as cnt2
from RequestAccepted rq
group by accepter_id
)

, res as(
select * from request_count
union ALL
Select * from accept_count
)



,final_res as (
Select requester_id , sum(cnt1) as xcnt
from res
group by requester_id

)

Select requester_id as id , xcnt as num from final_res
where xcnt = (SELECT max(xcnt) from final_res)
                `
```

41Department Top Three Salaries - LeetCode

```
with cte as (
Select * from
(SELECT
* , dense_rank()over(partition by departmentId order by salary desc)
rn
FROM
Employee e

)a
where a.rn<=3
)

Select d.name as  Department , cte.name as Employee , cte.salary
as Salary
from cte
left join Department d
on d.id = cte.departmentId
```

42 Fix Names in a Table - LeetCode

```
SELECT user_id,
    UPPER(LEFT(name,1)) + LOWER(RIGHT(name,len(name)-1)) AS
name
FROM Users
ORDER BY user_id
```

43Patients With a Condition - LeetCode

```
select *
from Patients
where conditions like '% DIAB1%'
or conditions like 'DIAB1%'
```

44 Delete Duplicate Emails - LeetCode

```
/* Write your T-SQL query statement below */
DELETE
```

```
FROM Person
Where id not in (Select min(id) from Person group by email  )
;
```

OTHER APPROACHES:

[Different ways to SQL delete duplicate rows from a SQL Table (sqlshack.com)]

45 Kth Highest Salary

[Second Highest Salary - LeetCode]

```
with temp as (
    select salary, dense_rank() over(order by salary desc) as rnk
    from employee
)

select max(salary) as secondHighestSalary from temp where rnk = 2
```

46[Group Sold Products By The Date - LeetCode]

```
Select
sell_date
,count(sell_date) as num_sold
,STRING_AGG (DISTINCT  product , ',' ) as  products
from Activities a
group by sell_date
```

47[List the Products Ordered in a Period - LeetCode]

```
With cte as(
SELECT
product_id ,
sum(unit)  as total
from Orders
where order_date between '2020-02-01' and '2020-02-29'
```

```
group by product_id
)

Select p.product_name , cte.total as unit
from cte
left join Products p
on cte.product_id = p.product_id
where total >= 100
```

48[Find Users With Valid E-Mails - LeetCode]

```
/* Write your T-SQL query statement below */
SELECT
    *
FROM
    Users
WHERE
    mail LIKE '[a-Z]%@leetcode.com'
    AND SUBSTRING(mail, 1, LEN(mail) - 13) NOT LIKE
'%[^0-9a-Z_.-]%'
```

49 [Product Price at a Given Date - LeetCode]

```
SELECT
    product_id,
    FIRST_VALUE(new_price) OVER (PARTITION BY product_id
ORDER BY change_date DESC) AS price
FROM Products
WHERE change_date <= '2019-08-16'
UNION
SELECT
    product_id,
    10 AS price
FROM Products
GROUP BY product_id
HAVING MIN(change_date) > '2019-08-16'
```

50[Restaurant Growth - LeetCode]

```
/* Write your T-SQL query statement below */
SELECT
    visited_on,
    SUM(SUM(amount)) OVER (ORDER BY visited_on ROWS
BETWEEN 6 PRECEDING AND CURRENT ROW) as amount,
    ROUND(AVG(SUM(amount*1.0)) OVER (ORDER BY visited_on
ROWS BETWEEN 6 PRECEDING AND CURRENT ROW), 2) as
average_amount

FROM Customer

GROUP BY visited_on

ORDER BY visited_on

OFFSET 6 ROWS
```

EXTRA//