Leetcode Basic SQL 50 :

1757 : **Recyclable and Low Fat Products**

```
SELECT product_id
FROM Products
WHERE low_fats='Y' and recyclable='Y';
```

584: **Find Customer Referee**

```
SELECT name
FROM Customer
WHERE referee_id is null or referee_id <>2
```

595: **Big Countries**

```
SELECT name, population, area
FROM world
WHERE area >= 3000000 or population >= 25000000
```

1148: **Article Views 1**

```
SELECT distinct author_id as id
FROM Views
WHERE author_id = viewer_id
ORDER BY author_id
```

1683: **Invalid Tweets**

```
SELECT tweet_id
FROM Tweets
WHERE length(content)>15
```

BASIC JOINS :

1378: Replace Employee ID With The Unique Identifier

```
SELECT unique_id, name
FROM
Employees
LEFT JOIN
EmployeeUNI
on
Employees.id = EmployeeUNI.id
```

1068: Product Sales Analysis 1

```
SELECT product_name, year, price
FROM
Sales
LEFT JOIN
Product
on
Sales.product_id = Product.product_id
```

1581: Customer Who Visited but Did Not Make Any Transactions:

```
select customer_id, count(customer_id) as count_no_trans
from Visits as V
left join Transactions as T
on V.visit_id = T.visit_id
where T.transaction_id is null
Group by customer_id
```

197: Rising Temperature

Method 1:

```
SELECT a.id
FROM Weather a,Weather b
WHERE a.Temperature > b.Temperature
AND DATEDIFF(a.Recorddate, b.Recorddate) = 1
```

Method 2:

```
WITH CTE AS (
SELECT *, DATE_ADD(recordDate,INTERVAL -1 DAY) AS yesterdays_date,
LAG(recordDate) over (order by recordDate) AS previous_record_date,
LAG(temperature) over (order by recordDate) AS previous_temperature
FROM Weather )

SELECT id
FROM CTE
WHERE yesterdays_date=previous_record_date
AND temperature>previous_temperature
```

Method 3:

```
WITH CTE AS
(SELECT *,
LAG(temperature) over (order by recordDate) AS previous_temperature
FROM Weather )

SELECT
id
FROM CTE
WHERE temperature>previous_temperature
```

1661: **Average Time of Process Per Machine**

Method 1:

```
SELECT a1.machine_id, Round(AVG(a2.timestamp-a1.timestamp),3) as processing_time
FROM activity a1
INNER join Activity a2
on a1.machine_id=a2.machine_id
AND a1.process_id = a2. process_id
AND a1.activity_type='start' and a2.activity_type='end'
group by machine_id
```

Method 2: (does not pass all test cases)

```
WITH CTE AS
(
SELECT *, lag(timestamp) over (order by machine_id) AS new
FROM Activity
)
SELECT machine_id, ROUND(AVG(timestamp-new),3) AS processing_time
```

```
FROM CTE
WHERE activity_type='end'
Group by machine_id
```

577: Employee Bonus

```
SELECT e.name,b.bonus
FROM Employee e
LEFT JOIN BONUS b
ON
e.empId=b.empId
WHERE bonus < 1000 or bonus is null
```

1280: Students and Examinations

```
SELECT st.student_id,st.student_name,su.subject_name,count(ex.student_id) as attended_exams
FROM Students st
CROSS JOIN
Subjects su
LEFT JOIN
Examinations ex
ON
st.student_id = ex.student_id
AND su.subject_name = ex.subject_name
GROUP BY st.student_name, su.subject_name
ORDER BY st.student_id, su.subject_name
```

570: Managers with at Least 5 Direct Reports

```
WITH CTE AS
(
SELECT e1.*,COUNT(e1.id) AS manager_count
FROM Employee e1
INNER JOIN Employee e2
ON e1.id = e2.managerId
GROUP BY e1.id
)
SELECT name
FROM CTE
```

```
WHERE manager_count >=5
```

1934: Confirmation Rate

```
SELECT s.user_id, IFNULL(Round(SUM(action='confirmed')/COUNT(*),2),0.00) AS
confirmation_rate
FROM Signups s
LEFT JOIN Confirmations c
on s.user_id = c.user_id
GROUP BY s.user_id
```

BASIC AGGREGATE FUNCTIONS:

620: Not Boring Movies

```
SELECT *
FROM CINEMA
WHERE id%2 = 1 and description <> 'boring'
ORDER BY rating DESC;
```

1251: Average Selling Price

```
SELECT p.product_id, IFNULL(ROUND(SUM(p.price*u.units)/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 >= p.start_date
and u.purchase_date <= p.end_date
GROUP BY p.product_id
```

Below code doesn't pass all test cases upon using WHERE:

```
SELECT p.product_id, IFNULL(ROUND(SUM(p.price*u.units)/SUM(u.units), 2), 0) as
average_price
FROM Prices p
LEFT JOIN UnitsSold u
ON p.product_id = u.product_id
WHERE u.purchase_date >= p.start_date
```

```
and u.purchase_date <= p.end_date
GROUP BY p.product_id
```

1075: Project Employees I

```
SELECT p.project_id,ROUND(AVG(e.experience_years),2) AS average_years
FROM Project p
INNER JOIN
Employee e
ON
p.employee_id = e.employee_id
GROUP BY project_id
```

1633: Percentage of Users Attended a Contest

```
SELECT contest_id, Round((COUNT(user_id))*100/(SELECT COUNT(user_id) FROM
Users),2) AS percentage
FROM Register
GROUP BY contest_id
ORDER BY percentage DESC,contest_id
```

1211: Queries Quality and Percentage.

```
SELECT query_name, ROUND(AVG(rating/position),2) AS quality,
ROUND(AVG(IF(rating<3,1,0) *100),2) AS poor_query_percentage
FROM Queries
WHERE Query_name is not null                    //Important line of code
GROUP BY query_name
```

1193: Monthly Transaction I

```
WITH CTE AS(
SELECT *,
   DATE_FORMAT(trans_date, '%Y-%m') AS month

FROM
   Transactions
```

```
)
select 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 CTE
GROUP BY
month, country
```

1174: Immediate Food Delivery II

```
WITH CTE AS
(
SELECT *,
rank() over (partition by customer_id order by order_date) rk ,
(CASE WHEN order_date=customer_pref_delivery_date THEN 1 else 0 END) AS col
FROM Delivery
)
SELECT ROUND((SUM(CASE WHEN col=1 THEN 1 else 0
END)/count(customer_id))*100,2) AS immediate_percentage
FROM CTE
WHERE rk=1
```

550: Game Play Analysis IV

```
WITH CTE AS
(
SELECT player_id,device_id,event_date,
datediff(event_date,min(event_date) over (partition by player_id))=1 as first_login
FROM Activity
)
SELECT ROUND(SUM(first_login)/COUNT(distinct player_id),2) AS fraction
 FROM CTE
```

SORTING AND GROUPING:

2356: Number of Unique Subjects Taught by Each Teacher

```
SELECT teacher_id, COUNT(DISTINCT subject_id) AS cnt
FROM teacher
GROUP BY teacher_id
```

1141: User Activity for the Past 30 Days I

```
SELECT activity_date as day, COUNT(distinct user_id) AS active_users
FROM Activity
Where activity_date>='2019-06-28' AND activity_date<='2019-07-27'
GROUP BY activity_date
```

1070: Product Sales Analysis III

```
WITH CTE AS
(
SELECT *,
rank() over (partition by product_id order by year) RK
 FROM SALES
)
SELECT
CTE.product_id, CTE.year as first_year,CTE.quantity,CTE.price
FROM CTE
LEFT JOIN Product p
ON CTE.Product_id = p.product_id
WHERE RK =1
```

596: Classes More Than 5 Students

```
WITH CTE AS(
SELECT class,count(student) AS CNT
FROM Courses
group by class)

SELECT class
FROM CTE
WHERE CNT >=5
```

1729: Find Followers Count

```
SELECT user_id, COUNT(follower_id) AS followers_count
FROM Followers
Group BY user_id
ORDER BY user_id
```

619: Biggest Single Number

```
SELECT MAX(num) AS num
FROM MyNumbers
WHERE NUM IN
(SELECT num
FROM MyNumbers
GROUP BY num
HAVING count(num)=1)
```

1045: Customers Who Bought All Products

```
SELECT customer_id FROM Customer
GROUP BY customer_id
HAVING COUNT(DISTINCT product_key) = (SELECT COUNT(distinct product_key) FROM
Product)
```

ADVANCED SELECT AND JOINS:

1731: The Number of Employees Which Report to Each Employee

```
SELECT e1.employee_id,e1.name,COUNT(e1.employee_id) AS
reports_count,ROUND(AVG(e2.age)) as average_age
FROM Employees e1
JOIN Employees e2
ON e1.employee_id = e2.reports_to
GROUP BY e1.employee_id
ORDER BY employee_id ASC
```

1789: Primary Department for Each Employee

```
WITH CTE AS
(
```

```
SELECT employee_id,department_id
FROM Employee e
WHERE e.primary_flag='Y'
UNION
SELECT employee_id,department_id
FROM Employee e
GROUP BY employee_id
HAVING count(employee_id)=1
)
SELECT * FROM CTE
order by employee_id
```

610: Triangle Judgement

```
SELECT *,
(CASE WHEN x+y>z AND y+Z>x AND x+z>y THEN 'Yes' ELSE 'No' END) AS triangle
FROM Triangle
```

180: Consecutive Numbers

```
# Write your MySQL query statement below
SELECT distinct(a.num) as ConsecutiveNums FROM Logs a #distinct is very imp
INNER JOIN Logs b
ON a.id+1=b.id AND a.num=b.num                    #IMP line of code
INNER JOIN Logs c
ON a.id+2=c.id AND a.num=c.num
```

1164: Product Price at a Given Date

```
WITH CTE AS
(
SELECT product_id, new_price as price
FROM Products
WHERE(product_id, change_date) IN (
    SELECT product_id, MAX(change_date)
    FROM Products
    WHERE change_date  <='2019-08-16'
    GROUP BY product_id
)
UNION
```

```
SELECT product_id,10 as price
FROM Products
WHERE Product_id NOT IN
(
   SELECT distinct(Product_id)
   FROM Products
   WHERE change_date<='2019-08-16'
   GROUP BY Product_id
)
)
SELECT * FROM CTE
order by product_id
```

1204. Last Person to Fit in the Bus

```
WITH CTE AS
(
SELECT turn,person_id,person_name,weight,
SUM(weight) over (order by turn) total_weight
FROM QUEUE
)

SELECT person_name FROM CTE
WHERE total_weight<=1000
ORDER BY total_weight DESC
limit 1
```

1907. Count Salary Categories

```
SELECT
'Low Salary' as category,
Count(account_id) as accounts_count
FROM Accounts
WHERE income < 20000

UNION

SELECT
'Average Salary' as category,
Count(account_id) as accounts_count
FROM Accounts
WHERE income >= 20000 AND income <=50000

UNION
```

```
SELECT
'High Salary' as category,
Count(account_id) as accounts_count
FROM Accounts
WHERE income >50000
```

SubQueries :

1978. Employees Whose Manager Left the Company

```
SELECT e1.employee_id
FROM Employees e1
LEFT JOIN Employees e2 ON
e1.manager_id = e2.employee_id
WHERE e1.salary<30000 and e2.employee_id is null and e1.manager_id is not null
ORDER BY employee_id
```

---

626. Exchange Seats

```
SELECT

(CASE
WHEN id=(SELECT MAX(id) FROM SEAT) AND id%2=1 THEN id
WHEN id%2=1 THEN id+1 else id-1 END) As id,
student
FROM Seat
Order by id
```

1341. Movie Rating

```
(Select
(u.name) as results
FROM MovieRating mr
JOIN users u
ON mr.user_id=u.user_id
GROUP BY u.name
order by COUNT(u.name) DESC,name
LIMIT 1)
UNION ALL                #imp line of code, movie name and user name can be same
(SELECT m.title AS results
```

```
FROM MovieRating mr
JOIN Users u
ON mr.user_id=u.user_id
JOIN movies m
ON mr.movie_id=m.movie_id
WHERE created_at>='2020-02-01' AND created_at<='2020-02-29'
GROUP BY title
ORDER BY AVG(mr.rating) DESC,m.title ASC
LIMIT 1
)
```

1321. Restaurant Growth

```
# https://www.geeksforgeeks.org/sql-rows-between/
# above is important link to study

WITH CTE AS
(
SELECT visited_on,
sum(amount) as total_amount
from customer
group by visited_on
),
CTE2 AS
(
select visited_on, SUM(total_amount) over (order by visited_on rows between 6 preceding and
current row) as amount ,Round(AVG(total_amount) over (order by visited_on rows between 6
preceding and current row),2) as average_amount
from cte
)
SELECT * FROM CTE2
WHERE visited_on >= (
   SELECT DATE_add(MIN(visited_on),interval 6 day)
   FROM cte2
   )
```

---

602. Friend Requests II: Who Has the Most Friends

```
# Write your MySQL query statement below
WITH CTE AS
(
SELECT requester_id as id
FROM RequestAccepted
```

```
UNION ALL
SELECT accepter_id as id
FROM RequestAccepted
)
SELECT id,Count(id) AS num FROM CTE
group by id
Order by num DESC
LIMIT 1
```

---

585. Investments in 2016

```
SELECT ROUND(SUM(tiv_2016), 2) AS tiv_2016
FROM Insurance
WHERE tiv_2015 IN (
   SELECT tiv_2015
   FROM Insurance
   GROUP BY tiv_2015
   HAVING COUNT(*) > 1
)
AND (lat, lon) IN (
   SELECT lat, lon
   FROM Insurance
   GROUP BY lat, lon
   HAVING COUNT(*) = 1
)
```

---

185. Department Top Three Salaries

```
WITH CTE AS(
SELECT D.name,E.name as Employee, Salary,
dense_rank() over (partition by D.name order by SALARY DESC) AS rk
FROM Department D
JOIN Employee E
ON D.id=E.departmentID
)

SELECT name AS Department,Employee,Salary FROM CTE
WHERE rk<=3
ORDER by Employee
```

Advanced String Function :

1667. Fix Names in a Table

```
SELECT user_id,
Concat(upper(left(name,1)),lower(right(name,length(name)-1))) AS name
FROM USERS
order by user_id
```

1527. Patients With a Condition

```
SELECT *
FROM Patients
WHERE conditions LIKE 'DIAB1%' OR conditions LIKE '% DIAB1%'
```

---

196. Delete Duplicate Emails

```
DELETE P1
FROM Person P1
JOIN Person P2
ON P1.email=P2.email
AND P1.id>P2.id
```

---

176. Second Highest Salary

```
WITH CTE AS(
SELECT * ,
dense_rank() over (order by salary DESC) rk  #dense_rank works rank won't
FROM Employee
)
SELECT MAX(salary) AS SecondHighestSalary FROM CTE
WHERE rk=2
```

1484. Group Sold Products By The Date

```
SELECT sell_date, COUNT(distinct product) as num_sold,
GROUP_concat(distinct product order by product separator ',') AS products
FROM Activities
```

```
GROUP BY sell_date
ORDER BY sell_date
```

1327. List the Products Ordered in a Period

```
SELECT p.product_name,SUM(o.unit) AS unit
FROM Orders o
LEFT JOIN Products p
ON o.product_id=p.product_id
WHERE o.order_date>='2020-02-01' AND o.order_date<='2020-02-29'
GROUP BY p.product_name
HAVING SUM(o.unit)>=100
```

1517. Find Users With Valid E-Mails

```
SELECT *
FROM USERS
WHERE mail REGEXP '^[a-zA-Z][a-zA-Z0-9_.\-]*@leetcode[.]com$'
```
