-- BASIC SELECT AND WHERE
1. https://leetcode.com/problems/big-countries/
select name ,population,area from world where area>=3000000 or population>=25000000;

2. https://leetcode.com/problems/article-views-i/description/
select distinct(author_id) as id from views where author_id=viewer_id order by author_id;

-- CONDITION FILTERING
1. https://leetcode.com/problems/not-boring-movies/
SELECT * FROM Cinema WHERE
MOD(id, 2) = 1 AND description != 'boring'
ORDER BY rating DESC;

2. https://leetcode.com/problems/find-customer-referee/
select name from Customer where referee_id!=2 or referee_id is null;

3. https://leetcode.com/problems/triangle-judgement/
SELECT *, IF(x+y>z and y+z>x and z+x>y, "Yes", "No") as triangle FROM Triangle

4. https://leetcode.com/problems/recyclable-and-low-fat-products/
Select product_id from products where low_fats='Y' and recyclable='y';

5. https://leetcode.com/problems/calculate-special-bonus/
SELECT employee_id, IF(name NOT LIKE "M%" AND employee_id%2 <> 0, salary, 0) AS bonus FROM Employees ORDER BY employee_id

6. https://leetcode.com/problems/the-latest-login-in-2020/description/
select user_id, max(time_stamp) as last_stamp from logins where year(time_Stamp)='2020' group by user_id ;

-- BASIC JOINS
1. https://leetcode.com/problems/combine-two-tables/
select p.firstName, p.lastName, a.city, a.state from person p left join address a on p.personId=a.personId;

2. https://leetcode.com/problems/product-sales-analysis-i/description/
select p.product_name, s.year, s.price from Sales s inner join Product p on s.product_id=p.product_id;

3. https://leetcode.com/problems/employee-bonus/

select e.name, b.bonus from employee e left join bonus b on e.empId=b.empId where b.bonus<1000 or bonus is null;

4. https://leetcode.com/problems/replace-employee-id-with-the-unique-identifier/
select u.unique_id , e.name from employees e left join Employeeuni u on e.id=u.id;

5. https://leetcode.com/problems/project-employees-i/description/
SELECT p.project_id, ROUND(AVG(e.experience_years),2) AS average_years FROM Project p LEFT JOIN Employee e
ON p.employee_id = e.employee_id GROUP BY p.project_id

-- BASIC GROUPING
1. https://leetcode.com/problems/duplicate-emails/
select email from person group by email having count(email)>1 ;

2. https://leetcode.com/problems/customer-placing-the-largest-number-of-orders/
select customer_number from orders group by customer_number order by count(order_number) desc limit 1;

3. https://leetcode.com/problems/classes-more-than-5-students/description/
select class from courses group by class having count(student)>=5;

4. https://leetcode.com/problems/actors-and-directors-who-cooperated-at-least-three-times/
select actor_id,director_id from ActorDirector group by actor_id,director_id Having count(timestamp)>=3;

5. https://leetcode.com/problems/game-play-analysis-i/description/
select player_id, min(event_date) as first_login from activity group by player_id;

6. https://leetcode.com/problems/daily-leads-and-partners/description/
select date_id, make_name , count(distinct(lead_id)) as unique_leads, count(distinct(partner_id)) as unique_partners from DailySales GROUP BY date_id, make_name;

7. https://leetcode.com/problems/find-followers-count/
select user_id , count(follower_id) as followers_count from followers group by user_id order by user_id;

8. https://leetcode.com/problems/find-total-time-spent-by-each-employee/
select event_day as day, emp_id, sum(out_time-in_time) as total_time from employees group by emp_id, event_day;

9. https://leetcode.com/problems/odd-and-even-transactions/description/

```
SELECT transaction_date,
   SUM(CASE WHEN amount%2=1 THEN amount ELSE 0 END) as odd_sum,
   SUM(CASE WHEN amount%2=0 THEN amount ELSE 0 END) as even_sum
FROM transactions GROUP BY transaction_date ORDER BY transaction_date;
```

-- SELF JOIN

1. https://leetcode.com/problems/employees-earning-more-than-their-managers/

```
select e1.name as Employee from Employee e1 inner join Employee e2 where e1.Managerid =
e2.id and e1.salary> e2.salary;
```

2. https://leetcode.com/problems/rising-temperature/

```
SELECT w1.id FROM Weather w1, Weather w2 WHERE DATEDIFF(w1.recordDate,
w2.recordDate) = 1 AND w1.temperature > w2.temperature;
```

3. https://leetcode.com/problems/average-time-of-process-per-machine/

```
select a1.machine_id, round(avg(a2.timestamp-a1.timestamp), 3) as processing_time
from Activity a1 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 a1.machine_id
```

-- SUBQUERIES

1. https://leetcode.com/problems/customers-who-never-order/description/

```
SELECT name as Customers from Customerswhere id not in (
   select customerId from Orders );
```

2. https://leetcode.com/problems/biggest-single-number/

```
select max(num) as num from (
select num from MyNumbers group by num having count(num)=1) as nums ;
```

3. https://leetcode.com/problems/sales-person/description/

```
SELECT sp.name FROM SalesPerson sp WHERE sp.name NOT IN(
   SELECT sp.name FROM SalesPerson sp
      LEFT JOIN Orders o ON sp.sales_id = o.sales_id
      LEFT JOIN Company c ON o.com_id = c.com_id
   WHERE c.name = 'Red'
);
```

4. https://leetcode.com/problems/primary-department-for-each-employee/

```
SELECT DISTINCT employee_id, department_id FROM Employee
WHERE employee_id IN (
   SELECT employee_id FROM Employee GROUP BY employee_id HAVING COUNT(*) = 1)
OR primary_flag = 'Y' ORDER BY employee_id;
```

5. https://leetcode.com/problems/employees-whose-manager-left-the-company/description/

```
SELECT employee_id FROM Employees WHERE salary < 30000
AND manager_id NOT IN (SELECT DISTINCT employee_id FROM Employees)  ORDER BY
employee_id
```

6. https://leetcode.com/problems/managers-with-at-least-5-direct-reports/

```
SELECT name FROM Employee WHERE id IN ( SELECT managerId  FROM Employee
GROUP BY managerId HAVING COUNT(*) >= 5)
```

7. https://leetcode.com/problems/investments-in-2016/solutions/3670790/best-optimum-solution-with-explanation/

```
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)
```

8. https://leetcode.com/problems/customers-who-bought-all-products/description/

```
select customer_id from Customer group by customer_id
having count(distinct product_key) = (select count(distinct product_key) from Product)
```

9. https://leetcode.com/problems/product-sales-analysis-iii/description/

```
select product_id, year as first_year, quantity, price
from Sales where (product_id, year) in (
   select product_id, min(year) from Sales group by product_id)
```

10. https://leetcode.com/problems/immediate-food-delivery-ii/description/

```
select ROUND(AVG(order_date = customer_pref_delivery_date) * 100, 2) as
immediate_percentage
from delivery where (customer_id, order_date) IN
(select customer_id, min(order_date) as first_order
from delivery
```

group by customer_id)

-- DELETE, UPDATE, REFORMAT
1. https://leetcode.com/problems/delete-duplicate-emails/

```
delete p1 from person p1,person p2
where p1.email=p2.email and p1.id>p2.id;
```

2. https://leetcode.com/problems/swap-salary/

```
UPDATE salary SET sex = CASE sex
   WHEN 'm' THEN 'f'
   ELSE 'm'
END;
```

3. https://leetcode.com/problems/reformat-department-table/

```
SELECT
   id,
   sum( if( month = 'Jan', revenue, null ) ) AS Jan_Revenue,
   sum( if( month = 'Feb', revenue, null ) ) AS Feb_Revenue,
   sum( if( month = 'Mar', revenue, null ) ) AS Mar_Revenue,
   sum( if( month = 'Apr', revenue, null ) ) AS Apr_Revenue,
   sum( if( month = 'May', revenue, null ) ) AS May_Revenue,
   sum( if( month = 'Jun', revenue, null ) ) AS Jun_Revenue,
   sum( if( month = 'Jul', revenue, null ) ) AS Jul_Revenue,
   sum( if( month = 'Aug', revenue, null ) ) AS Aug_Revenue,
   sum( if( month = 'Sep', revenue, null ) ) AS Sep_Revenue,
   sum( if( month = 'Oct', revenue, null ) ) AS Oct_Revenue,
   sum( if( month = 'Nov', revenue, null ) ) AS Nov_Revenue,
   sum( if( month = 'Dec', revenue, null ) ) AS Dec_Revenue
FROM Department GROUP BY id;
```

4. https://leetcode.com/problems/group-sold-products-by-the-date/

```
select sell_date, count( DISTINCT product ) as num_sold ,
GROUP_CONCAT( DISTINCT product order by product ASC separator ',' ) as products
FROM Activities GROUP BY sell_date order by sell_date ASC;
```

5. https://leetcode.com/problems/tree-node/description/

```
SELECT id, CASE WHEN p_id IS NULL THEN 'Root'
            WHEN id IN (SELECT DISTINCT p_id FROM Tree) THEN 'Inner'
            ELSE 'Leaf' END AS type
FROM Tree
```

6. https://leetcode.com/problems/capital-gainloss/

```
select stock_name
, sum(if(operation = 'Buy', -1, 1) * price) as capital_gain_loss
from stocks
group by stock_name
```

-- JOIN WITH CONDITIONS/GROUP by having

1. https://leetcode.com/problems/sales-analysis-iii/description/

```
SELECT Product.product_id, Product.product_name FROM Product JOIN Sales ON
Product.product_id = Sales.product_id
GROUP BY Sales.product_id HAVING MIN(Sales.sale_date) >= "2019-01-01" AND
MAX(Sales.sale_date) <= "2019-03-31";
```

2. https://leetcode.com/problems/average-selling-price/

```
SELECT p.product_id, IFNULL(ROUND(SUM(units*price)/SUM(units),2),0) AS
average_price FROM Prices p LEFT JOIN UnitsSold u
ON p.product_id = u.product_id where u.purchase_date BETWEEN start_date AND end_date
group by product_id;
```

3. https://leetcode.com/problems/user-activity-for-the-past-30-days-i/description/

```
SELECT activity_date AS day, COUNT(DISTINCT user_id) AS active_users FROM Activity
WHERE (activity_date > "2019-06-27" AND activity_date <= "2019-07-27") GROUP BY
activity_date;
```

4. https://leetcode.com/problems/students-and-examinations/

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

5. https://leetcode.com/problems/list-the-products-ordered-in-a-period/description/

```
SELECT p.product_name AS product_name, sum(o.unit) AS unit FROM Products p JOIN
Orders o USING (product_id)
WHERE YEAR(o.order_date)='2020' AND MONTH(o.order_date)='02' GROUP BY
p.product_id HAVING SUM(o.unit)>=100
```

6. https://leetcode.com/problems/top-travellers/

```
select u.name, if(sum(r.distance) is null, 0, sum(r.distance)) as travelled_distance
from Users u left join Rides r on u.id = r.user_id group by u.id
order by travelled_distance desc ,u.name
```

7. https://leetcode.com/problems/the-number-of-employees-which-report-to-each-employee/

```
SELECT emp1.employee_id, emp1.name, COUNT(emp2.employee_id) AS reports_count,
ROUND(AVG(emp2.age)) AS average_age
FROM Employees emp1 INNER JOIN Employees emp2 ON emp1.employee_id =
emp2.reports_to
GROUP BY emp1.employee_id ORDER BY emp1.employee_id
```

8.
https://leetcode.com/problems/customer-who-visited-but-did-not-make-any-transactions/description/

```
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;
```

9. https://leetcode.com/problems/bank-account-summary-ii/description/

```
select u.name , sum(t.amount) as balance from users u inner join transactions t on
u.account=t.account group by name having balance>10000;
```

10. https://leetcode.com/problems/percentage-of-users-attended-a-contest/

```
select contest_id, round(count(distinct user_id) * 100 /(select count(user_id) from Users) ,2) as
percentage
from  Register group by contest_id order by percentage desc,contest_id
```

11. https://leetcode.com/problems/consecutive-numbers/

```
SELECT DISTINCT l1.num AS ConsecutiveNums FROM Logs l1
JOIN Logs l2 ON l1.id = l2.id - 1 JOIN Logs l3 ON l1.id = l3.id - 2
WHERE l1.num = l2.num AND l2.num = l3.num;
```

12. https://leetcode.com/problems/game-play-analysis-iv/description/

```
WITH temp AS (
   SELECT player_id, MIN(event_date) AS first_login_date FROM Activity GROUP BY
player_id
)
SELECT
   ROUND(
```

```
      SUM(DATEDIFF(a.event_date, t.first_login_date) = 1) / COUNT(DISTINCT a.player_id),
2
   ) AS fraction
FROM Activity a JOIN temp t ON a.player_id = t.player_id;
```

13.https://leetcode.com/problems/market-analysis-i/

```
SELECT u.user_id as buyer_id, u.join_date, count(o.order_id) as 'orders_in_2019'
FROM users u LEFT JOIN Orders o ON o.buyer_id=u.user_id  GROUP BY u.user_id HAVING
YEAR(order_date)='2019'
```

14.https://leetcode.com/problems/confirmation-rate/description/

```
SELECT s.user_id , ROUND(AVG(IF(c.action="confirmed",1,0)),2) AS confirmation_rate
FROM Signups s LEFT JOIN Confirmations c ON s.user_id = c.user_id GROUP BY s.user_id;
```

-- USING SUM AS COUNTING SPECIFIC QUERIES

1. https://leetcode.com/problems/queries-quality-and-percentage/

```
select query_name, round(avg(cast(rating as decimal) / position), 2) as quality,
round(sum(case when rating < 3 then 1 else 0 end) * 100 / count(*), 2) as
poor_query_percentage
from queries group by query_name;
```

2. https://leetcode.com/problems/monthly-transactions-i/description/

```
SELECT
   LEFT(trans_date, 7) AS month,
   country,
   COUNT(id) AS trans_count,
   SUM(state = 'approved') AS approved_count,
   SUM(amount) AS trans_total_amount,
   SUM((state = 'approved') * amount) AS approved_total_amount
FROM Transactions GROUP BY month, country;
```

3. https://leetcode.com/problems/trips-and-users/

```
SELECT
   T.request_at AS 'Day',
   ROUND(
      1-( SUM(T.status='completed')/COUNT(T.id) )
   ,2) AS 'Cancellation Rate'
FROM Trips T JOIN Users C ON T.client_id=C.users_id
JOIN Users D ON T.driver_id=D.users_id
```

WHERE D.banned='No' AND C.banned='No' GROUP BY T.request_at HAVING T.request_at BETWEEN "2013-10-01" AND "2013-10-03"

-- STRINGS
1. https://leetcode.com/problems/find-users-with-valid-e-mails/
```
SELECT * FROM Users WHERE mail REGEXP
'^[A-Za-z][A-Za-z0-9_\.\-]*@leetcode(\\?com)?\\.com$';
```

2. https://leetcode.com/problems/patients-with-a-condition/
```
select patient_id,patient_name,conditions from Patients
where conditions like 'DIAB1%'  or  conditions like '% DIAB1%' ;
```

3.https://leetcode.com/problems/fix-names-in-a-table/description/
```
SELECT Users.user_id ,
CONCAT(UPPER(SUBSTR(Users.name,1,1)),LOWER(SUBSTR(Users.name,2))) AS name
FROM Users
ORDER BY Users.user_id ASC
```

4.https://leetcode.com/problems/fix-names-in-a-table/description/
```
SELECT Users.user_id ,
CONCAT(UPPER(SUBSTR(Users.name,1,1)),LOWER(SUBSTR(Users.name,2))) AS name
FROM Users
ORDER BY Users.user_id ASC
```

5. https://leetcode.com/problems/invalid-tweets/description/
```
select tweet_id from Tweets where CHAR_LENGTH(content) > 15;
```

6. https://leetcode.com/problems/find-valid-emails/
```
SELECT user_id, email FROM Users WHERE email LIKE '%.com'  AND email REGEXP
'^[a-zA-Z0-9_]+@[a-zA-Z]+\\.com$' ORDER BY user_id;
```

7. https://leetcode.com/problems/find-invalid-ip-addresses/description/
```
SELECT ip, COUNT(log_id) AS invalid_count
FROM (
   SELECT *,
      SUBSTRING_INDEX(ip, '.', 1) AS s1,
      SUBSTRING_INDEX(SUBSTRING_INDEX(ip, '.', 2), '.', -1) AS s2,
      SUBSTRING_INDEX(SUBSTRING_INDEX(ip, '.', 3), '.', -1) AS s3,
```

```
        SUBSTRING_INDEX(SUBSTRING_INDEX(ip, '.', 4), '.', -1) AS s4,
        LENGTH(ip) - LENGTH(REPLACE(ip, '.', '')) AS cnt
    FROM logs) tmp
WHERE s1 > 255 OR s2 > 255 OR s3 > 255 OR s4 > 255
OR LEFT(s1, 1) = 0 OR LEFT(s2, 1) = 0 OR LEFT(s3, 1) = 0 OR LEFT(s4, 1) = 0
OR cnt != 3 GROUP BY ip ORDER BY invalid_count DESC, ip DESC

-- UNION/UNION ALL
```
1. https://leetcode.com/problems/rearrange-products-table/
```
SELECT product_id, 'store1' AS store, store1 AS price FROM Products WHERE store1 IS NOT
NULL
UNION
SELECT product_id, 'store2', store2 FROM Products WHERE store2 IS NOT NULL
UNION
SELECT product_id, 'store3', store3 FROM Products WHERE store3 IS NOT NULL
```

2. https://leetcode.com/problems/employees-with-missing-information/
```
SELECT T.employee_id FROM
  (SELECT * FROM Employees LEFT JOIN Salaries USING(employee_id)
   UNION
   SELECT * FROM Employees RIGHT JOIN Salaries USING(employee_id))
AS T WHERE T.salary IS NULL OR T.name IS NULL ORDER BY employee_id;
```

3. https://leetcode.com/problems/friend-requests-ii-who-has-the-most-friends/description/
```
SELECT id, COUNT(*) AS num
FROM (
    SELECT requester_id AS id FROM RequestAccepted
    UNION ALL
    SELECT accepter_id FROM RequestAccepted
) AS friends_count
GROUP BY id ORDER BY num DESC LIMIT 1;
```

4. https://leetcode.com/problems/movie-rating/
```
(SELECT name AS results
FROM MovieRating JOIN Users USING(user_id)
GROUP BY name
ORDER BY COUNT(*) DESC, name
LIMIT 1)

UNION ALL
```

```
(SELECT title AS results
FROM MovieRating JOIN Movies USING(movie_id)
WHERE EXTRACT(YEAR_MONTH FROM created_at) = 202002
GROUP BY title
ORDER BY AVG(rating) DESC, title
LIMIT 1);
```

5. https://leetcode.com/problems/count-salary-categories/

```
SELECT 'Low Salary' as category, COUNT(*) as accounts_count FROM Accounts WHERE income<20000
UNION
SELECT 'Average Salary', COUNT(*) FROM Accounts WHERE income BETWEEN 20000 AND 50000
UNION SELECT 'High Salary', COUNT(*) FROM Accounts WHERE income > 50000
```

-- RANKING

1. https://leetcode.com/problems/second-highest-salary/

```
WITH CTE AS
(SELECT Salary, DENSE_RANK () OVER (ORDER BY Salary desc) AS RANK_desc
                        FROM Employee)
SELECT MAX(salary) AS SecondHighestSalary FROM CTE WHERE RANK_desc = 2
```

2. https://leetcode.com/problems/rank-scores/

```
SELECT score, DENSE_RANK() OVER(ORDER BY score DESC) as 'rank'
FROM scores
```

3. https://leetcode.com/problems/department-highest-salary/description/

```
with new as
(select *, dense_rank() over(partition by departmentId order by salary desc) as rn from employee)
select d.name as department, new.name as employee , salary from new
join department d on d.id=new.departmentId where rn=1
```

4. https://leetcode.com/problems/product-price-at-a-given-date/description/

```
WITH RankedProducts AS (
   SELECT product_id, new_price,
     RANK() OVER(
       PARTITION BY product_id ORDER BY change_date DESC
```

```
    ) AS `rank`
  FROM Products WHERE change_date <= '2019-08-16'
 ),
 ProductToLatestPrice AS (
  SELECT product_id, new_price FROM RankedProducts WHERE `rank` = 1
 )
SELECT
 Products.product_id, IFNULL(ProductToLatestPrice.new_price, 10) AS price
FROM Products LEFT JOIN ProductToLatestPrice USING (product_id) GROUP BY
product_id;
```

5. https://leetcode.com/problems/department-top-three-salaries/

```
SELECT Department, Employee, Salary
FROM (
SELECT Emp.id,Emp.name AS Employee,Emp.salary AS Salary, Dept.name AS Department,
DENSE_RANK() over (PARTITION BY Emp.departmentId ORDER BY Emp.salary DESC)
AS Salary_Rank
FROM Employee as Emp LEFT JOIN Department Dept
ON Emp.departmentId = Dept.id)AS TMP
WHERE Salary_Rank <= 3
```

```
-- Window Function
```

1. https://leetcode.com/problems/exchange-seats/description/

```
SELECT
   id,
   CASE
      WHEN id % 2 = 0 THEN LAG(student) OVER(ORDER BY id)
      ELSE COALESCE(LEAD(student) OVER(ORDER BY id), student)
   END AS student
FROM Seat
```

2. https://leetcode.com/problems/last-person-to-fit-in-the-bus/description/

```
WITH CumulativeSum AS (
   SELECT person_name, SUM(weight) OVER (ORDER BY turn) AS cumulative_sum FROM
Queue
)
SELECT person_name FROM CumulativeSum WHERE cumulative_sum <= 1000 ORDER BY
cumulative_sum DESC LIMIT 1;
```

4. https://leetcode.com/problems/restaurant-growth/

```
SELECT visited_on, amount, ROUND(amount/7, 2) average_amount
FROM (
   SELECT DISTINCT visited_on,  SUM(amount) OVER(ORDER BY visited_on RANGE
BETWEEN INTERVAL 6 DAY   PRECEDING AND CURRENT ROW) amount,
   MIN(visited_on) OVER() 1st_date FROM Customer) t
WHERE visited_on>= 1st_date+6;
```

```
SELECT a.visited_on AS visited_on, SUM(b.day_sum) AS amount, ROUND(AVG(b.day_sum),
2) AS average_amount
FROM
  (SELECT visited_on, SUM(amount) AS day_sum FROM Customer GROUP BY visited_on )
a,
  (SELECT visited_on, SUM(amount) AS day_sum FROM Customer GROUP BY visited_on ) b
WHERE DATEDIFF(a.visited_on, b.visited_on) BETWEEN 0 AND 6
GROUP BY a.visited_on
HAVING COUNT(b.visited_on) = 7
```

5. https://leetcode.com/problems/find-students-who-improved/

```
WITH CTE AS(
   SELECT student_id, subject,
   FIRST_VALUE(score) OVER(PARTITION BY student_id, subject ORDER BY exam_date)
AS first_score,
   FIRST_VALUE(score) OVER(PARTITION BY student_id, subject ORDER BY exam_date
DESC) AS latest_score
   FROM Scores
)
SELECT DISTINCT * FROM CTE WHERE latest_score > first_score
```

6. https://leetcode.com/problems/human-traffic-of-stadium/description/

```
with q1 as (
select *,
    count(*) over( order by id range between current row and 2 following ) following_cnt,
    count(*) over( order by id range between 2 preceding and current row ) preceding_cnt,
    count(*) over( order by id range between 1 preceding and 1 following ) current_cnt
from stadium
where people > 99
)
select id, visit_date, people
from q1
```

```
where following_cnt = 3 or preceding_cnt = 3 or current_cnt = 3
order by visit_date
```