SELECT

1821. Find Customers With Positive Revenue this Year

```
SELECT
customer_id
FROM Customers
where year=2021 AND revenue>0
```

---

183. Customers Who Never Order

```
SELECT name as Customers
FROM customers
WHERE id NOT IN
(
 SELECT customerID FROM Orders
)
```

1873. Calculate Special Bonus

```
SELECT Employee_id,
(CASE WHEN employee_id%2=0 THEN 0
    WHEN name LIKE 'M%' THEN 0
    ELSE salary END) as bonus
FROM Employees
ORDER by employee_id
```

1398. Customers Who Bought Products A and B but Not C

```
WITH CTE AS
(
SELECT  o.customer_id,customer_name,
GROUP_concat(distinct product_name order by product_name separator ',') AS new_products
FROM orders o
LEFT JOIN Customers c
ON o.customer_id=c.customer_id
GROUP BY o.customer_id
)
SELECT customer_id,customer_name FROM CTE
Where new_products like "A,B%" AND new_products NOT LIKE "%C%"
```

1112. Highest Grade For Each Student

```
WITH CTE AS
(
SELECT *,
RANK() OVER (partition by student_id order by course_id) rk
FROM Enrollments
WHERE (student_id, grade) IN
(
SELECT student_id, Max(grade) FROM Enrollments
GROUP BY student_id
)
ORDER BY student_id
)
SELECT CTE.student_id,CTE.course_id,CTE.grade FROM CTE
Where rk=1
```

Basic Joins

175. Combine Two Tables

```
SELECT p.firstName,p.lastname,a.city,a.state
FROM Person p
LEFT JOIN Address a
ON p.personId = a.personId
```

1607. Sellers With No Sales

```
WITH CTE AS
 (
 SELECT DISTINCT(seller_id)
 FROM orders
 WHERE sale_date BETWEEN '2020-01-01' AND '2020-12-31'
 )
SELECT  s.seller_name
FROM seller s
LEFT JOIN CTE c
ON s.seller_id = c.seller_id
```

```
WHERE c.seller_id is null
ORDER BY s.seller_name
```

1407. Top Travellers

```
SELECT u.name,  IFnull(SUM(r.distance),0) AS travelled_distance
FROM Users u
LEFT JOIN Rides r
ON r.user_id = u.id
GROUP by u.id                    # Imp line of Code
ORDER by SUM(r.distance) DESC, name
```

---

607. Sales Person

```
WITH CTE as
(
SELECT s.name,c.name AS company_name,
IFNULL(GROUP_concat(distinct c.name order by c.name separator ','),'hgh') As boom
FROM SalesPerson s
LEFT JOIN  Orders o
ON o.sales_id = s.sales_id
LEFT JOIN Company c ON
o.com_id = c.com_id
GROUP BY name
)
SELECT name FROM CTE
WHERE BOOM NOT LIKE "RED%"
```

1440. Evaluate Boolean Expression

```
SELECT e.*,
(CASE WHEN operator = '>' AND v1.value>v2.value THEN 'true'
   WHEN operator = '<' AND v1.value<v2.value THEN 'true'
   WHEN operator = '=' AND v1.value=v2.value THEN 'true' else 'false'
END) AS value
FROM Expressions e
LEFT JOIN Variables v1
ON e.left_operand = v1.name
LEFT JOIN Variables v2
ON e.right_operand = v2.name
```

1212. Team Scores in Football Tournament

```
WITH CTE AS(
SELECT *,
(CASE WHEN t.team_id = m.host_team AND m.host_goals>m.guest_goals THEN 3
    WHEN t.team_id = m.guest_team AND m.guest_goals>m.host_goals THEN 3
    WHEN (t.team_id =m.host_team OR t.team_id=m.guest_team) AND
(m.host_goals=m.guest_goals) then 1
    else 0 END) As points
FROM TEAMS t
left JOIN Matches m
ON t.team_id = m.host_team OR t.team_id = m.guest_team
)
SELECT team_id, team_name, SUM(points) as num_points FROM CTE
group by team_id
ORDER BY num_points DESC, team_id
```

1890. The Latest Login in 2020

```
WITH CTE AS
(
SELECT user_id, time_stamp AS last_stamp,
rank() OVER (partition by user_id order by time_stamp DESC ) rk
FROM Logins
WHERE time_stamp >= '2020-01-01 00:00:00' AND time_stamp <= '2020-12-31 23:59:59'
)

SELECT user_id, last_stamp FROM CTe
WHERE rk=1
```

---

511. Game Play Analysis I

```
SELECT player_id,MIN(event_date) AS first_login
FROM activity
GROUP BY player_id
```

1571. Warehouse Manager

```
SELECT w.name as warehouse_name,SUM(w.units*p.width*p.length*p.height) as volume
FROM Warehouse w
LEFT JOIN Products p
ON w.product_id = p.product_id
group by (w.name)
```

---

586. Customer Placing the Largest Number of Orders

```
SELECT customer_number
FROM Orders
GROUP BY customer_number
ORDER BY Count(customer_number) DESC
LIMIT 1
```

1741. 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
```

1173. Immediate Food Delivery I

```
WITH CTE AS
(
SELECT *,
(CASE WHEN order_date=customer_pref_delivery_date THEN 1 else 0 End) AS val
FROM Delivery
)
SELECT Round((SUM(val)*100)/(SELECT count(delivery_id) FROM Delivery),2) AS
immediate_percentage
FROM CTE
```

1445. Apples & Oranges:

```
SELECT s1.sale_date, (s1.sold_num-s2.sold_num) AS diff
FROM Sales s1
JOIN Sales s2
```

```
ON s1.sale_date=s2.sale_date AND s1.fruit='apples' AND s2.fruit='oranges'
```

1699. Number of Calls Between Two Persons

```
WITH CTE AS
(
SELECT
CASE WHEN from_id<to_id THEN from_id else to_id END AS person1,
CASE WHEN from_id>to_id THEN from_id else to_id END AS person2,
duration
FROM Calls
)
SELECT person1,person2,COUNT(person1) AS call_count, SUM(duration) AS total_duration
FROM CTE
GROUP BY person1, person2
```

1587. Bank Account Summary II

```
SELECT u.name AS NAME, SUM(t.amount) AS BALANCE
FROM Transactions t
LEFT JOIN Users u on
t.account =u.account
GROUP BY u.name
HAVING SUM(t.amount)>10000
```

---

182. Duplicate Emails

```
SELECT Email from Person
GROUP BY email
Having count(email)>1
```

1050. Actors and Directors Who Cooperated At Least Three Times

```
SELECT actor_id, director_id
FROM ActorDirector
GROUP BY actor_id, director_id
HAVING COUNT(actor_id)>=3
```

1511. Customer Order Frequency

```
WITH CTE AS
(
SELECT o.customer_id,c.name, p.price * o.quantity as total_price,
DATE_FORMAT(order_date,"%Y-%m") AS date
FROM ORDERS o
JOIN Customers c
ON o.customer_id = c.customer_id
JOIN product p
ON o.product_id=p.product_id
WHERE order_date BETWEEN '2020-06-01' AND '2020-07-31'
),
CTE2 AS
(
SELECT customer_id,name,date,SUM(total_price)  FROM CTE
group by name , date
having SUM(total_price) >=100
)
SELECT customer_id,name FROM CTE2
group by name
having count(name)=2
```

1693. Daily Leads and Partners:

```
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
```

1495. Friendly Movies Streamed Last Month

```
SELECT distinct(c.title) AS TITLE
FROM Content c
LEFT JOIN TVProgram t ON
c.content_id = t.content_id
WHERE content_type = 'Movies' AND kids_content = 'Y' AND program_date BETWEEN '2020-06-
01'
AND '2020-06-30'
```

1501. Countries You Can Safely Invest In

```
SELECT co.name AS country
FROM Person p
LEFT JOIN Calls c ON
p.id=c.caller_id OR p.id = c.callee_id
LEFT JOIN Country co ON
LEFT(P.phone_number,3) = co.country_code
GROUP by co.name
HAVING AVG(c.duration) > (SELECT AVG(duration) FROM Calls)
```

---

603. Consecutive Available Seats

```
WITH CTE AS
(
SELECT c1.seat_id, c2.seat_id as id
FROM Cinema c1
JOIN Cinema c2 ON
c1.seat_id+1 =c2.seat_id
WHERE c1.free =1 AND c2.free = 1
),
CTE2 AS
(
SELECT seat_id FROM CTE
UNION
Select id as seat_id FROM CTE
)
SELECT * FROM CTE2
order by seat_id
```

1795. Rearrange Products Table

```
SELECT product_id, 'store1' AS store, store1 AS price
FROM Products
WHERE Store1 is NOT null
UNION
SELECT product_id, 'store2' AS store, store2 AS price
FROM Products
WHERE Store2 is NOT null
UNION
SELECT product_id, 'store3' AS store, store3 AS price
```

```
FROM Products
WHERE Store3 is NOT null
```

---

613. Shortest Distance in a Line

```
WITH CTE AS
(
SELECT
(CASE WHEN p1.x>p2.x THEN p1.x ELSE p2.x END) AS xone,
(CASE WHEN p1.x<p2.x THEN p1.x ELSE p2.x END) AS xtwo
FROM POINT p1
JOIN POINT p2
ON p1.x <> p2.x
)
SELECT MIN(xone-xtwo) AS shortest FROM CTE
```

1965. Employees With Missing Information

```
WITH CTE AS
(
SELECT employee_id FROM Employees
UNION ALL
SELECT employee_id FROM Salaries
)
SELECT * FROM CTE
GROUP BY Employee_id
HAVING count(employee_id)=1
ORDER BY employee_id
```

1264. Page Recommendations

```
WITH CTE AS
(
SELECT l.* FROM
Friendship f
 JOIN Likes l ON
f.user1_id = l.user_id OR f.user2_id = l.user_id
WHERE f.user1_id = 1 OR f.user2_id = 1
)
```

```
SELECT distinct(page_id) as recommended_page FROM CTE
WHERE page_id NOT IN (
                SELECT page_id
                FROM Likes
                WHERE user_id=1
               )
```

---

608. Tree Node

```
SELECT id,
(CASE WHEN p_id is null THEN 'Root'
    WHEN p_id is not null AND id  IN (SELECT distinct p_id FROM Tree) THEN 'Inner'
    ELSE 'Leaf' END) AS type
FROM Tree
```

---

534. Game Play Analysis III

```
SELECT player_id, event_date,
SUM(games_played) OVER (partition by player_id order by event_date) AS
games_played_so_far
FROM Activity
```

1783. Grand Slam Titles

```
WITH CTE AS
(
SELECT year, Wimbledon
FROM Championships
UNION ALL
SELECT year, Fr_open
FROM Championships
UNION ALL
SELECT year, US_open
FROM Championships
UNION ALL
SELECT year, Au_open
FROM Championships
)
SELECT p.*, COUNT(player_id) AS grand_slams_count FROM CTE
JOIN Players p  ON
```

```
CTE.Wimbledon = p.player_id
GROUP BY player_id
```

1747. Leetflex Banned Accounts

```
WITH CTE AS
(
SELECT l1.*,
(CASE WHEN l2.logout < l1.login THEN 'Not_Banned'
     WHEN l2.login  > l1.logout THEN 'Not_Banned'
     ELSE 'banned' END) AS Status
FROM LogInfo l1
JOIN LogInfo l2
ON l1.account_id=l2.account_id  AND l1.ip_address <> l2.ip_address AND l1.ip_address <
l2.ip_address
)
SELECT distinct(account_id)
FROM CTE
Where Status ='banned'
```

1350. Students With Invalid Departments

```
SELECT s.id,s.name
FROM Students s
LEFT JOIN Departments d
ON s.department_id = d.id
WHERE d.id is null
```

1303. Find the Team Size

```
WITH CTE AS(
SELECT team_id,COUNT(team_id) as count
FROM Employee
GROUP BY team_id
)
SELECT  e.employee_id, c.count as team_size
FROM EMPLOYEE e
JOIN CTE c
ON E.team_id = C.team_id
```

---

512. Game Play Analysis II:

```
WITH CTE AS
(
SELECT *,
Rank() over(partition by player_id order by event_date) rk
FROM activity
)
SELECT player_id,device_id FROM CTE
where rk=1
```

---

184. Department Highest Salary:

```
WITH CTE as
(
SELECT d.name as Department,e.name as Employee,E.salary,
dense_rank() over (partition by d.name order by salary desc) rk
FROM Department d
JOIN Employee e
ON d.id = e.departmentId
)
SELECT Department, Employee, salary FROM CTE
WHERE rk=1
```

1549. The Most Recent Orders for Each Product

```
WITH CTE AS
(
SELECT p.product_name,o.product_id,o.order_id,o.order_date,
dense_rank() over (partition by product_id order by order_date DESC) rk
FROM products p
LEFT JOIN Orders o
ON
p.product_id = o.product_id
WHERE o.product_id is not null
)
SELECT product_name,product_id,order_id,order_date FROM CTE
WHERE rk=1
Order by product_name,order_id
```

1532. The Most Recent Three Orders

```
WITH CTE AS
(
SELECT c.name as customer_name, c.customer_id, o.order_id,o.order_date,
rank() over (partition by c.customer_id order by order_date DESC) rk
FROM
Orders o JOIN
Customers c
ON o.customer_id = c.customer_id
)
SELECT customer_name,customer_id,order_id,order_date FROM CTE
where rk<=3
order by customer_name,customer_id,order_date desc
```

1831. Maximum Transaction Each Day

```
WITH CTE AS
(
SELECT transaction_id,
DATE_FORMAT(day,'%Y-%m-%d') AS new_day,
amount
FROM Transactions
),
CTE2 AS(
SELECT *,
rank() over (partition by new_day order by amount DESC) rk
FROM CTE
)
SELECT transaction_id FROM CTE2
WHERE rk=1
ORDER BY transaction_id
```

1077. Project Employees III

```
WITH CTE AS
(
SELECT p.*,e.experience_years,
dense_rank() over (partition by project_id order by experience_years DESC) rk
FROM project p
JOIN Employee e
ON p.employee_id = e.employee_id
)
SELECT project_id,employee_id FROM CTE
where rk=1
```

1285. Find the Start and End Number of Continuous Ranges

```
WITH CTE AS
(
SELECT log_id,
log_id-(Row_number() over (order by log_id) ) as diff
FROM Logs
)
SELECT  min(log_id) as start_id, max(log_id) as end_id FROM CTE
GROUP BY diff
Order by diff
```

1596. The Most Frequently Ordered Products for Each Customer

```
WITH CTE AS
(
SELECT o.customer_id,o.product_id, p.product_name,count(customer_id) as new FROM
ORDERS o
JOIN Products p
ON o.product_id = p.product_id
GROUP BY Customer_id,product_id
)
SELECT customer_id,product_id,product_name FROM CTE
WHERE (customer_id,new) IN
(SELECT customer_id,Max(new) FROM CTE
GROUP BY customer_id)
order by customer_id,product_id
```

1709. Biggest Window Between Visits:

```
WITH CTE AS
(
SELECT user_id,visit_date,
LEAD(visit_date,1,'2021-01-01') over (partition by user_id order by visit_date) as new
FROM UserVisits
ORDER BY user_id,visit_date
)
SELECT user_id, MAX(DateDiff(new,visit_date)) as biggest_window FROM CTE
group by user_id
order by user_id
```

1270. All People Report to the Given Manager

```
WITH recursive CTE as
(
   SELECT employee_id from Employees
   WHERE employee_id<>1 AND manager_id=1
   UNION
   SELECT e.employee_id From Employees e
   INNER JOIN CTE c
   ON c.employee_id = e.manager_id
)
SELECT * FROM CTE
```

1412. Find the Quiet Students in All Exams

```
WITH CTE AS
(
SELECT e.exam_id,e.score,s.*,
dense_rank() over (partition by exam_id order by score ) worst,
dense_rank() over (partition by exam_id order by score desc) best
FROM
Exam e JOIN Student s
ON e.student_id = s.student_id
), CTE2 AS
(
SELECT distinct(student_id),student_name FROM CTE
WHERE worst = 1 or best =1
UNION ALL
SELECT distinct(student_id),student_name FROM CTE
)
SELECT * FROM CTE2
group by student_id
having COUNT(student_id)=1
```

Order by student_id

1767. Find the Subtasks That Did Not Execute

```
with recursive cte as
(
select task_id, subtasks_count
FROM tasks
UNION
select task_id, subtasks_count-1
FROM CTE
WHERE subtasks_count>1
)
SELECT  cte.task_id, cte.subtasks_count as subtask_id FROM CTE
LEFT JOIN Executed e  ON
cte.task_id= e.task_id and cte.subtasks_count = e.subtask_id
WHERE e.subtask_id is null
order by cte.task_id, cte.subtasks_count
```

1225. Report Contiguous Dates

```
WITH CTE AS
(
SELECT fail_date,'failed' AS status
FROM failed
UNION
SELECT success_date as fail_date,'succeeded' AS status
FROM Succeeded
), CTE2 AS
(
SELECT *,
row_number() over (partition by status order by fail_date) rn
FROM CTE
WHERE fail_date BETWEEN '2019-01-01' AND '2019-12-31'
order by fail_date
), CTE3 AS
(
SELECT *, DATE_SUB(fail_date, INTERVAL rn day) AS new_Date
FROM CTE2
)
```

```
SELECT status as period_state, min(fail_date) as start_date, max(fail_date) as end_date FROM
CTE3
group by status, new_date
```