Best SQL Interview Resources

Data Analytics Free Resources

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 != 2 OR referee_id IS null
```

595 - Big Countries

```
SELECT name, population, area
FROM WORLD
WHERE area >= 3000000

OR population >= 25000000
```

1148 - Article Views I

```
SELECT DISTINCT author_id as id
FROM Views
WHERE viewer_id >= 1
AND author_id = viewer_id

ORDER BY author_id
```

1683 - Invalid Tweets

```
SELECT tweet_id
FROM Tweets

WHERE length(content) > 15
```

1378 - Replace Employee ID With The Unique Identifier

```
SELECT unique_id, name
FROM Employees e
LEFT JOIN EmployeeUNI eu

ON e.id = eu.id
```

1068 - Product Sales Analysis I

```
SELECT product_name, year, price
FROM Sales s
LEFT JOIN Product p

ON s.product_id = p.product_id
```

1581 - Customer Who Visited but Did Not Make Any Transactions

```
SELECT customer_id, COUNT(*) as count_no_trans
FROM Visits
WHERE visit_id NOT IN (SELECT DISTINCT visit_id FROM Transactions)

GROUP BY customer_id
```

197 - Rising Temperature

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

-- OR
SELECT w1.id
FROM Weather w1, Weather w2
WHERE w1.temperature > w2.temperature

AND SUBDATE(w1.recordDate, 1) = w2.recordDate
```

1661 - Average Time of Process per Machine

```
SELECT machine_id, ROUND(AVG(end - start), 3) AS processing_time
FROM
(SELECT machine_id, process_id,
    MAX(CASE WHEN activity_type = 'start' THEN timestamp END) AS start,
    MAX(CASE WHEN activity_type = 'end' THEN timestamp END) AS end
 FROM Activity
  GROUP BY machine_id, process_id) AS subq

GROUP BY machine_id
```

577 - Employee Bonus

```
SELECT name, 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 a.student_id, a.student_name, b.subject_name, COUNT(c.subject_name) AS attended_exams
FROM Students a
JOIN Subjects b
LEFT JOIN Examinations c
ON a.student_id = c.student_id
AND b.subject_name = c.subject_name
GROUP BY 1, 3

ORDER BY 1, 3
```

570. Managers with at Least 5 Direct Reports

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

-- OR
SELECT a.name
FROM Employee a
JOIN Employee b
WHERE a.id = b.managerId
GROUP BY b.managerId

HAVING COUNT(*) >= 5
```

1934. Confirmation Rate

```
SELECT
  s.user_id,
  ROUND(
```

```
    COALESCE(
      SUM(
        CASE WHEN ACTION = 'confirmed' THEN 1 END
      ) / COUNT(*), 0),2)
  AS confirmation_rate
FROM Signups s
LEFT JOIN Confirmations c
ON s.user_id = c.user_id

GROUP BY s.user_id;
```

620. Not Boring Movies

```
-- odd id, "boring", rating desc
SELECT *
FROM Cinema
WHERE id % 2 <> 0
AND description <> "boring"

ORDER BY rating DESC
```

1251. Average Selling Price

```
-- avg(selling), round 2
SELECT p.product_id,
  ROUND(SUM(price * units) / SUM(units), 2) AS average_price
FROM Prices p
LEFT JOIN UnitsSold s
ON p.product_id = s.product_id
AND purchase_date BETWEEN start_date AND end_date

GROUP BY p.product_id
```

1075. Project Employees I

```
-- avg(exp_yr), round 2, by project
SELECT project_id, ROUND(AVG(experience_years), 2) average_years
FROM Project p
LEFT JOIN Employee e
ON p.employee_id = e.employee_id

GROUP BY project_id
```

1633. Percentage of Users Attended a Contest

```
-- % desc, contest_id asc, round 2
```

```
SELECT r.contest_id,
      ROUND(COUNT(DISTINCT r.user_id) * 100 / (SELECT COUNT(DISTINCT user_id) FROM Users),
2) AS percentage
FROM Register r
GROUP BY r.contest_id

ORDER BY percentage DESC, r.contest_id ASC;
```

1211 Queries Quality and Percentage

```
--quality - avg(rating/position), poor query % - %(rating < 3), round 2
SELECT query_name,
   ROUND(AVG(rating/position), 2) AS quality,
   ROUND(SUM(IF(rating < 3, 1, 0)) * 100/ COUNT(rating), 2) AS poor_query_percentage
FROM Queries
GROUP BY query_name

-- OR
SELECT query_name,
   ROUND(AVG(rating/position), 2) AS quality,
   ROUND(SUM(
      CASE WHEN rating < 3 THEN 1 ELSE 0 END
   ) * 100/ COUNT(rating), 2) AS poor_query_percentage
FROM Queries

GROUP BY query_name
```

1193. Monthly Transactions I

```
-- month, country, count(trans), total(amt), count(approved_trans), total(amt)
SELECT DATE_FORMAT(trans_date, '%Y-%m') month, country,
       COUNT(state) trans_count,
       SUM(IF(state = 'approved', 1, 0)) approved_count,
       SUM(amount) trans_total_amount,
       SUM(IF(state = 'approved', amount, 0)) approved_total_amount
FROM Transactions
GROUP BY 1, 2

-- OR
SELECT DATE_FORMAT(trans_date, '%Y-%m') month, country,
       COUNT(state) trans_count,
       SUM(CASE WHEN state = 'approved' THEN 1 ELSE 0 END) approved_count,
       SUM(amount) trans_total_amount,
       SUM(CASE WHEN state = 'approved' THEN amount ELSE 0 END) approved_total_amount
FROM Transactions

GROUP BY 1, 2
```

1174. Immediate Food Delivery II

```
SELECT
    ROUND((COUNT(CASE WHEN d.order_date = d.customer_pref_delivery_date THEN 1 END) /
COUNT(*)) * 100, 2)  immediate_percentage
FROM Delivery d
WHERE d.order_date = (
    SELECT
    MIN(order_date)
    FROM Delivery
    WHERE customer_id = d.customer_id
    );

-- OR
SELECT ROUND(AVG(temp.order_date=temp.customer_pref_delivery_date) * 100, 2)
immediate_percentage
FROM (
    SELECT *, RANK() OVER(partition by customer_id ORDER BY order_date) od
    FROM Delivery) temp

WHERE temp.od = 1
```

550. Game Play Analysis IV

```
WITH login_date AS (SELECT player_id, MIN(event_date) AS first_login
FROM Activity
GROUP BY player_id),

recent_login AS (
SELECT *, DATE_ADD(first_login, INTERVAL 1 DAY) AS next_day
FROM login_date)

SELECT ROUND((SELECT COUNT(DISTINCT(player_id))
FROM Activity
WHERE (player_id, event_date) IN

(SELECT player_id, next_day FROM recent_login)) / (SELECT COUNT(DISTINCT player_id) FROM
Activity), 2) AS fraction
```

2356. Number of Unique Subjects Taught by Each Teacher

```
SELECT teacher_id, COUNT(DISTINCT subject_id) 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 BETWEEN DATE_SUB('2019-07-27', INTERVAL 29 DAY) AND '2019-07-27'

GROUP BY activity_date
```

1070. Product Sales Analysis III

```
SELECT s.product_id, s.year AS first_year, s.quantity, s.price
FROM Sales s
JOIN (
  SELECT product_id, MIN(year) AS year
  FROM sales
  GROUP BY product_id
  ) p
ON s.product_id = p.product_id
AND s.year = p.year

-- OR
WITH first_year_sales AS (
  SELECT s.product_id, MIN(s.year) as first_year
  FROM Sales s
  INNER JOIN Product p
  ON s.product_id = p.product_id
  GROUP BY s.product_id)
SELECT f.product_id, f.first_year, s.quantity, s.price
FROM first_year_sales f
JOIN Sales s
ON f.product_id = s.product_id

AND f.first_year = s.year
```

596. Classes More Than 5 Students

```
SELECT class
FROM Courses
GROUP BY class

HAVING COUNT(student) >= 5
```

1729. Find Followers Count

```
SELECT user_id, COUNT(DISTINCT follower_id) AS followers_count
FROM Followers
GROUP BY user_id

ORDER BY user_id ASC
```

619. Biggest Single Number

```
SELECT COALESCE(
  (SELECT num
  FROM MyNumbers
  GROUP BY num
  HAVING COUNT(num) = 1
  ORDER BY num DESC
  LIMIT 1), null)

  AS num
```

1045. Customers Who Bought All Products

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

)
```

1731. The Number of Employees Which Report to Each Employee

```
SELECT e1.employee_id, e1.name, COUNT(e2.employee_id) reports_count, ROUND(AVG(e2.age))
average_age
FROM Employees e1, Employees e2
WHERE e1.employee_id = e2.reports_to
GROUP BY e1.employee_id
HAVING reports_count > 0

ORDER BY e1.employee_id
```

1789. Primary Department for Each Employee

```
SELECT employee_id, department_id
FROM Employee
WHERE primary_flag = 'Y'
UNION
SELECT employee_id, department_id
FROM Employee
GROUP BY employee_id
HAVING COUNT(employee_id)=1

-- OR
SELECT employee_id,department_id
```

```
FROM Employee
WHERE primary_flag = 'Y' OR employee_id IN
    (SELECT employee_id
     FROM employee
     GROUP BY employee_id
     HAVING COUNT(department_id) = 1

    )
```

610. Triangle Judgement

```
SELECT x, y, z,
CASE WHEN x + y > z AND x + z > y AND y + z > x THEN 'Yes'
ELSE 'No' END AS triangle

FROM Triangle
```

180. Consecutive Numbers

```
WITH cte AS (
  SELECT id, num,
    LEAD(num) OVER (ORDER BY id) AS next,
    LAG(num) OVER (ORDER BY id) AS prev
  FROM Logs
)
SELECT DISTINCT(num) AS ConsecutiveNums
FROM cte

WHERE num = next AND num = prev
```

1164. Product Price at a Given Date

```
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
WHEN product_id NOT IN
(
```

```
  SELECT product_id
  FROM products
  WHERE change_date <= '2019-08-16'

)
```

1978. Employees Whose Manager Left the Company

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

ORDER BY employee_id
```

185. Department Top Three Salaries

```
WITH RankedSalaries AS
(SELECT
    e.Id AS employee_id,
    e.name AS employee,
    e.salary,
    e.departmentId,
    DENSE_RANK() OVER (PARTITION BY e.departmentId ORDER BY e.salary DESC) AS salary_rank
FROM Employee e)
SELECT d.name AS Department,
r.employee,
r.salary
FROM Department d
JOIN RankedSalaries r ON r.departmentId = d.id

WHERE r.salary_rank <=3;
```

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 patient_id, patient_name, conditions
FROM patients
WHERE conditions LIKE '% DIAB1%'

OR conditions LIKE 'DIAB1%'
```

196. Delete Duplicate Emails

```
DELETE p
FROM Person p, Person q
WHERE p.id > q.id

AND q.Email = p.Email
```

176. Second Highest Salary

```
SELECT
(SELECT DISTINCT Salary
FROM Employee
ORDER BY Salary DESC
LIMIT 1 OFFSET 1)
AS SecondHighestSalary


-- HINT: subquery is used to return null if there is no SecondHighestSalary
```

1517. Find Users With Valid E-Mails

```
SELECT *
FROM Users

WHERE mail REGEXP '^[A-Za-z][A-Za-z0-9_\.\-]*@leetcode\\.com$'
```

1204. Last Person to Fit in the Bus

```
-- 1000 kg limit
-- name of last person

WITH CTE AS (
   SELECT person_name, weight, turn, SUM(weight)
   OVER(ORDER BY turn) AS 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, SUM(IF(income<20000,1,0)) AS accounts_count
FROM Accounts
UNION
SELECT 'Average Salary' AS category, SUM(IF(income>=20000 AND income<=50000,1,0)) AS
accounts_count
FROM Accounts
UNION
SELECT 'High Salary' AS category, SUM(IF(income>50000,1,0)) AS accounts_count

FROM Accounts
```

626. Exchange Seats

```
-- id, student
-- swap every two consecutives
-- num(students): odd? no swap for last one

SELECT id,
CASE WHEN MOD(id,2)=0 THEN (LAG(student) OVER (ORDER BY id))
ELSE (LEAD(student, 1, student) OVER (ORDER BY id))
END AS 'Student'

FROM Seat
```

1327. List the Products Ordered in a Period

```
-- name, amt
-- >= 100 units, feb 2020

SELECT p.product_name, SUM(o.unit) AS unit
FROM Products p
LEFT JOIN Orders o
ON p.product_id = o.product_id
WHERE DATE_FORMAT(order_date, '%Y-%m') = '2020-02'
GROUP BY p.product_name

HAVING SUM(o.unit) >= 100
```

1484. Group Sold Products By The Date

```
SELECT sell_date,
COUNT(DISTINCT product) AS num_sold,
GROUP_CONCAT(DISTINCT product) AS 'products'
FROM Activities
GROUP BY sell_date

ORDER BY sell_date
```

1341. Movie Rating

```
(SELECT name AS results
FROM Users u
LEFT JOIN MovieRating mr
ON u.user_id = mr.user_id
GROUP BY name
ORDER BY COUNT(rating) DESC, name ASC
LIMIT 1)

UNION ALL

(SELECT title
FROM Movies m
LEFT JOIN MovieRating mr
ON m.movie_id = mr.movie_id
WHERE DATE_FORMAT(created_at, '%Y-%m') = '2020-02'
GROUP BY title
ORDER BY AVG(rating) DESC, title ASC
LIMIT 1

)
```

1321. Restaurant Growth

```
-- pay: last 7 days (today inclusive) - avg.amt (round, 2)

SELECT visited_on, amount, ROUND(amount/7, 2) AS average_amount
FROM (
    SELECT DISTINCT visited_on,
    SUM(amount) OVER(ORDER BY visited_on RANGE BETWEEN INTERVAL 6 DAY PRECEDING AND
CURRENT ROW) AS amount,
    MIN(visited_on) OVER() day_1
    FROM Customer
) t

WHERE visited_on >= day_1+6;
```

602. Friend Requests II: Who Has the Most Friends

```
-- `union` selects only unique vals, so we use `union all` here

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)
```

Coding Projects

Jobs & Internship Opportunities

Crack your coding Interviews

Udemy Free Courses with Certificate

Free access to all the Paid Channels: https://t.me/addlist/4q2PYC0pH_VjZDk5

ENJOY LEARNING 👍👍