**TO-DO**

1. 𝐃𝐚𝐭𝐚 𝐑𝐞𝐭𝐫𝐢𝐞𝐯𝐚𝐥 (𝐒𝐄𝐋𝐄𝐂𝐓 𝐐𝐮𝐞𝐫𝐢𝐞𝐬)  
    ↳ Second Highest Salary – [**https://lnkd.in/dX6nNqxg**](https://lnkd.in/dX6nNqxg)
2. SELECT
3. (
4. SELECT DISTINCT salary
5. FROM employee
6. ORDER BY salary DESC
7. OFFSET 1 LIMIT 1
8. ) AS SecondHighestSalary

↳ Customer Who Never Orders – [**https://lnkd.in/djZAM7Mq**](https://lnkd.in/djZAM7Mq)

1. SELECT name AS Customers
2. FROM Customers c
3. WHERE NOT EXISTS(
4. SELECT customerid
5. FROM orders O1
6. WHERE o1.customerid=c.id
7. )

↳ Employees Earning More than Their Managers – [**https://lnkd.in/dAT9PT4u**](https://lnkd.in/dAT9PT4u)

1. SELECT e.name AS employee
2. FROM Employee e
3. JOIN Employee e2
4. ON e.managerid=e2.id
5. WHERE e.salary>e2.salary

↳ Find Duplicate Emails – <https://leetcode.com/problems/duplicate-emails>

1. SELECT email
2. FROM person
3. GROUP BY email
4. HAVING COUNT(\*)>1

✅ 2. 𝐀𝐠𝐠𝐫𝐞𝐠𝐚𝐭𝐢𝐨𝐧𝐬 (𝐂𝐎𝐔𝐍𝐓, 𝐒𝐔𝐌, 𝐀𝐕𝐆, 𝐆𝐑𝐎𝐔𝐏 𝐁𝐘, 𝐇𝐀𝐕𝐈𝐍𝐆)  
 ↳ Total Sales by Product – [**https://lnkd.in/dfRjTf22**](https://lnkd.in/dfRjTf22)

1. SELECT product\_name, year,price
2. FROM sales s
3. JOIN product p
4. ON p.product\_id= s.product\_id

↳ Average Selling Price – [**https://lnkd.in/dKyJFj2y**](https://lnkd.in/dKyJFj2y)

1. SELECT p.product\_id, COALESCE(ROUND(SUM(units\*price)/SUM(units)::NUMERIC,2),0) AS average\_price
2. FROM prices p
3. LEFT JOIN unitssold u
4. ON u.product\_id=p.product\_id AND (u.purchase\_date BETWEEN start\_date AND end\_date)
5. GROUP BY p.product\_id

↳ Sales by Day – [**https://lnkd.in/dih9VMqw**](https://lnkd.in/dih9VMqw)

1. Question not available.

↳ Department Highest Salary – [**https://lnkd.in/dVvYUWhR**](https://lnkd.in/dVvYUWhR)

1. WITH cte AS(
2. SELECT d.name AS department, e.name AS employee, salary,
3. DENSE\_RANK() OVER(PARTITION BY d.name ORDER BY salary DESC) AS rnk
4. FROM employee e
5. JOIN department d
6. ON d.id=e.departmentid
7. )
8. SELECT department, employee, salary
9. FROM cte
10. WHERE rnk=1

✅ 3. 𝐉𝐨𝐢𝐧𝐬 (𝐈𝐍𝐍𝐄𝐑, 𝐋𝐄𝐅𝐓, 𝐑𝐈𝐆𝐇𝐓, 𝐅𝐔𝐋𝐋)  
 ↳ Combine Two Tables – [**https://lnkd.in/d4YRKYmt**](https://lnkd.in/d4YRKYmt)

1. SELECT p.firstName, p.lastName, a.city, a.state
2. FROM Person p
3. LEFT JOIN Address a
4. ON p.personId = a.personId

↳ Rank Scores – [**https://lnkd.in/dykb52wT**](https://lnkd.in/dykb52wT)

1. SELECT score,
2. DENSE\_RANK() OVER (ORDER BY score DESC) AS rank
3. FROM Scores

✅ 4. 𝐒𝐮𝐛𝐪𝐮𝐞𝐫𝐢𝐞𝐬 𝐚𝐧𝐝 𝐂𝐓𝐄𝐬  
 ↳ Nth Highest Salary – [**https://lnkd.in/dqHJcDF9**](https://lnkd.in/dqHJcDF9)

1. CREATE OR REPLACE FUNCTION NthHighestSalary(N INT) RETURNS TABLE (Salary INT) AS $$
2. BEGIN
3. RETURN QUERY (
4. -- Write your PostgreSQL query statement below.
5. with cte as(
6. select DISTINCT employee.salary,
7. DENSE\_RANK() OVER(ORDER BY employee.salary DESC) AS rank
8. from employee
9. )
10. SELECT cte.salary FROM cte
11. WHERE cte.rank=N
13. );
14. END;
15. $$ LANGUAGE plpgsql;

↳ Customer's Orders – [**https://lnkd.in/dfyhuRvA**](https://lnkd.in/dfyhuRvA)

1. SELECT customer\_number FROM Orders
2. GROUP BY customer\_number
3. ORDER BY count(order\_number) DESC
4. LIMIT 1

↳ Employees With Missing Salaries – <https://leetcode.com/problems/employees-with-missing-information>

1. SELECT COALESCE(e.employee\_id,s.employee\_id) AS employee\_id  FROM Employees e
2. FULL JOIN Salaries s USING (employee\_id)
3. WHERE e.name IS NULL OR s.salary IS NULL
4. ORDER BY 1 ASC

✅ 5. 𝐖𝐢𝐧𝐝𝐨𝐰 𝐅𝐮𝐧𝐜𝐭𝐢𝐨𝐧𝐬 (𝐑𝐀𝐍𝐊, 𝐋𝐀𝐆, 𝐋𝐄𝐀𝐃, 𝐑𝐎𝐖\_𝐍𝐔𝐌𝐁𝐄𝐑)  
 ↳ Rank Scores – [**https://lnkd.in/dykb52wT**](https://lnkd.in/dykb52wT)

-- Write your MySQL query statement below

1. SELECT score,
2. DENSE\_RANK() OVER (ORDER BY score DESC) AS rank
3. FROM Scores

↳ Activity Rank by User – [**https://lnkd.in/dJiDFcVR**](https://lnkd.in/dJiDFcVR)

1. WITH first AS(
2. SELECT player\_id, min(event\_date) AS first\_login
3. FROM activity
4. GROUP BY player\_id
5. ),
6. second AS(
7. SELECT a.player\_id,a.event\_date
8. FROM activity a
9. JOIN first f
10. ON f.first\_login=a.event\_date - INTERVAL '1 DAY' AND F.PLAYER\_ID=A.PLAYER\_ID
11. )
12. SELECT ROUND(COUNT(s.player\_id)/(SELECT COUNT(DISTINCT player\_id) FROM activity)::NUMERIC,2) AS fraction FROM second s;

↳ Find Lagging Sales Trends – [**https://lnkd.in/drEavFAg**](https://lnkd.in/drEavFAg)  
✅ 6. 𝐃𝐚𝐭𝐚 𝐅𝐢𝐥𝐭𝐞𝐫𝐢𝐧𝐠 𝐚𝐧𝐝 𝐂𝐥𝐞𝐚𝐧𝐢𝐧𝐠  
 ↳ Delete Duplicate Records – [**https://lnkd.in/d5uNRzcJ**](https://lnkd.in/d5uNRzcJ)

1. # Write your MySQL query statement below
2. DELETE p2 FROM person p1
3. join person p2
4. ON p1.email=p2.email AND p1.id<p2.id

↳ Normalize Sales Data – [**https://lnkd.in/dEftX9Pn**](https://lnkd.in/dEftX9Pn)

1. SELECT p.product\_id, p.product\_name FROM Product p
2. JOIN Sales s
3. ON p.product\_id = s.product\_id
4. GROUP BY p.product\_id
5. HAVING MIN(s.sale\_date) >= "2019-01-01" AND MAX(s.sale\_date) <= "2019-03-31";

↳ Fix Outlier Sales – [**https://lnkd.in/dYeKgFqP**](https://lnkd.in/dYeKgFqP)

↳ Correct Employee Data – [**https://lnkd.in/dU2-ZAEC**](https://lnkd.in/dU2-ZAEC)

1. SELECT name, bonus
2. FROM employee e
3. left join bonus b
4. ON b.empid = e.empid
5. where bonus<1000 or bonus is null