# Reference: <https://www.datageekinme.com/projects/leetcode-sql-solutions/>

# All the codes are run in Zepl notebook

# 579. Find Cumulative Salary of an Employee

The Employee table holds the salary information in a year.

Write a SQL to get the cumulative sum of an employee’s salary over a period of 3 months but exclude the most recent month.

The result should be displayed by ‘Id’ ascending, and then by ‘Month’ descending.

Example  
A close up of a white wall

Description automatically generated

Explanation  
Employee ‘1’ has 3 salary records for the following 3 months except the most recent month ‘4’: salary 40 for month ‘3’, 30 for month ‘2’ and 20 for month ‘1’  
So the cumulative sum of salary of this employee over 3 months is 90(40+30+20), 50(30+20) and 20 respectively.

A screenshot of a social media post

Description automatically generated

**Analysis: 2 steps,**

**to get the cumulative sum of the salary over a period of 3,**

**then we need to exclude the most recent month**

SELECT E1.Id, E1.Month,

(IFNULL(E1.Salary,0) + IFNULL(E2.Salary,0) + IFNULL(E3.Salary,0)) AS Salary

FROM

(SELECT Id, MAX(Month) AS max\_month FROM Employee Group BY Id ) AS max\_month

LEFT JOIN Employee E1 ON E1.Id=max\_month.Id AND E1.Month < max\_month.max\_month

LEFT JOIN Employee E2 ON E2.Id=E1.Id AND E1.month = E2.Month +1

LEFT JOIN Employee E3 ON E3.Id=E2.Id AND E2.Month = E3.month +1

ORDER BY E1.Id, E1.Month DESC;

**# this solution, left join the newly created table with max month and 3 E tables**

**# parenthesis for 3 IFNULL()**

SELECT E1.Id, E1.Month,

(SELECT SUM(Salary) FROM Employee E2 WHERE E1.Id = E2.Id AND E2.Month <= E1.Month) AS salary

FROM Employee E1

WHERE (E1.Id, E1.Month) NOT IN ( SELECT Id, MAX(Month) FROM Employee GROUP BY Id)

GROUP BY Id, Month

ORDER BY Id, Month DESC

**# a parenthesis is a must here (E1.Id, E1.Month)**

**# This solution does not consider 3 months as a period. It sums all months exclude the latest month**

# 601. Human Traffic of Stadium

X city built a new stadium, each day many people visit it and the stats are saved as these columns: id, date, people

Please write a query to display the records which have 3 or more consecutive rows and the amount of people more than 100(inclusive).

For example, the table stadium:

A screenshot of a cell phone

Description automatically generated

**analysis:**

**step 1: get the days with people greater than or equal to 100**

**select** **distinct** t1.**\***

**from** stadium t1, stadium t2, stadium t3

**where** t1.people **>=** 100 **and** t2.people **>=** 100 **and** t3.people **>=** 100

;

**step 2: Considering t1, t2 and t3 are identical. Taking t1 for example, it could exist in the beginning of the consecutive 3 days, or the middle, or the last.**

**(when t1 is the fixed value, the order of t2 and t3 do not matter. We still get the same result.)**

* t1 in the beginning: (t1.id - t2.id = 1 and t1.id - t3.id = 2 and t2.id - t3.id =1) -- t1, t2, t3
* t1 in the middle: (t2.id - t1.id = 1 and t2.id - t3.id = 2 and t1.id - t3.id =1) -- t2, t1, t3
* t1 in the end: (t3.id - t2.id = 1 and t2.id - t1.id =1 and t3.id - t1.id = 2) -- t3, t2, t1

**step 3: filter the duplicates with DISTINCT**

Solution:

SELECT DISTINCT day1.\*

FROM stadium day1, stadium day2, stadium day3

WHERE day1.people >= 100 AND day2.people >= 100 AND day3.people >= 100

AND (

(day1.date-day2.date=1 AND day1.date-day3.date=2 AND day2.date- day3.date=1)

or

(day2.date-day1.date=1 AND day2.date-day3.date=2 AND day1.date- day3.date=1)

or

(day3.date-day2.date=1 AND day3.date-day1.date=2 AND day2.date- day1.date=1)

)

ORDER BY day1.id;

# 618. Students Report by Geography

A U.S graduate school has students from Asia, Europe and America. The students’ location information are stored in table student as below.

| **name** | **continent** |
| --- | --- |
| Jack | America |
| Pascal | Europe |
| Xi | Asia |
| Jane | America |

Pivot the continent column in this table so that each name is sorted alphabetically and displayed underneath its corresponding continent. The output headers should be America, Asia and Europe respectively. It is guaranteed that the student number from America is no less than either Asia or Europe.

For the sample input, the output is:

A screenshot of a cell phone

Description automatically generated

Follow-up: If it is unknown which continent has the most students, can you write a query to generate the student report?

**Analysis: use session variable ( create auto increment row id for each continent ) and join**

**(1). Create 3 temp tables**

**(2). Join these 3 tables on row\_id**

One issue you may encounter is the student list for America is not complete if you use regular inner join since there are more records in this list comparing with the other two.

So you may have a solution to use the **outer join.** Correct!

**But how to arrange the 3 tables?**

The trick is to **put the America list in the middle** so that we can use right (outer) join and left (outer) join to connect with other two tables.

SELECT Asia, America, Europe

FROM

(SELECT @as:=0, @am:=0, @eu:=0) t,

**(SELECT @as:= @as +1 AS asid, name AS Asia**

**FROM student**

**WHERE continent = 'Asia'**

**ORDER BY Asia) t1**

Right JOIN

**(SELECT @am:= @am +1 AS amid, name AS America**

**FROM student**

**WHERE continent = 'America'**

**ORDER BY America) t2**

ON t1.asid = t2.amid

LEFT JOIN

**(SELECT @eu:= @eu +1 AS euid, name AS Europe**

**FROM student**

**WHERE continent = 'Europe'**

**ORDER BY Europe) t3**

ON t2.amid = t3.euid

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# 615. Average Salary: Departments vs. Company

Given two tables as below, write a query to display the comparison result (higher/lower/same) of the average salary of employees in a department to the company’s average salary.

A screenshot of a cell phone

Description automatically generated

Explanation

In March, the company’s average salary is (9000+6000+10000)/3 = 8333.33…

The average salary for department ‘1’ is 9000, which is the salary of employee\_id ‘1’ since there is only one employee in this department. So the comparison result is ‘higher’ since 9000 > 8333.33 obviously.

The average salary of department ‘2’ is (6000 + 10000)/2 = 8000, which is the average of employee\_id ‘2’ and ‘3’. So the comparison result is ‘lower’ since 8000 < 8333.33.

With the same formula for the average salary comparison in February, the result is ‘same’ since both the department ‘1’ and ‘2’ have the same average salary with the company, which is 7000.

**Analysis: use avg() and case when**

**3 steps:**

**1.Average salary of the company for each month (each month should include the year too)**

**2.Average salary of the department for each month**

**3.Join two tables and then Compare the two average values with case when**

**# pay attention to the alias, don’t mix up**

**# “end as” not just “ as” when using case when**

**# put 3 conditions at the beginning and use the alias from the following tables**

**# two ways to format pay\_date column with extrac() or date\_format()**

DATE\_FORMAT( s1.pay\_date, '%Y-%m') AS pay\_month

EXTRACT(YEAR\_MONTH FROM s1.pay\_date) AS pay\_month

SELECT company\_average.pay\_month, department\_average.department\_id,

**CASE WHEN d\_avg > c\_avg THEN 'higher'**

**WHEN d\_avg < c\_avg THEN 'lower'**

**ELSE 'same'**

**END AS comparison**

FROM

**(SELECT EXTRACT(YEAR\_MONTH FROM s1.pay\_date) AS pay\_month,**

**AVG(amount) AS c\_avg**

**FROM salary s1**

**GROUP BY pay\_month)** as company\_average

JOIN

**(SELECT EXTRACT(YEAR\_MONTH FROM s2.pay\_date) AS pay\_month,**

**e2.department\_id, AVG(s2.amount) AS d\_avg**

**FROM salary s2**

**JOIN employee e2**

**ON e2.employee\_id = s2.employee\_id**

**GROUP BY pay\_month, e2.department\_id )** as department\_average

ON company\_average.pay\_month = department\_average.pay\_month

# 569. Median Employee Salary

The Employee table holds all employees. The employee table has three columns: Employee Id, Company Name, and Salary.

A close up of a white wall

Description automatically generated

Write a SQL query to find the median salary of each company. Bonus points if you can solve it without using any built-in SQL functions.

A close up of a white background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**Intuition**

By the definition of *median*, the count of the bigger numbers than itself should be equal to the count of the smaller ones in an *odd* array.

**The Key:**

**the median's frequency should be equal or greater than the absolute difference of its bigger elements and small ones in an array no matter whether it has odd or even amount of numbers and whether they are distinct.**

# bug here

SELECT ANY\_VALUE(Employee.Id), e1.Company, e1.Salary

FROM Employee e1, Employee e2

WHERE e1.Company = e2.Company

GROUP BY e1.Company, e1.Salary， Employee.Id

HAVING **SUM(CASE WHEN e1.Salary = e2.Salary THEN 1 ELSE 0 END)**  >=

**ABS(SUM( SIGN(e1.Salary - e2.Salary)) )**

# case when and sign() are used to return the frequency

ORDER BY Employee.Id

# 571. Find Median given frequency of Numbers

A screenshot of a social media post

Description automatically generated

**The Key:**

**the median's frequency should be equal or greater than the absolute difference of its bigger elements and small ones in an array no matter whether it has odd or even amount of numbers and whether they are distinct.**

**Analysis: most of the process is about “frequency” , last step is “average number”**

**Step 1: split the numbers into bigger elements part and smaller elements part**

**Step 2: absolute difference between sum of bigger elements frequency and sum of smaller elements frequency**

**Step 3: apply the result from previous two steps in where clause to get average value**

SELECT AVG(n.Number) AS median

FROM Numbers n

WHERE n.Frequency >=

ABS(

(SELECT SUM(n2.Frequency) FROM Numbers n2 WHERE n2.Number <= n.Number) -

(SELECT SUM(n3.Frequency) FROM Numbers n3 WHERE n3.Number >= n.Number )

)

# pay attention to English keyboard and Chinese keyboard