Headcount planning: The juggling act of future vision and past data

While planning future staffing needs is a critical function for organizations, it often relies heavily on historical data. As an HR Manager, you might find yourself crunching numbers with analysts, but be prepared for curveballs from business leaders. They may ask for headcount figures on specific dates, and sometimes even expect them on the spot.

Here's where things get real: A fabricated scenario

Imagine a business leader urgently requests headcount information for a specific date. The catch? They're unsure of the exact date themselves, and expect you to generate the numbers quickly. You turn to your self-service HR system, and it displays data in a complex format...

Employee_Name	Event_Date	Status
S.Kumar	1/7/2019	Joined
A. Swati	1/14/2019	Joined
C. Poornima	1/14/2019	Attrition
D. Manish	1/21/2019	Joined
R. Manoj	1/21/2019	Joined
E. Srini	1/28/2019	Joined
W. Suvasini	1/28/2019	Joined
R. Atul	2/4/2019	Joined
K. Amit	2/4/2019	Joined
J. Rishabh	2/4/2019	Joined
S. Prasad	2/11/2019	Attrition
T. Oli	2/18/2019	Attrition
S. Sanjay	2/18/2019	Attrition
F. Maneesh	3/25/2019	Attrition
Y. Sunil	4/1/2019	Attrition
R. Aneeta	4/22/2019	Joined
R. Ashwin	4/29/2019	Joined
J. Basanti	5/6/2019	Joined
T. Soni	5/6/2019	Joined

About the dataset – **Digging into the Data:**

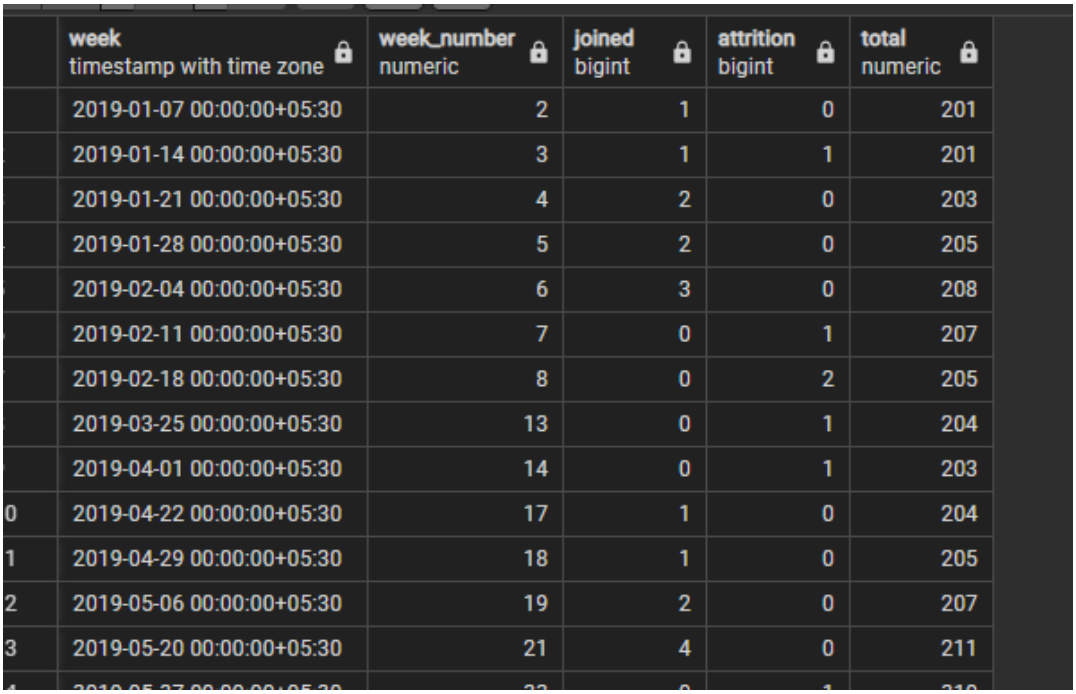
After examining the dataset, we discovered it includes dates ranging from January 1st, 2019, to July 25th, 2022. Since we need data from a prior date, December 31st, 2018, the first step is to inquire about the headcount for that specific date. Fortunately, the system returned a value of 200 employees. With this information in hand, let's explore the SQL code used to reach a solution...

SQL Code-

```sql
select * FROM headcount;



WITH weekly_status AS(

select

DATE_TRUNC('week', event_date) as week_start,
EXTRACT('week' FROM event_date) AS week_number,
sum(case when status = 'Joined' THEN 1 ELSE 0 END) AS Joined,
sum(case when status = 'Attrition' THEN 1 ELSE 0 END) AS Attrition
FROM headcount
GROUP BY week_start,week_number
ORDER BY week_start ASC
)

select
week_start as week,
week_number,
joined,
attrition,
SUM(joined - attrition) OVER (ORDER BY week_Start) + 200 AS total
FROM weekly_Status
GROUP BY week_start, joined, attrition, week_number
ORDER BY week_start;
```

Solution-

The below output shows us weekly attrition and addition information. If required, with some changes to the code, we can revise it to Year, Quarter, Month, Day.



	week timestamp with time zone	week_number numeric	joined bigint	attrition bigint	total numeric
	2019-01-07 00:00:00+05:30	2	1	0	201
	2019-01-14 00:00:00+05:30	3	1	1	201
	2019-01-21 00:00:00+05:30	4	2	0	203
	2019-01-28 00:00:00+05:30	5	2	0	205
	2019-02-04 00:00:00+05:30	6	3	0	208
	2019-02-11 00:00:00+05:30	7	0	1	207
	2019-02-18 00:00:00+05:30	8	0	2	205
	2019-03-25 00:00:00+05:30	13	0	1	204
	2019-04-01 00:00:00+05:30	14	0	1	203
0	2019-04-22 00:00:00+05:30	17	1	0	204
1	2019-04-29 00:00:00+05:30	18	1	0	205
2	2019-05-06 00:00:00+05:30	19	2	0	207
3	2019-05-20 00:00:00+05:30	21	4	0	211

Note – the entire code, data file is available on my GitHub page.