## SQL TIPS AND TRICKS

**PART 18** 

## **SQL** Aggregations ZERO TO ADVANCED

MAYURI DANDEKAR

### Example--

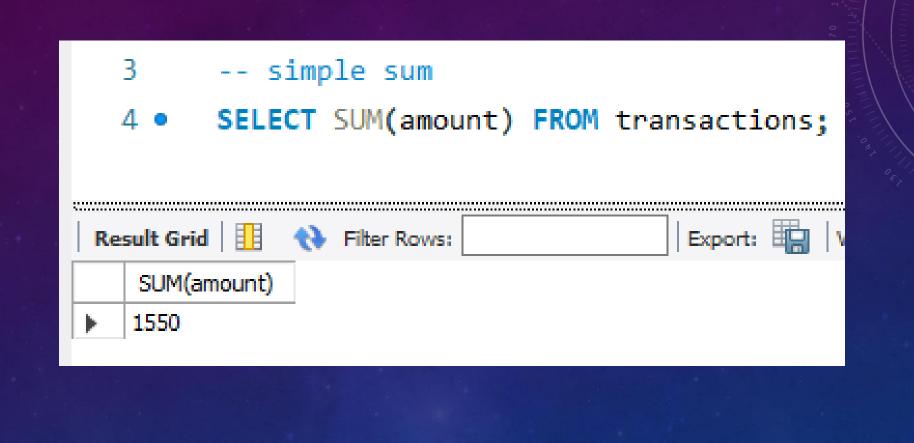




### 

Ехрх

	order_id	cust_id	order_date	amount
<b>-</b>	1	1	2020-01-15	150
	2	1	2020-02-10	150
	3	2	2020-01-16	200
	4	2	2020-02-25	150
	5	3	2020-01-10	300
	6	3	2020-02-20	300
	7	4	2020-01-20	150
	8	5	2020-02-20	150



- 7 SELECT cust\_id, SUM(amount)
- 8 FROM transactions
- 9 GROUP BY cust\_id;

Re	sult Grid	Filter I	ows:	Export
	cust_id	SUM(amount)		
•	1	300		
	2	350		
	3	600		
	4	150		
	5	150		

```
-- using window function

SELECT *,

SUM(amount) OVER(PARTITION BY cust_id) AS c_amount,

SUM(amount) OVER(ORDER BY order_date) AS running_amount,

SUM(amount) OVER(PARTITION BY cust_id ORDER BY order_date) AS amount

FROM transactions;
```

Result Grid							
	order_id	cust_id	order_date	amount	c_amount	running_amount	amount
•	1	1	2020-01-15	150	300	450	150
	2	1	2020-02-10	150	300	950	300
	3	2	2020-01-16	200	350	650	200
	4	2	2020-02-25	150	350	1550	350
	5	3	2020-01-10	300	600	300	300
	6	3	2020-02-20	300	600	1400	600
	7	4	2020-01-20	150	150	800	150
	8	5	2020-02-20	150	150	1400	150

```
18
       -- different variations to calculate rolling
19 •
      SELECT *,
      SUM(amount) OVER(ORDER BY order date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) AS rolling 3,
20
      SUM(amount) OVER(ORDER BY order date ROWS BETWEEN 2 PRECEDING AND 1 PRECEDING) AS rolling 2,
21
22
      SUM(amount) OVER(ORDER BY order date ROWS BETWEEN 2 PRECEDING AND 1 FOLLOWING) AS rolling 4,
      SUM(amount) OVER(ORDER BY order date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS rolling,
23
    SUM(amount) OVER(PARTITION BY cust id ORDER BY order date
24
                  ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS c rolling
25
26
      FROM transactions;
```

Result Grid   1									
	order_id	cust_id	order_date	amount	rolling_3	rolling_2	rolling_4	rolling	c_rolling
<b>&gt;</b>	1	1	2020-01-15	150	450	300	650	450	150
	2	1	2020-02-10	150	500	350	800	950	300
	3	2	2020-01-16	200	650	450	800	650	200
	4	2	2020-02-25	150	600	450	600	1550	350
	5	3	2020-01-10	300	300	NULL	450	300	300
	6	3	2020-02-20	300	600	300	750	1250	600
	7	4	2020-01-20	150	500	350	650	800	150
	8	5	2020-02-20	150	600	450	750	1400	150

- 28 -- analytics function without using them
- 29 **SELECT \***,
- 30 SUM(amount) OVER(ORDER BY order\_date ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING) AS lag\_amount,
- 31 SUM(amount) OVER(ORDER BY order\_date ROWS BETWEEN 1 FOLLOWING AND 1 FOLLOWING) AS lead\_amount
- 32 FROM transactions;

Result Grid   1							
	order_id	cust_id	order_date	amount	lag_amount	lead_amount	
<b>)</b>	5	3	2020-01-10	300	NULL	150	
	1	1	2020-01-15	150	300	200	
	3	2	2020-01-16	200	150	150	
	7	4	2020-01-20	150	200	150	
	2	1	2020-02-10	150	150	300	
	6	3	2020-02-20	300	150	150	
	8	5	2020-02-20	150	300	150	
	4	2	2020-02-25	150	150	NULL	

# THANK YOU

MAYURI DANDEKAR