Modern SQL Practice

CE384: Database Design
Maryam Ramezani
Sharif University of Technology
maryam.ramezani@sharif.edu

Question

Find the **ranking of each seller per month** based on their sales amount.

id	seller	month	amount
1	Ali	Jan	1000
2	Sara	Jan	1200
3	Ali	Feb	1100
4	Sara	Feb	1700
5	Reza	Feb	900
6	Ali	Jan	500

Output

seller	month	amount	monthly_rank
Sara	Jan	1200	1
Ali	Jan	1000	2
Ali	Jan	500	3
Sara	Feb	1700	1
Ali	Feb	1100	2
Reza	Feb	900	3

Version 1 - Query

```
SELECT
  s1.seller,
  s1.month,
  s1.amount,
  COUNT(*) AS monthly_rank
FROM sales s1
JOIN sales s2
  ON s1.month = s2.month AND s1.amount <= s2.amount
GROUP BY s1.seller, s1.month, s1.amount;
```

Version 2 - Query

```
SELECT
  seller,
  month,
  amount,
  RANK() OVER (PARTITION BY month ORDER BY amount DESC) AS monthly_rank
FROM sales;
```

Over: Rows Between vs Range Between

Feature	`ROWS BETWEEN`	`RANGE BETWEEN`
Based on	Physical row position	Values of the column in `ORDER BY`
Handles duplicate values	Ignores them — treats rows individually	Includes **all** rows with equal `ORDER BY` values
Precision	Exact row-level control	Value-based, broader inclusion
Use cases	Moving averages by fixed number of rows	Value-range-based calculations (e.g., date or numeric intervals)
Performance	Typically faster and simpler	Can be slower, more complex to compute

Rows Between vs Range Between

- Use ROWS BETWEEN when you want exact row counts (e.g., rolling 3-row average), and RANGE BETWEEN when you're aggregating over a range of values (e.g., all rows within same ORDER BY value or a date interval).
  - SUM(sales) OVER (ORDER BY date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
    - This sums the current row and the two immediately preceding rows, regardless of whether they have the same date or value.
  - SUM(sales) OVER (ORDER BY date RANGE BETWEEN INTERVAL 2 DAY PRECEDING AND CURRENT ROW)
    - This includes **all rows** with dates within 2 days of the current row's date — not just 2 physical rows — and includes **duplicates**.

Question

Write a SQL query to find the top 5 customers by total purchase amount.

The **orders** table has these columns:
- order_id
- customer_id
- purchase_amount
- order_date

Solution

```sql
SELECT
  customer_id,
  SUM(purchase_amount) as total_purchase
FROM orders
GROUP BY customer_id
ORDER BY total_purchase DESC
LIMIT 5
```

Question

Given the daily_sales table, how would you calculate 7-day moving average of daily sales?

The **daily_sales** table has these columns
- date
- daily_sales

Solution

```sql
SELECT
  date,
  AVG(daily_sales) OVER (
    ORDER BY date
    ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
  ) AS seven_day_moving_avg
FROM daily_sales_table
ORDER BY date
```

FETCH FIRST 3 ROWS WITH TIES

- Find top 3 salaries:

salary	name	id
5000	Ali	1
4800	Sara	2
4700	Reza	3
4700	Lila	4
4700	Amir	5
4600	Nima	6
4400	Elham	7