FRAME CLAUSE IN WINDOWS FUNCTIONS

The FRAME clause is an extension to window functions that allows you to define a subset, or "frame," of rows to be used for the calculation. It controls which rows are considered within the window for each calculation. This can be particularly useful when you need to compute running totals, moving averages, or any calculation that depends on a range of rows relative to the current one.

Components of the FRAME Clause

1. ROWS: Defines the frame by the physical number of rows.
2. RANGE: Defines the frame based on the value of the current row.
3. UNBOUNDED PRECEDING: Includes all rows from the beginning of the partition.
4. UNBOUNDED FOLLOWING: Includes all rows up to the end of the partition.
5. CURRENT ROW: Includes only the current row.
6. N PRECEDING or N FOLLOWING: Includes rows that are N rows before or after the current row.

Practical Uses of the FRAME Clause in Different Industries

1. Retail

- Use Case: Calculate a 7-day moving average of daily sales.

- Function: `AVG() OVER (ORDER BY sale_date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)`

- Explanation: Retailers track moving averages of sales to smooth out day-to-day fluctuations and spot trends. For example, a 7-day moving average can give insights into the sales pattern for the past week.

- Example:

```
SELECT store_id, sale_date, sale_amount,

AVG(sale_amount) OVER (ORDER BY sale_date ROWS BETWEEN 6 PRECEDING AND
CURRENT ROW) AS moving_avg_sales

FROM sales;
```

2. Finance

- Use Case: Calculate a rolling sum of deposits over the last 3 months.

- Function: `SUM() OVER (PARTITION BY account_id ORDER BY deposit_date RANGE BETWEEN INTERVAL 3 MONTH PRECEDING AND CURRENT ROW)`

- Explanation: Banks may use this to track the cumulative deposits made in the last three months for each account to understand saving trends and patterns.

- Example:

```
SELECT account_id, deposit_date, deposit_amount,

SUM(deposit_amount) OVER (PARTITION BY account_id ORDER BY deposit_date
RANGE BETWEEN INTERVAL 3 MONTH PRECEDING AND CURRENT ROW) AS rolling_sum

FROM deposits;
```

3. Healthcare

- Use Case: Monitor a patient's blood pressure over the last 5 visits.

- Function: `AVG() OVER (PARTITION BY patient_id ORDER BY visit_date ROWS BETWEEN 4 PRECEDING AND CURRENT ROW)`

- Explanation: Healthcare providers may calculate the moving average of a patient's blood pressure over their last 5 visits to monitor health trends and make data-driven clinical decisions.

- Example:

```
SELECT patient_id, visit_date, blood_pressure,

AVG(blood_pressure) OVER (PARTITION BY patient_id ORDER BY visit_date ROWS BETWEEN 4 PRECEDING AND CURRENT ROW) AS moving_avg_bp

FROM patient_visits;
```

4. Education

- Use Case: Calculate the cumulative grade average for a student as they complete each assignment in a course.

- Function: `AVG() OVER (PARTITION BY student_id, course_id ORDER BY assignment_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)`

- Explanation: Schools or universities may calculate a running cumulative GPA for each student in a course as new assignments are graded.

- Example:

```
SELECT student_id, course_id, assignment_date, grade,

AVG(grade) OVER (PARTITION BY student_id, course_id ORDER BY assignment_date ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_gpa

FROM student_grades;
```

5. Manufacturing

- Use Case: Track the moving average of production output over the last 7 days for each machine.

- Function: `AVG() OVER (PARTITION BY machine_id ORDER BY production_date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)`

- Explanation: Manufacturers use this to monitor machine performance and detect any decline or improvement in production rates over time.

- Example:

```
SELECT machine_id, production_date, output,

AVG(output) OVER (PARTITION BY machine_id ORDER BY production_date ROWS BETWEEN 6 PRECEDING AND CURRENT ROW) AS moving_avg_output

FROM production_logs;
```

6. Telecommunications

- Use Case: Calculate the total data usage over the last month for each customer.

- Function: `SUM() OVER (PARTITION BY customer_id ORDER BY usage_date RANGE BETWEEN INTERVAL 30 DAY PRECEDING AND CURRENT ROW)`

- Explanation: Telecom companies can track data usage trends for individual customers by calculating the total data usage over the last 30 days.

- Example:

```
SELECT customer_id, usage_date, data_usage,
SUM(data_usage) OVER (PARTITION BY customer_id ORDER BY usage_date RANGE BETWEEN INTERVAL 30 DAY PRECEDING AND CURRENT ROW) AS total_usage_last_month
FROM data_usage_logs;
```

7. Logistics

- Use Case: Calculate the average delivery time over the last 10 deliveries for each driver.

- Function: `AVG() OVER (PARTITION BY driver_id ORDER BY delivery_date ROWS BETWEEN 9 PRECEDING AND CURRENT ROW)`

- Explanation: Logistics companies track driver performance by calculating the average time taken for the last 10 deliveries to assess efficiency and performance.

- Example:

```
SELECT driver_id, delivery_date, delivery_time,
AVG(delivery_time) OVER (PARTITION BY driver_id ORDER BY delivery_date ROWS BETWEEN 9 PRECEDING AND CURRENT ROW) AS avg_delivery_time
FROM deliveries;
```

Summary:

The FRAME clause allows companies across various industries to fine-tune their analysis by specifying which rows should be included in calculations. This enables:

1. Moving averages for trends
2. Rolling sums for financial metrics
3. Running totals for cumulative insights

These tools are especially valuable in scenarios where recent data is more relevant, such as sales tracking in retail, patient health monitoring in healthcare, or performance tracking in manufacturing. The FRAME clause brings flexibility and precision to window functions, making them highly effective for dynamic data analysis.