# Advanced SQL Queries

Optimizing SQL for Analytics in the Cloud

# Objectives

1. Explain the purpose of window functions and how they differ from aggregate queries that use GROUP BY.
2. Apply basic window functions to analyze patient data without losing row-level detail.

# Window Functions

Key Idea: A way to do calculations across a set of rows related to the current row, without collapsing the rows like GROUP BY.

- Work over a “window” of rows defined by OVER()
- Don’t reduce the number of rows (unlike aggregations)
- Use cases: ranking, running totals, comparisons to group averages

# Window Function Syntax

```
<function name>() OVER (
        PARTITION BY <column(s)>
        ORDER BY <column(s)>
        [ROW/RANGE frame specification]
)
```

- PARTITION BY: defines which group of rows to use (similar to GROUP BY)
- ORDER BY: Defines the row sequence inside each partition
- Window frame (optional, advanced): limits which rows inside the partition are considered

# Example: Ranking

Question: Who are the top earners in each department?

```
SELECT employee_id, department, salary,
       RANK() OVER (
         PARTITION BY department
         ORDER BY salary DESC
       ) AS dept_salary_rank
FROM employees;
```

# Example: Running Totals

Question: How much has each customer spent over time?

```
SELECT customer_id, order_id, order_date, amount,
       SUM(amount) OVER (
         PARTITION BY customer_id
         ORDER BY order_date
       ) AS running_total
FROM orders;
```

# Use of Window Functions

- Replace awkward correlated subqueries
- Enable ranking, percentiles, and moving averages
- Keep row-level detail and aggregated context in one query.
- Widely used in analytics

Postgres Documentation on window functions:

- https://www.postgresql.org/docs/current/tutorial-window.html
- https://www.postgresql.org/docs/current/functions-window.html