Window Functions Mastery

Commondealth Bank Customer Analytics

A Practical SQL Project

FRIDAY MORNING

The Friday Morning Challenge

9:00 AM - Director's Urgent Request

> "I need a customer performance dashboard for Monday's board meeting!"

Requirements:

- Top 10 customers ranked by balance
- Customer percentiles - Who's in top 10%? Bottom 25%?
- Running total of deposits over time
- Year-over-year balance comparison
- State-wise customer rankings

The Problem

Traditional SQL can't easily compare rows to each other!

GROUP BY collapses rows

Subqueries become messy

Self-joins are complex

Solution: Window Functions 🪟

What Are Window Functions?

Definition

Perform calculations **ACROSS rows** while keeping individual row details

The Key Difference

Approach	Input	Output	Use Case
GROUP BY	20 rows	5 rows	Totals per state
Window Functions	20 rows	20 rows	Rank each customer

Syntax

```
SELECT
  customer_name,
  current_balance,
  ROW_NUMBER() OVER (ORDER BY current_balance DESC) AS rank
FROM customers;
```

Think of it as...

Looking through a "window" at nearby rows while keeping your current row

FUNCTION #1

Window Function #1 - ROW_NUMBER()

Purpose

Assign unique sequential numbers (1, 2, 3...)

Query Example

```
SELECT
  customer_name,
  current_balance,
  ROW_NUMBER() OVER (
    ORDER BY current_balance DESC
  ) AS rank
FROM customers;
```

Result

Rank	Customer	Balance
1	William Zhang	$890K
2	James Chen	$750K
3	Mason Taylor	$670K
4	Amelia Lee	$625K
5	Harper Singh	$540K

Business Use Case

- ✅ Top 10 customer lists
- ✅ Leaderboards
- ✅ Pagination (rows 1-10, 11-20...)

Key: Always unique, even with ties!

FUNCTION #2

Window Function #2 - RANK() vs DENSE_RANK()

The Difference

```
SELECT
  customer_name,
  current_balance,
  RANK() OVER (ORDER BY current_balance DESC) AS rank_gaps,
  DENSE_RANK() OVER (ORDER BY current_balance DESC) AS rank_no_gaps
FROM customers;
```

Visual Example (with ties at $500K)

Customer	Balance	RANK()	DENSE_RANK()
William	$890K	1	1
James	$750K	2	2
Mason	$670K	3	3
Amelia	$625K	4	4
Harper	$540K	5	5
Isabella	$510K	6	6
Ava	$500K	7	7
Sarah	$500K	7	7
Next person	$480K	9 ← Gap!	8 ← No gap

When to Use

RANK()

Olympic medals (Gold, Silver, Silver, 4th place)

DENSE_RANK()

Categories (Tier 1, Tier 2, Tier 3)

FUNCTION #3

Window Function #3 - NTILE() for Percentiles

Purpose

Divide customers into equal groups

Query Example

```
SELECT
  customer_name,
  current_balance,
  NTILE(4) OVER (ORDER BY current_balance DESC) AS quartile,
  CASE
    WHEN NTILE(4) OVER (...) = 1 THEN 'Top 25% (VIP)'
    WHEN NTILE(4) OVER (...) = 2 THEN 'Premium'
    WHEN NTILE(4) OVER (...) = 3 THEN 'Standard'
    ELSE 'Basic'
  END AS segment
FROM customers;
```

Customer Segmentation

Quartile	Segment	# Customers	Strategy
1	Top 25% (VIP)	5	Personal banker
2	Premium	5	Special offers
3	Standard	5	Regular service
4	Basic	5	Digital-only

Common Uses

- **NTILE(4)** = Quartiles
- **NTILE(10)** = Deciles (top 10%, 20%...)
- **NTILE(100)** = Percentiles

FUNCTION #4

Window Function #4 - LAG() & LEAD()

Purpose

Access previous or next row values

Syntax

```
LAG(column, offset) OVER (ORDER BY date)
-- Previous row

LEAD(column, offset) OVER (ORDER BY date)
-- Next row
```

Year-over-Year Growth Example

```
SELECT
  customer_name,
  balance_2023,
  balance_2024,
  balance_2024 - balance_2023 AS growth_amount,
  ROUND(((balance_2024 - balance_2023) / balance_2023) * 100, 1)
AS growth_%
FROM customers
WHERE balance_2023 > 0;
```

Results

Customer	2023	2024	Growth $	Growth %
Liam	$55K	$68K	$13K	23.6%
Ava	$360K	$425K	$65K	18.1%
Benjamin	$310K	$355K	$45K	14.5%

Use Cases

- ✅ Period-over-period (MoM, YoY)
- ✅ Sequential analysis
- ✅ Trend detection

FUNCTION #5

Window Function #5 - Running Totals

Purpose

Cumulative sum as you progress through rows

Query Example

```
SELECT
  customer_name,
  signup_date,
  current_balance,
  SUM(current_balance) OVER (
    ORDER BY signup_date
  ) AS running_total
FROM customers
ORDER BY signup_date;
```

Cumulative Deposit Growth

Date	Customer	Balance	Running Total
2020-05-10	William	$890K	$890K
2020-08-15	Amelia	$625K	$1,515K
2020-12-12	Mason	$670K	$2,185K
2021-03-12	Benjamin	$355K	$2,540K
...	...	...	...
2023-07-15	Noah	$95K	$7,200K

Business Insights

- When did we hit $5M in deposits?
- Year-to-date (YTD) revenue
- Cumulative customer acquisition

PARTITION BY - The Game Changer

Purpose

Perform calculations separately within each group

Query Example

```
SELECT
  state,
  customer_name,
  current_balance,
  -- Rank across ALL customers
  ROW_NUMBER() OVER (ORDER BY current_balance DESC) AS overall_rank,
  -- Rank WITHIN each state
  ROW_NUMBER() OVER (PARTITION BY state ORDER BY current_balance DESC) AS state_rank
FROM customers;
```

Result

State	Customer	Balance	Overall Rank	State Rank
NSW	James Chen	$750K	2	1
NSW	Sarah Mitchell	$485K	7	2
VIC	William Zhang	$890K	1	1
VIC	Mason Taylor	$670K	3	2
QLD	Amelia Lee	$625K	4	1

Business Use Cases

- ✅ Top 3 customers per state
- ✅ Sales rep vs regional average
- ✅ Product performance per category

The Complete Executive Dashboard

Production Query (Combines Everything!)

```
WITH customer_analytics AS (
  SELECT
    customer_name,
    state,
    current_balance,
    -- Rankings
    ROW_NUMBER() OVER (ORDER BY current_balance DESC) AS overall_rank,
    ROW_NUMBER() OVER (PARTITION BY state ORDER BY current_balance DESC) AS state_rank,
    -- Percentiles
    NTILE(4) OVER (ORDER BY current_balance DESC) AS quartile,
    -- Growth
    (balance_2024 - balance_2023) / balance_2023 * 100 AS yoy_growth_%,
    -- Running total
    SUM(current_balance) OVER (ORDER BY signup_date) AS cumulative_deposits
  FROM customers
)
SELECT *
FROM customer_analytics
WHERE overall_rank <= 10;
```

What Director Gets

Rank	Name	State	Balance	Quartile	YoY%	Cumulative
1	William	VIC	$890K	1 (VIP)	8.5%	$890K
2	James	NSW	$750K	1 (VIP)	10.3%	$1,640K
3	Mason	VIC	$670K	1 (VIP)	9.8%	$2,310K

✅ Rankings

✅ Segments

✅ Growth

✅ Trends

Practice Exercises

1

Top 5 Customers Per State (Intermediate)

```
WITH ranked AS (
  SELECT *,
    ROW_NUMBER() OVER (
      PARTITION BY state
      ORDER BY balance DESC
    ) AS rank
  FROM customers
)
SELECT *
FROM ranked
WHERE rank <= 5;
```

2

Moving Average (Advanced)

```
SELECT
  customer_name,
  balance,
  AVG(balance) OVER (
    ORDER BY signup_date
    ROWS BETWEEN 2 PRECEDING AND
CURRENT ROW
  ) AS moving_avg_3
FROM customers;
```

3

Percentile Rank (Intermediate)

```
SELECT
 customer_name,
 PERCENT_RANK() OVER (ORDER BY
balance) * 100 AS percentile
FROM customers;
```

Business Impact

The Thursday Afternoon Victory

Timeline:

9:00 AM: Director requests dashboard

10:00 AM: Window functions queries written

2:00 PM: Complete dashboard delivered

Friday: Board presentation ready ✅

Dashboard Delivers:

Metric	Result	Business Decision
Top 10 ranked	$890K to $425K	VIP program targeting
Percentile analysis	Top 25% = 5 customers	Tiered service levels
Running totals	$7.2M cumulative	Growth trajectory clear
YoY top performer	+23.6% growth	Upsell opportunities
State rankings	#1 per state identified	Regional benchmarks

Impact:

- ✅ Executive decision-making enabled
- ✅ Customer segmentation strategy defined
- ✅ Growth opportunities identified
- ✅ Regional performance tracked

What You Mastered

Technical Skills ✓

- **ROW_NUMBER()** - Sequential numbering
- **RANK() / DENSE_RANK()** - Handling ties
- **NTILE()** - Percentile buckets
- **LAG() / LEAD()** - Period comparisons
- **SUM() OVER()** - Running totals
- **AVG() OVER()** - Moving averages
- **PARTITION BY** - Group-wise calculations
- **PERCENT_RANK()** - Exact percentiles

Business Skills ✓

- Customer segmentation (VIP, Premium, Standard)
- Performance benchmarking (quartiles, percentiles)
- Trend analysis (YoY, MoM, running totals)
- Regional comparisons (rank within groups)

Real-World Applications

- 📊 Sales leaderboards
- 💰 Financial reporting (YTD, running totals)
- 👥 Customer analytics
- 📈 HR analytics (salary bands, rankings)
- 🛍️ Product analytics (top N per category)

Career Value

Window Functions = Advanced Analyst Skill

- Essential for data science roles
- Interview favorite: "Top 3 per group"
- Production dashboards rely on these

You now have SQL superpowers! 🚀

Quick Reference Card

Window Function Syntax Patterns

```
-- Basic ranking
ROW_NUMBER() OVER (ORDER BY column DESC)

-- Ranking with groups
ROW_NUMBER() OVER (PARTITION BY group ORDER BY column DESC)

-- Percentiles
NTILE(4) OVER (ORDER BY column DESC)

-- Previous/Next row
LAG(column) OVER (ORDER BY date)
LEAD(column) OVER (ORDER BY date)

-- Running total
SUM(column) OVER (ORDER BY date)

-- Moving average (3-row window)
AVG(column) OVER (ORDER BY date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
```

Remember:

- **OVER()** = What makes it a window function
- **PARTITION BY** = Like GROUP BY but doesn't collapse rows
- **ORDER BY** = Defines row order for calculation
- **ROWS BETWEEN** = Defines window frame size