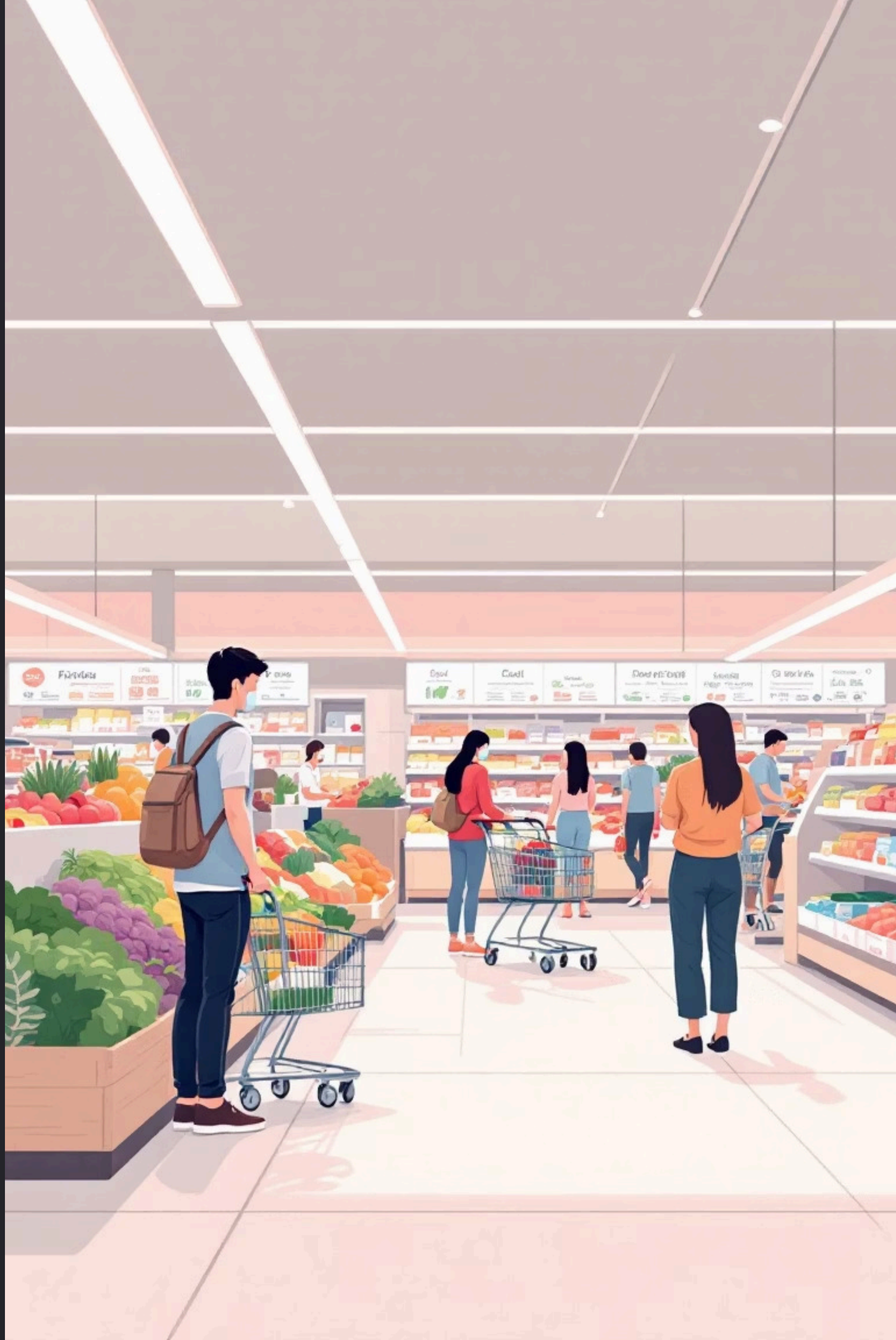
Wooltorths Time Intelligence Analysis



THE THURSDAY MORNING REQUEST

Time Intelligence for Business Reporting

Woolworths Sales Analysis | Video 4 of 15 | 17 minutes

> ### 10:00 AM Thursday - Your Email Pings
>
> **Commercial Manager:** "Hi - I need the weekly sales report for tomorrow's executive meeting. But this time, add something new: Year-over-Year comparisons and Month-over-Month trends. CEO wants to see if we're actually growing."

The Data You Have:

2 full years
of daily sales (2023-2024)

731 days
of transactions

$16+ billion
in total revenue

1,050 stores
across Australia

The Questions Executives Actually Ask:

- "How does 2024 compare to 2023?" (Year-over-Year)
- "Did December beat November?" (Month-over-Month)
- "What's the trend when you smooth out daily noise?" (Rolling average)
- "How does this Monday compare to the same Monday last year?" (Apples-to-apples)

The Excel Nightmare:

- Manual VLOOKUP for same month last year
- Copy/paste formulas for 24 months
- Calculate percentage changes manually
- One mistake = entire report is wrong
- **Time required: 4+ hours** (you'd work past midnight)

The SQL Solution:

- Date functions extract year/month automatically
- LAG() window function shows "previous period" in one line
- Rolling averages calculated with window functions
- **Time required: 30 minutes** (done by lunch!)

What You'll Master:

01

YEAR(), MONTH(), DATEPART() - Extract time periods

02

LAG() - Compare to previous period automatically

03

YoY and MoM percentage calculations

04

Rolling/moving averages

05

Time-based insights executives demand

UNDERSTANDING DATE FUNCTIONS

Breaking Dates Into Useful Pieces

> **The Problem:**
>
> Your database has: `2024-12-15`
>
> Executives want to know: "What YEAR? What MONTH? What QUARTER? What DAY OF WEEK?"

SQL Date Functions Are Your Tools:

Function	Example Input	Output
`YEAR(date)`	2024-12-15	2024
`YEAR(date)`	2023-06-20	2023
`MONTH(date)`	2024-12-15	12 (December)
`MONTH(date)`	2024-06-20	6 (June)
`DATEPART(QUARTER, date)`	2024-12-15	4 (Q4)
`DATEPART(QUARTER, date)`	2024-06-20	2 (Q2)
`DATEPART(WEEKDAY, date)`	2024-12-15 (Sunday)	1
`DATEPART(WEEKDAY, date)`	2024-12-16 (Monday)	2
`DATENAME(WEEKDAY, date)`	2024-12-15	"Sunday"
`DATENAME(MONTH, date)`	2024-12-15	"December"

Why This Matters:

Once you extract year/month/quarter, you can **GROUP BY** them:

- `GROUP BY YEAR(sale_date)` → Revenue per year
- `GROUP BY MONTH(sale_date)` → Revenue per month
- `GROUP BY DATEPART(WEEKDAY, sale_date)` → Revenue by day of week

The Business Application:

- "Show me total revenue by year" → `GROUP BY YEAR()`
- "Compare weekends vs weekdays" → `GROUP BY DATEPART(WEEKDAY)`
- "Which quarter performed best?" → `GROUP BY DATEPART(QUARTER)`

Real Query Example:

```
SELECT
    YEAR(sale_date) AS year,
    DATENAME(MONTH, sale_date) AS month,
    SUM(revenue) AS total_revenue
FROM daily_sales
GROUP BY YEAR(sale_date), MONTH(sale_date), DATENAME(MONTH, sale_date)
ORDER BY year, MONTH(sale_date);
```

Result:

- 2023, January, $680M
- 2023, February, $625M
- ...
- 2024, November, $710M
- 2024, December, $985M ← **Christmas spike!**

Key Insight: Date functions turn one date column into multiple time dimensions. Now you can analyse by year, month, quarter, day - whatever executives need.

YEAR-OVER-YEAR: THE #1 EXECUTIVE QUESTION

"Are We Growing?" - YoY Analysis with LAG()

> **The CEO's Question:** "I don't care about daily or monthly numbers. Tell me: Did we grow compared to LAST YEAR?"

Naive Approach (What Beginners Try):

1. Run two separate queries:
   `SELECT SUM(revenue) FROM sales WHERE YEAR = 2023;`
   `SELECT SUM(revenue) FROM sales WHERE YEAR = 2024;`
2. Manually calculate: (2024 - 2023) / 2023 * 100
3. Copy into Excel, create formula
4. Error-prone and slow

Professional Approach (LAG Window Function):

```
WITH yearly_summary AS (
  SELECT
    YEAR(sale_date) AS year,
    SUM(revenue) / 1000000 AS revenue_millions
  FROM daily_sales
  GROUP BY YEAR(sale_date)
)
SELECT
  year,
  revenue_millions,
  LAG(revenue_millions) OVER (ORDER BY year) AS
previous_year,
  revenue_millions - LAG(revenue_millions) OVER (ORDER
BY year) AS growth_amount,
  ROUND(((revenue_millions - LAG(revenue_millions)
OVER (ORDER BY year)) / LAG(revenue_millions) OVER
(ORDER BY year)) * 100, 2) AS yoy_percent
FROM yearly_summary;
```

What LAG() Does:

`LAG(revenue_millions) OVER (ORDER BY year)`

- Looks at the **PREVIOUS ROW** when ordered by year
- For 2024 row, it shows 2023 revenue
- For 2023 row, it shows NULL (no previous year in data)

The Result:

Year	Revenue	Previous Year	Growth $	YoY %
2023	$8,200M	NULL	NULL	NULL
2024	$8,856M	$8,200M	$656M	**8.0%**

CEO Reaction: "Perfect! 8% year-over-year growth. We're outpacing the industry (5-6% typical). Strategy is working!"

Why This Matters:

- One query, automatic calculation
- No manual Excel formulas
- Previous year shows automatically via LAG()
- Growth % calculated in SQL, always accurate

The Formula Explained:

(Current Year - Previous Year) / Previous Year × 100 = Growth %

($8,856M - $8,200M) / $8,200M × 100 = **8.0%**

Business Impact:

- **Board presentation:** "8% YoY growth"
- **Strategy validation:** Premium product mix working
- **Investor call:** "Outperforming retail sector"
- **Marketing budget:** Justified for next year

MONTH-OVER-MONTH: SPOTTING TRENDS

"Did We Grow This Month?" - MoM with LAG()

> **The CFO's Question:** "Year-over-year is great, but what happened THIS month compared to LAST month?"

Why MoM Matters:

- YoY is annual strategy validation
- MoM is operational performance tracking
- Seasonal businesses (retail!) live or die by monthly trends

The Query:

```
WITH monthly_summary AS (
  SELECT
    YEAR(sale_date) AS year,
    MONTH(sale_date) AS month,
    DATENAME(MONTH, sale_date) AS month_name,
    SUM(revenue) / 1000000 AS revenue_millions
  FROM daily_sales
  GROUP BY YEAR(sale_date), MONTH(sale_date), DATENAME(MONTH, sale_date)
)
SELECT
  year,
  month,
  month_name,
  revenue_millions,
  LAG(revenue_millions) OVER (ORDER BY year, month) AS prev_month,
  ROUND(((revenue_millions - LAG(revenue_millions) OVER (ORDER BY year, month)) / LAG(revenue_millions) OVER (ORDER BY year, month)) * 100, 2) AS mom_percent
FROM monthly_summary
WHERE year = 2024
ORDER BY year, month;
```

The Pattern You'll See:

1

November 2024

$710M

2

December 2024

$985M (+38.7% MoM) ← Christmas!

3

January 2025

$635M (-35.5% MoM) ← Post-holiday crash

This is NORMAL for retail!

- December spike every year (Christmas shopping)
- January drop every year (people spent out)
- Easter months (March/April) mini-spike
- July (EOFY sales) another spike

CFO Insight:

- "December up 39% MoM = Marketing campaign worked"
- "January down 36% = Expected, plan for lower staff"
- "Q4 strong = Hit annual targets"

LAG() Magic Again:

`LAG(revenue) OVER (ORDER BY year, month)`

- For December, shows November automatically
- For January, shows December automatically
- Calculates "previous month" without manual effort

Why Excel Fails Here:

- 24 months = 24 manual formulas
- One VLOOKUP mistake = entire trend wrong
- Change a month? Recalculate everything

SQL?

Change one WHERE clause, re-run, done.

Business Application:

- **Monthly exec meetings:** "Here's MoM performance"
- **Budget vs actual:** "November beat forecast by 8% MoM"
- **Staffing decisions:** "January down 35%, reduce shifts"

ROLLING AVERAGES: SMOOTHING THE NOISE

"Show Me the Real Trend" - 7-Day Moving Average

> **The CMO's Problem:** "Daily sales jump all over the place. Monday $18M, Tuesday $22M, Wednesday $20M, Saturday $32M. I can't see the actual TREND in all this noise!"

Why Daily Data Is "Noisy":

- Weekends spike (families shop Sat/Sun)
- Weekdays drop (people at work)
- Public holidays nosedive (stores closed)
- Random events (weather, sports events)

The Solution: Rolling Average

- Average the last 7 days (including today)
- Smooth out weekday/weekend fluctuations
- Reveal the underlying trend

The Window Function:

```
SELECT
  sale_date,
  total_revenue / 1000000 AS daily_revenue_millions,
  AVG(total_revenue / 1000000) OVER (
    ORDER BY sale_date
    ROWS BETWEEN 6 PRECEDING AND CURRENT ROW
  ) AS rolling_7day_avg
FROM daily_sales
WHERE sale_date >= '2024-11-01'
ORDER BY sale_date;
```

What "ROWS BETWEEN 6 PRECEDING AND CURRENT ROW" Means:

- **6 PRECEDING** = Previous 6 days
- **CURRENT ROW** = Today
- **Total** = 7 days (including today)

Example:

- Dec 15 rolling average = AVG(Dec 9, 10, 11, 12, 13, 14, 15)
- Dec 16 rolling average = AVG(Dec 10, 11, 12, 13, 14, 15, 16)
- Window "slides" forward one day at a time

The Result:

Date	Daily Revenue	7-Day Average
Dec 9 Mon	$22M	$24M
Dec 10 Tue	$21M	$24M
Dec 14 Sat	$34M	$25M ← Spike smoothed!
Dec 15 Sun	$35M	$26M ← Trend is UP
Dec 16 Mon	$23M	$27M

CMO Insight:

- Daily: Spikes up/down (confusing)
- 7-day average: Smooth upward trend (clear!)
- "December trend is UP. Christmas campaign working!"

Other Rolling Windows You Can Use:

- **30-day average:** `ROWS BETWEEN 29 PRECEDING AND CURRENT ROW`
- **90-day average:** `ROWS BETWEEN 89 PRECEDING AND CURRENT ROW`
- **Quarterly average:** `ROWS BETWEEN ...` (91 days for Q)

Why Executives Love This:

- Removes daily noise
- Shows TRUE directional trend
- Makes strategic decisions clearer
- Forecast future based on smooth trend

THE COMPLETE EXECUTIVE DASHBOARD

What You Delivered & Career Impact

The Final Query (Production-Ready):

```
WITH monthly_metrics AS (
 SELECT
 YEAR(sale_date) AS year,
 MONTH(sale_date) AS month,
 SUM(revenue) / 1000000 AS revenue_millions,
 SUM(online_revenue) / 1000000 AS online_millions
 FROM daily_sales
 GROUP BY YEAR(sale_date), MONTH(sale_date)
)
SELECT
 year,
 month,
 revenue_millions,

 -- Previous month (MoM)
 LAG(revenue_millions) OVER (ORDER BY year, month) AS prev_month,
 ROUND(((revenue_millions - LAG(revenue_millions) OVER (ORDER BY year, month))
 / LAG(revenue_millions) OVER (ORDER BY year, month)) * 100, 2) AS mom_percent,

 -- Same month last year (YoY)
 LAG(revenue_millions, 12) OVER (ORDER BY year, month) AS same_month_last_year,
 ROUND(((revenue_millions - LAG(revenue_millions, 12) OVER (ORDER BY year, month))
 / LAG(revenue_millions, 12) OVER (ORDER BY year, month)) * 100, 2) AS yoy_percent,

 -- Online penetration
 ROUND((online_millions / revenue_millions) * 100, 1) AS online_percent
```

```
FROM monthly_metrics
WHERE year = 2024;
```

What Executives See in ONE Table:

Month	Revenue	MoM%	YoY%	Online%
Nov	$710M	+5%	+8%	16.2%
Dec	$985M	+39%	+7%	18.5%

The Story This Tells:

- **MoM +39%** = Christmas spike (expected, healthy)
- **YoY +7-8%** = Consistent annual growth
- **Online% rising** = Digital transformation working

Business Decisions Made:

- ✅ 8% YoY growth → Strategy validated, continue course
- ✅ December spike +39% → Marketing campaign success
- ✅ Online% 16→18% → Invest more in e-commerce
- ✅ 7-day averages → True growth trend confirmed

The Friday Morning Meeting:

CEO: "Great report. I can see everything I need."

CFO: "MoM and YoY trends are perfect."

CMO: "Love the rolling averages - shows our campaign impact."

You: "Need anything else?"

Executives: "No, this is exactly what we wanted. Thanks!"

Career Impact:

- You delivered in **30 minutes** what takes others 4+ hours
- You understand time intelligence (rare skill!)
- You're now the "go-to" for trend analysis
- Next promotion: "Can do executive reporting independently"

What You Mastered in 17 Minutes:

✓ Date functions (YEAR, MONTH, DATEPART, DATENAME)

✓ LAG() window function for previous periods

✓ YoY calculations (same period last year)

✓ MoM calculations (previous month)

✓ Rolling averages (smoothing noise)

✓ Executive dashboard design (one query, all metrics)

✓ Business storytelling with time data

Real-World Applications:

- Every retail company needs this
- Every finance department needs this
- Every sales organisation needs this
- Every executive dashboard has time trends
- **You just became valuable to 10,000+ companies**

Practice Exercises:

Practice Exercises

1. Calculate quarter-over-quarter (QoQ) growth for 2024
2. Find best-performing day of week by revenue
3. Calculate 30-day rolling average for online sales
4. Compare weekend vs weekday average revenue
5. Identify month with highest YoY growth %

Next Video Preview

Video 5: Subqueries - Finding Outliers

AGL Energy - High Usage Customer Analysis

Learn:

- Subqueries
- Correlated subqueries
- EXISTS

Duration: 16 minutes