MAVEN DATA DRILL

MONTH-OVER-MONTH CALCULATIONS

Excel
SQL
Power BI
Python

Notes by
Amirhossein Tonekaboni

SAP Business One ERP Consultant
Inspired by Maven Analytics' YouTube tutorial

Linkedin.com/in/Tonekaboni
Allmylinks.com/Tonekaboni

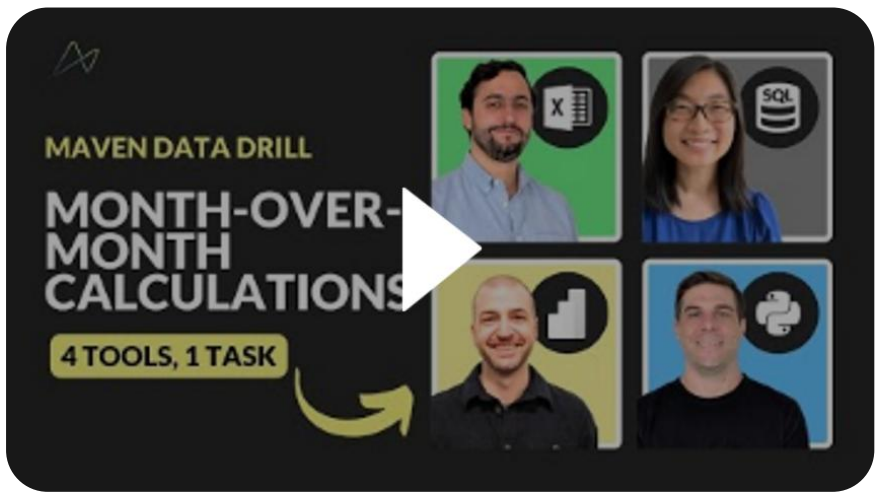
MAVEN ANALYTICS

What I Did

I turned raw coffee-shop transactions into month-over-month sales insights by aggregating by store/month and comparing each month's total to the prior one. Below are my notes on how I tackled this challenge using Excel, MySQL, Power BI, and Python.

Instructor Credits

Maven instructors **Enrique Ruiz, Chris Bruehl, Alice Zhao, Aaron Parry,** explain how to aggregate data and perform period-over-period calculations in Excel, Python, SQL, and Power BI respectively.



💡 Full walkthrough and full code are in Maven's video — I just flagged key takeaways here.

There is a raw transaction table and we need to aggregate sales by store/month, and calculate the dollar change versus the prior month.

- Dataset
- Data Drill (3) Rolling Up, Looking Back

Tools

Match your tool to the task—Excel for speed, pandas for scale, SQL for robustness, Power BI for visual impact.

1. **Microsoft Excel**
2. **Python**
3. **MySQL**
4. **Power BI**

1. Microsoft Excel 🥇

This was the most straightforward method for this particular task. Using a Pivot Table, I was able to process all 149,000 rows of data almost instantly. (The official Maven video walks through this with step-by-step clicks.)

- For period-over-period analysis, use the '% Difference From' option to see the previous month data.

Month	store	Total Sales	Month-over-Month
Jan	Astoria	$27,314	
Jan	Hell's Kitchen	$27,821	
Jan	Lower Manhattan	$26,543	
Feb	Astoria	$25,105	($2,208)
Feb	Hell's Kitchen	$25,720	($2,101)
Feb	Lower Manhattan	$25,320	($1,223)
Mar	Astoria	$32,835	$7,730
Mar	Hell's Kitchen	$33,111	$7,391
Mar	Lower Manhattan	$32,889	$7,569
Apr	Astoria	$39,478	$6,642
Apr	Hell's Kitchen	$40,304	$7,194
Apr	Lower Manhattan	$39,159	$6,271

2. Python 🥈

- Pandas 🐼 library in Jupyter Notebook was surprisingly fast and efficient. I believe it could be ranked second in terms of ease and speed for this task.

  ▶️ Learn more about Python & Pandas with Chris Bruehl's video.

```
monthly_sales = (
        df
       .groupby(['store',
      df['date'].dt.month])['sales']
       .sum()
       .reset_index()
       .rename(columns={'date': 'month'})
      )
```

- **Calculate Difference:** grouped with pandas and applied .diff()
  `monthly_sales['sales_vs_last_month'] = monthly_sales.groupby('store')['sales'].diff()`

	store	month	sales	sales_vs_last_month
0	Astoria	1	27313.66	NaN
1	Astoria	2	25105.34	-2208.32
2	Astoria	3	32835.43	7730.09
3	Astoria	4	39477.61	6642.18
4	Astoria	5	52428.76	12951.15
5	Astoria	6	55083.11	2654.35

3. MySQL 🥉

- With MySQL, the key was using a **Window Function** to look back at the previous month's sales for each store.
  - Want to know more about Window Functions and CTE ? See Alice Zhao's video
- **LOAD DATA INFILE** is so much faster than using **Table Data Import Wizard**
- Using **Common Table Expression (CTE)** can keep the query clean and more efficient than repeating aggregation:
  - The LAG() function allowed me to fetch the prior month's sales total within the same row
  - **`SUM(sales) - LAG(SUM(sales)) OVER (PARTITION BY store ORDER BY MONTH(date)) AS MoM`**
- For better display, I formatted the numbers by using **`CONCAT('$', FORMAT(ROUND(SUM(sales), 0), 0))`**

```
SELECT MONTHNAME(date) AS Month, Store, CONCAT('$', FORMAT(ROUND(SUM(sales), 0), 0)) AS Total,
CONCAT('$', FORMAT(ROUND(SUM(sales) - LAG(SUM(sales)) OVER (PARTITION BY store ORDER BY MONTH(date)), 0), 0)) AS MoM
FROM sales_data
GROUP BY MONTHNAME(date), MONTH(date), Store
ORDER BY MONTH(date), Store;
```

Month	Store	Total	MoM
January	Astoria	$27,314	NULL
January	Hell's Kitchen	$27,821	NULL
January	Lower Manhattan	$26,543	NULL
February	Astoria	$25,105	$-2,208
February	Hell's Kitchen	$25,720	$-2,101
February	Lower Manhattan	$25,320	$-1,223
March	Astoria	$32,835	$7,730
March	Hell's Kitchen	$33,111	$7,391
March	Lower Manhattan	$32,889	$7,569
April	Astoria	$39,478	$6,642
April	Hell's Kitchen	$40,304	$7,194
April	Lower Manhattan	$39,159	$6,271
May	Astoria	$52,429	$12,951
May	Hell's Kitchen	$52,599	$12,295
May	Lower Manhattan	$51,700	$12,541
June	Astoria	$55,083	$2,654
June	Hell's Kitchen	$56,957	$4,358
June	Lower Manhattan	$54,446	$2,746

4. Power BI

- Check Data Types
- Create a calendar table based on the date fields
- **Calendar Table Query:** This code generates a dynamic calendar (date) table that spans all dates present in data, from the earliest to the latest, *with no gaps*. Remember to mark the calendar table as Date Table for DAX to work right.

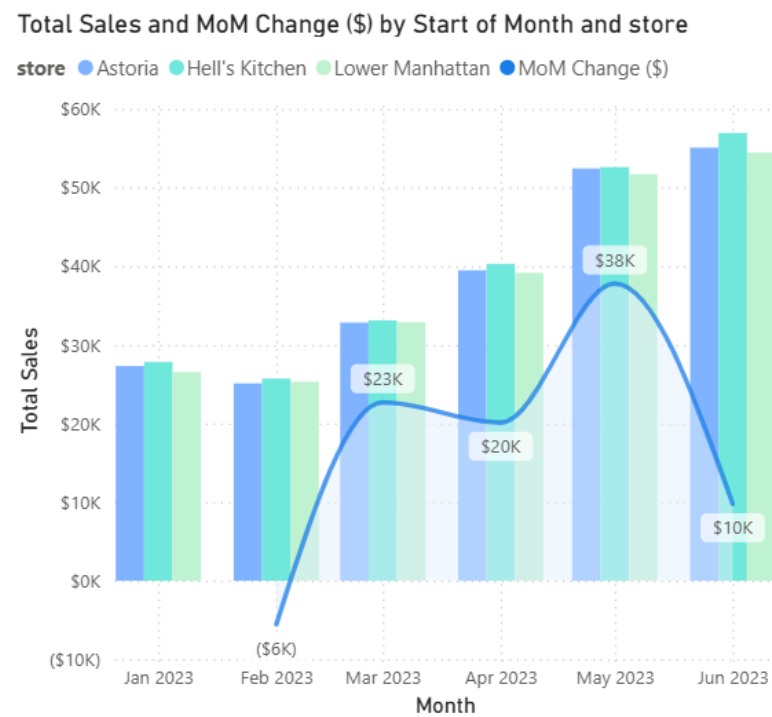
```
let
    // Reference & update based on data source name
    Source = #"coffee_shop_sales",
    Dates = Source[date],

    // Get the earliest and latest dates in the data
    StartDate = Date.StartOfMonth(List.Min(Dates)),
    EndDate = Date.EndOfMonth(List.Max(Dates)),

    // Create a list of all dates between StartDate and EndDate
    DateList = List.Dates(
        StartDate,
        Duration.Days(EndDate - StartDate) + 1,
        #duration(1, 0, 0, 0)
    ),

    // Convert the list into a table with a single Date column
    Calendar = Table.FromList(DateList, Splitter.SplitByNothing(), {"Date"}),

    // Set the Date column type
    #"Change Type" = Table.TransformColumnTypes(Calendar, {{"Date", type date}})
in
    #"Change Type"
```

- **Data Model**: Connect new calendar table to the sales data table
- **Matrix** for Month and Store
- Create a DAX measure to calculate *Total Sales* and *Previous month sales*
- **Measures**: Alt + H + NM
    - **`Total Sales = SUM(coffee_shop_sales[sales])`**
    - **`MoM Change:`** `use VAR + DATEADD() for clean month-over-month measures`

```
MoM Change ($) =
VAR CurrentSales = [Total Sales]
VAR PriorSales =
CALCULATE(
    [Total Sales],
    DATEADD(
        CalendarTable[Date],
        -1,
        MONTH
    )
)
RETURN
IF(
    ISBLANK(PriorSales),
    "-",
    CurrentSales - PriorSales
)
```


- Although Power BI presented the biggest challenge for this analysis, its strength lies in creating a powerful visual impact. This chart summarizes the month-over-month sales performance, with bars indicating total sales per store and a line tracking the monthly revenue changes.

Start of Month	store	Total Sales	MoM Change ($)
Jan 2023	Astoria	$27,314	-
	Hell's Kitchen	$27,821	-
	Lower Manhattan	$26,543	-
Feb 2023	Astoria	$25,105	($2,208)
	Hell's Kitchen	$25,720	($2,101)
	Lower Manhattan	$25,320	($1,223)
Mar 2023	Astoria	$32,835	$7,730
	Hell's Kitchen	$33,111	$7,391
	Lower Manhattan	$32,889	$7,569
Apr 2023	Astoria	$39,478	$6,642
	Hell's Kitchen	$40,304	$7,194
	Lower Manhattan	$39,159	$6,271
May 2023	Astoria	$52,429	$12,951
	Hell's Kitchen	$52,599	$12,295
	Lower Manhattan	$51,700	$12,541
Jun 2023	Astoria	$55,083	$2,654
	Hell's Kitchen	$56,957	$4,358
	Lower Manhattan	$54,446	$2,746



Key Takeaway

These approaches show how choosing the right platform can turn messy data into clear, actionable insights—no matter your tool of choice. Excel PivotTables for lightning-fast checks, pandas for heavy lifting, MySQL CTEs for seamless database reporting, and Power BI for compelling visuals. Mix and match these methods to speed up your workflow and get the insights you need.

Prepared by

Amirhossein Tonekaboni
SAP Business One ERP Consultant

Linkedin.com/in/Tonekaboni
Allmylinks.com/tonekaboni