MAVEN DATA DRILL #4

SPOT THE SALE

Date Range Matching

SQL
Excel
Python
Power BI

Notes by
Amirhossein Tonekaboni

Business Data Analyst
SAP Business One ERP Consultant

Linkedin.com/in/Tonekaboni
Atonekaboni.github.io

Overview

Learning notes from Maven Analytics Data Drill #4: matching sales orders to active promotional periods using date range analysis. This document compares solution approaches across **Excel**, **MySQL**, **Power BI**, and **Python** based on the expert walkthrough video, highlighting key differences in implementation. Additionally includes an **alternative** Python approach.

Dataset

The dataset consists of two tables that need to be joined based on date ranges:

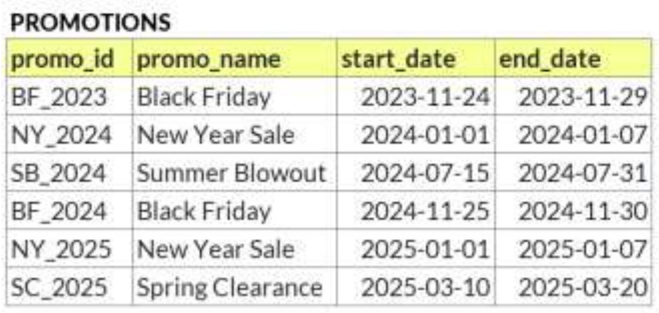
- **Orders:** Contains order transactions with order_id, order_date, and order_quantity
- **Promotions:** Contains promotional campaigns with promo_id, promo_name, start_date, and end_date

Data Drill (4) Spot the Sale

Solution - YouTube

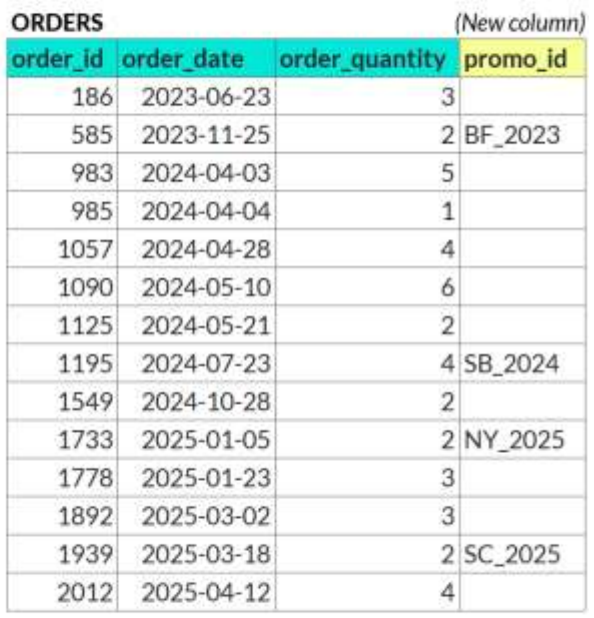
GitHub Repository

Example:



PROMOTIONS

promo_id	promo_name	start_date	end_date
BF_2023	Black Friday	2023-11-24	2023-11-29
NY_2024	New Year Sale	2024-01-01	2024-01-07
SB_2024	Summer Blowout	2024-07-15	2024-07-31
BF_2024	Black Friday	2024-11-25	2024-11-30
NY_2025	New Year Sale	2025-01-01	2025-01-07
SC_2025	Spring Clearance	2025-03-10	2025-03-20



ORDERS *(New column)*

order_id	order_date	order_quantity	promo_id
186	2023-06-23	3	
585	2023-11-25	2	BF_2023
983	2024-04-03	5	
985	2024-04-04	1	
1057	2024-04-28	4	
1090	2024-05-10	6	
1125	2024-05-21	2	
1195	2024-07-23	4	SB_2024
1549	2024-10-28	2	
1733	2025-01-05	2	NY_2025
1778	2025-01-23	3	
1892	2025-03-02	3	
1939	2025-03-18	2	SC_2025
2012	2025-04-12	4	

Objective

Match each order to its corresponding active promotion based on whether the order_date falls within the promotion's date range (start_date to end_date).

TOOLS	SOLUTION

MYSQL	
	```
	SELECT o.*, p.promo_id
	FROM orders o
	LEFT JOIN promotions p
	ON o.order_date BETWEEN p.start_date AND p.end_date
	-- WHERE promo_id is NULL
	```

EXCEL	➡ Formulas – Create from Selection `FILTER(promo_id, (start_date<=B2)*(end_date>=B2), "")`

PYTHON	
	```
	# merge_asof (dataset should be sorted!)

	pd.merge_asof(
	    orders.sort_values("order_date"),
	    promotions.sort_values("start_date"),
	    left_on="order_date",
	    right_on="start_date",
	    direction="backward"
	).query("order_date <= end_date")
	```

Alternative Solution	
	```
	# iterrows (Looping over DataFrame rows, Slower!)

	def promo(order_date):
	    for _, p in promotions.iterrows():
	        if p.start_date <= order_date <= p.end_date:
	            return p.promo_id
	    return None

	orders['promo_id'] = orders.order_date.apply(promo)
	orders[orders.promo_id.notnull()]
	```

POWER BI	
	```
	promo_id =
	    VAR o = orders[order_date]
	    RETURN
	    CALCULATE(
	        MAX(promotions[promo_id]), -- pick highest if overlap
	        FILTER(
	            ALL(promotions), -- ignore filters
	            promotions[start_date] <= o && promotions[end_date] >= o
	        )
	    )
	```

Key Takeaway

Tool Strengths

- **SQL:** Simplest approach - LEFT **JOIN** with **BETWEEN** clause. Would be my choice for this type of data problem and the best for large datasets.
- **Excel:** FILTER function was much easier than lookup formulas. Still king for quick analysis and exploration.
- **Python:** iterrows() approach was more readable, even though it's slower than **merge_asof**. Provides flexibility but not as easy as SQL here.
- **Power BI:** DAX syntax with Excel-like **CALCULATE / FILTER** functions and advanced context control.

Real-world Application

Date range matching like this shows up everywhere in business analysis - **campaign attribution, employee performance reviews, seasonal trend analysis**.

Prepared by

Amirhossein Tonekaboni
Business Data Analyst
SAP Business One ERP Consultant

Linkedin.com/in/Tonekaboni
Atonekaboni.github.io