ChatGPT-o1 Preview - Problem 3

Write a DAX measure that calculates a rolling 5-week average of sales, but only including weeks where sales were above the overall median weekly sales across all fiscal years.

Assume each new fiscal year starts on Oct 1 (for example FY2023 would start on Oct 1, 2022.

Assume weeks start on Mondays.

Here's the Sales data from a table called "Problem 3"
https://raw.githubusercontent.com/bjulius/GPT-o1Preview/main/problem3.csv

Certainly! Here's a DAX measure that calculates the rolling 5-week average of sales, including only weeks where the weekly sales were above the overall median weekly sales across all fiscal years. This measure assumes you have a Date table with a relationship to your "Problem 3" table, and that your Date table has a [Week Start Date] column where weeks start on Mondays, and a [Fiscal Year] column that starts on October 1.

DAX

```
Rolling 5-Week Average (Above Median) =
VAR MedianWeeklySales =
    CALCULATE(
        MEDIANX(
            VALUES('Date'[Week Start Date]),
            CALCULATE(SUM('Problem 3'[Sales]))
        ),
        ALL('Date')
    )
VAR CurrentWeekStartDate =
    MAX('Date'[Week Start Date])
VAR StartDate =
    CurrentWeekStartDate - 28
VAR Past5Weeks =
    FILTER(
        ALL('Date'),
        'Date'[Week Start Date] >= StartDate &&
        'Date'[Week Start Date] <= CurrentWeekStartDate
    )
VAR WeeksAboveMedian =
    FILTER(
        Past5Weeks,
        CALCULATE(SUM('Problem 3'[Sales])) > MedianWeeklySales
    )
VAR TotalSales =
    CALCULATE(
        SUM('Problem 3'[Sales]),
        WeeksAboveMedian
    )
VAR WeekCount =
    DISTINCTCOUNT(WeeksAboveMedian[Week Start Date])
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Explanation:

1. **MedianWeeklySales**: Calculates the overall median weekly sales across all fiscal years.
2. **CurrentWeekStartDate**: Identifies the current week's start date in context.
3. **StartDate**: Calculates the date 28 days (4 weeks) prior to the current week to cover a total of 5 weeks.
4. **Past5Weeks**: Filters all dates to get the past 5 weeks based on the week start dates.
5. **WeeksAboveMedian**: Filters those weeks where the weekly sales are above the median.
6. **TotalSales** and **WeekCount**: Sum the sales and count the weeks from the filtered weeks.
7. **Return**: Calculates the average by dividing **TotalSales** by **WeekCount**. If there are no weeks that meet the criteria, it returns BLANK.

Make sure your Date table is properly set up with the correct week start dates and fiscal year logic to ensure accurate calculations.

Rewrite assuming Date table is named Dates, Fact table is named Sales, the sales amount field is "Line Sales" and there is a Date Table with the "Week Start Date" field

Certainly! Here's the revised DAX measure using your specified table and field names:

DAX

```
Rolling 5-Week Average (Above Median) =
VAR MedianWeeklySales =
    CALCULATE(
        MEDIANX(
            VALUES('Dates'[Week Start Date]),
            CALCULATE(SUM('Sales'[Line Sales]))
        ),
        ALL('Dates')
    )
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
VAR StartDate =
    CurrentWeekStartDate - 28
VAR Past5Weeks =
    FILTER(
        ALL('Dates'),
        'Dates'[Week Start Date] >= StartDate &&
        'Dates'[Week Start Date] <= CurrentWeekStartDate
    )
VAR WeeksAboveMedian =
    FILTER(
        Past5Weeks,
        CALCULATE(SUM('Sales'[Line Sales])) > MedianWeeklySales
    )
VAR TotalSales =
    CALCULATE(
        SUM('Sales'[Line Sales]),
        WeeksAboveMedian
    )
VAR WeekCount =
    DISTINCTCOUNT(WeeksAboveMedian[Week Start Date])
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Explanation:

1. **MedianWeeklySales**: Calculates the overall median of weekly sales across all fiscal years.

   DAX

   ```
   MedianWeeklySales =
       CALCULATE(
           MEDIANX(
               VALUES('Dates'[Week Start Date]),
               CALCULATE(SUM('Sales'[Line Sales]))
           ),
           ALL('Dates')
       )
   ```

   2. **MEDIANX**: Iterates over each unique week start date and calculates the sum of 'Line Sales' for that week.
   3. **ALL('Dates')**: Ensures that the calculation considers all dates across all fiscal years.

4. **CurrentWeekStartDate**: Identifies the current context's week start date.

   DAX

   ```
   CurrentWeekStartDate = MAX('Dates'[Week Start Date])
   ```

5. **StartDate**: Determines the start date 28 days before the current week to cover a total of 5 weeks (including the current week).

   DAX

   ```
   StartDate = CurrentWeekStartDate - 28
   ```

6. **Past5Weeks**: Filters the 'Dates' table to include only the past 5 weeks.

   DAX

   ```
   Past5Weeks =
       FILTER(
           ALL('Dates'),
           'Dates'[Week Start Date] >= StartDate &&
           'Dates'[Week Start Date] <= CurrentWeekStartDate
       )
   ```

7. **WeeksAboveMedian**: Filters the weeks where the total 'Line Sales' is above the median weekly sales.

   DAX

   ```
   WeeksAboveMedian =
       FILTER(
           Past5Weeks,
           CALCULATE(SUM('Sales'[Line Sales])) > MedianWeeklySales
       )
   ```

8. **TotalSales**: Calculates the total 'Line Sales' for the filtered weeks.

   DAX

   ```
   TotalSales =
       CALCULATE(
           SUM('Sales'[Line Sales]),
           WeeksAboveMedian
       )
   ```

9. **WeekCount**: Counts the number of weeks that met the criteria.

   DAX

   ```
   WeekCount = DISTINCTCOUNT(WeeksAboveMedian[Week Start Date])
   ```

10. **Return Statement**: Calculates the average by dividing **TotalSales** by **WeekCount**. If no weeks meet the criteria, it returns BLANK.

    DAX

    ```
    RETURN
    IF(
        WeekCount > 0,
        TotalSales / WeekCount,
        BLANK()
    )
    ```

Notes:

- Ensure your 'Dates' table correctly identifies fiscal years starting on October 1 and weeks starting on Mondays.

- The measure filters weeks where sales are above the overall median weekly sales, as per your requirement.
- The calculation considers a rolling window of 5 weeks, adjusting for any weeks where sales didn't meet the criteria.

Additional Considerations:

- **Fiscal Year Handling**: If your 'Dates' table includes a [Fiscal Year] column, you can modify the measure to respect fiscal year boundaries if needed.
- **Week Calculation**: Make sure that the [Week Start Date] in your 'Dates' table correctly aligns weeks starting on Mondays.

Example Usage in a Report:

- Place this measure in a table or chart alongside the [Week Start Date] from your 'Dates' table to visualize the rolling 5-week average.
- Use slicers or filters on the [Fiscal Year] or other relevant date fields to refine your analysis.

Feel free to adjust any part of the measure to better fit the specifics of your data model or to handle additional requirements.

Code threw the following error:
Table variable 'WeeksAboveMedian' cannot be used in current context because a base table is expected

```
GPTo1Preview 5Wk Rolling Average GT median =
VAR MedianWeeklySales =
CALCULATE(
MEDIANX(
VALUES('Dates'[Week Start Date]),
CALCULATE(SUM('Sales'[Line Total]))
),
ALL('Dates')
)
VAR CurrentWeekStartDate =
MAX('Dates'[Week Start Date])
VAR StartDate =
CurrentWeekStartDate - 28
VAR Past5Weeks =
FILTER(
ALL('Dates'),
'Dates'[Week Start Date] >= StartDate &&
'Dates'[Week Start Date] <= CurrentWeekStartDate
)
VAR WeeksAboveMedian =
FILTER(
Past5Weeks,
CALCULATE(SUM('Sales'[Line Total])) > MedianWeeklySales
)
VAR TotalSales =
CALCULATE(
SUM('Sales'[Line Total]),
WeeksAboveMedian
)
VAR WeekCount =
DISTINCTCOUNT(WeeksAboveMedian[Week Start Date])
RETURN
IF(
WeekCount > 0,
TotalSales / WeekCount,
BLANK()
)
```

Apologies for the inconvenience! The error occurs because table variables like `WeeksAboveMedian` cannot be used directly in functions that expect a base table, such as when referencing columns with `[ColumnName]`. To fix this, we need to adjust how we calculate `WeekCount` and possibly other parts.

Here's the corrected DAX measure:

DAX

```
GPTo1Preview 5Wk Rolling Average GT median =
VAR MedianWeeklySales =
    CALCULATE(
        MEDIANX(
            VALUES('Dates'[Week Start Date]),
            CALCULATE(SUM('Sales'[Line Total]))
        ),
        ALL('Dates')
    )
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
VAR StartDate =
    CurrentWeekStartDate - 28
VAR Past5Weeks =
    FILTER(
        ALL('Dates'),
        'Dates'[Week Start Date] >= StartDate
            && 'Dates'[Week Start Date] <= CurrentWeekStartDate
    )
VAR WeeksAboveMedian =
    FILTER(
        Past5Weeks,
        CALCULATE(SUM('Sales'[Line Total])) > MedianWeeklySales
    )
VAR TotalSales =
    CALCULATE(
        SUM('Sales'[Line Total]),
        WeeksAboveMedian
    )
VAR WeekCount =
    CALCULATE(
        DISTINCTCOUNT('Dates'[Week Start Date]),
        WeeksAboveMedian
    )
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Explanation of the Fix:

- **Issue:** The error `"Table variable 'WeeksAboveMedian' cannot be used in current context because a base table is expected"` arises because we cannot reference a column from a table variable in functions like `DISTINCTCOUNT` in that manner.
- **Solution:** Instead of `DISTINCTCOUNT(WeeksAboveMedian[Week Start Date])`, we use:

  DAX

  ```
  VAR WeekCount =
      CALCULATE(
          DISTINCTCOUNT('Dates'[Week Start Date]),
          WeeksAboveMedian
      )
  ```

  Here, `'Dates'[Week Start Date]` is a column from the base table `'Dates'`, and we apply the filter `WeeksAboveMedian` within `CALCULATE`. This way, we correctly count the distinct weeks that are in the `WeeksAboveMedian` table.

Detailed Step-by-Step Explanation:

① **MedianWeeklySales:**

DAX

```
VAR MedianWeeklySales =
    CALCULATE(
        MEDIANX(
            VALUES('Dates'[Week Start Date]),
            CALCULATE(SUM('Sales'[Line Total]))
        ),
        ALL('Dates')
    )
```

② Calculates the median of weekly sales across all dates.

③ Uses `ALL('Dates')` to remove any filters from the 'Dates' table, ensuring the median is calculated over all weeks.

④ **CurrentWeekStartDate:**

DAX

```
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
```

⑤ Gets the current context's week start date.

⑥ **StartDate:**

DAX

```
VAR StartDate =
    CurrentWeekStartDate - 28
```

⑦ Calculates the date 28 days before the current week to cover a 5-week period.

⑧ **Past5Weeks:**

DAX

```
VAR Past5Weeks =
    FILTER(
        ALL('Dates'),
        'Dates'[Week Start Date] >= StartDate
            && 'Dates'[Week Start Date] <= CurrentWeekStartDate
    )
```

⑨ Filters the 'Dates' table to include only the past 5 weeks.

⑩ **WeeksAboveMedian:**

DAX

```
VAR WeeksAboveMedian =
    FILTER(
        Past5Weeks,
        CALCULATE(SUM('Sales'[Line Total])) > MedianWeeklySales
    )
```

⑪ Filters the `Past5Weeks` to include only weeks where the total sales are above the median weekly sales.

⑫ **TotalSales:**

DAX

```
VAR TotalSales =
    CALCULATE(
        SUM('Sales'[Line Total]),
        WeeksAboveMedian
    )
```

⑬ Calculates the total sales for the weeks above the median.

⑭ **WeekCount:**

DAX

```
VAR WeekCount =
    CALCULATE(
        DISTINCTCOUNT('Dates'[Week Start Date]),
        WeeksAboveMedian
    )
```

⑮ Counts the number of distinct weeks in `WeeksAboveMedian`.

⑯ Uses `CALCULATE` to apply the `WeeksAboveMedian` filter to the base `'Dates'` table.

⑰ **Return Statement:**

DAX

```
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

⑱ Calculates the average sales by dividing `TotalSales` by `WeekCount`.

⑲ If `WeekCount` is zero (no weeks met the criteria), it returns `BLANK()`.

Additional Notes:

- **Why We Can't Reference `WeeksAboveMedian[Week Start Date]`:** Table variables like `WeeksAboveMedian` exist only within the measure and are not part of the data model. Therefore, we can't reference their columns directly in functions that expect columns from the data model (like `DISTINCTCOUNT`).
- **Using `CALCULATE` with Filters:** By using `CALCULATE`, we can apply the `WeeksAboveMedian` filter to functions operating on base tables.
- **Ensure Relationships are Correct:** For this measure to work, ensure that:
    - The 'Sales' table is properly related to the 'Dates' table via a date column.
    - The 'Dates' table has correct [Week Start Date] entries, with weeks starting on Mondays.
    - The fiscal year in the 'Dates' table starts on October 1.

Testing the Measure:

- **Visualizing Results:** Add this measure to a table or chart in your report, along with the 'Dates'[Week Start Date], to see the rolling 5-week average over time.
- **Debugging Tips:**
    - If you still encounter issues, try breaking down the measure and outputting intermediate variables to check their values.
    - For example, create separate measures to display `MedianWeeklySales`, `TotalSales`, and `WeekCount`.

Alternate Approach (Using COUNTROWS):

If you prefer, you can use `COUNTROWS` instead of `DISTINCTCOUNT`, since `WeeksAboveMedian` already filters down to the distinct weeks:

DAX

```
VAR WeekCount =
    COUNTROWS(
        SUMMARIZE(
            WeeksAboveMedian,
            'Dates'[Week Start Date]
        )
    )
```

Explanation:

- `SUMMARIZE` creates a table of distinct 'Week Start Date' values from `WeeksAboveMedian`.
- `COUNTROWS` counts the number of rows in this summarized table.

Implementing the Alternate Approach:

Replace the `WeekCount` variable with the above code if you encounter any issues with the previous method.

Final Thoughts:

- Always ensure that when using table variables in DAX, you handle them appropriately within the context of functions that can accept them.
- When in doubt, use `CALCULATE` to apply filters from table variables to base tables or use functions like `COUNTROWS` with `SUMMARIZE` to work around limitations.

Let me know if you have any more questions or need further assistance!

when that measure is dropped in a table or line chart against week start date, it just returns blanks in each row

I apologize for the continued trouble. The issue arises because when the measure is placed in a visual against `Week Start Date`, the context is being overly filtered, causing variables like `MedianWeeklySales` and `WeeksAboveMedian` to return unexpected results or blanks. Let's adjust the measure to handle the context appropriately.

Here's the revised measure:

DAX

```
GPTo1Preview 5Wk Rolling Average GT median =
VAR MedianWeeklySales =
    CALCULATE(
        MEDIANX(
            SUMMARIZE(
                ALL('Dates'),
                'Dates'[Week Start Date],
                "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
            ),
            [WeeklySales]
        )
    )
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
VAR Past5Weeks =
    CALCULATETABLE(
        TOPN(
            5,
            FILTER(
                ALL('Dates'),
                'Dates'[Week Start Date] <= CurrentWeekStartDate
            ),
            'Dates'[Week Start Date],
            DESC
        ),
        ALL('Dates')
    )
VAR WeeksAboveMedian =
    FILTER(
        ADDCOLUMNS(
            Past5Weeks,
            "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
        ),
        [WeeklySales] > MedianWeeklySales
    )
VAR TotalSales =
    SUMX(WeeksAboveMedian, [WeeklySales])
VAR WeekCount =
    COUNTROWS(WeeksAboveMedian)
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Explanation of Adjustments:

1. **MedianWeeklySales:**
   2. Modified to use `ALL('Dates')` inside `SUMMARIZE` to ensure it considers all weeks, regardless of any filters applied by the visual.

      DAX

      ```
      VAR MedianWeeklySales =
          CALCULATE(
              MEDIANX(
                  SUMMARIZE(
                      ALL('Dates'),
                      'Dates'[Week Start Date],
                      "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
                  ),
                  [WeeklySales]
              )
          )
      ```

3. **Past5Weeks:**
   4. Used `CALCULATETABLE` with `ALL('Dates')` to remove filters and get the correct past 5 weeks regardless of the context from the visual.

      DAX

      ```
      VAR Past5Weeks =
          CALCULATETABLE(
              TOPN(
                  5,
                  FILTER(
                      ALL('Dates'),
                      'Dates'[Week Start Date] <= CurrentWeekStartDate
                  ),
                  'Dates'[Week Start Date],
                  DESC
              ),
              ALL('Dates')
          )
      ```

5. **WeeksAboveMedian:**
   6. Used `ADDCOLUMNS` to add weekly sales to each week in `Past5Weeks`.
   7. Filtered weeks where `WeeklySales` is greater than `MedianWeeklySales`.

      DAX

      ```
      VAR WeeksAboveMedian =
          FILTER(
              ADDCOLUMNS(
                  Past5Weeks,
                  "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
              ),
              [WeeklySales] > MedianWeeklySales
          )
      ```

8. **TotalSales and WeekCount:**
   9. Calculated `TotalSales` by summing `WeeklySales` in `WeeksAboveMedian`.
   10. Counted the number of weeks using `COUNTROWS`.

       DAX

       ```
       VAR TotalSales =
           SUMX(WeeksAboveMedian, [WeeklySales])
       VAR WeekCount =
           COUNTROWS(WeeksAboveMedian)
       ```

11. **Return Statement:**
    12. Returned the average by dividing `TotalSales` by `WeekCount` if `WeekCount` is greater than zero.

        DAX

        ```
        RETURN
        IF(
            WeekCount > 0,
            TotalSales / WeekCount,
            BLANK()
        )
        ```

Why the Original Measure Returned Blanks:

- **Context Filtering:** When you place the measure in a visual against `Week Start Date`, the context is filtered to that specific week. Variables like `MedianWeeklySales` and `Past5Weeks` were unintentionally being calculated within this narrow context, leading to no data or blanks.
- **Use of `ALL('Dates')`:** By using `ALL('Dates')` within the variables, we ensure that calculations like median and past weeks are not affected by the visual's filters.
- **Date Calculations:** Using date arithmetic (like subtracting 28 days) can be unreliable if there are missing dates or irregularities in your date table. By using `TOPN`, we get the last 5 available weeks based on `Week Start Date`.

Additional Considerations:

- **Ensure Proper Relationships:**
  - The 'Sales' table should have a relationship with the 'Dates' table via a date key.
  - The 'Dates' table must have continuous `Week Start Date` entries, with weeks starting on Mondays.
  - The fiscal year in the 'Dates' table should correctly start on October 1.
- **Handling Fiscal Years:**
  - If you need the measure to respect fiscal year boundaries, additional logic will be required to ensure weeks don't cross over into other fiscal years.
- **Testing Intermediate Results:**

    - To debug further, you can create separate measures to display values of `MedianWeeklySales`, `TotalSales`, and `WeekCount` to ensure they are calculating as expected.

Final Thoughts:

- **Robustness Against Visual Context:**
    - By carefully controlling the filter context using `ALL` and `CALCULATETABLE`, the measure becomes robust against the context applied by visuals.
- **Using `TOPN` Over Date Arithmetic:**
    - `TOPN` is preferred for getting the last N periods because it works based on the actual data in your table, avoiding issues with missing dates.
- **Avoiding Table Variable Limitations:**
    - By using functions like `ADDCOLUMNS` and `SUMX`, we can work with table variables effectively without running into context issues.

Example Usage in a Visual:

- **Line Chart or Table:**
    - Place `Week Start Date` from the 'Dates' table on the axis or rows.
    - Add the revised measure to display the rolling 5-week average of sales above the median.

Let me know if this revised measure resolves the issue or if you need further assistance!

still throwing all blanks

I apologize for the continued issues you're experiencing. The challenge arises due to the complex interplay of filter contexts when measures are placed in visuals like tables or line charts. Let's adjust the measure to correctly handle the context and ensure it returns the expected values.

Here's the revised DAX measure:

DAX

```
GPTo1Preview 5Wk Rolling Average GT median =
VAR MedianWeeklySales =
    MEDIANX(
        SUMMARIZE(
            ALL('Dates'[Week Start Date]),
            'Dates'[Week Start Date],
            "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
        ),
        [WeeklySales]
    )
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
VAR Last5WeekDates =
    TOPN(
        5,
        FILTER(
            ALLSELECTED('Dates'),
            'Dates'[Week Start Date] <= CurrentWeekStartDate
        ),
        'Dates'[Week Start Date], DESC
    )
VAR Last5WeeksWithSales =
    SUMMARIZE(
        Last5WeekDates,
        'Dates'[Week Start Date],
        "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
    )
VAR WeeksAboveMedian =
    FILTER(
        Last5WeeksWithSales,
        [WeeklySales] > MedianWeeklySales
    )
VAR TotalSales =
    SUMX(WeeksAboveMedian, [WeeklySales])
VAR WeekCount =
    COUNTROWS(WeeksAboveMedian)
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Explanation of Adjustments:

① **MedianWeeklySales:**

② We calculate the median weekly sales across all weeks, removing any filters on `'Dates'[Week Start Date]` but keeping other filters like fiscal year.

DAX

```
VAR MedianWeeklySales =
    MEDIANX(
        SUMMARIZE(
            ALL('Dates'[Week Start Date]),
            'Dates'[Week Start Date],
            "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
        ),
        [WeeklySales]
    )
```

③ **Note:** Using `ALL('Dates'[Week Start Date])` ensures we're considering all weeks for the median, but we keep other filters like fiscal year intact.

④ **CurrentWeekStartDate:**

⑤ Retrieves the current week's start date from the context.

DAX

```
VAR CurrentWeekStartDate =
    MAX('Dates'[Week Start Date])
```

⑥ **Last5WeekDates:**

⑦ We get the last 5 weeks up to the current week, respecting any filters applied to the 'Dates' table (e.g., fiscal year filters from slicers).

DAX

```
VAR Last5WeekDates =
    TOPN(
        5,
        FILTER(
            ALLSELECTED('Dates'),
            'Dates'[Week Start Date] <= CurrentWeekStartDate
        ),
        'Dates'[Week Start Date], DESC
    )
```

  ⑧ **Explanation:**
    ⑨ `ALLSELECTED('Dates')`: Removes filters applied by the visual but keeps filters from slicers or report-level filters.

    ⑩ This ensures we don't unintentionally remove important context like fiscal years.

⑪ **Last5WeeksWithSales:**

  ⑫ For the last 5 weeks, we calculate the weekly sales.

DAX

```
VAR Last5WeeksWithSales =
    SUMMARIZE(
        Last5WeekDates,
        'Dates'[Week Start Date],
        "WeeklySales", CALCULATE(SUM('Sales'[Line Total]))
    )
```

  ⑬ **Explanation:**
    ⑭ `SUMMARIZE` creates a table with 'Week Start Date' and 'WeeklySales'.

    ⑮ `CALCULATE(SUM('Sales'[Line Total]))` computes the sales for each week.

⑯ **WeeksAboveMedian:**

  ⑰ Filters the weeks where the weekly sales are above the median weekly sales.

DAX

```
VAR WeeksAboveMedian =
    FILTER(
        Last5WeeksWithSales,
        [WeeklySales] > MedianWeeklySales
    )
```

⑱ **TotalSales and WeekCount:**

  ⑲ Calculates the total sales and counts the number of weeks that met the criteria.

DAX

```
VAR TotalSales =
    SUMX(WeeksAboveMedian, [WeeklySales])
VAR WeekCount =
    COUNTROWS(WeeksAboveMedian)
```

⑳ **Return Statement:**

  ㉑ Computes the average sales for the weeks above the median.

DAX

```
RETURN
IF(
    WeekCount > 0,
    TotalSales / WeekCount,
    BLANK()
)
```

Why This Measure Should Work:

- **Context Preservation:** By using `ALLSELECTED('Dates')`, we ensure that filters applied by visuals (like 'Week Start Date' on the axis) don't interfere with our calculations, while still respecting filters from slicers (like fiscal year filters).
- **Avoiding Blanks:** The previous measures might have returned blanks due to over-filtering or context issues. This revised measure carefully manages the filter context to prevent unintended blank results.

Testing the Measure:

- **Visual Setup:**
  - Create a table or line chart.
  - Place `'Dates'[Week Start Date]` on the axis or rows.
  - Add the measure **GPTo1Preview 5Wk Rolling Average GT median** to the values.
- **Expected Outcome:**
  - The measure should display the rolling 5-week average of sales where weekly sales are above the median, for each week in your data.

Additional Notes:

- **Fiscal Year Considerations:**
  - Ensure that your 'Dates' table correctly handles fiscal years starting on October 1.
  - If your data spans multiple fiscal years, and you want the measure to consider only weeks within the same fiscal year, additional adjustments might be necessary.
- **Date Table Requirements:**
  - The 'Dates' table should have continuous 'Week Start Date' entries, with weeks starting on Mondays.
  - Ensure there are no gaps in the weeks unless intentionally designed that way.
- **Data Model Relationships:**
  - Verify that the 'Sales' table is correctly related to the 'Dates' table via the appropriate date key.
  - The relationship should be active and correctly configured (e.g., one-to-many from 'Dates' to 'Sales').
- **Debugging Tips:**
  - **Intermediate Measures:**
    - Create temporary measures to output the values of variables like `MedianWeeklySales`, `TotalSales`, and `WeekCount` to validate their correctness.
  - **Data Verification:**
    - Check the underlying data to ensure that sales amounts are present for the expected weeks.
    - Verify that the 'Dates' table covers all necessary dates and weeks.
- **Adjusting for Specific Requirements:**
  - If you need to adjust the measure to handle additional filters or to change how the weeks are calculated (e.g., handling missing weeks), further modifications may be needed.

    - For example, if you want the measure to reset at the start of each fiscal year, you could add additional filters to the `FILTER` functions to limit weeks to the current fiscal year.

Final Thoughts:

- **Measure Sensitivity to Context:**
    - DAX measures can be highly sensitive to the context in which they're evaluated.
    - Carefully controlling the filter context using functions like `ALL`, `ALLSELECTED`, and `CALCULATE` is crucial to obtaining accurate results.
- **Testing and Validation:**
    - Always test measures in the context of the visuals where they'll be used.
    - Validate intermediate results to ensure each part of the measure works as intended.

Let me know if this revised measure resolves the issue or if you need further assistance!

From ChatGPT, designed with FancyGPT