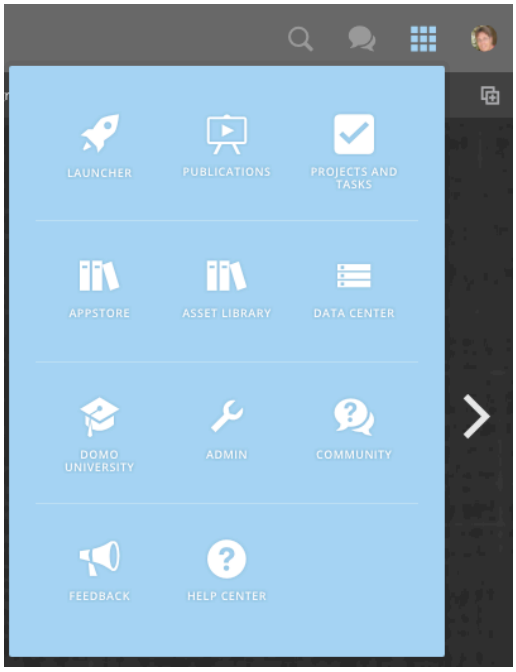
Advanced Platform Session

The best way to do these things where you get back from the conference:



Access a vibrant online community of customer peers, partners and Domo engineers around the globe. Includes answers to most common questions, often with examples, syntax and more.

Complete, up-to-date documentation with embedded how-to videos and syntax

Beast Mode Samples:

- *Many similar syntax examples in online documentation referenced above*

NOTE: The below is sample syntax- not officially released Domo code. This may not accommodate every use case, and may need to be tweaked for your unique use cases and requirements.

1. ***Dimension Based on Other Dimension Values:***

```
CASE

  WHEN `Measure` = 'Sales'

  THEN 'Sales'
```

```
  WHEN  `Measure` = 'Transactions'

  THEN 'Transactions'

END
```

2. Dimension Based on Measure Values:

```
CASE

  WHEN  `Actual Revenue` >= 2500

  THEN 'Tier 1 Customer'

  WHEN  `Actual Revenue` >= 1000

  THEN 'Tier 2 Customer'

  ELSE 'Tier 3 Customer'

END
```

3. Custom Time Series Dimension with Dynamic Time Horizons:

```
CASE

  WHEN ((DateDiff(AddDate(Current_Date(),-1), `date`) < 28)

       AND (DateDiff(Current_Date(), `date`) > 0))

  THEN 'Last 28 Days'

  WHEN ((DateDiff(AddDate(Current_Date(),-1), `date`) < (28 + 28))

     AND (DateDiff(Current_Date(), `date`) > 28))

  THEN '4 Weeks Prior'

  WHEN ((DateDiff(AddDate(Current_Date(),-1), `date`) < (28 + (52 * 7)))

    AND (DateDiff(Current_Date(), `date`) > (52 * 7)))

  THEN '52 Weeks Prior'
```

```
  ELSE ''
END
```

NOTE: Calcs syntax for Last 28 and other clauses embedded above can be reused in their own discrete Beast Mode Calcs.

4. Conditionally Rendered Measure (% Change - Orders):

```
CASE
  WHEN
    (sum((CASE
          WHEN ((DateDiff(AddDate(Current_Date(),-1), `date `) < (28 + (52 * 7)))
              AND (DateDiff(Current_Date(), `date `) > (52 * 7)))
          THEN  `orders `
          END ))=0)
  THEN 0
  ELSE ((sum((
    CASE
    WHEN ((DateDiff(AddDate(Current_Date(),-1), `date `) < 28)
          AND (DateDiff(Current_Date(), `date `) > 0))
    THEN  `orders `
    END ))
      - sum((
       CASE
        WHEN ((DateDiff(AddDate(Current_Date(),-1), `date `) < (28 + (52 * 7)))
```

```
        AND (DateDiff(Current_Date(), `date `) > (52 * 7)))
       THEN  `orders `
      END )))
     / sum((
      CASE
        WHEN ((DateDiff(AddDate(Current_Date(),-1), `date `) < (28 + (52 * 7)))
        AND (DateDiff(Current_Date(), `date `) > (52 * 7)))
      THEN  `orders `
      END )))
END
```

5. Year-over-year Syntax Example

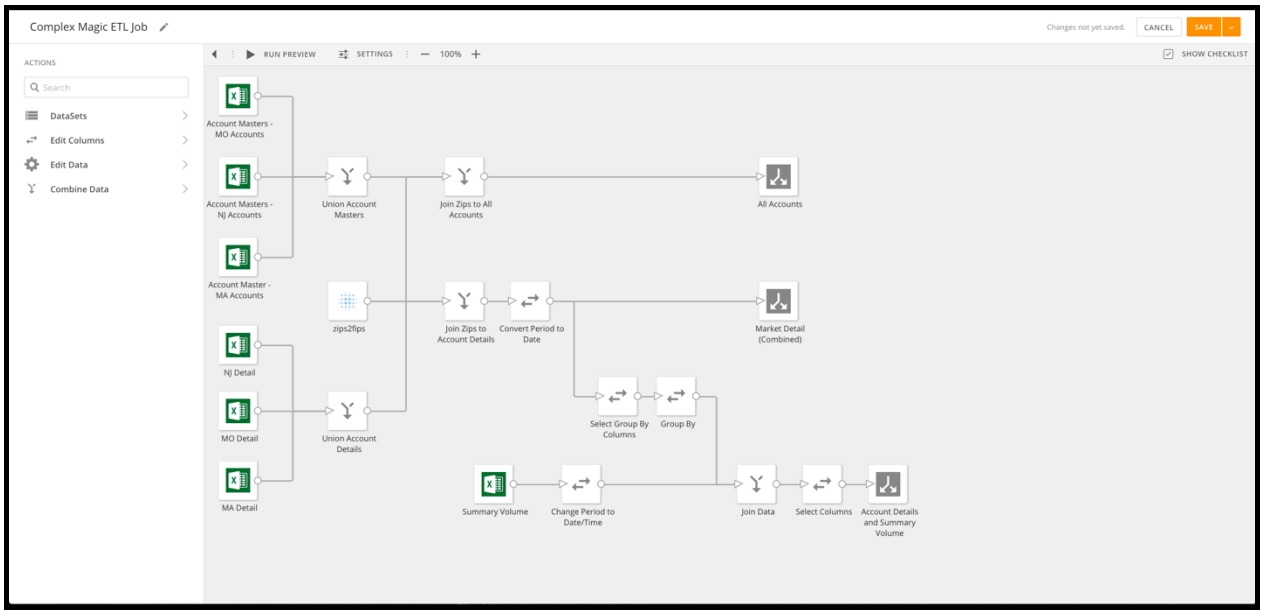
```
CASE
  WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < 28)
    AND (DateDiff(Current_Date(), `Date `) > 0))
  THEN 'Visits per day'
  WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < (28 + 28))
    AND (DateDiff(Current_Date(), `Date `) > 28))
  THEN '4 Weeks Prior'
  WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < (28 + (52 * 7)))
    AND (DateDiff(Current_Date(), `Date `) > (52 * 7)))
  THEN 'Visits per day 52 weeks ago'
```

```
    ELSE ''
  END
```

6. % Change Today vs Same Day Last Year:

```
 CASE
   WHEN (sum((
     CASE
       WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < (28 + (52 * 7)))
        AND (DateDiff(Current_Date(), `Date `) > (52 * 7)))
       THEN  `Daily Unique Visitors `
     END ))=0)
   THEN 0
   ELSE ((sum((
     CASE
       WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < 28)
         AND (DateDiff(Current_Date(), `Date `) > 0))
       THEN  `Daily Unique Visitors `
     END )) - sum((
     CASE
       WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < (28 + (52 * 7)))
         AND (DateDiff(Current_Date(), `Date `) > (52 * 7)))
       THEN  `Daily Unique Visitors `
     END ))) / sum((
     CASE
       WHEN ((DateDiff(AddDate(Current_Date(),-1), `Date `) < (28 + (52 * 7)))
         AND (DateDiff(Current_Date(), `Date `) > (52 * 7)))
       THEN  `Daily Unique Visitors `
     END )))
END
```

Sample Complex Magic ETL:



Sample Complex Magic SQL Dataflow Set up:

NOTE: As with the beast mode calcs syntax, the below is sample syntax- not officially released Domo code. This may not accommodate every use case, and may in fact need to be tweaked for your usage. In general, it works well though to pivot DataSets that need it.

- *Step 1* – Load provided sample dataset *LPrePivotData2.xls* into Domo and name it *LPrePivotData2*

- *Step 1* – Go to "Data Center" > Choose "DATAFLOWS" tab > select "+ NEW DATAFLOW"

- *Step2* – Under "Input DataSets" > choose "SELECT DATASET" and choose the dataset to be pivoted (leave the default dataset name as "*LPrePivotData2*")

- *Step 3* – Under "Transforms" > choose "+ADD TRANSFORM" and
    - *Step 3a* – Paste in the below SQL syntax
    - *Step 3b* – Name this step "*transpose_variables*"
    - *Step 3c* – Ensure that "*Generate Output Table*" *is checked*

```
Paste in Select '`Programs`' non_transpose_columns

, '`Date`, `Project Hours`' new_columns

, '``' excluded_columns

, '`lprepivotdata2`' source_table
```

- *Step 4* – Under "Transforms" > choose "+ADD TRANSFORM"
    - *Step 4a* – Paste in the below create procedure syntax
    - *Step 4b* – Ensure that "*Generate Output Table*" is **NOT** checked (this will also mean that you do not have to name it).

```
CREATE PROCEDURE column_transpose()

BEGIN

 DECLARE v_new_column VARCHAR(500);

 DECLARE v_excluded_columns VARCHAR(500);

 DECLARE v_non_transpose VARCHAR(500);

 DECLARE v_non_transpose_q VARCHAR(500);
```

```
DECLARE v_source_table VARCHAR(500);
DECLARE v_finished INTEGER DEFAULT 0;
DECLARE query_col VARCHAR(500);
DECLARE v_table_col VARCHAR(500);
DECLARE v_table_query VARCHAR(500);

DECLARE transpose_cursor CURSOR for SELECT COLUMN_NAME
FROM information_schema.COLUMNS
WHERE TABLE_SCHEMA= SCHEMA() AND TABLE_NAME=(select
replace(source_table,'`','') from transpose_variables);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;

SET @new_column = v_new_column;
SET @excluded_columns=v_excluded_columns;
SET @non_transpose=v_non_transpose;
SET @non_transpose_q=v_non_transpose_q;
SET @source_table=v_source_table;
SET @table_col=v_table_col;
SET @table_query=v_table_query;
```

```
-- Sets all the variables from the user defined transform

select non_transpose_columns from transpose_variables INTO @non_transpose;

select replace(excluded_columns, '`','\'') as excluded_columns from
transpose_variables INTO @excluded_columns;

select replace(non_transpose_columns, '`','\'') as non_transpose_q from
transpose_variables INTO @non_transpose_q;

select source_table as source_table from transpose_variables INTO
@source_table;


-- create new table based on the columns specified in the first transform

select replace(concat(non_transpose_columns,',',new_columns, ' varchar(500)'),
',', ' varchar(500),') as non_transpose_columns from transpose_variables into
@table_col;

SET @table_query = CONCAT('CREATE TABLE transpose (',@table_col,');');

PREPARE stmt FROM @table_query;

EXECUTE stmt;


 OPEN transpose_cursor;


 get_column: loop
```

```
-- Get the column name value from cursor
FETCH transpose_cursor INTO query_col;

-- Set variable to exist loop when cursor finishes looping through the columns
IF v_finished = 1 THEN LEAVE get_column;
END IF;

-- Set the cursor value so it can be inserted at it the new transpose column
SET @query_col=CONCAT('\'',query_col,'\'');

-- Set the cusrsor value with back ticks to pull the value in that column
SET @query_val=CONCAT('`',query_col,'`');

-- If statement to elimated cursor values that are columns that should be
excluded or not transposed (speficied in the 1st transform)
IF @non_transpose_q not like (concat('\'%',query_col,'%\'')) AND
@excluded_columns not like (concat('\'%',query_col,'%\'')) THEN
SET @ct_sql = CONCAT('INSERT INTO transpose (SELECT ',@non_transpose,', ',
@query_col, ', ',@query_val,' FROM ', @source_table ,')');
```

```
PREPARE stmt FROM @ct_sql;

EXECUTE stmt;

END IF;

END LOOP get_column;

CLOSE transpose_cursor;


END;
```

- *Step 5* – Under "Transforms" > choose "+ADD TRANSFORM"
    - *Step 5a* – Paste in the below *call stored procedure* syntax (one line)
    - *Step 5b* – Ensure that "*Generate Output Table*" is **NOT** checked (this will also mean that you do not have to name it).

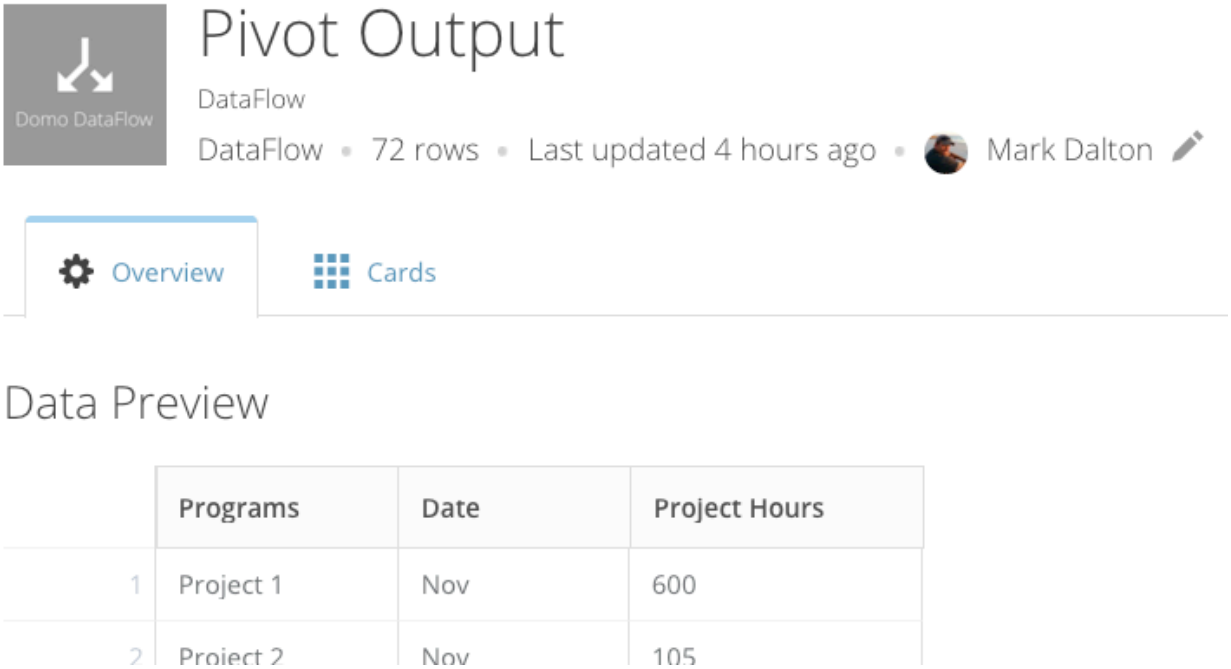
```
call column_transpose();
```

- *Step 6* – Under "Output DataSets" > choose "+ADD OUTPUT DATASETS"
    - *Step 4a* – Paste in the below SQL Select syntax (one line)
    - *Step 4b* – Name the output dataset "*Pivot Output*"

    NOTE: You may notice that the below SQL statement is referencing the table "transpose" even though we left "Generate Output Table" unchecked. In this case, the table of values is being generated and made available by the stored procedure, not the transform step output itself.

```
SELECT * FROM transpose
```

- *Step 7* – Go to the "DATASETS" tab in the "Data Center" to view the new output dataset "*Pivot Output*" and you should see the data, now pivoted like this:



	Programs	Date	Project Hours
1	Project 1	Nov	600
2	Project 2	Nov	105