Variance Analysis

Project

Presented by: Mohamed Selim

Overview

- What is ANOVA?
- Load Data into DB
- Data Discovering
- Data Preparation
- Load Data into BI
- Measures Creation
- Business Requirements
- Dashboard
- SQL Analysis.

ANOVA

Analysis of Variance AKA “ANOVA”

General concepts

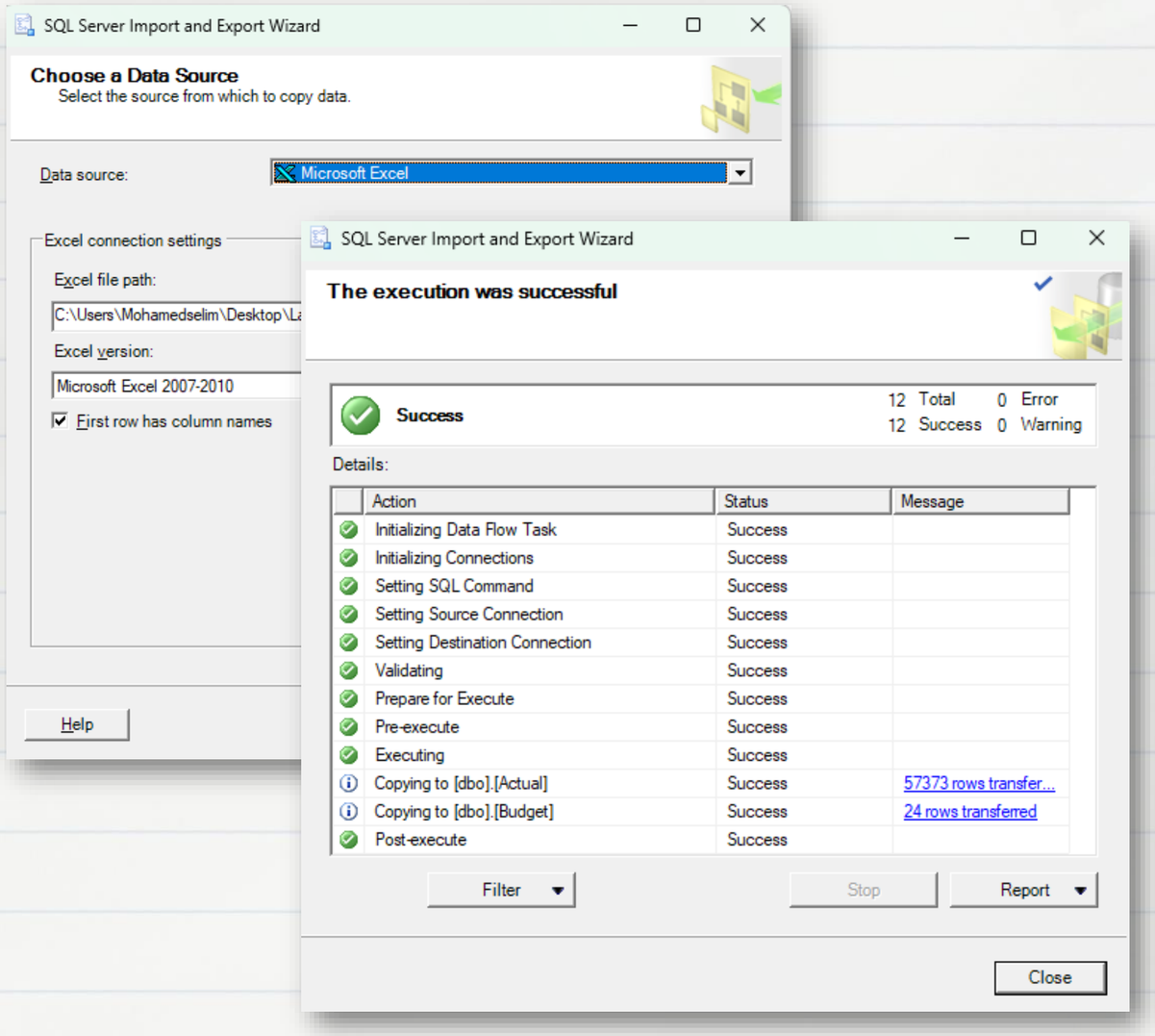
Variance Analysis is a financial and business analysis tool that examines the difference between actual performance and planned or expected performance. It is used to identify the reasons for discrepancies between budgets, forecasts, or standards and actual outcomes, helping organizations make informed decisions to improve their operations.

Variance Calculation

- Variance = Actual Value - Budgeted Value

Load The CSV file

```
CREATE DATABASE VarianceDB
USE VarianceDB
```



Data Discovering

```
SELECT * FROM budget;
SELECT * FROM actual;
```

Results | Messages

	EOMonth	Aspen	Carlota	Quad
1	1/31/2019	261026.28	165161.05	177625.63
2	2/28/2019	266139.2	205856.32	180197.12
3	3/31/2019	242084.62	229690.78	122247.42
4	4/30/2019	255810.62	242299.17	196508.89
5	5/31/2019	336485	207926.99	249977.68
6	6/30/2019	314705.98	239581.18	146357.82
7	7/31/2019	301014.26	184332.8	160984.85
8	8/31/2019	318504.4	210655.92	212605.12
9	9/30/2019	275060.39	172385.17	178230.74
10	10/31/2019	1375480.94	1110606.04	913136.07
11	11/30/2019	2830524.51	2506366.63	2036848.89
12	12/31/2019	2345748.75	2115293.08	1577168.89
13	1/31/2020	325463.31	240171.08	165238.34
14	2/29/2020	266471.58	164923.12	136111.22
15	3/31/2020	343068.28	171700.09	240491.86
16	4/30/2020	210544.84	216188.97	186988.46
17	5/31/2020	183812.48	212709.69	191078.15
18	6/30/2020	269180.63	284865.2	257248.81
19	7/31/2020	290873.23	227074.17	153180.66
20	8/31/2020	247731.24	266354.31	193006.78
21	9/30/2020	291061.68	262863.51	251355.99
22	10/31/2020	953278.45	779194.6	743120.17
23	11/30/2020	2542560.89	2231039.56	1778780.4
24	12/31/2020	3035612.89	2731748.74	1999713.39

	Date	Product	Sales
1	2020-10-23 00:00:00.000	Aspen	920.16
2	2020-12-13 00:00:00.000	Aspen	498.42
3	2019-12-21 00:00:00.000	Quad	131.85
4	2019-12-02 00:00:00.000	Quad	2215.08
5	2019-11-09 00:00:00.000	Aspen	316.31
6	2020-12-09 00:00:00.000	Aspen	747.63
7	2020-12-11 00:00:00.000	Carlota	365.81
8	2020-11-25 00:00:00.000	Aspen	747.63
9	2020-12-25 00:00:00.000	Aspen	1610.28
10	2019-12-11 00:00:00.000	Aspen	498.42
11	2019-12-23 00:00:00.000	Aspen	316.31
12	2019-10-23 00:00:00.000	Aspen	127.8
13	2019-12-16 00:00:00.000	Carlota	152.42
14	2019-03-18 00:00:00.000	Carlota	110.85
15	2019-11-14 00:00:00.000	Aspen	1150.2
16	2019-12-12 00:00:00.000	Aspen	1840.32
17	2020-11-18 00:00:00.000	Aspen	31.95
18	2019-04-14 00:00:00.000	Carlota	152.42
19	2019-03-17 00:00:00.000	Carlota	152.42
20	2019-11-10 00:00:00.000	Carlota	576.42
21	2020-09-05 00:00:00.000	Quad	87.9
22	2019-10-16 00:00:00.000	Carlota	365.81
23	2020-12-10 00:00:00.000	Aspen	498.42
24	2019-12-14 00:00:00.000	Quad	87.9
25	2020-12-22 00:00:00.000	Quad	2531.52
26	2019-11-11 00:00:00.000	Aspen	1150.2
27	2019-11-29 00:00:00.000	Quad	1582.2
28	2019-11-10 00:00:00.000	Carlota	365.81
29	2019-12-11 00:00:00.000	Aspen	95.85
30	2020-11-07 00:00:00.000	Aspen	1811.57
31	2020-12-04 00:00:00.000	Carlota	1064.16

Data Preparation

Unpivots columns into rows, converting non-tabular data into a tabular format.

```sql
DROP FUNCTION IF exists unpivot_budget;

CREATE FUNCTION unpivot_budget()
RETURNS @result TABLE (
  product NVARCHAR(50),
  EOMonth DATE,
  value NUMERIC(18, 2) )
AS BEGIN
  INSERT INTO @result
  SELECT 'Aspen' AS product, EOMonth, Aspen AS value
  FROM budget
  UNION ALL
  SELECT 'Carlota' AS product, EOMonth, Carlota AS value
  FROM budget
  UNION ALL
  SELECT 'Quad' AS product, EOMonth, Quad AS value
  FROM budget;
  RETURN;
END;

SELECT * FROM unpivot_budget();
```

Results | Messages

	product	EOMonth	value
1	Aspen	2019-01-31	261026.28
2	Aspen	2019-02-28	266139.20
3	Aspen	2019-03-31	242084.62
4	Aspen	2019-04-30	255810.62
5	Aspen	2019-05-31	336485.00
6	Aspen	2019-06-30	314705.98
7	Aspen	2019-07-31	301014.26
8	Aspen	2019-08-31	318504.40
9	Aspen	2019-09-30	275060.39
10	Aspen	2019-10-31	1375480.94
11	Aspen	2019-11-30	2830524.51
12	Aspen	2019-12-31	2345748.75
13	Aspen	2020-01-31	325463.31
14	Aspen	2020-02-29	266471.58
15	Aspen	2020-03-31	343068.28
16	Aspen	2020-04-30	210544.84
17	Aspen	2020-05-31	183812.48
18	Aspen	2020-06-30	269180.63
19	Aspen	2020-07-31	290873.23
20	Aspen	2020-08-31	247731.24
21	Aspen	2020-09-30	291061.68
22	Aspen	2020-10-31	953278.45
23	Aspen	2020-11-30	2542560.89
24	Aspen	2020-12-31	3035612.89
25	Carlota	2019-01-31	165161.05
26	Carlota	2019-02-28	205856.32
27	Carlota	2019-03-31	229690.78
28	Carlota	2019-04-30	242299.17

Data Preparation

Budget table

```sql
-- #2.2 Store the Function data in a new table (Tabular_table)
DROP TABLE IF EXISTS Tabular_budget;

CREATE TABLE Tabular_budget(
    Date DATE,
    Value NUMERIC(18, 2),
    Product NVARCHAR(50)
);

INSERT INTO Tabular_budget (Date, Value, Product)
SELECT eomonth, ROUND(SUM(value), 2) AS Total_budget, product
FROM unpivot_budget()
GROUP BY eomonth, product
ORDER BY eomonth, product;

SELECT * FROM Tabular_budget
```

Now we have a tabular table

	Date	Value	Product
1	2019-01-31	261026.28	Aspen
2	2019-02-28	266139.20	Aspen
3	2019-03-31	242084.62	Aspen
4	2019-04-30	255810.62	Aspen
5	2019-05-31	336485.00	Aspen
6	2019-06-30	314705.98	Aspen
7	2019-07-31	301014.26	Aspen
8	2019-08-31	318504.40	Aspen
9	2019-09-30	275060.39	Aspen
10	2019-10-31	1375480...	Aspen
11	2019-11-30	2830524...	Aspen
12	2019-12-31	2345748...	Aspen
13	2020-01-31	325463.31	Aspen
14	2020-02-29	266471.58	Aspen
15	2020-03-31	343068.28	Aspen
16	2020-04-30	210544.84	Aspen
17	2020-05-31	183812.48	Aspen
18	2020-06-30	269180.63	Aspen
19	2020-07-31	290873.23	Aspen
20	2020-08-31	247731.24	Aspen
21	2020-09-30	291061.68	Aspen
22	2020-10-31	953278.45	Aspen
23	2020-11-30	2542560...	Aspen
24	2020-12-31	3035612...	Aspen
25	2019-01-31	165161.05	Carlota
26	2019-02-28	205856.32	Carlota
27	2019-03-31	229690.78	Carlota
28	2019-04-30	242299.17	Carlota
29	2019-05-31	207926.99	Carlota
30	2019-06-30	239581.18	Carlota
31	2019-07-31	184332.80	Carlota
32	2019-08-31	210655.92	Carlota

Data Preparation

Actual table

```sql
-- #2.3 Handling the Actual Table
DROP TABLE IF EXISTS Tabular_Actual;

CREATE TABLE Tabular_Actual (
    Date DATE,
    Total_Sales NUMERIC(18, 2),
    Product NVARCHAR(50)
);

INSERT INTO Tabular_Actual (Date, Total_Sales, Product)
SELECT EOMONTH(date) AS Date, ROUND(SUM(sales), 2) AS Total_Sales,
Product
FROM actual
GROUP BY EOMONTH(date), Product
ORDER BY EOMONTH(date), Product;

SELECT * FROM Tabular_Actual
```

Results | Messages

	Date	Total_Sales	Product
1	2019-05-31	294545.00	Aspen
2	2019-06-30	179179.82	Quad
3	2020-04-30	240966.97	Carlota
4	2019-11-30	2915804.51	Aspen
5	2019-12-31	2076818.08	Carlota
6	2020-10-31	736777.17	Quad
7	2020-05-31	218412.69	Carlota
8	2019-11-30	2593056.63	Carlota
9	2020-04-30	229029.84	Aspen
10	2019-04-30	255106.62	Aspen
11	2019-02-28	182750.32	Carlota
12	2020-01-31	224586.08	Carlota
13	2020-09-30	204270.99	Quad
14	2019-01-31	149536.63	Quad
15	2020-08-31	214267.24	Aspen
16	2020-03-31	161877.86	Quad
17	2020-11-30	2224361.56	Carlota
18	2019-10-31	943606.07	Quad
19	2020-12-31	2701915.74	Carlota
20	2020-08-31	177787.78	Quad
21	2020-12-31	3049951.89	Aspen
22	2020-08-31	236020.31	Carlota
23	2019-02-28	260343.20	Aspen
24	2020-07-31	237512.17	Carlota
25	2019-03-31	243497.62	Aspen
26	2020-07-31	168228.66	Quad
27	2019-12-31	2436087.75	Aspen
28	2019-07-31	199273.85	Quad
29	2019-08-31	170893.12	Quad
30	2020-01-31	248027.31	Aspen
31	2019-05-31	224344.99	Carlota
32	2020-06-30	195078.81	Quad
33	2020-05-31	221176.48	Aspen
34	2019-04-30	234422.17	Carlota
35	2020-11-30	2634313.89	Aspen
36	2019-06-30	204462.18	Carlota

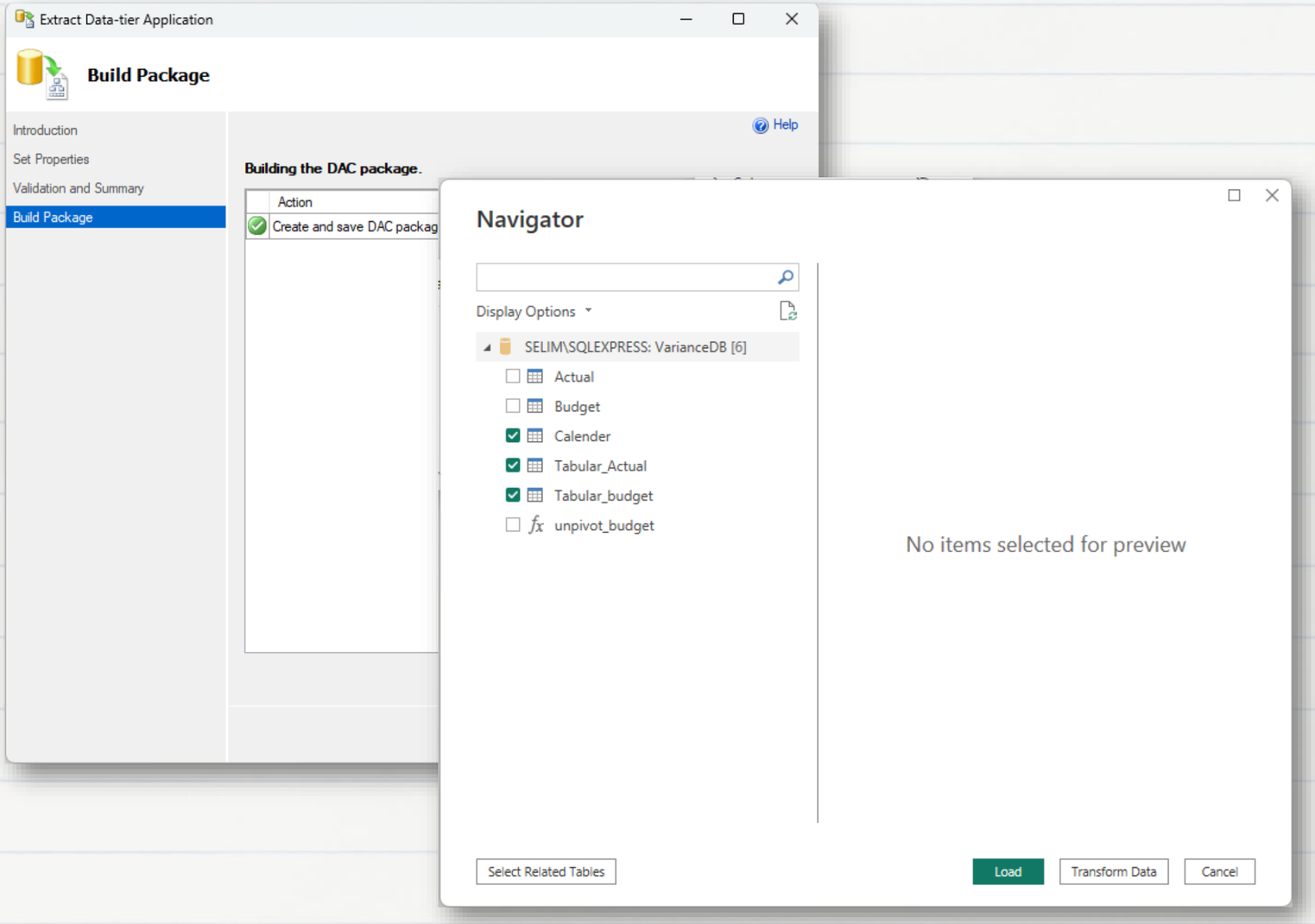
Data Preparation

Create a Calendar Table

```sql
-- #2.4 Create a Calendar Table
DROP TABLE IF EXISTS Calender;

CREATE TABLE Calender (
    Date DATE
);

INSERT INTO Calender (Date)
SELECT EOMONTH(date) AS Date FROM actual
GROUP BY EOMONTH(date)
ORDER BY EOMONTH(date);

SELECT * FROM Calender
```

	Date
1	2019-09-30
2	2019-06-30
3	2020-12-31
4	2019-10-31
5	2020-09-30
6	2020-01-31
7	2020-03-31
8	2019-12-31
9	2019-03-31
10	2020-11-30
11	2019-02-28
12	2020-05-31
13	2020-02-29
14	2019-01-31
15	2020-04-30
16	2019-11-30
17	2019-08-31
18	2019-05-31
19	2020-08-31
20	2020-10-31
21	2020-07-31
22	2019-04-30
23	2020-06-30
24	2019-07-31

Extract & Import data into Power BI



DAX Measures

```
Actual Aspen Sales = CALCULATE(
    SUM(Tabular_Actual[Sales]),
    Tabular_Actual[Product] = "Aspen"
)
```

```
Budget Aspen = CALCULATE(
    SUM(Tabular_Budget[Value]),
    Tabular_Budget[Product] = "Aspen"
)
```

```
Actual Carlota Sales = CALCULATE(
    SUM(Tabular_Actual[Total_Sales]),
    Tabular_Actual[Product] = "Carlota"
)
```

```
Budget Carlota = CALCULATE(
    SUM(Tabular_Budget[Value]),
    Tabular_Budget[Product] = "Carlota"
)
```

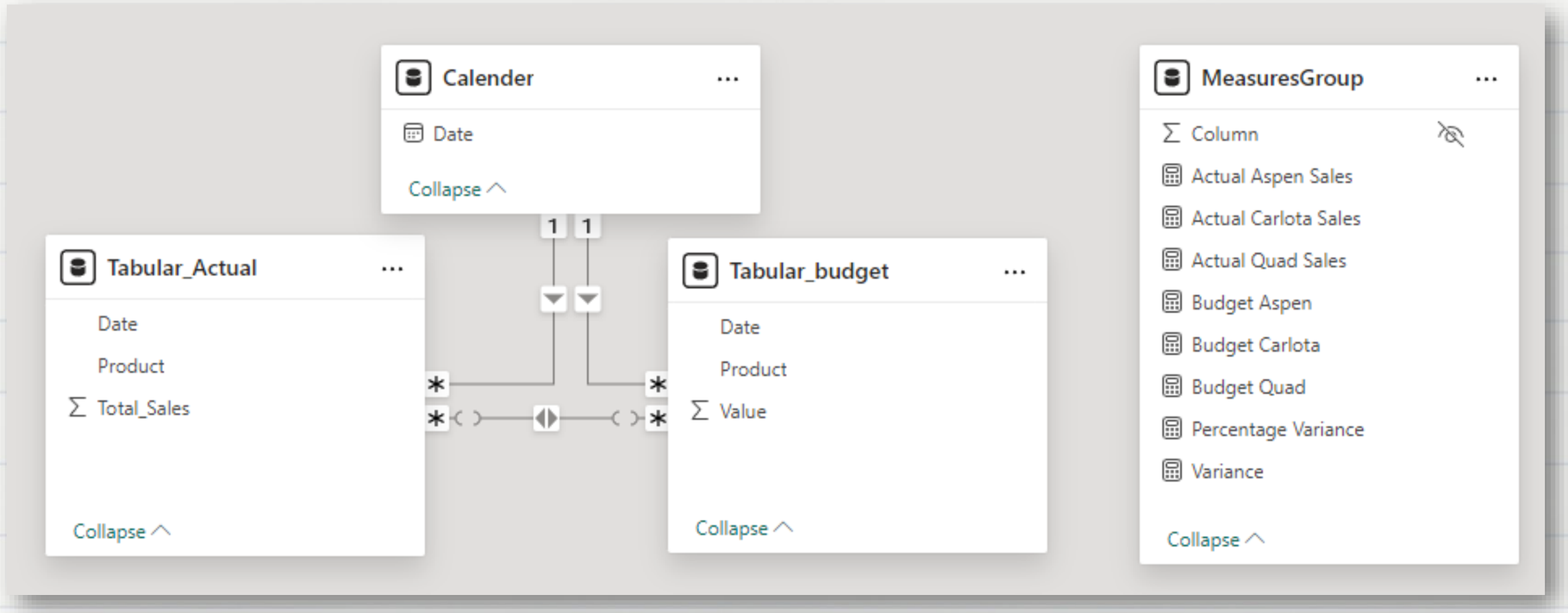
```
Actual Quad Sales = CALCULATE(
    SUM(Tabular_Actual[Total_Sales]),
    Tabular_Actual[Product] = "Quad"
)
```

```
Budget Quad = CALCULATE(
    SUM(Tabular_Budget[Value]),
    Tabular_Budget[Product] = "Quad"
)
```

```
Variance = SUM(Tabular_Actual[Total_Sales])
           - SUM(Tabular_budget[Value])
```

```
Percentage Variance = ((SUM(Tabular_Actual[Total_Sales])
                      - SUM(Tabular_budget[Value]))
    / SUM(Tabular_Actual[Total_Sales]) ) *100
```

Model View



Dashboard

Now Let’s Play with SQL <3

Total By Month

```sql
-- Total Budget By Month
SELECT eomonth,
       ROUND(SUM(value), 2) AS Total_budget
FROM unpivot_budget()
GROUP BY eomonth
ORDER BY eomonth;

-- Total Actual Sales By Month
SELECT EOMONTH(date) AS Date,
       ROUND(SUM(sales), 2) AS Total_Sales
FROM actual
GROUP BY EOMONTH(date)
ORDER BY EOMONTH(date);
```

	eomonth	Total_budget
1	2019-01-31	603812.96
2	2019-02-28	652192.64
3	2019-03-31	594022.82
4	2019-04-30	694618.68
5	2019-05-31	794389.67
6	2019-06-30	700644.98
7	2019-07-31	646331.91
8	2019-08-31	741765.44
9	2019-09-30	625676.30
10	2019-10-31	3399223.05
11	2019-11-30	7373740.03
12	2019-12-31	6038210.72
13	2020-01-31	730872.73
14	2020-02-29	567505.92
15	2020-03-31	755260.23
16	2020-04-30	613722.27
17	2020-05-31	587600.32
18	2020-06-30	811294.64
19	2020-07-31	671128.06
20	2020-08-31	707092.33
21	2020-09-30	805281.18
22	2020-10-31	2475593.22
23	2020-11-30	6552380.85
24	2020-12-31	7767075.02

	Date	Total_Sales
1	2019-01-31	615173.96
2	2019-02-28	590465.64
3	2019-03-31	645006.82
4	2019-04-30	662755.68
5	2019-05-31	714067.67
6	2019-06-30	627411.98
7	2019-07-31	675708.91
8	2019-08-31	670482.44
9	2019-09-30	630832.3
10	2019-10-31	3409807.05
11	2019-11-30	7560464.03
12	2019-12-31	6193694.72
13	2020-01-31	653756.73
14	2020-02-29	615472.92
15	2020-03-31	600761.23
16	2020-04-30	656604.27
17	2020-05-31	620596.32
18	2020-06-30	640892.64
19	2020-07-31	677471.06
20	2020-08-31	628075.33
21	2020-09-30	659813.18
22	2020-10-31	2455107.22
23	2020-11-30	6713277.85
24	2020-12-31	7798512.02

Total By Month

```sql
SELECT COALESCE(b.eomonth, a.Date) AS EOMonth,
        ROUND(SUM(b.TotalValue),2) AS Total_Value,
        ROUND(SUM(a.TotalSales),2) AS Total_Sales
FROM
    (SELECT EOMONTH(date) AS Date, SUM(sales) AS TotalSales FROM actual
     GROUP BY EOMONTH(date)) a
FULL OUTER JOIN
    (SELECT eomonth, SUM(value) AS TotalValue FROM unpivot_budget()
     GROUP BY eomonth) b
ON a.Date = b.eomonth
GROUP BY COALESCE(b.eomonth, a.Date)
ORDER BY EOMonth;
```

	EOMonth	Total_Value	Total_Sales
1	2019-01-31	603812.96	615173.96
2	2019-02-28	652192.64	590465.64
3	2019-03-31	594022.82	645006.82
4	2019-04-30	694618.68	662755.68
5	2019-05-31	794389.67	714067.67
6	2019-06-30	700644.98	627411.98
7	2019-07-31	646331.91	675708.91
8	2019-08-31	741765.44	670482.44
9	2019-09-30	625676.30	630832.3
10	2019-10-31	3399223.05	3409807.05
11	2019-11-30	7373740.03	7560464.03
12	2019-12-31	6038210.72	6193694.72
13	2020-01-31	730872.73	653756.73
14	2020-02-29	567505.92	615472.92
15	2020-03-31	755260.23	600761.23
16	2020-04-30	613722.27	656604.27
17	2020-05-31	587600.32	620596.32
18	2020-06-30	811294.64	640892.64
19	2020-07-31	671128.06	677471.06
20	2020-08-31	707092.33	628075.33
21	2020-09-30	805281.18	659813.18
22	2020-10-31	2475593.22	2455107.22
23	2020-11-30	6552380.85	6713277.85
24	2020-12-31	7767075.02	7798512.02

Variance By Month

```sql
SELECT COALESCE(b.eomonth, a.Date) AS EOMonth,
        ROUND((SUM(a.TotalSales) - SUM(b.TotalValue)),2) AS TotalSales
FROM
    (SELECT EOMONTH(date) AS Date, SUM(sales) AS TotalSales FROM actual
    GROUP BY EOMONTH(date)) a
FULL OUTER JOIN
    (SELECT eomonth, SUM(value) AS TotalValue FROM unpivot_budget()
     GROUP BY eomonth) b
ON a.Date = b.eomonth
GROUP BY COALESCE(b.eomonth, a.Date)
ORDER BY EOMonth;
```

	EOMonth	TotalSales
1	2019-01-31	11361
2	2019-02-28	-61727
3	2019-03-31	50984
4	2019-04-30	-31863
5	2019-05-31	-80322
6	2019-06-30	-73233
7	2019-07-31	29377
8	2019-08-31	-71283
9	2019-09-30	5156
10	2019-10-31	10584
11	2019-11-30	186724
12	2019-12-31	155484
13	2020-01-31	-77116
14	2020-02-29	47967
15	2020-03-31	-154499
16	2020-04-30	42882
17	2020-05-31	32996
18	2020-06-30	-170402
19	2020-07-31	6343
20	2020-08-31	-79017
21	2020-09-30	-145468
22	2020-10-31	-20486
23	2020-11-30	160897
24	2020-12-31	31437

Variance% By Month

```sql
SELECT COALESCE(b.eomonth, a.Date) AS EOMonth,
       CONCAT(ROUND((SUM(a.TotalSales) - SUM(b.TotalValue))
      /SUM(b.TotalValue) *100, 2), '%') AS TotalSales
FROM
    (SELECT EOMONTH(date) AS Date, SUM(sales) AS TotalSales FROM actual
    GROUP BY EOMONTH(date)) a
FULL OUTER JOIN
    (SELECT eomonth, SUM(value) AS TotalValue FROM unpivot_budget()
     GROUP BY eomonth) b
ON a.Date = b.eomonth
GROUP BY COALESCE(b.eomonth, a.Date)
ORDER BY EOMonth;
```

	EOMonth	TotalSales
1	2019-01-31	1.88%
2	2019-02-28	-9.46%
3	2019-03-31	8.58%
4	2019-04-30	-4.59%
5	2019-05-31	-10.11%
6	2019-06-30	-10.45%
7	2019-07-31	4.55%
8	2019-08-31	-9.61%
9	2019-09-30	0.82%
10	2019-10-31	0.31%
11	2019-11-30	2.53%
12	2019-12-31	2.58%
13	2020-01-31	-10.55%
14	2020-02-29	8.45%
15	2020-03-31	-20.46%
16	2020-04-30	6.99%
17	2020-05-31	5.62%
18	2020-06-30	-21%
19	2020-07-31	0.95%
20	2020-08-31	-11.17%
21	2020-09-30	-18.06%
22	2020-10-31	-0.83%
23	2020-11-30	2.46%
24	2020-12-31	0.4%

End
Of
Text <3

میم سین