Глава 27. Новые функции и переменные DAX

Это продолжение перевода книги Роб Колли. Формулы DAX для Power Pivot. Главы не являются независимыми, поэтому рекомендую начать сначала.

Предыдущая глава Содержание Следующая глава

Всё, что описано в этой главе будет работать, начиная с Excel 2016, или в Power BI Desktop.

```
1 ProductLifeSpan (Months) = DATEDIFF('Product'[StartDate]; 'Product'[EndDate]; MONTH)
```

ProductDescription	StartDate	EndDate	ProductLifeSpan (Months)
All-purpose bar for on or off-road.	01.07.2002	30.06.2003	11
All-purpose bar for on or off-road.	01.07.2003		
Tough aluminum alloy bars for downhill.	01.07.2002	30.06.2003	11
Tough aluminum alloy bars for downhill.	01.07.2003		
Flat bar strong enough for the pro circuit.	01.07.2002	30.06.2003	11
Flat bar strong enough for the pro circuit.	01.07.2003		
Unique shape provides easier reach to the levers.	01.07.2002	30.06.2003	11
Unique shape provides easier reach to the levers.	01.07.2003		

Рис. 27.1. Вычисляемый столбце на основе функция DATEDIFF() в Power BI Desktop

Функция DATEDIFF()

Возвращает количество единиц между двумя датами. Синтаксис:

DATEDIFF(<начальная дата>; <конечная дата>; <интервал>)

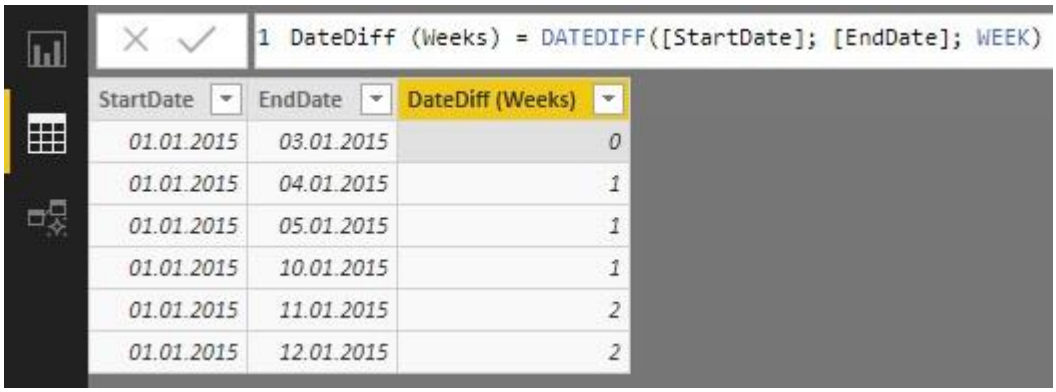
<интервал> может принимать значения: Second, Minute, Hour, Day, Week, Month, Quarter, Year

Например, мы можем создать вычисляемый столбец для определения срока службы продукта в месяцах:

ProductLifeSpan (Months) = DATEDIFF('Product'[StartDate]; 'Product'[EndDate]; MONTH)

Иногда мы не понимаем, каким должен быть правильный ответ, учитывая конкретные даты. Нам нравится, как документация объясняет возвращаемое значение: DATEDIFF возвращает количество границ интервалов, находящихся между двумя датами. Поэтому, если вы посмотрите на январь 2015 и границы недели (по умолчанию первый день недели – воскресенье)…

… вы поймете результаты DATEDIFF, показанные ниже:



```
1 DateDiff (Weeks) = DATEDIFF([StartDate]; [EndDate]; WEEK)
```

StartDate	EndDate	DateDiff (Weeks)
01.01.2015	03.01.2015	0
01.01.2015	04.01.2015	1
01.01.2015	05.01.2015	1
01.01.2015	10.01.2015	1
01.01.2015	11.01.2015	2
01.01.2015	12.01.2015	2

Рис. 27.3. Результат, возвращаемый DATEDIFF для интервалов WEEK

Функции MEDIAN() и PERCENTILE

Синтаксис – MEDIAN(<столбец>). Например, Median Sales = MEDIAN(Sales[SalesAmount])

Date	Total Sales	Average Sales	Median Sales
01.01.2003	$12 445	$1 556	$1 536
02.01.2003	$19 703	$1 642	$2 049
03.01.2003	$13 520	$1 931	$2 071
04.01.2003	$18 629	$2 329	$2 443
05.01.2003	$13 497	$1 928	$2 071
06.01.2003	$4 363	$2 182	$2 182
07.01.2003	$14 623	$1 625	$2 049
08.01.2003	$15 733	$1 573	$2 049
09.01.2003	$18 142	$2 016	$2 071
10.01.2003	$15 558	$1 414	$1 000
11.01.2003	$13 977	$1 997	$2 071

Рис. 27.4. Обратите внимание, что среднее и медиана отличаются

Процентиль представлен парой функций PERCENTILE.INC(<столбец>, <k>) PERCENTILE.EXC(<столбец>, <k>). Что означают суффиксы функций, и почему их две, см., например, КВАРТИЛЬ: какие формулы расчета использует Excel.

Функция PRODUCT()

Возможность умножения значений в столбце ранее отсутствовала. Расчет кумулятивной отдачи, вероятно, является наиболее распространенным примером того, когда нам это нужно. Предположим, у нас есть ежемесячная доходность S&P500:

Year	Month	Return	Factor
2012	Jan	4,36%	1,0436
2012	Feb	4,06%	1,0406
2012	Mar	3,13%	1,0313
2012	Apr	-0,75%	0,9925
2012	May	-6,27%	0,9373
2012	Jun	3,96%	1,0396
2012	Jul	1,26%	1,0126
2012	Aug	1,98%	1,0198
2012	Sep	2,42%	1,0242
2012	Oct	-1,98%	0,9802
2012	Nov	0,28%	1,0028
2012	Dec	0,71%	1,0071

Рис. 27.5. Ежемесячная доходность S&P500

Если месячная доходность за первый месяц равна R1, за второй месяц R2 и так далее, годовая доходность будет рассчитываться по формуле:

(1+R1) * (1+R2) *... * (1+R12) - 1

У нас уже есть значение 1+R, хранящееся в столбце Factor. Таким образом, мы можем определить нашу меру как:

Annual Return = PRODUCT(MonthlyReturn[Factor]) – 1

Year	Annual Return
2012	13,40%
2013	29,61%
2014	11,38%
Всего	**63,71%**

Рис. 27.6. Годовая доходность, рассчитанная на основе месячной доходности

Функции GEOMEAN() и GEOMEANX()

Существует несколько показателей центральной тенденции, таких как медиана, среднее арифметическое и среднее геометрическое. В некоторых сценариях среднее геометрическое более эффективно, чем среднее арифметическое. Среднее геометрическое является

предпочтительным подходом (по сравнению со средним арифметическим), когда значения являются процентами (например, норма прибыли) или в разных масштабах (например, рейтинги фильмов и кассовые сборы фильмов).

В примере ниже мы хотим рассчитать среднее арифметическое и среднее геометрическое годовой прибыли за несколько лет. Для этого мы будем использовать X-версию функций, которая позволяет нам перебирать годы и вычислять среднюю отдачу.

```
Annual Return Arithmetic Mean =
        AVERAGEX(VALUES(MonthlyReturn[Year]); [Annual Return])
Annual Return Geometric Mean =
        GEOMEANX(VALUES(MonthlyReturn[Year]); [Annual Return])
```

Year	Annual Return Arithmetic Mean	Annual Return Geometric Mean
2012	13,40%	13,40%
2013	29,61%	29,61%
2014	11,38%	11,38%
Всего	**18,13%**	**16,53%**

Рис. 27.7. Среднее геометрическое и среднее арифметическое различаются

Подробнее о геометрическом среднем см. http://ppvt.pro/geomeanMath

Заметим, что целый ряд статистических функций имеют X-версии. Например, MEDIANX, PERCENTILEX.INC, PERCENTILEX.EXC, PRODUCTX. Как и SUMX, все эти X-функции дают возможность работать с мерами (в отличие от обычных функций, которые обрабатывают значения в столбцах). X-функции позволяют управлять <таблицами>, над которой выполняется расчет, в том числе и виртуальными таблицами, созданными табличными функциями, такими как ALL и FILTER (см. главу 24. Нюансы функций CALCULATE и FILTER). X версии функций сложнее для понимания, но они не менее, если не более важны (основные X-функции DAX см. Глава 16. SUMX() и другие X функции (итераторы)).

CONCATENATEX

Эта функция полезна, когда нужно объединить в одной ячейке переменное количество текстовых значений. Чтобы проиллюстрировать идею, сначала мы определим меру Top N, которая считает продажи для лучших N продуктов (для определенности – трех).

```
Top 3 Products Sales =
CALCULATE(
        [Total Sales];
        TOPN(3; Product; [Total Sales])
)
```

Теперь определим меру, которая позволит записать три лучших продукта через запятую (рис. 27.8):

```
Top 3 Products Names =
CONCATENATEX(
        TOPN(3; Product; [Total Sales])
                ;Product[ProductName]
                ; ", "
                ;[Total Sales]
                ;DESC
)
```

CalendarYear	Month	Total Sales	Top 3 Products Sales	Top 3 Products Names
2001	Jul	$473 388	$261 214	Road-150 Red, 48, Road-150 Red, 44, Road-150 Red, 62
	Aug	$506 192	$271 949	Road-150 Red, 62, Road-150 Red, 56, Road-150 Red, 52
	Sep	$473 943	$279 105	Road-150 Red, 48, Road-150 Red, 52, Road-150 Red, 62
	Oct	$513 329	$264 792	Road-150 Red, 62, Road-150 Red, 56, Road-150 Red, 48
	Nov	$543 993	$289 840	Road-150 Red, 52, Road-150 Red, 56, Road-150 Red, 62
	Dec	$755 528	$400 766	Road-150 Red, 48, Road-150 Red, 62, Road-150 Red, 44

Рис. 27.8. CONCATENATEX выводит имена трех самых продаваемых продуктов

И, действительно, если отсортировать данные за июль 2001, то увидим:

CalendarYear	Month	ProductKey	ProductName	Total Sales
2001	Jul	312	Road-150 Red, 48	$100 192
2001	Jul	311	Road-150 Red, 44	$82 300
2001	Jul	310	Road-150 Red, 62	$78 722
2001	Jul	314	Road-150 Red, 56	$53 674
2001	Jul	313	Road-150 Red, 52	$42 939
2001	Jul	346	Mountain-100 Silver, 44	$30 600
2001	Jul	350	Mountain-100 Black, 44	$20 250
2001	Jul	344	Mountain-100 Silver, 38	$13 600
2001	Jul	347	Mountain-100 Silver, 48	$10 200
2001	Jul	348	Mountain-100 Black, 38	$10 125

Рис. 27.9. Топ 3 продуктов за июль 2001

Синтаксис:

CONCATENATEX(<таблица>; <выражение>; [разделитель]; [сортировать по]; [порядок сортировки])

Разберем работу меры Top 3 Products Sales подробнее. Как и другие X-функции, первый аргумент – <таблица>, над которой будут выполняться итерации. В нашем примере таблица – первые три строки с максимальными продажами за выбранный период (строка 3 формулы выше).

Второй аргумент (строка 4) – выражение, которое рассчитывается для каждой строки таблицы. В нашем случае – значения из столбца [ProductName]. Именно они будут сцепляться.

Разделитель (строка 5) – запятая.

Строка 5 и 6 – необязательные аргументы. Их использование позволяет вывести результаты в определенном порядке. Мы хотим отсортировать названия продуктов по объему продаж – [Total Sales] (строка 5) в порядке убывания – DESC (строка 7; подробнее см. описание функции RANKX() в Глава 16. SUMX() и другие X функции).

ISEMPTY()

Возвращает ИСТИНА, если указанная таблица (или табличное выражение) пуста. Синтаксис: ISEMPTY(<таблица или табличное выражение>). Определим меру ProductNotSold = ISEMPTY(Sales).

ProductKey	ProductName	ProductNotSold	Total Sales
210	HL Road Frame - Black, 58	True	
211	HL Road Frame - Red, 58	True	
212	Sport-100 Helmet, Red	True	
213	Sport-100 Helmet, Red	True	
214	Sport-100 Helmet, Red	False	$78 028
215	Sport-100 Helmet, Black	True	
216	Sport-100 Helmet, Black	True	
217	Sport-100 Helmet, Black	False	$72 954
218	Mountain Bike Socks, M	True	

Рис. 27.10. ISEMPTY() в работе

INTERSECT(), EXCEPT() and UNION()

Эти функции пригодятся, когда вы сравниваете списки или комбинируете их каким-то образом. Синтаксис: INTERSECT(<LeftTable >; <RightTable>), EXCEPT(<LeftTable >; <RightTable>), UNION(<table_expression1>; <table_expression2> [; <table_expression>]...)

INTERSECT: возвращает строки левой таблицы, которые присутствуют в правой таблице. EXCEPT возвращает строки левой таблицы, которых нет в правой таблице. UNION возвращает объединение двух таблиц, если их столбцы совпадают.

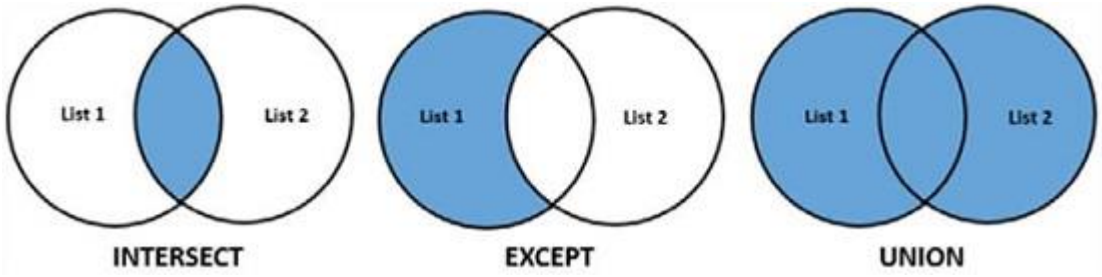


Рис. 27.11. Графическая иллюстрация работы функций

Допустим, у нас есть:

- Список 1: 10 лучших продуктов по объему продаж
- Список 2: 10 лучших продуктов по размеру прибыли

ProductName	Total Sales
Women's Mountain Shorts, L	$25 406
Women's Mountain Shorts, M	$24 636
Long-Sleeve Logo Jersey, L	$22 595
Long-Sleeve Logo Jersey, M	$22 096
Short-Sleeve Classic Jersey, XL	$22 082
Short-Sleeve Classic Jersey, M	$21 974
Short-Sleeve Classic Jersey, S	$21 920
Long-Sleeve Logo Jersey, S	$21 446
Women's Mountain Shorts, S	$21 277
Long-Sleeve Logo Jersey, XL	$20 646

ProductName	Margin
Women's Mountain Shorts, L	$15 904
Women's Mountain Shorts, M	$15 422
Women's Mountain Shorts, S	$13 319
Classic Vest, M	$7 910
Classic Vest, L	$7 751
Half-Finger Gloves, M	$7 650
Half-Finger Gloves, S	$7 481
Half-Finger Gloves, L	$6 792
Classic Vest, S	$6 678
Long-Sleeve Logo Jersey, L	$5 197

Рис. 27.12. Топ-10 продуктов по продажам и прибыли в категории «одежда»

INTERSECT позволит подсчитать число продуктов, присутствующие в обоих списках:

```
Count Products with TopSales AND TopMargin =
COUNTROWS(
        INTERSECT(
                TOPN(10; 'Product'; [Total Sales]);
                TOPN (10; 'Product'; [Margin])
        )
)
```

И таких продуктов 4. Используя EXCEPT, мы хотим определить количество продуктов, которые есть в списке 1, но нет в списке 2:

```
Count Products with TopSales and NOT TopMargin =
COUNTROWS(
        EXCEPT(
                TOPN(10; 'Product'; [Total Sales]);
                TOPN(10; 'Product'; [Margin])
        )
)
```

И таких продуктов 6. Используя UNION, мы хотим определить количество продуктов, которые отображаются в списке 1 или в списке 2. Это немного сложнее, так как UNION() сохраняет все строки обеих таблиц (в том числе и дубли):

```
Count Product with TopSales OR TopMargin =
VAR TopSalesOrMargin =
```

```
        UNION (
                TOPN (10; 'Product', [Total Sales]);
                TOPN (10; 'Product', [Margin])
        )
RETURN
        COUNTROWS(SUMMARIZE(TopSalesOrMargin, 'Product'[ProductKey])
```

VAR используется для объявления переменной (мы поговорим о них ниже). Наша мера осуществляет вычисления в два шага. 1). Функция UNION объединяет списки. 2). Функция SUMMARIZE возвращает временную таблицу в виде сводной с Product[ProductKey] в строках и пустотой в столбцах. А сводная, естественно, удаляет дубли. COUNTROWS подсчитывает строки в этой виртуальной сводной таблице. Результат – 16.

Переменные DAX

Переменные подобны магнитофону. Можно «записать» результат вычисления, а затем «воспроизвести» его несколько раз в других частях формулы.

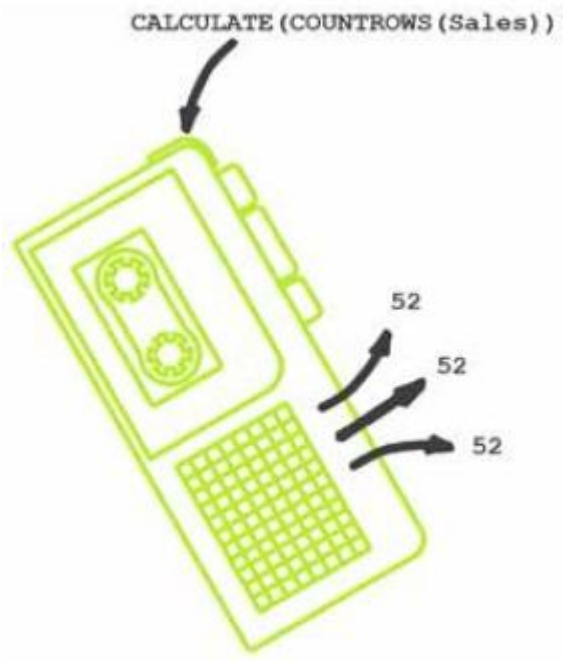


Рис. 27.13. Функции DAX позволяет записать значение в одной части формулы, и анализировать его в нескольких других местах

Переменные дают три преимущества:

- Они делают формулы более прозрачными, облегчают написание, а также дальнейшее редактирование, удаляя повторяющиеся фрагменты из формул.
- Они позволяют меньше беспокоиться о проблеме: контекст строк против контекста фильтра.
- Они ускоряют работу формул, не требуя, чтобы механизм DAX несколько раз оценивал одно и то же выражение.

Представим на минуту следующую формулу:

```
Transaction Count =
IF (
        CALCULATE ( COUNTROWS ( Sales ) ) > 0;
        CALCULATE ( COUNTROWS ( Sales ) );
        -1
)
```

Формула выполняет расчет, и если он больше нуля, то возвращаем тот же самый расчет. В противном случае возвращает "-1". К сожалению, это требует два раза вычислить выражение CALCULATE(COUNTROWS(...)). Это традиционный шаблон, который присутствует и в обычном Excel:

ЕСЛИ(ЕОШИБКА(<выражение>); 0; <выражение>). В Excel этот шаблон был упрощен, начиная с Excel 2016 с помощью функции ЕСЛИОШИБКА(<выражение>; значение если ошибка), которая позволяет написать <выражение> только один раз. Аналогичная функция появилась и в DAX, также начиная с версии 2016 г. – IFERROR.

Но в нашем случае мы не проверяем, возвращает ли выражение ошибку, т.е., не можем использовать IFERROR, чтобы не дублировать вычисление. Можно определить переменную:

```
Transaction Count =
VAR RowCount =
        CALCULATE(COUNTROWS(Sales))
RETURN
        IF(RowCount > 0; RowCount; -1)
```

В нашем примере с введением переменной запись не упростилась, а усложнилась. Но ведь это только пример, который иллюстрирует концепцию. В реальной жизни ваши выражения могут быть очень длинными, так что экономия будет очевидной.

Ключевое слово VAR

Если вы набирали формулу с клавиатуры, то заметите, что ключевое слово VAR не отображается в автозаполнении:

Рис. 27.14. VAR не появляются в списке автозаполнения, что немного странно

Не волнуйтесь, VAR существует! Просто введите имя меры знак := и VAR, нажмите пробел и вуаля:

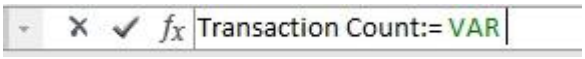


Рис. 27.15. Обратите внимание, что VAR выделяется цветом, указывая, что DAX знает, что это слово особенное

Как только механизм DAX увидит ключевое слово VAR, он ожидает, что вы далее введете блок <имя переменной> = <выражение>. Продолжайте вводить, как в примере выше:

```
VAR RowCount = CALCULATE(COUNTROWS(Sales))
```

…и посмотрите, что редактор формул сделает с этим:

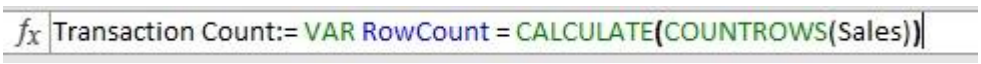


Рис. 27.16. RowCount также выделен цветом, указывая, что DAX распознал слово, как переменную

Вы можете создать несколько переменных в одной формуле, при этом каждому блоку <имя переменной> = <выражение> должно предшествовать собственное ключевое слово VAR.

Ключевое слово RETURN

Теперь, чтобы продолжить написание обычной формулы, нужно закрыть блок определения переменной ключевым словом RETURN.

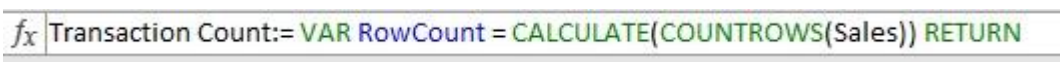


Рис. 27.17. Ключевое слово RETURN также не отображается в автозаполнении, но также, как и VAR, оно распознается и форматируется цветом

Ссылка на переменную

Для использование переменной просто введите ее имя.

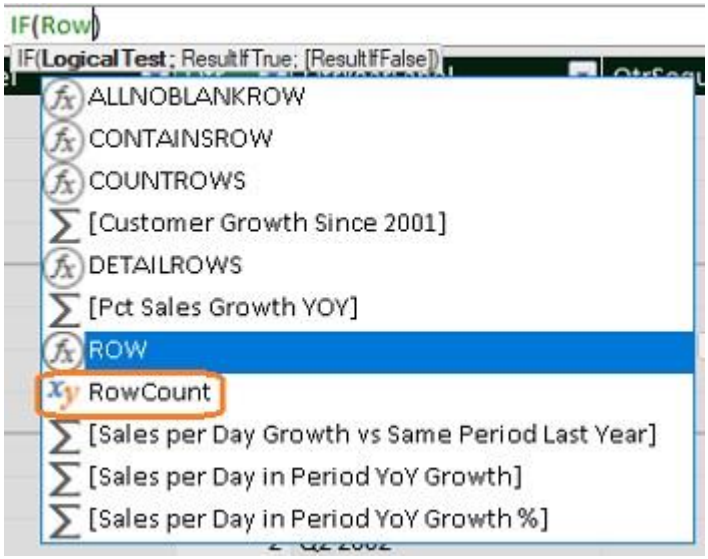


Рис. 27.18. Имена переменных появляются в автозаполнении и даже имеют оригинальный значок

После определения переменной внутри конструкции VAR … RETURN при каждом новом использовании она будет выделяться цветом, сигнализируя, что движок DAX ее распознал.

Переменная вместо функции EARLIER

Помните формулу из предыдущей главы для подсчета продаж по категории?

```
CategorySalesWithEarlier =
CALCULATE(
        [Total Sales];
        FILTER(
                'Product';
                'Product'[ProductCategory] =
                        EARLIER ('Product'[ProductCategory])
        )
)
```

Оказывается, что применение переменной позволяет преодолеть трудности с контекстом строки:

```
CategorySalesWithVariable =
VAR Category = 'Product'[ProductCategory]
RETURN
        CALCULATE (
                [Total Sales],
                FILTER ( 'Product',
                        'Product'[ProductCategory] = Category )
        )
```

ProductName	ProductCategory	CategorySalesWithEarlier	CategorySalesWithVariable
All-Purpose Bike Stand	Accessories	$700 759,96	$700 759,96
AWC Logo Cap	Clothing	$1 019 317,83	$1 019 317,83
Bike Wash - Dissolver	Accessories	$700 759,96	$700 759,96
Cable Lock	Accessories	$700 759,96	$700 759,96
Classic Vest, L	Clothing	$339 772,61	$339 772,61
Classic Vest, M	Clothing	$339 772,61	$339 772,61
Classic Vest, S	Clothing	$339 772,61	$339 772,61
Fender Set - Mountain	Accessories	$700 759,96	$700 759,96

Рис. 29.19. Переменная VAR и функция EARLIER дают одинаковый результат

Напомним, что функция EARLIER требовалась нам, что в процессе работы функции-итератора FILTER сравнить контекст строки внутри FILTER и вне ее. Поскольку блок VAR оценивается один раз, причем вне функции FILTER, для получения исходного значения контекста строки достаточно

просто сослаться на столбец. Значение переменной Category является статическим для остальной части формулы.

Это не означает, что переменные могут заменять EARLIER в любых формулах, потому что в некоторых сложных ситуациях вы можете жонглировать более чем двумя контекстами строк (функция EARLIER имеет второй, необязательный аргумент, задающий, на сколько уровней вверх выйти с текущего контекста строки; по умолчанию этот аргумент равен единице). В этом случае никакого одного значения не будет достаточно. Но в большинстве ситуацией EARLIER можно заменить переменными, если вам так больше нравится.

Переменная вместо ссылки на меру внутри функции FILTER

В главе, посвященной контексту строк и контексту фильтра, мы столкнулись с тем, что ссылка на меру внутри функции FILTER работала с ошибками, поэтому вместо меры мы рекомендовали использовать формулу, которая и задала меру.

Напомним. Исходная (работающая) формула:

```
Transaction for Highest Price = CALCULATE(
        COUNTROWS(Sales);
        FILTER(
                Products;
                Products[ListPrice] = MAX(Products[ListPrice])
        )
)
```

Однако, если сначала определить меру…

Highest Price = MAX(Products[ListPrice])

… то следующая формула переставала работать:

```
Transaction for Highest Price BROKEN = CALCULATE(
        COUNTROWS(Sales);
        FILTER(
                Products;
                Products[ListPrice] = [Highest Price]
        )
)
```

Использование переменной возвращает работоспособность формуле:

```
Transactions for Highest Price FIXED =
VAR HighPrice = [Highest Price]
RETURN
        CALCULATE (
                COUNTROWS ( Sales ),
                FILTER ( Products, Products[ListPrice] = HighPrice )
        )
```

Использование переменной позволяет вам вернуться к лучшей практике – «я пишу каждую формулу только один раз». От этого правила вы были вынуждены отступить при работе внутри FILTER. Маленькая победа, но все же победа!