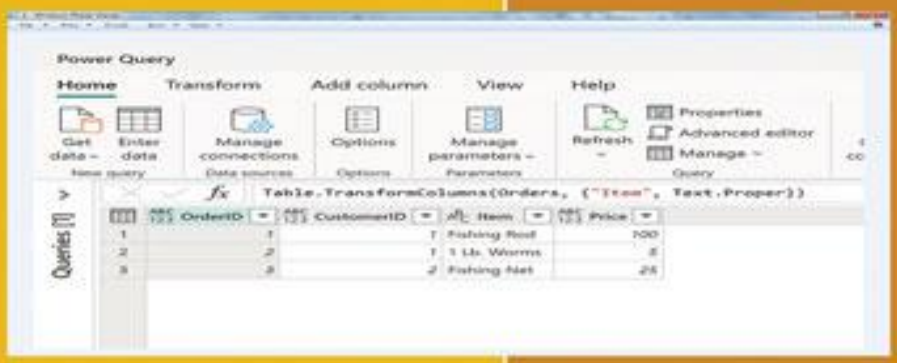
مؤسسه فرهنگی هنری
دیباگران تهران

مرجع زبان فرمول نویسی

M

در پاور کوئری



روزبه امیر عصامی

به نام خدا

مرجع زبان فرمول نویسی

M

در پاور کوئری

مؤلف:

روزبه امیرعصامی

مؤسسه فرهنگی هنری
دیباگران تهران

عنوان کتاب: مرجع زبان فرمول نویسی M در پاور کوئری

مولف : روزبه امیرعصامی

ناشر: موسسه فرهنگی هنری دیباگران تهران
ویراستار: مهدیه مخبری
صفحه آرایی: نازنین نصیری
طراح جلد:داریوش فرسایی
نوبت چاپ: اول
تاریخ نشر: ۱۴۰۲
چاپ و صحافی:صدف
تیراژ:۱۰۰ جلد

شابک: ۱-۷۲۳-۲۱۸-۶۲۲-۹۷۸

نشانی واحد فروش:تهران، خیابان انقلاب،خیابان دانشگاه
-تقاطع شهدای ژاندارمری-پلاک ۱۵۸ساختمان دانشگاه-
طبقه دوم-واحد۴ تلفن ها: ۶۶۹۶۵۷۴۹-۲۲۰۸۵۱۱۱

فروشگاههای اینترنتی دیباگران تهران :
WWW.MFTBOOK.IR
www.dibagarantehran.com

سرشناسه:امیر عصامی،روزبه،۱۳۶۱-
عنوان و نام پدیدآور:مرجع زبان فرمول نویسی M درپاور کوئری/مولف:روزبه امیر عصامی؛
ویراستار:مهدیه مخبری.
مشخصات نشر: تهران : دیباگران تهران :۱۴۰۲
مشخصات ظاهری:۲۰۶ ص:جدول،نمودار.
شابک: ۱-۷۲۳-۲۱۸-۶۲۲-۹۷۸
وضعیت فهرست نویسی: فیپا
موضوع:اکسل مایکروسافت(فایل کامپیوتر)
موضوع:Microsoft Excel(computer file)
موضوع:پایگاه های اطلاعاتی-مدیریت
موضوع:database-management
موضوع:پرس و جو
موضوع:Querying(computer science)
رده بندی کنگره:HF ۵۵۴۸/۴
رده بندی دیویی:۰۰۵/۵۴
شماره کتابشناسی ملی:۹۳۲۹۲۱۰

نشانی تلگرام:@mftbook نشانی اینستاگرام دیبا dibagaran_publishing

فهرست مطالب

فصل سوم

مقدمه ناشر

خط مشی انتشارات مؤسسه فرهنگی هنری دیباگران تهران در عرصه کتاب هایی با کیفیت عالی است که بتواند خواسته های به روز جامعه فرهنگی و علمی کشور را تا حد امکان پوشش دهد.

هر کتاب دیباگران تهران، یک فرصت جدید شغلی و علمی

حمد و سپاس ایزد منان را که با الطاف بیکران خود این توفیق را به ما ارزانی داشت تا بتوانیم در راه ارتقای دانش عمومی و فرهنگی این مرز و بوم در زمینه چاپ و نشر کتب علمی و آموزشی گام‌هایی هرچند کوچک برداشته و در انجام رسالتی که بر عهده داریم، مؤثر واقع شویم.

گستردگی علوم وسرعت توسعه روزافزون آن، شرایطی را به وجود آورده که هر روز شاهد تحولات اساسی چشمگیری در سطح جهان هستیم. این گسترش و توسعه، نیاز به منابع مختلف از جمله کتاب را به عنوان قدیمی‌ترین و راحت‌ترین راه دستیابی به اطلاعات و اطلاع‌رسانی، بیش از پیش برجسته نموده است.

در این راستا، واحد انتشارات مؤسسه فرهنگی هنری دیباگران تهران با همکاری اساتید، مؤلفان، مترجمان، متخصصان، پژوهشگران و محققان در زمینه های گوناگون و مورد نیاز جامعه تلاش نموده برای رفع کمبودها و نیازهای موجود، منابعی پُربار، معتبر و با کیفیت مناسب در اختیار علاقمندان قرار دهد.

کتابی که در دست دارید تألیف **"جناب آقای روزبه امیرعصامی"** است که با تلاش همکاران ما در نشر دیباگران تهران منتشرگشته و شایسته است از یکایک این گرامیان تشکر و قدردانی کنیم.

با نظرات خود مشوق و راهنمای ما باشید

با ارائه نظرات و پیشنهادات وخواسته های خود،به ما کمک کنید تا بهتر و دقیق تر در جهت رفع نیازهای علمی و آموزشی کشورمان قدم برداریم.برای رساندن پیام هایتان به ما از رسانه های دیباگران تهران شامل سایتهای فروشگاهی و صفحه اینستاگرام و شماره های تماس که در صفحه شناسنامه کتاب آمده استفاده نمایید.

مدیر انتشارات

مؤسسه فرهنگی هنری دیباگران تهران
dibagaran@mftplus.com

فصل اول

آشنایی با پاورکوئری و M Language

Power Query (پاور کوئری) چیست؟

پاور کوئری ابزاری است که به کاربران امکان می‌دهد قبل از اینکه داده‌ها را وارد اکسل و یا مدل‌های داده پاور پیوت کنند، آنها را فراخوانی کنند، استخراج نمایند و شکل‌دهی کنند.

پاور کوئری با استفاده از زبان M فعالیت می‌کند و در حال حاضر با نام Get And Transform شناخته می‌شود و در تب Data ریبون اکسل و در Power BI با عنوان Transform قرار دارد.

زبان فرمول‌نویسی M چیست؟

M نام غیررسمی زبان فرمول‌نویسی پاور کوئری است. به‌دلیل اینکه نام رسمی آن طولانی است کسی از آن استفاده نمی‌کند. بعضی از افراد اعتقاد دارند M مخفف "data mash up" است بعضی دیگر از افراد معتقدند M مخفف "data modeling" است، ولی نکته‌ای که اهمیت دارد این است که M یک زبان عملکردی است و ما می‌بایست عملکردهای آن را بدانیم. هر زبان دارای یک ساختار و ترکیب است که اولین سطح از یادگیری هر زبان را می‌بایست به این قسمت تخصیص داد. در این مطلب، به مرور ترکیب فرمول‌های M می‌پردازیم. قبل از شروع به یاد داشته باشید که:

M "بسیار قدرتمندتر از رابط کاربری گرافیکی است"

رابط گرافیکی کاربر هر ماه و شاید هفته دچار تغییر می‌شود و خاصیت‌ها و عملکردهای جدید به آن اضافه می‌شود، ولی حقیقت این است که تمام این خواص و عملکردها سال‌هاست که براساس یک زبان نوشته شده است.

ترکیب زبان فرمول‌نویسی M

ترکیب این زبان بسیار ساده است. این فرمول‌ها همیشه دارای دو قسمت فرمول‌نویسی هستند. قسمت عبارت let و قسمت عبارت in مثالی ساده از این دو قسمت، در پایین آورده شده است:

```
let
        x=1
in
        x
```

letو in کلمات از پیش تعریف شده هستند. قبل از ادامه مطلب شما می‌بایست درنظر داشته باشید که:

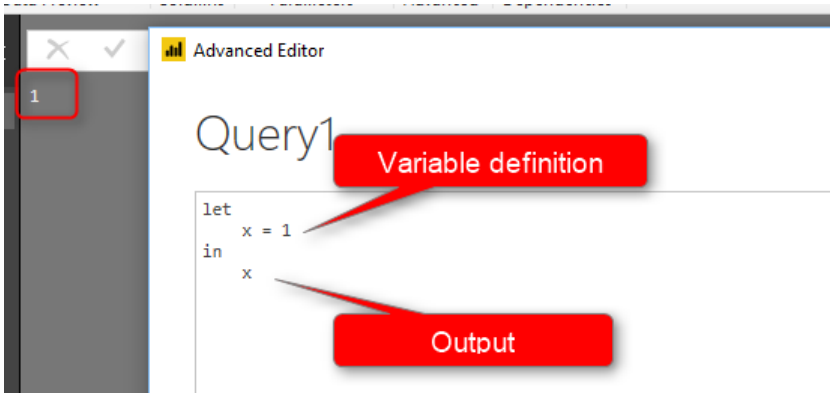
" زبان فرمول‌نویسی M نسبت به حروف بزرگ و کوچک حساس است و برای مثال بین X و x تفاوت وجود دارد."

مفهوم دو قسمت فرمول‌های زبان M چیست؟

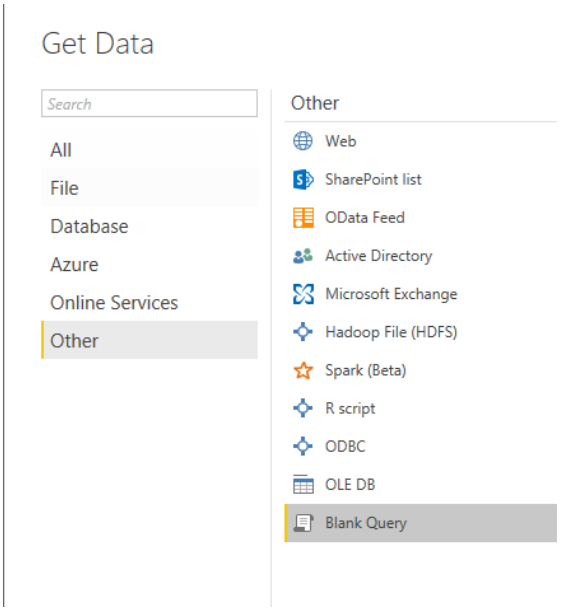
Let: تعریف تمام متغیرها

In: خروجی. هر چیزی که درون این قسمت قرار می‌دهید خروجی کوئری شما خواهد بود.

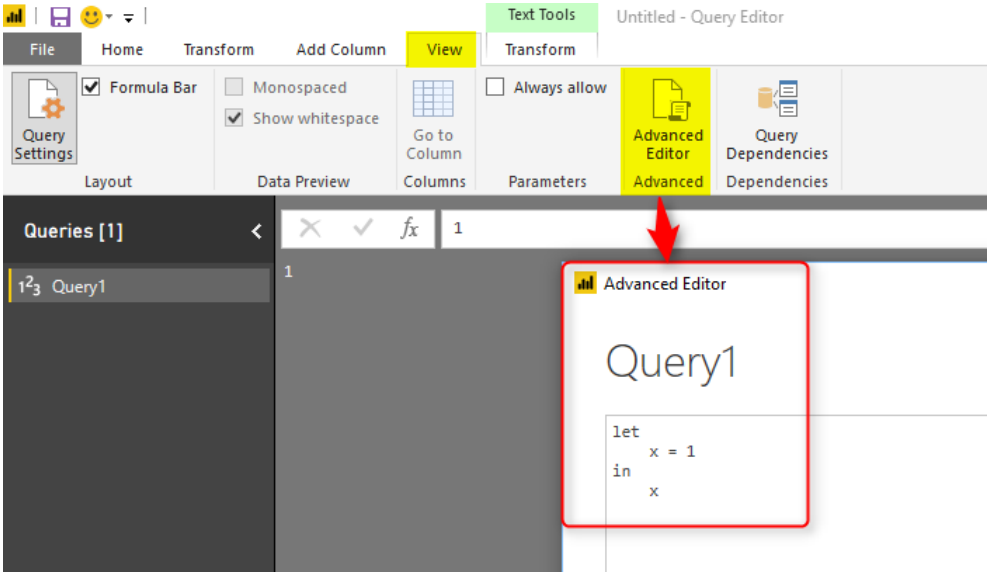
درنتیجه کوئری زیر به این معنی است که یک متغیر به نام X تعریف می‌شود، مقدار ۱ به آن تخصیص داده می‌شود و نشان‌دادن آن به‌عنوان نتیجه؛ بنابراین کوئری مقدار ۱ را برمی‌گرداند.



برای اجرای این مثال، می‌بایست power BI را باز کنید سپس به قسمت data بروید و یک کوئری خالی جدید ایجاد کنید.



سپس در تب view یا Home گزینه advanced editor را انتخاب کنید.

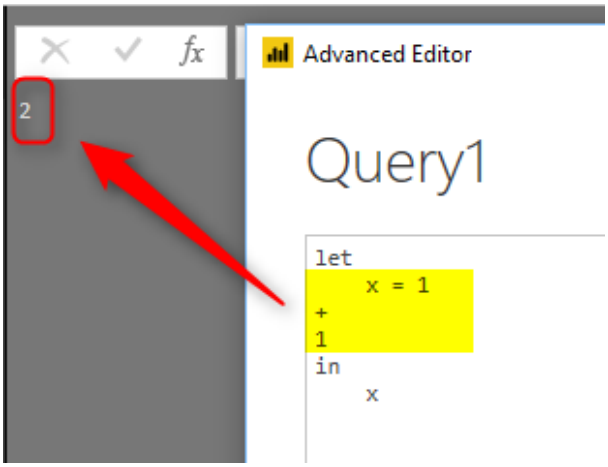


حتماً دقت داشته باشید که تمام کلمات تعریف شده مثل let و in را به‌صورت حروف کوچک بنویسید. همچنین متغیر شما نیز می‌بایست در هر دو قسمت فرمول از یک نوع حروف استفاده شده باشد و همانطور که مشاهده می‌کنید نیازی به تعریف نوع متغیرها نیست. پس از اولین تخصیص مقدار نوع متغیر به‌صورت اتوماتیک مشخص می‌شود.

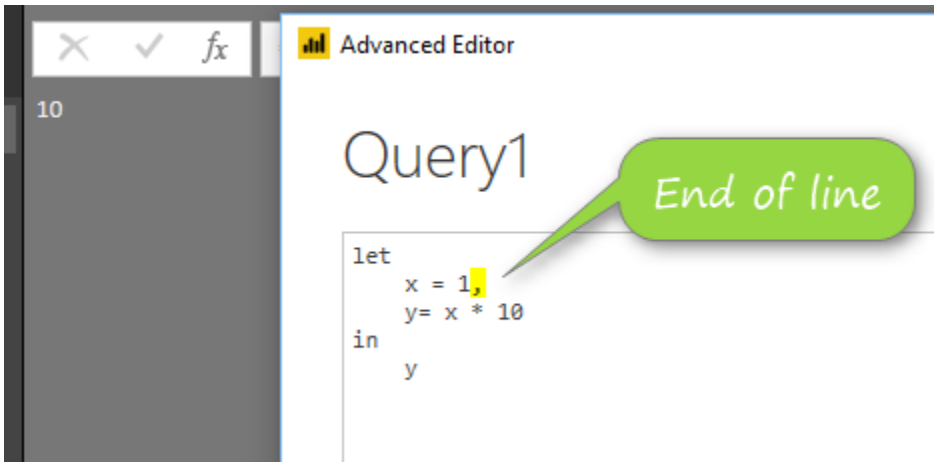
اگر یک مقدار متنی تعیین کنید آنگاه نوع متغیر به‌صورت اتوماتیک از نوع داده‌های text قرار می‌گیرد.

نکته مهم:

اگر کاراکتر مشخص‌کننده آخر فرمول را قرار ندهید، خط کدهای M ادامه می‌یابد.



همانطور که در مثال بالا مشاهده می‌کنید، خط فرمول ادامه می‌یابد و X مساوی است با1+1=x . اگر می‌خواهید یک خط فرمول را به پایان برسانید می‌بایست یک (,) در انتهای آن قرار دهید. به مثال زیر توجه کنید.

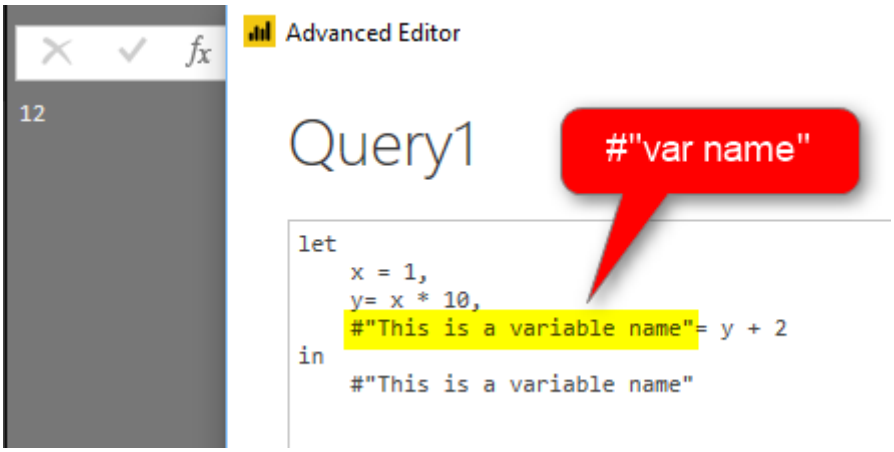


تمام خطوط فرمول می‌بایست یک (,) داشته باشند تا به پایان برسند به‌جز آخرین خط قبل از in.

نام متغیرها

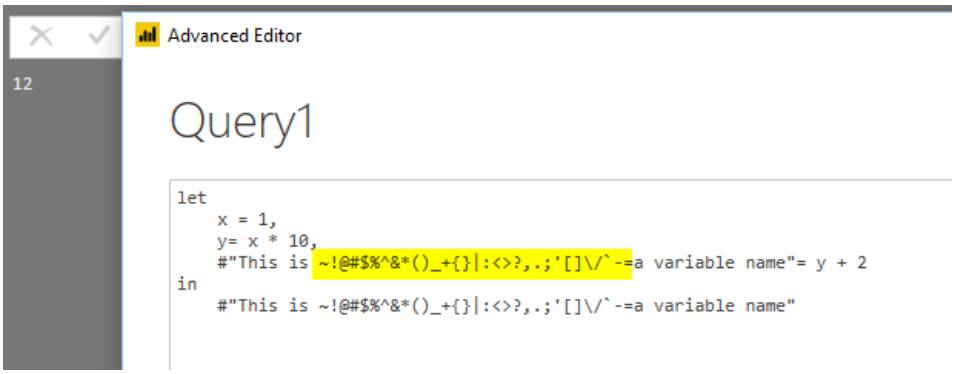
نام متغیرها می‌تواند به‌صورت یک کلمه‌ای باشد یا دارای فاصله جداکننده باشد. در شرایطی که دارای کاراکترهایی مثل فاصله باشید، می‌بایست نام متغیر را در میان (") قرار دهید و یک علامت # در ابتدای آن قرار دهید. برای مثال :

#"نام متغیر"



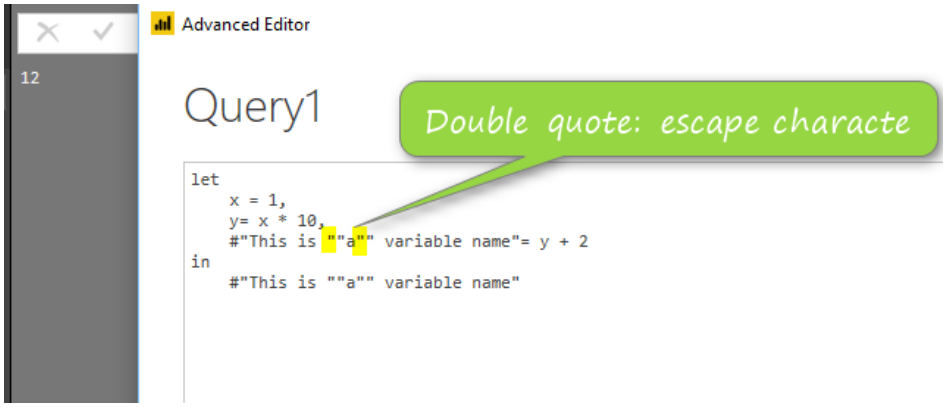
◄ کاراکترهای خاص

نام متغیرها می‌تواند دارای کاراکترهای خاص باشد. برای مثال در نمونه زیر مشاهده می‌کنید که نام متغیر دارای انواع مختلفی از کاراکترهاست و به‌خوبی نیز کار می‌کند.



کاراکتر گریز

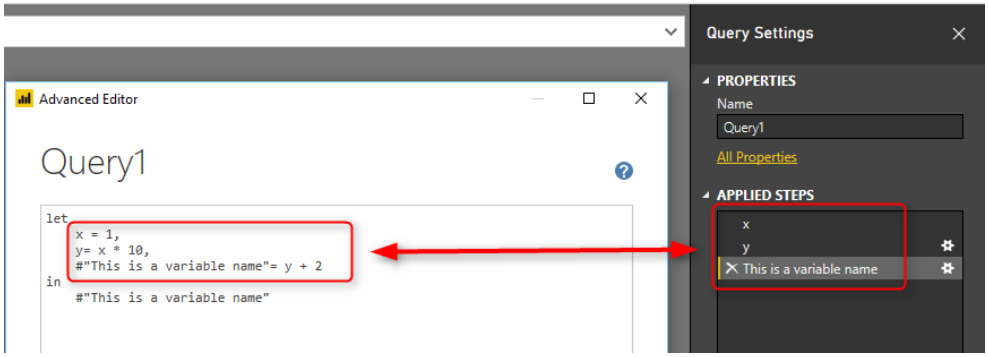
دابل کوتیشن کاراکتر گریز است. شما می‌توانید از این علامت برای تعریف نام‌هایی استفاده کنید که درون خود دارای یک دابل کوتیشن دیگر هستند. به مثال زیر دقت کنید:



اولین دابل کوتیشن (هایلایت شده) بالا می‌بایست قبل از دومین دابل کوتیشن (که قسمتی از نام متغیر است) قرار گیرد.

کدنویسی مرحله‌به‌مرحله

پاور کوئری یک تبدیل مرحله‌به‌مرحله است. زمانی‌که کدها را می‌نویسید، اگر به سمت راست توجه کنید، مشاهده خواهید کرد که هر متغیری یک مرحله را تشکیل می‌دهد.

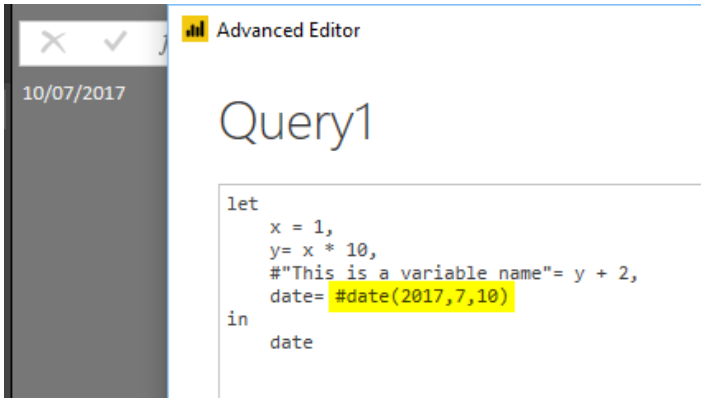


در تصویر بالا، می‌توانیم ببینیم که هر متغیر یک مرحله را تشکیل می‌دهد و اگر متغیر دارای یک فاصله در نام خود باشد، در لیست Applied steps نیز همراه با فاصله نشان داده می‌شود.

آخرین متغیر همیشه در قسمت in مشخص می‌شود.

مقادیر ثابت

دو راه برای تعریف هر مقدار ثابت در پاور کوئری وجود دارد. برای مثال، اگر می‌خواهید یک متغیر تاریخ مشخص کنید، می‌بایست به شکل زیر عمل کنید.

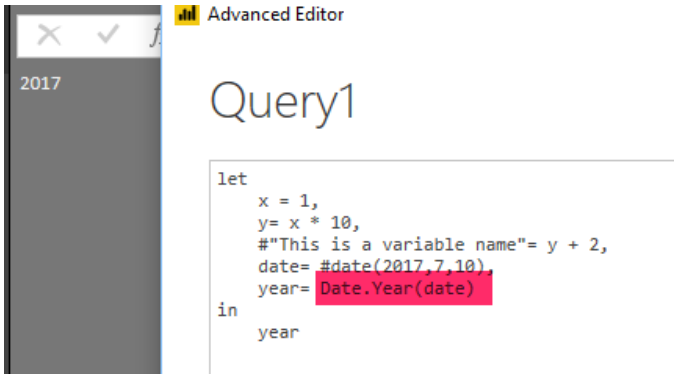


برای تعریف دیگر اقسام literal، به جدول زیر رجوع کنید:

Kind	Literal
Null	`null`
Logical	`true  false`
Number	`0  1  -1  1.5  2.3e-5`
Time	`#time(09,15,00)`
Date	`#date(2013,02,26)`
DateTime	`#datetime(2013,02,26, 09,15,00)`
DateTimeZone	`#datetimezone(2013,02,26, 09,15,00, 09,00)`
Duration	`#duration(0,1,30,0)`
Text	`"hello"`
Binary	`#binary("AQID")`
List	`{1, 2, 3}`
Record	`[ A = 1, B = 2 ]`
Table	`#table({"X","Y"},{{0,1},{1,0}})`
Function	`(x) => x + 1`
Type	`type { number }  type table [ A = any, B = text ]`

فراخوانی تابع در زبان فرمول‌نویسی M

زبان M براساس توابع است و شما برای انجام هرکاری در این زبان می‌بایست یک تابع را فراخوانی کنید. برای فراخوانی توابع فقط کافیست نام آنها را وارد کنید و پارامترهای ضروری آن را مشخص نمایید.



در تصویر بالا از Date.year استفاده شده است که قسمت سال تاریخ را مشخص می‌کند. نام توابع همیشه با حروف بزرگ شروع می‌شوند.

استفاده از کامنت در زبان فرمول‌نویسی M

مثل دیگر زبان‌های فرمول‌نویسی در اینجا هم شما می‌توانید در کدهای خود از کامنت استفاده کنید. این کامنت‌ها به دو شکل هستند.

کامنت‌نویسی یک خطی با استفاده دو خط اسلش

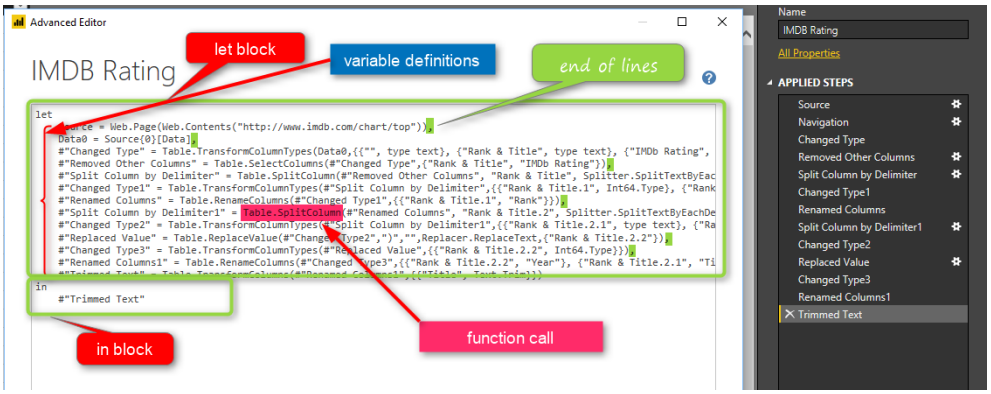
```
let
// this is a comment line and will not be executed
    x = 1,
    y= x * 10,
    #"This is a variable name"= y + 2,
    date= #date(2017,7,10),
    year= Date.Year(date)
in
    year
```

کامنت‌های چندخطی که می‌بایست بین خط اسلش و ستاره قرار گیرند.

```
let
/* this is a
multi
lin
comment and will not be executed
*/
    x = 1,
    y= x * 10,
    #"This is a variable name"= y + 2,
    date= #date(2017,7,10),
    year= Date.Year(date)
in
    year
```

مثال حقیقی

حال که با مقدمات کار آشنا شدید، به بررسی یک کوئری در حالت advanced editor می‌پردازیم:



در تصویر بالا می‌توانید تمام مقدماتی را که تاکنون به آن اشاره شد مشاهده کنید.

۱. قسمت‌های let وin
۲. تعریف نام برای متغیرها
۳. نام متغیرها همراه با علامت‌های "#" و("")
۴. کاراکترهای انتهای خط فرمول
۵. فراخوانی چندین تابع

فصل دوم

Expressions and values
عبارات و مقادیر

مقادیر

ساختار مرکزی در M از عبارات تشکیل شده است. یک عبارت را می‌توان ارزیابی (محاسبه) کرد و یک مقدار واحد به‌دست آورد.

اگرچه بسیاری از مقادیر را می‌توان به معنای واقعی کلمه به‌عنوان یک عبارت نوشت اما یک مقدار، یک عبارت نیست. به‌عنوان مثال، عبارت ۱ به مقدار ۱ ارزیابی می‌شود. عبارات ۱+۱ به مقدار ۲ ارزیابی می‌شود. این تمایز ظریف، اما مهم است. عبارات دستورالعمل‌هایی برای ارزیابی هستند. مقادیر نتایج محاسبه هستند.

مثال‌های زیر انواع مختلف مقادیر موجود در M را نشان می‌دهند. به‌عنوان یک قرارداد، یک مقدار با استفاده از شکل تحت‌اللفظی نوشته می‌شود که در آن در عبارتی ظاهر می‌شوند که فقط به آن مقدار ارزیابی می‌شود. توجه داشته باشید که // شروع یک کامنت را نشان می‌دهد که تا انتهای خط ادامه دارد.

- یک مقدار اولیه یک مقدار تک‌قسمتی است، مانند عدد، منطقی، متن یا تهی. برای نشان دادن عدم وجود هرگونه داده می‌توان از یک مقدار null استفاده کرد.

```
123// A number
true// A logical
"abc"// A text
null// null value
```

- یک لیست، یک توالی مرتب شده از مقادیر است.M بی‌نهایت لیست را پشتیبانی می‌کند. کاراکترهای { و } شروع و پایان یک لیست را نشان می‌دهند.

```
{123, true, "A"}// list containing a number, a logical, and a text
{1, 2, 3}//list of three numbers
```

- رکورد مجموعه‌ای از فیلدها است. فیلد یک جفت نام / مقدار است که در آن نام یک مقدار متنی است که در رکورد فیلد منحصربه‌فرد است. نحوۀ تحت‌اللفظی برای مقادیر رکورد اجازه می‌دهد تا نام‌ها بدون

نقل‌قول نوشته شوند، شکلی که به‌عنوان شناسه نیز شناخته می‌شود. در زیر یک رکورد حاوی سه فیلد به نام‌های "A" "B" و "C" نشان داده شده است که دارای مقادیر ۱، ۲ و ۳ هستند.

```
[
 A = 1,
 B = 2,
 C = 3
]
```

- جدول شامل مجموعه‌ای از مقادیر است که در ستون‌ها (که با نام مشخص می‌شوند) و ردیف‌ها قرار می‌گیرند. هیچ قاعده‌ای برای ایجاد جدول وجود ندارد، اما چندین تابع استاندارد وجود دارد که می‌توان از آنها برای ایجاد جداول از لیست‌ها یا رکوردها استفاده کرد.

```
#table({"A", "B"}, { {1, 2}, {3, 4} })
```

این دستور یک جدول به شکل زیر ایجاد می‌کند:

A	B
1	2
3	4

- تابع مقداری است که پس از فراخوانی، مقدار جدیدی تولید می‌کند. تابع با فهرست‌کردن پارامترهای تابع که به عنوان آرگومان داخل پرانتز قرار می‌گیرند، به دنبال علامت go-to =>، و به دنبال آن عبارتی که تابع را تعریف می‌کند، نوشته می‌شود. این عبارت معمولاً به پارامترها به‌وسیله نام آنها اشاره می‌کند.

```
(x, y) => (x + y) / 2
```

Evaluate

مدل ارزیابی زبان M پس از مدل ارزیابی که معمولاً در صفحات گسترده مانند اکسل دیده می‌شود، مدل‌سازی می‌شود، جایی که ترتیب محاسبه را می‌توان براساس وابستگی‌های بین فرمول‌ها در سلول‌ها تعیین کرد.

اگر فرمول‌هایی را در صفحه‌گسترده‌ای مانند اکسل بنویسید خواهید دید که فرمول‌های سمت چپ هنگام محاسبه به مقادیر سمت راست منجر می‌شوند:

	A
1	=A2 * 2
2	=A3 + 1
3	1

	A
1	4
2	2
3	1

درM ، بخش‌هایی از یک عبارت می‌توانند به قسمت‌های دیگر عبارت با نام ارجاع دهند و فرآیند ارزیابی به‌طور خودکار ترتیب محاسبه عبارات ارجاع‌شده را براساس اولویت‌های ریاضی و نوشتاری تعیین می‌کند.

می‌توانیم از یک رکورد برای تولید عبارتی که معادل مثال صفحه گسترده بالا است استفاده کنیم. هنگامی‌که مقدار یک فیلد را مقداردهی اولیه می‌کنیم، می‌توانیم با استفاده از نام فیلد به سایر فیلدهای داخل رکورد به‌صورت زیر مراجعه کنیم:

```
[
 A1 = A2 * 2,
 A2 = A3 + 1,
 A3 = 1
]
```

عبارت فوق معادل عبارت زیر است (که هر دو به مقادیر مساوی ارزیابی می‌شوند):

```
[
 A1 = 4,
 A2 = 2,
 A3 = 1
]
```

رکوردها را می‌توان داخل یا (تودرتو) در سایر رکوردها قرار داد. ما می‌توانیم از عملگر جستجو ([]) برای دسترسی به فیلدهای یک رکورد با نام استفاده کنیم. به‌عنوان مثال، رکورد زیر دارای یک فیلد به نام Sales که خود حاوی یک رکورد و یک فیلد به نام Total است که به فیلدهای FirstHalf و SecondHalf رکورد فروش دسترسی دارد:

```
[
 Sales = [FirstHalf = 1000, SecondHalf = 1100],
 Total = Sales[FirstHalf] + Sales[SecondHalf]
]
```

عبارت فوق در هنگام ارزیابی معادل عبارت زیر است:

```
[
 Sales = [FirstHalf = 1000, SecondHalf = 1100],
 Total = 2100
]
```

رکوردها می‌توانند در یک فهرست قرار گیرد. ما می‌توانیم از عملگر شاخص موقعیتی ({}) برای دسترسی به یک آیتم در فهرست با شاخص عددی (ایندکس) آن استفاده کنیم. مقادیر درون یک لیست با ایندکس‌هایی که از صفر آغاز می‌شوند مورد اشاره قرار می‌گیرند. به‌عنوان مثال، شاخص‌های ۰ و ۱ برای ارجاع به موارد اول و دوم در لیست زیر استفاده می‌شوند:

```
[
 Sales =
```

```
{
[
Year = 2007,
FirstHalf = 1000,
SecondHalf = 1100,
Total = FirstHalf + SecondHalf // 2100
],
[
Year = 2008,
FirstHalf = 1200,
SecondHalf = 1300,
Total = FirstHalf + SecondHalf // 2500
]
},
TotalSales = Sales{0}[Total] + Sales{1}[Total] // 4600
]
```

عبارات عضو لیست و رکورد (و همچنین عبارات let که در ادامه معرفی می‌شوند) به روش ارزیابی تنبل[۱] ارزیابی می‌شوند، به این معنی که آنها فقط در صورت نیاز ارزیابی می‌شوند. تمام عبارات دیگر با استفاده از ارزیابی مشتاق ارزیابی می‌شوند، به این معنی که آنها بلافاصله ارزیابی می‌شوند، زمانی که در طول فرآیند ارزیابی با آنها مواجه می‌شوند.

۱: ارزیابی تنبل‌وارانه (Lazy Evaluation) در زبان‌های برنامه‌نویسی تابع‌گرا ارزیابی تنبل‌وارانه یک استراتژی ارزیابی است که در آن ارزیابی یک دستور تا زمانی که مقدار آن مورد نیاز نباشد به تعویق می‌افتد. این موضوع از جهات زیادی مزیت دارد، اما یکی از مهم‌ترین آنها این است که از ارزیابی چند باره دستور جلوگیری می‌شود.

توابع

در M یک تابع، محاسبه‌ای است روی مجموعه‌ای از مقادیر ورودی و تبدیل آن به مقدار خروجی واحد است. یک تابع با نامگذاری مجموعه مورد نیاز از مقادیر ورودی (پارامترهای تابع) و سپس ارائه عبارتی نوشته می‌شود که نتیجه تابع را با استفاده از آن مقادیر ورودی (بدنه تابع) پس از go-to محاسبه می‌کند.(=>) نماد.

برای مثال:

```
(x) => x + 1// function that adds one to a value
(x, y) => x + y// function that adds two values
```

تابع یک مقدار است، درست مانند یک عدد یا یک مقدار متنی.

مثال زیر تابعی را نشان می‌دهد که نام آن Add است که، هنگامی‌که یک تابع فراخوانی می‌شود، مجموعه‌ای از مقادیر مشخص می‌شود که به‌طور منطقی جایگزین مجموعه مقادیر ورودی مورد نیاز در عبارت بدنه تابع می‌شود. (مقداردهی آگومان‌ها)

```
[
 Add = (x, y) => x + y,
 OnePlusOne = Add(1, 1),// 2
 OnePlusTwo = Add(1, 2)// 3
]
```

کتابخانه‌ها

M شامل مجموعه‌ای متداول از کلاس است که برای استفاده از متدهای آن ابتدا نام کلاس سپس نام متد نوشته می‌شود.

برای مثال:

```
Number.E// Euler's number e (2.7182...)

Text.PositionOf("Hello", "ll")// 2
```

اپراتور (عملگر) ها

M شامل مجموعه‌ای از عملگرها است که می‌توانند در عبارات استفاده شوند. عملگرها برای تشکیل عبارات نمادین روی عملوندها اعمال می‌شوند. به‌عنوان مثال، در عبارت ۱ + ۲ اعداد ۱ و ۲ عملوند هستند و عملگر جمع (+) است.

معنای یک عملگر بسته به اینکه عملوندهای آنچه نوع مقادیری هستند می‌تواند متفاوت باشد. به‌عنوان مثال، عملگر + را می‌توان با مقادیر دیگری به غیر از اعداد استفاده کرد:

```
1 + 2// numeric addition: 3
#time(12,23,0) + #duration(0,0,2,0)// time arithmetic: #time(12,25,0)
```

مثال دیگری از عملگر با معنای وابسته به عملوند، عملگر ترکیبی (&) است:

```
"A" & "BC"// text concatenation: "ABC"
{1} & {2, 3}// list concatenation {1, 2, 3}
[a = 1] & [b = 2]// record merge: [a = 1, b = 2]
```

توجه داشته باشید که همه ترکیبات مقادیر ممکن است توسط برنامه پشتیبانی نشوند.

برای مثال:

```
1 + "2"          // error: adding number and text is not supported
```

عباراتی که هنگام محاسبه، با شرایط عملگر تعریف نشده (استفاده از انواع داده که قابل تبدیل هم نیستند) مواجه می‌شوند، ایجاد خطا خواهند کرد.

در ادامه بیشتر به خطاهای M خواهیم پرداخت.

◄ فراداده (Metadata)

فراداده اطلاعاتی یک مقدار است که با یک مقدار مرتبط است. فراداده به‌عنوان یک مقدار رکورد نشان داده می‌شود که رکورد ابرداده نامیده می‌شود. از فیلدهای یک رکورد ابرداده می‌توان برای ذخیره یک مقدار استفاده کرد.

سوابق فراداده راهی برای مرتبط‌کردن اطلاعات اضافی با هر نوع ارزشی به روشی بدون مزاحمت فراهم می‌کند. مرتبط‌کردن یک رکورد ابرداده با یک مقدار، مقدار یا رفتار آن را تغییر نمی‌دهد.

یک مقدار رکورد فوق داده y با مقدار موجود x با استفاده از نحو x meta y مرتبط است. به‌عنوان مثال، نمونه زیر یک رکورد ابرداده را با فیلدهای رتبه‌بندی و برچسب‌ها با مقدار متن "Mozart" مرتبط می‌کند:

```
"Mozart" meta [Rating = 5, Tags = {"Classical"}]
```

برای مقادیری که از قبل دارای یک رکورد فراداده غیرخالی هستند، نتیجه اعمال متا، محاسبه و ادغام رکورد ابرداده موجود و جدید است. به‌عنوان مثال، دو عبارت زیر با یکدیگر و با عبارت قبلی معادل هستند:

```
("Mozart" meta [Rating = 5]) meta [Tags = {"Classical"}]
"Mozart" meta ([Rating = 5] & [Tags = {"Classical"}])
```

با استفاده از تابع Value.Metadata می‌توان به رکورد ابرداده برای یک مقدار معین دسترسی داشت. در مثال زیر، عبارت در فیلد ComposerRating به رکورد ابرداده مقدار در فیلد Composer دسترسی پیدا می‌کند و سپس به فیلد Rating دسترسی پیدا می‌کند.

```
[
Composer = "Mozart" meta [Rating = 5, Tags = {"Classical"}],
ComposerRating = Value.Metadata(Composer)[Rating] // 5
]
```

Let expression

عبارت let اجازه می‌دهد مجموعه‌ای از مقادیر محاسبه شود، نام‌هایی به آنها اختصاص داده شود و سپس در عبارت بعدی استفاده شود که در قسمت in به دنبال آن هستیم.

برای مثال:

```
let
 Sales2007 =
 [
 Year = 2007,
 FirstHalf = 1000,
 SecondHalf = 1100,
 Total = FirstHalf + SecondHalf// 2100
 ],
 Sales2008 =
 [
 Year = 2008,
 FirstHalf = 1200,
 SecondHalf = 1300,
 Total = FirstHalf + SecondHalf// 2500
 ]
 in
          Sales2007[Total] + Sales2008[Total]// 4600
```

نتیجه عبارت فوق یک مقدار عددی (۴۶۰۰) است که از مقادیر فیلدهای Sales2007 و Sales2008 به دست آمده‌اند.

If expression

عبارت if بین دو عبارت براساس یک شرط منطقی انتخاب می‌کند.

برای مثال:

```
if 2 > 1 then
 2 + 2
else
 1 + 1
```

اگر عبارت منطقی (2 > 1) درست باشد، عبارت اول (۲ + ۲) و اگر نادرست باشد، عبارت دوم (۱ + ۱) انتخاب می‌شود.

عبارت انتخاب شده با توجه به درستی شرط ۱ < ۲ مقدار ۲+۲ خواهد بود و نتیجه عبارت if (۴) می‌شود.

خطاها (Errors)

بروز خطا منجربه عدم تولید مقدار خواهد شد.

خطاها توسط عملگرها و توابع (استفاده نادرست) و در مواجهه با شرایط خطا یا با استفاده از عبارت خطا ایجاد می‌شوند.

خطاها را می‌توان با استفاده از عبارت try کنترل کرد. هنگامی‌که یک خطا رخ می‌دهد، پیغامی برای نشان‌دادن دلیل رخ دادن خطا نمایش داده می‌شود.

```
let Sales =
 [
 Revenue = 2000,
 Units = 1000,
 UnitPrice = if Units = 0 then error "No Units"
 else Revenue / Units
 ],
 UnitPrice = try Number.ToText(Sales[UnitPrice])
in "Unit Price: " &
 (if UnitPrice[HasError] then UnitPrice[Error][Message]
 else UnitPrice[Value])
```

مثال بالا به فیلد Sales[UnitPrice] دسترسی پیدا می‌کند و مقدار حاصل را تغییر نوع می‌دهد:

```
"Unit Price: 2"
```

اگر فیلد Units صفر بود، فیلد UnitPrice خطایی ایجاد می‌کرد که با try کنترل می‌شد. در این صورت مقدار حاصل"No Units" خواهد بود.

عبارت try مقادیر و خطاهای مناسب را به یک مقدار رکورد تبدیل می‌کند که نشان می‌دهد آیا عبارت try مدیریت شده و خطا دارد یا نه و یا مقدار مناسب یا رکورد خطای که هنگام مدیریت خطا استخراج شده است.

برای عنوان مثال، عبارت زیر را درنظر بگیرید که یک خطا را ایجاد می‌کند و بلافاصله آن را مدیریت می‌کند.

```
try error "negative unit count"
```

این عبارت به مقدار رکورد تودرتوی زیر رجوع می‌کند و جستجوهای فیلد [HasError]، [Error] و [Message] را در مثال قیمت واحد قبل توضیح می‌دهد.

```
[
 HasError = true,
 Error =
 [
 Reason = "Expression.Error",
 Message = "negative unit count",
 Detail = null
 ]
]
```

روش رایج استفاده از try، جایگزینی خطاها با مقادیر پیش‌فرض است.

کامنت‌ها Comments

برای ایجاد کامنت می‌توانید از کاراکترهای زیر استفاده کنید:

کامنت‌های تک‌خطی (//):

```
let
//Convert to proper case.
 Source = Text.Proper("hello world")
in
 Source
```

کامنت‌های چند خطی (/* ....*/):

```
/*
Capitalize each word in the Item column in the Orders table.Text.Proper
is evaluated for each Item in each table row.
*/
let
 Orders = Table.FromRecords({
 [OrderID = 1, CustomerID = 1, Item = "fishing rod", Price = 100.0],
 [OrderID = 2, CustomerID = 1, Item = "1 lb.worms", Price = 5.0],
 [OrderID = 3, CustomerID = 2, Item = "fishing net", Price = 25.0]}),
 #"Capitalized Each Word" = Table.TransformColumns(Orders, {"Item", Text.Proper})
in
 #"Capitalized Each Word"
```

Operators

زبان فرمول‌نویسی Power Query M شامل مجموعه‌ای از عملگرها است که می‌توانند در عبارت استفاده شوند. عملگرها برای تشکیل عبارات نمادین به عملوندها اعمال می‌شوند. به‌عنوان مثال، در عبارت ۱ + ۲ اعداد ۱ و ۲ عملوند هستند و عملگر جمع (+) است.

معنای عملگر بسته به نوع مقادیر عملوند می‌تواند متفاوت باشد. این زبان دارای عملگرهای زیر است:

Plus operator (+)

Expression	Equals
1 + 2	Numeric addition: 3
#time(12,23,0) + #duration(0,0,2,0)	Time arithmetic: #time(12,25,0)