

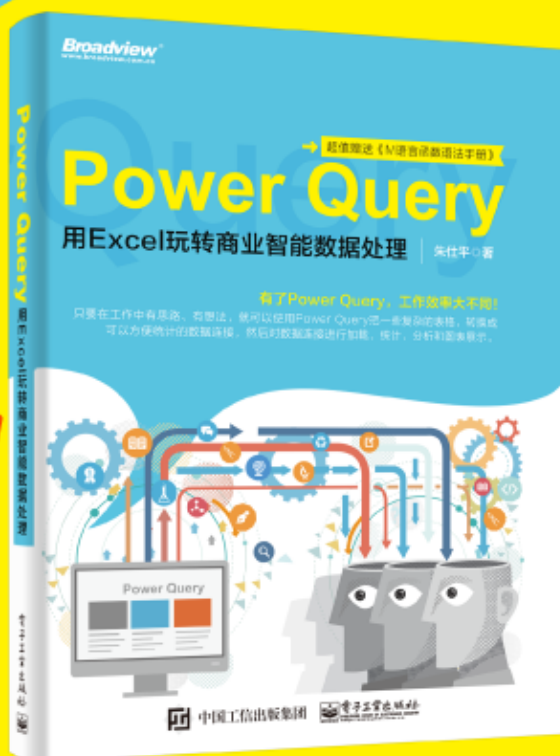
Power Query

用Excel 玩转商业智能数据处理

★
超值赠送
《M语言函数
语法手册》

有了Power Query
工作效率大不同!

- 朱仕平 著
- ISBN 978-7-121-31461-2
- 定价: 59.00元



Power Query, Excel的另一个江湖

- ★ 本书会为读者揭开Power Query的神秘面纱, 将其强大的功能呈现出来。
- ★ Power Query 在Office 2016 中已经被作为内置的工具嵌入【数据】选项卡中, 其集成了Access和Excel的功能, 可以对数据进行可视化菜单操作, 完成对数据的提取 (Extract)、(Transform)、加载 (Load)。

只要在工作中有思路、有想法，就可以使用Power Query把一些复杂的表格，转换成可以方便统计的数据连接，然后对数据连接进行加载、统计、分析、图表展示。



• 本书适合谁 •

本书适合从事财务、统计、人力资源、客服、售后服务、电商等工作，需要处理大量数据的读者

• 作者简介 •

朱仕平

- Excel资深爱好者，骨灰级玩家
- Excel精英培训网讲师团讲师
- 腾讯课堂在线教育签约讲师
- 网易云课堂在线教育签约讲师
- 爱学习创办人
- 具有15年Office职场实战应用经验，10年企业及在线教育培训经验。擅长Excel、Word、PPT软件应用，推崇使用综合实用技能提高工作效率，通过在线教育推广常用功能技巧、函数公式嵌套数组应用、数据透视表及SQL应用、图形图表、Power Query、Power Pivot、Power View、Power Map、Power BI等实战课程。



版权声明

此附件文档作为《Power Query 用 Excel 玩转商业智能数据处理》扩展阅读文件，受知识产权保护，仅供购书读者参考学习，任何形式的转载和分享都应保证该文件的完整性。禁止以任何形式侵犯作者的著作权。

视频教程送书活动

凡购买本书籍的读者，都有机会参与购买视频课程抵扣书费的活动
抵扣书费，以视频课程实时活动及实时购买书价为准
直接从作者购买签名书籍由读者承担运费(10 元/册)
视频课程地址：

https://ke.qq.com/course/139924#tuin=1e72540c&term_id=0

授课老师：朱仕平（本书作者）

联系 QQ: 510809100



完整章节目录

纸质出版书籍在前面 5 章, 本电子文档附件 A 只包含附件扩展阅读

第 1 章 Power Query, Excel 的另一个江湖

- 1.1 揭开 Power Query 的神秘面纱
- 1.2 Power Query 职场应用 5 招鲜

第 2 章 稳扎稳打, 练好 Power Query 的基本功

- 2.1 管理 Power Query
- 2.2 掌握 Power Query 的基础功能

第 3 章 学以致用, Power Query 应用案例

- 3.1 拆分数据
- 3.2 合并号段
- 3.3 对比数据
- 3.4 计算单元格字符
- 3.5 计算单元格数值
- 3.6 拆解信息
- 3.7 动态转换结构
- 3.8 合并统计数据
- 3.9 转换数据再创建数据透视表
- 3.10 拆分和转换数据结构
- 3.11 数据排名
- 3.12 合并文件夹
- 3.13 合并查询聚合计算
- 3.14 动态查询
- 3.15 多条件动态查询
- 3.16 综合案例: 制订采购计划

第 4 章 知己知彼, Power Query 的结构组成

- 4.1 查询表的结构关系
- 4.2 高级编辑器
- 4.3 创建一条 Record 记录
- 4.4 创建多行 Record 记录
- 4.5 创建一个 List 列表
- 4.6 创建列表中的 List 列表
- 4.7 创建由逗号分隔的 List 列表

4.8 创建 List 列表的基本约定

4.9 创建 table 表

4.10 创建多层嵌套 table 表

第 5 章 事半功倍, Power Query 的高级应用

5.1 深化钻取

5.2 获取重复值

5.3 快速计算

5.4 条件判断

5.5 模糊匹配

5.6 拆分数据

5.7 创建凭证表

5.8 文本和数值混合提取

5.9 统计核对数据

5.10 行列转置

5.11 横向排序

5.12 综合拆解数据

5.13 自定义函数

附件 A 常用 M 语言函数语法介绍

A.1 M 语言基础

A.2 Text 类函数

A.3 Number 类函数

A.4 Time 类函数

A.5 Date 类函数

A.6 DateTime 类函数

A.7 Duration 类函数

A.8 Record 类函数

A.9 List 类函数

A.10 Table 类函数

A.11 文件类函数

常用 M 语言函数语法手册

从前面的 5 章中我们已经了解到，使用 Power Query 菜单的操作过程都会被记录成一个公式，这种公式被称为 M 语言公式。

M 语言的函数体系非常庞大，包含了大约 90 个函数类别，总共涉及超过 600 个函数，要完全掌握 M 语言的所有函数几乎是不可能完成的任务，就像 Excel 工作表函数，能够熟练应用常用的几十个函数就已经非常了不起了。

在学习 M 语言的函数时，不应过分沉迷，因为任何工具都有其特点。尺有所短，寸有所长，取长补短，高效完成工作才是关键。

本章介绍几种常用的函数类别，一些比较偏门的应用，这里就不进行介绍了。本书所著亦是作者根据学习理解，通过大量案例演示总结出的观点，不代表开发者的设计初衷，可能会有理解偏差或不完善的地方。

A.1 M 语言基础

在前面的章节中，我们已经接触过大量的实用案例、公式函数和高级编辑器的应用。M 语言实际就是通过函数公式将结果传递给变量，每个变量对应一个步骤，每个变量的步骤环环相扣；这些公式可以使用现成的函数，也可以使用自定义函数。但需要特别注意的是，公式中的函数和参数对大小写都非常敏感。

每个查询公式都指向前一个步骤的变量名称，前一个步骤的变量名称就是一个实际的结果，这个结果可以是 Value、Record、List、Table。如果公式太长，则可以在中间任意地方强制换行；但是每个公式在最后都应该输入一个逗号，然后换行到第二个步骤。

直到最后一个步骤时，才不再需要输入逗号，将最后一个查询步骤作为最终的结果，使用 in 语句把这个步骤传递回【查询编辑器】。

在如下所示的案例中，需要在查询表中根据【品名】字段删除重复项，如图 A-1 所示。

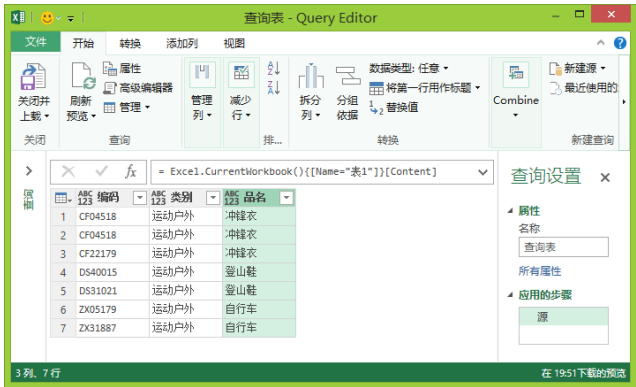


图 A-1

选择【品名】字段后，右击，在弹出的菜单中选择【删除重复项】选项，获得如图 A-2 所示的结果。

	ABC 123	编码	ABC 123	类别	ABC 123	品名
1		CF04518		运动户外		冲锋衣
2		DS40015		运动户外		登山鞋
3		ZX05179		运动户外		自行车

图 A-2

打开【高级编辑器】，可以看到 M 语言公式的步骤如图 A-3 所示。



图 A-3

公式为：

```
let
```

Powery Query：用 Excel 玩转商业智能数据处理

```
源 = Excel.CurrentWorkbook()[[Name="表 1"]][Content],
删除的副本 = Table.Distinct(源, {"品名"})
in
删除的副本
```

对于此公式的具体介绍下。如表 A-1 所示。

表 A-1

步 骤	描 述
Let	表示一个查询的开始
源	创建输入一个查询表
删除的副本	变量名称，这个变量通过函数 Table.Distinct 删除上一步骤中【品名】字段的重复项
in	表示一个查询的结束，查询结束后，输出到【查询编辑器】中的结果使用删除的副本这个步骤的结果

如果将这些批注信息直接写到【高级编辑器】中，则结果如下。

```
let          //一个查询的开始
    源 = Excel.CurrentWorkbook()[[Name="表 1"]][Content],
    删除的副本 = Table.Distinct(源, {"品名"})
in          //一个查询的结束
    删除的副本          //输出结果
```

在书写批注的时候，需要先输入两条斜线“//”；在书写批注的过程中，除最后一行过程语句不需要加逗号外，其他语句都必须加逗号。

每条语句都会生成一个查询结果。这个结果可以是以下类型：

- a) 值 (number, text, date, time, datetime)
- b) 记录 (Record)
- c) 列表 (List)
- d) 表格 (Table)

步骤公式语法分解如下（见表 A-2）。

```
删除的副本 = Table.Distinct(源, {"品名"})
```

表 A-2

参 数	说 明
删除的副本	变量名
Table	函数类别
Distinct	函数名称
源	输入一个查询表
{"品名"}	指定一个列字段名称，可以同时选择多个字段

1. 运算符

在 M 语言中，也可以像 Excel 工作表函数一样使用运算符。Excel 工作表函数仅对单元格进行操作运算，在 M 语言中，运算符可以对记录 (record)、列表 (list)、表格 (table) 进行操作运算。

要特别注意，不同数据类型的数据不可以直接进行计算，比如数值 1 不能与文本数值"1"直接计算，1+"1"会导致错误的发生。因此，如果需要使用不同数据类型的参数进行计算，一定要使用转换函数将数据转换成相同类型后再计算。

(1) 组合运算符 (适用于文本、列表、记录、表格等连接)

例如：

```
= "四川" & "成都" & "大熊猫基地"    输出    "四川成都大熊猫基地"
= {1,2,3} & {4,5} & {6,7,8,9}    输出    {1,2,3,4,5,6,7,8,9}
= [职务="园长"] & [年龄=36]        输出    [职务="园长" , 年龄=36]
= #table({"姓名"},{{"张飞"}})&#table({"姓名"},{{"李奎"}})
//输出    #table({"姓名"},{{"张飞"},"李奎"}})
```

(2) 比较运算符 (适用于逻辑值数字、时间、日期、日期时间时区、文本、二进制见表 A-3)

表 A-3

运算符	含 义
>	大于
>=	大于或等于
<	小于
<=	小于或等于
=	等于
<>	不等于

(3) 逻辑运算符 (见表 A-4)

表 A-4

运算符	含 义
or	或
and	与
not	非

(4) 算术运算符 (见表 A-5)

表 A-5

运算符	含 义
+	加
-	减
*	乘
/	除

(5) 表级运算符 (见表 A-6)

表 A-6

运算符	含 义
[]	记录单
{ }	列表

2. 数据转换

在 Power Query 中，数据转换所用的函数及说明如表 A-7 所示。

表 A-7

函 数	说 明
Number.From	从 value 转换成数值
Number.FromText	从文本数值转换成数值
Number.ToText	从数值转换成指定格式的文本数值
Text.From	返回文本表示的数值、时间、日期、二进制值等
Logical.FromText	从文本逻辑值转换成逻辑值
Logical.ToText	从逻辑值转换成文本逻辑值
Logical.From	从 value 转换成逻辑值
Date.FromText	从文本日期转换成日期
Date.From	从 value 转换成日期
Date.ToText	返回指定文本格式的日期
Date.ToRecord	返回年、月、日记录单

3. 函数分类

M 语言函数分类包含但不限于以下表 A-8 种类，读者可根据实际需求选择学习相关应用，如表 A-8 所示。

表 A-8

函 数	函 数	函 数	函 数
Access	Table	Record	Hdfs
Compression	Any	Time	Null
Error	Culture	BinaryFormat	SapBusinessObjects
Int8	ExtraValues	DateTime	Variable
Odbc	Lines	Function	Character
Splitter	PostgreSQL	MissingField	Double
ActiveDirectory	Teradata	Replacer	Int16
Csv	AzureStorage	Type	Number
Excel	Currency	BinaryOccurrence	SharePoint
JoinAlgorithm	Facebook	DateTimeZone	Web
OleDb	List	GroupKind	Combiner
Sql	Precision	MySQL	Duration
AdoDotNet	Text	RoundingMode	Int32
CsvStyle	Binary	Uri	OData
Exchange	DB2	Byte	Single
JoinKind	File	Day	Xml
Oracle	Logical	HdInsight	Comparer
Sybase	QuoteStyle	None	Embedded
AnalysisServices	TextEncoding	Salesforce	Int64
Cube	BinaryEncoding	Value	Occurrence
Expression	Date	ByteOrder	Soda
Json	Folder	Decimal	
Order	Marketplace		

4. 函数语法分解

下面以函数 :Number.Round(**number** as nullable number, *optional* **digites** as nullable number, *optional* **roundingMode** as nullable RoundingMode.Type) as nullable number 为例来介绍函数

此函数一共有 3 个参数，分别是 number、digites 和 roundingMode。

在这 3 个参数中，第一个参数 number 是一个必需参数，另外两个参数是可选参数。

在函数中必须输入必需参数，可选参数如果为省略的情况，则传递默认参数。

参数前面带有 optional，表示此参数是可选参数，不带此单词的参数是必需参数。

as 语法决定了输入参数和输出结果的数据类型，number 表示数字类型，RoundingMode.Type 是特殊参数类型，共有 5 组可选常数，决定数值的舍入方向。

数据类型前面如果有 nullable，则说明该参数可以是一个具备 null 值的空值。

此函数输出结果的数据类型为数字。

5. 部分函数参数说明

culture：地区语言选项，默认为安装软件地区的语言，参数如表 A-9 所示。

表 A-9

参 数	说 明
zh-cn	中文
en-us	English

comparer：识别比较规则，参数如表 A-10 所示。

表 A-10

参 数	说 明
Comparer.Ordinal	区分大小写
Comparer.OrdinalIgnoreCase	不区分大小写

occurrence：获取位置信息，参数如表 A-11 所示。

表 A-11

参 数	说 明
Occurrence.First	第 1 次位置
Occurrence.Last	最后 1 次位置
Occurrence.All	所有位置

RoundingMode：数值舍入，参数如表 A-12 所示。

表 A-12

参 数	说 明
RoundingMode.Up	向上舍入
RoundingMode.Down	向下舍入
RoundingMode.AwayFromZero	远离零方向舍入
RoundingMode.TowardZero	靠近零方向舍入
RoundingMode.ToEven	舍入到最近的偶数

Precision：计算数度参数，如表 A-13 所示。

6-13

参 数	说 明
Precision.Double	双精度
Precision.Decimal	十进制

missingField：搜索不到字段时返回的值，如表 A-14 所示。

6-14

参 数	说 明
MissingField.Error	错误提醒
MissingField.Ignore	忽略此错误
MissingField.UseNull	显示为 Null

includeNulls：参数计算时是否需要包含 null，如表 A-15 所示。

6-15

参 数	说 明
TRUE	包含 Null
FALSE	忽略 Null

comparisonCriteria：比较规则，如表 A-16 所示。

6-16

参 数	说 明
Order.Ascending	升序
Order.Descending	降序

replacer：替换规则，如表 A-17 所示。

6-17

参 数	说 明
Replacer.ReplaceValue	替换值
Replacer.ReplaceText	替换字符串

combiner：组合器，如表 A-18 所示。

6-18

参 数	说 明
Combiner.CombineTextByDelimiter	指定符号
Combiner.CombineTextByEachDelimiter	指定使用序列中每个字符分隔

续表

参 数	说 明
Combiner.CombineTextByLengths	指定长度
Combiner.CombineTextByPositions	指定位置
Combiner.CombineTextByRanges	偏移位置

splitter：拆分器，如表 A-19 所示。

6-19

参 数	说 明
Splitter.SplitByNothing	不拆分
Splitter.SplitTextByAnyDelimiter	指定使用序列中每个字符分隔
Splitter.SplitTextByDelimiter	指定符号
Splitter.SplitTextByEachDelimiter	指定符号拆成两列
Splitter.SplitTextByLengths	指定长度
Splitter.SplitTextByPositions	指定位置
Splitter.SplitTextByRanges	依稀位置
Splitter.SplitTextByWhitespace	指定空格

JoinKind：连接种类，如表 A-20 所示。

6-20

参 数	说 明
JoinKind.Inner	内部连接
JoinKind.LeftOuter	左外部连接
JoinKind.RightOuter	右外部连接
JoinKind.FullOuter	完全外部连接
JoinKind.LeftAnti	左反连接
JoinKind.RightAnti	右反连接

A.2 Text 类函数

Text 类函数说明如表 A-21 所示

表 A-21

类	分 类	函 数 名	概 述
1	计算	Text.Length	返回字符串长度
2	转换	Text.From	返回 Value 的文本表现形式
3	转换	Character.FromNumber	根据数值返回数值字符
4	转换	Character.ToNumber	根据数值字符返回数值
5	转换	Text.ToList	将字符串拆成独立字符的 list 列表
6	转换	Value.FromText	将文本数值转换成数值
7	获取	Text.At	返回指定位置的字符
8	获取	Text.Range	从指定位置开始返回指定个数的字符串
9	获取	Text.Middle	从中间获取字符串
10	获取	Text.Start	从起始位置获取字符串
11	获取	Text.End	从结束位置获取字符串
12	获取	Text.Insert	在指定的位置插入字符串
13	获取	Text.Remove	删除指定的字符集合 list
14	获取	Text.RemoveRange	从指定位置删除若干个字符
15	获取	Text.Replace	将指定字符串替换成新字符
16	获取	Text.ReplaceRange	从指定位置起将若干个字符替换成新字符串
17	判断	Text.Contains	判断 A 字符串是否包含 B 字符串
18	判断	Text.EndsWith	判断 A 字符串末尾位置是否包含 B 字符串
19	判断	Text.StartsWith	判断 A 字符串起始位置是否包含 B 字符串
20	判断	Text.PositionOf	判断 B 字符串在 A 字符串中的位置
21	判断	Text.PositionOfAny	判断 A 字符串中另一字符集合 list 列表位置
22	转换	Text.Clean	清除字符串中非打印字符
23	转换	Text.Lower	所有字母转换为小写
24	转换	Text.Upper	所有字母转换为大写
25	转换	Text.Proper	所有分隔的单词首字母大写
26	转换	Text.PadEnd	在其尾部插入指定字符满足定长
27	转换	Text.PadStart	在其前端插入指定字符满足定长
28	转换	Text.Repeat	字符串重复指定次数
29	转换	Text.Split	按指定分隔符拆分字符串为 list 列表
30	转换	Text.SplitAny	按指定字符串中的每个字符拆分字符串为 list 列表
31	转换	Text.Trim	清除字符串两端指定的字符
32	转换	Text.TrimStart	清除字符串前端指定的字符
33	转换	Text.TrimEnd	清除字符串末端指定的字符
34	转换	Text.Combine	将字符集 list 列表合并成字符串

Powery Query: 用 Excel 玩转商业智能数据处理

1. Text.Length

函数 : Text.Length

语法 : Text.Length(**text** as nullable text) as nullable number

说明 : 返回文本字符串的长度, 返回的结果是一个数字。

如果此参数是其他数据类型则, 需要先转换, 否则返回 Error 错误, 例如 :

= Text.Length("Power Query")	输出	11
------------------------------	----	----

如图 A-4 所示, 当参数不是文本字符串时, 返回 Error 错误。

字符串	字符串长度
1 Hello World	11
2 中国, 我爱你	6
3 文本数字	4
4 '12345	6
5 纯数字	3
6 12345	Error
7 TRUE	Error

图 A-4

2. Text.From

函数 : Text.From

语法 : Text.From(**value** as any, *optional* **culture** as nullable text) as nullable text

说明 : 返回 value 的文本表现形式, culture 是一个语言选项。

culture 常用的参数"en-us"表示英语, "zh-cn"表示中文, 省略时以安装的地区语言为参考, 例如 :

= Text.From (168)	输出	"168"
-------------------	----	-------

此函数支持的数据类型有 number、date、time、datetime、datetimezone、logical、duration 或 binary 值, 如果值为 null, 则返回结果为 null, 例如 :

= Text.From(#datetime(2016,10,1,10,12,03), "zh-cn")	输出	2016/10/1 10:12:03
= Text.From(#datetime(2016,10,1,10,12,03), "en-us")	输出	10/1/2016 10:12:03 AM

3. Character.FromNumber

函数 : Character.FromNumber

语法 : Character.FromNumber(**number** as nullable number) as nullable text

说明 : 返回数字对应的字符, 若参数为 null, 则返回的结果为 null, 例如 :

= Character.FromNumber(66)	输出	"B"
----------------------------	----	-----

4. Character.ToNumber

函数 : Character.ToNumber

语法 : Character.ToNumber(**character** as nullable text) as nullable number

说明 : 返回字符对应的数字, 若参数为 null, 则返回结果为 null, 例如 :

= Character.ToNumber("B")	输出	66
---------------------------	----	----

5. Text.ToList

函数 : Text.ToList

语法 : Text.ToList(**text** as text) as list

说明 : 将文本值 text 拆成独立字符的 list 列表。

= Text.ToList("ABCDEFGH")	//输出	{"A", "B", "C", "D", "E", "F", "G", "H"}
= Text.ToList("123456")	//输出	{"1", "2", "3", "4", "5", "6", "7"}

使用此函数必须要注意的是, 当需要对数字进行拆分时, 必要的文本参数不能直接将数字拆分成 list 列表, 必须转换成文本数值后才能拆分; 返回的结果依然是文本数字的 list 列表, 不能直接当成数字进行求和计算。

若要将数字转换成文本字符串, 则需要使用公式, 例如 :

=Text.From(1234567)	输出	"1234567"
---------------------	----	-----------

若要将文本数字 list 转换成数字 list, 则需要使用公式 :

=List.Transform({"1","2","3","4","5","6","7"},Number.From)

结果为 :

=List.Transform({"1","2","3","4","5","6","7"},Number.From)	输出	{1,2,3,4,5,6,7}
--	----	-----------------

6. Value.FromText

函数 : Value.FromText

语法 : Value.FromText(**text** as any, *optional* **culture** as nullable) as any

说明 : 将文本形式的 text 解码为实际数据类型的值, 例如 :

= Value.FromText("123")	输出	123
-------------------------	----	-----

使用 Value.From Text 函数, 不论什么样式的文本, 都能解码转换成最恰当的数据类型。

Powery Query：用 Excel 玩转商业智能数据处理

= Table.AddColumn(源, "转换", each Value.FromText([字符串]))	
字符串	转换
1 1234	1234
2 true	TRUE
3 10:20	10:20:00
4 2015-1-9	2015/01/09
5 ABC	ABC

图 A-5

7. Text.At

函数：Text.At

语法：Text.At(text as nullable text, index as number) as nullable text

说明：返回 text 字符串中 index 位置上的一个字符，字符位置从 0 开始，例如：

=Text.At("ABCDEFGH", 4)	输出	"E"
-------------------------	----	-----

8. Text.Range

函数：Text.Range

语法：Text.Range(text as nullable text, offset as number, optional count as nullable number) as nullable text

说明：从 text 中的指定偏移位置 Offset 开始，返回指定 count 个字符的新字符串；如果省略 count 参数，则返回从 offset 位置开始的所有字符组成的字符串，字符位置从 0 开始，例如：

=Text.Range("ABCDEFGH", 2, 3)	输出	"CDE"
=Text.Range("ABCDEFGH", 2)	输出	"CDEFGH"

9. Text.Middle

函数：Text.Middle

语法：Text.Middle(text as nullable text, Start as number, optional count as nullable number) as nullable text

说明：从 text 中的指定偏移位置 Start 开始，返回指定 count 个字符的新字符串；如果省略 count 参数，则返回从 offset 位置开始的所有字符组成的字符串，字符位置从 0 开始，例如：

=Text.Middle("ABCDEFGH", 2, 3)	输出	"CDE"
=Text.Middle("ABCDEFGH", 2)	输出	"CDEFGH"

10. Text.Start

函数：Text.Start

语法：Text.Start(text as nullable text, count as number) as nullable text

说明：返回 text 的前面 count 个字符的字符串，例如：

<code>=Text.Start("ABCDEFGH", 3)</code>	输出	"ABC"
---	----	-------

11. Text.End

函数：Text.End

语法：Text.End(text as nullable text, count as number) as nullable text

说明：返回 text 的后面 count 个字符的字符串，例如：

<code>=Text.End("ABCDEFGH", 3)</code>	输出	"EFG"
---------------------------------------	----	-------

12. Text.Insert

函数：Text.Insert

语法：Text.Insert(text as nullable text, offset as number, newText as text) as nullable text

说明：将新字符串 newText 插入到 text 的 offset 位置，获得新字符串，位置从 0 开始，例如：

<code>= Text.Insert("ABCDEFGH", 3, "学习")</code>	输出	"ABC 学习 DEFG"
---	----	---------------

13. Text.Remove

函数：Text.Remove

语法：Text.Remove(text as nullable text, removeChars as any) as nullable text

说明：返回已经在 text 中删除了 removeChars 中所有字符的新字符串。

removeChars 可以是单独的一个文本字符，也可以是文本字符的 list 列表。

<code>= Text.Remove("ABCDEFGH", "A")</code>	输出	"BCDEFGH"
<code>= Text.Remove("ABCDEFGH", {"A", "C", "G"})</code>	输出	"BDEFH"
<code>= Text.Remove("213AB014CDEFG354", {"A".."Z"})</code>	输出	"213014354"
<code>= Text.Remove("AB 我爱 CDEFG 学习", {"一".."𠄎"})</code>	输出	"ABCDEFGH"

14. Text.RemoveRange

函数：Text.RemoveRange

语法：Text.RemoveRange(text as nullable text, offset as number, optional count as nullable number) as nullable text

说明：返回 text 已经删除 offset 位置 1 个字符，若指定了 count 参数，则按指定的字符数删除；count 省略时默认为 1，offset 位置从 0 开始，例如：

<code>= Text.RemoveRange("ABCDEFGH", 3)</code>	输出	"ABCEFGH"
<code>= Text.RemoveRange("ABCDEFGH", 2, 4)</code>	输出	"ABG"

15. Text.Replace

函数：Text.Replace
语法：Text.Replace(**text** as nullable text, **old** as text, **new** as text) as nullable text
说明：将文本字符串 text 中的 old 字符串，替换成 new 新字符串，并区分大小写，例如：

= Text.Replace ("ABCDEFG", "BCD", "123")

输出

"A123EFG"

16. Text.ReplaceRange

函数：Text.ReplaceRange
语法：Text.ReplaceRange(**text** as nullable text, **offset** as number, **count** as number, **newText** as text) as nullable text
说明：从 text 字符串的 offset 位置开始，删除 count 个字符，并在相同位置用 newText 字符串替换。位置从 0 开始，例如：

= Text.ReplaceRange ("ABCDEFG", 2, 4, "123")

输出

"AB123G"

17. Text.Contains

函数：Text.Contains
语法：Text.Contains(**text** as nullable text, **Substring** as text, *optional* **comparer** as nullable function) as nullable logical
说明：检测文本 text 是否包含 Substring 字符串，如果包含则返回 TRUE，否则返回 FALSE。
如表 A-22 所示为可选参数 Comparer 的参数说明，默认区分大小写比较。

表 A-22

参 数	说 明
Comparer.Ordinal	区分大小写比较
Comparer.OrdinalIgnoreCase	不区分大小写比较
Comparer.FromCulture	区分地区语言

= Text.Contains ("ABCDEFG", "BCd")

输出

FALSE

= Text.Contains ("ABCDEFG", "BCd", Comparer.OrdinalIgnoreCase)

输出

TRUE

18. Text.EndsWith

函数：Text.EndsWith
语法：Text.EndsWith(**text** as nullable text, **Substring** as text, *optional* **comparer** as nullable function) as nullable logical

说明：检测文本 `text` 是否以 `substring` 字符串开头，如果是则返回 `TRUE`，否则返回 `FALSE`。

如表 A-23 所示为可选参数 `Comparer` 的参数说明，默认区分大小写比较。

表 A-23

参 数	说 明
<code>Comparer.Ordinal</code>	区分大小写比较
<code>Comparer.OrdinalIgnoreCase</code>	不区分大小写比较
<code>Comparer.FromCulture</code>	区分地区语言

```
= Text.EndsWith("ABCDEFGH", "EfG")           输出    FALSE
= Text.EndsWith("ABCDEFGH", "EfG", Comparer.OrdinalIgnoreCase)
                                           输出    TRUE
```

19. Text.StartsWith

函数：`Text.StartsWith`

语法：`Text.StartsWith(text as nullable text, Substring as text, optional comparer as nullable function)`
as nullable logical

说明：检测文本 `text` 是否以 `substring` 字符串开头，如果是则返回 `TRUE`，否则返回 `FALSE`。

如表 A-24 所示为可选参数 `Comparer` 的参数说明，默认区分大小写比较。

表 A-24

参 数	说 明
<code>Comparer.Ordinal</code>	区分大小写比较
<code>Comparer.OrdinalIgnoreCase</code>	不区分大小写比较
<code>Comparer.FromCulture</code>	区分地区语言

```
= Text.StartsWith("ABCDEFGH", "AbC")           输出    FALSE
= Text.StartsWith("ABCDEFGH", "AbC", Comparer.OrdinalIgnoreCase)
                                           输出    TRUE
```

20. Text.PositionOf

函数：`Text.PositionOf`

语法：`Text.PositionOf(text as nullable text, Substring as text, optional occurrence as nullable Occurrence.type, optional comparer as function) as any`

说明：返回 `text` 字符串中找到的文本值 `substring` 指定出现次数的位置，例如：

如表 A-25 所示为可选参数 `Occurrence` 的参数说明，默认返回第一次出现的位置。

```
= Text.PositionOf("ABC aBc aBC", "ABC")       输出    0
= Text.PositionOf("ABC aBc aBC", "AbC", Occurrence.First)
```

Powery Query：用 Excel 玩转商业智能数据处理

```

                                输出      -1
= Text.PositionOf("ABC aBc aBC","AbC",Occurrence.Last,Comparer.OrdinalIgnoreCase)

                                输出      8
= Text.PositionOf("ABC aBc aBC","AbC",Occurrence.All,Comparer.OrdinalIgnoreCase)

                                输出      {0,4,8}
```

表 A-25

参 数	说 明
Occurrence.Firs	检索第 1 次位置
Occurrence.Last	检索最后 1 次的位置
Occurrence.All	检索所有的位置

如表 A-26 所示为可选参数 Comparer 的参数说明，默认区分大小写比较。

表 A-26

参 数	说 明
Comparer.Ordinal	区分大小写比较
Comparer.OrdinalIgnoreCase	不区分大小写比较
Comparer.FromCulture	区分地区语言

21. Text.PositionOfAny

函数：Text.PositionOfAny

语法：Text.PositionOfAny(text as nullable text, characters as list, optional occurrence as nullable Occurrence.type) as any

说明：返回 text 字符串中，characters 中包含的任意字符串第 1 次出现的位置；第二参数仅支持 list 列表，必须使用大括号，例如{"A"}，并且区分大小写，例如：

```

= Text.PositionOfAny("ABCDE",{"B"})           输出      1
= Text.PositionOfAny("ABCDEEFG",{"M","K","N"},Occurrence.Last)
                                                输出      -1
= Text.PositionOfAny("ABCDEEFG",{"B","K","F"},Occurrence.Last)
                                                输出      6
= Text.PositionOfAny("ABC aBc aBCDEFG",{"B","K","F"},Occurrence.All)
                                                输出      {1,5,9,13}
```

如表 A-27 所示可选参数 Occurrence 的参数说明，默认返回第一次出现的位置。

表 A-27

参 数	说 明
Occurrence.Firs	检索第 1 次位置

Occurrence.Last	检索最后 1 次的位置
Occurrence.All	检索所有的位置

22. Text.Clean

函数 : Text.Clean

语法 : Text.Clean(**text** as nullable text) as nullable text

说明 : 返回删除非打印字符后的字符串，例如：

= Text.Clean("ABCD# (lf) EFG") //输出 "ABCDEFG"

23. Text.Lower (Text.Upper) (Text.Proper)

函数 : Text.Lower

语法 : Text.Lower(**text** as nullable text, *optional* **culture** as nullable text) as nullable text

说明 : 将所有字母转换为小写的字符串，例如：

= Text.Lower("Hello China") 输出 "hello china"

Text.Upper : 将所有字母转换为大写的字符串，例如：

= Text.Upper("Hello China") 输出 "HELLO CHINA"

Text.Proper : 将分隔的单词转换为首字母大写，例如：

= Text.Proper("HELLO CHINA") 输出 "Hello China"

24. Text.PadStart (Text.PadEnd)

函数 : Text.PadStart

语法 : Text.PadStart(**text** as nullable text, **count** as number, *optional* **character** as nullable text) as nullable text

说明 : 通过在文本值 text 的开头插入空格，返回填充到指定长度 count 的 text 字符串，可选字符串 character 可用于指定用于填充的字符，不指定时默认以空格填充，例如：

= Text.PadStart("ABCDE", 10) 输出 " ABCDE"
= Text.PadStart("ABCDE", 10, "#") 输出 "#####ABCDE"

Text.PadEnd 通过在文本值 text 的末尾插入空格，例如：

= Text.PadEnd("ABCDE", 10) 输出 " ABCDE "
= Text.PadEnd("ABCDE", 10, "^") 输出 " ABCDE^^^^^"

25. Text.Repeat

函数 : Text.Repeat

语法：Text.Repeat(**text** as nullable text, **count** as number) as nullable text

说明：返回由 text 重复 count 次组成的字符串，例如：

= Text.Repeat("A", 5)	输出	"AAAAA"
= Text.Repeat("Abc ", 5)	输出	"Abc Abc Abc Abc Abc "

26. Text.Split

函数：Text.Split

语法：Text.Split(**text** as nullable text, **separator** as text) as list

说明：返回按指定字符 separator 拆分字符串 text 而得到的文本值 list 列表，例如：

= Text.Split("A,B,C,D,E,F,G", ",") 输出 {"A", "B", "C", "D", "E", "F", "G"}

27. Text.SplitAny

函数：Text.SplitAny

语法：Text.SplitAny(**text** as nullable text, **separators** as text) as list

说明：返回按指定字符 separatorsk 中的任意字符拆分字符串 text 而得到的文本值 list 列表，例如：

= Text.SplitAny("A,B;C,D;E,F;G", ";,") 输出 {"A", "B", "C", "D", "E", "F", "G"}

28. Text.Trim

函数：Text.Trim

语法：Text.Trim(**text** as nullable text, *optional trim* as any) as nullable text

说明：返回删除文本值 text 两端的空格或指定字符的字符串，未指定 trim 时，则删除空格，指定 trim 时，则删除两端指定的字符；不删除中间多余的空格，例如：

= Text.Trim(" A B C D e F G ")	输出	"A B C D e F G"
= Text.Trim("aaabABCDaaab", {"a", "b"})	输出	"ABCD"

29. Text.TrimStart

函数：Text.TrimStart

语法：Text.TrimStart (**text** as nullable text, *optional trim* as any) as nullable text

说明：返回删除文本值 text 前导的空格或指定字符的字符串，未指定 trim 时，则删除空格，指定 trim 时，则删除前导指定的字符，例如：

```
= Text.TrimStart("  A B C D e F G  ")
           输出      "A B C D e F G  "
= Text.TrimStart("@@@@@A B C D e F G@@@@@", "@")
           输出      "A B C D e F G@@@@@"
```

30. Text.TrimEnd

函数：Text.TrimEnd

语法：Text.TrimEnd (**text** as nullable text, *optional* **trim** as any) as nullable text

说明：返回删除文本值 text 后置的空格或指定字符的字符串，未指定 trim 时，则删除空格，指定 trim 时，则删除后置指定的字符，例如：

```
= Text.TrimEnd("  A B C D e F G  ")
           输出      "  A B C D e F G"
= Text.TrimEnd("@@@@@A B C D e F G@@@@@", "@")
           输出      "@@@@@A B C D e F G"
```

31. Text.Combine

函数：Text.Combine

语法：Text.Combine(**texts** as list, *optional* **separator** as nullable text) as nullable text

说明：将字符 list 列表中的所有字符串合并成一个字符串，可以指定合并后的分隔符 separator。

```
= Text.Combine({"学习", "Power", "Query"})           输出      "学习 PowerQuery"
= Text.Combine({"1234", "5678", "9012"}, "-")         输出      "1234-5678-9012"
```

A.3 Number 类函数

Number 类函数说明如表 A-28 所示。

表 A-28

类	分 类	函数名	概 述
1	常量	Number.NaN	代表 0/0，返回 NaN
2	常量	Number.NegativeInfinity	代码 -1/0，返回负无穷大
3	常量	Number.PositiveInfinity	代码 1/0，返回正无穷大
4	常量	Number.Epsilon	4.94065645841247E-324
5	常量	Number.PI	圆周率 3.1415926535897931
6	随机数	Number.Random	返回介于 0~1 的小数
7	随机数	Number.RandomBetween	返回指定值之间的小数

Powery Query：用 Excel 玩转商业智能数据处理

8	判断	Number.IsNaN	判断是否是 0/0
9	判断	Number.IsEven	判断是否是偶数
10	判断	Number.IsOdd	判断是否是奇数
11	转换	Number.From	从可转换为值的文本转换为数字

续表

类	分 类	函数名	概 述
12	转换	Number.FromText	从文本数值转换为数字
13	转换	Number.ToText	从数字转换到指定的文本样式
14	舍入	Number.Round	舍入到指定位数
15	舍入	Number.RoundUp	向上舍入到指定位数
16	舍入	Number.RoundDown	向下舍入到指定位数
17	舍入	Number.RoundTowardZero	向接近 0 的方向舍入
18	舍入	Number.RoundAwayFromZero	向远离 0 的方向舍入
19	计算	Number.Abs	返回绝对值
20	计算	Number.Sign	返回正负数方向
21	计算	Number.IntegerDivide	返回两数相除所得结果的整数部分
22	计算	Number.Mod	返回两数相除所得结果的余数部分
23	计算	Number.Power	返回底数的指数结果
24	计算	Number.Sqrt	返回数字平方根
25	计算	Number.Factorial	返回数字的阶乘
26	计算	Number.Combinations	返回组合数
27	计算	Number.Permutations	返回排列数
28	三角函数	Number.Acos	反余弦
29	三角函数	Number.Asin	反正弦
30	三角函数	Number.Atan	反正切
31	三角函数	Number.Atan2	返回两个数相除的反正切
32	三角函数	Number.Cos	余弦
33	三角函数	Number.Cosh	双曲余弦
34	三角函数	Number.Sin	正弦
35	三角函数	Number.Sinh	双曲正弦
36	三角函数	Number.Tan	正切
37	三角函数	Number.Tanh	双曲正切
38	其他函数	Number.E	自然常数 2.7182818284590451
39	其他函数	Number.Exp	返回计算 e 的幂结果
40	其他函数	Number.Ln	返回数字的自然对数

41	其他函数	Number.Log	返回指定底数的数字对数
42	其他函数	Number.Log10	返回以 10 为底数的数字对数

1. Number.NaN

函数：Number.NaN

语法：Number.NaN

说明：无参数，代表 0/0，返回 NaN，例如：

= Number.NaN	输出	NaN
--------------	----	-----

2. Number.NegativeInfinity

函数：Number.NegativeInfinity

语法：Number.NegativeInfinity

说明：无参数，返回负无穷大，例如：

= Number.NegativeInfinity	输出	负无穷大
---------------------------	----	------

3. Number.PositiveInfinity

函数：Number.PositiveInfinity

语法：Number.PositiveInfinity

说明：无参数，返回正无穷大，例如：

= Number.PositiveInfinity	输出	正无穷大
---------------------------	----	------

4. Number.Epsilon

函数：Number.Epsilon

语法：Number.Epsilon

说明：无参数，返回常量科学计数 4.94065645841247E-324，例如：

= Number.Epsilon	输出	4.94065645841247E-324
------------------	----	-----------------------

5. Number.PI

函数：Number.PI

语法：Number.PI

说明：无参数，返回圆周率 3.1415926535897931，例如：

= Number.PI	输出	3.1415926535897931
-------------	----	--------------------

6. Number.Random

函数 : Number.Random

语法 : Number.Random()

说明 : 无参数, 返回介于 0~1 的随机小数, 例如 :

= Number.Random()	输出	0.68750802040449721
-------------------	----	---------------------

7. Number.RandomBetween

函数 : Number.RandomBetween

语法 : Number.RandomBetween(**bottom** as number, **top** as number) as number

说明 : 返回介于 bottom 到 top 的随机小数, 例如 :

= Number.RandomBetween(1, 5)	输出	2.0608574957870216
------------------------------	----	--------------------

8. Number.IsNaN

函数 : Number.IsNaN

语法 : Number.IsNaN(**number** as number) as logical

说明 : 判断值是否是 0/0 的 NaN 结果, 如果是则返回 TRUE, 否则返回 FALSE, 例如 :

= Number.IsNaN(0/0)	输出	TRUE
---------------------	----	------

9. Number.IsEven

函数 : Number.IsEven

语法 : Number.IsEven(**number** as number) as logical

说明 : 判断值是否是偶数, 如果是则返回 TRUE, 否则返回 FALSE, 例如 :

= Number.IsEven(510809100)	输出	TRUE
----------------------------	----	------

10. Number.IsOdd

函数 : Number.IsOdd

语法 : Number.IsOdd(**number** as number) as logical

说明 : 判断值是否是奇数, 如果是则返回 TRUE, 否则返回 FALSE, 例如 :

= Number.IsOdd(510809100)	输出	FALSE
---------------------------	----	-------

11. Number.From

函数 : Number.From

语法 : Number.From(**value** as any, *optional* **culture** as nullable text) as nullable number

说明：从给定的 value 返回数字 number。culture 是一个语言选项，例如：

= Number.From("510809100")	输出	510809100
= Number.From(#date(2016,10,1))	输出	42644

culture 常用的参数“en-us”表示英语，“zh-cn”表示中文，省略时以安装软件的地区语言为参考，如 6-29 所示

表 A-29

参 数	说 明
文本数字	返回数字
数字	返回数字
TRUE	返回 1
FALSE	返回 0
日期时间	返回序列值
科学记数	返回数字

12. Number.FromText

函数：Number.FromText

语法：Number.FromText(**text** as text, *optional* **culture** as nullable text) as nullable number

说明：从给定的文本数值 text 获得数字 number。culture 是一个语言选项，例如：

= Number.FromText("510809100")	输出	510809100
--------------------------------	----	-----------

culture 常用的参数“en-us”表示英语，“zh-cn”表示中文，省略时以安装的地区语言为参考。

13. Number.ToText

函数：Number.ToText

语法：Number.ToText(**number** as nullable number, *optional* **format** as nullable text, *optional* **culture** as nullable text) as nullable text

说明：根据 format 所指定的格式将数字 number 格式转换为文本值，例如：

= Number.ToText(12345.54321, "N")	输出	"12345.54"
= Number.ToText(Number.Random(), "0.00%")	输出	"64.93%"

culture 常用的参数“en-us”表示英语，“zh-cn”表示中文，省略时以安装的地区语言为参考。

format 可以是一个文本样式的数字格式，也可以是以下代码，如表 A-30 所示。

表 A-30

参 数	说 明
"D"或"d"	(十进制) 只支持整数
"E"或"e"	(指数) 指数记数化
"F"或"f"	(固定点) 两位小数
"G"或"g"	(常规) 最简洁的数字样式
"N"或"n"	(数字) 带千位分隔符的两位小数
"P"或"p"	(百分比) 以百分比样式显示

续表

参 数	说 明
"R"或"r"	(往返) 可往返转换同一数字的文本值
"X"或"x"	(十六进制) 十六进制文本值

14. Number.Round

函数：Number.Round

语法：Number.Round(**number** as nullable number, *optional* **digits** as nullable number, *optional* **roundingMode** as nullable RoundingMode.Type) as nullable number

说明：将 number 舍入到最接近的数字。其他参数全部省略时，舍入为最接近的整数，如果指定了 digits，则舍入到相应位数的小数，可选的 randingMode 参数指定可能要四舍五入的数字之间存在联系时的舍入方向，例如：

= Number.Round(12345.34500,2)	输出	12345.34
= Number.Round(12345.34501,2)	输出	12345.35
= Number.Round(12345.345,2)	输出	12345.34
= Number.Round(-12345.345,2,RoundingMode.Up)	输出	-12345.34
= Number.Round(-12345.345,2,RoundingMode.Down)	输出	-12345.35
= Number.Round(12345.345,2,RoundingMode.AwayFromZero)	输出	12345.35
= Number.Round(12345.345,2,RoundingMode.TowardZero)	输出	12345.34
= Number.Round(12345.345,2,RoundingMode.ToEven)	输出	12345.3

排除四舍六入的规则外，roundingMode 参数的舍入规则如表 A-31 所示。

表 A-31

参 数	规 则	数 值
RoundingMode.Up	向上舍入	0
RoundingMode.Down	向下舍入	1
RoundingMode.AwayFromZero	向远离 0 的方向舍入	2

RoundingMode.TowardZero	向靠近 0 的方向舍入	3
RoundingMode.ToEven	舍入到偶数	4

15. Number.RoundUp

函数 : Number.RoundUp

语法 : Number.RoundUp(**number** as nullable number, *optional* **digits** as nullable number) as nullable number

说明 : 将 number 向上舍入到最接近的结果, digits 指定要舍入的小数位数, 例如 :

= Number.RoundUp(12345.345,2)	输出	12345.35
-------------------------------	----	----------

16. Number.RoundDown

函数 : Number.RoundDown

语法 : Number.RoundDown(**number** as nullable number, *optional* **digits** as nullable number) as nullable number

说明 : 将 number 向下舍入到最接近的结果, digits 指定要舍入的小数位数, 例如 :

= Number.RoundDown(12345.345,2)	输出	12345.34
---------------------------------	----	----------

17. Number.RoundTowardZero

函数 : Number.RoundTowardZero

语法 : Number.RoundTowardZero(**number** as nullable number, *optional* **digits** as nullable number) as nullable number

说明 : 将 number 向靠近 0 的方向舍入到最接近的结果, digits 指定要舍入的小数位数, 例如 :

= Number.RoundTowardZero(12345.345,2)	输出	12345.34
= Number.RoundTowardZero(-12345.345,2)	输出	-12345.34

18. Number.RoundAwayFromZero

函数 : Number.RoundAwayFromZero

语法 : Number.RoundAwayFromZero(**number** as nullable number, *optional* **digits** as nullable number) as nullable number

说明 : 将 number 向远离 0 的方向舍入到最接近的结果, digits 指定要舍入的小数位数

= Number.RoundAwayFromZero(12345.345,2)	输出	12345.35
= Number.RoundAwayFromZero(-12345.345,2)	输出	-12345.35

19. Number.Abs

函数：Number.Abs

语法：Number.Abs(**number** as nullable number) as nullable number

说明：返回 number 的绝对值，例如：

= Number.Abs(18)	输出	18
= Number.Abs(-18)	输出	18

20. Number.Sign

函数：Number.Sign

语法：Number.Sign(**number** as nullable number) as nullable number

说明：number 如果是正数，则返回 1；如果是负数，则返回-1；如果是 0，则返回 0，例如：

= Number.Sign(18)	输出	1
= Number.Sign(-18)	输出	-1

21 Number.IntegerDivide

函数：Number.IntegerDivide

语法：Number.IntegerDivide(**number1** as nullable number, **number2** as nullable number, optional **precision** as nullable Precision.Type) as nullable number

说明：返回 number1 除以 number2 所得结果的整数部分，precision 确定了计算精度，例如：
其中参数说明如表 A-32 所示。

= Number.IntegerDivide(10,3)	输出	3
------------------------------	----	---

表 A-32

参 数	说 明
Precision.Double	双精度
Precision.Decimal	十进制

22. Number.Mod

函数：Number.Mod

语法：Number.Mod(**number** as nullable number, **divisor** as nullable number, *optional* **precision** as nullable Precision.Type) as nullable number

说明：返回 number 除以 divisor 所得结果的余数部分，precision 确定了计算精度，例如：

= Number.Mod(10,3)	输出	1
--------------------	----	---

23. Number.Power

函数 : Number.Power

语法 : Number.Power(**number** as nullable number, **power** as nullable number) as nullable number

说明 : 返回 number 计算 power 次幂所得的结果，例如：

= Number.Power(10,3)	输出	1000
----------------------	----	------

24. Number.Sqrt

函数 : Number.Sqrt

语法 : Number.Sqrt(**number** as nullable number) as nullable number

说明 : 返回 number 的平方根，如果 number 是负值，则返回 Number.NaN，例如：

= Number.Sqrt(81)	输出	9
-------------------	----	---

25. Number.Factorial

函数 : Number.Factorial

语法 : Number.Factorial(**number** as nullable number) as nullable number

说明 : 返回数 number 的阶乘，例如：

= Number.Factorial(7)	输出	5040
-----------------------	----	------

26. Number.Combinations

函数 : Number.Combinations

语法 : Number.Combinations(**setSize** as nullable number, **combinationSize** as nullable number) as nullable number

说明 : 从项目列表 setSize 返回具有指定组合大小的 combinationSize 的唯一组合数，例如：

= Number.Combinations(6,3)	输出	20
----------------------------	----	----

27. Number.Permutations

函数 : Number.Permutations

语法 : Number.Permutations(**setSize** as nullable number, **permutationSize** as nullable number) as nullable number

说明 : 从项目列表 setSize 返回具有指定组合大小的 permutationSize 的唯一排列数，例如：

= Number.Permutations(6, 3)	输出	120
-----------------------------	----	-----

28. 三角函数

三角函数包括以下几种，如表 A-33 所示。

表 A-33

函数名	说 明
Number.Acos	反余弦
Number.Asin	反正弦
Number.Atan	反正切
Number.Atan2	返回两个数相除的反正切
Number.Cos	余弦
Number.Cosh	双曲余弦
Number.Sin	正弦
Number.Sinh	双曲正弦
Number.Tan	正切
Number.Tanh	双曲正切

29. 其他函数

Number 类的其他函数包括以下几种，如表 A-34 所示。

表 A-34

函数名	说 明
Number.E	自然常数 2.7182818284590451
Number.Exp	返回计算 e 的幂结果
Number.Ln	返回数字的自然对数
Number.Log	返回指定底数的数字对数
Number.Log10	返回以 10 为底数的数字对数

A.4 Time 类函数

Time 类函数说明如表 A-35 所示。

表 A-35

类	分 类	函 数 名	概 述
1	计算	Time.Hour	返回时间的小时数
2	计算	Time.Minute	返回时间的分钟数
3	计算	Time.Second	返回时间的秒钟数

4	智能时间	Time.ToRecord	返回时间中的时、分、秒的 Record 记录
5	智能时间	Time.StartOfHour	返回当时小时的起始值
6	智能时间	Time.EndOfHour	返回当时小时的结束值
7	转换	Time.From	将给定的 value 返回时间
8	转换	Time.FromText	从文本表示的 Text 返回时间
9	转换	Time.ToText	返回指定文本样式的日期

1. 创建时间、日期、日期时间、日期时间时区

创建时间、日期、日期时间、日期时间时区。代码示例如下所示：

#time(10,24,36)	输出	10:24:36
#date(2016,10,1)	输出	2016/10/1
#datetime(2016,10,1,10,24,36)	输出	2016/10/1 10:24:36
#datetimezone(2016,10,1,18,26,34,8,0)	输出	2016/10/01 18:26:34 +08:00
#duration(0,10,0,0)	输出	0.10:00:00

2. Time.Hour

函数：Time.Hour

语法：Time.Hour(**dateTime** as any) as nullable number

说明：从提供的 time、datetime、datetimezone 值返回小时数，例如：

= Time.Hour(#time(10,26,34))	输出	10
= Time.Hour(#datetime(2016,10,1,14,45,42)))	输出	14
= Time.Hour(#datetimezone(2016,10,1,18,7,15,8,0)))	输出	18

3. Time.Minute

函数：Time.Minute

语法：Time.Minute(**dateTime** as any) as nullable number

说明：从提供的 time、datetime、datetimezone 值返回分钟部分数，例如：

= Time.Minute(#time(10,26,34))	输出	26
= Time.Minute(#datetime(2016,10,1,14,45,42)))	输出	45
= Time.Minute(#datetimezone(2016,10,1,18,7,15,8,0)))	输出	7

4. Time.Second

函数：Time.Second

语法：Time.Second(**dateTime** as any) as nullable number

说明：从提供的 time、datetime、datetimezone 值的秒钟部分。

= Time.Second(#time(10,26,34))	输出	34
= Time.Second(#datetime(2016,10,1,14,45,42))	输出	42
= Time.Second(#datetimezone(2016,10,1,18,7,15,8,0))	输出	15

5. Time.ToRecord

函数：Time.ToRecord

语法：Time.ToTecord(**time** as nullable time) as record

说明：返回包含给定时间值 time 的时、分、秒组成的 record 记录，例如：

= Time.ToRecord(#time(12, 34, 56))	输出	[Hour=12, Minute=34, Second=56]
------------------------------------	----	---------------------------------

6. Time.StartOfHour

函数：Time.StartOfHour

语法：Time.StartOfHour (**dateTime** as any) as nullable any

说明：从提供的 time、datetime、datetimezone 值返回当前小时的起始值，例如：

= Time.StartOfHour(#time(10,26,34))	输出	10:00:00
= Time.StartOfHour(#datetime(2016,10,1,14,45,42))	输出	2016/10/01 14:00:00
= Time.StartOfHour(#datetimezone(2016,10,1,18,7,15,8,0))	输出	2016/10/01 18:00:00+08:00

7. Time.EndOfHour

函数：Time.EndOfHour

语法：Time.EndOfHour (**dateTime** as any) as nullable any

说明：从提供的 time、datetime、datetimezone 值返回当前小时的结束值，例如：

= Time.EndOfHour(#time(10,26,34))	输出	10:59:59.9999999
= Time.EndOfHour(#datetime(2016,10,1,14,45,42))	输出	2016-10-01T14:59:59.9999999
= Time.EndOfHour(#datetimezone(2016,10,1,18,7,15,8,0))	输出	2016-10-01T18:59:59.9999999+08:00

8. Time.From

函数：Time.From

语法：Time.From(**value** as any, *optional culture* as nullable text) as nullable time

说明：从给定的 value 返回 time 值，例如：

= Time.From(0.45728214)	输出	10:58:29.1768960
= Time.From("10:23:55")	输出	10:23:55

```
= Time.From(#datetime(2016,10,1,21,9,12))      输出      21:09:12
= Time.From(#datetimezone(2016,10,1,21,12,55,8,0)) 输出      21:12:55
```

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

Value 支持的参数如表 A-36 所示。

图 A-36

参 数	说 明
text	文本形式的 time 值
datetime	value 的时间部分
datetimezone	value 的时间部分
number	介于 0 和 1 之间的小数

9. Time.FromText

函数：Time.FromText

语法：Time.FromText(**text** as nullable text, *optional* **culture** as nullable text) as nullable time

说明：按照 ISO8601 格式标准，从文本形式的 text 返回 time 值，例如：

```
= Time.FromText("12")      输出      12:00:00
= Time.FromText("1234")    输出      12:34:00
= Time.FromText("123456")  输出      12:34:56
= Time.FromText("4:34:56pm") 输出      16:34:56
= Time.FromText("2:34:56 下午","zh-cn") 输出      14:34:56
= Time.FromText("2 时 3 分 58 秒","zh-cn") 输出      2:03:58
= Time.FromText("下午 2 时 3 分 58 秒","zh-cn") 输出      2:03:58
= Time.FromText("3 时 12 秒","zh-cn") 输出      3:00:12
```

culture 常用的函数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

10. Time.ToText

函数：Time.ToText

语法：Time.ToText(**time** as nullable time, *optional* **format** as nullable text, *optional* culture as nullable text) as nullable text

说明：返回 time 时间值的文本表示形式 time，此函数可以采用可选参数 format 设置相关格式，例如：

```
= Time.ToText(#time(14,23,05))      输出      "14:23"
= Time.ToText(#time(14,23,05),"h:mm:ss") 输出      "2:23:05"
= Time.ToText(#time(14,23,05),"hh:mm:ss") 输出      "14:23:05"
= Time.ToText(#time(14,23,05),"h 时 m 分 s 秒") 输出      "2 时 23 分 5 秒"
```

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

A.5 Date 类函数

Date 类函数说明如表 A-37 所示。

表 A-37

类	分 类	函 数 名	概 述
1	计算	Date.Year	返回日期的年份
2	计算	Date.Month	返回日期的月份
3	计算	Date.MonthName	返回日期的中英文月份
4	计算	Date.Day	返回日期的日部分
5	计算	Date.ToRecord	返回日期中年、月、日组成的 Record 记录
6	计算	Date.QuarterOfYear	返回日期的季度数

续表

类	分 类	函 数 名	概 述
7	计算	Date.DayOfWeekName	返回日期的中英文星期名称
8	计算	Date.DayOfWeek	返回日期是一周中的第几天
9	计算	Date.DaysInMonth	返回日期当月的总天数
10	计算	Date.DayOfYear	返回日期当天是一年中的第几天
11	计算	Date.WeekOfMonth	返回日期属于当月的第几周
12	计算	Date.WeekOfYear	返回日期属于当年的第几周
13	日期增减	Date.AddDays	返回若干天数前后的新日期
14	日期增减	Date.AddMonths	返回若干个月前后的新日期
15	日期增减	Date.AddQuarters	返回若干季度前后的新日期
16	日期增减	Date.AddWeeks	返回若干周前后的新日期
17	日期增减	Date.AddYears	返回若干年前后的新日期
18	智能日期	Date.EndOfDay	返回日期当天的结束值
19	智能日期	Date.EndOfWeek	返回日期当周的结束值
20	智能日期	Date.EndOfMonth	返回日期当月的结束值
21	智能日期	Date.EndOfQuarter	返回日期当季的结束值
22	智能日期	Date.EndOfYear	返回日期当年的结束值
23	智能日期	Date.StartOfDay	返回日期当天的起始值
24	智能日期	Date.StartOfWeek	返回日期当周的起始值
25	智能日期	Date.StartOfMonth	返回日期当月的起始值

26	智能日期	Date.StartOfQuarter	返回日期当季的起始值
27	智能日期	Date.StartOfYear	返回日期当年的起始值
28	智能日期	Date.IsInCurrentYear	判断日期是否在系统日期的当年
29	智能日期	Date.IsInCurrentMonth	判断日期是否在系统日期的当月
30	智能日期	Date.IsInCurrentQuarter	判断日期是否在系统日期的当季
31	智能日期	Date.IsInCurrentWeek	判断日期是否在系统日期的当周
32	智能日期	Date.IsInNextYear	判断日期是否在系统日期的下一年
33	智能日期	Date.IsInNextMonth	判断日期是否在系统日期的下一月份
34	智能日期	Date.IsInNextQuarter	判断日期是否在系统日期的下一季度
35	智能日期	Date.IsInNextWeek	判断日期是否在系统日期的下一周
36	智能日期	Date.IsInPreviousYear	判断日期是否在系统日期的上一年
37	智能日期	Date.IsInPreviousMonth	判断日期是否在系统日期的上一月份
38	智能日期	Date.IsInPreviousQuarter	判断日期是否在系统日期的上一季度
39	判断	Date.IsInPreviousWeek	判断日期是否在系统日期的上一周
40	判断	Date.IsInYearToDate	判断日期是否在当年年份起始日期至当天的日期范围内

续表

类	分 类	函 数 名	概 述
41	判断	Date.IsLeapYear	判断日期是否在闰年
42	转换	Date.From	从给定的 value 值返回日期
43	转换	Date.FromText	根据从文本表示的 Text 返回日期
44	转换	Date.ToText	返回指定文本样式的日期

1. Date.Year

函数：Date.Year

语法：Date.Year(**datetime** as any) as nullable number

说明：返回提供的 datetime 值的年份，例如：

= Date.Year(#date(2010,2,14))	输出	2010
= Date.Year(#datetime(2016,10,2,14,45,42))	输出	2016
= Date.Year(#datetimezone(2013,7,25,18,7,15,8,0))	输出	2013

2. Date.Month

函数：Date.Month

语法：Date.Month(**datetime** as any) as nullable number

说明：返回提供的 datetime 值的月份，例如：

= Date.Month(#date(2010,2,14))	输出	2
--------------------------------	----	---

= Date.Month(#datetime(2016,10,2,14,45,42))	输出	10
= Date.Month(#datetimezone(2013,7,25,18,7,15,8,0))	输出	7

3. Date.MonthName

函数 : Date.MonthName

语法 : Date.MonthName(**date** as any, *optional* **culture** as nullable text) as nullable text

说明 : 返回日期的月份中英文名称, 例如 :

= Date.MonthName(#date(2010,2,14),"en-us")	输出	February
= Date.MonthName(#datetime(2016,10,2,14,45,42))	输出	十月
= Date.MonthName(#datetimezone(2013,7,25,18,7,15,8,0),"zh-cn")	输出	七月

culture 常用的系数"en-us"表示英语, "zh-cn"表示中文, 省略时以安装的地区语言为参考。

4. Date.Day

函数 : Date.Day

语法 : Date.Day(**datetime** as any) as nullable number

说明 : 返回提供的 datetime 值的日部分, 例如 :

= Date.Day(#date(2010,2,14))	输出	14
= Date.Day(#datetime(2016,10,2,14,45,42))	输出	2
= Date.Day(#datetimezone(2013,7,25,18,7,15,8,0))	输出	25

5. Date.ToRecord

函数 : Date.ToRecord

语法 : Date.ToRecord(**date** as date) as record

说明 : 返回包含给定时间值 date 的年、月、日组成的 record 记录, 例如 :

= Date.ToRecord(#date(2016, 10, 1))
输出 [Year=2016, Month=10, Day=1]

6. Date.QuarterOfYear

函数 : Date.QuarterOfYear

语法 : Date.QuarterOfYear(**datetime** as any) as nullable number

说明 : 返回日期在一年中的季度序数, 结果介于 1~4, 例如 :

= Date.QuarterOfYear(#date(2010,2,14),"en-us")	输出	1
= Date.QuarterOfYear(#datetime(2016,10,2,14,45,42))	输出	4
= Date.QuarterOfYear(#datetimezone(2013,7,25,18,7,15,8,0))	输出	3

7. Date.DayOfWeekName

函数 : Date.DayOfWeekName

语法 : Date.DayOfWeekName(**date** as any, *optional* **culture** as nullable text) as nullable text

说明 : 返回日期的中英文星期名称, 例如 :

```
= Date.DayOfWeekName(#date(2010,2,14),"en-us")      输出    Sunday
= Date.DayOfWeekName(#datetime(2016,10,2,14,45,42),"zh-cn")
                                                    输出    星期日
= Date.DayOfWeekName(#datetimezone(2013,7,25,18,7,15,8,0))
                                                    输出    星期四
```

culture 常用的参数"en-us"表示英语, "zh-cn"表示中文, 省略时以安装的地区语言为参考。

8. Date.DayOfWeek

函数 : Date.DayOfWeek

语法 : Date.DayOfWeek(**datetime** as any, *optional* **firstDayOfWeek** as nullable Day.Type) as any

说明 : 返回介于 0~6 的数值, 该值表示日期是星期几, 此函数可选参数 firstDayOfWeek 可以指定哪一天是一周的第一天, 第一天的默认值是 Day.Sunday。中国人的习惯是周一显示为 1, 周日显示为 7。如下面的代码所示, 第一个示例表示中国人对星期的使用习惯。

firstDayOfWeek 参数用于设置每周的第一天是哪一天, 如表 A-38 所示。

```
= Date.DayOfWeek(#date(2010,2,14),1)+1      输出    7
= Date.DayOfWeek(#datetime(2016,10,1,14,45,42),0)      输出    6
= Date.DayOfWeek(#datetimezone(2013,7,14,18,7,15,8,0))  输出    6
```

表 A-38

参 数	含 义	数 值
Day.Sunday	星期日	0
Day.Monday	星期一	1
Day.Tuesday	星期二	2
Day.Wednesday	星期三	3
Day.Thursday	星期四	4
Day.Friday	星期五	5
Day.Saturday	星期六	6

9. Date.DaysInMonth

函数 : Date.DaysInMonth

语法 : Date.DaysInMonth(**datetime** as any) as number

说明：返回指定日期当月的总天数，例如：

= Date.DaysInMonth(#date(2010,2,14))	输出	28
= Date.DaysInMonth(#datetime(2016,10,1,14,45,42))	输出	31
= Date.DaysInMonth(#datetimezone(2013,7,14,18,7,15,8,0))	输出	31

10. Date.DayOfYear

函数：Date.DayOfYear

语法：Date. DayOfYear(**datetime** as any) as nullable number

说明：返回指定日期是一年中的第几天，例如：

= Date.DayOfYear(#date(2010,2,14))	输出	45
= Date.DayOfYear(#datetime(2016,10,1,14,45,42))	输出	275
= Date.DayOfYear(#datetimezone(2013,7,14,18,7,15,8,0))	输出	195

11. Date.WeekOfMonth

函数：Date.WeekOfMonth

语法：Date. WeekOfMonth(**datetime** as any, *optional firstDayOfWeek* as nullable Day.Type) as nullable number

说明：返回指定日期是当前月的第几周，可以指定每周的第一天是星期几，例如：

= Date.WeekOfMonth(#date(2010,2,14),1)	输出	2
= Date.WeekOfMonth(#datetime(2016,10,1,14,45,42),0)	输出	1
= Date.WeekOfMonth(#datetimezone(2013,7,14,18,7,15,8,0))	输出	2

firstDayOfWeek 参数用于设置每周的第一天是哪一天。

12. Date.WeekOfYear

函数：Date.WeekOfYear

语法：Date. WeekOfYear(**datetime** as any, *optional firstDayOfWeek* as nullable Day.Type) as nullable number

说明：返回指定日期是当年的第几周。可以指定每周的第一天是星期几，例如：

= Date.WeekOfYear(#date(2010,2,14),1)	输出	7
= Date.WeekOfYear(#datetime(2016,10,1,14,45,42),0)	输出	40
= Date.WeekOfYear(#datetimezone(2013,7,14,18,7,15,8,0))	输出	28

firstDayOfWeek 参数设置每周的第一天是哪一天。

13. Date.AddDays

函数：Date.AddDays

语法： Date.AddDays(**datetime** as any, **numberOfDays** as number) as any

说明： 返回指定日期增减若干天后的新日期，例如：

```
= Date.AddDays(#date(2010,2,14),10)      输出      2010/02/24
= Date.AddDays(#datetime(2016,10,1,14,45,42),-50)
                                           输出      2016/08/12 14:45:42
= Date.AddDays(#datetimezone(2013,7,14,18,7,15,8,0),31)
                                           输出      2013/08/14 18:07:15 +08:00
```

14. Date.AddMonths

函数： Date.AddMonths

语法： Date.AddMonths(**datetime** as any, **numberOfMonths** as number) as any

说明： 返回指定日期增减若干月后的新日期，例如：

```
= Date.AddMonths(#date(2010,2,14),10)    输出      2010/12/14
= Date.AddMonths(#datetime(2016,10,1,14,45,42),-50)
                                           输出      2012/08/01 14:45:42
= Date.AddMonths(#datetimezone(2013,7,14,18,7,15,8,0),31)
                                           输出      2016/02/14 18:07:15 +08:00
```

15. Date.AddQuarters

函数： Date.AddQuarters

语法： Date.AddQuarters(**datetime** as any, **numberOfQuarters** as number) as any

说明： 返回指定日期增减若干季度后的新日期，例如

```
= Date.AddQuarters(#date(2010,2,14),10)   输出      2012/08/14
= Date.AddQuarters(#datetime(2016,10,1,14,45,42),-50)
                                           输出      2004/04/01 14:45:42
= Date.AddQuarters(#datetimezone(2013,7,14,18,7,15,8,0),31)
                                           输出      2021/04/14 18:07:15 +08:00
```

16. Date.AddWeeks

函数： Date.AddWeeks

语法： Date.AddWeeks(**datetime** as any, **numberOfWeeks** as number) as any

说明： 返回指定日期增减若干周后的新日期，例如：

```
= Date.AddWeeks(#date(2010,2,14),10)      输出      2010/04/25
= Date.AddWeeks(#datetime(2016,10,1,14,45,42),-50)
                                           输出      2015/10/17 14:45:42
= Date.AddWeeks(#datetimezone(2013,7,14,18,7,15,8,0),31)
                                           输出      2014/02/16 18:07:15 +08:00
```

17. Date.AddYears

函数 : Date.AddYears

语法 : Date.AddYears(**datetime** as any, **numberOfYears** as number) as any

说明 : 返回指定日期增减若干年后的新日期, 例如 :

```
= Date.AddYears(#date(2010,2,14),10)    输出    2020/02/14
= Date.AddYears(#datetime(2016,10,1,14,45,42),-50)
                                           输出    1966/10/01 14:45:42
= Date.AddYears(#datetimezone(2013,7,14,18,7,15,8,0),31)
                                           输出    2044/07/14 18:07:15 +08:00
```

18. Date.EndOfDay

函数 : Date.EndOfDay

语法 : Date.EndOfDay(**datetime** as any) as any

说明 : 返回指定日期当天的结束时间点, 例如 :

```
= Date.EndOfDay(#date(2010,2,14))    输出    2010/02/14
= Date.EndOfDay(#datetime(2016,10,1,14,45,42))
                                           输出    2016-10-01T23:59:59.9999999
= Date.EndOfDay(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013-07-14T23:59:59.9999999+08:00
```

19. Date.EndOfWeek

函数 : Date.EndOfWeek

语法 : Date.EndOfWeek(**datetime** as any) as any

说明 : 返回指定日期当前周的结束时间点, 例如 :

```
= Date.EndOfWeek(#date(2010,2,14))    输出    2010/02/14
= Date.EndOfWeek(#datetime(2016,10,1,14,45,42))
                                           输出    2016-10-02T23:59:59.9999999
= Date.EndOfWeek(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013-07-14T23:59:59.9999999+08:00
```

20. Date.EndOfMonth

函数 : Date.EndOfMonth

语法 : Date.EndOfMonth(**datetime** as any) as any

说明 : 返回指定日期当前月的结束时间点, 例如 :

```
= Date.EndOfMonth(#date(2010,2,14))    输出    2010/02/28
= Date.EndOfMonth(#datetime(2016,10,1,14,45,42))
```

```

      输出      2016-10-31T23:59:59.9999999
= Date.EndOfMonth(#datetimezone(2013,7,14,18,7,15,8,0))
      输出      2013-07-31T23:59:59.9999999+08:00

```

21. Date.EndOfQuarter

函数 : Date.EndOfQuarter

语法 : Date.EndOfQuarter(**datetime** as any) as any

说明 : 返回指定日期当前季度的结束时间点, 例如 :

```

= Date.EndOfQuarter(#date(2010,2,14))      输出      2010/03/31
= Date.EndOfQuarter(#datetime(2016,10,1,14,45,42))
      输出      2016-12-31T23:59:59.9999999
= Date.EndOfQuarter(#datetimezone(2013,7,14,18,7,15,8,0))
      输出      2013-09-30T23:59:59.9999999+08:00

```

22 Date.EndOfYear

函数 : Date.EndOfYear

语法 : Date.EndOfYear(**datetime** as any) as any

说明 : 返回指定日期当前年份的结束时间点, 例如 :

```

= Date.EndOfYear(#date(2010,2,14)) 输出      2010/12/31
= Date.EndOfYear(#datetime(2016,10,1,14,45,42))
      输出      2016-12-31T23:59:59.9999999
= Date.EndOfYear(#datetimezone(2013,7,14,18,7,15,8,0))
      输出      2013-12-31T23:59:59.9999999+08:00

```

23. Date.StartOfDay

函数 : Date.StartOfDay

语法 : Date.StartOfDay(**datetime** as any) as any

说明 : 返回指定日期当天的开始时间点, 例如 :

```

= Date.StartOfDay(#date(2010,2,14))      输出      2010/02/14
= Date.StartOfDay(#datetime(2016,10,1,14,45,42))
      输出      2016/10/01 0:00:00
= Date.StartOfDay(#datetimezone(2013,7,14,18,7,15,8,0))
      输出      2013/07/14 0:00:00 +08:00

```

24. Date.StartOfWeek

函数 : Date.StartOfWeek

语法 : Date.StartOfWeek(**datetime** as any) as any

说明：返回指定日期当前周的开始时间点，例如：

```
= Date.StartOfWeek(#date(2010,2,14))    输出    2010/02/08
= Date.StartOfWeek(#datetime(2016,10,1,14,45,42))
                                           输出    2016/09/26 0:00:00
= Date.StartOfWeek(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013/07/08 0:00:00 +08:00
```

25. Date.StartOfMonth

函数：Date.StartOfMonth

语法：Date.StartOfMonth(**datetime** as any) as any

说明：返回指定日期当前月的开始时间点，例如：

```
= Date.StartOfMonth(#date(2010,2,14))    输出    2010/02/01
= Date.StartOfMonth(#datetime(2016,10,1,14,45,42))
                                           输出    2016-10-01T0:00:00
= Date.StartOfMonth(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013-07-01T0:00:00+08:00
```

26. Date.StartOfQuarter

函数：Date.StartOfQuarter

语法：Date.StartOfQuarter(**datetime** as any) as any

说明：返回指定日期当前季度的开始时间点，例如：

```
= Date.StartOfQuarter(#date(2010,2,14))    输出    2010/01/01
= Date.StartOfQuarter(#datetime(2016,10,1,14,45,42))
                                           输出    2016/10/01 0:00:00
= Date.StartOfQuarter(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013/07/01 0:00:00 +08:00
```

27. Date.StartOfYear

函数：Date.StartOfYear

语法：Date.StartOfYear(**datetime** as any) as any

说明：返回指定日期当前年份的开始时间点，例如：

```
= Date.StartOfYear(#date(2010,2,14))    输出    2010/01/01
= Date.StartOfYear(#datetime(2016,10,1,14,45,42))
                                           输出    2016-01-01T0:00:00
= Date.StartOfYear(#datetimezone(2013,7,14,18,7,15,8,0))
                                           输出    2013-01-01T0:00:00+08:00
```

28. Date.IsInCurrentYear

函数： Date.IsInCurrentYear

语法： Date.IsInCurrentYear(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的当年，例如：

= Date.IsInCurrentYear(#date(2010,2,14))	输出 FALSE
= Date.IsInCurrentYear(#datetime(2016,10,1,14,45,42))	输出 TRUE
= Date.IsInCurrentYear(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

29. Date.IsInCurrentMonth

函数： Date.IsInCurrentMonth

语法： Date.IsInCurrentMonth(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的所在月份，例如：

= Date.IsInCurrentMonth(#date(2010,2,14))	输出 FALSE
= Date.IsInCurrentMonth(#datetime(2016,10,1,14,45,42))	输出 TRUE
= Date.IsInCurrentMonth(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

30. Date.IsInCurrentQuarter

函数： Date.IsInCurrentQuarter

语法： Date.IsInCurrentQuarter(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的所在季度，例如：

= Date.IsInCurrentQuarter(#date(2010,2,14))	输出 FALSE
= Date.IsInCurrentQuarter(#datetime(2016,10,1,14,45,42))	输出 TRUE
= Date.IsInCurrentQuarter(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

31. Date.IsInCurrentWeek

函数： Date.IsInCurrentWeek

语法： Date.IsInCurrentWeek(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的所在周，例如：

= Date.IsInCurrentWeek(#date(2010,2,14))	输出 FALSE
= Date.IsInCurrentWeek(#datetime(2016,10,1,14,45,42))	输出 TRUE
= Date.IsInCurrentWeek(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

32. Date.IsInNextYear

函数： Date.IsInNextYear

Power Query: 用 Excel 玩转商业智能数据处理

语法 : Date.IsInNextYear(**datetime** as any) as nullable logical

说明 : 判断指定日期是否在系统日期的下一年, 例如 :

= Date.IsInNextYear(#date(2010,2,14))	输出 FALSE
= Date.IsInNextYear(#datetime(2016,10,1,14,45,42))	输出 FALSE
= Date.IsInNextYear(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

33. Date.IsInNextMonth

函数 : Date.IsInNextMonth

语法 : Date.IsInNextMonth(**datetime** as any) as nullable logical

说明 : 判断指定日期是否在系统日期的下一月份, 例如 :

= Date.IsInNextMonth(#date(2010,2,14))	输出 FALSE
= Date.IsInNextMonth(#datetime(2016,10,1,14,45,42))	输出 FALSE
= Date.IsInNextMonth(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

34. Date.IsInNextQuarter

函数 : Date.IsInNextQuarter

语法 : Date.IsInNextQuarter(**datetime** as any) as nullable logical

说明 : 判断指定日期是否在系统日期的下一季度, 例如 :

= Date.IsInNextQuarter(#date(2010,2,14))	输出 FALSE
= Date.IsInNextQuarter(#date(2010,2,14))	输出 FALSE
= Date.IsInNextQuarter(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

35. Date.IsInNextWeek

函数 : Date.IsInNextWeek

语法 : Date.IsInNextWeek(**datetime** as any) as nullable logical

说明 : 判断指定日期是否在系统日期的下一周, 例如 :

= Date.IsInNextWeek(#date(2010,2,14))	输出 FALSE
= Date.IsInNextWeek(#datetime(2016,10,1,14,45,42))	输出 FALSE
= Date.IsInNextWeek(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

36. Date.IsInPreviousYear

函数 : Date.IsInPreviousYear

语法 : Date.IsInPreviousYear(**datetime** as any) as nullable logical

说明 : 判断指定日期是否在系统日期的上一年, 例如 :

= Date.IsInPreviousYear(#date(2010,2,14))	输出 FALSE
---	----------

```
= Date.IsInPreviousYear(#datetime(2016,10,1,14,45,42))      输出 FALSE
= Date.IsInPreviousYear(#datetimezone(2013,7,14,18,7,15,8,0)) 输出 FALSE
```

37. Date.IsInPreviousMonth

函数： Date.IsInPreviousMonth

语法： Date.IsInPreviousMonth(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的上一月份，例如：

```
= Date.IsInPreviousMonth(#date(2010,2,14))      输出 FALSE
= Date.IsInPreviousMonth(#datetime(2016,10,1,14,45,42)) 输出 FALSE
= Date.IsInPreviousMonth(#datetimezone(2013,7,14,18,7,15,8,0)) 输出 FALSE
```

38. Date.IsInPreviousQuarter

函数： Date.IsInPreviousQuarter

语法： Date.IsInPreviousQuarter(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的上一季度，例如：

```
= Date.IsInPreviousQuarter(#date(2010,2,14))      输出 FALSE
= Date.IsInPreviousQuarter(#datetime(2016,10,1,14,45,42)) 输出 FALSE
= Date.IsInPreviousQuarter(#datetimezone(2013,7,14,18,7,15,8,0)) 输出 FALSE
```

39. Date.IsInPreviousWeek

函数： Date.IsInPreviousWeek

语法： Date.IsInPreviousWeek(**datetime** as any) as nullable logical

说明： 判断指定日期是否在系统日期的上一周，例如：

```
= Date.IsInPreviousWeek(#date(2010,2,14))      输出 FALSE
= Date.IsInPreviousWeek(#datetime(2016,10,1,14,45,42)) 输出 FALSE
= Date.IsInPreviousWeek(#datetimezone(2013,7,14,18,7,15,8,0)) 输出 FALSE
```

40. Date.IsInYearToDate

函数： Date.IsInYearToDate

语法： Date.IsInYearToDate(**datetime** as any) as nullable logical

说明： 判断指定日期是否在当前年份起始日期至当天的日期范围内，例如：

```
= Date.IsInYearToDate(#date(2010,2,14))      输出 FALSE
= Date.IsInYearToDate(#datetime(2016,10,1,14,45,42)) 输出 TRUE
= Date.IsInYearToDate(#datetimezone(2013,7,14,18,7,15,8,0)) 输出 FALSE
```

41. Date.IsLeapYear

函数：Date.IsLeapYear
语法：Date.IsLeapYear(**datetime** as any) as nullable logical
说明：判断指定日期是否是闰年，例如：

= Date.IsLeapYear(#date(2010,2,14))	输出 FALSE
= Date.IsLeapYear(#date(2010,2,14))	输出 FALSE
= Date.IsLeapYear(#datetimezone(2013,7,14,18,7,15,8,0))	输出 FALSE

42. Date.From

函数：Date.From
语法：Date.From(**value** as any, *optional culture* as nullable text) as nullable date
说明：从给定的 value 值返回 date 值，例如：

= Date.From("2014-6-6")	输出	2014/06/06
= Date.From(#datetime(2016,10,1,14,45,42))	输出	2016/10/01
= Date.From(#datetimezone(2013,7,14,18,7,15,8,0))	输出	2013/07/14
= Date.From(43210)	输出	2018/04/20
= Date.From("4 月 19 日")	输出	2016/04/19
= Date.From("19 日")	输出	2016/01/19
= Date.From("6-19")	输出	2016/06/19
= Date.From("2016")	输出	2016/01/01
= Date.From("19 jun, 2017")	输出	2017/06/19

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考，具体分数说明如表 A-39 所示。

表 A-39

参 数	说 明
value	支持的数据类型
text	文本表现形式的 date 值
datetime	日期和时间
datetimezone	日期\时间\时区
number	等效于 value 表示日期的日期序列值

43. Date.FromText

函数：Date.FromText
语法：Date.FromText(**text** as nullbale text, *optional culture* as nullable text) as nullable date
说明：按照 ISO8601 格式标准，从文本形式 text 创建 date 日期。

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

= Date.FromText("2014-6-6")	输出	2014/06/06
= Date.FromText("20161001")	输出	2016/10/01
= Date.FromText("2013 年 4 月 19 日")	输出	2013/04/19
= Date.FromText("4 月 19 日")	输出	2016/04/19
= Date.FromText("19 日")	输出	2016/01/19
= Date.FromText("6-19")	输出	2016/06/19
= Date.FromText("2016")	输出	2016/01/01
= Date.FromText("19 jun,2017")	输出	2017/06/19
= Date.FromText("Aug1")	输出	2017/08/01

44. Date.ToText

函数：Date.ToText

语法：Date.ToText(**date** as nullable date, *optional* **format** as nullable text, *optional* culture as nullable text) as nullable text

说明：按 format 指定的格式返回日期文本格式，例如：

date = #date(2016,8,9)		
= Date.ToText(date,"yyyyMMdd")	输出	"20160809"
= Date.ToText(date,"yyyy 年 M 月 d 日")	输出	"2016 年 8 月 9 日"
= Date.ToText(date,"Yyyyy")	输出	"Y2016"
= Date.ToText(date,"d")	输出	"2016/8/9"
= Date.ToText(date,"M")	输出	"8 月 9 日"
= Date.ToText(date,"M","en-us")	输出	"August 9"
= Date.ToText(date,"MM")	输出	"08"
= Date.ToText(date,"MMM")	输出	"8 月"
= Date.ToText(date,"MMM","en-us")	输出	"Aug"
= Date.ToText(date,"MMMM")	输出	"八月"
= Date.ToText(date,"MMMM","en-us")	输出	"August"
= Date.ToText(date,"dd")	输出	"09"
= Date.ToText(date,"ddd")	输出	"周二"
= Date.ToText(date,"ddd","en-us")	输出	"Tue"
= Date.ToText(date,"dddd")	输出	"星期三"
= Date.ToText(date,"dddd","en-us")	输出	"Tuesday"
= Date.ToText(date,"yyyy 年 M 月 d 日")	输出	"公元 2016 年 10 月 10 日"

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

A.6 DateTime 类函数

DateTime 类函数说明如表 A-40 所示。

表 A-40

类	分 类	函 数 名	概 述
1	计算	DateTime.Date	返回日期时间的日期部分
2	计算	DateTime.Time	返回日期时间的时间部分
3	计算	DateTime.LocalNow	返回当前系统日期时间的动态值
4	计算	DateTime.FixedLocalNow	返回当前系统日期时间的固定值
5	计算	DateTime.ToRecord	返回日期的年、月、日、时、分、秒的 Recond 记录
6	计算	DateTime.FromFileTime	获得以 100 纳秒为间隔数字的日期
7	计算	DateTime.AddZone	设置日期时间的世界时区
8	转换	DateTime.From	从给定的 value 值返回日期时间
9	转换	DateTime.FromText	从文本表示的 Text 返回日期时间
10	转换	DateTime.ToText	返回指定文本格式的日期时间
11	智能时间	DateTime.IsInCurrentHour	判断时间是否是系统时间的当前小时
12	智能时间	DateTime.IsInCurrentMinute	判断时间是否在系统时间的当前分钟
13	智能时间	DateTime.IsInCurrentSecond	判断时间是否在系统时间的当前秒
14	智能时间	DateTime.IsInNextHour	判断时间是否在系统时间的下一小时
15	智能时间	DateTime.IsInNextMinute	判断时间是否在系统时间的下一分钟
16	智能时间	DateTime.IsInNextSecond	判断时间是否在系统时间的下一秒钟
17	智能时间	DateTime.IsInPreviousHour	判断时间是否在系统时间的上一小时
18	智能时间	DateTime.IsInPreviousMinute	判断时间是否在系统时间的上一分钟
19	智能时间	DateTime.IsInPreviousSecond	判断时间是否在系统时间的上一秒钟
20	智能时间	DateTime.IsInNextNHours	判断时间是否在系统时间的之后若干小时
21	智能时间	DateTime.IsInNextNMinutes	判断时间是否在系统时间的之后若干分钟

续表

类	分 类	函 数 名	概 述
22	智能时间	DateTime.IsInNextNSeconds	判断时间是否在系统时间的之后若干秒钟
23	智能时间	DateTime.IsInPreviousNHours	判断时间是否在系统时间的之前若干小时
24	智能时间	DateTime.IsInPreviousNMinutes	判断时间是否在系统时间的之前若干分钟
25	智能时间	DateTime.IsInPreviousNSeconds	判断时间是否在系统时间的之前若干秒钟

1. DateTime.Date

函数：DateTime.Date

语法 : DateTim.Date(**datetime** as any) as nullable date

说明 : 获取日期时间的日期部分, 例如 :

```
= DateTime.Date(#datetime(2013,9,10,12,34,56))
```

输出 2013/09/10

2. DateTime.Time

函数 : DateTime.Time

语法 : DateTim.Time(**datetime** as any) as nullable time

说明 : 获取日期时间的时间部分, 例如 :

```
= DateTime.Time(#datetime(2013,9,10,12,34,56))
```

输出 12:34:56

3. DateTime.LocalNow

函数 : DateTime.LocalNow

语法 : DateTim.LocalNow() as datetime

说明 : 无参数, 获取当前系统日期时间的动态值, 例如 :

```
= DateTime.LocalNow()
```

输出 2016-10-02T13:36:44.8302421

4. DateTime.FixedLocalNow

函数 : DateTime.FixedLocalNow

语法 : DateTim.FixedLocalNow() as datetime

说明 : 无参数, 获取当前系统日期时间的固定值, 例如 :

```
= DateTime.FixedLocalNow()
```

输出 2016-10-02T13:38:15.4562549

5. DateTime.ToRecord

函数 : DateTime.ToRecord

语法 : DateTim.ToRecord(**datetime** as datetime) as record

说明 : 返回日期时间 datetime 的年、月、日、时、分、秒的 Record 记录, 例如 :

```
= DateTime.ToRecord(#datetime(2016,9,12,10,23,40))
```

输出 [Year=2016, Month=9, Day=12, Hour=10, Minute=23, Second=40]

6. DateTime.FromFileTime

函数 : DateTime.FromFileTime

语法 : DateTim.FromFileTime(**value** as any, *optional* **culture** as nullable text) as nullable datetime

说明 : 从给定的 value 返回 datetime 值, 例如 :

```
= DateTime.FromFileTime(129896402829842245)
```

输出 2012-08-17T09:24:42.9842245

7. DateTime.AddZone

函数：DateTime.AddZone

语法：DateTim.AddZone(**datetime** as nullable datetime, **timezoneHours** as number, *optional* **timezoneMinutes** as nullable number) as nullable datetimezone

说明：给日期时间设置世界时区，例如：

```
= DateTime.AddZone(#datetime(2013,9,10,12,34,56),8)
                                输出      2013/09/10 12:34:56 +08:00
= DateTime.AddZone(#datetime(2013,9,10,12,34,56),7,30)
                                输出      2013/09/10 12:34:56 +07:30
```

8. DateTime.From

函数：DateTime.From

语法：DateTim.From(**value** as any, *optional* **culture** as nullable text) as nullable datetime

说明：从给定的 value 值返回 datetime 的值，例如：

```
= DateTime.From("2016-10-2 12:34:56")  输出      2016/10/02 12:34:56
= DateTime.From("2014 年 8 月 8 日")    输出      2014/08/08 0:00:00
= DateTime.From(#date(2016,8,8))        输出      2016/08/08 0:00:00
= DateTime.From(#time(10,8,8))          输出      1899/12/30 10:08:08
= DateTime.From(43210.12345)            输出      2018-04-20T02:57:4A.08
```

culture 常用的参数"en-us"表示英语，"zh-cn"表示中文，省略时以安装的地区语言为参考。

value 支持的数据类型如表 A-41 所示。

6-41

参 数	含 义
text	文本表示形式的 datetime 值
date	日期
datetimezone	日期\时间\时区
time	时间
number	整数表示年月日，小数表示时分秒

9. DateTime.FromText

函数 : DateTime.FromText

语法 : DateTime.FromText(**text** as nullable text, *optional* **culture** as nullable text) as nullable datetime

说明 : 按照 ISO8601 格式标准, 从文本格式的 text 创建日期时间 datetime, 例如 :

= DateTime.FromText("2016-10-2 12:34:56")	输出	2016/10/02 12:34:56
= DateTime.FromText("2014 年 8 月 8 日")	输出	2014/08/08 0:00:00
= DateTime.FromText("2014 年 8 月")	输出	2014/08/01 0:00:00
= DateTime.FromText("2014 年")	输出	2014/01/01 0:00:00
= DateTime.FromText("8 月 8 日")	输出	2016/08/08 0:00:00
= DateTime.FromText("8 日")	输出	2016/01/08 0:00:00
= DateTime.FromText("15 时 8 分 27 秒")	输出	2016/10/02 15:08:27
= DateTime.FromText("July27")	输出	2016/07/27 0:00:00

culture 常用的参数 "en-us" 表示英语, "zh-cn" 表示中文, 省略时以安装的地区语言为参考。

10. DateTime.ToText

函数 : DateTime.ToText

语法 : DateTime.ToText(**datetime** as nullable datetime, *optional* **format** as nullable text, *optional* culture as nullable text) as nullable datetime

说明 : 根据 format 提供的文本格式返回 datetime 的文本日期时间, 例如 :

datetime = #datetime(2016,9,12,10,23,40)		
= DateTime.ToText(datetime, "ss")	输出	"40"
= DateTime.ToText(datetime, "s")	输出	2016-09-12T10:23:40"
= DateTime.ToText(datetime, "m")	输出	"9 月 12 日"
= DateTime.ToText(datetime, "mm")	输出	"23"
= DateTime.ToText(datetime, "hh")	输出	"10"
= DateTime.ToText(datetime, "d")	输出	"2016/9/12"
= DateTime.ToText(datetime, "dd")	输出	"12"
= DateTime.ToText(datetime, "ddd")	输出	"周一"
= DateTime.ToText(datetime, "dddd")	输出	"星期一"
= DateTime.ToText(datetime, "MM")	输出	"09"
= DateTime.ToText(datetime, "MMM")	输出	"9 月"
= DateTime.ToText(datetime, "MMMM")	输出	"九月"
= DateTime.ToText(datetime, "Y")	输出	"2016 年 9 月"
= DateTime.ToText(datetime, "yy")	输出	"16"
= DateTime.ToText(datetime, "yyyy")	输出	"2016"
= DateTime.ToText(datetime, "Y")	输出	"2016 年 9 月"
= DateTime.ToText(datetime, "MMM", "en-us")	输出	"Sep"
= DateTime.ToText(datetime, "MMMM", "en-us")	输出	"September"
= DateTime.ToText(datetime, "ddd", "en-us")	输出	"Mon"

Power Query: 用 Excel 玩转商业智能数据处理

```
= DateTime.ToText(datetime, "dddd", "en-us") 输出    "Monday"
= DateTime.ToText(datetime, "MMMM d, yyyy dddd", "en-us")
      输出    "September 12, 2016 Monday"
= DateTime.ToText(datetime, "yyyy 年 M 月 d 日 dddd", "zh-cn")
      输出    "2016 年 9 月 12 日 星期一"
= DateTime.ToText(datetime, "gyyyy 年 MM 月 d 日 h 时 m 分 s 秒")
      输出    "公元 2016 年 09 月 12 日 10 时 23 分 40 秒"
```

culture 常用的参数"en-us"表示英语,"zh-cn"表示中文,省略时以安装的地区语言为参考。

11. DateTime.IsInCurrentHour

函数 : DateTime.IsInCurrentHour

语法 : DateTime.IsInCurrentHour(**datetime** as any) as nullable logical

说明 : 判断日期时间是否在系统时间的当前小时,例如 :

```
= DateTime.IsInCurrentHour(#datetime(2016,9,12,10,23,40)) 输出 FALSE
```

12. DateTime.IsInCurrentMinute

函数 : DateTime.IsInCurrentMinute

语法 : DateTime.IsInCurrentMinute(**datetime** as any) as nullable logical

说明 : 返回判断日期时间是否是系统时间的当前分钟,例如 :

```
= DateTime.IsInCurrentMinute(#datetime(2016,9,12,10,23,40)) 输出 FALSE
```

13. DateTime.IsInCurrentSecond

函数 : DateTime.IsInCurrentSecond

语法 : DateTime.IsInCurrentSecond(**datetime** as any) as nullable logical

说明 : 判断日期时间是否在系统时间的当前分钟,例如 :

```
= DateTime.IsInCurrentSecond (#datetime(2016,9,12,10,23,40)) 输出 FALSE
```

14. DateTime.IsInNextHour

函数 : DateTime.IsInNextHour

语法 : DateTime.IsInNextHour(**datetime** as any) as nullable logical

说明 : 返回时间是否是系统时间的下一小时。

```
= DateTime.IsInNextHour(#datetime(2016,9,12,10,23,40)) 输出 FALSE
```

15. DateTime.IsInNextMinute

函数 : DateTime.IsInNextMinute

语法 : DateTime.IsInNextMinute(**datetime** as any) as nullable logical

说明 : 判断日期时间是否在系统时间的下一分钟, 例如 :

```
= DateTime.IsInNextMinute(#datetime(2016,9,12,10,23,40))
```

 输出 FALSE

16. DateTime.IsInNextSecond

函数 : DateTime.IsInNextSecond

语法 : DateTime.IsInNextSecond(**datetime** as any) as nullable logical

说明 : 判断日期时间是否是系统时间的下一秒钟, 例如 :

```
= DateTime.IsInNextSecond (#datetime(2016,9,12,10,23,40))
```

 输出 FALSE

17. DateTime.IsInPreviousHour

函数 : DateTime.IsInPreviousHour

语法 : DateTime.IsInPreviousHour(**datetime** as any) as nullable logical

说明 : 判断日期时间是否是系统时间的上一小时, 例如 :

```
= DateTime.IsInPreviousHour(#datetime(2016,9,12,10,23,40))
```

 输出 FALSE

18. DateTime.IsInPreviousMinute

函数 : DateTime.IsInPreviousMinute

语法 : DateTime.IsInPreviousMinute(**datetime** as any) as nullable logical

说明 : 判断日期时间是否是系统时间的上一分钟, 例如 :

```
= DateTime.IsInPreviousMinute(#datetime(2016,9,12,10,23,40))
```

 输出 FALSE

19. DateTime.IsInPreviousSecond

函数 : DateTime.IsInPreviousSecond

语法 : DateTime.IsInPreviousSecond(**datetime** as any) as nullable logical

说明 : 判断日期时间是否是系统时间的上一秒钟, 例如 :

```
= DateTime.IsInPreviousSecond (#datetime(2016,9,12,10,23,40))
```

 输出 FALSE

20. DateTime.IsInNextNHours

函数 : DateTime.IsInNextNHours

语法 : DateTime.IsInNextNHours(**datetime** as any, **hours** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的接下来若干小时之内, 例如 :

```
= DateTime.IsInNextNHours(#datetime(2016,10,2,18,23,40),2)
```

 输出 TURE

21. DateTime.IsInNextNMinutes

函数 : DateTime.IsInNextNMinutes

语法 : DateTime.IsInNextNMinutes(**datetime** as any, **Minutes** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的接下来若干分钟之内, 例如 :

```
= DateTime.IsInNextNMinutes(#datetime(2016,10,2,18,23,40), 30) 输出 FALSE
```

22. DateTime.IsInNextNSeconds

函数 : DateTime.IsInNextNSeconds

语法 : DateTime.IsInNextNSeconds(**datetime** as any, **Seconds** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的接下来若干秒钟之内例如 :

```
= DateTime.IsInNextNSeconds(#datetime(2016,10,2,18,23,40), 200) 输出 FALSE
```

23. DateTime.IsInPreviousNHours

函数 : DateTime.IsInPreviousNHours

语法 : DateTime.IsInPreviousNHours(**datetime** as any, **hours** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的之前若干小时之内例如 :

```
= DateTime.IsInPreviousNHours(#datetime(2016,10,2,18,23,40), 2) 输出 False
```

24. DateTime.IsInPreviousNMinutes

函数 : DateTime.IsInPreviousNMinutes

语法 : DateTime.IsInPreviousNMinutes(**datetime** as any, **Minutes** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的之前若干分钟之内例如 :

```
= DateTime.IsInPreviousNMinutes(#datetime(2016,10,2,18,23,40), 30)
输出 FALSE
```

25. DateTime.IsInPreviousNSeconds

函数 : DateTime.IsInPreviousNSeconds

语法 : DateTime.IsInPreviousNSeconds(**datetime** as any, **Seconds** as number) as nullable logical

说明 : 判断日期时间是否在系统时间的之前若干秒钟之内, 例如 :

```
= DateTime.IsInPreviousNSeconds(#datetime(2016,10,2,18,23,40), 20)
输出 FALSE
```

A.7 Duration 类函数

Duration 类函数说明如表 A-42 所示

表 A-42

类	分 类	函 数 名	概 述
1	计算	Duration.Days	获取持续时间的日期部分
2	计算	Duration.Hours	获取持续时间的小时数
3	计算	Duration.Minutes	获取持续时间的分钟数
4	计算	Duration.Seconds	获取持续时间的秒数
5	创建	Duration.ToRecord	获取持续时间的各部分 Record
6	转换	Duration.TotalDays	将持续时间换算成总天数
7	转换	Duration.TotalHours	将持续时间换算成总小时数
8	转换	Duration.TotalMinutes	将持续时间换算成总分钟数
9	转换	Duration.TotalSeconds	将持续时间换算成总秒数
10	转换	Duration.From	从 value 获取持续时间
11	转换	Duration.FromText	从文本样式获取持续时间
12	转换	Duration.ToText	返回持续时间的文本样式

1. Duration.Days

函数 : Duration.Days

语法 : Duration.Days(**duration** as nullable duration) as nullable number

说明 : 获取持续时间的日期部分，例如：

= #duration(8, 10, 55, 17)	输出	8.10:55:17
= Duration.Days(#duration(8, 10, 55, 17))	输出	8

2. Duration.Hours

函数 : Duration.Hours

语法 : Duration.Hours(**duration** as nullable duration) as nullable number

说明 : 获取持续时间的小时数，例如：

= Duration.Hours(#duration(8, 10, 55, 17))	输出	10
--	----	----

3. Duration.Minutes

函数 : Duration.Minutes

语法 : Duration.Minutes(**duration** as nullable duration) as nullable number

说明：获取持续时间的分钟数，例如：

```
= Duration.Minutes(#duration(8, 10, 55, 17))
```

输出 55

4. Duration.Seconds

函数：Duration.Seconds

语法：Duration.Seconds(**duration** as nullable duration) as nullable number

说明：获取持续时间的秒钟部分。

```
= Duration.Seconds(#duration(8, 10, 55, 17))
```

输出 17

5. Duration.ToRecord

函数：Duration.ToRecord

语法：Duration.ToRecord(**duration** as duration) as record

说明：返回包含 duration 值的各部分的记录，例如：

```
= Duration.ToRecord(#duration(8, 10, 55, 17))
```

输出 [Days=8, Hours=10, Minutes=55, Seconds=17]

6. Duration.TotalDays

函数：Duration.TotalDays

语法：Duration.TotalDays(**duration** as nullable duration) as nullable number

说明：将持续时间换算成总天数，例如：

```
= Duration.TotalDays(#duration(8, 10, 55, 17))
```

输出 8.45505787037037

7. Duration.TotalHours

函数：Duration.TotalHours

语法：Duration.TotalHours(**duration** as nullable duration) as nullable number

说明：将持续时间换算成总小时数，例如：

```
= Duration.TotalHours(#duration(8, 10, 55, 17))
```

输出 202.92138888888889

8. Duration.TotalMinutes

函数：Duration.TotalMinutes

语法：Duration.TotalMinutes(**duration** as nullable duration) as nullable number

说明：将持续时间换算成总分钟数，例如：

```
= Duration.TotalMinutes(#duration(8, 10, 55, 17))
```

输出 12175.283333333333

9. Duration.TotalSeconds

函数 : Duration.TotalSeconds

语法 : Duration.TotalSeconds(**duration** as nullable duration) as nullable number

说明 : 将持续时间换算成总秒数, 例如 :

= Duration.TotalSeconds(#duration(8, 10, 55, 17)) 输出 730517

10. Duration.From

函数 : Duration.From

语法 : Duration.From(**value** as any) as nullable duration

说明 : 从给定的 value 值返回 duration 值, 例如 :

= Duration.From("10")	输出 10.00:00:00
= Duration.From("10.12:34")	输出 10.12:34:00
= Duration.From("10.12:34:56")	输出 10.12:34:56
= Duration.From(3.88765)	输出 3.21:18:12.96
= Duration.From(Number.Random())	输出 0.21:07:19.7688832
= Duration.From(Number.RandomBetween(1, 9))	输出 2.20:09:27.6801707

value 支持的数据类型如表 A-43 所示。

表 A-43

参 数	说 明
text	文本样式的持续时间
number	数值

11. Duration.FromText

函数 : Duration.FromText

语法 : Duration.FromText(**text** as nullable text) as nullable duration

说明 : 从指定的文本 text 返回持续时间值, 例如 :

= Duration.FromText("7.6:54:32.10") 输出 7.06:54:32.1000000

参数包含的范围如表 A-44 所示。

表 A-44

参 数	说 明	范 围
ddd	天数	为 0~24

hh	小时数	为 0 ~ 59
mm	分钟数	为 0 ~ 59
ss	秒钟数	为 0 ~ 9999999
ff	秒的小数部分	

12. Duration.ToText

函数：Duration.ToText

语法：Duration.ToText(**duration** as nullable duration, **format** as nullable text) as nullable text

说明：返回给定 duration 值的文本表现形式，例如：

= Duration.ToText(#duration(8, 10, 55, 17)) **输出** "8.10:55:17"

A.8 Record 类函数

Record 类函数说明如表 A-45 所示。

表 A-45

类	分 类	函 数 名	概 述
1	计算	Record.FieldCount	返回记录的字段数
2	判断	Record.HasFields	检查是否包含某个字段
3	选择	Record.Field	返回指定字段的值
4	选择	Record.FieldValues	返回记录中字段值的 list 列表
5	选择	Record.FieldNames	返回记录中字段名称的 list 列表
6	选择	Record.FieldOrDefault	返回记录中指定字段的值
7	选择	Record.SelectFields	返回记录中的指定列记录
8	转换	Record.AddField	添加一个字段
9	转换	Record.RemoveFields	删除指定字段
10	转换	Record.Combine	组合记录
11	转换	Record.TransformFields	转换字段数据类型
12	转换	Record.ReorderFields	重新排序字段顺序
13	转换	Record.RenameFields	重新命名字段标题
14	序列化	Record.FromTable	Table 转换成 Record

续表

类	分 类	函 数 名	概 述
15	序列化	Record.FromList	List 转换成 Record
16	序列化	Record.ToTable	Record 转换成 Table

17	序列化	Record.ToList	Record 转换成 List
----	-----	---------------	-----------------

1. Record.FieldCount

函数：Record.FieldCount

语法：Record.FieldCound(**record** as record) as number

说明：返回记录 record 中的字段数，例如：

```
let
    record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
    FieldCount = Record.FieldCount(record)
in
    FieldCount
```

如图 A-6 所示



图 A-6

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"]
FieldCount = Record.FieldCount(record)
```

输出 4

如图 A-6 所示。

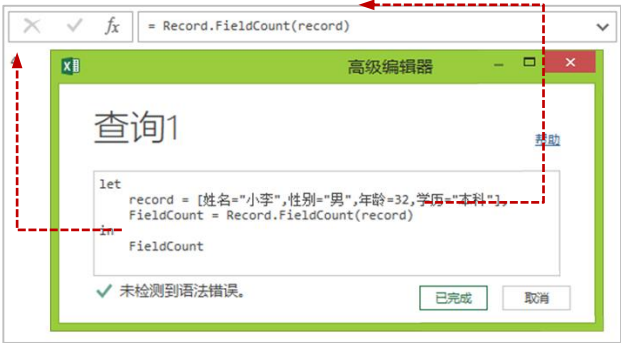


图 A-7

2. Record.HasFields

函数：Record.HasFields

语法：Record.HasFields(**record** as record, **fields** as any) as logical

说明：检查 record 记录中是否包含指定字段 fields，例如，如图 A-8 所示的表

✕	✓	fx	= [姓名="小李",性别="男",年龄=32,学历="本科"]	▼
姓名	小李			
性别	男			
年龄	32			
学历	本科			

图 A-8

```
record = [姓名="小李",性别="男",年龄=32,学历="本科"],
HasFields = Record.HasFields(record,"姓名")           //输出      TRUE
HasFields = Record.HasFields(record,{"姓名","年龄"})   //输出      TRUE
HasFields = Record.HasFields(record,{"姓名","身份证"}) //输出      FALSE
HasFields = Record.HasFields(record,"毕业院校")        //输出      FALSE
```

3. Record.Field

函数：Record.Field

语法：Record.Field(**record** as record, **field** as text) as any

说明：返回 record 中指定字段 field 的值，如果未找到字段，则将引发异常，例如：

```
record = [姓名="小李",性别="男",年龄=32,学历="本科"],
Field = Record.Field(record,"姓名")           //输出      小李
```

4. Record.FieldValues

函数：Record.FieldValues

语法：Record.FieldValues(**record** as record, **field** as text) as any

说明：返回 record 中的字段值的 list 列表，例如：

```
record = [姓名="小李",性别="男",年龄=32,学历="本科"],
FieldValues = Record.FieldValues(record)
```

如图 A-9 和图 A-10 所示。

姓名	小李
性别	男
年龄	32
学历	本科

record

图 A-9

	List
1	小李
2	男
3	32
4	本科

FieldValues

图 A-10

5. Record.FieldNames

函数：Record.FieldNames

语法：Record.FieldNames(**record** as record, **field** as text) as any

说明：返回 record 中的字段标题名称的 list 列表，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
FieldNames = Record.FieldNames(record)
```

如图 A-11 和图 A-12 所示。

姓名	小李
性别	男
年龄	32
学历	本科

record

图 A-11

	List
1	姓名
2	性别
3	年龄
4	学历

FieldNames

图 A-12

6. Record.FieldOrDefault

函数：Record.FieldOrDefault

语法：Record.FieldOrDefault(**record** as record, **field** as text, *optional* **defaultValue** as any) as any

说明：返回 record 中的指定字段 field 的值；如果未找到该字段，则用 defaultValue 值替换，若未指定，则返回 null，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
FieldOrDefault = Record.FieldOrDefault(record, "姓名", "清华大学")
//输出      小李
FieldOrDefault = Record.FieldOrDefault(record, "姓名", "清华大学")
//输出      小李
FieldOrDefault = Record.FieldOrDefault(record, "毕业院校")
//输出      null
FieldOrDefault = Record.FieldOrDefault(record, "毕业院校", "清华大学")
//输出      清华大学
```

7. Record.SelectFields

函数：Record.SelectFields

语法：Record.SelectFields(**record** as record, **field** as any, *optional* **missingField** as nullable MissingField.Type) as record

说明：从输入 record 返回一个记录，该记录仅包含在列表 fields 中指定的字段，例如：

MissingField.Type 参数说明，缺省时为 MissingField.Error，如表 A-46 所示。

表 A-46

参 数	说 明	值
-----	-----	---

Powery Query：用 Excel 玩转商业智能数据处理

MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略错误	1
MissingField.null	若无此字段，则返回 null	2

例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],  
SelectFields = Record.SelectFields(record, "姓名")
```

姓名	小李
----	----

```
SelectFields = Record.SelectFields(record, {"姓名", "毕业院校", "学历"}, 1)
```

姓名	小李
学历	本科

```
SelectFields = Record.SelectFields(record, {"姓名", "毕业院校", "学历"}, 2)
```

姓名	小李
毕业院校	null
学历	本科

8. Record.AddField

函数：Record.AddField

语法：Record.AddField(**record** as record, **fieldName** as text, value as any, *optional* **delayed** as nullable logical) as record

说明：给定字段 fieldName 的名称和值 value，将字段添加到记录 record，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],  
AddField = Record.AddField(record, "身高", 178)
```





			= Record.AddField(record, "身高", 178)	
姓名	小李			
性别	男			
年龄	32			
学历	本科			
身高	178			

图 A-13

9. Record.RemoveFields

函数：Record.RemoveFields

语法：Record.RemoveFields(**record** as record, **field** as any, *optional* **missingField** as nullable MissingField.Type) as record

说明：从输入 record 返回一个记录，该记录仅包含在列表 fields 中指定的字段，例如：

MissingField.Type 参数说明，缺省时为 MissingField.Error 如表 A-47 所示

表 A-7

参 数	说 明	值
MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略此错误	1
MissingField.null	若无此字段，则忽略此错误	2

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
RemoveFields = Record.RemoveFields(record, {"年龄", "学历"})
```

姓名	小李
性别	男

```
RemoveFields = Record.RemoveFields(record, {"年龄", "毕业院校"}, 1)
```

姓名	小李
性别	男
学历	本科

10. Record.Combine

函数：Record.Combine

语法：Record.Combine(records as list) as record

说明：从输入 record 返回一个记录，该记录仅包含在列表 fields 中指定的字段，例如：

```
record1 = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
record2 = [毕业院校="清华大学", 身高=178, 体重=78],
Combine = Record.Combine({record1, record2})
```

姓名	小李
性别	男
年龄	32
学历	本科

record1

图 A-14

毕业院校	清华大学
身高	178
体重	78

record2

图 A-15

✕	✓	f_x	= Record.Combine({record1, record2})	▼
姓名	小李			
性别	男			
年龄	32			
学历	本科			
毕业院校	清华大学			
身高	178			
体重	78			

Combine

图 A-16

11. Record.TransformFields

函数：Record.TransformFields

语法：Record.TransformFields(**record** as record, **transformOperations** as list, *optional* **missingField** as nullable MissingField.Type) as record

说明：将列表 transformOperation 中指定的转换应用到 record 后返回一个新的记录 record，一次可以转换一个或多个字段，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],  
TransformFields = Record.TransformFields(record, {{ "姓名", Text.From},  
        { "年龄", Text.From}})
```

transformOperation 的格式：

{{ "字段名称 1", 转换函数 1 }, { "字段名称 2", 转换函数 2 }}

MissingField.Type 参数说明如表 A-48 所示，缺省时为 MissingField.Error。

表 A-8

参 数	说 明	值
MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略此错误	1
MissingField.null	若无此字段，则返回 null	2

姓名	小李
性别	男
年龄	32
学历	本科

12. Record.ReorderFields

函数：Record.ReorderFields

语法：Record.ReorderFields(**record** as record, **fieldorder** as list, *optional* **missingField** as nullable MissingField.Type) as record

说明：对指定字段重新排列字段顺序，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],  
ReorderFields = Record.ReorderFields(record, { "姓名", "年龄", "学历", "性别" })
```

姓名	小李
性别	男
年龄	32
学历	本科

record

姓名	小李
年龄	32
学历	本科
性别	男

ReorderFields

MissingField.Type 参数说明如表 A-49 所示，缺省时为 MissingField.Error。

表 A-49

参 数	说 明	值
MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略此错误	1
MissingField.null	若无此字段，则返回 null	2

13. Record.RenameFields

函数：Record.RenameFields

语法：Record.RenameFields(**record** as record, **renames** as list, *optional* **missingField** as nullable MissingField.Type) as record

说明：重命名指定字段标题名称，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
RenameFields = Record.RenameFields(record, {"姓名", "名字"}, {"年龄", "岁数"}, 2)
```

姓名	小李
性别	男
年龄	32
学历	本科

record

名字	小李
性别	男
岁数	32
学历	本科

RenameFields

MissingField.Type 参数说明如表 A-50 所示，缺省时为 MissingField.Error。

表 A-50

参 数	说 明	值
MissingField.Error	若无此字段，错误警告	0
MissingField.Ignore	若无此字段，忽略此错误	1
MissingField.null	若无此字段，返回 null	2

14. Record.FromTable

函数：Record.FromTable

语法：Record.FromTable(**table** as table) as record

说明：从包含字段名称和值名称的记录 table 的表中返回记录，如果字段名称不是唯一的，则将引发异常，table 字段名称必须是 Name 和 Value，例如：

```
table = Table.FromRecords([Name="苹果", Value=A.5], [Name="香蕉", Value=3.5],
[Name="菠萝", Value=4.2])
```

Powery Query: 用 Excel 玩转商业智能数据处理

```
FromTable = Record.FromTable(table)
```

	ABC 123	Name	ABC 123	Value
1		苹果		6.5
2		香蕉		3.5
3		菠萝		4.2

table

苹果	6.5
香蕉	3.5
菠萝	6.2

FromTable

15. Record.FromList

函数：Record.FromList

语法：Record.FromList(**list** as list, **fields** as any) as record

说明：根据给定的一个字段值 list 和一组字段，返回一个记录。可以通过文本值列表或记录类型指定 fields，如果字段不是唯一，则将引发错误，例如

```
list = {A.5, 3.5, A.2},  
FromList = Record.FromList(list, {"苹果", "香蕉", "菠萝"}),  
FromList = Record.FromList(list, type [苹果=number, 香蕉=number, 菠萝=number])
```

	List
1	6.5
2	3.5
3	6.2

list

苹果	6.5
香蕉	3.5
菠萝	6.2

FromList

16. Record.ToTable

函数：Record.ToTable

语法：Record.ToTable(**record** as record) as table

说明：返回一个表，它包含 Name 和 Value 两个字段以及对应 Record 中每个字段的行，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],  
ToTable = Record.ToTable(record)
```

姓名	小李
性别	男
年龄	32
学历	本科

record

	A ^B C	Name	ABC 123	Value
1		姓名		小李
2		性别		男
3		年龄		32
4		学历		本科

ToTable

17. Record.ToList

函数：Record.ToList

语法：Record.ToList(**record** as record) as list

说明：返回包含输入 record 中的字段值 list 列表，例如：

```
record = [姓名="小李", 性别="男", 年龄=32, 学历="本科"],
ToList = Record.ToList(record)
```

姓名	小李
性别	男
年龄	32
学历	本科

record

List	
1	小李
2	男
3	32
4	本科

ToList

A.9 List 类函数

List 类函数说明如表 A-51 所示

表 A-51

类	分 类	函 数 名	概 述
1	计算	List.Sum	计算非空值的总和
2	计算	List.Count	计算总项行数
3	计算	List.NonNullCount	计算非空总项数
4	计算	List.Average	计算算术平均值
5	计算	List.Max	计算最大值，如果为空则可以选择另一值
6	计算	List.MaxN	计算前几个最大值
7	计算	List.Min	计算最小值，如果为空则可以选择另一值
8	计算	List.MinN	计算前几个最小值
9	计算	List.Median	计算中位值
10	计算	List.Mode	计算众数
11	计算	List.ModeS	计算多个众数
12	计算	List.Product	计算 list 中所有值的乘积
13	计算	List.Covariance	计算两个 list 之间协方差
14	计算	List.StandardDeviation	计算标准偏差
15	判断	List.IsEmpty	判断 list 是否为空
16	判断	List.Contains	判断 list 列表是否包含某值
17	判断	List.ContainsAll	判断 list1 中是否包含 list2 的所有值
18	判断	List.ContainsAny	判断 list1 中是否包含 list2 的任意值
19	判断	List.AnyTrue	判断任意表达式是否为 true
20	判断	List.AllTrue	判断所有表达式是否为 true

Powery Query：用 Excel 玩转商业智能数据处理

21	判断	List.MatchesAll	判断列表中是否满足所有条件
22	判断	List.MatchesAny	判断列表中是否有部分满足条件
续表			
类	分 类	函 数 名	概 述
23	判断	List.IsDistinct	判断是否有重复项
24	判断	List.PositionOf	返回指定值在 list 中的位置
25	判断	List.PositionOfAny	返回 list2 中任意值在 list1 的位置
26	选择	List.Select	按给定条件选择项目
27	选择	List.First	返回第一个值，如果为空，则可以返回另外一个值
28	选择	List.FirstN	返回前 N 个值的 list 列表
29	选择	List.Last	返回最后值，如果为空可以返回另外一个值
30	选择	List.LastN	返回后 N 个值的 list 列表
31	选择	List.Range	选择从指定位置开始若干个项目的子集
32	选择	List.Single	如果列表中只有一项，则返回该项，否则错误
33	选择	List.SingleOrDefault	如果列表中只有一项，则返回该项，否则返回指定项
34	选择	List.FindText	返回包含某文本的文本 list 列表
35	选择	List.Alternate	根据复杂的隔行规则取数组成新的 list 列表
36	操作	List.Sort	排序
37	操作	List.Distinct	删除重复项
38	操作	List.Positions	返回 list 的位置列表
39	操作	List.ReplaceRange	从指定的位置起替换新的 list
40	操作	List.InsertRange	在指定位置插入新的 list 列表
41	操作	List.RemoveRange	从指定的位置起删除若干个值
42	操作	List.RemoveFirstN	删除前面的几个值
43	操作	List.RemoveLastN	删除后面的几个值
44	操作	List.RemoveMatchingItems	从 list1 中删除 list2 中出现的值
45	操作	List.RemoveItems	从 list1 中删除 list2 所指定位置出现的值
46	操作	List.RemoveNulls	删除 list 中的空值
47	操作	List.Skip	跳过 list 中的前几项
48	操作	List.Transform	通过转换函数获得新的 list 列表
49	操作	List.Repeat	按指定次数重复 list 获得一个新的 list
50	操作	List.Reverse	逆序 list 列表
51	操作	List.ReplaceMatchingItems	替换 list1 中在 list2 指定位置中的值为 list3 的值
52	操作	List.ReplaceValue	替换 list 中指定的值为新值
53	集合	List.Combine	合并多个 list 为新 list

54	集合	List.Difference	返回 list1 与 list2 的差集
55	集合	List.Intersect	返回所有 list 的交集
56	集合	List.Union	返回所有 list 的并集

续表

类	分 类	函 数 名	概 述
57	创建	List.DateTimes	根据规则创建日期时间列表
58	创建	List.Dates	根据规则创建日期列表
59	创建	List.Times	根据规则创建时间列表
60	创建	List.DateTimeZones	根据规则创建日期时间时区列表
61	创建	List.Durations	根据规则创建持续时间列表
62	创建	List.Numbers	根据规则创建数字列表
63	创建	List.Random	创建随机值的列表

1. List.Sum

函数：List.Sum

语法：List.Sum(**list** as list, *optional precision* as nullable Precision.Type) as any

说明：计算列表 list 中所有非表 A-52 null 值的总和。precision 确定了计算精度，参数说明如表 A-78 所示。

表 A-52

参 数	说 明
Precision.Double	双精度
Precision.Decimal	十进制

例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Sum = List.Sum(list) //输出 219
```

= {32,56,null,45,10,40,19,null,17,null} ▾	
List	
1	32
2	56
3	null
4	45
5	10
6	40
7	19
8	null
9	17
10	null

2. List.Count

函数：List.Count

语法：List.Count(**list** as list) as number

说明：计算列表 list 中的总项数，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},  
Count = List.Count(list) //输出 10
```

3. List.NonNullCount

函数：List.NonNullCount

语法：List.NonNullCount(**list** as list) as number

说明：计算列表 list 中的非 null 项数，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},  
NonNullCount = List.NonNullCount(list) //输出 7
```

4. List.Average

函数：List.Average

语法：List.Average(**list** as list, *optional precision* as nullable Precision.Type) as any

说明：计算列表 list 中各项目的平均值，支持处理同一数据类型的输出结果。可以处理 number、date、time、datetime、datetimezone 和 duration 值。若列表为空则返回 null，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},  
= List.Average(list) //输出 31.285714285714285
```

precision 确定了计算精度参数如表 A-53 所示。

表 A-53

参 数	说 明
Precision.Double	双精度
Precision.Decimal	十进制

5. List.Max

函数：List.Max

语法：List.Max(**list** as list, *optional default* as any, *optional comparisonCriteria* as any, *optional includeNulls* as nullable logical) as any

说明：返回列表 list 中各项目的最大值；如果列表为空，则返回可选的默认值 default。可以指定可选的 comparisonCriteria 值来确定如何在列表中比较项，如果此参数为 null，将使用默

认的比较；可选的 `includeNulls` 决定 `list` 是否在包含 `null` 值的情况下比较，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Max = List.Max(list) //输出 56
Max = List.Max({null},-1) //输出 -1
Max = List.Max({},-1) //输出 -1
Max = List.Max(list,-1,0,true) //输出 56
Max = List.Max(list,-1,0,false) //输出 56
Max = List.Max(list,-1,1,false) //输出 10
```

6. List.MaxN

函数： `List.MaxN`

语法： `List.MaxN(list as list, countOrCondition as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as list`

说明： 返回列表 `list` 中的最大值；参数 `countOrCondition` 指定要返回最大值的个数或筛选条件，如果指定一个数，则返回以降序排序的最多包含几个最大值的 `list` 列表，如果指定一个条件，则返回最初满足该条件所有项目中的若干最大值，不再考虑之后的其他项目。如果条件不满足时，则返回列表中的最大值。可以指定可选的 `comparisonCriteria` 值来确定如何在列表中比较项，如果此参数为 `null`，则将使用默认的比较；可选的 `includeNulls` 决定 `list` 是否在包含 `null` 值的情况下比较，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
MaxN = List.MaxN(list,1) //输出 56
MaxN = List.MaxN(list,3) //输出 {56,45,40}
MaxN = List.MaxN(list,each _<40) //输出 {56,45}
MaxN = List.MaxN(list,each _>0,1) //输出 {10,17,19,32}
MaxN = List.MaxN(list,7,1) //输出 {10,17,19,32,40,45,56}
MaxN = List.MaxN(list,4,0,true) //输出 {56,45,40,32}
```

7. List.Min

函数： `List.Min`

语法： `List.Min(list as list, optional default as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as any`

说明： 返回列表 `list` 中各项目的最小值；如果列表为空，则返回可选的默认值 `default`。可以指定可选的 `comparisonCriteria` 值来确定如何在列表中比较项，如果此参数为 `null`，则使用默认的比较；可选的 `includeNulls` 决定 `list` 是否在包含 `null` 值的情况下比较，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Min = List.Min(list) //输出 10
Min = List.Min({null},-1) //输出 -1
```

Powery Query: 用 Excel 玩转商业智能数据处理

```

Min = List.Min({}, -1)           //输出  -1
Min = List.Min(list, -1, 0, true) //输出  null
Min = List.Min(list, -1, 0, false) //输出  10
Min = List.Min(list, -1, 1, false) //输出  56

```

8. List.MinN

函数 : List.MinN

语法 : List.MinN(**list** as list, **countOrCondition** as any, *optional* **comparisonCriteria** as any, *optional* includeNulls as nullable logical) as list

说明 : 返回列表 list 中的最小值 ; 参数 countOrCondition 指定要返回最小值的个数或筛选条件 , 如果指定一个数 , 则返回以升序排序的最多包含几个最小值的 list 列表 , 如果指定一个条件 , 则返回最初满足该条件所有项目中的若干最小值 , 不再考虑之后的其他项目。如果条件不满足时 , 则返回列表中的最小值。可以指定可选的 comparisonCriteria 值来确定如何在列表中比较项 , 如果此参数为 null , 则使用默认的比较 ; 可选的 includeNulls 决定 list 是否在包含 null 值的情况下比较 , 例如 :

```

list = {32, 56, null, 45, 10, 40, 19, null, 17, null},
MinN = List.MinN(list, 1)           //输出  10
MinN = List.MinN(list, 3)           //输出  {10, 17, 19}
MinN = List.MinN(list, each _ < 40) //输出  {10, 17, 19, 32}
MinN = List.MinN(list, each _ > 40, 1) //输出  {56, 45}
MinN = List.MinN(list, 7, 1)         //输出  {56, 45, 40, 32, 19, 17, 10}
MinN = List.MinN(list, 4, 0, true)   //输出  {null, null, null, 10}

```

9. List.Median

函数 : List.Median

语法 : List.Median(**list** as list, *optional* **comparisonCriteria** as any) as any

说明 : 返回列表 list 的中位数项 , 如果列表为空 , 则会引发异常 , 如果有偶数项 , 则取两个中位值中的较小者 , 例如 :

```

list = {32, 56, null, 45, 10, 40, 19, null, 17, null},
Median = List.Median(list)           //输出  32
Median = List.Median({1, 2, 3, 4}, 0) //输出  2
Median = List.Median({1, 2, 3, 4}, 1) //输出  3

```

10. List.Mode

函数 : List.Mode

语法 : List.Mode(**list** as list, *optional* **equationCriteria** as any) as any

说明 : 返回列表 list 中出现次数最多的项目 , 如果列表为空 , 则会引发异常 ; 如果出现最多的项

有多个，则选择其中的最后一项。可以指定可选参数 `equationCriteria` 来控制相等测试，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Mode = List.Mode(list) //输出 "B"
Mode = List.Mode({3,4,4,"A",5,5,"A","A",5}) //输出 5
Mode = List.Mode({5,3,4,4,"A",5,"A","A",5}) //输出 "A"
```

11. List.Modes

函数：List.Modes

语法：List.Modes(**list** as list, *optional* **equationCriteria** as any) as list

说明：返回列表 `list` 中出现次数最多的项目的 `list` 列表，如果列表为空，则会引发异常。可以指定可选参数 `equationCriteria` 来控制相等测试，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Modes = List.Modes({3,4,4,"A",5,5,"A","A",5}) //输出 {"A",5}
```

12. List.Product

函数：List.Product

语法：List.Product(**numbersList** as list, *optional* **precision** as nullable Precision.Type) as nullable number

说明：计算列表 `list` 中各项目中非 `null` 的乘积，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
Product = List.Product(list) //输出 10418688000
```

参数说明如表 A-54 所示。

表 A-54

参 数	说 明
Precision.Double	双精度
Precision.Decimal	十进制

13. List.Covariance

函数：List.Covariance

语法：List.Covariance(**numberList1** as list, **numberList2** as list) as nullable number

说明：计算 `numberList1` 和 `numberList2` 之间的协方差，`numberList1` 和 `numberList2` 必须包含相同数目的 `number` 值，例如：

```
numberlist1 = {1,3,5},
```

```
numberList2 = {2,1,3},  
Covariance = List.Covariance(numberList1,numberList2)  
//输出 0.666666666666667
```

14. List.StandardDeviation

函数 : List.StandardDeviation

语法 : List.StandardDeviation(**numbersList** as list) as nullable number

说明 : 计算基于样本估计的列表 numbersList 中的值的标准偏差, 例如 :

```
numbersList = {1..10},  
StandardDeviation = List.StandardDeviation(numbersList)  
//输出 3.0276503540974917
```

15. List.IsEmpty

函数 : List.IsEmpty

语法 : List.IsEmpty(**list** as list) as logical

说明 : 判断 list 是否是为空, 如果是, 则返回 TRUE, 否则返回 FALSE 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},  
IsEmpty = List.IsEmpty(list) //输出 FALSE  
IsEmpty = List.IsEmpty({null}) //输出 FALSE  
IsEmpty = List.IsEmpty({}) //输出 TRUE
```

16. List.Contains

函数 : List.Contains

语法 : List.Contains(**list** as list, **value** as any, *optional* **equationCriteria** as any) as logical

说明 : 判断 list 中是否包含 value, 如果包含则返回 TRUE, 否则返回 FALSE。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},  
Contains = List.Contains(list,null) //输出 TRUE  
Contains = List.Contains(list,32) //输出 TRUE  
Contains = List.Contains(list,99) //输出 FALSE
```

17. List.ContainsAll

函数 : List.ContainsAll

语法 : List.ContainsAll(**list** as list, **values** as list, *optional* **equationCriteria** as any) as logical

说明 : 判断 list 中是否同时包含 values 的所有项目, 如果同时包含, 则返回 TRUE, 否则返回 FALSE。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
ContainsAll = List.ContainsAll(list,{null,40})           //输出  TRUE
ContainsAll = List.ContainsAll(list,{32,56,10})          //输出  TRUE
ContainsAll = List.ContainsAll(list,{32,50,99})          //输出  FALSE
```

18. List.ContainsAny

函数 : List.ContainsAny

语法 : List.ContainsAny(**list** as list, **values** as list, *optional equationCriteria* as any) as logical

说明 : 判断 list 中是否包含 values 的任意一个项目, 如果包含则返回 TRUE, 否则返回 FALSE。

可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
ContainsAny = List.ContainsAny(list,{1,40,9})           //输出  TRUE
ContainsAny = List.ContainsAny(list,{56,10,99})          //输出  TRUE
ContainsAny = List.ContainsAny(list,{1,2,3})             //输出  FALSE
```

19. List.AnyTrue

函数 : List.AnyTrue

语法 : List.AnyTrue(**list** as list) as logical

说明 : 判断 list 列表中的任意表达式为 TRUE, 则返回 TRUE, 否则返回 FALSE, 例如 :

```
list = {true,false,1>2,"A">"B"},
AnyTrue = List.AnyTrue(list)                             //输出  TRUE
```

20. List.AllTrue

函数 : List.AllTrue

语法 : List.AllTrue(**list** as list) as logical

说明 : 判断 list 列表中的所有表达式均为 TRUE, 则返回 TRUE, 否则返回 FALSE, 例如 :

```
list = {true,false,1>2,"A">"B"},
AllTrue = List.AllTrue(list)                             //输出  FALSE
AllTrue = List.AllTrue({1<2,"A"<"B",Logical.From(1)}) //输出  TRUE
```

21. List.MatchesAll

函数 : List.MatchesAll

语法 : List.MatchesAll(**list** as list, **condition** as function) as logical

说明 : 如果 list 中的所有值均满足条件函数 condition, 则返回 TRUE, 否则返回 FALSE, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
MatchesAll = List.MatchesAll(list,each not null)         //输出  TRUE
```

```
MatchesAll = List.MatchesAll({75,88,92}, each _>59)           //输出  TRUE
MatchesAll = List.MatchesAll({75,88,92,59}, each _>59)         //输出  FALSE
```

22. List.MatchesAny

函数：List.MatchesAny

语法：List.MatchesAny(**list** as list, **condition** as function) as logical

说明：如果 list 中的所有值均满足条件函数 condition，则返回 TRUE，否则返回 FALSE，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
MatchesAny = List.MatchesAny(list,each not null)           //输出  TRUE
MatchesAny = List.MatchesAny({75,88,92}, each _>59)         //输出  TRUE
MatchesAny = List.MatchesAny({75,88,92,59}, each _>59)     //输出  FALSE
```

23. List.IsDistinct

函数：List.IsDistinct

语法：List.IsDistinct(**list** as list, *optional* **equationCriteria** as any) as logical

说明：判断 list 中是否有重复项，如果不重复则返回 TRUE，否则返回 FALSE，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
IsDistinct = List.IsDistinct(list)                           //输出  FALSE
IsDistinct = List.IsDistinct({1..9,"A".. "z"})                //输出  TRUE
```

24. List.PositionOf

函数：List.PositionOf

语法：List.PositionOf(**list** as list,value as any, *optional* **occurrence** as nullable Occurrence.Type, *optional* **equationCriteria** as any) as any

说明：返回值 value 在列表 list 中出现的位置，如果值不出现，则返回-1，可以指定可选的出现次数参数 occurrence，位置从 0 开始。可以指定可选参数 equationCriteria 来控制相等测试，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},
PositionOf = List.PositionOf(list,null)                       //输出  2
PositionOf = List.PositionOf(list,null,1)                     //输出  9
PositionOf = List.PositionOf(list,null,2)                     //输出  {2,7,9}
```

occurrence 参数说明如表 A-55 所示，缺省时为 Occurrence.First。

6-55

参 数	说 明	值
Occurrence.First	第一次出现的位置	0

Occurrence.Last	最后一次出现的位置	1
Occurrence.All	出现的所有位置	2

25. List.PositionOfAny

函数 : List.PositionOfAny

语法 : List.PositionOfAny(**list** as list, **values** as list, *optional occurrence* as nullable Occurrence.Type, *optional equationCriteria* as any) as any

说明 : 返回值 values 列表中的任意一个值在列表 list 中出现的位置, 如果值不出现, 则返回-1, 可以指定可选的出现次数参数 occurrence, 位置从 0 开始。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
PositionOfAny = List.PositionOfAny(list, {null,17,32,45})           //输出   0
PositionOfAny = List.PositionOfAny(list, {null,17,32,45},1)       //输出   9
PositionOfAny = List.PositionOfAny(list, {null,17,32,45},2)
//输出   {0,2,3,7,8,9}
```

occurrence 参数说明如表 A-56 所示, 缺省时为 Occurrence.First。

表 A-56

参 数	说 明	值
Occurrence.First	第一次出现的位置	0
Occurrence.Last	最后一次出现的位置	1
Occurrence.All	出现的所有位置	2

26. List.Select

函数 : List.Select

语法 : List.Select(**list** as list, **selection** as function) as list

说明 : 从 list 列表返回匹配选择条件 selection 的值的 list 列表, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
Select = List.Select(list, each _ <> null)
//输出   {32,56,45,10,40,19,17}
Select = List.Select(list, each _<30) //输出   {10,19,17}
Select = List.Select(list, each _<30 or _>40)
//输出   {45,45,10,19,17}
```

27. List.First

函数 : List.First

Powery Query: 用 Excel 玩转商业智能数据处理

语法 : List.First(**list** as list, *optional* **defaultValue** as any) as any

说明 : 返回 list 列表的第一项 ; 如果列表为空 , 则返回可选的 defaultValue , 若未指定可选 , 则返回 null , 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
First = List.First(list)                //输出    32
First = List.First({})                  //输出    null
First = List.First({}, "ABC")           //输出    "ABC"
```

28. List.FirstN

函数 : List.FirstN

语法 : List.FirstN(**list** as list, **countOrCondition** as any) as any

说明 : 返回 list 列表的前面多项的 list 列表 ; 如果 countOrCondition 指定一个数 , 则返回指定的项目数 , 如果 countOrCondition 指定一个条件 , 则返回最初满足条件的所有项 , 一旦不满足条件则不考虑后面的项 , 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
FirstN = List.FirstN(list,1)             //输出    {32}
FirstN = List.FirstN(list,1)             //输出    {32}
FirstN = List.FirstN(list,5)             //输出    {32,56,58,45,-1}
= List.FirstN(list,each _ > 0)           //输出    {32,56,58,45}
```

29. List.Last

函数 : List.Last

语法 : List.Last(**list** as list, *optional* **defaultValue** as any) as any

说明 : 返回 list 列表的最后一项 ; 如果列表为空 , 则返回可选的 defaultValue , 若未指定可选 , 则返回 null , 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
Last = List.Last(list)                   //输出    10
Last = List.Last({})                     //输出    null
Last = List.Last({}, "ABC")              //输出    "ABC"
```

30. List.LastN

函数 : List.LastN

语法 : List.LastN(**list** as list, **countOrCondition** as any) as any

说明 : 返回 list 列表的后面多项的 list 列表 ; 如果 countOrCondition 指定一个数 , 则返回指定的项目数 , 如果 countOrCondition 指定一个条件 , 则返回最初满足条件的所有项 , 一旦不满足条件则不考虑后面的项 , 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
LastN = List.LastN(list,1)           //输出   {10}
LastN = List.LastN(list,5)           //输出   {18,17,10}
LastN = List.LastN(list,each _<>null) //输出   {40,19,18,17,10}
```

31. List.Range

函数 : List.Range

语法 : List.Range(**list** as list, **offset** as number, *optional* **count** as nullable number) as list

说明 : 从 list 列表的偏移位置 offset 开始获取 list 列表, 如果指定可选参数 count, 则返回对应项数的 list 列表, 位置从 0 开始, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
Range = List.Range(list,3)           //输出   {45,-1,40,19,18,17,10}
Range = List.Range(list,3,4)         //输出   {45,-1,40,19}
```

32. List.Single

函数 : List.Single

语法 : List.Single(**list** as list) as any

说明 : 如果 list 列表只有一个值, 则返回该值, 否则引发异常报警, 例如 :

```
Single = List.Single({32})           //输出   32
```

33. List.SingleOrDefault

函数 : List.SingleOrDefault

语法 : List.SingleOrDefault(**list** as list, *optional* **default** as any) as any

说明 : 如果 list 列表只有一个值, 则返回该值; 如果列表为空, 则返回 null, 可以指定返回可选项 default 的值; 如果 list 列表有多个项目, 将引发异常报警, 例如 :

```
SingleOrDefault = List.SingleOrDefault({32})           //输出   32
SingleOrDefault = List.SingleOrDefault({})           //输出   null
SingleOrDefault = List.SingleOrDefault({},100)         //输出   100
```

34. List.FindText

函数 : List.FindText

语法 : List.FindText(**list** as list, **text** as text) as list

说明 : 从包含值 text 的 list 列表中, 返回新的 list 列表, 仅支持文本数据类型, 区分大小写, 例如 :

```
list = {"abc","bc","Ab","CbA","ca","CD","PA"},
FindText = List.FindText(list,"a")           //输出   {"abc","ca"}
```

```
FindText = List.FindText(list,"A")           //输出   {"Ab","CbA","PA"}
```

35. List.Alternate

函数：List.Alternate

语法：List.Alternate(**lists** as list, **count** as number, *option* **repearInterval** as nullable number, *optional* **offset** as nullable number) as list

说明：若不指定可选项，则返回 list 列表中跳过 count 个项目后的新 list 列表；指定的 repearInterval 表示需要获取的值；offset 表示执行此过程的起始位置，位置从 0 开始，例如：

```
list = {1..50},  
Alternate = List.Alternate(list,10,2,1)      //输出   {1,22.23,44,45}  
Alternate = List.Alternate(list,10,1,3)      //输出   {1,2,3,13,23,33,43}  
Alternate = List.Alternate(list,9,1)         //输出   {10,20,30,40,50}
```

36. List.Sort

函数：List.Sort

语法：List.Sort(**list** as list, *optional* **comparisonCriteria** as any) as list

说明：根据指定的可选条件对数据列表 list 排序，例如：

comparisonCriteria 参数说明如表 A-57 所示，默认升序排序。

表 A-57

参 数	说 明	值
Order.Ascending	升序	0
Order.Descending	降序	1

```
list = {32,56,null,45,10,40,19,null,17,null},  
Sort = List.Sort(list)           //输出   {null,null,null,10,17,19,32,40,45,56}  
Sort = List.Sort(list,1)         //输出   {56,45,40,32,19,17,10,null,null,null}
```

37. List.Distinct

函数：List.Distinct

语法：List.Distinct(**list** as list, *optional* **equationCriteria** as any) as list

说明：保留 list 中的所有项且删除重复的内容，如果列表为空，则返回空列表，例如：

```
list = {32,56,null,45,10,40,19,null,17,null},  
Distinct = List.Distinct(list)      //输出   {32,56,null,45,10,40,19,17}
```

38. List.Positions

函数：List.Positions

语法 : List.Positions(**list** as list) as list

说明 : 返回 list 列表每个偏移量的列表, 位置从 0 开始, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
Positions = List.Positions(list)           //输出   {0,1,2,3,4,5,6,7,8,9,10}
```

39. List.ReplaceRange

函数 : List.ReplaceRange

语法 : List.ReplaceRange(**list** as list, **index** as number, **values** as list) as list

说明 : 在 list 列表的 index 位置开始, 使用 replaceWith 列表替换 list 列表中的 count 个项目的值。

```
list = {32,56,58,45,-1,40,19,18,17,10},
ReplaceRange = List.ReplaceRange(list,2,3,{"A".. "D"})
                输出      {32,56,"A","B","C","D",40,19,18,17,10}
```

40. List.InsertRange

函数 : List.InsertRange

语法 : List.InsertRange(**list** as list, **index** as number, **values** as list) as list

说明 : 在 list 列表的 index 位置插入新的 list 列表来生成一个新的 list 列表, 位置从 0 开始, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
InsertRange = List.InsertRange(list,3,{"A","B","C"})
                //输出   {32,56,58,"A","B","C",45,-1,40,19,18,17,10}
InsertRange = List.InsertRange({"A".. "D"},2,{0..3})
                //输出      {"A","B",0,1,2,3,"C","D"}
```

41. List.RemoveRange

函数 : List.RemoveRange

语法 : List.RemoveRange(**list** as list, **index** as number, **count** as nullable number) as list

说明 : 在 list 列表的 index 位置开始, 删除 count 个项目; 如果不指定 count, 则默认删除 1 个项目, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
RemoveRange = List.RemoveRange(list,3)
                //输出   {32,56,58,-1,40,19,18,17,10}
RemoveRange = List.RemoveRange(list,3,5)           //输出   {32,56,58,17,10}
```

42. List.RemoveFirstN

函数 : List.RemoveFirstN

语法 : List.RemoveFirstN(**list** as list, *optional* **countOrCondition** as any) as list

说明 : 在 list 列表中删除前面若干项目的列表, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},  
RemoveFirstN = List.RemoveFirstN(list,each _>0)  
//输出 {-1,40,19,18,17,10}
```

43. List.RemoveLastN

函数 : List.RemoveLastN

语法 : List.RemoveLastN(**list** as list, *optional* **countOrCondition** as any) as list

说明 : 在 list 列表中删除后面若干项目的列表, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},  
RemoveLastN = List.RemoveLastN(list,3)  
//输出 {32,56,58,45,-1,40,19}  
RemoveLastN = List.RemoveLastN(list,each _>0) //输出 {32,56,58,45,-1}
```

44. List.RemoveMatchingItems

函数 : List.RemoveMatchingItems

语法 : List.RemoveMatchingItems(**list1** as list, **list2** as list, *optional* **equationCriteria** as any) as list

说明 : 从 list1 列表中删除 list2 列表中出现的的所有项目。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},  
RemoveMatchingItems = List.RemoveMatchingItems(list,{-1..30})  
//输出 {32,56,58,45,40}
```

45. List.RemoveItems

函数 : List.RemoveItems

语法 : List.RemoveItems(**list1** as list, **list2** as list) as list

说明 : 从 list1 列表中删除 list2 列表中出现的给定值, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},  
RemoveItems = List.RemoveItems(list1,{-1..30})  
//输出 {32,56,58,45,40}
```

46. List.RemoveNulls

函数 : List.RemoveNulls

语法 : List.RemoveNulls(**list** as list) as list

说明 : 从 list 列表中删除所有 null 值, 例如 :

```
list = {32,56,null,45,10,40,19,null,17,null},
RemoveNulls = List.RemoveNulls(list)
//输出 {32,56,45,10,40,19,17}
```

47. List.Skip

函数 : List.Skip

语法 : List.Skip(**list** as list, *optional* **countOrCondition** as any) as list

说明 : 返回跳过 list 列表第一个后的列表, 如果 List 为空列表, 则返回空列表。若指定 countOrCondition 是一个数, 则最多跳过多少项, 如果指定的是条件, 则返回的表以 list 满足条件的第一个元素开头, 一旦条件不满足, 就不再考虑后面的其他项目, 例如 :

```
list = {32,56,58,45,-1,40,19,18,17,10},
Skip = List.Skip(list,6) //输出 {19,18,17,10}
Skip = List.Skip(list,each _>0) //输出 {-1,40,19,18,17,10}
```

48. List.Transform

函数 : List.Transform

语法 : List.Transform(**list** as list, **transform** as function) as list

说明 : 通过将转换函数 transform 应用到 list 列表, 来获得新的 list 列表, 例如 :

```
list = {32,56,58,45,-1,40},
Transform = List.Transform(list,each _*10)
//输出 {320,560,580,450,-10,400}
Transform = List.Transform(list,Text.From)
//输出 {"32","56","58","45","1","40"}
Transform = List.Transform(list,each Number.ToText(_, "0 元"))
//输出 {"32 元","56 元","58 元","45 元","1 元","40 元"}
Transform = List.Transform(list,each Text.End(Text.From(_),1))
//输出 {"2","6","8","5","1","0"}
Transform = List.Transform(list,each Number.Abs(Number.Mod(_,10)))
//输出 {2,6,8,5,1,0}
```

49. List.Repeat

函数 : List.Repeat

语法 : List.Repeat(**list** as list, **count** as number) as list

说明 : 将 list 列表重复 count 次创建新的 list 列表。

```
list = {"a","b","c"},
Combine = List.Combine(list,3)
//输出 {"a","b","c","a","b","c","a","b","c"}
```

50. List.Reverse

函数 : List.Reverse

语法 : List.Reverse(**list** as list) as list

说明 : 将 list 列表逆序后创建新的 list 列表, 例如 :

```
list = {32,56,58,45,-1,40},
Reverse = List.Reverse(list)           //输出   {40,-1,45,58,56,32}
```

51. List.ReplaceMatchingItems

函数 : List.ReplaceMatchingItems

语法 : List.ReplaceMatchingItems(**list** as list, **replacements** as list, *optional* **equationCriteria** as list) as list

说明 : 将 list 列表中指定的旧值替换成新值, 一个替换操作 replacements 由两个值的列表, 列表中提供的旧值和新值组成{{旧值,新值},{旧值,新值}}, 例如 :

```
list = {32,56,58,45,-1,40},
ReplaceMatchingItems = List.ReplaceMatchingItems(list,{{-1,1},{40,400}})
//输出   {32,56,58,45,1,400}
```

52. List.ReplaceValue

函数 : List.ReplaceValue

语法 : List.ReplaceValue(**list** as list, **oldValue** as any, **newValue** as any, **replacer** as function) as list

说明 : 将 list 列表中指定的旧值替换成新值, 例如 :

```
list = {"abc","bc","a","CbA","ca","CD","a"},
ReplaceValue = List.ReplaceValue(list,"a","A",Replacer.ReplaceText)
//输出   {"Abc","bc","A","CbA","cA","CD","A"}
ReplaceValue = List.ReplaceValue(list,"a","A",Replacer.ReplaceValue)
//输出   {"abc","bc","A","CbA","ca","CD","A"}
```

Replacer 参数说明如表 A-58 所示。

表 A-58

参 数	说 明	值
Replacer.ReplaceText	替换字符串	0
Replacer.ReplaceValue	替换值	1

53. List.Combine

函数 : List.Combine

语法 : List.Combine(**lists** as list) as list

说明 : 提取列表中的 lists 列表并将它们合并成一个新列表, 例如 :

```
list1 = {"a","b","c"},
list2 = {"A","B","C"},
Combine = List.Combine({list1,list2}) //输出 {"a","b","c","A","B","C"}
Combine = List.Combine({{"A".. "C"},{1..3}}) //输出 {"A","B","C",1,2,3}
```

54. List.Difference

函数 : List.Difference

语法 : List.Difference(**list1** as list, **list2** as list, *optional* **equationCriteria** as any) as list

说明 : 返回未出现在 list2 列表中的 list 列表的项, 支持重复的值, 返回 list1 与 list2 的差集。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list1 = {1,2,2,3,4,5,3},
list2 = {3,4,5,6},
Difference = List.Difference(list1,list2) //输出 {1,2,2,3}
```

55. List.Intersect

函数 : List.Intersect

语法 : List.Intersect(**lists** as list, *optional* **equationCriteria** as any) as list

说明 : 返回 lists 中所有项目的交集的 list 列表。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list1 = {1,2,2,3,4,5,3},
list2 = {3,4,5,6},
Intersect = List.Intersect({list1,list2}) //输出 {3,4,5}
```

56. List.Union

函数 : List.Union

语法 : List.Union(**lists** as list, *optional* **equationCriteria** as any) as list

说明 : 返回 lists 中所有项目的合集的 list 列表。可以指定可选参数 equationCriteria 来控制相等测试, 例如 :

```
list1 = {1,2,2,3,4,5,3},
list2 = {3,4,5,6},
Union = List.Union({list1,list2}) //输出 {1,2,2,3,4,5,3,6}
```

57. List.Datetimes

函数 : List.Datetimes

语法 : List.Datetimes(**start** as datetime, **count** as number, **step** as duration) as list

说明 : 返回从 start 为起始日期, 间隔时间为 duration, 总共 count 个日期时间的 list 列表。

输出 : 如下代码所示, 获得从 2016/10/1 12:34:56 开始, 每次间隔 1 天的 10 个日期组成的 list 列表。

```
= List.Datetimes(#datetime(2016,10,1,12,34,56),10,#duration(1,0,0,0))
```

输出 : 如下代码所示, 获得从 2016/10/1 12:34:56 开始, 每次间隔 12 小时的 10 个日期的 list 列表。

```
= List.Datetimes(#datetime(2016,10,1,12,34,56),10,#duration(1,0,0,0))
```

58. List.Dates

函数 : List.Dates

语法 : List.Dates(**start** as date, **count** as number, **step** as duration) as list

说明 : 返回从 start 为起始日期, 间隔时间为 duration, 总共 count 个日期时间的 list 列表。

输出 : 如下代码所示, 获得从 2016/10/1 12:34:56 开始, 每次间隔 1 天的 10 个日期组成的 list 列表。

```
= List.Dates(#date(2016,10,1),10,#duration(1,0,0,0))
```

输出 : 如下代码所示, 获得从 2016/10/1 12:34:56 开始, 每次间隔 1.5 天的 10 个日期的 list 列表。

```
= List.Dates(#date(2016,10,1),10,#duration(1,12,0,0))
```

59. List.Times

函数 : List.Times

语法 : List.Times(**start** as time, **count** as number, **step** as duration) as list

说明 : 返回从 start 为起始日期, 间隔时间为 duration, 总共 count 个日期时间的 list 列表。

输出 : 获得从 8:00:00 开始, 每次间隔 1 小时的 10 个时间组成的 list 列表。

```
= List.Times(#time(8,0,0),10,#duration(0,1,0,0))
```

输出 : 如下代码所示, 获得从 8:00:00 开始, 每次间隔 1.5 小时的 10 个日期的 list 列表。

```
= List.Times(#time(8,0,0),10,#duration(0,1,30,0))
```

60. List.Durations

函数 : List.Durations

语法 : List.Durations(**start** as duration, **count** as number, **step** as duration) as list

说明 : 返回从 start 为起始日期, 间隔时间为 duration, 总共 count 个日期时间的 list 列表, 例如 :

```
= List.Durations(#duration(0,1,0,0),10,#duration(0,1,0,0))
```

61. List.Numbers

函数 : List.Numbers

语法 : List.Numbers(**start** as number, **count** as number, *optional increment* as nullable number) as list

说明 : 给定初始值, 计数和可选的增量值来返回 list, 增量值默认为 1, 例如 :

输出 : 如下代码所示, 获得从 1 开始, 每次间隔为 1 的 10 个数。

```
= List.Numbers(1,10)
```

输出 : 如下代码所示, 获得从 1 开始, 每次间隔为 0.3 的 10 个数。

```
= List.Numbers(1,10,0.3)
```

62. List.Random

函数 : List.Random

语法 : List.Random(**count** as number, *optional seed* as nullable number) as list

说明 : 创建随机值 list 列表。

输出 : 如下代码所示, 创建 5 个介于 0~1 的随机值。

```
= List.Random(5)
```

输出 : 如下代码所示, 创建每次都相同以 seed 为种子的 5 个随机值。

```
= List.Random(5,2)
```

A.10 Table 类函数

Table 类函数介绍如表 A-59 所示。

表 A-59

类	分 类	函 数 名	概 述
1	转换	Table.ToRows	转换成行的 list 表格
2	创建	Table.FromRows	从多行 list 创建表格
3	转换	Table.ToColumns	转换成列的 list 表格
4	创建	Table.FromColumns	从多列 list 创建表格
5	转换	Table.ToList	使用指定符号合成 list 列表

Powery Query：用 Excel 玩转商业智能数据处理

6	创建	Table.FromList	从 list 创建表格
7	转换	Table.ToRecords	转换成记录表
8	创建	Table.FromRecords	从 Record 创建表格

续表

类	分 类	函 数 名	概 述
9	创建	Table.FromValue	从值创建表格
10	创建	Table.Repeat	创建重复次数的表
11	创建	Table.Column	返回表格指定列的 list 列表
12	创建	Table.ColumnNames	返回表格的列名称 list 列表
13	计算	Table.RowCount	统计表的行数
14	计算	Table.ColumnCount	计算表的列数
15	判断	Table.IsEmpty	判断表格是否为空表
16	判断	Table.Contains	判断表中指定列是否包含某值
17	判断	Table.ContainsAll	判断是否包含比较列的所有行
18	判断	Table.ContainsAny	判断是否包含比较列的部分行
19	判断	Table.MatchesAllRows	检测是否所有行都满足条件
20	判断	Table.MatchesAnyRows	检测是否部分行都满足条件
21	判断	Table.HasColumns	判断是否包含某些列字段
22	选择	Table.First	返回第一行
23	选择	Table.FirstN	返回前几行
24	选择	Table.Last	返回最后一行
25	选择	Table.LastN	返回后几行
26	选择	Table.Min	返回指定列具有最小值的行
27	选择	Table.MinN	返回前几个最小值的所有行
28	选择	Table.Max	返回指定列具有最大值的行
29	选择	Table.MaxN	返回前几个最大值的所有行
30	选择	Table.SelectRowsWithErrors	选择包含错误的行
31	选择	Table.SingleRow	返回一行表格的单一行，若有多行则返回错误
32	选择	Table.Range	从指定位置开始选择指定行的表格
33	选择	Table.AlternateRows	返回间隔规则行数的表格
34	选择	Table.Skip	不包含前几行或前面指定条件的表格
35	选择	Table.FindText	返回包含文本值的行的表格
36	选择	Table.SelectRows	根据指定条件选择行
37	选择	Table.SelectColumns	仅选择某些列
38	操作	Table.Sort	排序

39	操作	Table.ReverseRows	逆序表格
40	操作	Table.ReplaceRows	替换指定行为新的表行
41	操作	Table.ReplaceValue	替换值
42	操作	Table.ReplaceErrorValues	替换错误值

续表

类	分 类	函 数 名	概 述
43	操作	Table.ReplaceMatchingRows	替换所有指定条件的行
44	操作	Table.InsertRows	插入表行
45	操作	Table.Distinct	删除重复项
46	操作	Table.RemoveRowsWithErrors	删除有错误值的行
47	操作	Table.RemoveFirstN	删除前面若干行的表格
48	操作	Table.RemoveLastN	删除后面若干行的表格
49	操作	Table.RemoveRows	从指定位置开始删除指定行的表格
50	操作	Table.RemoveMatchingRows	删除所有指定条件的行
51	操作	Table.RemoveColumns	删除指定的某些列
52	操作	Table.RenameColumns	重命名列标题
53	操作	Table.AddColumn	添加新列
54	操作	Table.AddIndexColumn	添加索引列
55	操作	Table.PromoteHeaders	将第一行升级为标题
56	操作	Table.DemoteHeaders	将标题降为第一行值
57	操作	Table.CombineColumns	合并列
58	操作	Table.SplitColumn	拆分列
59	操作	Table.FillDown	向下填充
60	操作	Table.FillUp	向上填充
61	操作	Table.Transpose	转置
62	判断	Table.PositionOf	查找筛选一个条件的行在表格中的位置
63	判断	Table.PositionOfAny	查找筛选任意条件的行在表格中的位置
64	操作	Table.Pivot	透视列
65	操作	Table.Unpivot	逆透视列
66	操作	Table.UnpivotOtherColumns	逆透视其他列
67	集合	Table.Group	分组依据, 分组统计
68	集合	Table.Combine	追加查询合并所有表
69	集合	Table.Join	合并查询
70	集合	Table.NestedJoin	合并查询
71	集合	Table.AddJoinColumn	输出类似于外连接的合并查询

Powery Query：用 Excel 玩转商业智能数据处理

72	集合	Table.AggregateTableColumn	聚合多个列
73	操作	Table.ExpandTableColumn	展开表格
74	操作	Table.ExpandListColumn	纵向展开表格中的 list
75	操作	Table.ExpandRecordColumn	横向展开表格中的 Record

若非特殊说明，本节的案例将全部使用查询名称为【销售】的查询表为基础表格，如图 A-17 所示。

查询 [1]	Excel.CurrentWorkbook() [Name="表1"] [Content]	查询设置
销售		
1	CF04518 运动户外 冲锋衣 哥伦比亚防水三合一冲...	5
2	CF04518 运动户外 冲锋衣 哥伦比亚三合一冲锋衣两件...	19
3	CF22179 运动户外 冲锋衣 软壳高领防风三合一冲锋衣...	94
4	D540015 运动户外 登山鞋 诺诗兰 Goretex 男式三穿...	64
5	D531021 运动户外 登山鞋 LOWA 男式户外防水登山...	98
6	ZK05179 运动户外 自行车 永久山地车变速 26 寸铝合...	12
7	ZK31887 运动户外 自行车 禧玛诺 16 寸女式 7 速变速轻...	24
8	8B14578 箱包 背包 SPRAYGROUND 背包	24
9	TB31475 箱包 提包 ELLE 5 层牛皮女包	54
10	TB18669 箱包 提包 FICH 5 层牛皮印花女包	6
11	S100017 箱包 双肩包 JanSport 电脑双肩背包	26
12	S191250 箱包 双肩包 JanSport 双肩背包	8
13	X000174 电器 相机 索尼 HX5-5TL 微单	66
14	X031547 电器 相机 卡西欧 ZR1500	79
15	X044587 电器 相机 富士拍立得 mini8	55
16	DN94782 电器 平板电脑 IPAD MINI 16G WIFI	25
17	DN51257 电器 平板电脑 IPAD AIR2 64G WIFI	59
18	S815841 饰品 手表 三宅一生水元素理念手链...	86
19	ZB34587 饰品 珠宝 周生生足金 Charm 皇冠吊...	86

图 A-17

1. Table.ToRows

函数：Table.ToRows

语法：Table.ToRows(**table** as table) as list

说明：从表 table 中创建嵌套表 list 列表，每个 list 列表都包含行值的内部 list 列表，例如：

```
let
    源 = 销售,
    ToRows = Table.ToRows(源)
in
    ToRows
```

为了节省篇幅，本节涉及的公式都只写 let 与 in 之间的过程代码。

输出结果为一个与表格相同行数的 list 列表，每个 list 项目是表格中的每一行数据组成的内部 list 列表。选择每个内部 list 列表，可以看到每个 list 的内部预览项目，如图 A-18 所示。

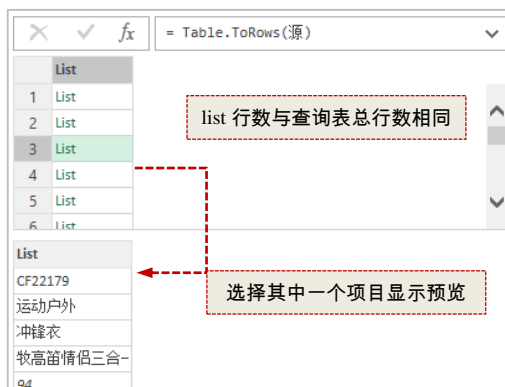


图 A-18

2. Table.FromRows

函数：Table.FromRows

语法：Table.FromRows(**list** as list, *optional columns* as any) as table

说明：从 list 列表创建一个查询表，其中该列表的每个元素都是一个包含用于单行的列值的内部列表；可以提供可选的字段标题或数据类型作为第二参数，若不指定，则字段标题以 Column1、Column2 的样式设置。此步骤数据源使用上述 TableToRows 的结果，例如：

```
源 = TableToRows,
FromRows=Table.FromRows(源,{ "编码", "品类", "大类", "商品名称", "单价" })
```

输出结果是以每个 list 元素顺序展开为一行的查询表，行数与 List 列表相同，如图 A-19 所示

ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1	CF04518		运动户外		冲锋衣		哥伦步防风防...		5
2	CF04518		运动户外		冲锋衣		哥伦步三合一...		19
3	CF22179		运动户外		冲锋衣		牧高笛情侣三...		94
4	DS40015		运动户外		登山鞋		诺诗兰 Gore-tex...		64
5	DS31021		运动户外		登山鞋		LOWA男式户外...		98
6	ZX05179		运动户外		自行车		永久山地车赛...		12
7	ZX31887		运动户外		自行车		福特16寸女式7...		24
8	BB14578		箱包		背包		SPRAYGROUND...		24

图 A-19

可以指定列字段的标题和数据类型，例如：

```
源 = TableToRows,
FromRows = Table.FromRows(源,type table [编码=text,品类=text,大类=text,
商品名称=text,数量=number])
```

结果如图 A-20 所示。

Powery Query：用 Excel 玩转商业智能数据处理

	A ^B _C 编码	A ^B _C 品类	A ^B _C 大类	A ^B _C 商品名称	1.2 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-te...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24

图 A-20

如果不指定字段标题，则以 Column1 和，Column2 的样式设置标题，例如

```
源 = TableToRows,
FromRows = Table.FromRows(源)
```

结果如图 A-21 所示。

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-te...	64
5	DS31021	运动户外	登山鞋	LOWA男式户...	98
6	ZX05179	运动户外	自行车	永久山地车变...	12
7	ZX31887	运动户外	自行车	福特16寸女式...	24
8	BB14578	箱包	背包	SPRAYGROUN...	24

6-21

3. Table.ToColumns

函数：Table.ToColumns

语法：Table.ToColumns(**table** as table) as list

说明：从表 table 中创建嵌套的 list 列表，每个 list 列表都包含列值的内部 list 列表，例如：

```
源 = 销售,
ToColumns=Table.ToColumns(源)
```

输出结果为一个与表格相同列数的 list 列表，每个 list 项目是表格中的每一列数据组成的内部 list 列表。选择每个内部 list 列表，可以看到每个 list 的内部预览项目，如图 A-22 所示。

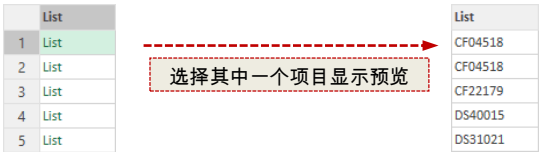


图 A-22

4. Table.FromColumns

函数 : Table.FromColumns

语法 : Table.FromColumns(**list** as list, *optional columns* as any) as table

说明 : 从 list 列表创建一个查询表, 其中该列表的每个元素都是一个包含用于单一列的行值的内部列表; 可以提供可选的字段标题或数据类型作为第二参数, 若不指定, 则字段标题以 Column1、Column2 的样式设置。如下面的代码所示, 此步骤数据源使用上述 TableToColumns 的结果。

```
源 = TableToColumns,
FromColumns=Table.FromColumns(源,{ "编码", "品类", "大类", "商品名称", "单价" })
```

输出结果是以每个 list 元素顺序展开为一列的查询表, 列数与 List 列表相同, 如图 A-23 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24

图 A-23

可以指定列字段的标题和数据类型, 例如:

```
源 = TableToColumns,
FromColumns = Table.FromColumns(源, type table [编码=text, 品类=text,
        大类=text, 商品名称=text, 数量=number])
```

结果如图 A-24 所示。

	A ^B _C 编码	A ^B _C 品类	A ^B _C 大类	A ^B _C 商品名称	1.2 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24

图 A-24

如果不指定字段标题, 则以 Column1、Column2 的样式设置标题:

Powery Query：用 Excel 玩转商业智能数据处理

```
源 = TableToColumns,
FromColumns = Table.FromColumns(源)
```

结果如图 A-25 所示。

	ABC 123	Column1	ABC 123	Column2	ABC 123	Column3	ABC 123	Column4	ABC 123	Column5
1	CF04518	运动户外	冲锋衣	哥伦步防风防...		5				
2	CF04518	运动户外	冲锋衣	哥伦步三合一...		19				
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...		94				
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-te...		64				
5	DS31021	运动户外	登山鞋	LOWA男式户...		98				

图 A-25

5. Table.ToList

函数：Table.ToList

语法：Table.ToList(**table** as table, *optional combiner* as nullable as function) as list

说明：通过给定的可选组合函数 combiner，将查询表中的每一行值转换成 list 列表；若不指定分隔符，则默认以逗号分隔各列数据。此函数仅支持文本数据类型，如果字段是数字类型，则需要先转换数据类型，例如：

```
源 = 销售,
ToList = Table.ToList(源)
```

直接使用此公式，会发生异常警报，提示数字类型字段不能转换；如图 A-26 所示。

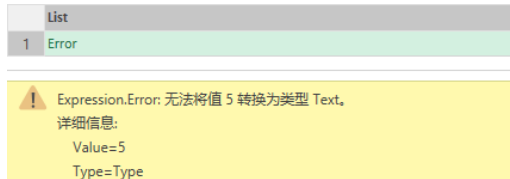


图 A-26

因此，需要增加一个 Table 函数转换字段的数据类型。以下函数将指定字段设置为文本类型，后面会有更详细的介绍，例如：

```
源 = 销售,
ToList = Table.ToList(Table.TransformColumnTypes(源,{{"数量", type text}}))
```

结果如图 A-27 所示。

	List
1	CF04518-运动户外-冲锋衣-哥仑步防风防雨三合一冲锋衣两件套-5
2	CF04518-运动户外-冲锋衣-哥仑步三合一冲锋衣两件套透气防风防雨-19
3	CF22179-运动户外-冲锋衣-牧高笛情侣三合一冲锋衣-94
4	DS40015-运动户外-登山鞋-诺诗兰 Gore-tex男式三穿冲锋衣-64
5	DS31021-运动户外-登山鞋-LOWA男式户外防水登山中帮鞋-98
6	ZX05179-运动户外-自行车-永久山地车变速26寸铝合金架双碟刹减震前叉单车-12
7	ZX31887-运动户外-自行车-福特16寸女式7级变速轻便携折叠自行车-24
8	BB14578-箱包-背包-SPRAYGROUND背包-24

图 A-27

使用 combiner 函数添加分隔符，例如：

```
源 = 销售,
ToList = Table.ToList(Table.TransformColumnTypes(源,{{"数量",
type text}}), Combiner.CombineTextByDelimiter("-"))
```

结果如图 A-28 所示。

	List
1	CF04518-运动户外-冲锋衣-哥仑步防风防雨三合一冲锋衣两件套-5
2	CF04518-运动户外-冲锋衣-哥仑步三合一冲锋衣两件套透气防风防雨-19
3	CF22179-运动户外-冲锋衣-牧高笛情侣三合一冲锋衣-94
4	DS40015-运动户外-登山鞋-诺诗兰 Gore-tex男式三穿冲锋衣-64

图 A-28

6. Table.FromList

函数：Table.FromList

语法：Table.FromList(**table** as table, *optional* **Splitter** as nullable as function, *optional* **columns** as any, *optional* **default** as any, *optional* **extraValues** as nullable ExtraValues.Type) as table

说明：通过给定的可选拆分函数 splitter，将查询表中的每一行值转换成 list 列表；若不指定分隔符，则默认以逗号分隔各列数据，返回结果字段数据类型全部是文本。可选的 columns 可以指定字段标题名称和字段数据类型，还可以指定可选的 default 和 extraValues。此步骤数据源使用上述 TableToList 的结果，例如：

```
源 = TableToList,
FromList = Table.FromList(源)
```

直接使用此公式，默认以逗号为分隔符拆分到列，但如果 Table.ToList 指定的分隔符不是空格，则将不会将每行数据拆分，如图 A-29 所示。

Powery Query：用 Excel 玩转商业智能数据处理

	ABC 123 Column1
1	CF04518-运动户外-冲锋衣-哥仑步防风防雨三合一-冲锋衣两件套-5
2	CF04518-运动户外-冲锋衣-哥仑步三合一-冲锋衣两件套透气防风防雨-19
3	CF22179-运动户外-冲锋衣-牧高笛情侣三合一-冲锋衣-94
4	DS40015-运动户外-登山鞋-诺诗兰 Gore-tex男式三穿冲锋衣-64
5	DS31021-运动户外-登山鞋-LOWA男式户外防水登山中帮鞋-98
6	ZX05179-运动户外-自行车-永久山地车变速26寸铝合金架双碟刹减震前叉单车-12
7	ZX31887-运动户外-自行车-福特16寸女式7级变速轻便携折叠自行车-24
8	BB14578-箱包-背包-SPRAYGROUND背包-24
9	TB31475-箱包-提包-ELLE头层牛皮女包-54

图 A-29

使用 Splitter 拆分器函数添加分隔符，并指定字段名称。

```
源 = TableToList,
FromList = Table.FromList(源, Splitter.SplitTextByDelimiter("-"),
    {"编码", "品类", "大类", "商品名称", "单价"})
```

结果如图 A-30 所示

	A ^B _C 编码	A ^B _C 品类	A ^B _C 大类	A ^B _C 商品名称	A ^B _C 单价
1	CF04518	运动户外	冲锋衣	哥仑步防风防雨...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变速...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24
9	TB31475	箱包	提包	ELLE头层牛皮女...	54

图 A-30

下面的代码使用 Splitter 拆分器函数添加分隔符，并指定字段名称和数据类型。

```
源 = TableToList,
FromList = Table.FromList(源, Splitter.SplitTextByDelimiter("-"),
    type table [编码=text, 品类=text, 大类=text, 商品名称=text, 数量=number])
```

结果如图 A-31 所示。

	A ^B _C 编码	A ^B _C 品类	A ^B _C 大类	A ^B _C 商品名称	1.2 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防雨...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变速...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24
9	TB31475	箱包	提包	ELLE头层牛皮女...	54

图 A-31

如果不指定字段标题，则以 Column1、Column2 的样式设置标题，例如：

```
源 = TableToList,
FromList = Table.FromList(源, Splitter.SplitTextByDelimiter("-"))
```

结果如图 A-31 所示。

	A _C ^B Column1	A _C ^B Column2	A _C ^B Column3	A _C ^B Column4	A _C ^B Column5
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变...	12
7	ZX31887	运动户外	自行车	福特16寸女式7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24
9	TB31475	箱包	提包	ELLE头层牛皮...	54

图 A-31

7. Table.ToRecords

函数：Table.ToRecords

语法：Table.ToRecords(**table** as table) as list

说明：从表 table 中创建嵌套 record 的 list 列表，每个 list 列表都包含每一行的 record 记录。

```
源 = 销售,
ToRecords=Table.ToRecords(源)
```

输出结果为一个与表格具有相同行数的 list 列表，每个 list 项目是表格中的每一行数据组成的内部 record 记录。选择每个内部 record 记录，可以看到每个 list 的内部预览项目，如图 A-33 所示。

List	
1	Record
2	Record
3	Record
4	Record
5	Record

选择其中一个项目显示预览

编码	CF04518
品类	运动户外
大类	冲锋衣
商品名称	哥仑步防风防雨三合一冲锋衣两件套
数量	5

图 A-33

8. Table.FromRecords

函数：Table.FromRecords

语法：Table.FromRecords(**records** as list, *optional columns* as any) as table

说明：将记录列表 records 转换为查询表，可选的 columns 可以额外修改原来的字段标题名称。

Powery Query：用 Excel 玩转商业智能数据处理

此步骤数据源使用上述 TableToRecord 的结果，例如：

```
源 = TableToRecords,  
FromRecords = Table.FromRecords(源)
```

输出结果是以每个 list 元素顺序展开为一列的查询表，列数与 List 列表相同，如图 A-34 所示。

	源. 编码	源. 品类	源. 大类	源. 商品名称	源. 单价
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨...	5
2	CF04518	运动户外	冲锋衣	哥伦步三合一冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变速...	12
7	ZX31887	运动户外	自行车	福特 16 寸女式 7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24
9	TB31475	箱包	提包	ELLE 头层牛皮女...	54

图 A-34

重新修改字段数据类型：

```
源 = TableToRecords,  
FromRecords = Table.FromRecords(源, type table [编码=text, 品类=text,  
    大类=text, 商品名称=text, 数量=number])
```

结果如图 A-35 所示

	源. 编码	源. 品类	源. 大类	源. 商品名称	1.2 数量
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨...	5
2	CF04518	运动户外	冲锋衣	哥伦步三合一冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
5	DS31021	运动户外	登山鞋	LOWA男式户外...	98
6	ZX05179	运动户外	自行车	永久山地车变速...	12
7	ZX31887	运动户外	自行车	福特 16 寸女式 7...	24
8	BB14578	箱包	背包	SPRAYGROUND...	24

图 A-35

9. Table.FromValue

函数：Table.FromValue

语法：Table.FromValue(Value as any, optional options as nullable record) as table

说明：将包含值的 list 列表转换为查询表，可选记录参数 options 可以指定为控制选项

DefaultColumnName 设置字段标题名称。

例如：

```
源 = TableToList,  
FromValue = Table.FromValue(1)
```

如图 A-36 所示。

1.2 Value
1 1

图 A-36

```
FromValue = Table.FromValue("A")
```

如图 A-37 所示。

A ^B _C Value
1 A

6-37

```
FromValue = Table.FromValue({1..5},[DefaultColumnName="number"])
```

如图 A-38 所示

ABC 123	number
1	1
2	2
3	3
4	4
5	5

图 A-38

使用前述函数 Table.ToRows 结果作为数据源：

```
FromValue = Table.FromValue(Table.ToRows)
```

如图 A-39 所示。

ABC 123	Value
1	List
2	List
3	List
4	List
5	List

选择其中一个项目显示预览

List
CF04518
运动户外
冲锋衣
哥仑步防风防雨三合一
5

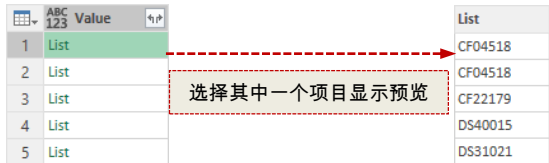
图 A-39

使用前述函数 Table.ToColumns 结果作为数据源：

```
FromValue = Table.FromValue(Table.ToColumns)
```

如图 A-40 所示。

Power Query: 用 Excel 玩转商业智能数据处理



ABC 123	Value
1	List
2	List
3	List
4	List
5	List

选择其中一个项目显示预览

List
CF04518
CF04518
CF22179
DS40015
DS31021

图 A-40

10. Table.Repeat

函数 : Table.Repeat

语法 : Table.Repeat(**table** as table, **count** as number) as table

说明 : 将 table 重复 count 次创建一个新的 table 查询表 , 例如 :

```
源 = Table.FromRecords({[产品 ID = 1023, 价格 = 4], [产品 ID= 1024, 价格 = 5],  
    [产品 ID = 1025, 价格 = 6], [产品 ID = 1026, 价格 = 7]}),  
Repeat = Table.Repeat(源, 2)
```

如图 A-41 所示。

ABC 123	产品ID	ABC 123	价格
1	1023	4	
2	1024	5	
3	1025	6	
4	1026	7	

ABC 123	产品ID	ABC 123	价格
1	1023	4	
2	1024	5	
3	1025	6	
4	1026	7	
5	1023	4	
6	1024	5	
7	1025	6	
8	1026	7	

图 A-41

11. Table.Column

函数 : Table.Column

语法 : Table.Column(**table** as table, **Column** as txtst) as list

说明 : 返回 table 中由 column 指定的数据列返回的 list 列表 , 例如

```
Column = Table.Column(销售, "商品名称")
```

此函数可以直接通过深化获得相同结果 , 如图 A-42 所示。

```
=源[商品名称]
```

	List
1	哥伦步防风防雨三合一冲锋衣两件套
2	哥伦步三合一冲锋衣两件套透气防风防雨
3	牧高笛情侣三合一冲锋衣
4	诺诗兰 Gore-tex男式三穿冲锋衣
5	LOWA男式户外防水登山中帮鞋
6	永久山地车变速26寸铝合金架双碟刹减震前叉单车
7	福特16寸女式7级变速轻便携折叠自行车
8	SPRAYGROUND背包
9	ELLE头层牛皮女包

图 A-42

12. Table.ColumnNames

函数：Table.ColumnNames
语法：Table.ColumnNames(**table** as table) as list
说明：返回 table 中由所有字段名称组成的 list 列表，例如：

```
源 = 销售，  
ColumnNames = Table.ColumnNames(源)
```

如图 A-43 所示。

	List
1	编码
2	品类
3	大类
4	商品名称
5	数量

图 A-43

13. Table.RowCount

函数：Table.RowCount
语法：Table.RowCount(**table** as table) as number
说明：返回 table 表的行数，例如：

```
源 = 销售，  
RowCount = Table.RowCount(源)           //输出      19
```

14. Table.ColumnCount

函数：Table.ColumnCount
语法：Table.ColumnCount(**table** as table) as number
说明：返回 table 表的列数，例如：

```
源 = 销售,
ColumnCount = Table.ColumnCount(源)           //输出      5
```

15. Table.IsEmpty

函数 : Table.IsEmpty

语法 : Table.IsEmpty(**table** as table) as logical

说明 : 判断表格为空表是返回 TRUE, 否则返回 FALSE, 例如 :

```
源 = 销售,
IsEmpty = Table.IsEmpty(源)                   //输出      FALSE
```

16. Table.Contains

函数 : Table.Contains

语法 : Table.Contains(**table** as table, **row** as record, *optional equationCriteria* as any) as logical

说明 : 判断表格是否包含一条 row 的 record 记录, 如果包含返回 TRUE, 否则返回 FALSE, 例如 :

```
源 = 销售,
Contains = Table.Contains(源, [品类="箱包"])           //输出      TRUE
Contains=Table.Contains(源, [品类="箱包", 大类="背包"]) //输出      TRUE
Contains=Table.Contains(源, [品类="箱包", 大类="手提包"]) //输出      FALSE
```

17. Table.ContainsAll

函数 : Table.ContainsAll

语法 : Table.ContainsAll(**table** as table, **rows** as list, *optional equationCriteria* as any) as logical

说明 : 判断表格是否同时包含多条 rows 的 list 列表记录, 如果包含则返回 TRUE, 否则返回 FALSE, 例如 :

```
源 = 销售,
ContainsAll = Table.ContainsAll(源, {[品类="箱包"]})           //输出      TRUE
ContainsAll = Table.ContainsAll(源, {[品类="箱包"], [大类="背包"]}) //输出      TRUE
ContainsAll = Table.ContainsAll(源, {[品类="箱包"], [数量=5]}) //输出      TRUE
ContainsAll = Table.ContainsAll(源, {[品类="箱包"], [大类="手提包"]}) //输出      FALSE
```

18. Table.ContainsAny

函数 : Table.ContainsAny

语法 : Table.ContainsAny(**table** as table, **rows** as list, *optional equationCriteria* as any) as logical

说明：判断表格是否包含多条 rows 的 list 列表记录中的任意一条，如果包含则返回 TRUE，否则返回 FALSE，例如：

```
源 = 销售,
ContainsAny = Table.ContainsAny(源, {[品类="箱包"], [数量=5]})
//输出 TRUE
ContainsAny = Table.ContainsAny(源, {[编码="ABC"], [数量=94]})
//输出 TRUE
ContainsAny = Table.ContainsAny(源, {[编码="ABC"], [数量=104]})
//输出 FALSE
```

19. Table.MatchesAllRows

函数：Table.MatchesAllRows

语法：Table.MatchesAllRows(**table** as table, **condition** as function) as logical

说明：判断 table 表中所有行是否都满足 condition 条件，如果都满足则返回 TRUE，否则返回 FALSE，例如：

```
源 = 销售,
MatchesAllRows = Table.MatchesAllRows(源, each [数量]>0)
//输出 TRUE
MatchesAllRows = Table.MatchesAllRows(源, each Text.Length([编码])=7)
//输出 TRUE
MatchesAllRows = Table.MatchesAllRows(源, each Text.Contains([大类], "车"))
//输出 FALSE
```

20. Table.MatchesAnyRows

函数：Table.MatchesAnyRows

语法：Table.MatchesAnyRows(**table** as table, **condition** as function) as logical

说明：判断 table 表中任意行是否满足 condition 条件，如果都满足则返回 TRUE，否则返回 FALSE，例如：

```
源 = 销售,
MatchesAnyRows = Table.MatchesAnyRows(源, each [数量]>100)
//输出 FALSE
MatchesAnyRows = Table.MatchesAnyRows(源, each Text.Length([编码])=7)
//输出 TRUE
MatchesAnyRows = Table.MatchesAnyRows(源, each Text.Contains([大类], "车"))
//输出 TRUE
```

21. Table.HasColumns

函数：Table.HasColumns

Powery Query：用 Excel 玩转商业智能数据处理

语法：Table.HasColumns(**table** as table, **columns** as any) as logical

说明：判断 table 表中是否包含指定的所有字段，如果都包含则返回 TRUE，否则返回 FALSE，
例如：

```
源 = 销售,
HasColumns = Table.HasColumns(源,"数量") //输出 FALSE
HasColumns = Table.HasColumns(源,{"数量","名称"}) //输出 TRUE
HasColumns = Table.HasColumns(源,{"编码","数量"}) //输出 TRUE
```

22. Table.First

函数：Table.First

语法：Table.First(**table** as table, *optional default* as any) as any

说明：返回 table 表的第一行，如果表格为空，则返回可选参数 default。

```
源 = 销售,
First = Table.First(源)
```

结果如图 A-44 所示。

编码	CF04518
品类	运动户外
大类	冲锋衣
商品名称	哥伦步防风防雨三合一冲锋衣两件套
数量	5

图 A-44

23. Table.FirstN

函数：Table.FirstN

语法：Table.FirstN(**table** as table, **countOrCondition** as any) as table

说明：返回 table 表的前几行；如果 countOrCondition 是数字，则从顶部开始返回指定行数，若 countOrCondition 是一个条件，则返回满足此条件的行，直到不满足条件为止，例如：

```
源 = 销售,
FirstN = Table.FirstN(源,3)
```

结果如图 A-45 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		CF04518		运动户外		冲锋衣		哥伦步防风防雨三合...		5
2		CF04518		运动户外		冲锋衣		哥伦步三合一冲锋衣...		19
3		CF22179		运动户外		冲锋衣		牧高笛情侣三合一冲...		94

图 A-45

```
FirstN = Table.FirstN(源, each [数量]<50)
```

结果如图 A-46 所示

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨三合...	5
2	CF04518	运动户外	冲锋衣	哥伦步三合一冲锋衣...	19

图 A-46

24. Table.Last

函数：Table.Last

语法：Table.Last(**table** as table, *optional default* as any) as any

说明：返回 table 表的最后一行，如果表格为空，则返回可选参数 default，例如：

```
源 = 销售,
Last = Table.Last(源)
```

结果如图 A-47 所示。

编码	ZB34587
品类	饰品
大类	珠宝
商品名称	周生生足金 Charmé 皇冠串珠
数量	86

图 A-47

```
Last = Table.Last(Table.FromRecords({}), [产品="电水壶", 价格=199])
```

结果如图 A-48 所示。

产品	电水壶
价格	199

图 A-48

25. Table.LastN

函数：Table.LastN

语法：Table.LastN(**table** as table, **countOrCondition** as any) as table

说明：返回 table 表的最后几行；如果 countOrCondition 是数字，则从结束端开始返回指定行数，若 countOrCondition 是一个条件，则返回满足此条件的行，直到不满足条件为止，例如：

```
源 = 销售,
LastN = Table.LastN(源, 5)
```

结果如图 A-49 所示。

Powery Query：用 Excel 玩转商业智能数据处理

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		XJ44587		电器		相机		富士拍立得mini8		55
2		DN94782		电器		平板电脑		IPAD MINI 16G WIFI		25
3		DN51257		电器		平板电脑		IPAD AIR2 64G WIFI		59
4		SB15841		饰品		手表		三宅一生水元素理念...		86
5		ZB34587		饰品		珠宝		周生生足金Charme...		86

图 A-49

```
LastN = Table.LastN(源,each [大类]<>"平板电脑")
```

结果如图 A-50 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		SB15841		饰品		手表		三宅一生水元素理念...		86
2		ZB34587		饰品		珠宝		周生生足金Charme...		86

图 A-50

26. Table.Min

函数：Table.Min

语法：Table.Min(**table** as table, **comparisonCriteria** as any ,*optional default* as any) as any

说明：在给定 comparisonCriteria 的条件下，返回 table 中的最小值行。如果表格为空，则返回可选参数 default，例如：

```
源 = 销售，  
Min = Table.Min(源,"数量")
```

结果如图 A-51 所示。

编码	CF04518
品类	运动户外
大类	冲锋衣
商品名称	哥仑步防风防雨三合一冲锋衣两件套
数量	5

图 A-51

```
Min = Table.Min(源, "编码")
```

结果如图 A-52 所示。

编码	BB14578
品类	箱包
大类	背包
商品名称	SPRAYGROUND背包
数量	24

图 A-52

```
Min = Table.Min(#table({"产品名称"},{}), "产品名称", 100) //输出 100
```

27. Table.MinN

函数：Table.MinN

语法：Table.MinN(**table** as table, **comparisonCriteria** as any, *optional* **countOrCondition** as any) as table

说明：在给定 comparisonCriteria 的条件下，返回 table 中的若干个最小值的行；如果 countOrCondition 是数字，则从结束端开始返回指定行数，若 countOrCondition 是一个条件，则返回满足此条件的行，直到不满足条件为止，例如：

```
源 = 销售,
MinN = Table.MinN(源, "编码", 3)
```

结果如图 A-53 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	BB14578	箱包	背包	SPRAYGROUND背包	24
2	CF04518	运动户外	冲锋衣	哥仑步防风防雨三合...	5
3	CF04518	运动户外	冲锋衣	哥仑步三合一冲锋衣...	19

图 A-53

```
MinN = Table.MinN(源, "数量", each [数量]<20)
```

结果如图 A-54 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防雨三合...	5
2	TB18669	箱包	提包	FION淑女邮戳印花女...	6
3	SJ91250	箱包	双肩包	JanSport双肩背包	8

图 A-54

28. Table.Max

函数：Table.Max

语法：Table.Max(**table** as table, **comparisonCriteria** as any, *optional* **default** as any) as any

说明：在给定 comparisonCriteria 的条件下，返回 table 中的最大值行。如果表格为空，则返回可选参数 default，例如：

```
源 = 销售,
Max = Table.Max(源, "数量")
```

结果如图 A-55 所示。

Powery Query：用 Excel 玩转商业智能数据处理

编码	DS31021
品类	运动户外
大类	登山鞋
商品名称	LOWA男式户外防水登山中帮鞋
数量	98

图 A-55

```
Max = Table.Max(#table({"产品名称"},{}), "产品名称", 100)//输出 100
```

29. Table.MaxN

函数：Table.MaxN

语法：Table.MaxN(**table** as table, **comparisonCriteria** as any ,*optional* **countOrCondition** as any) as table

说明：在给定 comparisonCriteria 的条件下，返回 table 中的若干个最小值的行；如果 countOrCondition 是数字，则从结束端开始返回指定行数，若 countOrCondition 是一个条件，则返回满足此条件的行，直到不满足条件为止，例如：

```
源 = 销售，  
MaxN = Table.MaxN(源, "编码", 3)
```

结果如图 A-56 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	ZX31887	运动户外	自行车	福特16寸女式7级...	24
2	ZX05179	运动户外	自行车	永久山地车变速26...	12
3	ZB34587	饰品	珠宝	周生生足金Charme...	86

图 A-56

```
MaxN = Table.MaxN(源, "数量", each [数量]>80)
```

结果如图 A-57 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	DS31021	运动户外	登山鞋	LOWA男式户外防...	98
2	CF22179	运动户外	冲锋衣	牧高笛情侣三合一...	94
3	SB15841	饰品	手表	三宅一生水元素理...	86
4	ZB34587	饰品	珠宝	周生生足金Charme...	86

图 A-57

30. Table.SelectRowsWithErrors

函数：Table.SelectRowsWithErrors

语法：Table.SelectRowsWithErrors(**table** as table, *optional* **columns** as nullable list) as table

说明：选择 table 查询表中至少一个单元中包含错误的行，如果指定可选参数 columns，则只检

查指定的字段单元格中是否有错误，例如：

```
源 = 销售,
SelectRowsWithErrors = Table.SelectRowsWithErrors(源,{"数量"})
```

结果如图 A-58 所示。



图 A-58

31. Table.SingleRow

函数：Table.SingleRow

语法：Table.SingleRow(**table** as table) as record

说明：选择一行 table 查询表的唯一行。如果 table 有多行，则引发错误警告，例如：

```
源 = 销售,
SingleRow = Table.SingleRow(源)
```

结果如图 A-59 所示。



图 A-59

```
SingleRow = Table.SingleRow(Table.FromRecords({[a=1,b=2]}))
```

结果如图 A-60 所示。

a	1
b	2

图 A-60

32. Table.Range

函数：Table.Range

语法：Table.Range(**table** as table, **offset** as number, *optional count* as nullable number) as table

说明：选择 table 查询表中，从 offset 位置开始返回 count 行的数据；如果不指定可选项 count，则默认返回 offset 位置开始的所有行。offset 位置从 0 开始，例如：

Power Query: 用 Excel 玩转商业智能数据处理

```
源 = 销售,
```

```
Range = Table.Range(源, 3, 4)
```

结果如图 A-61 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
2	DS31021	运动户外	登山鞋	LOWA男式户外...	98
3	ZX05179	运动户外	自行车	永久山地车变速...	12
4	ZX31887	运动户外	自行车	福特 16寸女式7...	24

图 A-61

```
Range = Table.Range(源, 3)
```

结果如图 A-62 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	XJ00174	电器	相机	索尼NEX-5TL微单	66
2	XJ31547	电器	相机	卡西欧TR500	79
3	XJ44587	电器	相机	富士拍立得mini8	55
4	DN94782	电器	平板电脑	IPAD MINI 16G...	25
5	DN51257	电器	平板电脑	IPAD AIR2 64G W...	59
6	SB15841	饰品	手表	三宅一生水元素...	86
7	ZB34587	饰品	珠宝	周生生足金Char...	86

图 A-62

33. Table.AlternateRows

函数：Table.AlternateRows

语法：Table.AlternateRows(**table** as table, **offset** as number, **skip** as number, **take** as number) as table

说明：保留初始偏移量，然后交替选取和跳过下列或行。参数说明如表 A-60 所示

表 A-60

参 数	说 明
table	输入表
offset	在开始迭代之前要保留的行数
skip	每次迭代中要删除的行数
take	每次迭代中要保留的行数

例如：

```
源 = 销售,
```

```
AlternateRows = Table.AlternateRows(源, 2, 5, 1)
```

结果如图 A-63 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		CF04518		运动户外		冲锋衣		哥伦步防风防雨...		5
2		CF04518		运动户外		冲锋衣		哥伦步三合一冲...		19
3		BB14578		箱包		背包		SPRAYGROUND...		24
4		XJ31547		电器		相机		卡西欧TR500		79

图 A-63

34. Table.Skip

函数：Table.Skip

语法：Table.Skip(**table** as table, **countOrCondition** as any) as table

说明：在 table 查询表中跳过最前面的若干行数据。

如果忽略 countOrCondition，则只跳过第一行；

如果 countOrCondition 为数字，则跳过数字指定的行数；

如果 countOrCondition 为条件，则跳过满足此条件的行，直接到不满足条件的行为止。

例如：

```
源 = 销售,
Skip = Table.Skip(源,12)
```

结果如图 A-64 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		XJ00174		电器		相机		索尼NEX-5TL微单		66
2		XJ31547		电器		相机		卡西欧TR500		79
3		XJ44587		电器		相机		富士拍立得mini8		55
4		DN94782		电器		平板电脑		IPAD MINI 16G...		25

图 A-64

```
Skip = Table.Skip(源,each not Text.Contains([编码],"DN"))
```

结果如图 A-65 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		DN94782		电器		平板电脑		IPAD MINI 16G...		25
2		DN51257		电器		平板电脑		IPAD AIR2 64G...		59
3		SB15841		饰品		手表		三宅一生水元...		86
4		ZB34587		饰品		珠宝		周生生足金Char...		86

图 A-65

35. Table.FindText

函数：Table.FindText

语法：Table.FindText(**table** as table, **text** as text) as table

说明：返回 table 查询表中包含文本 text 的行。如果找不到 text，则返回空表。此函数只支持文

Powery Query：用 Excel 玩转商业智能数据处理

本数据，例如：

```
源 = 销售，  
FindText = Table.FindText(源,"DN")
```

结果如图 A-66 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	DN94782	电器	平板电脑	IPAD MINI 16G...	25
2	DN51257	电器	平板电脑	IPAD AIR2 64G...	59

图 A-66

```
FindText = Table.FindText(源,"水")
```

结果如图 A-67 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	DS31021	运动户外	登山鞋	LOWA男式户外...	98
2	SB15841	饰品	手表	三宅一生水元...	86

图 A-67

36. Table.SelectRows

函数：Table.SelectRows

语法：Table.SelectRows(**table** as table, **condition** as function) as table

说明：返回指定条件 condition 匹配的行的 table 查询表，例如：

```
源 = 销售，  
SelectRows = Table.SelectRows(源,each [数量]>80)
```

结果如图 A-68 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
2	DS31021	运动户外	登山鞋	LOWA男式户外...	98
3	SB15841	饰品	手表	三宅一生水元素...	86
4	ZB34587	饰品	珠宝	周生生足金Char...	86

图 A-68

```
SelectRows = Table.SelectRows(源,each [品类]="运动户外" and [数量]>50)
```

结果如图 A-69 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
2	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
3	DS31021	运动户外	登山鞋	LOWA男式户外...	98

图 A-69

37. Table.SelectColumns

函数 : Table.SelectColumns

语法 : Table.SelectColumns(**table** as table, **columns** as any, *optional* **missingField** as nullable MissingField.Type) as table

说明 : 返回只含有指定 columns 的 table 查询表。

table : 提供的查询表 ;

columns : 要从表中选择的列 , 可以指定多列顺序的字段标题名字 list 列表 ;

missingField : 如果此列不存在 , 则要执行什么操作 , 具体介绍如表 A-61 所示。

表 A-61

参 数	说 明	值
missingField.Error	返回错误警告	0
missingField.Ignore	忽略此列	1
missingField.UseNull	使用 null 空值	2

例如 :

```
源 = 销售,
SelectColumns = Table.SelectColumns(源,{ "编码", "数量" })
```

结果如图 A-70 所示。



	ABC 123 编码	ABC 123 数量
1	CF04518	5
2	CF04518	19
3	CF22179	94
4	DS40015	64

图 A-70

```
SelectColumns = Table.SelectColumns(源,{ "编码", "数量", "单价"},2)
```

结果如图 A-71 所示。



	ABC 123 编码	ABC 123 数量	ABC 123 单价
1	CF04518	5	null
2	CF04518	19	null
3	CF22179	94	null
4	DS40015	64	null

图 A-71

38. Table.Sort

函数：Table.Sort

语法：Table.Sort(**table** as table, **comparisonCriteria** as any) as table

说明：使用一个或多个列字段对 table 查询表进行排序。

comparisonCriteria 参数使用 Order.Ascending 时使用升序排序(可以使用常量 0)，使用 Order.Descending 时使用降序排序(可以使用常量 1)，如果缺省此参数默认为升序排序。

{{"字段 1",Order1},{ "字段 2",Order2}}

表 A-62

参 数	排序类型	常 量
Order.Ascending	升序	0
Order.Descending	降序	1

```
源 = 销售，  
Sort = Table.Sort(源,{{"品类",0}})
```

结果如图 A-72 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	XJ31547	电器	相机	卡西欧TR500	79
2	XJ44587	电器	相机	富士拍立得mini8	55
3	XJ00174	电器	相机	索尼NEX-5TL微单	66
4	DN51257	电器	平板电脑	IPAD AIR2 64G...	59

图 A-72

```
Sort = Table.Sort(源,{{"品类",0},{ "数量",1}})
```

结果如图 A-73 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	XJ31547	电器	相机	卡西欧TR500	79
2	XJ00174	电器	相机	索尼NEX-5TL微单	66
3	DN51257	电器	平板电脑	IPAD AIR2 64G...	59
4	XJ44587	电器	相机	富士拍立得mini8	55

图 A-73

39. Table.ReverseRows

函数：Table.ReverseRows

语法：Table.ReverseRows(**table** as table) as table

说明：逆序排序 table 查询表，例如：

```
源 = 销售，
```

```
ReverseRows = Table.ReverseRows(源)
```

结果如图 A-74 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	ZB34587	饰品	珠宝	周生生足金Char...	86
2	SB15841	饰品	手表	三宅一生水元素...	86
3	DN51257	电器	平板电脑	IPAD AIR2 64G W...	59
4	DN94782	电器	平板电脑	IPAD MINI 16G W...	25

图 A-74

40. Table.ReplaceRows

函数：Table.ReplaceRows

语法：Table.ReplaceRows(**table** as table, **offset** as number, **count** as number, **rows** as list) as table

说明：在 table 查询表中，从偏移位置 offset 开始替换 count 行的内容为新插入的行 rows，例如：

源 = 销售，

```
ReplaceRows = Table.ReplaceRows(源, 0, 16, {})
```

结果如图 A-75 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	DN51257	电器	平板电脑	IPAD AIR2 64G WI...	59
2	SB15841	饰品	手表	三宅一生水元素...	86
3	ZB34587	饰品	珠宝	周生生足金Char...	86

图 A-75

```
ReplaceRows = Table.ReplaceRows(源, 1, 16, {[编码="A", 品类="B", 大类="C",  
商品名称="D", 数量=99]} )
```

结果如图 A-76 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨...	5
2	A	B	C	D	99
3	SB15841	饰品	手表	三宅一生水元素...	86
4	ZB34587	饰品	珠宝	周生生足金Char...	86

图 A-76

41. Table.ReplaceValue

函数：Table.ReplaceValue

语法：Table.ReplaceValue(**table** as table, **oldValue** as any, **newValue** as any, **replacer** as function, **columnsToSearch** as list) as table

说明：在 table 查询表中，将 oldValue 替换为 newValue，其中：

Powery Query：用 Excel 玩转商业智能数据处理

table：输入表；
oldValue：等替换的旧值；
newValue：替换结果的新值；
replacer：替换规则；
Replacer.ReplaceValue：替换完整值；
Replacer.ReplaceText：替换字符串；
columnsToSearch：每次迭代中要保留的行数。

```
源 = 销售，  
ReplaceValue = Table.ReplaceValue(源,"运动户外","户外",  
    Replacer.ReplaceValue,{"品类"})
```

结果如图 A-77 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	户外	冲锋衣	哥仑步防风防...	5
2	CF04518	户外	冲锋衣	哥仑步三合一...	19
3	CF22179	户外	冲锋衣	牧高笛情侣三...	94
4	DS40015	户外	登山鞋	诺诗兰 Gore-tex...	64

图 A-77

```
ReplaceValue = Table.ReplaceValue(源,"三合一","3 合 1",  
    Replacer.ReplaceText,{"商品名称"})
```

结果如图 A-78 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步3合1冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣3合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64

图 A-78

42. Table.ReplaceErrorValues

函数：Table.ReplaceErrorValues

语法：Table.ReplaceErrorValues(**table** as table, **errorReplacement** as list) as table

说明：使用 errorReplacement 列表中的新值替换 table 查询表指定列中的错误值。

errorReplacement 替换格式为：{{"列名 1","替换 1"},"列名 2","替换 2"}}

例如：

```
源 = 销售，  
ReplaceErrorValues = Table.ReplaceErrorValues(源,{{"数量",0}})
```

结果如图 A-79 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		CF04518		运动户外		冲锋衣		哥仑步防风防雨...		5
2		CF04518		运动户外		冲锋衣		哥仑步三合一冲...		19
3		CF22179		运动户外		冲锋衣		牧高笛情侣三合...		94
4		DS40015		运动户外		登山鞋		诺诗兰Gore-tex...		64

图 A-79

43. Table.ReplaceMatchingRows

函数：Table.ReplaceMatchingRows

语法：Table.ReplaceMatchingRows(**table** as table, **replacements** as list, *optional* **equationCriteria** as any) as table

说明：使用提供的行替换 table 查询表中所有指定的行，要替换的行以及替换项在 replacements 中使用 {old,new} 格式指定。可以指定一个可选的 equationCriteria 参数，以控制各行之间的比较。

replacement 替换格式为：{{"old","new"},"old","new"}}

例如：

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],
    [品名="雪梨",数量=4],[品名="苹果",数量=2]}),
FromRecords = Table.ReplaceMatchingRows(源,{{[品名="雪梨",数量=2],
    [品名="香蕉",数量=10]},[品名="雪梨",数量=4],[品名="芒果",数量=20]})
```

结果如图 A-80 所示。

	ABC 123	品名	ABC 123	数量
1		雪梨		2
2		苹果		3
3		雪梨		4
4		苹果		2

源

	ABC 123	品名	ABC 123	数量
1		香蕉		10
2		苹果		3
3		芒果		20
4		苹果		2

ReplaceMatchingRows

图 A-80

44. Table.InsertRows

函数：Table.InsertRows

语法：Table.InsertRows(**table** as table, **offset** as number, **rows** as list) as table

说明：使用行列表 rows 插入到 table 查询表的指定偏移位置 offset。插入表行与 table 结构需要完全相同，例如：

Powery Query：用 Excel 玩转商业智能数据处理

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],  
    [品名="雪梨",数量=4],[品名="苹果",数量=2]}),  
InsertRows = Table.InsertRows(源,2,{[品名="香蕉",数量=10],  
    [品名="芒果",数量=20]})
```

结果如图 A-81 所示。

源		InsertRows																																															
<table><tr><th>ABC 123</th><th>品名</th><th>ABC 123</th><th>数量</th></tr><tr><td>1</td><td>雪梨</td><td></td><td>2</td></tr><tr><td>2</td><td>苹果</td><td></td><td>3</td></tr><tr><td>3</td><td>雪梨</td><td></td><td>4</td></tr><tr><td>4</td><td>苹果</td><td></td><td>2</td></tr></table>	ABC 123	品名	ABC 123	数量	1	雪梨		2	2	苹果		3	3	雪梨		4	4	苹果		2	<table><tr><th>ABC 123</th><th>品名</th><th>ABC 123</th><th>数量</th></tr><tr><td>1</td><td>雪梨</td><td></td><td>2</td></tr><tr><td>2</td><td>苹果</td><td></td><td>3</td></tr><tr><td>3</td><td>香蕉</td><td></td><td>10</td></tr><tr><td>4</td><td>芒果</td><td></td><td>20</td></tr><tr><td>5</td><td>雪梨</td><td></td><td>4</td></tr><tr><td>6</td><td>苹果</td><td></td><td>2</td></tr></table>	ABC 123	品名	ABC 123	数量	1	雪梨		2	2	苹果		3	3	香蕉		10	4	芒果		20	5	雪梨		4	6	苹果		2
ABC 123	品名	ABC 123	数量																																														
1	雪梨		2																																														
2	苹果		3																																														
3	雪梨		4																																														
4	苹果		2																																														
ABC 123	品名	ABC 123	数量																																														
1	雪梨		2																																														
2	苹果		3																																														
3	香蕉		10																																														
4	芒果		20																																														
5	雪梨		4																																														
6	苹果		2																																														

图 A-81

45. Table.Distinct

函数：Table.Distinct

语法：Table.Distinct(**table** as table, *optional equationCriteria* as any) as table

说明：从 table 查询表中删除重复的行；如果指定 equationCriteria 中的列字段名称，则仅按指定的列字段测试删除重复项，例如：

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],  
    [品名="雪梨",数量=4],[品名="苹果",数量=2]}),  
Distinct = Table.Distinct(源)
```

结果如图 A-82 所示。

源		Distinct																																					
<table><tr><th>ABC 123</th><th>品名</th><th>ABC 123</th><th>数量</th></tr><tr><td>1</td><td>雪梨</td><td></td><td>2</td></tr><tr><td>2</td><td>苹果</td><td></td><td>3</td></tr><tr><td>3</td><td>雪梨</td><td></td><td>2</td></tr><tr><td>4</td><td>苹果</td><td></td><td>4</td></tr></table>	ABC 123	品名	ABC 123	数量	1	雪梨		2	2	苹果		3	3	雪梨		2	4	苹果		4		<table><tr><th>ABC 123</th><th>品名</th><th>ABC 123</th><th>数量</th></tr><tr><td>1</td><td>雪梨</td><td></td><td>2</td></tr><tr><td>2</td><td>苹果</td><td></td><td>3</td></tr><tr><td>3</td><td>苹果</td><td></td><td>4</td></tr></table>	ABC 123	品名	ABC 123	数量	1	雪梨		2	2	苹果		3	3	苹果		4	
ABC 123	品名	ABC 123	数量																																				
1	雪梨		2																																				
2	苹果		3																																				
3	雪梨		2																																				
4	苹果		4																																				
ABC 123	品名	ABC 123	数量																																				
1	雪梨		2																																				
2	苹果		3																																				
3	苹果		4																																				

图 A-82

```
Distinct = Table.Distinct(源,"品名")
```

结果如图 A-83 所示

源	品名	数量
1	雪梨	2
2	苹果	3
3	雪梨	2
4	苹果	4

Distinct	品名	数量
1	雪梨	2
2	苹果	3

图 A-83

46. Table.RemoveRowsWithErrors

函数：Table.RemoveRowsWithErrors

语法：Table.RemoveRowsWithErrors(**table** as table, *optional columns* as nullable list) as table

说明：从 table 查询表中删除至少有一个单元中有错误的行，如果指定 columns 的列字段标题名称，则只检查指定列中是否有错误。

columns 格式为：{"column1","column2","column3"}

例如：

```
源 = 销售,
RemoveRowsWithErrors = Table.RemoveRowsWithErrors(源)
```

结果如图 A-84 所示。

ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1 CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2 CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3 CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4 DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64

图 A-84

47. Table.RemoveFirstN

函数：Table.RemoveFirstN

语法：Table.RemoveFirstN(**table** as table, **countOrCondition** as any) as table

说明：从 table 查询表中删除前几行；如果 countOrCondition 是数字，则从顶部开始删除指定行数，若 countOrCondition 是一个条件，则删除满足此条件的行，直到不满足条件为止，例如。

```
源 = 销售,
RemoveFirstN = Table.RemoveFirstN(源, 15)
```

结果如图 A-85 所示。

Power Query: 用 Excel 玩转商业智能数据处理

ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1	DN94782		电器		平板电脑		IPAD MINI 16G...		25
2	DN51257		电器		平板电脑		IPAD AIR2 64G...		59
3	SB15841		饰品		手表		三宅一生水元...		86
4	ZB34587		饰品		珠宝		周生生足金Char...		86

图 A-85

```
RemoveFirstN = Table.RemoveFirstN(源, each [品类] <> "饰品")
```

结果如图 A-86 所示。

ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1	SB15841		饰品		手表		三宅一生水元...		86
2	ZB34587		饰品		珠宝		周生生足金Char...		86

图 A-86

48. Table.RemoveLastN

函数：Table.RemoveLastN

语法：Table.RemoveLastN(table as table, countOrCondition as any) as table

说明：从 table 查询表中删除最后几行；如果 countOrCondition 是数字，则从最末行开始删除指定行数，若 countOrCondition 是一个条件，则删除满足此条件的行，直到不满足条件为止，例如：

源 = 销售，

```
RemoveLastN = Table.RemoveLastN(源, 15)
```

结果如图 A-87 所示。

ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1	CF04518		运动户外		冲锋衣		哥仑步防风防雨...		5
2	CF04518		运动户外		冲锋衣		哥仑步三合一冲...		19
3	CF22179		运动户外		冲锋衣		牧高笛情侣三合...		94
4	DS40015		运动户外		登山鞋		诺诗兰Gore-tex...		64

图 A-87

```
RemoveLastN = Table.RemoveLastN(源, each [品类] <> "运动户外")
```

结果如图 A-88 所示。

ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1	CF04518		运动户外		冲锋衣		哥仑步防风防雨...		5
2	CF04518		运动户外		冲锋衣		哥仑步三合一冲...		19
3	CF22179		运动户外		冲锋衣		牧高笛情侣三合...		94
4	DS40015		运动户外		登山鞋		诺诗兰Gore-tex...		64
5	DS31021		运动户外		登山鞋		LOWA男式户外...		98
6	ZK05179		运动户外		自行车		永久山地车变速...		12
7	ZK31887		运动户外		自行车		福特16寸女式7...		24

图 A-88

49. Table.RemoveRows

函数 : Table.RemoveRows**语法** : Table.RemoveRows(**table** as table, **offset** as number, *optional* **count** as nullable number) as table**说明** : 从 table 查询表中从偏移位置 offset 开始删除 count 指定的行数；如果不指定 count，则默认删除 offset 位置的 1 行，例如：

```
源 = 销售,
RemoveRows = Table.RemoveRows(源, 3)
```

结果如图 A-89 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	DS31021	运动户外	登山鞋	LOWA男式户外...	98
5	ZX05179	运动户外	自行车	永久山地车莫...	12

图 A-89

```
RemoveRows = Table.RemoveRows(源, 3, 14)
```

结果如图 A-90 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94
4	SB15841	饰品	手表	三宅一生水元...	86
5	ZB34587	饰品	珠宝	周生生足金Char...	86

图 A-90

50. Table.RemoveMatchingrows

函数 : Table.RemoveMatchingrows**语法** : Table.RemoveMatchingrows(**table** as table, rows as list, *optional* **equationCriteria** as any) as table**说明** : 从 table 查询表中删除出现在指定的 rows 行。可以指定 equationCriteria 控制表各行之间的比较，例如：

```
源 = 销售,
RemoveMatchingRows = Table.RemoveMatchingRows(源, {[品类="运动户外"],
[品类="箱包"]}, "品类")
```

Powery Query：用 Excel 玩转商业智能数据处理

结果如图 A-91 所示。

	ABC 123	编码	ABC 123	品类	ABC 123	大类	ABC 123	商品名称	ABC 123	数量
1		XJ00174		电器		相机		索尼NEX-5TL微单		66
2		XJ31547		电器		相机		卡西欧TR500		79
3		XJ44587		电器		相机		富士拍立得mini8		55
4		DN94782		电器		平板电脑		IPAD MINI 16G...		25
5		DN51257		电器		平板电脑		IPAD AIR2 64G...		59
6		SB15841		饰品		手表		三宅一生水元...		86
7		ZB34587		饰品		珠宝		周生生足金Char...		86

图 A-91

51. Table.RemoveColumns

函数：Table.RemoveColumns

语法：Table.RemoveColumns(**table** as table, columns as any, *optional* **missingField** as nullable MissingField.Type) as table

说明：从 table 查询表中删除指定的列。

MissingField.Type 参数说明如表 A-63，缺省时为 MissingField.Error。

表 A-63

参 数	说 明	值
MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略错误	1
MissingField.null	若无此字段，则返回 null	2

例如：

```
源 = 销售，  
RemoveColumns = Table.RemoveColumns(源,{"品类","大类","商品名称"})
```

结果如图 A-92 所示。

	ABC 123	编码	ABC 123	数量
1		CF04518		5
2		CF04518		19
3		CF22179		94
4		DS40015		64

图 A-92

52. Table.RenameColumns

函数 : Table.RenameColumns

语法 : Table.RenameColumns(**table** as table, **renames** as any, *optional* **missingField** as nullable MissingField.Type) as table

说明 : 从 table 查询表中删除指定的列。

renames 格式为 : {{"old","new"},{"old","new"}}

MissingField.Type 参数说明如表 A-64 所示，缺省时为 MissingField.Error。

表 A-64

参 数	说 明	值
MissingField.Error	若无此字段，则错误警告	0
MissingField.Ignore	若无此字段，则忽略错误	1
MissingField.null	若无此字段，则返回 null	2

例如：

```
源 = 销售,
RenameColumns = Table.RenameColumns(源,{{"编码","ID"},{"品类","分类"}})
```

结果如图 A-93 所示。

	ABC 123 ID	ABC 123 分类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨...	5
2	CF04518	运动户外	冲锋衣	哥伦步三合一冲...	19
3	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64

图 A-93

53. Table.AddColumn

函数 : Table.AddColumn

语法 : Table.AddColumn(**table** as table, **newColumnName** as text, **columnGenerator** as function, *optional* **columnType** as nullable type) as table

说明 : 将名称为 newColumnName 的新列添加到 table ,可以使用常量 ,也可以指定 columnGenerator 函数来计算其他字段的列，支持 Text、Number、Date、Time、DateTime 等类别的函数应用，columnType 可以指定列字段的数据类型，例如：

```
源 = 销售,
AddColumn = Table.AddColumn(源,"折扣",each 0.8)
```

结果如图 A-94 所示。

Powery Query：用 Excel 玩转商业智能数据处理

	ABC 123 编码	ABC 123 品	ABC 123 大类	ABC 123 商品名称	ABC 123 数量	ABC 123 折扣
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5	0.8
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19	0.8
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94	0.8
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64	0.8

图 A-94

```
AddColumn = Table.AddColumn(源,"备料计划",each [数量]*1.5,type number)
```

结果如图 A-95 所示。

	ABC 123 编码	ABC 123 品	ABC 123 大类	ABC 123 商品名称	ABC 123 数量	1.2 备料计划
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5	7.5
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19	28.5
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94	141
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64	96

图 A-95

```
AddColumn = Table.AddColumn(源,"唛头",each Text.Start([编码],2),type text)
```

结果如图 A-96 所示。

	ABC 123 编码	ABC 123 品	ABC 123 大类	ABC 123 商品名称	ABC 123 数量	ABC 123 唛头
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5	CF
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19	CF
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94	CF
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64	DS

图 A-96

54. Table.AddIndexColumn

函数：Table.AddIndexColumn

语法：Table.AddIndexColumn(table as table, optional newColumnName as text, initialValue as function, increment as nullable type) as table

说明：添加索引列，可选值 initialValue 为初始索引值，可选值 increment 指定每个索引值的步进增量值，例如：

源 = 销售，

```
AddIndexColumn = Table.AddIndexColumn(源,"索引",1,2)
```

结果如图 A-97 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	ABC 123 商品名称	ABC 123 数量	1.2 索引
1	CF04518	运动户外	冲锋衣	哥仑步防风防...	5	1
2	CF04518	运动户外	冲锋衣	哥仑步三合一...	19	3
3	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94	5
4	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64	7

图 A-97

55. Table.PromoteHeaders

函数 : Table.PromoteHeaders

语法 : Table.PromoteHeaders(**table** as table, *optional options* as nullable record) as table

说明 : 将第一行值升级为新的列标题, 例如 :

```
源 = 销售,
PromoteHeaders = Table.PromoteHeaders(源)
```

结果如图所示。

	ABC 123 CF04518	ABC 123 运动户外...	ABC 123 冲锋衣	ABC 123 哥伦步防...	ABC 123 5
1	CF04518	运动户外	冲锋衣	哥伦步三合一冲...	19
2	CF22179	运动户外	冲锋衣	牧高笛情侣三合...	94
3	DS40015	运动户外	登山鞋	诺诗兰 Gore-tex...	64
4	DS31021	运动户外	登山鞋	LOWA男式户外防...	98

图 A-98

options 参数说明 :

promoteAllScalars : 如果设置为 true ,则使用 Culture 区域设置将第一行中的所有值升级为标题。

对于无法转换为文本的值, 将使用默认的列名。

Culture : 区域性名称, "en-us"表示英文地区, "zh-cn"表示中文地区, 例如 : [PromoteAllScalars = true, Culture = "en-US"]

56. Table.DemoteHeaders

函数 : Table.DemoteHeaders

语法 : Table.DemoteHeaders(**table** as table) as table

说明 : 将字段标题降级为第一行的值, 例如 :

```
源 = 销售,
DemoteHeaders = Table.DemoteHeaders(源)
```

结果如图 A-99 所示。

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5
1	编码	品类	大类	商品名称	数量
2	CF04518	运动户外	冲锋衣	哥伦步防风防...	5
3	CF04518	运动户外	冲锋衣	哥伦步三合一...	19
4	CF22179	运动户外	冲锋衣	牧高笛情侣三...	94

图 A-99

57. Table.CombineColumns

函数：Table.CombineColumns

语法：Table.CombineColumns(**table** as table, **sourceColumns** as list, **combiner** as function, **column** as text) as table

说明：将 table 查询表中指定的列字段 sourceColumns，通过 combiner 函数合并连接成一个新的字段，并删除替换前的字段，例如：

```
源 = 销售,
CombineColumns = Table.CombineColumns(源,{ "品类", "大类"},
Combiner.CombineTextByDelimiter("-", "品类-大类")
```

结果如图 A-100 所示。

	ABC 123 编码	ABC 123 品类-大类	ABC 123 商品名称	ABC 123 数量
1	CF04518	运动户外-冲锋衣	哥仑步防风防雨三合一冲...	5
2	CF04518	运动户外-冲锋衣	哥仑步三合一冲锋衣两件...	19
3	CF22179	运动户外-冲锋衣	牧高笛情侣三合一冲锋衣	94
4	DS40015	运动户外-登山鞋	诺诗兰Gore-tex男式三穿...	64

图 A-100

58. Table.SplitColumn

函数：Table.SplitColumn

语法：Table.SplitColumn(**table** as table, **sourceColumns** as text, **splitter** as function, *optional* **columnNamesOrNumber** as any, *optional* **default** as any, *optional* **extraColumns** as any) as table

说明：使用指定的拆分器功能，将指定的一列拆分成一组其他列，参数说明如表 A-65 所示。

表 A-65

参 数	说 明
table	输入表
sourceColumns	待拆分的列
splitter	拆分器函数
columnNamesOrNumber	输出结果的列字段名称或返回的列数
default:	如果缺失时，返回的值
extraColumns	使用额外的列字段

例如：

```
源 = 销售,
SplitColumn = Table.SplitColumn(源,"商品名称",
```

```
Splitter.SplitTextByRepeatedLengths(5),3)
```

结果如图 A-101 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	A ^B _C 商品名称.1	A ^B _C 商品名称.2	A ^B _C 商品名称.3
1	CF04518	运动户外	冲锋衣	哥伦步防风	防雨三合一	冲锋衣两件
2	CF04518	运动户外	冲锋衣	哥伦步三合	一冲锋衣两	件套透气防
3	CF22179	运动户外	冲锋衣	牧高笛情侣	三合一冲锋	衣
4	DS40015	运动户外	登山鞋	诺诗兰Go	re-te	x男式三穿

图 A-101

```
SplitColumn = Table.SplitColumn(源,"商品名称",Splitter.  
SplitTextByRepeatedLengths(5),{"名称1","名称2","品名称3"})
```

结果如图 A-102 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	A ^B _C 名称1	A ^B _C 名称2	A ^B _C 品名称3
1	CF04518	运动户外	冲锋衣	哥伦步防风	防雨三合一	冲锋衣两件
2	CF04518	运动户外	冲锋衣	哥伦步三合	一冲锋衣两	件套透气防
3	CF22179	运动户外	冲锋衣	牧高笛情侣	三合一冲锋	衣
4	DS40015	运动户外	登山鞋	诺诗兰Go	re-te	x男式三穿

图 A-102

```
SplitColumn = Table.SplitColumn(源,"商品名称",  
Splitter.SplitTextByPositions({0, 5}, false),2)
```

结果如图 A-103 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	A ^B _C 商品名称.1	A ^B _C 商品名称.2
1	CF04518	运动户外	冲锋衣	哥伦步防风	防雨三合一冲锋...
2	CF04518	运动户外	冲锋衣	哥伦步三合	一冲锋衣两件套...
3	CF22179	运动户外	冲锋衣	牧高笛情侣	三合一冲锋衣
4	DS40015	运动户外	登山鞋	诺诗兰Go	re-tex男式三穿冲...

图 A-103

```
SplitColumn = Table.SplitColumn(源,"商品名称",  
Splitter.SplitTextByPositions({0, 5}, true),2)
```

结果如图 A-104 所示。

	ABC 123 编码	ABC 123 品类	ABC 123 大类	A ^B _C 商品名称.1	A ^B _C 商品名称.2
1	CF04518	运动户外	冲锋衣	哥伦步防风防雨...	锋衣两件套
2	CF04518	运动户外	冲锋衣	哥伦步三合一冲...	气防风防雨
3	CF22179	运动户外	冲锋衣	牧高笛情侣三	合一冲锋衣
4	DS40015	运动户外	登山鞋	诺诗兰Gore-tex...	三穿冲锋衣

图 A-104

59. Table.FillDown

函数：Table.FillDown

语法：Table.FillDown(**table** as table, **columns** nullable as list) as table

说明：在 table 查询表中，使用前一个单元格的值填充到指定列字段 columns 的下一个为 null 值的单元，例如：

```
源 = Table.FromRecords({[品名="苹果",数量=2],[品名=null,数量=3],  
    [品名="雪梨",数量=4],[品名=null,数量=2]}),  
FillDown = Table.FillDown(源,{"品名"})
```

结果如图 A-105 所示。

ABC 123	品名	ABC 123	数量
1	苹果		2
2		null	3
3	雪梨		4
4		null	2

源

ABC 123	品名	ABC 123	数量
1	苹果		2
2	苹果		3
3	雪梨		4
4	雪梨		2

FillDown

图 A-105

60. Table.FillUp

函数：Table.FillUp

语法：Table.FillUp(**table** as table, **columns** nullable as list) as table

说明：在 table 查询表中，使用前一个单元格的值填充到指定列字段 columns 的上一个为 null 值的单元，例如：

```
源 = Table.FromRecords({[品名=null,数量=2],[品名="苹果",数量=3],  
    [品名=null,数量=4],[品名="雪梨",数量=2]}),  
FillUp = Table.FillUp(源,{"品名"})
```

结果如图 A-106 所示。

ABC 123	品名	ABC 123	数量
1		null	2
2	苹果		3
3		null	4
4	雪梨		2

源

ABC 123	品名	ABC 123	数量
1	苹果		2
2	苹果		3
3	雪梨		4
4	雪梨		2

FillUp

图 A-106

61. Table.Transpose

函数：Table.Transpose

语法：Table.Transpose(**table** as table, *optional columns* as any) as table

说明：转置 table 查询表，使行成为列，列成为行。可选的 columns 可以指定转置后的列名称，
例如：

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],  
    [品名="雪梨",数量=4],[品名="苹果",数量=2]}),  
Transpose = Table.Transpose(源)
```

结果如图 A-107 所示。

ABC 123	品名	ABC 123	数量
1	雪梨		2
2	苹果		3
3	雪梨		4
4	苹果		2

源

ABC 123	Column1	ABC 123	Column2	ABC 123	Column3	ABC 123	Column4
1	雪梨		苹果		雪梨		苹果
2		2		3		4	
							2

transpose

图 A-107

```
Transpose = Table.Transpose(源,{ "水果 1","水果 2","水果 3","水果 4"})
```

结果如图 A-108 所示。

ABC 123	水果1	ABC 123	水果2	ABC 123	水果3	ABC 123	水果4
1	雪梨		苹果		雪梨		苹果
2		2		3		4	
							2

图 A-108

```
Transpose = Table.Transpose(源,List.Transform({1..Table.RowCount(源)},  
    Text.From))
```

结果如图 A-109 所示。

ABC 123	1	ABC 123	2	ABC 123	3	ABC 123	4
1	雪梨		苹果		雪梨		苹果
2		2		3		4	
							2

图 A-109

```
Transpose = Table.Transpose(源,List.Transform({1..Table.RowCount(源)},  
    each "水果"&Text.From(_)))
```

结果如图 A-110 所示。

Powery Query：用 Excel 玩转商业智能数据处理

ABC 123	水果1	ABC 123	水果2	ABC 123	水果3	ABC 123	水果4
1	雪梨		苹果		雪梨		苹果
2		2		3		4	2

图 A-110

62. Table.PositionOf

函数：Table.PositionOf

语法：Table.PositionOf(**table** as table, **row** as record, *optional occurrence* as any, *optional equationCriteria* as any) as any

说明：返回 row 在指定的 table 中出现的行位置；如果找不到该值，则返回-1。位置从 0 开始，必须完全匹配每个字段，其中：

table：输入表。

row：表中要查找其位置的行。

occurrence：指定要返回的行的出现位置，默认为第 1 次出现的位置，具体介绍如表 A-66 所示。

表 A-66

值	说 明
0	第 1 次出现的位置
1	最后 1 次出现的位置
2	所有出现的位置

equationCriteria：控制表行之间的比较。

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],  
    [品名="雪梨",数量=4],[品名="苹果",数量=3]}),  
PositionOf = Table.PositionOf(源,[品名="苹果",数量=3])           //输出 1
```

例如对于图 A-111

ABC 123	品名	ABC 123	数量
1	雪梨		2
2	苹果		3
3	雪梨		4
4	苹果		3

图 A-111

```
PositionOf = Table.PositionOf(源,[品名="苹果",数量=3],1) //输出 3  
PositionOf = Table.PositionOf(源,[品名="苹果",数量=3],2) //输出 {1,3}
```

63. Table.PositionOfAny

函数：Table.PositionOfAny

语法：Table.PositionOfAny(**table** as table, **rows** as list, *optional occurrence* as any, *optional equationCriteria* as any) as any

说明：返回 rows 中任意一条记录在指定的 table 中出现的行位置；如果找不到该值，则返回-1。
位置从 0 开始，必须完全匹配每个字段。

table：输入表。

row：表中要查找其位置的行。

occurrence：指定要返回的行的出现位置，默认为第 1 次出现的位置，具体介绍如表 A-67 所示。

表 A-67

值	说 明
0	第 1 次出现的位置
1	最后 1 次出现的位置
2	所有出现的位置

equationCriteria：控制表行之间的比较。

例如，对于图 A-112：



品名	数量
雪梨	2
苹果	3
雪梨	4
苹果	3

图 A-112

```
源 = Table.FromRecords({[品名="雪梨",数量=2],[品名="苹果",数量=3],
    [品名="雪梨",数量=4],[品名="苹果",数量=3]}),
PositionOfAny = Table.PositionOfAny(源,{[品名="雪梨",数量=2],
    [品名="雪梨",数量=4]}) //输出 0
PositionOfAny = Table.PositionOfAny(源,{[品名="雪梨",数量=2],
    [品名="雪梨",数量=4]},1) //输出 2
PositionOfAny = Table.PositionOfAny(源,{[品名="雪梨",数量=2],
    [品名="雪梨",数量=4]},2) //输出 {0,2}
```

64. Table.Pivot

函数：Table.Pivot

语法：Table.Pivot(**table** as table, **PivotValues** as list, **attributeColumn** as text, **valueColumn** as text, *optional aggregationFunction* as nullable function) as table

说明：透视 table 查询表。

table：输入表。

Powery Query：用 Excel 玩转商业智能数据处理

PivotValues：放到列字段的名称 list 列表（透视到行区域）。

attributeColumn：属性列（透视到列区域）。

valueColumn：值列（透视到值区域）。

aggregationFunction：聚合函数，省略时不聚合。

例如，对于图 A-113：

ABC 123

日期

ABC 123

产品

数量

1

1月

登山鞋

34

2

1月

冲锋衣

38

3

1月

登山鞋

46

4

1月

冲锋衣

21

5

1月

冲锋衣

6

6

2月

登山鞋

14

7

2月

登山鞋

41

8

2月

冲锋衣

22

9

2月

冲锋衣

78

ABC 123

产品

ABC 123

1月

ABC 123

2月

1

冲锋衣

65

100

2

登山鞋

80

55

源

Pivot

图 A-113

```
源 = Excel.CurrentWorkbook(){[Name="透视"]}[Content],
Pivot = Table.Pivot(源, List.Distinct(源[日期]), "日期", "数量", List.Sum)

源 = Excel.CurrentWorkbook(){[Name="透视"]}[Content],
Pivot = Table.Pivot(源, List.Distinct(源[产品]), "产品", "数量", List.Sum)
```

结果如图 A-114 所示。

ABC 123 日期	ABC 123 登山鞋	ABC 123 冲锋衣
1 1月	80	65
2 2月	55	100

图 A-114

65. Table.Unpivot

函数：Table.Unpivot

语法：Table.Unpivot(table as table, pivotColumns as list, attributeColumn as text, valueColumn as text) as table

说明：逆透视 table 查询表的指定列。数据源使用 Table.Pivot 的结果，参数介绍如表 A-68 所示。

表 A-68

参 数	说 明
table	输入表
pivotColumns	需要透视的字段名称 list
attributeColumn	属性列（原列区域名称）
valueColumn	值列（原数值区域的值）

例如：

```
源 = TablePivot,
Unpivot = Table.Unpivot(源, {"登山鞋", "冲锋衣"}, "属性", "值")
```

日期	登山鞋	冲锋衣
1月	80	65
2月	55	100

源

日期	属性	值
1月	登山鞋	80
1月	冲锋衣	65
2月	登山鞋	55
2月	冲锋衣	100

unPivot

66. Table.UnPivotOtherColumns

函数：Table.UnPivotOtherColumns

语法：Table.UnPivotOtherColumns(**table** as table, **pivotColumns** as list, **attributeColumn** as text, **valueColumn** as text) as table

说明：逆透视 table 查询表其他列，参数说明如表 A-69 所示。

表 A-69

参 数	说 明
table	输入表
pivotColumns	透视保留的列字段（原行字段）
attributeColumn	属性列（原列区域名称）
valueColumn	值列（原数值区域的值）

例如：

```
源 = TablePivot,
UnpivotOtherColumns = Table.UnpivotOtherColumns(源, {"日期"}, "品名", "数量")
```

日期	登山鞋	冲锋衣
1月	80	65
2月	55	100

源

日期	品名	数量
1月	登山鞋	80
1月	冲锋衣	65
2月	登山鞋	55
2月	冲锋衣	100

UnpivotOtherColumns

67. Table.Group

函数：Table.Group

语法：Table.Group(**table** as table, **key** as any, **aggregatedColumns** as list, *optional* **groupKind** as nullable GroupKind.Type, *optional* **comparer** as nullable function) as table

说明：按每行指定的列中的值，对 table 查询表的行进行分组；对于每个组将构造一行记录，该

Powery Query：用 Excel 玩转商业智能数据处理

记录包含字段名、聚合函数的值和指定的数据类型。分组后的字段顺序不能保证按为原顺序参数说明如表 A- 所示。

table：输入表。
key：分组的列字段。

Key 参数可以指定一个字段名称作为分组依据，如“字段名”，也可以同时指定多个字段名称做为分组依据，如多个字段的 list 列表{"字段 1","字段 2"}。

aggregatedColumns：每一行的聚合操作

此参数针对需要计算单列或多列进行聚合统计，可以指定字段名称，聚合计算函数，数据类型，使用语法如下两种结构

{{"字段 1", each 聚合函数, 数据类型},{ "字段 2", each 聚合函数, 数据类型}}
{{"字段 1", each 聚合函数},{ "字段 2", each 聚合函数}}

groupKind：应用范围。
comparer：是否区分大小写。

例如：

```
源 = 销售,
Group = Table.Group(源, {"品类", "大类"}, {{ "订单数", each Table.RowCount(_),
type number}, {"总数量", each List.Sum([数量]), type number}})
```

	ABC 123 品类	ABC 123 大类	1.2 订单数	1.2 总数量
1	运动户外	冲锋衣	3	118
2	运动户外	登山鞋	2	162
3	运动户外	自行车	2	36
4	箱包	背包	1	24
5	箱包	提包	2	60
6	箱包	双肩包	2	34
7	电器	相机	3	200
8	电器	平板电脑	2	84
9	饰品	手表	1	86
10	饰品	珠宝	1	86

图 A-115

68. Table.Combine

函数：Table.Combine

语法：Table.Combine(**tables** as list) as table

说明：将多个列字段结构完全相同的 table 表，追加查询按先后顺序合并成一张新 table 查询表，
例如，对于图 A-116：

ABC 123	A	ABC 123	B	ABC 123	C
1	1	2	3		
2	4	5	6		
3	7	8	9		

tab1 1

ABC 123	A	ABC 123	B	ABC 123	C
1	10	12	13		
2	14	15	16		
3	17	18	19		

tabl 2

ABC 123	A	ABC 123	B	ABC 123	C
1	21	22	23		
2	24	25	26		
3	27	28	29		

tabl 3

图 A-116

```

tab1= #table({"A","B","C"},{{1,2,3},{4,5,6},{7,8,9}}),
tab2= #table({"A","B","C"},{{10,12,13},{14,15,16},{17,18,19}}),
tab3= #table({"A","B","C"},{{21,22,23},{24,25,26},{27,28,29}}),
Combine = Table.Combine({table1,table2,table3})

```

结果如图 A-117 所示。

ABC 123	A	ABC 123	B	ABC 123	C
1	1	2	3		
2	4	5	6		
3	7	8	9		
4	10	12	13		
5	14	15	16		
6	17	18	19		
7	21	22	23		
8	24	25	26		
9	27	28	29		

combine

图 A-117

69. Table.Join

函数：Table.Join

语法：Table.Join(**table1** as table, **key1** as any, **table2** as table, **key2** as any, *optional* **joinKind** as nullable JoinKind.Type, *optional* **joinAlgorithm** as nullable JoinAlgorithm.Type) as table

说明：使用 table1 和 table2 指定 key1 字段与 key2 字段匹配后进行合并查询，默认情况下使用内部连接，例如对于图 A-118：

ABC 123	产品ID	ABC 123	商品名称	ABC 123	数量
1	CF12		挎包		42
2	CF17		双肩包		27
3	CF17		双肩包		19
4	CF12		挎包		18

tab1 1

ABC 123	产品ID	ABC 123	单价
1	CF12		98
2	CF17		169

tabl

图 A-118

```

table1 = #table({"产品 ID","商品名称","数量"},{{"CF12","挎包",42},
{"CF17","双肩包",27},{"CF17","双肩包",19},{"CF12","挎包",18}}),
table2 = #table({"产品 ID","单价"},{{"CF12",98},{"CF17",169}}),
Join = Table.Join(table1,"产品 ID",table2,"产品 ID",JoinKind.Inner)

```

结果如图 A-119 所示。

	ABC 123 产品ID	ABC 123 商品名称	ABC 123 数量	ABC 123 单价
1	CF12	挎包	42	98
2	CF17	双肩包	27	169
3	CF17	双肩包	19	169
4	CF12	挎包	18	98

Join

图 A-119

70. Table.NestedJoin

函数：Table.NestedJoin

语法：Table.NestedJoin(**table1** as table, **key1** as any, **table2** as table, **key2** as any, **newColumnName** as text, *optional NestdJoinKind* as nullable NestdJoinKind.Type) as table

说明：使用 table1 和 table2 指定 key1 字段与 key2 字段匹配后进行合并查询，默认情况下使用左外部连接，返回结果是一个嵌套的 table 表格字段，例如，对于图 A-120：

	ABC 123 产品ID	ABC 123 商品名称	ABC 123 数量
1	CF12	挎包	42
2	CF17	双肩包	27
3	CF17	双肩包	19
4	CF12	挎包	18

tabl 1

	ABC 123 产品ID	ABC 123 单价
1	CF12	98
2	CF17	169

tabl 2

图 A-120

```
table1 = #table({"产品 ID", "商品名称", "数量"}, {{ "CF12", "挎包", 42 },  
    { "CF17", "双肩包", 27 }, { "CF17", "双肩包", 19 }, { "CF12", "挎包", 18 } } ),  
table2 = #table({"产品 ID", "单价"}, {{ "CF12", 98 }, { "CF17", 169 } } ),  
NestedJoin = Table.NestedJoin(table1, {"产品 ID"}, table2, {"产品 ID"},  
    "查询结果", JoinKind.Inner)
```

结果如图 A-121 所示。

	ABC 123 产品ID	ABC 123 商品名称	ABC 123 数量	NestedJoin
1	CF12	挎包	42	Table
2	CF17	双肩包	27	Table
3	CF17	双肩包	19	Table
4	CF12	挎包	18	Table

NestedJoin

图 A-121

71. Table.AddJoinColumn

函数：Table.AddJoinColumn

语法：Table.AddJoinColumn(**table1** as table, **key1** as any, **table2** as table, **key2** as any,

newColumnName as text) as table

说明：使用 table1 和 table2 指定 key1 字段与 key2 字段匹配后进行合并查询，类似于使用左外部连接，返回结果是一个嵌套的 table 表格字段，例如，对于图 A-122：

ABC 123	产品ID	ABC 123	商品名称	ABC 123	数量
1	CF12		挎包		42
2	CF17		双肩包		27
3	CF17		双肩包		19
4	CF12		挎包		18

tab1

ABC 123	产品ID	ABC 123	单价
1	CF12		98
2	CF17		169

tab2

图 A-122

```
table1 = #table({"产品 ID","商品名称","数量"},{{"CF12","挎包",42}, {"CF17","双肩包",27}, {"CF17","双肩包",19}, {"CF12","挎包",18}}),
table2 = #table({"产品 ID","单价"},{{"CF12",98}, {"CF17",169}}),
AddJoinColumn = Table.AddJoinColumn(table1,{"产品 ID"},table2, {"产品 ID"},"AddJoinColumn")
```

结果如图 A-123 所示。

ABC 123	产品ID	ABC 123	商品名...	ABC 123	数量	AddJoinColumn
1	CF12		挎包		42	Table
2	CF17		双肩包		27	Table
3	CF17		双肩包		19	Table
4	CF12		挎包		18	Table

AddJoin

图 A-123

72. Table.AggregateTableColumn

函数：Table.AggregateTableColumn

语法：Table.AggregateTableColumn(**table** as table, **column** as text, **aggregations as list**) as table

说明：在 table[column]中的表聚合到包含这些表的聚合值的多个列，aggregations 用于指定包含要聚合的表的列。数据源使用上述 Table.NestedJoin 的结果。参数说明如表 A-70 所示。

表 A-70

参 数	说 明
table	输入表
column	需要聚合的 table 字段
aggregations	聚合操作

此参数针对需要聚合的单列或多列进行聚合统计，可以指定字段名称，聚合计算函数，聚合后的新名称，使用语法如下两种结构

Powery Query：用 Excel 玩转商业智能数据处理

{{"原字段",聚合函数,"新名称"},{"原字段",聚合函数,"新名称"}}

示例：

```
NestedJoin = Table.NestedJoin,
AggregateTableColumn = Table.AggregateTableColumn(NestedJoin,
    "NestedJoin", {{ "数量", List.Sum, "数量" }})
```

结果如图 A-124 所示。

	ABC 123 产品ID	ABC 123 商品名称	ABC 123 数量	NestedJoin
1	CF12	挎包	42	Table
2	CF17	双肩包	27	Table
3	CF17	双肩包	19	Table
4	CF12	挎包	18	Table

NestedJoin

	ABC 123 产品ID	ABC 123 单价	ABC 123 数量
1	CF12	98	60
2	CF17	169	46

AggregateTableColumn

图 A-124

73. Table.ExpandTableColumn

函数：Table.ExpandTableColumn

语法：Table.ExpandTableColumn(**table** as table, **column** as text, **columnNames** as list, *optional newColumnNames* as nullable list) as table

说明：在 table[column]中的表扩展为多个行和列。ColumnNames 用于从内部表中选择要扩展的表。指定 newColumnNames 以避免现有列与新列之间名称相同产生冲突。数据源使用上述 Table.NestedJoin 的结果。参数说明如表 A-71 所示。

表 A-71

参 数	说 明
table	输入表
column	需要展开的 table 字段
columnNames	原字段名称
newColumnNames	新字段名称

例如：

```
NestedJoin = Table.NestedJoin,
ExpandTableColumn = Table.ExpandTableColumn(NestedJoin, "NestedJoin",
    {"商品名称", "数量"}, {"品名", "销量"})
```

	ABC 123 产品ID	ABC 123 商品名称	ABC 123 数量	NestedJoin
1	CF12	挎包	42	Table
2	CF17	双肩包	27	Table
3	CF17	双肩包	19	Table
4	CF12	挎包	18	Table

NestedJoin

	ABC 123 产品ID	ABC 123 单价	ABC 123 品名	ABC 123 销量
1	CF12	98	挎包	42
2	CF17	169	双肩包	27
3	CF17	169	双肩包	19
4	CF12	98	挎包	18

ExpandTableColumn

74. Table.ExpandListColumn

函数：Table.ExpandListColumn

语法：Table.ExpandListColumn(**table** as table, **column** as text) as table

说明：在 table[column]中的内存 list，针对每个值将列表拆分为一行，在每个创建的新行中复制其他字段的值，例如，对于图 A-125，

```
table = #table({"姓名","部门","KPI"},{{{{"大壮","二丫","三多"},"技术部",9.8},
{{{"四喜","五奎","六顺"},"营销部",9.7}}}),
ExpandListColumn = Table.ExpandListColumn(table, "姓名")
```

结果如图 A-125 所示。

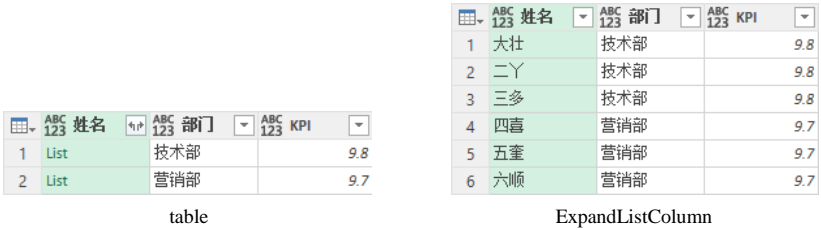


图 A-125

75. Table.ExpandRecordColumn

函数：Table.ExpandRecordColumn

语法：Table.ExpandRecordColumn(**table** as table, **column** as text, **fieldNames** as list, *optional* **newColumnNameNames** as nullable list) as table

说明：在 table[column]中的内存 Record，针对每个值将列表拆分为每一列，例如，对于图 A-126 结果如图 A-126 所示。

Powery Query：用 Excel 玩转商业智能数据处理

	ABC 123 月份	ABC 123 指标
1	10月	Record
2	11月	Record

table

图 A-126

	ABC 123 月份	ABC 123 营销部	ABC 123 开发部	ABC 123 财务部
1	10月	0.985	0.98	0.99
2	11月	0.98	0.985	0.985

ExpandRecordColumn

图 A-127

```
table = #table({"月份", "指标"}, {{ "10 月", [营销部 = 0.985, 开发部 = 0.98, 财务部= 0.99]}, {"11 月", [营销部 = 0.98, 开发部 = 0.985, 财务部= 0.985]}}),
ExpandRecordColumn = Table.ExpandRecordColumn(table, "指标", {"营销部", "开发部", "财务部"})
```

A.11 文件类函数

文件类函数说明如表 A-72 所示。

表 A-72

类	分 类	函数名	概 述
1	文件	Excel.CurrentWorkbook	从当前表导入
2		File.Contents	从文件夹导入
3		Excel.Workbook	从指定路径导入表格
4		Csv.Document	从文本文件导入表格
5		Access.Database	
6		Folder.Contents	

续表

类	分 类	函数名	概 述
7		Folder.Files	
8		Web.Contents	
9		Web.Page	

1. Excel.CurrentWorkbook

函数：Excel.CurrentWorkbook

语法：Excel.CurrentWorkbook() as table

说明：返回当前工作簿中创建的所有表的列表。例如，对于图 A-128：

产品名称	单价
香蕉	3
苹果	4.5
桔子	3.5

产品名称	数量
香蕉	213
桔子	34
香蕉	231
苹果	546

产品ID	产品名称
BA	香蕉
AP	苹果
OR	桔子

工作簿中创建的三个表

图 A-128

```
源 = Excel.CurrentWorkbook()
```

结果如图 A-129 所示。

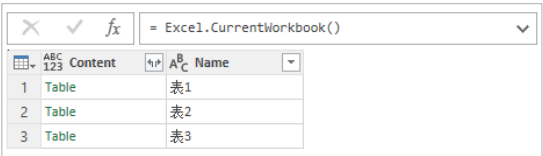


图 A-129

平时我们都是通过功能区中的【Power Query】→【从表】选项导入查询编辑器的公式，是在此函数的基础上【深化】向下钻取获得的其中一个表格。下述公式表示获得上表中，Name 字段等于【表 1】时，返回 Content 字段的结果。

```
源 = Excel.CurrentWorkbook() {[Name="表 1"]}[Content]
```

结果如图 A-130 所示。

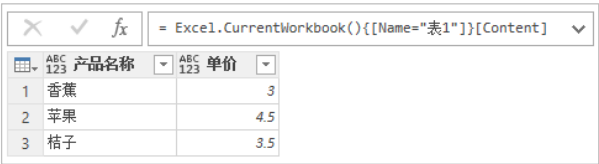


图 A-130

下述两条语句可以分别获取【表 2】和【表 3】是的内容：

```
源 = Excel.CurrentWorkbook() {[Name="表 2"]}[Content]
```

结果如图 A-131 所示。

Powery Query: 用 Excel 玩转商业智能数据处理

= Excel.CurrentWorkbook(){[Name="表2"]}[Content]	
ABC 123 产品名称	ABC 123 数量
1 香蕉	213
2 桔子	34
3 香蕉	231
4 苹果	546

图 A-131

```
源 = Excel.CurrentWorkbook(){[Name="表3"]}[Content]
```

结果如图 A-132 所示。

= Excel.CurrentWorkbook(){[Name="表3"]}[Content]	
ABC 123 产品ID	ABC 123 产品名称
1 BA	香蕉
2 AP	苹果
3 OR	桔子

图 A-132

2. File.Contents

函数：File.Contents

语法：File.Contents(path as text) as binary

说明：通过一个指定地址的文件，文件包含扩展名，获得二进制文件。此函数获得二进制文件后，可以根据文件类型自动识别添加转换函数获得文件内容（如 Excel.Workbook）。

3. Excel.Workbook

函数：Excel.Workbook

语法：Excel.Workbook(worbook as binary, optional useHeaders as nullable logical, optional delayTypes as nullable logical) as table

说明：通过一个二进制文件获得工作簿中所有工作表。此函数不可以单独执行，必须搭配嵌套 File.Contents 函数才可以获得一个二进制文件。

以下示例将 F 盘的【账龄分析实操.xlsx】表格，导入查询编辑器。

```
源 = Excel.Workbook(File.Contents("F:\账龄分析实操.xlsx"))
```

获得整个工作簿的所有工作表清单，结果如图 A-133 所示。

	A ^B C Name	Data	A ^B C Item	A ^B C Kind	Hidden
1	货品资料	Table	货品资料	Sheet	FALSE
2	账龄分析	Table	账龄分析	Sheet	FALSE
3	收款流水	Table	收款流水	Sheet	FALSE
4	帐龄分析入门	Table	帐龄分析入门	Sheet	FALSE
5	去年	Table	去年	Sheet	FALSE
6	今年	Table	今年	Sheet	FALSE
7	费用预算执行...	Table	费用预算执行...	Sheet	FALSE
8	Sheet4	Table	Sheet4	Sheet	FALSE
9	实际发生	Table	实际发生	Sheet	FALSE
10	绩效工资	Table	绩效工资	Sheet	FALSE

图 A-133

通过右击，在弹出的菜单中选择【深化】选项，向下钻取需要分析的表格，以下两条公式都可以获得相同的结果（见图 A-134）。

	A ^B C Name	Data	A ^B C Item	A ^B C
1	货品资料	Table	货品资料	Sheet
2	账龄分析	Table		
3	收款流水	Table		
4	帐龄分析入门	Table		

图 A-134

```
源 = Excel.Workbook(File.Contents("F:\账龄分析实操.xlsx"))
导航 = 源{[Name="货品资料"]}[Data]
```

结果如图 A-135 所示。

	A ^B C Column1	A ^B C Column2	A ^B C Column3	A ^B C Column4	A ^B C Col
1	物料编码	物料名称	规格型号	单位	数量
2		1001 放大管		1890 个	
3		1002 电阻	50R	个	
4		1003 电阻	80R	个	
5		1004 电阻	200R	个	
6		1005 穿心电容	322J	个	
7		1006 环形器		1900 个	
8		1007 平桥	1A1305	个	

图 A-135

4. Csv.Document

函数：Csv.Document

语法：Csv.Document(source as any, optional columns as any, optional delimiter as any, optional extraValues as nullable number, optional encoding as nullable TextEncoding.Type) as table

说明：返回表形式的 csv 文档内容。码数 encoding 表示文件的原始格式，和安装的语言环境相

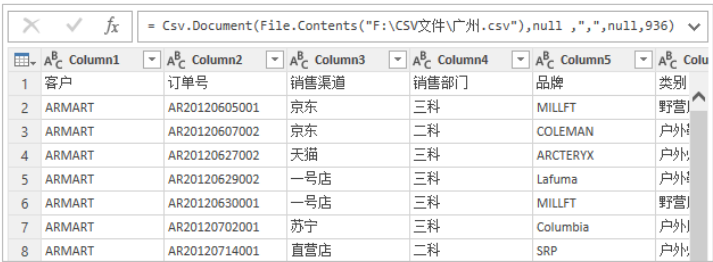
Powery Query: 用 Excel 玩转商业智能数据处理

关，在中国大陆使用的简体中文版 (GB2312) 使用固定值 936。

下面的不例为打开一个 csv 文件：

源 = Csv.Document(File.Contents("F:\CSV 文件\广州.csv"),null,"",",",null,936)

结果如图 A-136 所示。



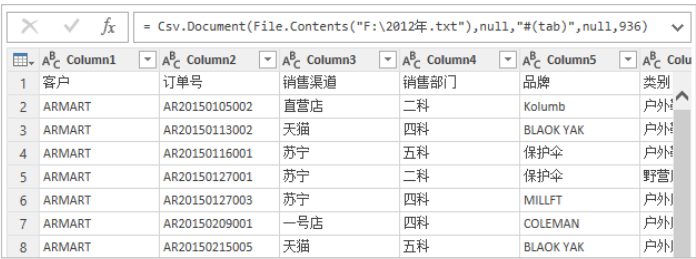
	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	A ^B _C Column4	A ^B _C Column5	A ^B _C Column6
1	客户	订单号	销售渠道	销售部门	品牌	类别
2	ARMART	AR20120605001	京东	三科	MILLFT	野营
3	ARMART	AR20120607002	京东	二科	COLEMAN	户外
4	ARMART	AR20120627002	天猫	三科	ARCTERYX	户外
5	ARMART	AR20120629002	一号店	三科	Lafuma	户外
6	ARMART	AR20120630001	一号店	三科	MILLFT	野营
7	ARMART	AR20120702001	苏宁	三科	Columbia	户外
8	ARMART	AR20120714001	直营店	二科	SRP	户外

图 A-136

下面的示例为打开一个以制表符分隔的 txt 文件：

源 = Csv.Document(File.Contents("F:\2012 年.txt"),null,"#(tab)",null,936)
源 = Csv.Document(File.Contents("F:\2012 年.txt"),总列数,分隔符,换行符,语言)

结果如图 A-137 所示。



	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	A ^B _C Column4	A ^B _C Column5	A ^B _C Column6
1	客户	订单号	销售渠道	销售部门	品牌	类别
2	ARMART	AR20150105002	直营店	二科	Kolumb	户外
3	ARMART	AR20150113002	天猫	四科	BLACK YAK	户外
4	ARMART	AR20150116001	苏宁	五科	保护伞	户外
5	ARMART	AR20150127001	苏宁	二科	保护伞	野营
6	ARMART	AR20150127003	苏宁	四科	MILLFT	户外
7	ARMART	AR20150209001	一号店	四科	COLEMAN	户外
8	ARMART	AR20150215005	天猫	五科	BLACK YAK	户外

图 A-137

5. Access.Database

函数：Access.Database

语法：Access.Database(database as binary, optional options as nullable record) as table

说明：返回 Access 数据库 database 的结构表示形式，可以指定可选的记录参数 options 来控制是否包含关系统选项，例如，对于图 A-138：

= Access.Database(File.Contents("F:\数据库.accdb"))									
ABC 123	Name	ABC 123	Data	ABC 123	Schema	ABC 123	Item	ABC 123	Kind
1	DimChannel		Table				DimChannel		Table
2	DimDate		Table				DimDate		Table
3	DimEntity		Table				DimEntity		Table
4	DimProduct		Table				DimProduct		Table
5	DimProductSubcategory		Table				DimProductSubcategory		Table
6	DimPromotion		Table				DimPromotion		Table
7	实际销售		Table				实际销售		Table

图 A-138

源 = Access.Database(File.Contents("F:\数据库.accdb"))

通过右击，在弹出的菜单中选择【深化】选项，向下钻取需要分析的表格（见图 A-139），以下两条公式都可以获得相同的结果。

6	DimPromotion	Table		Dim
7	实际销售	Table		实际销售

复制

深化

作为新查询添加

图 A-139

源 = Access.Database(File.Contents("F:\数据库.accdb"))
导航 = 源{[Schema="",Item="实际销售"]}[Data]

结果如图 A-140 所示。

= 源{[Schema="",Item="实际销售"]}[Data]										
123	销售编...	销售日期	123	渠道编...	123	商店编...	123	产品编...	123	推广编...
1	7077	2008/04/13 0:00:00		1	297		1086			
2	7078	2009/06/14 0:00:00		1	203		904			
3	7079	2009/11/01 0:00:00		3	200		221			
4	7080	2008/12/11 0:00:00		1	162		1132			
5	7081	2007/04/16 0:00:00		1	265		693			
6	7082	2007/06/08 0:00:00		1	185		222			
7	7084	2007/05/08 0:00:00		2	306		282			
8	7085	2009/09/22 0:00:00		2	307		381			

图 A-140

6. Folder.Contents

函数：Folder.Contents

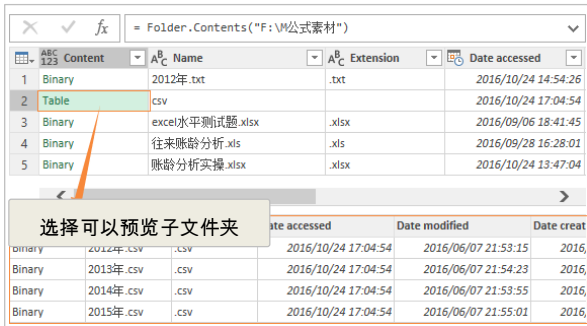
语法：Folder.Contents(path as text) as table

说明：返回一个表，它包含在文件夹路径 path 中找到的每个文件夹和文件所对应的行。每一行都包含所对应的文件夹或文件的属性以及指向其内容的链接，例如：

源 = Folder.Contents("F:\M 公式素材")

Powery Query: 用 Excel 玩转商业智能数据处理

结果如图 A-142 所示。



Content	Name	Extension	Date accessed
1 Binary	2012年.txt	.txt	2016/10/24 14:54:26
2 Table	excel水平测试题.xlsx	.xlsx	2016/10/24 17:04:54
3 Binary	往来账龄分析.xls	.xls	2016/09/06 18:41:45
4 Binary	账龄分析实操.xlsx	.xlsx	2016/09/28 16:28:01
5 Binary			2016/10/24 13:47:04

Content	Name	Extension	Date accessed	Date modified	Date created
Binary	2012年.csv	.csv	2016/10/24 17:04:54	2016/06/07 21:53:15	2016
Binary	2013年.csv	.csv	2016/10/24 17:04:54	2016/06/07 21:54:23	2016
Binary	2014年.csv	.csv	2016/10/24 17:04:54	2016/06/07 21:53:55	2016
Binary	2015年.csv	.csv	2016/10/24 17:04:54	2016/06/07 21:55:01	2016

图 A-142

7. Folder.Files

函数：Folder.Files

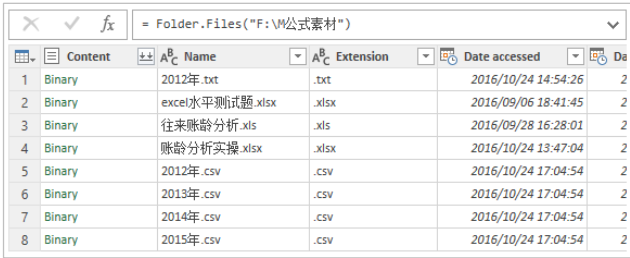
语法：Folder.Files(path as text) as table

说明：返回一个表，它包含在文件夹路径 path 和所有子文件夹中找到的每个文件所对应的行。

每一行都包含所对应的文件的属性以及指向其内容的链接，例如：

源 = Folder.Files("F:\M 公式素材")

结果如图 A-143 所示。



Content	Name	Extension	Date accessed	
1 Binary	2012年.txt	.txt	2016/10/24 14:54:26	2
2 Binary	excel水平测试题.xlsx	.xlsx	2016/09/06 18:41:45	2
3 Binary	往来账龄分析.xls	.xls	2016/09/28 16:28:01	2
4 Binary	账龄分析实操.xlsx	.xlsx	2016/10/24 13:47:04	2
5 Binary	2012年.csv	.csv	2016/10/24 17:04:54	2
6 Binary	2013年.csv	.csv	2016/10/24 17:04:54	2
7 Binary	2014年.csv	.csv	2016/10/24 17:04:54	2
8 Binary	2015年.csv	.csv	2016/10/24 17:04:54	2

图 A-143

8. Web.Contents

函数：Web.Contents

语法：Web.Contents(url as text, options as nullable record) as binary

说明：将从 URL 下载的内容返回为二进制，可以提供可选记录参数 options 以指定其他属性。

此函数不能获得查询结果，需要搭配 Web.Page 函数才能获得表格。

9. Web.Contents

函数：Web.Page

语法：Web.Page(html as any) as table

说明：返回将 HTML 文档的内容（分解为其组成结构），以及完整文档的表示形式及其删除标记后的文本。

以下示例为通过网页查询“欧洲杯” 百度百科所获得查询表格。

```
源 = Web.Page (
    Web.Contents (
        "http://baike.baidu.com/link?url=QABCdpD7Dff-
eV5Ji59_sgmXQHv4T4JuZjm3xrgzCrMidXDX23F4N4181H1W4YUueBTHiPIwrgwVnbXbEuT-
TOT1wiLzqUsmR2pZ1qn_kvEgTafG60Lotd-x8-
ytDjenHldM6vySHkanUPiWklsBH9z7b7OY_VjZpoaYbMAofrtSmwTywtnOsDPCAFPyS8g7izrZ6k
Am6VwW2zVb14BE5KH2LyUuZngB09VE-fDxjJ2vrmoU-41YtjFPURLM6jq1"

    )
)
```

结果如图所示 6-144 所示。

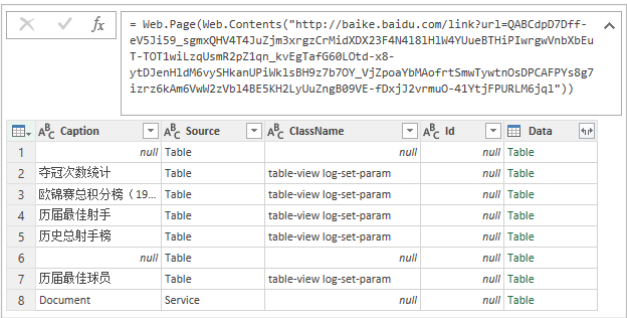


图 A-144

如果希望获得“历史总射手榜”，则可以在对应的“Table” 上右击，在弹出的菜单中选择【深化】选项获得，如图 A-145 和图 A-146 所示。



图 A-145

	<div><div>✕</div><div>✓</div><div><i>fx</i></div></div>	<div>= 源(4)[Data]</div>	
	<div>A^B_C 排名</div>	<div>A^B_C 球员</div>	<div>A^B_C 进球数</div>
1	1	普拉蒂尼（法国...	9
2	1	C.罗纳尔多（葡...	9
3	3	阿兰·希勒（英...	7
4	4	格里兹曼（法国...	6
5	4	范尼斯特鲁伊（...	6
6	4	亨利（法国）	6
7	4	姆希塔扬（阿...	6
8	4	克鲁伊维特（荷...	6
9	4	伊布拉希莫维奇...	6
10	4	鲁尼（英格兰）	6
11	11	马尔科·范·巴斯...	5
12	11	齐达内（法国）	5
13	11	克林斯曼（德国...	5
14	11	戈麦斯（德国）	5
15	11	米洛舍维奇（南...	5
16	11	巴罗什（捷克）	5
17	11	托雷斯（西班牙...	5

图 A-146

结语

书写到这里，也算是耗尽了作者的洪荒之力，本书内容基本覆盖了 Power Query 在工作应用中的常用案例和技巧，对于特殊领域，因为编者的知识面的欠缺而深感无能为力。函数部分的描写大部分参考的是软件自带的帮助文件，并通过自己的扩展理解编辑成册，希望对大家有所帮助。

版权声明

此附件文档作为《Power Query 用 Excel 玩转商业智能数据处理》扩展阅读文件，受知识产权保护，仅供购书读者参考学习，任何形式的转载和分享都应保证该文件的完整性。禁止以任何形式的侵犯作者著

视频教程送书活动

凡购买本书籍的读者，都有机会参与购买视频课程抵扣书费的活动
抵扣书费，以实时购买书价为准

直接从作者购买签名书籍由读者承担运费(10 元/册)

视频课程地址:

https://ke.qq.com/course/139924#tuin=1e72540c&term_id=0

授课老师: 朱仕平 (本书作者)

联系 QQ: 510809100

