

# 准备工作：

2020年12月15日 9:13

# 1.电脑硬件及操作系统要求：

2020年12月12日 21:00

## 操作系统

**Windows 7 及以上，不支持 XP，不支持 Mac 系统，Mac 版 Excel 不能使用 Power Pivot。必须安装IE10或以上版本。**

## 推荐的配置：

**四核CPU+8G+固态硬盘**

**CPU无论是什么牌子的和型号，能装上win7及以上系统就可以，10多年前的都可以  
内存DDR2\DDR3\DDR4都可以  
固态硬盘 SATA2\SATA3\M.2都行**

## 2.相关课程推荐：【建议用最新版】

2020年12月7日 8:47



<https://www.bilibili.com/video/BV1Dk4y1d71j>

学数据分析一定要懂数据库，我说了不算，你们可以到招聘网站上去查看看看数据分析岗对知识的要求



<https://www.bilibili.com/video/BV1oa4y1j75e>

数据清洗同样可以在Excel或PowerQuery中完成，如果是Excel表格，那么你直接用Excel就可以了，但是如果数据保存在数据库上呢？你不想通过SQL保存到本地，而是想通过PowerBI做图后每当数据库数据更新时，你的图表可以通过刷新随着更新，这个时候，建议用PowerQuery进行数据清洗。

PS:学习PowerQuery之前，建议您先看一下我的Python基础篇前8集，学习PowerQuery需要有一些编程基础，易于理解。



<https://www.bilibili.com/video/BV1HE41157bu>

### 3.PowerBI软件下载

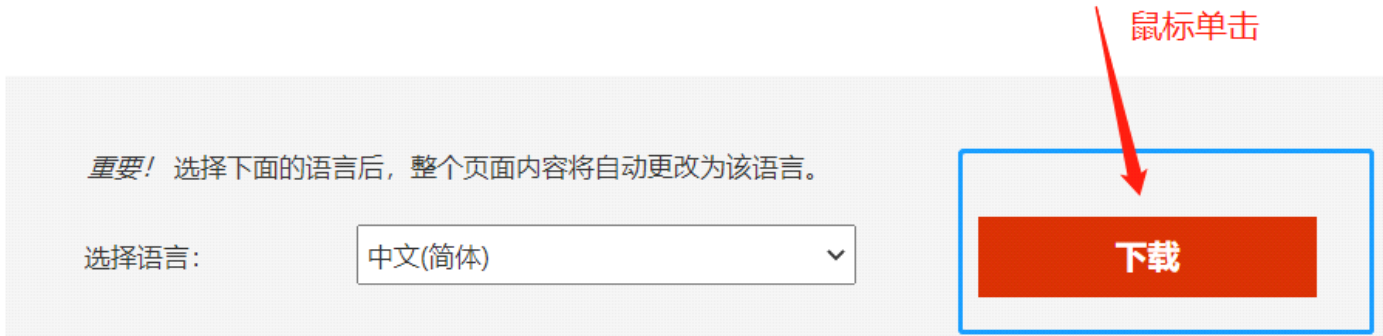
2020年12月25日 23:55

下载地址：

[Download Microsoft Power BI Desktop from Official Microsoft Download Center](#)

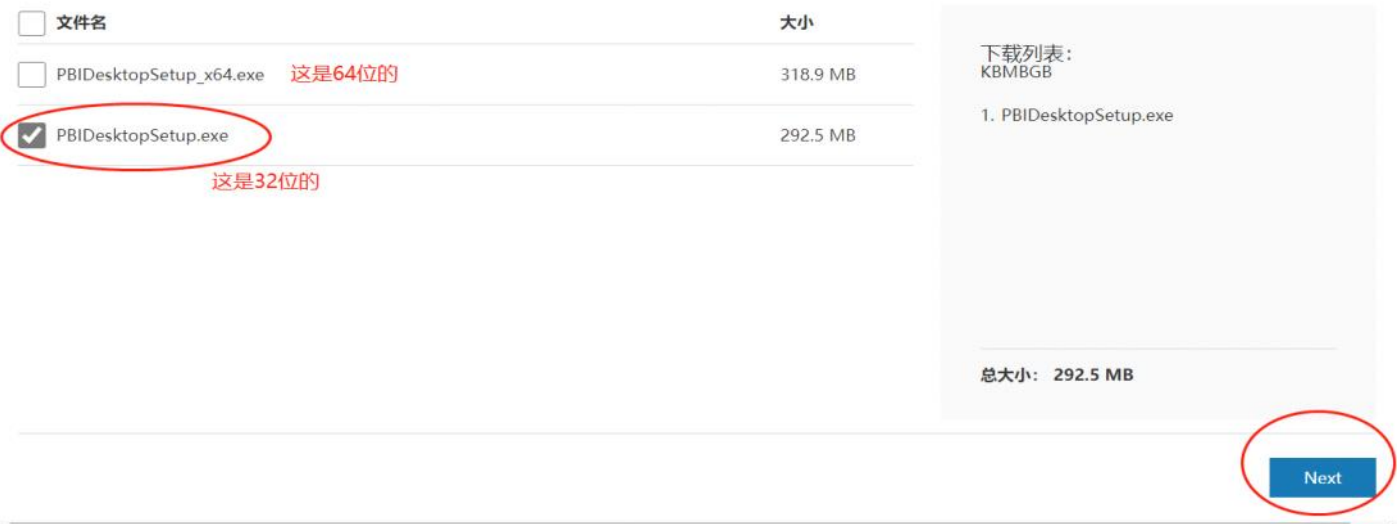
第一步：

Microsoft Power BI Desktop



第二步：

选择您要下载的程序



# 00.界面认识与矩阵表操作

2020年12月14日 12:53

矩阵和表格：

- (1) 整体大小：大小
- (2) 行高：行填充
- (3) 列宽：列宽
- (4) 隐藏列：先关掉【自动换行】，再用鼠标拖拽
- (5) 隐藏行：筛选器里面进行筛选
- (6) 居中显示：【行标题】与【字段格式设置】中分别居中
- (7) 分级显示：+
- (8) 行标题缩进：缩进
- (9) 行字段名显示在不同列：渐变
- (10) 行总计：小计 【按行级别，产品名称】
- (11) 值在行上显示：值
- (12) 条件格式：数据条
- (13) 条件格式：图标

# 01.运算符与新建列

2020年12月8日 15:24

# 01.运算符

2020年12月7日 8:47

运算符	含义
+	加
-	减
*	乘
/	除
^	幂

运算符	含义
=	等于
>	大于
<	小于
>=	大于等于
<=	小于等于
<>	不等于

运算符	含义
&&	与
	或
not	非
&	字符串连接
in	包含
not in	不包含

文本加数值 = [字符]+[数值] # "520" + 1314 = 1834

文本连接数值 = [字符] & [数值] # "520" & 1314 = 5201314

数值连接数值 = [数值] & [数值1] # 1314 & 521 = 1314521

## 02.新建列

2020年12月7日 16:29

新的销售量 = [销售数量]\*10

新的销售量 = '销售表'[销售数量]\*10

→ 功能相同

什么时候表示列的时候必须带表名呢？

销售金额 = [销售数量]\*RELATED('商品表'[进价])

来自：销售表

来自：商品表

总结：新建列的时候，当另一个需要计算的列来自其它表的时候，一定要指名表名是什么。

举例说明：

一班	二班	三班
张三	刘德华	邓光荣
李四	古天乐	陈惠敏
王五	刘青云	向华强

我是一班班主任，学校搞活动，我可以在自己班里这样读：

张三，二班古天乐，三班向华强

也可以这样读

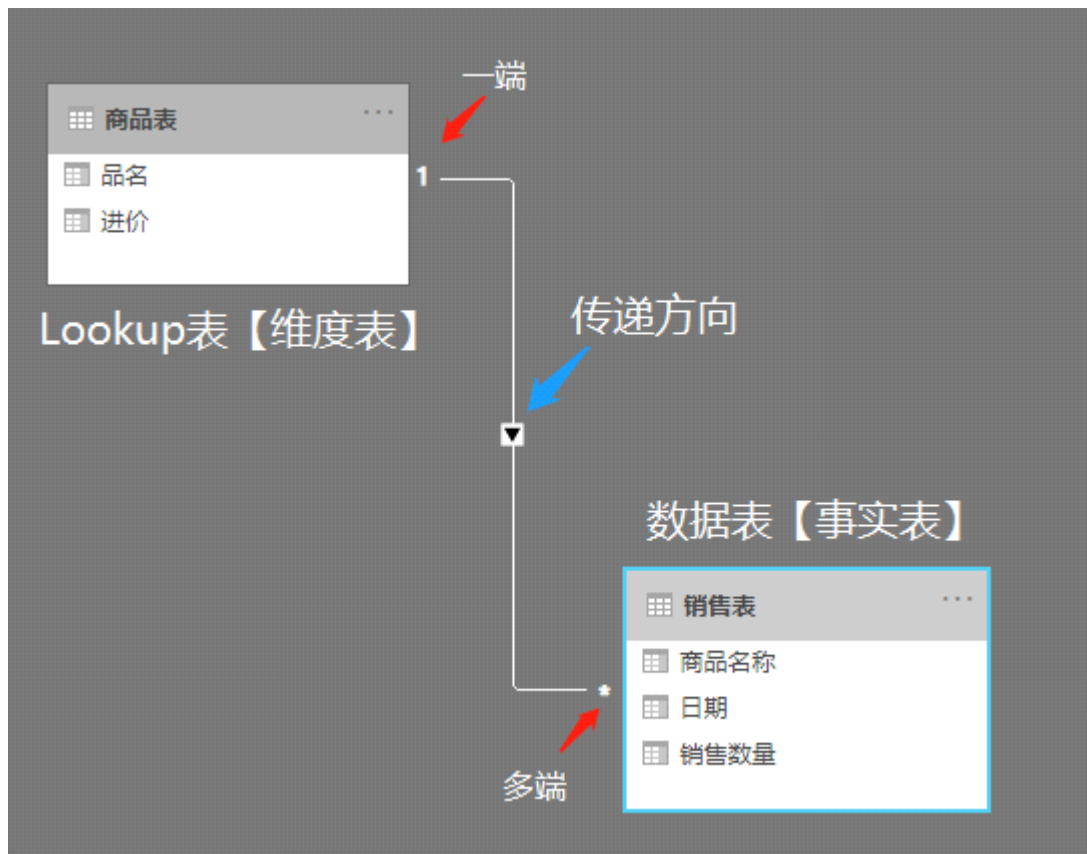
一班张三，二班古天乐，三班向华强



## 02.建模与关系函数

2020年12月7日 16:12

**多端可以向一端索取，一端可以控制传递方向相同的多端数据**



01.Lookupvalue函数 第7课时，你就知道他的用处了。

2020年12月7日 17:03

Lookupvalue函数：就是Vlookup

语法：Lookupvalue(把哪张表的哪个列拿过来，找那张表上的谁？， 找自己表里的谁？)

单价 = LOOKUPVALUE('商品表'[进价], '商品表'[品名], '销售表'[商品名称])

日期	商品名称	销售数量
2020年12月24日	A	1
2020年12月24日	B	2
2020年12月24日	C	3
2020年12月24日	D	4
2020年12月24日	E	5
2020年12月25日	A	10
2020年12月25日	B	20
2020年12月25日	C	30
2020年12月25日	D	40
2020年12月25日	E	50

销售表

品名	进价
A	0.1
B	0.2
C	0.3
D	0.4
E	0.5

商品表

日期	商品名称	销售数量	单价
2020年12月24日	A	1	0.1
2020年12月24日	B	2	0.2
2020年12月24日	C	3	0.3
2020年12月24日	D	4	0.4
2020年12月24日	E	5	0.5
2020年12月25日	A	10	0.1
2020年12月25日	B	20	0.2
2020年12月25日	C	30	0.3
2020年12月25日	D	40	0.4
2020年12月25日	E	50	0.5

完成表

## 02.Related函数与Relatedtable函数

2020年12月7日 17:29

**Related函数：**（多端找一端）（事实表找维度表）（数据表找基础表）

**销售金额 = [销售数量]\*RELATED('商品表'[进价])**

**[销售数量]**这个列来自销售表，销售表是多端  
**[进价]**这个列来自商品表，销售表是一端

**Relatedtable函数：**（一端找多端）（维度表找事实表）（基础表找数据表）

**订单数量 = COUNTROWS(RELATEDTABLE('销售表'))**

品名	进价
A	0.1
B	0.2
C	0.3
D	0.4
E	0.5

商品表  
【一端】

日期	商品名称	销售数量
2020年12月24日	A	1
2020年12月24日	B	2
2020年12月24日	C	3
2020年12月24日	D	4
2020年12月24日	E	5
2020年12月25日	A	10
2020年12月25日	B	20
2020年12月25日	C	30
2020年12月25日	D	40
2020年12月25日	E	50

销售表  
【多端】

品名	进价	订单数量
A	0.1	2
B	0.2	2
C	0.3	2
D	0.4	2
E	0.5	2

完成表

03.多表建模是针对数据库的，数据分析必须懂数据库

2020年12月7日 18:49

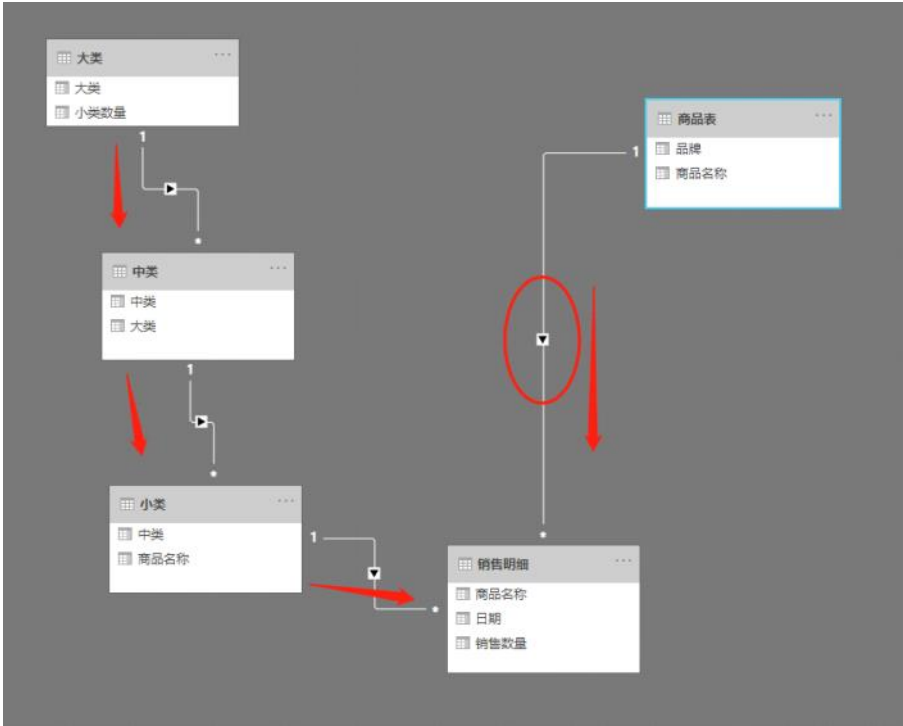


大类	订单数量	中类数量
粮油	4	2
水奶	4	2
肉	4	2

订单数量 = COUNTROWS(RELATEDTABLE('销售明细'))  
中类数量 = COUNTROWS(RELATEDTABLE('中类'))

错误的方式：

2020年12月7日 20:29

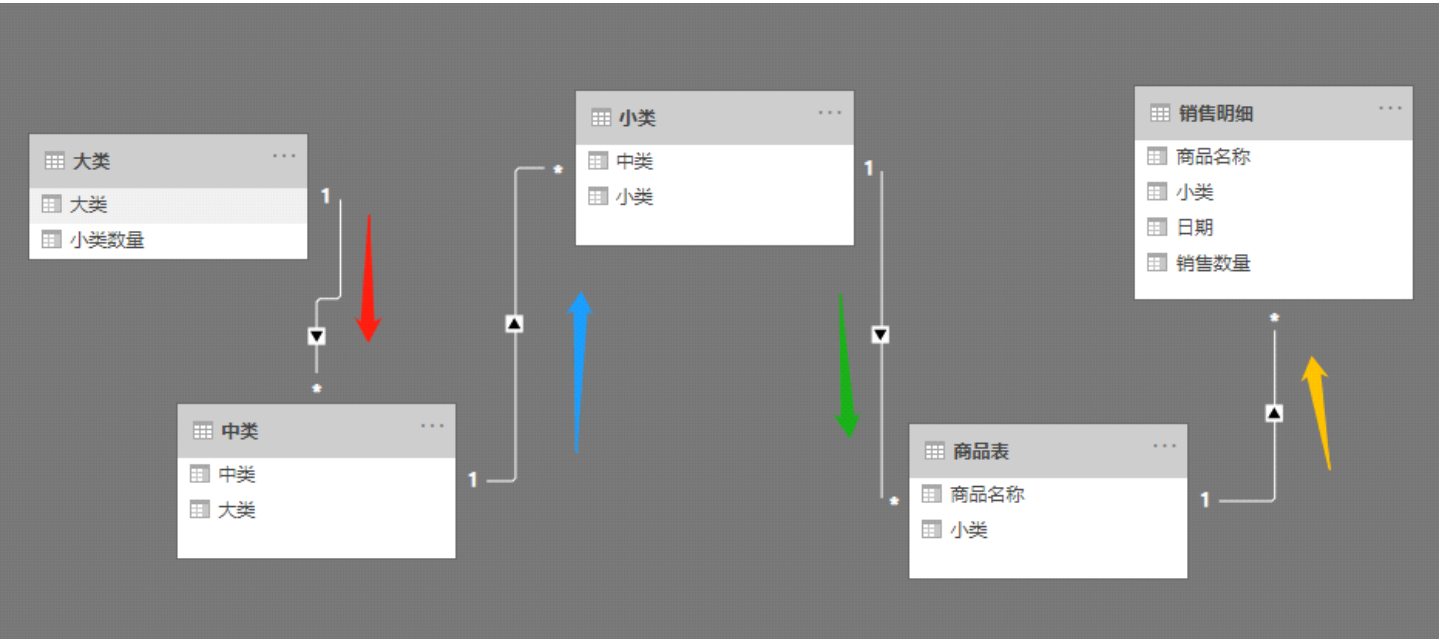


大类	品牌数量
手机	12
电脑	12
图书	12

品牌数量 = COUNTROWS (RELATEDTABLE (' 商品表' ))

# 正确的方式：

2020年12月7日 20:28



大类	品牌数量
手机	4
电脑	4
图书	4

品牌数量 = COUNTROWS (RELATEDTABLE (‘ 商品表’))

数据分析必须懂数据库：

2020年12月8日 14:54

商品编码	商品名称	进价	售价	供货商编码
A001	XX手机	3300	3999	B001
A002	XX电脑	2500	3000	B001
A003	XX图书	25	49.9	B001
A004	XX服装	200	299	B002
A005	XX鞋	150	330	B002
A006	XX相机	12000	14000	B001
A007	XX打印机	300	380	B001
A008	XX电视	8500	9999	B001
A009	XX空调	8900	10200	B001
A010	XX冰箱	6500	7999	B001
商品表				

供货商编码	供货商名称	供货商电话
B001	ABCDEFGFG	1234567
B002	EWTWTE	5678976
供货商表		

日期	商品编码	销售数量
2020/12/1	A001	1
2020/12/1	A002	2
2020/12/1	A003	3
2020/12/1	A004	4
2020/12/1	A005	5
2020/12/1	A006	6
2020/12/1	A007	7
2020/12/1	A008	8
2020/12/1	A009	9
2020/12/1	A010	10
2020/12/2	A001	11
2020/12/2	A002	12
2020/12/2	A003	13
2020/12/2	A004	14
2020/12/2	A005	15
2020/12/2	A006	16
2020/12/2	A007	17
2020/12/2	A008	18
2020/12/2	A009	19
2020/12/2	A010	20
销售表		

## 04.度量值、Calculate引擎、Calculatedtable筛选表

2020年12月8日 15:28



## 01.度量值

2020年12月8日 15:29

总销量 = SUM([销售数量])

总销量 = SUM('销售表'[销售数量])

$$1+2+3+4+5=?$$

这是等差数列因为后一项与前一项的差相同

高斯公式：首项加末项乘以项数除以2

$$(1+5) * 5 / 2 = 15$$

日	总销量	总销量1
24	15	15
25	150	150
总计	165	165

班级	姓名	分数
1班	张三	100
2班	李四	90
1班	王五	80
合计：		270

PowerBI中的度量值，自带筛选功能  
不占用内存。且可以循环使用。

Excel聚合计算：只是把最后的结果算出来摆在那里。

02.筛选引擎Calculate

2020年12月8日 16:29

总销量 = sum('销售表'[销售数量])

A产品销量1 = CALCULATE([总销量], '商品表'[品名]="A")

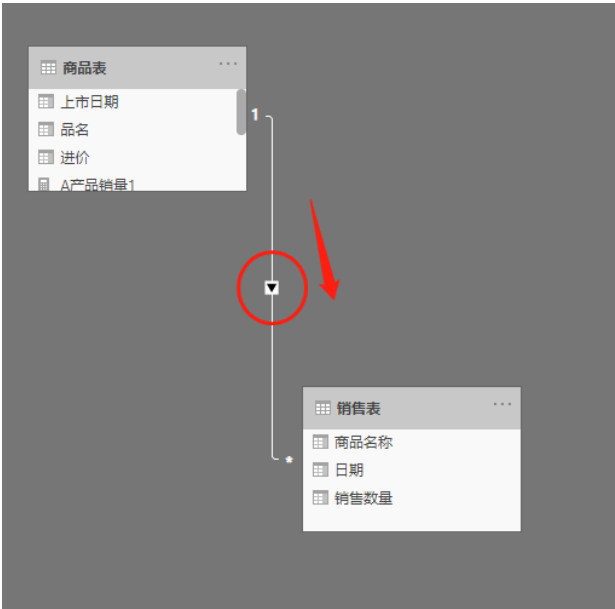
A产品销量2 = CALCULATE([总销量], '销售表'[商品名称]="A")

只限于字符，数值型

关于日期和时间的问题，会放在时间智能函数那里去讲

品名	总销量	A产品销量1	A产品销量2
A	11	11	11
B	22	11	
C	33	11	
D	44	11	
E	55	11	
总计	165	11	11

商品名称	总销量	A产品销量1	A产品销量2
A	11	11	11
B	22		11
C	33		11
D	44		11
E	55		11
总计	165	11	11



## 03.多条件Calculate与筛选表

2020年12月8日 17:00

### 一、多条件Calculate【多列】

多条件 = CALCULATE([总销量], '商品表'[品名] = "A", '商品表'[进价] = 0.1)

多条件1 = CALCULATE([总销量], '商品表'[品名] in {"A", "B", "C"})

多条件2 = CALCULATE([总销量], not '商品表'[品名] in {"A", "B", "C"})

### 二、筛选表

Calculatedtable(表, 筛选条件)

与Calculatedtable区别在于, 可以多表运作筛选, 最终返回一张表



表 = CALCULATETABLE('销售表', '商品表'[品名] = "A", '商品表'[进价] = 0.1)

日期	商品名称	销售数量
2020/12/24 0:00:00	A	1
2020/12/25 0:00:00	A	10

### 类似, SQL语句

SELECT \* FROM `销售表` WHERE 销售数量 > 250;

## 04.推荐度量值的存放方法

2020年12月8日 16:57



在输入数据中新建一个表，放度量值拖拽到这个表中，删除不需要的内容，隐藏一下字段，再打开

## 05.高级筛选器Filter与Values人工造表

2020年12月8日 17:58

返回一个表，用于表示另一个表或表达式的子集，不能单独使用  
Filter函数对筛选的表进行横向的**逐行扫描**，这样的函数叫迭代函数。

Countrows(Filter(表, 筛选条件))

记住这四个字

Calculate(表达式(度量值),Filter('表名', 筛选条件))

一、效果相同，为什么使用Filter函数？

总分 = SUM('成绩表'[分数])

1班男生a = CALCULATE([总分], '花名册'[班级]="1班", '花名册'[性别]="男")

1班男生b = CALCULATE([总分], FILTER('花名册', '花名册'[班级]="1班" && '花名册'[性别]="男"))

姓名	1班男生a	1班男生b
小李	230	230
张三	230	230
总计	460	460

二、什么时候使用Filter函数

在Calculate函数中的直接筛选条件里，我们只能输入：

‘表’ [列] = 固定值    或    ‘表’ [列] <> 固定值

‘表’ [列] >= 固定值    或    ‘表’ [列] <= 固定值

‘表’ [列] > 固定值    或    ‘表’ [列] < 固定值

但是遇到如下情况，就要使用Filter函数

[列]=[度量值]、 [列]=公式、 [列]=[列]

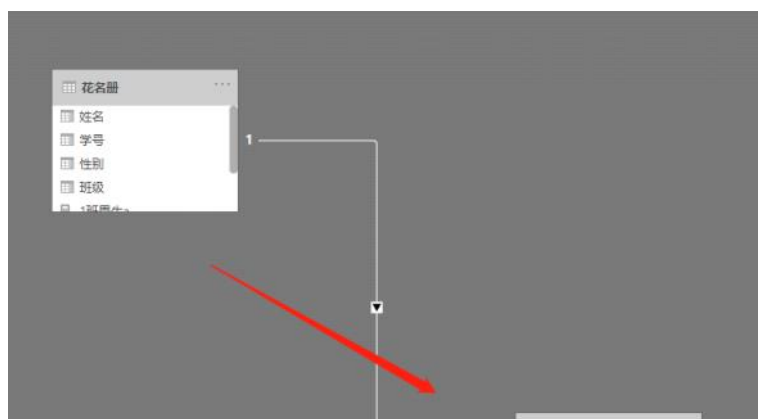
[度量值]=[度量值]、 [度量值]=公式、 [度量值]=固定值

举例：总分大于250分的学生一共考了多少分？

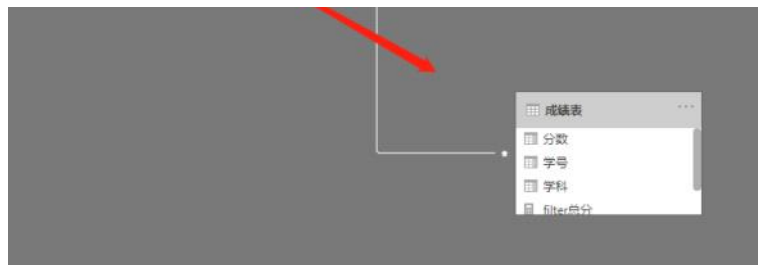
filter总分1 = CALCULATE([总分], FILTER('花名册', [总分]>250))

filter总分2 = CALCULATE([总分], FILTER('成绩表', [总分]>250))    错误

姓名	总分	filter总分	filter总分2
李四	259	259	
王五	277	277	
小李	230		
小三	180		
小四	230		
小王	204		
小五	282	282	



小王	204		
小五	282	282	
小张	203		
张三	230		
总计	2095	818	



如果我只有一张表怎么办？我没有唯一表，只有数据表！那就自己造一张表！

没有枪没有炮，敌人给我们造！

**filter总分2 = CALCULATE([总分],FILTER(VALUES('成绩表'[学号]),[总分]>250))**

学号	总分	filter总分	filter总分2
A001	230		
A002	259	259	259
A003	277	277	277
A004	203		
A005	204		
A006	230		
A007	180		
A008	230		
A009	282	282	282
总计	2095	818	818

返回由一列构成的一个表，该表包含来自指定表或列的非重复值。表中重复值将被删除，返回唯一值。

Values函数生成的表是一张虚拟表。

在很多时候使用Filter、Calculate、Countrows、SumX、TopN这些函数，需要引用表而不能直接引用列。

Values函数的功能是把列转化成包含列的表，这时只需要嵌套一个Values函数就可以引用了。

注意：这张表，是不重复的信息表！

**经典语句：** Calculate([度量值],Filter(Values( '表' [列名]),...))

## 06.被翻译耽误的上下文

2020年12月8日 19:08

结构					
1 总分 = SUM('成绩表'[分数])					
学号	姓名	班级	性别	总分	
A001	张三	1班	男	2095	
A002	李四	3班	女	2095	
A003	王五	2班	男	2095	
A004	小张	2班	女	2095	
A005	小王	3班	男	2095	
A006	小李	1班	男	2095	
A007	小三	1班	女	2095	
A008	小四	2班	女	2095	
A009	小五	1班	女	2095	

新建列是行上下文

学号	数学	英语	语文	总计
A001	51	90	89	230
A002	99	87	73	259
A003	100	98	79	277
A004	77	74	52	203
A005	64	77	63	204
A006	72	66	92	230
A007	54	71	55	180
A008	99	68	63	230
A009	96	90	96	282
总计	712	721	662	2095

度量值是筛选上下文

行
学号
列
学科
值
度量值总分

同样一个公式：

新建列总分 = SUM('成绩表'[分数])

度量值总分 = SUM('成绩表'[分数])

写在【新建列】和【度量值】中效果不一样

总分2 = CALCULATE(SUM('成绩表'[分数]))

学号	姓名	班级	性别	总分	总分2
A001	张三	1班	男	2095	230
A002	李四	3班	女	2095	259
A003	王五	2班	男	2095	277
A004	小张	2班	女	2095	203
A005	小王	3班	男	2095	204
A006	小李	1班	男	2095	230
A007	小三	1班	女	2095	180
A008	小四	2班	女	2095	230
A009	小五	1班	女	2095	282

1.度量值天生具有筛选功能

2.新建列是行上下文，行上下文没有筛选功能

3.想让行上下文实现筛选功能就要在外面套一个Calculate

# 01.Filter与上下文

2020年12月8日 20:35

filter总分1 = CALCULATE([总分],FILTER('花名册',[总分]>250)) # 正确  
filter总分2 = CALCULATE([总分],FILTER('花名册',SUM('成绩表'[分数])>250)) # 错误

班级	filter总分1	filter总分2
1班	552	1192
2班	277	710
3班	259	463
总计	1088	2365

既然，[总分]=SUM('成绩表'[分数])，那么为什么不能写成第二种情况

filter总分3 = CALCULATE([总分],FILTER('花名册',CALCULATE(SUM('成绩表'[分数]))>250))

班级	filter总分1	filter总分2	filter总分3
1班	552	1192	552
2班	277	710	277
3班	259	463	259
总计	1088	2365	1088

所谓的筛选上下文，就是度量值，度量值自带天然的Calculate函数  
没有筛选功能的新建列就是行上下文，如果行上下文想转成筛选上下文，它是不会自动转的，你要手动套上一个Calculate函数  
SUM('成绩表'[分数]) 不是度量值，【总分】才是度量值。



## 07.ALL函数、Allexcept函数、ALLSELECTED函数

2020年12月8日 19:11

一、ALL函数【作用：清除筛选，返回：清除筛选后的表格或列】

商品表中商品总数 = CountRows('商品表')

销售表中的商品数量 = Calculate(CountRows('商品表'), '销售表')

不能筛选的总数 = CountRows(ALL('商品表'))

商品表中商品总数	销售表中的商品数量	不能筛选的总数
26	8	26

CountRows(ALL('商品表'))

CountRows('商品表')

语法:

ALL(表)

ALL(表[列])

占比 = [销售表中的商品数量] / [不能筛选的总数]

商品表中商品总数	销售表中的商品数量	不能筛选的总数	占比
26	8	26	30.77%

案例一【应用于表】：计算销售表中每个商品的占比

总销量 = SUM('销售表'[销售数量])

禁止筛选的总销量 = CALCULATE([总销量],all('销售表'))

每个商品的占比 = [总销量] / [禁止筛选的总销量]

商品编码	总销量	禁止筛选的总销量	每个商品的占比
A002	99	495	20.00%
A005	11	495	2.22%
A007	33	495	6.67%
A008	77	495	15.56%
A009	55	495	11.11%
A012	44	495	8.89%
A014	22	495	4.44%
A019	154	495	31.11%
总计	495	495	100.00%

总结：当ALL参数为表时，忽略所有的筛选条件，无论是该图表内还是外部切片器

如果在矩阵表上将商品编码换成商品名称，会出现下在的情况

商品名称	总销量	禁止筛选的总销量	每个商品的占比
A		495	
B	99	495	20.00%
C		495	
D		495	
E	11	495	2.22%
F		495	
G	33	495	6.67%
H	77	495	15.56%
I	55	495	11.11%
总计	495	495	100.00%

内部筛选

ALLSELECTED函数替换ALL函数解决占比100%

解决方法：如果想在矩阵表上使用商品名称，可以用LookupValue将商品名称V过来，或者不使用红框内的条件

【新建列】商品名称 = LOOKUPVALUE('商品表'[商品名称], '商品表'[商品编码], '销售表'[商品编码])

外部筛选同样被忽略，比如使用销售表中的日期做切片器，禁止筛选的总量这一列仍然不受影响

商品名称	总销量	禁止筛选的总销量	每个商品的占比
A		495	
B	9	495	1.82%
C		495	
D		495	
E	1	495	0.20%
F		495	
G	3	495	0.61%
H	7	495	1.41%
I	5	495	1.01%
总计	45	495	9.09%

日期

☒ 2020年12月1日

☐ 2020年12月2日

案例二【应用于列】：

由于商品表中，没有大类这个列，我们先把大类V过来再测试

【新建列】大类 = LOOKUPVALUE('商品表'[大类], '商品表'[商品编码], '销售表'[商品编码])

【新建列】规格 = LOOKUPVALUE('商品表'[规格], '商品表'[商品编码], '销售表'[商品编码])

【度量值】取消列筛选 = CALCULATE([总销量], all('销售表'[规格]))

大类	总销量	取消列筛选	日期	规格
食品	165	165	<input type="checkbox"/> 2020年12月1日	<input type="checkbox"/> 袋
用品	330	330	<input type="checkbox"/> 2020年12月2日	<input type="checkbox"/> 盒
总计	495	495		

可以筛选      禁止筛选

大类	总销量	取消列筛选	占比
食品	165	165	100.00%
袋	154	165	93.33%
盒	11	165	6.67%
用品	330	330	100.00%
袋	99	330	30.00%
盒	231	330	70.00%
总计	495	495	100.00%

占比 = [总销量]/[取消列筛选]

总结：当ALL参数为列时，忽略该列筛选，其它图表字段或外部筛选对其产生作用

注意：ALL函数在引用列的时候，必需与矩阵的行和列在同一张表

假设，你ALL引用的是销售表，但是矩阵行和列来自商品表就会出错。

Allexcept函数：除...之外

语法：Allexcept(表名[列名])      等同于    All(表名[列名1],表名[列名2],表名[列名3],.....)

## 08.ALLNOBLANKROW返回表中除空白行以外的所有行

2020年12月9日 16:54

例如：新建表

表 = ALLNOBLANKROW('子表')

表 = ALLNOBLANKROW('子表'[姓名])

姓名
孙悟空
猪八戒
沙僧
小白龙
唐僧

返回：去重（即便只有一列，它也是表）

行数 = COUNTROWS('子表')

行数2 = COUNTROWS(all('子表'))

行数3 = COUNTROWS ( ALLNOBLANKROW ( '子表' ) )

姓名	行数	行数2	行数3
沙僧	1	7	7
孙悟空	2	7	7
唐僧	1	7	7
小白龙	1	7	7
猪八戒	2	7	7
总计	7	7	7

姓名字段来是子表。父表中姓名是唯一的，不用看有几行  
ALL与ALLNOBLANKROW看不出区别

行数 = COUNTROWS('子表')

行数2 = COUNTROWS(all('父表'))

行数3 = COUNTROWS ( ALLNOBLANKROW ( '父表' ) )

姓名	行数	行数2	行数3
沙僧	1	5	4
孙悟空	2	5	4
唐僧	1	5	4
小白龙	1	5	4
猪八戒	2	5	4
总计	7	5	4

姓名来自子表时，能看出来ALLNOBLANKROW比ALL少1，问题  
就出来父表中没有唐僧

姓名	行数	行数2	行数3
	1	5	4
沙僧	1	5	4
孙悟空	2	5	4
小白龙	1	5	4
猪八戒	2	5	4
总计	7	5	4

姓名来自父表时就明显了，空出那一行，就是唐僧

总结：

ALL函数会直接为父表添加一行“空行”，这一行ALL函数计算在内。  
ALLNOBLANKROW则会忽略这一空行，只计算父表中存在的数据

案例：

ALL与ALLNOBLANKROW 最常用的方式就是这样用，核对差异

公司：咦？我给你调货只配了售价为500元的货物，你咋销售了1000元？咋回事？兄弟？

门店：我也不知道啊！要不咱俩来核对一下？

总部&分店：（¥ ¥ % # & @ !）这咋核对啊！！

商品名称 总销量 ALL总销售

	5	115
冬瓜	33	115
胡萝卜	44	115
土豆	22	115
西红柿	11	115
总计	115	115

总销量 = sum('销售表'[销售数量])  
ALL总销售 = CALCULATE ([总销量], ALL ( '商品表' ) )  
  
ALLNOBLANKROW总销售 = CALCULATE ([总销量], ALLNOBLANKROW( '商品表' ) )

名称 总销量 ALL总销售 ALLNOBLANKROW总销售

冬瓜	33	33	33
胡萝卜	44	44	44
键盘	5	5	
土豆	22	22	22
西红柿	11	11	11
总计	115	115	110

# 09.聚合函数与迭代函数

2020年12月9日 23:25

# 01.聚合函数

2020年12月9日 23:29

总钱数 = SUM('表'[捡钱])

平均钱数 = AVERAGE('表'[捡钱]) # 算数平均值

1					
2		AVERAGE(A1:A4)			
a		AVERAGEA(A1:A4)			
3					

Average: (1+2+3)/3  
Averagea: (1+2+3)/4

最大值 = Max('表'[捡钱])

最小值 = Min('表'[捡钱])

记录数 = Countrows('表')

人数 = DISTINCTCOUNT('表'[姓名])

CountA函数：计算列中单元格不为空的数目

Countblank函数：计算列中单元格为空白数量

Product函数：计算列中单元格乘积

## 02.迭代函数

2020年12月9日 23:50

净值 = [捡钱]-[丢钱]

列 = sumx('表',[捡钱]-[丢钱])

列 = CALCULATE((sumx('表',[捡钱]-[丢钱])))

如果这两列不在同一张表里怎么办？用关系函数，详见笔记02.02

计算过程：

日期	姓名	捡钱	丢钱
2020/12/1	孙悟空	1	0.1
2020/12/1	猪八戒	2	0.9
2020/12/1	沙僧	3	0.3
2020/12/2	孙悟空	4	0.1
2020/12/2	猪八戒	5	0.8

对每一行逐行扫描，比如：

第一行， $1-0.1=0.9$

第二行， $2-0.9=0.1$

.....

第五行， $5-0.8=4.2$

将所有值求和，因为是行上下文，没有筛选功能，所以所有行都是最后那个总数

如果没有Sumx函数：

净值 = [捡钱]-[丢钱]

求和=sum(净值)

语法：sumx(表, 算术表达式)

解释：将每一行按算术表达式计算后，再将计算结果求和。

AverageX MaxX MinX CountX CountaX ProductX.... 它们与Filter函数一样都是行上下文函数

## 10. Earlier函数【当前行】

2020年12月10日 14:42

### 案例一：计算下一个订单日期

下个订单日期 = SUMX(FILTER('表','表'[序号]=EARLIER('表'[序号])+1),'表'[销售日期])

序号	销售日期	商品	销售量	下个订单日期
1	2020年12月1日	电视	1	2020/12/1 0:00:00
2	2020年12月1日	手机	2	2020/12/2 0:00:00
3	2020年12月2日	电视	3	2020/12/3 0:00:00
4	2020年12月3日	手机	4	2020/12/4 0:00:00
5	2020年12月4日	电视	5	2020/12/5 0:00:00
6	2020年12月5日	电视	6	2020/12/6 0:00:00
7	2020年12月6日	手机	7	

分析：FILTER('表','表'[序号]=EARLIER('表'[序号])+1)

将当前行序号+1的表筛选出来

这里用哪一个迭代函数都可以，SumX AverageX MinX MaxX都可以

因为只有一个日期，求和，求平均，求最大，求最小，都是一个结果

### 案例二：累计求和

累计销量 = SUMX(FILTER('表','表'[序号]<=EARLIER('表'[序号])), '表'[销售量])

序号	销售日期	商品	销售量	累计销量
1	2020年12月1日	电视	1	1
2	2020年12月1日	手机	2	3
3	2020年12月2日	电视	3	6
4	2020年12月3日	手机	4	10
5	2020年12月4日	电视	5	15
6	2020年12月5日	电视	6	21
7	2020年12月6日	手机	7	28

分析：FILTER('表','表'[序号]<=EARLIER('表'[序号]))

将小于等于当前行序号的表筛选出来

因为肯定不是一行，所以这个时候，求和用SumX 求平均用AverageX

### 案例三：分组累计求和

商品累计求和 = SUMX(FILTER('表','表'[序号]<=EARLIER('表'[序号]) && '表'[商品]=EARLIER('表'[商品])), '表'[销售量])

序号	销售日期	商品	销售量	商品累计求和
1	2020年12月1日	电视	1	1
2	2020年12月1日	手机	2	2
3	2020年12月2日	电视	3	4
4	2020年12月3日	手机	4	6
5	2020年12月4日	电视	5	9
6	2020年12月5日	电视	6	15
7	2020年12月6日	手机	7	13

分析：FILTER('表','表'[序号]<=EARLIER('表'[序号]) && '表'[商品]=EARLIER('表'[商品]))

小于当前行序号且等于当前行商品的表筛选出来

同时考虑日期和商品进行求和，比如：累计计算电视每天的累加值，手机每天的累加值

累计求和 = SUMX(FILTER('表','表'[销售日期]<=EARLIER('表'[销售日期]) && '表'[商品]=EARLIER('表'[商品])), '表'[销售量])

序号	销售日期	商品	销售量	累计求和
1	2020年12月1日	电视	1	1
2	2020年12月1日	手机	2	2
3	2020年12月2日	电视	3	4
4	2020年12月3日	手机	4	6
5	2020年12月4日	电视	5	9
6	2020年12月5日	电视	6	15
7	2020年12月6日	手机	7	13

### 案例四：累计购买次数

第几次购买 = COUNTROWS(FILTER('表1','表1'[姓名]=EARLIER('表1'[姓名]) && '表1'[序号]<=EARLIER('表1'[序号])))

序号	姓名	第几次购买
1	张三	1
2	李四	1
3	张三	2
4	李四	2
5	王五	1
6	张三	3
7	王五	2
8	李四	3
9	张三	4

分析：FILTER('表1','表1'[姓名]=EARLIER('表1'[姓名]) && '表1'[序号]<=EARLIER('表1'[序号]))

将 序号比当前行小的 与 等于当前行姓名的 表筛选出来



序号	姓名	第几次购买
1	张三	1
2	李四	1
3	张三	2
4	李四	2
5	王五	1
6	张三	3
7	王五	2
8	李四	3
9	张三	4
10	李四	4

分析：FILTER('表1','表1'[姓名]=EARLIER('表1'[姓名]) && '表1'[序号]<=EARLIER('表1'[序号]))

将 序号比当前行小的 与 等于当前行姓名的 表筛选出来

# 11.Values与Distinct区别

2020年12月9日 23:10

第05课，我们已经体验了Values的作用

Values(表[列]) 表1 = VALUES('表'[姓名]) # 把姓名不重复的列提取出来，形成一个单列的表 【人造基础表（单列）】

Values(表) 表2 = VALUES('表') # 返回表的所有可见行，就是复制一张表 【可见行三个字是废话】 【人造虚拟表】

DISTINCT(表名[字段名]) 返回：去重后，唯一值的列

DISTINCT(表名) 返回：只包含非重复行的表

DISTINCT(返回表的表达式)

表3 = DISTINCT('表1'[姓名])

姓名
孙兴华
张三
李四
王五

表4 = DISTINCT('表1')

姓名	性别
孙兴华	男
张三	男
李四	男
王五	男
张三	女

注意，必须是整行的与其它行重复才被认定为重复

表5 = DISTINCT(FILTER('表1','表1'[性别]="男"))

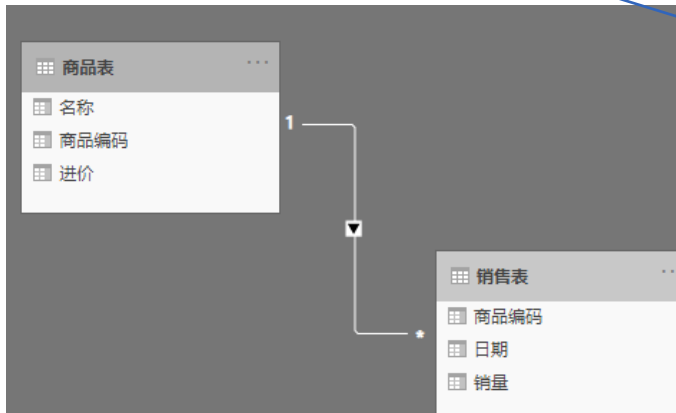
姓名	性别
孙兴华	男
张三	男
李四	男
王五	男

## 对于空白行的区别：

2020年12月10日 19:26

对于空白行的处理上，VALUES包括没有匹配的空白行。但是DISTINCT不返回没有匹配的空白行。

专业术语：参照完整性不匹配



说人话：销售表上有的数据，基础表上没有

行数 = COUNTROWS('商品表')

Values姓名 = COUNTROWS(VALUES('商品表'[商品编码]))

Distinct姓名 = COUNTROWS(Distinct('商品表'[商品编码]))

Values进价 = COUNTROWS(VALUES('商品表'[进价]))

Distinct进价 = COUNTROWS(Distinct('商品表'[进价]))

名称	行数	Values姓名	Values进价	Distinct姓名	Distinct进价
		1	1		
冬瓜	1	1	1	1	1
茄子	1	1	1	1	1
土豆	1	1	1	1	1
总计	3	4	3	3	3

## 12.条件判断函数[基础]

2020年12月11日 17:37

# 01.IFERROR 遇到错误时使用指定数值替换

2020年12月11日 17:44

销售金额 = [销售数量]\*[单价]

销售数量	单价	销售金额
100	0.1	#ERROR
孙兴华	1	#ERROR
300	3	#ERROR
2	5	#ERROR

孙兴华不能跟数字进行计算

销售金额 = IFERROR([销售数量]\*[单价],BLANK())

销售数量	单价	销售金额
100	0.1	10
孙兴华	1	
300	3	900
2	5	10

这里必须是数值

销售金额 = IFERROR([销售数量]\*[单价],"不能计算") 错误!!!

销售数量	单价	销售金额
100	0.1	#ERROR
孙兴华	1	#ERROR
300	3	#ERROR
2	5	#ERROR

销售金额 = IFERROR([销售数量]\*[单价],999)

销售数量	单价	销售金额
100	0.1	10
孙兴华	1	999
300	3	900
2	5	10

## 02.if 条件判断

2020年12月11日 18:01

If (西瓜坏了，换货，**[退货]**)

第三参数可以省略不写，省略时返回为空。

If适合条件比较少的时候使用，如果条件太多，嵌套太多，不方便使用

称呼 = IF([性别]="男","先生","女士")

性别	称呼
男	先生
女	女士
男	先生
女	女士

案例5：

间隔 = IF([取款日期]=BLANK(), BLANK(), [取款日期]-[存款日期])

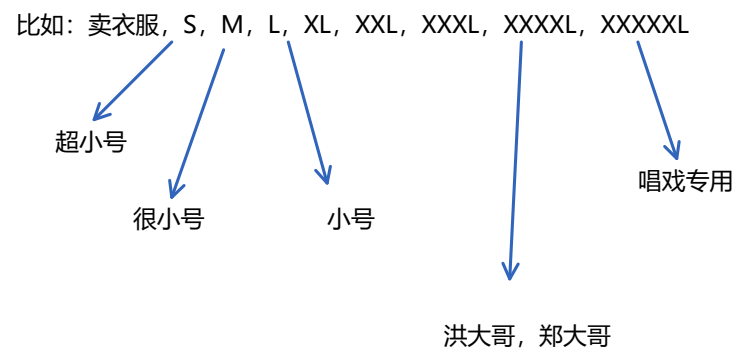
序号	存款日期	取款日期	间隔
1	2020年12月1日	2020年12月2日	1
2	2020年12月2日	2020年12月5日	3
3	2020年12月5日	2020年12月7日	2
4	2020年12月7日	2020年12月9日	2
5	2020年12月9日	2020年12月10日	1
6	2020年12月10日	2020年12月11日	1
7	2020年12月11日	2020年12月12日	1
8	2020年12月12日	2020年12月12日	0
9	2020年12月12日	2020年12月14日	2
10	2020年12月14日	2020年12月15日	1
11	2020年12月15日		

### 03.switch多项条件判断

2020年12月11日 18:08

switch (表达式, 值1, 结果1, ..., [else])  
最后一个参数可以省略不写, 省略时返回为空。

```
月份 = SWITCH('例3'[月],  
1, "一月", 2, "二月",  
3, "三月", 4, "四月",  
5, "五月", 6, "六月",  
7, "七月", 8, "八月",  
9, "九月", 10, "十月",  
11, "十一月", 12, "十二月",  
"未能识别")
```



Switch特殊用法:

**使用 TRUE 作为第一参数的作用是, 返回条件判断 List 中为第一个为 TRUE 的结果**

```
年龄段 = SWITCH(  
TRUE(),  
'例4'[年龄] < 30, "30岁以下",  
'例4'[年龄] < 40, "30-40岁",  
'例4'[年龄] < 50, "40-50岁",  
"50岁以上"  
)
```

如果是多个字段同时判断呢? 假设:  
'例4'[年龄] < 30 && '例4'[性别] = "男"

# 13.安全除法DIVIDE与按层级计算ISINSCOPE函数

2020年12月11日 19:18



# 01.安全除法

2020年12月11日 18:44

语法：DIVIDE(分子,分母,[替换结果])  
替换结果可以省略不写，省略时返回为空。

除法 = [分子]/[分母]

分子	分母	除法
10	5	2
3	0	∞
16	4	4

安全除法 = DIVIDE([分子],[分母],BLANK())  
安全除法 = DIVIDE([分子],[分母],0)

## 02.按层级计算ISINSCOPE函数

2020年12月11日 19:20

产品类别	占比 层级	商品名称
☐ 手机	46.12%	■ 充电器
小米10	16.84%	■ 耳机
小米8	83.16%	☐ 手机壳
☐ 手机配件	53.88%	■ 数据线
充电器	57.66%	☐ 贴膜
耳机	16.22%	■ 小米10
数据线	26.13%	☐ 小米11
总计	100.00%	■ 小米8
		☐ 小米9

**ISINSCOPE函数**的意思就是是否在范围内，官方释义为：当指定的列是级别层次结构中的级别时，返回true。

总金额 = SUM('销售表'[销售])

占比 层级 =

SWITCH(TRUE(),

ISINSCOPE('商品表'[商品名称]), DIVIDE([总金额],CALCULATE([总金额],ALLSELECTED('商品表'[商品名称]))),

ISINSCOPE('商品表'[产品类别]), DIVIDE([总金额],CALCULATE([总金额],ALLSELECTED('商品表'[产品类别]))),

DIVIDE([总金额],CALCULATE([总金额],ALLSELECTED('商品表'[商品名称]))))

## 14.ISCROSSFILTERED函数和ISFILTERED函数的区别

2020年12月11日 19:48

**IS——是的意思；**

**CROSS——交叉的意思；**

**FILTERED——筛选的意思；**

**因此，**

**ISCROSSFILTERED函数代表判断是否受到交叉筛选影响； 【需要进行判断的表或列】**

**ISFILTERED函数代表判断是否受到直接筛选影响。【只能针对列】**

**ISFILTERED函数用于判断某一字段是否有筛选关系**

**ISCROSSFILTERED与它相近，它用于判断某一个表中是否含有筛选字段**

## 01.ISFILTERED函数判断是否被筛选

2020年12月11日 20:05

### ISFILTERED

#### ISFILTERED(列名)

当直接筛选【列名】时，返回 TRUE。

如果列没有筛选器，或由于正在筛选同一个表或相关表中的不同列而发生筛选，则函数返回 FALSE

ISFILTERED函数用于判断某一字段是否有筛选关系

ISCROSSFILTERED与它相近，它用于判断某一个表中是否含有筛选字段

语法：

#### ISCROSSFILTERED

#### ISCROSSFILTERED(列名)

当正在筛选【列名】或相同表或相关表中的其他列时，返回 TRUE

比如我们将某个表中的某一个字段当做筛选器，这个函数的返回值则为TRUE。

案例：先做一张日期表，切片器使用日期表中的【年月】

剪贴板	数据	查询
年月	  > ...	ISCROSSFILTERED ISFILTERED
<input type="checkbox"/> 202001		True FALSE
<input type="checkbox"/> 202002		
<input checked="" type="checkbox"/> 202003		
<input type="checkbox"/> 202004		
<input type="checkbox"/> 202005		
<input type="checkbox"/> 202006		
<input type="checkbox"/> 202007		
<input type="checkbox"/> 202008		

ISFILTERED = ISFILTERED ( '日期表'[Date])

ISCROSSFILTERED = ISCROSSFILTERED ( '日期表'[Date])

## 02.日期表

2020年12月11日 20:09

```
日期表 = ADDCOLUMNS(  
    CALENDAR(date(2019,1,1),date(2020,12,31)),  
    "年", YEAR ( [Date] ),  
    "季度", ROUNDUP(MONTH([Date])/3,0),  
    "月", MONTH([Date]),  
    "周", weeknum([Date]),  
    "年季度", year([date]) & "Q" &  
    ROUNDUP(MONTH([Date])/3,0),  
    "年月", year([Date]) * 100 + MONTH([Date]),  
    "年周", year([Date]) * 100 + weeknum([Date]),  
    "星期几", WEEKDAY([Date])  
)
```

### 03.案例分析

2020年12月11日 20:18

分析企业销售额的时候想要选中某一个月，然后可以观察到之前N个月份的情况，应该怎么做呢？

案例：展示某指定年月之前12个月份的销售额

- (1) 做一张日期表【注意不要连线】
- (2) 在日期表里新建【年月序号】列

```
年月序号 = SWITCH(  
TRUE(),  
'日期表'[年]=2019,0+[月],  
'日期表'[年]=2020,12+[月]  
)
```

- (3) 打开今天的数据
- (4) 新建度量值：作用往前显示一年的数据

```
前12个月 = IF (  
ISCROSSFILTERED ( '日期表'[年月] ),  
CALCULATE (  
VALUES ( '表'[年月] ),  
FILTER ( '表', '表'[年月序号] <= MAX ( '日期表'[年月序号] ) && '表'[年月序号] > MAX ( '日期表'[年月序号] ) - 12 )  
) ,  
1  
)
```

- (5) 切片器用日期表中的字段，矩阵里用数据表里面的年月

年月	销售金额	年月的计数
201901	2215	2
201902	1785	1
201903	2557	1
201904	1485	1
201905	2710	1
201906	2608	1
201907	1487	1
201908	1379	1
201909	1433	1
201910	2039	1
201911	2802	1
201912	2169	1
总计	24669	13

- (6) 将度量值放到外部筛选器

筛选器

搜索

此视觉对象上的筛选器

年月

是(全部)

前12个月

不为空

√

×

🔒

🔍

👁

销售金额

是(全部)

年月的计数

是(全部)

## 15.FIRSTNONBLANK与LASTNONBLANK函数

2020年12月11日 23:29

FIRSTNONBLANK函数与LASTNONBLANK函数属于“表”函数，当然，有些时候也可以作为“值函数”使用。

语法：

FIRSTNONBLANK(<列>,<表达式>)

LASTNONBLANK(<列>,<表达式>)

列：任何列，或者具有单列的表，也可以是表达式。

表达式：计算空值的表达式，也就是判定条件。

返回结果：单列的表，只有一行数据。也就是只返回一个值，只不过这个值在表中。

案例：应用于表函数

表 2 = FIRSTNONBLANK ( '表'[日期], CALCULATE ( SUM ( '表'[销售] ) ) )

求出产生第一笔销售额的日期。如果这里不嵌套CALCULATE函数的话，会导致计算忽略计算上下文，只考虑行上下文，那么呈现的结果将是2020年1月1日。

案例：应用于值函数

每人第一个不为空的销售 = CALCULATE ( SUM ( '表'[销售] ), FIRSTNONBLANK ( '表'[日期], CALCULATE ( SUM ( '表'[销售] ) ) ) )

姓名	每人第一个不为空的销售
李四	111
王五	135
张三	100
总计	246



# 16.进阶条件判断函数

2020年12月11日 23:53

## 01.HASONEFILTER函数【判断是否被筛选】

2020年12月11日 22:11

语法: HASONEFILTER (列名 )

HASONEFILTER 是被直接筛选影响的。

当指定的列有且只有一个由直接筛选产生的值时，返回 true。参数必须使用物理列，不支持返回列的表达式

案例1:

姓名	总分	老师	HASONEFILTER总成绩	姓名
赵六	198	宋浩	99	<input type="checkbox"/> 李四
总计	198	叶问	99	<input type="checkbox"/> 王五
		总计	198	<input type="checkbox"/> 张三
				<input checked="" type="checkbox"/> 赵六

如果这里用[总分]  
那永远显示所有老师  
当[姓名]被筛选时，只显示被筛选出来的这些人所对应的老师

总分 = SUM('案例1'[成绩])

HASONEFILTER总成绩 = if(HASONEFILTER('案例1'[姓名]),[总分],Blank())

案例2:

总金额 = SUM('案例2'[销售金额])

外汇转人民币 = If(HASONEFILTER('汇率'[货币]),Firstnonblank('汇率'[汇率],"")\*[总金额],Blank())

人民币转外汇 = If(HASONEFILTER('汇率'[货币]),Divide([总金额],Firstnonblank('汇率'[汇率],1)),Blank())

Firstnonblank 返回按当前上下文筛选的 column 列中的第一个值

lastnonblank 返回按当前上下文筛选的 column 列中的最后一个值

货币	日期	总金额	人民币转外汇
<input type="checkbox"/> 美元	2020/12/1	10000	
<input type="checkbox"/> 欧元	2020/12/2	20000	
<input type="checkbox"/> 日元	2020/12/3	30000	
	2020/12/4	40000	
	2020/12/5	50000	
	总计	150000	

## 02.HASONEVALUE函数【判断是否只有一行数据】

2020年12月11日 23:53

语法: Hasonevalue(列)

通常和If函数搭配使用, 判断某列是否只有一行数据

经典语句: `if(Hasonevalue( '表名' [列名]),[度量值],blank())`

案例1:

总分 = SUM('案例1'[成绩])

姓名唯一的人分数 =if(HASONEVALUE('案例1'[姓名]),[总分],BLANK())

老师	姓名唯一的人分数
李小龙	64
叶问	99
总计	

因为李小龙和叶问这两个老师是唯一的, 没有重复的, 所以只显示他们的分数

当你熟悉了另一个函数 SELECTEDVALUE 之后, 你会发现它的写法更简单, 比如上面的度量值等价于  
姓名唯一的人分数2=SELECTEDVALUE('案例1'[成绩])

那什么时候用if+HASONEVALUE, 一般想让总计不显示的时候我们会使用, 我们讲RankX函数时再介绍

## 【扩展知识】DATATABLE人工建表

2020年12月12日 1:34

参数表 =DATATABLE("字段名",STRING,{{"数据11"},"数据12"}})

字段名
数据11
数据12

数据类型包括：INTEGER、DOUBLE、STRING、BOOLEAN、CURRENCY、DATETIME

多字段这样写：

参数表 = DATATABLE("字段名1",STRING,"字段名2",STRING,{{"数据11","数据12"},"数据21","数据22"}})

字段名1	字段名2
数据11	数据12
数据21	数据22

### 03.SELECTEDVALUE函数

2020年12月12日 0:50

当指定列中只有一个值时返回该值，否则返回备选结果，省略备选结果时返回空值

当指定列中只有一个值时返回该值，否则返回替代结果，省略则返回空值。

SELECTEDVALUE 与 IF+HASONEVALUE 等价，而且公式更简洁

if(Hasonevalue( '表名' [列名]) ,返回值,blank())



- 如果是唯一值，那就返回值，否则返回空

SELECTEDVALUE ( '表名' [列名], 【代替 (省略返回空) 】 )



- 当指定列中只有一个值时返回该值，否则返回替代结果，省略则返回空值。

#### 案例3：

##### 人工建表：

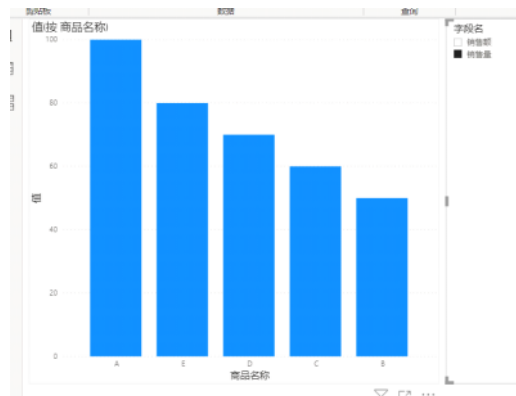
参数表 =DATATABLE("字段名",STRING,{{"销售量"},"销售额"}})

##### 度量值：

值 = SWITCH(SELECTEDVALUE('参数表'[字段名]),  
"销售量",SUM('案例3'[销售量]),  
"销售额",SUM('案例3'[销售金额]))

柱状图：轴-商品名称，值-值

切片器：参数表中的字段名



# 17.转换函数

2020年12月12日 14:53

# 01.CURRENCY函数 【数字转货币】

2020年12月12日 15:01

作用：将表达式结果转换为货币类型

- (1) CURRENCY 函数四舍五入第 5 位有效小数的值，返回第 4 位小数;如果第 5 位有效小数等于或大于 5，则进行舍入。例如：5.88888888转换后得到 \$5.8889。5.44444444转换后得到 \$5.4444
- (2) 如果表达式的数据类型是 True/False，那么 CURRENCY()将为真值返回\$1.0000，为假返回\$0.0000。
- (3) 如果表达式的数据类型是文本，则 CURRENCY()将尝试将文本转换为数字;如果转换成功，数字将被转换为货币，否则将返回一个错误。例如："5.88888888"是可以转换成功的。但是"孙兴华"肯定无法转转换。
- (3) 如果表达式的数据类型是 DateTime，那么 CURRENCY()将把 DateTime 值转换为一个数字，然后将这个数字转换为货币。DateTime 值包含一个整数部分，表示给定日期到 1899-12-30 之间的天数，以及一个分数，表示一天的分数(其中 12 小时或中午是 0.5 天)。如果表达式的值不是正确的 DateTime 值，则返回一个错误。



- 1899-12-29 第-1天
- 1899-12-30 第0天
- 1899-12-31 第1天
- 1900-1-1 第2天

## 02.INT函数【向下舍入】

2020年12月12日 15:01

结构	
1 列 = INT([数值])	
数值	列
-2.9	-3
-2.1	-3
-0.9	-1
-0.1	-1
0.1	0
0.9	0
2.1	2
2.9	2

将表达式转换为整数(向下舍入)

说人话：

当数值为正数的时候，取整数抹零

当数值为负数的时候，凑零补整



## 03.TRUNC函数【直接取整】

2020年12月12日 15:01

结构	
<div>✕ ✓</div> <div>1 列 = TRUNC([数值])</div>	
数值	列
5.689	5
3.444	3
0	0
-3.444	-3
-5.689	-5

# 04.ROUND函数【四舍五入】

2020年12月12日 15:30

✕	✓	1 列 = ROUND([数值],2)
数值	列	
5.4444	5.44	
3.8888	3.89	
0	0	
-3.8888	-3.89	
-5.4444	-5.44	

语法：ROUND(数值,保留小数点的位数)

注意：当数值为0时，无论你小数点保留几位都是0

## 05.MROUND【取数值的倍数】

2020年12月12日 15:33

语法: MROUND(数值, 哪个数的倍数)

MROUND(1.3,0.2) => 1.4

MROUND(-10,-3) => -9

MROUND(5,-2) => error

注意: 符号必须一致



数值	转换
6.05	6
7.05	7.1

当为小数值舍入时, 最终的结果的位数是不确定的。这是 Excel 中一个已知 BUG, DAX 继承了此 BUG。

讲人话: MROUND(数值, 哪个数的倍数)

小数点位数要一致

## 06.日期和时间的转换

2020年12月12日 15:42

**DATE ( 年, 月, 日 )**

**TIME ( 小时, 分钟, 秒 )**

## 07.CONVERT函数【转换为指定数据类型】

2020年12月12日 15:52

**CONVERT (表达式, 数据类型 )**

**INTEGER整型、DOUBLE双精度、STRING字符、BOOLEAN布尔、CURRENCY货币、DATETIME日期时间**

**列 = CONVERT([单价]\*[销售数量],STRING)**

**列 = CONVERT([单价],STRING)**

# 09.DATEVALUE函数【文本日期转日期格式】

2020年12月12日 17:02

<div>✕ ✓</div> <div>1 列 = DATEVALUE([文本日期])</div>	
文本日期	列
2020年7月30日	2020/7/30 0:00:00
1980年7月30日	1980/7/30 0:00:00
2020年7月30日	2020/7/30 0:00:00

列 = DATEVALUE([文本日期])

## 10.TIMEVALUE函数

2020年12月12日 17:21

TIMEVALUE

TIMEVALUE ( <TimeText> )

将文本格式的时间转换为日期时间格式

# 11.VALUE【将文本转数值】

2020年12月12日 20:11

**VALUE ( <文本> )**

**返回结果是：数值类型，十进制数字**

**如果文本不是常量、数字、日期或时间格式。 则返回错误。**



## 18.FORMAT函数【格式化】

2020年12月12日 16:03

### FORMAT(数值或日期, 格式)

将值转换为指定数字格式的文本, 通常与 DateTime 格式一起使用。

1 列 = FORMAT(DATE([年],[月],[日]),"yyyy年mm月dd日")			
年	月	日	列
1980	7	30	1980年07月30日
1983	10	7	1983年10月07日
1986	12	9	1986年12月09日
1984	5	4	1984年05月04日

注意几个特殊情况:

1 列 = FORMAT(DATE([年],[月],[日]),"""孙兴华"" yyyy")			
年	月	日	列
1980	7	30	孙兴华 1980
1983	10	7	孙兴华 1983
1986	12	9	孙兴华 1986
1984	5	4	孙兴华 1984

列 = FORMAT(DATE([年],[月],[日]),""孙兴华"" yyyy)

1 列 = FORMAT(DATE([年],[月],[日]),"yyyy \Qq")			
年	月	日	列
1980	7	30	1980 Q3
1983	10	7	1983 Q4
1986	12	9	1986 Q4
1984	5	4	1984 Q2

列 = FORMAT(DATE([年],[月],[日]),"yyyy \Qq")

1 列 = FORMAT(DATE([年],[月],[日]),"dd/mm/yyyy")			
年	月	日	列
1980	7	30	30/07/1980
1983	10	7	07/10/1983
1986	12	9	09/12/1986
1984	5	4	04/05/1984

列 = FORMAT(DATE([年],[月],[日]),"dd/mm/yyyy")

1 列 = FORMAT(TIME([时],[分],[秒]),"h:nn:ss")			
时	分	秒	列
12	35	15	12:35:15
7	8	21	7:08:21
21	15	33	21:15:33

列 = FORMAT(TIME([时],[分],[秒]),"h:nn:ss")

"hh:nn:ss"  
07:08:21

等价于"ttttt"

1 列 = FORMAT(TIME([时],[分],[秒]),"hh:mm:mm AMPM")			
时	分	秒	列
12	35	15	12:35:12 下午
7	8	21	07:08:12 上午
21	15	33	09:15:12 下午

## 附1：日期格式

2020年12月12日 16:40

格式参数	输出	特殊说明
d	1	
dd	01	
ddd	Mon	
dddd	Monday	
dddddd	1980/7/30	
dddddd	1980年7月30日	
aaa	周一	
aaaa	星期一	
m	1	
mm	01	
mmm	Jan	
mmmm	January	
OOO	1月	
OOOO	一月	
q	1	
yy	20	
yyyy	2020	
yyyymm	202007	
yyyymmdd	20200730	
w	3	将一周中的天显示为数字（1 代表星期天，依次排列到 7，7 代表星期六）
ww	31	将一年中的周显示为数字 (1-54)。
h	9	将小时显示为不带前导零的数字 (0-23)
hh	09	将小时显示为带有前导零的数字(00-23)。
n	7	将分显示为不带前导零的数字 (0-59)。
nn	07	将分显示为带有前导零的数字(00-59)。
s	3	将秒显示为不带前导零的数字 (0-59)。
ss	03	将秒显示为带有前导零的数字 (00-59)
General Date	2020/7/30	显示日期和/或时间。例如，2008/3/12 上午 11:07:31。日期显示取决于应用程序的当前区域性值。
Long Date	2020年7月30日	
Medium Date	20-07-30	
Short Date	2020/7/30	使用你当前区域性的短日期格式显示日期。例如，2008/3/12
Long Time	0:00:00	使用你当前区域性的长时间格式显示时间；通常包括小时、分钟、秒。例如，上午 11:07:31。
Medium Time	11:00 AM	以 12 小时格式显示时间。例如，上午 11:07

Short Time	0:00	以 24 小时格式显示时间。例如，11:07
------------	------	------------------------

## 附2：预定义的数字格式

2020年12月12日 16:09

参数类型	参数说明（注意：返回的都是文本）
General Number	显示没有千分符的数字
Currency	显示带千分符的数字，对于符合要求的数字，在十进制分隔符的右边显示两个数字。输出的内容基于系统区域设置。
Fixed	在十进制分隔符的左边显示至少一个数字，右边显示至少两个数字。
Standard	显示带有千分符的数字，小数点分隔符的左边至少有一个数字，右边至少有两个数字。
Percent	将显示的数字乘以 100，并在右侧添加百分号(%)；始终在十进制分隔符的右边显示两个数字。
Scientific	使用标准科学符号，提供两个有效数字。
Yes/No	如果数字为 0，则显示 No；否则显示 Yes。
True/False	如果数字为 0，则显示 False；否则显示 True。
On/Off	如果数字为 0，则显示 Off，否则显示 On。

```
FORMAT( 12345.67, "General Number") = 12345.67
FORMAT( 12345.67, "Currency") = ¥12345.67
FORMAT( 12345.67, "Fixed") = 12345.67
FORMAT( 12345.67, "Standard") = 12,345.67
FORMAT( 12345.67, "Percent") = 1,234,567.00 %
FORMAT( 12345.67, "Scientific") = 1.23E+04
```

### 附3：自定义数字格式

2020年12月12日 16:15

格式参数	输出
0	1235
0.0	1234.6
0%	123456%
0.00%	123456.00%
0.00E+00	1.23E+03
#,##	1,235
#,## 0.00	1,234.56
¥ #,## 0.00	¥ 1,234.56

视频里没讲，但是这里有必要注明一下

**字符意义：**

- 0** 显示一数字，若此位置没有数字则补 0
- #** 显示一数字，若此位置没有数字则不显示
- %** 数字乘以 100 并在右边加上“ %” 号 字符

# 19.日期时间函数【非智能函数】

2020年12月12日 16:58

# 01.提取：年月日时分秒.季度.当前时间

2020年12月12日 16:59

时间日期	年	月	日	时	分	季度	当前时间
2020/7/30 12:01:30	2020	7	30	12	1	3	2020/12/12 17:20:20
1983/10/7 5:01:21	1983	10	7	5	1	4	2020/12/12 17:20:20

- 年 = YEAR([时间日期])
- 月 = MONTH([时间日期])
- 日 = DAY([时间日期])
- 时 = HOUR([时间日期])
- 分 = MINUTE([时间日期])
- 秒 = SECOND([时间日期])
- 季度 = QUARTER([时间日期])
- 当前日期和时间 = NOW()
- 当前日期 = TODAY()

## 02.星期和周

2020年12月12日 17:16

时间日期	当前日期在本周中的第几天	当前日期在本年中的第几周
2020/7/30 12:01:30	4	31
1983/10/7 5:01:21	5	41

当前日期在本周中的第几天 = WEEKDAY([时间日期],2)

当前日期在本年中的第几周 = WEEKNUM([时间日期],2)

注：第二参数，指从星期一开始算做一周的开始。



### 03.平移指定月份

2020年12月12日 17:36

EDATE ( 日期,平移月数 ) 【返回按指定月数平移后的日期】

EOMONTH函数 【返回指定月数平移后的月份的最后一天】

# 使用正数向后，使用负数向前

时间日期	EDATE	EOMNTH
2020/7/30 12:01:30	2020/9/30 0:00:00	2020/9/30 0:00:00
1983/10/7 5:01:21	1983/12/7 0:00:00	1983/12/31 0:00:00

例如：

EDATE ( 日期,平移月数 )

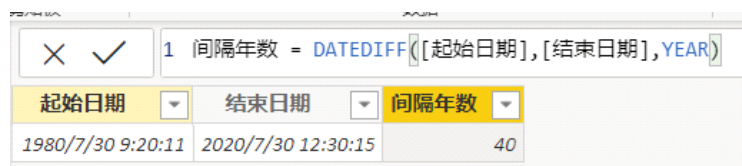
EOMONTH(TODAY(),-1)代表今天的日期向前推移一个月的月末日期。

## 04.DATEDIFF间隔日期时间

2020年12月12日 17:41

**DATEDIFF ( 起始日期, 结束日期, <间隔单位> )**

# 如果计算保质期要加1



## 05.YEARFRAC函数更精确计算间隔

2020年12月12日 18:05

**YEARFRAC ( 起始日期, 结束日期, <计算标准> )**

第一个参数：开始日期  
第二个参数：结束日期  
第三个参数：计算标准, (忽略时默认为0)  
0-美国30/360  
1-实际/实际  
2-实际/360  
3-实际/365  
4-欧洲30/360

yearfrac也是计算两个日期的间隔，不过这个间隔不用具体的年月天数来表示，而是通过相隔天数占一年的比例来表示。

起始日期	结束日期	DATEDIFF	YEARFRAC
1980/7/30 9:20:11	2020/7/30 12:30:15	40	40.3666666666667
1980/12/31 9:20:11	2020/12/31 12:30:15	40	39.95

例如：死亡证明上写的年龄，就必须是精准的。比如：你的生日是1980/12/31 今天是2020/12/30 那么你的死亡年龄就是39岁

## 20.文本函数

2020年12月12日 18:22

# 01.CONCATENATE【将两个字符串连接】

2020年12月12日 18:45

CONCATENATE ( <文本 1 或 数字 1>, <文本 2 或 数字 1> )

文本1	文本2	数字1	数字2	连接文本	连接数字
孙兴华	赵丽颖	520	1314	孙兴华赵丽颖	5201314

连接文本 = CONCATENATE([文本1],[文本2])

连接数字 = CONCATENATE([数字1],[数字2])

## 02.EXACT 【判断字符是否相同】

2020年12月12日 18:51

**EXACT ( <文本 1 或 数字 1>, <文本 2 或 数字 1> )**

文本1 ▾	文本2 ▾	数字1 ▾	数字2 ▾	文本判断 ▾	数字判断 ▾
孙兴华	赵丽颖	520	1314	False	False
孙兴华	孙兴华	520	520	True	True

**文本判断 = EXACT([文本1],[文本2])**

**数字判断 = EXACT([数字1],[数字2])**

**应用场景举例：配合if语句，如果这两列内容一样，返回什么，否则返回什么**

### 03.FIND【找字符串在另一个字符串的起始位置】

2020年12月12日 18:55

返回一个文本字符串在另一个文本字符串中的起始位置，**区分大小写，不支持通配符。**

**FIND ( <待查找内容>, <查找范围>, [<起始位置>], [<备选结果>] )**

参数	属性	描述
待查找内容		要找的文本。使用双引号(空文本)匹配查找中的第一个字符；不允许使用通配符
查找范围		包含要查找的文本的文本
起始位置	可选	开始搜索的位置；如果省略，则起始位置=1，也就是查找范围中的第一个字符
备选结果	可选	未找到文本时返回的数值，可设为 0、-1 或 BLANK()；如果省略，则返回一个错误

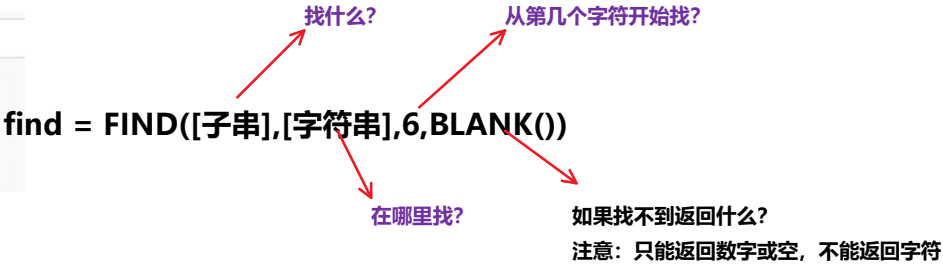
**FIND 不支持通配符。要使用通配符，请使用 SEARCH。**

×

✓

1 find = FIND([子串],[字符串],6,BLANK())

字符串	子串	find
跟着孙兴华学PowerBI	孙兴华	
孙兴华讲VBA	vba	
跟着孙兴华学PowerBI孙兴华	孙兴华	14



## 04.SEARCH【不区分大小写，可用通配符】

2020年12月12日 19:06

功能与Find相似，SEARCH不区分大小写，可用通配符

SEARCH ( <待查找内容>, <查找范围>, [<起始位置>], [<备选结果>] )

参数	属性	描述
待查找内容		要找的文本。使用双引号(空文本)匹配查找中的第一个字符；支持?和*通配符
查找范围		包含要查找的文本的文本
起始位置	可选	开始搜索的位置；如果省略，则起始位置=1，也就是查找范围中的第一个字符
备选结果	可选	未找到文本时返回的数值，可设为 0、-1 或 BLANK()；如果省略，则返回一个错误

字符串	子串	SEARCH	find
跟着孙兴华学PowerBI	孙兴华	3	3
孙兴华讲VBA	vba	5	
跟着孙兴华学PowerB孙兴华	孙兴华	3	3

find = FIND([子串],[字符串],,BLANK())

SEARCH = SEARCH([子串],[字符串],,BLANK())

通配符 = SEARCH("孙?华",[字符串],,BLANK())

通配符 = SEARCH("孙?华",[字符串],,BLANK())

如果要查找的字符是? 或\*号本身，需要在? 或\*前面加一个~转义字符



## 05.截取函数

2020年12月12日 19:25

**left从左向右取**=left([字段名],取几个字符)

**right从右向左取**=right([字段名],取几个字符)

**mid从中间开始取**=mid([字段名], 从第几个取, 取几个)

**len长度**=len([字段名])

例如：

文本
跟着孙兴华学习VBA

left([文本],2) # 返回：跟着

right([文本],3) # 返回：VBA

mid([文本],3,3) # 返回：孙兴华

len([文本]) # 返回：10

**应用场景举例：与Find与SEARCH配合使用，使用先用left将子串截取出来，再放到Find里查询**

## 06.FIXED 【数值转字符按指定小数位四舍五入】

2020年12月12日 19:24

将数值舍入到指定的小数位数并将结果返回为文本。 可以指定返回的结果是否包含逗号。

**FIXED ( <数字>, [<小数位数>], [<逻辑值>] )**

参数	属性	描述
数字		要舍入并转换为文本的数字，或包含数字的列。
小数位数	可选	小数点右侧的位数；如果省略，则为 2
逻辑值	可选	一个逻辑值：如果为 1，则不在返回的文本中显示逗号；如果为 0 或省略，则在返回的文本中显示逗号

**返回：一个字符串，文本形式的数字。**

数值	不带千位分隔符	带千位分隔符
99999.99999	100000.000	100,000.000
66666.44444	66666.444	66,666.444
0	0.000	0.000
-666666.44444	-666666.444	-666,666.444
-55555.99999	-55556.000	-55,556.000

第三参数，写1时不带千位分隔符。

第三参数，写0或省略时，带千位分隔符。

不带千位分隔符 = FIXED([数值],3,1)

带千位分隔符 = FIXED([数值],3)

## 07.大小写转换

2020年12月12日 20:00

**LOWER ( <文本> )** 将文本字符串中的所有字母都转换为小写。

**UPPER ( <文本> )** 将文本字符串转换为全大写字母。

## 08.删除空格

2020年12月12日 20:04

删除文本前后的所有空格和单词之间多余一个的空格，保留内部的单个空格

TRIM ( <Text> )

字符串：

空空空孙空兴空华空空空

如果只删除左、右、中 可以配合截取函数left,right,mid并配合len函数

## 09.重复字符串

2020年12月12日 20:10

**REPT ( <文本>, <重复次数> )**     按给定的次数重复文本

如果重复次数为 0，REPT 返回空白。

如果重复次数 不为整数，则截断该值。

REPT 函数的结果不能超过 32,767 个字符，否则 REPT 将返回错误

## 21. 替换函数

2020年12月12日 20:18

**REPLACE【将指定位置的字符串替换为新的字符串】**

**学计算机逻辑比英语更重要**

**0 1 2 3 4 5 6 7 8 9 10 11**

**SUBSTITUTE【按指定内容替换字符串】**

**学计算机逻辑比英语更重要**

## 01.REPLACE【按指定位置替换字符串】

2020年12月12日 20:18

### REPLACE【将指定位置的字符串替换为新的字符串】

**REPLACE ( <文本>, <起始位置>, <替换长度>, <替换内容> )**

参数	属性	描述
文本		包含要替换字符的文本字符串，或对文本列的引用
起始位置		要替换的内容在原文本中的起始位置
替换长度		要替换的字符数
替换内容		用于替换原文本中指定字符的替换文本

如果替换的长度为空，或引用包含空值的列，则替换内容字符串插入到起始位置，不替换任何字符。

REPLACE 从指定的位置执行替换，如果你需要替换指定的文本，请使用 SUBSTITUTE

字符串	按位置替换
跟着赵丽颖学VBA	跟着孙兴华学VBA
跟着赵丽颖学PowerBI	跟着孙兴华学PowerBI

**按位置替换 = REPLACE([字符串],3,3,"孙兴华")**

# 02.SUBSTITUTE【按指定内容替换字符串】

2020年12月12日 20:18

SUBSTITUTE ( <文本>, <替换哪些内容>, <新内容>, [<匹配项>] )

将指定的字符串替换为新的字符串，**区分大小写**

参数	属性	描述
文本		要在其中替换字符的文本或对文本的列的引用
替换内容		要替换的现有文本
新内容		替换后的新文本
匹配项	可选	替换内容的匹配项 。如果省略，则会替换所有命中的内容

按指定字符替换 = SUBSTITUTE([字符串],"孙兴华","小孙")

字符串	按指定字符替换
孙兴华昨天做了100个俯卧撑	小孙昨天做了100个俯卧撑
跟着孙兴华学VBA	跟着小孙学VBA

第四参数【可选】： 替换第几个， 省略就替换所有



## 22.三角、数学、信息函数【不常用】

2020年12月12日 20:33

# 01.三角函数

2020年12月12日 20:36

**COS ( <Number> )** 返回给定角度的余弦值

**COSH ( <Number> )** 返回给定数字的双曲余弦值

**COT ( <Number> )** 返回以弧度表示的角度的余切值

**COTH ( <Number> )** 返回双曲角度的双曲余切值

**SIN ( <Number> )** 返回给定角度的正弦值

**SINH ( <Number> )** 返回一个数字的双曲正弦值

**TAN ( <Number> )** 返回给定角度的正切值

**TANH ( <Number> )** 返回给定数字的双曲正切值

以上函数增加前缀 **A** 可以用于计算弧度 (相当于 arcsine,arccosine 等等) , 此处不再赘述

**DEGREES ( <Number> )** 将弧度转换为角度

**RADIANS ( <Number> )** 将角度转换为弧度

**SQRT ( <Number> )** 返回给定数字的平方根

**SQRTPI ( <Number> )** 返回(number \* pi)的平方根

## 02.数学函数

2020年12月12日 20:36

### ABS ( <Number> )

返回给定数字的绝对值，具有相同数据类型但没有符号的数值

### EXP ( <Number> )

返回自然常数 e 的给定数字次方

### FACT ( <Number> )

返回一个数字的阶乘，等于  $1*2*3*...*<Number>$

### LN ( <Number> )

返回一个数字的自然对数

### LOG ( <Number>, [<Base>] )

返回以指定数字为底的对数，省略 Base 以 10 为底数

### LOG10 ( <Number> )

返回以 10 为底的对数

### PI ( )

返回圆周率 3.14159265358979 的值，精确到 15 位

### POWER ( <Number>, <Power> )

返回提升到幂的数字的结果

例如：POWER(3,2) 就是3的2次方

### SIGN ( <Number> )

返回数字的符号：如果数字是正数，则返回 1；如果数字是零，则返回 0；如果数字是负数，则返回-1

### SQRT ( <Number> )

返回给定数字的平方根

### RAND ( )

返回一个大于等于 0 且小于 1 的随机数，均匀分布。随机数在重新计算时发生变化。

### RANDBETWEEN ( <Bottom>, <Top> )

返回指定数字之间的随机数

### EVEN ( <Number> )

**向上舍入到最近的偶数数字<你可以使用此函数来处理成对出现的项**

**例如：**

**EVEN(1.5) 返回2**

**EVEN(3) 返回4**

**EVEN(2) 返回2**

**EVEN(-1) 返回-2**

**ODD ( <Number> )**

**向上舍入到最近的奇数数字**

**例如：**

**ODD(1.5) 返回3**

**ODD(3) 返回3**

**ODD(2) 返回3**

**ODD(-1) 返回-1**

**ODD(-2) 返回-3**

**GCD ( <Number1>, <Number2> )**

**返回两个整数的最大公约数。最大公约数是除 1 和 2 而没有余数的最大整数。**

**LCM ( <Number1>, <Number2> )**

**返回整数的最小公倍数。最小公倍数是最小的正整数，它是两个整数参数 1 和 2 的倍数。使用 LCM 添加不同分母的分数**

**QUOTIENT ( <Numerator>, <Denominator> )**

**返回除法的整数部分**

**例如：**

**QUOTIENT ( 0, 2 ) 返回0**

**QUOTIENT ( 1, 2 ) 返回0**

**QUOTIENT ( 4, 2 ) 返回2**

**QUOTIENT ( 5, 2 ) 返回2**

**QUOTIENT ( 6, 2 ) 返回3**

**QUOTIENT ( 6, 3 ) 返回2**

**QUOTIENT ( 7, 3 ) 返回2**

### 03.信息函数

2020年12月12日 20:36

**ISEVEN(值)**

判定是否为偶数，并返回 TRUE 或 FALSE

**ISODD(值)**

判定是否为奇数，并返回 TRUE 或 FALSE

有了它们，MOD函数就可以退休了

**ISBLANK ( <Value> )**

检查值是否为空，并返回 TRUE 或 FALSE

**ISERROR ( <Value> )**

检查值是否为错误，并返回 TRUE 或 FALSE

**ISLOGICAL ( <Value> )**

检查值是否是逻辑值(TRUE 或 FALSE)，并返回 TRUE 或 FALSE

**ISNONTEXT ( <Value> )**

检查值是否为非文本(空白单元格是非文本)，并返回 TRUE 或 FALSE

**ISNUMBER ( <Value> )**

检查值是否为数字，并返回 TRUE 或 FALSE

**ISTEXT ( <Value> )**

检查值是否为文本，并返回 TRUE 或 FALSE

注意：当使用列 (而不是表达式) 作为参数时, 函数 ISNUMBER、ISTEXT 和 ISNONTEXT 始终返回 TRUE 或 FALSE, 具体结果取决于列的数据类型和每个单元格的空值类型

**ISEMPTY(<table\_expression>)**

检查表或表表达式是否为空

使用 COUNTROWS 统计表的行数也可以检查表是否为空，但这种做法的弊端在于当表不为空时，公式还计算出了它的行数，增加计算时间

## 04.取余数MOD

2020年12月12日 20:55

**MOD ( <被除数>, <除数> )**

**返回指定数字被整除后的余数**

**MOD ( 0, 2 ) 返回0**

**MOD ( 5, 2 ) 返回1**

**MOD ( 6, 3 ) 返回0**

**MOD ( 7, 3 ) 返回1**

**MOD ( 8, 3 ) 返回2**



## 23.分组与连接函数

2020年12月12日 21:22

# 01.SUMMARIZECOLUMNS

2020年12月12日 21:34

SUMMARIZECOLUMNS 函数是一种更灵活、更高效的 SUMMARIZE 实现方式。在编写查询的时候，你可以优先考虑 SUMMARIZECOLUMNS。

语法：

SUMMARIZECOLUMNS( <groupBy\_columnName>, [<groupBy\_columnName >] ..., [<filterTable>]..., [<name>, <expression>] ... )

位置	参数	描述
可重复第1参数	GroupBy_ColumnName	分组依据，可重复。可用于小计和总计函数
可选重复第2参数	FilterTable	可对原表进行筛选
可选第重复3参数	Name	新增加的列名
可选重复第4参数	Expression	新增加的列的内容表达式

## 一、返回不重复姓名

不重复姓名 = SummarizeColumns('重复姓名'[姓名])

姓名

丁智敏

李平平

王松

王刚

丁智敏

李平平

卢海军

王刚

张伊

→

姓名

丁智敏

李平平

王松

王刚

卢海军

张伊

对姓名列去重

## 二、返回多列不重复

多列不重复 = SummarizeColumns('多列重复'[年份], '多列重复'[姓名])

注：可以不是同一张表，但是必须有关系的多张表

年份

姓名

2019

丁智敏

2019

丁智敏

2019

王松

2019

王刚

2019

卢海军

2019

张伊

2020

丁智敏

2020

李平平

2020

王松

2020

王刚

2020

卢海军

2020

张伊

→

年份

姓名

2019

丁智敏

2020

丁智敏

2019

王松

2020

王松

2019

王刚

2020

王刚

2019

卢海军

2020

卢海军

2019

张伊

2020

张伊

2020

李平平

两列都重复才去重

## 三、返回汇总表【分组求和】使用了第1.3.4参数

汇总表 = SUMMARIZECOLUMNS('分组求和'[年份], '分组求和'[姓名], "总分", sum('分组求和'[成绩]))

年份

姓名

科目

成绩

2019

丁智敏

数学

89

2019

丁智敏

语文

73

2019

丁智敏

英语

63

2020

丁智敏

数学

85

2020

丁智敏

语文

83

2020

丁智敏

英语

72

→

年份

姓名

总分

2019

丁智敏

225

2020

丁智敏

240



2020	丁智敏	数学	85
2020	丁智敏	语文	83
2020	丁智敏	英语	72
2019	李平平	数学	60
2019	李平平	语文	64
2019	李平平	英语	77
2020	李平平	数学	77
2020	李平平	语文	100
2020	李平平	英语	75
2019	王松	数学	85
2019	王松	语文	73
2019	王松	英语	72
2020	王松	数学	61
2020	王松	语文	62
2020	王松	英语	62
2019	王刚	数学	76

年份	姓名	总分
2019	丁智敏	225
2020	丁智敏	240
2019	李平平	201
2020	李平平	252
2019	王松	230
2020	王松	185
2019	王刚	259
2020	王刚	222
2019	卢海军	249
2020	卢海军	227
2019	张伊	245
2020	张伊	285

#### 四、返回带筛选功能的汇总表【第二参数】

汇总表筛选 = SUMMARIZECOLUMNS('分组求和'[年份], '分组求和'[姓名], Filter('分组求和', '分组求和'[科目] = "数学"), "数学", sum('分组求和'[成绩]))

年份	姓名	科目	成绩
2019	丁智敏	数学	89
2019	丁智敏	语文	73
2019	丁智敏	英语	63
2020	丁智敏	数学	85
2020	丁智敏	语文	83
2020	丁智敏	英语	72
2019	李平平	数学	60
2019	李平平	语文	64
2019	李平平	英语	77
2020	李平平	数学	77
2020	李平平	语文	100
2020	李平平	英语	75
2019	王松	数学	85
2019	王松	语文	73
2019	王松	英语	72
2020	王松	数学	61
2020	王松	语文	62
2020	王松	英语	62
2019	王刚	数学	76

年份	姓名	数学
2019	丁智敏	89
2020	丁智敏	85
2019	李平平	60
2020	李平平	77
2019	王松	85
2020	王松	61
2019	王刚	76
2020	王刚	81
2019	卢海军	86
2020	卢海军	85
2019	张伊	71
2020	张伊	90

**注意：SUMMARIZECOLUMNS只适合新建表**

SUMMARIZECOLUMNS 不能在绝大部分度量值中使用，虽然在某些环境下可以得到正确结果，但你最好不要依赖这种方法。

当需要在度量值中执行分组和新建列时，最可靠的方式是SUMMARIZE+ADDCOLUMNS

为什么不支持度量值呢？

SUMMARIZECOLUMNS 不支持上下文转换时发生的计算，这个特性使它无法在大多数度量值中使用。

## 02.ADDMISSINGITEMS【就是一个开关】

2020年12月12日 22:01

**ADDMISSINGITEMS**([<展示列>,...],<汇总表>,[<分组列>,...], [筛选条件])

**展示列：**（可选，可重复）需要展示出来的列。

**汇总表：**经过筛选处理之后的表。

**分组列：**（可选，可重复）用来分组的列。

**例如：使用SUMMARIZECOLUMNS它不显示分数为空白的人员**

年份	姓名	总分
2019	丁智敏	225
2020	丁智敏	240
2019	李平平	201
2020	李平平	252
2019	王松	230
2020	王松	185
2019	王刚	259
2020	王刚	222
2019	卢海军	249
2020	卢海军	227
2019	张伊	245
2020	张伊	285

汇总表 = SUMMARIZECOLUMNS('分组求和'[年份],'分组求和'[姓名], "总分",sum('分组求和'[成绩]))

## ADDMISSINGITEMS+SUMMARIZECOLUMNS组合就可以显示了 这个组合与单独用SUMMARIZE出来的效果是一样的

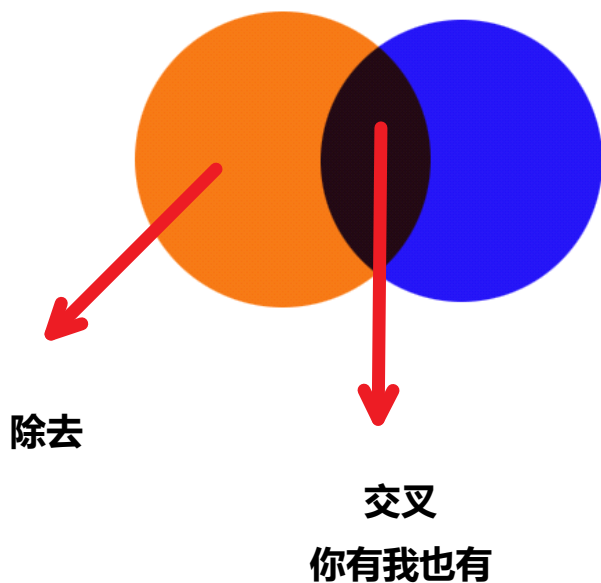
年份	姓名	总分
2019	丁智敏	225
2020	丁智敏	240
2019	李平平	201
2020	李平平	252
2019	王松	230
2020	王松	185
2019	王刚	259
2020	王刚	222
2019	卢海军	249
2020	卢海军	227
2019	张伊	245
2020	张伊	285
2019	孙兴华	
2020	孙兴华	

汇总表2 = ADDMISSINGITEMS('分组求和'[年份],'分组求和'[姓名],  
SUMMARIZECOLUMNS('分组求和'[年份],'分组求和'[姓名], "总分",sum('分组求和'[成绩])),  
'分组求和'[年份],'分组求和'[姓名])

**ADDMISSINGITEMS** 是软件在自动生成查询时经常使用的函数，某些图表会在运行时调用这个函数。函数的作用是添加由于新列的表达式返回空值而被 SUMMARIZECOLUMNS 隐藏的行。

### 03.交叉INTERSECT，除去EXCEPT，全部UNION

2020年12月12日 22:59



**交叉 = INTERSECT('刺杀名单','死亡名单')**

谁在我的刺杀名单上，并且已经死了

**除去 = EXCEPT('刺杀名单','死亡名单')**

谁在我的刺杀名单上还没死？

如果想知道死的人谁不是我名单中的，把表调换一下位置

**全部 = UNION('刺杀名单','死亡名单')**

两张表的全部数据，肯定有重复的内容

## 04.笛卡尔积CROSSJOIN

2020年12月12日 23:03

**笛卡尔积** = CROSSJOIN('表1', '表2')

年份	姓名
2018	孙兴华
2019	孙兴华
2020	孙兴华
2018	赵丽颖
2019	赵丽颖
2020	赵丽颖

## 24.查找匹配函数

2020年12月12日 21:17

## 01.CONTAINS 【多条件查找】

2020年12月12日 21:05

**CONTAINS ( <Table>, <ColumnName>, <Value>, [ <ColumnName>, <Value>, [ ... ] ] )**

如果指定的<table>至少存在一行满足所有<ColumnName>都有对应的<Value>, 返回 true, 否则 CONTAINS 返回 false。

参数	属性	描述
Table		物理表或返回表的表达式
ColumnName	可重复	使用标准 DAX 语法的现有列的名称, 不支持表达式
Value	可重复	任何返回单个标量值的 DAX 表达式, 该值将在 columnName 中查找

可以有第四参数和第五参数, 类似第二和第三参数必须成对出现, 以此类推

例如: 度量值【为什么不能新建列, 注意上下文】

是否购买过 = contains('表', '表'[日期], Date(2020, 12, 2), '表'[姓名], "李四")

李四在2020/12/2是否购买过

**特别注意**

ColumnName 和 Value 必须成对使用, 否则将报错

ColumnName 必须属于第一参数或第一参数的扩展表

当 ColumnName 属于扩展表时, 必须使用完全限定的名称 (表[列名])

CONTAINS 为简单筛选提供了更好的性能, 而 CALCULATETABLE 适用于复杂的筛选表达式。

CONTAINS 只检查精确匹配, 如果需要更复杂的筛选条件, 则必须使用 CALCULATETABLE、FILTER 或混合使用。不过, CONTAINS 函数允许你组合多列作为条件

## 02.TREATAS 函数【无关系情况下查找匹配】

2020年12月12日 21:19

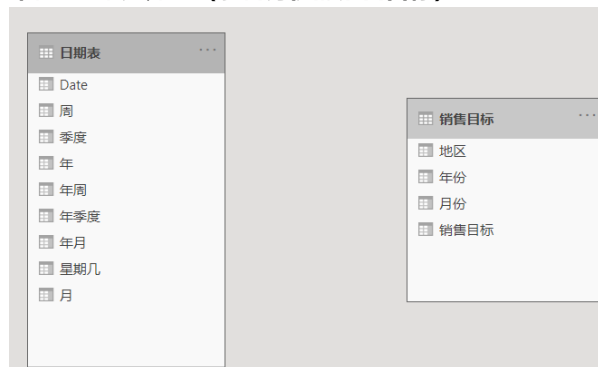
TREATAS 函数是无关系情况下执行查找匹配的最佳选择，它通过映射**数据沿袭**的方式使无关系筛选成为可能，了解 TREATAS，你需要先了解关于数据沿袭的知识。

TREATAS(table\_expression, <column>[, <column>[, <column>[,...]]])

作用是：把什么当作什么

解释：就是把一参当作二参的筛选器，通过一参筛选二参。

### 案例1：单列筛选（求各月份的销售目标）



销售目标中没有详细日期，无法与日期表建立直接的关系

我们希望通过日期表中的【月】筛选销售目标中的【月份】，所以我们将日期表的【月】用作筛选条件（一参），而销售目标的【月份】则是被一参筛选的列（二参）

年月	月度销售目标
202001	1294
202002	1309
202003	1118
202004	937
202005	1392
202006	1485
202007	1381
202008	1414
202009	1071
202010	1428
202011	1199
202012	1046
202101	1794

月度销售目标 =  
CALCULATE (  
SUM ( '销售目标'[销售目标] ),  
TREATAS ( VALUES ( '日期表'[月] ), '销售目标'[月份] )  
)

把一参当做二参的筛选器，通过一参筛选二参

### 案例2：多列筛选（求各年份月份的销售目标）

年	1	2	3	4	5	6	7	8	9	10	11	12	总计
2020	581	709	441	457	697	733	835	673	452	676	604	544	7402
2021	713	600	677	480	695	752	546	741	619	752	595	502	7672
总计	1294	1309	1118	937	1392	1485	1381	1414	1071	1428	1199	1046	15074

年月销售目标1 =

CALCULATE (  
SUM ( '销售目标'[销售目标] ),  
TREATAS ( VALUES ( '日期表'[年] ), '销售目标'[年份] ),  
TREATAS ( VALUES ( '日期表'[月] ), '销售目标'[月份] )  
)



多条件筛选时，如果考虑速度问题：可以使用Summarize进行优化

年月销售目标 2 =

CALCULATE (  
SUM ( '销售目标'[销售目标] ),  
TREATAS ( SUMMARIZE ( '日期表', '日期表'[年], '日期表'[月] ), '销售目标'[年份], '销售目标'[月份] )  
)

这里要么用SUMMARIZE，要么用ADDMISSINGITEMS+SUMMARIZECOLUMNS组合，个人建议还是用SUMMARIZE

## 附.日期表

2020年12月11日 20:09

```
日期表 = ADDCOLUMNS(  
    CALENDAR(date(2020,1,1),date(2021,12,31)),  
    "年", YEAR ( [Date] ),  
    "季度", ROUNDUP(MONTH([Date])/3,0),  
    "月", MONTH([Date]),  
    "周", weeknum([Date]),  
    "年季度", year([date]) & "Q" &  
    ROUNDUP(MONTH([Date])/3,0),  
    "年月", year([Date]) * 100 + MONTH([Date]),  
    "年周", year([Date]) * 100 + weeknum([Date]),  
    "星期几", WEEKDAY([Date])  
)
```



### 03.Row函数【返回一个单行表，其中包含由 DAX 表达式指定的新列】

2020年12月14日 11:36

**ROW ( <Name>, <Expression>, [ <Name>, <Expression> ], [ ... ] )**

参数	属性	描述
Name	可重复	新列的名称
Expression	可重复	新列使用的表达式

说人话：**就是创建一张表**

**Row("新列名",创建这个新列用什么表达式)**

案例：

查询表 =

```
row(  
    "总销量",SUM('Row'[销售量]),  
    "总金额",SUM('Row'[销售金额])  
)
```

## 04.综合案例分析：利用Treatas函数自由切换坐标轴

2020年12月14日 9:51

### (1) 利用笛卡尔积做表

坐标轴	城市
城市	北京
城市	上海
城市	广州
产品	A
产品	B
产品	C
年份	2015
年份	2016
年份	2017
年份	2018
年份	2019
年份	2020



类别	坐标轴
北京	城市
上海	城市
广州	城市
A	产品
B	产品
C	产品
2015	年份
2016	年份
2017	年份
2018	年份
2019	年份
2020	年份

可以手工改一下列名，改成类别更好理解

坐标轴 =

```
UNION(  
    UNION(  
        CROSSJOIN(ROW("坐标轴","城市"),VALUES('案例'[城市])),  
        CROSSJOIN(ROW("坐标轴","产品"),VALUES('案例'[产品]))  
    ),  
    CROSSJOIN(ROW("坐标轴","年份"),VALUES('案例'[年份]))  
)
```

为什么我用了两次UNION？因为他一次只能连接两张表，我先把前两张表变成一张表，再去连接第3张表

### (2) 第二个切片器

切换 = UNION(ROW("结果","销量"),ROW("结果","销售金额"))

结果
销量
销售金额

### (3) 把两个总度量值写出来

总金额 = SUM('案例'[销售金额])

总销量 = SUM('案例'[销量])

### (4) 分别写显示销量与显示销售金额的度量值

显示销量 =

```
SWITCH(  
    SELECTEDVALUE('坐标轴'[坐标轴]),  
    "城市",CALCULATE([总销量],TREATAS(VALUES('坐标轴'[类别]),'案例'[城市])),  
    "产品",CALCULATE([总销量],TREATAS(VALUES('坐标轴'[类别]),'案例'[产品])),  
    "年份",CALCULATE([总销量],TREATAS(VALUES('坐标轴'[类别]),'案例'[年份]))  
)
```

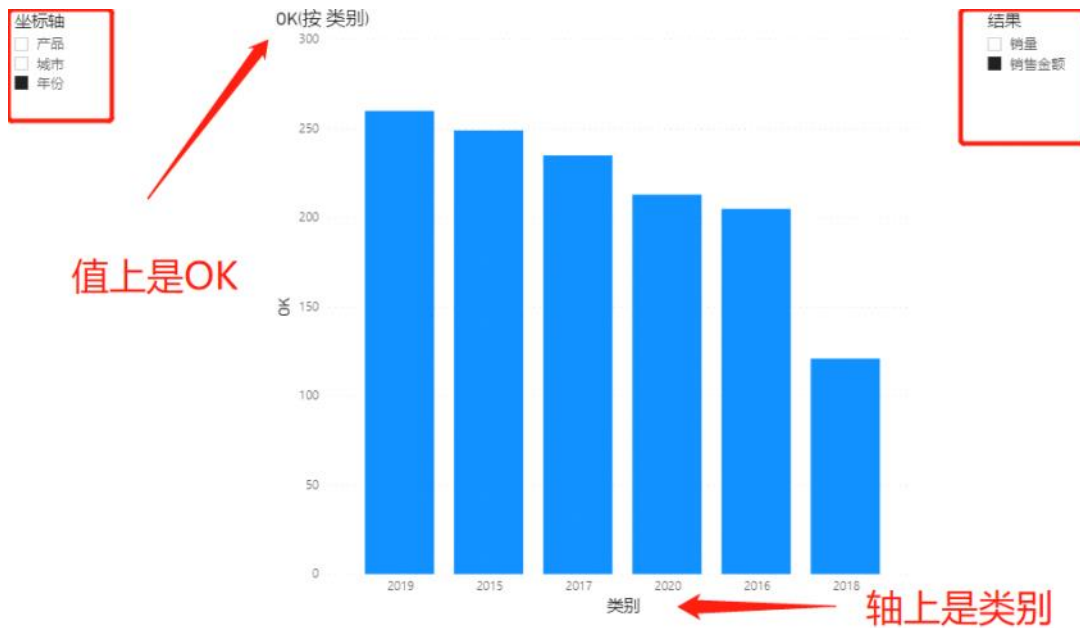
当切片器单选的时候，如果选“城市”，显示筛选后的销量

显示金额 =

```
SWITCH(  
    SELECTEDVALUE('坐标轴'[坐标轴]),  
    "城市",CALCULATE([总金额],TREATAS(VALUES('坐标轴'[类别]),'案例'[城市])),  
    "产品",CALCULATE([总金额],TREATAS(VALUES('坐标轴'[类别]),'案例'[产品])),  
    "年份",CALCULATE([总金额],TREATAS(VALUES('坐标轴'[类别]),'案例'[年份]))  
)
```

### (5) 最后把结果做出来

OK = SWITCH(SELECTEDVALUE('切换'[结果]),"销售金额",[显示金额],"销量",[显示销量])



## 05.VAR类似于VBA里面的Dim【这不是查找匹配函数】

2020年12月14日 10:46

VBA里面

**Dim** 变量名 **As** 数据类型

在PBI里，VAR就是声明变量的。当然它可以代替当前行EARLIER函数，但是我不推荐大家用，我在2020.6.27录制的视频第10集起名大师VAR中介绍过，就不在重复了。



<https://www.bilibili.com/video/BV1tf4y117Sh>

接上节课案例：

显示金额和显示销售我们做了两个度量值，又做了一个OK的度量值  
用VAR一步搞定

孙兴华 =

**VAR** xiaoliang =

```
SWITCH(  
    SELECTEDVALUE('坐标轴'[坐标轴]),  
    "城市",CALCULATE([总销量],TREATAS(VALUE('坐标轴'[类别]),'案例'[城市])),  
    "产品",CALCULATE([总销量],TREATAS(VALUE('坐标轴'[类别]),'案例'[产品])),  
    "年份",CALCULATE([总销量],TREATAS(VALUE('坐标轴'[类别]),'案例'[年份]))  
)
```

**VAR** money=

```
SWITCH(  
    SELECTEDVALUE('坐标轴'[坐标轴]),  
    "城市",CALCULATE([总金额],TREATAS(VALUE('坐标轴'[类别]),'案例'[城市])),  
    "产品",CALCULATE([总金额],TREATAS(VALUE('坐标轴'[类别]),'案例'[产品])),  
    "年份",CALCULATE([总金额],TREATAS(VALUE('坐标轴'[类别]),'案例'[年份]))  
)
```

**return**

```
SWITCH(SELECTEDVALUE('切换'[结果]),"销售金额",money,"销量",xiaoliang)
```

除了可以定义度量值，还可以定义新建列：

```
评价 =  
VAR zongfen = '案例2'[数学] + '案例2'[语文] + '案例2'[英语]  
return  
if (zongfen >= 270, "优秀", "一般")
```

如果没有这个VAR，你要先做一次总分列，再做一个评价列。

## 【实战案例】模糊查找

2020年12月27日 9:00

类型	电影	销售颜色
<input type="checkbox"/> 爱情	A	喜剧/科幻
<input type="checkbox"/> 传记	B	传记/历史
<input type="checkbox"/> 惊悚	C	战争/爱情
<input type="checkbox"/> 科幻	D	爱情/科幻
<input type="checkbox"/> 恐怖	E	惊悚/恐怖
<input type="checkbox"/> 历史	F	喜剧/恐怖
<input type="checkbox"/> 喜剧	G	爱情/历史
<input type="checkbox"/> 战争	H	喜剧/恐怖
	总计	战争/爱情

查询类型 =

VAR a =

VALUES ( '查询表'[类型] )

VAR b =

MAX ( '销售表'[类型] )

RETURN

IF (

COUNTROWS ( FILTER ( a, SEARCH ( [类型], b,, 0 ) > 0 ) ) > 0,

b,

BLANK ()

)

写Find也可以

MAX ( '销售表'[类型] )

这里的作用单纯的是为了将[类型]列进行聚合，否则无法在查询函数中使用

通过IF语句进行判断，符合条件显示，否则显示为空，内部利用FILTER函数进行上下文传递，将原本没有联系的维度和事实表中的颜色查询链接在一起。

FIND函数区分大小写，不支持通配符；

SEARCH函数不区分大小写，支持通配符

## 06.检查字符串是否被包含

2020年12月14日 11:12

FIND 和 SEARCH 函数可以查找指定字符串所在的位置，如果你只需要检查字符串是否被包含，可以使用 CONTAINSSTRING 和 CONTAINSSTRINGEXACT，它们只进行逻辑判断，计算效率更高。

**(1) CONTAINSSTRING 支持通配符，不区分大小写**

CONTAINSSTRING ( <WithinText>, <FindText> )

**列 = CONTAINSSTRING("跟着孙兴华学VBA","孙?华")**

如果一个文本字符串包含另一个文本字符串，则返回 TRUE。

CONTAINSSTRING **支持通配符，不区分大小写**，可以执行模糊匹配，使用时注意它的参数顺序。

问号(?): 问号匹配任何单个字符

星号(\*): 星号匹配任何字符序列

如果你想找到的是问号或星号本身，请在字符前键入一个波浪号(~)

**(2) CONTAINSSTRINGEXACT 不支持通配符，区分大小写**

CONTAINSSTRINGEXACT ( <WithinText>, <FindText> )

**列 = CONTAINSSTRINGEXACT("跟着孙兴华学VBA","vba")**

如果一个文本字符串包含另一个文本字符串，则返回 TRUE。

CONTAINSSTRINGEXACT **不支持通配符，区分大小写**。使用时注意它的参数顺序。

## 25.排名函数

2020年12月14日 11:51



## 01.RANKX排名详细讲解

2020年12月14日 11:59

**RANKX(<table>, <expression>, [ <value> ], [ <order> ], [ <ties> ])**

**RANKX** 为的每一行计值表达式并得到一个值列表，在当前筛选上下文中计值，将得到的结果与列表中的值进行比较，根据排名规则和的设置，返回最终排名。

参数	属性	描述
Table		表或返回表的表达式 <b>第一参 表：可以是直接的表，也可以是用函数生成的表。</b>
Expression		沿着 Table 每行计值的表达式 <b>第二参 表达式：聚合表达式，或者写好的度量值。</b>
Value	可选	需要返回排名的 DAX 表达式，返回标量值。当<value>省略时，用<expression>代替 <b>第三参 值：可选。可以是聚合表达式，也可以是一个直接的数值。</b>
Order	可选	排名依据。0 或 False 代表降序；1 或 True 代表升序，默认使用降序 <b>第四参 排序：可以是0或1，也可以是ASC或DESC，升降序使用。</b>
Ties	可选	处理相同排名时的依据，skip 代表稀疏排名，下一名的排序等于之前所有排序的数量+1；dense 代表稠密排名，只累加排序，不考虑数量。默认使用 skip <b>第五参 排序方法：Skip国际排序；Dense中国式排序。</b>

**案例：**

**总金额 = sum('销售表'[销售金额])**

**排名 = RANKX ( ALL ( '销售表' ), [总金额])**

商品编码	总金额	排名
A001	463	2
A002	377	3
A003	358	4
A004	232	5
A005	225	6
A006	587	1
A007		7
A008		7
总计	2242	1

**有两个问题：**

- 1.A007 A008不应该显示出来**
- 2.排名总计怪怪的**

**排名优化1 = IF(HASONEVALUE('销售表'[商品编码]),RANKX(ALL('销售表'),[总金额]))**

商品编码	总金额	排名	排名优化1
A001	463	2	2
A002	377	3	3
A003	358	4	4
A004	232	5	5
A005	225	6	6
A006	587	1	1
A007		7	7
A008		7	7
总计	2242	1	

想让总计不显示的通用方法，就是  
if+HASONEVALUE

详见16.02

去掉A007和A008

排名优化2 = IF(AND( HASONEVALUE ('销售表'[商品编码] ), [总金额] >0),RANKX (FILTER ( ALL ('销售表'), [总金额] >0), [总金额],,1))

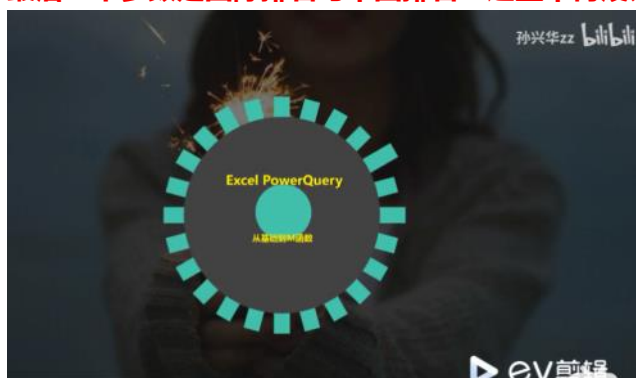
AND(True,True) 才返回True  
OR(True,False) 就返回True

ALL函数是绝对排名的用法，如果是相对排名，可以自行替换成ALLSELECTED。

如果第三参我们不输入的情况下，第二参也是第三参，一旦输入第三参，结果展示的是第三参按照第二参排名的位置。

假如我有一份普通中学学生成绩的分数，和一份重点中学成绩排名，我想看看普通中学的学生在重点校能排第几，这不就是实际第三参的用途么

最后一个参数是国际排名与中国排名？这里不再废话，我在PowerQuery里讲过了



S01E11 排名

<https://www.bilibili.com/video/BV1oa4y1j75e>

## [进阶].RANKX进行明细级别的排名

2020年12月14日 19:44

### 新建列

单品销售排名 = RANKX(FILTER('案例2','案例2'[品种]=EARLIER('案例2'[品种])), '案例2'[销售金额])

02.TOPN 【返回满足条件的前N行记录】

2020年12月14日 19:53

RANKX 适合计算明细的名次数据，那么 TOPN 则可以批量返回结果，从一张表中返回所有满足条件的前 N 行记录。

它的特别之处是返回的不是值，是前N行的表



	属性	描述
第1参		需要返回的行数
第2参		用来返回行记录的表表达式
第3参	可选 可重复	用来排序的表达式
第4参	可选 可重复	排序方式. 0/FALSE/DESC – 降序; 1/TRUE/ASC – 升序.

使用行上下文【新建列】

前2名的销售金额之和 = sumx(TOPN(2,'TOPN','TOPN'[销售金额],0),'TOPN'[销售金额])

1 前2名的销售金额之和 = sumx(TOPN(2,'TOPN','TOPN'[销售金额],0),'TOPN'[销售金额])			
日期	店号	销售金额	前2名的销售金额之和
2020年7月1日	1	40	275
2020年7月1日	2	39	275
2020年7月1日	3	85	275
2020年7月1日	4	53	275
2020年7月1日	5	63	275
2020年7月2日	1	17	275
2020年7月2日	2	91	275
2020年7月2日	3	44	275
2020年7月2日	4	12	275
2020年7月2日	5	65	275
2020年7月3日	1	91	275
2020年7月3日	2	71	275
2020年7月3日	3	54	275
2020年7月3日	4	55	275
2020年7月3日	5	16	275

TOPN返回的是表，本身就是表函数【新建表】

查询销售前2名的记录 = TOPN(2,'TOPN','TOPN'[销售金额],0)

1 表 = TOPN(2,'TOPN','TOPN'[销售金额],0)			
日期	店号	销售金额	
2020/7/3 0:00:00	1	91	
2020/7/2 0:00:00	2	91	
2020/7/4 0:00:00	5	93	

可以进入正题了：

【度量值】销售额 = SUM('TOPN'[销售金额])

【新建表测试】测试表 = TOPN(3,all('TOPN'),[销售额],0)

【度量值】销售量前3名共卖了多少 = CALCULATE([销售额],TOPN(3,all('TOPN'),[销售额],0) )

【度量值】销售量前3名的店铺共卖了多少 = CALCULATE([销售额],TOPN(3,all('TOPN'[店号]),[销售额],0) )

什么叫销售量前3名的店铺共卖了多少？

相当于进行了一次分组求和Sumif，找出销售最高的3家店铺，然后把金额加在一起。

前3名的店铺销售占比= **Divide(销售额前3名的店铺共卖了多少,calculate([销售额],all('TOPN'[店号]))**

### 03.CONCATENATEX【将多个值连接到一起，以文本的形式输出】

2020年12月14日 20:41

参数	属性	描述
Table		用于表达式每行计值的表 <b>第1参数</b>
Expression		用于逐行计值的表达式 <b>第2参数</b>
Delimiter	可选	连接表达式结果的连接符 <b>第3参数</b>
OrderBy_Expression	可选 可重复	排序使用的表达式 <b>第4参数</b>
Order	可选 可重复	排序逻辑. 0/FALSE/DESC – 降序; 1/TRUE/ASC – 升序 (默认) <b>第5参数</b>

#### 案例1:

度量值 = CONCATENATEX(VALUES('CONCATENATEX'[人物]), 'CONCATENATEX'[人物], ",")

小说 度量值

射雕	杨过,郭靖
神雕	杨过,小龙女
倚天	张无忌,周芷若,赵敏,小昭
总计	杨过,小龙女,郭靖,张无忌,周芷若,赵敏,小昭

- 第一参数：对哪张表进行排名
- 第二参数：显示来自哪里
- 第三参数：连续符号
- 第四参数：排序
- 第五参数：升序还是降序

#### 案例2：对结果进行排序

度量值 = CONCATENATEX ( RELATEDTABLE ( '流水' ), FORMAT ( '流水'[购物日期], "yyyy/mm/dd" ),  
",", '流水'[购物日期])

姓名 度量值

李四	2020/12/03, 2020/12/09
王五	2020/12/05, 2020/12/08
张三	2020/12/01, 2020/12/11
总计	2020/12/01, 2020/12/03, 2020/12/05, 2020/12/08, 2020/12/09, 2020/12/11



## 04.【案例】CONCATENATEX综合案例：

2020年12月27日 7:14

买家昵称	购买物品1	购买数量	消费金额	客户等级
阿佛罗迪	红玫瑰、白玫瑰	2	9,990.00	金牌客户
阿鲁迪巴	牛角	1	1,000.00	银牌客户
艾奥里亚	羊	1	5,998.00	金牌客户
艾俄洛斯	弓、箭	2	1,499.00	银牌客户
迪斯马斯克	黄酒、姜、蒜、醋	4	37.74	铜牌客户
卡妙	花瓶	1	299.00	铜牌客户
米罗	针	1	50.00	铜牌客户
穆	土豆、红薯	2	69.30	铜牌客户
撒加	教皇头盔	1	2,500.00	银牌客户
沙加	袈裟	1	199.00	铜牌客户
童虎	枪、三节棍、盾、双节棍、拐	6	2,986.00	银牌客户
修罗	剑	1	599.00	铜牌客户
总计				

第00课就讲了如何让总计消费

买家昵称	购买物品
阿佛罗迪	红玫瑰、白玫瑰
阿鲁迪巴	牛角
艾奥里亚	羊
艾俄洛斯	弓、箭
迪斯马斯克	黄酒、姜、蒜、醋
卡妙	花瓶
米罗	针
穆	土豆、红薯
撒加	教皇头盔
沙加	袈裟
童虎	枪、三节棍、盾、双节棍、拐
修罗	剑
总计	土豆、红薯、牛角、教皇头盔、黄酒、姜、蒜、醋、羊、袈裟、枪、三节棍、盾、双节棍、拐、箭、剑、花瓶、红玫瑰、白玫瑰

第一步：笔记25.03 利用CONCATENATEX函数将多个值连接在一起

购买物品 = CONCATENATEX (VALUES('补充案例'[商品]),'补充案例'[商品]、")

第二步：让总计栏显示为空的通用方法 笔记16.02 if+HASONEVALUE

购买物品1 = IF(HASONEVALUE('补充案例'[买家昵称]),CONCATENATEX (VALUES ('补充案例'[商品]),'补充案例'[商品]、"))

买家昵称	购买物品1
阿佛罗迪	红玫瑰、白玫瑰
阿鲁迪巴	牛角
艾奥里亚	羊
艾俄洛斯	弓、箭
迪斯马斯克	黄酒、姜、蒜、醋
卡妙	花瓶
米罗	针
穆	土豆、红薯
撒加	教皇头盔
沙加	袈裟
童虎	枪、三节棍、盾、双节棍、拐
修罗	剑
总计	

### 第三步：计算购买数量和金额

购买数量 = sum('补充案例'[销售数量])

消费金额 =SUMX ('补充案例','补充案例'[单价] \*'补充案例'[销售数量])

### 第四步：计算客户等级

客户等级 = IF( HASONEVALUE ('补充案例'[买家昵称]),

SWITCH(

TRUE(),

[消费金额] >=3000,"金牌客户",

[消费金额] >=1000,"银牌客户",

"铜牌客户"

))



## 26.人工造表最终方案

2020年12月14日 21:14

**ADDCOLUMNS** 从指定的表开始添加列

**SELECTCOLUMNS** 从空表开始添加列

其实，ADDCOLUMNS可以代替SELECTCOLUMNS。

但是，需要配合DISTINCT和Values使用，相对麻烦

2019年10月以前的版本可能不支持SELECTCOLUMNS

解决方案是更新版本，而不是在旧函数上绕自己

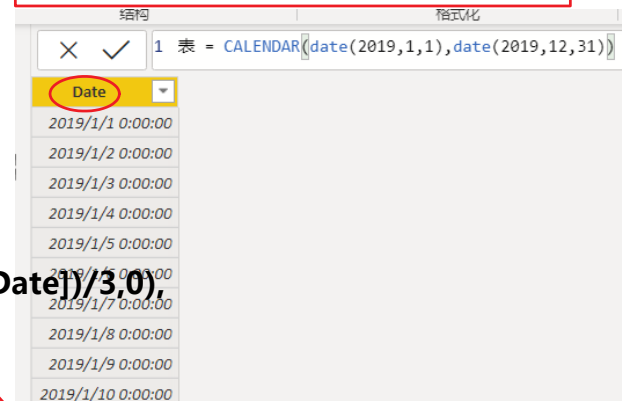
# 01.ADDCOLUMNS

2020年12月14日 21:05

日期表 = ADDCOLUMNS(  
CALENDAR(date(2019,1,1),date(2019,12,31)),  
"年", YEAR ([Date] ),  
"季度", ROUNDUP(MONTH([Date])/3,0),  
"月", MONTH([Date]),  
"周", weeknum([Date]),  
"年季度", year([date]) & "Q" & ROUNDUP(MONTH([Date])/3,0),  
"年月", year([Date]) \* 100 + MONTH([Date]),  
"年周", year([Date]) \* 100 + weeknum([Date]),  
"星期几", WEEKDAY([Date])  
)

表函数

CALENDAR(<开始日期>,<结束日期>)



The screenshot shows the DAX editor with the formula: 1 表 = CALENDAR(date(2019,1,1),date(2019,12,31)). The output table has a column named 'Date' with the following values:

Date
2019/1/1 0:00:00
2019/1/2 0:00:00
2019/1/3 0:00:00
2019/1/4 0:00:00
2019/1/5 0:00:00
2019/1/6 0:00:00
2019/1/7 0:00:00
2019/1/8 0:00:00
2019/1/9 0:00:00
2019/1/10 0:00:00

为什么要乘以100呢?

year([Date]) 提取出来的年假设是2019

month([Date]) 提取出来的月假设是10

year([Date]) & month([Date]) 结果是201910

如果:

month([Date]) 提取出来的月假设是9

year([Date]) & month([Date]) 结果是20199

可是:

year([Date]) \* 100 相当时2019\*100=201900

用201900+1=201901 用 201900+10=201910

RoundUp(1.88) 返回: 2

RoundUp(1.81,1) 返回: 1.9

数学题:

1/3=0.3333 向上舍入保留0位小数点 就是1

2/3=0.6667 向上舍入保留0位小数点 就是1

4/3 = 1.3333 向上舍入保留0位小数点 就是2

## 附：案例

2020年12月14日 21:40

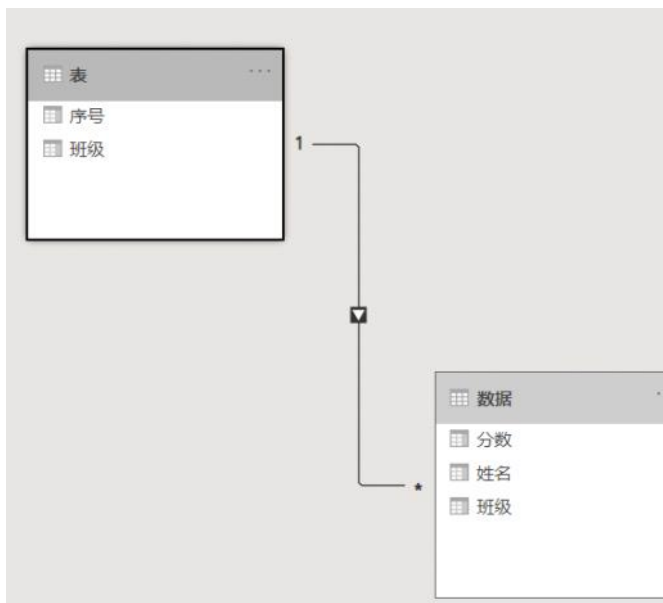


表 2 = ADDCOLUMNS('表',  
"总分",CALCULATE(SUM('数据'[分数])),  
"平均分",CALCULATE(AVERAGE('数据'[分数]))  
)

序号	班级	总分	平均分
1	1班	101	50.5
2	2班	92	46
3	3班	83	41.5

表 2 = FILTER(ADDCOLUMNS('表',  
"总分",CALCULATE(SUM('数据'[分数])),  
"平均分",CALCULATE(AVERAGE('数据'[分数]))),[平均分]<50)

序号	班级	总分	平均分
2	2班	92	46
3	3班	83	41.5

## 02.SELECTCOLUMNS

2020年12月14日 21:39

SELECTCOLUMNS ( <Table>, <Name>, <Expression>, [ <Name>, <Expression> ], [ ... ] )

属性	描述
	从中选择列的表 <b>第1参数</b>
可重复	要添加的新列的名称 <b>第2参数</b>
可重复	要添加的新列的表达式 <b>第3参数</b>

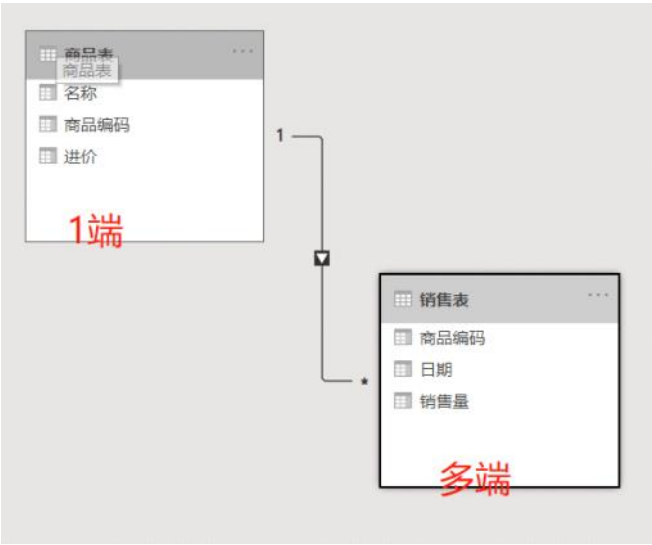


表 =  
SELECTCOLUMNS('销售表',  
"销售日期",'销售表'[日期],  
"商品名称",RELATED('商品表'[名称]),  
"销售数量",'销售表'[销售量],  
"销售金额",RELATED('商品表'[进价])\*'销售表'[销售量]  
)

销售日期	商品名称	销售数量	销售金额
2020/12/1 0:00:00	土豆	10	20
2020/12/1 0:00:00	茄子	20	60
2020/12/1 0:00:00	黄瓜	30	120
2020/12/2 0:00:00	土豆	40	80
2020/12/2 0:00:00	茄子	50	150
2020/12/2 0:00:00	黄瓜	60	240
2020/12/3 0:00:00	土豆	70	140
2020/12/3 0:00:00	茄子	80	240
2020/12/3 0:00:00	黄瓜	90	360

# 27.时间智能函数

2020年12月15日 9:35

## 【特别提示】日期表标记作用

2020年12月15日 9:44

如果日期表与其它表是通过日期列连线的，那么时间智能函数直接可以生效

如果不是通过日期列连线，而是通过索引列，或是 序号列进行连线，时间智能函数不生效



但是正常情况下，我们都是用日期列去连线的

另外。新版PowerBI可以支持对数据表进行操作，也就是说你可以不用再新建日期表

**切记：你的日期列是唯一值，否则使用Values创建一个唯一值的表或者使用去重方法均可**  
**如果你都不愿意这样做，就用日期表连线吧**

## 附：日期表

2020年7月1日 9:46

```
日期表 = ADDCOLUMNS(  
    CALENDAR(date(2019,1,1),date(2019,12,31)),  
    "年", YEAR ( [Date] ),  
    "季度", ROUNDUP(MONTH([Date])/3,0),  
    "月", MONTH([Date]),  
    "周", weeknum([Date]),  
    "年季度", year([date]) & "Q" & ROUNDUP(MONTH([Date])/3,0),  
    "年月", year([Date]) * 100 + MONTH([Date]),  
    "年周", year([Date]) * 100 + weeknum([Date]),  
    "星期几", WEEKDAY([Date])  
)
```

# 01. 计算累加值

2020年7月1日 9:46

计算累计有两类常用的时间智能函数：

以 DATESYTD 为代表的返回日期值的表函数

和

以 TOTALYTD 为代表的返回标量值的函数

计算本年的年初至今(YTD)、季初至今(QTD)和月初至今(MTD)



# 01.DATESYTD【已淘汰】

2020年12月15日 9:51

## DATESYTD ( <Dates>, [<YearEndDate>] )

属性	描述
	日期格式的列或返回单列的表达式，通常使用日期表的日期列 <b>第1参数</b>
可选	年截止日，日期字符串，忽略年 <b>第2参数</b>

### 通过【新建表】观察

×	✓	1 年初至今 = DATESYTD('表'[日期])
日期	▼	
2021/1/1 0:00:00		
2021/3/2 0:00:00		
2021/4/1 0:00:00		
2021/6/3 0:00:00		
2021/6/30 0:00:00		

×	✓	1 季初至今 = DATESQTD('表'[日期])
日期	▼	
2021/4/1 0:00:00		
2021/6/3 0:00:00		
2021/6/30 0:00:00		

×	✓	1 月初至今 = DATESMTD('表'[日期])
日期	▼	
2021/6/3 0:00:00		
2021/6/30 0:00:00		

年初至今 = DATESYTD('表'[日期])

季初至今 = DATESQTD('表'[日期])

月初至今 = DATESMTD('表'[日期])

总结：它返回的是一张表

- (1) YTD(年初至今)：是根据你的日期数据决定的。它相当于把日期列排序，然后找到最大的年，再求年求至今。比如今天是2020/12/25。如果你的数据中最大的年是2021年，那它就计算2021年，你的数据中最大的年是2030年，同样也是计算2030年。
- (2) QTD(季初至今)：先找到最大年，再找最大月，确定最大月所在的季度后，将这个时间表拿出来。
- (3) MTD(月初至今)：先找到最大年，再找最大月，将这个时间表拿出来。

案例：计算年初至今的销售量

总销售 = sum('表'[销售])

年初至今 = CALCULATE([总销售], DATESYTD('表'[日期]))

但是有一个更简单的函数，可以代替 Calculate+DATESYTD这样的组合

## 02.TOTALYTD 【推荐】

2020年12月15日 10:49

总销售 = sum('表'[销售])

年初至今 = TOTALYTD([总销量],'表'[日期])

**TOTALYTD ( <表达式>, <日期列>, [<筛选器>], [截止日期] )**

参数	属性	描述
表达式		返回标量值的表达式
日期列		包含日期的列
筛选器	可选	应用于当前上下文的筛选器参数，可以是布尔表达式或表表达式
截止日期	可选	带有日期的文本字符串，用于定义年末日期，默认值为 12 月 31 日

**计算本年的年初至今(YTD)、季初至今(QTD)和月初至今(MTD)**

总销售 = sum('表'[销售])

年初至今 = TOTALYTD([总销售],'表'[日期])

年初至今1 = TOTALYTD([总销售],'表'[日期],FILTER('表','表'[销售]<50))

年初至今1 = TOTALYTD([总销售],'表'[日期],FILTER('表','表'[销售]<50),"03/31")

因为6月30日是财务年度的结束日期，每年的7月1日是财务年的开始日

TOTALYTD ([总销售], '表'[日期], "06-30" )

CALCULATE ([总销售], DATESYTD('表'[日期], "06-30"))

**注： 闰年02-29 平年写02-28**

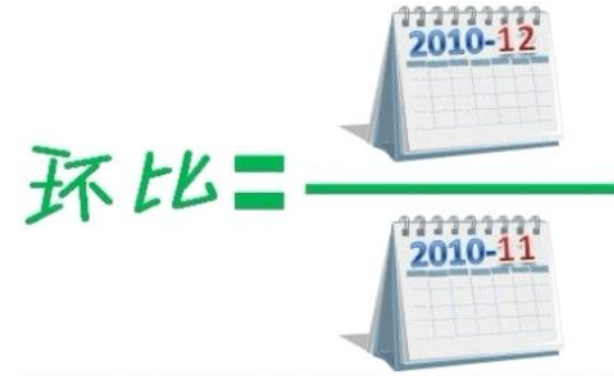
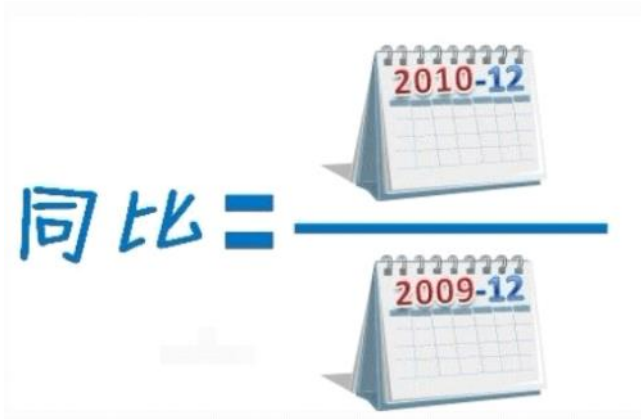
## 02.同比与环比

2020年12月15日 11:13

**同比率=(本期-去年同期)/去年同期×100%**

**环比率=(本期-上期)/上期×100%**

上期可以是上周, 上月, 上季度, 甚至上一日



[同比\\_百度百科 \(baidu.com\)](http://baike.baidu.com)

## 01.同比推荐：SAMEPERIODLASTYEAR【前一年】

2020年12月15日 11:01

### SAMEPERIODLASTYEAR ( <日期列> )

✕ ✓	1 返回的是一张表 = SAMEPERIODLASTYEAR('表'[日期])
日期	
2020/1/1 0:00:00	
2020/3/2 0:00:00	
2020/4/1 0:00:00	
2020/6/3 0:00:00	
2020/6/30 0:00:00	

官方：返回当前筛选上下文中前一年的一组日期

说人话：

在你的表中，找到日期列最大的年、月、日  
找到它的同比时间

举例：

日期	销售
2020年1月1日	1
2020年1月2日	0
2020年3月2日	2
2020年4月1日	3
2020年6月3日	4
2020年6月30日	5
2020年12月1日	6
2020年12月31日	7
2021年1月1日	10
2021年3月2日	20
2021年4月1日	30
2021年6月3日	40
2021年6月30日	50

先确定好哪个是今年，然后再找到去年

计算同比：

同比销售金额 = CALCULATE([总销售], SAMEPERIODLASTYEAR('表'[日期]))

结果就是：(1+5)\*5/2=15

## 02.环比: DATEADD

2020年12月15日 11:18

**DATEADD ( <日期列>, <偏移量>, <偏移单位> )**

参数	属性	描述
日期列	第1参	包含日期的列
偏移量	第2参	一个整数, 从日期列中添加或减去的时间间隔数
偏移单位	第3参	偏移量使用的单位: Day, Month, Quarter, Year

返回是一张表, 放到新建列里试一下:

返回的是一张表 = DATEADD('DATEADD' [日期], 0, YEAR)

日期
2001/1/1 0:00:00
2002/1/1 0:00:00
2003/1/1 0:00:00
2004/1/1 0:00:00
2005/1/1 0:00:00
2006/1/1 0:00:00
2007/1/1 0:00:00
2008/1/1 0:00:00
2009/1/1 0:00:00
2010/1/1 0:00:00
2011/1/1 0:00:00
2012/1/1 0:00:00
2013/1/1 0:00:00
2014/1/1 0:00:00
2015/1/1 0:00:00
2016/1/1 0:00:00
2017/1/1 0:00:00
2018/1/1 0:00:00
2019/1/1 0:00:00
2020/1/1 0:00:00

当第2参数为1的时候: 2001年不见了

日期
2002/1/1 0:00:00
2003/1/1 0:00:00
2004/1/1 0:00:00
2005/1/1 0:00:00
2006/1/1 0:00:00
2007/1/1 0:00:00
2008/1/1 0:00:00
2009/1/1 0:00:00
2010/1/1 0:00:00
2011/1/1 0:00:00
2012/1/1 0:00:00
2013/1/1 0:00:00
2014/1/1 0:00:00
2015/1/1 0:00:00
2016/1/1 0:00:00
2017/1/1 0:00:00
2018/1/1 0:00:00
2019/1/1 0:00:00
2020/1/1 0:00:00

意思就是:

2001, 2002, 2003, ....2019, 2020

当第3参数为1的时候, 向右平移1年

2002, 2003, ....2019, 2020

当第2参数为-1的时候: 2020年不见了

日期
2001/1/1 0:00:00
2002/1/1 0:00:00
2003/1/1 0:00:00
2004/1/1 0:00:00
2005/1/1 0:00:00
2006/1/1 0:00:00
2007/1/1 0:00:00
2008/1/1 0:00:00
2009/1/1 0:00:00
2010/1/1 0:00:00
2011/1/1 0:00:00
2012/1/1 0:00:00
2013/1/1 0:00:00
2014/1/1 0:00:00
2015/1/1 0:00:00
2016/1/1 0:00:00
2017/1/1 0:00:00
2018/1/1 0:00:00
2019/1/1 0:00:00

计算环比之前, 要先看看他的规律

总销售 = sum('表2'[销售])

上年 = CALCULATE([总销售], DATEADD('表2'[日期], -1, YEAR))

上月 = CALCULATE([总销售], DATEADD('表2'[日期], -1, MONTH))

上季 = CALCULATE([总销售], DATEADD('表2'[日期], -1, Quarter))

日期	总销售	上年	上月	上季
2019年4月1日	1			
2019年5月1日	2		1	
2019年6月1日	3		2	
2019年7月1日	4		3	1
2019年8月1日	5		4	2
2019年9月1日	6		5	3
2019年10月1日	7		6	4

2011/1/1 0:00:00
2012/1/1 0:00:00
2013/1/1 0:00:00
2014/1/1 0:00:00
2015/1/1 0:00:00
2016/1/1 0:00:00
2017/1/1 0:00:00
2018/1/1 0:00:00
2019/1/1 0:00:00

日期	总销售	上年	上月	上季
2019年4月1日	1			
2019年5月1日	2		1	
2019年6月1日	3		2	
2019年7月1日	4		3	1
2019年8月1日	5		4	2
2019年9月1日	6		5	3
2019年10月1日	7		6	4
2019年11月1日	8		7	5
2019年12月1日	9		8	6
2020年1月1日	10		9	7
2020年2月1日	11		10	8
2020年3月1日	12		11	9
2020年4月1日	13	1	12	10
总计	91	1	78	55

去年

算环比:

本月销售 = TOTALMTD([总销售], '表2'[日期])

上月销售 = TOTALMTD([总销售], DATEADD('表2'[日期], -1, MONTH))

以前, DATEADD要求是连续日期, 且必须使用日期表的日期, 现在的版本改进了

### 03.利用时间计算累积值 PARALLELPERIOD

2020年12月15日 13:04

区别：

- 1、PARALLELPERIOD函数返回的是**完整的时间范围**，而DATEADD函数返回的结果**可以是间断的**。
- 2、DATEADD函数通常用来计算环比同比问题，针对的是某一个点；而PARALLELPERIOD函数针对的是一段范围的数据汇总。

**PARALLELPERIOD ( <日期列>, <偏移量>, <间隔单位> )**

属性	描述
第1参数	包含日期的列
第2参数	一个整数，从日期列中添加或减去的时间间隔数；正数向未来推移，反之向过去推移
第3参数	日期偏移的间隔，可以是以下值之一： <code>year</code> 、 <code>quarter</code> 、 <code>month</code>

因为我的案例每个月只有一天的数据，所以我就不用月打比方了，因为每个月的总和仍然是这一天的数字。我用季度打比方

总销售 = sum('表2'[销售])

PARALLELPERIOD = CALCULATE([总销售],PARALLELPERIOD('表2'[日期], -1, QUARTER))

DATEADD = CALCULATE([总销售],DATEADD('表2'[日期], -1, QUARTER))

日期	总销售	DATEADD	PARALLELPERIOD
2019年4月1日	1		
2019年5月1日	2		
2019年6月1日	3		
2019年7月1日	4	1	6
2019年8月1日	5	2	6
2019年9月1日	6	3	6
2019年10月1日	7	4	15
2019年11月1日	8	5	15
2019年12月1日	9	6	15
2020年1月1日	10	7	24
2020年2月1日	11	8	24
2020年3月1日	12	9	24
2020年4月1日	13	10	33
总计	91	55	78

本季销售 = CALCULATE([总销售],PARALLELPERIOD('表2'[日期], 0, QUARTER))

上季销售 = CALCULATE([总销售],PARALLELPERIOD('表2'[日期], -1, QUARTER))

本季销售1 = TOTALQTD([总销售],PARALLELPERIOD('表2'[日期],0,QUARTER))

上季销售1 = TOTALQTD([总销售],PARALLELPERIOD('表2'[日期],-1,QUARTER))





## 03.计算移动总计

2020年12月15日 20:03

说人话：**移动**就是指开始日期,向前或向后多少天、月、季度、年

# 01.DATESINPERIOD

2020年12月15日 20:08

**DATESINPERIOD ( <日期列>, <起始日期>, <偏移量>, <间隔单位> )**

属性	描述
第1参数	包含日期的列
第2参数	日期表达式
第3参数	一个整数，从日期列中添加或减去的时间间隔数；正数向未来推移，反之向过去推移
第4参数	日期偏移的间隔，可以是以下值之一：year、quarter、month

返回给定区间中的所有日期组成的单列形式的表

日期	销售
2019年4月1日	1
2019年5月1日	2
2019年6月1日	3
2019年7月1日	4
2019年8月1日	5
2019年9月1日	6
2019年10月1日	7
2019年11月1日	8
2019年12月1日	9
2020年1月1日	10
2020年2月1日	11
2020年3月1日	12
2020年4月1日	13
2020年5月1日	14
2020年6月1日	15

例如：统计你数据表中，最后一天开始的最近 12 个月的销售额

总销售 = SUM('表3'[销售])

测试 = CALCULATE ([总销售], DATESINPERIOD ('表3'[日期], MAX ('表3'[日期]), -1, YEAR ))

表3中最大日期向前一年

$$(1+15) * 15 / 2 = 120 - 6 = 114$$

LastDate  
FirstDate

日期
2019/7/1 0:00:00
2019/8/1 0:00:00
2019/9/1 0:00:00
2019/10/1 0:00:00
2019/11/1 0:00:00
2019/12/1 0:00:00
2020/1/1 0:00:00
2020/2/1 0:00:00
2020/3/1 0:00:00
2020/4/1 0:00:00
2020/5/1 0:00:00
2020/6/1 0:00:00

例如：求最近30天的移动平均

总销量 = SUM('表4'[销量])

移动平均 = Calculate([总销量], DATESINPERIOD('表4'[日期], max('表4'[日期]), -30, DAY)) / 30

从表4的日期列中，找最大日期，向前移动30天的这张表

拿到这30天的销售量之后，除以30天，算移动平均值

**拓展知识：**从上月算起，过去的三个月的日期

DATESINPERIOD('某个表'[日期], EOMONTH(TODAY(), -1), -3, MONTH)

笔记19.03 平移指定月份

## 补充1【拓展知识】LastDate与max的区别

2020年12月25日 23:14

LastDate返回的是一个表，max是一个值

销售表中的商品数量 = Calculate(CountRows('商品表'), '销售表')

还记得第7课学的这个公式？Calculate第二参数是筛选器，可以是一张表

calculate(【度量值】，LastDate('某张表'[日期列]))

calculate(【度量值】，filter('某张表', '某张表'[日期列] = Max('某张表'[日期列])))

当用于Calculate或CaluculateTable筛选器时，我们习惯用LastDate或FirstDate而不是用Max或Min

## 补充2【动态日期表】

2020年12月26日 22:08

```
日期表 = ADDCOLUMNS(  
    CALENDAR(FIRSTDATE('表'[日期]),LASTDATE('表'[日期])),  
    "年", YEAR ( [Date] ),  
    "季度", ROUNDUP(MONTH([Date])/3,0),  
    "月", MONTH([Date]),  
    "周", weeknum([Date]),  
    "年季度", year([date]) & "Q" &  
    ROUNDUP(MONTH([Date])/3,0),  
    "年月", year([Date]) * 100 + MONTH([Date]),  
    "年周", year([Date]) * 100 + weeknum([Date]),  
    "星期几", WEEKDAY([Date])  
)
```

## 02.ENDOFMONTH与STARTOFMONTH

2020年12月15日 20:31

**ENDOFMONTH(日期列)** # 返回当月最后一天

**STARTOFMONTH(日期列)** # 返回当月第一天

每月最后3天销量 = CALCULATE([总销量],DATESINPERIOD('表4'[日期],**ENDOFMONTH**('表4'[日期]),-3,DAY))

月份	每月最后3天销量
January	153
February	86
March	188
April	240
May	125
June	174
July	199
August	198
September	183
October	106
November	91
December	153
总计	153

每月前3天销量 = CALCULATE([总销量],DATESINPERIOD('表4'[日期],**STARTOFMONTH**('表4'[日期]),3,DAY))

月份	每月前3天销量
January	99
February	77
March	200
April	126
May	133
June	141
July	140
August	140
September	160
October	217
November	230
December	249
总计	99

进阶:

**ENDOFMONTH**('表4'[日期]) 返回当前上下文中最后一个日期，类似Max或LastDate

**STARTOFMONTH**('表4'[日期]) 返回当前上下文中第一个日期，类似Min或FirstDate

# [季度] ENDOFQUARTER与STARTOFQUARTER

2020年12月15日 20:34

ENDOFOQUARTER(日期列)	# 返回当季度最后一天
STARTOFOQUARTER(日期列)	# 返回当季度第一天

# 【年】 ENDOFYEAR与STARTOFYEAR

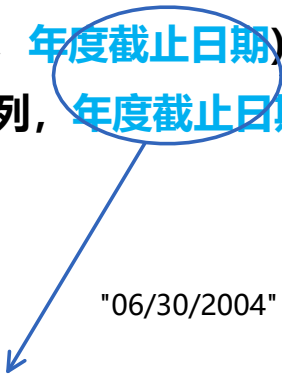
2020年12月15日 20:34

ENDOFYEAR(日期列, 年度截止日期)

# 返回当年最后一天

STARTOFYEAR(日期列, 年度截止日期)

# 返回当年第一天



"06/30/2004"

包含日期的文本字符串，用于定义年末日期。默认值为 12 月 31 日

但是本人亲测这里不生效，可能是我操作不对，欢迎大家查实指证

[https://docs.microsoft.com/zh-cn/previous-versions/sql/sql-server-2014/ee634245\(v=sql.120\)?redirectedfrom=MSDN](https://docs.microsoft.com/zh-cn/previous-versions/sql/sql-server-2014/ee634245(v=sql.120)?redirectedfrom=MSDN)

# 润年处理方法

2020年12月15日 11:00

**你何必去考虑这个问题？**

**你计算环比时，有没有考虑过4月份是30天，环比3月份是31天呢？**

**那同比时，何必考虑润年？**



### 03.区间日期Datesbetween 【推荐新人使用】

2020年12月25日 23:32

**Datesbetween** ('表'[日期],开始日期, 结束日期)

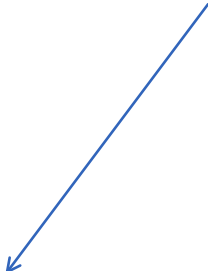
总销量 = sum('表4'[销量])

指定时间销量 = CALCULATE([总销量],**DATESBETWEEN('表4'[日期],DATE(2020,1,1),DATE(2020,1,2))**)

指定时间销量 = CALCULATE([总销量],**DATESBETWEEN('表4'[日期],DATE(2020,1,2),DATE(2020,1,1))**)

注意:

如果开始日期大于结束日期, 结果是空。



当开始日期大于结束日期时, 这个表返回为空  
在Calculate中筛选器为空就是没有筛选, 所以结果还是【总销量】这个度量值

# 04.NEXT系列函数【下一个】

2020年12月27日 8:56

表函数:

NEXTDAY【下一天】，NEXTMONTH【下一月】，NEXTQUARTER【下一季】，NEXTYEAR【下一年】

例如:

DATESBETWEEN ( '表'[日期], DATE ( 2020, 1, 1 ), DATE ( 2020, 2, 28 ) )

返回: 2020/1/1 至 2020/2/28 这个期间的日期

如果将NextDAY套在这个公式最外层，就得到了2020/2/29这个日期的表。

案例: NextDAY可以实现错位值

日期	销售金额	NextDay
2019年1月1日	1	2
2019年1月2日	2	3
2019年1月3日	3	4
2019年1月4日	4	5
2019年1月5日	5	6
2019年1月6日	6	7
2019年1月7日	7	8
2019年1月8日	8	9

NextDay = CALCULATE(SUM('NEXT'[销售金额]), NEXTDAY('NEXT'[日期]))

例如:

DATESBETWEEN ( '表'[日期], DATE ( 2020, 1, 1 ), DATE ( 2020, 2, 28 ) )

返回: 2020/1/1 至 2020/2/28 这个期间的日期

如果将NextMONTH套在这个公式最外层，就得到了2020/3/1 至 2020/3/31 这个期间的日期表。

案例: NextMONTH可实现下个月销售的总计

日期	销售金额	NextDay	NEXTMONTH
2019年1月1日	1	2	1274
2019年1月2日	2	3	1274
2019年1月3日	3	4	1274
2019年1月4日	4	5	1274
2019年1月5日	5	6	1274
2019年1月6日	6	7	1274
2019年1月7日	7	8	1274
2019年1月8日	8	9	1274
2019年1月9日	9	10	1274
2019年1月10日	10	11	1274
2019年1月11日	11	12	1274
总计	267546		

**NEXTMONTH = CALCULATE(SUM('NEXT'[销售金额]), NEXTMONTH ('NEXT'[日期]))**

**那么季度和年同月份是一样，得到下个季度或下一年的全部日期**

# 【上一个】PREVIOUS系列

2020年12月27日 8:58

## 表函数

PREVIOUSDAY上一天

PREVIOUSMONTH上一月

PREVIOUSQUARTER上一季

PREVIOUSYEAR上一年

## 05.【期初库存】OPENINGBALANCE系列

2020年12月27日 8:57

“值函数”

OPENINGBALANCEMONTH函数

OPENINGBALANCEQUARTER函数

OPENINGBALANCEYEAR函数

用途：计算月/季度/年的期初库存

语法：函数(<表达式>,<日期>[,<筛选器>])

**月初库存 = OPENINGBALANCEMONTH ( SUM ( '期初'[库存] ), '期初'[日期] )**

注意：所谓的月初库存就是上月底最后一天的库存

比如：2019年3月的期初库存，就是2019年2月28日闭店后，结算完毕，得到的库存数。

我们的数据最小是2019年1月1日，没有2018年12月31日，所以2019年1月不显示期初库存

## 28.筛选器函数

2020年12月14日 11:28

## 01.REMOVEFILTERS 【不推荐】

2020年12月26日 16:52

是“**筛选**”函数，其本身不属于表函数，也不属于值函数，仅作为**CALCULATE**函数的调节器使用。

我们之前接触的函数，要么是返回表，要么是返回值。

**REMOVEFILTERS**



移除筛选时  
我们多了一种选择

度量值 = CALCULATE(SUM('REMOVEFILTERS'[销售金额]),REMOVEFILTERS('REMOVEFILTERS'))

度量值 = CALCULATE(SUM('REMOVEFILTERS'[销售金额]),REMOVEFILTERS('REMOVEFILTERS'[日期]))

跟ALL函数功能一样，但是只能放到Calculate的筛选器里，它是“筛选函数”

注意ALL特性：REMOVEFILTERS与ALL特性相同

- 【1】当ALL参数为表时，忽略所有的筛选条件，无论是该图表内还是外部切片器
- 【2】当ALL参数为列时，忽略该列筛选，其它图表字段或外部筛选对其产生作用
- 【3】ALL函数在引用列的时候，必需与矩阵的行和列在同一张表

## 02.KEEPFILTERS【追加筛选】

2020年12月26日 17:12



# KEEPFILTERS 追加筛选

A商品销售 = CALCULATE(sum('KEEPFILTERS'[销售金额]),'KEEPFILTERS'[商品]="A")

A商品销售keep = CALCULATE(sum('KEEPFILTERS'[销售金额]),KEEPFILTERS('KEEPFILTERS'[商品]="A"))

日期	销售金额	A商品销售	A商品销售keep
2020年12月1日	1	1	1
2020年12月2日	2	2	2
2020年12月3日	3		
2020年12月4日	4		
2020年12月5日	5		
2020年12月6日	6		
2020年12月7日	7	7	7
2020年12月8日	8		
2020年12月9日	9		
2020年12月10日	10	10	10
总计	55	20	20

商品	销售金额	A商品销售	A商品销售keep
A	20	20	20
B	15	20	
C	20	20	
总计	55	20	20

将条件限定为“A”，因此这一条件直接覆盖了当前列中的其他筛选条件，只显示这一项。

而使用KEEPFILTERS函数的结果，是将“A”这一条件，追加到当前筛选中。

举个例子：有人不讲“武德”使用卑鄙手段注册小号恶意攻击他人，这个人是“A”

CALCULATE是找到“A”这个人，发现他是某某组织的，那这个组织一起跟着被曝光。

而KEEPFILTERS是追加筛选是什么意思？就是找到“A”这个人了，和其他人无关，只对“A”进行曝光。



A商品销售values = CALCULATE(SUM('KEEPFILTERS'[销售金额]),'KEEPFILTERS'[商品]= "A",VALUES('KEEPFILTERS'[商品]))

商品	销售金额	A商品销售	A商品销售keep	A商品销售values
A	20	20	20	20
B	15	20		
C	20	20		
总计	55	20	20	20

KEEPFILTER是内部条件与外部筛选取交集；**CALCULATE是内部条件取交集**，不涉及外部。

A商品销售values = CALCULATE(SUM('KEEPFILTERS'[销售金额]),'KEEPFILTERS'[商品]= "A",VALUES('KEEPFILTERS'[商品]))

单独这一个条件的时候，它得到的是A商品的销售金额，可以通过日期筛选它，但是无法通过商品进行筛选

我们再给它一个条件，让他用商品去筛选

### 03.CROSSFILTER【交叉筛选】

2020年12月26日 19:47

**CROSSFILTER(多端固定列名, 一端固定列名, 方向)**

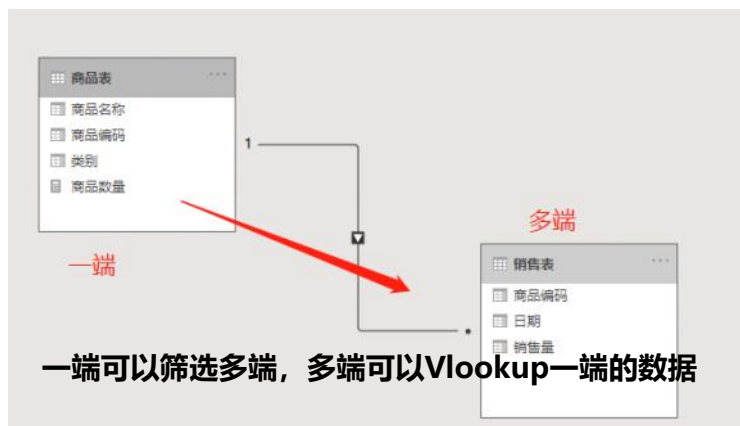
必须是固定列名，不能是表达式生成的

方向：ONEWAY：单向筛选；BOTH：双向筛选；NONE：无交叉筛选。

功能一：改变筛选方向

商品编码 商品数量

A001	5
A002	5
A003	5
总计	5



商品数量 = COUNTROWS (RELATEDTABLE(' 商品表' ))

商品编码 品牌数量 改变连接方向后

A001	5	1
A002	5	1
A003	5	1
总计	5	5

度量值 = CALCULATE(COUNTROWS(RELATEDTABLE('商品表')),CROSSFILTER('销售表'[商品编码],'商品表'[商品编码],Both))

功能二：使用模型关系筛选时，数量过大会导致模型运载变慢，这个时候可以使用CROSSFILTER函数进行优化

商品编码 销售表数量 改变连接方向后

A001	6	6
A002	7	7
A003	8	8
总计	21	21

销售表数量 = COUNTROWS('销售表')

改变连接方向后 =

CALCULATE(COUNTROWS(RELATEDTABLE('销售表')),CROSSFILTER('销售表'[商品编码],'商品表'[商品编码],OneWay))

## 注意事项：CROSSFILTER

2020年12月26日 21:40

- 1、如果模型关系是一对一的情况，使用ONEWAY和BOTH没区别。
- 2、如果多端列和一端列位置反了，函数本身会自我修正。
- 3、此函数只能在接受筛选器作为参数的函数中使用：

CALCULATE系列【包括CalculateTable】、TOTAL系列 其它系列我们没有讲

- 4、CROSSFILTER函数会覆盖任何现有筛选关系。
- 5、如果两个参数没有任何链接关系，那么返回结果会报错。
- 6、如果使用多个CROSSFILTER，最内层的会覆盖外面的。

## 04.USERRELATIONSHIP

2020年12月26日 21:43

用途:

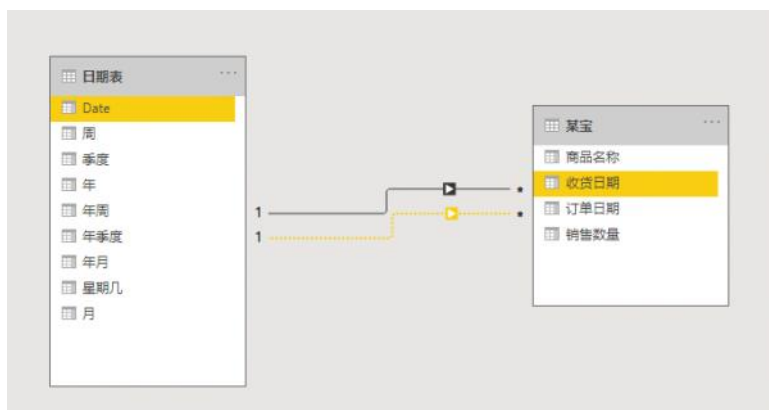
- 1、用来激活指定的关系;
- 2、适用于做关联度分析 (购物篮)



### P20 19.实战分析002.购物篮分析

[Excel黑科技! Power Pivot & Power BI教程, 让你站在Excel肩膀上【本季完】PowerBI教程](#)  
[哔哩哔哩 \(゜-゜\)つロ 干杯~~bilibili](#)

案例: 处理订单数量与收货数量



下单数量 = SUM ( '某宝'[销售数量] )

送达数量 = CALCULATE([下单数量], USERELATIONSHIP ( '某宝'[收货日期], '日期表'[Date] ))

月	下单数量	送达数量
1	25	11
总计	25	14

还没收货的数量

收货数量

筛选器

搜索

此视觉对象上的筛选器

下单数量 是(全部)

月 不为空

筛选类型 高级筛选

显示值满足以下条件的项: 不为空

且 或

应用筛选器

# 动态日期表

2020年12月27日 7:23

```
日期表 = ADDCOLUMNS(  
    CALENDAR(FIRSTDATE('某宝'[订单日期]),LASTDATE('某宝'[收货日期])),  
    "年", YEAR ( [Date] ),  
    "季度", ROUNDUP(MONTH([Date])/3,0),  
    "月", MONTH([Date]),  
    "周", weeknum([Date]),  
    "年季度", year([date]) & "Q" & ROUNDUP(MONTH([Date])/3,0),  
    "年月", year([Date]) * 100 + MONTH([Date]),  
    "年周", year([Date]) * 100 + weeknum([Date]),  
    "星期几", WEEKDAY([Date])  
)
```

05.【案例】USERRELATIONSHIP案例

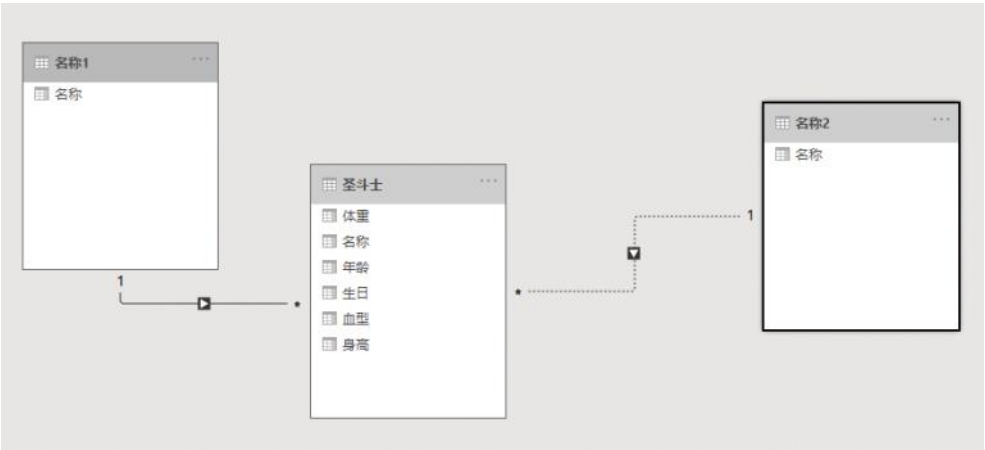
2020年12月27日 12:37

名称	左	项目	右	名称
<input type="checkbox"/> 阿波罗迪		阿鲁迪巴 名称	艾俄洛斯	<input type="checkbox"/> 阿波罗迪
<input checked="" type="checkbox"/> 阿鲁迪巴		20.00 年龄	14.00	<input type="checkbox"/> 阿鲁迪巴
<input type="checkbox"/> 艾奥里亚				<input type="checkbox"/> 艾奥里亚
<input type="checkbox"/> 艾俄洛斯				<input checked="" type="checkbox"/> 艾俄洛斯
<input type="checkbox"/> 迪斯马斯克				<input type="checkbox"/> 迪斯马斯克...
<input type="checkbox"/> 卡妙				<input type="checkbox"/> 卡妙
<input type="checkbox"/> 米罗				<input type="checkbox"/> 米罗
<input type="checkbox"/> 穆				<input type="checkbox"/> 穆
<input type="checkbox"/> 撒加				<input type="checkbox"/> 撒加
<input type="checkbox"/> 沙加				<input type="checkbox"/> 沙加
<input type="checkbox"/> 童虎				<input type="checkbox"/> 童虎

一、新建两张不重复的姓名表

名称1 = VALUES('圣斗士'[名称])

名称2 = VALUES('圣斗士'[名称])



总年龄A = sum('圣斗士'[年龄])

总年龄B = CALCULATE([总年龄A],all('名称1'[名称]),USERRELATIONSHIP('圣斗士'[名称],'名称2'[名称]))

新建一个表，方法你随意，我就手工建立了

索引	项目
1	名称
2	年龄

左 =



```
左 =  
VAR a =  
    SELECTEDVALUE ( '表'[索引] )  
VAR b =  
    SWITCH (  
        TRUE (),  
        a = "1", SELECTEDVALUE ( '名称1'[名称] ),  
        a = "2", [总年龄A]  
    )  
RETURN  
b
```

```
右 =  
VAR a =  
    SELECTEDVALUE ( '表'[索引] )  
VAR b =  
    SWITCH (  
        TRUE (),  
        a = "1", SELECTEDVALUE ( '名称2'[名称] ),  
        a = "2", [总年龄B]  
    )  
RETURN  
b
```

# 29.中文排序与连接MySQL数据库

2020年12月26日 22:05



# 01.中文排序

2020年12月26日 22:14

操作步骤详见视频

都是鼠标点击，不方便制作笔记，请见谅

## 02.连接MySQL数据库

2020年12月26日 22:14

操作步骤详见视频

都是鼠标点击，不方便制作笔记，请见谅

### 30.图表与案例与书籍网站推荐

2020年12月26日 22:14

微软官方DAX语言查询：中文版

<https://docs.microsoft.com/zh-cn/dax/new-dax-functions>

Dax圣经

<https://www.powerbigeek.com/what-is-dax/>

powerbi进阶篇 第一季 销售案例

<https://www.bilibili.com/video/BV18C4y1H7b9>

powerbi进阶篇 第二季 商品案例

<https://www.bilibili.com/video/BV14Z4y1p7fy>

powerbi进阶篇 第三季 人力与财务案例

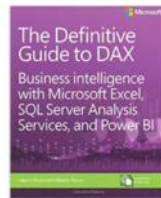
<https://www.bilibili.com/video/BV1Je411W76y>

powerbi进阶篇 第四季 可视化图表

<https://www.bilibili.com/video/BV1W54y1i7dE>

Powerbi进阶篇 第五季 自定义图表

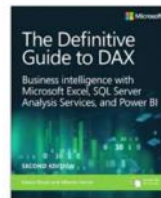
<https://www.bilibili.com/video/BV1r54y1i75n>



第一版



Marco Russo



第二版



Alberto Ferrari



成员

ArcSaber

#4009

这个网站很好，网上摄入理解dax的文章并不多。但是问题也很突出，那就是，这个网站仅仅是“翻译”而已，而不是通过作者理解而去“复述”。

我看这篇文章就像是在看本科学生翻译的论文。

👍 0

🗨️ 回复

🕒 2天之前 | 隐藏回复 (1) ^



作者

高飞

#4013

感谢建议，个人精力有限，先把内容放上来，后面会逐步迭代，网站会越办越好的 😊

网站的作者很好，帮助英语不好的人，在此表示感谢

但是大家不要学某位UP主

这只是理论，并没有联系实际，只有联系实际的理论才有用！

以前没电脑时，世界人民照样分析

以前没有PBI，世界人民照样分析

现在有了PBI，是提高我们的效率，不是降低我们对业务的学习

能不能分析好，你的业务逻辑才是最重要的

如果一个UP在网上录视频，是把书上逻辑讲出来，收视率肯定低呀

如果你是自己理解，联系自己的工作经验，把自己的想法讲出来，效果就不一样了。