

3.4 Notion 函数基础入门（二）：掌握更自由的打卡统计与日期间隔计算

布尔值与打卡计算从本文开始我将逐渐涉及一些相对复杂和烦琐的 Notion 函数书写，不过请各位不要看到长长的一串公式就感到害怕，其实很多时候它只是看着长一点、密集一点，我们唯一需要的就是耐住性子，然后花一.....

布尔值与打卡计算

从本文开始我将逐渐涉及一些相对复杂和烦琐的 Notion 函数书写，不过请各位不要看到长长的一串公式就感到害怕，其实很多时候它只是看着长一点、密集一点，我们唯一需要的就是耐住性子，然后花一点点时间即可掌握这些函数的原理和写法。

基础概念

在上一篇文章中，我们介绍了 Notion 的数字、日期、以及字符串这三种数据类型，而 Checkbox 则是剩下的最后一种类型，即布尔值。
布尔值是一种计算机编程术语，并且有真（True）和假（False）之分，其中

True=1, False=0，基于这点我们可以利用布尔值在 Notion Formula 中进行一些比较复杂的操作。

Checkbox 与布尔值

在 Notion Formula 中，我们可以将 Checkbox 的打勾和非打勾状态分别对应布尔值的 True 和 False。
例如在下图中我创建了一个 Checkbox 字段和 Formula 字段，当我用

`toNumber()` 函数将 Checkbox 转换为数字的时候可以看到，打勾的值是 1，不打勾的值是 0，如此一来我们就可以对 Checkbox 进行各种计算统计了。

布尔值

1 more property

Table

Aa Name	☒ Checkbox	Σ Formula	+ ...
	☒	1	
	☐	Notion Formula ⓘ Checkbox . toNumber() = 0	
	☐		
+ New		No results	

逻辑判断与布尔值

在 Formula 中，我们可以直接使用 $>$ 、 $<$ 或者两个等号 $=$ 来进行逻辑判断。

例如在下面的数据库中，我们用公式来判断「消费额」这个字段是否超过了 300，如果超过，则布尔值为 true 并将自动显示为打钩状态，否则布尔值为 false 并显示为非打钩状态：

Table			
#	营业额	Σ 是否超过 300	Aa Name
	100	☐	
	400	☒	
	500	☒	
+ New			
Notion Formula ?			
营业额 > 300			
= ☒			

如此一来我们就可以对这个 Formula 进行分组操作，分别是「打勾组」和「非打勾组」：

Table +

▼

☒

3

⋮

+

Aa Name

营业额

Σ 营业额大于 1 万

20000

☒

21000

☒

11000

☒

+ New

▼

☐

4

⋮

+

Aa Name

营业额

Σ 营业额大于 1 万

5000

☐

3000

☐

4000

☐

3500

☐

+ New

Filter Sort 🔍 ↶ ⋮ New ▼

← Group ×

Group by

营业额大于 1 万 >

Hide empty groups ☒

Visible groups

Hide all

⋮

☒

👁

⋮

☐

👁

🗑 Remove grouping

🔗 Learn about grouping

另外，我们还可以用两个等号 `==` 来判断两个字段是否相同，如果相同则为 `true`，不同则为 `false`：

```
prop("Number 1") == prop("Number 2")
```

Table

Aa Name	# Number 1	# Number 2	Σ 判断字段是否相同	+ ...
	100	100	☒	
	200	300	☐	
	300	300	☒	

日期判断与布尔值

除了可以直接对数字进行大小对比，我们也同样可以用

`>`、`<`、`==` 这三个符号来对比日期并获得相应的布尔值：

Table

Aa Name	📅 Date 1	📅 Date 2	Σ Formula	+ ...
	2024/01/01	2024/01/07	☐	
	2024/01/22	2024/01/01	☒	

+ New

Notion Formula ?

Date 1 > Date 2

= ☒

Checkbox 与打卡计算

在前面的基础介绍中，我们已经可以用 `toNumber()` 这个函数来将 Checkbox 的打勾和非打勾状态转换为数字 1 或者 0，如此一来我们就可以去计算多个 Checkbox 的打勾百分比：

Table				
Aa Name	☒ Checkbox 1	☒ Checkbox 2	Σ Formula	+ ...
	☒	☒	2	
	☒	☐	1	
	☐	☒	1	
+ New				

Notion Formula ?

Checkbox 1.toNumber()+Checkbox 2.toNumber()

= 1

在下面的案例中，我们将一周七天的 Checkbox 都用

`toNumber()` 函数转换为数字，将这些数字相加，并将求得的和除以 7，如此就能计算每周的打卡完成度：

Table

Aa Name	☒	☒	☒	☒	☒	☒	☒	Σ 每周打卡完成度	Σ 百分比进度	+ ...
打卡习惯 01	☒	☐	☒	☐	☒	☒	☐	0.571428571429	57%	
打卡习惯 02	☒	☒	☒	☒	☒	☒	☐	Notion Formula ? (星期 1.toNumber()+ 星期 2.toNumber()+ 星期 3.toNumber()+ 星期 4.toNumber()+ 星期 5.toNumber()+ 星期 6.toNumber()+ 星期 7.toNumber()) / 7 = 0.714285714286		
打卡习惯 03	☒	☐	☒	☐	☐	☒	☒			
打卡习惯 04	☒	☐	☐	☐	☒	☒	☒			
打卡习惯 05	☒	☐	☒	☐	☒	☒	☒			
+ New										

公式写法可参考：

```
(
  prop("星期 1").toNumber()+
  prop("星期 2").toNumber()+
  prop("星期 3").toNumber()+
  prop("星期 4").toNumber()+
  prop("星期 5").toNumber()+
  prop("星期 6").toNumber()+
  prop("星期 7").toNumber()

) / 7
```

但这时你可能会发现，上面的计算结果的小数位有点多，就算将 Formula 字段的数字格式设置为百分比（percent）的话，似乎也无法保留小数点后两位：

Table

Aa Name	☒	☒	☒	☒	☒	☒	☒	Σ 每周打卡完成度	
打卡习惯 01	☒	☐	☒	☐	☒	☒	☐	57.1428571429%	
打卡习惯 02	☒	☒	☒	☒	☒	☒	☐	85.7142857143%	
打卡习惯 03	☒	☐	☒	☐	☐	☒	☒	57.1428571429%	
打卡习惯 04	☒	☐	☐	☐	☒	☒	☒	57.1428571429%	
打卡习惯 05	☒	☐	☒	☐	☒	☒	☒	71.4285714286%	
+ New									

这里我们可以用 `round()` 这个函数来进行优化操作，它的作用是对数字进行四舍五入，如下图所示：

Table

Aa Name	# 数字	Σ round 函数	+ ...
	1001	1001	
	4.4	4	
	4.6	5	
	0.4	0	
	0.6	1	

所以如果我们希望将 `0.857142` 变成

`86%`，就需要以下 3 个步骤：

1. 将 `0.857142` 扩大 100 倍，变成 `85.7142`
2. 使用 `round()` 函数将 `85.7142` 四舍五入取整为 `86`
3. 将 `86` 缩小 100 倍，变成 `0.86`

所以公式写法如下：

```
round(prop("每周打卡完成度") * 100) / 100
```

Table

Aa Name	☒	☒	☒	☒	☒	☒	☒	Σ 每周打卡完成度	Σ 百分比进度	+	...
打卡习惯 01	☒	☐	☒	☐	☒	☒	☐	57.1428571429%	0.57		
打卡习惯 02	☒	☒	☒	☒	☒	☒	☐	85.7142857143%	0.86		
打卡习惯 03	☒	☐	☒	☐	☐	☒	☒	57.1428571429%	0.57		
打卡习惯 04	☒	☐	☐	☐	☒	☒	☒	57.1428571429%	0.57		
打卡习惯 05	☒	☐	☒	☐	☒	☒	☒	71.4285714286%	0.71		
+ New											

Notion Formula ?

round(每周打卡完成度 * 100) / 100

= 0.71

另外再补充两个也许你可能会用到的函数：

`floor()`

函数的作用是将数字的小数位全部舍去，即向下取整

Table

Aa Name	# Number	Σ floor()	+	...
	2.1	2		
	2.4	2		
	2.7	2		
	1000.8888	1000		
+ New				

`ceil()` 函数的作用则是向上取整

Table

Aa Name	# Number	Σ ceil()	+	...
	2.1	3		
	2.3	3		
	2.7	3		
+ New				

日期间隔计算（`dateBetween`）

日期间隔计算同样是 Notion 函数中相对高频的一项需求，它的基础写法如下，请注意

`dateBetween`

的

B 不能小写

```
dateBetween(prop("时间 1"), prop("时间 2"), "days")
```

这个公式的计算逻辑是，用 时间 1 减去

时间 2，然后用第三个参数 “days”

来作为相差时间的单位，如下图所示：

```
dateBetween(prop("开始时间"), prop("结束时间"), "days")
```

Table

Aa Name	开始时间	结束时间	Σ 时间间隔	+ ...
项目 01	2023/06/10	2023/04/25	46	
项目 02	2023/04/26	2023/04/03	23	
+ New				

Notion Formula ?

dateBetween(开始时间 , 结束时间 , "days")

= 23

注意这里的 days

还可以按照实际的计算需求，替换下面的另外几种单位：

日期间隔的参数	含义
years	年数差
months	月数差
days	天数差
hours	小时差
minutes	分钟数差
seconds	秒数差

所以在相同的日期计算结果之下，如果用不同的日期间隔参数，就可以求出不同的效果，如下图所示：

Table

Date 1	Date 2	Σ 年数差	Σ 月数差	Σ 天数差	Σ 小时差	Σ 分钟差	Σ 秒数差	Aa Name	+ ...
2023/04/22	2023/04/30	0	0	-8	-192	-11520	-691200		
2023/04/22	2021/04/22	2	24	730	17520	1051200	63072000	dateBetween(prop("Date 1"), prop("Date 2"), "seconds")	Done

案例

1、计算单个日期字段内的时间差

当我们在 Date 字段中开启了 End time

的选项后，我们将可以同时设定它的开始时间与结束时间：

项目排期表

Table

Aa Name	📅 项目排期	+ ...
项目 01	2023/07/01 → 2023/07/31	
项目 02	2023/07/21 → 2023/09/30	
项目 03	2023/11/02 → 2024/04/28	
+ New		

但这样一来我们就没办法直接使用 `dateBetween` 这个函数来对它进行计算，所以我们需要执行以下步骤：

使用 `dateStart()` 和 `dateEnd()` 两个函数将「开始时间」和「结束时间」提取出来

项目排期表

Table

Aa Name	📅 项目排期	Σ dateStart()	Σ dateEnd()
项目 01	2023/07/01 → 2023/07/31	July 1, 2023	July 31, 2023
项目 02	2023/07/21 → 2023/09/30	July 21, 2023	September 30, 2023
项目 03	2023/11/02 → 2024/04/28	November 2, 2023	April 28, 2024
+ New			

Notion Formula ?
dateEnd(项目排期)

= April 28, 2024

然后再用 `dateBetween()` 函数对提取出来的时间进行计算

Table

Aa Name	📅 项目排期	Σ dateStart()	Σ dateEnd()	Σ 时间跨度 (方法 1)
项目 01	2023/07/01 → 2023/07/31	July 1, 2023	July 31, 2023	30
项目 02	2023/07/21 → 2023/09/30	July 21, 2023	September 30, 2023	71
项目 03	2023/11/02 → 2024/04/28	November 2, 2023	April 28, 2024	178
+ New				

Notion Formula ?
dateBetween(dateEnd(), dateStart(), "days")

= 178

或者也可以将 `dateStart()` 以及 `dateEnd()` 直接写进 `dateBetween` 函数中：

Table					
Aa Name	📅 项目排期	Σ dateStart()	Σ dateEnd()	Σ 时间跨度 (方法 1)	Σ 时间跨度 (方法 2)
项目 01	2023/07/01 → 2023/07/31	July 1, 2023	July 31, 2023	30	30
项目 02	2023/07/21 → 2023/09/30	July 21, 2023	September 30, 2023	71	71
项目 03	2023/11/02 → 2024/04/28	November 2, 2023	April 28, 2024	178	178
+ New					

Notion Formula ⓘ

`dateBetween(dateEnd(项目排期), dateStart(项目排期), "days")`

= 178

案例 2、客户定期联络表

在一个客户关系维护表中，我们决定为每个客户设置一个最长 3 个月的定期联络间隔，当上一次的联络时间与今天的日期间隔相差 90 天以上，则判定为「需要联络」。

具体操作步骤如下：

- 1. 使用 `now()` 函数获取今天的日期
- 2. 用 `dateBetween` 计算今天与上次联络的时间间隔，单位设置为

`days`

客户定期联络表				
<code>dateBetween(now(), prop("上次联络时间"), "days")</code>				
Table				
Aa 客户姓名	🏢 所属公司	📅 最近联络时间	Σ 时间间隔	+ ...
王小明	百度	2023/12/06	32	
张美丽	阿里巴巴	2023/07/22	169	
李晨阳	腾讯	2024/01/05		
赵小强	美团	2023/10/30		
李婷婷	字节跳动	2023/07/22		
刘明阳	京东	2023/10/15		
宋丽华	滴滴	2023/03/08		
王大明	搜狐	2023/04/01		
王小红	网易	2023/06/17		
李琳琳	新浪	2023/03/20		
+ New				

Notion Formula ⓘ

`dateBetween(now(), 最近联络时间, "days")`

= 169

No results

接下来我们需要寻找那些间隔日期大于 90 天的计算结果，并将其判定为「需要联系」。

这里我们有两种筛选方式，一种是新建一个视图，然后对 Formula 进行筛选，只要筛选出那些间隔大于 90 天的字段即可

需要联系

Σ 时间间隔 ≥ 90 ▾

Aa 客户姓名	📍 所属公司	📅 上次联络时间	Σ 时间间隔	+	...
刘明阳	京东	2022/12/25	120		
李晨阳	腾讯	2023/01/10	104		
李婷婷	字节跳动	2023/01/22	92		
+ New					

另一种则是直接使用前文提到的布尔值来进行判断，我们只需要将

`dateBetween()` 的计算结果与 90 进行大小比较，只要大于 90 即为 True，小于 90 即为 False，如此一来就可以通过查看 Checkbox 的勾选状态快速判断，现在有哪一位客户是需要联络的。

Table					
Aa 客户姓名	📍 所属公司	📅 最近联络时间	Σ 时间间隔	Σ 是否需要联络	+
王小明	百度	2023/12/06	32	☐	
张美丽	阿里巴巴	2023/07/22	169	☒	
李晨阳	腾讯	2024/01/05	2	Notion Formula ⓘ <code>dateBetween(now(), 最近联络时间, "days") > 90</code> = ☒	
赵小强	美团	2023/10/30	69		
李婷婷	字节跳动	2023/07/22	169		

我们还可以应用《Notion Button 详解》一文中学到的内容，为每个客户单独创建一个 Button，点击 Button 即可刷新「上次联络时间」这个字段，这样想必会更方便一些。

三 < > + 二一的笔记 / ... / dateBetween / 客户定期联络表 - 步骤 01

Edited 1m ago Share 消息 时钟 星标 ...

dateBetween(now(), prop("上次联络时间"), "days")

王小明

张美丽

李晨阳

赵小强

李婷婷

+ New

Aa 客户姓名	所属公司	最近联络时间	Σ 时间间隔	Σ 是否需要联络	+ ...
王小明	百度	2023/06/09	43	☐	
张美丽	阿里巴巴	2023/06/24	28	☐	
李晨阳	腾讯	2023/01/10	193	☒	
赵小强	美团	2023/04/02	111	☒	
李婷婷	字节跳动	2023/05/25	58	☐	
刘明阳	京东	2022/12/25	209	☒	
宋丽华	滴滴	2023/03/08	136	☒	
王大明	搜狐	2023/04/01	112	☒	
王小红	网易	2023/06/17	35	☐	
李琳琳	新浪	2023/03/20	124	☒	

+ New

?

案例 3、更准确的倒数日计算

在案例 1 和案例 2

的倒数日计算中，其实我们忽略了一个问题，那就是直接将日期与日期、或者将日期与

`now()` 进行计算，都有可能得出不准确的倒数日。

例如下图，7 号、8 号与 `now()` 计算得出的倒数日都是 0

Table

Aa Name	日期	Σ 倒数	+ ...
	2024/01/07	0	
	2024/01/08	0	

+ New

Notion Formula ?

dateBetween(日期, now(), "days")

= 0

而在下图中，因为「项目 02」的 Date 2 字段包含了具体时间点（5:00 AM），于是 Date 1 和 Date 2 的日期间隔就出现了错误：

Table

Aa Name	Date 1	Date 2	Σ 间隔时间	+ ...
项目 01	2023/07/23	2023/07/22		1
项目 02	2023/07/23	2023/07/22 5:00 AM		0
+ New				

问题的根源出现在 `dateBetween()` 函数的时间单位上。
前面我们对 `dateBetween()` 函数的写法一直都是

`dateBetween(Date 1,Date 2,"days")`，也就是用

`days` 来作为时间间隔的单位，那么当两个日期之间，间隔的小时数不满 24，
`days` 就不会将它算作一天，于是就得出错误的间隔天数。

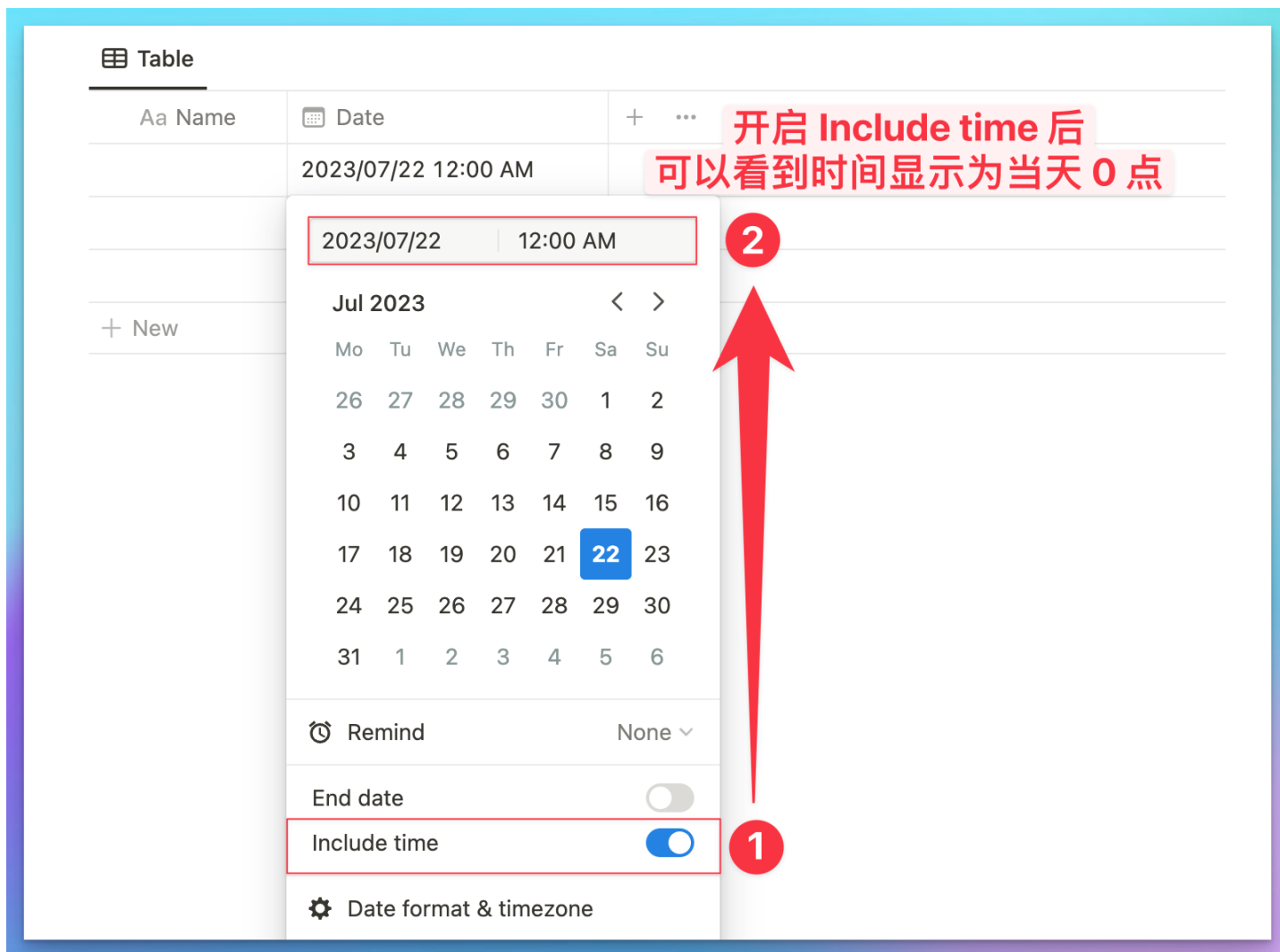
要解决这个问题，我们需要先梳理一下 `now()` 函数和一般 Date 字段的一些特性。

`now()` 这个函数除了会获取日期，还会获取当前的时间点：

Table

Aa Name	Σ now()	+ ...
	July 21, 2023 7:48 PM	
	July 21, 2023 7:48 PM	
	July 21, 2023 7:48 PM	
+ New	now()	Done

Date 这个字段在默认情况下都是将时间设定为当天的零点



所以：

1. 当今天的 `now()` 与明天的 Date 进行计算时，计算出的结果就有可能不满 24 小时
2. 或者当不带时间的 Date 1 与带具体时间的 Date 2 进行计算，也有可能得出不满 24 小时的结果
3. 当我们使用 `"days"` 作为 `dateBetween` 的间隔单位时，在不满 1 天的情况下，它就会返回 0 的值

找到原因之后，我们继续往下分析。

既然 `now()` 返回的日期会带上时间，并且 Date 的默认时间点都是

`12:00 AM`，那如果我们把 `now()`

或者 Date 字段的时间都设置成

`12:00 AM`

这个默认值的话，就可以正确地与任意的 Date 进行计算了。

为了达到这个目的，我们还需要继续学习函数 `hour()` 与

`dateSubtract()` 的用法。

`hour()` 函数

我们可以利用 `hour()`

从某个日期中截取出它的时间是处在第几个小时，如下图所示：

Table

Aa Name	Date	Σ hour()	+ ...
	2023/07/22 12:00 AM	0	
	2023/07/23 5:00 PM	17	
	2023/07/28 7:00 AM	7	
+ New		hour(prop("Date"))	

同一批公式还有

`day()`、`month()`、`year()`、`minute()`，分别可以将一周的第几天，一年的第几个月，日期所在的年份，或者时间的第几分钟计算出来。

dateSubtract() 函数

这个函数的作用是为某个时间减去你所指定的时间长度，它的写法是

`dateSubtract(日期字段, 数字, 单位)`，括号内的参数按照顺序解读即是：

- 1. 读取某个确定的日期
- 2. 设置要减去的数字
- 3. 设置数字所对应的时间单位

以下图为例，我对 Date 这个字段设置的函数效果是「减去 3 个月」，所以：

- 1. 2023/07/22 变成了 2023/04/22
- 2. 2023/03/31 变成了 2022/12/31

dateSubtract 函数

Aa Name	Date	Σ Formula	+ ...
	2023/07/22 5:00 AM	2023/04/22 5:00 AM	
	2023/03/31 12:00 AM	2022/12/31 12:00 AM	
+ New		dateSubtract(prop("Date"), 3, "months")	

接下来让我们耐住性子，再来回顾一下最初的需求：

- 1. 我们的最终目的是要将 `now()` 或者 Date 所获得的时间设置成默认的 `12:00 AM`
- 2. 假设 `now()` 的时间是 `16:30 PM`，我们就要减去 `16:30 PM` 才能获得 `12:00 AM`
- 3. 所以我们需要先用 `hour()` 和 `minute()` 这两个函数分别获得 `now()` 的小时和分钟
- 4. 然后再用 `dateSubtract()` 这个函数减去

`now()` 的小时和分钟

具体流程写法如下

使用 `dateSubtract()` 减去 `now()` 的小时

```
dateSubtract(now(), hour(now()), "hours")
```

在步骤 1 的基础上，继续使用 `dateSubtract()` 减去

`now()` 的分钟

```
dateSubtract(dateSubtract(now(), hour(now()), "hours"), minute(now()), "minutes")
```

在步骤 2 的基础上，使用 `dateBetween()`

计算正确的日期间隔（这里假设 Date 没有带上具体时间）

```
dateBetween(prop("Date"), dateSubtract(dateSubtract(now(), hour(now()), "hours"), minute(now()), "minutes"), "days")
```

最终完整流程如下

The screenshot shows a Notion table with columns for Name, Date, now(), hour(now()), minute(now()), and calculated date differences. A formula editor is open on the right, showing the formula: `dateBetween(prop("Date"), dateSubtract(dateSubtract(now(), hour(now()), "hours"), minute(now()), "minutes"), "days")`. The editor also displays the properties of the `Name` property.

Aa Name	Date	Σ now()	Σ hour(now())	Σ minute(now())	Σ 减去小时	Σ 减去分钟	Σ 计算 Date 和 now() 的正确间隔
	July 23, 2023	July 22, 2023 12:03 PM	12	3	July 22, 2023 12:03 AM	July 22, 2023 12:00 AM	1
	July 24, 2023	July 22, 2023 12:03 PM	12	3	July 22, 2023 12:03 AM	July 22, 2023 12:00 AM	2
	July 25, 2023	July 22, 2023 12:03 PM	12	3	July 22, 2023 12:03 AM	July 22, 2023 12:00 AM	3

通过这个案例想必大家应该已经能够感受到，如果你有意学习 Notion Formula 的用法，那么必然会在这个过程中遭遇非常多的坎坷和挫折，我本人作为 Formula 的新手实在感触颇深，很多时候为了解决一个小小的问题，背后却需要引出一连串更加复杂的问题，所以还请做好必要的心理准备。

> 本文由简悦 SimpRead 转码