Excel 数据分析 Cookbook

♥ Made by Nanmiao Ling

1. 基本操作

调整行高、列宽

- 拖动行号或列标之间的边界进行调整。
- 双击边界，可根据内容自动调整行高或列宽。
- 精确设置：选中行或列 → `开始` → `格式` → `行高` / `列宽`。

冻结窗格

- 冻结首行：`视图` → `冻结窗格` → `冻结首行`。
- 冻结首列：`视图` → `冻结窗格` → `冻结首列`。
- 同时冻结多行、多列：选中固定区域右下方的单元格，再选择 `冻结窗格`。
- 例如要冻结第 1 行和 A 列，应先选中 `B2`。

常用快捷键

- `Ctrl + 方向键`：跳到连续数据区域的边缘。
- `Ctrl + Shift + 方向键`：快速选择连续数据区域。
- `Ctrl + 1`：打开“设置单元格格式”。

2. 单元格数据格式

2.1. 值类型（Value Type）

单元格数据底层使用的 Type。不是显示格式。

类型	存储内容	单元格输入示例	运算行为
Number	数值；日期和时间也以数值存储。**精度为15位10进制**。	`123`、`0.5`、`2026-09-03`	可直接进行算术和大小比较；日期输入会转换为日期序列号
Text	字符串	张三、`'123`、`'2026-09-03`	普通文本可直接输入；前导 `'` 强制后续内容按 Text 存储，且不属于实际值
Boolean	逻辑值	`TRUE`、`FALSE`	可直接输入，用于逻辑判断
Error	错误值	`#N/A`、`#VALUE!`	可直接输入，通常会传递到后续计算
Blank	空单元格	不输入内容	不同函数对 Blank 的处理不同

Formula 是计算表达式，不是独立的 Value Type；计算结果仍属于上述类型。

函数参数使用是 Value Type，而不是下面的 Number Format。

2.2. 格式设置（Number Format）

Excel 所设置的格式指的都是 Number Format，仅为**显示格式**。

每一种 Number Format 与一个 Value Type 对应。

- 常规：可能是所有的 Type
- 特殊：Number 或 Text
- 文本：Text
- 自定义：通常是 Number
- **其他：全是 Number**。

你可以看到 Excel 对文本的支持是很少的，主要还是对数据进行处理。

3. 外观

3.1. 条件格式

常见用途

突出重复值、异常金额、指定文本和排名靠前的数据。

- 选中区域 → `开始` → `条件格式`。
- 可使用“突出显示单元格规则”“数据条”“色阶”或自定义公式。

示例：根据处理状态显示不同颜色

- 选中状态列，例如 `A2:A100`。
- 打开 `条件格式` → `突出显示单元格规则` → `文本包含`。
- 分别创建三条规则：
  - 包含“正常”：显示绿色。
  - 包含“关注”：显示黄色。
  - 包含“逾期”：显示红色。

示例：使用公式匹配文本

- 打开 `条件格式` → `新建规则` → `使用公式确定要设置格式的单元格`。
- 输入 `=ISNUMBER(SEARCH("逾期",A2))`，然后设置显示格式。
- `A2` 是所选区域左上角单元格，引用会逐行变化。
- `SEARCH` 不区分大小写；需要区分大小写时使用 `FIND`。

4. 基础公式与函数

输入公式

- 公式以 `=` 开头，例如 `=A1+B1`。
- 四则运算：加 `+`、减 `-`、乘 `*`、除 `/`。
- 使用括号明确计算顺序，例如 `=(A1+B1)*C1`。

4.1. 单元格引用

引用类型	示例	含义
相对引用	`A1`	引用 A1；复制公式时，行和列随位置变化
绝对引用	`$A$1`	固定 A 列和第 1 行；复制公式时保持不变
固定列	`$A1`	固定 A 列，行随位置变化
固定行	`A$1`	固定第 1 行，列随位置变化
行引用	`1:1`、`1:3`	引用整行或连续多行
列引用	`A:A`、`A:C`	引用整列或连续多列
区域引用	`A1:C10`	引用由左上角 A1 到右下角 C10 组成的矩形区域
跨表引用	`Sheet2!A1`	引用工作表 Sheet2 中的 A1
跨表区域引用	`Sheet2!A1:C10`	引用工作表 Sheet2 中的 A1:C10

- 工作表名称包含空格或特殊字符时，使用单引号包围名称，例如 `'客户明细 2026'!A1`。
- 编辑公式时选中单元格引用并按 `F4`，可在 `A1`、`$A$1`、`A$1` 和 `$A1` 之间切换。

4.2. 常用函数

函数	定义	语法	示例
`SUM`	对指定数值求和	`SUM(number1, [number2], ...)`	`=SUM(B2:B10)`
`AVERAGE`	计算指定数值的算术平均值	`AVERAGE(number1, [number2], ...)`	`=AVERAGE(B2:B10)`
`MAX`	返回指定数值中的最大值	`MAX(number1, [number2], ...)`	`=MAX(B2:B10)`

函数	定义	语法	示例
`MIN`	返回指定数值中的最小值	`MIN(number1, [number2], ...)`	`=MIN(B2:B10)`
`COUNT`	统计参数中包含 Number 的单元格数量	`COUNT(value1, [value2], ...)`	`=COUNT(B2:B10)`
`COUNTA`	统计参数中非空单元格的数量	`COUNTA(value1, [value2], ...)`	`=COUNTA(B2:B10)`
`IF`	根据逻辑条件返回两个结果之一	`IF(logical_test, value_if_true, [value_if_false])`	`=IF(B2>=60,"合格","不合格")`
`SUMIF`	对满足一个条件的单元格对应数值求和	`SUMIF(range, criteria, [sum_range])`	`=SUMIF(A2:A100,"储蓄",B2:B100)`
`COUNTIF`	统计满足一个条件的单元格数量	`COUNTIF(range, criteria)`	`=COUNTIF(A2:A100,"储蓄")`

4.3. logical_test

`logical_test` 是计算结果为 Boolean `TRUE` 或 `FALSE` 的表达式。

写法	定义	示例
`=`	等于	`A1=100`
`<>`	不等于	`A1<>100`
`>` / `<`	大于 / 小于	`A1>100`
`>=` / `<=`	大于等于 / 小于等于	`A1>=100`
Boolean 单元格引用	直接使用单元格中的 Boolean	`A1`
逻辑函数	组合或反转多个条件	`AND(A1>0,A1<=100)`、`OR(A1="是",B1="是")`、`NOT(A1="停用")`
判断函数	判断值的类型或状态	`ISNUMBER(A1)`、`ISTEXT(A1)`、`ISERROR(A1)`

- Text 常量需要使用双引号，例如 `A1="正常"`。
- 示例：`=IF(B2>=60,"合格","不合格")` 中，`B2>=60` 是 `logical_test`。

4.4. criteria

`criteria` 是 `COUNTIF`、`SUMIF` 等函数用来逐个匹配 `range` 中单元格的条件描述；匹配过程产生 Boolean，但 `criteria` 本身不要求是 Boolean 表达式。

写法	匹配关系	示例
Number	等于该 Number	`32`
Text	等于该 Text	`"储蓄"`
单元格引用	等于该单元格的值	`B5`
比较条件	按比较运算符匹配	`">32"`、`"<=100"`、`"<>0"`
拼接的比较条件	将运算符与单元格值组成条件	`">"&B5`、`"<="&TODAY()`
通配符 `*`	匹配任意数量的字符	`"*储蓄*"` 表示包含“储蓄”
通配符 `?`	匹配任意一个字符	`"储?"`
转义符 `~`	匹配字符 `*`、`?` 或 `~` 本身	`"~*"`

- Text 条件以及带比较符的条件需要放在双引号中；纯 Number 不需要。
- `A1>32` 会先计算成 Boolean，不表示对 `range` 中每个单元格执行 `>32`；应写为 `">32"`。

4.5. 条件汇总函数

函数	定义	语法	示例
`COUNTIF`	统计满足一个 `criteria` 的单元格数量	`COUNTIF(criteria_range, criteria)`	`=COUNTIF(A2:A100,"储蓄")`
`COUNTIFS`	统计同时满足多组 `criteria` 的行数	`COUNTIFS(criteria_range1, criteria1, ...)`	`=COUNTIFS(A2:A100,"储蓄",B2:B100,">=10000")`
`SUMIF`	对满足一个 `criteria` 的单元格对应数值求和。	`SUMIF(criteria_range, criteria, [sum_range])`	`=SUMIF(A2:A100,"储蓄",C2:C100)`
`SUMIFS`	对同时满足多组 `criteria` 的行所对应的数值求和。	`SUMIFS(sum_range, criteria_range1, criteria1, ...)`	`=SUMIFS(C2:C100,A2:A100,"储蓄",B2:B100,">=10000")`
`AVERAGEIF`	计算满足一个 `criteria` 的单元格对应数值的平均值	`AVERAGEIF(criteria_range, criteria, [average_range])`	`=AVERAGEIF(A2:A100,"储蓄",C2:C100)`

函数	定义	语法	示例
`AVERAGEIFS`	计算同时满足多组 `criteria` 的行所对应数值的平均值	`AVERAGEIFS(average_range, criteria_range1, criteria1, ...)`	`=AVERAGEIFS(C2:C100,A2:A100,"储蓄",B2:B100,">=10000")`

- 不带 `S` 的函数只接收一个 `criteria`；带 `S` 的函数可以接收多组 `criteria_range + criteria`，各组条件之间是 `AND` 关系。
- `SUMIF` 的求和区域在最后：`SUMIF(range, criteria, sum_range)`；`SUMIFS` 的求和区域在最前：`SUMIFS(sum_range, ...)`。
- `AVERAGEIF` 与 `AVERAGEIFS` 的参数顺序同样不同。
- `sum_range`、`average_range` 和各个 `criteria_range` 应具有相同的行列尺寸，并按位置一一对应。
- 多个条件需要使用 `OR` 关系时，通常分别计算后相加，例如 `=SUM(COUNTIF(A2:A100,{"储蓄","存款"}))`；如果同一单元格可能同时满足多个条件，需要注意重复计数。

4.6. 数据查找

- `VLOOKUP`
  - 定义：在表格区域的第一列中纵向查找值，返回同一行中指定列的值。
  - 语法：`VLOOKUP(lookup_value, table_array, col_index_num, [range_lookup])`。
  - 示例：`=VLOOKUP(E2,A2:C100,3,FALSE)`。
    - 在 `A2:A100` 中精确查找 `E2`。
    - 找到后返回同一行中 `A2:C100` 的第 3 列，即 C 列的值。
  - 限制：只能在 `table_array` 的第一列中查找，只能返回其右侧列；硬编码的列序号也容易因**表格结构变化**而返回错误列。
  - 使用近似匹配时，第一列通常必须按升序排列；无明确需要时使用 `FALSE`。
- `XLOOKUP`
  - 定义：在一个区域中查找值，并从另一个对应区域返回结果。
  - 语法：`XLOOKUP(lookup_value, lookup_array, return_array, [if_not_found], [match_mode], [search_mode])`。
  - 参数：
    - `lookup_value`：要查找的值。
    - `lookup_array`：执行查找的一行或一列。
    - `return_array`：返回结果的一行或一列，**应与 `lookup_array` 尺寸对应**。
    - `if_not_found`：未找到时返回的自定义内容。
    - `match_mode`：匹配方式；默认 `0` 为精确匹配，`-1` 为精确或下一个较小值，`1` 为精确或下一个较大值，`2` 为通配符匹配。
    - `search_mode`：搜索顺序；默认从第一项向最后一项搜索。
  - 示例：`=XLOOKUP(E2,A2:A100,C2:C100,"未找到")`。
    - 在 `A2:A100` 中精确查找 `E2`。

    - 找到后返回 `C2:C100` 中对应位置的值；否则返回“未找到”。
- `lookup_array` 和 `return_array` 分开指定，因此既可以向右查找，也可以向左查找，不依赖列序号。
- 当前 Excel 版本支持时，通常优先使用 `XLOOKUP`。

4.7. 类型判断

- `ISNUMBER()`
- `ISTEXT()`
- `ISERROR()`
- `ISERR()`
- `ISNA()`
- `ISLOGICAL()`
- `ISBLANCK()`

4.8. 类型转换

修改 Number Format 不会改变 Value Type。需要通过函数、运算或重新输入显式转换。

转换	方法	示例	说明
Text → Number	`VALUE`	`=VALUE(A1)`	按当前区域设置解析数字、日期或时间 Text
Text → Number	`NUMBERVALUE`	`=NUMBERVALUE(A1,".",",")`	显式指定小数点和分组符，适合外部数据
Text → Number	算术运算	`=--A1`、`=A1*1`	简写；无法解析时返回 Error
Text 日期 → Number	`DATEVALUE`	`=DATEVALUE(A1)`	返回日期序列号，不包含时间部分
Text 时间 → Number	`TIMEVALUE`	`=TIMEVALUE(A1)`	返回一天的小数，不包含日期部分
Number → Text	`TEXT`	`=TEXT(A1,"0.00")`	按指定格式生成 Text
Number 或 Text → Text	拼接空文本	`=A1&""`	将值转换为 Text，不保留 Number Format 的显示效果
Boolean → Number	`N` 或算术运算	`=N(A1)`、`=--A1`	`TRUE` 转为 `1`，`FALSE` 转为 `0`
值 → Boolean	比较表达式	`=A1<>0`、`=A1="正常"`	根据比较结果生成 `TRUE` 或 `FALSE`

- `VALUE`、`DATEVALUE` 和 `TIMEVALUE` 的解析受区域设置影响；跨地区导入数字时优先使用 `NUMBERVALUE`。

- `TEXT` 返回 Text，后续计算通常应保留原 Number，并只对最终展示结果使用 `TEXT`。
- 比较 Number `123` 与 Text `123` 时，可先统一为 Text：`=A1&""=B1&""`。
- Number 最多只能保留 15 位数字。

4.9. IF 分支

函数	定义	语法	示例
`IF`	根据一个 `logical_test` 返回两个结果之一	`IF(logical_test, value_if_true, [value_if_false])`	`=IF(A2>=60,"合格","不合格")`
`IFS`	按顺序判断多个条件，返回第一个为 `TRUE` 的分支	`IFS(logical_test1, value_if_true1, ...)`	`=IFS(A2>=90,"优秀",A2>=60,"合格",TRUE,"不合格")`
`SWITCH`	将一个表达式依次与多个值进行 equality matching	`SWITCH(expression, value1, result1, ..., [default])`	`=SWITCH(A2,"A","正常","B","关注","未知")`
`IFERROR`	`value` 为任意 Error 时返回备用值，否则返回原结果	`IFERROR(value, value_if_error)`	`=IFERROR(A2/B2,"计算错误")`
`IFNA`	`value` 为 `#N/A` 时返回备用值，否则返回原结果	`IFNA(value, value_if_na)`	`=IFNA(XLOOKUP(E2,A2:A100,C2:C100),"未找到")`

- `IFS` 最后的 `TRUE` 可作为默认分支；没有条件匹配且没有默认分支时返回 `#N/A`。
- `SWITCH` 适合枚举值，不适合直接处理分数区间等范围条件。
- 查找不到时优先使用 `IFNA`，避免 `IFERROR` 掩盖其他 Error。
- `AND`、`OR`、`NOT` 和 `XOR` 用于生成或组合 `logical_test`，不是分支函数。
- `COUNTIF(S)`、`SUMIF(S)` 和 `AVERAGEIF(S)` 是条件汇总函数，不是分支函数。

4.10. 常用文本方法、数字方法

`IF` 需要 `logical_test`，但比较运算符无法独立处理复杂的匹配逻辑，因此需要使用 Text 和 Number 处理方法生成 Boolean。

Text

方法	定义	生成 `logical_test` 的示例
`=` / `<>`	判断 Text 相等 / 不等；不区分英文大小写	`A1="储蓄"`、`A1<>"停用"`

方法	定义	生成 `logical_test` 的示例
`EXACT`	判断两个 Text 完全相同；区分英文大小写	`EXACT(A1,"ABC")`
`SEARCH`	查找子串并返回起始位置；不区分大小写，支持通配符	`ISNUMBER(SEARCH("储蓄",A1))`
`FIND`	查找子串并返回起始位置；区分大小写，不支持通配符	`ISNUMBER(FIND("VIP",A1))`
`LEFT`	从左侧提取指定数量的字符	`LEFT(A1,2)="个人"`
`RIGHT`	从右侧提取指定数量的字符	`RIGHT(A1,2)="账户"`
`MID`	从指定位置提取指定数量的字符	`MID(A1,3,2)="储蓄"`
`LEN`	返回字符数量	`LEN(A1)=18`
`TRIM`	删除多余的普通空格	`TRIM(A1)="正常"`
`SUBSTITUTE`	将指定子串替换为其他 Text	`SUBSTITUTE(A1," ","")=B1`

- `SEARCH` 和 `FIND` 找不到内容时返回 `#VALUE!`，因此通常使用 `ISNUMBER(...)` 将“找到的位置”转换为 Boolean。
- 判断不包含：`=NOT(ISNUMBER(SEARCH("储蓄",A1)))`。
- 判断包含多个关键字中的任意一个：`=OR(ISNUMBER(SEARCH("储蓄",A1)),ISNUMBER(SEARCH("存款",A1)))`。
- 判断同时包含多个关键字：`=AND(ISNUMBER(SEARCH("个人",A1)),ISNUMBER(SEARCH("储蓄",A1)))`。

Number

方法	定义	生成 `logical_test` 的示例
比较运算符	判断相等、不等或大小关系	`A1>=100`、`A1<>0`
`AND`	判断 Number 是否同时满足多个条件	`AND(A1>=60,A1<=100)`
`OR`	判断 Number 是否满足任一条件	`OR(A1<0,A1>100)`
`ROUND`	四舍五入到指定小数位	`ROUND(A1,2)=ROUND(B1,2)`
`ROUNDUP`	向远离 `0` 的方向舍入	`ROUNDUP(A1,0)>=100`
`ROUNDDOWN`	向靠近 `0` 的方向舍入	`ROUNDDOWN(A1,0)=100`
`INT`	向下舍入到整数	`A1=INT(A1)`
`MOD`	返回除法余数	`MOD(A1,2)=0`
`ABS`	返回绝对值	`ABS(A1-B1)<0.001`

- 判断闭区间：`=AND(A1>=下限,A1<=上限)`。
- 判断整数：`=MOD(A1,1)=0`；对于负数，使用 `A1=INT(A1)` 也可以判断是否为整数。

- 判断 `n` 的倍数：`=MOD(A1,n)=0`。
- 浮点计算可能产生微小误差；需要判断近似相等时，可使用 `=ABS(A1-B1)<容差`，不要直接使用 `A1=B1`。

5. 数据整理

排序

- 选中数据区域 → `数据` → `排序`。
- 可按文本、数字、日期、单元格颜色进行单级或多级排序。
- 排序前确认整张表被选中，避免单列移动后造成数据错位。

筛选

- 选中标题行 → `数据` → `筛选`，或按 `Ctrl + Shift + L`。
- 可按值、文本条件、数字条件、日期或颜色筛选。
- 筛选只隐藏不符合条件的行，不会删除数据。

查找与替换

- `Ctrl + F`：查找；`Ctrl + H`：查找并替换。
- 批量替换前检查查找范围以及“区分大小写”“单元格匹配”等选项。

删除重复值

- 选中数据 → `数据` → `删除重复项` → 选择用于判断重复的列。
- 操作会直接删除重复记录，建议先复制备份或确认选择范围。

分列

- 选中一列 → `数据` → `分列`。
- 按分隔符拆分：适合逗号、空格、制表符等规则数据。
- 按固定宽度拆分：适合各字段位置固定的数据。
- 设置目标区域，避免覆盖右侧已有内容。

数据验证与下拉列表

- 选中区域 → `数据` → `数据验证`。
- 创建下拉列表：`允许` 选择“序列”，再指定选项区域或输入逗号分隔的选项。
- 可同时设置输入提示和无效数据的错误警告。

6. 简单数据分析

创建表格

- 选中连续数据区域 → `插入` → `表格`，或按 `Ctrl + T`。
- 确认“表包含标题”，即可获得筛选、格式和自动扩展功能。
- 新增行或列时，表格公式和格式通常会自动填充。

基础图表

- 选中包含标题的数据 → `插入` → 选择图表类型。
- 柱形图：比较不同类别；折线图：观察时间趋势；饼图：展示少量类别的占比。
- 图表应设置清晰的标题、坐标轴和单位，避免使用与分析无关的装饰。

数据透视表

- 选中规范的明细数据 → `插入` → `数据透视表`。
- 将分类字段拖入“行”或“列”，将统计字段拖入“值”，将限制条件拖入“筛选”。
- 在“值字段设置”中切换求和、计数、平均值等汇总方式。
- 源数据变化后，右键数据透视表并选择“刷新”。
- 源数据应具有唯一标题，避免空白列、合并单元格和混合数据类型。

7. 数据透视表

数据透视表用于将明细表按维度分组并汇总。它不会修改源数据，而是根据字段布局生成可交互的汇总视图。

7.1. 准备源数据

推荐结构

- 第一行只放字段标题，每列标题应唯一且非空。
- 一行表示一条完整记录，一列只保存一种含义和一种 Value Type。
- 删除明细区域中的合并单元格、空白行、空白列和手工小计。
- 日期列应保存为真正的 Date，金额列应保存为 Number，不要把单位写入单元格值。
- 建议先将数据转换为 Excel Table：选中数据 → `Ctrl + T` → 确认“表包含标题”。新增记录后，Table 会自动扩展数据透视表的数据源范围。

示例源数据

交易日期	网点	客户类型	业务类型	柜员	笔数	金额
2026-09-01	A 网点	个人	存款	张三	1	10000
2026-09-01	B 网点	对公	转账	李四	1	50000

7.2. 新建数据透视表

从区域或 Table 创建

- 单击源数据中的任意单元格。
- 选择 `插入` → `数据透视表` → `从表格/区域`。
- 检查“表格/区域”是否包含全部字段和记录。
- 选择放置位置：通常使用“新工作表”；需要与其他分析并排时指定“现有工作表”的空白起始单元格。
- 单击“确定”，然后在`数据透视表分析` -> `显示` → `字段列表` 窗格中配置字段。

字段区域

区域	作用	示例
筛选	对整张数据透视表设置页面级条件	网点
列	横向展开分类	月份
行	纵向展开分类或层级	客户类型、业务类型
值	对字段执行汇总计算	金额求和、业务类型计数

- 同一个字段可以多次拖入“值”区域。例如，第一个“金额”显示求和，第二个“金额”显示占总计百分比。
- “行”和“列”区域中多个字段的上下顺序决定分组层级；可直接拖动字段调整顺序。
- 文本字段拖入“值”区域时通常执行“计数”；Number 字段通常执行“求和”。应检查实际汇总方式，不要只依赖默认值。

7.3. 配置值字段

更改汇总方式

- 在“值”区域单击字段 → `值字段设置`。
- 选择“求和”“计数”“平均值”“最大值”“最小值”等汇总方式。
- “计数”统计非空记录；需要统计业务笔数时，应确认一行是否确实代表一笔业务。
- 若 Number 字段只能使用“计数”，通常表示源列中混有 Text、Error 或空字符串。先修正源数据类型，再刷新数据透视表。

设置显示方式

- 打开 `值字段设置` → `值显示方式`。
- 常用方式：
  - `% 总计`：当前值占全部值的比例。
  - `% 行汇总`：当前值占所在行合计的比例。
  - `% 列汇总`：当前值占所在列合计的比例。
  - `差异` / `% 差异`：与指定基准项比较。
  - `按某一字段汇总`：生成累计值。
- “汇总值方式”决定原始值如何聚合；“值显示方式”决定聚合结果如何再次表示，两者可以组合使用。

设置数字格式

- 在 `值字段设置` 中选择 `数字格式`，设置金额、百分比或小数位数。

- 不要只对数据透视表单元格使用普通格式刷；刷新或布局变化后，普通单元格格式可能不能覆盖新增区域。

7.4. 分组、筛选和切片器

日期分组

- 在数据透视表中右键单个日期 → `组合`。
- 可选择“年”“季度”“月”“日”等层级。
- 出现“无法组合所选内容”时，检查源日期列是否包含 Text、Blank 或 Error。

数值分组

- 右键行标签中的任意数值 → `组合`。
- 设置起始值、终止值和步长，例如每 `10000` 元划分一个金额区间。
- 分组区间只影响数据透视表的展示，不会更改源金额。

筛选器和切片器

- 字段筛选：使用行标签、列标签或“筛选”区域中的下拉菜单。
- 切片器：选中数据透视表 → `数据透视表分析` → `插入切片器`，再选择网点、客户类型等字段。
- 时间线：日期字段可使用 `插入时间线`，按年、季度、月或日筛选。
- 一个切片器需要控制多张数据透视表时，打开切片器的“报表连接”，勾选共享同一数据源缓存的目标数据透视表。

7.5. 计算字段与复杂计算

数据透视表的普通“值”区域先对源记录进行汇总。需要新增计算结果时，应先判断计算发生在**每一行明细**还是**汇总结果**上。

需求	推荐位置	原因
每条明细先判断、清洗或计算，再参与汇总	源 Table 公式列或 Power Query Custom Column	计算可以逐行访问同一条记录中的字段
普通数据透视表中用已有数值字段组成简单公式	计算字段	公式作用于字段汇总，适合简单比例或差额
按筛选上下文计算去重客户数、复杂占比或时间分析	Data Model measure（DAX）	measure 按当前行、列、筛选器的 filter context 求值
只改变汇总结果的展示	值显示方式	无需新增源字段或公式

新建普通计算字段

- 选中数据透视表 → `数据透视表分析` → `字段、项目和集` → `计算字段`。
- 输入名称，例如“平均单笔金额”。
- 在“公式”中使用字段名，例如 `=金额/笔数`，然后选择“添加”。
- 计算字段引用的是源字段，不是工作表单元格地址，也不能直接引用数据透视表中的某个结果单元格。
- 计算字段通常按汇总后的字段值计算。例如 `=金额/笔数` 的结果相当于 `SUM(金额)/SUM(笔数)`，不等同于先计算每行 `金额/笔数` 再求平均。

复杂逐行计算：先添加源 Table 公式列

- 在源 Table 右侧添加“金额等级”列，并在第一条记录输入：
  - `=IFS([@金额]>=100000,"大额",[@金额]>=10000,"中额",TRUE,"小额")`
- Table 会将公式应用到整列，每一行中的 `[@金额]` 只引用当前记录的金额。
- 再刷新数据透视表，将“金额等级”拖入“行”“列”或“筛选”区域。
- 需要组合多个字段时，可使用：
  - `=IF(AND([@客户类型]="个人",[@金额]>=50000),"个人大额","其他")`
- 逐行逻辑较长、需要清洗多个来源或需要重复刷新时，优先在 Power Query 中创建 Custom Column。

Data Model measure

- 创建数据透视表时勾选“将此数据添加到数据模型”。
- 在 Power Pivot 或数据透视表的 measure 功能中创建 measure，例如：
  - `总金额 := SUM(交易明细[金额])`
  - `总笔数 := SUM(交易明细[笔数])`
  - `平均单笔金额 := DIVIDE([总金额],[总笔数])`
- measure 不写入每个源单元格，而是在数据透视表的当前 filter context 中动态计算；切片器或字段布局变化时，结果会重新计算。
- 普通计算字段与 Data Model measure 属于不同机制。工作簿未使用 Data Model 时，不要把 DAX measure 当作普通计算字段输入。

7.6. 刷新与维护

刷新

- 右键数据透视表 → `刷新`，只刷新当前数据透视表。
- 选择 `数据` → `全部刷新`，刷新工作簿中的查询、连接和相关数据透视表。
- 若源是普通区域，新增行可能不在原范围内；改用 Excel Table 或手动执行 `数据透视表分析` → `更改数据源`。

常见问题

现象	检查项
新增记录没有出现	数据源是否为 Table、是否已刷新
金额被计数而不是求和	源列是否混有 Text、Blank 或 Error
日期无法分组	日期是否为真正的 Date，是否含空值或错误值
刷新后格式变化	是否通过“值字段设置”配置数字格式
计算结果与逐行公式不同	计算字段是否在汇总后才执行；需要时改用源公式列或 Power Query

8. Power Query

Power Query 用于连接、清洗、转换和合并数据。每次刷新都会按 Applied Steps 的顺序重新执行同一套规则，适合替代重复的复制、分列、替换和公式填充操作。

8.1. 新建查询

从当前工作簿的 Table 导入

- 先将源区域转换为 Excel Table：选中数据 → `Ctrl + T`。
- 单击 Table 中任意单元格 → `数据` → `从表格/区域`。
- Excel 打开 Power Query Editor，并创建一个引用该 Table 的查询。

从文件导入

- 选择 `数据` → `获取数据` → `从文件`，再选择 Excel Workbook、Text/CSV、PDF 或 Folder 等来源。
- 在 Navigator 中预览对象，选择需要的 Sheet、Table 或文件。
- 选择“转换数据”进入 Power Query Editor；只有确认无需转换时才直接选择“加载”。

Query 与 Applied Steps

打开 `查询` → `编辑`，弹出此查询表的 power query 窗口。

- 左侧“查询”窗格保存不同 Query；中间显示当前步骤的结果预览；右侧“查询设置”显示名称和 Applied Steps。
- 每个 Applied Step 接收上一步的结果并返回新的值。删除或移动中间步骤可能使后续步骤引用失效。
- 双击步骤名称可改成“筛选有效交易”“添加金额等级”等含义明确的名称。
- Power Query 只显示预览数据；转换规则会在刷新时应用到完整数据源。

8.2. 数据类型与基础转换

设置数据类型

- 单击列标题左侧的类型图标，选择 Text、Whole Number、Decimal Number、Date、Date/Time 或 Logical 等类型。
- 设置类型会新增 `Changed Type` 步骤，而不只是改变显示格式。
- 身份证号、账号和带前导零的代码应使用 Text；金额使用 Decimal Number 或 Fixed Decimal Number；业务日期使用 Date。
- 转换失败的值会成为 Error。应检查 Error 的原因，而不是直接删除全部错误行。

常用转换

- `选择列` / `删除列`：只保留分析需要的字段。
- `筛选行`：按值、Text、Number 或 Date 条件筛选。
- `替换值`：统一状态、代码或缺失标记。
- `拆分列`：按分隔符、字符数或位置拆分 Text。
- `分组依据`：按一个或多个字段执行求和、计数、最大值等聚合。
- `透视列`：把某列的不同值展开成多个字段。
- `逆透视列`：把多个同类字段整理成“属性—值”的纵向明细结构。

8.3. 合并、追加与加载

合并查询（Merge）

- `主页` → `合并查询`，选择主表、匹配表和两侧用于匹配的列。
- 常用 Left Outer Join：保留主表全部行，只带回匹配表中找到的记录。
- 多列匹配时，按住 `Ctrl` 依次选择两张表的对应列，并保持选择顺序一致。
- 合并后生成嵌套 Table 列；单击列标题右侧的展开按钮，选择需要带回的字段。
- 匹配字段的 Value Type 必须一致；Number `123` 与 Text `"123"` 不会可靠匹配。

追加查询（Append）

- `主页` → `追加查询`，将结构相同的多张明细表纵向堆叠。
- Power Query 按列名对齐字段，而不是按列所在位置对齐；名称不一致会生成不同列并产生 `null`。
- 来源字段不完全一致时，先统一列名与类型，再追加。

关闭并加载

- `主页` → `关闭并加载到...`。
- 可加载为工作表 Table、仅创建连接，或添加到 Data Model。
- 中间 Query 只为其他 Query 提供数据时，选择“仅创建连接”，避免生成不需要的工作表。

8.4. Custom Column：对每一行应用公式

选择 `添加列` → `自定义列`，输入 M expression。Custom Column 会针对 Table 的每一行执行一次表达式；当前行是一条 Record，通过 `[列名]` 访问其中的字段。

基本写法

- 新列“手续费”：`[金额] * [费率]`
- 新列“客户业务”：`[客户类型] & "-" & [业务类型]`
- 新列“月份”：`Date.Month([交易日期])`

- 新列“是否大额”：`[金额] >= 50000`
- Custom Column 对话框中不输入 Excel 公式的开头 `=`，也不能使用 `A2`、`B2` 等单元格地址。
- `[金额]` 表示当前行 Record 中名为“金额”的字段；它不是整列引用。

`each` 的含义

- `each` 是单参数 function 的简写，参数默认名为 `_`。
- `each [金额] >= 50000` 等价于 `(_) => _[金额] >= 50000`。
- Custom Column 对话框通常只填写 `[金额] >= 50000`；Power Query 生成的完整步骤会使用 `each`：

```
Table.AddColumn(
    #"Changed Type",
    "是否大额",
    each [金额] >= 50000,
    type logical
)
```

- `Table.AddColumn` 的第 3 个参数是逐行执行的 function，第 4 个参数指定新列类型。

8.5. 条件、Text、Number 和 Date 组合

`if ... then ... else`

- M 的条件表达式必须同时包含 `if`、`then` 和 `else`。
- 多条件分类：

```
if [金额] >= 100000 then "大额"
else if [金额] >= 10000 then "中额"
else "小额"
```

- 组合条件使用 `and`、`or` 和 `not`：

```
if [客户类型] = "个人" and [金额] >= 50000 then "个人大额"
else if [客户类型] = "对公" or [业务类型] = "转账" then "重点复核"
else "普通"
```

常用逐行方法

类型	M expression 示例	结果
Text 清理	`Text.Trim([客户名称])`	删除首尾空白
Text 包含	`Text.Contains([摘要], "转账", Comparer.OrdinalIgnoreCase)`	返回 Logical，忽略英文大小写
Text 拼接	`Text.Combine({[网点], [柜员]}, "-")`	合并字段并添加分隔符

类型	M expression 示例	结果
Number 取绝对值	`Number.Abs([账面金额] - [核对金额])`	返回差额的绝对值
Number 舍入	`Number.Round([金额] * [费率], 2)`	四舍五入到 2 位小数
Date 年月	`Date.ToText([交易日期], "yyyy-MM")`	返回年月 Text
Date 间隔	`Duration.Days(Date.From(DateTime.LocalNow()) - [开户日期])`	返回相隔天数
Null 判断	`if [金额] = null then 0 else [金额]`	用 `0` 替代 null

- M 区分大小写：`Text.Trim`、`Date.Year`、`Number.Round` 的拼写和大小写必须正确。
- `null` 表示缺失值，不等同于空 Text `""`，也不等同于 Number `0`。
- 对可能为 `null` 的 Text 直接调用 `Text.Trim` 可能报错；应先判断或使用错误处理。

8.6. 使用 `let ... in` 编写复杂单元格逻辑

`let` 用于把复杂表达式拆分为多个有名称的中间结果，`in` 指定最终返回值。Custom Column 每处理一行，都会使用当前行字段重新计算这些中间结果。

示例：计算并分类手续费

```
let
    Amount = if [金额] = null then 0 else [金额],
    Rate = if [客户类型] = "VIP" then 0.0005 else 0.001,
    RawFee = Amount * Rate,
    RoundedFee = Number.Round(RawFee, 2),
    FinalFee = if RoundedFee < 2 then 2 else if RoundedFee > 50 then 50 else RoundedFee
in
    FinalFee
```

- `Amount`：先处理缺失金额。
- `Rate`：根据当前行客户类型选择费率。
- `RawFee`：计算未舍入手续费。
- `RoundedFee`：保留 2 位小数。
- `FinalFee`：使用最低 `2` 元、最高 `50` 元的边界。
- 中间变量只在本次表达式中有效，不会成为 Table 中的新列。

示例：多个字段生成复核标记

```
let
    CleanSummary = if [摘要] = null then "" else Text.Trim([摘要]),
    ContainsTransfer = Text.Contains(CleanSummary, "转账", Comparer.OrdinalIgnoreCase),
    IsLarge = [金额] <> null and [金额] >= 50000,
    MissingId = [证件号码] = null or Text.Trim([证件号码]) = ""
```

```
in
    if MissingId then "证件缺失"
    else if ContainsTransfer and IsLarge then "大额转账复核"
    else "正常"
```

8.7. Error 处理与调试

`try ... otherwise`

- `try expression otherwise fallback` 在 expression 出错时返回备用值。
- 安全转换金额：`try Number.FromText([金额文本]) otherwise null`。
- 安全计算：`try [金额] / [笔数] otherwise null`。
- 不要无差别地用 `otherwise 0` 隐藏数据质量问题；`0` 可能被误认为真实业务值。无法计算时通常返回 `null`，并另建错误标记列。

保留错误信息

```
let
    Attempt = try Number.FromText([金额文本])
in
    if Attempt[HasError]
    then "金额格式错误"
    else "正常"
```

- `try` 返回一条 Record，可检查 `[HasError]`；成功时可读取 `[Value]`，失败时可读取 `[Error]`。
- 调试复杂 Custom Column 时，先把长表达式拆成多个简单列，检查每一步的类型和值，再合并为 `let ... in`。

8.8. 新建可复用的自定义 M function

当同一套复杂规则需要用于多个 Query 或多列时，可建立独立 function，避免复制长表达式。

创建 function

- 在 Power Query Editor 中选择 `主页` → `新建源` → `其他源` → `空白查询`。
- 打开 `主页` → `高级编辑器`，输入：

```
(amount as nullable number, customerType as nullable text) as nullable number =>
let
    SafeAmount = if amount = null then 0 else amount,
    Rate = if customerType = "VIP" then 0.0005 else 0.001,
    Fee = Number.Round(SafeAmount * Rate, 2),
    FinalFee = if Fee < 2 then 2 else if Fee > 50 then 50 else Fee
in
    FinalFee
```

- 将 Query 重命名为 `fnCalculateFee`。以 `fn` 开头只是常见命名方式，不是 M 的语法要求。
- 参数上的 `nullable number`、`nullable text` 表示允许传入 `null`；返回类型也声明为 `nullable number`。

对每一行调用 function

- 回到明细 Query → `添加列` → `自定义列`。
- 输入：`fnCalculateFee([金额], [客户类型])`。
- Power Query 会把当前行的“金额”和“客户类型”传入 function，并为每行生成结果。
- 完整步骤类似：

```
Table.AddColumn(
    #"Changed Type",
    "手续费",
    each fnCalculateFee([金额], [客户类型]),
    type number
)
```

接收当前行 Record 的 function

需要读取很多字段时，可以把整条当前行 Record 传给 function：

```
(row as record) as text =>
let
    Amount = Record.FieldOrDefault(row, "金额", null),
    CustomerType = Record.FieldOrDefault(row, "客户类型", null),
    Result =
        if Amount = null then "金额缺失"
        else if CustomerType = "个人" and Amount >= 50000 then "个人大额"
        else "普通"
in
    Result
```

- 在 Custom Column 中输入 `fnClassifyRecord(_)`，其中 `_` 表示当前行 Record。
- `Record.FieldOrDefault` 可在字段不存在时返回默认值，适合不同来源的字段可能不完全一致的情况。
- 如果字段结构固定，直接使用 `[金额]` 等字段访问方式更简洁，也更容易发现字段名错误。

8.9. Advanced Editor 与步骤引用

完整 Query 通常由一个 `let ... in` expression 组成：

```
let
    Source = Excel.CurrentWorkbook(){[Name="交易明细"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(
        Source,
        {{"交易日期", type date}, {"金额", type number}, {"客户类型", type text}}
    ),
    #"Added Fee" = Table.AddColumn(
```

```
        #"Changed Type",
        "手续费",
        each fnCalculateFee([金额], [客户类型]),
        type number
    )
in
    #"Added Fee"
```

- `Source`、`#"Changed Type"`、`#"Added Fee"` 是依次执行的步骤名称。
- 名称含空格或特殊字符时使用 `#"..."`。
- 每个步骤通常引用前一个步骤；`in` 后面的名称决定 Query 最终输出哪一步。
- 步骤之间使用逗号分隔，最后一个步骤后不加逗号。

8.10. 刷新与数据透视表联动

- 源文件或源 Table 更新后，选择 `数据` → `全部刷新`，Power Query 会重新执行全部步骤。
- Power Query 结果加载为 Table 后，可以基于该 Table 创建数据透视表。
- 推荐的数据流：原始明细 → Power Query 清洗与逐行计算 → 加载到 Table 或 Data Model → 数据透视表汇总分析。
- 修改 Query 不会自动改写原始数据；输出会在目标 Table 或 Data Model 中重新生成。
- 移动、重命名或删除外部源文件可能导致刷新失败。使用 Folder 数据源时，应保持文件结构、列名和类型一致。
- 若刷新顺序导致数据透视表仍显示旧结果，使用 `数据` → `全部刷新`，并确认 Query 已完成加载后再刷新数据透视表。