Power Query (M): Split space-separated text into multiple columns on multiple rows

Owen Price
flexyourdata.com, @flexyourdata

Split the text into rows and columns as shown

Some points to bear in mind:

- Pay attention to the output dates
- The formatting you see in the output in the image is cell formatting:
  - Dates should be output as dates and not as text
  - When parsing dates, use explicit culture settings
  - Output the amount as a decimal number

	A	B	C	D	E	F	G	H	I
1									
2		From this:							
3		APS Deposit 04/01/2022 $5,174.27 APS ACH Deposit 04/04/2022 $65,186.66							
4									
5		To this:							
6		APS Deposit	2022-01-04	$ 5,174.27					
7		APS ACH Deposit	2022-04-04	$ 65,186.66					
8									

Note that the output date is January 4th, not April 1st

Owen Price
flexyourdata.com, @flexyourdata

There were many great answers on the post. To read more, go to my profile and look at the recent posts

Or go here

Owen Price
flexyourdata.com, @flexyourdata

These slides describe the approach I took

First, some notes:

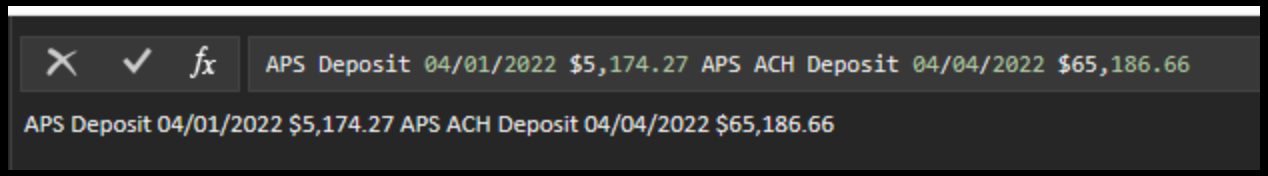
We need some way of finding where one row ends and another begins. In this case, we know that each row ends with a number and each row begins with text.

	A	B	C	D	E	F	G	H	I
1									
2		From this:							
3		APS Deposit	04/01/2022	$5,174.27	APS ACH Deposit	04/04/2022	$65,186.66		
4									
5		To this:							
6		APS Deposit	2022-01-04	$ 5,174.27					
7		APS ACH Deposit	2022-04-04	$ 65,186.66					
8									

We need some way to separate the columns. There are spaces between each column, but there are also spaces between the text values in the description, so care is needed.

For convenience, let's start by setting the Source to the test string

```
let
    Source = "APS Deposit 04/01/2022 $5,174.27 APS ACH Deposit
04/04/2022 $65,186.66"
in
    Source
```



Next, split the string using space as a delimiter

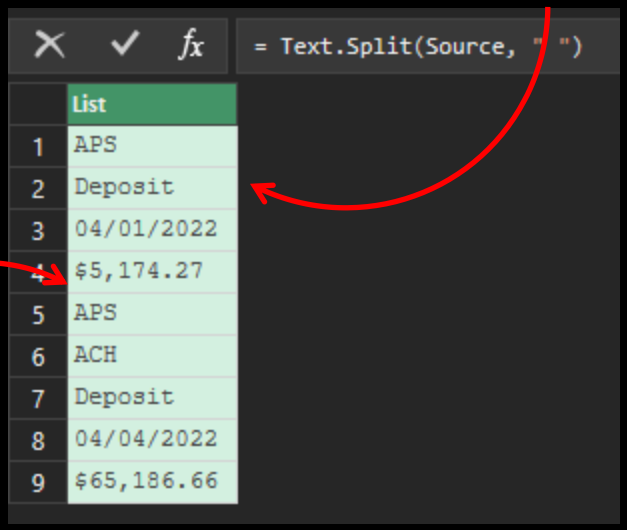
```
let
    Source = "APS Deposit 04/01/2022 $5,174.27 APS ACH Deposit
04/04/2022 $65,186.66",
    Split = Text.Split(Source, " ")
in
    Split
```

= Text.Split(Source, " ")

	List
1	APS
2	Deposit
3	04/01/2022
4	$5,174.27
5	APS
6	ACH
7	Deposit
8	04/04/2022
9	$65,186.66

Now we need to demarcate the list elements

If a text element is preceded by a text element, it's part of the same column (the description)



Rows end after a number

The numeric and date-like elements belong in columns of their own

Owen Price
flexyourdata.com, @flexyourdata

To make this easier, let's ascribe types to each list element

```
let
    …
    Split = Text.Split(Source, " "),
    Typed = List.Transform(Split,
        each try Number.From(_) otherwise
             try Date.From(_, "en-gb") otherwise _ )
in
    Typed
```

For each list element, try converting it to a number. If that fails, try converting it to a date, if that fails, do nothing

Explicitly stating the culture (en-gb uses dd/mm/yyyy) ensures the date is parsed correctly

= List.Transfo

	List
1	APS
2	Deposit
3	2022-01-04
4	5174.27
5	APS
6	ACH
7	Deposit
8	2022-04-04
9	65186.66

The alignment shows us we've properly ascribed the date and numeric types

Next, we'll iterate through the list with List.Accumulate (1/4)

```
let
    …
    Typed = List.Transform(…)
    Accumulate = List.Accumulate({0..List.Count(Typed)-1}, "",
        (a, b) =>
            let
                c = Typed{b},
                p = try Typed{b-1} otherwise "",
                d = if p = "" then ""
                    else
                        if c is text then
                            if p is text then " " else ";"
                        else ","
            in
                a & d & Text.From(c)
    )
in
    Accumulate
```

Next, we'll iterate through the list with List.Accumulate (2/4)

Iterate through the list element indices (from 0 to the count of elements minus 1)

Start with an empty string

```
…
Accumulate = List.Accumulate({0..List.Count(Typed)-1}, "",
    (a, b) =>
        let
            c = Typed{b},
            p = try Typed{b-1} otherwise "",
    …
),
```

In this function, a is the accumulated value (a text string) and b is the element of the list being iterated over (the indices). So, b is a number and is passed into the list index of the Typed step to get the current list element, which is named c

We get the prior list element by passing b-1. This expression returns an error for b=0, so we use a default empty string to indicate "first element"

Now we build the delimiter d (3/4)

If the previous value p is an empty string, then this is the first list element, so we don't need a delimiter...

```
…
            p = try Typed{b-1} otherwise "",
            d = if p = "" then ""
                else
                    if c is text then
                        if p is text then " " else ";"
                    else ","
    …
),
```

...otherwise, if the current value c is text, then, if the previous value is also text, then use a space (because they're both part of the description).

Otherwise, the previous value is not text, so it's a number and it's the end of the row, so use a semi-colon.

Otherwise, use a comma to separate the columns.

And finally combine the accumulator with the delimiter and the current value (4/4)

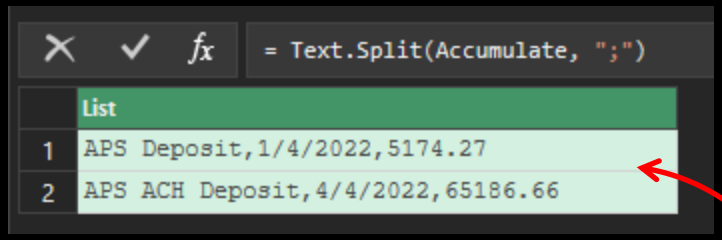
```
    …
            in
                a & d & Text.From(c)
    )
in
    Accumulate
```

```
= List.Accumulate({0..List.Count(Typed)-1}, "",
APS Deposit,1/4/2022,5174.27;APS ACH Deposit,4/4/2022,65186.66
```

The result is a text string with commas delimiting the columns and semi-colons delimiting the rows

Next, split the string into rows using the semi-colon

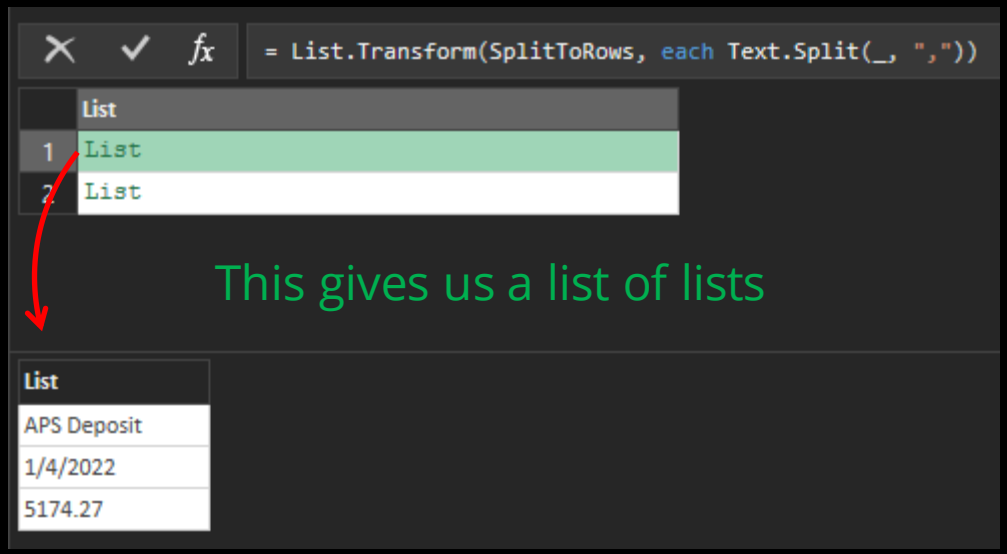
```
…
    Accumulate = List.Accumulate(…),
    SplitToRows = Text.Split(Accumulate, ";")
in
    SplitToRows
```



This gives us a list of comma-separated text strings

And split the columns using the comma

```
…
    SplitToRows = Text.Split(Accumulate, ";"),
    SplitToColumns = List.Transform(SplitToRows,
                                    each Text.Split(_, ","))
in
    SplitToColumns
```



Owen Price
flexyourdata.com, @flexyourdata

Finally, we convert the list of lists to a Table, then assign the correct data types to each column

```
…
    AsTable = Table.FromRows(SplitToColumns,
                             {"Description", "Date", "Amount"}),
    Result = Table.TransformColumnTypes(
                AsTable,
                {
                    {"Description", type text},
                    {"Date", type date},
                    {"Amount", Currency.Type}
                })
in
    Result
```

= Table.TransformColumnTypes(

	Description	Date	Amount
1	APS Deposit	2022-01-04	5,174.27
2	APS ACH Deposit	2022-04-04	65,186.66

Takeaways:

1. Parsing long strings of data into Table-like formats can be tricky
2. When faced with ambiguous dates, it’s always best to explicitly state the culture
3. Identifying where rows and columns end and begin before starting is critical

To explore, grab the code

https://gist.github.com/ncalm

GitHub Gist | Search... | All gists | Back to GitHub

Owen Price
ncalm

Data guy. Interested in Excel, Python, Tableau, DAX, M, R, SQL

All gists 175 | Forked 1 | Starred 16 | Type: All

ncalm / powerquery-TextSplit.m — 1 file — 0 forks

Created 1 minute ago

This code demonstrates one way to parse a string of fields into multiple rows and columns

```
let
    Source = "APS Deposit 04/01/2022 $5,174.27 APS ACH Deposit 04/04/2022 $65,186.66",
    Split = Text.Split(Source, " "),
    Typed = List.Transform(Split,
        each try Number.From(_) otherwise
            try Date.From(_, "en-gb") otherwise _ ),
    Accumulate = List.Accumulate({0..List.Count(Typed)-1}, "",
        (a, b) =>
            let
                c = Typed{b},
```