# The Finer Things In Alteryx

# Ken Black 10/2/17

**Topic 4: Regex and Date Operations (Multiple weekly examples)**

From Week 4 of the Weekly challenges

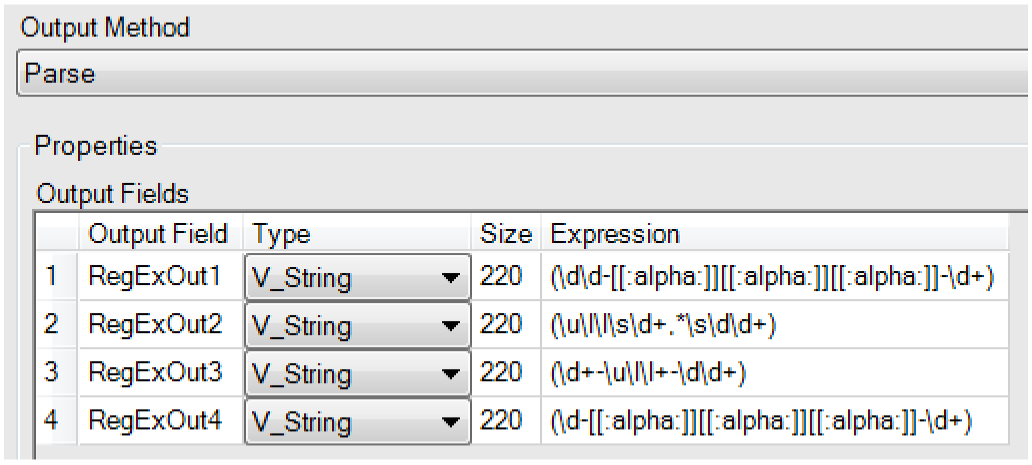
.*(\d\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+).* | .*(\u\l\l\s\d+,*\s\d\d+).* | .*(\d+-\u\l\l+-\d\d+).* | .*(\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+).*

There are four regex searches here -> and the example data that matches the search:

1. .*(\d\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+).* -> 16-APR-2005
2. .*(\u\l\l\s\d+,*\s\d\d+).* -> Nov 16, 1900
3. .*(\d+-\u\l\l+-\d\d+).* -> 9-July-2001
4. .*(\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+).* -> 4-SEP-00

**Notice that the pipe (|) is used to delimit the searches and that ".*" is used at the beginning and the end of the searches to be able to find the 4 search patterns anywhere in the search area.**

Alteryx creates 4 output fields sized at 220 to handle the content of the four searches, when the Parse method is used.



Output Method

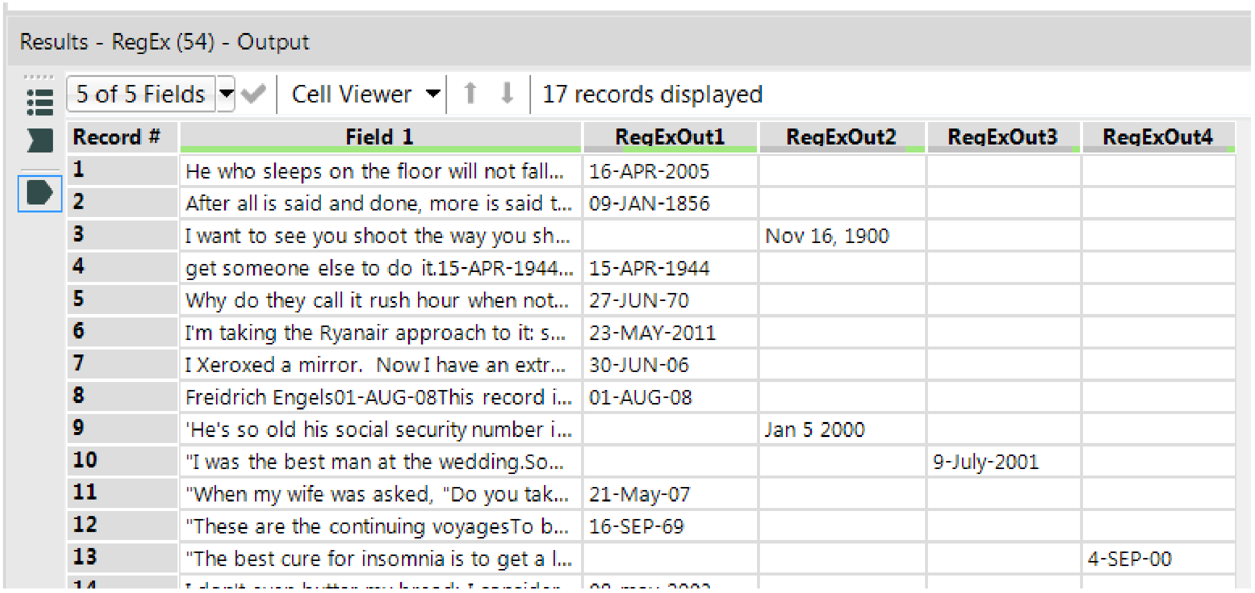
Parse

Properties

Output Fields

| | Output Field | Type | Size | Expression |
|---|---|---|---|---|
| 1 | RegExOut1 | V_String | 220 | (\d\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+) |
| 2 | RegExOut2 | V_String | 220 | (\u\l\l\s\d+,*\s\d\d+) |
| 3 | RegExOut3 | V_String | 220 | (\d+-\u\l\l+-\d\d+) |
| 4 | RegExOut4 | V_String | 220 | (\d-[[:alpha:]][[:alpha:]][[:alpha:]]-\d+) |

Example matches of these are:

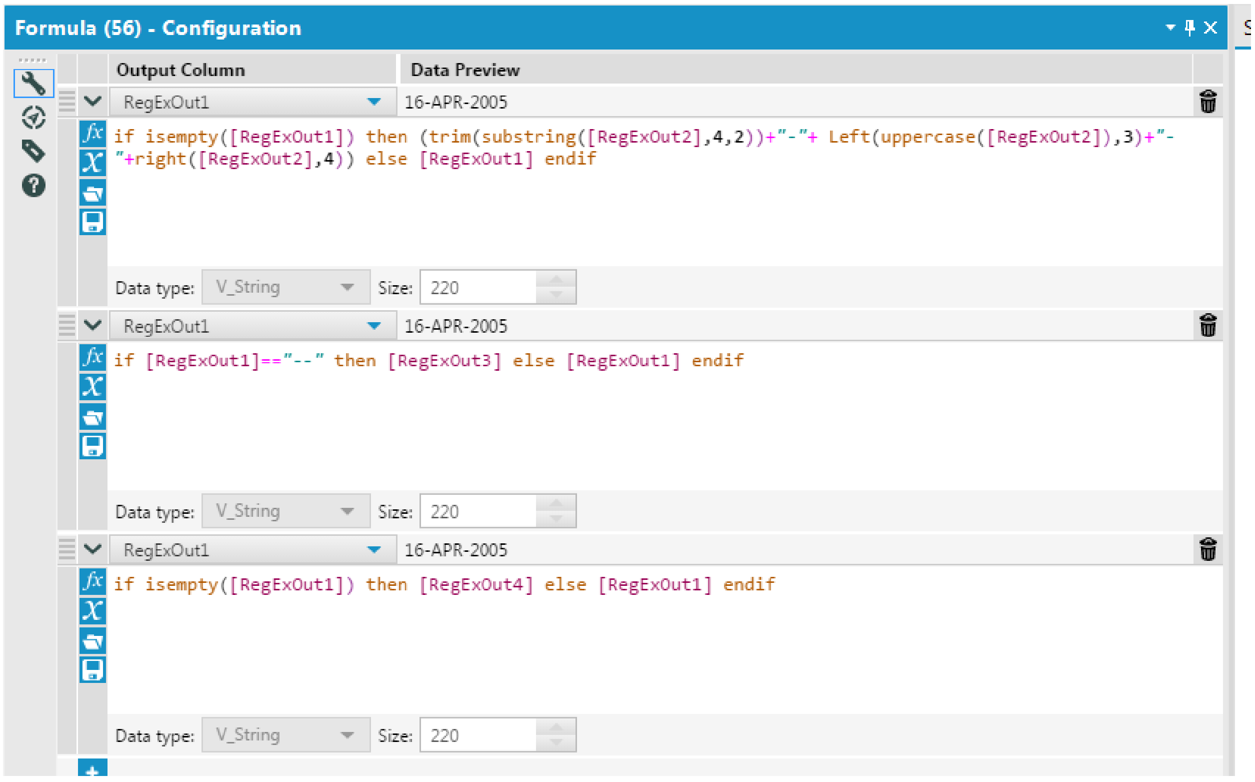


Results - RegEx (54) - Output

5 of 5 Fields · Cell Viewer · 17 records displayed

| Record # | Field 1 | RegExOut1 | RegExOut2 | RegExOut3 | RegExOut4 |
|---|---|---|---|---|---|
| 1 | He who sleeps on the floor will not fall... | 16-APR-2005 | | | |
| 2 | After all is said and done, more is said t... | 09-JAN-1856 | | | |
| 3 | I want to see you shoot the way you sh... | | Nov 16, 1900 | | |
| 4 | get someone else to do it.15-APR-1944... | 15-APR-1944 | | | |
| 5 | Why do they call it rush hour when not... | 27-JUN-70 | | | |
| 6 | I'm taking the Ryanair approach to it: s... | 23-MAY-2011 | | | |
| 7 | I Xeroxed a mirror.  Now I have an extr... | 30-JUN-06 | | | |
| 8 | Freidrich Engels01-AUG-08This record i... | 01-AUG-08 | | | |
| 9 | 'He's so old his social security number i... | | Jan 5 2000 | | |
| 10 | "I was the best man at the wedding.So... | | | 9-July-2001 | |
| 11 | "When my wife was asked, "Do you tak... | 21-May-07 | | | |
| 12 | "These are the continuing voyagesTo b... | 16-SEP-69 | | | |
| 13 | "The best cure for insomnia is to get a l... | | | | 4-SEP-00 |
| 14 | [illegible] | [illegible] | | | |

After some additional work using a formula tool,



Formula (56) - Configuration

| Output Column | Data Preview |
|---|---|
| RegExOut1 | 16-APR-2005 |

```
if isempty([RegExOut1]) then (trim(substring([RegExOut2],4,2))+"-"+ Left(uppercase([RegExOut2]),3)+"-
"+right([RegExOut2],4)) else [RegExOut1] endif
```

Data type: V_String Size: 220

| Output Column | Data Preview |
|---|---|
| RegExOut1 | 16-APR-2005 |

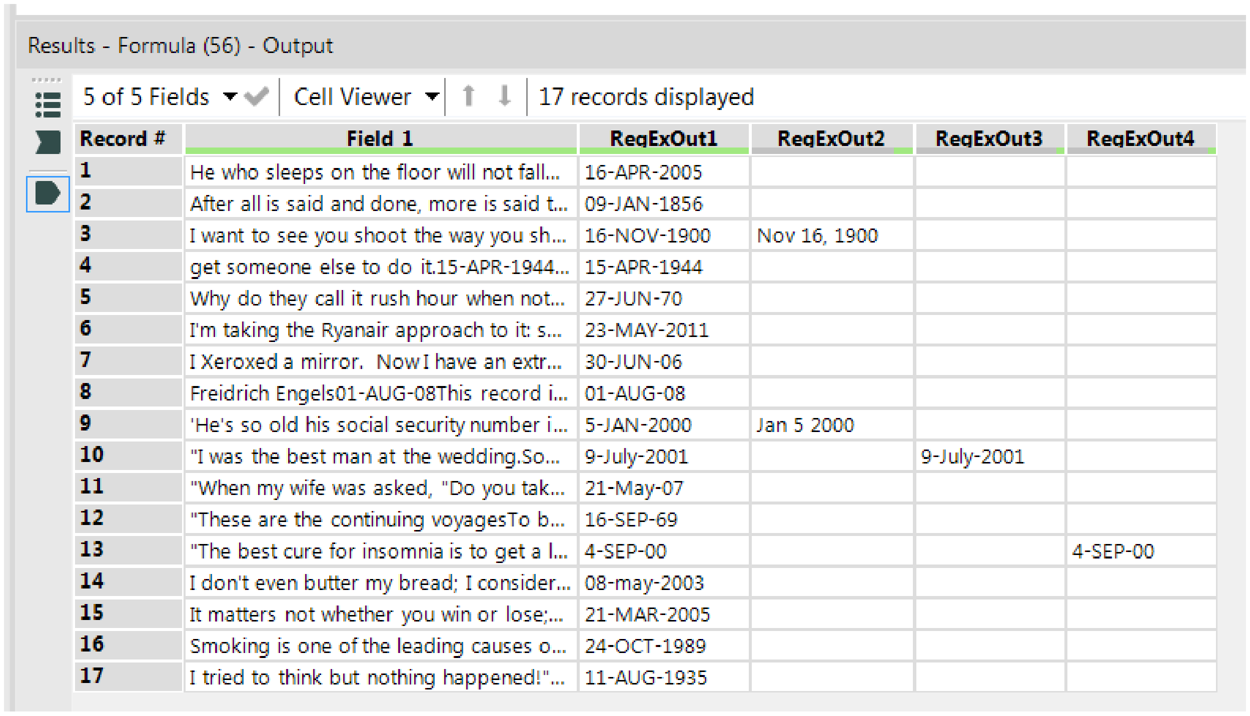
```
if [RegExOut1]=="--" then [RegExOut3] else [RegExOut1] endif
```

Data type: V_String Size: 220

| Output Column | Data Preview |
|---|---|
| RegExOut1 | 16-APR-2005 |

```
if isempty([RegExOut1]) then [RegExOut4] else [RegExOut1] endif
```

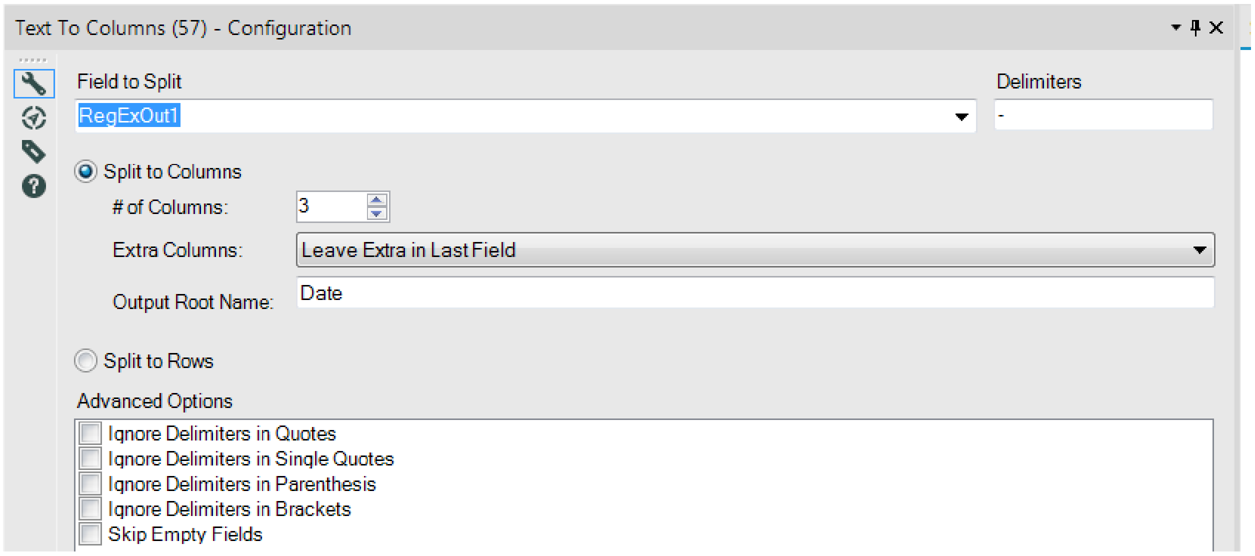
Data type: V_String Size: 220


Results - Formula (56) - Output

5 of 5 Fields | Cell Viewer | 17 records displayed

| Record # | Field 1 | RegExOut1 | RegExOut2 | RegExOut3 | RegExOut4 |
|---|---|---|---|---|---|
| 1 | He who sleeps on the floor will not fall... | 16-APR-2005 | | | |
| 2 | After all is said and done, more is said t... | 09-JAN-1856 | | | |
| 3 | I want to see you shoot the way you sh... | 16-NOV-1900 | Nov 16, 1900 | | |
| 4 | get someone else to do it.15-APR-1944... | 15-APR-1944 | | | |
| 5 | Why do they call it rush hour when not... | 27-JUN-70 | | | |
| 6 | I'm taking the Ryanair approach to it: s... | 23-MAY-2011 | | | |
| 7 | I Xeroxed a mirror.  Now I have an extr... | 30-JUN-06 | | | |
| 8 | Freidrich Engels01-AUG-08This record i... | 01-AUG-08 | | | |
| 9 | 'He's so old his social security number i... | 5-JAN-2000 | Jan 5 2000 | | |
| 10 | "I was the best man at the wedding.So... | 9-July-2001 | | 9-July-2001 | |
| 11 | "When my wife was asked, "Do you tak... | 21-May-07 | | | |
| 12 | "These are the continuing voyagesTo b... | 16-SEP-69 | | | |
| 13 | "The best cure for insomnia is to get a l... | 4-SEP-00 | | | 4-SEP-00 |
| 14 | I don't even butter my bread; I consider... | 08-may-2003 | | | |
| 15 | It matters not whether you win or lose;... | 21-MAR-2005 | | | |
| 16 | Smoking is one of the leading causes o... | 24-OCT-1989 | | | |
| 17 | I tried to think but nothing happened!"... | 11-AUG-1935 | | | |

And a text to columns parse:


Text To Columns (57) - Configuration

Field to Split: RegExOut1

Delimiters: -

[✓ Split to Columns]

# of Columns: 3

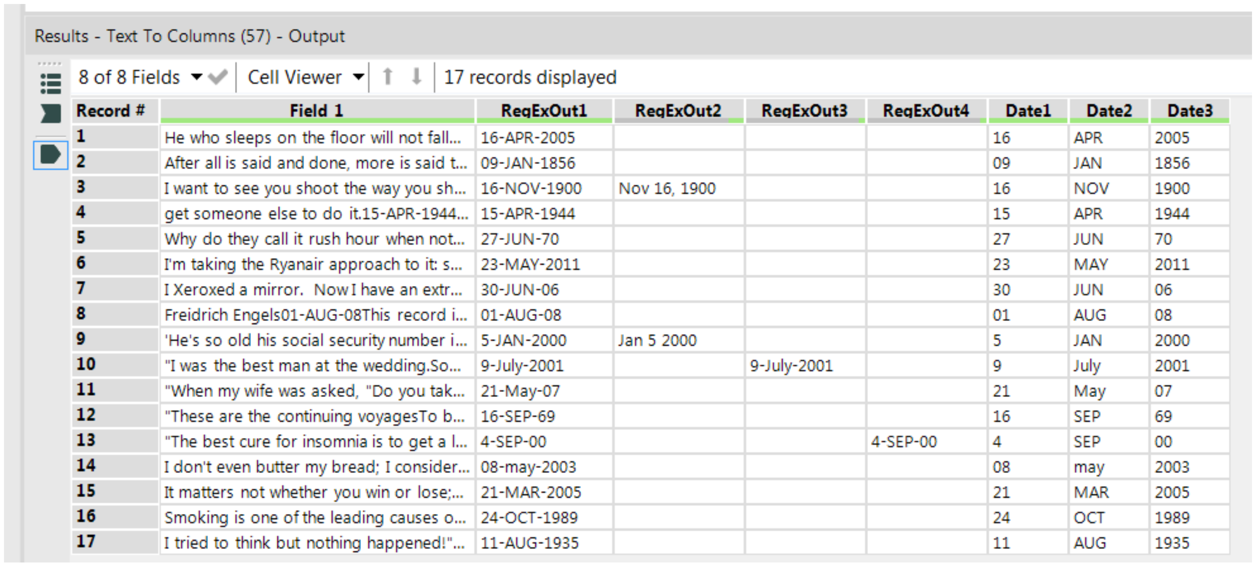
Extra Columns: Leave Extra in Last Field

Output Root Name: Date

Split to Rows

Advanced Options

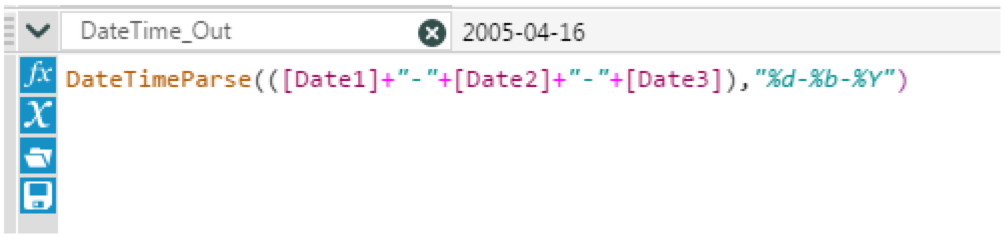
- Ignore Delimiters in Quotes
- Ignore Delimiters in Single Quotes
- Ignore Delimiters in Parenthesis
- Ignore Delimiters in Brackets
- Skip Empty Fields


Results - Text To Columns (57) - Output

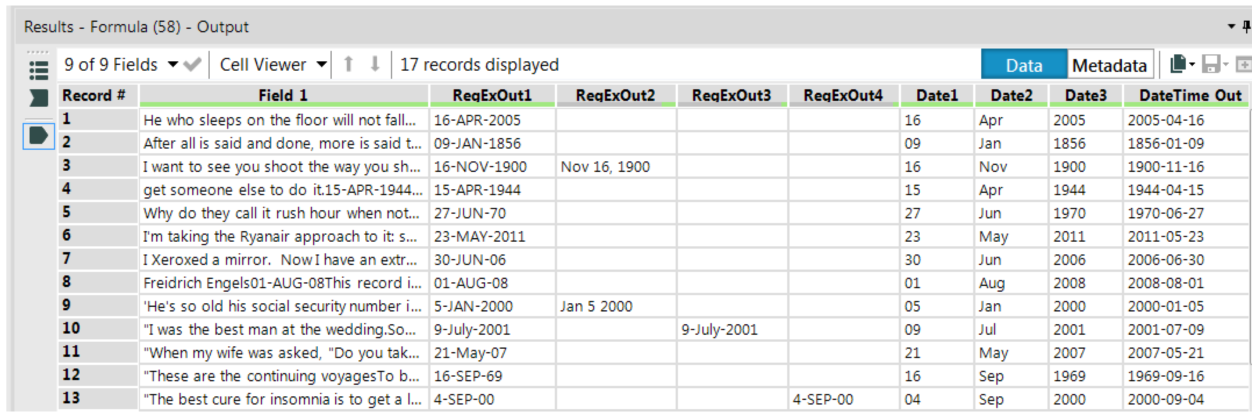
8 of 8 Fields | Cell Viewer | 17 records displayed

| Record # | Field 1 | RegExOut1 | RegExOut2 | RegExOut3 | RegExOut4 | Date1 | Date2 | Date3 |
|---|---|---|---|---|---|---|---|---|
| 1 | He who sleeps on the floor will not fall... | 16-APR-2005 | | | | 16 | APR | 2005 |
| 2 | After all is said and done, more is said t... | 09-JAN-1856 | | | | 09 | JAN | 1856 |
| 3 | I want to see you shoot the way you sh... | 16-NOV-1900 | Nov 16, 1900 | | | 16 | NOV | 1900 |
| 4 | get someone else to do it.15-APR-1944... | 15-APR-1944 | | | | 15 | APR | 1944 |
| 5 | Why do they call it rush hour when not... | 27-JUN-70 | | | | 27 | JUN | 70 |
| 6 | I'm taking the Ryanair approach to it: s... | 23-MAY-2011 | | | | 23 | MAY | 2011 |
| 7 | I Xeroxed a mirror. Now I have an extr... | 30-JUN-06 | | | | 30 | JUN | 06 |
| 8 | Freidrich Engels01-AUG-08This record i... | 01-AUG-08 | | | | 01 | AUG | 08 |
| 9 | 'He's so old his social security number i... | 5-JAN-2000 | Jan 5 2000 | | | 5 | JAN | 2000 |
| 10 | "I was the best man at the wedding.So... | 9-July-2001 | | 9-July-2001 | | 9 | July | 2001 |
| 11 | "When my wife was asked, "Do you tak... | 21-May-07 | | | | 21 | May | 07 |
| 12 | "These are the continuing voyagesTo b... | 16-SEP-69 | | | | 16 | SEP | 69 |
| 13 | "The best cure for insomnia is to get a l... | 4-SEP-00 | | | 4-SEP-00 | 4 | SEP | 00 |
| 14 | I don't even butter my bread; I consider... | 08-may-2003 | | | | 08 | may | 2003 |
| 15 | It matters not whether you win or lose;... | 21-MAR-2005 | | | | 21 | MAR | 2005 |
| 16 | Smoking is one of the leading causes o... | 24-OCT-1989 | | | | 24 | OCT | 1989 |
| 17 | I tried to think but nothing happened!"... | 11-AUG-1935 | | | | 11 | AUG | 1935 |

the final dates are assembled using the DateTimeParse function:


DateTime_Out 2005-04-16

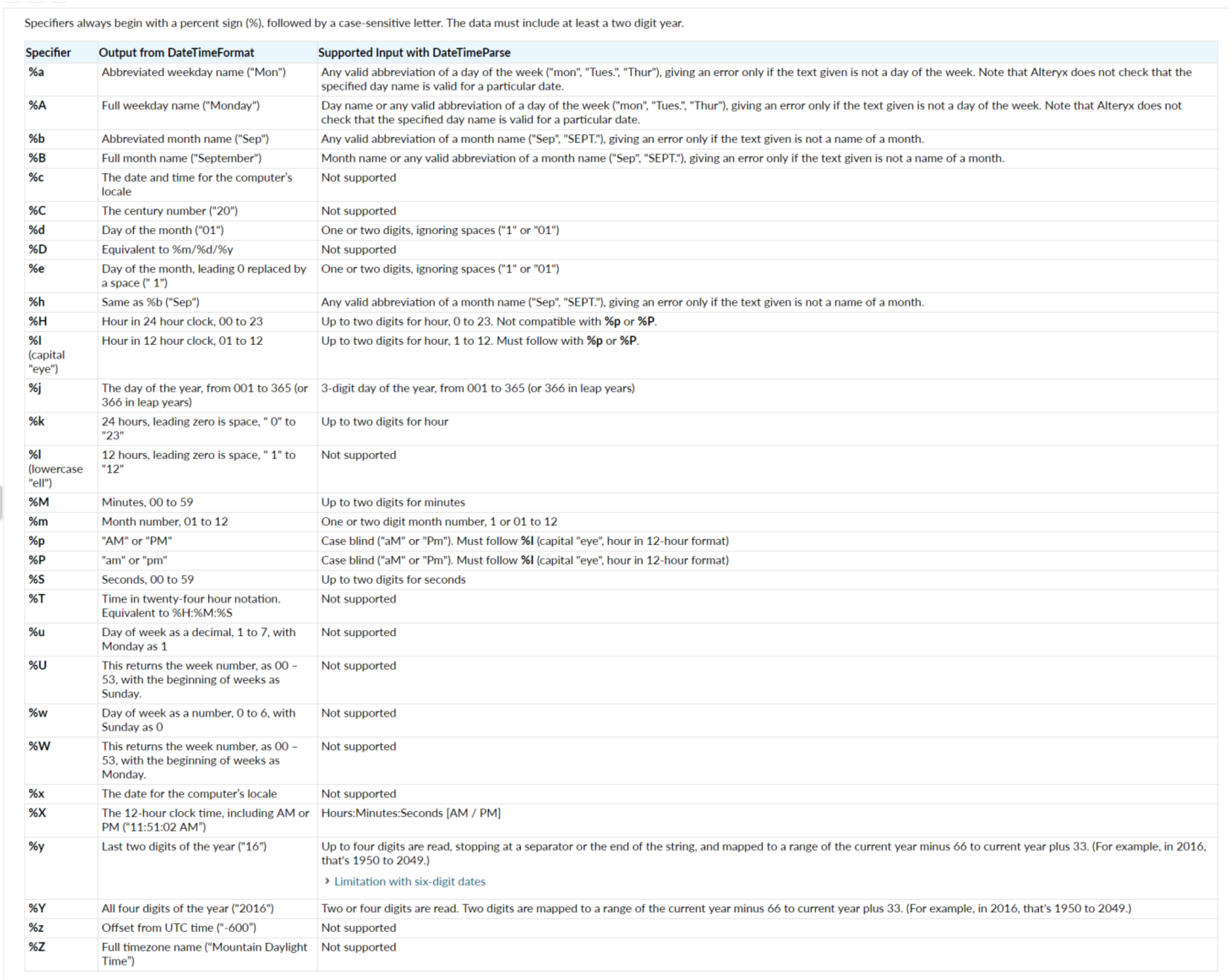
```
DateTimeParse(([Date1]+"-"+[Date2]+"-"+[Date3]),"%d-%b-%Y")
```


Results - Formula (58) - Output

9 of 9 Fields | Cell Viewer | 17 records displayed | Data | Metadata

| Record # | Field 1 | RegExOut1 | RegExOut2 | RegExOut3 | RegExOut4 | Date1 | Date2 | Date3 | DateTime Out |
|---|---|---|---|---|---|---|---|---|---|
| 1 | He who sleeps on the floor will not fall... | 16-APR-2005 | | | | 16 | Apr | 2005 | 2005-04-16 |
| 2 | After all is said and done, more is said t... | 09-JAN-1856 | | | | 09 | Jan | 1856 | 1856-01-09 |
| 3 | I want to see you shoot the way you sh... | 16-NOV-1900 | Nov 16, 1900 | | | 16 | Nov | 1900 | 1900-11-16 |
| 4 | get someone else to do it.15-APR-1944... | 15-APR-1944 | | | | 15 | Apr | 1944 | 1944-04-15 |
| 5 | Why do they call it rush hour when not... | 27-JUN-70 | | | | 27 | Jun | 1970 | 1970-06-27 |
| 6 | I'm taking the Ryanair approach to it: s... | 23-MAY-2011 | | | | 23 | May | 2011 | 2011-05-23 |
| 7 | I Xeroxed a mirror. Now I have an extr... | 30-JUN-06 | | | | 30 | Jun | 2006 | 2006-06-30 |
| 8 | Freidrich Engels01-AUG-08This record i... | 01-AUG-08 | | | | 01 | Aug | 2008 | 2008-08-01 |
| 9 | 'He's so old his social security number i... | 5-JAN-2000 | Jan 5 2000 | | | 05 | Jan | 2000 | 2000-01-05 |
| 10 | "I was the best man at the wedding.So... | 9-July-2001 | | 9-July-2001 | | 09 | Jul | 2001 | 2001-07-09 |
| 11 | "When my wife was asked, "Do you tak... | 21-May-07 | | | | 21 | May | 2007 | 2007-05-21 |
| 12 | "These are the continuing voyagesTo b... | 16-SEP-69 | | | | 16 | Sep | 1969 | 1969-09-16 |
| 13 | "The best cure for insomnia is to get a l... | 4-SEP-00 | | | 4-SEP-00 | 04 | Sep | 2000 | 2000-09-04 |

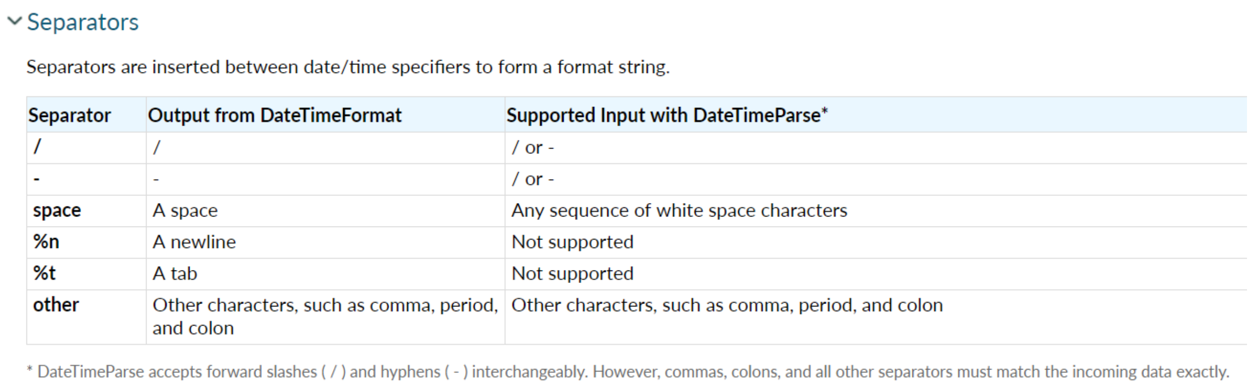
For Reference, here are the specifiers used for dates/time for Alteryx:



Specifiers always begin with a percent sign (%), followed by a case-sensitive letter. The data must include at least a two digit year.

| Specifier | Output from DateTimeFormat | Supported Input with DateTimeParse |
|---|---|---|
| **%a** | Abbreviated weekday name ("Mon") | Any valid abbreviation of a day of the week ("mon", "Tues.", "Thur"), giving an error only if the text given is not a day of the week. Note that Alteryx does not check that the specified day name is valid for a particular date. |
| **%A** | Full weekday name ("Monday") | Day name or any valid abbreviation of a day of the week ("mon", "Tues.", "Thur"), giving an error only if the text given is not a day of the week. Note that Alteryx does not check that the specified day name is valid for a particular date. |
| **%b** | Abbreviated month name ("Sep") | Any valid abbreviation of a month name ("Sep", "SEPT."), giving an error only if the text given is not a name of a month. |
| **%B** | Full month name ("September") | Month name or any valid abbreviation of a month name ("Sep", "SEPT."), giving an error only if the text given is not a name of a month. |
| **%c** | The date and time for the computer's locale | Not supported |
| **%C** | The century number ("20") | Not supported |
| **%d** | Day of the month ("01") | One or two digits, ignoring spaces ("1" or "01") |
| **%D** | Equivalent to %m/%d/%y | Not supported |
| **%e** | Day of the month, leading 0 replaced by a space (" 1") | One or two digits, ignoring spaces ("1" or "01") |
| **%h** | Same as %b ("Sep") | Any valid abbreviation of a month name ("Sep", "SEPT."), giving an error only if the text given is not a name of a month. |
| **%H** | Hour in 24 hour clock, 00 to 23 | Up to two digits for hour, 0 to 23. Not compatible with **%p** or **%P**. |
| **%I** (capital "eye") | Hour in 12 hour clock, 01 to 12 | Up to two digits for hour, 1 to 12. Must follow with **%p** or **%P**. |
| **%j** | The day of the year, from 001 to 365 (or 366 in leap years) | 3-digit day of the year, from 001 to 365 (or 366 in leap years) |
| **%k** | 24 hours, leading zero is space, " 0" to "23" | Up to two digits for hour |
| **%l** (lowercase "ell") | 12 hours, leading zero is space, " 1" to "12" | Not supported |
| **%M** | Minutes, 00 to 59 | Up to two digits for minutes |
| **%m** | Month number, 01 to 12 | One or two digit month number, 1 or 01 to 12 |
| **%p** | "AM" or "PM" | Case blind ("aM" or "Pm"). Must follow **%I** (capital "eye", hour in 12-hour format) |
| **%P** | "am" or "pm" | Case blind ("aM" or "Pm"). Must follow **%I** (capital "eye", hour in 12-hour format) |
| **%S** | Seconds, 00 to 59 | Up to two digits for seconds |
| **%T** | Time in twenty-four hour notation. Equivalent to %H:%M:%S | Not supported |
| **%u** | Day of week as a decimal, 1 to 7, with Monday as 1 | Not supported |
| **%U** | This returns the week number, as 00 – 53, with the beginning of weeks as Sunday. | Not supported |
| **%w** | Day of week as a number, 0 to 6, with Sunday as 0 | Not supported |
| **%W** | This returns the week number, as 00 – 53, with the beginning of weeks as Monday. | Not supported |
| **%x** | The date for the computer's locale | Not supported |
| **%X** | The 12-hour clock time, including AM or PM ("11:51:02 AM") | Hours:Minutes:Seconds [AM / PM] |
| **%y** | Last two digits of the year ("16") | Up to four digits are read, stopping at a separator or the end of the string, and mapped to a range of the current year minus 66 to current year plus 33. (For example, in 2016, that's 1950 to 2049.)<br>› Limitation with six-digit dates |
| **%Y** | All four digits of the year ("2016") | Two or four digits are read. Two digits are mapped to a range of the current year minus 66 to current year plus 33. (For example, in 2016, that's 1950 to 2049.) |
| **%z** | Offset from UTC time ("-600") | Not supported |
| **%Z** | Full timezone name ("Mountain Daylight Time") | Not supported |

The separators:



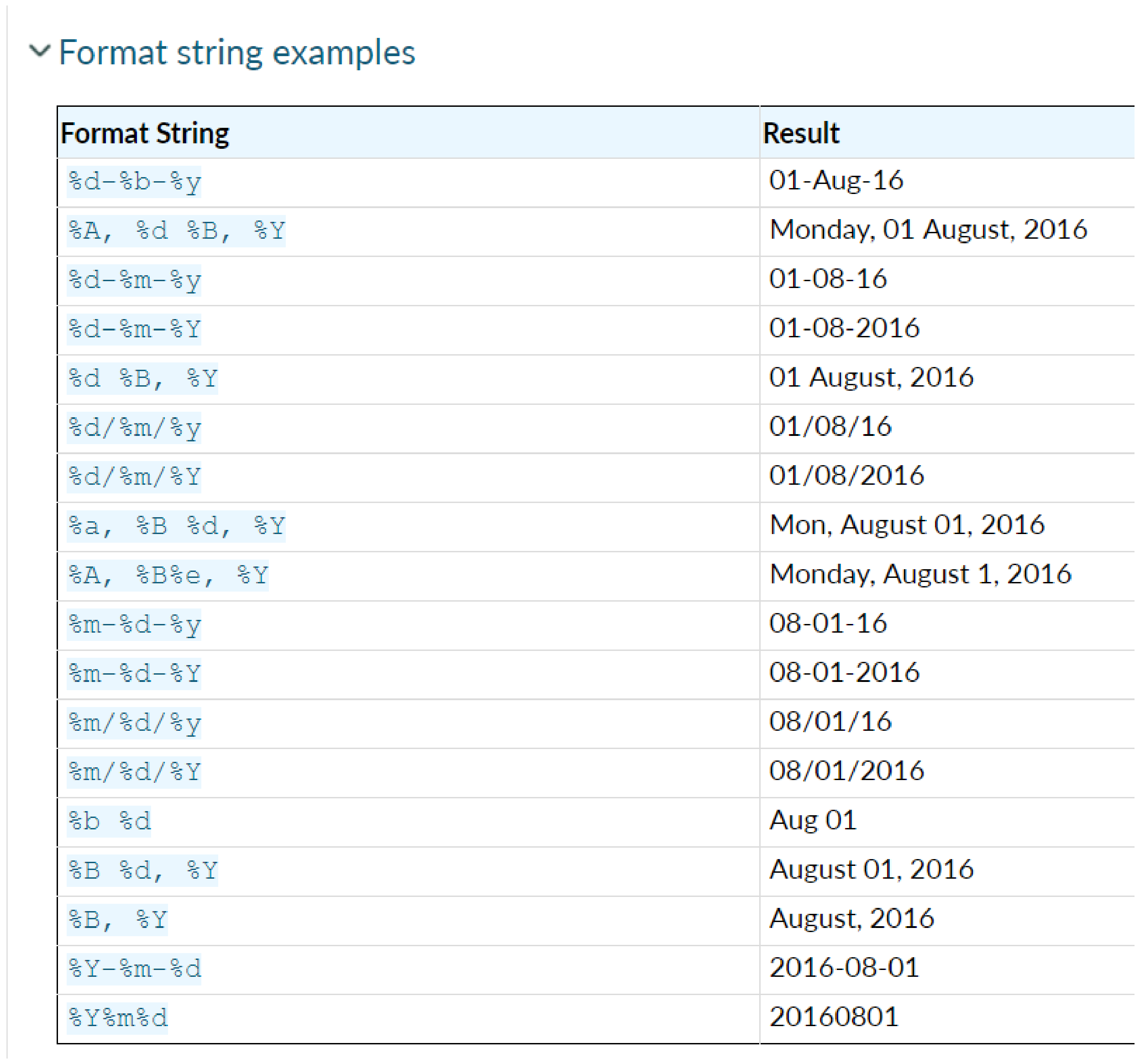
## Separators

Separators are inserted between date/time specifiers to form a format string.

| Separator | Output from DateTimeFormat | Supported Input with DateTimeParse* |
|---|---|---|
| **/** | / | / or - |
| **-** | - | / or - |
| **space** | A space | Any sequence of white space characters |
| **%n** | A newline | Not supported |
| **%t** | A tab | Not supported |
| **other** | Other characters, such as comma, period, and colon | Other characters, such as comma, period, and colon |

* DateTimeParse accepts forward slashes ( / ) and hyphens ( - ) interchangeably. However, commas, colons, and all other separators must match the incoming data exactly.

And the Date/Time Examples:



## Format string examples

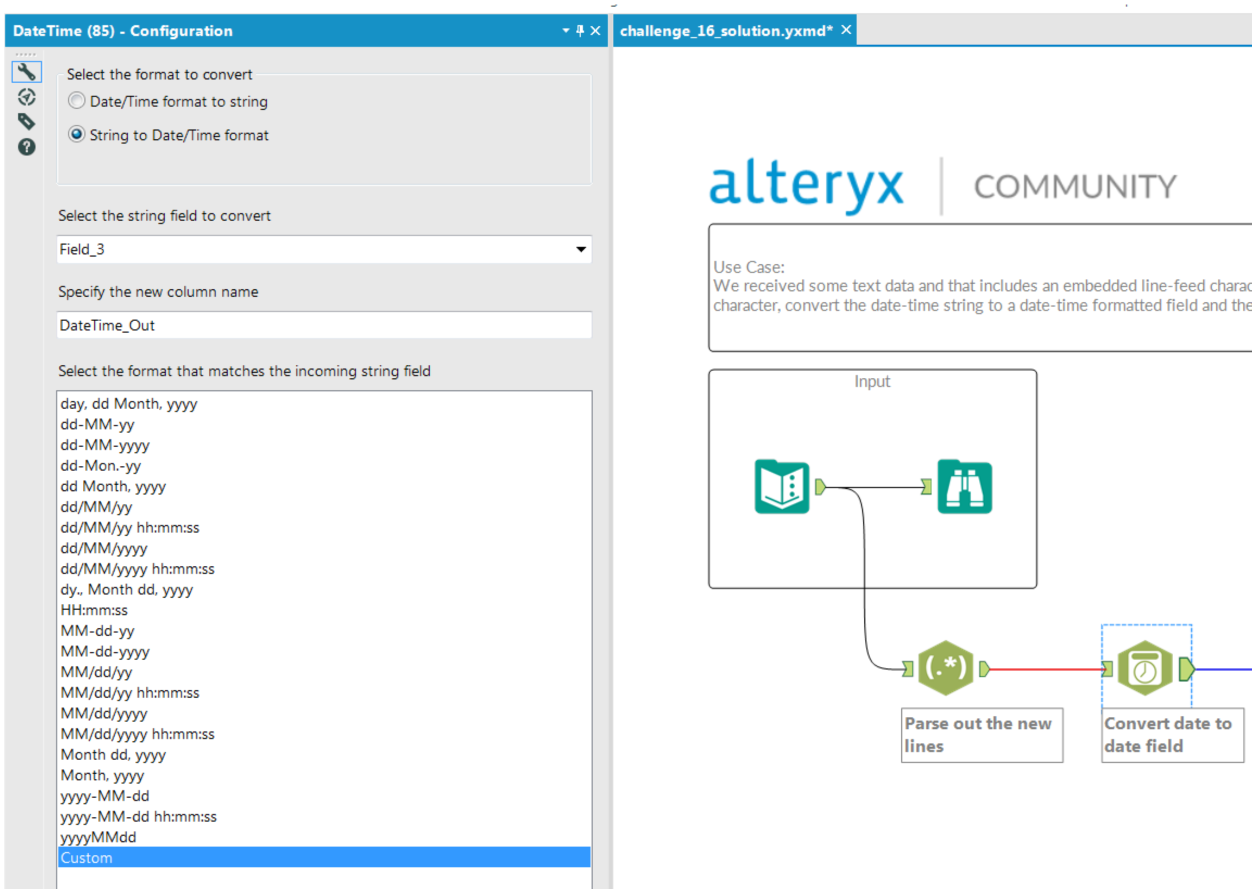
| Format String | Result |
|---|---|
| `%d-%b-%y` | 01-Aug-16 |
| `%A, %d %B, %Y` | Monday, 01 August, 2016 |
| `%d-%m-%y` | 01-08-16 |
| `%d-%m-%Y` | 01-08-2016 |
| `%d %B, %Y` | 01 August, 2016 |
| `%d/%m/%y` | 01/08/16 |
| `%d/%m/%Y` | 01/08/2016 |
| `%a, %B %d, %Y` | Mon, August 01, 2016 |
| `%A, %B%e, %Y` | Monday, August 1, 2016 |
| `%m-%d-%y` | 08-01-16 |
| `%m-%d-%Y` | 08-01-2016 |
| `%m/%d/%y` | 08/01/16 |
| `%m/%d/%Y` | 08/01/2016 |
| `%b %d` | Aug 01 |
| `%B %d, %Y` | August 01, 2016 |
| `%B, %Y` | August, 2016 |
| `%Y-%m-%d` | 2016-08-01 |
| `%Y%m%d` | 20160801 |

To find a match for anything:

*(.*?)*

## Datetime Tool Example 1: Custom format date string

From Week 16, a custom formatted string (16-JUN-01) is converted to a date (2001-06-16) using the datetime tool.



DateTime (85) - Configuration

challenge_16_solution.yxmd*

Select the format to convert

Date/Time format to string [✓ String to Date/Time format]

Select the string field to convert

Field_3

Specify the new column name

DateTime_Out

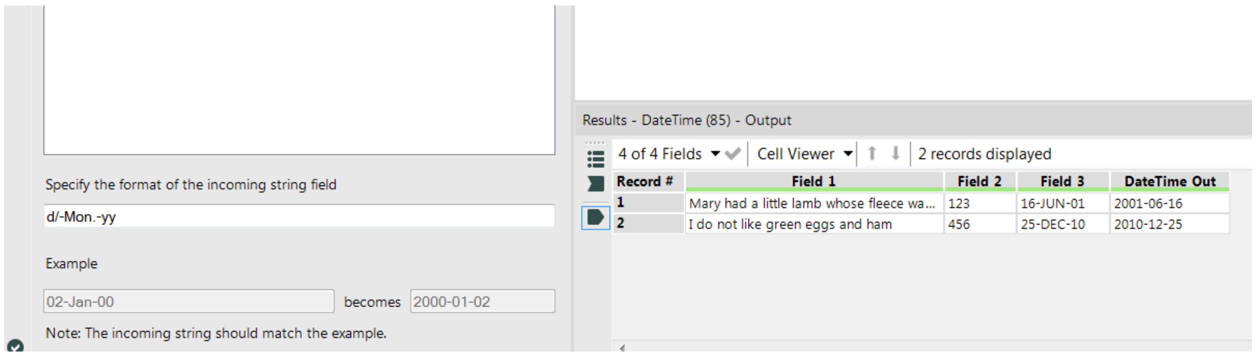
Select the format that matches the incoming string field

day, dd Month, yyyy
dd-MM-yy
dd-MM-yyyy
dd-Mon.-yy
dd Month, yyyy
dd/MM/yy
dd/MM/yy hh:mm:ss
dd/MM/yyyy
dd/MM/yyyy hh:mm:ss
dy., Month dd, yyyy
HH:mm:ss
MM-dd-yy
MM-dd-yyyy
MM/dd/yy
MM/dd/yy hh:mm:ss
MM/dd/yyyy
MM/dd/yyyy hh:mm:ss
Month dd, yyyy
Month, yyyy
yyyy-MM-dd
yyyy-MM-dd hh:mm:ss
yyyyMMdd
[✓ Custom]

alteryx | COMMUNITY

Use Case:
We received some text data and that includes an embedded line-feed charac
character, convert the date-time string to a date-time formatted field and the

The custom setting is shown below as d/-Mon.-yy .When this is used, Field 3 becomes a DateTime Out.



Specify the format of the incoming string field

d/-Mon.-yy

Example

02-Jan-00 becomes 2000-01-02

Note: The incoming string should match the example.

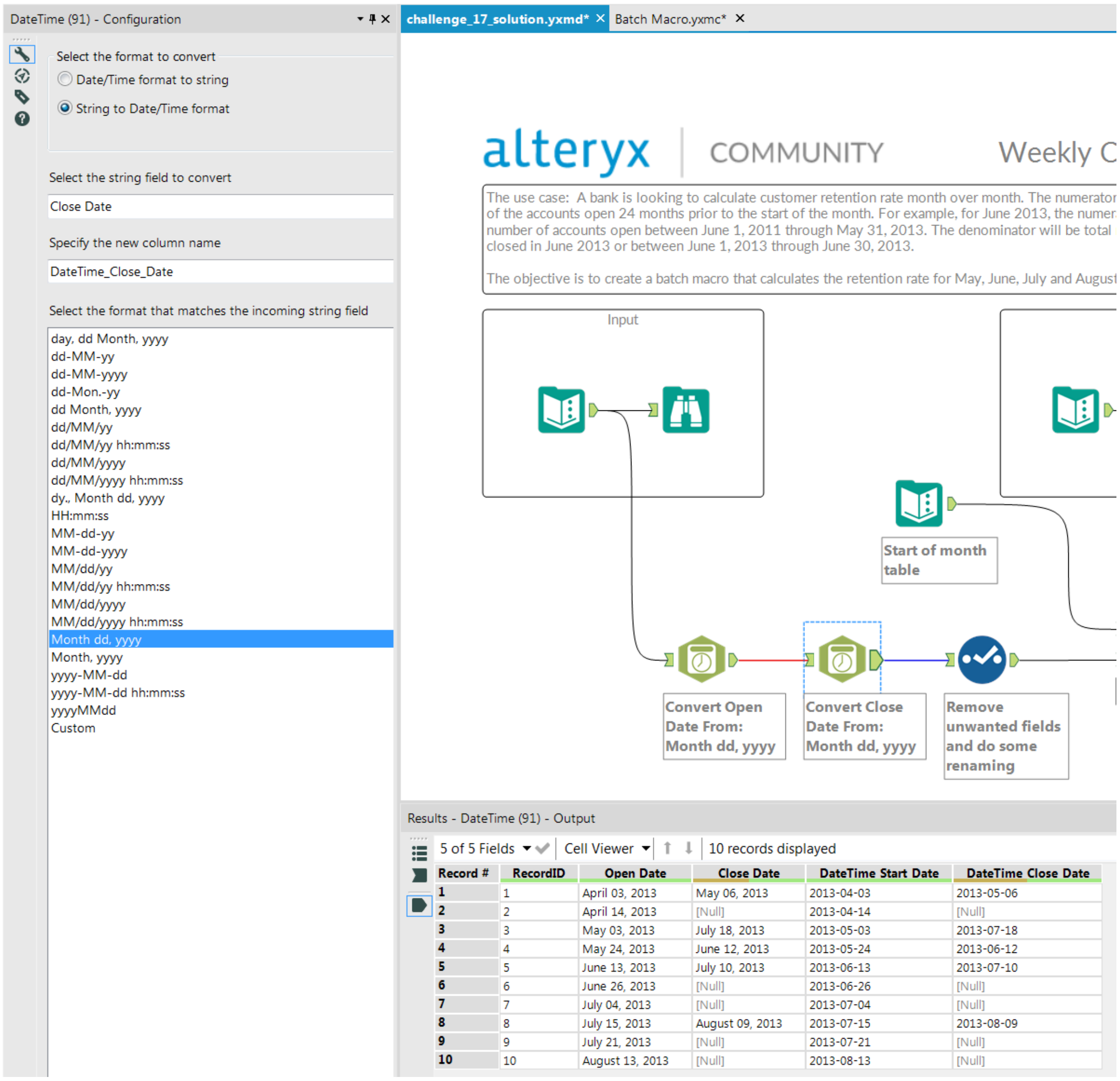
Results - DateTime (85) - Output

4 of 4 Fields · Cell Viewer · 2 records displayed

| Record # | Field 1 | Field 2 | Field 3 | DateTime Out |
|---|---|---|---|---|
| 1 | Mary had a little lamb whose fleece wa... | 123 | 16-JUN-01 | 2001-06-16 |
| 2 | I do not like green eggs and ham | 456 | 25-DEC-10 | 2010-12-25 |

## Datetime Tool Example 2: Standard format date string

From Week 17, a standard formatted string (April 03, 2013) is converted to a date (2013-04-03) using the datetime tool.



DateTime (91) - Configuration

Select the format to convert

Date/Time format to string [✓ String to Date/Time format]

Select the string field to convert: Close Date

Specify the new column name: DateTime_Close_Date

Select the format that matches the incoming string field

```
day, dd Month, yyyy
dd-MM-yy
dd-MM-yyyy
dd-Mon.-yy
dd Month, yyyy
dd/MM/yy
dd/MM/yy hh:mm:ss
dd/MM/yyyy
dd/MM/yyyy hh:mm:ss
dy., Month dd, yyyy
HH:mm:ss
MM-dd-yy
MM-dd-yyyy
MM/dd/yy
MM/dd/yy hh:mm:ss
MM/dd/yyyy
MM/dd/yyyy hh:mm:ss
Month dd, yyyy   [✓]
Month, yyyy
yyyy-MM-dd
yyyy-MM-dd hh:mm:ss
yyyyMMdd
Custom
```

challenge_17_solution.yxmd* Batch Macro.yxmc*

The use case: A bank is looking to calculate customer retention rate month over month. The numerator
of the accounts open 24 months prior to the start of the month. For example, for June 2013, the numer
number of accounts open between June 1, 2011 through May 31, 2013. The denominator will be total
closed in June 2013 or between June 1, 2013 through June 30, 2013.

The objective is to create a batch macro that calculates the retention rate for May, June, July and August

alteryx
COMMUNITY
Weekly C
Input
Start of month table
Convert Open Date From: Month dd, yyyy
Convert Close Date From: Month dd, yyyy
Remove unwanted fields and do some renaming

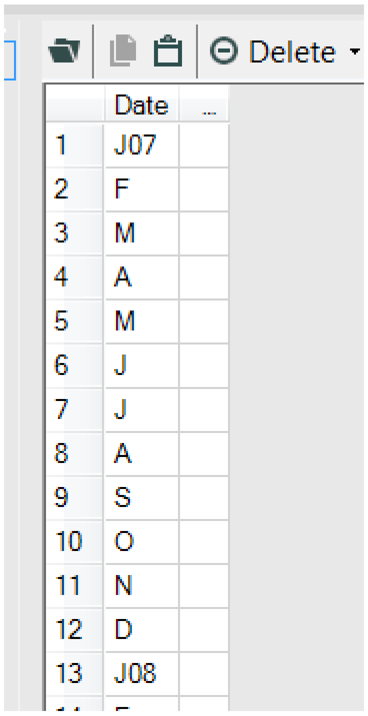
Results - DateTime (91) - Output

5 of 5 Fields | Cell Viewer | 10 records displayed

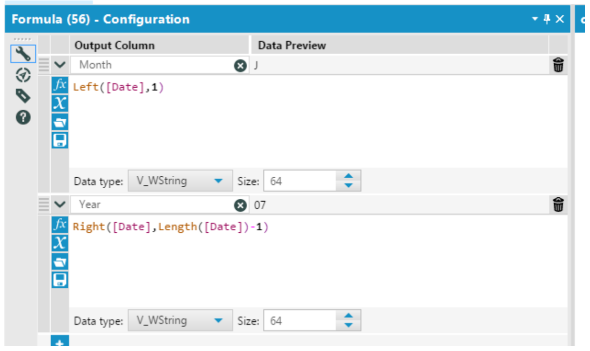
| Record # | RecordID | Open Date | Close Date | DateTime Start Date | DateTime Close Date |
|---|---|---|---|---|---|
| 1 | 1 | April 03, 2013 | May 06, 2013 | 2013-04-03 | 2013-05-06 |
| 2 | 2 | April 14, 2013 | [Null] | 2013-04-14 | [Null] |
| 3 | 3 | May 03, 2013 | July 18, 2013 | 2013-05-03 | 2013-07-18 |
| 4 | 4 | May 24, 2013 | June 12, 2013 | 2013-05-24 | 2013-06-12 |
| 5 | 5 | June 13, 2013 | July 10, 2013 | 2013-06-13 | 2013-07-10 |
| 6 | 6 | June 26, 2013 | [Null] | 2013-06-26 | [Null] |
| 7 | 7 | July 04, 2013 | [Null] | 2013-07-04 | [Null] |
| 8 | 8 | July 15, 2013 | August 09, 2013 | 2013-07-15 | 2013-08-09 |
| 9 | 9 | July 21, 2013 | [Null] | 2013-07-21 | [Null] |
| 10 | 10 | August 13, 2013 | [Null] | 2013-08-13 | [Null] |

**Week 21 – More Custom Date Work**

In this example, very sketchy date details are provided and complete month/years are created from the information. Here is the initial sketchy data followed by the parsing of month and year.



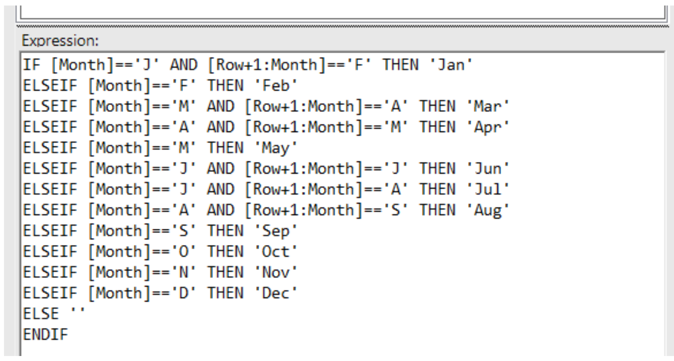
| | Date |
|---|---|
| 1 | J07 |
| 2 | F |
| 3 | M |
| 4 | A |
| 5 | M |
| 6 | J |
| 7 | J |
| 8 | A |
| 9 | S |
| 10 | O |
| 11 | N |
| 12 | D |
| 13 | J08 |



| Output Column | Data Preview |
|---|---|
| Month | J |
| Year | 07 |

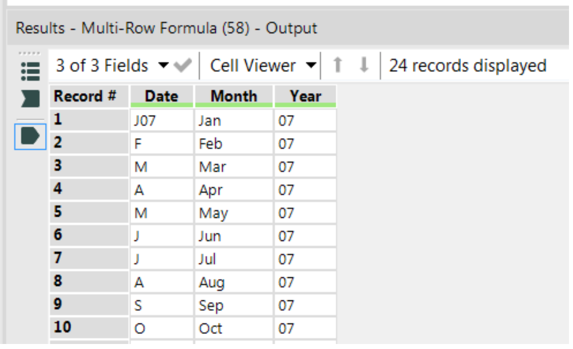
```
Month: Left([Date],1)
Data type: V_WString  Size: 64

Year: Right([Date],Length([Date])-1)
Data type: V_WString  Size: 64
```

Here is the final date output, showing the clever logic used to rename the months:



Expression:

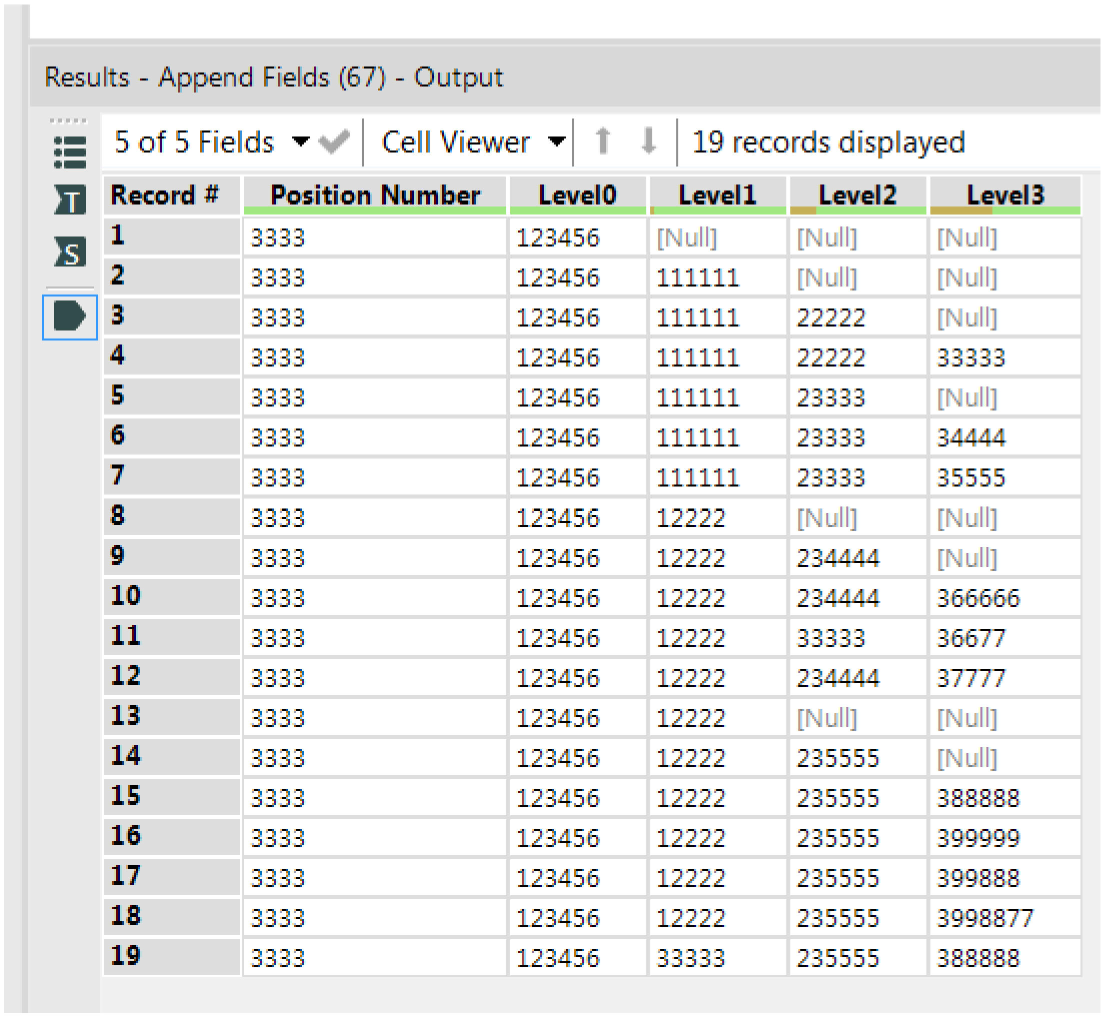
```
IF [Month]=='J' AND [Row+1:Month]=='F' THEN 'Jan'
ELSEIF [Month]=='F' THEN 'Feb'
ELSEIF [Month]=='M' AND [Row+1:Month]=='A' THEN 'Mar'
ELSEIF [Month]=='A' AND [Row+1:Month]=='M' THEN 'Apr'
ELSEIF [Month]=='M' THEN 'May'
ELSEIF [Month]=='J' AND [Row+1:Month]=='J' THEN 'Jun'
ELSEIF [Month]=='J' AND [Row+1:Month]=='A' THEN 'Jul'
ELSEIF [Month]=='A' AND [Row+1:Month]=='S' THEN 'Aug'
ELSEIF [Month]=='S' THEN 'Sep'
ELSEIF [Month]=='O' THEN 'Oct'
ELSEIF [Month]=='N' THEN 'Nov'
ELSEIF [Month]=='D' THEN 'Dec'
ELSE ''
ENDIF
```



| Record # | Date | Month | Year |
|---|---|---|---|
| 1 | J07 | Jan | 07 |
| 2 | F | Feb | 07 |
| 3 | M | Mar | 07 |
| 4 | A | Apr | 07 |
| 5 | M | May | 07 |
| 6 | J | Jun | 07 |
| 7 | J | Jul | 07 |
| 8 | A | Aug | 07 |
| 9 | S | Sep | 07 |
| 10 | O | Oct | 07 |

**Topic 5: Multifield searching and matching (Week 5)**

The append tool is used to create combinations of an input value and records in a database such that the input field can be found in any of the columns of the database. The append operation creates the combinations needed for this to be possible, and a simple if block does the comparisons.

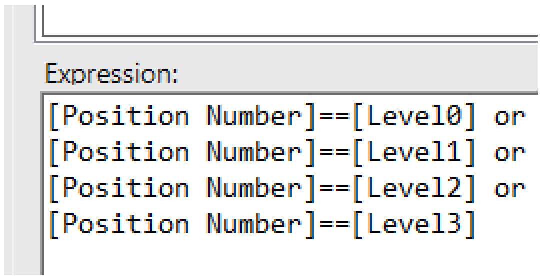


Results - Append Fields (67) - Output

5 of 5 Fields | Cell Viewer | 19 records displayed

| Record # | Position Number | Level0 | Level1 | Level2 | Level3 |
|---|---|---|---|---|---|
| 1 | 3333 | 123456 | [Null] | [Null] | [Null] |
| 2 | 3333 | 123456 | 111111 | [Null] | [Null] |
| 3 | 3333 | 123456 | 111111 | 22222 | [Null] |
| 4 | 3333 | 123456 | 111111 | 22222 | 33333 |
| 5 | 3333 | 123456 | 111111 | 23333 | [Null] |
| 6 | 3333 | 123456 | 111111 | 23333 | 34444 |
| 7 | 3333 | 123456 | 111111 | 23333 | 35555 |
| 8 | 3333 | 123456 | 12222 | [Null] | [Null] |
| 9 | 3333 | 123456 | 12222 | 234444 | [Null] |
| 10 | 3333 | 123456 | 12222 | 234444 | 366666 |
| 11 | 3333 | 123456 | 12222 | 33333 | 36677 |
| 12 | 3333 | 123456 | 12222 | 234444 | 37777 |
| 13 | 3333 | 123456 | 12222 | [Null] | [Null] |
| 14 | 3333 | 123456 | 12222 | 235555 | [Null] |
| 15 | 3333 | 123456 | 12222 | 235555 | 388888 |
| 16 | 3333 | 123456 | 12222 | 235555 | 399999 |
| 17 | 3333 | 123456 | 12222 | 235555 | 399888 |
| 18 | 3333 | 123456 | 12222 | 235555 | 3998877 |
| 19 | 3333 | 123456 | 33333 | 235555 | 388888 |

The user input of 3333 is appended to the database records. The following logic identifies the records where 3333 is found.

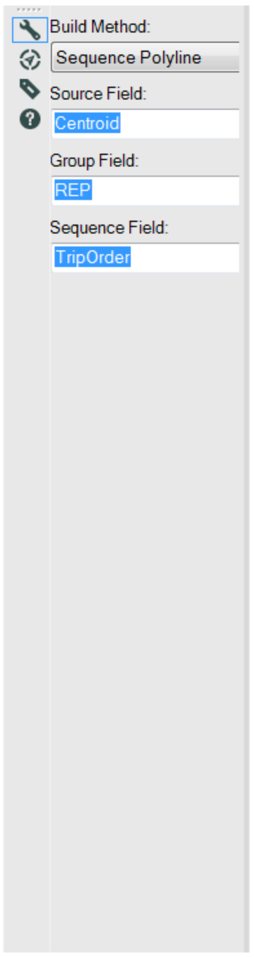


Expression:

```
[Position Number]==[Level0] or
[Position Number]==[Level1] or
[Position Number]==[Level2] or
[Position Number]==[Level3]
```

**Topic 6: Length along a Polyline (Week 6)**

A sequence of airport trips are strung together to find out which sales rep as traveled the most miles. The airport lat/longs are given as centroids so all that is necessary is to produce polylines for each sales rep and use the spatial info tool to calculate the distance traveled by each sales rep.

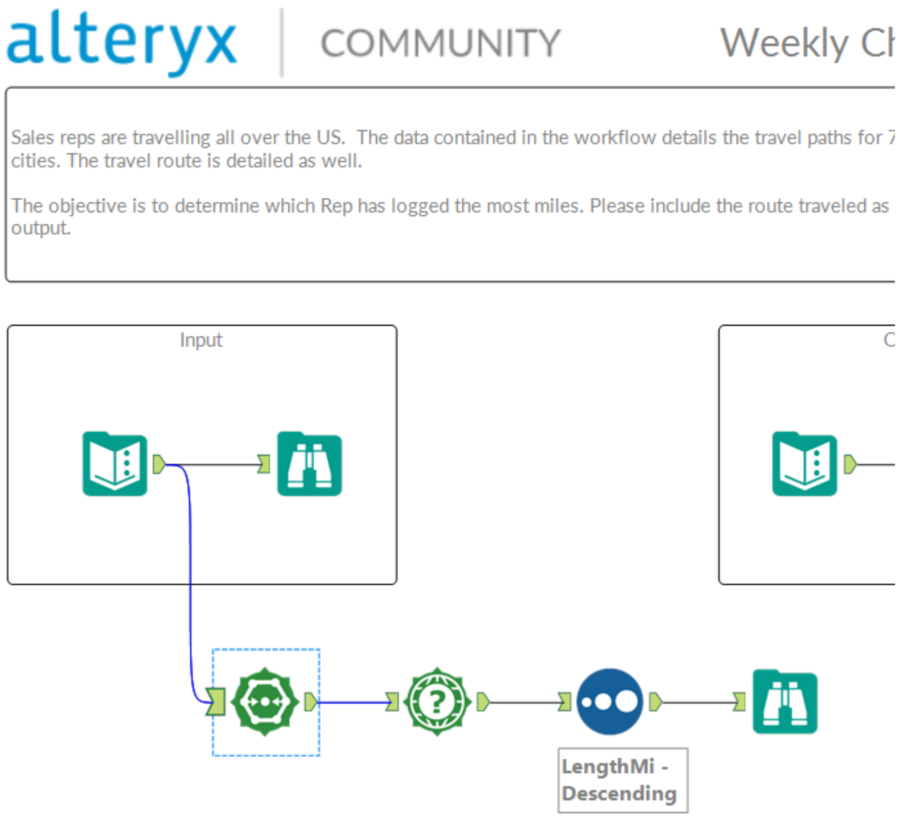


Build Method: Sequence Polyline

Source Field: Centroid

Group Field: REP

Sequence Field: TripOrder



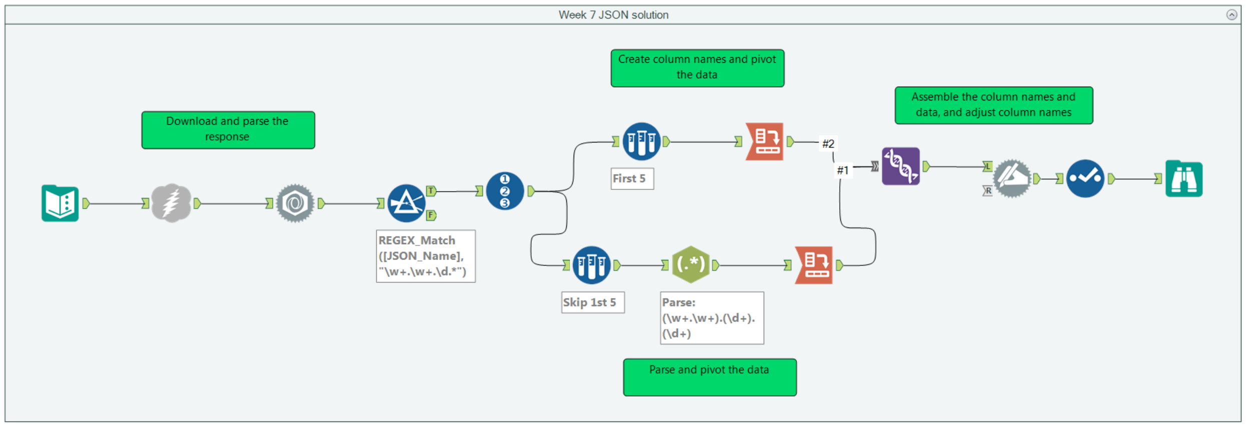
alteryx | COMMUNITY Weekly Ch

Sales reps are travelling all over the US. The data contained in the workflow details the travel paths for 7 cities. The travel route is detailed as well.

The objective is to determine which Rep has logged the most miles. Please include the route traveled as output.

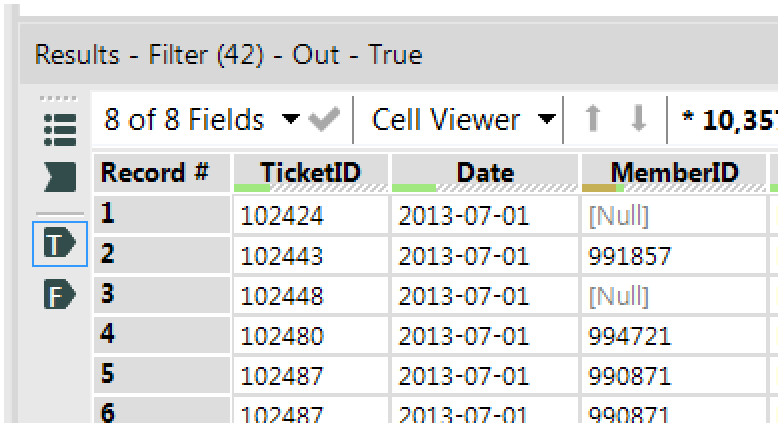
**Topic 7: Parsing JSON Data (Week 7)**

This is an excellent example in so many ways. The methods used to identify the JSON data elements are insightful and efficient. There are so many excellent maneuvers in this example that it is one of the best exercises to date. I have rarely used the sample tool, and it is used in two different ways here. I have never used the JSON tool, so it was good to learn. Finally, the use of regex and the dynamic rename tool were both good.


Week 7 JSON solution
Download and parse the response
REGEX_Match ([JSON_Name], "\w+.\w+.\d.*")
Create column names and pivot the data
First 5
Skip 1st 5
Parse: (\w+.\w+).(\d+).(\d+)
Parse and pivot the data
#2
#1
Assemble the column names and data, and adjust column names

**Topic 8: Filtering by date (week 8)**

**Given date data like:**

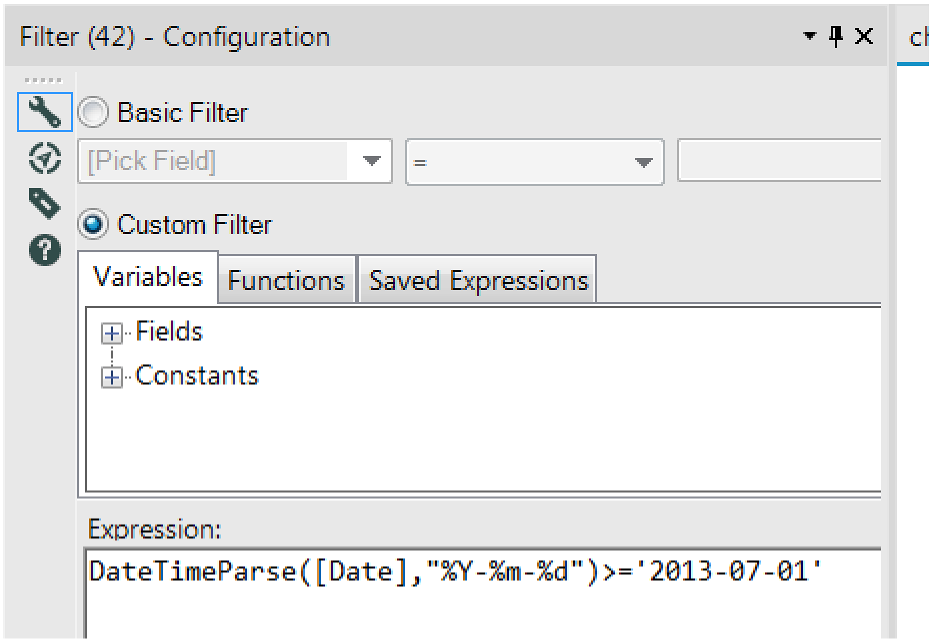


Results - Filter (42) - Out - True

8 of 8 Fields | Cell Viewer | * 10,35

| Record # | TicketID | Date | MemberID |
|---|---|---|---|
| 1 | 102424 | 2013-07-01 | [Null] |
| 2 | 102443 | 2013-07-01 | 991857 |
| 3 | 102448 | 2013-07-01 | [Null] |
| 4 | 102480 | 2013-07-01 | 994721 |
| 5 | 102487 | 2013-07-01 | 990871 |
| 6 | 102487 | 2013-07-01 | 990871 |

**Configure a filter to allow date-based filtering**



Filter (42) - Configuration

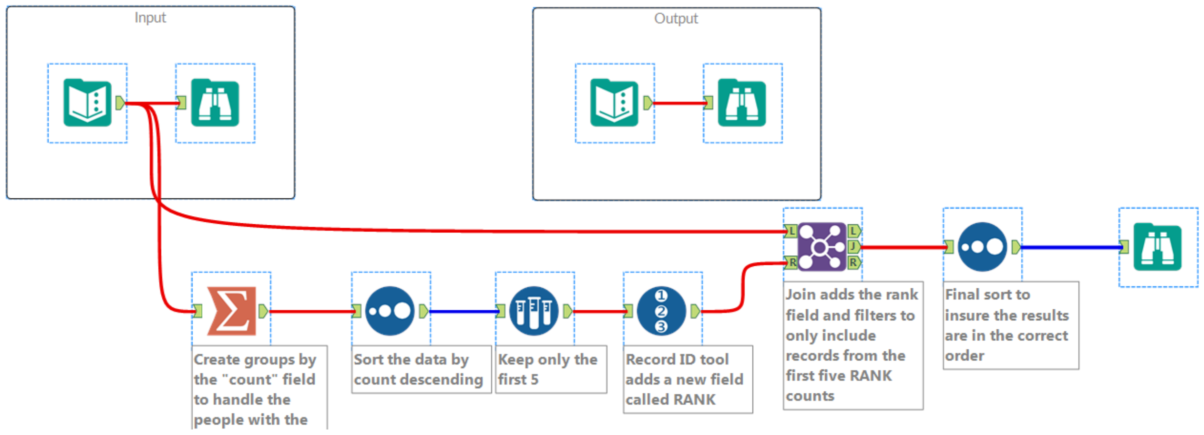
Basic Filter: [Pick Field] =

[✓ Custom Filter]

Expression:

```
DateTimeParse([Date],"%Y-%m-%d")>='2013-07-01'
```

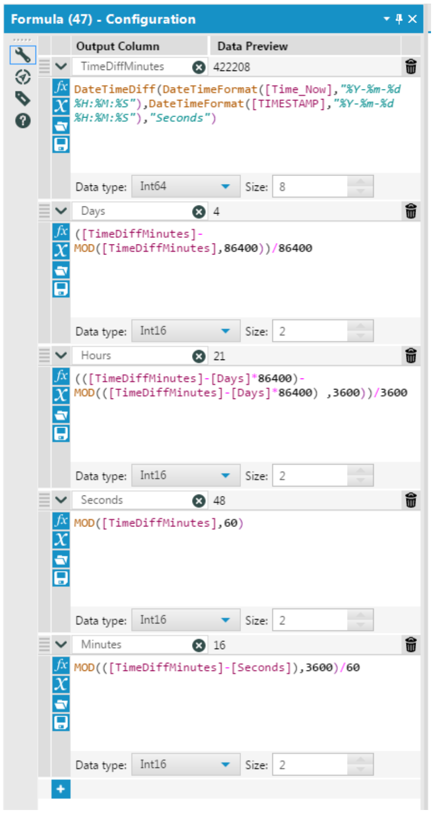
**Topic 9: Ranking items where there can be more than 1 at the same rank level, and performing a top N calculation (Week 9)**


Input
Output
Create groups by the "count" field to handle the people with the
Sort the data by count descending
Keep only the first 5
Record ID tool adds a new field called RANK
Join adds the rank field and filters to only include records from the first five RANK counts
Final sort to insure the results are in the correct order

I like this example because of the use of the sample tool to identify the top N ranks, and also for the use of the clever technique used to assign the ranks (using a join).

## Topic 10: Calculating Time (Days, hours, minutes, seconds)

Has an error in the naming of the first formula. This says it is a time difference in minutes but is actually a difference in seconds. Otherwise, excellent instructional on how to calculate discrete time blocks.


Formula (47) - Configuration

| Output Column | Data Preview |
|---|---|
| TimeDiffMinutes | 422208 |
| Days | 4 |
| Hours | 21 |
| Seconds | 48 |
| Minutes | 16 |

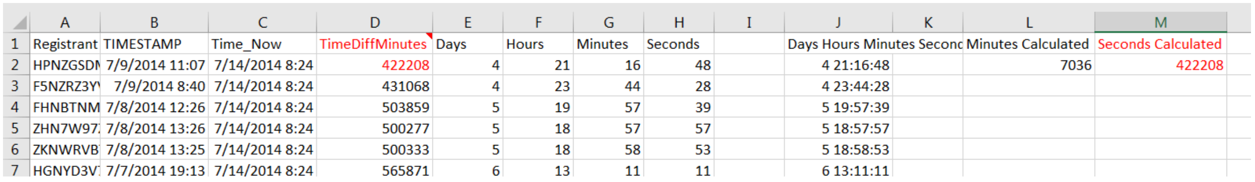
TimeDiffMinutes:
```
DateTimeDiff(DateTimeFormat([Time_Now],"%Y-%m-%d
%H:%M:%S"),DateTimeFormat([TIMESTAMP],"%Y-%m-%d
%H:%M:%S"),"Seconds")
```
Data type: Int64 Size: 8

Days:
```
([TimeDiffMinutes]-
MOD([TimeDiffMinutes],86400))/86400
```
Data type: Int16 Size: 2

Hours:
```
(([TimeDiffMinutes]-[Days]*86400)-
MOD(([TimeDiffMinutes]-[Days]*86400) ,3600))/3600
```
Data type: Int16 Size: 2

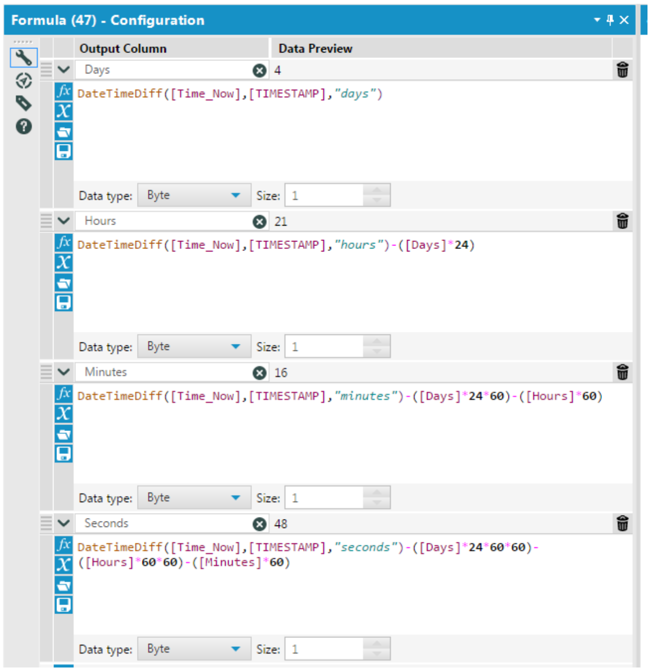
Seconds:
```
MOD([TimeDiffMinutes],60)
```
Data type: Int16 Size: 2

Minutes:
```
MOD(([TimeDiffMinutes]-[Seconds]),3600)/60
```
Data type: Int16 Size: 2



| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Registrant | TIMESTAMP | Time_Now | TimeDiffMinutes | Days | Hours | Minutes | Seconds | | Days Hours Minutes Seconc | | Minutes Calculated | Seconds Calculated |
| 2 | HPNZGSDI | 7/9/2014 11:07 | 7/14/2014 8:24 | 422208 | 4 | 21 | 16 | 48 | | 4 21:16:48 | | 7036 | 422208 |
| 3 | F5NZRZ3Y | 7/9/2014 8:40 | 7/14/2014 8:24 | 431068 | 4 | 23 | 44 | 28 | | 4 23:44:28 | | | |
| 4 | FHNBTNM | 7/8/2014 12:26 | 7/14/2014 8:24 | 503859 | 5 | 19 | 57 | 39 | | 5 19:57:39 | | | |
| 5 | ZHN7W97 | 7/8/2014 13:26 | 7/14/2014 8:24 | 500277 | 5 | 18 | 57 | 57 | | 5 18:57:57 | | | |
| 6 | ZKNWRVB | 7/8/2014 13:25 | 7/14/2014 8:24 | 500333 | 5 | 18 | 58 | 53 | | 5 18:58:53 | | | |
| 7 | HGNYD3V | 7/7/2014 19:13 | 7/14/2014 8:24 | 565871 | 6 | 13 | 11 | 11 | | 6 13:11:11 | | | |

For a more efficient solution, see the following formulas



Formula (47) - Configuration

| Output Column | Data Preview |
|---|---|
| Days | 4 |

```
DateTimeDiff([Time_Now],[TIMESTAMP],"days")
```

Data type: Byte Size: 1

| | |
|---|---|
| Hours | 21 |

```
DateTimeDiff([Time_Now],[TIMESTAMP],"hours")-([Days]*24)
```

Data type: Byte Size: 1

| | |
|---|---|
| Minutes | 16 |

```
DateTimeDiff([Time_Now],[TIMESTAMP],"minutes")-([Days]*24*60)-([Hours]*60)
```

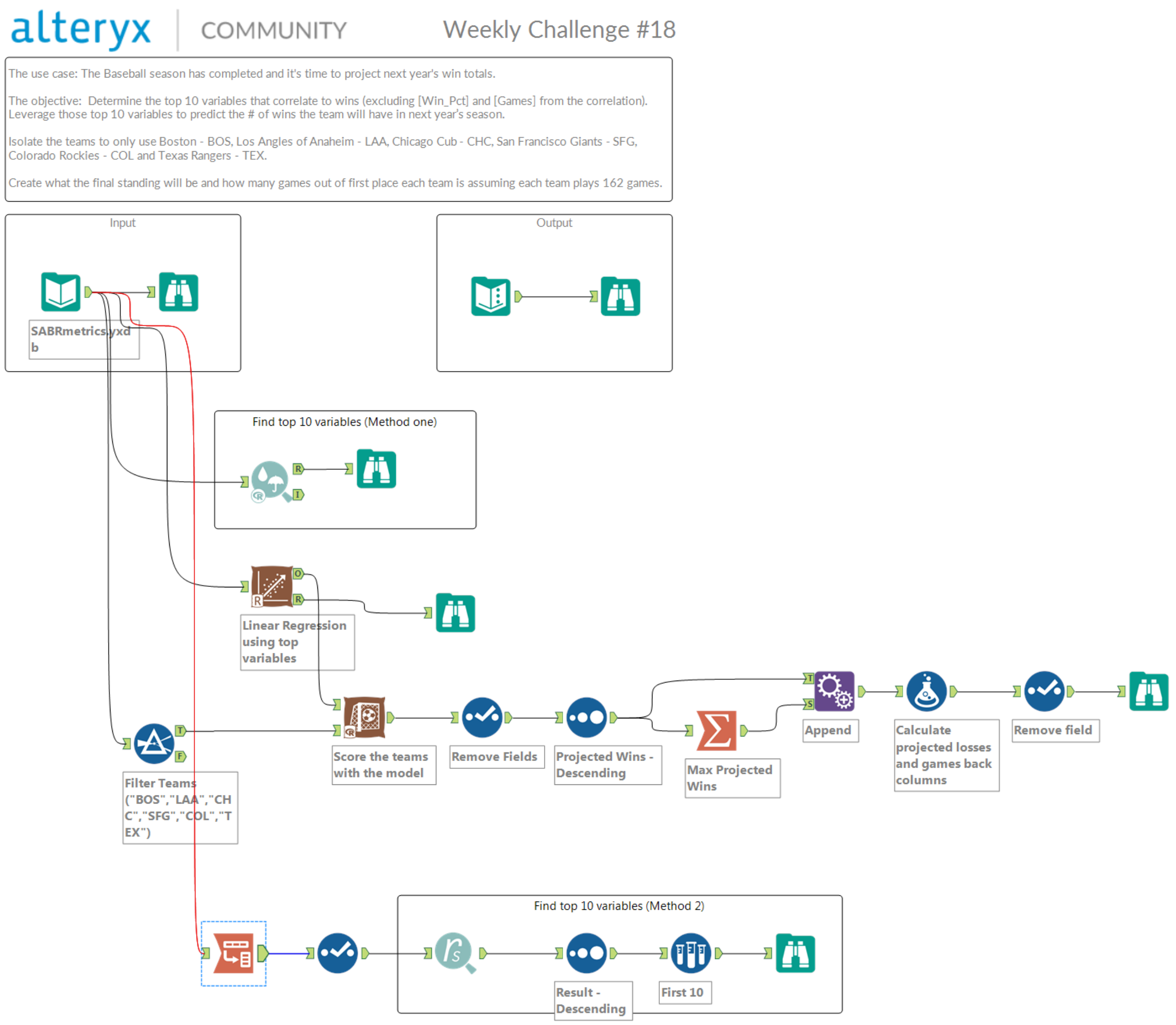
Data type: Byte Size: 1

| | |
|---|---|
| Seconds | 48 |

```
DateTimeDiff([Time_Now],[TIMESTAMP],"seconds")-([Days]*24*60*60)-
([Hours]*60*60)-([Minutes]*60)
```

Data type: Byte Size: 1

## Topic 11: Linear Regression Modeling


alteryx | COMMUNITY
Weekly Challenge #18
Input
SABRmetrics.yxdb
Output
Find top 10 variables (Method one)
Linear Regression using top variables
Filter Teams ("BOS","LAA","CHC","SFG","COL","TEX")
Score the teams with the model
Remove Fields
Projected Wins - Descending
Max Projected Wins
Append
Calculate projected losses and games back columns
Remove field
Find top 10 variables (Method 2)
Result - Descending
First 10

The use case: The Baseball season has completed and it's time to project next year's win totals.

The objective: Determine the top 10 variables that correlate to wins (excluding [Win_Pct] and [Games] from the correlation). Leverage those top 10 variables to predict the # of wins the team will have in next year's season.

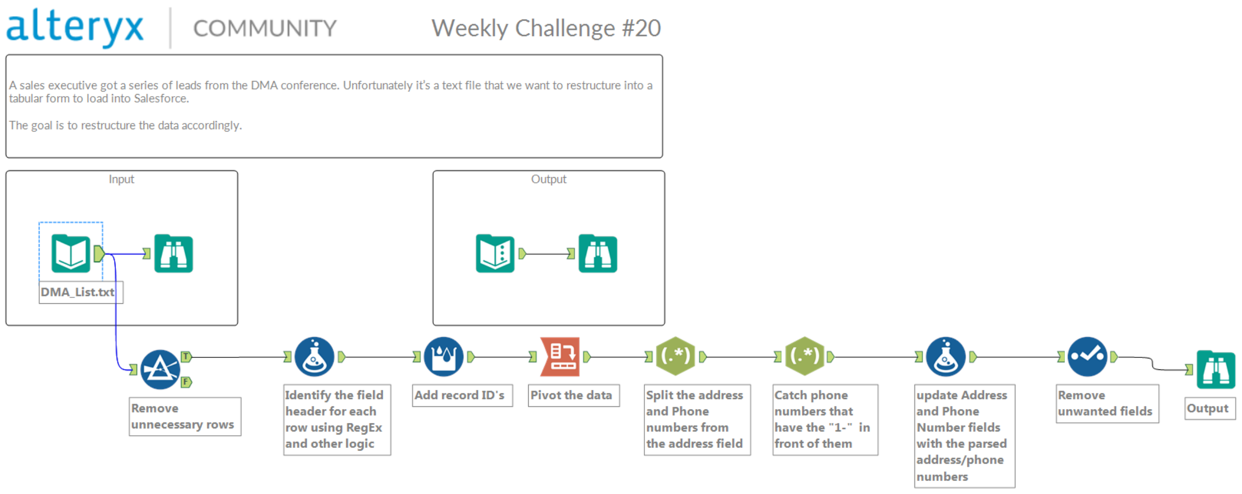
Isolate the teams to only use Boston - BOS, Los Angles of Anaheim - LAA, Chicago Cub - CHC, San Francisco Giants - SFG, Colorado Rockies - COL and Texas Rangers - TEX.

Create what the final standing will be and how many games out of first place each team is assuming each team plays 162 games.

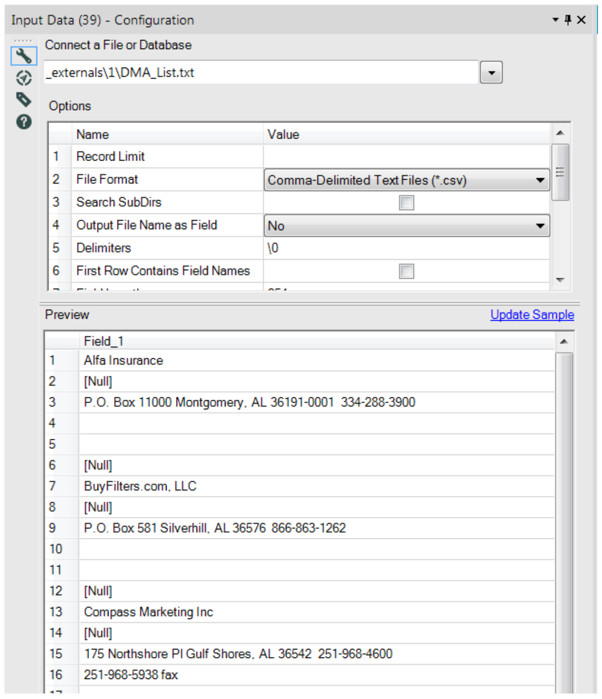
I like this example because it uses the Spearman Correlation tool to identify the top 10 statistics that are most strongly correlated to winning baseball games (lower part of workflow) than then these terms are used in a linear regression model to estimate how teams will do in the following season. I especially like the use of the scoring tool to determine the teams which are best positioned to win the following year. It would be an interesting study to take historical data, apply this approach and see how accurate the results were. I'd like to do the same thing for football.

**Topic 12: Identifying Data Fields in Sloppy Data**

This is example 20 and I like it a lot because of how regex parsing is used to identify different data type elements like addresses, phone numbers, etc. The buckets are created to hold these fields and I think the approach is novel and robust. There are many real-work examples that could use this approach.


alteryx | COMMUNITY
Weekly Challenge #20
A sales executive got a series of leads from the DMA conference. Unfortunately it's a text file that we want to restructure into a tabular form to load into Salesforce.
The goal is to restructure the data accordingly.
Input
DMA_List.txt
Output
Remove unnecessary rows
Identify the field header for each row using RegEx and other logic
Add record ID's
Pivot the data
Split the address and Phone numbers from the address field
Catch phone numbers that have the "1-" in front of them
update Address and Phone Number fields with the parsed address/phone numbers
Remove unwanted fields
Output

The incoming data looks like this:

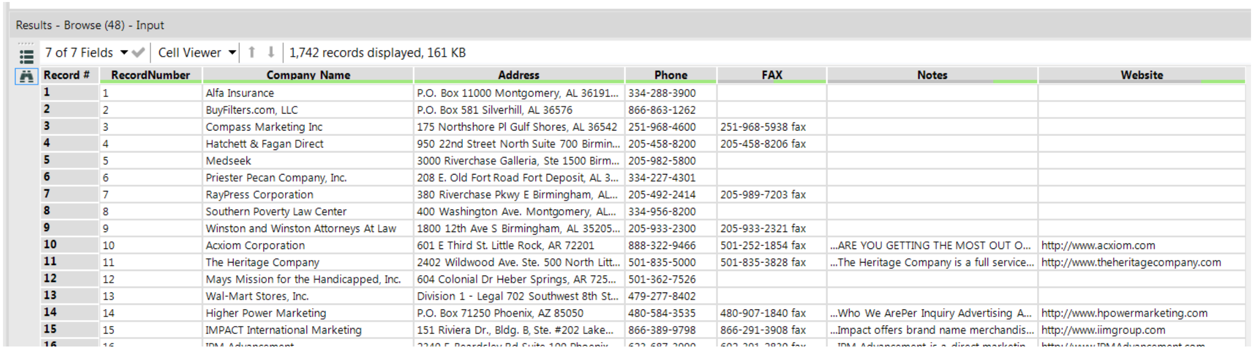


_externals\1\DMA_List.txt

| | Name | Value |
|---|---|---|
| 1 | Record Limit | |
| 2 | File Format | Comma-Delimited Text Files (*.csv) |
| 3 | Search SubDirs | ☐ |
| 4 | Output File Name as Field | No |
| 5 | Delimiters | \0 |
| 6 | First Row Contains Field Names | ☐ |

| | Field_1 |
|---|---|
| 1 | Alfa Insurance |
| 2 | [Null] |
| 3 | P.O. Box 11000 Montgomery, AL 36191-0001  334-288-3900 |
| 4 | |
| 5 | |
| 6 | [Null] |
| 7 | BuyFilters.com, LLC |
| 8 | [Null] |
| 9 | P.O. Box 581 Silverhill, AL 36576  866-863-1262 |
| 10 | |
| 11 | |
| 12 | [Null] |
| 13 | Compass Marketing Inc |
| 14 | [Null] |
| 15 | 175 Northshore Pl Gulf Shores, AL 36542  251-968-4600 |
| 16 | 251-968-5938 fax |

Once the cleaning and parsing is complete, a nice output structure is achieved:

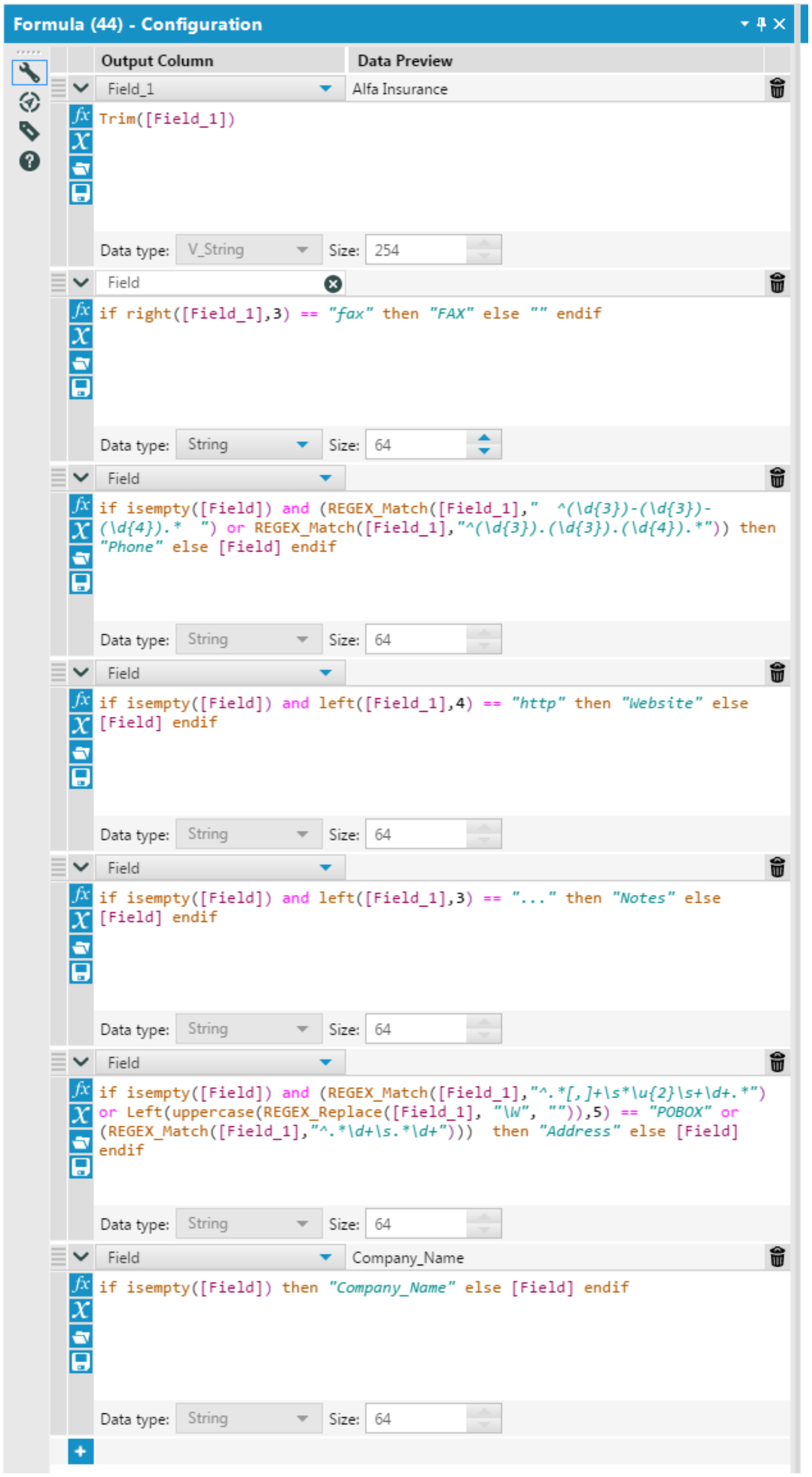


Results - Browse (48) - Input

7 of 7 Fields | Cell Viewer | 1,742 records displayed, 161 KB

| Record # | RecordNumber | Company Name | Address | Phone | FAX | Notes | Website |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Alfa Insurance | P.O. Box 11000 Montgomery, AL 36191... | 334-288-3900 | | | |
| 2 | 2 | BuyFilters.com, LLC | P.O. Box 581 Silverhill, AL 36576 | 866-863-1262 | | | |
| 3 | 3 | Compass Marketing Inc | 175 Northshore Pl Gulf Shores, AL 36542 | 251-968-4600 | 251-968-5938 fax | | |
| 4 | 4 | Hatchett & Fagan Direct | 950 22nd Street North Suite 700 Birmin... | 205-458-8200 | 205-458-8206 fax | | |
| 5 | 5 | Medseek | 3000 Riverchase Galleria, Ste 1500 Birm... | 205-982-5800 | | | |
| 6 | 6 | Priester Pecan Company, Inc. | 208 E. Old Fort Road Fort Deposit, AL 3... | 334-227-4301 | | | |
| 7 | 7 | RayPress Corporation | 380 Riverchase Pkwy E Birmingham, AL... | 205-492-2414 | 205-989-7203 fax | | |
| 8 | 8 | Southern Poverty Law Center | 400 Washington Ave. Montgomery, AL... | 334-956-8200 | | | |
| 9 | 9 | Winston and Winston Attorneys At Law | 1800 12th Ave S Birmingham, AL 35205... | 205-933-2300 | 205-933-2321 fax | | |
| 10 | 10 | Acxiom Corporation | 601 E Third St. Little Rock, AR 72201 | 888-322-9466 | 501-252-1854 fax | ...ARE YOU GETTING THE MOST OUT O... | http://www.acxiom.com |
| 11 | 11 | The Heritage Company | 2402 Wildwood Ave. Ste. 500 North Litt... | 501-835-5000 | 501-835-3828 fax | ...The Heritage Company is a full service... | http://www.theheritagecompany.com |
| 12 | 12 | Mays Mission for the Handicapped, Inc. | 604 Colonial Dr Heber Springs, AR 725... | 501-362-7526 | | | |
| 13 | 13 | Wal-Mart Stores, Inc. | Division 1 - Legal 702 Southwest 8th St... | 479-277-8402 | | | |
| 14 | 14 | Higher Power Marketing | P.O. Box 71250 Phoenix, AZ 85050 | 480-584-3535 | 480-907-1840 fax | ...Who We ArePer Inquiry Advertising A... | http://www.hpowermarketing.com |
| 15 | 15 | IMPACT International Marketing | 151 Riviera Dr., Bldg. B, Ste. #202 Lake... | 866-389-9798 | 866-291-3908 fax | ...Impact offers brand name merchandis... | http://www.iimgroup.com |
| 16 | 16 | IPM Advancement | 2340 E Beardsley Rd Suite 100 Phoenix... | 623-687-2000 | 602-291-2830 fax | ...IPM Advancement is a direct marketin... | http://www.IPMAdvancement.com |

Here are the details of how the data fields are identified: (Awesome regex examples)



Field_1 — Alfa Insurance

```
Trim([Field_1])
```

Data type: V_String Size: 254

Field

```
if right([Field_1],3) == "fax" then "FAX" else "" endif
```

Data type: String Size: 64

Field

```
if isempty([Field]) and (REGEX_Match([Field_1],"  ^(\d{3})-(\d{3})-
(\d{4}).*  ") or REGEX_Match([Field_1],"^(\d{3}).(\d{3}).(\d{4}).*")) then
"Phone" else [Field] endif
```

Data type: String Size: 64

Field

```
if isempty([Field]) and left([Field_1],4) == "http" then "Website" else
[Field] endif
```

Data type: String Size: 64

Field

```
if isempty([Field]) and left([Field_1],3) == "..." then "Notes" else
[Field] endif
```

Data type: String Size: 64

Field

```
if isempty([Field]) and (REGEX_Match([Field_1],"^.*[,]+\s*\u{2}\s+\d+.*")
or Left(uppercase(REGEX_Replace([Field_1], "\W", "")),5) == "POBOX" or
(REGEX_Match([Field_1],"^.*\d+\s.*\d+")))  then "Address" else [Field]
endif
```

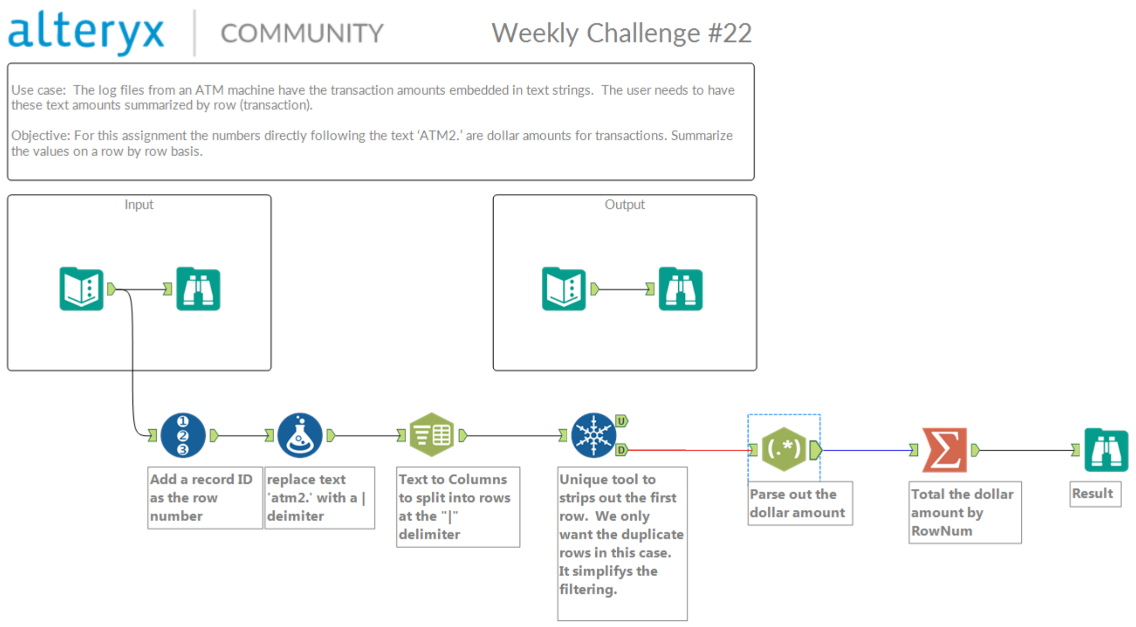
Data type: String Size: 64

Field — Company_Name

```
if isempty([Field]) then "Company_Name" else [Field] endif
```

Data type: String Size: 64

Continuing with the theme of sloppy data, Week 22 has ATM data in a really ugly format and the dollar transactions need to be extracted. This is another nice regex example. Here is the workflow:

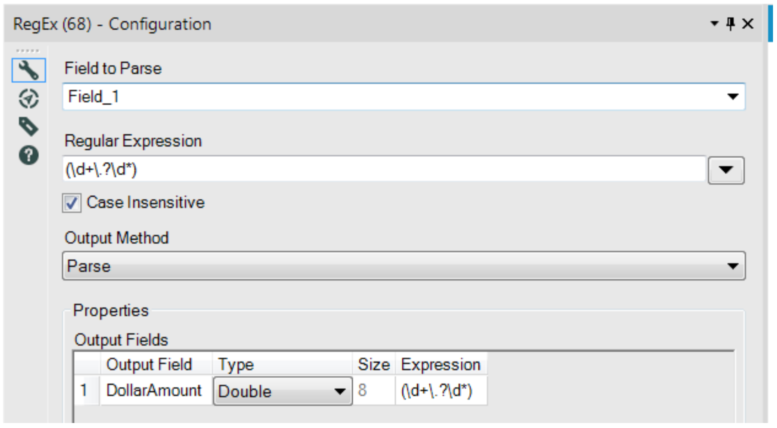

alteryx | COMMUNITY Weekly Challenge #22

Use case: The log files from an ATM machine have the transaction amounts embedded in text strings. The user needs to have these text amounts summarized by row (transaction).

Objective: For this assignment the numbers directly following the text 'ATM2.' are dollar amounts for transactions. Summarize the values on a row by row basis.

Input
Output
Add a record ID as the row number
replace text 'atm2.' with a | deimiter
Text to Columns to split into rows at the "|" delimiter
Unique tool to strips out the first row. We only want the duplicate rows in this case. It simplifys the filtering.
Parse out the dollar amount
Total the dollar amount by RowNum
Result

Here is **the regex for extracting the dollar values of the transactions**:


RegEx (68) - Configuration

Field to Parse: Field_1

Regular Expression: (\d+\.?\d*)

[✓] Case Insensitive

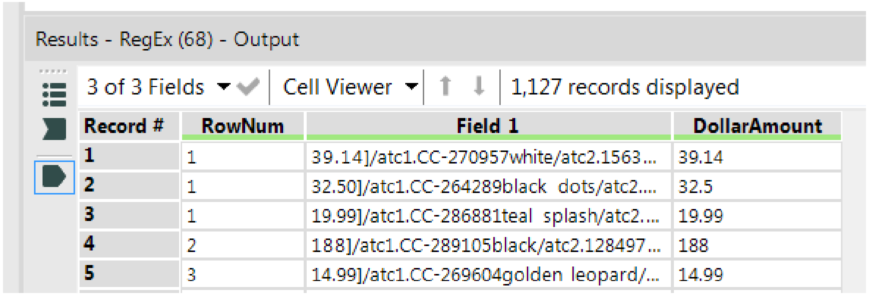
Output Method: Parse

Properties

Output Fields

| | Output Field | Type | Size | Expression |
|---|---|---|---|---|
| 1 | DollarAmount | Double | 8 | (\d+\.?\d*) |

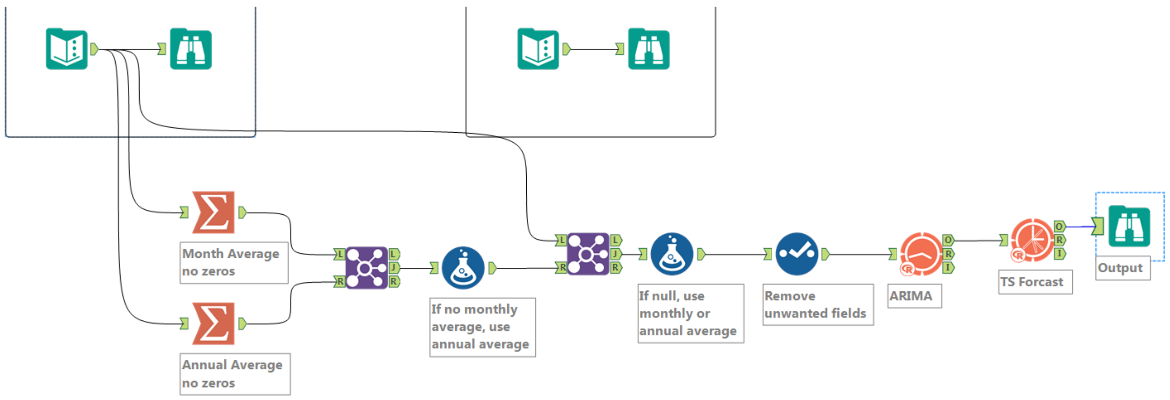
Here is the result:

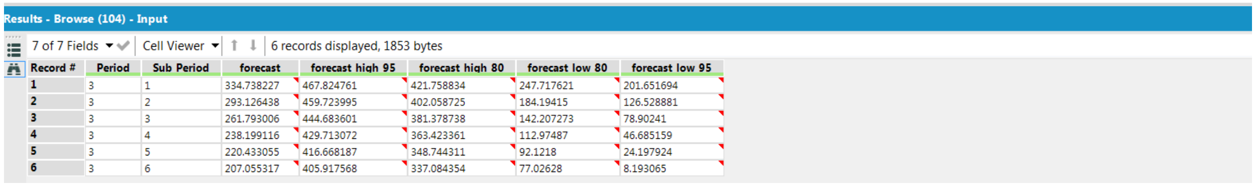

Results - RegEx (68) - Output

3 of 3 Fields | Cell Viewer | 1,127 records displayed

| Record # | RowNum | Field 1 | DollarAmount |
|---|---|---|---|
| 1 | 1 | 39.14]/atc1.CC-270957white/atc2.1563... | 39.14 |
| 2 | 1 | 32.50]/atc1.CC-264289black dots/atc2.... | 32.5 |
| 3 | 1 | 19.99]/atc1.CC-286881teal splash/atc2.... | 19.99 |
| 4 | 2 | 188]/atc1.CC-289105black/atc2.128497... | 188 |
| 5 | 3 | 14.99]/atc1.CC-269604golden leopard/... | 14.99 |

**Topic 13: Time Series Forecasting Using An autoregressive integrated moving average (ARIMA) model**




Results - Browse (104) - Input

7 of 7 Fields | Cell Viewer | 6 records displayed, 1853 bytes

| Record # | Period | Sub Period | forecast | forecast high 95 | forecast high 80 | forecast low 80 | forecast low 95 |
|---|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 334.738227 | 467.824761 | 421.758834 | 247.717621 | 201.651694 |
| 2 | 3 | 2 | 293.126438 | 459.723995 | 402.058725 | 184.19415 | 126.528881 |
| 3 | 3 | 3 | 261.793006 | 444.683601 | 381.378738 | 142.207273 | 78.90241 |
| 4 | 3 | 4 | 238.199116 | 429.713072 | 363.423361 | 112.97487 | 46.685159 |
| 5 | 3 | 5 | 220.433055 | 416.668187 | 348.744311 | 92.1218 | 24.197924 |
| 6 | 3 | 6 | 207.055317 | 405.917568 | 337.084354 | 77.02628 | 8.193065 |

I really like this example for a few different reasons. Using Alteryx to make predictions is a very practical usage of the software. I especially like the forecasting at 95% and 80% high and low.