dun & bradstreet

Market Insight Expressions User Guide v 1.0

D&B Market Insight User Guide

Expressions

Manual Version:	1.0
Software Version:	2017 Q4
System:	Training (UK)

Contents

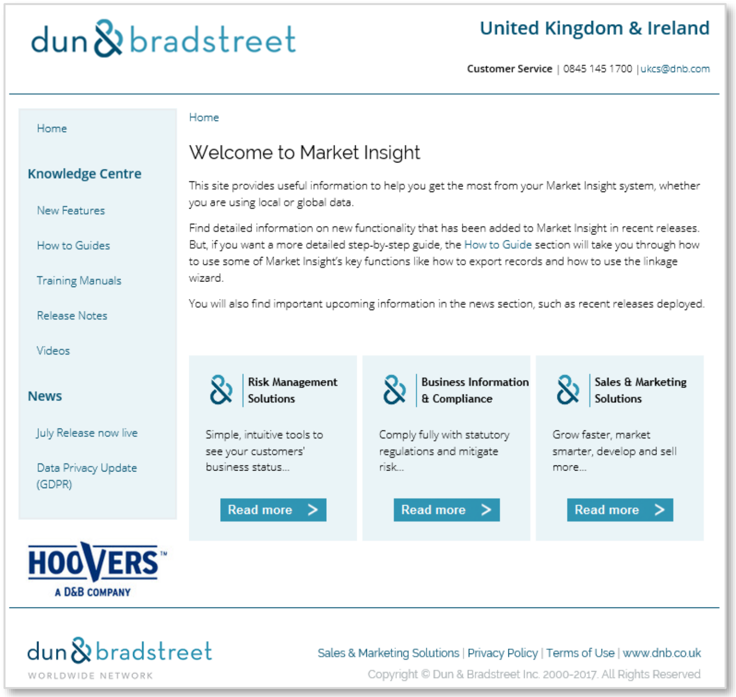
Introduction

Market Insight provides powerful and interactive marketing analysis of customer data overlaid on a D&B data universe. The system is web based with a truly easy to use Windows interface. Using a consistent and intuitive "drag and drop" approach throughout, every action automatically results in a query that can be saved and reused with ease. With a wide range of descriptive and predictive analytical tools, Market Insight's analysis options are virtually unlimited as any technique can be applied to any results in any order. Market Insight provides a unique combination of speed, power and accessibility for data exploration and understanding.

Market Insight holds your data overlaid on a D&B universe. This enables you to accurately measure your customer data in proportion to the opportunities in the market place. Hence the product's name: it enables insight of your activities in comparison to the market place rather than just within your business.

The D&B data universe in your Market Insight system will be adjusted to suit your licensing and measurement requirements. Your customer data is loaded from extract file(s) you provide and although this process allows for some cleaning and manipulation of the data, what you see within Market Insight is a reflection of the data you provide.

The Market Insight view of the data is a snapshot at the time that the data was loaded. Market Insight is an analytical system able to provide insight and understanding but it can also provide data feeds to your operational marketing systems to implement your targeting decisions.


United Kingdom & Ireland
Customer Service | 0845 145 1700 | ukcs@dnb.com
Home
Knowledge Centre
New Features
How to Guides
Training Manuals
Release Notes
Videos
News
July Release now live
Data Privacy Update (GDPR)
Home
Risk Management Solutions
Business Information & Compliance
Sales & Marketing Solutions
Read more
Sales & Marketing Solutions | Privacy Policy | Terms of Use | www.dnb.co.uk
Copyright © Dun & Bradstreet Inc. 2000-2017. All Rights Reserved

Welcome to Market Insight

This site provides useful information to help you get the most from your Market Insight system, whether you are using local or global data.

Find detailed information on new functionality that has been added to Market Insight in recent releases. But, if you want a more detailed step-by-step guide, the How to Guide section will take you through how to use some of Market Insight's key functions like how to export records and how to use the linkage wizard.

You will also find important upcoming information in the news section, such as recent releases deployed.

Simple, intuitive tools to see your customers' business status...

Comply fully with statutory regulations and mitigate risk...

Grow faster, market smarter, develop and sell more...

Market Insight Splash Screen – D&B Website

N.B. The counts and figures in this manual may differ to those seen when you use this system as the data changes over time. Not all the functionality shown in this manual may be available in the system you are using.

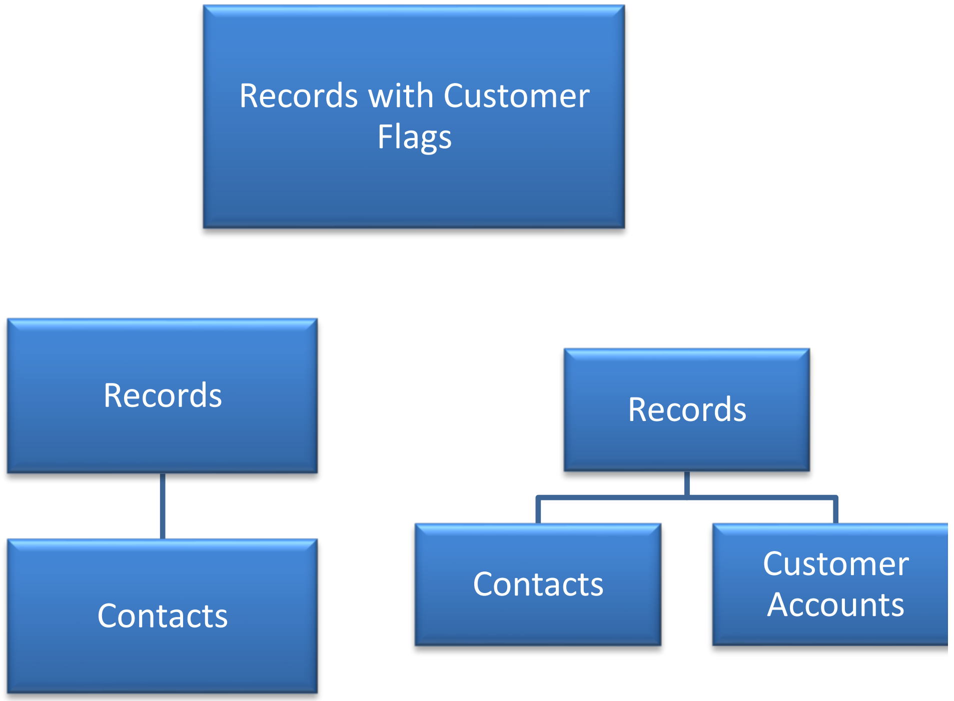
N.B. Where suitable variables were not available in the Market Insight Training system a holiday companies database has been used to provide the examples.

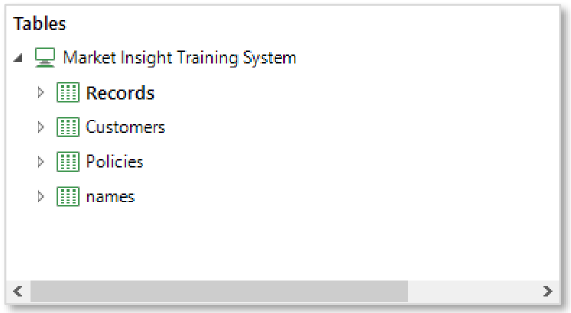
Data Structure

The structure of your Market Insight system can vary. The elements shown here are typical – each Record may be simply flagged with Customer data or can have many related Names. A Record may also have many matched Customer Accounts. The data loaded for each matched Customer Account is configurable – for example you may have multiple Transactions or Divisional Summaries or Product Summaries etc.

The detail present on each table of data depends on the Market Insight administrator. The data is arranged into folders to assist the user to navigate and find data items.

The structure used in the Training System, illustrated in this manual, uses a simple structure that has Records (organisations) with Names (contacts at the organisation). Also a subset of the Records called Customers (the Users customers) is held with a related table Policies (activity of the Users customers).





Accessing Market Insight

The Market Insight software is downloaded automatically to your PC when you click a link to launch the system. Once the software has been downloaded, it will automatically update from the server whenever necessary. You will normally receive a welcome email with details of this process.

To access Market Insight you need:

- Windows PC – Market Insight is a Windows.NET application that combines the best of the Windows interface with web based systems. Market Insight is not available on Mac or UNIX computers

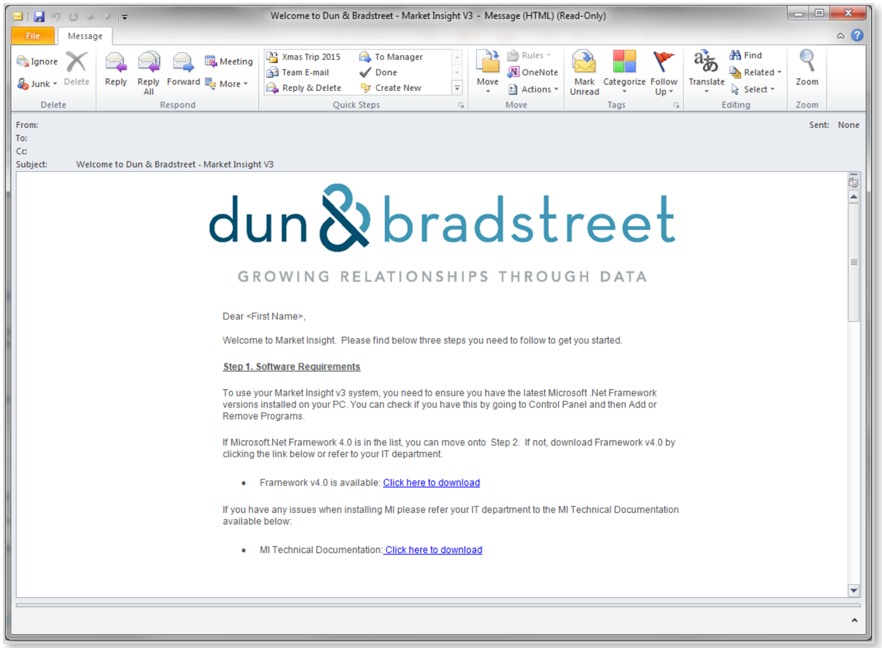
- The latest Windows.NET framework version installed. This can be obtained by visiting www.windowsupdate.com or from your IT team

To launch your Market Insight system, use a browser to view:

https://www.dnbmi.com/disco_systems/v3/new/milauncher.msi

Alternatively use the links within your welcome email.

N.B. The "https" prefix, which establishes a secure connection between your browser and the D&B Server.



From:
To:
Cc:
Subject: Welcome to Dun & Bradstreet - Market Insight V3
Sent: None

GROWING RELATIONSHIPS THROUGH DATA

Dear <First Name>,

Welcome to Market Insight. Please find below three steps you need to follow to get you started.

Step 1. Software Requirements

To use your Market Insight v3 system, you need to ensure you have the latest Microsoft .Net Framework versions installed on your PC. You can check if you have this by going to Control Panel and then Add or Remove Programs.

If Microsoft.Net Framework 4.0 is in the list, you can move onto Step 2. If not, download Framework v4.0 by clicking the link below or refer to your IT department.

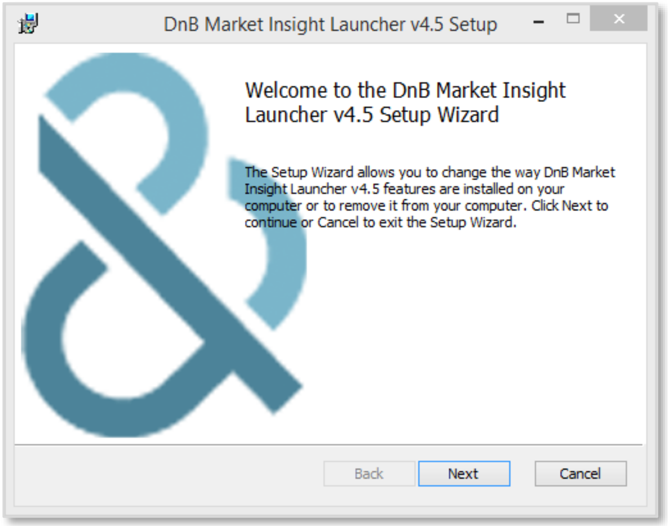
- Framework v4.0 is available: Click here to download

If you have any issues when installing MI please refer your IT department to the MI Technical Documentation available below:

- MI Technical Documentation: Click here to download

Welcome to D&B – Market Insight V3 Email

- Navigate to where you saved the downloaded file and double click it. Agree to run when prompted, and then follow the on screen instructions

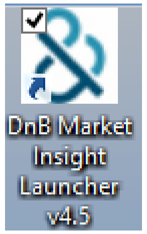


Welcome to the DnB Market Insight Launcher v4.5 Setup Wizard

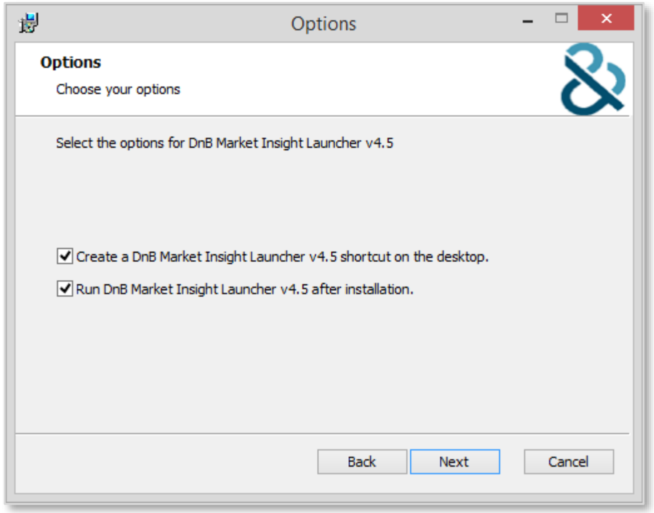
The Setup Wizard allows you to change the way DnB Market Insight Launcher v4.5 features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Launcher Setup Wizard

- The installation process will result in an icon on your desktop and in a D&B Start Menu folder



- On subsequent uses of Market Insight, you can simply double click this icon. The software will automatically update from the D&B server whenever new releases are made available



Options

Choose your options

Select the options for DnB Market Insight Launcher v4.5

☑ Create a DnB Market Insight Launcher v4.5 shortcut on the desktop.

☑ Run DnB Market Insight Launcher v4.5 after installation.

Options

- You can install Market Insight on as many computers as you wish – it is your user id that controls your access. This means, for example, you can use Market Insight when working from home

How to Login

To use Market Insight, you need to have an Internet connection.

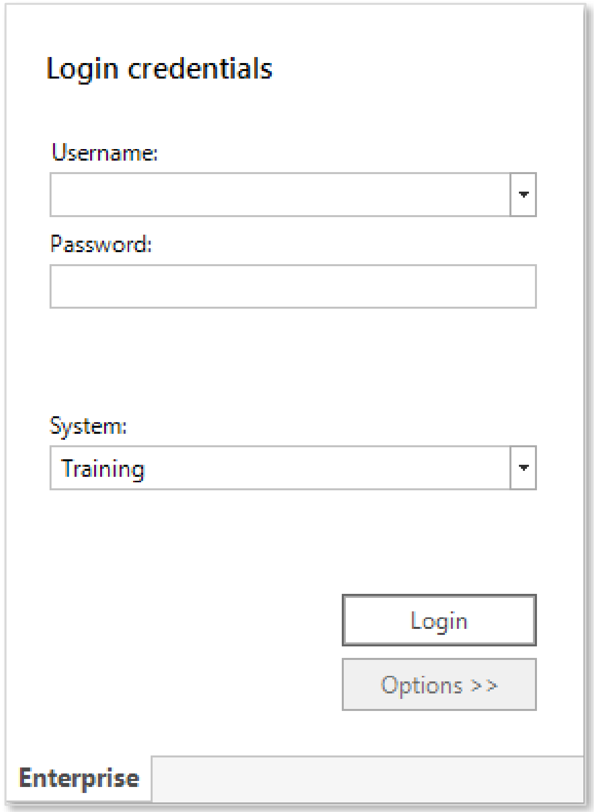
Start Market Insight by:

- Clicking on the **Market Insight** icon on your desktop, or by navigating to the program using Windows Explorer

In the upper left hand corner of the screen you will see a Login window that gives you the opportunity to connect to a Market Insight system containing data available to you for analysis.

Enterprise Tab

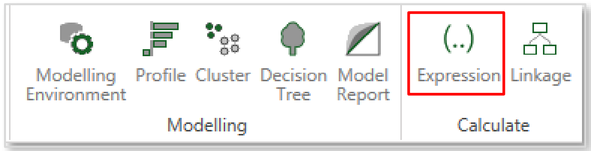
Your Market Insight system operates on a secure and resilient web connected server enabling you to access the system from any location with an Internet connection. A number of users may access the system at the same time, each of whom is authorised by a user account and password. Your Market Insight Administrator will provide you with a Username and Password.



Login Window

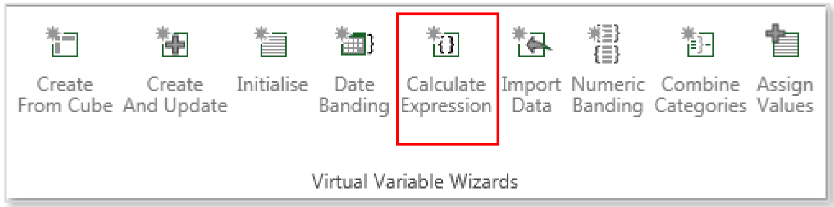
Introduction

Expressions are a powerful tool within Market Insight that allows the end user to significantly enhance the way in which they can manipulate the data. Multiple functions are available for building expressions. Expressions can use constants, mathematical, logical and date functions, FastStats variables and FastStats Queries as elements of an expression.



Expressions are currently used in four places in Market Insight:

- Dragged from the Expression tool onto a Cube as a cube statistic
- Dragged from the Expression tool onto a Data Grid as an output column
- Dragged from the Expression tool onto a Selection and used like a variable
- In the Calculate Expression wizard to populate a new variable according to a mathematical expression or logical rule
- Dragged from the Expression tool onto a Cube as a cube dimension



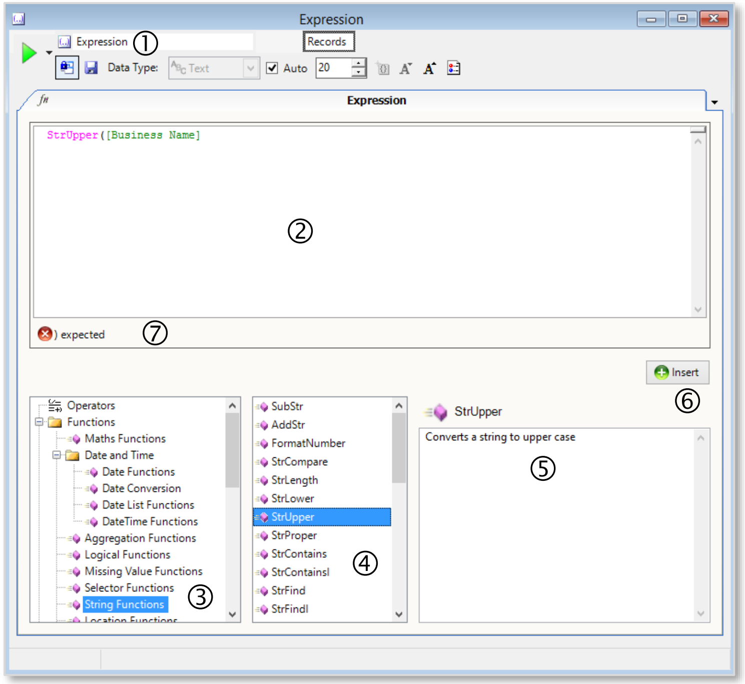
Expressions may be saved and edited independently and are automatically saved within the tools they are used on.

This document aims to guide you through how to use the Expression window and give some examples of how the various functions can be used.

The Expression Window

The Expression tool can be found under the Calculate section of the Toolbox ribbon bar. The components of this window are as described below:

1. Set the table level and data type relating to your expression e.g. The variable Business Name is used in the example opposite and was created as a text variable at the Records table
2. The expression builder area where the expression is created
3. The section containing the different groups of functions
4. The section containing a list of functions for a selected group
5. The section containing a brief description of the function selected
6. The button that will insert the function selected into the expression builder window
7. The prompt bar will suggest what might come next in the expression and will indicate when a valid expression has been created



```
StrUpper([Business Name]
```

) expected

Converts a string to upper case

N.B. The icon opposite allows the user to convert certain expressions in to selector Variables via a Wizard.

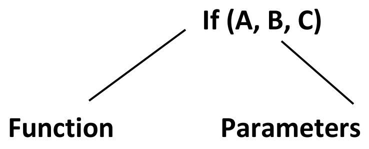
Components of an Expression

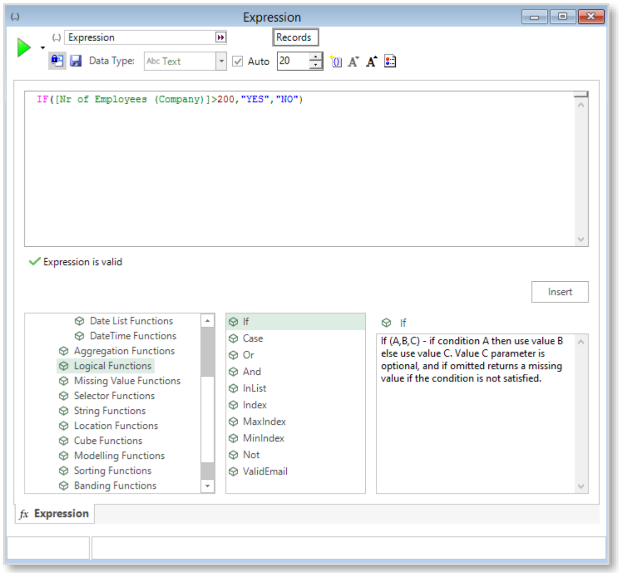
As an example take the If logic function to explore the makeup of an expression.

- ➢ Drag the **Expression** tool from the **Toolbox** ribbon bar onto the workspace

This window allows you to build an Expression as shown in the screen shot opposite.

The breakdown of this Expression is as follows:





Data Type: Abc Text ✓ Auto 20

```
IF([Nr of Employees (Company)]>200,"YES","NO")
```

✓ Expression is valid

If

If (A,B,C) - if condition A then use value B else use value C. Value C parameter is optional, and if omitted returns a missing value if the condition is not satisfied.

You could read this as:

If (A [condition]

then B [outcome if condition met]

else C [outcome if condition not met])

Here a Condition is the test between two values. Those values can result from a Variable, Numeric, String, Date or another Expression.

The example opposite has used a test on a variable (the Nr of Employees company is greater than 200) and if that test is met display the word Yes otherwise display the word No. This can be seen when used on a Data Grid.

As you build an Expression you need to be aware of the different elements and how they are constructed.

Functions
Pink in colour and inserted with an open round bracket

Variables
Green in colour and inserted with enclosed square brackets

Numbers
Red in colour

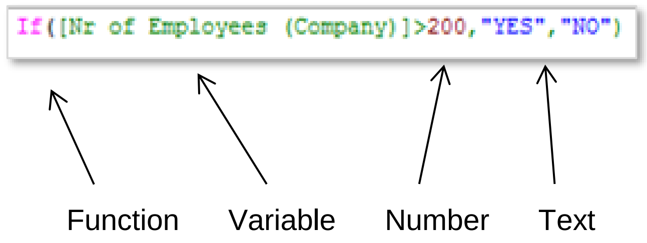
Text (String)
Blue in colour and enclosed in speech marks

Selection Query
Lime green in colour inserted with the query number and name within curly brackets (Brace {})

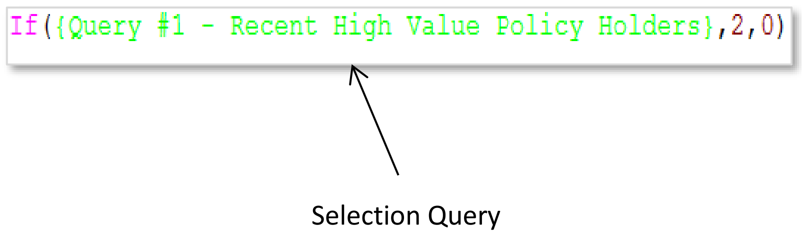
Operators
Arithmetical operators like +, - etc.

Cube
Some functions can reference a cube e.g. CubeLookup()

N.B. In most cases you will separate each element of your Expression with a comma. Also remember to ensure for every open bracket in your Expression you have an equivalent closed bracket. Errors will be underlined by a red wavy line.



```
If([Nr of Employees (Company)]>200,"YES","NO")
```



```
If({Query #1 - Recent High Value Policy Holders},2,0)
```

Creating a Conditional Expression

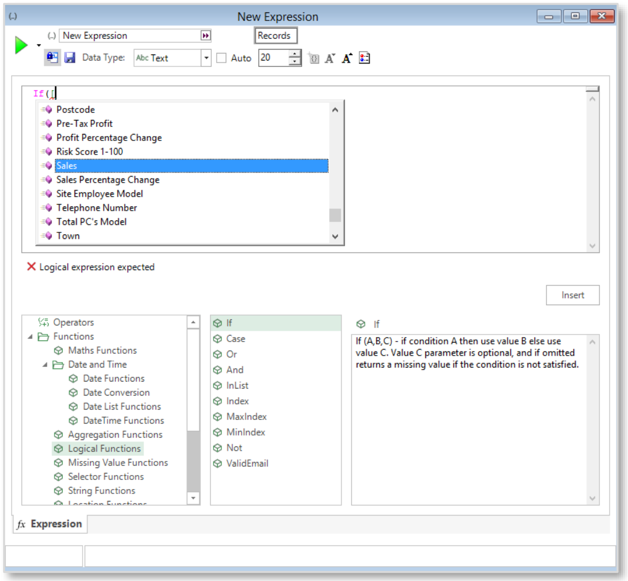
- Ensure the **Table** is set to **Records** and the **Data Type** to **Text**
- Select the **If** statement from the **Logical Functions** option in the **Functions** folder and then click the **Insert** button
- Next to the open black bracket type a left hand square bracket. This will display the available variables that can be used with this function
- Double click on the **Sales** variable. Alternatively you could have typed the variable name within square brackets to obtain the same result

The next part of the Expression is to create a test to find values greater than 2000, therefore you need to insert an Inequality Function.

- From the **Functions** button select **Inequality Functions** and the **> Greater Than** option. Click **Insert**. You may find it quicker to type the symbol directly after the variable
- Type **200000** followed by a comma

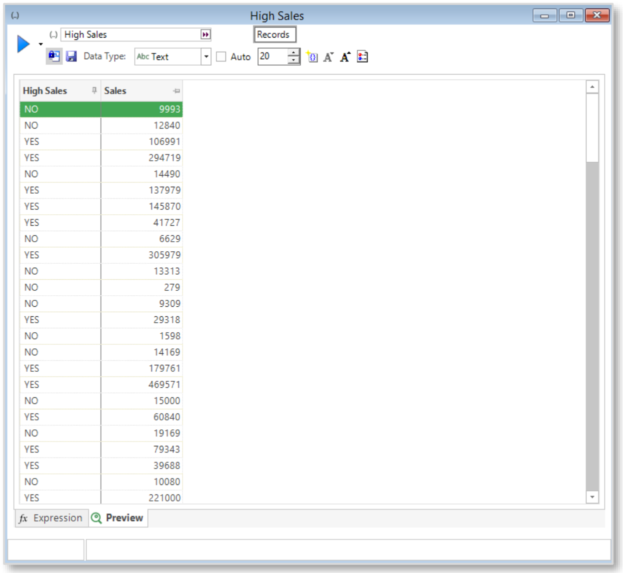
The next part of the Expression is to determine the output when the condition is met and when it is not met. As this example is outputting a word (String) in each case you need to ensure they are enclosed in double quotes.

- Type **"Yes","No"** followed by a closing bracket **)**
- Name the Expression window **High Sales**
- Press the **Build** button to see a **Preview** of the results. This can now be used in conjunction with the **Data Grid** tool



Logical expression expected

If (A,B,C) - if condition A then use value B else use value C. Value C parameter is optional, and if omitted returns a missing value if the condition is not satisfied.



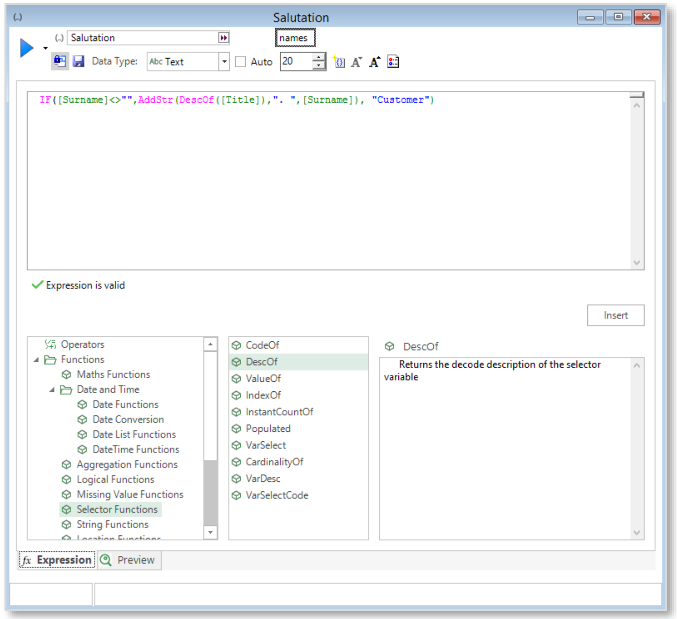
High Sales	Sales
NO	9993
NO	12840
YES	106991
YES	294719
NO	14490
YES	137979
YES	145870
YES	41727
NO	6629
YES	305979
NO	13313
NO	279
NO	9309
YES	29318
NO	1598
NO	14169
YES	179761
YES	469571
NO	15000
YES	60840
NO	19169
YES	79343
YES	39688
NO	10080
YES	221000

Creating a Salutation Expression

Building upon the previous example you will now see how you can use variable data as part of the output and also combine the results in the display.

Here you will test to see if your records hold a customer's name in preparation for a mail shot. If a record does hold a name the letter will start *Dear* [Title]. [Surname] e.g. Mr. Smith. Otherwise the letter will start *Dear* Customer.

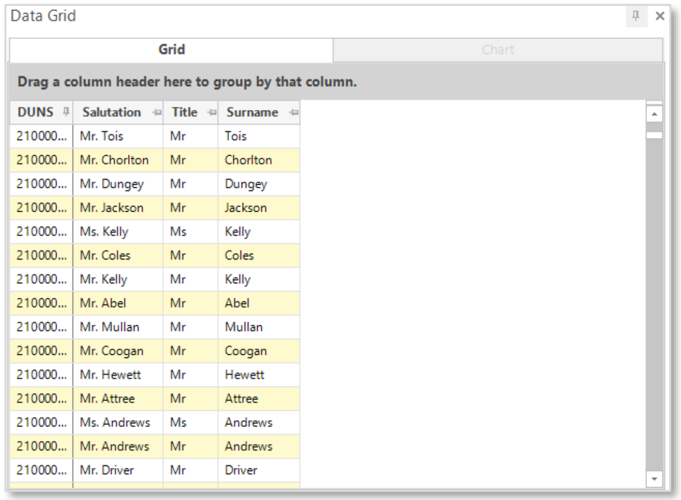
- Open an **Expression** window and set it to the **names** table.
- Select the **If(** function, as in the last example
- Change the **Data Type** to **Text**
- Drag the **Surname** variable on after the bracket and then type **<> "",** to test if there is a text value
- From the **Strings Function Category** of the **Function** window select **AddStr(**
- From the **Selector Functions Category** Insert **DescOf(**
- Insert after the bracket **[Title], ". ", [Surname]** which will display e.g. Mr. Smith if a name is present
- Type a **,** after the bracket and then **"Customer"))**. This will then display the word Customer if no name is present
- Click the **Build** button to see a **Preview** or view on a data grid



```
IF([Surname]<>"",AddStr(DescOf([Title]),". ",[Surname]), "Customer")
```

Expression is valid

Returns the decode description of the selector variable

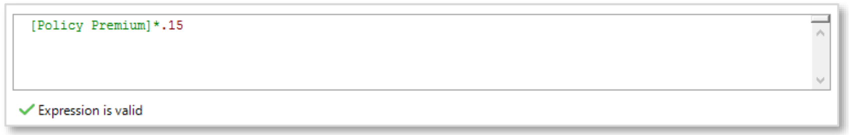


Drag a column header here to group by that column.

DUNS	Salutation	Title	Surname
210000...	Mr. Tois	Mr	Tois
210000...	Mr. Chorlton	Mr	Chorlton
210000...	Mr. Dungey	Mr	Dungey
210000...	Mr. Jackson	Mr	Jackson
210000...	Ms. Kelly	Ms	Kelly
210000...	Mr. Coles	Mr	Coles
210000...	Mr. Kelly	Mr	Kelly
210000...	Mr. Abel	Mr	Abel
210000...	Mr. Mullan	Mr	Mullan
210000...	Mr. Coogan	Mr	Coogan
210000...	Mr. Hewett	Mr	Hewett
210000...	Mr. Attree	Mr	Attree
210000...	Ms. Andrews	Ms	Andrews
210000...	Mr. Andrews	Mr	Andrews
210000...	Mr. Driver	Mr	Driver

Expressions and Cubes

In this example an Expression will be used as a statistic on a Cube display. The Expression itself will calculate profit based upon 15% of the **Policy Premium**.

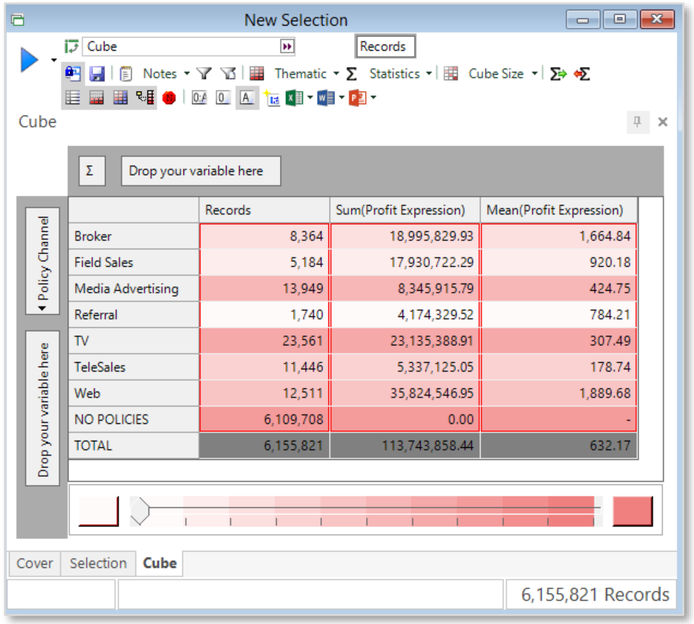


```
[Policy Premium]*.15
```

- Open an **Expression** window
- Ensure the **Table** is set to **policies**
- Drag on the **Policy Premium** variable
- Type ***0.15** which will multiply the Policy Premium by 0.15 to calculate 15% of the spend
- Name the expression Profit

To use this expression to help find the profit per Channel:

- Drag on a **Cube** and set the vertical dimension to **Policy Channel**
- Left drag your **Profit Expression** onto the centre of the **Cube** and the right drag your **Profit Expression** onto the centre of the **Cube** and chose **mean**
- Press the **Build** button to build the display



	Records	Sum(Profit Expression)	Mean(Profit Expression)
Broker	8,364	18,995,829.93	1,664.84
Field Sales	5,184	17,930,722.29	920.18
Media Advertising	13,949	8,345,915.79	424.75
Referral	1,740	4,174,329.52	784.21
TV	23,561	23,135,388.91	307.49
TeleSales	11,446	5,337,125.05	178.74
Web	12,511	35,824,546.95	1,889.68
NO POLICIES	6,109,708	0.00	-
TOTAL	6,155,821	113,743,858.44	632.17

The display now shows the number of Records in each Policy Channel. Also it shows the Sum and Mean profit made through each Channel.

Expressions and Selection Queries

In the following example selection queries will be used in the creation of an expression and the result then can be used to form part of a new selection query. The aim of this scenario is to give a score to a Record depending on their spend on the Policies they have with the company. Some policies will be of more value than others.

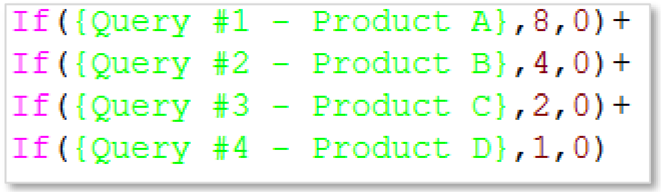
- Create and save **4** selections identifying people who have **Policy A, Policy B, Policy C and Policy D** respectively
- Open an **Expression** window and ensure it is set to **Records** and data type **Integer**. Name it **Policy Score Expression**
- Enter the **If** statement expression in the first screen shot opposite

This will add together the scores of these destinations e.g. Policy A scores 8, Policy A and B score 12.

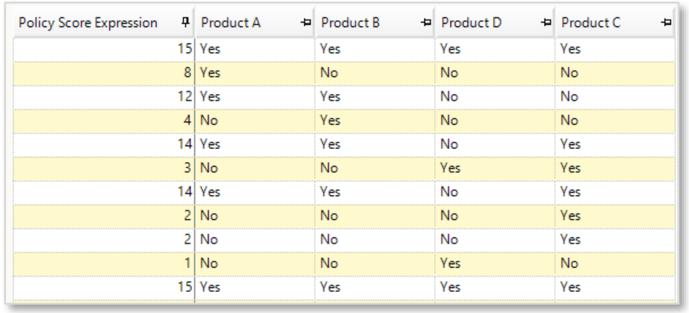
- Press the **Build** button to build the display and the Preview window will be populated as shown opposite (in this example a preview selection has been applied due to the sample size – see page 14)
- Drag the **Policy Score Expression** onto a blank selection window
- Type **>=5** to find Records with Policies whose total score meets the criteria

In this example it would select Records that have Policy A irrespective of if they hold any other policies. It would also select records with Policies B and C as well as records with policies B, C and D.

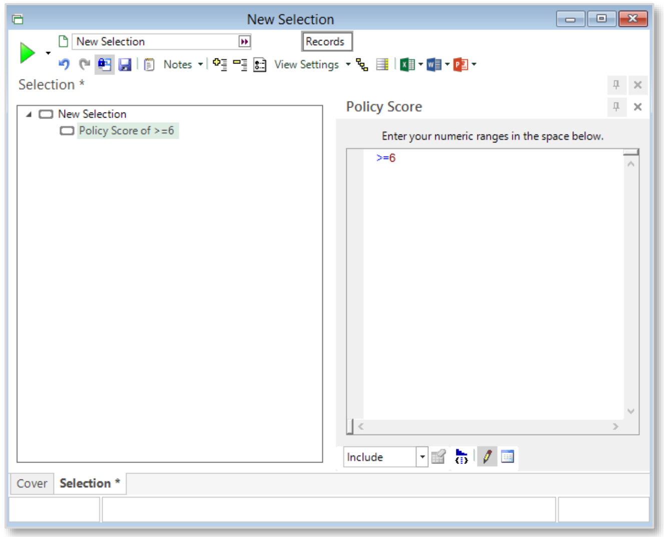
N.B. See CASE function as an alternative to achieve this example.



```
If({Query #1 – Product A},8,0)+
If({Query #2 – Product B},4,0)+
If({Query #3 – Product C},2,0)+
If({Query #4 – Product D},1,0)
```



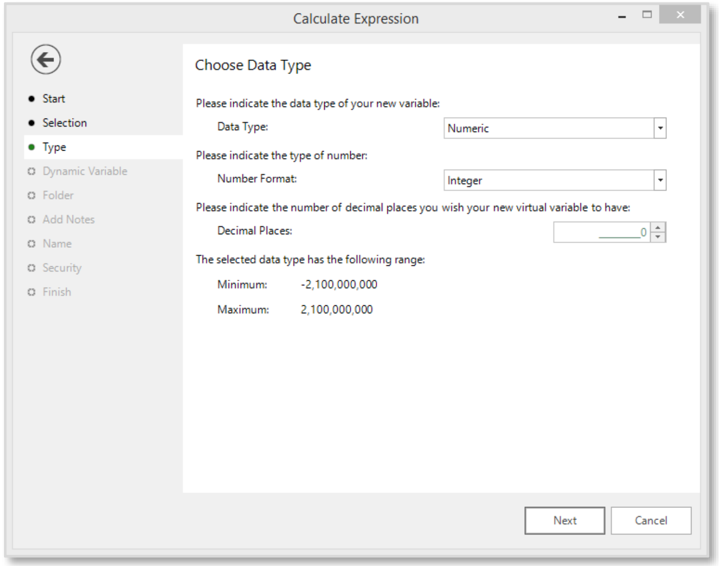
Policy Score Expression	Product A	Product B	Product D	Product C
15	Yes	Yes	Yes	Yes
8	Yes	No	No	No
12	Yes	Yes	No	No
4	No	Yes	No	No
14	Yes	Yes	No	Yes
3	No	No	Yes	Yes
14	Yes	Yes	No	Yes
2	No	No	No	Yes
2	No	No	No	Yes
1	No	No	Yes	No
15	Yes	Yes	Yes	Yes



Calculate Expression Wizard

Using this wizard a virtual variable can be generated from an expression. This example will use the policy Score Expression from the previous section.

- Open the **Calculate Expression** wizard by clicking on the appropriate tool in the Wizard bar or the icon on the **Expression** tool
- **Start** - Drag the **Policy Score Expression** onto the drop zone and click **Next** (not needed if used)
- **Selection** – Drag a selection onto the drop zone to define the records to be affected. Ignore this and click **Next**
- **Type** – Select the **Data Type** and **Format** to be used. Leave these on their defaults and click **Next**
- **Dynamic Variable** – Chose Variable type and click **Next**
- **Folder** – Leave the **Others** folder highlighted so it will receive the created virtual variable and click **Next**
- **Notes** – Enter any optional notes and click **Next**
- **Name** – Enter **Policy Score** as the description and click **Next**
- **Security** – This step is only visible if you are running an Enterprise system and you have ticked the **Modify Security Attributes** box in the previous step (Options for who and what others can see and do with your variable)
- **Finish** – Tick **Show new variable as a selection** and click **Finish**



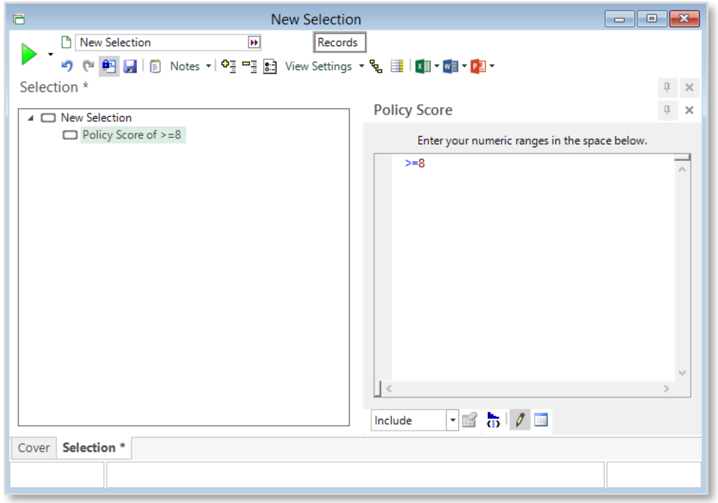
Choose Data Type

Please indicate the data type of your new variable:
Data Type: Numeric

Please indicate the type of number:
Number Format: Integer

Please indicate the number of decimal places you wish your new virtual variable to have:
Decimal Places: 0

The selected data type has the following range:
Minimum: -2,100,000,000
Maximum: 2,100,000,000



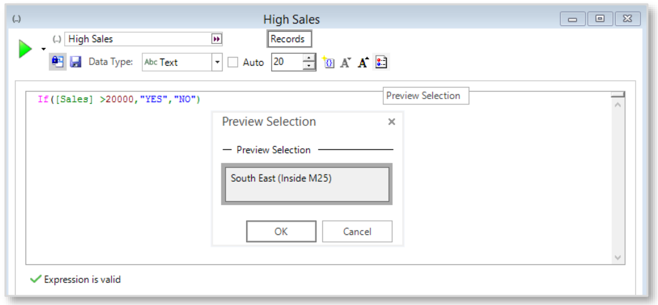
New Selection
Policy Score of >=8

Enter your numeric ranges in the space below.
>=8

Preview of Results

In the examples we have used so far the preview of the results are based upon the first 1000 records in the system. There may be some situations where you would like the preview to be based upon records that meet specific criteria.

Using the High Sales example (see pg. 9) it may be of interest to preview records that are specific to a certain Region.

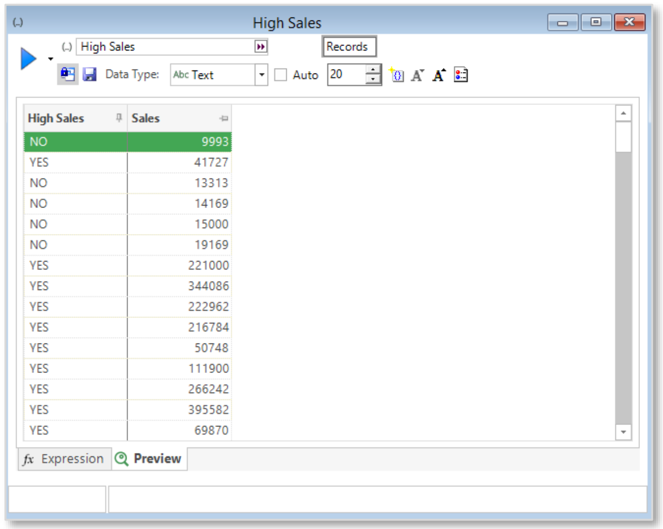


```
If([Sales] >20000,"YES","NO")
```

South East (Inside M25)

✓ Expression is valid

- ➢ Open a blank selection window at the **Records** table level
- ➢ Drag the **Economic Region** variable onto the selection window and select South East (Inside M25)
- ➢ Recreated the **High Sales** Expression as shown on pg. 9
- ➢ Click on the **Preview Selection** icon and drag the **Economic Region Selection** on to the drop box. Click **OK**
- ➢ Click on the **Build** button to see the new preview list of records



High Sales	Sales
NO	9993
YES	41727
NO	13313
NO	14169
NO	15000
NO	19169
YES	221000
YES	344086
YES	222962
YES	216784
YES	50748
YES	111900
YES	266242
YES	395582
YES	69870

Inserting Comments into an Expression

It may be useful to add comments within your expressions as a reminder or as an explanation as to how something is calculated.

This can be done by inserting symbols into the expression that are recognised as containing a comment.

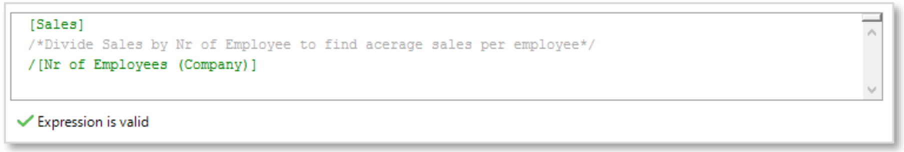
To start a comment you would first type **/*** add the comment and then close by typing ***/**

The example opposite is designed to find the average Sales per Employee:

> [Sales]/[Nr of Employees Company]

By using the symbols described above we can insert a comment in the middle of that expression to explain how this is calculated:

> /*Divide Sales by Nr of Employees Company to find average sales volume per employee*/



```
[Sales]
/*Divide Sales by Nr of Employee to find acerage sales per employee*/
/[Nr of Employees (Company)]
```

✓ Expression is valid

Functions

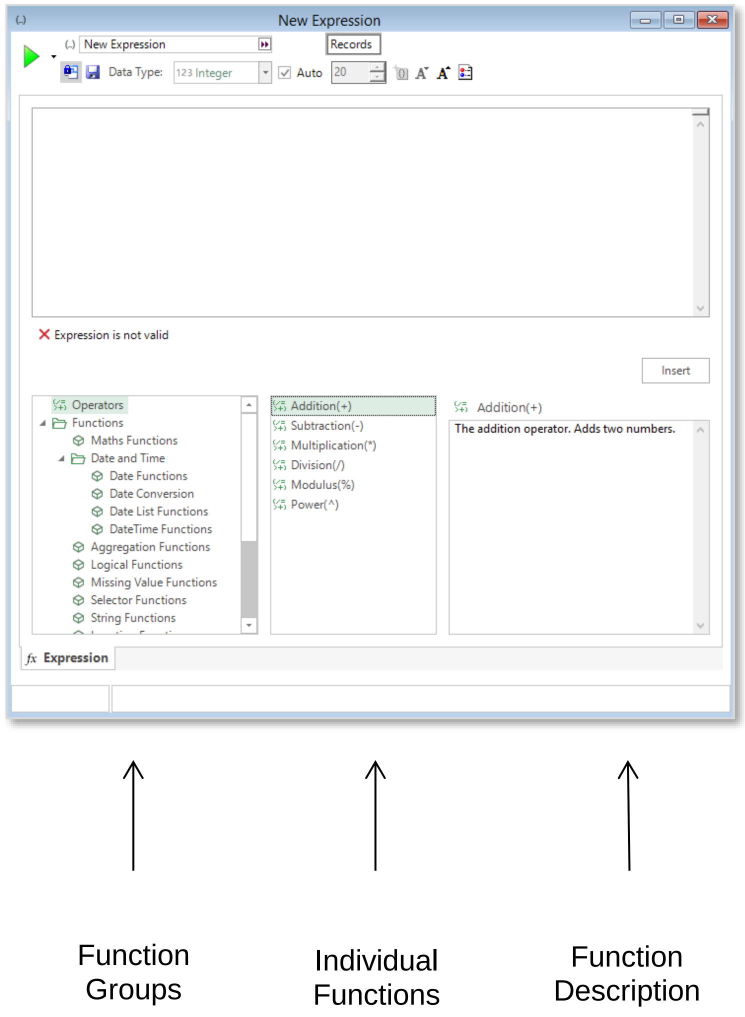
The following pages list the current available functions that can be used in an Expression. Each section gives the function name, the group it belongs to, a brief description and an example of its use.

In some cases examples only make sense when a function is used in conjunction with another function. To illustrate this further a number of short scenarios using various Expressions as their solution will follow this section. The appendices expand upon those functions with multiple options that cannot be described adequately in this section.

N.B. Please read the definitions in this section in conjunction with the description panel in the Expression window.

N.B. There are some gaps between the descriptions below to allow for future functions to be inserted.

N.B. The first section below are operators and inequalities that are used in the building of Expressions.


Expression is not valid

The addition operator. Adds two numbers.

Function:	Addition (+)	Group: Operators
Description:	Use this operator to add figures together. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] + [VAT] [Cost] + 100	
Function:	Multiplication (*)	Group: Operators
Description:	Use this operator to multiply figures together. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] * [Quantity] [Total] * 0.2	
Function:	Modulus (%)	Group: Operators
Description:	Use this operator to divide the value of one figure by the value of another to return the remainder. This could be the data held in a Numeric or Count variable.	
Example:	[Value] % [Cost] [Value] % 3	

Function:	Subtraction (-)	Group: Operators
Description:	Use this operator to subtract one figure from another. This could be the data held in a Numeric or Count variable.	
Example:	[Total] – [Discount] [Total] - 100	
Function:	Division (/)	Group: Operators
Description:	Use this operator to divide figures. This could be the data held in a Numeric or Count variable.	
Example:	[Total Cost] / [No. of Customers] [Cost] / 5	
Function:	Power (^)	Group: Operators
Description:	Use this operator to calculate power of one figure by another e.g. a^b where the result is the bth power of a.	
Example:	[Value] ^ [Numeric] 5 ^ 3 (5 x 5 x 5 = 125)	

Function:	Greater than (>)	Group: Inequality Operators
Description:	Use this operator to test if one figure is greater than another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] > 1000 [Cost] > [Revenue]	
Function:	GE (>=)	Group: Inequality Operators
Description:	Use this operator to test if one figure is greater than or equal to another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] >= 1000 [Cost] >= [Revenue]	
Function:	Equal to (=)	Group: Inequality Operators
Description:	Use this operator to test if one figure is equal to another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] = 1000 [Cost] = [Revenue]	

Function:	Less than (<)	Group: Inequality Operators
Description:	Use this operator to test if one figure is greater than another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] < 1000 [Cost] < [Revenue]	
Function:	LE (<=)	Group: Inequality Operators
Description:	Use this operator to test if one figure is less than or equal to another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] <= 1000 [Cost] <= [Revenue]	
Function:	Not equal to (<>)	Group: Inequality Operators
Description:	Use this operator to test if one figure is not equal to another. This could be the data held in a Numeric or Count variable.	
Example:	[Cost] <> 1000 [Cost] <> [Revenue]	

Function:	Log	Group: Maths Functions
Description:	Use this function to scale the value of figures where they cover a wide range. Uses Base 10.	
Example:	Log([Value]) Log(100) = 2 Log(1000) = 3	
Function:	Sqrt	Group: Maths Functions
Description:	Use this function to find the square root of a number. This could be the data held in a Numeric or Count variable.	
Example:	Sqrt([Value]) Sqrt(4) = 2 Sqrt(64) = 8	
Function:	Floor	Group: Maths Functions
Description:	Use this function to always round down a value to an integer. This could be the data held in a Numeric or Count variable.	
Example:	Floor([Value]) Floor(123.1) =123 Floor(123.9) =123	

Function:	Abs	Group: Maths Functions
Description:	Use this function to find the absolute value	
Example:	Abs([Value]) Abs(123.45) = 123.45 Abs(-123.45) = 123.45	
Function:	Ceil	Group: Maths Functions
Description:	Use this function to always round up a value to an integer. This could be the data held in a Numeric or Count variable.	
Example:	Ceil([Value]) Ceil(123.1) = 124 Ceil(123.9) = 124	
Function:	Ln	Group: Maths Functions
Description:	Use this function to find the natural log of a given value.	
Example:	Ln([Value]) Ln(16) = 2.77	

Function:	Exp	Group: Maths Functions
Description:	Use this function to find the exponent of the value i.e. e^value. The inverse of a natural log.	
Example:	Exp([Value])	
Function:	RoundDown	Group: Maths Functions
Description:	Use this function to round down a value to a specified precision. This could be the data held in a Numeric or Count variable.	
Example:	RoundDown([Value],[Precision]) RoundDown(123,10) - rounds down to the next 10 i.e.120	
Function:	Power	Group: Maths Functions
Description:	Use this operator to calculate power of one figure by another e.g. Power(x,y) where the result is the yth power of x.	
Example:	Power(x,y) Calculates x to the power of y. Power(x,y) is equivalent to x ^ y. Power(3,2) = 3 ^ 2 = 9	

Function:	RoundUp	Group: Maths Functions
Description:	Use this function to round up a value to a specified precision. This could be the data held in a Numeric or Count variable.	
Example:	RoundUp([Value],[Precision]) RoundUp(123,10) - rounds up to the next 10 i.e.130	
Function:	Round	Group: Maths Functions
Description:	Use this function to round off a value to a specified precision. This could be the data held in a Numeric or Count variable.	
Example:	Round([Value]) Round([Value], [Precision]) Round(x) rounds x off to the nearest integer. Round(123,10) rounds off to the nearest 10, i.e. 120	
Function:	Rand	Group: Maths Functions
Description:	Use this function to return a random number between 0 and its argument.	
Example:	Rand([Value]) Rand(50) = will return any number randomly between 0 & 50	

Function:	SequenceNumber	Group: Maths Functions
Description:	Use this function to return a new number for each record processed. Useful when having to create a unique row number to otherwise unidentified data.	
Example:	SequenceNumber(1) When used on a Data Grid the first row will show 1, the second row will show 2 etc.	
Function:	HexToDec	Group: Group: Maths Functions
Description:	Use this function to convert hexadecimal strings into decimal numbers.	
Example:	e.g. HexToDec("29A") will return the decimal number 666	
Function:	DecToHex	Group: Group: Maths Functions
Description:	Use this function to convert decimal numbers into hexadecimal strings.	
Example:	e.g. HexToDec("29A") will return the decimal number 666	

Function:		
Description:		
Example:		
Function:		Group:
Description:		
Example:		
Function:		Group:
Description:		
Example:		

Function:	Today	Group: Date Functions
Description:	Use this function to return today's date shifted by n days	
Example:	Today(0) returns today's date Today(1) returns tomorrows date Today(-1) returns yesterday's date	
Function:	AgeQuarters	Group: Date Functions
Description:	Use this function to find the number of quarters (3 calendar months) relative to today	
Example:	AgeQuarters([Policy Renewal Date]) The result would be 3 where today is 12 October 2016 and the Policy Renewal Date is 5 January 2016	
Function:	AgeMonths	Group: Date Functions
Description:	Use this function to find the number of months from a date relative to today	
Example:	AgeMonths([Policy Renewal Date]) The result would be 12 where today is 30 May 2016 and the Policy Renewal date is 20 May 2015	

Function:	AgeDays	Group: Date Functions
Description:	Use this function to find the number of days from a date relative to today	
Example:	AgeDays([Date]) The result would be 10 where today is 30 May 2016 and the Date used is 20 May 2016	
Function:	AgeWeeks	Group: Date Functions
Description:	Use this function to find the number of whole weeks relative to today	
Example:	AgeWeeks([Policy Renewal Date]) The result would be 1 where today is 30 May 2016 and the Policy Renewal Date is 20 May 2016	
Function:	AgeYears	Group: Date Functions
Description:	Use this function to find the number of years from a date relative to today	
Example:	AgeYears([Policy Renewal Date]) The result would be 2 where today is 30 May 2016 and the Policy Renewal Date is 20 May 2014	

Function:	DateDay	Group: Date Functions
Description:	Use this function to return the day of a date, between 1 and 31	
Example:	DateDay([Policy Renewal Date]) 12th August 2016 = 12	
Function:	DateMonth	Group: Date Functions
Description:	Use this function to return the month of a date, between 1 and 12	
Example:	DateMonth([Policy Renewal Date]) 12th August 2016 = 8	
Function:	DateYear	Group: Date Functions
Description:	Use this function to return the year of a date	
Example:	DateYear([Policy Renewal Date]) 12th August 2016 = 2016	

Function:	DateWeek	Group: Date Functions
Description:	Use this function to return the week of a date, between 1 and 53	
Example:	DateWeek([Policy Renewal Date]) 24th April 2016 = 17	
Function:	DateQuarter	Group: Date Functions
Description:	Use this function to return the quarter of a date, between 1 and 4	
Example:	DateQuarter([Policy Renewal Date]) 12th August 2016 = 3	
Function:	DayOfWeek	Group: Date Functions
Description:	Use this function to return the day number where Monday=1, Tuesday=2,…Sunday=7	
Example:	DayOfWeek([Policy Renewal Date]) 12th August 2016 = 3 Wednesday	

Function:	DayOfYear	Group: Date Functions
Description:	Use this function to return the day number in a date for that year	
Example:	DayOfYear([Policy Renewal Date]) 28 May 2016 = 148	
Function:	DateDiff	Group: Date Functions
Description:	Use this function to measure the difference between 2 dates in terms of years, quarters, months, weeks or days. If no unit specified days will be used. If only 1 date used measure will be against today's date.	
Example:	DateDiff([Policy Inception Date],[Policy Cancellation Date],"Days") Will return the number of days between the Policy Inception and Cancellation	
Function:	MakeDate	Group: Date Functions
Description:	Use this function to generate a date from integer variables or constants	

Example:	MakeDate(Year,Month,Day)
	MakeDate(DateYear([Customer Start Date],01,01)
	
	Returns start of the year in which the Customer Start Date fell	

Function:	BuildDate	Group: Date Functions
Description:	Use this function to return the FastStats system build date shifted by n days	
Example:	BuildDate(0) returns the build date BuildDate(-1) returns the day before the build date	
Function:	DateShift	Group: Date Functions
Description:	Use this function to shift a date by a number of units (years, quarters, months, weeks or days)	
Example:	DateShift([Policy Renewal Date],-30,"Days") Calculates 30 days before Date Invoice is due DateShift([DOB],18,"Years") calculates 18th birthday	
Function:	DaysInMonth	Group: Date Functions
Description:	Use this function to return the number of days in a month	
Example:	DaysInMonth([Policy Renewal Date] returns days in month DaysInMonth([Date],"1111100") returns week days in the month, where the sequence represents Monday to Sunday and where 1 returns the day and 0 does not	

Function:	DateStart	Group: Date Functions
Description:	Use this function to find the start of a time period and then offset by a number of days	
Example:	DateStart([Policy Renewal Date],5,"Months") If Policy Renewal Date is 12 August 2016 result = 6 August 2016	
Function:	SelectedDays	Group: Date Functions
Description:	Use this function to find the number of days between 2 dates based upon the patter given	
Example:	SelectedDays([Date1], [Date2],"1111100") This would select the number of week days (Mon to Fri) between the 2 dates	
Function:	DateISO	Group: Date Conversion
Description:	Use this function to convert a date into ISO 8601 format https://en.wikipedia.org/wiki/ISO_8601	
Example:	DateISO([Policy Inception Date]) 01-12-2016 = 2016-W49-1	

Function:	DateEnd	Group: Date Functions
Description:	Use this function to find the end of a time period and then offset by a number of days	
Example:	DateEnd([Policy Renewal Date],5,"Months") If Policy Renewal Date is 12 August 2016 result = 6 September 2016	
Function:	FormatDate	Group: Date Conversion
Description:	Use this function to convert a date variable to its string representation. See Appendix 1 for more details	
Example:	FormatDate([Policy Inception Date],"%d-%m-%Y") This would represent the date in the format 12-08-2016	
Function:	FinancialDate	Group: Date Conversion
Description:	Use this function to return a date relative to the financial year and unit specified (i.e. y, q, m, w, d). This will require a Financial dates to have been setup in your system. The default date is 6th April	
Example:	FinancialDate([Policy Inception Date],"m") 14-08-2015 = 2015-2016 M05	

Function:	FirstDate	Group: Date List Functions
Description:	Use this function to find the first date in a list of dates	

Example:	FirstDate([D1],[D2],[D3])
	
	D1=2014, D2=2011, D3=2016 D2 2011 will be selected	

Function:	TimeDiff	Group: DateTime Functions
Description:	Use this function to calculate the difference between 2 DateTimes, measured in the specified units. If no units are specified, seconds are used. If only 1 DateTime specified it will compare to the current date and time.	
Example:	TimeDiff([DateTime1, DateTime2, “Hours”) The difference between the 2 variables calculated in hours	
Function:	TimeMinute	Group: DateTime Functions
Description:	Use this function to calculate the minute of hour of a DateTime	
Example:	TimeMinute([Communication Date]) 12-08-2016 10:16:23 = 16	

Function:	LastDate	Group: Date List Functions
Description:	Use this function to find the last date in a list of dates	

Example:	LastDate([D1],[D2],[D3])
	
	D1=2014, D2=2011, D3=2015 D3 2016 will be selected	

Function:	TimeHour	Group: DateTime Functions
Description:	Use this function to calculate the hour of day of a DateTime	
Example:	TimeHour([Communication Date]) 12-08-2016 10:16:23 = 10	
Function:	TimeSecond	Group: DateTime Functions
Description:	Use this function to calculate the second of minute of a DateTime	
Example:	TimeSecond([Communication Date]) 12-08-2016 10:16:23 = 23	

Function:	MakeDateTime	Group: DateTime Functions
Description:	Use this function to create a fixed reference date for use in Expressions	
Example:	Timediff([Date of Communication],MakeDate Time(2016,08,06,09,00,00) This will give you the time in seconds between last communication and 9am 6th August 2016	
Function:	SelectedHours	Group: DateTime Functions
Description:	Use this function to calculate the number of hours between two date time values.	
Example:	SelectedHours([Communication Date],MakeDateTime(2016,07,01,00,00,00),"1111100",8.30,17.30) Here we are considering the hours in a working week, Monday to Friday between 8.30am to 5.30pm (24 hour time display).	
Function:	Mean	Group: Aggregation Functions
Description:	Use this function to find the mean average of a list of values	
Example:	Mean(2.3,3.4,5.2,2.2) the result will be 3.28 (2.3+3.4+5.2+2.2)/4 = 3.28	

Function:	DateNext	Group: Date Functions
Description:	Use this function to return the specified next occurrence of a date in years or months	
Example:	DateNext([DOB],1,"years") This example would find someone's next birthday. E.g. Today is 06.10.15, DOB is 20.04.66 next date would be 20.04.16	
Function:	Max	Group: Aggregation Functions
Description:	Use this function to find the maximum value in a list	
Example:	Min(2.3,3.4,5.2,2.2) the result will be 5.2	
Function:	Median	Group: Aggregation Functions
Description:	Use this function to find the median average of a list of values. List the values in order and select the middle value	
Example:	Median(2.3,3.4,5.2,2.2) the result will be 2.85 2.2, 2.3, 3.4, 5.2 (2.3+3.4)/2 = 2.85	

Function:	Mode	Group: Aggregation Functions
Description:	Use this function to find the modal value from a list of values e.g. the most frequently occurring value. The mode is undefined if there is a tie but by convention we return the first tied candidate	
Example:	Mode(1,2,3,2,4,2,3) = 2	
Function:	Sum	Group: Aggregation Functions
Description:	Use this function to find the sum of a list of values	
Example:	Sum(1,2,3,2,4,2,3) = 17	
Function:	If	Group: Logical Functions
Description:	Use this function to test a condition and then determine the output. If(A,B,C) value C is optional and if omitted returns a missing value if the condition is not satisfied	
Example:	If([Customer Level Revenue]>100000, “High Value”, “Low Value”) 21598 = High Value 879 = Low Value	

Function:	StdDev	Group: Aggregation Functions
Description:	Use this function to find the standard deviation of a list of values	
Example:	StdDev(1,2,3,2,4,2,3) = 1.39	
Function:	Min	Group: Aggregation Functions
Description:	Use this function to find the minimum value in a list	
Example:	Min(2.3,3.4,5.2,2.2) the result will be 2.2	
Function:	Case	Group: Logical Functions
Description:	Use this function to create a list of tests and their outputs if met. This function can be used instead of using nested If statements	
Example:	Case([Division 1 Spend] <100,”A”,[Cost]<200,”B”,[Cost]>=200”C”) If Cost is equal to 100 output B etc.	

Function:	Or	Group: Logical Functions
Description:	Use this function to evaluate if 1 or more of a series of conditions can be met. If so True/1, if not then False/0	
Example:	Or([Policy premium]>3000,"A", [Customer Level Revenue]>10000, [Net Worth]>2000000) One of the conditions must be true	
Function:	InList	Group: Logical Functions
Description:	Use this function to return an index of a given value in a list.	
Example:	InList([Surname],"Smith","Jones","Brown") 0=Not in list 1=Smith 2=Jones 3=Brown	
Function:	MaxIndex	Group: Logical Functions
Description:	Use this function to return the index of the maximum value in a list of values.	
Example:	MaxIndex([Division 1 Spend],[Division 2 Spend], [Division 3 Spend]) Will return 1 if Division 1 Spend is the highest, 2 if Division 2 Spend is …	

Function:	And	Group: Logical Functions
Description:	Use this function to evaluate if a number of conditions are met and if so are marked as True/1, if not False/0	
Example:	And([Policy Premium]>2000,[Net Worth]<20000) All of the conditions must be true	
Function:	Index	Group: Logical Functions
Description:	Use this function to select the X th value in a list. See Scenario 1 later in this manual.	
Example:	Index(5,1,5,10,15,20) The 5th value in the list = 20 Index([No of Sites in GU group],1,5,10,15,20) 1 Sites = 1, 2 Sites = 5, 3 Sites = 10 etc.	
Function:	MinIndex	Group: Logical Functions
Description:	Use this function to return the index of the minimum value in a list of values.	
Example:	MinIndex([Division 1 Spend],[Division 2 Spend], [Division 3 Spend]) Will return 1 if Division 1 Spend is the lowest, 2 if Division 2 Spend is …	

Function:	Not	Group: Logical Functions
Description:	Use this function to negate the logical value inside the brackets	
Example:	Not([Policy Premium]<100) Outputs 1 if over 100 or 0 if under 100 If(Not([Policy Premium]<100),"High","Low")	
Function:		Group:
Description:		
Example:		
Function:	NotMissing	Group: Missing Value Functions
Description:	Use this function to indicate if a values is not missing.	
Example:	NotMissing([Email]) Outputs 1 if not missing or 0 if missing	

Function:	ValidEmail	Group: Logical Functions
Description:	Use this function to validate if an email address structure is correct. See Appendix 2	
Example:	ValidEmail([Email]) Outputs 1 if valid or 0 if not valid	
Function:	IsMissing	Group: Missing Value Functions
Description:	Use this function to indicate if a value is missing.	
Example:	IsMissing([Email]) Outputs 1 if missing or 0 if present	
Function:	MissingValue	Group: Missing Value Functions
Description:	Use this function to return a constant missing value.	
Example:	If([Age]<110,[Age],MissingValue()) This will treat anyone with an age over 110 as a missing value	

Function:	CodeOf	Group: Selector Functions
Description:	Use this function to return the code of a selector variable	

Example:	CodeOf([Gender])
	
	Male = M Female =F	

Function:	ValueOf	Group: Selector Functions
Description:	Use this function to identify a number from the description of a selector category.	
Example:	ValueOf([Carrier Route Code]) e.g. BO50 = 50	
Function:	InstantCountOf	Group: Selector Functions
Description:	Use this function to identify the count of a selector category.	
Example:	InstantCountOf([Carrier Route Code]) e.g. BO50 = 16041	

Function:	DescOf	Group: Selector Functions
Description:	Use this function to return the decode description of a selector variable	

Example:	DescOf([Gender])
	
	M = Male F = Female	

Function:	IndexOf	Group: Selector Functions
Description:	Use this variable if you want to return the index of the code of the selector variable as a number	

Example:	IndexOf([CEO Gender])
	
	Female = 1 Male = 2 Unknown = 3	

Function:	Populated	Group: Selector Functions
Description:	Use this function to find non missing values. For single selectors this will be 0 or 1. Arrays & Flag Arrays will give the number of slots populated.	
Example:	Populated([Newspapers]) e.g. N where N is the number of different newspapers read	

Function:	VarSelect	Group: Selector Functions
Description:	Use this function to return the index of the first matching code in a list.	
Example:	VarSelect([Newspapers],"The Financial Times"," The Guardian","The Record") Will return 0 if there is no match	
Function:	VarDesc	Group: Selector Functions
Description:	Use this function to return the description of or the index value of a selector variable category	
Example:	VarDesc([UK SIC 2 digit],"1") = The first MRC code VarDesc([Occupation],1) = Index of Manual Worker	
Function:		Group:
Description:		
Example:		

Function:	CardinalityOf	Group: Selector Functions
Description:	Use this function to return the number of category codes for a selector variable.	
Example:	CardinalityOf([Gender]) = 4 e.g. Unclassified, Female, Male, Unknown	
Function:	VarSelectCode	Group: Selector Functions
Description:	Use this function to return the first matched code based on each selector variable evaluating along the parameters. Multiple sets of selectors and values can be used.	
Example:	VarSelectCode([Postcode], "CV34 4EH","SW1A 2AA") Adding "AsDesc" after the function will output the descriptions if available.	
Function:		Group:
Description:		
Example:		

Function:	SubStr	Group: String Functions
Description:	Use this function to return a proportion of a string. The start point is 0	
Example:	SubStr([Surname],0,3) Smith = Smi SubStr([Surname],1,3) Smith = mit	
Function:	FormatNumber	Group: String Functions
Description:	Use this function to format a number to a given precision	
Example:	FormatNumber([Cost],0) 56.78 = 56	
Function:	StrLength	Group: String Functions
Description:	Use this function to find the length of a string as a number	
Example:	StrLength([Surname]) e.g. Smith = 5	

Function:	AddStr	Group: String Functions
Description:	Use this function to concatenate strings together	
Example:	AddStr([Title],” “,[Initial],” “,[Surname]) This may return Mr J Smith	
Function:	StrCompare	Group: String Functions
Description:	Compares two strings alphabetically. Returns 1 if the first string is greater than the second string. Returns -1 if the first string is less than the second string. Returns 0 if both strings are equal. There is an optional third parameter, a numeric flag (0 or 1) indicating whether the comparison is case sensitive.	
Example:	StrCompare(“Market Insight”,”D&B”,1) e.g. the above example will return 1	
Function:	StrLower	Group: String Functions
Description:	Use this function to convert a string into lower case	
Example:	StrLower([Surname]) e.g. Smith = smith	

Function:	StrUpper	Group: String Functions
Description:	Use this function to convert a string into upper case	
Example:	StrUpper([Surname]) e.g. Smith = SMITH	
Function:	StrContains	Group: String Functions
Description:	Use this function to identify substrings in a case sensitive text variable and return the index of the first match. No match returns zero.	
Example:	StrContains([Surname],”ar”,”be”,”ca”) e.g. Parkin = 1 Roberts = 2 Cable =0 **Case** sensitive	
Function:	StrFind	Group: String Functions
Description:	Use this function to find a substring within a string. Returns the zero based start offset of the substring or -1 if not found.	
Example:	StrFind([Surname],”Red”) (Case Sensitive) e.g. Redmond = 0 Brown = -1	

Function:	StrProper	Group: String Functions
Description:	Use this function to convert a string to capitalise first letters where appropriate	
Example:	StrProper([First Name],” “,[Surname]) e.g. james smith = James Smith	
Function:	StrContainsI	Group: String Functions
Description:	Case insensitive version of the StrContains function.	
Example:	StrContains|([Surname],”ar”,”Be”,”ca”) e.g. Parkin = 1 Roberts = 2 Cable =3 **Not** case sensitive	
Function:	StrFindI	Group: String Functions
Description:	Use this function to find a substring within a string. Returns the zero based start offset of the substring or -1 if not found.	
Example:	StrFind([Surname],”odd”) (Case Insensitive) e.g. Stoddart = 2 Jones = -1	

Function:	StrReverse	Group: String Functions
Description:	Use this function to return a string in reverse	
Example:	StrReverse([Surname]) e.g. Grant = tnarG	
Function:	Right	Group: String Functions
Description:	Use this function to return the first *n* characters from the right part of a string.	
Example:	Right([Surname],2) e.g. Grant = nt	
Function:	LeftTrim	Group: String Functions
Description:	Use this function to remove leading spaces from a string	
Example:	LeftTrim(“ Text”) e.g. “ Apteco” = “Apteco”	

Function:	Left	Group: String Functions
Description:	Use this function to return the first *n* characters from the left part of a string.	
Example:	Left([Surname],2) e.g. Grant = Gr	
Function:	Mid	Group: String Functions
Description:	Use this function to return a portion of a string. This will extract *n* characters between 2 points	
Example:	Mid([Surname],2,4) e.g. Thomson = homs	
Function:	Trim	Group: String Functions
Description:	Use this function to remove all spaces except for leaving a single space between words	
Example:	Trim(String) e.g. “ James Smith Apteco” = “James Smith Apteco”	

Function:	StrClean	Group: String Functions
Description:	Use this function to remove all characters from string 1 that are listed in string 2 (case sensitive)	
Example:	e.g. StrClean(Apteco,e) = Aptco StrClean([Telephone],”+“) = 15184992341	
Function:	StrBegins	Group: String Functions
Description:	Use this function to check if string X begins with certain letters and return an index of the first match	
Example:	StrBegins([Surname],”A”,”B”,”C”) e.g. Alcock =1 Birine =2 Cheshire = 3 Pardoe = 0	
Function:	StrBeginsI	Group: String Functions
Description:	Use this function to check if string X begins with certain letters and return an index of the first match (case insensitive)	
Example:	StrBeginsI([Surname],”a”,”B”,”c”) e.g. Alcock =1 Birine =2 Cheshire = 3 Pardoe = 0	

Function:	StrReplace	Group: String Functions
Description:	Use this function to replace all occurrences of A with B in string X (case sensitive)	
Example:	StrReplace(StringX,StringA,StringB) e.g. StrReplace(Apteco,A,a) = apteco	
Function:	StrEnds	Group: String Functions
Description:	Use this function to check if string X ends with certain letters and return an index of the first match	
Example:	StrEnds([Surname],”e”,”y”,”n”) e.g. Pardoe = 1 Grebby = 2 Thomson = 3 Alcock = 0	
Function:	StrEndsI	Group: String Functions
Description:	Use this function to check if string X ends with certain letters and return an index of the first match (case insensitive)	
Example:	StrEndsI([Surname],”E”,”y”,”n”) e.g. Pardoe = 1 Grebby = 2 Thomson = 3 Alcock = 0	

Function:	StrListItems	Group: String Functions
Description:	Use this function to select the nth item from a delimited list as a string	
Example:	StrListItems(Text,Delimier,N) StrListItems(“RoadIStreetICloseISquare”,”I”,3) = Close	
Function:	StrDeleteRepeats	Group: String Functions
Description:	Use this function to remove duplicate characters from a string.	
Example:	StrDeleteRepeats(“ABCCCBBBAA”) This would return ABCBA	
Function:	Telephone	Group: String Functions
Description:	Use this function to convert a text entry into a number which is compatible with TPS 1.2 “Without Spaces” format. See: https://corporate.tpsonline.org.uk/index.php/tps/technical	
Example:	Telephone([Telephone],”TPS”) e.g. 01224-631892 to 01224631892 This UK specific option will also remove any 44 prefix from the number.	

Function:	StrNumber	Group: String Functions
Description:	Use this function to covert a string value to its numeric representation. If a number cannot be represented it will evaluate to a missing value.	
Example:	StrNumber([Telephone]) e.g. 01224-631892 = 1224 03 654 896541 = 3	
Function:	StrScore	Group: Group: String Functions
Description:	Use this function to score a string by a function with values provided in a file	
Example:	StrScore([Address],”sum”,””Public:Expressions\Textmodel.txt”) See Scenario 4 for a more detailed example	
Function:	StrShred	Group: String Functions
Description:	This function allows you to convert a text entry into a delimited, alphabetical list of the constituent words. The expression format is as follows – StrShred(Text,MinSize,MaxSize,Case,Delimiter). The min and max size will determine the number of letters allowed in the words that are returned.	
Example:	StrSHred([Address],4,20,”Proper”,”,”) e.g. 9 Park Drive to Drive,Park	

Function:	StrShredContains	Group: String Functions
Description:	This function allows you to convert a text entry into a delimited, list as described in StrShred and determine which words to search for. This will return a 1 for a match on the first word searched for, 2 on the second etc. and a 0 if nothing matches.	
Example:	StrShredContains(Text,MinSize,MaxSize,Case,String1,String2…) Where String1, String2 etc. are the words you are searching for. StrShredContains([Address],4,20,"Proper","Close","Road") e.g. 10 Park Road = 2, 4 Carlton Close = 2, 4 The Lane = 0	
Function:	StrStreak	Group: String Functions
Description:	Use this function to return the first characters of a string until there is a change in characters.	
Example:	StrStreak("abbccdde") will return a StrStreak("bbbaccdd") will return bbb	
Function:		
Description:		
Example:		

Function:	StrHash	Group: String Functions
Description:	Use this function to take a string and apply one of a number of algorithms that will encrypt the string so that it is secure when passed over a network.	
Example:	e.g. StrHash([Email Address],"SHA256")	
Function:		
Description:		
Example:		
Function:		
Description:		
Example:		

Function:	GeoDistMin	Group: Location Functions
Description:	Use this function to return the distance from the reference lat/long to the nearest other lat/long	
Example:	GeoDistMin([Postcode Latitude],[Postcode Longitude],51.47,-0.45,52.45,-1.74,55.95,-3.36) This will return the distance to the nearest airport in the list	
Function:	GeoNth	Group: Location Functions
Description:	Use this function to find the Nth nearest to a lat/long reference. This is determined by the first parameter in the expression.	
Example:	GeoNth([Postcode Latitude],[Postcode Longitude],2,51.47,-0.45,52.45,-1.74,55.95,-3.36) This example will find the 2nd nearest lat/long in the list	
Function:	UKPostcodeDistance	Group: Location Functions
Description:	Use this function to return the distance in miles between 2 UK postcodes	
Example:	UKPostcodeDistance([Postcode],”CV344EH”)	

Function:	GeoDistNth	Group: Location Functions
Description:	Use this function to find the distance in miles to the Nth nearest to a lat/long reference. This is determined by the first parameter in the expression.	
Example:	GeoDistNth([Postcode Latitude],[Postcode Longitude],2,51.47,-0.45,52.45,-1.74,55.95,-3.36) This example will find the distance to the 2nd nearest lat/long in the list	
Function:	UKPostcode	Group: Location Functions
Description:	Use this function to locate a feature from a UK postcode such as “Valid”, “Easting”, “Northing”, “Latitude”, “Longitude”	
Example:	UKPostcode([Postcode],”Easting”)	
Function:	UKPostcodeDistMin	Group: Location Functions
Description:	Use this function to return the distance from the reference UK postcode to the nearest other postcode	
Example:	UKPostcodeDistMin([Postcode],”CV34 4EH”, “B4 7UL”)	

Function:	UKPostcodeNearest	Group: Location Functions
Description:	Use this function to return the index of the nearest postcode to the reference UK postcode	
Example:	UKPostcodeDistMin([Postcode],”CV34 4EH”, “B4 7UL”) Nearer to CV34 = 1 Nearer to B4 = 2	
Function:	UKPostcodeNth	Group: Location Functions
Description:	Use this function to find the Nth nearest to a postcode. This is determined by the first parameter in the expression.	
Example:	UKPostcodeNth(2,[Postcode],”EX38 8LH”, “CV34 4EH”,”M90 1QX”) This example will find the 2nd nearest postcode in the list	
Function:	GeoDist	Group: Location Functions
Description:	Use this function to calculate the straight line distance between two points using latitude and longitude	
Example:	GeoDist([Postcode Latitude],[Postcode Longitude],51.47,-0.45) This would find the distance between a person’s location and Heathrow airport using latitude & longitude	

Function:	UKPostcodeFormat	Group: Location Functions
Description:	Use this function to return a correctly formatted UK postcode using “Area”, “District”, “Sector”, “Outward”, “Inward”, “Postcode”	
Example:	UKPostcodeFormat([Postcode],”District”)	
Function:	UKPostcodeDistNth	Group: Location Functions
Description:	Use this function to find the distance in miles to the Nth nearest to a postcode. This is determined by the first parameter in the expression.	
Example:	UKPostcodeDistNth(2,[Postcode],”EX38 8LH”, “CV34 4EH”,”M90 1QX”) This example will find the distance to the 2nd nearest postcode in the list	
Function:	GeoNearest	Group: Location Functions
Description:	Use this function to return the index of the nearest lat/long to the reference lat/long location	
Example:	GeoNearest([Postcode Latitude],[Postcode Longitude],51.47,-0.45,52.45,-1.74,55.95,-3.36) Nearest to Heathrow Airport = 1, Birmingham Airport = 2, Edinburgh Airport = 3	

Function:	CubeLookup	Group: Cube Functions
Description:	Use this function to look up a cell value in a Cube based upon the dimension values for the current record	
Example:	CubeLookup({Cube #1 – Cube}) See Scenario 3 for a more detailed example	
Function:	ModelScore	Group: Modelling Functions
Description:	Use this function to calculate a score which is based upon a model defined in a file in the PMML format. The referenced model must use the same variable references as in your FastStats system	
Example:	ModelScore(“Public:ModelExample.pmml”,[Emp Total Range],[Manufacturing Indicator],[Location Type])	
Function:	ZScore	Group: Modelling Functions
Description:	Use this function to find the statistical significance value given a promotional response rate, a control response rate, number of people contacted	
Example:	ZScore(“Proportion”,0.04,0.03,500) e.g. a result of 1.31 is >95% statistically significant	

Function:		Group:
Description:		
Example:		
Function:	ConfidenceInterval	Group: Modelling Functions
Description:	Use this function to calculate the confidence interval given statistic, sample size and significance values	
Example:	ConfidenceInterval(“Proportion”,0.04,1000,0.95) This example is based upon a 4% response rate, a sample size of 1000 and a significance value of 95%	
Function:	MinSample	Group: Modelling Functions
Description:	Use this function to find the minimum sample size needed given a promotional response rate, a control response rate and significance values	
Example:	MinSample(“Proportion”,0.04,0.03,0.95) This example would require a minimum sample of 1,117	

Function:	RCall	Group: Modelling Functions
Description:	Use this function to call a function in R.	
Example:	e.g. RCall(“Min”, [Region 1 Spend], [Region 2 Send], [Region 3 Spend])	
Function:	Nth	Group: Sorting Functions
Description:	Use this function to select the value of a list once the values have been sorted in ascending order.	
Example:	Nth(2,8,10,4,6) e.g. when ordered 4,6,8,10 the 2nd value = 6	
Function:	BandUp	Group: Banding Functions
Description:	Use this function to allocate an item to a band. BandUp returns an index taking the parameters as ascending bands	
Example:	BandUp([Transaction Date],2013,2014,2015,2016) e.g. 12-5-2014 = 2 09-04-2016 = 4	

Function:	RScript	Group: Modelling Functions
Description:	Use this function to run a script in R. @ represents a place holder substituting a variable.	
Example:	e.g. RScript(“min(Mean @1,@2,@3”, [Division 1 Spend], [Division 2 spend], Division 3 Spend])	
Function:	NthIndex	Group: Sorting Functions
Description:	Use this function to select the index of a value within a list once the values have been sorted in ascending order.	
Example:	NthIndex(2,8,10,4,6) e.g. the 2nd value when ordered is 6 and the index of that value in the list is 4	
Function:	BandDown	Group: Banding Functions
Description:	Use this function to allocate an item to a band. BandDown returns an index taking the parameters as descending bands	
Example:	BandUp([Transaction Date],2013,2014,2015,2016) e.g. 12-5-2014 = 3 09-04-2016 = 1	

Expression Scenarios

The following pages are examples of User requests that have been resolved by using Expressions as part of the solution.

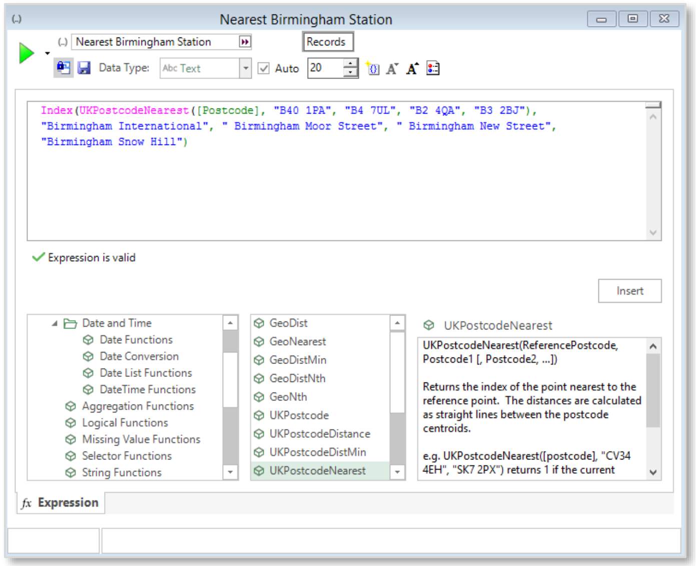
Scenario 1 – Location Functions

The User wants to find the closest retail outlet to where a customer lives and the distance from their postcode.

To illustrate this scenario we will look to find which train station Records, in the West Midlands Region, are closest to and their distance from that station. It is important to note that this method uses an as the 'Crow' flies calculation and <u>not</u> the road network.

Here we can make use of a number of Expression functions to calculate the information we are looking for and then display it on a Data Grid. Before we start we need to know the postcode of the locations and the name we want to associate with those locations. In our example we will use the locations and names of 4 train stations in Birmingham. (This example assumes you have a rudimentary understanding of the Expression tool)

- ➢ Open the Expression tool and complete as shown opposite



```
Index(UKPostcodeNearest([Postcode], "B40 1PA", "B4 7UL", "B2 4QA", "B3 2BJ"),
"Birmingham International", " Birmingham Moor Street", " Birmingham New Street",
"Birmingham Snow Hill")
```

Expression is valid

UKPostcodeNearest

UKPostcodeNearest(ReferencePostcode, Postcode1 [, Postcode2, ...])

Returns the index of the point nearest to the reference point. The distances are calculated as straight lines between the postcode centroids.

e.g. UKPostcodeNearest([postcode], "CV34 4EH", "SK7 2PX") returns 1 if the current

Using the **UKPostcodeNearest** function will reference the postcode in the data and then return the nearest postcode in the list. Using the **Index** function at the beginning of this expression will then take the description associated with the selected train station postcode.

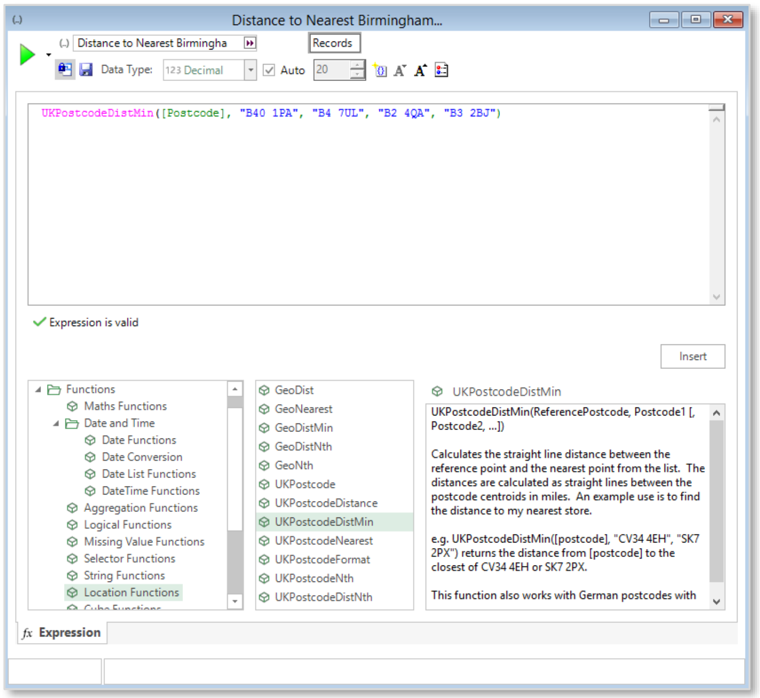
- ➢ Save the Expression as **Nearest Birmingham Station**

To find the distance from the customer’s home to their nearest train station another expression can be built using the **UKPostcodeDistMin** function.

- Open a new Expression window and complete as opposite
- Save the Expression as **Distance to Nearest Birmingham Station**

We can now take these expressions and add them to a Data Grid to see the results.

- Create a selection of Records from the **West Midlands Region**
- Drag a Data Grid on to your selection and add the variables and expressions as below then Click the **Build** button



```
UKPostcodeDistMin([Postcode], "B40 1PA", "B4 7UL", "B2 4QA", "B3 2BJ")
```

Expression is valid

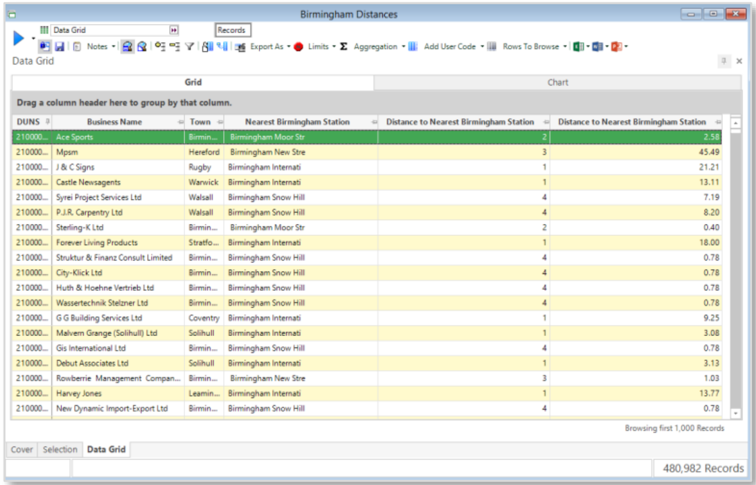
UKPostcodeDistMin

UKPostcodeDistMin(ReferencePostcode, Postcode1 [, Postcode2, ...])

Calculates the straight line distance between the reference point and the nearest point from the list. The distances are calculated as straight lines between the postcode centroids in miles. An example use is to find the distance to my nearest store.

e.g. UKPostcodeDistMin([postcode], "CV34 4EH", "SK7 2PX") returns the distance from [postcode] to the closest of CV34 4EH or SK7 2PX.

This function also works with German postcodes with



Drag a column header here to group by that column.

DUNS	Business Name	Town	Nearest Birmingham Station	Distance to Nearest Birmingham Station	Distance to Nearest Birmingham Station
210000...	Ace Sports	Birmin...	Birmingham Moor Str	2	2.58
210000...	Mpsm	Hereford	Birmingham New Stre	3	45.49
210000...	J & C Signs	Rugby	Birmingham Internati	1	21.21
210000...	Castle Newsagents	Warwick	Birmingham Internati	1	13.11
210000...	Syrei Project Services Ltd	Walsall	Birmingham Snow Hill	4	7.19
210000...	P.J.R. Carpentry Ltd	Walsall	Birmingham Snow Hill	4	8.20
210000...	Sterling-K Ltd	Birmin...	Birmingham Moor Str	2	0.40
210000...	Forever Living Products	Stratfo...	Birmingham Internati	1	18.00
210000...	Struktur & Finanz Consult Limited	Birmin...	Birmingham Snow Hill	4	0.78
210000...	City-Klick Ltd	Birmin...	Birmingham Snow Hill	4	0.78
210000...	Huth & Hoehne Vertrieb Ltd	Birmin...	Birmingham Snow Hill	4	0.78
210000...	Wassertechnik Stelzner Ltd	Birmin...	Birmingham Snow Hill	4	0.78
210000...	G G Building Services Ltd	Coventry	Birmingham Internati	1	9.25
210000...	Malvern Grange (Solihull) Ltd	Solihull	Birmingham Internati	1	3.08
210000...	Gis International Ltd	Birmin...	Birmingham Snow Hill	4	0.78
210000...	Debut Associates Ltd	Solihull	Birmingham Internati	1	3.13
210000...	Rowberrie Management Compan...	Birmin...	Birmingham New Stre	3	1.03
210000...	Harvey Jones	Leamin...	Birmingham Internati	1	13.77
210000...	New Dynamic Import-Export Ltd	Birmin...	Birmingham Snow Hill	4	0.78

Browsing first 1,000 Records

480,982 Records

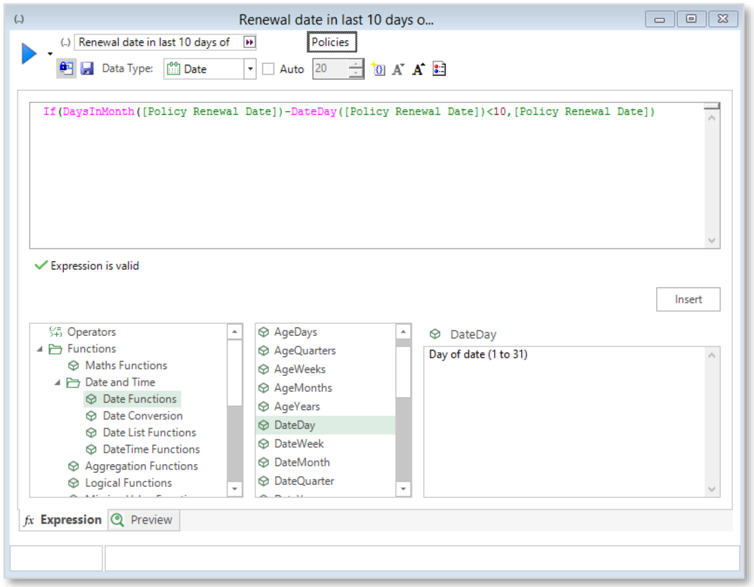
Scenario 2 – Date Functions

The User wanted to identify the number of transactions in the last 10 days in each month and then compare the results to see which customers have made purchases in that time period in subsequent months?

To illustrate this scenario we will use Date functions within an Expression to find Policy Renewal in the last 10 days of a month. This Expression can then be used with a collapsible Tree to find those who have made subsequent bookings in the same time frame.

- ➢ Open the Expression tool and complete as shown opposite

If(DaysInMonth([Policy Renewal Date])-DateDay([Policy Renewal Date])<10, [Policy Renewal Date])



Renewal date in last 10 days o...

Renewal date in last 10 days of — Policies

Data Type: Date — Auto 20

```
If(DaysInMonth([Policy Renewal Date])-DateDay([Policy Renewal Date])<10,[Policy Renewal Date])
```

Expression is valid

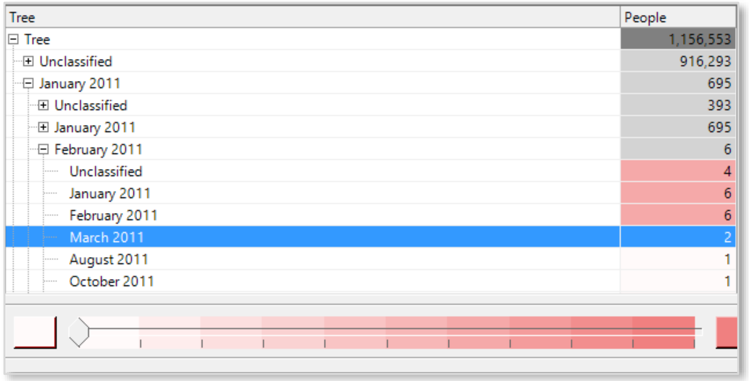
DateDay: Day of date (1 to 31)

This Expression will identify the number of days in the month of a given transaction date (DaysInMonth) and then minus from this figure the day of that transaction date (DateDay). If the result of this calculation is less than 10 then that renewal is in the last 10 days of that month.

- ➢ Use the **Calculate Expression** wizard to convert the Expression into a **Virtual Variable**

- ➢ Open a **Tree** tool set to the collapsible option

- ➢ Right drag the **Virtual Variable** on to the drop box and select Months. Repeat for the next 2 drop boxes. Click the **Build** button



Tree	People
⊟ Tree	1,156,553
⊞ Unclassified	916,293
⊟ January 2011	695
⊞ Unclassified	393
⊞ January 2011	695
⊟ February 2011	6
Unclassified	4
January 2011	6
February 2011	6
March 2011	2
August 2011	1
October 2011	1

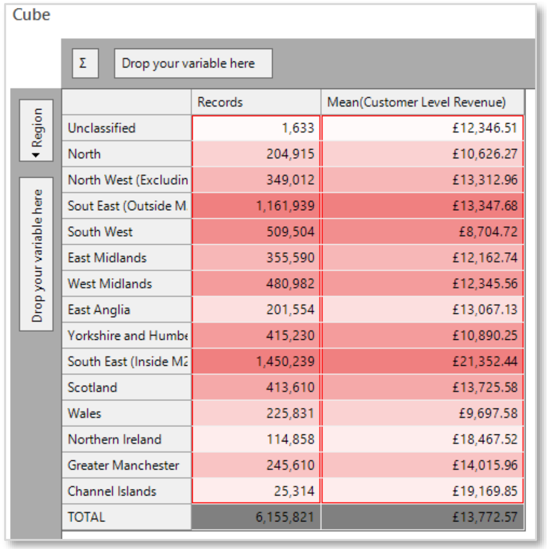
By drilling into January 2011 and then February 2011 and then view the result of March 2011. Here we can see that 2 policies were renewed in the last 10 days of each of those months.

Scenario 3 – Cube Functions

The User wants to find Records with a higher average Customer Level Revenue than the Region within which they live.

To illustrate this scenario we will use the CubeLookUp function to find the mean Customer Level Revenue from a Cube. We can then use a mean cost virtual variable to then create an expression to identify those with a higher average value than their Region.

- Open a blank Selection at the **Records** table level and drop a **Cube** on top
- Drag **Region** onto the vertical dimension and then right drag **Customer Level Revenue** into the centre of the **Cube** and select **Mean**
- Click on the **Build** button

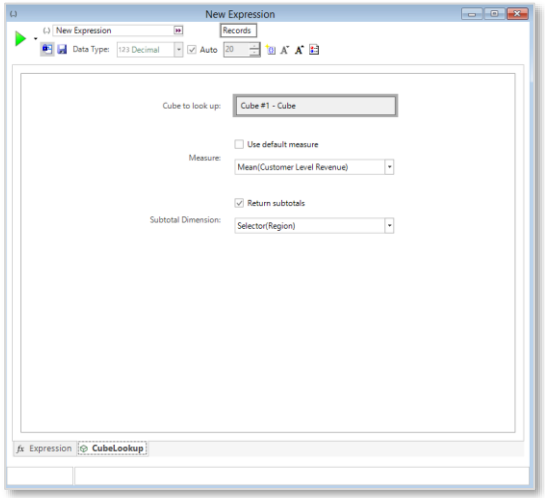


	Records	Mean(Customer Level Revenue)
Unclassified	1,633	£12,346.51
North	204,915	£10,626.27
North West (Excludin	349,012	£13,312.96
Sout East (Outside M	1,161,939	£13,347.68
South West	509,504	£8,704.72
East Midlands	355,590	£12,162.74
West Midlands	480,982	£12,345.56
East Anglia	201,554	£13,067.13
Yorkshire and Humbe	415,230	£10,890.25
South East (Inside Mi	1,450,239	£21,352.44
Scotland	413,610	£13,725.58
Wales	225,831	£9,697.58
Northern Ireland	114,858	£18,467.52
Greater Manchester	245,610	£14,015.96
Channel Islands	25,314	£19,169.85
TOTAL	6,155,821	£13,772.57

- Open an Expression window and insert the **CubeLookUp** function
- Drag your Cube by its drag handle and drop it into the Expression and close the bracket

  CubeLookup({Cube #1 - Cube})

- Ensure the expression is set to the **Records** table
- Click on the **CubeLookup tab** on the Expression window and deselect the tick box and use the settings in the screenshot opposite



Cube to look up: Cube #1 - Cube

Measure: ☐ Use default measure
Mean(Customer Level Revenue)

Subtotal Dimension: ☑ Return subtotals
Selector(Region)

- Click on the **Build** button to now preview the correct figures being looked up in the Cube

To find out if an individual has a higher mean value we can take a pre created virtual variable (use the Aggregation wizard if you wish to create this) and add it to the Expression.

- Add your virtual variable to the beginning of the Expression followed by a minus sign

- Click the **Build** button to see the preview of the results

N.B. A positive figure will indicate a Record with a higher average revenue.

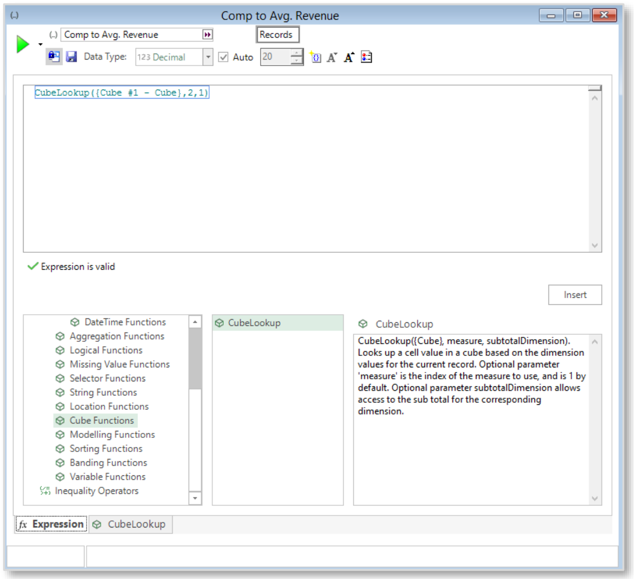
To identify these people as a selection:

- Open a blank Selection window set to the **Records** table level

- Drag the Expression on to the Selection window and type **>0** in the free form panel

- Click the **Build** button

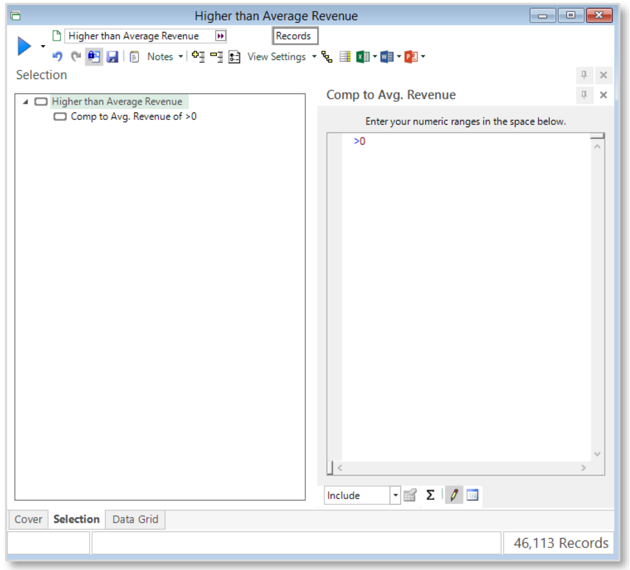
This will identify all the Records with positive values who are in fact those with a higher average spend than that of the Region within which they are based.


Comp to Avg. Revenue
Comp to Avg. Revenue — Records
Data Type: 123 Decimal — Auto: ✓ — 20

```
CubeLookup({Cube #1 - Cube},2,1)
```

✓ Expression is valid

CubeLookup({Cube}, measure, subtotalDimension). Looks up a cell value in a cube based on the dimension values for the current record. Optional parameter 'measure' is the index of the measure to use, and is 1 by default. Optional parameter subtotalDimension allows access to the sub total for the corresponding dimension.


Higher than Average Revenue
Higher than Average Revenue — Records

Selection
- Higher than Average Revenue
  - Comp to Avg. Revenue of >0

Comp to Avg. Revenue
Enter your numeric ranges in the space below.

```
>0
```

Include

46,113 Records

Scenario 4 – String Functions

The user would like to score social media feedback. This will require allocating a score to certain words that can be identified in a string to be used against a text variable within your FastStats system.

As we do not have any social media data in the Market Insight database we will use the Physical Street Address text variable to illustrate our example.

Before we can construct our expression we need to create a file of the words and their scores that can be referenced. An example is shown in the screen shot opposite with words that will appear in the Address text variable and an allocated score. These words could easily be substituted for those you want to look for in your social media e.g. good or bad, beautiful or ugly etc.

The location of this file will then need to be referenced within the Expression.

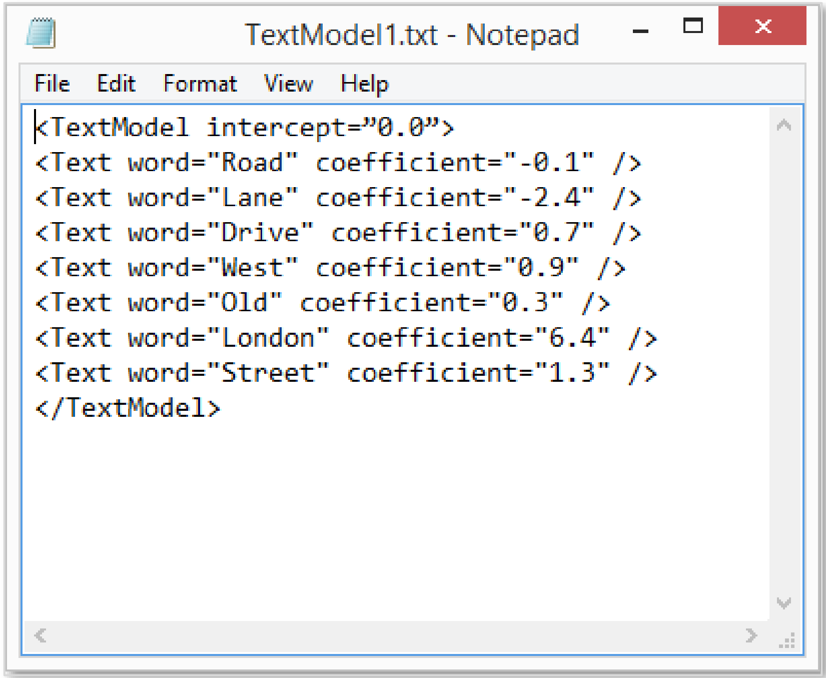
- Open an **Expression** window set to the **People** table level

- Enter the following Expression:

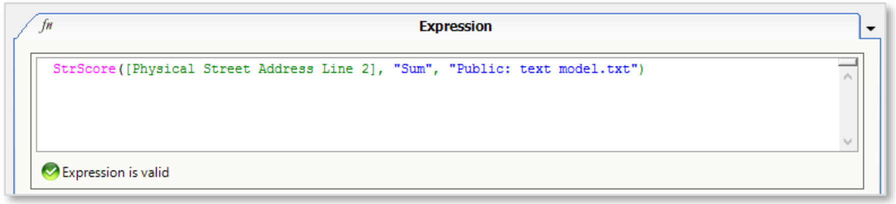
  StrScore([Address],"sum","Public:text model.txt")

- The final parameter of this Expression should be the location of your txt file

- Click the **Build** button to see a preview of the results



```
<TextModel intercept=”0.0”>
<Text word="Road" coefficient="-0.1" />
<Text word="Lane" coefficient="-2.4" />
<Text word="Drive" coefficient="0.7" />
<Text word="West" coefficient="0.9" />
<Text word="Old" coefficient="0.3" />
<Text word="London" coefficient="6.4" />
<Text word="Street" coefficient="1.3" />
</TextModel>
```



```
StrScore([Physical Street Address Line 2], "Sum", "Public: text model.txt")
```

Expression is valid

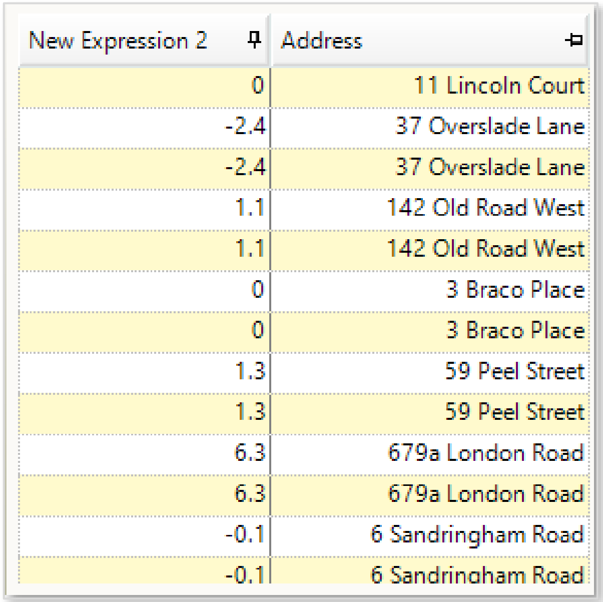
In this expression we used the “sum” parameter to give us a total score for each record.

e.g. 679a London Road = 6.3

London = 6.4 & Road = -0.1

6.4 + -0.1 = 6.3

Alternative statistical parameters are mean, min and max. The use of “Sum|All” will count every occurrence of a word found in the text.



New Expression 2	Address
0	11 Lincoln Court
-2.4	37 Overslade Lane
-2.4	37 Overslade Lane
1.1	142 Old Road West
1.1	142 Old Road West
0	3 Braco Place
0	3 Braco Place
1.3	59 Peel Street
1.3	59 Peel Street
6.3	679a London Road
6.3	679a London Road
-0.1	6 Sandringham Road
-0.1	6 Sandringham Road

Appendix 1 – FormatDate Options

To use a date as a string (e.g. Friday 4 December) in an expression it is necessary to specify the format. Dates are formatted into strings by using the FormatDate function.

FormatDate([Date Variable],”Format Specifier”)

Examples:

FormatDate([Date Variable],”%d-%m-%Y”)	20-08-2015
FormatDate([Date Variable],””)	2015-08-20
FormatDate([Date Variable],”%A, %B %#d”)	Wednesday, August 20
FormatDate([Date Variable],”Date: %#x”)	Date: Wednesday 20 August 2015

The format specifiers are detailed on the following pages, with examples of the output produced by each format specifier.

Format Specifier	Description	Example Output
%a	Abbreviated weekday name	Wed
%A	Full weekday name	Wednesday
%b	Abbreviated month name	Feb
%B	Full month name	February
%d	Day of month as decimal number (01 – 31)	31
%j	Day of year as decimal number (001 – 366)	047
%m	Month as decimal number (01 – 12)	01
%U	Week of year as decimal number, with Sunday as first day of week (00 – 53)	09
%w	Weekday as decimal number (0 – 6; Sunday is 0)	0
%W	Week of year as decimal number, with Monday as first day of week (00 – 53)	09

%x	Date representation for current locale	07/03/2000
%y	Year without century, as decimal number (00 – 99)	15
%Y	Year with century, as decimal number	2015
%z, %Z	Either the time-zone name or time zone abbreviation, depending on registry settings; no characters if time zone is unknown	GMT Standard Time
%%	Percent sign	%

The # flag may prefix any formatting code. In that case, the meaning of the format code is changed as follows.

Format Code	**Meaning**
%#x	Long date representation, appropriate to current locale. For example: "Wednesday, August 19, 2015"
%#d, %#H, %#I, %#j, %#m, %#M, %#S, %#U, %#w, %#W, %#y, %#Y	Remove leading zeros (if any)

Appendix 2 – ValidEmail Rules

The following rules are used when the ValidEmail function is used.

- Email must not be < 6 characters in length
- Email must not be >= 254 characters in length
- Email local part must not be > 64 characters in length
- Email local part must not contain any invalid characters i.e. <sp> (),;:"[]\<>
- Email domain part must only contain valid characters letters, digits, hyphens, dots

N.B. In theory you can have an apostrophe in the first half of email address but not after the @   e.g. David.O'Callaghan@bigcompany.com

- Email must not be missing a ‘@’ character
- Email must not contain more than one ‘@’ character
- Email must not begin with ‘@’ character
- Email must not be missing a ‘.’ character
- Email must not have less than 2 characters after the ‘.’
- Email must not have less than 1 characters between the first ‘@’ and the ‘.’
- Email must not be missing a ‘.’ after the ‘@’ character
- Email must not end with ‘@’ character
- Email may contain '.' but not as first or last character of local part and '..' not allowed