



# オンライン技術勉強会 Qlik Sense アプリ開発 ベーシックトレーニング (2025年Ver)

2日目：ビジネスアナリスト中級

Shohei Miwa

Senior Partner Solution Architect QlikTech Japan

# トレーニングの目的

Qlik Senseの分析アプリ開発者に必要な基本的な知識と実践的なテクニックを 1日3時間×3日間で習得します。

- 1日目：ビジネスアナリスト初級

- Qlik Senseの概要を理解することに始まり、一通りのアプリ作成の流れとシートやチャートの作成・編集などについて学習します。ロードスクリプトやチャート関数など複雑なプログラムや数式を記述をすることなく、ドラッグ&ドロップなどのマウス操作中心でセルフサービスでチャートを作成し分析できるようになります。

- 2日目：ビジネスアナリスト中級～ ← 本日

- チャート関数やSET分析などを利用した高度なチャート作成のテクニックを学習します。シート開発における複雑なチャート作成要件などに対応できるビジネスアナリスト中級レベルのスキル習得を目的とします。

- 3日目：データアーキテクト初級～中級

- Qlik Senseのデータモデリングと強力なデータ変換(ロードスクリプト)について学習します。あらゆるデータをQlik Sense上で変換・統合し、分析用のデータモデルを1から作成できるデータアーキテクトのスキル習得を目指します。

## 2日目 アジェンダ

- チャート関数でより詳細な集計・表示
  - 集計関数
  - 数値関数、演算子
  - カラー関数、レコード間関数、範囲関数
  - その他のベーシックな関数
- 自由な集計を実現するSET分析
- アプリパフォーマンスベストプラクティス
- 2日目 まとめ



チャート関数でより詳細な集計・表示

# チャート関数とは

- チャートとは、数値データをグラフィカルに表現したもので、軸・メジャー・集計方法・チャートタイプから構成されます。
- 本編で取り扱うチャート関数とは、チャートの中で定義が可能な関数、数式、演算子等を表現しています。

チャートとチャートの中で使用された関数

Sum、Total				...	×
製品種別	Q	Sum([販売価格])	Sum(total [販売価格])		
合計		2,125,461,134	2,125,461,134		
エアコン		216,265,674	2,125,461,134		
テレビ		993,418,240	2,125,461,134		
パソコン		118,289,877	2,125,461,134		
ビデオカメラ		110,157,207	2,125,461,134		
レコーダー		477,295,054	2,125,461,134		
携帯電話		210,035,082	2,125,461,134		

Sum [販売価格] > ⋮

Sum total [販売価格] v ⋮

数式

Sum(total [販売価格]) fx

ラベル

Sum(total [販売価格]) fx

数値書式

数値 ▾

書式 ☒

シンプル

1,000 ▾

# チャート関数の説明における記述の見方

後述のチャート関数の説明では、以下のフォーマットを使用しております。

- 構文

数式上に定義をする際のサポートされた文法に基づいた記載を行っております。

- 例

関数を使用した構文の一例および、この数式によって行われる処理の意味を記載しております。

- 代表的な用途

関数を使用した代表的な利用シーンを記載しております。

- 使用例

チャートのキャプチャと共に使用例を記載し、どのような利用方式であるかを記載しております。



# 集計関数

# 集計関数

集計関数は複数の項目値を入力として取得し、結果を 1 つ返す関数で構成されます。

- 集計はチャートの軸やデータロードスクリプト内の group by 条件によって定義されます。
- 集計関数には、Sum()、Count()、Min()、Max() などがあります。
- チャート関数としての集計関数の使用
  - 1つの集計関数の引数式に、別の集計関数を含めることはできません。
  - 内部集計に TOTAL 修飾子(totalの項で説明)が含まれない限り、数式に集計関数を含めることはできません。 ネストされた集計関数が必要な場合、計算軸と組み合わせて高度な集計関数 Aggr (AGGRの項で説明) を使用します。
  - 集計関数は、選択内容によって定義されたレコードセットを集計します。ただし、代替のレコード セットは、Set 分析 (SET分析の章で説明) で Set 数式を用いることで定義できます。

# 集計関数 - Sum関数

値を合計するためにはSum関数を使用します。

- 構文
  - Sum(項目)
- 例
  - Sum([販売金額])
    - 「販売金額」項目の値を合計
- 代表的な用途
  - 金額を合計する
  - 販売数、在庫数などの数を合計する
  - (1と0で構成されるような)フラグをカウントする

- 使用例：製品種別ごとの合計売上金額
  - チャートタイプ：テーブル
  - 軸：製品種別
  - メジャー：Sum([販売価格])

製品種別ごとの合計売上金額

製品種別	Q	Sum([販売価格])
合計		2,125,461,134
テレビ		993,418,240
レコーダー		477,295,054
エアコン		216,265,674
携帯電話		210,035,082
パソコン		118,289,877
ビデオカメラ		110,157,207

# 集計関数 - Count関数

レコード数を算出するためにはCount関数を使用します。

- 構文
  - Count(項目)
- 例
  - Count([製品名])
    - 「製品名」項目のレコード数を算出
- 代表的な用途
  - 製品の販売数量を算出する
  - Webページのログインユーザー数を算出する

- 使用例：製品種別ごとの伝票数
  - チャートタイプ：テーブル
  - 軸：製品種別
  - メジャー：Count([伝票日付])

製品種別ごとの伝票数

製品種別	Count([伝票日付])
合計	22,034
エアコン	2,832
テレビ	8,502
パソコン	1,085
ビデオカメラ	2,529
レコーダー	4,962
携帯電話	2,124

# total 修飾子

## 全体、軸全体の合計を算出する。

- 集計関数において全体、軸全体の集計値を算出する場合に使用
- 構文 (Sumの場合)
  - Sum(Total <項目, 項目…> 数式)
- 例
  - Sum(Total [販売金額])
    - 「販売金額」の全体の合計を算出
  - Sum(Total <製品種別> [販売金額])
    - 「販売金額」の「製品種別」毎の全体の合計を算出
- 代表的な用途
  - 全体の売上に対する各カテゴリの売上構成比
  - 棒グラフでの相対値表示
- 使用例：軸(製品種別)を無視した合計の売上
  - チャートタイプ：テーブル
  - 軸：製品種別
  - メジャー：Sum([販売価格])、Sum(Total [販売価格])

軸(製品種別)を無視した合計の売上

製品種別	Q	Sum([販売価格])	Sum(Total [販売価格])
合計		1,431,667,034	1,431,667,034
エアコン		148,914,374	1,431,667,034
テレビ		645,493,940	1,431,667,034
パソコン		82,505,077	1,431,667,034
ビデオカメラ		75,557,107	1,431,667,034
レコーダー		335,787,254	1,431,667,034
携帯電話		143,409,282	1,431,667,034

# distinct 修飾子

## ユニークな値を算出する

- 集計関数においてユニーク値を算出する場合に使用
- 構文 (Countの場合)
  - Count(distinct 項目)
- 例
  - Count(distinct [伝票日付])
    - 「伝票日付」のユニーク数を算出
- 代表的な用途
  - ECサイトの訪問者数
  - 1人当たりの集計値（売上、販売個数など）の分母

製品種別毎および全体の伝票日付のユニーク数

製品種別	Q	Count([伝票日付])	Count(distinct [伝票日付])
合計		13978	351
エアコン		1799	331
テレビ		5420	347
パソコン		693	258
ビデオカメラ		1598	327
レコーダー		3137	344
携帯電話		1331	317



アプリ確認



数值関数

演算子

# Ceil 関数

## 数値の切り上げ

- step (offset によりシフト) に最も近い倍数に数値を切り上げます。
- 構文
  - Ceil(数値 [, ステップ, オフセット])
  - ステップ：切り上げの間隔。デフォルト値は1
  - オフセット：ステップの基準。デフォルト値は0

- 例
  - Ceil(Sum( 販売価格 \* 消費税))
    - 消費税込みの販売価格を切上げて算出

- 代表的な用途
  - 消費税込み価格の金額を切上げて算出
  - 原価管理の際に原価に為替レートを乗じ、切上げて算出

値

123.456 Q

Ceil(123.456)  
Ceil(123.456,0.5)  
Ceil(123.456,0.5,0.1)

### Ceil関数を使用した出力結果と数値範囲

123.456	数値範囲
124	...0 < x <= 1, ..., 123 < x <= 124, ...
123.5	...0.5 < x <= 1.0, ..., 123.5 < x <= 124.0, ...
123.6	...0.6 < x <= 1.1, ..., 123.6 < x <= 124.1, ...

# Floor 関数

## 数値の切り下げ(切り捨て)

- step (offset によりシフト) に最も近い倍数に数値を切り下げます。
- 構文

- Floor(数値 [, ステップ, オフセット])

- 例

- Floor(Sum( 販売価格 \* 消費税))
  - 消費税込みの販売価格切り下げて算出

- 代表的な用途

- 消費税込み価格の金額を切下げて算出
- 原価管理の際に原価に為替レートを乗じ、切下げて算出

値	123.456
Floor(123.456)	123
Floor(123.456,0.4)	123.2
Floor(123.456,0.4,0.1)	123.3

Floor関数を使用した出力結果と数値範囲

123.456	数値範囲
123	$\dots 0 \leq x < 1, \dots, 123 \leq x < 124, \dots$
123.2	$\dots 0.4 \leq x < 0.8, \dots, 123.2 \leq x < 123.6, \dots$
123.3	$\dots 0.2 < x \leq 0.6, \dots, 123.3 < x \leq 123.7, \dots$

# Round 関数

## 四捨五入

- offset によりシフトされた step の最も近い倍数で切り上げた/切り下げた結果を返します。
- 構文
  - Round(数値 [, ステップ, オフセット])

- 例
  - Round(Sum( 販売価格 \* 消費税))
    - 消費税込みの販売価格を四捨五入して算出

値

Round(123.456)  
Round(123.456,0.5)  
Round(123.456,0.5,0.1)

123.456

### Round 関数を使用した出力結果と数値範囲

123.456	数値範囲
123	...0 ≤ x < 1, ..., 123 ≤ x < 124, ...
123.5	...0.5 ≤ x < 1.0, ..., 123.5 ≤ x < 124.0, ...
123.6	...0.6 ≤ x < 1.1, ..., 123.1 ≤ x < 123.6, ...

- 代表的な用途
  - 消費税込み価格の金額計算
  - 原価管理の際に原価に為替レート乗じて四捨五入して算出

# 論理演算子

- オペランド(演算対象の値)を論理的に解釈し、結果として True (-1) または False (0) を返します。

演算子	説明
not	論理否定。オペランドの論理否定を返します。
and	論理積 (and)。オペランドの論理積を返します。
or	論理和 (or)。オペランドの論理和を返します。
Xor	排他的論理和。オペランドの排他的論理和を返します。論理和と似ていますが、両方のオペランドが True の場合の結果が False になるという違いがあります。

## 計算結果

not 'ABC' = 'ABC'	0
'ABC' = 'ABC' and 'DEF' = 'DEF'	-1
'ABC' = 'ABC' or 'DEF' = 'ABC'	-1
'ABC' = 'ABC' Xor 'DEF' = 'DEF'	0

# 数値演算子

- オペランドの数値を使用し、結果として数値を返します。

演算子	説明
+	正の数値 (単項演算子) または加算を表す記号。2 つのオペランドの和を返します。
-	負の数値 (単項演算子) または減算を表す記号。単項演算子は -1 倍したオペランドを返し、二項演算子は 2 つのオペランドの差を返します。
*	乗算。2 つのオペランドの積を返します。
/	除算。2 つのオペランドの割合を返します。

## 計算結果

1 + 2	3
1 - 2	-1
1 * 2	2
1 / 2	0.5

# 関係演算子

- オペランドの値を比較し、結果としてTrue (-1) または False (0) を返します。

演算子	説明
<	小なり。未満。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。
<=	以下。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。
>	大なり。超。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。
>=	以上。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。
=	等しい。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。
<>	等しくない。両方のオペランドが数値と解釈できる場合、数値比較が行われます。比較の評価の論理値を返します。

## 計算結果

5 < 5	0
5 <= 5	-1
5 > 5	0
5 >= 5	-1
5 = 5	-1
5 <> 5	0

# 文字列演算子

- 文字列演算子は「&」と「like」の2種類があります。
  - & - 文字列連結。2つのオペランド文字列を順に連結したテキスト文字列を返します。
  - like - ワイルドカード文字列を使用した文字列比較。

演算子の前の文字列が演算子の後の文字列と一致した場合、この演算は論理値 True (-1) を返します。2番目の文字列には、ワイルドカード文字 \* (任意の数の任意の文字) または ? (1つの任意の文字) が含まれることがあります。

## 計算結果

'abc' & 'def'	abcdef
'abc' like 'a*'	-1
'abcd' like 'a?c?'	-1
'abcd' like 'a??bc'	0



アプリ確認



カラー関数

レコード間関数

範囲関数

# カラー関数

## ARGB、RGB、Color、etc...

- チャートオブジェクトのカラープロパティを設定および評価する数式やデータロード スクリプトで使します。
- 主なカラー関数

カラー関数名	説明	構文
ARGB	カラープロパティを設定または評価する数式で使用されます。色は <b>alpha</b> のアルファ係数 (不透明度) を使用した、赤の要素 <b>r</b> 、緑の要素 <b>g</b> 、青の要素 <b>b</b> によって定義されます。	ARGB(alpha,r,g,b)
RGB	カラー プロパティを設定または評価する数式で使用されます。色は、0 から 255 の値を使用した、赤の要素 <b>r</b> 、緑の要素 <b>g</b> 、青の要素 <b>b</b> により定義されます。	RGB(r,g,b)
Color	カラー マップの色番号 <b>n</b> の色表現を返す数式で使用されます。色表現はデュアル値で構成され、テキスト表現は 'RGB(r, g, b)' 形式で提供されます。	Color(n)

### 計算結果

カラー関数	q	構文	カラー表現
ARGB		ARGB(60,0,0,255)	ARGB(60,0,0,255)
RGB		RGB(0,0,255)	RGB(0,0,255)
Color		Color(1)	RGB(141,170,203)

### Colorパレット



# カラー関数

## 定義済みのカラー関数

- 定義済みの色の数式で使用できます。
- 各関数は、RGB カラー表現を返します。任意で、アルファ係数のパラメータを指定できます。その場合、ARGB カラー表現が返されます。アルファ係数 0 は完全な透明に相当し、255 は完全な不透明色に相当します。アルファの値が入力されていない場合、255 と見なされます。

### 計算結果

定義済みカラー関数	α	カラー表現
Blue()		RGB(0,0,128)
Blue(128)		ARGB(128,0,0,128)
Blue(255)		ARGB(255,0,0,128)

構文	RGB 値
black([alpha])	(0,0,0)
blue([alpha])	(0,0,128)
brown([alpha])	(128,128,0)
cyan([alpha])	(0,128,128)
darkgray([alpha])	(128,128,128)
green([alpha])	(0,128,0)
lightblue([alpha])	(0,0,255)
lightcyan([alpha])	(0,255,255)
lightgray([alpha])	(192,192,192)
lightgreen([alpha])	(0,255,0)
lightmagenta([alpha])	(255,0,255)
lightred([alpha])	(255,0,0)
magenta([alpha])	(128,0,128)
red([alpha])	(128,0,0)
white([alpha])	(255,255,255)
yellow([alpha])	(255,255,0)

# 計算軸の作成 - ValueList

データ モデルの項目ではなく、関数で生成した値を基に軸を作成

- 合成軸を形成するリストされた値のセットを返します。

- 構文

- ValueList( 静的な値 ,静的な値 , ...)

- 例

- ValueList( '都道府県' , '伝票番号' , '製品名' )

- 合成軸の内容によって使用する数式を使用

- 代表的な用途

- 項目別の統計情報の表示

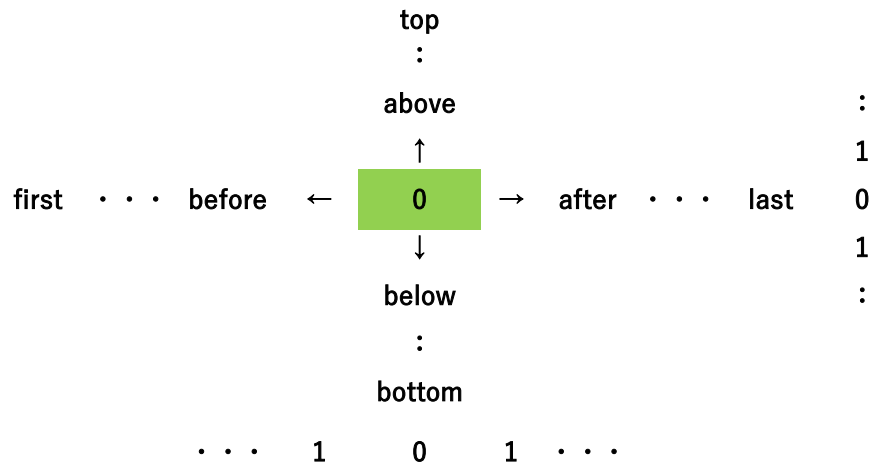
## 計算結果

Q	
=ValueList('都道府県','伝票番号','製品名')	if(ValueList('都道府県','伝票番号','製品名') = '都道府県',count(都道府県), if(ValueList('都道府県','伝票番号','製品名') = '伝票番号',count(伝票番号), count(製品名))
製品名	57
伝票番号	22,034
都道府県	1,975

# レコード間関数

## チャートでの上下左右の位置を指定

- チャート内の上下左右の位置を指定します。
- 構文
  - Above ([TOTAL] 数式 [, 位置 [, 範囲]])
  - Below ([TOTAL] 数式 [, 位置 [, 範囲]])
  - Before([TOTAL] 数式 [, 位置 [, 範囲 ]])



	2020-01				2020-02			
	sum(販売価格)	Above(sum(販売価格))	Below(sum(販売価格))	Before(sum(販売価格))	sum(販売価格)	Above(sum(販売価格))	Below(sum(販売価格))	Before(sum(販売価格))
関西支店	21,603,079	-	82,061,442	-	19,648,546	-	80,248,253	21603079
関東支店	82,061,442	21,603,079	2,635,424	-	80,248,253	19,648,546	781,840	82061442
九州支店	2,635,424	82,061,442	5,214,540	-	781,840	80,248,253	9,693,539	2635424
中国四国支店	5,214,540	2,635,424	29,573,221	-	9,693,539	781,840	24,410,881	5214540
中部支店	29,573,221	5,214,540	7,304,413	-	24,410,881	9,693,539	7,193,943	29573221
東北支店	7,304,413	29,573,221	2,751,448	-	7,193,943	24,410,881	4,365,673	7304413
北海道支店	2,751,448	7,304,413	-	-	4,365,673	7,193,943	-	2751448

# レコード間関数 - Above関数

## 前年比、前月比

- 日付型のデータ（年、年月など）を軸にした場合にAbove関数を利用して前年比、前月比を算出できます。

- 構文

- 前年比      数式 / above(数式, 12)、数式 / above(TOTAL 数式, 12)
- 前月比      数式 / above(数式)、数式 / above(TOTAL 数式)

- 例

- 軸：年月
  - 前年比     $\text{sum(販売価格)} / \text{above}(\text{sum(販売価格)}, 12)$
  - 前月比     $\text{sum(販売価格)} / \text{above}(\text{sum}() \text{数式}, 1)$

※軸が複数（例：年、月）の場合、TOTAL修飾子を使用

年月	Sum([販売価格])	sum(販売価格) / above(sum(販売価格),12)	sum(販売価格) / above(sum(販売価格))
2018-04	98,675,300	-	-
2018-05	228,284,300	-	231.3%
2018-06	75,959,100	-	33.3%
2018-07	91,550,400	-	120.5%
2018-08	74,335,400	-	81.2%
2018-09	40,735,300	-	54.8%
2018-10	44,694,300	-	109.7%
2018-11	38,460,800	-	86.1%
2018-12	1,099,200	-	2.9%
2019-01	87,529,300	-	7963.0%
2019-02	81,553,300	-	93.2%
2019-03	42,990,800	-	52.7%
2019-04	48,086,500	48.7%	111.9%
2019-05	39,178,600	17.2%	81.5%

# 範囲関数

## 値の配列を取得し、結果として1つの値を生成

- チャート内の上下左右の位置を指定します。

- 構文

- RangeSum( 数式 [, 数式])
- RangeAvg ( 数式 [, 数式])
- RangeMax( 数式 [, 数式])

- 使用例

- RangeSum( above(sum(販売価格), 0, RowNo()))
  - 売上の累積を算出
- RangeAvg( above(sum(販売価格), 0, 7))
  - 売上の移動平均を算出

年月	Q	Sum(販売価格)
2018-04		98,675,300
2018-05		228,284,300
2018-06		75,959,100
2018-07		91,550,400
2018-08		74,335,400
2018-09		40,735,300
2018-10		44,694,300
2018-11		38,460,800
2018-12		1,099,200
2019-01		87,529,300
2019-02		81,553,300
2019-03		42,990,800

rangesum(above(sum(販売価格),0,RowNo()))	RowNo()	rangeavg(above(sum(販売価格),0,12))
98,675,300	1	98,675,300
326,959,600	2	163,479,800
402,918,700	3	134,306,233
494,469,100	4	123,617,275
568,804,500	5	113,760,900
609,539,800	6	101,589,967
654,234,100	7	93,462,014
692,694,900	8	86,586,863
693,794,100	9	77,088,233
781,323,400	10	78,132,340
862,876,700	11	78,443,336
905,867,500	12	75,488,958

# 範囲関数 - RangeAvg関数

## 移動平均

- 移動平均(指定した範囲の平均)をRangeAvg関数を利用して算出できます。
- 構文
  - RangeAvg(above(数式, 0, 範囲))
- 例
  - 軸：年月
    - 12ヶ月移動平均      RangeAvg( above(sum(販売価格), 0,12))
  - 軸：日付
    - 7日移動平均          RangeAvg( above(sum(販売価格), 0,7))

年月 ▲	Q	Sum([販売価格])	rangeavg(above(sum(販売価格),0,12))	伝票日付 ▲	Q	Sum([販売価格])	rangeavg(above(sum(販売価格),0,7))
2018-04		98,675,300	98,675,300	2018/4/1		4,508,900	4,508,900
2018-05		228,284,300	163,479,800	2018/4/2		3,531,700	4,020,300
2018-06		75,959,100	134,306,233	2018/4/3		1,255,400	3,098,667
2018-07		91,550,400	123,617,275	2018/4/4		961,400	2,564,350
2018-08		74,335,400	113,760,900	2018/4/5		3,452,900	2,742,060
2018-09		40,735,300	101,589,967	2018/4/8		3,836,600	2,924,483
2018-10		44,694,300	93,462,014	2018/4/9		5,266,800	3,259,100
2018-11		38,460,800	86,586,863	2018/4/10		2,941,000	3,035,114
2018-12		1,099,200	77,088,233	2018/4/11		3,851,900	3,080,857

# 範囲関数 - RangeSum関数

## 累計

- 集計値の累計をRangeSum関数を利用して算出できます。

- 構文

- RangeSum(above(数式, 0, 範囲))

- 例

- 軸：支店、営業所

- RangeSum( above(sum(販売価格), 0, RowNo()))

- 支店ごとの営業所の売上累計

- 軸：年月

- RangeSum( above(sum(販売価格), 0, RowNo()))

- 年間売上の月ごとの累計

		Sum([販売価格])	RangeSum(above(sum([販売価格]),0,RowNo()))
中国四国支店	合計	74,175,691	0
	岡山課	30,749,864	30,749,864
	広島課	21,782,207	52,532,071
	四国西課	21,643,620	74,175,691
東北支店	合計	64,222,954	0
	宮城課	33,805,786	33,805,786
	山形課	12,207,853	46,013,639
	青森課	18,209,315	64,222,954



アプリ確認



## その他のベーシックな関数

# Concat 関数

## 項目の値の文字連結

- 文字列を組み合わせるために使用
- 構文
  - concat ([ distinct ] 文字列 [, 区切り文字 [, 連結順序]])

- 例
  - concat(distinct [製品名], ',')
  - 「製品名」のリストを算出

- 代表的な用途
  - 地域での販売製品リストの表示
  - ユーザーの購入リストの表示
  - 条件式の条件に利用

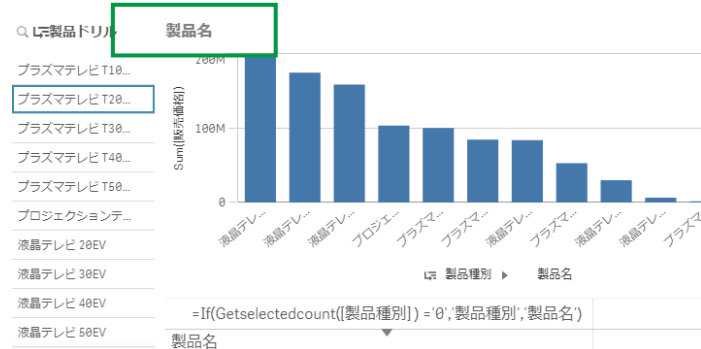
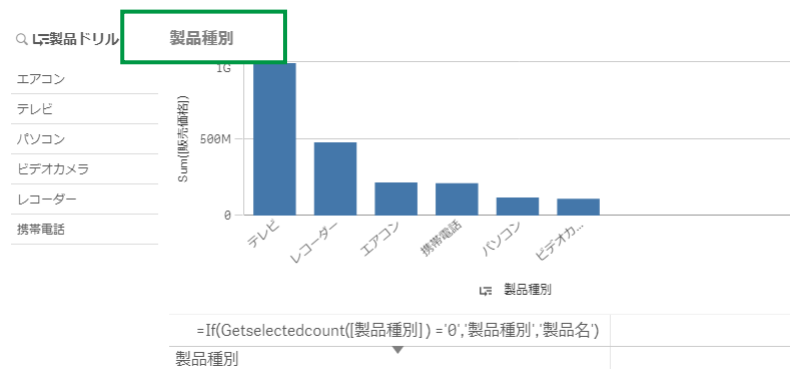
### 計算結果

営業員名	concat(distinct 製品名, ',')
安倍 礼子	DVDレコーダー 32T1, DVDレコーダー 32T3, プロジェクションテレビ J32T
井上 恵梨香	ブルーレイレコーダー AT100, プロジェクションテレビ J32T
横川 明慶	デスクトップパソコン MG200, プロジェクションテレビ J32T, 液晶テレビ 50EV
岡山 大五郎	スライド型携帯電話 420SA, デスクトップパソコン MG200, プラズマテレビ T1000, プロジェクションテレビ J32T, 液晶テレビ 20EV
河田 遥	ブルーレイレコーダー AT100, ブルーレイレコーダー AT110, プラズマテレビ T1000
河本 理紗	DVDビデオカメラ FK400, ブルーレイレコーダー AT100, ブルーレイレコーダー AT140, プラズマテレビ T2000, 液晶テレビ 50EV
会田 まひる	8畳用エアコン H21SX, DVDレコーダー 32T1, DVDレコーダー 32T2, 液晶テレビ 20EV, 液晶テレビ 50EV

# getselectedcount 関数

## 選択された項目の数を算出

- チャートなどで選択された項目の件数を算出します。
- 構文
  - GetSelectedCount (項目 [, 除外値のカウント [, 並列ステート名]])
- 例
  - If(Getselectedcount([製品種別]) = '0', '製品種別', '製品名')
  - 「製品種別」の選択状態に応じて、タイトルを切り替える



# dimensionality 関数

現在の行の軸の数を返します。

- テーブル、ピボットテーブルで表示されている軸の数を返します。

- 構文

- dimensionality ()

- 例

- If(dimensionality() = 1 , lightblue())

- 軸列の合計数が1の場合、背景をlightblueに設定

背景色の設定 : if(Dimensionality() = 1,LightBlue())

支店名 営業所名 製品種別			値	
			Sum([販売価格])	Dimensionality()
● 関西支店			427,662,356	1
● 関東支店	● 横浜課		163,221,364	2
	● 埼玉1	エアコン	5,891,582	3
		テレビ	34,373,944	3
		パソコン	4,813,826	3
		...		

# 日付,時刻関数

## 日付や時刻データを転送、変換するために使用

- 日付、時刻データを様々な形式のデータを変換します。
- 構文
  - Year(日付項目)
  - Month (日付項目)
  - Week (日付項目, 最初の曜日)
  - Hour (時間項目)
- 例
  - Year(伝票日付)
    - 伝票日付の年を表示

### 計算結果

伝票日付	q	Year(伝票日付)	Month(伝票日付)	Week(伝票日付, 0)	WeekDay(伝票日付)	Day(伝票日付)	Date(伝票日付, 'YYYYMMDD')	MonthStart(伝票日付)	Hour('12:00:00')	NetWorkDays('2020/01/01',伝票日付)
2020/1/1		2020	1月	1	水	1	20200101	2020/01/01	12	1
2020/1/2		2020	1月	1	木	2	20200102	2020/01/01	12	2
2020/1/3		2020	1月	1	金	3	20200103	2020/01/01	12	3
2020/1/4		2020	1月	1	土	4	20200104	2020/01/01	12	3
2020/1/5		2020	1月	1	日	5	20200105	2020/01/01	12	3
2020/1/7		2020	1月	2	火	7	20200107	2020/01/01	12	5
2020/1/8		2020	1月	2	水	8	20200108	2020/01/01	12	6
2020/1/9		2020	1月	2	木	9	20200109	2020/01/01	12	7

# AGGR 関数

## 指定した軸をグループピングして集計

- 指定された軸上で計算された数式の値の配列を返します。
- 構文
  - Aggr( [DISTINCT] [NODISTINCT] 数式, 軸項目 { ,軸項目} )

- 例
  - Aggr( sum(販売価格), 支店名)
    - 支店名毎の販売価格の合計値を計算

- 代表的な用途
  - 数値項目の計算軸
  - 支店内の営業所の売上割合算出

## 計算結果

支店名 🔍

営業所名 🔍

値

	Sum([販売価格])	aggr(Sum([販売価格]),支店名)	aggr(nodistinct Sum([販売価格]),支店名)
九州支店	4,110,409	4,110,409	4,110,409
鹿児島課	2,910,519	-	4,110,409
大分課	1,199,890	4,110,409	4,110,409
中国四国支店	18,570,891	18,570,891	18,570,891
岡山課	10,191,264	-	18,570,891
広島課	5,752,807	18,570,891	18,570,891
四国西課	2,626,820	-	18,570,891
中部支店	81,280,534	81,280,534	81,280,534
茨城課	14,925,683	81,280,534	81,280,534
岐阜課	9,326,177	-	81,280,534
全国平均	10,204,400	-	81,280,534

# Class 関数

## 値の範囲でのグループ化

- 数値データを任意の値の範囲でグループ化してクラス分けします。
- 構文
  - Class (項目, 範囲 [, ラベル [, オフセット ]])

- 例
  - Class(販売価格, 10000, '価格')
    - 「販売価格」の範囲でクラス分け

- 代表的な用途
  - テスト測定値のヒストグラムの作成
  - 年代別分布

### 計算結果

class(販売価格, 10000, '価格') 🔍	Count([販売価格])
0 <= 価格 < 10000	1,982
10000 <= 価格 < 20000	2,614
20000 <= 価格 < 30000	2,132
30000 <= 価格 < 40000	2,102
40000 <= 価格 < 50000	1,665
50000 <= 価格 < 60000	1,450
60000 <= 価格 < 70000	1,202
70000 <= 価格 < 80000	1,088

# Rank 関数

## 集計値の順位（ランク）付け

- 集計値データを順位付けします。
- 構文
  - Rank ([TOTAL] 項目[, 引数[,引数]])
    - 引数には数値が入り、数値に応じた avg,mix,max等のランク種別の変更
- 例
  - Rank(sum(販売価格) )
    - 「販売価格」の合計で順位付け
- 代表的な用途
  - 営業所の売上順位
  - 顧客数の順位

### 計算結果

営業所名	Q	rank(sum([販売価格]))
東京南課		1
横浜課		2
神奈川課		3
西東京課		4
千葉1課		5
東京中央2課		6
大阪3課		7
東京中央1課		8

# 文字列関数

## 文字列の取り扱いと操作を行う

- すべての関数は、データ ロード スクリプトおよびチャート数式の両方で使用できます。
- 主な文字列関数

関数名	説明	構文
Left	項目を左から文字数分切り取った文字列を返します。	Left(項目, 文字数 )
Right	項目を右から文字数分切り取った文字列を返します。	Right(項目, 文字数)
Mid	項目の開始位置から文字数分切り取った文字列を返します。	Mid(項目, 開始位置, 文字数)
Subfield	区切り文字で区切られた部分から指定した番号の文字列を返します。	Subfield(項目, 区切り文字, 番号)
Replace	対象文字を置換文字に置き換えた文字列を返します。	Replace(項目, 対象文字, 置換文字)
Len	項目の文字数を返します。	Len(項目)

### 計算結果

年月	Q	Left(年月,4)	Right(年月,2)	Mid(年月,3,5)	Subfield(年月,'-',1)	Replace(年月,'2020','2021')	Len(年月)
2020-01		2020	01	20-01	2020	2021-01	7
2020-02		2020	02	20-02	2020	2021-02	7
2020-03		2020	03	20-03	2020	2021-03	7
2020-04		2020	04	20-04	2020	2021-04	7



アプリ確認

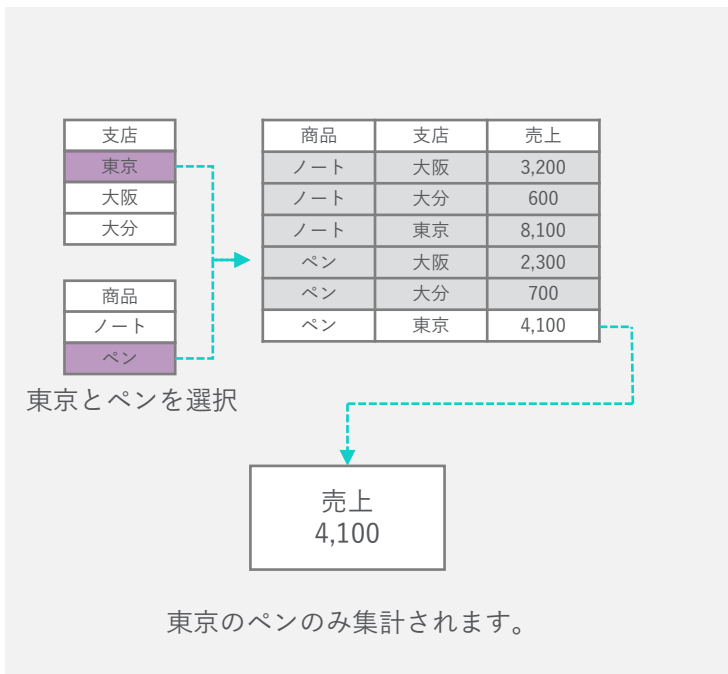


# 自由な集計を実現する SET分析

# 自由な集計を実現するSET分析

連想技術により、選択された値に関連するデータのみ表示・集計されます。

## 通常のデータモデルの挙動



## SET分析

現在の選択とは異なる範囲を表示・集計します。  
例えば次のような場合に使用します。

- 異なる範囲を並べて比較

東京売上 12,200	大阪売上 5,500
----------------	---------------

- 前年比・全体構成比などを計算

支店	構成比
東京	64%
大阪	29%
大分	7%

- 常に特定の範囲を除外

ノート売上 (大分除く) 11,300
---------------------------

# SET分析の基本構文

## SET数式

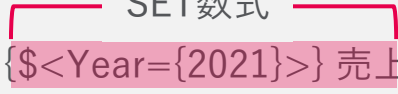
通常の数式

関数(対象項目)

例 Sum(売上)

SET分析の数式

関数({SET数式}対象項目)

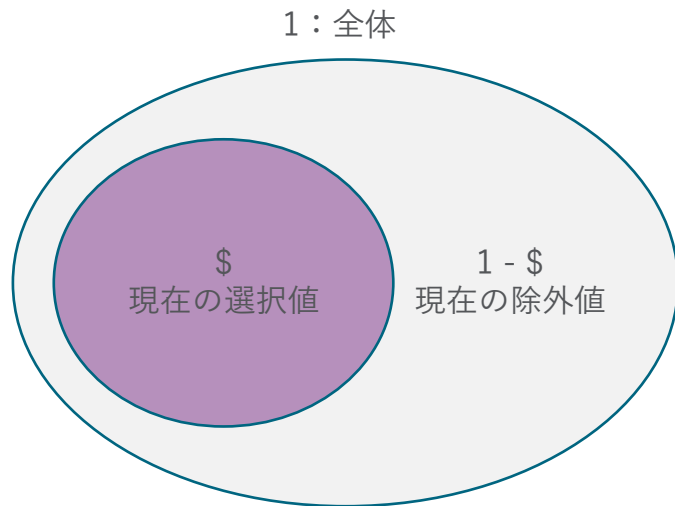
例 sum(  \$<Year={2021}> 売上 )

SET数式は波括弧{}で囲む必要があります。  
ここで、演算の範囲を指定します。  
識別子と修飾子から構成されます。

# SET数式の識別子

## 識別子の基本的な形式

SET分析において、選択状態やブックマークの状態を識別します。



識別子	集計対象
1	選択を無視
\$	現在の選択状態
\$1	「1段階戻る」ボタンと同じで、1つ前の選択状態
\$_1	「1段階進む」ボタンと同じで1つ次の選択状態
ブックマーク名	ブックマークの選択状態

識別子を省略すると\$:現在の選択の意味となります。

# 識別子を使ったSET数式の例

## ① データ

日付	支店	商品	売上
2020/03/25	東京	ノート	4,200
2020/03/25	東京	ペン	2,000
2020/04/25	大阪	ノート	1,800
2020/04/25	大阪	ペン	1,000
2020/05/25	大分	ノート	600
2020/05/25	大分	ペン	700
2021/02/25	東京	ノート	3,900
2021/02/25	東京	ペン	2,100
2021/06/25	大阪	ノート	1,400
2021/06/25	大阪	ペン	1,300

## ② 選択

まず「東京」を選択、その後「大阪」を選択

ひとつ前の選択

支店
東京
大阪
大分



現在の選択

支店
東京
大阪
大分

## ③ ブックマーク

BM01に「東京」と「大阪」を設定

支店
東京
大阪
大分

数式	集計対象	商品別テーブルの結果	
Sum(売上)	現在の選択「大阪」を集計	ノート	3,200
		ペン	2,300
Sum({1} 売上)	選択を無視してすべて集計	ノート	11,900
		ペン	7,100
Sum({\$} 売上)	現在の選択「大阪」を集計	ノート	3,200
		ペン	2,300
Sum({\$1} 売上)	1つ前の選択状態「東京」を集計	ノート	8,100
		ペン	4,100
Sum({BM01} 売上)	ブックマークの選択状態「東京」「大阪」を集計	ノート	11,300
		ペン	6,400

# SET数式の修飾子

## 修飾子の基本的な形式

SET修飾子は選択された範囲を変更するために使用します。<> で囲みます。

例：Sum( { \$<SET修飾子> } 売上)

SET修飾子の構文：< 項目名1 = 値,項目名2 = 値,... >

値は {} で囲みます。

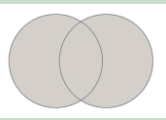
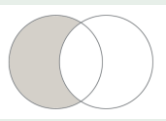
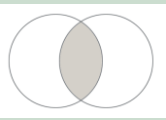
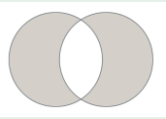
数式	集計対象	売上合計の結果（選択無し）
Sum(売上)	全ての売上	19,000
Sum({\$<支店={'大阪'}>} 売上)	大阪の売上	5,500
Sum({\$<支店={'東京','大阪'}>} 売上)	東京と大阪の売上	17,700
Sum({\$<支店={"大*"}>} 売上)	支店が「大」で始まる売上	6,800
Sum({\$<支店=>} 売上)	全ての支店の売上	19,000
Sum({\$<商品={'ノート'}, 支店={'東京'}>} 売上)	ノートの東京の売上	8,100

SET修飾子で記述されていない項目は、現在の選択が適用されます。数値はそのまま、文字の値は「'」シングルクォートで囲みます。ただしワイルドカードや演算子・等号で始まる場合は「"」ダブルクォートで囲みます。

# SET数式の演算子

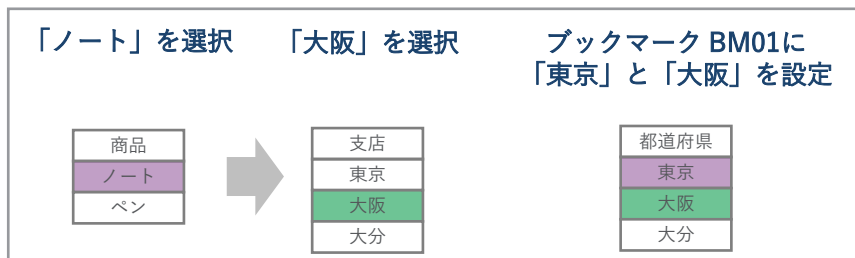
## 演算子の種類

識別子が2つ以上ある場合に、それぞれの識別子によって表されるデータ セットをどのように組み合わせるかを指定して、集計範囲を指定することができます。

演算子	名称	集計対象	
+	Union 和集合		二つのSET識別子のいずれかに属するデータすべて
-	Exclusion 排除		二つのSET識別子のうち、最初の SETにのみ属するデータ
*	Intersection 共通集合		双方の SET 識別子に属するデータ
/	Symmetric Difference 排他的論理和		両方ではなくどちらか一方の SET 識別子にのみ属するデータ

# SET数式の演算子

## 演算子を使ったSET数式の例



数式	集計対象	売上合計の結果
Sum(売上)	大阪のノートの売上	3,200
Sum({\$<支店 += {'東京'}>} 売上)	現在の支店の選択に東京を追加した売上	11,300
Sum({1 - \$} 売上)	全体から現在の選択を除外した売上	15,800
Sum({\$ * BM01} 売上)	現在の選択とブックマークBM01の共通範囲の売上	3,200
Sum({\$<支店 /= {"大*"}>} 売上)	現在の支店の選択、または「大」で始まる支店のいずれかの売上	600



SET 演算子を複数のテーブルからの項目を含む基本集計関数とともに使用するの、予測不可能な結果の原因となりますので、避けて下さい。例えば、*Quantity* と *Price* が別々のテーブルの項目の場合sum({\$\*BM01}Quantity\*Price) という数式は避けて下さい。

# 入力補助を利用してSET分析を実装する

## 数式エディタの利用 1 - SET数式を含む数式の自動作成

1. 作成したいSET数式に合わせて選択します。

「ノート」を選択 「大阪」を選択

商品	支店
ノート	東京
ペン	大阪
	大分

2. オブジェクトを作成し、数式エディタを開きます。

3. 数式エディタのプロパティパネルの項目タブで、項目と集計関数を選び、「set数式」をチェックして、挿入ボタンをクリックします。

4. SET数式を含む数式が追加されます。

**Sum**([<支店=['大阪'],商品=['ノート']>] **[売上]**)

# 入力補助を利用してSET分析を実装する

## 数式エディタの利用 2 - SET数式のみ自動作成

1. 前頁と同様に作成したいSET数式に合わせて選択し、オブジェクトの数式エディタを開きます。
2. set数式タブで「現在の選択条件を使用」をチェックして、挿入ボタンをクリックします。



3. SET数式部分のみ作成されるので、適切な場所に配置します。

{<支店=['大阪'],商品=['ノート']>}

4. 「ブックマークを使用」を選択した場合は、ブックマークを選ぶと、ブックマークに適用されている選択のSET数式が作成されます。



{<支店=['東京','大阪']>}

# より高度なSET分析を実装する

## 数値の範囲を指定する

不等号で指定された範囲が集計対象となります。範囲は一方のみの指定でもかまいません。日付の範囲を指定するような場合に使用します。

`Sum({$<日付={">2020/04/30 <=2021/02/25"}>}売上)`

指定する範囲が等号や不等号で始まる場合は、「"」ダブルクォートで囲みます。日付が2020/04/30より大きく、2021/02/25以前のデータのみ集計されます。

集計対象  
7,300

日付	売上
2020/03/25	4,200
2020/03/25	2,000
2020/04/25	1,800
2020/04/25	1,000
2020/05/25	600
2020/05/25	700
2021/02/25	3,900
2021/02/25	2,100
2021/06/25	1,400
2021/06/25	1,300

# より高度なSET分析を実装する

## 変数を使用する

変数を利用して集計範囲を設定することができます。

データロードや選択に応じて変数が変わるように設定しておく、動的に集計範囲を変更することもできます。

変数 `vToday` に常に今日の日付が入るとすると、次の数式は、常に今日の売上を集計します。

`vToday` `'2021/02/25'`

`Sum({$<日付={$(vToday)}>}売上)`

売上  
6,000

変数を使用する場合は `$()` で囲みます。

# より高度なSET分析を実装する

## 関数を使用する

関数の結果を集計範囲の条件に使用することができます。

関数は “\$()” で囲んで使用します。

`Sum({$<[日付.autoCalendar.Year]={"$(<Year(Max(日付)))"}>}売上)`

当年（日付の年が最大）の売上が集計されます。

売上  
8,700

`Sum({$<[日付.autoCalendar.Year]={"$(<Year(Max(日付))-1")"}>}売上)`

昨年（日付の年が最大の1年前）の売上が集計されます。

売上  
10,300



データマネージャーで日付項目を取り込むと、自動的にautoCalendar項目が作成されます。  
Yearのほかに、Month、Week、Day、Quarter があります。

# より高度なSET分析を実装する

## 指定条件に合致する範囲を取得する

売上が5,000円以上の支店（東京・大阪）のみ集計するというように、条件に合致するメンバー（値の一覧）を取得して集計対象として計算することができます。集計対象を指定する数式は「」ダブルクォートで囲みます。

Sum({\$<支店={"=Sum(売上)>5000"}>}売上)



支店名でなく、売上が5000円以上という条件にあてはまる支店を対象としています。

売上  
17,700

さらに条件の数式にSET数式を加えることもできます。  
2020年に売上が5,000円以上の支店（東京）のみ集計します。

Sum({\$<支店={"=Sum({<[日付.autoCalendar.Year]={2020}>}売上)>5000"}>}売上)

売上  
12,200

# より高度なSET分析を実装する

## 選択による絞込値と除外値の利用

選択による関連性を利用して、関連のある絞込値や関連のない除外値の範囲を集計することができます。

「2021」を選択

日付.Year
2020
2021

支店
東京
大阪
大分

絞込値 : P

除外値 : E

2021年を選択した場合に売上が存在する支店の売上

$\text{Sum}(\{\$ < \text{支店} = P(\{ < \text{日付.autoCalendar.Year} = \{2021\} > \}) > \} \text{売上})$

売上  
17,700

2021年を選択した場合に売上が存在しない支店の売上

$\text{Sum}(\{\$ < \text{支店} = E(\{ < \text{日付.autoCalendar.Year} = \{2021\} > \}) > \} \text{売上})$

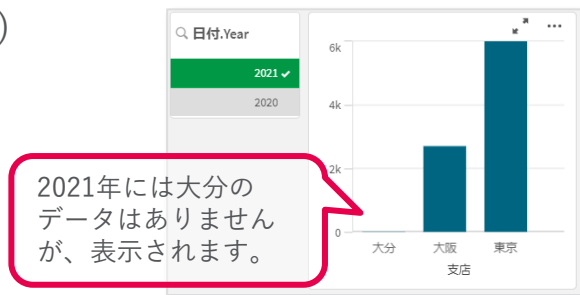
売上  
1,300

# SET識別子{1}の便利な使い方

{1}は現在の選択に関わらず全体を集計範囲とするので、次のような場合に使用すると効果があります。

選択により値のなくなる軸も常に表示する。

Sum(売上)+ 0 \* Sum({1}売上)



選択に関わらず最新年を集計する。

Sum({\$<日付.autoCalendar.Year={"\$(=Year(Max({1}日付)))"}>}売上)

売上  
8,700

# 新しいSet数式の構文

Set数式を集計の外側にも記述できるようになりました。これにより構文が簡素化され、記述しやすくなります。

これまで説明したSet数式

Sum( {\$<Year={2021}>} 売上 )

- 複数の集計がある場合、何度もSet数式を書く必要がある
- マスターメジャーと組合せが困難



外部のSet数式

{<Year={2021}>} Sum(売上) / Count(distinct Customer)

複数回のSet数式を書く必要がない

{<Year={2021}>} [Master Measure]

マスターメジャーと組合せられる

# 外部Set数式の書き方

- 複数の集計を含む式があり、すべての集計関数で同じ Set 式を繰り返し記述する手間を省略したい際に、集計関数の外で Set 式を使用します。
- 外部Set数式を使用する場合は、式の先頭に配置する必要があります。

`{<Year={2021}>}` Sum(売上) / Count(distinct Customer)

- 外部Set数式を使用すると、既存のマスターメジャーにもSetを適用できます。

`{<Year={2021}>}` [Master Measure]

# 外部Set数式の書き方

- レキシカルスコープ

- 外部Set数式と集計式を括弧で囲むことで、外部Set数式の適用範囲を定義できます。
- 括弧を使用しない場合、外部Set式は数式全体に影響します。

( {<Year={2021}>} Sum(Amount) / Count(distinct Customer) ) - Avg(CustomerSales)

括弧内の数式にのみ、{<Year={2021}>}を適用

- 例：「2021年の東京支店の売上合計」と「東京支店全体の売上平均」のギャップを示す指標

( {<日付.autoCalendar.Year={2021}, 支店={"東京"}>} Sum(売上) / Count(distinct 支店) ) - Avg({<支店={"東京"}>} 売上)

# SET分析のまとめ

- SET分析は、現在の選択とは異なる範囲を表示・集計します。
- SET分析の基本構文は、関数({SET数式}対象項目)、SET数式は識別子と修飾子で構成されます。
- 識別子は選択状態やブックマークの状態を識別します。例えば「\$」は現在の選択、「1」は全体など。
- 修飾子は選択された範囲を変更するために使用します。例：Sum({ \$<SET修飾子>} 売上)
- 演算子を使うと識別子によって表されるデータ セットの組み合わせを指定できます。+ - \* /
- 外部SET数式はマスターメジャーとの組み合わせや、複数のSET数式を定義せずに利用する事ができます。
- 高度なSET分析として以下のような使い方ができます。
  - 数値の範囲を指定する
  - 変数や関数を使用する
  - 指定条件に合致する範囲を取得する



アプリ確認



# アプリパフォーマンス ベストプラクティス

# アプリパフォーマンス ベストプラクティス

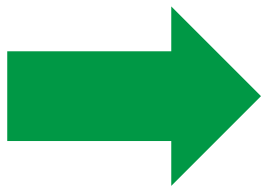
- Qlik Senseはインメモリによる高速な集計を実現していますが、データ量の増加やシート設計の複雑さなどの原因により思うような集計・描画のパフォーマンスが得られないケースがあります。
- 本章では、アプリパフォーマンスが落ちるときの典型的な原因を紹介します。
- 本章のポイントに気を配りながらアプリを開発することで、ユーザーがアプリを快適に利用できるようにしましょう。

# 数式内のIF文の使い方に気を付ける

- If条件のネスト(入れ子)は集計パフォーマンスに影響します。例えば、Pick(match())と置き換えられないか検討しましょう。

ネストが深いIf条件

```
if([地域]='関東',100,  
  if([地域]='関西',110,  
    if([地域]='中部',90,  
      if...  
    if... )))))
```



pickとmatchで書き換え

```
pick(match([地域],'関東','関西','中部'), 100,110,90)
```

Pick() は、指定した配列の値をリストから取得する。  
Match() は、指定した値がリストの 何番目 にあるかを返す。  
Pick(Match())` を組み合わせることで、  
「条件によって異なる値を返す」処理が可能。

66

# 数式内のIF文の使い方に気を付ける

- IF条件文では、文字列比較よりも数値比較の方がパフォーマンスが良いです。さらに、Set数式の方がIF条件文よりもパフォーマンスが良いので、該当するIF条件文の置き換えが可能な検討しましょう。

パフォーマンス

低

`Sum(If([地域]='関東', Sales) )`

`Sum(If([地域No]=1,Sales))`

高

`Sum({<[地域No]={1}>} Sales)`

# 演算実行条件で演算リソースを制御

- テーブルやピボットテーブルなど計算量が多くなりがちなチャートに対しては、データをフィルタ等である程度絞り込んだ後に表示(演算)するよう演算実行条件で制御しましょう。

軸・メジャー・レコード数が多く、  
結果的に表示にかかる都度の演算量が膨大になりがちな明細テーブル

販売明細

発注日	発注	四半期	店別	商品区分名	商品名	販売開始日	クラス	セグメント	出店/都道府県	地域	取引先名	発注コード	単位	数量	売上	Avg(平均単価)	Sum(総計)
1/14/2017	2017	Q1	2017-ann	卸売	なまじょうが	南 裕子	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1054-023	200	50	4520	0.29	1190
1/18/2017	2017	Q1	2017-ann	卸売	なまじょうが	南川 秀人	シルバー	レストラン カフェ	東京都	関東	みちのく本舗	1054-024	200	50	6700	0.42	2050
1/24/2017	2017	Q1	2017-ann	卸売	ピリカラカシコ	中田 実	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1057-023	200	50	6600	0.29	1100
1/24/2017	2017	Q1	2017-ann	卸売	ピリカラカシコ	中田 実	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1057-042	200	50	9980	-0.17	-1680
1/24/2017	2017	Q1	2017-ann	卸売	ピリカラカシコ	中田 実	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1057-007	200	50	2344	0.21	454
1/25/2017	2017	Q1	2017-ann	卸売	オタシ白ラベル	小田 健治	シルバー	レストラン カフェ	徳島県	関西	協成でん	1059-049	300	80	20180	0.47	9440
1/18/2017	2017	Q1	2017-ann	卸売	オタシ白ラベル	南川 秀人	シルバー	レストラン カフェ	東京都	関東	みちのく本舗	1054-022	300	60	14400	0.88	12700
1/14/2017	2017	Q1	2017-ann	卸売	オタシ白ラベル	南 裕子	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1054-020	300	50	7020	0.47	3230
1/24/2017	2017	Q1	2017-ann	卸売	スポーツ飲料パワー	中田 実	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1057-041	180	50	7980	0.75	5860
1/25/2017	2017	Q1	2017-ann	卸売	ナイトワイン	小田 健治	シルバー	レストラン カフェ	徳島県	関西	協成でん	1059-051	500	50	23780	0.96	22660
1/24/2017	2017	Q1	2017-ann	卸売	バードワイン	中田 実	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1057-037	250	50	9425	0.56	5425
1/25/2017	2017	Q1	2017-ann	卸売	バードワイン	小田 健治	シルバー	レストラン カフェ	徳島県	関西	協成でん	1059-050	250	30	9100	-0.1	-460
1/14/2017	2017	Q1	2017-ann	卸売	バードワイン	南 裕子	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1054-021	250	40	8300	0.34	2790
1/25/2017	2017	Q1	2017-ann	卸売	ピリビリアール	小田 健治	シルバー	レストラン カフェ	徳島県	関西	協成でん	1059-048	200	100	21580	0.66	14580
1/18/2017	2017	Q1	2017-ann	卸売	ピリビリアール	南川 秀人	シルバー	レストラン カフェ	東京都	関東	みちのく本舗	1054-021	200	50	12320	0.96	11790
1/24/2017	2017	Q1	2017-ann	卸売	ピリビリアール	南 裕子	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1054-023	280	50	8680	-0.34	-2680
1/25/2017	2017	Q1	2017-ann	卸売	ボトルワイスキー	小田 健治	シルバー	レストラン カフェ	徳島県	関西	協成でん	1059-052	1200	50	36490	0.17	6190
1/31/2017	2017	Q1	2017-ann	卸売	林檎果汁	山神 裕子	ゴールド	ホテル	茨城県	関東	高淳亭	1060-064	100	50	3700	0.21	790
1/24/2017	2017	Q1	2017-ann	卸売	林檎果汁	南川 秀人	ゴールド	ヘルスクア	静岡県	中部	自然派のチヤラ	1059-047	100	50	3680	1	3640
1/31/2017	2017	Q1	2017-ann	卸売	清涼レモン	山神 裕子	ゴールド	ホテル	茨城県	関東	高淳亭	1060-062	180	50	7885	0.34	4175
1/24/2017	2017	Q1	2017-ann	卸売	清涼レモン	南 裕子	ゴールド	レストラン カフェ	東京都	関東	みちのく本舗	1054-022	180	50	8825	0.03	275
1/18/2017	2017	Q1	2017-ann	卸売	こちぢははん	南川 秀人	シルバー	レストラン カフェ	東京都	関東	みちのく本舗	1054-023	180	50	7200	0.61	4360
1/25/2017	2017	Q1	2017-ann	卸売	ふりかけ梅干味	南川 秀人	ゴールド	コンビニエンスストア	静岡県	関東	協成でん	1060-054	1300	50	58900	0.03	1720
1/25/2017	2017	Q1	2017-ann	卸売	ふりかけ梅干味	南川 秀人	ゴールド	コンビニエンスストア	静岡県	関東	協成でん	1060-055	1300	50	87200	0.55	31620
1/18/2017	2017	Q1	2017-ann	卸売	清涼レモン	南川 秀人	シルバー	レストラン カフェ	東京都	関東	みちのく本舗	1054-026	300	50	11300	0.33	5870

## 演算実行条件で演算リソースを制御

- チャートプロパティの拡張機能タブにデータの処理＞演算実行条件の設定項目があります。

テーブルのプロパティ > 拡張機能 > データの処理

[illegible]

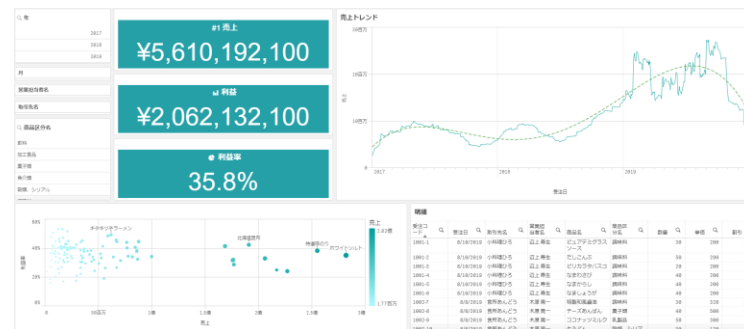
# シート上のオブジェクト数を減らす

- ひとつのシートに大量のオブジェクト(チャート等)を配置するとシートを開いたときに多くの演算リソースを使用することになりパフォーマンスに影響します。オブジェクト数を減らせないか検討しましょう。

悪い例



良い例



むやみに大量のオブジェクトを配置せず、必要最低限に

# 計算軸の利用を避ける

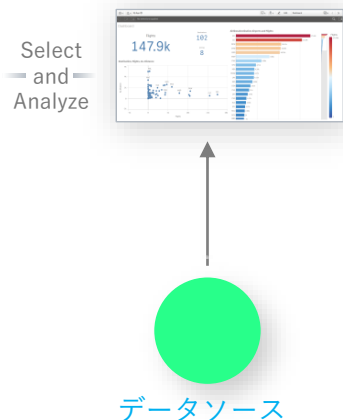
- 計算軸（軸に数式ロジックを組み込むこと）を多用すると演算量が増えます。データモデル側に予め項目として組み込めないか検討しましょう。

DimensionType	DimensionName	Object Count
<b>Totals</b>		<b>101</b>
Calculated	=[Field]	1
Calculated	=Dual(Year([OrderDate])&'-'&Month([OrderDate]),MonthStart([OrderDate]))	2
Calculated	=[Table]	1
Calculated	=aggr(only({<Division={'EMEA'}>}[Company Name]), [Company Name])	1
Calculated	=ValueLoop(1,100)	1
Calculated	=Aggr(	2

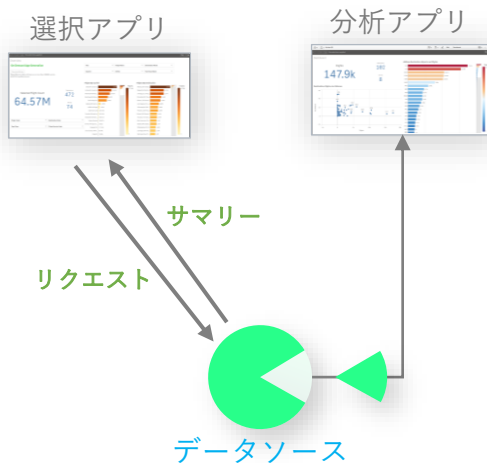
# ライブクエリを検討する

- データ量が膨大になりインメモリでも高速なパフォーマンスを得られない場合は、ライブクエリ機能であるダイナミックビューやオンデマンドアプリ生成、Direct Query といった処理の検討をしましょう

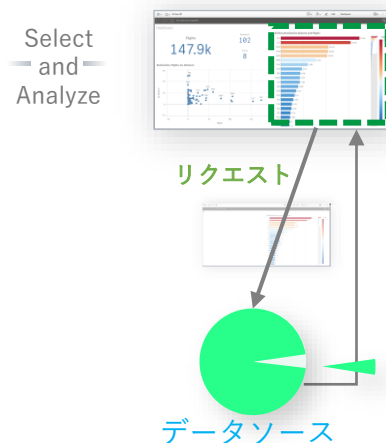
インメモリキャッシング



オンデマンドアプリ生成  
(アプリ単位のライブクエリ)



ダイナミックビュー  
(チャート単位のライブクエリ)



# データモデルの最適化

その他、データモデリングにおいてもシートのパフォーマンスを保つ、あるいは向上させる方法がいくつかあります。

- 合成キーを排除する
- タイムスタンプ (YYYY/MM/DD hh:mm:ss) を日付と時刻に分割する
- 分析に使用しないほど粒度の細かいデータを集計する
- 未使用の項目を削除する
- 項目の少ないテーブルを非正規化する (スノーフレイクスキーマのスタースキーマ化)
- データ量が非常に大きい場合、細分のために中間テーブルを利用するのではなく、Concatenate句(異なるデータセットを中央ファクトテーブルへ追加する関数)の利用を検討する。
- 文字列キーよりも数値キーのほうがメモリ効率が良いため、複合キー項目を数値化する。

# アプリパフォーマンス ベストプラクティス まとめ

- Qlik Senseはインメモリによる高速な集計を実現していますが、データ量の増加やシート設計の複雑さなどの原因により思うような集計・描画のパフォーマンスが得られないケースがあります。
- 以下のポイントに気を配りながらアプリを開発することで、ユーザーがアプリを快適に利用できるようにしましょう。
  - IF条件文の使い方
  - 演算実行条件
  - シート上のオブジェクト数
  - 計算軸
  - ライブクエリの検討
- アプリ公開後にデータ量が増えたり、ユーザーがチャートを追加したりすることによってパフォーマンスが落ちる可能性があります。このようなケースにおいても以上のようなアプリの検証や改善は有効なケースが多く存在します。



## 2日目 まとめ

## 2日目 まとめ

- 2日目で学習した内容
  - より詳細な集計・表示を実現するチャート関数
  - 自由な集計を実現するSET分析
  - Qlik Senseの性能を最大限活かすアプリ開発におけるパフォーマンスベストプラクティス

- 本日紹介した基本的な関数・SET数式の使い方になれることでシート上で様々な集計や制御を行うことができるようになります。
- 本日紹介したもの以外にも数百の関数がQlik Senseには用意されていますので実現したい集計など要件に応じて使える関数を増やしていくことでさらなるスキルアップが望めます。



# Q & A

# Q&A一覧

ご質問	回答
above関数を使用すると、列内が一つ下にずれる理由を教えてください。	above関数はデフォルトでは現在の行の1行上を参照して計算する為、一段下の行に結果が求まる仕様となっています。
aggr(Sum([販売価格]),支店名)列において、鹿児島課や岡山課などに値が表示されていないのはなぜですか？	Aggrで支店名別に計算を行うのに対して、テーブルの軸が支店より細かい課となっています。 Aggr支店と軸の組み合わせにつき1回計算される為、九州支店の最初に取得される組み合わせが鹿児島課のデータである場合、大分課では結果が戻らないこととなります。  このような場合にNODISTINCTを使用して計算を繰り返すと金額が求まります。
Aggr関数の軸項目に、階層型のマスターアイテムを指定して、選択状況に合わせた結果を求めるといった事は可能でしょうか	Aggr関数で階層型のマスターアイテムを軸に指定することはできません。
シート状のオブジェクト数がパフォーマンスに影響するとの事ですが、アプリにシートを増やせば増やした場合も影響しますでしょうか。または直接影響するのは、表示中のシートのみでしょうか	ユーザーの選択などのアクションに対するレスポンスへの直接的な影響としては表示中のシートのみです。
ストレートテーブルでSET関数を利用したいのですが、可能でしょうか？	ストレートテーブルでもSET数式を利用することは可能です。



Thank you