Budgeting:

* How to handle fact table data (e.g., Sales) at a finer "grain" than the available budget data (e.g., Sales data is at the "day" grain while budget data is at the "year" grain) DIFFERENT GRANULARITY
* Step 1: Create a table
	+ Column 1: dates[date] (i.e., days)
	+ Column 2: total sales
	+ Column 3: total budget = date context portion of the annual budget
		- Days in Year = CALCULATE( COUNTROWS( Dates ), ALLEXCEPT( Dates, Dates[Year] ) )
		- Days in Date Context = COUNTROWS( Dates )
* Step 2: Budget Allocation = ([Days in Date Context] / [Days in Year] \* [Total Budget]

Forecasting:

* Create core measures for past years (e.g., [Total Sales])
* Create forecast measures for the next year
	+ e.g., Sales Forecast 10% = 10% [Total Sales] \* 0.10
		- i.e., use your historical data to set a target for future data

Cumulative Totals:

* Use CALCULATE to change the context from the particular selected date to all dates before and including the current date
* Cumulative Sales =
	+ CALCULATE( [Total Sales],
		- FILTER( ALLSELECTED( Dates ),
			* Dates[Date] <= MAX( Dates[Date] ) ) )

Ranking:

* Use RANKX to determine the ranking of the group (e.g., ranking of customers by sales, descending [i.e., the customer with the highest sales is ranked 1[first] while the customer with the lowest sales is ranked last)
* Customer Rank = RANKX( ALLSELECTED( Customers ), [Total Sales], , DESC )
* Top 7 Customers =
	+ VAR MaxRank = 7
	+ RETURN
	+ IF( CustomerRank <= MaxRank, [Total Sales], BLANK() )

Moving Averages:

* Use AVERAGEX
* Sales 1M MA =
* AVERAGEX(
	+ DATESINPERIOD( Dates[Date], LASTDATE( Dates[Date] ), -1, MONTH ),
		- [Total Sales] )

Grouping / Segmentation:

* Create supporting table with the groups of interest (e.g., invoice aging groups)
* Create calculated column to show calculation of interest (e.g., invoice aging days)
	+ Aging Days = DATEDIFF( Invoices[Transaction Date], TODAY(), DAY )
* Sales Growth per Group =
	+ CALCULATE( [Total Sales],
		- FILTER( VALUES( Customers[Customer Name] ),
		- CONTROWS(
			* FLITER( SalesGrowthGroups,
			* [Sales Growth] >= SalesGrowthGroups[Min]
			* && [Sales Growth] < SalesGrowthGroups[Max] ) ) > 0 ) )
			* (// using countrows > 0 to return only the row in the group table that is applicable)
	+ // showing customer sales by growth segment

Hide Future Dates:

* Add 1 new column to Dates table
	+ DatesWithSales = Dates[Date] <= MAX( Sales[OrderDate] )
* Modify Measure to use new column as a filter (need to wrap time intelligence function in CALCULATETABLE)
	+ Sales YTD hide = CALCULATE( [Sales Amount], CALCULATETABLE( DATESYTD( Dates[Date] ), Dates[DatesWithSales] = TRUE ) )

How to fix total errors for measure columns:

* OOTB totals don’t have the right context for complex total errors
* Create a virtual table of just the results you want to display
	+ This will provide a new context for the calculation
* Minimum Testing =
* SUMX(
	+ SUMMARIZE( VALUES( Sales[Customer ID] ),
	+ “Sales”, [Total Sales],
	+ “Sales LY”, [Sales LY] ),
		- MIN( [Sales], [Sales LY] ) )
* <https://www.sqlbi.com/articles/obtaining-accurate-totals-in-dax/>

Events In Progress:

* e.g., when you have 2 dates in your fact table (e.g., Order Date and Ship Date) and you want to find our how many orders are "active" in the period (between order date and ship date)
* Setup 2 relationships between [Dates] table and [Sales] table and ensure both are **INACTIVE**
* Sales in Progress =
* CALCULATE( SUM( Sales[TotalRevenue] ),
	+ FILTER( VALUES( Sales[OrderDate] ), Sales[OrderDate] <= MAX( Dates[Date] ) ),
	+ FILTER( VALUES( Sales[ShipDate] ), Sales[ShipDate] >= MIN( Dates[Date] ) ) )
* (don't use a measure that is OUTSIDE the main table; rather, use a SUM of a table column or SUMX of related table columns)

Lost Customers:

Lost Customers =

* VAR CustomersPurchased = CALCULATE( VALUES( Sales[Customer ID] ),
	+ FILTER( ALL( Dates ),
		- Dates[Date] > MIN( Dates[Date] ) - 365 &&
		- Dates[Date] <= MIN( Dates[Date] ) - [Churn Time Frame Value] ) )
* VAR PriorCustomers = CALCULATE( VALUES( Sales[Customer ID] ),
	+ FILTER( ALL( Dates ),
		- Dates[Date] > MAX( Dates[Date] ) - [Churn Time Frame Value] &&
		- Dates[Date] <= MAX( Dates[Date] ) ) )
* RETURN
* COUNTROWS( EXCEPT( CustomersPurchased, PriorCustomers ) ) \* -1