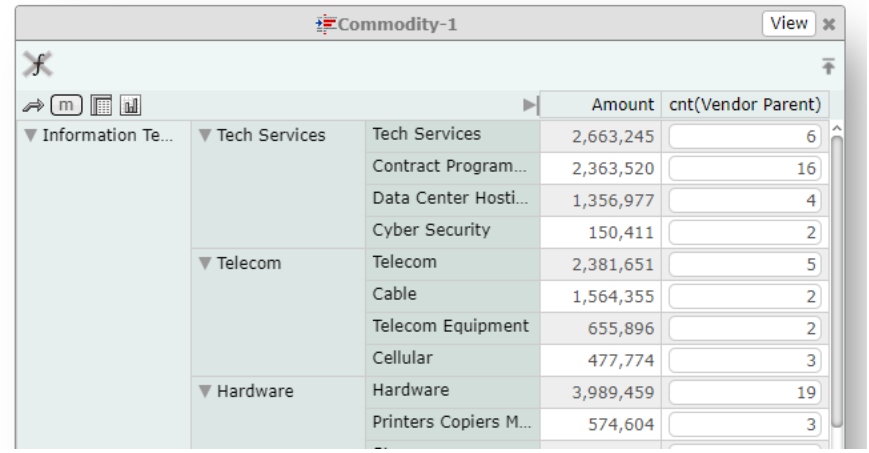
View-Based Measure Columns

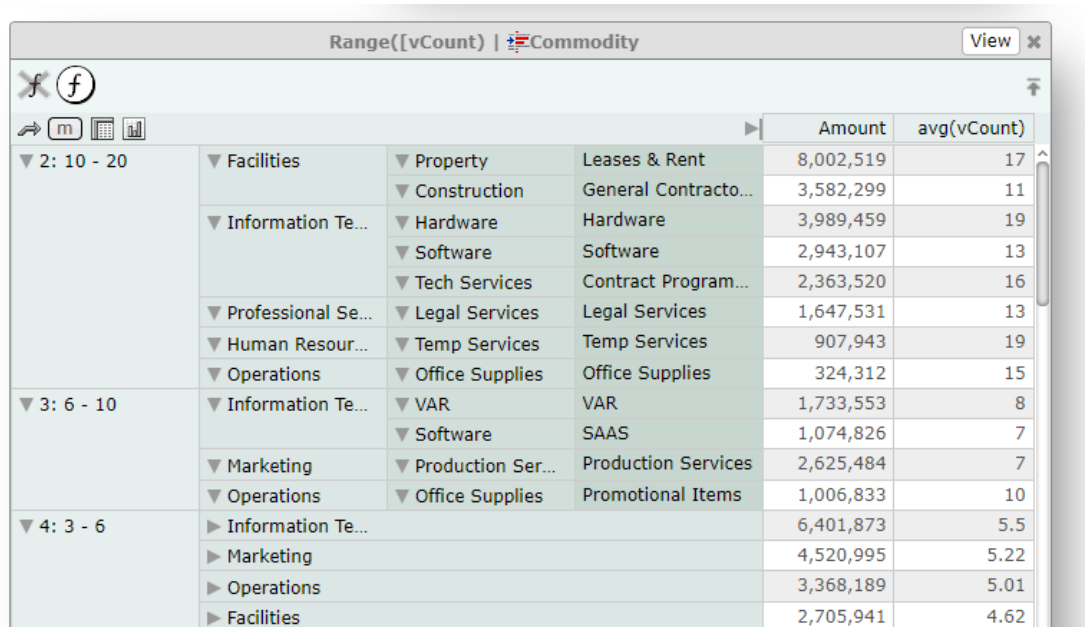
It's often useful to reference a rolled-up measure outside the context of the View that calculates it; in other words, to turn a View-based measure into a real transaction-based measure that can be accessed anywhere. And, because the View-based measure is dynamic, so also is the transaction-based measure, allowing for advanced real-time modeling.

Here is a simple example. Let's say you have a count of Vendor by Commodity inside a View, like this::



			Amount	cnt(Vendor Parent)
Information Te...	Tech Services	Tech Services	2,663,245	6
		Contract Program...	2,363,520	16
		Data Center Hosti...	1,356,977	4
		Cyber Security	150,411	2
	Telecom	Telecom	2,381,651	5
		Cable	1,564,355	2
		Telecom Equipment	655,896	2
		Cellular	477,774	3
	Hardware	Hardware	3,989,459	19
		Printers Copiers M...	574,604	3

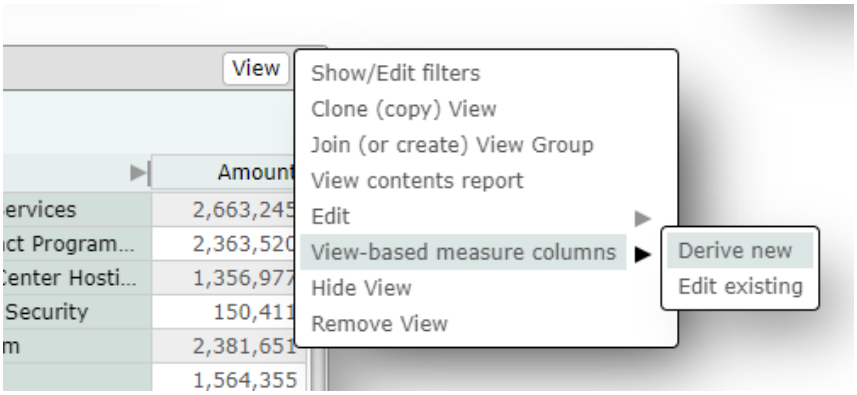
Suppose you now want to further segment Commodity by a range on those Vendor counts – in other words, Commodities with 2-3 Vendors, 3-6 Vendors, 6-10 Vendors, and so on. Like this:



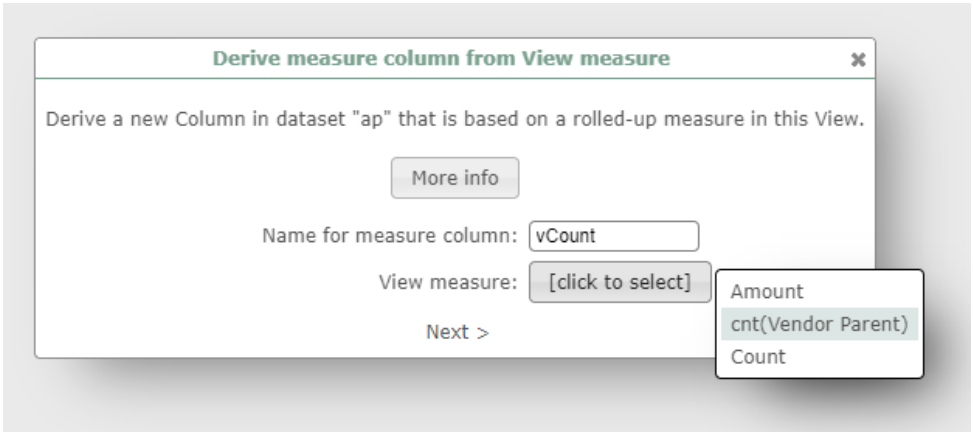
				Amount	avg(vCount)
2: 10 - 20	Facilities	Property	Leases & Rent	8,002,519	17
		Construction	General Contracto...	3,582,299	11
	Information Te...	Hardware	Hardware	3,989,459	19
		Software	Software	2,943,107	13
		Tech Services	Contract Program...	2,363,520	16
	Professional Se...	Legal Services	Legal Services	1,647,531	13
	Human Resour...	Temp Services	Temp Services	907,943	19
	Operations	Office Supplies	Office Supplies	324,312	15
3: 6 - 10	Information Te...	VAR	VAR	1,733,553	8
		Software	SAAS	1,074,826	7
	Marketing	Production Ser...	Production Services	2,625,484	7
	Operations	Office Supplies	Promotional Items	1,006,833	10
4: 3 - 6	Information Te...			6,401,873	5.5
	Marketing			4,520,995	5.22
	Operations			3,368,189	5.01
	Facilities			2,705,941	4.62

In the general case, you would have to manually connect the count of Vendor by Commodity back to your source data, for example by dumping it out and then reading it back in as a new dataset. That has the disadvantage that you would need to perform this manual step on every data refresh, or every time you wanted to modify the source View with amended counts (by filtering it, for example). Of course, with Spendata you could build a script-only Column that computes the result directly, and that would work for refresh. It wouldn't respond to real-time filtering, though, and it is difficult to construct.

That's where View-based measure columns come in. We derive the new column right from the View menu, like this:



We name the new measure and identify its source measure within the View:



And the measure is created. To build the final View above, we create a Range Column based on the new vCount measure and then crosstab Range by Commodity. Note that we use avg(vCount) inside the View, so that values will be meaningful for intermediate nodes and not summed for leaf nodes.

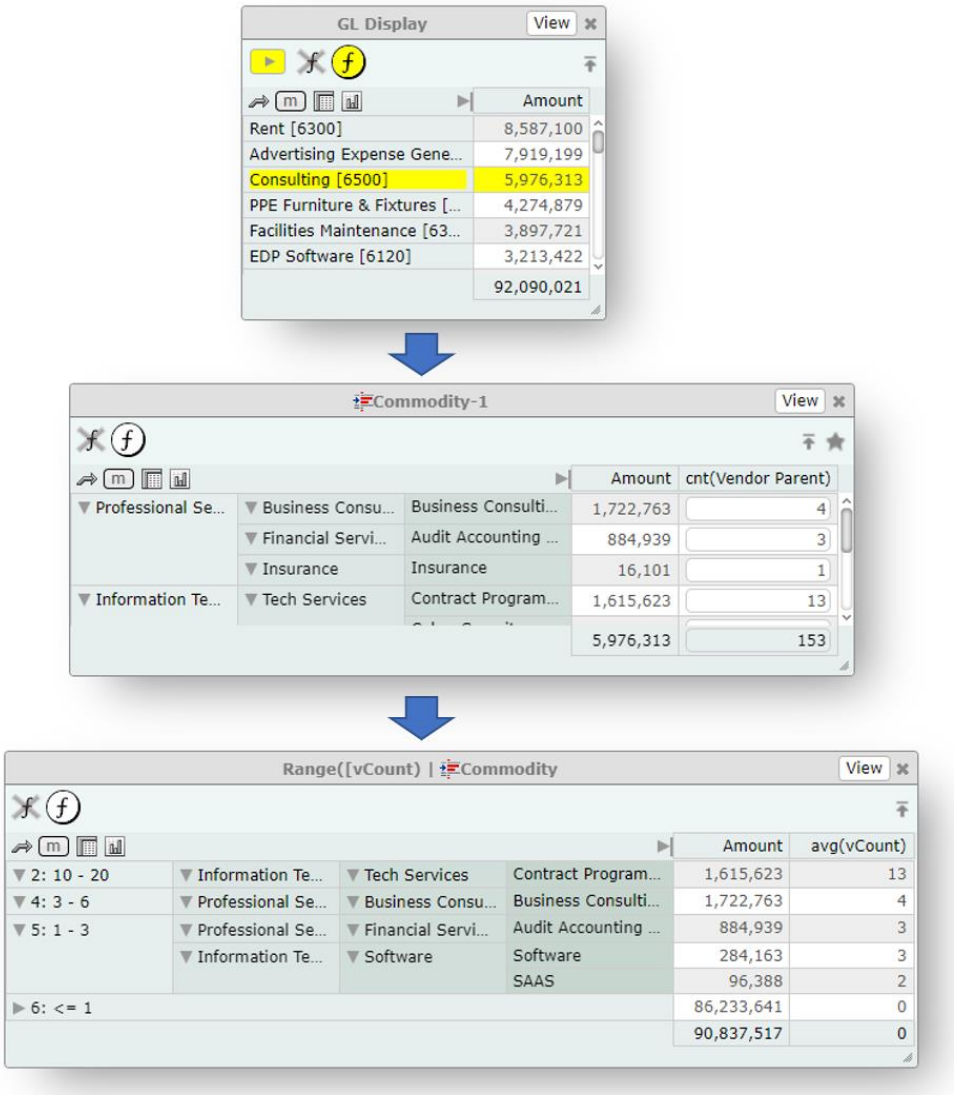
Dynamic Filtering

Now for the fun part – filtering the source View. If we were to filter the View in which the View-based measure column is defined, the following occurs:

1. The View is filtered.
2. The View-based measure column “vCount” is re-derived.
3. The Range column based on vCount is re-derived.
4. The crosstab of Range by Commodity is re-calculated.

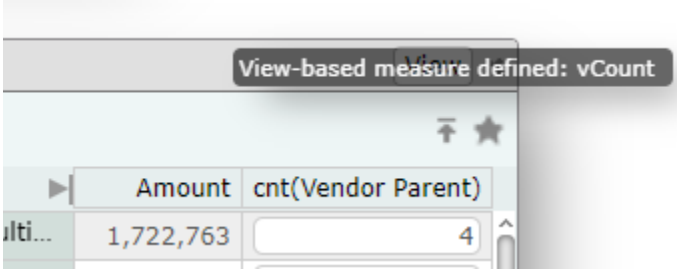
So now we have a real-time mechanism for driving models dependent on the View-based measure – an incredibly powerful capability.

For example, here we’ve filtered the source View by GL “Consulting”, illustrating the cascading effects:



View Annotations

When a View is the source of one or more View-based measure columns, it is annotated with a star. Mousing over the star provides a list of the names of the View-based measure columns defined by the View.



Use cases

- Benchmarking against averages. For example, suppose we have # of FTE per Cost Center. We can then do a comparison by cost center of their cost/FTE per commodity vs the overall average for this commodity. Ditto for travel (flight cost), office supplies, etc.
- Segment vendors into buckets based on the number of invoices (count of invoices by vendor), or the number of POs.
- Range of spend by vendor, by channel. Is this channel low cost to operate? Or is each invoice a separate piece of paper to process?
- Conveniently see if the spend in a particular View cell is the total spend for {vendor, commodity, etc.}.
- Create a segmentation of vendors that are 1) not mapped; 2) partially mapped; or 3) 95% mapped.

Notes

1. The new vCount measure is meaningful only in Views or filters where Commodity is involved. Using it without narrowing by Commodity means that you are pulling vCount values from multiple Commodities, which is meaningless.
2. The new vCount measure must be divided by <count of transactions> to be meaningful; otherwise it is summed by transaction. This can be done inside a View with the Advanced Measure “average” function, or anywhere (such as in a script-only Column) by explicit division by transaction count.