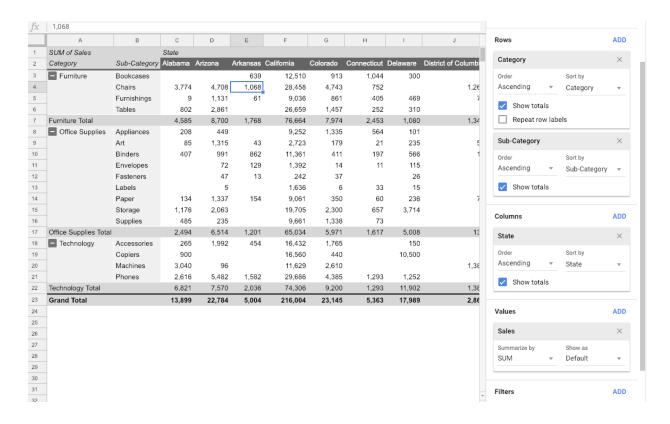# Crisp - Front End Take Home Test

Introduction

This problem documentation is not intending to be sufficient to deliver a "production product" - it is deliberately unspecific. In this project, you will implement a basic pivot table as a React component. Although your pivot table component should not be tied to a specific data set, we will provide you with a sample tabular data set in JSON describing product sales for a store. Take a look at an example of this data in a Google Sheet with a Pivot Table (feel free to make a copy of it to play around!). We also have a JSON representation for you.

When working with pivot tables, we generally speak of two types of fields:

1. Dimensions - fields that represent the rows and / or columns of the table (e.g Product Category, Customer)
2. Metrics - fields that are in the cells of the pivot table and can be aggregated (Sales, Quantity, Profit)

Below is a screenshot of the spreadsheet linked to above. In this case "Category" and "Sub-category" are row dimensions, "State" is a column dimension and "Sales" is the metric value.

The metric fields are aggregated to the level described by the row and column dimension. In the screenshot, the value *61* at E5 is the sum of all `sales` that matches the row predicate `category == 'Furniture' and subCategory == 'Furnishing' and state == 'Arkansas'`. The value *1,768* in E7 is the sum of all `sales` that matches the row predicate without the `subCategory` part - in essence the subtotal of the Sub-category rows in that column.

Formula bar: `1,068`

Category	Sub-Category	Alabama	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	District of Columb...
SUM of Sales		*State*							
⊟ Furniture	Bookcases			639	12,510	913	1,044	300	
	Chairs	3,774	4,708	1,068	28,458	4,743	752		1,26
	Furnishings	9	1,131	61	9,036	861	405	469	7
	Tables	802	2,861		26,659	1,457	252	310	
Furniture Total		4,585	8,700	1,768	76,664	7,974	2,453	1,080	1,34
⊟ Office Supplies	Appliances	208	449		9,252	1,335	564	101	
	Art	85	1,315	43	2,723	179	21	235	5
	Binders	407	991	862	11,361	411	197	566	1
	Envelopes		72	129	1,392	14	11	115	
	Fasteners		47	13	242	37		26	
	Labels		5		1,636	6	33	15	
	Paper	134	1,337	154	9,061	350	60	236	7
	Storage	1,176	2,063		19,705	2,300	657	3,714	
	Supplies	485	235		9,661	1,338	73		
Office Supplies Total		2,494	6,514	1,201	65,034	5,971	1,617	5,008	13
⊟ Technology	Accessories	265	1,992	454	16,432	1,765		150	
	Copiers	900			16,560	440		10,500	
	Machines	3,040	96		11,629	2,610			1,38
	Phones	2,616	5,482	1,582	29,686	4,385	1,293	1,252	
Technology Total		6,821	7,570	2,036	74,306	9,200	1,293	11,902	1,38
Grand Total		13,899	22,784	5,004	216,004	23,145	5,363	17,989	2,86

Rows ADD

Category ✕
Order: Ascending — Sort by: Category
☑ Show totals
☐ Repeat row labels

Sub-Category ✕
Order: Ascending — Sort by: Sub-Category
☑ Show totals

Columns ADD

State ✕
Order: Ascending — Sort by: State
☑ Show totals

Values ADD

Sales ✕
Summarize by: SUM — Show as: Default

Filters ADD

PRODUCTS — **STATES** ›

Category	Sub-Category	Alabama	Arizona	Arkansas	California	Colorado	Connecticut	Delaware	District of Columbia	Delaware
⊟ Furniture	Bookcases	0	0	12,510	12,510	913	1,044	300	300	300
	Chairs	3,774	4,708	1,068	28,458	4,743	752	0	1,200	0
	Furnishings	9	1,131	61	9,036	816	405	469	232	469
	Tables	802	2,861	0	26,659	1,457	252	310	300	310
Furniture total		4,585	8,700	1,768	76,664	7,974	2,453	1,080	2,032	1,080
⊟ Office supplies	Appliances	208	449	0	9,252	1,335	564	101	300	101
	Arts	85	1,315	43	2,723	179	21	235	1,200	235
	Binders	407	991	862	11,361	411	197	566	232	566
	Envelopes	0	72	129	1,392	14	11	115	300	115
	Fasteners	0	47	13	242	37	0	26	47	26
	Labels	0	5	0	1,636	6	33	15	5	15
	Paper	134	1,337	154	9,061	350	60	236	1,337	236
	Storage	1,176	2,063	0	19,705	2,300	657	3,714	2,063	3,714
	Supplies	485	235	0	9,661	1,338	63	0	235	0
Office supplies total		2,494	6,514	1,201	65,034	5,971	1,617	5,008	5,719	5,008
⊟ Technology	Accessories	265	1,992	454	16,432	1,765	0	150	300	150
	Copiers	900	0	0	16,560	440	0	10,500	1,200	10,500
	Machines	3,040	96	0	11,629	2,610	0	0	232	0
	Phones	2,616	5,482	1,582	29,686	4,385	1,293	1,252	300	1,252

Requirements

- Built "from scratch", no use of existing UI component libraries (but feel free to use the other supporting libraries that "come with" create-react-app)
- Dimensions should be configurable in code and multiple dimensions should be supported, at least on the row dimensions
- Only one metric is supported, we only support SUM aggregations and only number types need to be supported
- Hooks for loading the data from an API, rather than assuming that the entire data set is loaded into the browser

Non-functional requirements

- Please make your submission *directionally* similar to the design supplied in the above screenshot
- In order for us to properly assess the submission, please use Typescript or vanilla ESx.

Out of scope

- No need for UI for configuration (like the Google Sheets Pivot Table Editor) is not required
- No need for additional, non-specified "bells and whistles" (i.e., dark mode) – just focus on the core functionality
- No need for API for serving the data, just serve it statically or embed it into your app for now

Deliverables

- Running code and test suite provided through online code repo or in a tar-ball
- Instructions on how to build and run
- *Short* architectural overview
- List of assumptions or simplifications made
- List of the next steps you would want to do if this were a real project

Hints to what we're looking for

- Try balancing some common sense foresight with the simplest thing that could work.
- Pick a reasonable approach and be prepared to speak to alternatives in the review.

How much time should I spend?

Short answer:

Spend six hours and turn it in. If your mindset won't let you do that, spend an extra six hours, and *then* turn it in.

Longer answer:

Here at Crisp, we're trying to do what makes sense, instead of what's always been done before. The traditional interview process for a developer usually includes a series of phone screens, an on-line coding test, and culminates with an all-day, on-site interview. The process is costly, intrusive, and fails to replicate what developers actually do on a day-to-day basis.

We're trying to streamline the process, and get at what actually matters: how likely are you to be successful, and thrive, if we welcome you onto our team? Towards that end we've replaced the on-line coding tests, and the on-site interviews with a realistic project that represents something you might actually build at work.

As with any project, you will be presented with the traditional tradeoff between functionality, quality, and cost. We're not looking for perfection, or coverage of every imaginable use case. And we're not looking for you to invest more time in our interview process than you would in a more traditional, on-site interview. Senior developers, using their preferred tools, should be able to deliver a quality implementation of our take home project, with significant levels of functionality, in about a day.

If that seems insurmountable to you, it does not automatically mean that you are not qualified to work with us. Please, feel free to narrow the implementation scope, or spend a bit more time iterating on your solution - whichever you are more comfortable with. We will ask *you* to set a deadline that you can meet while working at your own pace. It's okay to set that deadline several weeks out, if that's what you will need due to work, life, or family circumstances.

Please do *not* set an extended deadline, intending to invest extraordinary time in this project. We've had candidates be successful having spent as little as four hours, and we've had candidates spend over thirty hours and not advance to the final part of the interview process. We want your time commitment to reflect the investment you would put into any interview process, and not be an undue burden.

Good luck, and have fun!

Submission Format: Please zip up or provide a Google Drive link to the code and documentation by uploading it to Rippling