AppSheet “Build from Scratch” Demo Script

Read time: reading through this document and practicing this demo should take ~30-60 minutes, and less time the 2nd or Nth time you run through it. We recommend practicing at least 3 times.

Intro

This is a hello world AppSheet demo meant to be conducted by a sales rep or other non-technical person. This demo is great for discovery calls that may also have a slightly technical focus or technical audience. This demo is not recommended for Directors, VP’s or decision makers unless they specifically ask for a demo like this (which can happen from time to time). Although it takes a bit of practice, there’s nothing *overly technical* about this demo - please use your best judgement.

You will know this is the right demo if the prospect asks you: “Can I get a quick 5 minute demo?”, “Can you connect to my data and build an app in 5 minutes?” or “I have (for example) 18 specific questions about specific functions in your platform” or similar.

We also have two other demo flows: an “analyze an existing app” demo using this standard app which is a bit more thorough, as well as a “full blown multiple personas” demo called *Bricklin*

Holdings which is even more thorough. This demo, however, is a great place to start if you are in a discovery phase...

Value statements are shown in green below. You should memorize these value statements. Value statements are required for selling. Selling is required for generating revenue. Revenue is required for continued selling. Ergo, *selling requires value statements*.

Steps

At first glance this may seem like a lot of steps, but after you run through them a few times you will realize that the below steps are filled with many **value statements** and other filler/background/noise. In practice, this demo is a very quick process.

First things first! ...make a copy of the google sheet referenced below, so that you have your own working copy!

- Open up your copy of companion google sheet "Build from Scratch" and show it to the prospect. Take note that there are two tabs, *accounts* and *activities*. This is the beginning of a parent child relationship.
- Ask the prospect if they have any typical parent-child relationships. We bet they do! Houses have rooms, doctors have patients, Facilities have safety inspections, cleaners have clients, and on and on and on. You can hint, of course, that AppSheet is not limited to 1:1 or just one of these things. There can be many.

AppSheet lets anyone design and manage their data using tools they are already comfortable with.

- Ask questions about your prospect. You may want to change the column headers (by industry terminology) as you get more comfortable with this demo, but for now at first, don't change the header names.

- Ask questions about the prospect's data sources: which data sources, how much data, how many years of it.
    - Bonus round: open a new tab, go to your AppSheet account, and show all the various types of data sources that we support connecting to. When done, close that out and come back to this demo.
- Leave the google sheet open - you will come back to it later!
- Now open appsheet.com, create a new app, and connect to this google sheet. Appendix A has screenshots and steps for this.

AppSheet lets you connect to your data where it resides, without the need for coding or technical work. Designers can connect to multiple types of data all from within one app.

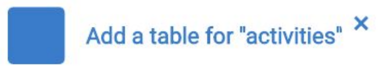
- Once the app loads, give a brief overview of the UX (user experience/interface):
    - Web based designer
    - Built-in emulator (right hand side)
    - Various pages for data, UX, behavior, management, users/permissions, etc.
- If you land on a map view, don't worry about it! AppSheet detected a field called *location* and decided to give you a map. No big deal... in fact it's great.
- In the emulator, click the plus sign:
- Enter some data and then click save. It doesn't matter what you type in at this point.
- After the badge icon (for data sync) goes away:
- ...go back to your google sheet and show them the data that was just entered.

AppSheet lets you read and write data to a multitude of data sources in real time with no coding.

- In google sheets, *delete the row that was created,* then go back to AppSheet. Then click the sync icon. The record will disappear.

Data is a two way street - if you have other systems operating and updating on your data, AppSheet can see those changes in real time.

- In AppSheet go to the *data* view and then the *tables* tab, and notice that AppSheet is asking if you would like to connect to the second google sheet tab called *activities*:



- Go ahead and click this. The app will save and reload automatically.

Users worldwide have created millions of apps using the AppSheet platform. As a result, our built-in intelligence and predictive features can predict: 1. next steps, 2. field types based upon name, 3. views, actions and workflows based upon data, and 4. much more. This is a huge time saver for an AppSheet designer.

- Now you have two data sources in AppSheet connected to two tabs in a google sheet.
- Next we will connect them together and create a parent child relationship.
- Go to the columns view for your new *activities* data.
- Edit the field called AccountKey and create the reference to the accounts data. Detailed screenshots and steps are in Appendix B.

With zero coding, we have created a relationship between two types of data, a task typically handled by a software developer or database administrator (DBA). This unlocks the ability to create data relationships and app designs for regular non-developer folks.

- Since you are looking at the columns for *activities*, this is a good time to point out other predictive features and decisions that AppSheet performed automatically:
  - We discovered the field *UniqueID* and decided to make it of type *Key*. We also placed a calculation into its *initial value*. (scroll rightward to see the *initial values* area)
  - We discovered *Timestamp* and made it a *DateTime*, and added *Now()* to its *initial value*.
  - We decided that field *NumberValue* should be of type *Price*.
  - We decided that *LatLong* should be of type *LatLong*
  - Bonus round: add this function to the *initial value* for the *LatLong* field: *Here()*

All of the above "functional discoveries" are part of AppSheet's built-in data science and intelligence which saves your team a ton of time.

There is a short list of extra-credit cleanup items in Appendix C which you are welcome to perform at this point in the demo, or not, as you see fit.

- On the left hand side of the screen, click on UX:

  (UX is synonymous with "View Type" or "User Interface", it stands for "User Experience". Now you know...)
- Delete any map or calendar views, we don't need them... however, while you do that:
  - Point out all of the built-in View Types
  - What kind of View Type is it? (map, calendar, list, chart, dashboard, etc)
  - What data is it connecting to? (in this case, accounts or activities)
  - Where in your app do you want to place this View Type? (bottom panel, menu, hidden/ref)

By combining data, UX design elements, and positioning into one web-based, no-code solution, we allow anyone to create powerful navigation and data systems in the AppSheet platform. This is a huge unlock for regular business-savvy people!

- In the emulator you should see two remaining views, *accounts* and *activities*. If not, create two *deck* views for *accounts* and *activities*. Leave all defaults as is.
- Create a new account record - make sure you fill in all the fields (just so it looks decent). You should be able to see the data in google sheets as a result.
- Create a new activity record. This time notice you can refer to the account record you just created. This is proof of the parent-child relationship.
- At this point you take one of the following steps:
  - A bunch of data entry using the emulator, e.g.:
    - Create some *accounts* data, then some *activities* data for that *account*. And maybe even play around with the UX views. See Appendix D.
    - Deploy this app and then immediately share it with your prospect. See Appendix E - *however please note that Appendix E and this part of the demo is more advanced and fraught with potential issues (i.e. getting your*

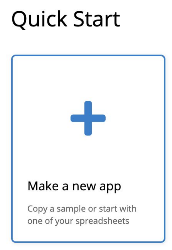
prospect to check their email; then getting them to log into AppSheet; and then getting them to navigate and use the app you just built .... These all have inherent associated risks)

Wrap up, ask more questions, pivot to other discussion points, show another pre-built app if there is interest. Finished.

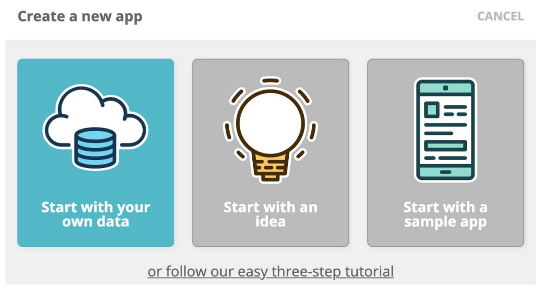
Appendix A: Connecting to a google sheet

Screenshots

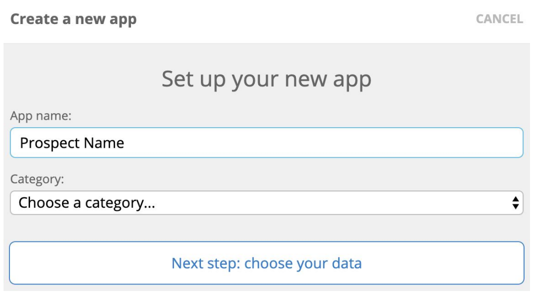
1. Click to make a new app:



2. Choose “start with your own data”:



3. Give it a name and category:


Create a new app

CANCEL

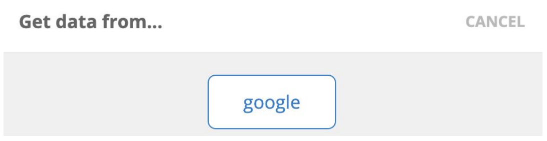
Set up your new app

App name: Prospect Name

Category: Choose a category...

Next step: choose your data

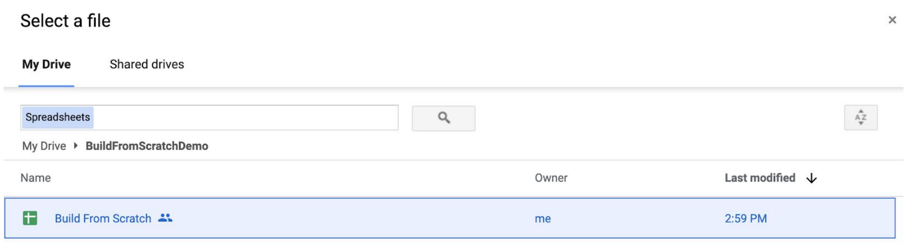
4. Choose Google as your data source:


Get data from...

CANCEL

google

5. Find and select the “Build from Scratch” google sheet:


Select a file

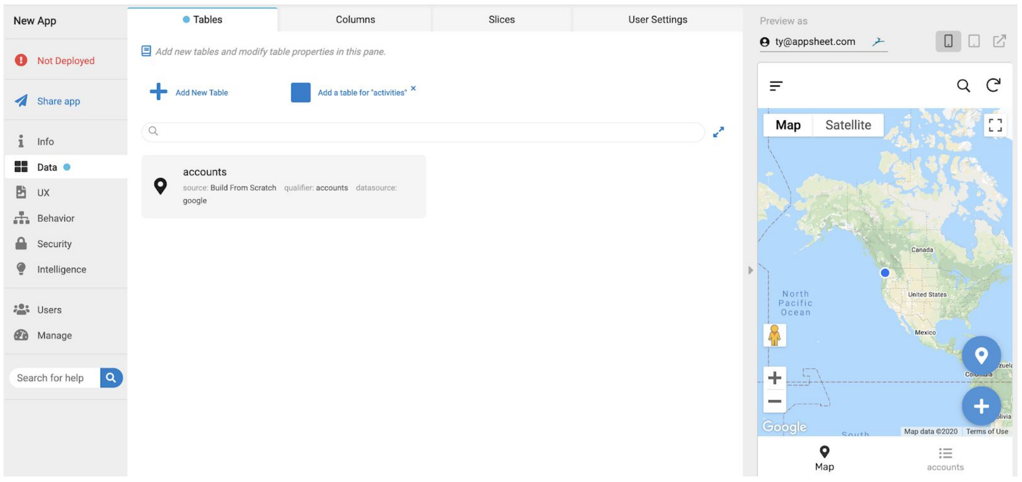
My Drive | Shared drives

Spreadsheets

My Drive ▸ BuildFromScratchDemo

Name	Owner	Last modified
Build From Scratch	me	2:59 PM

6. Wait while AppSheet sets up your new app! When it loads, it should look something like this:


New App

Not Deployed

Share app

Info

Data

UX

Behavior

Security

Intelligence

Users

Manage

Search for help

Tables | Columns | Slices | User Settings

Add new tables and modify table properties in this pane.

Add New Table

Add a table for "activities"

accounts
source: Build From Scratch qualifier: accounts datasource: google

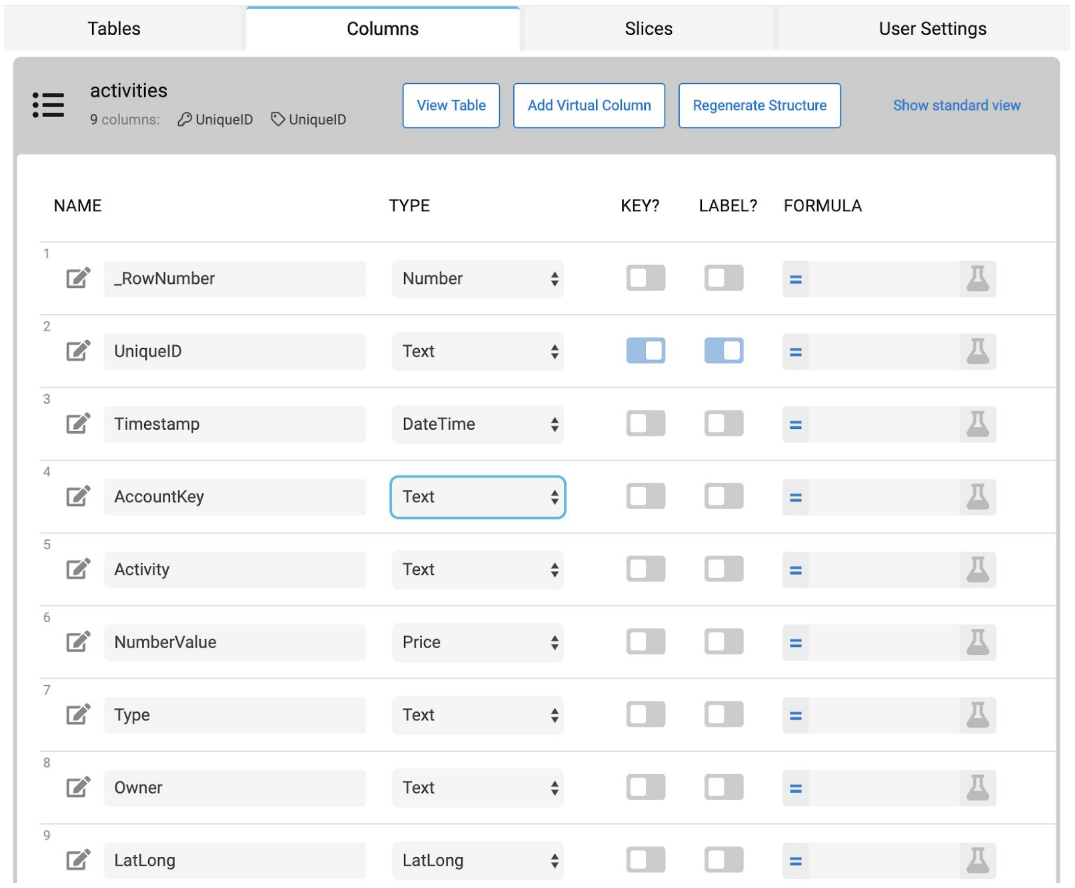
Preview as
ty@appsheet.com

Finished. Go back to the overview steps.

Appendix B: Creating the parent child relationship

Screenshots

1. Click the pencil icon next to the field *AccountKey:*

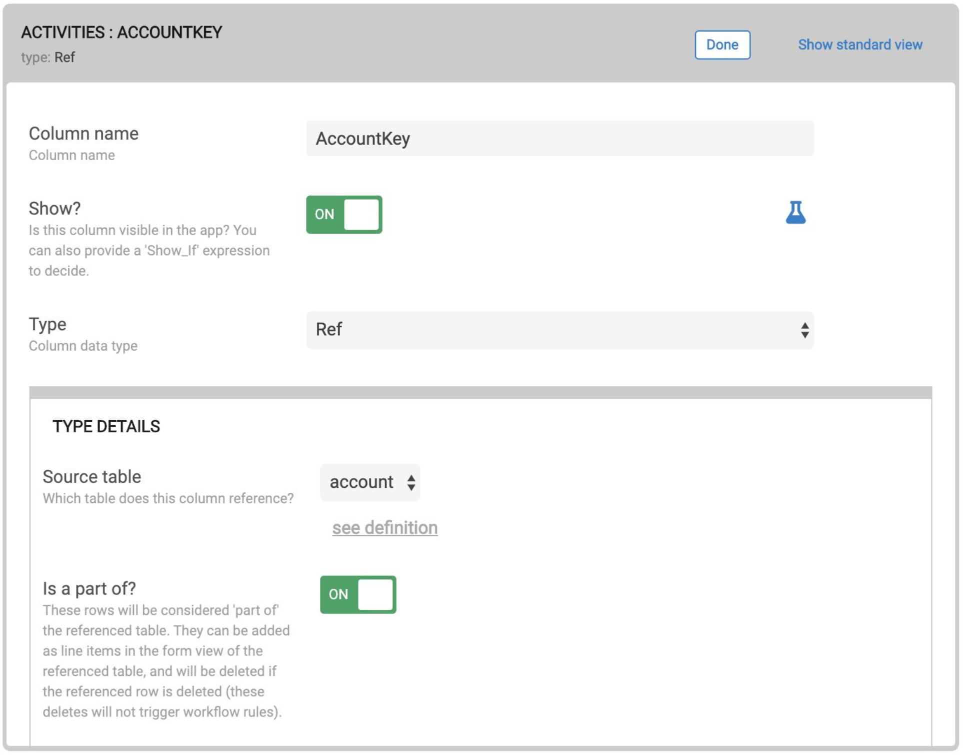


	NAME	TYPE	KEY?	LABEL?	FORMULA
1	_RowNumber	Number	☐	☐	=
2	UniqueID	Text	✓	✓	=
3	Timestamp	DateTime	☐	☐	=
4	AccountKey	Text	☐	☐	=
5	Activity	Text	☐	☐	=
6	NumberValue	Price	☐	☐	=
7	Type	Text	☐	☐	=
8	Owner	Text	☐	☐	=
9	LatLong	LatLong	☐	☐	=

2. Change the type to *Ref*, then

3. Select the *accounts* table, then

4. Enable *is part of*. It should look like this:


ACTIVITIES : ACCOUNTKEY
type: Ref

Column name
Column name
AccountKey

Show?
Is this column visible in the app? You can also provide a 'Show_If' expression to decide.
ON

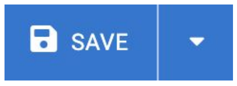
Type
Column data type
Ref

TYPE DETAILS

Source table
Which table does this column reference?
account

Is a part of?
These rows will be considered 'part of' the referenced table. They can be added as line items in the form view of the referenced table, and will be deleted if the referenced row is deleted (these deletes will not trigger workflow rules).
ON

5. Click Done. Then Click Save in the upper right:

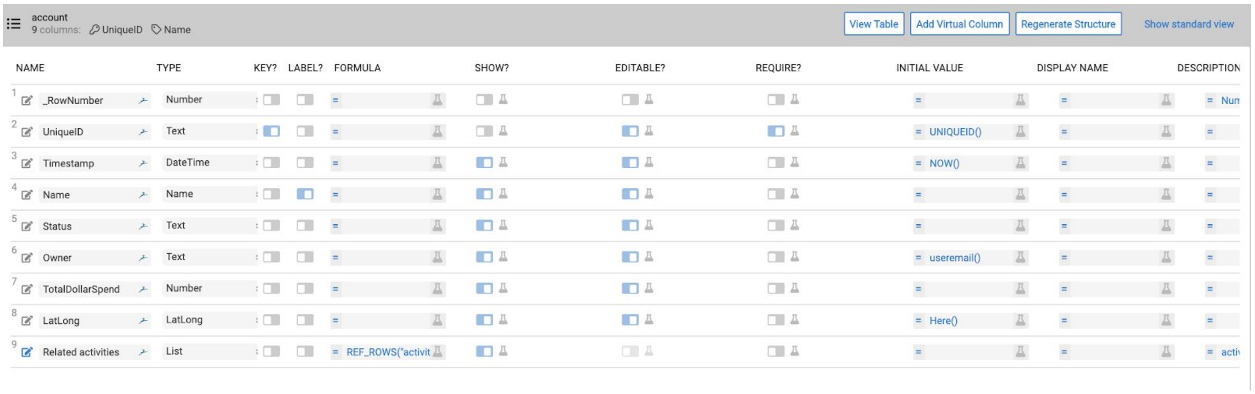


Finished. Go back to the overview steps.

Appendix C: Clean up the various field settings

The following are zen and OCD-oriented extra steps you can take to clean up what the fields look like in this demo.

1. Make the *accounts* data columns look like the following:

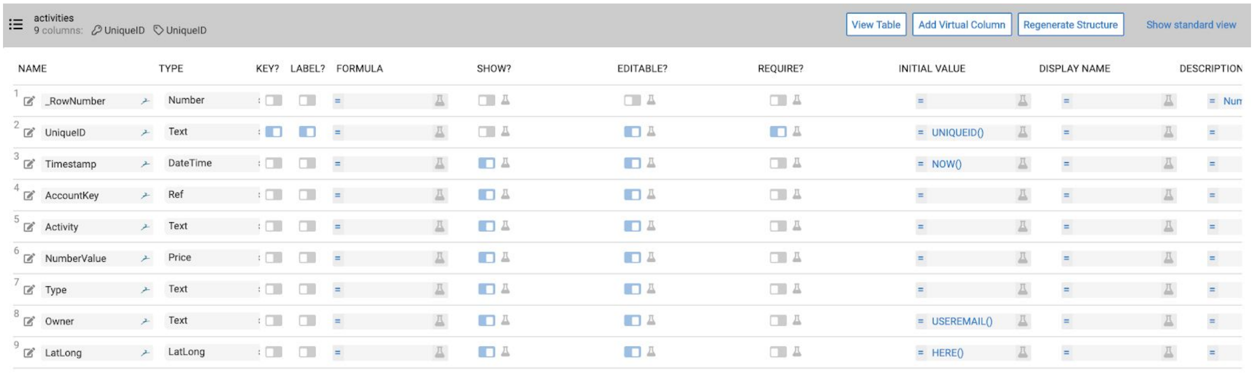


account
9 columns: UniqueID Name

	NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?	EDITABLE?	REQUIRE?	INITIAL VALUE	DISPLAY NAME	DESCRIPTION
1	_RowNumber	Number									Nun
2	UniqueID	Text	✓				✓	✓	UNIQUEID()		
3	Timestamp	DateTime				✓	✓		NOW()		
4	Name	Name		✓		✓	✓				
5	Status	Text				✓	✓				
6	Owner	Text				✓	✓		useremail()		
7	TotalDollarSpend	Number				✓	✓				
8	LatLong	LatLong				✓	✓		Here()		
9	Related activities	List			REF_ROWS("activit	✓					acti

2. Notice the calcs in the *initial value* column. Also, make sure you turn off *Show* for the *UniqueID* field. We don't need our end users seeing these key fields.

3. Same basic premise for *activities*: make the *activities* data columns look like the following:

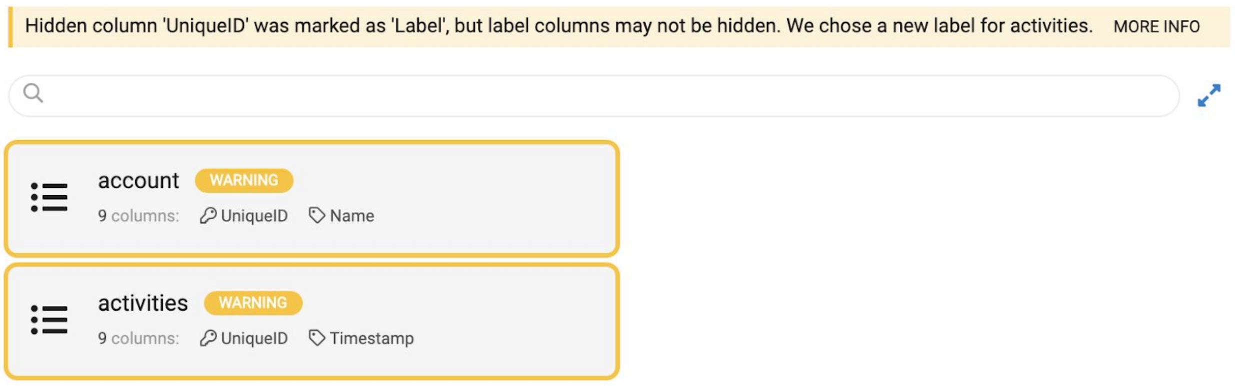


activities
9 columns: UniqueID UniqueID

	NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?	EDITABLE?	REQUIRE?	INITIAL VALUE	DISPLAY NAME	DESCRIPTION
1	_RowNumber	Number									Nun
2	UniqueID	Text	✓	✓			✓	✓	UNIQUEID()		
3	Timestamp	DateTime				✓	✓		NOW()		
4	AccountKey	Ref				✓	✓				
5	Activity	Text				✓	✓				
6	NumberValue	Price				✓	✓				
7	Type	Text				✓	✓				
8	Owner	Text				✓	✓		USEREMAIL()		
9	LatLong	LatLong				✓	✓		HERE()		

4. Make sure you turn off *Show* for the *UniqueID* field. We don't need our end users seeing these key fields.

5. You will get warnings after you click SAVE:


Hidden column 'UniqueID' was marked as 'Label', but label columns may not be hidden. We chose a new label for activities. MORE INFO

account WARNING
9 columns: UniqueID Name

activities WARNING
9 columns: UniqueID Timestamp

This is totally normal. In fact, it's a great feature: the AppSheet platform is providing visual feedback to the app designer about changes they have made.

Finished. Go back to the overview steps.

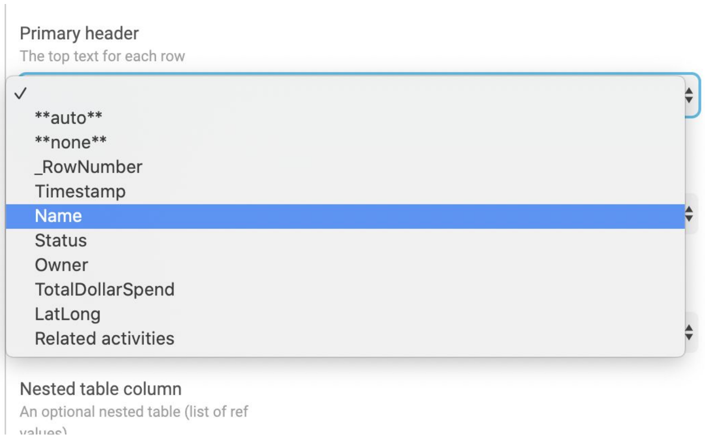
Appendix D: Creating, editing and deleting data

Some interesting tidbits here:

- LatLong. This gives you a map widget and lets you drag the pin around on the map. Because the initial value is set to the Here() function, the app will do its best job in trying to "find you". In a web browser this is realistically your nearest router or ISP location.
- "Owner" field should pick up the current users' email.
- The rest of the fields are pretty obvious
- Bonus round: for the field status on data accounts, change it to type Enum and put in three values. Make it a button type.

If you have created some accounts and activities data, you could change the UX “deck view” as follows:

1. Expand the *accounts* deck view so that you can see its details.
2. Change *primary header* to field *name*:

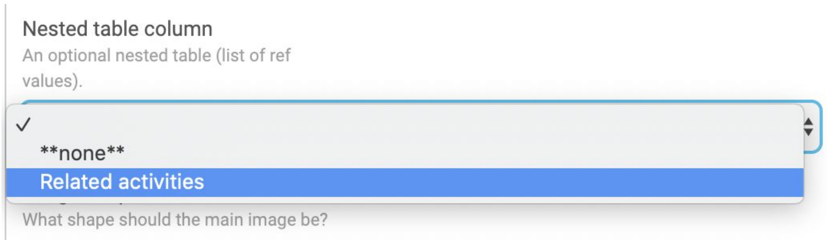


Primary header
The top text for each row

✓
auto
none
_RowNumber
Timestamp
Name
Status
Owner
TotalDollarSpend
LatLong
Related activities

Nested table column
An optional nested table (list of ref values)

3. Change *secondary header* to field *owner*
4. Change *nested table column* to *Related activities*:



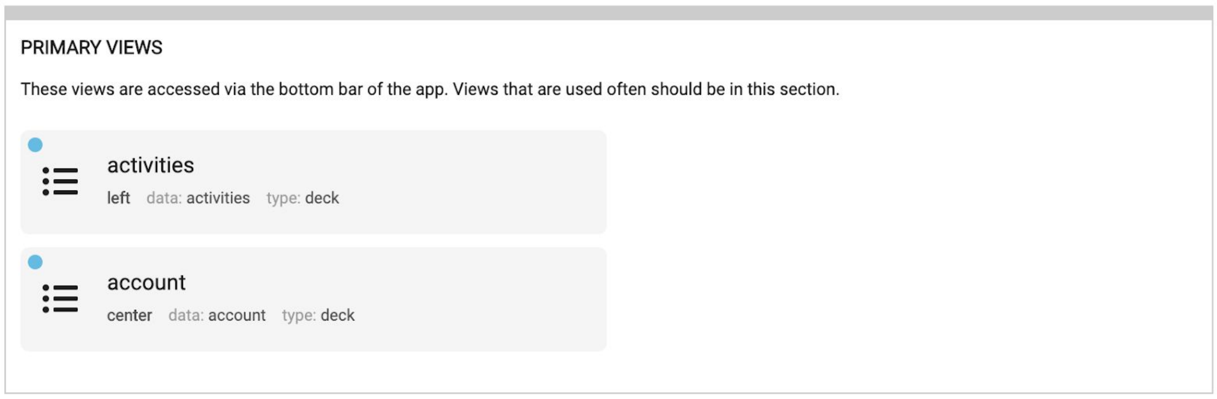
Nested table column
An optional nested table (list of ref values).

✓
none
Related activities

What shape should the main image be?

5. Save your changes and see what the emulator looks like now.

6. Bonus round: figure out how to edit the *Details* UX elements. You may need to turn on *show system views*:


PRIMARY VIEWS

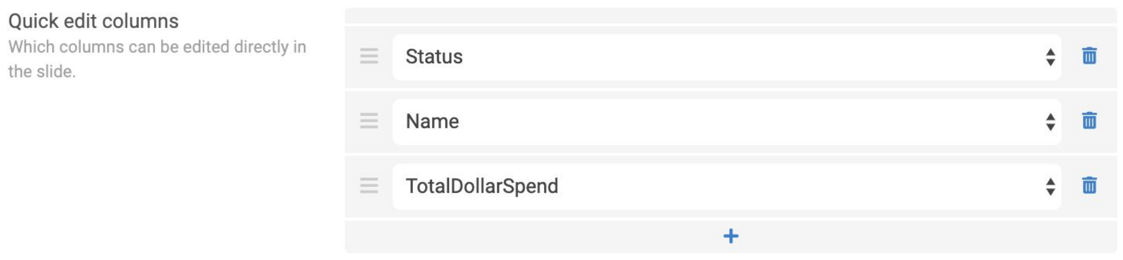
These views are accessed via the bottom bar of the app. Views that are used often should be in this section.

activities
left data: activities type: deck

account
center data: account type: deck

Show system views

7. Then for each of the two *detail* views (accounts and activities), enable *Quick edit columns* and add some columns for editing, e.g:


Quick edit columns
Which columns can be edited directly in the slide.

Status
Name
TotalDollarSpend

8. Save your changes and experiment with what this looks like.

Finished. [Go back to the overview steps](#).

Appendix E: Deploy the app and share with your audience

A rough overview of steps is shown below. We are deliberately not handing this to you on a silver platter as we need to ensure that you can be self-sufficient for this portion of the experience. Practice, practice, practice!

1. This is all from within the app designer.

2. Add the prospects' email as a coauthor on this new hello world app. Send invite email.
3. For "auth", make sure this app is set to "any provider".
4. Be prepared to troubleshoot your prospect logging into appsheet. 99% of the time this is not a big deal... until all of a sudden it's a rathole...
5. The ideal end state is that they can see and edit the app you just created. They are a co-author!
6. *Bonus round: show them how to "copy" the entire app, including making a copy of the data.*

Good luck! You got this...