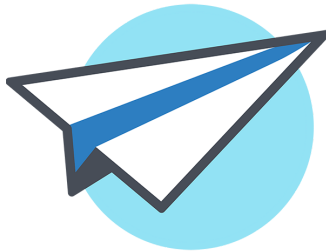


# AppSheet Product Training Course

Prepared for AppCamp







Version 1.1

<b>Guide to Symbols</b>	<b>3</b>
<b>Introduction</b>	<b>4</b>
1.1 Organization	4
1.2 Prerequisites	4
<b>Data</b>	<b>4</b>
2.1 Create a spreadsheet to hold your data	5
<b>2.2 Decide structure for job sites</b>	<b>5</b>
2.3 Create an app and connect your data	6
2.4 Add remaining tables	8
2.5 Update column settings for Reports table	9
2.5.1 Update Report ID	11
2.5.2 Create a reference between Reports and Job Sites	12
2.5.3 Create a reference between Reports and Users	13
2.5.4 Create a label	14
2.5.5 Update Review Timestamp	15
2.5.6 Test the Form	16
2.5.7 Set “Display name”	18
2.5.8 Create option buttons	18
2.5.9 Yes/No data type	19
2.5.10 Show and Require	19
2.6 Update column settings for Users table	20
2.7 Update column settings for Job Sites table	22
2.7.1 Create a slice for Site Leads	23
2.7.2 Update the “Site Lead Email” column	23
2.8 Update column settings for Maintenance Tickets table	23
2.9 Add reference between Reports and Maintenance Tickets	25
2.10 Wrap Up	26
<b>Checkpoint #1</b>	<b>28</b>
<b>3. Views</b>	<b>29</b>
3.1 Create a view to manage user roles	29
3.2 Create form view to report events	31
3.3 Create table view to see reports for logged in user	32
3.4 Add a security filter to Reports table	32
3.5 Create form view for Maintenance Tickets	33
3.6 Create deck view for Maintenance Tickets	33
3.7 Update the Job Sites map view and list view	34
3.8 Clean up unused views	35
3.9 Create a dashboard for the Site Lead	35
3.9.1 Create view to show new reports	36

3.9.2 Create a detail view	37
3.9.3 Create a dashboard	38
3.10 Create a dashboard for the Safety Board	38
3.10.1 Create a table of reviewed reports	39
3.10.2 Create a bar chart	40
3.10.3 Create a breakdown of events by priority	41
3.10.4 Create a breakdown of events by category	42
3.10.5 Create a dashboard	43
3.11 Set opening view	44
3.12 Wrap Up	45
<b>Checkpoint #2</b>	<b>46</b>
<b>4. Behaviors</b>	<b>47</b>
4.1 Create actions for the Site Lead	47
4.1.1 Set “Needs Review” to False	47
4.1.2 Set “Review Timestamp” to now	48
4.1.3 Create a grouped action	48
4.1.4 Testing	49
4.1.5 Remove the action bar from the Needs Review panel	50
4.1.6 Add a confirmation	50
4.2 Create an automated email	50
4.2.1 Create a Workflow	51
4.2.2 Setup the Email Content	51
4.2.3 Create an email body template	52
4.2.4 Create an email attachment template	52
4.2.5 Troubleshooting Workflows	53
4.3 Enable offline access	53
4.4 Sync settings	54
4.5 Wrap up	54
<b>5. Switching roles</b>	<b>55</b>
<b>Checkpoint #3</b>	<b>56</b>
<b>6. Customizing User Experience</b>	<b>57</b>
6.1 Format rules	57
6.1.1 Critical red	57
6.1.2 Normal yellow	57
6.1.3 Low green	57
<b>6.2 Branding</b>	<b>58</b>
6.3 Localization	58
<b>7. User Management</b>	<b>59</b>
7.1 Adding users	59

7.2 Domain Authentication	60
7.3 Version control	60
7.5 Transferring an App	61
7.6 Create a public sample app	61
<b>8. Scalability</b>	<b>61</b>
8.1 Security Filters	61
8.2 User Settings	62
8.3 Partitioning	63
<b>Checkpoint #4</b>	<b>64</b>
<b>9. Resources</b>	<b>65</b>
<b>Revision Record</b>	<b>65</b>

## Guide to Symbols

	Big Picture	These sections describe the ‘why’ behind the ‘what’. They are intended to describe the objective the app creator is trying to reach. The detailed process then serves as an example of how to accomplish that objective.
	Best Practice	There are many ways to get to the same result. Best practices call out techniques that should be adhered to. They generally are related to performance and scalability.
	Pro Tip	Helpful tips and tricks.
	Breeze By	Some sections repeat skills already covered in previous sections. These sections are kept for completeness. If you feel comfortable with the related skills, feel free to breeze by these sections.

# 1. Introduction

Welcome to the AppSheet product training course! This is an 8-hour course that will guide you step by step through building a process management app in the AppSheet platform. During the course, you will gain experience with many fundamental skills used in the platform. At the end of the course, you will have built an intermediate-level app and gained the AppSheet knowledge needed to create a wide range of useful apps to enhance your business.

## 1.1 Organization

Each section of this course will walk you through a stage of app creation. There will be checkpoints along the way where a sample app is provided. At each checkpoint, it is suggested to copy the sample app to your account and use it as a starting point for subsequent sections. This will ensure the description in the text aligns to the app you are working on.

## 1.2 Prerequisites

This course is aimed at helping beginning app creators become intermediate app creators. It assumes a basic familiarity with the layout of the AppSheet editor. Recommended activities before beginning the course are as follows:

- Complete the free [AppSheet Academy](#) course on Udemy
- Build at least one simple app to be familiar with the organization of the AppSheet editor
- View the [AppSheet Resources Portal](#) and view [Praveen's Demo](#)

# 2. Data



Data organization influences the usability and scalability of the app. Our goal for this app is to capture safety reporting information. There are two types of safety reports: incidents and near-misses. An incident is an event where something bad happened. A near-miss is one where conditions existed for something bad to happen but no damage was realized. For both these events, we need to collect the same information and process it in approximately the same way. The commonality indicates both event types should be stored in a single table with a column to differentiate which type.

We also want to create and track maintenance tickets related to the events. The maintenance tickets are related to the safety events, but have different information. This indicates they should be stored in a separate table and have a reference to the reports table.

We also are tracking the job site where each event occurs. We have to choose whether the job site information for a report should be stored in the reports table or as a separate table with a reference. Excogitate on that and we'll revisit the idea later in the course.

We know that the experience a user of the app has will depend on the user's role. To make this customization, we need to create a location to connect the user's email to their role. This suggests a users table is required.

Before opening AppSheet, we can create the data structure to hold this information. At this stage, it looks like we need tables for Reports, Maintenance Tickets, and Users. Maybe another table for Job Sites.

## 2.1 Create a spreadsheet to hold your data

Start by creating a spreadsheet. Tables are often stored as separate tabs within a single spreadsheet file. It is also possible to store tables as individual files. Using tabs makes it a little easier to switch between tables in the spreadsheet. Using files allows you to store more data before reaching the limitation of the data provider. We'll use tabs in this course.

### Steps

1. [Open this sample](#)
2. Go to File > Make a Copy
3. Save a copy of the file in your Drive

## 2.2 Decide structure for job sites



Try to avoid saving duplicate data. One of the powers of AppSheet is the ability to create references between tables. If you see a lot of duplicate information, consider moving the duplicate data into its own table and creating a reference.

Think about what information is needed for the job sites.

- Name
- Address
- Associated site lead

If we had only one column of information, say the job site name, then storing it in a separate table would not actually reduce the amount of data in the app. With the list above, using a separate table will be more efficient and will also set up the app to easily scale if more job site specific information needs to be added in the future.

Additionally, it allows us to specify the site lead for each job site. If we need to change the site lead, we would only need to do that in the relevant row of the Job Sites table rather than doing a risky find-and-replace-all.

The outcome: Job Sites will be a separate table.

### Steps

1. Add a new tab to the spreadsheet and name it "Job Sites"
2. Add the following Column headers:
  - a. Job Site ID

- b. Name
- c. Address
- d. Created On
- e. Site Lead Email

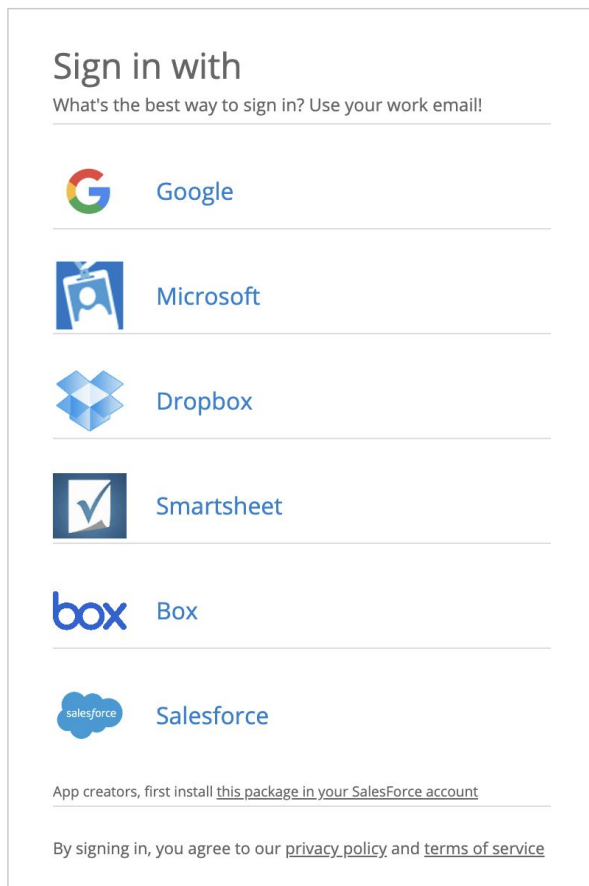
3. Copy/paste in the example data below

Job Site ID	Name	Address	Created On	Site Lead Email
JobSite_1	Pine Street Apartments	715 E Pine St, Seattle, WA 98122	7/30/2019 9:24:24	lead@example.com
JobSite_2	Madison Office Complex	2201 E Madison St, Seattle, WA 98112	7/30/2019 10:24:24	lead@example.com

## 2.3 Create an app and connect your data

At the AppSheet login, you are prompted to choose a service.

Screenshot of login options:



The service you login with is the service AppSheet will use to store files associated with your apps.

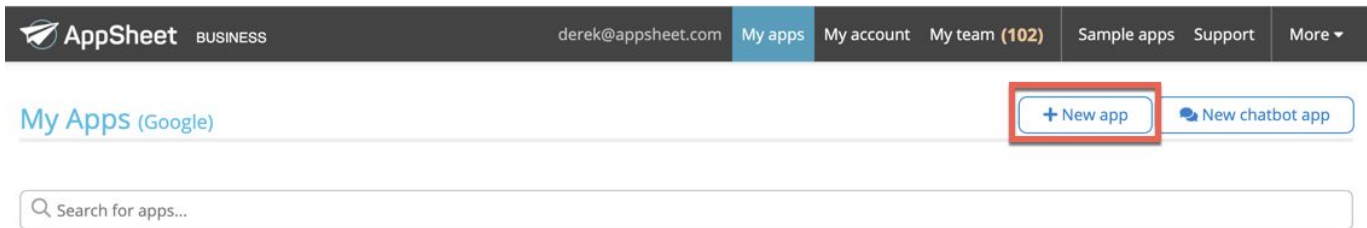


**Pro Tip:** If you authenticate through two different services, AppSheet will create two separate AppSheet accounts. This is true even if the email address you use for both services is the same. This means that Apps you create after logging in through one service will not be visible if you login through a different service. To keep all your apps in one account, it is easiest to choose one service to use for login. After logging in, you have the option to pull data from other sources.



Steps:

1. Login to [AppSheet](#)
2. Go to “My Apps” and click “+ New app”



3. Click “Start with your own data”
4. Give your app a name and a category
5. Choose where your data is stored
  - a. If your data is stored in a data source that is different from the one you used to login, then you will need to add the data source. You will need to provide AppSheet with credentials to access the location where the data is stored
  - b. Otherwise, select the data source you used for authentication
6. Navigate to the location where you saved the spreadsheet in Section 2.1.
7. Select the spreadsheet

AppSheet will automatically create the app and load the first table. Additionally, AppSheet will automatically create views for the first table. AppSheet has seen hundreds of thousands of apps. Based on this data set, AppSheet applies a machine learning algorithm to make intelligent assumptions about what the app creator wants to see based on the data.

## 2.4 Add remaining tables

Skills:

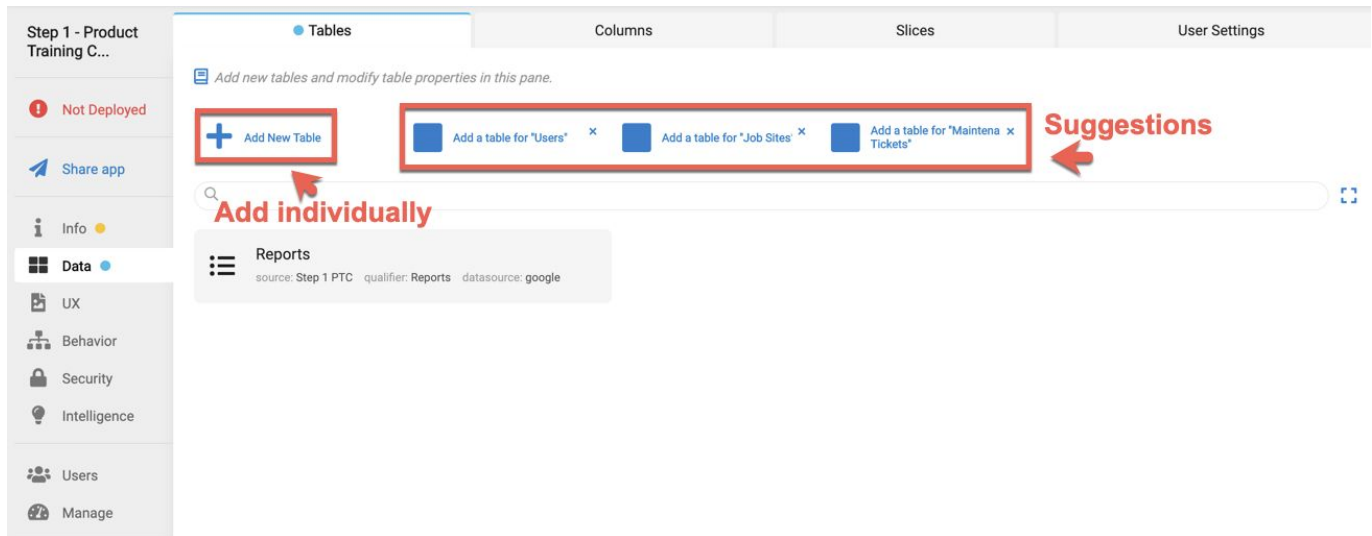
- [Tables: The Essentials](#)

Add the remaining tables from the spreadsheet into the app. If the remaining tables exist in tabs within a single file, AppSheet will make suggestions. Otherwise, the tables can be added individually by clicking the “Add New Table” button.



You can have multiple data sources in a single app; you can have a single data source serve multiple apps. The experience in the editor is the same regardless of where the data is stored and how many apps that data source serves.

## Screenshot of Data page in AppSheet:



### Steps:

1. Click the suggestions one at a time to load each table into the app; Or use the “Add New Table” button to add each table individually
2. Go to Data > Tables
3. Click “Reports”
4. Set “Are updates allowed?” to permit adding rows

## 2.5 Update column settings for Reports table

### Skills:

- [Columns: The Essentials](#)



The app definition and the data are connected but separate. You could think of the app definition as a set of directions that explain how to treat the data in the spreadsheet. Updating the column settings is a first step toward adding intelligence to the app and making it more than just a clunky old spreadsheet.






AppSheet made assumptions about the type of data in each column when adding the tables. It is the app creator’s responsibility to verify the data type assumptions and add additional settings unique to the app.

In addition to data types, the column settings view includes many other settings. You can access many of these settings quickly inside the editor. Depending on screen size, you’ll generally need to scroll to the right to see all the settings.

### Screenshot of columns settings:

Reports													
11 columns: Report ID Report ID													
NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?	EDITABLE?	REQUIRED?	INITIAL VALUE	DISPLAY NAME	DESCRIPTION	SEARCH?	SCAN?	NP?
RowNumber	Number	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			Number of this row	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Report ID	Text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	UNIQUE()			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Resolution Date	Date	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	TODAY()			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Job Site ID	Url	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
User Email	Email	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Report Type	Text	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Priority	Text	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Report Date	Date	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	TODAY()			<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Description	LongText	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Image	Image	<input type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Needs Review	Text	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

You can access these settings in addition to more advanced settings by clicking the edit pencil next to the column name:

Reports													
11 columns: Report ID Report ID													
View Table Add Virtual Column Regenerate Structure													
NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?								
 _RowNumber	Number	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/>								
 Report ID	Text	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=	<input checked="" type="checkbox"/>								
 Resolution Date	Date	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>								
 Job Site ID	Url	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>								
 User Email	Email	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/>								

The settings in this view will vary based on the data type of the column.

Screenshot of column settings for a Text column:

REPORTS : REPORT ID

type: Text

Done

Show standard view

Column name

Column name

Report ID

Show?

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

ON

Type

Column data type

Text

TYPE DETAILS

Maximum length

—

+

Minimum length

—

+



**Pro tip:** In the top right of the column settings, there is an options to “Show expanded view.” When creating apps, it can be helpful to see the expanded view by default. You can set this behavior as follows:

1. Go to Manage > Author
2. Click “Editor Settings”
3. Set “Expand all views” to “ON”

## 2.5.1 Update Report ID

Skills:

- [Keys](#)
- [UNIQUEID\(\)](#)

The Report ID is going to be the key for this table.

There are four rules to keep in mind about keys:

1. Exist - Every table must have a key.
2. Non-blank - Every row of the table must have a value in the key column (i.e. the field cannot be blank).
3. Unique - The value must be unique from every other value in the column.
4. Constant - The value must never change.

It is best to use a random alphanumeric sequence for a key. By virtue of the number of possibilities, this ensures the key will not be repeated. AppSheet makes this easy by providing the `UNIQUEID()` function. We'll see an exception to this in the Users table where we use email addresses as the key.

It is best to avoid a sequence of numbers that increases by one for each row (such as `_RowNumber`). When AppSheet is operating in offline mode, a user can create new records. Each new record will be assigned a key according to the rules setup in the app. If the key is assigned based on row number and two people are adding data while working offline, they could create records with the same key. This is a problem when those users come online and push the new records to the data source. Using `UNIQUEID()` will eliminate this issue.

Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to "Report ID"
3. Ensure "Type" is set to "Text"
4. Ensure "Initial value" is set to `UNIQUEID()`
5. Ensure Key is set to On

## 2.5.2 Create a reference between Reports and Job Sites

Skills:

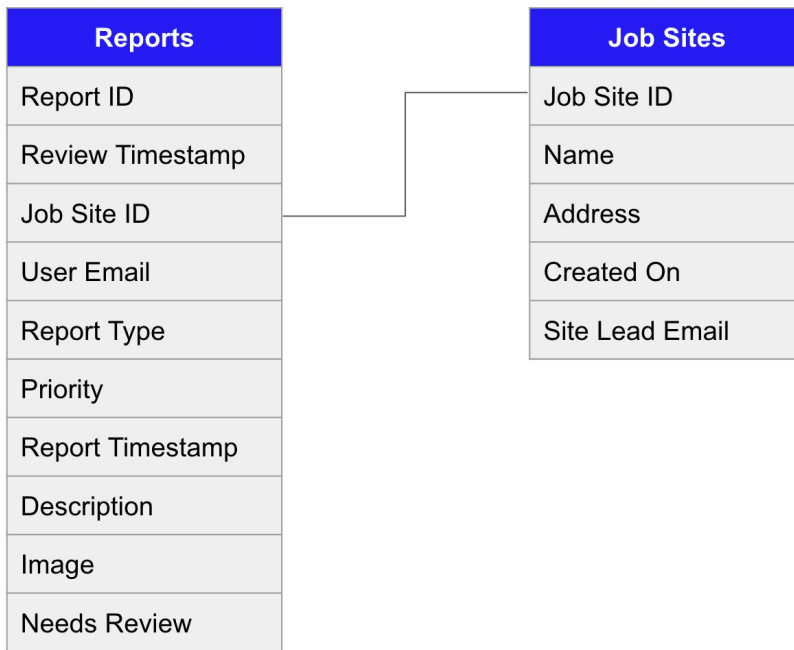
- [References Between Tables](#)

One powerful feature of AppSheet is the ability to create references between tables. A reference allows us to link two pieces of information together. There are many benefits to using references in your apps.



References allow you to connect one record (i.e. the parent record) to many other records (i.e. the child records) without repeating information. In our app, a job site could have many safety reports. Without references, we would need to store all the information about the job site in every safety report. A more efficient way to structure this data is to store the information about the report in one table and the information about the job site in another table. The report table only needs one column to connect it to the job site. The report then has access to all information about the job site without having to duplicate it.

Diagram of data connection:



Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to "Job Site ID"
3. Set "Type" to "Ref"
4. Set "Source table" to "Job Sites"

### 2.5.3 Create a reference between Reports and Users

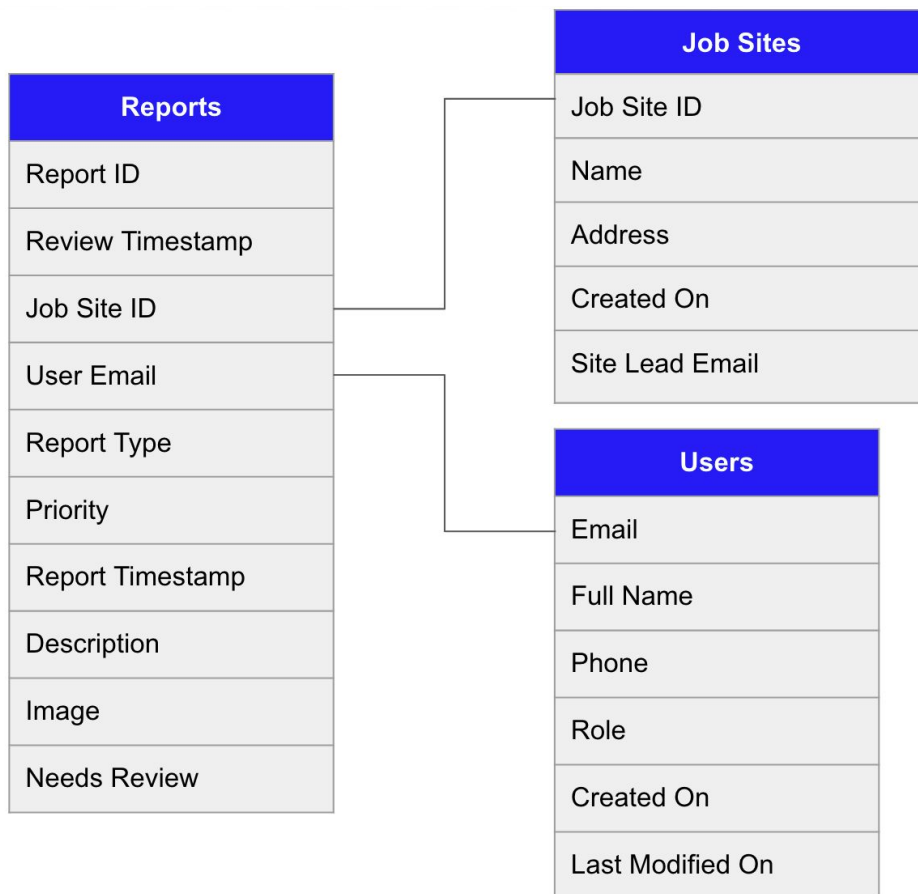
Skills:

- [References Between Tables](#)
- [USEREMAIL\(\)](#)

Similar to the reference between Reports and Job Sites, the reference between Reports and Users will allow us to access information about the user that created the report without duplicating that user's information in the Reports table.

For this reference, we will set the initial value equal to the logged in user's email using the `USEREMAIL()` function. The value stored in a reference column must be equal to the key of the referenced table. By setting the initial value to the logged in user's email address, I have dropped a hint about what column will be used as the primary key in the Users table.

Diagram of data connection:



#### Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to "User Email"
3. Set "Type" to "Ref"
4. Set "Source table" to "Users"
5. Set "Initial value" to `USEREMAIL()`
  - a. The app may throw a **red colored error** for a moment, but you can ignore this.
6. Click "Save"

## 2.5.4 Create a label

#### Skills:

- [Row Labels](#)
- [Virtual Columns](#)
- [CONCATENATE\(\)](#)

In addition to a key, every table must have a label. The label does not need to be unique and it can change. It will be the way that humans refer to the record. Comparing this to humans themselves, you can think of the **key** as a person's **passport number** and the **label** as their **name**. You'll never have the same passport number as someone else. Conversely, two people can have the same name and can change their names, but it is easiest if there aren't too many Jimmys in the room and if Cathy doesn't decide to go by Sue without telling anyone.

When describing some tables, the best label may not correspond to a specific column. In this case, we are going to make a virtual column that will be used as the label.

A virtual column is one that exists in the app, but not in the data source. The virtual column is often made up of information from other columns. You can apply functions to this information, such as arithmetic for numbers or concatenation for text. When the information used in a virtual column changes, the virtual column will be updated.

Steps:

1. Go to Data > Columns > Reports
2. Click "Add Virtual Column"
3. Set "Column name" to "Report Name"
4. Set "App formula" to `CONCATENATE([User Email].[Full Name], " - ", [Report Timestamp])`
5. Set "Label" to "On"
6. Click "Done"
7. Scroll through the list of columns in the Reports table and set "Label" to "Off" for all columns except this one. "Report Name" should be the only column with "Label" set to "On."

## 2.5.5 Update Review Timestamp

Skills:

- [NOW\(\)](#)
- [TODAY\(\)](#)
- [TIMENOW\(\)](#)

AppSheet makes an intelligent assumption about what type of data is in each column based on the column header. Following are three common column header keywords related to date and time. When a keyword is present, AppSheet will choose the corresponding data type and initial value.

Column Header Keyword	Data Type	Fomat	Initial Value
Timestamp	DateTime	MM/DD/YYYY HH:MM:SS AM/PM <i>4/1/2010 3:14:00 AM</i>	NOW()
Date	Date	MM/DD/YYYY <i>4/1/2010</i>	TODAY()
Time	Time	HH:MM:SS AM/PM <i>3:14:00 AM</i>	TIMENOW()



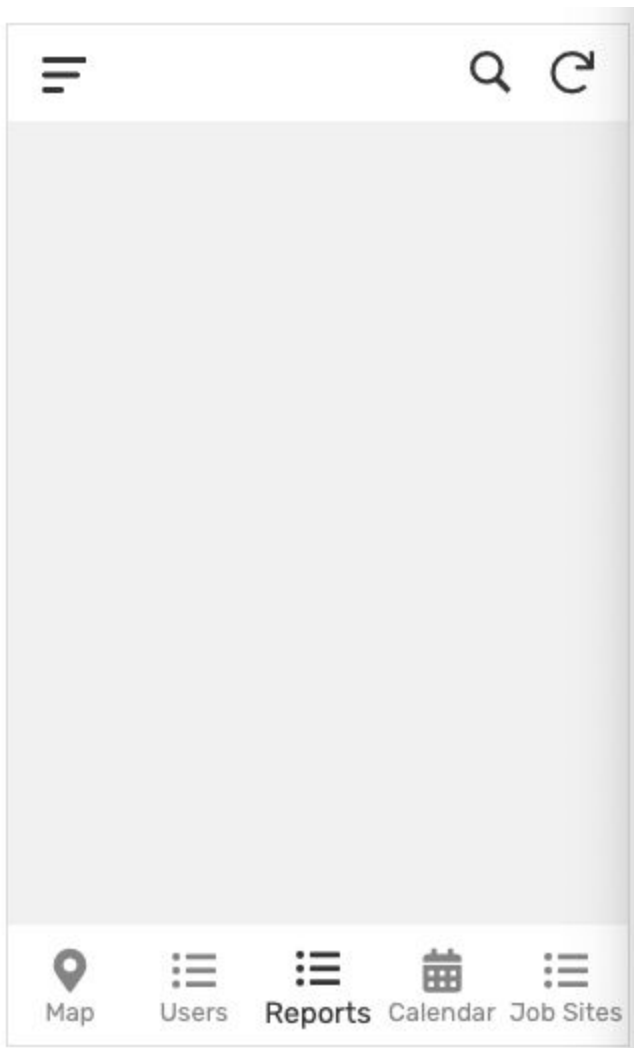
Since the column header includes the word “timestamp,” AppSheet has assumed we want the DateTime type and set the initial value to NOW(). In this case, the review timestamp should not be set until a safety report is resolved.

Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to Review Timestamp
3. Erase the expression from “Initial value”

## 2.5.6 Test the Form

We have completed the initial steps to setup the Reports table. This is a good time to see how things look in the app. When you added the Reports table, AppSheet created a view called Reports. Click that view in the Preview screen.



There should be an add button in the bottom right.



If you do not see an add button, it is because the table does not allow the user to add new records. To change this:

1. Go to Data > Tables > Reports
2. Click “Adds”

To see what the form looks like so far, click the add button in the bottom right of the Preview. Here’s what we have so far (note: the email address will be set to the email address you logged in with):

←

Report ID\*  
f99ff66b

Resolution Timestamp  
mm/dd/yyyy, --:--:-- -

Job Site ID

User Email  
derek@appsheet.com

Cancel Save

You can scroll through the form to see the remaining fields. The remainder of Section 2.4 will explain how to update settings to fine-tune the Reports form:

## 2.5.7 Set “Display name”

Skills:

- [Display Name](#)

The display name is what will appear in the form-view of a table. This is useful when you want the field label to be different from the column header. For example, the Reports table has a column called “Job Site ID.” This is a useful column header, but the “ID” part is unnecessary information inside the form. The following steps will update the field label of this column.

Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to “Job Site ID”
3. Set “Display name” to “Job Site”

Follow the same process to set the “Display name” for the “User Email” column to “Reported by.”

## 2.5.8 Create option buttons

Skills:

- [Dropdown from Simple List of Values](#)
- [Column Types: Diving Deeper](#)

Two common ways to create buttons in your form are using the data types Enum or EnumList. These two data types differ by the number of options you are allowed to select.

An Enum type allows you to select one option from a group. This is similar to how radio buttons work.

An EnumList type allows you select multiple options from a group. This is similar to how check boxes work.

The following steps will create button options for “Report Type.”

Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to “Report Type”
3. Set “Type” to “Enum”
4. Next to “Values,” click the “+” button to add two options:
  - a. “Incident”
  - b. “Near Miss”
5. Set “Base type” to “Text”
6. Set “Input mode” to “Buttons”

Follow the same process to create buttons for the “Priority” column. The button options are “Critical,” “Normal,” and “Low.”

## 2.5.9 Yes/No data type

Skills:

- [Yes/No Expressions](#)

The “Needs Review” column will be used in the app to determine if a record should be used in a specific view. This type of column is often referred to as a “flag” and given a Yes/No data type. A Yes/No data type is the same as True/False. AppSheet will interpret “Yes” and “True” to mean the same thing for this column type. Similarly for “No” and “False.” The following steps setup this functionality for the “Needs Review” column.

Steps:

1. Go to Data > Columns > Reports
2. Click the edit pencil next to “Needs Review”
3. Set “Type” to “Yes/No”
4. Set “Initial value” to True

## 2.5.10 Show and Require

Skills:

- [Show\\_If Column Constraint](#)
- [CONTEXT\(\)](#)
- [AND\(\)](#)

Often times, we need to control when a column is shown. AppSheet allows you to quickly toggle Show on or off. Additionally, you can use a formula for cases where you need to show the column under specific circumstances.

The CONTEXT() function refers to the name or type of view. This is frequently used for fields that are automatically set using an initial value and therefore do not need to be shown in a form. For example, see “Needs Review” in the list below.

The “Require?” function controls whether a form can be saved without filling in a specific column. Fields with “Require?” set to “On” will have an asterisk next to them in the form.

Following is the list of columns for the “Reports” table along with the target “Show?” And “Require?” settings. To set a formula, click the edit pencil next to the column name, then click the flask next to the “Show?” setting.

Show?

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



Update your app settings to match the highlighted fields below:

Column Name	Show?	Require?
_RowNumber	Off	Off
Report ID	Off	On
Review Timestamp	CONTEXT("ViewType") <> "Form"	Off
Job Site ID	On	On
User Email	On	On
Report Type	On	On
Priority	On	On
Report Timestamp	Off	On
Description	On	On
Image	On	On
Needs Review	CONTEXT("ViewType") <> "Form"	On
Report Name	CONTEXT("ViewType") <> "Form"	Off

Note: when “Show?” is set to “Off,” the field must also have “Searchable” set to “Off.” AppSheet will update “Searchable” automatically and list a Warning to inform you of the change. If the change in searchability for the field is acceptable, no additional action is required.



**Pro Tip:** When you click the flask next to the Show toggle, a text field will become available to enter a show\_if formula. Saving without entering a formula will result in an error similar to the following:

Column Name 'Report ID' in Schema 'Reports\_Schema' of Column Type 'Text' has an invalid show\_if constraint '='. Expression " does not match the expected format of an AppSheet expression [MORE INFO](#)

Fix this by entering a formula or clicking the “x” next to the text field to cancel. After making the update, you must Save to clear the error message.

## 2.6 Update column settings for Users table

Skills:

- [References Between Tables](#)
- [Tracking Changes Using “Change” Column Types](#)

Go to Data > Columns > Users. Notice that a virtual column called “Related Reports” was created. This column allows each row of the Users table to hold a list of Related Reports. A Related Report is a record from the Reports table that points to a record in the Users table. AppSheet creates and updates this column automatically.

Using the same tools described in Section 2.4, set the “Email” column to be the “Key.” Set the “Full Name” column to be the “Label.”

Update your app settings to match the highlighted fields below:

Column Name	Type	Show?	Require?	Initial Value
_RowNumber*	Number	Off	Off	
Email	Email	On	On	
Full Name	Name	On	On	
Phone	Phone	On	On	
Role	Enum	On	On	
Created On	DateTime	Off	On	NOW()
Last Modified On	ChangeTimestamp	Off	On	NOW()
Related Reports*	List	On	Off	

\*These columns are created and maintained automatically by AppSheet

For the “Role” column, add Type Details as shown in the following screenshot:

TYPE DETAILS

Values

Onsite Safety

Site Lead

Safety Board

+

Allow other values

OFF

Auto-complete other values

OFF

Base type

Text

Input mode

Auto

Buttons

Stack

Dropdown

## 2.7 Update column settings for Job Sites table

Using the same tools described in Section 2.4, set the “Job Site ID” column to be the “Key.” Set the “Name” column to be the “Label.”

Update your app settings to match the highlighted fields below:

Column Name	Type	Show?	Require?	Initial Value
_RowNumber*	Number	Off	Off	
Job Site ID	Text	On	On	UNIQUEID()
Name	Text	On	On	
Address	Address	On	On	
Created On	DateTime	On	On	NOW()
Site Lead Email	Ref	Off	On	
Related Reports*	List	On	Off	

\*These columns are created and maintained automatically by AppSheet

For the “Site Lead Email” column, add Type Details as are shown in the following screenshot:

TYPE DETAILS

Source table

Which table does this column reference?

Users

[see definition](#)

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).

☐ OFF

External relationship name

Relationship name in the underlying data source (use for Salesforce data only)

Input mode

AutoButtonsDropdown

## 2.7.1 Create a slice for Site Leads

Skills:

- [Slices: The Essentials](#)

Slices are subsets of data. These are useful anytime you need to access a particular subset of data on a regular basis. You can use a formula to define the slice. Then access the slice throughout your app in the same way you access a table.

In the “Site Lead Email” column of the “Users” table, it would be great if we can limit the dropdown of emails to only those that correspond to a person with the role “Site Lead.” We can achieve this with a slice.

Steps:

1. Go to Data > Slices
2. Click “Add New Slice”
3. Set “Slice Name” to “Site Leads”
4. Set “Source Table” to “Users”
5. Set the “Row filter condition” to `[Role] = "Site Lead"`
6. Unhighlight “Deletes”
7. Save the app

## 2.7.2 Update the “Site Lead Email” column

Now we can reference the newly created slice inside the “Site Lead Email” column. This will limit the dropdown to only include users with role “Site Lead.”

Steps:

1. Go to Data > Columns > Job Sites
2. Click the edit pencil next to “Site Lead Email”
3. Set “Source table” to “Site Leads (slice)”

## 2.8 Update column settings for Maintenance Tickets table

Using the same tools described in Section 2.4, set the “ID” column to be the “Key.” Set the “Title” column to be the “Label.”

Update your app settings to match the highlighted fields below:

Column Name	Type	Show?	Require?	Initial Value
_RowNumber*	Number	Off	Off	
ID	Text	Off	On	UNIQUEID()
Job Site ID	Ref	Off	On	



Created By	Ref	Off	On	USEREMAIL()
Created On	DateTime	Off	On	NOW()
Title	Text	On	On	
Description	LongText	On	On	
Priority	Enum	On	On	
Set Deadline	Yes/No	On	On	
Required By	Date	[Set Deadline]=True	Off	
Status	Enum	CONTEXT(ViewType) <> "Form"	On	"Open"

\*These columns are created and maintained automatically by AppSheet

For the "Job Site ID" column, set the "Source table" to "Job Sites."

For the "Created By" column, set the "Source table" to "Users."

For the "Priority" column, add Type Details as shown in the following screenshot:

**TYPE DETAILS**

Values

- Critical
- Normal
- Low

Allow other values: OFF

Auto-complete other values: OFF

Base type: Text

Input mode: Auto, **Buttons**, Stack, Dropdown

For the "Status" column, add Type Details as shown in the following screenshot:

TYPE DETAILS

Values

Open

Closed

+

Allow other values

OFF

Auto-complete other values

OFF

Base type

Text

Input mode

Auto

Buttons

Stack

Dropdown

## 2.9 Add reference between Reports and Maintenance Tickets

Skills:

- [Regenerate Table](#)



We know Reports and Maintenance Tickets need to be connected, but which one is the parent and which is the child? Remember, a parent can have many children, but a child can have only one parent. Think about how events transpire at a job site. Something breaks and a bad thing happens. A report is issued about the bad thing and a maintenance ticket is created for the fix. Another bad thing happens because of the same thing while waiting for maintenance. So another report is created and it references the same ticket as the first event. In this scenario, there are multiple reports referencing the same ticket. The maintenance ticket is the parent and the reports are the children.

To set this up, the Reports table needs a column to reference the Maintenance Tickets table. Each record in the Reports table will populate this column with the ID of the relevant maintenance ticket. If a ticket doesn't already exist, the user will create a new one. After creating the column with data type Ref on the Reports table, AppSheet will automatically create a virtual column on the Maintenance Tickets table that holds a list of related reports for each ticket.

When any data source is modified by adding, removing, or renaming a *column*, the associated table must be regenerated. When a table is regenerated, AppSheet reads the table from the data source and updates the structure in the app definition. A table does not need to be regenerated when a *row* of data is added, removed, or modified.

Steps:

1. Go to Data > Tables > Reports
2. Click "View Source" to open the spreadsheet
3. Ensure you are in the "Reports" table of the spreadsheet
4. Add a column to the right of the last column
5. Set the column header to "MT ID" (This stands for "Maintenance Ticket ID")
6. Go to Data > Columns > Reports
7. Click "Regenerate Structure"
8. Click "Are you sure?"
9. Click the edit pencil next to "MT ID"
10. Add the following Show\_if formula: `LOOKUP(USEREMAIL(), Users, Email, Role)<>"Onsite Safety"`
11. Set "Type" to "Ref"
12. Set "Source table" to "Maintenance Tickets"

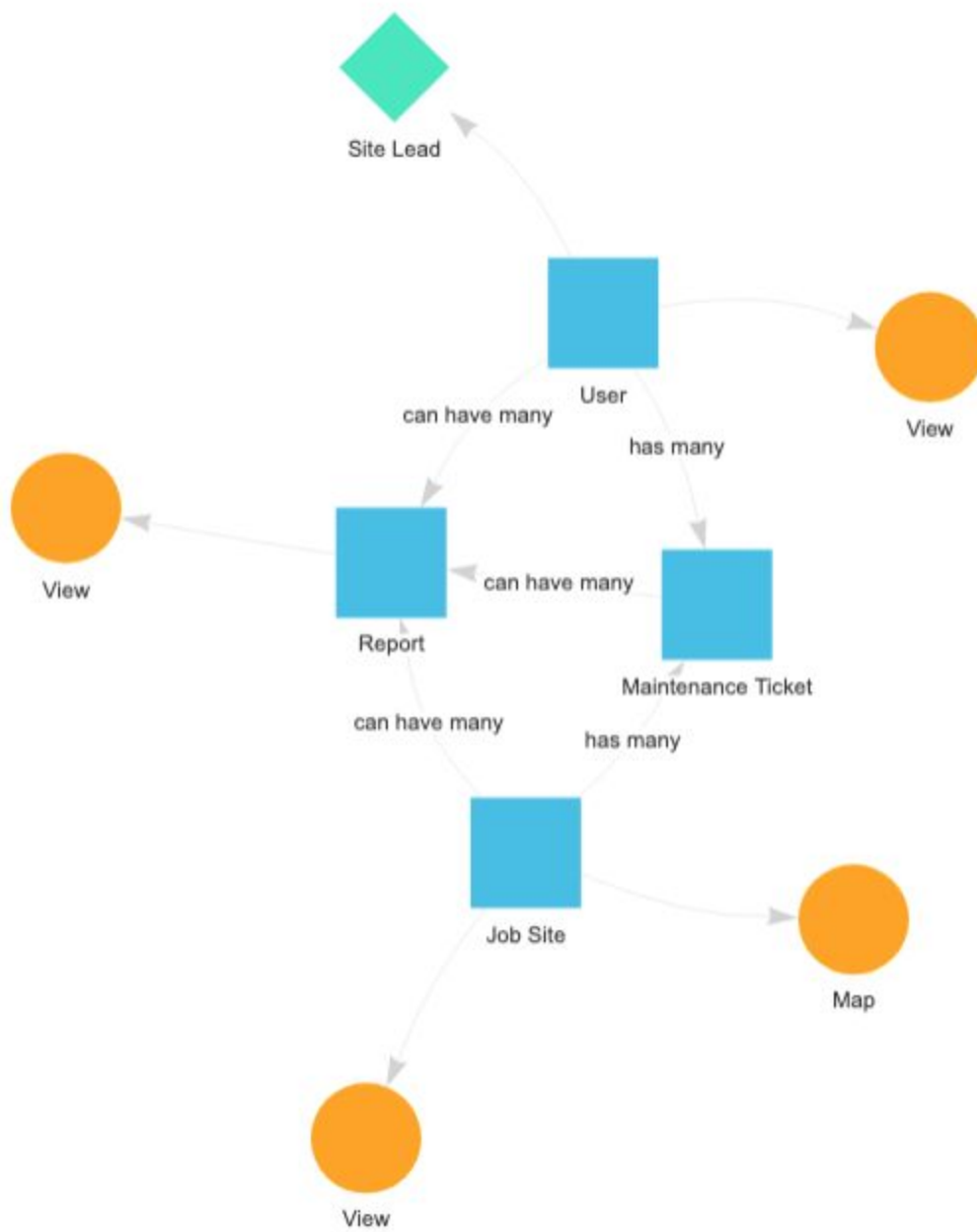


**Pro Tip:** It is best to make small incremental changes when adding, removing, or modifying a table. If you need to make several changes to a table, try doing them one at a time and regenerating after each change. This will minimize the chance that your pre-existing column settings will be affected.

## 2.10 Wrap Up

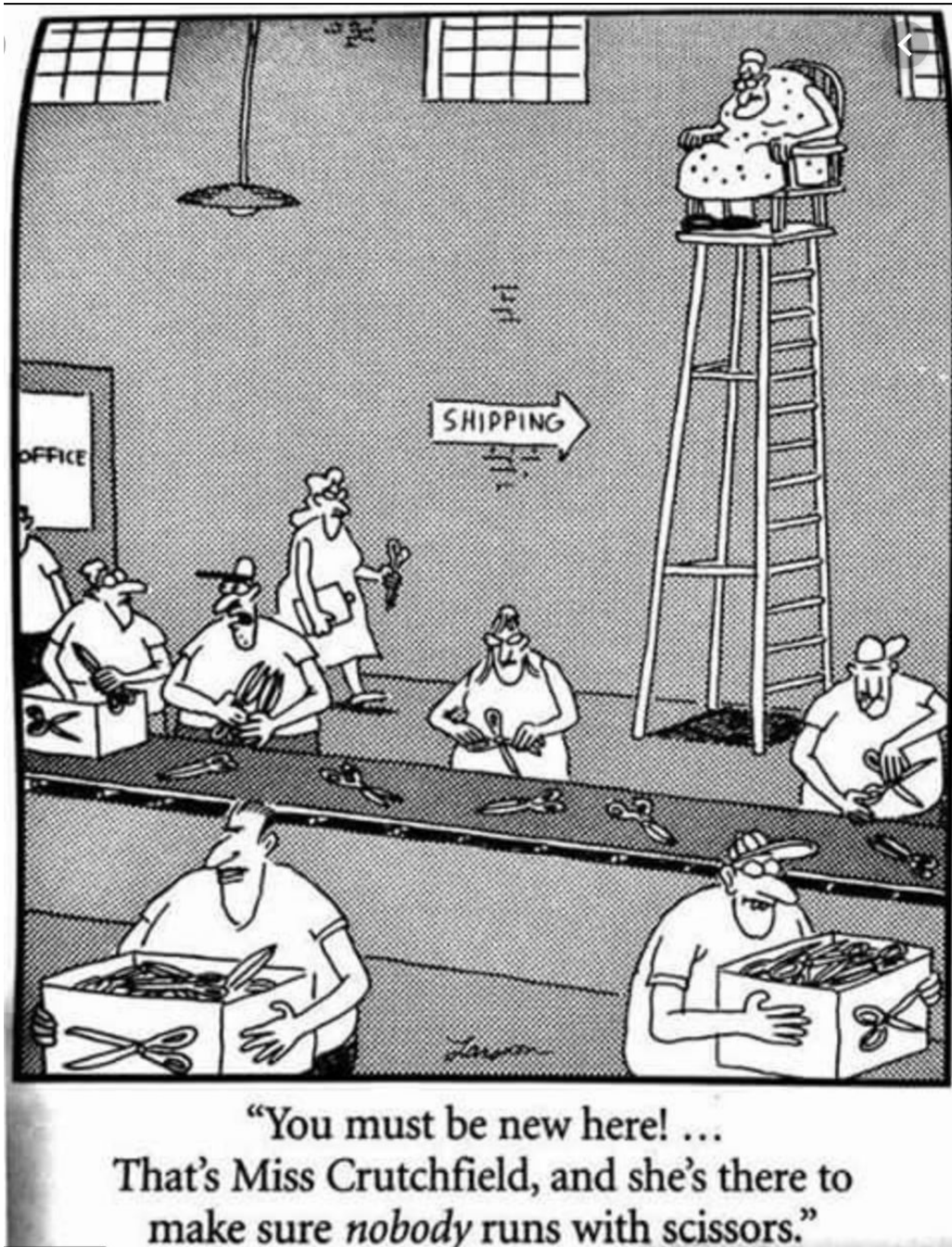
Congratulations! You have setup the tables and column settings that will be the foundation of the app. The next section describes how to setup views.

Go to Info > Spec to see the relationships between your tables. You will see connectors drawn between all the tables with references setup.



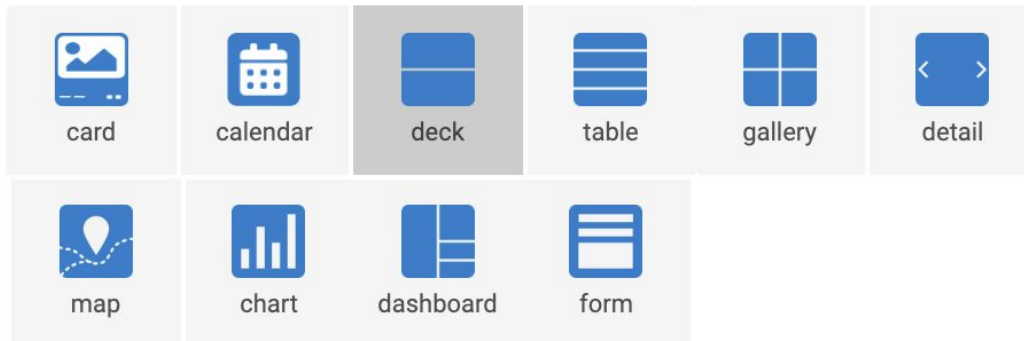
## Checkpoint #1

[Click here](#) to view the sample app for this checkpoint. Copy the app to your account by clicking “Copy and Customize.” After copying the app, add your email address to the Users table.



## 3. Views

This section focuses on setting up views for your app. AppSheet offers several ways to view your data. The options at the time of writing are listed below:



So you've got all this data. You know you need to look at it. And you have all these options. What do? Think about what your user cares about. Do they want to see a lot of information about a lot of records all at once? Table View. Do they want to see enough information about the records to make a selection but not so much as to be overwhelmed? Deck View paired with a Detail View. Before looking at the options, try sketching out what your user needs to see. Understand your intent. Then open AppSheet and choose which view most closely aligns with your intent.

### 3.1 Create a view to manage user roles

Skills:

- [User roles](#)



We want the experience in the app to be tailored to the role of the user. This means certain views should only be visible to users with a certain role. We have already setup the data to support this by including a “Role” column in the Users table. Now we need a view where we can read and edit what role a person is assigned to.

AppSheet also offers built-in functionality to manage user roles. However, when more than two roles are needed, it is necessary to create your own user management system. In our case, we need three roles and we want them to have specific names (i.e. Onsite Safety, Site Lead, Safety Board). So we will create our own method for managing each user's role.

Screenshot of Users table:

	A	B	C	D	E	F
1	<b>Email</b>	<b>Full Name</b>	<b>Phone</b>	<b>Role</b>	<b>Created On</b>	<b>Last Modified On</b>
2	derek@bricklinholdings.com	Derek Covey	206-549-4234	Safety Board	7/30/2019 9:24:24	9/19/2019 8:31:32
3	john@example.com	John Jones	555-555-5555	Onsite Safety	7/30/2019 9:24:24	7/30/2019 9:24:56
4	millie@example.com	Millie Fleer	555-555-5555	Site Lead	7/30/2019 9:24:24	7/30/2019 9:24:56
5	flora@example.com	Flora Dora	555-555-5555	Onsite Safety	7/30/2019 9:24:24	7/30/2019 9:24:56
6	george@example.com	George Jones	555-555-5555	Safety Board	7/30/2019 9:24:24	7/30/2019 9:24:56
7	derek@appsheet.com	Derek Saftee	206-549-4234	Site Lead	7/31/2019 16:23:36	9/23/2019 8:13:18

AppSheet already created a Users view when we added the table. Next we'll move that view to the menu and use it to create a profile for ourselves inside the app.



**Pro tip:** adding a user to the Users table does not automatically give them access to the app. The existence of a user in the Users table allows you to control what information a person sees inside the app. Access to the app itself is managed through the Users tab of the app editor. We will revisit user management later in this course.

Steps:

1. Go to UX > Views
2. Click on "Users"
3. Change "Position" to "menu"
4. Save the app
5. Click on the hamburger menu in the top left of the preview and on the Users table
6. Click the "+" button to create a new record. Enter your information into the form.
7. Save the form



## 3.2 Create form view to report events

Skills:

- [LOOKUP\(\)](#)
- [Yes/No Expressions](#)



We want to enable to Onsite Lead to quickly enter information on the job. We already setup the Reports table that will hold this information. Now we need a way to add records to the table. A Form View is the best choice for adding new records. And now that we have the Users table setup, we can use the information from the Role column to limit visibility of the form to users that have the role “Onsite Safety.”

To ensure you can see the view after it is created, open the Users table and change your role to “Onsite Safety.”

Steps:

1. Go to UX > Views
2. Click “+ Add New View”
3. Set “View Name” to “Report an Event”
4. Set “For this data” to “Reports”
5. Set “View type” to “form”
6. Set “Position” to “center”
7. Set “Auto re-open” to “ON”
8. Set “Icon” to “bullhorn”
9. Add the following “Show if” formula:

```
LOOKUP(USEREMAIL(), Users, Email, Role)="Onsite Safety"
```

The show\_if formula uses the LOOKUP() function, which is equivalent to vlookup in most spreadsheet applications.

In this app, we are using a simple single-page form. App creators interested in creating multi-page forms will benefit from the following resources:

- [Multi-page Forms with Conditional Branching](#)
- [Sample app](#) showing minimum setup for multi-page forms



### 3.3 Create table view to see reports for logged in user



It would be helpful if users could see all the reports they have entered. This could be used for tracking status or for confirming they got the information entered. We can accomplish this by creating a slice to grab records associated with the logged in user. Then create a view to show the slice. This view is only helpful for Onsite Safeties, so we can limit visibility as well.

Steps:

1. Go to Data > Slices
2. Click "Add New Slice"
3. Set "Slice Name" to "My Reports"
4. Set "Source Table" to "Reports"
5. Add the following "Row filter condition" formula:

```
AND([Needs Review] = True, [User Email] = USEREMAIL())
```

6. Highlight "Deletes" by clicking the button
7. Go to UX > Views
8. Click "Add New View"
9. Set "View Name" to "My Reports"
10. Set "For this data" to "My Reports (slice)"
11. Set "View type" to "table"
12. Set "Position" to "right most"
13. Set "Icon" to "ad"
14. Add the following "Show if" formula

```
OR(LOOKUP(USEREMAIL(), Users, Email, Role)="Onsite Safety",  
ISBLANK(LOOKUP(USEREMAIL(), Users, Email, Role)))
```

### 3.4 Add a security filter to Reports table

Skills:

- [Security Filters: The Essentials](#)
- [IF\(\)](#)

We added a view that allows the Onsite Safety to see and edit reports. If we set everything up correctly, they should only see events they created. However, with the use of slices and the interconnection of data in the app, it is possible we missed something.



Security filters give us a single location where we can authoritatively limit the data accessible to a user. Security filters are applied before the data is sent to the user's device. Data that does not pass the filter is never loaded on the device. So there is no risk that an inadvertent series of clicks could expose sensitive information.

Steps:

1. Go to Security > Security Filters
2. Click on "Reports"
3. Add the following "Security filter" formula:

```
IF(
OR(
LOOKUP(USEREMAIL(), Users, Email, Role)="Site Lead",
LOOKUP(USEREMAIL(), Users, Email, Role)="Safety Board"
),
true,
[User Email]=USEREMAIL()
)
```

### 3.5 Create form view for Maintenance Tickets



When the Site Lead is reviewing a report, she may want to create a maintenance ticket. We already setup the table to hold this information, but we need a Form View so the user can add information to the table.

AppSheet automatically creates a form view when a table is added. The system-generated form view cannot be customized. If a custom form is needed, the app creator must create a new form view. A single table can have multiple views of any type created for it.

We are creating this form so that it will be displayed in the primary navigation bar for "Safety Board" users.

Steps:

1. Go to UX > Views
2. Click on "Add New View"
3. Set "View Name" to "Create Maintenance Ticket"
4. Set "For this data" to "Maintenance Tickets"
5. Set "View type" to "form"
6. Set "Position" to "right most"
7. Set "Icon" to "wrench"
8. Set "Show if" to `LOOKUP(USEREMAIL(), Users, Email, Role)="Safety Board"`

### 3.6 Create deck view for Maintenance Tickets



In addition to creating new Maintenance Tickets, our users probably want to see what Maintenance Tickets already exist. The plan is to create a dashboard that will show all the safety reports, details about the reports, and a list of maintenance tickets with enough

information for the Site Lead to know if an open ticket already exists to resolve the event or if she should create a new one. A list of records with basic information is generally best accomplished with a Deck View.

The view created by the steps below gives “Site Lead” users quick access to a list of open maintenance tickets. Something is wrong with how it is set up though. See if you can find the problem!

Steps:

1. Go to UX > Views
2. Click “Add New View”
3. Set “View name” to “Open Maintenance Tickets Deck”
4. Set “For this data” to “Maintenance Tickets”
5. Set “View type” to “deck”
6. Set “Position” to “right most”
7. Set “Primary header” to “Title”
8. Set “Secondary header” to “Description”
9. Set “Show action bar” to “OFF”
10. Set “Icon” to “wrench”
11. Set “Show if” to `LOOKUP(USEREMAIL(), Users, Email, Role) = "Site Lead"`

Here’s the problem: The name suggests the view includes only *open* maintenance tickets. However, the view is based on the Maintenance Tickets table. This will result in the entire Maintenance Tickets table being displayed.

Slices are the tool that allow us to display a subset of data. Let’s create a slice that captures only the open maintenance tickets.

Steps:

1. Go to Data > Slices
2. Click “Add New Slice”
3. Set “Slice Name” to “Open Maintenance Tickets”
4. Set “Source table” to “Maintenance Tickets”
5. Set “Row filter condition” to `[Status] = "Open"`
6. Go back to the “Open Maintenance Tickets Deck” view
7. Change “For this data” to “Open Maintenance Tickets (slice)”

## 3.7 Update the Job Sites map view and list view

When a table is added with “Address” in a column header, AppSheet will automatically create a map view. The default view is close to what we need. The steps below will customize the label and add logic for visibility.

Steps:

1. Go to UX > Views
2. Click on “Map”

3. Set “View name” to “Job Sites”
4. Set “Position” to “right”
5. Add the following “Show if” formula:

```
OR(  
  LOOKUP(USEREMAIL(), Users, Email, Role)="Site Lead",  
  LOOKUP(USEREMAIL(), Users, Email, Role)="Safety Board"  
)
```

AppSheet added a “2” to the view title to avoid a naming conflict with the other view called “Job Sites.” The other “Job Sites” view is a list view created automatically when we added the “Job Sites” table. We can update the settings for the list view and change the name.

Steps:

1. Click on the “Job Sites” view
2. Set “View name” to “Job Sites deck”
3. Set “Position” to “ref”
4. Add a “Sort by” setting
  - a. Set column to “Name”
  - b. Set order to “Ascending”
5. Set “Primary header” to “Name”
6. Set “Secondary header” to “Address”
7. Set “Show action bar” to “OFF”
8. Go back to “Job Sites 2” view
9. Change the “View name” to “Job Sites”

## 3.8 Clean up unused views

The “Reports” view and the “Calendar” view in the Primary Views section were created automatically when we added tables. We aren’t using these views in the app and can delete them.

Steps:

1. Go to UX > Views
2. Click on “Calendar”
3. Click “Delete”
4. Click “Are you sure”
5. Repeat for “Reports”

## 3.9 Create a dashboard for the Site Lead

Skills:

- [View Types](#)



In our process, the Onsite Safety collects information and creates a report. Then the Site Lead reviews the report, verifies the details, and creates a maintenance ticket if necessary. If a maintenance ticket already exists for the issue in the safety report, then she needs to add the report to the existing ticket. A dashboard allows us to combine multiple views into a single screen so the Site Lead can perform all these tasks from a single location.

Since dashboards display a lot of information in one view, it is generally aimed at desktop users. For mobile users, you can choose to display each view as a separate tab or stack them vertically.

### 3.9.1 Create view to show new reports

Again, we need to show a subset of data from the reports table. Specifically, all reports where the [Needs Review] column is "True." To do this, we will create a slice. Then we can create a view to display data from the slice.

Steps:

1. Go to Data > Slices
2. Click "Add New Slice"
3. Set "Slice Name" to "Needs Review"
4. Set "Source Table" to "Reports"
5. Set "Row filter condition" to `[Needs Review] = True`
6. Save the changes
7. Go to UX > Views
8. Click "Add New View"
9. Set "View name" to "Needs Review"
10. Set "For this data" to "Needs Review (slice)"
11. Set "View type" to "deck"
12. Set "Position" to "ref"
13. Set the following fields:

Sort by  
Sort the rows by one or more columns.

≡	Priority	⬇	Ascending	⬆	🗑
+					

Group by  
Group rows by the values in one or more of their columns.

≡	Job Site ID	⬇	Ascending	⬆	🗑
+					

Group aggregate  
Display a numeric summary of the rows in each group.

COUNT	⬆
-------	---

14. Set the following fields:

Primary header  
The top text for each row

Report Name	⬆
-------------	---

Secondary header  
The bottom text for each row.

Priority	⬆
----------	---

### 3.9.2 Create a detail view

Skills:

- [Quick Edit](#)

We need to display details for a report after it is selected from the list. We can do this with a detail view. To improve the appearance, we can select only the specific rows the Site Lead needs to see.

The Site Lead needs to add the report to a maintenance ticket. To make this easy, we can create a quick edit column. Adding a column to the quick edit list allows the user to edit the value from the detail view. Alternatively, the user would need to click the edit button, make modifications in a form, and save the changes. Quick edit can speed up the process and is especially useful when the user only needs to edit a subset of the fields.

Steps:

1. Go to UX > Views
2. Click “Add New View”
3. Set “View name” to “Needs Review\_Detail”
4. Set “View type” to “detail”
5. Set “Position” to “ref”
6. Set the following fields:

Quick edit columns

Which columns can be edited directly in the slide.

≡

MT ID

⬆⬇⬆

🗑

+

Sort by

Sort the rows by one or more columns.

+

Column order

Display columns in a different order than they appear in the original data.

≡

MT ID

⬆⬇⬆

🗑

≡

Review Timestamp

⬆⬇⬆

🗑

≡

Job Site ID

⬆⬇⬆

🗑

≡

Report Type

⬆⬇⬆

🗑

≡

Priority

⬆⬇⬆

🗑

≡

Description

⬆⬇⬆

🗑

≡

Image

⬆⬇⬆

🗑

+

### 3.9.3 Create a dashboard

When creating a dashboard, there is an option for “Interactive mode.” This affects the behavior when a user clicks a row inside a table view of a dashboard.

When interactive mode is off, clicking a row will bring the user to a new screen to see details for that row.

When interactive mode is on, clicking a row will not change the screen. Instead, the user will stay on the dashboard, but all other views in the dashboard that depend on that row will be updated. In our example, when a user clicks on a row in the “Needs Review” section, the “Needs Review\_Detail” section will update to show information about the row. The “Open Maintenance Tickets Deck” section does not depend on the “Needs Review” section and will remain unchanged.

For mobile users, you have the option to display each view as a separate tab. If this is set to Off, then the views will be stacked vertically for mobile users. When using interactive mode, the user will need to select a row in one view, then change tabs to see the related views. Try both and see which you prefer using the preview on the right side of the screen.

Steps:

1. Go to UX > Views
2. Click “Add New View”
3. Set “View name” to “Review New Events”
4. Set “View type” to “dashboard”
5. Set “Position” to “left most”
6. Add the following “View entries”

**View entries**  
The other views to show inside the dashboard.

≡	Needs Review	↕	Tall	↕	🗑️
≡	Needs Review_Detail	↕	Tall	↕	🗑️
≡	Open Maintenance Tickets Deck	↕	Tall	↕	🗑️
+					

7. Set “Use tabs in mobile view” to “ON”
8. Set “Interactive mode” to “ON”
9. Set “Icon” to “globe”
10. Set “Show if” to `LOOKUP(USEREMAIL(), Users, Email, Role)="Site Lead"`

## 3.10 Create a dashboard for the Safety Board

Skills:

- [View Types](#)



The Safety Board is an example of a managerial user that wants to see overall statistics about their organization. We can create charts for everything the Safety Board needs to see. We can

also use “interactive mode” to connect the different views within the app.

To support the Safety Board role, the dashboard will display the following information:

- A list of reviewed reports, sortable by job site
- A bar chart showing the number of reports by job site
- A breakdown of events by priority
- A breakdown of reports by category

### 3.10.1 Create a table of reviewed reports

This view will show a list of reports where the [Needs Review] column is “False.” This will require another slice to capture the subset of reports.

The view will be a table with specific columns chosen to capture the information relevant to the Safety Board. When creating a table view you have the option to set which columns are included and the sequence. You can also limit the columns when creating a slice.

This view will also use a “Display Name.” This setting allows you to modify the way a view is displayed inside the app. This is useful when you need a descriptive name inside the editor, but want to simplify the user experience.

Steps:

1. Go to Data > Slices
2. Click “Add New Slice”
3. Set “Slice Name” to “Reviewed Events”
4. Set “Source Table” to “Reports”
5. Set “Row filter condition” to `[Needs Review] = False`
6. Go to UX > Views
7. Click “Add New View”
8. Set “View name” to “Reviewed Reports Table”
9. Set “For this data” to “Reviewed Events (slice)”
10. Set “View type” to “table”
11. Set “Position” to “ref”
12. Set the “View Options” as follows:



#### Sort by

Sort the rows by one or more columns.

☰	Priority	↕	Ascending	↕	🗑️
+					

#### Group by

Group rows by the values in one or more of their columns.

☰	Report Type	↕	Ascending	↕	🗑️
+					

#### Group aggregate

Display a numeric summary of the rows in each group.

NONE	↕
------	---

#### Column order

Display columns in a different order than they appear in the original data.

☰	Report Type	↕	🗑️
☰	Priority	↕	🗑️
☰	MT ID	↕	🗑️
☰	User Email	↕	🗑️
☰	Review Timestamp	↕	🗑️
☰	Description	↕	🗑️
+			

13. Set “Display name” to “Report Details”

### 3.10.2 Create a bar chart

Skills:

- [Charts: The Essentials](#)

A bar chart (also known as a histogram) is a great way to display information across different categories. In this case, we want to see the total number of reports for each Job Site.

When creating a bar chart, you must select the chart columns to display. This information will be displayed along the x-axis of the chart. In this case, we want to see each job site displayed on the x-axis, so we can set “Chart columns” to “Job Site ID.” Job Site ID is a reference to the Job Sites table. When you select a reference column, the column identified as “label” in the referenced table will be displayed.

On the y-axis, we want to see the total number of reports for each job site. So we will set “Group aggregate” to “COUNT.” This will display the number of rows matching each Job Site ID.

The chart will automatically adjust the scale to display more bars along the x-axis as more job sites are added.

Steps:

1. Add New View
2. Set "View name" to "Events by Job Site"
3. Set "For this data" to "Reviewed Events (slice)"
4. Set "View type" to "chart"
5. Set "Position" to "ref"
6. Set "Chart type" to "histogram"
7. Set "Group aggregate" to "COUNT"
8. Set "Chart columns" to "Job Site ID"

### 3.10.3 Create a breakdown of events by priority

An aggregate donut chart is one way of displaying the number of records that fall into distinct categories. In this case, we want to see the number of reports in each priority level. The chart will display the number of reports and give a quick visual reference of the distribution of reports across the categories.



**Pro Tip:** It is easy for donut charts to get overcrowded and become ineffective. Histograms are often a good alternative when comparing data across multiple categories.

Steps:

1. Add New View
2. Set "View name" to "Events by Priority"
3. Set "For this data" to "Reviewed Events (slice)"
4. Set "View type" to "chart"
5. Set "Position" to "ref"
6. Set "Chart type" to "aggregate donutchart"
7. Set View Options as follows:

Group aggregate  
How to aggregate data for aggregate charts.

COUNT

Chart columns  
Which columns to include.

Priority

Chart colors  
Use custom colors for this chart.

red

orange

green

Label type  
How to label a pie or donut chart.

Value

Show legend  
Include a chart legend.

ON

### 3.10.4 Create a breakdown of events by category

We also need a chart to show the distribution of incidents vs near misses.

Steps:

1. Add New View
2. Set "View name" to "Events by Category"
3. Set "For this data" to "Reviewed Events (slice)"
4. Set "View type" to "chart"
5. Set "Position" to "ref"
6. Set "Chart type" to "aggregate donutchart"

## 7. Set View Options as follows:

The screenshot shows the 'View Options' configuration for a chart in AppSheet. It is organized into five sections:

- Group aggregate:** A dropdown menu set to 'COUNT'.
- Chart columns:** A list of columns to include, currently showing 'Report Type'.
- Chart colors:** Two color selection rows. The first row is labeled 'red' and the second row is labeled 'orange'. Each row has a set of color swatches and a trash icon.
- Label type:** A dropdown menu set to 'Value'.
- Show legend:** A toggle switch set to 'ON'.

### 3.10.5 Create a dashboard

We can bring these views together into a single screen with a dashboard. AppSheet will set the initial size of each component view using the size set in the view definition. Choices are: Large, Wide, Tall, Small. In the next section, we will customize the size of each component view inside the dashboard.

Note: we will set a `show_if` condition on this dashboard so it is only visible to users with the “Safety Board” role. To change your role, click the hamburger button, click on “Users,” and edit your row. Later in this project, we will create buttons to streamline this process for demo purposes.

Steps:

1. Add New View
2. Set “View name” to “Safety Board Dashboard”
3. Set “View type” to “dashboard”
4. Set “Position” to “right”
5. Set “View Options” as follows:

### View entries

The other views to show inside the dashboard.

≡	Job Sites deck	↕	Large	↕	🗑
≡	Reviewed Reports Table	↕	Large	↕	🗑
≡	Events by Job Site	↕	Large	↕	🗑
≡	Events by Priority	↕	Large	↕	🗑
≡	Events by Category	↕	Large	↕	🗑
+					

### Use tabs in mobile view

When enabled, the mobile view will show tab controls to navigate between views, instead of a vertical scrolling list.



### Interactive mode

When enabled, tapping rows won't navigate to a different view. Instead, it will affect other views in the dashboard.



6. Set “Icon” to “users-cog”

7. Set “Show if” to `LOOKUP(USEREMAIL(), Users, Email, Role) = "Safety Board"`

## 3.11 Set opening view

Skills:

- [Choosing the Launch Page](#)
- [SWITCH\(\)](#)

You can set the view a user will be shown when they initially open the app. In this case, since we the app uses formulas to determine show views based on the user's role, it is useful to set a formula for the opening view.

The formula used below includes a SWITCH() statement. A switch statement is a great choice when you need to specify multiple possible outcomes based on what is found in a specified location. In this case, the “single location” is the user's role. Each line of the switch statement specifies a different outcome based on what the user's role is.

Steps:

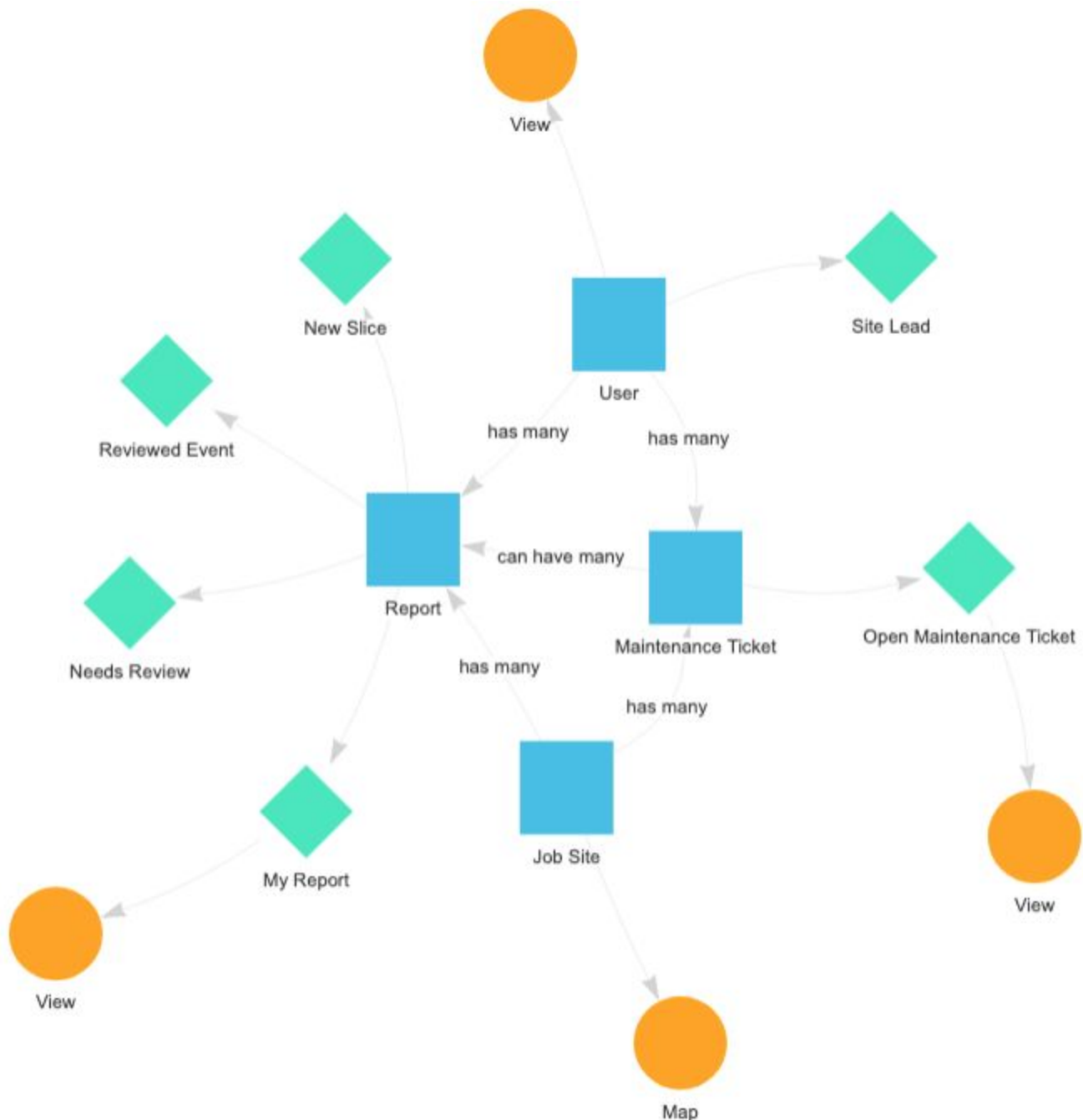
1. Go to UX > Options
2. Click the toggle button to the right of “Starting view” to turn on the formula
3. Click the gray bar to open the Expression Assistant
4. Enter the following expression:

```
SWITCH(LOOKUP(USEREMAIL(), Users, Email, Role),  
"Safety Board", "Safety Board Dashboard",
```

```
"Site Lead", "Review New Events",  
"Onsite Safety", "Report an Event",  
"Report an Event"  
)
```

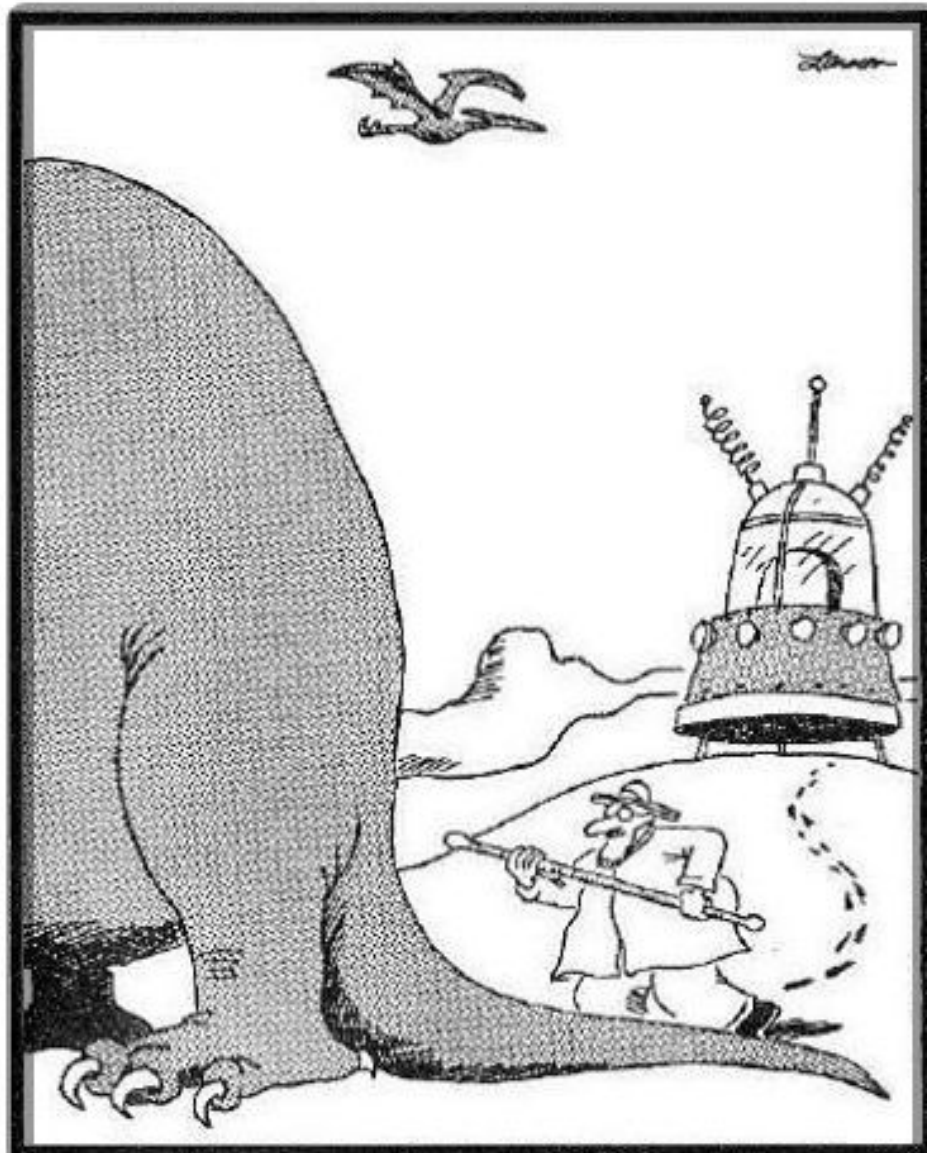
## 3.12 Wrap Up

Go to Info > Spec to see the relationships between the tables, slices, and views. We have added many new views in this section. In the next section, we will polish the views and add behaviors to create a streamlined user interface.



## Checkpoint #2

[Click here](#) to view the sample app for this checkpoint. Copy the app to your account by clicking “Copy and Customize.” After copying the app, add your email address to the Users table.



An instant later, both Professor Waxman and his time machine are obliterated, leaving the cold-blooded/warm-blooded dinosaur debate still unresolved.

## 4. Behaviors

Behaviors give us the ability to tie together the data and views inside the app. Buttons inside the app are all controlled under the Actions tab inside of Behavior. Automatic emails and text messages are controlled under the Workflow and Reports tabs inside of Behavior.

- [Actions](#) are a way to modify information and navigate to different parts of the app. The “+” button in the bottom right corner of a table is an example of an action. Multiple actions can be combined to perform many functions by clicking a single button
- [Workflows and Reports](#) are a way to send notifications in the form of emails, text messages, or push notifications. A Workflow is triggered when a row of data is added, deleted, or changed. A report is the same as a workflow, but instead of being triggered, a report is scheduled to occur on a daily, weekly, or monthly basis.

### 4.1 Create actions for the Site Lead









We want to make it easy for the Site Lead to review new reports. Common actions for the Site Lead after reviewing a report are to change the “Needs Review” column from True to False and add today’s date and time to the “Review Timestamp” column. Behaviors allow us to automate these common actions. In the next steps, we will create actions for the Site Lead to complete her review by simply clicking a button.

#### 4.1.1 Set “Needs Review” to False

The first action will set the “Needs Review” column to “False.”

Steps:

1. Go to Behavior > Actions
2. Click “Add New Action”
3. Input the following settings:

Action name A unique name for this action	Set Needs Review to False														
For a record of this table This action applies to rows of which table?	Reports <a href="#">see definition</a>														
Do this The type of action to perform	Data: set the values of some columns in this row														
Set these columns To the constant or expression values defined	<table><tr><td>≡</td><td>Needs Review</td><td>↕</td><td>=</td><td>False</td><td></td><td></td></tr><tr><td colspan="7">+</td></tr></table>	≡	Needs Review	↕	=	False			+						
≡	Needs Review	↕	=	False											
+															

4. Set “Action icon” to flag-checked










5. Set “Prominence” to “Do not display”

### 4.1.2 Set “Review Timestamp” to now

The second action we create will set the “Review Date” column to today.

Steps:

1. Go to Behavior > Actions
2. Click “Add New Action”
3. Input the following settings:

Action name <small>A unique name for this action</small>	Set Review Timestamp to now
For a record of this table <small>This action applies to rows of which table?</small>	Reports  <a href="#">see definition</a>
Do this <small>The type of action to perform</small>	Data: set the values of some columns in this row 
Set these columns <small>To the constant or expression values defined</small>	<div><div></div><div>Review Timestamp </div><div>=</div><div>NOW()</div><div></div><div></div></div> <div></div>

4. Set “Action icon” to flag-checkered
5. Set “Prominence” to “Do not display”

### 4.1.3 Create a grouped action

The third action will create a single button in the app that triggers both actions 1 and 2 above.

We will also add a formula to limit visibility for this action to user’s with the “Site Lead” role only.












**Pro Tip:** You need to decide where the action will appear in the app. You do this through the “For a record of this table” setting. Even if the action is modifying data of another table, the button will show up on views based off the table specified here.

For a record of this table <small>This action applies to rows of which table?</small>	Reports 
	<a href="#">see definition</a>

Steps:

1. Go to Behavior > Actions
2. Click “Add New Action”

### 3. Input the following settings:

Action name <small>A unique name for this action</small>	Grouped: Finish Review
For a record of this table <small>This action applies to rows of which table?</small>	Reports  <a href="#">see definition</a>
Do this <small>The type of action to perform</small>	Grouped: execute a sequence of actions 
Actions <small>The ordered list of actions to execute</small>	<div><div> Set Needs Review to False  </div><div> Set Review Timestamp to now  </div><div></div></div>

4. Set the “Display name” to “Complete Review”
5. “Action icon” to flag-checked
6. Set “Prominence” to “Display prominently”
7. Set the following formula for “Only if this condition is true”

```
LOOKUP(USEREMAIL(), Users, Email, Role)="Site Lead"
```

## 4.1.4 Testing

Skills:

- [Developing and Testing Your App](#)

Let’s take the new action for a test run. All apps are fully responsive out of the box. This means the layout of content is automatically adapted to fit screen sizes from mobile phones to tablets to desktops. In this step, we will preview the app on a full screen.

Steps:

1. In the preview screen, click “Preview other form factors”
2. Click “Open full screen”
3. Click the hamburger button in the top left
4. Click “Users”
5. Click the edit pencil for your name
6. Change your role to “Site Lead” and Save
7. Click “Review New Events” in the primary navigation bar
8. Click on an event in the “Needs Review” column to see its Details
9. Click on the “Complete Review” button in the Details panel

We can see the checkered flag shows up in both the “Needs Review” panel and the “Details” panel. This could be a good or bad thing. In this case, we want to encourage the Site Lead to only complete her review after she has seen the details.

#### 4.1.5 Remove the action bar from the Needs Review panel

We can change visibility for specific actions or remove the entire action bar from a deck view.

Steps:

1. Return to the editor
2. Go to UX > Views
3. Click “Needs Review”
4. Set “Show action bar” to “OFF”

#### 4.1.6 Add a confirmation

We can set a custom confirmation message for an action.

Steps:

1. Go to Behavior > Actions
2. Click “Grouped: Finish Review”
3. Set “Needs Confirmation” to “ON”
4. Set “Confirmation Message” to “Please confirm your review is complete.”

### 4.2 Create an automated email

Skills:

- [Workflow and Reports: The Essentials](#)
- [Sending Email from a Workflow Rule or Scheduled Report](#)



*“Shout it from the top of a mountain.”  
-Ron Burgundy*

Often times we want to notify users when a specific event happens. While we don’t have a mountain top, we do have Workflows and email notifications.



Workflows are triggered when a record in a table is added, deleted, or modified. They can be used to send a notification, trigger an action, or trigger a webhook. This section will focus on using a Workflow to notify the site lead when a new safety report is created on a job.

### 4.2.1 Create a Workflow

The email can be sent to a static (unchanging) address. It can also be sent to the email address associated with the record that triggered the workflow. For this workflow, we will setup both.



**Pro Tip:** A common need among app creators is to send an email when a new record is added and a specific column has a specific value. This can be set using a “Condition.” An example of the syntax commonly used in the “Condition” formula is `[column name] = "specific value"`

Steps:

1. Go to Behavior > Workflow
2. Click “Add New Workflow Rule”
3. Set “Rule name” to “Safety Report”
4. Set “Target data” to “Reports”
5. Set “Update event” to “ADDS\_ONLY”
6. Set “Reaction” to “Email”
7. Set “Workflow Action Name” to “Safety Report”
8. Set “To” as shown below, but replace [sally@example.com](mailto:sally@example.com) with your email address:

To

Send email to these email addresses.  
(Expressions that yield email addresses are supported.)

≡

=

≡

+

Note: the formula button is toggled on for the email address that comes from data in the spreadsheet.

### 4.2.2 Setup the Email Content

When defining fields in the Email Content section, you can mix static and dynamic content. To identify when a reference to dynamic content is being made, you must use angle brackets: `<<[column_name]>>`.

You have the option to specify the email body in two locations: 1) using the “Email Body,” or 2) using the “Email Body Template.” You can use a combination of static and dynamic text in either location. When using a template you have the additional ability to format the text of the email using the same tools in a word processing application (e.g. color, font, tabs).



**Pro Tip:** When using an email template, set the margins of the document to zero. This will avoid undesired spaces inside the email

Steps:

1. Set the “Email Subject” to:

```
[Safety Report] - <<[Job Site ID].[Name]>> by <<[User Email].[Full Name]>>
```

2. Set "Reply To" to "admin@bricklinholdings.com"
3. Set "From Display" to "Bricklin Safety"
4. Set "Attachment HTTP Content Type" to "PDF"
5. Set "Attachment Name" to "Safety Report"
6. Set "Attachment Archive" to "AttachAndDoNotArchive"
7. Set "Attachment Page Orientation" to "Portrait"
8. Set "Attachment Page Size" to "A4"
9. Save the app

### 4.2.3 Create an email body template

After saving, a button to create an email body template and attachment template should appear.

Steps:

1. Next to "Email Body Template" click "Create"
2. Wait a moment for the template to be created
3. Click "View." The template will list labels and dynamic connections for each column of the table.
4. Open this [example template](#)
5. Copy the example into your template
6. Update the link inside the template
  - a. Go to Users > Links
  - b. Open the "Browser Link" in a new tab
  - c. Navigate to the "Review New Events" view
  - d. Copy the url
  - e. Paste the url in for the "click here" link at the bottom of the template.

### 4.2.4 Create an email attachment template

Skills:

- [Template Start Expressions](#)

For this example, the email attachment template will be very similar to the email body. When creating apps, the attachment does not need to share any commonality with the email body. The rules to create the attachment are the same as used to create the email.

**Extra Credit:** A fixed size table of information can be displayed in an email or attachment. This is straightforward as will be shown in the attachment template. You can also create tables that dynamically resize to hold the amount of data provided. This is helpful when you need to show a list of data in a table and don't know what size the list is. AppSheet does this using a Template Start Expression. Full details on how to set up this kind of expression are available in the help article at the beginning of this section.

Steps:

1. Next to "Email Attachment Template" click "Create"
2. Wait a moment for the template to be created
3. Click "View." The template will list labels and dynamic connections for each column of the table.

4. Open this [example template](#)
5. Copy the example into your template
6. Click File > Page setup
7. Set all margins to zero

## 4.2.5 Troubleshooting Workflows

Skills:

- [Audit History Log](#)

The email will not send if there is an error in the email template. Errors in the template aren't always noticed by AppSheet until the workflow is triggered in the app. A common example of an error is a mismatch between the name of the column specified in the template and the name of the column in the data source.

To confirm if the workflow sent correctly, review the audit history log. The log has a full list of all operations performed in the app. Note: it can take a couple of minutes for an event to show up in the log.



**Pro tip:** Any time it seems like there might be an error or strange behavior, start by checking the audit log to see if there is an error message.

Steps:

1. Go to Manage > Monitor > Audit History
2. Click "Get audit history records"
3. Click the binoculars in the "Details" column to see more information
  - a. The binoculars are red for operations with errors

## 4.3 Enable offline access

Skills:

- [Offline/Sync](#)



The Onsite Safety may not always have an internet connection when he needs to make a safety report. By enabling offline access, users are able to open the app and make changes without an internet connection. Changes made while offline will be stored on the device and synced to the datasource when the user comes back online.

Steps:

1. Go to Behavior > Offline/Sync
2. Set "The app can start when offline" to "ON"
3. Set "Store content for offline use" to "ON"

## 4.4 Sync settings

Skills:

- [Sync: Between the App and the Backend](#)



It is best practice to sync the app on start. This ensures the data accessed in the app is up to date.

Turning on “Delayed sync” and “Automatic updates” will streamline the app behavior so that changes will be synced to the datasource in the background without affecting the user’s experience.

Steps:

1. Go to Behavior > Offline/Sync
2. Set the following options:

^ SYNC: APP TO CLOUD

**Sync on start**  
Sync data every time the app starts to make sure the user has the latest data. ☒

**Delayed sync**  
Only sync when the user presses the sync button, instead of syncing after every change. ☒

**Automatic updates**  
Automatically send changes the user makes as they occur. Automatically retrieves changes made by other users about every 30 minutes. ☒

## 4.5 Wrap up

We have completed the primary structure of the app. The roles we setup will apply automatically based on the email address and the assigned role of the logged in user. The next sections will show how to manage and scale the app.

## 5. Switching roles



This app demonstrates a customized experience based on the user's role. In a production app, the user's role does not change frequently, and we could lock down the edit capability for the Users table to only specific administrators.

In the demo version of the app, we need to switch roles in order to show the customized experience from each perspective. The app you will copy at the next checkpoint includes additional functionality to make it very easy for the logged in user to change their role.

The high-level features used to create this functionality have been covered during this course. The detailed steps are not listed, but after copying the next checkpoint, the curious reader can look in the following locations to see the changes:

- A new data table called "Demo Modes"
- A new view called "Demo Mode"
- Several new actions for the "Demo Modes" table and the "Users" table



## Checkpoint #3

[Click here](#) to view the sample app for this checkpoint. Copy the app to your account by clicking "Copy and Customize." After copying the app, add your email address to the Users table.



## 6. Customizing User Experience

This section describes tools to enhance the aesthetic experience of applications.

### 6.1 Format rules

Skills:

- [Format Rules](#)

Formulas can be applied to format rules to create visual distinctions between categories in the app. The following sections establish a color scheme for critical, normal, and low priority reports throughout the app.



**Pro tip:** As an app matures, the number of format rules often increases. It is helpful to choose a naming convention that makes it easy to identify the purpose of a format rule by its title.

#### 6.1.1 Critical red

Steps:

1. Go to UX > Format Rules
2. Click “Add New Format Rule”
3. Set “Rule name” to “Critical priority red”
4. Set “For this data” to “Reports”
5. Set “If this condition is true” to `[Priority] = "Critical"`
6. Under “Format these columns and actions” select “Priority”
7. Set “Text color” to “red”
8. Set “Bold” to “ON”

#### 6.1.2 Normal yellow

Steps:

9. Click “Add New Format Rule”
10. Set “Rule name” to “Normal priority yellow”
11. Set “For this data” to “Reports”
12. Set “If this condition is true” to `[Priority] = "Normal"`
13. Under “Format these columns and actions” select “Priority”
14. Set “Text color” to “yellow”

#### 6.1.3 Low green

Steps:

15. Click “Add New Format Rule”
16. Set “Rule name” to “Low priority green”
17. Set “For this data” to “Reports”
18. Set “If this condition is true” to `[Priority] = "Low"`
19. Under “Format these columns and actions” select “Priority”

20. Set “Text color” to “green”



**Pro tip:** You can select colors beyond the automatically generated pallet by inputting the hexadecimal code associated with your desired color into the field labeled “custom”.

## 6.2 Branding

Skills:

- [Branding](#)

You can customize the theme of an app with branding. Check out the settings by going to UX > Brand.

Two popular color themes to try are “Dark - Orange” and “AppSheet Official”

AppSheet provides several icons to choose from. Most app creators make a custom logo and upload it to the app. To upload a custom logo, first save it on one of the cloud storage locations connected to the app. This is typically the storage location connected to the account used for authentication by the app owner. Then point AppSheet to the location of the file.

Launch images can be customized in the same way as the app logo. To add some flair to the app, try specifying a gif for the launch image. For example:

<https://media.giphy.com/media/3o7TKzhPqGvUDm3pba/giphy.gif>

## 6.3 Localization

Skills:

- [Localization](#)

Localization enables you to change the name of buttons inside the app. For example, in the form filled out by the Onsite Safety, you can change the “Save” button to “Submit.”

Steps:

1. Go to UX > Localize
2. Toggle the formula for “Save”
3. Enter the following formula:

```
IF(CONTEXT(View)="Report an Event", "Submit", "Save")
```

By using the CONTEXT() function inside an IF() statement, the change only applies to the specified view; other views continue to show “Save.”

## 7. User Management

When adding users to an app, there are two distinct concepts to consider:

- 1) Authentication - The purpose is to confirm a user is who they claim to be. Specifically they are the correct human for the email address or username used to log in.
- 2) Permissions - The access allowed for the user after they are signed in. This dictates what the person is allowed to see and do inside the app.

In AppSheet, authentication can be completed through any of the following authentication providers:

- Google
- Dropbox
- Box
- Office365
- Smartsheet
- Salesforce

Permissions are established in many locations throughout the app definition. Every `show_if` statement defined in Section 3 is an example of permissions.

### 7.1 Adding users

Skills:

- [Require User Sign in](#)
- [Granting Application Access](#)
- [USERROLE\(\)](#)

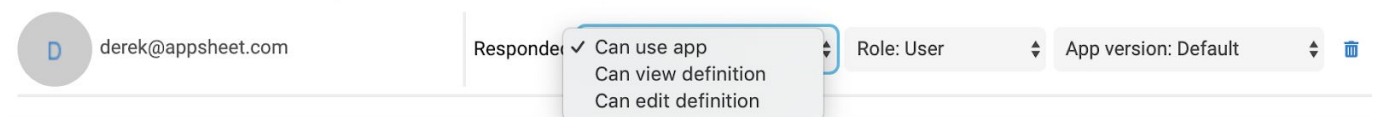
The app creator and co-authors can add users to the app. Users can be added individually or by an entire domain.

When adding users, you have the option to specify a specific authentication provider or allow the users to authenticate using any provider supported by AppSheet.

Steps:

1. Go to Users > Users
2. Type the new user's email address
3. Confirm human status (reCAPTCHA)
4. Click to add users and send invite

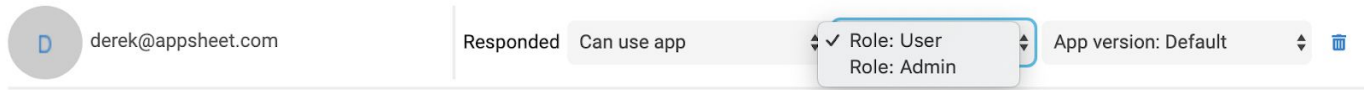
In the whitelist, there are three options under “responded”



- Can use app: unable to see or edit the app definition

- Can view definition: can open the app definition in their own AppSheet account, but are unable to modify any settings.
- Can edit definition: can open and modify the definition in their own AppSheet account. Adding a user as a co-author and setting this to “Can edit definition” have the same effect.

Role has two options: User and Admin. These have no implicit impact on the app or on a user’s permission. However, this setting can be accessed inside the app definition via USERROLE() to define permissions.



App version has three options. These will be covered in a later section devoted to version control.

## 7.2 Domain Authentication

Skills:

- [Domain Integration: The Essentials](#)

AppSheet integrates with many domain authentication tools including AWS Cognito, Azure AD, and Okta. If your app users are all part of a corporate domain, you can use domain security groups to control access to an AppSheet app. To get more information about domain authentication, please contact [sales@appsheet.com](mailto:sales@appsheet.com).

## 7.3 Version control

Skills:

- [Version Control](#)
- [Maintaining a Stable App Version](#)
- [App Upgrades](#)

When changes are made and saved to an app, they will be visible to users the next time the user syncs the app. This ability to rapidly deploy changes can be advantageous for small tweaks. However, it is often necessary to test changes before they are deployed to the entire user based.

Version control allows the app creator to set a stable version of the app. After making changes, you will have two versions: 1) the stable version, and 2) the latest version. You have the ability to control which version each user sees.

A typical deployment strategy is:

1. Set a stable version
2. Make necessary changes
3. Identify test users
4. Go to Users > Users and set all test users to see the latest version
5. Complete testing

6. Advance stable version to latest version



**Pro tip:** Setting a stable version prevents production users from seeing incremental changes. It does **not** prevent production users from seeing changes to the data source.

## 7.5 Transferring an App

Skills:

- [Transfer an app between users or accounts](#)

In addition to adding co-authors, you can transfer an app to another account. This process is detailed in the linked reference under “Skills” above.

## 7.6 Create a public sample app

Skills:

- [Public sample apps](#)

The apps provided at checkpoints during this course are examples of public samples. Everyone with an AppSheet account has the ability to add apps to their public portfolio. The process is detailed in the linked reference under “Skills” above.

AppSheet offers an extensive library of [sample apps](#). Sample apps can be copied to your account and used as an example or as a starting point for app creation. They are free and there is no limit to how many you can copy.

# 8. Scalability

AppSheet is designed to scale. There are many tools available to ensure large datasets can be worked with effectively. Three of the most powerful tools are described in the following sections.

## 8.1 Security Filters

Skills:

- [Scaling using Security Filters](#)

In AppSheet you can apply a security filter to limit the data that is sent to the user's device. When your app is based on a SQL database, the security filter is converted to a SQL query. Only records that match the filter will be read from the database.

Regardless of how many rows of data exist in the database, the app performance only depends on the amount of data that passes the security filter.

Security filters can improve performance with spreadsheets as well. However, the AppSheet server reads the full spreadsheet before applying the filter. So the performance benefit is less than what you get with a SQL database.

## 8.2 User Settings

Skills:

- [User Settings](#)
- [Advanced Techniques: Horizontal Scaling](#)

An app creator can enable user settings in any app. The user settings are stored locally on a user's device and can be accessed by formulas in the app settings.

Consider an application that connects to a database with over 1 million rows of data. To maintain a reasonable sync time the data sent to the device must be carefully chosen. Turning on the user settings table enables a user to choose the criteria that will be applied to the security filter.

Steps:

1. Go to Data > User Settings
2. Click on \_Per User Settings
3. Click the edit pencil next to "Options Heading"
4. Set "Show" to "ON"
5. Set "Category" to "Page\_Header"
6. Set "Content" to "Select the Job Sites to see in the app"
7. Click "Done"
8. Click the edit pencil on the next row down
9. Set "Column Name" to "Job Sites"
10. Set "Show" to "ON"
11. Set "Type" to "EnumList"
12. Set "Allow other values" to "ON"
13. Set "Auto-complete other values" to "ON"
14. Set "Base type" to "Text"
15. Set "Input mode" to "Buttons"
16. Set "Valid if" to `Job_Sites[Name]`
17. Click "Done"
18. Save the app

Go check the user settings to make sure they look right

Steps:

1. In the app preview, click the hamburger menu
2. Click "Settings"
3. Confirm there is a list of buttons representing all the job sites

Add a slice based on the User Settings

Steps:

1. Data > Slices
2. Click “Add New Slice”
3. Set “Slice Name” to “Selected Job Sites”
4. Set “Source Table” to “Job Sites”
5. Set “Row filter condition” to `IN([Name], USERSETTINGS("Job Sites"))`
6. Edit views that are currently based on the the Job Sites table to use this slice

Add a security filter to the Reports table based on the Job Sites slice.

Steps:

1. Go to Security > Security Filters
2. Click on “Reports”
3. In the existing formula, replace “true” with `IN([Job Site ID], Selected Job Sites[Job Site ID])`

## 8.3 Partitioning

Skills:

- [Scaling using Data Partitions](#)

You can hold the data for a large-scale app in a spreadsheet. Partitioning enables you to break the data up across multiple tabs or multiple files. When setup correctly, this allows AppSheet to only read the relevant partition which reduces sync time. [Here's an article about scaling with partitioning.](#)

The downside occurs if you need to modify the structure of the table. For example, if you add a column to a partition, then you need to add the same column to every other partition.

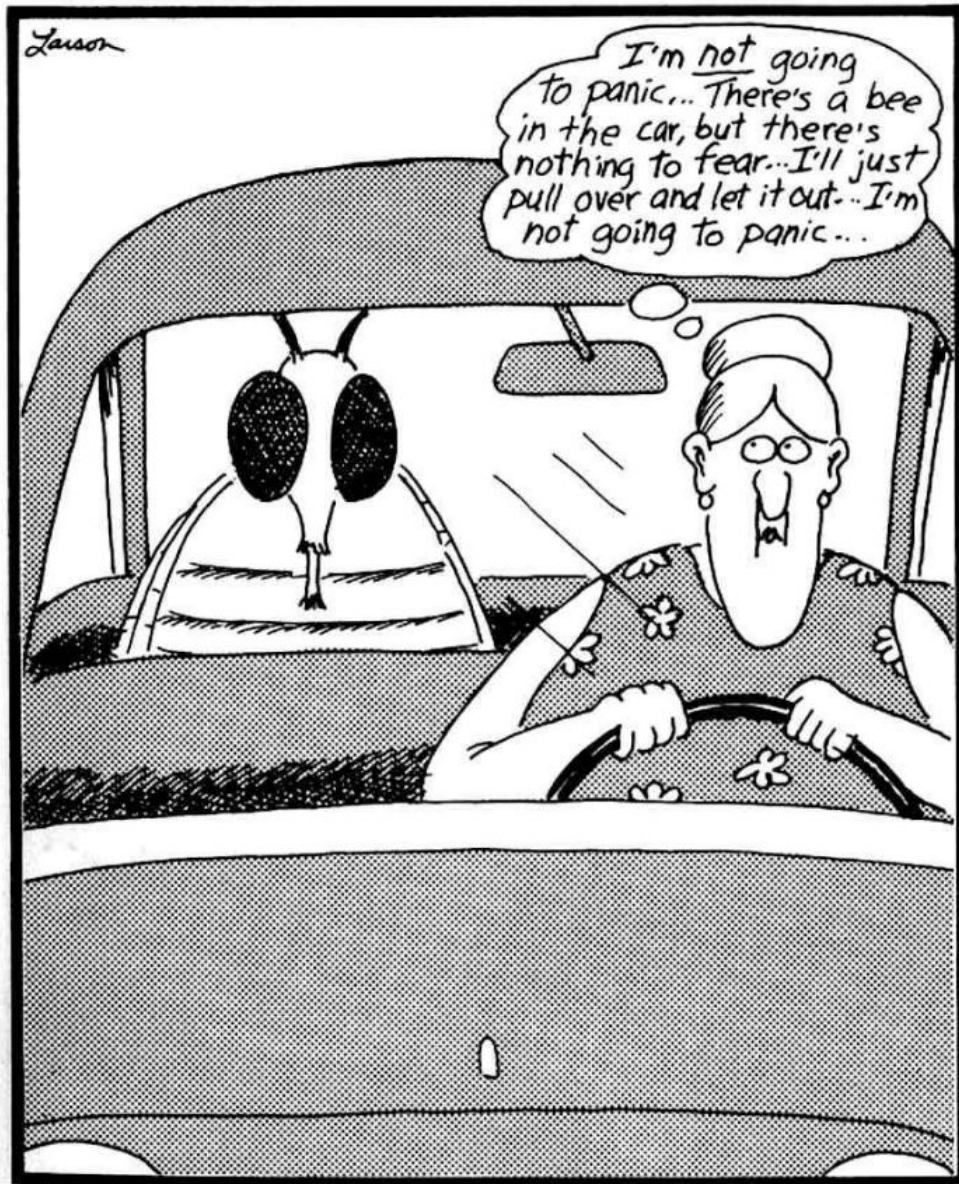
To get the full efficiency of partitioning, you want to set it up so a user only needs to access a single partition when interacting with the app. For example, you could partition a list of meeting rooms by building. When a user logs in, they only see the meeting rooms that correspond to their building.

In Smartsheet, you are limited to 5000 rows per table. If your dataset requires more than 5000 rows, you can use partitioning to automatically write new data to separate tables based on a formula. When partitioning data, you can separate the data in whatever way works best for you. If you create a formula based on Customer, you will end up with a database structure similar to what you already have. The difference is in how you work with the data inside AppSheet. By partitioning, AppSheet will treat the collection as a single table. Generally, this makes it much easier to work with the data, especially as the quantity of data scales. For additional information about partitioning, please check out [this article](#).



## Checkpoint #4

[Click here](#) to view the sample app for this checkpoint. Copy the app to your account by clicking "Copy and Customize." After copying the app, add your email address to the Users table. Congratulations, you have completed the Product Training Course!



## 9. Resources

### AppSheet Community

AppSheet has a vibrant user community. If you do not already have an account, please take a minute to create one. <https://community.appsheet.com/>

### Resource Portal

1. Open [AppSheet Suite](#)
2. Bookmark AppSheet Suite
3. Go to Business > Resource Portal

### Demo script

1. In the Resource Portal, go to the “Sales Content” view.
2. Under “Sales Tag”, click “Demo Tool”
3. Click “Construction Safety Inspection Demo Script”
4. Click the “Link” action to open the script

## Revision Record

Version	Date Completed	Description of Change	Point of Contact
1.0	10/3/2019	- Created pilot version of course content.	Derek Covey
1.1	10/20/2019	- Incorporated comments from pilot session. - Added certification requirements - Updated show_if formula for Onsite Safety form by removing the ability for people with no role to see the view. - Added “big picture” explanations throughout document.	Derek Covey