



# AppSheet Automation

*GA Release - Primer*

**Version 1.4**

Updated: April 13, 2021

## **Disclaimer**

This primer describes the basic capabilities of the AppSheet Automation GA release.

## Version History

Version 1.0	Initial version. Introduction, Contact Sync use case
Version 1.1	Added a new Sheets eventing use case - Conference leads appointments generator.
Version 1.2	Added new concepts and use cases for Employee onboarding and Document processing.
Version 1.3	Added schema image for configuring Employee table, fixed some broken links
Version 1.4	Updated product images and added a few new comments

[Version History](#)

[Introducing AppSheet Automation](#)

[Key AppSheet concepts](#)

[Automation](#)

[Bot](#)

[Process](#)

[Step](#)

[Run a task](#)

[Branch on a condition](#)

[Wait for a condition](#)

[Call a process](#)

[Return values](#)

[Task](#)

[Event](#)

[Document Type](#)

[Fixed Schema Table](#)

[Folder Data Source](#)

[Extraction Confidence](#)

[Before you begin](#)

[Employee Onboarding](#)

[Creating the bot](#)

[Configuring the event](#)

[Configuring the process](#)

[Testing your automation](#)

[Automation Monitoring](#)

[Contact Sync](#)

[Creating the bot](#)

[Configuring the event](#)

[Configuring the process](#)

[Testing your automation](#)

[Conference leads appointment generator](#)

[Configuring the Appointment Confirmation Process](#)

[Creating the bot](#)

[Configuring the event](#)

[Configuring the main process](#)

[Testing your automation](#)

[Invoice Extraction Automation](#)

[Creating Invoice Table from Folder Data Source](#)

[File Processing](#)

[Adding Application Views](#)

[Low Confidence Extraction Handling](#)

[Failure Handling](#)

[Folder Contents as a Table](#)

[Creating an App](#)

[Creating a Table from Folder Data Source](#)

[Creating a useful view](#)

[Creating Slices to filter data](#)

[Linking Files to other Data](#)

[Appendix](#)

[Configuring Employees entity from Google Sheets datasource](#)

[Configuring the AppSheet Events add-on for Google Sheets](#)

[Configuring Company Contacts entity from Google Sheets datasource](#)

[Configuring Contact entity from Salesforce datasource](#)

[Configuring Leads entity from Google Sheets datasource](#)

[Configuring appointments entity from Google Sheets datasource](#)

[Configuring an email template for ops notification](#)

[Document processing states and status codes](#)

[Potential status codes and values](#)

[Why is there a separate currency code column?](#)

[Where can I store my files for Document processing?](#)

[What does 'collection of files' mean?](#)

## Introducing AppSheet Automation

AppSheet is an intent-aware, no-code application development platform. AppSheet empowers businesses to create applications faster and with less money than a traditional, code-based approach. With AppSheet's true no-code platform, a business user does not need to know how to code in order to create a piece of software. The platform handles the technical heavy-lifting, allowing the business user to focus on the creation of effective apps.

This GA release introduces AppSheet Automation, a robust extension of AppSheet's platform offering new, integrated workflow automation capabilities.

Appsheet Automation:

- Unifies AppSheet's intent-aware, no code-platform with process/workflow automation as a first class feature.
- Offers Intelligent Process Authoring & Runtime with rich connectivity, allowing users to author and execute their business processes using entities.
- Introduces new concepts (including Bots and Process) to enrich the core AppSheet platform.
- Addresses the long tail of human centric processes, document based workflows and application integration use cases.

## Key AppSheet concepts

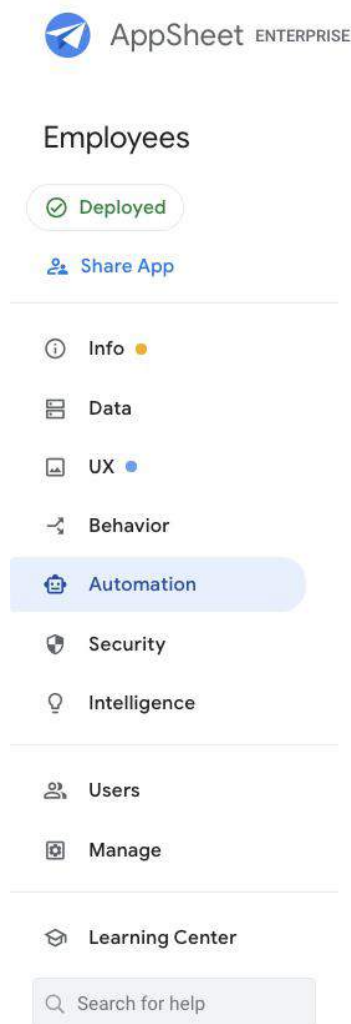
This section describes some of the basic concepts introduced with the GA release of AppSheet Automation.

Please visit the help [site](#) for additional product documentation.

## Automation

AppSheet *Automation* enables you to automate common business processes and document based workflows. You can access the new automation features from the new **Automation** tab, as shown in the navigation image below.

The platform encourages reusability, allowing you to build components once and use them in various automations, saving time and accelerating development. Events, processes and tasks are completely re-usable. That means you can reuse tasks within new processes, reuse processes within new bots, reuse events within new bots etc.



## Bot

*Bots* are used to create automations by combining events with processes. Bots can be configured to trigger a process on detection of a specified event or according to a predetermined schedule. Once enabled, bots run in the background to listen for event triggers or trigger processes on a schedule. To terminate the process automation, you can disable the bot.

In the example shown below, the bot is configured to trigger the “Onboard new employees” process anytime a new “Employee” record is added to a Google Sheet. This bot continues to listen for the new employee creation event and trigger the associated process whenever the event occurs until it is disabled.



**Bots**   Events   Processes   Tasks

### Employee onboarding

EVENT: "A new employees record is created" >  
PROCESS: "Onboard new employees"

When this EVENT occurs:

✂ A new employees record is created  
employees

Run this PROCESS: "Onboard new employees" ▾

+

Send welcome email to employee

+

END

Go to: [Process](#)

Appearance ▾

Documentation ▾

## Process

A *process* represents a typical business process. For example, an “Onboard new employees” is a business process. A process contains the following elements:

- A sequence of steps (control steps or tasks)
- An input
- An output (this is optional)

For example, in the screenshot below, you can see a sample “Onboard new employees” process.

The screenshot shows the AppSheet automation editor interface. At the top, the title is "Onboard new employees" with "Copy" and "Delete" buttons. Below the title, it says "steps: Send welcome email to employee". The main area is titled "Entity" and features a dropdown menu currently set to "employees". Below the dropdown is the question "What type of entity does this process apply to?". A blue plus icon in a circle is positioned above a white box containing the text "Send welcome email to employee". A dashed arrow points from the bottom of this box to a blue button labeled "+ Add a step". Below the button is the word "END". At the bottom of the editor, there are two expandable sections: "Appearance" and "Documentation", each with a downward-pointing chevron icon.

## Step

A *step* is a basic element in a process that defines a task to be carried. A process can have several steps.

Steps can be of several types:

- Run a task
- Branch on a condition
- Wait for a condition

- Call a process
- Return values

### Run a task

This step type allows you to run one of the following task types - “Send an email”, “Send a notification”, “Send an SMS”, “Change data”, “Call a webhook” or “Create a new file”.

In our example, the *step* is “Update contact list.” The step runs a task (of type “Change data”) against the data provided to the process on the *input*. In this case, the input is the new contact record.

Step name

Run a task  Branch on a condition  Wait for a condition  Call a process

Return values

Task to run


AMP (Preview)  Send an email  Send a notification  Send an SMS


Change data  HTTP Call a webhook  Create a new file


### *Branch on a condition*


This step type allows you to implement conditional logic in your process; for example, `if <condition> - then <do some step> - else <do some other step>`. The condition value can be any valid expression.


Step name

 Run a task

 **Branch on a condition**

 Wait for a condition

 Call a process


 Return values


Condition to check


### *Wait for a condition*


This step type allows the process to pause execution until the condition evaluates to true, at which point the process resumes execution.


Step name

 Run a task


 Branch on a condition

 Wait for a condition

 Call a process

 Return values

**Wait until this condition is true**

= [Status] = "Approved" 

### *Call a process*

This step type allows you to invoke another process within the same application. You can pass data to the process being invoked using literal values or expressions.

Step name

Run a task    Branch on a condition    Wait for a condition    **Call a process**

Return values

**Process name**

**Add or lookup process input?**

A new record will be created using the data you supply. If another record exists with the same primary key, that record will be updated and used.

**Process inputs**


⋮	Date	= TODAY()	⚙	🗑
⋮	Request Re...	= [Reason]	⚙	🗑
⋮	Issue Resol...	= "false"	⚙	🗑


[Add](#)


## Return values


This step type allows you to return specific values from within your process and access those values during later steps. For example, you may want to have a parent process invoke a sub-process, and use the *return value* of the sub-process to determine the next steps to be performed.


Step name

  
 Run a task

  
 Branch on a condition

  
 Wait for a condition

  
 Call a process

  
 Return values

**Return these values**

⋮	ApprovalAmount	= [Amount]	⚙️	🗑️
⋮	ApprovalReason	= [Reason]	⚙️	🗑️

Add

In the example above, a parent process has a step called “Send for approval” that returns an approval decision using the *Return values* step type shown above. Once the return value is set, it is accessible within the parent process through the expression “[Send for approval].[ApprovalStatus].”

## Task

As noted above, a step defines a *task* to be carried out in the process. There are six types of tasks:



- [Send an email](#)
- [Send a notification](#)
- [Send an SMS](#)
- [Change data](#)
- [Call a webhook](#)
- [Create a new file](#)

Our example shows configuration of a “Send an email” task type:

The screenshot shows the configuration page for a task named "Send welcome email". At the top, there are navigation tabs for "Bots", "Events", "Processes", and "Tasks". The task is titled "Send welcome email" with "Copy" and "Delete" buttons. Below the title, the "Task category" is "Send an email", highlighted among other options like "AMP (Preview)", "Send a notification", "Send an SMS", "Change data", "Call a webhook", and "Create a new file".

The configuration fields include:

- Task name:** "Send welcome email" (Unique name for this task)
- Table name:** "employees" (What entity table does this task work against? Includes a "View Definition" link)
- Via channel:** "System Default" (The messaging channel to use)
- To:** "someone@demo.com" (Send email to these email addresses. Includes an "Add" button)
- Use default content?:** A toggle switch is currently turned off. (If enabled, the task constructs a sensible default message. If disabled, you should define the content explicitly.)
- Email Content:** A text area for defining the email content.

## Event

An *event* represents a change to an entity (table). Each event has the following attributes:

- **Event type.** This attribute indicates the kind of change that triggers the event.
- **Condition (optional).** If specified, this condition is checked before firing the Action.
- **Target data.** Indicates which schema would trigger the event, if changed.
- **Update event.** Indicates the type of schema change that would trigger the event (i.e. additions, updates, deletions, etc.).

In our example, as shown in the image below, the New Event is a “Data Change” event only triggered by additions to the “Employee” table.

Bots **Events** Processes Tasks

employees

**A new employees record is created** Copy Delete

Data Change

**Event Type**  
What type of change triggers this event?

Data Change | Schedule

**Table**  
Data changes to which table should trigger this event?

employees

**Condition**  
Optional condition that is checked before firing the Action

=

**Data change type**  
What type of data changes should trigger this event?

ADDS\_ONLY

**Bypass Security Filters?**  
Execute this event and the Bots that it triggers as though there are no security filters on the data sources.

Appearance

Documentation

Below is an example of a “Schedule” event type. In the case below, where **ForEachRowInTable** is not enabled, you can select any event except “Data change” as a valid step type in the process. You may want to use this option if your process doesn’t rely on data in any table.

Bots **Events** Processes Tasks

Other

### Onboard new employees every Monday morning

Copy
Delete
⌵

Scheduled

**Event Type**  
What type of change triggers this event?

Data Change

Schedule

**ForEachRowInTable**  
Is this trigger for each row in the given table?

**Schedule**  
What schedule will trigger this workflow rule?

Weekly

⌵

**Day of the week**

× Sun
✓ Mon
× Tue
× Wed
× Thu
× Fri
× Sat

**Time**

9:00 a|m

**Time zone**  
What time zone should the schedule use?

UTC

⌵

**Bypass Security Filters?**  
Execute this event and the Bots that it triggers as though there are no security filters on the data sources.

Appearance ⌵

Documentation ⌵

However, if the **ForEachRowInTable** option is enabled, the option to specify a filter condition appears. Use this option if you want to run a process for each row of data in the table.

Bots **Events** Processes Tasks

Other

Onboard new employees every Monday morning

Copy
Delete
⌵

Scheduled

**Event Type**  
What type of change triggers this event?

Data Change

Schedule

**ForEachRowInTable**  
Is this trigger for each row in the given table?

**Table**  
What table to query?

employees ▾

[View Definition](#)

**Filter Condition**  
Filter condition to specify which rows of the table?

=

[Department] = "HR"

⌵

**Schedule**  
What schedule will trigger this workflow rule?

Weekly ▾

Day of the week

× Sun

✓ Mon

× Tue

× Wed

× Thu

× Fri

× Sat

Time

9:00 am

**Time zone**  
What time zone should the schedule use?

UTC ▾

**Bypass Security Filters?**  
Execute this event and the Bots that it triggers as though there are no security filters on the data sources.

## Document Type

*Document types* are available in document-based workflows. AppSheet automation classifies document types based on the document's content, regardless of mime type or extension type. For example, *document types* include: receipt, invoice, resume, work order, W9, etc. The

*document type* does not refer to mime types (such as spreadsheet, presentation, or photo) or extensions (such as `.pdf`, `.docx`, `.png`). A *document type* can be represented in multiple formats depending on how it is received - exported as a PDF, scanned as a PDF, photographed as a `.png`, or otherwise. In any of those cases the *document type* remains "invoice" because of its content.

## Fixed Schema Table

For document extraction to work properly, the table schemas for document extraction must match the content fields to be extracted. As a result, *fixed schema tables* created for the purpose of document extraction are immutable and can't be altered by application editors. A user may still edit the UX of the table views to suit their needs but must ensure that required fields are present for creation of entries into these tables.

## Folder Data Source

For document-based workflows that utilize *document types*, a new data source type is also available to support folders as data sources. Google Drive is the first system supported, with the intention of adding more *folder data source* systems in the future.

Folders are fixed schema tables, where the attributes (table columns) are the properties of files (such as name, size, last modified at, etc.) and the entries (table rows) are the files themselves. In the AppSheet Automation GA release, each folder should contain files of the same *document type*, although the files can vary in mime and extension type.

**Note:** *Supported extensions in the GA Release: `.tiff`, `.gif`, `.pdf`, `.jpg` and `.png`. Folders containing mixed document types can be used for document viewing and upload inside applications, but can not be automatically extracted.*

For the GA Release, *folder data sources* can be either **implicitly** used as part of configuring document extraction, or **explicitly** used as data sources for building views to show documents in the application. For example, a "user manual view" in an application can explicitly use a Google Drive *folder data source* containing documents that can be viewed, updated, added, or deleted within an application.

## Extraction Confidence

Some source documents may have defects that result in extraction accuracy that is less than 100%. For example, some documents may be based on scanned sources, utilize complex images, use unusual fonts, or have spelling errors or omissions. As a result, the extraction service includes an *extraction confidence* score that is then used by the automation process

to determine if the extracted data should be manually reviewed. If the *extraction confidence* score falls below the configured threshold, an exception flow is triggered to involve manual review by a human.

**Note:** In the GA Release, the confidence threshold cannot be adjusted by application editors.

## Before you begin

Before testing the AppSheet Automation GA features, make sure you have completed the following steps:

- Created an account on [appsheet.com](https://appsheet.com).
- Ensure that you have access to the Automation release (Be able to see Automation logo in the left nav bar).
- You have created an app.

Now you are ready to explore AppSheet Automation!

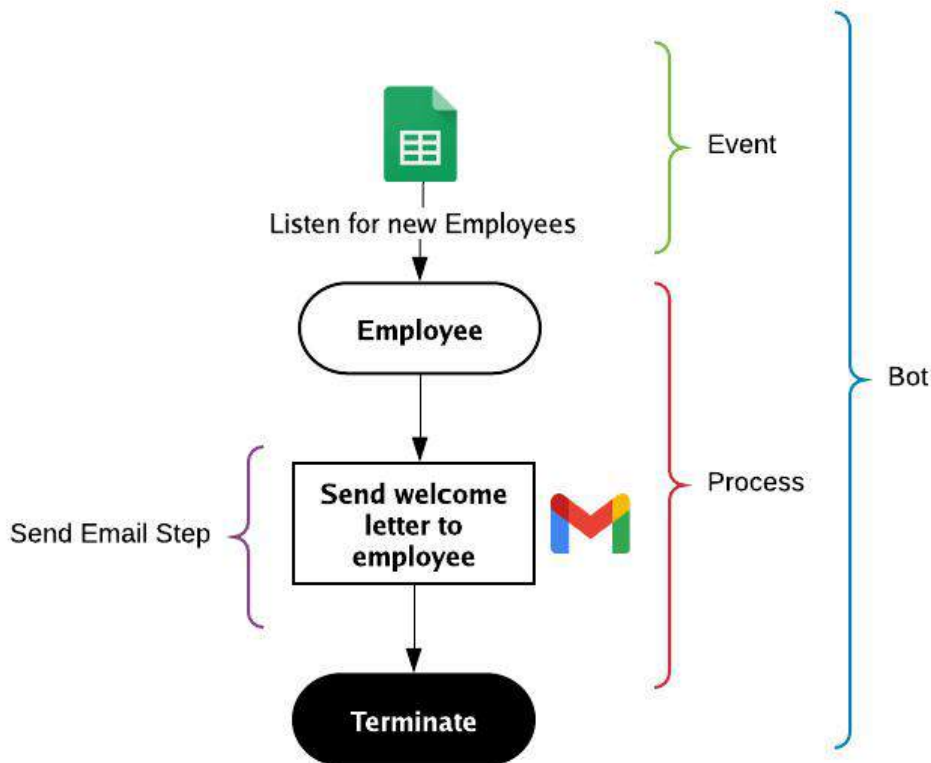
## Employee Onboarding

This brief tutorial shows you how to set up an automated process to onboard new employees. When a new employee is added in a Google Sheet, AppSheet automatically emails a welcome letter to that employee.

Implementing Employee Onboarding with AppSheet Automation involves the following steps:

1. [Creating the Bot](#)
2. [Configuring the Event](#)
3. [Configuring the Process](#) (steps and actions)

The following figure describes the high level components:



### Before you begin this tutorial:

- Make sure you can log into the AppSheet Editor UI and access the Sample Automation App you created as a prerequisite in [Before you begin](#).
- Follow the steps in the Appendix to configure a [Employee Entity](#) from Google Sheets data source and install the AppSheet [Events Add-on](#).

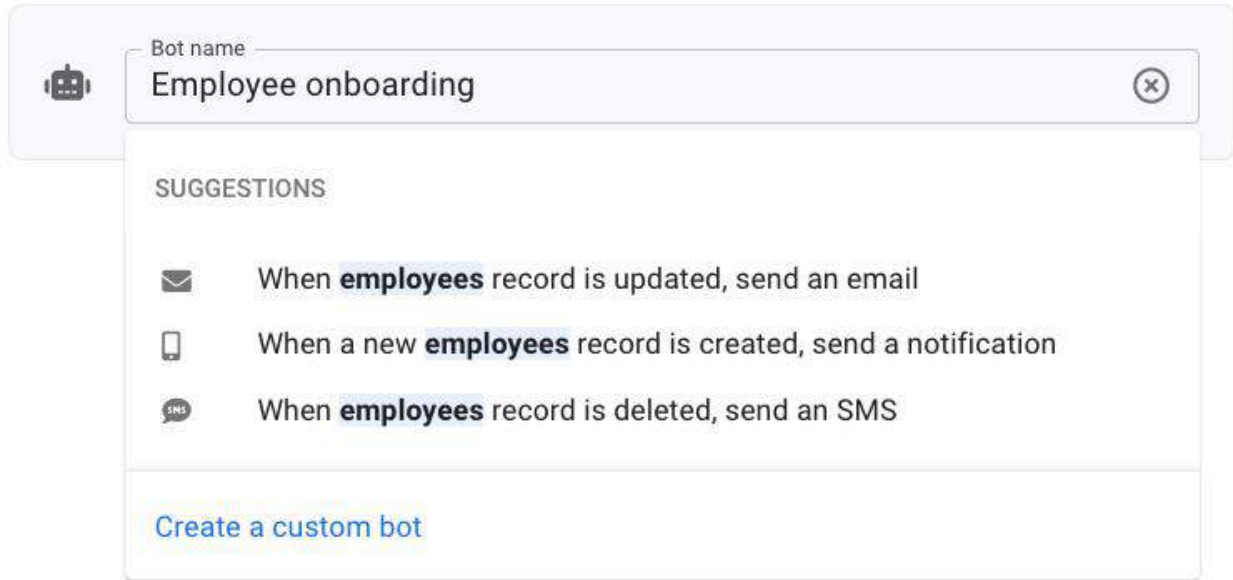
## Creating the bot

Follow the steps below to create a new bot:

1. From the AppSheet UI, navigate to the **Bots** tab and click **New Bot**.
2. Type `Employee Onboarding` into the dialog and suggested actions appear, as shown in the image below.

*(You can select one of these suggestions - in which case a fully configured bot is ready for you - for now we will create a custom bot)*

3. Select **Create a custom bot** to create the bot.



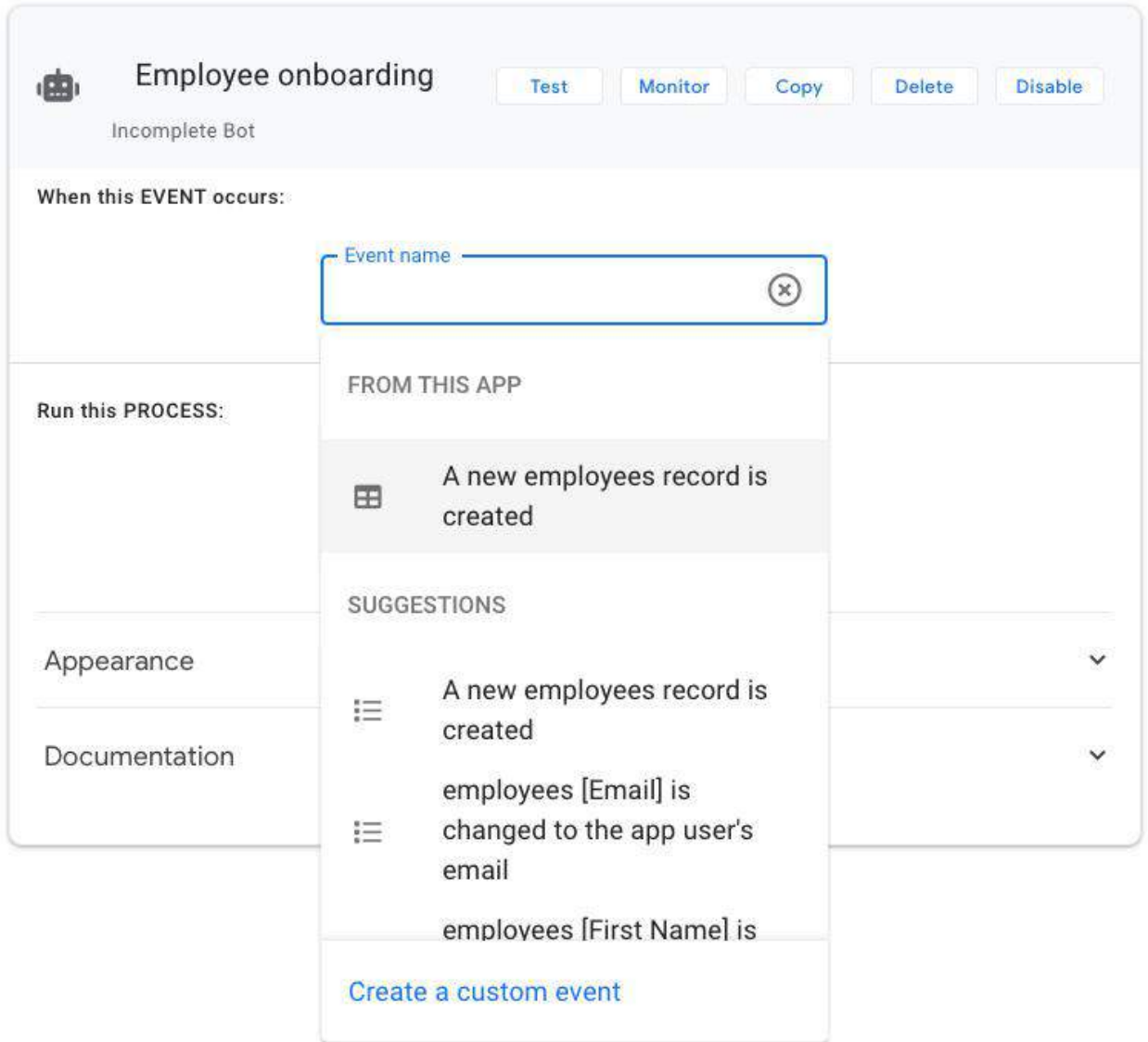
## Configuring the event

Once the bot is created, you can define the event for the bot. In this example, the event trigger is the addition of a new employee to Google Sheets.

To create an event:

1. Select the bot you just created in the previous section.
2. Click **Choose an event**.
3. Suggested events should appear. (Skip to step 5 if there aren't any suggested events)





4. Select **A new Employee record is created** to have the event automatically created for you. The platform does this for you and saves you valuable time.

**Note:** If you want to manually configure the event, you can click on the “Create a custom event” link.

The screenshot displays the AppSheet Enterprise interface for configuring an automation bot. The main workspace is titled "Employee onboarding" and is currently in an "Incomplete Bot" state. It is divided into three sections:

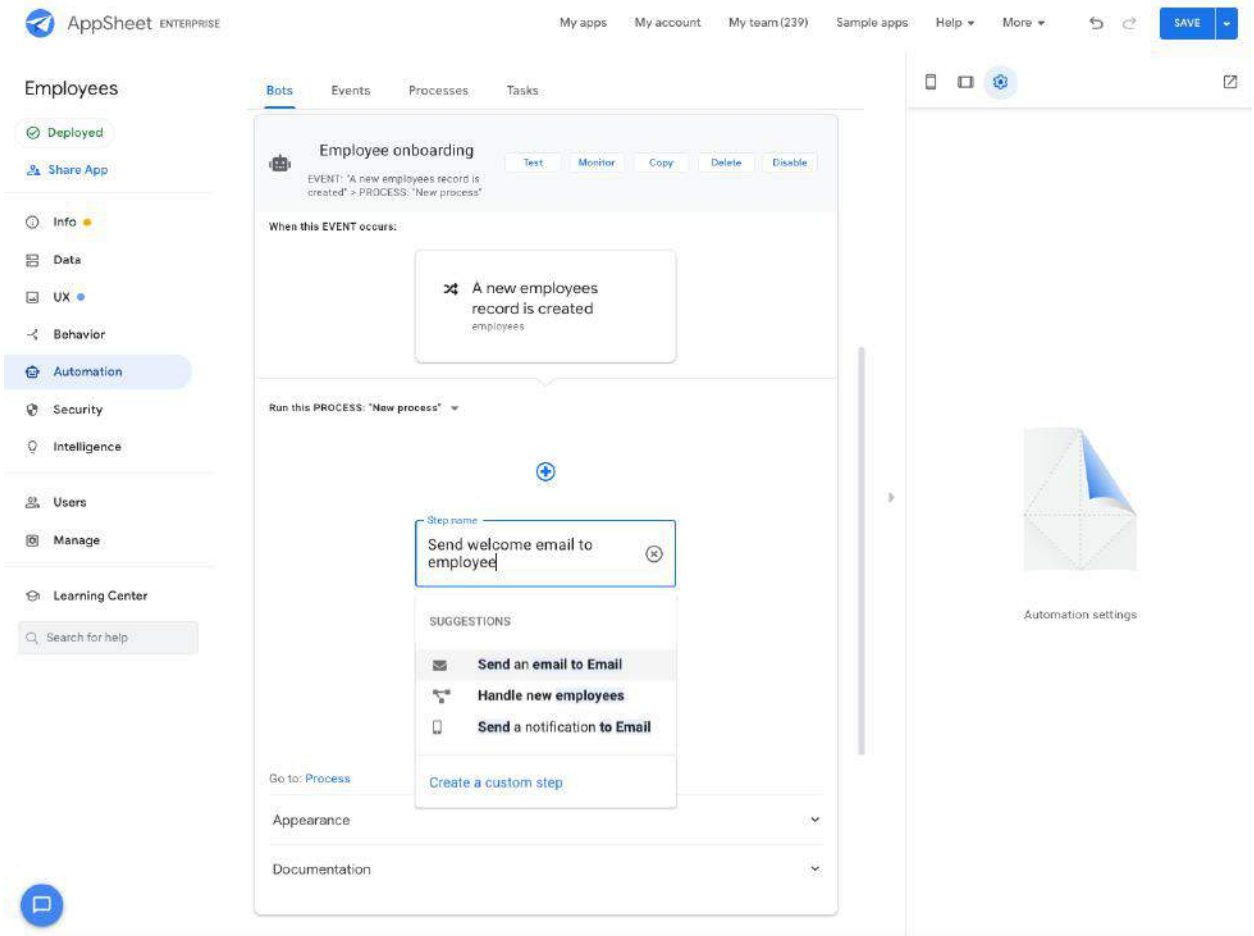
- When this EVENT occurs:** A blue callout box indicates the event: "A new employees record is created" on the "employees" table.
- Run this PROCESS:** A section with a "+ Add a step" button and dropdown menus for "Appearance" and "Documentation".
- Configuration Panel (Right):**
  - Event name:** "A new employees record is created"
  - Event Type:** "Data Change" (selected over "Schedule")
  - Table:** "employees"
  - Condition:** "="
  - Data change type:** "ADDS\_ONLY"
  - Bypass Security Filters?:** A toggle switch is currently turned off.
  - Appearance:** A dropdown menu.
  - Documentation:** A dropdown menu.

The top navigation bar includes "My apps", "My account", "My team (239)", "Sample apps", "Help", "More", and a "SAVE" button. The left sidebar shows navigation options like "Info", "Data", "UX", "Behavior", "Automation", "Security", "Intelligence", "Users", "Manage", and "Learning Center".

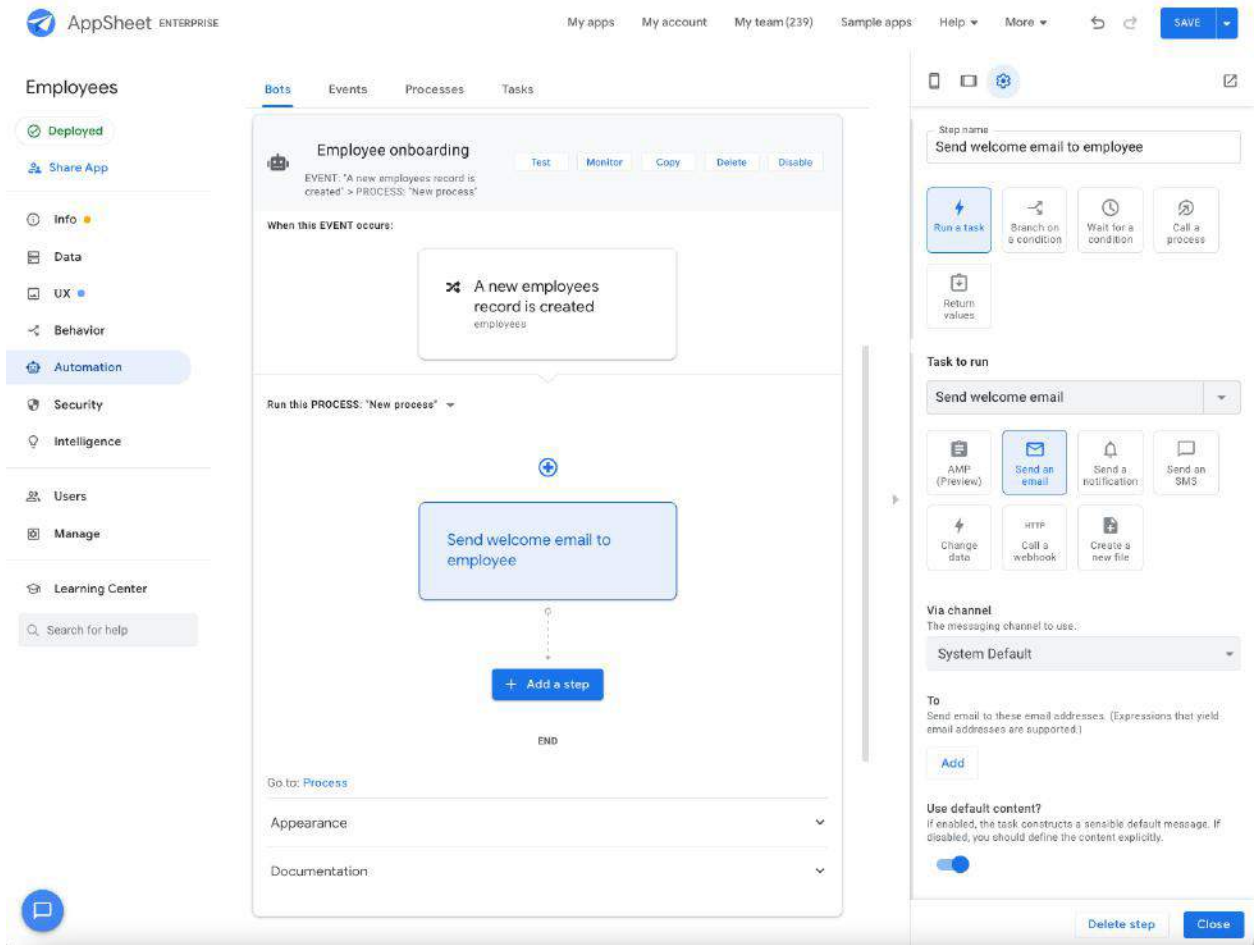
## Configuring the process

Configure the process the bot should trigger based on the event:

1. In the **Run this process** section of the editor, click **+Add a step**. Type `Send welcome email to employee` into the text box and hit **enter** to automatically create an empty step. This implicitly creates a process.



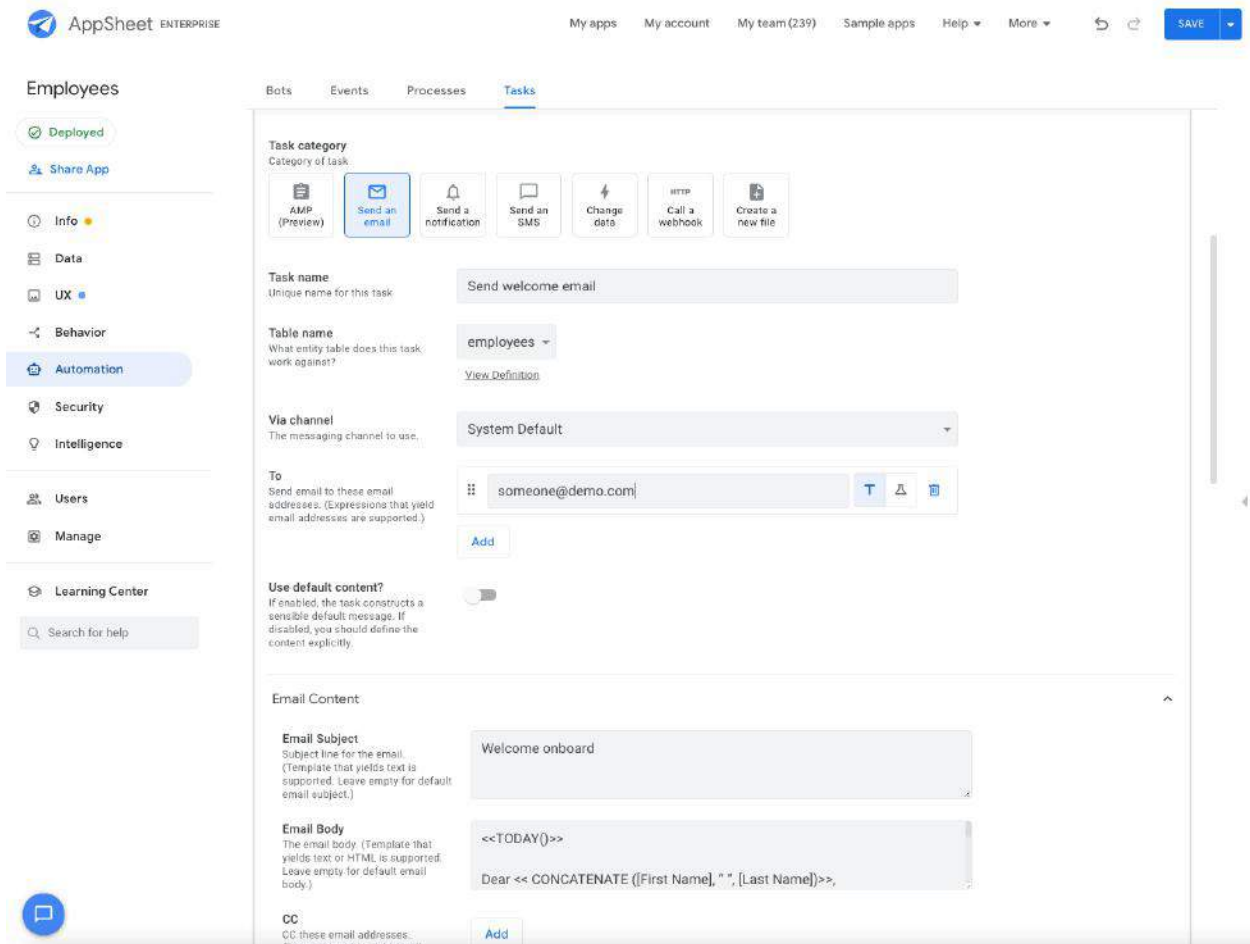
2. Click on the **Send welcome email to employee** step to see its definition on the settings pane to the right.
3. In the **Settings** pane, configure the step as follows:
  - a. Select **Run a task**.
  - b. In the **Task to run** section, click **Create new task**.



- c. Type in the name you want for the task and click on the **Go to : Task** link on the bottom of the right panel to configure the task.
- d. Refer to the table below for the rest of the task configuration:

Field Name	Value
Do this	Run a task
Task category	Send an email
Task name	Email welcome letter
Table Name	employees
To	<enter your email address>
Use default content?	Disabled

Email Subject	Welcome onboard
Email body	<p>&lt;&lt;TODAY()&gt;&gt;</p> <p>Dear &lt;&lt; CONCATENATE ([First Name], " ", [Last Name])&gt;&gt;,</p> <p>We are excited to have you on board. We are delighted to have you join us as a &lt;&lt;[Job Title]&gt;&gt; in our team. Your role is critical in fulfilling our mission.</p> <p>The Human Resources team is here to support you throughout your career.</p> <p>Sincerely, HR Manager</p>



4. Click **Save** to save all your changes.

Your bot should now look like this:

The screenshot displays the AppSheet Enterprise interface for configuring a bot named "Employee onboarding". The bot is currently in a "Deployed" state, as indicated by the green checkmark in the top-left sidebar. The main configuration area shows the following details:

- Event:** "A new employees record is created" (with a sub-label "employees").
- Process:** "New process".
- Task:** "Send welcome email to employee".
- Buttons:** "Test", "Monitor", "Copy", "Delete", and "Disable" are visible in the top right of the configuration panel.
- Navigation:** The "Bots" tab is selected, with other tabs for "Events", "Processes", and "Tasks" visible.
- Footer:** A "Go to: Process" dropdown menu is visible at the bottom of the configuration panel, with options for "Appearance" and "Documentation".

You have created and enabled your bot. All the required components, including the event, process, and tasks, have been created. Each component can be accessed and edited from individual tabs if required.

**Note:** Make sure your app is **deployed**. If not, deploy it for the bot to run. The bot listens for events and trigger processes only if the app is deployed and the specific bot is enabled.

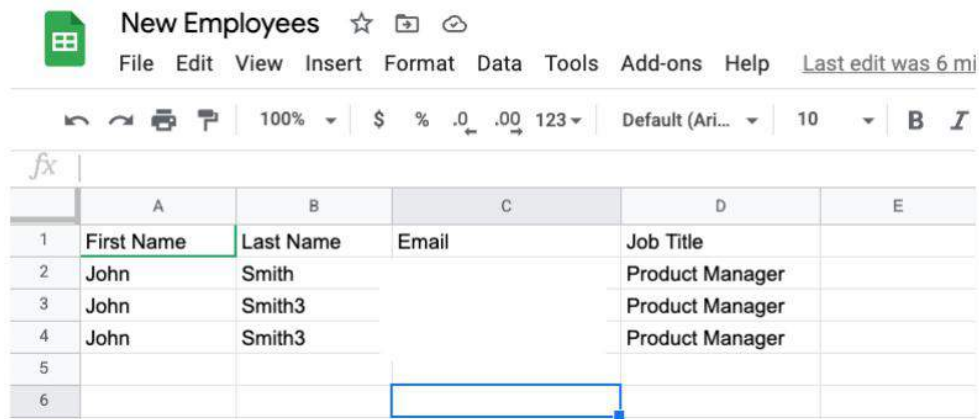
If you explore the Events, Tasks and Processes tabs, you can see the reusable artifacts that have been generated when you were creating your bot. Each of these components can be re-used inside processes and inside other bots.

Now you are ready to test your bot.

## Testing your automation

Follow these steps to test your automation:

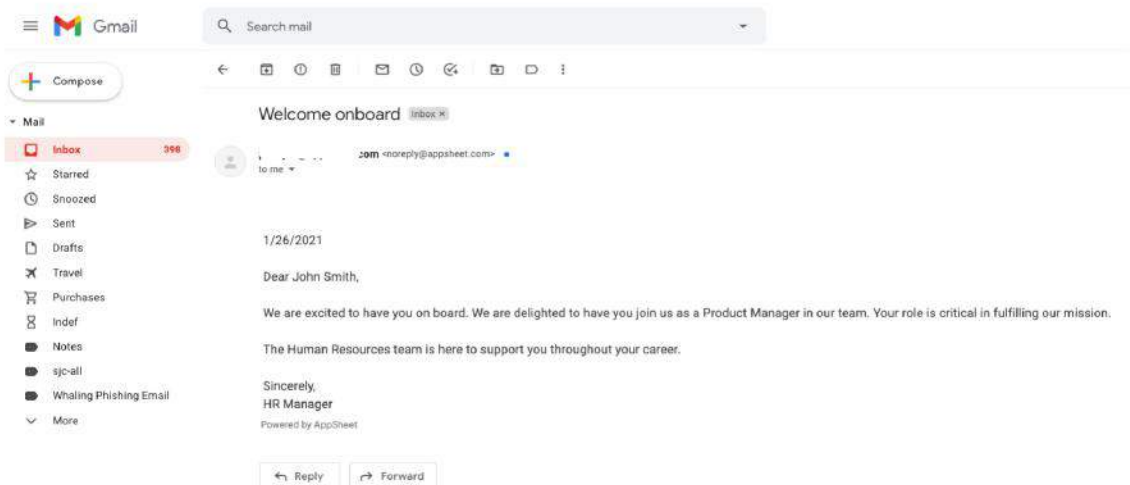
1. Make sure your bot is **Enabled**.
2. Add a new employee row to your sheet. You can copy a whole row (from a separate sheet) and paste it into your employees sheet or you can manually enter values in each cell



	A	B	C	D	E
1	First Name	Last Name	Email	Job Title	
2	John	Smith		Product Manager	
3	John	Smith3		Product Manager	
4	John	Smith3		Product Manager	
5					
6					

The ADD event is fired as soon as you are done entering values in the row (currently the add-on has a 20 second wait as a default wait time to allow for data entry manually)

3. As soon as the event is detected, the bot should send an email at the “To” address configured in the task:



## Automation Monitoring

As various bots are configured, events are detected and processes start executing, it becomes important to keep a track of which executions were successful, which ones are pending and which ones encountered errors. Automation monitoring is delivered as an AppSheet app that can be easily accessed by clicking **Monitor** on your bot definition.

In the current version of the monitoring app, you have two views:

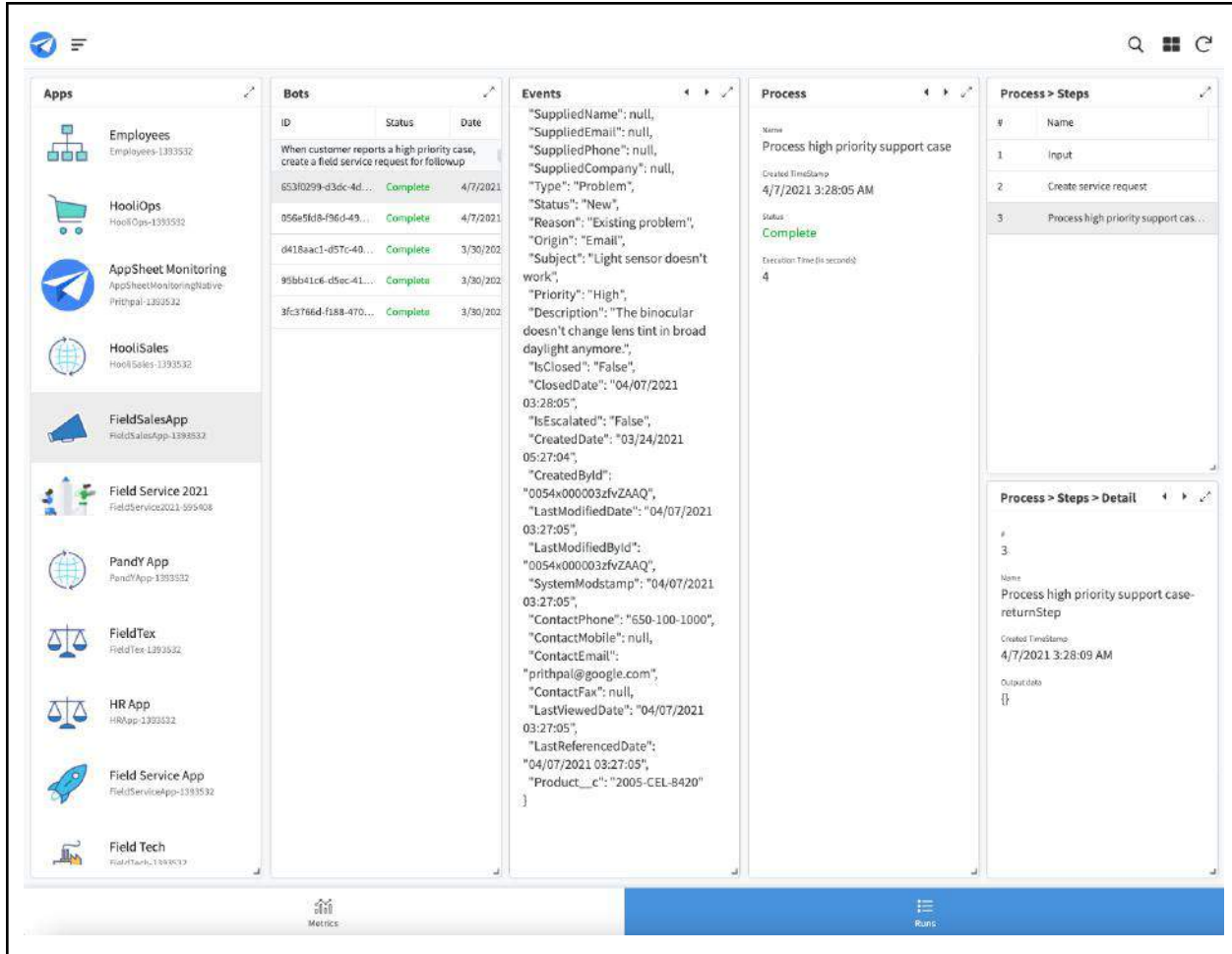
- 1) **Metrics** (High level metrics for bots) and
- 2) **Runs** (Bots execution logs).

We are constantly adding new views/features to this monitoring app to make it even more useful.

**Note:** *Your view may look different from the screen shots presented below. Time is in UTC.*







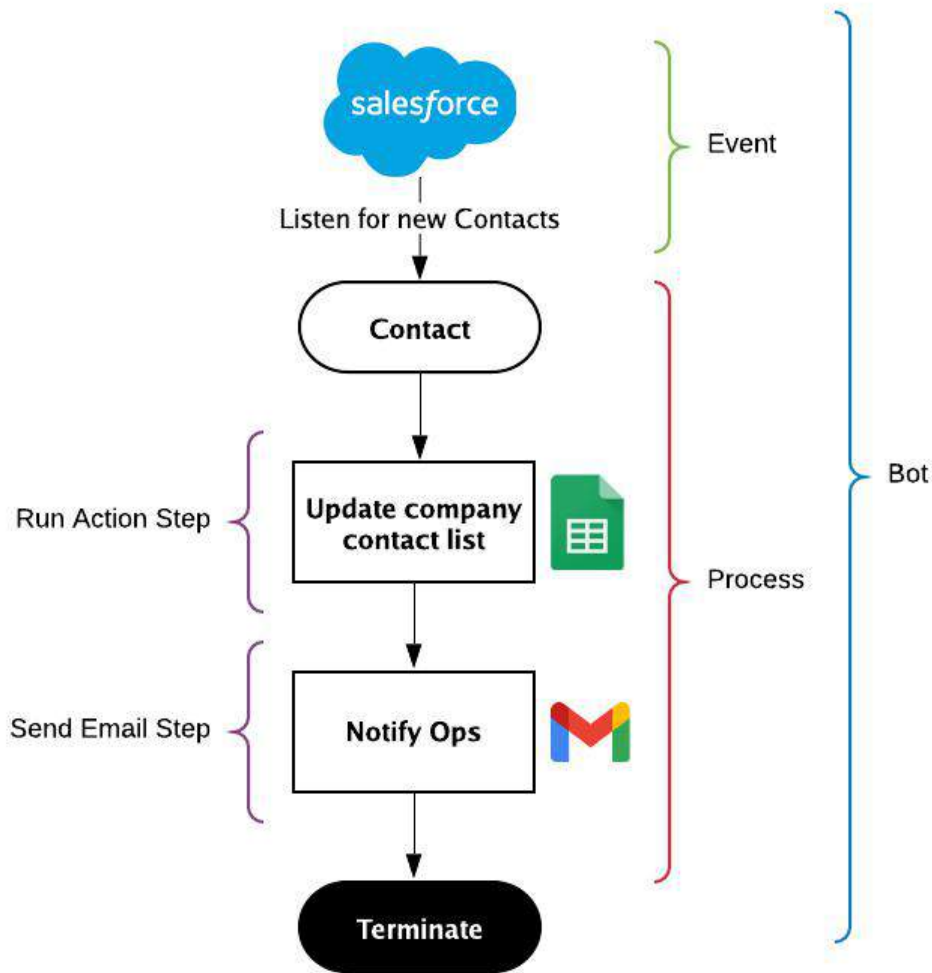
## Contact Sync

This brief tutorial shows you how to set up an automated process to sync contacts between two systems. This example implements a solution to a common business process scenario. When a new contact is created in Salesforce, AppSheet automatically updates a Google Sheet with the new contact.

Implementing contact sync with AppSheet Automation involves the following steps:

1. [Creating the Bot](#)
2. [Configuring the Event](#)
3. [Configuring the Process \(steps and actions\)](#)

The following figure describes the high level components:



### Before you begin this tutorial:

- Make sure you have completed the prerequisite in the [Before you begin](#) section.
- Follow the steps in the Appendix to configure a [Contact entity](#) from a Salesforce data source and [Company Contacts](#) entity from a Google Sheets data source.

Automations begin with configuring the bot. A bot binds an *event* (ex: a new contact created in Salesforce) to a *process* (ex. a sequence of steps to update a contact Sheet). The intuitive AppSheet editor enables you to create all the components of the automation/bot in-situ. That is, you can configure the event and process (including steps/tasks) without needing to switch between tabs to configure individual components.

In addition, AppSheet Automation is an intent-aware platform. The platform understands user intent and recommends configuration options that align with what you are trying to achieve.

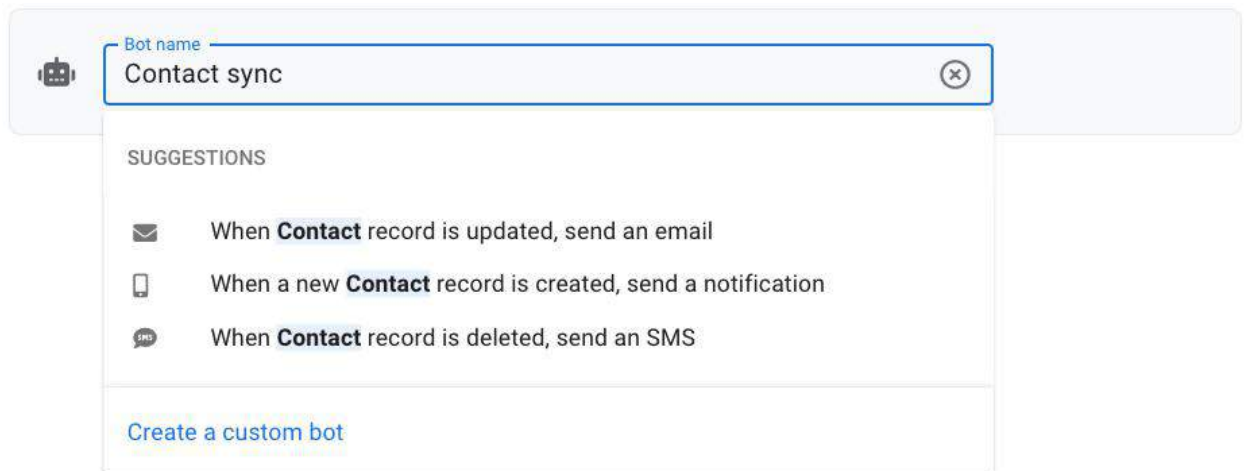
**Note:** The suggestions you see depend on your data configuration, existing event configurations and may appear different from what you see in some of the images below.

The following sections guide you through configuring your end to end automation:

## Creating the bot

Follow the steps below to create a new bot:

1. From the AppSheet UI, navigate to the **Bots** tab and click **New Bot**.
2. Type `Contacts Sync` into the dialog and suggested actions appear, as shown in the image below.
3. Select **Create a custom** to create the bot.

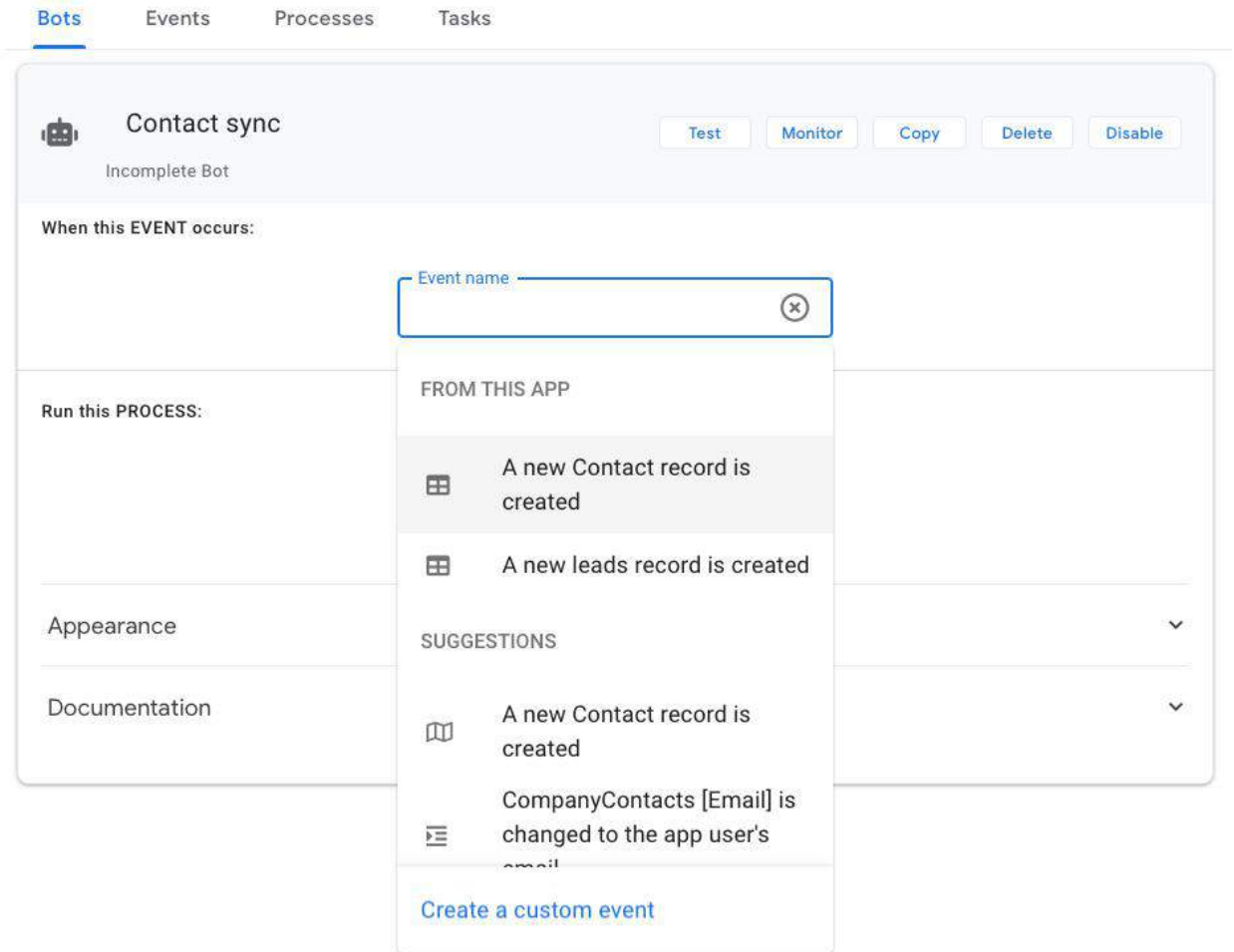


## Configuring the event

Once the bot is created, you can define the event for the bot. In this example, the event trigger is the addition of new contacts to Salesforce.

To create an event:

1. Select the bot you just created in the previous section.
2. Click **Choose an event**.
3. Suggested events appear.
4. Select **A new Contact record is created** to have the event automatically created for you. The platform does this for you and saves you valuable time.



**Note:** If you want to manually configure the event, you can navigate to the **Events** tab, select **Create new event**, and enter in the relevant details.

Once your event has been created, your bot editor should look like this:

The screenshot shows the AppSheet automation editor interface. At the top, there are navigation tabs: 'Bots' (selected), 'Events', 'Processes', and 'Tasks'. Below the tabs, the bot's name 'Contact sync' is displayed with a robot icon and the status 'Incomplete Bot'. To the right of the name are five buttons: 'Test', 'Monitor', 'Copy', 'Delete', and 'Disable'. Underneath, the trigger section is labeled 'When this EVENT occurs:' and contains a blue box with a checkmark icon and the text 'A new Contact record is created' and 'Contact'. Below this is the 'Run this PROCESS:' section, which has a dropdown arrow and a blue button labeled '+ Add a step'. At the bottom, there are two expandable sections: 'Appearance' and 'Documentation', each with a downward arrow.

## Configuring the process

Configure the process your event should trigger:

1. In the **Run this process** section of the editor, click **+Add a step**. Type `Update contact list` into the text box and click on **Create a custom step** to automatically create an empty step. This implicitly creates a process.


**Bots**   Events   Processes   Tasks

### Contact sync

EVENT: "A new Contact record is created" > PROCESS: "New process 3"


Test   Monitor   Copy   Delete   Disable

When this EVENT occurs:

 A new Contact record is created




Contact

Run this PROCESS: "New process 3" ▼



Step name

SUGGESTIONS

-  **Update company contact**
-  Set LeadSource to **Purchased List**
-  **Sync contact process**

[Create a custom step](#)

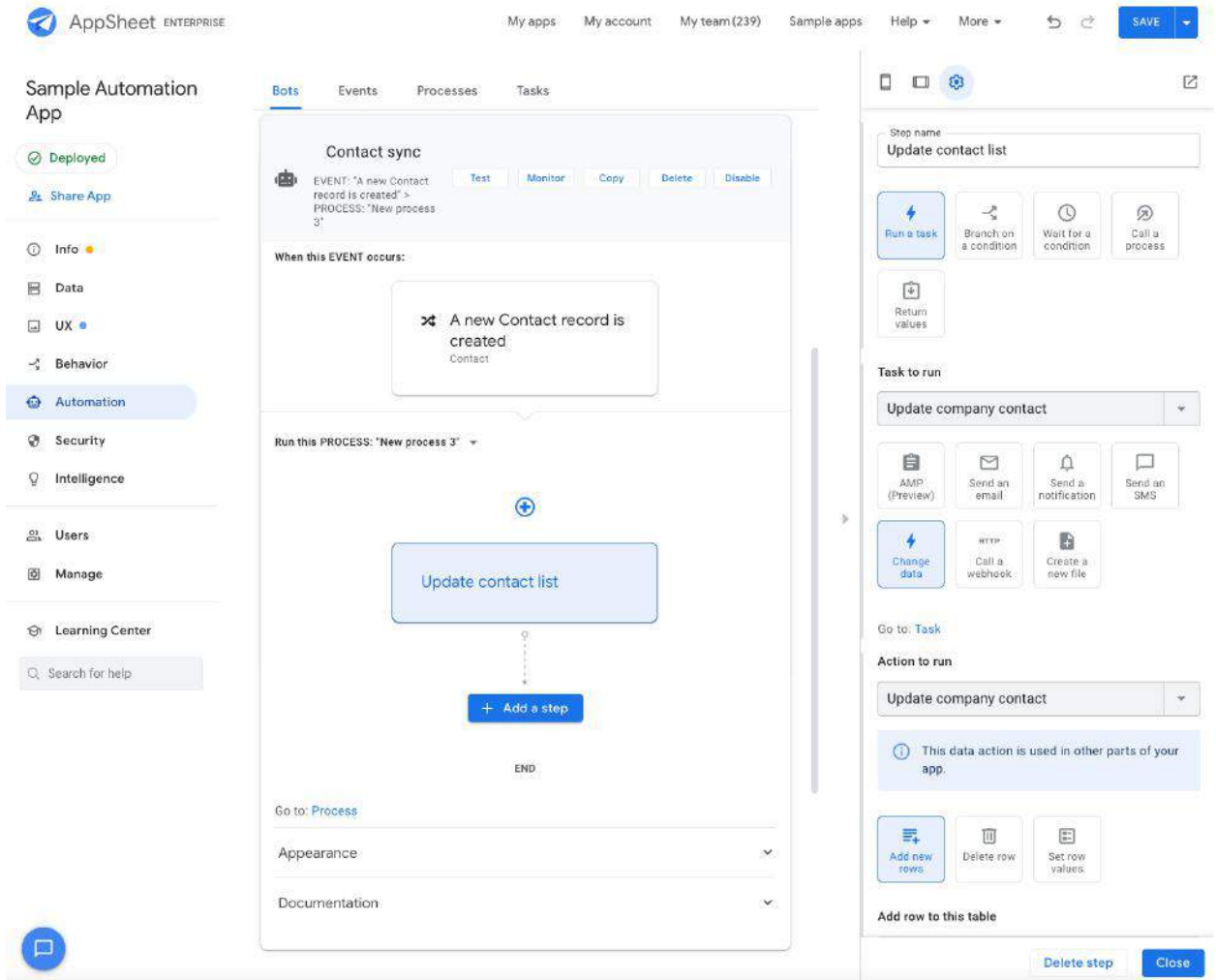
Go to: [Process](#)

Appearance ▼

Documentation ▼

2. Click on the **Update contact list** step to expand its definition
3. In the **Settings** pane, configure the step as follows:
  - a. Select **Run a task**.
  - b. In the **Task** section, click **Create new task**.

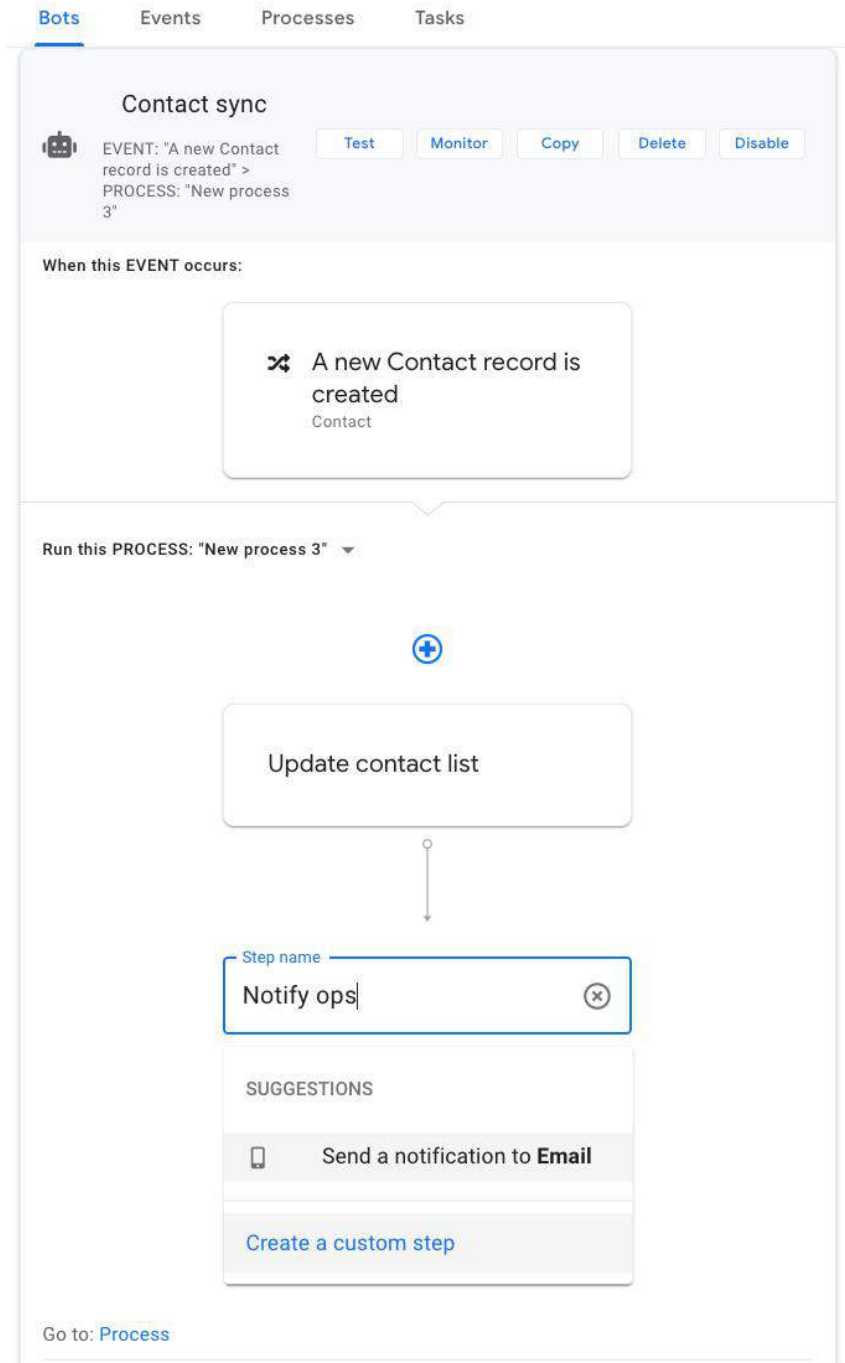
- c. And under **Task Category**, select **Change data**.
- d. Type `Update company contact` as the **Task name**.
- e. Select **Contact** as the **Table name**.
- f. Select **Update company contact** as the **Data Change Action Name**.



- g. Click **Save** to save all your changes.

4. Add another step to notify the Ops team of the new contact:
  - a. Click **+ Add a step**.
  - b. Type `Notify ops` as the step name and select **Create a custom step** from the suggested options.





- c. In the **Settings** pane, Select **Run a task** .
- d. From the **Task** dropdown, select **Create new Task**.
- e. Select **Send an email** as the **Task category**.

Step name  
Notify ops

Run a task  
Branch on a condition  
Wait for a condition  
Call a process

Return values

Task to run  
Send an email to ops

AMP (Preview)  
Send an email  
Send a notification  
Send an SMS

Change data  
Call a webhook  
Create a new file

Via channel  
The messaging channel to use.  
System Default

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

Use default content?  
If enabled, the task constructs a sensible default message. If disabled, you should define the content explicitly.

Go to: [Task](#)

- f. Type `Send an email to ops` for the **Task name**. (Click on [Go to: Task](#) link to finish configuring the test of the task configuration)
- g. Select **Contact** as the **Table name**.
- h. Enter `<your email address>` in the **To** field.
- i. For **Email Subject**, type `New contact notification`.
- j. For **Email Body**, type `A new contact - <<Name>>, email <<Email>> just got added`. Leave all the other fields at their default values.

Bots Events Processes **Tasks**

Send an email to ops Copy Delete

**Task category**  
Category of task

AMP (Preview) **Send an email** Send a notification Send an SMS Change data HTTP Call a webhook Create a new file

**Task name**  
Unique name for this task: Send an email to ops

**Table name**  
What entity table does this task work against? Contact [View Definition](#)

**Via channel**  
The messaging channel to use: System Default

**To**  
Send email to these email addresses. (Expressions that yield email addresses are supported.) someone@demo.com T 👤 🗑️  
Add

**Use default content?**  
If enabled, the task constructs a sensible default message. If disabled, you should define the content explicitly.

**Email Content**

**Email Subject**  
Subject line for the email. (Template that yields text is supported. Leave empty for default email subject.) New contact notification

**Email Body**  
The email body. (Template that yields text or HTML is supported. Leave empty for default email body.) A new contact - <<Name>>, email <<Email>> just got added.

5. Click **Save** and then click **Done**.

Your bot should now look like this:

The screenshot shows the 'Contact sync' bot configuration in AppSheet. At the top, there are navigation tabs for 'Bots', 'Events', 'Processes', and 'Tasks'. The bot name 'Contact sync' is displayed, along with a 'Test' button and a 'Monitor' button. Below the bot name, the event and process definitions are shown: 'EVENT: "A new Contact record is created" > PROCESS: "New process 3"'. Under the 'When this EVENT occurs:' section, a trigger event is defined: 'A new Contact record is created' (Contact). Under the 'Run this PROCESS: "New process 3"' section, a flowchart is shown with two steps: 'Update contact list' followed by 'Notify ops'. Below the flowchart is a '+ Add a step' button and an 'END' label. At the bottom left, there is a 'Go to: Process' link and an 'Appearance' dropdown menu.

You have created and enabled your bot. All the required components, including the event, process, and tasks, have been created. Each component can be accessed and edited from individual tabs if required.

**Note:** Make sure your app is **deployed**. If not, deploy it for the bot to run. The bot listens for events and trigger processes only if the app is deployed and the specific bot is enabled.

If you explore the Events, Tasks and Processes tabs, you can see the reusable artifacts that have been generated when you were creating your bot. Each of these components can be

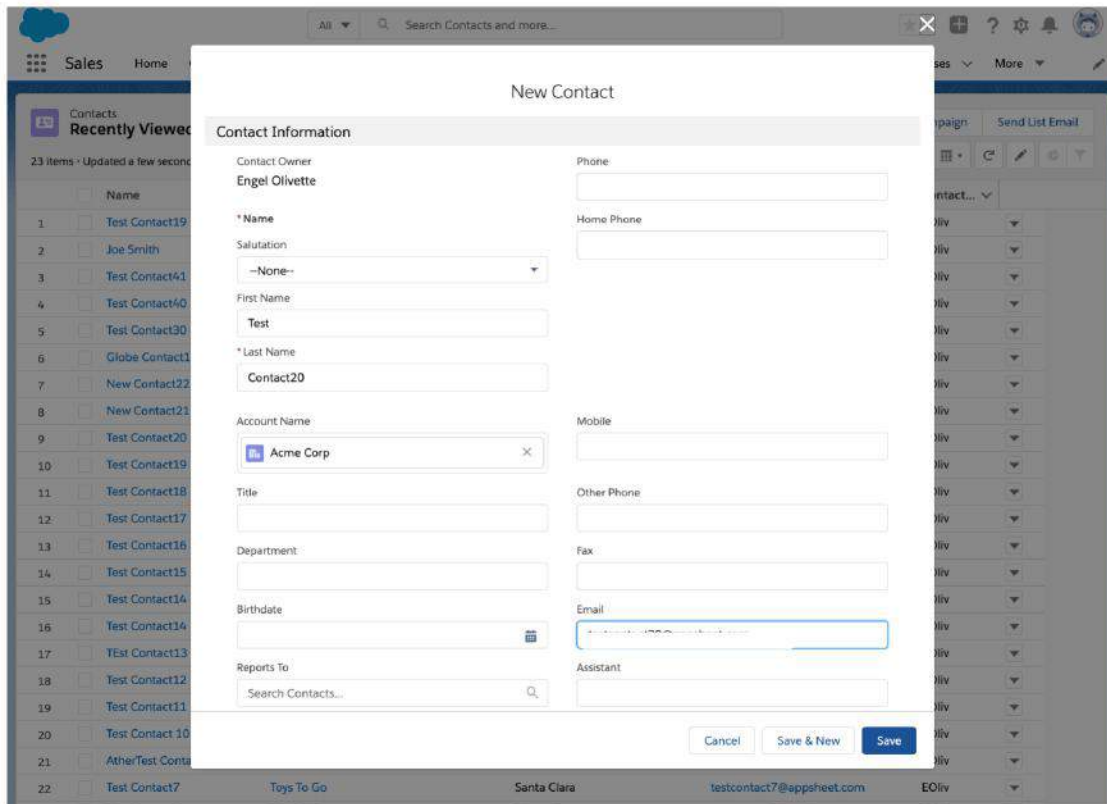
re-used inside processes and inside other bots.

Now you are ready to test your bot.

## Testing your automation

Follow these steps to test your automation:

1. Make sure your bot is **Enabled**.
2. Login to your Salesforce Instance and create a new contact.

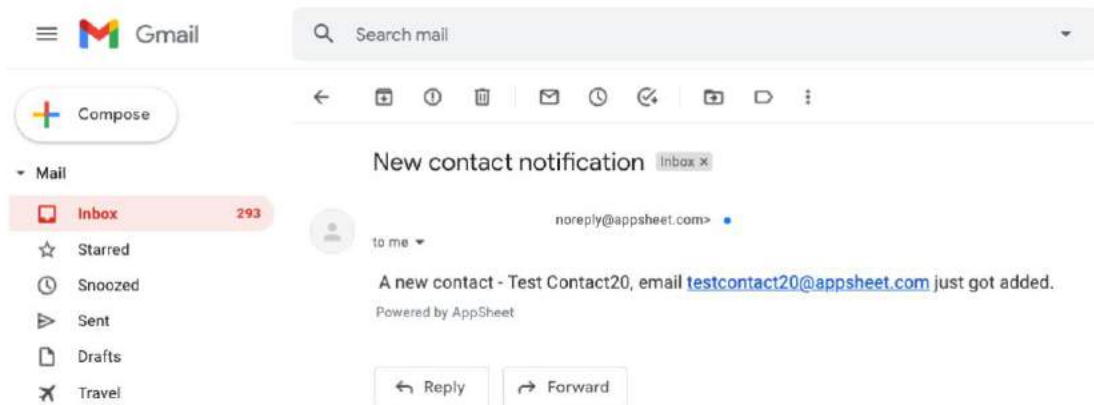


The Bot listens for new Contact events every 60 seconds.

3. In about 60 seconds, check to confirm that a new contact is added to your CompanyList Sheet.

	A	B	C	D	E
1	First Name	Last Name	Email	Company	Date Created
2	Test	Contact11	.com	0016g0000Nno8PAAR	10/31/2020
3	Test	Contact12	.com	0016g0000Nno8PAAR	10/31/2020
4	Test	Contact13	.com	0016g0000Nno8PAAR	11/2/2020
5	Test	Contact14	.com	0016g0000Nno8PAAR	11/4/2020
6	Test	Contact18	.com	0016g0000NnUrgAAF	1/19/2021
7	Test	Contact19	.com	0016g0000NnUrgAAF	11/5/2020
8	Test	Contact20	sheet.com	0016g0000NnUrgAAF	1/20/2021
9					
10					

4. You should also receive an email similar to this:

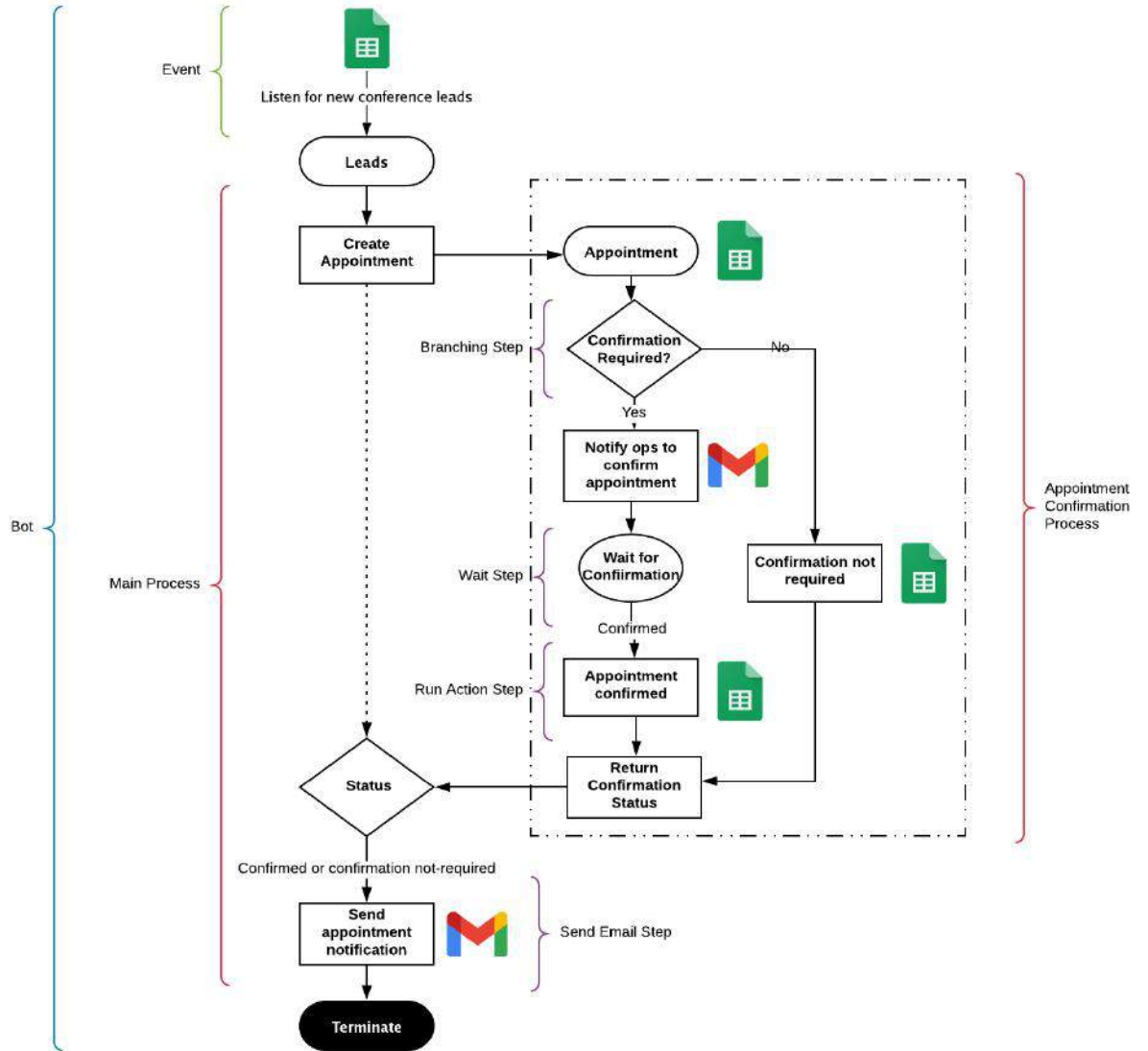


## Conference leads appointment generator

This brief tutorial shows you how to set up an automated process to create appointments for leads attending a conference who have requested a follow up. When a new lead is created in Google Sheets, AppSheet determines if a follow up has been requested by the individual. If so, AppSheet automatically creates an appointment entry in another Google Sheet and notify the individual via email.

A specific lead can have a profiled value of “High”, “Medium” or “Low”. If the lead is a “High” profile lead, then a phone based confirmation is required for the lead. Once the appointment is confirmed by phone, a second notification (SMS) is sent to the individual.

The following figure depicts the high level process flow:



Implementing this use case with AppSheet Automation involves the following steps:

1. [Configuring the Appointment Confirmation Process](#) (A sub process invoked from the main process)
2. [Creating the Bot](#)
3. [Configuring the Event](#)
4. [Configuring the Main Process](#) (steps and tasks)

**Before you begin this tutorial:**

- Configure the [leads](#) and [appointments](#) entities (tables) in your Sample Automation App.
- Install the [AppSheet Events add-on](#) for Google Sheets.

- Configure an [email template](#) for ops notification

The following sections guide you through configuring your end to end automation *in-situ* using a bot.

## Configuring the Appointment Confirmation Process

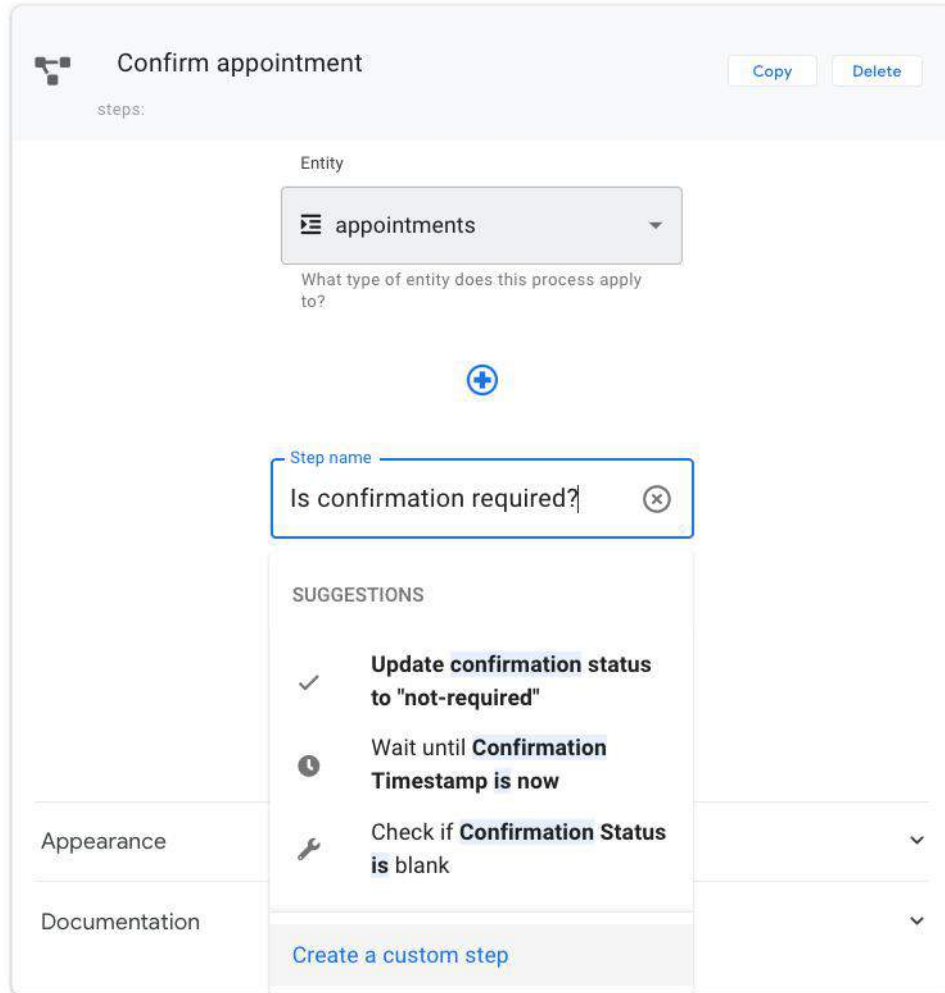
Navigate to the Process Tab from the top.

1. Click **+ New Process** to create a new empty process.
2. Type `Confirm appointment` as the **Process name**.
3. Select **appointments** as the entity name from the **Entity** drop down. At this point your process should look like this:

The screenshot shows the configuration interface for a process named "Confirm appointment". At the top right, there are "Copy" and "Delete" buttons. Below the title, there is a "steps:" label. The main area is titled "Entity" and contains a dropdown menu with "appointments" selected. Below the dropdown, there is a question: "What type of entity does this process apply to?" and a blue button labeled "+ Add a step". At the bottom, there are two expandable sections: "Appearance" and "Documentation", each with a downward arrow.

4. To add steps to complete the process, click **+ Add a step**.
5. In **Step name**, type `Is confirmation required?` and select **Create a custom step** from the dropdown.

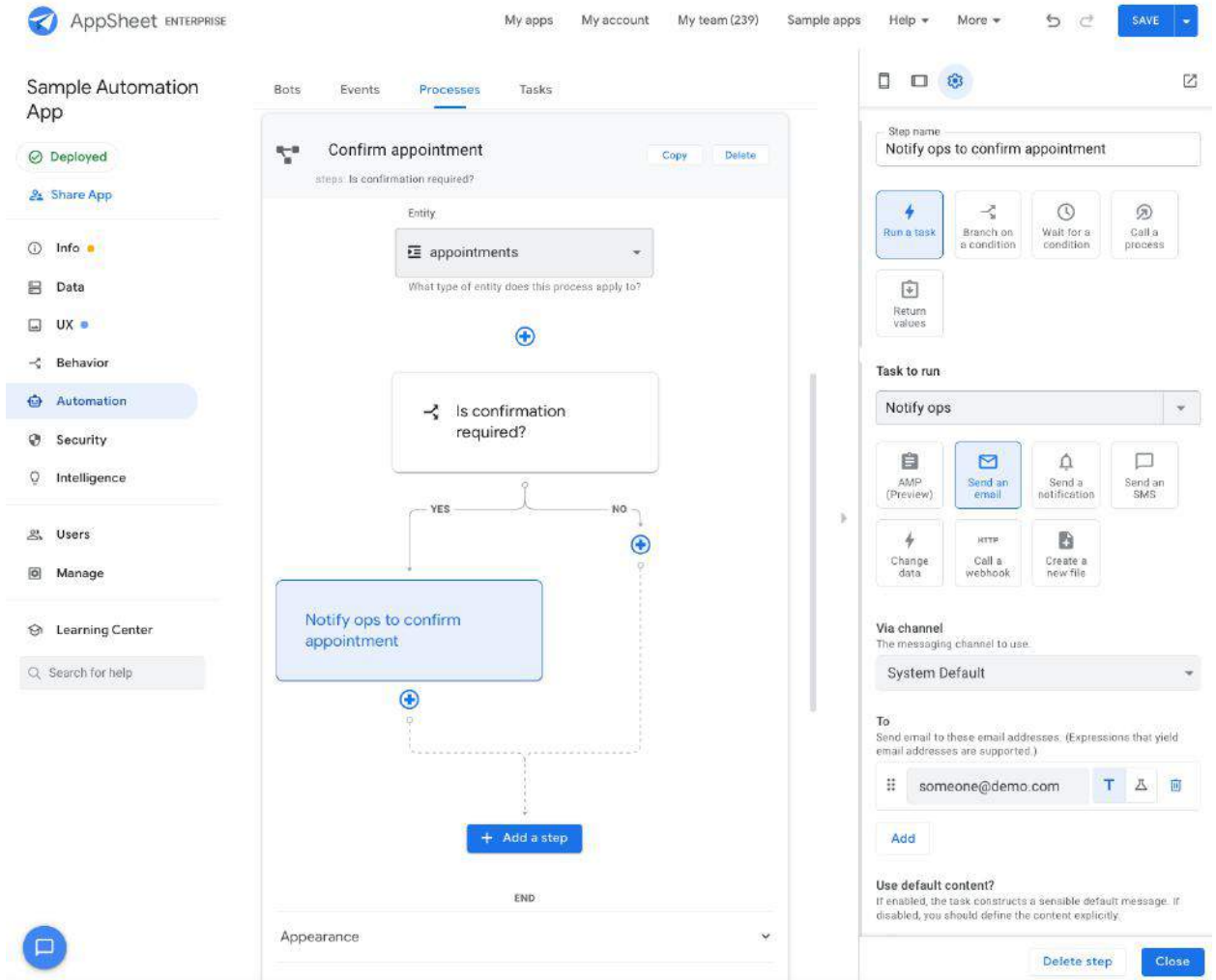




6. In the **Settings** pane, select **Branch on a condition** under **Do this**.
7. Enter `[Confirmation Required] = "Y"` under **Condition to check**. Your step should look like this:

The screenshot displays the AppSheet interface for configuring a process. The main workspace shows a process named "Confirm appointment" with the entity "appointments". A decision step "Is confirmation required?" is added, with "YES" and "NO" branches leading to "Add a step" buttons. The "Settings" pane on the right is open, showing the step name "Is confirmation required?" and the condition "[Confirmation Required] = 'Y'".

8. Click the **+** icon on the **Yes** branch.
9. In the **Settings** pane, enter `Notify ops` to confirm appointment as the **Step Name**, select **Create a custom step** and configure this step as described in the table below:



Field Name	Value
Step type	Run a task
Task to run	Notify Ops
<p>Click on “Go to: Task” at the bottom of the pane to configure the rest of the task as described below:</p>	
Table Name	Appointments
To	<your email address>
Email Subject	Appointment confirmation request
Email body template	Ops Notification Template (selected from Google Drive) <b>Note:</b> For more information, see <a href="#">Configure an email template</a> for ops notification.

Use default content?	Disabled
----------------------	----------

Leave default settings for other fields. The following screenshots show the step config in detail:

Bots   Events   Processes   **Tasks**

appointments

**Notify ops** Copy   Delete   ⌵

**Task category**  
Category of task

AMP  
(Preview)

Send an email

Send a notification

Send an SMS

Change data

HTTP  
Call a webhook

Create a new file

**Task name**  
Unique name for this task:

**Table name**  
What entity table does this task work against?  ⌵  
[View Definition](#)

**Via channel**  
The messaging channel to use.  ⌵

**To**  
Send email to these email addresses. (Expressions that yield email addresses are supported.)  
 T   👤   🗑️  
Add

**Use default content?**  
If enabled, the task constructs a sensible default message. If disabled, you should define the content explicitly.

---

**Email Content** ⌵

**Email Subject**  
Subject line for the email. (Template that yields text is supported. Leave empty for default email subject.)

Bots Events Processes **Tasks**

### Email Content

**Email Subject**  
Subject line for the email. (Template that yields text is supported. Leave empty for default email subject.)

Appointment confirmation request

**Email Body**  
The email body. (Template that yields text or HTML is supported. Leave empty for default email body.)

**CC**  
CC these email addresses. (Expressions that yield email addresses are supported.)

Add

**BCC**  
BCC these email addresses. (Expressions that yield email addresses are supported.)

Add

**Reply To**  
Replies to the email will go to this address. (Expression that yields an email address is supported.)

T A

**From Display**  
Send email with this From Display name. (Template that yields text is supported.)

**PreHeader**  
Email preview seen in mobile email apps. (Template that yields text is supported.)

**Email Body Template**  
The template file used to format the email body. (Leave empty if no body template used.)

DocId=1XG5lwOFR9VigrG1yO3MtVPA6Dl8VB3z3p4T\_QWuY Create View

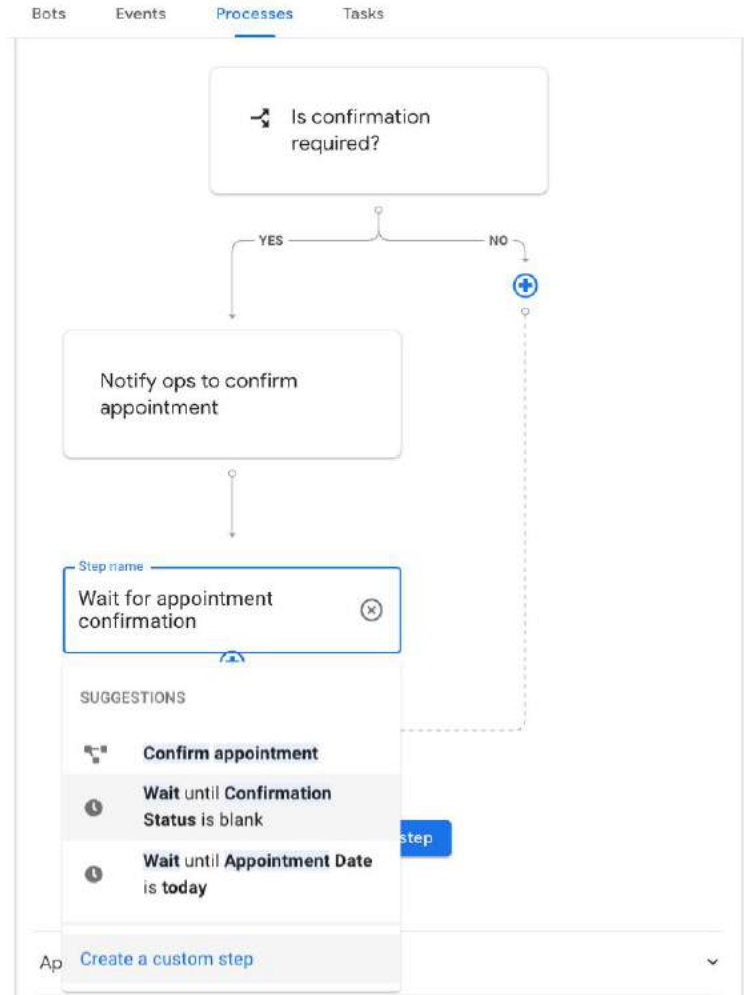
**Attachment Content Type**  
Content type for the email attachment

PDF

**Attachment Template**  
The template file used to format the email attachment. (Leave empty if no attachment template used.)

Create

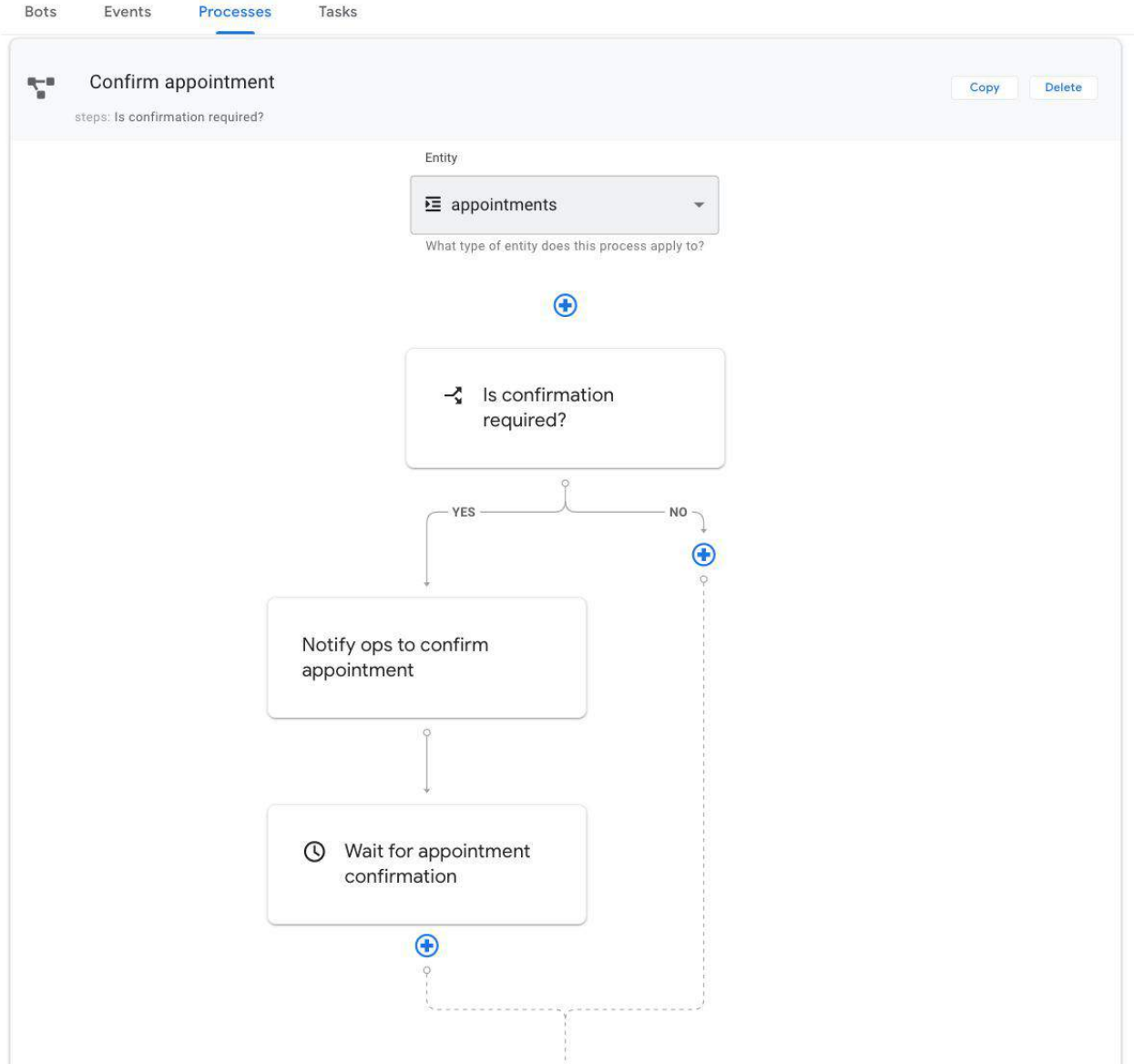
10. Click **Save** to save your changes.
11. Add a wait step to pause execution in the process until the condition (Ops have appointment confirmation) is satisfied.
  - a. Click the **+** icon below **Notify ops to confirm appointment** to create a new step.
  - b. Type `Wait for appointment confirmation` into **Step Name** and select **Create a custom step**



- c. In the **Settings** pane, select **Wait for a condition** under **Do this**.
- d. Type `UPPER([Confirmation Status]) = "CONFIRMED"` in **Wait until this condition is true** as follows:

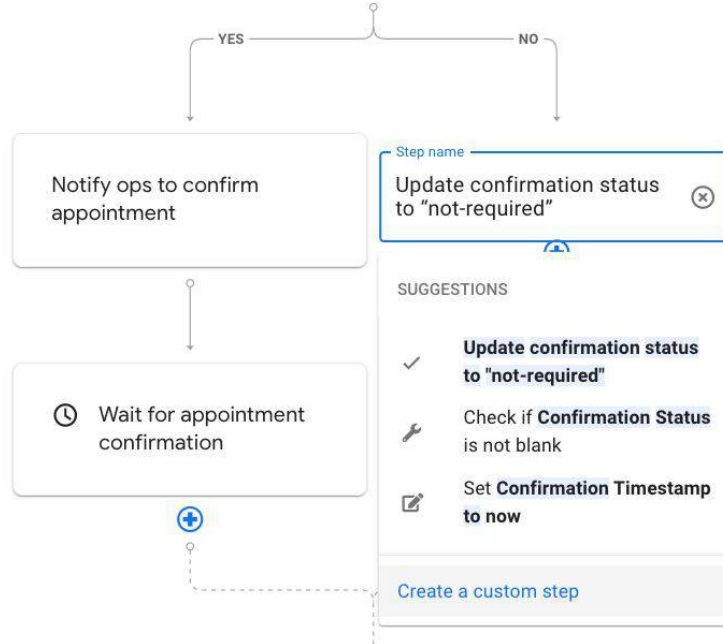
The screenshot displays the AppSheet Automation interface. On the left, a sidebar lists various app management options, with 'Automation' selected. The main workspace shows a process flowchart under the 'Processes' tab. The flow starts with a decision step 'Is confirmation required?'. If 'YES', it proceeds to 'Notify ops to confirm appointment', then to a 'Wait for appointment confirmation' step. If 'NO', it bypasses the notification and wait steps. Both paths converge at an 'Add a step' button, which leads to the 'END' of the process. On the right, a configuration panel for the 'Wait for appointment confirmation' step is open. It shows a condition: '= UPPER([Confirmation Status]) = \*CONFIF'. The interface includes a top navigation bar with 'SAVE' and a bottom right panel with 'Delete step' and 'Close' buttons.

12. Click **Save** and then **Close**. Your process configuration should look like this:



13. Finish the “No” branch of the process.
  - a. Click + to create a new step.
  - b. Type Update confirmation status to “not-required” as the **Step Name**, click on **Create a custom step** to create this step.





c. In the **Settings** pane, configure the fields as follows:

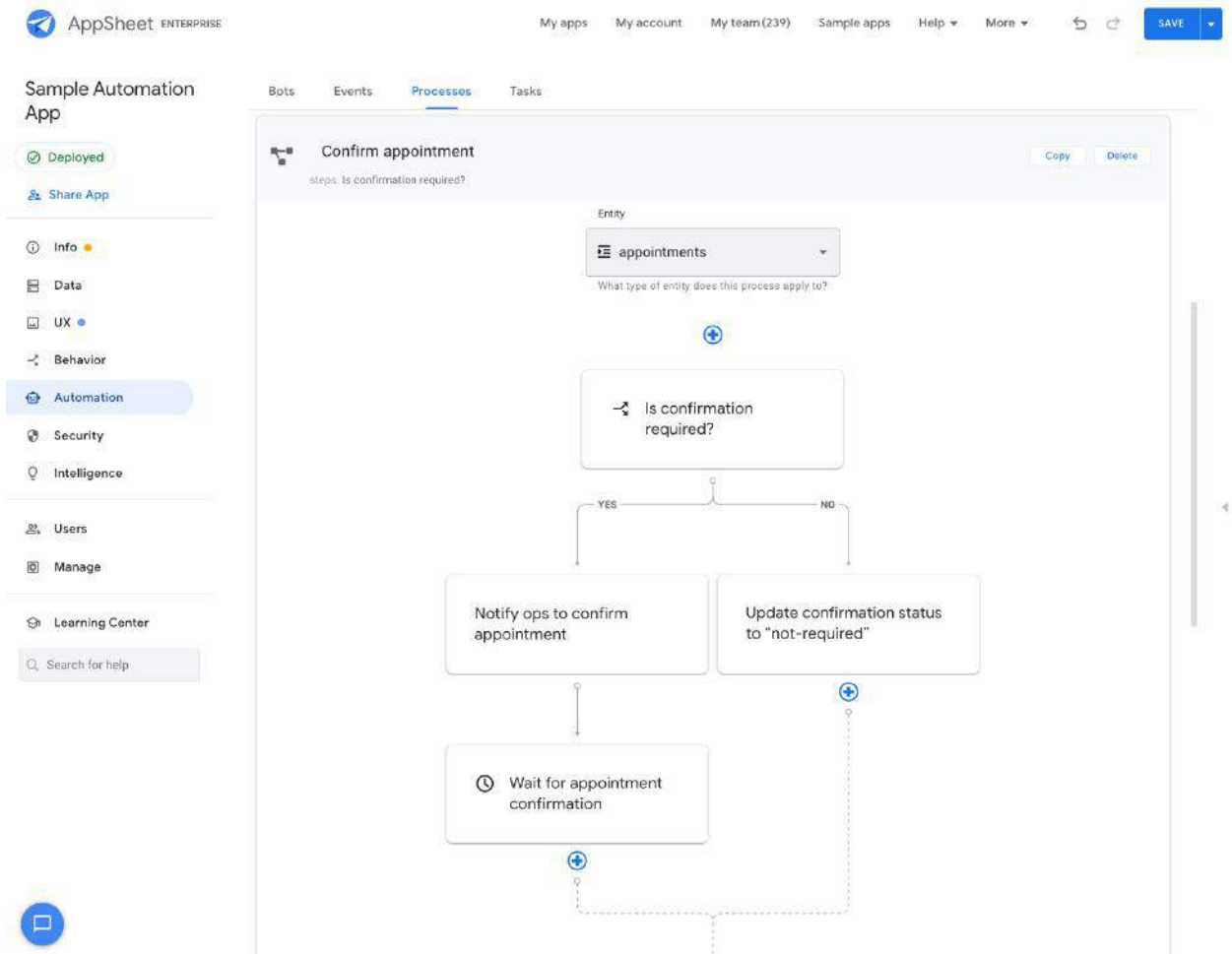
Field name	Value
Task type	Run a task
Task	Create new task
Task category	Change data
Task name	Update confirmation status
<i>Click on “Go to: Task” at the bottom of the pane to configure the rest of the task as described below:</i>	
Table name	appointments
Data change action name	Update confirmation status to “not-required”

The screenshot shows the AppSheet interface with a process flow editor. The process is titled "Confirm appointment" and is currently in the "Processes" tab. The flow starts with an entity selection of "appointments". A decision step asks "Is confirmation required?". If "YES", the flow goes to "Notify ops to confirm appointment", followed by "Wait for appointment confirmation". If "NO", the flow goes to "Update confirmation status to 'not-required'". A right-hand panel shows the configuration for the "Update confirmation status to 'not-required'" step, including options for task category, task name, table name, and data change action name.

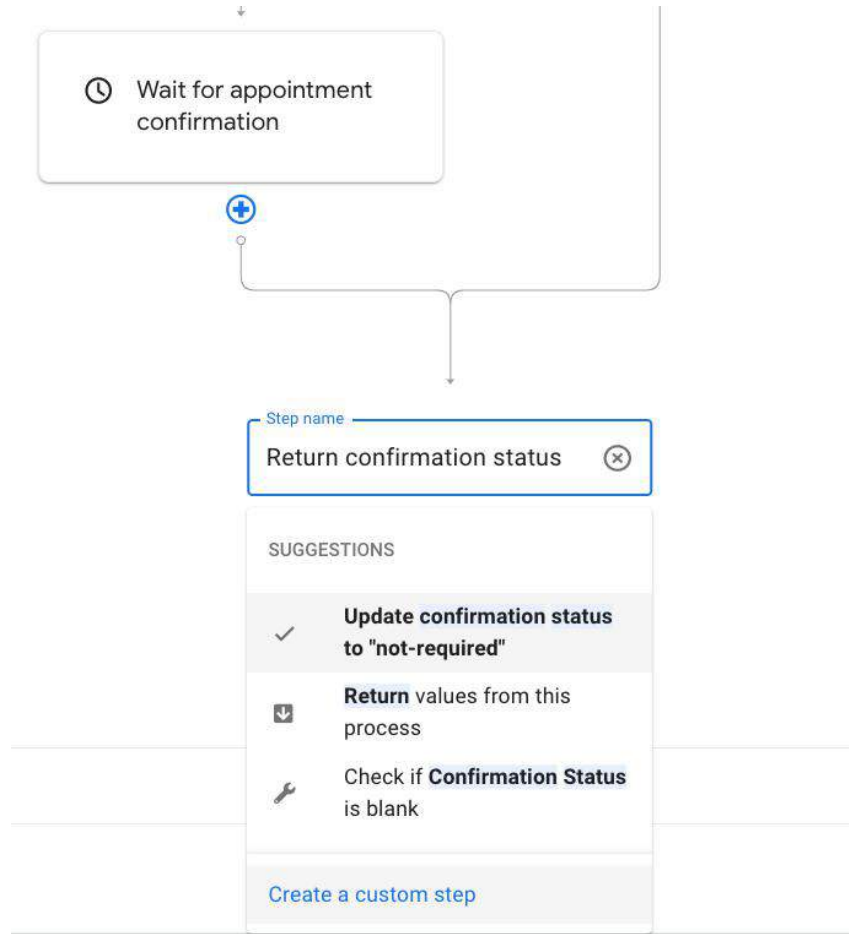
This screenshot shows the configuration panel for the "Update confirmation status" task. It includes the following fields and options:

- Task category:** A list of icons for different task types: AMP (Preview), Send an email, Send a notification, Send an SMS, Change data (selected), HTTP Call a webhook, and Create a new file.
- Task name:** A text input field containing "Update confirmation status".
- Table name:** A dropdown menu set to "appointments". Below it is a link for "View Definition".
- Data Change Action Name:** A dropdown menu set to "Update confirmation status to 'not-required'".

- d. Click **Save** and **Close** to save your changes so far. At this point your process should look like this:



14. Add a final step to return the **confirmation status** value used by the main process to make further decisions.
  - a. Click **+ Add a step** to create a new step.
  - b. Set **Step name** to **Return confirmation status** and click on **Create a custom step**.



- c. In the **Settings pane**, select **Return values** as the task type.
- d. Click on “Add”.
- e. Type `ConfirmationStatus = [Confirmation Status]` in **Return these values**.

The screenshot displays the AppSheet interface for configuring an automation process. On the left, a sidebar lists various app components, with 'Automation' selected. The main workspace shows a process flow diagram with the following steps:

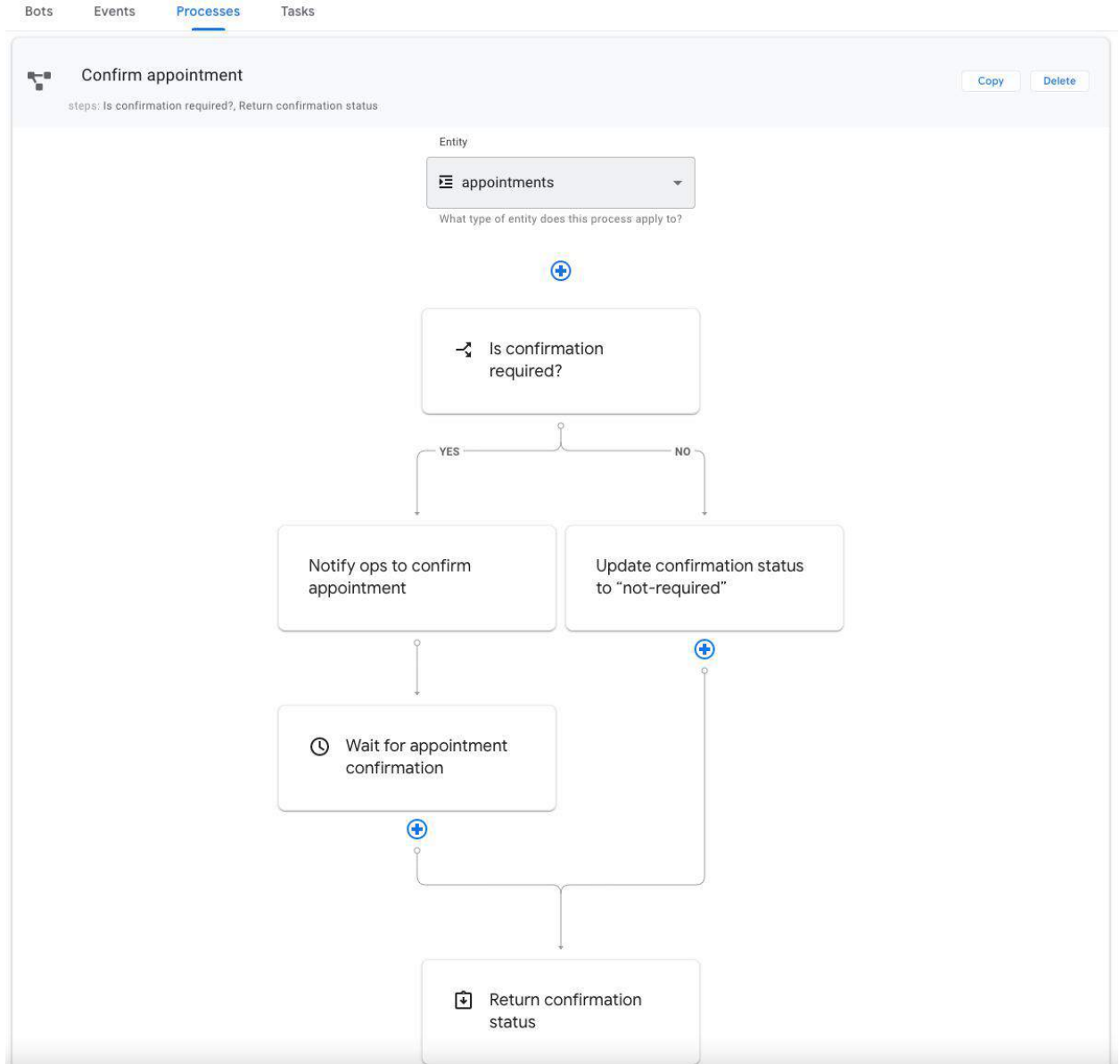
- Is confirmation required?** (Decision step)
- YES** branch leads to **Notify ops to confirm appointment** (Task step)
- NO** branch leads to **Update confirmation status to "not-required"** (Task step)
- Both branches merge and lead to **Wait for appointment confirmation** (Wait step)
- The process concludes with **Return confirmation status** (Return Values step).

The configuration panel on the right for the 'Return confirmation status' step shows:

- Step name:** Return confirmation status
- Return Values:** ConfirmationStat = [Confirms]

Buttons for 'Save', 'Delete step', and 'Close' are visible at the top and bottom of the interface.

f. Click **Save** and then **Close**. Your process should now look like this:

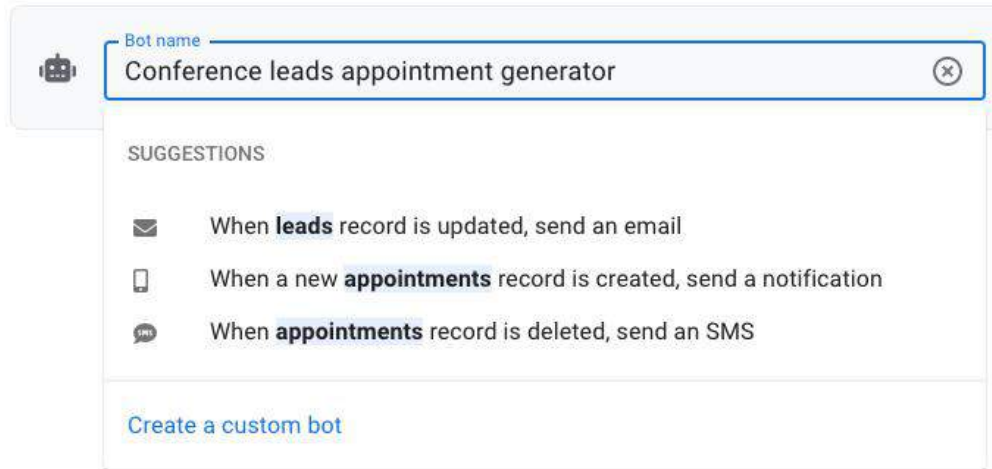


Creation of the Appointment Confirmation Process is complete!

## Creating the bot

Follow the steps below to create a new bot:

1. From the AppSheet UI, navigate to the **Bots** tab and click **New Bot**.
2. Type `Conference leads appointment generator` into the text box and suggested actions appear, as shown in the image below.
3. Select **Create custom bot named** to create the bot.

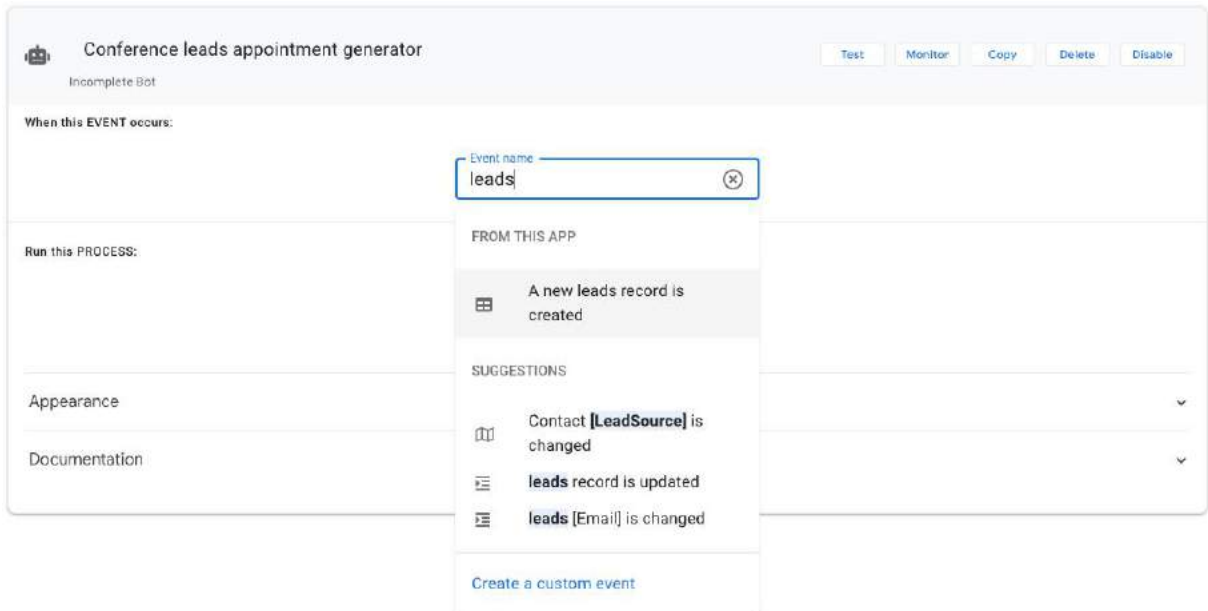


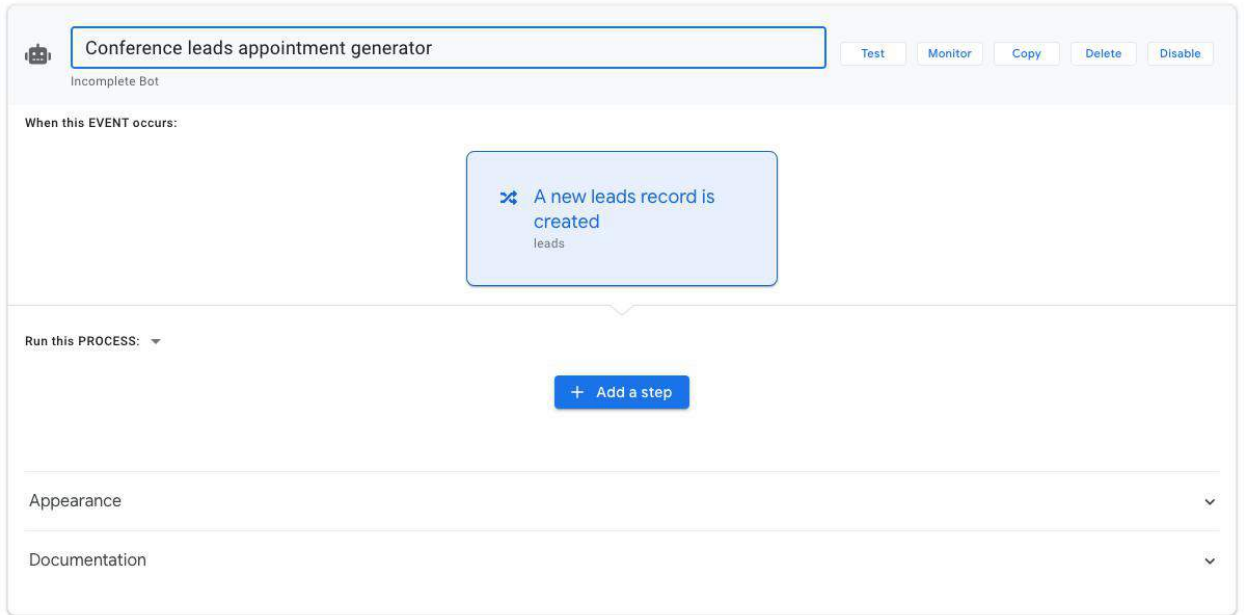
## Configuring the event

Once the bot is created, you can define the event for the bot. In this example, the event trigger is the addition of new leads to the Leads Google Sheet.

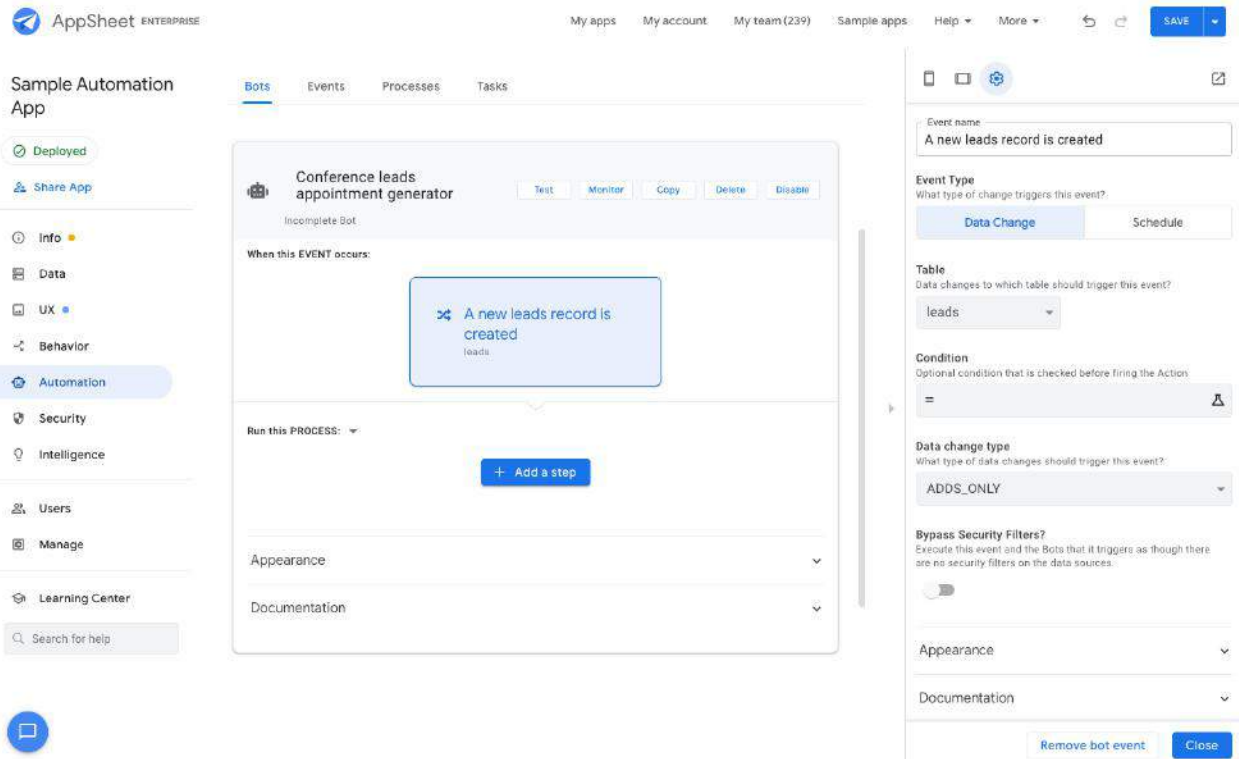
To create an event:

1. From the AppSheet UI, select the bot you created in the previous section.
2. Click **Choose an event**.
3. Type `leads` and suggested events appear.
4. Select **A new leads record is created** from the suggestions.





5. Click on the event and edit the configuration from the right hand panel as follows:



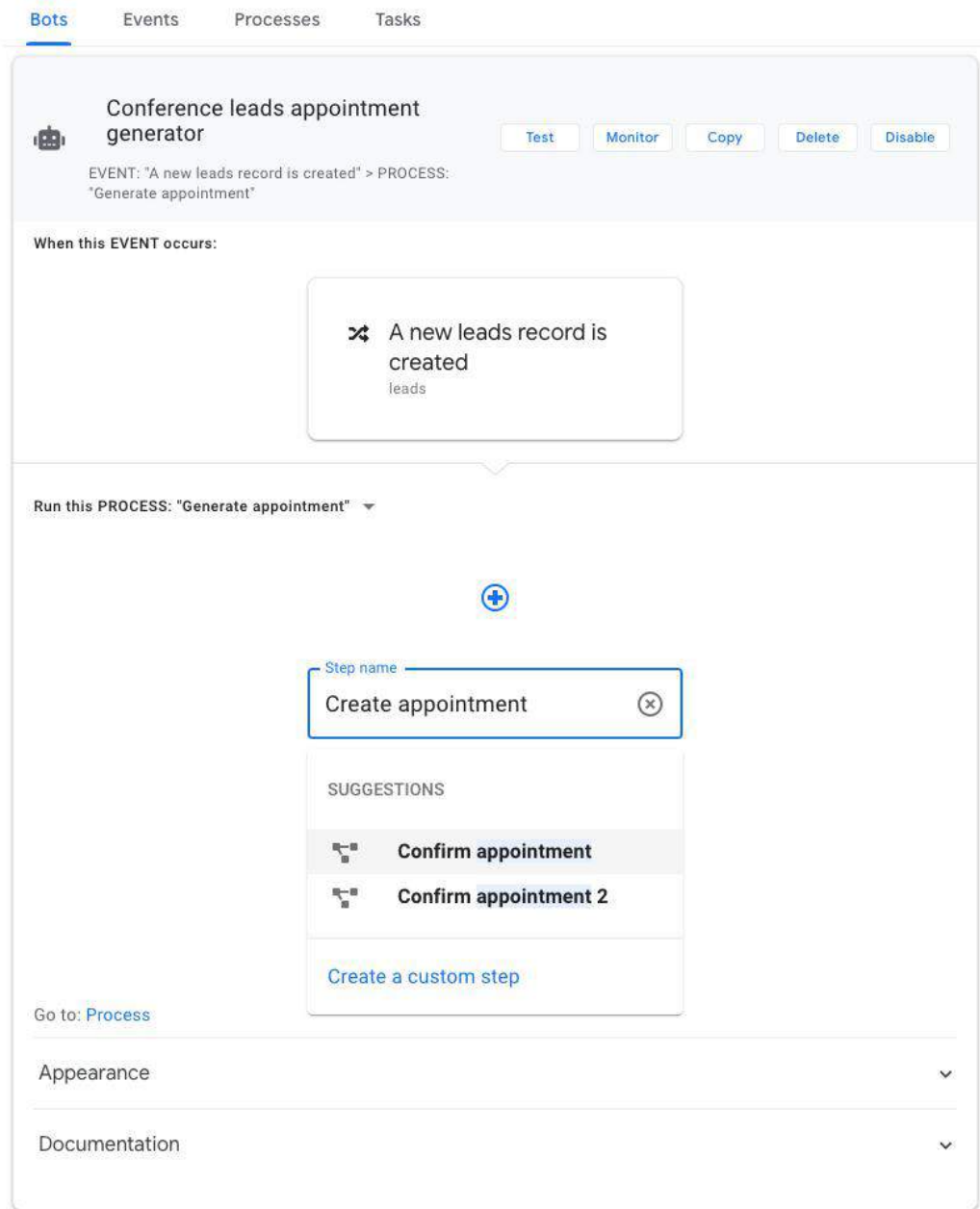
6. Click **Close** and then **Save**.

## Configuring the main process



Configure the process your event should trigger:

1. In the **Run 'New Process'** section of the editor, click **+Add a step**.
2. Type `Create appointment` into the text box and click on **Create a custom step**. This implicitly creates a process.



3. In the **Settings** pane, configure the fields as follows:

Field	Value
-------	-------

Do this	Call a process
Process name	Confirm appointment
Process inputs	<ul style="list-style-type: none"> <li>• Full name: CONCATENATE([First Name], " ", [Last Name])</li> <li>• Email: [Email]</li> <li>• Appointment date: [DATE]+[FollowupAfterDays]</li> <li>• Confirmation required: "SWITCH(UPPER([Profile]), "HIGH", "Y", "N")"</li> </ul>

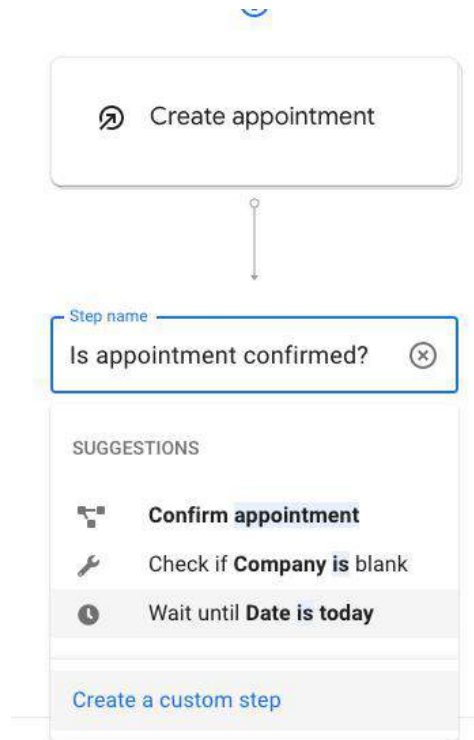
Your step config should look like this:

The screenshot displays the AppSheet 'Automation' editor. The main workspace shows a flow starting with the event 'A new leads record is created' leading to the process 'Generate appointment', which then triggers the 'Create appointment' step. The right-hand pane is open to the configuration for the 'Create appointment' step. The 'Process name' is set to 'Confirm appointment'. Under 'Process inputs', four fields are configured: 'Full Name' with the formula 'CONCATENATE([First Name], " ", [Last Name])', 'Email' with '[Email]', 'Appointment date' with '[DATE]+[FollowupAfterDays]', and 'Confirmation required' with the formula 'SWITCH(UPPER([Profile]), "HIGH", "Y", "N")'. The 'Add' button is visible at the bottom of the configuration pane.

4. Click **Save** and **Done** to save your changes so far. At this point your bot should look like this:

The screenshot shows the AppSheet automation editor interface. At the top, there are navigation tabs for 'Bots', 'Events', 'Processes', and 'Tasks'. The main title is 'Conference leads appointment generator'. Below the title, there are buttons for 'Test', 'Monitor', 'Copy', 'Delete', and 'Disable'. The event trigger is 'A new leads record is created' with the field 'leads'. The process is 'Generate appointment'. The workflow diagram shows a single step 'Create appointment' followed by an 'Add a step' button and an 'END' label. At the bottom, there are links for 'Go to: Process', 'Appearance', and 'Documentation'.

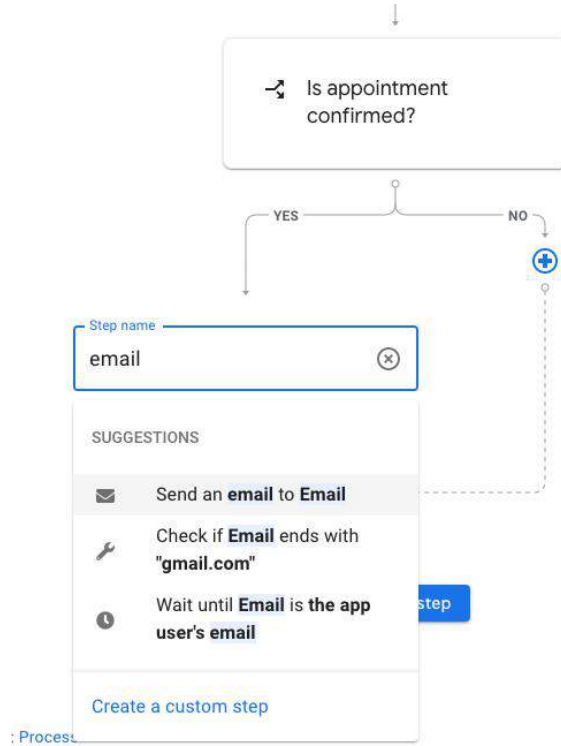
5. Add a branch step to determine if an appointment notification needs to be sent out.
  - a. Click **+ Add a step**.
  - b. In **Step Name**, type `Is appointment confirmed?` and click on **Create a custom step**.



- c. In the **Settings** pane, select **Branch on a condition** for the step type.
- d. Under **Condition to check**, type `OR (UPPER ([Create appointment].[ConfirmationStatus]) = "CONFIRMED", UPPER ([Create appointment].[ConfirmationStatus]) = "NOT-REQUIRED")`. The step configuration should look like this:

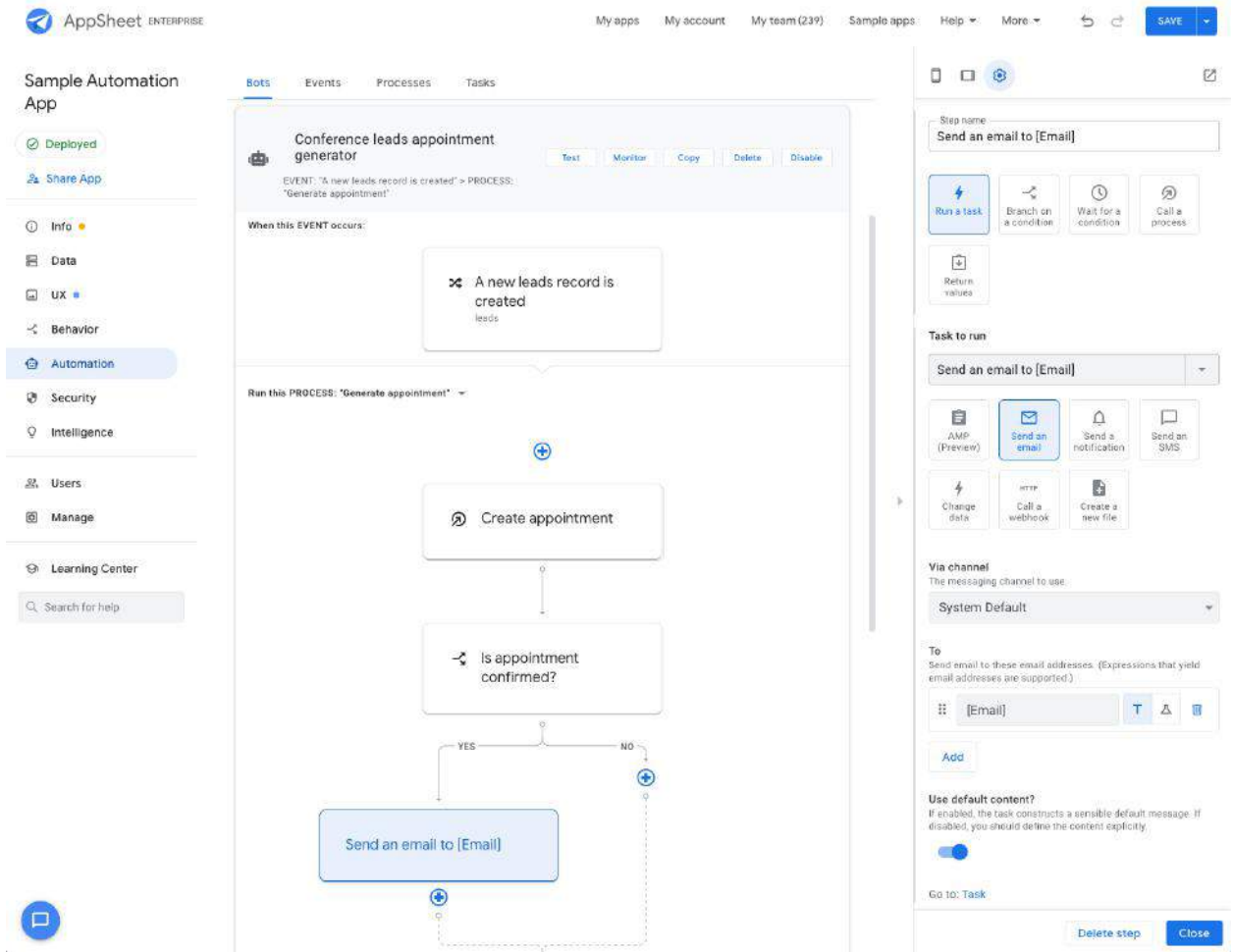
The screenshot displays the AppSheet interface for configuring an automation. On the left, a sidebar lists various app categories like 'Automation', 'Security', and 'Users'. The main workspace shows a flowchart for a process named 'Conference leads appointment generator'. The process begins with an event trigger: 'A new leads record is created'. This is followed by a task 'Create appointment'. A decision step 'Is appointment confirmed?' follows, with a 'YES' branch leading to a step 'Add a step' and a 'NO' branch leading to another 'Add a step'. The right-hand panel provides configuration options for the 'Is appointment confirmed?' step, including a 'Condition to check' field containing the formula '= OR(UPPER([Create appointment]),[Confin])'. Buttons for 'Test', 'Monitor', 'Copy', 'Delete', and 'Disable' are visible at the top of the process configuration area.

6. Add one final step to the process.
  - a. Click the + icon in the **Yes** branch.
  - b. In **Step Name**, type `email` and select the suggested step **Send an email to [Email]**.



**Note:** The platform detects your intent based on the text you enter, correlates that the Leads table has an email address column type and pre-configures the step/task for you.

At this point your step should look like this.



c. Click on **Go to: Task** and configure the remaining fields as follows:

Field	Value
Table name	appointments
To	[Email]
Email subject	Appointment confirmation request
Email body	Dear <<[Full Name]>> Your appointment has been scheduled for <<[Appointment Date]>>.

Leave all other default values.

Bots Events Processes **Tasks**

Send an email to [Email] Copy Delete ↕

**Task category**  
Category of task

AMP (Preview) **Send an email** Send a notification Send an SMS Change data HTTP Call a webhook Create a new file

**Task name**  
Unique name for this task: Send an email to [Email]

**Table name**  
What entity table does this task work against? appointments [View Definition](#)

**Via channel**  
The messaging channel to use. System Default

**To**  
Send email to these email addresses. (Expressions that yield email addresses are supported.) [Email] T △ 🗑️  
Add

**Use default content?**  
If enabled, the task constructs a sensible default message. If disabled, you should define the content explicitly.

---

**Email Content**

**Email Subject**  
Subject line for the email. (Template that yields text is supported. Leave empty for default email subject.) Appointment confirmation request

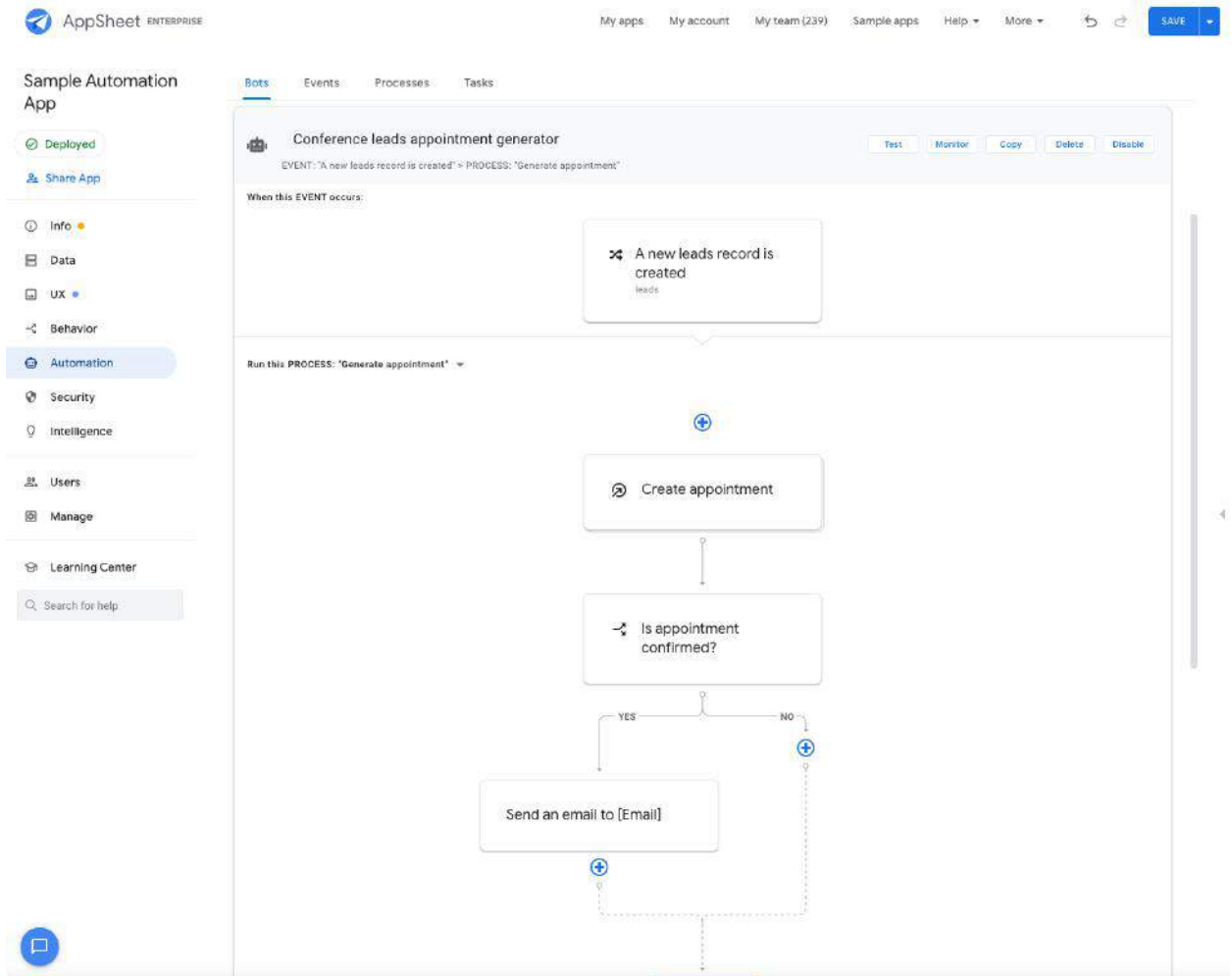
**Email Body**  
The email body. (Template that yields text or HTML is supported. Leave empty for default email body.) Dear <<[Full Name]>> Your appointment has been scheduled for <<[Appointment Date]>>.

**CC**  
CC these email addresses. (Expressions that yield email addresses are supported.) Add

d. Click **Save** and **Done** to save your changes.

7. Your completed bot should look like this:





## Testing your automation

Follow these steps to test your automation:

1. Make sure your bot is **Enabled**.
2. Go to your “Leads” Sheet and enter an entire leads record into a new row.

*Tip: Create a temporary Sheet and add your row of data there, copy it and then paste it into the “Leads” Sheet)*

**Note:** Automation only supports adding a full record at this point. Do not add a partial lead record.

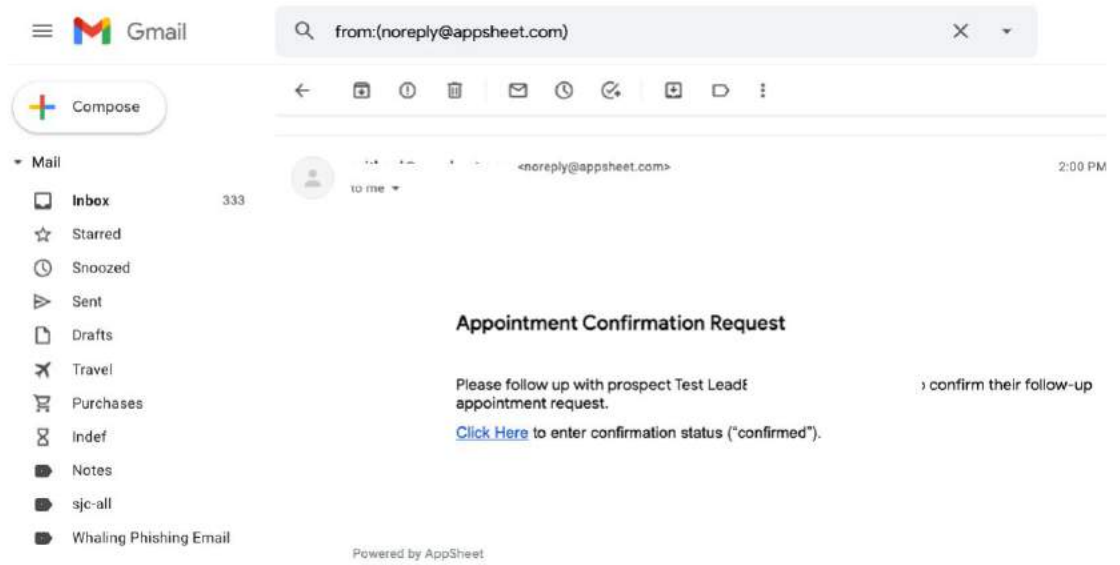
	A	B	C	D	E	F	G	H	I
1	First Name	Last Name	Email	Phone	Company	Date	FollowupRequested	FollowupAfterDays	Profile
2	Test	Lead1		3148889292	Acme Inc	11/1/2020	Y		5 High
3	Test	Lead2		4082224455	Acme Inc	11/1/2020	Y		6 Medium
4	Test	Lead3		9256667788	Acme Inc	11/10/2020	Y		7 Low
5	Test	Lead6		6691112222	Acme Inc	1/6/2021	Y		5 Low
6	Test	Lead7		9256667788	Acme Inc	1/20/2021	Y		5 High
7	Test	Lead8		9256667788	Acme Inc	1/21/2021	Y		7 High
8									

3. Within a few seconds, an appointment record is added to your Appointments Sheet.

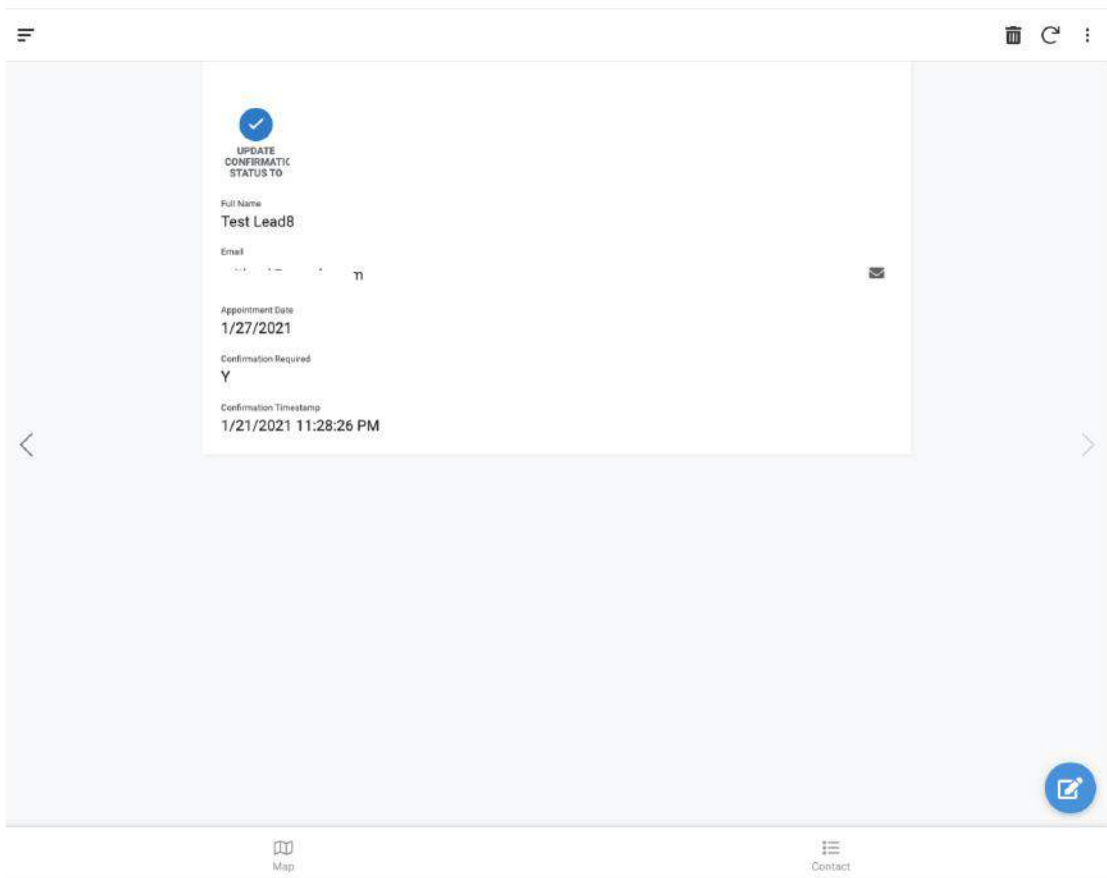
	A	B	C	D	E	F
1	Full Name	Email	Appointment Date	Confirmation Required	Confirmation Status	Confirmation Timestamp
2	Test Lead1	te	11/6/2020	Y	confirmed	1/20/2021 12:34:56
3	Test Lead2	te	11/7/2020	N	not-required	1/19/2021 14:28:43
4	Test Lead3	te	11/17/2020	N	not-required	1/20/2021 18:35:27
5	Test Lead8	te	1/28/2021	Y		1/22/2021 0:07:23
6						

Based upon the process created in previous steps, a new appointment record is only created when the value of the “FollowupRequested” column is “Y” in the Leads Sheet. The “Full Name” field in the Appointment Sheet is a concatenation of the “First Name” and “Last Name” values from the Leads Sheet. The “Appointment Date” in the Appointments Sheet should reflect the date value calculated from the Expression Assistant formula.

4. Because this is a “High” Profile lead the ops team should received an email at the email address in the Sheet



5. Click **Click Here**.
6. When the form opens, on the form, click the **pencil** icon.
7. In **Confirmation Status**, type `confirmed`.
8. Click **Save**.



A screenshot of an AppSheet form with a back arrow in the top left corner. The form contains the following fields:

- Full Name\***: Test Lead8
- Email\***: [redacted]@m
- Appointment Date\***: 01/27/2021 (with a calendar icon)
- Confirmation Required\***: Y
- Confirmation Status**: confirmed
- Confirmation Timestamp\***: 01/21/2021, 11:28:26 PM (with a calendar icon)

At the bottom of the form are two buttons: "Cancel" and "Save".

9. Once the record updates, a second (appointment confirmation) email is sent to the email address.

## Invoice Extraction Automation

This brief tutorial shows you how to configure an Invoice Extraction Automation workflow. In the GA release of AppSheet Automation, there are three supported document type extraction models for document automation: invoices, receipts, and W9 documents. Additional supported document type extraction models are planned for subsequent releases.

The tutorial below walks through the steps required to automate extraction of invoice documents. However, the same steps can be used to configure extraction automation for all supported document types.

## Creating Invoice Table from Folder Data Source

In the following steps, you can:

- Create a table for storing invoice data extracted from *invoice* document types.
- Implicitly configure a table (not visible to users) representing a Google Drive folder containing the *invoice* documents to be automatically extracted.

Once an invoice document is processed, users can move those documents to another folder. In a future release, users can configure a process to automatically move the processed documents to another folder.

To create a table for document extraction:

1. Prepare a Drive folder with invoices. You can obtain some sample invoices [here](#) and upload them to a Google Drive folder of your choice.
2. In the Data section of the application using the extraction, under the **Table** tab, select **New Table**.
3. If you do not have **Google** as a configured data source, click **+** and connect **Google** as a source.
4. Select **Documents on Google Drive**.
5. Under **Documents**, select **Invoices**.
6. Select the Drive folder where you placed the sample invoices and click **Select**.

**Document Sources:**  
AppSheet can extract structured data from unstructured documents (folders that contain invoices, receipts, etc), automating document processing and saving you time.

**Documents:**

Choose one of these supported document types and AppSheet will automatically read and process the information for you.

**Folder:**

Connect to files within a Google folder. AppSheet will provide a table with the folder contents and file metadata to use within your application.

**Source Invoice Folder**  
Connect and extract invoices from a Google Drive folder

**Automatic Invoice Detection**  
New invoices added to this folder will automatically be read and processed.

**Table name**  
Name for this table (user-visible)

[Learn more](#)

7. Select **Create Table**. This creates two tables:

- Invoice Files
- Invoice Files\_line\_Item

## File Processing

Once the table is created, invoice processing begins automatically. The invoices in the folder when the table is created are processed first. If new documents are added to the folder at any point while the table is included as part of the application, the new documents are automatically processed.

**Note:** Processing may take some time, depending on the number of invoices in the folder. In the GA release, the UI shows a loading indicator until the processing is completed. It is recommended to create your table initially with a small number of files in it.

Upon successful completion, the screen looks like this:

The screenshot displays the AppSheet configuration interface. On the left, the configuration for the 'Invoice Files' table is shown, including options for updates, adds, and deletes. Below it, the configuration for the 'Invoice Files\_line\_item' table is visible. On the right, a preview of the 'Invoices' table is shown, displaying a list of invoice records with columns for Invoice Number, Invoice Date, and Total Amount.

Invoice Number	Invoice Date	Total Amount
	1/14/2021	\$2,420.00
1001-MODEM	1/14/2021	\$1,020.00
1A33ADF2	1/13/2021	\$2,112.00
00189	7/12/2018	\$147.49
1588-2	5/9/1984	
0089546	4/8/2018	\$1,564.81
261309	9/10/2018	\$35.00

There are some limitations to keep in mind while using this capability. These limitations are likely to change over time as the service matures.

**Note:** *Limitations:*

- *There is a quota per month for documents and depends on your subscription plan. Please check the [pricing page](#) for additional details.*
- *Documents need to be under some thresholds to be processed. Otherwise the file processing fails. Please review the [platform limits](#) for “Size of document” and “Number of pages in a document” limits.*
- *The only extension types supported in GA are `.tiff`, `.gif`, `.pdf`, `.jpg`, `.png`*

## Adding Application Views

As with any table, views can be added to the application using document extraction tables. However, in document-based workflows, App Editors should be careful not to change the underlying table schema.


A few views are automatically created:

- Invoice\_Files\_Details
- Invoice\_Files\_Form
- Invoice Files\_line\_item\_Detail
- Invoice Files\_line\_item\_Form
- Invoice Files\_line\_item\_Inline




## Ref Views

### Invoice Files



**Invoice Files\_Detail**

data: Invoice Files type: detail



**Invoice Files\_Form**

data: Invoice Files type: form

### Invoice Files\_line\_item



**Invoice Files\_line\_item\_Detail**

data: Invoice Files\_line\_item type: detail



**Invoice Files\_line\_item\_Form**

data: Invoice Files\_line\_item type: form



**Invoice Files\_line\_item\_Inline**

data: Invoice Files\_line\_item type: table

## Low Confidence Extraction Handling

In the event that an extraction confidence score is below the threshold considered successful, a user should review the results and adjust the values if needed.

The information about the low quality documents, including which field(s) triggered the review, are logged to your applications audit logging. In the Details of the audit log include urls to your document file, as well as the row in a Sheet that requires human review. Once you have addressed the potential issues, change the IsEdited column to TRUE for that document.

**Note:** *Your document data will not appear in your application until you have reviewed the information and updated the IsEdited column.*

## Failure Handling

In some cases, a complete failure may occur during document extraction. Common causes

include attempts to extract documents that are totally illegible (i.e written in cursive handwriting), written in languages not supported by our AI services, in an incorrect file format, or that are of the incorrect document type.

In the event that the extraction fails, the extraction results logged to your applications audit logs with the details about the reason for failure. As in the low confidence condition, you can manually enter data into the sheet and once done, set the `IsEdited` column to `TRUE`. If the failure is not resolvable (e.g. wrong document) simply leave the row as it is and it will not appear in your application.

## Folder Contents as a Table

This brief tutorial shows you how to configure an application that can represent files in a Google Drive folder as a table in AppSheet. This "folder-backed" table can be further utilized by AppSheet with *views* and *slices*. The ability to use folder contents as a table provides a simple mechanism for including files in your application.

The following steps demonstrate how to:

- [Creating an app using the contents of a Drive folder](#)
- [Creating a table from a folder data source](#)
- [Creating a useful view for your app](#)
- [Creating slices to filter data](#)
- [Adding virtual columns](#)
- [Linking files to other data](#)











## Creating an App

This [ZIP file](#) contains a folder with a Sheet and images detailing information about inventory in an office setting. To begin, download the ZIP file and extract the contents. Each inventory item is represented by the following pieces of data:

- Name
- Location
- Type
- Model identification number

When you create a new app with this Sheet, you may see an error message about the location base image. To resolve the error, edit the location column and set the base image to "<https://storage.googleapis.com/geodata-appsheet-demos/office%20floorplan.jpg>", as shown below.

Tables Columns Slices User Settings

	NAME	TYPE	KEY?	LABEL?
1	 _RowNumber	Number	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	 item id	Decimal	<input type="checkbox"/>	<input type="checkbox"/>
3	 name	Name	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4	 description	Text	<input type="checkbox"/>	<input type="checkbox"/>
5	 image	Image	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6	 category	Enum	<input type="checkbox"/>	<input type="checkbox"/>
7	 location	XY	<input type="checkbox"/>	<input type="checkbox"/>
8	 status	Text	<input type="checkbox"/>	<input type="checkbox"/>
9	 last date inspected	DateTime	<input type="checkbox"/>	<input type="checkbox"/>
10	 last inspected by	Email	<input type="checkbox"/>	<input type="checkbox"/>

**assets : location** Done

type: XY

---

**Column name**  
Column name:

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

**Type**  
Column data type:

---

**Type Details**

**Optional Url for KML File**

**Background image for the XY coordinates**

---

**Data Validity**

---

**Auto Compute**

---

**Update Behavior**

Once you have edited the location, the app should appear as follows:



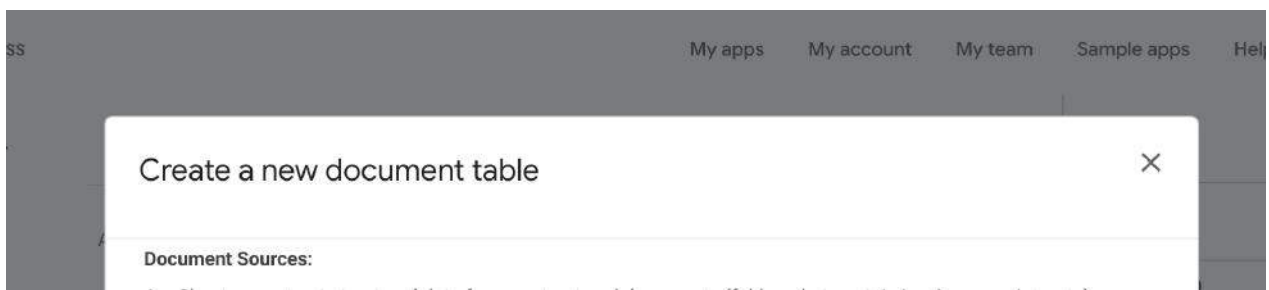
The app should have created the following views:

- *Map and assets* UX view
- *assets\_Detail* view
- *assets\_Form* view

## Creating a Table from Folder Data Source

To create a table for document extraction:

1. Prepare a Drive folder with files. For this example, you can obtain some sample files here (content is not important, only the names) and upload them to a Google Drive folder of your choice.
2. In the Data section of the app, select **Table > New Table**.
3. If you do not have **Google** as a configured data source, click **+** and connect **Google** as a source.
4. Select **Documents on Google Drive**.
5. Select **Documents > Collection of files**.
6. Located the Drive folder where you placed the sample invoices and click **Select**. Drive



7. Select **Create Table**. This will create a table named *User manuals*. The table name is, based on the folder name selected.

Once the table is created, a set of metadata available for files in the folder is provided, as shown below:

	NAME	TYPE	KEY?	LABEL?
1	<input type="text" value="_ID"/>	<input type="text" value="Text"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2	<input type="text" value="Path"/>	<input type="text" value="Text"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input type="text" value="File"/>	<input type="text" value="File"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	<input type="text" value="CreateTime"/>	<input type="text" value="DateTime"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	<input type="text" value="LastModifiedBy"/>	<input type="text" value="Email"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Note:** In the release, the table is read-only. You can upload and delete files directly from the app in future releases.

Additionally, eventing support is not available. This means it is not possible to couple events on the folder with automation action. Eventing support is planned for a future release.

## Creating a useful view

As with any table, views can be added to apps created with folder-backed tables. However, App Editors should be careful not to change the underlying table schema. Two views are

automatically created:

- User\_Manuals\_Details
- User\_Manuals\_Form

The two most common views of the table enable users to:

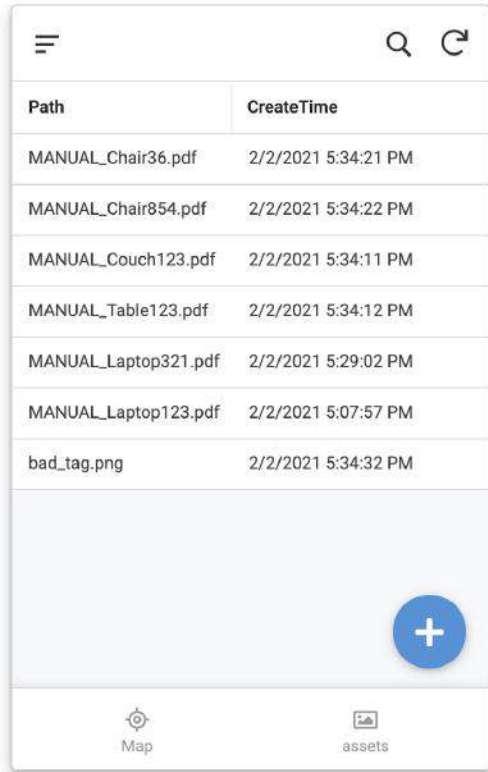
- List files
- Click on the list to open a file

You can create a file viewer by creating a new UX view. For simplicity, you can create the UX view as a table view, using **Path** and **Creation Date** as the two columns, as shown below:

The screenshot shows the configuration interface for a table view in AppSheet. At the top, there are three view style options: 'form', 'onboarding', and 'card'. Below these is a 'Position' section with a description 'Where the button to access this view is located.' and a row of buttons: 'left most', 'left' (selected), 'center', 'right', 'right most', and 'menu'. A 'ref' field is located below the position buttons. The 'View Options' section is expanded and contains the following settings:

- Sort by:** Sort the rows by one or more columns. An 'Add' button is present.
- Group by:** Group rows by the values in one or more of their columns. An 'Add' button is present.
- Group aggregate:** Display a numeric summary of the rows in each group. The dropdown is set to 'NONE'.
- Column order:** Display columns in a different order than they appear in the original data. Two columns are listed: 'Path' and 'CreateTime', each with a dropdown arrow and a trash icon. An 'Add' button is present below the list.

This view displays to app users like this:



Path	CreateTime
MANUAL_Chair36.pdf	2/2/2021 5:34:21 PM
MANUAL_Chair854.pdf	2/2/2021 5:34:22 PM
MANUAL_Couch123.pdf	2/2/2021 5:34:11 PM
MANUAL_Table123.pdf	2/2/2021 5:34:12 PM
MANUAL_Laptop321.pdf	2/2/2021 5:29:02 PM
MANUAL_Laptop123.pdf	2/2/2021 5:07:57 PM
bad_tag.png	2/2/2021 5:34:32 PM

## Creating Slices to filter data

You may wish to create data slices for your app. For example, you can create a slice that only shows the `.pdf` files with the prefix `MACHINE`.

To do this, filter for files where the **Path** field contains the prefix `MANUAL_` and the suffix `.pdf`. Use the following query:

```
AND(FIND(".pdf", LOWER([Path])) > 0, FIND("MANUAL",[Path]) > 0)
```

The resulting file list displays the slice containing only the machine files. For example, if the view is changed to target this slice, the `bad_tag.png` file is no longer listed.



Path	CreateTime
MANUAL_Chair36.pdf	2/2/2021 5:34:21 PM
MANUAL_Chair854.pdf	2/2/2021 5:34:22 PM
MANUAL_Couch123.pdf	2/2/2021 5:34:11 PM
MANUAL_Table123.pdf	2/2/2021 5:34:12 PM
MANUAL_Laptop321.pdf	2/2/2021 5:29:02 PM
MANUAL_Laptop123.pdf	2/2/2021 5:07:57 PM

## Adding Virtual Columns

In addition to slices, you can add virtual columns to the table. For example, you can create a virtual column that only displays files with the structure `MANUAL_[ID].[extension]`. To create a virtual column to extract the ID content, use the following expression in the **App formula** field:

```
IF(AND(FIND(".pdf", LOWER([Path])) > 0, FIND("MANUAL",[Path]) > 0),
MID([Path], 8, LEN([PATH])-11), "")
```

**User Manuals : New Virtual Column (virtual)**

type: Text formula: =IF(AND(FIND(".pdf",...))

Delete Done ↕

---

**Column name**  
Column name

Manual ID|

---

**App formula**  
Compute the value for this column instead of allowing user input.

= IF(AND(FIND(".pdf", LOWER([Path])) > 0, FIND("MANUAL",[Path]) > 0)

---

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

---

**Type**  
Column data type

Text ▼

---

**Type Details**

**Maximum length** - +

**Minimum length** - +

^

---

**Data Validity**

▼

## Linking Files to other Data

With the slices and virtual column created, you can link the User Manual for each machine by Part ID to the data in the spreadsheet. For example, add a virtual column to the assets table using the configuration below:

**assets : New Virtual Column (virtual)** Delete Done

type: File formula: =LOOKUP([name], "Use...

---

**Column name**  
Column name

**App formula**  
Compute the value for this column instead of allowing user input.  🔗

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

**Type**  
Column data type  ▼

---

**Type Details** ^

**Image/File folder path**  
Folder path where images or files are saved (only respected by some data sources). Leave blank for default behavior.  🔗

---

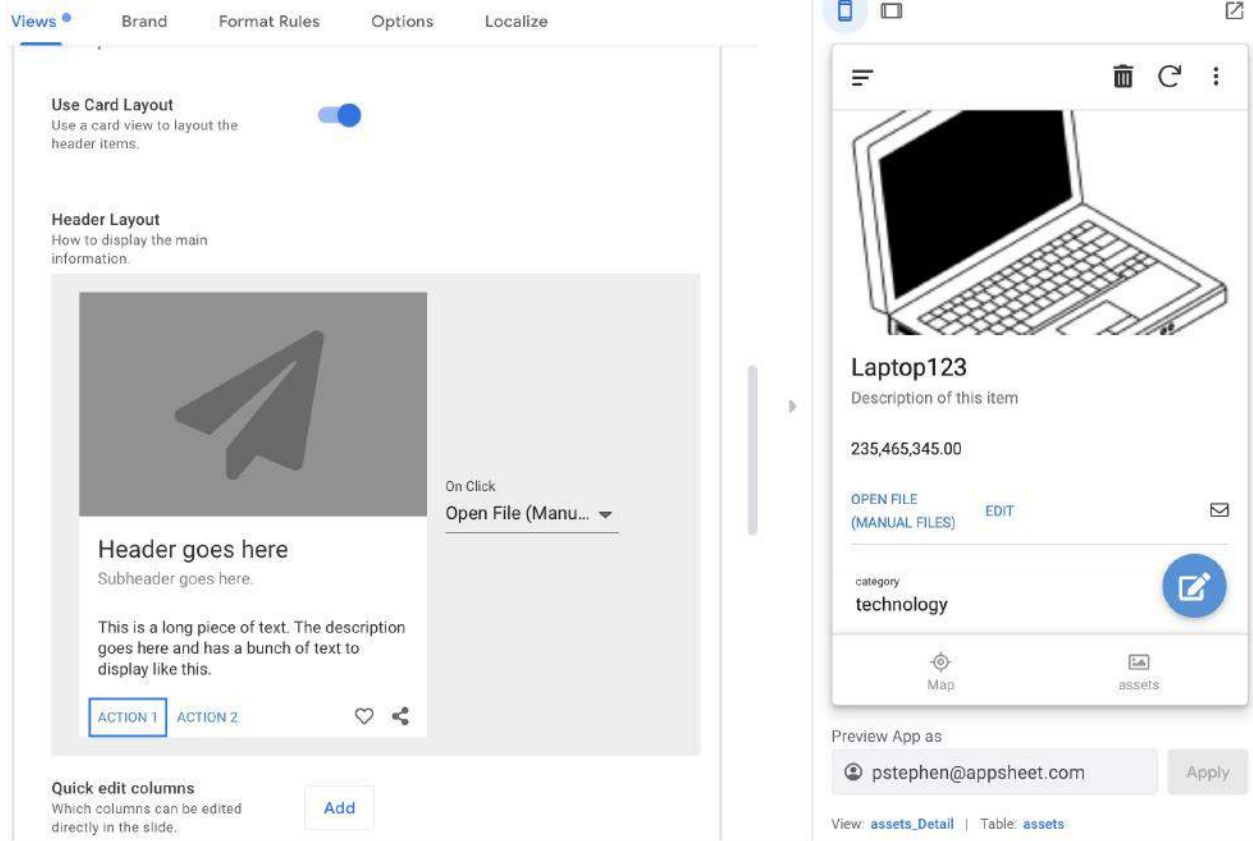
**Data Validity** ▼

---

**Display** ▼

To make this file accessible in the detailed view for each inventory item:

1. Select **Use Card Layout**.
2. Change **ACTION 1** from **Delete** to **Open File (User Manuals)**.

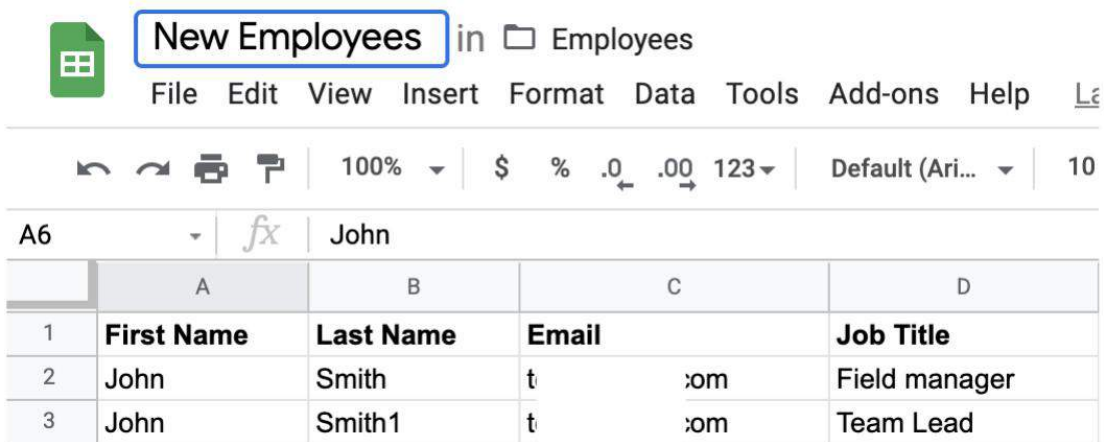


## Appendix

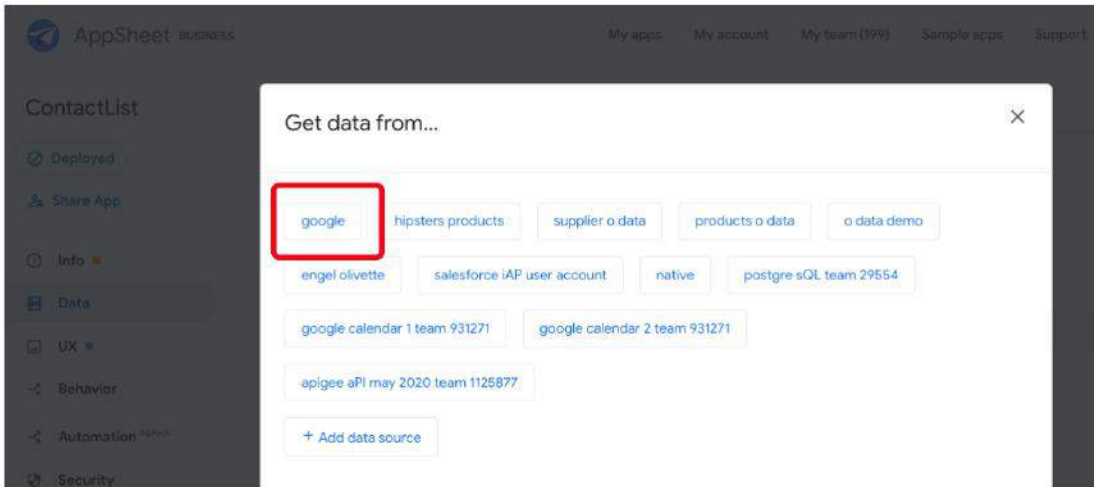
### Configuring Employees entity from Google Sheets datasource

To configure your “employees” entity from Google Sheets:

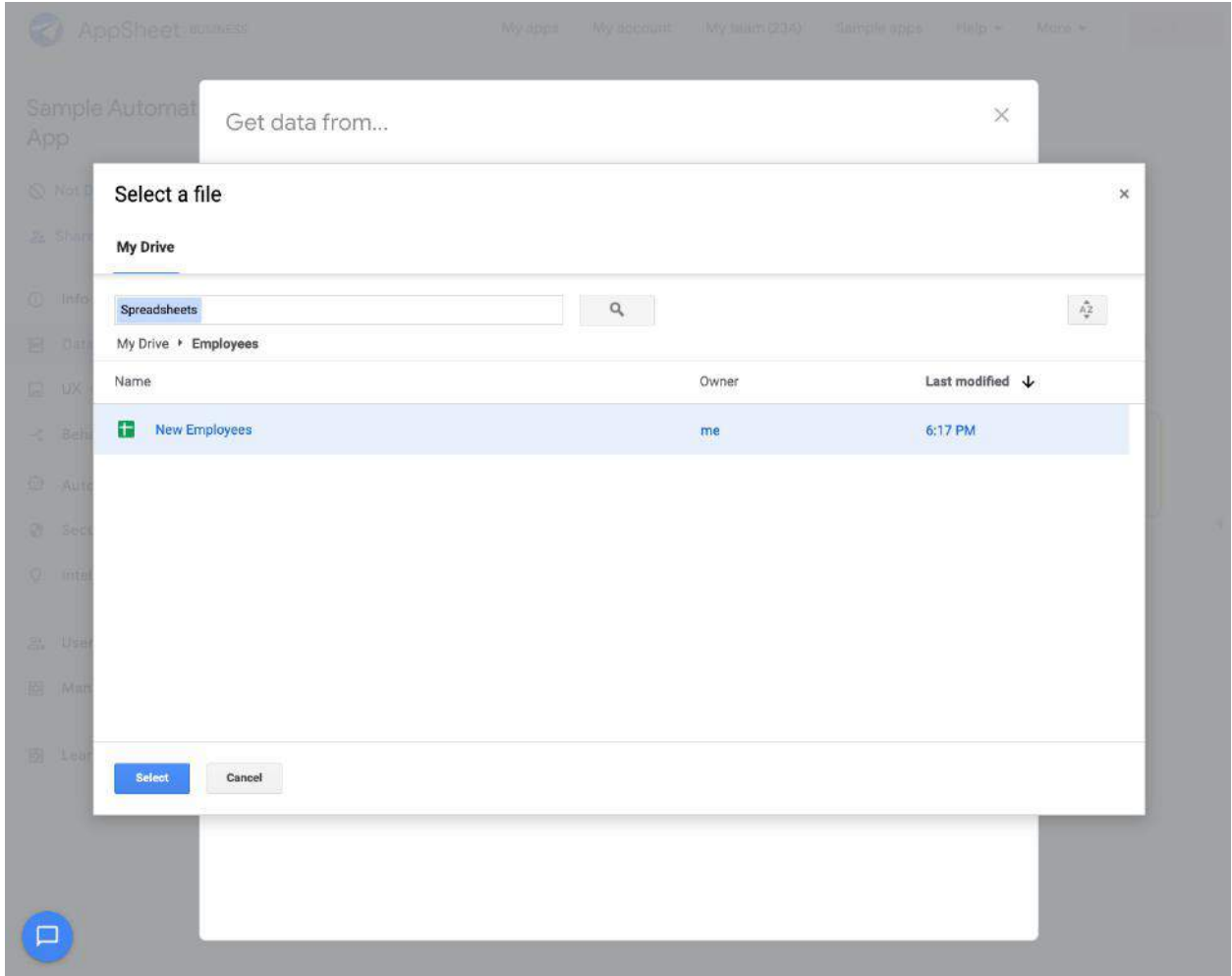
1. Create a new sheet called “New Employees” like this

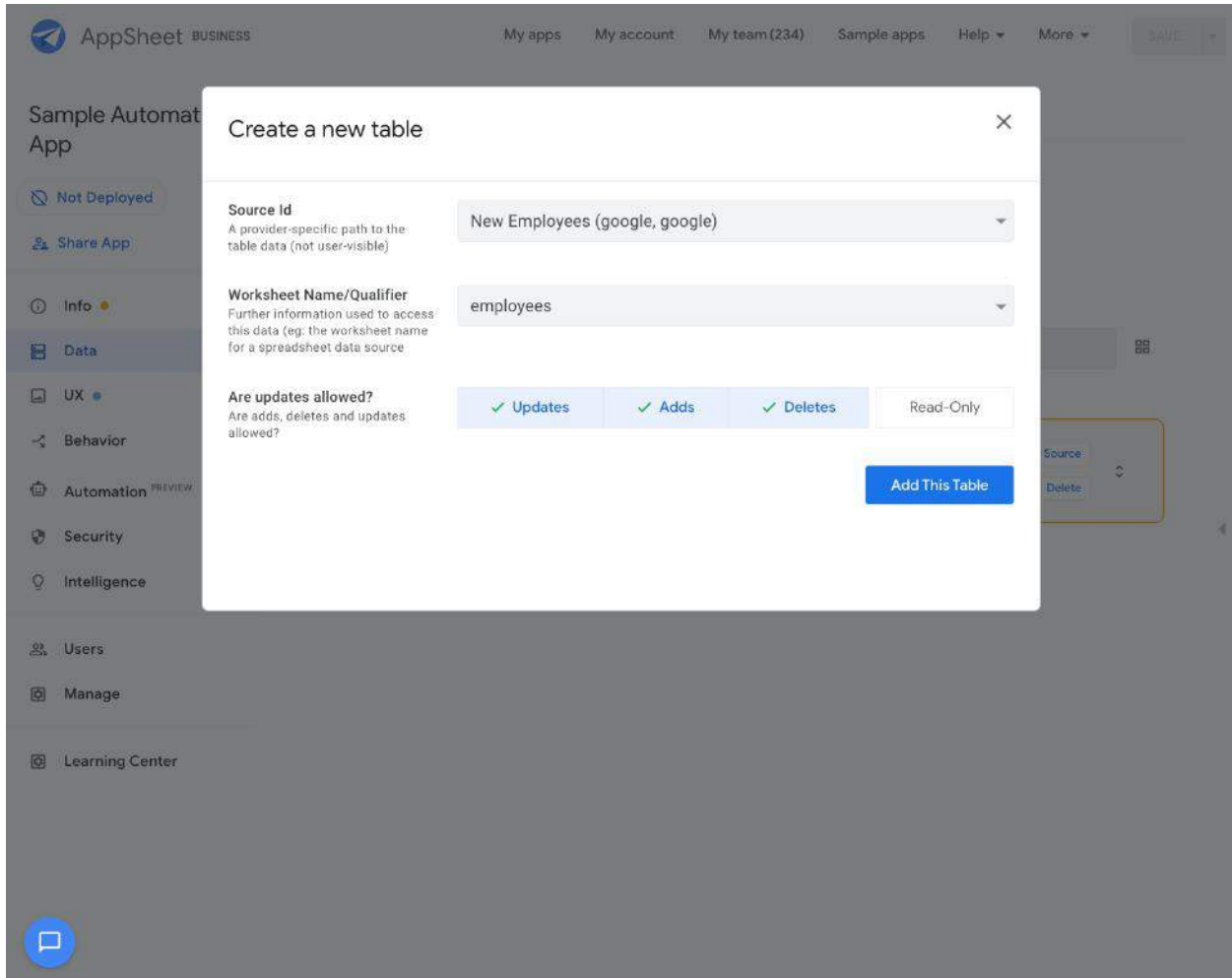


2. From the AppSheet UI, click **Data** in the left navigation bar.
3. Click **New Table**.
4. Select your configured Google Data Source.



5. Use the file browser to select the appropriate **New Employees** Sheet as the data source.





6. Select **employees** as the table for your entity.

## Configuring the AppSheet Events add-on for Google Sheets

Follow the steps below to configure the AppSheet Events add-on for Google Sheets:

1. Navigate to:  
[https://gsuite.google.com/marketplace/app/appsheets\\_events/592572205846](https://gsuite.google.com/marketplace/app/appsheets_events/592572205846)
2. Click **Install** to install the add-on. Follow the authorization steps and authorize.

**Note:** Make sure you are only logged into one Google account in your browser or that you are using an Incognito tab for this step. This is only required for the API keys authorization process. Once this step is complete, you can use the sheet in a different window with multiple accounts logged in.

Follow the steps below to configure the permissions and authorization required for your Sheet to communicate with AppSheet:

1. Open the Google Sheet you want to use for automation.
2. Click **Add-ons > Appsheet Events** in the main menu.
3. Click **Update App Keys** and authorize if necessary
 

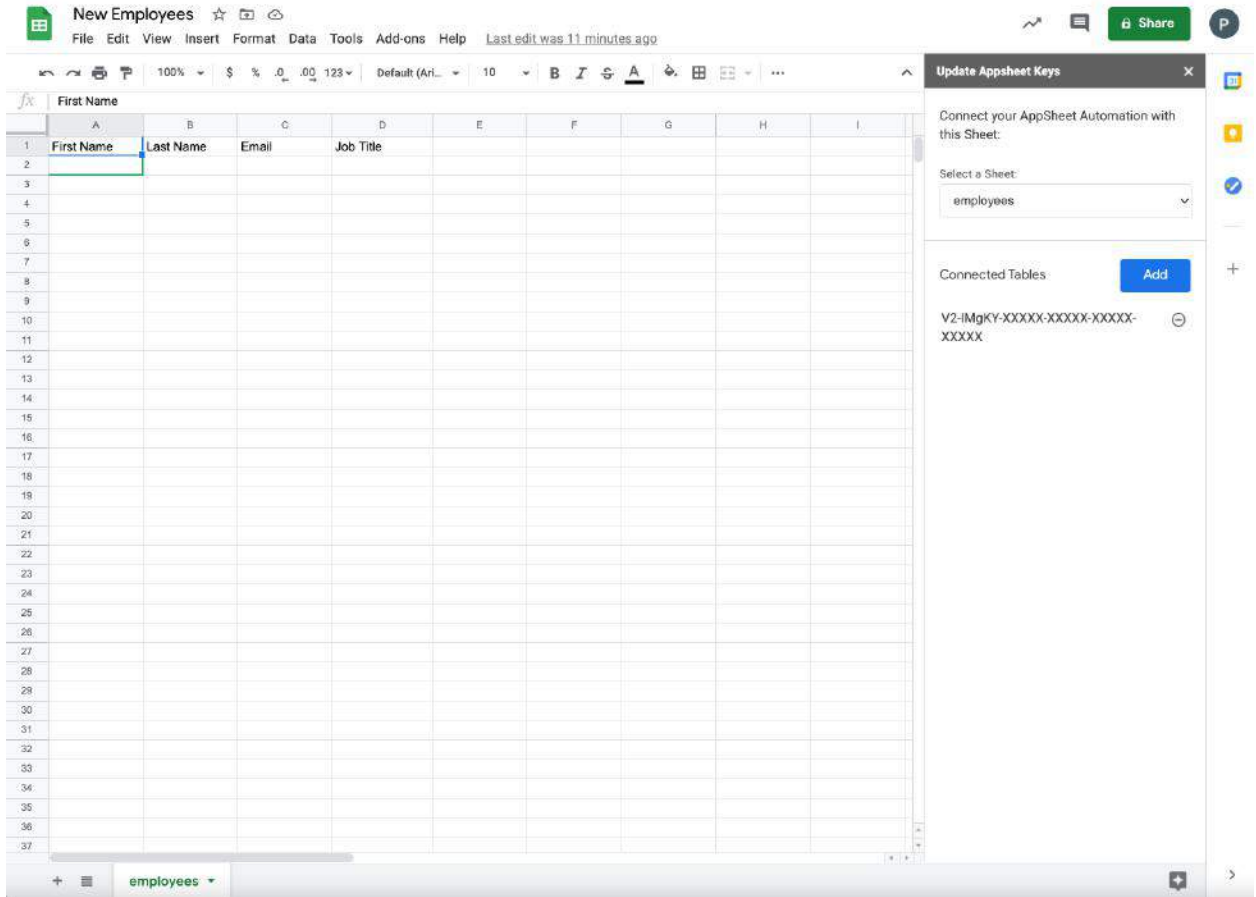
**Note:** *If the current Sheet is not automatically selected, and there are no Sheets in the drop down, please ensure you are only logged into Google account or you are using an Incognito tab.*
4. If the selected Sheet is not the one you intend to link, change the Sheet in the drop-down.
5. Click **Add**.
6. Obtain an API Key:
  - a. Log in and open your app in AppSheet UI.
  - b. Click **Manage** in the side navigation.
  - c. Select the **Integrations** tab.
  - d. Click on the **IN: from cloud services to your app** to enable your Sheet to communicate with your app.
    - i. Click the **Enable** toggle to enable the integration, if it is not enabled.
    - ii. Under **Application Access Keys**, check to ensure the access key is not expired.
 

**Note:** *If the key is valid, you should see a message like this “This access key will stop working on <YYYY-MM-DD>” containing a future date.*
    - iii. If the key is expired, click on **Create Application Access Key** to generate a new key.
    - iv. Click **Show Access Key** to display the valid key string. For example, “V2-itrnz-sF0Z5-MQxNn-TPrB2-RCaIr-x7qdV-JYct6-QoXiy”
    - v. Copy the access key into a notepad. The key string is used in a later step.
7. Go back to the Sheet and paste the API key in the text box.
8. Click **Add**. The API key should now appear in the list. The key is masked.
9. Your Sheet is now linked to your app for events.

Notes:

- Adds and Updates are supported. We do not support deletes.
- For your appsheet table, your primary key should be the first column of your google sheet, or it should include the first column of your google sheet (in the case of a composite key).
- There is at least a 20 second delay between events.
- Events do not trigger if the file is open in read-only (view or comment) mode.

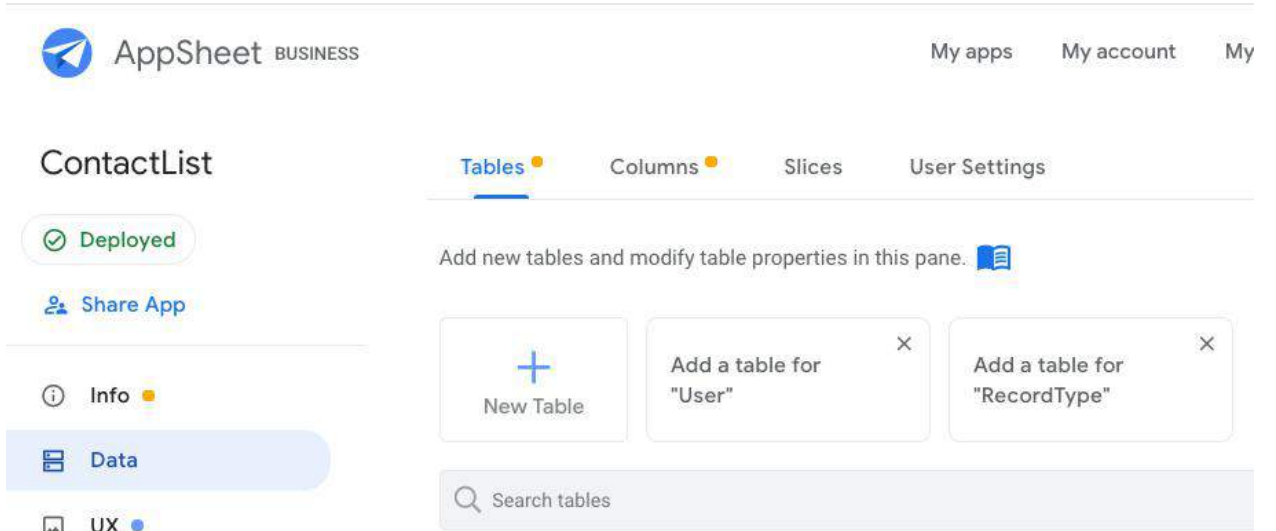




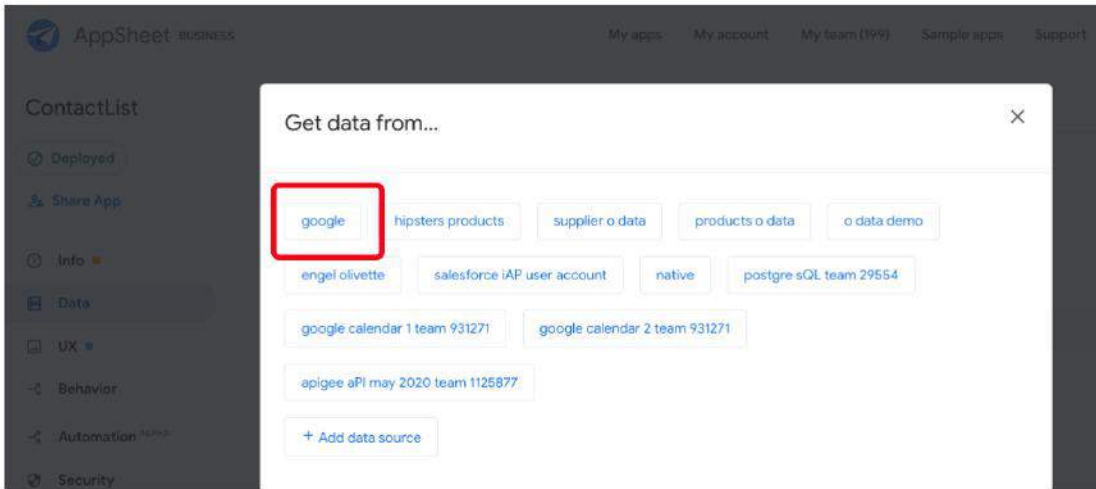
## Configuring Company Contacts entity from Google Sheets datasource

To configure your “Company Contacts” entity from Google Sheets:

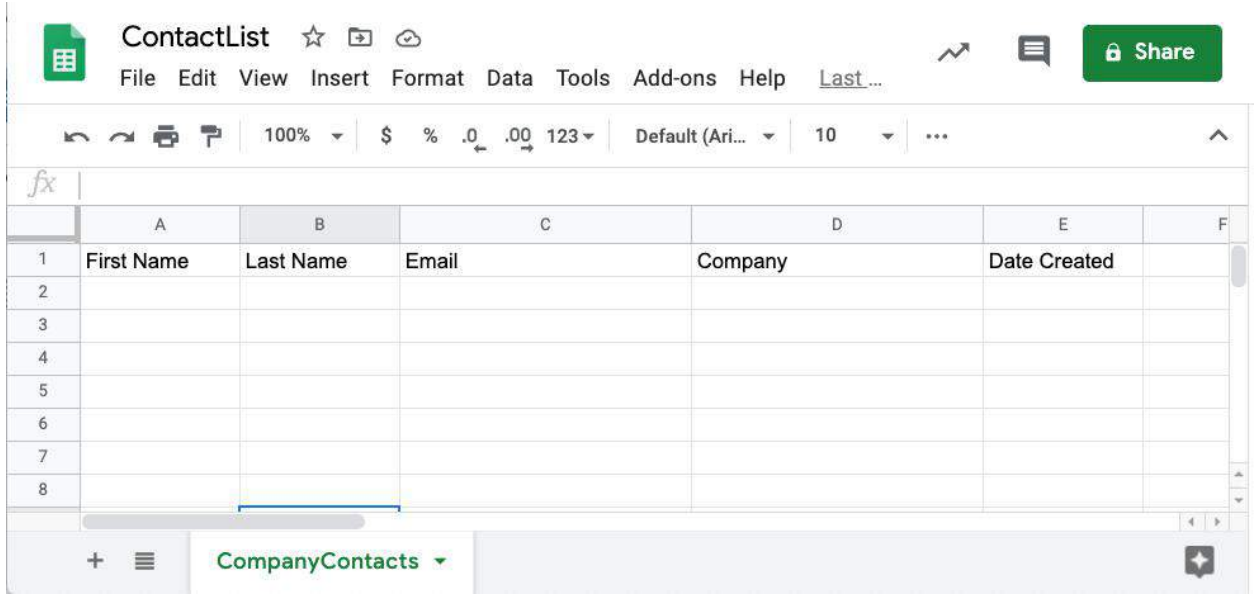
1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.



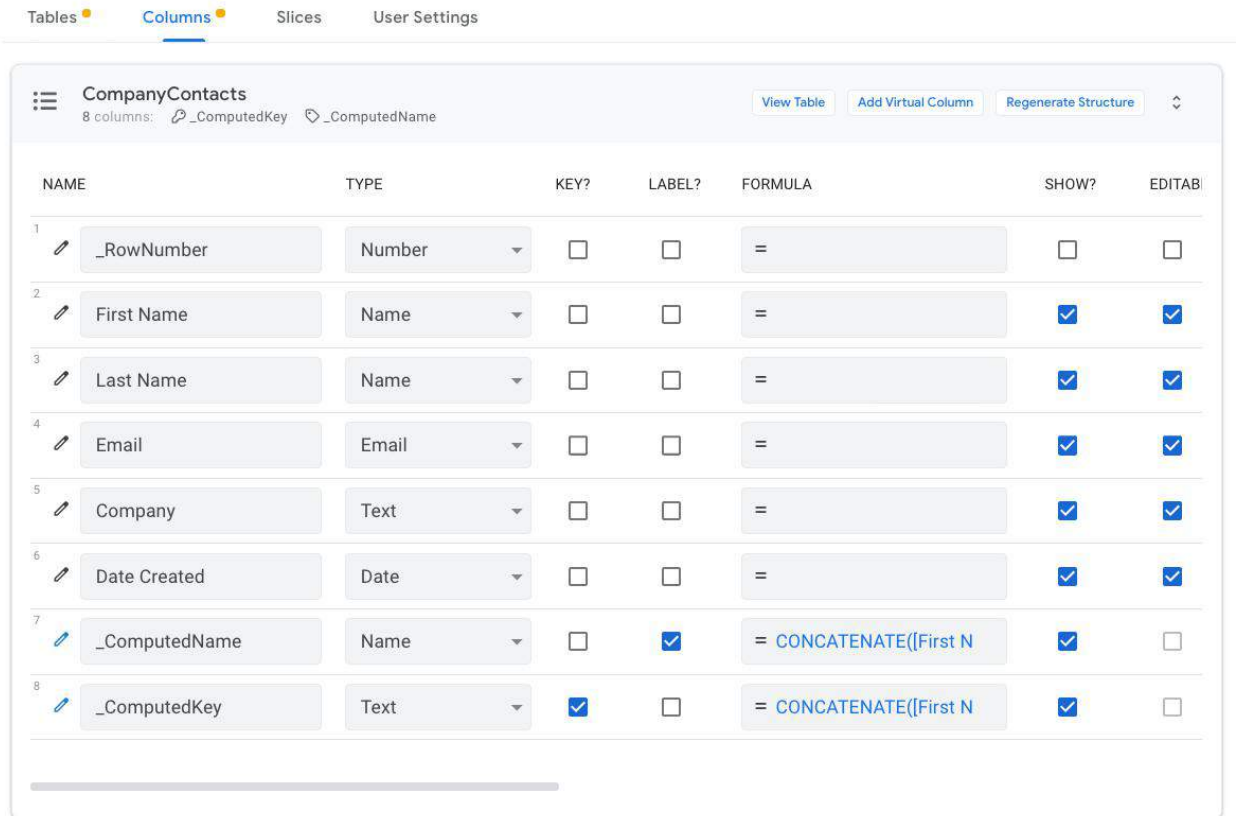
3. Select your configured Google Data Source.



4. Use the file browser to select the appropriate **ContactList** Sheet as the data source.



5. Select **Company Contacts** as the table for your entity.



## Configuring Contact entity from Salesforce datasource

Follow these steps to ensure that you have correctly configured a “Contact” entity in

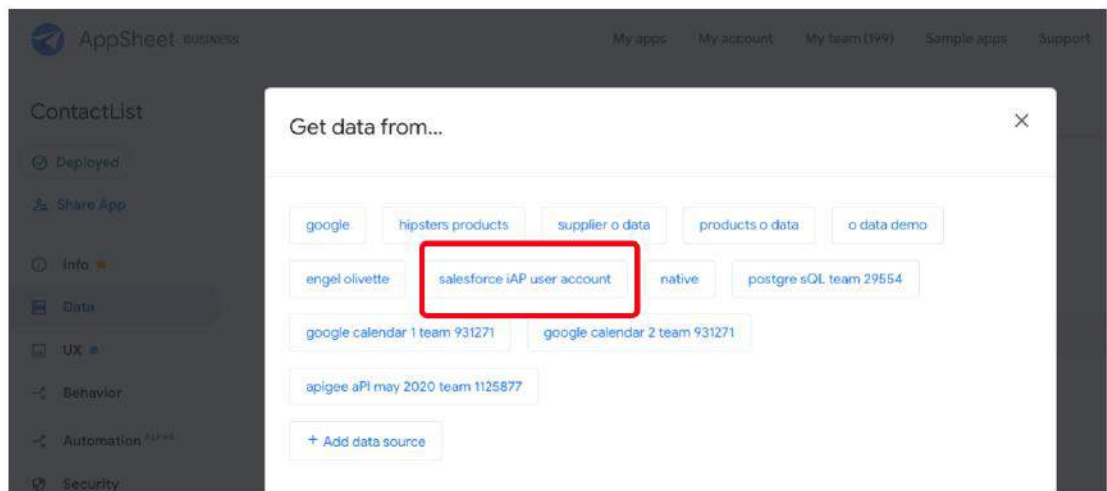
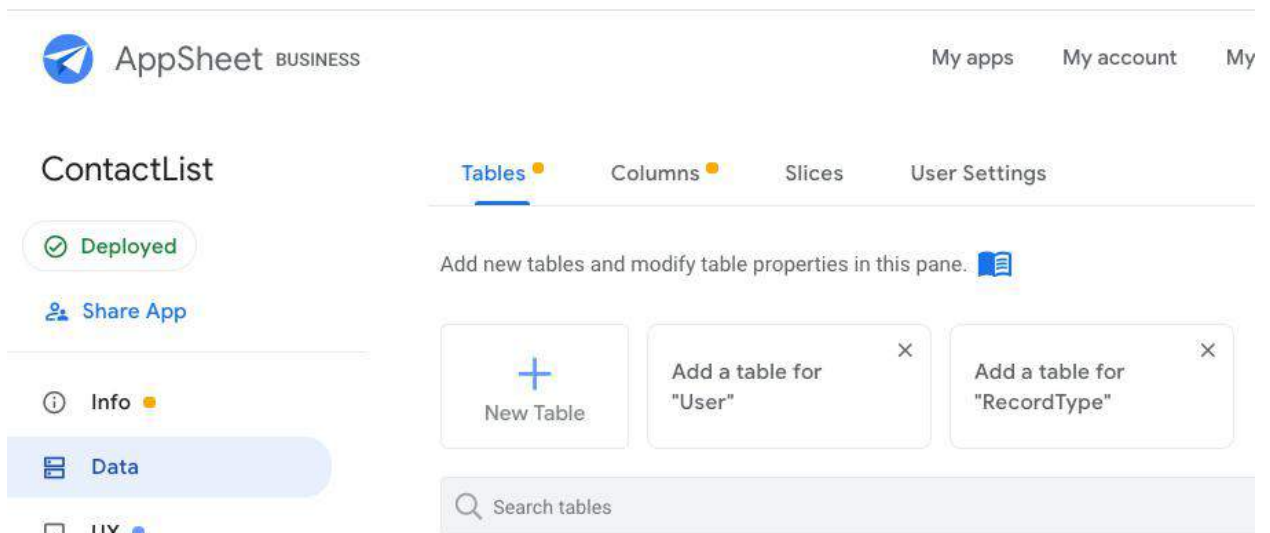
Salesforce.

Before you begin:

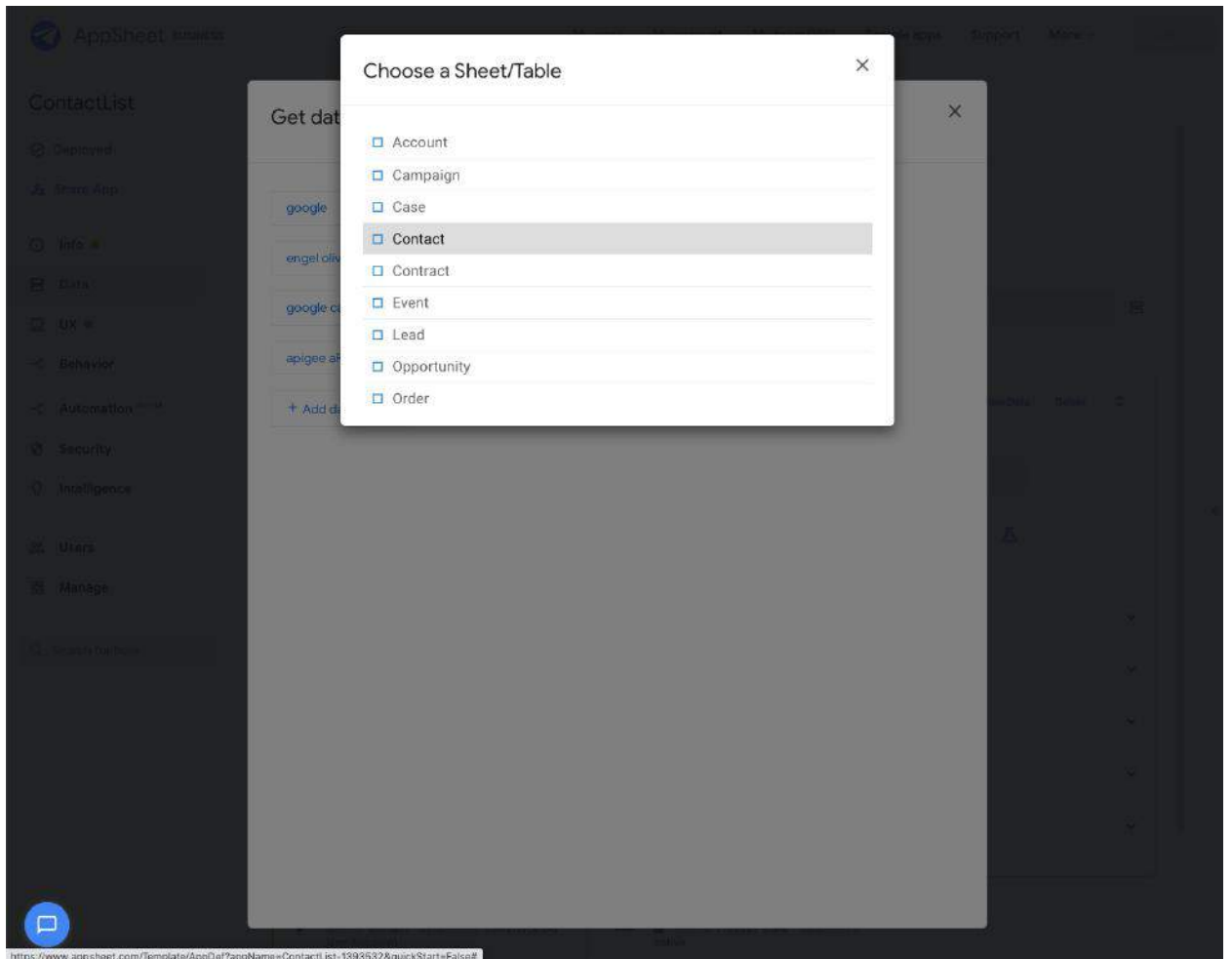
- Confirm that you have access to a Salesforce account.
- Confirm that the appropriate AppSheet package has been deployed in Salesforce. (Refer to this [help article](#) for further detail).

To configure a “Contact” entity in AppSheet using your Salesforce account data:

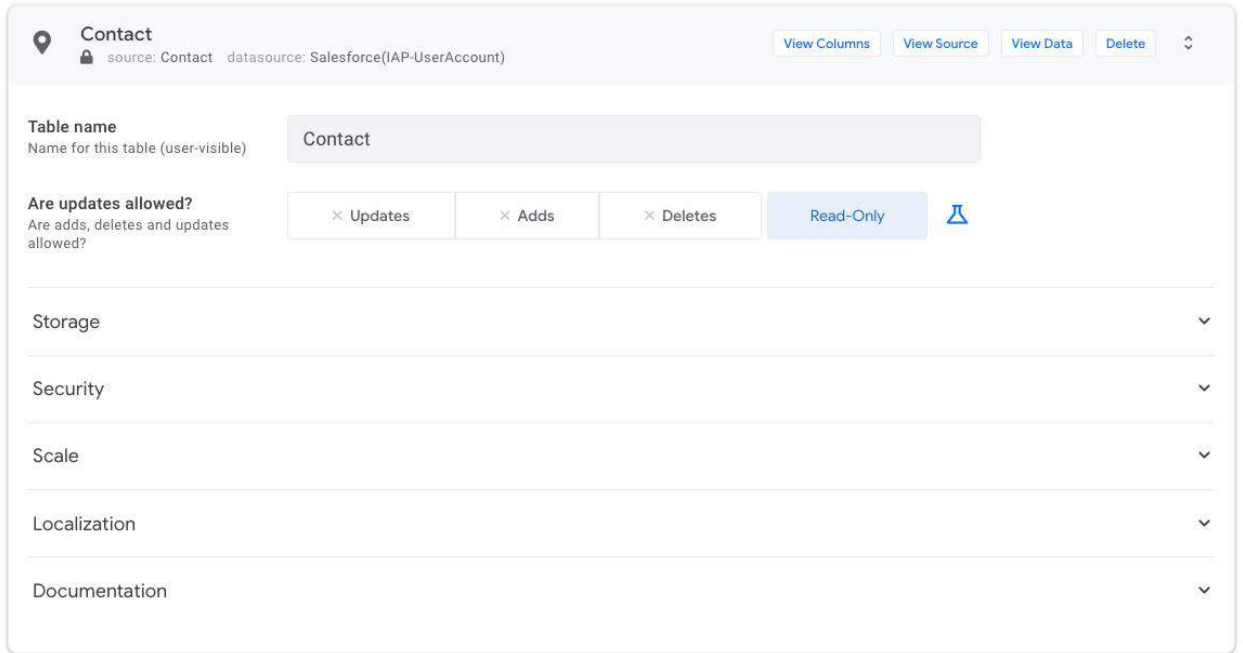
1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.
3. Select your configured Salesforce Data Source, as shown in the image below.



4. Select **Contact** as the table for your entity.



5. For **Are updates allowed?** change the selection to **Read-Only**.



6. Click **View Table**, as shown below:

**Contact** Warning  
 46 columns: [Row ID](#) [Name](#) [View Table](#) [Add Virtual Column](#) [Regenerate Structure](#)

⚠ Column "OwnerId" in Contact\_Schema has a reference to an unknown table or slice "User". [More info](#)

⚠ Column "CreatedById" in Contact\_Schema has a reference to an unknown table or slice "User". [More info](#)

⚠ Column "LastModifiedById" in Contact\_Schema has a reference to an unknown table or slice "User". [More info](#)

	NAME	TYPE	KEY?	LABEL?	FORMULA	SHOW?
1	<a href="#">_RowNumber</a>	Number	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/> <a href="#">⌵</a>
2	<a href="#">Row ID</a>	Text	<input checked="" type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/> <a href="#">⌵</a>
3	<a href="#">IsDeleted</a>	Yes/No	<input type="checkbox"/>	<input type="checkbox"/>	=	<input type="checkbox"/> <a href="#">⌵</a>
4	<a href="#">MasterRecordId</a>	Ref	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/> <a href="#">⌵</a>
5	<a href="#">AccountId</a>	Ref	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/> <a href="#">⌵</a>
6	<a href="#">LastName</a>	Text	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/> <a href="#">⌵</a>
7	<a href="#">FirstName</a>	Text	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/> <a href="#">⌵</a>
8	<a href="#">Salutation</a>	Enum	<input type="checkbox"/>	<input type="checkbox"/>	=	<input checked="" type="checkbox"/> <a href="#">⌵</a>
9						

7. Create a new Action (under **Behavior**) by configuring the fields as follows:

The screenshot displays the AppSheet interface for configuring an automation. The top navigation bar includes 'AppSheet BUSINESS', 'My apps', 'My account', 'My team (233)', 'Sample apps', 'Help', and 'More'. A 'Save' button is visible in the top right corner. The left sidebar shows navigation options: 'Not Deployed', 'Share App', 'Info', 'Data', 'UX', 'Behavior', 'Automation (PREVIEW)', 'Security', 'Intelligence', 'Users', 'Manage', and 'Learning Center'. The main content area is titled 'Update company contact' and shows the following configuration:

- Action name:** Update company contact
- For a record of this table:** Contact
- Do this:** Data: add a new row to another table using values from this row
- Table to add to:** CompanyContacts
- Set these columns:**

First Name	=	[FirstName]	⏏	🗑️
Last Name	=	[LastName]	⏏	🗑️
Email	=	[Email]	⏏	🗑️
Company	=	[AccountId]	⏏	🗑️
Date Created	=	[CreatedDate]	⏏	🗑️

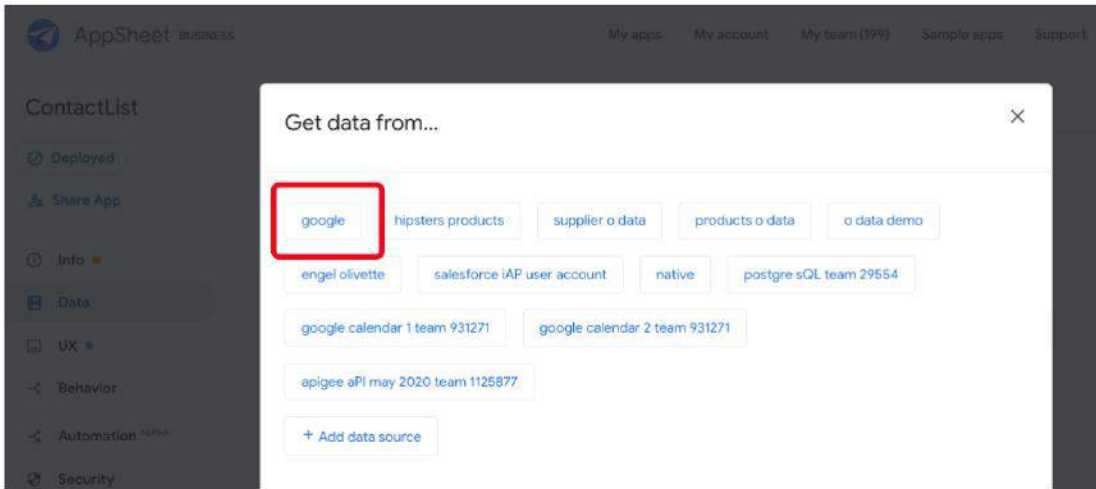
An 'Add' button is located at the bottom of the column configuration section.

## Configuring Leads entity from Google Sheets datasource

To configure your “Leads” entity from Google Sheets:

1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.
3. Select your configured Google Data Source.

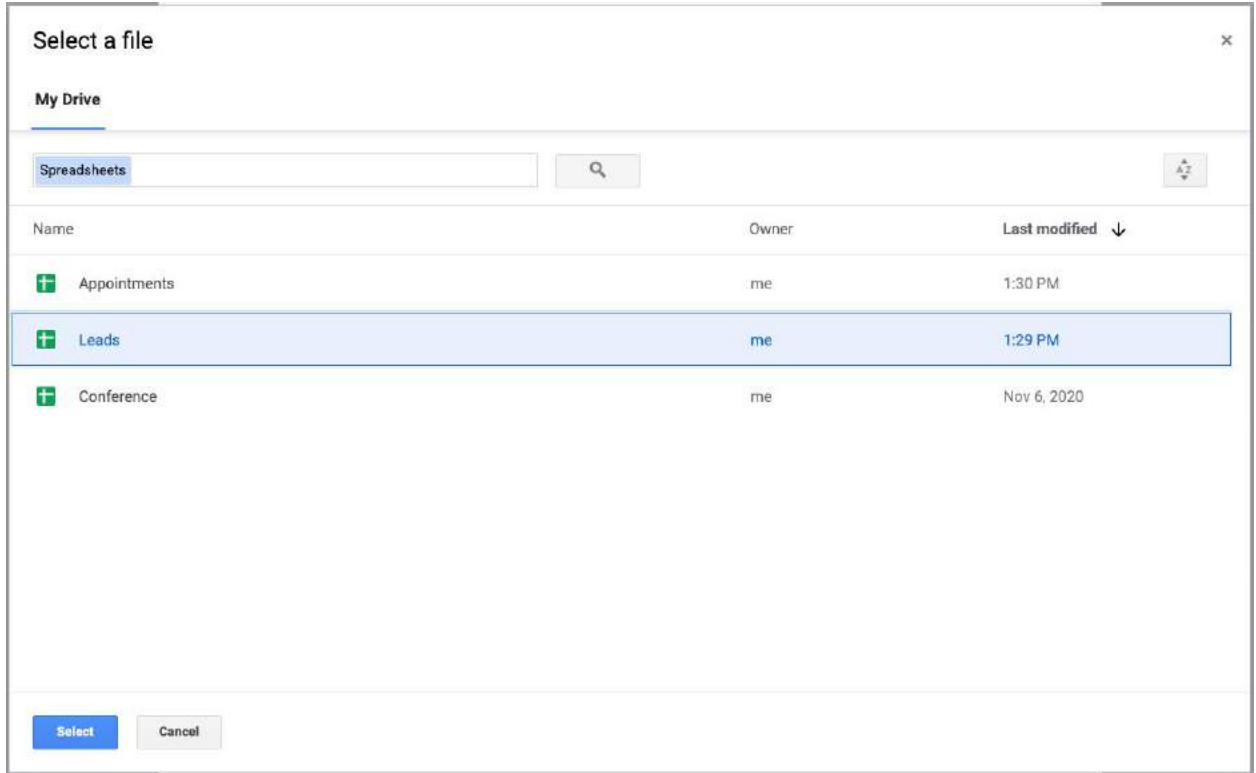




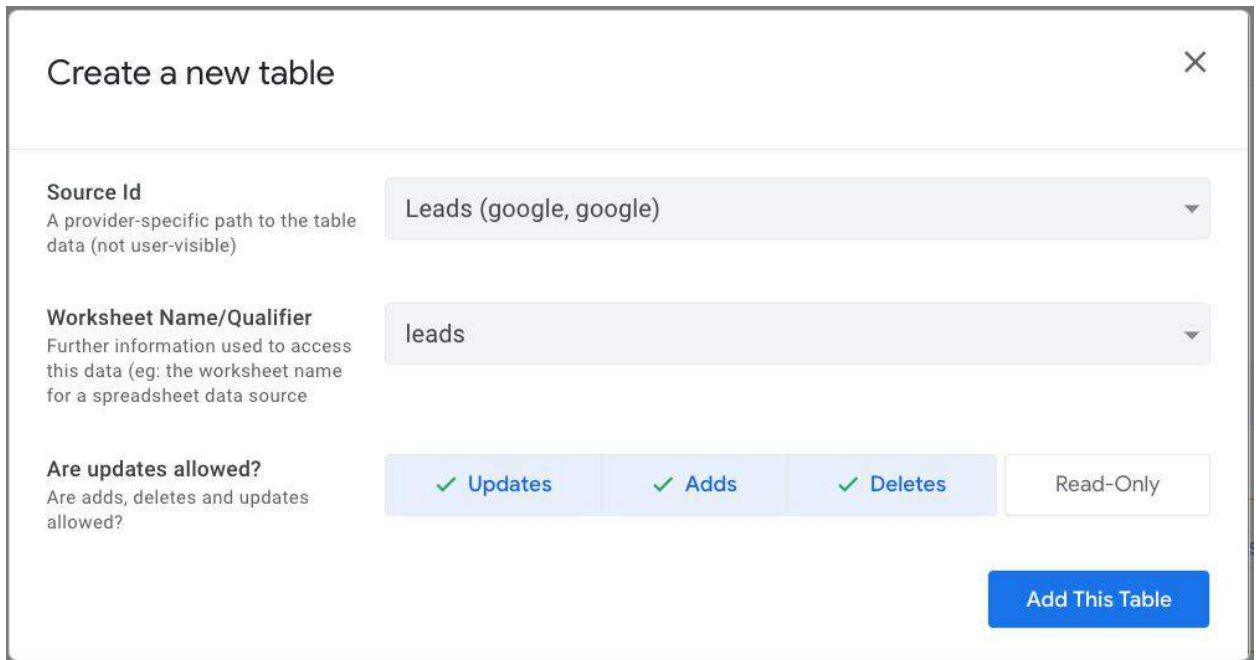
4. Create a new Google Sheet named “Leads.” Use the column names shown below and name the Sheet (tab) “leads”. The Sheet should be in the Google Drive accessible to your AppSheet account.

	A	B	C	D	E	F	G	H	I
1	First Name	Last Name	Email	Phone	Company	Date	FollowupRequested	FollowupAfterDays	Profile
2	Test	Lead1		m 3148889292	Acme Inc	11/1/2020	Y		5 High
3	Test	Lead2		m 4082224455	Acme Inc	11/1/2020	Y		6 Medium
4	Test	Lead3		m 9256667788	Acme Inc	11/10/2020	Y		7 Low
5	Test	Lead6		m 6691112222	Acme Inc	1/6/2021	N		5 Low
6									

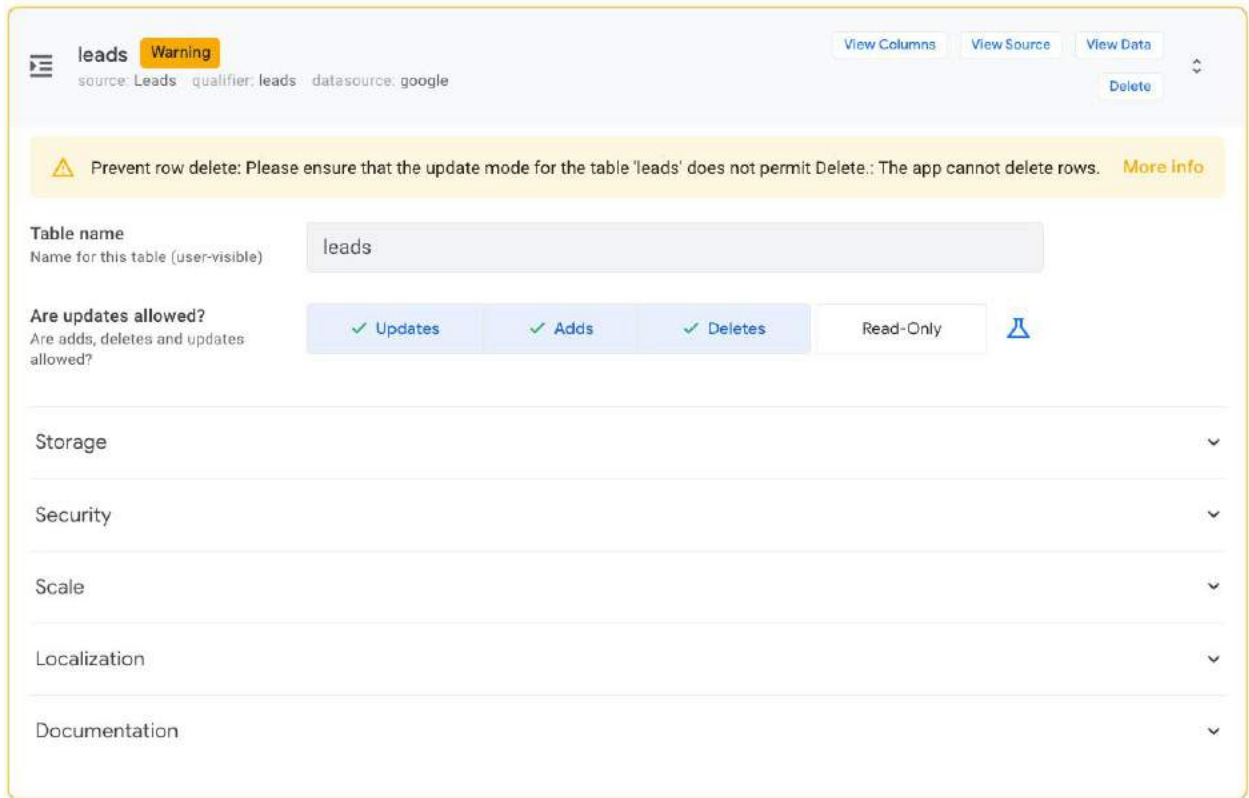
5. Use the file browser to select the appropriate **Leads** Sheet as the data source.



6. Select **leads** as the table for your entity.



7. Click **Add This Table**.

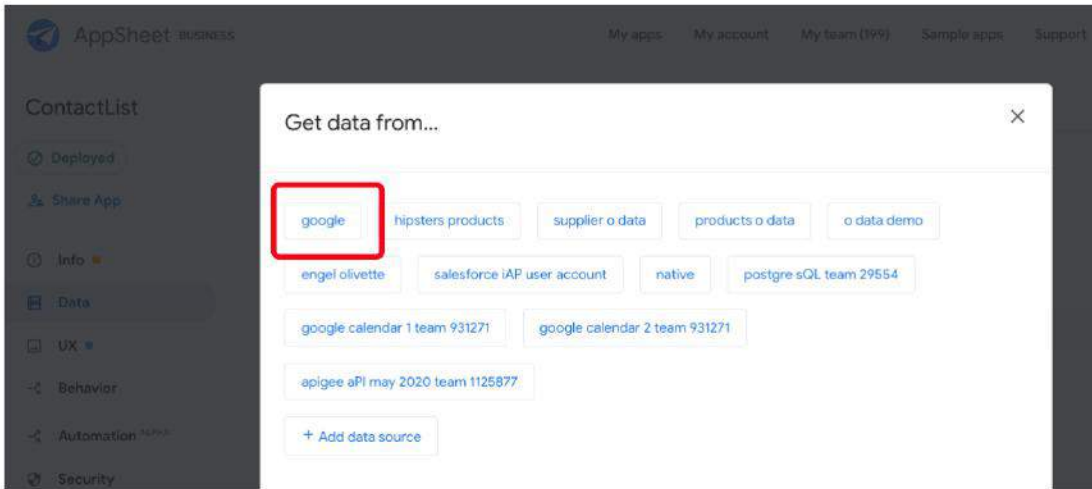


8. Click on “View Columns” and ensure that the data type of “FollowupAfterDays” column is “Number” and clear out initial entry of NOW() if it exists

## Configuring appointments entity from Google Sheets datasource

To configure your “appointments” entity from Google Sheets:

1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.
3. Select your configured Google Data Source.

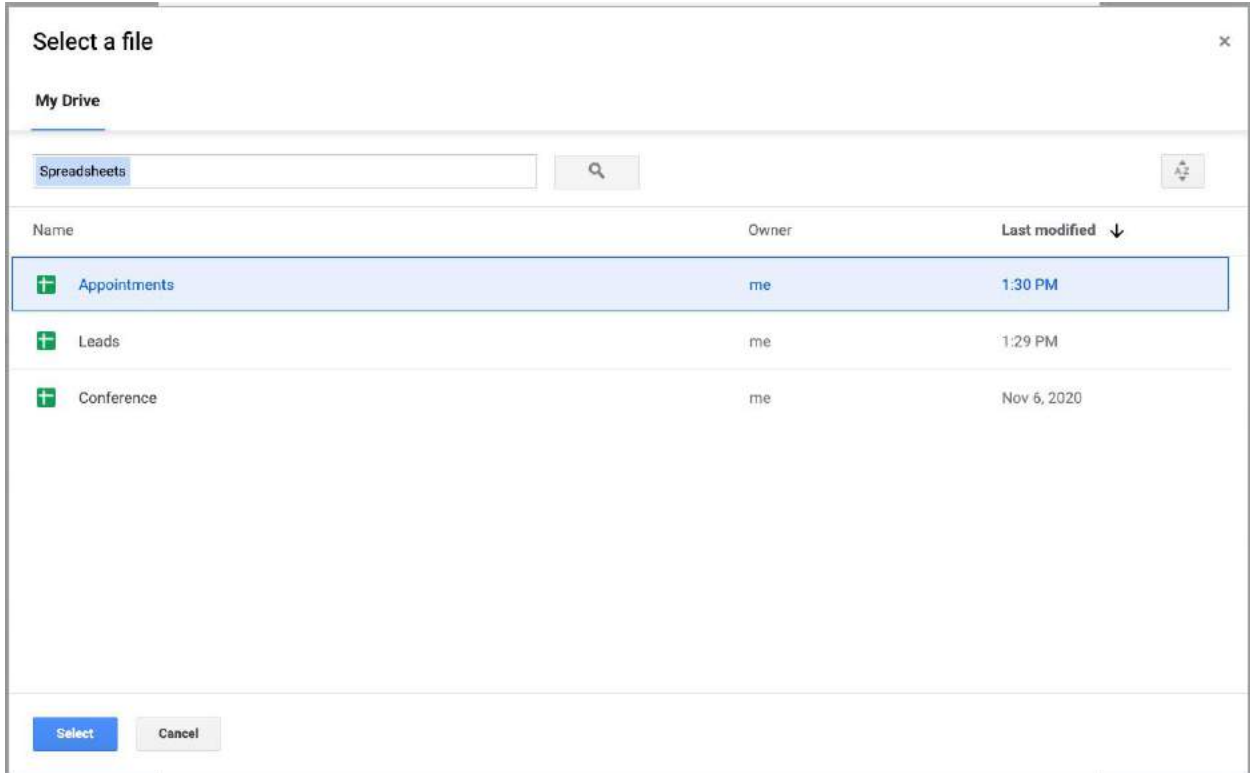


4. Ensure you create a new Google Sheet named “Appointments” (with the column names as below, the Sheet (tab) name should be “appointments”) in the Google Drive accessible to your AppSheet account

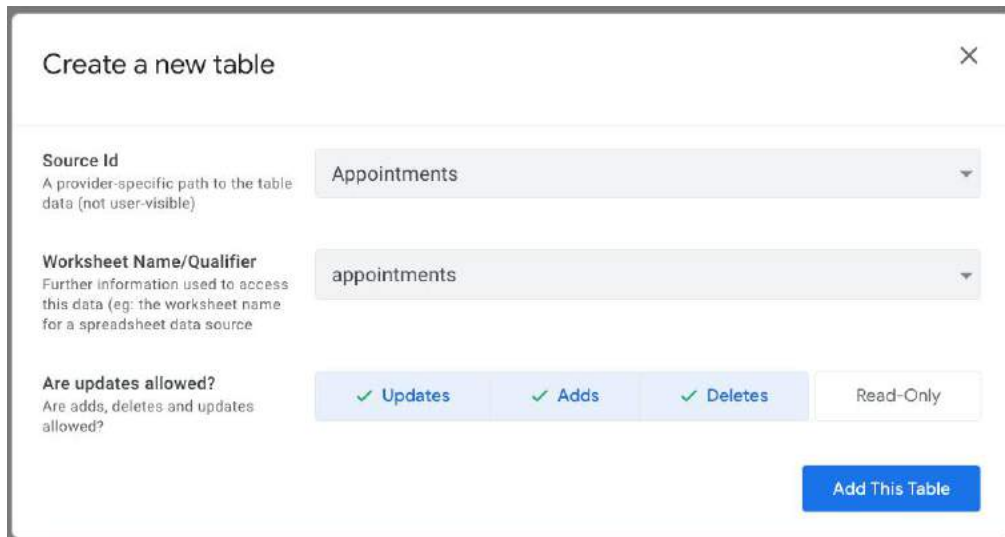
A screenshot of a Google Sheet titled 'Appointments'. The sheet has a menu bar (File, Edit, View, Insert, Format, Data, Tools, Add-ons, Help) and a toolbar with various editing tools. The data is organized in a table with the following columns: Full Name, Email, Appointment Date, Confirmation Required, Confirmation Status, and Confirmation Timestamp. The first row is the header, and the next three rows contain test data for 'Test Lead1', 'Test Lead2', and 'Test Lead3'.

	A	B	C	D	E	F
1	Full Name	Email	Appointment Date	Confirmation Required	Confirmation Status	Confirmation Timestamp
2	Test Lead1		11/6/2020	Y	confirmed	1/20/2021 12:34:56
3	Test Lead2		11/7/2020	N	not-required	1/19/2021 14:28:43
4	Test Lead3		11/17/2020	N	not-required	1/20/2021 18:35:27
5						

5. Use the file browser to select the appropriate **Appointments** Sheet as the data source.



6. Select **appointments** as the table for your entity.



7. Make sure the **Confirmation Required** column type is **Text**.

**appointments** View Table Add Virtual Column Regenerate Structure

7 columns: Full Name Full Name

NAME	TYPE	KEY?	LABEL?	FORMULA
_RowNumber	Number	<input type="checkbox"/>	<input type="checkbox"/>	=
Full Name	Name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	=
Email	Email	<input type="checkbox"/>	<input type="checkbox"/>	=
Appointment Date	Date	<input type="checkbox"/>	<input type="checkbox"/>	=
Confirmation Required	Text	<input type="checkbox"/>	<input type="checkbox"/>	=
Confirmation Status	Text	<input type="checkbox"/>	<input type="checkbox"/>	=
Confirmation Timestamp	DateTime	<input type="checkbox"/>	<input type="checkbox"/>	=

8. Create a new Action (under **Behavior**) as follows:

The screenshot displays the AppSheet interface for configuring an automation action. The top navigation bar includes the AppSheet logo, user information (My apps, My account, My team (233)), and utility links (Sample apps, Help, More, SAVE). The left sidebar shows the 'Sample Automation App' with a 'Deployed' status and various management options like Info, Data, UX, Behavior, Automation, Security, Intelligence, Users, Manage, and Learning Center. The main content area is titled 'appointments' and shows a configuration window for the action 'Update confirmation status to "not-required"'. The configuration includes:

- Action name:** Update confirmation status to "not-required"
- For a record of this table:** appointments
- Do this:** Data: set the values of some columns in this row
- Set these columns:** Confirmation Status = "not-required"

At the bottom of the configuration window, there are expandable sections for Appearance, Behavior, and Documentation. Below the window, there are links for 'Rearrange actions' and 'Show system actions'.

## Configuring an email template for ops notification

To configure an email template, create a document in your Google drive. Follow the steps below to configure your email template:

1. Navigate to your Drive and create a new document named **Ops Notification Template**.

The screenshot shows the Google Drive interface. On the left is a navigation sidebar with options: New, Priority, My Drive, Shared drives, Shared with me, Recent, Starred, Trash, and Storage (24.2 MB used). The main area shows the path 'My Drive > Leads'. A table lists the contents of the 'Leads' folder:

Name	Owner
Appointments	me
Conference	me
Leads	me
<b>Ops Notification Template</b>	me

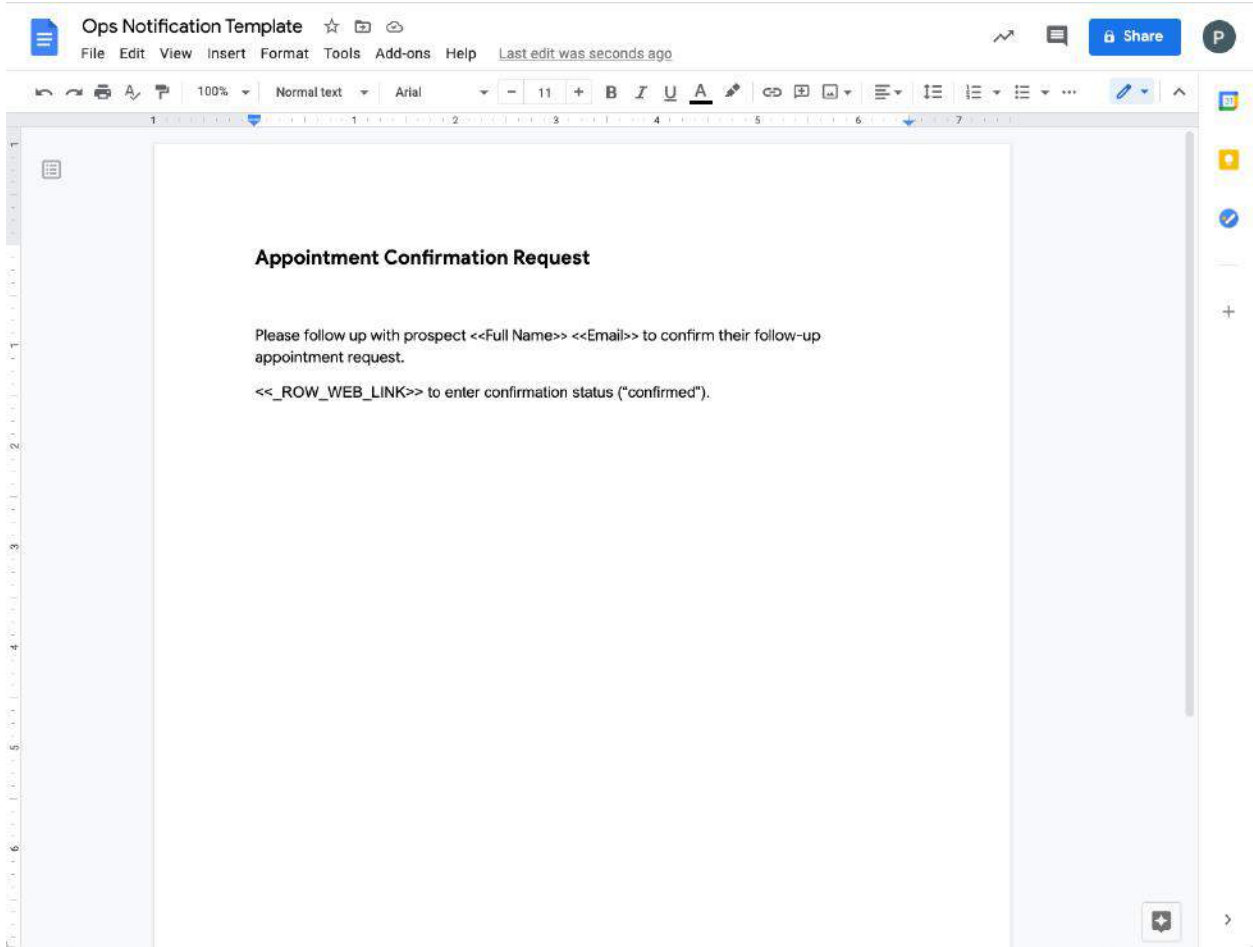
2. Enter the following text in the document, as shown in the images below:

## Appointment Confirmation Request

Please follow up with prospect <<Full Name>> <<Email>> to confirm their follow-up appointment request.

<<\_ROW\_WEB\_LINK>> to enter confirmation status (“confirmed”).





## Document processing states and status codes

Document processing in AppSheet relies on AI to identify and extract content from documents. Under some circumstances, the AI may fail to extract high quality structured data from a document. To help understand the status of the content extraction, a `StatusCode` column is included in the document data. In addition to the status code, a textual description of the potential issue is included in a column named `AttentionDetails`. This is intended to help you understand what may be the issue so you can correct it.

### Potential status codes and values

As noted earlier, document extraction may result in the following status codes:

StatusCode	AttentionDetails
EXTRACTION_ERROR	The document is either of an invalid type or did not appear to have the correct content.
POOR_QUALITY	Some of the values extracted have low confidence scores in the AI model. This implies that there is uncertainty in the accuracy of the data.
INCONSISTENT_CURRENCY	Extracted data from the document appears to reference multiple currencies.
EMPTY_CURRENCY	The document does not specify a currency, though one is expected.

The intended flow of document extraction is to allow you to automatically utilize the extracted data in downstream processes and actions. In order to support this, two additional indicator columns are included in the document data:

- `IsEdited`
- `NeedsAttention`

These columns can be used to understand if the document data is ready for downstream use.

When the value in the `NeedsAttention` column is `TRUE` (Yes), this indicates that a `StatusCode` is present requiring manual review. For example, if the content of a field may have low quality, or the currency code present in a field is unknown, the `NeedsAttention` flag indicates that the data should be reviewed manually, corrected, and results approved for downstream uses. When an edit is made to the data after processing, the `NeedsAttention` column is set to `FALSE` and the `IsEdited` column is set to `TRUE`. This indicates that the data is now ready for downstream use.

Below is a brief description of several common `StatusCodes` returned on data slices, and the implications for the resulting document data.

- `AND([NeedsAttention], [StatusCode] = "EXTRACTION_ERROR")`: This slice of data represents the files that could not be extracted. This usually indicates files of

the wrong file format or files with content that is not aligned with the intended use.

- `OR(NOT([NeedsAttention], [StatusCode] != "EXTRACTION_ERROR")`: This slice of data represents all of the files that had content extracted. This slice may contain potentially incomplete or invalid data. This slice is useful for viewing all the updated invoice data in one place, for example, as a view in an app.
- `AND([NeedsAttention], [StatusCode] != "EXTRACTION_ERROR"`: This slice of data represents all of the files that have been flagged as needing manual review. This slice is helpful to build a human into the process to verify and correct any potential data issues.
- `NOT[NeedsAttention]`: This slice represents all of the document data that is considered high quality and/or human verified. This slice is appropriate for integration into downstream use cases for automation or other purposes.

## Why is there a separate currency code column?

For documents that have some type of monetary column, the value of the monetary columns and the currency code are separated out. This provides more flexibility in using the extracted data, such as supporting a centralized repository of documents spanning multiple currencies to allow customization of downstream processes. However, there is only ever considered a single currency for a single document; inconsistencies in currency type found in the extraction are flagged.

## Where can I store my files for Document processing?

At this time, AppSheet supports processing documents found on Google Drive. Support for additional file storage providers for document processing is underway.

## What does 'collection of files' mean?

In addition to document content extraction, AppSheet now supports exposing your folder contents as a table in your application itself. This feature allows app creators to expose files directly in their app, and use file metadata in their application logic (e.g. filtering by names, dates modified, etc).