AppSheet Automation

Preview Release – Primer

Version 1.2

Updated: Feb 3, 2021

Disclaimer

This primer describes the basic capabilities of the AppSheet Automation Preview release. The Preview features described are not production ready and are intended for use in test environments only. **We do not recommend using these Preview features in production or with production data sources.** The Preview release is subject to change at any time and is provided to you for evaluation purposes only, without any SLAs. For more information, see the product launch stages.

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.

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 preview 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 preview release of AppSheet Automation.

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**^{PREVIEW} 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.
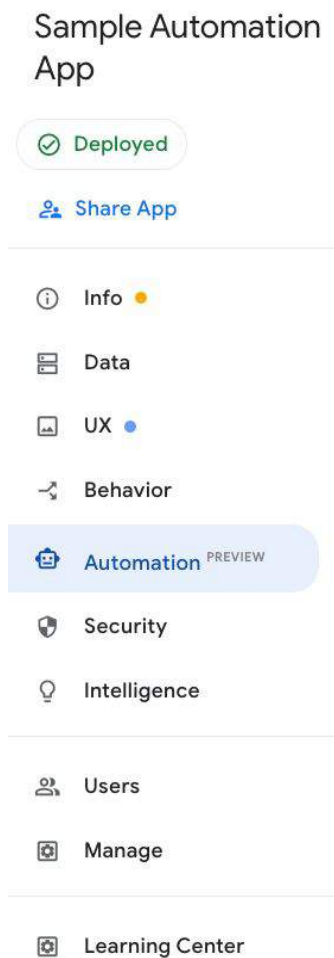
Bot

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 "Sync Contacts" process anytime a new "Contact" record is created in Salesforce. This bot continues to listen for the order creation event and fire the associated process when it event occurs until it is disabled.

Process

A *process* represents a typical business process. For example, an "order approval" is a business process. A process contains the following elements:

- A sequence of steps
- An input
- An output (this is optional)
- An event that triggers the process

For example, in the screenshot below, you can see a sample "Sync Contacts" process. In this example, the *event* that triggers the process is the addition of a new contact record.
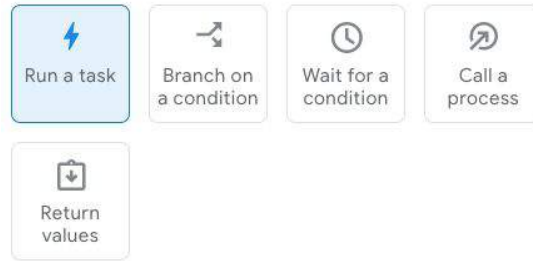
Step

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
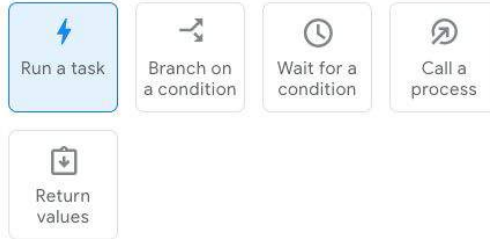- Call a process
- Return values

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.
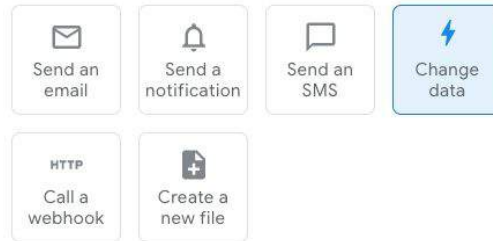
Step name

Update contact list

Do this

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

Return values

Task
What task do you want to run?

Update contact

Task category
Category of task

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.

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.

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

Update contact list

Do this

⚡ Run a task	↗ Branch on a condition	🕐 Wait for a condition	↗ Call a process
⬇ Return values			

Process name

Sync contact list ▾

Process inputs

First Name

= [FirstName]   ⚗

Last Name

= [LastName]   ⚗

Email

= [Email]   ⚗

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.

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 "Change data" task type along with the associated data action.

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 "Contact" data schema.

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.

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.

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`). An   *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 Preview 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 Preview Release: `.tiff`, `.gif`, and `.pdf`. Folders containing mixed document types can be used for document viewing and upload inside applications, but can not be automatically extracted.*

For the Preview 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 Preview Release, the confidence threshold cannot be adjusted by application editors.

Before you begin

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

- Created an account on appsheet.com.
- Ensure that you have access to the Automation preview release (Be able to see Automation PREVIEW logo in the left nav bar).
- Recommend that you create a new Sample Automation App that you specifically use for creating and testing your new automations.

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:
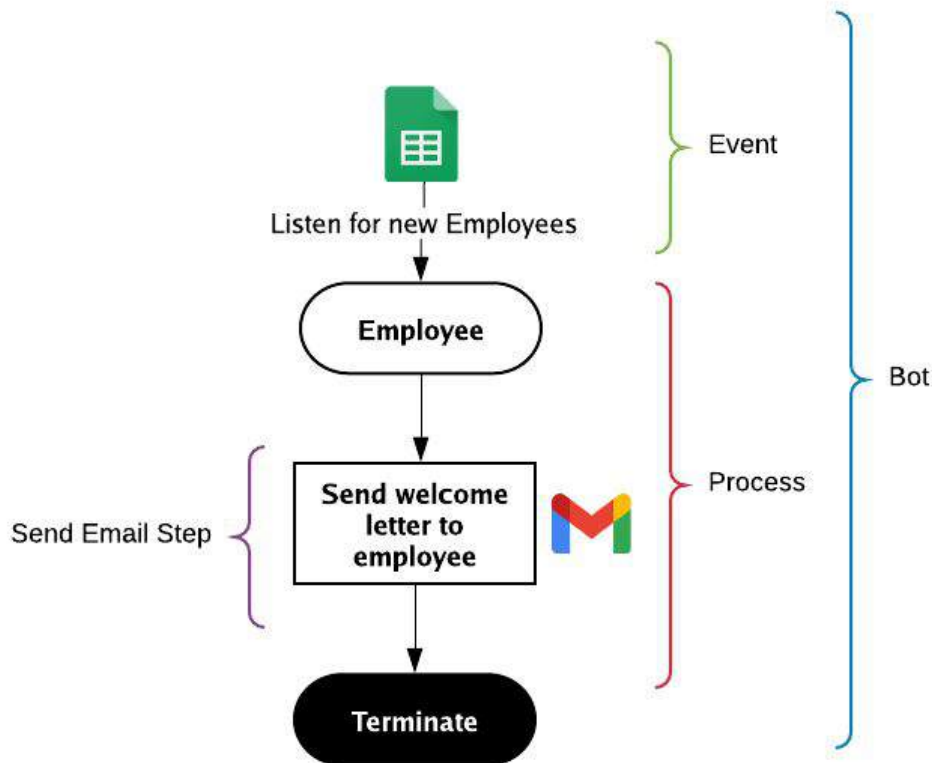
Before you begin this tutorial:

- 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.
3. Select **Create empty bot named 'Employee Onboarding'** to create the bot.
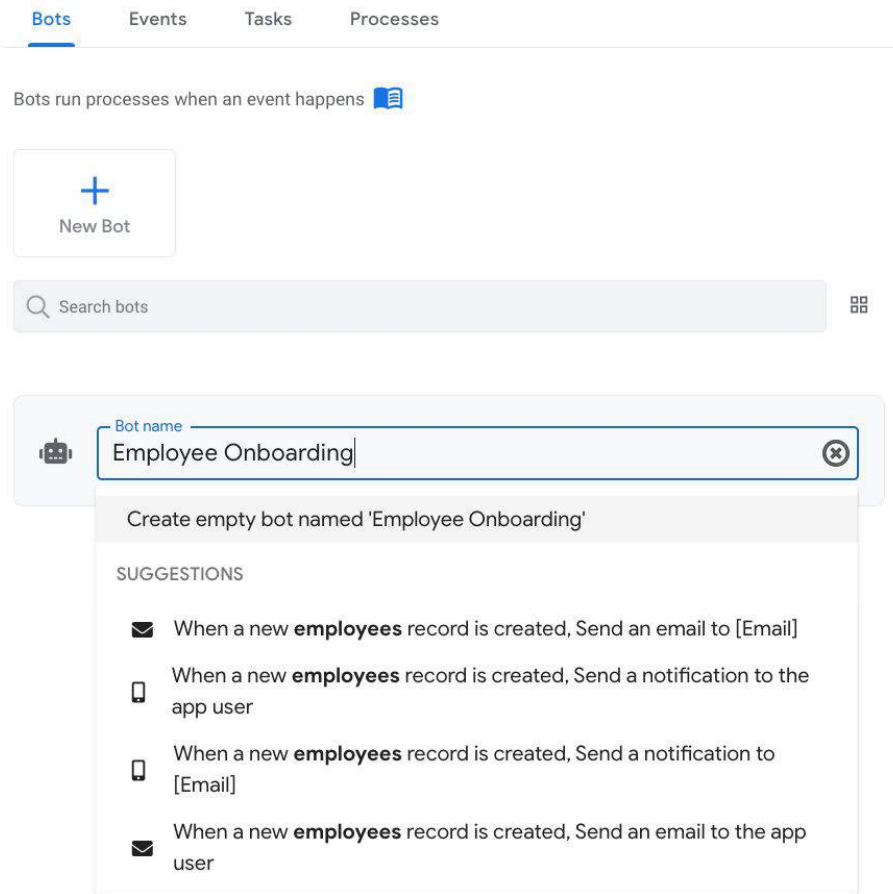
Configuring the event

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 may 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.

24

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.*

5. Enter **A new Employee record is created** as the event name and configure the event as follows:

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

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 `Send welcome letter to employee` into the text box and hit **enter** to automatically create an empty step. This implicitly creates a process.
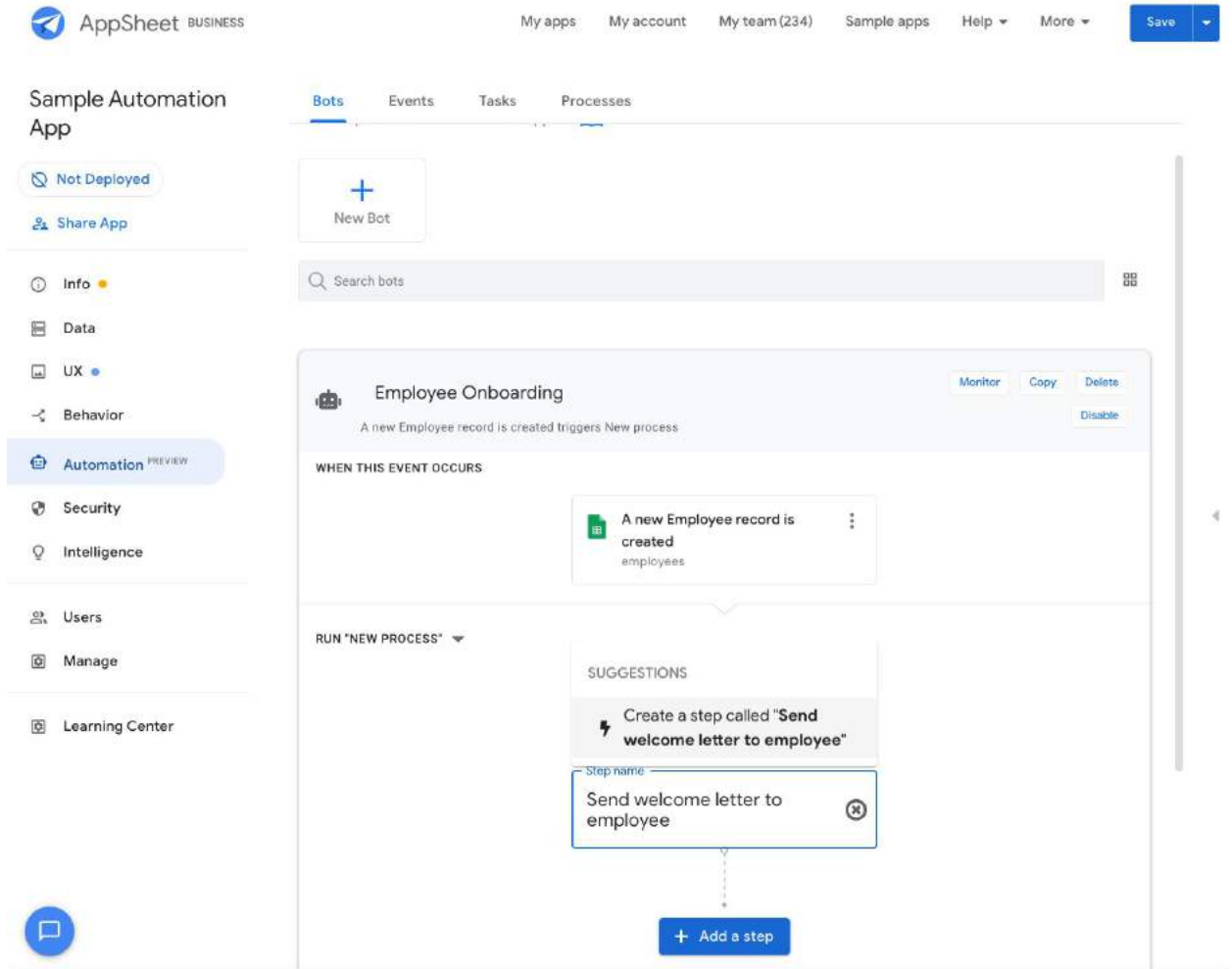
2. Click on the **Send welcome letter to employee** step to expand its definition, as shown below:
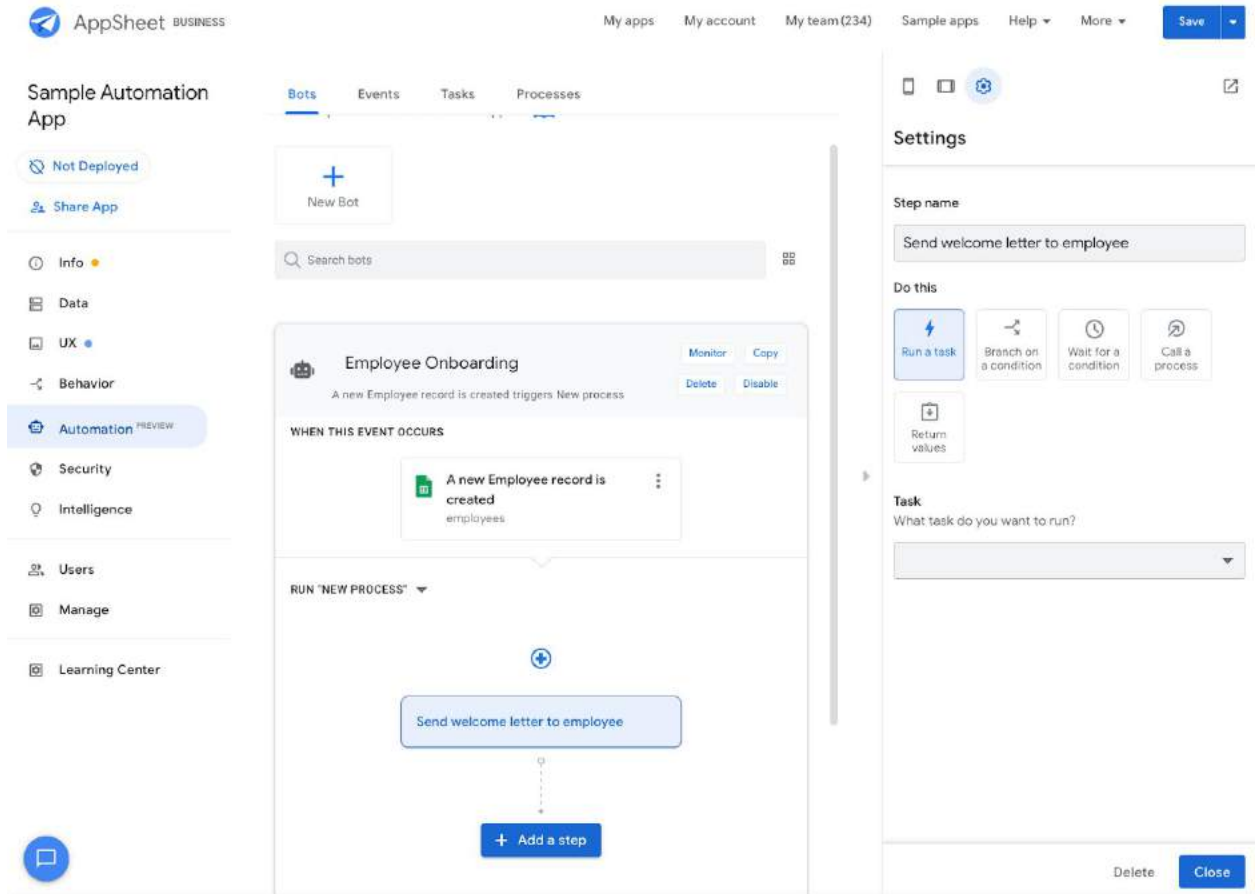
3. In the **Settings** pane, configure the step as follows:
    a. In the **Do this** section, select **Run a task**.
    b. In the **Task** section, click **Create new task**.
    c. Refer to the table below for the rest of the step configuration:

Field Name	Value
Do this	Run a task
Task category	Send an email
Task name	Email welcome letter
Table Name	employees
To	[Email]

Use default content?	Disabled
Email Subject	Welcome onboard
Email body	<<TODAY()>>\n\nDear << CONCATENATE ([First Name], " ", [Last Name])>>,\n\nWe are excited to have you on board. We are delighted to have you join us as a <<[Job Title]>> in our team. Your role is critical in fulfilling our mission.\n\nThe Human Resources team is here to support you throughout your career.\n\nSincerely,\nHR Manager

4. Click **Save** to save all your changes, click **Close** to hide the right pane.

Your bot should now look like this:

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.
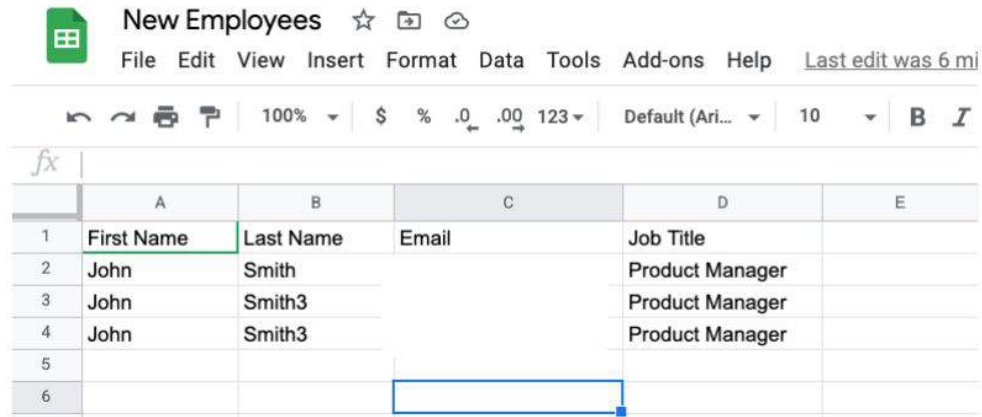
Now you are ready to test your bot.

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

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:
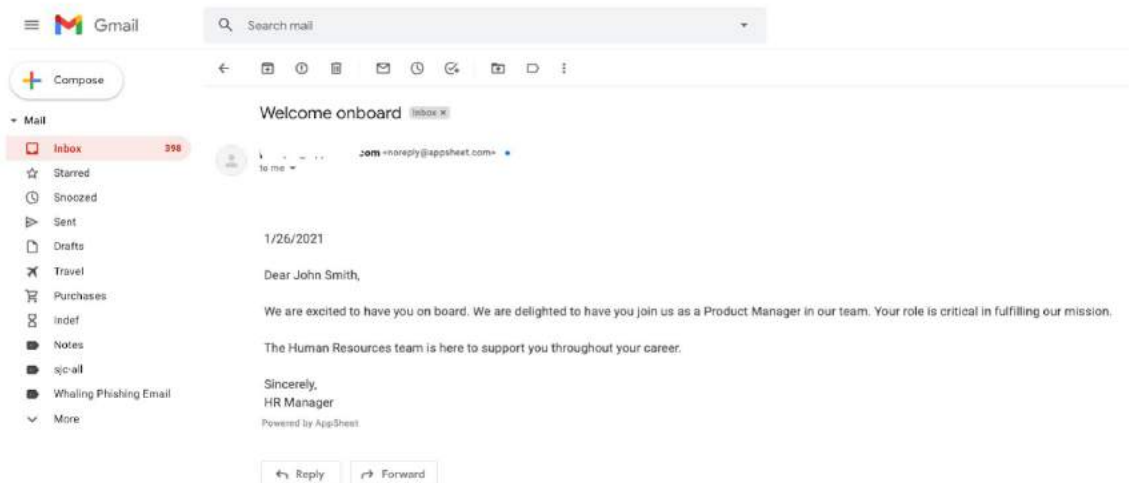

Automation Monitoring

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) Automation Metrics View (High level metrics for bots/events/processes) and 2) Automation Diagnostics View (Bots/Events/Processes 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.
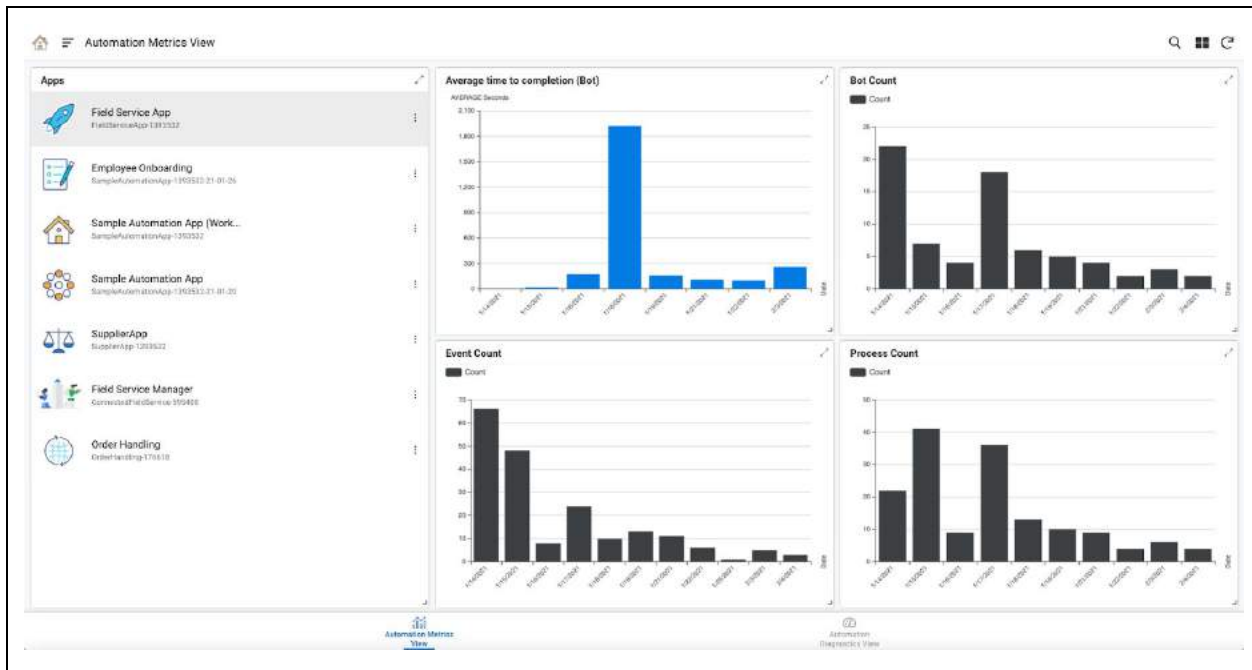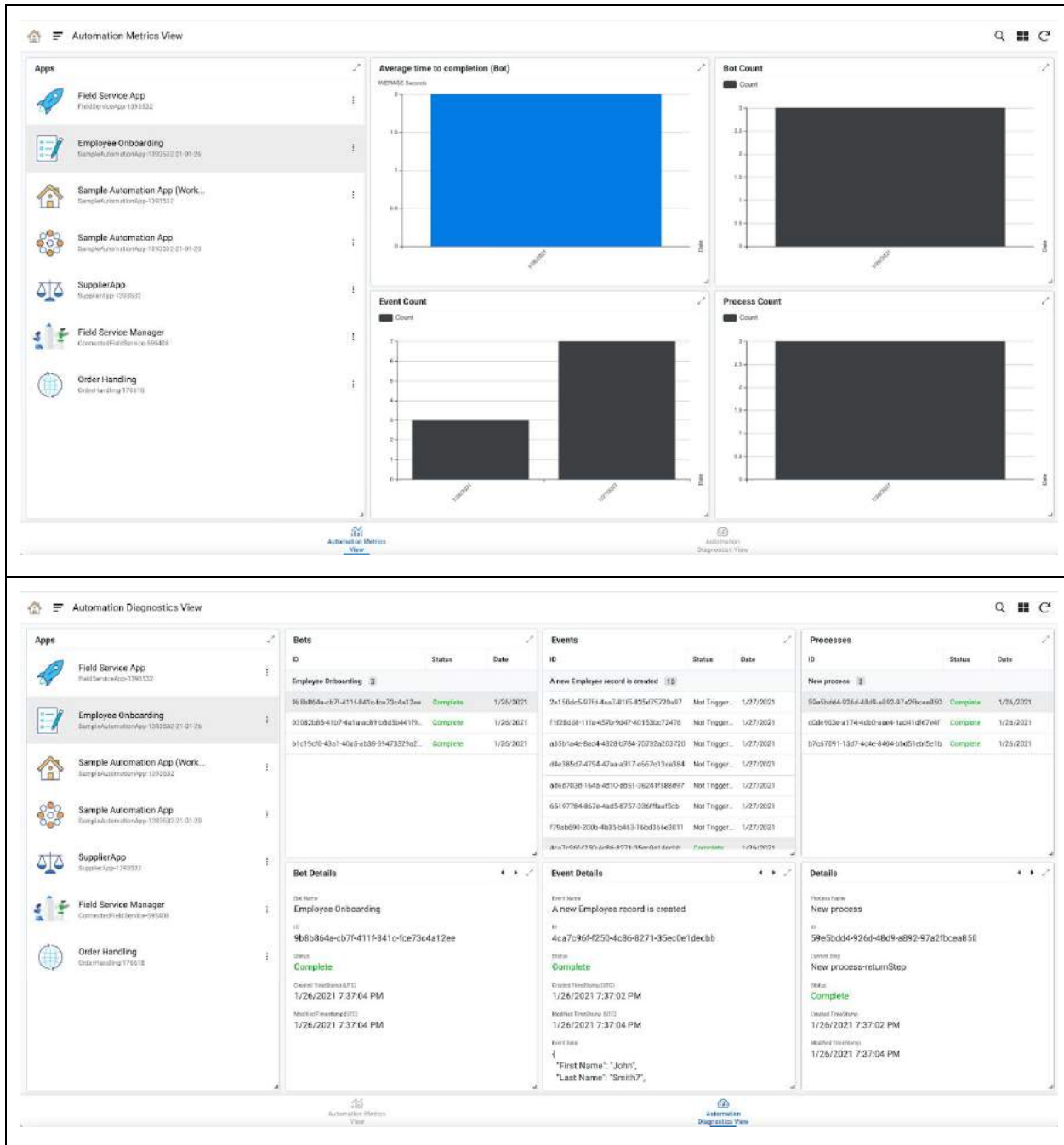
Contact Sync

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:
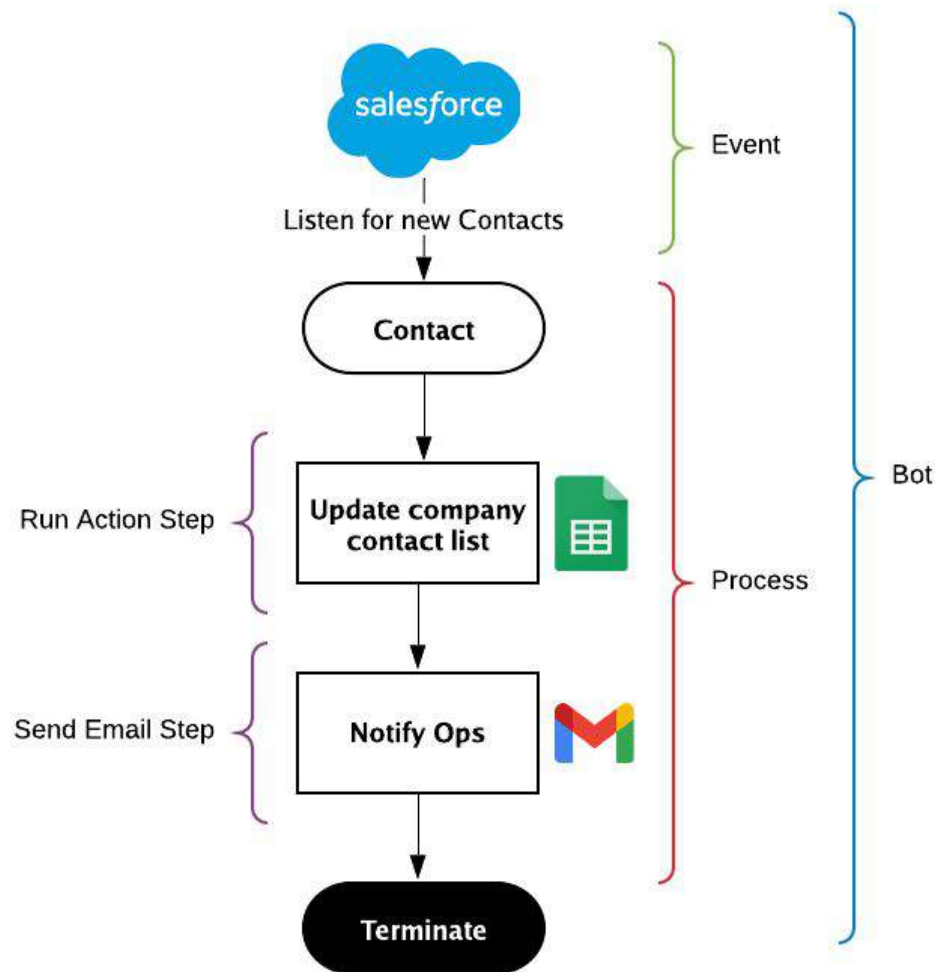
Before you begin this tutorial:

- Make sure you 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 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 empty bot named 'Contacts Sync'** to create the bot.
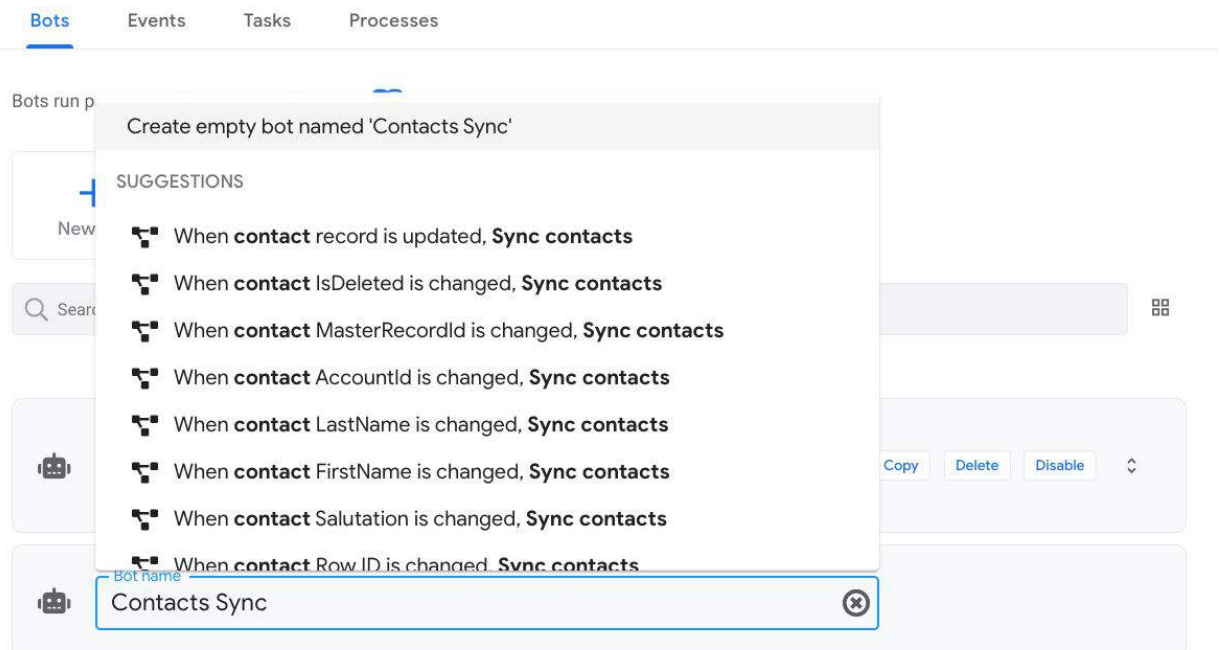
Google Confidential - Internal Use Only (Do not print, distribute or share without explicit permission)   40

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:

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 hit **enter** to automatically create an empty step. This implicitly creates a process.
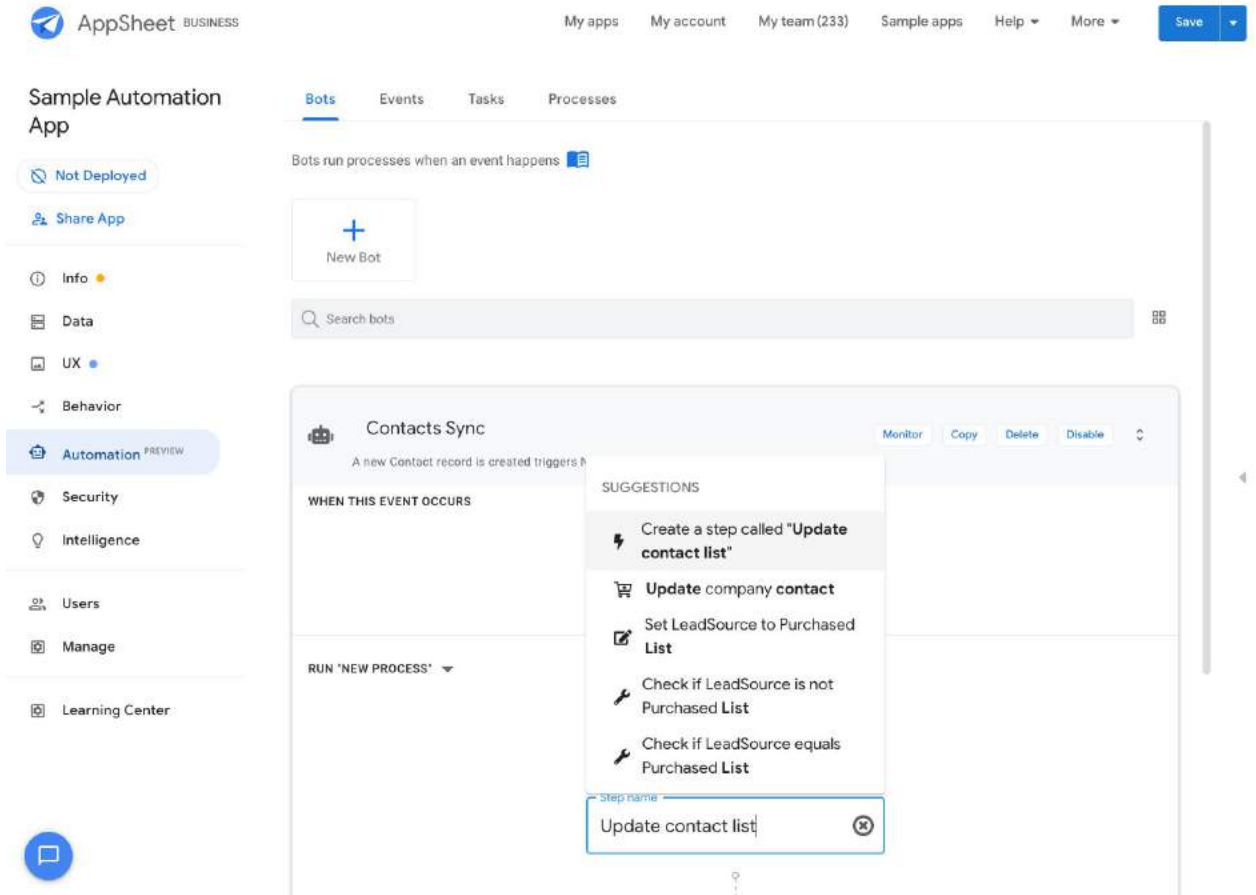
2. Click on the **Update contact list** step to expand its definition, as shown below:
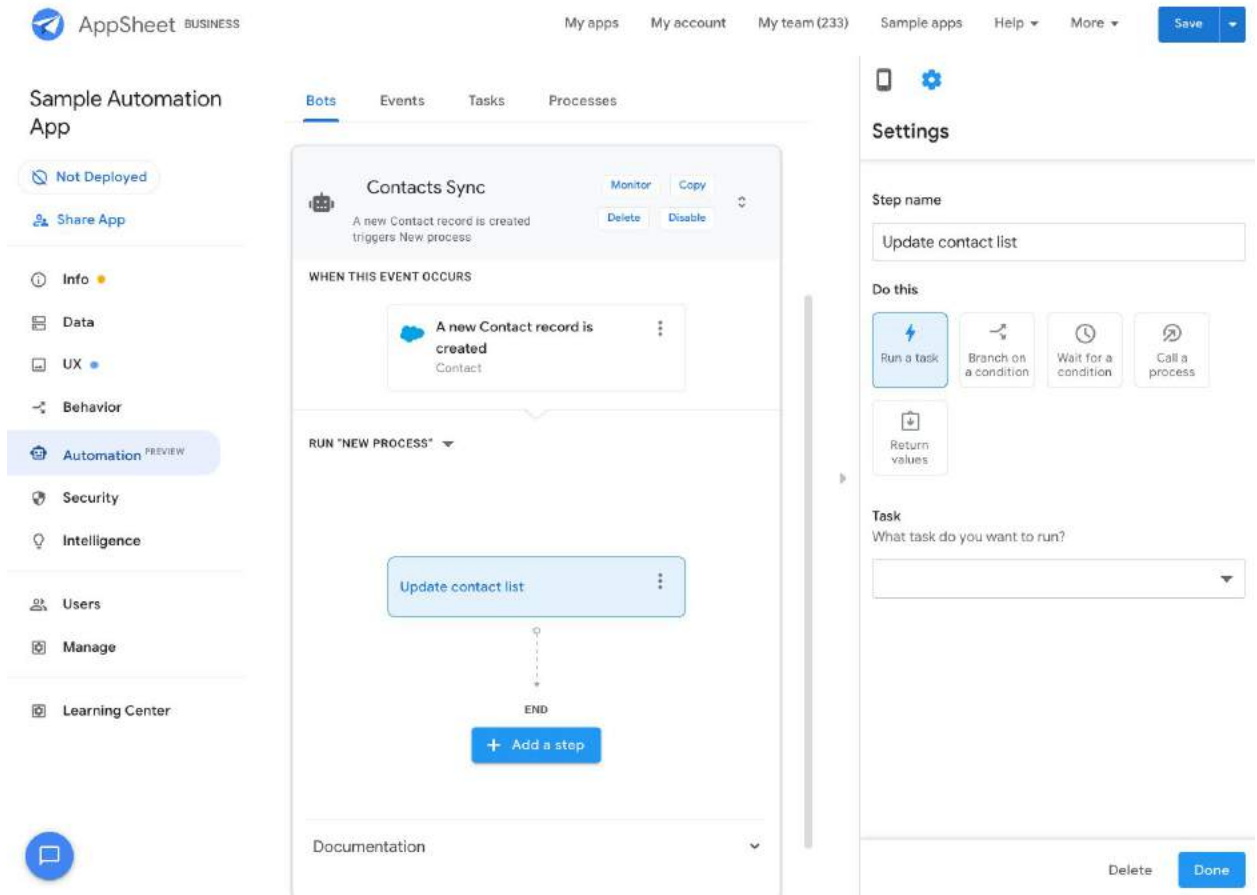
3. In the **Settings** pane, configure the step as follows:
   a. In the **Do this** section, select **Run a task**.
   b. In the **Task** section, click **Create new task**.
   c. And under **Task Category**, select **Change data**.
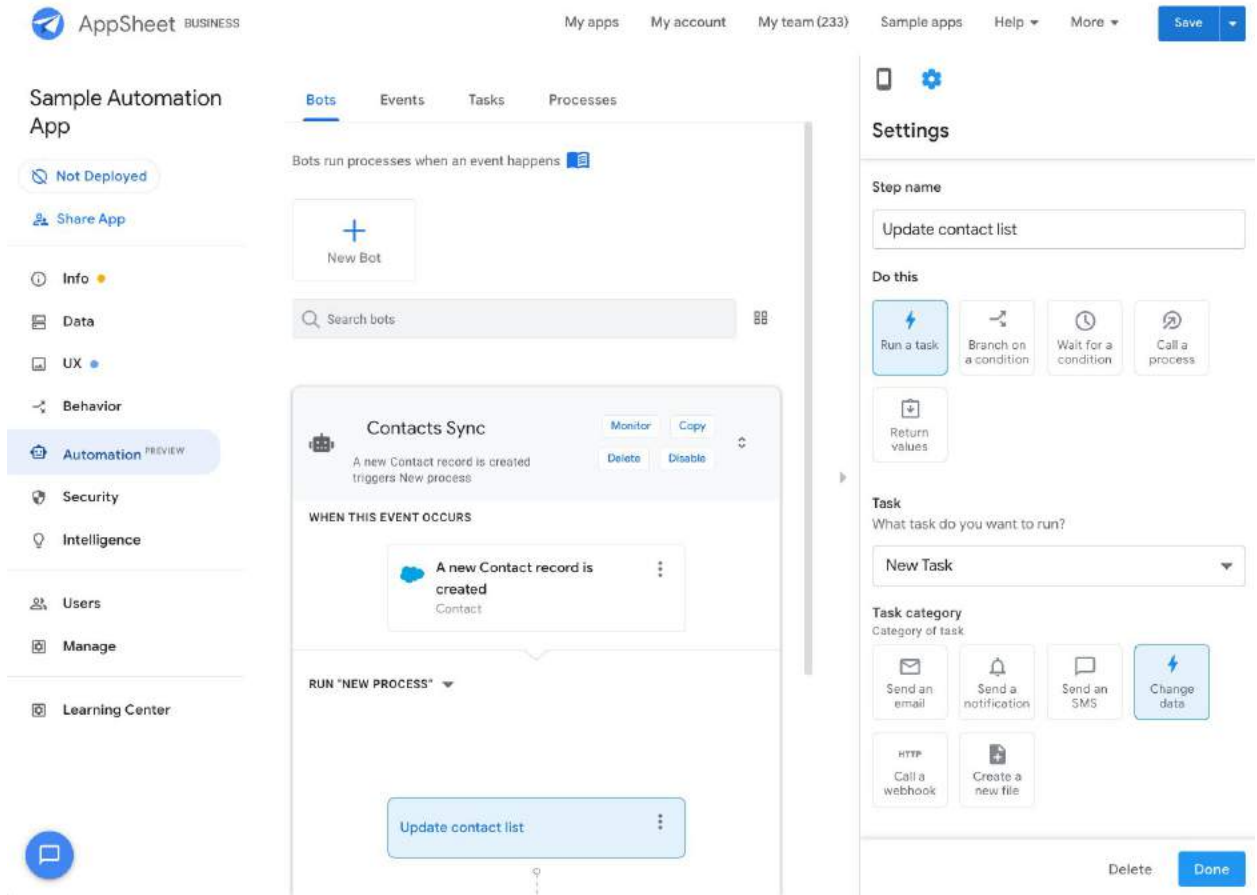
d. Type `Update company contact` as the **Task name**.
e. Select **Contact** as the **Table name**.
f. Select **Update company contact** as the **Data Change Action Name**.

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 step called Notify Ops** from the suggested options.
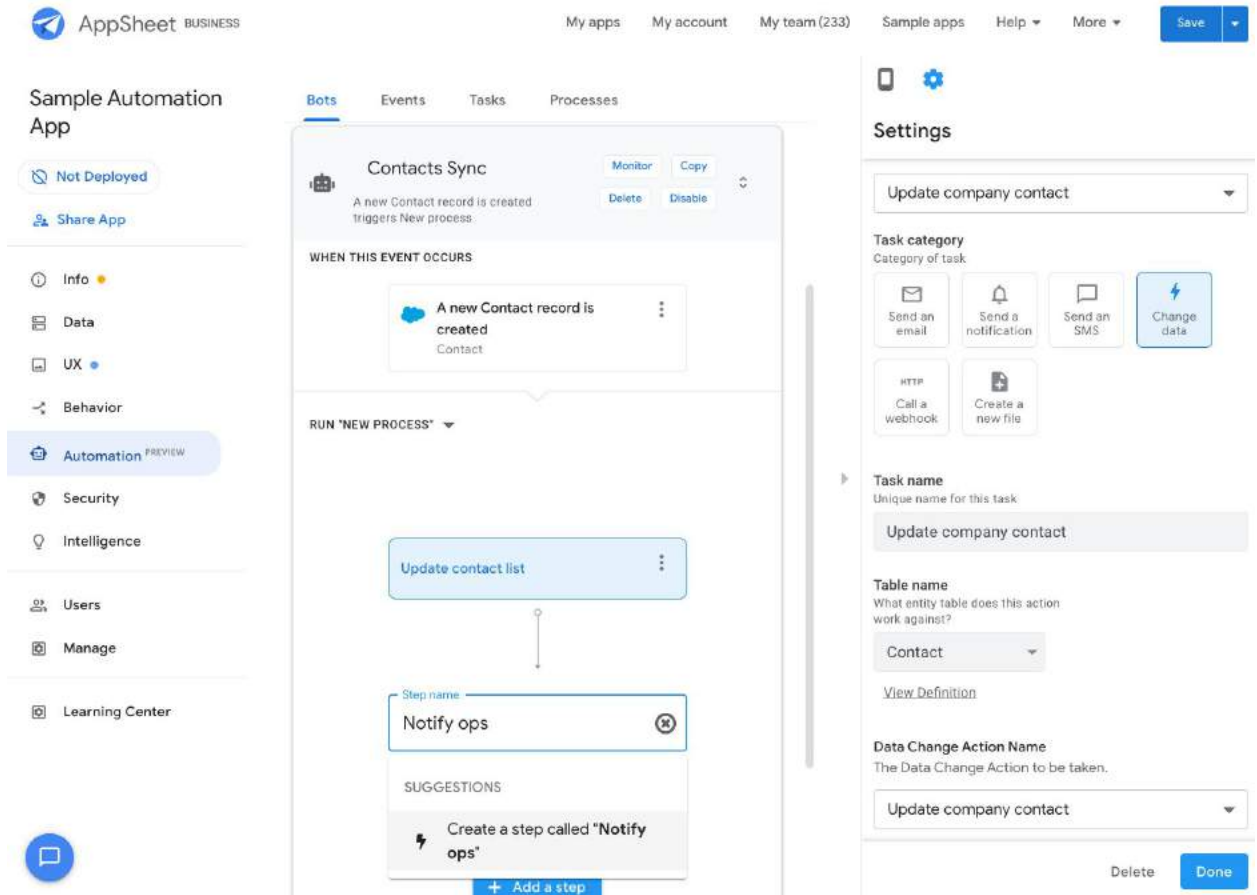
c. In the **Settings** pane, under **Do this** select **Run a task** .

d. From the **Task** dropdown, select **New Task.**

e. Select **Send an email** as the **Task category**.

f. Type `Send an email to ops` for the **Task name**.
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.
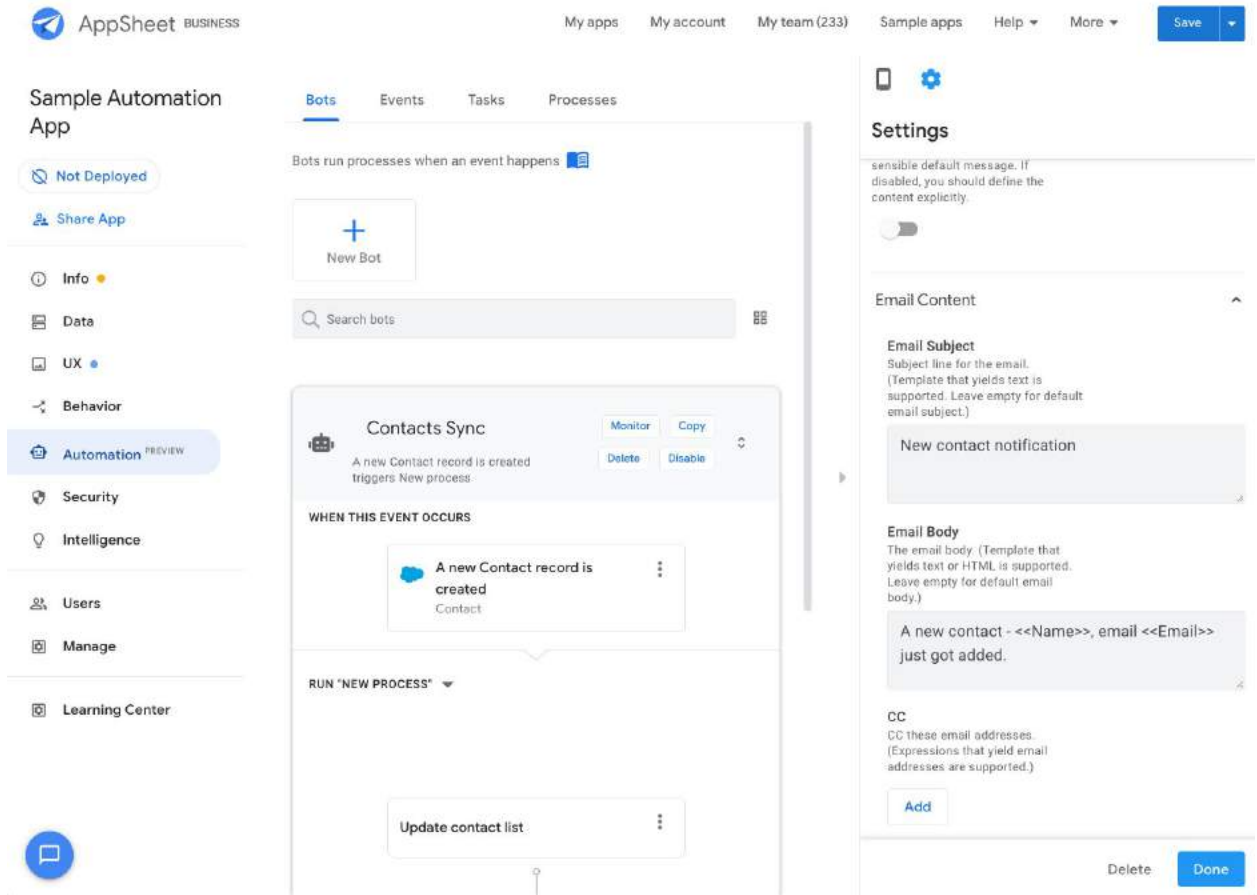
5. Click **Save** and then click **Done.**

Your bot should now look like this:

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.
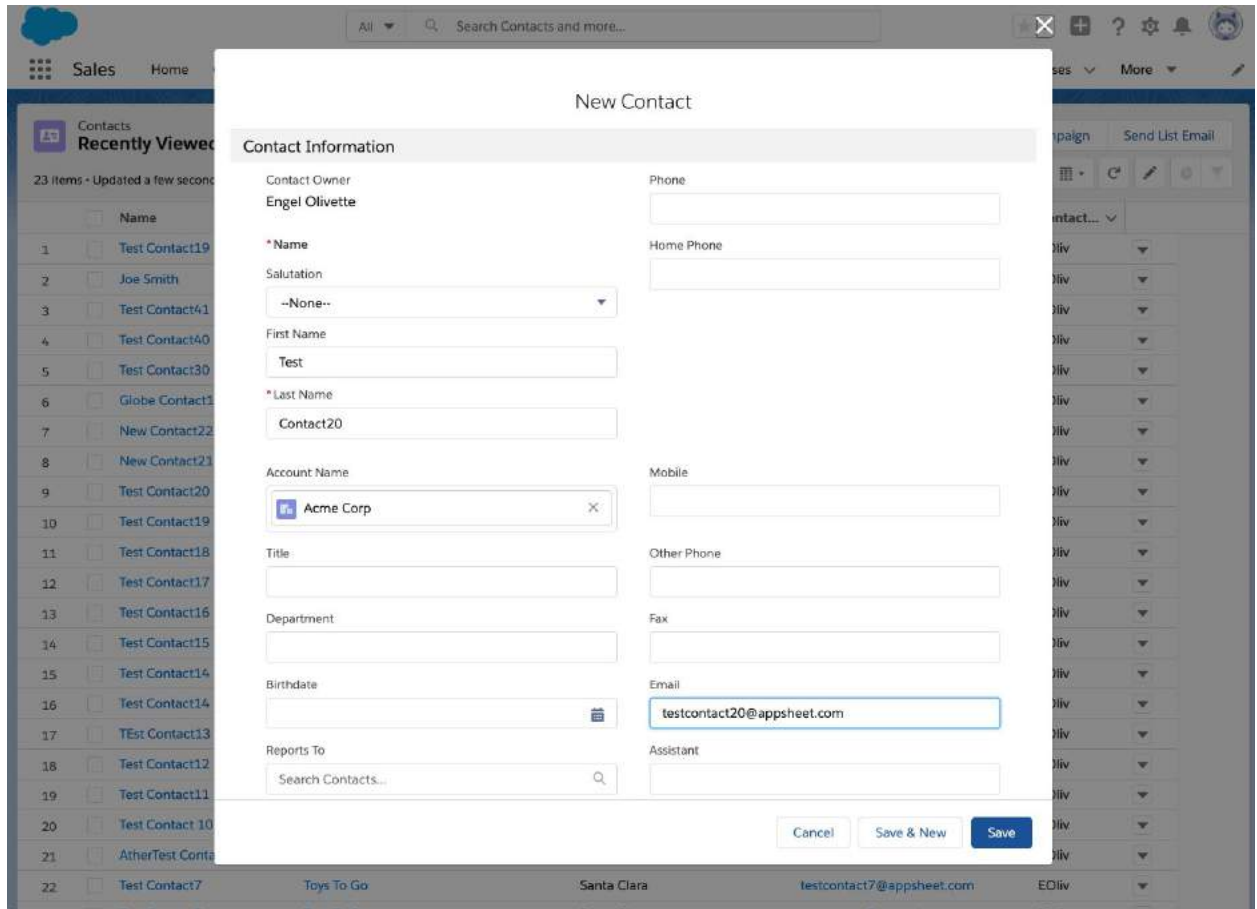
The Bot listens for new Contact events every 60 seconds.

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

4. You should also receive an email similar to this:
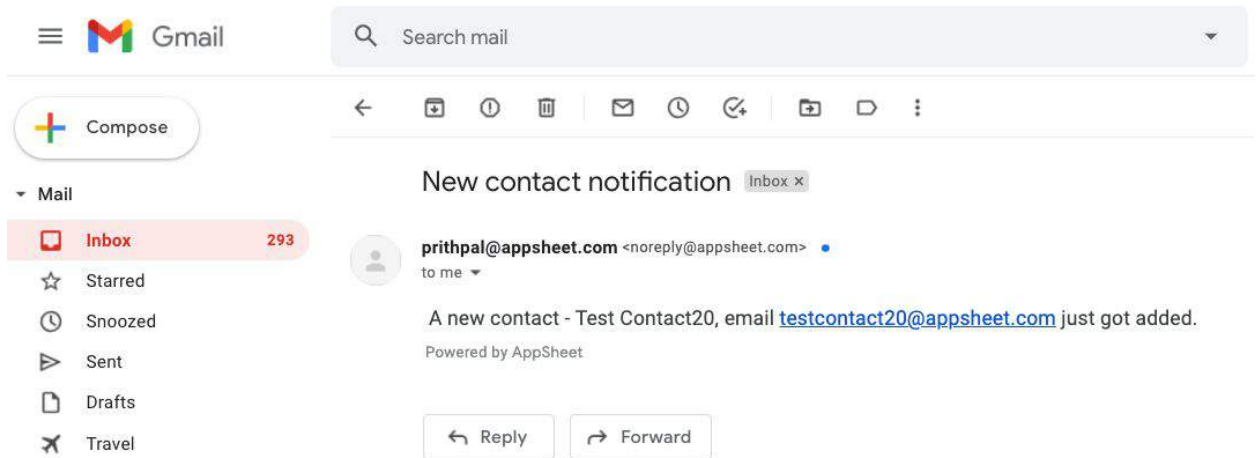
Conference leads appointment generator

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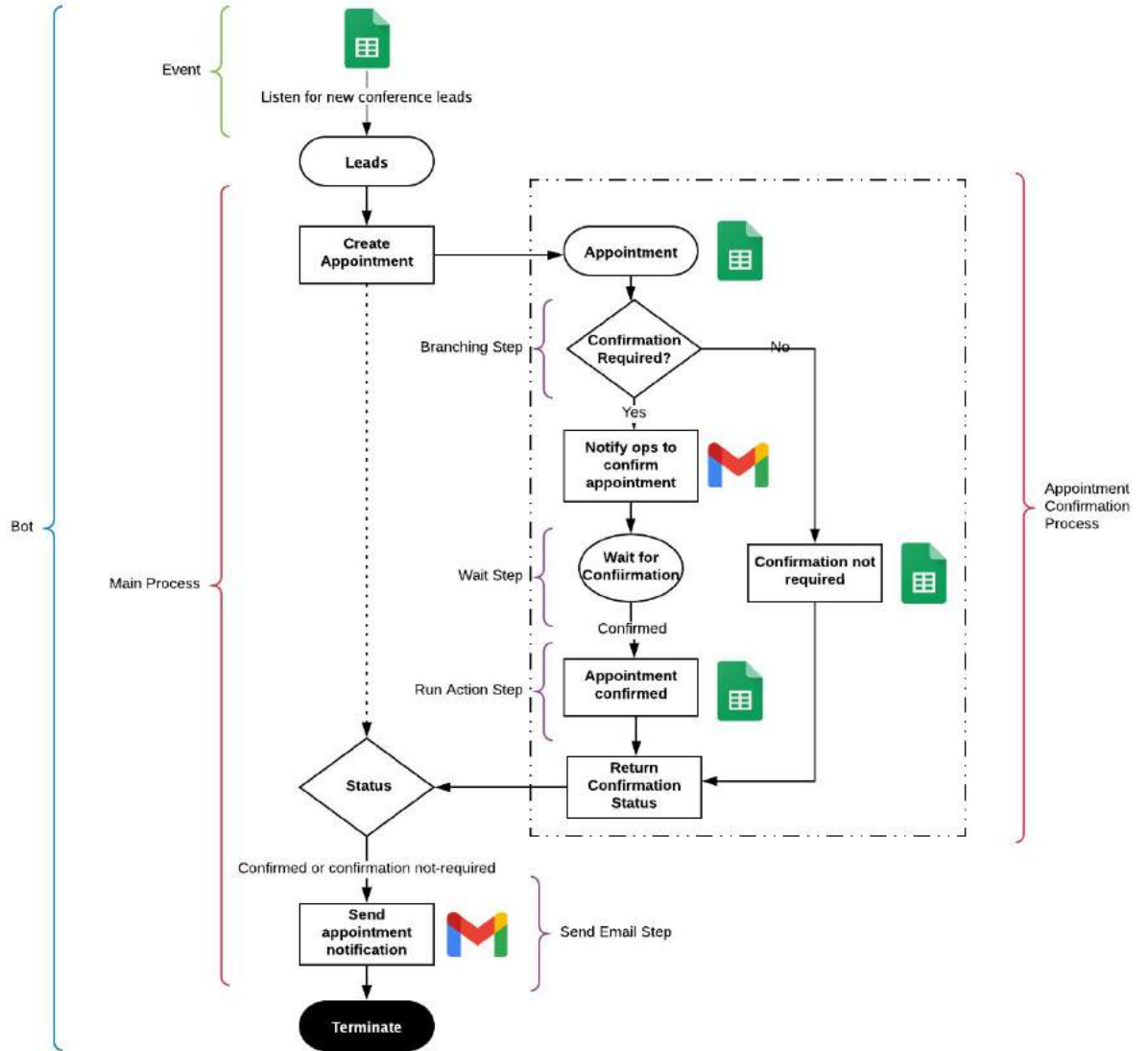
Implementing this use case with AppSheet Automation involves the following steps:

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:
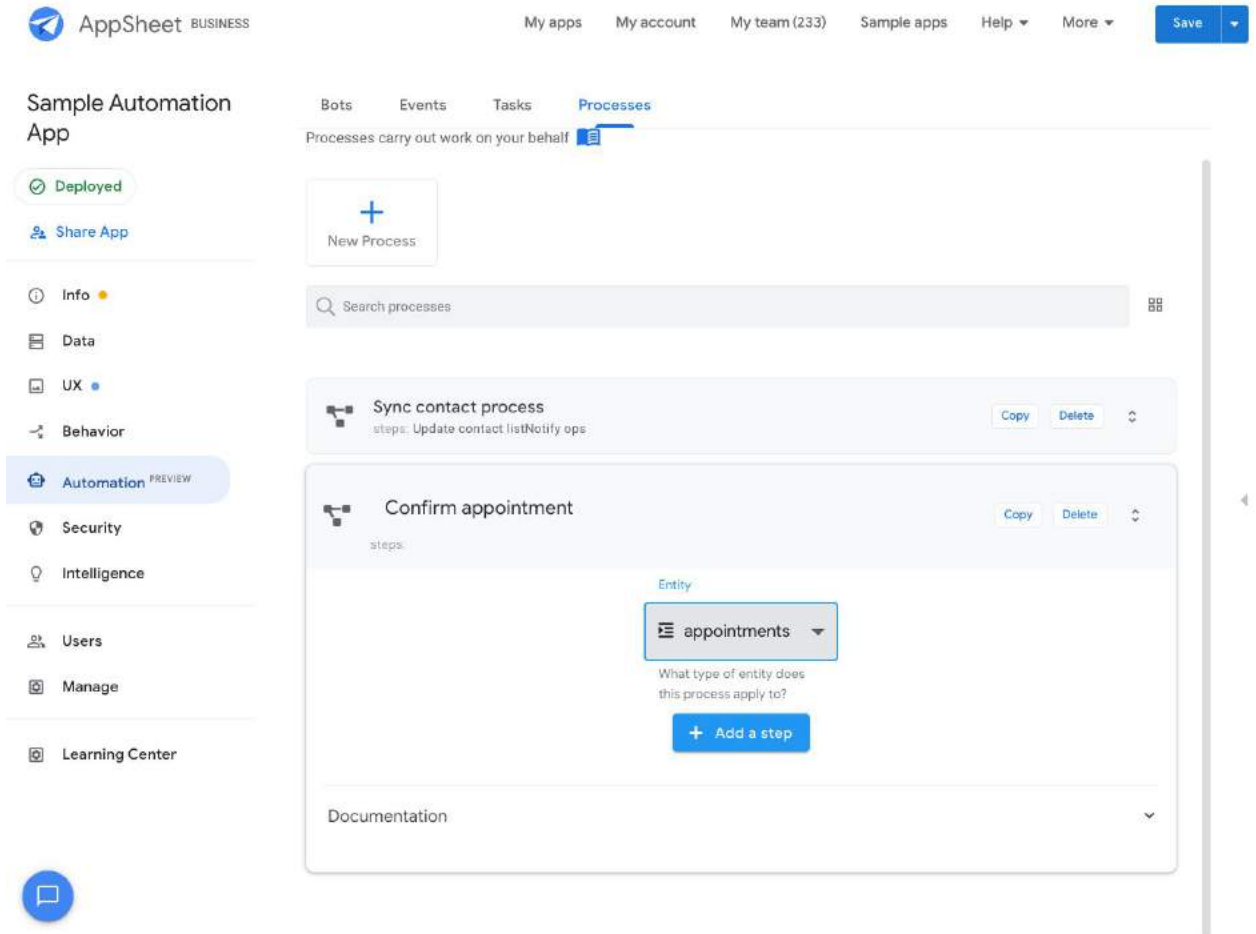
4. To add steps to complete the process, click **+ Add a step.**
5. In **Step name**, type `Is confirmation required?` and select **Create a step called 'Is confirmation required?'** from the dropdown.
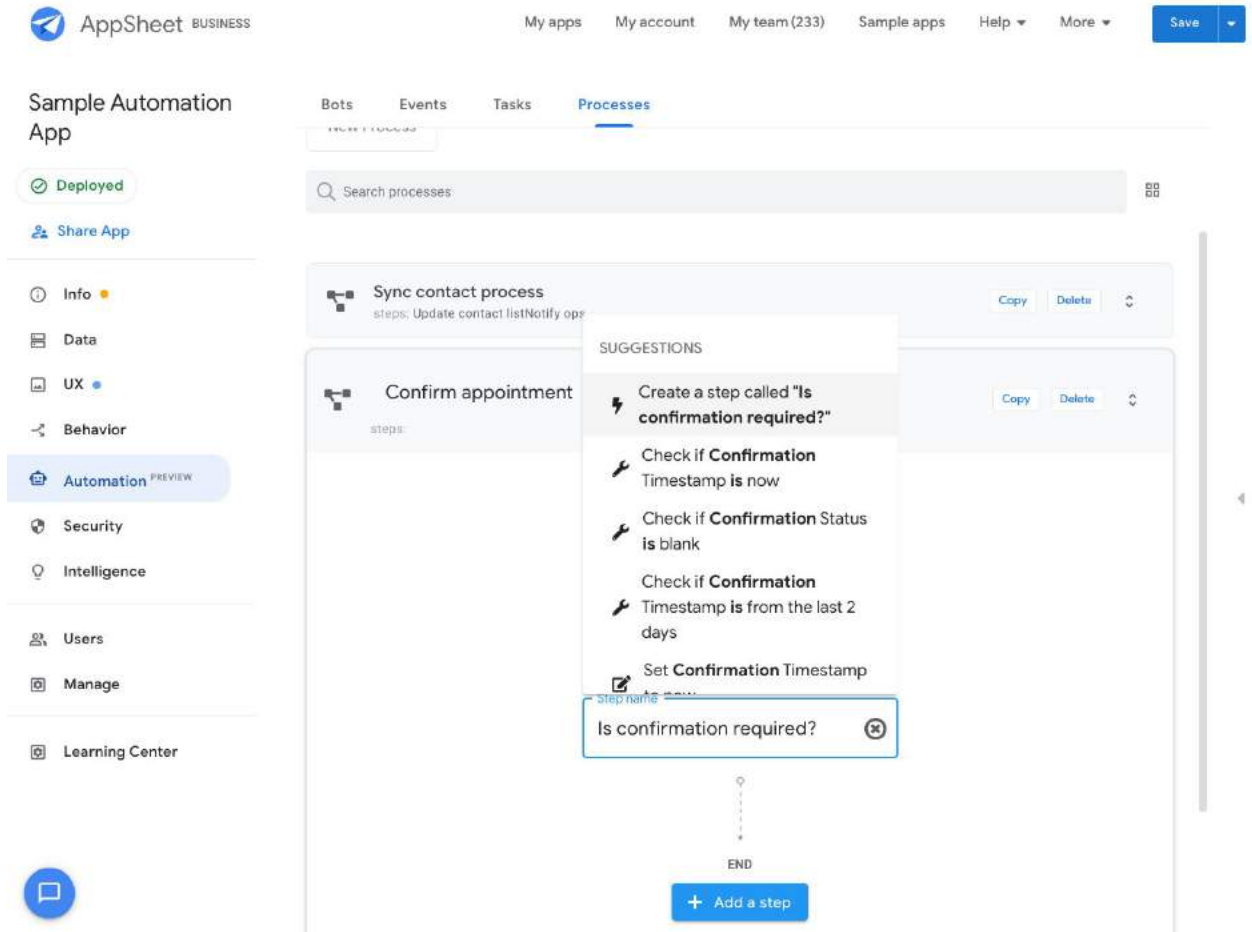
6. In the **Settings** pane, select **Branch on a condition** under **Do this**.

7. Enter `[Confirmation Required] = "Y"` under **Condition to check**. Your step should look like this:
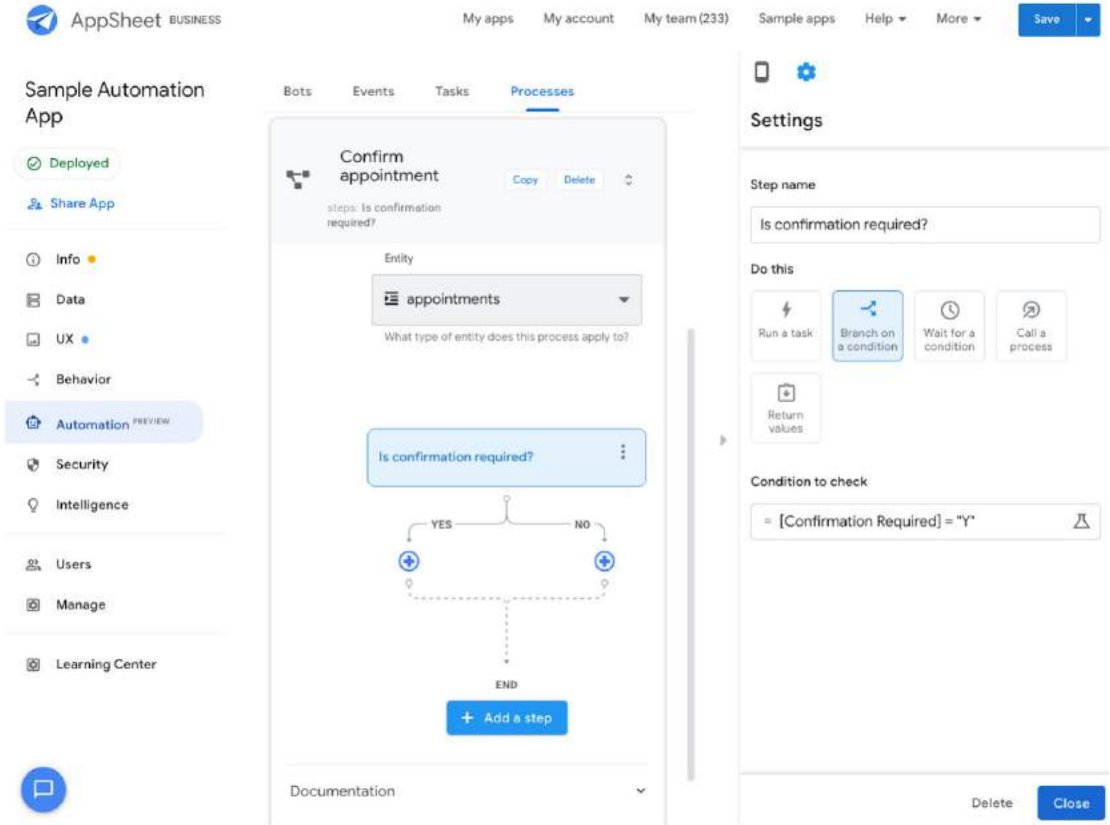
8. Click the **+** icon on the **Yes** branch.
9. In the **Settings** pane, enter `Notify ops to confirm appointment` as the **Step Name** and configure this step as follows:

Field Name	Value
Do this	Run a task
Table Name	Appointments
To	<your email address>
Email Subject	Appointment confirmation request
Email body template	Ops Notification Template (selected from Google Drive) **Note:** *For more information, see Configure an email template for ops notification.*
Use default content?	Disabled

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

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 step called "Wait for appointment confirmation**."
    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:

12. Click **Save** and then **Close**. Your process configuration should look like this:
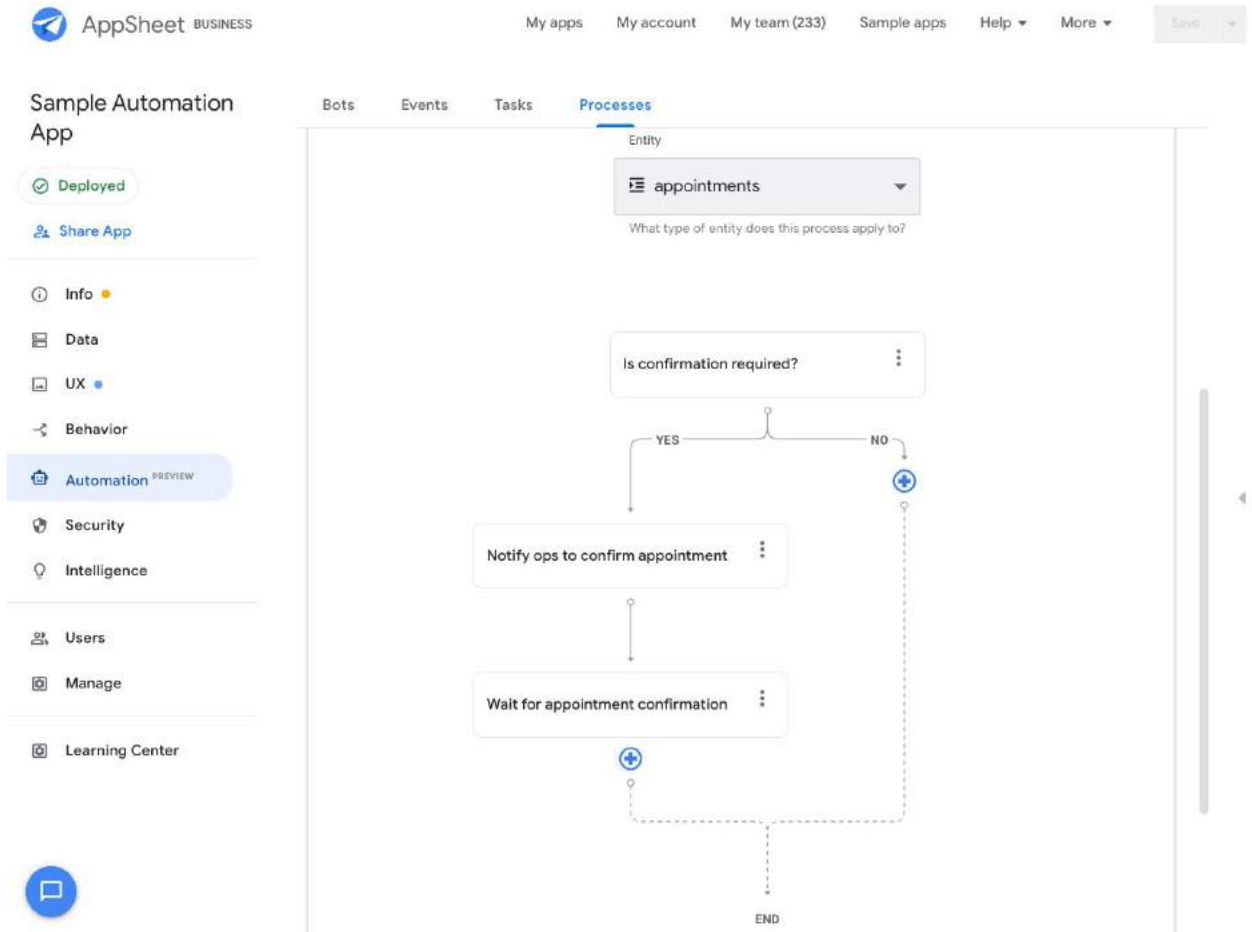
13. Finish the "No" branch of the process.
    a. Click **+** to create a new step.
    b. Type `Update confirmation status to "not-required"` as the **Step Name**.
    c. In the **Settings** pane, configure the fields as follows:

Field name	Value
Do this	Run a task
Task	Create new task
Task category	Change data
Task name	Update confirmation status
Table name	appointments
Data change action name	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:
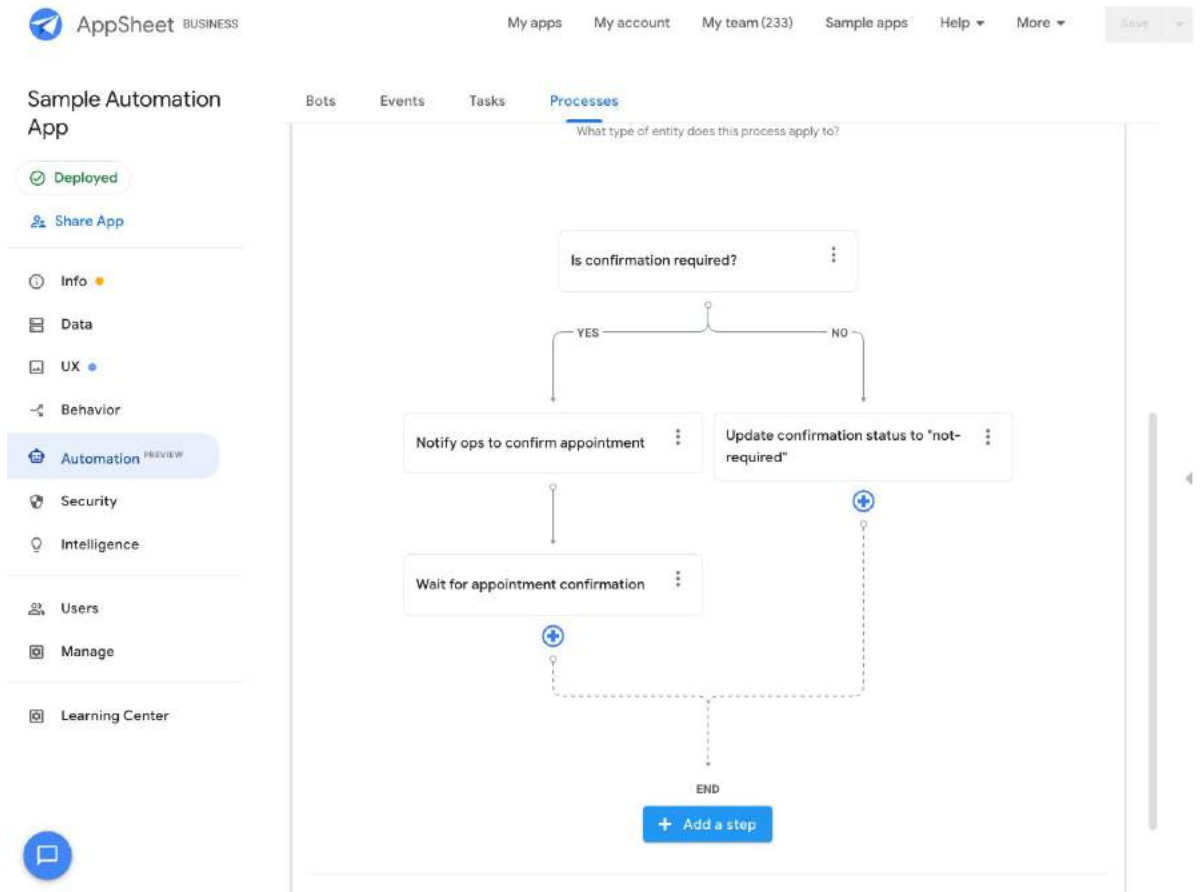
14. Add a final step to return the **confirmation status** value used by the main process to make further decisions.
    a. Click **+ Add a step** to create a new step.
    b. Set **Step name** to **Return confirmation status**.
    c. In the **Settings pane,** under **Do this**, select **Return values**.
    d. Type `ConfirmationStatus = [Confirmation Status]` in **Return these values**.

e. Click **Save** and then **Close**. Your process should now look like this:
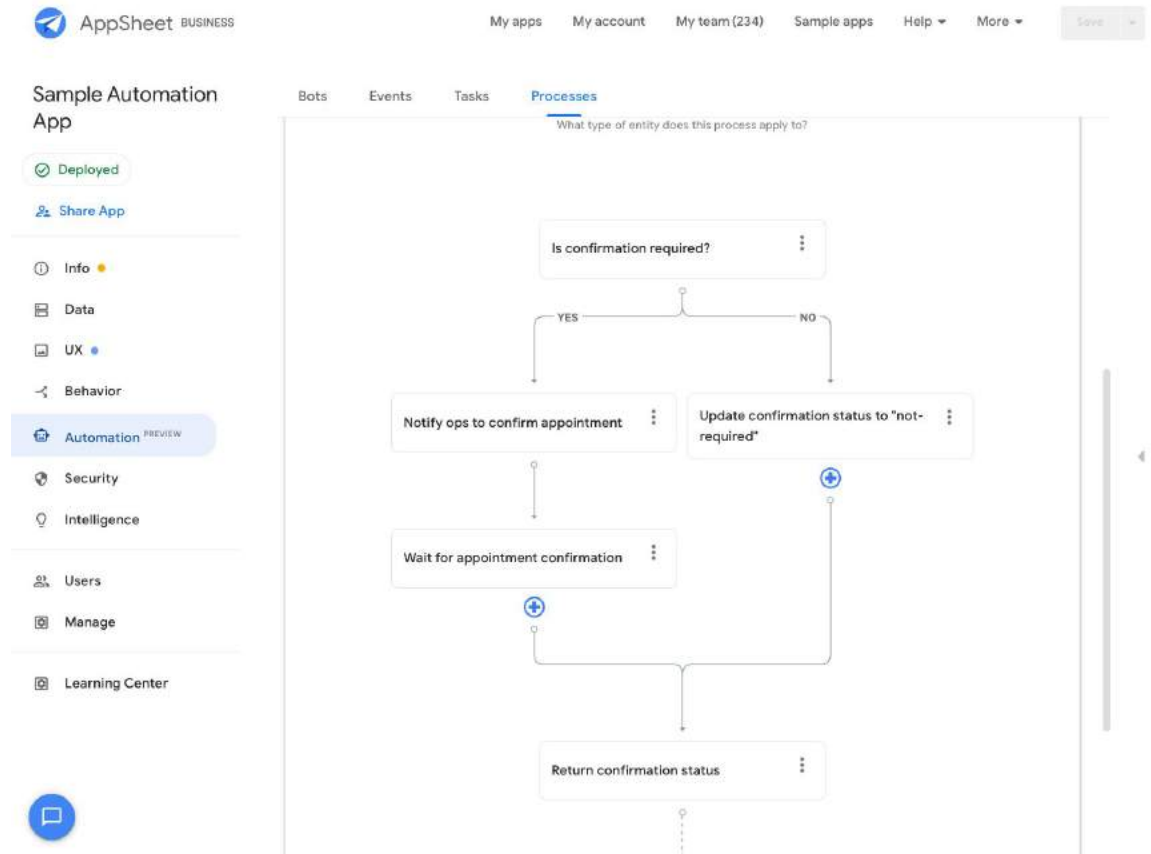
Creation of the Appointment Confirmation Process is complete!

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 empty bot named 'Conference leads appointment generator'** 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 select **Edit** from the drop down list.
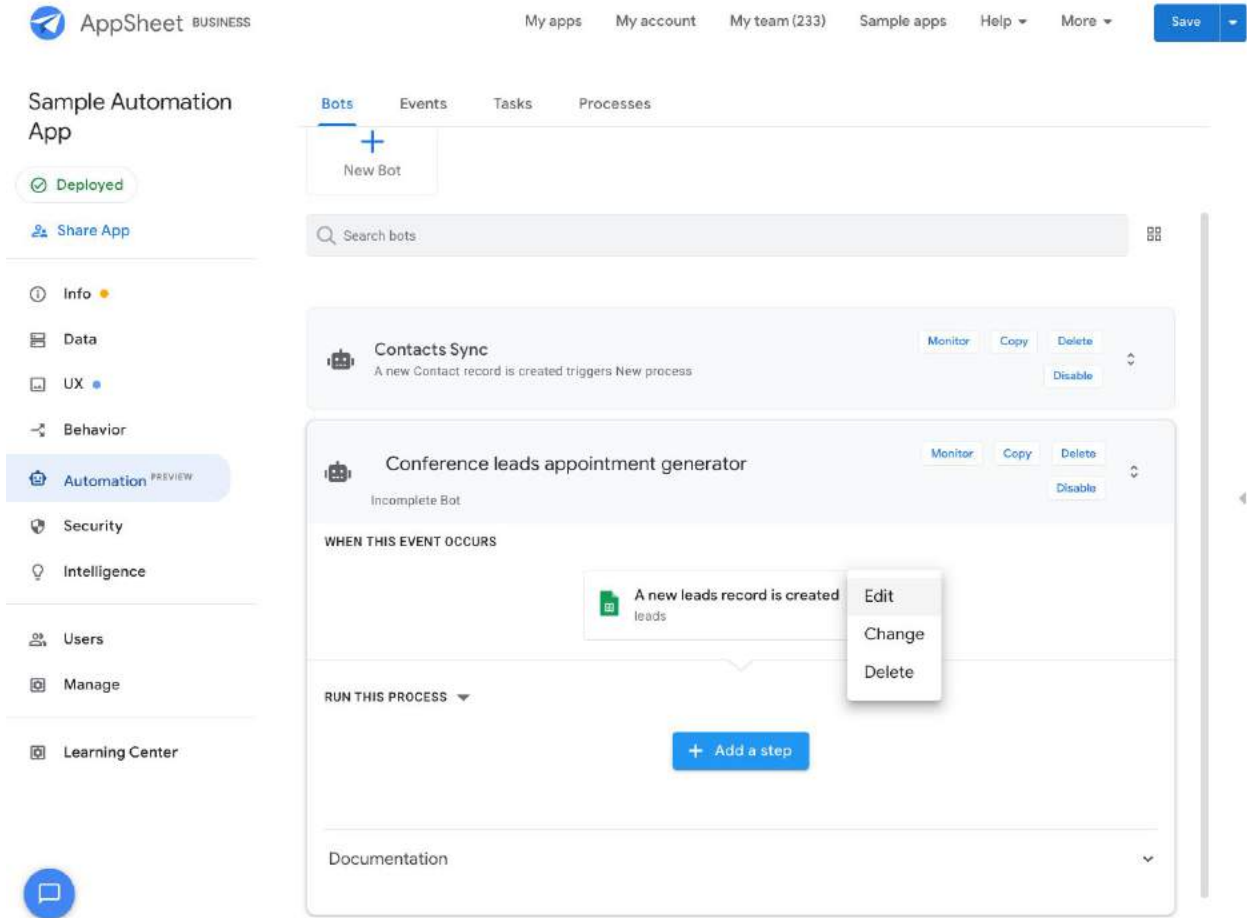
6. On the **Event editor** page, configure the event as shown below:
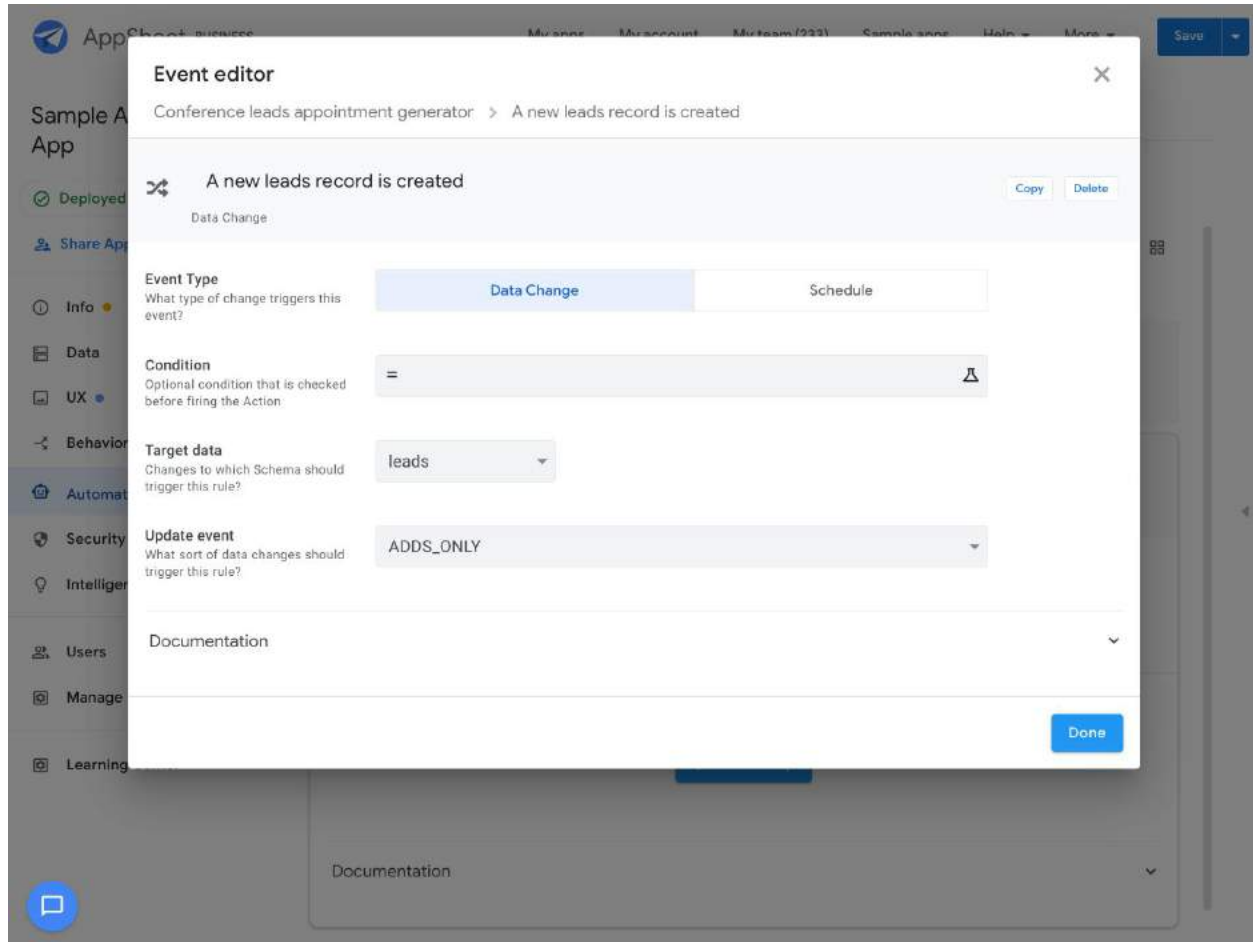
7. Click **Done.**

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 hit **Enter** to automatically create an empty step. This implicitly creates a process.
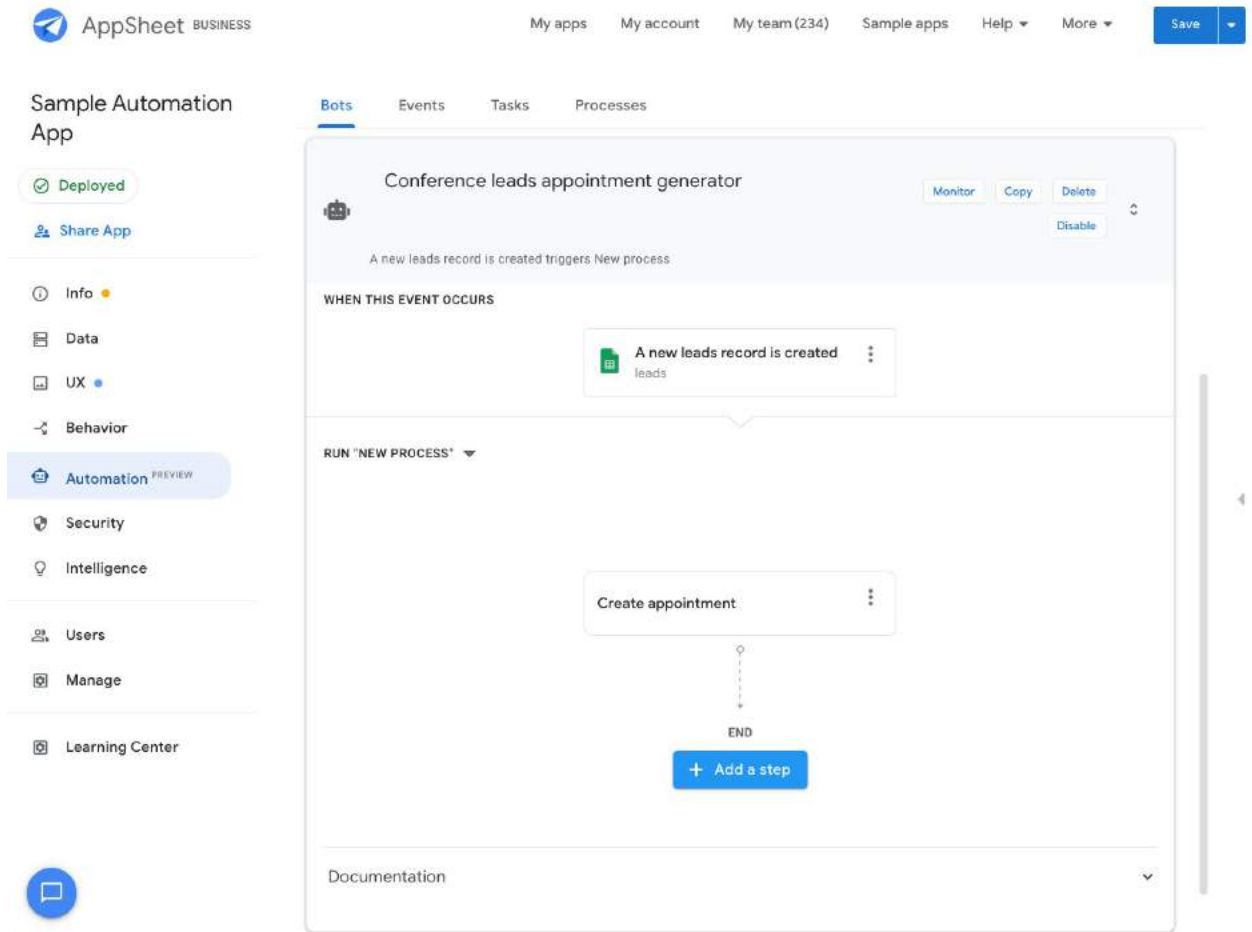
3. In the **Settings** pane, configure the fields as follows:

Field	Value
Do this	Call a process
Process name	Confirm appointment
Process inputs	- Full name: CONCATENATE([First Name], " ", [Last Name]) - Email: [Email] - Appointment date: [DATE]+[FollowupAfterDays] - Confirmation required: "SWITCH(UPPER([Profile]), "HIGH", "Y", "N")"

Your step config should look like this:

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

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 create a step with that name.

   c. In the **Settings** pane, under **Do this**, select **Branch on a condition**.

   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:
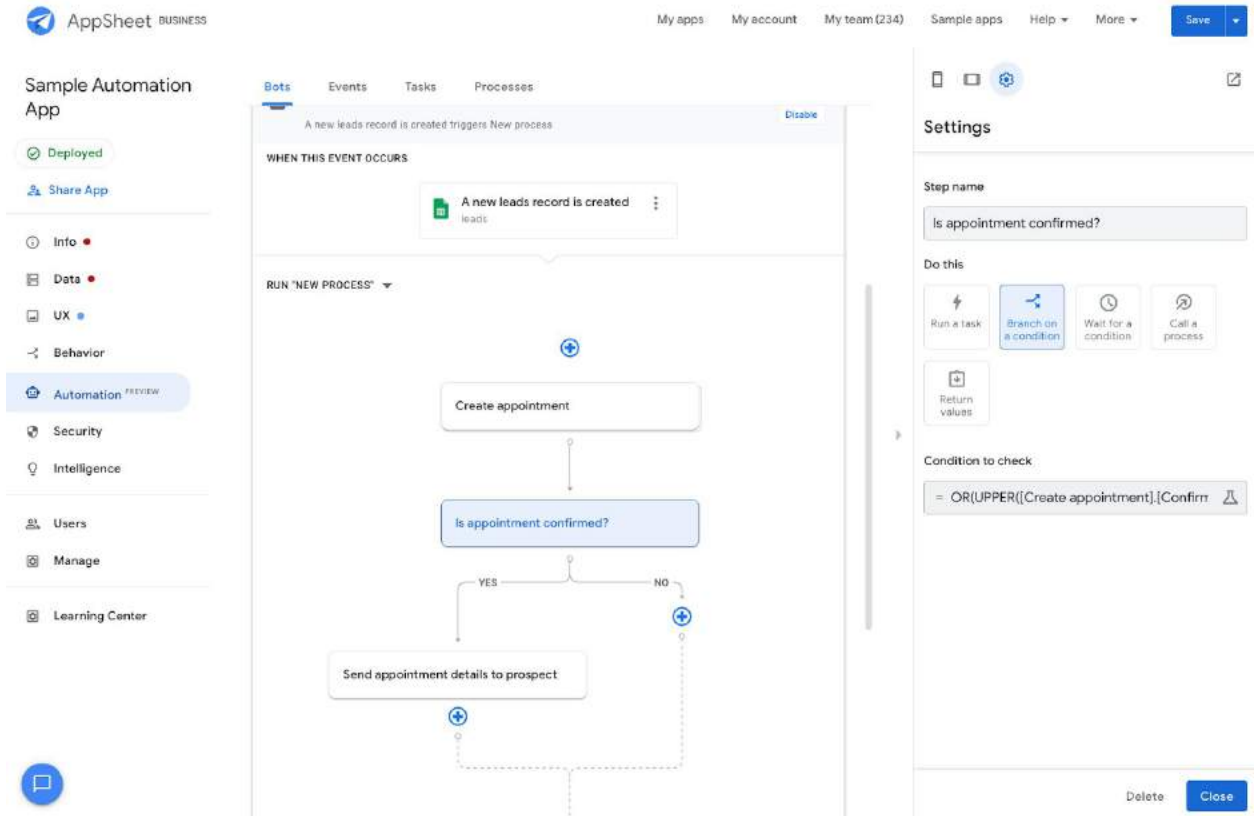
6. Add one final step to the process.

   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.
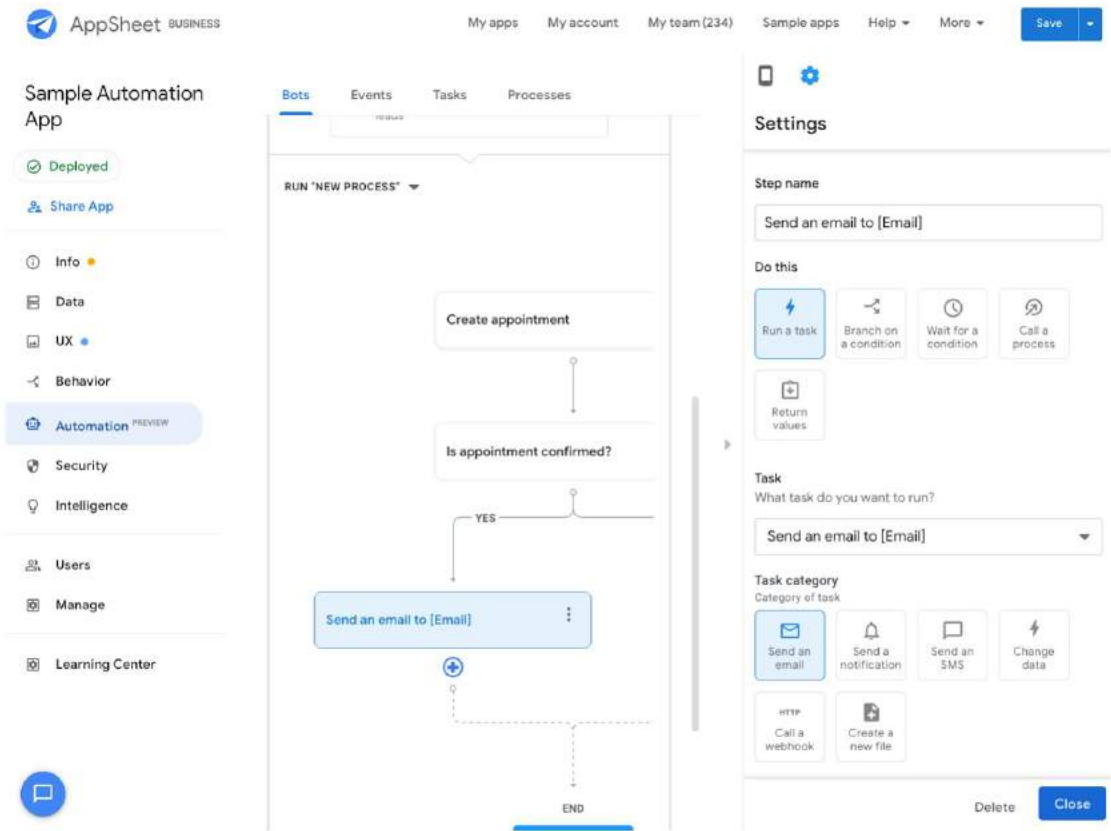
c. Configure the remaining fields as follows:

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.

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.

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

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
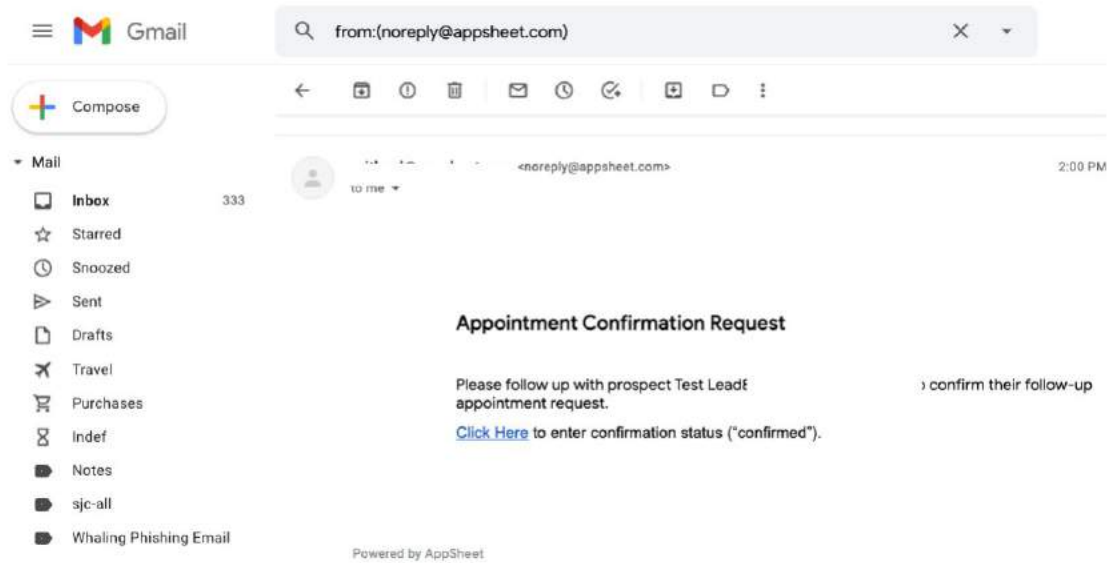
5. Click **Click Here**.

6. When the form opens, nce on the form, click the **pencil** icon.

7. In **Confirmation Status**, type `confirmed`.
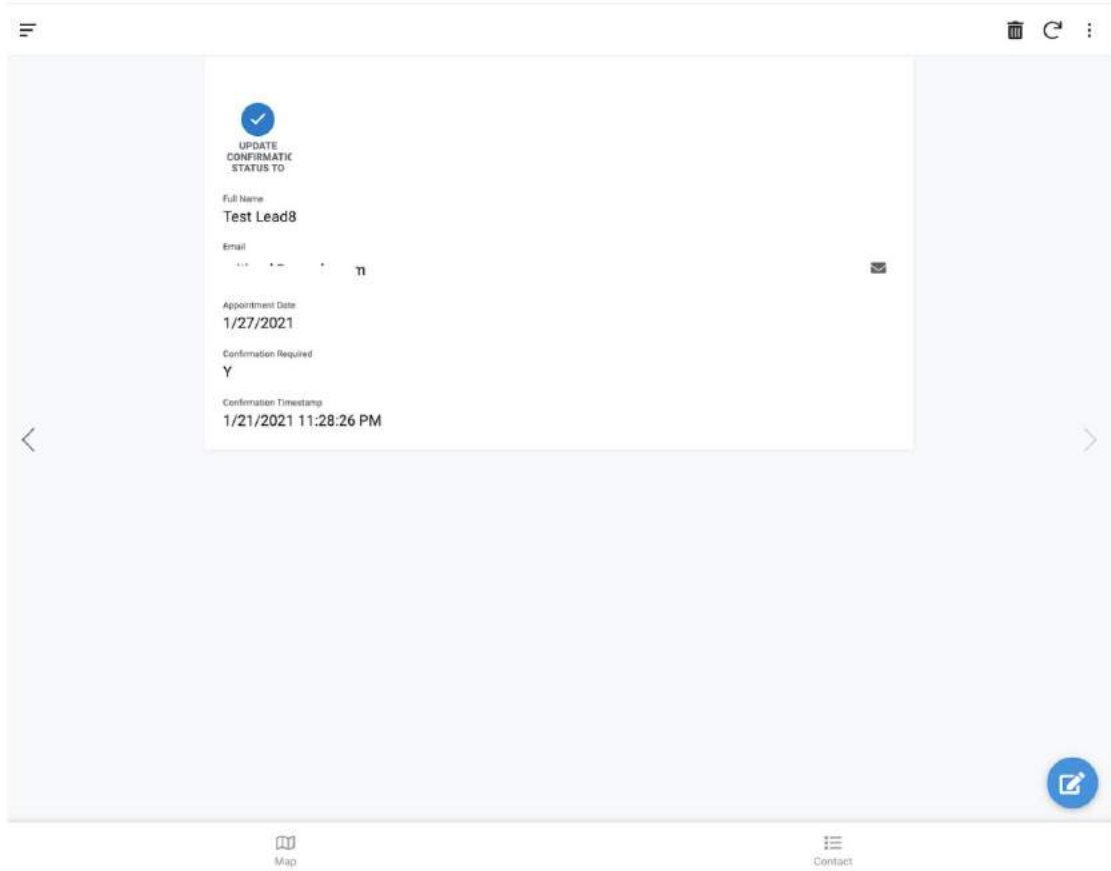
8. Click **Save**.

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 Preview 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.

87

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**.

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 Preview release, the UI shows a loading indicator until the processing is completed.

Upon successful completion, the screen looks like this:

There are some limitations to keep in mind while using this capability. These limitations are

Google Confidential – Internal Use Only (Do not print, distribute or share without explicit permission)　　　　89

likely to change over time as the service matures.

Note: Limitations for Preview release:

- *Processing only works for 1,000 documents for a given table. New files are not processed after 1,000 files have been processed.*
- *Documents need to be under 20 MB and 5 pages to be processed. Otherwise the file processing fails.*
- *The only extension types supported in preview are* `.tiff, .gif, .pdf`

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.

In the Preview release, the user performing the manual review can create a new slice identifying the documents that need attention using this expression:

```
AND([NeedsAttention], [StatusCode] != "EXTRACTION_ERROR")
```

With the slice created, the user can build a view to see the processed invoices that need attention. Users can also leverage the capabilities of process automation to detect "adds" inside a filtered table and trigger additional flows.

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 are captured by a 'Failed Invoices' slice with the following expression:

```
AND([NeedsAttention], [StatusCode] == "EXTRACTION_ERROR")
```

A view can be created to see which documents failed. Users can also leverage the capabilities of process automation to detect "adds" inside the slice and trigger additional flows.

For more details about status codes and how to use them, see [Document processing states and status codes](#).

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%20floor plan.jpg", as shown below .

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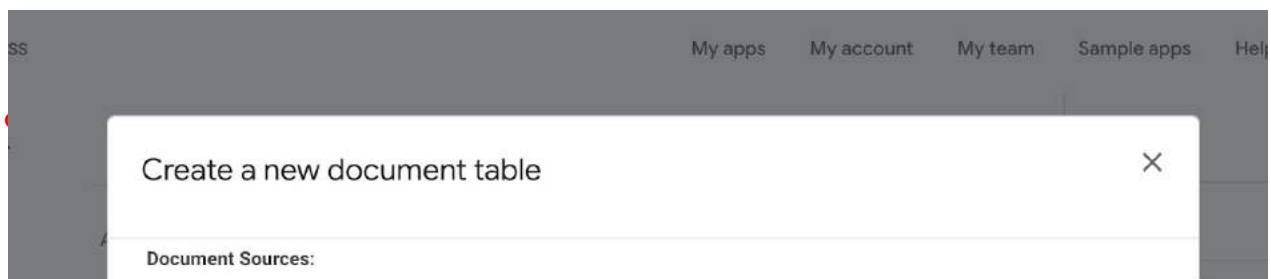
7. Select **Create Table**. This will create a table named *User manuals*. The table name is, based on the folder name selected.

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

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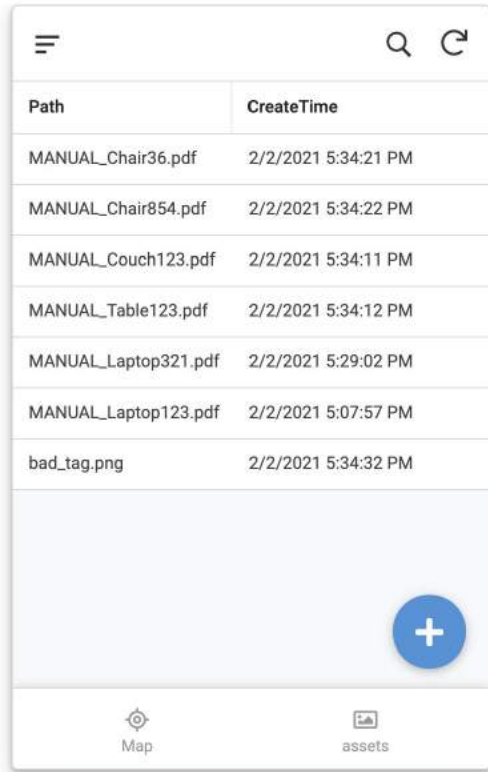
- List files
- Click on the list to open a file

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:

This view displays to app users like this:

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.

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), "")
```

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:

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)**.
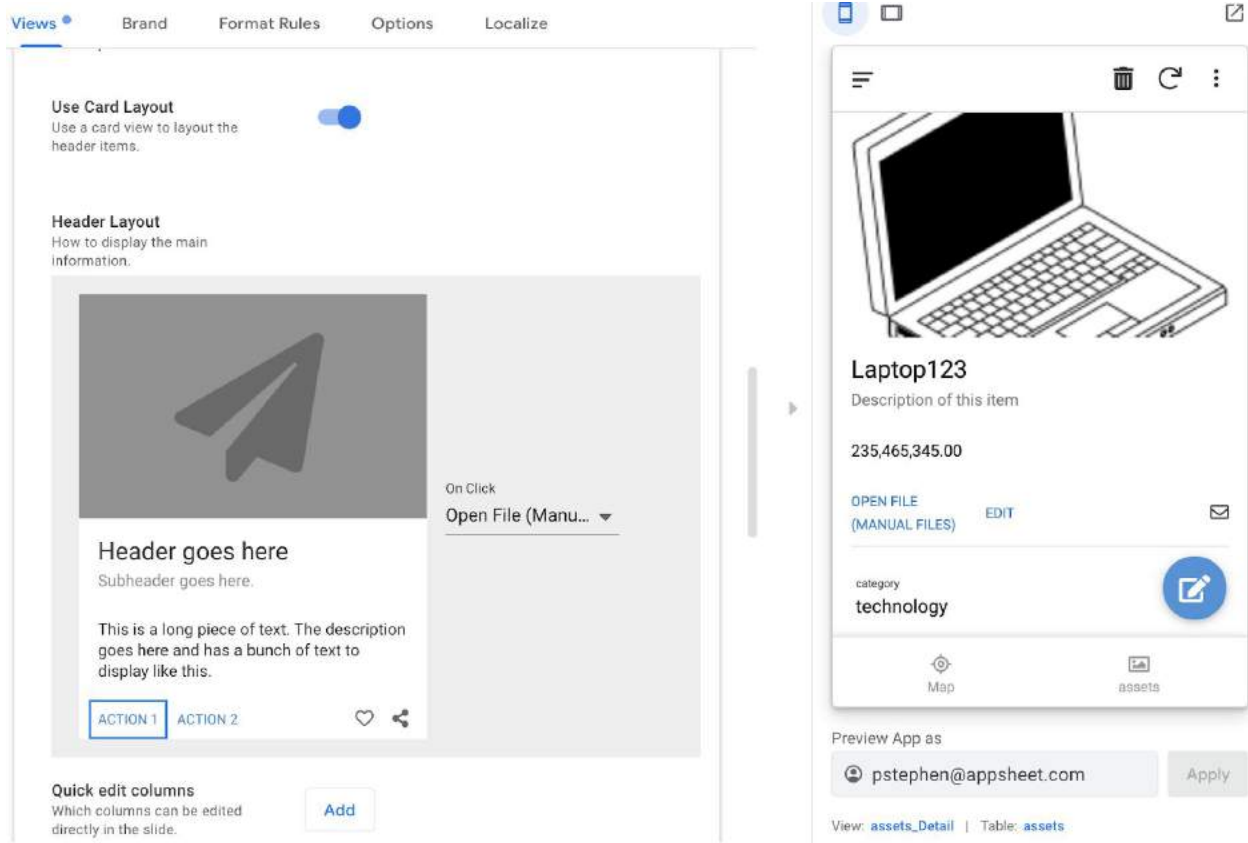
Appendix

Configuring Employees entity from Google Sheets datasource

To configure your "employees" entity from Google Sheets:

1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.
3. Select your configured Google Data Source.
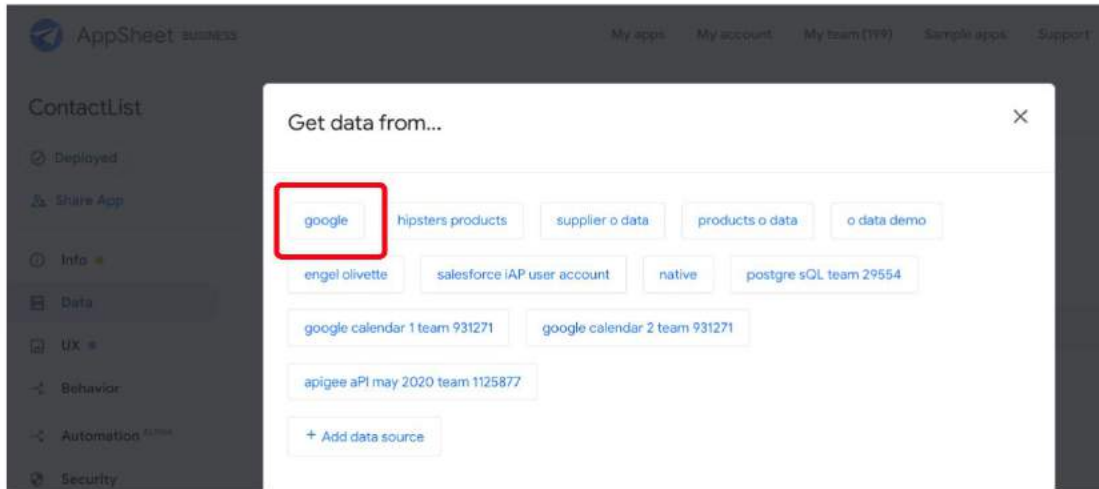
4. Use the file browser to select the appropriate **New Employees** Sheet as the data source.
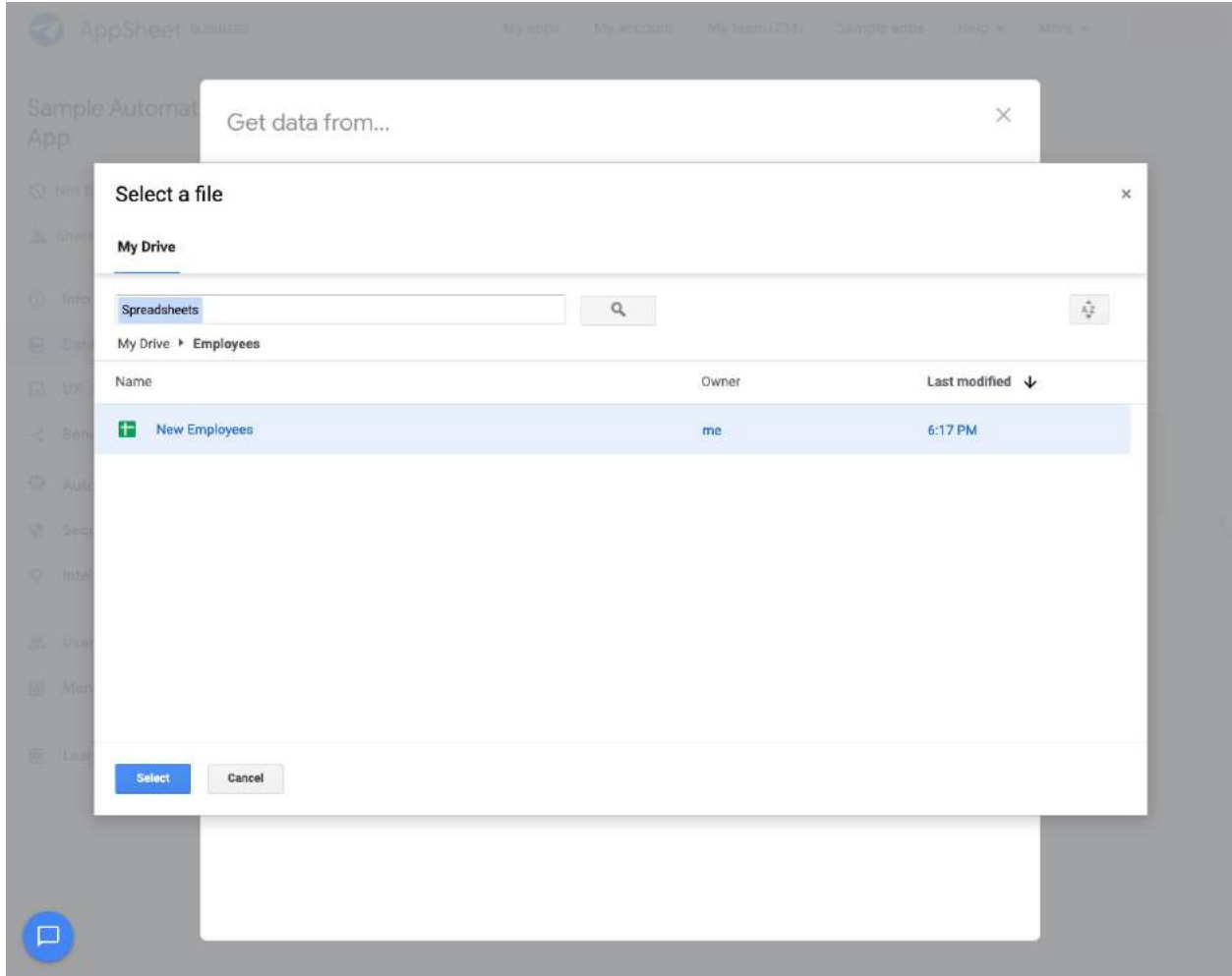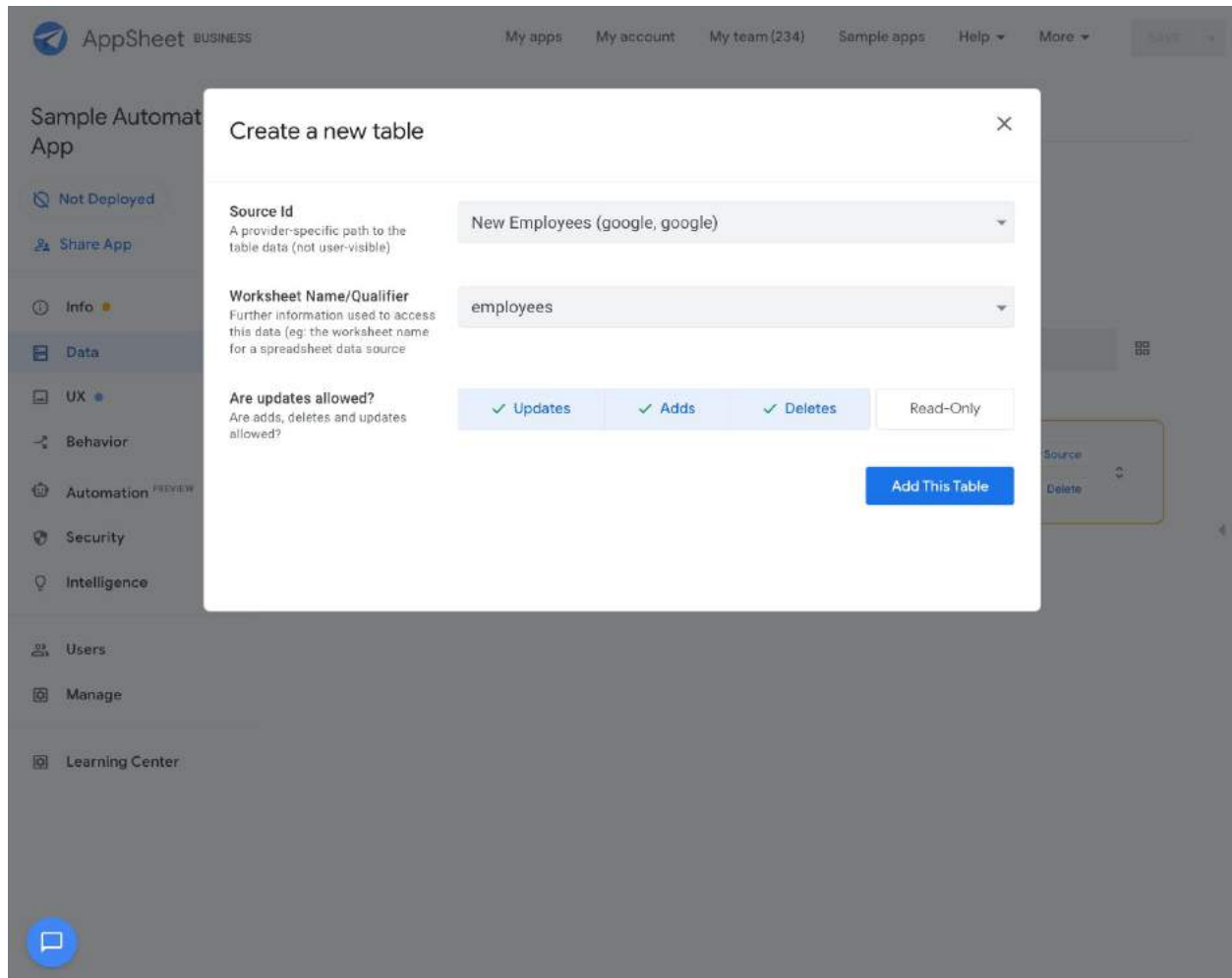
5. Select **employees** as the table for your entity.

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/appsheet_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-RCalr-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.
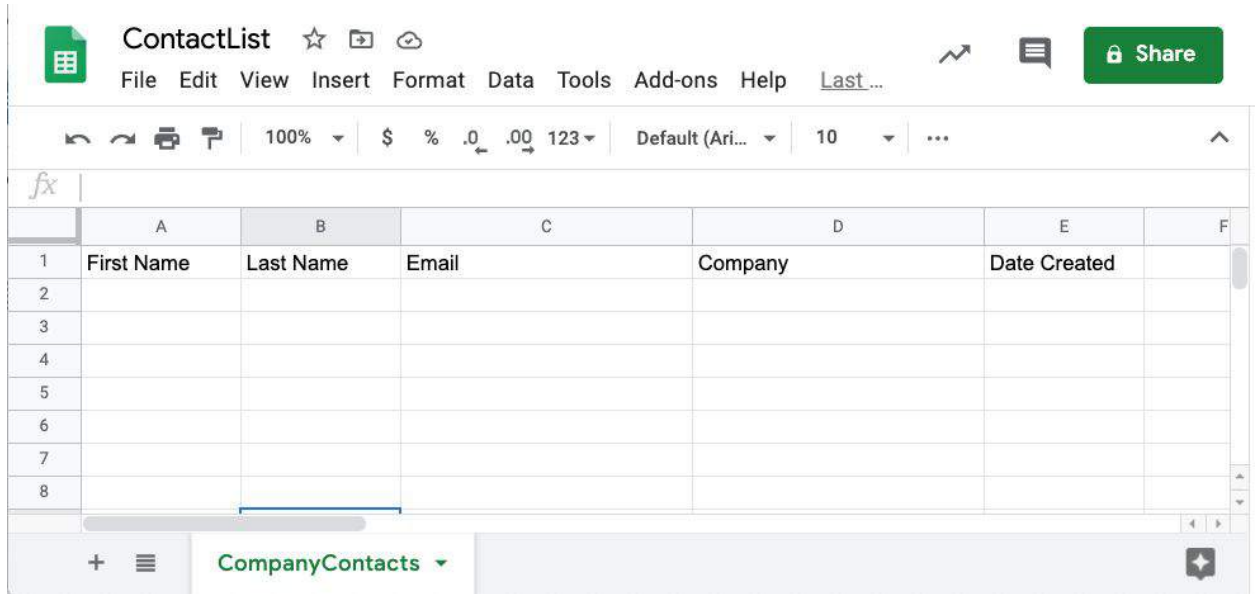
5. Select **Company Contacts** as the table for your entity.

Configuring Contact entity from Salesforce datasource

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

Salesforce.
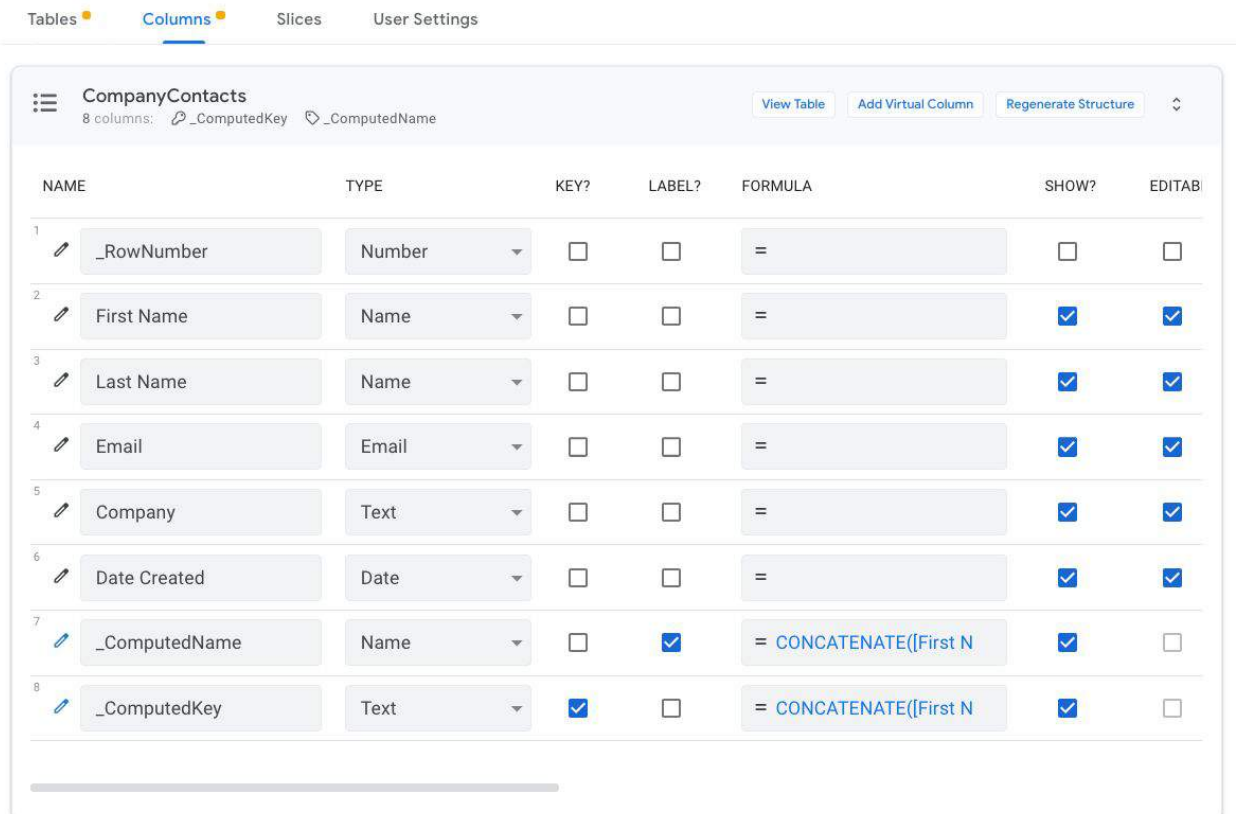
Before you begin:

- 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.
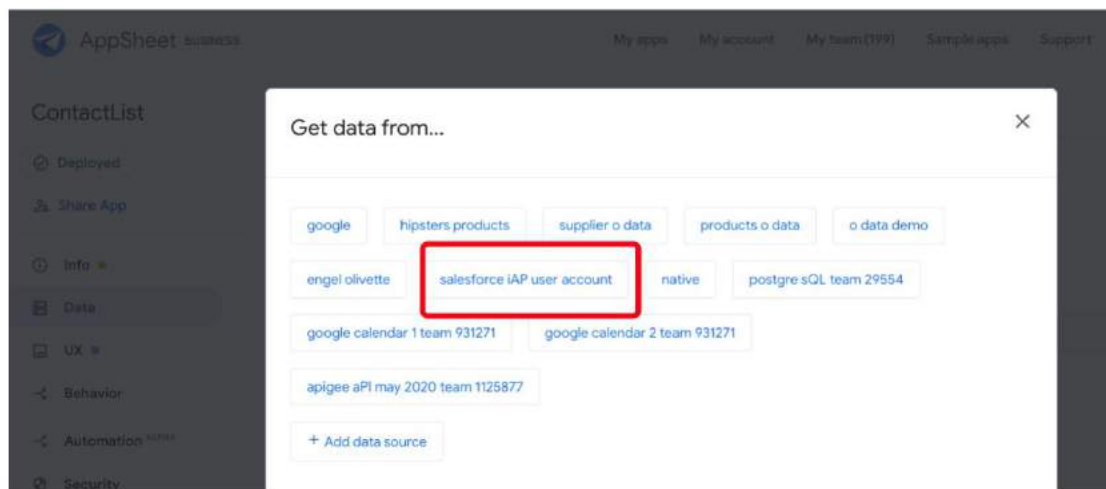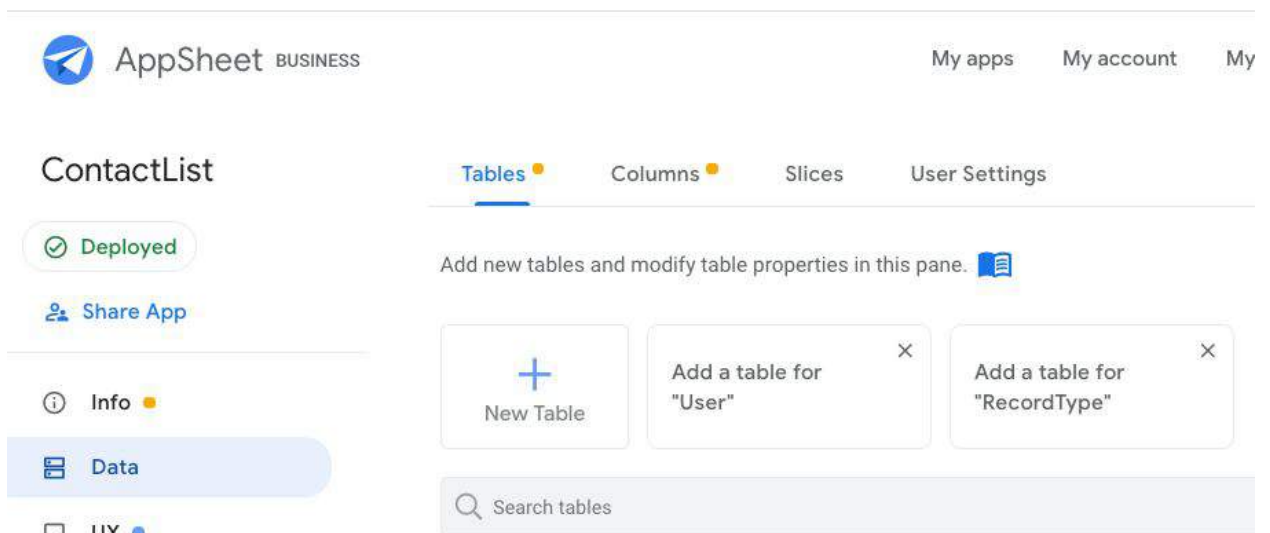
4. Select **Contact** as the table for your entity.
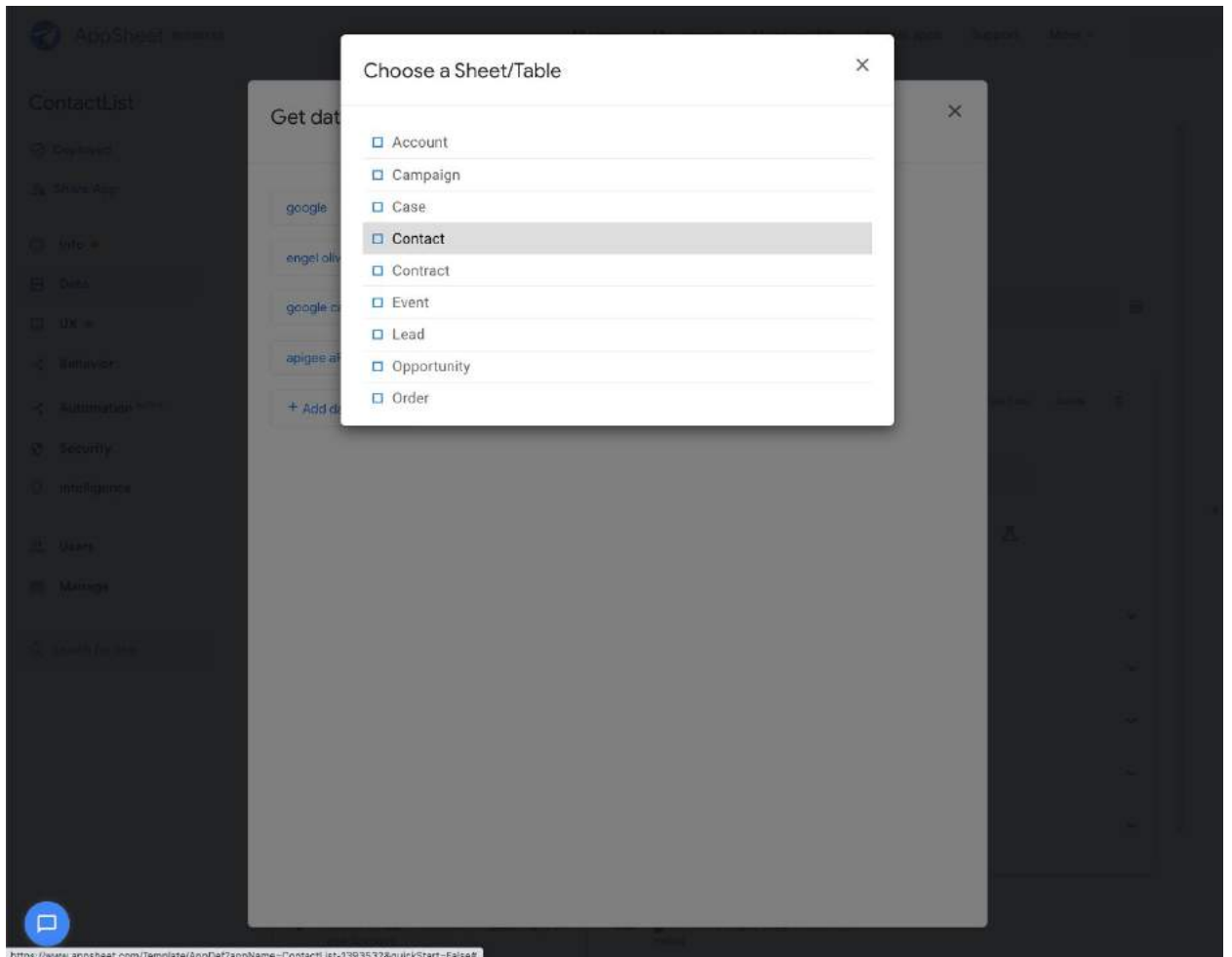
5. For **Are updates allowed?** change the selection to **Read-Only**.

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

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

Configuring Leads entity from Google Sheets datasource

To configure your "Leads" entity from Google Sheets:
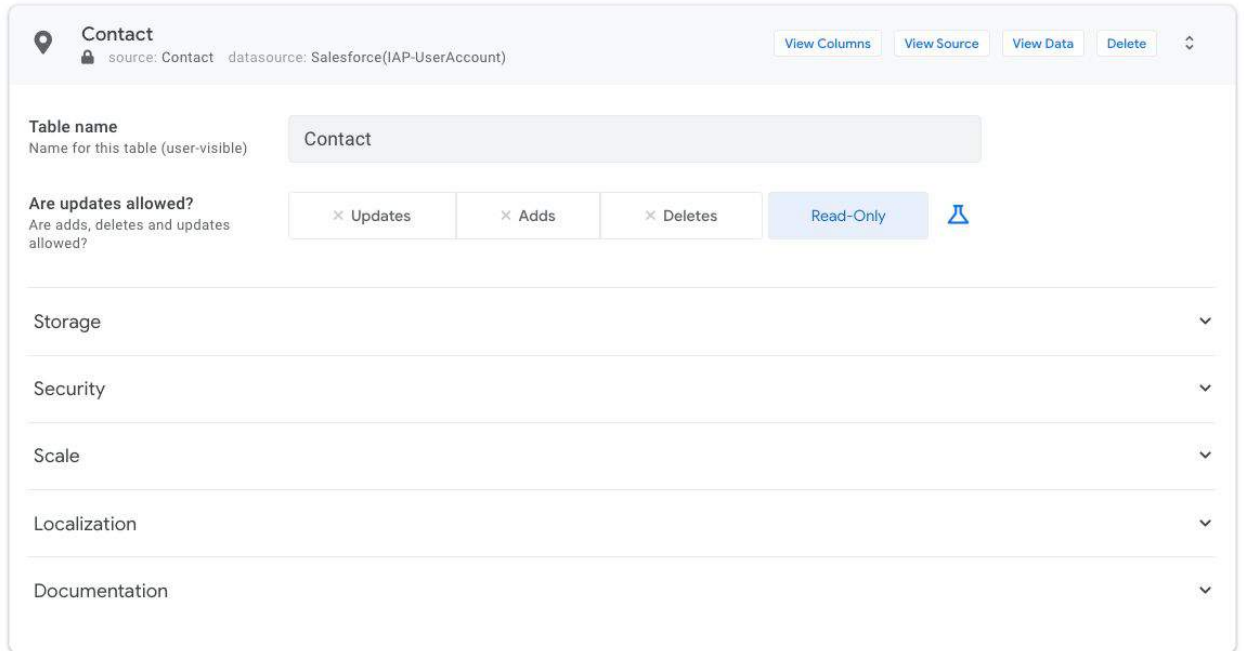
1. From the AppSheet UI, click **Data** in the left navigation bar.
2. Click **New Table**.
3. Select your configured Google Data Source.
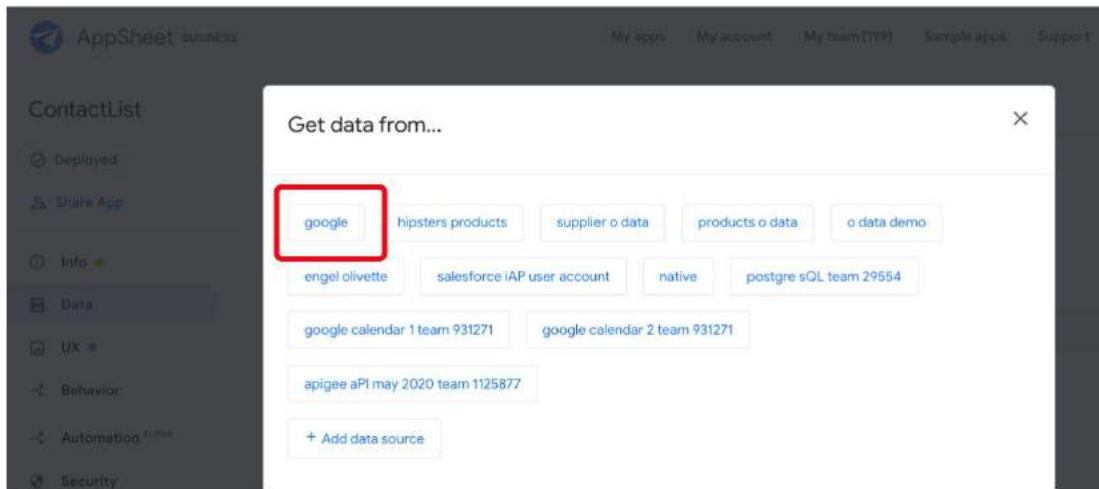
4. Create a new Google Sheet named "Leads." Use the column names shown below and name the Sheet (tab) "leads". The Sheet should be in the Google Drive accessible to your AppSheet account.

5. Use the file browser to select the appropriate **Leads** Sheet as the data source.
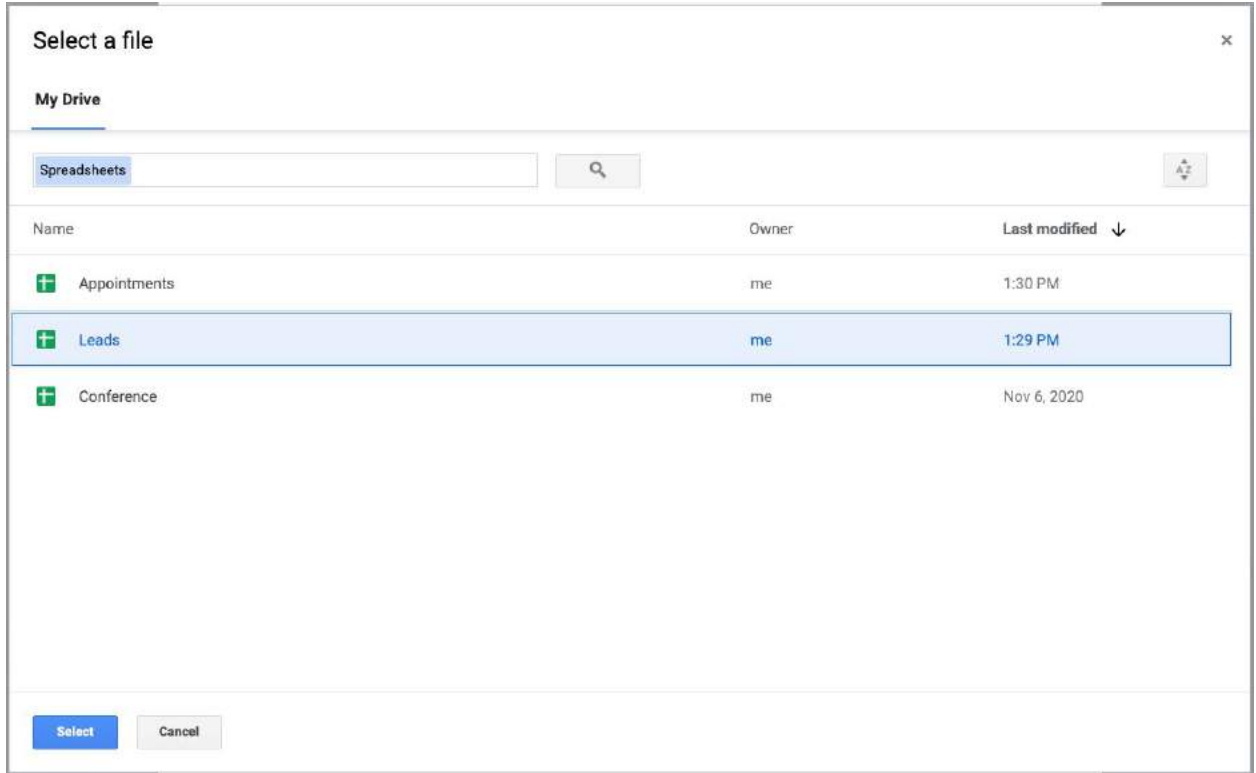
6. Select l**eads** as the table for your entity.
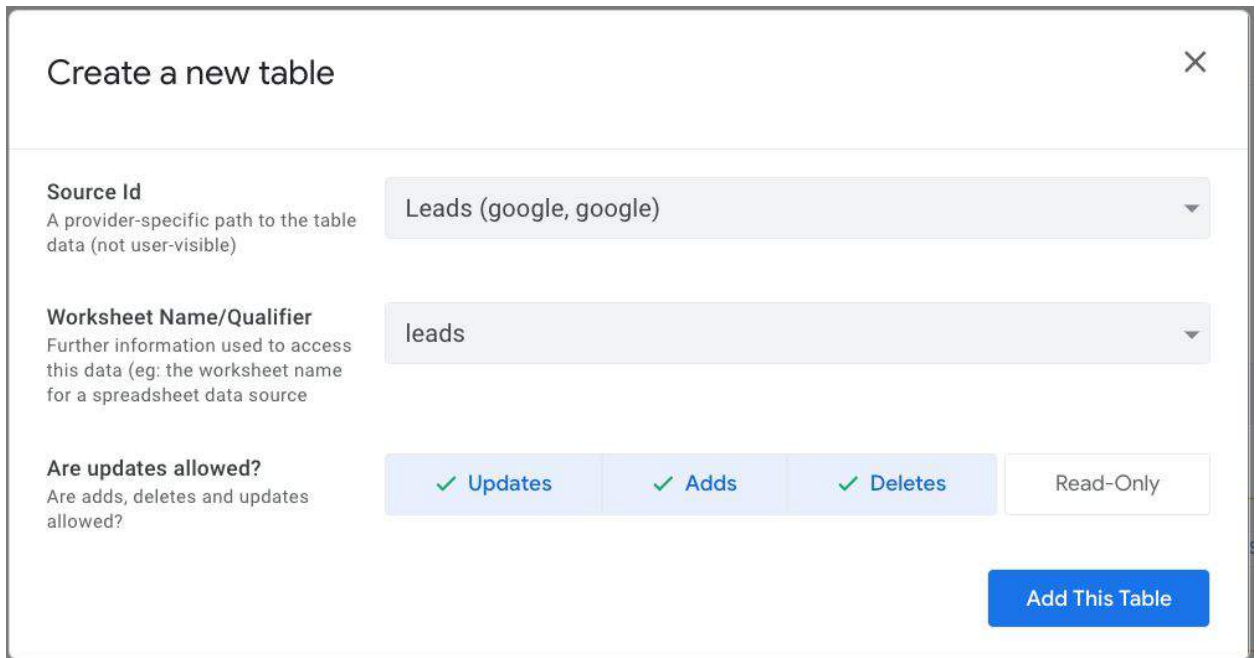
7.   Click **Add This Table**.

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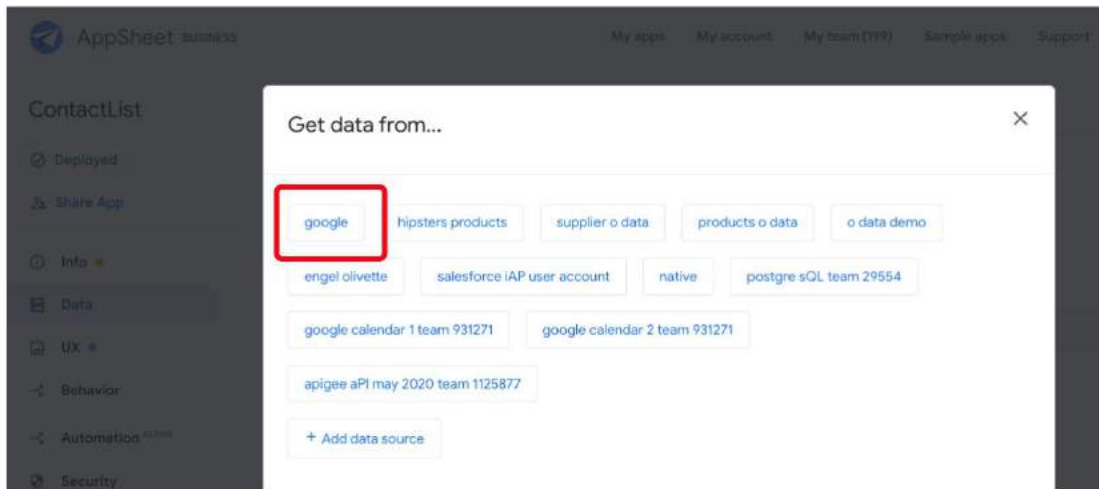
4. Ensure you create a new Google Sheet named "Appointments" (with the column names as below, the Sheet (tab) name should be "appointments") in the Google Drive accessible to your AppSheet account

5. Use the file browser to select the appropriate **Appointments** Sheet as the data source.
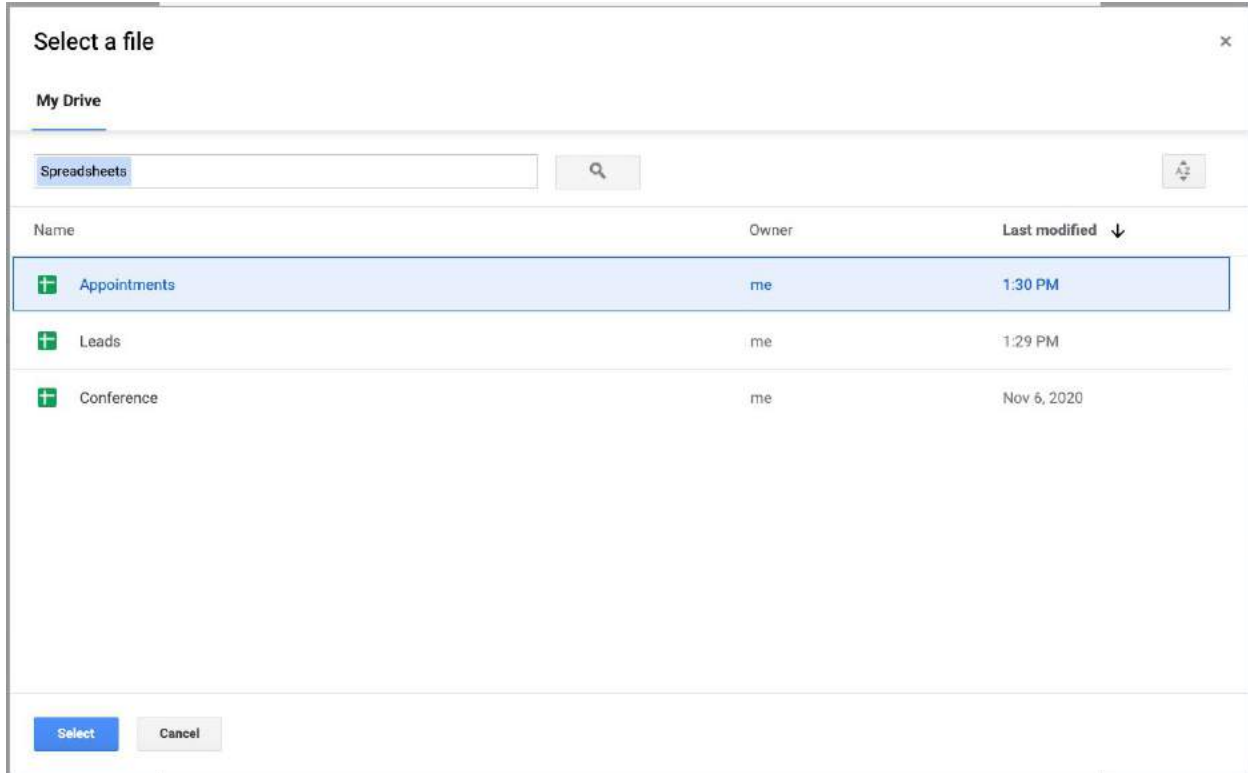
6. Select **appointments** as the table for your entity.
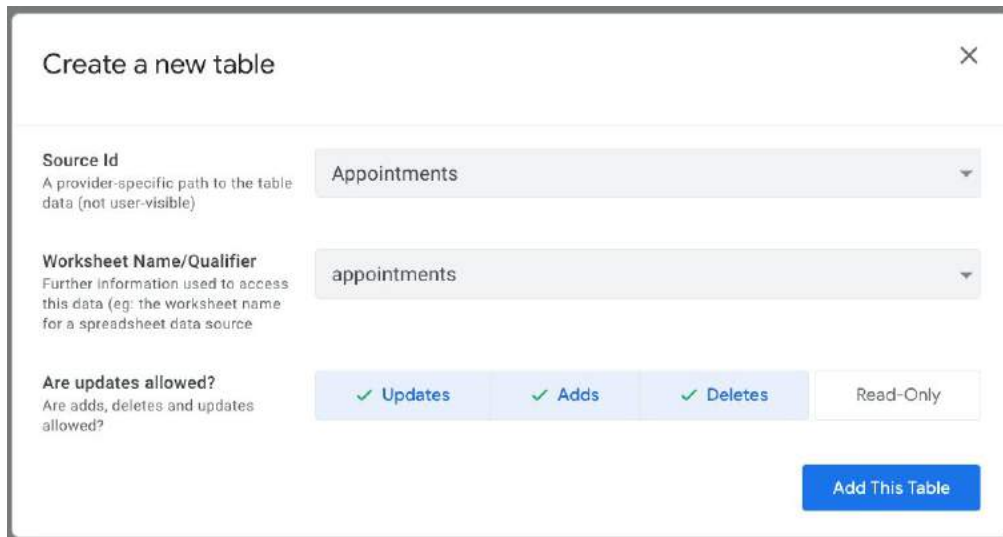
7. Make sure the **Confirmation Required** column type is **Text**.

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

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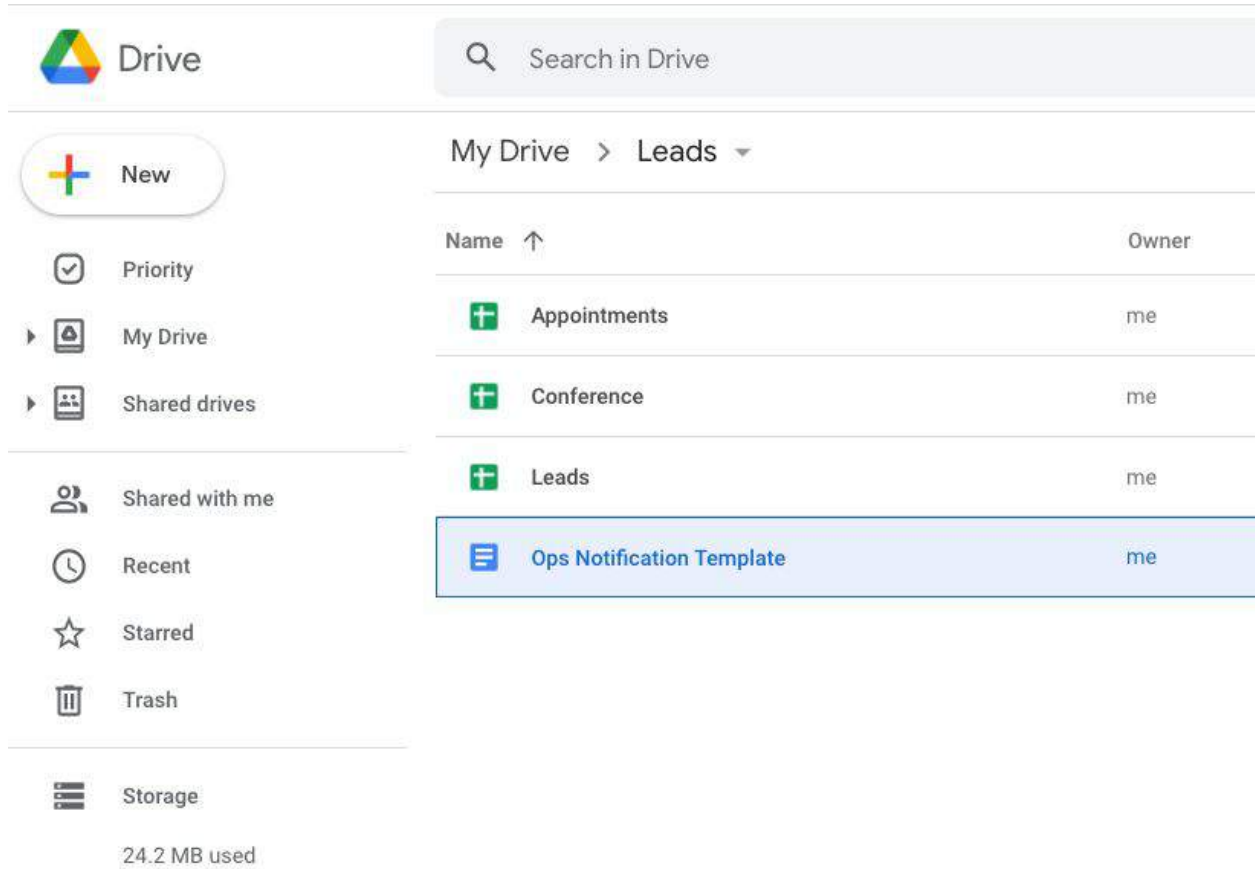
2. Enter the following text in the document, as shown in the images below:

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").
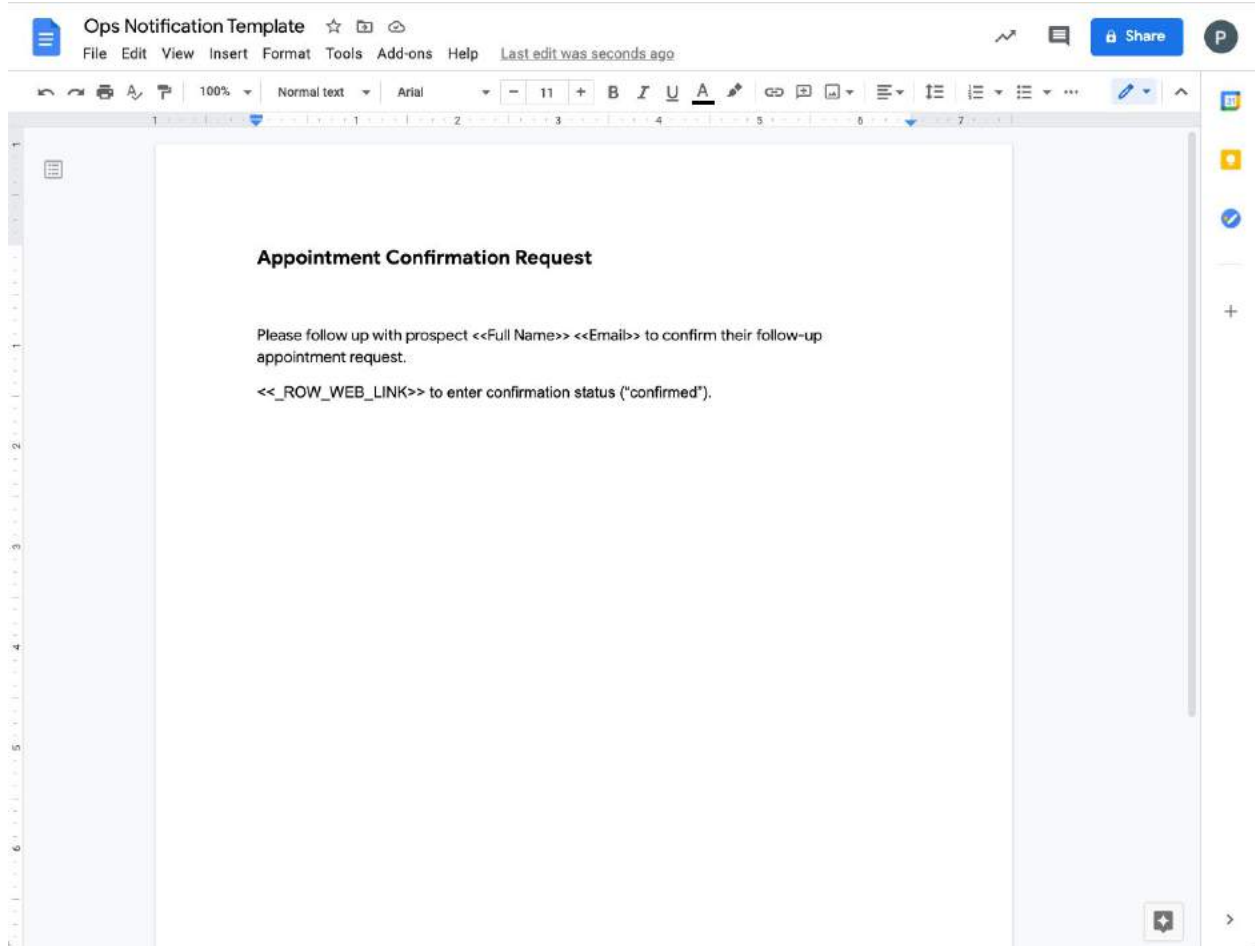
Document processing states and status codes

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).