Application Control Framework

Sample Data Enrichment

Demo Setup Guide

Version 1

<AUTHOR>

<DATE>

# Revision Summary

| **Date** | **Revision History** | **Comments** |
| --- | --- | --- |
| <REV\_DATE> | 1.0 | Initial Version |

# Table of Contents

[**Prerequisites 4**](#_kumxb0czmehl)

[**Disclaimer 4**](#_mxcfp5m5k19)

[**Key Native App Components 5**](#_b3znclu21j4n)

[Application Logic 5](#_vnb10gjzt43l)

[Multiple Native App Modes 5](#_4m4uif1pu9g8)

[Setup Script 6](#_cjktyahncaw4)

[Manifest File 6](#_5cmuedb2js68)

[Readme 6](#_dr4fq1ci0n8)

[Sidecar Loader 6](#_6sihrpxing11)

[**App Control Manager 7**](#_kbfd9dqgps02)

[**Build and Deploy Demo Native App 9**](#_fi8h6xkew0fn)

[Step 1: Build the Demo Native App 9](#_yu0cc7hy6b13)

[Part 1: Create the Application Package 9](#_jyaujrav3b9q)

[Part 2: Create/Patch a Version for the Application Package 11](#_sqjt74hzut6q)

[Part 3: Release Patch 16](#_jj1uq2sl77n8)

[Step 2: Create the Native App Listing 17](#_lpsmf44762d1)

[Step 3: Install the Native App in the Consumer Account 18](#_x0frofw5itdt)

[Step 4: Run the Scripts from the Native App’s Readme 18](#_aix0ivj82eej)

[**Additional Test Scenarios 24**](#_2qsfvuyzfasz)

[Reach the Five Requests Limit Then Increase the Limit 24](#_qv4289l7w0m2)

[Disable the Consumer 25](#_tzq1kmjm0ga0)

# 

# Prerequisites

* The Events and ACF accounts have been configured, per the Deployment Guide
* The ACF Demo scripts have been executed, per the Deployment Guide
* The user that will build and deploy the native app must be granted the P\_<APP\_CODE>\_ACF\_ADMIN role.
* For demo purposes, a demo consumer account should be created in the same organization as the provider’s account.

**NOTE:** <APP\_CODE> = an abbreviated representation for the name of the app (e.g. SDE for Sample Data Enrichment)

# Disclaimer

Any screenshots included in this guide are examples. Please refer to the text in the steps below when installing this app.

# 

# Key Native App Components

## 

## Application Logic

Application logic is the critical component of the native app. It provides the intended functionality the consumer executes in their Snowflake account. Application logic is deployed in the form of one or more stored procedures. Apps built by the ACF can support multiple functionality/procedures and allows the provider the ability to control which consumers get access to what functionality.

For this demo, the application logic resides in the ENRICH stored procedure. The instructions later in this document will build a native app that includes this stored procedure.

### Multiple Native App Modes

The framework comes with the ability to build a native app with custom functionality, depending on the version of the app. For example, the consumer can evaluate the “free” version of the app from the Snowflake Marketplace, without interaction from the provider. Once the consumer is interested in the one or more paid versions of the app, they can be granted access to the desired version.

By default, the ACF supports three app modes:

* **FREE**: a free version of the app that is publicly available in the Snowflake Marketplace. This version offers limited functionality, meant to entice the consumer to convert to a paid version of the app. Each consumer of this app version has the same entitlements/limits (i.e. five requests).
* **PAID**: a paid version of the app that is publicly available in the Snowflake Marketplace. This version offers more or complete app functionality. Each consumer of this app has the same entitlements/limits (if any) enforced (i.e. process 1MM records every 30 days).
* **ENTERPRISE**: a version of the app where unique entitlements/limits can be set for each consumer. The entitlements/limits are managed via the ACF’s App Control Manager. This is ideal for providers that want to create custom deals with consumers where the default entitlements/limits of the other app versions are not ideal for the consumer. Enterprise versions of the app should be listed privately and only made available to a single consumer.

For the purposes of this demo, the **FREE** version of the native app will be built.

## Setup Script

The setup script contains SQL statements, including application logic DDL, that are executed when the consumer installs or upgrades an application or when a provider installs or upgrades an application for testing. Every application must contain a setup script. The setup script defines the objects that are created when an application is installed or upgraded. For more information, visit <https://docs.snowflake.com/en/developer-guide/native-apps/creating-setup-script>.

The ACF includes a setup script template that is used to construct each app version/patch’s setup script. The ACF’s App Control Manager UI automatically generates the setup script, based on the selected table/view, functions, and procedures required for each version/patch.

## Manifest File

The Snowflake native app Framework requires that every application package contains a manifest file. This file defines properties required by the application package, including the location of the setup script and version definitions.

* The manifest file has the following requirements:
  + The name of the manifest file must be manifest.yml.
  + The manifest file must be uploaded to a named stage so that it is accessible when creating an application package or Snowflake native app.
  + The manifest file must exist at the root of the directory on the named stage.
  + For more information, visit <https://docs.snowflake.com/en/developer-guide/native-apps/creating-manifest>.

The ACF includes a manifest template that is used to construct each app version/patch’s setup script. The ACF’s App Control Manager UI automatically generates the manifest file.

## Readme

A readme file is included when the consumer installs the corresponding version. Each readme is slightly different due to setup steps required for each app version.

## 

## Sidecar Loader

Sidecar is a utility that allows the consumer to execute pre-set commands in their account that cannot be executed by the application during installation. The Sidecar loads commands into a table to be executed by the consumer, via the SidecarRunner stored procedure. The commands are visible and can be evaluated prior to execution.

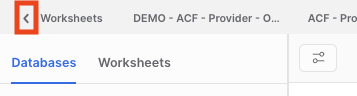
# App Control Manager

The Application Control Framework includes the App Control Manager, a Streamlit UI available in the provider account. The App Control Manager allows the provider to easily build and manage an app built on the ACF, manage consumers, and remove the ACF if/when desired.

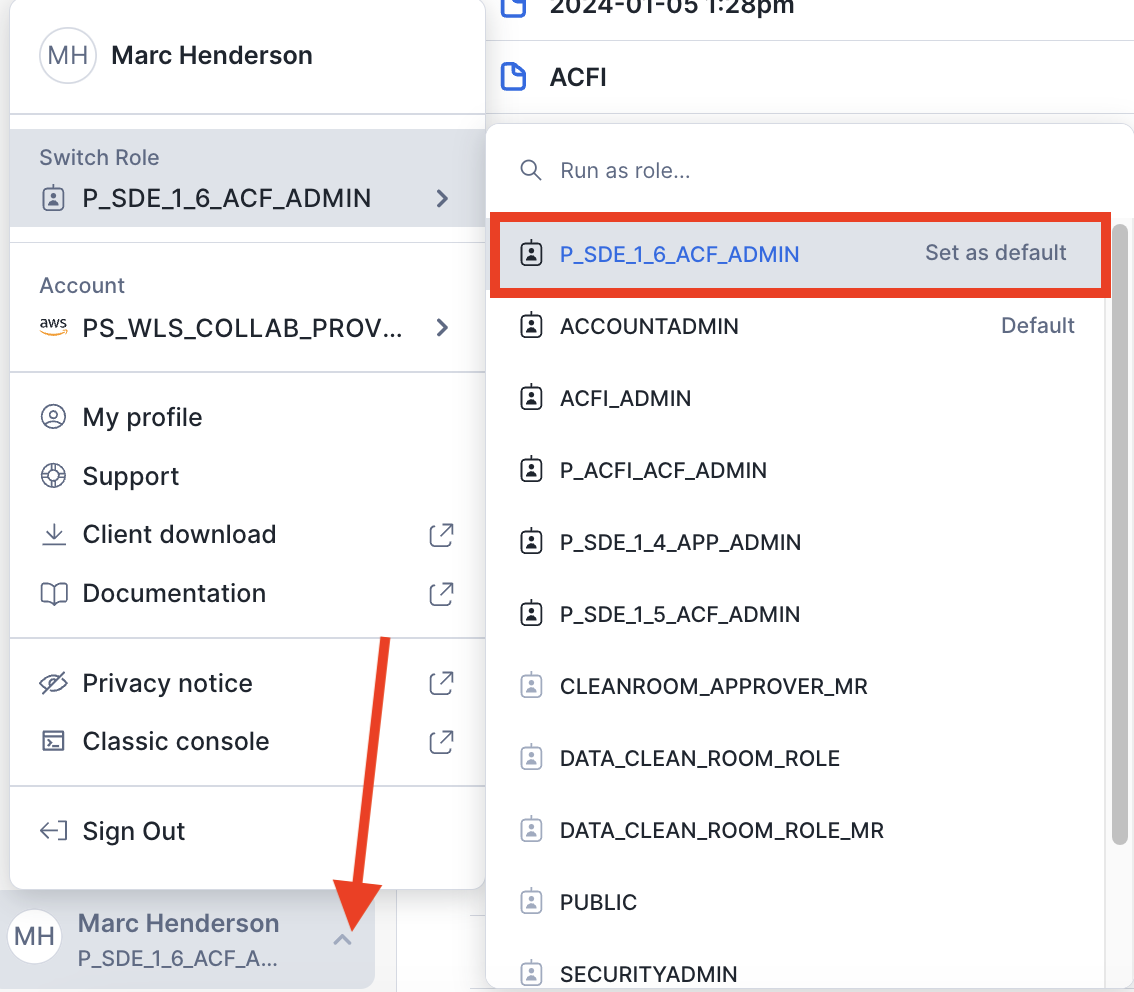
The App Control Manager can be accessed via the following steps:

**Step 1**: Log into **Snowsight**.

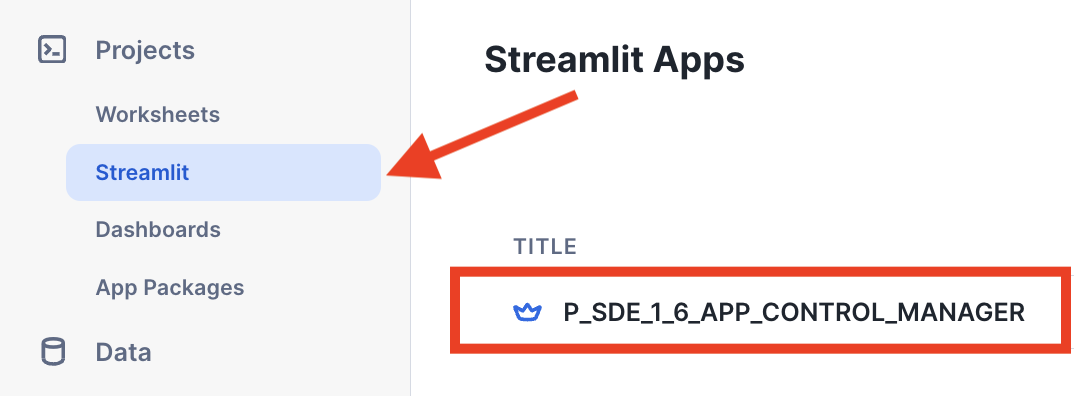
**Step 2**: Once logged in, if not at the Snowsight home screen, click the **Back** button, in the top left area of the UI, to open the left navigation menu.

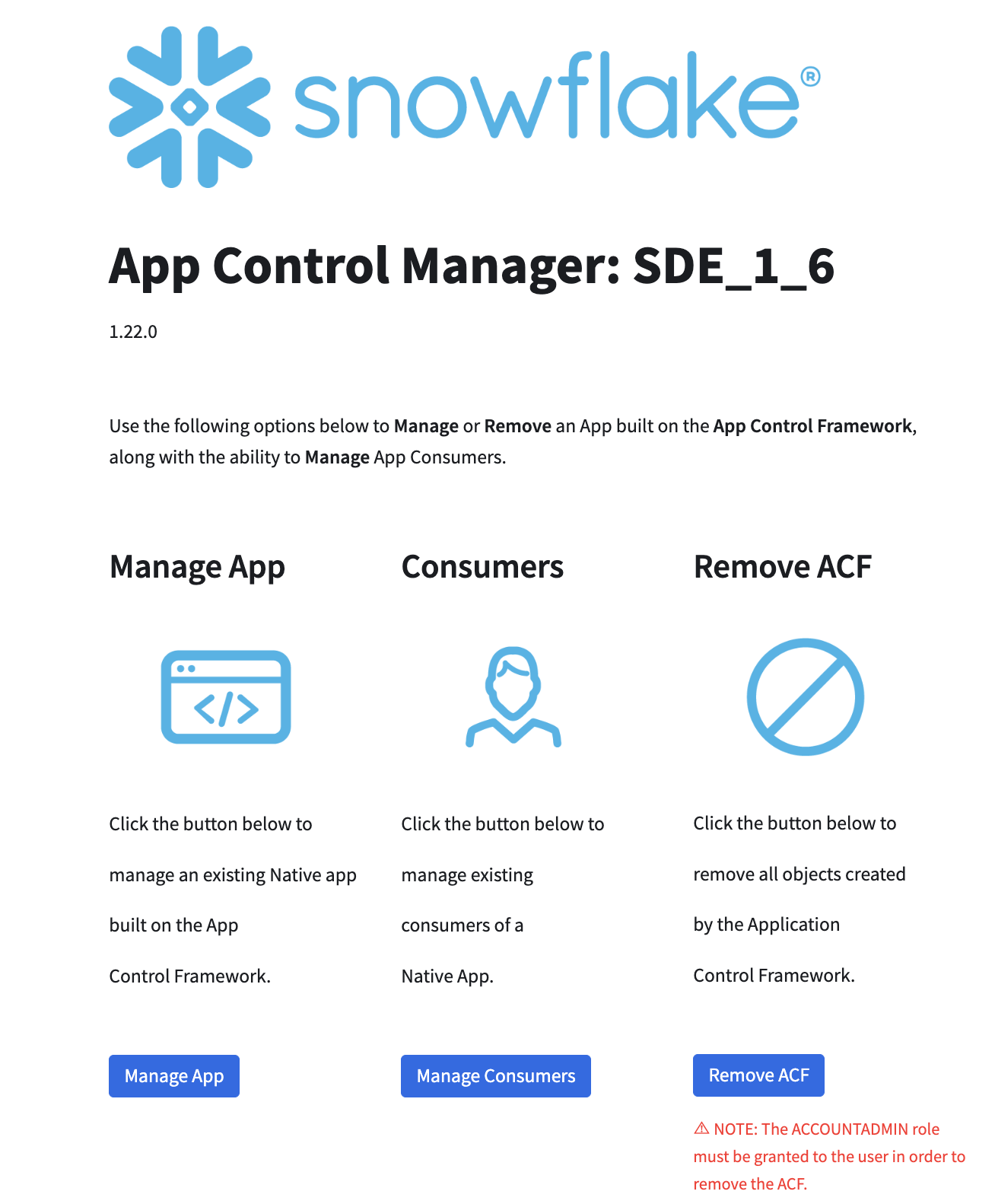


**Step 3**: Switch to the **P\_<APP\_CODE>\_ACF\_ADMIN** role, by clicking the drop-down in the bottom left area of the UI, then hovering over the Switch Role menu item.



**Step 4**: Click **Projects** >> **Streamlit**, then **P\_<APP\_CODE>\_APP\_CONTROL\_MANAGER**.





(App Control Manager Home)

# Build and Deploy Demo Native App

## Step 1: Build the Demo Native App

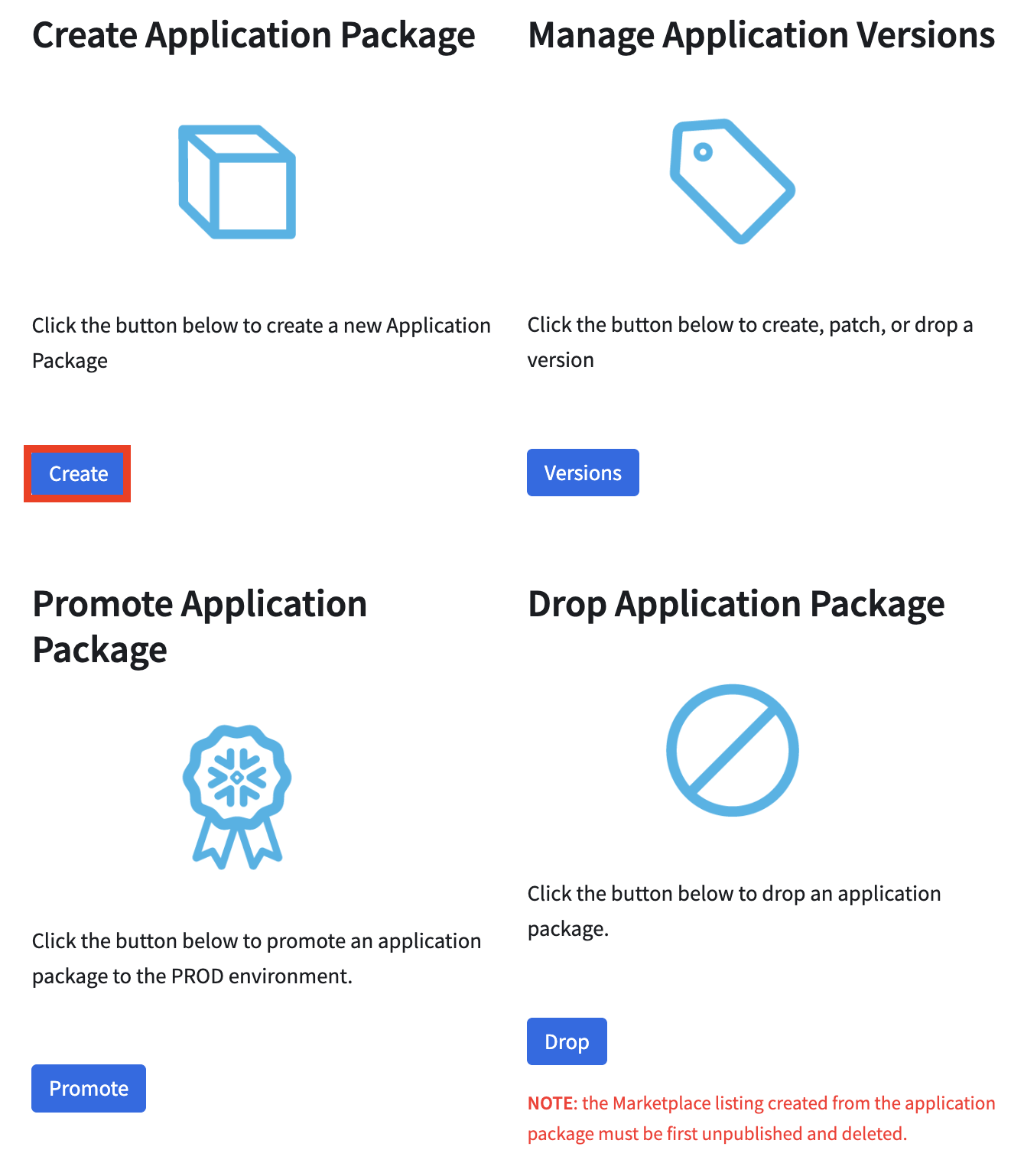
The following sections outline how to use the App Control Manager to build the demo native app, included with the ACF.

This demo app is a simple data enrichment app that enriches a consumer’s dataset with attributes from the provider’s dataset, when the records have the same email address.

### Part 1: Create the Application Package

The native app’s application package consists of views of the native app’s source data, along with views of the required Application Control Framework tables used to enforce the rules defined in the ACF and collect logs/metrics. In this demo, an application package will be created:

**Step 1**: In the App Control Manager, click **Manage App** >> **App Package** >> **Create**.

****

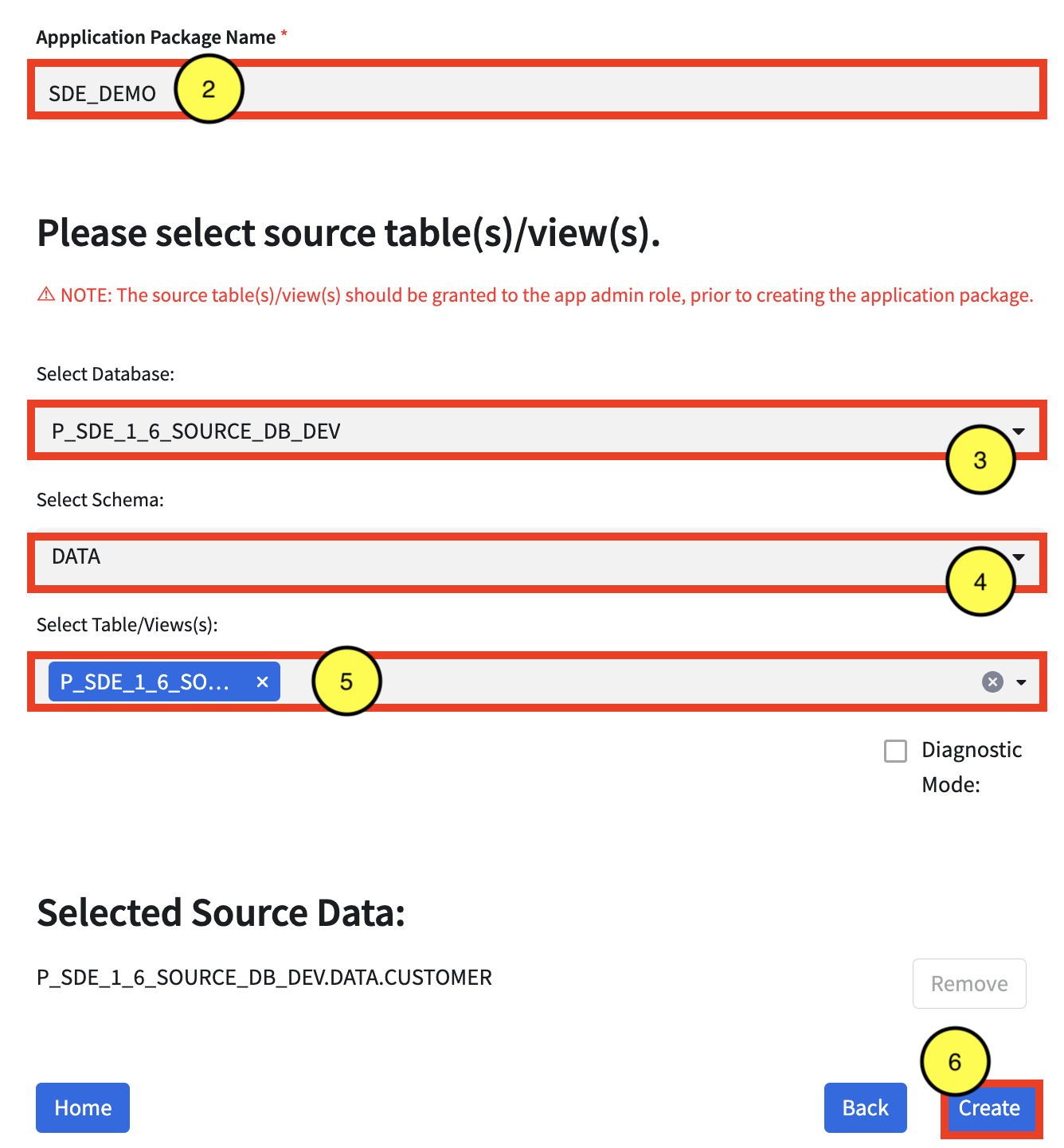
**Step 2**: Enter **SDE\_DEMO** in the Application Package Name text field. This name will be prefixed by: ***P\_<APP\_CODE>\_APP\_PKG*\_**.

**Step 3**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV** database from the Select Database drop-down.

**Step 4:** Select the **DATA** schema from the Select Schema drop-down

**Step 5**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV.DATA.CUSTOMER** table from the Select Table(s)/View(s) drop-down.

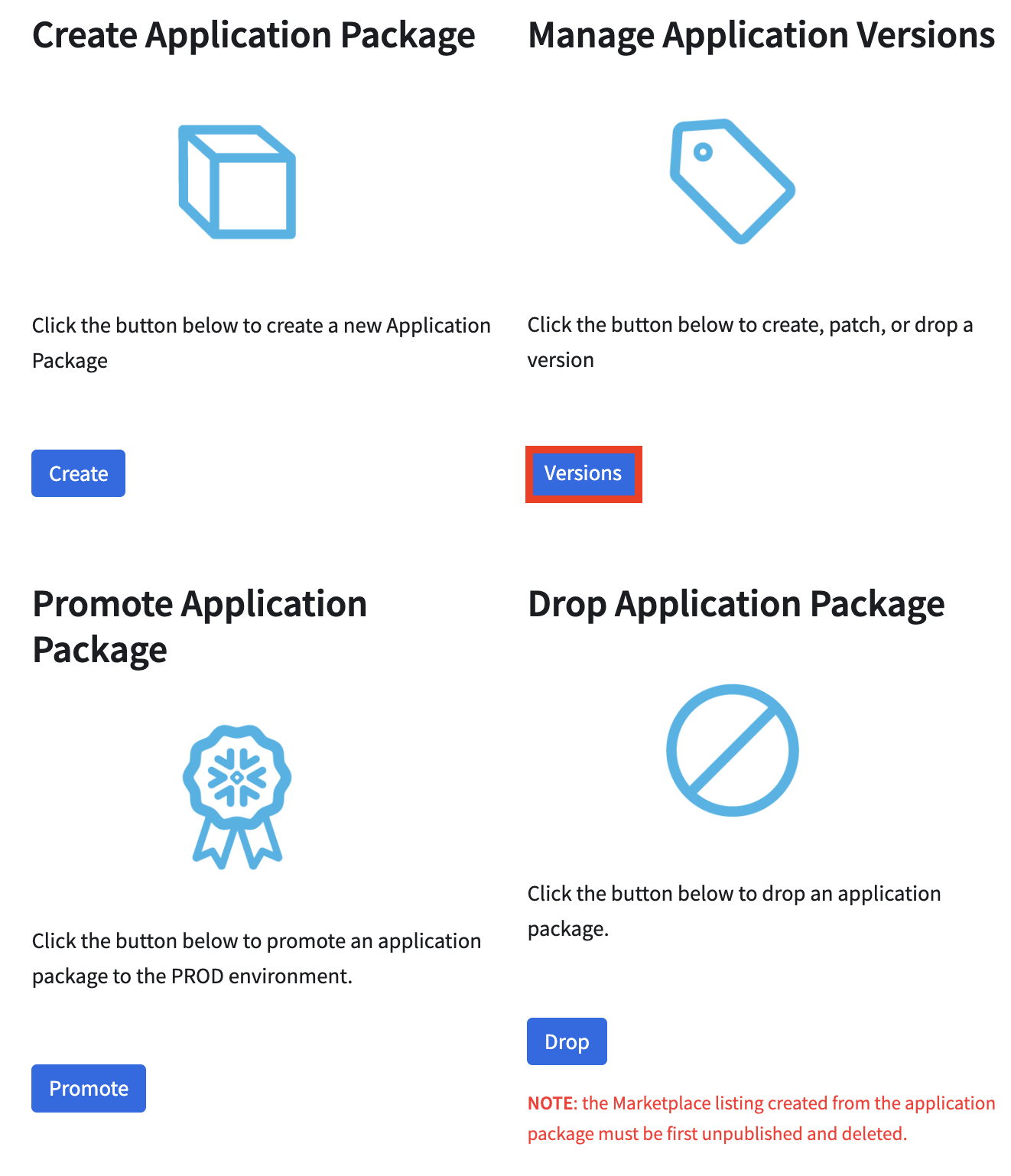
**Step 6**: Click **Create**.



### Part 2: Create/Patch a Version for the Application Package

Once the application package is created, the application functions/procedures are tied to the application package by creating a version and a patch. In this demo, the App Control Manager will be used to create a new version and patch:

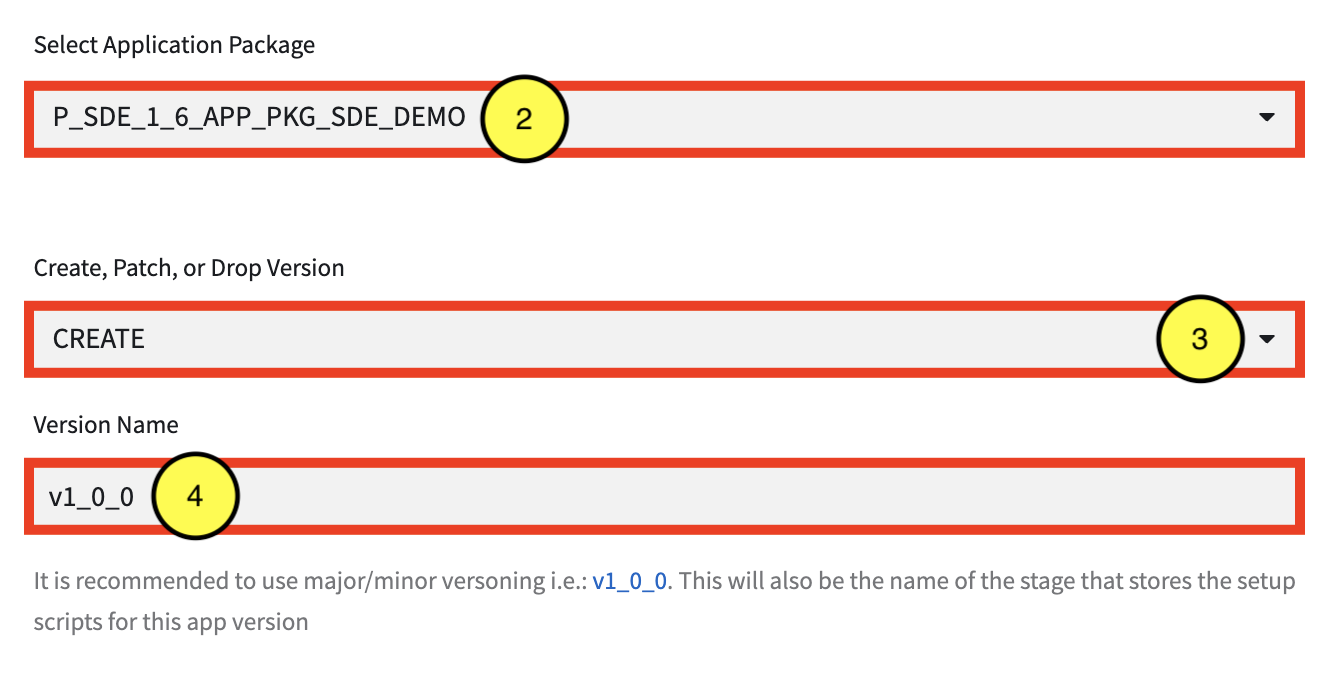
**Step 1**: In the App Control Manager, click **Manage App** >> **App Package** >> **Versions**.



**Step 2**: Select **P\_<APP\_CODE>\_APP\_PKG\_SDE\_DEMO** from the Select Application Package drop-down.

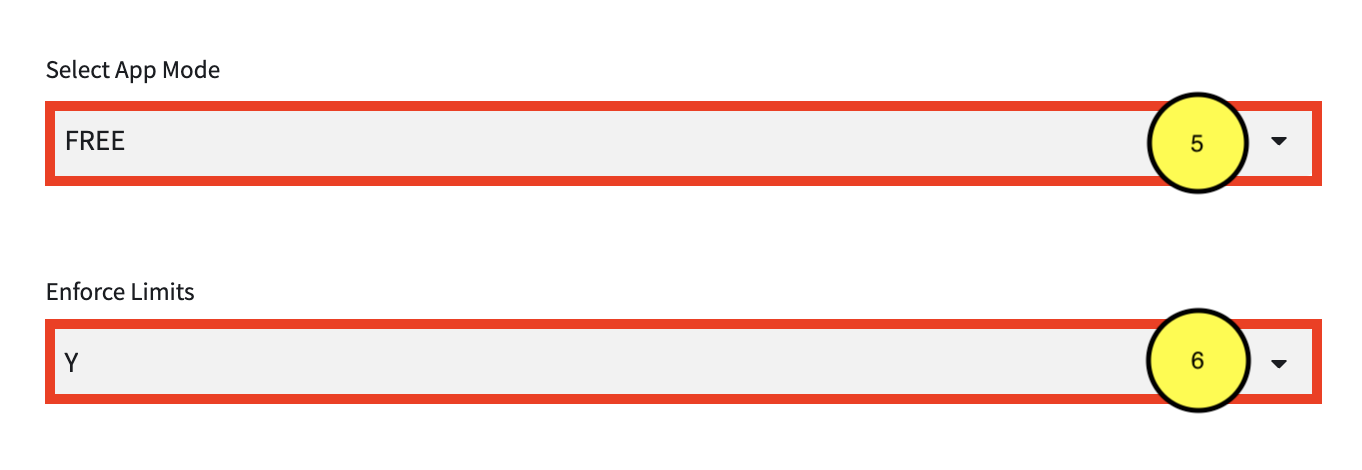
**Step 3**: Select **CREATE** from the Create, Patch, or Drop Version drop-down.

**Step 4**: Enter a **v1\_0\_0** in the Version Name field.

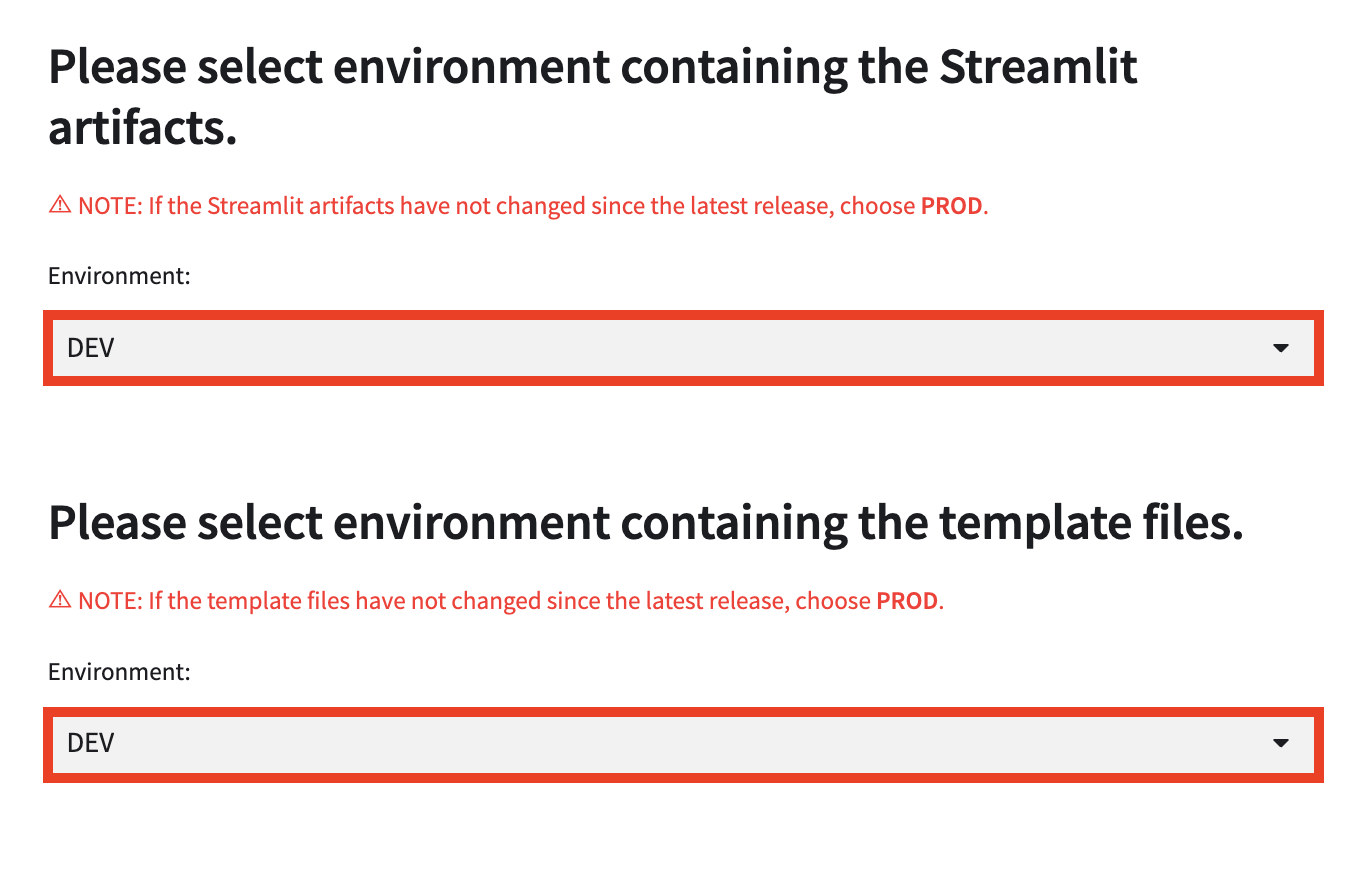


**Step 5**: For this demo, select **FREE** from the Select App Mode drop-down

**Step 6**: Select **Y** or **N** from the Enforce Limits drop-down. By default, limits should be enforced, but this can be set to N if limit enforcement should be turned off (i.e. during testing).

****

**Step 6**: Select DEV from the Environment drop-downs to pull the **Streamlit** and **template** files from the dev environment.

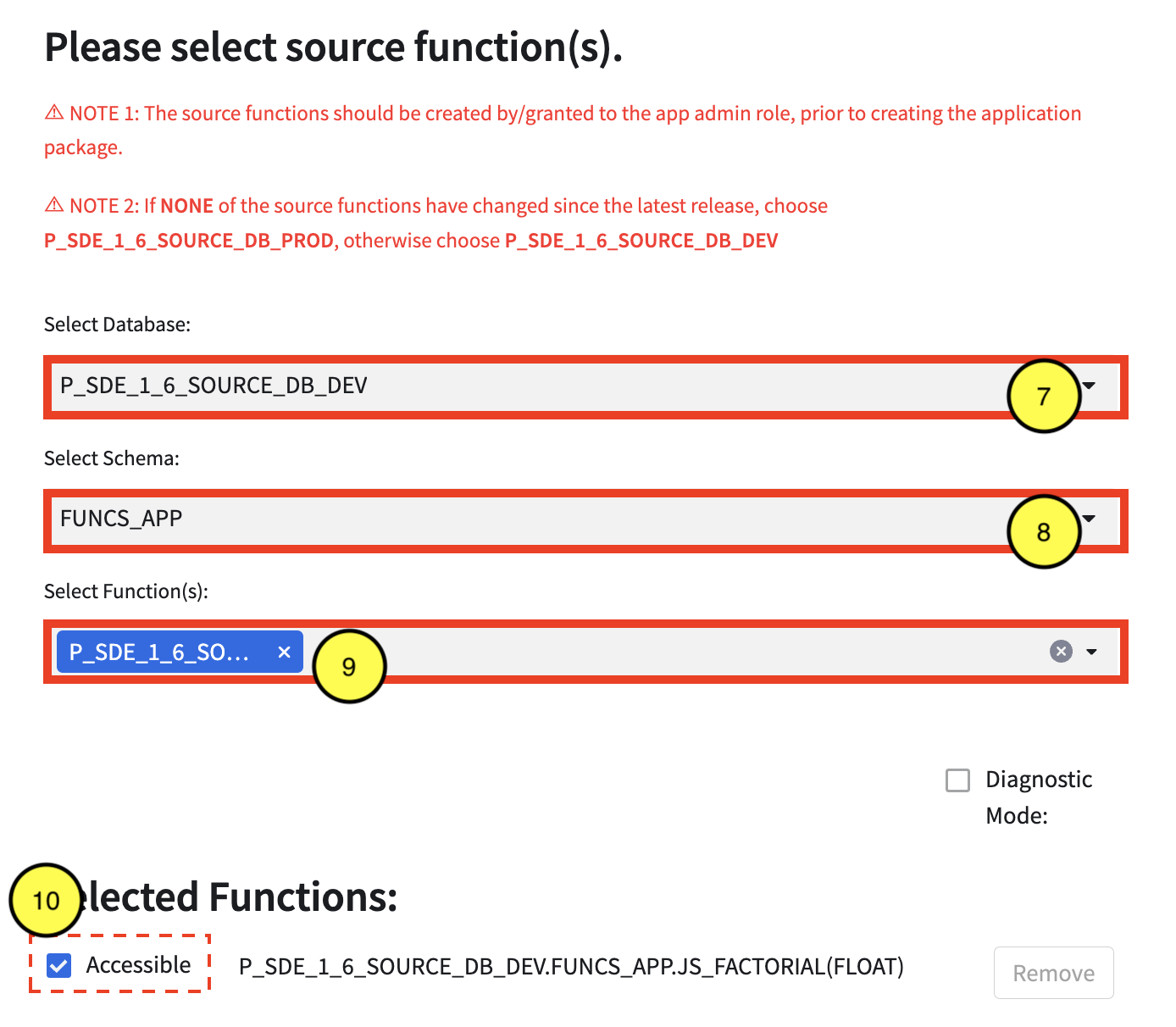


**Step 7**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV** database from the Select Database drop-down.

**Step 8**: Select the **FUNCS\_APP** schema from the Select Schema drop-down.

**Step 9**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV.FUNCS\_APP.JS\_FACTORIAL(FLOAT)** function from the Select Function(s) drop-down.

**Step 10**: Select the **Accessible** checkbox next to the function under Selected Functions to make the function accessible to the consumer.

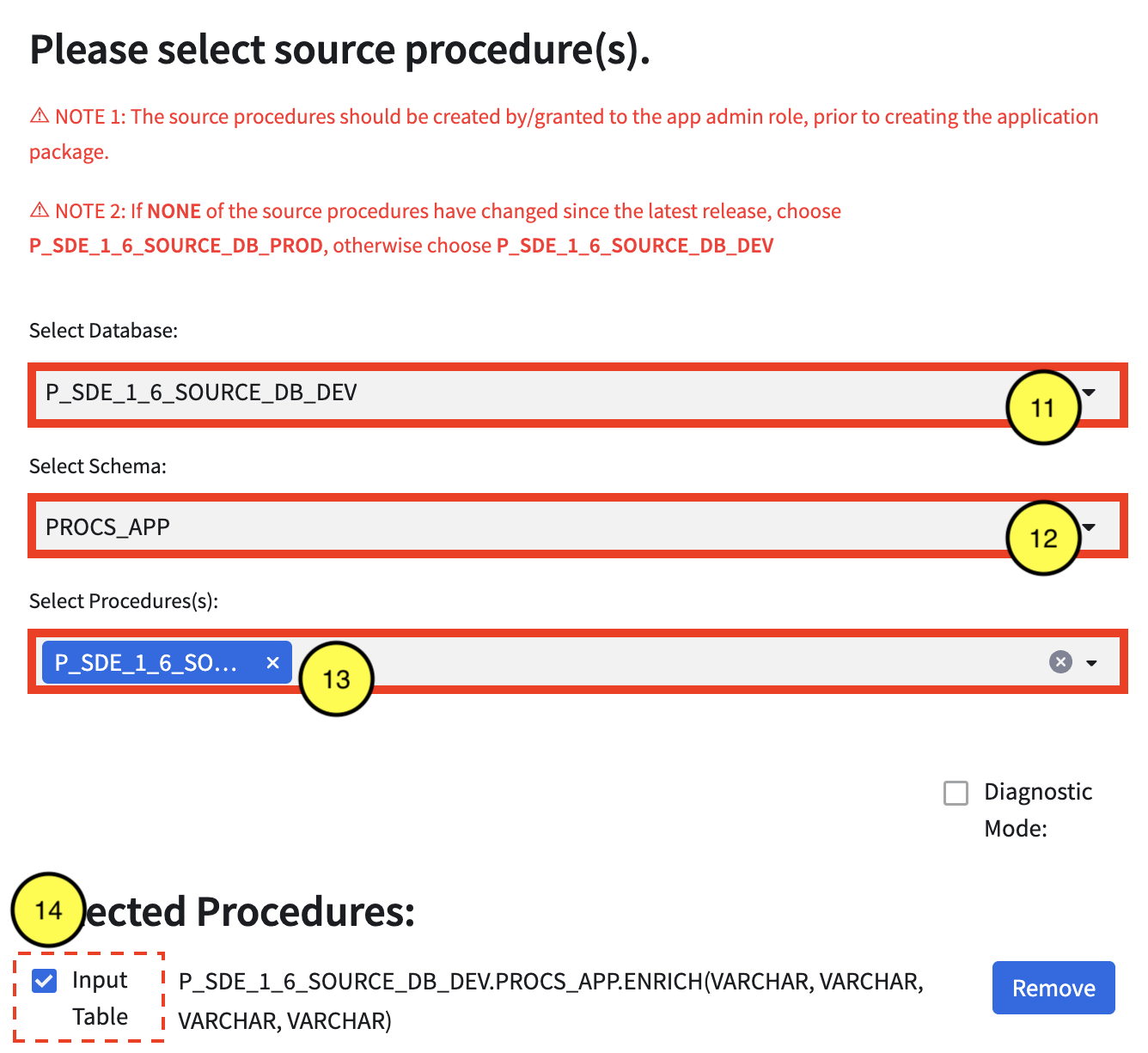


**Step 11**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV** database from the Select Database drop-down.

**Step 12**: Select the **PROCS\_APP** schema from the Select Schema drop-down.

**Step 13**: Select the **P\_<APP\_CODE>\_SOURCE\_DB\_DEV.PROCS\_APP.ENRICH(VARCHAR,VARCHAR,VARCHAR,VARCHAR)** procedure from the Select Procedure(s) drop-down.

**Step 14**: Select the **Input Table** checkbox next to the procedure under Selected Procedures.

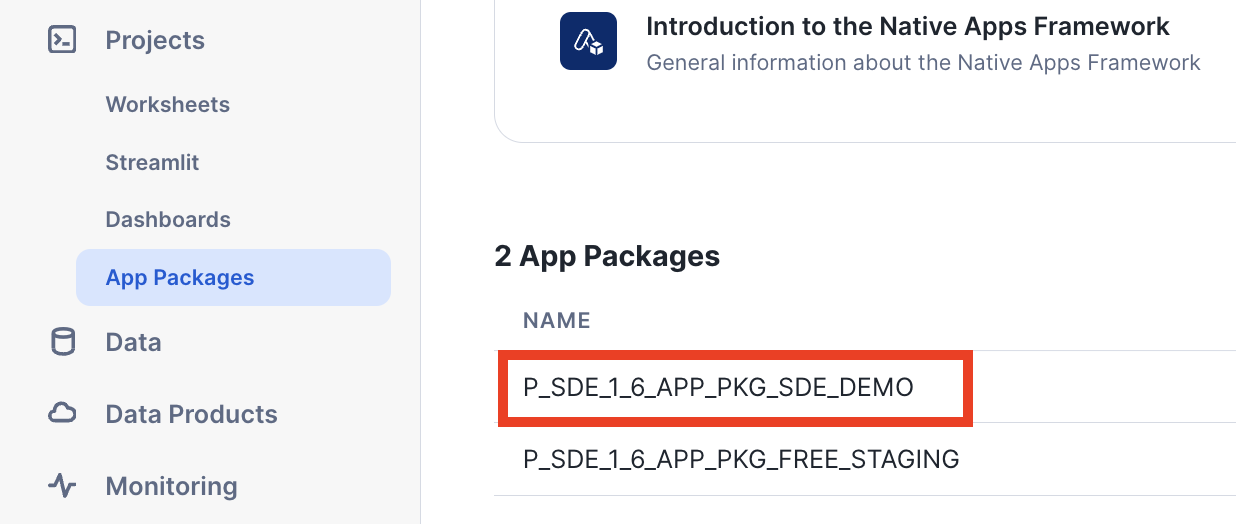


**Step 9:** Click the **CREATE** button.

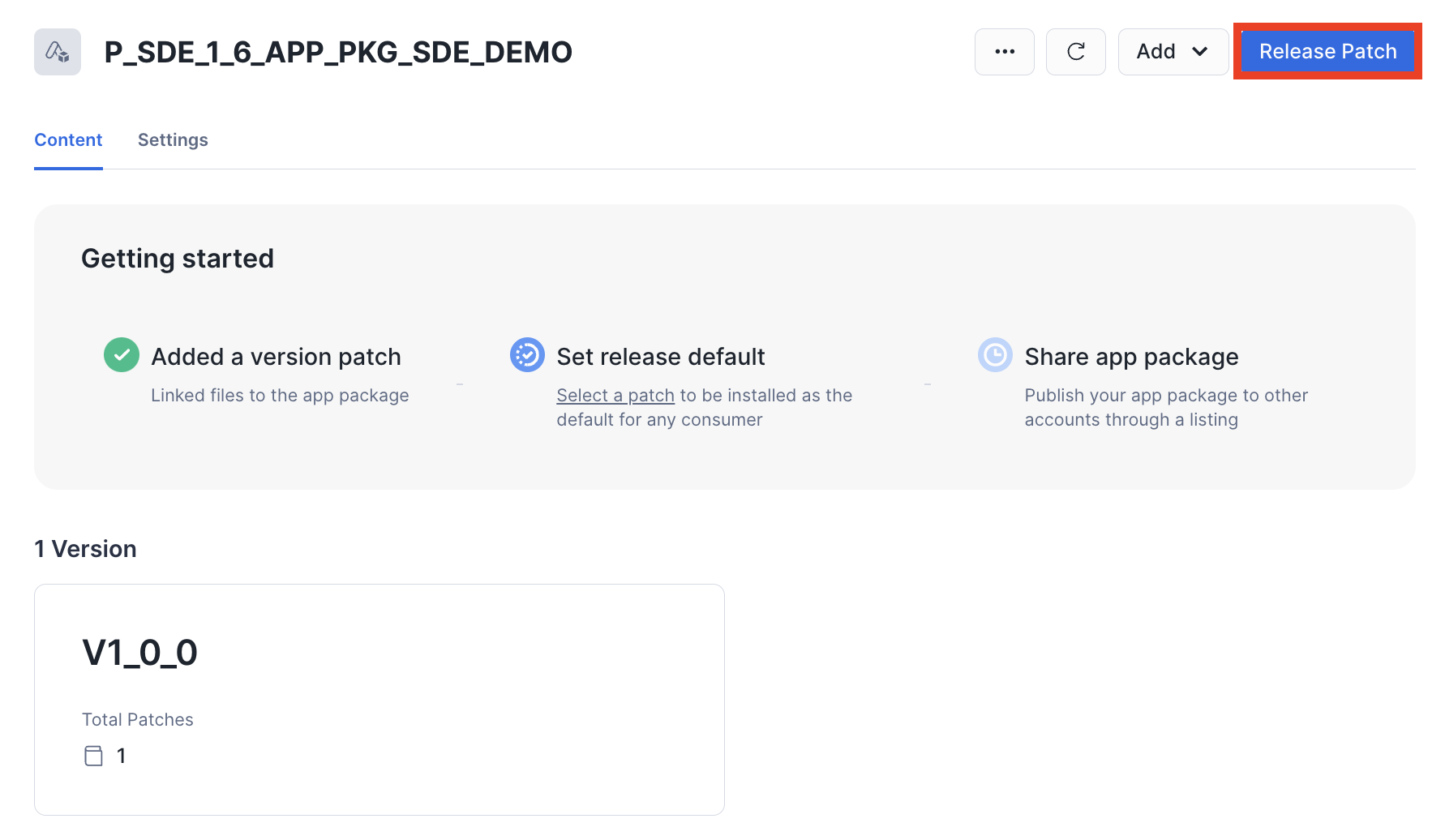
### Part 3: Release Patch

In order to grant consumers access to the latest version/patch, the provider must create a release for the application package. The following steps detail how to release an application package’s version/patch:

**Step 1**: In Snowsight, click **Projects** >> **App Packages**. Select the **P\_<APP\_CODE>\_APP\_PKG\_SDE\_DEMO** application package created in [Part 2](#_sqjt74hzut6q).



**Step 2**: Click **Release Patch**.

****

**Step 3**: Select **V1\_0\_0 patch 0** from the Patch to release drop-down.

**Step 4**: Choose **Set this patch to default**.

**Step 5**: Click **Save**.



## 

## Step 2: Create the Native App Listing

Once the application package has a released version and patch, the native app is ready to be privately listed. For instructions on how to create a Private Listing for the native app and add a demo consumer account, visit the **Create a Listing for Your Application** section: <https://docs.snowflake.com/en/developer-guide/native-apps/tutorials/getting-started-tutorial#publish-and-install-your-application>.

## 

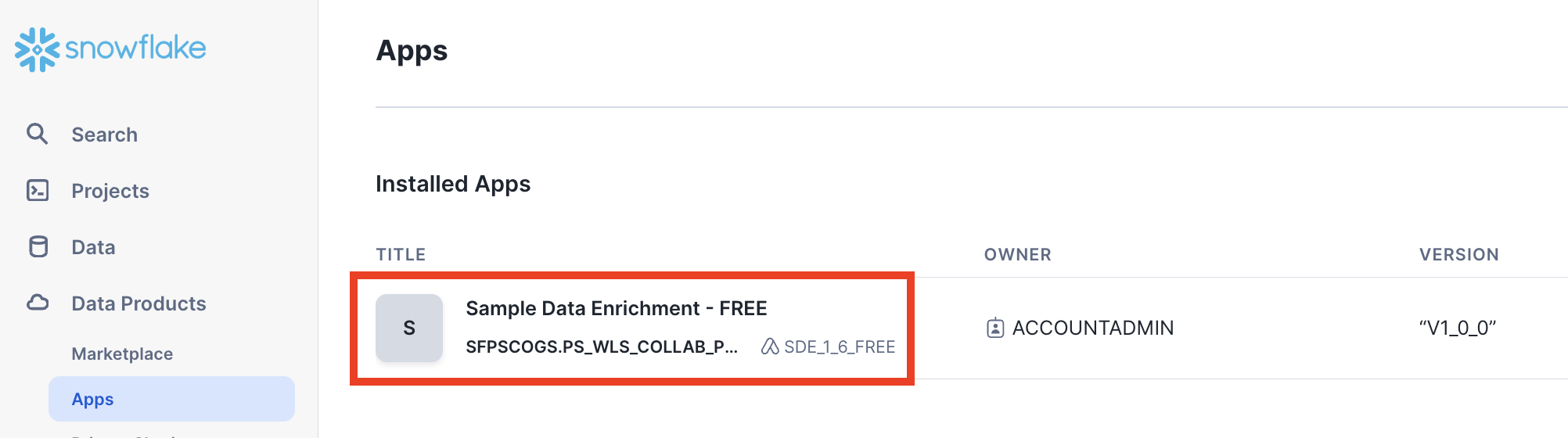
## Step 3: Install the Native App in the Consumer Account

In the demo consumer account added to the listing created in Step 2, install the native app available from the private listing. For instructions on how to install a native app, visit the **Install the Application** section <https://docs.snowflake.com/en/developer-guide/native-apps/tutorials/getting-started-tutorial#id6>.

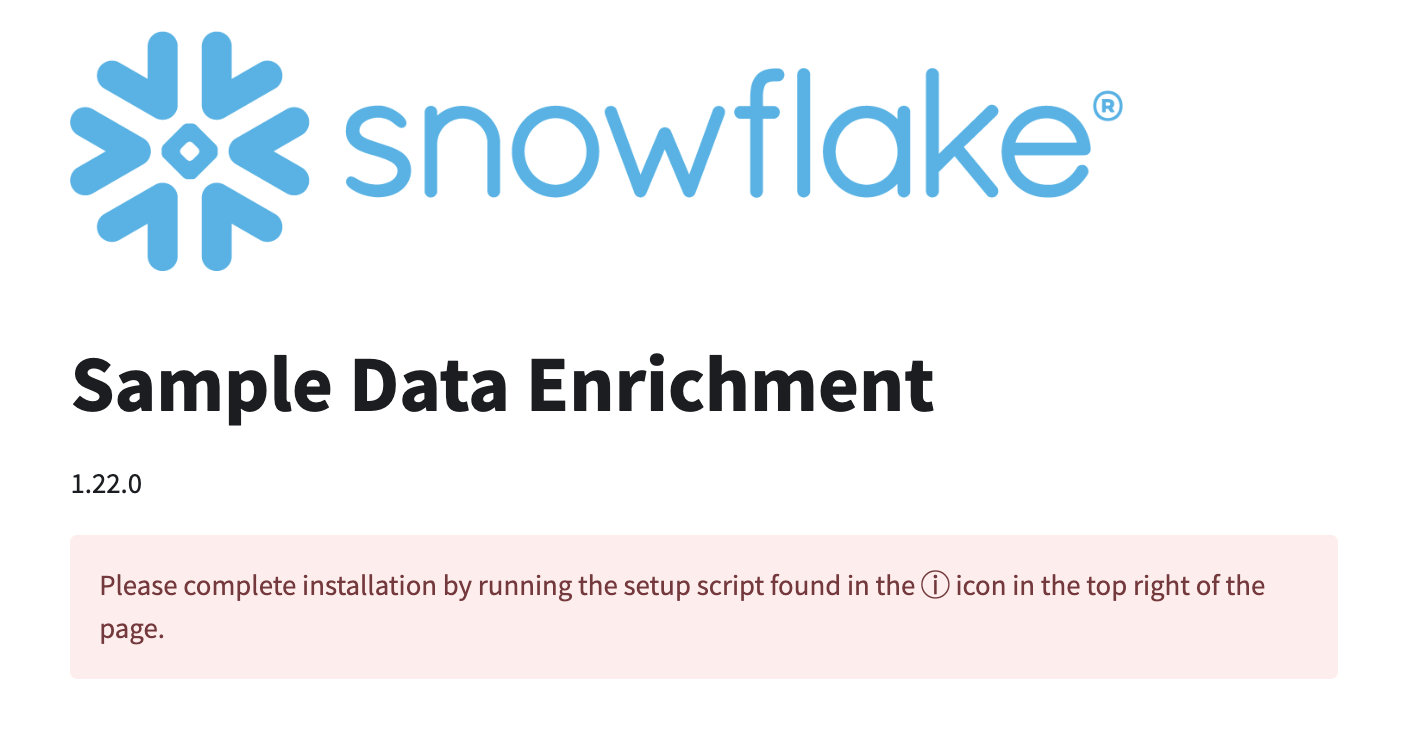
## Step 4: Run the Scripts from the Native App’s Readme

With each installation, the consumer is barred from using the native app until the consumer has executed a series of commands, including sharing events with the provider. These commands are located in the native app’s Readme.

**Step 1**: In Snowsight, click **Data Products** >> **Apps**, under the Installed Apps section, click the native app installed in the previous section.



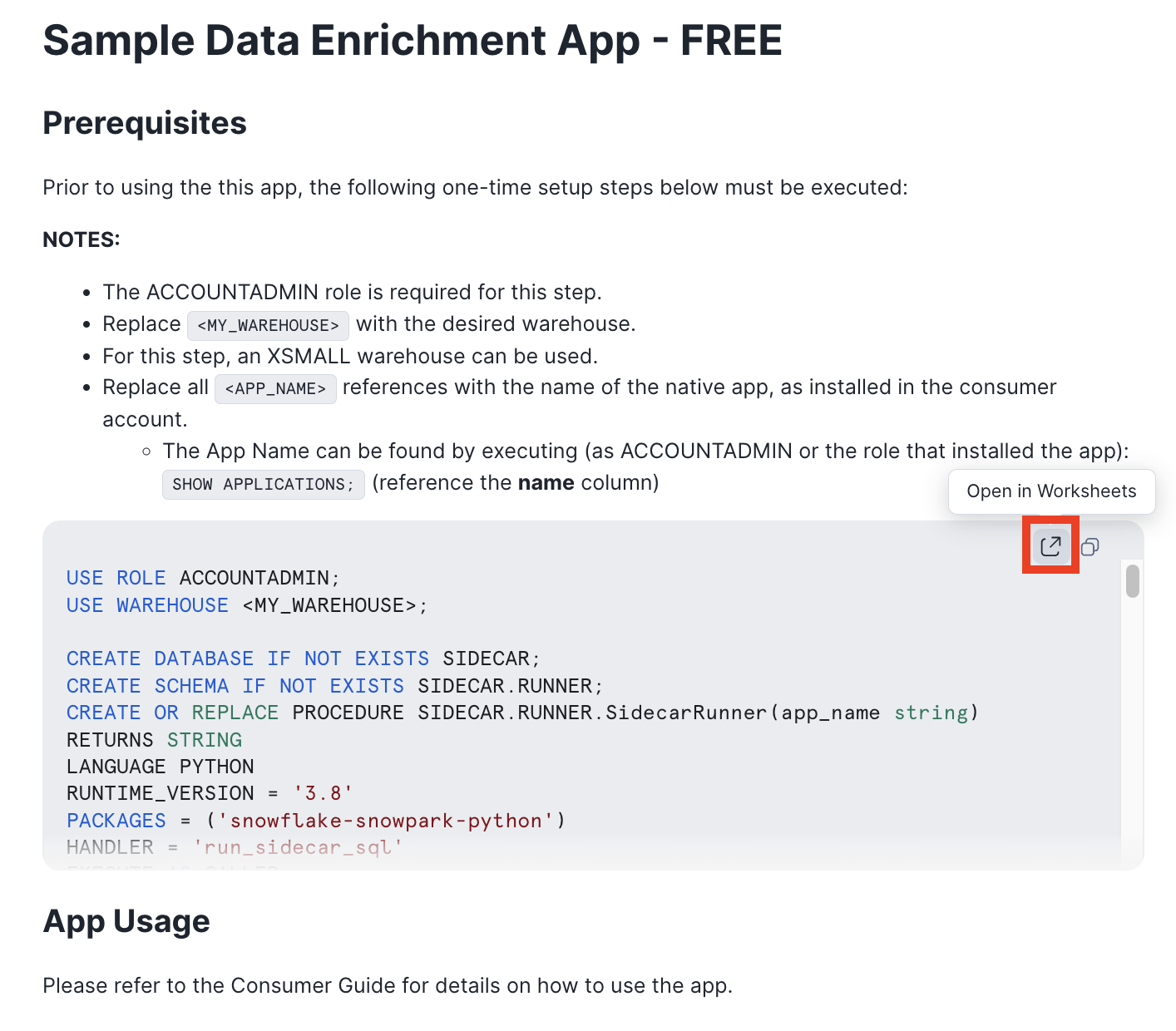
**Step 2**: Once loaded, the Streamlit UI will display the following error message:

****

**Step 3**: Click the **ⓘ** icon in the top right of the page.

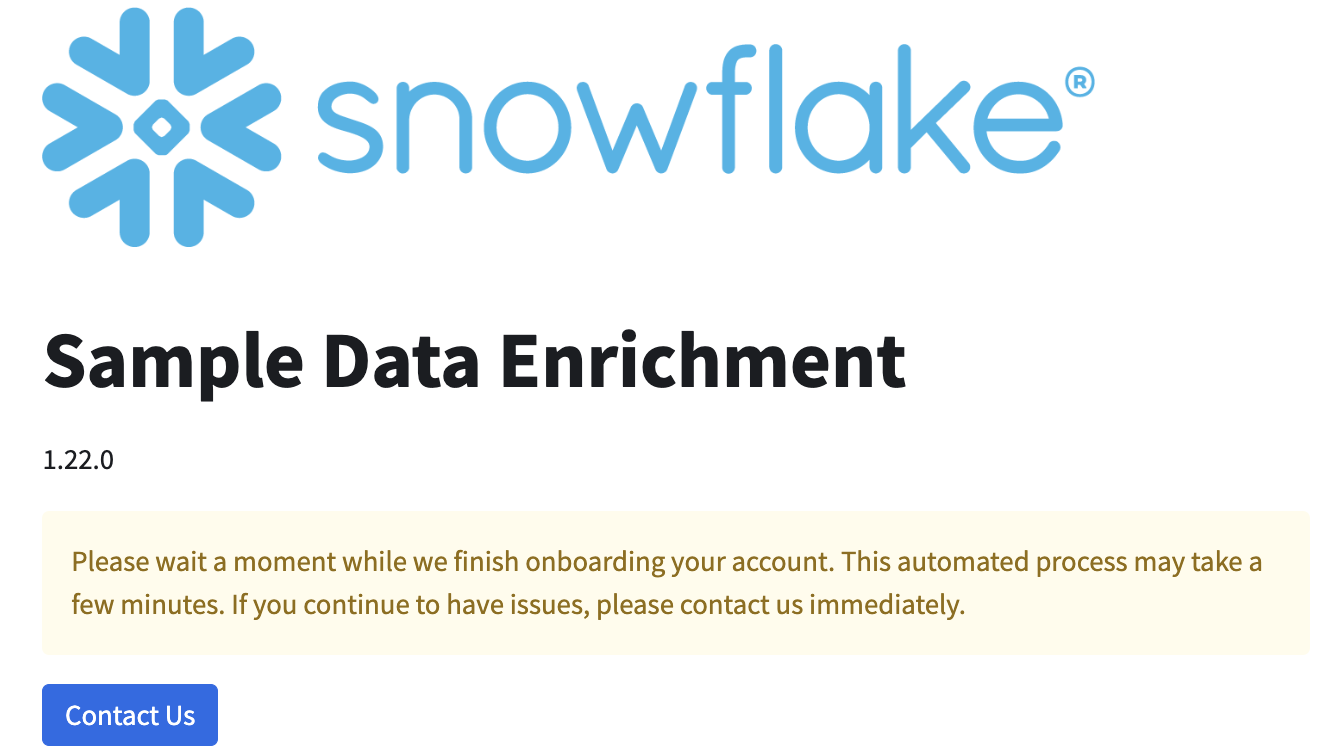


**Step 4**: Make note of the NOTES section of the Readme. Hover over and click the **Open in Worksheets** button.



**Step 5**: In the new worksheet, replace the macros specified in the Readme with the appropriate values. Once the macros are replaced, **run all commands**.

**Step 6**: Open the native app. The Streamlit UI will display the warning message below. It will take a few minutes for the native app to be enabled.

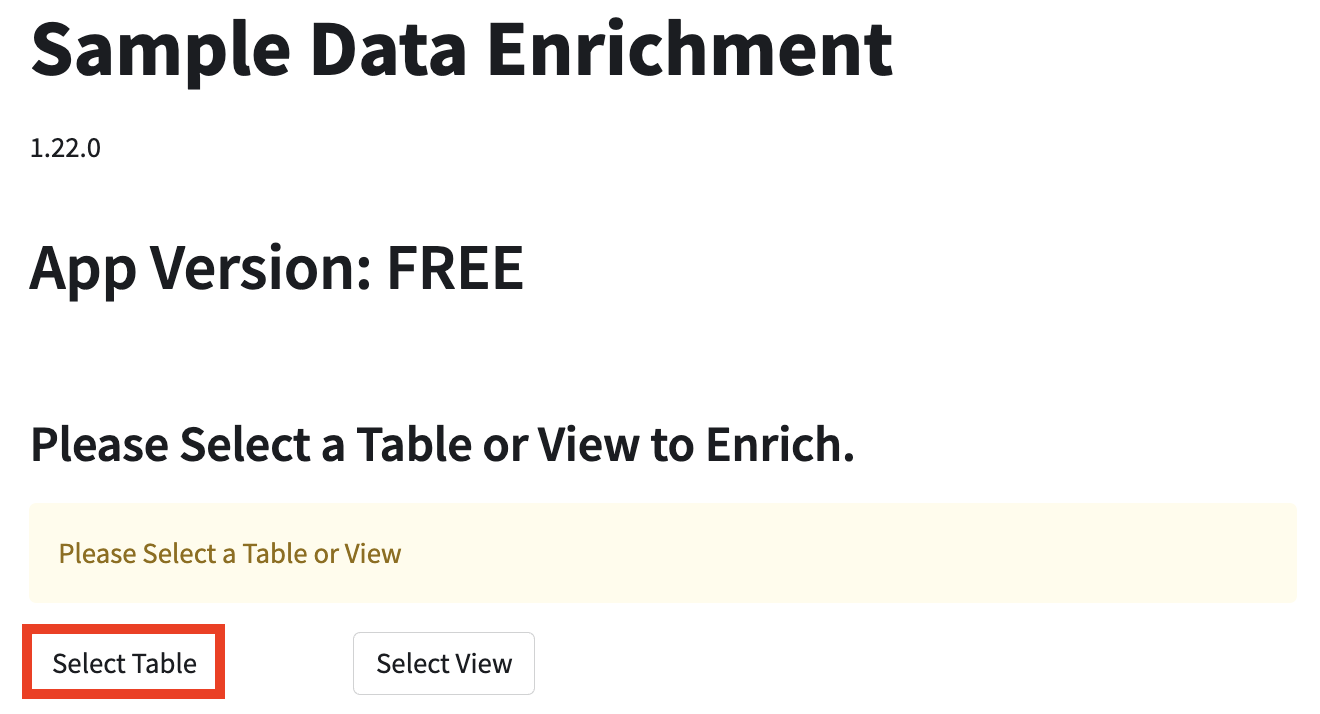


Step 5: App Usage

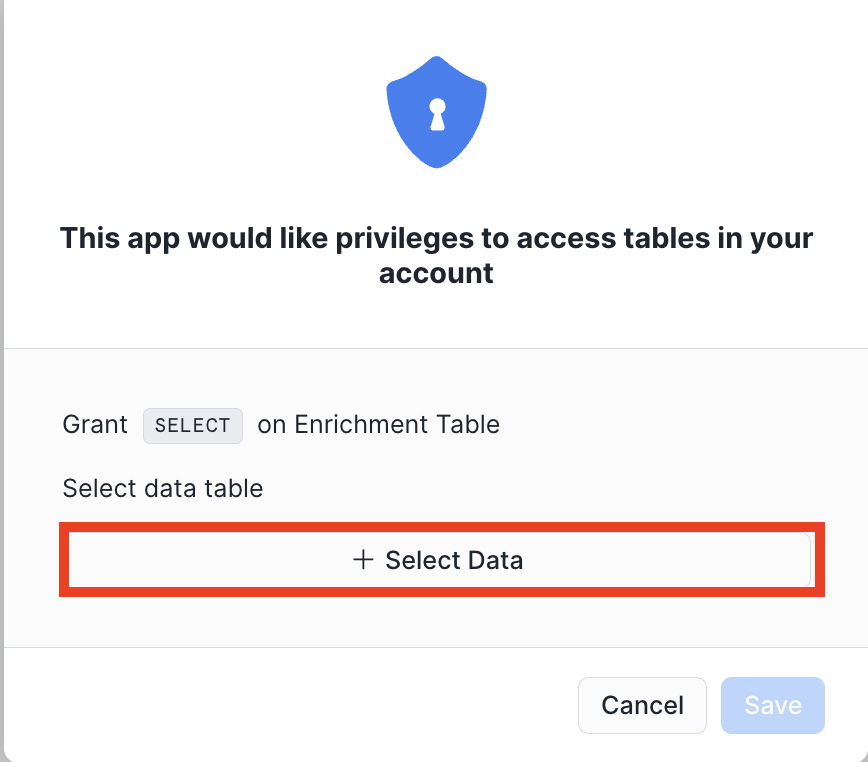
Once the native app has been enabled, the consumer can now use the app to enrich data. Sample data was created during executing the Readme commands. The following steps detail how to use the app to enrich data:

**Step 1**: Open the app. Once the app is enabled, the warning message is removed.

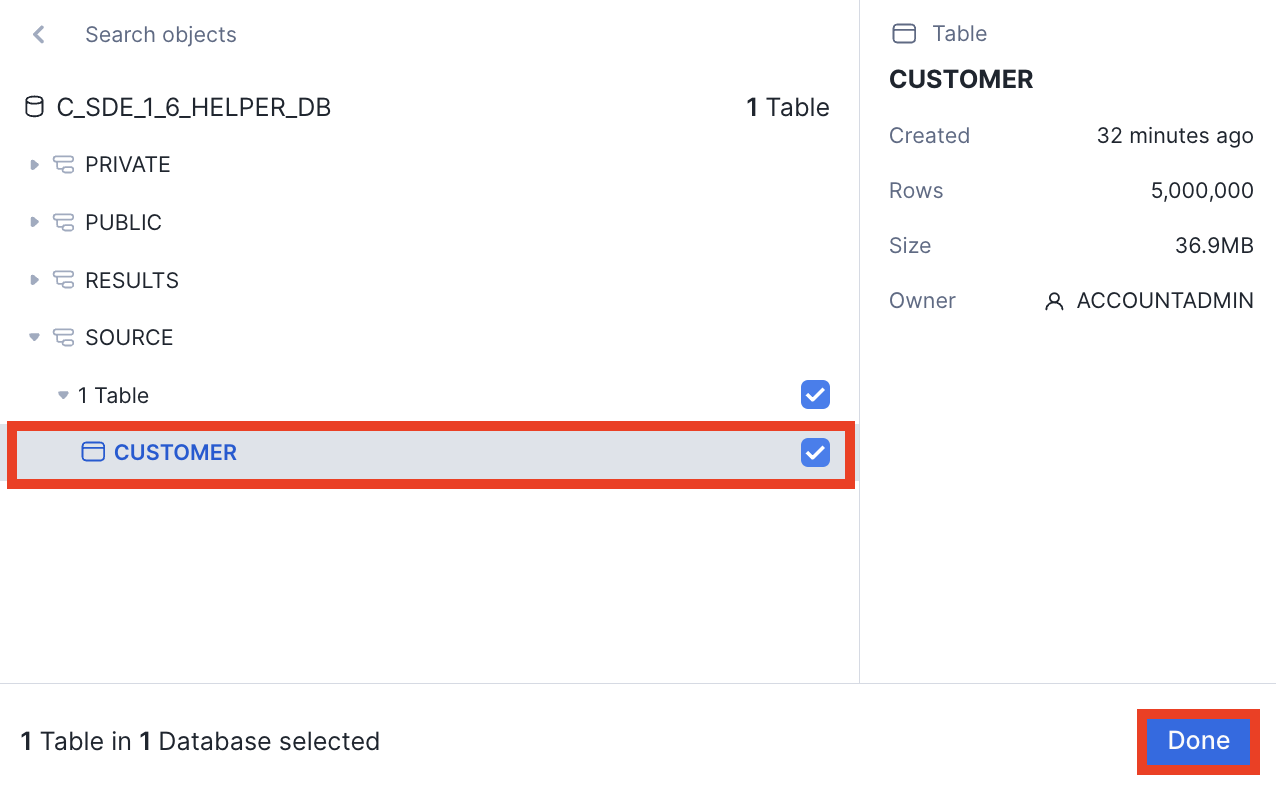
**Step 2**: Click **Select Table**. Since this is the first time using the app, it needs to be granted privileges to the sample data table.



**Step 3**: The app’s security window will appear. Click **+ Select Data**.



**Step 4**: Select **C\_<APP\_CODE>\_HELPER\_DB.SOURCE.CUSTOMERS**, then click **Done**.



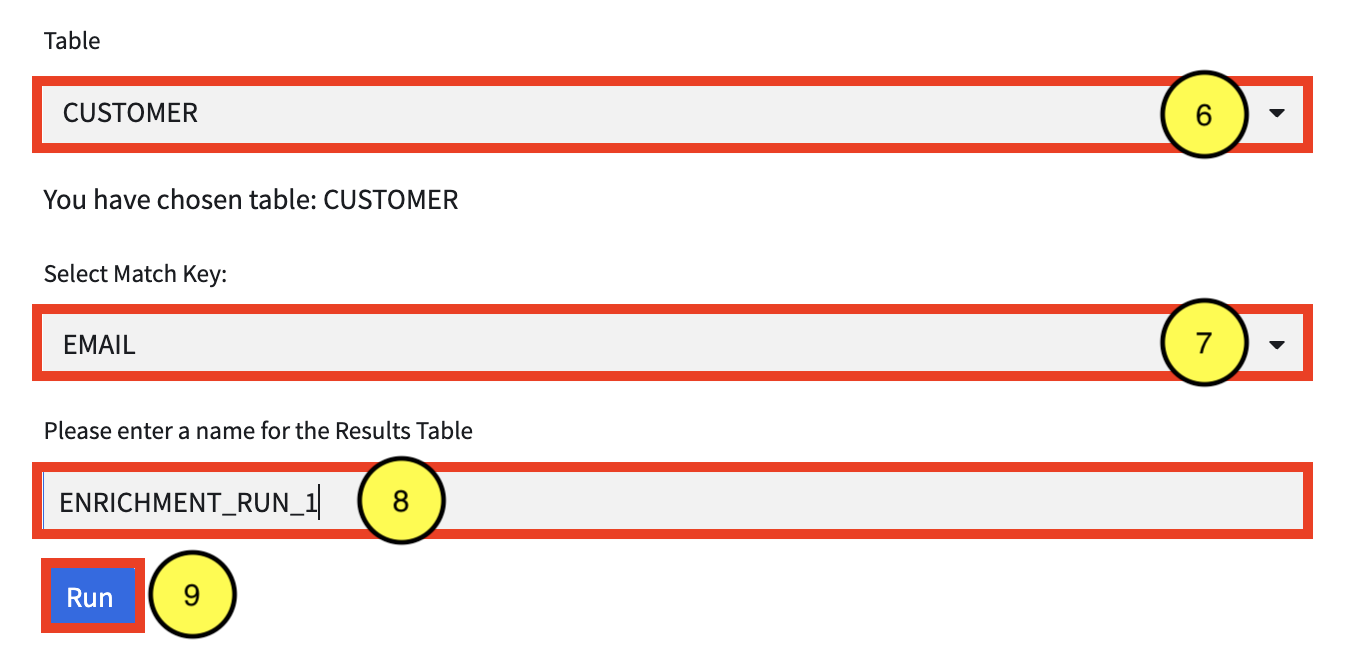
**Step 5**: In the security window, click **Save**.

**Step 6**: Verify that the **CUSTOMER** table is selected from the Table drop-down.

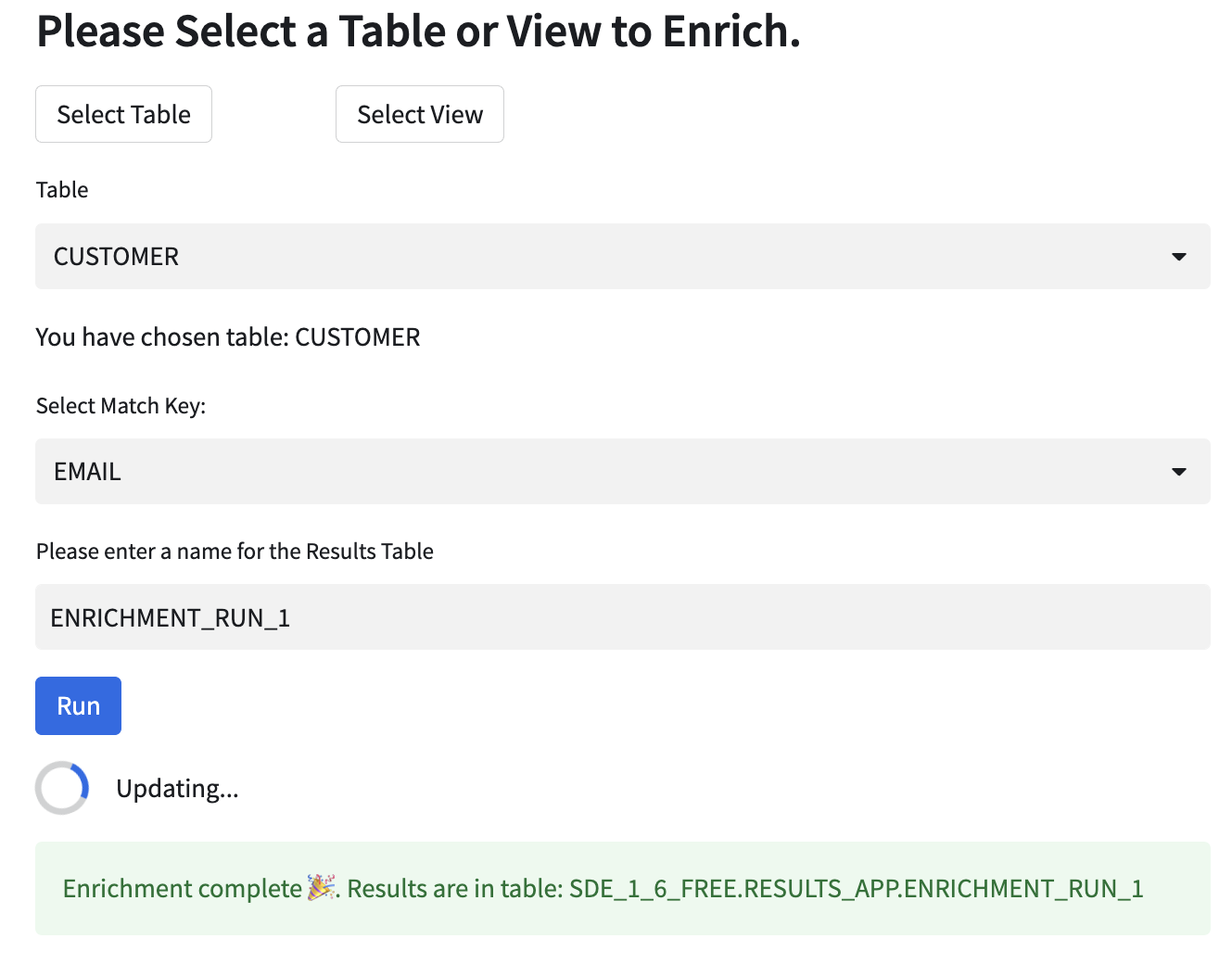
**Step 7**: Select **EMAIL** from the Match Key drop-down.

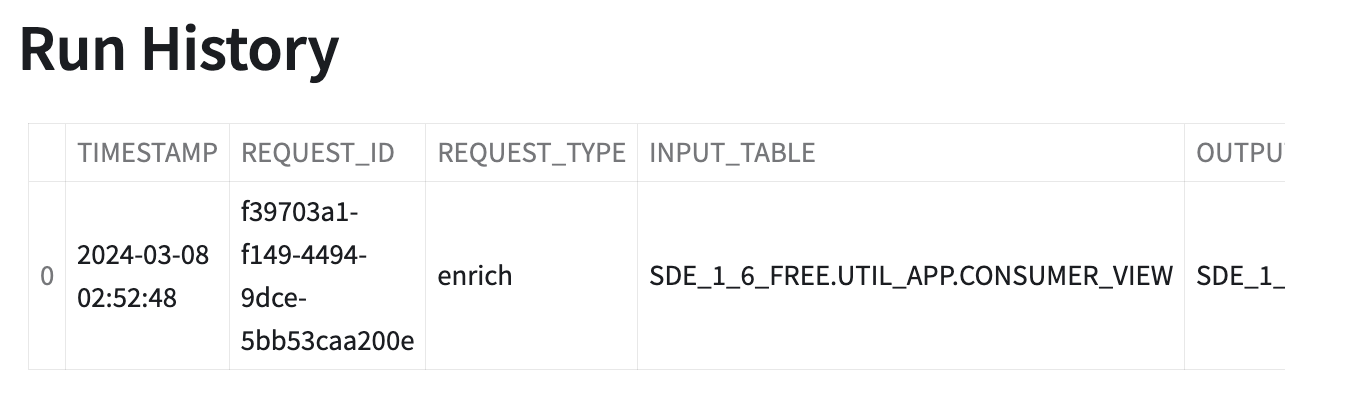
**Step 8**: Enter a **name** for the results table in the Please enter a name for the Results Table field.

**Step 9**: Click **Run**.



**Step 10**: Once complete, the following message will briefly appear. In addition a new entry will be added to the Run History section.





**Step 11**: If desired, the output table can be inspected. In a worksheet, execute the following command: SELECT \* FROM <APP\_NAME>.RESULTS\_APP.<TABLE\_NAME> WHERE email IS NOT NULL LIMIT 100;

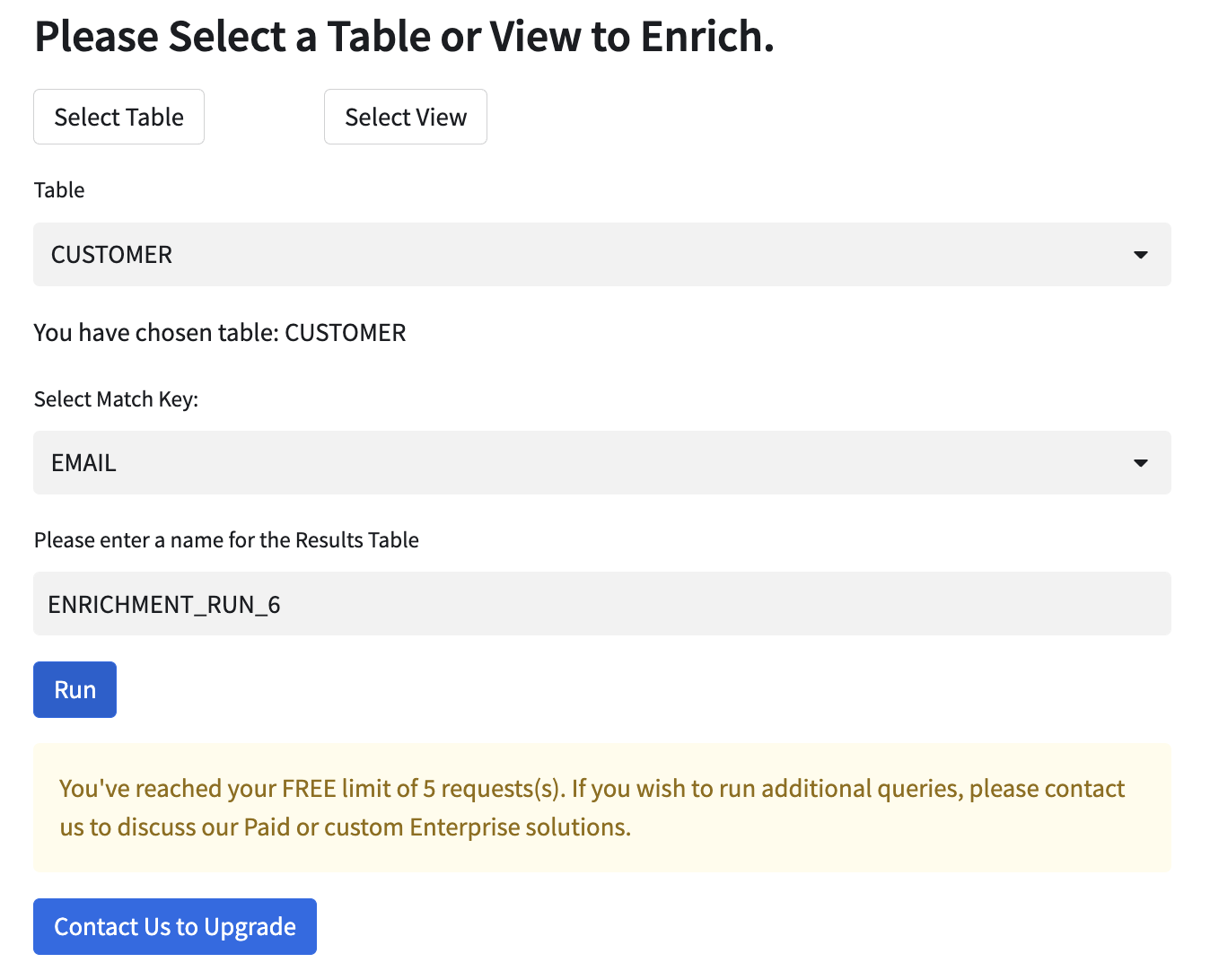
**NOTES:**

* <APP\_NAME> = the name of the app
* <TABLE\_NAME> = the name of the results table entered in Step 8.

# Additional Test Scenarios

## Reach the Five Requests Limit Then Increase the Limit

By default, the ACF sets limits of the FREE version of the app to five requests. Once the limit is reached, the app is no longer usable, until the limit is increased by the provider in the App Control Manager. In the app, make five requests to reach the limit, then attempt a sixth, which will fail.

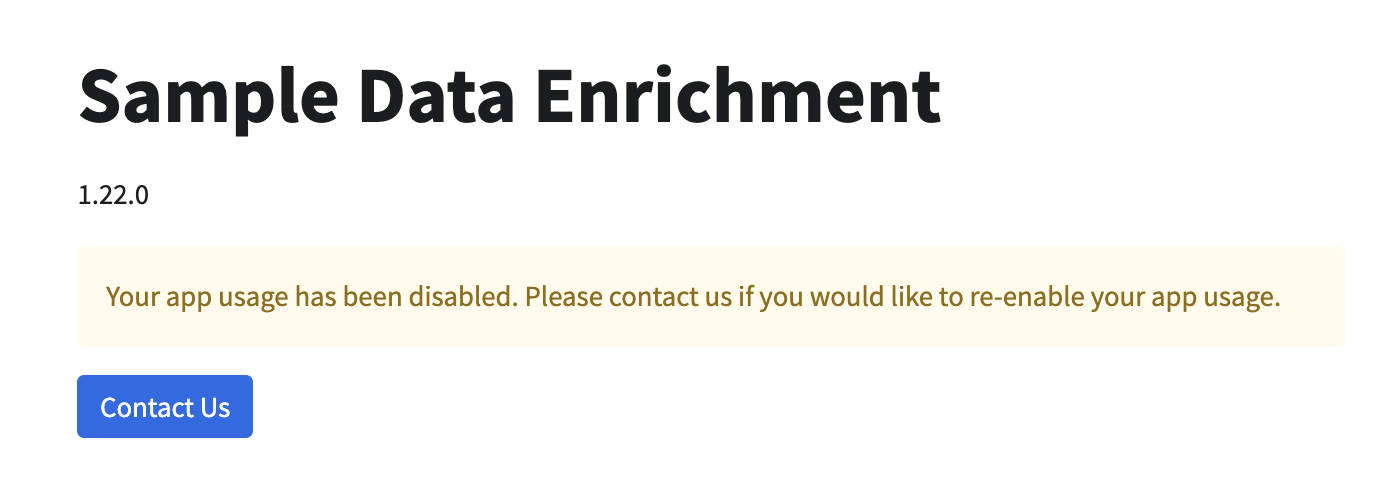


Once the limit is reached, increase the limit. Verify that the app runs, as normal.

## 

## Disable the Consumer

The App Control Manager allows the provider to enable/disable a consumer’s access to the app. In the App Control Manager, change the enable value to N. Then attempt to run the app, this should generate a message stating that usage has been disabled.



Switch the enable value back to Y and verify that app usage is enabled.