data
iku

Agenda

- *Dataiku introduction*
- *Demo - basic project*
- *Session 1 - hands-on Dataiku project*
- *Session 2 - hands-on Dataiku project*
- **Session 3 - hands-on Dataiku project**
    - Quick overview of Security Controls
    - Workflow automation
    - Metrics and checks

DSS Security Overview

Permission Model

Multi-faceted tools to control security in the system:

- Users:
  - Must exist to login into DSS
  - Belong to a GROUP
  - Have a PROFILE

- User Profile:
  - Mainly a licensing mechanism
  - Designer: R/W access
    - aka Data Scientist/Data Analyst
  - Explorer: R access only
    - aka Reader

- Group:
  - Collection of users
  - Defines Global Permissions (i.e. are you an admin? Can you create connections? etc)

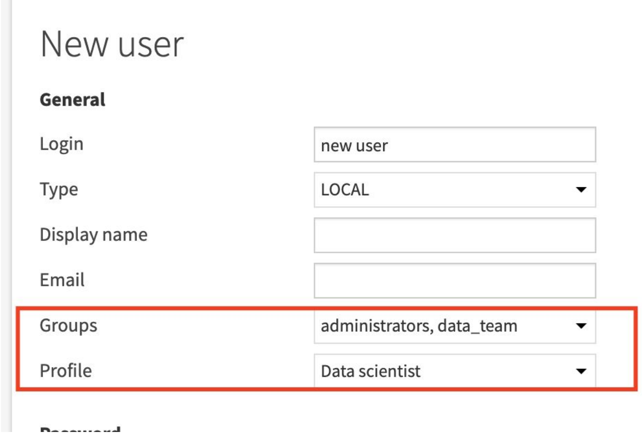
- Projects:
  - Determines privilege of each GROUP
  - Can enforce project-level settings (lock code env, etc)

- Data Connections:
  - Grant access to GROUPS
  - Some connections allow per-user credentials

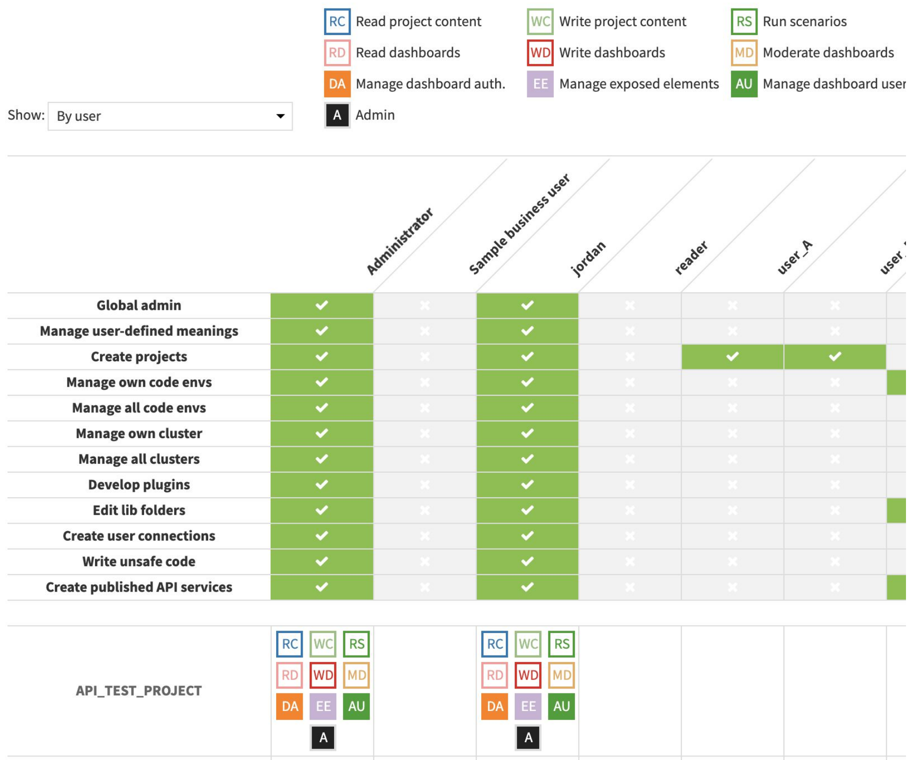
Permission Model

Users

- Users get assigned profile + group.
    - Can determine this automatically via mapping rules, as discussed previously



- Auth Matrix shows all projects that a user has access to and privileges granted. Ditto for groups.



Permission Model

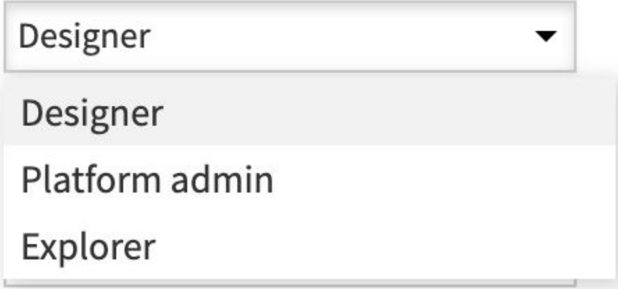
User Profiles

Each user in DSS has a single “user profile” assigned to it.

The three possible profiles are:

- **Explorer**: users with this profile only have access to dashboards and DIY visualisations on shared items
- **Designer**: data analysts can create datasets, perform visualizations, use all visual processing recipes, and more generally perform most of the actions in the DSS interface. They can also use code-based recipes (Python, R, …) and the machine learning components of DSS
- (Visual Designer): same as designer but no code
- **Platform Admin**: perform all admin configurations of DSS

User profile is not a security feature, but a licensing-related concept. DSS licenses are restricted in number of each profile. Use the regular groups authorization model described later.



-

Permission Model

Global Group Permissions

Edit group "my_group"

Description	my group description
Type	LOCAL

GLOBAL PERMISSIONS

Administrator	☐	Grants all permissions on all scopes
Manage user-defined meanings	☐	Can these users create/delete global UDMs?
Create projects	☐	Can these users create their own projects?
Write unsafe code	☐	Can these users run local code (Python, R) which runs as DSS service user? Hostile users with this permission may circumvent the permissions system.
Manage all code envs	☐	Can these users manage code envs, even others' ?
Manage own code envs	☐	Can these users create code envs, manage code envs they created or were given access to?
Manage all clusters	☐	Can these users manage clusters, even others' ?
Manage own clusters	☐	Can these users create clusters, manage clusters they created or were given access to?
Develop plugins	☐	Can these users create and edit development plugins? Hostile users with this permission may circumvent the permissions system.
Edit lib folders	☐	Can these users edit the Python & R libraries and the static web resources in the DSS instance? Hostile users with this permission may circumvent the permissions system.
Create connections	☐	Can these users create personal connections? Does not include FS connections
View indexed Hive connections	☐	Can these users view indexed Hive connections in the catalog?
Create published API services	☐	Can these users create a published API service?

SAVE

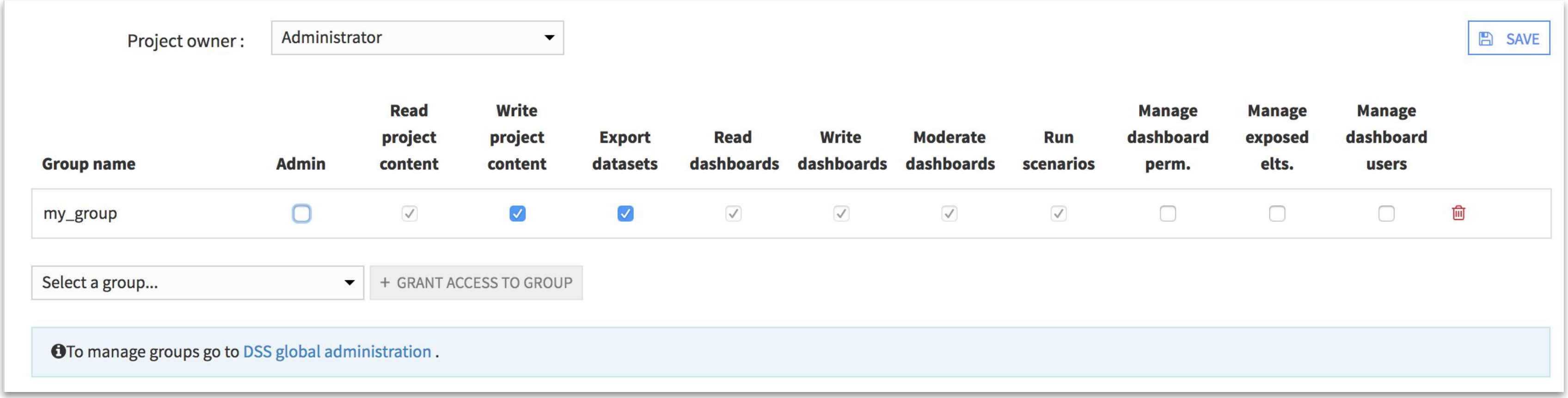
Users can be assigned to one or more groups. Groups are defined by permissions their members are granted (e.g. write code, create projects, access to projects etc)

Do not rely on user profiles to enforce permissions. We do not provide any guarantee that the user profile is strictly applied. For real security, use groups.

We will also see that per-project permissions can be defined to curb permissions of the users that have access to the project (except for members of an "Administrator" group)

Permission Model

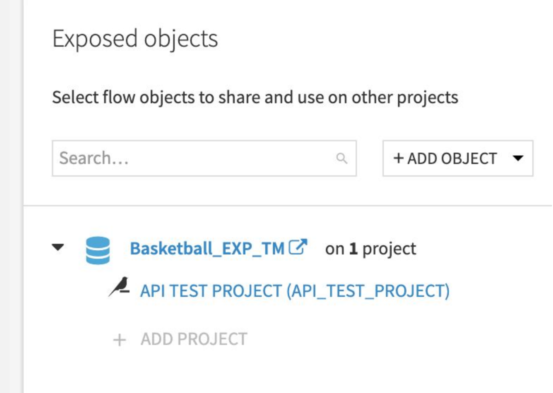
Per-Project Group Permissions



- On each project, you can configure an arbitrary number of groups who have access to this project. Adding permissions to projects is done in the Project Settings, in the Security section.
- Each group can have one or several of the following permissions. By default, groups don't have any kind of access to a project.
- Being the owner of a project does not grant any additional permissions compared to being in a group who has Administrator access to this project.
- This owner status is used mainly to actually grant access to a project to the user who just created it.

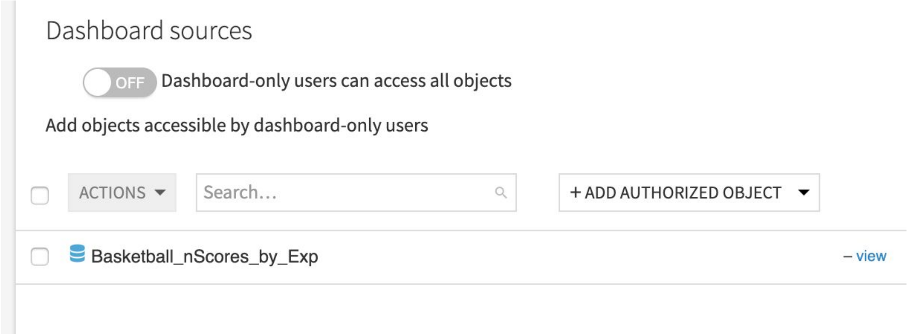
Permission Model

Additional Project Security

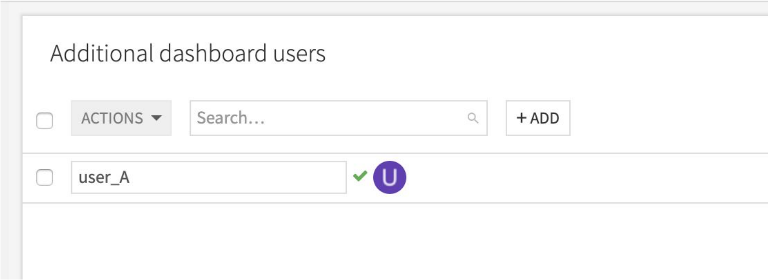


PROJECT > SECURITY can manage other aspects of security:

- Exposed Elements
  - High level view of which elements are exposed to other projects. Project admins can modify.
- Dashboard Authorizations
  - Which Objects can be accessed Dashboard-only users



- Dashboard Users
  - Add external users who are able to access Dashboards

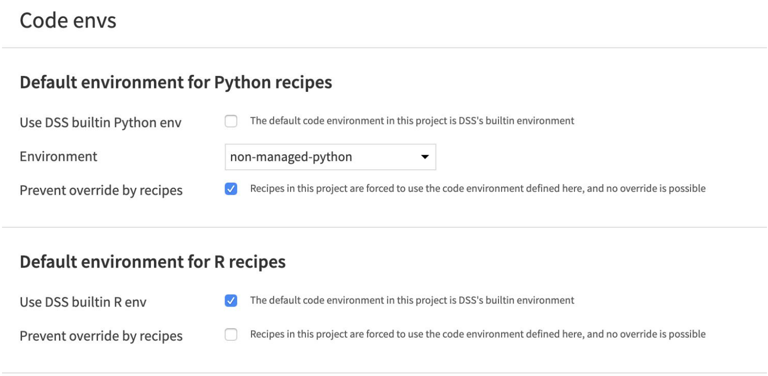


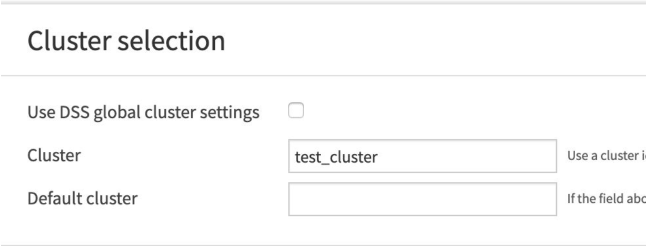
Permission Model

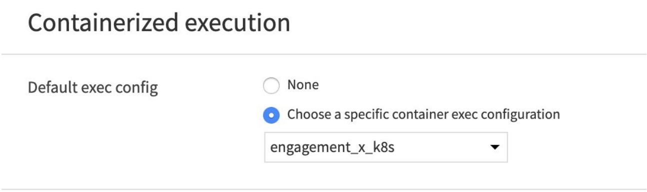
Additional Project Settings

PROJECT > SETTINGS can manage other aspects of configuration:

- Code Envs
  - Set default code env and prevent modification
- Cluster Selection
  - Select default Cluster to use
- Container Exec
  - Specify default container env
- Engines & Connections
  - Restrict Engines for use in Recipes
  - Change default Spark/Hive config



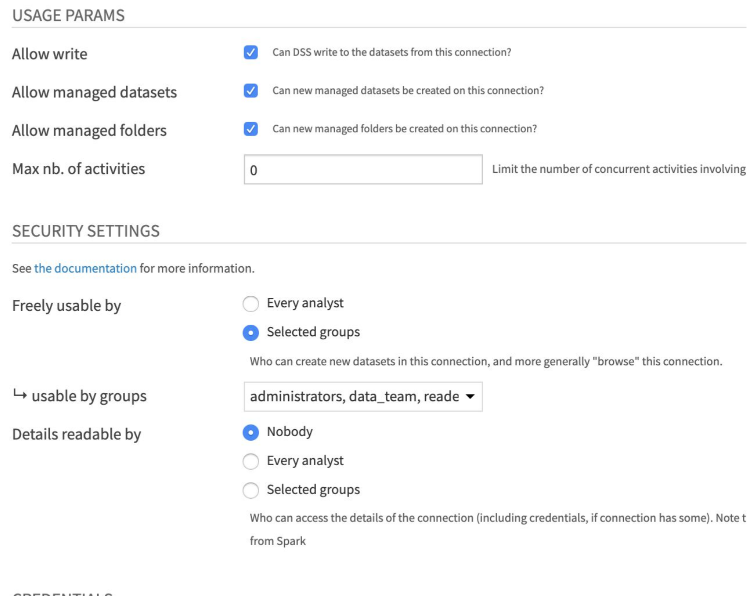


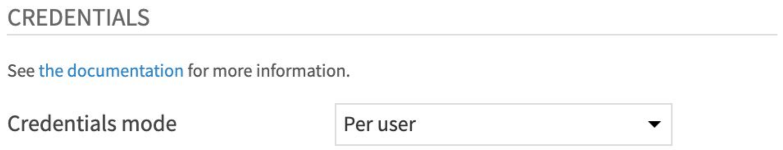


Permission Model

Data Connections

- Data Connections should be restricted to only groups who should have access.
- You can create many connections and limit use + details readability group by group. Details include file path, connection params, credentials, etc.
- Connections can be made read only
- Some connection support per-user credentials (DB, etc). Users can then specify in their User settings.



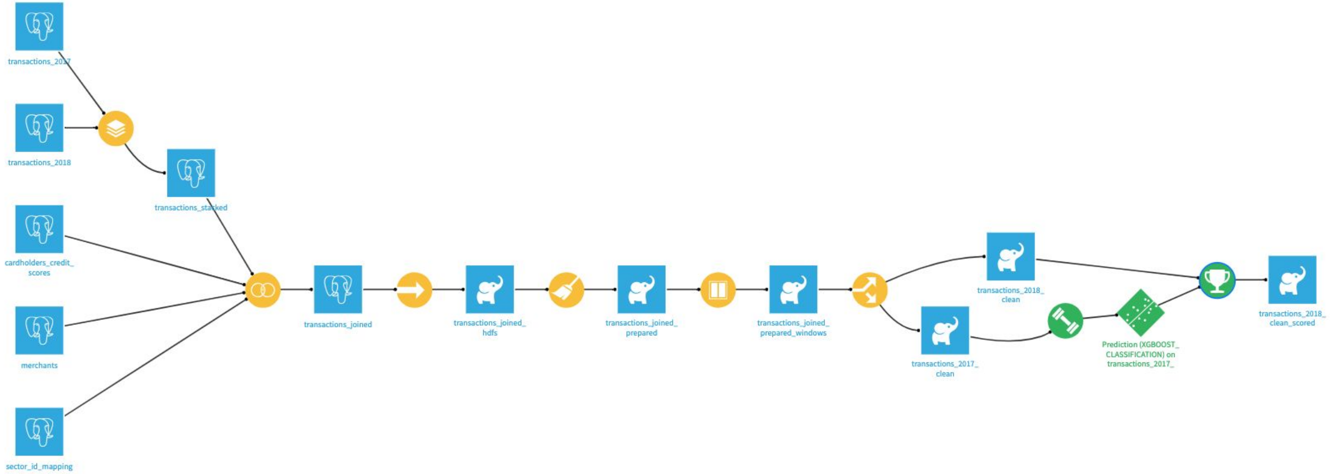


Credit Card Transactions

Initial Project

Background

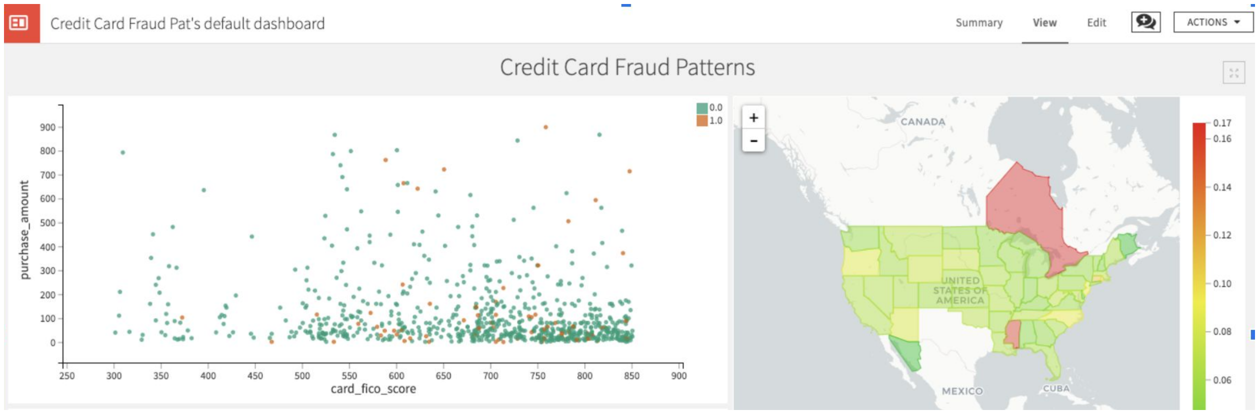
During the Discovery training, we created a pipeline and a model to predict which credit card transactions are fraudulent



Automation in DSS

Goal

Automate the data processing pipeline to run daily and predict which transactions from the previous day were fraudulent



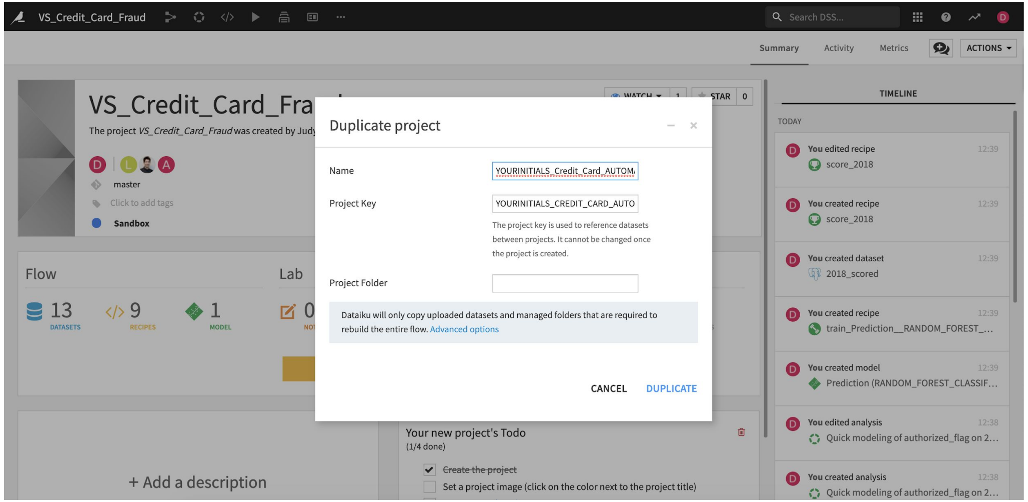
Login
Import the Project

Copy an existing Project into DSS

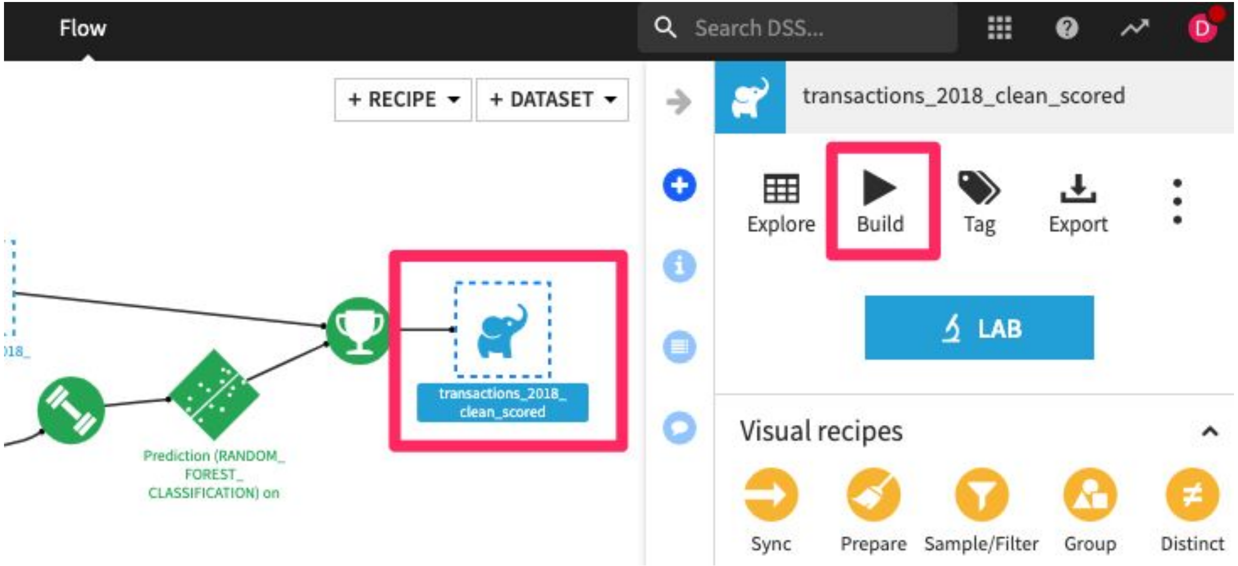
You can Duplicate an existing project from Dataiku DSS

https://dss0-design-scb.training2.dataiku.com/projects/VS_CREDIT_CARD_FRAUD/

Change the project key to ‘`YOURINITIALS_CREDIT_CARD_AUTOMATION`’



Recursively Build the Project Flow



Data Quality: Metrics and Checks

Data Quality

Objective :

Automate our transaction processing pipeline to run everyday.

Problem :

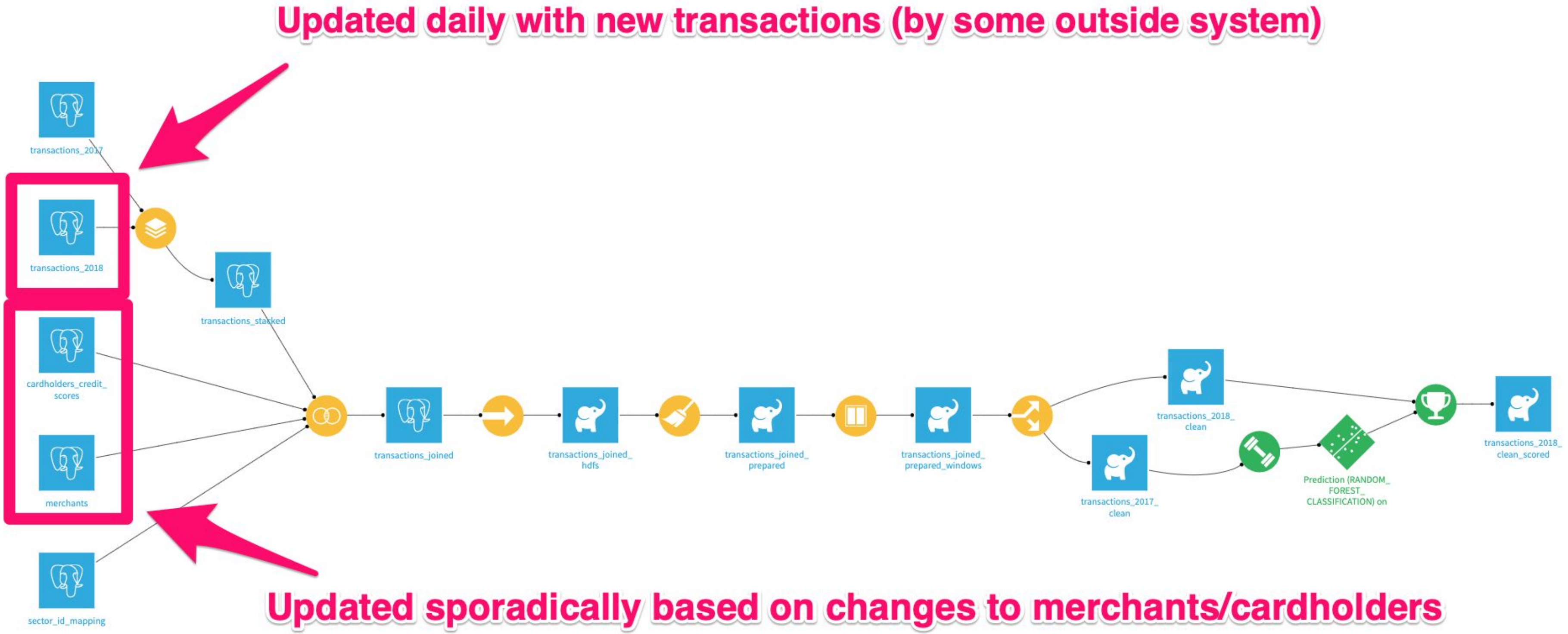
As we automate workflows, we risk ingesting bad quality data without knowing it, which could affect dashboards, models, and output datasets.

ex : An extra column is added to an input dataset, which breaks the flow.

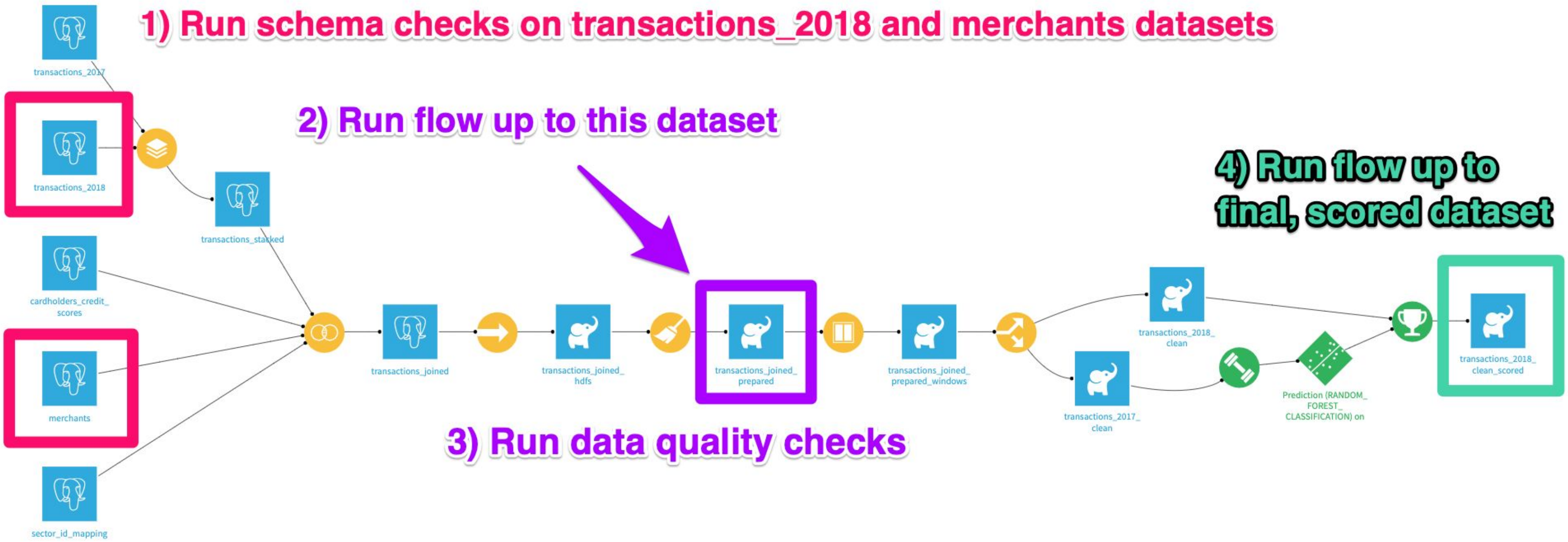
Solution :

Add important metrics and checks on key datasets in the flow.

Our Automated Workflow



Our Automated Workflow



Metrics

Metrics are used to take measurements on flow items (datasets, managed folders, and saved models).

Ex.

- **Record Count** of a Dataset
- **Avg** column value
- **Size, Mb** of a Dataset
- **Number of Distinct Values** of a column
- Model **Accuracy**
- ...many other, including using custom python code

Checks

Use Checks to validate that the latest computed Metrics do not raise any concerns

- Defined from metrics, custom python code or plugins
- Return OK, WARNING, ERROR or EMPTY after each run
- Used in Scenarios, for example, to stop Flow rebuilds or send warning messages

Ex.

- Check that a dataset has at least 10000 rows
- The average value of a column is above 0.2
- The number of columns is 22
- The AUC of a model is greater than 0.85

Create Metrics and Checks

Create metrics and checks on the 'transactions_2018' and 'merchants' datasets

Metrics (both datasets)

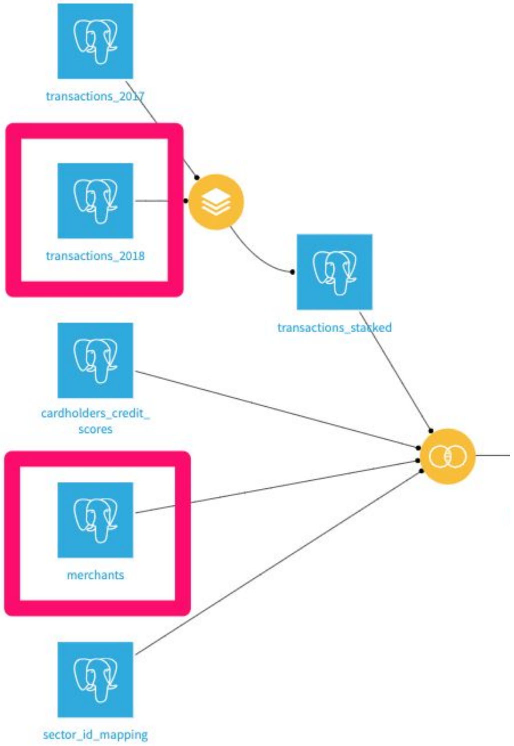
- Column count
- Record Count

Checks (transactions_2018)

- Column count = 13
- Record Count > 10,000

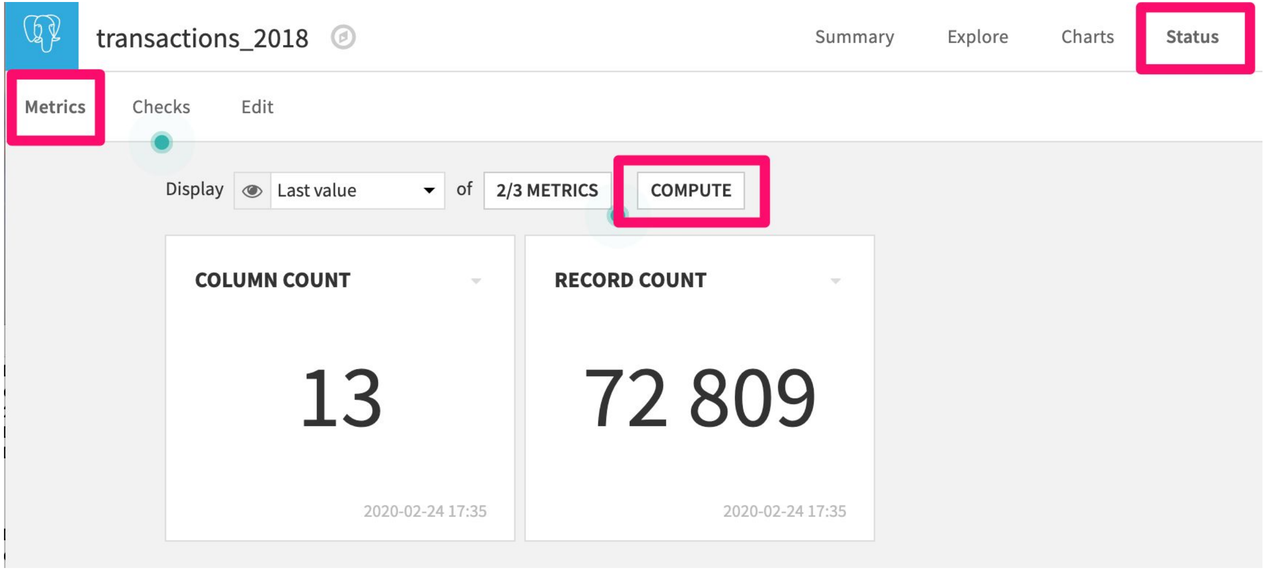
Checks (merchants)

- Column count = 7
- Record Count > 10,000



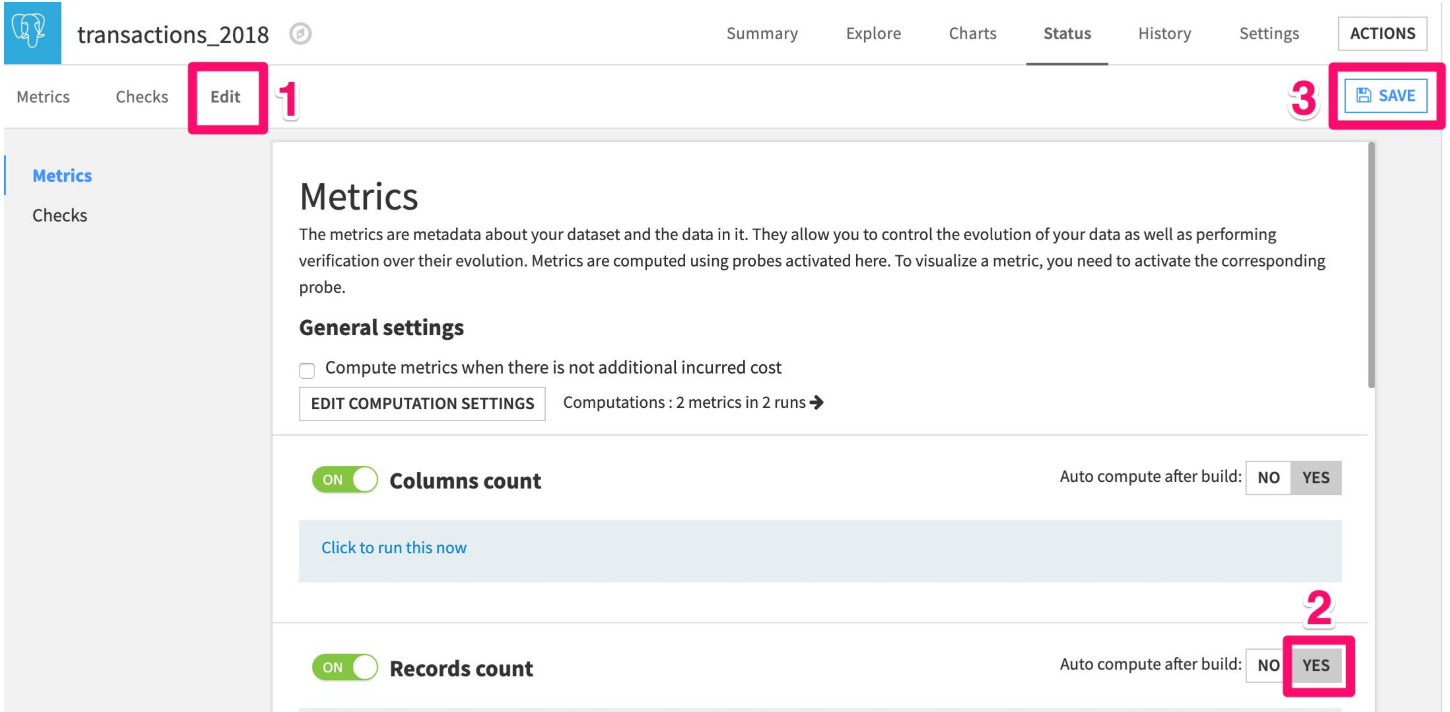
Create Metrics - ‘transactions_2018’

Column count and record count are added by default



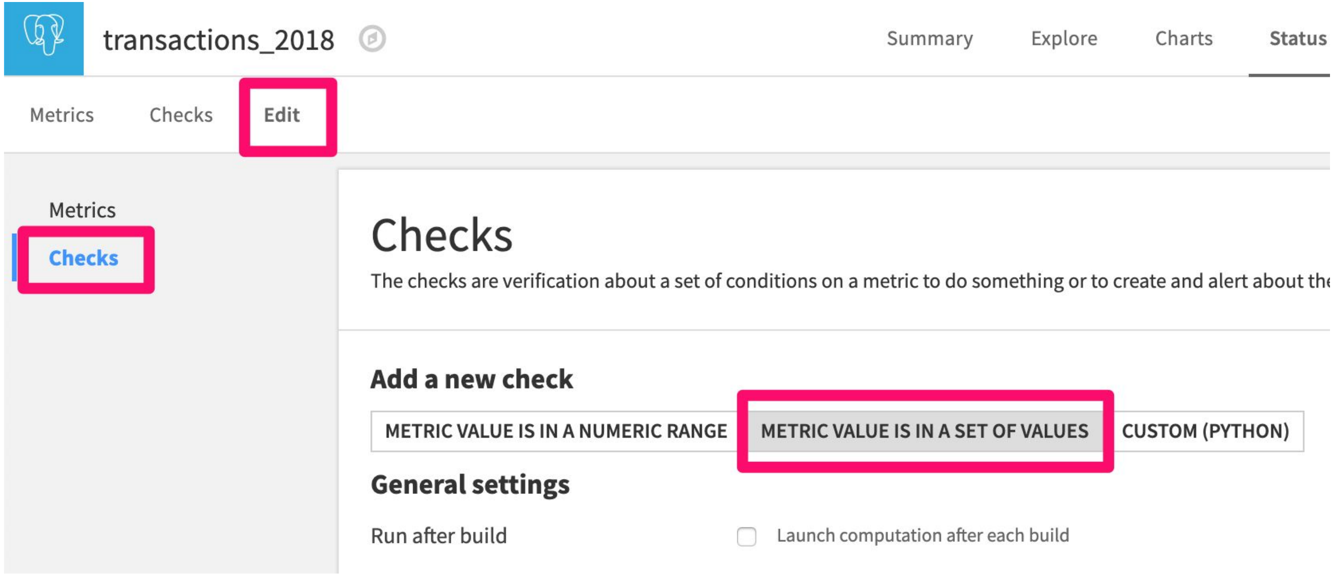
Create Metrics - ‘transactions_2018’

Enable metric computation to happen automatically whenever the dataset is rebuilt



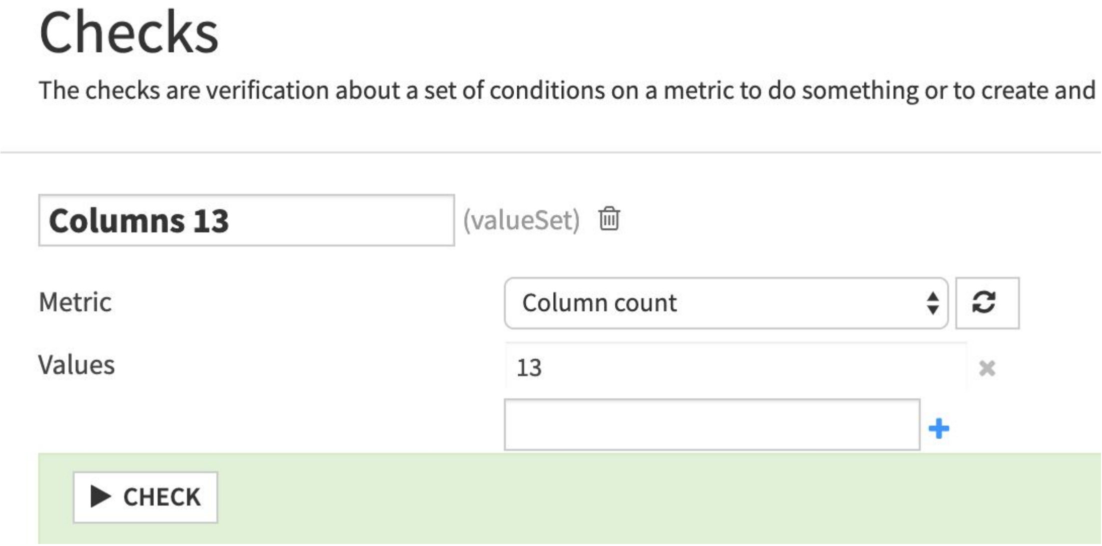
Create Metric Checks - ‘transactions_2018’

Add a new check: column count is 13



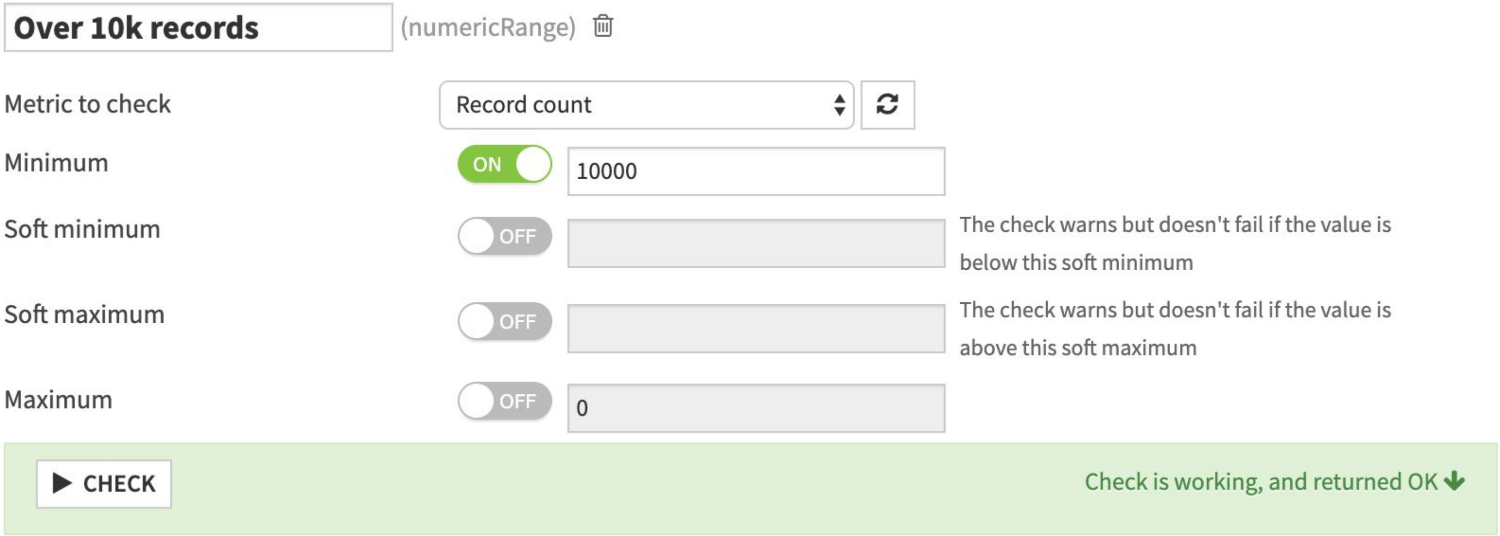
Create Metric Checks - ‘transactions_2018’

Add a new check: column count is 13



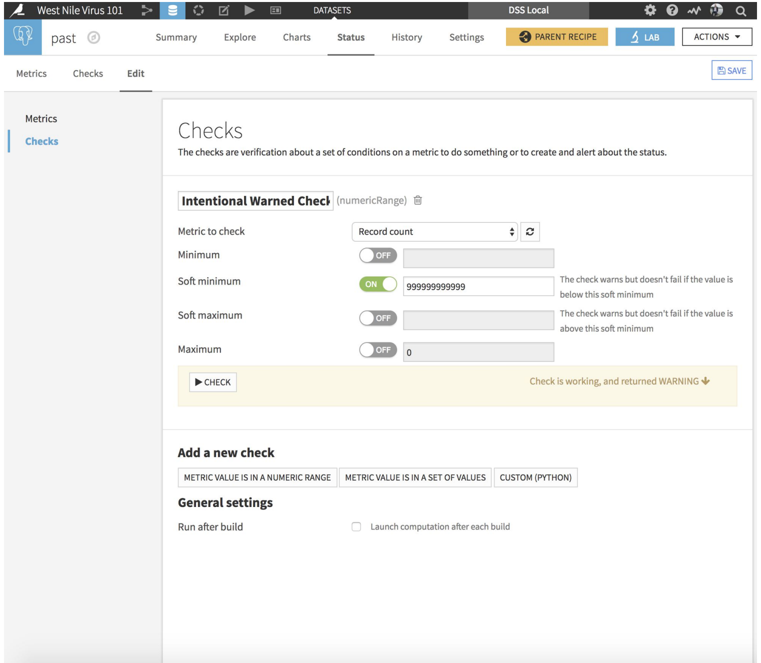
Create Metric **Checks** - ‘transactions_2018’

Add a new check: record count is greater than 10,000

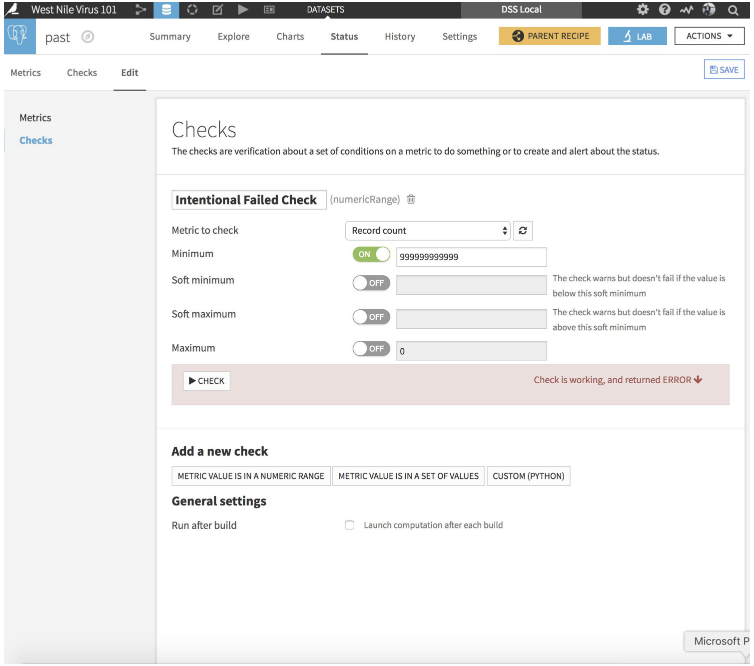


Hard vs. Soft Min/Max for Checks

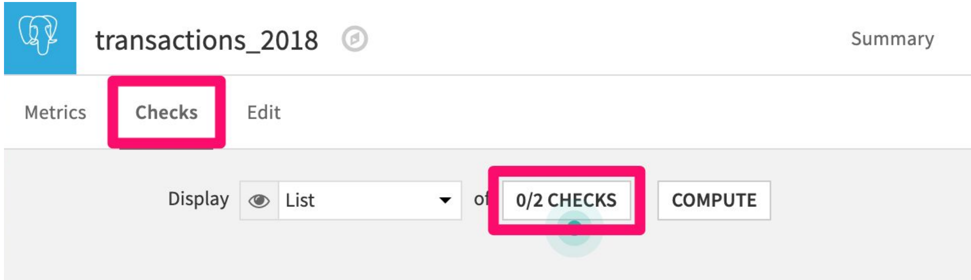
Soft min/max causes a *Warning*
→ *Scenario continues running*



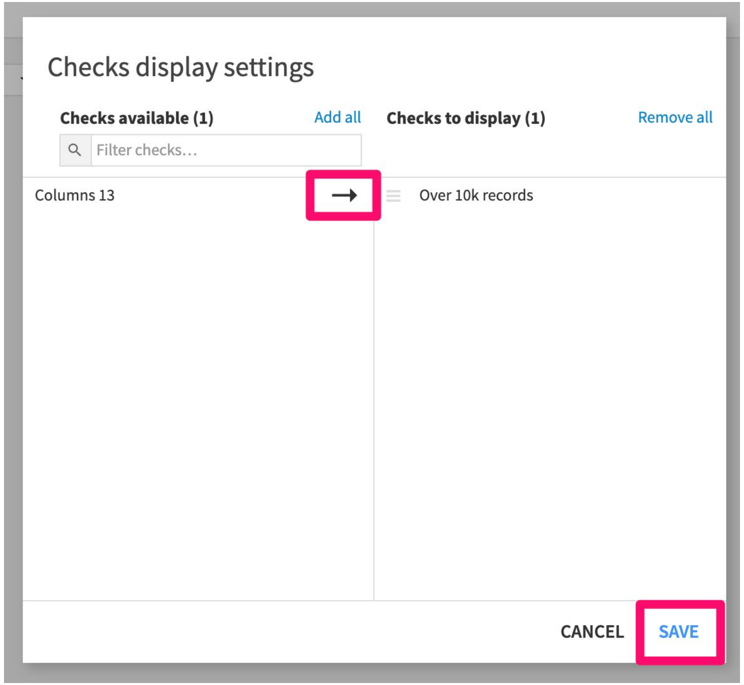
Hard min/max causes a *Fail*
→ *Scenario stops*



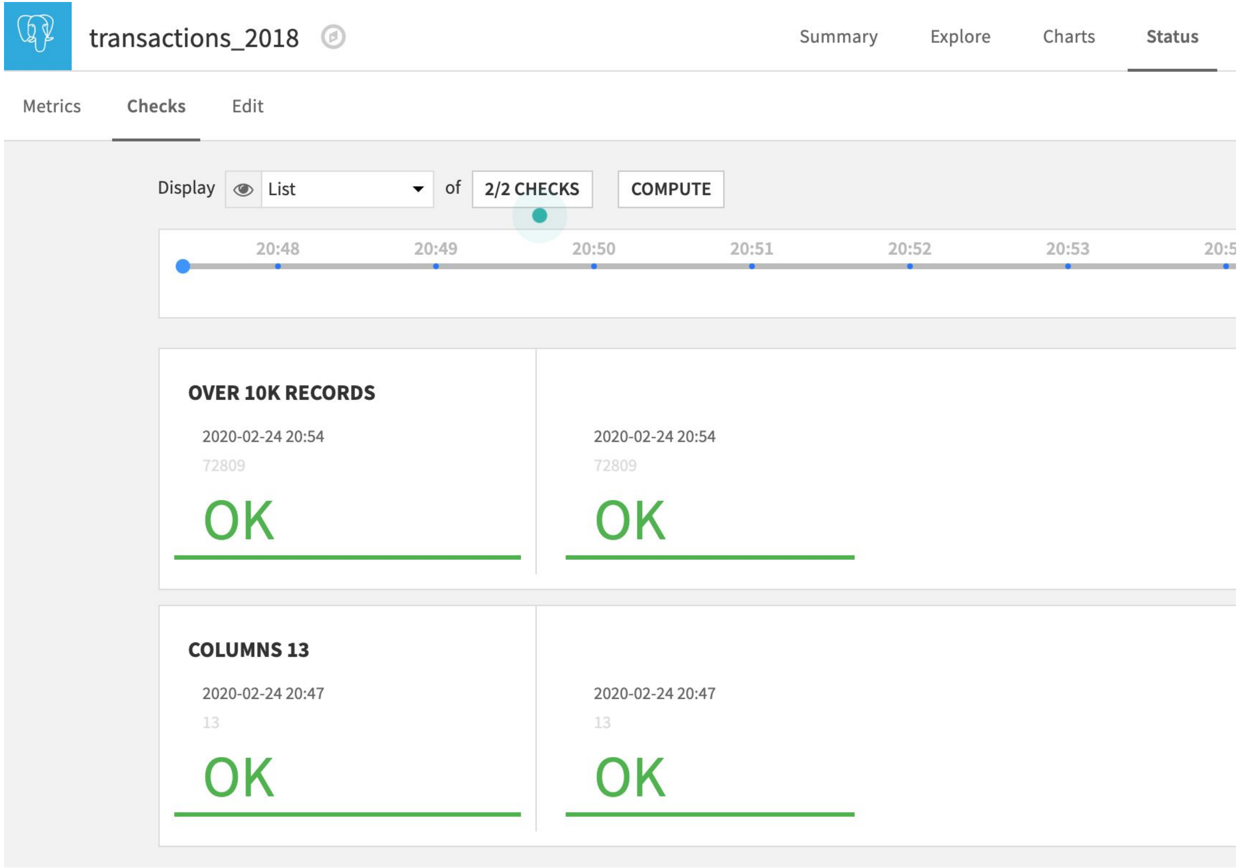
Add Checks to the home screen - ‘transactions_2018’



Bring the new checks to the main tile screen



You can monitor Check history



Create Metrics and Checks - ‘`merchants`’

Now add **metrics** and **checks** for the ‘`merchants`’ dataset

Metrics

- Column count
- Record Count

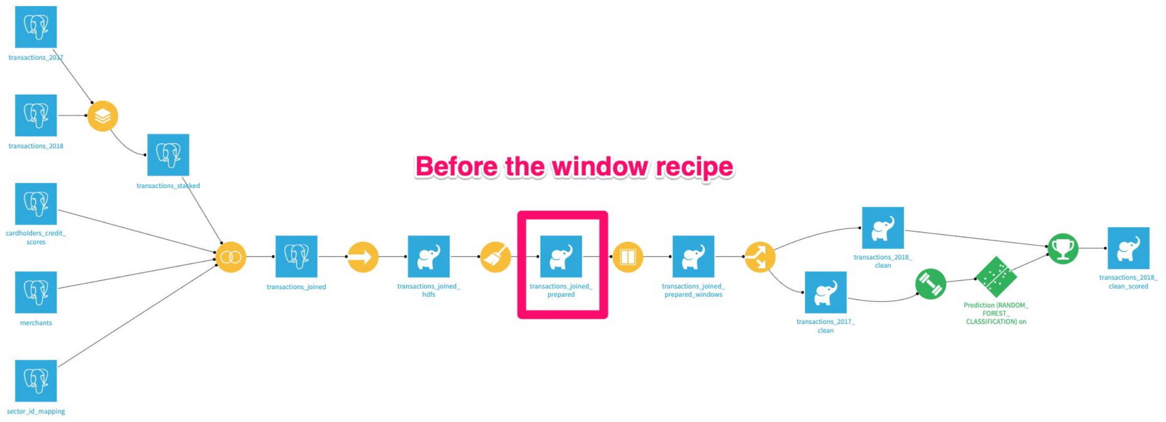
Checks

- Column count = 15
- Record Count > 10,000

Create **Metrics** and **Checks** - ‘transactions_joined_prepared’

After checking that the column count and record count of ‘transactions_2018’ and ‘merchants’ are ok, our scenario (yet to be created) can rebuild the flow up to a key dataset, ‘transactions_joined_prepared’

Here, we’ll want to do more advanced metrics and checks



Create Metrics and Checks - ‘transactions_joined_prepared’

Add two metrics

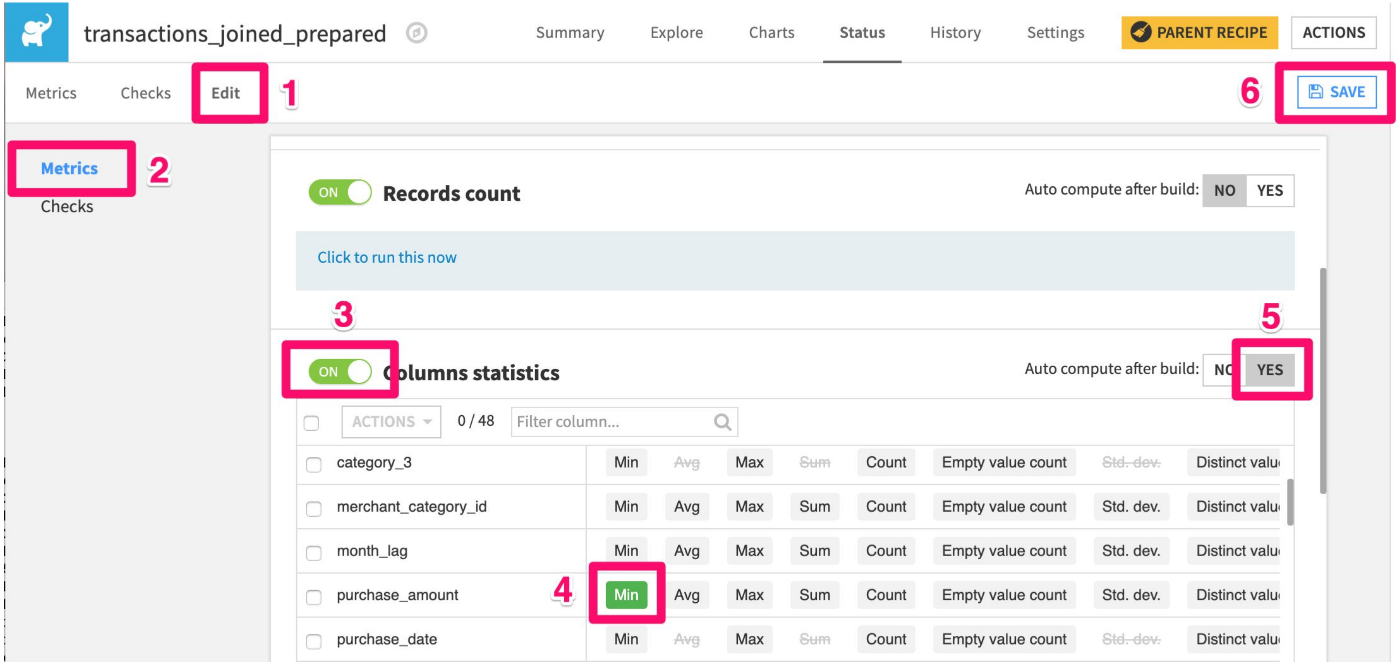
- Minimum purchase_amount
- Average fico_score

Then add two checks

- Minimum purchase_amount > 0
- 525 < Average fico_score < 650

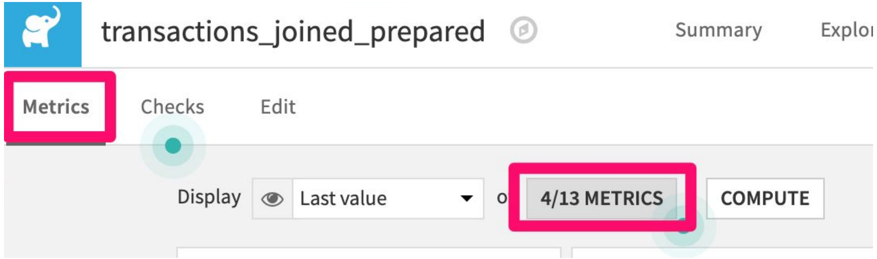
Create Metrics - ‘transactions_joined_prepared’

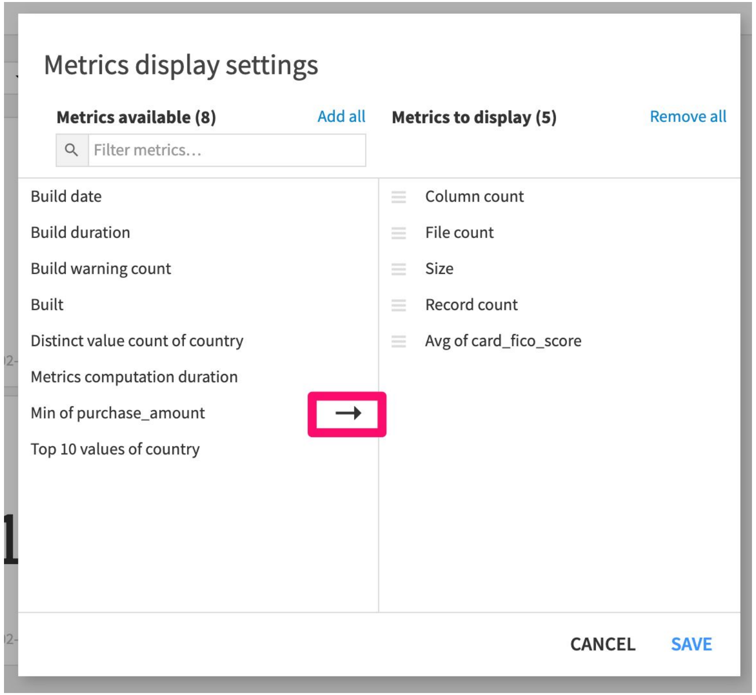
Add two metrics
- Minimum purchase_amount
- Average fico_score



Create Metrics - ‘transactions_joined_prepared’

Add these two metrics to the home screen tiles

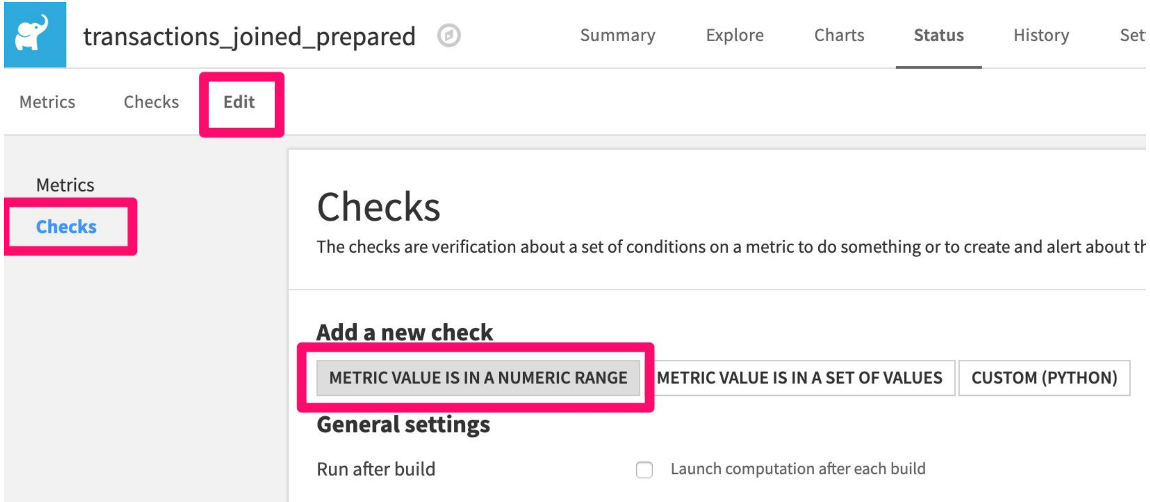




Create Metric Checks - ‘transactions_joined_prepared’

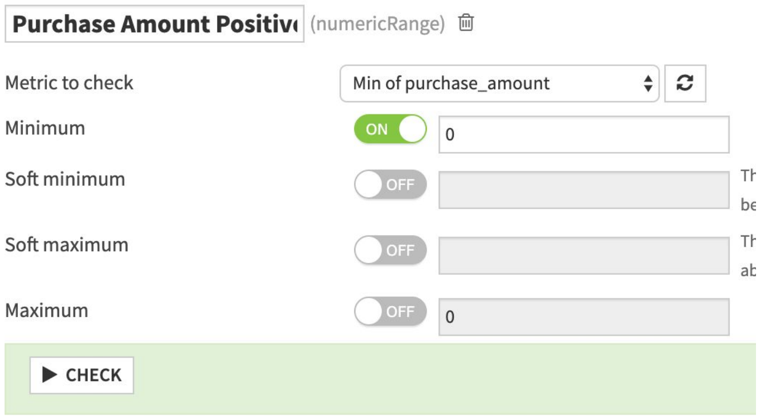
Add two checks

- Minimum purchase_amount > 0
- 525 > Average fico_score < 650

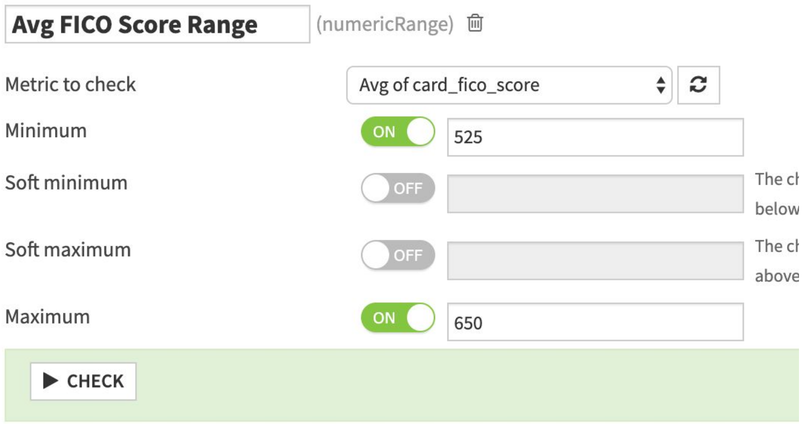


Create Metric Checks - ‘transactions_joined_prepared’

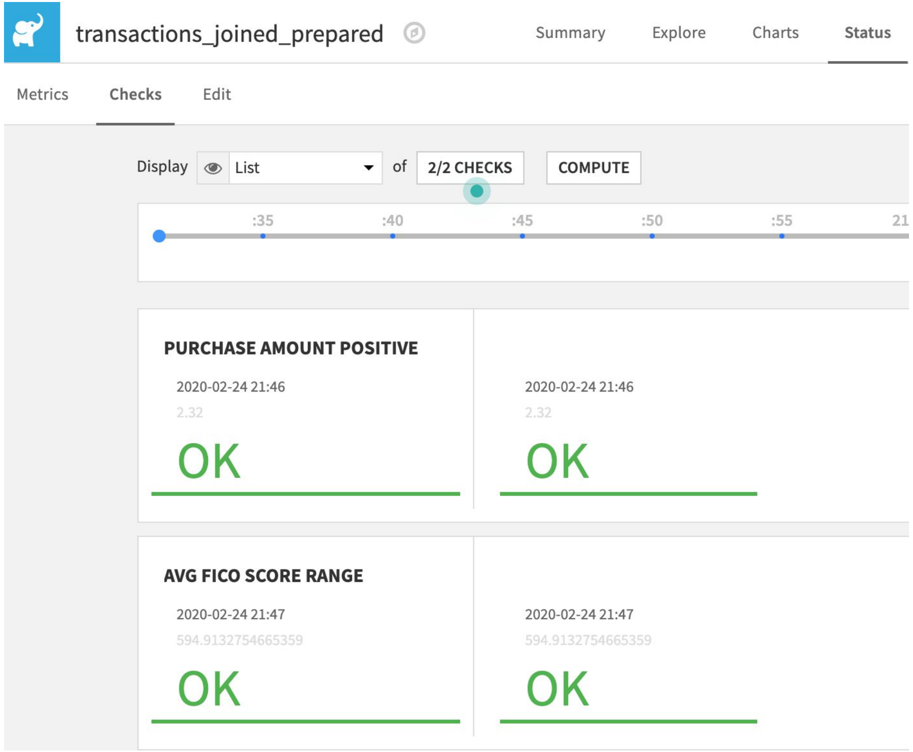
Minimum purchase_amount > 0



525 > Average fico_score < 650



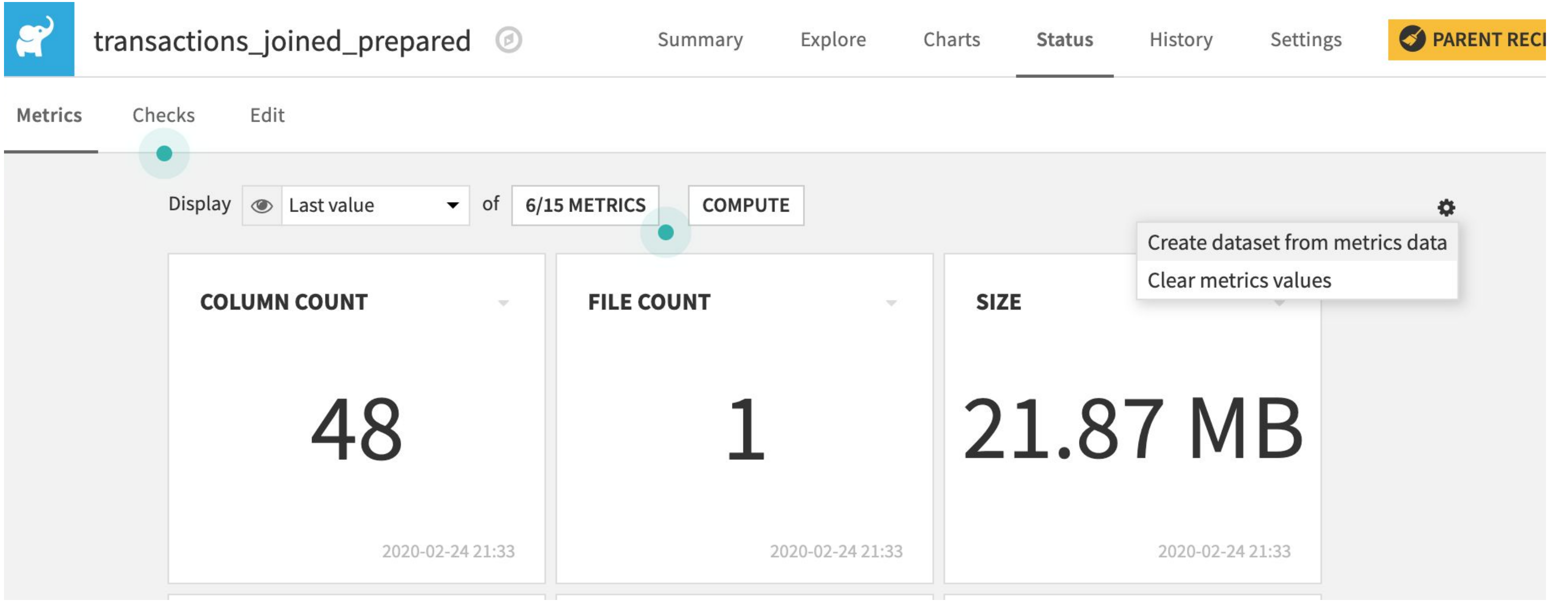
Add Checks to the home screen - ‘transactions_joined_prepared’



Exporting Metrics and Checks to a Dataset

It is possible to store the results of metrics and checks to a dataset.

Go to the metrics or checks tab and click on the gear sign.



Onto Scenarios

We have seen how Metrics and Checks could be defined and run to check data quality.

Although not covered in this training, Metrics and Checks can also be used on Folders and Saved Models.

However, checks alone won't let us achieve our goal of a fully productionized data pipeline.

It is now time to introduce another powerful DSS concept: Scenarios

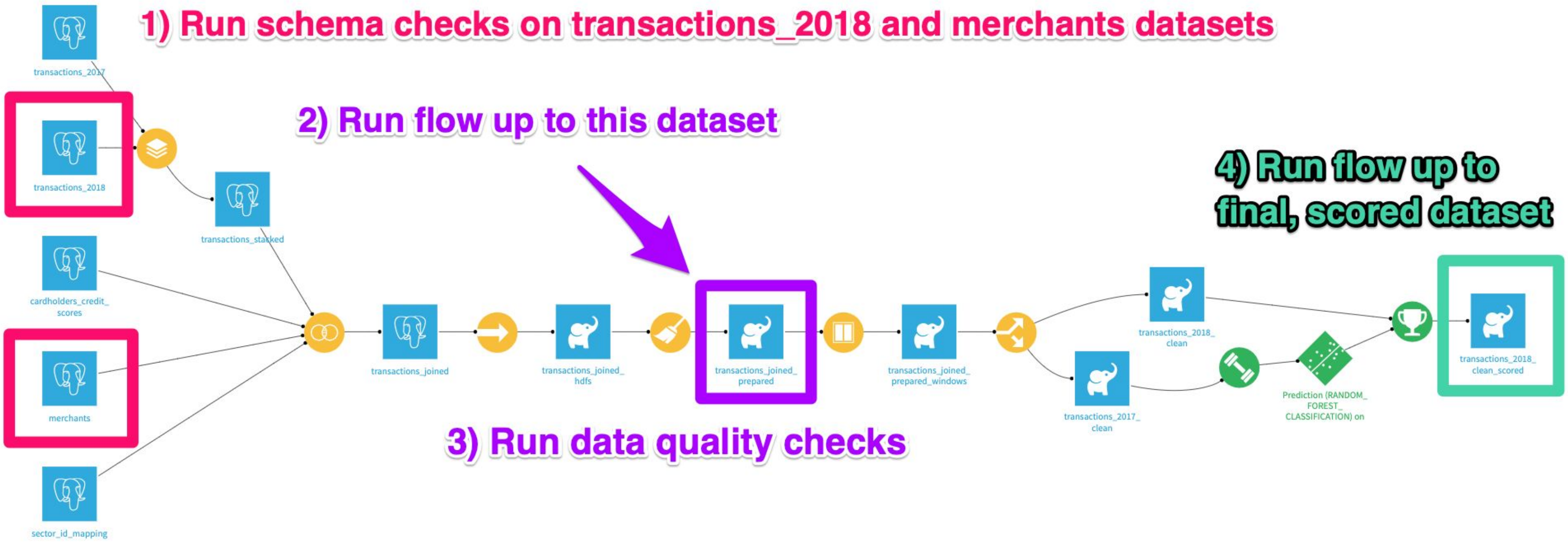
Automation: Scenarios

Scenarios

Scenarios consist of three main components:

1) Sequence of Steps or Custom Python script:
   - Build datasets
   - Train model
   - Perform checks
   - Run code snippets
   - And much more
2) Triggers:
   - Define when to execute the above steps or script.
   - Time-based, on Dataset change, after other scenario, custom python or SQL query
3) Reporters:
   - Information about Scenarios (success, fail, warning or any ad- hoc info on steps) can be sent to users via email, Slack, Microsoft Teams, store in dataset, shell command, slack etc...

Recall our Automated Workflow



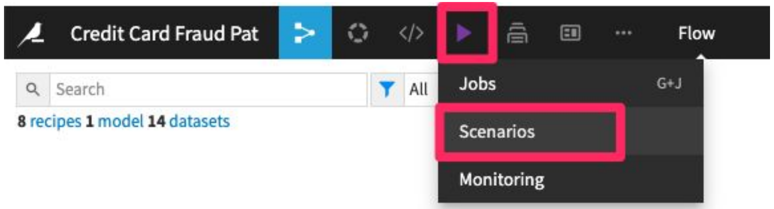
Create Scenario Trigger, Reporter, and Steps

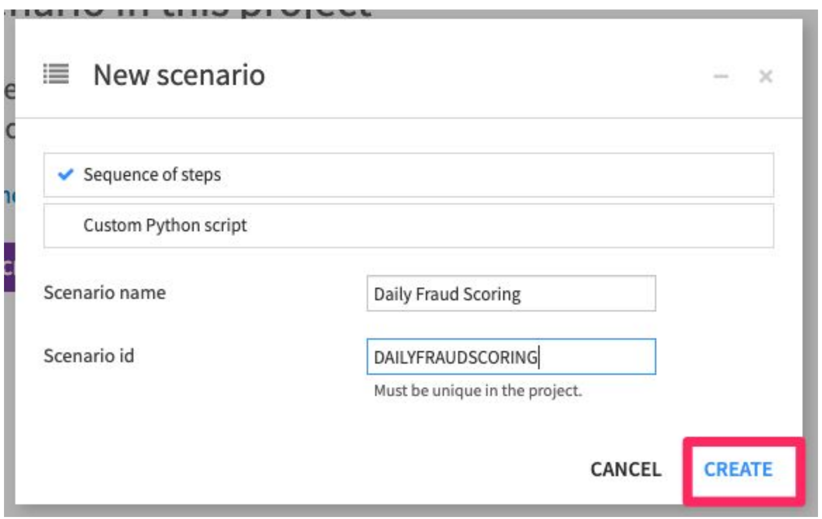
1. Add a Time-based Trigger:
   a. Scenario will run every day at 2AM

2. Add a Reporter to email us details of the scenario

3. Execute Steps
   a. Run checks on ‘`transactions_2018`’ and ‘`merchants`’
   b. Build ‘`transactions_joined_prepared`’
   c. Run checks on ‘`transactions_joined_prepared`’
      - Stop the scenario if the check Fails
   d. Build ‘`transactions_2018_clean_scored`’

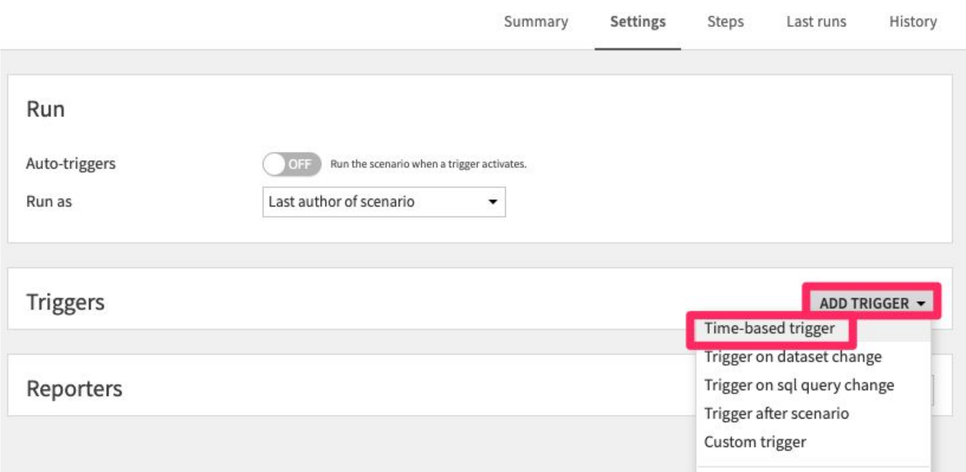
Create Scenario

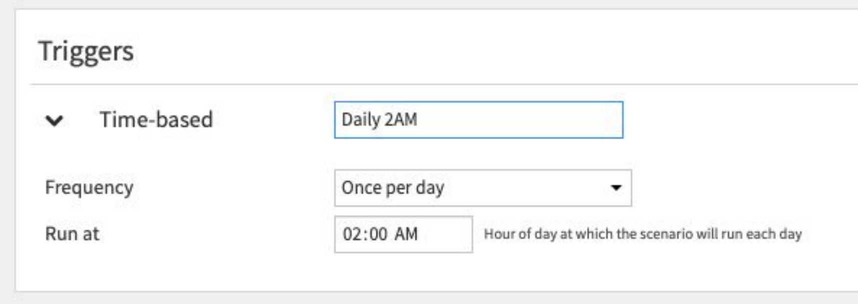




Create Scenario Trigger

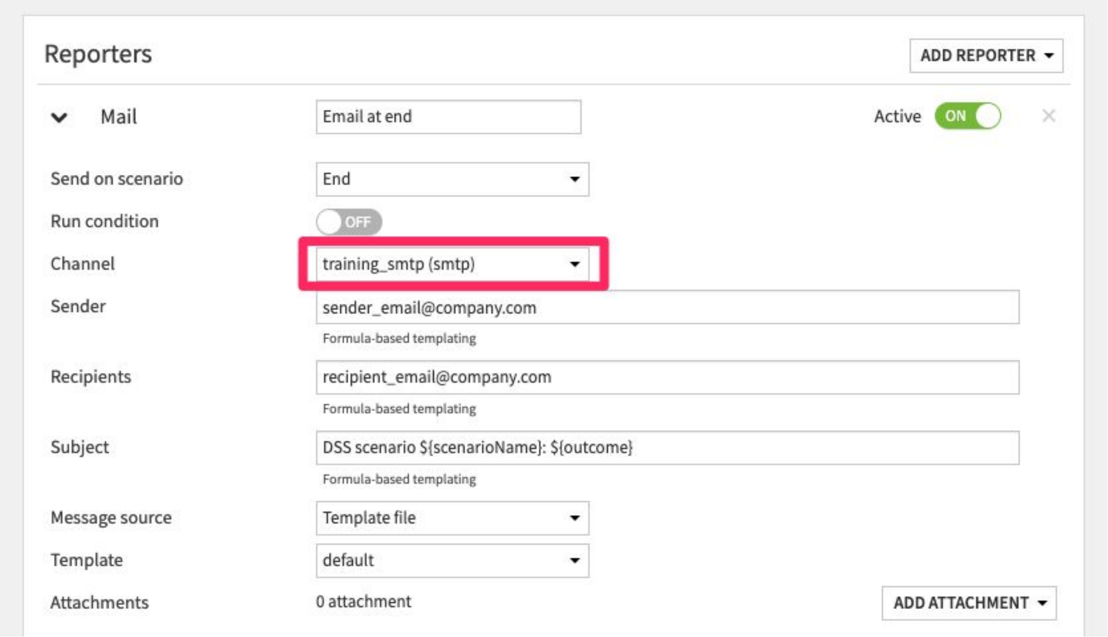
1. Add a Time-based Trigger:
   a. Scenario will run every day at 2AM





Create Scenario Reporter

2. Add a Reporter to email us details of the scenario

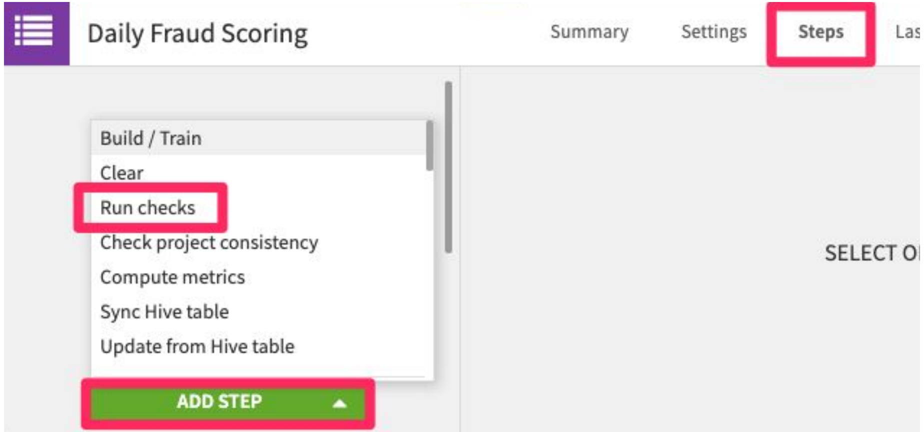


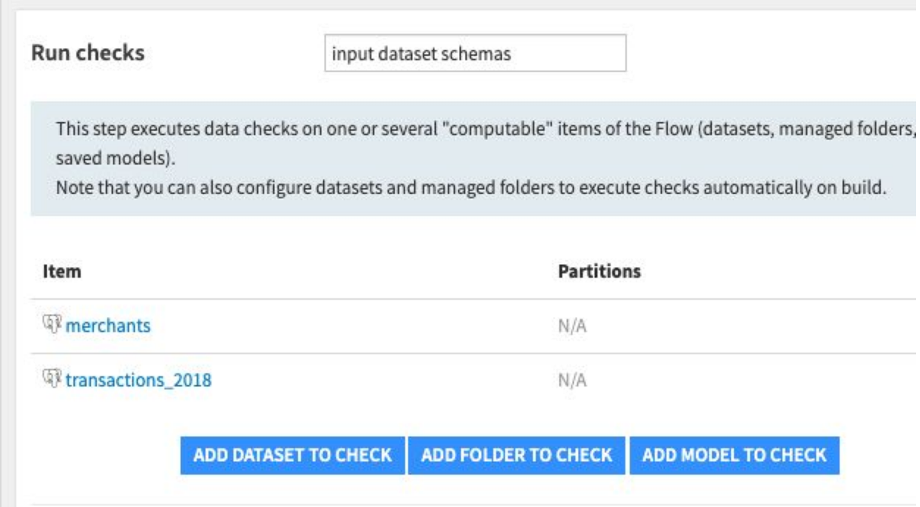
Create Scenario Steps

3. Execute Steps
    a. Run checks on ‘transactions_2018’ and ‘merchants’
        - Stop the scenario if the check Fails
    b. Build ‘transactions_joined_prepared’
    c. Run checks on ‘transactions_joined_prepared’
        - Stop the scenario if the check Fails
    d. Build ‘transactions_2018_clean_scored’

Create Scenario Steps

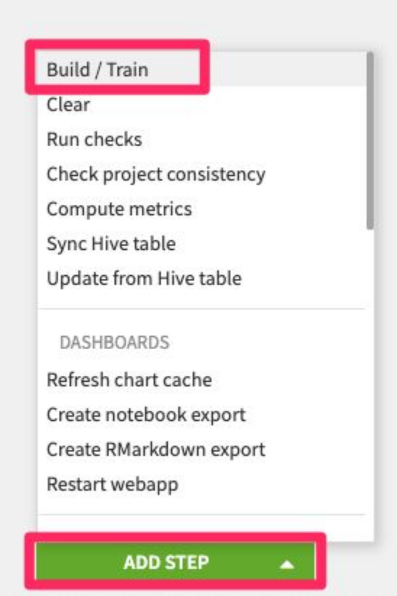
a. **Run checks** on ‘`transactions_2018`’ and ‘`merchants`’

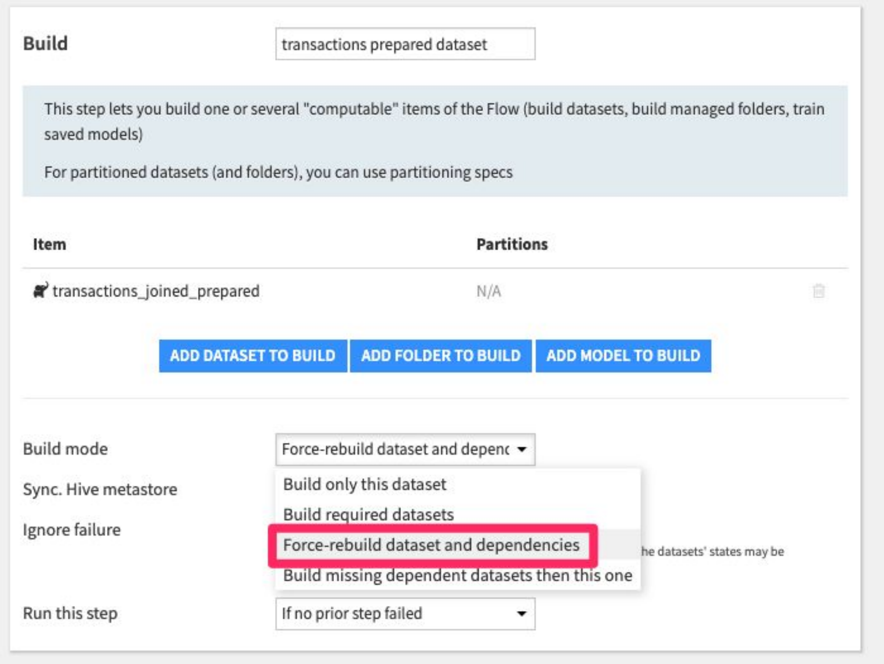




Create Scenario Steps

b. **Build** ‘`transactions_joined_prepared`’





Notice the Build mode

You can rebuild:

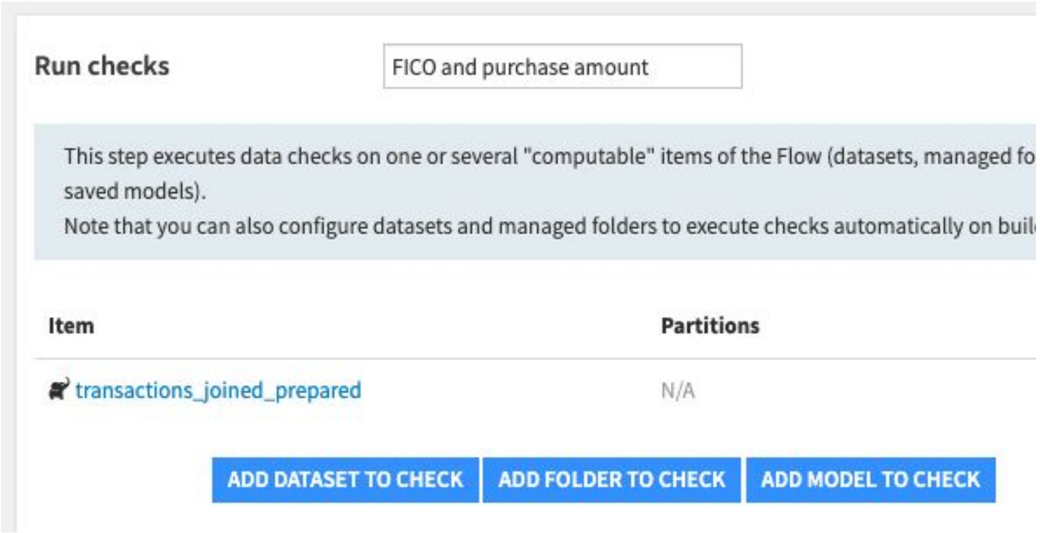
- the whole flow
- just this dataset
- required datasets (smart recursion)

Today we’ll choose “Force-rebuild” so that we can see all recipes re-run via the scenario

In reality, you’ll probably want “Build required datasets”

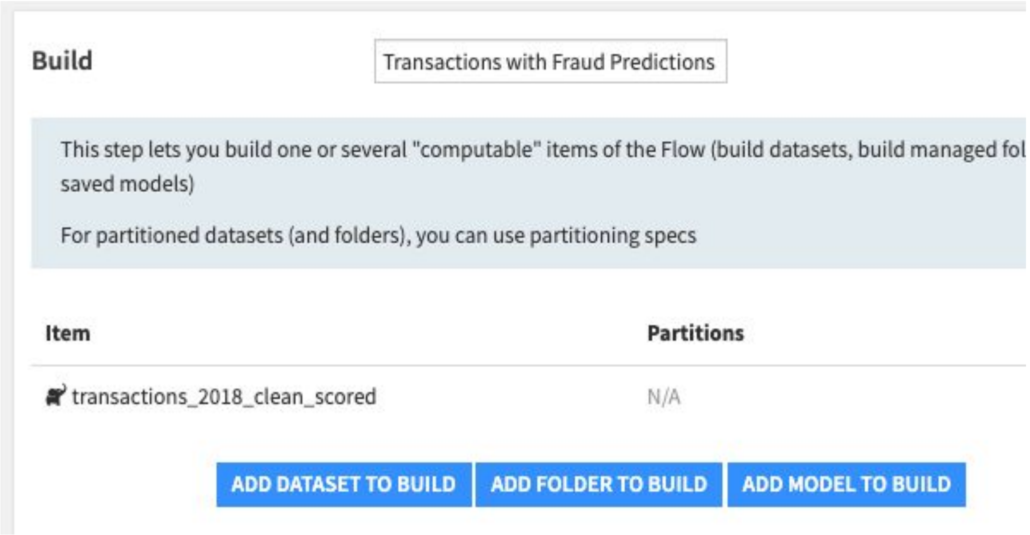
Create Scenario Steps

c. **Run checks** on ‘`transactions_joined_prepared`’



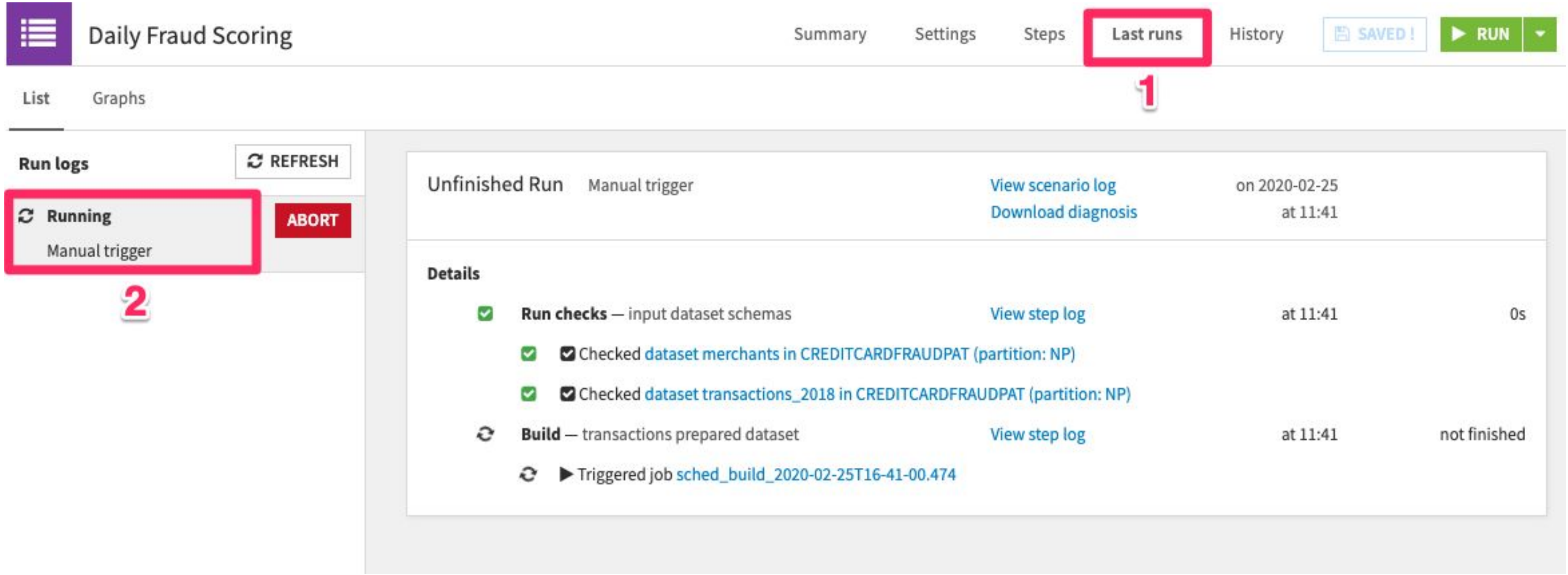
Create Scenario Steps

d. **Build** ‘`transactions_2018_clean_scored`’



Run your Scenario

Click Run, then check out the Last runs tab - monitor your scenario



Automation: Project Variables

Variables

DSS projects can access variables defined:

1. Across an entire DSS instance
    - Only admins can manage these
2. In a project
3. In a scenario
    - These variables last for the duration of a scenario run

Example Variables

Dates

“yesterday” : “06/13/2019”

“last_quarter_end_month_year” : “03-2019”

Decimals

“prev_year_valuation” : 1.456

Integers

“run_number” : 14

Lists

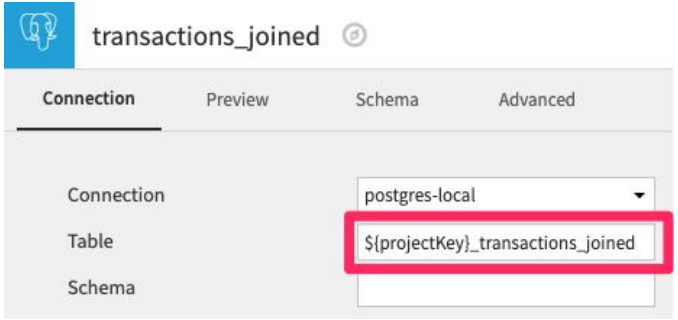
“boston_sports_teams” : [“celtics”, “red_sox”, “patriots”, “bruins”]

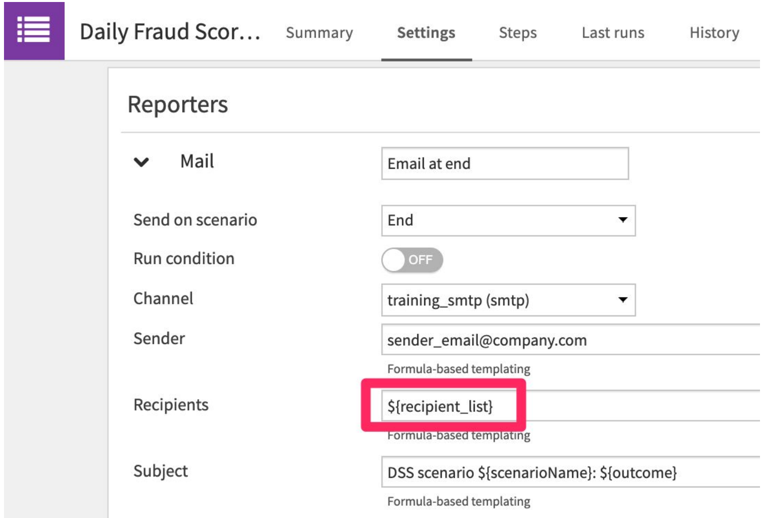
Be careful with lists. DSS returns the variable value as a string, so you’ll have to convert it back into a list

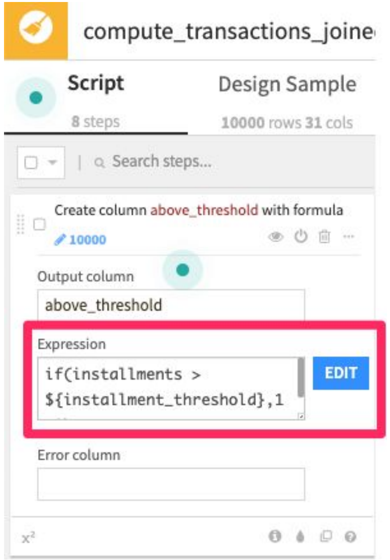
Accessing Variables

Variables can be accessed

- In any notebook / recipe / scenario / etc…
- Directly in configuration fields (Ex. dataset table name)
- By multiple users







Accessing Variables - Syntax

Prepare recipe Formula processor

```
if(installments > ${VARIABLE_NAME}, 1, 0)
```

SQL, Hive

```
SELECT * FROM table_name WHERE column = '${VARIABLE_NAME}'
```

Python

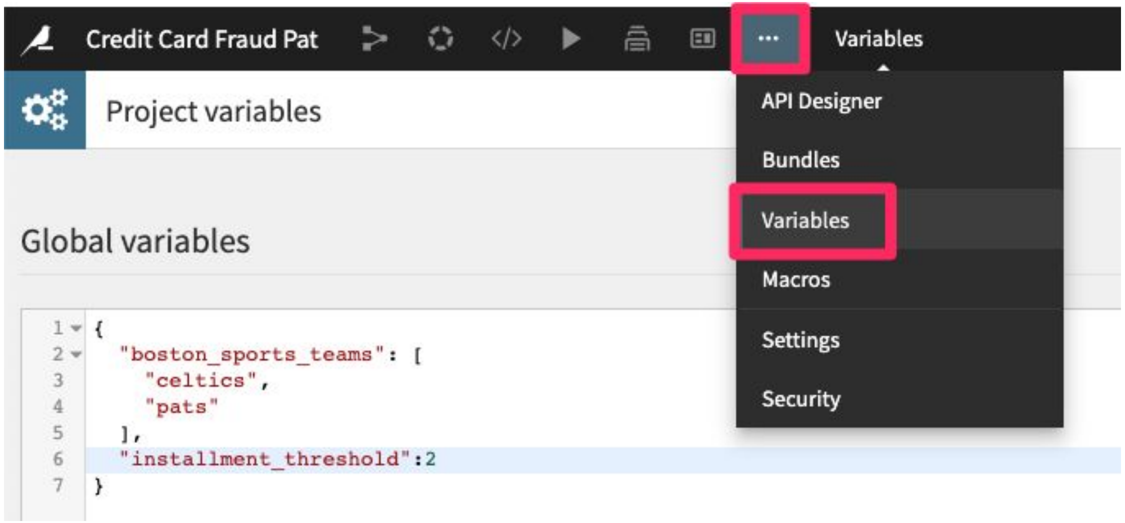
```
import dataiku
variable = dataiku.get_custom_variables()['VARIABLE_NAME']
```

R

```
library(dataiku)
variable <- dkuCustomVariable("VARIABLE_NAME")
```

Defining Variables

Set project variables in the settings of a project



Syntax

- Dictionary format
- Double quotes for all dictionary keys (variable names) and strings

Variables - Project Goal

After running the previous scenario to rebuild the flow with new data, we want to receive an email with the scored fraudulent transaction predictions just for the previous day.

Steps

- Define a project-level variable called "`previous_day`"
- Create a filter recipe after '`transactions_2018_clean_scored`' to filter for just transactions from the previous day
- Change our scenario to:
  - Update the "`previous_day`" variable
  - Rebuild the new dataset after the filter
  - Attach the filtered dataset to the email reporter

Variables - Project Goal

After running the previous scenario to rebuild the flow with new data, we want to receive an email with the scored fraudulent transaction predictions just for the previous day.

Variables - Project Steps

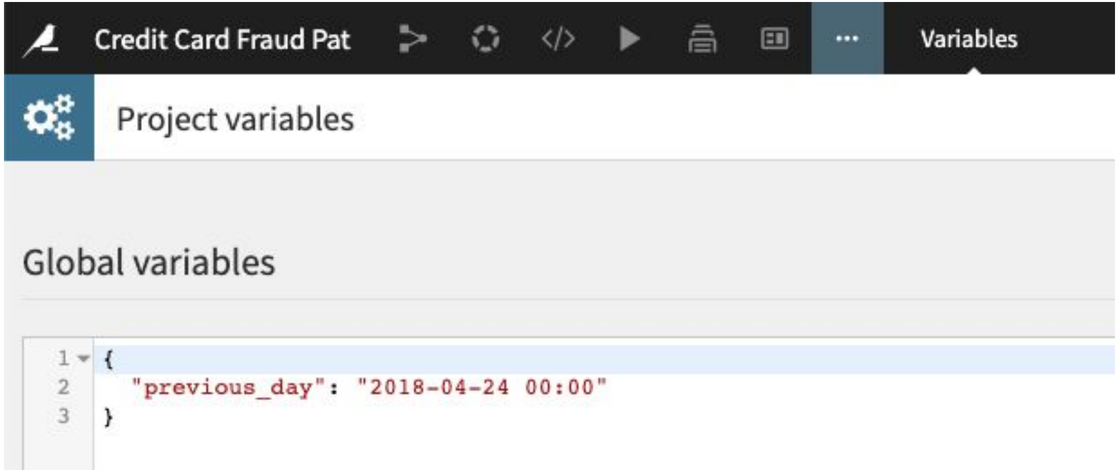
Steps

- Define a project-level variable called "`previous_day`"
- Create a filter recipe after '`transactions_2018_clean_scored`' to filter for just transactions from the previous day
- Change our scenario to:
  - Update the "`previous_day`" variable with a python step
  - Rebuild the new dataset after the filter
  - Attach the filtered dataset to the email reporter

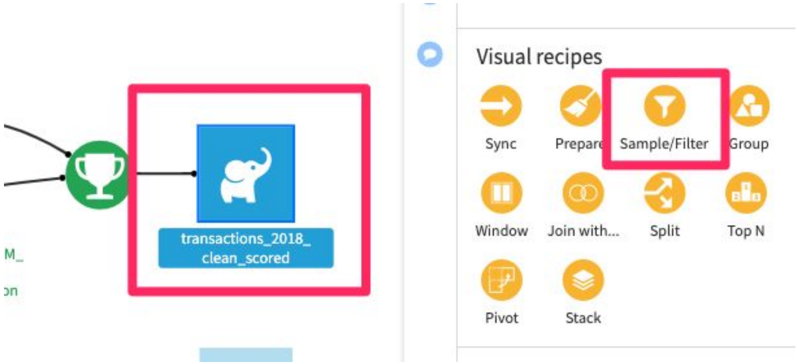
Create a Project Variable

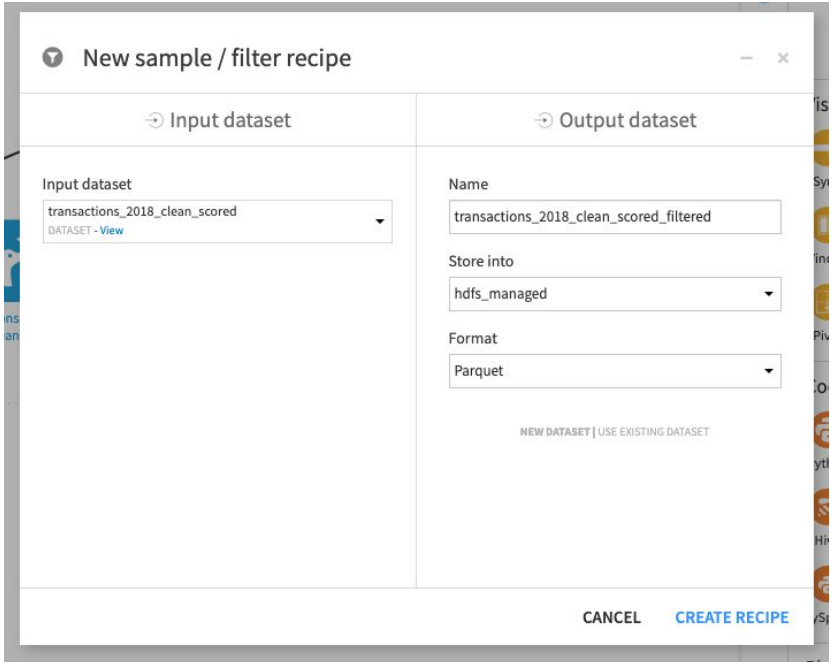
Start with this value:

```
"previous_day": "2018-04-24 00:00"
```

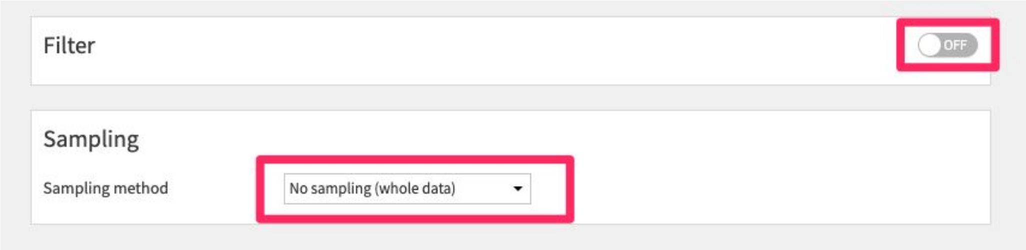


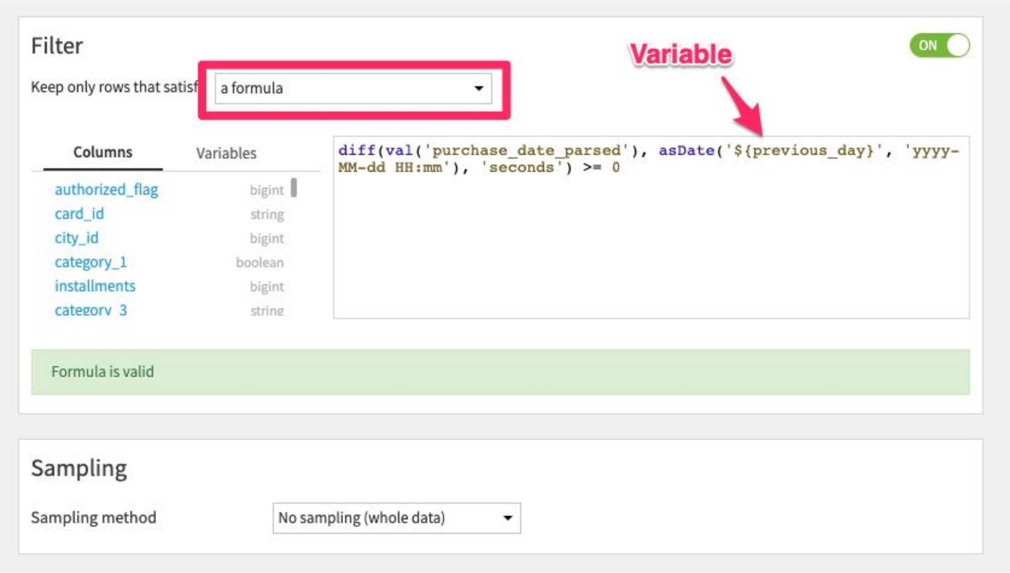
Create a New Filter Recipe





Use the Formula to Filter Rows on the Project Variable

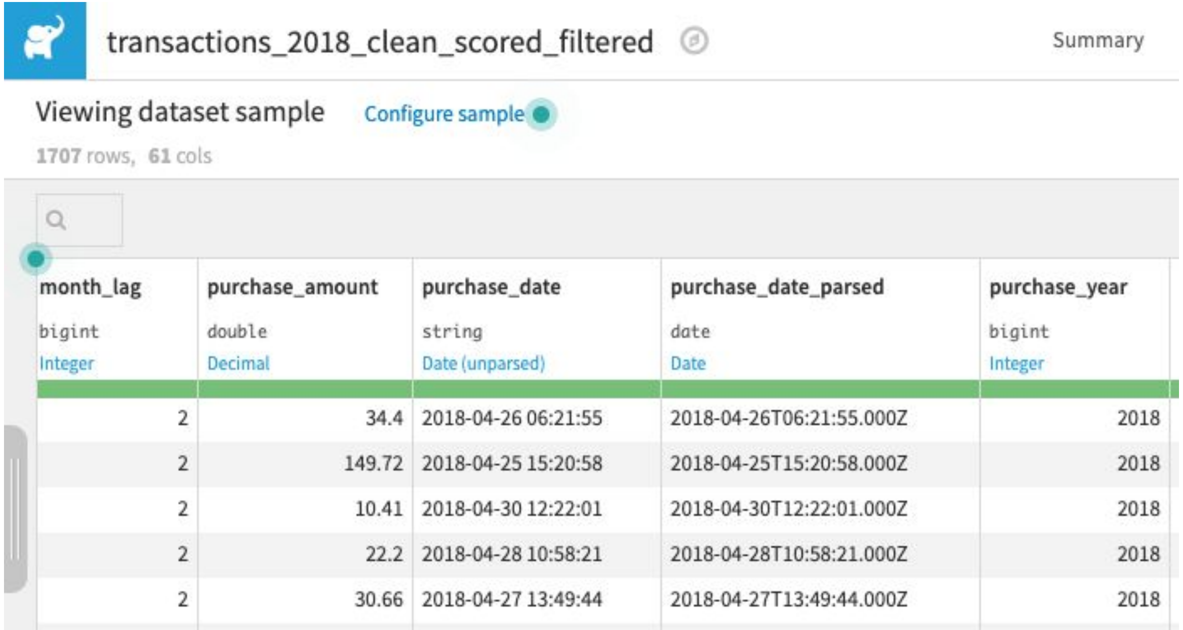




Formula text

```
diff(val('purchase_date_parsed'),
asDate('${previous_day}', 'yyyy-MM-dd
HH:mm'), 'seconds') >= 0
```

Run the Filter Recipe and Explore the Output Dataset



transactions_2018_clean_scored_filtered

Summary

Viewing dataset sample Configure sample

1707 rows, 61 cols

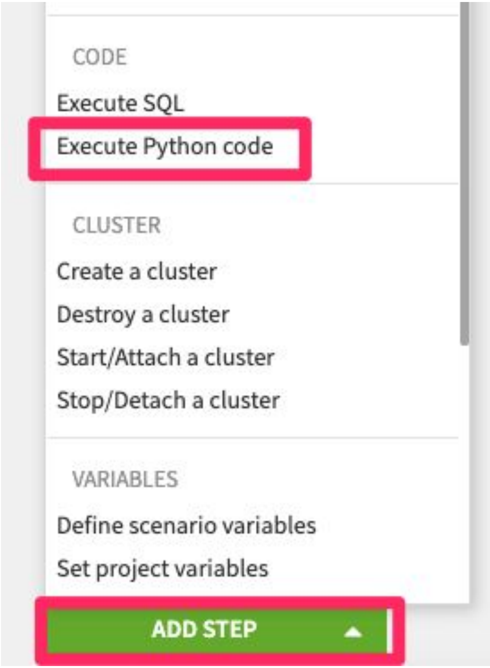
month_lag	purchase_amount	purchase_date	purchase_date_parsed	purchase_year
bigint	double	string	date	bigint
Integer	Decimal	Date (unparsed)	Date	Integer
2	34.4	2018-04-26 06:21:55	2018-04-26T06:21:55.000Z	2018
2	149.72	2018-04-25 15:20:58	2018-04-25T15:20:58.000Z	2018
2	10.41	2018-04-30 12:22:01	2018-04-30T12:22:01.000Z	2018
2	22.2	2018-04-28 10:58:21	2018-04-28T10:58:21.000Z	2018
2	30.66	2018-04-27 13:49:44	2018-04-27T13:49:44.000Z	2018

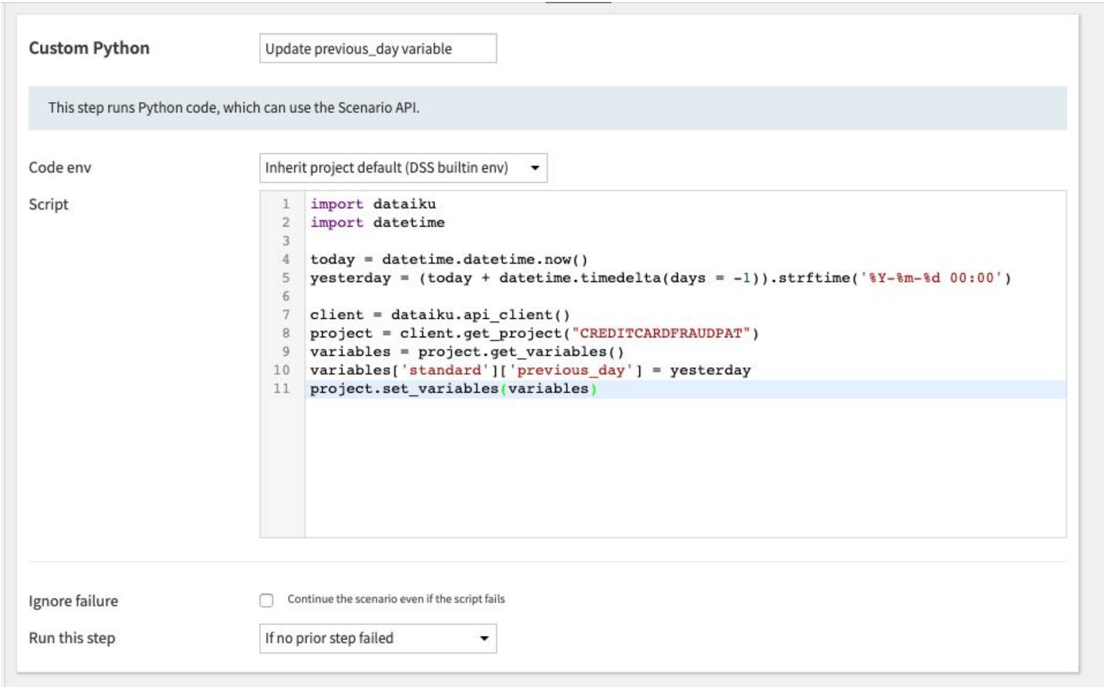
Notice all transactions are post-April 24, 2018 (our variable date)

Go to your previous Scenario

Add a new python step

(Code to copy/paste on next slide)





Scenario - Python Step

(Python code)

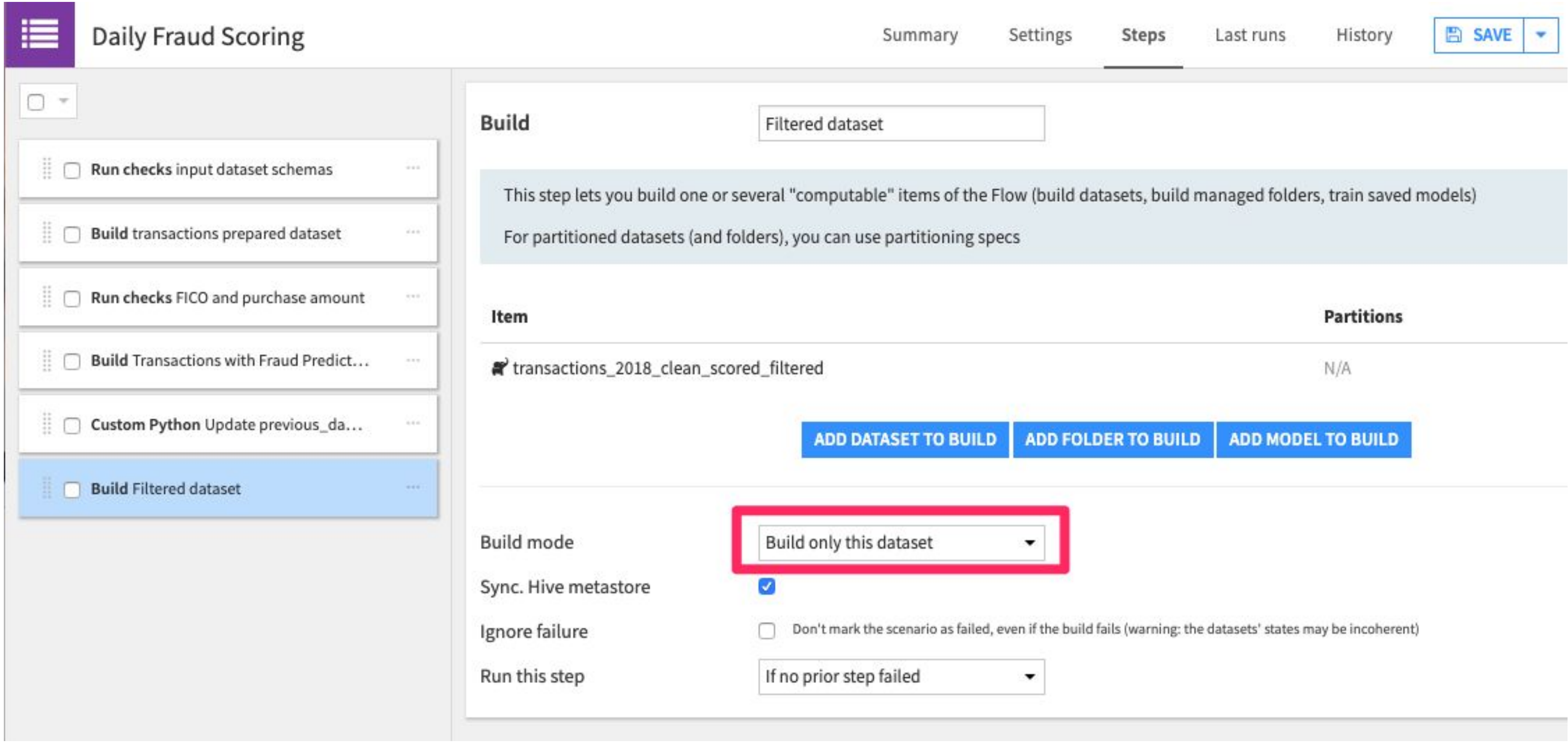
```
import dataiku
import datetime

today = datetime.datetime.now()
yesterday = (today + datetime.timedelta(days = -1)).strftime('%Y-%m-%d 00:00')

client = dataiku.api_client()
project = client.get_project("REPLACE_WITH_YOUR_PROJECT_KEY")
variables = project.get_variables()
variables['standard']['previous_day'] = yesterday
project.set_variables(variables)
```

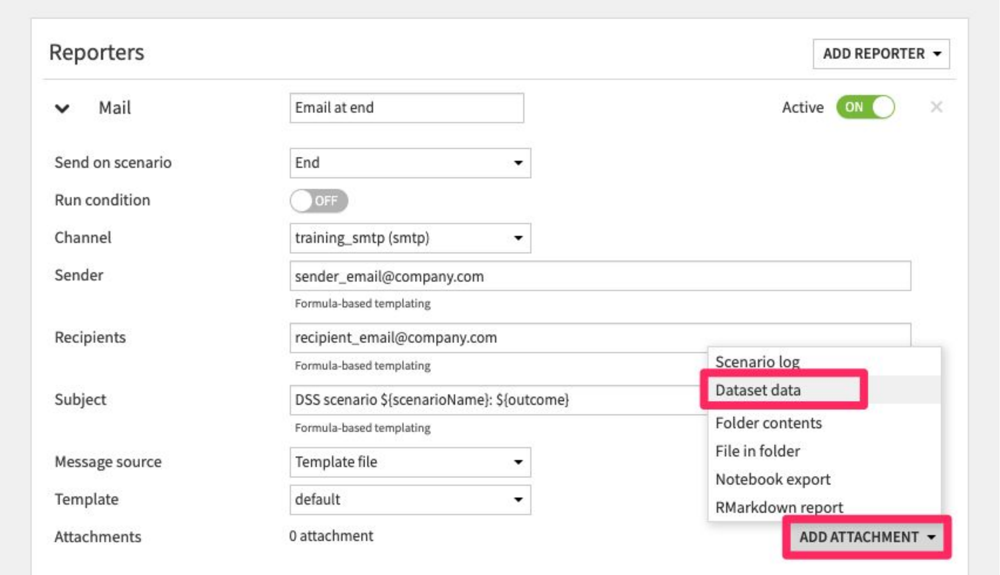
Scenario - Build the Filtered Dataset

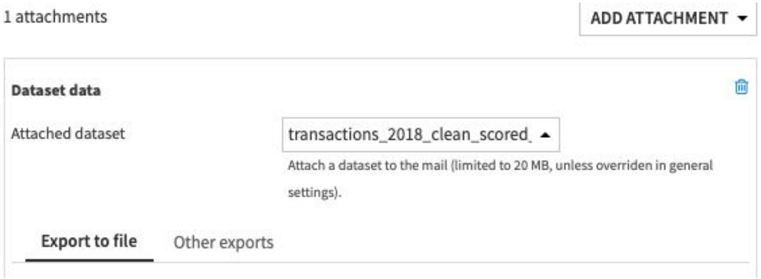
Add another step to build the filtered dataset



Attach Dataset to Email Reporter

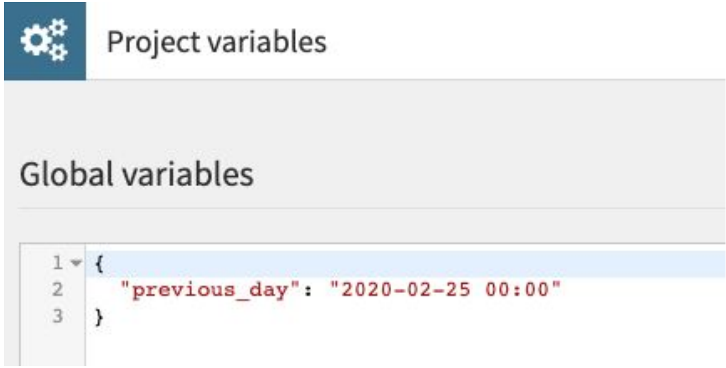
Attach ‘`transactions_2018_clean_scored_filtered`’ to the email reporter



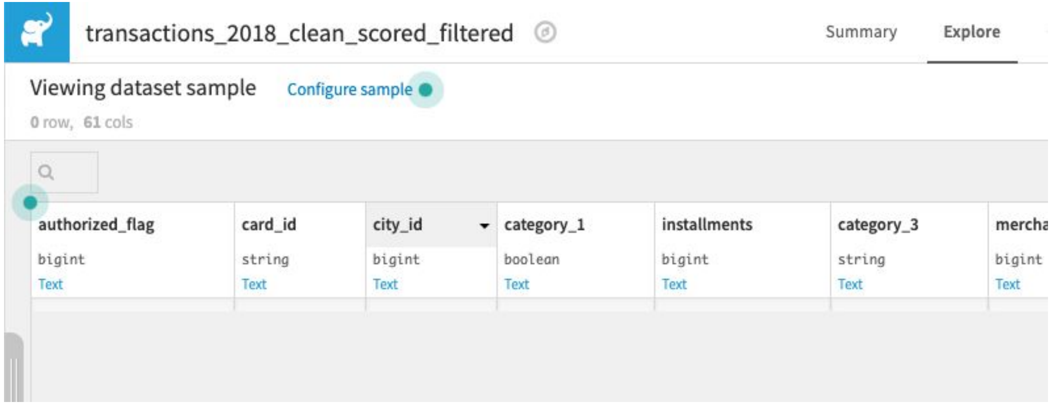


Check the Result

Check your project variable 'previous_day' - it should have changed to yesterday's date



Explore the new filtered dataset - it should be empty, as we have no transactions from yesterday in this database



Deployment to Production (Demo)

Deployment to Production - Batch Scoring Pipeline

Objectives :

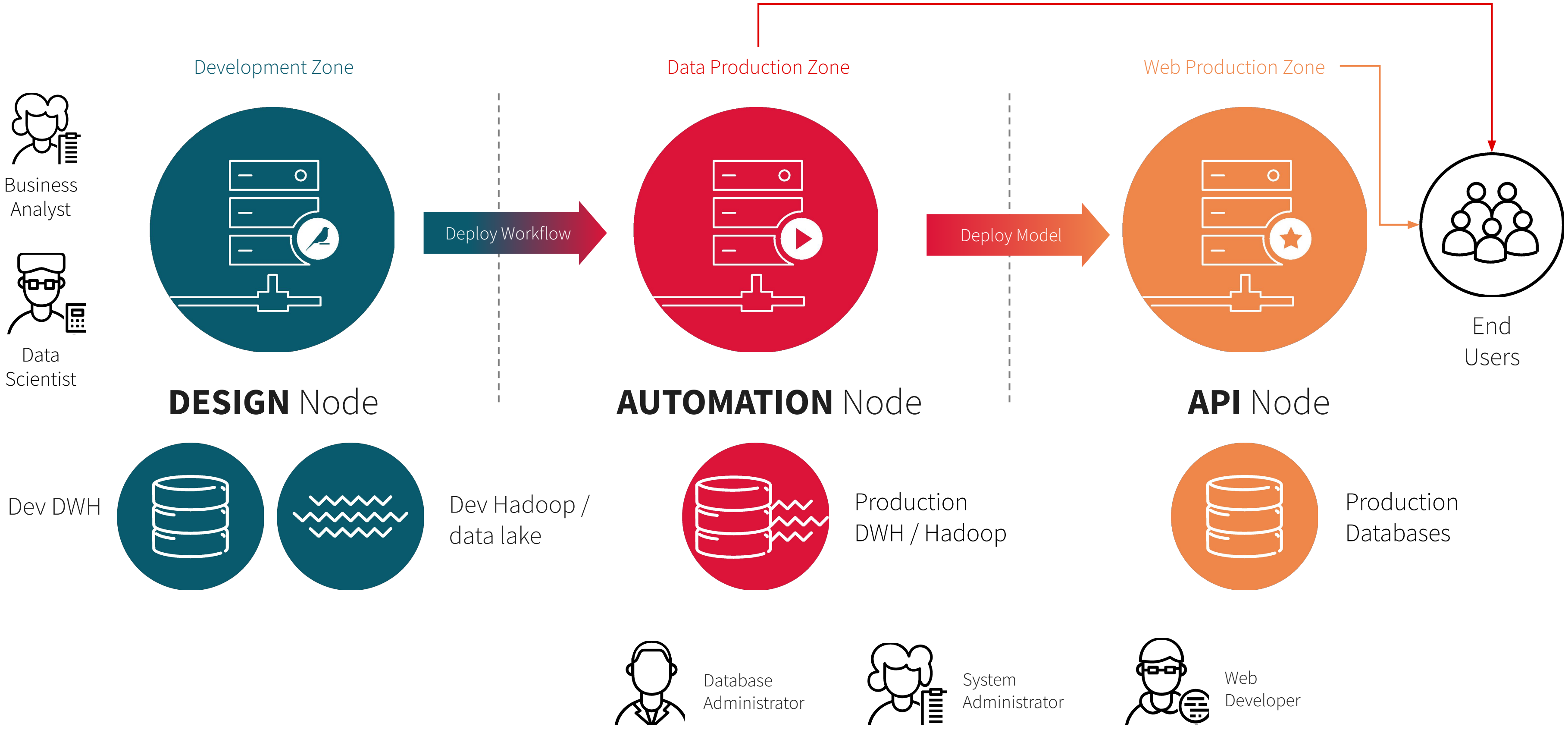
We want to have a safe environment where our scoring pipeline is not at risk of being altered by modifications in the flow.

We also want to be connected to our production databases.

Solution :

Deploy our project to the Automation Node.

DSS Nodes



Development Zone
Data Production Zone
Web Production Zone
Business Analyst
Data Scientist
Deploy Workflow
Deploy Model
End Users
DESIGN Node
AUTOMATION Node
API Node
Dev DWH
Dev Hadoop / data lake
Production DWH / Hadoop
Production Databases
Database Administrator
System Administrator
Web Developer

Deployment to Production - Batch Scoring Pipeline

Moving a project from a DSS Design Node to Automation Node:

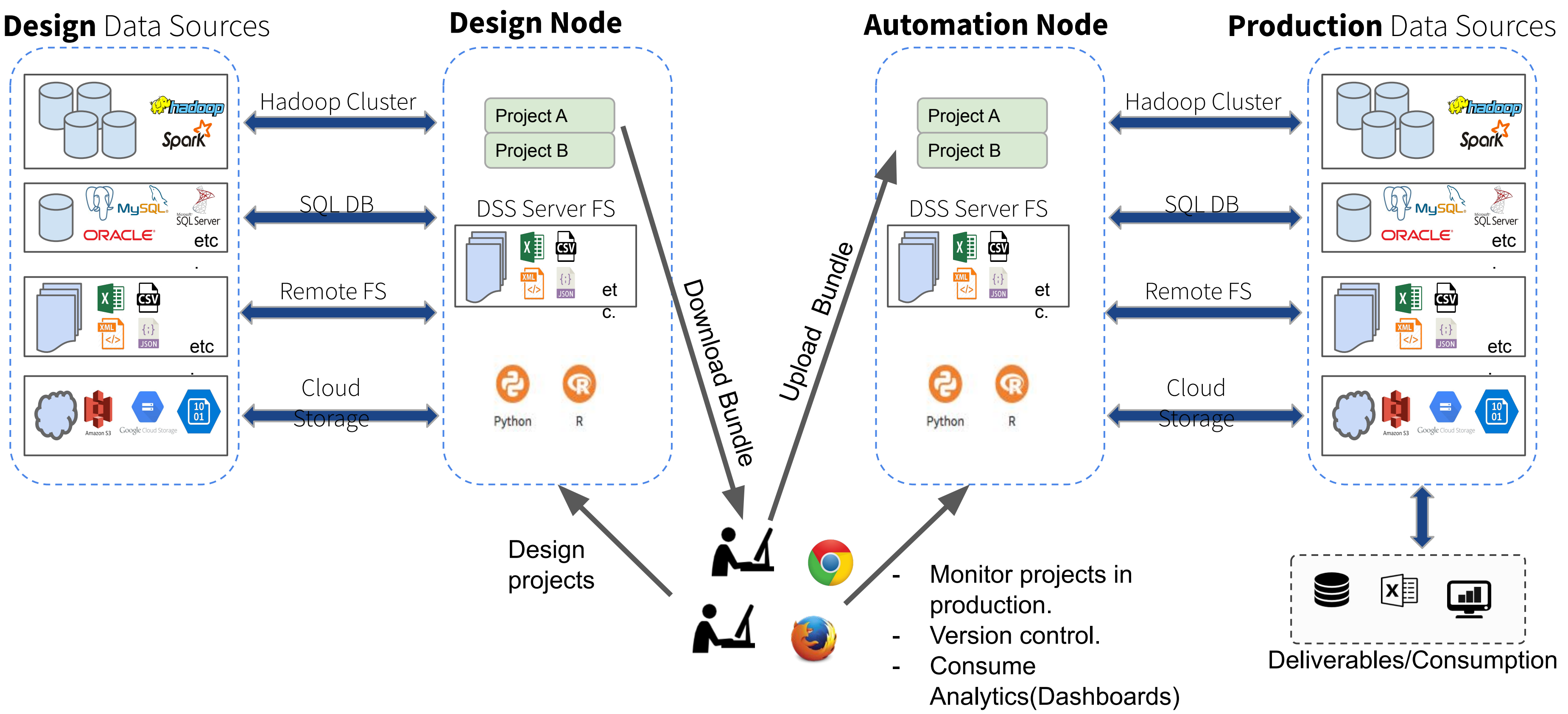
1. "Bundle" your project in the Design Node - this will create a zip file with the project's config
2. Download the bundle to your local machine
3. Upload the bundle to the Automation Node to create a new project (or update an existing one)

⚠️ This step may require dataset connection remapping

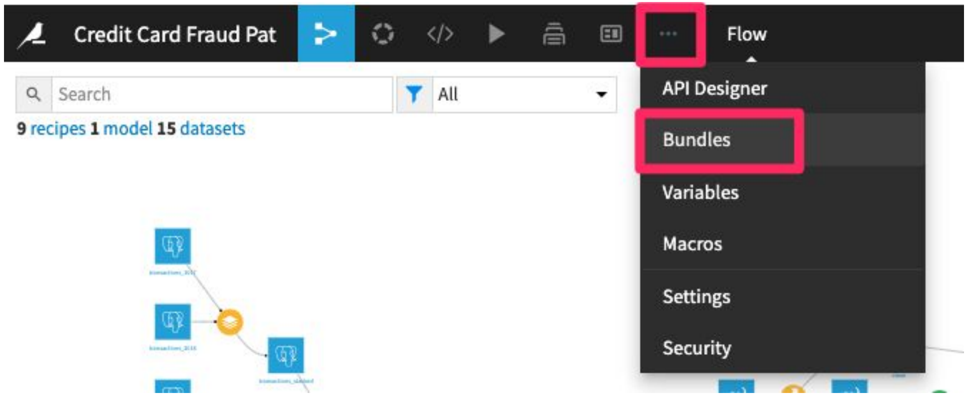
4. Activate the bundle on the on the Automation Node

Note: all of these steps can be automated using a DSS Macro or using the Public API

From Design Node to Automation Node



Create a Project Bundle



No bundles in this project

Bundles are versioned snapshots of your project for deployment in a DSS automation node.

Read the documentation to learn more.

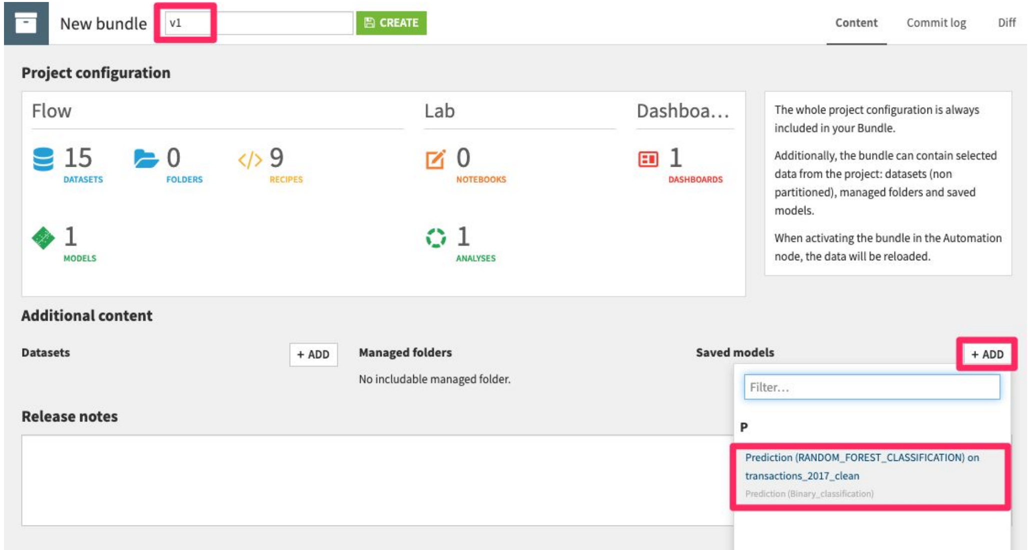
+ CREATE YOUR FIRST BUNDLE

Create a Project Bundle

By default, only the project metadata are bundled. As a result, all datasets will come empty and models will come untrained by default.

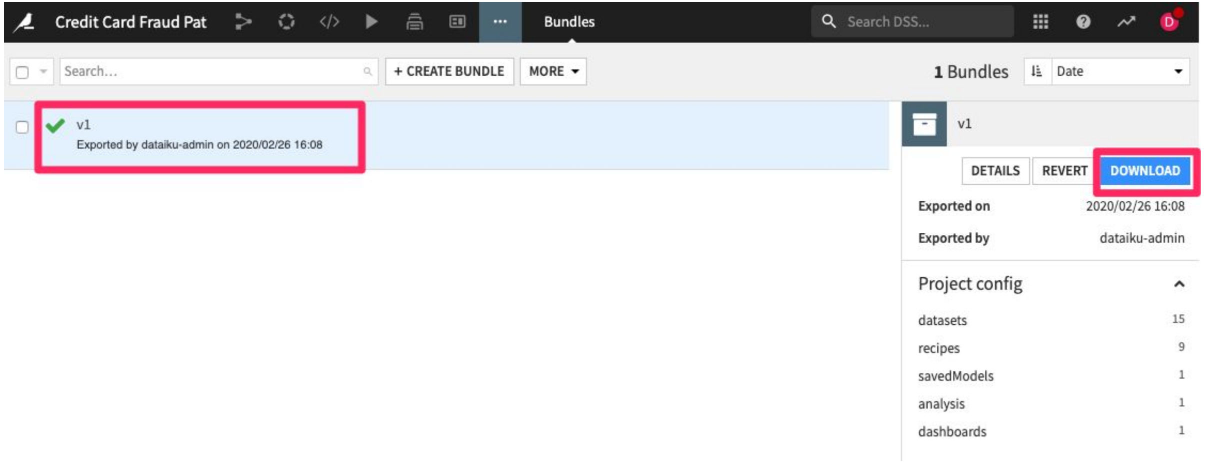
A good practice is to have the Automation Node connected to separate production data sources. Dataset connections can be remapped after uploading the bundle.

1. Name the bundle "v1"
2. Add your model to the bundle
3. Click Create



Download the Bundle

Select the bundle and download it



Upload the Bundle to the Automation Node

Go to the Automation Node

[https://dssX-**automation**-COMPANYNAME.training.dataiku.com](https://dssX-automation-COMPANYNAME.training.dataiku.com)

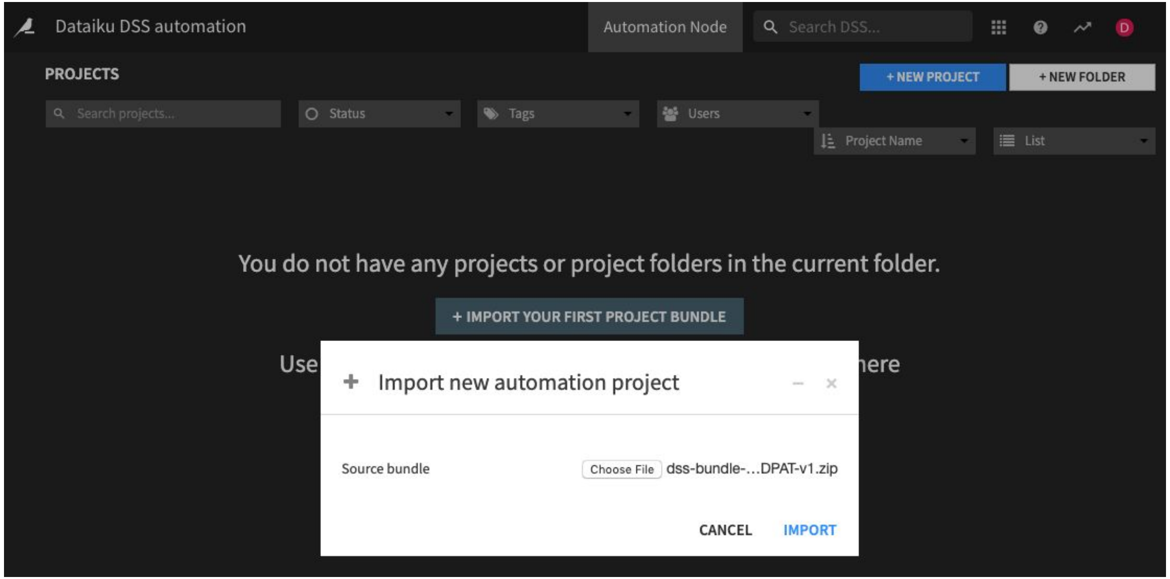
(replace the X with your instance number)

Username: userY

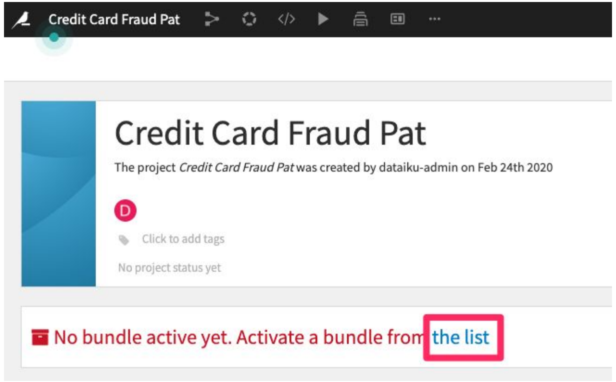
Password: PASSWORD_FROM_INSTANCE

Upload the Bundle to the Automation Node

Click Import Your First Project Bundle, choose the bundle file on your computer and click Import

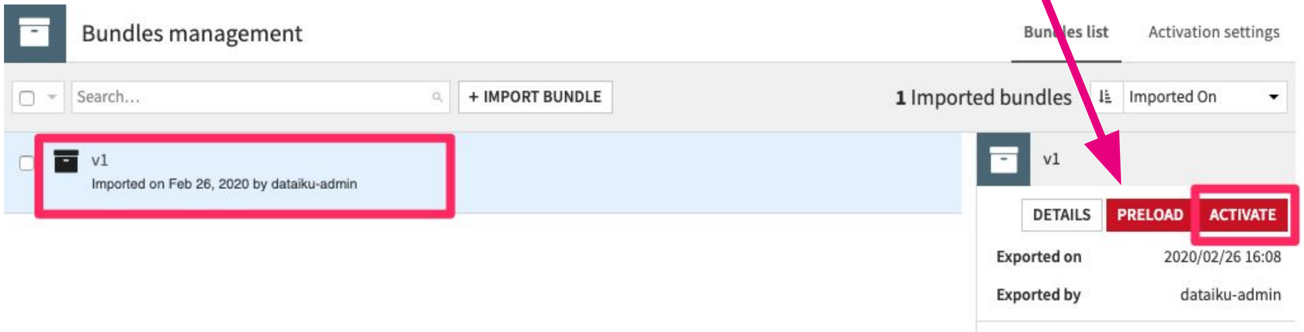


Activate the Bundle

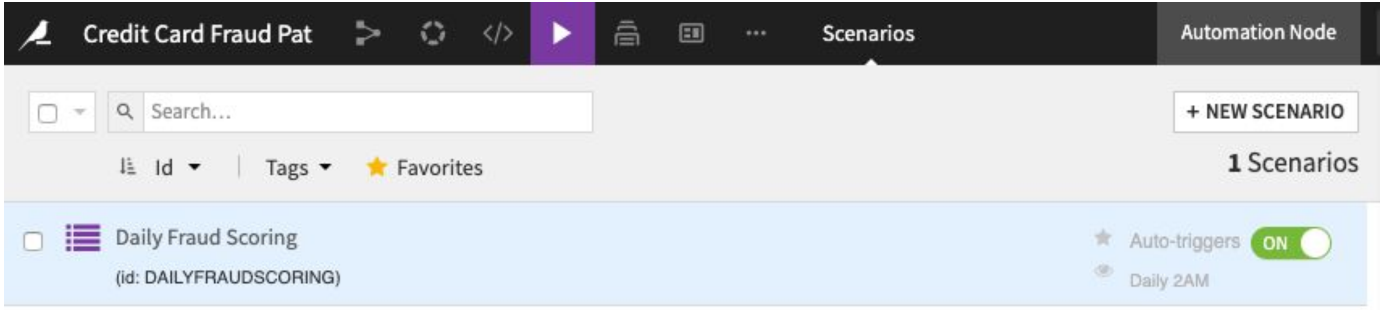


Use the "Preload" button for projects with python or R code environments.

- This will update and properly version the code environments on the Automation Node to reflect the latest from the Design Node



Finally, Activate your Scenario

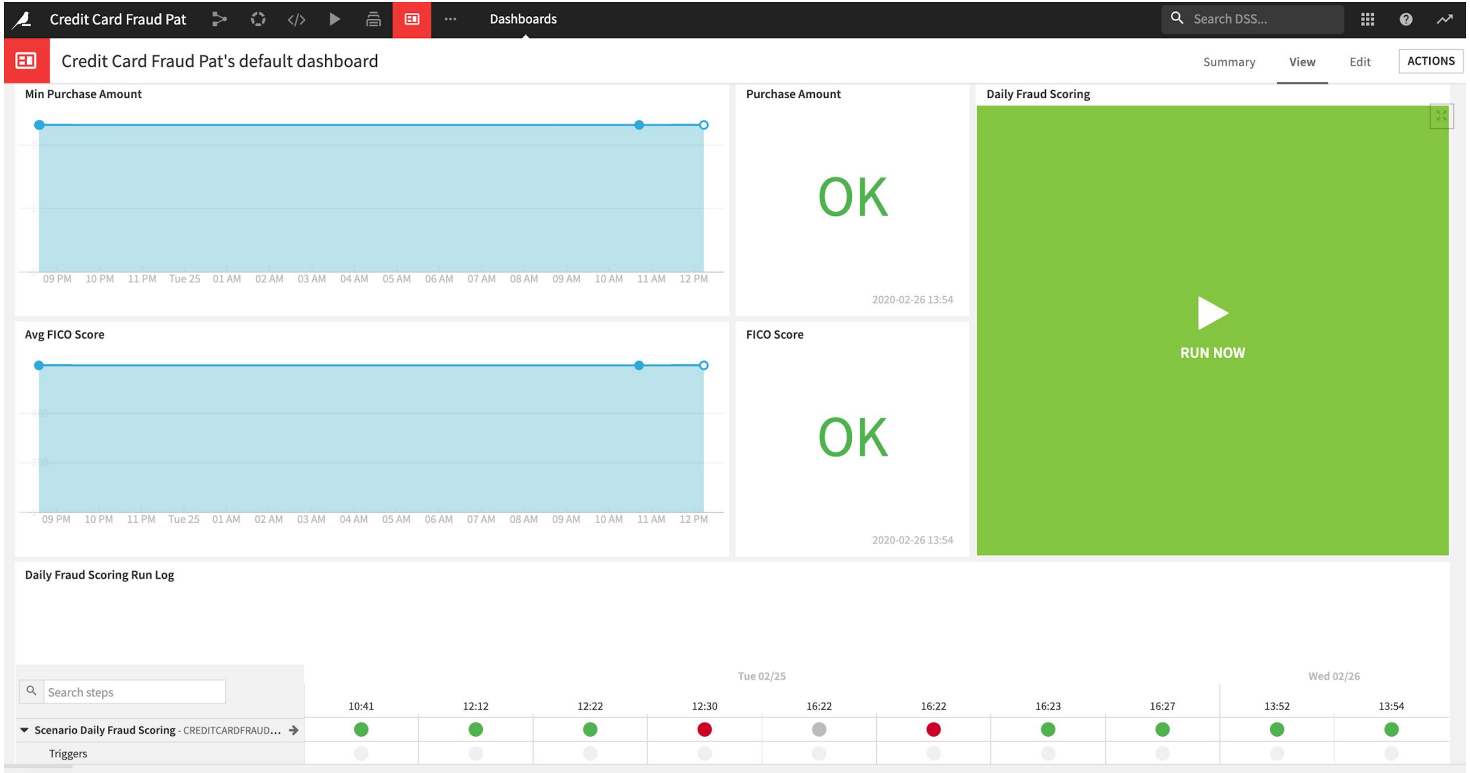


Dashboards to Monitor Scenarios

Create a new slide on your project dashboard with:

- The average FICO_score and minimum purchase_amount metrics from the ‘`transactions_joined_prepared`’ dataset
- Both data quality checks from the above metrics (also from the ‘`transactions_joined_prepared`’ dataset)
- A run button for your daily scoring scenario
    - Read-only dashboard users can use this
- A “last runs” log for your scenario

Dashboards to Monitor Scenarios



Automation Node

Thank you

Additional Resources

community.dataiku.com

- Public Q&A
- (like stack overflow)

doc.dataiku.com

- Extensive technical documentation about DSS

academy.dataiku.com

- Tutorials
- Videos

gallery.dataiku.com

- Sample projects