

**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

### New user

**General**

Login	new user
Type	LOCAL
Display name	
Email	
Groups	administrators, data_team
Profile	Data scientist

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

Show: By user ▾

A Admin

	Administrator	Sample business user	jordan	reader	user_A	user_B
<b>Global admin</b>	✓	✗	✓	✗	✗	✗
<b>Manage user-defined meanings</b>	✓	✗	✓	✗	✓	✗
<b>Create projects</b>	✓	✗	✓	✗	✓	✗
<b>Manage own code envs</b>	✓	✗	✓	✗	✗	✗
<b>Manage all code envs</b>	✓	✗	✓	✗	✗	✗
<b>Manage own cluster</b>	✓	✗	✓	✗	✗	✗
<b>Manage all clusters</b>	✓	✗	✓	✗	✗	✗
<b>Develop plugins</b>	✓	✗	✓	✗	✗	✗
<b>Edit lib folders</b>	✓	✗	✓	✗	✗	✗
<b>Create user connections</b>	✓	✗	✓	✗	✗	✗
<b>Write unsafe code</b>	✓	✗	✓	✗	✗	✗
<b>Create published API services</b>	✓	✗	✓	✗	✗	✗

API_TEST_PROJECT	<table border="1"> <tr> <td>RC</td><td>WC</td><td>RS</td></tr> <tr> <td>RD</td><td>WD</td><td>MD</td></tr> <tr> <td>DA</td><td>EE</td><td>AU</td></tr> </table>	RC	WC	RS	RD	WD	MD	DA	EE	AU	<table border="1"> <tr> <td>RC</td><td>WC</td><td>RS</td></tr> <tr> <td>RD</td><td>WD</td><td>MD</td></tr> <tr> <td>DA</td><td>EE</td><td>AU</td></tr> </table>	RC	WC	RS	RD	WD	MD	DA	EE	AU				
RC	WC	RS																						
RD	WD	MD																						
DA	EE	AU																						
RC	WC	RS																						
RD	WD	MD																						
DA	EE	AU																						

# Permission Model

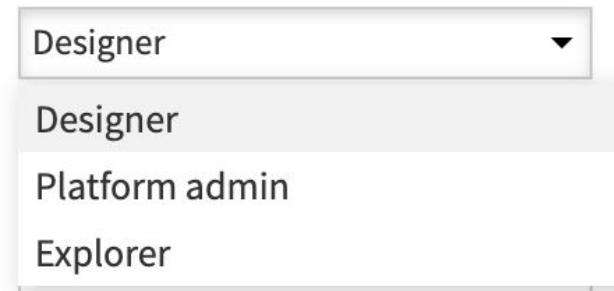
## User Profiles

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
<b>GLOBAL PERMISSIONS</b>	
Administrator	<input type="checkbox"/> Grants all permissions on all scopes
Manage user-defined meanings	<input type="checkbox"/> Can these users create/delete global UDMs?
Create projects	<input type="checkbox"/> Can these users create their own projects?
Write unsafe code	<input type="checkbox"/> 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	<input type="checkbox"/> Can these users manage code envs, even others'?
Manage own code envs	<input type="checkbox"/> Can these users create code envs, manage code envs they created or were given access to?
Manage all clusters	<input type="checkbox"/> Can these users manage clusters, even others'?
Manage own clusters	<input type="checkbox"/> Can these users create clusters, manage clusters they created or were given access to?
Develop plugins	<input type="checkbox"/> Can these users create and edit development plugins? Hostile users with this permission may circumvent the permissions system.
Edit lib folders	<input type="checkbox"/> 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	<input type="checkbox"/> Can these users create personal connections? Does not include FS connections
View indexed Hive connections	<input type="checkbox"/> Can these users view indexed Hive connections in the catalog?
Create published API services	<input type="checkbox"/> Can these users create a published API service?
<b>SAVE</b>	

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

Project owner : **Administrator** ▼ SAVE

Group name	Admin	Read project content	Write project content	Export datasets	Read dashboards	Write dashboards	Moderate dashboards	Run scenarios	Manage dashboard perm.	Manage exposed elts.	Manage dashboard users	
my_group	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<span>✖</span>						

Select a group... + GRANT ACCESS TO GROUP

ⓘTo manage groups go to [DSS global administration](#).

- 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 sources

OFF Dashboard-only users can access all objects

Add objects accessible by dashboard-only users

<input type="checkbox"/>	ACTIONS ▾	Search...	<input type="button" value="+ ADD AUTHORIZED OBJECT ▾"/>
<input type="checkbox"/>	Basketball_nScores_by_Exp	- view	

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

Exposed objects

Select flow objects to share and use on other projects

Search...

▼  Basketball\_EXP\_TM  on 1 project  
 API TEST PROJECT (API\_TEST\_PROJECT)

+ ADD PROJECT

Additional dashboard users

ACTIONS ▾    
 user\_A 

# 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

### Containerized execution

Default exec config

None

Choose a specific container exec configuration

engagement\_x\_k8s

### Code envs

#### Default environment for Python recipes

Use DSS builtin Python env  The default code environment in this project is DSS's builtin environment

Environment  non-managed-python

Prevent override by recipes  Recipes in this project are forced to use the code environment defined here, and no override is possible

#### Default environment for R recipes

Use DSS builtin R env  The default code environment in this project is DSS's builtin environment

Prevent override by recipes  Recipes in this project are forced to use the code environment defined here, and no override is possible

### Cluster selection

Use DSS global cluster settings

Cluster

test\_cluster

Use a cluster i

Default cluster

If the field abc

# 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.

### USAGE PARAMS

Allow write	<input checked="" type="checkbox"/> Can DSS write to the datasets from this connection?
Allow managed datasets	<input checked="" type="checkbox"/> Can new managed datasets be created on this connection?
Allow managed folders	<input checked="" type="checkbox"/> Can new managed folders be created on this connection?
Max nb. of activities	<input type="text" value="0"/> Limit the number of concurrent activities involving

### SECURITY SETTINGS

See [the documentation](#) for more information.

Freely usable by	<input type="radio"/> Every analyst <input checked="" type="radio"/> Selected groups
	Who can create new datasets in this connection, and more generally "browse" this connection.
↳ usable by groups	<input type="text" value="administrators, data_team, reade"/> ▾
Details readable by	<input checked="" type="radio"/> Nobody <input type="radio"/> Every analyst <input type="radio"/> Selected groups

### CREDENTIALS

See [the documentation](#) for more information.

Credentials mode

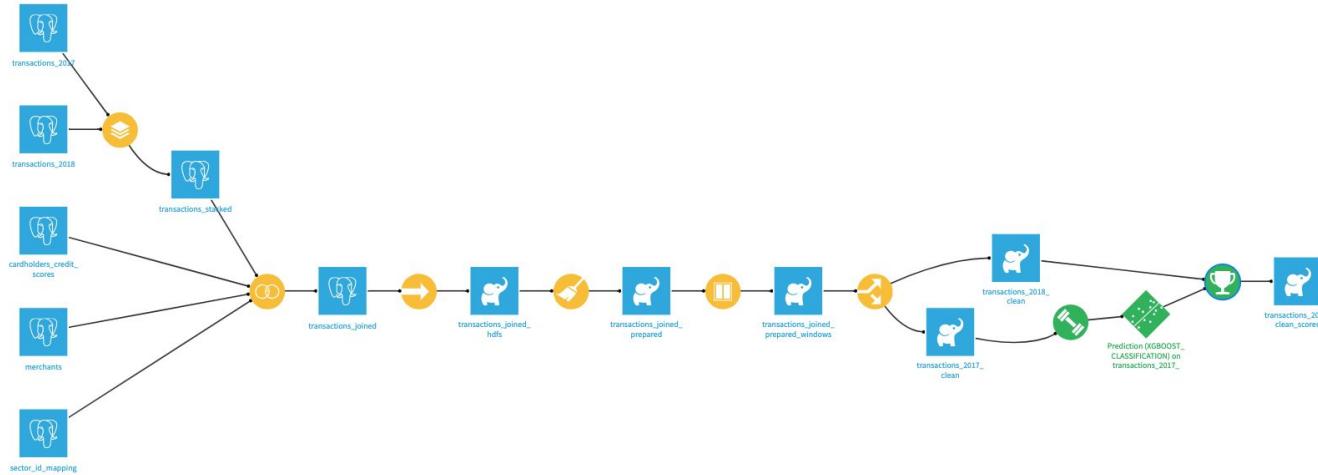
Per user	▼
----------	---

# Credit Card Transactions

# Initial Project

## Background

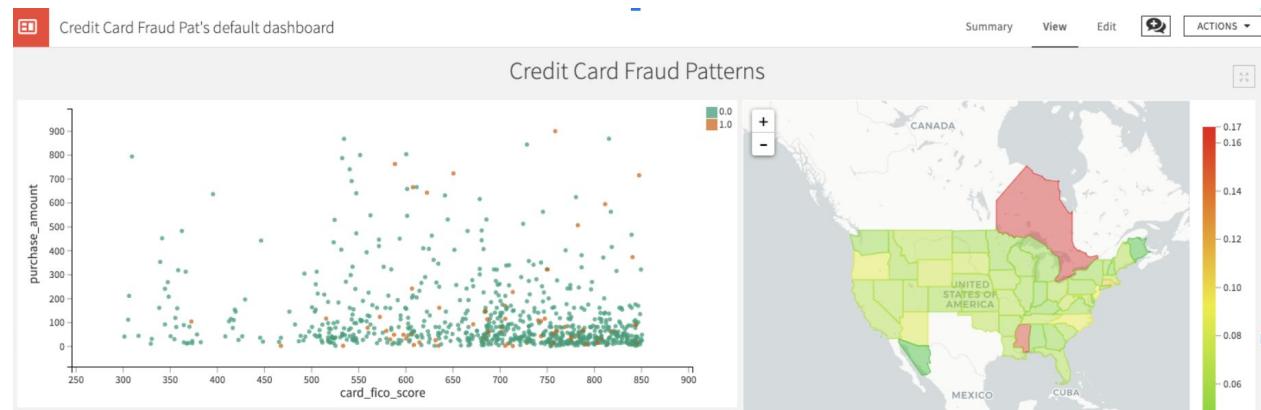
During the Discovery training, we created a pipeline and a model to predict which credit card transactions are fraudulent



# Automation in DSS

## Goal

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



# Login

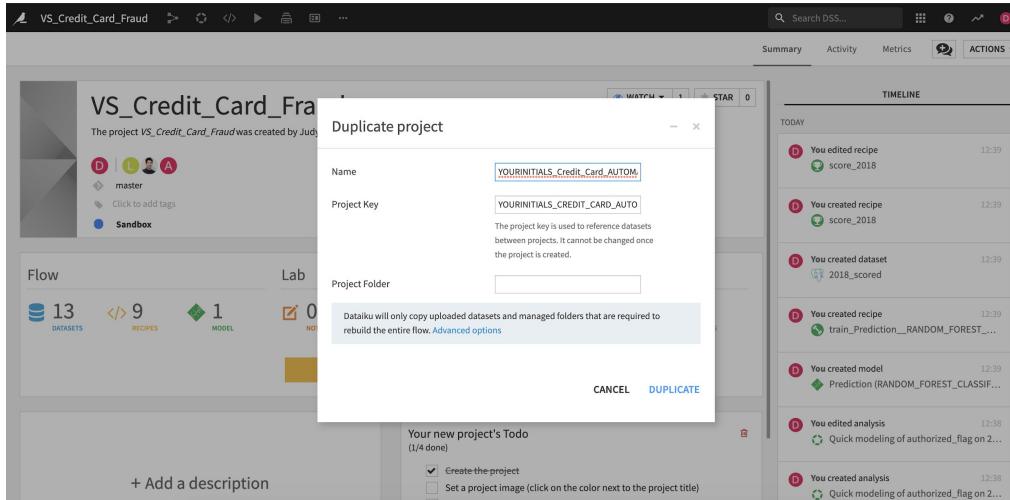
# Import the Project

# Copy an existing Project into DSS

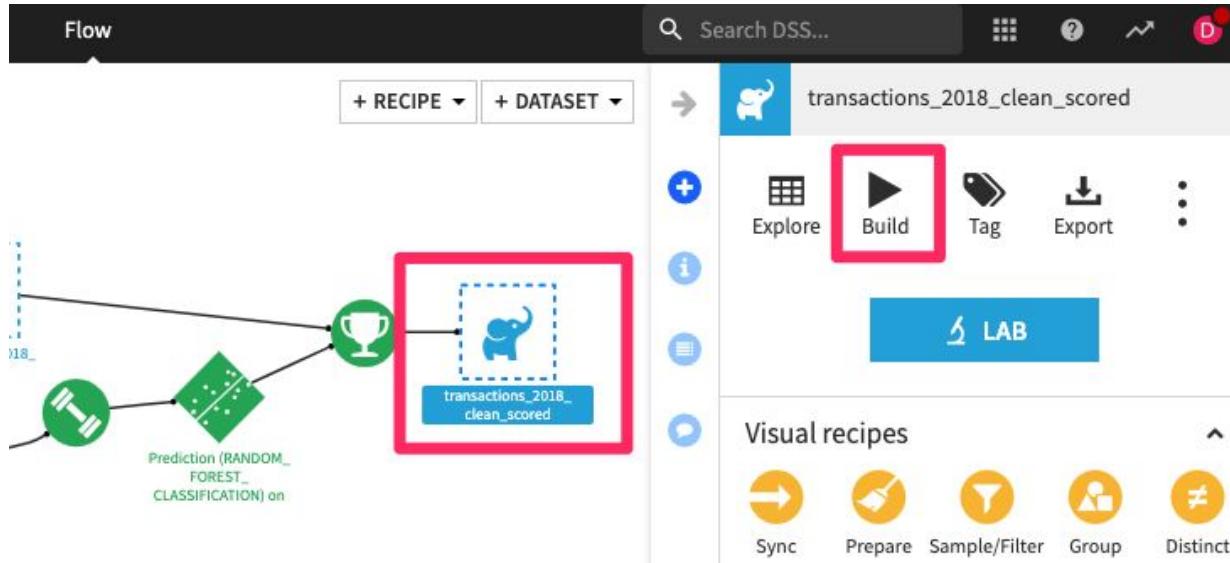
You can Duplicate an existing project from Dataiku DSS

[https://dss0-design-scb.training2.dataiku.com/projects/VS\\_CREDIT\\_CARD\\_FRAUD/](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 :

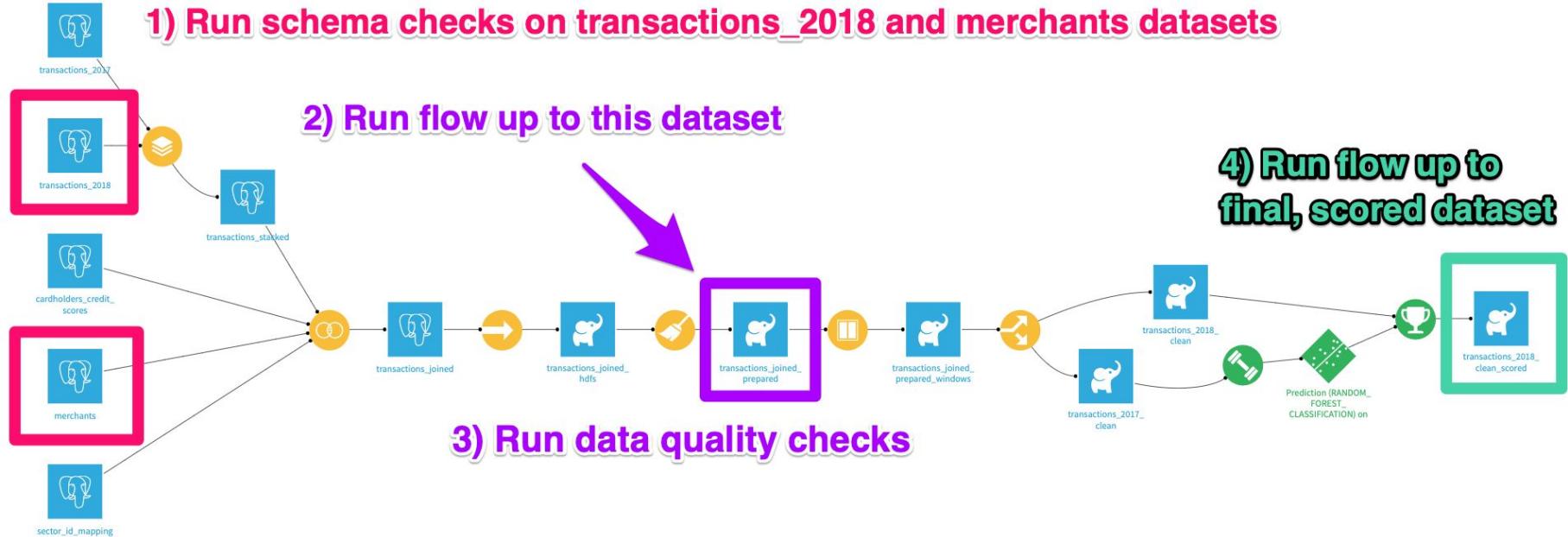
Add important metrics and checks on key datasets in the flow.

# Our Automated Workflow

Updated daily with new transactions (by some outside system)



# 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)

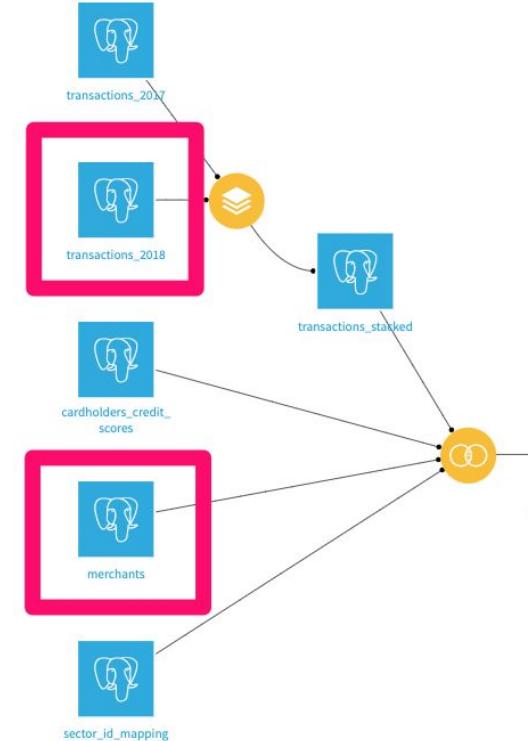
- Column count
- Record Count

## Checks (transactions\_2018)

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

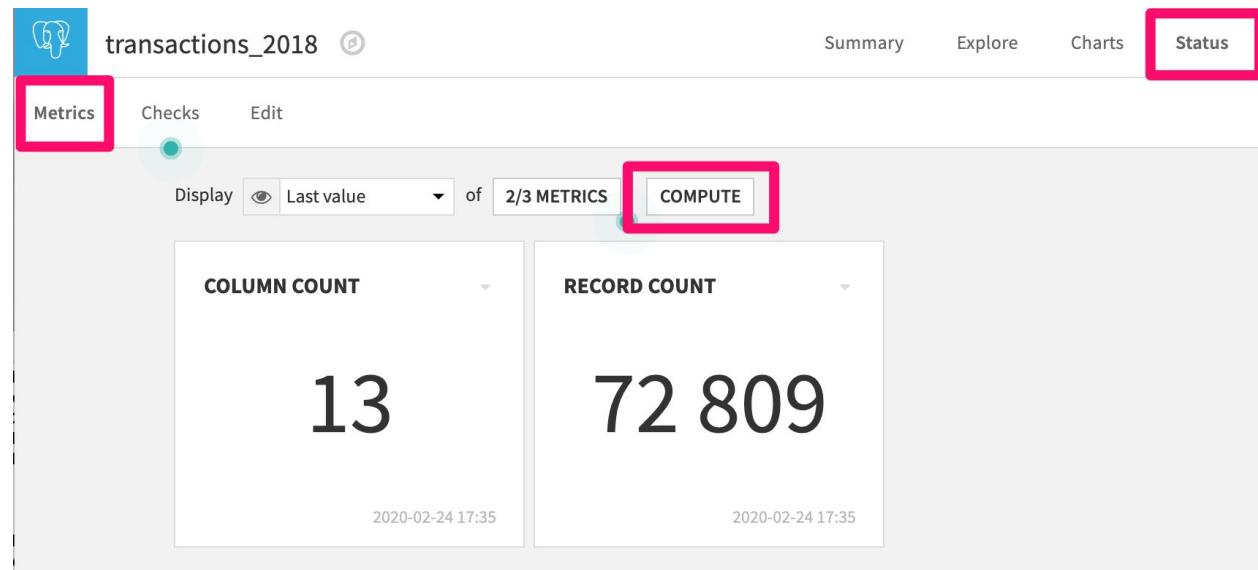
## Checks (merchants)

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



# Create Metrics - 'transactions\_2018'

Column count and record count are added by default

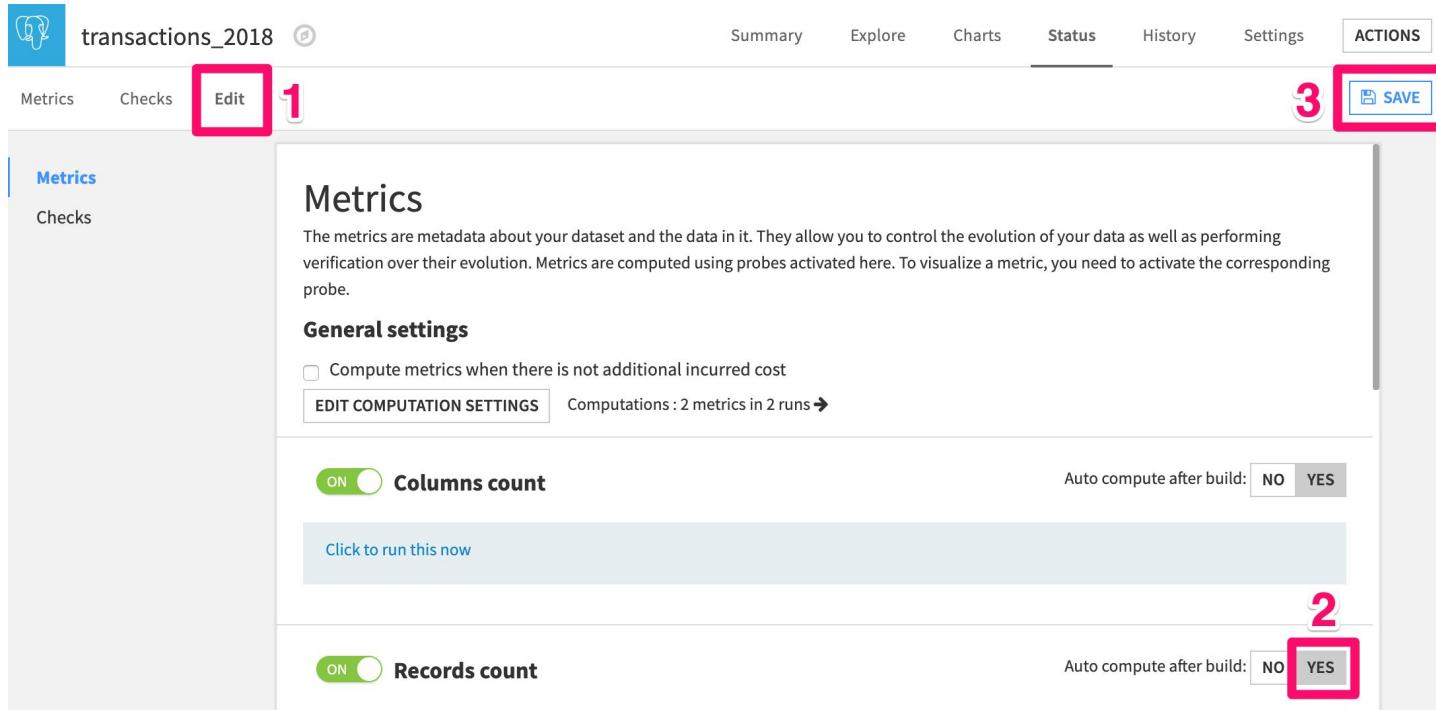


The screenshot shows the 'transactions\_2018' metrics page in the data iku interface. The 'Metrics' tab is active, and the 'Status' tab is highlighted with a red box. A 'COMPUTE' button is also highlighted with a red box. The interface displays two cards: 'COLUMN COUNT' (13) and 'RECORD COUNT' (72 809). The 'Status' tab includes a 'Summary', 'Explore', and 'Charts' option.

Display	Last value	of	2/3 METRICS	COMPUTE
COLUMN COUNT	13			
RECORD COUNT	72 809			

# Create Metrics - 'transactions\_2018'

Enable metric computation to happen automatically whenever the dataset is rebuilt



The screenshot shows the 'transactions\_2018' dataset in the data iku interface. The top navigation bar includes Summary, Explore, Charts, Status, History, Settings, and ACTIONS. Below the navigation is a sub-navigation bar with Metrics, Checks, and Edit, where 'Edit' is highlighted with a red box and the number '1'. The main content area is titled 'Metrics' and contains a descriptive paragraph about metrics. Under 'General settings', there is a checkbox for 'Compute metrics when there is not additional incurred cost' and a 'EDIT COMPUTATION SETTINGS' button. Two metrics are listed: 'Columns count' (ON) and 'Records count' (ON). Each metric has an 'Auto compute after build:' switch; for 'Columns count', it is set to 'NO' (grayed out), and for 'Records count', it is set to 'YES' (highlighted with a red box and the number '2'). A large red box highlights the 'SAVE' button in the top right corner.

transactions\_2018

Summary Explore Charts Status History Settings ACTIONS

Metrics Checks Edit 1

3 SAVE

**Metrics**

The metrics are metadata about your dataset and the data in it. They allow you to control the evolution of your data as well as performing verification over their evolution. Metrics are computed using probes activated here. To visualize a metric, you need to activate the corresponding probe.

**General settings**

Compute metrics when there is not additional incurred cost

**EDIT COMPUTATION SETTINGS** Computations : 2 metrics in 2 runs →

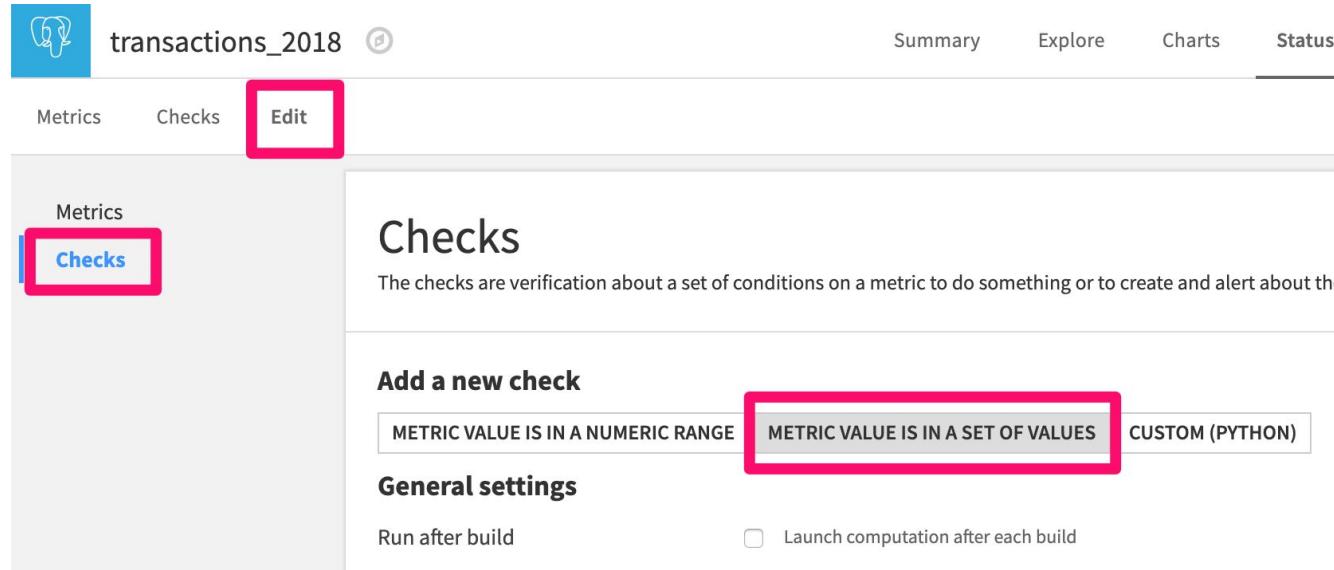
**ON Columns count** Auto compute after build: NO YES

Click to run this now

**ON Records count** Auto compute after build: NO YES

# Create Metric Checks - 'transactions\_2018'

Add a new check: column count is 13



The screenshot shows the dataiku interface for the dataset 'transactions\_2018'. The top navigation bar includes icons for Metrics, Checks, and Edit, with 'Edit' highlighted by a red box. The main content area has a sidebar with 'Metrics' and 'Checks' tabs, where 'Checks' is also highlighted by a red box. The main panel displays the 'Checks' section, which is described as a way to verify conditions on a metric. It features an 'Add a new check' button and three options: 'METRIC VALUE IS IN A NUMERIC RANGE', 'METRIC VALUE IS IN A SET OF VALUES' (which is highlighted by a red box), and 'CUSTOM (PYTHON)'. Below this, there's a 'General settings' section with a checkbox for 'Run after build' and an unchecked checkbox for 'Launch computation after each build'.

# Create Metric Checks - 'transactions\_2018'

Add a new check: **column count is 13**

## Checks

The checks are verification about a set of conditions on a metric to do something or to create and

**Columns 13**

(valueSet) 

Metric

Column count



Values

13



 **CHECK**

# Create Metric Checks - 'transactions\_2018'

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

**Over 10k records** (numericRange) 

Metric to check **Record count** 

Minimum  10000

Soft minimum  The check warns but doesn't fail if the value is below this soft minimum

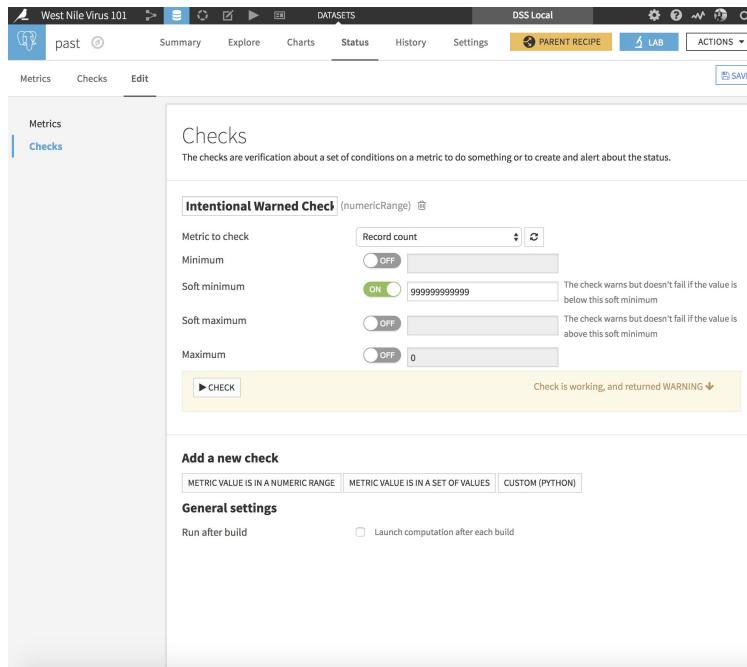
Soft maximum  The check warns but doesn't fail if the value is above this soft maximum

Maximum  0

**► CHECK** Check is working, and returned OK 

# Hard vs. Soft Min/Max for Checks

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



West Nile Virus 101

Datasets DSS Local Actions

past Summary Explore Charts Status History Settings Parent Recipe Lab

Metrics Checks Edit Save

**Checks**

The checks are verification about a set of conditions on a metric to do something or to create and alert about the status.

**Intentional Warned Check** (numericRange)

Metric to check: Record count

Minimum: OFF

Soft minimum: ON (9999999999) The check warns but doesn't fail if the value is below this soft minimum

Soft maximum: OFF The check warns but doesn't fail if the value is above this soft minimum

Maximum: OFF 0

**► CHECK** Check is working, and returned WARNING

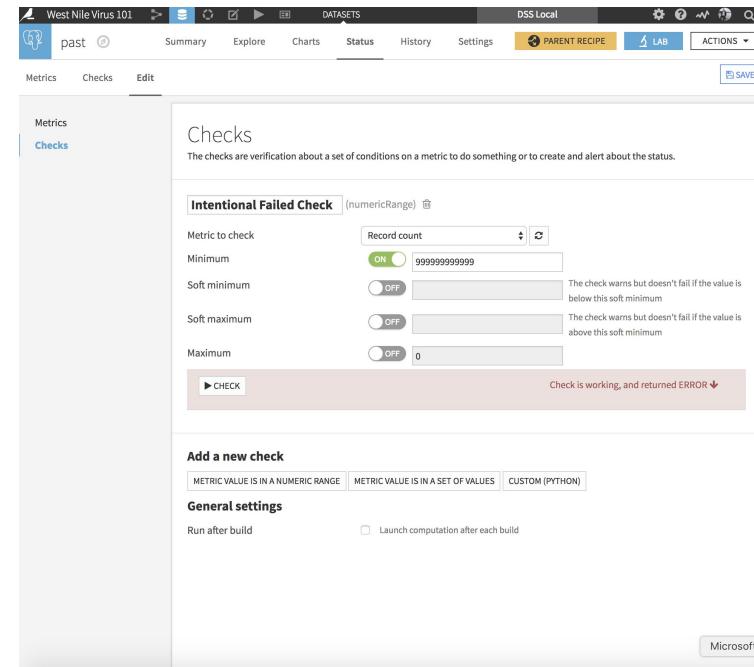
Add a new check

METRIC VALUE IS IN A NUMERIC RANGE METRIC VALUE IS IN A SET OF VALUES CUSTOM (PYTHON)

**General settings**

Run after build  Launch computation after each build

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



West Nile Virus 101

Datasets DSS Local Actions

past Summary Explore Charts Status History Settings Parent Recipe Lab

Metrics Checks Edit Save

**Checks**

The checks are verification about a set of conditions on a metric to do something or to create and alert about the status.

**Intentional Failed Check** (numericRange)

Metric to check: Record count

Minimum: ON (9999999999) The check warns but doesn't fail if the value is below this soft minimum

Soft minimum: OFF

Soft maximum: OFF The check warns but doesn't fail if the value is above this soft minimum

Maximum: OFF 0

**► CHECK** Check is working, and returned ERROR

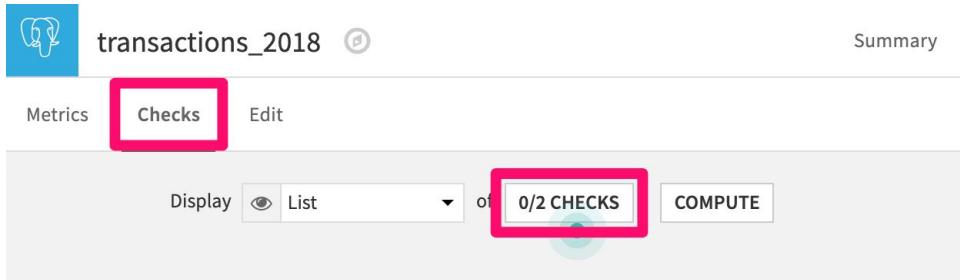
Add a new check

METRIC VALUE IS IN A NUMERIC RANGE METRIC VALUE IS IN A SET OF VALUES CUSTOM (PYTHON)

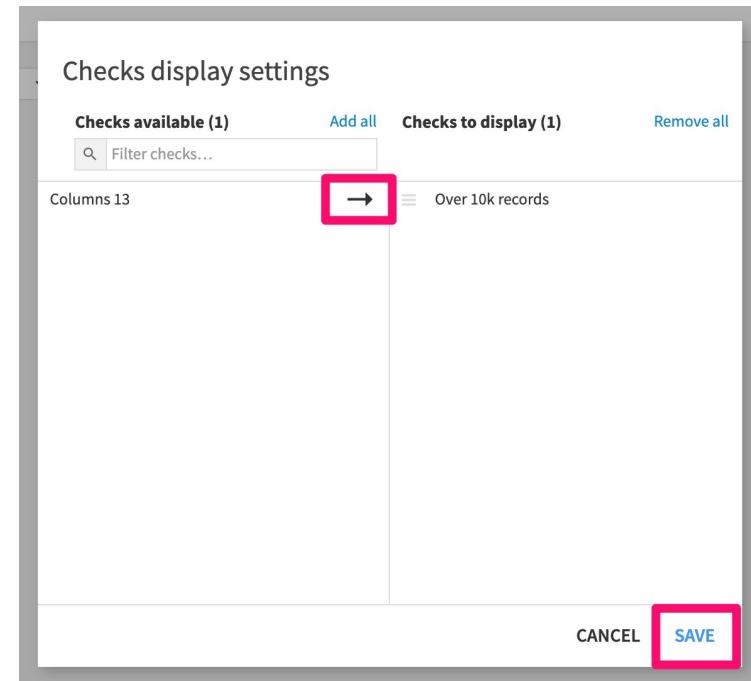
**General settings**

Run after build  Launch computation after each build

# Add Checks to the home screen - 'transactions\_2018'



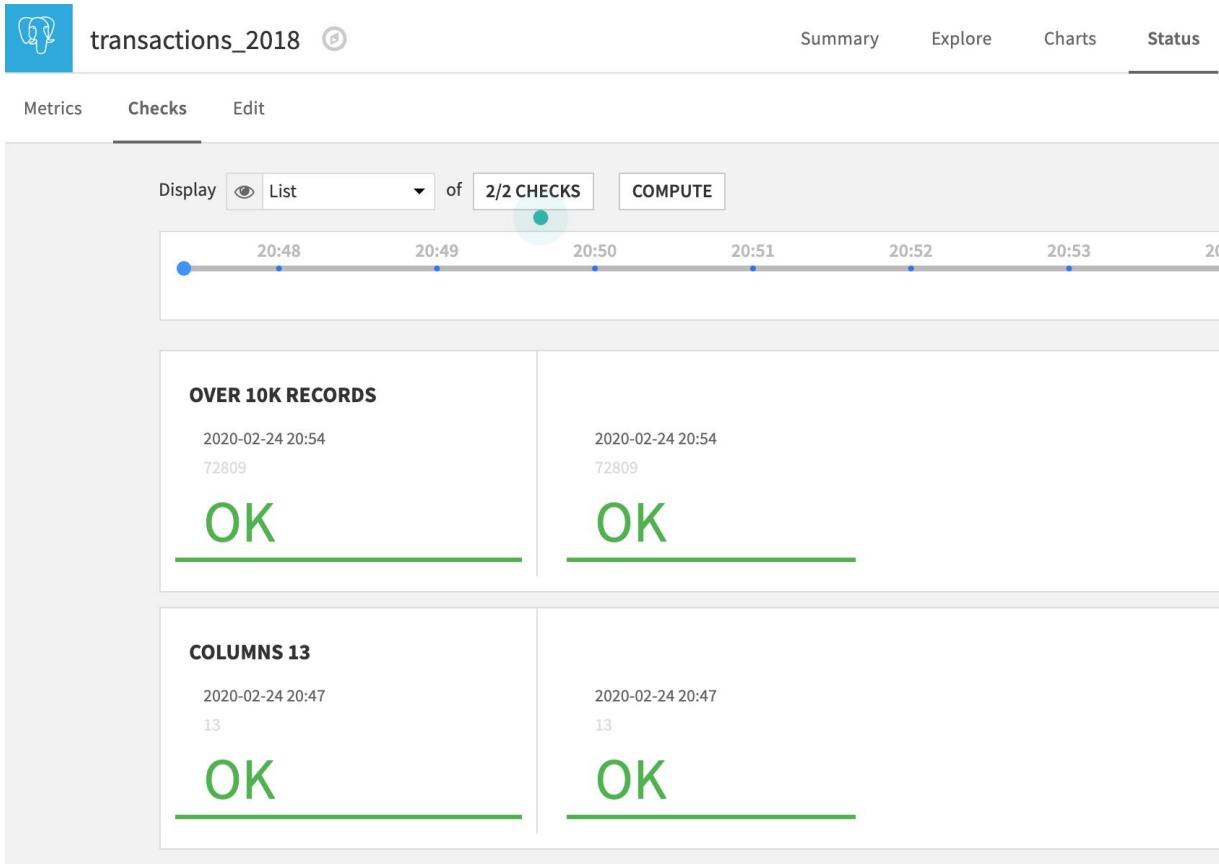
The screenshot shows the 'transactions\_2018' dashboard. At the top, there are tabs for 'Metrics' (disabled), 'Checks' (selected and highlighted with a red box), and 'Edit'. Below the tabs, there's a 'Display' section with a dropdown set to 'List' and a status message '0/2 CHECKS'. A 'COMPUTE' button is also present.



The dialog box is titled 'Checks display settings'. It has three main sections: 'Checks available (1)' with a 'Filter checks...' input field, 'Checks to display (1)' with a 'Remove all' link, and a central area with an '→' button. The status bar at the bottom indicates 'Columns 13' and 'Over 10k records'. At the bottom right are 'CANCEL' and 'SAVE' buttons, with 'SAVE' highlighted by a red box.

Bring the new **checks** to the main tile screen

# You can monitor Check history



The screenshot shows the data iku interface for monitoring check history. The top navigation bar includes a logo, the project name "transactions\_2018", and tabs for Summary, Explore, Charts, and Status. The "Checks" tab is selected.

The main area displays two sections of check history:

- OVER 10K RECORDS:** This section shows two entries from 2020-02-24 at 20:54. Both entries have a status of "OK".
  - First entry: 2020-02-24 20:54, 72809
  - Second entry: 2020-02-24 20:54, 72809
- COLUMNS 13:** This section shows two entries from 2020-02-24 at 20:47. Both entries have a status of "OK".
  - First entry: 2020-02-24 20:47, 13
  - Second entry: 2020-02-24 20:47, 13

At the top of the interface, there is a display configuration section with "Display" set to "List", showing "2/2 CHECKS" with a "COMPUTE" button, and a timeline showing check times from 20:48 to 20:53.

# Create Metrics and Checks - ‘merchants’

Now add metrics and checks for the ‘merchants’ dataset

## Metrics

- Column count
- Record Count

## Checks

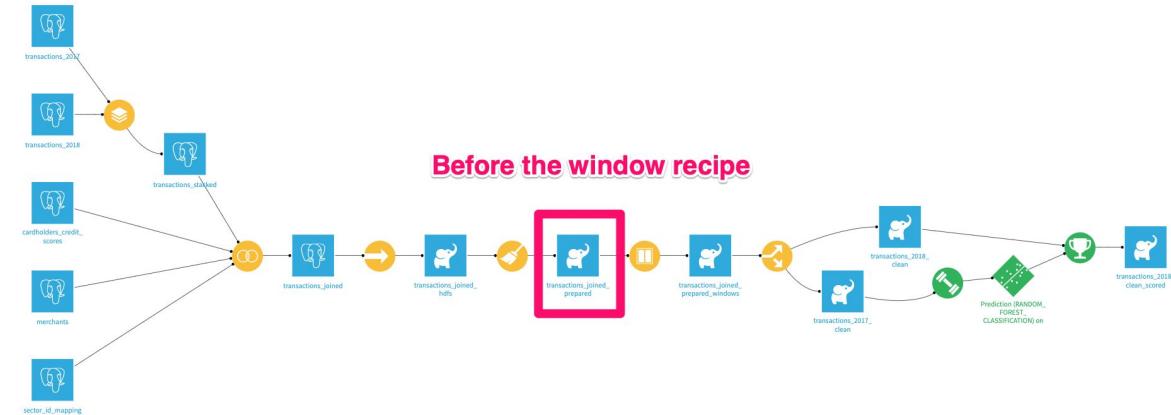
- Column count = 15
- Record Count > 10,000

# 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'

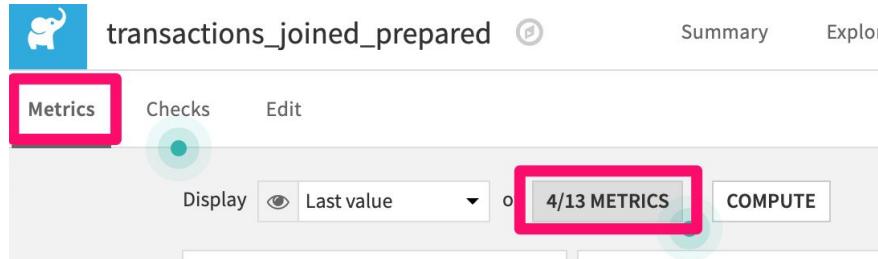
Add two metrics

- Minimum purchase\_amount
- Average fico\_score

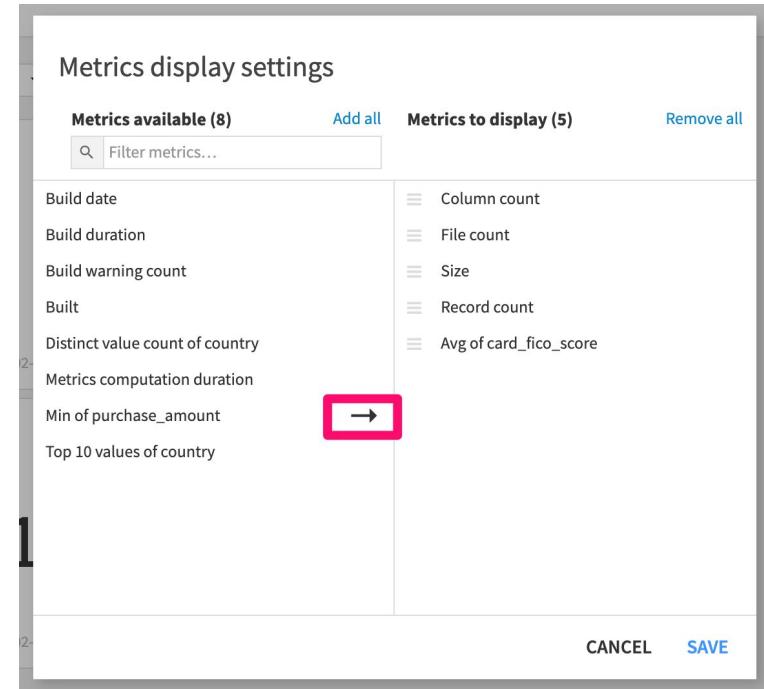
The screenshot shows the Dataiku interface for the dataset 'transactions\_joined\_prepared'. The top navigation bar includes 'Summary', 'Explore', 'Charts', 'Status', 'History', 'Settings', 'PARENT RECIPE', and 'ACTIONS' (with a 'SAVE' button highlighted with a red box and number 6). The main area has tabs for 'Metrics' (highlighted with a red box and number 2) and 'Checks'. The 'Edit' button is also highlighted with a red box and number 1. The 'Metrics' tab displays two sections: 'Records count' (ON) and 'Columns statistics' (ON). The 'Columns statistics' section is expanded, showing a table for the 'purchase\_amount' column with buttons for 'Min', 'Avg', 'Max', 'Sum', 'Count', 'Empty value count', 'Std. dev.', and 'Distinct value'. The 'Auto compute after build' setting for 'Columns statistics' is set to 'YES' (highlighted with a red box and number 5). The 'Min' button for the 'purchase\_amount' column is highlighted with a red box and number 4.

# Create Metrics - 'transactions\_joined\_prepared'

Add these two metrics to the home screen tiles



The screenshot shows the Dataiku interface for a metric named 'transactions\_joined\_prepared'. The 'Metrics' tab is selected and highlighted with a pink box. Below it, there are tabs for 'Checks' and 'Edit'. On the right, there's a 'Display' section with 'Last value' and a dropdown, followed by a button labeled '4/13 METRICS' which is also highlighted with a pink box. A 'COMPUTE' button is to its right.



The screenshot shows the 'Metrics display settings' dialog box. It has two main sections: 'Metrics available (8)' on the left and 'Metrics to display (5)' on the right. The 'Metrics available' section includes items like 'Build date', 'Build duration', 'Build warning count', 'Built', 'Distinct value count of country', 'Metrics computation duration', 'Min of purchase\_amount', and 'Top 10 values of country'. The 'Metrics to display' section includes 'Column count', 'File count', 'Size', 'Record count', and 'Avg of card\_fico\_score'. At the bottom, there are 'CANCEL' and 'SAVE' buttons, with a pink box highlighting the '→' button between the two sections.

# Create Metric Checks - 'transactions\_joined\_prepared'

Add two checks

- Minimum purchase\_amount > 0
- 525 > Average fico\_score < 650

The screenshot shows the Dataiku interface for a dataset named "transactions\_joined\_prepared". The top navigation bar includes tabs for Summary, Explore, Charts, Status, History, and Set. Below the navigation, there are two main tabs: Metrics and Checks, with Checks being the active tab and highlighted by a pink box. An "Edit" button is also highlighted with a pink box. On the left, a sidebar has "Metrics" and "Checks" buttons, with "Checks" also highlighted by a pink box. The main content area is titled "Checks" and contains the text: "The checks are verification about a set of conditions on a metric to do something or to create and alert about them". Below this is a section titled "Add a new check" with three options: "METRIC VALUE IS IN A NUMERIC RANGE" (highlighted with a pink box), "METRIC VALUE IS IN A SET OF VALUES", and "CUSTOM (PYTHON)". At the bottom of the page, under "General settings", there are checkboxes for "Run after build" and "Launch computation after each build".

# Create Metric Checks - 'transactions\_joined\_prepared'

**Minimum purchase\_amount > 0**

**Purchase Amount Positive** (numericRange) 

Metric to check

Min of purchase\_amount  

Minimum

ON 0

Soft minimum

OFF  The cl be

Soft maximum

OFF  The cl at

Maximum

OFF 0

 CHECK

**525 > Average fico\_score < 650**

**Avg FICO Score Range** (numericRange) 

Metric to check

Avg of card\_fico\_score  

Minimum

ON 525

Soft minimum

OFF  The cl below

Soft maximum

OFF  The cl above

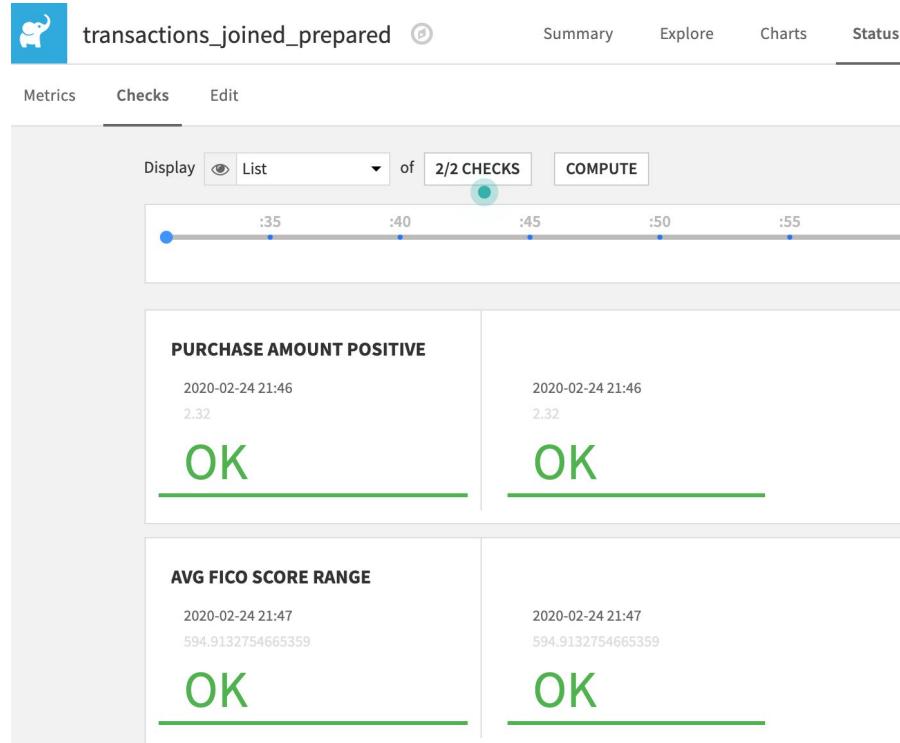
Maximum

ON 650

 CHECK

# Add Checks to the home screen -

## 'transactions\_joined\_prepared'



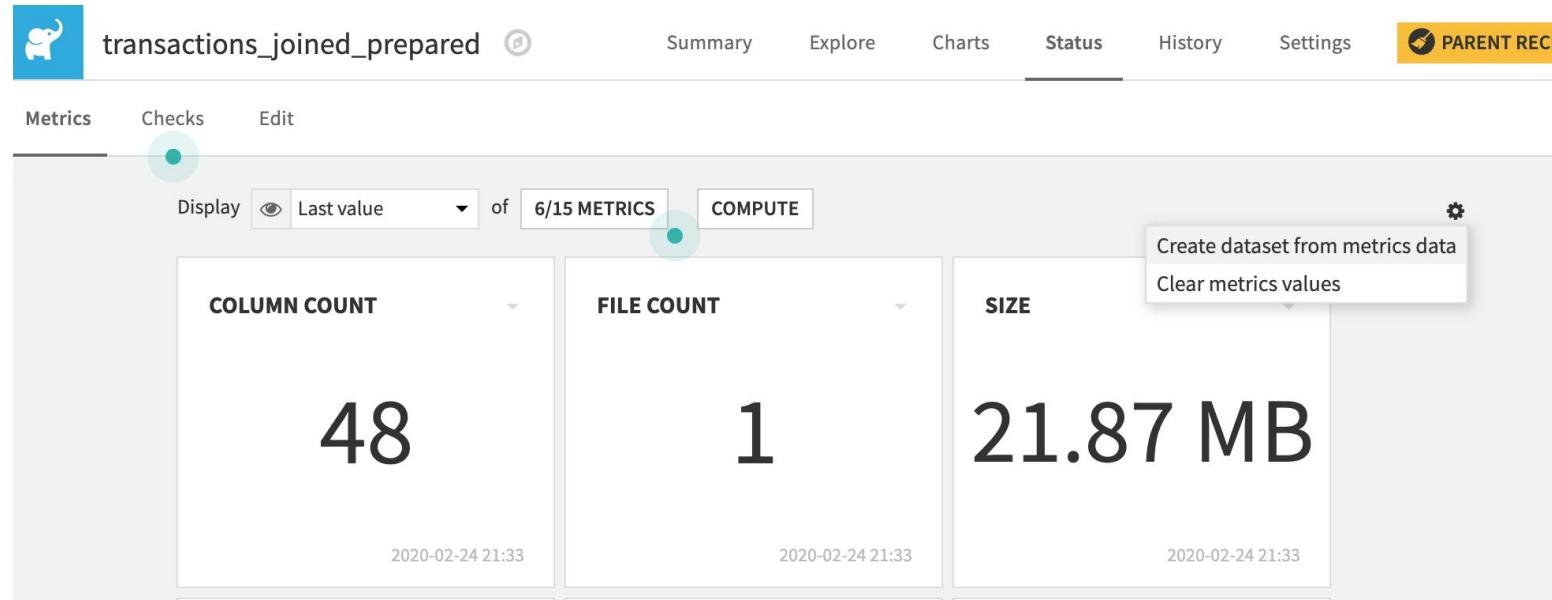
The screenshot shows the dataiku interface with the following details:

- Dashboard Title:** transactions\_joined\_prepared
- Header Buttons:** Summary, Explore, Charts, Status (selected)
- Sub-Header Buttons:** Metrics, Checks (selected), Edit
- Display Options:** Display dropdown set to "List", showing "2/2 CHECKS". A "COMPUTE" button is visible.
- Timeline Slider:** Shows time points :35, :40, :45, :50, :55, and 21.
- Check 1: PURCHASE AMOUNT POSITIVE**
  - Timestamp: 2020-02-24 21:46
  - Value: 2.32
  - Status: OK
- Check 2: AVG FICO SCORE RANGE**
  - Timestamp: 2020-02-24 21:47
  - Value: 594.9132754665359
  - Status: OK

# 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.



The screenshot shows the dataiku interface for a dataset named "transactions\_joined\_prepared". The top navigation bar includes tabs for Summary, Explore, Charts, Status (which is currently selected), History, and Settings, along with a yellow "PARENT RECIPIENT" button. Below the navigation, there are two tabs: "Metrics" (selected) and "Checks". The main area displays three metrics cards:

- COLUMN COUNT**: Value 48, Last value 48, updated 2020-02-24 21:33
- FILE COUNT**: Value 1, Last value 1, updated 2020-02-24 21:33
- SIZE**: Value 21.87 MB, Last value 21.87 MB, updated 2020-02-24 21:33

A context menu is open over the SIZE card, with options: "Create dataset from metrics data" and "Clear metrics values".

## 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

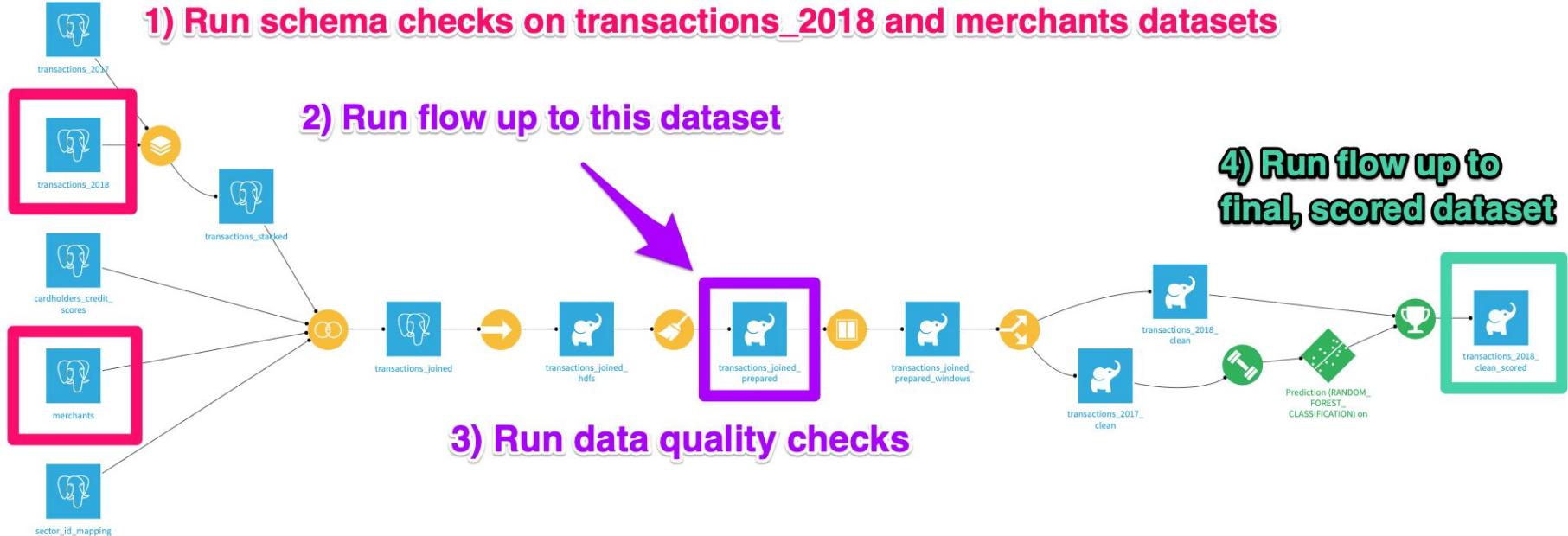
2) **Triggers:**

- 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...

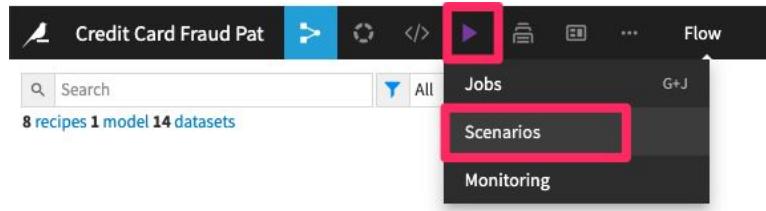
# Recall our Automated Workflow



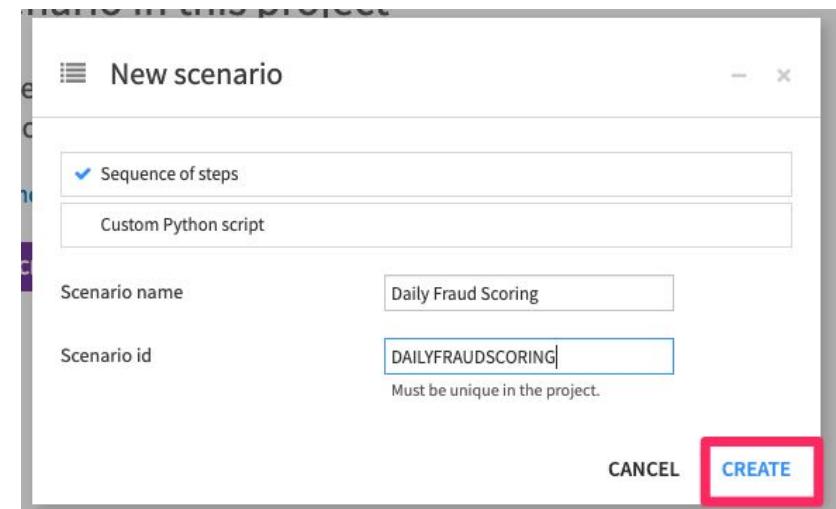
# Create Scenario Trigger, Reporter, and Steps

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 YOUR FIRST SCENARIO



The screenshot shows a modal dialog box titled "New scenario". Inside the dialog, there are two checkboxes: "Sequence of steps" (which is checked) and "Custom Python script". Below these checkboxes are two input fields: "Scenario name" containing "Daily Fraud Scoring" and "Scenario id" containing "DAILYFRAUDSCORING". A note below the "Scenario id" field states "Must be unique in the project." At the bottom of the dialog are two buttons: "CANCEL" and "CREATE", with a red box highlighting the "CREATE" button.

New scenario

Sequence of steps

Custom Python script

Scenario name: Daily Fraud Scoring

Scenario id: DAILYFRAUDSCORING

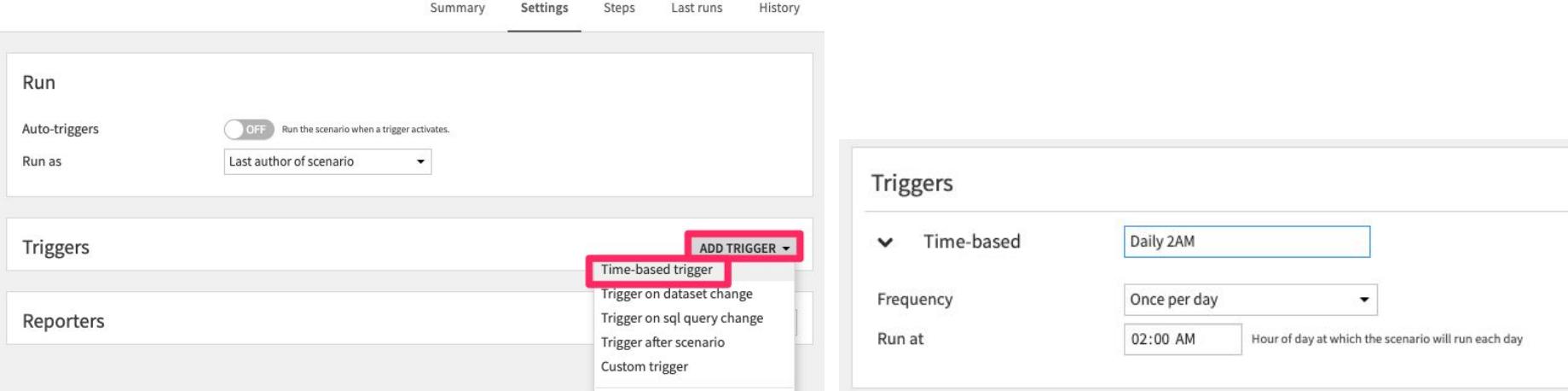
Must be unique in the project.

CANCEL CREATE

# Create Scenario Trigger

## 1. Add a Time-based Trigger:

a. Scenario will run every day at 2AM



The screenshot shows the 'Settings' tab of a scenario configuration page. On the left, under 'Run', the 'Auto-triggers' section has a toggle switch labeled 'OFF' and a dropdown menu set to 'Last author of scenario'. Below it, the 'Triggers' section has a dropdown menu labeled 'ADD TRIGGER ▾' with options: 'Time-based trigger' (highlighted with a red box), 'Trigger on dataset change', 'Trigger on sql query change', 'Trigger after scenario', and 'Custom trigger'. On the right, a modal window titled 'Triggers' is open, showing the selected 'Time-based' trigger. It includes fields for 'Frequency' (set to 'Once per day') and 'Run at' (set to '02:00 AM'). A note below says 'Hour of day at which the scenario will run each day'.

# Create Scenario Reporter

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

Reporters

**ADD REPORTER ▾**

Mail	Email at end	Active <b>ON</b>	<b>X</b>
Send on scenario	End		
Run condition	<input type="radio"/> OFF		
Channel	training_smtp (smtp)		
Sender	sender_email@company.com		
Recipients	recipient_email@company.com		
Subject	DSS scenario \${scenarioName}: \${outcome}		
Message source	Template file		
Template	default		
Attachments	0 attachment	<b>ADD ATTACHMENT ▾</b>	

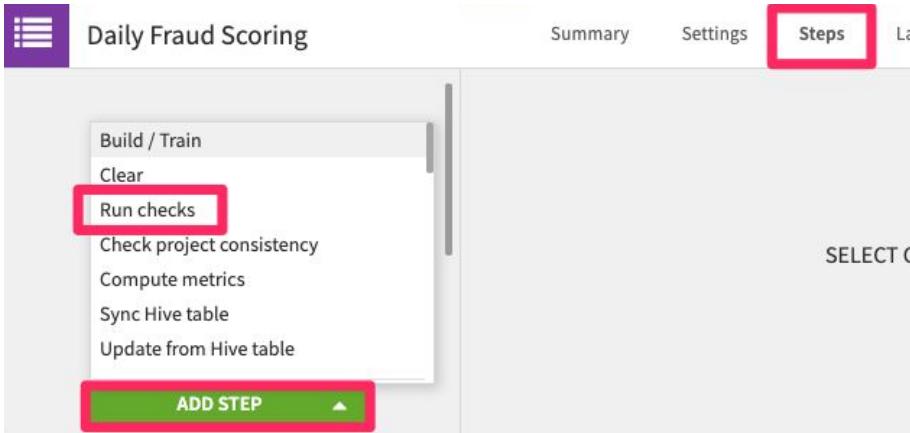
# Create Scenario Steps

## 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'



The screenshot shows the 'Daily Fraud Scoring' project interface. The 'Steps' tab is selected, indicated by a red box. A dropdown menu is open, showing options: 'Build / Train', 'Clear', 'Run checks' (which is highlighted with a red box), 'Check project consistency', 'Compute metrics', 'Sync Hive table', and 'Update from Hive table'. Below the dropdown is a green button labeled 'ADD STEP' with a red box around it.

**Run checks** input dataset schemas

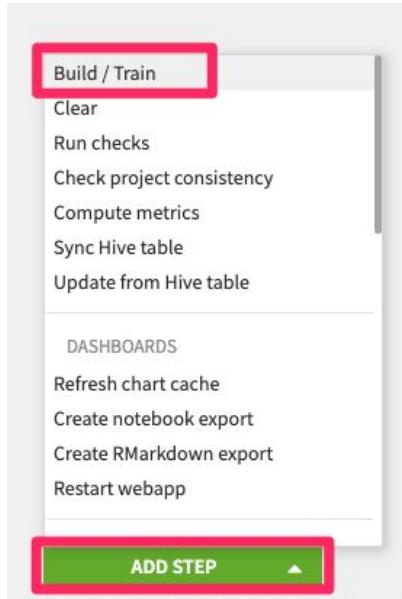
This step executes data checks on one or several "computable" items of the Flow (datasets, managed folders, saved models). Note that you can also configure datasets and managed folders to execute checks automatically on build.

Item	Partitions
merchants	N/A
transactions_2018	N/A

**ADD DATASET TO CHECK** **ADD FOLDER TO CHECK** **ADD MODEL TO CHECK**

# Create Scenario Steps

## b. Build 'transactions\_joined\_prepared'



A screenshot of the 'Build' step configuration screen. The title bar says 'Build' and 'transactions prepared dataset'. The main area contains text about building computable items like datasets and models, and notes about partitioning. A table shows one item: 'transactions\_joined\_prepared' with 'N/A' partitions. Below the table are three buttons: 'ADD DATASET TO BUILD', 'ADD FOLDER TO BUILD', and 'ADD MODEL TO BUILD'. Underneath the table, there are three configuration sections: 'Build mode' (set to 'Force-rebuild dataset and dependencies'), 'Sync. Hive metastore' (with options for 'Build only this dataset', 'Build required datasets', and the selected 'Force-rebuild dataset and dependencies'), and 'Ignore failure' (with options for 'Build missing dependent datasets then this one' and 'If no prior step failed').

## 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'

**Run checks** FICO and purchase amount

This step executes data checks on one or several "computable" items of the Flow (datasets, managed folders, saved models). Note that you can also configure datasets and managed folders to execute checks automatically on build.

Item	Partitions
 transactions_joined_prepared	N/A

[ADD DATASET TO CHECK](#) [ADD FOLDER TO CHECK](#) [ADD MODEL TO CHECK](#)

# Create Scenario Steps

d. Build 'transactions\_2018\_clean\_scored'

**Build**      Transactions with Fraud Predictions

This step lets you build one or several "computable" items of the Flow (build datasets, build managed fol saved models)

For partitioned datasets (and folders), you can use partitioning specs

Item	Partitions
 transactions_2018_clean_scored	N/A

**ADD DATASET TO BUILD**    **ADD FOLDER TO BUILD**    **ADD MODEL TO BUILD**

# Run your Scenario

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

The screenshot shows the 'Daily Fraud Scoring' application interface. At the top, there are tabs: Summary, Settings, Steps, **Last runs** (which is highlighted with a red box and has a red number '1' above it), History, and a 'SAVED!' button. Below these are 'List' and 'Graphs' buttons.

**Run logs** section (left side):

- Running** (highlighted with a red box and has a red number '2' above it)
- Manual trigger
- REFRESH
- ABORT

**Last runs** section (right side):

Unfinished Run	Manual trigger	View scenario log	on 2020-02-25
		Download diagnosis	at 11:41
<b>Details</b>			
<input checked="" type="checkbox"/> <b>Run checks</b> – input dataset schemas	<a href="#">View step log</a>	at 11:41	0s
<input checked="" type="checkbox"/> Checked dataset <a href="#">merchants</a> in CREDITCARDFRAUDPAT (partition: NP)			
<input checked="" type="checkbox"/> Checked dataset <a href="#">transactions_2018</a> in CREDITCARDFRAUDPAT (partition: NP)			
<input checked="" type="checkbox"/> <b>Build</b> – transactions prepared dataset	<a href="#">View step log</a>	at 11:41	not finished
<input checked="" type="checkbox"/> Triggered job <a href="#">sched_build_2020-02-25T16-41-00.474</a>			

# 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

The screenshot shows the 'transactions\_joined' dataset configuration. It includes tabs for Connection, Preview, Schema, and Advanced. Under the Connection tab, the 'Connection' dropdown is set to 'postgres-local'. The 'Table' dropdown is highlighted with a red box and contains the value `\${projectKey}\_transactions\_joined`.

The screenshot shows the 'Daily Fraud Score...' scenario settings. Under the 'Reporters' section, there is a 'Mail' configuration. The 'Recipients' field contains the variable `\${recipient\_list}`. The 'Subject' field contains the formula-based templating 'DSS scenario \${scenarioName}: \${outcome}'.

The screenshot shows a 'Script' step named 'compute\_transactions\_joined' with 8 steps and 10000 rows. The 'Expression' field contains the formula: `if(installments > ${installment_threshold},1`. This expression is highlighted with a red box.

# Accessing Variables - Syntax

## Prepare recipe Formula processor

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

## SQL, Hive

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

## Python

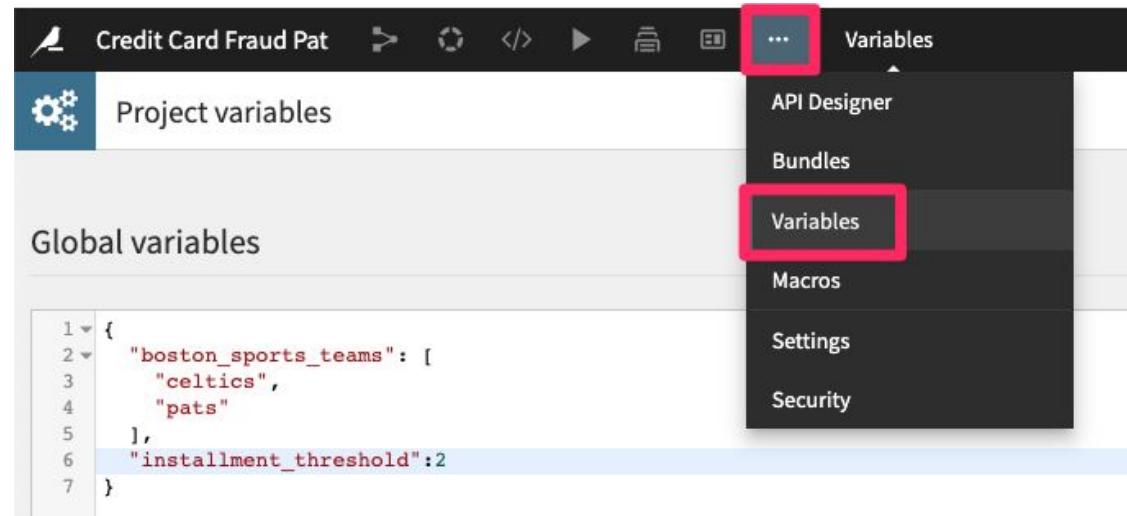
```
import dataiku
variable = dataiku.get_custom_variables()['VARIABLE_NAME']
```

## 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

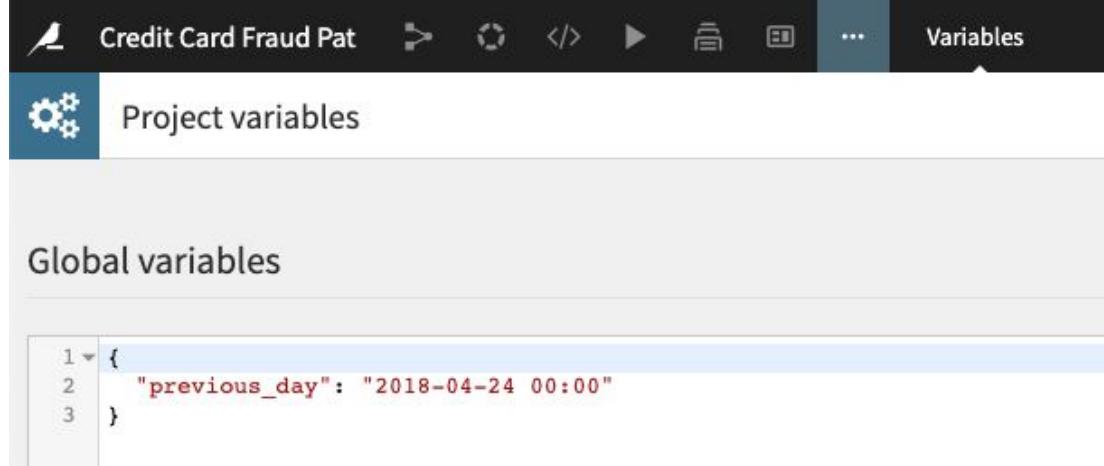
## Steps

- 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"
```



Credit Card Fraud Pat

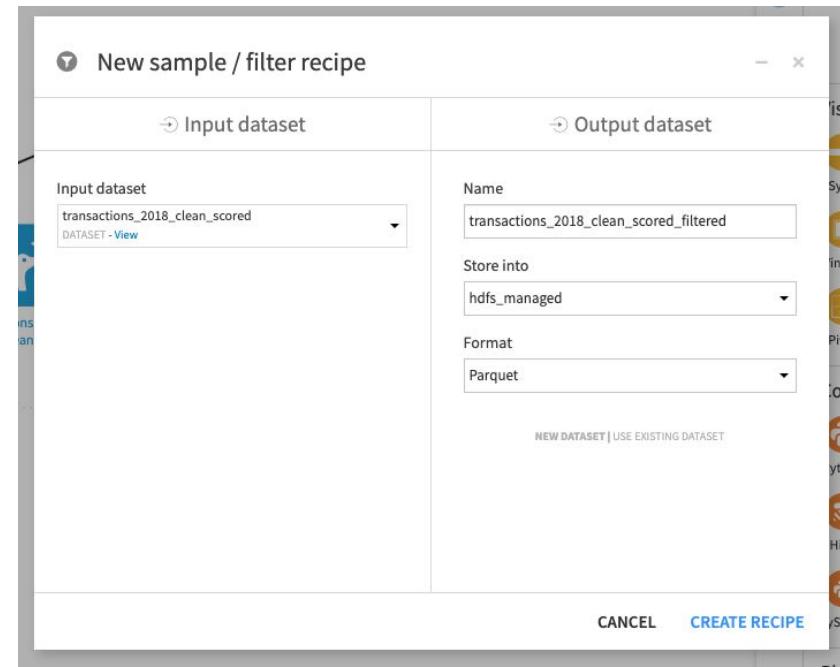
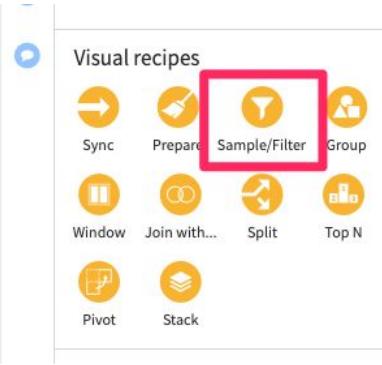
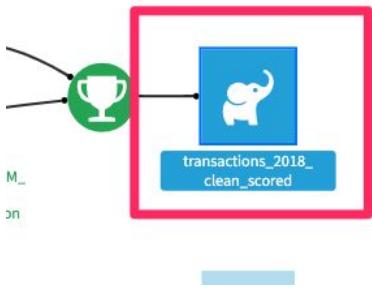
Variables

Project variables

Global variables

```
1 {  
2   previous_day: "2018-04-24 00:00"  
3 }
```

# Create a New Filter Recipe

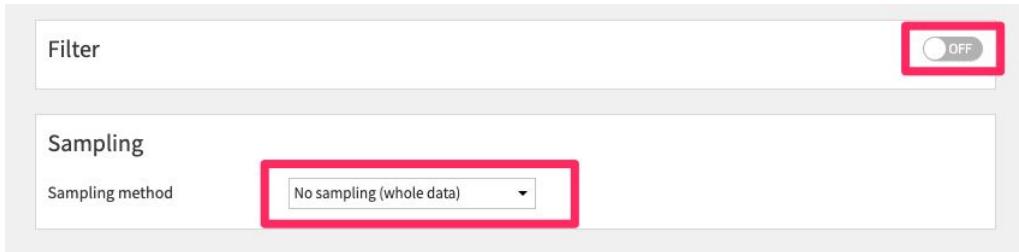


A screenshot of the "New sample / filter recipe" dialog box. The input dataset is set to "transactions\_2018\_clean\_scored". The output dataset is configured with the following settings:

- Name: transactions\_2018\_clean\_scored\_filtered
- Store into: hdfs\_managed
- Format: Parquet

At the bottom right are "CANCEL" and "CREATE RECIPE" buttons.

# Use the Formula to Filter Rows on the Project Variable



Filter

Sampling

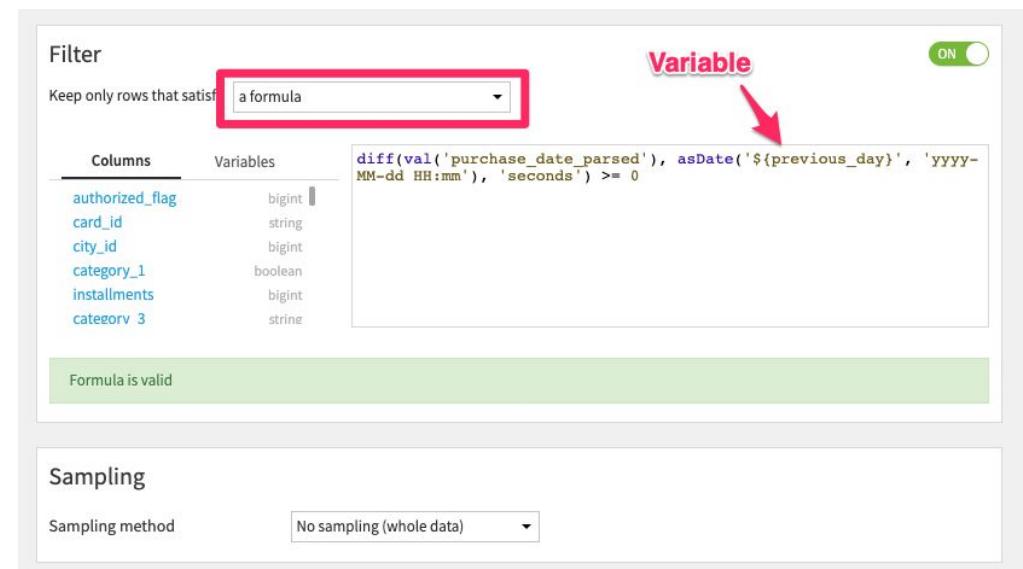
Sampling method

No sampling (whole data)

OFF

## Formula text

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



Filter

Keep only rows that satisfy

a formula

**Variable**

ON

Columns	Variables
authorized_flag	bigint
card_id	string
city_id	bigint
category_1	boolean
installments	bigint
category_3	string

diff(val('purchase\_date\_parsed'), asDate('\${previous\_day}', 'yyyy-MM-dd HH:mm'), 'seconds') >= 0

Formula is valid

Sampling

Sampling method

No sampling (whole data)

# Run the Filter Recipe and Explore the Output Dataset

transactions\_2018\_clean\_scored\_filtered  Summary

Viewing dataset sample [Configure sample](#) 

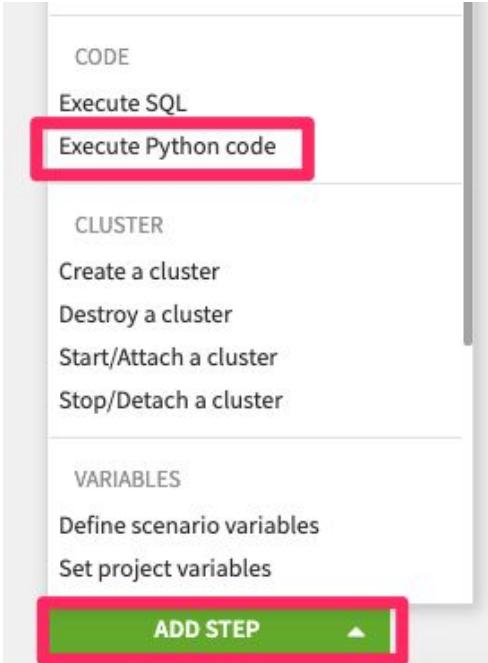
1707 rows, 61 cols

month_lag	purchase_amount	purchase_date	purchase_date_parsed	purchase_year
bigint Integer	double Decimal	string Date (unparsed)	date Date	bigint 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

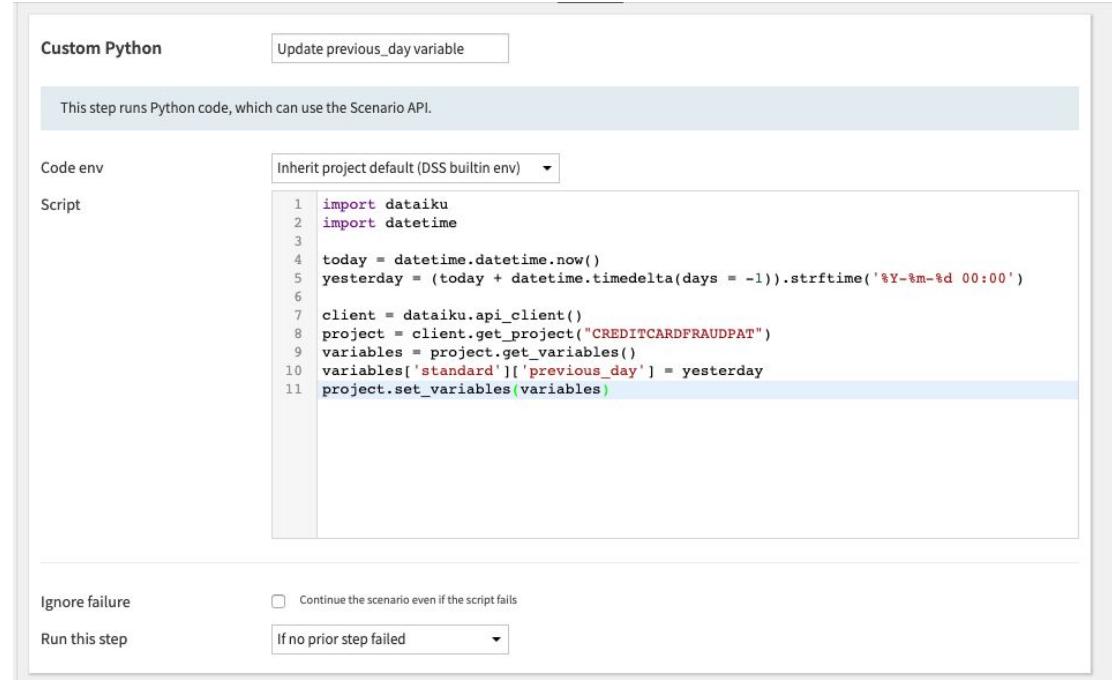


The screenshot shows the Dataiku interface with a sidebar on the left and a main workspace on the right.

- Sidebar:**
  - CODE**
    - Execute SQL
    - Execute Python code** (highlighted with a red box)
  - CLUSTER**
    - Create a cluster
    - Destroy a cluster
    - Start/Attach a cluster
    - Stop/Detach a cluster
  - VARIABLES**
    - Define scenario variables
    - Set project variables

**Bottom:** A green button labeled "ADD STEP" with a red border and a small upward arrow icon.

(Code to copy/paste on next slide)



The screenshot shows the "Custom Python" step configuration dialog.

**Step Name:** Update previous\_day variable

**Description:** This step runs Python code, which can use the Scenario API.

**Code env:** Inherit project default (DSS builtin env)

**Script:**

```
1 import dataiku
2 import datetime
3
4 today = datetime.datetime.now()
5 yesterday = (today + datetime.timedelta(days = -1)).strftime('%Y-%m-%d 00:00')
6
7 client = dataiku.api_client()
8 project = client.get_project("CREDITCARDFRAUDPAT")
9 variables = project.get_variables()
10 variables['standard']['previous_day'] = yesterday
11 project.set_variables(variables)
```

**Ignore failure:**  Continue the scenario even if the script fails

**Run this step:** If no prior step failed

# 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

Daily Fraud Scoring

Summary    Settings    **Steps**    Last runs    History    **SAVE** ▾

Run checks input dataset schemas ...

Build transactions prepared dataset ...

Run checks FICO and purchase amount ...

Build Transactions with Fraud Predict... ...

Custom Python Update previous\_da... ...

**Build Filtered dataset** ...

**Build**    Filtered dataset

This step lets you build one or several "computable" items of the Flow (build datasets, build managed folders, train saved models)

For partitioned datasets (and folders), you can use partitioning specs

Item	Partitions
transactions_2018_clean_scored_filtered	N/A

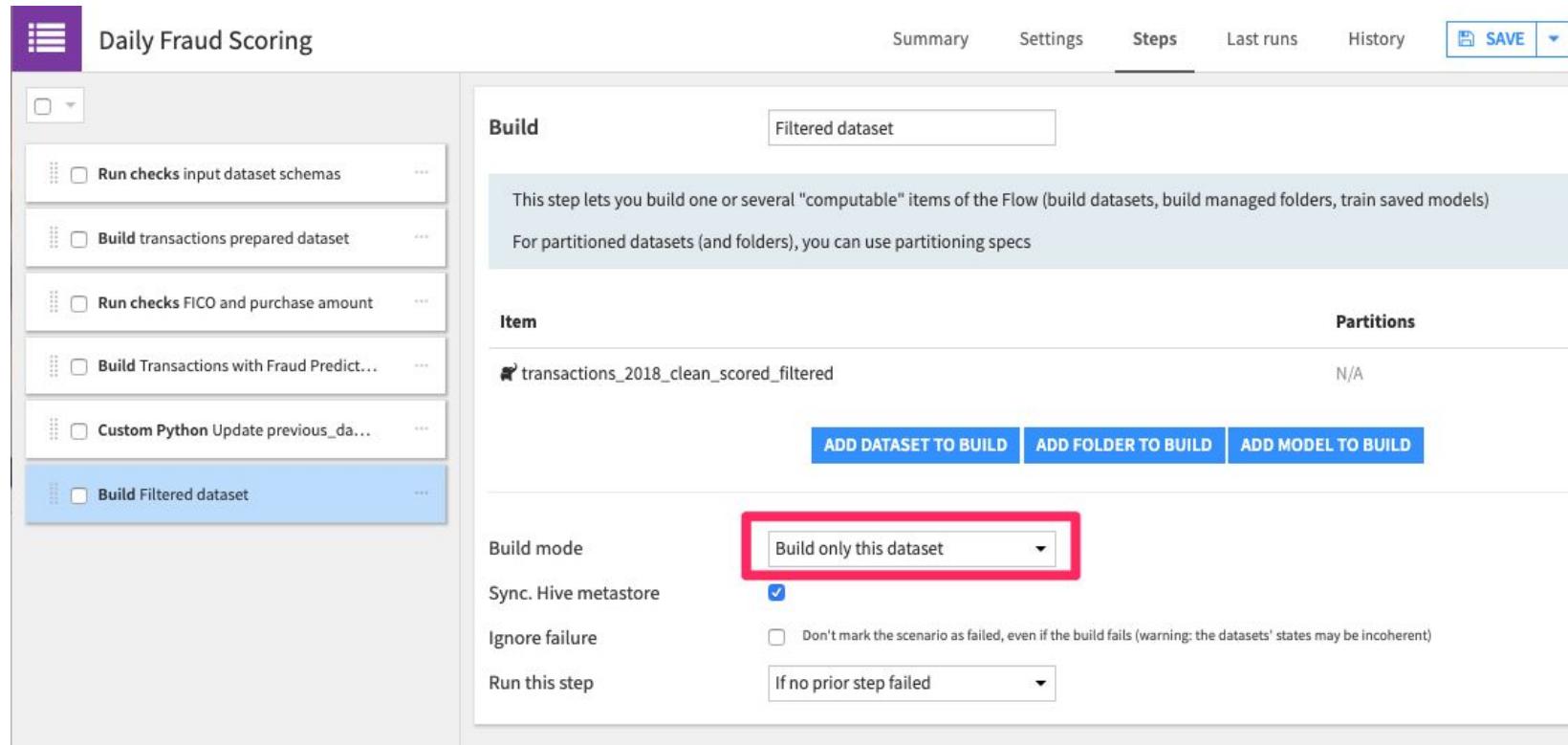
**ADD DATASET TO BUILD**    **ADD FOLDER TO BUILD**    **ADD MODEL TO BUILD**

Build mode    **Build only this dataset** ▾

Sync. Hive metastore

Ignore failure  Don't mark the scenario as failed, even if the build fails (warning: the datasets' states may be incoherent)

Run this step **If no prior step failed** ▾



# Attach Dataset to Email Reporter

Attach 'transactions\_2018\_clean\_scored\_filtered' to the email reporter

Reporters

ADD REPORTER ▾

Mail

Email at end

Active

Send on scenario

End

Run condition

OFF

Channel

training\_smtp (smtp)

Sender

sender\_email@company.com

Formula-based templating

Recipients

recipient\_email@company.com

Formula-based templating

Subject

DSS scenario \${scenarioName}: \${outcome}

Formula-based templating

Message source

Template file

Template

default

Attachments

0 attachment

ADD ATTACHMENT ▾

Scenario log

Dataset data

Folder contents

File in folder

Notebook export

RMarkdown report

1 attachments

ADD ATTACHMENT ▾

Dataset data

Attached dataset

transactions\_2018\_clean\_scored.

Export to file Other exports

Attach a dataset to the mail (limited to 20 MB, unless overridden in general settings).

# Check the Result

Check your project variable

'previous\_day' - it should have changed to yesterday's date

Project variables

Global variables

```
1 ▾ {  
2   "previous_day": "2020-02-25 00:00"  
3 }
```

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

transactions\_2018\_clean\_scored\_filtered

Viewing dataset sample   [Configure sample](#)

0 row, 61 cols

authorized_flag	card_id	city_id	category_1	installments	category_3	mercha
bigint	string	bigint	boolean	bigint	string	bigint
Text	Text	Text	Text	Text	Text	Text

# Deployment to Production (Demo)

# Deployment to Production - Batch Scoring Pipeline

## Objectives :

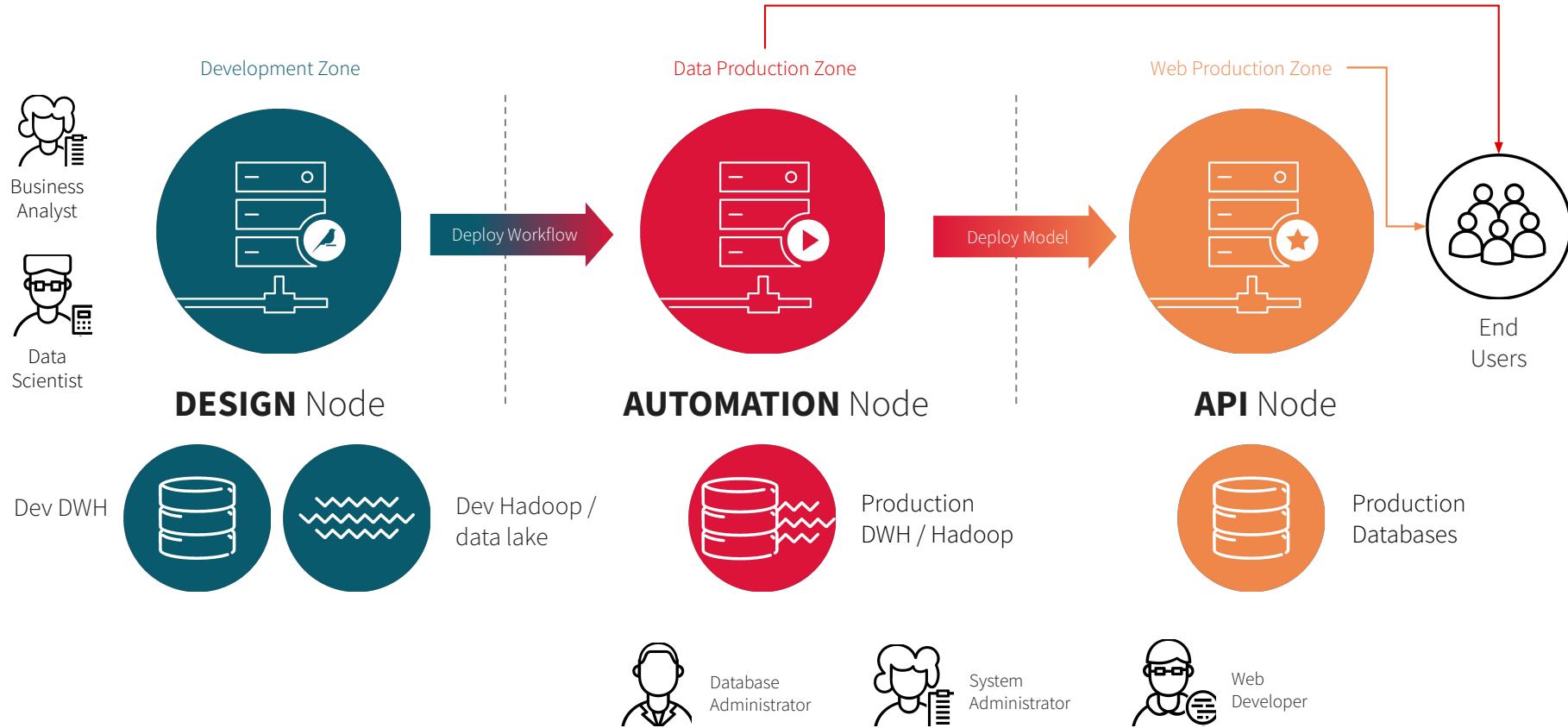
We want to have a **safe environment** where our scoring pipeline is not at risk of being altered by modifications in the flow.

We also want to be connected to our production databases.

## Solution :

Deploy our project to the **Automation Node**.

# DSS Nodes



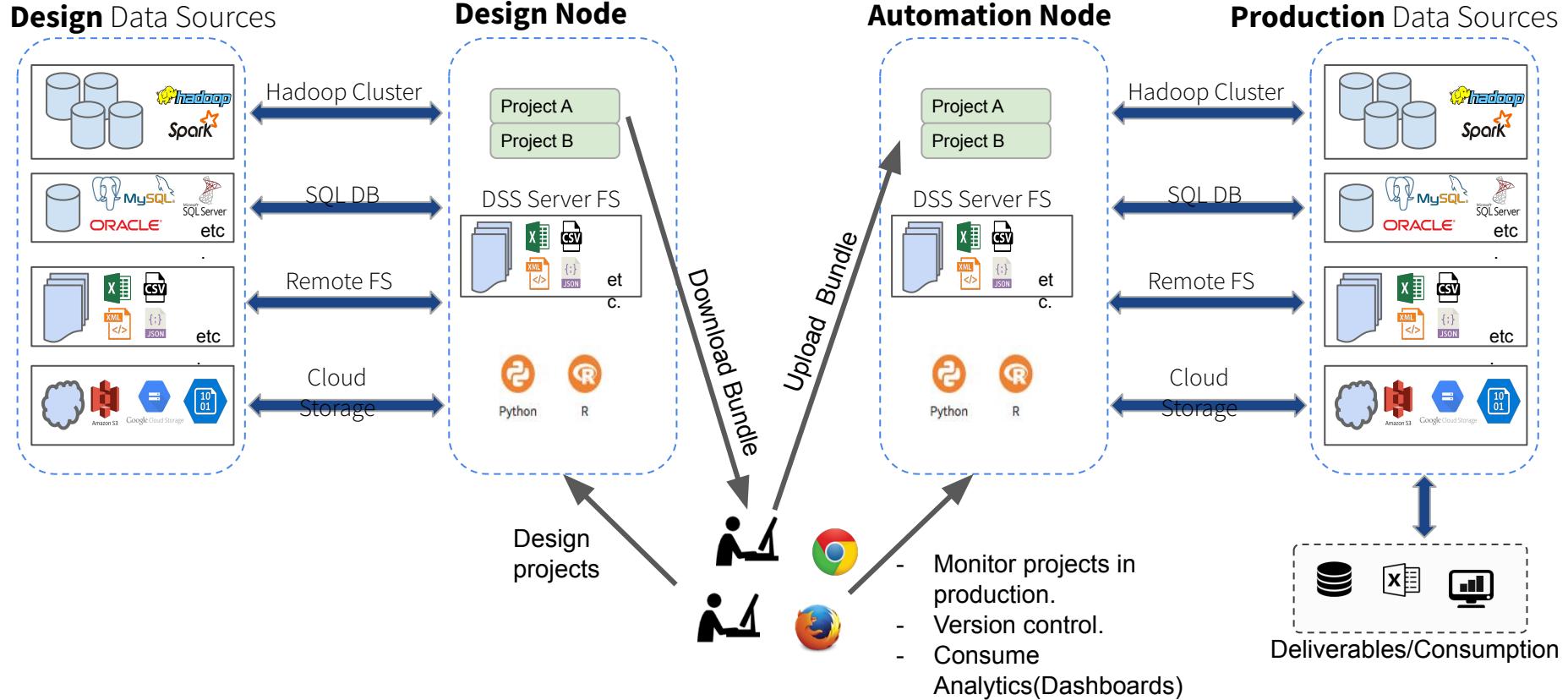
# Deployment to Production - Batch Scoring Pipeline

Moving a project from a DSS [Design Node](#) to [Automation Node](#):

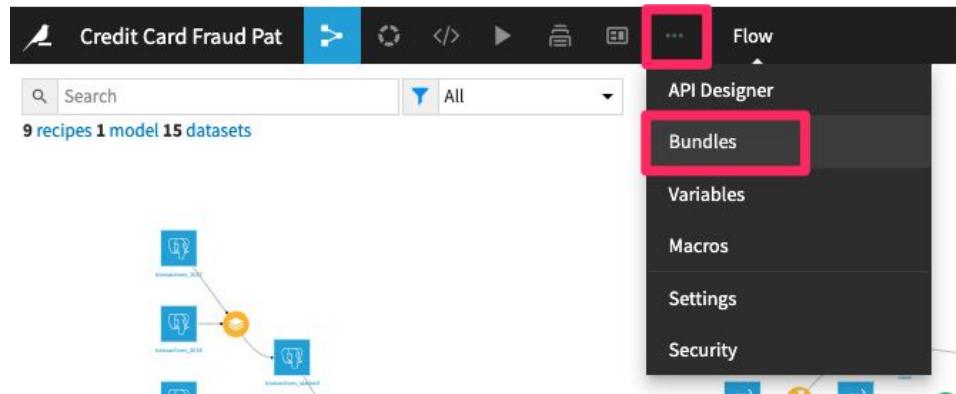
1. “[Bundle](#)” your project in the [Design Node](#) - this will create a zip file with the project’s config
2. [Download](#) the [bundle](#) to your local machine
3. [Upload](#) the [bundle](#) to the [Automation Node](#) to create a new project (or update an existing one)
  -  This step may require dataset connection remapping
4. [Activate](#) the [bundle](#) on the [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**

New bundle v1 CREATE Content Commit log Diff

**Project configuration**

Flow	Lab	Dashboard
15 DATASETS	0 FOLDERS	9 RECIPES
1 MODELS	0 NOTEBOOKS	1 DASHBOARDS
	1 ANALYSES	

The whole project configuration is always included in your Bundle.  
Additionally, the bundle can contain selected data from the project: datasets (non partitioned), managed folders and saved models.  
When activating the bundle in the Automation node, the data will be reloaded.

**Additional content**

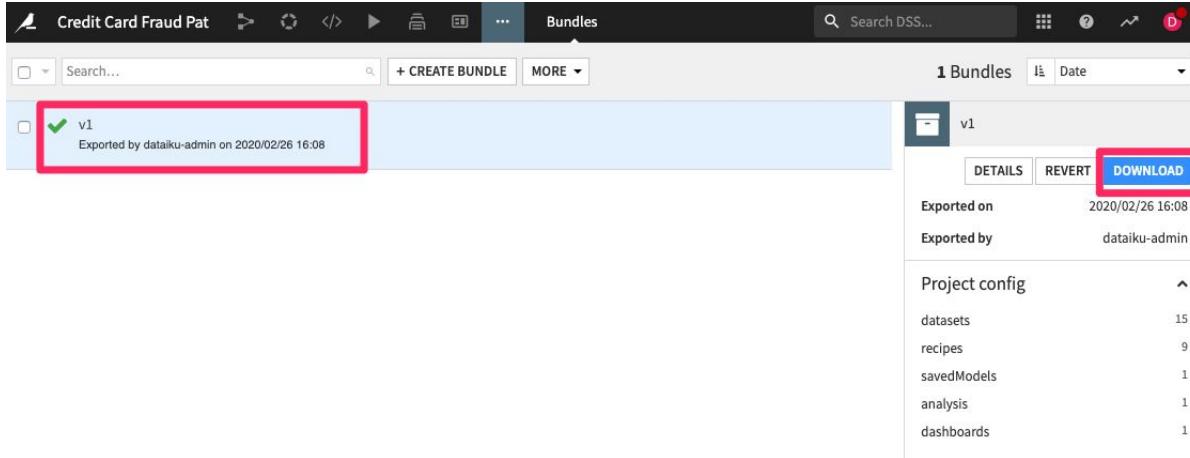
Datasets + ADD Managed folders No includable managed folder.

Saved models + ADD Filter...  
Prediction (RANDOM\_FOREST\_CLASSIFICATION) on transactions\_2017\_clean  
Prediction (Binary\_classification)

**Release notes**

# Download the Bundle

Select the **bundle** and **download** it



The screenshot shows the Dataiku interface with the following details:

- Bundles Page:** The top navigation bar includes icons for back, forward, search, and bundles. The "Bundles" tab is active.
- Search Bar:** A search bar with placeholder text "Search..." and a magnifying glass icon.
- Create Bundle:** A button labeled "+ CREATE BUNDLE".
- More Options:** A "MORE" dropdown menu.
- Bundles List:** A table showing one bundle entry:
  - v1**: The bundle name, preceded by a green checkmark icon.
  - Exported by dataiku-admin on 2020/02/26 16:08**: The export details.
- Details Panel:** A sidebar for the selected bundle "v1":
  - Exported on**: 2020/02/26 16:08
  - Exported by**: dataiku-admin
  - Project config** section with counts:
    - datasets: 15
    - recipes: 9
    - savedModels: 1
    - analysis: 1
    - dashboards: 1
- Buttons:** In the sidebar, there are "DETAILS", "REVERT", and a large blue "DOWNLOAD" button, which is highlighted with a red box.

# Upload the Bundle to the Automation Node

Go to the [Automation Node](#)

<https://dssX-automation-COMPANYNAME.training.dataiku.com>

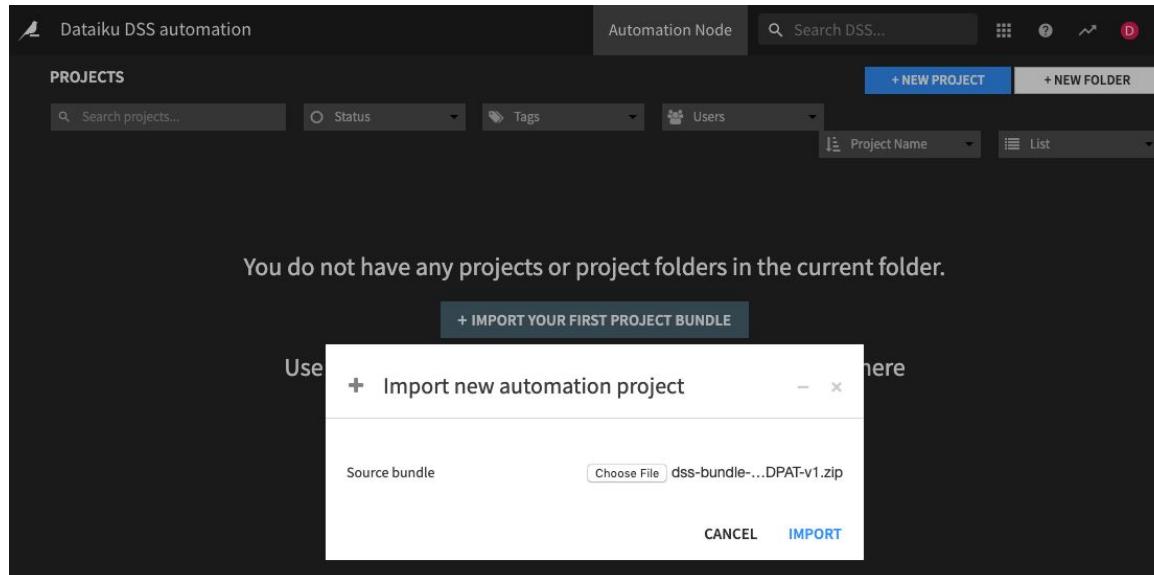
(replace the X with your instance number)

Username: userY

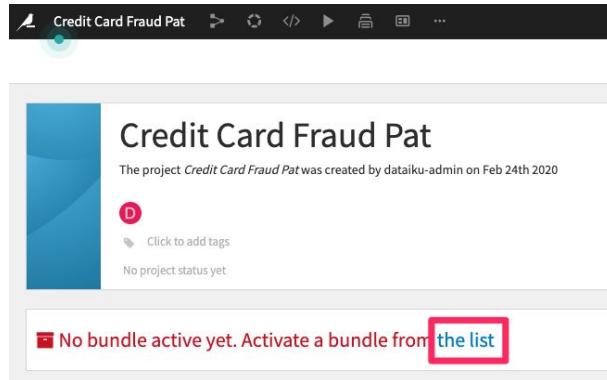
Password: PASSWORD\_FROM\_INSTANCE

# 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



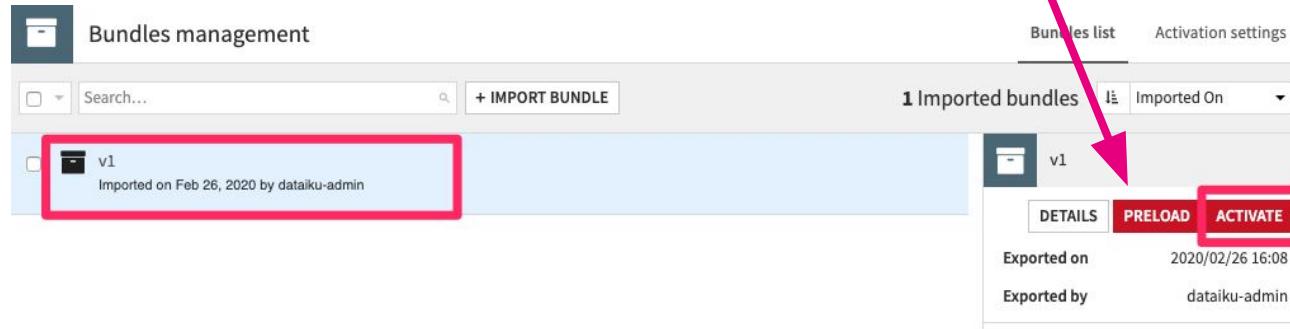
Credit Card Fraud Pat

The project *Credit Card Fraud Pat* was created by dataiku-admin on Feb 24th 2020

D Click to add tags

No project status yet

No bundle active yet. Activate a bundle from the list



Bundles management

+ IMPORT BUNDLE

v1 Imported on Feb 26, 2020 by dataiku-admin

Bundles list Activation settings

1 Imported bundles

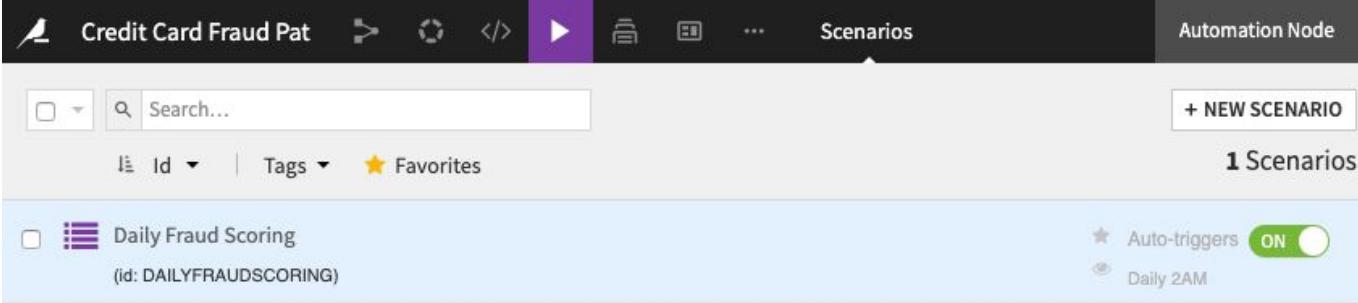
v1

DETAILS PRELOAD ACTIVATE

Exported on 2020/02/26 16:08

Exported by dataiku-admin

# Finally, Activate your Scenario



The screenshot shows the Dataiku interface with the 'Automation Node' tab selected in the top navigation bar. The main view displays a list of scenarios. One scenario is listed: 'Daily Fraud Scoring' (id: DAILYFRAUDSCORING). The scenario has an 'Auto-triggers' toggle switch set to 'ON' and is scheduled to run 'Daily 2AM'. There is also a '+ NEW SCENARIO' button.

Credit Card Fraud Pat

Scenarios

Automation Node

+ NEW SCENARIO

1 Scenarios

Daily Fraud Scoring  
(id: DAILYFRAUDSCORING)

Auto-triggers ON

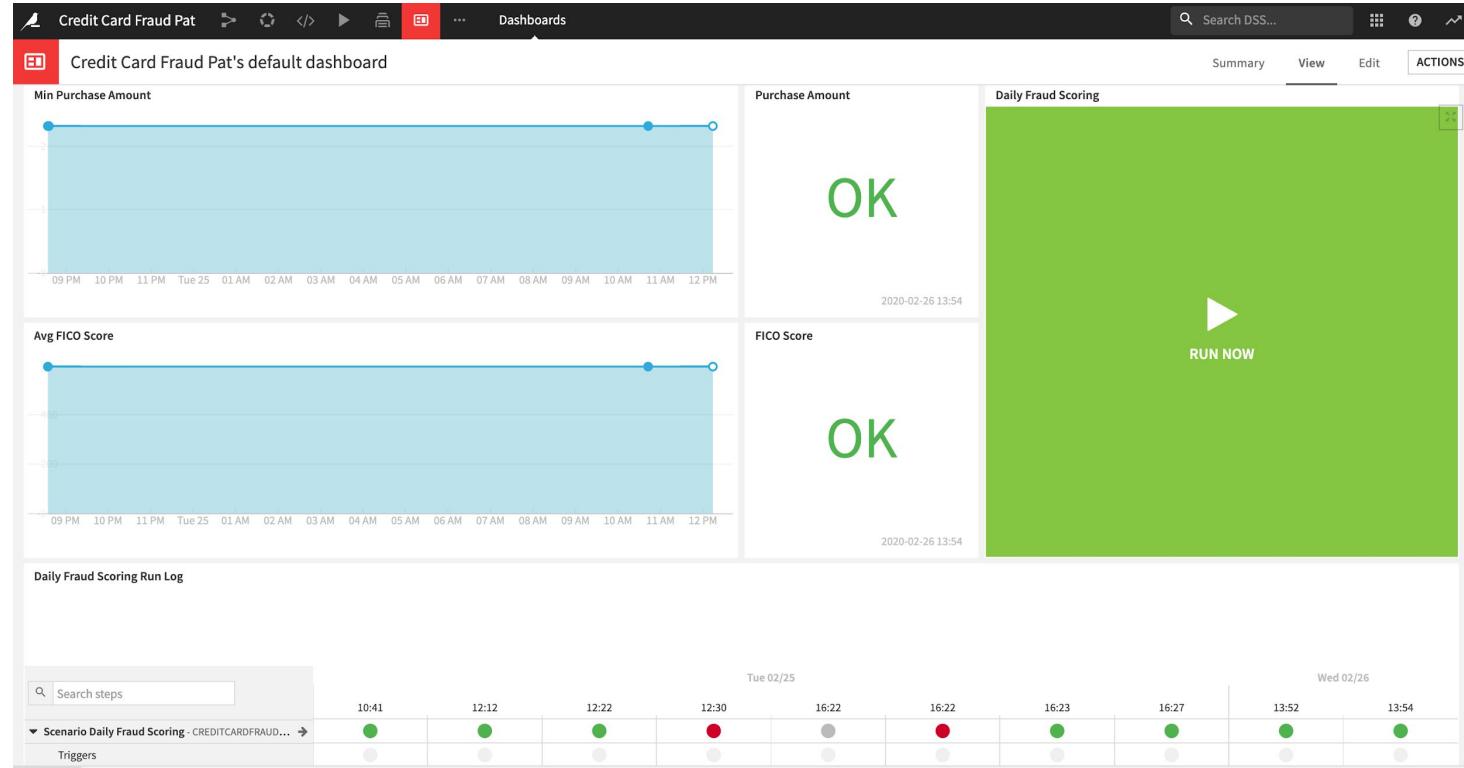
Daily 2AM

# 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

## community.dataiku.com

- Public Q&A
- (like stack overflow)

# Additional Resources

## academy.dataiku.com

- Tutorials
- Videos

## doc.dataiku.com

- Extensive technical documentation about DSS

## gallery.dataiku.com

- Sample projects