

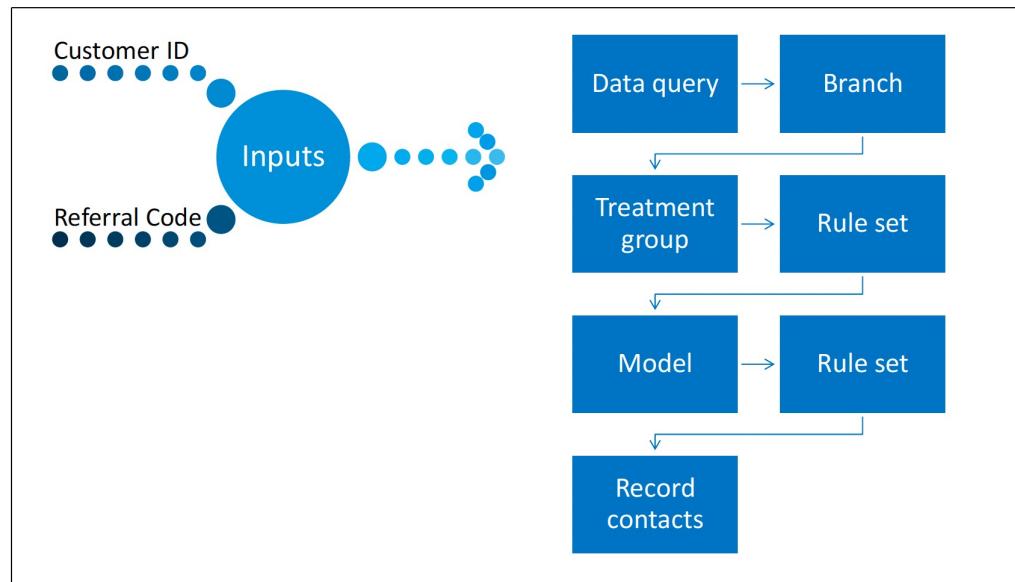


2) SAS Intelligent Decisioning (BDIDOD)



Objective

- Understand an overview of SAS Intelligence Decisioning (ID)
- Be able to create a decision flow
- Be able to perform a simple test and publish a decision flow



The inputs to the decision are the customer's ID and a referral code. The decision contains a series of nodes to perform the needed processing.

Outline

1. SAS ID: The Big Picture
 1. Decision flow: (1) define the process -> (2) input/output -> (3) configure node
 2. Decision types
2. Decision Basics
 1. Rule sets and Rules
 2. Branching and Cross-Branch Linking
3. Advanced Topics
 1. Custom code (data query, DS2 code, Python code)
 2. Models (Build -> Test -> Register)
4. Treatment & Treatment Group (optional)
5. Testing, Publishing, Validating, and Troubleshooting
 1. Basic test
 2. Publishing and validation test

Lesson 1: SAS Intelligent Decisioning: The Big Picture

1.1 Introduction to SAS Intelligent Decisioning

1.2 Introduction to Decisions

Lesson 1: SAS Intelligent Decisioning: The Big Picture

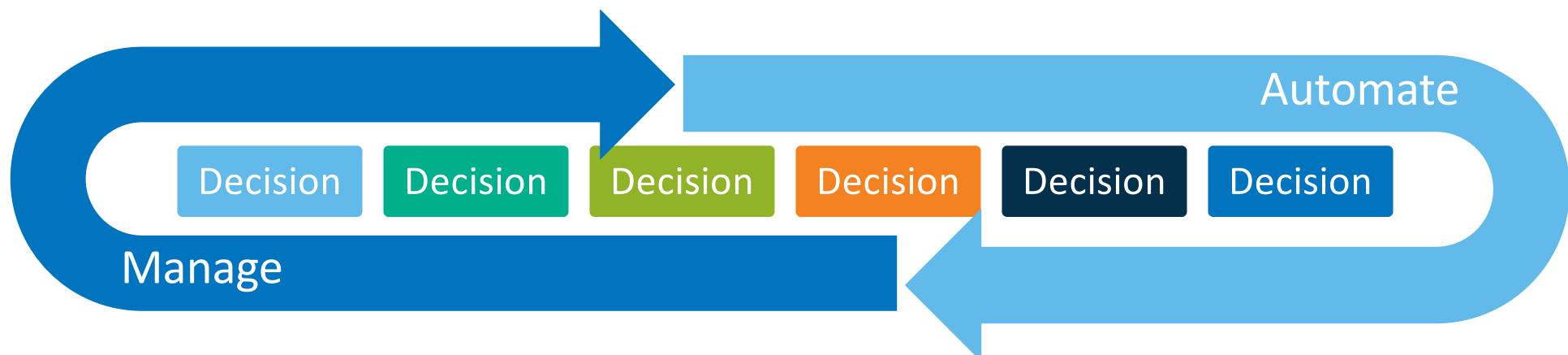
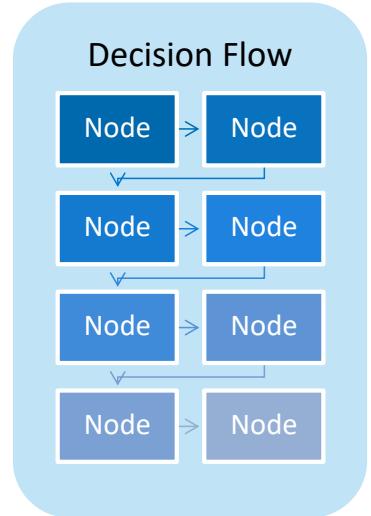
1.1 Introduction to SAS Intelligent Decisioning

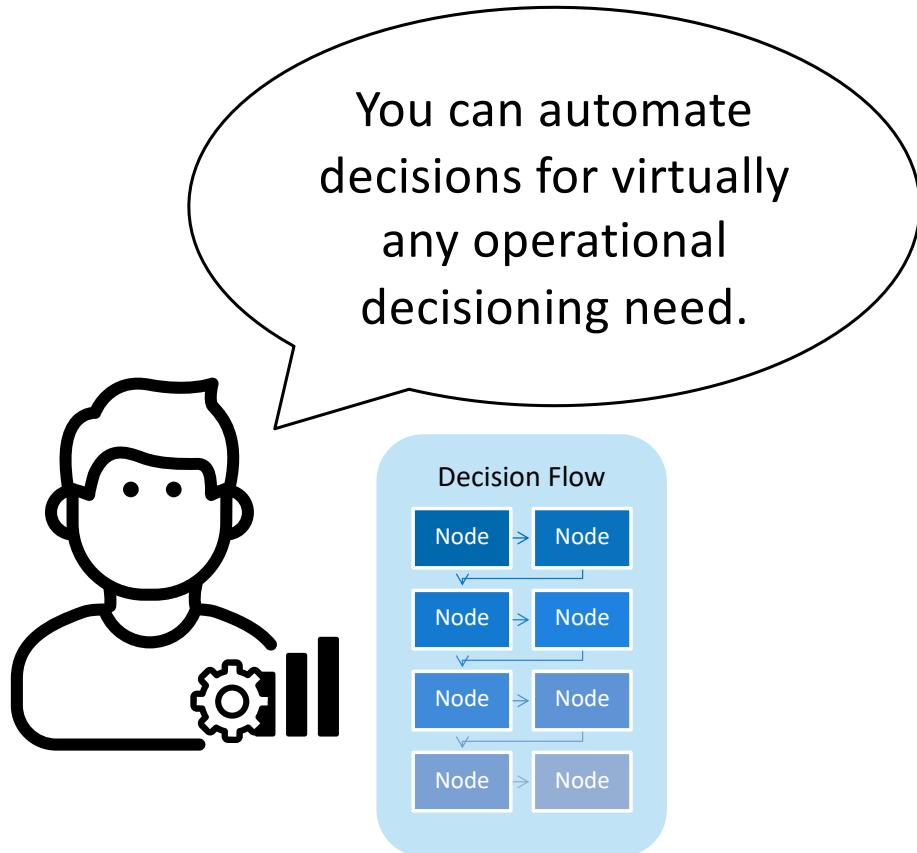
1.2 Introduction to Decisions

SAS Intelligent Decisioning



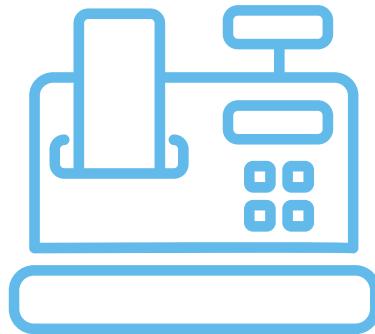
Analytically
Powered







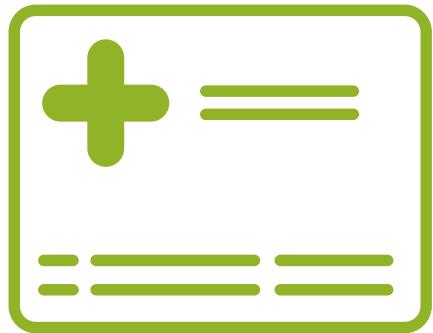
Should we approve
this loan?



Is this a fraudulent
transaction?



Is this machine about to
experience a defect?



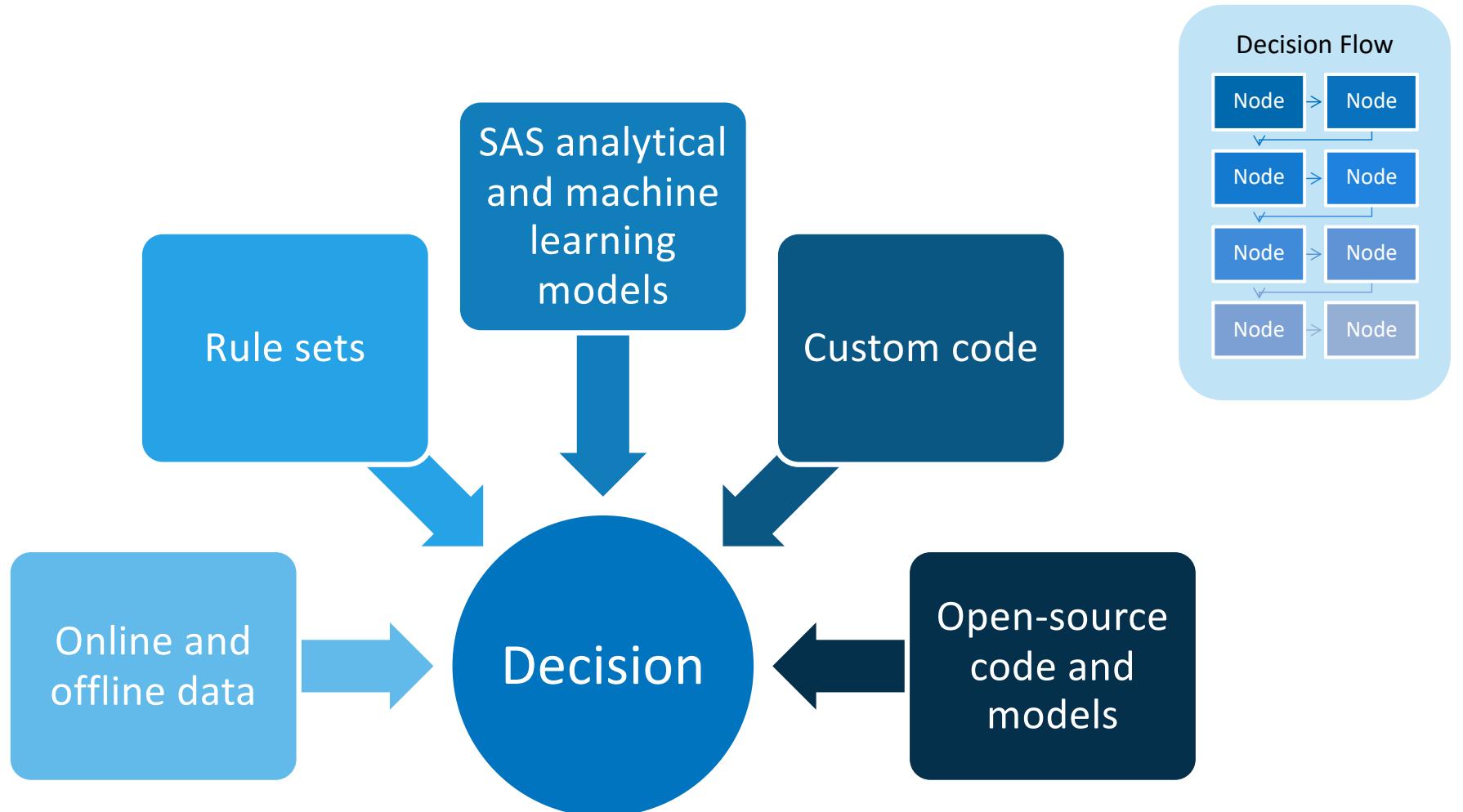
Is this claim valid?

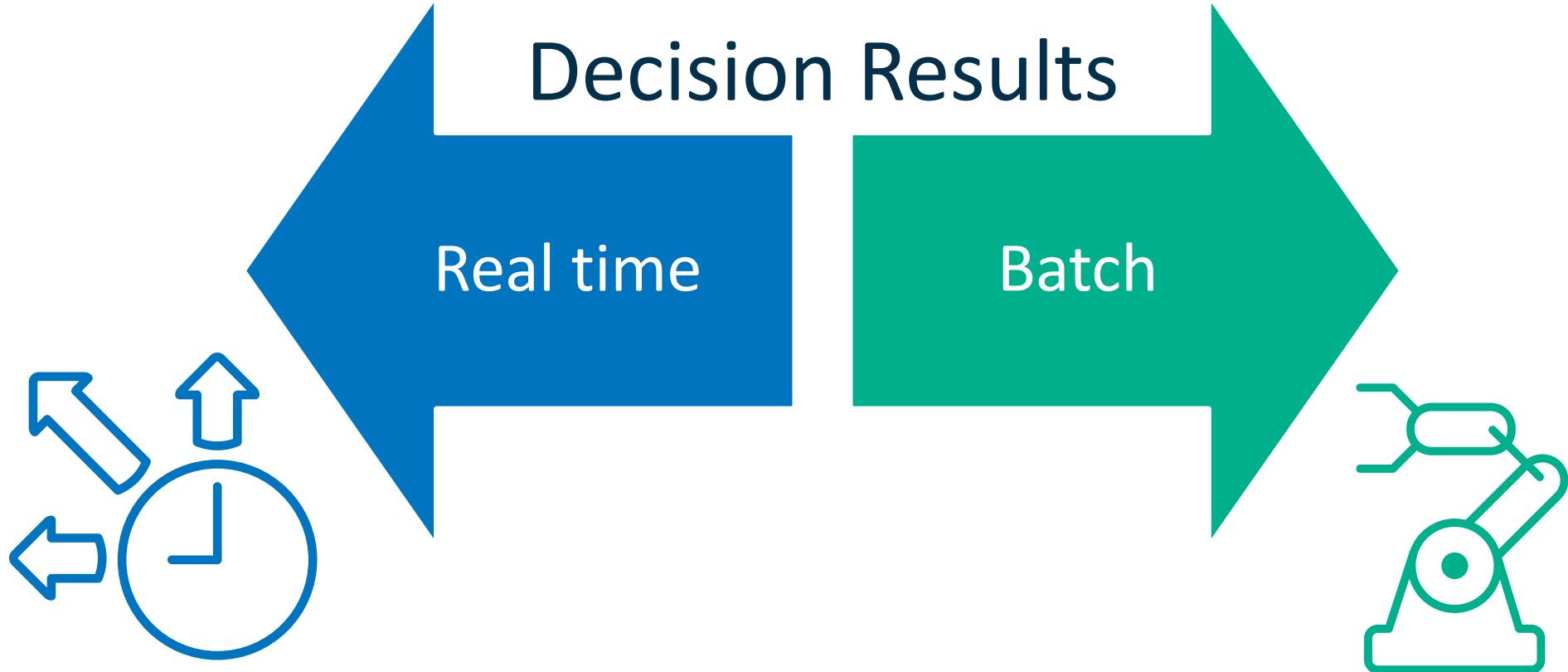


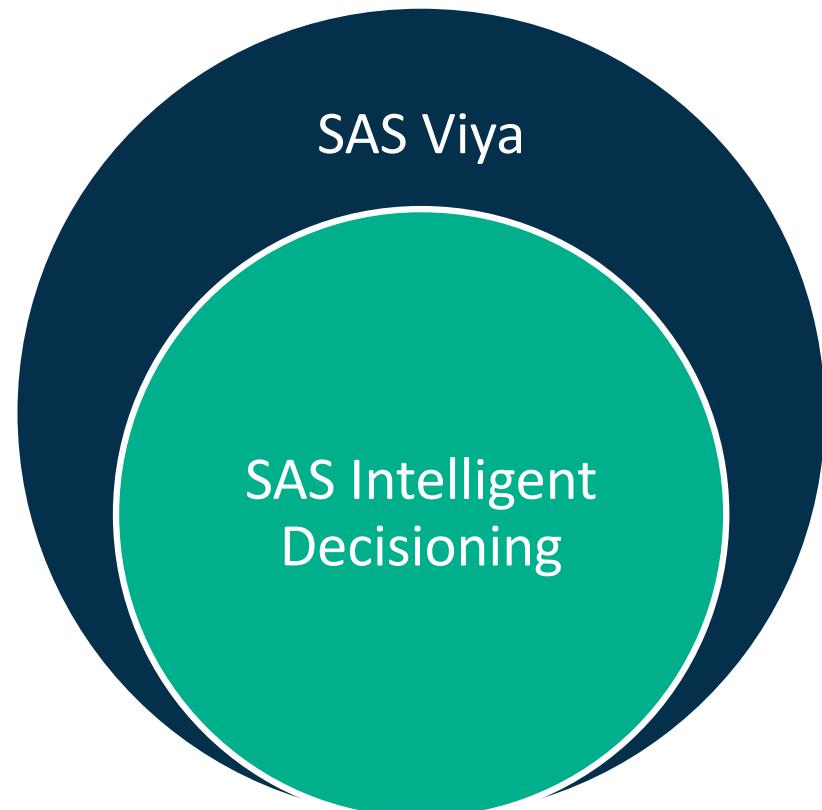
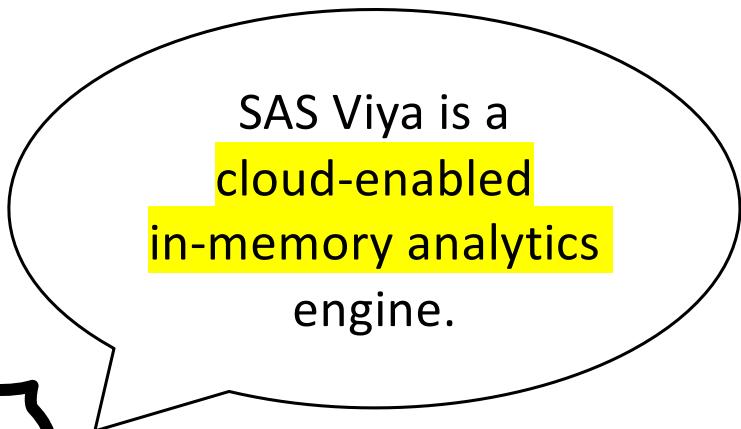
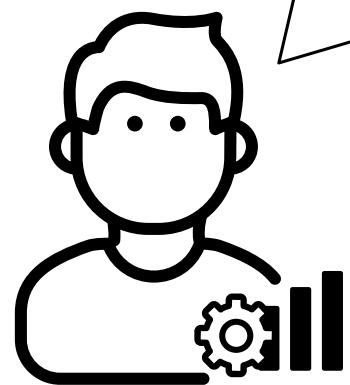
What is the next best action
for this customer?



How should we price
this product?

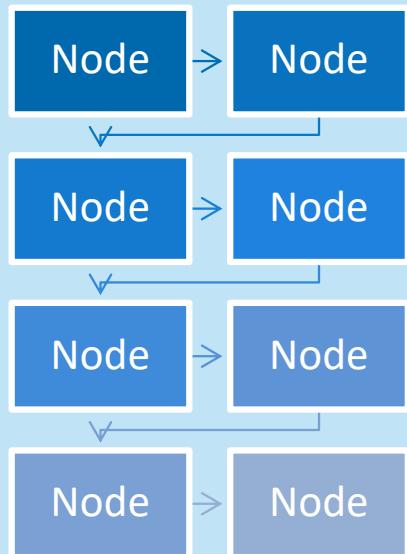




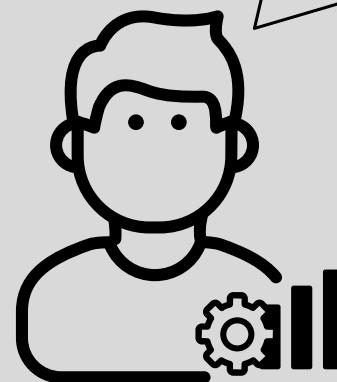


Design Environment: Step1 - Create decision flow

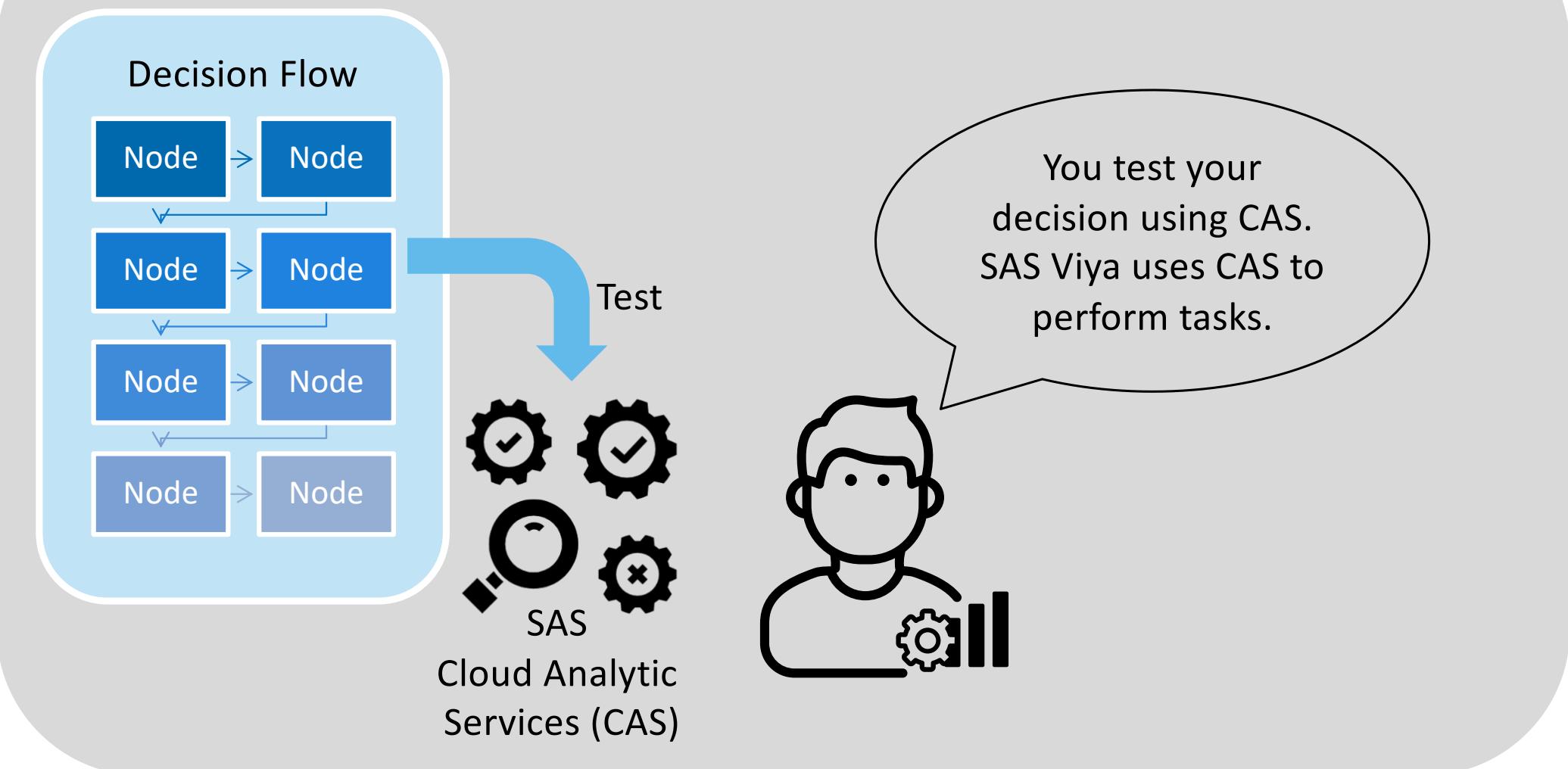
Decision Flow



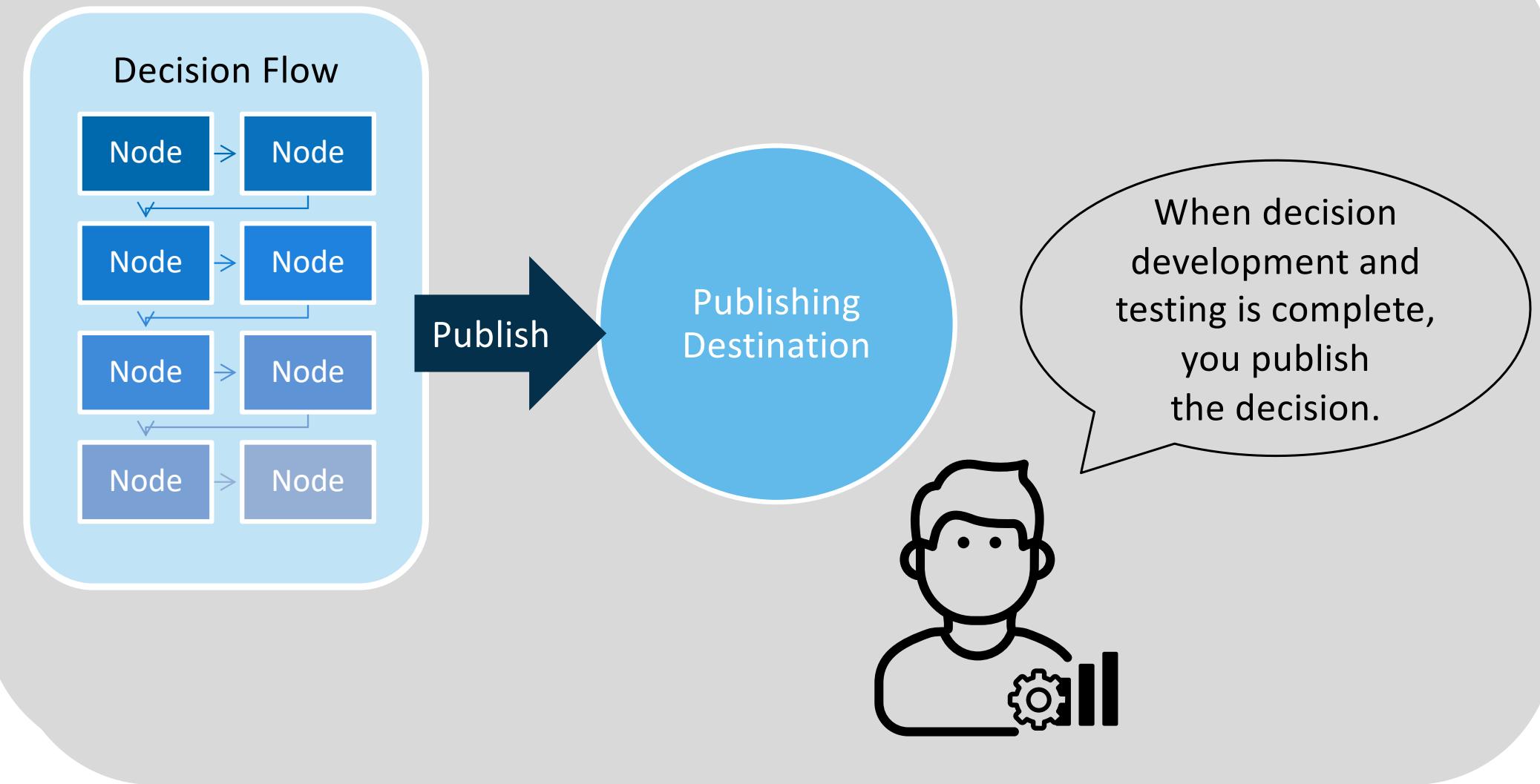
You add and
configure nodes that
perform different
types of processing.



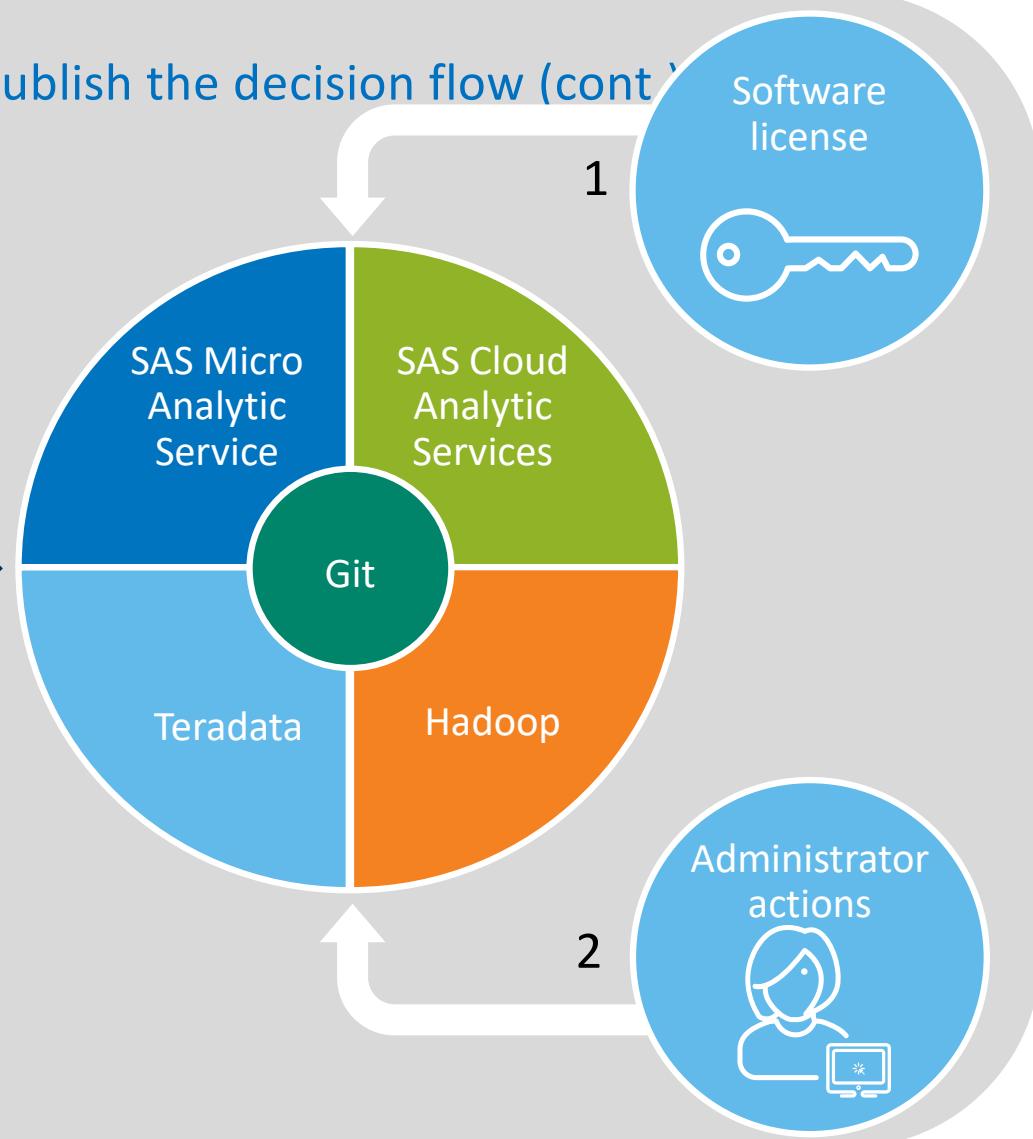
Design Environment : Step2 - test the decision flow



Design Environment: Step3 – Publish the decision flow



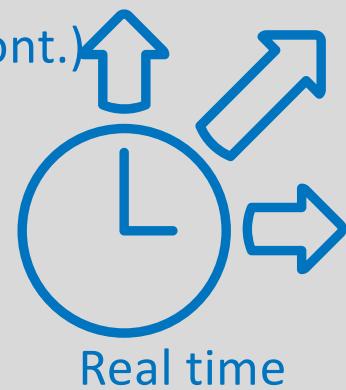
Design Environment: Step3 – Publish the decision flow (cont.)



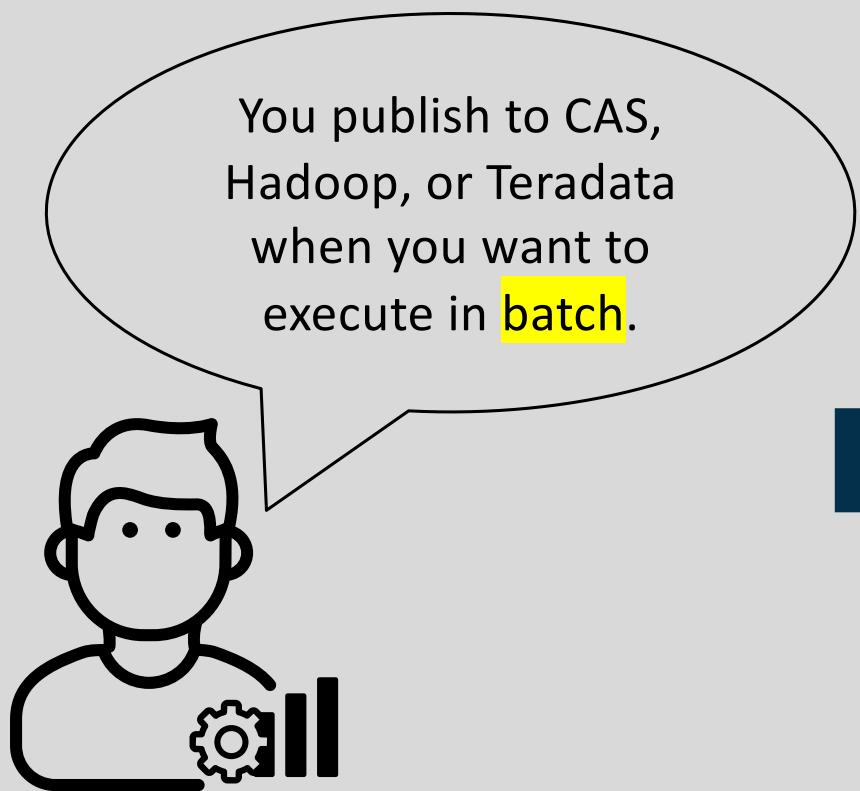
Design Environment: Step3 – Publish the decision flow (cont.)



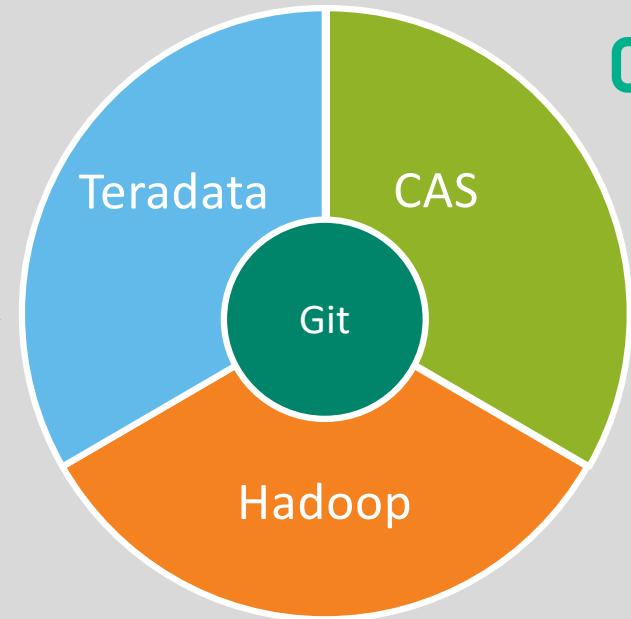
Publish



Design Environment: Step3 – Publish the decision flow (cont.)



→ Publish



Batch

Step4 – From development to the production environment

Design Environment
(Development)

Command-Line
Interface



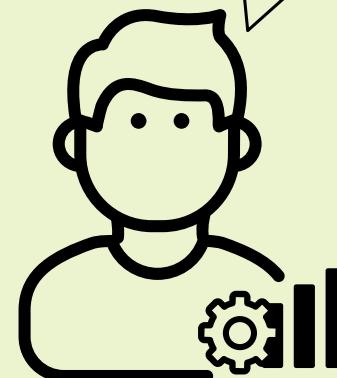
Published
decision

Custom Scripts

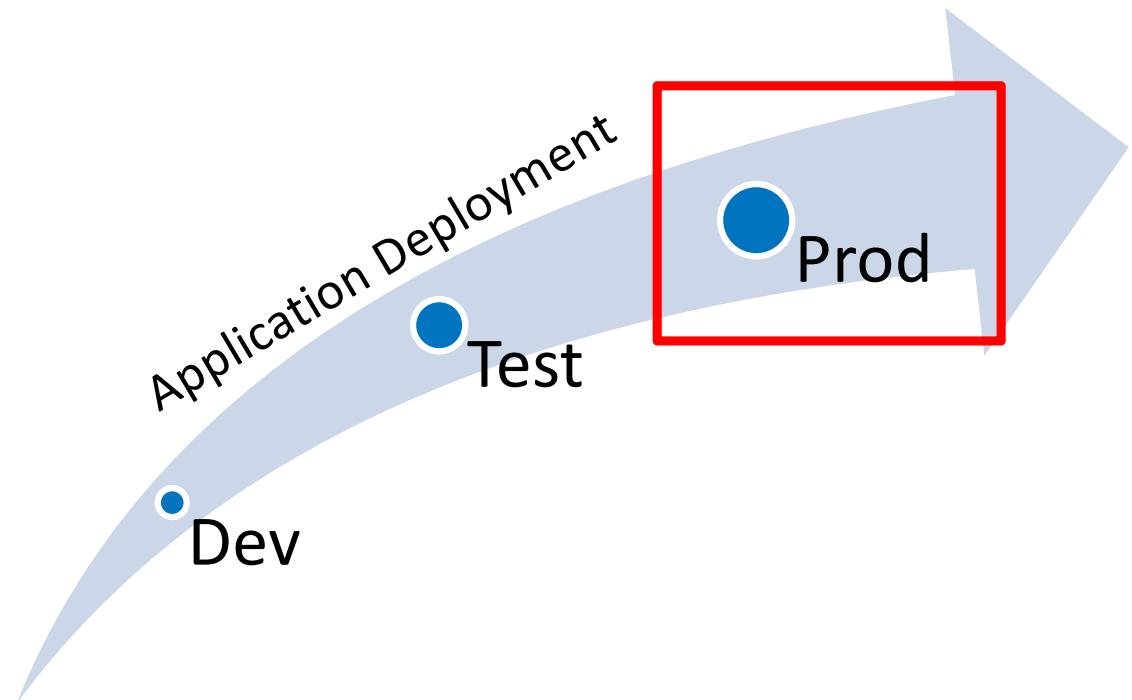


Deployment Environment
(Production)

Published
content is moved
to a deployment
environment.

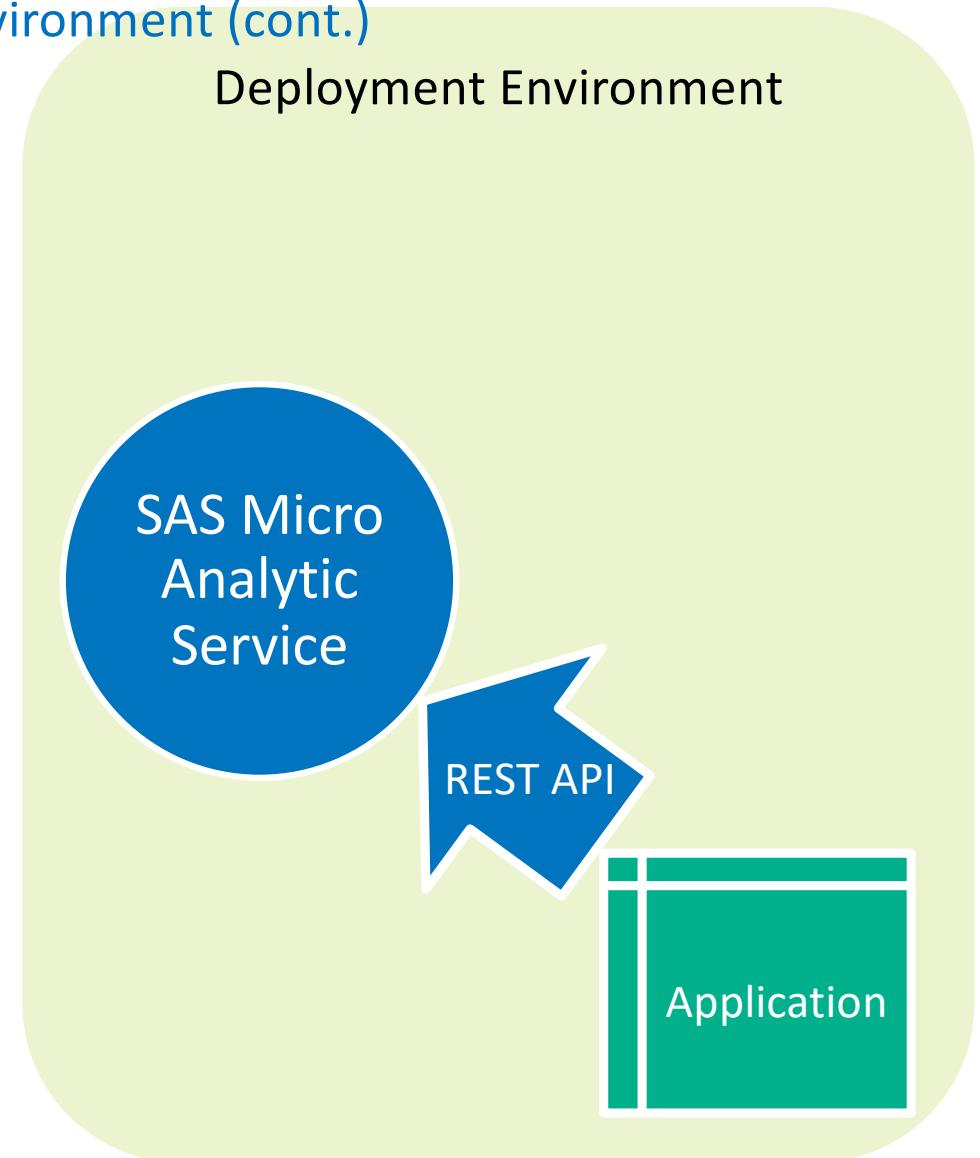
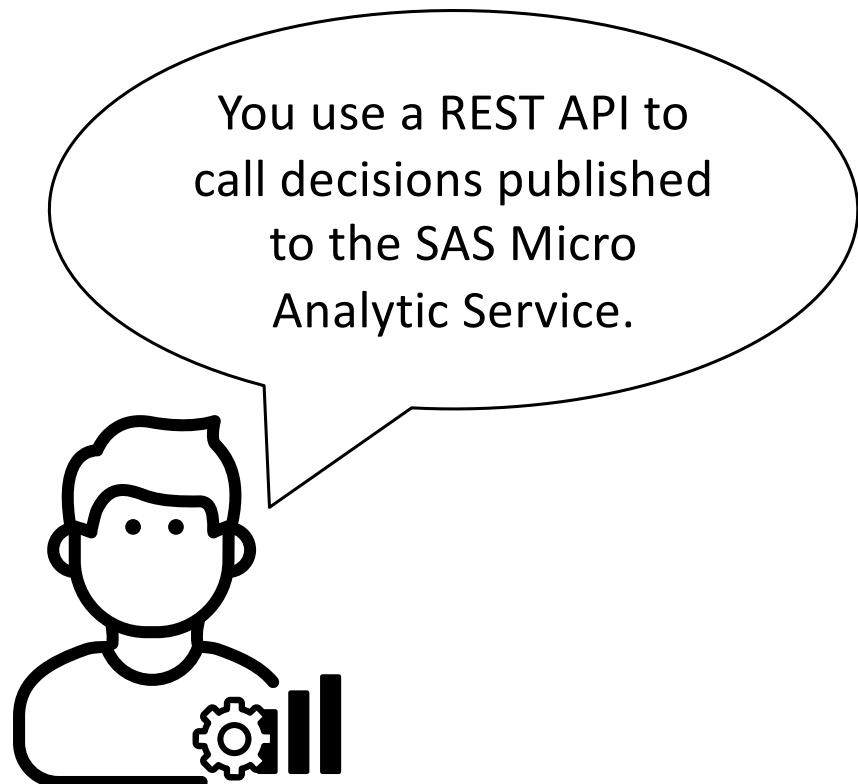


Step4 – From development to the production environment (cont.)



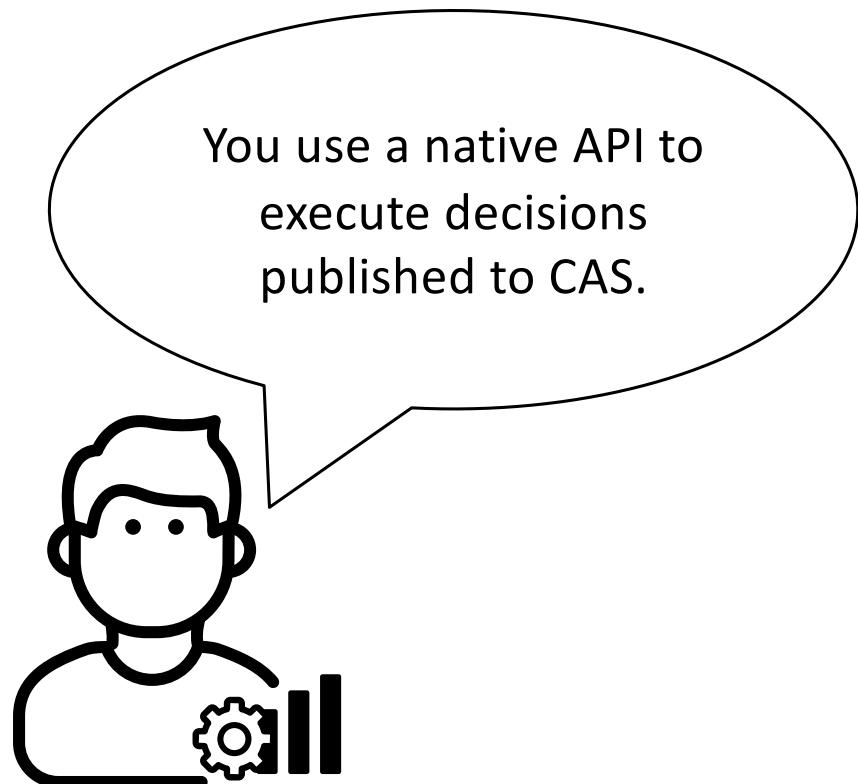
Step4 – From development to the **production** environment (cont.)

1) REST API for SAS MAS

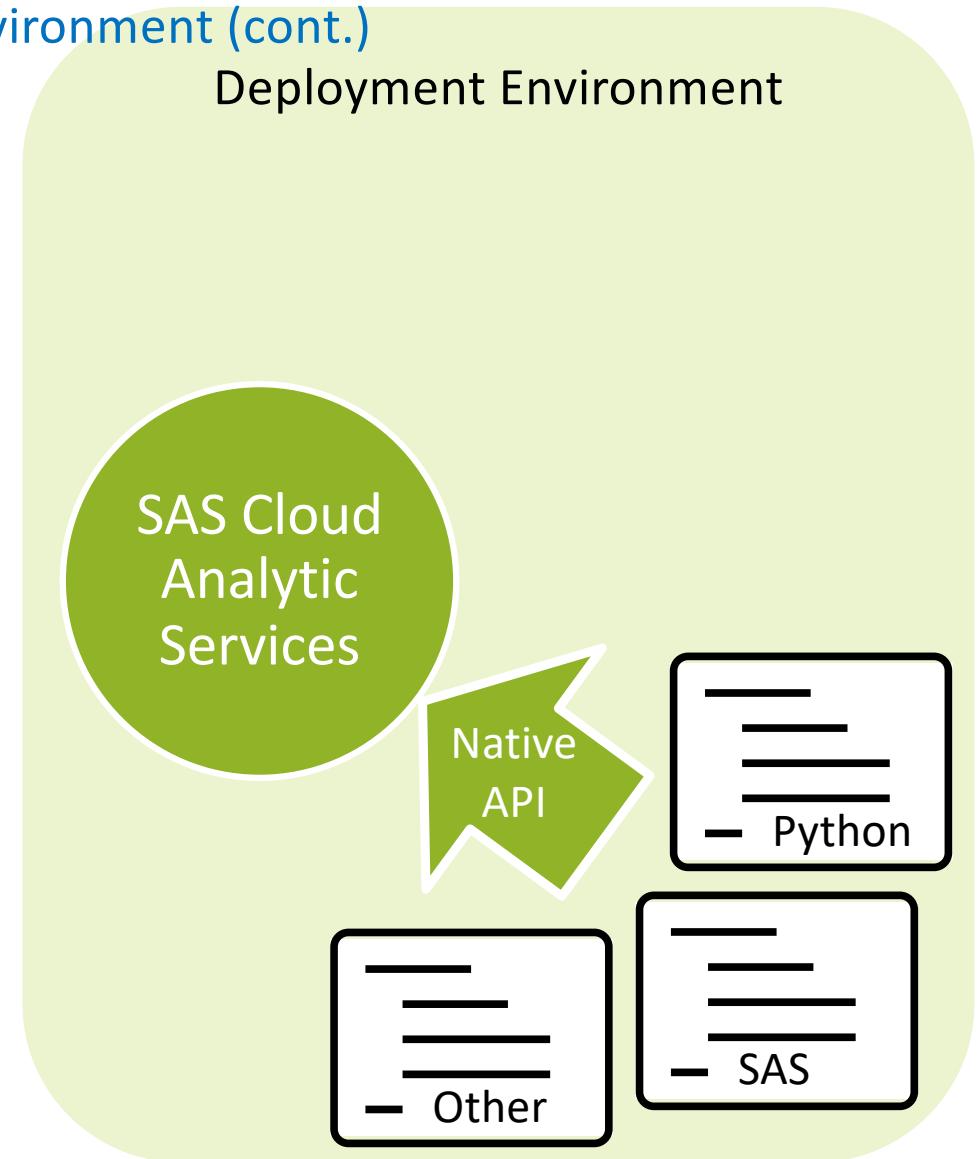


Step4 – From development to the production environment (cont.)

2) Native API for SAS CAS

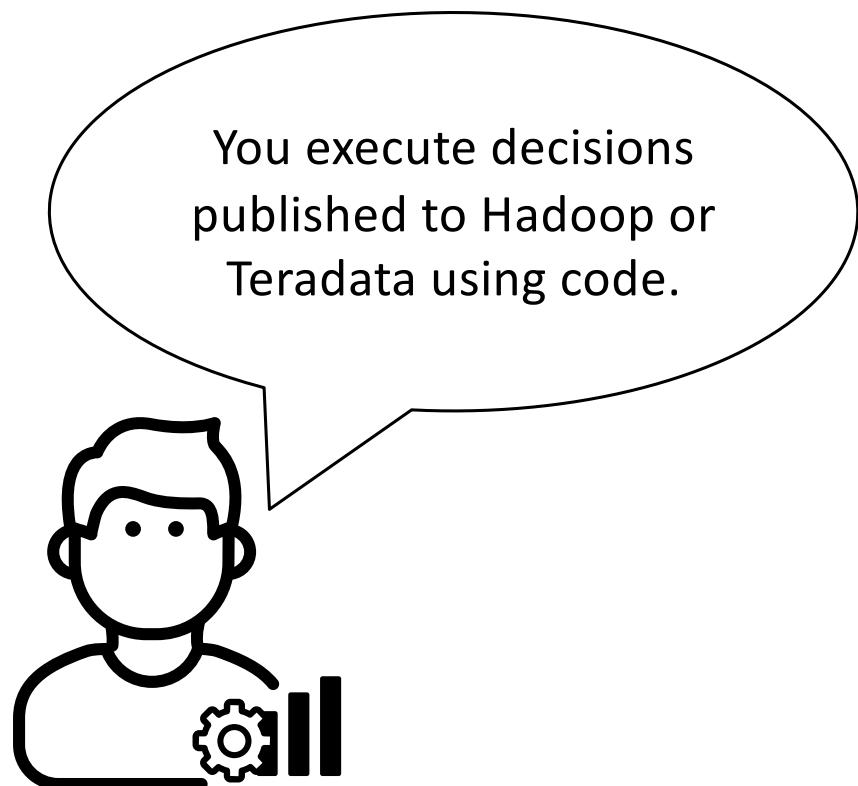


You use a native API to execute decisions published to CAS.

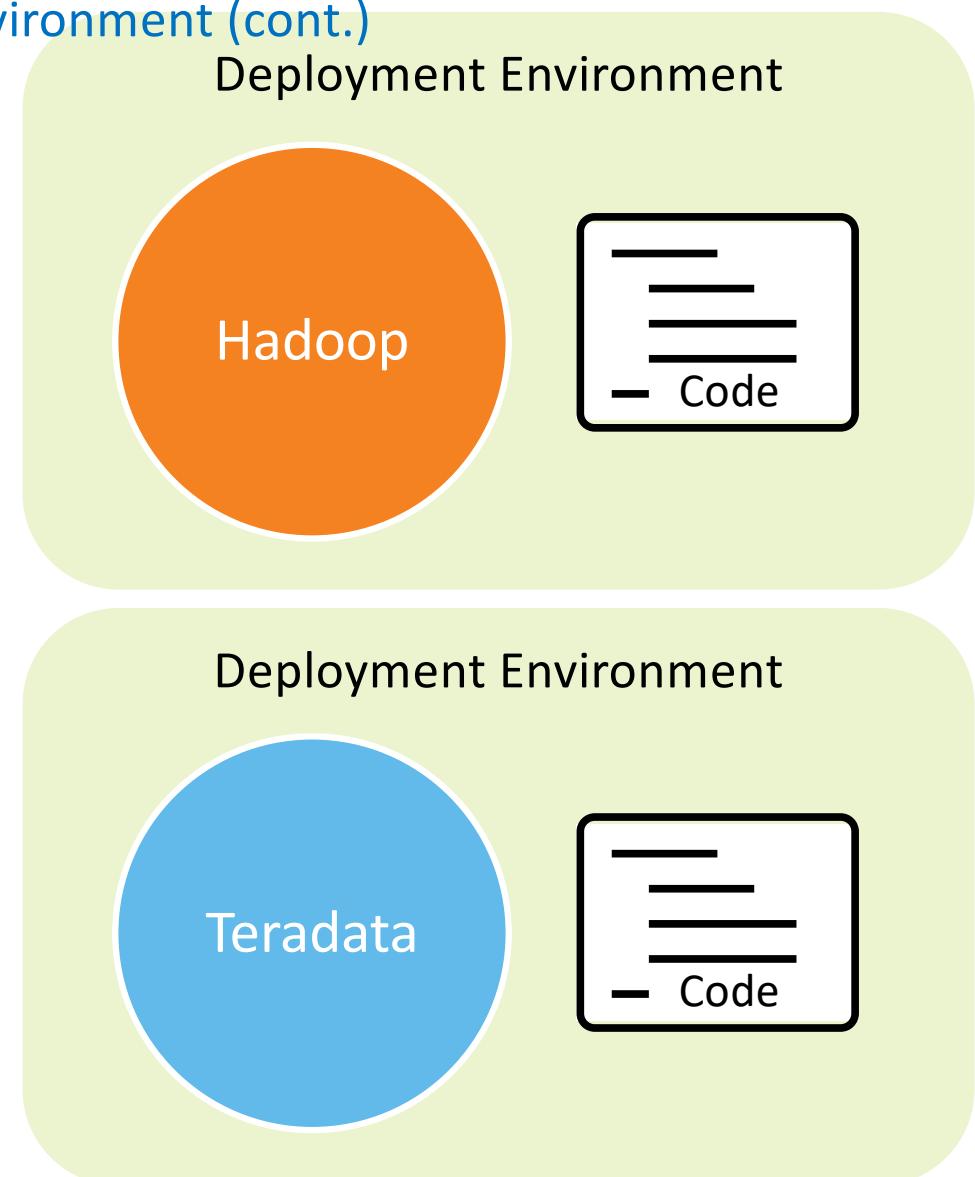


Step4 – From development to the **production** environment (cont.)

3) Code for other environments



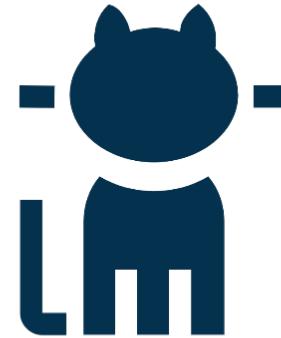
You execute decisions published to Hadoop or Teradata using code.



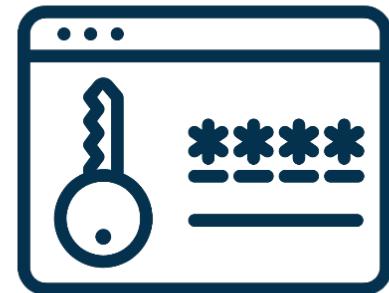
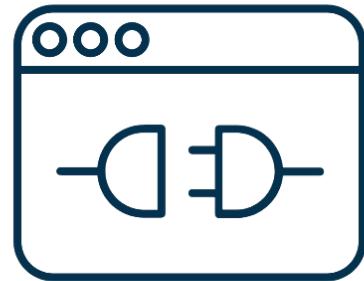
Git



GitLab



GitHub

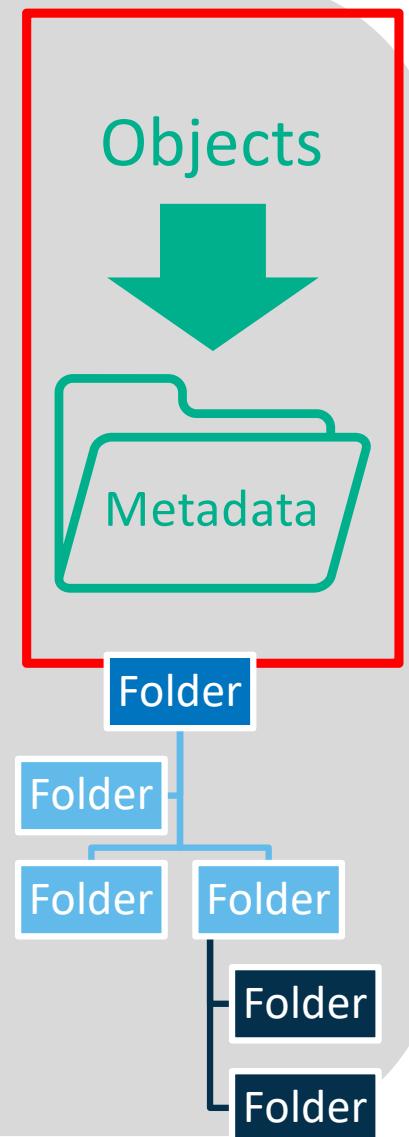
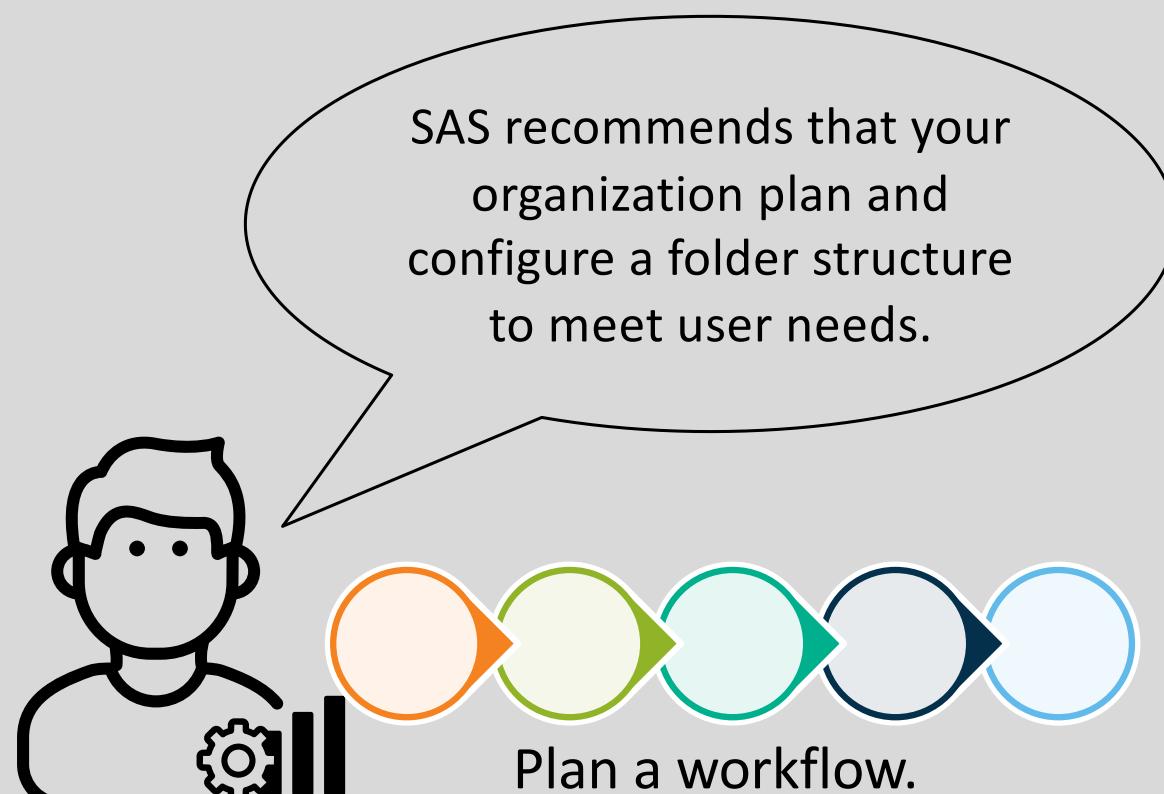


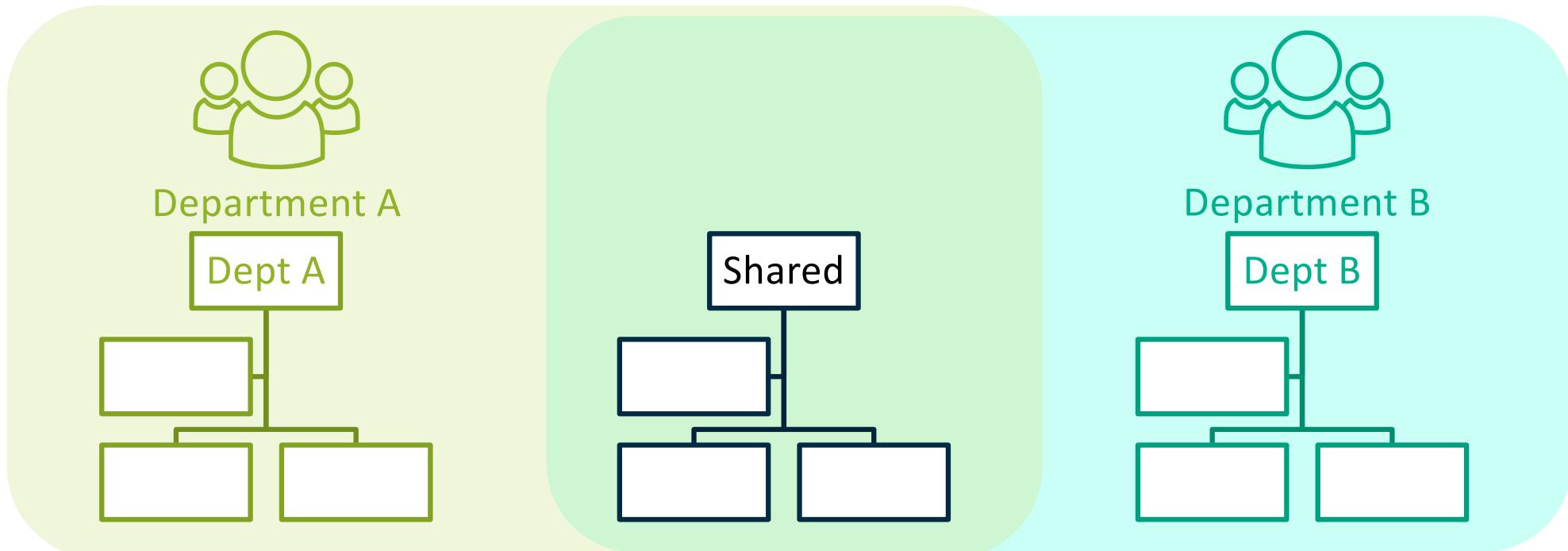
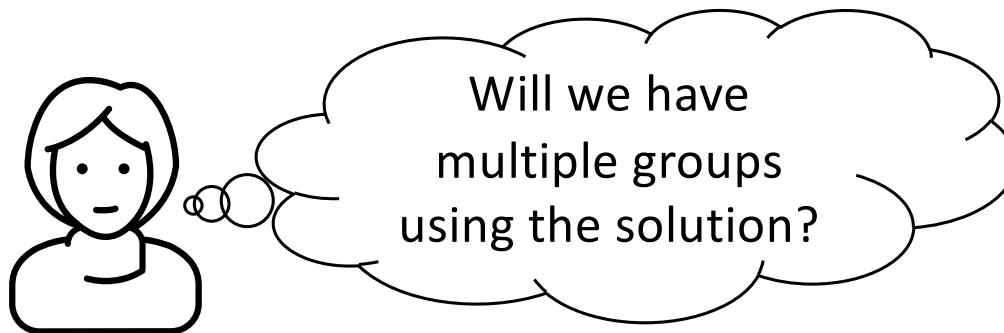
Configure Git for the Model
Publish service

Use the REST API to create
a credential domain

Store the credential
information

Design Environment: Let's implement the decision flow

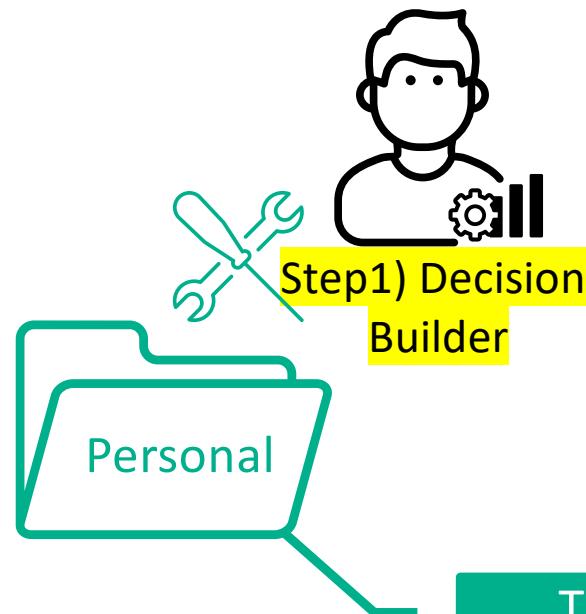
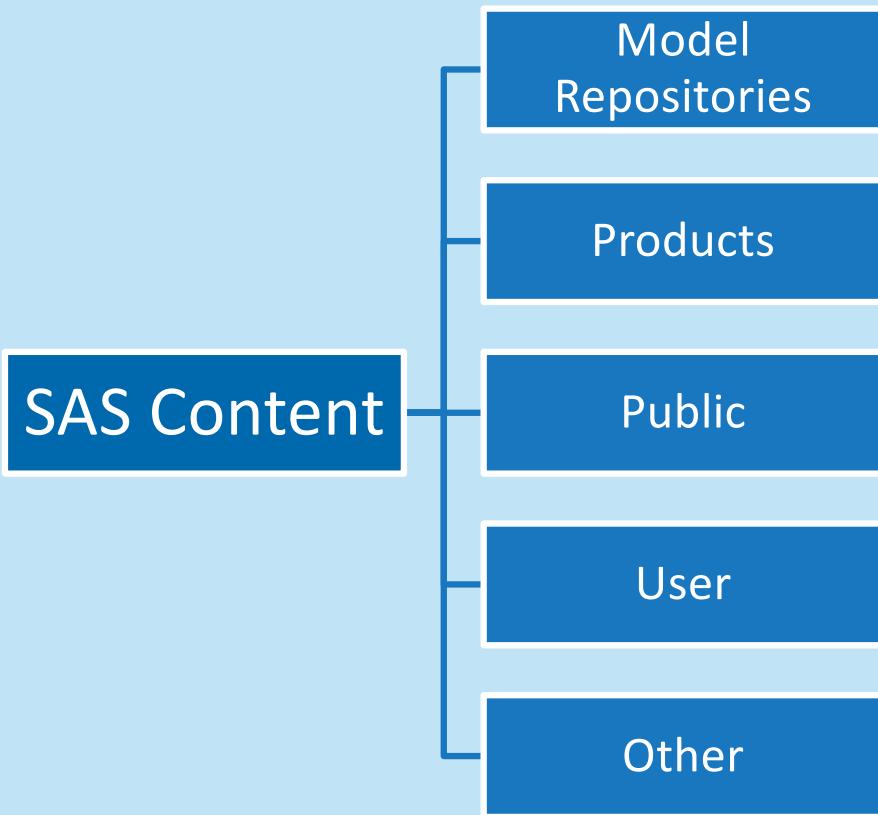




Plan a workflow.



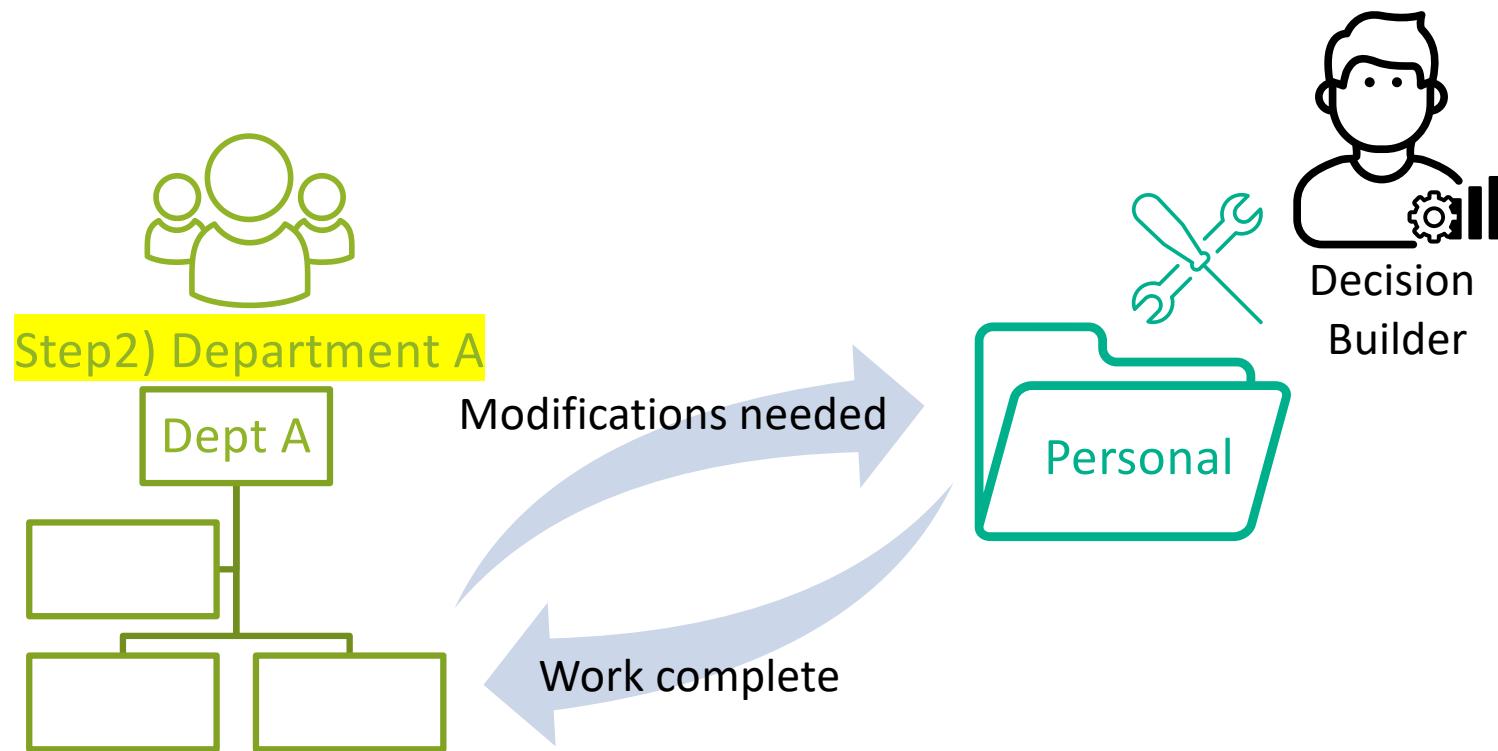
System-Created Folders

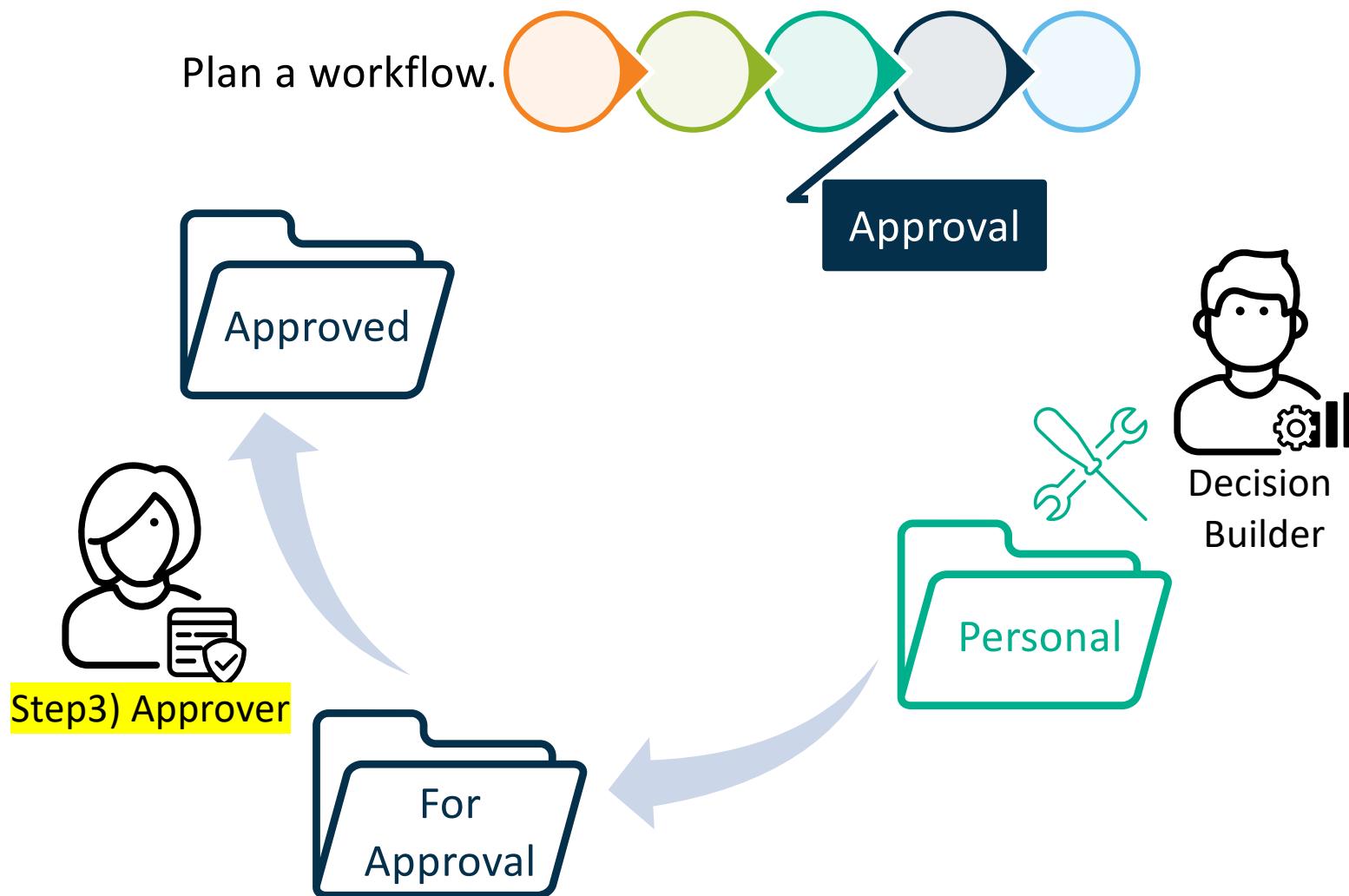


Step1) Decision
Builder

The My Folder
location or another
limited access folder

Plan a workflow.

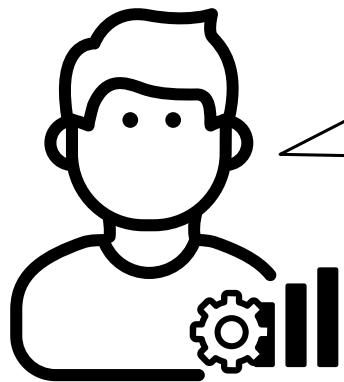




Lesson 1: SAS Intelligent Decisioning: The Big Picture

1.1 Introduction to SAS Intelligent Decisioning

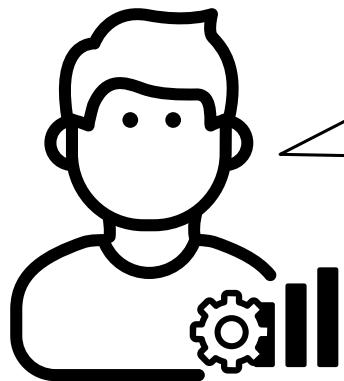
1.2 Introduction to Decisions



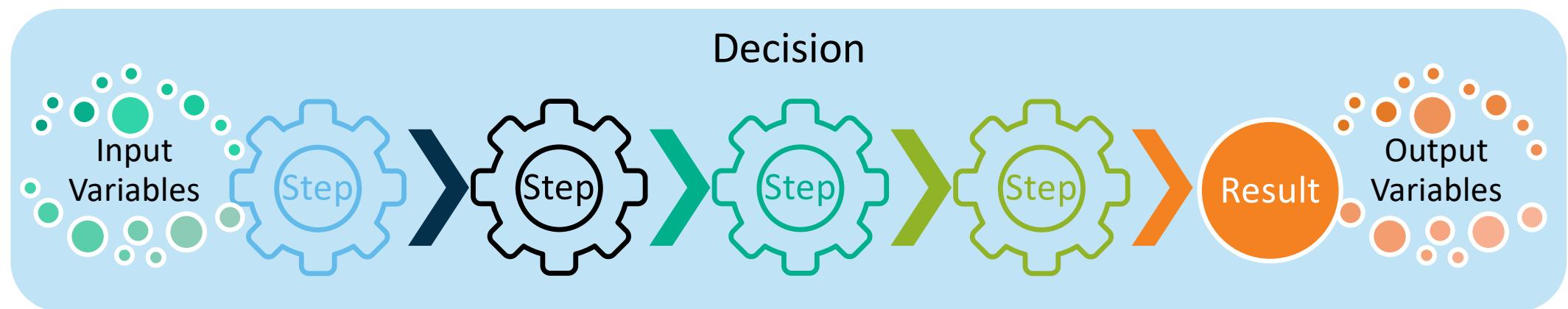
A decision enables you to
combine steps that
perform many different
types of processing to
return a **result**.

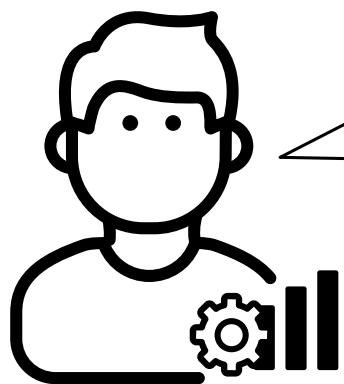
Decision



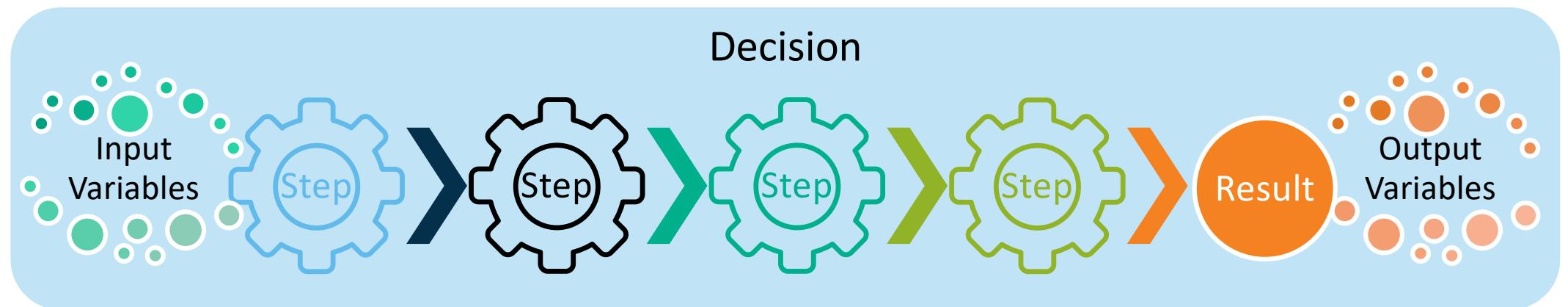


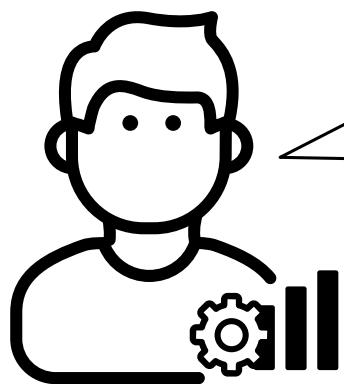
A decision accepts **input variables** and returns **output variables**.



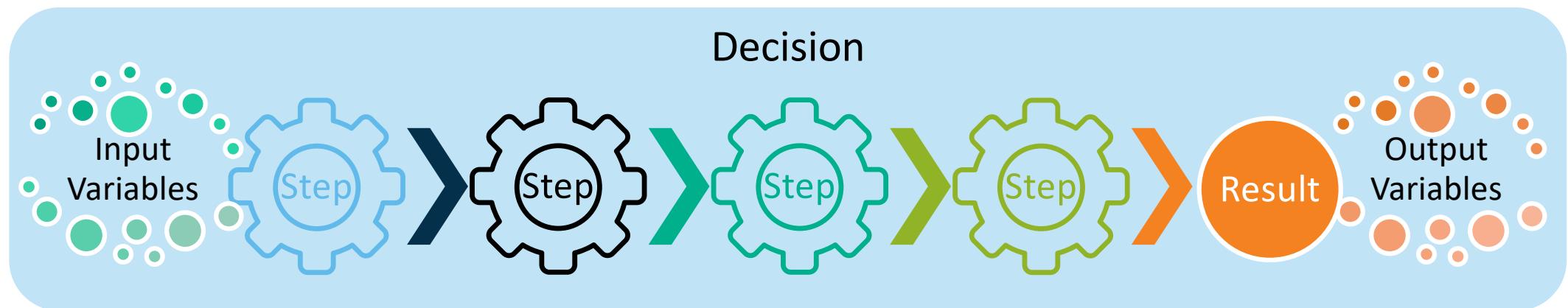


1) You **define** the
processing steps and the
variables.
(for each step)

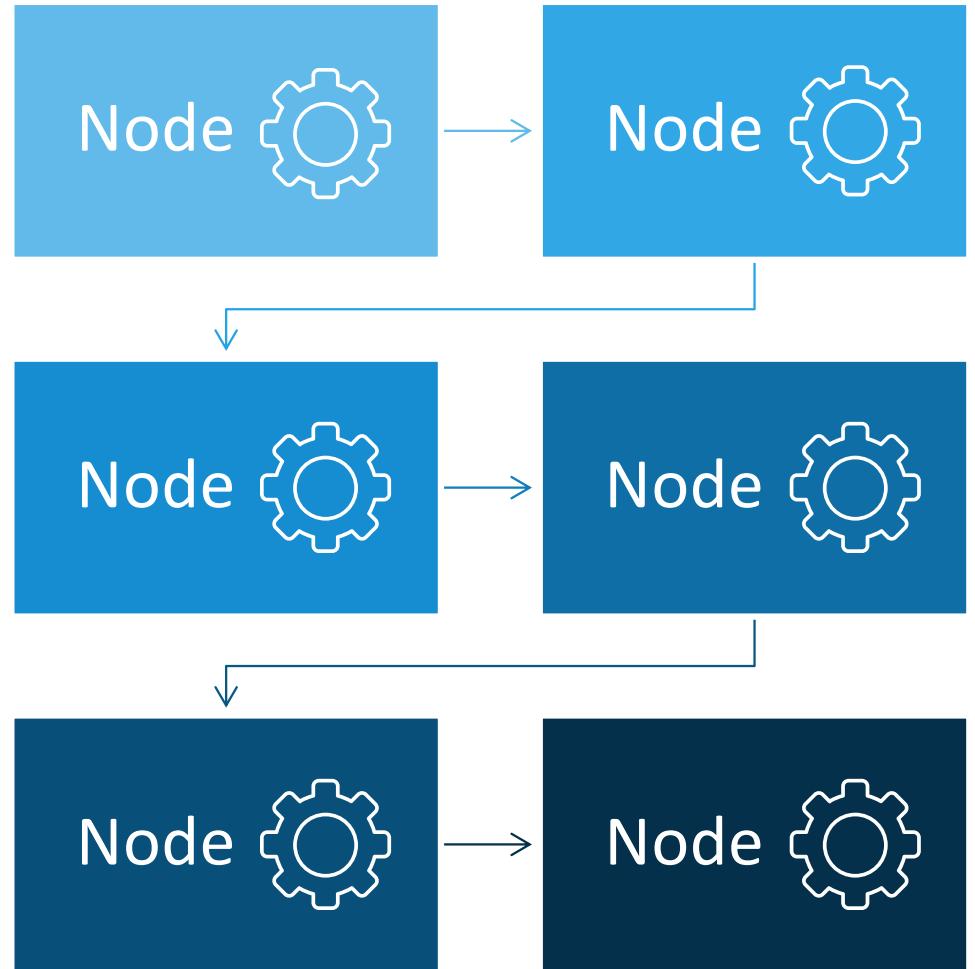
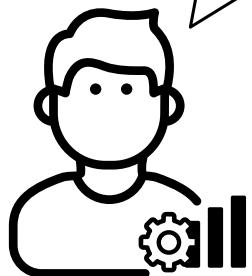


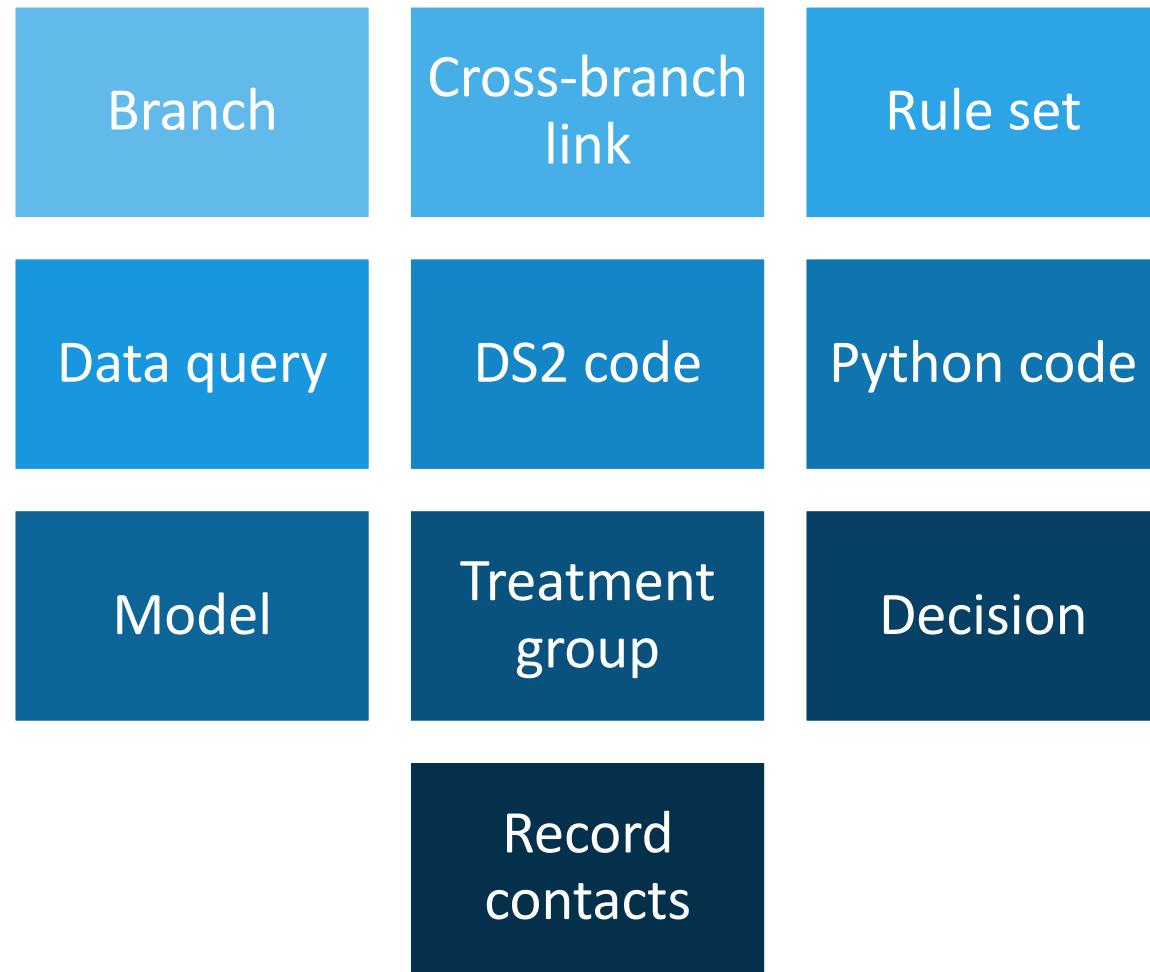
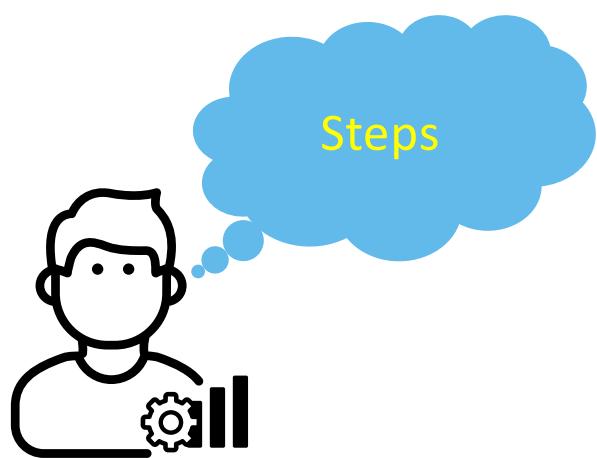


2) You must determine what **inputs** are available and what **outputs** are required.
(for each step)



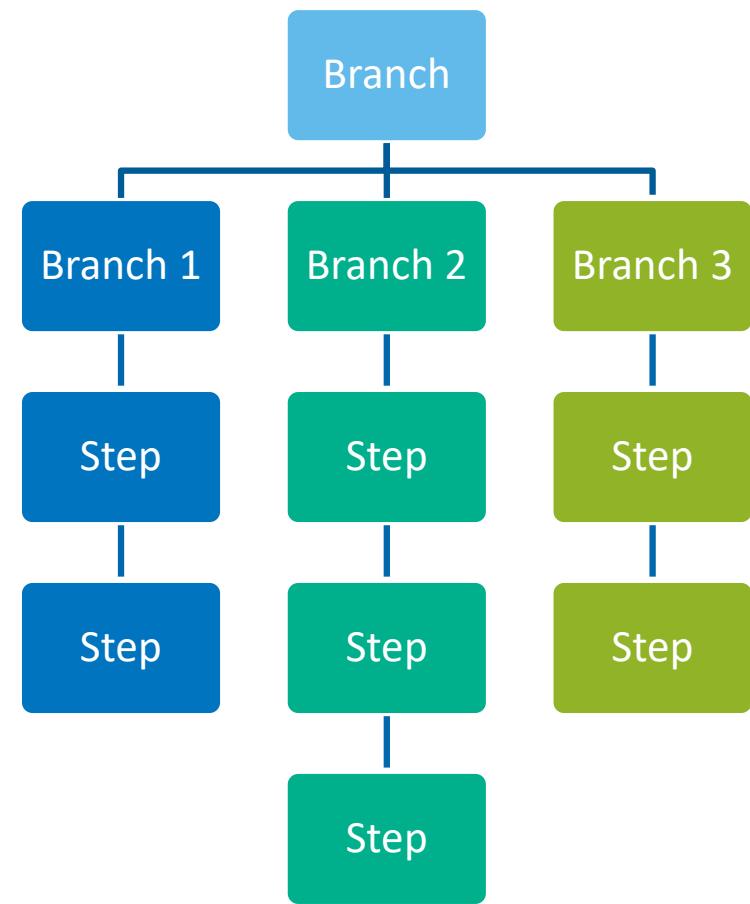
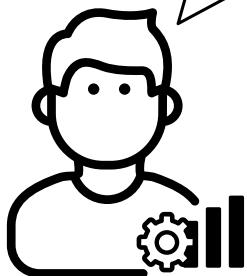
3) You define the steps that you want to perform by adding and configuring nodes that perform different types of processing.
(for each step)





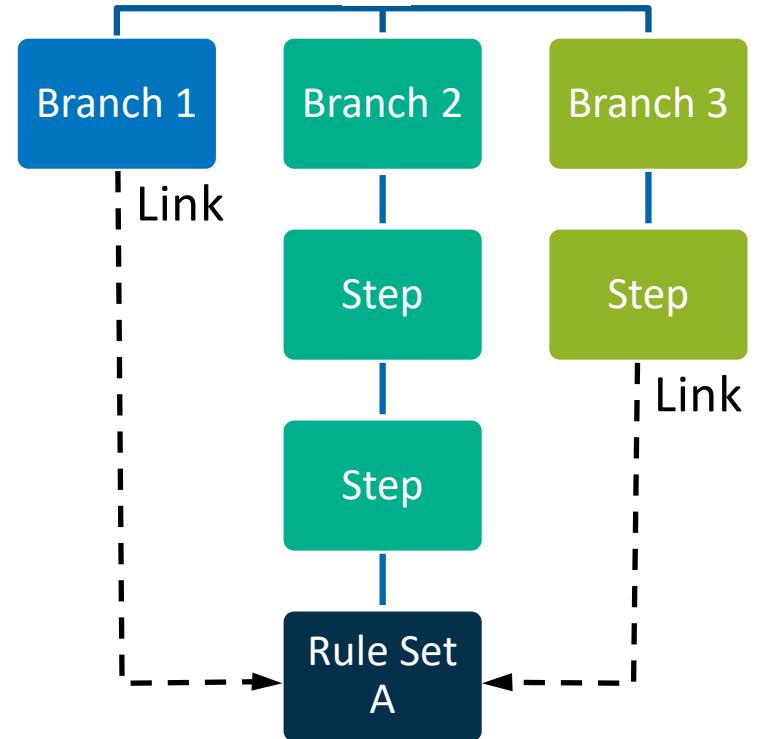
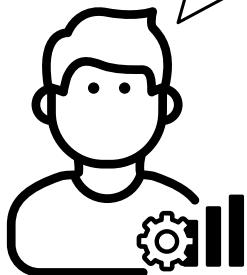
1) Branch

You use branch nodes to create separate paths in the decision flow.

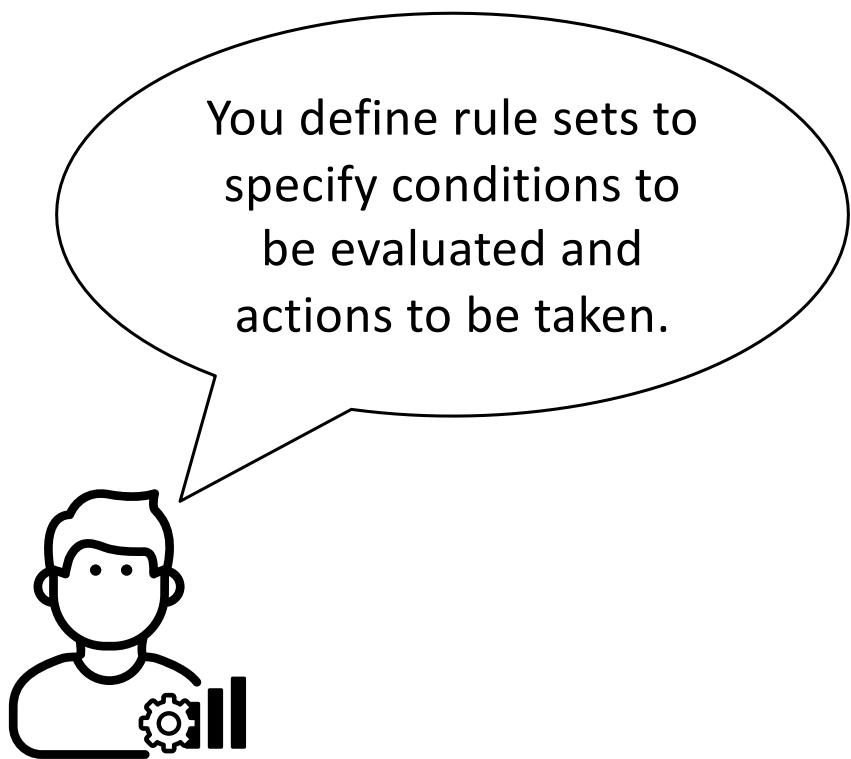


2) Cross-Branch Link

You use cross-branch links to easily reuse components across branches.



3) Rule set



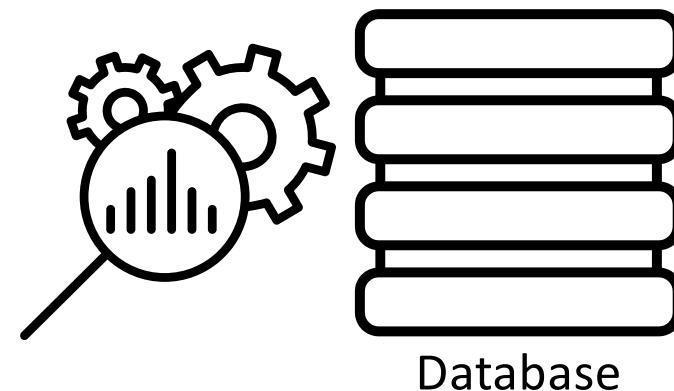
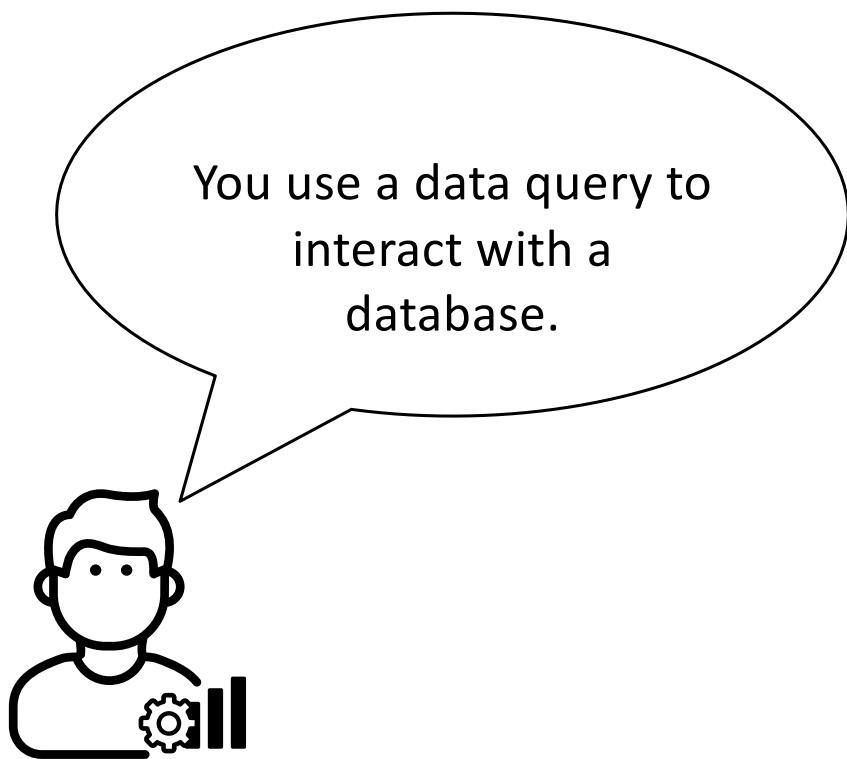
Rule

Condition

Action

Condition + Action

4) Data query

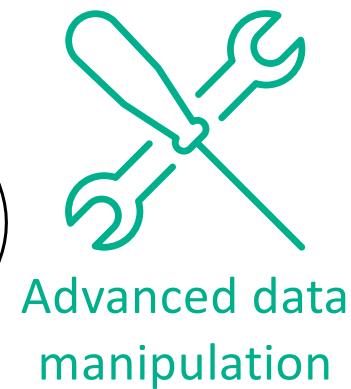


Database

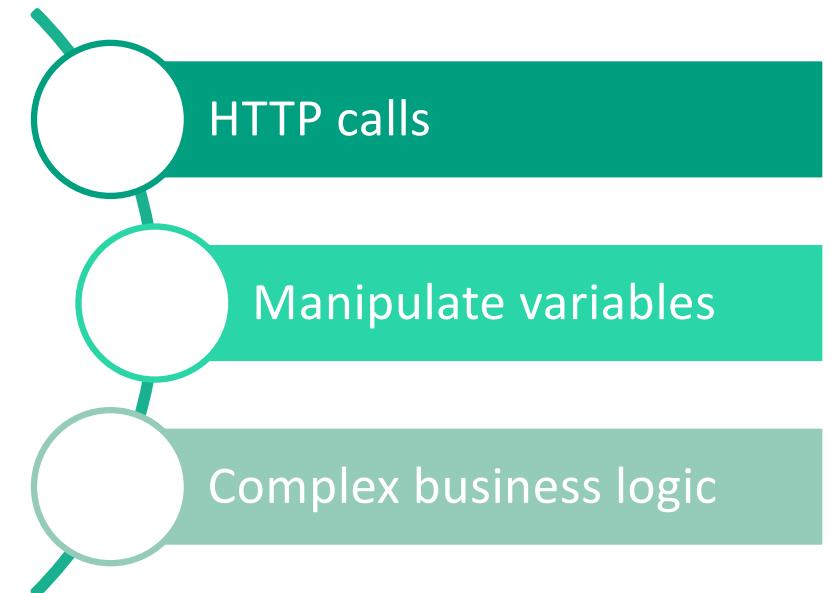
5) DS2 code



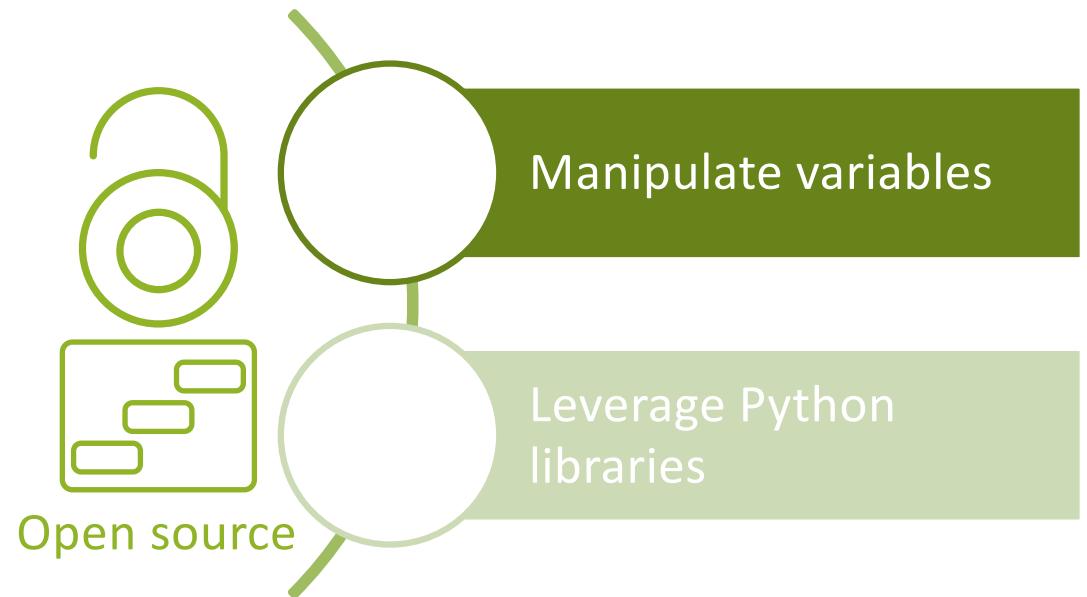
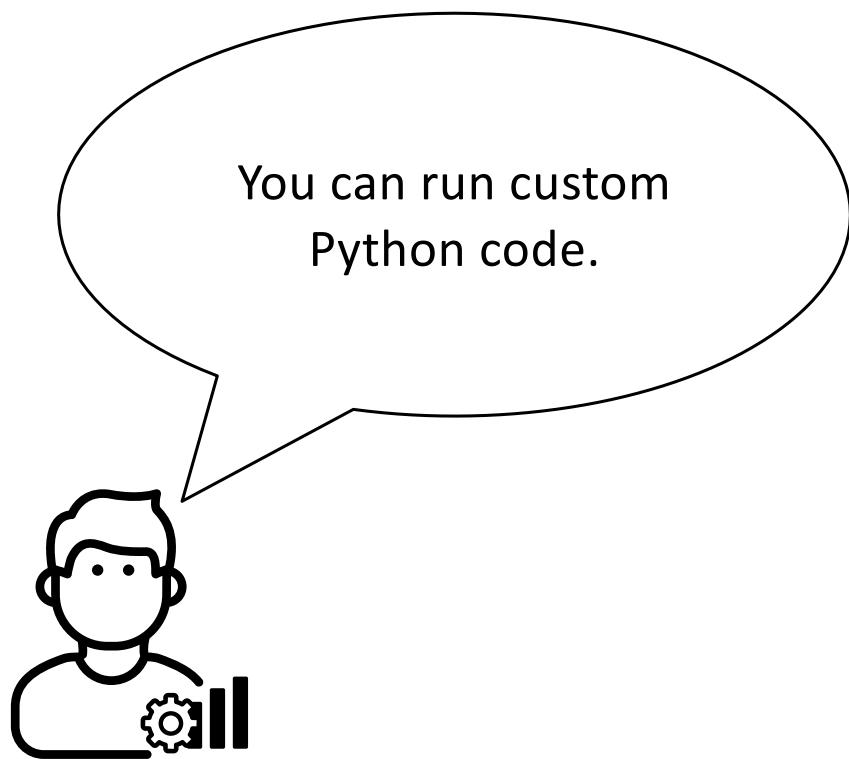
You can run custom DS2 code.



Advanced data manipulation

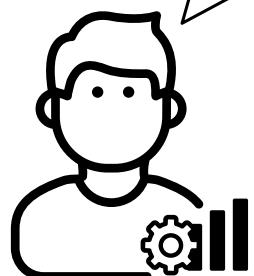


6) Python code



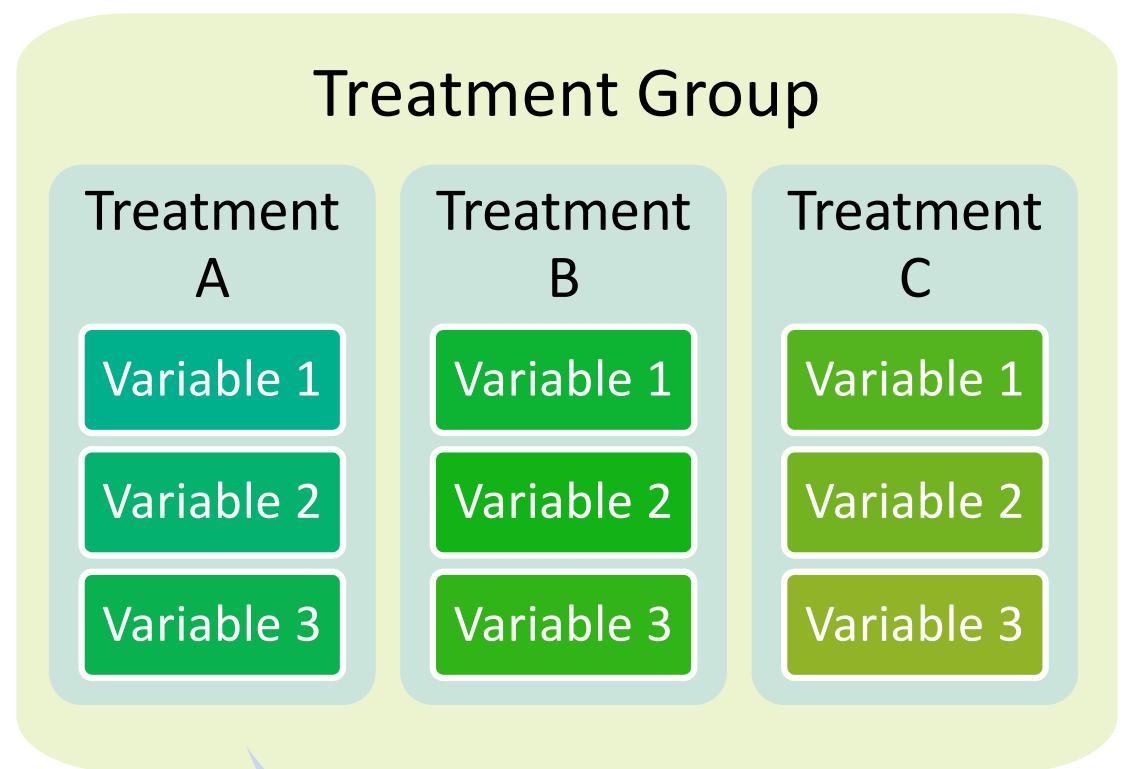
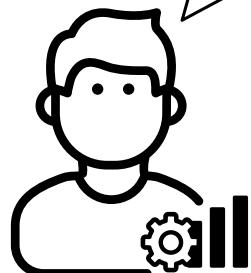
7) Model

You can return a result from a SAS or Python analytical model.



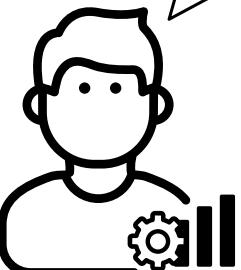
8) Treatment group

Treatments groups define detailed variables for specific decision outcomes.



Specific decision outcomes

9) Decision



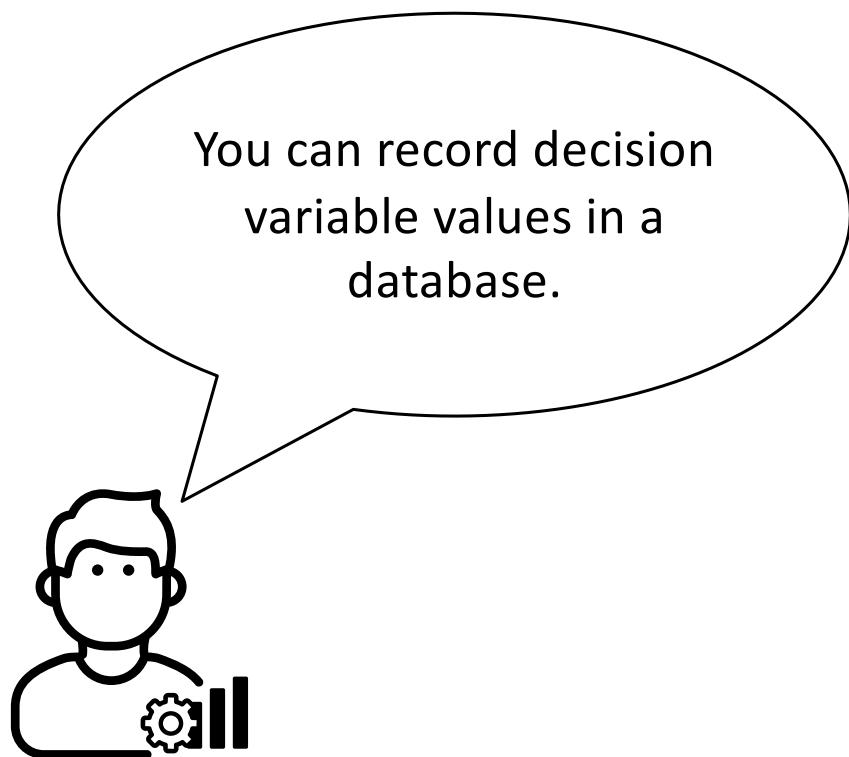
You can call another decision from within a decision.

Decision 1

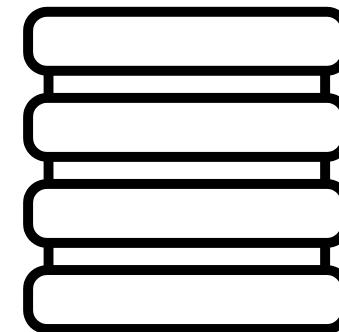


Decision 2

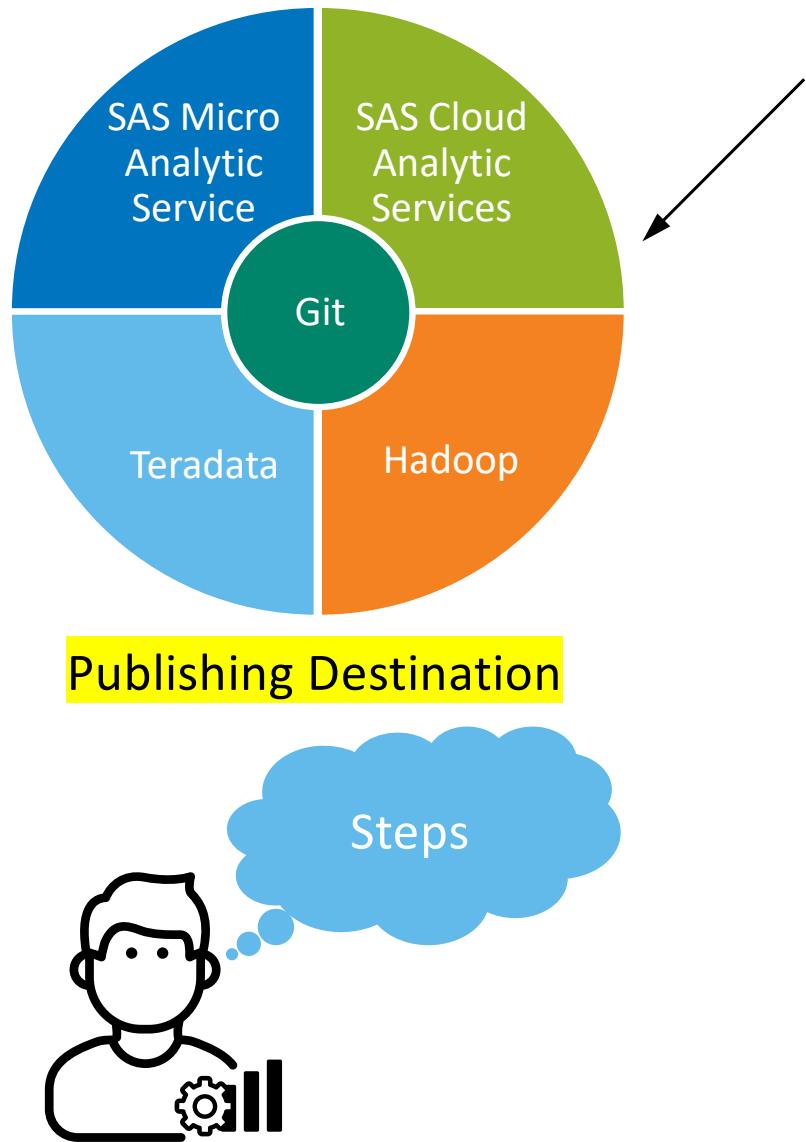
10) Record contacts



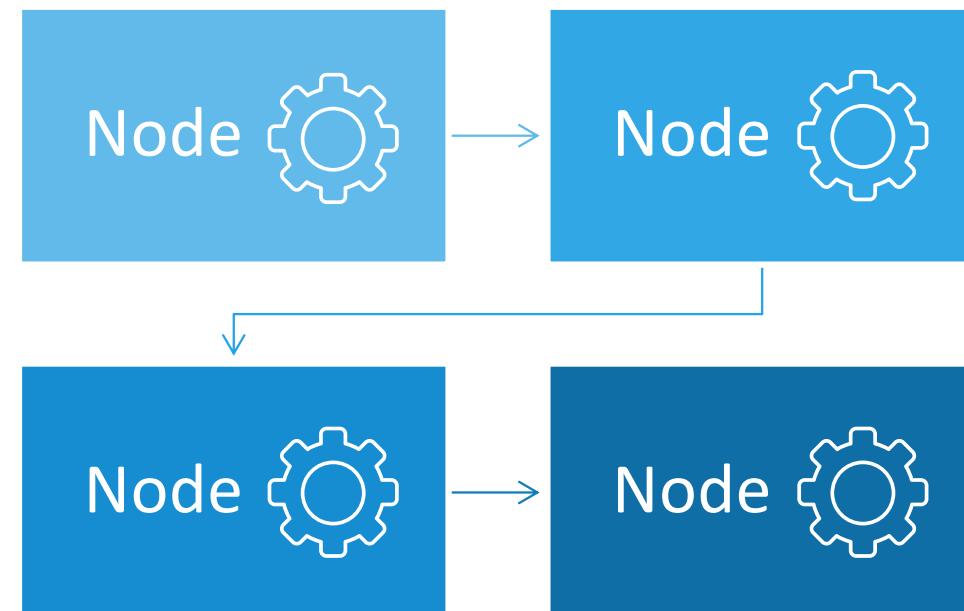
Decision variables



Database

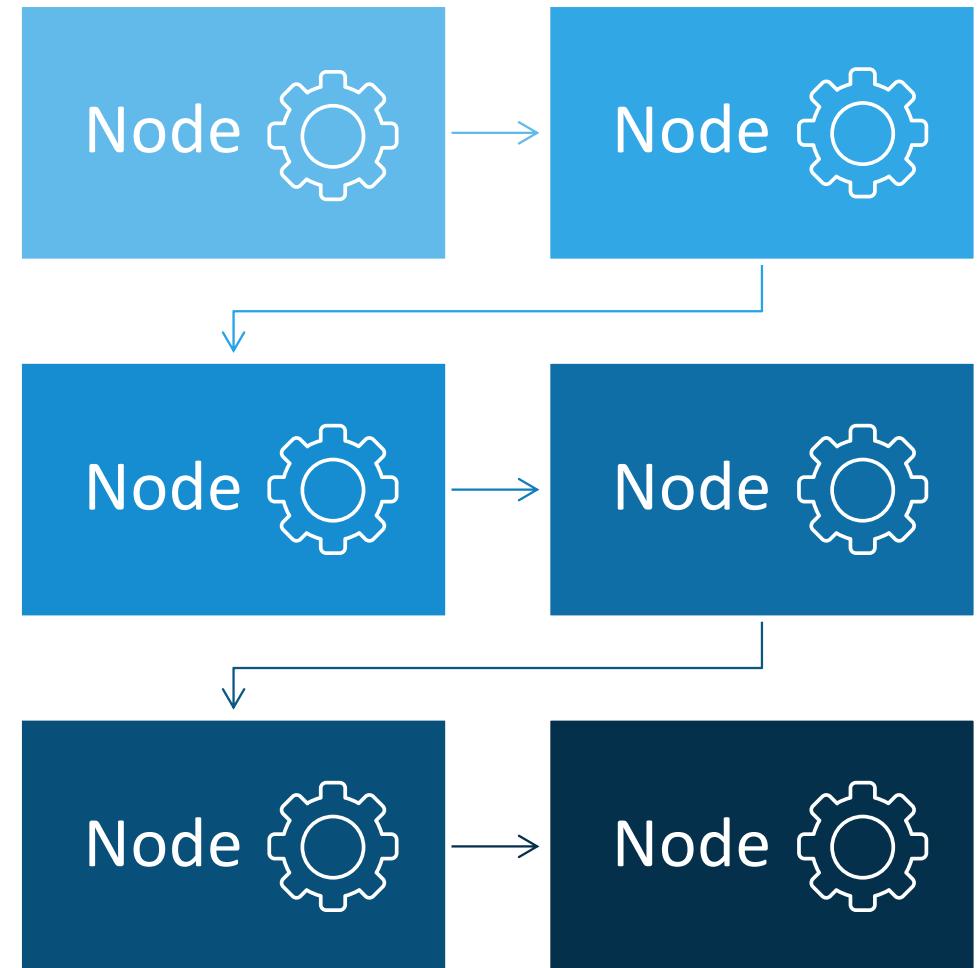


Important consideration
when choosing and
configuring nodes.





Example: Provide different offers





Determine whether customer is eligible for upgrade.



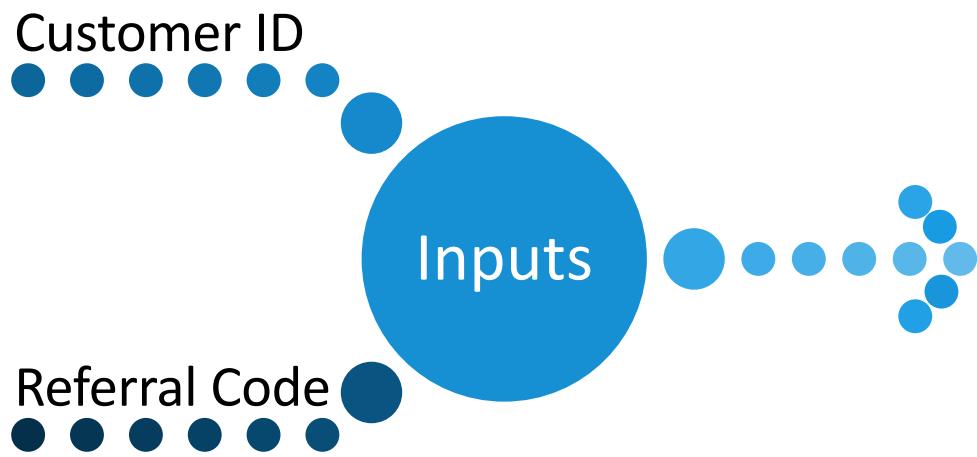
Assign possible offers to eligible customers.



Apply a discount to the offer prices, if appropriate.



Return the one best offer to each customer.



?

Determine whether customer is eligible for upgrade.

✳️

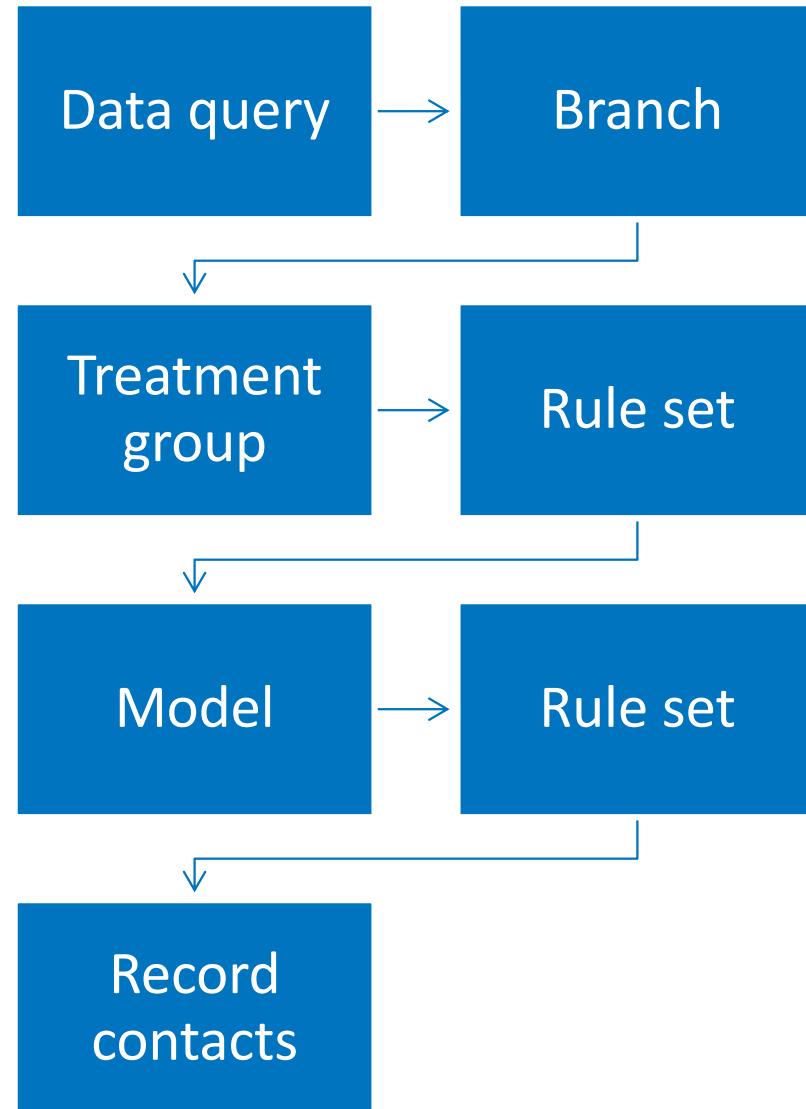
Assign possible offers to eligible customers.

\$

Apply a discount to the offer prices, if appropriate.

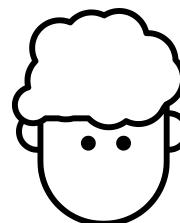
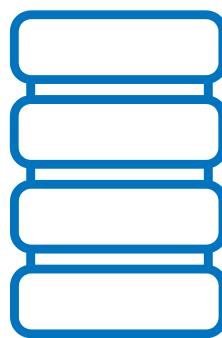
🤝

Return the one best offer to each customer.



Data query

CustomerID



Customer data

Retrieved Information



- Eligible for Upgrade
- Phone Model
- Plan Type
- Age
- Gender
- ...and more

?

Determine whether customer is eligible for upgrade.

★

Assign possible offers to eligible customers.

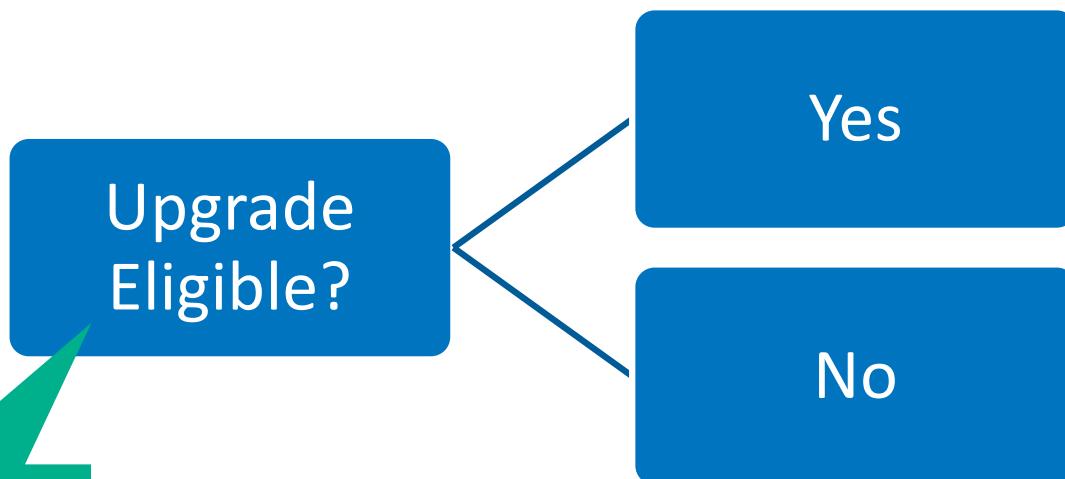
\$

Apply a discount to the offer prices, if appropriate.

🤝

Return the one best offer to each customer.

Branch



Retrieved by
data query.

?

Determine whether customer is eligible for upgrade.

⊕

Assign possible offers to eligible customers.

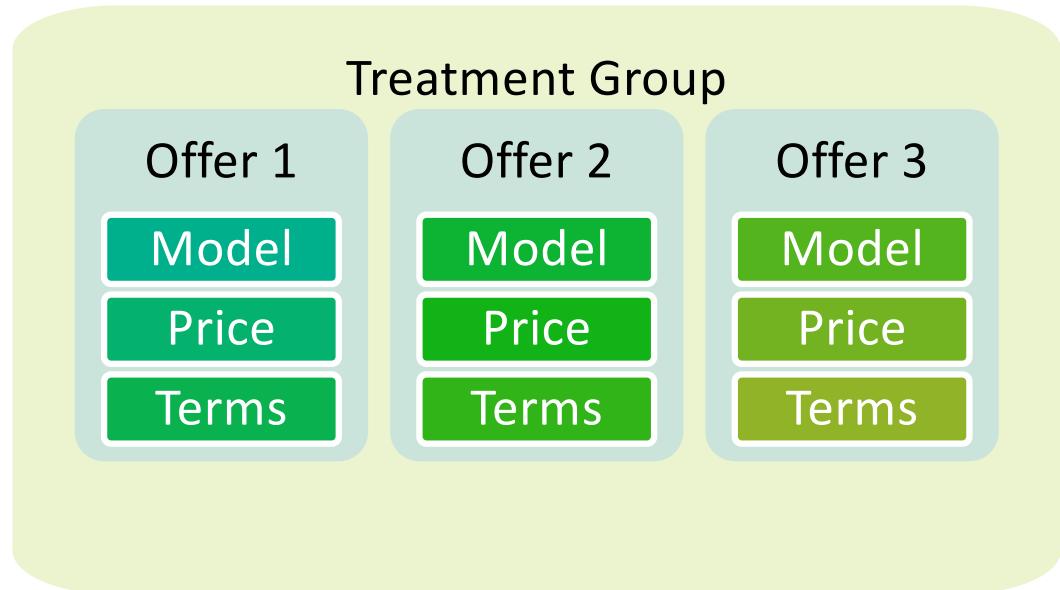
\$

Apply a discount to the offer prices, if appropriate.

🤝

Return the one best offer to each customer.

Upgrade Eligible=Yes



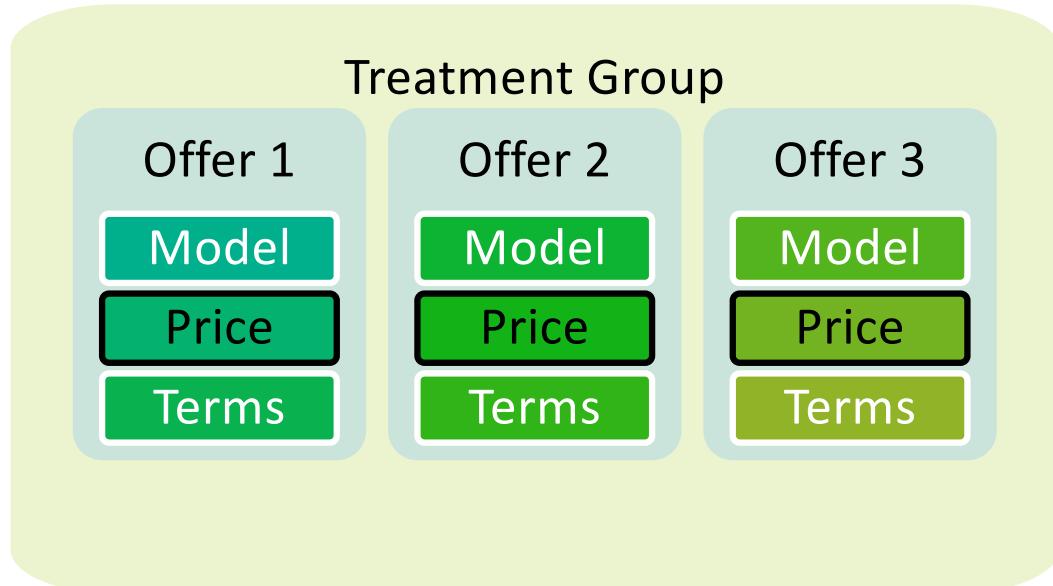
① Determine whether customer is eligible for upgrade.

② Assign possible offers to eligible customers.

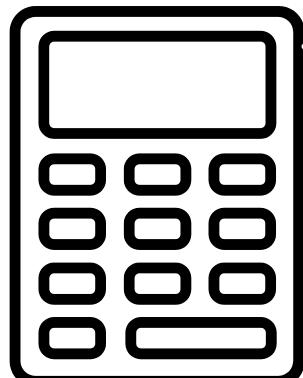
③ Apply a discount to the offer prices, if appropriate.

④ Return the one best offer to each customer.

Upgrade Eligible=Yes



Referral Code



Price = Price x %

?

Determine whether customer is eligible for upgrade.

?

Assign possible offers to eligible customers.

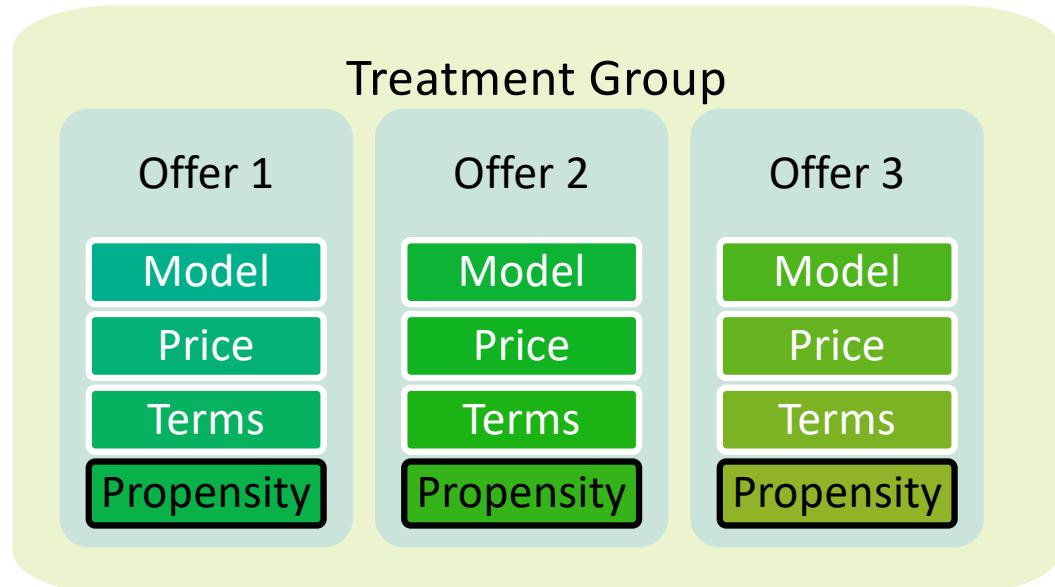
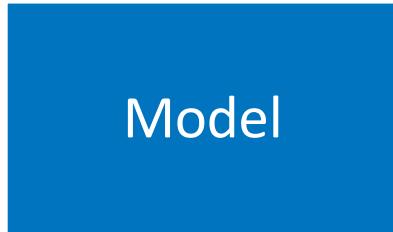
?

Apply a discount to the offer prices, if appropriate.

?

Return the one best offer to each customer.

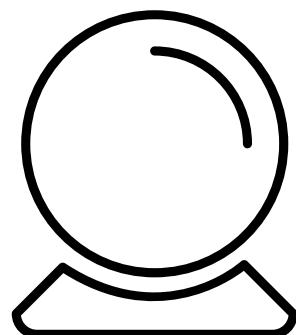
Upgrade Eligible=Yes



Current plan details

Demographics

Predicted propensity



?

Determine whether customer is eligible for upgrade.

?

Assign possible offers to eligible customers.

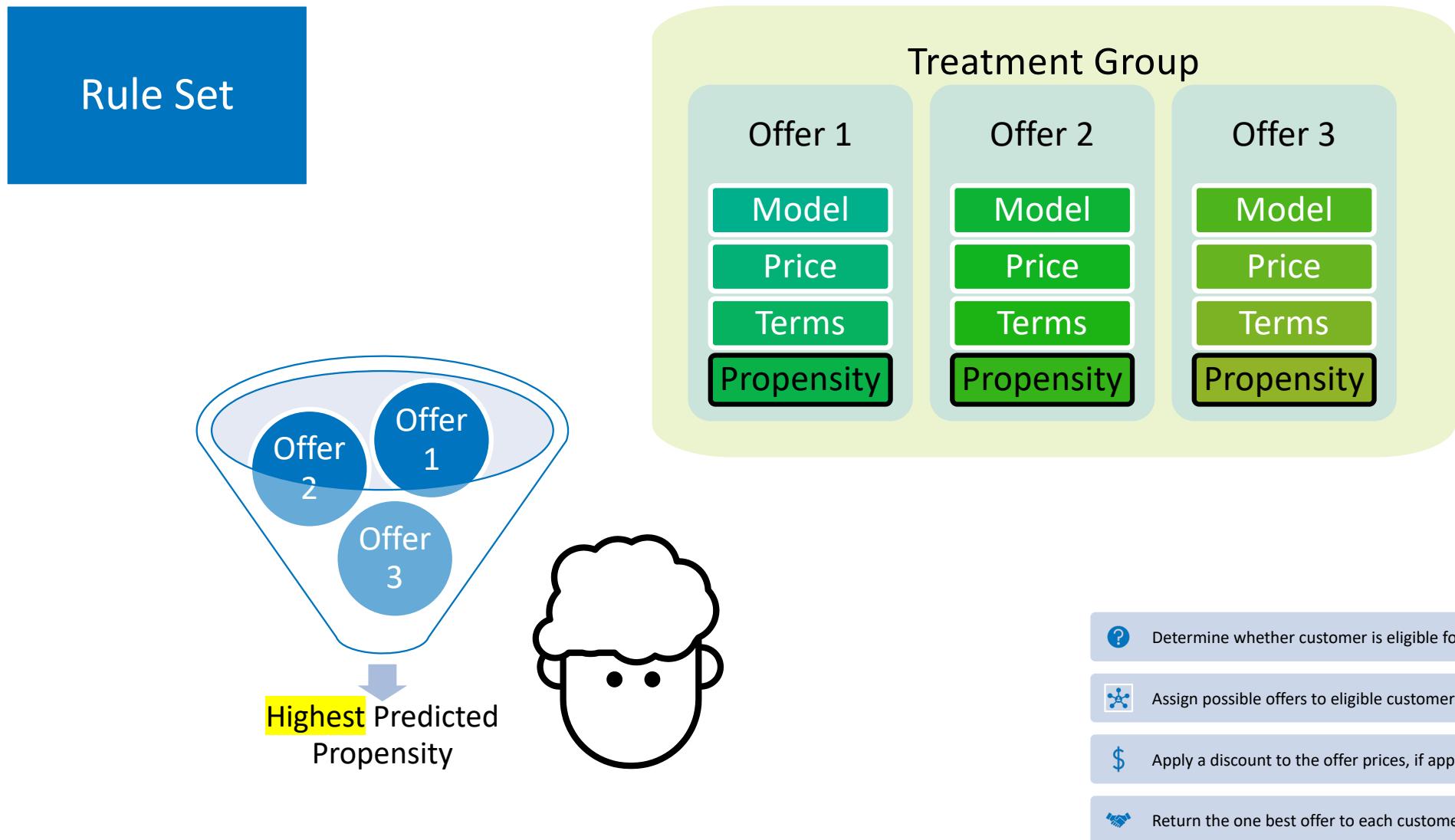
\$

Apply a discount to the offer prices, if appropriate.

?

Return the one best offer to each customer.

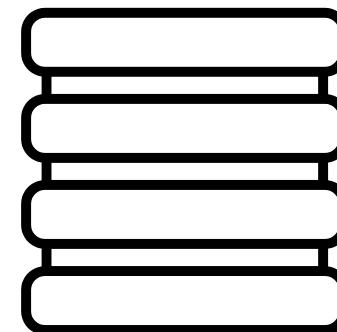
Upgrade Eligible=Yes



Upgrade Eligible=Yes



Decision variables



Database

?

Determine whether customer is eligible for upgrade.

?

Assign possible offers to eligible customers.

\$

Apply a discount to the offer prices, if appropriate.

?

Return the one best offer to each customer.



1) Working in SAS Intelligent Decisioning

This **demonstration** explores the SAS Intelligent Decisioning user interface.

Decisions

Search name

<input type="checkbox"/>	Name	Description	Location	Modified By	Date Modified
<input type="checkbox"/>	Demo_RecordContacts		/Users/lynn/My Folder/Decisioning/Demonstrations	lynn	Jun 9, 2021 11:42 AM
<input type="checkbox"/>	PhonePromoSolution		/Users/lynn/My Folder/Decisioning/Customer Intelligence/Solutions	lynn	Jun 9, 2021 11:37 AM
<input type="checkbox"/>	Practice_PublishValidate		/Users/lynn/My Folder/Decisioning/Practices	lynn	Jun 9, 2021 11:37 AM
<input type="checkbox"/>	Demo_TreatmentArbit...		/Users/lynn/My Folder/Decisioning/Demonstrations	lynn	Jun 9, 2021 11:32 AM
<input type="checkbox"/>	Practice_TreatmentArb...		/Users/lynn/My Folder/Decisioning/Practices	lynn	Jun 9, 2021 11:31 AM

New Decision

Properties

Select a decision to view its properties.

SAS® Intelligent Decisioning - Build Decisions

My First Decision (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Search name All types

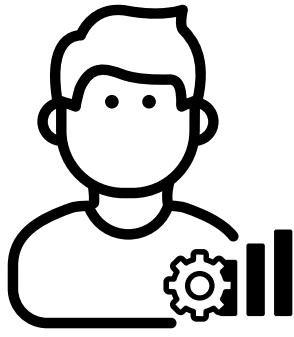
Start

End

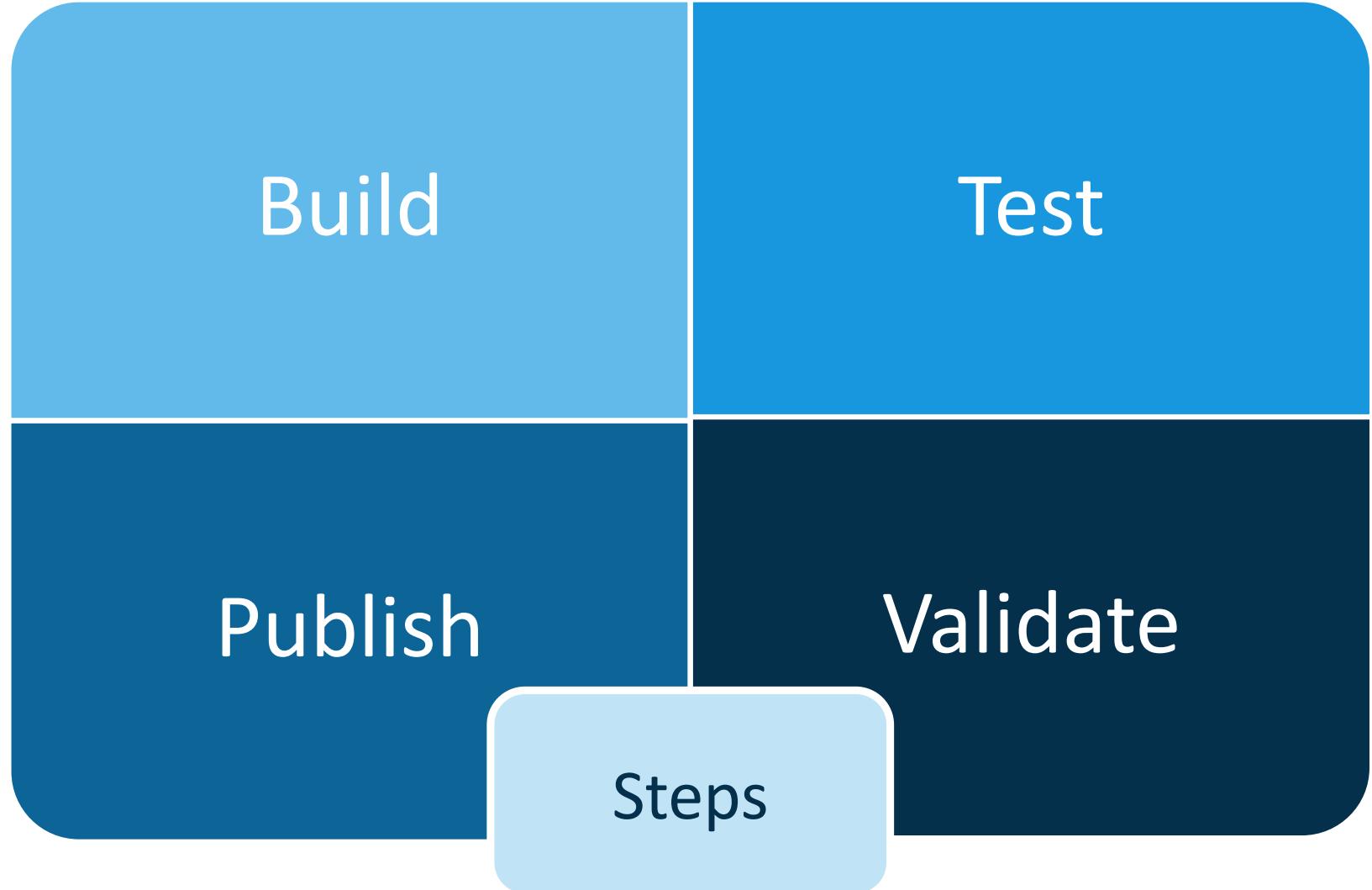
Select a node on the diagram to view its properties.

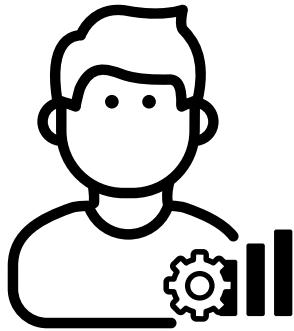
Properties

```
graph TD; Start[Start] --> End[End]
```

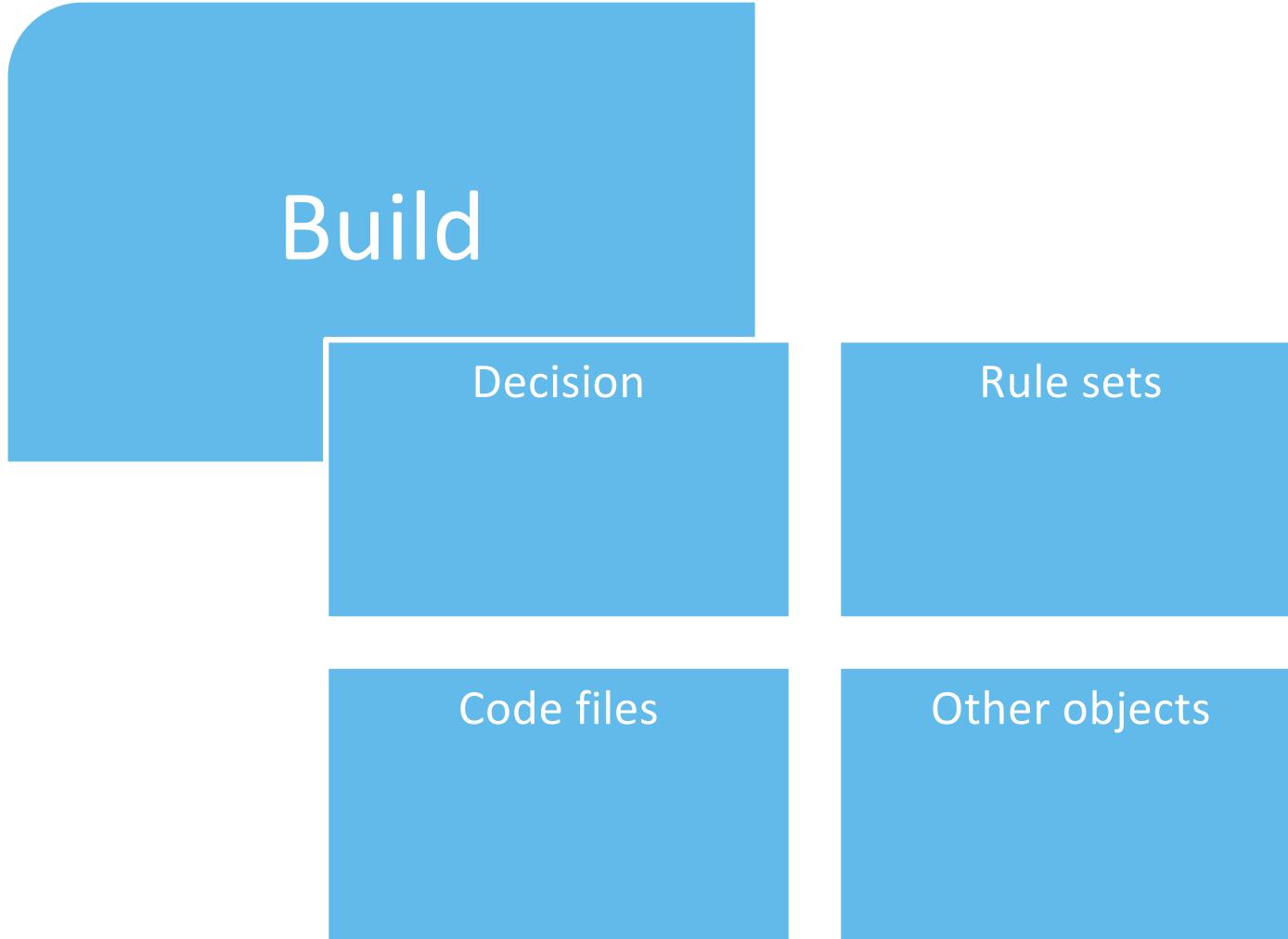


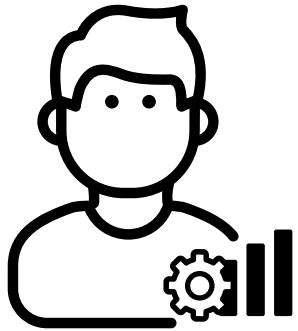
Decision
Builder





Decision
Builder





Decision
Builder

Recommended

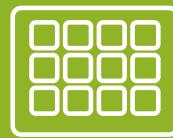
Decisions,
rule sets,
and code files

Design Environment



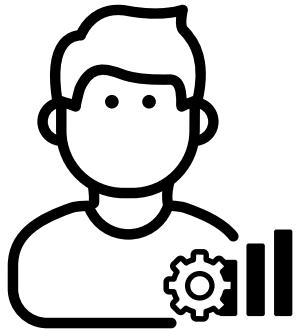
CAS

Preparation required



Data
Grid
Input

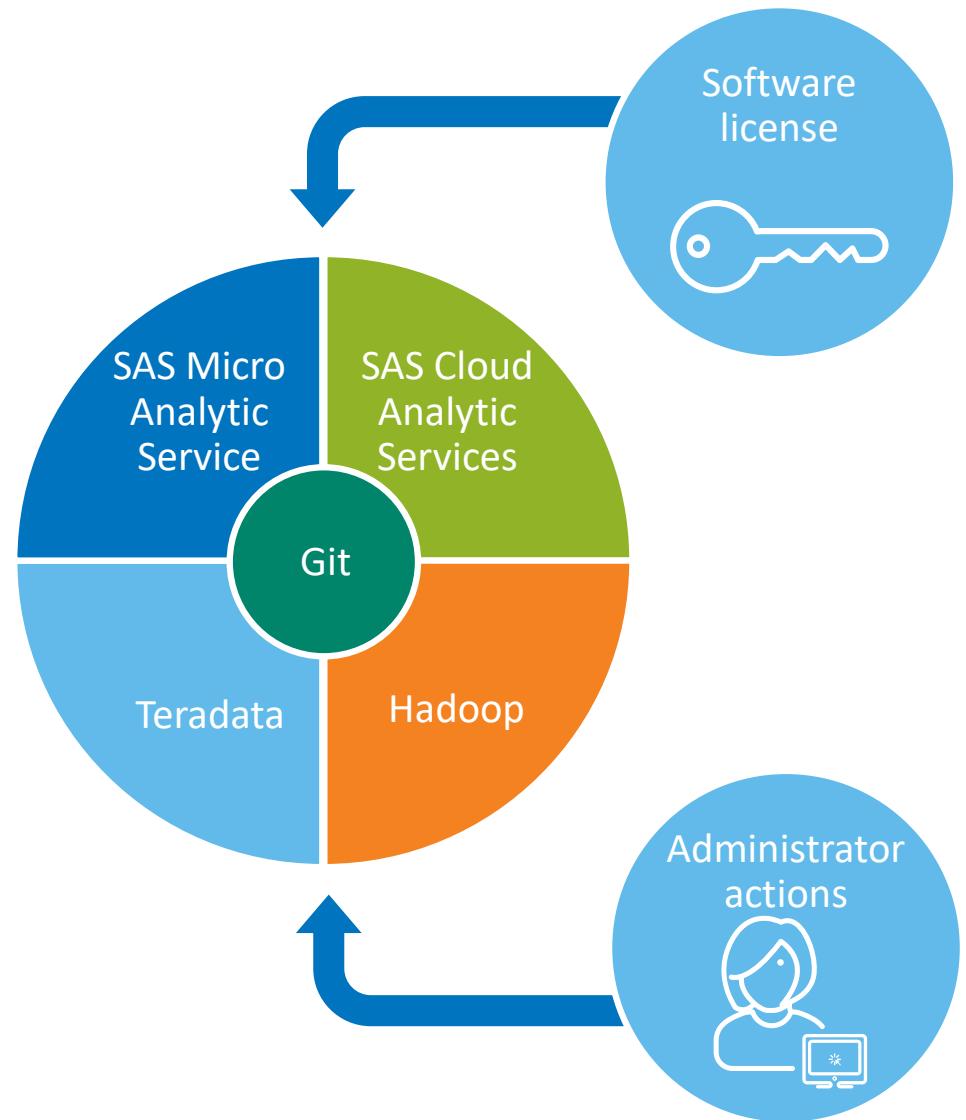
Test

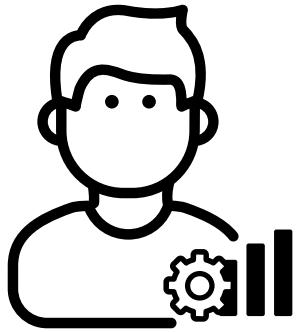


Decision
Builder

Decisions and
rule sets

Publish





Decision
Builder

Recommended

Target Publishing
Destination

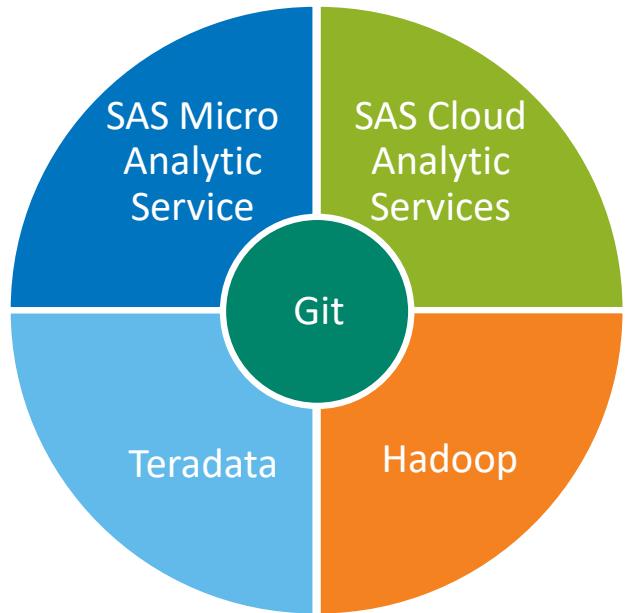
Decisions and
rule sets

Preparation required

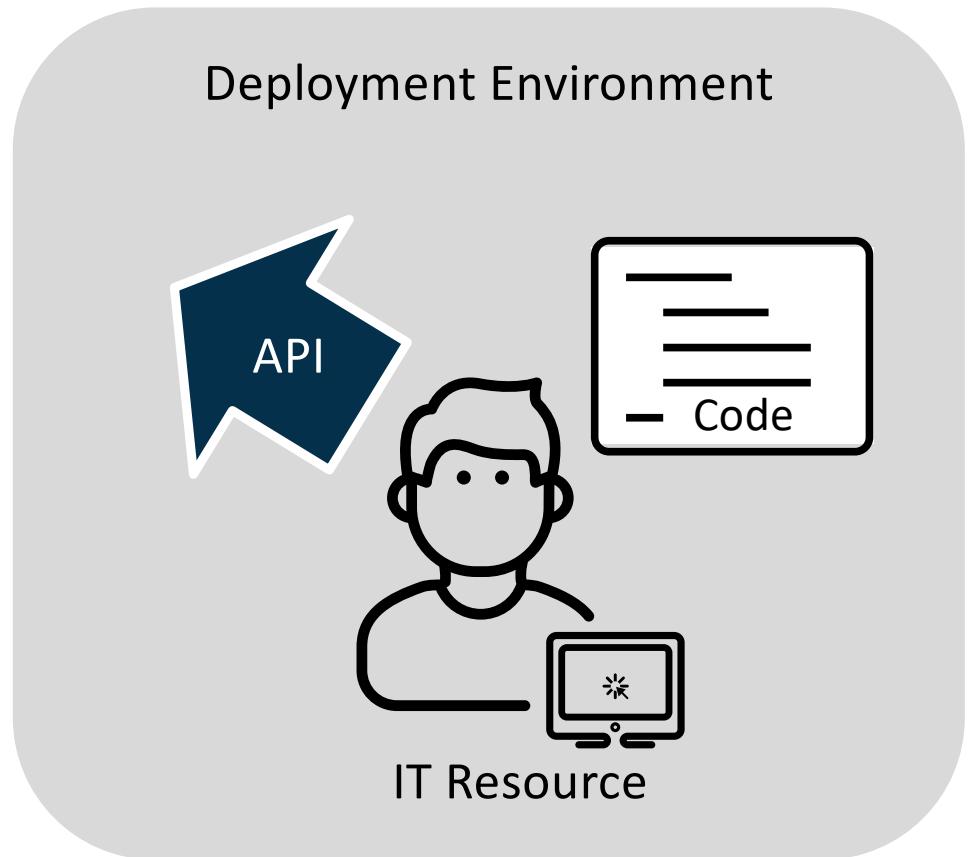


Data
Grid
Input

Validate



Next steps to process the decision are performed by an IT resource in the deployment environment.



SAS
Micro
Analytic
Service



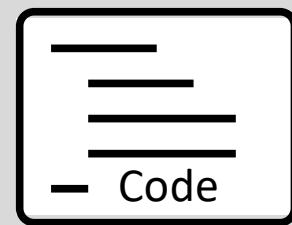
*SAS Micro Analytic Service:
Programming and
Administration Guide*

CAS
Hadoop
Git
Teradata



*SAS Visual Analytics:
Programming Guide*

Deployment Environment



IT Resource



2) Loading Tables into CAS

This **demonstration** shows how to load tables into CAS that will be used for testing.

SAS® Data Explorer - Manage Data

Available Data Sources Import

Filter

Public

- HMEQAPPS 06/10/23 09:17 AM • lynn
- hmeqapps.sashdat 05/26/21 02:35 PM
- INVESTMENTS.sashdat 05/26/21 02:51 PM
- predef_svrtdist.sashdat 06/09/23 05:26 PM
- SALESREPS.sashdat 05/27/21 01:57 PM
- tsaclaims.sashdat 05/26/21 02:35 PM
- VENDORLIST.sashdat 05/27/21 02:21 PM

HMEQAPPS

Details Sample Data Profile

Filter

#	Name	Label	Type	Raw Length	Formatted Length	Format	Tags
1	LOAN		double	8	12		
2	ACCOUNT		char	13	13		
3	PURPOSE		char	2	2		

Available Data Sources Import

Filter

- HMEQAPPS 06/10/23 09:17 AM • lynn
- INVESTMENTS 06/10/23 09:17 AM • lynn
- SALESREPS 06/10/23 09:18 AM • lynn
- TSACLAIMS 06/10/23 09:18 AM • lynn
- VENDORLIST 06/10/23 09:19 AM • lynn

- hmeqapps.sashdat
- investments.sashdat
- TSAClaims.sashdat
- SalesReps.sashdat
- VendorList.sashdat

Lesson 2: Decision Basics

2.1 Variables

2.2 Rule Sets and Rules

2.3 Global Variables

2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning

Lesson 2: Decision Basics

2.1 Variables

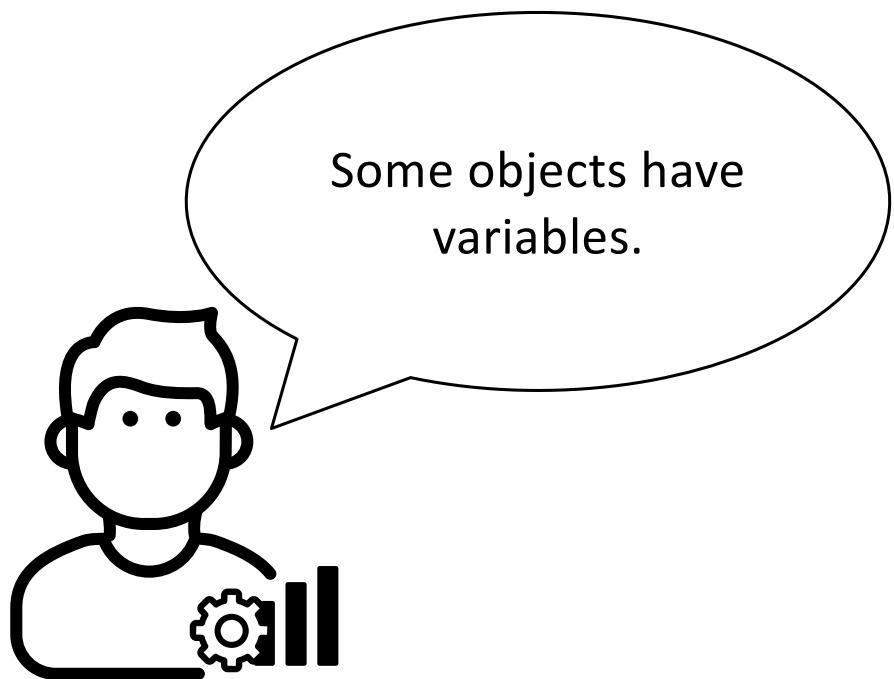
2.2 Rule Sets and Rules

2.3 Global Variables

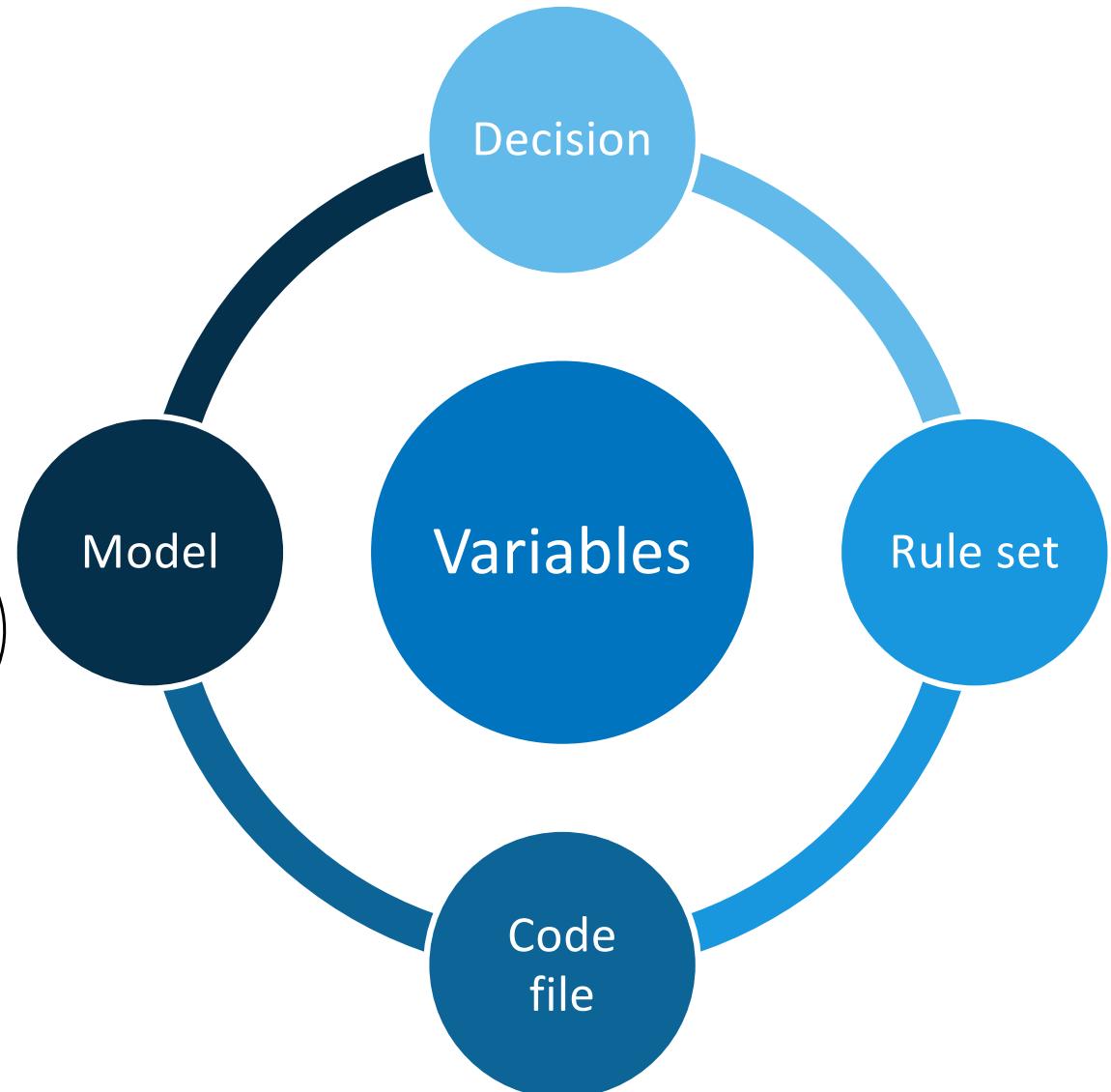
2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning



Some objects have variables.



Variable Properties

Name

Data Type

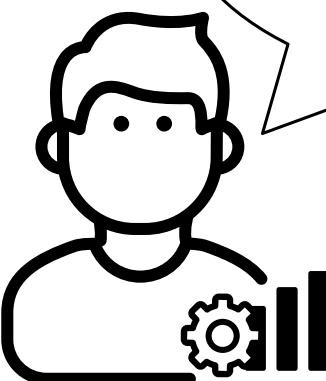
Input

Output

Length

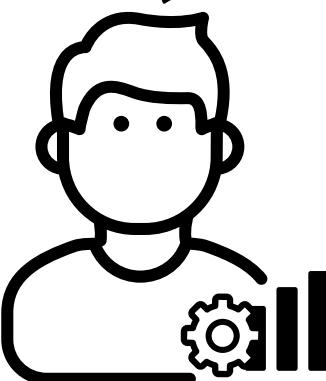
Initial Value

Description



Variables have
properties.

Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------



You must follow naming rules.

Begin

letter or underscore

Contain

alphanumeric and underscore

Length

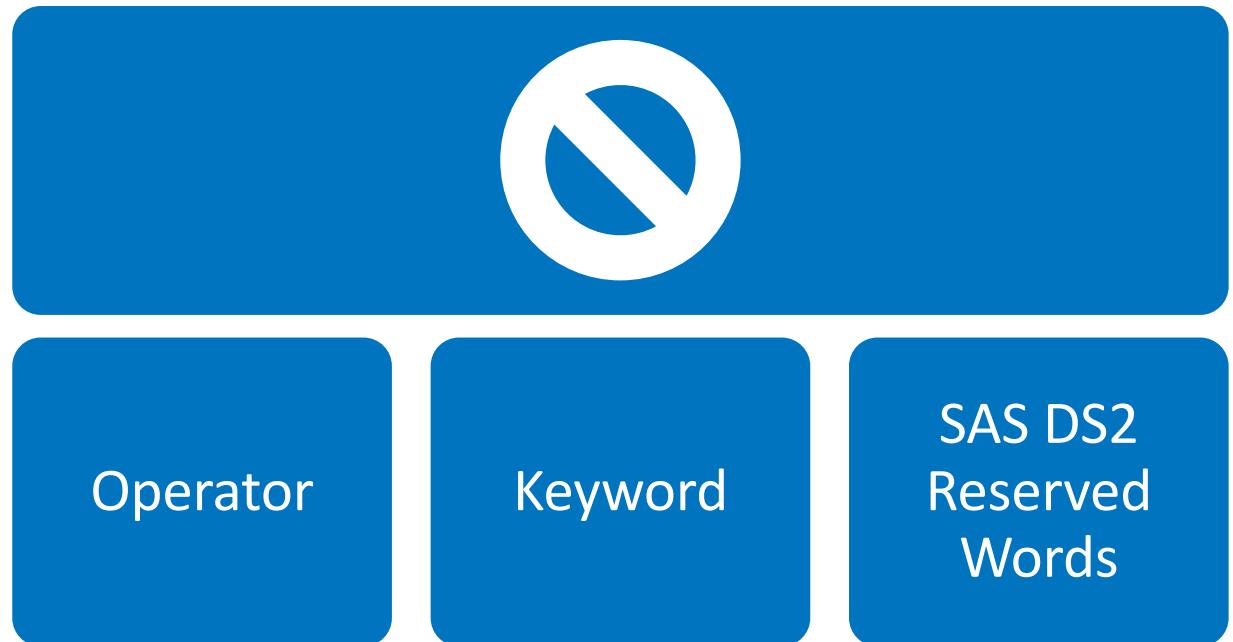
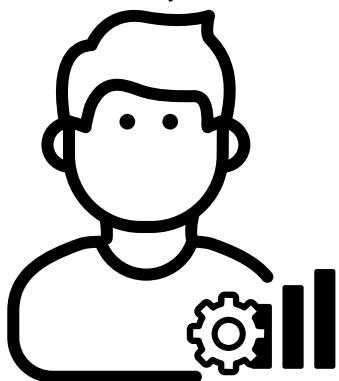
up to 32

Unique

within the object

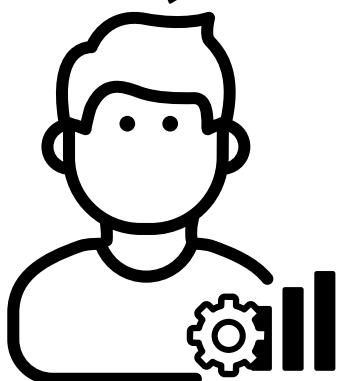
Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------

Some names are
not allowed.



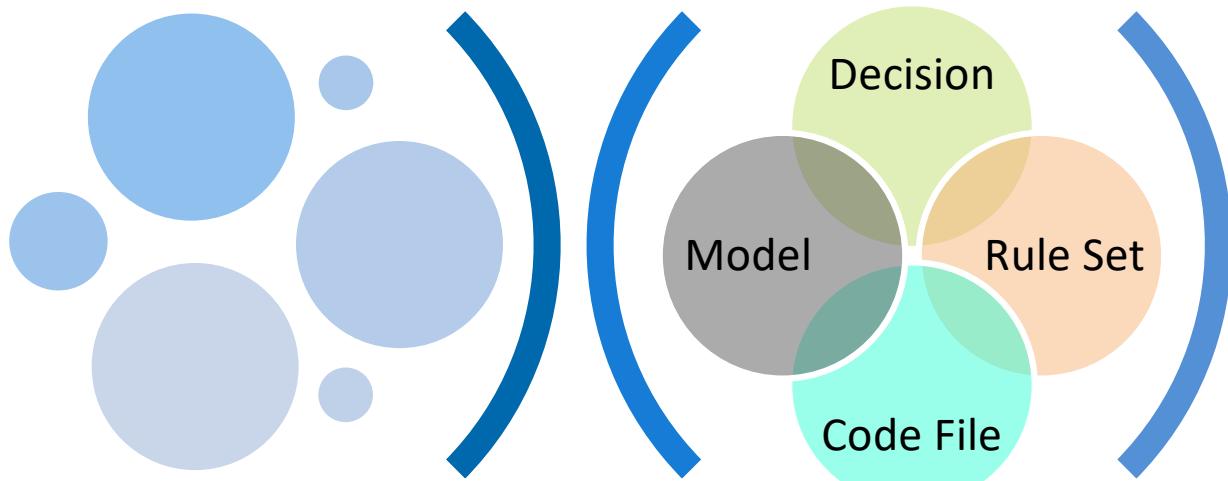
Name	Data Type	Input	Output	Length	Initial Value	Description
	Boolean					data grid
	date					decimal
	datetime					
	integer					varying-length binary
	binary					

Variables have a
data type.



9

Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------



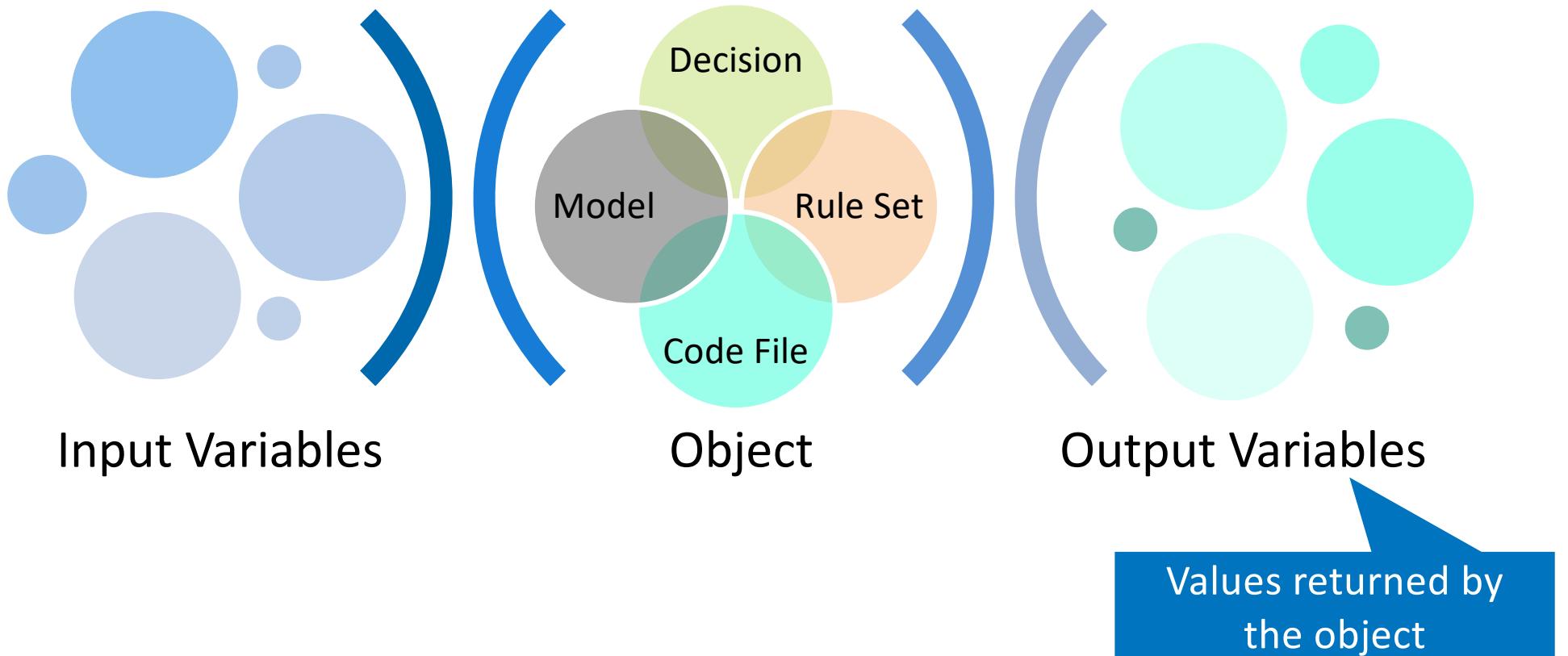
Input Variables

Object

Present in input to
object

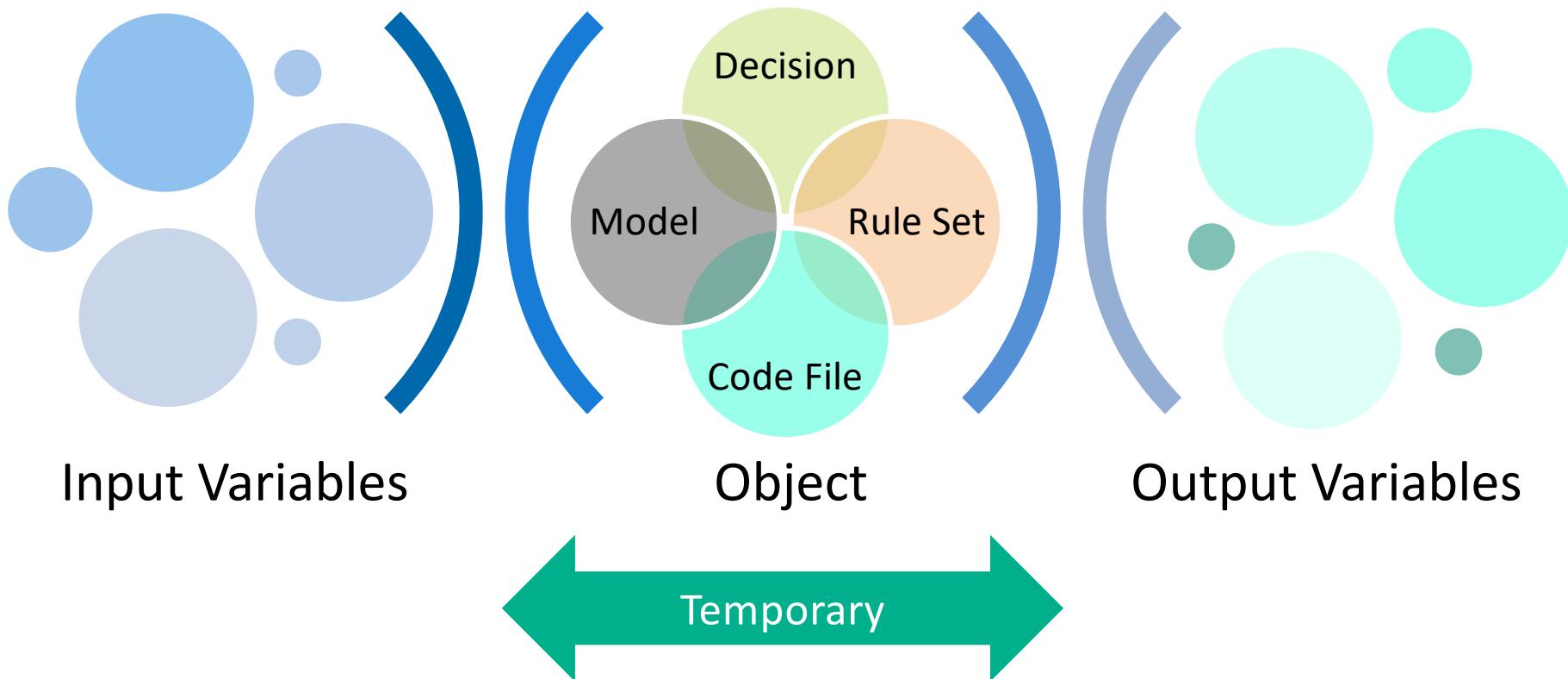
10

Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------

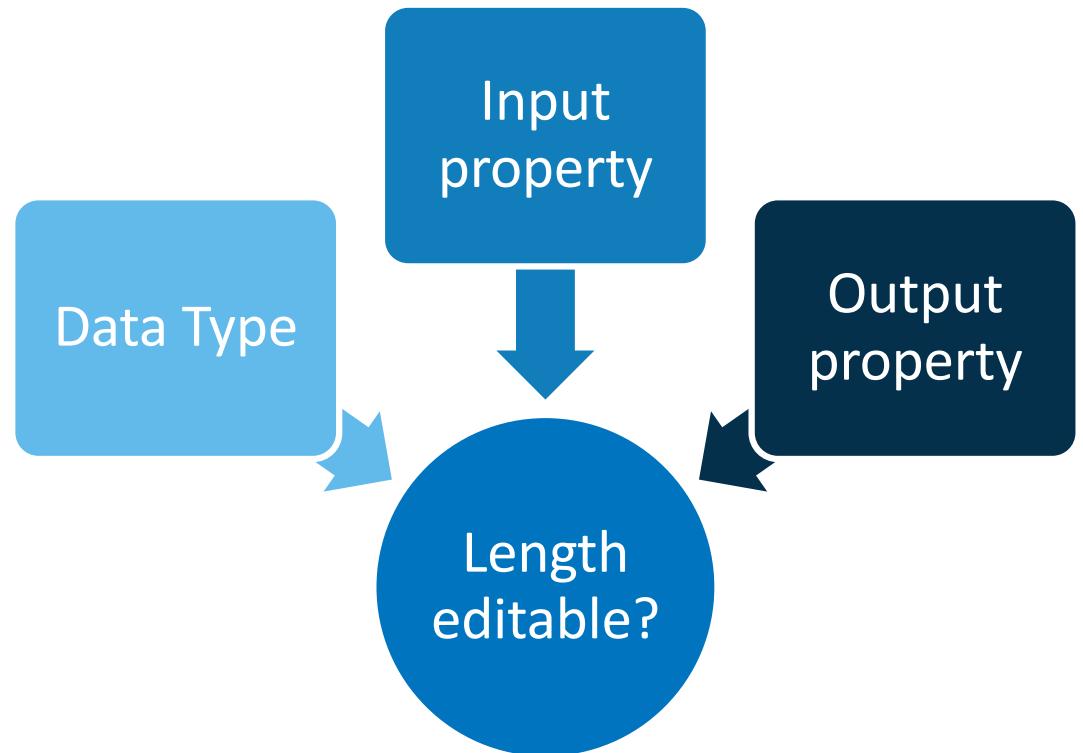
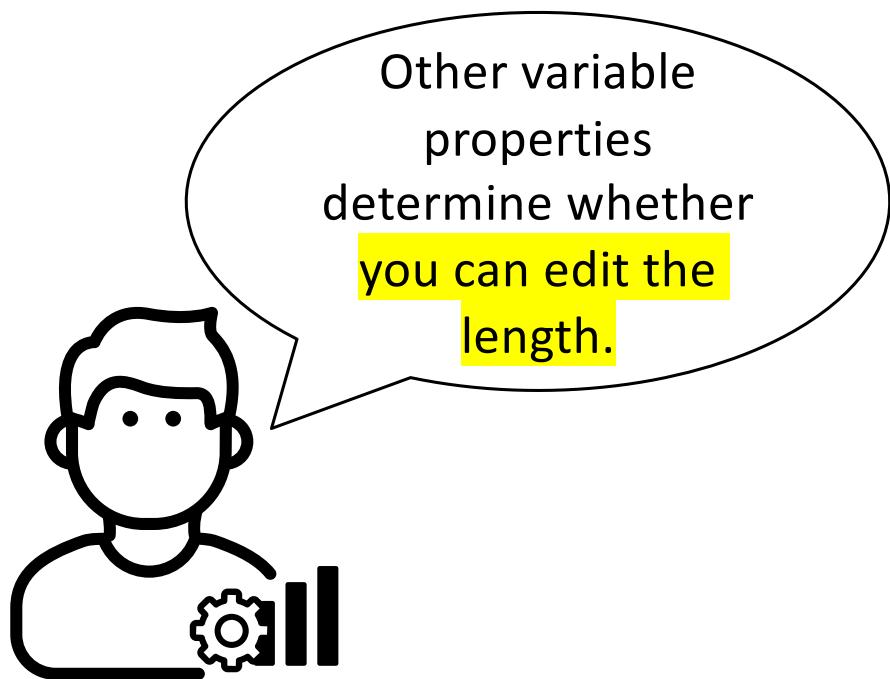


11

Name	Data Type	Input 	Output 	Length	Initial Value	Description
------	-----------	---	--	--------	---------------	-------------



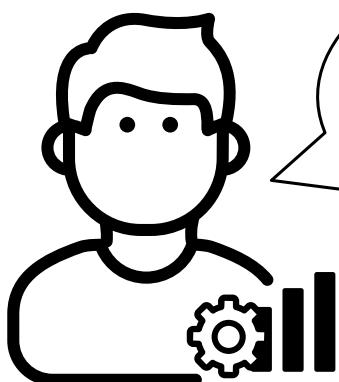
Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------



Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------

Length
potentially **editable**

Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------



You can specify initial values only for variables with certain properties.

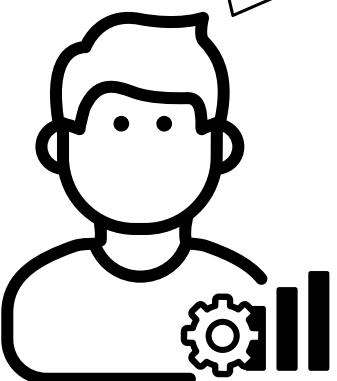
Not:

- data grid
- binary
- varying-length binary

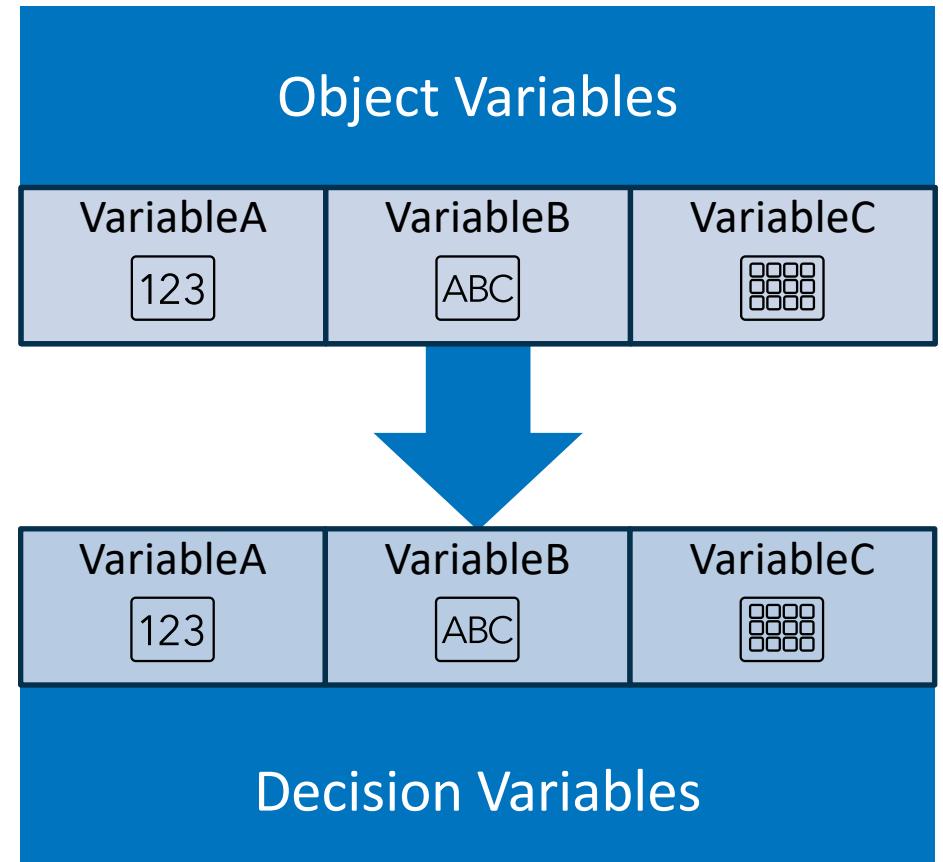
Name	Data Type	Input	Output	Length	Initial Value	Description
------	-----------	-------	--------	--------	---------------	-------------

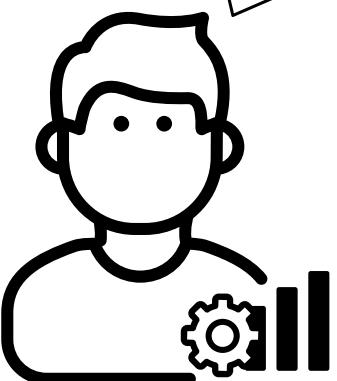
Optional

Up to 256
characters

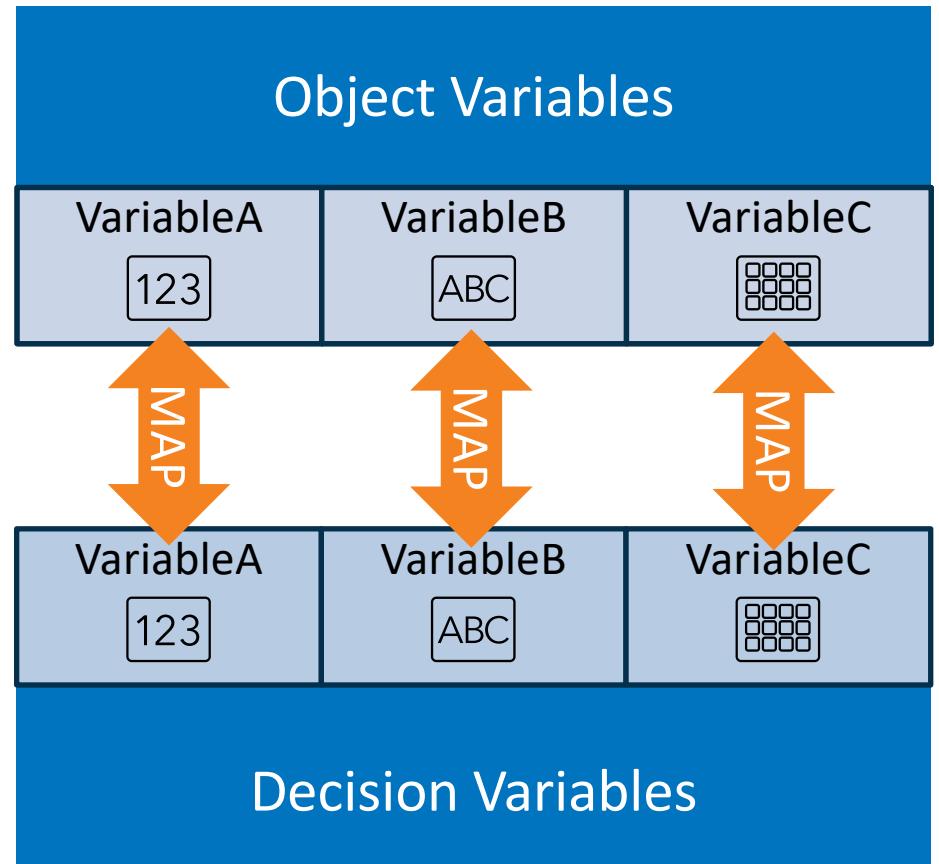


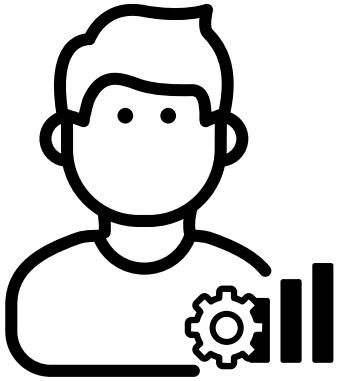
When you **add an object** that includes variables to a decision,
SAS Intelligent Decisioning
automatically creates variables for the decision.



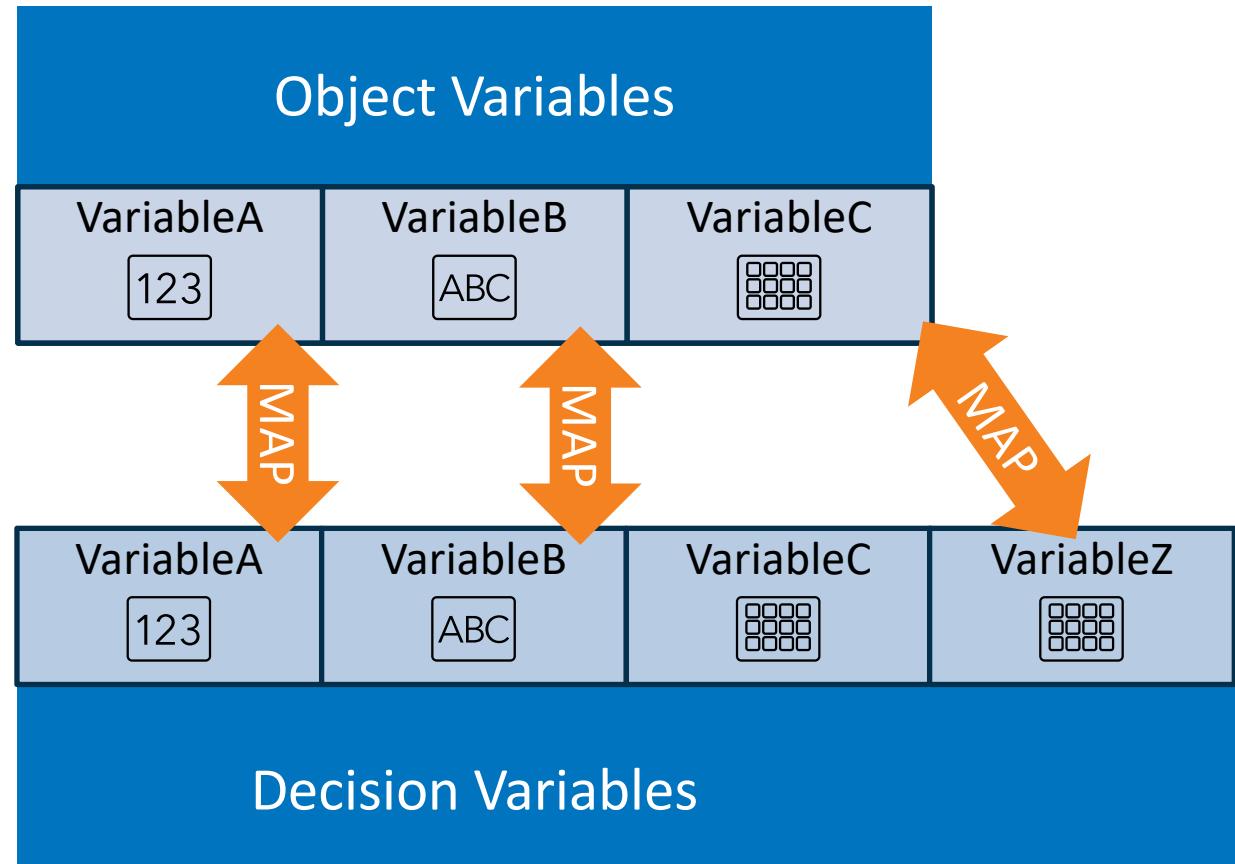


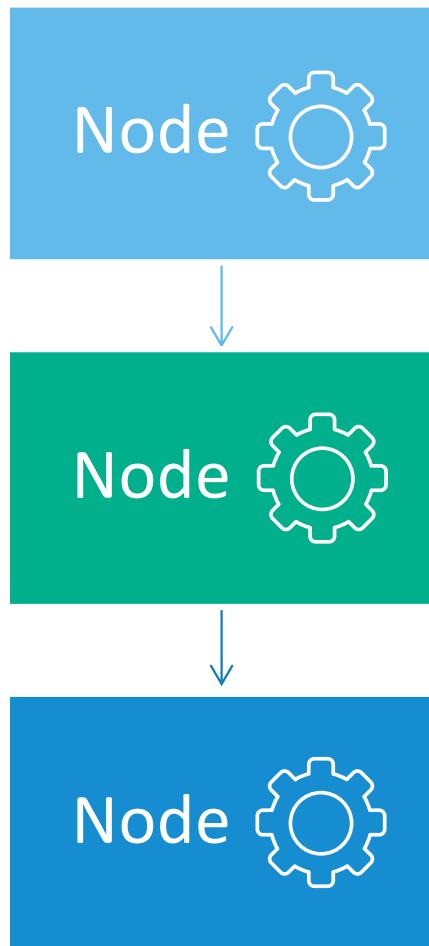
SAS Intelligent Decisioning also automatically maps the object variables to the corresponding variables for the decision.



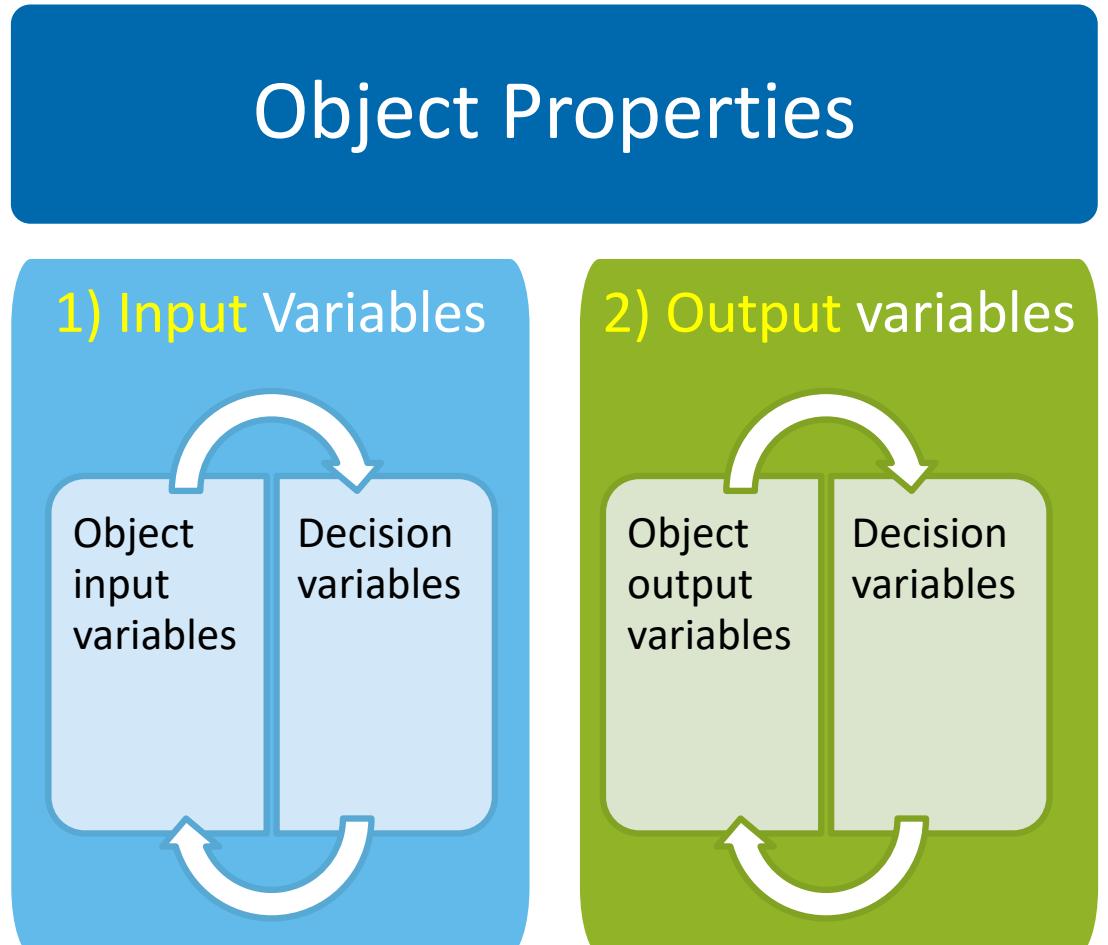


You can **change** the variable mappings.





Mapping 





1) Managing and Mapping **Variables** in a Decision

This demonstration illustrates importing variables from a data table and mapping decision variables.

1) Add variable in Decision

SAS® Intelligent Decisioning - Build Decisions

Decision Flow Decision Properties Variables Scoring Versions History

Decision Variables Global Variables

Object type: All objects Object: Select an item

Variables Data Type Input Output Initial Value

LOAN PURPOSE

Import Export Add variable Comma-delimited (*.csv) JSON (*.json)

Add value Code file Data table Decision Rule set Custom variable

Import Export Add variable Comma-delimited (*.csv) JSON (*.json)

Add value Code file Data table Decision Rule set Custom variable

Import Export Add variable Comma-delimited (*.csv) JSON (*.json)

Add value Code file Data table Decision Rule set Custom variable

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

* Decision_1 (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Decision Variables Global Variables

Object type: All objects Object: Select an item

Variables Data Type Input Output Initial Value

LOAN PURPOSE

Import Export Add variable

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

2) View Rule Sets

SAS® Intelligent Decisioning Not secure | server.demo.sas.com/SASDecisionManager/ SAS Drive SAS Studio JupyterHub Jupyter Notebook pgAdmin 4

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule Sets Rule sets Lookup tables Treatments Treatment groups Code files

Demo_variabl Description Location Type Modified By Date Modified Properties

Demo_VariableMa... /Users/lynn/My Folder/Decisioning/ Demonstrations Assignment lynn Apr 28, 2021 02:51 PM Select a rule set to view its properties.

New Rule Set

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Rule Set Properties Variables Scoring Versions History

Default_rule_1 Record rule-fired data

IF PURPOSE = 'DC'

AND LOANAMOUNT > 10000

THEN ASSIGN Queue 1

ELSE PURPOSE = 'HI'

AND LOANAMOUNT > 20000

THEN ASSIGN Queue 2

+ Add Rule

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Rule Set Properties Variables Scoring Versions History

Rule Set Variables Global Variables

Search variable

Variables	Data Type	Input
LOANAMOUNT	Decimal	<input checked="" type="checkbox"/>
PURPOSE	Character	<input checked="" type="checkbox"/>
Queue	Integer	<input type="checkbox"/>

3) Add Rule Sets to Decision Flow

SAS® Intelligent Decisioning - Build Decisions

Decisions * Decision_1 (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Search name All types

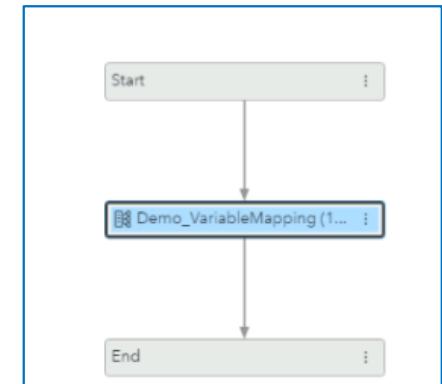
Start

Add below >

Properties

End

Branch
Data query
Decision
DS2 code file
Model
Python code file
Record contacts
Rule set
Treatment group



SAS® Intelligent Decisioning - Build Decisions

Decisions * Decision_1 (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Search name All types

Start

Demo_VariableMapping (1...)

End

Branch
Data query
Decision
DS2 code file
Model
Python code file
Record contacts
Rule set
Treatment group

Input Variables

Score rows in this data grid

Select an item

Input Variable Maps To

LOANAMOUNT	LOAN
PURPOSE	PURPOSE

4) Map input & output variables

Lesson 2: Decision Basics

2.1 Variables

2.2 Rule Sets and Rules

2.3 Global Variables

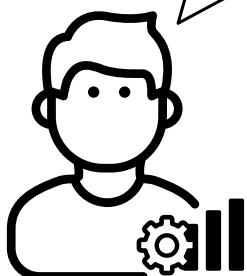
2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning

Rule set

You can define rules to specify conditions to be evaluated and actions to be taken.

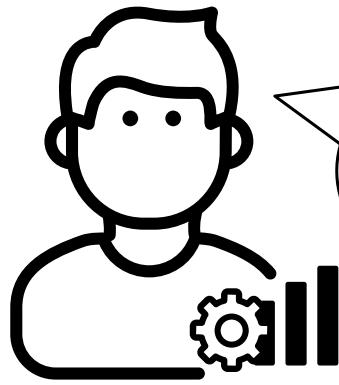


Rule

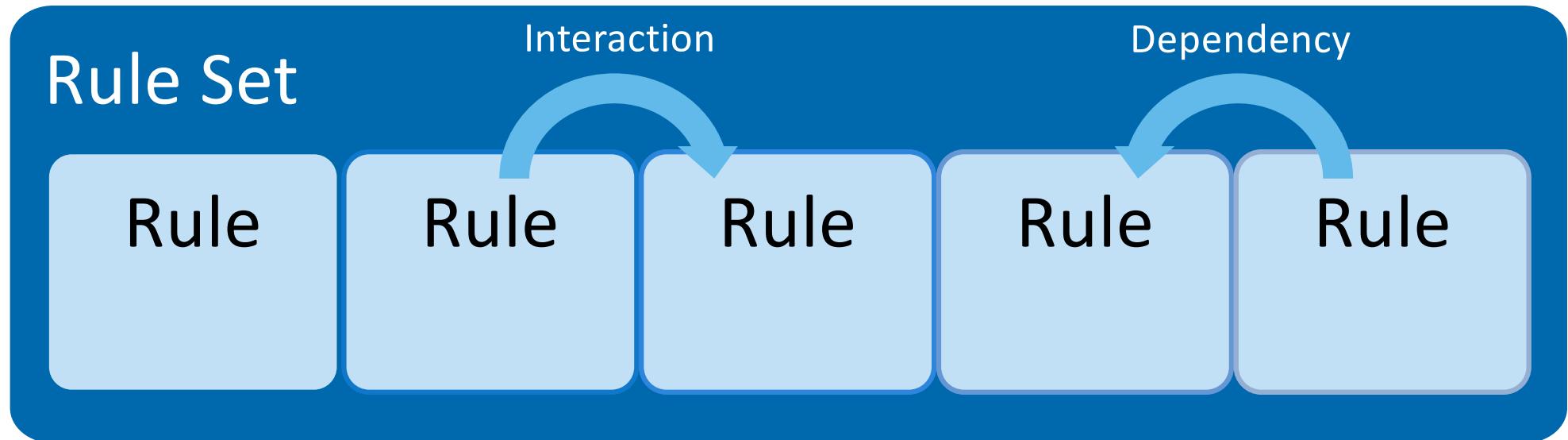
Condition

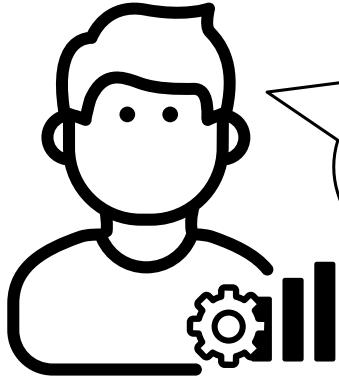
Action

Condition + Action



Rule sets are logical collections of rules.



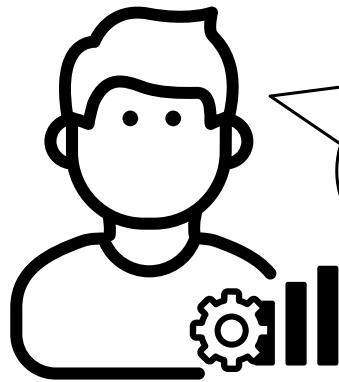


There are two types
of statements that
you can use in rules.

Rule Statement Types

Condition
(If-Else)

Action
(Assignment)



A given rule can potentially include either or both types of statements.

Rule 1

Action

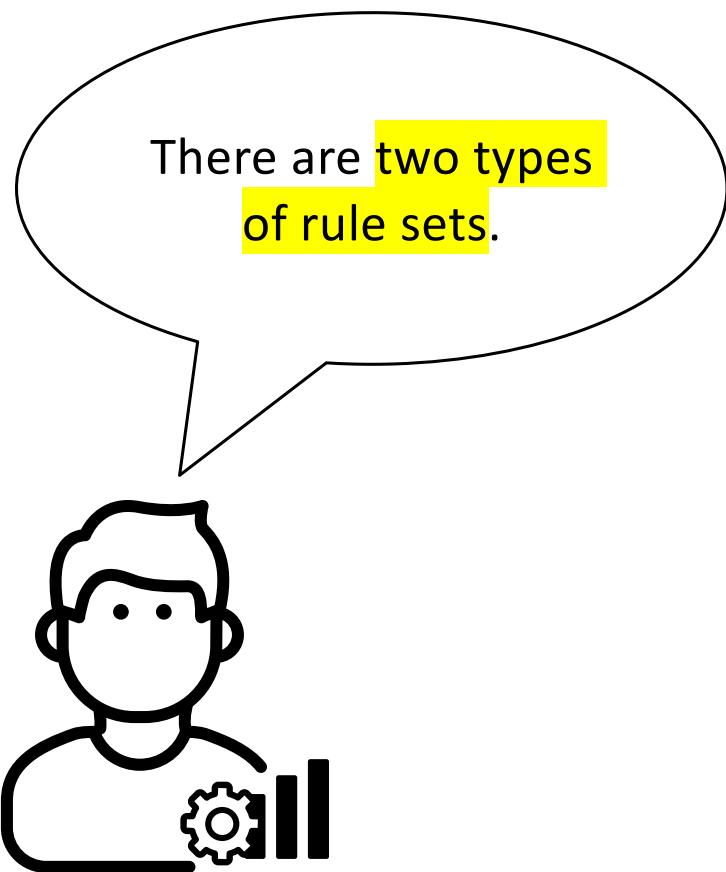
Rule 2

Condition

Rule 3

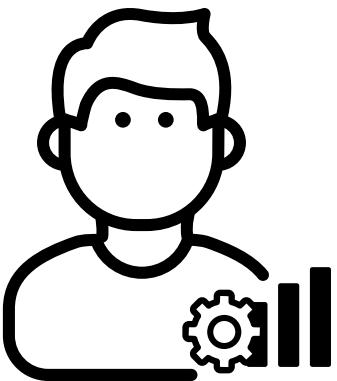
Condition

Action



There are two types
of rule sets.





Assignment rule sets can include rules that contain both statement types.

1) Assignment Rule Set

Rule

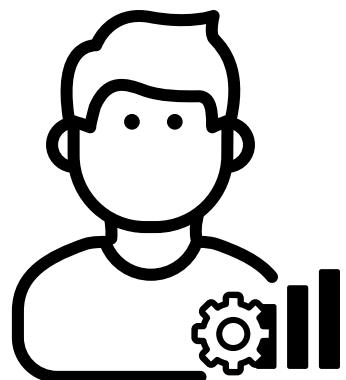
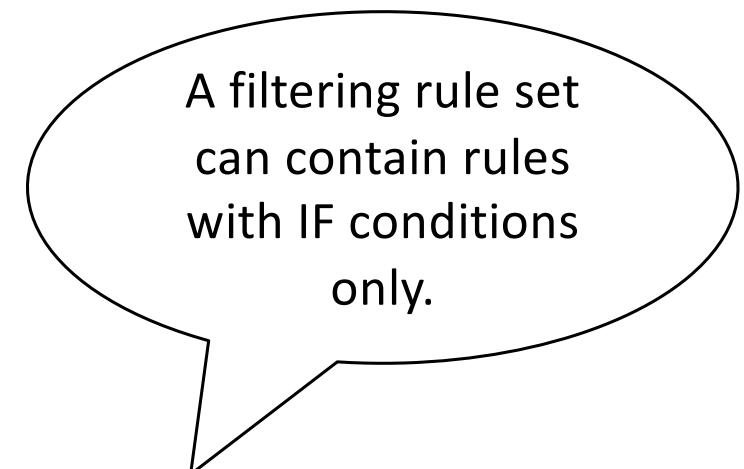
IF Condition

Rule

Action

Rule

Condition
+ Action



2) Filtering Rule Set

Rule

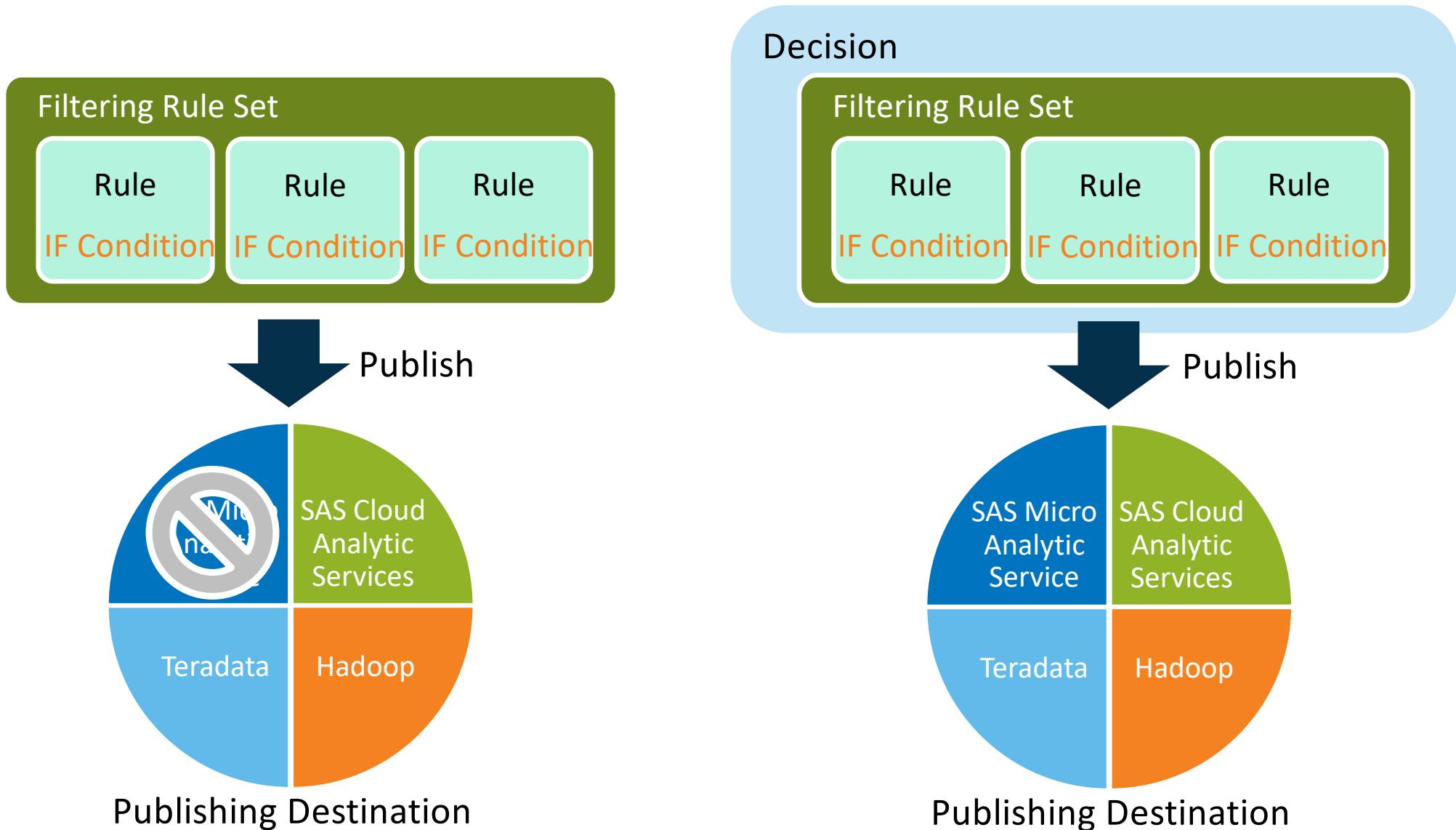
IF Condition

Rule

IF Condition

Rule

IF Condition

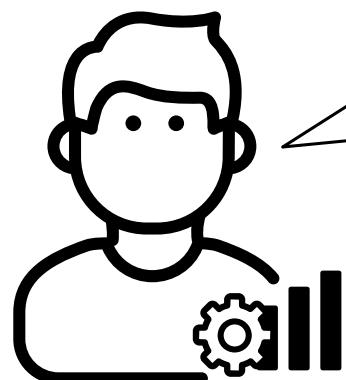




Construct Rules

1) Rule set editor

2) Expression Editor



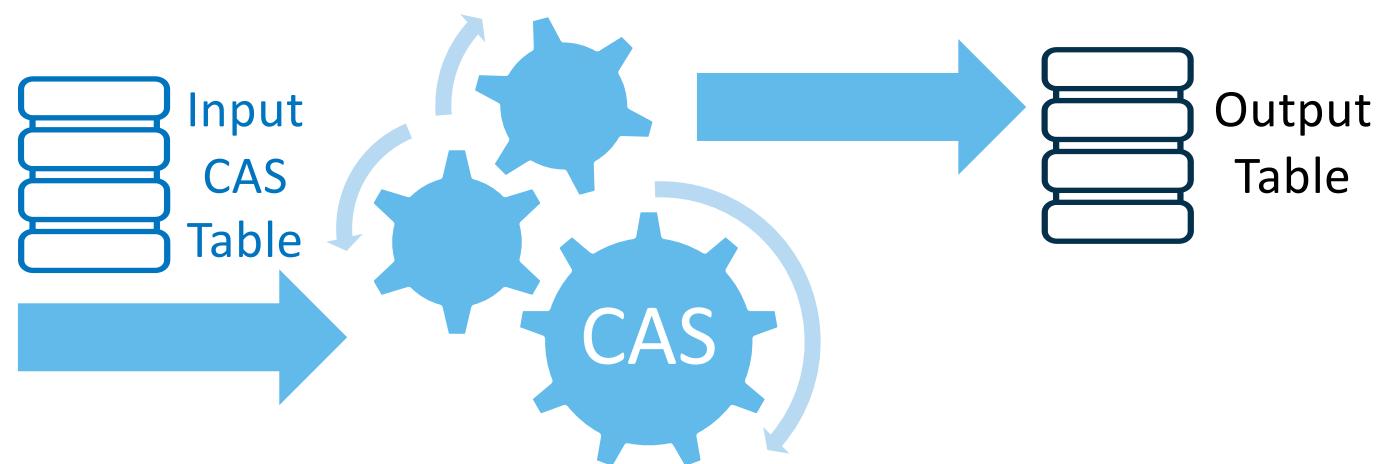
You use the rule set editor to construct simpler rules and the Expression Editor to construct more complicated rules.

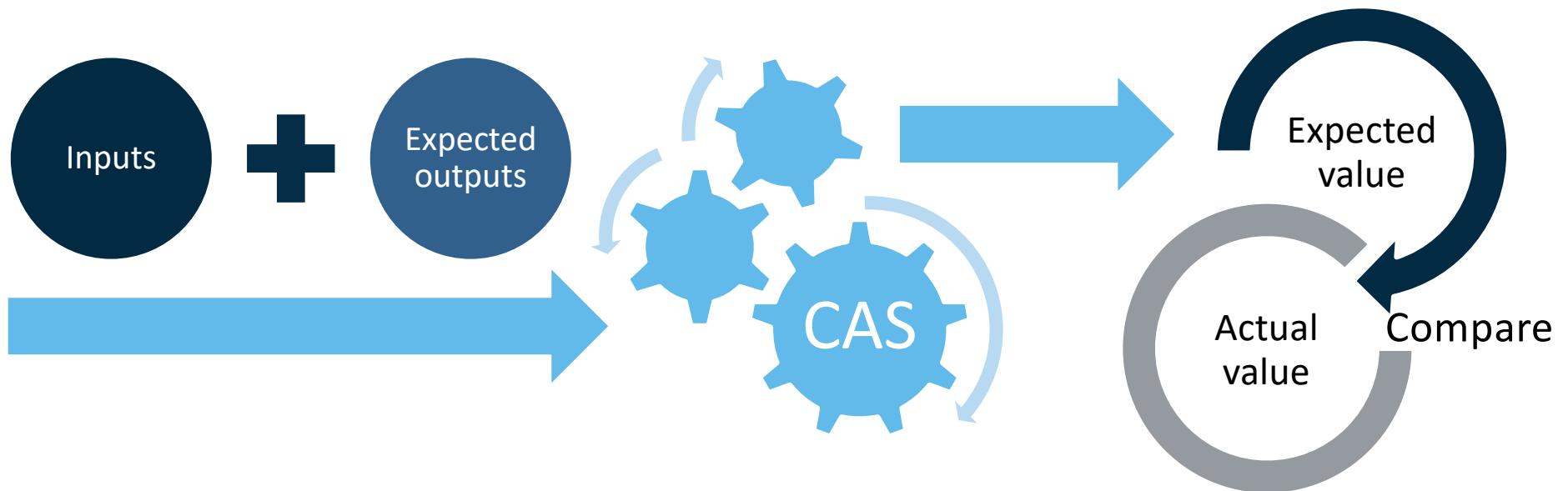
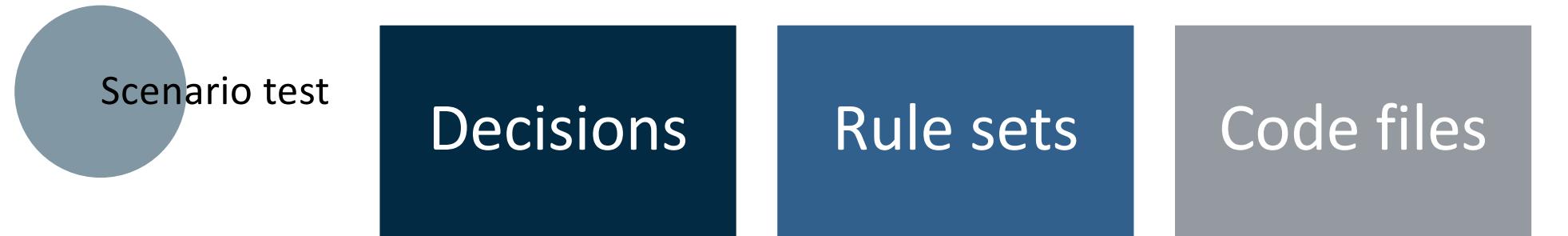
Basic test

Decisions

Rule sets

Code files







2) Creating a Rule Set

This demonstration illustrates how to create an assignment rule set.

1) New Rule Sets

This screenshot shows the 'Rule Sets' section of the SAS Intelligent Deciding interface. A new rule set named 'Demo_variabl' has been created. The properties for this rule set are displayed: Name is 'Demo_variabl', Description is empty, Location is '/Users/lynn/My Folder/Decisioning/Demonstrations', Type is 'Assignment', Modified By is 'lynn', and Date Modified is 'Apr 28, 2021 02:51 PM'. A message at the bottom right says 'Select a rule set to view its properties.'

2) Variable to Rule Sets

This screenshot shows the 'Variables' tab of a rule set named 'Demo_LoanApplication (1.0)'. A modal dialog titled 'Add Variables' is open, prompting for a variable name and data type. The 'Data type' dropdown is set to 'Character'. The 'Optional' section is expanded, showing checkboxes for 'Input' and 'Output'. The 'Output' checkbox is checked. Other columns include 'Length', 'Initial Value', and 'Description'. A sidebar on the right lists variable types: Code file, Data table, Decision, Rule set, and Custom variable.

2.1) Add variables from Decision

This screenshot shows the 'Variables' tab of a rule set. A modal dialog titled 'Add Variables' is open, showing variables selected from a decision. The 'Selected items (2):' list contains 'DELINQ' and 'DEROG'. The 'Variables' table shows these variables with their data types: 'DELINQ' is 'integer-output-Demo_AddRuleSet' and 'DEROG' is 'integer-output-Demo_AddRuleSet'. The 'Data Type' column shows 'Character' for both. The 'Input' and 'Output' checkboxes are checked for both variables. The 'Length' and 'Initial Value' columns are empty. The 'Description' column is also empty.

2.2) Add custom variables

This screenshot shows the 'Variables' tab of a rule set. A modal dialog titled 'Add Variables' is open, showing a single variable 'AppStatus' with a data type of 'Character'. The 'Optional' section is expanded, showing checkboxes for 'Input' and 'Output'. The 'Output' checkbox is checked. Other columns include 'Length', 'Initial Value', and 'Description'. A sidebar on the right lists variable types: Code file, Data table, Decision, Rule set, and Custom variable.

2.3) Adjust input & output

This screenshot shows the 'Variables' tab of a rule set. The variables 'AppStatus', 'DELINQ', and 'DEROG' are listed. The 'Data Type' column shows 'Character' for 'AppStatus' and 'Integer' for 'DELINQ' and 'DEROG'. The 'Input' and 'Output' checkboxes are checked for all three variables. The 'Length' and 'Initial Value' columns are empty. The 'Description' column is also empty.

3) Create rule in Rule Set

Demo_LoanApplication (1.0)

Rule Set Properties Variables Scoring Versions History

Reject high delinquencies or derogatory reports

IF THEN AppStatus

Record rule-fired data

+ Add Rule

Expression Editor: Default_rule_1

Variables Functions

Filter +

AppStatus
DELINQ
DEROG

DELINQ > 2 OR DEROG > 4

The expression is valid.

Description

Save Cancel

The screenshot shows the Expression Editor window for a rule named 'Default_rule_1'. On the left, there are tabs for 'Variables' and 'Functions', and a 'Filter' input field. Below these are three variables listed: 'AppStatus', 'DELINQ', and 'DEROG'. The main area contains a text input field with the expression 'DELINQ > 2 OR DEROG > 4'. Above the input field are various operators and functions: '+', '*', '/', '**', '|', '||', '()', '?', ',', 'LIKE', 'IN', 'NOTIN', 'AND', 'OR', 'NOT', '=', '<', '>'. Below the input field, a green message box states 'The expression is valid.' At the bottom of the editor are 'Validate' and 'Clear' buttons. At the very bottom of the entire interface are 'Save' and 'Cancel' buttons.

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

Demo_LoanApplication (1.0)

Rule Set Properties Variables Scoring Versions History

Tests Scenarios Publishing Validation

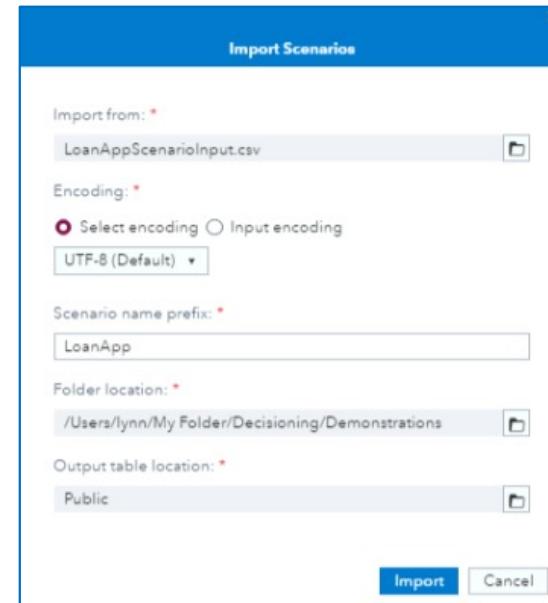
Filter

Name	Results	Status	Date Modified	Rule Set Version
------	---------	--------	---------------	------------------

New Test Import Scenarios Run Properties

Select a scenario to view its properties.

4) Scoring in Scenarios



4.1) Scoring in Scenarios

	A	B	C
1	DELINQ	DEROG	AppStatus_expected
2	0	0	
3	0	5	REJECT
4	5	0	REJECT
-			

5) Scoring results

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Demo_LoanApplication (1.0) Publish Import Export Close

Rule Set Properties Variables Scoring Versions History

Tests Scenarios Publishing Validation

Filter

Name	Results	Status	Date Modified	Rule Set Version
LoanApp_7	grid	✓	Jun 10, 2023 09:25 PM	1.0
LoanApp_8	grid	✓	Jun 10, 2023 09:25 PM	1.0
LoanApp_9	grid	✓	Jun 10, 2023 09:25 PM	1.0

Properties

	A	B	C
1	DELINQ	DEROG	AppStatus_expected
2	0	0	
3	0	5	REJECT
4	5	0	REJECT
5			

Name: LoanApp_9
Status: ✓ Completed successfully
Date last run: Jun 11, 2023 01:01

Edit Scenario Test

Search

Input Variable	Type	Value (Required)
DELINQ	Integer	5
DEROG	Integer	0

....

Search

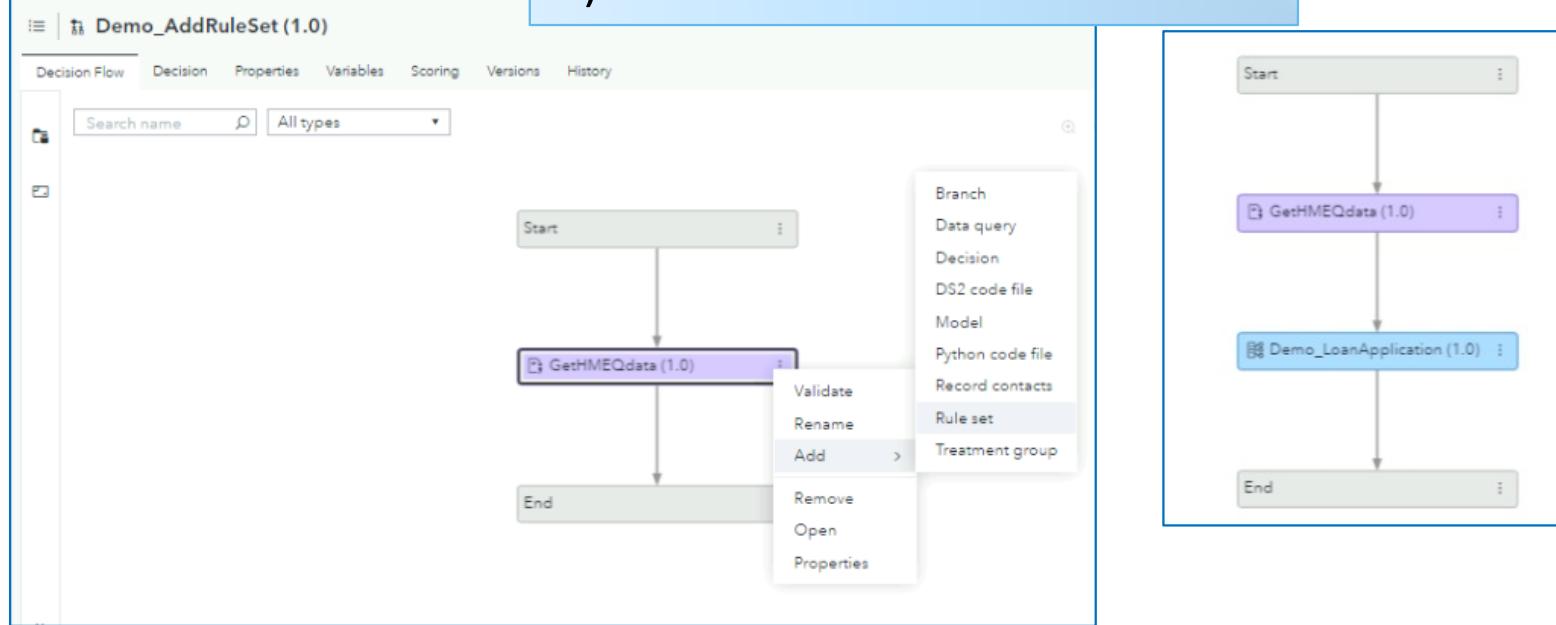
Output Variable	Type	Include	Expected Output (Optional)
AppStatus	Character	✓	REJECT
DELINQ	Integer	□	
DEROG	Integer	□	



3) Adding a Rule Set to a Decision

This demonstration illustrates how to add a rule set to a decision.

1) Add Rule Set to Decision Flow



The screenshot shows the 'Rule Set' tab of the 'Demo_LoanApplication (1.0)' application. It displays a single rule:

```
IF DELINQ > 2 OR DEROG > 4
THEN ASSIGN AppStatus = 'REJECT'
```

The 'Record rule-fired data' checkbox is checked. Other tabs visible include 'Properties', 'Variables', 'Scoring', 'Versions', and 'History'. There is a 'Publish', 'Import', 'Export', and 'Close' button bar at the top right.

2) Test in Scoring of Decisions

SAS® Intelligent Decisioning - Build Decisions

Decisions Demo_AddRuleSet (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Tests Scenarios Publishing Validation

New Test Run Properties

Name	Results	Status	Date Modified	Decision Version
Demo_AddRuleSet_Test_1		Green	Apr 28, 2021 02:51 PM	1.0

Name: Demo_AddRuleSet_Test_1
Status:

Edit Test

Name: *
Demo_AddRuleSet_Test_1

Description:

Location: /Users/lynn/My Folder/Decisioning/Demonstrations

Inputtable: *
HMEQAPPS

Advanced

Save Run Cancel

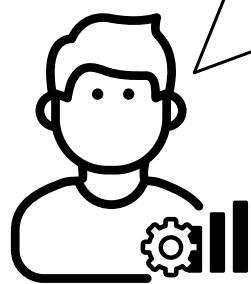
Demo_AddRuleSet (1.0) > Demo_AddRuleSet_Test_1

Test Results

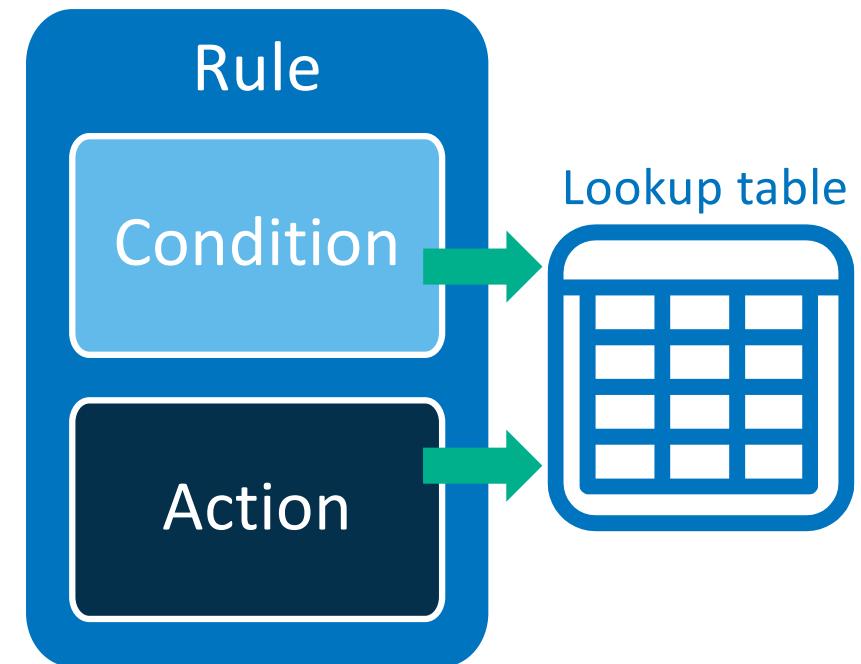
- Output
- Code
- Log
- Rule-Fired Analysis
- Decision Path Tracking

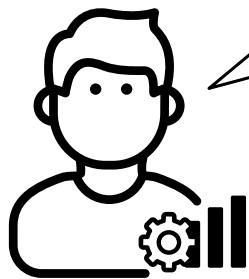
Output Table

Rules Fired Count	AppStatus	DELINQ	DEROG
1	REJECT	6	2
1	REJECT	6	0
1	REJECT	3	1
0		2	2
0		2	1
0		2	0

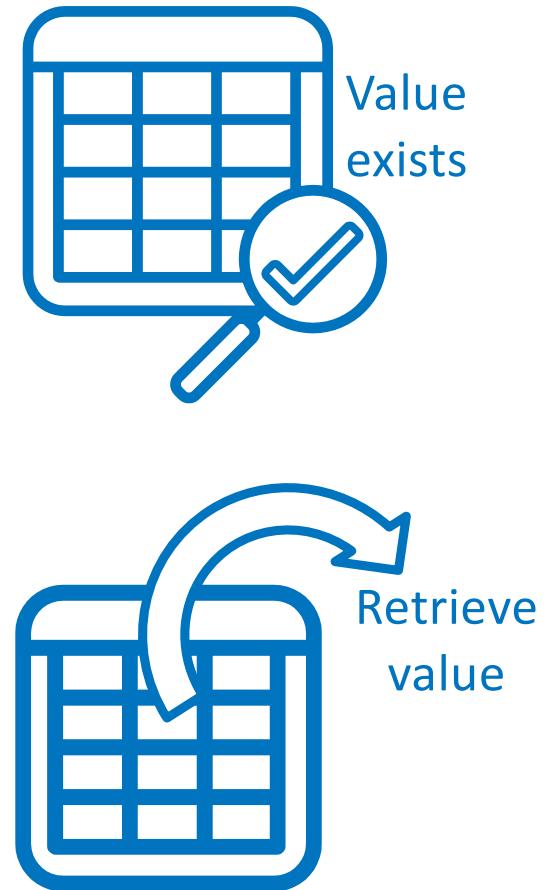


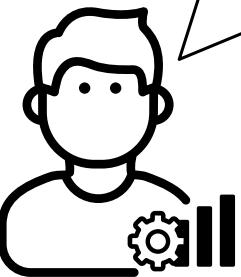
You can use **lookup tables** in rules.





You use lookup tables to determine whether a value exists in a table or to retrieve a lookup value.





A lookup table is a table of key-value pairs.

Must be unique

Lookup Table

Key

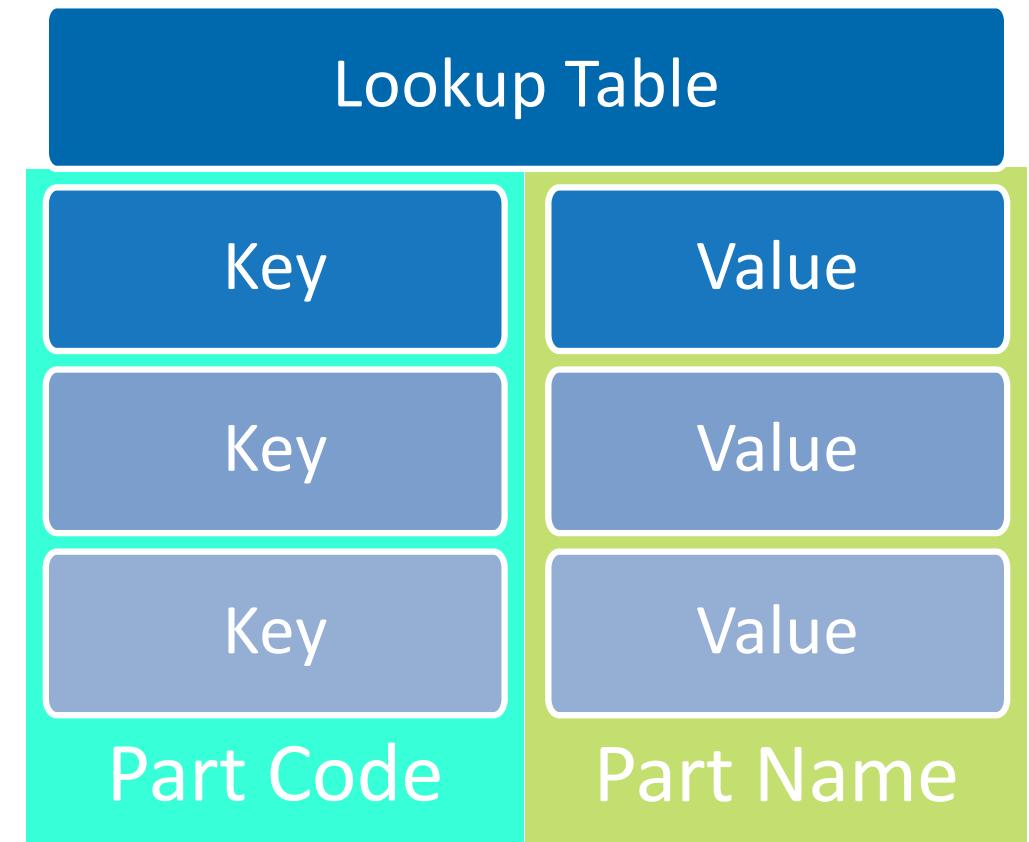
Value

Key

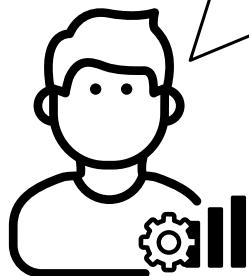
Value

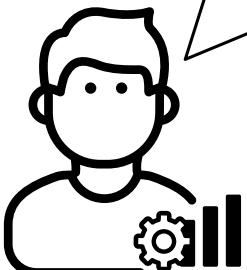
Key

Value



For example, you could retrieve a part name using the part code.



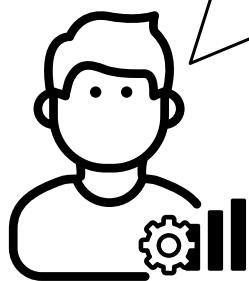


Lookup tables are intended to store reference data that is relatively small and changes infrequently.

Lookup table



Maximum of
20,000 rows



If you need to store larger or frequently changing data, use an external data source.

- Larger data
- Frequently changing data

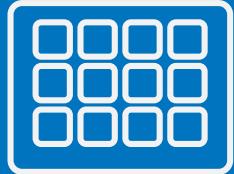
Lookup table



Data query

Database

Rule



Lookup
Table

Condition

LOOKUP



Value
exists

Action

LOOKUPVALUE



Assign
value



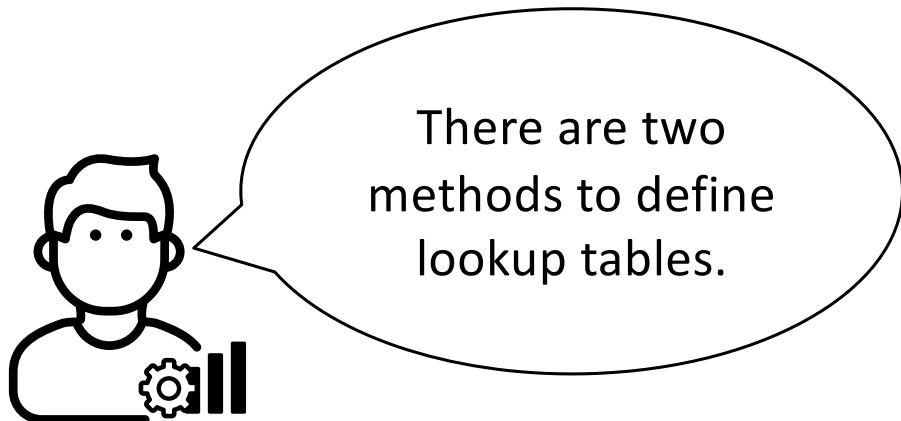
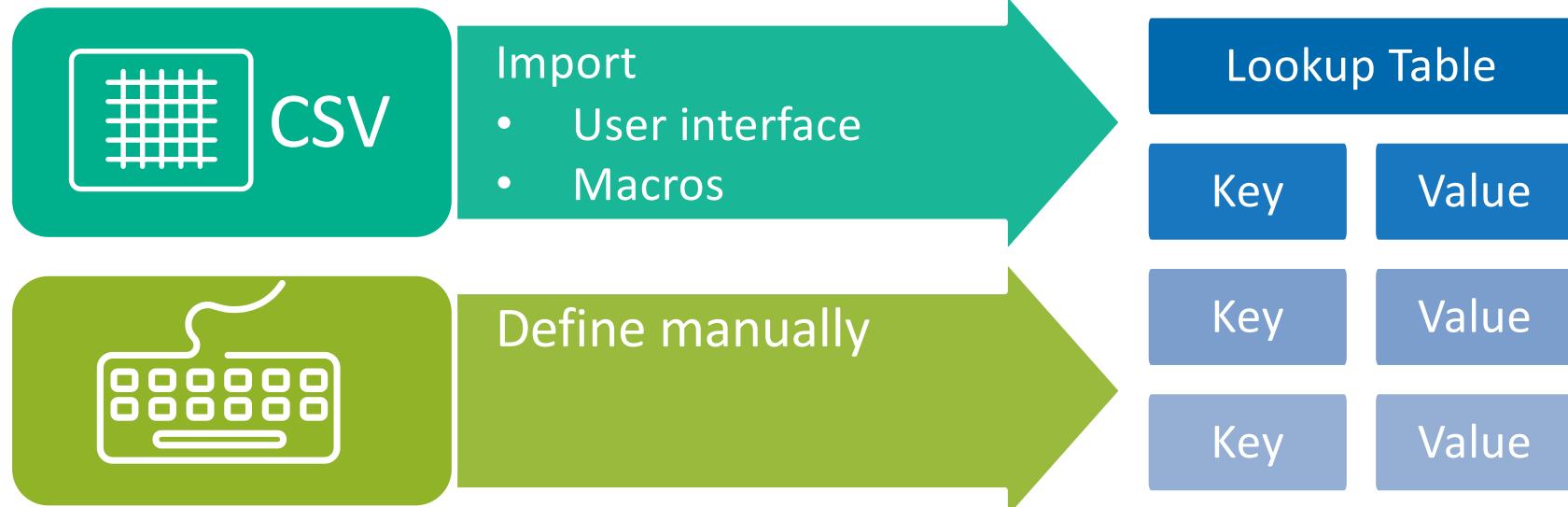
Construct Rules

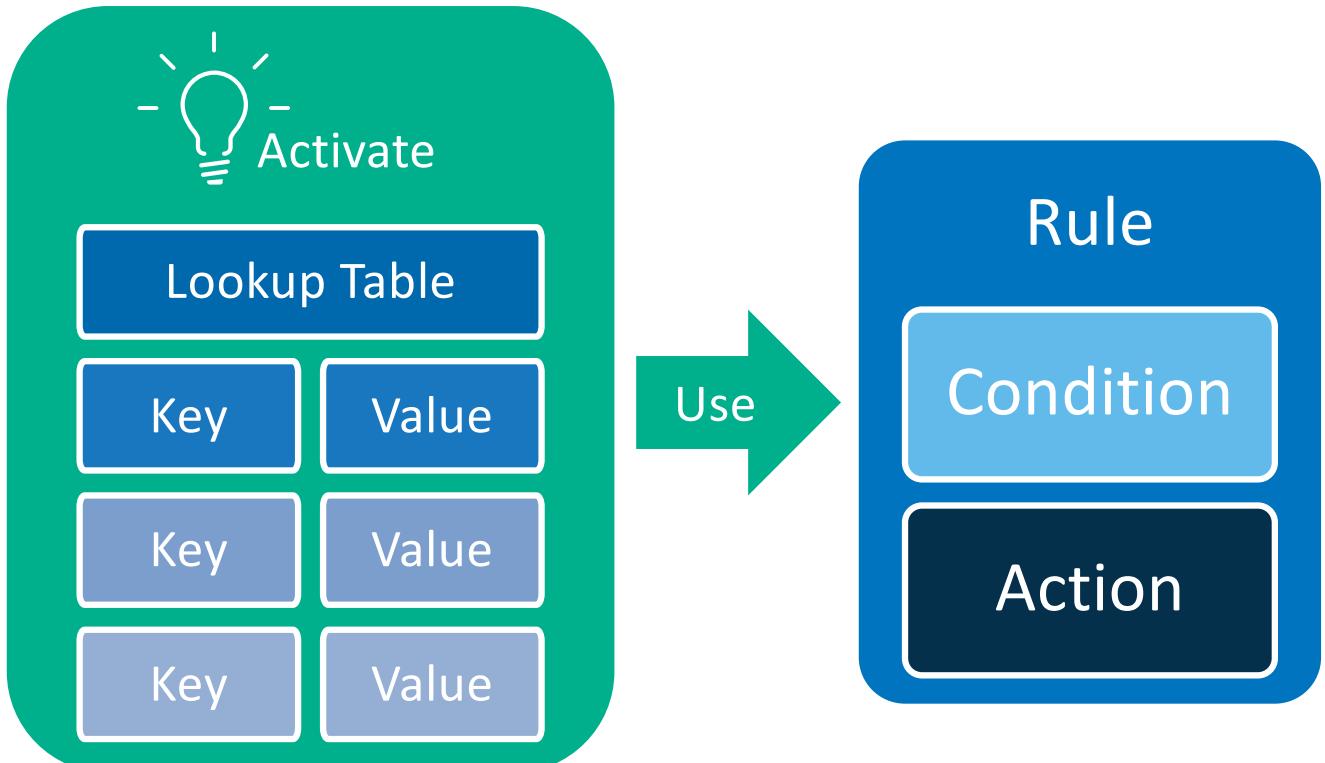
Rule set editor

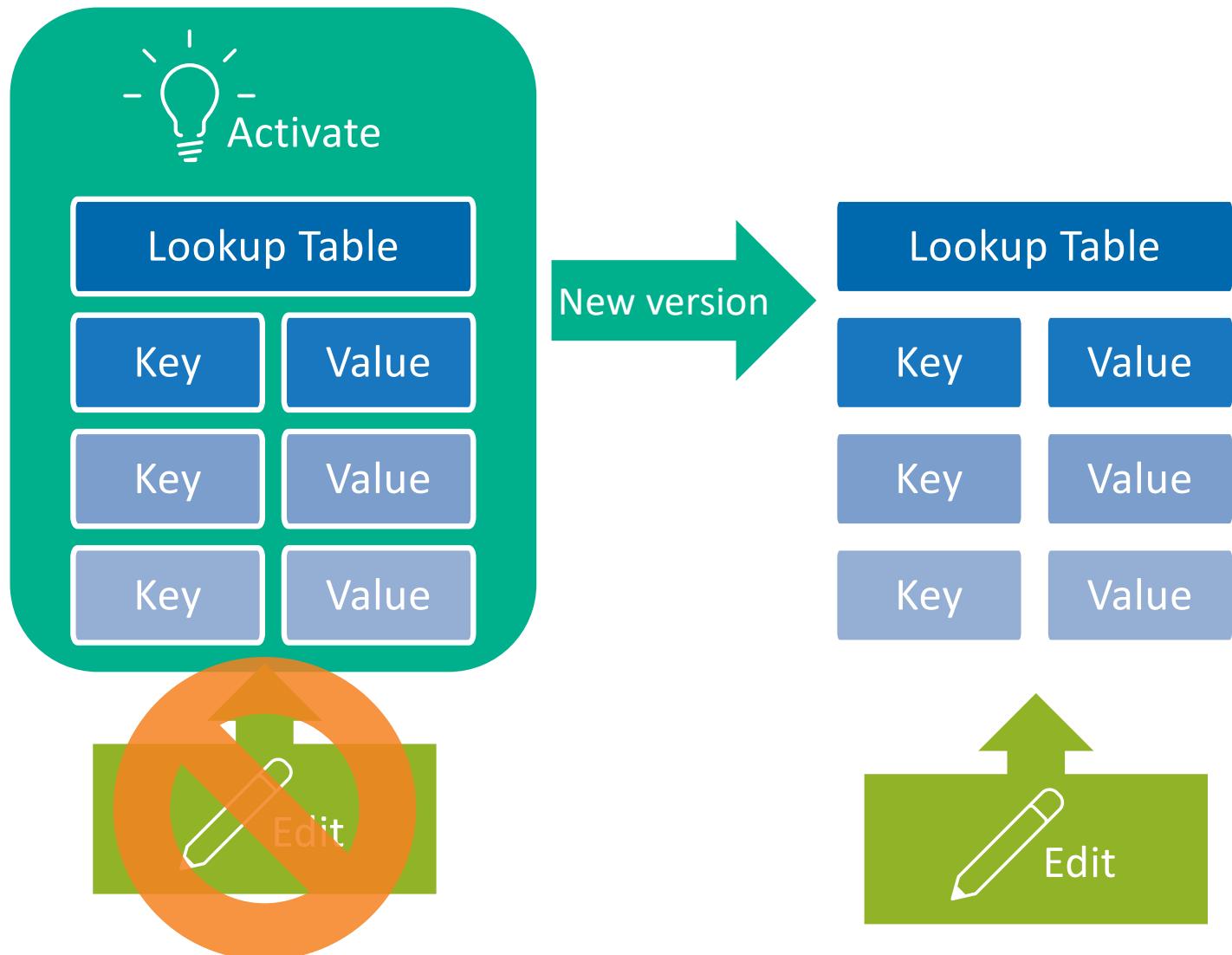
Expression Editor



LOOKUP
LOOKUP VALUE









4) Creating a Lookup Table

This demonstration illustrates how to create a lookup table.

1) Create Lookup Table

The screenshot shows the SAS Intelligent Decisioning - Build Decisions interface. On the left, a sidebar menu includes 'Decisions', 'Rule sets', 'Lookup tables' (selected), 'Treatments', 'Treatment groups', 'Code files', 'Custom functions', and 'Global variables'. The main area is titled 'Lookup Tables' and contains a search bar and a list of existing lookup tables: 'Subject Level', 'Treatment Channels', 'LoanTypeSolution', 'ClaimTypeSolution', 'Solution Registered Me...', and 'ReferralDiscountSolution'. A modal window titled 'New Lookup Table' is open, prompting for 'Name' (set to 'LoanType'), 'Description' (empty), and 'Location' (set to '/My Folder/Decisioning/Demonstrations'). Buttons for 'Save' and 'Cancel' are at the bottom.

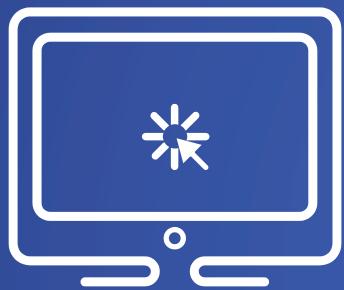
2) Import CSV

This block shows the 'Import Lookup Table' dialog. The 'Name' field is set to 'LoanTypeLookup.csv'. Under 'Encoding', 'Select encoding' is selected with 'UTF-8 (Default)' chosen. Buttons for 'Import' and 'Cancel' are at the bottom. To the right, the 'LoanType (1.0)' table is displayed with two columns: 'A' and 'B'. The data rows are:

A	B
1 DC	Debt Consolidation
2 HI	Home Improvement
3 AF	Auto Financing
4 BU	Business
5 WE	Wedding
6 ME	Moving Expense
7 MD	Medical Expense

3) Activate

The screenshot shows the 'Activate' dialog box. It contains the message: 'The active version is the version that is used when you run a test for a rule set or decision that uses the lookup table.' Below this is the question: 'Do you want to create a new minor version and activate version 1.0?'. At the bottom are 'Yes' and 'No' buttons, and the word 'Wedding' is visible at the bottom right.



5) Using Lookup Table Functions in a Rule Set

This demonstration illustrates using the LOOKUP and LOOKUPVALUE functions in a rule set.

1) New version

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Demo_LoanApplication (1.0)

Rule Set Properties Variables Scoring Versions History

New Version Set Version

Version	Displayed Version	Notes	Date Modified	Modified By
1.0	✓		Jun 11, 2023 12:58 AM	lynn

2) Add variables from decision & custom variables

Add Variables

Decision: /My Folder/Decisioning/Demonstrations/Demo_AddRuleSet

Available items (0 of 19):

PURPOSE character-inO

Search variable	Selected items
purpo	No items

* Demo_LoanApplication (1.1)

Rule Set Properties Variables Scoring Versions History

Rule Set Variables Global Variables

Search variable	Data Type	Input	Output
Variables	Character	<input type="checkbox"/>	<input checked="" type="checkbox"/>
AppStatus	Character	<input type="checkbox"/>	<input checked="" type="checkbox"/>
DELINQ	Integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
DEROG	Integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
PURPOSE	Character	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
PurposeDescription	Character	<input type="checkbox"/>	<input checked="" type="checkbox"/>

3) Add Lookup table in Rule Set

SAS® Intelligent Decisioning - Build Decisions

Demo_LoanApplication (1.1)

Rule Set Properties Variables Scoring Versions History

Add ▾

Reject high delinquencies or derogatory reports Record rule-fired data

IF DELINQ > 2 OR DEROG > 4

THEN ASSIGN AppStatus 'REJECT'

Assign purpose description Record rule-fired data

IF PURPOSE LOOKUP LoanType

THEN LOOKUPVALUE PurposeDescription LoanType

ELSE

THEN ASSIGN AppStatus 'REJECT'

+ Add Rule

The screenshot shows the SAS Intelligent Decisioning - Build Decisions software interface. On the left, there's a sidebar with options like Decisions, Rule sets (which is selected), Lookup tables, Treatments, Treatment groups, Code files, Custom functions, and Global variables. The main area is titled 'Demo_LoanApplication (1.1)' and shows a 'Rule Set' tab. There are two sections of rules: 'Reject high delinquencies or derogatory reports' and 'Assign purpose description'. The 'Assign purpose description' section is highlighted with a red box. It contains an IF condition 'PURPOSE LOOKUP LoanType', a THEN block 'LOOKUPVALUE PurposeDescription LoanType', an ELSE block, and another THEN block 'ASSIGN AppStatus 'REJECT''. There are also 'Record rule-fired data' checkboxes for both sections.

4) Change version in Decision Flow

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

Demo_AddRuleSet (1.0)

Decision Flow Decision Properties Variables Scoring Versions

Start → GetHMEQdata (1.0) → Demo_LoanApplication (1.0)

Properties

Name: Demo_LoanApplication

Description:

Version: 1.0

Select an item
1.0
1.1

5 Test

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

Demo_AddRuleSet (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Tests Scenarios Publishing Validation

New Test Run Properties

Name: Demo_AddRuleSet_Test_1

Demo_AddRuleSet (1.0) > Demo_AddRuleSet_Test_1

Output Table

AppStatus	DELINQ	DEROG	PURPOSE	PurposeDescription
REJECT	0	1		
REJECT	3	1	DC	
REJECT	2	2		
REJECT	2	0		
REJECT	1	3		

Actions

Lesson 2: Decision Basics

2.1 Variables

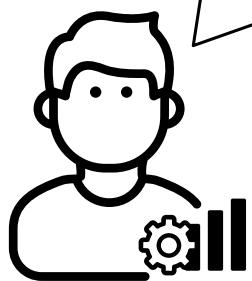
2.2 Rule Sets and Rules

2.3 Global Variables

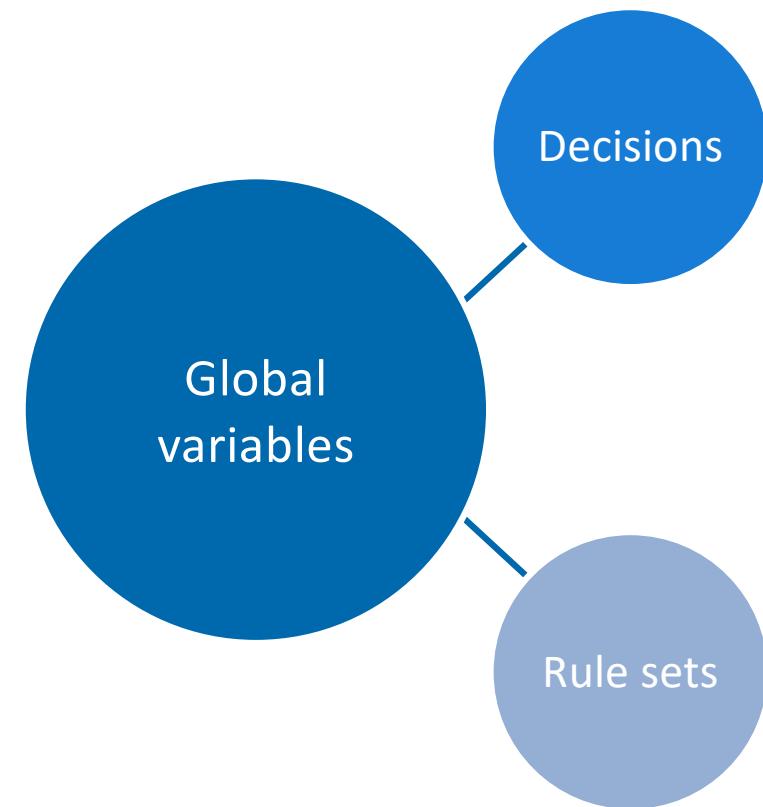
2.4 Branching and Cross-Branch Linking

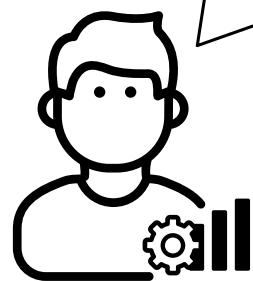
2.5 Record Contacts

2.6 Object Versioning



You can use global
variables in
decisions and rule
sets.



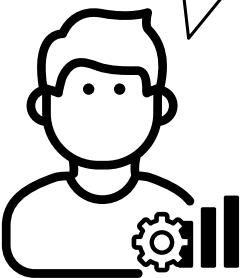


Global variables are defined and maintained in one place.

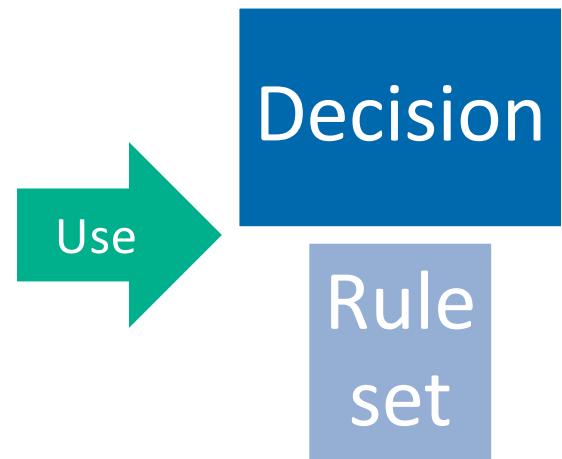
Global variables



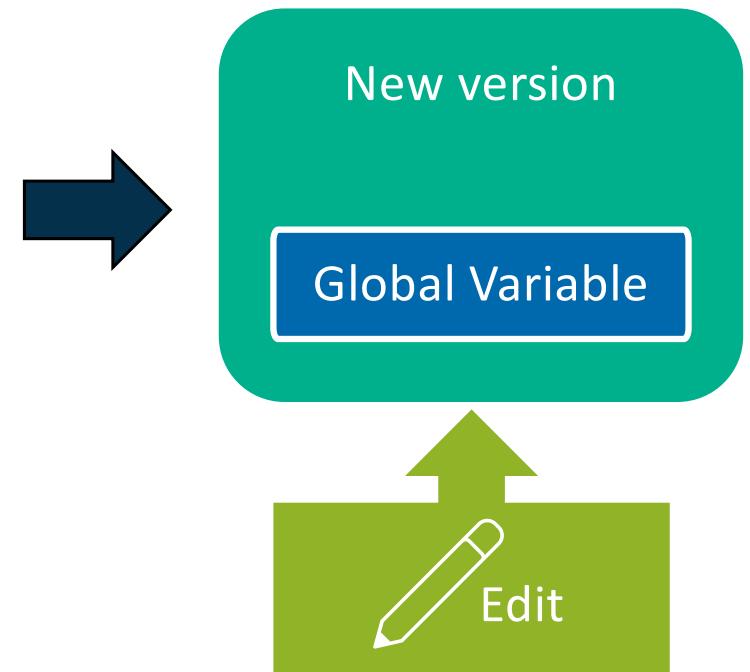
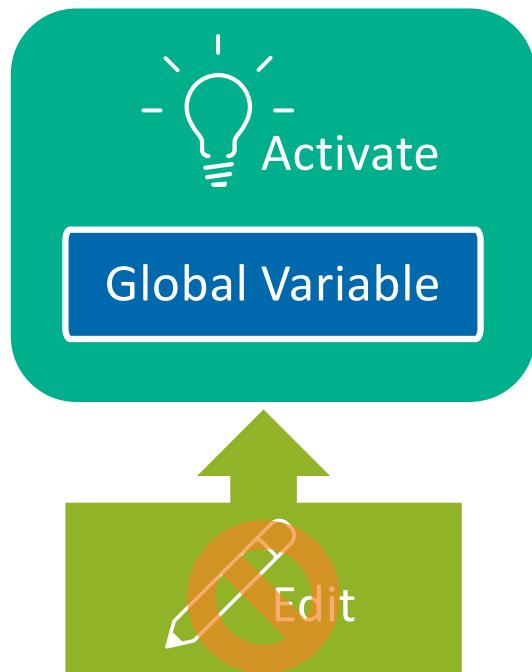
- Interest rate
- Exchange rate
- Inventory level



You must **activate** the variable before using it.



Activated in all configured publishing destinations.





6) Creating and Using a **Global Variable**

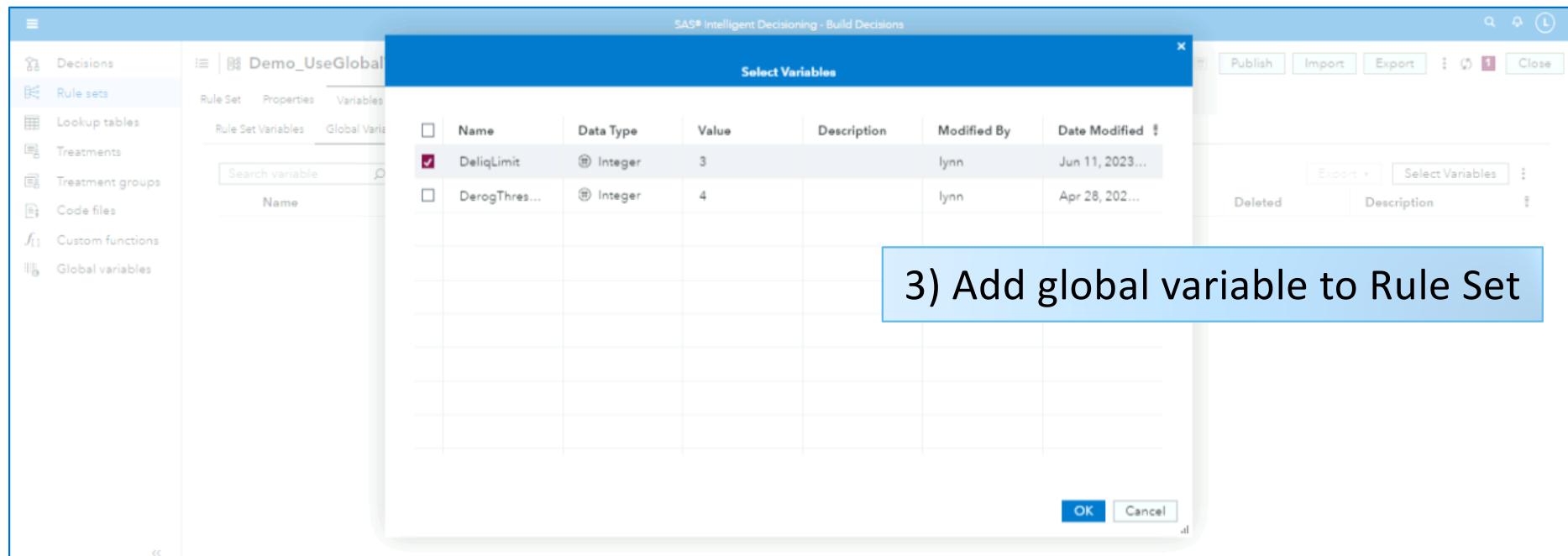
This demonstration illustrates how to create and use a global variable.

1) New global variable

The screenshot shows the 'Global Variables' section of the SAS Intelligent Decisioning interface. On the left, a sidebar lists various project components: Decisions, Rule sets, Lookup tables, Treatments, Treatment groups, Code files, Custom functions, and Global variables. The 'Global variables' option is selected. The main area displays a table of existing global variables with columns for Name, Type, Description, Last modified, and Properties. A modal dialog titled 'New Global Variable' is open, prompting for a Name (set to 'Global_Variable_1'), Data type (set to 'Boolean'), and Value (set to 'True'). The 'Save' button is visible at the bottom right of the modal.

2) Activate global variable

The screenshot shows the 'Edit Global Variable' dialog for version 1.0. The 'Activate' button is highlighted. A confirmation message box is displayed, asking if the user wants to create a new minor version and activate version 1.0. The message states: 'This version cannot be activated because it is not locked. Do you want to create a new minor version and activate version 1.0?'. The 'Yes' button is selected. The background shows the global variable properties and versions table.



3) Add global variable to Rule Set

The screenshot shows the rule editor for the 'Demo_UseGlobalVariable' rule set. The left sidebar lists the same categories as the previous screenshot. The main area shows a single rule: 'Reject high delinquencies or derogatory reports'. The condition part of the rule is 'IF DELINQ > DeliqLimit OR DEROG > 4'. The action part is 'THEN ASSIGN AppStatus = 'REJECT''. A checkbox labeled 'Record rule-fired data' is checked. At the bottom left is a '+ Add Rule' button. A blue box highlights the rule editor area with the text '4) Modify Rule Set'.

4) Modify Rule Set

Lesson 2: Decision Basics

2.1 Variables

2.2 Rule Sets and Rules

2.3 Global Variables

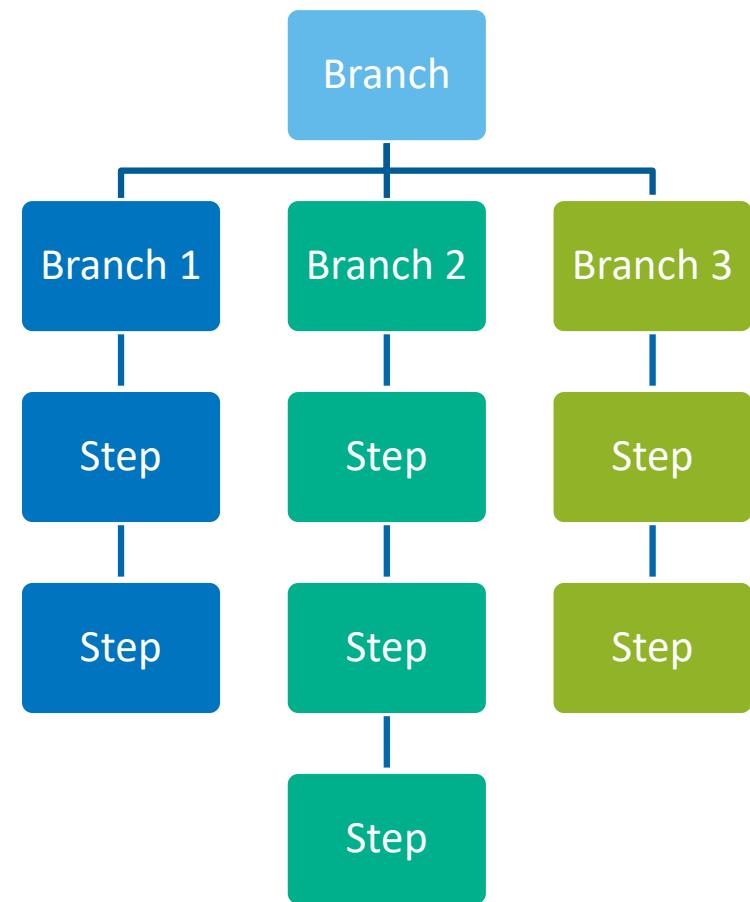
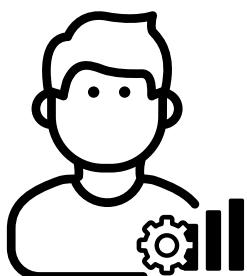
2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning

Branch

You use branch nodes to create separate paths in the decision flow.

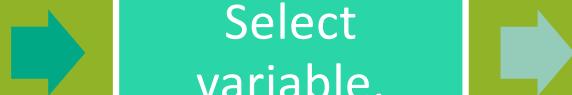


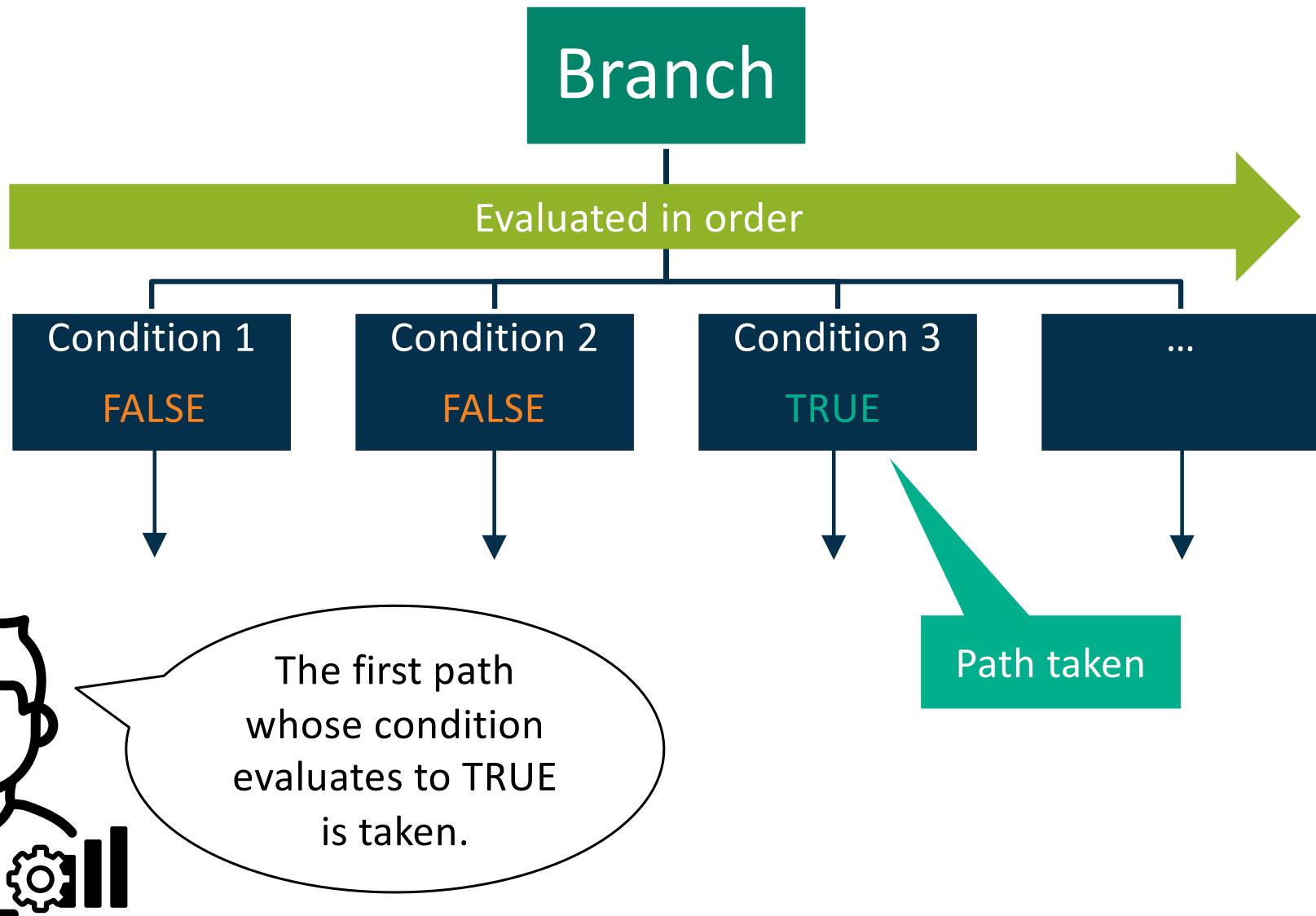
Branch Node

Select branch type.

Select variable.

Specify conditions.





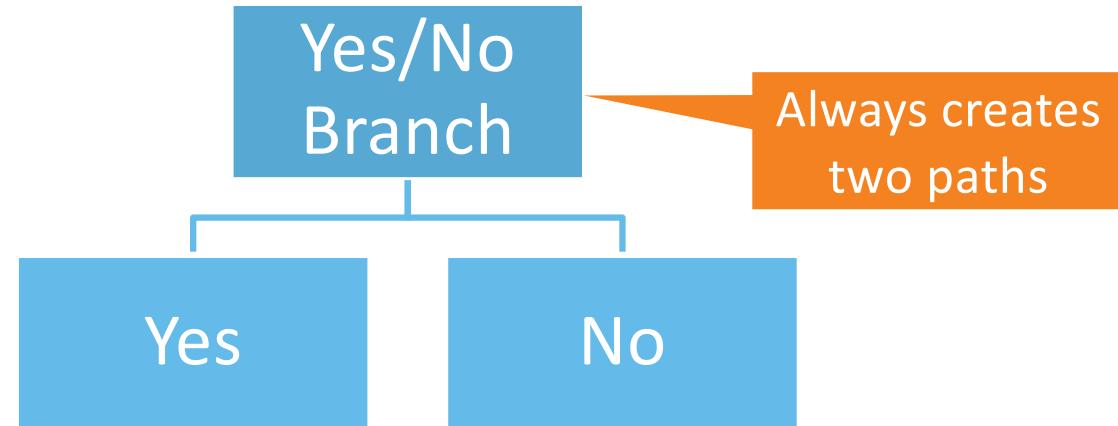
Branch Type

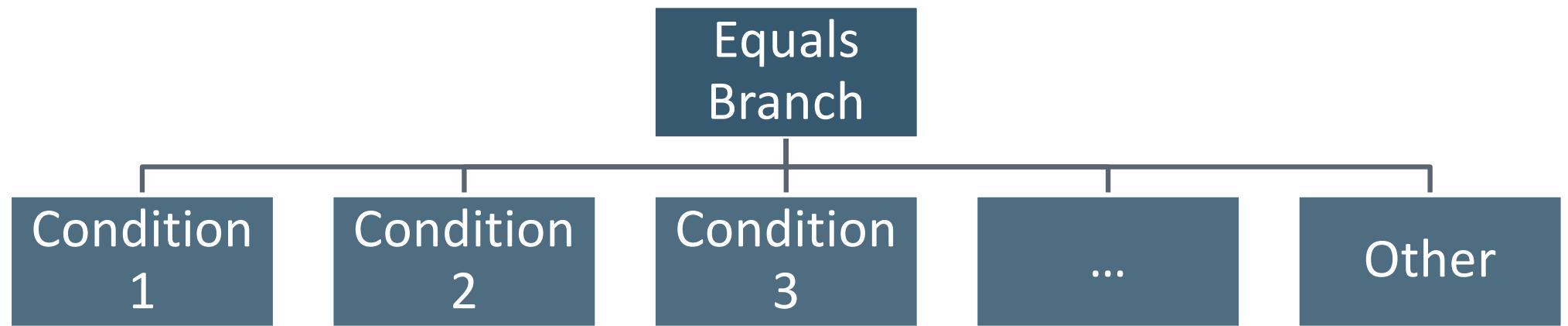
Yes/No

Equals

Range

LIKE



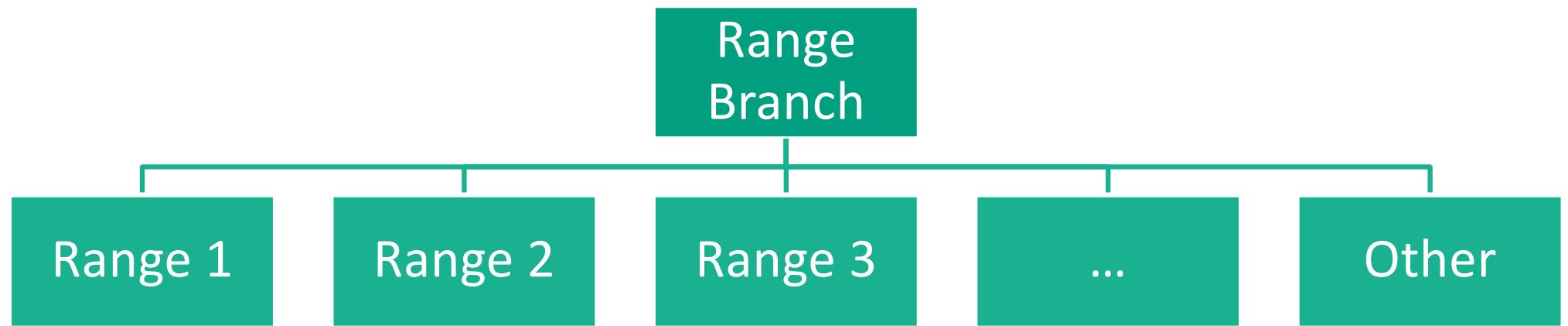


Conditions

Variable1 = Variable2

Variable = Value

Variable = Value1
OR
Variable=Value2

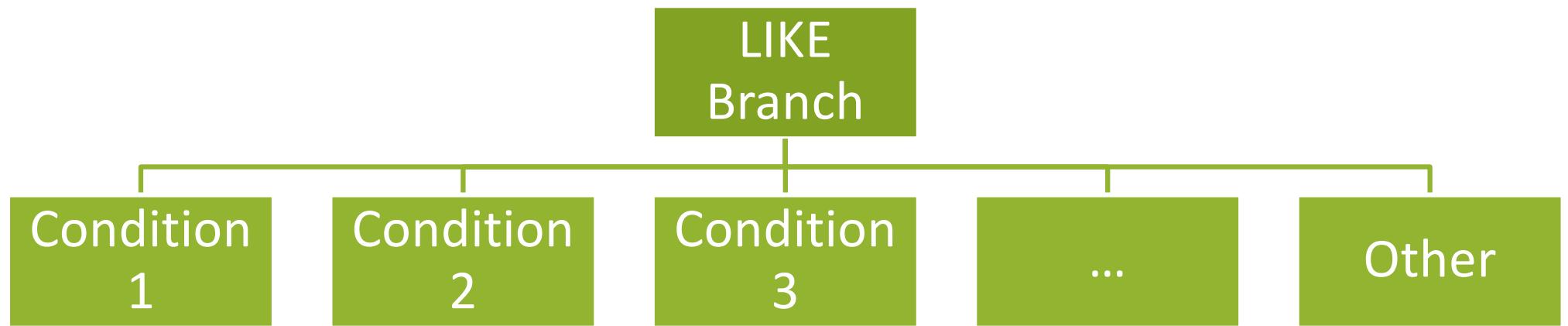


Minimum

- Variable
- Constant
- No minimum

Maximum

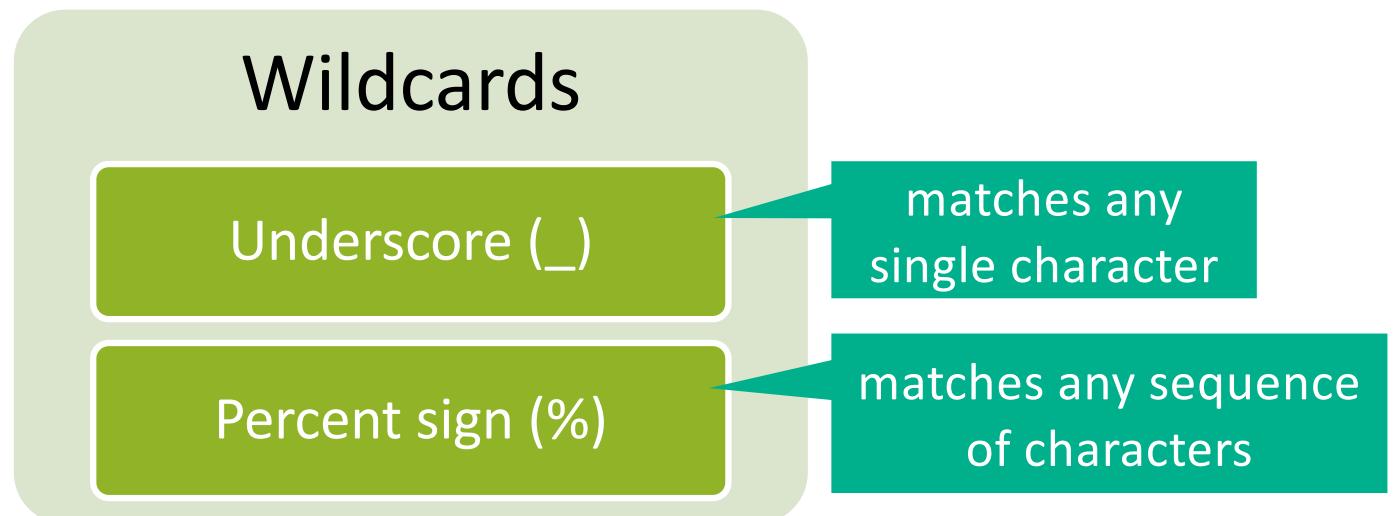
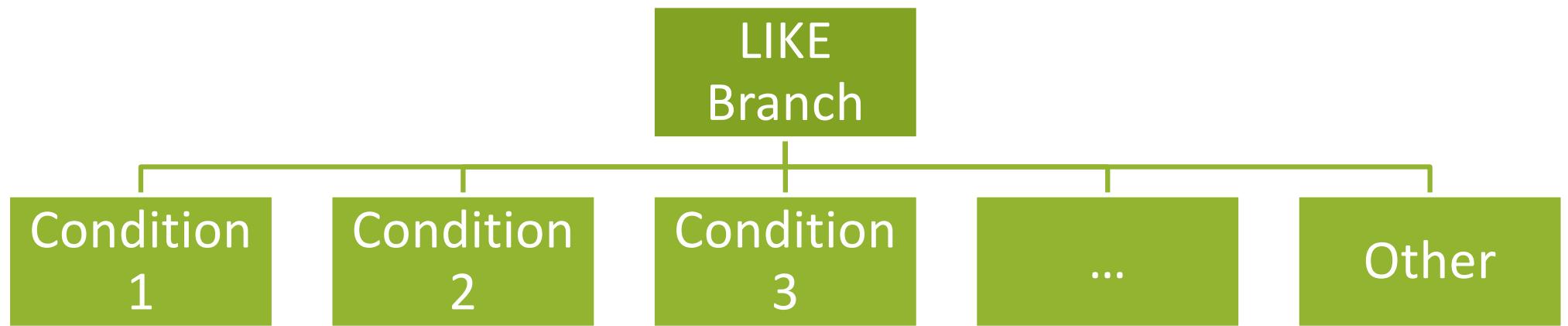
- Variable
- Constant
- No maximum

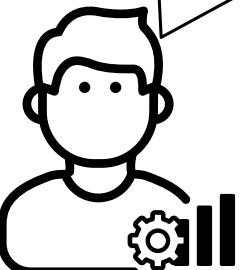


Conditions

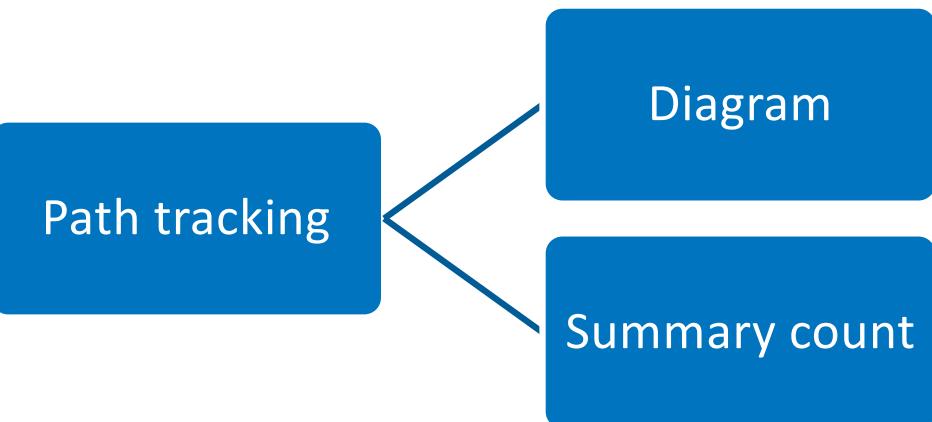
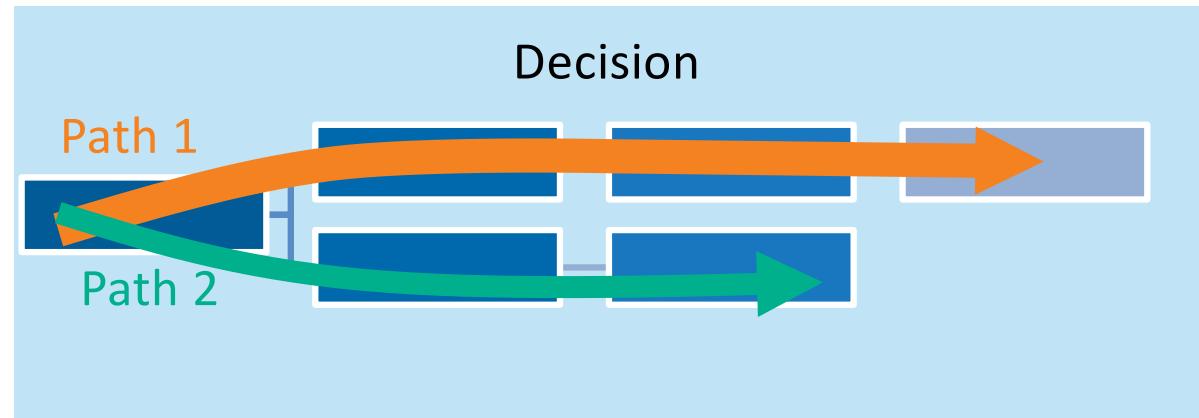
Variable LIKE String1

Variable LIKE String1
OR
Variable LIKE String2





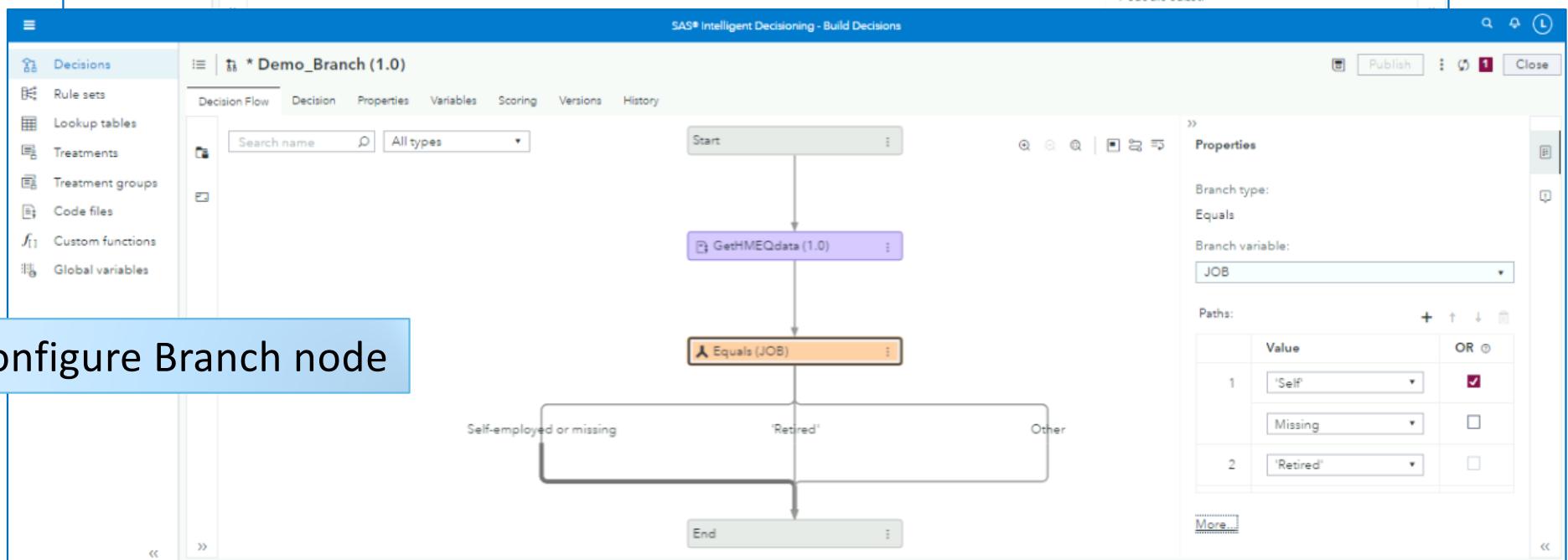
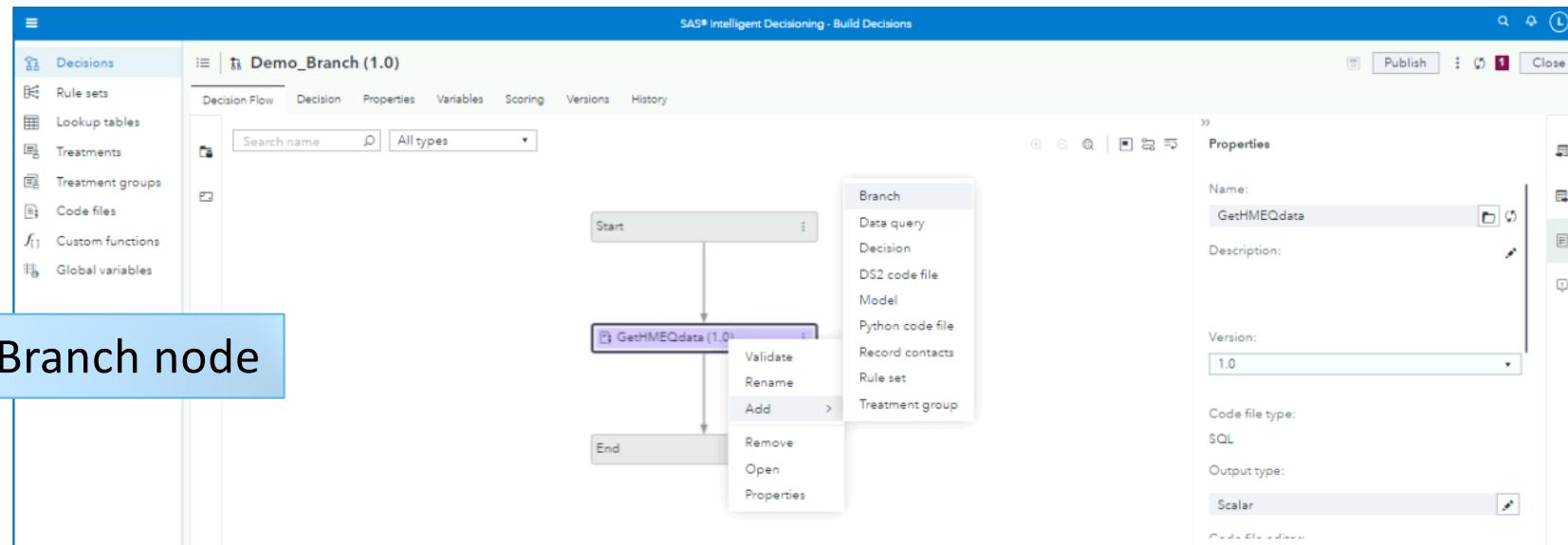
Path tracking provides information about the routes that records take through your decision.

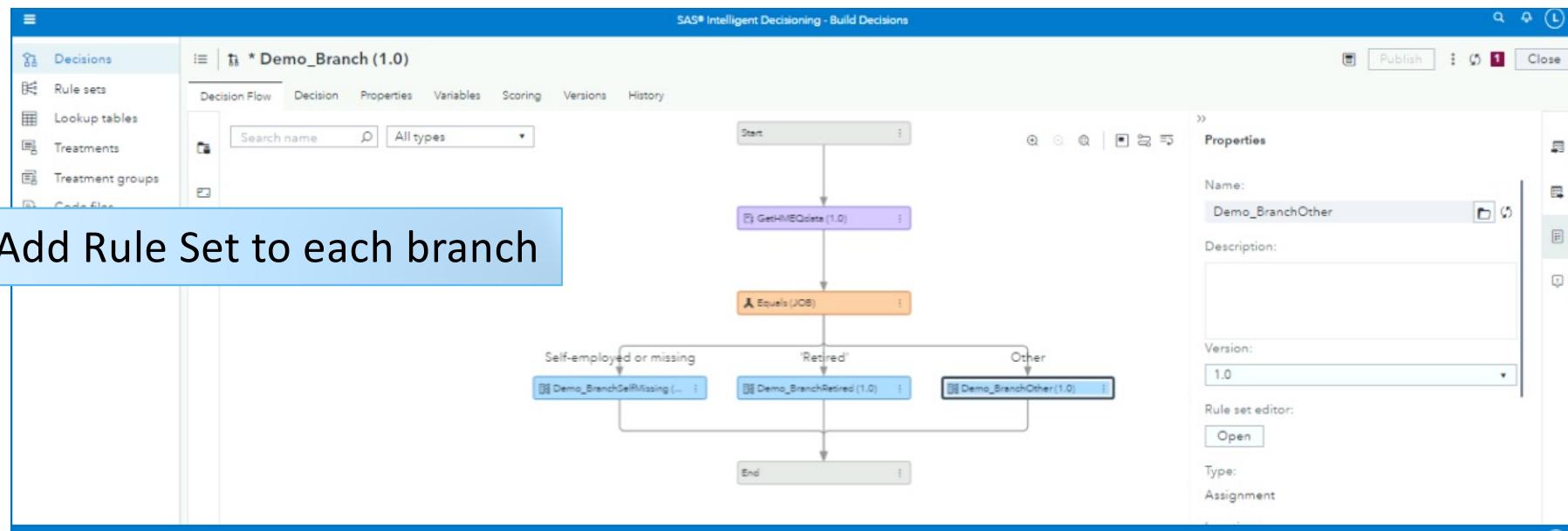




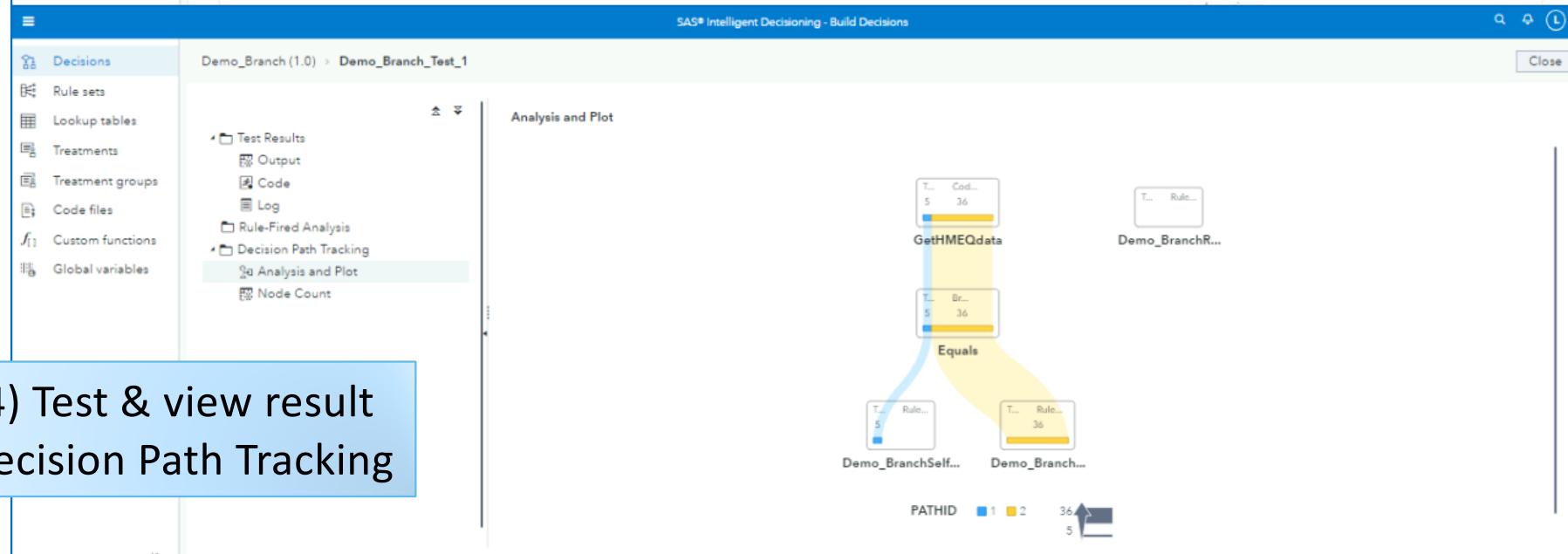
7) Configuring a Branch Node

This demonstration illustrates configuring an Equals branch.

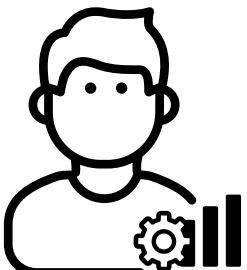




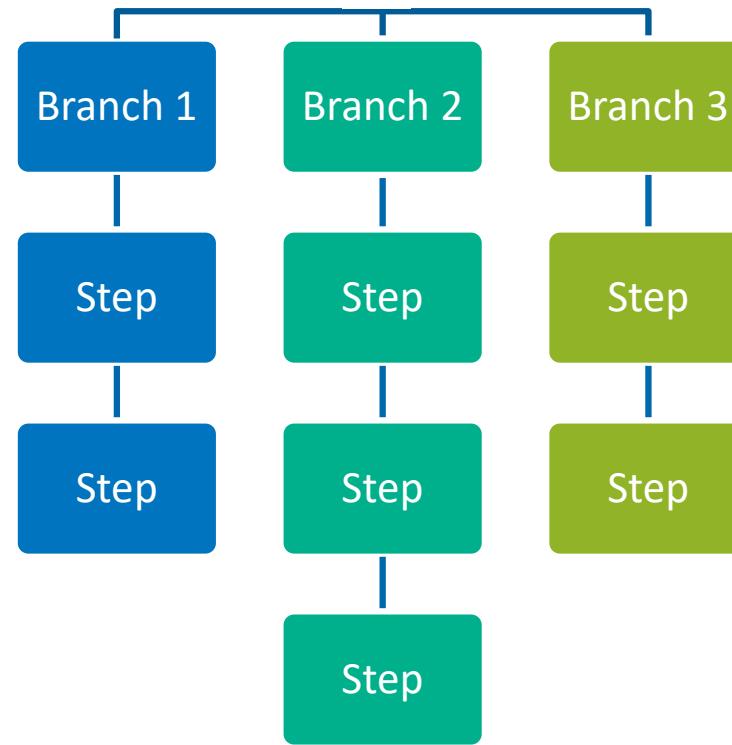
3) Add Rule Set to each branch



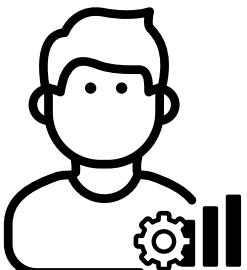
4) Test & view result
Decision Path Tracking



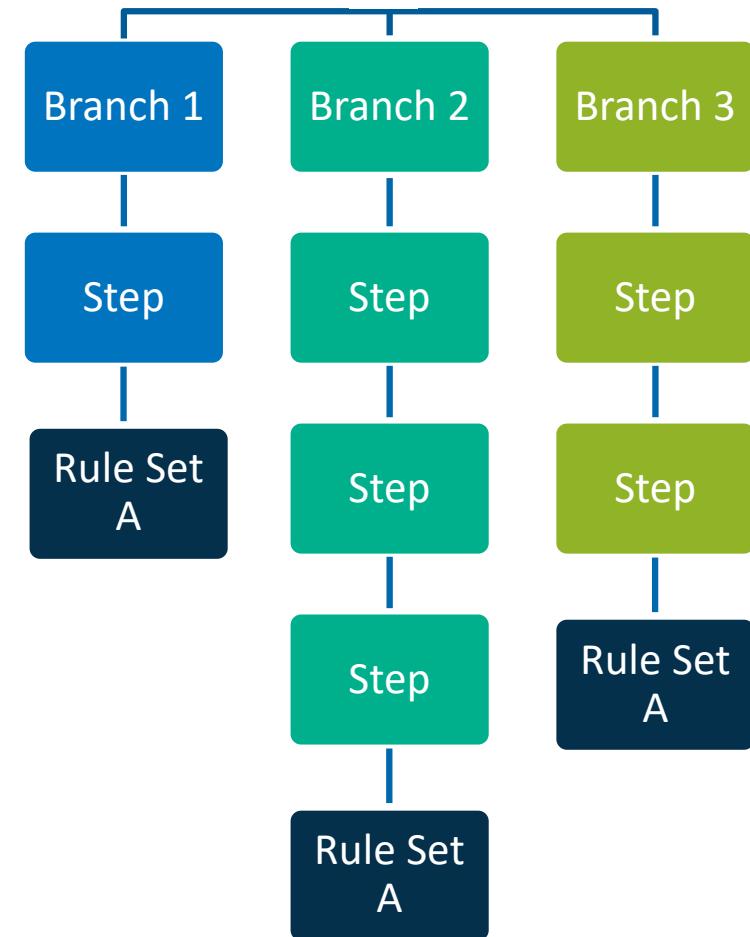
Cross-branch links
enable you to easily
reuse components
across branches.

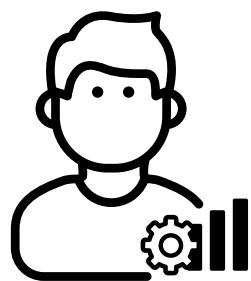


Easily reuse components



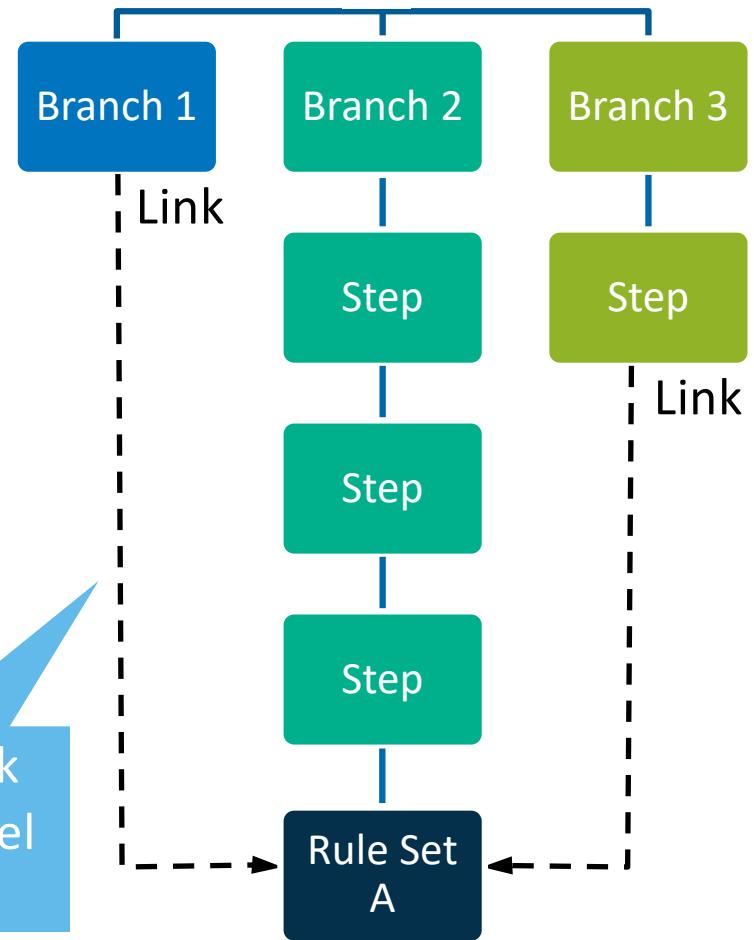
Suppose you want to
use a given rule set
across branches.





A simpler approach could be to use cross-branch links.

You can link to same level or lower.

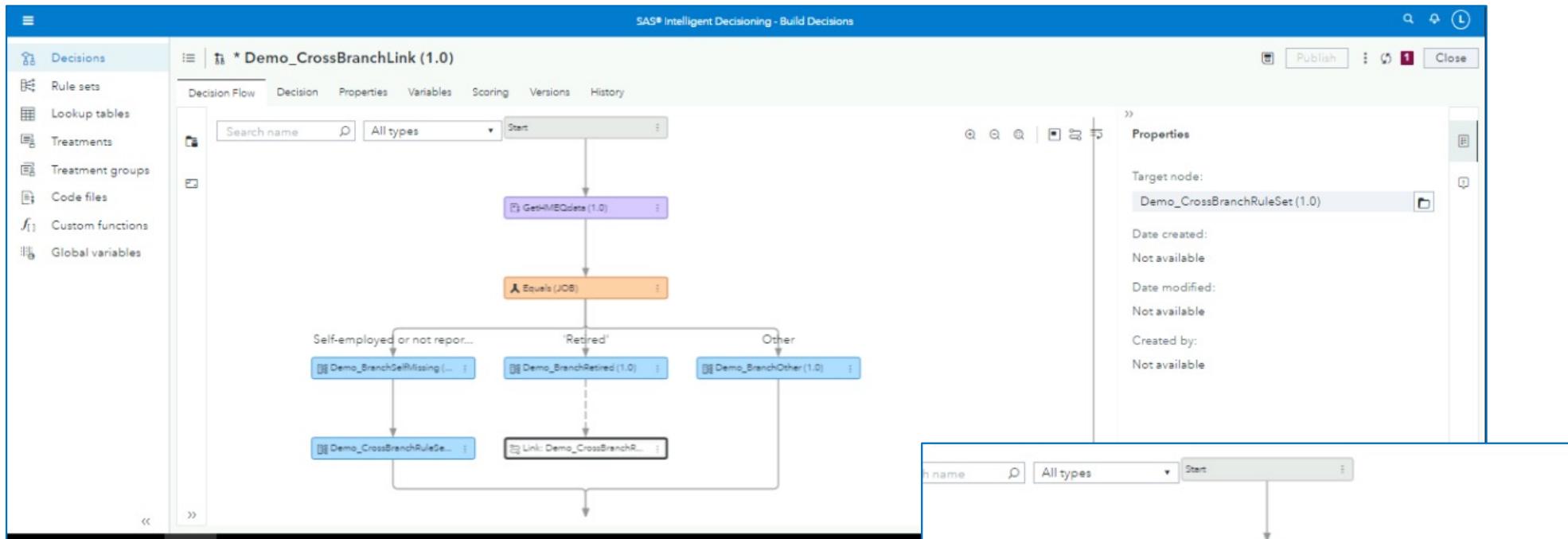




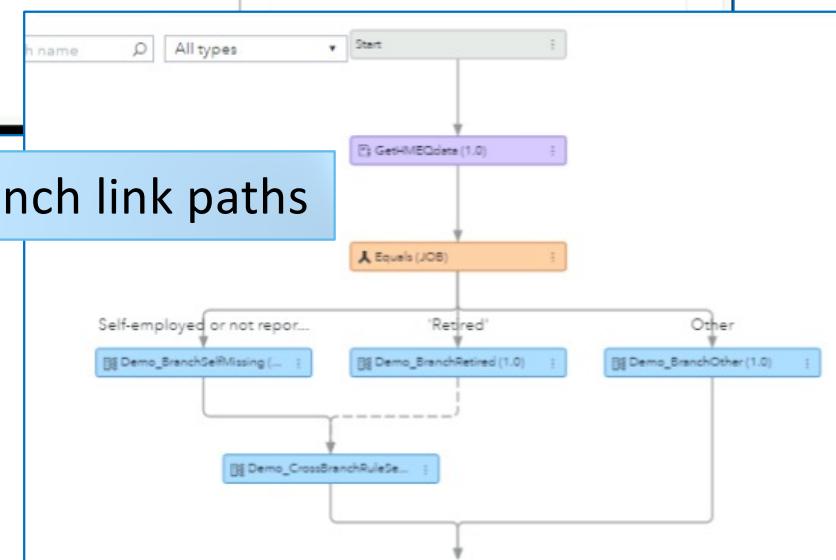
8) Configuring a Cross-Branch Link

This demonstration illustrates configuring a cross-branch link.

1) Add cross-branch link



2) Show cross-branch link paths



Lesson 2: Decision Basics

2.1 Variables

2.2 Rule Sets and Rules

2.3 Global Variables

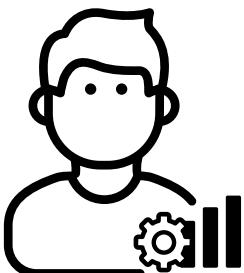
2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning

Record Contacts

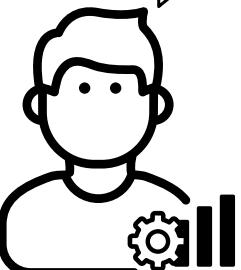
You can record contacts
to keep track of the
decision outcome.



Contact History

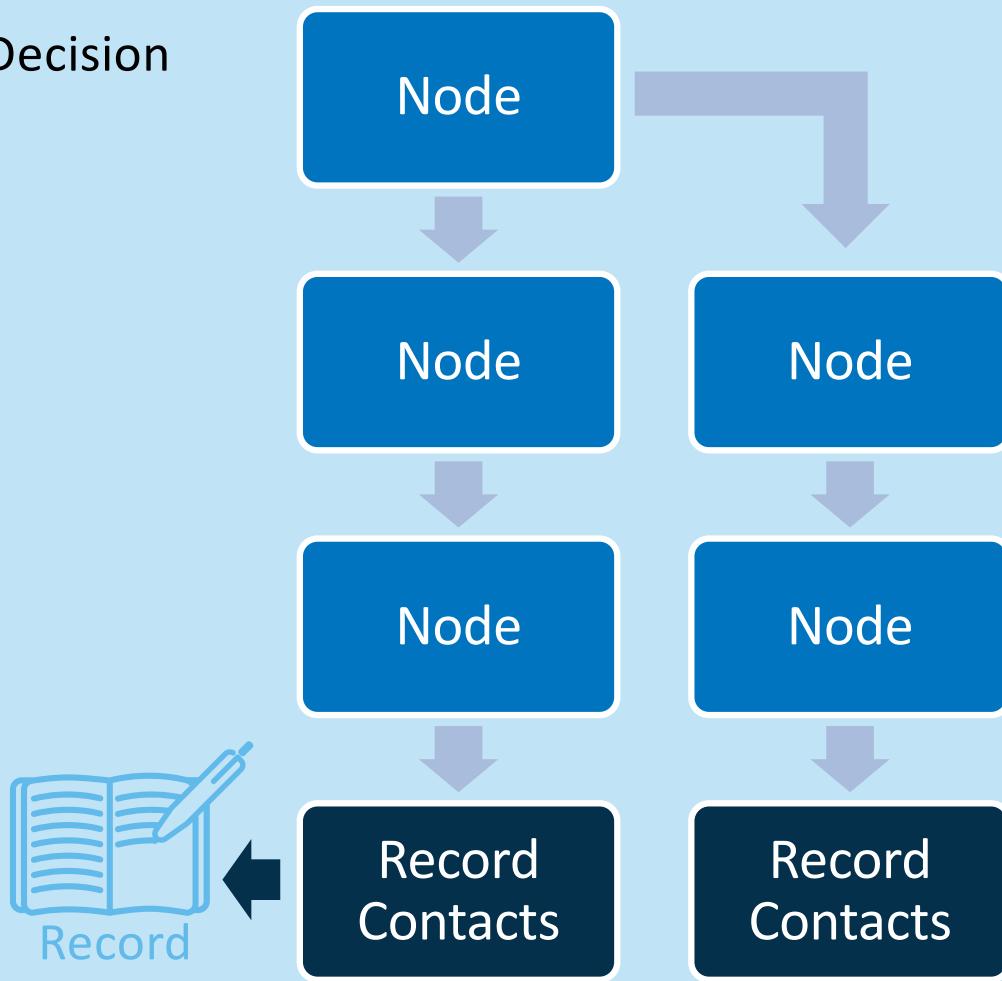


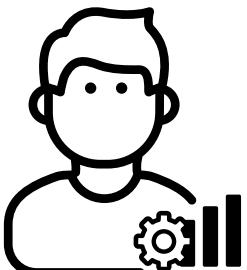
Decision Outcome



You might typically want to include this node toward the end of your business logic processing.

Decision



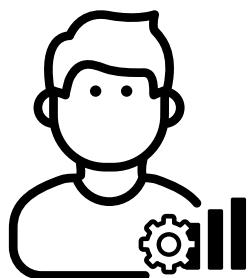


You define
these settings.

- Channel
- Variables to track
- Track treatments
- Record rule-fired data
- Record path tracking
- Include in contact policy

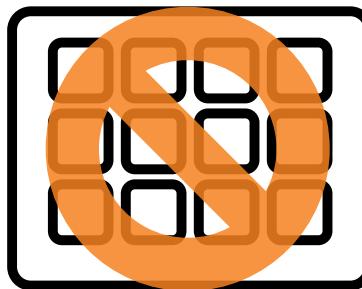
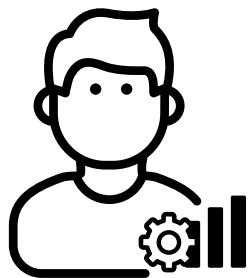
Channel

You can assign
a variable to
channel.

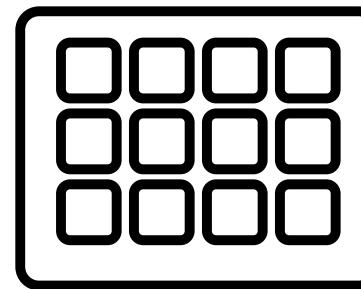


Variables to track

You choose
the variables
to track.

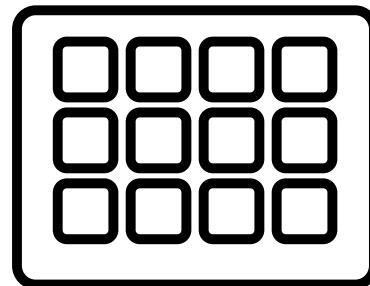


Data grid



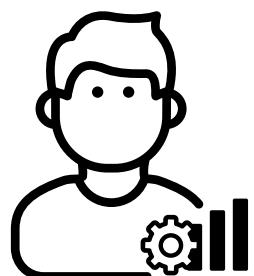
Treatment
data grid

Track Treatments



Treatment
data grid

You can select
a treatment
data grid.





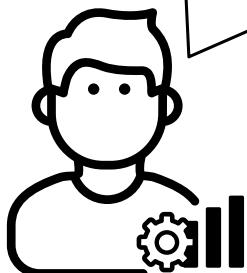
Record rule-fired data

Rule
conditions
true

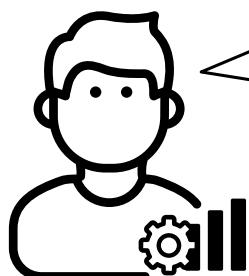
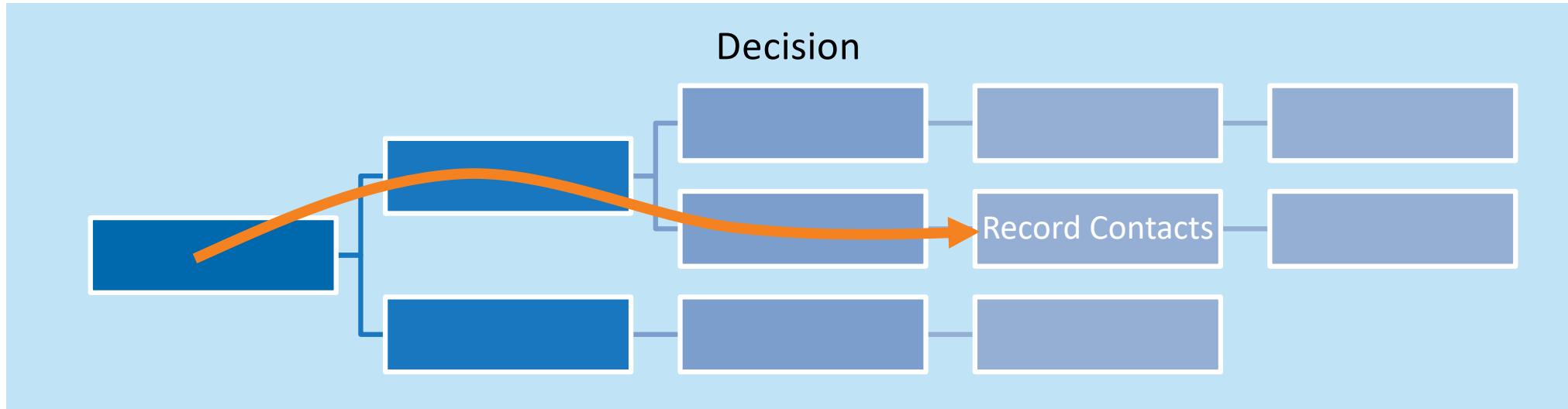


Rule fired

You can opt to
record rule-fired
data.



Record path tracking

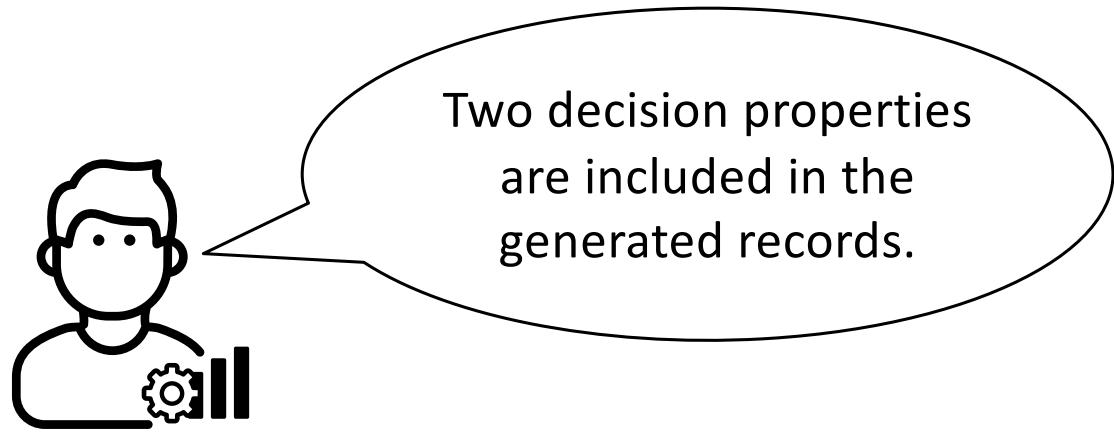


You can opt to
record path
tracking.

Decision Properties

Subject level

Subject ID



Decision Properties

Subject level

Subject ID

Subject Level



Household

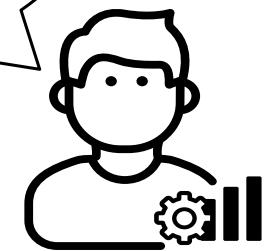


Customer



Account

The subject level
describes the
type of entity.



Decision Properties

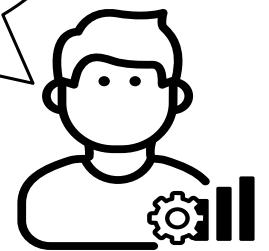
Subject level

Subject ID

Subject Level

Account

Subject ID is a value that uniquely identifies that entity.



Subject ID

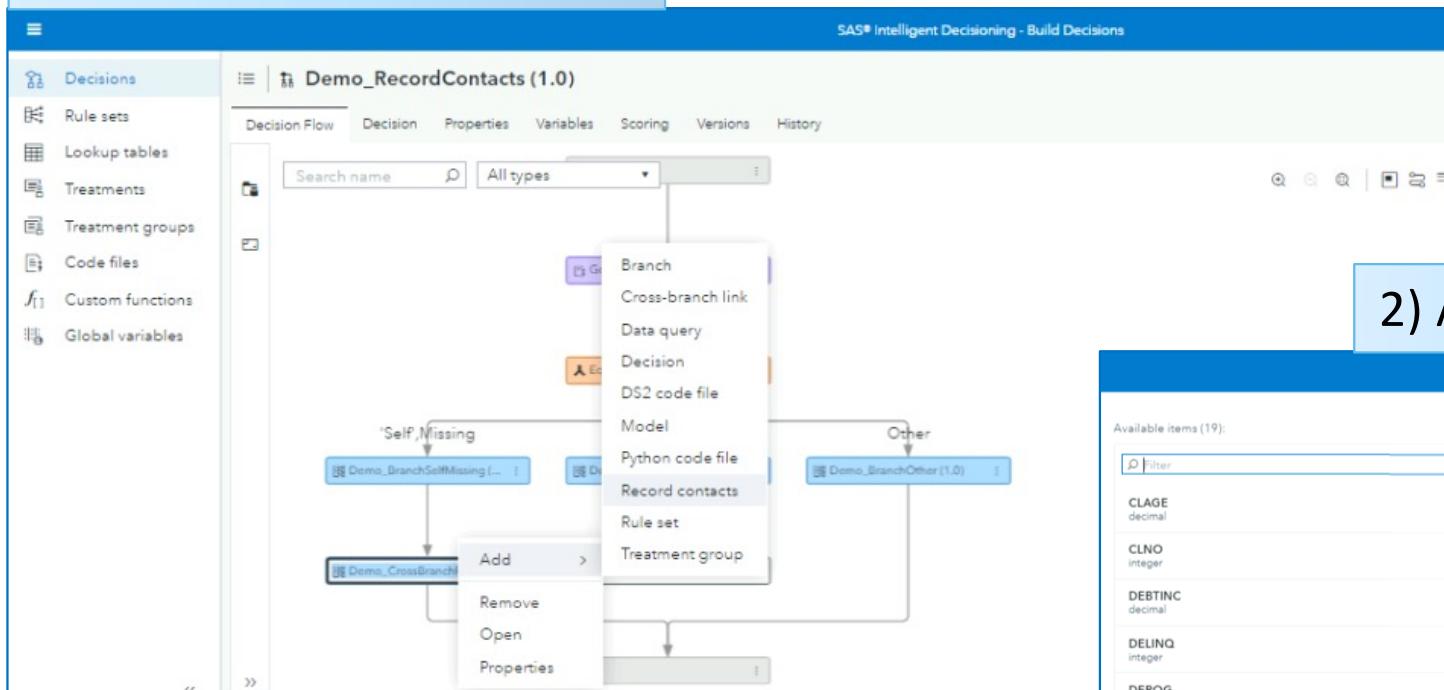
Account Number



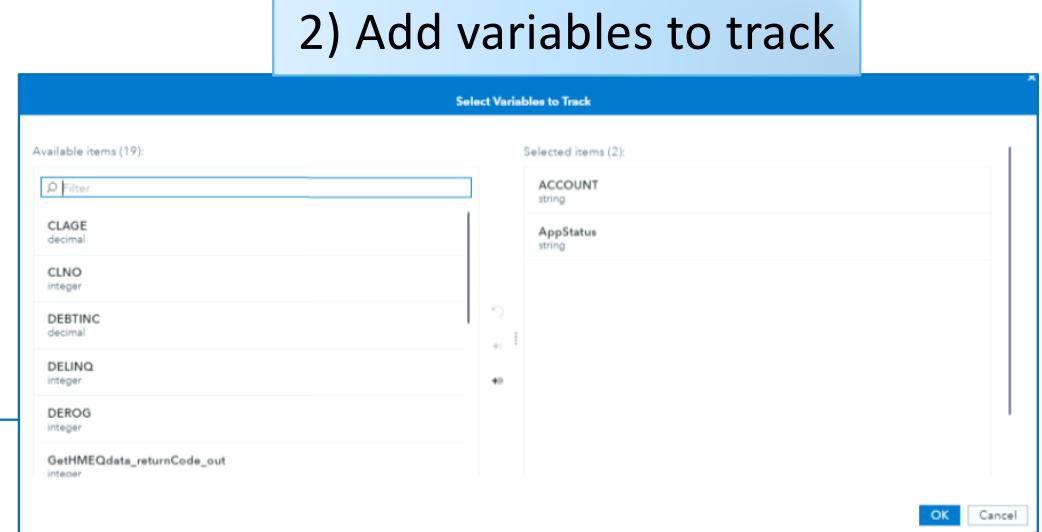
9) Configuring a Record Contacts Node

This demonstration illustrates configuring a record contacts node.

1) Add record contact node



2) Add variables to track



Lesson 2: Decision Basics

2.1 Variables

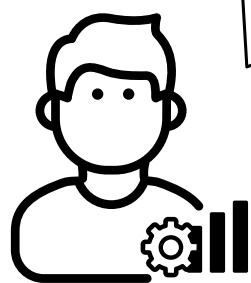
2.2 Rule Sets and Rules

2.3 Global Variables

2.4 Branching and Cross-Branch Linking

2.5 Record Contacts

2.6 Object Versioning



Many types of objects in
SAS Intelligent Decisioning
support versioning.

Queries created using
SAS Studio do not
support versioning.

Versioning

Decisions

Rule sets

Lookup tables

Global variables

Data queries

DS2 code

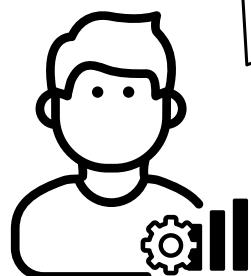
Python code

Treatments

Treatment groups

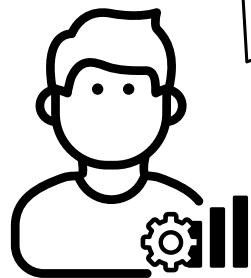


Versioning enables you to understand and track changes to objects.

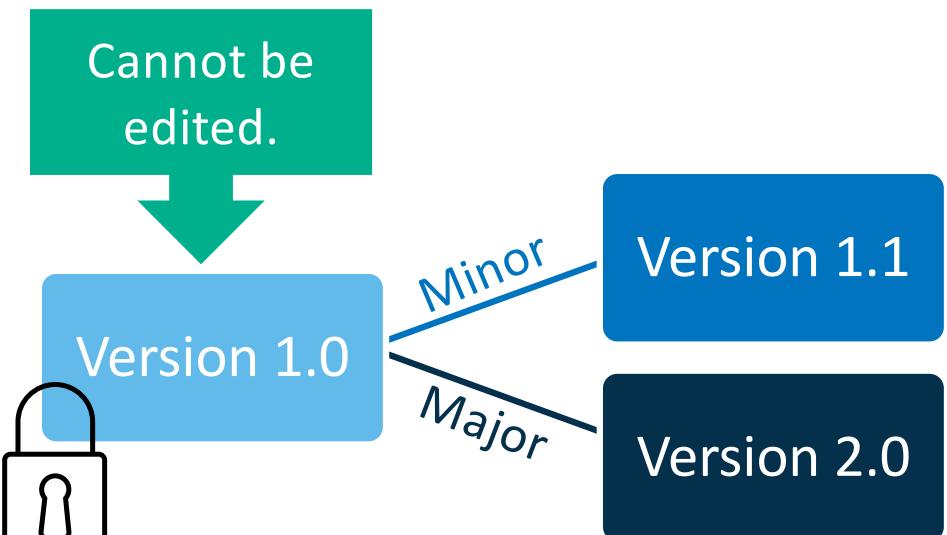


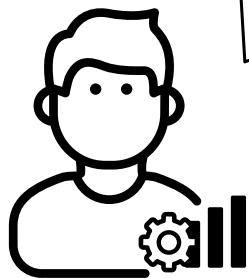
Understood

Trusted



You can create a new
major or minor version.



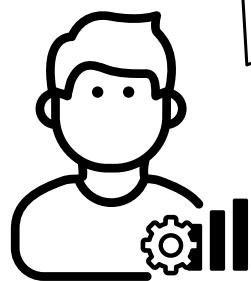


You can **perform these actions** for all objects that support versioning.

Set displayed version.

Copy a version.

Delete a version.



For some objects, you can **compare** the content or code for two versions.

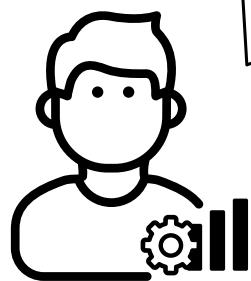
Compare versions

Decisions

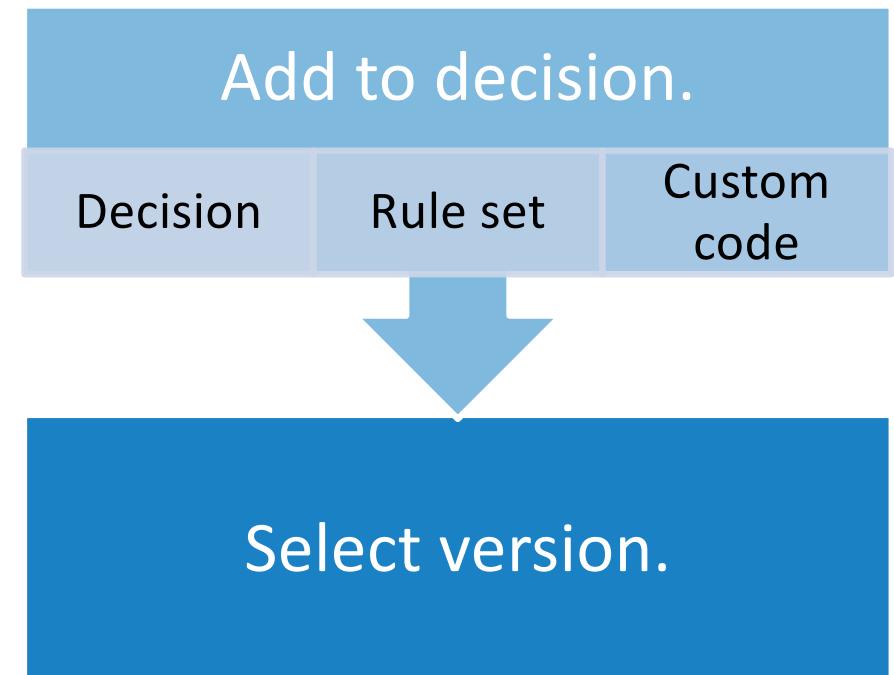
Rule sets

Treatment groups

Code files



After you add some types
of objects to a decision,
**you can choose the
version to use.**





10) Creating and Comparing Rule Set Versions

This demonstration illustrates creating a new version of a rule set and comparing versions.

1) New version

New Version

Version: 2.1

Type: Minor

Notes:

Save Cancel

2) Compare objects contents

3) Review the results & export

SAS® Intelligent Decisioning - Build Decisions

Demo_Versions (2.1) > Compare Object Contents

Rule Sets Variables

Show Differences Show All

Select Versions

Select the base version of "Demo_Versions": 2.0

Select the comparison version of "Demo_Versions": 1.0

Compare Cancel

Demo_Versions (2.0)

Reject high delinquencies or derogatory reports

```
IF DELINQ > 2 OR DEROG > DerogThreshold  
THEN ASSIGN AppStatus='REJECT'
```

Reject invalid purpose values

```
IF Lookup("LoanTypeSolution", PURPOSE)  
THEN LOOKUPVALUE PurposeDescription =LookupValue("LoanTypeSolution", PURPOSE)  
ELSE ASSIGN AppStatus='REJECT'
```

Demo_Versions (1.0)

Reject high delinquencies or derogatory reports

```
IF DELINQ > 2 OR DEROG > 4  
THEN ASSIGN AppStatus='REJECT'
```

Lesson 3: Advanced Topics

3.1 Custom Code

3.2 Models

3.3 Data Grids

Lesson 3: Advanced Topics

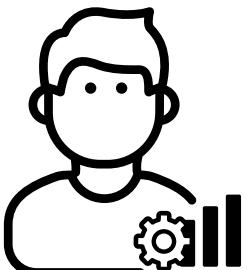
3.1 Custom Code

3.2 Models

3.3 Data Grids

Custom Code

SAS DS2
code

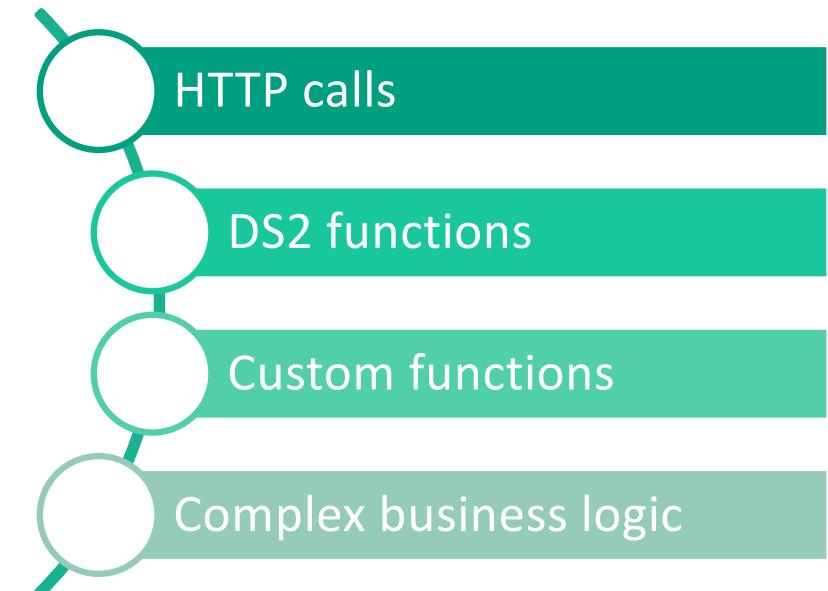


You can run
custom code in
a decision.

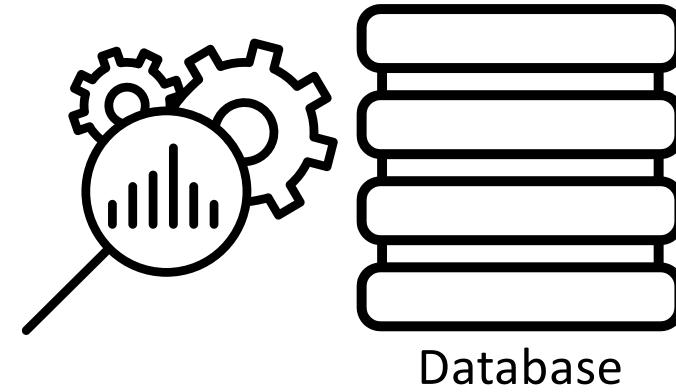
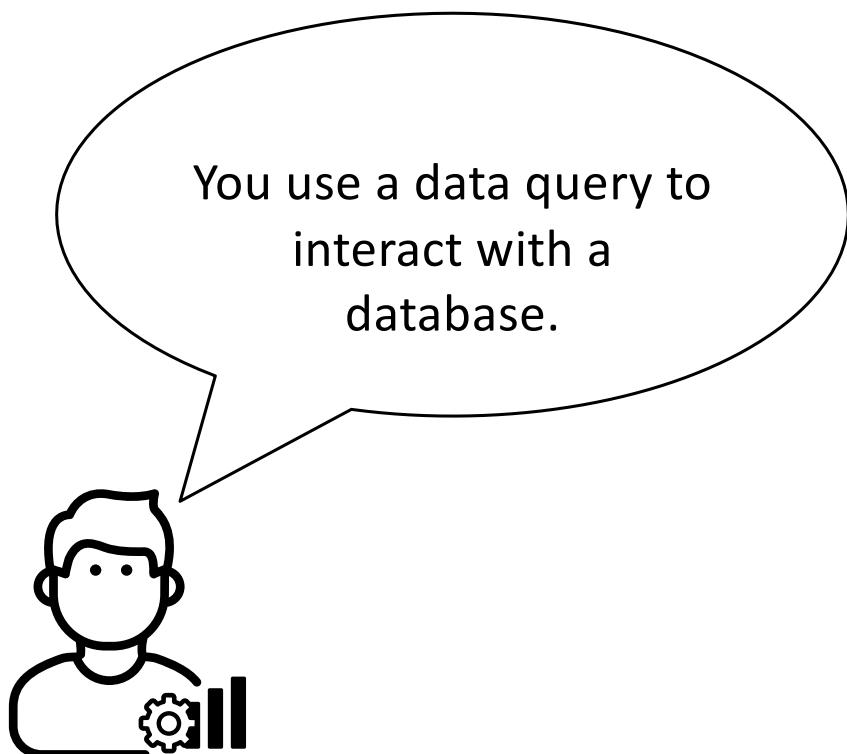
Data Query
(SQL)

Python
code

DS2 code



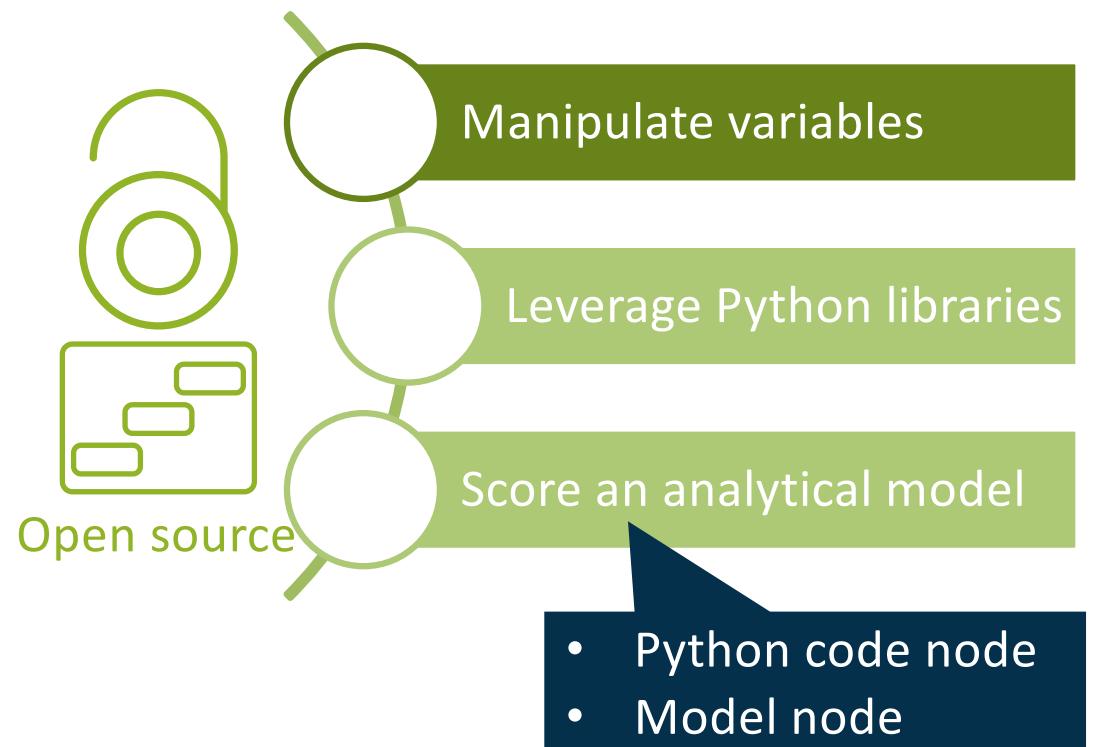
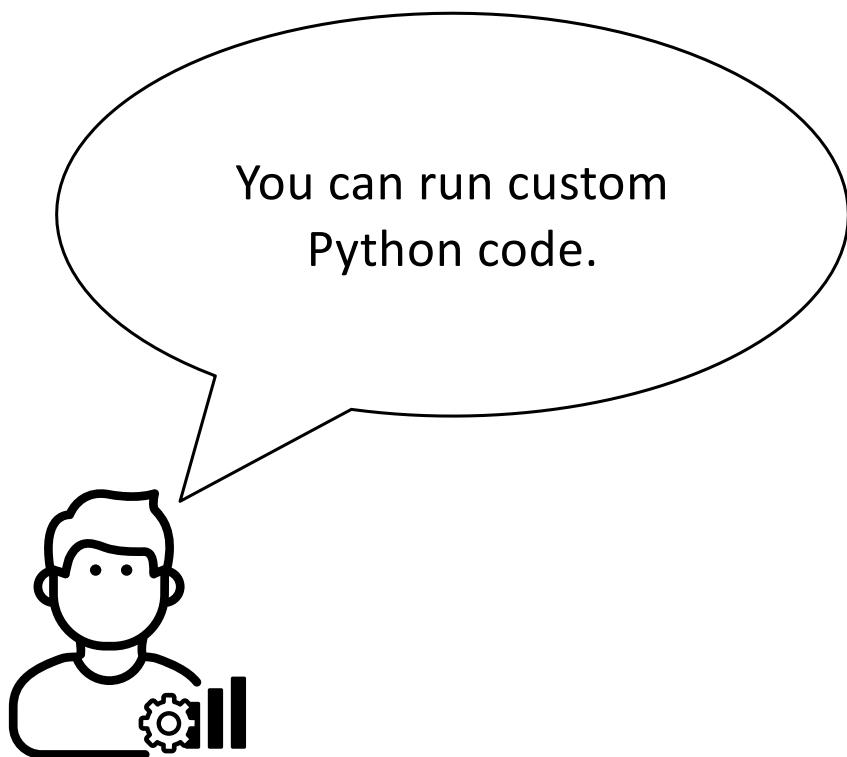
Data query

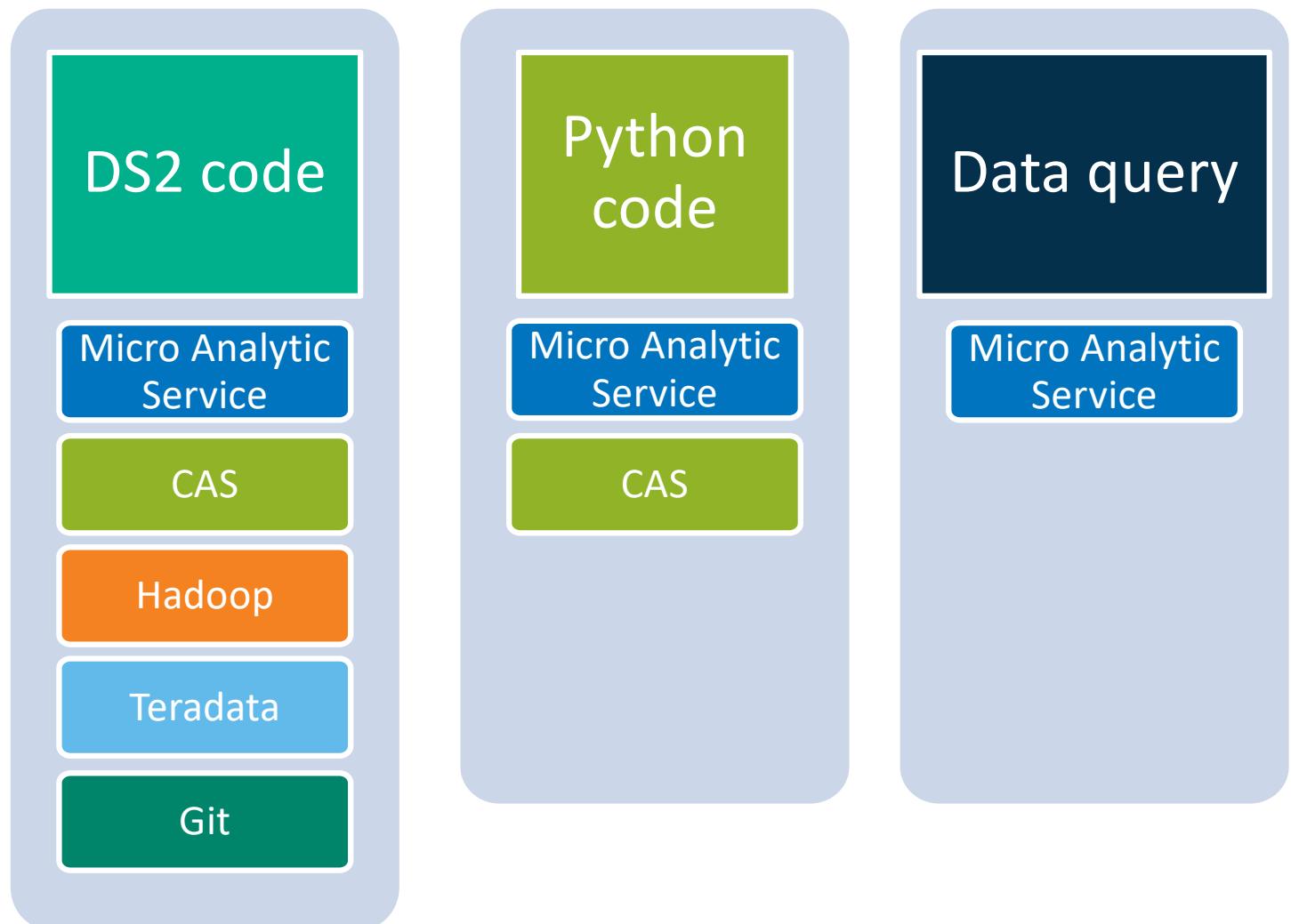
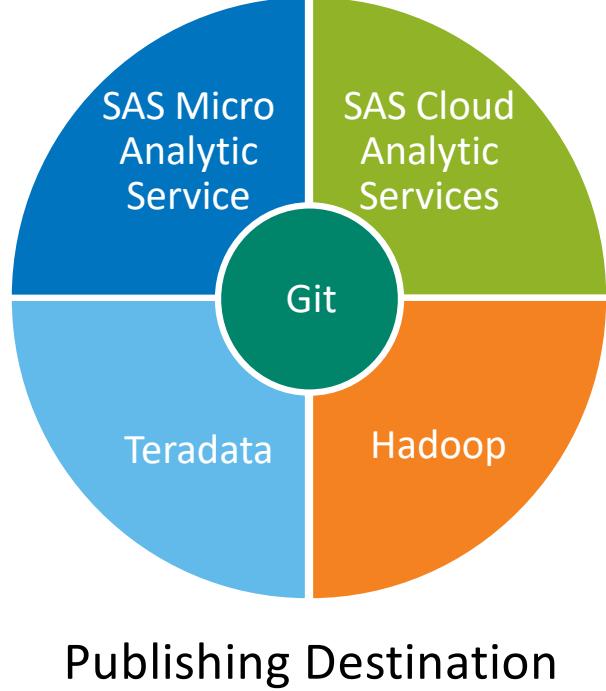


Database

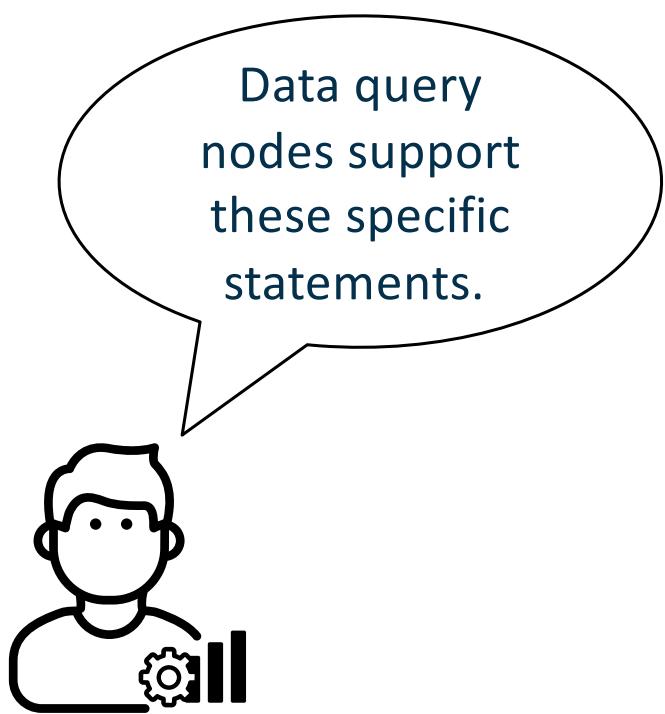
-  BigQuery
-  DB2
-  Oracle
-  Postgres
-  Snowflake
-  SQL Server
-  Teradata

Python code





Data query



Data query nodes support these specific statements.

SELECT
INSERT
UPDATE
DELETE

Use with caution.

Data query

FedSQL

ANSI SQL: 1999

Displays syntax
help



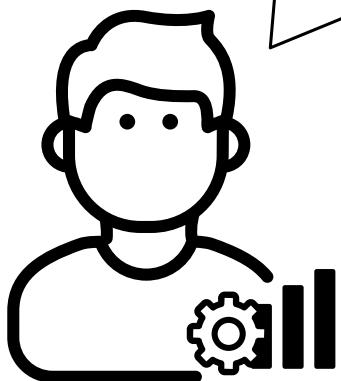
SQL editor

Option to generate
FedSQL

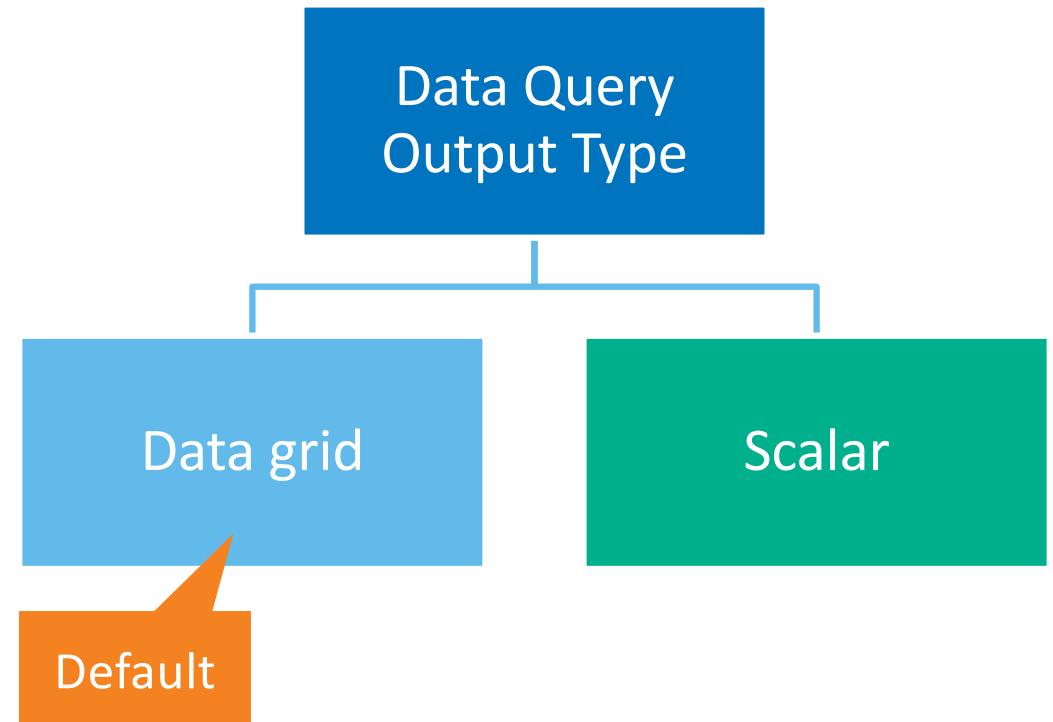


SAS Studio

Data query



A data grid is a variable with rows and columns, like a table.



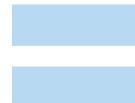
Data query

Data Query
Output Type

Data grid

Query
columns

Data
grid
columns

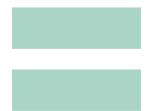


Data query

Data Query
Output Type

Scalar

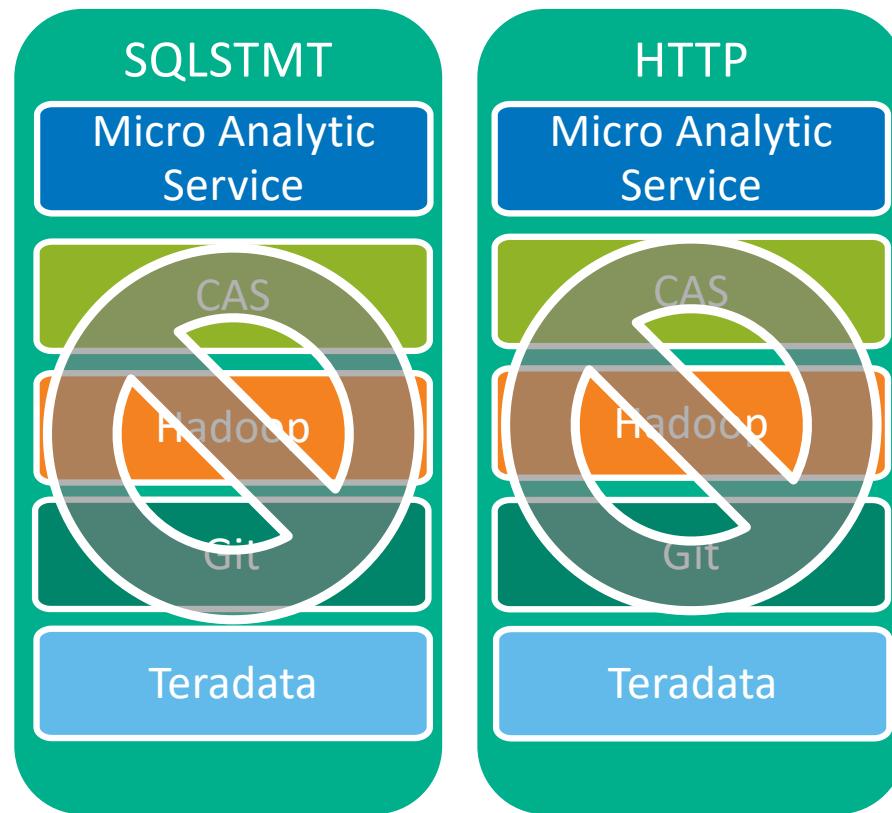
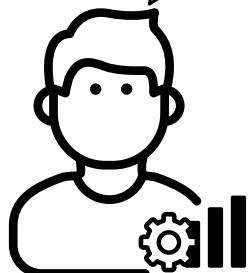
Query
columns



Decision
variables

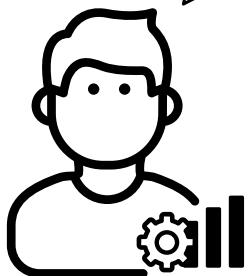
DS2 code

You should use these packages only when publishing to the Micro Analytic Service.



Python code

An administrator must configure support for testing Python code in a decision.



Configure
Python
support.

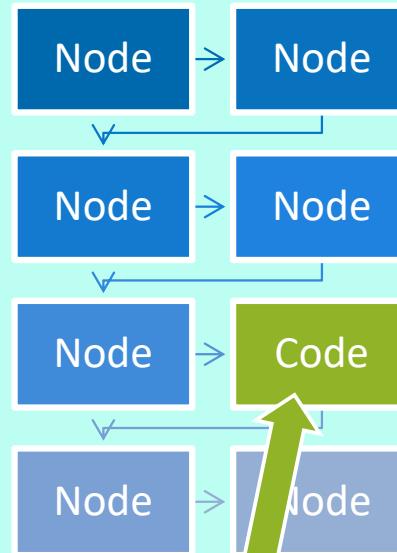
Test Python
code.



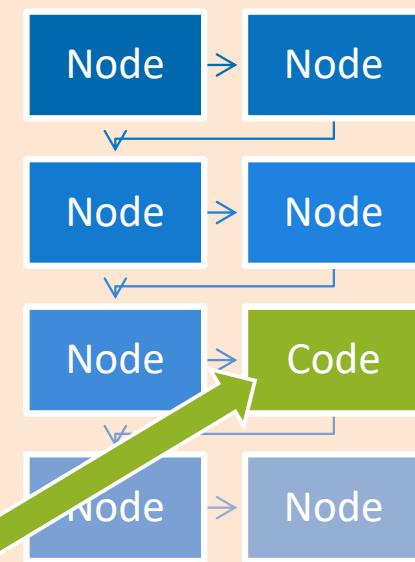
Administrator

Design Environment

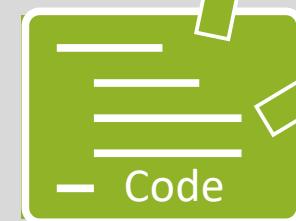
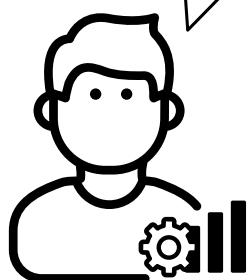
Decision 1



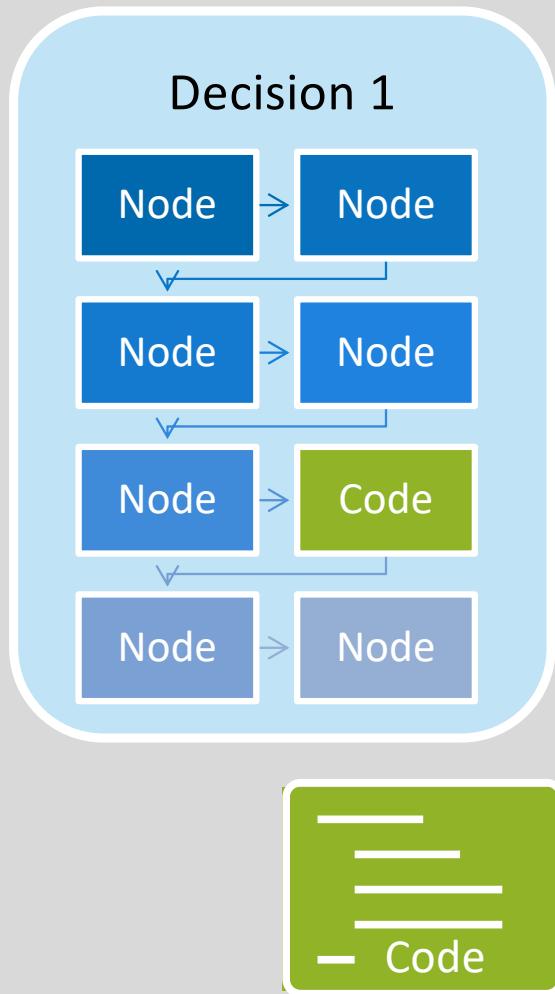
Decision 2



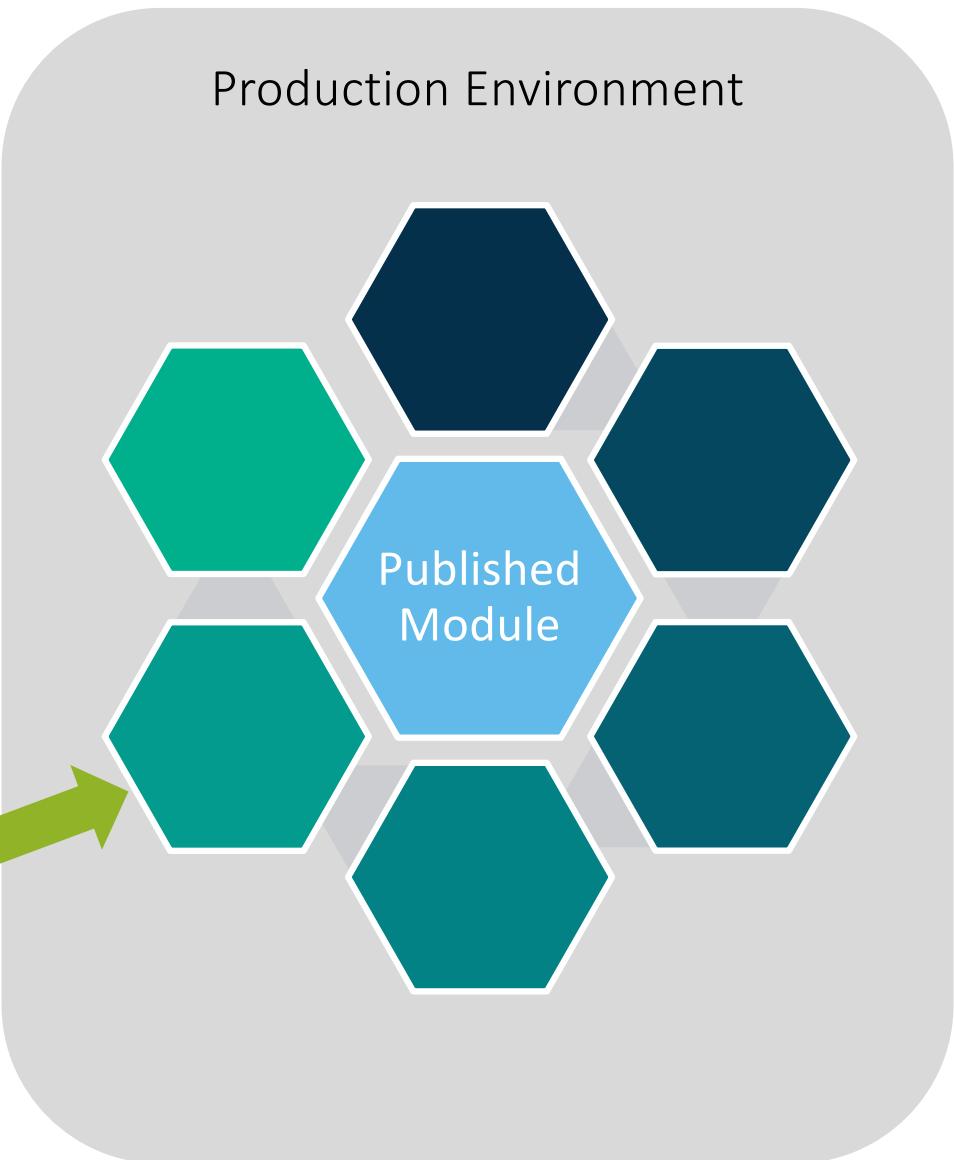
You can specify the decisions that use an upgraded version of a code file.



Design Environment



Production Environment



Published

Encapsulated



1) Creating a Data Query for Use in a Decision

This demonstration illustrates using the SQL editor to create a data query.

SAS® Intelligent Decisioning - Build Decisions

1) New code

```

Code Properties Variables Scoring Versions
Syntax Help
• SELECT COLUMN1 AS (:COL1:string:1000), COLUMN2 AS (:COL2:decimal) FROM TABLE_NAME WHERE COL1 = ?:VAL1:string:1000

1 /* include sqlReturnInfo */
2 select HomeValue as {:HomeValue:integer}, CLAge as {:CLAge:decimal}, Job as {:Job:string:8}
3 from EBDID.HMEQDATA
4 where Account={:Account:string:13}
Sync Variables

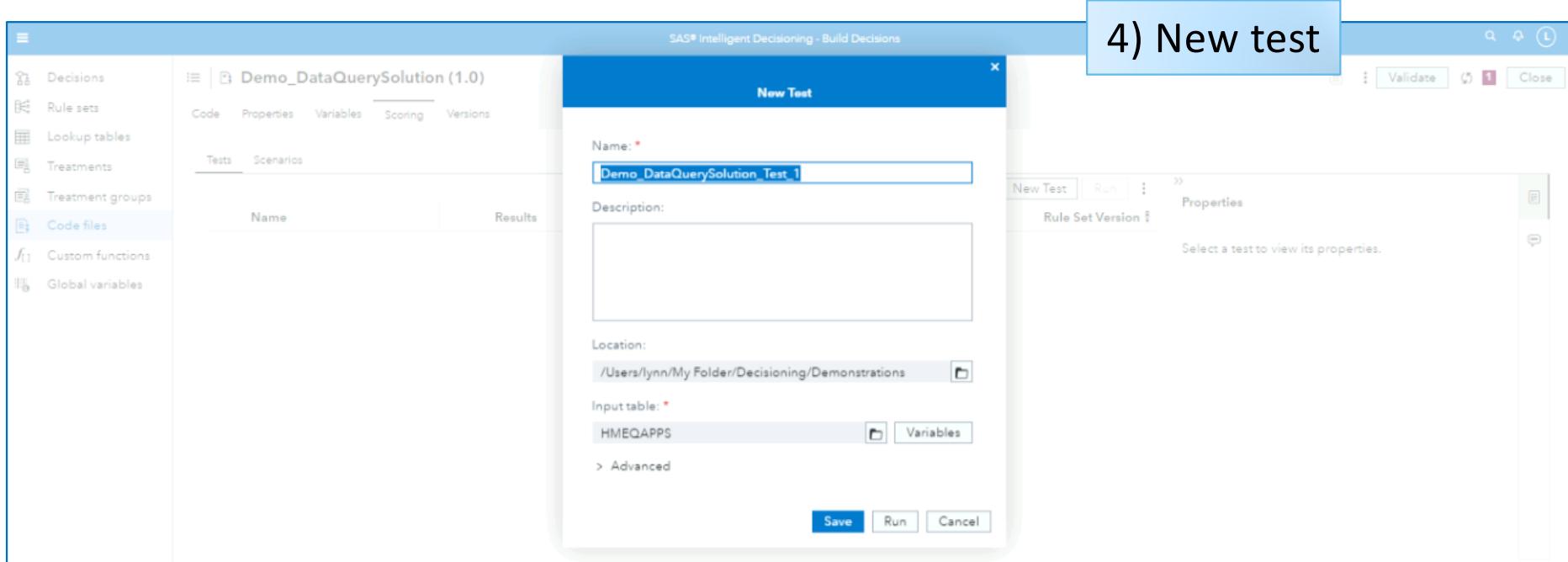
```

2) Review variable tab

Name	Data Type	Input	Output
Account	Character	<input checked="" type="checkbox"/>	<input type="checkbox"/>
CLAge	Decimal	<input type="checkbox"/>	<input checked="" type="checkbox"/>
HomeValue	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Job	Character	<input type="checkbox"/>	<input checked="" type="checkbox"/>
returnCode	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
rowCount	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>

3) Review output type in properties tab

Name:	Demo_DataQuerySolution
Type:	SQL
Location:	/Users/lynn/My Folder/Decisioning/Demonstrations
Description:	(empty)
Output Type:	<input type="radio"/> Data grid <input checked="" type="radio"/> Scalar
Date created:	Apr 28, 2021 02:51 PM
Created by:	lynn
Date modified:	Apr 28, 2021 02:51 PM



5) Review there are 3 variables in the select statement

CLAGE	HomeValue	Job	returnCode
122.9402886	85848	Other	0
219.7356734	160306	ProfExe	0
311.86989931	36191	Other	0
98.433333333	65000	Other	0
65.846117176	93839	ProfExe	0
246.19523933	51990	Other	0
180.63333333	134347	ProfExe	0
168.74982105	100937	Other	0
115.55852962	107245	ProfExe	0



2) Adding a Data Query to a Decision

This demonstration illustrates adding a data query to a decision.

SAS® Intelligent Decisioning - Build Decisions

Demo_AddQuery (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Search name: All types

Add below >

- Branch
- Data query
- Decision
- DS2 code file
- Model
- Python code file
- Record contacts
- Rule set
- Treatment group

Select a node on the diagram to view its properties.

```

graph TD
    Start([Start]) --> End([End])
  
```

*** Demo_AddQuery (1.0)**

Decision Flow Decision Properties Variables Scoring Versions History

Search name: All types

Output Variables

Output Variable	Maps To
CLAge	CLAge
HomeValue	HomeValue
Job	Job
returnCode	Demo_Data...
rowCount	Demo_Data...

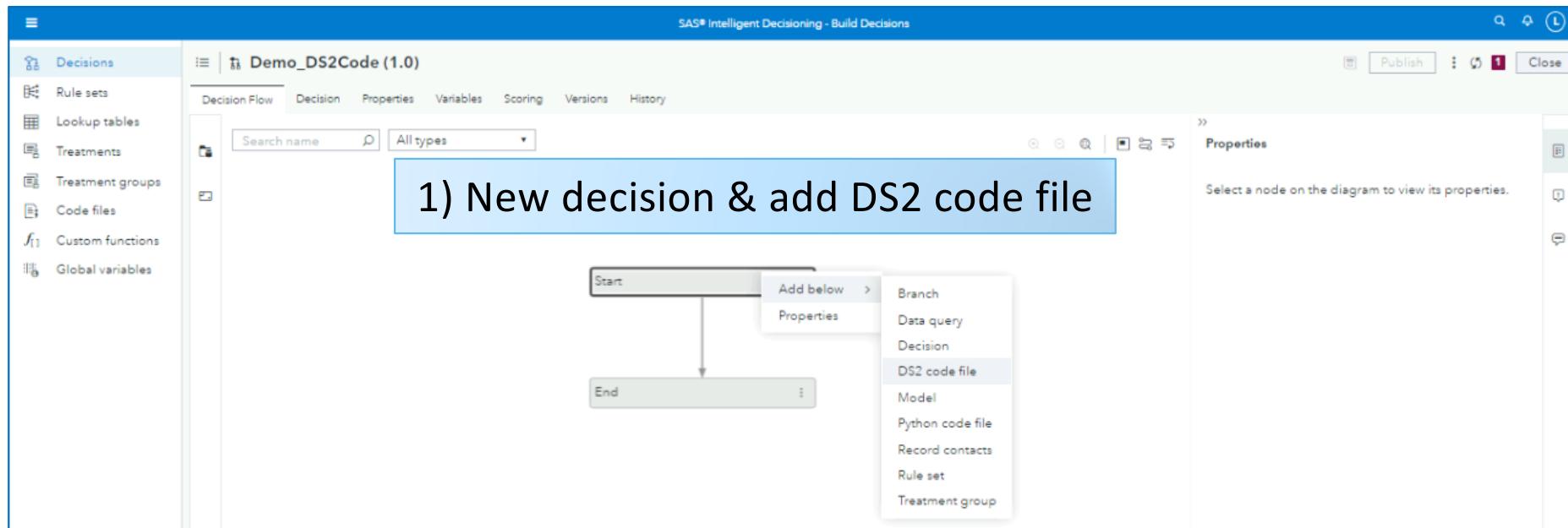
```

graph TD
    Start([Start]) --> End([End])
  
```



3) Adding DS2 Code to a Decision

This demonstration illustrates adding DS2 code to a decision.



2) Add code

Demo_DS2Code (1.0) > Demo_DS2Code (1.0)

Code Properties Variables Scoring Versions

```

1 package "${PACKAGE_NAME}" /inline;
2 method execute();
3 end;
4 endpackage;

```

Add Variables

Name:	<input type="text"/>					
Data type:	<input type="text"/> Decimal					
> Optional						
Variables	Data Type	<input checked="" type="checkbox"/> Input	<input checked="" type="checkbox"/> Output	Length	Initial Value	Description
String	Ch...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Delete"/>
Number	De...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Delete"/>

SAS® Intelligent Decisioning - Build Decisions

3) Update code & validate, then check variable tab

The screenshot shows the SAS Intelligent Decisioning interface. On the left, a sidebar lists various decisioning components: Decisions, Rule sets, Lookup tables, Treatments, Treatment groups, Code files, Custom functions, and Global variables. The main workspace displays a code editor for a package named "Demo_DS2Code". The code defines a method "execute" that takes four parameters: InterestRate, Investment, Years, and Interest, all of type double. It initializes interest to 0, loops from count=1 to years, calculates interest = Investment * InterestRate, and then ends the loop and the package.

Demo_DS2Code (1.0) > Demo_DS2Code (1.0)

Code Properties Variables Scoring Versions

```

1 package "${PACKAGE_NAME}" /inline;
2   method execute(
3     in_out double InterestRate,
4     in_out double Investment,
5     in_out double Years,
6     in_out double Interest);
7   dcl double count;
8   interest=0;
9   do count=1 to years;
10    Interest=Interest+Investment*InterestRate;
11   end;
12 end;
13 endpackage;

```

Demo_DS2Code (1.0) > * Demo_DS2Code (1.0)

Code Properties Variables Scoring Versions

Search variable ↗

Name	Data Type	Input	Output
Interest	Decimal	<input type="checkbox"/>	<input checked="" type="checkbox"/>
InterestRate	Decimal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Investment	Decimal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Years	Decimal	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

New Test

Name: * Demo_DS2Code_Test_1

Description:

Location: /Users/lynn/My Folder/Decisioning/Demonstrations

Inputtable: * INVESTMENTS

Save Run Cancel

4) Test & check results (Interest)

Demo_DS2Code (1.0) > Demo_DS2Code (1.0) > Demo_DS2Code_Test_1

Test Results

- Output
- Code
- Log

Output Table

	Interest	InterestRate	Investment
	150	0.03	1000
	210	0.04	750
	840	0.07	3000
	750	0.05	1500
	37.5	0.01	1250
	320	0.08	500
	56	0.07	200
	30	0.05	100

Lesson 3: Advanced Topics

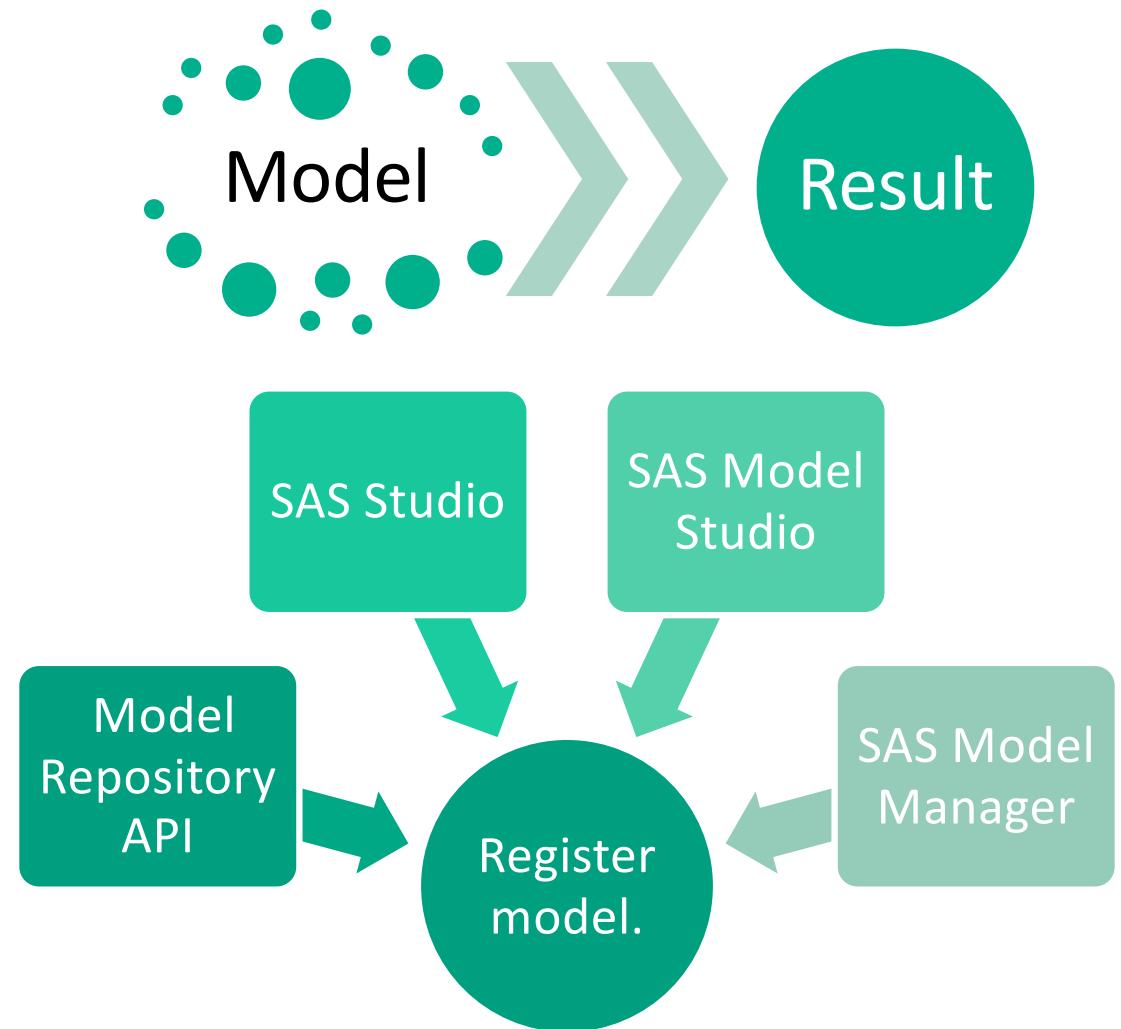
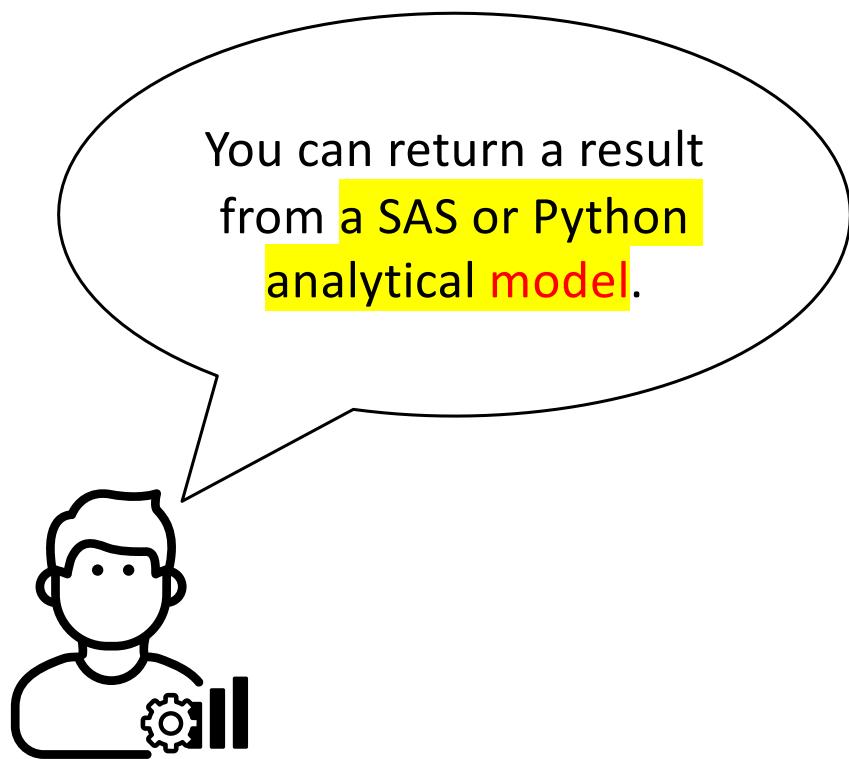
3.1 Custom Code

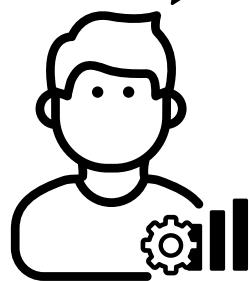
3.2 Models

3.3 Data Grids

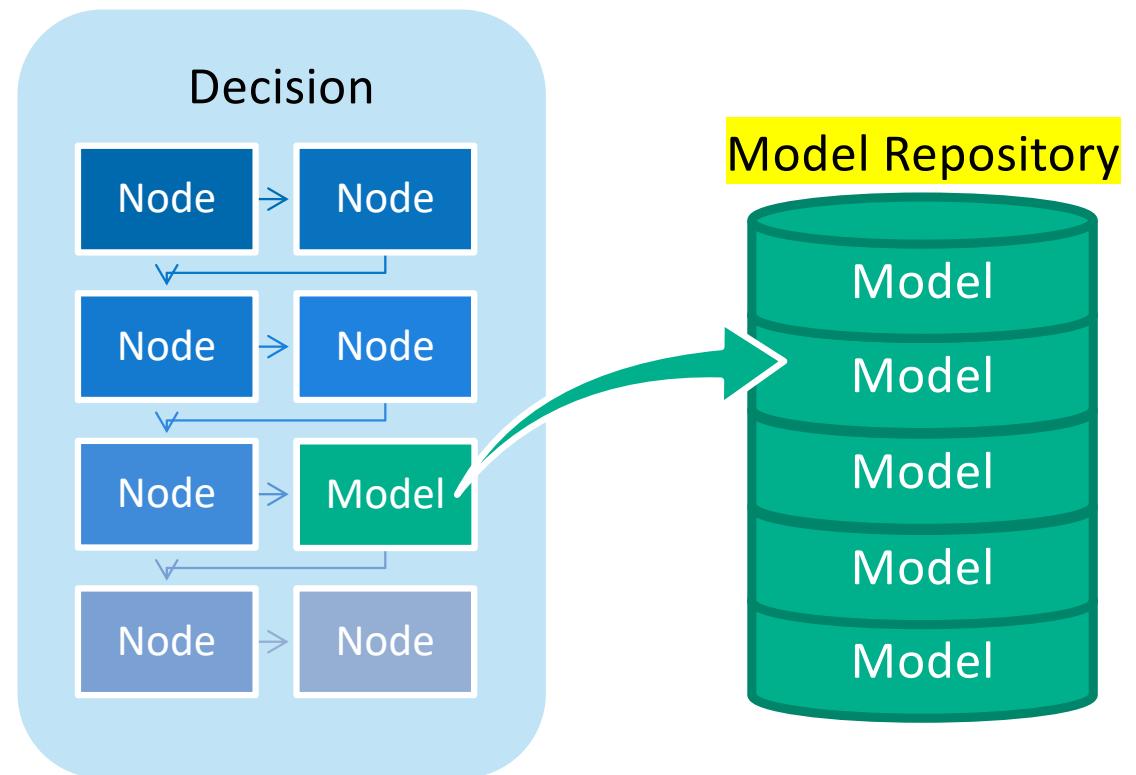


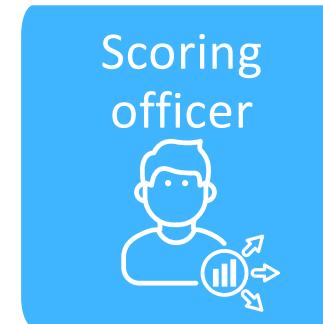
Models



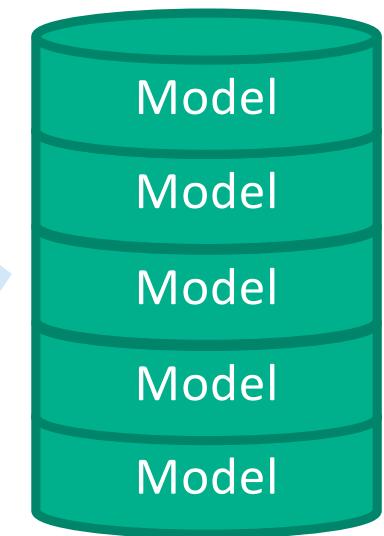


You choose from
models in a
repository.

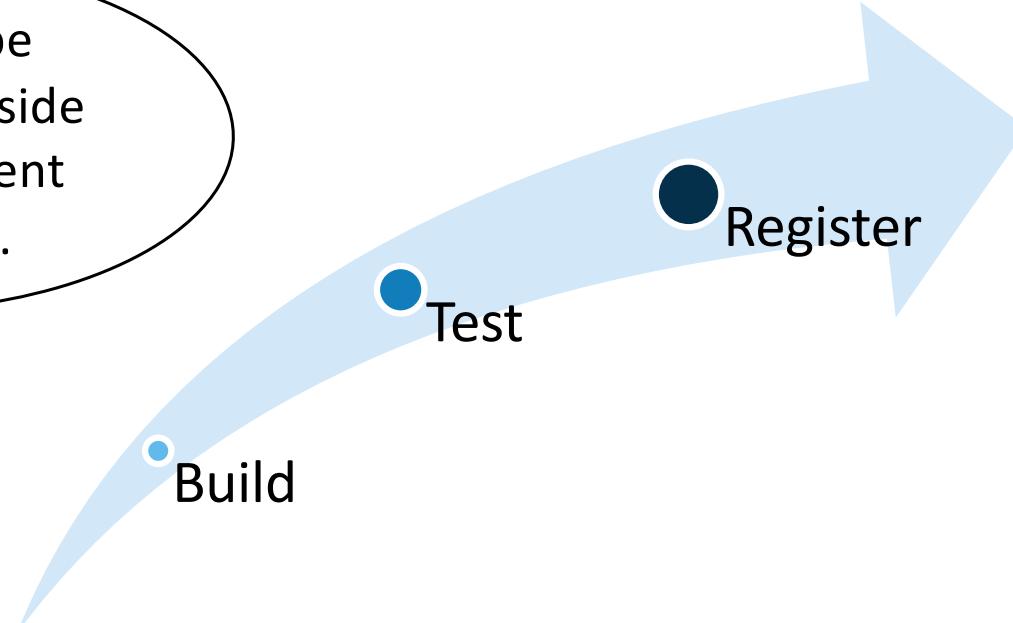
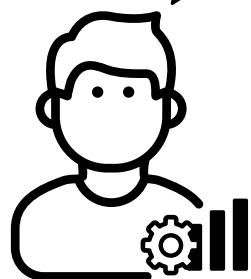




Model Repository

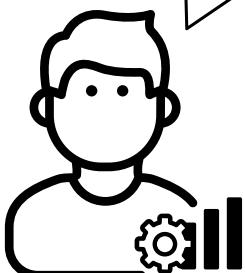


Steps must be performed outside of SAS Intelligent Decisioning.



1) Build Model

Users can build models using SAS applications.



SAS Visual Statistics

SAS Visual Data Mining and Machine Learning

Decision Tree

Linear Regression

Logistic Regression

Neural Network

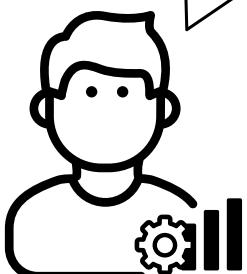
Gradient Boosting

Support Vector Machine

...and more

2) Test Model

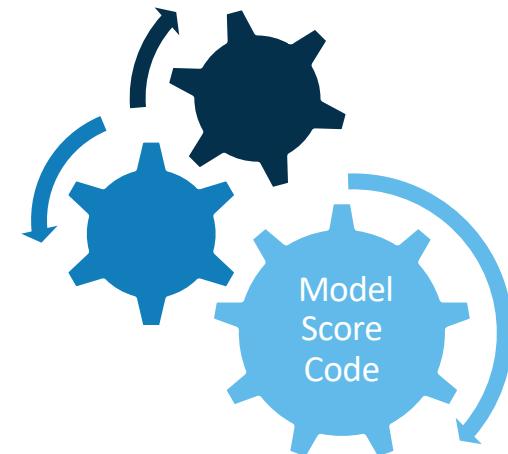
The model should be tested before registration.



SAS Visual Statistics

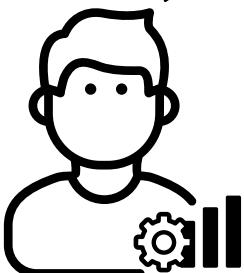
SAS Visual Data Mining and Machine Learning

SAS Model Manager



3) Register Model

Model versioning is controlled outside of SAS Intelligent Decisioning.



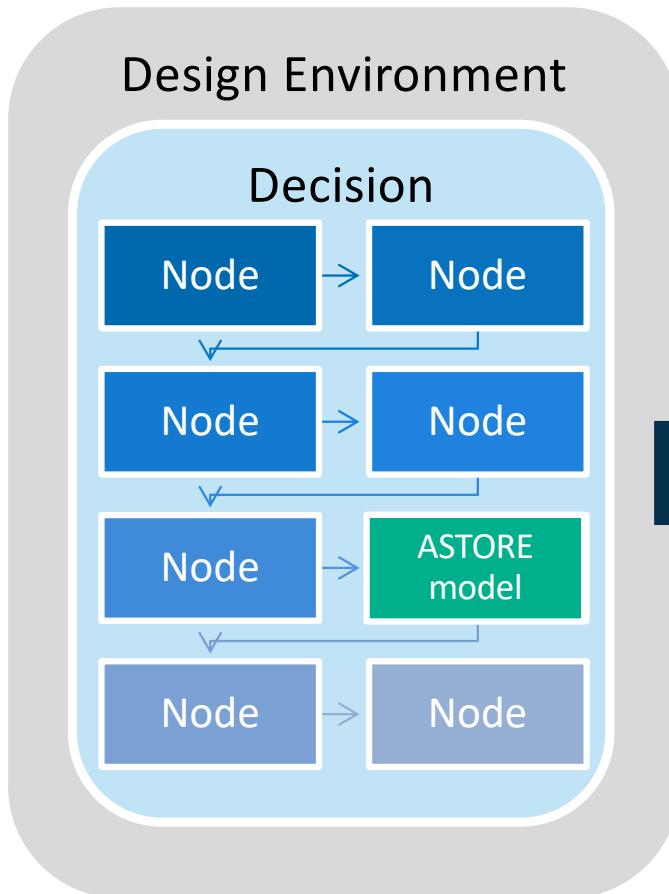
Registration and Versioning

Model
Repository
API

SAS Studio

SAS Model
Studio

SAS Model
Manager



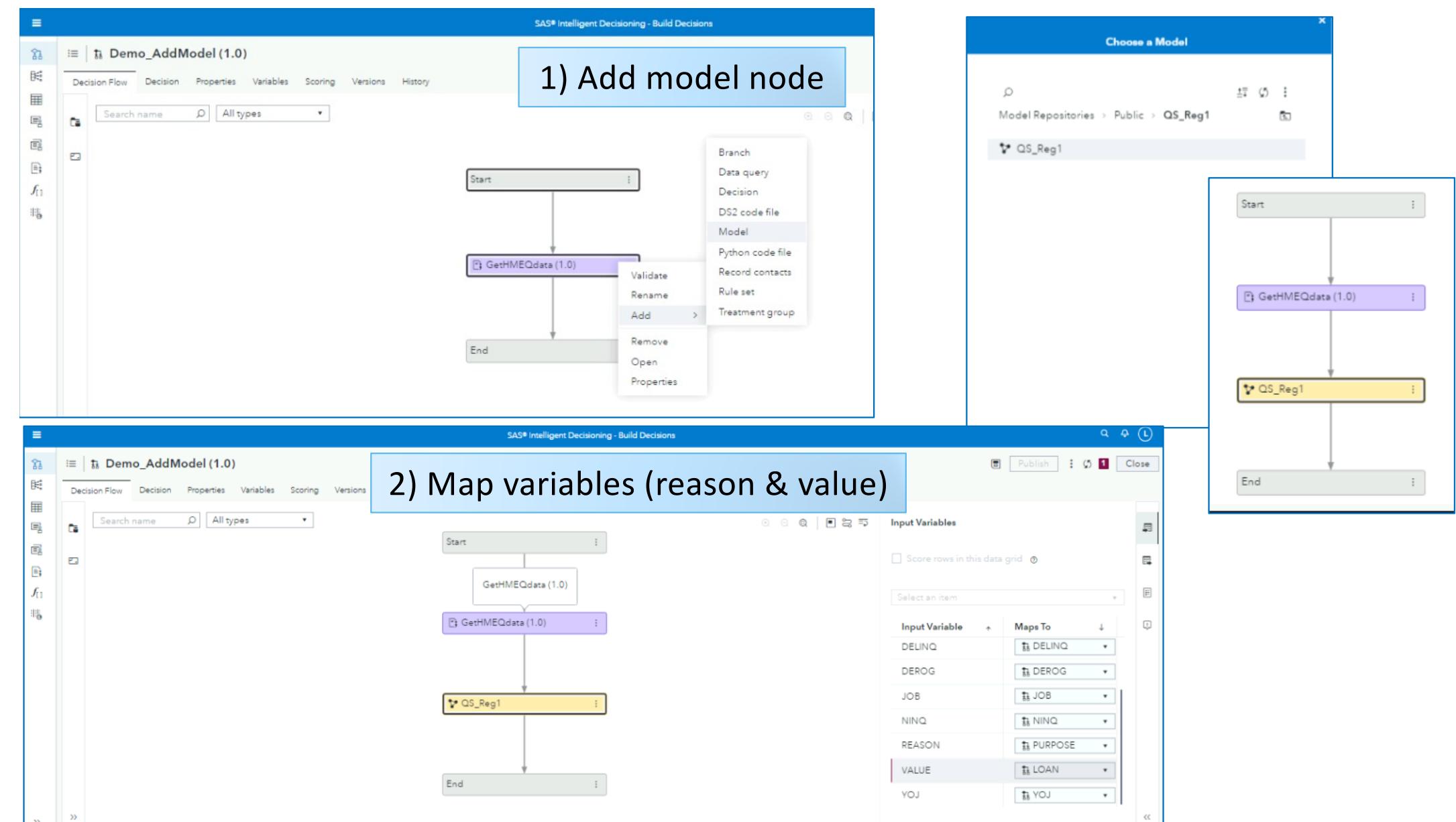
→ Publish





4) Adding a Model to a Decision

This demonstration illustrates how to add a model to a decision.



3) Test in the scoring tab

SAS® Intelligent Decisioning - Build Decisions

Demo_AddModel (1.0) > Demo_AddModel_Test_2

Close

Actions ▾

Output Table

Rules Fired Count	EM_CLASSIFICATION	EM_EVENTPROBABILITY	EM_PROBABILITY	I_BAD
0	0	0.1800913289	0.8199086711	0
0	0	0.1022369382	0.8977630618	0
0	0	0.0574396392	0.9425603608	0
0	0	0.4516623914	0.5483376086	0
0	1	0.8257108469	0.8257108469	1
0	0	0.0868929539	0.9131070461	0
0	0	0.3204310558	0.6795689442	0
0	0	0.1386429821	0.8613570179	0
0	0	0.1876071489	0.8123928511	0
0	0	0.0767462839	0.9232537161	0
0	0	0.0799637207	0.9200362793	0

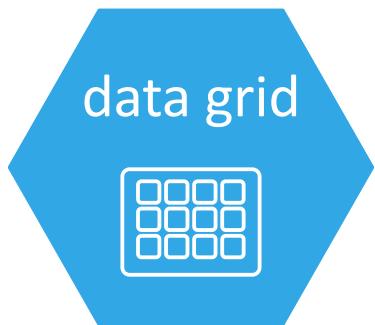
The screenshot shows the SAS® Intelligent Decisioning interface. On the left, there's a sidebar with various icons and a tree view showing 'Demo_AddModel (1.0) > Demo_AddModel_Test_2'. Under 'Test Results', 'Output' is selected. The main area is titled 'Output Table' and displays a table with five columns: 'Rules Fired Count', 'EM_CLASSIFICATION', 'EM_EVENTPROBABILITY', 'EM_PROBABILITY', and 'I_BAD'. The table contains 10 rows of data. The 'EM_CLASSIFICATION' column has values 0, 0, 0, 0, 1, 0, 0, 0, 0, 0. The 'EM_EVENTPROBABILITY' column has values 0.1800913289, 0.1022369382, 0.0574396392, 0.4516623914, 0.8257108469, 0.0868929539, 0.3204310558, 0.1386429821, 0.1876071489, 0.0767462839. The 'EM_PROBABILITY' column has values 0.8199086711, 0.8977630618, 0.9425603608, 0.5483376086, 0.8257108469, 0.9131070461, 0.6795689442, 0.8613570179, 0.8123928511, 0.9232537161. The 'I_BAD' column has values 0, 0, 0, 0, 1, 0, 0, 0, 0, 0.

Lesson 3: Advanced Topics

3.1 Custom Code

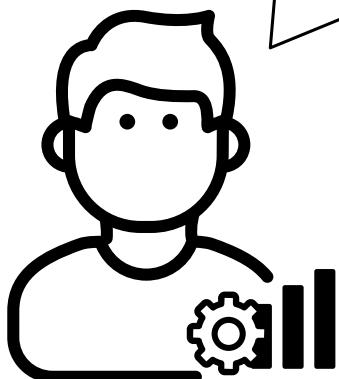
3.2 Models

3.3 Data Grids

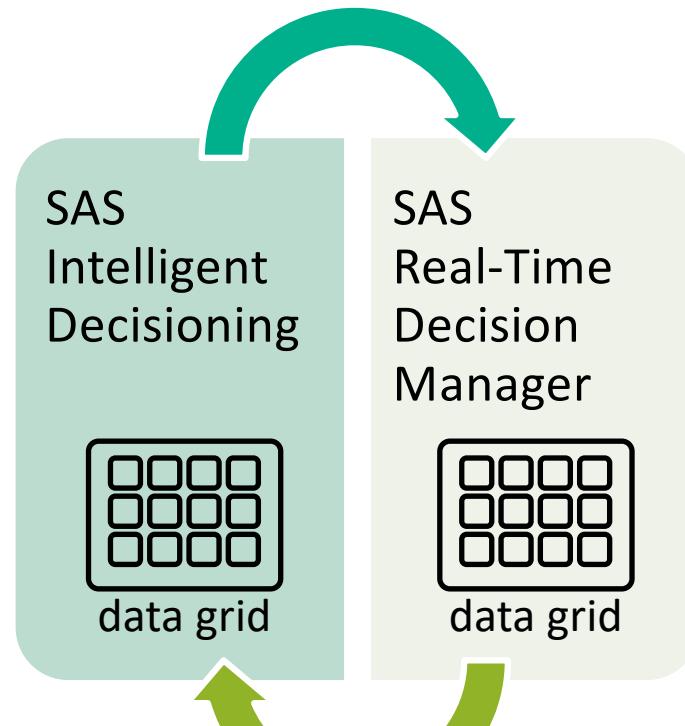


data grid

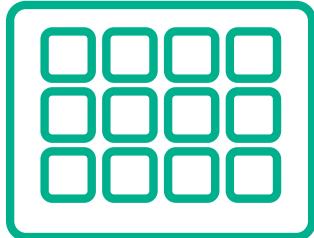
A data grid is a variable with rows and columns, like a table.



Conceptually similar



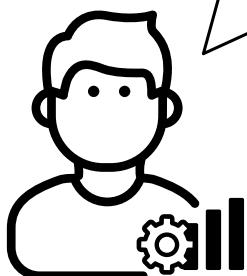
Implemented differently

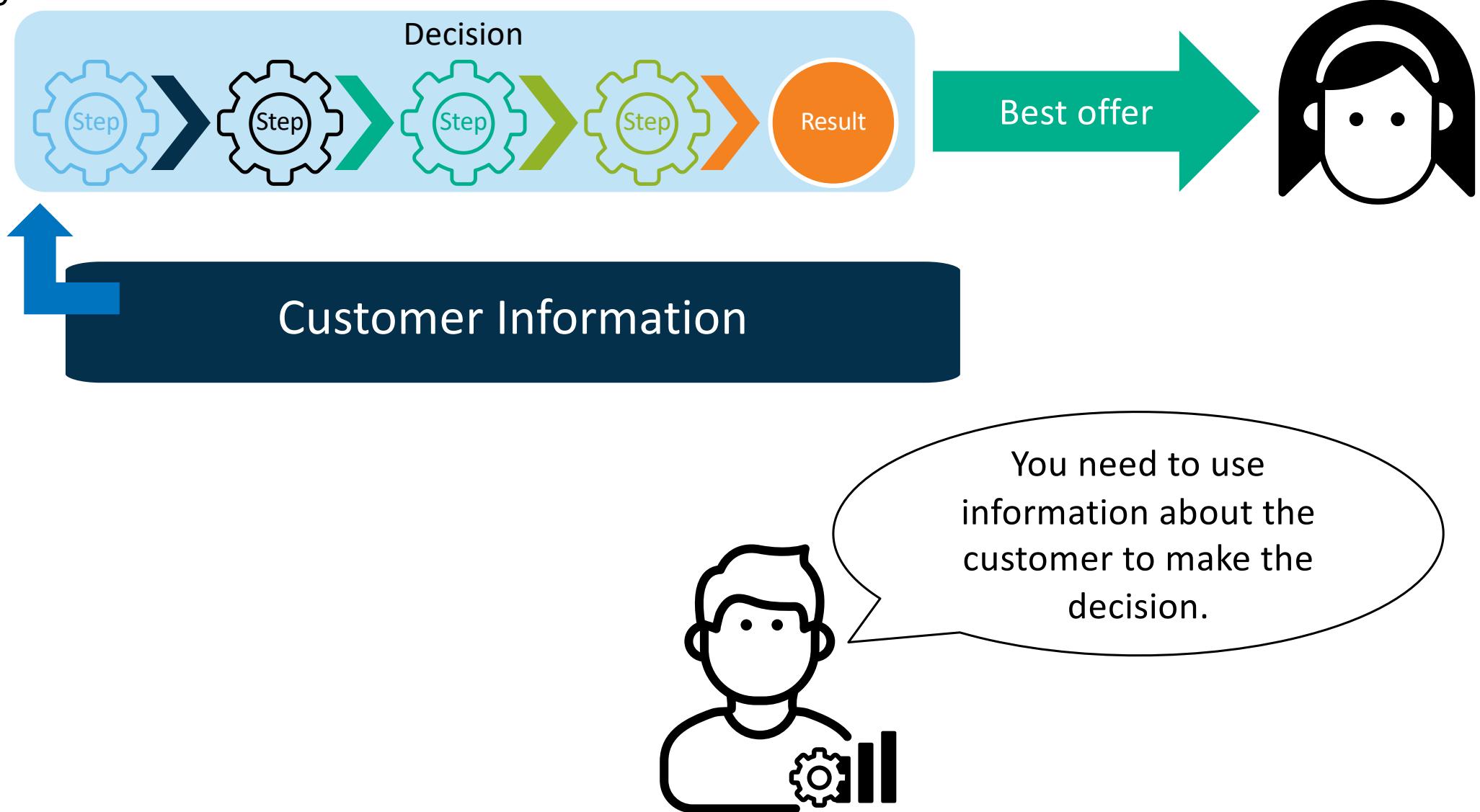


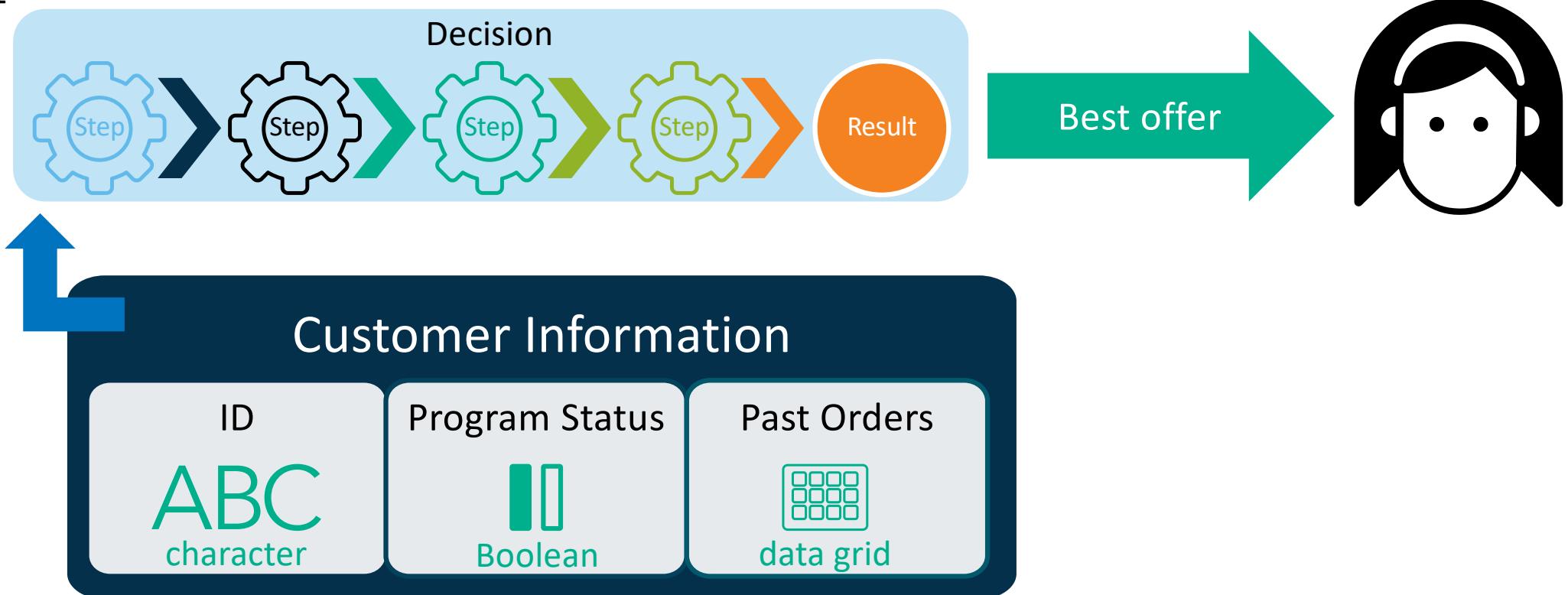
JSON string

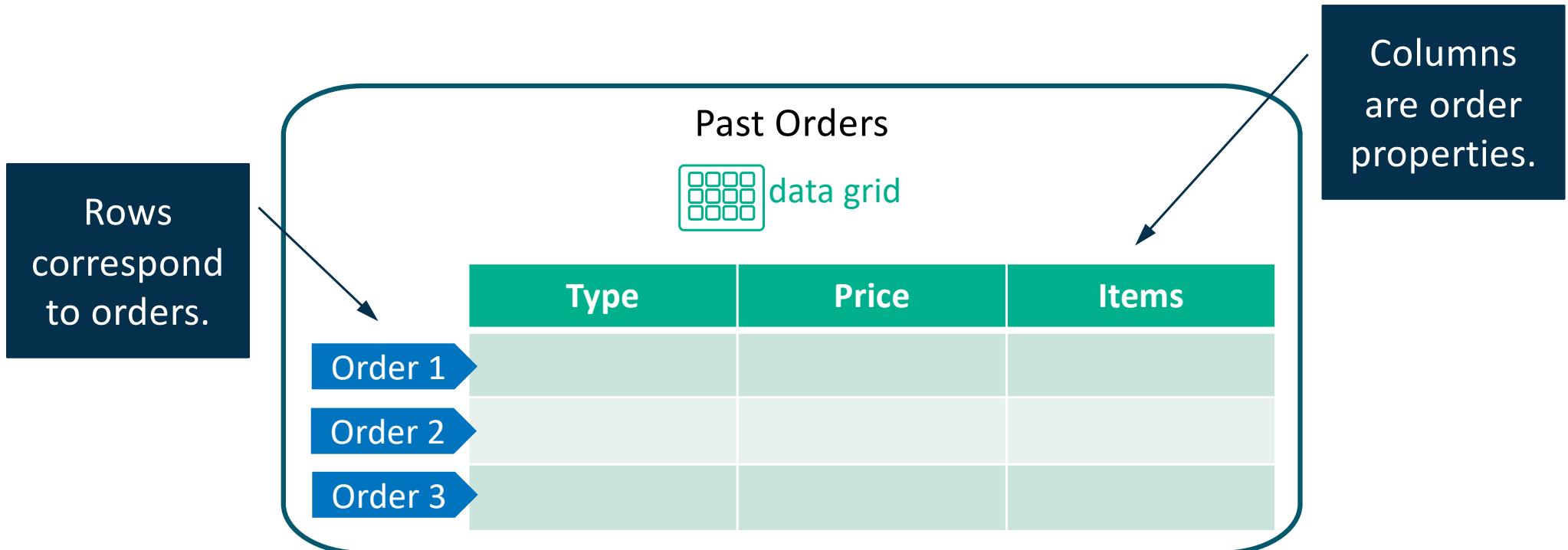
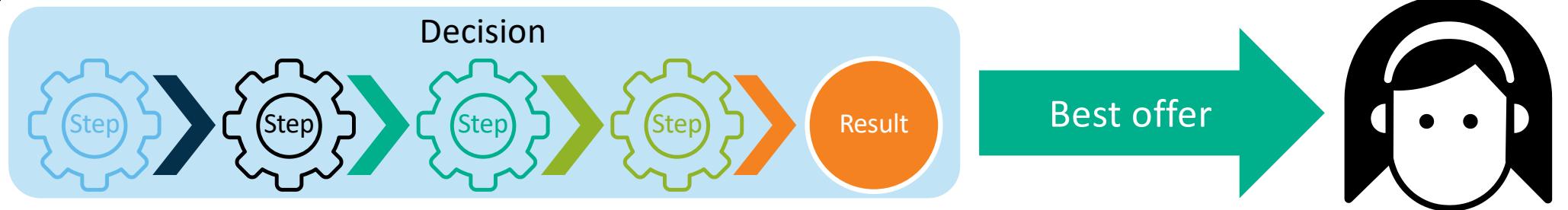
Data Grid Variable

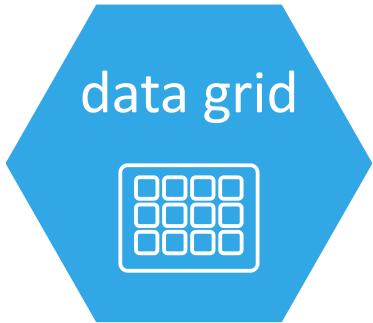
A JSON string describes
the columns and values
in the data grid.



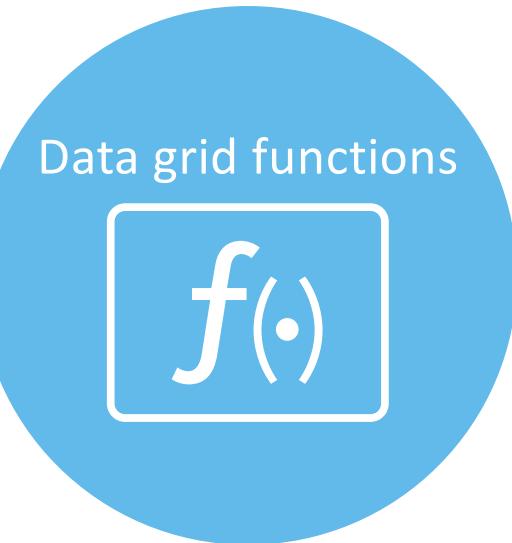
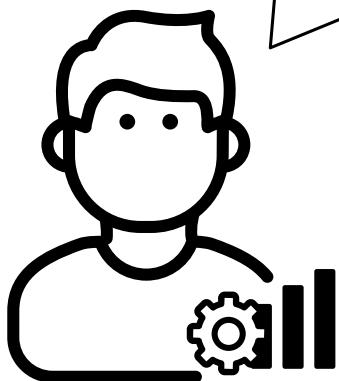




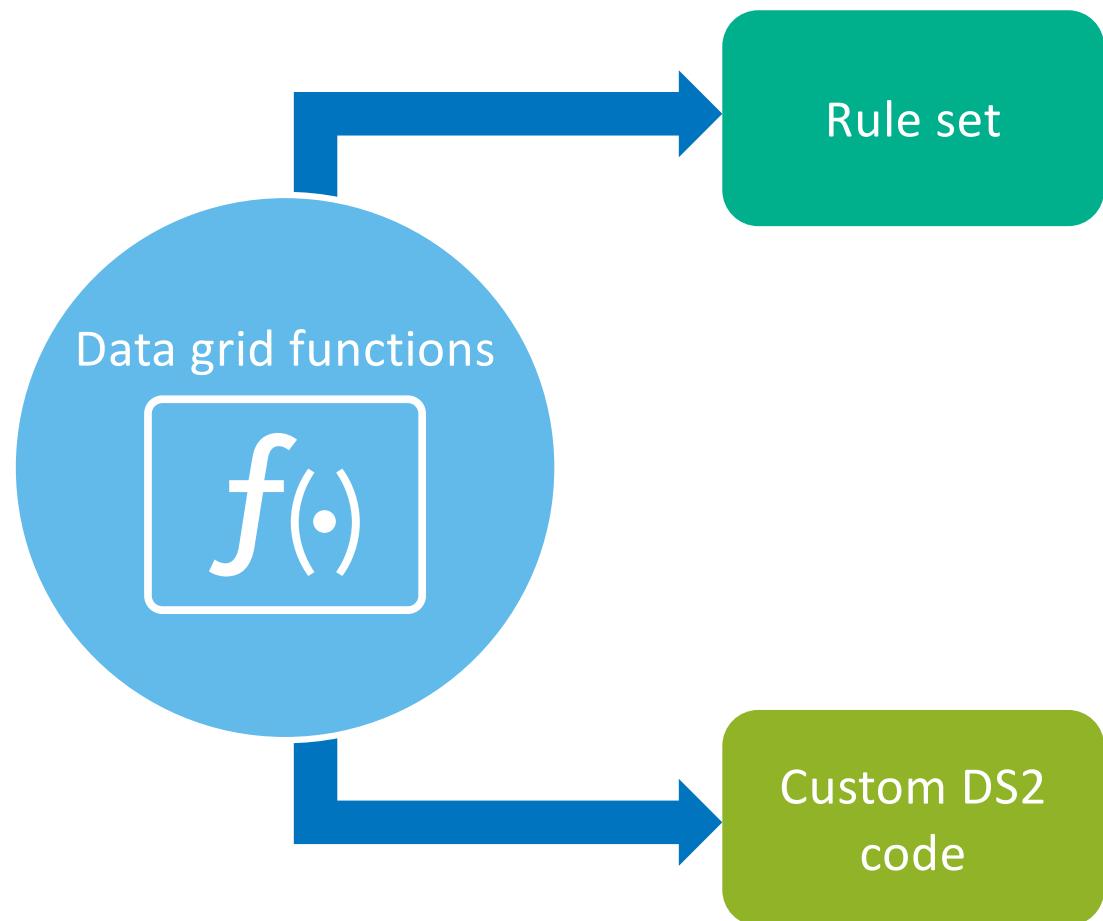


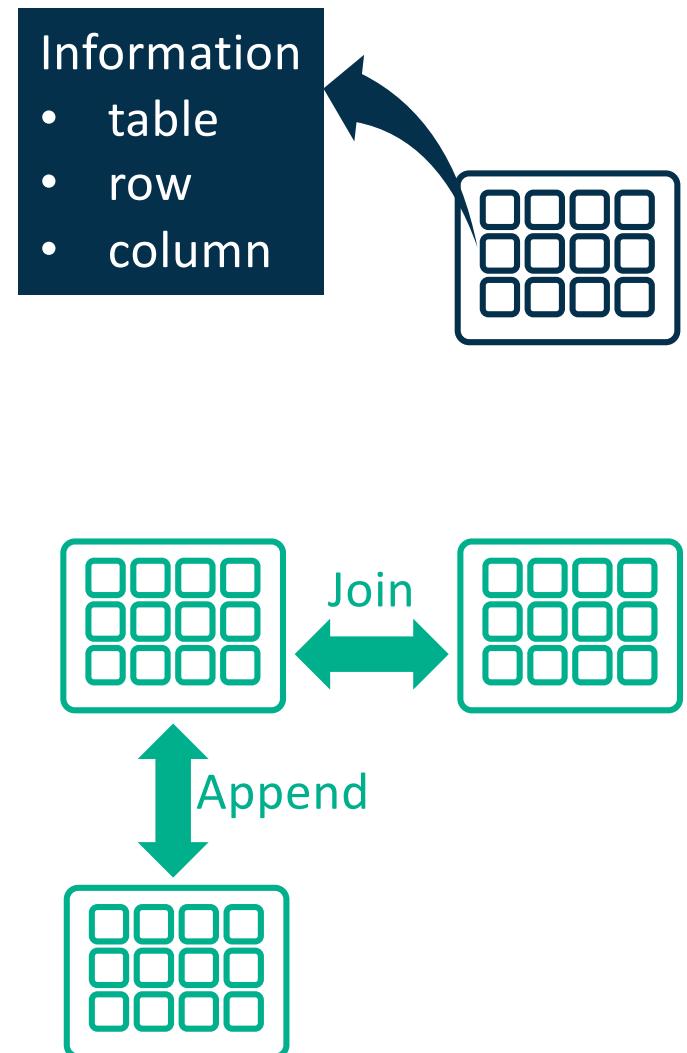
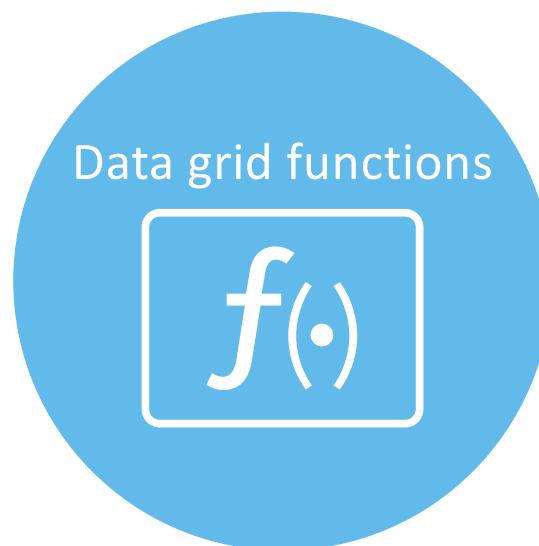
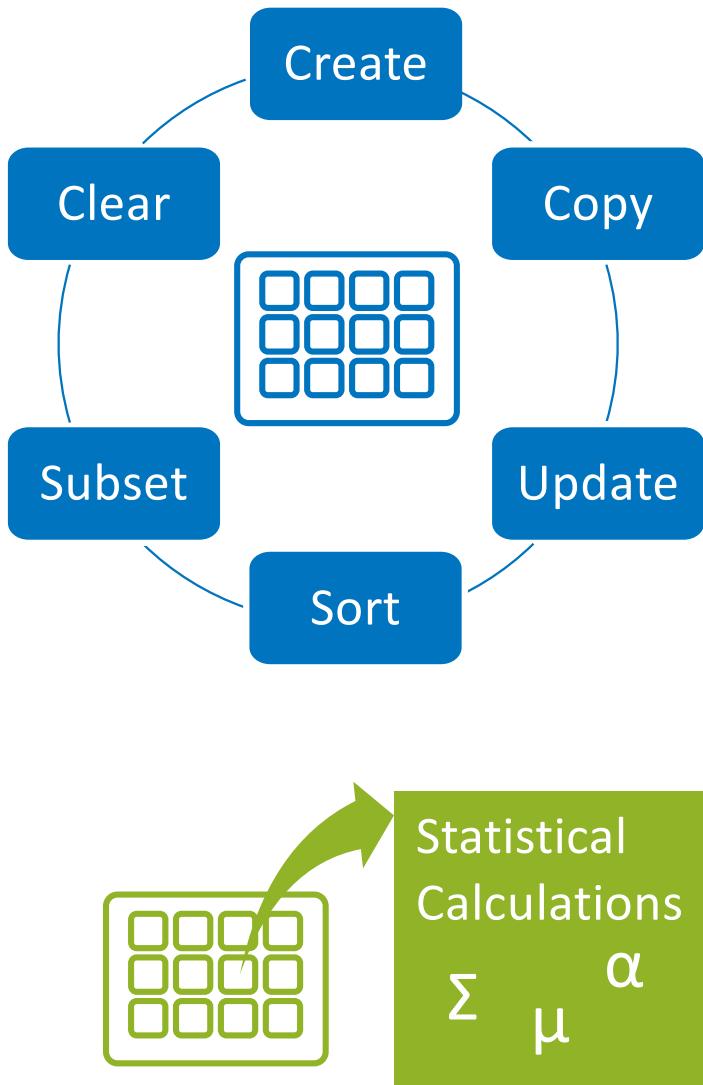


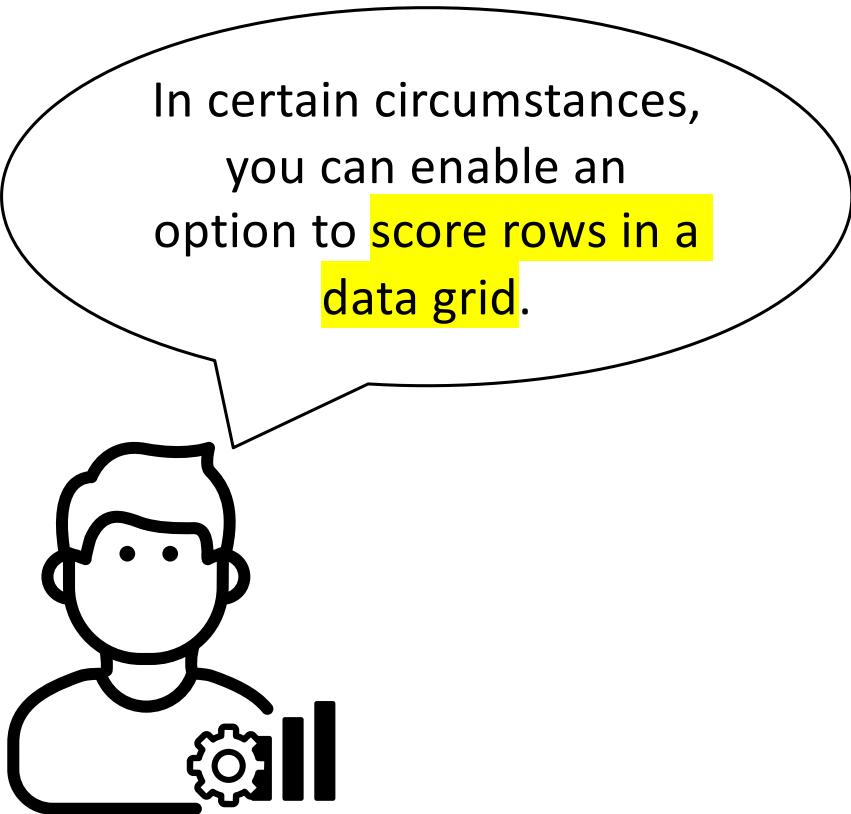
There are two ways
that you can process
data grids.



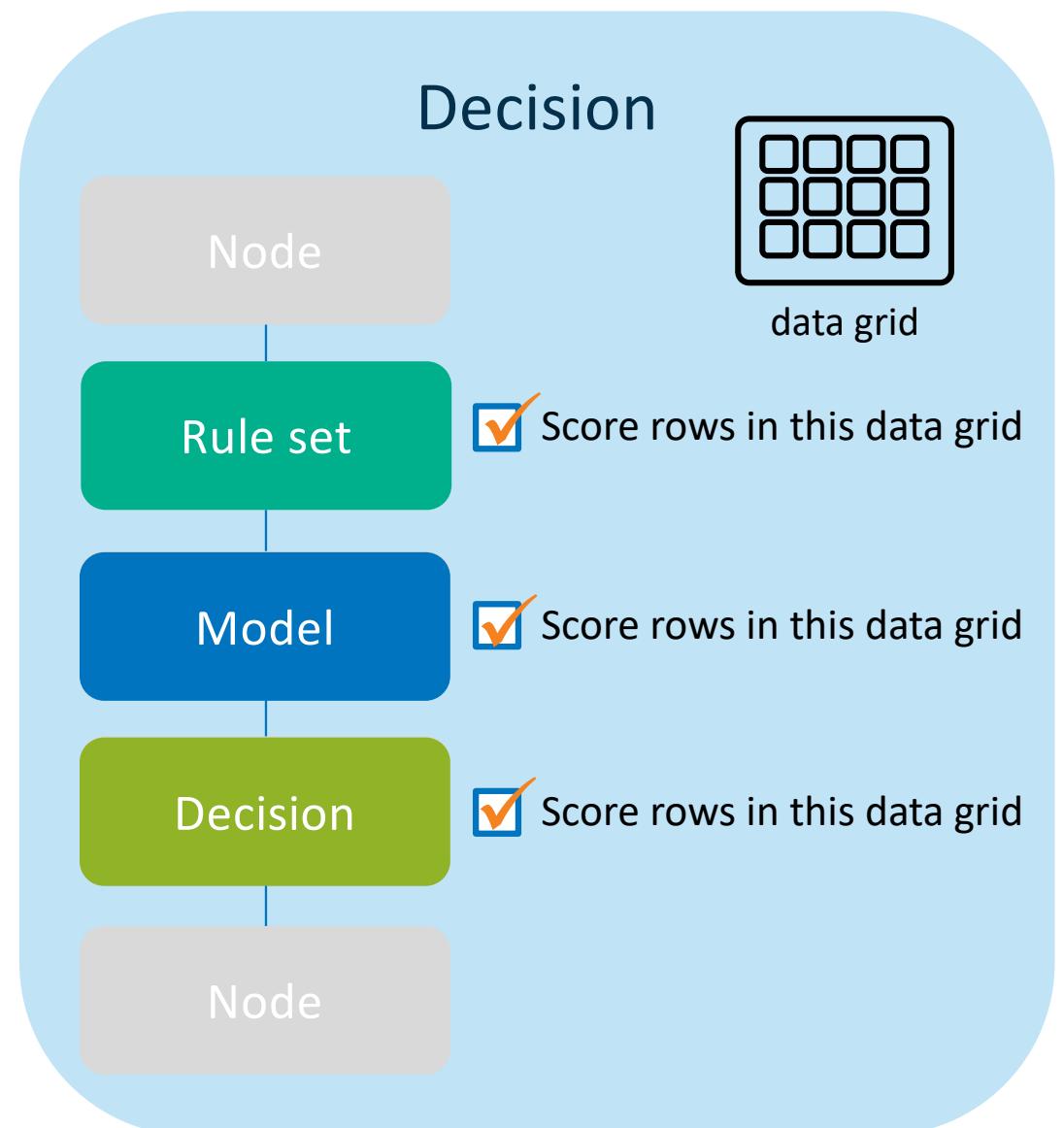
Score rows in this data grid





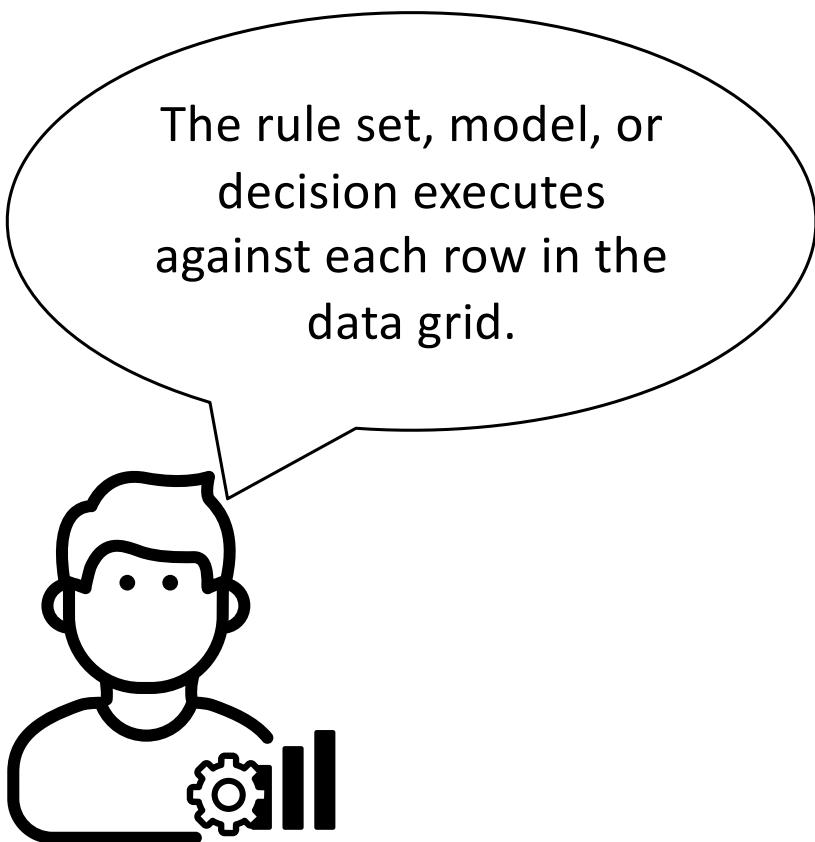


In certain circumstances,
you can enable an
option to score rows in a
data grid.



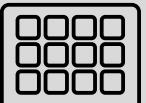


Score rows in this data grid



	Col1	Col2	Col3
Execute			

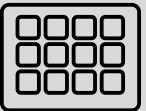
Examples of Data Grid Processing

 Cell Phone Packages

data grid

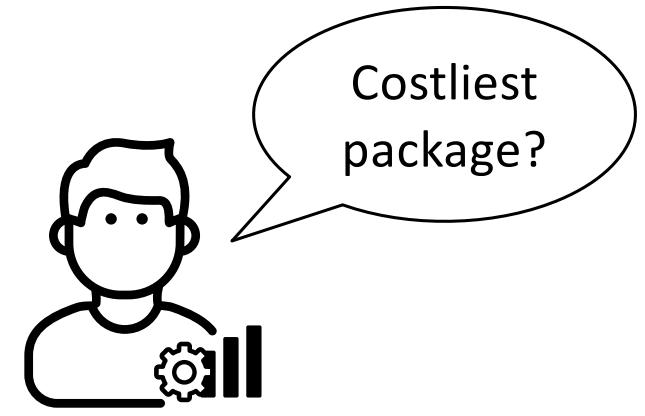
Offer Code	Phone ID	Cost
AB1	A	120
JK2	B	100
XZ3	C	150

Examples of Data Grid Processing

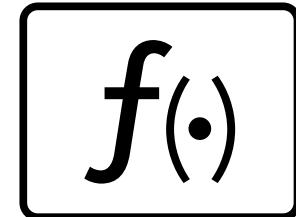
 Cell Phone Packages

data grid

Offer Code	Phone ID	Cost
AB1	A	120
JK2	B	100
XZ3	C	150

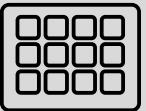


Rule set



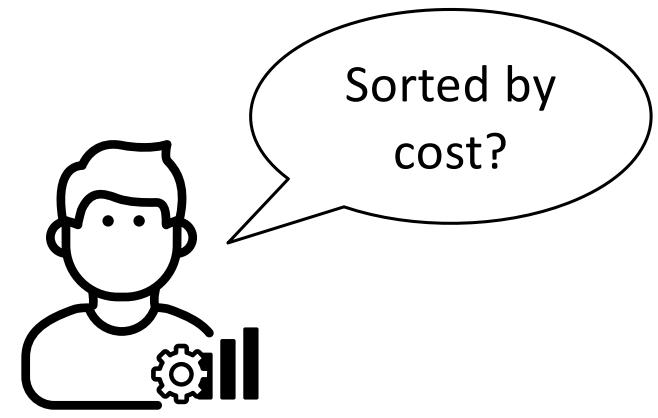
DATAGRID_MAX

Examples of Data Grid Processing

 Cell Phone Packages

data grid

Offer Code	Phone ID	Cost
JK2	B	100
AB1	A	120
XZ3	C	150

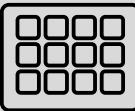


Rule set

$f(\cdot)$

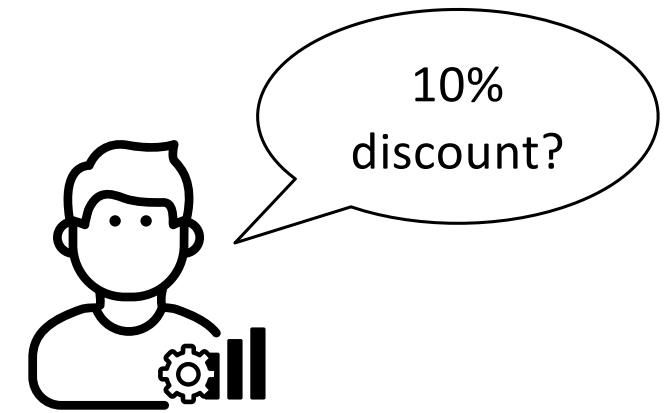
DATAGRID_SORT

Examples of Data Grid Processing

 data grid

Cell Phone Packages

Offer Code	Phone ID	Cost	DiscountCost
AB1	A	120	108
JK2	B	100	90
XZ3	C	150	135

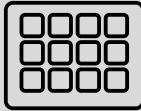


Rule set

Score rows in this data grid

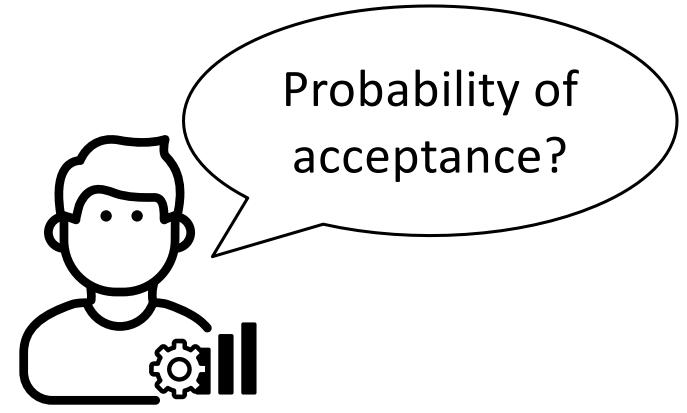
$\text{DiscountCost} = \text{Cost} * 0.9$

Examples of Data Grid Processing

 Cell Phone Packages

data grid

Offer Code	Phone ID	Cost
AB1	A	120
JK2	B	100
XZ3	C	150

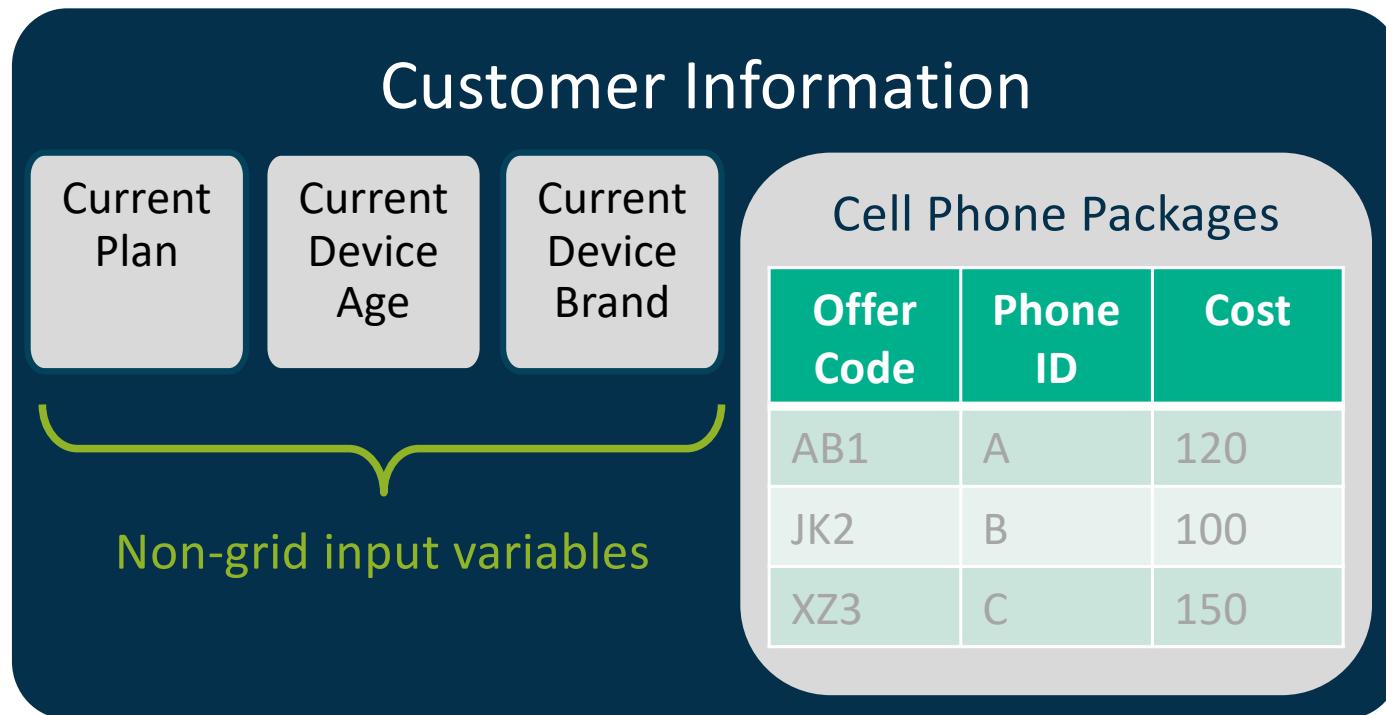
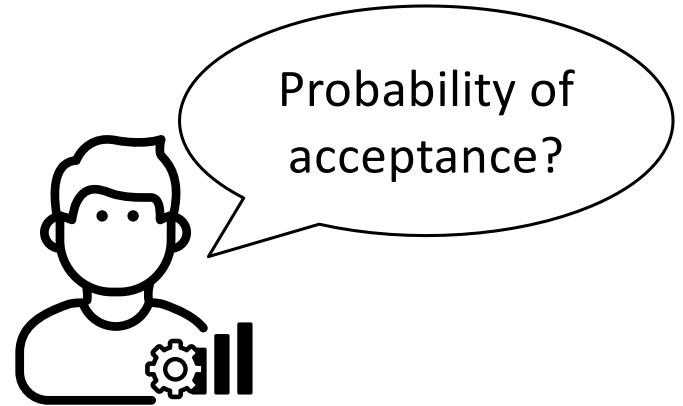


Model

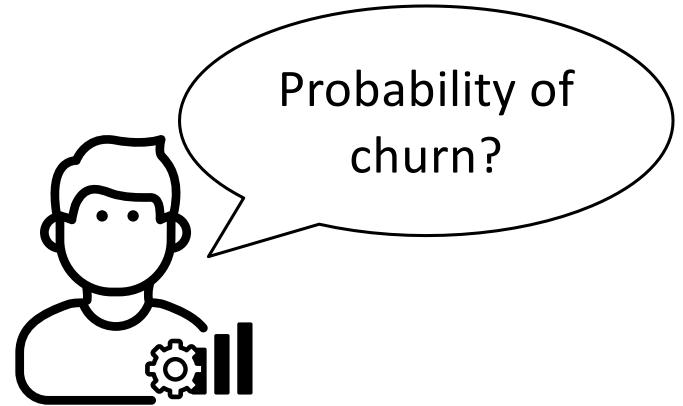
Score rows in this data grid

Predicted propensity

Examples of Data Grid Processing



Examples of Data Grid Processing



Customer Information

Current Plan Current Device Age Current Device Brand

Customer-level variables

Cell Phone Packages

Offer Code	ID	Cost
AB1	A	20
JK2	B	200
XZ3	C	150

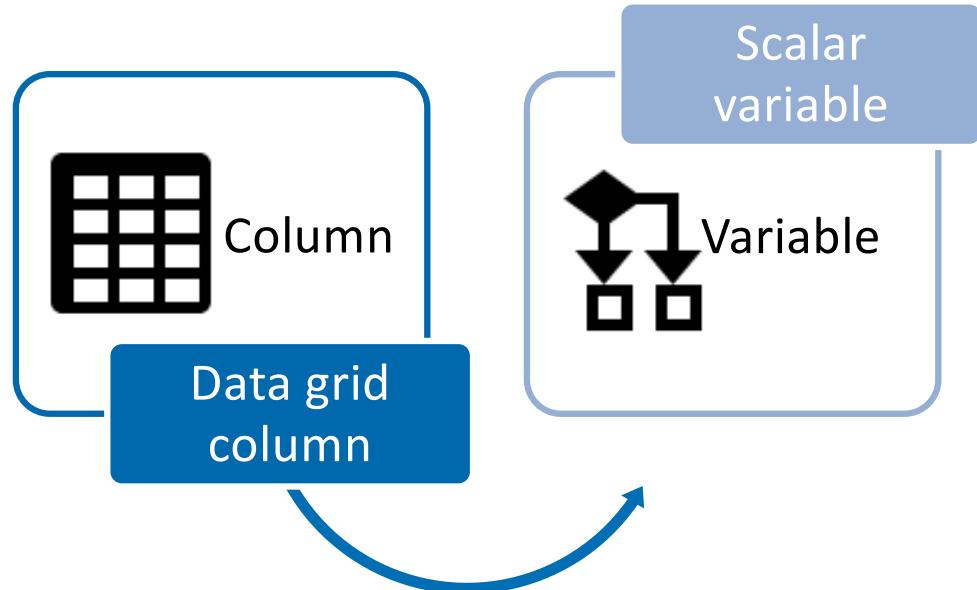
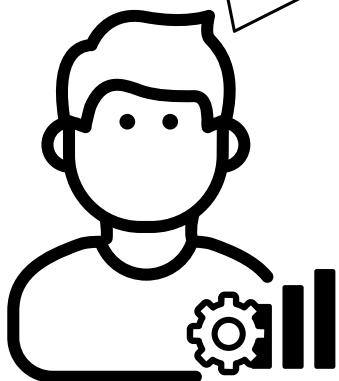
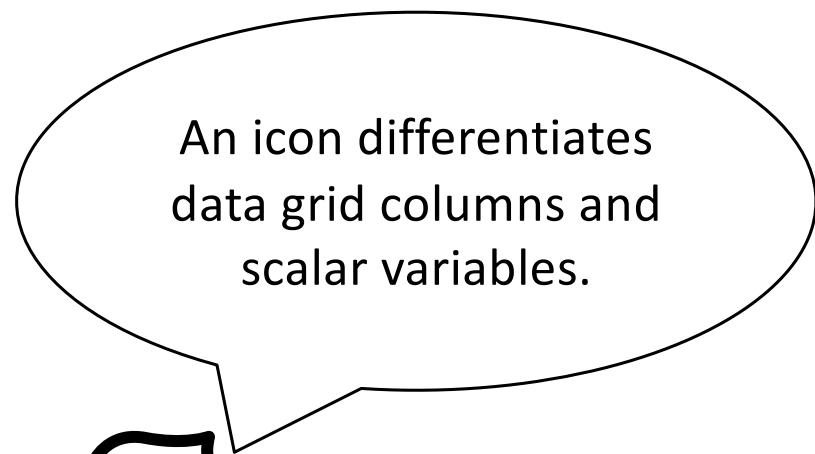
A large orange circle with a diagonal slash through it is overlaid on the "Cell Phone Packages" grid, indicating that rows from this grid cannot be processed.

Model

Score rows in this data grid

Predicted churn

Score rows in this data grid





5) Configuring a Data Query to Return a Data Grid

This demonstration shows how to configure a data query to return a data grid.

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

Demo_QueryReturnDataGrid (1.0)

Code Properties Variables Scoring Versions

Syntax Help

- SELECT COLUMN1 AS [:COL1:string:1000], COLUMN2 AS [:COL2:decimal] FROM TABLE_NAME WHERE COL1 = ?:VAL1:string:1000

```
/* include sqlReturnInfo */  
select HomeValue as {:Homevalue:integer}, CLAge as {:CLAge:decimal}, Job as {:Job:string:8}  
from EBDID.HMEQDATA  
where Account=?:Account:string:13
```

Sync Variables

1) Check data query in Code files

This screenshot shows the SAS Intelligent Decisioning interface. On the left, there's a sidebar with various project components like Decisions, Rule sets, and Treatments. The 'Code files' option is selected. In the main area, a decision named 'Demo_QueryReturnDataGrid' is open. The 'Code' tab is selected, displaying an SQL query. The query selects 'COLUMN1' and 'COLUMN2' from 'TABLE_NAME' where 'COL1' equals a parameter '?VAL1'. Below the code, there's a snippet of code starting with '/* include sqlReturnInfo */'.

2) Set output type to be “data grid”

* Demo_QueryReturnDataGrid (1.0)

Code Properties Variables Scoring Versions

Name: Demo_QueryReturnDataGrid

Type: SQL

Location: /Users/lynn/My Folder/Decisioning/Demonstrations

Description:

Output Type: Data grid Scalar

Date created: Apr 28, 2021 02:51 PM

3) Preview variable tab

Demo_QueryReturnDataGrid (1.0)

Code Properties Variables Scoring Versions

Search variable

Name	Data Type	Input	Output
Account	Character	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Demo_QueryReturnDataGrid_d go	Data grid	<input type="checkbox"/>	<input checked="" type="checkbox"/>
returnCode	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
rowCount	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>

4) Test in scoring tab

SAS® Intelligent Decisioning - Build Decisions

Decisions Rule sets Lookup tables Treatments Treatment groups Code files Custom functions Global variables

Demo_QueryReturnDataGrid (1.0)

Code Properties Variables Scoring Versions

Tests Scenarios

New Test Run Properties

Name: Demo_QueryReturnDataGrid_Test_1
Status: Completed successfully

Demo_QueryReturnDataGrid (1.0) > Demo_QueryReturnDataGrid_Test_1

Output Table

Test Results Output Code Log Actions

Demo_QueryReturnDataGrid_dgo

	returnCode	rowCount
["metadata":{("HOMEVALUE":"decimal")("CLAGE"...	0	1

Demo_QueryReturnDataGrid_dgo

Data Grid Formatted Plain

HOMEVALUE	CLAGE	JOB	#
85848	122.9402886	Other	

```
1 /* include sqlReturnInfo */
2 select HomeValue as {:Homevalue:integer}, CLAge as {:CLAge:decimal}, Job as {:Job:string:8}
3 from EBDID.HMEQDATA
4 where Account={?:Account:string:13}
```



6) Using Data Grid Functions

This demonstration illustrates using the functions
`DATAGRID_MEAN` and `DATAGRID_SUBSETBYVALUE` in a
rule set.

1) Preview ProductQuery (SQL)

Demo_DataGridFunctions (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Search name: All types

```
graph TD; Start[Start] --> ProductQuery[ProductQuery (1.0)]; ProductQuery --> End[End]
```

Demo_DataGridFunctions (1.0)

Decision Flow Decision Properties **Variables** Scoring Versions History

Object type: All objects Object: Select an item Import

Variables	Data Type	Input	Output
ProductQuery_out	Data grid	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ProductQuery_returnCode_out	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ProductQueryRowCount_out	Integer	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Supplier_ID	Character	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Demo_DataGridFunctions (1.0) > ProductQuery (1.0)

Code Properties Variables Scoring Versions

```
/* include sqlReturnInfo */
select PRODUCT_NAME as {PRODUCT_NAME:string:45}, PRICE as {PRICE:decimal}
from EBDID.PRODUCTLIST
where SUPPLIER_ID = {SUPPLIER_ID:string:32}
```

2) Create Rule Set

- Add variable
- Add assignments

Demo_DataGridFunctions (1.0) > Demo_DGF (1.0)

Rule Set Properties Variables Scoring Versions History

Rule Set Variables Global Variables

Variables	Data Type	Input	Output
AveragePrice	Decimal	<input type="checkbox"/>	<input checked="" type="checkbox"/>
ProductQuery_out	Data grid	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ProductsPriceGreater100	Data grid	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Expression Editor: Default_assignment_1

Variables Functions + - * ÷ ** || () ! , LIKE IN NOTIN AND OR NOT = ≠ < >

```
AveragePrice = DATAGRID_MEAN(ProductQuery_out,'Price')
```

The expression is valid.

Validate Clear

Description	Column Name	Data Type	Length
	Price	decimal	
	Product_Name	string	45
	Supplier_ID	string	8

Expression Editor: Default_assignment_3

Variables Functions + - * ÷ ** || () ! , LIKE IN NOTIN AND OR NOT = ≠ < >

```
DATAGRID_SUBSETBYVALUE(ProductQuery_out,'Price','GT','100',ProductsPriceGreater100)
```

The expression is valid.

Validate Clear

Description	Column Name	Data Type	Length
	Price	decimal	
	Product_Name	string	45
	Supplier_ID	string	8

Demo_DataGridFunctions (1.0) > * Demo_DGF (1.0)

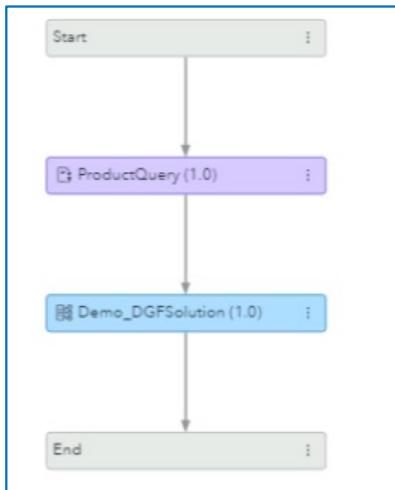
Rule Set Properties Variables Scoring Versions History

```
AveragePrice = DATAGRID_MEAN(ProductQuery_out,'Price')

DATAGRID_SUBSETBYVALUE(ProductQuery_out,'Price','GT','100',ProductsPriceGreater100)

+ Add Rule
```

3) Add Rule Set (with DataGrid) to Decision Flow



Demo_GridFunctions (1.0)

Decision Flow Decision Properties Variables Scoring Versions History

Tests Scenarios Publishing Validation

Name	Results	Status	Date Modified	Rule Set Version
<input checked="" type="checkbox"/> Demo_GridFunctions ...			Jun 11, 2023 06:33 AM	1.0

New Test Run :

Rule Set Version:

4) Preview the results > \$100

ProductQuery_out

PRODUCT_NAME	PRICE
Bill London Calling Shoes	130
Bill V-3-London Calling Shoes	140
Cruise Roadstar Leather Shoes	125
Vector Low Men's Shoes	215
Men's Nubuck-Retro Running Shoes Ro...	124
Allinall Star 2000 Women's Canvas Shoes	115
Allinall Star Men's Canvas Shoes	78
Cruise 1984 Men's Street Shoes	109

ProductsPriceGT100

PRODUCT_NAME	PRICE
Bill London Calling Shoes	130
Bill V-3-London Calling Shoes	140
Cruise Roadstar Leather Shoes	125
Vector Low Men's Shoes	215
Men's Nubuck-Retro Running Shoes Ro...	124
Allinall Star 2000 Women's Canvas Shoes	115
Cruise 1984 Men's Street Shoes	109
Cruise Men's Road Chunk Street Shoes	121

Close **Close**



7) Scoring Rows in a Data Grid

This demonstration illustrates using a rule set to score rows in a data grid and calculate a discount price for products offered by vendors with an average price of more than 150.

1) New Rule set & add variables

The screenshot displays the SAS Intelligent Decisions - Build Decisions interface with three main windows:

- Edit Columns (Top Window):** A modal window titled "Edit Columns" with a blue header. It contains three radio button options:
 - Add a new column
 - Add columns from a data table
 - Add columns from a data gridA table with three columns (Columns, Data Type, Length) is shown, with one row added:

Columns	Data Type	Length
DiscountPrice	Decimal	
Price	Decimal	
Product_Name	Character	45
Supplier_ID	Character	8
- Add Variables (Middle Window):** A modal window titled "Add Variables" with a blue header. It shows the "Decision:" path: "/My Folder/Decisioning/Demonstrations/Demo_DataGridScoreRows". The "Available items (5):" list includes:
 - ProductQuery_out
 - ProductQuery_returnCode_out
 - ProductQueryRowCount_out
 - ProductsPriceGT100
 - Supplier_IDThe "Selected items (1):" list contains "AveragePrice" with a description "decimal-output-Demo_DataGridScoreRows".
- Rule Set Variables (Bottom Window):** The main application window showing the "Rule Set" tab selected. It lists variables with their data types and checkboxes for "Input" and "Output":

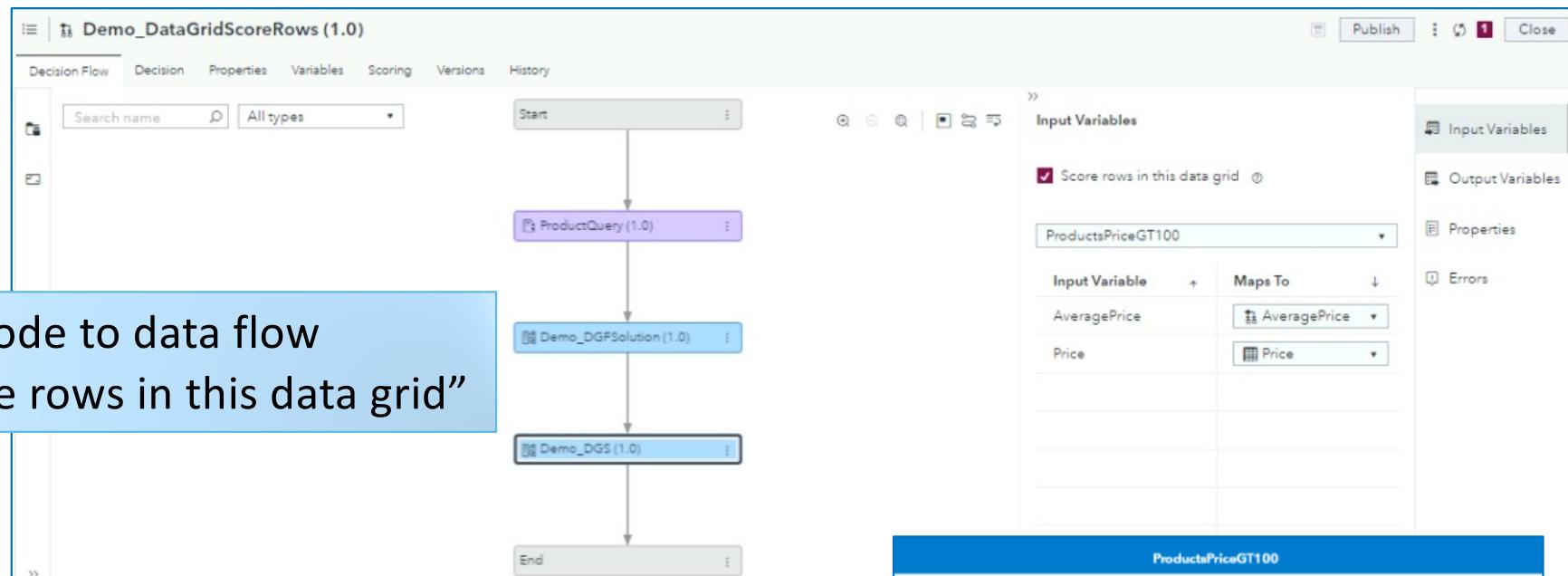
Variables	Data Type	Input	Output
AveragePrice	Decimal	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DiscountPrice	Decimal	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Price	Decimal	<input checked="" type="checkbox"/>	<input type="checkbox"/>

2) Add rules in Rule Set

The screenshot shows the 'Rule Set' tab selected in the 'Demo_DGS (1.0)' interface. The rule 'Default_rule_1' is expanded, showing the following logic:

- IF**: AveragePrice > 150 THEN ASSIGN DiscountPrice = Price * 0.9
- ELSE** THEN ASSIGN DiscountPrice = Price

A checkbox labeled "Record rule-fired data" is checked. A blue box highlights the rule title and the first condition.



4) Test in scoring tab
 - The discount column is an output in the data grid

The screenshot shows the 'Scoring' tab of the 'Demo_DataGridScoreRows (1.0)' project. The table lists tests under the 'Tests' tab. One test is selected, showing the 'Name' column as 'Demo_DataGridScoreRow...' and the 'Status' column as 'Success'. The 'Results' column shows a small icon. The 'Date Modified' column shows 'Apr 28, 2021 02:51 PM'. The 'Rule Set Version' column shows '1.0'. The 'Properties' section on the right shows the 'Name' as 'Demo_DataGridScoreRows (1.0)' and the 'Status' as 'Success'.

ProductsPriceGT100

The screenshot shows a data grid titled 'ProductsPriceGT100'. The grid has three columns: 'PRODUCT_NAME', 'PRICE', and 'DISCOUNTPRICE'. The data includes the following rows:

PRODUCT_NAME	PRICE	DISCOUNTPRICE
Bill London Calling Shoes	130	130
Bill V-3-London Calling ...	140	140
Cruise Roadstar Leathe...	125	125
Vector Low Men's Shoes	215	215
Men's Nubuck-Retro Ru...	124	124
Allinall Star 2000 Wom...	115	115
Cruise 1984 Men's Stre...	109	109
Cruise Men's Road Chu...	121	121

Lesson 4: Treatments, Treatment Groups, and Arbitration

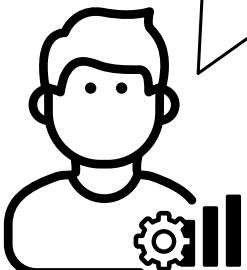
4.1 Treatments and Treatment Groups

4.2 Treatment Arbitration

Lesson 4: Treatments, Treatment Groups, and Arbitration

4.1 Treatments and Treatment Groups

4.2 Treatment Arbitration



Suppose you want to assign values to a set of variables based on a condition.

Rule Set

If Condition

Assign Var1

Assign Var2

Assign Var3

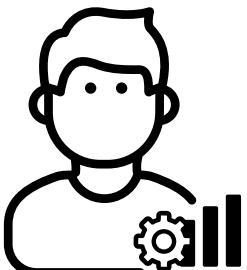
Treatment

Eligibility rule

Attribute 1

Attribute 2

Attribute 3

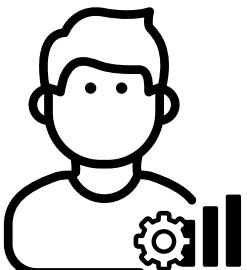


Treatments are a collection of attribute values.

Treatment A

- Text
- Datetime
- \$ € £ ₣
- Other

Attributes



Treatments can have **eligibility rules**.

Treatment A

Text

Datetime

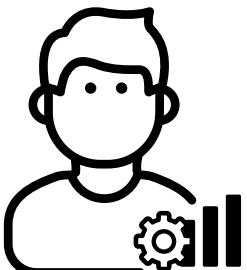
\$ € £ ₣

Other

Eligibility rule

Filtering Rule Set

IF condition



You add treatments
to a treatment group
for use in a decision.

Treatment Group

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

Attribute 3

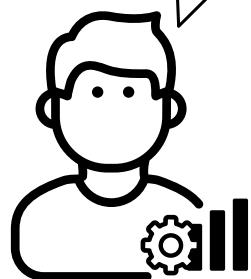
Treatment C

Attribute 1

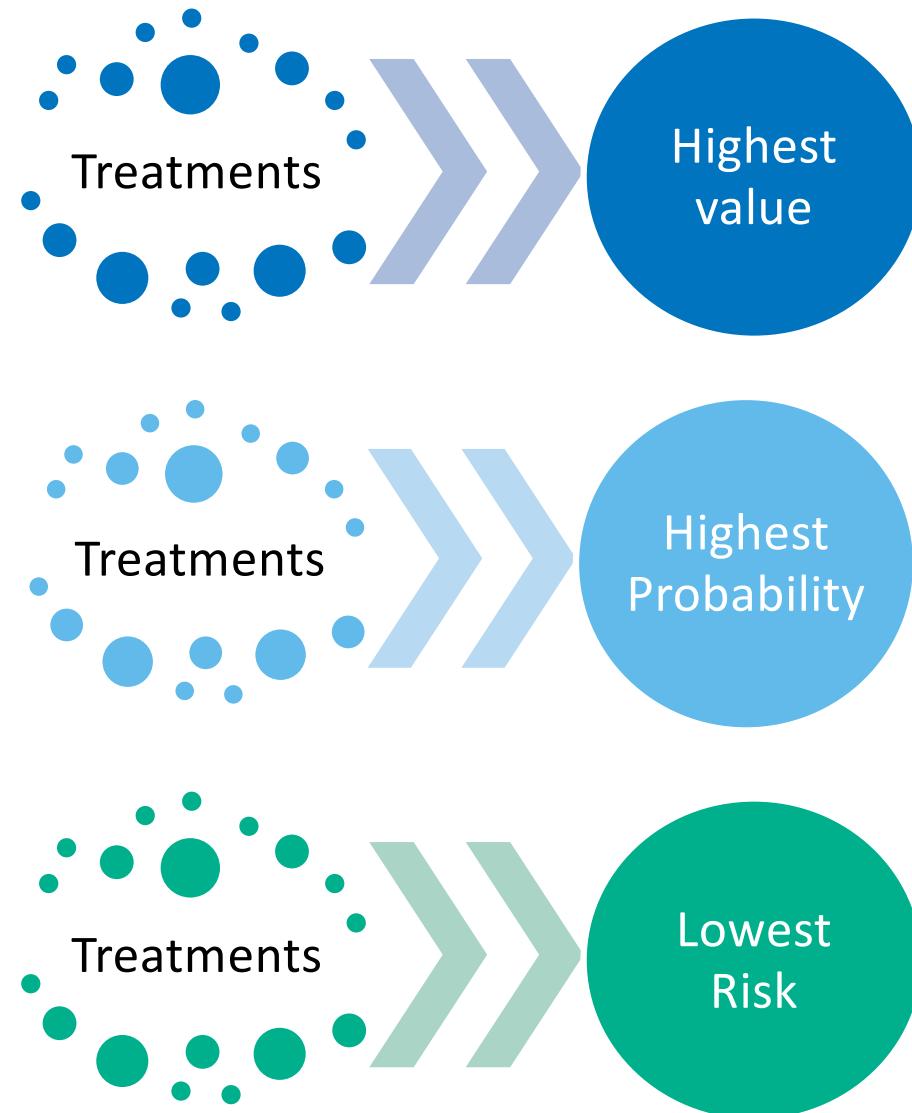
Attribute 2

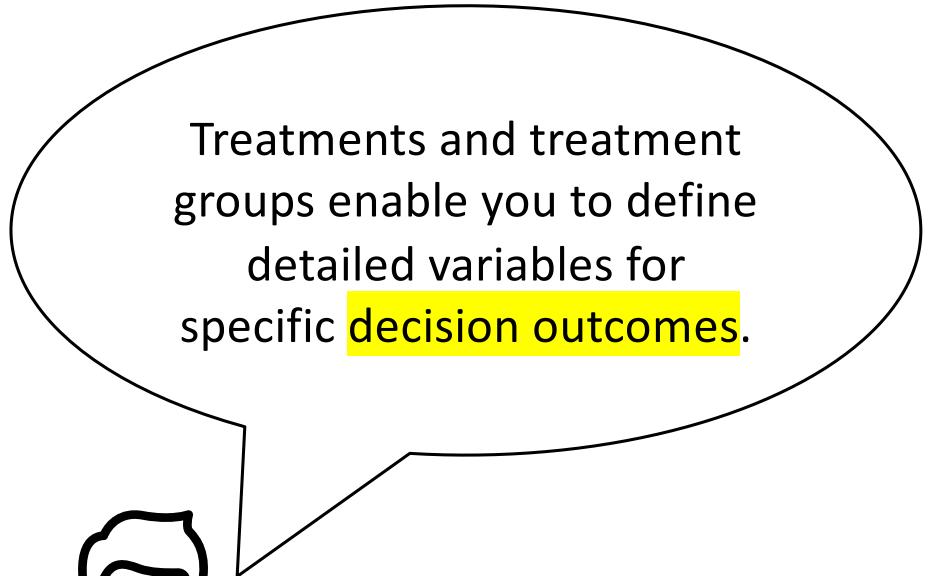
Attribute 3

Treatment Arbitration

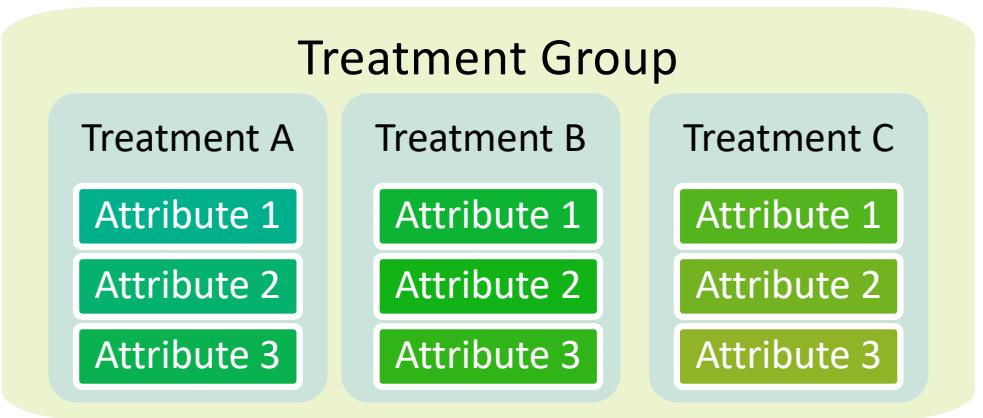
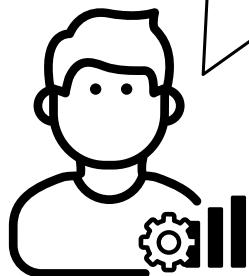


You can further refine the treatments returned by the decision.





Treatments and treatment groups enable you to define detailed variables for specific decision outcomes.

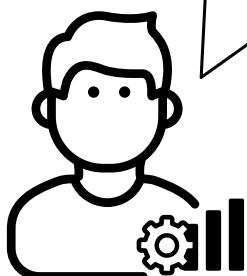


Specific
decision
outcomes

↑ Consistency

↑ Efficiency

You can use treatment groups in multiple decision flows.



Treatment Group

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

Attribute 3

Treatment C

Attribute 1

Attribute 2

Attribute 3

Decision 1

Decision 2

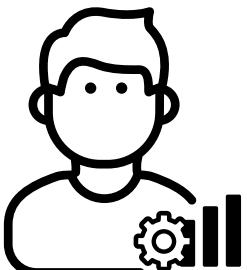
Decision 3



Treatments



Copyright © SAS Institute Inc. All rights reserved.

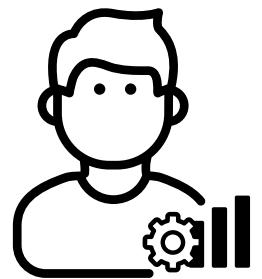


Treatments are a collection of attribute values.

Treatment A

- Text
- Datetime
- \$ € £ ₣
- Other

Attributes

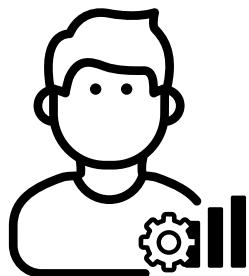


You define
treatment
properties.





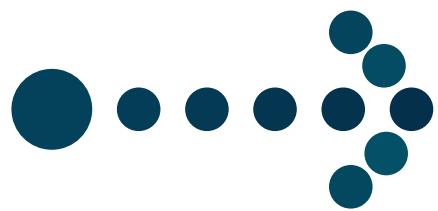
Treatments have attributes.

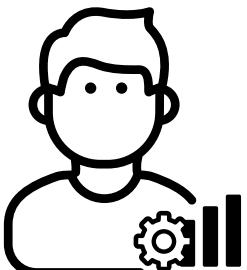


Attributes

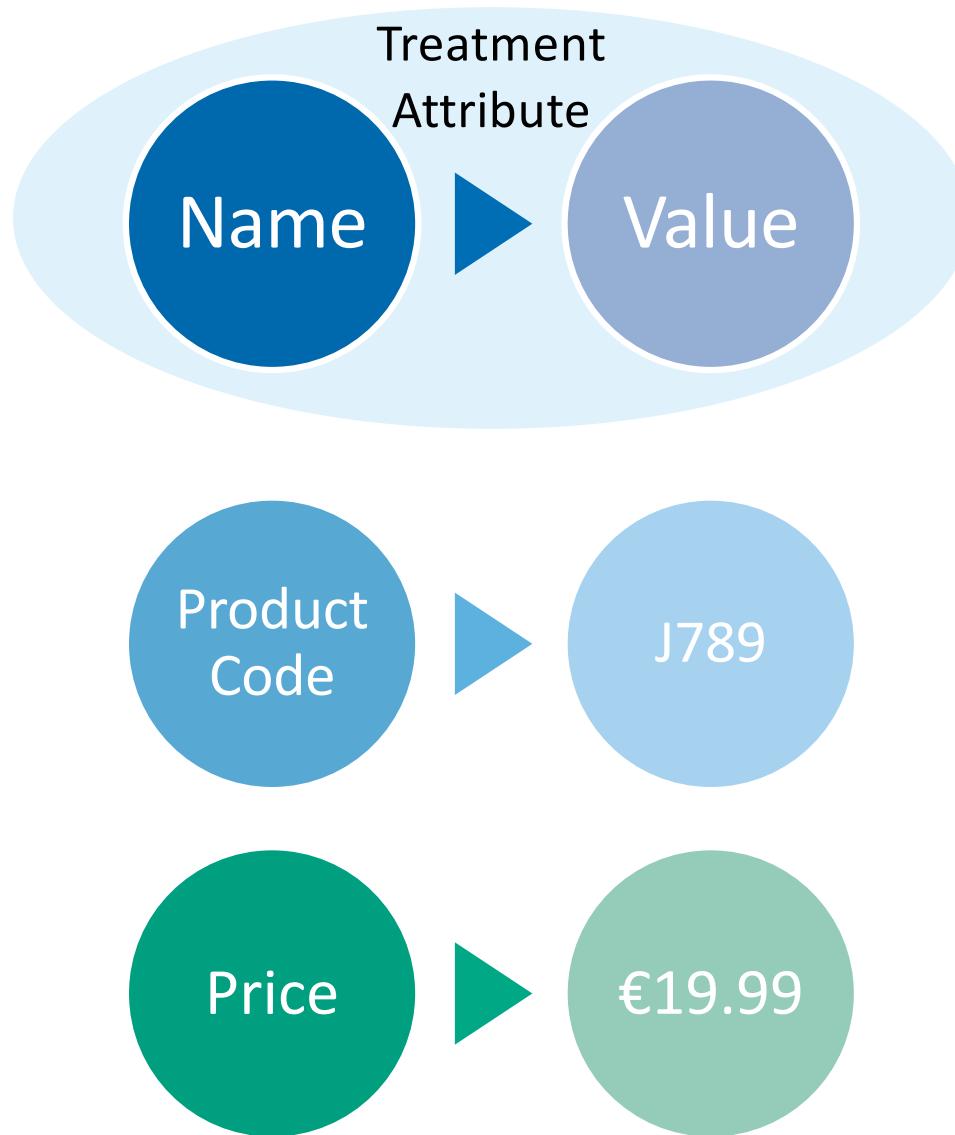


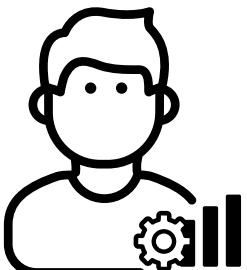
Treatment





Attributes are name-value pairs that define specific details.





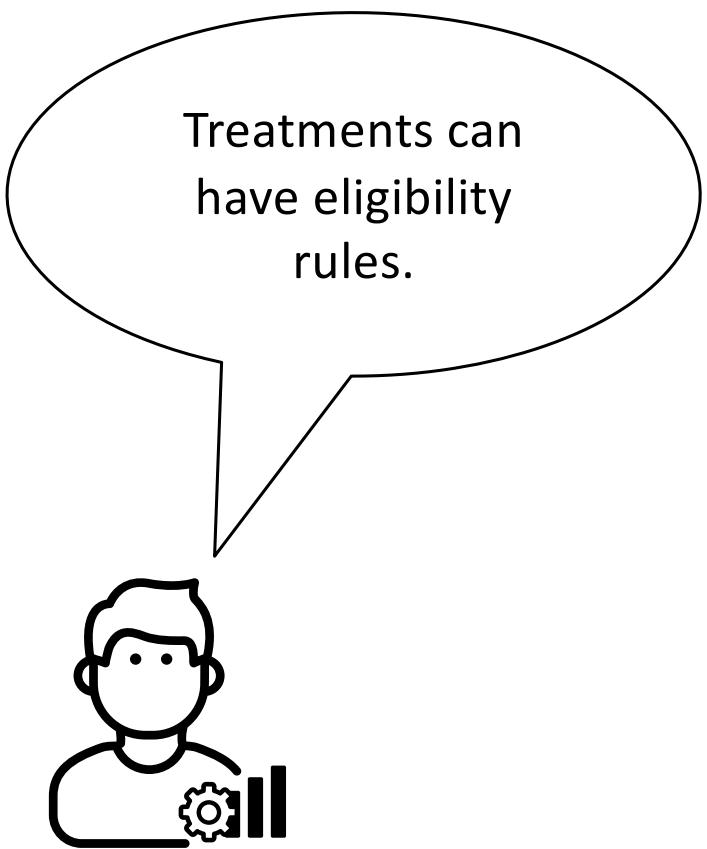
The attribute can
be fixed or
dynamic.

Fixed

Each decision uses the
same value.

Dynamic

Each decision can use a
different value.

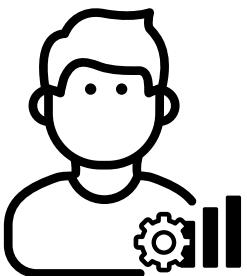


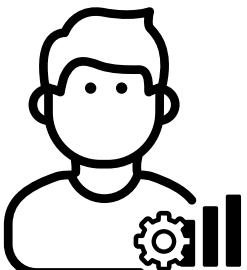
Treatments can
have eligibility
rules.

Eligibility rules



Treatment





Eligibility is determined by the criteria in a filtering rule set.

Treatment eligibility



Filtering Rule Set

Rule

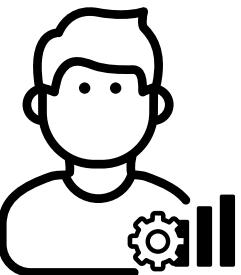
IF Condition

Rule

IF Condition

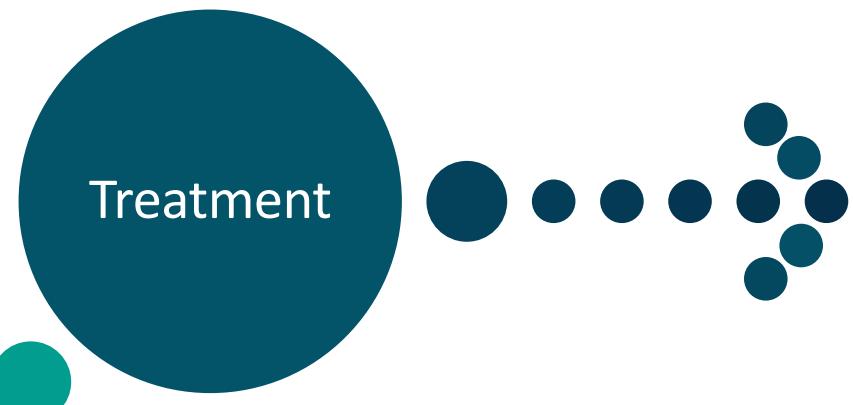
Rule

IF Condition

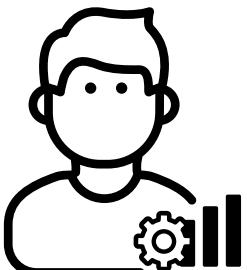


Treatments can have effective dates.

Effective dates

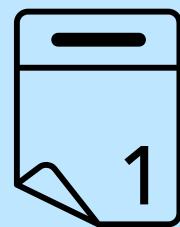


Treatment

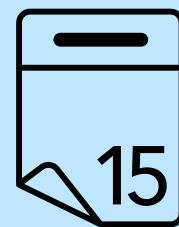


You can set a starting date and ending date for a treatment.

Treatment Effective Dates



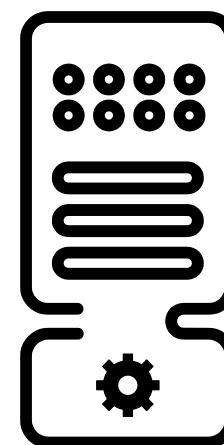
Start

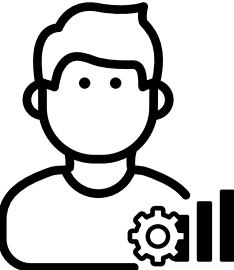


End



Server
time zone





You should **lock** a treatment before adding to a treatment group.

Best Practice

Treatment A

Attribute 1

Attribute 2

Attribute 3

Lock



Treatment Group

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

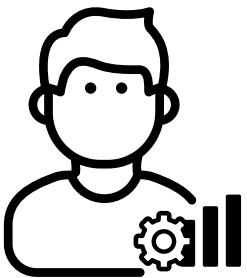
Attribute 3

Treatment C

Attribute 1

Attribute 2

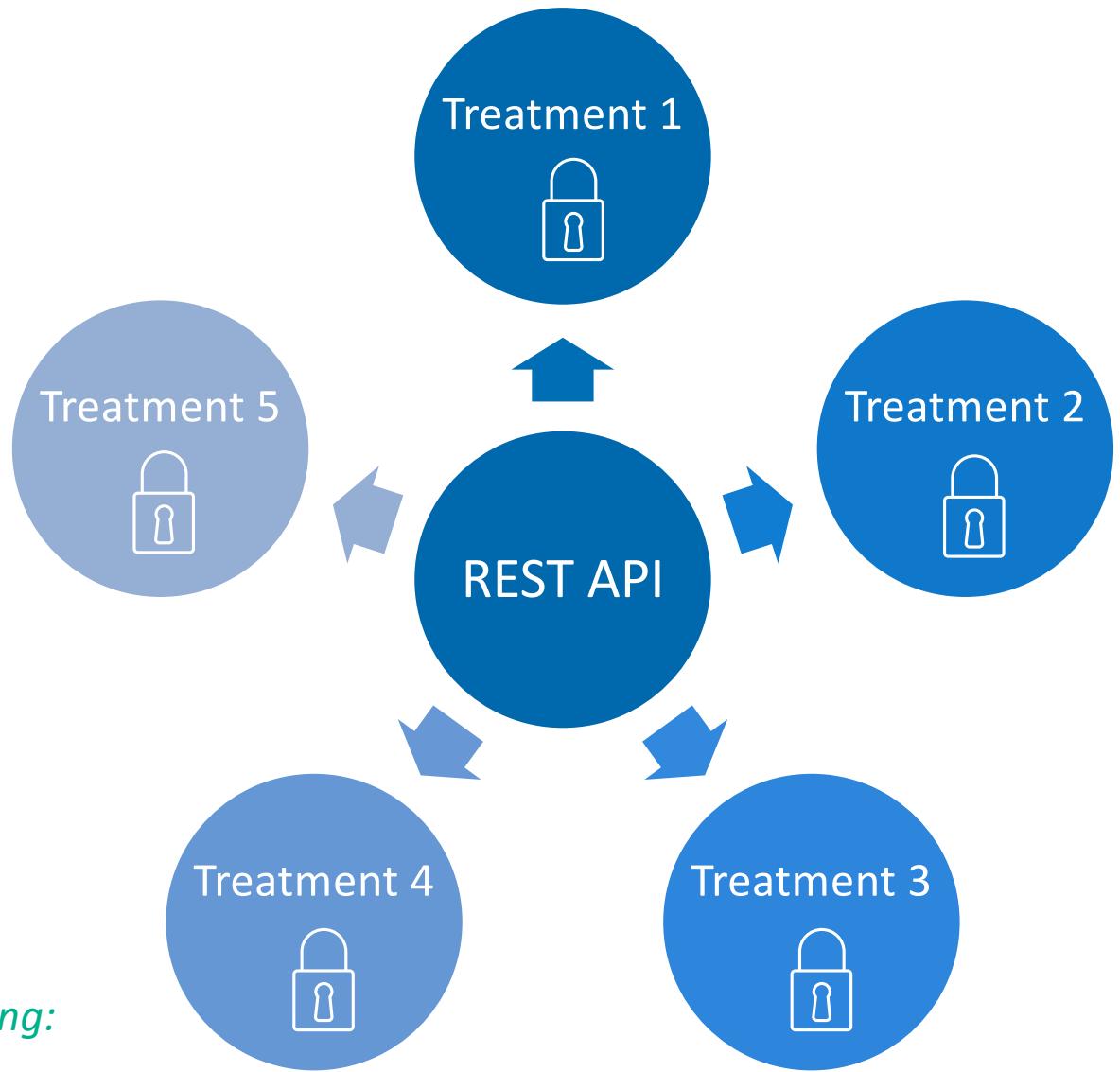
Attribute 3



You can use an API
to lock multiple
treatments.



*SAS Intelligent Decisioning:
REST API Examples*





Creating and Locking a Treatment

This demonstration shows how to create a treatment.



Creating a Treatment Eligibility Rule

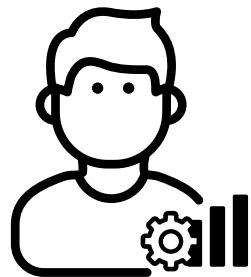
This demonstration, shows how to create a treatment eligibility rule.

Treatment Groups



Copyright © SAS Institute Inc. All rights reserved.

A treatment group is a group of treatments that you can use in a decision.



Treatment Group

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

Attribute 3

Treatment C

Attribute 1

Attribute 2

Attribute 3

Treatments

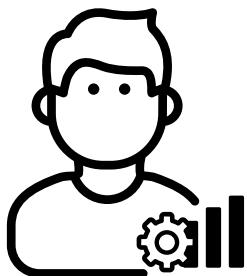
Treatment 1

Treatment 2

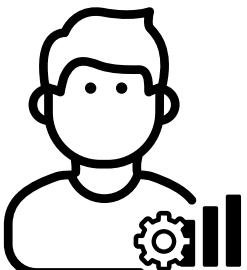
Treatment 3

...

You select existing treatments to add to the group.



Treatments



You can select the treatment version.

Treatment 1

Treatment 2

Treatment 3

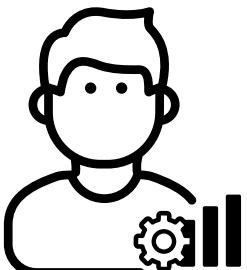
...



- Version 1.1
- Version 1.0



Select locked versions.



You can assign
aliases to the
treatment
attributes.

Treatment Group

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

Attribute 3

Treatment C

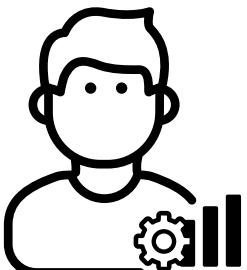
Attribute 1

Attribute 2

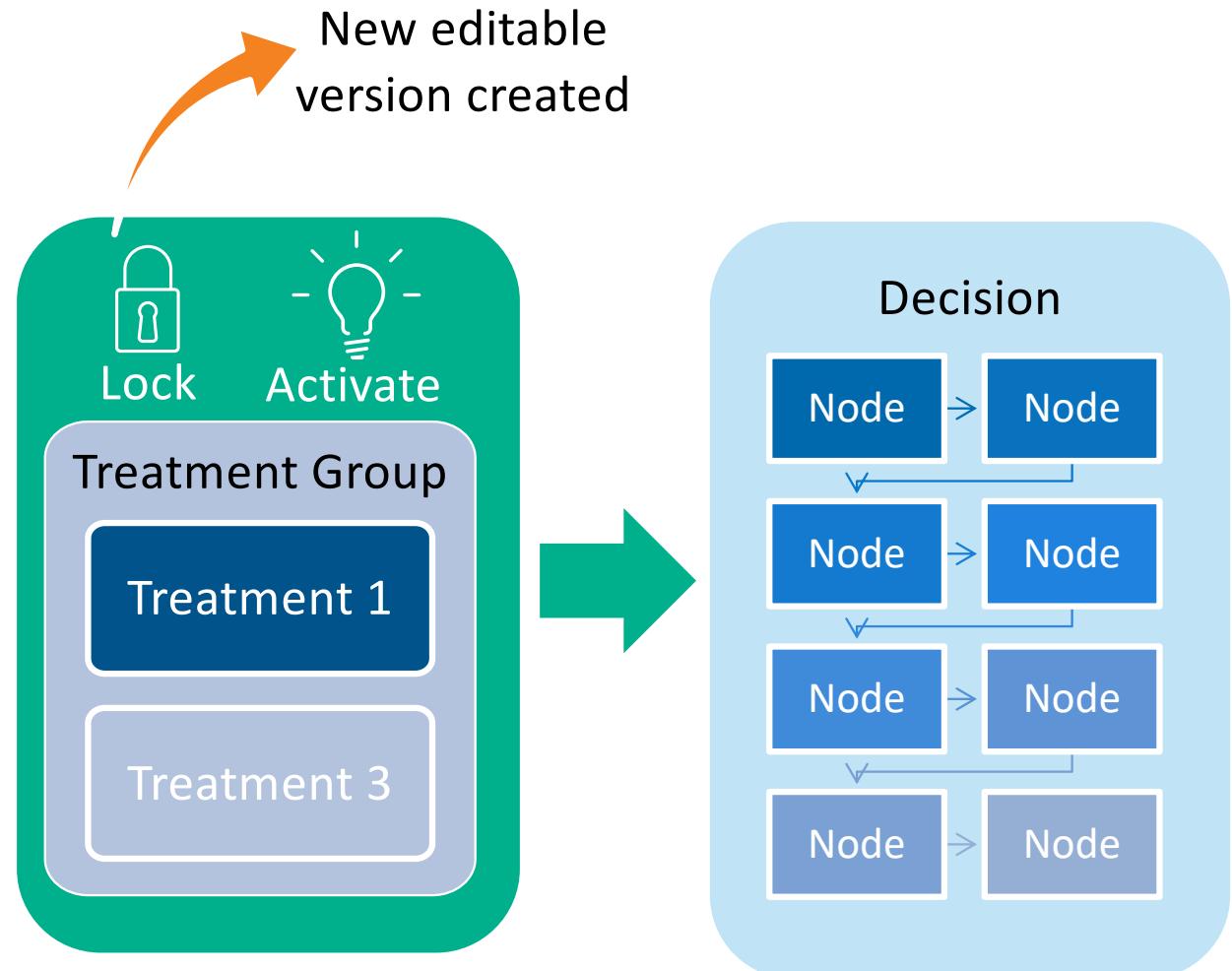
Attribute 3

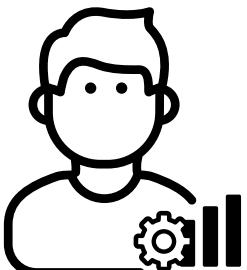


Data types
must match.



To use a treatment group in a decision, you must **lock** and then **activate** it.



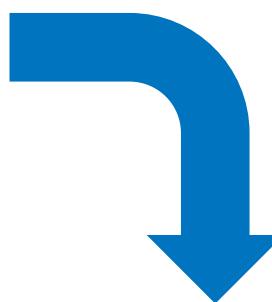
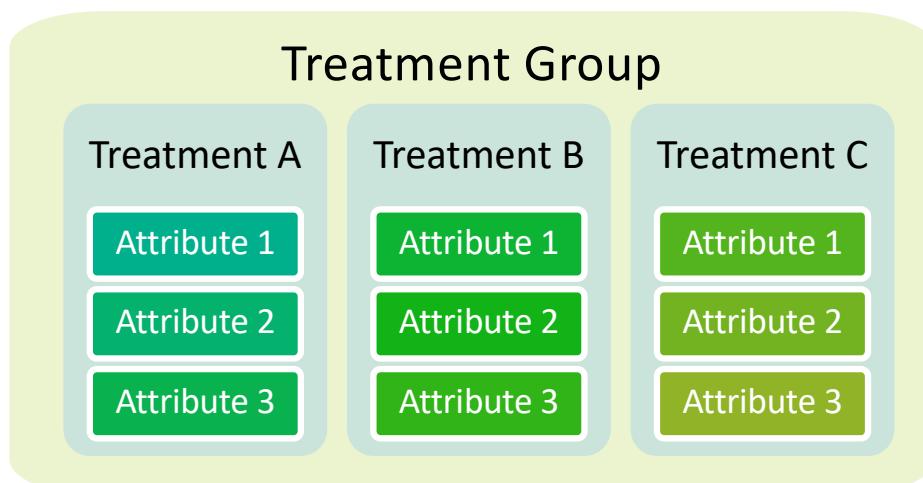


Dynamic treatment
attributes and
eligibility variables
are **input variables**.

Treatment Group Input Variables



- Dynamic Attribute 1
- Dynamic Attribute 2
- Eligibility Variable A
- Eligibility Variable B
- ...



Data Grid

Treatment	Attribute 1	Attribute 2	Attribute 3	Property 1	Property 2
Treatment A					
Treatment B					
Treatment C					

Treatment Attributes

Treatment Properties

Treatment Group 1

Treatment A

Attribute 1

Attribute 2

Attribute 3

Treatment B

Attribute 1

Attribute 2

Attribute 3

Treatment C

Attribute 1

Attribute 2

Attribute 3

Treatment Group 2

Treatment X

Attribute 1

Attribute 2

Attribute 3

Treatment Y

Attribute 1

Attribute 2

Attribute 3

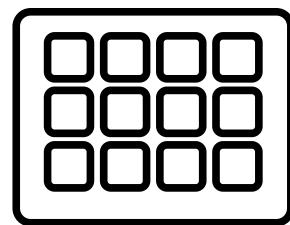
Treatment Z

Attribute 1

Attribute 2

Attribute 3

Combine



data grid

Data grid functions





Creating a Treatment Group

This demonstration illustrates how to create and activate a treatment group.



Adding a Treatment Group to a Decision

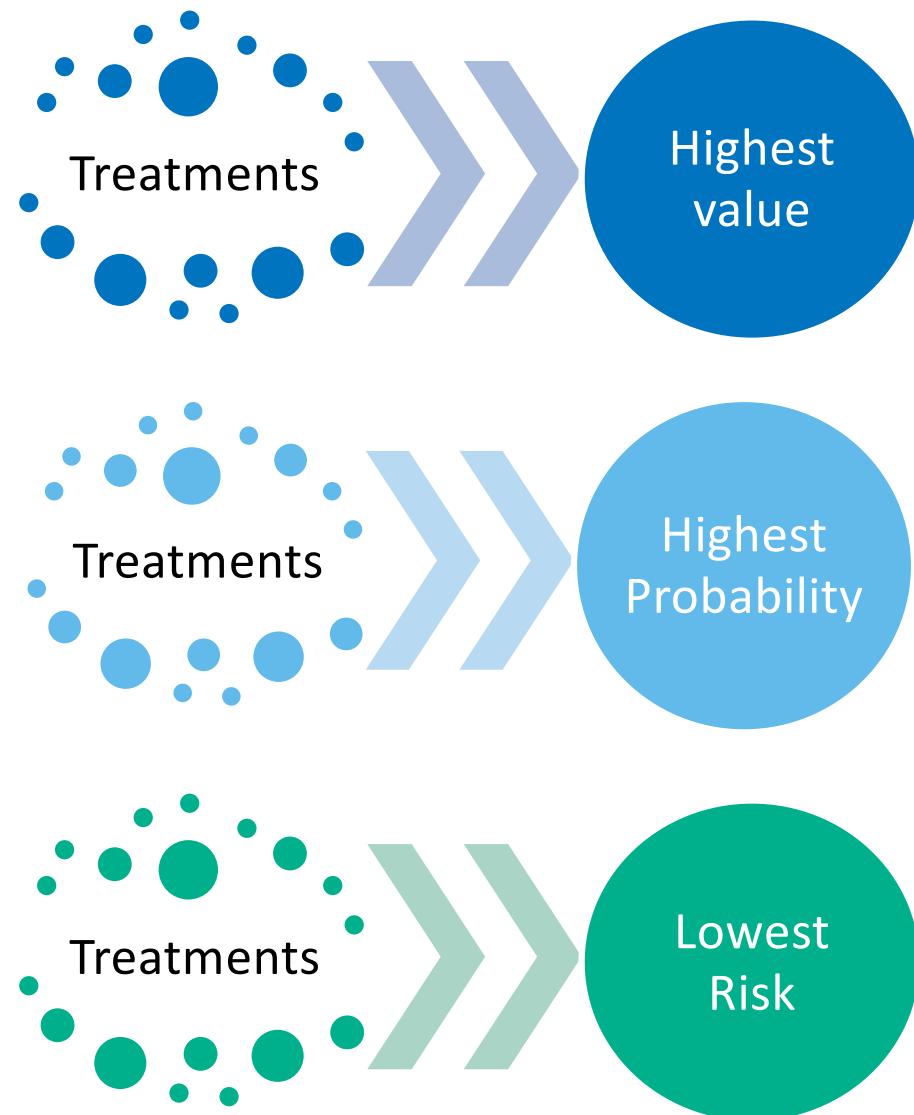
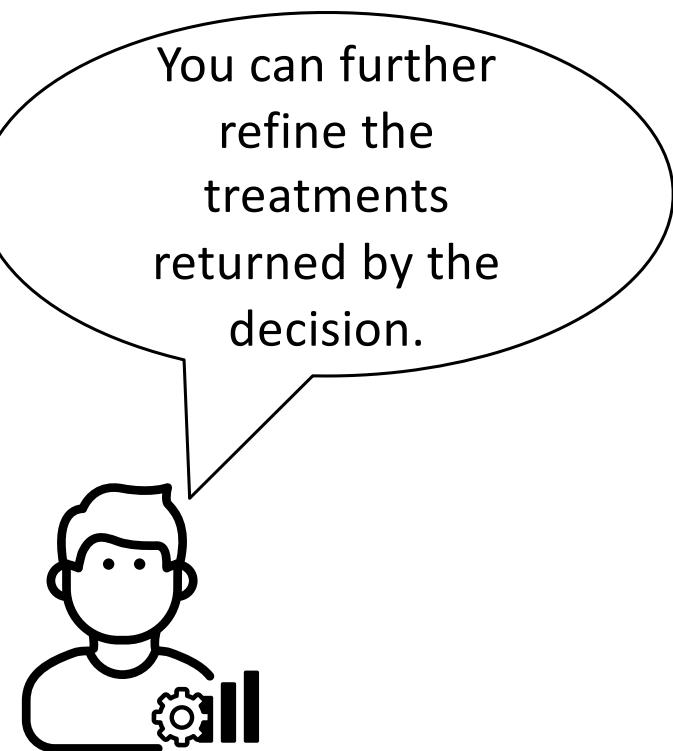
This demonstration illustrates how to add a treatment group to a decision and configure input variables.

Lesson 4: Treatments, Treatment Groups, and Arbitration

4.1 Treatments and Treatment Groups

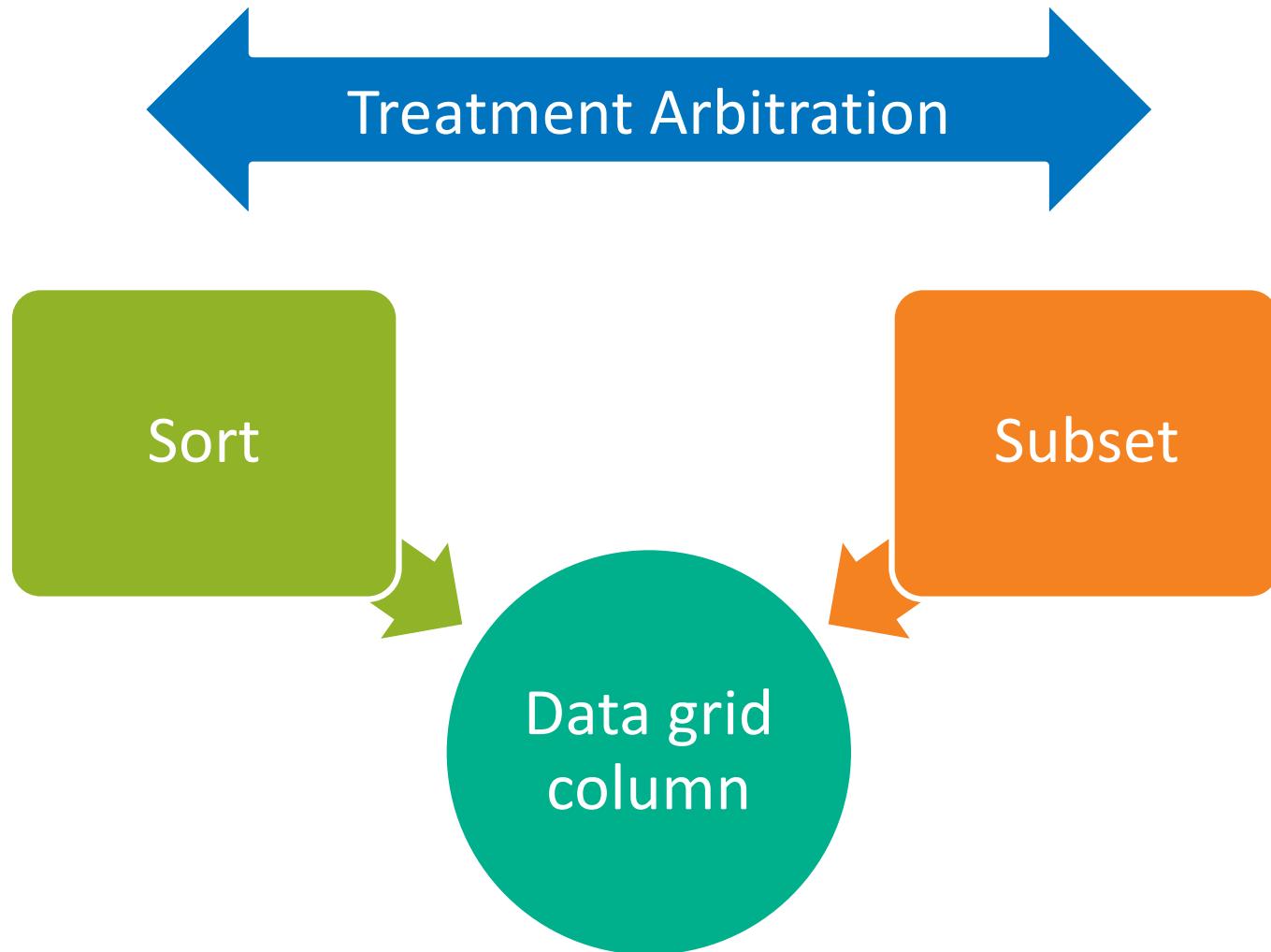
4.2 Treatment Arbitration

Treatment Arbitration



Eligibility Rules Met







Using an Assignment Rule Set for Treatment Arbitration

This demonstration illustrates creating and adding an assignment rule set to arbitrate treatments based on a model propensity score.

Lesson 5: Testing, Publishing, Validating, and Troubleshooting

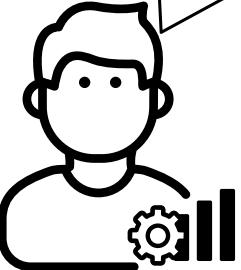
5.1 Testing

5.2 Publishing and Validating

Lesson 5: Testing, Publishing, Validating, and Troubleshooting

5.1 Testing

5.2 Publishing and Validating



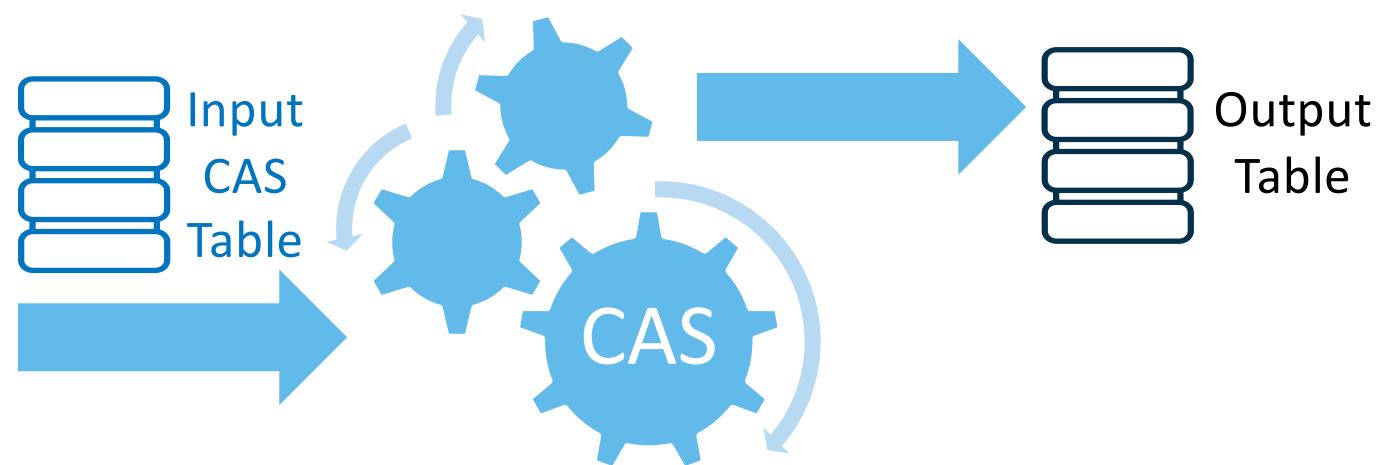
You can perform three types of tests.

Basic test

Scenario test

Publishing validation test

1) Basic test

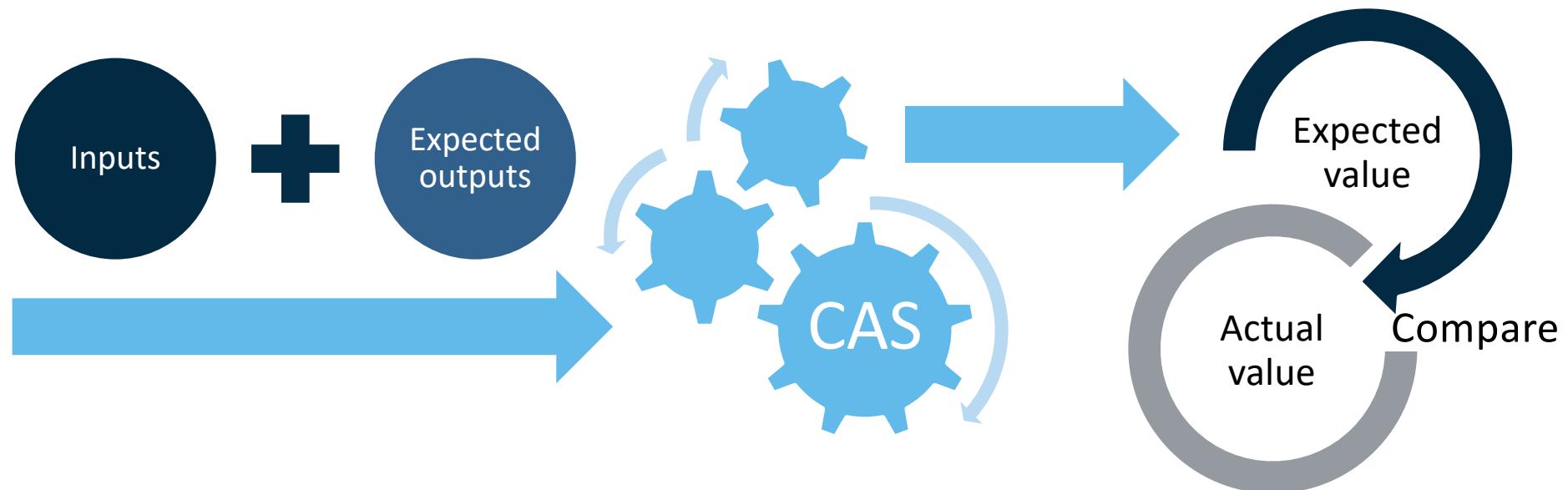


2) Scenario test

Decisions

Rule sets

Code files



3) Publishing validation test

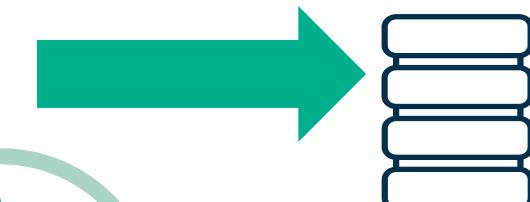


Publish

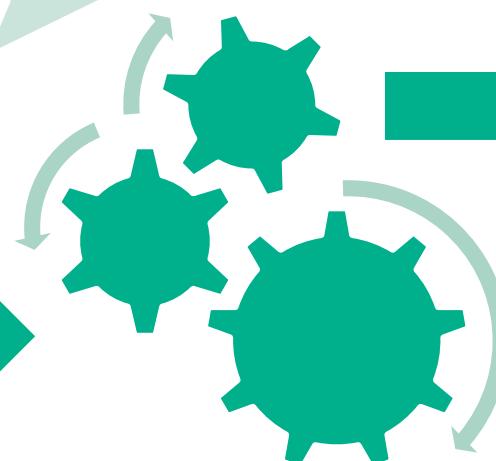
Validation
test defined



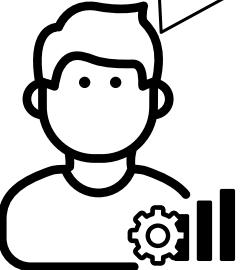
Input
Table



Output
Table



Publishing Destination

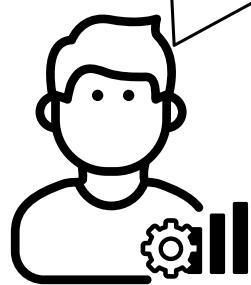


In some situations, you can generate additional results.

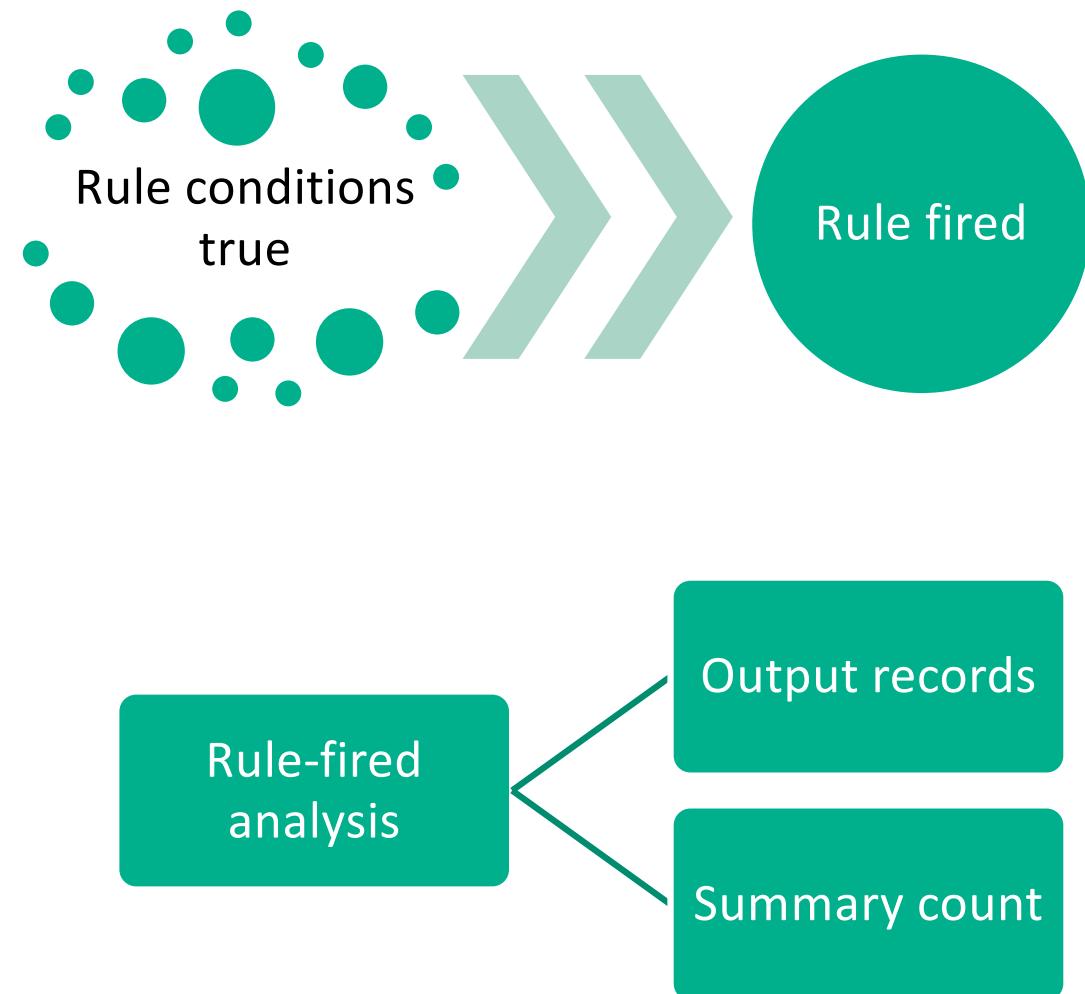
Additional Results

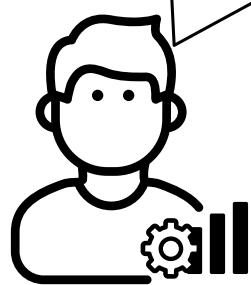
Rule-fired analysis

Decision path tracking

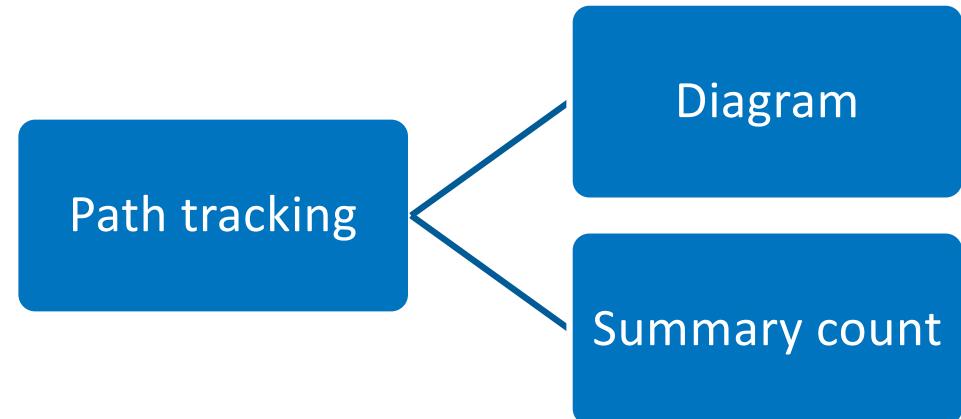
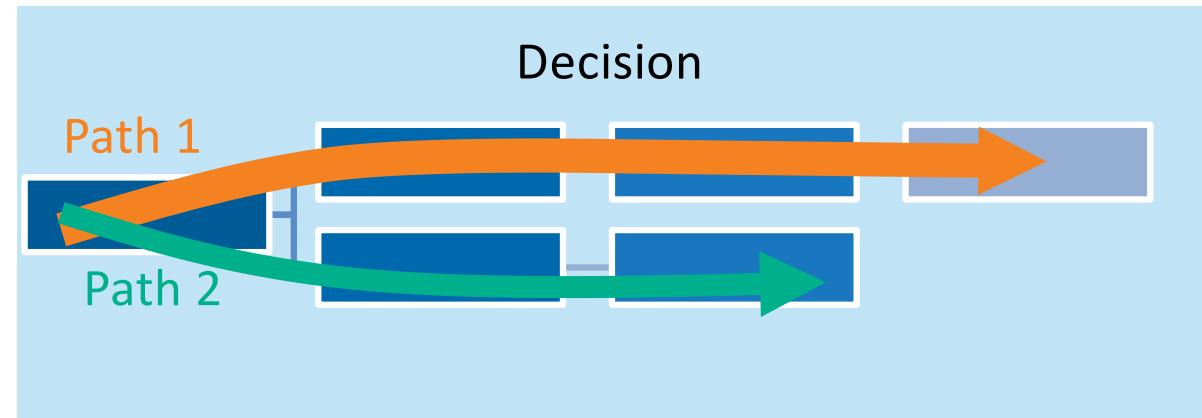


1) Rule-fired analysis shows information about when a rule evaluates as true.





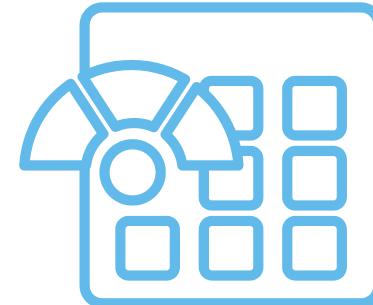
2) Path tracking
provides information
about the routes that
records take through
your decision.



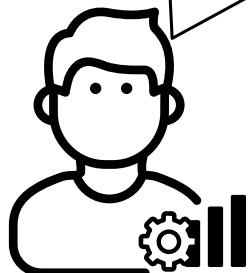
Output data library

CAS

Other repository



Test Results



You select a data library
for the test results.
Follow naming rules for
your repository.

Table name = test name

Version



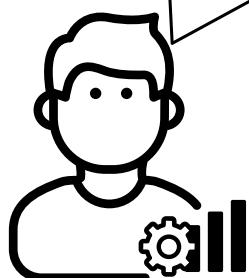
1.0

1.1

1.2

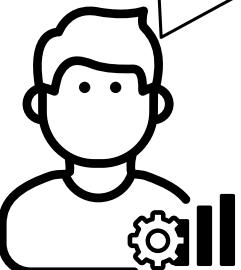
2.0

You select the version to
use for the test.





Best Practice



You can follow best practices for testing as you develop a decision.

Test incrementally.

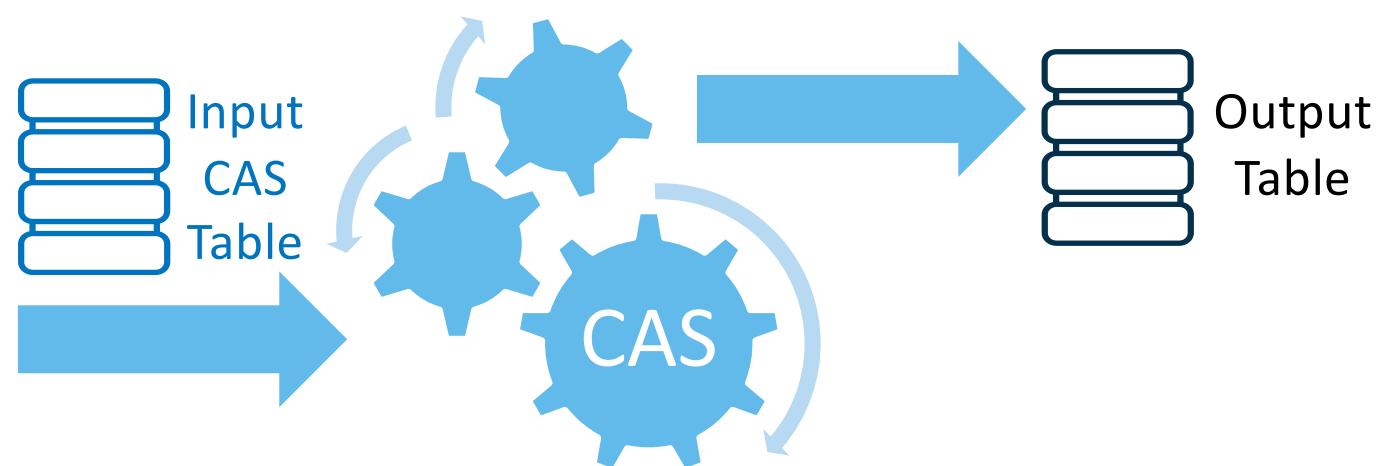


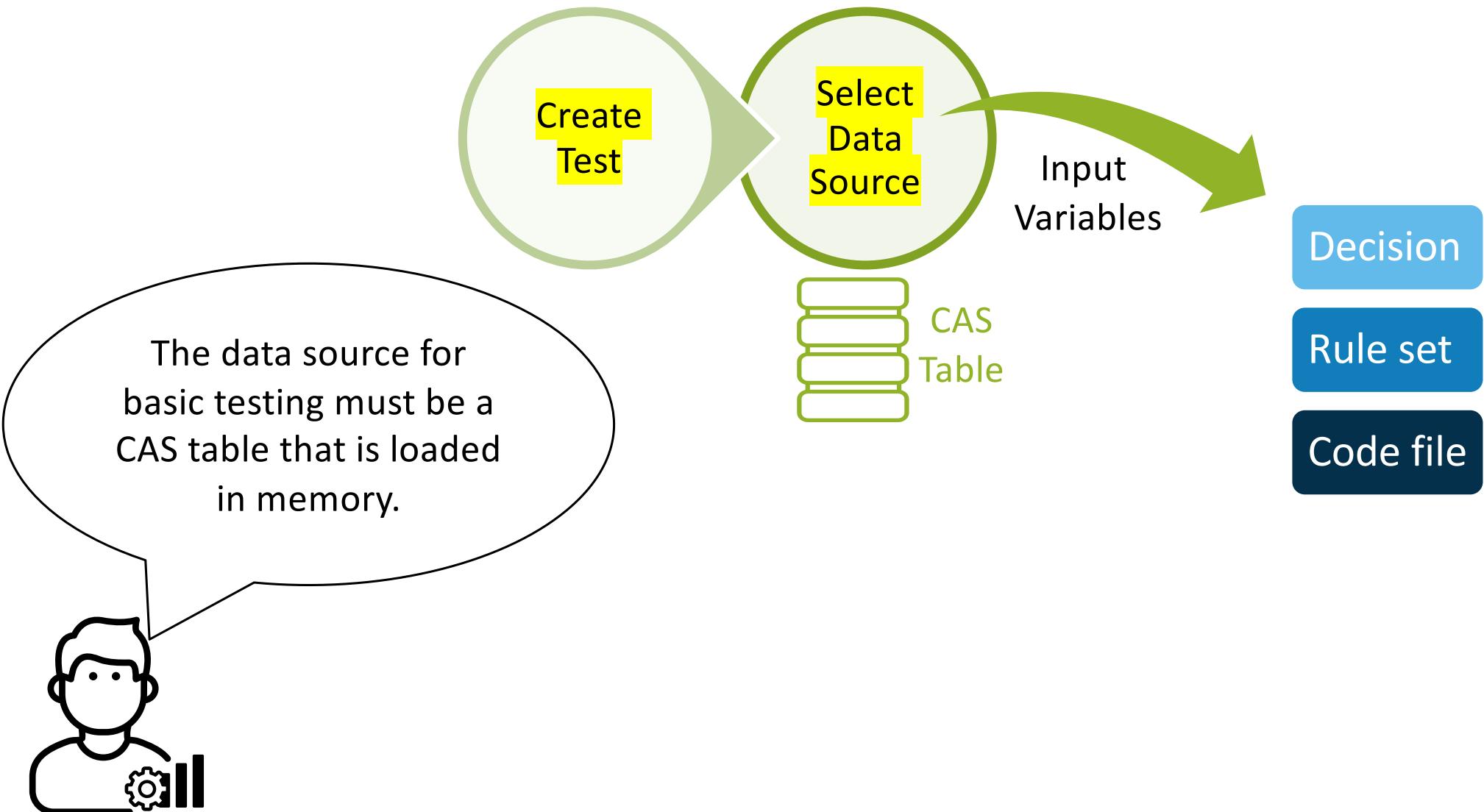
Develop

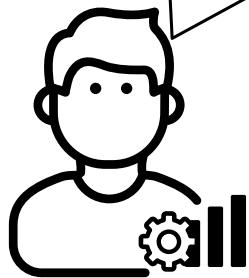


Test

1) Basic test







SAS Intelligent
Decisioning
automatically maps
columns using their
name and type.

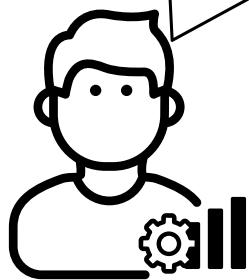


CAS
Table



- Same name
- Same type

- Decision
- Rule set
- Code file



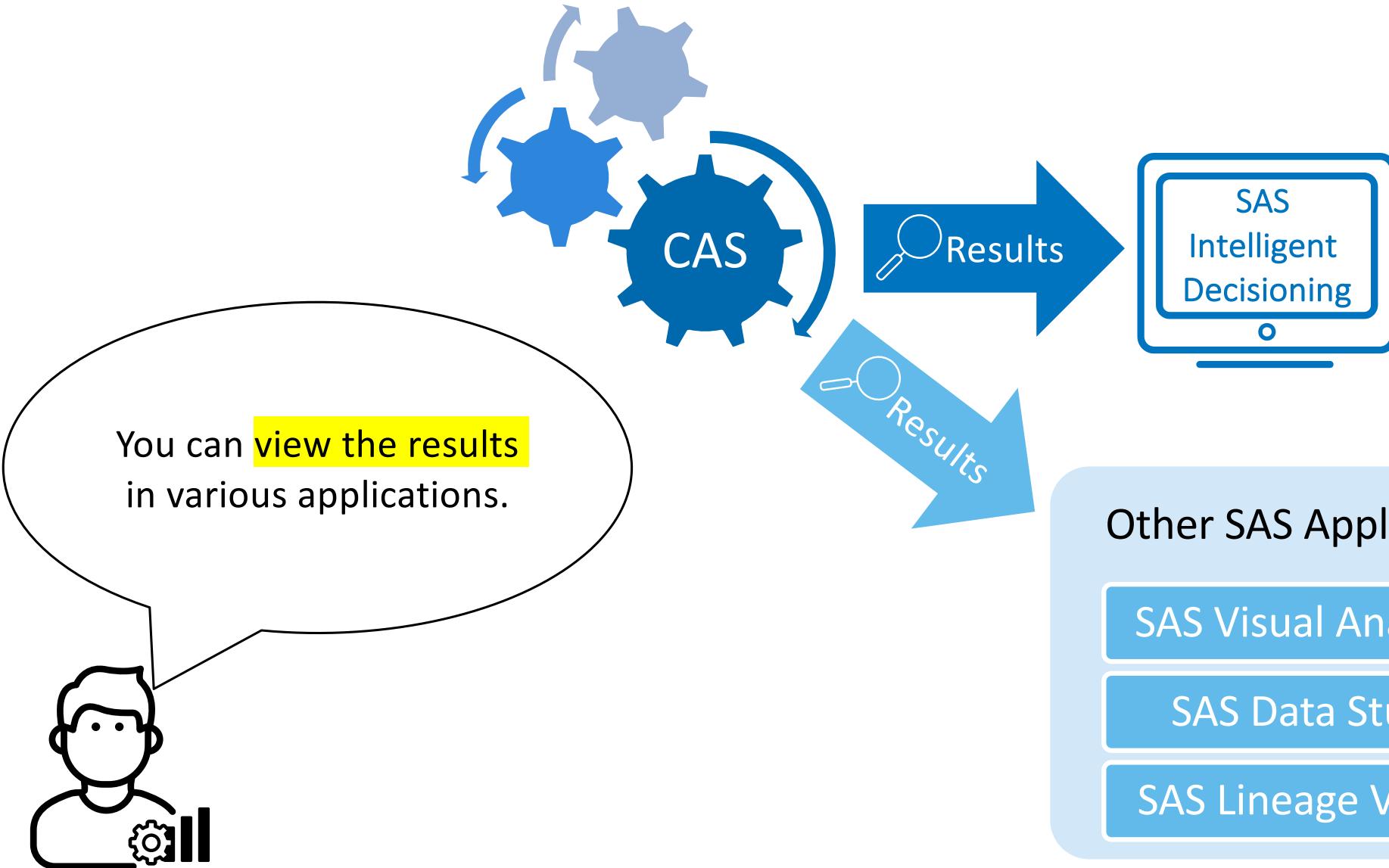
You must map any unmatched input variables manually.

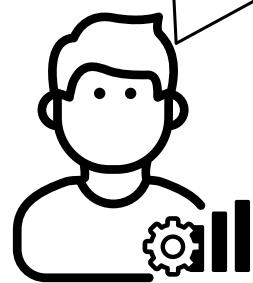


CAS
Table

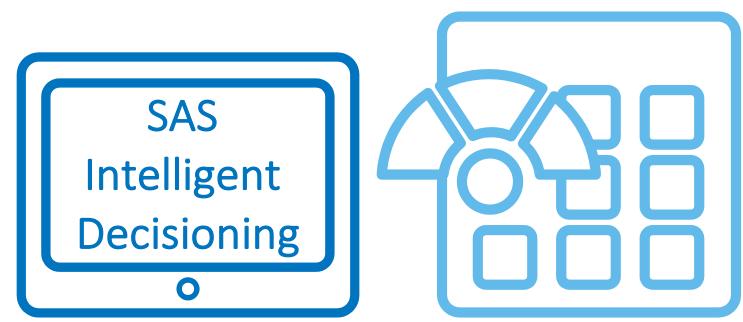
- Map manually.
- Enter constant.

- Decision
- Rule set
- Code file

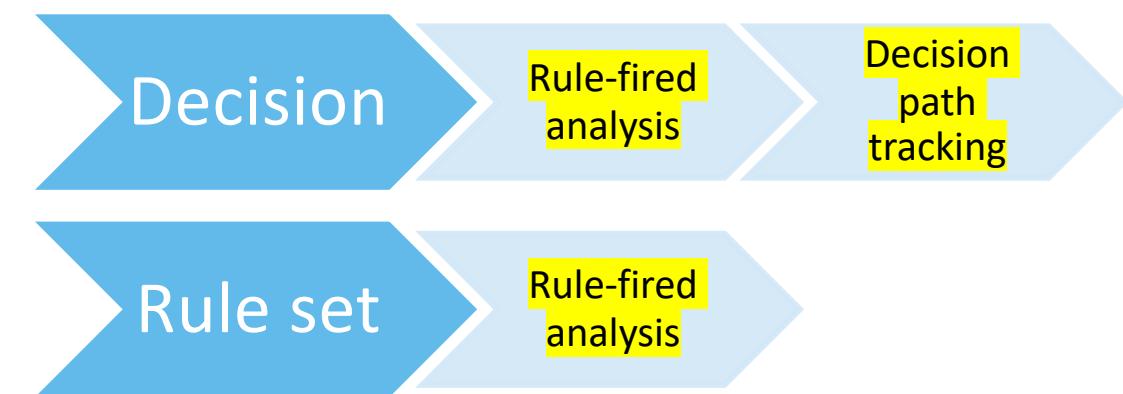




You can execute
additional tests from
basic test results.



Basic Test Results





1) Basic Testing of a Rule Set

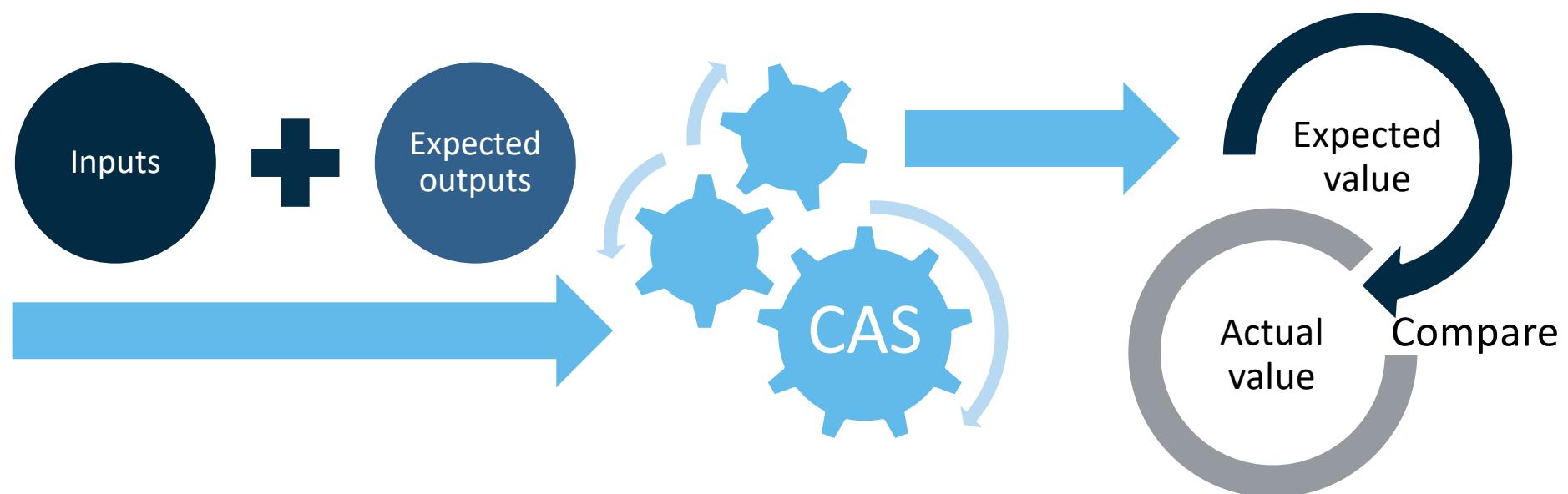
This demonstration how to perform basic testing of a rule set with advanced options.

2) Scenario test

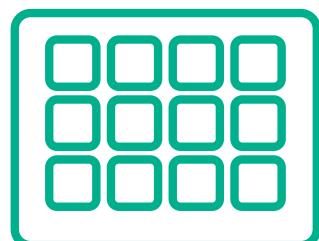
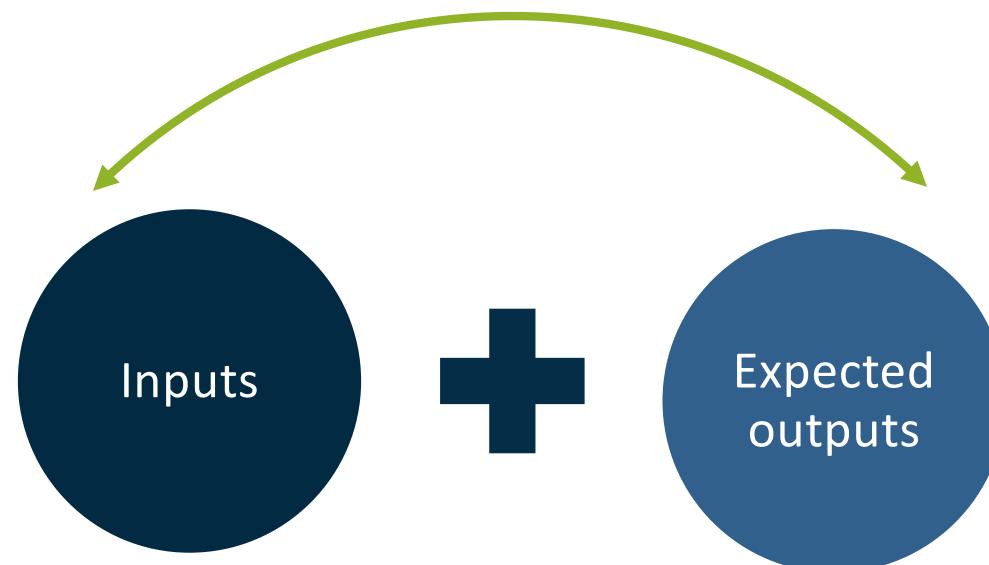
Decisions

Rule sets

Code files



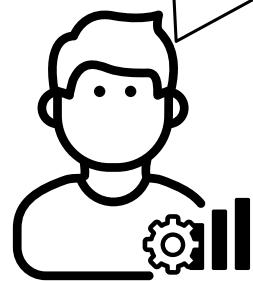
- Define manually.
- Import from CSV.



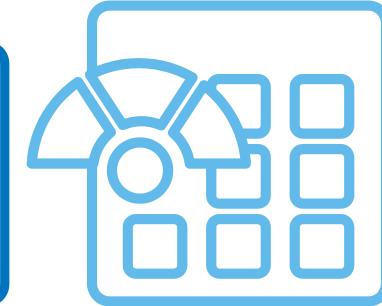
Data Grid Variable



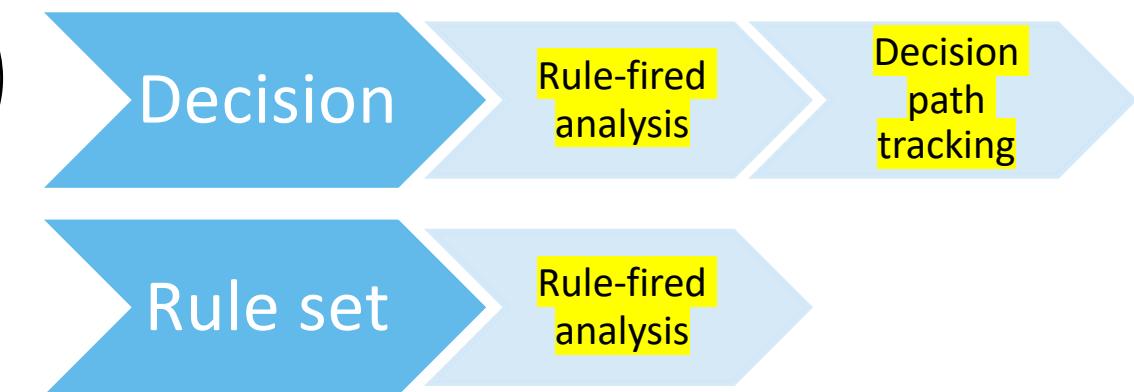
JSON string



You can execute additional tests from scenario test results.



Scenario Test Results





2) Scenario Testing of a Decision

This demonstration illustrates how to perform scenario testing of a decision.

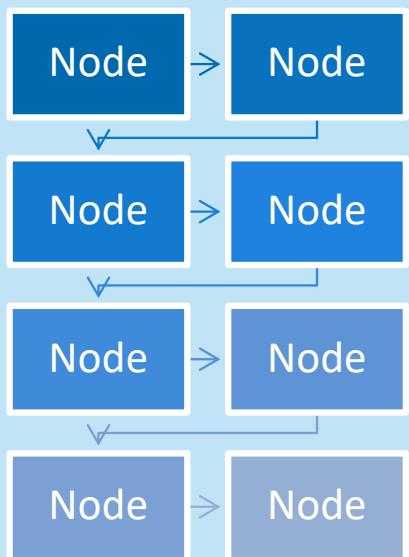
Lesson 5: Testing, Publishing, Validating, and Troubleshooting

5.1 Testing

5.2 Publishing and Validating

Design Environment

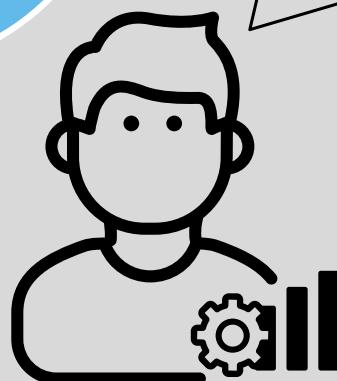
Decision Flow



Publish

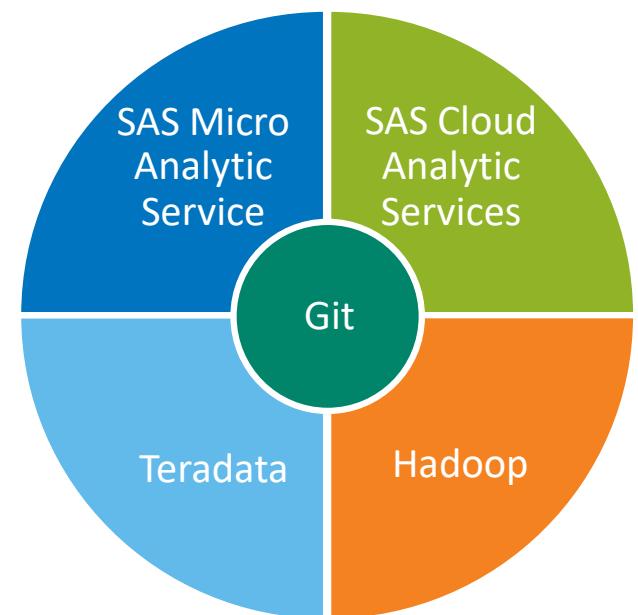
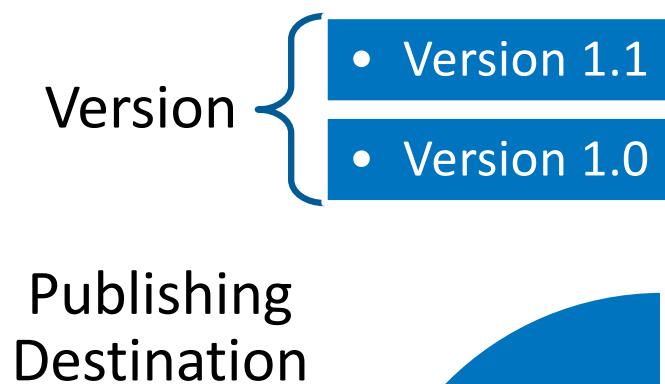
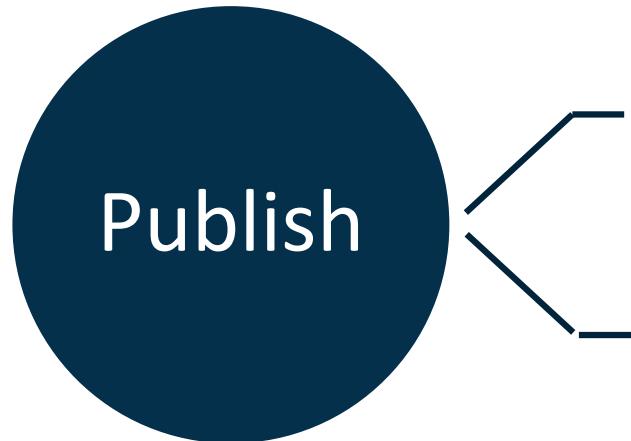
Publishing
Destination

When decision
development and
testing is complete,
you publish.





You choose a version
and publishing destination.



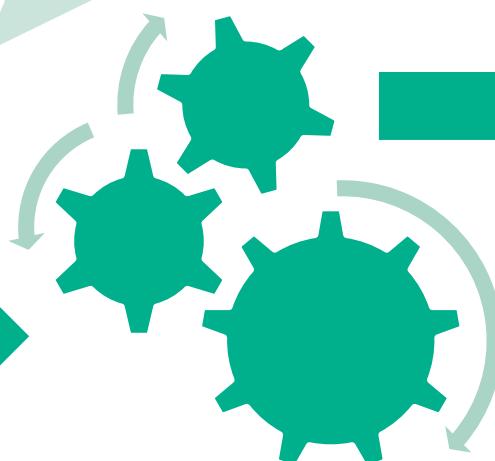
Publishing validation test

Decisions

Rule sets

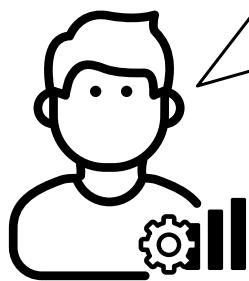
Publish

Validation test defined



Output Table

Publishing Destination

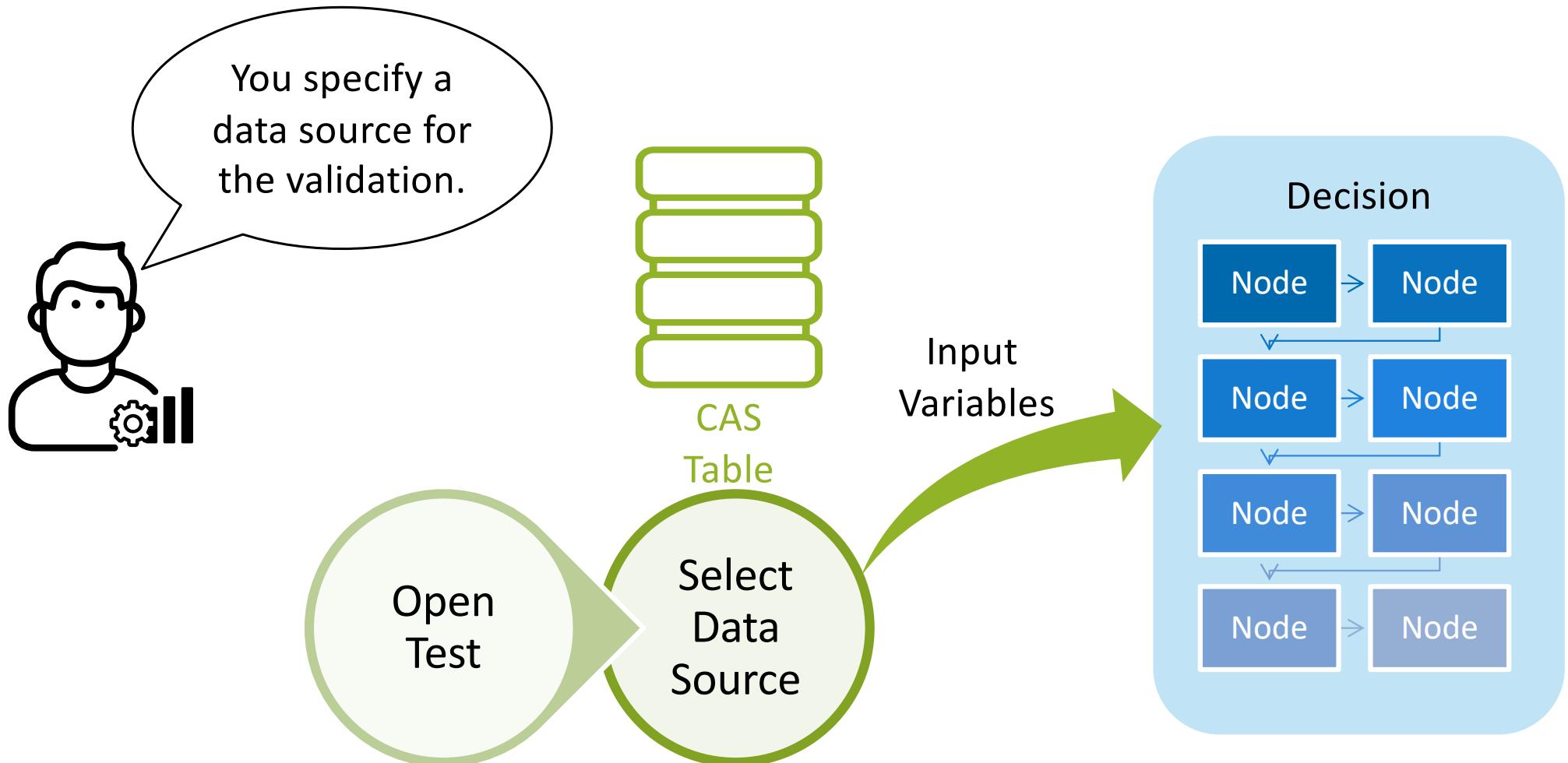


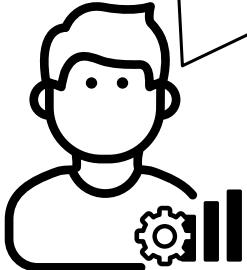
SAS Intelligent Decisioning
automatically creates a
publishing validation test.

Publish

Validation
test defined

Published name
with timestamp

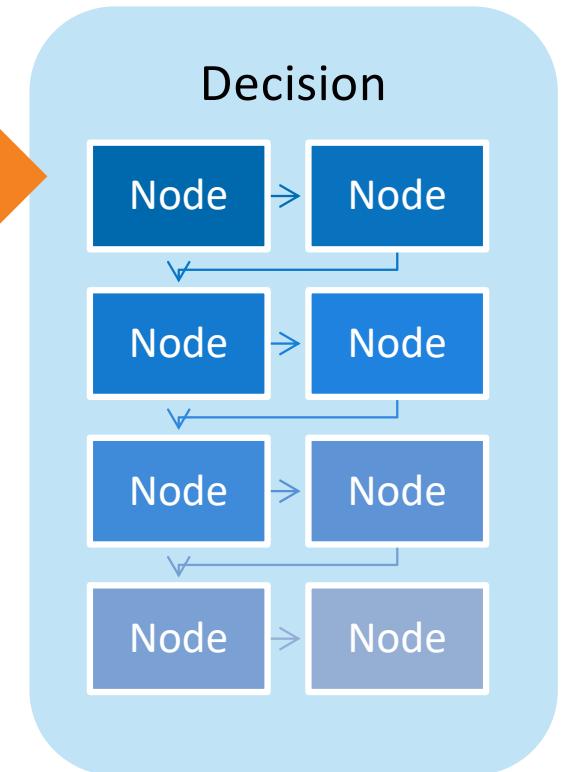




There is no provision for variable mapping after publishing.

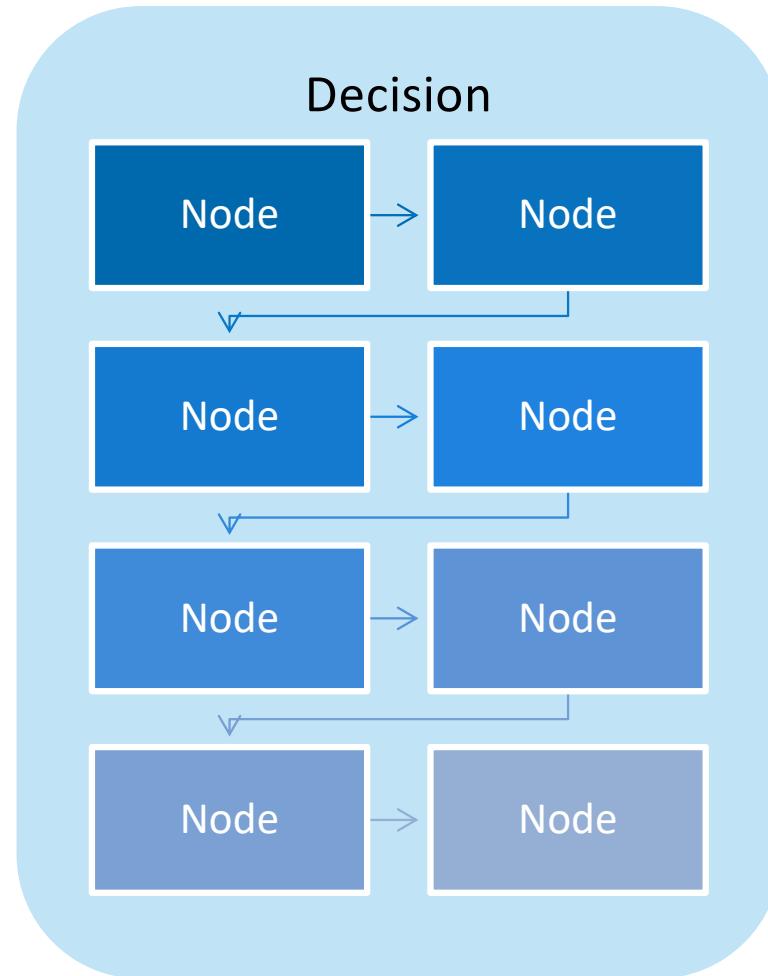
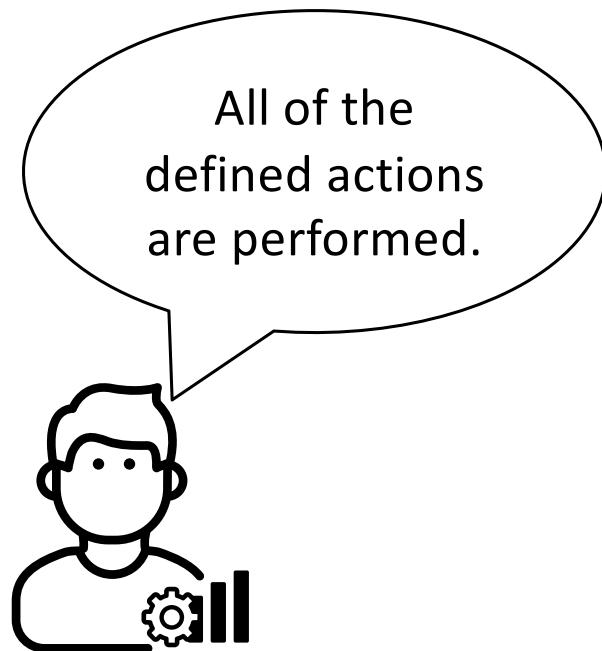


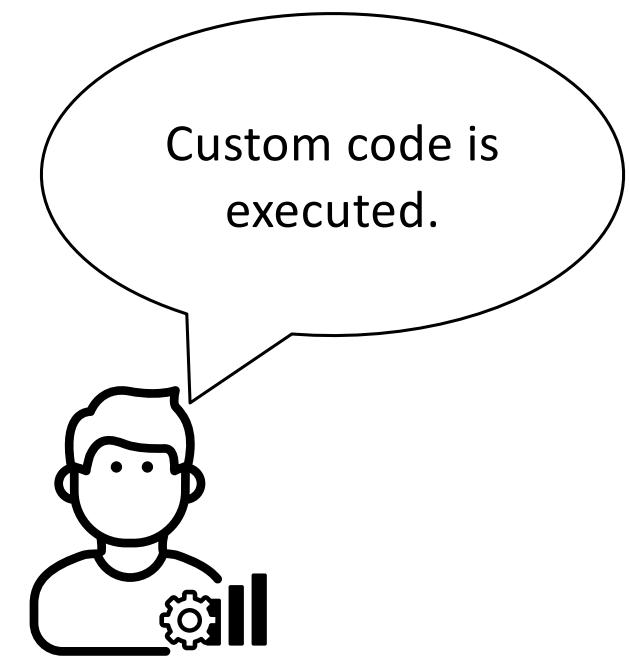
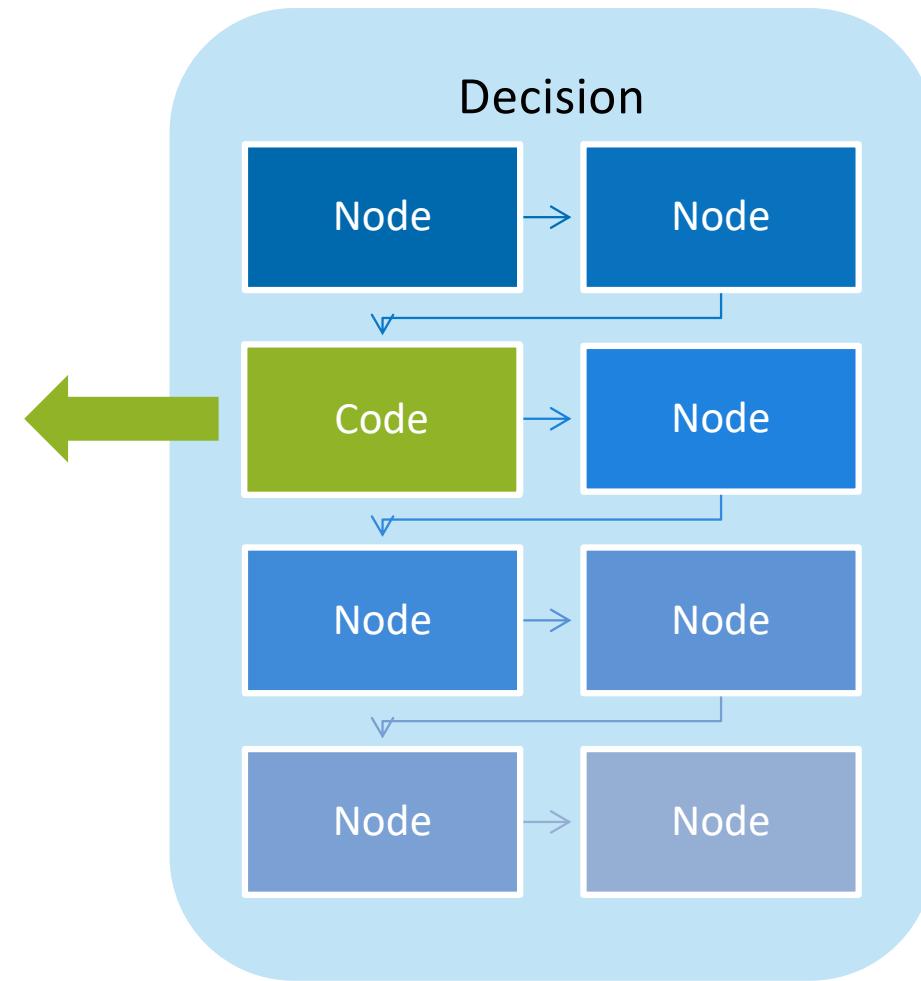
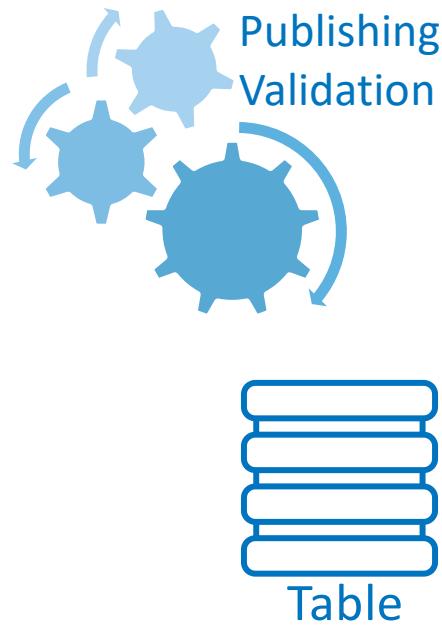
Must have matching variables

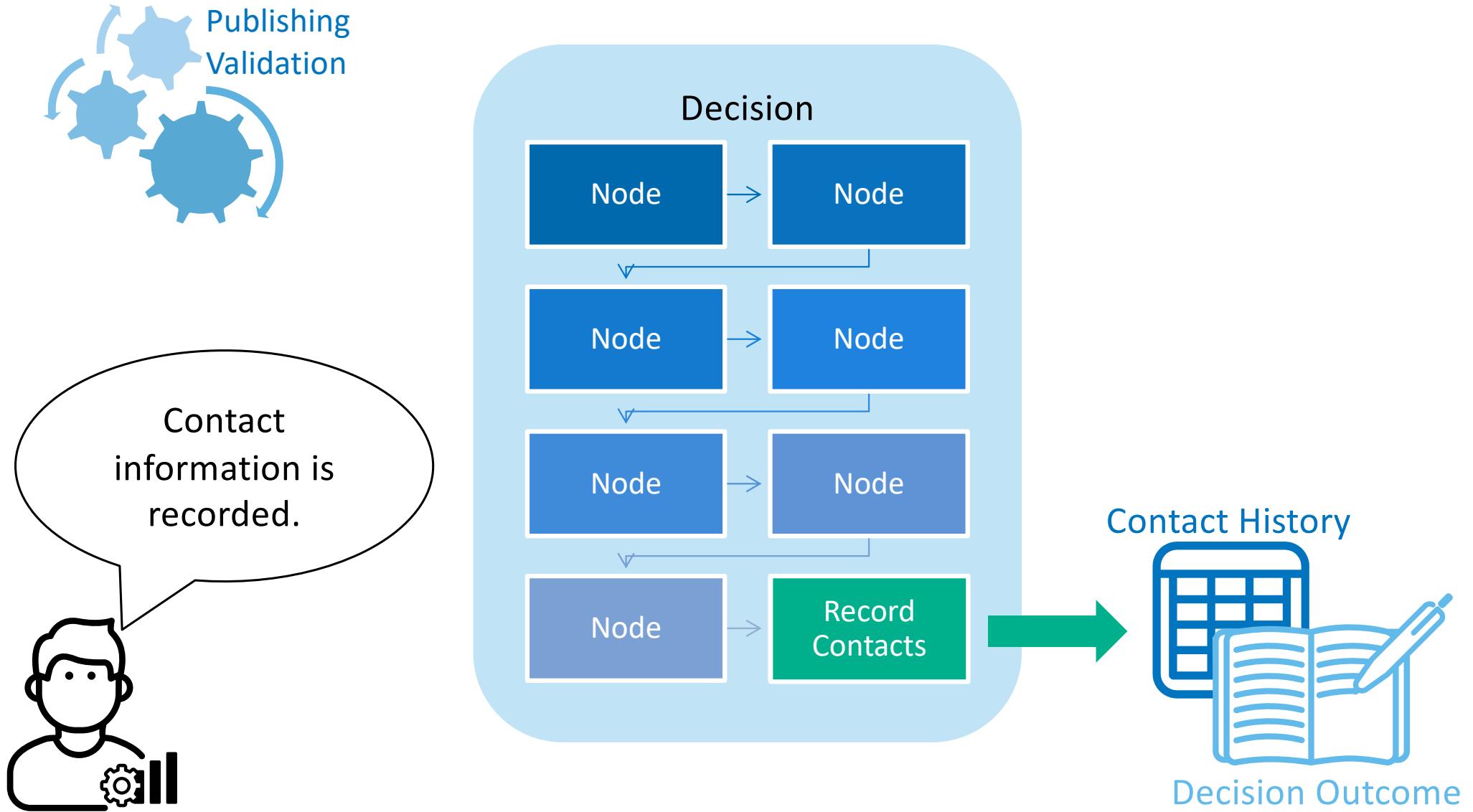
An orange arrow points from the "CAS Table" icon towards the "Decision" block.



Publishing
Validation







SAS Micro Analytic Service



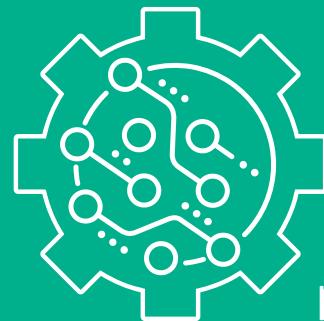
HTTP call



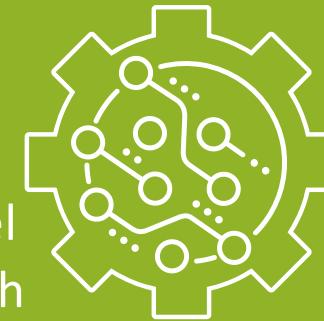
CAS

Publishing
Validation

Teradata



Model
publish
actions



Model
publish
actions

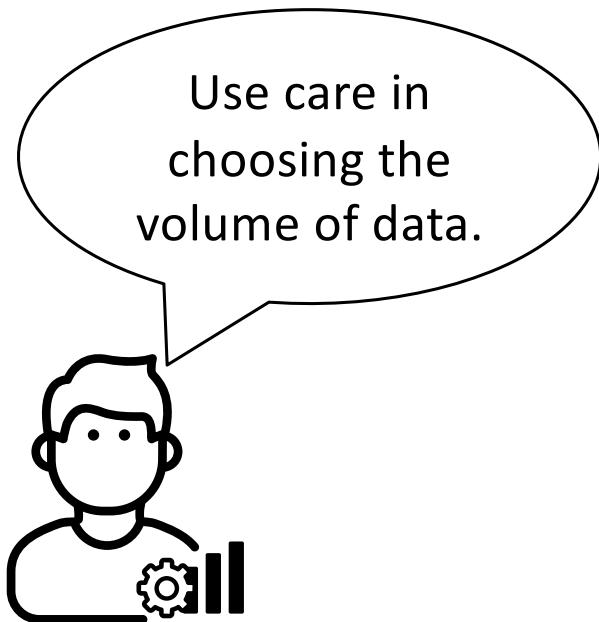
Hadoop



Publishing
Validation

SAS Micro Analytic Service

Row of data → REST request



Data volume



3) Publishing and Validating a Published Decision

This demonstration illustrates publishing and validating a published decision.