Print

Essential Functions of SAS® Intelligent Decisioning

Demo Steps

Course code EBDID0D, prepared date: Tue, Feb 22 2022

Essential Functions of SAS® Intelligent Decisioning
Lesson 01, Section 2 Demo: Working in SAS Intelligent Decisioning

1. Sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Explore the sections of the user interface.
    a. At the bottom left of the interface, click **Expand navigation bar**.
    b. Notice that **Decisions** is currently selected and that a list of decisions is displayed to the right.
    c. Click **Rule Sets** and notice that a list of rule sets is displayed. Notice that you can also navigate to lookup tables, treatments, treatment groups, code files, and global variables.
4. Create a new decision.
    a. Click **Decisions**.
    b. Click **New Decision.**
    c. For **Name**, enter **My First Decision**.
    d. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
    e. Click **Save**. The decision opens.
    f. Notice the tabs at the top of the interface with decision settings.
    g. Move the mouse pointer over the box with the numeral 1 at the top right of the interface. Notice the tooltip showing that one object is open.
    h. Click **Close** to close the decision.
5. Sort decisions by name.
    a. Click the **Name** column heading in the list of decisions. Notice that an upward arrow appears next to **Name** and the decisions are listed in alphabetical order by name.
    b. Click the **Name** column heading in the list of decisions. Notice that a downward arrow appears next to **Name** and the decisions are listed in reverse alphabetical order by name.
6. Search the list of decisions.
    a. In the **Search name** field at the top left of the list of decisions, enter **first** and click **Start search**.
    b. Notice that only decisions with the text first in the name are displayed.
7. Delete the decision My First Decision.
    a. Enable the check box next to My First Decision.
    b. At the top right of the interface, click **Actions** and notice the available actions.
    c. Click **Delete**.
    d. Click **Delete** to confirm.
    e. Click **Close** to close the Delete Decisions window.
8. Restore the deleted decision.
    a. Click **Actions**, **Manage Folders**.
    b. Click **Recycle Bin**.
    c. Notice that you can empty the Recycle Bin.
    d. Click **My First Decision** and click **Restore**.
    e. Click **Close** to close the Manage Folders window.
    f. Notice that the decision has been restored.
9. Access the Help Center.
    a. Click the **L** in the circle at the top right of the interface.
    b. Click **Help Center**. The SAS Intelligent Decisioning User's Guide opens in a new browser tab.
10. Click the SAS Intelligent Decisioning browser tab to return to the application.
11. Click **Clear search text** to return to the complete list of decisions.
12. Sign out of SAS Intelligent Decisioning.
    a. Click the **L** in the circle at the top right of the interface.
    b. Click **Sign out**.

Essential Functions of SAS® Intelligent Decisioning
Lesson 01, Section 2 Demo: Loading Tables into CAS

1. Sign in to SAS with the user ID **lynn** and the password **Student1**.
2. Load tables needed for testing into CAS.
    a. At the top left of the interface, click **Show list of applications**, **Manage Data**.
    b. Click **Import**, **Local files**. Notice that you can import a local file or Excel file.
    c. Click **Data Sources**.
    d. Navigate to **cas-shared-default**, **Public**.
    e. Notice that there are several data sources with the extension **.sashdat**.

    Note: A disk file that is in SASHDAT format can easily be loaded into CAS. You can use SAS Studio to create SASHDAT files.

    f. Load the table **hmeqapps.sashdat**.
        - Click **hmeqapps.sashdat**.
        - Notice that column details for the selected table are displayed.
        - At the top right of the interface, click **Load into memory**.
        - Notice that **HMEQAPPS** appears as a loaded table in the list on the left.
    g. Repeat the steps above to load the tables Investments.sashdat, TSAClaims.sashdat, SalesReps.sashdat and VendorList.sashdat into CAS.
    h. Click the **Available** tab and confirm that all five tables are listed.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 1 Demo: Managing and Mapping Variables in a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a decision.
    a. If necessary, click **Decisions**.
    b. Click **New Decision**.
    c. For **Name**, accept the default value.
    d. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
    e. Click **Save**.
4. Add variables.
    a. Click **Variables**.
    b. Click **Import**. Notice that you can import variables from a comma-delimited or JSON file.
    c. Click **Add variable** and notice that you can add variables from a code file, data table, decision, or rule set. Click **Data table**.
    d. Select **HMEQAPPS**.
    e. Hold down the Ctrl key and click **Loan** and **Purpose**.
    f. Click **Add**.
    g. Click **Add**.
    h. Notice that the **Input** and **Output** properties are both enabled by default.
5. View the properties of the rule set named Demo_VariableMapping.
    a. Click **Rule Sets**
    b. Click **Demo_VariableMapping** to open it.
    c. Click **Variables**. Notice that the rule set includes the variables listed below.

Variable	Data Type	Input	Output
LoanAmount	Decimal	✓	✓
Purpose	Character	✓	✓
Queue	Integer		✓

    d. Click **Close** to close the rule set.
6. Add the Demo_VariableMapping rule set to the previously created decision.
    a. Click **Decisions** to return to the decision.
    b. Click **Decision Flow**.
    c. Right-click **Start** and select **Add below**, **Rule set**.
    d. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_VariableMapping**. Click **OK**.
    e. Click **Variables**. Notice that the variables **LoanAmount** and **Queue** have been added to the decision.
    f. For **LoanAmount**, disable the **Input** property.
7. Map decision variables.
    a. Click **Decision Flow**.
    b. If necessary, click the **Demo_VariableMapping** node.
    c. Click **Input Variables**.
    d. Notice that the rule set input variables **Purpose** and **LoanAmount** are automatically mapped to decision flow variables of the same name.
    e. In the **Maps To** column for the input variable **LoanAmount**, select **Loan**.
    f. Click **Output Variables**.

g. Notice that the rule set input variables **Purpose** and **Queue** are mapped to decision flow variables of the same name and that **LoanAmount** maps to **Loan** as specified previously.

8. Save the decision.
   a. Click **Save**.
   b. Click **Yes** in the window to confirm removal of the **LoanAmount** variable from the decision. This variable from the rule set maps to **Loan** in the decision and is not needed as a decision variable.
9. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 2 Demo: Creating a Rule Set

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a rule set.
   a. Click **Rule sets**.
   b. Click **New Rule Set**.
   c. For **Name**, enter **Demo_LoanApplication**.
   d. For **Type**, confirm that **Assignment** is selected.
   e. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
   f. Click **Save**.
4. Add variables.
   a. Add input variables **Delinq** and **Derog** from the decision **Demo_AddRuleSet**.
      - If necessary, click **Variables**.
      - Click **Add variable**, **Decision**.
      - Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_AddRuleSet**. Click **OK**.
      - Press the Ctrl key and click **Delinq** and **Derog**.
      - Click **Add**.
      - Click **Add**.
      - For both variables, enable the **Input** property.
   b. Add a custom character output variable named **AppStatus**.
      - Click **Add variable**, **Custom variable**.
      - For **Name**, enter **AppStatus**.
      - For **Data type**, confirm that **Character** is selected.
      - Click **Add**.
      - Disable the **Input** property.
      - For **Length**, enter **6**.
      - Click **OK**.
   c. The variable properties should match those shown below.

Variable	Data Type	Input	Output
AppStatus	Character		✓
Delinq	Integer	✓	✓
Derog	Integer	✓	✓

5. Define a rule to set the variable **AppStatus** to *REJECT* if the variables **Delinq** or **Derog** exceed specified limits.
   a. Click **Rule Set**.
   b. Click **Add Rule**. Notice that an IF condition and an ASSIGN action are populated in the rule set editor.
   c. Specify the condition for **Delinq**.
      - For the IF condition, confirm that **Delinq** is selected.
      - For the operator, select **> - (Is greater than)**.
      - For the value, enter **2**.
   d. Add an OR condition for **Derog**.
      - Move the mouse pointer over the condition for **Delinq** and click **Open the expression editor**.
      - Notice that the expression editor includes the following:
        - a Variables tab where you can search for and select variables to add to the expression
        - a Functions tab where you can search for and select functions to add to the expression
        - a variety of operators that you can add to the expression
        - a **Clear** button to clear the expression
        - a **Validate** button to validate the expression
      - Click **OR**.
      - In the list of variables, double-click **Derog**.
      - Click **>** (the greater than operator).
      - Enter **4**.
      - The final expression should be similar to the following:

        Delinq > 2 OR Derog > 4

      - Click **Validate** and make corrections if necessary.
      - Click **Save**.
   e. Assign the value to **AppStatus**.
      - Next to ASSIGN, click **Delinq** and select **AppStatus**.
      - For the value, enter **'REJECT'**.

6. Rename the rule.
    a. Click **Actions**, **Rename rule**.
    b. Enter **Reject high delinquencies or derogatory reports**.
    c. Click **Rename**.
7. Click **Save** to save the rule set.
8. Perform scenario testing.
    a. Click **Scoring**.
    b. Click **Scenarios**.
    c. View the settings for scenario test.
        1. Click **New Test**
        2. Notice that you must enter values for input variables and that you can specify expected values for output variables.
        3. Click **Close**.
    d. Import scenarios.
        - Click **Import Scenarios**.
        - Click in the **Import from** field, navigate to **D:\Workshop\ebdid\Demonstrations** and select **LoanAppScenarioInput.csv**. Click **Open**.
        - For **Encoding**, confirm that **UTF-8 (Default)** is selected.
        - For **Scenario name prefix**, enter **LoanApp**.
        - For **Folder Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
        - Click **Output data library**. Click **cas-shared-default**, **Public**, **OK**.
        - Click **Import**.
        - Click **Close** in the Import Scenarios window.
        - Notice that three scenarios have been imported.
    e. Run the first scenario.
        - Click the first scenario to open it.
        - Notice that the input variables **Delinq** and **Derog** both have a value of *0*.
        - In this case, the rule should not fire and **AppStatus** should have a null value. Notice that the expected value for **AppStatus** is null.
        - Click **Run**.
        - When the test completes, click **Results**.
        - Notice that the expected and actual values for **AppStatus** match.
    f. Run the second scenario.
        - Click the second scenario to open it.
        - Notice that the input variable **Derog** has a value of *5*.
        - In this case, the rule should fire and **AppStatus** should have the value *REJECT*. Notice that the expected value for **AppStatus** is *REJECT*.
        - Click **Run**.
        - When the test completes, click **Results**.
        - Notice that the expected and actual values for **AppStatus** match.
    g. Modify and run the third scenario.
        - Click the third scenario to open it.
        - Notice that the input variable **Delinq** has a value of *5*.
        - In this case, the rule should fire and **AppStatus** should have the value *REJECT*. Notice that the expected value for **AppStatus** is *REJECT*.
        - Enter **reject** as the expected output for **AppStatus**.
        - Click **Run**.
        - When the test completes, notice that the Status column indicates that the test completed with warnings.
        - Click **Results**.
        - Click **Show Differences**.
        - Notice that the expected and actual values for **AppStatus** do not match.
        - Click **Close** to close the scenario test.
9. Click **Close** to close the rule set.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 2 Demo: Adding a Rule Set to a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_AddRuleSet.
    a. If necessary, click **Decisions**.
    b. Click **Demo_AddRuleSet**.
4. View the properties of the data query named GetHMEQdata. This query returns information about loan applicants based on an account number.
    a. On the Decision Flow tab, click the data query node named **GetHMEQdata**.
    b. Click **Input variables**.
    c. Notice that the data query accepts an input variable named **Account**.
    d. Click **Output variables**.
    e. Notice that the data query returns output variables including **Delinq** and **Derog**.
5. Add the Demo_LoanApplication rule set created in the last demonstration to follow the GetHMEQdata query.
    a. Click **GetHMEQdata** and select **Add**, **Rule set**.
    b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_LoanApplication**, **OK**.

c. Click **Input Variables** and notice that the input variables for the rule set were automatically mapped to decision variables with the same name.

6. Test the decision.
   a. Click **Scoring**.
   b. Notice that a test has been previously defined. Click the test name to open it.
   c. View the properties of the input table for the test.
      - Notice that the test uses an input table named HMEQAPPS.
      - Click **Variables**. Notice that the table includes the variable **Account**, which is the input variable to the data query.
      - Click **Cancel** to close the Variable Mappings window.
   d. Click **Run**.
   e. After the test runs, confirm that the Status column displays a green check mark to indicate that the test ran successfully.
7. View the test results.
   a. Click **Results**.
   b. Hide all columns in the output table except for **Rules Fired Count**, **AppStatus**, **Delinq**, and **Derog**.
      - At right of the list of columns in the output table, click **Manage columns**.
      - Click in the list of displayed columns and click **Remove all**.
      - Hold down the Ctrl key and select **AppStatus**, **Delinq**, **Derog**, and **Rules Fired Count**.
      - Click **Add**.
      - Click **OK**.
   c. Sort the results by **Delinq** and **Derog**.
      - Click the **Delinq** column heading twice to sort in descending order.
      - Hold down the Ctrl key and click **Derog** twice to sort in descending order while maintaining the sort for **Delinq**.
   d. Notice that **AppStatus** has a value of *REJECT* for the first three rows, where **Delinq** is greater than *2* as specified in the rule. Notice also that for these three rows **Rules Fired Count** has a value of *1*, indicating that a rule evaluated as *True*.
   e. Notice that for the remaining rows **AppStatus** does not have a value and **Rules Fired Count** has a value of *0*. Neither **Delinq** nor **Derog** exceeded the limits specified in the rule for any of these rows.
   f. Click **Close** to close the test results.
8. Click **Save** to save the decision.
9. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 2 Demo: Creating a Lookup Table

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a lookup table.
   a. Click **Lookup tables**.
   b. Click **New Lookup Table**.
   c. For **Name**, enter **LoanType**.
   d. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
   e. Click **Save**.
4. View the file D:\Workshop\ebdid\Demonstrations\LoanTypeLookup.csv in Notepad.
   a. Launch File Explorer and navigate to D:\Workshop\ebdid\Demonstrations.
   b. Right-click **LoanTypeLookup.csv** and select **Open with**, **Notepad**.
   c. Notice that the file contains comma-separated key-value pairs without headers.
   d. Close the file.
   e. Close File Explorer.
5. Import key-value pairs from a file.
   a. Click **Import**.
   b. In the Import Lookup Table window, click **Browse**.
   c. Navigate to D:\Workshop\ebdid\Demonstration and select **LoanTypeLookup.csv**. Click **Open**.
   d. For **Encoding**, confirm that **UTF-8 (Default)** is selected.
   e. Click **Import**.
6. Activate and close the lookup table.
   a. Click **Activate**.
   b. Click **Yes**.
   c. Click **Close** to close the lookup table.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 2 Demo: Using Lookup Table Functions in a Rule Set

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Modify the Demo_LoanApplication rule set.
   a. Click **Rule sets**.

b. Click **Demo_LoanApplication** to open the rule set.

4. Create a new version of the rule set.
   a. Click **Versions**.
   b. Click **New Version**.
   c. For **Type**, maintain the default selection of **Minor**.
   d. In the **Notes** field, enter **Add purpose rule**.
   e. Click **Save**.
5. Add variables.
   a. Add the variable **Purpose** from the decision **Demo_AddRuleSet** and set it as an input variable.
      - Click **Variables**.
      - Click **Add variable**, **Decision**.
      - Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_AddRuleSet**. Click **OK**.
      - Click **Purpose**.
      - Click **Add**.
      - Click **Add**.
      - If necessary, enable the **Input** property.
   b. Add a custom character output variable named **PurposeDescription**.
      - Click **Add variable**, **Custom variable**.
      - For **Name**, enter **PurposeDescription**.
      - For **Data type**, confirm that **Character** is selected.
      - Click **Add**.
      - Disable the **Input** property.
      - For **Length**, enter **18**.
      - Click **OK**.
   c. The newly added variable properties should match those shown below.

Variable	Data Type	Input	Output
Purpose	Character	✓	✓
PurposeDescription	Character		✓

6. Define a rule to assign a value to **PurposeDescription** from the **LoanType** lookup table if **Purpose** is found in the table.
   a. Click **Rule Set**.
   b. Click **Add Rule**. Notice that an IF condition and an ASSIGN action are populated in the rule set editor.
   c. Rename the rule.
      - Click **Actions**, **Rename rule**.
      - Enter **Assign purpose description**.
   d. Specify the IF condition.
      - Next to IF, select **Purpose**.
      - For the operator, select **LOOKUP**.
      - Click **Select a lookup table**. Navigate to **My Folder**, **Decisioning**, **Demonstrations**, and select **LoanType**.
   e. Specify the THEN condition.
      - Next to THEN, select **LOOKUPVALUE**.
      - For the variable, select **PurposeDescription**.
      - Click **Select a lookup table**. Navigate to **My Folder**, **Decisioning**, **Demonstrations**, and select **LoanType**.

        If you did not create and activate the **LoanType** lookup table in a previous demonstration, use the table named **LoanTypeSolution**.

      - For the lookup key, select **Purpose**.
7. Add an ELSE rule to assign the value *REJECT* to **AppStatus** if not Purpose is not found in the table.
   a. Click **Add rule**.
   b. Rename the rule.
      - Click **Actions**, **Rename rule**.
      - Enter **Reject invalid purpose values**.
   c. Click **IF** and select **ELSE**.
   d. Next to ELSE, click **Delete the selected condition**.
   e. Next to ASSIGN, select **AppStatus**.
   f. For the value, enter **'REJECT'**.
8. Notice that there are three rules that could evaluate to true in the rule set:
   a. The IF rule named **Reject high delinquencies or derogatory reports.**
   b. The IF rule named **Assign purpose description**.
   c. The ELSE rule named **Reject invalid purpose codes**.
9. Save and close the rule set.
   a. Click **Save.**
   b. Click **Close**.
10. Update the Demo_AddRuleSet decision to use the new version of the rule set.
    a. Click **Decisions**.
    b. Click **Demo_AddRuleSet**.
    c. On the Decision Flow tab, click the **Demo_LoanApplication** node.
    d. In the Properties pane, select **1.1** for **Version**.
    e. Click **Save** to save the decision.
11. Test the decision.
    a. Click **Scoring**.
    b. Click the name for the existing test to open it.
    c. Click **Run**.
    d. After the test runs, confirm that the Status column displays a green check to indicate that the test ran successfully.
12. View the test results.
    a. Click **Results**.

b. Run rule-fired analysis.
   - Click **Rule-Fired Analysis.**
   - Click **Run Rule-Fired Analysis**.

c. When the rule-fired analysis processing is complete, hide all columns for the output records except for **AppStatus**, **Delinq**, **Derog**, **Purpose**, **PurposeDescription**, and **Rules Fired Count**.
   - To the right of the list of output records, click **Manage columns**.
   - Click in the list of displayed columns and click **Remove all**.
   - Hold down the Ctrl key and select **AppStatus**, **Delinq**, **Derog**, **Purpose**, **PurposeDescription**, and **Rules Fired Count**.
   - Click **Add**.
   - Click **OK**.

d. Move **Rules Fired Count** to the top of the list of columns.
   - Click **Rules Fired Count** in the list of displayed columns.
   - Click **Move to the top**.

e. Click the **AppStatus** column heading twice to sort in descending order. Recall that in the first version of the LoanApplication rule set, **AppStatus** was equal to **REJECT** for three records in the test data. Now that is true for eight records.

f. Click **Rules Fired Count** for a record in the results. Notice that the results list the name, order, and logic for the rules that fired.

g. Click **Close to close the Rules Fired Count window.**

h. Click **Close** to close the test results.

13. Close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 3 Demo: Creating and Using a Global Variable

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a global variable.
   a. Click **Global Variables**.
   b. Click **New Global Variable**.
   c. For **Name**, enter **DelinqLimit**.
   d. For **Description**, enter **Delinquency Limit for automatic rejection**.
   e. For **Data Type**, select **Integer**.
   f. For **Value**, enter **3**.
   g. Click **Save**.
4. Activate the variable.
   a. Click **DelinqLimit** to edit the variable.
   b. Click **Versions**.
   c. Click **Activate**.
   d. Click **Yes** to create a new minor version.
   e. Click **Cancel**.
5. Use the global variable in a rule set.
   a. Click **Rule sets**.
   b. Click **Demo_UseGlobalVariable** to open the rule set.
   c. Add the global variable to the rule set.
      - Click **Variables**.
      - Click **Global Variables**.
      - Click **Select Variables**.
      - Enable the check box for **DelinqLimit**.
      - Click **OK**.
   d. Modify the existing rule to use the global variable instead of a constant.
      - Click **Rule Set**.
      - Move the mouse pointer over the existing IF condition criteria and click **Open the expression editor**.
      - Select the constant value **2** in the condition for **Delinq** and double-click **DelinqLimit**.
      - The final expression should be similar to the following:

        Delinq > DelinqLimit OR Derog > 4

      - Click **Validate** and make corrections if necessary.
      - Click **Save**.
   e. Click **Save** to save the rule set.
   f. Click **Close** to close the rule set.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 4 Demo: Configuring a Branch Node

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_Branch.
    a. If necessary, click **Decisions**.
    b. Click **Demo_Branch**.
4. View the properties of the data query named GetHMEQdata. This query returns information about loan applicants based on an account number.
    a. On the Decision Flow tab, click the data query node named **GetHMEQdata**.
    b. Click **Output variables**.
    c. Notice that the data query returns an output variable named **Job**.
5. Create an Equals branch based on the variable **Job**.
    a. Right-click the **GetHMEQdata** node and select **Add**, **Branch**.
    b. For **Branch type**, select **Equals**.
    c. Click **OK**.
    d. For **Branch variable**, select **Job**.
    e. Define a branch corresponding to **Job** values of *Self* or missing.
        - Click **Add branch path**.
        - In the **Value** field, enter **'Self'**.
        - Click **Add branch path**.
        - In the **Value** field, select **Missing**.
        - Enable the OR check box next to the path labeled 1.
    f. Define a branch corresponding to **Job** values of *Retired*.
        - Click **Add branch path**.
        - In the **Value** field, enter **'Retired'**.
    g. Notice that in addition to the two defined branches, the diagram includes an Other branch by default.
6. Modify branch labels.
    a. If necessary, click the **Branch** node.
    b. In the Properties pane, click **More**.
    c. In the **Alternate Label** field for the first branch, enter **Self-employed or missing**.
    d. Notice that you can change the order of the paths.
    e. Click **Close**.
7. Add the rule set named Demo_BranchSelfMissing to the branch labeled Self-employed or missing.
    a. Right-click the branch labeled **Self-employed or missing** and click **Add**, **Rule set**.
    b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_BranchSelfMissing**, **OK**.
8. Add the rule set named Demo_BranchRetired to the branch labeled 'Retired'.
    a. Right-click the branch labeled **'Retired'** and click **Add**, **Rule set**.
    b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_BranchRetired**, **OK**.
9. Add the rule set named Demo_BranchOther to the branch labeled Other.
    a. Right-click the branch labeled **Other** and click **Add**, **Rule set**.
    b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_BranchOther**, **OK**.
10. Test the decision.
    a. Click **Scoring**.
    b. Notice that a test has been previously defined. Click the test name to open it.
    c. Click **Run**.
    d. After the test runs, confirm that the Status column displays a green check to indicate that the test ran successfully.
11. View the test results.
    a. Click **Results**.
    b. Run decision path tracking.
        - Click **Decision Path Tracking**.
        - Click **Run Path Tracking**.
        - When the analysis completes, notice that five of the records in the input data followed the path to the rule set named Demo_BranchSelfMissing and 36 followed the path to the rule set named Demo_BranchOther. None of the records followed the path to the rule set named Demo_BranchRetired.
    c. Click **Close** to close the test results.
12. Click **Save** to save the decision.
13. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 4 Demo: Configuring a Cross-Branch Link

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_CrossBranchLink.
    a. If necessary, click **Decisions**.
    b. Click **Demo_CrossBranchLink**.
    c. Notice that the decision includes an Equals branch with three paths.
4. Add the rule set named Demo_CrossBranchRuleSet to follow the existing rule set in the first branch.
    a. Right-click **Demo_BranchSelfMissing** and select **Add**, **Rule set**.
    b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_CrossBranchRuleSet**, **OK**.
5. Add a cross-branch link to the newly added rule set from the second branch of the node.
    a. Right-click **Demo_BranchRetired** and select **Add**, **Cross-branch link**.
    b. For **Target node**, select **Demo_CrossBranchRuleSet (1.0)**.

c. Click **OK**.
6. At the top right of the decision flow, click **Show cross-branch link paths**.
7. Click **Save** to save the decision.
8. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 5 Demo: Configuring a Record Contacts Node

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_RecordContacts.
   a. If necessary, click **Decisions**.
   b. Click **Demo_RecordContacts**.
4. Add a Record Contacts node following the rule set named Demo_CrossBranchRuleSet.
   a. Right-click **Demo_CrossBranchRuleSet** and select **Add**, **Record Contacts**.
   b. In the Properties pane, click **Select one or more variables**
   c. Hold down the Ctrl key and click **Account** and **AppStatus**.
   d. Click **Add**.
   e. Click **OK**.
   f. Notice that you can select a variable that corresponds to a channel for the decision.
   g. Notice that you can enable an option to track treatments.
   h. Notice that you can specify options to **Record rule-fired data**, **Record path tracking**, and **Include in contact policy**.
5. Save and close the decision.
   a. Click **Save**.
   b. Click **Close**.

Essential Functions of SAS® Intelligent Decisioning
Lesson 02, Section 6 Demo: Creating and Comparing Rule Set Versions

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the rule set named Demo_Versions.
   a. Click **Rule sets**.
   b. Click **Demo_Versions**.
4. Create a new version of the rule set.
   a. Click **Versions**.
   b. Notice that there are currently two versions: Version 1.0 (which is locked) and Version 2.0 (which is not).
   c. Click **New Version**.
   d. Maintain the default **Type** value of **Minor**.
   e. Notice that you can enter notes.
   f. Click **Save**.
   g. Notice that version 2.1 has been created and version 2.0 is now locked.
5. Compare object contents for versions 1.0 and 2.1.
   a. Click **Actions**, **Compare object contents**.
   b. For the base version, select **1.0**.
   c. Click **Compare**.
6. Interpret the comparison results.
   a. Notice that **Show Differences** is selected.
   b. Notice that both versions contain a rule named **Reject high delinquencies or derogatory reports**, but version 1.0 includes the condition **Derog > 4** while version 2.1 includes the condition **Derog > DerogThreshold**.
   c. Notice that version 2.1 contains a rule named **Reject invalid purpose values** but version 1.0 does not.
   d. Click **Variables**. Notice that version 2.1 includes the variables **DerogThreshold**, **Purpose**, and **PurposeDescription** but version 1.0 does not.
7. Export the comparison results.
   a. Click **Export**.
   b. Disable the option to export data grid metadata and click **Export**.
   c. Click the link for the downloaded PDF to open it.
   d. Scroll through the results, and then close the browser tab containing the PDF.
8. Compare code for versions 1.0 and 2.1.
   a. Click **Actions**, **Compare object contents**.
   b. For the base version, select **1.0**.
   c. Click **Compare**.
   d. Notice that a comparison of the generated code is displayed.
   e. Click **Close**.
9. Click **Actions**. Notice that you can copy a version and upgrade decisions to use a version.
10. Click **Close** to close the rule set.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 1 Demo: Creating a Data Query for Use in a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a new data query using the SQL editor.
    a. Click **Code files**.
    b. Click **New Code File**.
    c. For **Name**, enter **Demo_DataQuery**.
    d. For **Type**, select **Data Query**.
    e. For **Editor**, confirm that **SQL editor** is selected.
    f. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
    g. Click **Save**.
4. Click the **Variables** tab. Notice that by default the query returns the variable **ReturnCode**.
5. Modify the query properties to return scalar variables.
    a. Click the **Properties** tab.
    b. For **Output Type**, select **Scalar**.
6. Define the query code.
    a. Click the **Code** tab. Notice that this tab displays syntax help.
    b. Specify a SELECT clause to return the variables listed below.

Name	Data type	Length
HomeValue	Integer	N/A
CLAge	Decimal	N/A
Job	String	8

    c. Specify a FROM clause to query the table **ebdid.hmeqdata**.
    d. Specify a WHERE clause to return rows matching the column **account**, which is a string variable with length 13.

    The final query should be similar to the following:

    select HomeValue as {:HomeValue:integer}, CLAge as {:CLAge:decimal}, Job as {:Job:string:8} from EBDID.HMEQDATA where Account={?:Account:string:13}

7. Validate the query and make corrections if necessary.
    a. Click **Validate**.
    b. Notice the message indicating the statement must be run. Click **Run Validation**.
    c. Click **Close** to close the window showing the validation results.
    d. If validation was not successful, make corrections and repeat the validation.
8. Synchronize and view the variables.
    a. Click **Sync Variables**. Notice the message indicating that the Variables tab has been updated.
    b. Click the **Variables** tab.
    c. Notice that in addition to **ReturnCode**, the query now includes the input variable **Account** and output variables **HomeValue**, **CLAge**, **Job**, and **RowCount**.
9. Test the query.
    a. Click the **Scoring** tab.
    b. Click **New Test**.
    c. For **Name**, accept the default value.
    d. For **Location**, confirm that **/Users/lynn/My Folder/Decisioning/Demonstrations** is selected.
    e. For **Input table**, click **Select an input table**. Click **HMEQApps** and click **OK**.
    f. Click **Run**.
10. View the test results.
    a. When the test completes, confirm that the **Status** column contains a green check mark indicating that the test ran successfully.
    b. Click **Results**.
    c. Notice that the output table includes the three variables specified in the SELECT clause along with **ReturnCode** and **RowCount**.
    d. Click **Close**.
11. Close the code file.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 1 Demo: Adding a Data Query to a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a new decision.
    a. If necessary, click **Decisions**.
    b. Click **New Decision**.

c. For **Name**, enter **Demo_AddQuery**.
d. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
e. Click **Save**.

4. Add a data query to follow the Start node.
   a. Right-click **Start** and select **Add below**, **Data query**.
   b. Click **My Folder**, **Decisioning**, **Demonstrations**, **Demo_DataQuery**, **OK**.
5. Access the output type settings.
   a. In the Properties pane, click **Select output type**.
   b. Notice that you can change the output type from within a decision.
   c. Click **Cancel**.
6. Open the query in the SQL editor.
   a. In the Properties pane under Code file editor, click **Open**. The code file appears in the SQL editor.
   b. Click **Close** to close the code file.
7. Return to the decision. Save and close the decision.
   a. Click **Decisions** to return to the decision.
   b. Click **Save**.
   c. Click **Close**.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 1 Demo: Adding DS2 Code to a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a decision.
   a. If necessary, click **Decisions**.
   b. Click **New Decision**.
   c. For **Name**, enter **Demo_AddDS2Code**.
   d. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
   e. Click **Save**.
4. Add a new DS2 code file to follow the Start node.
   a. Click the **Decision Flow** tab.
   b. Right-click the Start node and select **Add below**, **DS2 code file**.
   c. Click **New DS2 Code File**.
   d. For **Name**, enter **Demo_DS2Code**.
   e. Next to **Location**, click **Choose a location**. Click **My Folder**, **Decisioning**, **Demonstrations**, **OK**.
   f. Click **Save**.
5. Save the decision.
6. Open the code file and view the code.
   a. If necessary, click the code node in the decision flow.
   b. Click **Open** in the Properties pane.
   c. Notice that the code tab is populated with the starter code shown below.

```
package "${PACKAGE_NAME}" /inline;
method execute();
end;
endpackage;
```

7. Define custom variables and sync to code.
   a. Click the **Variables** tab.
   b. Click **Add variable**, **Custom variable**.
   c. Define a character variable named **String** that is both an input and output variable.
      - For **Name**, enter **String**.
      - For **Data type**, confirm that **Character** is selected.
      - Click **Add**.
      - Enable the **Input** and **Output** properties
   d. Define a decimal variable named **Number** that is both an input and output variable.
      - For **Name**, enter **Number**.
      - For **Data type**, select **Decimal**.
      - Click **Add**.
      - Enable the **Input** and **Output** properties
   e. Click **OK**.
   f. Sync the variables to code.
      - Click **Sync to Code**.
      - Click the **Code** tab and notice that the variables are specified in the METHOD statement.
8. Specify the code shown below on the Code tab. **Note**: You can copy this code from the file Demo_DS2Code.txt in D:\workshop\ebdid\Demonstrations.

```
package "${PACKAGE_NAME}" /inline;
method execute(in_out double InterestRate,
in_out double Investment,
in_out double Years,
in_out double Interest);
```

```
dcl double count;
interest=0;
do count=1 to years;
Interest=Interest+Investment*InterestRate;
end;
end;
endpackage;
```

9. Click **Validate** and make any corrections if the validation fails.
10. Sync the variables and view the variables on the Variables tab.
    a. Click **Sync Variables**.
    b. Click the **Variables** tab.
    c. Notice that the previously defined custom variables no longer appear and the variables **Interest**, **InterestRate**, **Investment**, and **Years** that are listed on the METHOD statement have been defined.
11. Enable the **Input** property for the variables **InterestRate**, **Investment**, and **Years**.
12. Test the code using the Investments table.
    a. Click the **Scoring** tab.
    b. Click **New Test**.
    c. Save the code when prompted.
    d. For the **Input table**, select **Investments**.
    e. Click **Run**.
13. View the test results.
    a. When the test completes, confirm that the **Status** column contains a green check mark indicating that the test ran successfully.
    b. Click **Results**.
    c. Notice that the output table includes the input variables **InterestRate**, **Investment**, and **Years** along with the variable **Interest** that was calculated in the code.
14. Close the test results and the code file.
    a. Click **Close**.
    b. Click **Close**.
15. Save and close the decision.
    a. Click **Decisions** to return to the decision.
    b. Click **Save**.
    c. Click **Close**.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 2 Demo: Adding a Model to a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_AddModel.
    a. If necessary, click **Decisions**.
    b. Click **Demo_AddModel** to open the decision.
    c. Notice that the decision includes a data query named GetHMEQdata.
4. View decision variables.
    a. Click the **Variables** tab.
    b. Click the **Input** column heading twice to sort the variables by the presence of the Input property.
    c. Notice that the decision includes input variables **Account**, **Loan**, and **Purpose**. **Account** is also an output variable, but all other variables are temporary.
5. Add a model to the decision. The model uses logistic regression to predict the probability of defaulting on a loan.
    a. Click the **Decision Flow** tab.
    b. Click the node labeled **GetHMEQData (1.0)** and select **Add**, **Model**.
    c. Navigate to **Public**, **QS_Reg1** and select **QS_Reg1**. Click **OK**.
6. View decision variables.
    a. Click the **Variables** tab.
    b. Click the **Input** column heading twice to sort the variables by the presence of the Input property.
    c. Notice that the decision includes the two additional input variables **Reason** and **Value**. These are input variables to the model but were not previously defined in the decision.
    d. Notice that the decision includes additional output variables such as **EM_Classification** and **EM_EventProbability**. These are outputs from the model.
7. Map the model input variables **Reason** and **Value** to the decision variables **Purpose** and **Loan**, respectively.
    a. Click the **Decision Flow** tab.
    b. If necessary, click the node labeled **QA_Reg1**.
    c. Click **Input Variables**.
    d. Click in the **Maps To** column next to **Reason** and select **Purpose**.
    e. Click in the **Maps To** column next to **Value** and select **Loan**.
8. Save the decision.
    a. Click **Save**.
    b. Click **Yes** in the window to confirm removal of the variables **Reason** and **Value** from the decision. These variables from the model map to variables in the decision and are not needed as decision variables.
9. Test the decision.
    a. Click **Scoring**.

   b. Click the name of the existing test to open it.
   c. Click **Run**.
   d. When the test completes, confirm that the **Status** column indicates that the test completed successfully.
   e. Click **Results**.
   f. Notice the column **EM_PROBABILITY**, which is the predicted probability of default.
   g. Click **Close** to close the test results
10. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 3 Demo: Configuring a Data Query to Return a Data Grid

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the data query named Demo_QueryReturnDataGrid.
   a. Click **Code files**.
   b. Click **Demo_QueryReturnDataGrid**.
4. View query properties.
   a. On the Code tab, notice that the SELECT clause includes the variables **HomeValue**, **CLAge**, and **Job**.
   b. Click the **Properties** tab and notice that **Scalar** is selected for **Output Type**.
   c. Click the **Variables** tab and notice the three variables that were specified on the SELECT clause are listed as output variables.
5. Modify the query properties to return a data grid instead of scalar variables.
   a. Click the **Properties** tab.
   b. For **Output Type**, select **Data grid**.
6. Synchronize and view the variables.
   a. Click the **Code** tab.
   b. Click **Sync Variables**.
   c. Click the **Variables** tab.
   d. Notice that the variables **HomeValue**, **CLAge**, and **Job** no longer appear and the variable **dgo** has been added. The variable **dgo** is a data grid with columns corresponding the three columns that were specified on the SELECT clause.
7. Test the query.
   a. Click the **Scoring** tab.
   b. Click the name of the existing test to open it.
   c. Click **Save** to save the code.
   d. Click **Run**.
8. View the test results.
   a. When the test completes, confirm that the **Status** column contains a green check indicating that the test ran successfully.
   b. Click **Results**.
   c. Notice that the output table includes the data grid along with **ReturnCode** and **RowCount**.
   d. View the properties of the data grid variable.
      - Click in the data grid column for the first row to view the contents of the data grid.
      - On the Data Grid tab, notice that the data grid includes a single row and that the columns correspond to the three variables specified in the SELECT clause.
      - Click the **Formatted** tab. Notice that this tab displays a formatted version of the JSON string.
      - Click the **Plain** tab. Notice that this tab displays an unformatted version of the JSON string.
      - Click **Close** to close the data grid.
   e. Notice that **RowCount** has a value of *1* for each row in the input table. For this test data, only one row matches the query for each row of the input data. Therefore, the output data grids all contain a single row. However, it is possible for a data grid to include multiple rows.
9. Close the code file.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 3 Demo: Using Data Grid Functions

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. View properties of the decision named Demo_DataGridFunctions.
   a. If necessary, click **Decisions**.
   b. Click **Demo_DataGridFunctions** to open the decision.
   c. View the decision flow.
      - On the Decision Flow tab, notice that the flow includes a data query named ProductQuery.
      - Click the **ProductQuery** node and notice that the data query is configured to return a data grid.
   d. View decision variables.
      - Click the **Variables** tab.
      - Click **ProductQuery_out** to view the properties of the output data grid.
      - Notice that the data grid includes the columns **Price**, **Product_Name**, and **Supplier_ID**.

- Click **Cancel**.

4. Create a rule set.
   a. Click **Rule sets**.
   b. Click **New Rule Set**.
   c. For **Name**, enter **Demo_DGF**.
   d. For **Type**, confirm that **Assignment** is selected.
   e. For **Location**, click **Choose a location** and navigate to **My Folder**, **Decisioning**, **Demonstrations**. Click **OK**.
   f. Click **Save**.
5. Define rule set variables.
   a. If necessary, click **Variables**.
   b. Add the ProductQuery data grid from the Demo_DataGridFunctions decision as an input variable.
      - Click **Add variable**, **Decision**.
      - Navigate to **My Folder**, **Decisioning**, **Demonstrations**.
      - Click **Demo_DataGridFunctions**.
      - Click **OK**.
      - Click **ProductQuery_out**.
      - Click **Add**.
      - Click **Add**.
      - Enable the **Input** Property.
   c. Duplicate the ProductQuery_out data grid.
      - Enable the check box next to **ProductQuery_out**.
      - Click **Actions**, **Duplicate**.
      - For **Name**, enter **ProductsPriceGT100**.
      - Click **Duplicate**.
      - Disable the **Input** property.
   d. Add a custom variable to store the average price.
      - Click **Add variable**, **Custom variable**.
      - For **Name**, enter **AveragePrice**.
      - For **Data type**, select **Decimal**.
      - Click **Add**.
      - Disable the **Input** property.
      - Click **OK**.
   e. The variable properties should match those shown below.

Variable	Data Type	Input	Output
AveragePrice	Decimal		✓
ProductQuery_out	Data grid	✓	✓
ProductsPriceGt100	Data grid		✓

6. Define rules.
   a. Click **Rule Set**.
   b. Define an assignment to calculate the average of the **Price** column of the ProductQuery_out data grid.
      - Click **Add Assignment**.
      - Next to ASSIGN, select **AveragePrice**.
      - Click **Open the expression editor**.
      - Click **Functions**.
      - Expand the **Data Grid** folder.
      - Double-click **DATAGRID_MEAN** to add it to the expression.
      - Click **Variables**.
      - Double-click **ProductQuery_out** to add it as the first argument to the function.
      - Select **colName** in the second argument and enter **Price**. The expression should appear as shown below:

        AveragePrice = DATAGRID_MEAN(ProductQuery_out,'Price')

      - Click **Validate** to validate the expression. Make corrections if necessary.
      - Click **Save**.
   c. Define an assignment to create a data grid containing the products with a price greater than 100.
      - Click **Add**, **Add assignment**.
      - Move the mouse pointer over the newly added assignment and click **Open the expression editor**.
      - Click **Clear** to clear the expression.
      - Click **Functions**.
      - Expand the **Data Grid** folder.
      - Double-click **DATAGRID_SUBSETBYVALUE** to add it to the expression.
      - Click **Variables**.
      - Double-click **ProductQuery_out** to add it as the first argument to the function.
      - Select **filterCol** in the next argument and enter **Price**.
      - Select **operator** in the next argument and enter **GT**.
      - Select **'criteria'** in the next argument and enter **100**.
      - Select **tgtGrid** in the final argument. Click **Variables** and double-click **ProductsPriceGT100**. The expression should appear as shown below:

        DATAGRID_SUBSETBYVALUE(ProductQuery_out,'Price','GT',100,ProductsPriceGT100)

      - Click **Validate** to validate the expression. Make corrections if necessary.
      - Click **Save**.
7. Save and close the rule set.
8. Add the rule set to the Demo_DataGridFunctions decision.
   a. Click **Decisions** to return to the Demo_DataGridFunctions decision.

b. Click **Decision Flow**.
c. Right-click the **ProductQuery** node and select **Add**, **Rule set**.
d. Navigate to **My Folder**, **Decisioning**, **Demonstrations**.
e. Click **Demo_DGF**.
f. Click **OK**.
g. Click **Save** to save the decision.

9. Test the decision.
   a. Click **Scoring**.
   b. Click the test named **Demo_DataGridFunctions_Test_1** to open it.
   c. Click **Run**.
   d. When the test completes, confirm that the **Status** column indicates that the test completed successfully.
   e. Click **Results**.
   f. Notice that the **AveragePrice** column is populated with values.
   g. Click **ProductQuery_out** to open the data grid created by the data query for the last row in the output table. Notice that none of the rows have a price greater than 100. Click **Close**.
   h. Click **ProductsPriceGT100** to open the data grid created in the rule set for the last row in the output table. Notice that the data grid contains no rows. Click **Close**.
   i. Click **Close** to close the test results.
10. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 03, Section 3 Demo: Scoring Rows in a Data Grid

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. View properties of the decision named Demo_DataGridScoreRows.
   a. If necessary, click **Decisions**.
   b. Click **Demo_DataGridScoreRows** to open the decision.
   c. View decision variables.
      - Click the **Variables** tab.
      - Notice that the decision includes a decimal variable named **AveragePrice** and a data grid variable named **ProductsPriceGT100**.
   d. Add a column to the **ProductsPriceGT100** data grid to store the discounted price.
      - Click **ProductsPriceGT100**.
      - Notice that it includes the columns **Price**, **Product_Name**, and **Supplier_ID**.
      - Click **Edit**.
      - Confirm that **Add a new column** is selected.
      - Enter the name **DiscountPrice**.
      - Select **Decimal**.
      - Click **Add**.
      - Click **OK** to close the Edit Columns window.
      - Click **OK** to close the Edit Variable window.
4. Create a rule set.
   a. Click **Rule sets**.
   b. Click **New Rule Set**.
   c. For **Name**, enter **Demo_DGS**.
   d. For **Type**, confirm that **Assignment** is selected.
   e. For **Location**, click **Choose a location** and navigate to **My Folder**, **Decisioning**, **Demonstrations**. Click **OK**.
   f. Click **Save**.
5. Define rule set variables.
   a. Click **Variables**.
   b. Add the **AveragePrice** variable from the Demo_DataGridScoreRows decision.
      - Click **Add variable**, **Decision**.
      - Navigate to **My Folder**, **Decisioning**, **Demonstrations**.
      - Click **Demo_DataGridScoreRows**.
      - Click **OK**.
      - Click **AveragePrice**.
      - Click **Add**.
      - Click **Add**.
      - Enable the **Input** Property.
      - Disable the **Output** Property.
   c. Add a custom variable for the price.
      - Click **Add variable**, **Custom variable**.
      - For **Name**, enter **Price**.
      - For **Data type**, select **Decimal**.
      - Click **Add**.
      - Disable the **Output** property.
      - Keep the Add Variables window open.
   d. Add a custom variable for the discounted price.
      - For **Name**, enter **DiscountPrice**.
      - For **Data type**, maintain the selection **Decimal**.
      - Click **Add**.

- Disable the **Input** property.
- Click **OK**.
- The variable properties should match those shown below.

Variable	Data Type	Input	Output
AveragePrice	Decimal	✓	
DiscountPrice	Decimal		✓
Price	Decimal	✓	

6. Define a rule to calculate the discount price.
    a. Click **Rule Set**.
    b. Define the IF rule.
        - Click **Add Rule**.
        - Next to IF, select **AveragePrice** if necessary.
        - For the operator, select **Is greater than**.
        - Enter the value **150**.
        - Next to THEN, confirm that **ASSIGN** is selected.
        - For the variable, select **DiscountPrice**.
        - Enter the expression **Price*0.9**.
    c. Define the ELSE rule.
        - Click **Add**, **ELSE rule**.
        - Next to THEN, confirm that **ASSIGN** is selected.
        - For the variable, select **DiscountPrice**.
        - Enter the expression **Price**.
7. Save and close the rule set.
8. Add the rule set to the Demo_DataGridScoreRows decision.
    a. Click **Decisions** to return to the Demo_DataGridScoreRows decision.
    b. Click **Decision Flow**.
    c. Right-click the **Demo_DGF** node and select **Add**, **Rule set**.
    d. Navigate to **My Folder**, **Decisioning**, **Demonstrations**.
    e. Click **Demo_DGS**.
    f. Click **OK**.
9. Configure the rule set to score rows in a data grid.
    a. Click **Input variables**.
    b. Enable the **Score rows in this data grid** option.
    c. Notice that the data grid **ProductQuery_out** is selected by default. Click the data grid and select **ProductsPriceGT100**.
    d. Click **Yes** to change the data grid.
    e. Configure variable mapping.
        - Under Input Variables, confirm that the input variable **AveragePrice** maps to a scalar variable of the same name.
        - Confirm that the input variable **Price** maps to a data grid column of the same name.
        - Click **Output variables**. Confirm that the output variable **DiscountPrice** maps to a data grid column of the same name.
10. Save the decision. Click **Yes** to confirm the removal of any variables no longer referenced by objects in the decision.
11. Test the decision.
    a. Click **Scoring**.
    b. Click the test named **Demo_DataGridScoreRows_Test_1** to open it.
    c. Click **Run**.
    d. When the test completes, confirm that the **Status** column indicates that the test completed successfully.
    e. Click **Results**.
    f. Recall that the rule set applies a discount for vendors for whom the average price is greater than *150*.
    g. Right-click the **AveragePrice** column and select **Sort**, **sort (descending)**.
    h. Click **ProductsPriceGT100** to open the data grid for the vendor with the highest value of **AveragePrice**. Notice that the discount was applied to the rows of this data grid. Click **Close**.
    i. Click **ProductsPriceGT100** to open the data grid for any vendor with **AveragePrice** less than *150*. Notice that the discount was not applied to the rows of this data grid. Click **Close**.
    j. Click **Close** to close the test results
12. Click **Close** to close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 04, Section 1 Demo: Creating and Locking a Treatment

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a new treatment.
    a. Click **Treatments**.
    b. Click **New Treatment**.
    c. For **Name**, enter **Demo_Treatment**.
    d. For **Location**, click **Choose a location** and navigate to **My Folder**, **Decisioning**, **Demonstrations**. Click **OK**.
    e. Click **Save**.
4. Add all attributes from the treatment named **BasicTreatment**.
    a. Click **Add Attribute**, **Treatment**.

   b. Navigate to **My Folder**, **Decisioning**, **Demonstrations** and select **BasicTreatment**. Click **OK**.
   c. Click **Add all**.
   d. Click **Add**.
   e. Notice that the attributes **Description**, **URL**, and **Value** are added.
5. Modify attribute values.
   a. Click **Description**. In the **Value** field, enter **Descriptive text**. Click **OK**.
   b. Click **URL**. In the **Value** field, enter **http://www.megacorp.com/treatment1.html**. Click **OK**.
   c. Click **Value**. In the **Value** field, enter **200**. Click **OK**.
6. Add custom attributes.
   a. Click **Add Attribute**, **Custom attribute**.
   b. Define a dynamic date attribute.
      - For **Name**, enter **ValidUntil**.
      - For **Data type**, select **Date** and **Dynamic**.
      - Click **Add**.
   c. Define a fixed Boolean attribute.
      - For **Name**, enter **Customizable**.
      - For **Data type**, select **Boolean** and **Fixed**.
      - Maintain the default selection of **True**.
      - Click **Add**.
   d. Click **OK**.
7. Set a treatment effective start date.
   a. Click the **Properties** tab.
   b. Next to **Start date**, click **Select a date and time**.
   c. Select a date one month from the current date.
   d. For time, select **12**, **00**, **00**, **AM**.
   e. Click **OK**.
8. Save the treatment.
9. Create a locked version for use in a treatment group.
   a. Click the **Versions** tab.
   b. Click **New Version**.
   c. Maintain the default selection of **Minor**.
   d. Click **Save**.
   e. Notice that version 1.1 is created and version 1.0 is locked.
10. Close the treatment.

Essential Functions of SAS® Intelligent Decisioning
Lesson 04, Section 1 Demo: Creating a Treatment Eligibility Rule

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a filtering rule set.
   a. Click **Rule sets**.
   b. Click **New Rule Set**.
   c. For **Name**, enter **Demo_EligibilityRule**.
   d. For **Type**, select **Filtering**.
   e. For **Location**, click **Choose a location** and navigate to **My Folder**, **Decisioning**, **Demonstrations**. Click **OK**.
   f. Click **Save**.
4. Add the variable **Loan** from the data table **HMEQApps**.
   a. Click **Add variable**, **Data table**.
   b. If necessary, click **Select data table**, select **HMEQApps** and click **OK**.
   c. Click **Loan**.
   d. Click **Add**.
   e. Click **Add**.
5. Define the treatment rule.
   a. Click the **Rule Set** tab.
   b. Click **Add Rule**.
   c. Notice that the variable **Loan** is populated in the IF condition.
   d. For the operator, select **< - (Is less than)**.
   e. For the value, enter **5000**.
6. Rename the rule.
   a. Click **Actions**, **Rename rule**.
   b. Enter **Loan less than 5000**.
   c. Click **Rename**.
7. Create a locked version of the rule set for use in a treatment.
   a. Click the **Versions** tab.
   b. Click **New Version**.
   c. Maintain the default selection of **Minor**.
   d. Click **Save**.
   e. Notice that version 1.1 is created and version 1.0 is locked.
   f. Click **Close**.
8. Specify that the newly created rule set is the eligibility rule for the treatment named Demo_Treatment.
   a. Click **Treatments**.

b. Click **Demo_Treatment** to open it.
c. Click the **Eligibility Rule Set** tab.
d. Click **Add Rule Set**.
e. Navigate to **My Folder**, **Decisioning**, **Demonstrations** and select **Demo_EligibilityRule**.
f. Click **OK**.

9. Save and close the treatment.

Essential Functions of SAS® Intelligent Decisioning
Lesson 04, Section 1 Demo: Creating a Treatment Group

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Create a new treatment group.
   a. Click **Treatments groups**.
   b. Click **New Treatment Group**.
   c. For **Name**, enter **Demo_TreatmentGroup**.
   d. For **Location**, click **Choose a location** and navigate to **My Folder**, **Decisioning**, **Demonstrations**. Click **OK**.
   e. Click **Save**.
4. Add treatments.
   a. Click **Add Treatments**.
   b. Enable the check boxes for the following treatments:
      - CheckingOffer
      - InsuranceOffer
      - SavingsOffer
   c. Click **OK**.
5. View the properties of the CheckingOffer treatment.
   a. Click **CheckingOffer**.
   b. Click **Versions** and notice that version 1.0 is locked.
   c. Click in the row for version 1.0 and click **Set Version**. Notice that Version 1.0 is now the displayed version.
   d. Click **Attributes**. Notice that the treatment has fixed attributes **Description**, **Priority**, and **URL** and the dynamic attribute **ValidUntil**.
   e. Click **Eligibility Rule Set** and notice that the eligibility rule is named **No current checking account** and includes the condition **If HASCHECKING = 0**.
   f. Click **Close** to close the treatment.
   g. Click **Treatment Groups** to return to the treatment group.

   Note: The remaining two treatments have similar settings.

6. Select version 1.0 of each treatment.
7. Verify attribute settings.
   a. Notice the message that states *Click **Set Attributes** and verify the attribute settings for this treatment group.*
   b. Click **Set Attributes**.
   c. Notice that the CheckingOffer treatment is displayed. Notice that as seen previously the treatment has fixed attributes **Description**, **Priority**, and **URL** and the dynamic date attribute **ValidUntil**.
   d. Notice that you can disable the dynamic property for **ValidUntil** and enter a value.
   e. Click each of the two remaining treatments and notice that they all share the same three attributes.
   f. Click **OK**.

      Important: You must click each treatment and then click **OK** (not **Cancel**) to confirm the dynamic treatment attribute settings. If you do not do so, the dynamic attribute will not appear as an input variable for the treatment group.

8. Click **Attribute Aliases** and notice that you can specify aliases for attributes.
9. View eligibility variables.
   a. Click **Eligibility Variables**.
   b. Notice that there are three eligibility variables. Each applies to one of the three treatments.
10. Save the treatment group.
11. Activate version 1.0 of the treatment group.
    a. Click **Versions**.
    b. Click **New Version**.
    c. Accept the default type of **Minor**. Click **Save**.
    d. Notice that version 1.1 is created and version 1.0 is locked.
    e. Click in the row for version 1.0.
    f. Click **Activate**. Click **Yes** to confirm.
12. Close the treatment group.

Essential Functions of SAS® Intelligent Decisioning
Lesson 04, Section 1 Demo: Adding a Treatment Group to a Decision

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_AddTreatmentGroup.
    a. If necessary, click **Decisions**.
    b. Click **Demo_AddTreatmentGroup**.
    c. Notice that the decision includes a data query, a model, a range branch, and a rule set on the Other branch.
4. View the output variables of the data query GetHMEQdata.
    a. Click **GetHMEQData**.
    b. In the Properties pane, click **Output variables**.
    c. Notice that the query includes the output variables **HasChecking**, **HasInsure**, and **HasChecking**.
5. View the output variables of the rule set ValidUntilDate.
    a. Click **ValidUntilDate**.
    b. In the Properties pane, click **Output variables**.
    c. Notice that the rule set returns the output variables **ValidFor10Days**, **ValidFor30Days**, and **ValidFor60Days**.

    Note: The rule set calculates the dates that are 10, 30, and 60 days from the date that the rule set executes.

6. Add the Demo_TreatmentGroup treatment group to follow the ValidUntilDate rule set.
    a. Right-click **ValidUntilDate** and select **Add**, **Treatment group**.
    b. Navigate to **My Folder**, **Decisioning**, **Demonstrations**. Select **DemoTreatmentGroup** and click **OK**.
7. Configure the input variables for the treatment group.
    a. In the Properties pane, click **Input variables**.
    b. Notice that the treatment group includes the input variables **HasChecking**, **HasInsure**, **HasChecking**, and **ValidUntil**.
        - **HasChecking**, **HasInsure**, and **HasChecking** are inputs to the treatment group because they are used in treatment eligibility rules. SAS Intelligent Decisioning has automatically mapped them to variables of the same name that are outputs of the data query.
        - **ValidUntil** is an input to the treatment group because it is a dynamic treatment attribute.
    c. Click in the **Maps To** column for **ValidUntil** and select **More**, **ValidFor30Days**. Click **OK**.
8. Save the decision. Click **Yes** to confirm removal of the variable **ValidUntil**. This variable has been mapped to another decision variable and is no longer needed at the decision level.
9. View the treatment group output variable.
    a. If necessary, click the node for the treatment group.
    b. In the Properties pane, click **Output Variables**.
    c. Notice the treatment outcome maps to a data grid named **Demo_TreatmentGroup_out**.
    d. Notice that the columns of the data grid include standard treatment properties such as **treatment_definition_name** and treatment attributes such as **ValidUntil**.
10. Close the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 04, Section 2 Demo: Using an Assignment Rule Set for Treatment Arbitration

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the decision named Demo_TreatmentArbitration.
    a. If necessary, click **Decisions**.
    b. Click **Demo_TreatmentArbitration**.
    c. Notice that the decision includes a treatment group, a model, and a rule set at the end of one of the branches. The rule set performs arbitration of the treatments in the group using predictions from the model.
4. View the properties of the treatment group.
    a. Click the treatment group node named **Demo_FinancialOffers**.
    b. View input variable properties.
        - In the Properties pane, click **Input variables**.
        - Notice that **Model_Prediction** is an input variable. It is an input variable because it is a dynamic treatment attribute.
    c. View output variable properties.
        - In the Properties pane, click **Output variables**.
        - Notice that the output data grid is named **Demo_FinancialOffers_out**.
        - Notice that the output data grid includes many columns including **Model_Prediction**, **AnnualCost**, and **ProductCategory**.
5. View the properties of the model.
    a. Click the model node named **FinancialOffersModel**.
    b. View input variable properties.
        - In the Properties pane, click **Input variables**.
        - Notice that the **Score rows in this data grid** option is enabled and that the output data grid from the treatment group is selected.
        - Notice that the model input variables **AnnualCost** and **ProductCategory** are mapped to columns of the same name in the treatment data grid. The model is using these treatment-level variables in the prediction.
        - Notice that the remaining model input variables are mapped to scalar decision variables. The model is using these customer-level variables in the decision.
    c. View output variable properties.
        - In the Properties pane, click **Output variables**.

- Notice that the model output variable **EM_Probability** is mapped to the treatment data grid column named **Model_Prediction**.

6. View the properties of the rule set.
   a. Click the rule set node named **Demo_Arbitration**.
   b. In the Properties pane, click **Open**.
   c. In the rule set, click the **Variables** tab. Notice that the rule set includes an input data grid variable named **Demo_FinancialOffers_out** and an output data grid variable named **BestOffer**.
   d. Click the **Rule Set** tab. Notice that the rule set includes an assignment that uses the DATAGRID_TOPN function to assign the row of the **Demo_Financial_Offers_out** data grid with the highest value of **Model_Prediction** to the **BestOffer** data grid.
   e. Click **Close** to close the rule set.
7. Test the decision.
   a. Click **Decisions** to return to the decision.
   b. Click the **Scoring** tab.
   c. Click the name of the existing test to open it.
   d. Click **Run**.
   e. When the test completes, confirm that the **Status** column indicates that the test completed successfully.
   f. Click **Results**.
   g. Notice that the **Demo_FinancialOffers_out** and **BestOffer** data grids are populated for only some rows in the test data. These are the rows that followed the path in the decision where the treatment group, model, and arbitration rule set were added.
   h. Click in the row for **Demo_FinancialOffers_out** for the first row where it is populated.
   i. Make a note of the offer that has the highest value of **Model_Prediction**. Click **Close**.
   j. Click in the row for **Demo_BestOffer** for the first row where it is populated. Confirm that it includes one row for the same offer.
8. Close the test results and the decision.

Essential Functions of SAS® Intelligent Decisioning
Lesson 05, Section 1 Demo: Basic Testing of a Rule Set

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the rule set named Demo_BasicTesting and view the rules.
   a. Click **Rule Sets**.
   b. Click **Demo_BasicTesting**.
   c. Notice that the rule set includes an IF rule named **European Supplier** that assigns the value *Europe* to **Supplier_Continent** when **Supplier_Country** has a value that corresponds to certain European countries.
   d. Notice that the rule set includes a similar ELSE rule for North America named *North American Supplier*.

      Note: To view the name of the ELSE rule, place your mouse pointer over the rule until a box appears around it. Right-click and select **Rename rule**. View the name and then click **Cancel**.

4. Run the test that has been previously defined.
   a. Click the **Scoring** tab.
   b. Enable the check box next to the existing test and click **Run**.
5. View the test results.
   a. Click **Results**.
   b. In the output table, notice that **Rules Fired Count** has a value of *1* or *0*. The rule did not fire for rows where **Supplier_Country**=*ES* because this value was not specified in either rule.
6. Run Rule-Fired Analysis.
   a. Click **Rule-Fired Analysis**.
   b. Click **Run Rule-Fired Analysis**.
   c. In the analysis results, notice that output records appear only if at least one rule fired.
   d. Click in the **Rules Fired Count** column for the first row.
      - On the Details tab, notice that the rule named **European Supplier** fired and that this rule is numbered as 1.
      - Click the **Analysis Output columns** tab and notice the values.
      - Click **Close**.
   e. Click in the **Rules Fired Count** column for the second row.
      - On the Details tab, notice that the rule named **North American Supplier** fired and that this rule is numbered as 2.
      - Click the **Analysis Output columns** tab and notice the values.
      - Click **Close**.

      Note: The value of the column **Rule Order** listed in the Rule Fired Count window reflects the static ordering in the diagram and not necessarily the order that the rule fired at execution time.

   f. Close the rule fired analysis results and the test results.
7. Close the rule set.

Essential Functions of SAS® Intelligent Decisioning
Lesson 05, Section 1 Demo: Scenario Testing

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the code file named Demo_ScenarioTesting and view the code.
    a. Click **Code files**.
    b. Click **Demo_ScenarioTesting**.
    c. Notice that the code file is a data query that returns the US state and airport name that corresponds to an airport code.
4. Create a scenario test.
    a. Click **Scoring**.
    b. Click **Scenarios**.
    c. Click **New Test**.
    d. Notice that **Output table location** is required and does not have a value. Click **Output data library** and select **cas-shared-default**, **Public**. Click **OK**.
5. Enter input and output values and run the scenario test.
    a. For **Airport_Code**, enter **LAX**.
    b. Enable the **Include** check box for **State** and enter **CA**.
    c. Click **Save**.
    d. Click **Run**.
    e. When the test completes, notice the green check-mark in the **Status** column indicating that the scenario test completed successfully.
    f. Click **Results**.
    g. Notice that **Airport_Name** = *Los Angeles International Airport* and that the actual and expected values for **State** match.
6. Duplicate the scenario test. Add an expected output value for **Airport_Name** and rerun the test.
    a. Enable the check box next to the existing test.
    b. Click **Actions**, **Duplicate**.
    c. Accept the default name for the test and click **Duplicate**.
    d. Click the name of the duplicated test to open it.
    e. Enable the **Include** check box for **Airport_Name** and enter **Los Angeles Intl Airport**.
    f. Click **Save**.
    g. Click **Run**.
    h. When the test completes, notice the warning icon over the green check mark in the **Status** column. This icon indicates that the scenario test completed with warnings.
    i. Click **Results**.
    j. Notice the message *There are some actual output values that do not match the expected output values.*
    k. Click **Show Differences**. Notice that the expected and actual values for **Airport_Name** do not match.
7. Close the test results and the rule set.

Essential Functions of SAS® Intelligent Decisioning
Lesson 05, Section 2 Demo: Publishing and Validating a Published Rule Set

1. If necessary, sign in to SAS with the user ID **lynn** and the password **Student1**.
2. At the top left of the interface, click **Show list of applications**, **Build Decisions**.
3. Open the rule set named Demo_PublishValidate.
    a. Click **Rule sets**.
    b. Click **Demo_PublishValidate**.
    c. Click the **Versions** tab. Notice that the decision has two versions and that Version 1.1 is the displayed version.
4. View publish settings.
    a. Click **Publish**.
    b. Notice that the displayed version is selected to be published.
    c. Click **SAS Micro Analytic Service (maslocal)** and notice that it is the only destination that is defined.
    d. Notice the following:
        - By default, the published name is the name of the rule set with the version appended but is editable.
        - You can enable the **Replace** option to replace a previously published rule set with the same published name.
        - The **Rule-fired Tracking** option is not selectable. It is not available for the SAS Micro Analytic Service.
    e. Click **Cancel**.
5. Publish version 1.0 to the maslocal destination.
    a. On the Versions tab, click the row for version 1.0 and click **Set Version**.
    b. Confirm that version 1.0 is set as the displayed version.
    c. Click **Publish**.
    d. For **Published Name**, edit the value to remove the version number at the end.
    e. Click **Publish**.
    f. When the process completes, confirm that the **Status** column indicates that the rule set was published successfully.
    g. Click **Close**.
6. Attempt to publish again using the same settings.
    a. Click **Publish**.
    b. For **Published Name**, edit the value to remove the version number at the end.
    c. Click **Publish**.
    d. When the process completes, confirm that the **Status** column indicates that an error occurred. You must enable the Replace option if you want to use a value of published name that has been used previously.

e. Click **Close**.

7. Run the publishing validation test using the input table **TSAClaims**.
    a. Click the **Scoring** tab.
    b. Click the **Publishing Validation** tab.
    c. Notice that a publishing validation test has been created. Place your mouse pointer over the test name and notice that it is the name of the rule set with a datetime stamp added.
    d. Click the name of the test to open it.
    e. Under **Input table**, click **Select a table**.
    f. Click **TSAClaims**. Click **OK**.
    g. Click **Run**.
    h. When the test completes, confirm that the status column indicates that the test ran successfully.
8. View the test results.
    a. Click **Results**.
    b. Notice that the publishing validation test creates an output table similar to basic testing results.
    c. Notice that unlike basic testing, there is no option to run rule-fired analysis.
9. Close the test results and the rule set.