Demos - DeepSee Fundamentals

# Demos - Overview Module

1. Installation of thess demos along with a lot of other stuff that students will use in class can be done using the %Installer routine located in the /deepseetraining directory.
   1. Put these files in C:\ deepseetraining or docker shared volume dir
   2. Import Installer.xml into SAMPLES namespace.
      1. From command line:

SAMPLES>Do $System.OBJ.Load("/tmp/ deepseetraining /setup/Installer.xml","c")

* 1. In Samples namespace execute the following from the command line
     1. SAMPLES >Do ##class(Training.Analytics.Installer).Install("/tmp/deepseetraining") -- assumes files to install are in c:\deepsee\setup -- if they are in a different folder provide the path as the first and only argument to the method.

1. Launch the Demonstration Dashboard from the User Portal
   1. Describe 3 types of widgets
      1. Pivot Table widget
      2. Chart widget: graphical view of a table of data.
      3. Meter: displays a single value
   2. Show the various user interactions:
      1. DrillDown
         1. Use double-clicking row headers
         2. Use drilldown button - needed for use on touch screen interfaces where double-clicking might not work the greatest.
         3. Look at the chart. Notice that HoleFoods generates a lot of revenue / transaction and units sold via snack sales. Drill down into the individual products for the Snack category.
      2. Roll backup
      3. Filter
      4. Drillthrough (aka Detail Listing)
         1. Stress that this is drilling through the cube all the way to the source data. SQL query against source. Returns details not aggregations.
      5. Geo Listing WARNING: REQUIRES INTERNET ACCESS!!
         1. Point out it's again a drillthrough query, but this time the query results include latitude and longitude data for each record.
            1. Note that you must filter and use North American records. These are the only records in the HoleFoods sample that have the latitude and longitude data
      6. Show the "Use Form" button. "Active Analytics"
         1. Use the "Columns" selector to select the "No Channel" columns.
         2. View detail listing, select a single record, click show form.
         3. We now have a form that allows us to edit that record. Maybe we can update it to fill in the missing channel. We could imagine another form that would allow us to assign the record for follow up with an appropriate sales representative to get a channel recorded for this record. Stress that this form is not part of DeepSee. The point is to show Active Analytics - driving action based on the data located in the aggregations uncovered by DeepSee.
            1. Note that in Health Insight there are very cool activities based on this.

Detail listing shows a list of patients who are at risk for diabetes, but have not had a follow up in 30 days.

Actions include: view the composite record for the patient, add the patient to a program, etc.

* 1. Show the print buttons on both the table and the chart. Generate pdfs.
     1. Note that Java must be installed correctly on the server machine.
        1. The pdf renderer is Java based.

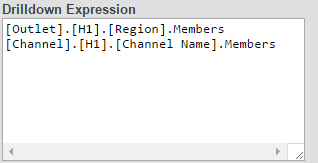
1. Show the real or "right time"
   1. Add some records to the HoleFoods data.
      1. Note that this is adding data to the source.
      2. do ##class(HoleFoods.Utils).AddData(5000,1,1)
   2. Note a few numbers on the dashboard
   3. Synchronize the cube, using the following command.
   4. do ##class(%DeepSee.Utils).%SynchronizeCube("HoleFoods")
   5. Note that this just updated the cube for the changed records. It did not rebuild everything. Users can still query (use dashboards) while this runs.
   6. Might schedule cube synchronizations as tasks. Cube Manager tool covered later makes this easy. Run nightly, or however often makes sense. By default Health Insight does it every night at 8pm. By default TrakCare analytics synchronizes every 5 minutes.
2. Click the refresh button on the dashboard and see all the widgets reflect the added data.
3. Show the different tools:
   1. User Portal
      1. Place to view dashboards. Launch dashboard editor from here.
   2. Analyzer
      1. Tool to create pivot tables / queries to display on dashboards
   3. Architect
      1. Tool to create models for querying.

# Demos - Pivot Table and Analyzer

## Part 1: Analyzer Basics

During these demos we will create the Units and Revenue by Year and Category pivot.

1. Show the following: Measures, Dimensions, Filters, creating and saving a pivot, generated MDX.
   1. Analyzer Layout
      1. Model contents Pane: Most elements defined in cube. Some are calculated and defined in pivot table.
      2. Point out the drop-down list that lets us see Dimensions (model contents), pivots, quality measures, and listings. We will work with Dimensions to start.
      3. Measures
         1. Double click to change the measures.
         2. Typically numerical things that can be aggregated in different ways: Simply adding them up, finding the MAX, AVG, etc.
         3. Determine the number that appears in a cell.
         4. Add some measures. Discuss how number displayed in cube changes.
         5. Currently aggregating for all elements of the cube
      4. Dimensions
         1. Characteristic of data: usually non numeric.
         2. Show some of the dimensions. Point out the dimension, hierarchy, level, member structure. Contrast with measures
   2. Create new pivot table
      1. Name: Training/DeepSee/Demos/Units and Revenue by Year and Category
      2. Category: Training
      3. Begin by putting Categories and Years on the rows and columns, respectively and Units Sold on the Measures. Note that Count was the default.
      4. Add revenue as another measure.
      5. Show the MDX query.
      6. Add YearSold as a filter. Show how it works.
         1. Add another filter: month sold. Show how it works together with year sold.
         2. Add an unrelated filter. Channel
      7. Disable filters before moving on. Show off new checkbox disable.
      8. Remove the Revenue measure.
2. Show the following: Drill Down, Layering, All Level, Detail Listing, Custom Listing.
   1. Drill Down two Types: Double-click and Drag and drop (also the layered look)
      1. Show double-clicking row header. Note drilling down through the level of the hierarchy. Note the parent-child, or containment relationships between higher and lower levels. Will discuss this more thoroughly when we do the modeling.
      2. Show dragging an unrelated level and dropping it on a row header.
      3. Show creating the layered look:
         1. Point out that they need to be bit careful with this.
         2. Temptation to begin creating a "report". Not really the goal of this tool.
         3. Creating potentially costly "CrossJoin" queries: Point out in the MDX viewer. Note the logic of what's happening and how the work grows quickly.
   2. Custom double-click drill down. Rather than drilling down through a hierarchy, we can decide how the drill down should proceed by providing a list of MDX member expressions.
      1. Launch the "Row Options" editor by clicking the little gear icon in the Rows box in Analyzer. Create the following drill-down spec:



1. Put product category on the rows. Double-click a row header. Now rather than drilling down to Product Names, we drill down to Regions, double-clicking again we drill down to country and then that's it.
2. Note that putting none (no quotes) in first line will eventually disable double-click drilling entirely, for the pivot table.
   1. For right now (2015.2) either a space or none followed by enter.
3. Add All Product as the last row of the table
   1. Discuss a single level with one member. Optional
   2. Useful for creating summaries.
   3. Context aware
      1. Show by adding Median Revenue measure to the table. Note that it doesn't just blindly add the cells in the preceding rows, but instead uses their aggregation logic against the cells in the column.
4. Detail listing
   1. Select a cell click binoculars button. What's happening?
      1. Cell represents a certain set of records
      2. Running an SQL query against source data to retrieve a specific set of details for that set of records.
      3. With the table in listing mode, click the show query button and point out the SQL being displayed.
      4. Go to pivot options (wrench) and note the available listing. Select another listing and show it. Point out that available listings (except for custom listing) are defined in cube.
5. Custom listing.
   1. Create one using the drop down list over the dimension tree.
   2. One custom listing per pivot table.
   3. Fields available for custom listing defined in cube definition.
      1. Double-clicking a Category (Row Header) should drill down to regions (Custom drill-down spec).
6. Pivot Options (wrench button)
   1. Note that we've already used this to changes the listing that is shown when binoculars clicked.
   2. Some other options
      1. Max listing rows
      2. Zebra striping
      3. Formatting for row and column headers as well as cells.
      4. Summary rows and columns
         1. Note that we specify here the aggregation that will be used. We kind of hard code it.
            1. Contrast this behavior with using the "All Product" or some other "All Level". In that case the row/column will respect the aggregation logic of the cells that it's aggregating.
7. Row and Column options. Little gear on header or Rows and Columns boxes.
   1. 80/20 split
   2. Median -- easier to show on Column options after using a filter to eliminate one of the regions. Then there are 3. We can see the median for ourselves.
8. Level Options
   1. Click the little gear next to the Product Category Rows
      1. Show some of the options
         1. Formatting the cell values
         2. Cell and Header styles
         3. Filtering
      2. "Show" options top of the box:
         1. Display a single value
         2. Empty rows
         3. MDX expression
            1. Measures.[Units Sold], for example.
9. Chart Options
   1. Note the view button in upper left hand corner:
      1. View Table only (default)
      2. View Cart
      3. View Table and Chart
   2. Click the button that looks like a line chart for the chart options.
      1. Note that there aren't many options. The are only a few chart types available. The intent of this is only to give a quick preview of what a chart for the given table might look like.
      2. When we display charts on the dashboards we'll have many, many more chart options and nearly complete control of the formatting and display.
10. Conditional Formatting (Star Button).
    1. Show the coloring cells proportionately. Select a color. Note that the intensity of the color will vary with the magnitude of the value in the table cells. For example, using blue, the cells with the larger numbers will be more blue, not darker.
       1. Click on the white box to remove the proportional color.
    2. Add a conditional formatting rule.
       1. Click the plus sign.
       2. Create the following rule:
          1. Apply: > 20000
          2. Style: background color green, foreground color blue
       3. Click OK and look at the table.
       4. Click the star burst to open the editor
       5. Show that to edit an existing rule, click the little clipboard on the listing entry for the rule.

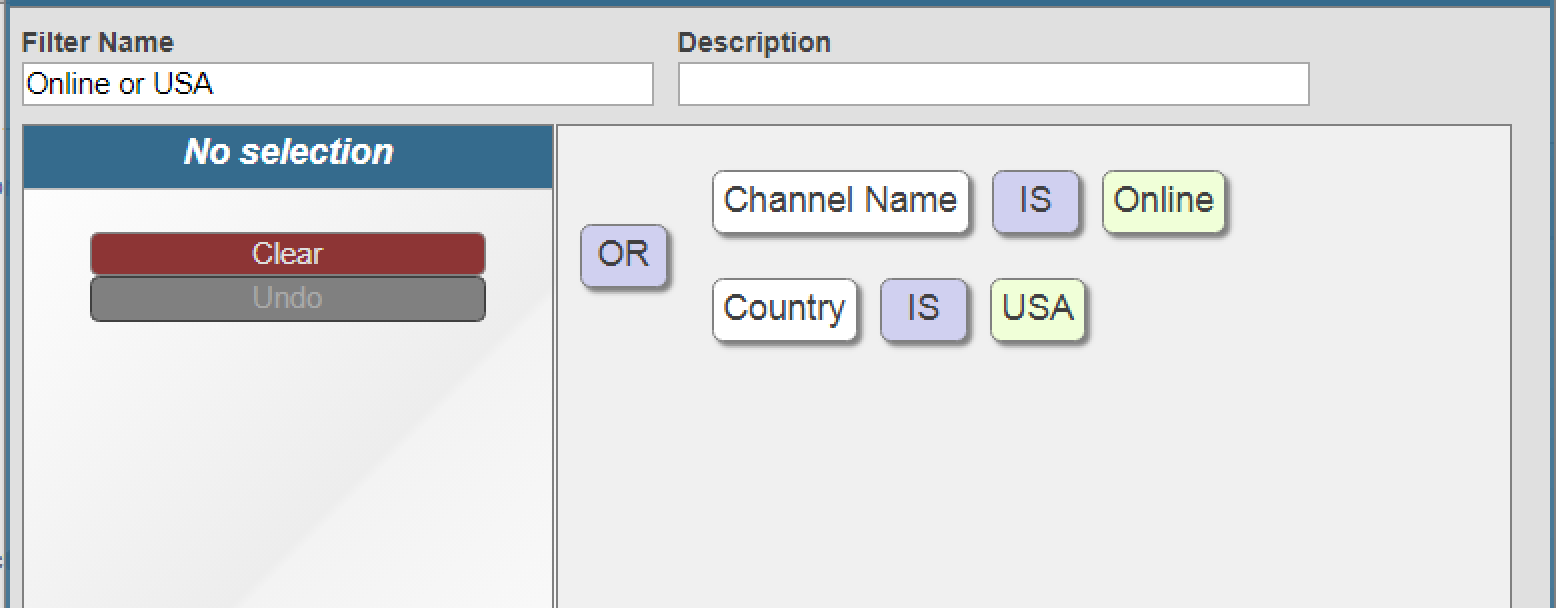
Change the rule to use an icon rather than the cell value

BACK TO SLIDES

## Part 2: Extending the model

Named and Complex Filters

* 1. Create two named filters:
     1. Units Sold > 5 - Searchable measures.
        1. Cannot use measures as simple filters, only complex/named filters and then the measure must be declared as "searchable"
        2. Note that we can also create a dimension and level using UnitsSold and then we could use the members as filters.
     2. Online or USA remember
        1. Set up logical structure first
        2. Add Branch adds new connective (AND, OR)
        3. Add condition adds a new conjunct or disjunct.



* 1. Point out the complex filter. Same idea, but complex filter is added to the pivot and tied directly to it. The named filter is available for any pivot made in this cube.
  2. Pivot options (wrench)
  3. Level options
     1. Rename total row. Shade the whole thing green
  4. Add some conditional formatting
  5. Chart views and chart formatting.

Listing Groups

* 1. Click Tools🡪Listing Group Manager
     1. The first time, a wizard will launch to create a Listing Group.
        1. A Listing Group is a class for storing definitions of detail listing separately from the cube. Analysts who will likely not have ability to modify the cube will now have ability to add new listings using the listing group
     2. Listing Group Name: HoleFoods Demo Listing Group
     3. Listing Group Class Name: Training.DeepSee.Demo.ListingGroup
     4. Listing Group Description: Demonstration listing group for the HoleFoods Cube
     5. Click OK
  2. On the next screen configure the following:
     1. Listing Group Display Name: HoleFoods Demo Listings
     2. Listing Group Target Cubes: HOLEFOODS,HOLEFOODSBUDGET
  3. Click Add Listing Button. Provide the following configuration
     1. Name: Product and Amount
     2. Display Name: Product and Amount
     3. Field List: %ID,AmountOfSale As Amount,Product->Name As Product
     4. Disabled Box: cleared
  4. Compile the Listing Group
  5. Show in Analyzer
     1. Open Analyzer
     2. Display the listings in the Model Contents Pane. Product and Amount should appear.
     3. Put the analyzer in listing mode and select the new listing/
     4. Show that the new listing is also listed under the pivot options (wrench) config options.

Calculated members

* 1. Create a calculated measure:
     1. Units per transaction
     2. Expression: [Measures].[Units Sold]/ [Measures].[%COUNT]
  2. More interesting and maybe more useful calculated member: Last 3 Months - add it to the DateOfSale dimension.
     1. %OR({[DateOfSale].[Actual].[MonthSold].&[NOW-2]:&[NOW]})
        1. %OR provides some optimizations for selecting sets of members that belong to the same level.
        2. Special variable NOW will keep track of the current month.
  3. Another calculated measure using the %MDX function to execute a subquery
     1. Measures.[Amount Sold]/%MDX(“Select measures.[Amount Sold] on 0 from HoleFoods”)

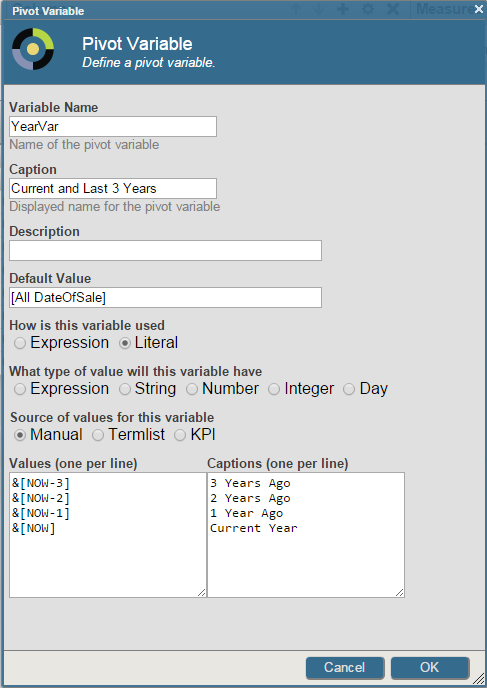
BACK TO SLIDES

Pivot Variables

1. Pivot Variables - Use a pivot variable to create a "filter" with customized filter options. We will be able to use the filter to see this year's data or any of the previous 3 years data.
   1. Create a pivot table (Training/DeepSee/ Demos/PivotVarTest) with Years on the rows and Regions on the columns.
   2. Create a new pivot variable that looks like the following. Click

Machine generated alternative text: 

to launch the editor.

1. 

&[NOW -3]

&[NOW -2]

&[NOW -1]

&[NOW]

3 Years Ago

2 Years Ago

1 Year Ago

Current year

1. Create a named filter to use the new pivot variable. It should look like this

Machine generated alternative text: Named Filter Editor Ç____
Named Filte
1 tenamedflfte
Filter Name Description
Year Filter
Apply
Cancel OK
4

1. Add both the named filter and the pivot variable to the pivot table by dragging them to the "Filters" box. Use the dropdown list (looks like the usual filter control) to swap in different values for the pivot variable. ($variable.yearvar)

Machine generated alternative text: Rows + Q X Columns + Q X j Measures + Q Filters + * X
YearSold O X Reqien O X e’ YearVar X
Curreet YearS Previeee 3
trica S. America
43 18
62 27
D 3YearsAgo 42 17
D 2YearsAgo 74 24
D 1 Year Ago 69 40
Q Current Year 15 3

1. Test the filtering action by changing the selections. Note that selecting "All" on the control causes the query to use the "Default" pivot variable value.

KPI Plugins

Note: This is an advanced topic and you probably won't have time to get too far into it. Right now DeepSee Fundamentals does not include any material on actually building the plugin. Creating the plugins involves lots of coding. There are examples (listed on the slides) and there is good documentation in the Modeling book in the docs.

1. In Analyzer, point out the Median Revenue calculated measure. The measure is actually defined in the cube definition. Note that the little beaker is not blue.
2. Create a pivot table with Product Category on the rows. Use Median Revenue as a measure.
   1. Each cell represents a set of records -- all the candy transactions, for example. Half of those records have revenue below the number in the cell, half are above. Aggregations like this can't be done simply in the way that MAX, MIN, SUM, AVG can be. Those can all use the BitSlice indexes for quick computation. For things like Medians, Percentiles, and Unique counts, we have to actually go through all the records.
3. OPTIONAL
   1. Create the Median Revenue calculated measure again in the pivot table to show how the plugin is used.
      1. First create a pivot table named Training/DeepSee/Demos/Plugin Test. Put Product Category on the rows.
      2. Create a calculated measure
      3. Machine generated alternative text: ç___
         Calculated Member
         Add or edit a calculated member
         Member type
         • Measure Dimension
         Dimension
         MEASURES
         Thrnension for th -
         Member name
         Med Rev
         Name of the ‘:
         Dimension level
         V
         Choose a dimension level if you want to select existing members to populate the MDX
         expression
         Existing members
         Q
         Select any combination of existing dimension members to populate the MDX expression
         Expression
         %KPI (“%p!!e.Median”, “MEDIAN”, 1, “%rneasure”, “Axr.ount
         Sold”, “%CCTTEXT”)
         ‘[! ,FCCCIr C
         Format
         For exarnole
         Cancel
         4
         I
      4. Note the following about the expression:
         1. %KPI is an InterSystems MDX function
         2. %DeepSee.Median is the name of the class that defines the plugin.
         3. "MEDIAN" picks out the right property in the plugin.
         4. 1 picks out the first series defined in the plugin.
         5. %measure, and "Amount Sold" tells the plugin that we're using the amount sold measure.
         6. %Context tells the plugin what pivot table context to factor into its calculations: rows, columns, filters.
      5. Maybe, if it looks like class will not be overwhelmed, open %DeepSee.Plugin.Median in Studio and show the code. Note that this is a special case of a KPI. In a later module we will be using KPIs for other purposes. In short, KPIs are classes that allow us to add a fair bit of customization to DeepSee.

OPTIONAL

Use %DeepSee.Distinct plugin to create a distinct count measure, for example, the number of distinct cities represented by a set of transactions. Use the following MDX expression (%KPI() function) to create the calculated member:

%KPI(“%DeepSee.Distinct”,”DISTINCT”,,”%level”,”[Outlet].[H1].[City]”,”%CONTEXT”)

# Demos - Dashboards

## Tables and Charts

1. **Create a new data source (Pivot) for use in these demos using Analyzer**
   1. Name: Training/DeepSee/Demos/Units by Regions and Years
      1. Rows: Years
      2. Cols: Regions
      3. Measure: Units Sold
2. **Create a new dashboard using User portal**
   1. Discuss create options
      1. Menu --> New Dashboard
      2. Click + and then Add dashboard
      3. Click the > and then then Add Dashboard.
         1. Folder: Training/DeepSee/Demos
         2. Name: Pivot and Chart
         3. Dashboard Title: Pivot and Chart
         4. Public
   2. Discuss various Dashboard Settings:
      1. Page Settings --> Company Name and Logo
      2. Category: Category for the folders view of the user portal.
         1. List view organizes dashboards according to their folders.
      3. Locked: Prevent Updating widgets?
      4. Resize / Modify - should users be allowed to resize widgets or modify (add/remove/configure) widgets.
      5. Scalable Grid layout:
         1. Preferred layout: Each widget will occupy a certain set of grid cells (by default they are 100). Widgets occupy a subset of the cells. When dashboard is resized the cells resize. Number of cells doesn't change.
      6. Other layout options:
         1. Snap into place: layout manager attempts to fill rows and columns on the page.
         2. Position wherever you like.
      7. Work Lists: Boxes on left hand side of dashboard for displaying filters and other controls as well as favorites, recently viewed dashboards, alerts.
3. **Add a table widget to the dashboard:**
   1. Data Source: (Pivot) Training/DeepSee/Demos/Units by Regions and Years -- Pivot table created in step 1 above.
   2. Widget Name: Table1
   3. On the dashboard, expand to a good size and save.
4. **Add Settings to the Widget:**
   1. Print - Add an icon to allow PDF generation
   2. Chart Toggle - Button for toggling between chart and table view of widget. Note that to use this we would first need to configure the chart view.
   3. DrillDown - Arrow buttons for drilldown and undrill. Select a cell and click button and drill to the next level. Has the same effect as double clicking the row header. Using this button better than double-clicking when user is using a touch screen?
   4. Analyzer - Mini Analyzer: analyzer restricted to current pivot. No ability to create new pivots or edit a different pivot. Creates a "local" copy of the data source: restricted to the current user and the current widget.
   5. Excel Export - Self Explanatory.
   6. Row Selection - Can be used only when data source is a KPI. Provides check boxes next to rows that the user can select: one (single) or more than one (multi).
   7. Show Dimensions - icon that shows dimensions that can be used for drilldown via drag and drop. Dimension source is by default the cube used for the current data source.
   8. Dimension Source - An alternative to the default dimension source. (Don't believe this works in 2013.1)
   9. Initial Execute - Enabled by default. Widget query executes when dashboard loads. If this is disabled, must add a refresh button so that the user has some way to trigger the query execution.
5. **Add Controls to the widget:**
   1. Filters:
      1. Filter by Product Category
         1. Location: Widget -- Note that Dashboard displays on a work list on the left hand side. Will see later. Onclick Event is something entirely different. Will see this later.
         2. Target: leave blank for current widget, \* means all the widgets on the dashboard, if there were other widgets there names would appear here.
         3. Action: Apply Filter
         4. Filter: Product Category
         5. Type: Auto (default filter control)
         6. Control Label or icon: Product Category
      2. Show filter working on dashboard. Note that the filter is on the widget
      3. Add a Product Name filter. Exactly like Product Category except that it uses Product Category level.
      4. Show on dashboard how the filters work together.
         1. Select Fruit Category
         2. Product Name filter is itself now filtered to show just Fruit Loops and Pineapples.
   2. Refresh button.
      1. Re-executes query.
      2. Target: same options as above.
   3. Add Listing:
      1. Action: Show Listing
      2. Listing: Listing
      3. Show on dashboard. Click a cell and then click the binoculars. Click the grid button to return to table mode. Note the toggling between the binocular and little grid button.
   4. Add map listing
      1. Action: Show Geo Listing
      2. Geo Listing: Customer Info
      3. Note when demoing, choose data from the North American region. The HoleFoods sample data provides the required latitude and longitude data only for customers in this region.
   5. Add the action to launch a different web page in a new tab. The point of this action is to open an individual transaction in a form for editing. The Active Analytics idea is that through the aggregations provided by DeepSee we can find a record or set of records that we are particularly interested in. We use the detail listing to display the details for the records and then select a record to open in the editing page.
      1. Action: New Window
      2. URL: DemoHFTransactionForm.CSP?OBJID=$$$VALUELIST
         1. Note that the above csp page was loaded if the setup %Installer routine ran successfully.
         2. Name of CSP page is case-sensitive, except for the extension (CSP).
      3. Active When: 1 listing item selected
         1. The particular CSP page only works for a single object id. If a comma separated list of ids is sent to the page, it won't know what to do.
         2. The behavior of $$$VALUELIST when a table is in listing mode is to create a comma separated list of the source ids for each selected row of the listing and send them to the web page as a URL parameter.
   6. Change Data Source
      1. Control allows user to swap in different data sources for the widget.
      2. DataSources are pivot tables listed as values in TermLists
      3. Using the TermList Manager, import the termlist
         1. Management Portal --> DeepSee --> Tools --> Term List Manager
         2. Click New
            1. Name: HF Pivots
            2. Choose File: C:\DeepSee\Demos\HoleFoodsPivotsTL.csv
         3. Back in the dashboard editor
            1. Add the Choose DataSource Control

Note that the wizard does not allow you to select the data source at this point

On widget you should see the drop down list control, unpopulated.

* + - * 1. Back to the dashboard editor.

Edit the Data control

Under DataSourceList, select the HFPivots termlist.

NOTE on b603 can't select Termlist, but can type HFPivots into the right box, at least on Chrome.

Click Save

* + - * 1. Demonstrate on the dashboard. Selecting a pivot swaps in a new datasource for the widget
        2. Note the following related controls:

Set DataSource: Specify one data source. User clicks button and the widget displays that data source.

Choose / Set Column Specs: Instead of swapping out entire data source, changed the member definitions for the column. Utilizar de ejemplo HFRowAndColSpecs.termlist

Choose / Set Row Specs : Instead of swapping out entire data source, change the member definitions for the rows. Utilizar de ejemplo HFRowAndColSpecs.termlist

1. Pivot Variables (New in 2014.1) **(NO FUNCIONA!! , no realizar!!!)**
   1. Go back to Analyzer and create the following Expression Pivot Variable in the Units by Regions and Years pivot
      1. Note the following: It is an Expression pivot variable (can be used directly like a filter), Manual entry of expressions in the editor, there is a bug so that using the member key (with &) does not work in the Default Value, use the member name.
      2. Machine generated alternative text: Pivot Variable
         Pivot Variable
         Define a pivot variable. )
         Variable Name
         urrent
         Nanie of thE . nt
         Caption
         LispiaŸod ra I.
         Description
         Default Value
         [DateOfSale].[AflDateOfSale]
         How is this variable used
         . Expression Literal
         What type of value will this variable have
         • Expression String Number Integer Day
         Source of values for this variable
         . Manual Termlist KPI
         Values (one per line) __________ Captions (one per line)
         [Date0fSale] . [Year5cld) .& Current Year
         [NOW] 0.arrent Month
         [ÐateOf5ale] . [Mcnth5old) .& Today
         [NOW)
         [DateCfSa].eJ . [DayScid) .&
         [NOW)
         /il ¿I
         r Cancel

1. Test in the Analyzer. Use like a filter.
2. Back on the dashboard, add the "Apply Pivot Variable" control. Use the Current variable.

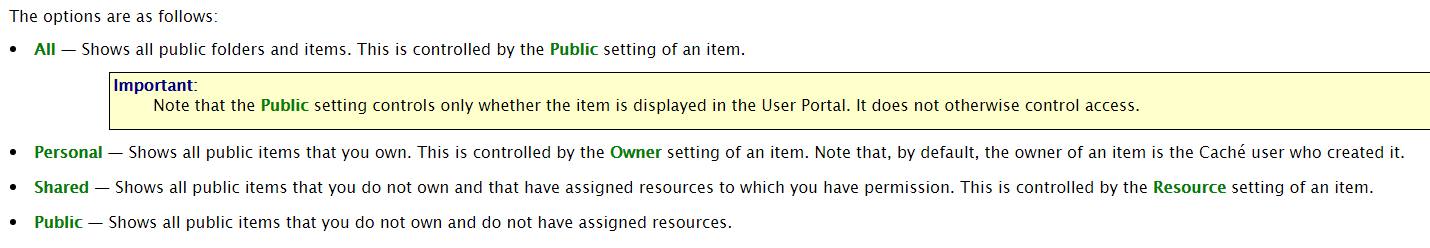
1. **Change Appearance of the Table Widget**.
   1. Under chart and Pivot select Table Settings
      1. Size and Appearance
      2. Colors and Style
         1. Row Style: Row Headers
         2. Column Style: Column Headers
         3. Cell Style: cell style
         4. Stripes: Zebra striping
         5. Stripe Style.
2. **Add a Chart Widget to the dashboard**.
   1. Type: Bar Chart
   2. Data Source: (Pivot) Training/DeepSee/Demos/Units by Regions and Years
   3. Widget Name: Chart1
   4. Note that it will add the chart in the upper left hand corner, directly over the first widget. Drag it to a new spot and click Save or the dashboard will constantly put the chart back on the same spot.
   5. Show some different charts:
      1. Under Chart Settings --> Chart Type, can just point cursor at different charts to see them update without even clicking.
      2. Settle on the stacked bar chart.
      3. Under Chart Settings, also show
         1. Size and Appearance
         2. Titles and Labels
         3. Colors and Style
         4. Series Details (Note that series correspond to columns in the data source).
   6. Note that controls, settings, etc. all apply just as in the table widget.
      1. Add a filter to the chart. Configure it to point at the chart only.
      2. Reconfigure one of the controls on the table (Apply Pivot Variable)
         1. Target is \* -- all widgets.
         2. Location is Dashboard
      3. Show how the control now updates both widgets.
   7. Show how to configure the chart's appearance: Under Chart Settings:
      1. Size and Appearance
      2. Titles and Labels
      3. Colors and Style
      4. X-axis
      5. Y-axis
      6. Series Details:
         1. For use with combo charts. Different data series using different chart types: line, graph, area, etc.
   8. Note that the Chart Legend is configured separately under "Chart Legend"
   9. Show Data Driven Colors
      1. Create a TermList Named Region Colors. Note that any CSS3 color can be used.
      2. Machine generated alternative text: Terms
         Key
         I Asia
         2 Europe
         3 It America
         4 S America
         SeaGreen
         Coral
         rgb(255,O, O)
         Value
         I#4B0082 I
      3. Configure the chart to use the termlist
         1. Chart Settings --> Chart Colors, Data-driven colors (at bottom) (Escribir el nombre de " **Region Colors" porque no lo toma normal**
   10. Show the "On Click Event"
       1. Add the "On Click Event" control to the table widget.
       2. Apply Filter
       3. Target: Chart1
   11. Show how it works. Note that you can clear the onclick filter on both widgets by clicking the row caption (cell in upper left-hand corner) on the table.

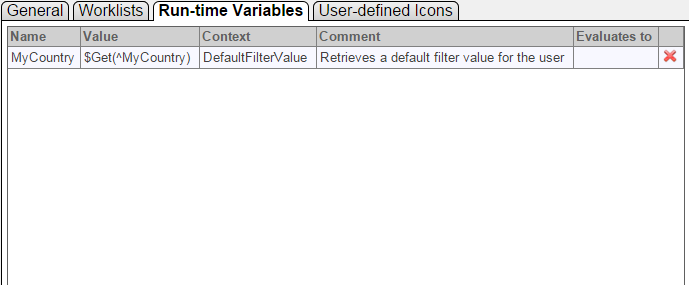
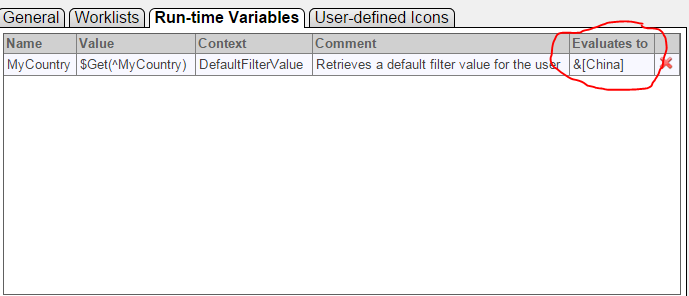
## Scorecards and Meters

1. **In the User Portal tab: Create a new Dashboard**
   1. Training/DeepSee/Demos/Scorecards and Meters
2. **Add a Scorecard Widget**
   1. Regular rather than Big
   2. DataSource: Training/DeepSee/Demos/Count Units and Amount by Product Category Pivot.pivot
   3. Widget Name: Scorecard
   4. Add a Column
      1. Scorecard Settings --> Scorecard --> Columns
      2. Value: Units Sold
         1. Point out that available values are the columns from the data source (above listed pivot table).
      3. Label: Units Sold
      4. Display: Value
         1. Point out that Value is the default value for the Display option.
   5. Add another column to show the row labels from the underlying pivot table
      1. Value: Label
      2. Label: Categories
      3. Might refer to the Analyzer tab to show that the labels are the row labels from the pivot.
   6. Show UnitsSold as %ofTarget - numerically
      1. Display: Value
      2. Show As: %ofTarget
   7. Add additional columns showing the UnitsSold property using the graphical elements:
      1. Arrow
         1. Display: Arrow
         2. Value: Units Sold
         3. MinValue:10000 or some other suitable number. If a value is less than the MinValue the arrow points down, otherwise it points up.
      2. Lamp
         1. Display: Lamp
         2. Value: UnitsSold
         3. MaxValue: 75000 or some other suitable number based on your data. The close the value to the MaxValue the brighter the lamp. Note that the documentation suggests that values below MinValue should be displayable in a different color, red is the default. I haven't seen that work, yet. Note that the default MaxValue is the maximum of the current data set.
      3. PlotBox
         1. Display: PlotBox
         2. Value: UnitsSold
         3. Label: Units Sold
         4. Max Value: 100000
         5. Min Value: blank or 0
         6. Target Value: 50000 -- places a target line in the plot box
         7. Lower: 10000 - lower region
         8. Upper: 90000 - upper region
         9. Base Value: 25000 - values below displayed one way, values above displayed another.
         10. Styles for all this can be set using:
             1. Scorecard Options --> Size and Appearance
             2. Scorecard Options --> Colors and Style
3. **Add a Meter to the Dashboard**.
   1. Add the Widget to the dashboard
      1. Widget Type: speedometer
      2. Data Source: Training/DeepSee/Demos/Total Revenue Units Sold Count for Current Year.pivot
   2. Add a Data Property to the widget
      1. Label: Revenue
      2. Data Value: Revenue
      3. Format: $#,###.## --- For odometer display.
      4. Upper Range: 500000 (Agregar luego en settings del medidor)
      5. Lower Range: 0
      6. Upper Threshold: 450000
      7. Lower Threshold: 50000
   3. Widget can be styled using styles menu.
   4. High and low value colors can be changed using Widget Settings.
4. Add a second meter to the widget
   1. Under Meters click +
      1. Type: textMeter
      2. Data Property: Units Sold
      3. Label: Units Sold
5. If you add a smiley face -- some people will really want to see this
   1. Use Upper Range to control the smile. The closer that the data value is to the Upper Range the bigger the smile. If the number exceeds the Upper Range, you get the "evil" smiley face.
   2. As the data value gets closer to the lower range, the smiley turns more sad faced. When the data value is less than the lower range, the smiley face turns "evil".

## User Portal

1. Portal Overview:
   1. Types of things:
      1. Pivots
      2. Dashboards
      3. Zen Reports (Launch the Zen Report)
      4. URLs
   2. Covers vs List view
      1. Categories
      2. Folders
      3. Customizing Covers
   3. Visibility of items. Summary of options from documentation:



1. Settings: General
   1. Color Scheme
   2. Series scheme
   3. Email
   4. Dashboard
2. WorkLists
3. Runtime Variables: create a runtime variable to use as a default filter value
4. In Admin 🡪 Settings 🡪 Run-time Variables, click New and create the following variable:
   * Name: MyCountry
   * Value: $Get(^MyCountry)
   * Context: DefaultFilterValue
   * Comment: Retrieves a default filter value for the user
5. Click Apply. The Run-time variables tab should look like this:  
   
6. In terminal set a value for ^MyCountry. Value should be one of the countries in the HoleFoods data. The country name must be surrounded by &[ and ]. That is the value must be the member key and not the member value  
     
   SAMPLES>Set ^MyCountry = “&[China]”
7. Refresh the page. Now the Run-time Variables tab looks like the following. Notice the “Evaluates to” column.  
     
   
8. Open the Demonstration Dashboard and edit the Transactions widget. Add a new control with the following configuration:
   * Location: Dashboard
   * Target: \*
   * Action: Apply Filter
   * Filter: Country
   * Type: hidden
   * Default Value:
     + Use a run-time variable: My Country
9. Save the dashboard. Refresh the web page. Only data for China is displayed.
10. In terminal, change the value of the default filter:  
    SAMPLES>Set ^MyCountry = “&[USA]”
11. Return to the dashboard and refresh the web page. The data should now be filtered to show only USA data.
12. Restoring deleted folder items:
    1. In the User Portal, delete the demonstration dashboard.
    2. In Studio, in the SAMPLES namespace, look in the workdspace under the “Other” category. You will see that the Demonstration dashboard is in the $TRASH folder
       1. $TRASH-Training-DeepSee-Demos-Demo.Dashboard.dashboard.DFI
    3. Open the file and change the folderName attribute of the <dashboard> element by removing the “$TRASH/”
    4. Click File 🡪 Save As
       * File name: Training-DeepSee-Demos-Demo Dashboard.dashboard.DFI
    5. Return to the User Portal. Find the restored dashboard under the Training category.
    6. Delete the copy of the dashboard in the $TRASH folder
       1. In the studio workspace, right click on the dashboard in the $TRASH folder and click deleted

# Demos - Modeling and Architect

## Modeling Part 1: Cubes

1. Look at HoleFoods definitions for basic cube members in Architect and compare with Analyzer. Using two tabs, open HoleFoods Sales in Analyzer and Architect:
   1. Start with Measures:
      1. Look at AmountOfSale
         1. Based on source property
         2. Name changed to Revenue for viewing in the cube.
         3. Look at format string
      2. Units Sold
         1. Based directly on source property.
         2. Flagged as searchable - This makes UnitsSold measure usable in a Named Filter.
            1. Additional Index
            2. Uses %Search MDX function
      3. Max Units
         1. Based directly on source property
         2. Aggregation is MAX rather than SUM
      4. Big Sale
         1. Uses Source Expression
         2. $Select - in this case returns 1 if units sold >5 and 0 otherwise
         3. Note the use of %source in the expression. Cannot just put UnitsSold source property into the expression box, must preface with %source.
   2. Look at some dimensions - Note: postpone talking about properties. Save for later.
      1. Channel
         1. Data dimension
         2. One hierarchy, and one level.
         3. Null replacement - No Channel
         4. Based directly on source property
         5. Note All Level
      2. Outlet
         1. Data dimension
         2. One hierarchy and 3 levels
         3. Containment / Parent Child relationships between levels.
      3. Comment that on the data dimension the source properties are associated with levels. The levels have the members that need to be derived from the data
      4. Look at DateOfSale
         1. Time dimension
         2. One hierarchy, 3 levels.
         3. Contrast with data dimension: source class property is associated with the dimension. The levels use "extraction" functions to pull the relevant pieces of the date: month part, year part, quarter part, etc.
         4. Date must be in Caché format: $H, %Date, %Timestamp etc.
         5. If Date not in that format use source expression to put it into that format and then use the extraction functions as normal.
      5. Point out the UnitsperTransaction dimension
         1. Based on the UnitsSold source property.
         2. UnitsSold is used both as a dimension and a measure.
         3. Note that you can then filter based on this numeric dimension. Can' t filter on the measure.
   3. Maybe take a 5 minute or so break
2. Create a cube:HoleFoodsDemoCube
   1. Cube level properties:
      1. Change Count measure caption to # Transactions
      2. Point out null replacement string: applies cube wide unless it's overridden at individual levels.
   2. Create some measures:
      1. Drag and Drop Amount of Sale onto measures
         1. Note aggregation - Sum
         2. Format String $#,###.##
         3. Sacar el tipo de dato
         4. We can also change name to Revenue
      2. Drag and drop Amount of Sales
         1. Change aggregation to AVG
         2. Change name to Avg Sale
      3. Create some dimensions
         1. Channel
            1. All Channel (desde la propiedad de la dimensión)
            2. No Channel for Null Member
            3. Channel Name level
   3. Compile and Build. Show in Analyzer.
      1. Compile and Build.
         1. Discuss what's going on in the build.
         2. For each row in source table, building a fact (row) in the fact table according to these rules.
   4. Add Date Dimension based on DateOfSale
      1. YearSold level - YEAR function
      2. Month Sold level - Month Year function (describe difference with MonthNumber. Group records according to month number crossing years. Can't put this as level 2 in this hierarchy)
      3. Day Sold level - DayMonthYear
      4. Mention that in exercises they will create some date hierarchies using other arrangements and functions.
3. Add Outlet dimension.
   1. Data dimension
   2. Include All Level
   3. Create single hierarchy
      1. Country - Outlet.Country.Name
      2. City - Outlet.
      3. Discuss order
      4. Use drag and drop and Create Member
4. Discuss problem of different members with the same name. Possible that there are two different outlets: one in France, one in US (Texas, Maine?) with the same City name: Paris.
5. Compile and build cube. Look in analyzer. Drop Paris on a row and look at the generated query - &[Paris] – means that the query is using the key not the name. Want to force keys to be unique while allowing names to overlap.
6. In Architect change the source property of the Outlet.City level to be Outlet. This will cause the source property to be the ID. Could also use Outlet.%ID.
7. Compile and Open the cube in Analyzer and check it out.
   * 1. We don't have city names, we have ids
     2. Go back and add City Name as a property
     3. Compile, build and check out in Analyzer
     4. Still not right. Go back and check "use as member names"
8. Note that this is an instance of the more general “Code” and “Description” issue. Not two different levels. Code is the level and Description is the property which may or may not be used as a member name.
9. Go back and add a couple more dimensions
   1. Add in a range expression.
      1. Based on Units Sold source class property.
         1. [0,2.5): Small
         2. [2.5,5):Medium
         3. [5,):Large
      2. Dimension Name - Sale Type
      3. Level Name - Sale Type
      4. All Level - All Sale Types.
10. Back to the HoleFoods Sales Cube
    1. Point out the listings
    2. Point out some of the calculated members
       1. Dimensions
       2. Measures
       3. Mention the KPI Plugin to calculate the median. Uses the %KPI function to call out to the KPI. It gives it the ids associated with the cell, the KPI calculates the median value and returns it
          1. We could do this with some kind of function or expression
          2. The KPI plugin is, however, cube independent.
11. Add a simple listing to HoleFoodsDemo cube.
    1. ID
    2. DateOfSale
    3. City
    4. Channel
    5. AmountSold
12. Add a calculated member to Holefoods Demo Cube
    1. Dimension: Age Ranges
    2. Last 3 Months
    3. %OR({[DateOfSale].[H1].[Month Sold].&[NOW-2]:&[NOW]})
    4. Compile the cube.
    5. Question: Do we need to rebuild?
    6. Answer: No

## Modeling, Part 2: Subject Areas

1. In Architect create a subject area
   1. Name: HoleFoods Demo SA
   2. Display name: HoleFoods Demo SA
   3. Base Cube: HoleFoods Sales
   4. Filter: None
   5. Class name: DeepSeeFundamentals.HoleFoodsDemoSA
2. Compile.
3. Open in Analyzer. Appears identical to HoleFoods Sales cube at this point.
4. Edit and remove some dimensions: After making changes, only need to recompile subject area. No such thing as building a subject area. It gets built when underlying cube is built.
5. Remove the Amount Sold measure. Click Add Element and select
   1. Amount solde – Select Hidden (Details tab) for each.
6. Recompile and then look at Analyzer. Measures removed.
7. Select Product dimension, click on the levels. Again click Add Element to bring them onto the screen.
   1. Change the display name of Product Category to Product Group
   2. Change the display name Product Name to Product
8. Add a Filter using
   1. [OUTLET].[H1].[COUNTRY].&[USA] (ojo si es que no muestra nada, puede ser que no actualice bien!!!!)
9. Recompile and then look at Analyzer again. USA is the only visible country level.
10. Look at DeepSeeFundamentals.HoleFoodsDemoSA in Studio. Note the definitions of the different elements. Note the “filterSpec” defining the above filter.
11. To create a more dynamic filter need to remove hardcoded filterSpec and then override %OnGetFilterSpec – Use the Class  Override wizard.
12. Here’s the code:
    1. Set country = $Get(^MyCountry)
    2. Set pFilterSpec = “[OUTLET].[H1].[COUNTRY].&[“\_country\_”]”
13. Open Terminal. Set ^MyCountry to a country name used in the HoleFoods sample  
    SAMPLES>Set ^MyCountry=”China”
14. Open the subject area in Analyzer and show that only records for China are visible.

## Modeling, Part 3: Build Restrictions, Cube Relationships, Cube Inheritance, Data Connector.

1. Build Restrictions:
   1. Look at Build Restriction on HoleFoods sales cube
   2. Add a build restriction to HoleFoods Demo Cube
      1. Use Actual=1 -- to mimic HoleFoods Sales -- Actual as opposed to Budgeted sales.
      2. Or, use something like DateOfSale > TO\_DATE('31-12-2012','DD-MM-YYYY') to show only this year's sales.
2. Cube Relationships
   1. Necessary code is in NewRelationshipDemo.xml -- code is loaded using the installer routine. Contains the following code:
      1. Training.DeepSee.Customer - source class
      2. Training.DeepSee.Order -- source class
      3. Training.DeepSee.Address - source class
      4. Training.DeepSee.OrderUtils -- utility class (populate and update data)
      5. Training.DeepSee.CustomerCube -- DeepSee cube built using Training.DeepSee.Customer as source class.
      6. Training.DeepSee.OrderCube -- DeepSee cube built using Training.DeepSee.Order as source class. Note that it contains the relationship, but disabled. First part of demo does not use relationship.
   2. Use the Order Cube and Customer Cube: cubes and source classes are imported using the automated setup.
      1. Talk about the source classes: Order contains some details about orders: SaleDate, UnitsSold, Amount whereas Customer contains customer info: Address, Channel (how the customer has bought stuff in the past: retail only, online only, or both retail and online).
      2. Populate the source classes:
         1. Do ##class(Training.DeepSee.OrderUtils).Populate() -- 1000 orders, 500 customers
      3. Open the two cubes in architect -- use two browser tabs. Point out that right now, the Order cube duplicates the dimensions of the Customer cube. Note the relationship is defined in the Order cube, but currently disabled.
         1. Build the two cubes.
         2. Show each in Analyzer.
            1. For each, show Channel dimension on rows.
         3. Leave analyzer open for each cube.
      4. Show how when using unrelated cubes if we update some customer data and synchronize the cubes, we see updates in the customer cube but not the order cube
         1. Write ##class(Training.DeepSee.OrderUtils).UpdateRetailCustomers() -- changes 50% of retail only customers to "Retail & Online".
         2. Synchronize the cubes:
            1. Do ##class(%DeepSee.Utils).%SynchronizeCube("Customer Cube") -- output should show a non-zero number of facts updated
            2. Do ##class(%DeepSee.Utils).%SynchronizeCube("Order Cube") -- output should show zero facts updated.
         3. Look at cubes in Analyzer
            1. In Customer Cube refresh query. About 50% of Retail customers shift to Retail & Online
            2. In Order cube refresh query. No changes. Would expect about 50% of the orders in Retail only to now be associated with Retail & Online customers, if the cube had been synchronized.
      5. Update the Order cube to use the relationship:
         1. Disable the Channel and Address dimensions.
         2. Enable the Customer Cube relationship
         3. Point out the details of the relationship:
            1. Name: could be anything. I use Customer Cube for clarities sake.
            2. Source Property or Source expression: must pick out the appropriate id in the source class of the related cube.
            3. Cardinality: in this case there is one customer for each order.
            4. Inverse: Would use in the event of a two-way relationship. It would be the name of the relationship as defined in the other cube.
            5. Related Cube: Logical name of related cube.
         4. Re-populate the source classes and rebuild both cubes, just to start fresh. Do these steps in the following order:
            1. Do ##class(Training.DeepSee.OrderUtils).Populate -- blow away existing data and repopulate with 500 customers and 1000 orders.
            2. From Architect Rebuild Customer Cube
            3. From Architect Rebuild Order Cube
         5. Open Order Cube in Analyzer
            1. Note the Customer Cube dimensions. Create a pivot showing the Channels on the rows. Works just like before.
         6. Open the Customer Cube in Analyzer
            1. No changes from earlier. This cube is entirely independent.
         7. Update the Retail Customers and then Synchronize the Customers cube
            1. Write ##class(Training.DeepSee.OrderUtils).UpdateSomeRetailCustomers()
            2. Do ##class(%DeepSee.Utils).%SynchronizeCube("Customer Cube")
            3. Refresh the Analyzer showing the Order Cube. Data is updated.

# Demos - Updating Cubes

## DSTIME and ^OBJ.DSTIME

1. As preparation for the demo (and ultimately for the exercise) everyone needs to purge^OBJ.DSTIME
   1. Run the purge command in Samples  
      SAMPLES>Do ##class(%DeepSee.Utils).%PurgeDSTIME()
   2. Synchronized various cubes: HoleFoods, HoleFoodsBudget, HoleFoodsTransactionCube, HoleFoodsDemoCube, ABCCube, Orders Cube, Customers Cube
   3. Now run the purge command again. You might need to find other cubes to synchronize and then purge again until all the entries have a third subscript that says either dstime, or “lastDataUpdate”
2. The “lastDataUpdate” node contains date and time ($h format) of last synchronization. The “cubes” node contains the next batch number to synchronize.
3. Use Zwrite to display ^OBJ.DSTIME it should be empty.
4. Add some data to HoleFoods.Transaction

SAMPLES>Do ##class(HoleFoods.Utils).AddData(5)

1. Use Zwrite to examine ^OBJ.DSTIME again. You should see something like:

^OBJ.DSTIME("HoleFoods.Transaction",0,41)=1

^OBJ.DSTIME("HoleFoods.Transaction",0,42)=1

^OBJ.DSTIME("HoleFoods.Transaction",0,43)=1

^OBJ.DSTIME("HoleFoods.Transaction",0,44)=1

^OBJ.DSTIME("HoleFoods.Transaction",0,45)=1

Note: the third subscript is the ID of the HoleFoods.Transaction record. The specific values will vary according to your particular setup. The value 1 for the global means the change is a new record. A value of 0 indicates an updated record. A value of 2 indicates a deleted record.

1. Now synchronize HoleFoods cube using %SynchronizeCube

SAMPLES>Do ##class(%DeepSee.Utils).%SynchronizeCube(“HoleFoods”)

1. Use Zwrite to examine ^OBJ.DSTIME. It should be empty. %SynchronizeCube deleted the entries and moved them to ^DeepSee.Update
2. Examine ^DeepSee.Update. Should see the entries that were in ^OBJ.DSTIME now in ^DeepSee.Update with only modification to the “batch” number.
3. %Synchronize did the following to ^DeepSee.Update
   1. Incremented the top “counter” node
   2. Updated the “lastDataUpdate” node of the “cubes” entry for “HoleFoods”
   3. Updated the “dstime” update to indicate the next batch number to start for synchronization. Where dstime=N, cube is synchronized through N-1.
4. Next look at the DSTIME parameter of HoleFoods.Transaction. Currently its value is “auto” which means track changes in ^OBJ.DSTIME and ^DeepSee.Update. Change value to “manual” and recompile. Examine how this affects synchronization.
5. Add a record to HoleFoods.Transaction

SAMPLES>Do ##class(HoleFoods.Utils).AddData(1)

1 row(s) created

1. Synchronize. This time it finds no changes. The changes are no longer being logged in ^OBJ.DSTIME

SAMPLES>Do ##class(%DeepSee.Utils).%SynchronizeCube("HoleFoods")

0 fact(s) updated

Elapsed time: .00021s

Note that if you run Synchronize a second time, you will get different output.

SAMPLES>Do ##class(%DeepSee.Utils).%SynchronizeCube("HoleFoods")

No changes detected.

1. Now we can use %ProcessFact to update the id. You will need the id of the new record which should be the next id, N, from the last entry in ^OBJ.DSTIME, if you've been following these demo steps.

SAMPLES>Do ##class(%DeepSee.Utils).%ProcessFact("HoleFoods",N,1)

Building fact table: 1 fact(s) updated

Note that third argument is the verbose flag.

**Cube Manager Demo - Note keep this demo simple.**

1. Demo using Order Cube and Customer Cube. Start with no relationship between Order Cube and Customer Cube. If necessary, undo the relationship from the earlier demo and rebuild each cube.
2. Create a cube group with an update plan.
   1. First time, Cube manager will prompt for creation of a Cube Registry class. Create one (Training.DeepSee.CubeReg).
   2. Drag Group containing Customer Cube and drop on the Registered Groups
   3. Drag Group containing Order Cube and drop on the Registered Groups. This will cause a prompt for "Merging" the first group with the second. Click OK.
   4. Deselect "Exclude" on the Details for the group, Customer Cube, and Order Cube. The GUI rows are no longer greyed out. They are now participating the in the update plans.
   5. At the group level, change the Sync Plan to Synch every 2 minutes.
      1. Click "Apply to All Cubes in Group" button.
      2. Synch Frequency column updates for both cubes.
   6. **Click SAVE**
   7. **IMPORTANT: ALL CUBES IN A GROUP MUST BE BUILT THROUGH THE CUBE MANAGER IN ORDER FOR THE TASK TO START. SO BUILD ALL CUBES IN THE GROUP THROUGH THE MANAGER**
3. View the Log Table for the cube registry.
   1. Look at **%DeepSee\_CubeManager.CubeEvent** in System Explorer --> SQL (SAMPLES namespace).
      1. Select \* From %DeepSee\_CubeManager.CubeEvent
      2. Notice the Register and Update events.
      3. Soon you should start seeing Synch Events for each cube. Order cube goes first. No facts updated, however, Notice the Ncores column, if "Build Synchronously" selected in Cube Manager, then Ncores = 1.
      4. Using Terminal, add some customers to the data  
         SAMPLES>Do ##class(Training.DeepSee.OrderUtils).AddCustomers() -- This adds 5 customers.
      5. Soon there should be another entry in %DeepSee\_CubeManager.CubeEvent showing that 5 facts were updated for the Customer Cube.
   2. Talk a little about some other features of the Cube Manager
      1. Order of the building of the cubes might matter. Suppose we re-instituted the relationship between the Order Cube and the Customer Cube. Probably, we would want to synch the independent cube (Customer Cube) before the dependent cube (Order Cube). Why?
      2. Suppose we do not have a relationship between Order Cube and Customer Cube. Remember that changes to the Customer records are not automatically visible to the Order Cube through ^OBJ.DSTIME
         1. Run the method that changes some online customers to online and retail. This change needs to be reflected in our Order cube. Maybe open Order Cube in Analyzer. The "Channel" dimension shows orders by customers classified as Online only, Retail only, or Online and Retail. Run the following command to convert some of the customers.

SAMPLES>Do ##class(Training.DeepSee.OrderUtils).UpdateRetailCustomers(.5)

Converting 50 percent of Retail customers to Online & Retail.

1. Now, if we look at synch history in SQL, we see that a number of facts were updated for the Customer Cube but none for the Order Cube.
2. One way to fix this is to run some "pre-synch" code in the update plan for the Order Cube which will look for changes in the Customer class and record which order records need to be updated. We have a utility in our Training.DeepSee.OrderUtils class. Enter the following for Pre-Synch for the Order Cube.  
     
   Do ##class(Training.DeepSee.OrderUtils).UpdateDSTime("Order Cube", "Training.DeepSee.Customer")
3. Run the update Retail Customers method again.
4. Look at the synch history again. On next synch we'll see the synch code run and we'll see that there are facts updated for both the Customer Cube and the Order Cube!

# iKnow and DeepSee

Purpose: Demonstrate the basic features of iKnow and the use of iKnow indexing results in DeepSee cubes. Demonstrate how to configure a DeepSee cube

Outcome: iKnow domain created in USER namespace using the movie data. Aviation demo setup in

## iKnow Basics

1. Point out the movie data in c:\DeepSee\iknow\movies
   1. 221 text files. Each includes a movie “plot synopsis” taken from IMDB (SPOILER ALERT)
   2. Open one or more of the text files to show learners what we are working with.
      1. Totally unstructured data: pure text.
      2. English language.
      3. Text format, not pdf.
2. In management portal, switch to USER namespace, and navigate to iKnow Domain Architect (System Explorer 🡪 iKnow 🡪 Domain Architect)
   1. Note that the iKnow tools will be greyed out. Must enable iKnow access in the default web application for the namespace.
   2. Open web application /csp/user. (System Administration 🡪 Security 🡪 Applications 🡪 Web Applications
   3. In /csp/user web application, next to the “Enabled” label, select iKnow. And click Save.
3. Navigate back to iKnow Domain Architect in the USER namespace (System Explorer 🡪 iKnow 🡪 Domain Architect)
4. Create a new iKnow domain for indexing the movies plots
   1. Click New
      1. Domain Name: MoviePlots (note that name can have spaces and special characters, but this will mess up the automatically generated class name)
      2. Class Name: Training.DeepSee.MoviePlotsDomain
      3. Allow Custom Updated: Keep default (not selected). Note: if anyone asks, selecting this allows us to add data or dictionaries to the domain using the command line.
   2. Point out the main areas on the Architect
      1. Top Row: Change domain name, custom updates, disable the domain
      2. Domain settings. Select a language. Add parameters. Documentation contains list of parameters including setting the default sorting (frequency or spread), whether or not we want iKnow to report status information during builds
         1. Set Status parameter to 1
      3. Metadata Fields. Non iKnow data to associate with sources. Use to filter or sort. For example we could use movie ratings as metadata and then filter to see only data from sources with a rating greater than some value that we specify.
      4. Data Locations. Note the different types of sources that we can load data from . We are going to load data from files
         1. Click Add Data from Files
            1. Name: movie plots
            2. Path: C:\DeepSee\iKnow\movies
            3. Extensions: txt
            4. Encoding: UTF-8
            5. Batch Mode: Enabled
      5. Blacklists. We could create lists of terms that we don’t want to appear in our indexing results.
   3. Click Save.
   4. Click Compile
   5. Click Build.
5. After successful build click Tools menu and then launch the knowledge portal.
   1. Contains the indexing results for the entire domain.
6. Explore the various features. Start with top concepts in upper left-hand corner.
   1. Top Concepts: Most frequently occurring in the sources. We have both frequency and spread (number of sources in which the term occurs) information.
      1. The list of the most frequently occurring concepts seems about right. Maybe “one” is a little strange.
      2. Click one and look at the related concepts (3rd table from left on first row) and at the CRC’s bottom row left. This begins to make more sense. We see lots of constructions like "one of (the) men”, “one by one”, “one of (the) attackers”, phrases we’d expect to see frequently.
      3. Notice the CRC “neo is one” – “the” was thrown away as a noise word. This one looks more interesting. Click on drill and we see the source. “Neo is the one”
      4. Note of if “one” is confusing or doesn’t make sense we can black list it.
         1. In the architect create a new black list
            1. Name: Black List #1
            2. Entry: one (note lower case, click enter)
            3. Recompile. No need to rebuild.
            4. Return to knowledge portal and select the blacklist using the black list button. Click apply. May have to refresh web page.   
               “One” no longer appears a “Top Concept”.
      5. Return to the “top concept” box. Click the toggle button below it. Now rather than “Top Concepts” the box displays the dominant concepts in the sources. Semantic dominance is a measure of the importance rather than simple frequency of a concept’s occurrence. The algorithm for calculating dominance (see documentation) within a source takes into account factors like frequency, diversity of entities in a source, diversity of entities in a source, the frequency and diversity of the component words of the concept. Dominance across a domain is a weighted score. Highest possible value is 1000
         1. Click Luke (most dominant concept)
         2. Note that it makes some sense that Luke is the dominant concept. Notice that on the source table. There are 4 separate sources containing the concept. 3 Star Wars and one “Cool Hand Luke”. From our knowledge of these movies, we know how central Luke is to the plots, so it stands to reason that the concept “Luke” would be very important within each of the sources.
      6. Click on the “Indexing Results” for one of the Star Wars plot summaries. Point out the following:
         1. Concepts highlighted in yellow.
         2. Relations underlined.
         3. Negation contexts in red.
         4. Lists of most concepts and CRCs by frequency below.
            1. Button at the top allows us to switch to sorting by dominance.
         5. Manual Input. We can enter an arbitrary sentence and see the indexing results (Note in 2016.1.1 this doesn’t seem to work)
      7. Click on the link showing the External Id for a source. This opens the summary box. Using this box we can generate iKnow Summaries of whatever length text we want.
         1. Generate summaries of length 10 (sentences) and 5 (sentences). Note that iKnow isn’t creating new sentences, it is displaying original sentences with the most frequent occurrences of sentences with dominant scores. It displays them in the original order.

## Smart Matching

1. Demonstrate Smart Matching by adding a dictionary to the MoviePlots domain.
   1. The dictionary matches different ways of referring to the 3 main characters in the movie “2001 a Space Odyssey” Note: Youngsters in the class might need to have this movie explained to them.
2. We cannot add the dictionary to the domain using the iKnow Architect. So, close the iKnow Architect and then open the class definition in Studio (Training.DeepSee.MoviePlotsDomain)
3. Point out the following:
   1. Domain definition contained in XData block named Domain
      1. Point out parameters with the locations of the data, black list elements, configuration, etc.
      2. Going to add a matching dictionary. Set of XML elements.
         1. Copy from file: c:\DeepSee\iknow\2001CharactersDictXML.txt
         2. Paste into XData block in domain definition class. Paste after the <blacklist> </blacklist> element and before the </domain> tag.
      3. Points to make about the XML
         1. One matching element contains 1 or more dictionaries
         2. Each dictionary contains one or more items which in turn contain one or more terms.
            1. The terms are the entities that iKnow will be looking for in the texts
            2. The items are the entities that iKnow will make occurrences of the terms to. In this way iKnow creates a higher level of granularity for analysis of the texts.
            3. Matches do not have to be exact. The exactness can be configured.
      4. Compile the domain definition.
      5. Close Studio
   2. Reopen the domain definition in iKnow Architect
      1. Delete the existing domain. Click Delete
         1. Click Drop domain contents only
      2. Click Build
         1. Point that at the end of the progress report we see output from the matching process
            1. Creating dictionaries and profiles, finished creating…
   3. Open the Knowledge portal. Need to locate the source text for 2001 a Space Odyssey.
      1. Toggle the most frequent concepts to Dominant concept list Scroll through the list to about the 7th page. Click “Dave”
      2. Under Sources find 2001 a Space Odyssey. Click Matching Results
      3. Exact matches for terms are highlighted. Inexact matches are outlined (Hal 9000 unit, for example.)

## iKnow and DeepSee

Use the Aviation demo in Samples to demonstrate use of iKnow from within DeepSee

1. Backgound:
   1. 1200 NTSB incident reports downloaded from NTSB Web site.
   2. Open sample incident report pdf (Aviation Incident Reports directory in classroom materials. Not typically distributed to learners).
   3. Reports contain mix of structured and unstructured data
      1. Structured data contains dates and times, locations, and importantly the most critical injury which can have one of four possible values: None, Minor, Serious, Fatal.
      2. Unstructured data: the narrative field where investigator describes the accident.
   4. This data has been stored in a set of tables in SAMPLES
      1. The table has a row for each incident.
      2. The table has columns for the different structured elements in the report.
      3. The table has a column called “NarrativeFull” which contains the full narrative as a text blob. It might be helpful to draw a quick diagram along the following lines. Note names and numbers of columns not exact.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EventDate | EventTime | Location | InjuriesHighest | NarrativeFull |
|  |  |  | Serious | Blah blah blah blah blah blah …. |
|  |  |  | Minor | Blah blah blah blah blah blah … |

1. To set up the Aviation Demo, open Terminal, point it at the SAMPLES namespace and execute the following command:  
     
   SAMPLES>Do ##class(Aviation.Utils).Setup()  
     
   Do not close the Terminal until the iKnow processing completes
2. Show the Aviation event reports dashboard
   1. Open the Aviation Events dashboard from the DeepSee User Portal in SAMPLES
   2. Bottom right widget shows the top concepts from the narratives listed in order of spread (how many of the sources the concept appears in)
   3. The top right widget is probably the most interesting. It displays the values drawn from the InjuriesHighest source table column along the columns and it displays values drawn from iKnow analysis of the Narratives along the rows.
      1. The iknow values come from applying a matching dictionary. The dictionary contains 4 items (none, minor, serious, fatal) and for each item a number of matching terms. For example, matches for none include “no injuries”, “survived”, “not injured”. Matches for “fatal” include “died”, “fatal injuries”, “was killed”, and so on.
      2. In the analysis of the narrative, iKnow finds matches for these terms. The matches are displayed on the rows.
      3. Looking at the right most “fatal” column. This column contains only reports where the InjuriesHighest value is Fatal. In the narrative, iKnow has found descriptions of injuries of other degrees of severity as well. 5 reports also mentioning persons who were not injured, 14 reports mentioning minor injuries, and so on.
      4. The “Serious” column is interesting. This column contains only reports where the InjuriesHighest value is “Serious”, but notice that iKnow has found 2 reports where the narrative seems to describe fatal injuries. There appears to be disagreement between the structured and the unstructured data. Notice that the Minor and None columns contain similar disagreements.
   4. Show the iKnow “Report” interface
      1. On the upper right widget select a cell and launch the detail listing
      2. The detail listings are standard except for the Report column which contains a magnifying glass icon. Click an icon to launch the report
      3. Notice that the report wizard contains the following
         1. Completed text of a narrative
         2. Highlighted dictionary matches
         3. Negation context in red
      4. Scrolling to the bottom there are a number of different iknow options to apply or remove, including:
         1. Generate a summary
         2. Seed the different dictionaries
         3. View the dominant concepts
   5. Finally the widget on the left hand side uses iKnow in the geo listing.
      1. Launch the geo listing and point at one of the pins. The information displayed includes an iKnow generated 5 sentence summary of the narrative.
3. Open the Aviation Events cube in the Analyzer. Point out the following
   1. Dimensions drawn from structured data:
      1. Event Date, Location, Sky conditions
   2. Dimensions based on unstructured data
      1. Entities: Entities indexed by iknow. Sorted in decreasing order by spread. Note that in model contents pane, only the first 100 appear. However when using as a filter all are available. Note: Due to the configuration of the iKnow measure only concepts and not relations are included in the cube.
      2. Dictionaries: Matching dictionaries. First level contains the dictionaries. The second level contains the items. In order to see the terms for the items you need to look at the termlists using the Termlist manager.
4. Open the Aviation Events cube in Architect. Point out the following:
   1. iKnow Measure: Report (most important piece). Configures the iKnow connection for the cube
      1. Source Values: NarrativeFull (the narratives are contained in this property)
      2. Aggregate: Value must be Count for iKnow measure.
      3. iKnow Source: String or Stream means the source is a property of the source class. File means that the sources are in files. Domain means that there is an iKnow domain, separate from DeepSee, that the cube is going to use (note: exercise uses Domain option)
      4. Dictionaries: The iKnow dictionaries. They must exist as termlists defined using the termlist manager (DeepSee🡪Tools🡪Term List Manager.
      5. Note that there are other configuration parameters that can only be added to the measure by editing the source class using the IDE. For example, there is an iKnowParameters attribute that configures this iKnow measure to include concepts but not relations in the iKnow indexing.
   2. iKnow Dimension: EntitiesD
      1. Dimension Type: iKnow
      2. iKnow Type: entity
      3. iKnow measure: Report
      4. One level containing the entities.
   3. iKnow Dimension: DictionariesD
      1. Dimension Type: iKnow
      2. iKnow Type: dictionary
      3. iKnow Measure: Report
      4. Two Levels. First level contains the names of the dictionaries. The second level contains the dictionary items.
   4. $$$IKSUMMARY, $$$IKLINK – Used in Detail Listings.
      1. $$$IKLINK: See the Simple Listing. Creates the iKnow Report tool.
         1. Can use $$$IKLINK[measure] to indicate which iKnow measure to use.
      2. $$$IKSUMARRY. See the Short Listing. Includes an iKnow summary in the listing.
         1. Default is 5 sentence summary
         2. $$$IKSUMMARY[measure,length] – specify a different iKnow measure and summary length.
      3. Use Analyzer to show both Simple Listing and Short Listing.

# Demos - KPI

## Part One: KPI, Properties, and Filters.

* 1. Open the Sales Against Targets dashboard (How To Category)

1. Dashboard displays a scorecard which uses a KPI for its data source
2. Notice that the History column displays a trendline. This requires multiple data points. Multiple data points in a single cell requires that the data source be a KPI.
3. Demonstrate the filters. Notice that the filters do not allow multi-select/.
   1. Open SalesAgainstTargets in Studio (HoleFoods.KPISalesVsTargets). Note that there’s no Architect-like GUI for developing KPIs.
      1. Note the following about the class def:
         1. Extends %DeepSee.KPI
         2. XData block KPI
         3. <property> elements and <filter> elements. Notice how they correspond to what we see in the widget.
      2. This query uses MDX – See sourceType value of the <kpi> value.
      3. %OnGetMDX displays the query that is run each time that the scorecard (or other widget using the KPI) is displayed. Remind that the query could use SQL (%OnGetSQL) or a custom built result set (%OnExecute).
      4. Note the filtering logic hand coded into %OnGetMDX. Compare to pivot table filters which provide their own filtering automatically.
      5. Note that the %OnLoadKPI() method executes after %OnGetMDX() and adds the last column containing the “History” data – a list of values – to the dataset before returning the dataset to the client. The dataset is contained in the %data variable.

* 1. Create a dashboard and add a scorecard widget using the KPI SalesAgainstTargets

Dashboard

Folder: Test

Dashboard Name: KPI Test

Public

* 1. Add a scorecard widget to the dashboard using the SalesAgainstTargets KPI as the Data Source.
     1. Add some Data Properties:
     2. Add a label column: Note that a KPI creates a dataset – rows and columns. The rows are displayed by the label, the cols are the properties.
        1. Label: Category
        2. Display As: Label
     3. Add the following properties with appropriate labels: AmountOfSale, Target, for both display as value.
     4. Remember that the point here is not the scorecard, but the KPI
     5. Add a couple of filters: Year and Product. Use drop down boxes rather than the default control (Type: dropdown). The default control allows selecting multiple filter values, but the filter logic in the KPI does not support this functionality. Demonstrate the filter functionality on the dashboard.

**BACK TO SLIDES**

## Part Two: Build a manual KPI with properties, and filters

Purpose: Show how to build and test a manual KPI.

Setup: Import Utils.xml from C:\DeepSee\Utils if you have not already. \

Optionally import ManualKPI.xml from C:\DeepSee\DemoCode\Other. Import if

you do not plan to build the KPI in the demo and merely want to display it

instead.

Outcome: Training.DeepSee.Demos.ManualKPI and a dashboard containing a pivot table that displays the KPI data.

Note that rather than building the KPI for the demo, you can also import the solution: ManualKPI.xml from C:\DeepSee\DemoCode\Other. The students can all import the solution into their own environments as well.

Note also that part of the motivation for the demo is that we have a way to get information about a set of servers (disk space, server is up or down) and we want to display it on a dashboard, perhaps side-by-side with our other data.

**Create a new KPI and display it on a dashboard.**

* 1. Use the new KPI wizard
     1. File🡪New, click custom tab, double-click “New DeepSee KPI”
        1. Package Name: Training.DeepSee.Demos
        2. Class Name: DemoKPI
        3. KPI Name Training/Demos/Demo KPI
        4. Source Type: manual
        5. Add the following properties:
           + Used
           + Available
           + Status
           + Location
        6. Add the following filter:
           + Server
        7. Add the following actions:
           + MyAction
        8. Click Finish.
  2. Add the following code to %OnExecute() method of the KPI class. Note that this can be copied from C:\DeepSee\Demos\KPIDemoPart1\_manualquery.txt

//Get some randomized disk use/capacity/server status info.   
 Set list = ##class(Training.DeepSee.Utils).GetDiskInfo()  
 Set tFilter = ..%filterValues  
 Set ..%seriesCount=0  
 for i=1:1:$ListLength(list) {  
 Set ..%seriesCount=..%seriesCount + 1  
 Set ..%seriesNames(..%seriesCount)=$List($List(list,i),1)  
 Set ..%data(..%seriesCount,"Used")=$List($List(list,i),2)  
 Set ..%data(..%seriesCount,"Available")=$List($List(list,i),3)  
 Set ..%data(..%seriesCount,"Status")=$List($List(list,i),4)  
 Set ..%data(..%seriesCount,"Location")=$List($List(list,i),5) }  
 Quit $$$OK

* 1. Say a few words about the demo code Uses an API to get server status and disk info. In actuality data is randomly generated. Based on an actual customer example.
  2. Compile. Click View Web page to test.
  3. Note that this example is purely manual KPI. The HoleFoods Sales vs. Targets KPI is an mdx KPI that uses %OnLoadAction to add additional data to the data set retrieved by the mdx query.
  4. Add a pivot table to the KPI Test dashboard that displays all the data.
  5. Add the Server filter to the widget. Note that the filter shows up on the list, but there are no options to select and there is no logic behind it. It does nothing.

Note

**Implement the Filter Logic**

1. Return to the KPI class in Studio. Point out that the filter logic needs to be added to the %OnExecute().
2. Paste the following code over the existing code in %OnExecute(). Find the code in C:\DeepSee\Demos\KPIDemoPart2\_filters.txt  
     
    //Get some randomized disk use/capacity/server status info.   
    Set list = ##class(Training.DeepSee.Utils).GetDiskInfo()  
      
    Set tFilter = ..%filterValues  
     
    Set ..%seriesCount=0  
     
      
    for i=1:1:$ListLength(list) {  
      
    If ( (tFilter.Server = "") || (tFilter.Server = $List($List(list,i),1)))  
    {  
    Set ..%seriesCount=..%seriesCount + 1  
    Set ..%seriesNames(..%seriesCount)=$List($List(list,i),1)  
    Set ..%data(..%seriesCount,"Used")=$List($List(list,i),2)  
    Set ..%data(..%seriesCount,"Available")=$List($List(list,i),3)  
    Set ..%data(..%seriesCount,"Status")=$List($List(list,i),4)  
    Set ..%data(..%seriesCount,"Location")=$List($List(list,i),5)  
    }  
      
     Else{  
       
     continue  
       
     }  
      
      
    }  
      
    Quit $$$OK
3. This new version of the code will filter by the Server name: (Sammy, Deano, or Frank) depending on which has been selected.
4. We also need to update the

## Part Three: Add custom actions to the KPI .

Purpose: Show how custom actions work and how to add them to a KPI.

Setup: None

Outcome: Custom action added to Demos/ManualKPI.

* 1. Open Demos/ManualKPI in Studio and create the following action:
     1. Add the following <action/> element to <kpi>

<action name="MyAction"/>

* 1. Add the following code to %OnDashboardAction

If pAction="MyAction"

{

Set pContext.command = “alert: “\_pContext.valueList

}

* 1. Compile
  2. Return to the KPITest dashboard and add “MyAction” to the pivot table.
     1. Location: Widget
     2. Action: MyAction
     3. Type: auto
     4. Control Label or icon: “MyAction”
  3. Set the table to “Multi-Select
  4. On the scorecard, select a row and then click the “MyAction” button.

# Demos – Security

Purpose: Show how to secure DeepSee tools (Analyzer, Architect, etc.) and DeepSee elements (pivots, folder, cubes, etc).

Setup: Assumes installation with Normal security settings.

If security settings are minimal, have everyone who wants to follow the demonstration do the following:

Note: to avoid session caching issues, use 2 browsers throughout the demo. Make security configuration changes in one browser. Demonstrate the effects of those changes using the other browser, logging in as the demo user whose permissions are changing throughout the demonstration.

1. Create a new user (System Administration 🡪 Security 🡪 Users)
   * 1. Name: DSTrainingUser
     2. Password: SYS
     3. Assign No roles
2. Login to the portal as DSTrainingUser. User has no privileges to has access to no tools
3. Look at DSTrainingUser’s profile (System Administration 🡪 Security 🡪Users)
   * 1. Does have %DeepSee\_Portal. Why is the user unable to access the portal? No permissions on any databases.
4. Update DSTrainingUser by adding the %DB\_SAMPLES role.
5. Logout and then login as DSTrainingUser. Note the following:
   * 1. DeepSee menu available, but only in SAMPLES.
     2. User Portal tool available. Analyzer and Architect are not.
        1. Admin 🡪 Logs available. Other Admin tools are not.
        2. Tools 🡪 Model Browser available. Other Tools are not.
        3. User Portal: Can launch dashboards. Cannot edit them. Cannot create new ones. Why? Has only %DeepSee\_Portal and not %DeepSee\_PortalEdit.
        4. User Portal cannot launch pivots. Why not? Pivots launch in Analyzer. DSTrainingUser does not have privileges to use Analyzer. What privileges would DSTrainingUser need? Needs at least %DeepSee\_Analyzer.
6. To give DSTrainingUser %DeepSee\_Analyzer privileges, we need to create a role and then assign that resource to the role.
7. Create the DSTraininRole role (System Administration 🡪 Security 🡪 Roles).
   * 1. Name: DSTrainingRole
     2. Privileges: %DeepSee\_Analyzer:U
8. Assign DSTrainingUser to DSTrainingRole.
9. Login as DSTrainingUser. Note the following:
   * 1. Analyzer no longer greyed out on DeepSee Menu
     2. User can open Analyzer
     3. User can launch pivot in Analyzer from User portal.
     4. User cannot save or modify any pivot tables.
10. Assign a custom resource to a folder of pivots and dashboards (aka DeepSee Folder Items)
    * 1. First look at the User Portal in List mode and point out the Training/DeepSee folder. We have been putting all of our pivots and dashboards in this folder or its sub folders all week.
      2. Create a custom resource (System Administration 🡪 Security Management 🡪 Resources)
         1. Resource Name: DSTrainingResource
         2. Public Permissions: None
         3. Click Save
      3. Open the Folder Manager (DeepSee 🡪 Admin 🡪 Folder Manager)
      4. Click the Training Row.
      5. Select DSTrainingResource on the left under “Resource”
      6. Click Save Folder
11. Log back into portal as DSTrainingUser. Note the following:
    * 1. Training folder and sub folders not visible in User Portal (use list mode to make this obvious)
      2. Training Folder not available in Analyzer. Open Analyzer and then click open. Training folder not available here either.
12. Edit the DSTrainingRole to add privileges on the DSTrainingResource
    * 1. System Administration 🡪 Security 🡪 Roles
         1. Add DSTrainingResource:U to DSTrainingRole
13. Test DSTrainingUser’s re-enabled access to the Training folder.

# Demos – DeepSee APIs (Appendix)

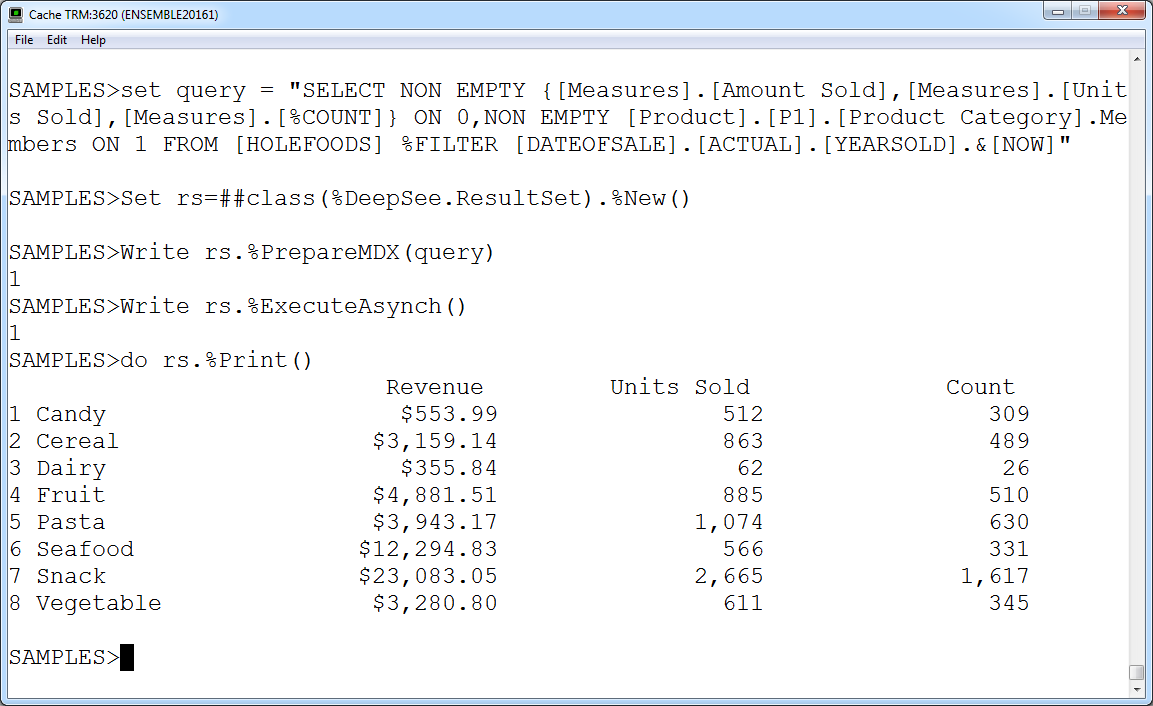
## Demo 1: ResultSet API

Purpose: Show how to use %DeepSee.ResultSet to execute an MDX query and work with the results.

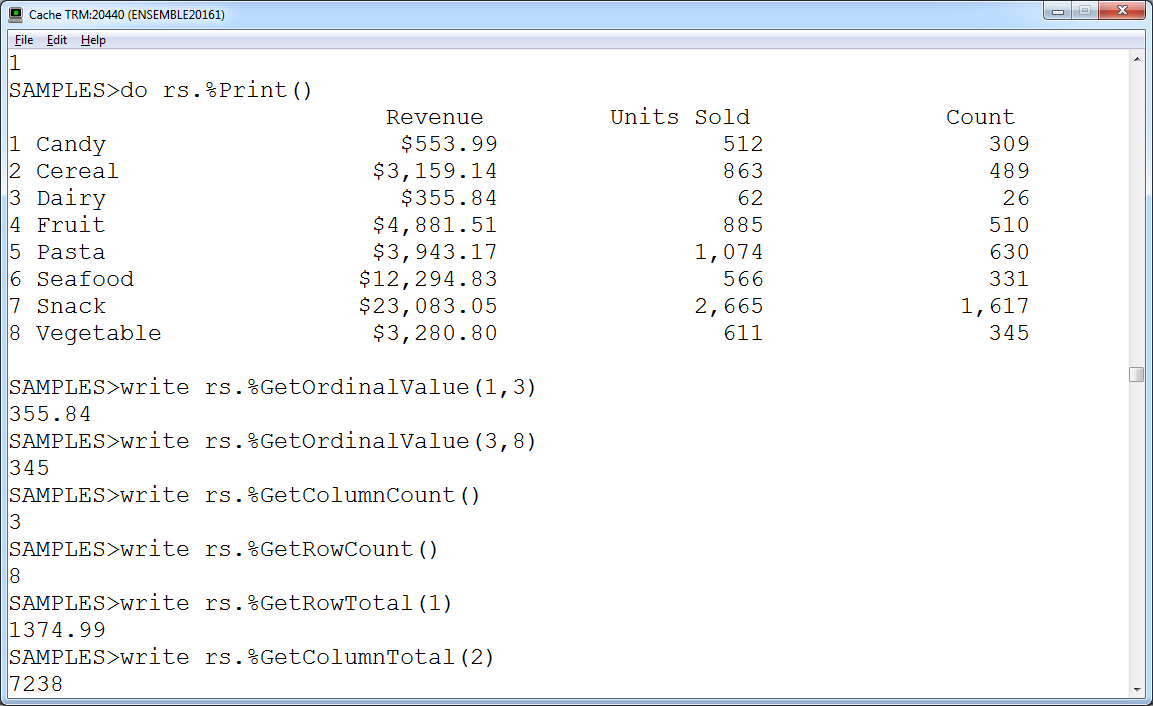
Setup: None

Outcome: None.

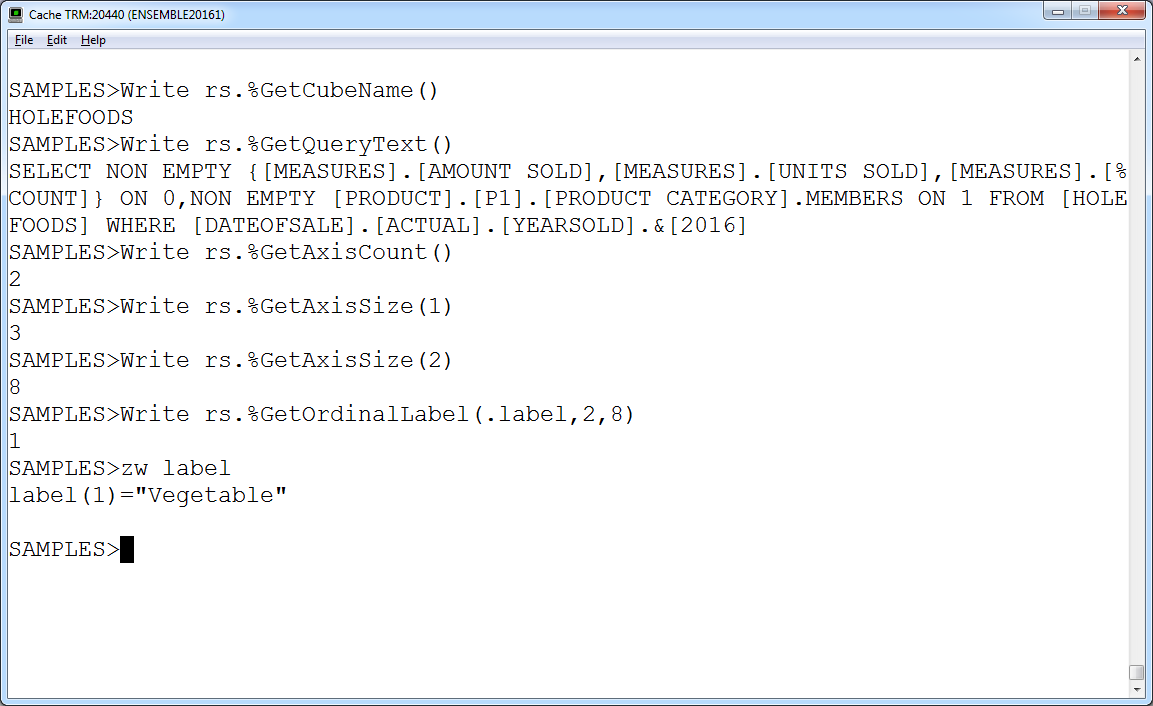
1. Open Terminal and change to the SAMPLES namespace. Execute the following steps to show the basic instantiation of a result set, the preparation and execution of a query, and the display of the results. Note that the query text can be copied from C:\DeepSee\Demos\DeepSeeAPIDemoCode



1. A couple of points to note about the query execution:
   1. %ExecuteAsynch() uses DeepSee process, so query executes in parallel.
   2. %Execute() uses only the current process, so no parallel execution.
   3. %ExecuteDirect() prepares and executes the query, so no need to use %PrepareMDX(). Executes using only the current process
2. Continue with some more methods for extracting data, including:
   1. %GetOrdinalValue()
   2. %GetColumnCount()
   3. %GetRowCount()
   4. %GetColumnTotal()
   5. %GetRowTotal()



1. Finally, show some methods for extracting meta data, including:
   1. %GetCubeName()
   2. %GetQueryText()
   3. %GetAxisSize()
   4. %GetGetOrdinalLabel()



## Demo 2: REST API

Purpose: Show how to configure and use the DeepSee REST APIs

Setup: holefoodsrest.csp installed in SAMPLES. Note that successfully running Training.DeepSee.Installer:Install() will install this code.

1. Open the sample DeepSee.RESTClient.cls in a web browser. You can do this by first opening it in Studio and then clicking View 🡪 Web Page
   1. Note that this is a demonstration page available in SAMPLES out of the box. It is not something we added for the training class.
   2. Depending on the security settings, you may get prompted for user names and passwords whenever you try to invoke one of these services.
2. Note that the left column of the table shows the REST services exposed by DeepSee. Demonstrate a couple of the “informational” services:
   1. Click /Info/TestConnection and then click Submit
   2. Click /Info/Cubes and then click Submit
   3. Click /Info/Pivots and then click Submit
   4. Note that none of these require that we provide any information in our “Posts”.
   5. Note also that the first element in the JSON object we get back is “Info” and it has an “Error” property. If the service returned an error, we would see it here.
3. Show the Info/Pivots service
   1. Add some content: {“BaseCube”:”HoleFoods”} – note that must add this content must be a JSON object {Name:Value} pair.
   2. Note a couple of things about the Request
      1. Accept: sets the Accept property of the Request Header to “application/json”. This is required. The client must accept JSON. If you set the value to Text then the value of this property becomes “text/plain” and you will get an error.
      2. Content-Type: sets the Content-Type property of the request header, which specifies the MIME type of the body of the request. Used only with POST and PUT. Both test/plain and application/json seem to work, but I think the latter is to be preferred.
4. Demonstate /Data/PivotExecute
   1. Copy one of the Pivot Names from the list of Pivots returned at step 3, for example:   
      Training/DeepSee/Test/Revenue And Units Sold by Category
   2. Click /Data/PivotExecute
   3. For the content create the following, pasting the pivot name that was copied at an earlier step:  
      {“PIVOT”:” Training/DeepSee/Test/Revenue And Units Sold by Category”
5. Demonstrate /Data/MDXExecute
   1. Click /Data/MDXExecute
   2. Copy and paste the JSON object containing the MDX to execute out from C:\DeepSee\Demos\DeepSeeAPIDemoCode.txt
   3. Execute the query
   4. Note all the information returned, data as well as meta-data.
6. Demonstrate holefoodsrest.csp
   1. Open in studio. Click View🡪Web Page. Make sure the username, pwd, host, and port are set correctly. Click Execute MDX.
   2. Note that the intent of this is to be much simpler than DeepSee.RESTClient. it executes a hard-coded MDX query and dumps the JSON response data into the browser.
   3. Look at the code in Studio. Basic outline of steps are:
      1. Note that we are using JavaScript in this code. Other languages: Java, .NET, Python, etc., will have similar libraries.
      2. Create the right url. Notice the "/api/deepsee/v1/SAMPLES/Data/MDXExecute" part.
      3. Create and configure the XMLHttpRequest object. Configuration includes setting the credentials and the details of the request header.
      4. Use the XMLHttpRequest send method to invoke the service sending the contents, in this case the MDX query to execute. Note that send is only used with PUT or POST. Not used with GET or HEAD
      5. Method for handling the response when it returns. In this case just dump the JSON to the screen. In a real application you would add code here to parse the returned JSON and pull out the data of interest.

## Demo 3: JavaScript API

Purpose: Show how to configure and use the DeepSee JavaScript APIs

Setup: holefoodschartdc.csp installed in SAMPLES. Note that successfully running Training.DeepSee.Installer:Install() will install this code.

1. Open test3rdPartyCharts.html in a web browser. Note that cannot open in Studio and use View 🡪 Web Page. Url is http://localhost:<port>/CSP/samples/test3rdPartyCharts.html
2. Purpose is to show the DeepSee JavaScript API being used by JavaScript charting and graphics libraries. You can copy and paste example queries from C:\DeepSee\Demos\DeepSeeAPIDemoCode.txt to see the results.
3. Open holefoodschartdc.csp in Studio and then click View🡪Web page to show it in the browser.
   1. Put in correct connection information and click “Show Chart”. Show off some of the various features:
      1. Bubbles when pointing at the different bars.
      2. Using the squares at the bottom to isolate different measures.
      3. Filtering by region.
   2. Note this is using AM charts. Watermark in left-hand corner can be removed if you pay for use of the charting package.
   3. This is intended to be a simplified version of the earlier page allowing us to read and understand the code more easily.
4. Look at the code in Studio.
   1. Start with <script> tags doing the includes. Note that we need to include the following:
      1. DeepSee.js, and zenCSLM.cs – these provide the InterSystems JavaScript library for interacting with DeepSee. These must always be included in order to use the DeepSee javascript library.
      2. amcharts.js and serial.js – these are provide the amcharts charting package. Different javascript based charting packages will, of course, have different files to include.
   2. Next, look at initializePage(). Note the following:
      1. Here we create a configuration object which contains lots of information, including:
         1. The MDX query that we want to execute.
         2. The Caché connection information which we use to instantiate a DeepSee connection object.
      2. We create the DeepSeeDataController object. This object is where the magic really happens. It manages the connection between the web page and DeepSee running on the Caché server. It creates “state” for the page across requests to the server. Remember the actual connection between Caché and the web client is really stateless. But having this object giving the appearance of state will allow us to do things like filter, drilldown, run listings, etc.
      3. Notice that the second argument to the DeepSeeDataController object is the name of the method to run in the client after the runQuery method executes.
      4. Note the runQuery() method which executes the query defined in the DeepSeeDataController’s configuration.
   3. Next look at the drawChart() method. Point out the following:
      1. Retrieving the data from the data controller
      2. Looping through the result set and creating an array of data points. This array will be given to the chart to populate it.
   4. Look at the AmCharts.makeChart() method.
      1. This method was mostly generated using online tools found on the amcharts web site. There you can visually create your chart.
      2. Scroll down towards the end of the method and notice the “dataProvider” key. Here is where we configure the chart to use the array of data created in drawChart().
   5. Finally, talk about the filter
      1. Look back at drawChart(), we see the population of the dropdown control for the filter using the DeepSeeUtils.getMembersForFilter method.
      2. Look at applyFilter(). Here is the method that executes when the user clicks “Apply Filter”. The most important part is that it executes the data controller’s applyFilter() method giving it the filter partially constructed by the user’s selection on the filter dropdown.