Pawsitivity Vets Case Study

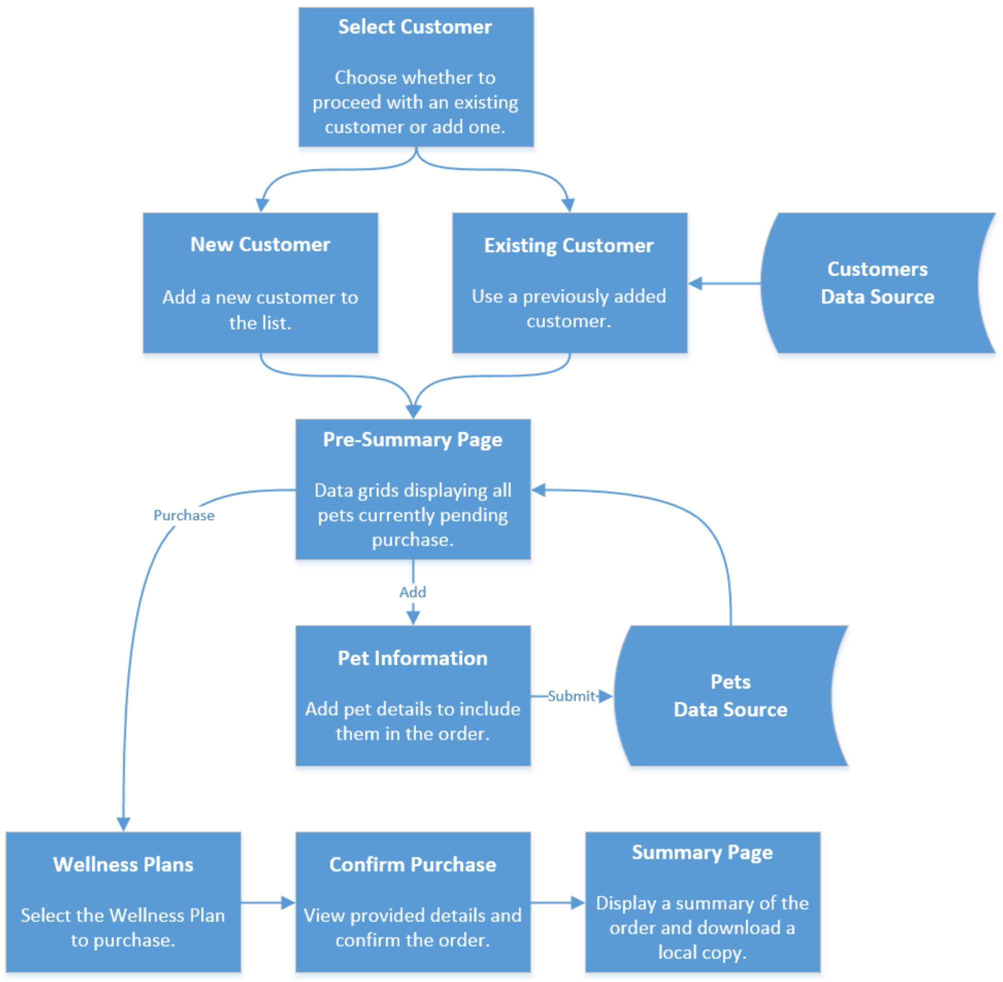
Difficulty Rating: Advanced

27 April 2015

KnowledgeKube

Pawsitivity Vets

You have been tasked with creating a model for Pawsitivity Vets, an American insurance company that sells veterinary insurance plans to cat and dog owners. The questionnaire will allow users to purchase one of Pawsitivity's new Wellness Plans.


Select Customer
Choose whether to proceed with an existing customer or add one.
New Customer
Add a new customer to the list.
Existing Customer
Use a previously added customer.
Customers Data Source
Pre-Summary Page
Data grids displaying all pets currently pending purchase.
Purchase
Add
Pet Information
Add pet details to include them in the order.
Submit
Pets Data Source
Wellness Plans
Select the Wellness Plan to purchase.
Confirm Purchase
View provided details and confirm the order.
Summary Page
Display a summary of the order and download a local copy.

To begin with the user can select if they want to use an existing customer profile or create a new one. After this they will be asked to add the pets they want to insure and provide details about them, before they get to select which Wellness Plan they wish to purchase. Once they have selected their preferred plan, they are asked to confirm the provided details and then complete the order. At this point they will be shown a summary of the plan they chose, which they can download a as local file if required.

Create Question Groups

The Pawsitivity Vets model will consists of eight question groups in total. Create the following question groups:

- **SelectCustomer** - Asks the user whether they want to create a new customer or use an existing one.
- **NewCustomer** - Allows the user to add a new customer to the database by providing personal information.
- **ExistingCustomer** - Here the user can select an already existing customer from a list.
- **PreSummaryPage** - This group serves as the hub area where the user is shown all pets they have added, separated into two data grids. They have the option to add more pets or proceed to select a plan.
- **PetInformation** - Here the user can provide information about a pet they want to purchase an insurance plan for.
- **WellnessPlans** - In this group the user will be presented with the available insurance plans, allowing them to select the one they wish to purchase.
- **ConfirmPurchase** - Here the user will see a brief summary of their chosen plan - including any fees they are liable to pay - and asked to confirm their purchase.
- **PurchaseSummary** - Lastly, the user should be shown a summary of their order and have the option to download a local document containing the same information.

Add Navigation Buttons

Add navigation buttons to each question group. The *NewCustomer* and *ExistingCustomer* groups should both lead to the *PreSummaryPage*. This should in turn feature a **Submit** button that leads to the *WellnessPlans* group.

PetInformation should lead back to the *PreSummaryPage*. The *WellnessPlans* group should feature a **Proceed To Purchase** button which points to *ConfirmPurchase*. Add a **Purchase** button to this group and direct it to the *PurchaseSummary* group.

Add **Back** buttons to all groups taking you back to the previous group. Make sure to add a Cause Button Validation attribute to all Back buttons so that they can be used without triggering the mandatory validations in the current question group.

Add Buttons

The *SelectCustomer* question group offers a very simple option for the end user, asking them to select whether they want to create an order for a **New Customer** or an **Existing Customer**. Create one button for each option and use the ShowGroupForm expression so that each button navigates the user to the respective question group.

The *PreSummaryPage* group should also have an **Add** button which points to the *PetInformation* group, in order to allow the user to add another pet to their order.

Questions

The *NewCustomer* question group is intended to gather personal information about the customer, including their name, address, email, contact number and a Date question where they can select when they want their insurance cover to begin.

Meanwhile, the *ExistingCustomer* group should only consist of a multiple choice question where the user can select a previously created customer. There is no need to add responses to this question as it will later be connected to a data source to automatically fill with customers that are created in the *NewCustomer* group.

Add a series of questions to the *PetInformation* group that relate to the pets the customer wants to add. This should start with a Yes/No question where they can select either "Cat" or "Dog". The questions following this should inquire about the animal's name, gender, date of birth, weight and another Yes/No question asking if they have been spayed or neutered.

One of the questions in the *NewCustomer* group should ask which **State** the customer is from so add all 50 US states as responses. Add a response called "- - Please Select - -" to the State multiple choice question in the *NewCustomer* group and set it so that this can not be selected as a valid answer.

Add Visibility Expression to Breed Questions

Underneath the first question in the *PetInformation* group you should add two multiple choice questions called *CatBreed* and *DogBreed* respectively. Apply a visibility expression to each question so that they only appear when the respective species has been selected in the previous question.

Use Read Only Text Questions to Display Information

Set up a series of Read Only Text questions in the *ConfirmPurchase* group to present the end user with a brief summary of the choices they have made. Create the questions so that they display the chosen cover start date, the name of the selected plan, the monthly fee and the cost of enrolment. Finally, add a read-only question that displays the final amount the customer will pay at the time of purchase. This figure should be calculated by combining the monthly fee and the cost of enrolment using an expression.

Create Variables

Create the following set of variables:

- **CustomerName** - This should be set as the Result Target Property of two questions; the name question in *NewCustomer* and the drop-down menu in *ExistingCustomer*. This will make it easier to use the customer's name in later questions without having to account for two different question keywords.
- **PetBreed** - Similar to the CustomerName variable, this will allow you to grab the selected breed regardless of whether the customer added a cat or a dog. Once created, set it as the Result Target Property of both the *CatBreed* and *DogBreed* questions.
- **SelectedPlan** - This variable will be used to grab which plan the customer has selected after they have added all their pets.
- **PetStatus** - The status will be used later on to determine which pets have been submitted and which ones are still pending.

Set Up Data Sources

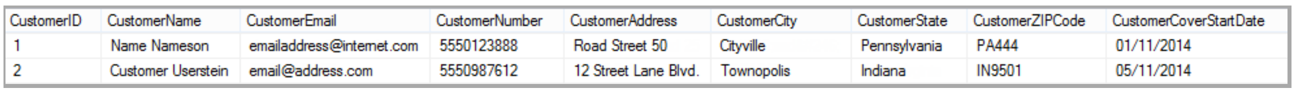
Create the following SQL data source tables:

- **Customers** - This table will be used to hold all customers added to the site. When one is created in the *NewCustomer* question group, their information will be written to the table. The table should also be connected to the *ExistingCustomers* group, allowing you to select them at a later date.
- **CatInformationGrid/DogInformationGrid** - These tables are used to hold the details of all pets added to the system by customers. Most of the data will be gathered using the *PetInformation* group, although each pet's details will include information about the owner in addition to a status noting whether the pet's insurance order is submitted or pending.
- **WellnessPlans** - This table is used to display information about each of the available Wellness Plans from which the customer can select the one they want to purchase.

- **CatBreeds**/**DogBreeds** - These tables are used to provide responses to the Breed drop-down questions in the *PetInformation* question group.

Customers Table

The *Customers* data source should be set up to collect all customer information whenever a new user is added in the *NewCustomer* question group. This is done by constructing a WriteData expression and attaching it to the Next button in that group. The expression should save all the user's information, including their name, email, number, address, city, state, zip code and the date their cover should start.



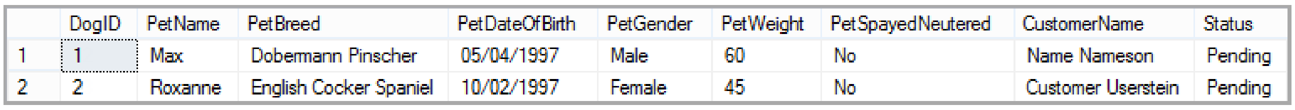
CustomerID	CustomerName	CustomerEmail	CustomerNumber	CustomerAddress	CustomerCity	CustomerState	CustomerZIPCode	CustomerCoverStartDate
1	Name Nameson	emailaddress@internet.com	5550123888	Road Street 50	Cityville	Pennsylvania	PA444	01/11/2014
2	Customer Userstein	email@address.com	5550987612	12 Street Lane Blvd.	Townopolis	Indiana	IN9501	05/11/2014

Figure 2-1: Example of a Customers data table.

The same data source should then be connected to the drop-down menu in the *ExistingCustomer* group. Set it up so that the column containing the customer's name is used as the list of responses. By doing this you will allow the list of existing customers to automatically contain any customers that are added in the *NewCustomer* group.

Pet Information Tables

The *CatInformationGrid* and *DogInformationGrid* data sources should be used to gather the responses in the *PetInformation* question group with an expression. The tables should be identical and contain the pet's name, breed, date of birth, gender, weight and whether they have been spayed or neutered. In addition, they should also have columns for the customer name and the pet status.



	DogID	PetName	PetBreed	PetDateOfBirth	PetGender	PetWeight	PetSpayedNeutered	CustomerName	Status
1	1	Max	Dobermann Pinscher	05/04/1997	Male	60	No	Name Nameson	Pending
2	2	Roxanne	English Cocker Spaniel	10/02/1997	Female	45	No	Customer Userstein	Pending

Figure 2-2: Example of how a Dog Information table might look.

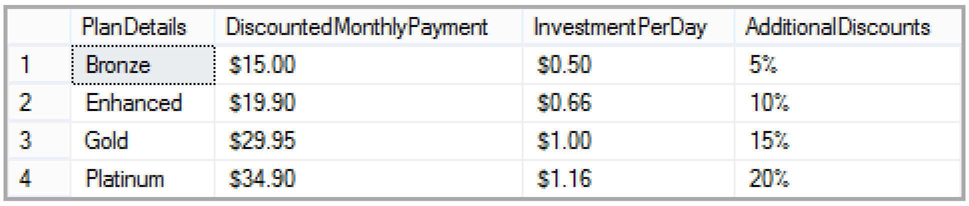
The reason you will be using two separate grids is to keep cats and dogs separate. However, this requires that you set up a slightly more advanced expression so that KnowledgeKube knows which data table to write the information into. You can accomplish this by creating an if-expression and adding it to the **Submit** button.

In this expression, the first statement should check whether "Cat" or "Dog" has been selected in the Pet Type question. The following functions should be two WriteData expressions which should be set up to write all the pet's details as well as the customer's name and the *PetStatus* variable to the respective data source table. Because the column names are the same in both the cat table and the dog table, the expression can be used to correctly pass values to either one.

When a pet is added it should have the value "Pending" passed to its *PetStatus* column, which will later be used to filter a data grid. Use an expression to update the *PetStatus* variable and make sure it runs before the WriteData expression so that the correct value is captured.

Wellness Plans

This data source will be used to create a simple, static data grid where the user will be shown information about the different available plans. Pawsitivity provide four plans; **Bronze**, **Enhanced**, **Gold** and **Platinum** and the table should show the **Discounted Monthly Payment**, **Investment Per Day** and **Discount** percentage offered by each selection.

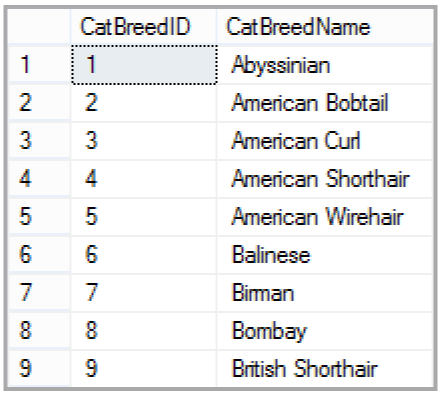


	PlanDetails	DiscountedMonthlyPayment	InvestmentPerDay	AdditionalDiscounts
1	Bronze	$15.00	$0.50	5%
2	Enhanced	$19.90	$0.66	10%
3	Gold	$29.95	$1.00	15%
4	Platinum	$34.90	$1.16	20%

Figure 2-3: A Wellness Plans data table.

Breed Responses

One of the questions in the *PetInformation* group asks the customer to select the breed of their pet. Since the list of available breeds can be very long, it would be easier to manage the responses by connecting the questions to a data source. Create two data tables, one for cat breeds and the other for dog breeds. Add a list of the available breeds to the tables and connect them to the respective questions. As these tables should only consist of a static list of responses you will not need to employ any filters.



	CatBreedID	CatBreedName
1	1	Abyssinian
2	2	American Bobtail
3	3	American Curl
4	4	American Shorthair
5	5	American Wirehair
6	6	Balinese
7	7	Birman
8	8	Bombay
9	9	British Shorthair

Figure 2-4: An example of how a table of cat breed responses might look.

Data Grids

Create two data grids in the *PreSummaryPage* question group and connect them to the *DogInformationGrid* and *CatInformationGrid* data sources. These will be used to display the responses gathered in the *PetInformation* group.

These grids should only display pets added by the current user and no pets which they have already purchased a plan for, so you will need to create a filter. This filter should exclude any rows that do not match the necessary criteria and also refreshes the grids when a new pet has been added, so all in all you will need to add three conditions:

- The first condition refreshes the filter when the *ID* column is greater than 1. Since this will be true as soon as one pet is added, it will automatically cause the grid to update whenever a new item is added.
- The second condition should make sure that the value in the *CustomerName* variable matches the value in the *CustomerName* column.
- The last condition checks that the value in the *PetStatus* column is "Pending".

Use the *WellnessPlans* data source to create a data grid in the *WellnessPlans* question group. This grid should present the end user with information about the various available plans.

To allow the user to select a plan, add an Action Link column and create an action which passes the keyword for the column that holds the name of the plan into the *SelectedPlan* variable. When the button is clicked it will save the name of the selected plan in the variable.

Use Action Links to Remove Pets

Create a filter for each data source that compares the value in the *ID* data source column with the corresponding *ID* column in the data grid.

Follow this by creating two actions, one for each of the pet information data sources, which should consist of a DeleteData expression. The first argument in the expression should be the name of the data source while the second should be the filter you just created. The last argument should be set as "False".

Now add an Action Link column to the *PreSummaryPage* data grids and add the actions. Make sure that you properly connect the action to remove a dog to the *DogInformationGrid* and vice versa.

Update PetStatus Variable on Purchase

When a pet is initially added it will be given the *PetStatus* value "Pending" to indicate that they have not been committed to a purchase yet. However, when an order is submitted the *PetStatus* of all pending pets should change to "Submitted" in order for the data grid filters to work correctly.

Create a custom action named *StatusChange* containing two expressions:

- The first expression should change the value of the *PetStatus* variable to "Submitted".
- The second is a WriteData expression which updates the *PetStatus* column in the *InformationGrid* data table.

Add a filter to the *CatInformationGrid* and *DogInformationGrid* data tables that returns only rows containing the name stored in the *CustomerName* variable, and have a *PetStatus* of "Pending".

Finally, add a ForEachDataSourceRow expression to the **Purchase** button in the *ConfirmPurchase* question group. This expression should call the *StatusChange* action, write to the *PetInformationGrid* and use the *Pending* filter. When clicked it will update the *PetStatus* variable and change the status of all pending pets to "Submitted".

Document Template

Create a Word template document using the **Mercato Document Manager**. This document should act as a receipt for the end user and incorporate keywords from the Pawsitivity model to present dynamic information. The receipt document should show the following information:

- The end user's name, contact number and address.
- The number of animals they purchased cover for.
- The start date for the policy.
- The applicable fees, such as monthly payment, the enrolment fee and the total cost paid at purchase.
- Pawsitivity Vets' contact information and address.

Purchase Summary

When the user clicks the **Purchase** button they should be taken to the *PurchaseSummary* question group where they are informed that the purchase was successful and given a brief summary including the name of the plan they purchased, the date the cover starts and the applicable fees. Use Read Only Text questions to display this information.

The user should also be able to download a document containing a copy of the summary information. Write an action that downloads the Word template you created and connect it to a button at the bottom of the *PurchaseSummary* group.

Actions	✓
Attributes	✓
Buttons	✓
CDS	
Data Processing	
Data Source Grids	✓
Data Sources	✓
Documents	✓
E-Mail Templates	
Expressions	✓
Language Translations	
Model Scheduler	
Model Variables	✓
Output Designer	
Rating Definitions	
Repeating Question Groups	
States	
Summary Page Designer	
Users/Roles/Workgroups	
Validators	

Abstract

The Pawsitivity Vets Case Study was written to illustrate how data sources can be used to create a purchasing application.

The questionnaire is built to allow users to buy veterinary insurance for their cats and dogs. New customers can be added to the system, while existing customers can be re-used for additional, subsequent transactions.

Multiple pets can be added to an order and they will be saved in an SQL data table and displayed on the site using a data grid.

After providing details about themselves and their pets, the customer will be able to choose from a range of suitable cover plans. They will then be asked to confirm their details and can choose to proceed with the order.

Finally,the user will be given a short summary and have the opportunity to download a PDF summary of their order.

Prerequisites

Before you begin this case study, you will need the following:

- You must have the facility and basic understanding to set up a SQL-compatible database. If you are not sure of any details, such as your SQL server address, you may need to contact a system administrator.