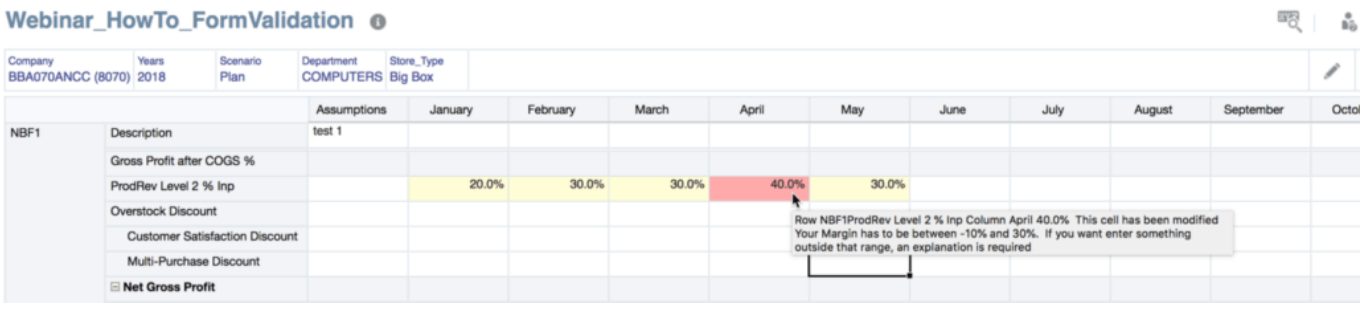
Adventures in Groovy – Part 32: Require Explanations When Data Is Outside Acceptable Ranges

Groovy offers creative ways to force data into acceptable ranges. But, what happens when there is an acceptable reason for abnormal trends, or drivers? With Groovy, you can not only add limits to data cells, but you can also allow entry outside of those ranges if the user enters an explanation. With Groovy, you get the best of both worlds!

Use Case

Requiring data within limits is a huge improvement to your application. It improves the accuracy and ownership of the budget, and reduces fire-drills at the end of your budget cycle. In this example, we will require a rate to be between -10 and 30 percent. If a user enters data outside that range, the form will not be able to be saved.

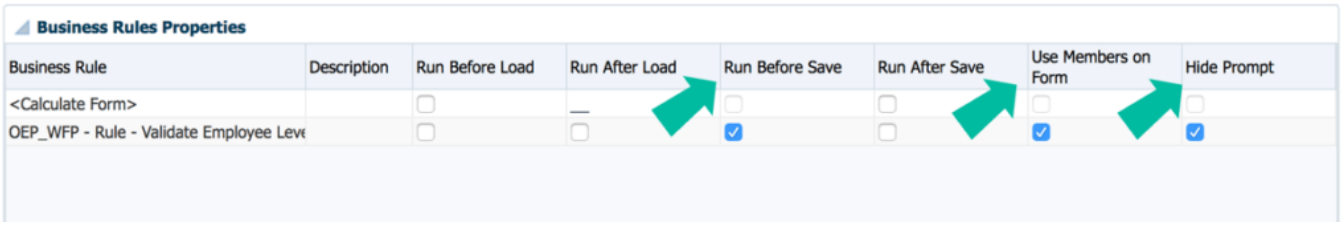


However, if a user enters a comment, Groovy will then allow the form to be saved. Is this perfect? Of course not. A user can still go in and enter a poor explanation, or just enter a space to bypass the validation rule. I have always felt that no system can make people be responsible, so we will assume that your users are doing what they are instructed and

your leadership will hold them accountable. To show this functionality, view this short video.

The Code

This validation is actually quite simple. This follows the same rules as other validations and must be “Run Before Save.” Use Members on Form and Hide Prompts is also suggested.




```
// Define cell colors for errors
def BackErrColor = 16755370
// Iterate through the row with the rate and only the months - exclude quarters and totals
operation.grid.dataCellIterator('ProdRev_2_%_Inp','Jan','Feb','Mar','Apr','May','Jun’,'Jul','Aug','Sep','Oct','Nov','Dec').each {
  // If data is above .3 or lower than -.1 and does NOT have a cell note, throw the error
  if( (it.data > 0.3 || it.data < -0.1) && !it.hasCellNote() ) {
    it.addValidationError(BackErrColor,"Your Margin has to be between -10% and 30%.  If you want enter something outside that range, an explanation is required", false)
  }
}
```

Finishing Up

I really hope this inspires you to be creative about using Groovy to add useful user functionality. As you can see, you can add awesome functionality relatively easily. If you have any creative solutions you would like to share, please add a comment.