Provo City Employee Essentials Training

I work for Provo City in the HR department. All employees (except seasonal, on-call, and contract employees) are required to take 10 online "Employee Essentials" courses. These include courses on Sexual Harassment, Workplace Safety, Defensive Driving Etc... When employees complete a course it shows up in a spreadsheet on the course website. I download this spreadsheet and check the time stamp for each course to make sure they were spending the appropriate time on the course and not just clicking through. When I could confirm that someone has not been completing the courses correctly I wrote and sent them an email telling them to do it again. I would then save and upload the spreadsheet into our HRMS (Peoplesoft). This involves changing the formatting on the sheet to match what the HRMS is looking for and saving it to the correct directory with the correct file type and name.

Once uploaded, we did not have a great way of tracking who had and had not completed these courses. We could only look them up one-at-a-time. I have created a way for us to generate a list of people who have not completed the courses so we can track completion more easily.

There are three parts to my project: Processing Course Data, Generating Rejection Emails, and Tracking Training over time

Processing Course Data

I created a Sub that performs preliminary checks on course data, highlights all acceptable courses in Yellow, leaves incomplete and failed courses un-highlighted, and highlights all courses that need to be rejected in Red.

My code checks the following things:

- Was the course passed? If the course was incomplete or failed the row is not highlighted and the code moves to the next course.
- Was the course completed in sufficient time? The system will say a course was passed as long as the quiz is passed, even if it was done in much less time than the actual course length. These courses must be rejected.
- Does it look like the course was completed too quickly because the employee was finishing a previous training session that was interrupted, or retaking a quiz? This is indicated by the "Training Method" column. In this case, the row is highlighted in Yellow and the training method cell is given a different border to call for additional investigation.

The code goes through a Do Until loop to look at each row and assess the course on the above criteria. It also sorts the data by the employee's last name, creates appropriate headers and formats dates and ID numbers as required for upload.

Once I run this code, I do still need to go in and look at incomplete courses next to courses highlighted in red, as these may have been interrupted and the employee neglected to make the indication in the

Training Method. Once I've double checked everything, I create and send emails for the courses being rejected (See Below) and delete all rows that are not highlighted in yellow.

I can then save the file with a formatted name, the correct File Type, and in the correct directory. Using the Save subprocedure I created. This Sub contains code which allows it to be used with another set of training data which I will not be discussing in this write-up. The code knows which course I'm dealing with by checking a certain cell which is always empty for one of the sets of data and always contains a value for the other set. It then saves the file with an appropriate name. This Sub also performs a simple check on the ID numbers for all of these courses. Sometimes employees enter their ID Card number instead of their ID number which, uncorrected, would cause the upload to be rejected. My code checks to make sure all of the ID numbers are the length of a correct ID number and then displays a message box if something needs to be changed.

Generating Rejection Email

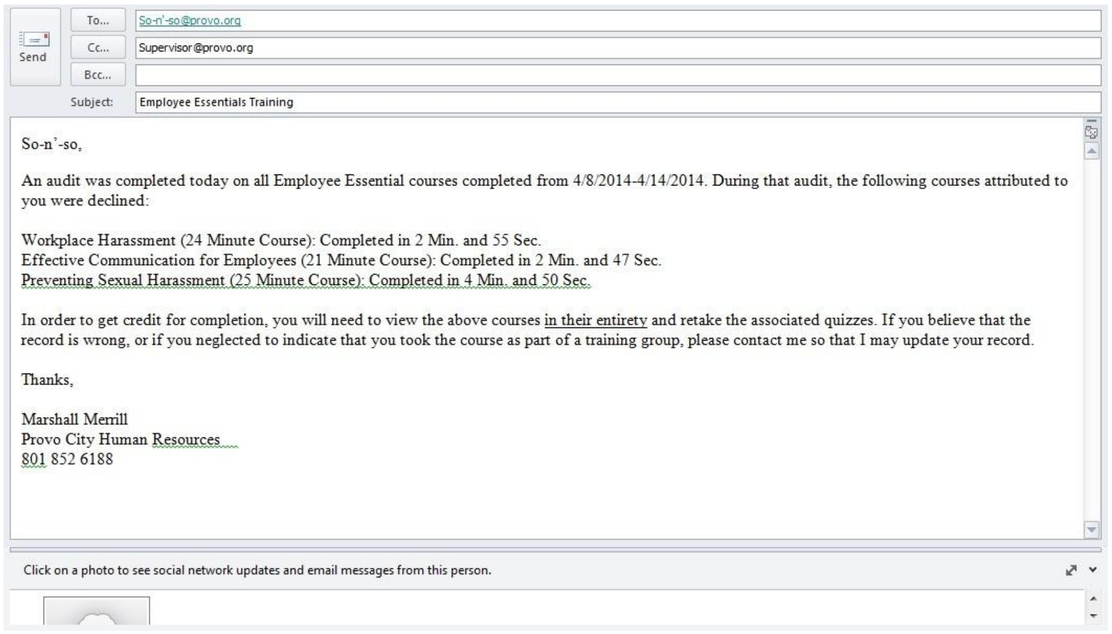
I generate a rejection email for unacceptable courses using the Sub "CreateTrainingEmail." I begin by selecting any one of the courses taken by the employee who will be receiving the email. When I run the procedure it starts at the first course taken by that employee and simply checks to see if it's highlighted in red. If it is, the course title is assigned to the Course variable. I then use a select case to determine how long that course should take as well as the appropriate way to abbreviate the course name for easier readability. I then assign the abbreviated course name and the correct course length to a string variable called CourseAndCL. I then check to make sure the course the code is looking at is still one completed by the employee whose name we started with. It then finds the actual amount of time spent on the course in the current row and assigns that with some additional text to a string variable called ActualTime. It then adds a break for a new line, CourseAndCL and Actual Time to the variable Courses. Each time through the loop all rejected courses are added on a new row to Courses along with a message. In the end it looks like this:

The code then creates an email in Outlook (Dim objMail As MailItem, Set ObjMail = olApp.CreateItem(olMailItem). The body of the mail item is formatted in HTML. It starts with the Employees name and a comma, breaks, contains a message which includes the timeframe in which the course was completed and then inserts the Courses variable. It then ends with a brief conclusion and my contact information.

The "To" field is filled by looking at the email address column in the training data.

The ".CC" field is filled, when possible, by looking at the employee's title. There are a few groups of employees that tend to cheat on their training a lot and they all have one of a handful of supervisors. If it's one of these employees the system will automatically insert the supervisor's email address, otherwise it will leave it blank and I will have to look it up somewhere else.

The email is displayed so I can double check it and then send it. In the end it looks something like this:



To: So-n'-so@provo.org
Cc: Supervisor@provo.org
Subject: Employee Essentials Training

So-n'-so,

An audit was completed today on all Employee Essential courses completed from 4/8/2014-4/14/2014. During that audit, the following courses attributed to you were declined:

Workplace Harassment (24 Minute Course): Completed in 2 Min. and 55 Sec.
Effective Communication for Employees (21 Minute Course): Completed in 2 Min. and 47 Sec.
Preventing Sexual Harassment (25 Minute Course): Completed in 4 Min. and 50 Sec.

In order to get credit for completion, you will need to view the above courses in their entirety and retake the associated quizzes. If you believe that the record is wrong, or if you neglected to indicate that you took the course as part of a training group, please contact me so that I may update your record.

Thanks,

Marshall Merrill
Provo City Human Resources
801 852 6188

Click on a photo to see social network updates and email messages from this person.

Tracking Training

As I mentioned, previously we did not have a good way to check who has and has not completed all of their courses in the last three years. The data gets stored in large database of all training ever completed by city employees. I created a query in PeopleSoft which pulls out just these “Employee Essentials” courses and downloads a long list of all Employee Essentials courses ever completed, but this data was still very difficult to look through and use. I created a Sub that looks at this data and returns a list of employees who have not completed all of their training as well as a list of their outstanding courses.

First my code sorts this data. It sorts it by employee ID, and then by course. It then selects the first employee in the list and begins a loop. It goes through each of the courses under that employee’s name and checks to see if the course was completed within 3 years of today and makes sure it is not a duplicate of the previous course (Sometimes the data I download from PeopleSoft contains duplicates). If the course meets these criteria it is added to an array called ValidCourses. Otherwise, that item in the array is left with a zero-length string.

ValidCourses is then printed to an empty column in the worksheet, excluding blanks. That column is then counted. The count is assigned to the integer variable CountOfName. Count of name is used for several things. The first is to decide if this person is current on their training or not. It also indicates how long the JoinArray will be, it is later used to determine how far to offset to get to the next name in the list.

If CountOfName is less than 10, a few things happen:

- The array NameRangeCount becomes one item bigger and the number in CountOfName gets added to it
- The array NameRange is redimmed one item bigger and the Employee ID number is added to it
- The array NameRangeCourses gets one bigger and all courses that are NOT completed get added to it. This uses the function COURSESfunc which will be discussed later.

If CountOfName is not less than 10 no data is added to any of these three arrays.

Next the code moves to the next person in the list by offsetting by the value in CountOfName. This works unless the employee has completed zero courses, in which case this would result in an endless loop. If CoutOfName = 0 then the code offsets by 1.

Once this loop runs for all employees we have three arrays: NameRange, NameRangeCount, and NameRangeCourses. These three aligned with each other such that the index of each one is the same as the associated data in the other two arrays.

The code creates a new worksheet and writes the values in each of the three arrays to three different columns on the new sheet.

The code then uses R1C1 formulas in the new sheet to reference the employee id number and bring over data including each employee's name, department, title, Employment Status and Hire Date.

The code then sorts the data by department and then supervisor and does a little formatting.

It then goes through and checks which employees with incomplete training records were hired more than 3 months ago and highlights them in orange (red seemed to mean and yellow seemed to passive).

At this point the code goes through and deletes all seasonal, on-call, or contract employees. They are not required to take these courses. Yes, they could have been taken out earlier or even not included in the original query, but we are sometimes interested in seeing which of these employees have taken the training and it is easy to just put a break point in at the end of the code and run it.

Finally the code adds appropriate headers.

COURSESfunc Function

This function is used to input the courses an employee has taken and return the courses they still need to take.

First it uses the Split function to create the array SPlitCoursesTaken which contains all of the courses that were passed in a single string.

Then it does a select case for each of the courses in this array and changes the name of the course to a more understandable abbreviated name.

Then it joins them all back up in the string variable ReJoined.

The array RealCourses contains the abbreviated names for all 10 courses. The code loops through RealCourses and uses InStr to check for each RealCourses within the string ReJoined. If it is found, it moves to the next RealCourses. If it is not found, that RealCourses is added to the array FinalCourses which is made one index larger to accommodate the new item.

Finally, FinalCourses is joined into one string which is what COURSESfunc returns.