**The Task:**

We have an email list for a Yoga studio that has over a couple thousand entries. To comply with the CAN-SPAM Act, we need to ensure we are not emailing people who unsubscribed. This can lead to legal issues if not dealt with. We are provided two different documents from the client, (1) a master roster list, (2) a monthly issued unsubscribe list of all emails that unsubscribed within the given month, and (3) an up-to-date subscribed list independent from the master list. The master list exists merely because the client wants to keep all the emails, which surmounts to about 8,000 emails.

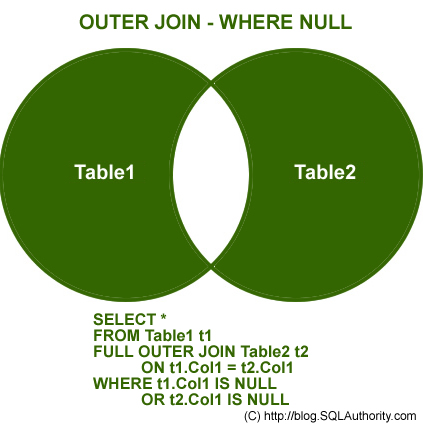
Prior to this, the client spent hours each bi-annual period reviewing the subscribed list they created with the unsubscribe list, provided by MailChimp, to find and remove emails one by one. The client had at one point taken an incredible amount of time creating a subscribed list using the master list and a long list of unsubscribed emails that piled up over the years. This new subscribed list is what the client maintains and combs through every 6 months when unsubscribe emails add up. With the line of query code I researched and wrote, I turned this bi-annual nightmare process into something that can be done within 5 minutes. All we want to do is automate the process of combining two separate lists BUT delete what they have in common and still maintain separate categories to ensure the success.

**How I Did it:**

I created two separate excel files, one for the subscribed members (this being the list they maintained by hand over the past few years but had an abundant of unnecessary data) and a unsubscribed list that we took from MailChimp. In SQL, we want to ensure quality data free of errors as the SQL code will run even if errors exist. This involves the skills I learned from my Economics undergrad thesis project where cleaning data was a skill in itself. Also, SQL views each excel file we create as a table. SQL will NOT recognize any additional sheets we make in the Excel files we upload which is why two Excel files are created. Because the client wants to maintain a master list and retain old emails, I went ahead and created these two excel files that we could clean and remove unnecessary data from.

I needed to create a “Cleaned Yoga Email List” to update our email list to ensure we are not sending out emails to those who unsubscribed. Because unsubbed emails will be removed from the list, we only need to maintain the “Cleaned List” until the next period where it will then become the next “subbed list.” We will “JOIN” a list onto another with the criteria we want to set. We set the criteria by using different JOIN commands and the order we list the tables. In this case, I want to use a FULL OUTER JOIN. A Full Outer Join is going to combine both tables. This function will return all matching records from both tables whether the other table matches or not. So, if there are rows in "subbed" that do not have matches in "unsubbed", or if there are rows in "subbed" that do not have matches in "unsubbed", those rows will be listed as well.

With the help of NULL, we will be able to see where the two lists do not match and will be assigned a null value. NULLS will show if the value does not match on both tables. The FULL OUTER JOIN with exclusion function (NULL) will look at both tables, find where matches are in both tables and remove them. The full outer join looks something like this:



Here, we see the middle of the Venn diagram is not filled. We have joined two tables and where they have similarities, we remove it with NULL. By the way, a normal FULL OUTER JOIN without our exclusion NULL will have the area in the center filled in, meaning that we have quite literally mashed two tables together which may cause duplicates to show if the value exists in both tables.

**The Code:**

SELECT \* FROM subbed

FULL OUTER JOIN unsubbed

ON subbed.subs\_email = unsubbed.unsub\_emails

WHERE unsubbed.unsub\_emails IS null

In the next period, I will take the “Cleaned Yoga List”, rename it to “subbed list” and download the unsubscribed list from MailChimp to repeat the process. Why I change the names is because we want to ensure we use the same titles in my lines of code as SQL will obviously not run it as it will try to pull the table that it thinks exists. We will get an error return.

**The Results:**

|  |  |
| --- | --- |
| **subs\_email** | **unsub\_emails** |
| \_garnie@msn.com | NULL |
| 101tristyn@gmail.com | NULL |
| 1114jax@gmail.com | NULL |
| 128juliette@gmail.com | NULL |
| 143trees@gmail.com | NULL |
| … |  |
| zsuzsadobie@aol.com | NULL |
| zuniga751978@gmail.com | NULL |
| zwanine@gmail.com | NULL |
| zzzart@aol.com | NULL |
|  |  |
| NULL | donutzdenove@hotmail.com |
| NULL | mspa2009@gmail.com |
| NULL | coryandbev@gmail.com |
| NULL | chelseatorres34@yahoo.com |
| NULL | daolicht@yahoo.com |
| NULL | jslockwood@gmail.com |
| NULL | tweetpiyo@yahoo.com |
| NULL | irene.flores300@gmail.com |

Now this is one table but with two different columns. The NULL value shows that value does not exist on the other table. This is good, because the FULL OUTER JOIN we did has already deleted the duplicates. I wanted duplicates because we want to combine two tables BUT delete what they have in common and still maintain separate categories to ensure the success.