

## Kimball Design Tip #12: Accurate Counting With A Dimensional Supplement

By Warren Thornthwaite

In the previous design tip we talked about performing accurate counts within a dimension. We can add even more value to these counts when we introduce a separate table with additional attributes that intersects with the dimension table.

We recently loaded a simple example of this kind of "supplemental" table that maps zip code to Media Market Areas (MMA). Our Marketing folks were interested in seeing how our customers break out by MMA compared to the overall population. In other words, we want to know where are we getting strong geographical penetration, and where are we not doing as well. If this supplemental data proves valuable to the organization, we would go ahead and add it into the Customer dimension as an additional attribute. But first, we'll want to do some initial queries to make sure it is worth the effort.

To run these queries, we join the supplemental table to our customer table and do customer counts by MMA. However, we have to be careful because the two sets do not overlap 100%. There are some zip codes in the MMA table with no corresponding customers, and there are some customers whose zip codes have no corresponding MMA. An inner join will undercount both sides of the query. We can use the following two tables to illustrate this:

Media_Market_Areas		Current_Customer	
Zip	MMA	Customer Key	Zip
94025	SF-Oak-SJ	27	94303
94303	SF-Oak-SJ	33	94025
97112	Humboldt	47	24116
98043	Humboldt	53	97112
00142	Gloucester	55	94025

If we'd like to see how many Customers we have by MMA, an inner join gives us the following:

MMA	Count(Customer Key)
Humboldt	1
SF-Oak-SJ	3

The inner join is an "equal" join. Since there is no MMA for zip 24116, the query undercounts our subscriber base giving us 4 customers when we really have 5. We also lose information on the other side of the join because the results don't tell us the MMAs where we have zero penetration (e.g., Gloucester). Re-writing the query with a full outer join gives us the following:

MMA	Count(Customer Key)
NULL	1
Gloucester	0
Humboldt	1
SF-Oak-SJ	3

Now we are counting all 5 of our customers, and we see we have no customers in Gloucester. We could use an IFNULL function on the character column to replace those NULLs with friendlier values, like 'MMA Unknown'. Note that what you count makes a big difference in your results. In our case, we counted Customer\_Key. If we had count(\*), we would have gotten a total of 7 items, since the \* counts rows, and the full results set has 7 rows. If we had counted (MMA\_to\_Zipcode.Zip\_Code), we would have returned a count of 6 because 94025 is counted twice.

You need to be cautious using outer joins because you can easily undermine the logic by putting a constraint on one of the participating tables. A constraint on Customer\_Age < 25 would yield a report with exactly the same structure and headings but reduced counts. If the report is not explicitly labeled, it would be misleading.

We've discovered that combining the case statement with the sum function is a great trick to get counts of various subsets of the full results set from both sides of the outer join in a single pass. Using the data above, we could create a query that gives us total counts for all three areas of the data set. In the Select list you could write

```
Sum(case when Media_Market_Area.Zip IS NULL then 1 else 0 end) AS  
Customer_Count_with_No_MMA,
```

```
Sum(case when count(customer_key) = 0 then 1 else 0 end) as  
MMA_Count_with_No_Customers,
```

```
Sum(case when not(Media_Market_Area.Zip IS NULL or count(customer_key) = 0) then 1  
else 0 end as Count_MMAs_with_Customers)
```

This gives you three columns: the count of customers with no MMAs, the count of MMAs with no customers, and a count where they match. Essentially, the constraints are built into the CASE statement, so they don't limit the results of the outer join.