Data model analysis

In this project I wanted to go over multiple ways you could model data before creating reports in Power BI to see what are the best options in terms of file size, visuals, and refresh performance.

I’ve picked these 3 criteria because these are the main issues I find when developing Power BI reports. It might not be as evident at first, but after the report is in use for a while and the amount of data increases, these 3 criteria can become problematic and need to be addressed.

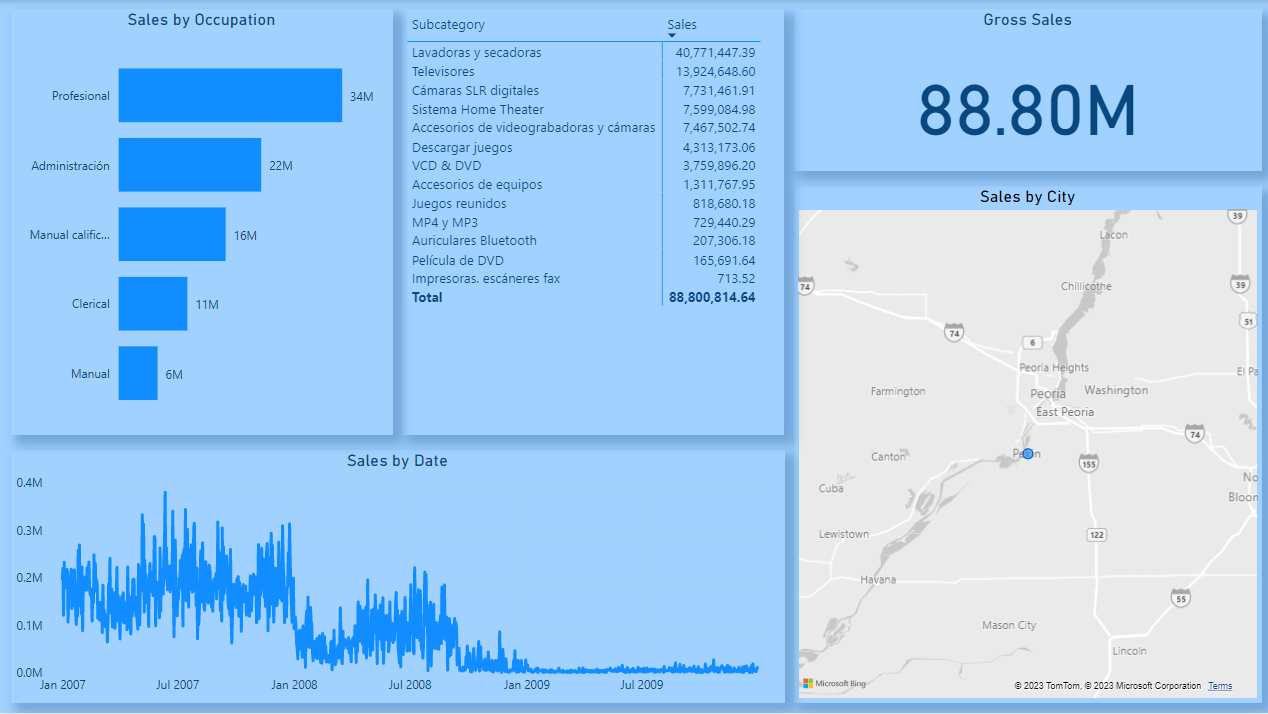
## Dataset

To be able to do this analysis, I found a [dataset](https://www.kaggle.com/datasets/majohidalgo/segunda-versin-de-contoso) that resembles pretty well the messy data found in real life. The data is in a .csv file which adds another level of complication because the Query Folding option from Power BI can’t be used on CSV files.

## Data model – v1

To establish a benchmark, I imported the data as is in Power BI. Then I made one measure “Total Sales” and created the following 5 visualizations with it:

1. Sales by occupation
2. Sales by date
3. Sales by Product Subcategory
4. Gross Sales
5. Sales by city



Then I went and measure the performance of the visuals by using the “Performance analyzer” function from Power BI and calculated their average. The results are as follows:

1. Sales by occupation – 458 ms
2. Sales by date – 659 ms
3. Sales by Product Subcategory – 737 ms
4. Gross Sales – 412 ms
5. Sales by city – 455 ms
6. Average – 544,2 ms

I used the “Right-click -> Properties” function from Windows to measure the file size. The result was 16 MB.

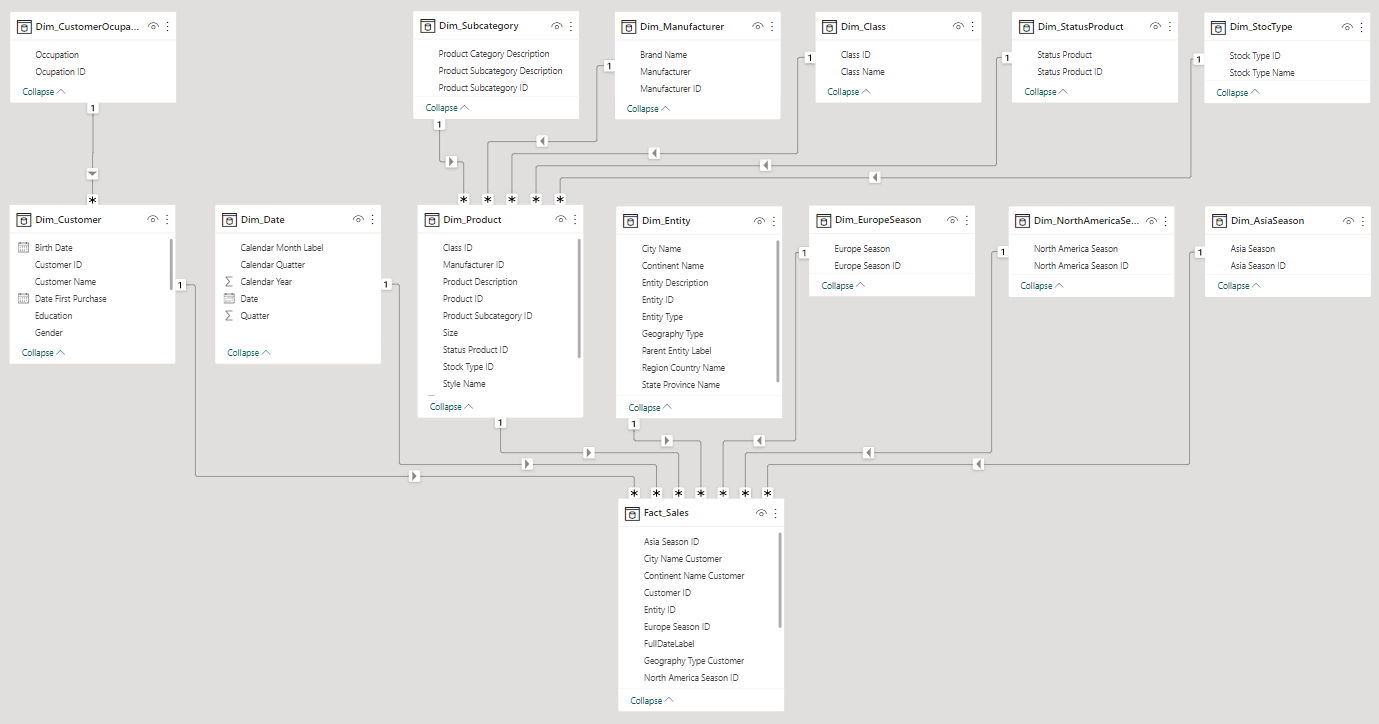
The refresh performance measurement was done in Power BI Service because when you refresh a dataset there, you have the start and end time available and the result was one subtraction away. In this case, the refresh was approximately 1 minute.

With the benchmark set, I have used the same visualizations for the following example. The only thing that changed was the data model. And the first option was to use another feature provided by Power BI.

## Data model – v2

In the second scenario, I wanted to use only Power Query inside Power BI to try to optimize the data model. The idea was to try and break the original file into a Star Schema structure to see if it improves performance. In theory, it does, but sometimes practice tells a different story.

While I was doing all the transformation, I figured out that a Snowflake Schema was best in this scenario and I ended up with the following result:



In the image above you can see a fact table, and 13 dimension tables.

For these results I’ve removed columns and duplicated values, I’ve added index columns and merged tables between them to add index columns from one table to another.

Results:

Visuals:

1. Sales by occupation – 324 ms
2. Sales by date – 550 ms
3. Sales by Product Subcategory – 681 ms
4. Gross Sales – 287 ms
5. Sales by city – 473 ms
6. Average – 463 ms

Files Size: 13.1 MB

Refresh Performance: 2 minutes

When comparing with the first iteration, there are significant results in terms of visuals performance overall and file size. On average, visuals were 15% faster in v2 vs v1. The size of the .pbix file decreased by 18%. The one aspect that didn’t improve, rather doubled in time, was the refresh performance. This is not a problem because Power BI has other functions that help with the refresh, like incremental refresh that can help with the refresh performance. But as it stands now, refresh is worse in this case without any other Power BI functions implemented.

The results were not satisfying enough so I wanted to try this next option.

## Data model – v3

The third iteration is similar to the second in that it is the same Snowflake Schema, and it still uses Power Query for some transformation. The difference is that I’ve imported the .csv file in a SQL Server database and used SQL to pass as much transformation as possible to the SQL Server rather than Power BI.

Since I didn’t do any transformation of the original data in SQL Server, I still had to add index columns and merge tables in Power Query, everything else was pushed to the data source using SQL.

Results:

Visuals:

1. Sales by occupation – 279 ms
2. Sales by date – 417 ms
3. Sales by Product Subcategory – 464 ms
4. Gross Sales – 248 ms
5. Sales by city – 639 ms
6. Average – 409,4 ms

Files Size: 13.1 MB

Refresh Performance: 8 minutes

In this situation, the only thing that was improved compared to v2 is the visual performance which was 11% faster. Compared to v1, visuals from v3 were 25% faster. But when it comes to file size, nothing changes between the second version and the third. And when it comes to refresh performance, things took 4 times as long as the second iteration and 8 times more than the first one. Again, this can be mitigated using incremental refresh if needed.

Since there weren’t improvements on all criteria of comparison, I wanted to try another thing.

## Data model – v4

This time I went all SQL and scripted the result only. I ended up with 5 SQL queries from Power BI which give me all the data needed for my 5 visuals. The results were improved significantly, but this solution can have drawbacks that we will discuss shortly. Until then, let’s go over the results first.

Results:

Visuals:

1. Sales by occupation – 231 ms
2. Sales by date – 361 ms
3. Sales by Product Subcategory – 407 ms
4. Gross Sales – 205 ms
5. Sales by city – 274 ms
6. Average – 295,6 ms

Files Size: 0,2 MB

Refresh Performance: 0,1 minutes

To better show the results of this iteration compared with the previous ones, I’ve made the dashboard below. In it, you can see that this method was almost twice as good as the previous scenarios.



At first glance, this sounds awesome, but it has some limitations that will change the actual results. First of all, in the previous situations, I could have done other visualizations with the data already imported, but in this situation, I need to get more data to handle other visualization. And second, it’s not as easy as the first 2 options that rely entirely on Power BI knowledge, it requires SQL knowledge as well.

## Conclusion

In conclusion, some ways to increase the performance of a power bi report include transformation in Power Query and SQL scripting. Which, depending on a project’s requirements might not default to the fourth option which was the better-performing option in this case. In some cases, if there is no option for an SQL Server or equivalent, the first or second option might be better. In other cases, the 3rd option might be better. The purpose of this analysis was to see if performance improved from one iteration to another.