PROJECT DOCUMENTATION

Middle-earth Strategy Battle Game Interactive Dashboard
by Marcin Czerkas

DATE: 17.12.2024

SKILLS & TOOLS COVERED: SQL, database design, Power BI, DAX, data visualization

Dear Reader, thank you for checking out my project!

In this document I am going to introduce you to the project by providing you with a general description as well as all details and technicalities.

I divided this text into **six parts**:

1. **PROJECT OVERVIEW**
   a general introduction to the project and key methodologies I used
2. **CONCEPT**
   my assumptions from before starting the work
3. **DATA**
   a quick look at the dataset
4. **DETAILS**
   the core of the project and a detailed description of what I did and how I did it
5. **RESULTS ANALYSIS**
   key insights emerging from the data
6. **GROWTH AND IMPROVEMENT**
   thoughts on what could be improved in the future and ideas on a possible business use of the project

Without further ado, let's dive into the first section.

1. Project Overview

Welcome to the Middle-earth Strategy Battle Game! It is a tabletop game based on J.R.R. Tolkien's works and played with miniature models collected by the players. Me and 4 of my friends formed a little community and have been playing the game for more than 10 years.

The idea behind this project is to collect all possible data on how we play the game, when we play it, what are the results, who wins more often etc., store it in one database and deep dive into it by building an interactive dashboard.

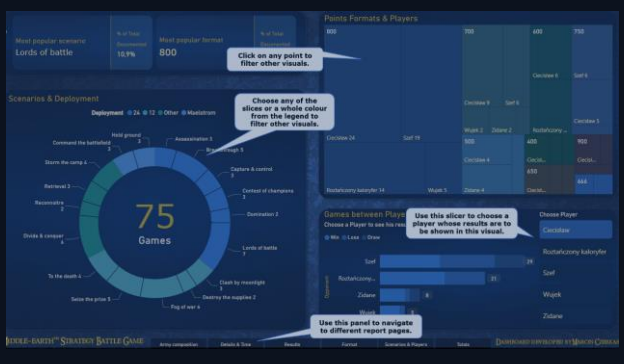
This is just a glance of the final effect:

To achieve this goal, I used a couple of tools such as PostgreSQL and Power BI (more details on tools and methodology in the section 4.1.). Alongside with some built-in features, the dashboard also contains customized visuals set up by me. Here are some examples:

I added a slicer to switch between measures we want to see. The title adapts to the slicer settings.

Page navigation inside the visual with a possibility to choose the number of displayed items.

Slicer panel.



Info pane to help users make better use of the dashboard.

These are just a couple of best features. However, the Power BI dashboard is merely the tip of the iceberg in the whole project. If you are interested to see more details, check out the next sections!

2. Concept

Before we take a closer look at the data, let us stop for a moment and answer the following questions:

What exactly did I want to build and why? How can I achieve my goals?

What should I take into consideration and what are the limitations?

The main idea was to build an interactive dashboard in Power BI which would allow the users to learn more about details related to the game and our local community, for example: how different players build their armies in the game, what are the patterns in their behaviour, which army won the biggest number of games. In order to do this, I deep dived into the tools I wanted to use and learned new things or refreshed my knowledge on DAX, Power BI, SQL and data visualisation rules.

I decided to work on this project for two reasons. Firstly, because I am really into the game (call me a nerd if you like!). Secondly, because I wanted to use this opportunity to showcase my skills in data analysis. I thought a combination of hobby and more work-oriented passion would make a perfect project. At first glance, this project has a very limited business due to its niche topic. Additionally, the dataset is quite small (there are not millions of rows there). I will try to face these objections in the last section of this documentation.

3. Data

Before I introduce you to the specificities of my dataset, let me describe the process behind designing and populating the database.

At first, I needed to design and create the whole database. In order to quickly and effectively design the database schema, I used a free online tool called *Quick Database Diagrams* (www.quickdatabasediagrams.com).

```
Fact_Battles as Bat
---
BattleID int PK
ReportName text NULL
Date date
LocationID int FK >- Loc.LocationID
ScenarioID int FK >- Sc.ScenarioID
Duration float NULL
ReportPages int NULL
Format int NULL

Dim_Location as Loc
---
LocationID int PK
City text
Host text

Dim_Scenario as Sc
---
ScenarioID int PK
Pool text
Scenario text UNIQUE
EndCondition text
Deployment text
```

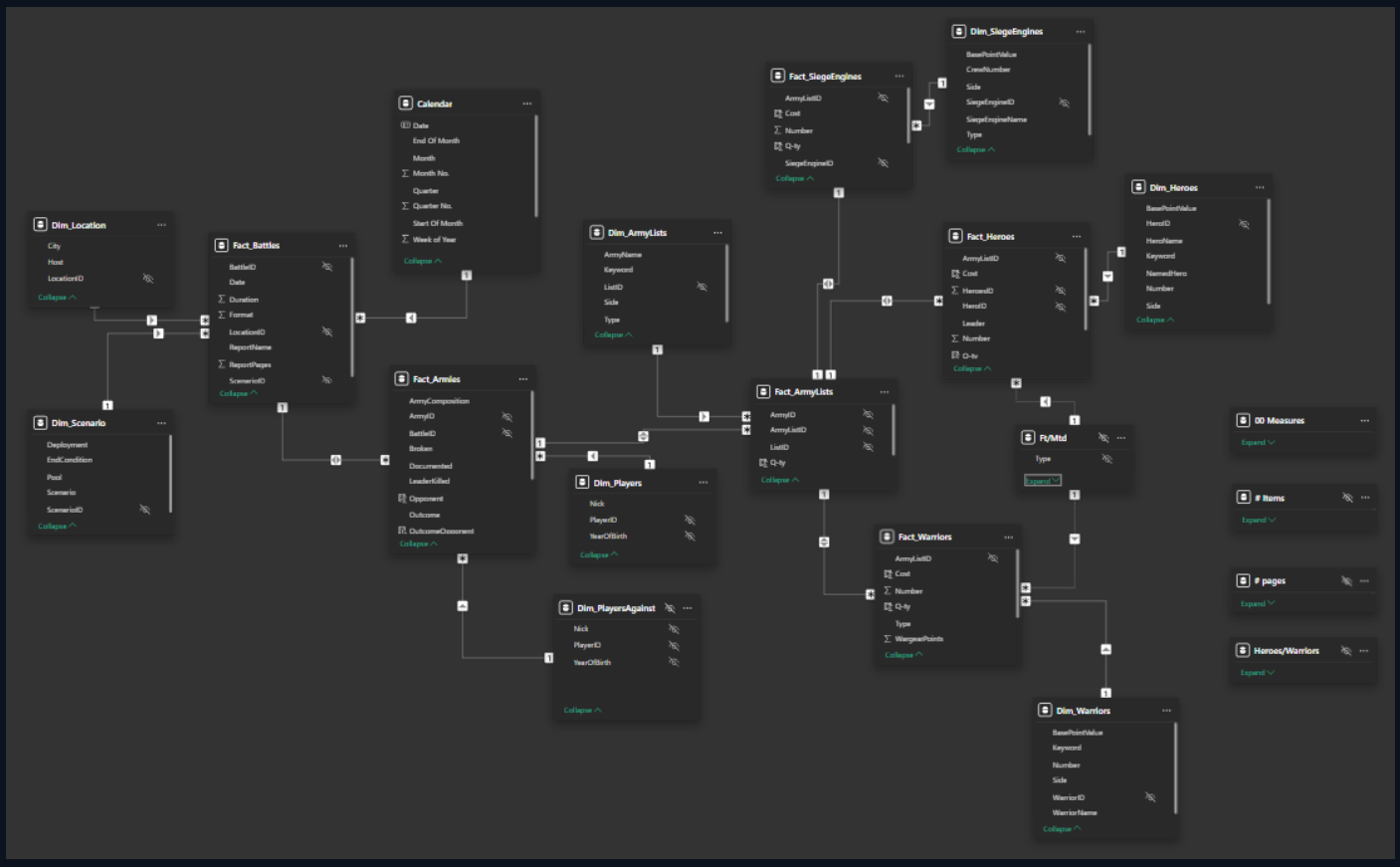
Here I was able to create tables, define primary and foreign keys and choose data types in an easy and transparent way - just by typing. Once it was done, I was ready to move on to PostgreSQL. I exported my database design from QuickDBD to SQL and made occasional adjustments here and there to make sure that the final effect meets my expectations:

```
CREATE TABLE "Fact_Battles" (
    "BattleID" int   NOT NULL,
    "ReportName" text   NULL,
    "Date" date   NOT NULL,
    "LocationID" int   NOT NULL,
    "ScenarioID" int   NOT NULL,
    "Duration" float   NULL,
    "ReportPages" int   NULL,
    "Format" int   NULL,
    CONSTRAINT "pk_Fact_Battles" PRIMARY KEY (
        "BattleID"
    )
);
```

At this point, I could start collecting the data and populating the database. During this process I decided to use Excel. I created different sheets each representing one table. And now came the most tiresome part of the project - the data had not been stored before in any structured way. I needed to look for archival chats between players, Excel files with army lists in different formats etc. I gathered all this data and assigned it to the appropriate tables. Once it was done, I saved it as separate CSV files and loaded it to my PostgreSQL database.

Now it is time to let you explore the data more precisely.

As you can see in the picture below, I decided to go for a snowflake schema. I wanted my database to be as normalised as possible and at the same time I needed to keep it at the appropriate level of granularity. The process of army building in the last edition of Middle-earth SBG was pretty complicated, so I ended up with 11 tables (5 dimensions and 6 facts):



Starting from left, the first fact table is Fact_Battles. It gathers all data referring to our games on the lowest level of granularity where one row represents one game. It is connected to two dimensions: Dim_Scenario, Dim_Location.

The primary key of Fact_Battles (BattleID) is also used as foreign key in the next fact table - Fact_Armies. This one breaks the data into the opposing forces, as each game/battle needs two armies to take place. Fact_Armies has one dimension associated with it - Dim_Players, as each army is led by one player.

Fact_ArmyLists is related to Fact_Armies in a similar way. Each army can be composed of one or more army lists (which can be understood as factions), so that players can build their forces either as single armies or alliances of two or more factions. Another additional information stored in this table is the number of warbands: each army can be built of several warbands (e.g. there can be an alliance of Gondor and Rohan where two warbands are taken from the Gondor army list and one from the Rohan one). This is finally the level of granularity of the individual factions and for this reason Fact_ArmyLists has one related dimension table: Dim_ArmyLists.

We have reached the central point of our snowflake. Now the "path" splits into three directions: Fact_Heroes, Fact_Warriors and Fact_SiegeEngines. These are the three types of units that can be used to build warbands. They have their respective dimensions as well: Dim_Heroes, Dim_Warriors and Dim_SiegeEngines.

As a result, my database meets the Third Normal Form (3NF).

One final remark: not all data regarding our games was available to me. Some armies are flagged as "undocumented" because they lack some important information (e.g. the exact number of warriors) and for some games we do not have information on scenario or points format. All these things have been taken into account by me later on, while analysing the data in DAX.

If you want to learn more about the tables and the columns they are composed of, feel free to open the PBIX file with the dashboard and navigate to the modelling tab.

4. Details

You already know how I built the database. Now it is high time to tell you more about the analysis I performed on this dataset.

4.1. Import to Power BI

While connecting to my local PostgreSQL server I chose the option *Import*. The reason why I did not use *DirectQuery* is that there was no need to refresh the query that often.

In Power BI there was still some work to do. I had to make sure that all relationships had been detected properly and in some instances I needed to change the relationship direction to bidirectional (between the fact tables). Obviously, I had to add a calendar table which I created using Power Query:

```
let
    Today = Date.From(DateTime.LocalNow()),
    StartDate = #date(2020, 1, 1),
    EndDate = Date.EndOfYear(Today),
    Calendar = List.Generate(
  () => StartDate,
  each _ <= EndDate,
  each Date.AddDays(_,1)
),
    ConvertedIntoTable = Table.RenameColumns(Table.FromList(Calendar, Splitter.SplitByNothing()),{{"Column1", "Date"}}),
    // Remember that 0 is Sunday, 6 is Saturday
    WeekdayNo = Table.AddColumn(ConvertedIntoTable, "Weekday No.", each Date.DayOfWeek([Date])),
    Weekday = Table.AddColumn(WeekdayNo, "Weekday", each Date.DayOfWeekName([Date], "en-US")),
    WeekOfYear = Table.AddColumn(Weekday, "Week of Year", each Date.WeekOfYear([Date])),
    MonthNo = Table.AddColumn(WeekOfYear, "Month No.", each Date.Month([Date])),
    Month = Table.AddColumn(MonthNo, "Month", each Date.MonthName([Date], "en-US")),
    QuarterNo = Table.AddColumn(Month, "Quarter No.", each Date.QuarterOfYear([Date])),
    Quarter = Table.AddColumn(QuarterNo, "Quarter", each "Q"&Text.From([#"Quarter No."])),
    Year = Table.AddColumn(Quarter, "Year", each Date.Year([Date])),
    SOM = Table.AddColumn(Year, "Start Of Month", each Date.StartOfMonth([Date])),
    EOM = Table.AddColumn(SOM, "End Of Month", each Date.EndOfMonth([Date])),
    #"Changed Type" = Table.TransformColumnTypes(EOM,{{"Date", type date}, {"Weekday No.", Int64.Type}, {"Weekday", type text},
      {"Week of Year", Int64.Type}, {"Month No.", Int64.Type}, {"Month", type text}, {"Quarter No.", Int64.Type}, {"Quarter", type
      text}, {"Year", Int64.Type}, {"Start Of Month", type date}, {"End Of Month", type date}})
in
    #"Changed Type"
```

4.2. Dashboard setup

At this stage began the greatest fun. I was about to plan the data visualisation in my dashboard. I decided to split it into six pages and one additional homepage to be sure I will have enough space to make all visuals as clear as they should be.

First of all, I prepared the layout of the graphs that I had in mind. Then, I used PowerPoint to edit the background of each report page. I saved the background files in the SVG format and uploaded them to my dashboard.

The next step was choosing the right type of visuals and actually setting them up. Some of them were very straightforward but others required much more DAX and sometimes even workarounds.

The best way to check it out is to simply open the PBIX file attached to this project (Middle-earth SBG Project Dashboard). Feel free to play with the dashboard as you like and even look under the hood. Nevertheless, I will use the opportunity to present a couple of visuals I am most proud of.

4.2.1. Page navigation inside visual with ranking

In this visual I decided to change the default layout a little bit by giving the dashboard user a possibility to choose how many items should be displayed at the same. Then, I added a small slicer to navigate between pages.

Additionally, I changed the positioning of the labels. To achieve this effect, I needed to add a dummy measure to act as a placeholder for the rank labels.

```
Rank Warriors =
RANKX(
    ALL(Dim_Warriors[WarriorName]),
    [Warriors Count],,,Dense)
```

```
Warrior Filter =
VAR _Page = [# pages Value]
VAR _Number_of_Warriors = [# Items Value]
VAR _Warrior_Rank = [Rank Warriors]
VAR _Warriors_Filtered_Tbl =
    FILTER(
        ALLSELECTED(Dim_Warriors[WarriorName]),
        _Warrior_Rank > (_Page - 1) * _Number_of_Warriors
        && _Warrior_Rank <= (_Page) * _Number_of_Warriors)
VAR _Warriors_Shown =
    IF(
        SELECTEDVALUE(Dim_Warriors[WarriorName]) IN _Warriors_Filtered_Tbl,
        1,
        0)
RETURN
    _Warriors_Shown
```

```
# Items = GENERATESERIES(5, 12, 1)
```

Items
5
6
7
8
9
10
11
12

```
# pages = GENERATESERIES(1, 4, 1)
```

pages
1
2
3
4

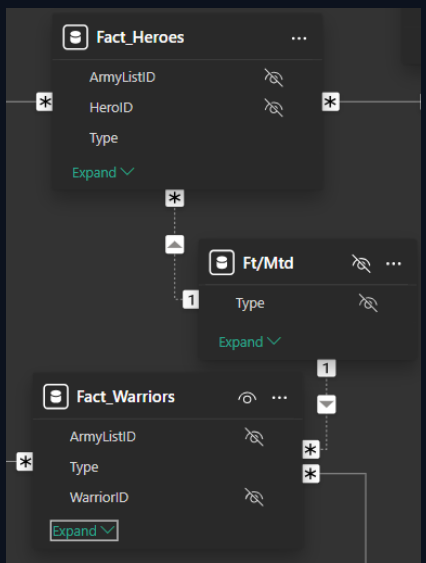
The user can input their own value of displayed items by typing in the top right corner of the visual. I turned this box to a slicer that uses the parameter table # Items. Similarly, the page navigation uses the parameter table # pages. Both tables are disconnected from my model and serve only for this purpose.

4.2.2. Category based on user selection (measure)

This visual gives the user a possibility to switch between two categories from two different fact tables.

To achieve this effect I connected Fact_Heroes and Fact_Warriors by a table containing only one column with two values: “Ft” and “Mtd” in order to have one unified category that would act as a dimension. Additionally, I wrote a measure that captures the user selection from the slicer.


```
Selected Values =
SWITCH(
    TRUE(),
    [User Selection] = "Heroes", CALCULATE([Documented Heroes ALL], USERELATIONSHIP(Fact_Heroes[Type], 'Ft/Mtd'[Type])),
    [User Selection] = "Warriors", CALCULATE([Documented Warriors ALL], USERELATIONSHIP(Fact_Warriors[Type], 'Ft/Mtd'[Type])),
    "Choose between Heroes and Warriors")
```



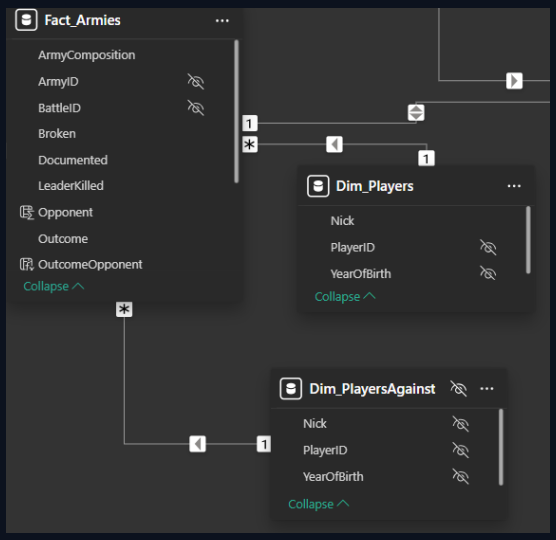
4.2.3. Duplicated table to show mirrored players

Another challenge was to show each player's opponent in a specific game. What I did first was adding a calculated column in Fact_Armies, which is the granularity level of the players. The DAX calculation goes one level of granularity down, to Fact_Battles, and captures the other PlayerID assigned to the same BattleID (foreign key).


```
Opponent =
    VAR CurrentBattleID = Fact_Armies[BattleID]
    VAR CurrentPlayer = Fact_Armies[PlayerID]
    RETURN
        CALCULATE(
            MAX(Fact_Armies[PlayerID]),
            FILTER(
                Fact_Armies,
                Fact_Armies[BattleID] = CurrentBattleID && Fact_Armies[PlayerID] <> CurrentPlayer))
```

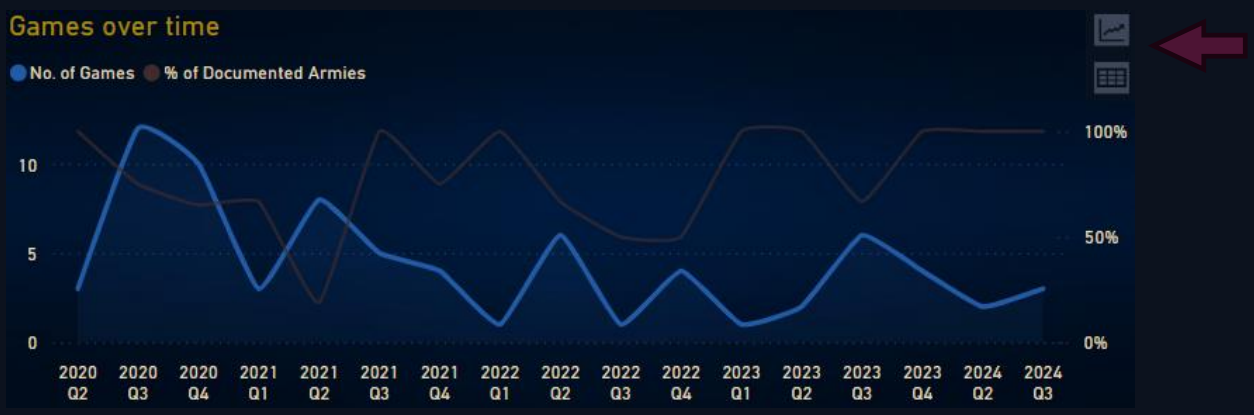
Then, I duplicated Dim_Players to display the opponents of the players selected in the slicer.

```
Dim_PlayersAgainst = ALL(Dim_Players)
```



4.2.4. Timeline vs matrix – swapping between visualization types

In this particular example I wanted to give the user a possibility to choose between two visualization types: a line chart and a matrix. I set it up by creating these two visualizations in the same spot and adding two buttons connected to bookmarks: one hides the timeline and shows the matrix, the other one hides the matrix and shows the timeline.



Year	Games	YoY	YoY Growth	Cumulative	Average Duration	Documented Armies (%)
⊞ 2020	25			25	4,1	74,0%
⊞ 2021	20	25	-20%	45	2,1	57,5%
⊟ 2022	12	20	-40%	57	2,4	62,5%
Q1	1	3	-67%	46	3,0	100,0%
Q2	6	8	-25%	52	2,0	66,7%
Q3	1	5	-80%	53	2,0	50,0%
Q4	4	4	0%	57	2,5	50,0%
⊟ 2023	13	12	8%	70	2,8	84,6%
Q1	1	1	0%	58	3,5	100,0%
Q2	2	6	-67%	60	3,3	100,0%
Total	75	70	7%	75	3,2	71,3%

The matrix uses a couple of additional measures which I am presenting below:

```
Cumulative No of Battles =
CALCULATE(
    COUNTROWS(Fact_Battles),
    FILTER(
        ALL('Calendar'[Date]),
        'Calendar'[Date] <= MAX('Calendar'[Date])))
```

```
YoY Battles =
CALCULATE(
    [No of Battles],
    SAMEPERIODLASTYEAR('Calendar'[Date]))
```

```
YoY Growth % =
DIVIDE(
    ([No of Battles] - [YoY Battles]),
    [YoY Battles])
```

4.2.5. Other DAX measures

As my dashboard may seem quite complicated to someone who does not play Middle-earth SBG, I tried to add as many descriptive visuals as possible. This includes cards and smart narratives which I customized by myself. All these visuals use the same type of measures (showing e.g. the concatenated year and quarter with highest number of games or the warrior with the highest model count). Here is an example of such measure that is used to return the most played scenario:

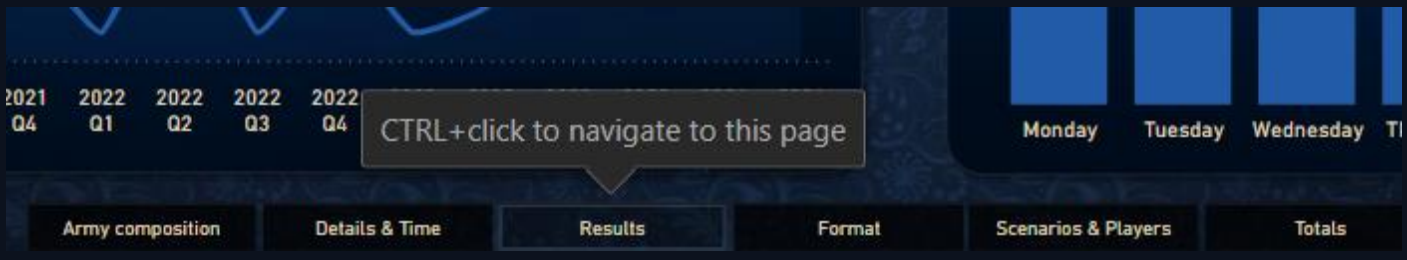
```
Highest No of Games Scenario =
VAR Fact_BattlesByScenario =
    ADDCOLUMNS(
        SUMMARIZE(
            FILTER(Fact_Battles, NOT(ISBLANK(Fact_Battles[ScenarioID]))),
            Dim_Scenario[Scenario]),
        "BattleCount", CALCULATE(COUNT(Fact_Battles[BattleID]), Fact_Battles[ScenarioID]<>BLANK()))
VAR MaxBattleCount =
    MAXX(Fact_BattlesByScenario, [BattleCount])
VAR Result =
    TOPN(
        1,
        Fact_BattlesByScenario,
        [BattleCount],
        DESC)
RETURN
IF(
    COUNTROWS(Result) > 1,
    "(Multiple values)",
    CONCATENATEX(
        Result,
        Dim_Scenario[Scenario],
        ", "))
```

4.3. Additional features

Apart from the visuals, I added some extra features to make the dashboard a little more user-friendly. One of them is the slicer panel which I built using bookmarks (See).

Another similar functionality is the info pane. I created it in PowerPoint, as it was the case with the background. It serves to educate the report users about the possibilities within the tool and can be opened by clicking on an icon on the left side, similarly to the slicer panel (See).

Last but not least is the page navigation. Although there is no need to create a special page navigation system in Power BI Desktop, I decided to add it anyway. This way, my dashboard is better prepared to be published to the online service if needed:



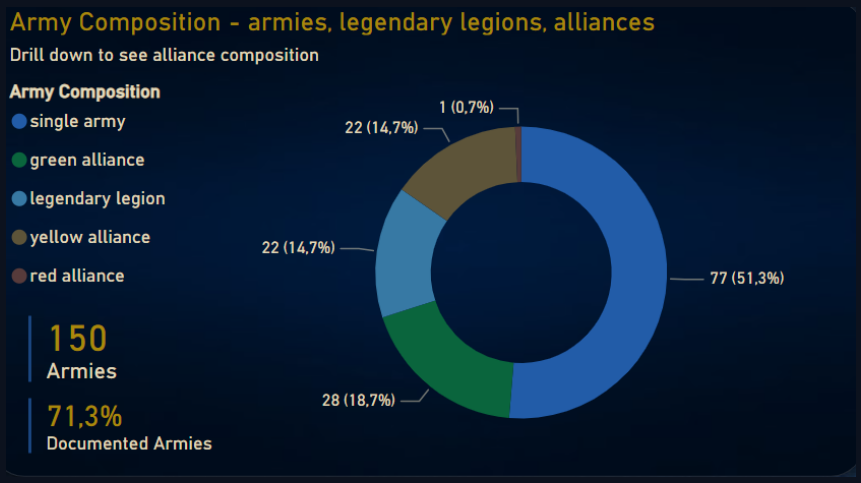
5. Results analysis

Knowing how to use the tools is already an asset, isn't it? In my opinion, however, it still does not make someone a good data analyst. This is why I added this section: a small summary of my analysis with key insights emerging from the data and my critical review of these results.

To keep things concise, I prepared a list of 5 bullet points containing my main findings:

- The prevalent army type is single army (over 50% of all armies).
- The game is played less and less over time in our local community.
- On average, the expenses in terms of points in an army are divided equally to heroes and warriors (with heroes having a slight prevalence).
- The most popular points format is 800 pts. Maelstrom is the least popular deployment type in scenarios (only 10% of games).
- The most frequently used models are orcs.

Now, let me guide you through each of these points.



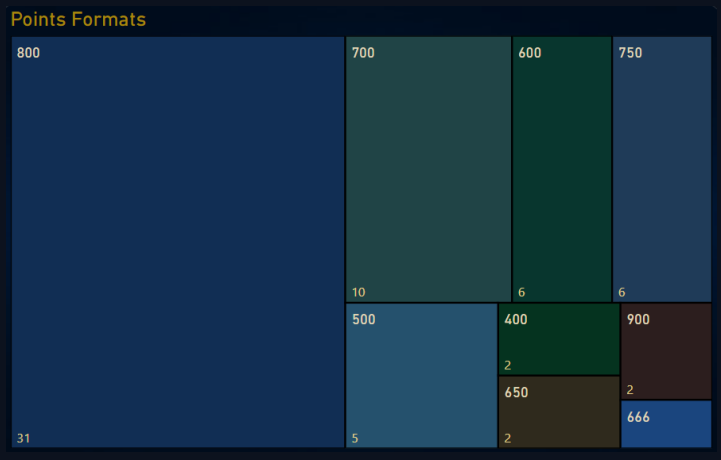
On the left you can see the distribution of army types in terms of army composition. The best documented category is green alliance (82,1% documented armies). Legendary legions, which are the only available option in the new edition of the game (after December 2024) were not very popular (only 14,7% of all armies).

Year	Games	YoY	YoY Growth	Cumulative	Average Duration	Documented Armies (%)
2020	25			25	4,1	74,0%
2021	20	25	-20%	45	2,1	57,5%
2022	12	20	-40%	57	2,4	62,5%
2023	13	12	8%	70	2,8	84,6%
2024	5	13	-62%	75	2,7	100,0%

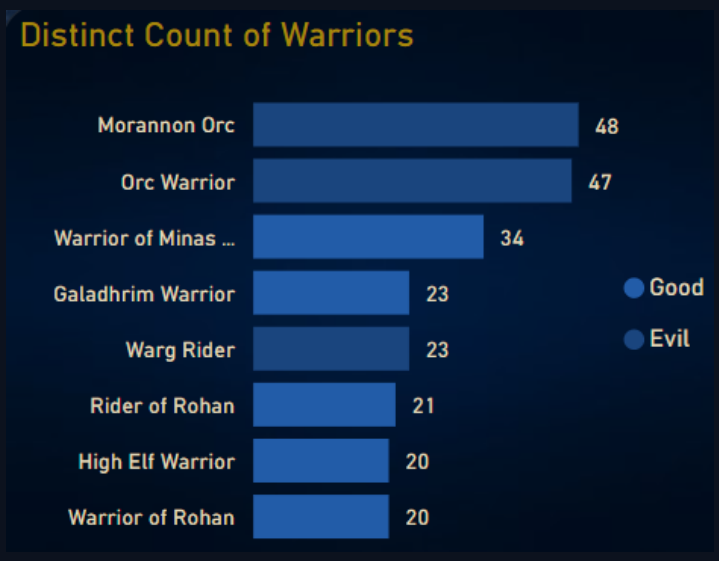
With time passing, our local community plays less games yearly. If we break these numbers by players, we can see that two of them actually stopped playing (Wujek in 2022 and Zidane in 2023).

Format	Average Heroes Share (Pt)	Average Warriors Share (Pt)	Average Siege Engine Share (Pt)
400	45,3%	54,7%	
500	50,2%	49,8%	
600	47,7%	50,6%	1,7%
650	57,4%	42,6%	
666	50,3%	37,3%	12,4%
700	49,9%	48,2%	1,8%
750	45,9%	52,6%	1,5%
800	54,7%	43,2%	2,1%
900	51,2%	43,0%	5,8%
Total	**52,4%**	**45,4%**	**2,2%**

As you may notice, the average heroes and warriors share in the total point expenses is pretty similar across all point formats. The one exception is the siege engine share which is prone to extremes - they are quite expensive but are not included in every army. You can see that they made up 12,4% of expenses in the 666 point format but there is only one game (two armies) in this format.



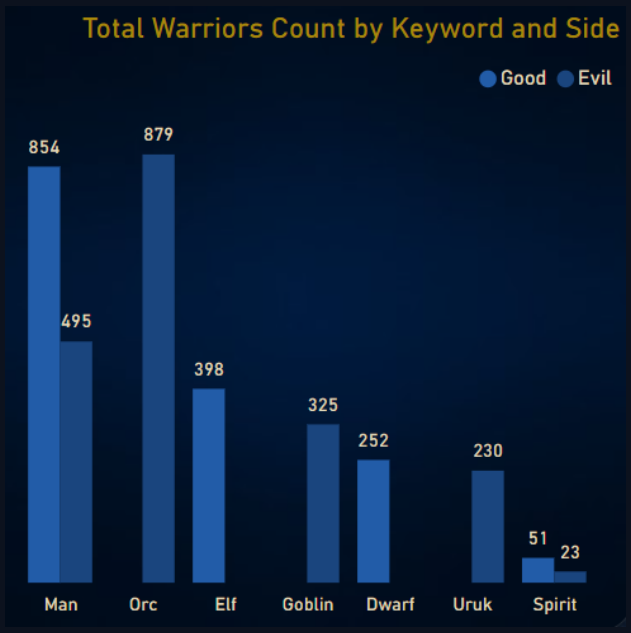
The 800 pts format was definitely the prevalent format. In terms of scenarios, they are split fairly evenly. The most popular one was "Lord of battle" (7 games). There is, however, one more observation emerging from this analysis: seemingly, the players avoided scenarios with the "Maelstrom" deployment type (which forces them to split their warbands and deploy them in different parts of the table, which is, understandably, very inconvenient).



Out of 4235 models that appeared in all games, 3588 were warriors. Two types of orcs were the most popular choices - Morannon Orcs (appear in 48 games) and Orc Warriors (47 games). These two groups reach a total model count of 722, which is roughly 20% of all warriors.

A simplified split of warriors by keyword (race) and side (Good/Evil) can also be seen on the chart to the right.

Because orcs appear only on the Evil side, Men are the most popular race for both sides combined (1349 warriors in total).



6. Growth and improvement

Each project is characterized by one important and unchangeable feature called limited time. There is always a moment when one has to decide it is enough. Of course, in many cases there is still a space for improvement.

What could I improve in my project if I had more time? Which areas would I develop better if I had enough resources?

Well, one big open topic is the process of data collection. In this particular case there was no pre-built database. The one that I created simulates a database of a hypothetical computer game system, where each table is automatically populated based on user actions. As Middle-earth SBG is, of course, not a computer game, we would need a good and unified system to collect and save the data. There are web applications (like Tabletop Admiral: Link) that use their own databases and allow users to build army lists and save them in a specified format. Using this solution, I would just need to store all such files in one place and use Python or Power Automate to extract the data and load it to my database.

Another solution would require much more work. I would have to create my own "app" for army and game documentation (using Power Apps, for example). Then, as above, you would need to create an ETL process responsible for systematically loading data into the database (or directly to Power BI).

Finally, how relevant is my project in terms of a possible business use?

Well, unless your business has at least something to do with selling miniatures for tabletop games, this project as such presents very little value. However, the idea behind it is quite universal and can be applied in different areas. What I tried to showcase is the ability to extract as much data as possible, be accurate in designing databases and be creative in building dashboards. Let us imagine a small company that uses Excel as their only data storage tool - in such a case using my approach could help them to gather, unify and analyze all their data in an efficient and transparent way.

Working on this project has been great fun for me. I could test my skills and keep learning new ones. Can't wait to do more similar projects!