# BI project report

## 1. Introduction

There are many people around us who has grown up with Lego and are still playing with the bricks. It can be quite interesting for the LEGO fans community to learn more about the development of LEGO sets, which kinds of sets are worth buying and which sets the fan can build from the sets he/she have already owned.

The datasets used in the project come from two main sources, which are rebrickable and brickset. Rebrickable is a website which provides up-to-date information of sets and parts contained in each set with all the details such as theme, color, release year and quantity. Brickset is also a similar website containing information of LEGO sets in the past 70 years, but it has additional information, such as price, rating and number of pieces, which is not contained in rebrickable.

### 1.1 Problems addressed

**Pricing strategy of sets**

Lego has adopted a mid-premium pricing policy for its high-end products with strictly high quality control. Its target customers are children belonging to age-group of 3-15 from upper-class and middle-class families but it also has a large base of adult customers that actually contribute to a lot of sales of complex sets in creator export, technic and ideas theme lines.

In the past couple decades, the price of LEGO sets seems to become extremely high with price of set up to 800 euros. However, lots of market studies are based on the price per set and the pieces of bricks contained in the set and the affect of inflation over years is neglected. Therefore, it can be meaningful to study the change of price per brick over the past 70 years with the consideration of inflation index and check if there will be linear correlation between number of pieces in a set and the set price. After the price adjustment, the project also studies the top theme lines with highest prices each year to give the audience an overview of the different pricing strategy among different theme lines.

**Rating of sets**

Apart from the price, rating is also a key factor for customers to purchase a set. The price and rating may not have strong correlation become both the simple sets and complex sets may have the same rate scores. The project studies the average rating flow over years, theme lines with the highest rating and theme lines with the lowest rating. Here, the project mainly focuses on theme line because usually the products from the same theme category share the similar characteristics.

**Other characteristics of sets**

LEGO's productivity has increased rapidly during the past 10 years and it opens new factories in China with automatic operations by robots and machines. With the increase of productivity, LEGO's speed of releasing new sets is also accelerating. Therefore, the project checks the yearly release quantity of sets and parts contained in a set and finds out which kinds of sets contribute the most.

The color of LEGO bricks is also a key element that makes the set unique and recognizable. White, black, red and yellow are mostly used and those colors are even contained in the brand logo. However, with the time changes, the consumer preferences for colors may change as well. LEGO are becoming more diversified in color choices to satisfy the market needs. The project also studies the color composition changes over time and the representative color in its most popular themes.

**Recommendation of sets**

As the target audience of the project is LEGO fans community, many of them have already got several LEGO sets, so it can be highly helpful to build a recommendation system which indicates them which sets they can build from the sets in hand. LEGO bricks are all compatible among different sets and may share some identical parts between sets. The recommendation system will contain an algorithm to work out which sets can be built from all the parts the customer has. The suggestion can be valuable for customers because they can save money and enjoy more fun with the current sets they own.

### 1.2 Choice of tools

**Python**

The project will use python scrapy package to crawl the data on websites. The output data will be composed of 59 files from 1961 to 2019. The crawled data contains information including price, set_ID, name, rating and pieces.

Python will also be used to clean and preprocess the datasets including operations such as fill missing values, merge datasets, data pivoting and group by functions.

In order to plot the graphs that are difficult to plot in Tableau, the python plotly package is also used to create interactive graphs with high customization level.

Finally, the recommendation algorithm mentioned before is built with python.

**WinDesign**

The project will use WinDesign to plot the conceptual entity-relationship schema and the logical snowflake schema.

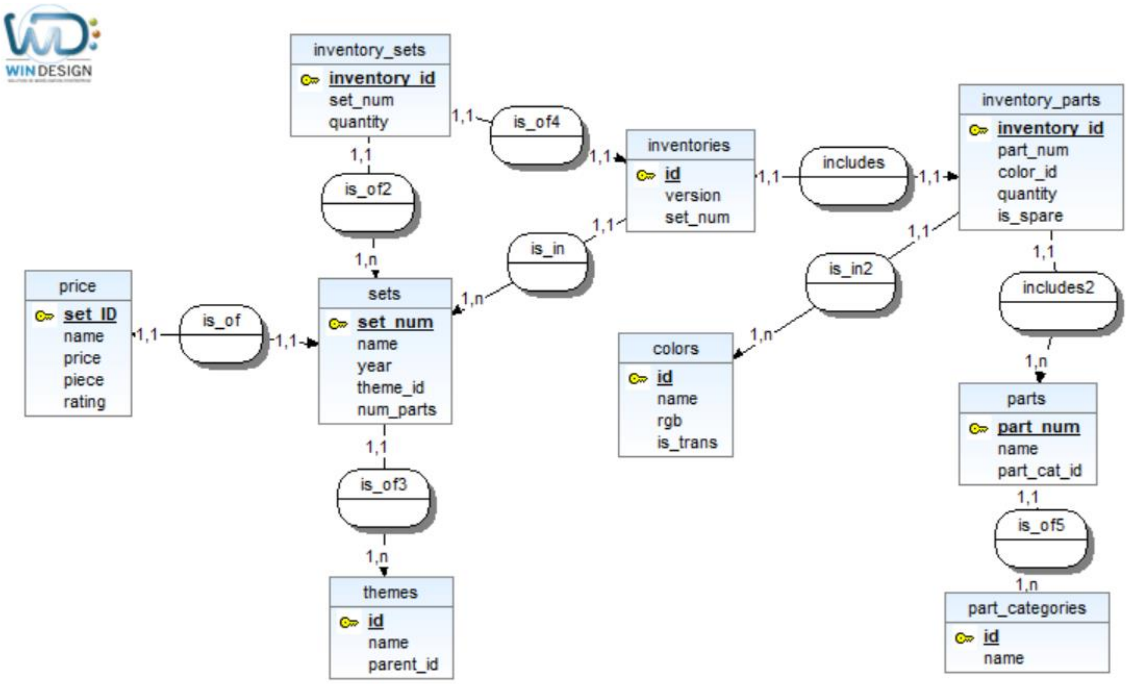
**Tableau**

After the preprocess of raw data, the project will use tableau for data visualization to deliver some insights regarding the LEGO sets and to present the topics mentioned above.

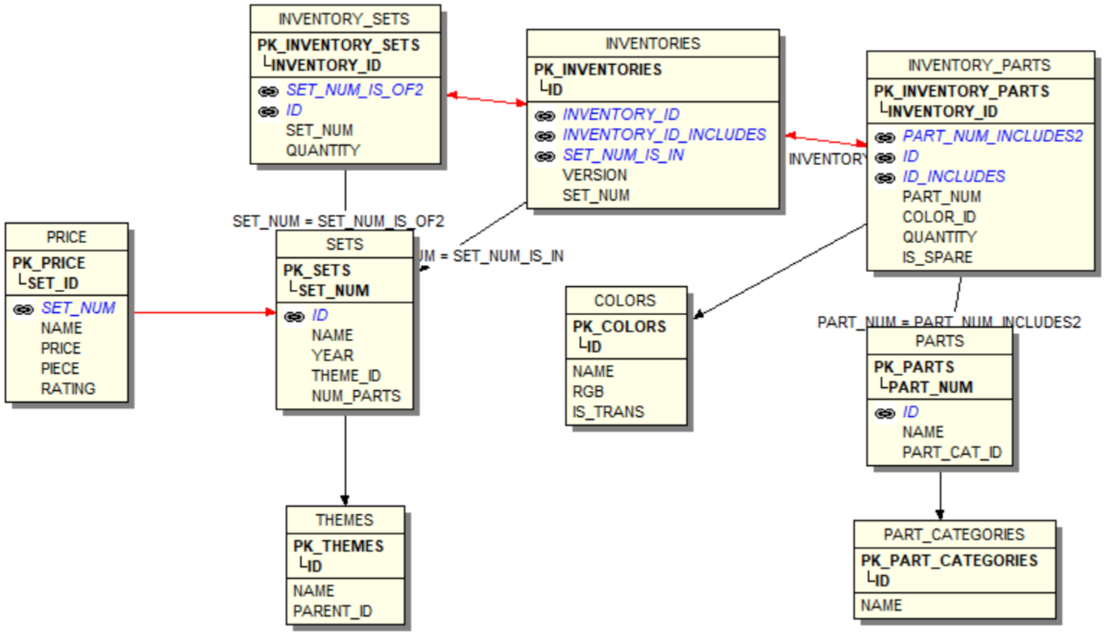
## 2. Data modeling and preparation

### 2.1. Data modeling

**Conceptual entity relationship schema**

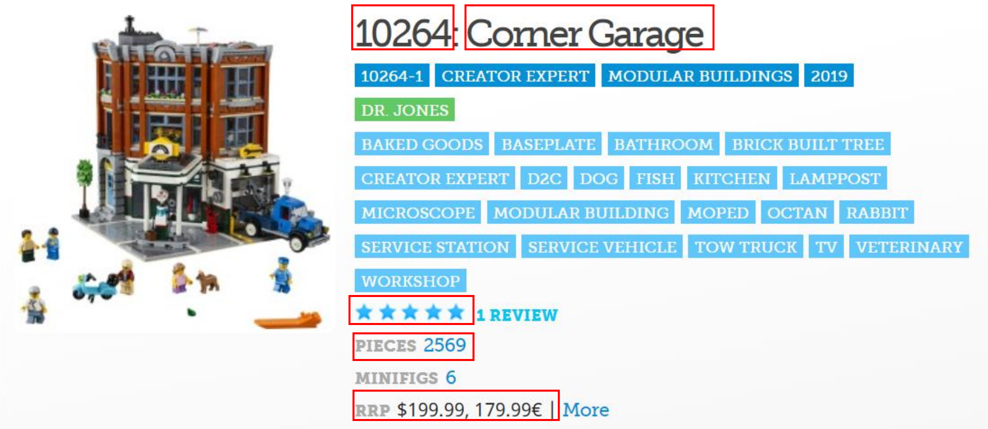


## Logical snowflake schema



## 2.2. Data preparation

## Web scrape



10264: Corner Garage

PIECES 2569

MINIFIGS 6

RRP $199.99, 179.99€ | More

The project uses python to crawl the data from brickset.com. The picture above shows the original layout of the website. The scrapy script will crawl the data containing information of product code (10264), name (Corner Garage), pieces (2569), retail price ($199.99) and rating (5.0). The useful information is highlighted in the graph.

The output will be stored in csv file with separate year because the web url of different years are separated.

## Clean datasets

The 9 datasets including themes, colors, part_categories, parts, inventories, sets, inventory_parts, inventory_sets, part_relationships from rebrickable.com are already quite clean without missing values. However, the brickset datasets scraped from the web are quite messy.

The original file contains 59 files from year 1961 to year 2019. Each file has the similar structure, so a function is defined to systematically preprocess all the files at the same time with the same operations.

First of all, the items without price and set_ID will be useless, so all rows containing NA in these two columns are deleted. Then, the original format of price contains a list of string with different types, for example, “9,$0.50 ,5.6c,Box,Retail,Normal”, “$0.50”, “0.5€|” or “$19.99, 17.99€”. To unify the price format, strings in a cell is split by comma and the function will only return the value with dollar sign. If the value only contains the euro sign, the function will transfer the currency into euro. The result returned will be in float type and will be excluded with blank space and symbols like “$”, “€” and “|”. In raw datasets, ratings are stores as strings, so they were transformed to float type. As the file doesn’t contain information of the year, so the year will be added according to the filename that is under preprocessing as the file is named after the year. Finally, the set_ID is in a format like “10264:”, so the “:” is excluded and to prevent the potential transformation of “001” to “1” by csv file, the set_ID is stored in a string format like “10264”.

## Merge datasets

Firstly, an additional dataset called CPI.csv is included to adjust the price with inflation. As is shown in the data frames below, the original CPI file contains year and CPIAUCSL which represents for the CPI level of the current year. To transform the price to the same CPI level in year 2019, an CPI index compared to year 2019 is calculated with the formula: index = CPIAUCSL_2019/CPIAUCSL_year. For example, the real price of products in year 1947 will be adjusted by the nominal price_USD * Index in year 1947 (11.37). The output of the real price is in the column “price_adjust”.

The output file is called **price_adj.csv**.

| | year | CPIAUCSL | Index_19 |
|---|---|---|---|
| **0** | 1947 | 22.331667 | 11.374404 |
| **1** | 1948 | 24.045000 | 10.563918 |
| **2** | 1949 | 23.809167 | 10.668555 |
| **3** | 1950 | 24.062500 | 10.556235 |
| **4** | 1951 | 25.973333 | 9.779623 |

| price_USD | year | Index_19 | price_adjust |
|---|---|---|---|
| 0.5 | 1961 | 8.494824 | 4.25 |
| 0.5 | 1961 | 8.494824 | 4.25 |
| 0.5 | 1961 | 8.494824 | 4.25 |
| 0.5 | 1961 | 8.494824 | 4.25 |
| 0.5 | 1961 | 8.494824 | 4.25 |

Secondly, to create customized plots in parts data, the “inv_part” file is merged with “colors” file on “color_id” to obtain detailed information of color name and rgb code. Then it’s easier for the python plotly package to generate customized graphs which match with the actual color of LEGO parts with the rgb code. The output file is called **inv_parts2.csv**

| | inventory_id | part_num | color_id | quantity | is_spare | color_name | rgb | is_trans |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 48379c01 | 72 | 1 | f | Dark Bluish Gray | 6C6E68 | f |
| **1** | 1 | 48395 | 7 | 1 | f | Light Gray | 9BA19D | f |
| **2** | 1 | mcsport6 | 25 | 1 | f | Orange | FE8A18 | f |
| **3** | 1 | paddle | 0 | 1 | f | Black | 05131D | f |
| **4** | 3 | 11816pr0005 | 78 | 1 | f | Light Flesh | F6D7B3 | f |

Thirdly, sets file is merged with “themes” and “price_adj” to obtain more information related to sets such as rating, price, pieces number, price per piece, theme name and parent theme name. The following table shows an example of output of **sets_all** file.

| | set_num | set_name | year | theme_id | num_parts | rating | price_adjust | pcs_num | price_per_pcs | theme_name | parent_theme |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | "001" | Gears | 1965 | 1 | 43 | NaN | 39.88 | 43.0 | 0.93 | Technic | Technic |
| **1** | "0011" | Town Mini-Figures | 1978 | 84 | 12 | NaN | NaN | NaN | NaN | Supplemental | Classic Town |
| **2** | "0011" | Castle 2 for 1 Bonus Offer | 1987 | 199 | 2 | 5.0 | 3.35 | 151.0 | 0.02 | Lion Knights | Castle |
| **3** | "0012" | Space Mini-Figures | 1979 | 143 | 12 | NaN | NaN | NaN | NaN | Supplemental | Space |
| **4** | "0013" | Space Mini-Figures | 1979 | 143 | 12 | NaN | NaN | NaN | NaN | Supplemental | Space |

Finally, to study the color of parts in different theme lines and different year, inv_parts2 dataset is merged with sets_all. It includes all the information above. The file is called **inv_all.csv**

# 3. Application

In this section, the results related to the problems proposed in section 1.1 are presented in the same order.

## 3.1. Pricing strategy of sets

Figure 3.1.1 shows the price of per brick changes from 1961 to 2019. The red line represents for the real price after adjustment of inflation and the yellow line represents for the nominal price. The nominal price of brick shows an increasing trend but actually after the transformation, the real price fluctuates a lot especially in the period from 1962 to 1981. The reason for the huge fluctuation and price peak can result from the release of unique sets with extremely high price compared to others. The total number of sets in the early period is limited, so a small part of sets with high price will influence the whole part significantly. After LEGO accelerated the productivity in after the year 1987 (shown in figure 3.3.1), the price of bricks became more stable and shows a decreasing trend as is shown in the dashed line. In contrast to the consumers idea that LEGO sets price is increasing in these years, the actual price per brick is surprisingly decreasing. The reason behind can be traced back to the increasing number of pieces contained in per set (shown in figure 3.3.2) and the economy of scale due to improved productivity with advanced technologies.

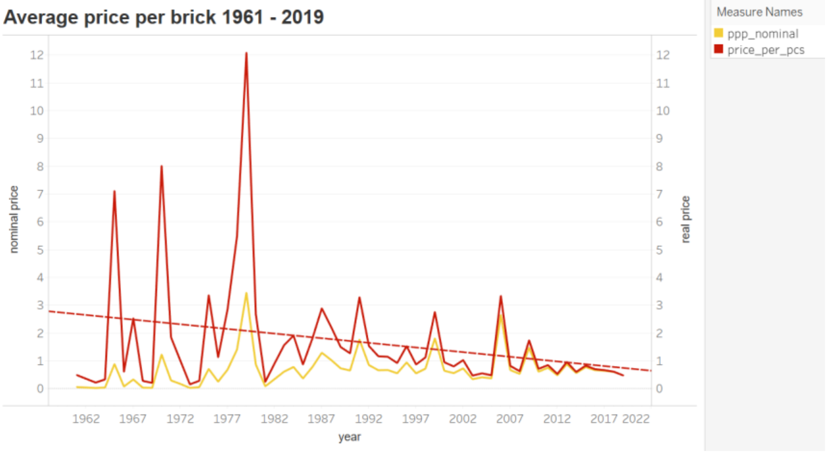


*Figure 3.1.1*

Figure 3.1.2 shows the top 10 themes with highest price per brick in year 2019. Minifigures have much higher prices than others because it's a unique piece of brick to represent people and animals and the ingrediants are more complex than a simple brick as is shown the pictures below. Duplo ranks the second because the brick is specially designed for children less than 5 years old, so the bricks are larger than the classic bricks in size and they are smoother without pointed edges in order to protect the children from hurting their fingers. However, the price per brick is almost the same in other product lines.

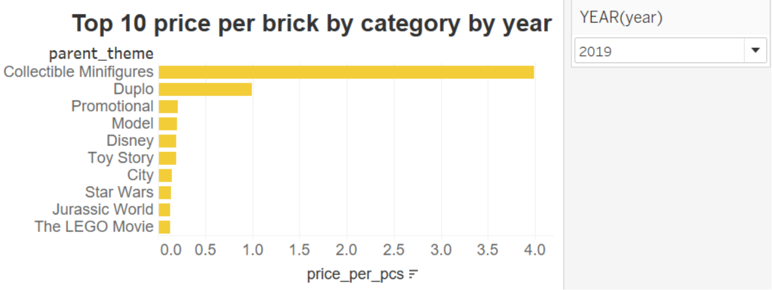
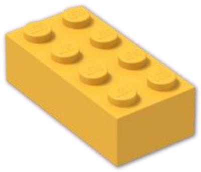


*Figure 3.1.2*

*Duplo*

*Minifigure*



*Simple brick*

The following scatter plot indicates a linear relationship between price and piece and it's highly significant with a p-value less than 0.0001 and a R-square of 0.66. However, some points are not quite fit with the dashed line and the slope is higher. Two major groups were detected as the color is assigned with the theme category. The circled area in grey contains products under the theme line called Mindstorms, and the price per piece is much higher because the sector applies more advanced technology to build robots. As is mentioned before, the price per piece for the duplo part is also high. Therefore, in this chart, we can further prove that LEGO applies different pricing strategy for different theme lines and that most of its products pricing depends on number of bricks except for those who use special materials.

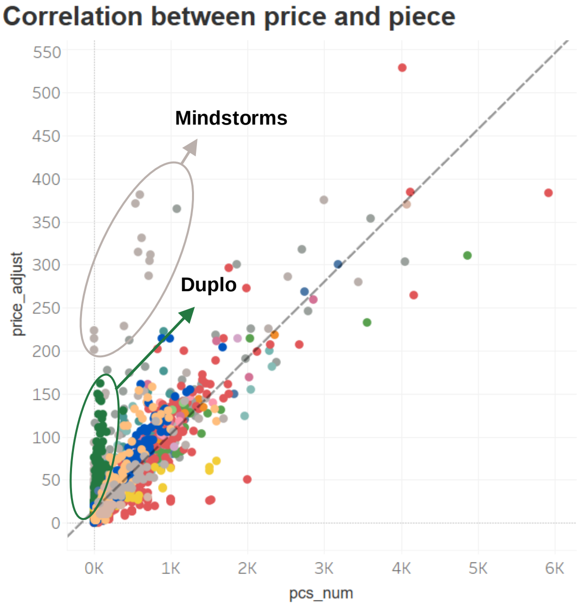
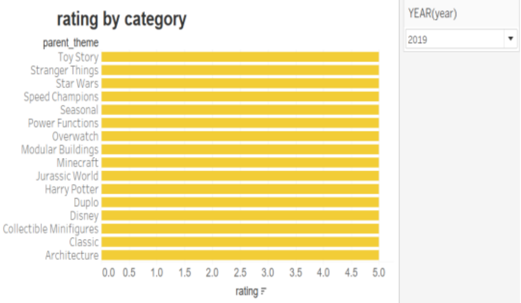


*Figure 3.1.3*

## 3.2. Rating of sets

Figure 3.2.2 shows the average rating of LEGO sets from 1969 to 2019 as the earliest record of rating traces back to 1969. The grey line shows the average rating in general, and the rating remains stable over the years with slight fluctuation around average line. The fluctuation even becomes smaller in 21st century. Therefore, LEGO is quite persistent in its quality standard since its foundation and the quality is one of its competitive advantages against other products and copycats.

In figure 3.2.1, the audience are allowed to view product lines with an average of full rating 5.0 in each year. All the products in these lines show no negative feedback, so it can be a valuable reference for selecting products.



YEAR(year)
2019

*Figure 3.2.1*

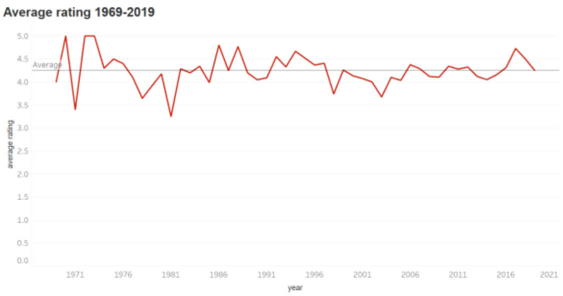


*Figure 3.2.2*

## 3.3. Other characteristics of sets

## Product release

The number of sets released in each year increases dramatically after 1995, but the number of new themes slowly increases. Therefore, the number of sets included in each theme is booming during the past two decades. (see figure 3.3.1)

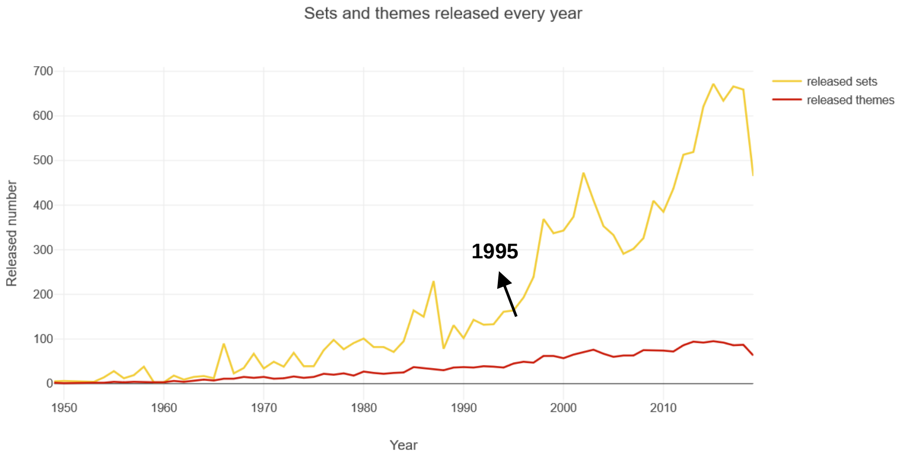


*Figure 3.3.1*

In figure 3.3.2, the average number of parts included in per set increases sharply along with the increase of sets released. However, the median number of parts in a set remains quite stable. It indicates that the number of parts in a set shows larger variety in these years. To explore more deeply, figure 3.3.3 shows the number of parts in each quantile. Extremely large sets in 95th quantile mainly drives the growth of average number of parts each year and small sets under the median almost remain the same. Therefore, large sets grow larger and small sets remain unchanged.

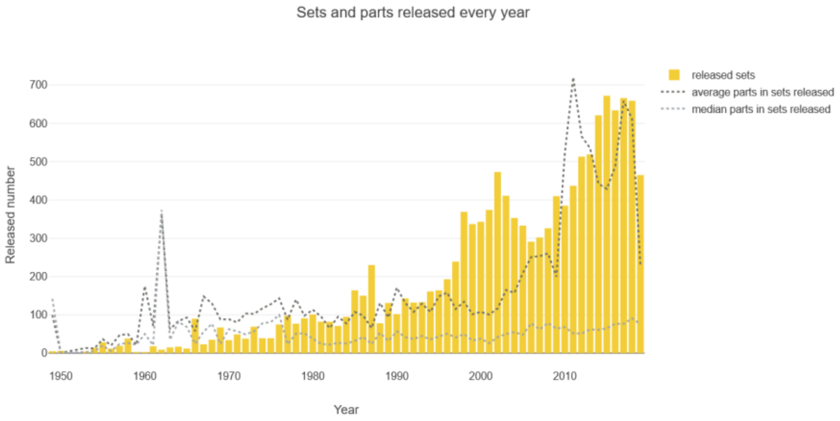


*Figure 3.3.2*

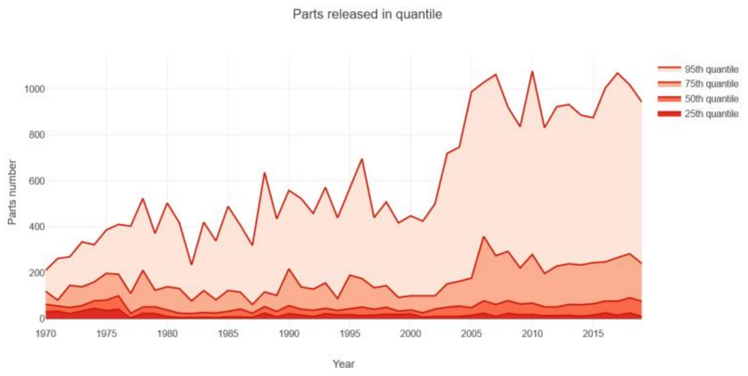


*Figure 3.3.3*

**Color composition**

Figure 3.3.4 shows the color composition of LEGO bricks in each decade from 1940s to 2010s. The classic LEGO colors, red, yellow, white, black, still occupies a large proportion, but more diversified colors appear in 21st century. With the development of new themes and changes in customer preferences, LEGO is becoming more and more colorful.

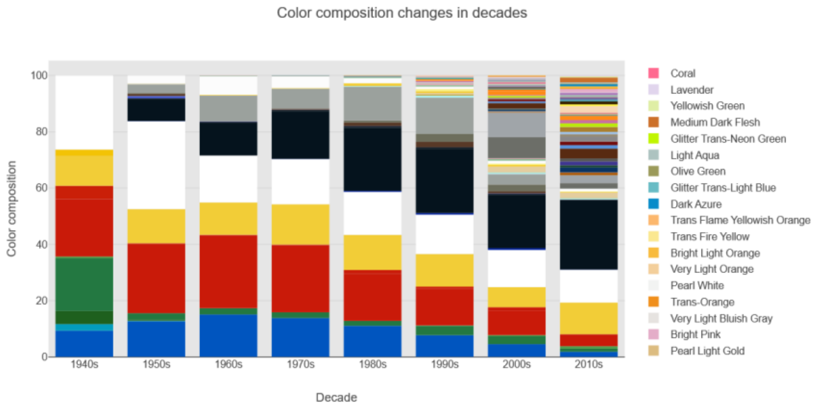
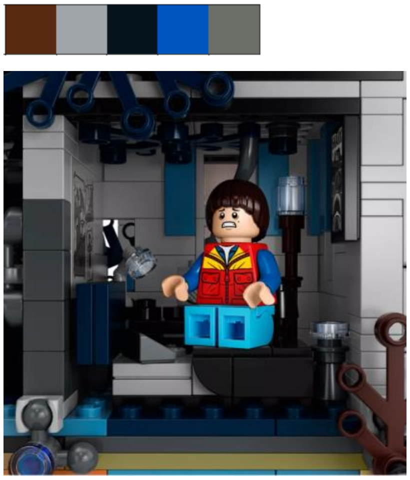


*Figure 3.3.4*

The color palettes (generated by python) help identify themes from each other and it also extracts unique features of a theme. For example. Duplo is designed for kids, so its more colorful and use colors with high saturation. The classic sets use traditional LEGO colors. And in stranger things, the color is even darker as the theme relates to horror.

*Duplo*

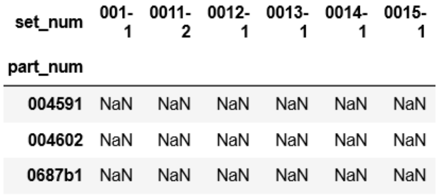
*Classic*



*Stranger things*

### 3.4. Recommendation of sets

To work out which sets can be built from the bricks we currently own, information of all parts in each set is needed. In this project, a data frame with all parts ID in rows and all sets ID in columns is built by manipulating “inventories”, “inventory_parts” and “inventory_sets” files. Figure 3.4.1 shows part of the data frame and the data is very sparse as it lists all part_num and set_num.



| set_num | 001-1 | 0011-2 | 0012-1 | 0013-1 | 0014-1 | 0015-1 |
|---|---|---|---|---|---|---|
| part_num | | | | | | |
| 004591 | NaN | NaN | NaN | NaN | NaN | NaN |
| 004602 | NaN | NaN | NaN | NaN | NaN | NaN |
| 0687b1 | NaN | NaN | NaN | NaN | NaN | NaN |

*Figure 3.4.1*

To explain the algorithm easily, I’ll use my own example. My current sets in hands are porsche 911 (42056-1), detective's office (10246-1), mickey mouse (41624-1), minnie mouse (41625-1), the beatles yellow submarine (21306-1), downtown dinner (10260-1), thanos (41605-1), iron man (41604-1) and steamboat willie (21317-1). Then, the vector contains all parts that I own can be calculated by extract the columns with these set_ID and simply add them. The algorithm will calculate the completion score with all the other sets.

The completion score measures how much percentage I can complete a new set with my sets. The completion score between two vectors is defined as (number of parts in set A exists in my sets) / (number of parts in set A). For example, the added vector for my sets is (0,0,35,6,8) and vector for set A is (2,0,30,9,5). The gap (A – my sets) equals to (2,0,-5,3,-3). Only the positive number is extracted as it represents that parts in these positions are lacked. So the number of bricks we don’t have in current sets is 2+3=5. Number of parts in set A is 2+0+30+9+5 = 46. The completion score is (1-5/46) * 100 = 89.13.

Eventually, the algorithm works out 93 sets that I can build with current bricks. Some interesting examples are shown below. With this tool, many sets that no longer existed in the market or sets with limited edition can be built and LEGO fans can enjoy the experience of recreation.

*Valentine's Day Card*

*Frankfurt*

*Santa*

## 4. Conclusion

Faced with large volume of data sources in this project, python can be a quite efficient tool to preprocess and build the cleaned datasets for visualization. It can read all the files in the same directory with a single line of code rather than manually add hundreds of files. Tableau is a user-friendly tool for visualization of well-designed graphs. I used Tableau for most of the visualization parts, but sometimes it's different to do some systematical adjustments. For example, when I want to build the stacked bar plot (figure 3.3.4), it's difficult to assign the color palette to the original color in data frame, but python plotly can easily deal with that. To make the style of graphs consistent, I adjusted the theme used in plotly. I think that I made the right choice of tools to tackle with corresponding problems, make use of the advantages and make up for the disadvantage with another tool.

I quite enjoy working for the project and I am still expecting further improvement of it. As the data structure is quite complex, the logical snowflake schema really helps me to get some ideas of how to use and merge the data sets. When practicing with the visualization in Tableau, it improves the sense of which kind of graph can be suitable for a certain topic. The project also gives me the chance to apply plotly for further customization of the plot, and the experience can really help me in the future career.

Last but not the least, as a LEGO fan, it's quite interesting to study the evolution of LEGO and the recommendation system really helps me a lot to make full use of my sets and enjoy the joy of recreation.

## 5. Sources and appendices

1) https://rebrickable.com/downloads/ updated on June 3, 2019
2) https://brickset.com/sets/year-2019 updated on June 3, 2019
3) https://github.com/twinklenoisland/LEGO-sets include all raw data, processed data and python code