# Introduction

Today we'll dive deep into a dataset all about LEGO. From the dataset we can ask whole bunch of interesting questions about the history of the LEGO company, their product offering, and which LEGO set ultimately rules them all:

- What is the most enormous LEGO set ever created and how many parts did it have?
- How did the LEGO company start out? In which year were the first LEGO sets released and how many sets did the company sell when it first launched?
- Which LEGO theme has the most sets? Is it one of LEGO's own themes like Ninjago or a theme they licensed liked Harry Potter or Marvel Superheroes?
- When did the LEGO company really expand its product offering? Can we spot a change in the company strategy based on how many themes and sets did it released year-on-year?
- Did LEGO sets grow in size and complexity over time? Do older LEGO sets tend to have more or fewer parts than newer sets?

**Data Source**

Rebrickable has compiled data on all the LEGO pieces in existence. I recommend you use download the .csv files provided in this lesson.

# Import Statements

```
import pandas as pd

import matplotlib.pyplot as plt
```

# Data Exploration

**Challenge**: How many different colours does the LEGO company produce? Read the colors.csv file in the data folder and find the total number of unique colours. Try using the .nunique() method to accomplish this.

```
color_df = pd.read_csv('data/colors.csv')
color_df.head()
```

✓ 0s completed at 2:08 PM

| | id | name | rgb | is_trans |
|---|---|---|---|---|
| **0** | -1 | Unknown | 0033B2 | f |
| **1** | 0 | Black | 05131D | f |
| **2** | 1 | Blue | 0055BF | f |
| **3** | 2 | Green | 237841 | f |
| **4** | 3 | Dark Turquoise | 008F9B | f |

```
color_df.nunique()
```

```
id          135
name        135
rgb         124
is_trans      2
dtype: int64
```

**Challenge**: Find the number of transparent colours where `is_trans == 't'` versus the number of opaque colours where `is_trans == 'f'`. See if you can accomplish this in two different ways.

```
color_df.value_counts("is_trans")
```

```
is_trans
f    107
t     28
dtype: int64
```

## Understanding LEGO Themes vs. LEGO Sets

Walk into a LEGO store and you will see their products organised by theme. Their themes include Star Wars, Batman, Harry Potter and many more.


**Architecture**

LEGO® Architecture presents some of the iconic buildings of world architecture, all perfectly realized as LEGO models. From well-known buildings to more imaginative choices that still reflect architectural excellence, these will make a great addition to any desk, home or playroom.

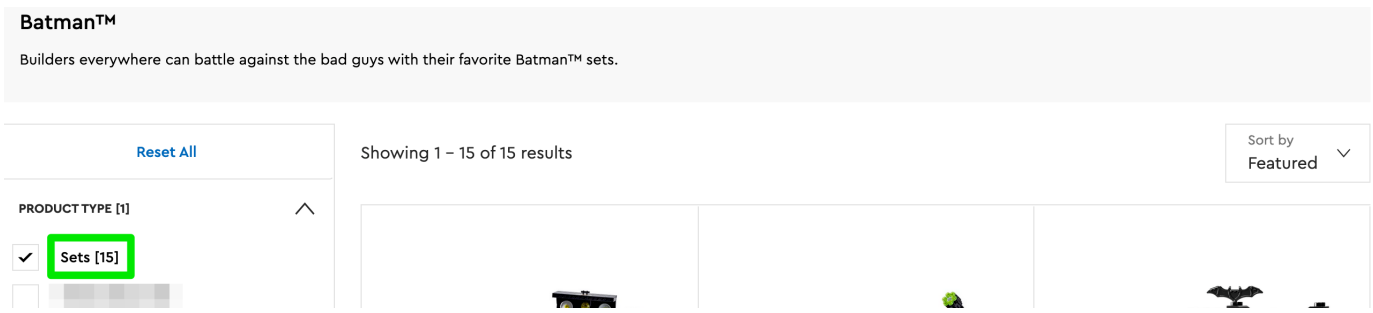

**Batman™**

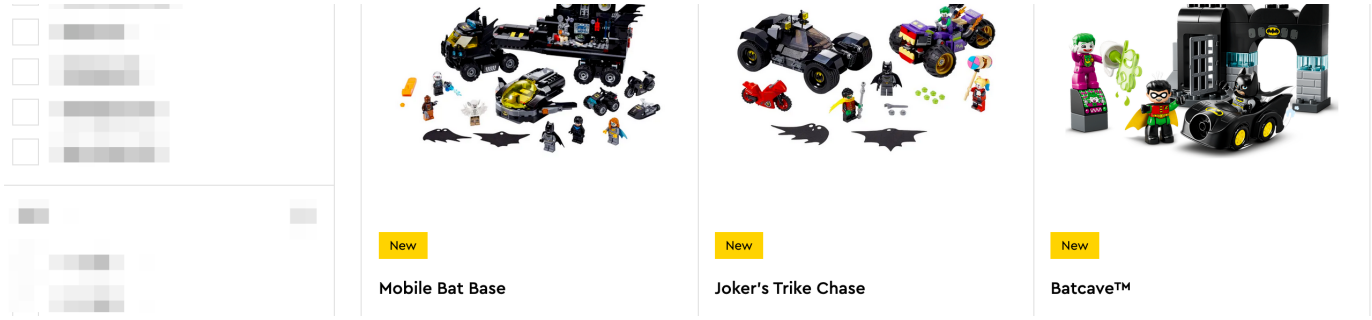
Night has fallen on Gotham City™ and builders everywhere are ready for Batman sets. They can battle against the bad guys with their favorite Dark Knight.


**BOOST**

LEGO® BOOST lets children create models with motors and sensors, and then bring their creations to life through simple, icon-based coding commands. The free LEGO BOOST tablet app includes easy step-by-step building instructions for creating and coding multifunctional models.

A lego set is a particular box of LEGO or product. Therefore, a single theme typically has many different sets.






The `sets.csv` data contains a list of sets over the years and the number of parts that each of these sets contained.

**Challenge**: Read the sets.csv data and take a look at the first and last couple of rows.

```
sets_df = pd.read_csv('data/sets.csv')
```

```
sets_df.head()
```

| | set_num | name | year | theme_id | num_parts |
|---|---|---|---|---|---|
| **0** | 001-1 | Gears | 1965 | 1 | 43 |
| **1** | 0011-2 | Town Mini-Figures | 1978 | 84 | 12 |
| **2** | 0011-3 | Castle 2 for 1 Bonus Offer | 1987 | 199 | 0 |
| **3** | 0012-1 | Space Mini-Figures | 1979 | 143 | 12 |
| **4** | 0013-1 | Space Mini-Figures | 1979 | 143 | 12 |

```
sets_df.tail()
```

| | set_num | name | year | theme_id | num_parts |
|---|---|---|---|---|---|
| **15705** | wwgp1-1 | Wild West Limited Edition Gift Pack | 1996 | 476 | 0 |
| **15706** | XMASTREE-1 | Christmas Tree | 2019 | 410 | 26 |
| **15707** | XWING-1 | Mini X-Wing Fighter | 2019 | 158 | 60 |
| **15708** | XWING-2 | X-Wing Trench Run | 2019 | 158 | 52 |
| **15709** | YODACHRON-1 | Yoda Chronicles Promotional Set | 2013 | 158 | 413 |

**Challenge**: In which year were the first LEGO sets released and what were these sets called?

```
earliest_year = sets_df.min()
print(earliest_year)
```

```
set_num          00-6
name          Spectre
year             1949
theme_id            1
num_parts           0
dtype: object
```

```
first_year_sets_df = sets_df.loc[sets_df['year'] == earliest_year.year]
print(first_year_sets_df)
```

```
      set_num                              name  year  theme_id  num_parts
9521  700.1-1         Extra-Large Gift Set (ABB)  1949       365        142
9534  700.2-1               Large Gift Set (ABB)  1949       365        178
9539  700.3-1              Medium Gift Set (ABB)  1949       365        142
9544  700.A-1              Small Brick Set (ABB)  1949       371         24
9545  700.B-1  Small Doors and Windows Set (ABB)  1949       371         12
```

**Challenge**: How many different sets did LEGO sell in their first year? How many types of LEGO products were on offer in the year the company started?

```
len(first_year_sets_df)
```

```
5
```

**Challenge**: Find the top 5 LEGO sets with the most number of parts.

```
sets_df.sort_values("num_parts", ascending=False)
```

| | set_num | name | year | theme_id | num_parts |
|---|---|---|---|---|---|
| **15004** | BIGBOX-1 | The Ultimate Battle for Chima | 2015 | 571 | 9987 |
| **11183** | 75192-1 | UCS Millennium Falcon | 2017 | 171 | 7541 |
| **10551** | 71043-1 | Hogwarts Castle | 2018 | 246 | 6020 |
| **295** | 10256-1 | Taj Mahal | 2017 | 673 | 5923 |
| **221** | 10189-1 | Taj Mahal | 2008 | 673 | 5922 |
| **...** | ... | ... | ... | ... | ... |
| **1782** | 20216-1 | MBA Robot & Micro Designer (Kits 2 - 3 Redesign) | 2013 | 432 | 0 |
| **1780** | 20214-1 | MBA Adventure Designer (Kits 7 - 9 Redesign) | 2013 | 432 | 0 |
| **6822** | 5005539-1 | Brick Pouch (Yellow) | 2018 | 501 | 0 |
| **9026** | 66319-1 | Power Miners 3 in 1 Superpack | 2009 | 439 | 0 |
| **12946** | 853471-1 | C-3PO Key Chain | 2015 | 503 | 0 |

15710 rows × 5 columns

**Challenge**: Use `.groupby()` and `.count()` to show the number of LEGO sets released year-on-year. How do the number of sets released in 1955 compare to the number of sets released in 2019?

```
sets_df.groupby("year").count()
```

| | set_num | name | theme_id | num_parts |
|---|---|---|---|---|
| **year** | | | | |
| **1949** | 5 | 5 | 5 | 5 |
| **1950** | 6 | 6 | 6 | 6 |
| **1953** | 4 | 4 | 4 | 4 |
| **1954** | 14 | 14 | 14 | 14 |
| **1955** | 28 | 28 | 28 | 28 |
| **...** | ... | ... | ... | ... |

| | | | | |
|---|---|---|---|---|
| **2017** | 786 | 786 | 786 | 786 |
| **2018** | 816 | 816 | 816 | 816 |
| **2019** | 840 | 840 | 840 | 840 |
| **2020** | 674 | 674 | 674 | 674 |
| **2021** | 3 | 3 | 3 | 3 |

71 rows × 4 columns

```
sets_df.groupby("year").count()
```

1 to 25 of 71 entries Filter

| year | set_num | name | theme_id | num_parts |
|---|---|---|---|---|
| **1949** | 5 | 5 | 5 | 5 |
| **1950** | 6 | 6 | 6 | 6 |
| **1953** | 4 | 4 | 4 | 4 |
| **1954** | 14 | 14 | 14 | 14 |
| **1955** | 28 | 28 | 28 | 28 |
| **1956** | 13 | 13 | 13 | 13 |
| **1957** | 20 | 20 | 20 | 20 |
| **1958** | 46 | 46 | 46 | 46 |
| **1959** | 4 | 4 | 4 | 4 |
| **1960** | 3 | 3 | 3 | 3 |
| **1961** | 22 | 22 | 22 | 22 |
| **1962** | 41 | 41 | 41 | 41 |
| **1963** | 20 | 20 | 20 | 20 |
| **1964** | 18 | 18 | 18 | 18 |
| **1965** | 13 | 13 | 13 | 13 |
| **1966** | 111 | 111 | 111 | 111 |
| **1967** | 28 | 28 | 28 | 28 |
| **1968** | 39 | 39 | 39 | 39 |
| **1969** | 81 | 81 | 81 | 81 |
| **1970** | 37 | 37 | 37 | 37 |
| **1971** | 51 | 51 | 51 | 51 |
| **1972** | 39 | 39 | 39 | 39 |
| **1973** | 70 | 70 | 70 | 70 |
| **1974** | 39 | 39 | 39 | 39 |
| **1975** | 39 | 39 | 39 | 39 |

Show 25 per page 1 2 3

Like what you see? Visit the data table notebook to learn more about interactive tables.

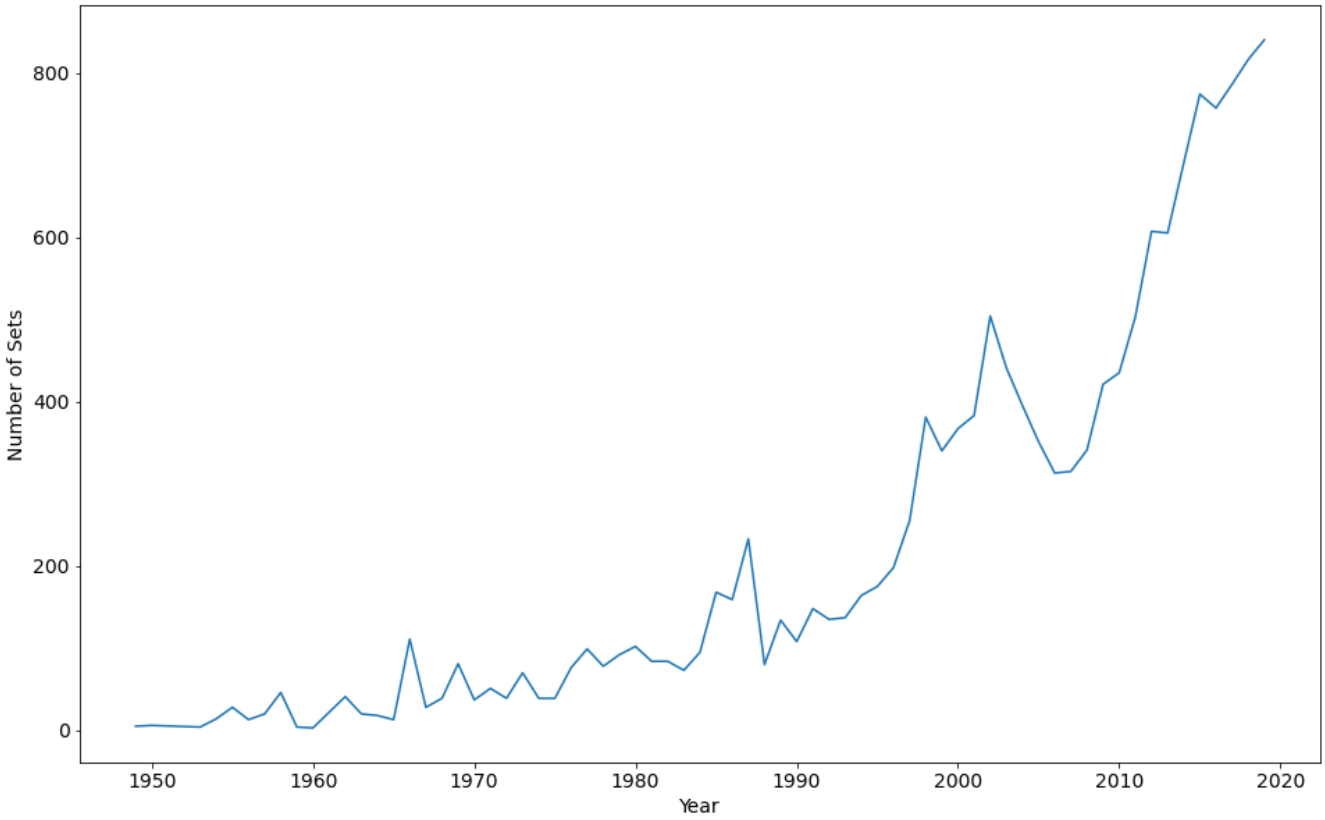
**Challenge**: Show the number of LEGO releases on a line chart using Matplotlib.

Note that the .csv file is from late 2020, so to plot the full calendar years, you will have to exclude some data from your chart. Can you use the slicing techniques covered in Day 21 to avoid plotting the last two years? The same syntax will work on Pandas DataFrames.

```
sets_by_year = sets_df.groupby("year").count()
# sets_by_year['set_num'].head()
plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Sets', fontsize=14)
```

```
plt.ylabel('Number of Sets', fontsize=14)
plt.plot(sets_by_year.index[:-2], sets_by_year['set_num'][:-2])
```

```
[<matplotlib.lines.Line2D at 0x7fc94f867050>]
```




## Aggregate Data with the Python .agg() Function

Let's work out the number of different themes shipped by year. This means we have to count the number of unique theme_ids per calendar year.

```
themes_by_year = sets_df.groupby("year").agg({'theme_id': pd.Series.nunique})
```

```
themes_by_year.rename(columns = {"theme_id":"nr_themes"}, inplace=True)
themes_by_year.head()
```

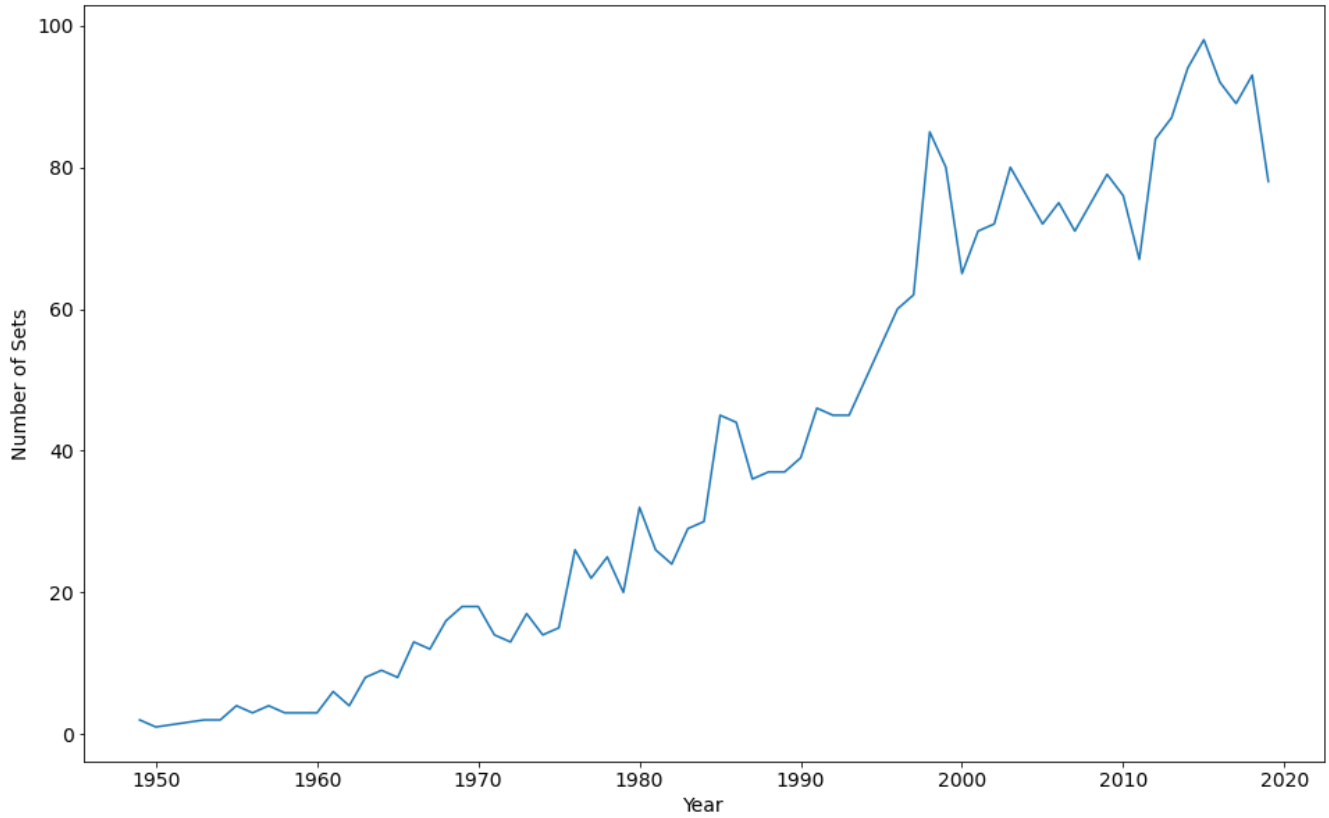
| year | nr_themes |
|---|---|
| **1949** | 2 |
| **1950** | 1 |
| **1953** | 2 |
| **1954** | 2 |
| **1955** | 4 |

**Challenge**: Plot the number of themes released by year on a line chart. Only include the full calendar years (i.e., exclude

2020 and 2021).

```
plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Sets', fontsize=14)
plt.plot(themes_by_year.index[:-2], themes_by_year['nr_themes'][:-2])
```

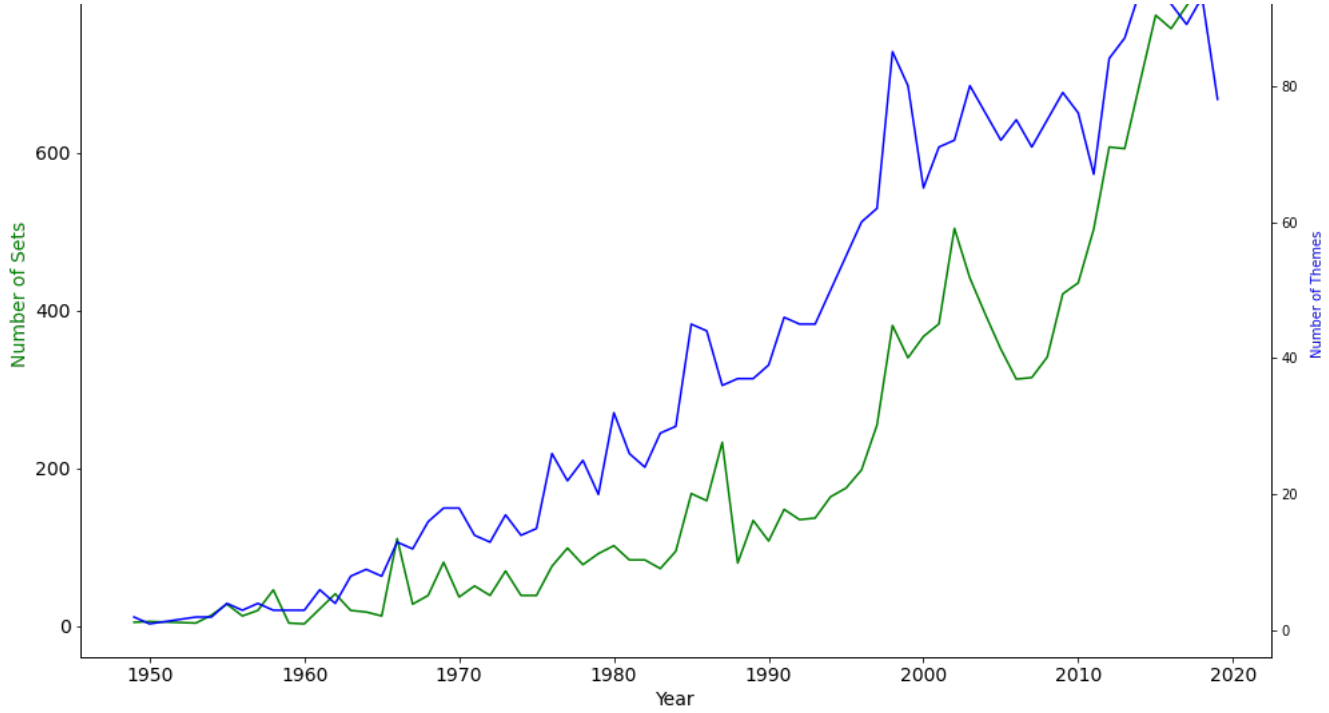
```
[<matplotlib.lines.Line2D at 0x7fc94f705890>]
```




## Line Charts with Two Seperate Axes

```
plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Number of Sets', fontsize=14)
ax1 = plt.gca() # get current axes
ax2 = ax1.twinx()
ax1.plot(sets_by_year.index[:-2], sets_by_year['set_num'][:-2], color="green")
ax2.plot(themes_by_year.index[:-2], themes_by_year['nr_themes'][:-2], color="blue")
ax1.set_ylabel("Number of Sets", color="green")
ax2.set_ylabel("Number of Themes", color="blue")
```

```
Text(0, 0.5, 'Number of Themes')
```




**Challenge**: Use the `.groupby()` and `.agg()` function together to figure out the average number of parts per set. How many parts did the average LEGO set released in 1954 compared to say, 2017?

```
parts_per_set = sets_df.groupby('year').agg({'num_parts': pd.Series.mean})
### Use Mean instead of nunique or count
parts_per_set.head()
```

| year | num_parts |
|---|---|
| **1949** | 99.600000 |
| **1950** | 1.000000 |
| **1953** | 13.500000 |
| **1954** | 12.357143 |
| **1955** | 36.607143 |

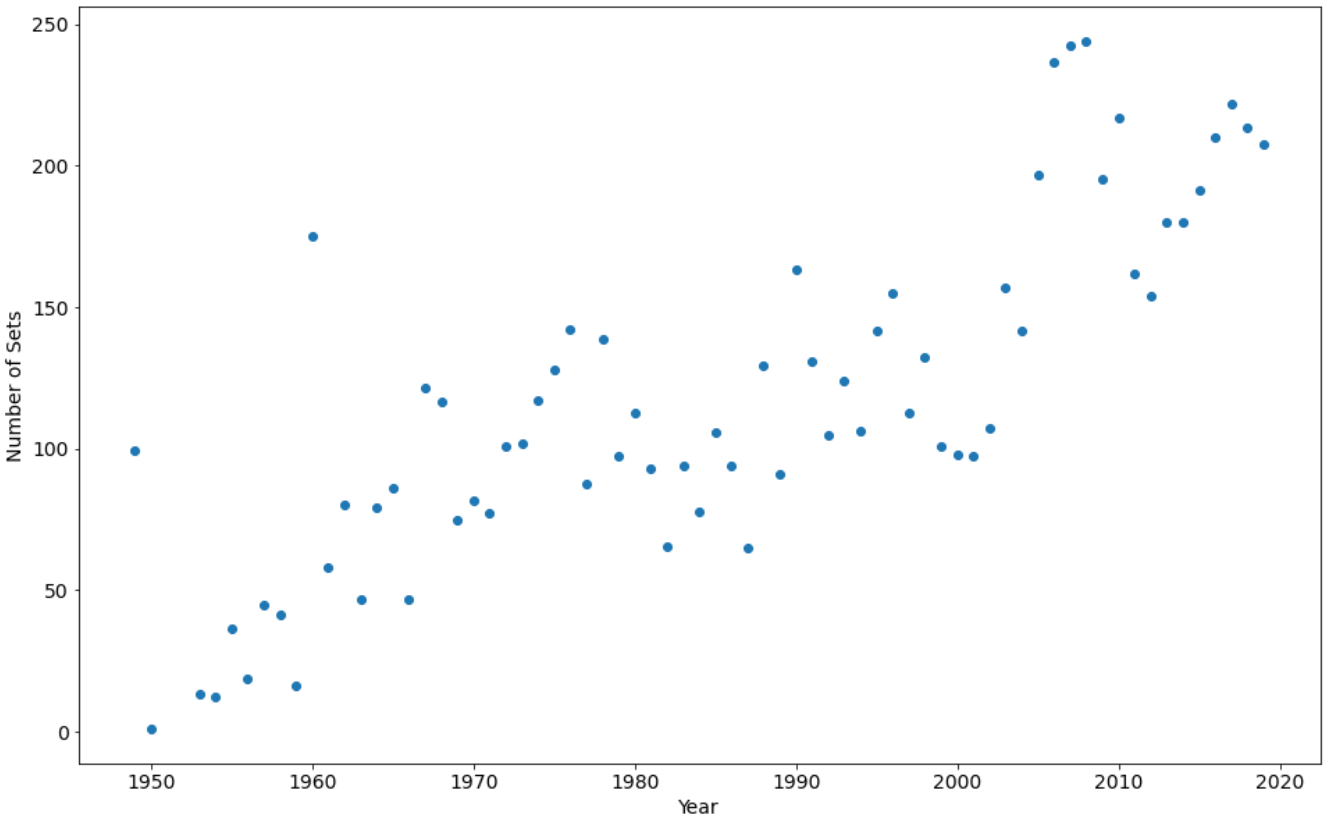
## Scatter Plots in Matplotlib

**Challenge**: Has the size and complexity of LEGO sets increased over time based on the number of parts? Plot the average number of parts over time using a Matplotlib scatter plot. See if you can use the [scatter plot documentation](#) before I show you the solution. Do you spot a trend in the chart?

```
plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
```

```
plt.yticks(fontsize=14)
plt.xlabel('Year',·fontsize=14)
plt.ylabel('Number·of·Sets',·fontsize=14)
plt.scatter(parts_per_set.index[:-2], parts_per_set['num_parts'][:-2])
```

```
<matplotlib.collections.PathCollection at 0x7fc94d7a9e10>
```



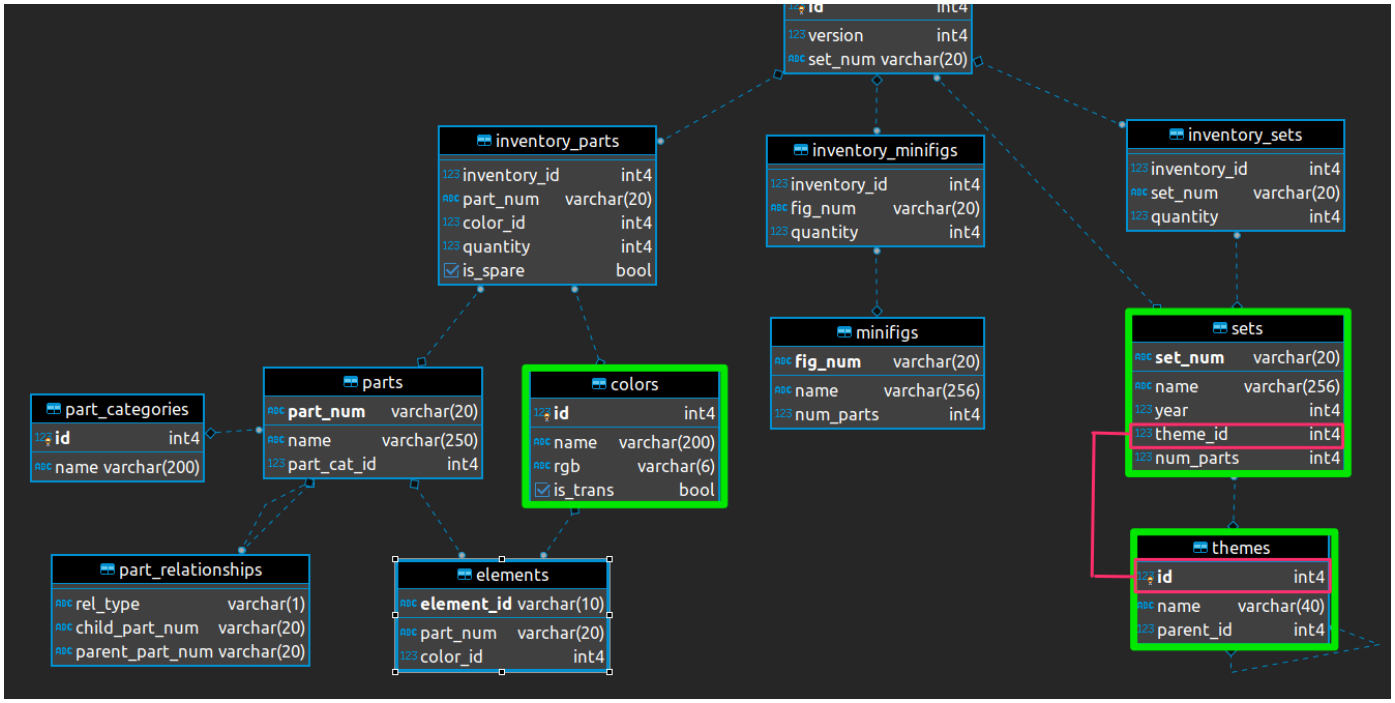

## Number of Sets per LEGO Theme

LEGO has licensed many hit franchises from Harry Potter to Marvel Super Heros to many others. But which theme has the largest number of individual sets?

```
set_theme_count = sets_df['theme_id'].value_counts()
set_theme_count.head()
```

```
158    753
501    656
494    398
435    356
503    329
Name: theme_id, dtype: int64
```

**Challenge** Use what you know about HTML markup and tags to display the database schema: https://i.imgur.com/Sg4lcjx.png




## Database Schemas, Foreign Keys and Merging DataFrames

The themes.csv file has the actual theme names. The sets .csv has `theme_ids` which link to the `id` column in the themes.csv.

**Challenge**: Explore the themes.csv. How is it structured? Search for the name 'Star Wars'. How many `id`s correspond to this name in the themes.csv? Now use these `id`s and find the corresponding the sets in the sets.csv (Hint: you'll need to look for matches in the `theme_id` column)

```
themes_df = pd.read_csv('data/themes.csv')
```

```
themes_df.head()
```

| | id | name | parent_id |
|---|---|---|---|
| **0** | 1 | Technic | NaN |
| **1** | 2 | Arctic Technic | 1.0 |
| **2** | 3 | Competition | 1.0 |
| **3** | 4 | Expert Builder | 1.0 |
| **4** | 5 | Model | 1.0 |

```
themes_df[themes_df.name == "Star Wars"]
```

| | id | name | parent_id |
|---|---|---|---|
| **17** | 18 | Star Wars | 1.0 |
| **150** | 158 | Star Wars | NaN |
| **174** | 209 | Star Wars | 207.0 |
| **211** | 261 | Star Wars | 258.0 |

## Merging (i.e., Combining) DataFrames based on a Key

```
set_theme_count = pd.DataFrame({'id': set_theme_count.index, 'set_count': set_theme_count.values})
set_theme_count.head()
```

| | id | set_count |
|---|---|---|
| **0** | 158 | 753 |
| **1** | 501 | 656 |
| **2** | 494 | 398 |
| **3** | 435 | 356 |
| **4** | 503 | 329 |

```
merged_df = pd.merge(set_theme_count, themes_df, on='id')
```

```
merged_df[:3]
```

1 to 3 of 3 entries Filter

| index | id | set_count | name | parent_id |
|---|---|---|---|---|
| **0** | 158 | 753 | Star Wars | NaN |
| **1** | 501 | 656 | Gear | NaN |
| **2** | 494 | 398 | Friends | NaN |

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

```
plt.figure(figsize=(16,10))
plt.xticks(fontsize=14, rotation=45)
plt.yticks(fontsize=14)
plt.xlabel('Theme_Name', fontsize=14)
plt.ylabel('Number of Sets', fontsize=14)
plt.bar(merged_df.name[:10], merged_df.set_count[:10])
```

```
<BarContainer object of 10 artists>
```