1. Introduction

Everyone loves Lego (unless you ever stepped on one). Did you know by the way that "Lego" was derived from the Danish phrase leg godt, which means "play well"? Unless you speak Danish, probably not.

In this project, we will analyze a fascinating dataset on every single lego block that has ever been built!

In [124]:

```
# Nothing to do here
```

In [125]:

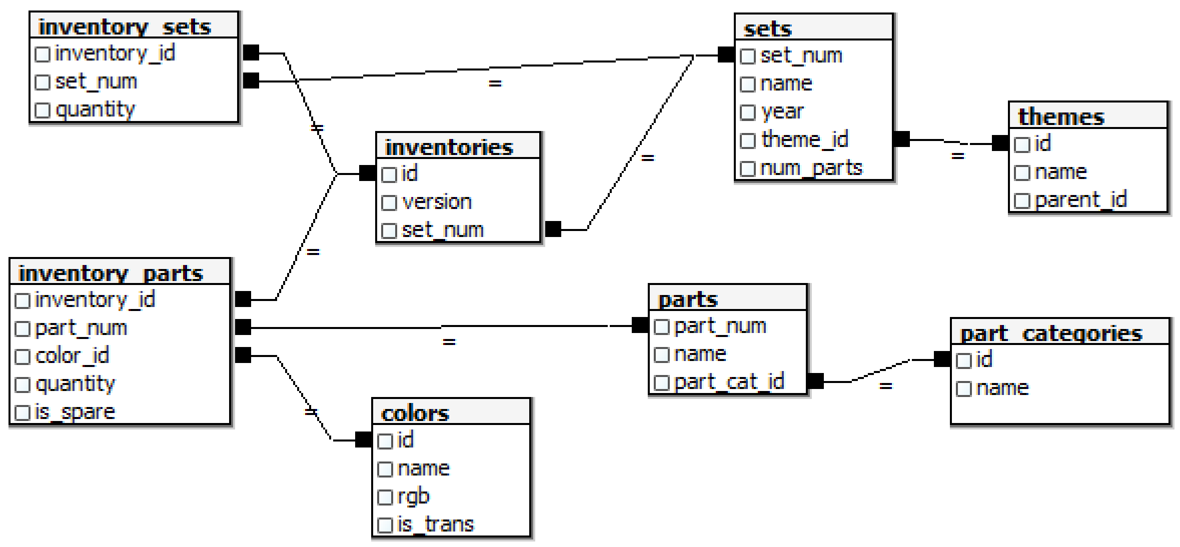
```
%%nose
def test_default():
  assert True
```

Out[125]:

```
1/1 tests passed
```

2. Reading Data

A comprehensive database of lego blocks is provided by [Rebrickable (https://rebrickable.com/downloads/)](https://rebrickable.com/downloads/). The data is available as csv files and the schema is shown below.



Let us start by reading in the colors data to get a sense of the diversity of lego sets!

In [126]:

```
# Import modules
import pandas as pd

# Read colors data
colors = pd.read_csv('datasets/colors.csv')

# Print the first few rows
colors.head()
```

Out[126]:

	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

In [127]:

```
%%nose
def test_colors_exists():
    assert 'colors' in globals(), "You should read the data into a variable named `colo
rs`"
```

Out[127]:

```
1/1 tests passed
```

3. Exploring Colors

Now that we have read the `colors` data, we can start exploring it! Let us start by understanding the number of colors available.

In [128]:

```
# How many distinct colors are available?
num_colors = colors.shape[0]
print(num_colors)
```

```
135
```

In [129]:

```
%%nose
def test_num_colors():
    assert num_colors == 135, "The variable num_colors should equal 135"
```

Out[129]:

```
1/1 tests passed
```

4. Transparent Colors in Lego Sets

The `colors` data has a column named `is_trans` that indicates whether a color is transparent or not. It would be interesting to explore the distribution of transparent vs. non-transparent colors.

In [130]:

```
# colors_summary: Distribution of colors based on transparency
colors_summary = colors.groupby('is_trans').count()
print(colors_summary)
```

```
           id  name  rgb
is_trans                
f         107   107  107
t          28    28   28
```

In [131]:

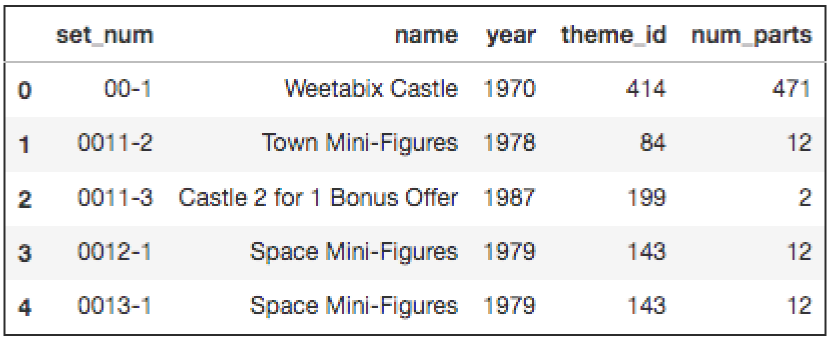
```
%%nose
def test_colors_summary_exists():
    assert 'colors_summary' in globals(), "You should have defined a variable named `co
lors_summary`"
def test_colors_summary():
    assert colors_summary.shape == (2, 3), "The DataFrame colors_summary should contain
 2 rows and 3 columns"
```

Out[131]:

```
2/2 tests passed
```

5. Explore Lego Sets

Another interesting dataset available in this database is the `sets` data. It contains a comprehensive list of sets over the years and the number of parts that each of these sets contained.



	set_num	name	year	theme_id	num_parts
0	00-1	Weetabix Castle	1970	414	471
1	0011-2	Town Mini-Figures	1978	84	12
2	0011-3	Castle 2 for 1 Bonus Offer	1987	199	2
3	0012-1	Space Mini-Figures	1979	143	12
4	0013-1	Space Mini-Figures	1979	143	12

Let us use this data to explore how the average number of parts in Lego sets has varied over the years.

In [132]:

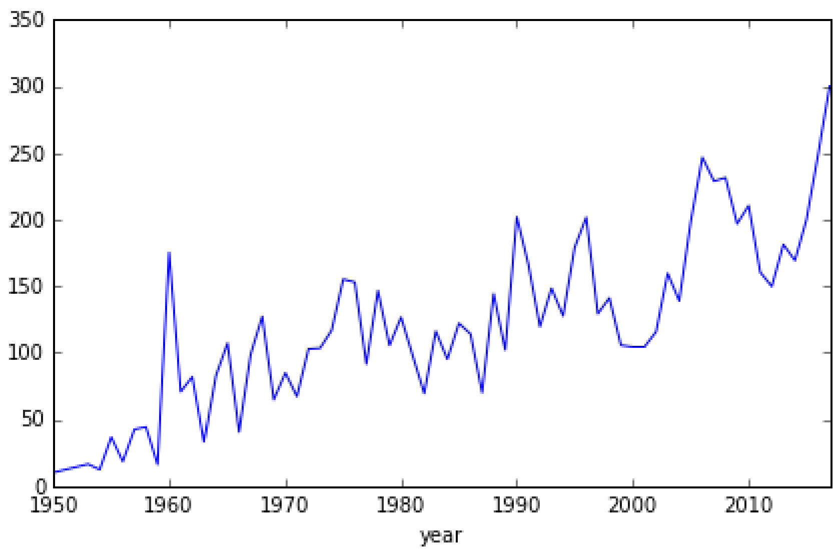
```
%matplotlib inline
# Read sets data as `sets`
sets= pd.read_csv('datasets/sets.csv')
sets.head()
# Create a summary of average number of parts by year: `parts_by_year`
parts_by_year = sets.groupby('year').mean()['num_parts']
print(parts_by_year)
# Plot trends in average number of parts by year
parts_by_year.plot(x ='year', y='average')
```

```
year
1950     10.142857
1953     16.500000
1954     12.357143
1955     36.857143
1956     18.500000
1957     42.619048
1958     44.452381
1959     16.250000
1960    175.333333
1961     70.588235
1962     81.750000
1963     33.333333
1964     82.636364
1965    107.100000
1966     40.651685
1967     98.666667
1968    127.200000
1969     64.594203
1970     84.793103
1971     67.022222
1972    102.842105
1973    103.367647
1974    116.769231
1975    155.225806
1976    153.029412
1977     91.500000
1978    146.616438
1979    105.414634
1980    126.636364
1981     97.835443
           ...
1988    144.250000
1989    102.061404
1990    202.035294
1991    166.424528
1992    119.617391
1993    148.432432
1994    127.640625
1995    179.039062
1996    201.770833
1997    129.221649
1998    141.126154
1999    105.543333
2000    104.376147
2001    104.365782
2002    115.700224
2003    159.681928
2004    138.862534
2005    198.745455
2006    246.904594
2007    229.025078
2008    231.644699
2009    196.898263
2010    210.646396
2011    160.452191
2012    149.808130
2013    181.359191
2014    169.320280
2015    200.223881
2016    248.945813
```

```
2017    300.121277
Name: num_parts, dtype: float64
```

Out[132]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f654284fe10>
```



In [133]:

```
%%nose
def test_sets_exists():
    assert 'sets' in globals(), "You should read the data into a variable named `sets`"
def test_parts_by_year_exists():
    assert 'parts_by_year' in globals(), "You should have defined a variable named `par
ts_by_year`"
```

Out[133]:

```
2/2 tests passed
```

6. Lego Themes Over Years

Lego blocks ship under multiple themes (https://shop.lego.com/en-US/Themes). Let us try to get a sense of how the number of themes shipped has varied over the years.

In [134]:

```
# themes_by_year: Number of themes shipped by year
themes_by_year = sets[['year', 'theme_id']].groupby('year', as_index = False).agg({"the
me_id": pd.Series.count})
themes_by_year.head(2)
```

Out[134]:

	year	theme_id
0	1950	7
1	1953	4

In [135]:

```
%%nose
def test_themes_by_year_exists():
    assert 'themes_by_year' in globals(), "You should have defined a variable named `th
emes_by_year`"
def test_themes_by_year():
    assert themes_by_year.shape == (66, 2), "The DataFrame themes_by_year should contai
n 66 rows and 2 columns"
def test_themes_by_year_names():
    colnames = ['year', 'theme_id']
    assert all(name in themes_by_year for name in colnames), "Your DataFrame, bnames, s
hould have columns named: year, theme_id"
```

Out[135]:

```
3/3 tests passed
```

7. Wrapping It All Up!

Lego blocks offer an unlimited amount of fun across ages. We explored some interesting trends around colors, parts, and themes.

In [136]:

```
# Nothing to do here
```

In [137]:

```
%%nose
def test_default():
  assert True
```

Out[137]:

```
1/1 tests passed
```