```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from datetime import datetime


# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Sa
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/omparison-of-two-revit-projects/2house.csv
/kaggle/input/omparison-of-two-revit-projects/1house.csv
/kaggle/input/omparison-of-two-revit-projects/1house.dae
/kaggle/input/omparison-of-two-revit-projects/2house.dae
/kaggle/input/revitproject-3-and-4/Projekt Version1_25052022.dae
/kaggle/input/revitproject-3-and-4/Projekt_Version2_17052022.dae
/kaggle/input/revitproject-3-and-4/Projekt_Version2_17052022.csv
/kaggle/input/revitproject-3-and-4/Projekt Version1_25052022.csv
/kaggle/input/revitproject-3-and-4/Project version3_28052022.csv
/kaggle/input/revitproject-3-and-4/Project version3_28052022.dae
/kaggle/input/pdf-report/pdf_13.txt
/kaggle/input/pdf-report/construction-sample-logo.png
/kaggle/input/pdf-report/OD.jpg
/kaggle/input/pdf-report/pipeline.jpg
/kaggle/input/pdf-report/pdf_8.txt
/kaggle/input/pdf-report/pipelinem.jpg
/kaggle/input/pdf-report/pdf_6.txt
/kaggle/input/pdf-report/pdf_11.txt
/kaggle/input/pdf-report/pdf_15.txt
/kaggle/input/pdf-report/pdf_7.txt
/kaggle/input/pdf-report/pdf_3.txt
/kaggle/input/pdf-report/Exampe_Code2.png
/kaggle/input/pdf-report/skript.jpg
/kaggle/input/pdf-report/pdf_5.txt
/kaggle/input/pdf-report/pdf_4.txt
/kaggle/input/pdf-report/comparetwoprojects.jpg
/kaggle/input/pdf-report/pdf_9.txt
/kaggle/input/pdf-report/Exampe_Code.png
/kaggle/input/pdf-report/pdf_14.txt
/kaggle/input/pdf-report/noBIM.jpg
/kaggle/input/pdf-report/OpenDataBIM.jpg
/kaggle/input/pdf-report/pdf_12.txt
/kaggle/input/pdf-report/pdf_1.txt
/kaggle/input/pdf-report/OpenDataBIMlogo.jpg
/kaggle/input/pdf-report/pdf_16.txt
/kaggle/input/pdf-report/OpenDataBIMlogo2.jpg
/kaggle/input/pdf-report/pdf_10.txt
/kaggle/input/pdf-report/pdf_2.txt
/kaggle/input/pdf-report/white.png
/kaggle/input/pdf-report/grey.png
```

```
#Path to project files
pathf = '../input/revitproject-3-and-4/'
v1name = 'Projekt Version1_25052022'
v2name = 'Projekt_Version2_17052022'
project_v1 = pd.read_csv(pathf+v1name+'.csv', low_memory=False, index_col = 0)
project_v2 = pd.read_csv(pathf+v2name+'.csv', low_memory=False, index_col = 0)
```

더블클릭 또는 Enter 키를 눌러 수정

```
# Presentation of the Revit file as a DataFrame (Project version 1)

project_v1
```

	Design Option	Category	Family	Type	Horizontal Profile Offset	Vertical Profile Offset	Length	Family and Type	Type I
7738377	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10973195	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11008197	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10445208	None	OST_Fascia	Fascia	Fascia	0.0	-200.0	75686.0	Fascia	Fascia
10870571	None	OST_StructuralColumns	SHS150x150x9	SHS150x150x9	NaN	NaN	3929.0	SHS150x150x9	SHS150x150x9
...	...	...	...	...	...	...	...	...	.
10937820	None	OST_Walls	plaster	plaster	NaN	NaN	6747.0	plaster	plaste
10937821	None	OST_Walls	plaster	plaster	NaN	NaN	11102.0	plaster	plaste
10937822	None	OST_Walls	plaster	plaster	NaN	NaN	13920.0	plaster	plaste
10937823	None	OST_Walls	plaster	plaster	NaN	NaN	6602.0	plaster	plaste
10940768	None	OST_Walls	plaster	plaster	NaN	NaN	1222.0	plaster	plaste

216 rows × 379 columns

```
# Presentation of the Revit file as a DataFrame (Project version 2)

project_v2
```

	Design Option	Category	Family	Type	Horizontal Profile Offset	Vertical Profile Offset	Length	Family and Type	Type I
7738377	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10973195	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
11008197	None	None	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10445208	None	OST_Fascia	Fascia	Fascia	0.0	-200.0	75686.0	Fascia	Fascia
10870571	None	OST_StructuralColumns	SHS150x150x9	SHS150x150x9	NaN	NaN	3929.0	SHS150x150x9	SHS150x150x9
...	...	...	...	...	...	...	...	...	.
10937820	None	OST_Walls	plaster	plaster	NaN	NaN	6747.0	plaster	plaste
10937821	None	OST_Walls	plaster	plaster	NaN	NaN	11102.0	plaster	plaste
10937822	None	OST_Walls	plaster	plaster	NaN	NaN	13920.0	plaster	plaste
10937823	None	OST_Walls	plaster	plaster	NaN	NaN	6602.0	plaster	plaste
10940768	None	OST_Walls	plaster	plaster	NaN	NaN	1222.0	plaster	plaste

216 rows × 379 columns

코딩을 시작하거나 AI로 코드를 생성하세요.

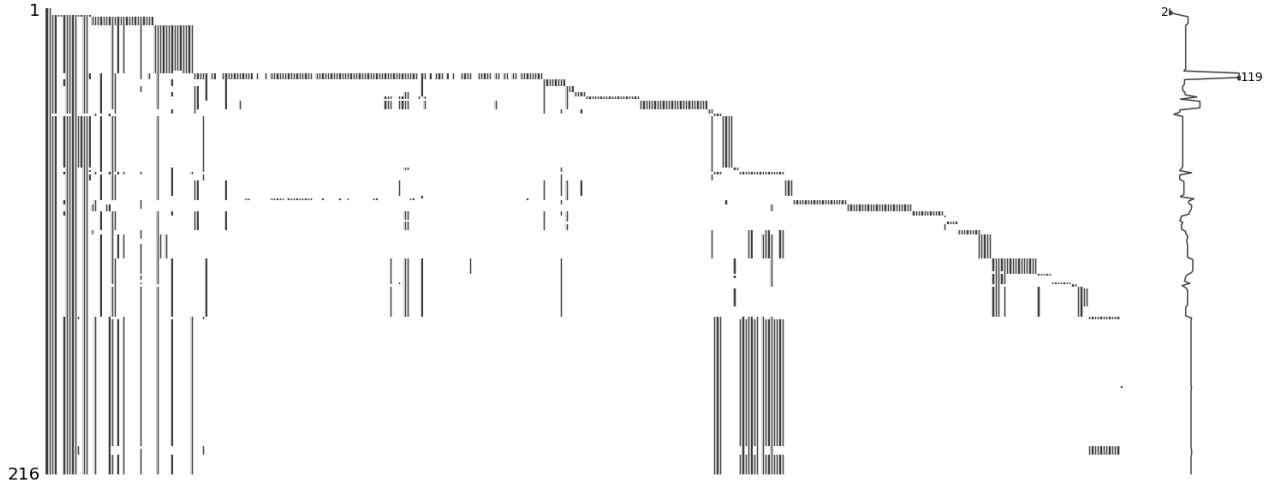
NoBIM concept

Although BIM ends the very moment the model is ready and you start processing model data, model-centric approaches (both openBIm and closedBIM) force you to address the model with model-centric software. The noBIM approach changes the game because it is a data-centric approach, it shifts the focus from the model to the model data itself, and it unleashes the full power of data

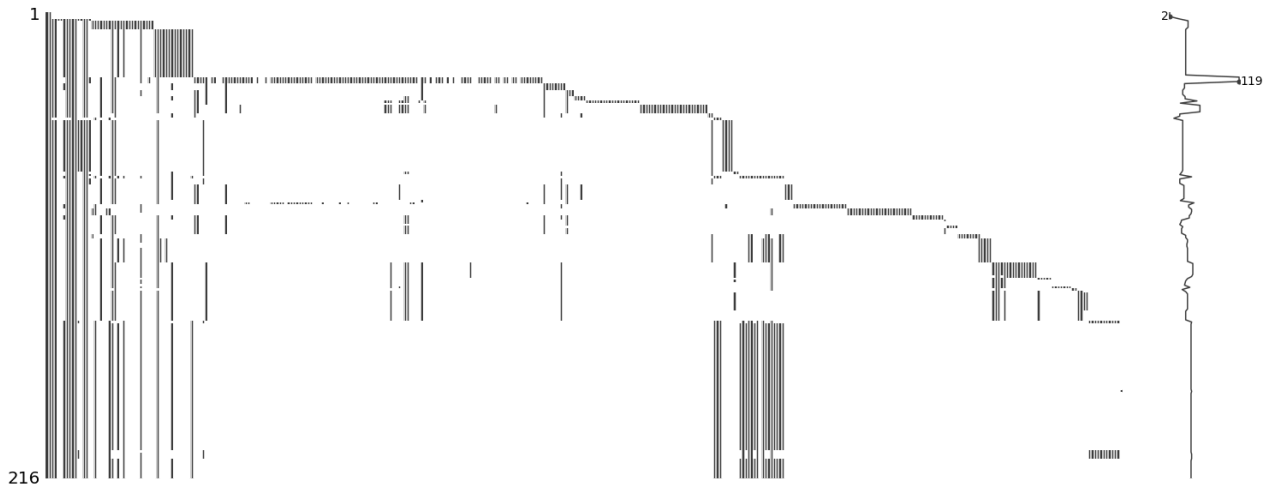
management tools from finance, science, or any other field. This Pipeline can be applied to hundreds or thousands of similar projects to generate automated reports.

More on the NoBIM concept: opendatabim.io

```
# The msno.matrix nullity matrix is a data-dense display which
# lets you quickly visually pick out patterns in data completion

import missingno as msno
fig = msno.matrix(project_v1)
fig_copy = fig.get_figure()
fig_copy.savefig('plot_pr1.png', bbox_inches = 'tight')
```




```
# The msno.matrix nullity matrix is a data-dense display which
# lets you quickly visually pick out patterns in data completion

import missingno as msno
fig = msno.matrix(project_v2)
fig_copy = fig.get_figure()
fig_copy.savefig('plot_pr2.png', bbox_inches = 'tight')
```




```
# Compare the two dataframes and highlight the differences in the data

df1 = project_v1.compare(project_v2, align_axis=1).rename(columns={'self': 'Version1', 'other': 'Version2'}, level=-1)
def highlight_diff(data, color='yellow'):
    attr = 'background-color: {}'.format(color)
    other = data.xs('Version2', axis='columns', level=-1)
    return pd.DataFrame(np.where(data.ne(other, level=0), attr, ''),
                        index=data.index, columns=data.columns)
fname = 'VersionComparison' + datetime.now().strftime("%Y_%m_%d-%I_%M_%S_%p") + '.csv'
fnamep = 'VersionComparison' + datetime.now().strftime("%Y_%m_%d-%I_%M_%S_%p") + '.png'

df1.to_csv(fname)
df1.style.apply(highlight_diff, axis=None)
```

	Volume		Area		Elevation at Bottom		Elevation at Bottom Survey		Width		Rebar Cover - Bottom Face	
	Version1	Version2	Version1	Version2	Version1	Version2	Version1	Version2	Version1	Version2	Version1	Versio
10783511	0.75 m³	0.93 m³	3 m²	4 m²	nan	nan	nan	nan	nan	nan	nan	nan
10938050	0.22 m³	0.40 m³	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
10783244	26.66 m³	1.97 m³	22 m²	9 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783257	9.51 m³	0.81 m³	8 m²	4 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783270	17.55 m³	1.35 m³	14 m²	6 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783283	4.08 m³	0.12 m³	3 m²	1 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783300	12.96 m³	1.11 m³	11 m²	5 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783309	12.13 m³	0.93 m³	10 m²	4 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783326	10.98 m³	0.93 m³	9 m²	4 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783339	10.56 m³	0.89 m³	9 m²	4 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783352	12.56 m³	1.05 m³	10 m²	5 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783371	9.07 m³	0.70 m³	7 m²	3 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783384	4.41 m³	0.29 m³	4 m²	1 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783406	4.26 m³	0.41 m³	3 m²	2 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783419	4.49 m³	0.34 m³	4 m²	1 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783432	5.65 m³	0.43 m³	5 m²	2 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan
10783454	2.40 m³	0.27 m³	2 m²	1 m²	36570.000000	37570.000000	36570.000000	37570.000000	1700.000000	700.000000	nan	nan

```
# Compare the two dataframes, the arrangement of data from different versions in the lines

df2 = project_v1.compare(project_v2, align_axis=0).rename(index={'self': 'Version1', 'other': 'Version2'})
df2
```

		Volume	Area	Elevation at Bottom	Elevation at Bottom Survey	Width	Rebar Cover - Bottom Face	Rebar Cover - Other Faces	Rebar Cover - Top Face	Foundation Thickness
10783511	**Version1**	0.75 m³	3 m²	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Version2	0.93 m³	4 m²	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10938050	**Version1**	0.22 m³	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Version2	0.40 m³	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10783244	**Version1**	26.66 m³	22 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.97 m³	9 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783257	**Version1**	9.51 m³	8 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.81 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783270	**Version1**	17.55 m³	14 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.35 m³	6 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783283	**Version1**	4.08 m³	3 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.12 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783300	**Version1**	12.96 m³	11 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.11 m³	5 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783309	**Version1**	12.13 m³	10 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.93 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783326	**Version1**	10.98 m³	9 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.93 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783339	**Version1**	10.56 m³	9 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.89 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783352	**Version1**	12.56 m³	10 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.05 m³	5 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783371	**Version1**	9.07 m³	7 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.70 m³	3 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783384	**Version1**	4.41 m³	4 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.29 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783406	**Version1**	4.26 m³	3 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.41 m³	2 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783419	**Version1**	4.49 m³	4 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.34 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0

```
# Replacing long column names with shorter names
# to create a table that will later appear in the final PDF

import re
df3 = df2.rename(columns=lambda x: re.sub('Elevation at Bottom','EB',x))
df3 = df3.rename(columns=lambda x: re.sub('Rebar Cover - ','RC ',x))
df3
```

		Volume	Area	EB	EB Survey	Width	RC Bottom Face	RC Other Faces	RC Top Face	Foundation Thickness
10783511	**Version1**	0.75 m³	3 m²	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Version2	0.93 m³	4 m²	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10938050	**Version1**	0.22 m³	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
	Version2	0.40 m³	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10783244	**Version1**	26.66 m³	22 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.97 m³	9 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783257	**Version1**	9.51 m³	8 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.81 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783270	**Version1**	17.55 m³	14 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.35 m³	6 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783283	**Version1**	4.08 m³	3 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.12 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783300	**Version1**	12.96 m³	11 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.11 m³	5 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783309	**Version1**	12.13 m³	10 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.93 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783326	**Version1**	10.98 m³	9 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.93 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783339	**Version1**	10.56 m³	9 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.89 m³	4 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783352	**Version1**	12.56 m³	10 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	1.05 m³	5 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783371	**Version1**	9.07 m³	7 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.70 m³	3 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783384	**Version1**	4.41 m³	4 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.29 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783406	**Version1**	4.26 m³	3 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.41 m³	2 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783419	**Version1**	4.49 m³	4 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.34 m³	1 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0
10783432	**Version1**	5.65 m³	5 m²	36570.0	36570.0	1700.0	NaN	NaN	NaN	1230.0
	Version2	0.43 m³	2 m²	37570.0	37570.0	700.0	NaN	NaN	NaN	230.0

```
# Visualization of the table as an image, which is saved in the Output folder
# and then will be used during the formation of the final PDF report

import numpy as np
import six
import matplotlib.pyplot as plt
def render_mpl_table(data, col_width=3.0, row_height=0.525, font_size=15,
                     header_color='#40466e', row_colors=['#f1f1f2', 'w'], edge_color='w',
                     bbox=[0, 0, 1, 1], header_columns=0,header_row=0,
                     ax=None, **kwargs):
    if ax is None:
        size = (np.array(data.shape[::-1]) + np.array([0, 1])) * np.array([col_width, row_height])
        fig, ax = plt.subplots(figsize=size)
        ax.axis('off')

    mpl_table = ax.table(cellText=data.values,colLabels=data.columns,rowLabels=data.index, bbox=bbox,**kwargs)

    mpl_table.auto_set_font_size(False)
    mpl_table.set_fontsize(font_size)
```

```
    for k, cell in  six.iteritems(mpl_table._cells):
        cell.set_edgecolor(edge_color)
        if k[0] == 0 or k[1] < header_columns:
            cell.set_text_props(weight='bold', color='w')
            cell.set_facecolor(header_color)
        else:
            cell.set_facecolor(row_colors[k[0]%len(row_colors) ])
    plt.tight_layout()
    #fig.savefig(pathro + "tabella3.png")
    return ax


fig = render_mpl_table(df3, header_columns=0, col_width=2.0, header_color='#40466e', row_colors=['#f1f1f2', 'w'], edge_color='w')
#plt.savefig("tabella.png",bbox_inches="tight")
#fig.savefig("table_mpl.png")
fig.get_figure().savefig('table_mpl.png', bbox_inches='tight')
```

	Volume	Area	EB	EB Survey	Width	RC Bottom Fac	RC Other Face	RC Top Face	Foundation Thickn
(10783511, 'Version1')	0.75 m³	3 m²	nan	nan	nan	nan	nan	nan	nan
(10783511, 'Version2')	0.93 m³	4 m²	nan	nan	nan	nan	nan	nan	nan
(10938050, 'Version1')	0.22 m³	nan	nan	nan	nan	nan	nan	nan	nan
(10938050, 'Version2')	0.40 m³	nan	nan	nan	nan	nan	nan	nan	nan
(10783244, 'Version1')	26.66 m³	22 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783244, 'Version2')	1.97 m³	9 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783257, 'Version1')	9.51 m³	8 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783257, 'Version2')	0.81 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783270, 'Version1')	17.55 m³	14 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783270, 'Version2')	1.35 m³	6 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783283, 'Version1')	4.08 m³	3 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783283, 'Version2')	0.12 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783300, 'Version1')	12.96 m³	11 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783300, 'Version2')	1.11 m³	5 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783309, 'Version1')	12.13 m³	10 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783309, 'Version2')	0.93 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783326, 'Version1')	10.98 m³	9 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783326, 'Version2')	0.93 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783339, 'Version1')	10.56 m³	9 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783339, 'Version2')	0.89 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783352, 'Version1')	12.56 m³	10 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783352, 'Version2')	1.05 m³	5 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783371, 'Version1')	9.07 m³	7 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783371, 'Version2')	0.70 m³	3 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783384, 'Version1')	4.41 m³	4 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783384, 'Version2')	0.29 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783406, 'Version1')	4.26 m³	3 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783406, 'Version2')	0.41 m³	2 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783419, 'Version1')	4.49 m³	4 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783419, 'Version2')	0.34 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783432, 'Version1')	5.65 m³	5 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783432, 'Version2')	0.43 m³	2 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783454, 'Version1')	2.40 m³	2 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783454, 'Version2')	0.27 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10890307, 'Version1')	0.00 m³	nan	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10890307, 'Version2')	0.02 m³	nan	37570.0	37570.0	700.0	Rebar Cover 1	Rebar Cover 1	Rebar Cover 1	230.0

```
# The same table in a different representation

import matplotlib.pyplot as plt
def render_mpl_table(data, col_width=2.0, row_height=0.525, font_size=15,
                     header_color='#40466e', row_colors=['#f1f1f2', 'w'], edge_color='w',
                     bbox=[0, 0, 1, 1], header_columns=0,header_row=0,
                     ax=None, **kwargs):
```

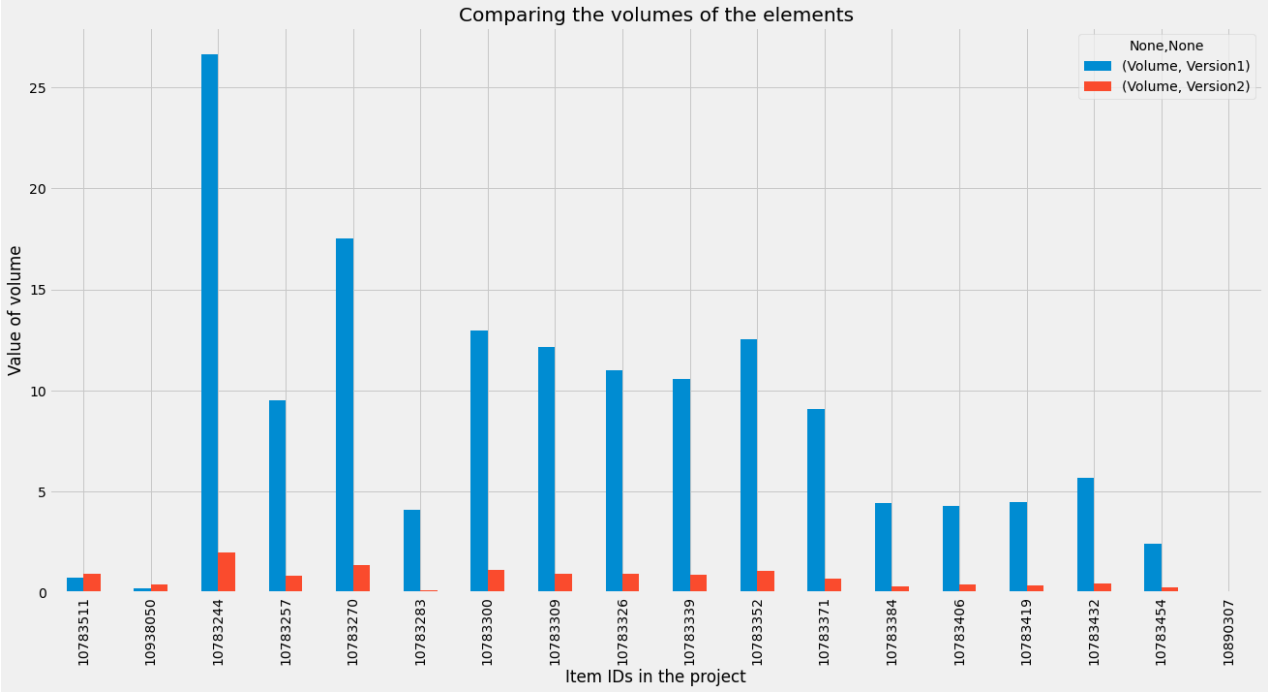
```
    if ax is None:
        size = (np.array(data.shape[::-1]) + np.array([0, 1])) * np.array([col_width, row_height])
        fig, ax = plt.subplots(figsize=size)
        ax.axis('off')

    mpl_table = ax.table(cellText=data.values,colLabels=data.columns,rowLabels=data.index, bbox=bbox,**kwargs)

    mpl_table.auto_set_font_size(False)
    mpl_table.set_fontsize(font_size)

    #for k, cell in  six.iteritems(mpl_table._cells):
    #    cell.set_edgecolor(edge_color)
    #    if k[0] == 0 or k[1] < header_columns:
    #        cell.set_text_props(weight='bold', color='w')
    #        cell.set_facecolor(header_color)
    #   else:
    #       cell.set_facecolor(row_colors[k[0]%len(row_colors) ])
    plt.tight_layout()
    plt.savefig("tabella.png")
    return ax


fig = render_mpl_table(df3, header_columns=0, col_width=2.0)
#plt.savefig("tabella.png",bbox_inches="tight")
#fig.savefig("table_mpl.png")
fig.get_figure().savefig('table_mpl2.png')
```

	Volume	Area	EB	EB Survey	Width	RC Bottom Face	RC Other Faces	RC Top Face	Foundation Thickne
(10783511, 'Version1')	0.75 m³	3 m²	nan	nan	nan	nan	nan	nan	nan
(10783511, 'Version2')	0.93 m³	4 m²	nan	nan	nan	nan	nan	nan	nan
(10938050, 'Version1')	0.22 m³	nan	nan	nan	nan	nan	nan	nan	nan
(10938050, 'Version2')	0.40 m³	nan	nan	nan	nan	nan	nan	nan	nan
(10783244, 'Version1')	26.66 m³	22 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783244, 'Version2')	1.97 m³	9 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783257, 'Version1')	9.51 m³	8 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783257, 'Version2')	0.81 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783270, 'Version1')	17.55 m³	14 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783270, 'Version2')	1.35 m³	6 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783283, 'Version1')	4.08 m³	3 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783283, 'Version2')	0.12 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783300, 'Version1')	12.96 m³	11 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783300, 'Version2')	1.11 m³	5 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783309, 'Version1')	12.13 m³	10 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783309, 'Version2')	0.93 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783326, 'Version1')	10.98 m³	9 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783326, 'Version2')	0.93 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783339, 'Version1')	10.56 m³	9 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783339, 'Version2')	0.89 m³	4 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783352, 'Version1')	12.56 m³	10 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783352, 'Version2')	1.05 m³	5 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783371, 'Version1')	9.07 m³	7 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783371, 'Version2')	0.70 m³	3 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783384, 'Version1')	4.41 m³	4 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783384, 'Version2')	0.29 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783406, 'Version1')	4.26 m³	3 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783406, 'Version2')	0.41 m³	2 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783419, 'Version1')	4.49 m³	4 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783419, 'Version2')	0.34 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783432, 'Version1')	5.65 m³	5 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783432, 'Version2')	0.43 m³	2 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10783454, 'Version1')	2.40 m³	2 m²	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10783454, 'Version2')	0.27 m³	1 m²	37570.0	37570.0	700.0	nan	nan	nan	230.0
(10890307, 'Version1')	0.00 m³	nan	36570.0	36570.0	1700.0	nan	nan	nan	1230.0
(10890307, 'Version2')	0.02 m³	nan	37570.0	37570.0	700.0	Rebar Cover 1	Rebar Cover 1	Rebar Cover 1	230.0

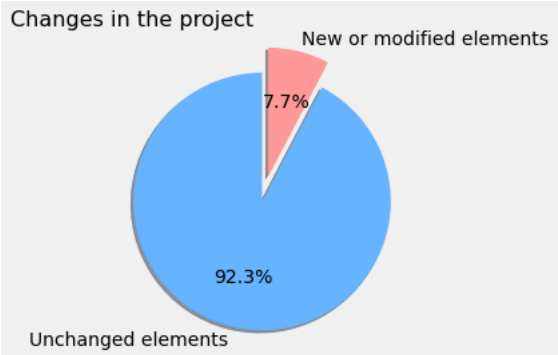
더블클릭 또는 Enter 키를 눌러 수정

```
# Modified elements with volumes

import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

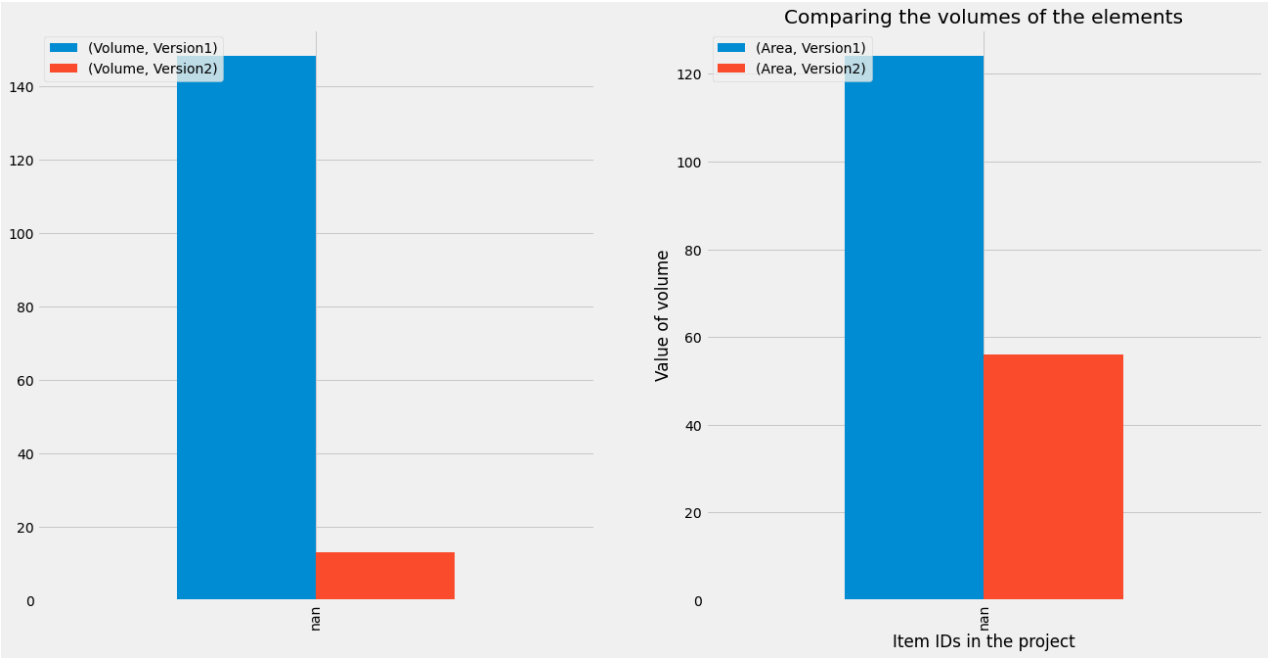
```
df1['Volume'] = df1['Volume'].replace('m³ ', '', regex=True)
df1['Area'] = df1['Area'].replace('m² ', '', regex=True)

df1[('Volume','Version1')] = pd.to_numeric(df1[('Volume','Version1')])
df1[('Volume','Version2')] = pd.to_numeric(df1[('Volume','Version2')])
df1[('Area','Version1')] = pd.to_numeric(df1[('Area','Version1')])
df1[('Area','Version2')] = pd.to_numeric(df1[('Area','Version2')])

df2 = pd.DataFrame(df1, columns=[('Volume','Version1'), ('Volume','Version2')])
df2.plot.bar(figsize=(20,10))

plt.title("Comparing the volumes of the elements")
plt.xlabel("Item IDs in the project")
plt.ylabel("Value of volume")
plt.savefig(fname = "category_comparison.png", bbox_inches='tight')
```




```
# In this visualization example, we can use the code to display data on the number of elements in
# which there are changes and display the percentage of changed elements to the number of all elements

import matplotlib.pyplot as plt

# Pie chart
labels = ['Unchanged elements', 'New or modified elements']
sizes = [ len(project_v2), len(df1)]
# only "explode" the 2nd slice (i.e. 'Hogs')
explode = (0, 0.2)
#add colors
colors = ['#66b3ff', '#ff9999']
fig1, ax1 = plt.subplots()
plt.title("Changes in the project", fontsize=16, fontweight=5, loc='left')
ax1.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
        shadow=True, startangle=90)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
#plt.tight_layout()
#plt.rcParams['figure.figsize'] = [10,6]
#plt.show()
plt.savefig(fname = "modifiedelements.png", transparent=True)
```




```
# Display only the changed data in the new table and show the values from the
# volume and area columns as a diagram

df3 = df1.groupby(by={'self': 'Version1'}, dropna=False).sum()
df4 = pd.DataFrame(df3, columns=[('Volume','Version1'), ('Volume','Version2')])
df5 = pd.DataFrame(df3, columns=[('Area','Version1'), ('Area','Version2')])

fig = plt.figure()
axes = fig.subplots(nrows=1, ncols=2)


df4.plot.bar(figsize=(20,10),ax=axes[0])
df5.plot.bar(figsize=(20,10),ax=axes[1])

axes[1].legend(loc=2)
axes[0].legend(loc=2)

plt.title("Comparing the volumes of the elements")
plt.xlabel("Item IDs in the project")
plt.ylabel("Value of volume")
plt.savefig(fname = "comparing_the_volumes.png", bbox_inches='tight')
```

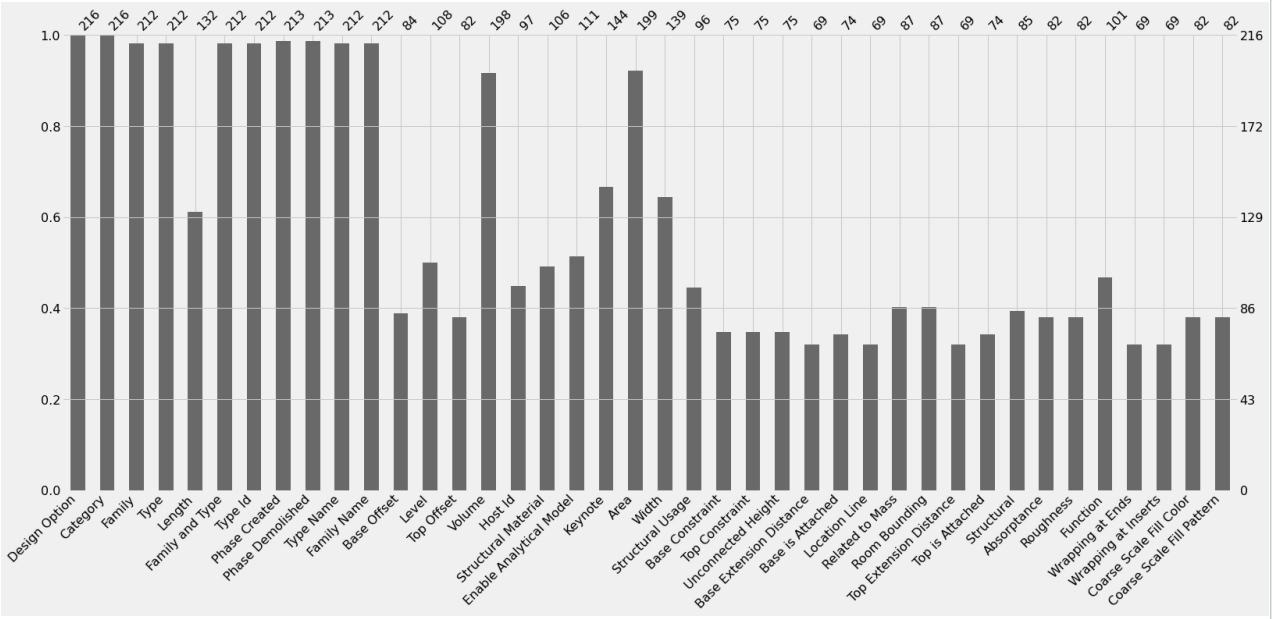


더블클릭 또는 Enter 키를 눌러 수정

```
# Remove parameters (columns) with few filled elements

df = project_v1
null_percentage = df.isnull().sum()/df.shape[0]*100
```

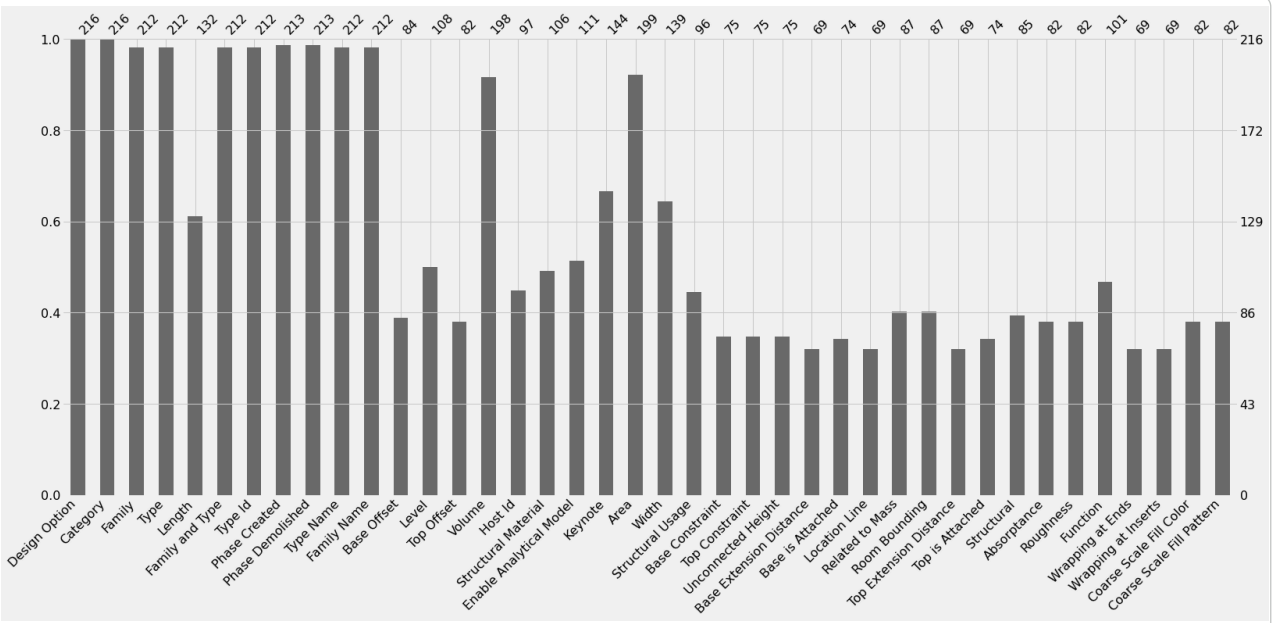
```
col_to_drop = null_percentage[null_percentage>70].keys()
df = df.drop(col_to_drop, axis=1)
```

```
# V1 Checking attributes in the project

fig2 = msno.bar(df)
fig_copy2 = fig2.get_figure()
fig_copy2.savefig('plot_pr2a.png', bbox_inches = 'tight')
```

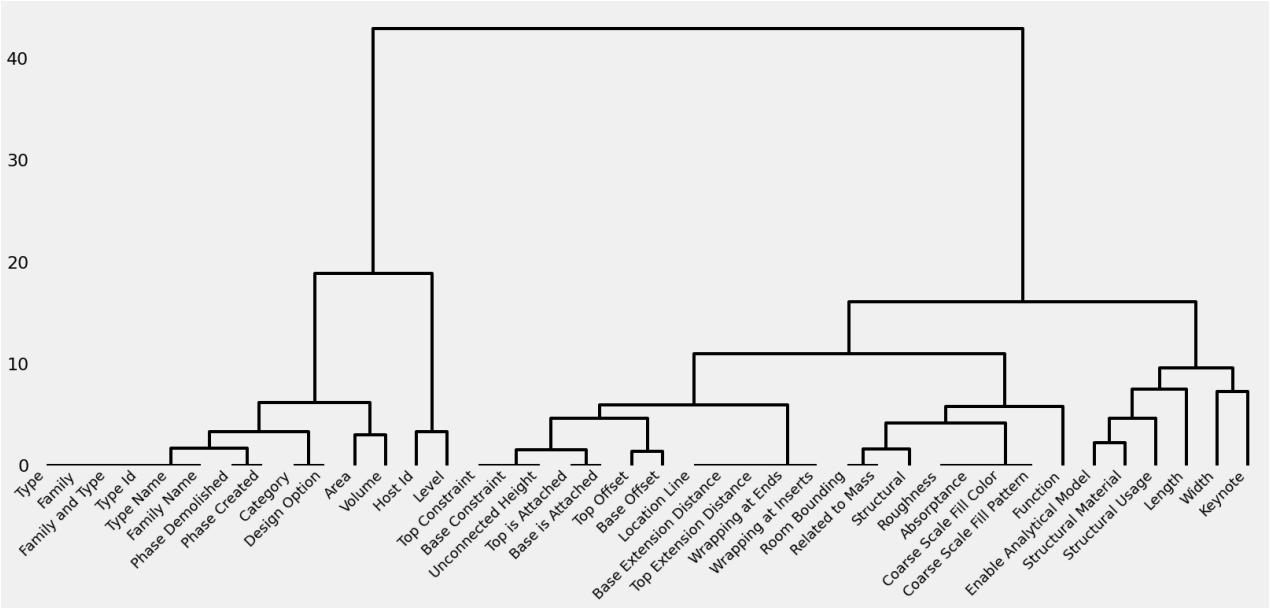



```
dfs = project_v2
null_percentage = dfs.isnull().sum()/dfs.shape[0]*100
col_to_drop = null_percentage[null_percentage>70].keys()
dfs = dfs.drop(col_to_drop, axis=1)
```

```
# V2 Checking attributes in the project

fig2 = msno.bar(dfs)
fig_copy2 = fig2.get_figure()
fig_copy2.savefig('plot_pr2as.png', bbox_inches = 'tight')
```




```
# A dendrogram is a diagram representing a tree. The figure factory called create_dendrogram
# performs hierarchical clustering on data and represents the resulting tree

fig = msno.dendrogram(df, orientation="top", method="ward", fontsize=18)
fig_copy = fig.get_figure()
fig_copy.savefig('dendrogramh.png', bbox_inches = 'tight', transparent=True)
```




```
# Translate the string values in the volume and area columns to sum the values for the groups

propstr = ['Area', 'Volume']

for el in propstr:
    df[el+'_str'] = df[el]

for el in propstr:
    df[el] = df[el].astype(str)
    df[el] = df[el].str.extract('(Wd*.?Wd*)')
    df[el] = df[el].fillna(0)
    df[el] = df[el].replace(r'n',0, regex=True)
    df[el] = df[el].astype(float)

df = df.groupby('Family')['Volume', 'Area'].sum()
df = df[(df.T != 0).any()]
df = df.round()
df = df[df['Area'] >1]


df['Volume'] = df['Volume'].astype(str) + " m³ "
df['Area'] = df['Area'].astype(str) + " m² "
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:15: FutureWarning: Indexing with multiple keys (implicitly converted to a t
  from ipykernel import kernelapp as app
```

```
# With a single group_by function, we can quickly display groups of elements and their main
# parameters using a table. In this example, we display all groups of elements for which the area and
# volume attributes have been summed in the grouping.



import numpy as np
import six
import matplotlib.pyplot as plt
def render_mpl_table(data, col_width=3.0, row_height=0.525, font_size=15,
                     header_color='#A85D1C', row_colors=['#C4551D', 'w'], edge_color='w',
                     bbox=[0, 0, 1, 1], header_columns=0,header_row=0,
                     ax=None, **kwargs):
```

```
    if ax is None:
        size = (np.array(data.shape[::-1]) + np.array([0, 1])) * np.array([col_width, row_height])
        fig, ax = plt.subplots(figsize=size)
        ax.axis('off')

    mpl_table = ax.table(cellText=data.values,colLabels=data.columns,rowLabels=data.index, bbox=bbox,**kwargs)

    mpl_table.auto_set_font_size(False)
    mpl_table.set_fontsize(font_size)

    for k, cell in  six.iteritems(mpl_table._cells):
        cell.set_edgecolor(edge_color)
        if k[0] == 0 or k[1] < header_columns:
            cell.set_text_props(weight='bold', color='w')
            cell.set_facecolor(header_color)
        else:
            cell.set_facecolor(row_colors[k[0]%len(row_colors) ])
    plt.tight_layout()
    #fig.savefig(pathro + "tabella3.png")
    return ax

middle = float(len(df))/2
dfp1 = df.iloc[:int(middle)]



fig = render_mpl_table(dfp1, header_columns=0, col_width=2.0, header_color='#A85D1C', row_colors=['#E4AD91', 'w'], edge_color='w')
#plt.savefig("tabella.png",bbox_inches="tight")
#fig.savefig("table_mpl.png")
fig.get_figure().savefig('table_mpl4.png', bbox_inches='tight')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:31: UserWarning: Tight layout not applied. The left and right margins cannc
```

	Volume	Area
0915 x 1220 mm	0.0 m³	2.0 m²
100 Masonary wall	10.0 m³	96.0 m²
15mm Tiles	0.0 m³	9.0 m²
200 Masonary wall	90.0 m³	453.0 m²
400x550mm	0.0 m³	2.0 m²
72" x 36" x 32"	0.0 m³	12.0 m²
BED_NEW_9026	2.0 m³	21.0 m²
BED_NEW_no Pillow	3.0 m³	36.0 m²
Bath_-_Deonne	0.0 m³	3.0 m²
Bearing Footing - 700x230	147.0 m³	121.0 m²
Bearing Footing - 900 x 300	4.0 m³	15.0 m²
Chair	1.0 m³	6.0 m²
Curtain Wall (AG)Flush	0.0 m³	2.0 m²
Curtain Wall 4	0.0 m³	4.0 m²
DA - 813Hx2135W	1.0 m³	21.0 m²
DB - 900x2134	0.0 m³	3.0 m²
DF - 3300Wx2135H	0.0 m³	9.0 m²
DG - 4800Wx2135	0.0 m³	12.0 m²
DH 1000x2135	0.0 m³	3.0 m²
DJ 2700Wx2135H	0.0 m³	8.0 m²
Default	6.0 m³	27.0 m²
Fencing	0.0 m³	6.0 m²
Generic 150mm	24.0 m³	161.0 m²

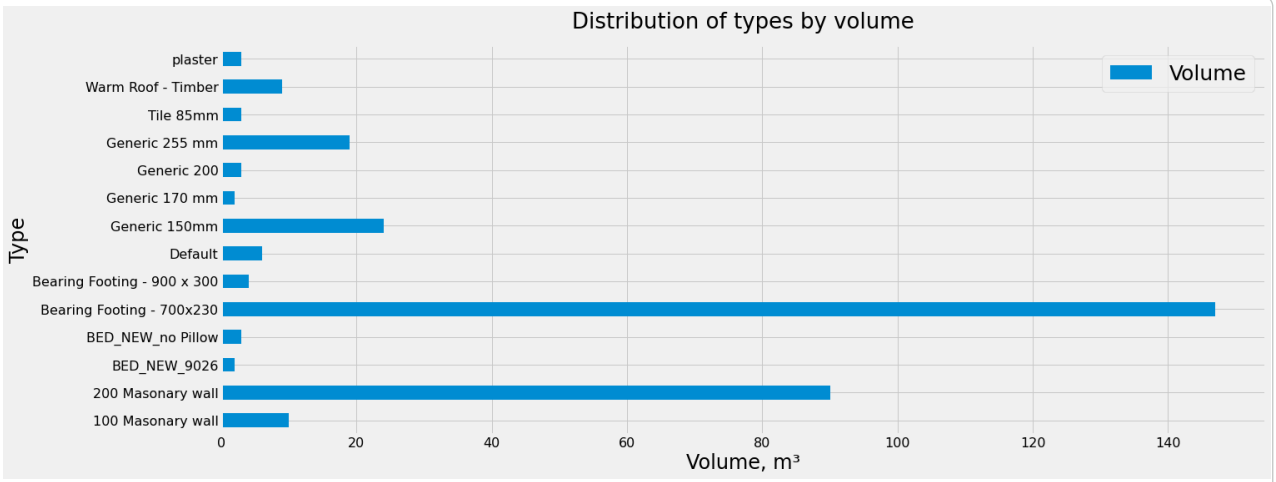
```
dfp2 = df.iloc[int(middle):]
fig = render_mpl_table(dfp2, header_columns=0, col_width=2.0, header_color='#A85D1C', row_colors=['#E4AD91', 'w'], edge_color='w')
fig.get_figure().savefig('table_mpl5.png', bbox_inches='tight')
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:31: UserWarning: Tight layout not applied. The left and right margins cannc
```

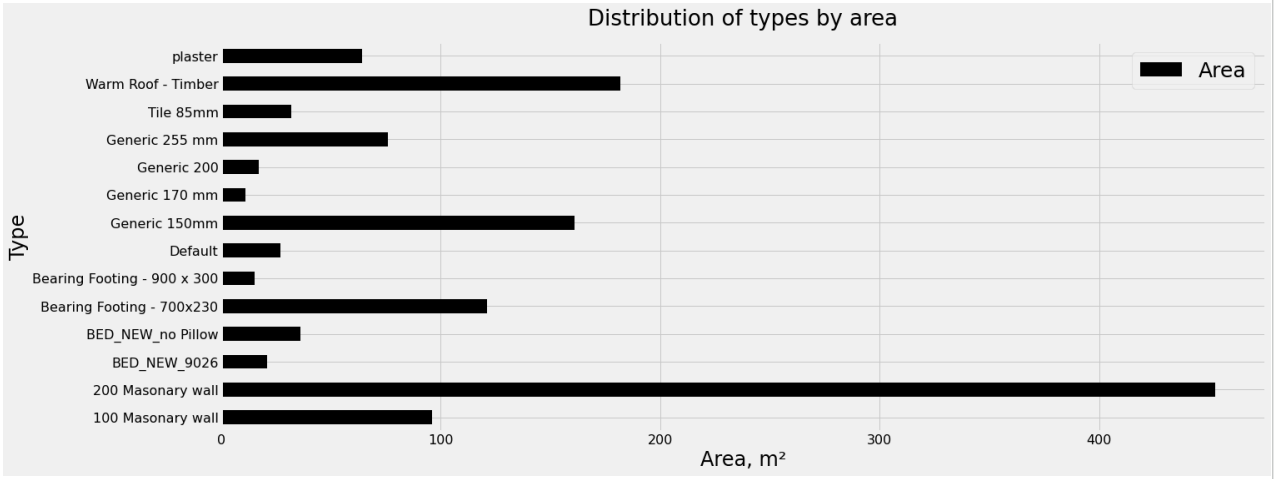
	Volume	Area
Generic 170 mm	2.0 m³	11.0 m²
Generic 200	3.0 m³	17.0 m²
Generic 255 mm	19.0 m³	76.0 m²
Generic 85mm	1.0 m³	7.0 m²
Glazed (AG)Flush glazing	0.0 m³	2.0 m²
Hospotality_table_11133	0.0 m³	2.0 m²
Mesh Fence	0.0 m³	6.0 m²
Settee	1.0 m³	15.0 m²
Shwr Pan 30 x 36	0.0 m³	3.0 m²
Simple_Parametric_Wardrobe_1800	1.0 m³	70.0 m²
Simple_Parametric_Wardrobe_2400	1.0 m³	53.0 m²
Tile 85mm	3.0 m³	32.0 m²
WA - Three Panel Window with Large Fixed Panel	0.0 m³	3.0 m²
WD - 1500Wx1800H	0.0 m³	6.0 m²
WE - 1200Wx1800H	0.0 m³	6.0 m²
WF - 2200Wx1800H	0.0 m³	12.0 m²
WG - Three Panel Window	0.0 m³	2.0 m²
WH - Three Panel Window 750	0.0 m³	6.0 m²
WK - Three Panel Window 600mm	0.0 m³	2.0 m²
Wall_Hung_WC_2427	0.0 m³	3.0 m²
Warm Roof - Timber	9.0 m³	182.0 m²
archiclassic (1)	0.0 m³	5.0 m²
plaster	3.0 m³	64.0 m²
w/- Tap VT0392B	0.0 m³	2.0 m²

```
propstr = ['Area', 'Volume']

for el in propstr:
    df[el+'_str'] = df[el]

for el in propstr:
    df[el] = df[el].astype(str)
    df[el] = df[el].str.extract('(Wd*.?Wd*)')
    df[el] = df[el].fillna(0)
    df[el] = df[el].replace(r'n',0, regex=True)
    df[el] = df[el].astype(float)
df = df[df['Volume'] >1]
```

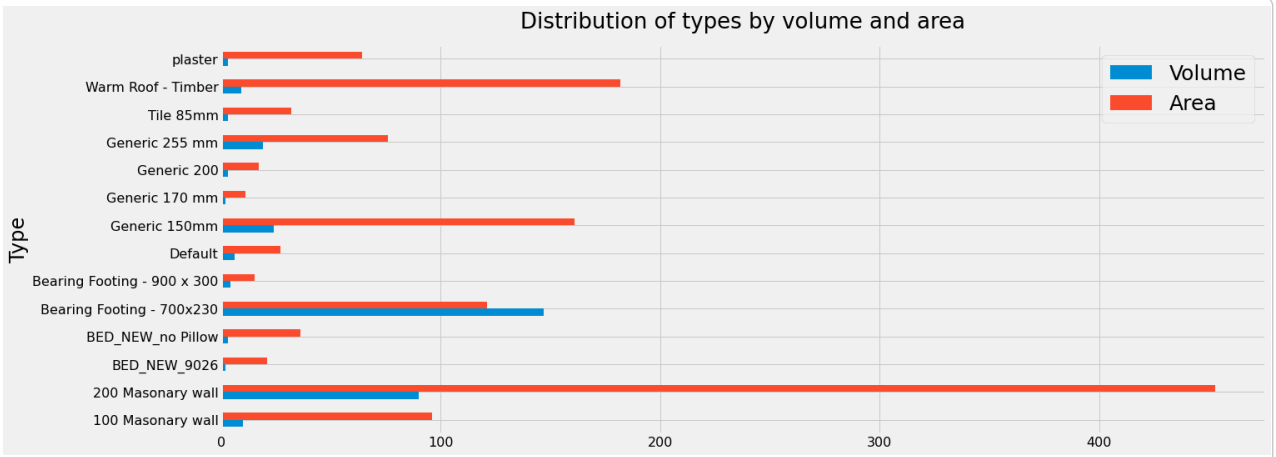
```
# Similar ready-made solutions and ready-made pieces of code exist today for any data problem and
# its visualization. Let us present the groups of model elements in the form of horizontal graphs

plt.rcParams['figure.figsize'] = [20,8]

ax1 = df.plot(y=['Volume'], kind="barh", title = 'Distribution by Volume', fontsize = 16)
ax1.legend(fontsize = 25)
ax1.set_ylabel('Type', fontdict={'fontsize':24})
ax1.set_xlabel('Volume, m³ ', fontdict={'fontsize':24})
ax1.set_title('Distribution of types by volume', fontdict={'fontsize':26}, pad=20)
plt.savefig('table_vol.png', bbox_inches = 'tight')
```




```
plt.rcParams['figure.figsize'] = [20,8]

ax1 = df.plot(y=['Area'], kind="barh", color='k', fontsize = 16)
ax1.legend(fontsize = 25)
ax1.set_ylabel('Type', fontdict={'fontsize':24})
ax1.set_xlabel('Area, m² ', fontdict={'fontsize':24})
ax1.set_title('Distribution of types by area', fontdict={'fontsize':26}, pad=20)
plt.savefig('table_area.png', bbox_inches = 'tight')
```



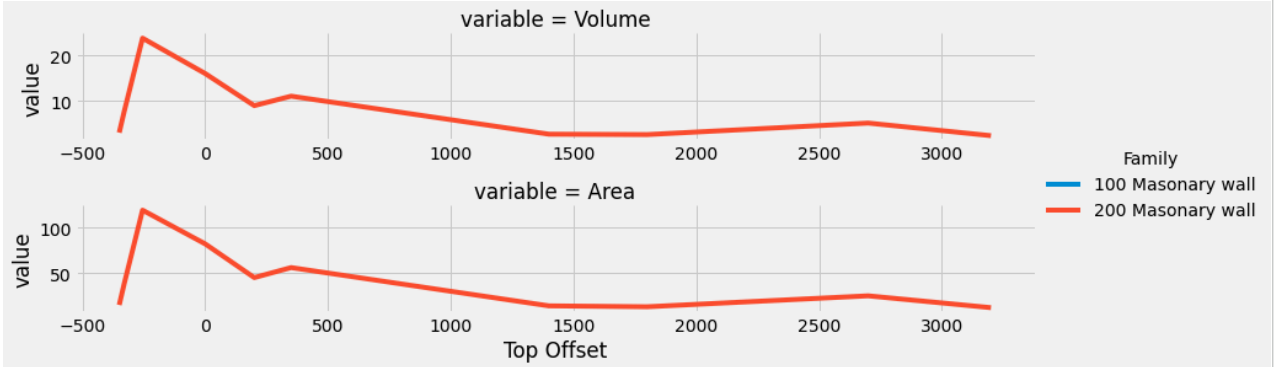

```
plt.rcParams['figure.figsize'] = [20,8]

ax1 = df.plot(y=['Volume', 'Area'], kind="barh", fontsize = 16)
ax1.legend(fontsize = 25)
ax1.set_ylabel('Type', fontdict={'fontsize':24})
ax1.set_title('Distribution of types by volume and area', fontdict={'fontsize':26}, pad=20)
plt.savefig('table_two.png', bbox_inches = 'tight')
```



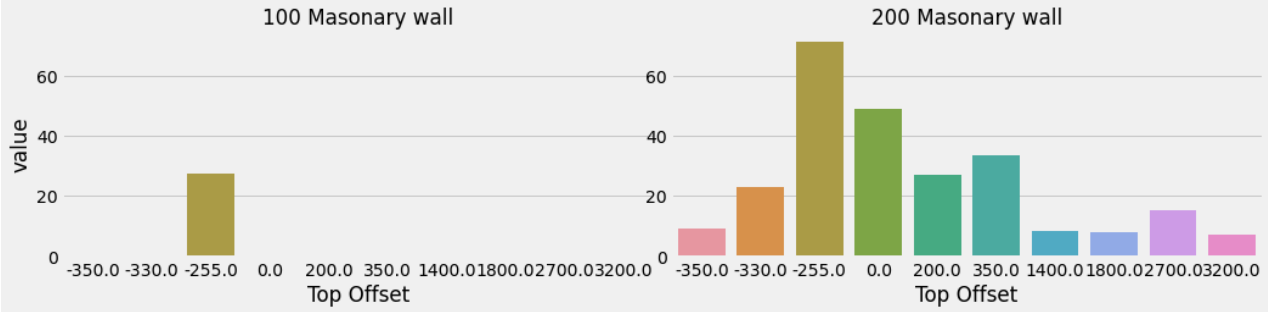

```
df = project_v1
for el in propstr:
    df[el+'_str'] = df[el]

propstr = ['Area', 'Volume']
for el in propstr:
    df[el] = df[el].astype(str)
    df[el] = df[el].str.extract('(\d*.?\d*)')
    df[el] = df[el].fillna(0)
    df[el] = df[el].replace(r'\n',0, regex=True)
    df[el] = df[el].astype(float)
df = df[df['Volume'] >1]

df = df.groupby(['Family', 'Top Offset'])['Volume', 'Area'].sum()
df = df.reset_index()
df = df.melt(id_vars=['Family', 'Top Offset'])
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Indexing with multiple keys (implicitly converted to a t
```

```
# In this project there are no floors and levels, a graph on which would allow to clearly show the
# representation of volumes or areas by groups and additionally by floors. In this example, we can
# show the distribution of volumes and areas depending on the Offset value, which allows you to
# quickly determine the model saturation not by geometry, but by data


import seaborn as sns

#df = df[df.Family != '100 Masonary wall']
sns.relplot(data=df, kind='line', col='variable', col_wrap=1, x='Top Offset', y='value', hue='Family', aspect=5.5,
            height=2.2, facet_kws={'sharey': False, 'sharex': False})
plt.savefig('replot.png')
```



```
g = sns.catplot(kind='bar', data=df, col='Family', x='Top Offset', y='value', col_wrap = 3, order=sorted(df['Top Offset'].unique()),
            col_order=sorted(df.Family.unique()), legend = True, ci=None, height=3.8, aspect=1.9, dodge=False)

(g.set_titles("{col_name}")
  .set_xlabels(label=None, clear_inner=False)
  .despine(left=False))
sns.despine(offset=1, trim=True)
plt.savefig('catplot.png')
```




```
pip install fpdf==1.7.2
```

```
Collecting fpdf==1.7.2
  Downloading fpdf-1.7.2.tar.gz (39 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: fpdf
  Building wheel for fpdf (setup.py) ... done
  Created wheel for fpdf: filename=fpdf-1.7.2-py2.py3-none-any.whl size=40722 sha256=eff43c149efcd19a98e42af78ef2bf81f8ec1d195afd448b72a
  Stored in directory: /root/.cache/pip/wheels/d7/ca/c8/86467e7957bbbcbdf4cf4870fc7dc95e9a16404b2e3c3a98c3
Successfully built fpdf
Installing collected packages: fpdf
Successfully installed fpdf-1.7.2
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It i
Note: you may need to restart the kernel to use updated packages.
```

```
from datetime import date
today = date.today()

# dd/mm/YY
d1 = today.strftime("%d/%m/%Y")
from fpdf import FPDF
pathr = "../input/pdf-report/"
```

OpenDataBIM free construction data from proprietary software & formats ecosystem

OpenDataBIM free construction data from proprietary software & formats ecosystem to use it in long-established data processing tools. OpenDataBIM lets customers to fuel with data many business cases in construction, maintenance processes in standard and affordable way and automate processes with free tools. A new OpenDataBIM Converter 2.0 is now available! Join now and get get your Data in an open format. Take the complex data out of your processes and become the one who can use open and simple data!

More on the NoBIM concept: opendatabim.io

```
# Creating a header and footer in a PDF document

class PDF(FPDF):
    def header(self):
        # Logo
        self.image(pathr + 'OpenDataBIMlogo2.jpg', 17, 5, 50)
        # Arial bold 15
        self.set_font('Arial', 'B', 12)
        # Move to the right
        self.cell(80)
        # Title
        self.cell(200, 6, 'CHECK REPORT-001', 50, 100, 'L')
        self.cell(100, 6, 'Compare two versions of a project ', 50, 100,  'L')
        self.cell(100, 6, 'Revit or IFC Projects', 50, 100, 'L')
        self.set_font('Arial', '', 12)
        pdf.set_text_color(70,75,145)
        self.cell(200, 6, 'opendatabim.io', 50, 3, 'L',  link = 'https://opendatabim.io/')
        pdf.set_text_color(255,255,255)
```

```
    # Page footer
    def footer(self):
        # Position at 1.5 cm from bottom
        self.set_y(-35)
        # Arial italic 8
        self.set_font('Arial', '', 8)
        # Page number
        self.image(pathr + 'OD.jpg', x = 50, y = None, w = 100, h = 0, type = '', link = 'https://opendatabim.io/')
        self.cell(0, 10, 'Page ' + str(self.page_no()) + '/{nb}' + '         OpenDataBIM Ltd.       ' + d1, 0, 0, 'C')


    def chapter_title(self, num, label):
        # Arial 12
        pdf.cell(20, 5, '', 2, 1, 'L')

        self.cell(1,30)
        self.set_text_color(255, 255, 255)
        self.set_font('Arial', 'B', 12)
        # Background color
        self.set_fill_color(60, 90, 154)
        # Title
        self.cell(0, 10, 'Part %d : %s' % (num, label), 0, 1, 'L', 1)
        # Line break
        pdf.set_text_color(0, 0, 0)
        self.ln(4)
    def chapter_body(self, name):
        # Read text file
        with open(name, 'rb') as fh:
            txt = fh.read().decode('latin-1')
        # Times 12
        self.set_font('Arial', '', 12)
        # Output justified text
        self.multi_cell(0, 5, txt)
        # Line break
        self.ln()
        # Mention in italics
        self.set_font('', 'B')

    def print_chapter(self, num, title, name):
        self.chapter_title(num, title)
        self.chapter_body(name)
```

Modern data pipelines automate many of the manual steps

A data pipeline is a means of moving data from one place (the source) to a destination (such as a data warehouse or PDF). Along the way, data is transformed and optimized, arriving in a state that can be analyzed and used to develop business insights. A data pipeline essentially is the steps involved in aggregating, organizing, and moving data. Modern data pipelines automate many of the manual steps involved in transforming and optimizing continuous data loads. Typically, this includes loading raw data into a staging table for interim storage and then changing it before ultimately inserting it into the destination reporting tables.

In our case, we run our data from the models through Pipeline to produce summary tables or automatic PDF reports.

More on the NoBIM concept: opendatabim.io

```
# Generating a PDF document from the data obtained for these files

pdf = PDF()

pdf.add_page()
pdf.image(pathr + 'white.png', x = 0, y = 0, w = 900, h = 1000, type = '', link = '')
pdf.image(pathr + 'OpenDataBIMlogo.jpg', x = 40, y = 25, w = 130, type = '', link = '')
pdf.image(pathr + 'grey.png', x = 0, y = 80, w = 300, h = 30, type = '', link = '')
pdf.set_text_color(255, 255, 255)
pdf.set_font('Arial', 'B', 16)
pdf.cell(20, 53, '' , 2, 1, 'C')
pdf.cell(190, 8, 'CHECK REPORT-001' , 2, 1, 'C')
pdf.cell(190, 8, 'COMPARE VERSIONS', 2, 1, 'C')
pdf.image(pathr +'comparetwoprojects.jpg', x = 15, y =150, w = 170, type = '', link = '')



# Background color

# Title
```

```
pdf.set_text_color(0, 0, 0)
#pdf.chapter_titlefs(4,'sdfdfgdg')
#pdf.cell(50, 7, '!This PDF document was created automatically with JN from RVT files ', 2, 1, 'C')
#pdf.cell(50, 7, 'without using plugins, Revit or Forge', 2, 1, 'C')
pdf.alias_nb_pages()
pdf.set_font('Arial', '', 12)


pdf.add_page()
pdf.set_font('Arial', 'B', 14)
pdf.cell(20, 10, '', 2, 1, 'L')

pdf.cell(200, 7, '!This PDF document was created automatically from RVT files ', 2, 1, 'L')
pdf.cell(5, 7, 'without using plugins, Revit or Forge', 2, 1, 'L')
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.set_font('Arial', '', 12)
pdf.image(pathr + 'skript.jpg', 12, 60, 25)

pdf.cell(30)
pdf.cell(10, 7, 'Link to the Pipeline code that generated this document: ', 10, 50, 'L')
pdf.set_font('','U')
pdf.set_text_color(0,102,204)
pdf.cell(10, 7, 'JupyterLab Pipeline: Create a data report', 10, 50, 'L', link = "https://www.kaggle.com/code/artemboiko/comparison-o
pdf.set_font('','')

pdf.cell(20, 5, '', 2, 1, 'L')
pdf.print_chapter(1, 'Go to noBIM, Work Efficiently', pathr + 'pdf_1.txt')
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.image(pathr + 'OpenDataBIM.jpg', x = 15, y = None, w = 180, h = 0, type = '', link = '')
pdf.set_font('', 'B', 14)
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.cell(40,10, 'Compare two versions of a project: ', 2, 1)
pdf.set_font('', '', 12)
v1names = v1name + '.rvt'
v2names =  v2name + '.rvt'
pdf.cell(40,5, v1names, 2, 1)
pdf.cell(40,5, v2names, 2, 1)

pdf.add_page()
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.print_chapter(2, 'NoBIM concept', pathr + 'pdf_2.txt')
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.image(pathr + 'noBIM.jpg', x = 25, y = 95, w = 150, h = 0, type = '', link = '')

pdf.add_page()
pdf.print_chapter(3, 'Tasks that rely on tabular data', pathr + 'pdf_3.txt')
pdf.cell(40,10, 'Example of a graph creation block ', 2, 1)
pdf.image(pathr + 'Exampe_Code.png', x = 15, y = None, w = 160, h = 0, type = '', link = '')
pdf.cell(40,10, 'An example of generating a page in a PDF document', 2, 1)
pdf.image(pathr + 'Exampe_Code2.png', x = 15, y = None, w = 160, h = 0, type = '', link = '')

pdf.cell(20, 5, '', 2, 1, 'L')
pdf.set_font('Arial', '', 12)
pdf.image(pathr + 'skript.jpg', 12, 190, 25)
pdf.cell(30)

pdf.cell(10, 7, 'Link to the Pipeline code that generated this document: ', 10, 50, 'L')
pdf.set_font('','U')
pdf.set_text_color(0,102,204)
pdf.cell(10, 7, 'JupyterLab Pipeline: Create a data report', 10, 50, 'L', link = "https://www.kaggle.com/code/artemboiko/comparison-o
pdf.set_font('','')
pdf.set_text_color(255,255,255)

pdf.add_page()
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.print_chapter(4, 'V1 Checking attributes in the project', pathr + 'pdf_4.txt')
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.cell(40,5,v1name+ '.rvt', 2, 1)
pdf.image('./plot_pr2.png', x = 15, y = None, w = 180, h = 0, type = '', link = '')
pdf.cell(40,20,v1name+ '.rvt', 2, 1)
pdf.image('./plot_pr2a.png', x = 15, y = None, w = 180, h = 0, type = '', link = '')


pdf.add_page()
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.print_chapter(5, 'V2 Checking attributes in the project', pathr + 'pdf_5.txt')
pdf.cell(20, 5, '', 2, 1, 'L')
pdf.cell(40,5,v2name+ '.rvt', 2, 1)
pdf.image('./plot_pr1.png', x = 15, y = None, w = 180, h = 0, type = '', link = '')
pdf.cell(40,20,v2name+ '.rvt', 2, 1)
pdf.image('./plot_pr2as.png', x = 15, y = None, w = 180, h = 0, type = '', link = '')
```

```
pdf.add_page()
pdf.print_chapter(6, 'Group the large project table into any number of smaller tables', pathr + 'pdf_6.txt')

pdf.cell(20, 5, '', 2, 1, 'L')
```