

```

# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = ':https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F51153%2F95503%2Fbundle%2Farchive.zip%3FX-Goog-Algorit

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
    os.symlink(KAGGLE_INPUT_PATH, os.path.join(".", 'input'), target_is_directory=True)
except FileExistsError:
    pass
try:
    os.symlink(KAGGLE_WORKING_PATH, os.path.join(".", 'working'), target_is_directory=True)
except FileExistsError:
    pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
                with ZipFile(tfile) as zfile:
                    zfile.extractall(destination_path)
            else:
                with tarfile.open(tfile.name) as tarfile:
                    tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')

Downloading , 189065 bytes compressed
[=====] 189065 bytes downloaded
Downloaded and uncompressed:
Data source import complete.

```

```
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%config InlineBackend.figure_format = 'retina'
sns.set(style="ticks")
plt.rc('figure', figsize=(6, 3.7), dpi=100)
plt.rc('axes', labelpad=20, facecolor="#ffffff",
      linewidth=0.4, grid=True, labelsz=10)
plt.rc('patch', linewidth=0)
plt.rc('xtick.major', width=0.2)
plt.rc('ytick.major', width=0.2)
plt.rc('grid', color='EEEEEE', linewidth=0.25)
plt.rc('font', family='Arial', weight='400', sz=10)
plt.rc('text', color='#282828')
plt.rc('xtick', labelsz=10)
plt.rc('ytick', labelsz=10)
plt.rc('savefig', pad_inches=0.3, dpi=300)
```

```
import pandas as pd
import numpy as np
```

```
dataset = pd.read_csv("../input/AmesHousing.csv")
```

```
# Configuring float numbers format
pd.options.display.float_format = '{:20.2f}'.format
dataset.head(n=5)
```



	Order	PID	MS SubClass	MS Zoning	Lot Frontage	Lot Area	Street	Alley	Lot Shape	Land Contour
0	1	526301100	20	RL	141.00	31770	Pave	NaN	IR1	Lvl
1	2	526350040	20	RH	80.00	11622	Pave	NaN	Reg	Lvl
2	3	526351010	20	RL	81.00	14267	Pave	NaN	IR1	Lvl
3	4	526353030	20	RL	93.00	11160	Pave	NaN	Reg	Lvl
4	5	527105010	60	RL	74.00	13830	Pave	NaN	IR1	Lvl

5 rows x 82 columns

```
dataset.describe(include=[np.number], percentiles=[.5]) \
    .transpose().drop("count", axis=1)
```

	mean	std	min	50%	max
Order	1465.50	845.96	1.00	1465.50	2930.00
PID	714464496.99	188730844.65	526301100.00	535453620.00	1007100110.00
MS SubClass	57.39	42.64	20.00	50.00	190.00
Lot Frontage	69.22	23.37	21.00	68.00	313.00
Lot Area	10147.92	7880.02	1300.00	9436.50	215245.00
Overall Qual	6.09	1.41	1.00	6.00	10.00
Overall Cond	5.56	1.11	1.00	5.00	9.00
Year Built	1971.36	30.25	1872.00	1973.00	2010.00
Year Remod/Add	1984.27	20.86	1950.00	1993.00	2010.00
Mas Vnr Area	101.90	179.11	0.00	0.00	1600.00
BsmtFin SF 1	442.63	455.59	0.00	370.00	5644.00
BsmtFin SF 2	49.72	169.17	0.00	0.00	1526.00
Bsmt Unf SF	559.26	439.49	0.00	466.00	2336.00
Total Bsmt SF	1051.61	440.62	0.00	990.00	6110.00
1st Flr SF	1159.56	391.89	334.00	1084.00	5095.00
2nd Flr SF	335.46	428.40	0.00	0.00	2065.00
Low Qual Fin SF	4.68	46.31	0.00	0.00	1064.00
Gr Liv Area	1499.69	505.51	334.00	1442.00	5642.00
Bsmt Full Bath	0.43	0.52	0.00	0.00	3.00
Bsmt Half Bath	0.06	0.25	0.00	0.00	2.00
Full Bath	1.57	0.55	0.00	2.00	4.00
Half Bath	0.38	0.50	0.00	0.00	2.00
Bedroom AbvGr	2.85	0.83	0.00	3.00	8.00
Kitchen AbvGr	1.04	0.21	0.00	1.00	3.00
TotRms AbvGrd	6.44	1.57	2.00	6.00	15.00
Fireplaces	0.60	0.65	0.00	1.00	4.00
Garage Yr Blt	1978.13	25.53	1895.00	1979.00	2207.00
Garage Cars	1.77	0.76	0.00	2.00	5.00
Garage	172.00	215.00	0.00	100.00	1400.00

```
# Getting the number of missing values in each column
num_missing = dataset.isna().sum()
# Excluding columns that contains 0 missing values
num_missing = num_missing[num_missing > 0]
# Getting the percentages of missing values
percent_missing = num_missing * 100 / dataset.shape[0]
# Concatenating the number and perecentage of missing values
# into one dataframe and sorting it
pd.concat([num_missing, percent_missing], axis=1,
          keys=['Missing Values', 'Percentage']).\
    sort_values(by="Missing Values", ascending=False)
```

	Missing Values	Percentage	
Pool QC	2917	99.56	
Misc Feature	2824	96.38	
Alley	2732	93.24	
Fence	2358	80.48	
Fireplace Qu	1422	48.53	
Lot Frontage	490	16.72	
Garage Cond	159	5.43	
Garage Qual	159	5.43	
Garage Finish	159	5.43	
Garage Yr Blt	159	5.43	
Garage Type	157	5.36	
Bsmt Exposure	83	2.83	
BsmtFin Type 2	81	2.76	
BsmtFin Type 1	80	2.73	
Bsmt Qual	80	2.73	
Bsmt Cond	80	2.73	
Mas Vnr Area	23	0.78	
Mas Vnr Type	23	0.78	
Bsmt Half Bath	2	0.07	
Bsmt Full Bath	2	0.07	
Total Bsmt SF	1	0.03	
Bsmt Unf SF	1	0.03	
Garage Cars	1	0.03	
Garage Area	1	0.03	
BsmtFin SF 2	1	0.03	
BsmtFin SF 1	1	0.03	
Electrical	1	0.03	

```
dataset["Pool Area"].value_counts()
```

```
0      2917
144      1
480      1
576      1
555      1
368      1
444      1
228      1
561      1
519      1
648      1
800      1
512      1
738      1
Name: Pool Area, dtype: int64
```

```
dataset["Pool QC"].fillna("No Pool", inplace=True)
```

```
dataset["Misc Val"].value_counts()
```

```
0      2827
400     18
500     13
450      9
600      8
700      7
2000     7
1500      3
1200      3
650      3
480      2
3000      2
2500      2
```

```
4500      2
455       1
1512      1
17000     1
1000      1
15500     1
460       1
8300      1
1300      1
560       1
620       1
900       1
1150      1
6500      1
1400      1
750       1
800       1
12500     1
350       1
490       1
80        1
54        1
3500      1
300       1
420       1
Name: Misc Val, dtype: int64
```

```
dataset['Misc Feature'].fillna('No feature', inplace=True)
```

```
dataset['Alley'].fillna('No Alley', inplace=True)
dataset['Fence'].fillna('No Fence', inplace=True)
dataset['Fireplace Qu'].fillna('No Fireplace', inplace=True)
```

```
dataset['Lot Frontage'].fillna(0, inplace=True)
```

```
garage_columns = [col for col in dataset.columns if col.startswith("Garage")]
dataset[dataset['Garage Cars'].isna()][garage_columns]
```

	Garage Type	Garage Yr Blt	Garage Finish	Garage Cars	Garage Area	Garage Qual	Garage Cond
2236	Detchd	NaN	NaN	NaN	NaN	NaN	NaN

```
dataset[~pd.isna(dataset['Garage Type']) &
pd.isna(dataset['Garage Qual'])][garage_columns]
```

	Garage Type	Garage Yr Blt	Garage Finish	Garage Cars	Garage Area	Garage Qual	Garage Cond
1356	Detchd	NaN	NaN	1.00	360.00	NaN	NaN
2236	Detchd	NaN	NaN	NaN	NaN	NaN	NaN

```
dataset['Garage Cars'].fillna(0, inplace=True)
dataset['Garage Area'].fillna(0, inplace=True)
```

```
dataset.loc[~pd.isna(dataset['Garage Type']) &
pd.isna(dataset['Garage Qual']), "Garage Type"] = "No Garage"
```

```
for col in ['Garage Type', 'Garage Finish', 'Garage Qual', 'Garage Cond']:
    dataset[col].fillna('No Garage', inplace=True)
```

```
dataset['Garage Yr Blt'].fillna(0, inplace=True)
```

```
bsmt_columns = [col for col in dataset.columns if "Bsmt" in col]
dataset[dataset['Bsmt Half Bath'].isna()][bsmt_columns]
```

	Bsmt Qual	Bsmt Cond	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF	Bsmt Full Bath	Bsmt Half Bath
1341	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
dataset[~pd.isna(dataset['Bsmt Cond']) &
pd.isna(dataset['Bsmt Exposure'])][bsmt_columns]
```

	Bsmt Qual	Bsmt Cond	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF	Bsmt Full Bath
66	Gd	TA	NaN	Unf	0.00	Unf	0.00	1595.00	1595.00	0.00
1796	Gd	TA	NaN	Unf	0.00	Unf	0.00	725.00	725.00	0.00

```
dataset[~pd.isna(dataset['Bsmt Cond']) &
pd.isna(dataset['BsmtFin Type 2'])][bsmt_columns]
```

	Bsmt Qual	Bsmt Cond	Bsmt Exposure	BsmtFin Type 1	BsmtFin SF 1	BsmtFin Type 2	BsmtFin SF 2	Bsmt Unf SF	Total Bsmt SF	Bsmt Full Bath
--	-----------	-----------	---------------	----------------	--------------	----------------	--------------	-------------	---------------	----------------

```
for col in ["Bsmt Half Bath", "Bsmt Full Bath", "Total Bsmt SF",
            "Bsmt Unf SF", "BsmtFin SF 2", "BsmtFin SF 1"]:
    dataset[col].fillna(0, inplace=True)
```

```
dataset.loc[~pd.isna(dataset['Bsmt Cond']) &
pd.isna(dataset['Bsmt Exposure']), "Bsmt Exposure"] = "No"
dataset.loc[~pd.isna(dataset['Bsmt Cond']) &
pd.isna(dataset['BsmtFin Type 2']), "BsmtFin Type 2"] = "Unf"
```

```
for col in ["Bsmt Exposure", "BsmtFin Type 2",
            "BsmtFin Type 1", "Bsmt Qual", "Bsmt Cond"]:
    dataset[col].fillna("No Basement", inplace=True)
```

```
dataset['Mas Vnr Area'].fillna(0, inplace=True)
dataset['Mas Vnr Type'].fillna("None", inplace=True)
```

```
dataset['Electrical'].fillna(dataset['Electrical'].mode()[0], inplace=True)
```

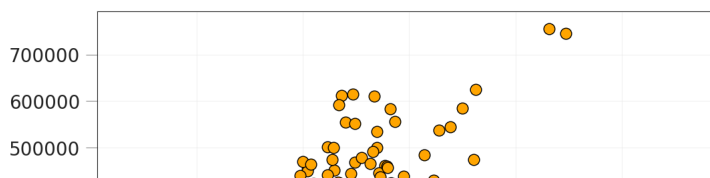
```
dataset.isna().values.sum()
```

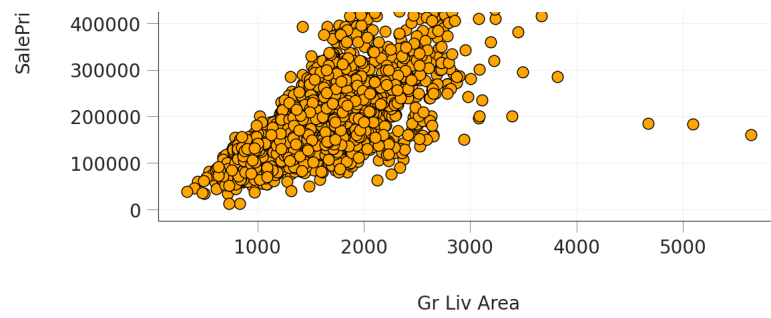
0

```
from matplotlib import pyplot as plt
import seaborn as sns
```

```
plt.scatter(x=dataset['Gr Liv Area'], y=dataset['SalePrice'],
            color="orange", edgecolors="#000000", linewidths=0.5);
plt.xlabel("Gr Liv Area"); plt.ylabel("SalePrice");
```

U





```
outlirt_columns = ["Gr Liv Area"] + \
    [col for col in dataset.columns if "Sale" in col]
dataset[dataset["Gr Liv Area"] > 4000][outlirt_columns]
```

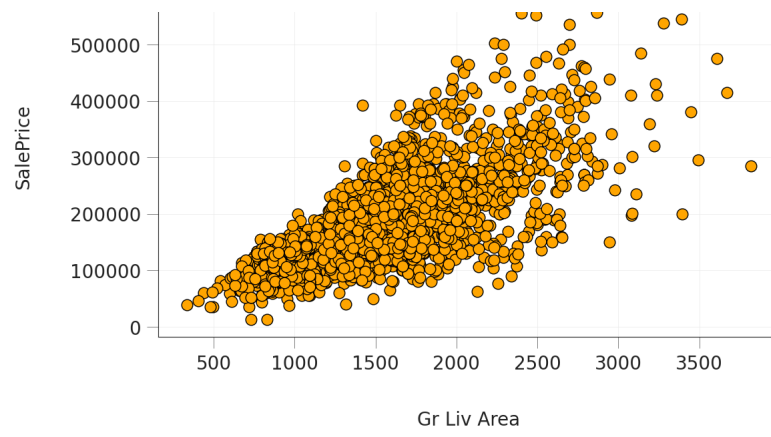
	Gr Liv Area	Sale Type	Sale Condition	SalePrice	
1498	5642	New	Partial	160000	
1760	4476	WD	Abnorml	745000	
1767	4316	WD	Normal	755000	
2180	5095	New	Partial	183850	
2181	4676	New	Partial	184750	

```
dataset = dataset[dataset["Gr Liv Area"] < 4000]
```

```
plt.scatter(x=dataset['Gr Liv Area'], y=dataset['SalePrice'],
            color="orange", edgecolors="#000000", linewidths=0.5);
plt.xlabel("Gr Liv Area"); plt.ylabel("SalePrice");
```


600000

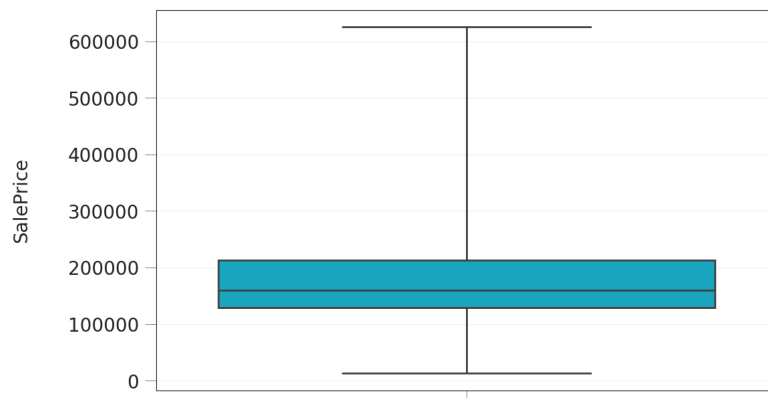




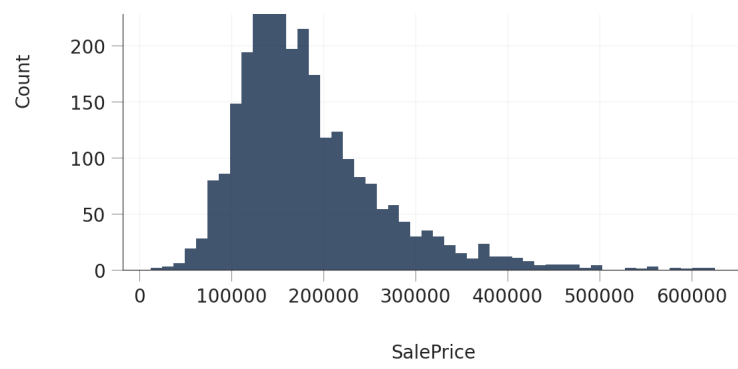
```
dataset.reset_index(drop=True, inplace=True)
```

```
dataset.drop(['Order', 'PID'], axis=1, inplace=True)
```

```
sns.boxplot(dataset['SalePrice'], whis=10, color="#00B8D9");
```

[illegible]

```
sns.distplot(dataset['SalePrice'], kde=False,
              color="#172B4D", hist_kws={"alpha": 0.8});
plt.ylabel("Count");
```

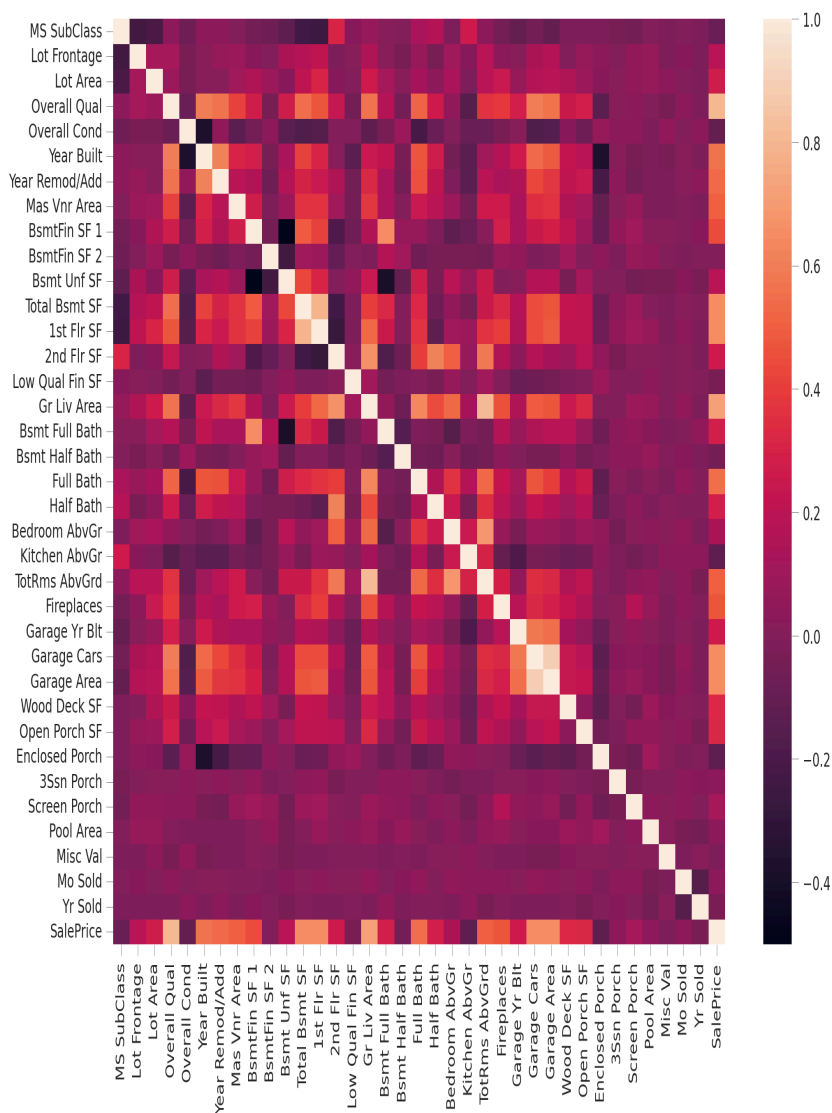
```
fig, ax = plt.subplots(figsize=(12,9))
sns.heatmap(dataset.corr(), ax=ax);
```

[illegible]

[illegible]

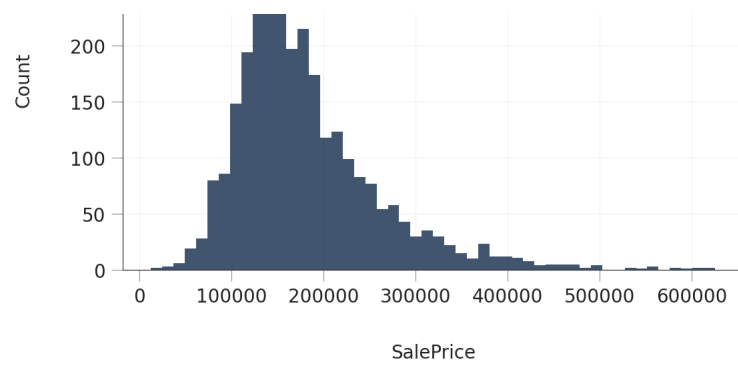
[illegible]

[illegible]

[illegible]

```
sns.distplot(dataset['SalePrice'], kde=False,  
             color="#172B4D", hist_kws={"alpha": 0.8});  
plt.ylabel("Count");
```

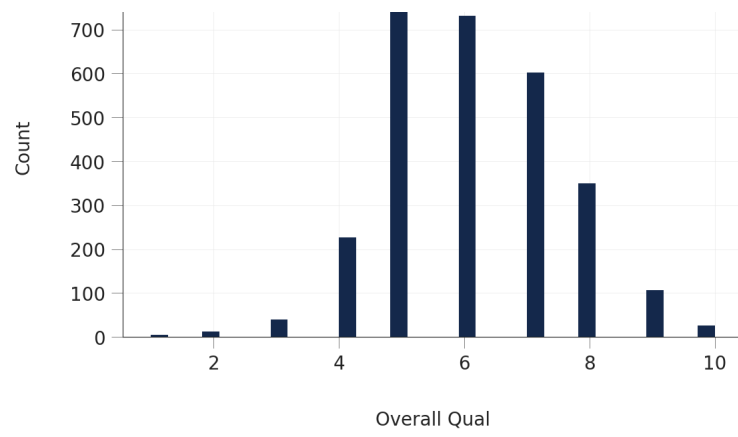
Gender	Number of People
Male	320
Female	240



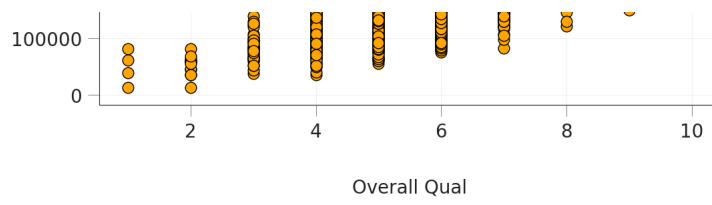
We can see that most house prices fall between 100,000 and 200,000. We see also that there is a number of expensive houses to the right of the plot. Now, we move to see the distribution of Overall Qual variable:

```
sns.distplot(dataset['Overall Qual'], kde=False,  
             color="#172B4D", hist_kws={"alpha": 1});  
plt.ylabel("Count");
```

800



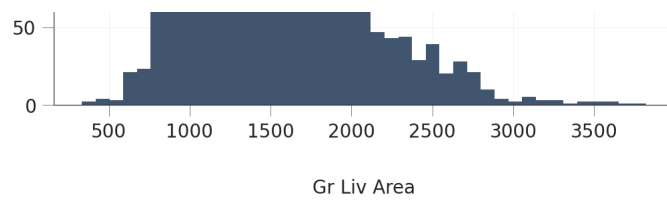
```
plt.scatter(x=dataset['Overall Qual'], y=dataset['SalePrice'],  
            color="orange", edgecolors="#000000", linewidths=0.5);  
plt.xlabel("Overall Qual"); plt.ylabel("SalePrice");
```

```
sns.distplot(dataset['Gr Liv Area'], kde=False,  
             color="#172B4D", hist_kws={"alpha": 0.8});  
plt.ylabel("Count");
```

A histogram showing the frequency of children per family. The x-axis is labeled 'Number of children' and ranges from 0 to 10. The y-axis is labeled 'Count' and ranges from 0 to 250. The bars are dark blue. The distribution peaks at 4 children with a count of approximately 235.

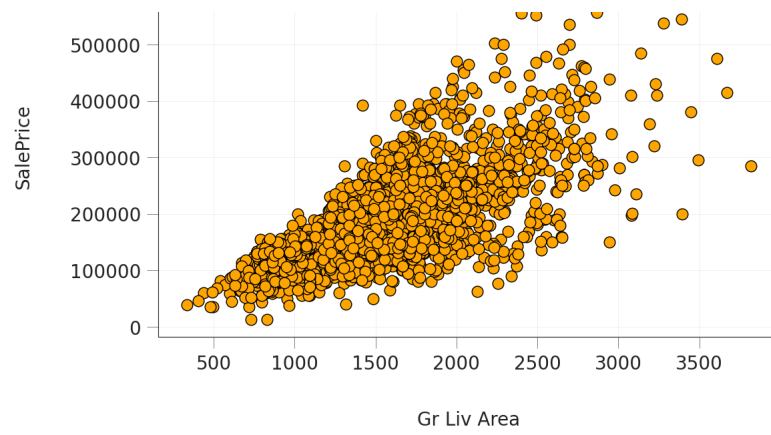
Number of children	Count
0	20
1	185
2	155
3	215
4	235
5	215
6	160
7	125
8	90
9	90



```
plt.scatter(x=dataset['Gr Liv Area'], y=dataset['SalePrice'],  
            color="orange", edgecolors="#000000", linewidths=0.5);  
plt.xlabel("Gr Liv Area"); plt.ylabel("SalePrice");
```

600000



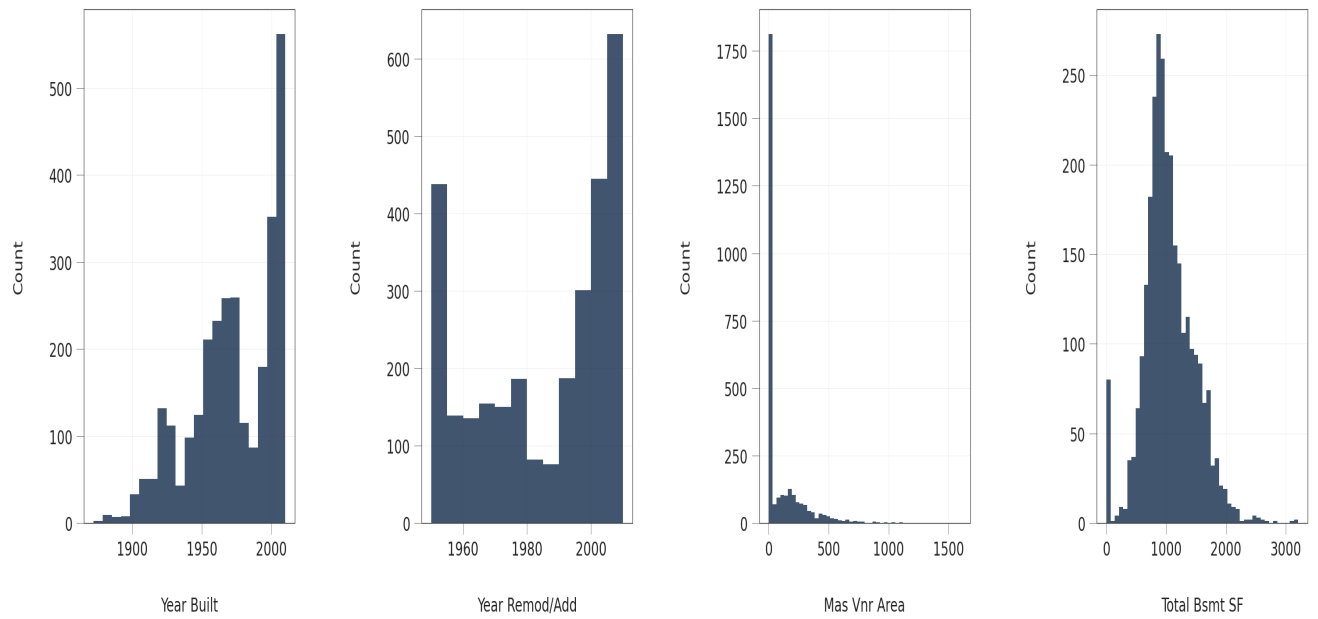


```
fig, axes = plt.subplots(1, 4, figsize=(18,5))
fig.subplots_adjust(hspace=0.5, wspace=0.6)
for ax, v in zip(axes.flat, ["Year Built", "Year Remod/Add",
                             "Mas Vnr Area", "Total Bsmt SF"]):
    sns.distplot(dataset[v], kde=False, color="#172B4D",
                  hist_kws={"alpha": 0.8}, ax=ax)
    ax.set(ylabel="Count");
```

[illegible]

[illegible]

WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.



```
x_vars = ["Year Built", "Year Remod/Add", "Mas Vnr Area", "Total Bsmt SF"]  
g = sns.PairGrid(dataset, y_vars=["SalePrice"], x_vars=x_vars);  
g.map(plt.scatter, color="orange", edgecolors="#000000", linewidths=0.5);
```


[illegible]

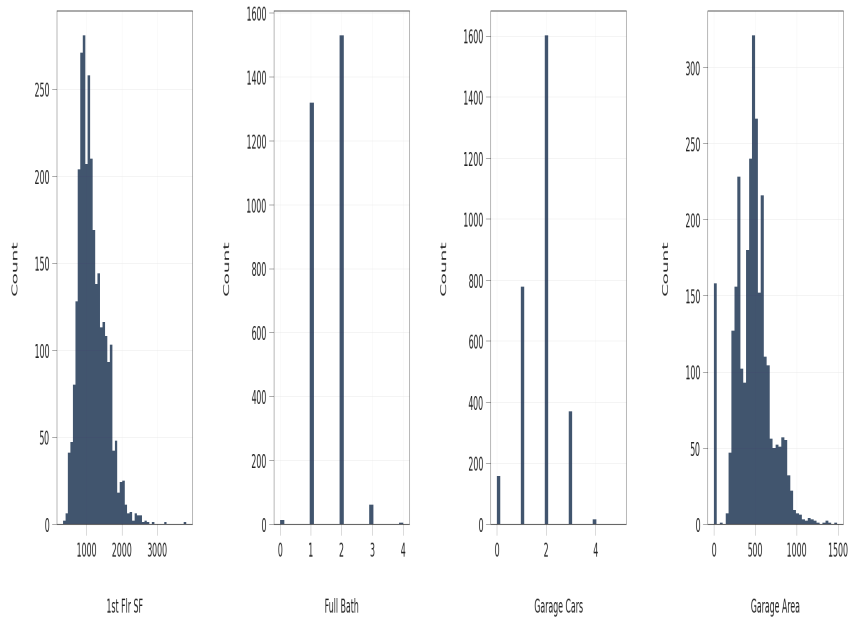
[illegible]

[illegible]

```

WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.

```



```

x_vars = ["1st Flr SF", "Full Bath", "Garage Cars", "Garage Area"]
g = sns.PairGrid(dataset, y_vars=["SalePrice"], x_vars=x_vars);
g.map(plt.scatter, color="orange", edgecolors="#000000", linewidths=0.5);

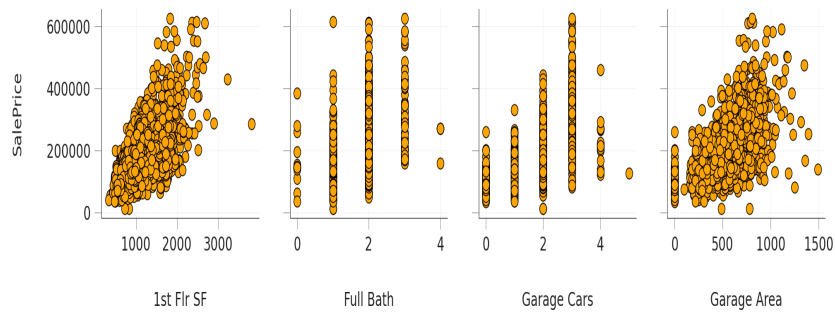
```

[illegible]

```

WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.

```



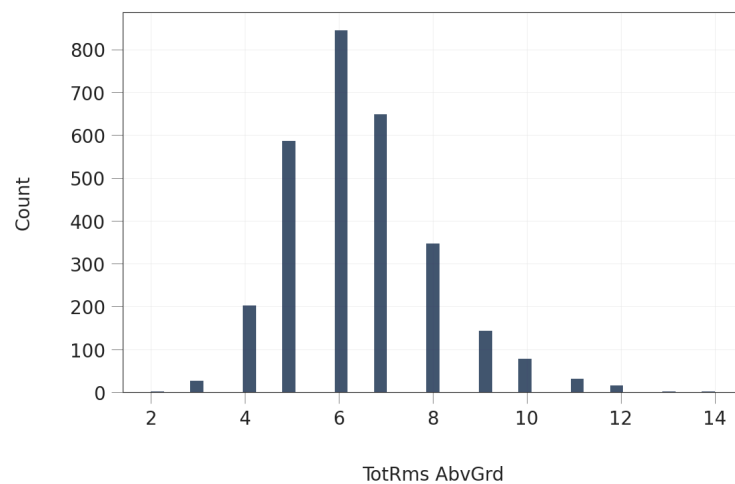
```

sns.distplot(dataset['TotRms AbvGrd'], kde=False,
              color="#172B4D", hist_kws={"alpha": 0.8});
plt.ylabel("Count");

```

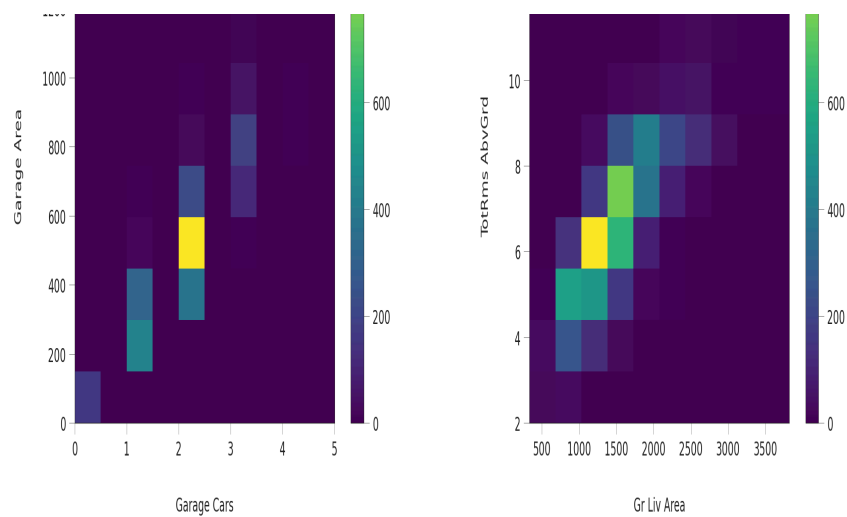
[illegible]

WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
 WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
 WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
 WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
 WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.
 WARNING:matplotlib.font_manager:findfont: Font family 'Arial' not found.



```
plt.rc("grid", linewidth=0.05)
fig, axes = plt.subplots(1, 2, figsize=(15,5))
fig.subplots_adjust(hspace=0.5, wspace=0.4)
h1 = axes[0].hist2d(dataset["Garage Cars"],
                    dataset["Garage Area"],
                    cmap="viridis");
axes[0].set(xlabel="Garage Cars", ylabel="Garage Area")
plt.colorbar(h1[3], ax=axes[0]);
h2 = axes[1].hist2d(dataset["Gr Liv Area"],
                    dataset["TotRms AbvGrd"],
                    cmap="viridis");
axes[1].set(xlabel="Gr Liv Area", ylabel="TotRms AbvGrd")
plt.colorbar(h2[3], ax=axes[1]);
plt.rc("grid", linewidth=0.25)
```

[illegible]



```
fig, axes = plt.subplots(1, 3, figsize=(16,5))
fig.subplots_adjust(hspace=0.5, wspace=0.6)
for ax, v in zip(axes.flat, ["Bsmt Unf SF", "BsmtFin SF 1", "Bsmt Full Bath"]):
    sns.distplot(dataset[v], kde=False, color="#172B4D",
                  hist_kws={"alpha": 0.8}, ax=ax);
    ax.set(ylabel="Count")
```

[illegible]