Section 01 - Vectors, matrices and arrays

1.1 Creating a vector

```
import numpy as np
vector_row = np.array([1,2,3])
```

```
vector_row
```

```
array([1, 2, 3])
```

```
vector_column = np.array([[1],
                          [2],
                          [3]])
```

```
vector_column
```

```
array([[1],
       [2],
       [3]])
```

1.2 Creating a Matrix

```
matrix = np.array([[1,2],
                   [3,4],
                   [5,6]])
```

```
matrix
```

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

1.3 Selecting Elements

```
vector_row[2]
```

```
3
```

```
matrix[1,1]
```

```
4
```

```
vector_row[:2]
```

```
array([1, 2])
```

```
vector_row[2:]
```

```
array([3])
```

```
vector_row[-1]
```

```
3
```

1.4 Applying operations to elements

```
add_100 = lambda i: i + 100
vectorized_add_100 = np.vectorize(add_100)
```

```
vectorized_add_100(matrix)
```

```
array([[101, 102],
       [103, 104],
       [105, 106]])
```

```
matrix + 100
```

```
array([[101, 102],
       [103, 104],
       [105, 106]])
```

1.5 Finding min, max values & average, variance, standard deviation

```
np.max(matrix)
```

```
6
```

```
np.min(matrix)
```

```
1
```

```
np.max(matrix, axis=0)
```

```
array([5, 6])
```

```
np.max(matrix, axis=1)
```

```
array([2, 4, 6])
```

```
np.mean(matrix)
```

```
3.5
```

```
np.var(matrix)
```

```
2.9166666666666665
```

```
np.std(matrix)
```

```
1.707825127659933
```

1.6 Reshaping Arrays

```
matrix.size
```

```
6
```

```
matrix.reshape(2,3)
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

1.7 Transposing matrix

```
matrix.T
```

```
array([[1, 3, 5],
       [2, 4, 6]])
```

1.8 Finding eigenvalues and eignevectors

```
matrix = np.array([[1, -1, 3],
                   [1, 1, 6],
                   [3, 8, 9]])
```

```
eigenvalues, eigenvectors = np.linalg.eig(matrix)
```

```
eigenvalues
```

```
     array([13.55075847,  0.74003145, -3.29078992])
```

```
eigenvectors
```

```
     array([[-0.17622017, -0.96677403, -0.53373322],
            [-0.435951  ,  0.2053623 , -0.64324848],
            [-0.88254925,  0.15223105,  0.54896288]])
```

1.9 calculating Dot Products

```
vector_a = np.array([1,2,3])
vector_b = np.array([4,5,6])
```

```
np.dot(vector_a, vector_b)
```

```
     32
```

```
vector_a @ vector_b
```

```
     32
```

1.10 Multiplying Matrices

```
matrix_a = np.array([[1,1],
                     [1,2]])

matrix_b = np.array([[1,3],
                     [1,2]])
```

```
np.dot(matrix_a, matrix_b)
```

```
array([[2, 5],
       [3, 7]])
```

```
matrix_a @ matrix_b
```

```
array([[2, 5],
       [3, 7]])
```

1.11 Genarating Random values

```
np.random.seed(0)
```

Section 02 - Loading Data

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_rem
```

```
import pandas as pd
```

```
df = pd.read_csv('/content/gdrive/My Drive/train.csv')
```

```
df_x = pd.read_excel('/content/gdrive/My Drive/train')
```

Section 03 - Data wrangling

3.1 Describing the data

```
df.shape
```

```
    (1460, 81)
```

```
df.head(10)
```

Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour Utilities LotConfig LandSlop

```
df.describe()
```

vGrd	Fireplaces	GarageYrBlt	GarageCars	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorc
0000	1460.000000	1379.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.00000
7808	0.613014	1978.506164	1.767123	472.980137	94.244521	46.660274	21.954110	3.409589	15.06095
5393	0.644666	24.689725	0.747315	213.804841	125.338794	66.256028	61.119149	29.317331	55.75741
0000	0.000000	1900.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
0000	0.000000	1961.000000	1.000000	334.500000	0.000000	0.000000	0.000000	0.000000	0.00000
0000	1.000000	1980.000000	2.000000	480.000000	0.000000	25.000000	0.000000	0.000000	0.00000
0000	1.000000	2002.000000	2.000000	576.000000	168.000000	68.000000	0.000000	0.000000	0.00000
0000	3.000000	2010.000000	4.000000	1418.000000	857.000000	547.000000	552.000000	508.000000	480.00000

```
df.info()
```

```
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   object
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object 
 43  1stFlrSF       1460 non-null   int64  
 44  2ndFlrSF       1460 non-null   int64  
 45  LowQualFinSF   1460 non-null   int64  
 46  GrLivArea      1460 non-null   int64  
 47  BsmtFullBath   1460 non-null   int64  
 48  BsmtHalfBath   1460 non-null   int64  
 49  FullBath       1460 non-null   int64  
 50  HalfBath       1460 non-null   int64  
 51  BedroomAbvGr   1460 non-null   int64  
 52  KitchenAbvGr   1460 non-null   int64  
 53  KitchenQual    1460 non-null   object 
 54  TotRmsAbvGrd   1460 non-null   int64  
 55  Functional     1460 non-null   object 
 56  Fireplaces     1460 non-null   int64  
 57  FireplaceQu    770 non-null    object 
 58  GarageType     1379 non-null   object 
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object 
 61  GarageCars     1460 non-null   int64  
 62  GarageArea     1460 non-null   int64  
 63  GarageQual     1379 non-null   object 
 64  GarageCond     1379 non-null   object 
 65  PavedDrive     1460 non-null   object 
 66  WoodDeckSF     1460 non-null   int64  
 67  OpenPorchSF    1460 non-null   int64  
 68  EnclosedPorch  1460 non-null   int64  
 69  3SsnPorch      1460 non-null   int64  
 70  ScreenPorch    1460 non-null   int64  
 71  PoolArea       1460 non-null   int64  
 72  PoolQC         7 non-null      object 
 73  Fence          281 non-null    object 
 74  MiscFeature    54 non-null     object 
 75  MiscVal        1460 non-null   int64  
 76  MoSold         1460 non-null   int64  
 77  YrSold         1460 non-null   int64  
 78  SaleType       1460 non-null   object 
 79  SaleCondition  1460 non-null   object 
 80  SalePrice      1460 non-null   int64  
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

3.2 Navigating dataframes

```
df.iloc[0]
```

```
Id                    1
MSSubClass           60
MSZoning             RL
LotFrontage          65
LotArea            8450
                  ...
MoSold                2
YrSold             2008
SaleType             WD
SaleCondition    Normal
SalePrice        208500
Name: 0, Length: 81, dtype: object
```

```
df.iloc[1:4]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlop
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	G
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	G
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	G

3 rows × 81 columns

```
df.iloc[:4]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlop
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	G
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	G
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	G

```
df_row1ToRow4 = df.iloc[:4]
```

4 rows × 81 columns

```
df_row1ToRow4
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlop
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	G
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	G
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	G
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	G

4 rows × 81 columns

```
df = df.set_index(df['LotConfig'])
```

```
df.loc['Inside']
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig
LotConfig											
Inside	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside
Inside	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside
Inside	6	50	RL	85.0	14115	Pave	NaN	IR1	Lvl	AllPub	Inside
Inside	7	20	RL	75.0	10084	Pave	NaN	Reg	Lvl	AllPub	Inside
Inside	9	50	RM	51.0	6120	Pave	NaN	Reg	Lvl	AllPub	Inside
...	...	...	...	...	...	...	...	...	...	...	...
Inside	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	Inside

3.3 Selecting rows based in conditionals

```
df[(df['HouseStyle'] == '2Story') & (df['SalePrice'] >= 180921)]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig
LotConfig											
Inside	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside
Inside	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside

3.4 Finding Unique values

Inside 12 60 RL 85.0 11924 Pave NaN IR1 Lvl AllPub Inside

```
df['Neighborhood'].unique()
```

```
array(['CollgCr', 'Veenker', 'Crawfor', 'NoRidge', 'Mitchel', 'Somerst',
       'NWAmes', 'OldTown', 'BrkSide', 'Sawyer', 'NridgHt', 'NAmes',
       'SawyerW', 'IDOTRR', 'MeadowV', 'Edwards', 'Timber', 'Gilbert',
       'StoneBr', 'ClearCr', 'NPkVill', 'Blmngtn', 'BrDale', 'SWISU',
       'Blueste'], dtype=object)
```

Inside 1448 60 RL 80.0 10000 Pave NaN Reg Lvl AllPub Inside

```
df['Neighborhood'].value_counts()
```

```
NAmes      225
CollgCr    150
OldTown    113
Edwards    100
Somerst     86
Gilbert     79
NridgHt     77
Sawyer      74
NWAmes      73
SawyerW     59
BrkSide     58
Crawfor     51
Mitchel     49
NoRidge     41
Timber      38
IDOTRR      37
ClearCr     28
SWISU       25
StoneBr     25
```

```
MeadowV     17
Blmngtn     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
```

3.5 Handling missing values

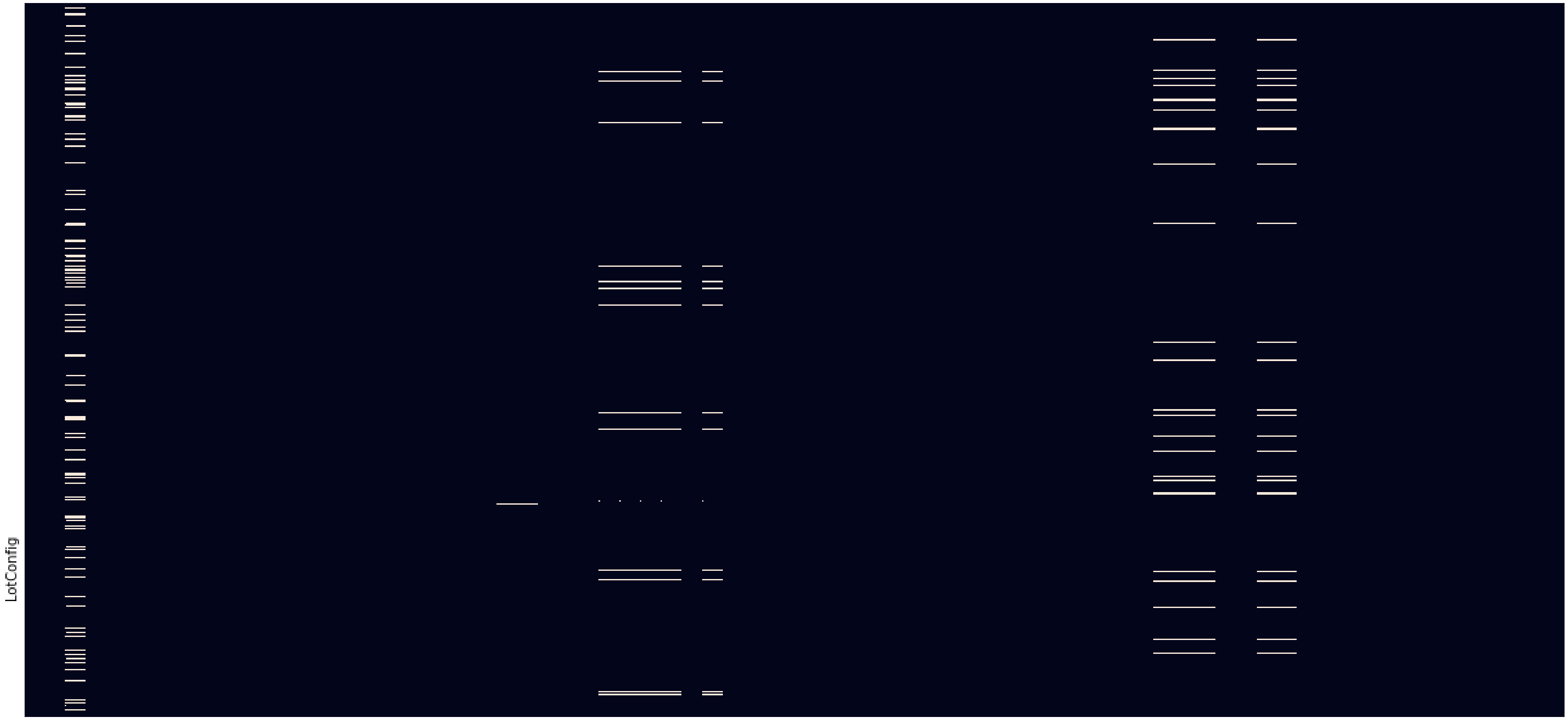
```
df.isnull().sum()
```

```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      259
LotArea            0
                ... 
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

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

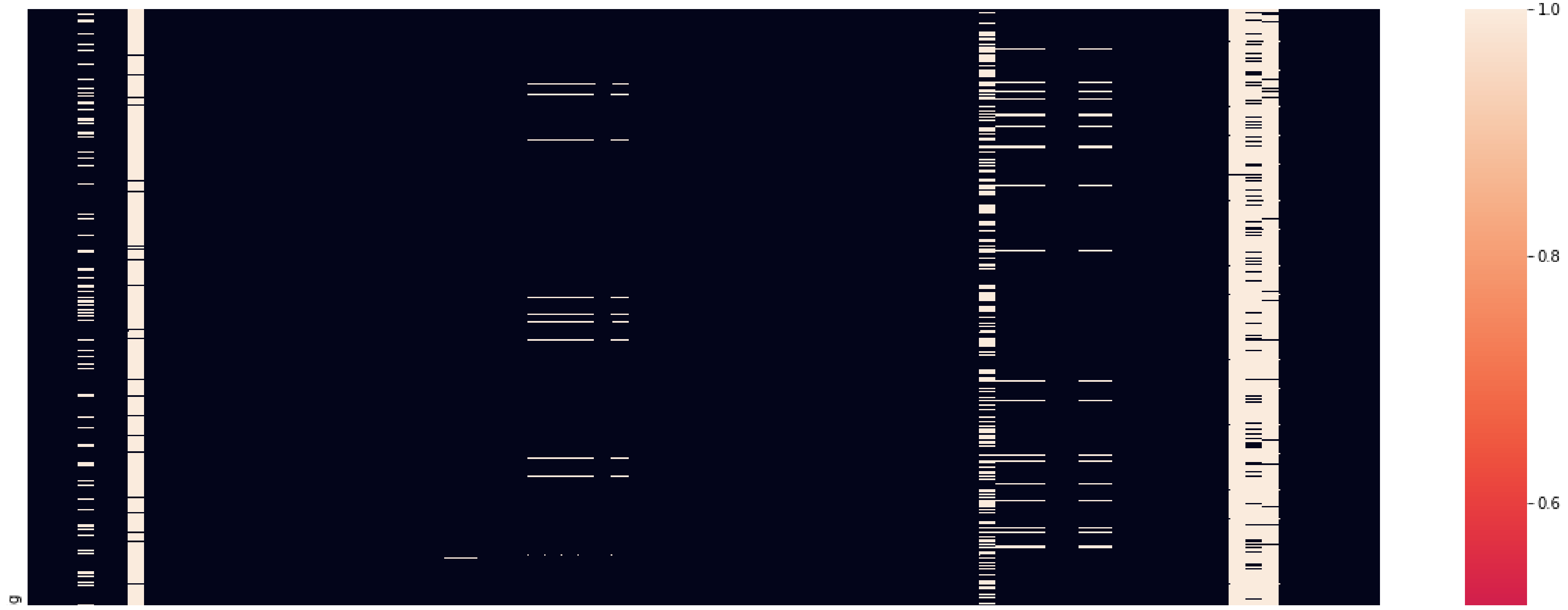
```
plt.subplots(figsize=(20,15))
sns.heatmap(df.isnull(), yticklabels=False, cbar=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9320c6c450>
```



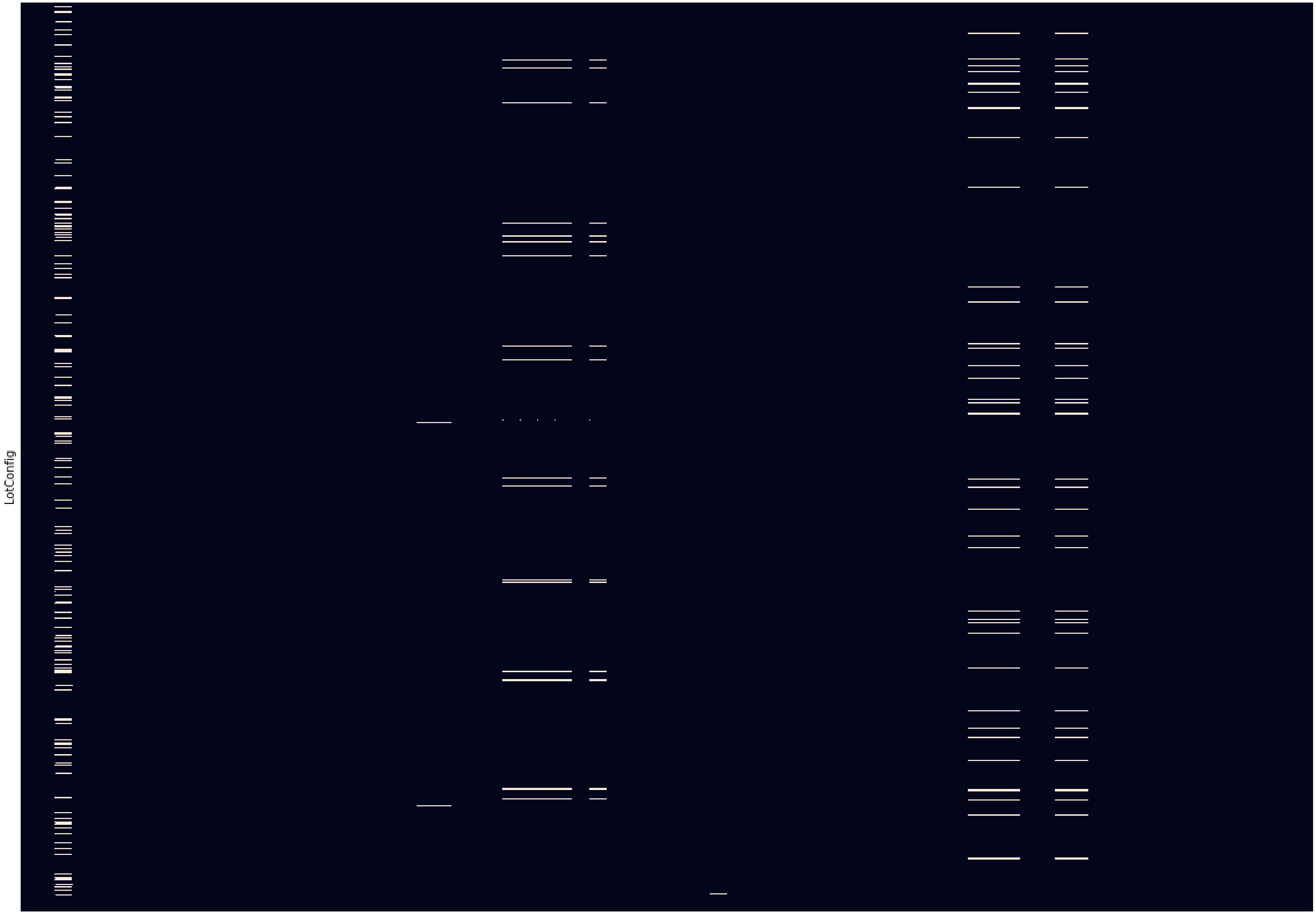
```
plt.subplots(figsize=(20,15))
sns.heatmap(df.isnull(), yticklabels=False, cbar=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f932c759610>
```



```
plt.subplots(figsize=(20,15))
sns.heatmap(df.isnull(), yticklabels=False, cbar=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9320d82cd0>
```



```
df.drop(['PoolQC','Fence','MiscFeature','Alley','FireplaceQu','Id'], axis=1,inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-85-a69692254665> in <module>()
----> 1 df.drop(['PoolQC','Fence','MiscFeature','Alley','FireplaceQu','Id'], axis=1,inplace=True)

                              3 frames
/usr/local/lib/python3.7/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   5285         if mask.any():
   5286             if errors != "ignore":
-> 5287                 raise KeyError(f"{labels[mask]} not found in axis")
   5288             indexer = indexer[~mask]
   5289         return self.delete(indexer)

KeyError: "['PoolQC' 'Fence' 'MiscFeature' 'Alley' 'FireplaceQu' 'Id'] not found in axis"
```

```
df.shape
```

```
(1460, 75)
```

```
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mean())
```

```
df['GarageCond'] = df['GarageCond'].fillna(df['GarageCond'].mode()[0])
```

```
df['GarageYrBlt'] = df['GarageYrBlt'].fillna(df['GarageYrBlt'].mean())
```

```
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].mean())
```

```
df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
```

```
df['GarageFinish'] = df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['Foundation'] = df['Foundation'].fillna(df['Foundation'].mode()[0])
df['BsmtQual'] = df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['BsmtCond'] = df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
```

```
df['BsmtFinType1'] = df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
df['BsmtFinType2'] = df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0])
df['MasVnrType'] = df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])


plt.subplots(figsize=(20,15))
sns.heatmap(df.isnull(),yticklabels= False, cbar= False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9320c5da90>
```

3.6 Dropping duplicate rows

```
df.duplicated()
```

```
LotConfig
Inside    False
FR2       False
Inside    False
Corner    False
FR2       False
            ...
Inside    False
Inside    False
Inside    False
Inside    False
Inside    False
Length: 1460, dtype: bool
```

```
df.drop_duplicates
df.drop_duplicates
```

```
<bound method DataFrame.drop_duplicates of            MSSubClass MSZoning  LotFrontage  ...  SaleType SaleCondition Sal
LotConfig                                    ...
Inside             60       RL         65.0  ...        WD        Normal    208500
FR2                20       RL         80.0  ...        WD        Normal    181500
Inside             60       RL         68.0  ...        WD        Normal    223500
Corner             70       RL         60.0  ...        WD       Abnorml    140000
FR2                60       RL         84.0  ...        WD        Normal    250000
...               ...      ...          ...  ...       ...           ...       ...
Inside             60       RL         62.0  ...        WD        Normal    175000
Inside             20       RL         85.0  ...        WD        Normal    210000
Inside             70       RL         66.0  ...        WD        Normal    266500
Inside             20       RL         68.0  ...        WD        Normal    142125
Inside             20       RL         75.0  ...        WD        Normal    147500

[1460 rows x 75 columns]>
```

3.7 Grouping rows by values

```
df.groupby('Neighborhood').mean()
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	Bs
Neighborhood									
Blmngtn	114.117647	51.185287	3398.176471	7.176471	5.000000	2005.235294	2005.764706	45.588235	1
Blueste	160.000000	24.000000	1625.000000	6.000000	6.000000	1980.000000	1980.000000	0.000000	1
BrDale	160.000000	21.562500	1801.000000	5.687500	5.437500	1971.437500	1973.625000	307.562500	2
BrkSide	49.741379	59.023271	7360.413793	5.051724	6.137931	1931.431034	1968.586207	7.396552	1
ClearCr	52.500000	76.276763	30875.750000	5.892857	5.678571	1966.571429	1983.750000	84.571429	6
CollgCr	43.300000	71.421327	9619.146667	6.640000	5.240000	1997.886667	1999.140000	97.917902	4
Crawfor	58.235294	71.460776	11809.686275	6.274510	6.588235	1941.549020	1979.196078	83.150691	4
Edwards	56.800000	68.363997	10218.650000	5.080000	5.440000	1955.970000	1975.110000	50.470000	4
Gilbert	58.227848	76.145554	11379.151899	6.556962	5.126582	1998.253165	1998.822785	42.831459	2
IDOTRR	53.648649	63.112159	8109.162162	4.756757	5.540541	1927.945946	1964.378378	16.216216	1
MeadowV	163.529412	32.770583	2324.000000	4.470588	5.529412	1972.588235	1976.705882	4.705882	3
Mitchel	56.632653	70.074479	11624.285714	5.591837	5.367347	1981.755102	1985.551020	61.306122	6
NAmes	38.777778	75.350882	10139.915556	5.360000	5.791111	1959.995556	1971.622222	101.142222	4
NPkVill	142.222222	40.677769	3267.444444	6.000000	5.555556	1976.444444	1976.444444	0.000000	4
NWAmes	44.589041	76.978066	11833.630137	6.328767	5.945205	1975.630137	1981.520548	177.561644	5
NoRidge	53.658537	87.619504	14218.902439	7.926829	5.219512	1995.439024	1996.658537	420.024390	8
NridgHt	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
SWISU	72.200000	59.803997	8127.560000	5.440000	5.920000	1925.240000	1969.680000	10.080000	2

3.8 Applying a function over all elements in a column

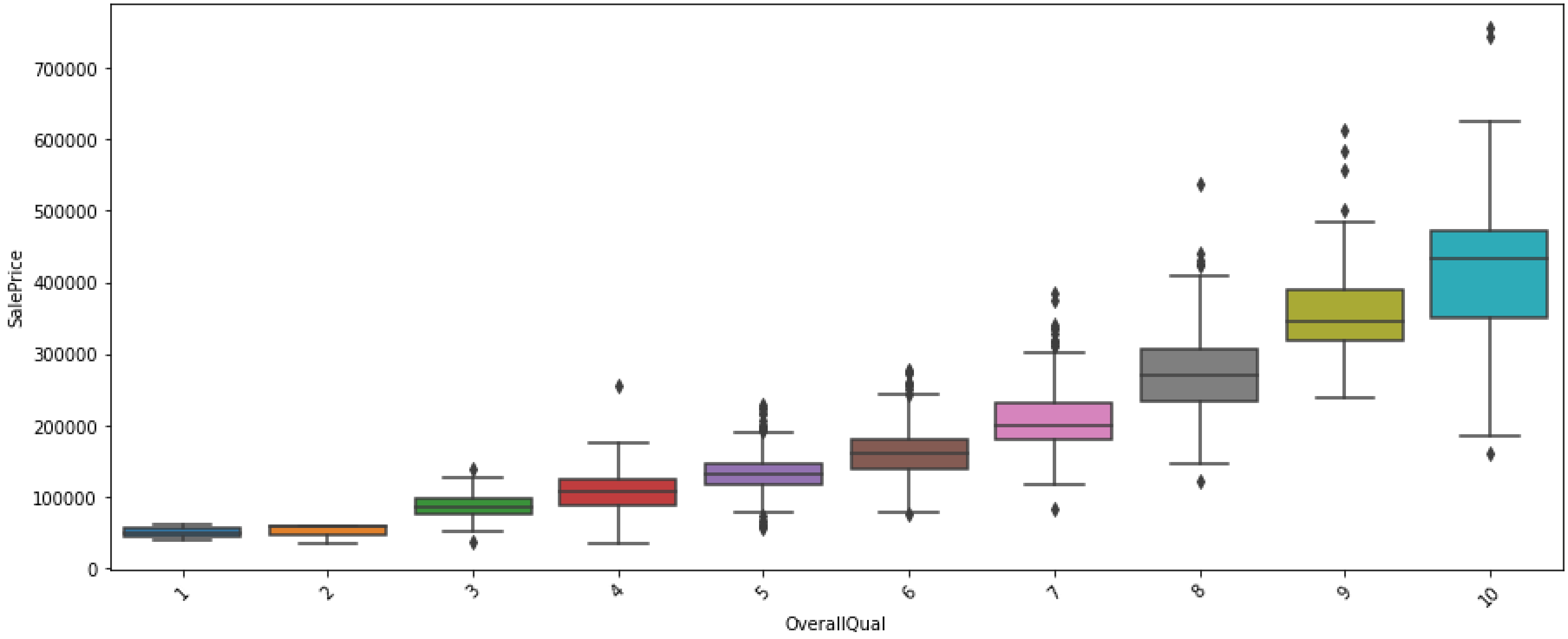
```
def uppercase(x):
  return x.upper()

  df['Neighborhood'].apply(uppercase)[0:2]
```

Section 04 - Outliers, handlingcatergorical data , scaling

4.1 Deceting Outliers

```
plt.figure(figsize=(15,6))
sns.boxplot(x='OverallQual', y= 'SalePrice',data=df)
xt = plt.xticks(rotation=45)
```



4.2 Handling outliers

```
import numpy as np
q25, q75 = np.percentile(df['SalePrice'], 25), np.percentile(df['SalePrice'], 75)


iqr = q75-q25


cut_off = iqr*1.5
lower, upper = q25 - cut_off , q75 + cut_off


df_out = df[(df['SalePrice'] > lower) & (df['SalePrice'] < upper)]
df_out
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	LandSlope
LotConfig										
Inside	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub	Inside	Gtl
FR2	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub	FR2	Gtl
Inside	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub	Inside	Gtl
Corner	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub	Corner	Gtl
FR2	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub	FR2	Gtl
...	...	...	...	...	...	...	...	...	...	...
Inside	60	RL	62.0	7917	Pave	Reg	Lvl	AllPub	Inside	Gtl
Inside	20	RL	85.0	13175	Pave	Reg	Lvl	AllPub	Inside	Gtl
Inside	70	RL	66.0	9042	Pave	Reg	Lvl	AllPub	Inside	Gtl
Inside	20	RL	68.0	9717	Pave	Reg	Lvl	AllPub	Inside	Gtl
Inside	20	RL	75.0	9937	Pave	Reg	Lvl	AllPub	Inside	Gtl

1399 rows × 75 columns

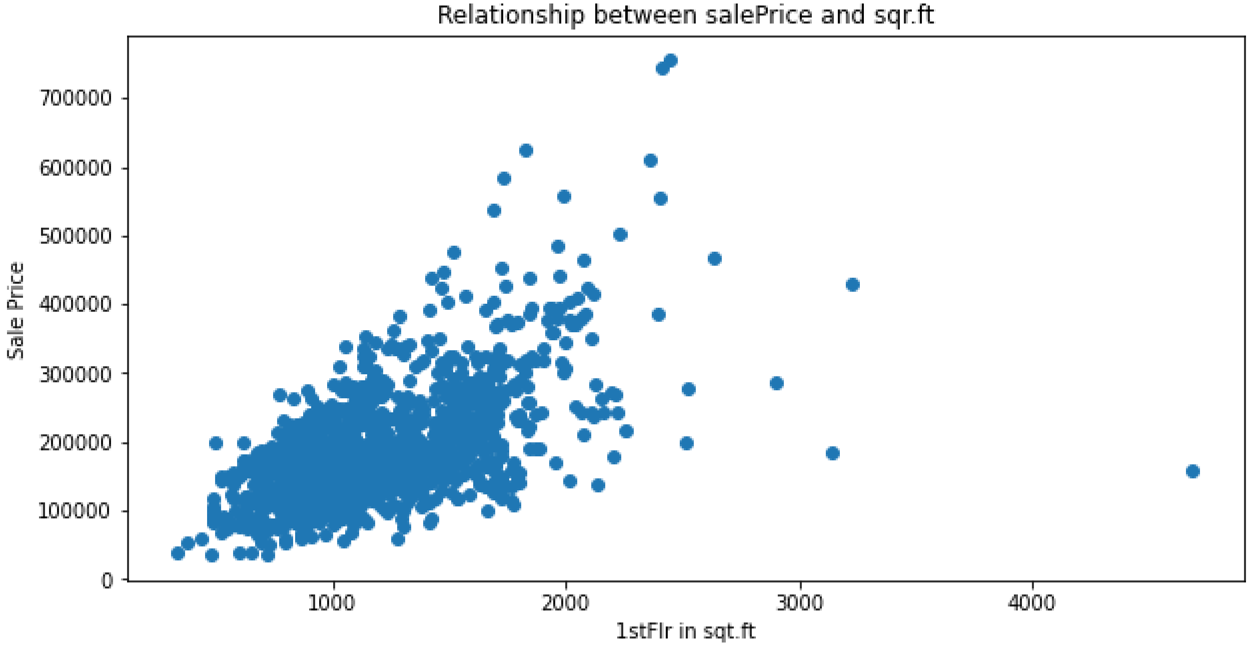
4.3 Correlation

```
corr = df_out.corr()
corr
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtF
MSSubClass	1.000000	-0.362001	-0.149714	0.066951	-0.070420	0.045599	0.055086	0.046524	-0.0
LotFrontage	-0.362001	1.000000	0.313771	0.183424	-0.037783	0.091363	0.058047	0.135931	0.1
LotArea	-0.149714	0.313771	1.000000	0.070548	0.004398	0.001485	0.012715	0.091415	0.2
OverallQual	0.066951	0.183424	0.070548	1.000000	-0.071040	0.561141	0.532226	0.326573	0.1
OverallCond	-0.070420	-0.037783	0.004398	-0.071040	1.000000	-0.361703	0.093423	-0.118063	-0.0
YearBuilt	0.045599	0.091363	0.001485	0.561141	-0.361703	1.000000	0.579099	0.283230	0.2
YearRemodAdd	0.055086	0.058047	0.012715	0.532226	0.093423	0.579099	1.000000	0.129447	0.0
MasVnrArea	0.046524	0.135931	0.091415	0.326573	-0.118063	0.283230	0.129447	1.000000	0.2
BsmtFinSF1	-0.050465	0.196989	0.203555	0.151949	-0.019725	0.211784	0.086505	0.215198	1.0
BsmtFinSF2	-0.066088	0.040311	0.057139	-0.050548	0.039380	-0.042351	-0.058630	-0.061459	-0.0
BsmtUnfSF	-0.137873	0.106554	0.014446	0.309623	-0.139107	0.141662	0.174562	0.085048	-0.5
TotalBsmtSF	-0.224965	0.335844	0.252314	0.466350	-0.151769	0.355419	0.251765	0.291655	0.4
1stFlrSF	-0.240781	0.397192	0.307689	0.388258	-0.132360	0.240259	0.194085	0.263065	0.3
2ndFlrSF	0.312578	0.046998	0.047520	0.289944	0.023791	0.001734	0.132715	0.126391	-0.1
LowQualFinSF	0.043961	0.036965	-0.001544	-0.050663	-0.000974	-0.170855	-0.065626	-0.068301	-0.0
GrLivArea	0.099421	0.340190	0.270307	0.537984	-0.077649	0.163352	0.254245	0.299727	0.1
BsmtFullBath	0.017438	0.074201	0.120442	0.067658	-0.038908	0.166577	0.102583	0.061976	0.6
BsmtHalfBath	-0.010330	-0.001480	0.067959	-0.036230	0.118188	-0.035965	-0.008017	0.030428	0.0
FullBath	0.148676	0.146441	0.120896	0.533141	-0.198730	0.461483	0.426281	0.217767	0.0

```
plt.figure(figsize = (10,5))
plt.scatter(x=df['1stFlrSF'], y = df['SalePrice'])
plt.ylabel('Sale Price')
plt.xlabel('1stFlr in sqt.ft')
plt.title('Relationship between salePrice and sqr.ft')
```

```
Text(0.5, 1.0, 'Relationship between salePrice and sqr.ft')
```



```
MoSold     -0.016176     0.014495   0.001886     0.073287     -0.001734    0.009607       0.018855     0.007052     -0.0
```

4.4 Handling categotical features

```
categorical_features = [column for column in df.columns if df[column].dtype == object]
```

```
print('number of cat fet ', len(categorical_features))
```

```
number of cat fet  38
```

```
categorical_features
```

```
['MSZoning',
 'Street',
 'LotShape',
 'LandContour',
 'Utilities',
 'LotConfig',
 'LandSlope',
 'Neighborhood',
 'Condition1',
 'Condition2',
 'BldgType',
 'HouseStyle',
 'RoofStyle',
 'RoofMatl',
 'Exterior1st',
 'Exterior2nd',
 'MasVnrType',
 'ExterQual',
 'ExterCond',
 'Foundation',
 'BsmtQual',
 'BsmtCond',
 'BsmtExposure',
 'BsmtFinType1',
 'BsmtFinType2',
 'Heating',
 'HeatingQC',
 'CentralAir',
 'Electrical',
 'KitchenQual',
 'Functional',
 'GarageType',
 'GarageFinish',
 'GarageQual',
 'GarageCond',
 'PavedDrive',
 'SaleType',
 'SaleCondition']
```

```
df_dummies = pd.get_dummies(df)
df_dummies
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
LotConfig									
Inside	60	65.0	8450	7	5	2003	2003	196.0	706
FR2	20	80.0	9600	6	8	1976	1976	0.0	978
Inside	60	68.0	11250	7	5	2001	2002	162.0	486
Corner	70	60.0	9550	7	5	1915	1970	0.0	216
FR2	60	84.0	14260	8	5	2000	2000	350.0	655
...	...	...	...	...	...	...	...	...	...
Inside	60	62.0	7917	6	5	1999	2000	0.0	0
Inside	20	85.0	13175	6	6	1978	1988	119.0	790
Inside	70	66.0	9042	7	9	1941	2006	0.0	275
Inside	20	68.0	9717	5	6	1950	1996	0.0	49

```
df['GarageType'].unique()
```

```
array(['Attchd', 'Detchd', 'BuiltIn', 'CarPort', 'Basment', '2Types'],
      dtype=object)
```

4.5 Rescaling Dataset

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0,1))
rescaled_df_dummies = scaler.fit_transform(df_dummies)
```

```
rescaled_df_dummies
```

```
array([[0.23529412, 0.15068493, 0.0334198 , ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.20205479, 0.03879502, ..., 0.        , 1.        ,
        0.        ],
       [0.23529412, 0.1609589 , 0.04650728, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.29411765, 0.15410959, 0.03618687, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.1609589 , 0.03934189, ..., 0.        , 1.        ,
        0.        ],
       [0.        , 0.18493151, 0.04037019, ..., 0.        , 1.        ,
        0.        ]])
```

4.6 Discretizating features

```
from sklearn.preprocessing import Binarizer
```

```
age = np.array([[6],
                [12],
                [20],
                [36],
                [66]])
```

```
binarizer = Binarizer(18)
```

```
binarizer.fit_transform(age)
```

```
array([[0],
       [0],
       [1],
       [1],
       [1]])
```

```
np.digitize(age,bins=[20,30,64])
```

```
array([[0],
       [0],
       [1],
       [2],
       [3]])
```