Exercise

For this exercise, you will be working with the House Price Dataset.

Please grab the train.csv file from Kaggle and explore this dataset. You need to perform explroatory data analysis and see if there is any correlation between the variables and analyze the distribution of the dataset. The question is open-ended and basically you're asked to perform EDA.

1- Write a summary of your findings in one page (e.g., summary statistics, plots) and submit the pdf file. Therefore, for part 3 of your assignment, you need to submit at least one jupyter notebook file and one pdf file.

2- Push your code and project to github and provide the link to your code here. Ensure that your github project is organized to at least couple of main folders, ensure that you have the README file as well:

- Src
- Data
- Docs
- Results

Read this link for further info:
https://gist.github.com/ericmjl/27e50331f24db3e8f957d1fe7bbbe510

Overview

There are **1460** instances of training data

Read Data

In [1]:
```
import pandas as pd
df_ss=pd.read_csv('..\\Data\\sample_submission.csv')
df_test=pd.read_csv('..\\Data\\test.csv')
df_train=pd.read_csv('..\\Data\\train.csv')
```

In [2]:
```
df_ss.head()
```

Out[2]:

	Id	SalePrice
0	1461	169277.052498
1	1462	187758.393989
2	1463	183583.683570
3	1464	179317.477511
4	1465	150730.079977

In [3]:
```
df_test.head()
```

Out[3]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilit
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllP
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllP
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllP
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllP
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllP

5 rows × 80 columns

In [4]:
```
df_train.head()
```

Out[4]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub

5 rows × 81 columns

To get access to all columns including the dependent variable for prediction **SalePrice**, the dataframe chose to this EDA is *df_train*

General review

First, it is important to know the data, including its ***data type*** of all dataframe attributes and the number of ***missing values***</i> *present*

Data Type

Before checking the missing values, it is important identify if the data type are well defined regarding its content

In [5]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       1460 non-null   object 
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64  
 5   Street         1460 non-null   object 
 6   Alley          91 non-null     object 
 7   LotShape       1460 non-null   object 
 8   LandContour    1460 non-null   object 
 9   Utilities      1460 non-null   object 
 10  LotConfig      1460 non-null   object 
 11  LandSlope      1460 non-null   object 
 12  Neighborhood   1460 non-null   object 
 13  Condition1     1460 non-null   object 
 14  Condition2     1460 non-null   object 
 15  BldgType       1460 non-null   object 
 16  HouseStyle     1460 non-null   object 
 17  OverallQual    1460 non-null   int64  
 18  OverallCond    1460 non-null   int64  
 19  YearBuilt      1460 non-null   int64  
 20  YearRemodAdd   1460 non-null   int64  
 21  RoofStyle      1460 non-null   object 
 22  RoofMatl       1460 non-null   object 
 23  Exterior1st    1460 non-null   object 
 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
```

There are inconsistence in the columns types for example:

- *All columns that refers to Year are int type. Even though they are numbers, it does not make sence if sum 2013+2016 or 2013*2016, we won´t do calculation with this numbers. In order how to identify all over the columns if they are set in a aproppiated data type is check one by one.*

There are inconsistences in the data type of the dataframe columns:

There are inconsistences in the data type of the dataframe columns:

For example:

- *All* ***"Year"*** *columns are Integer type.*
- ***OverallCond*** *is integer type in the dataset since its content are numbers, but it is actually categories.*

Even though the column content are numbers, it does not mean they are.

For example:*There is no sense if we do these calculations:*

$2013 + 2016$ *or* $2013 * 2016$

We do not do these calculation with Years. So,in order to identify all over the columns with its aproppiated data type, the unique values needs to be identify, as it's shown in the following code:

In [6]:
```
d={}
for i in df_train.columns:
    d.update({i:(len(df_train[i].unique()), list(df_train[i].unique()) )})
```

In [7]:
```
pd.set_option('display.max_rows', None, 'display.max_columns', None)
df_unique=pd.DataFrame(d).transpose().sort_values(by=0,ascending=False)
#df_unique[df_unique.Len<5]
df_unique.columns=['Len','Categories']
df_unique
```

Out[7]:

	Len	Categories
Id	1460	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...
LotArea	1073	[8450, 9600, 11250, 9550, 14260, 14115, 10084,...
GrLivArea	861	[1710, 1262, 1786, 1717, 2198, 1362, 1694, 209...
BsmtUnfSF	780	[150, 284, 434, 540, 490, 64, 317, 216, 952, 1...
1stFlrSF	753	[856, 1262, 920, 961, 1145, 796, 1694, 1107, 1...
TotalBsmtSF	721	[856, 1262, 920, 756, 1145, 796, 1686, 1107, 9...
SalePrice	663	[208500, 181500, 223500, 140000, 250000, 14300...
BsmtFinSF1	637	[706, 978, 486, 216, 655, 732, 1369, 859, 0, 8...
GarageArea	441	[548, 460, 608, 642, 836, 480, 636, 484, 468, ...
2ndFlrSF	417	[854, 0, 866, 756, 1053, 566, 983, 752, 1142, ...
MasVnrArea	328	[196.0, 0.0, 162.0, 350.0, 186.0, 240.0, 286.0...
WoodDeckSF	274	[0, 298, 192, 40, 255, 235, 90, 147, 140, 160,...
OpenPorchSF	202	[61, 0, 42, 35, 84, 30, 57, 204, 4, 21, 33, 21...
BsmtFinSF2	144	[0, 32, 668, 486, 93, 491, 506, 712, 362, 41, ...
EnclosedPorch	120	[0, 272, 228, 205, 176, 87, 172, 102, 37, 144,...
YearBuilt	112	[2003, 1976, 2001, 1915, 2000, 1993, 2004, 197...
LotFrontage	111	[65.0, 80.0, 68.0, 60.0, 84.0, 85.0, 75.0, nan...
GarageYrBlt	98	[2003.0, 1976.0, 2001.0, 1998.0, 2000.0, 1993....
ScreenPorch	76	[0, 176, 198, 291, 252, 99, 184, 168, 130, 142...
YearRemodAdd	61	[2003, 1976, 2002, 1970, 2000, 1995, 2005, 197...
Neighborhood	25	[CollgCr, Veenker, Crawfor, NoRidge, Mitchel, ...
LowQualFinSF	24	[0, 360, 513, 234, 528, 572, 144, 392, 371, 39...
MiscVal	21	[0, 700, 350, 500, 400, 480, 450, 15500, 1200,...
3SsnPorch	20	[0, 320, 407, 130, 180, 168, 140, 508, 238, 24...
Exterior2nd	16	[VinylSd, MetalSd, Wd Shng, HdBoard, Plywood, ...
MSSubClass	15	[60, 20, 70, 50, 190, 45, 90, 120, 30, 85, 80,...
Exterior1st	15	[VinylSd, MetalSd, Wd Sdng, HdBoard, BrkFace, ...
MoSold	12	[2, 5, 9, 12, 10, 8, 11, 4, 1, 7, 3, 6]
TotRmsAbvGrd	12	[8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14]
OverallQual	10	[7, 6, 8, 5, 9, 4, 10, 3, 1, 2]
SaleType	9	[WD, New, COD, ConLD, ConLI, CWD, ConLw, Con, ...
OverallCond	9	[5, 8, 6, 7, 4, 2, 3, 9, 1]
Condition1	9	[Norm, Feedr, PosN, Artery, RRAe, RRNn, RRAn, ...

	Len	Categories
BedroomAbvGr	8	[3, 4, 1, 2, 0, 5, 6, 8]
PoolArea	8	[0, 512, 648, 576, 555, 480, 519, 738]
RoofMatl	8	[CompShg, WdShngl, Metal, WdShake, Membran, Ta...
Condition2	8	[Norm, Artery, RRNn, Feedr, PosN, PosA, RRAn, ...
HouseStyle	8	[2Story, 1Story, 1.5Fin, 1.5Unf, SFoyer, SLvl,...
BsmtFinType2	7	[Unf, BLQ, nan, ALQ, Rec, LwQ, GLQ]
GarageType	7	[Attchd, Detchd, BuiltIn, CarPort, nan, Basmen...
BsmtFinType1	7	[GLQ, ALQ, Unf, Rec, BLQ, nan, LwQ]
Functional	7	[Typ, Min1, Maj1, Min2, Mod, Maj2, Sev]
Electrical	6	[SBrkr, FuseF, FuseA, FuseP, Mix, nan]
RoofStyle	6	[Gable, Hip, Gambrel, Mansard, Flat, Shed]
Heating	6	[GasA, GasW, Grav, Wall, OthW, Floor]
GarageQual	6	[TA, Fa, Gd, nan, Ex, Po]
Foundation	6	[PConc, CBlock, BrkTil, Wood, Slab, Stone]
SaleCondition	6	[Normal, Abnorml, Partial, AdjLand, Alloca, Fa...
FireplaceQu	6	[nan, TA, Gd, Fa, Ex, Po]
GarageCond	6	[TA, Fa, nan, Gd, Po, Ex]
MiscFeature	5	[nan, Shed, Gar2, Othr, TenC]
Fence	5	[nan, MnPrv, GdWo, GdPrv, MnWw]
GarageCars	5	[2, 3, 1, 0, 4]
YrSold	5	[2008, 2007, 2006, 2009, 2010]
HeatingQC	5	[Ex, Gd, TA, Fa, Po]
LotConfig	5	[Inside, FR2, Corner, CulDSac, FR3]
BldgType	5	[1Fam, 2fmCon, Duplex, TwnhsE, Twnhs]
MasVnrType	5	[BrkFace, None, Stone, BrkCmn, nan]
ExterCond	5	[TA, Gd, Fa, Po, Ex]
MSZoning	5	[RL, RM, C (all), FV, RH]
BsmtQual	5	[Gd, TA, Ex, nan, Fa]
BsmtCond	5	[TA, Gd, nan, Fa, Po]
BsmtExposure	5	[No, Gd, Mn, Av, nan]
KitchenQual	4	[Gd, TA, Ex, Fa]
Fireplaces	4	[0, 1, 2, 3]
ExterQual	4	[Gd, TA, Ex, Fa]
PoolQC	4	[nan, Ex, Fa, Gd]

	Len	Categories
LotShape	4	[Reg, IR1, IR2, IR3]
LandContour	4	[Lvl, Bnk, Low, HLS]
BsmtFullBath	4	[1, 0, 2, 3]
GarageFinish	4	[RFn, Unf, Fin, nan]
FullBath	4	[2, 1, 3, 0]
KitchenAbvGr	4	[1, 2, 3, 0]
PavedDrive	3	[Y, N, P]
BsmtHalfBath	3	[0, 1, 2]
HalfBath	3	[1, 0, 2]
LandSlope	3	[Gtl, Mod, Sev]
Alley	3	[nan, Grvl, Pave]
Street	2	[Pave, Grvl]
CentralAir	2	[Y, N]
Utilities	2	[AllPub, NoSeWa]

Prior, I have checked the output of the previous table to identify the correct data type. The list that contains this results is saved into datatype variable:

In [8]:
```
datatype=['int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'int', 'int',
```

Let´s merge this list into the dataframe df_unique

In [9]:
```
df_unique['Correct_datatype']=datatype
```

In [10]:
```
df_unique
```

Out[10]:

	Len	Categories	Correct_datatype
Id	1460	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14...	int
LotArea	1073	[8450, 9600, 11250, 9550, 14260, 14115, 10084,...	int
GrLivArea	861	[1710, 1262, 1786, 1717, 2198, 1362, 1694, 209...	int
BsmtUnfSF	780	[150, 284, 434, 540, 490, 64, 317, 216, 952, 1...	int
1stFlrSF	753	[856, 1262, 920, 961, 1145, 796, 1694, 1107, 1...	int
TotalBsmtSF	721	[856, 1262, 920, 756, 1145, 796, 1686, 1107, 9...	int
SalePrice	663	[208500, 181500, 223500, 140000, 250000, 14300...	int
BsmtFinSF1	637	[706, 978, 486, 216, 655, 732, 1369, 859, 0, 8...	int
GarageArea	441	[548, 460, 608, 642, 836, 480, 636, 484, 468, ...	int
2ndFlrSF	417	[854, 0, 866, 756, 1053, 566, 983, 752, 1142, ...	int
MasVnrArea	328	[196.0, 0.0, 162.0, 350.0, 186.0, 240.0, 286.0...	float
WoodDeckSF	274	[0, 298, 192, 40, 255, 235, 90, 147, 140, 160,...	int
OpenPorchSF	202	[61, 0, 42, 35, 84, 30, 57, 204, 4, 21, 33, 21...	int
BsmtFinSF2	144	[0, 32, 668, 486, 93, 491, 506, 712, 362, 41, ...	int
EnclosedPorch	120	[0, 272, 228, 205, 176, 87, 172, 102, 37, 144,...	int
YearBuilt	112	[2003, 1976, 2001, 1915, 2000, 1993, 2004, 197...	cat
LotFrontage	111	[65.0, 80.0, 68.0, 60.0, 84.0, 85.0, 75.0, nan...	float
GarageYrBlt	98	[2003.0, 1976.0, 2001.0, 1998.0, 2000.0, 1993....	cat
ScreenPorch	76	[0, 176, 198, 291, 252, 99, 184, 168, 130, 142...	int
YearRemodAdd	61	[2003, 1976, 2002, 1970, 2000, 1995, 2005, 197...	cat
Neighborhood	25	[CollgCr, Veenker, Crawfor, NoRidge, Mitchel, ...	cat
LowQualFinSF	24	[0, 360, 513, 234, 528, 572, 144, 392, 371, 39...	int
MiscVal	21	[0, 700, 350, 500, 400, 480, 450, 15500, 1200,...	int
3SsnPorch	20	[0, 320, 407, 130, 180, 168, 140, 508, 238, 24...	int
Exterior2nd	16	[VinylSd, MetalSd, Wd Shng, HdBoard, Plywood, ...	cat
MSSubClass	15	[60, 20, 70, 50, 190, 45, 90, 120, 30, 85, 80,...	cat
Exterior1st	15	[VinylSd, MetalSd, Wd Sdng, HdBoard, BrkFace, ...	cat
MoSold	12	[2, 5, 9, 12, 10, 8, 11, 4, 1, 7, 3, 6]	cat
TotRmsAbvGrd	12	[8, 6, 7, 9, 5, 11, 4, 10, 12, 3, 2, 14]	cat
OverallQual	10	[7, 6, 8, 5, 9, 4, 10, 3, 1, 2]	cat
SaleType	9	[WD, New, COD, ConLD, ConLI, CWD, ConLw, Con, ...	cat
OverallCond	9	[5, 8, 6, 7, 4, 2, 3, 9, 1]	cat
Condition1	9	[Norm, Feedr, PosN, Artery, RRAe, RRNn, RRAn, ...	cat

	Len	Categories	Correct_datatype
BedroomAbvGr	8	[3, 4, 1, 2, 0, 5, 6, 8]	cat
PoolArea	8	[0, 512, 648, 576, 555, 480, 519, 738]	int
RoofMatl	8	[CompShg, WdShngl, Metal, WdShake, Membran, Ta...	cat
Condition2	8	[Norm, Artery, RRNn, Feedr, PosN, PosA, RRAn, ...	cat
HouseStyle	8	[2Story, 1Story, 1.5Fin, 1.5Unf, SFoyer, SLvl,...	cat
BsmtFinType2	7	[Unf, BLQ, nan, ALQ, Rec, LwQ, GLQ]	cat
GarageType	7	[Attchd, Detchd, BuiltIn, CarPort, nan, Basmen...	cat
BsmtFinType1	7	[GLQ, ALQ, Unf, Rec, BLQ, nan, LwQ]	cat
Functional	7	[Typ, Min1, Maj1, Min2, Mod, Maj2, Sev]	cat
Electrical	6	[SBrkr, FuseF, FuseA, FuseP, Mix, nan]	cat
RoofStyle	6	[Gable, Hip, Gambrel, Mansard, Flat, Shed]	cat
Heating	6	[GasA, GasW, Grav, Wall, OthW, Floor]	cat
GarageQual	6	[TA, Fa, Gd, nan, Ex, Po]	cat
Foundation	6	[PConc, CBlock, BrkTil, Wood, Slab, Stone]	cat
SaleCondition	6	[Normal, Abnorml, Partial, AdjLand, Alloca, Fa...	cat
FireplaceQu	6	[nan, TA, Gd, Fa, Ex, Po]	cat
GarageCond	6	[TA, Fa, nan, Gd, Po, Ex]	cat
MiscFeature	5	[nan, Shed, Gar2, Othr, TenC]	cat
Fence	5	[nan, MnPrv, GdWo, GdPrv, MnWw]	cat
GarageCars	5	[2, 3, 1, 0, 4]	cat
YrSold	5	[2008, 2007, 2006, 2009, 2010]	cat
HeatingQC	5	[Ex, Gd, TA, Fa, Po]	cat
LotConfig	5	[Inside, FR2, Corner, CulDSac, FR3]	cat
BldgType	5	[1Fam, 2fmCon, Duplex, TwnhsE, Twnhs]	cat
MasVnrType	5	[BrkFace, None, Stone, BrkCmn, nan]	cat
ExterCond	5	[TA, Gd, Fa, Po, Ex]	cat
MSZoning	5	[RL, RM, C (all), FV, RH]	cat
BsmtQual	5	[Gd, TA, Ex, nan, Fa]	cat
BsmtCond	5	[TA, Gd, nan, Fa, Po]	cat
BsmtExposure	5	[No, Gd, Mn, Av, nan]	cat
KitchenQual	4	[Gd, TA, Ex, Fa]	cat
Fireplaces	4	[0, 1, 2, 3]	cat
ExterQual	4	[Gd, TA, Ex, Fa]	cat
PoolQC	4	[nan, Ex, Fa, Gd]	cat

	Len	Categories	Correct_datatype
LotShape	4	[Reg, IR1, IR2, IR3]	cat
LandContour	4	[Lvl, Bnk, Low, HLS]	cat
BsmtFullBath	4	[1, 0, 2, 3]	cat
GarageFinish	4	[RFn, Unf, Fin, nan]	cat
FullBath	4	[2, 1, 3, 0]	cat
KitchenAbvGr	4	[1, 2, 3, 0]	cat
PavedDrive	3	[Y, N, P]	cat
BsmtHalfBath	3	[0, 1, 2]	cat
HalfBath	3	[1, 0, 2]	cat
LandSlope	3	[Gtl, Mod, Sev]	cat
Alley	3	[nan, Grvl, Pave]	cat
Street	2	[Pave, Grvl]	cat
CentralAir	2	[Y, N]	cat
Utilities	2	[AllPub, NoSeWa]	cat

Each value in the column Categories from dataframe df_unique corresponds to:

int	float	cat
Integer	Float	Category

Proceed to converted the columns into its correspond type showed in the datafarame df_unique.

In [11]:
```
for x in df_unique.reset_index().values:
    if x[3]=='int':
        df_train[x[0]]=df_train[x[0]].astype('int64')
    elif x[3]=='float':
        df_train[x[0]]=df_train[x[0]].astype('float64')
    elif x[3]=='cat':
        df_train[x[0]]=df_train[x[0]].astype('category')
```

In [12]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   category
 2   MSZoning       1460 non-null   category
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   category
 6   Alley          91 non-null     category
 7   LotShape       1460 non-null   category
 8   LandContour    1460 non-null   category
 9   Utilities      1460 non-null   category
 10  LotConfig      1460 non-null   category
 11  LandSlope      1460 non-null   category
 12  Neighborhood   1460 non-null   category
 13  Condition1     1460 non-null   category
 14  Condition2     1460 non-null   category
 15  BldgType       1460 non-null   category
 16  HouseStyle     1460 non-null   category
 17  OverallQual    1460 non-null   category
 18  OverallCond    1460 non-null   category
 19  YearBuilt      1460 non-null   category
 20  YearRemodAdd   1460 non-null   category
 21  RoofStyle      1460 non-null   category
 22  RoofMatl       1460 non-null   category
 23  Exterior1st    1460 non-null   category
 24  Exterior2nd    1460 non-null   category
 25  MasVnrType     1452 non-null   category
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   category
 28  ExterCond      1460 non-null   category
 29  Foundation     1460 non-null   category
 30  BsmtQual       1423 non-null   category
 31  BsmtCond       1423 non-null   category
 32  BsmtExposure   1422 non-null   category
 33  BsmtFinType1   1423 non-null   category
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   category
 36  BsmtFinSF2     1460 non-null   int64
 37  BsmtUnfSF      1460 non-null   int64
 38  TotalBsmtSF    1460 non-null   int64
 39  Heating        1460 non-null   category
 40  HeatingQC      1460 non-null   category
 41  CentralAir     1460 non-null   category
 42  Electrical     1459 non-null   category
 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   category
 48  BsmtHalfBath   1460 non-null   category
 49  FullBath       1460 non-null   category
 50  HalfBath       1460 non-null   category
 51  BedroomAbvGr   1460 non-null   category
 52  KitchenAbvGr   1460 non-null   category
 53  KitchenQual    1460 non-null   category
 54  TotRmsAbvGrd   1460 non-null   category
```

```
 55  Functional     1460 non-null   category
 56  Fireplaces     1460 non-null   category
 57  FireplaceQu    770 non-null    category
 58  GarageType     1379 non-null   category
 59  GarageYrBlt    1379 non-null   category
 60  GarageFinish   1379 non-null   category
 61  GarageCars     1460 non-null   category
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   category
 64  GarageCond     1379 non-null   category
 65  PavedDrive     1460 non-null   category
 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      category
 73  Fence          281 non-null    category
 74  MiscFeature    54 non-null     category
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   category
 77  YrSold         1460 non-null   category
 78  SaleType       1460 non-null   category
 79  SaleCondition  1460 non-null   category
 80  SalePrice      1460 non-null   int64
dtypes: category(60), float64(2), int64(19)
memory usage: 350.1 KB
```

*We got **60 Categorical** variables , **18 Numerical** Variables (excluding Id) and **2 float** Variables.*

In [13]:
```
categories=list(filter(lambda x: x if df_train[x].dtypes=='category' else False
floats=list(filter(lambda x: x if df_train[x].dtypes=='float64' else False,df_t
integers=list(filter(lambda x: x if df_train[x].dtypes=='int64' else False,df_t
integers.remove('Id')
```

Once the convertion is succeded, let´s use df.describe()

In [14]:
```
df_train.describe()
```

Out[14]:

	Id	LotFrontage	LotArea	MasVnrArea	BsmtFinSF1	BsmtFinSF2	Bsr
count	1460.000000	1201.000000	1460.000000	1452.000000	1460.000000	1460.000000	1460
mean	730.500000	70.049958	10516.828082	103.685262	443.639726	46.549315	567
std	421.610009	24.284752	9981.264932	181.066207	456.098091	161.319273	441
min	1.000000	21.000000	1300.000000	0.000000	0.000000	0.000000	0
25%	365.750000	59.000000	7553.500000	0.000000	0.000000	0.000000	223
50%	730.500000	69.000000	9478.500000	0.000000	383.500000	0.000000	477
75%	1095.250000	80.000000	11601.500000	166.000000	712.250000	0.000000	808
max	1460.000000	313.000000	215245.000000	1600.000000	5644.000000	1474.000000	2336

Missing Data

Actual data with null values are:

In [15]:
```
for x in df_train.columns:
    if (df_train[x].isna().sum()>0):
        print('{}: {} records are null out of {}. This means {:.2f}% of all rec
```

```
LotFrontage: 259 records are null out of 1460. This means 17.74% of all record
s are missing
Alley: 1369 records are null out of 1460. This means 93.77% of all records are
missing
MasVnrType: 8 records are null out of 1460. This means 0.55% of all records ar
e missing
MasVnrArea: 8 records are null out of 1460. This means 0.55% of all records ar
e missing
BsmtQual: 37 records are null out of 1460. This means 2.53% of all records are
missing
BsmtCond: 37 records are null out of 1460. This means 2.53% of all records are
missing
BsmtExposure: 38 records are null out of 1460. This means 2.60% of all records
are missing
BsmtFinType1: 37 records are null out of 1460. This means 2.53% of all records
are missing
BsmtFinType2: 38 records are null out of 1460. This means 2.60% of all records
are missing
Electrical: 1 records are null out of 1460. This means 0.07% of all records ar
e missing
FireplaceQu: 690 records are null out of 1460. This means 47.26% of all record
s are missing
GarageType: 81 records are null out of 1460. This means 5.55% of all records a
re missing
GarageYrBlt: 81 records are null out of 1460. This means 5.55% of all records
are missing
GarageFinish: 81 records are null out of 1460. This means 5.55% of all records
are missing
GarageQual: 81 records are null out of 1460. This means 5.55% of all records a
re missing
GarageCond: 81 records are null out of 1460. This means 5.55% of all records a
re missing
PoolQC: 1453 records are null out of 1460. This means 99.52% of all records ar
e missing
Fence: 1179 records are null out of 1460. This means 80.75% of all records are
missing
MiscFeature: 1406 records are null out of 1460. This means 96.30% of all recor
ds are missing
```

To see a general view of proportion of null values all over the data.

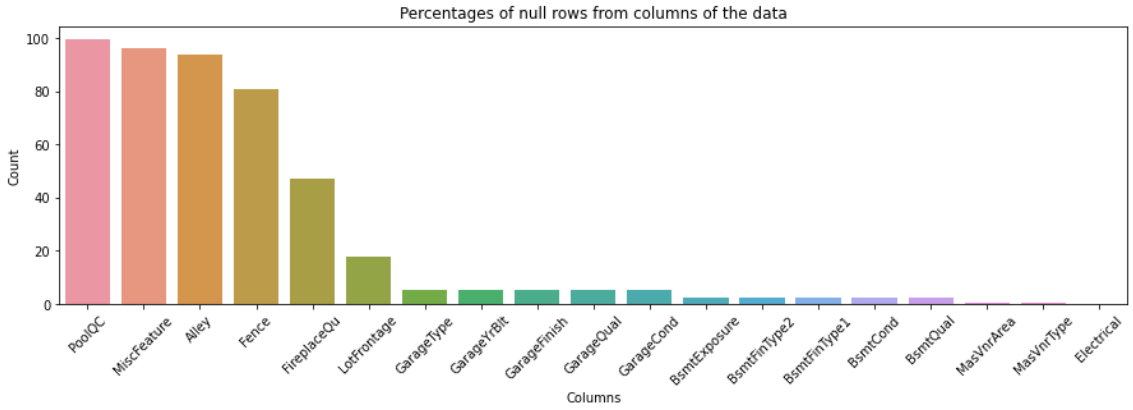
The next graph helps to analyze better this event

In [16]:
```
missing={}
for i in df_train:
    if df_train[i].isna().sum()>0:
        missing.update({i:df_train[i].isna().sum()})
```

In [17]:
```
df_missing=pd.DataFrame(missing,index=['Count']).transpose().sort_values(by='Co
df_missing.columns=['missing']
for x in df_missing.index:
    df_missing.loc[x,'missing_perc']=df_missing.loc[x,'missing']/len(df_train[x
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(15,4))
sns.barplot(x=df_missing.index,y=df_missing.missing_perc)
plt.title('Percentages of null rows from columns of the data')
plt.xlabel('Columns')
```

```
plt.ylabel('Count')
plt.xticks(rotation=45)
#plt.plot([0, 18], [10, 10], 'g', lw=2)
plt.show()
```



From this, it can be seen:

*About **100%** of houses do not have Pool, Miscellaneous feature and No alley access*

*More than **80%** of records do not have Fence*

*About **50%** of records do not have fireplaces*

*This means that the empty values not always mean missing values but tell **another story.***

However, there are others variables which we cannot say the same.
For example:

- ***LotFrontage: Null values in the linear feet of street connected to property. It might be data quality lack.***
- ***MSZoning: Null values in the general zoning classification of the sale. It might be data quality lack too.***
- ***So on...***

So It is important to procced a data cleaning process.

Data Cleaning

In this section, some attributes will be filled due to its values are important for the later analysis

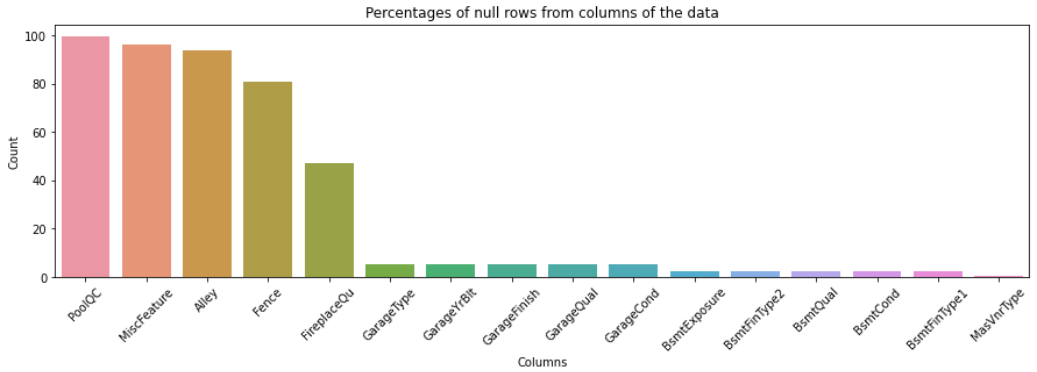
In [19]:
```
# numerical
df_train.LotFrontage=df_train.groupby("Neighborhood")["LotFrontage"].tran
df_train.MasVnrArea=df_train.MasVnrArea.fillna(0) # Those values in Nan f
#categorical
df_train['MSZoning'] = df_train.groupby("Neighborhood")["MSZoning"].trans
df_train['Electrical'] = df_train.groupby("Neighborhood")["Electrical"].t
```

Since the previous code was only sustituted the columns: LotFrontage,MSZoning, MasVnrArea and Electrical,the rest of the attributes with null values can be substituted by None because this value makes sense in the categories.

In [20]:
```
missing={}
for i in df_train:
    if df_train[i].isna().sum()>0:
        missing.update({i:df_train[i].isna().sum()})
```

In [21]:
```
df_missing=pd.DataFrame(missing,index=['Count']).transpose().sort_values
df_missing.columns=['missing']
for x in df_missing.index:
    df_missing.loc[x,'missing_perc']=df_missing.loc[x,'missing']/len(df_
```

In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(15,4))
sns.barplot(x=df_missing.index,y=df_missing.missing_perc)
plt.title('Percentages of null rows from columns of the data')
plt.xlabel('Columns')
plt.ylabel('Count')
plt.xticks(rotation=45)
#plt.plot([0, 18], [10, 10], 'g', lw=2)
plt.show()
```



Now, it left 16 attributes with NaN

```
df_train['PoolQC']=df_train['PoolQC'].cat.add_categories(['None'])
```

In [23]:
```
attri_null=pd.DataFrame(df_train.isnull().sum()[df_train.isnull().sum(
for i in attri_null.index:
    print(i)
    if 'None' not in df_train[i].cat.categories:
        df_train[i]=df_train[i].cat.add_categories(['None'])
    df_train[i] = df_train[i].fillna('None')
```

```
PoolQC
MiscFeature
Alley
Fence
FireplaceQu
GarageType
GarageYrBlt
GarageFinish
GarageQual
GarageCond
BsmtExposure
BsmtFinType2
BsmtQual
BsmtCond
BsmtFinType1
MasVnrType
```

In [24]:
```
print('There are {} columns with NULL values'.format(len(df_train.isnu
```

```
There are 0 columns with NULL values
```

Great!. There is not more NULL values in the dataset.

Univariable analysis

Numerical variables

In [25]:
```
def plot_subplot(itera,n,label,compare=''):
    row=int(len(itera)/n)+len(itera)%n
    colu=len(itera)-row*n

    if label=='hist':
        plt.figure(figsize=(16,6*row))
        for i,j in enumerate(itera):
            i+=1
            plt.subplot(row,n,i)
            sns.violinplot(data=df_train,y=j,kde=True,hue='Neighborho
            plt.title('Distribution of '+j)
    elif label=='cross':
            a=0
            b=0
            fig, axes=plt.subplots(row,n,figsize=(16,8*row))
```

```
            #print(row,n)
            for i,j in enumerate(itera):
                i+=1
                my_crosstab = df_train[j].value_counts(sort=False)/su
                if row==1 or n==1:
                    #print('a',a,b)
                    ax=my_crosstab.plot(kind='bar', stacked=True, rot
                    #ax.set_xlabel(j)
                    ax.set_ylabel('% Count')
                else:
                    #print(a,b)
                    ax=my_crosstab.plot(kind='bar', stacked=True, rot
                    ax.set_xlabel(j)
                    ax.set_ylabel('% Count')
                if b==n-1:
                    b=0
                    a+=1
                else:
                    b+=1
```

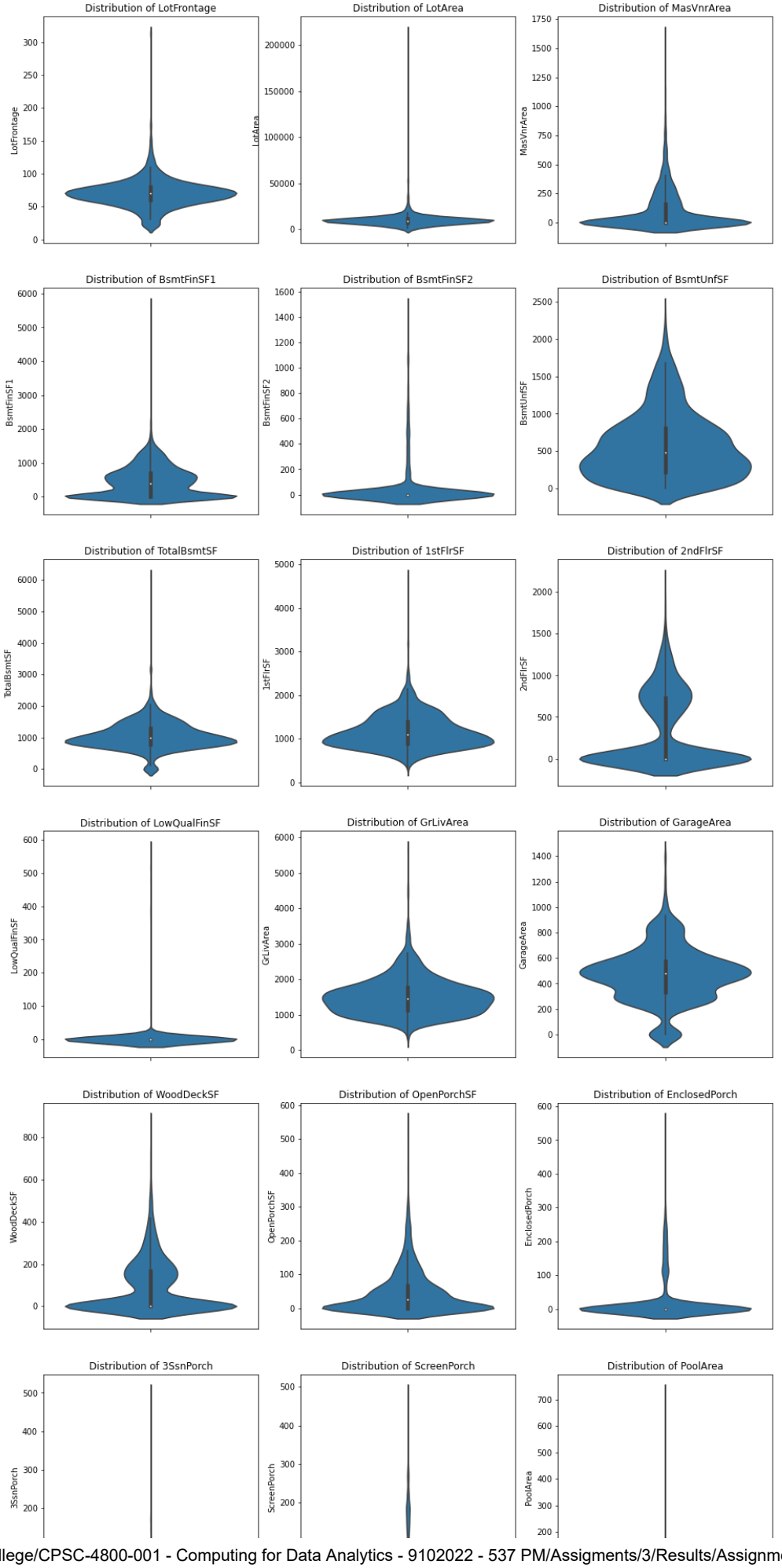
In [26]:
```
df_train.describe().round(1)
```

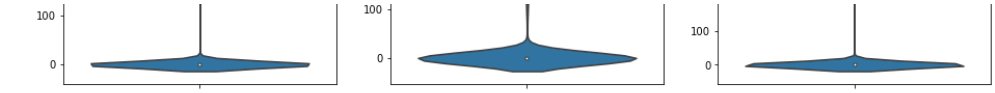
Out[26]:

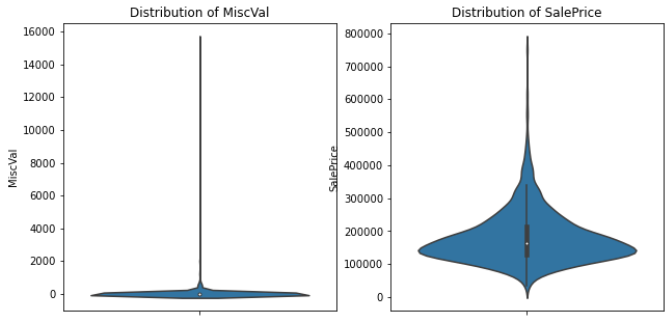
	Id	LotFrontage	LotArea	MasVnrArea	BsmtFinSF1	BsmtFinSF2	Bsr
count	1460.0	1460.0	1460.0	1460.0	1460.0	1460.0	
mean	730.5	70.2	10516.8	103.1	443.6	46.5	
std	421.6	22.4	9981.3	180.7	456.1	161.3	
min	1.0	21.0	1300.0	0.0	0.0	0.0	
25%	365.8	60.0	7553.5	0.0	0.0	0.0	
50%	730.5	70.0	9478.5	0.0	383.5	0.0	
75%	1095.2	80.0	11601.5	164.2	712.2	0.0	
max	1460.0	313.0	215245.0	1600.0	5644.0	1474.0	

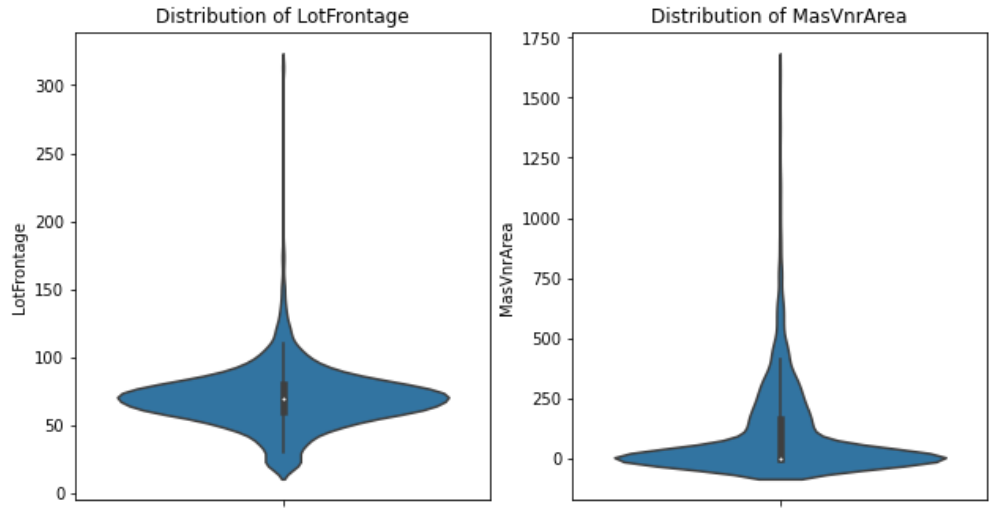
In [27]:
```
integers=list(df_train.describe().round(1).columns.drop('Id'))
```

In [28]:
```
plot_subplot(integers,3,'hist')
plot_subplot(floats,3,'hist')
```


Distribution of LotFrontage
LotFrontage
Distribution of LotArea
LotArea
Distribution of MasVnrArea
MasVnrArea
Distribution of BsmtFinSF1
BsmtFinSF1
Distribution of BsmtFinSF2
BsmtFinSF2
Distribution of BsmtUnfSF
BsmtUnfSF
Distribution of TotalBsmtSF
TotalBsmtSF
Distribution of 1stFlrSF
1stFlrSF
Distribution of 2ndFlrSF
2ndFlrSF
Distribution of LowQualFinSF
LowQualFinSF
Distribution of GrLivArea
GrLivArea
Distribution of GarageArea
GarageArea
Distribution of WoodDeckSF
WoodDeckSF
Distribution of OpenPorchSF
OpenPorchSF
Distribution of EnclosedPorch
EnclosedPorch
Distribution of 3SsnPorch
3SsnPorch
Distribution of ScreenPorch
ScreenPorch
Distribution of PoolArea
PoolArea

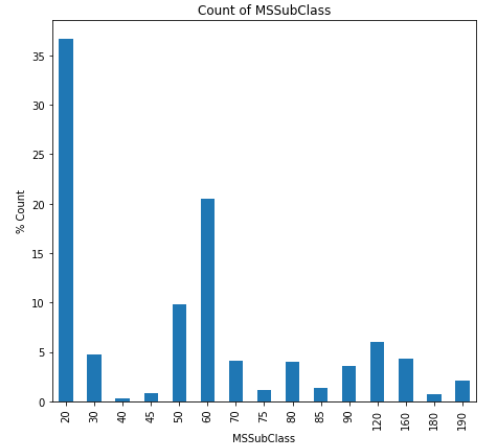


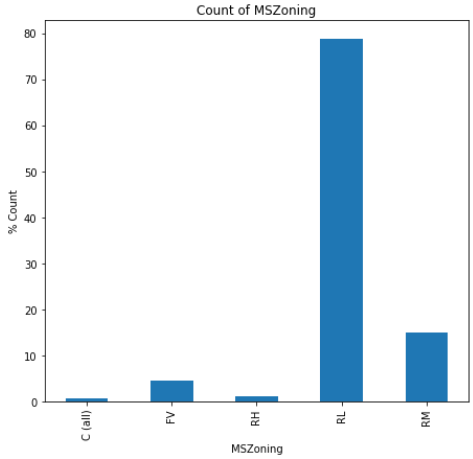


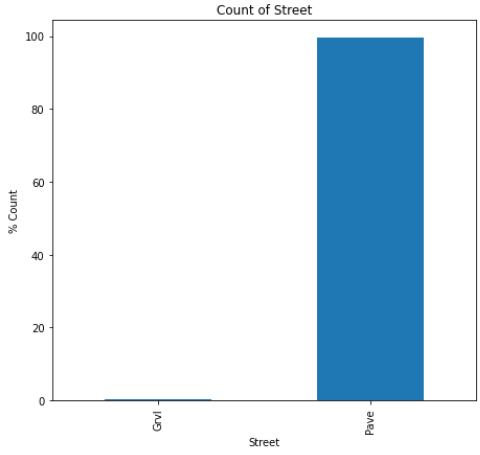


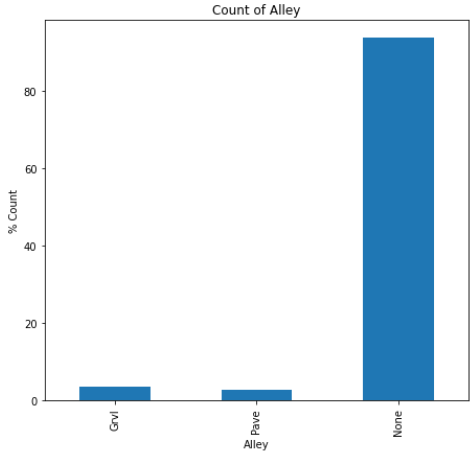
Categorical variables

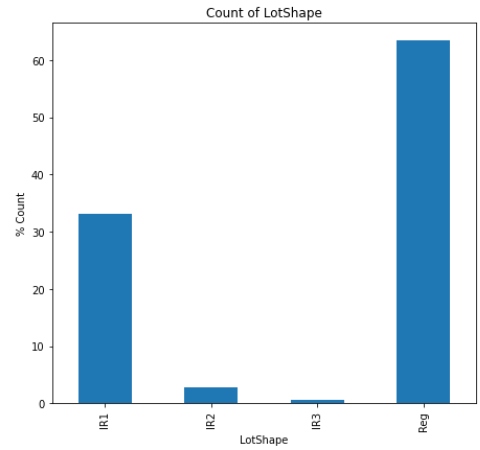
In [29]:
```
plot_subplot(categories,2,'cross','GarageQual')
```

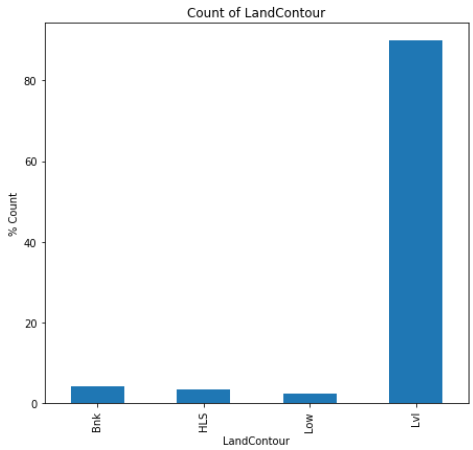

Count of MSSubClass
% Count
MSSubClass
20
30
40
45
50
60
70
75
80
85
90
120
160
180
190

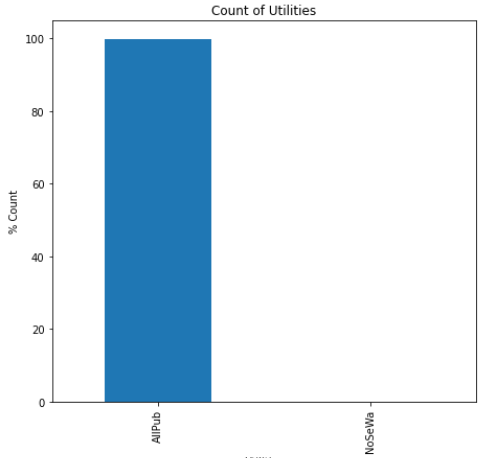

Count of MSZoning
% Count
MSZoning
C (all)
FV
RH
RL
RM

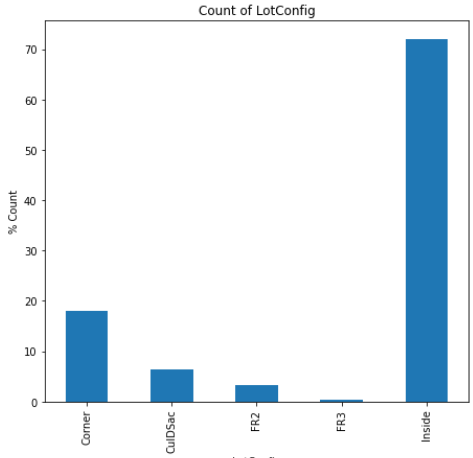

Count of Street
% Count
Street
Grvl
Pave

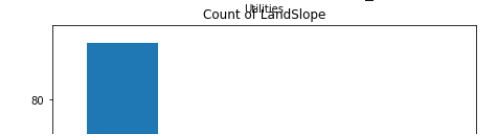

Count of Alley
% Count
Alley
Grvl
Pave
None


Count of LotShape
% Count
LotShape
IR1
IR2
IR3
Reg

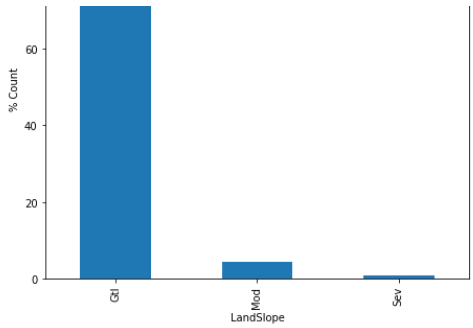

Count of LandContour
% Count
LandContour
Bnk
HLS
Low
Lvl

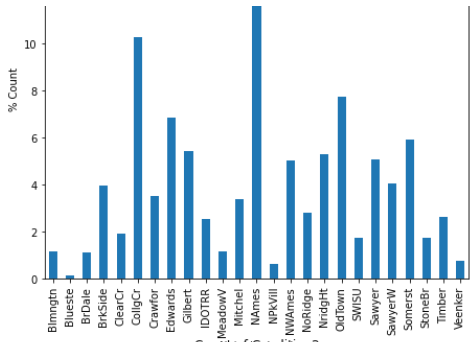

Count of Utilities
% Count
Utilities
AllPub
NoSeWa

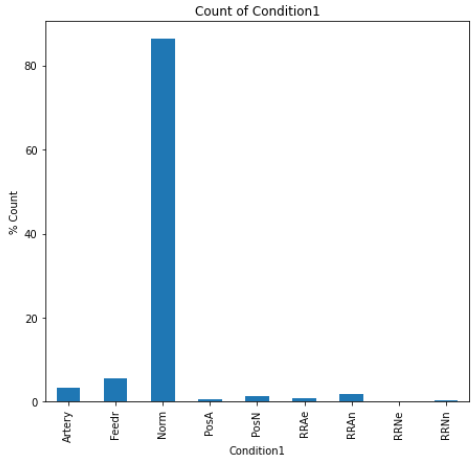

Count of LotConfig
% Count
LotConfig
Corner
CulDSac
FR2
FR3
Inside

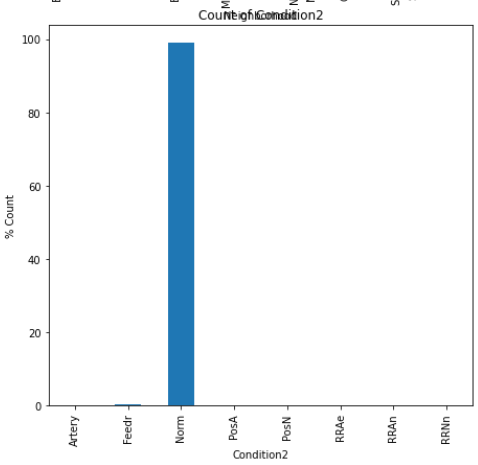

Count of LandSlope

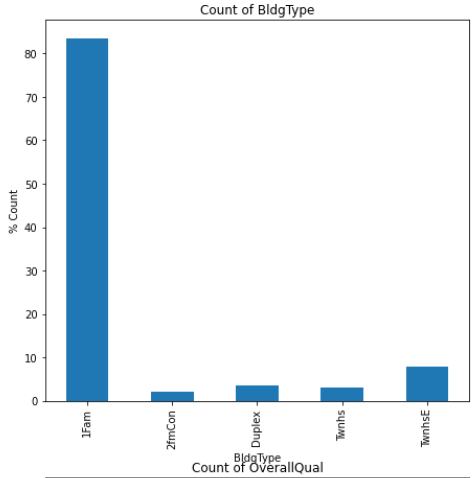

Count of Neighborhood

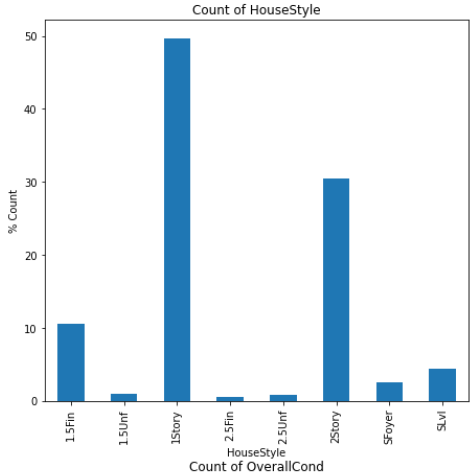

% Count
LandSlope
Gtl
Mod
Sev

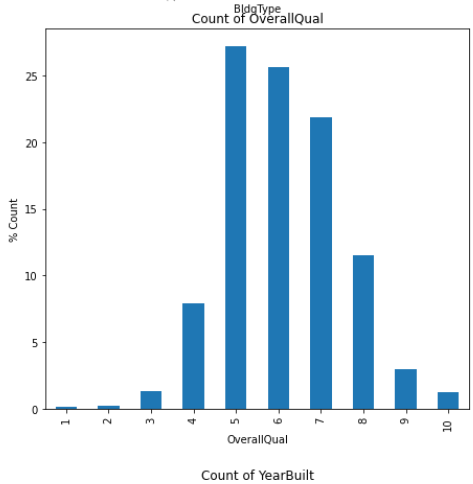

% Count
Blmngtn
Blueste
BrDale
BrkSide
ClearCr
CollgCr
Crawfor
Edwards
Gilbert
IDOTRR
MeadowV
Mitchel
NAmes
NPkVill
NWAmes
NoRidge
NridgHt
OldTown
SWISU
Sawyer
SawyerW
Somerst
StoneBr
Timber
Veenker

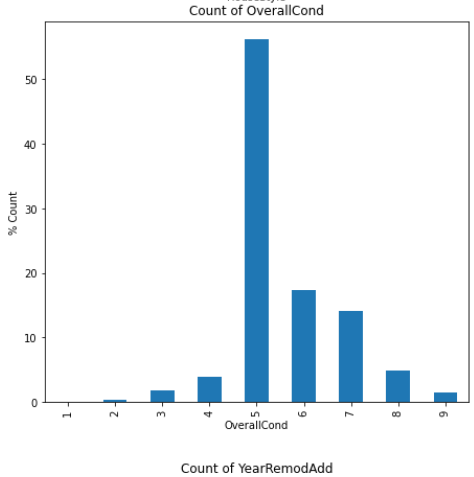

Count of Condition1
% Count
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNe
RRNn
Condition1

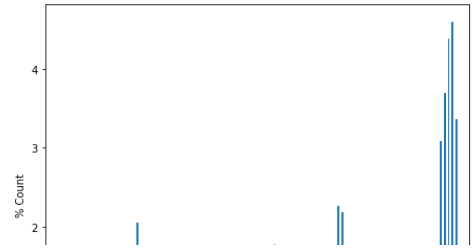

Count of Condition2
% Count
Artery
Feedr
Norm
PosA
PosN
RRAe
RRAn
RRNn
Condition2


Count of BldgType
% Count
1Fam
2fmCon
Duplex
Twnhs
TwnhsE
BldgType

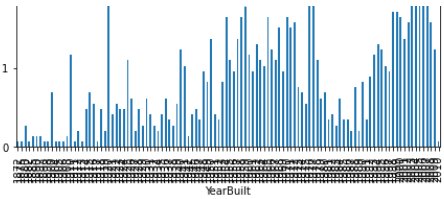

Count of HouseStyle
% Count
1.5Fin
1.5Unf
1Story
2.5Fin
2.5Unf
2Story
SFoyer
SLvl
HouseStyle

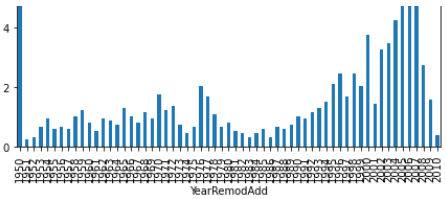

Count of OverallQual
% Count
OverallQual

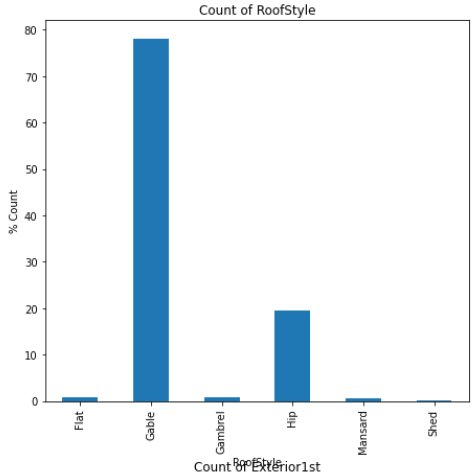

Count of OverallCond
% Count
OverallCond

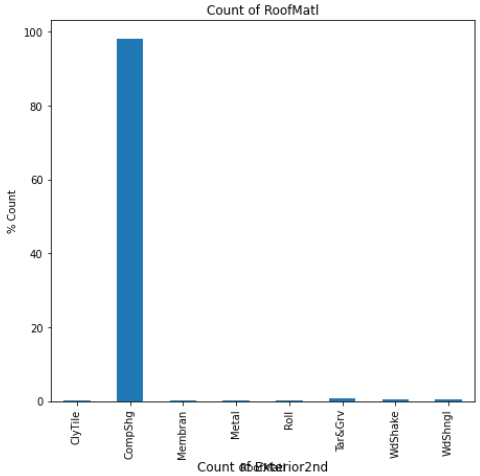

Count of YearBuilt
% Count

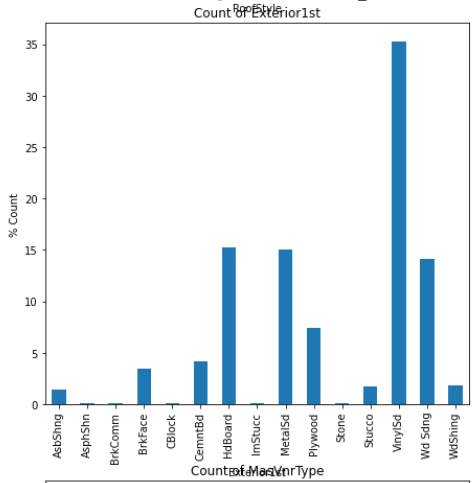

Count of YearRemodAdd
% Count

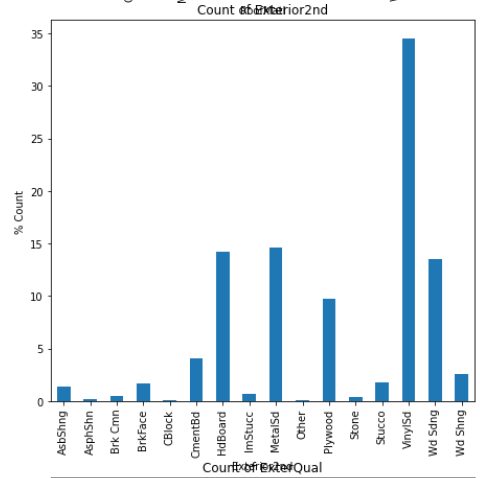

YearBuilt

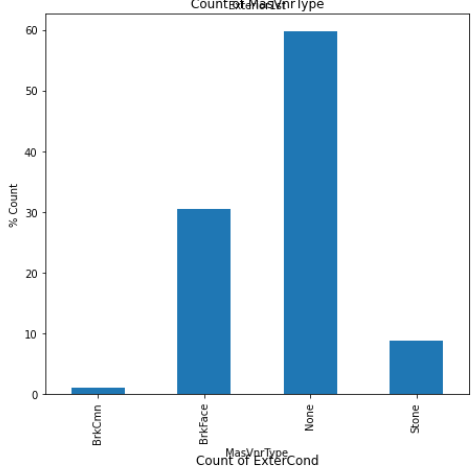

YearRemodAdd

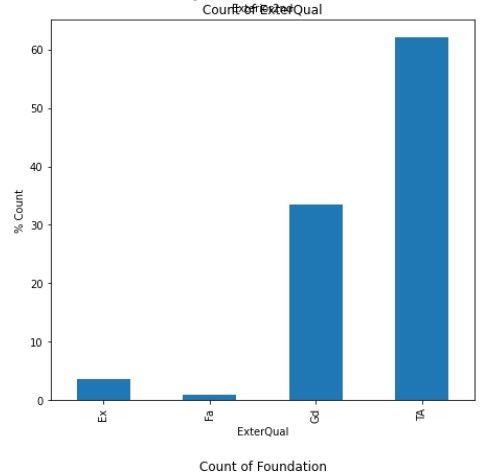

Count of RoofStyle
% Count
Flat
Gable
Gambrel
Hip
Mansard
Shed
RoofStyle

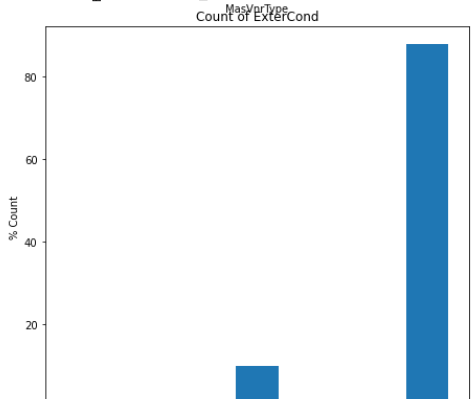

Count of RoofMatl
% Count
ClyTile
CompShg
Membran
Metal
Roll
Tar&Grv
WdShake
WdShngl

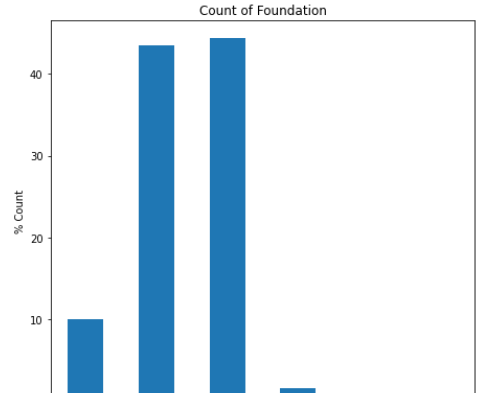

Count of Exterior1st
% Count
AsbShng
AsphShn
BrkComm
BrkFace
CBlock
CemntBd
HdBoard
ImStucc
MetalSd
Plywood
Stone
Stucco
VinylSd
Wd Sdng
WdShing

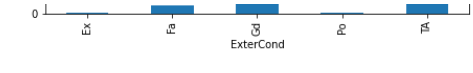

Count of Exterior2nd
% Count
AsbShng
AsphShn
Brk Cmn
BrkFace
CBlock
CmentBd
HdBoard
ImStucc
MetalSd
Other
Plywood
Stone
Stucco
VinylSd
Wd Sdng
Wd Shng


Count of MasVnrType
% Count
BrkCmn
BrkFace
None
Stone
MasVnrType

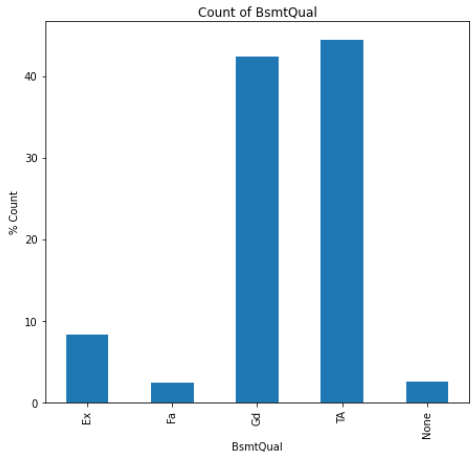

Count of ExterQual
% Count
Ex
Fa
Gd
TA
ExterQual

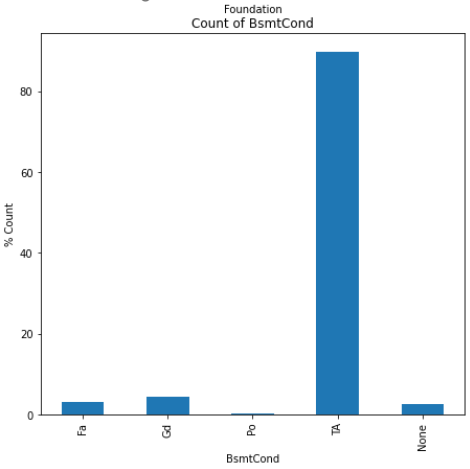

Count of ExterCond
% Count

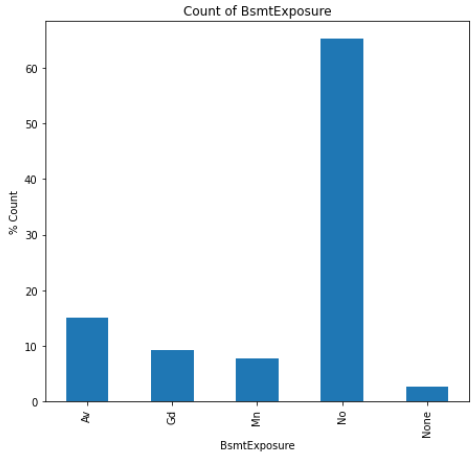

Count of Foundation
% Count

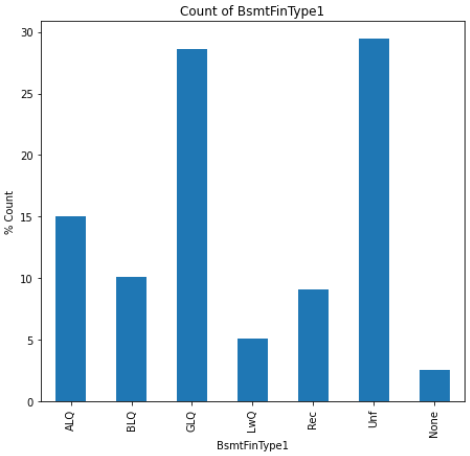

0
Ex
Fa
Gd
Po
TA
ExterCond

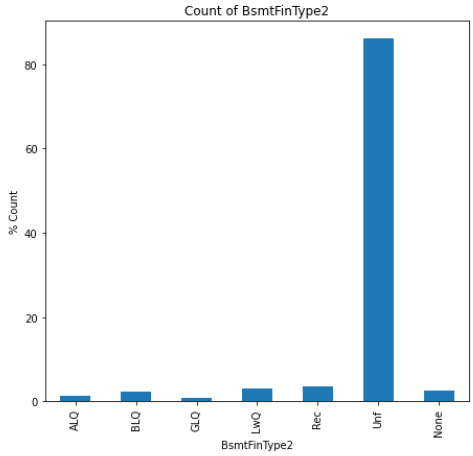

0
BrkTil
CBlock
PConc
Slab
Stone
Wood
Foundation

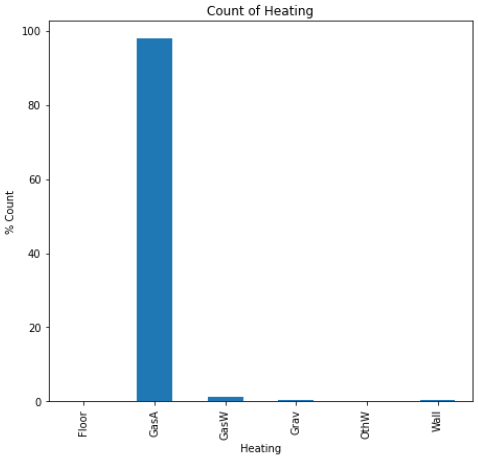

Count of BsmtQual
% Count
0
10
20
30
40
Ex
Fa
Gd
TA
None
BsmtQual

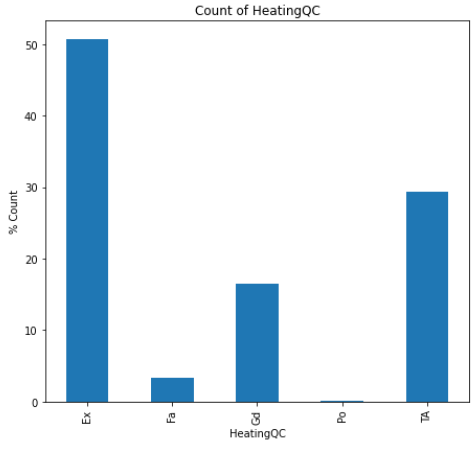

Count of BsmtCond
% Count
0
20
40
60
80
Fa
Gd
Po
TA
None
BsmtCond

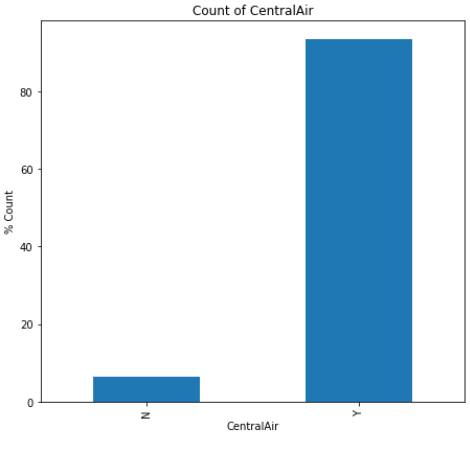

Count of BsmtExposure
% Count
0
10
20
30
40
50
60
Av
Gd
Mn
No
None
BsmtExposure

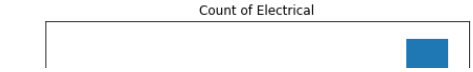

Count of BsmtFinType1
% Count
0
5
10
15
20
25
30
ALQ
BLQ
GLQ
LwQ
Rec
Unf
None
BsmtFinType1

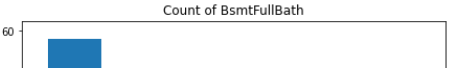

Count of BsmtFinType2
% Count
0
20
40
60
80
ALQ
BLQ
GLQ
LwQ
Rec
Unf
None
BsmtFinType2

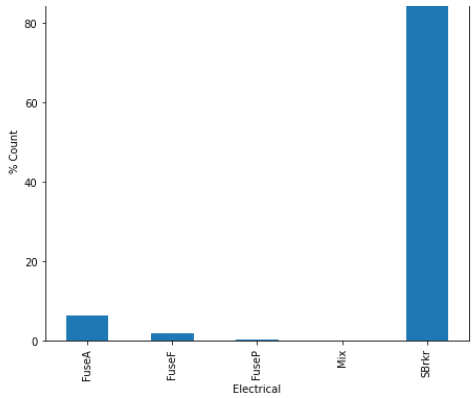

Count of Heating
% Count
0
20
40
60
80
100
Floor
GasA
GasW
Grav
OthW
Wall
Heating

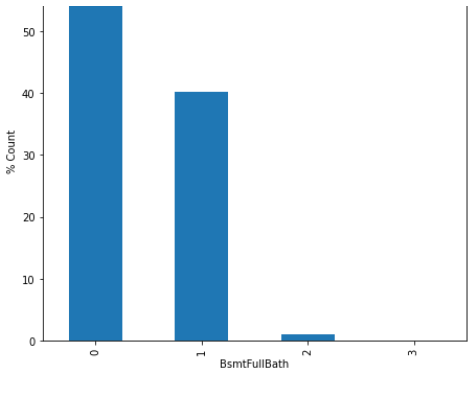

Count of HeatingQC
% Count
0
10
20
30
40
50
Ex
Fa
Gd
Po
TA
HeatingQC

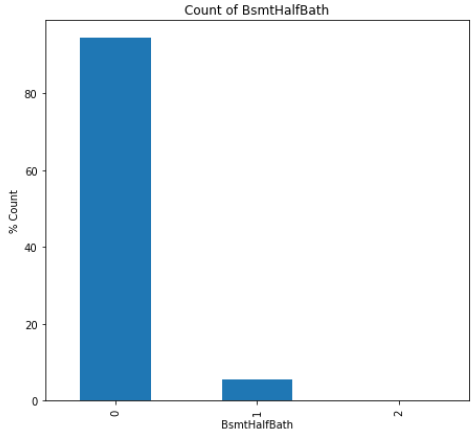

Count of CentralAir
% Count
0
20
40
60
80
N
Y
CentralAir

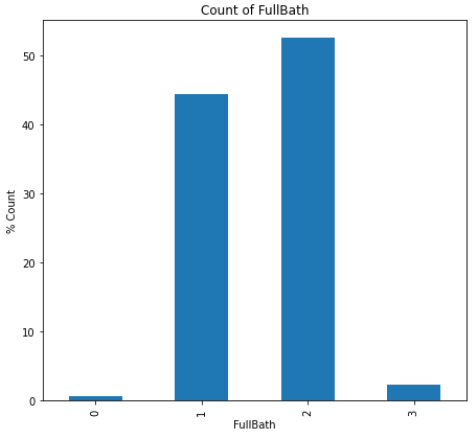

Count of Electrical

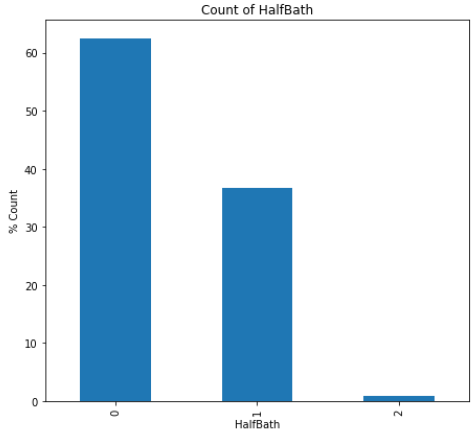

Count of BsmtFullBath
60

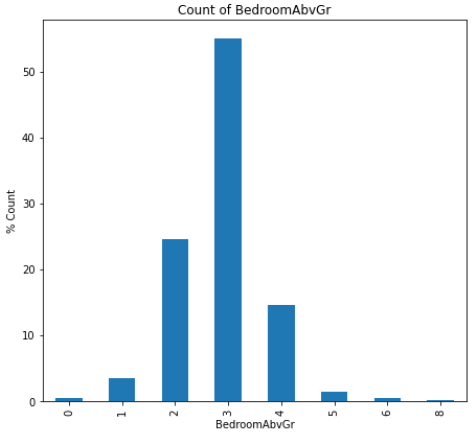

% Count
Electrical
FuseA
FuseF
FuseP
Mix
SBrkr

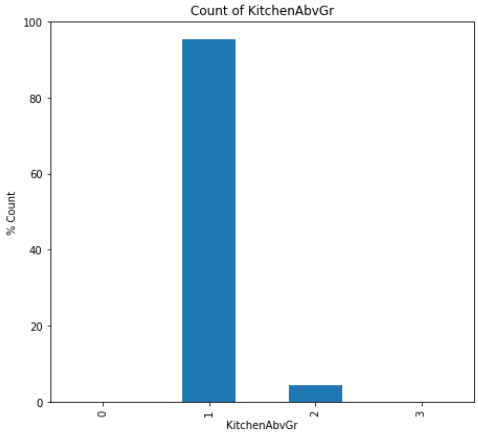

% Count
BsmtFullBath

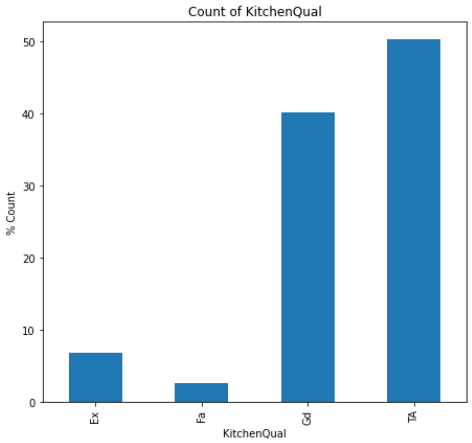

Count of BsmtHalfBath
% Count
BsmtHalfBath

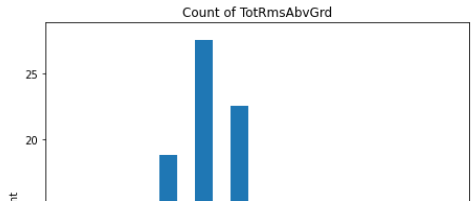

Count of FullBath
% Count
FullBath

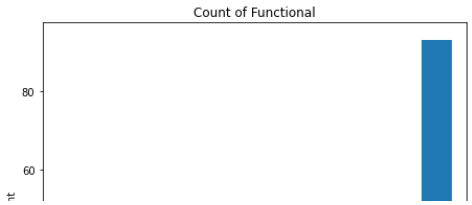

Count of HalfBath
% Count
HalfBath

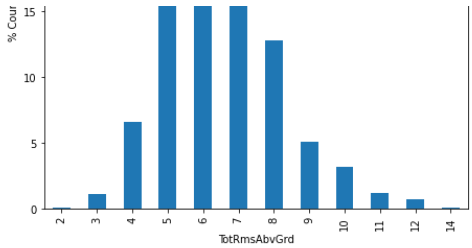

Count of BedroomAbvGr
% Count
BedroomAbvGr

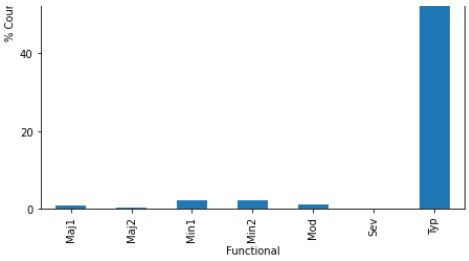

Count of KitchenAbvGr
% Count
KitchenAbvGr

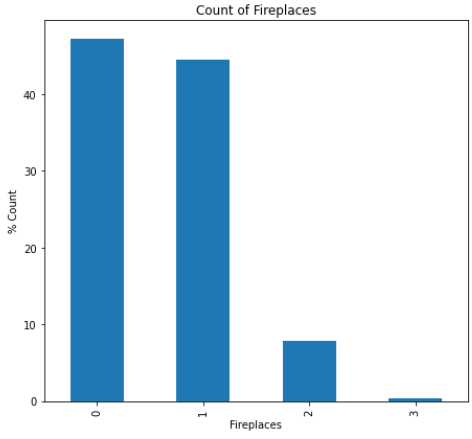

Count of KitchenQual
% Count
KitchenQual
Ex
Fa
Gd
TA

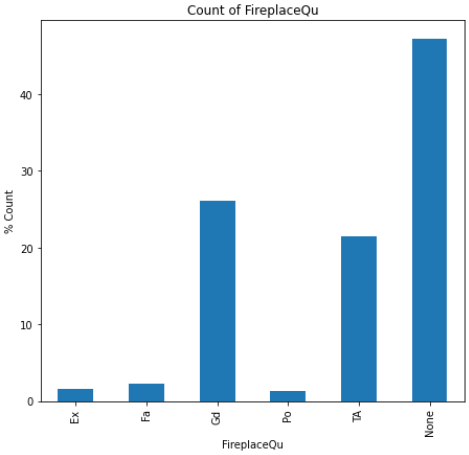

Count of TotRmsAbvGrd

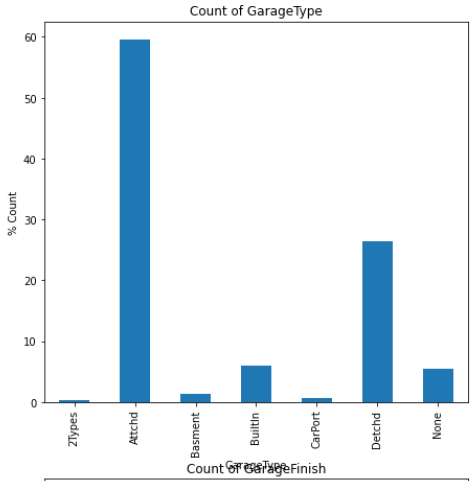

Count of Functional

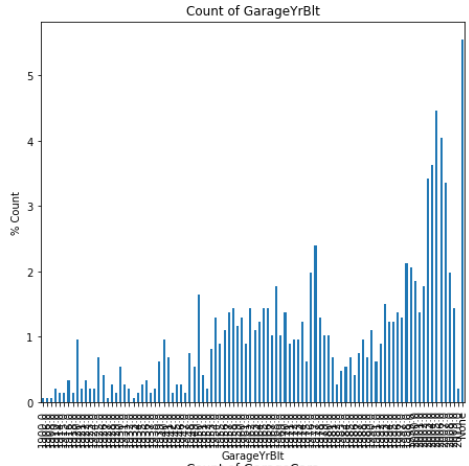

% Cour
TotRmsAbvGrd

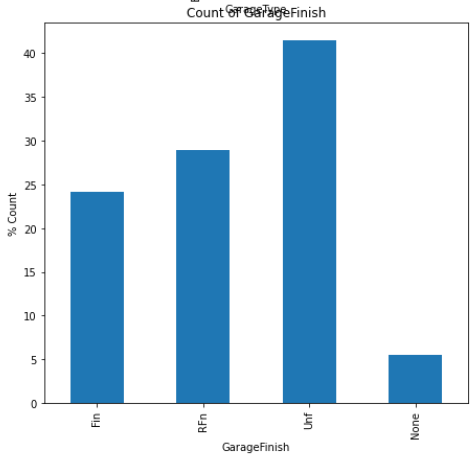

% Cour
Maj1
Maj2
Min1
Min2
Mod
Sev
Typ
Functional

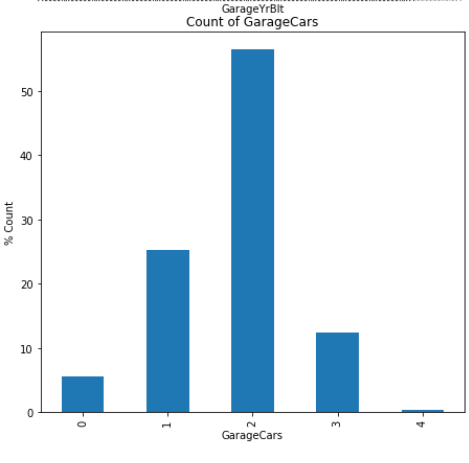

Count of Fireplaces
% Count
Fireplaces

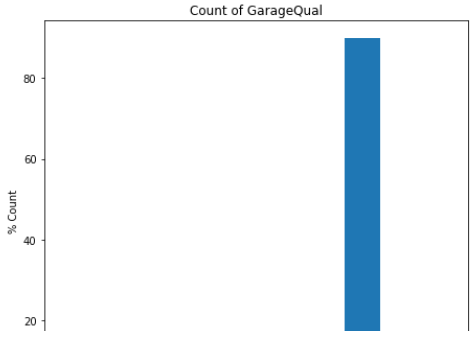

Count of FireplaceQu
% Count
Ex
Fa
Gd
Po
TA
None
FireplaceQu

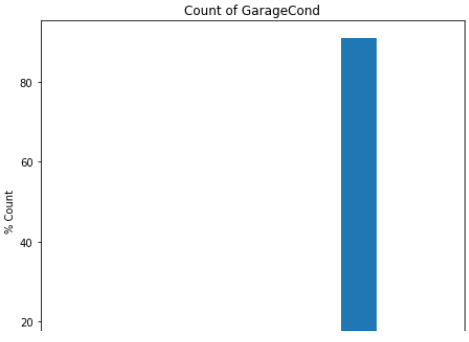

Count of GarageType
% Count
2Types
Attchd
Basment
BuiltIn
CarPort
Detchd
None
GarageType

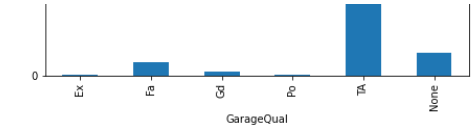

Count of GarageYrBlt
% Count
GarageYrBlt

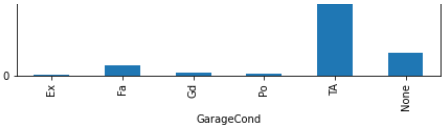

Count of GarageFinish
% Count
Fin
RFn
Unf
None
GarageFinish

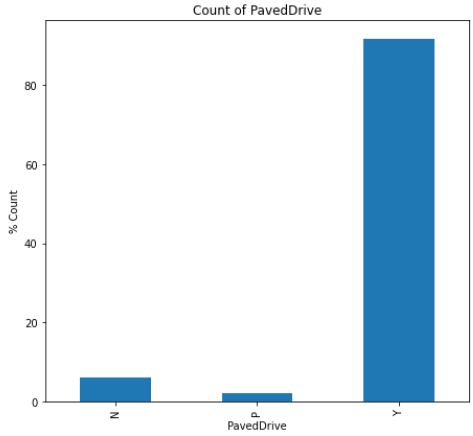

Count of GarageCars
% Count
GarageCars

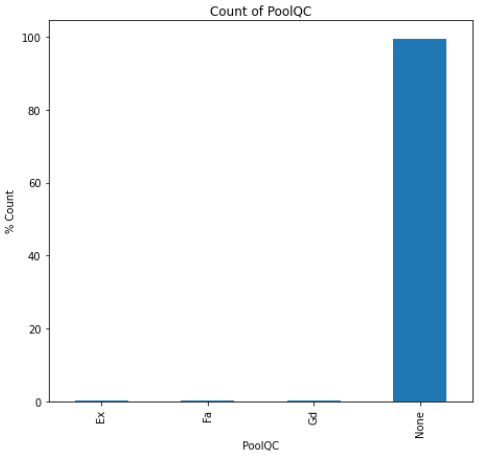

Count of GarageQual
% Count

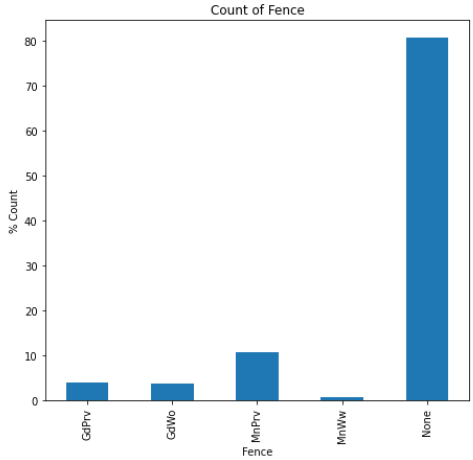

Count of GarageCond
% Count

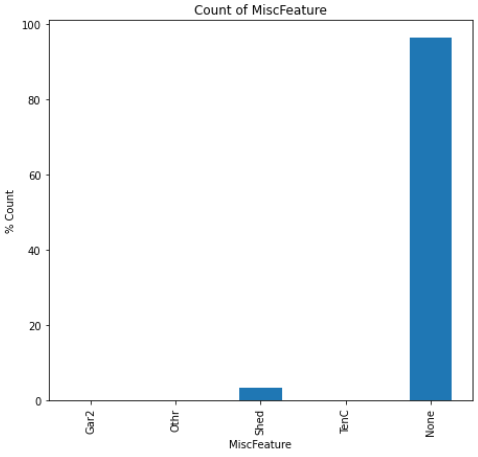

0
Ex
Fa
Gd
Po
TA
None
GarageQual

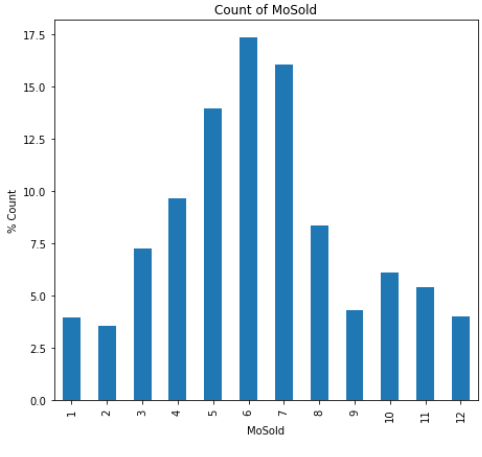

0
Ex
Fa
Gd
Po
TA
None
GarageCond

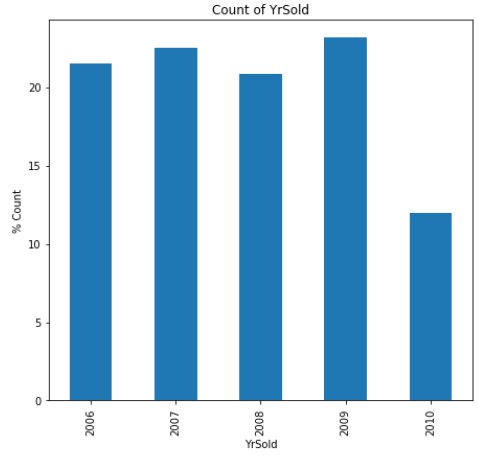

Count of PavedDrive
% Count
0
20
40
60
80
N
P
Y
PavedDrive

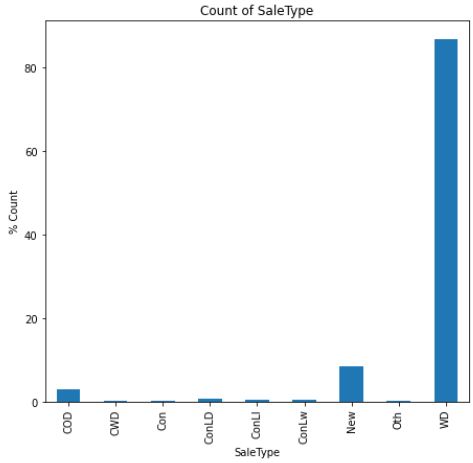

Count of PoolQC
% Count
0
20
40
60
80
100
Ex
Fa
Gd
None
PoolQC

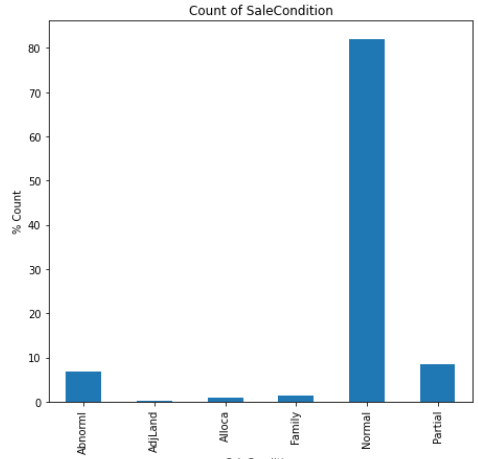

Count of Fence
% Count
0
10
20
30
40
50
60
70
80
GdPrv
GdWo
MnPrv
MnWw
None
Fence


Count of MiscFeature
% Count
0
20
40
60
80
100
Gar2
Othr
Shed
TenC
None
MiscFeature


Count of MoSold
% Count
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
1
2
3
4
5
6
7
8
9
10
11
12
MoSold


Count of YrSold
% Count
0
5
10
15
20
2006
2007
2008
2009
2010
YrSold


Count of SaleType
% Count
0
20
40
60
80
COD
CWD
Con
ConLD
ConLI
ConLw
New
Oth
WD
SaleType


Count of SaleCondition
% Count
0
10
20
30
40
50
60
70
80
Abnorml
AdjLand
Alloca
Family
Normal
Partial

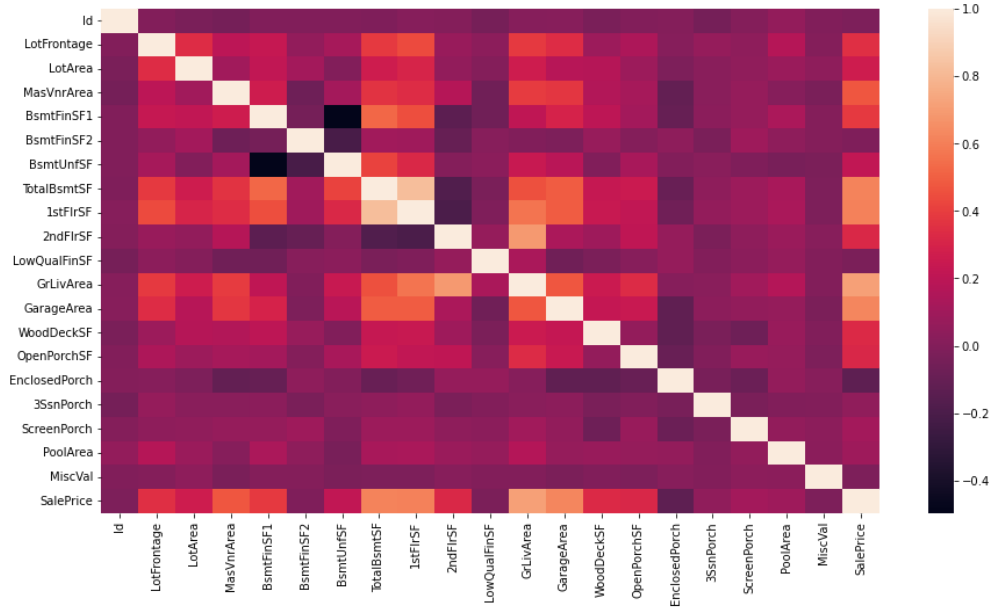
SaleCondition

Bivariable analysis

Numerical variables

In [30]:
```
plt.figure(figsize=(15,8))
sns.heatmap(df_train.corr())
```

Out[30]: <AxesSubplot:>



Categorical variables

Hypothesis 1

This analyzis will be answered within some hypothesis proposted.

Is the Exterior covering on house associated with lot configuration ?

LotConfig: Lot configuration

Category	*Meaning*
Inside	*Inside lot*
Corner	*Corner lot*
CulDSac	*Cul-de-sac*
FR2	*Frontage on 2 sides of property*

Category	Meaning
FR3	Frontage on 3 sides of property

Exterior1st: Exterior covering on house

Category	Meaning
AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

In [31]:
```
def plot_subplot2(itera,n,label,compare=''):
    row=int(len(itera)/n)+len(itera)%n
    colu=len(itera)-row*n

____
    if label=='hist':
        plt.figure(figsize=(16,6*row))
        for i,j in enumerate(itera):
            i+=1
            plt.subplot(row,n,i)
            sns.violinplot(data=df_train,y=j,kde=True,hue='Neighborh
            plt.title('Distribution of '+j)
    elif label=='cross':
            a=0
            b=0
            fig, axes=plt.subplots(row,n,figsize=(16,6*row))
            #print(row,n)
            for i,j in enumerate(itera):
                i+=1
                my_crosstab = pd.crosstab(index=df_train[j],
                                          columns=df_train[compare])
                if row==1:
```

```
                    #print('a',a,b)
                    my_crosstab.plot(kind='bar', stacked=True, rot=
                else:
                    #print(a,b)
                    my_crosstab.plot(kind='bar', stacked=True, rot=
                if b==n-1:
                    b=0
                    a+=1
                else:
                    b+=1
```

In [32]:
```
def chi_square_mul(itera):
    d={}
    from scipy.stats import chi2_contingency
    for j in itera:
        for i in itera:
            if not i==j:
                my_crosstab = pd.crosstab(index=df_train[i], columns
                # Chi-square test of independence.
                c, p, dof, expected = chi2_contingency(my_crosstab)
                # Print the p-value
                d.update({(i,j):p})
    return d
```

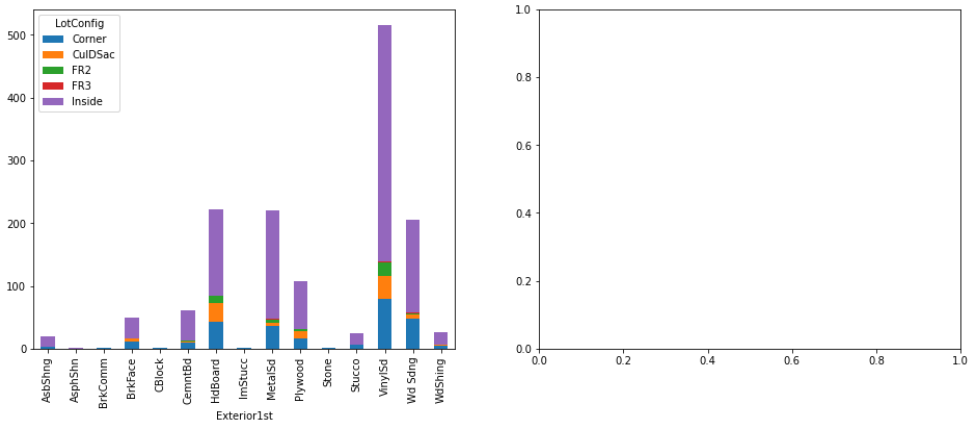
The hypothesis will be test looks like:

H_0***: LotConfig and Exterior1st are independent to each other among all subjects in the population***

H_a***: LotConfig and Exterior1st are NOT independent to each other among all subjects in the population***

Visual demostration

In [33]:
```
plot_subplot2(['Exterior1st'],2,'cross','LotConfig')
```



Statistical demostration

In [34]:
```
p_value_interest=chi_square_mul(('LotConfig','Exterior1st'))
print('With 95% of confidence, the p-value for LotConfig and Exteri
```

```
With 95% of confidence, the p-value for LotConfig and Exterior1st i
s 0.07
```

Taking account this:

- ***p(X) < 0.01 very strong evidence,***
- ***p(X) ∈ (0.01, 0.05) strong evidence,***
- ***p(X) ∈ (0.05, 0.1) weak evidence,***
- ***p(X) > 0.1 little or no evidence.***

This mean as there is a weak evidence to rejected the null hypothesis, it can be concluded that LotConfig and Exterior1st are independent to each other among all population

Hypothesis 2

Is the roof style of the house associated with Neighborhood ?

RoofStyle: Type of roof

Category
Flat
Gable
Gambrel
Hip
Mansard
Shed

Neighborhood: Physical locations within Ames city limits

Category	Details
Blmngtn	Bloomington Heights
Blueste	Bluestem
BrDale	Briardale
BrkSide	Brookside

Category	Details
ClearCr	Clear Creek
CollgCr	College Creek
Crawfor	Crawford
Edwards	Edwards
Gilbert	Gilbert
IDOTRR	Iowa DOT and Rail Road
MeadowV	Meadow Village
Mitchel	Mitchell
Names	North Ames
NoRidge	Northridge
NPkVill	Northpark Villa
NridgHt	Northridge Heights
NWAmes	Northwest Ames
OldTown	Old Town
SWISU	South & West of Iowa State University
Sawyer	Sawyer
SawyerW	Sawyer West
Somerst	Somerset
StoneBr	Stone Brook
Timber	Timberland
Veenker	Veenker

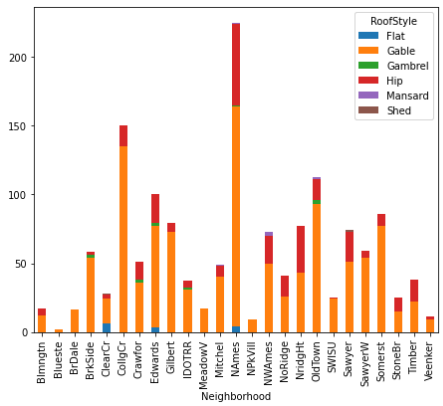
The hypothesis will be test looks like:

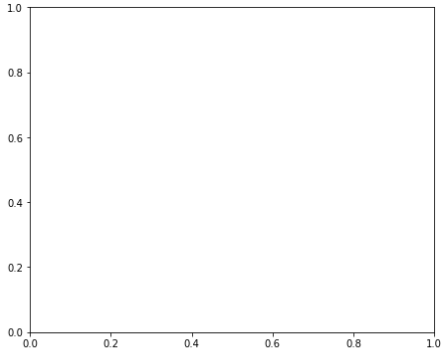
H_0***: RoofStyle and Neighborhood are independent to each other among all subjects in the population***

H_a***: RoofStyle and Neighborhood are NOT independent to each other among all subjects in the population***

Visual demostration

In [35]:
```
plot_subplot2(['Neighborhood'],2,'cross','RoofStyle')
```





Statistical demostration

In [36]:
```
p_value_interest=chi_square_mul(('Neighborhood','RoofStyle'))
print('With 95% of confidence, the p-value for Neighborhood and R
```

With 95% of confidence, the p-value for Neighborhood and RoofStyle is 0.00

Taking account this:

- ***p(X) < 0.01 very strong evidence,***
- ***p(X) ∈ (0.01, 0.05) strong evidence,***
- ***p(X) ∈ (0.05, 0.1) weak evidence,***
- ***p(X) > 0.1 little or no evidence.***

This mean as there is a very strong evidence that null hypothesis can be rejected, it can be concluded that LotConfig and Exterior1st are not significant independent to each other among all population

Hypothesis 3

Is the type of dwelling involved in the sale associated with the general zoning classification of the sale. ?

MSSubClass: Identifies the type of dwelling involved in the sale.

Category	Details
20	1-STORY 1946 & NEWER ALL STYLES

Category	Details
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

Category	Meaning
A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park
RM	Residential Medium Density

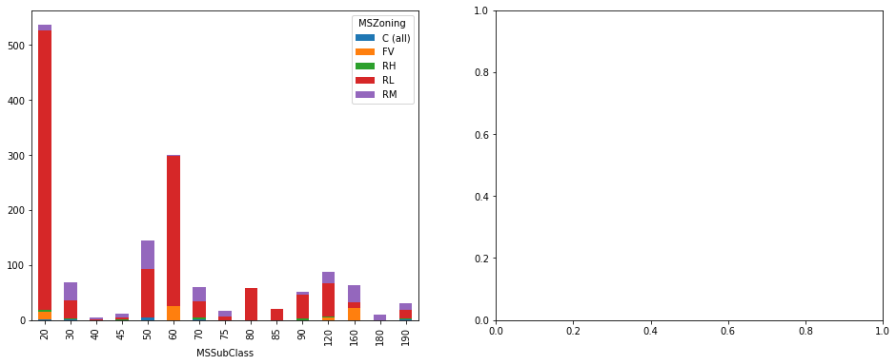
The hypothesis will be test looks like:

H_0***: MSSubClass and MSZoning are independent to each other among all subjects in the population***

H_a***: MSSubClass and MSZoning are NOT independent to each other among all subjects in the population***

Visual demostration

In [37]:
```
plot_subplot2(['MSSubClass'],2,'cross','MSZoning')
```



Statistical demostration

In [38]:
```
p_value_interest=chi_square_mul(('MSSubClass','MSZoning'))
print('With 95% of confidence, the p-value for Neighborhood an
```

With 95% of confidence, the p-value for Neighborhood and Roof Style is 0.00

Taking account this:

- ***p(X) < 0.01 very strong evidence,***
- ***p(X) ∈ (0.01, 0.05) strong evidence,***
- ***p(X) ∈ (0.05, 0.1) weak evidence,***
- ***p(X) > 0.1 little or no evidence.***

This mean as there is a very strong evidence that null hypothesis can be rejected, it can be concluded that LotConfig and Exterior1st are not significant independent to each other among all population