```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
sns.set(rc = {'figure.figsize':[10, 10]}, font_scale = 1.2)
```

```
# I'll import all csv files
df_train = pd.read_csv("/content/drive/MyDrive/House price prediction/train.csv")
df_test = pd.read_csv("/content/drive/MyDrive/House price prediction/test.csv")
```

```
df_train.head()
```

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

5 rows × 81 columns

From describtion we can see that

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

```
20    1-STORY 1946 & NEWER ALL STYLES
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.

```
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
```

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

```
Grvl    Gravel
Pave    Paved
```

Alley: Type of alley access to property

```
Grvl    Gravel
Pave    Paved
NA     No alley access
```

LotShape: General shape of property

```
Reg    Regular
IR1    Slightly irregular
IR2    Moderately Irregular
IR3    Irregular
```

LandContour: Flatness of the property

```
Lvl    Near Flat/Level
Bnk    Banked - Quick and significant rise from street grade to building
HLS    Hillside - Significant slope from side to side
Low    Depression
```

Utilities: Type of utilities available

```
AllPub    All public Utilities (E,G,W,& S)
NoSewr    Electricity, Gas, and Water (Septic Tank)
NoSeWa    Electricity and Gas Only
ELO    Electricity only
```

LotConfig: Lot configuration

```
Inside    Inside lot
Corner    Corner lot
CulDSac    Cul-de-sac
FR2    Frontage on 2 sides of property
FR3    Frontage on 3 sides of property
```

LandSlope: Slope of property

```
Gtl    Gentle slope
Mod    Moderate Slope
Sev    Severe Slope
```

Neighborhood: Physical locations within Ames city limits

```
Blmngtn    Bloomington Heights
Blueste    Bluestem
BrDale    Briardale
BrkSide    Brookside
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
```

Condition1: Proximity to various conditions

```
Artery    Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm    Normal
RRNn    Within 200' of North-South Railroad
RRAn    Adjacent to North-South Railroad
PosN    Near positive off-site feature--park, greenbelt, etc.
PosA    Adjacent to postive off-site feature
RRNe    Within 200' of East-West Railroad
RRAe    Adjacent to East-West Railroad
```

Condition2: Proximity to various conditions (if more than one is present)

```
Artery    Adjacent to arterial street
Feedr    Adjacent to feeder street
Norm    Normal
RRNn    Within 200' of North-South Railroad
RRAn    Adjacent to North-South Railroad
PosN    Near positive off-site feature--park, greenbelt, etc.
PosA    Adjacent to postive off-site feature
RRNe    Within 200' of East-West Railroad
RRAe    Adjacent to East-West Railroad
```

BldgType: Type of dwelling

```
1Fam    Single-family Detached
2FmCon    Two-family Conversion; originally built as one-family dwelling
Duplx    Duplex
```

```
TwnhsE    Townhouse End Unit
TwnhsI    Townhouse Inside Unit
```

HouseStyle: Style of dwelling

```
1Story    One story
1.5Fin    One and one-half story: 2nd level finished
1.5Unf    One and one-half story: 2nd level unfinished
2Story    Two story
2.5Fin    Two and one-half story: 2nd level finished
2.5Unf    Two and one-half story: 2nd level unfinished
SFoyer    Split Foyer
SLvl    Split Level
```

OverallQual: Rates the overall material and finish of the house

```
10    Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

OverallCond: Rates the overall condition of the house

```
10    Very Excellent
9    Excellent
8    Very Good
7    Good
6    Above Average
5    Average
4    Below Average
3    Fair
2    Poor
1    Very Poor
```

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

```
Flat    Flat
Gable    Gable
Gambrel    Gabrel (Barn)
Hip    Hip
Mansard    Mansard
Shed    Shed
```

RoofMatl: Roof material

```
ClyTile    Clay or Tile
CompShg    Standard (Composite) Shingle
Membran    Membrane
Metal    Metal
Roll    Roll
Tar&Grv    Gravel & Tar
WdShake    Wood Shakes
WdShngl    Wood Shingles
```

Exterior1st: Exterior covering on house

```
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
```

Exterior2nd: Exterior covering on house (if more than one material)

```
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
```

MasVnrType: Masonry veneer type

```
BrkCmn    Brick Common
BrkFace    Brick Face
CBlock    Cinder Block
None    None
Stone    Stone
```

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
Po    Poor
```

ExterCond: Evaluates the present condition of the material on the exterior

```
Ex    Excellent
Gd    Good
```

```
TA    Average/Typical
Fa    Fair
Po    Poor
```

Foundation: Type of foundation

```
BrkTil    Brick & Tile
CBlock    Cinder Block
PConc    Poured Contrete
Slab    Slab
Stone    Stone
Wood    Wood
```

BsmtQual: Evaluates the height of the basement

```
Ex    Excellent (100+ inches)
Gd    Good (90-99 inches)
TA    Typical (80-89 inches)
Fa    Fair (70-79 inches)
Po    Poor (<70 inches
NA    No Basement
```

BsmtCond: Evaluates the general condition of the basement

```
Ex    Excellent
Gd    Good
TA    Typical - slight dampness allowed
Fa    Fair - dampness or some cracking or settling
Po    Poor - Severe cracking, settling, or wetness
NA    No Basement
```

BsmtExposure: Refers to walkout or garden level walls

```
Gd    Good Exposure
Av    Average Exposure (split levels or foyers typically score average or above)
Mn    Mimimum Exposure
No    No Exposure
NA    No Basement
```

BsmtFinType1: Rating of basement finished area

```
GLQ    Good Living Quarters
ALQ    Average Living Quarters
BLQ    Below Average Living Quarters
Rec    Average Rec Room
LwQ    Low Quality
Unf    Unfinshed
NA    No Basement
```

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

```
GLQ    Good Living Quarters
ALQ    Average Living Quarters
BLQ    Below Average Living Quarters
Rec    Average Rec Room
LwQ    Low Quality
Unf    Unfinshed
NA    No Basement
```

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

```
Floor    Floor Furnace
GasA    Gas forced warm air furnace
GasW    Gas hot water or steam heat
Grav    Gravity furnace
OthW    Hot water or steam heat other than gas
Wall    Wall furnace
```

HeatingQC: Heating quality and condition

```
Ex    Excellent
Gd    Good
TA    Average/Typical
Fa    Fair
Po    Poor
```

CentralAir: Central air conditioning

```
N    No
Y    Yes
```

Electrical: Electrical system

```
SBrkr    Standard Circuit Breakers & Romex
FuseA    Fuse Box over 60 AMP and all Romex wiring (Average)
FuseF    60 AMP Fuse Box and mostly Romex wiring (Fair)
FuseP    60 AMP Fuse Box and mostly knob & tube wiring (poor)
Mix    Mixed
```

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

```
Ex    Excellent
Gd    Good
TA    Typical/Average
Fa    Fair
Po    Poor
```

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

```
Typ    Typical Functionality
Min1    Minor Deductions 1
```

```
Min2    Minor Deductions 2
Mod    Moderate Deductions
Maj1    Major Deductions 1
Maj2    Major Deductions 2
Sev    Severely Damaged
Sal    Salvage only
```

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

```
Ex    Excellent - Exceptional Masonry Fireplace
Gd    Good - Masonry Fireplace in main level
TA    Average - Prefabricated Fireplace in main living area or Masonry Fireplace in basement
Fa    Fair - Prefabricated Fireplace in basement
Po    Poor - Ben Franklin Stove
NA    No Fireplace
```

GarageType: Garage location

```
2Types    More than one type of garage
Attchd    Attached to home
Basment    Basement Garage
BuiltIn    Built-In (Garage part of house - typically has room above garage)
CarPort    Car Port
Detchd    Detached from home
NA    No Garage
```

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

```
Fin    Finished
RFn    Rough Finished
Unf    Unfinished
NA    No Garage
```

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

GarageCond: Garage condition

```
Ex   Excellent
Gd   Good
TA   Typical/Average
Fa   Fair
Po   Poor
NA   No Garage
```

PavedDrive: Paved driveway

```
Y   Paved
P   Partial Pavement
N   Dirt/Gravel
```

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

```
Ex   Excellent
Gd   Good
TA   Average/Typical
Fa   Fair
NA   No Pool
```

Fence: Fence quality

```
GdPrv   Good Privacy
MnPrv   Minimum Privacy
GdWo   Good Wood
MnWw   Minimum Wood/Wire
NA   No Fence
```

MiscFeature: Miscellaneous feature not covered in other categories

```
Elev   Elevator
Gar2   2nd Garage (if not described in garage section)
Othr   Other
Shed   Shed (over 100 SF)
TenC   Tennis Court
NA   None
```

MiscVal: $Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

```
WD    Warranty Deed - Conventional
CWD   Warranty Deed - Cash
VWD   Warranty Deed - VA Loan
New   Home just constructed and sold
COD   Court Officer Deed/Estate
Con   Contract 15% Down payment regular terms
ConLw   Contract Low Down payment and low interest
ConLI   Contract Low Interest
ConLD   Contract Low Down
Oth   Other
```

SaleCondition: Condition of sale

```
Normal   Normal Sale
Abnorml   Abnormal Sale -  trade, foreclosure, short sale
AdjLand   Adjoining Land Purchase
Alloca   Allocation - two linked properties with separate deeds, typically condo with a garage un
Family   Sale between family members
Partial   Home was not completed when last assessed (associated with New Homes)
```

```
df_train.describe().T # T for transform as we have like 81 column
# from describtion we should see if there were any outliers
```

	count	mean	std	min	25%	50%	7
Id	1460.0	730.500000	421.610009	1.0	365.75	730.5	1095.2
MSSubClass	1460.0	56.897260	42.300571	20.0	20.00	50.0	70.0
LotFrontage	1201.0	70.049958	24.284752	21.0	59.00	69.0	80.0
LotArea	1460.0	10516.828082	9981.264932	1300.0	7553.50	9478.5	11601.5
OverallQual	1460.0	6.099315	1.382997	1.0	5.00	6.0	7.0
OverallCond	1460.0	5.575342	1.112799	1.0	5.00	5.0	6.0
YearBuilt	1460.0	1971.267808	30.202904	1872.0	1954.00	1973.0	2000.0
YearRemodAdd	1460.0	1984.865753	20.645407	1950.0	1967.00	1994.0	2004.0
MasVnrArea	1452.0	103.685262	181.066207	0.0	0.00	0.0	166.0
BsmtFinSF1	1460.0	443.639726	456.098091	0.0	0.00	383.5	712.2

```
df_train.info()
```

```
 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
```

```
 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
```

```
df_train.shape
```

```
(1460, 81)
```

```
## by looking to 81 columns we could see our first insights
null_percentage = df_train.isnull().sum() * 100 / len(df_train)
null_percentage = pd.DataFrame(data=null_percentage)
pd.set_option('display.max_rows', 81)
null_percentage
```

	0
Id	0.000000
MSSubClass	0.000000
MSZoning	0.000000
LotFrontage	17.739726
LotArea	0.000000
Street	0.000000
Alley	93.767123
LotShape	0.000000
LandContour	0.000000
Utilities	0.000000
LotConfig	0.000000
LandSlope	0.000000
Neighborhood	0.000000
Condition1	0.000000
Condition2	0.000000
BldgType	0.000000
HouseStyle	0.000000
OverallQual	0.000000
OverallCond	0.000000
YearBuilt	0.000000
YearRemodAdd	0.000000
RoofStyle	0.000000
RoofMatl	0.000000
Exterior1st	0.000000
Exterior2nd	0.000000
MasVnrType	0.547945
MasVnrArea	0.547945
ExterQual	0.000000
ExterCond	0.000000
Foundation	0.000000

BsmtQual	2.534247
BsmtCond	2.534247
BsmtExposure	2.602740
BsmtFinType1	2.534247
BsmtFinSF1	0.000000
BsmtFinType2	2.602740
BsmtFinSF2	0.000000
BsmtUnfSF	0.000000
TotalBsmtSF	0.000000
Heating	0.000000
HeatingQC	0.000000
CentralAir	0.000000
Electrical	0.068493
1stFlrSF	0.000000
2ndFlrSF	0.000000
LowQualFinSF	0.000000
GrLivArea	0.000000
BsmtFullBath	0.000000
BsmtHalfBath	0.000000
FullBath	0.000000
HalfBath	0.000000
BedroomAbvGr	0.000000
KitchenAbvGr	0.000000
KitchenQual	0.000000
TotRmsAbvGrd	0.000000
Functional	0.000000
Fireplaces	0.000000
FireplaceQu	47.260274
GarageType	5.547945
GarageYrBlt	5.547945
GarageFinish	5.547945
GarageCars	0.000000

GarageArea	0.000000
GarageQual	5.547945
GarageCond	5.547945
PavedDrive	0.000000
WoodDeckSF	0.000000
OpenPorchSF	0.000000
EnclosedPorch	0.000000
3SsnPorch	0.000000
ScreenPorch	0.000000
PoolArea	0.000000

As a first insight more than 50% of data we should drop columns from it

- PoolQC is 99% null
- MiscFeature is 96% null
- Fence is 80.7% null
- Alley 93.767123% null

```
df_train_copy = df_train.copy()
```

SaleType	0.000000

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

```
df_train_copy.describe().T
```

	count	mean	std	min	25%	50%	7
Id	1460.0	730.500000	421.610009	1.0	365.75	730.5	1095.
MSSubClass	1460.0	56.897260	42.300571	20.0	20.00	50.0	70.
LotFrontage	1201.0	70.049958	24.284752	21.0	59.00	69.0	80.
LotArea	1460.0	10516.828082	9981.264932	1300.0	7553.50	9478.5	11601.
OverallQual	1460.0	6.099315	1.382997	1.0	5.00	6.0	7.
OverallCond	1460.0	5.575342	1.112799	1.0	5.00	5.0	6.
YearBuilt	1460.0	1971.267808	30.202904	1872.0	1954.00	1973.0	2000.
YearRemodAdd	1460.0	1984.865753	20.645407	1950.0	1967.00	1994.0	2004.
MasVnrArea	1452.0	103.685262	181.066207	0.0	0.00	0.0	166.
BsmtFinSF1	1460.0	443.639726	456.098091	0.0	0.00	383.5	712.
BsmtFinSF2	1460.0	46.549315	161.319273	0.0	0.00	0.0	0.
BsmtUnfSF	1460.0	567.240411	441.866955	0.0	223.00	477.5	808.
TotalBsmtSF	1460.0	1057.429452	438.705324	0.0	795.75	991.5	1298.
1stFlrSF	1460.0	1162.626712	386.587738	334.0	882.00	1087.0	1391.
2ndFlrSF	1460.0	346.992466	436.528436	0.0	0.00	0.0	728.
LowQualFinSF	1460.0	5.844521	48.623081	0.0	0.00	0.0	0.
GrLivArea	1460.0	1515.463699	525.480383	334.0	1129.50	1464.0	1776.
BsmtFullBath	1460.0	0.425342	0.518911	0.0	0.00	0.0	1.
BsmtHalfBath	1460.0	0.057534	0.238753	0.0	0.00	0.0	0.
FullBath	1460.0	1.565068	0.550916	0.0	1.00	2.0	2.
HalfBath	1460.0	0.382877	0.502885	0.0	0.00	0.0	1.
BedroomAbvGr	1460.0	2.866438	0.815778	0.0	2.00	3.0	3.
KitchenAbvGr	1460.0	1.046575	0.220338	0.0	1.00	1.0	1.
TotRmsAbvGrd	1460.0	6.517808	1.625393	2.0	5.00	6.0	7.
Fireplaces	1460.0	0.613014	0.644666	0.0	0.00	1.0	1.
GarageYrBlt	1379.0	1978.506164	24.689725	1900.0	1961.00	1980.0	2002.
GarageCars	1460.0	1.767123	0.747315	0.0	1.00	2.0	2.

```
pd.set_option('display.max_columns', 77)
```

WoodDeckSF	1460.0	94.244521	125.338794	0.0	0.00	0.0	168.

```
df_train_copy.head()
```

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

```
## So our final goal is to predict price column depending on various inputs
## So we should tune values and going through each column
df_train_copy['MSZoning'].value_counts()
```

```
RL         1151
RM          218
FV           65
RH           16
C (all)      10
Name: MSZoning, dtype: int64
```

```
df_train_copy['Street'].value_counts()
```

```
Pave    1454
Grvl       6
Name: Street, dtype: int64
```

```
df_train_copy['LotShape'].value_counts()
```

```
Reg    925
IR1    484
IR2     41
IR3     10
Name: LotShape, dtype: int64
```

```
df_train_copy['LandContour'].value_counts()
```

```
Lvl    1311
Bnk      63
HLS      50
Low      36
Name: LandContour, dtype: int64
```

```
df_train_copy['Utilities'].value_counts()
```

```
AllPub    1459
NoSeWa       1
Name: Utilities, dtype: int64
```

```
df_train_copy['LotConfig'].value_counts()
```

```
Inside     1052
Corner      263
CulDSac      94
FR2          47
FR3           4
Name: LotConfig, dtype: int64
```

```
df_train_copy['LandSlope'].value_counts()
```

```
Gtl    1382
Mod      65
Sev      13
Name: LandSlope, dtype: int64
```

```
df_train_copy['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
StoneBr     25
SWISU       25
MeadowV     17
Blmngtn     17
BrDale      16
Veenker     11
NPkVill      9
Blueste      2
Name: Neighborhood, dtype: int64
```

```
df_train_copy['Condition1'].value_counts()
```

```
Norm      1260
Feedr       81
```

```
Artery      48
RRAn        26
PosN        19
RRAe        11
PosA         8
RRNn         5
RRNe         2
Name: Condition1, dtype: int64
```

```
df_train_copy['BldgType'].value_counts()
```

```
1Fam      1220
TwnhsE     114
Duplex      52
Twnhs       43
2fmCon      31
Name: BldgType, dtype: int64
```

```
df_train_copy['HouseStyle'].value_counts()
```

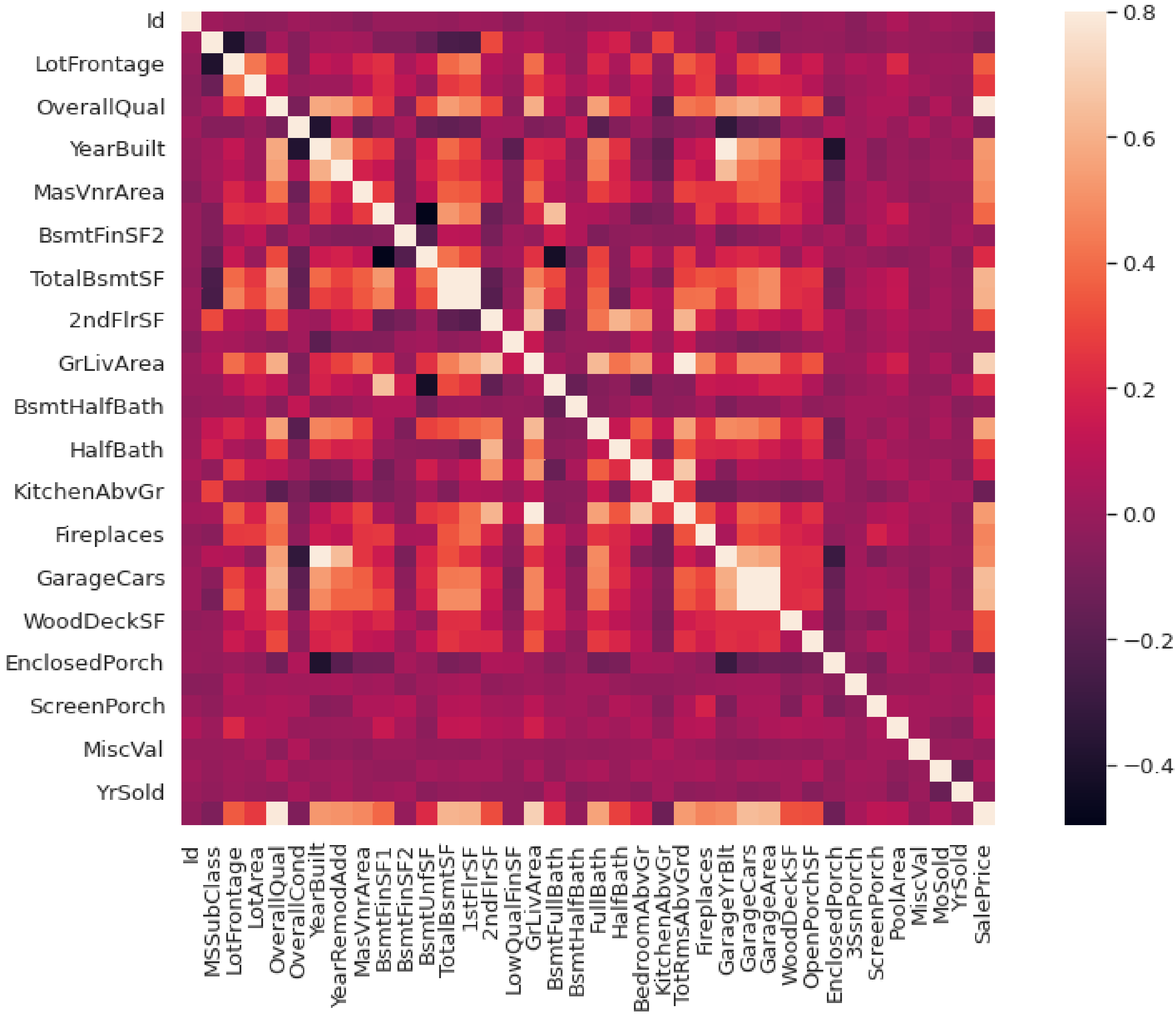
```
1Story    726
2Story    445
1.5Fin    154
SLvl       65
SFoyer     37
1.5Unf     14
2.5Unf     11
2.5Fin      8
Name: HouseStyle, dtype: int64
```

```
df_train_copy.shape
```

```
(1460, 77)
```

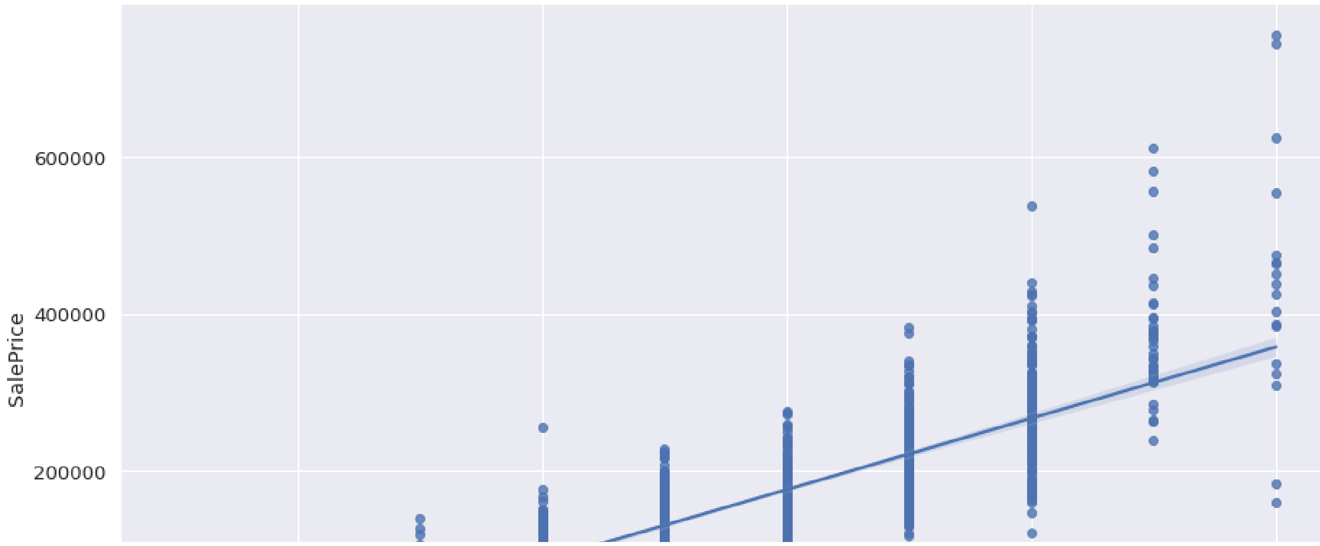
- MSZoning: Identifies the general zoning classification of the sale.
- Street: Type of road access to property
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions

```
plt.rcParams['figure.figsize'] = (15.0, 9.0)
train_corr = df_train_copy.corr()
sns.heatmap(train_corr, vmax=.8, square=True);
```



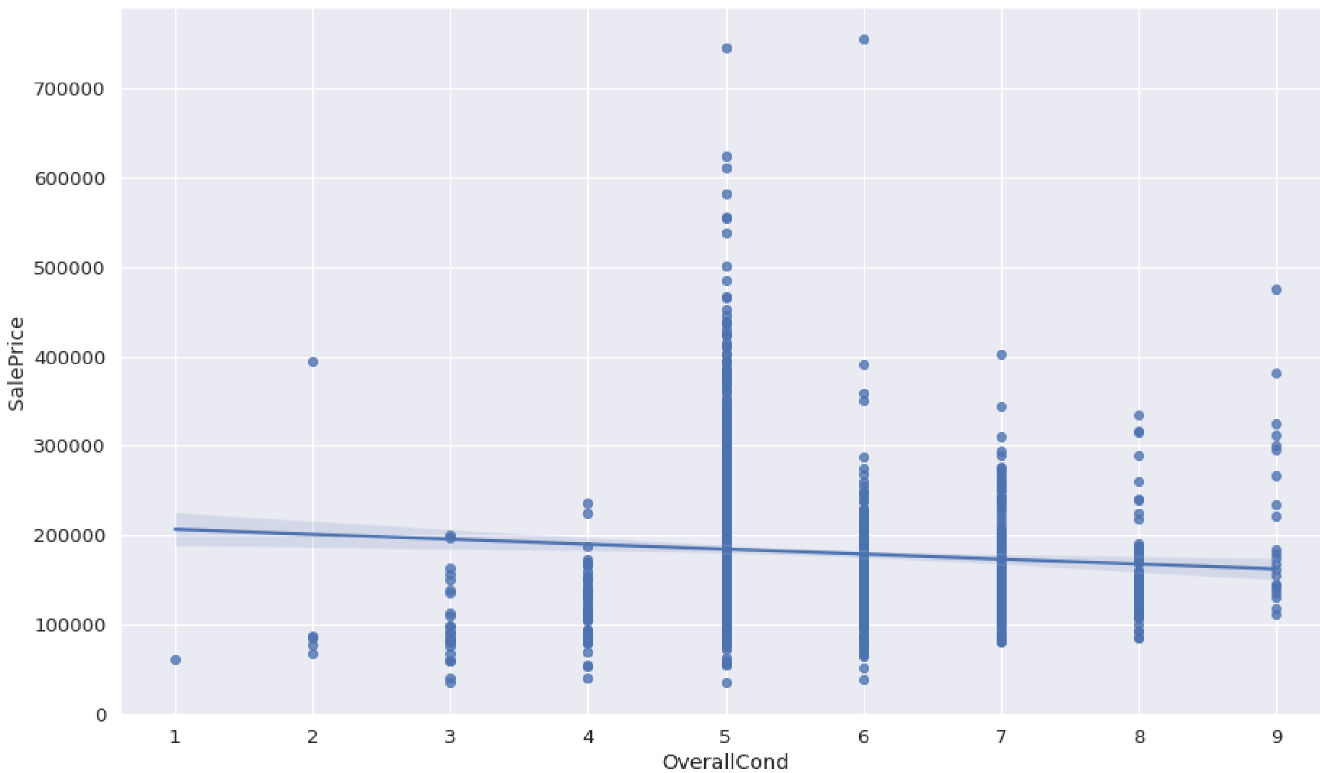

```
## We will be getting correlation between price and overall quality, condition quality
sns.regplot(x=df_train_copy["OverallQual"], y=df_train_copy["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c251ffb90>
```




```
sns.regplot(x=df_train_copy["OverallCond"], y=df_train_copy["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c251e1910>
```



- lotFrontage and LotArea in square feet so we will convert it into meters

- square feet * 0.092903 = 1 meter

- LotArea is feet length So to convert from feet length into meter length
  - feet length * 0.3048 = 1 meter
- Is there like a relation between LotFrontage and LotArea
- OverallQual and condition is categorical values Like from bad to excellent
- Is there any changing happend? YearRemodAdd - YearBuilt = if 0 = No if else = yes
- Is there any specific physical location that make our saleprice high

conditions 1 & 2 are conditions that may exist Norm or not

- We can make a new column that have 3 types of conditions
  - if condition 1 = condition 2 return norm elif condition 1 = norm and condition 2 != norm return half norm if condition 1 != condition 2 return not norm

Exterior 1st & 2nd: Exterior covering on house

Masonry veneer type and ExterQual and ExterCond

- Evaluates the quality of the material on the exterior relation and will keep ExterQual
- ExterCond: Evaluates the present condition of the material on the exterior Masonry veneer area in square feet

BsmtQual in inches --> height of basement

- Ex excellent (100+ inches)
- Gd Good (90-99 inches)
- TA Typical (80-89 inches)
- Fa Fair (70-79 inches)
- Po Poor (<70 inches)
- NA No Basement

1 inch * 0.0254 = 1 meter

BsmtFin

BsmtCond general condition

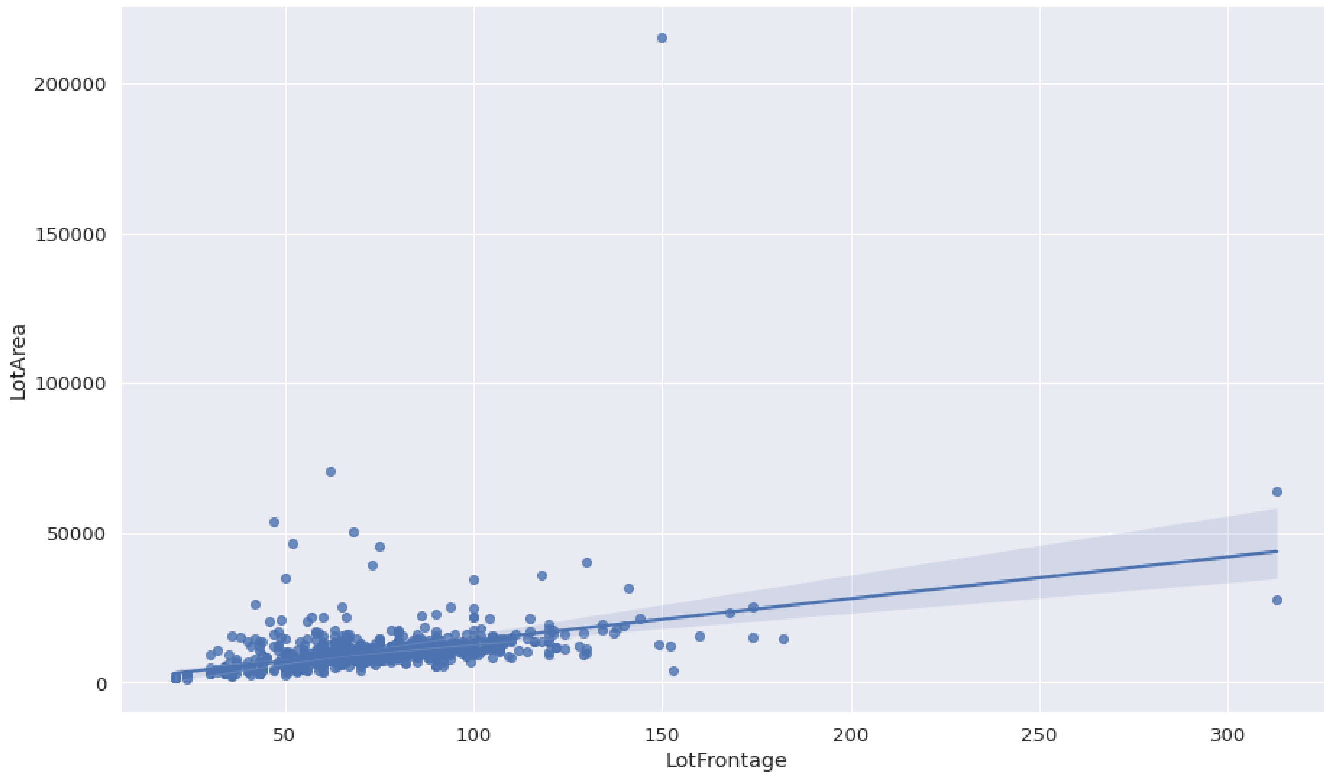
BldgType: Type of dwellingType get_dummies

Is there like a relation between LotFrontage and LotArea?

```
# Is there like a relation between LotFrontage and LotArea
```

```
sns.regplot(x=df_train_copy["LotFrontage"], y=df_train_copy["LotArea"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c23979ad0>
```



We can see like there's an outlier in LotArea

```
## remove outliers
q1 = df_train_copy["LotArea"].quantile(0.25)
q3 = df_train_copy["LotArea"].quantile(0.75)
iqr = q3-q1
min_wisk = q1 - 1.5 * iqr
max_wisk = q3 + 1.5 * iqr


clean_train = df_train_copy[  df_train_copy['LotArea'].between(min_wisk, max_wisk)  ]
clean_train
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	Reg	Lvl
2	3	60	RL	68.0	11250	Pave	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	IR1	Lvl
...	...	...	...	...	...	...	...	...
1455	1456	60	RL	62.0	7917	Pave	Reg	Lvl
1456	1457	20	RL	85.0	13175	Pave	Reg	Lvl
1457	1458	70	RL	66.0	9042	Pave	Reg	Lvl
1458	1459	20	RL	68.0	9717	Pave	Reg	Lvl
1459	1460	20	RL	75.0	9937	Pave	Reg	Lvl

1391 rows × 77 columns

```
# after removing outliers
sns.regplot(x=clean_train["LotFrontage"], y=clean_train["LotArea"])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f5c2387bc50>


```
## No duplicates
df_train_copy[  df_train_copy.duplicated()  ]
```

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utiliti

```
df_train_copy.groupby('YearBuilt').sum()
```

YearBuilt	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearRemod
1872	1350	70	50.0	5250	8	5	19
1875	1138	50	54.0	6342	5	8	19
1880	2817	285	292.0	48986	25	26	79
1882	992	70	121.0	17671	8	9	19
1885	1524	220	120.0	22140	8	13	39
...	...	...	...	...	...	...	
2006	52205	3480	5120.0	696963	507	335	134
2007	33947	2100	3833.0	512359	379	249	98
2008	20009	980	2122.0	323885	199	115	46
2009	10660	940	1273.0	159521	134	90	36
2010	379	20	88.0	11394	9	2	20

112 rows × 37 columns

```
df_train_copy.groupby('YearRemodAdd').sum()
```

YearRemodAdd	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearB
1950	138589	9870	9878.0	1582031	894	984	34
1951	3415	80	356.0	46045	18	20	[illegible]
1952	3161	200	328.0	41454	23	24	[illegible]
1953	5248	410	712.0	114691	54	48	1
1954	11092	505	878.0	146056	69	73	2
1955	5246	590	507.0	114802	40	49	1
1956	6142	225	490.0	100921	48	54	1
1957	6615	180	579.0	88731	46	47	1
1958	11794	730	947.0	150570	79	83	2
1959	12051	450	1198.0	187173	93	97	3
1960	6785	420	704.0	150114	63	62	2
1961	5100	230	466.0	70089	39	39	1
1962	10298	685	826.0	161477	73	76	2
1963	10680	505	861.0	155837	68	69	2
1964	8867	510	583.0	137140	62	62	2
1965	12113	965	1315.0	544798	98	103	3
1966	11865	735	780.0	148502	79	82	2
1967	8614	615	515.0	142829	60	60	2
1968	14388	620	1064.0	172267	94	92	3
1969	13117	690	743.0	175337	76	75	2
1970	20854	2435	1138.0	208276	138	143	5
1971	12026	1370	704.0	129629	92	93	3
1972	12859	1465	966.0	156412	102	119	3
1973	7659	1200	478.0	64036	63	58	2
1974	6708	455	525.0	72463	40	36	1
1975	6018	480	545.0	208763	55	53	1
1976	21564	2420	1071.0	284073	177	171	5
1977	20750	1470	1440.0	251230	142	139	4
1978	13532	1180	582.0	157150	94	90	3

1979	8206	750	466.0	107860	55	52	1
1980	8126	1010	590.0	96526	71	70	2
1981	4920	325	365.0	96178	50	48	1
1982	3610	530	375.0	55900	34	37	1
1983	3625	165	151.0	46755	26	28	
1984	4470	630	162.0	61262	46	40	1
1985	7157	450	340.0	79235	55	49	1
1986	3786	400	288.0	52782	35	31	
1987	6061	550	533.0	149513	65	60	1
1988	6764	410	565.0	102390	58	52	1
1989	5795	690	446.0	125657	69	61	2
1990	11179	745	1042.0	177638	97	85	2
1991	10245	840	879.0	155973	88	83	2
1992	13380	1005	973.0	199900	107	96	3
1993	9362	1165	847.0	195381	125	114	3
1994	19574	1215	1331.0	291724	138	132	4
1995	20375	1725	2140.0	330605	185	186	6
1996	27997	2100	2042.0	375554	235	211	7
1997	14524	1375	1403.0	315222	153	154	4
1998	22719	2210	2014.0	363408	240	202	7
1999	22760	2360	1597.0	275728	200	162	5
2000	42218	4025	2495.0	471712	351	319	10
2001	13197	960	815.0	190305	137	125	4
2002	34060	2095	2480.0	597444	322	278	9
2003	37933	2845	2745.0	523190	331	296	10
2004	40260	4225	3556.0	525751	410	345	12
2005	55009	4550	4957.0	734927	503	408	14
2006	71672	4920	6809.0	1160478	675	522	19
2007	58523	3940	5229.0	747226	538	426	15
2008	33342	1900	3198.0	497236	306	216	7
2009	14431	1010	1711.0	228470	174	121	4
2010	4100	260	407.0	59743	47	30	1

Is there any changing happend through years??

- YearRemodAdd - YearBuilt = if 0 = No if else = yes

Why this we want to know if there was any change or not and we will drop 2 year columns

```
# clean_train
clean_train['year_diff'] = clean_train['YearRemodAdd']- clean_train['YearBuilt']
clean_train['year_diff']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

0        0
1        0
2        1
3       55
4        0
        ..
1455     1
1456    10
1457    65
1458    46
1459     0
Name: year_diff, Length: 1391, dtype: int64
```

```
clean_train['year_diff'] = clean_train['year_diff'].apply(lambda x: 0 if x == 0 else 1)
clean_train['year_diff']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """Entry point for launching an IPython kernel.
0       0
1       0
2       1
3       1
4       0
       ..
1455    1
1456    1
1457    1
1458    1
1459    0
Name: year_diff, Length: 1391, dtype: int64
```

```
clean_train.loc[clean_train['SalePrice'] == clean_train['SalePrice'].max()]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	U
1182	1183	60	RL	160.0	15623	Pave	IR1	Lvl	

1 rows × 78 columns

```
clean_train.loc[clean_train['SalePrice'] == clean_train['SalePrice'].min()]
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Ut
495	496	30	C (all)	60.0	7879	Pave	Reg	Lvl	

1 rows × 78 columns

- 1996 Year built without any modification and sold on 2007 but 745000 SalePrice
- While 1920 and modified on 1950 and sold on 2009 is the minimum with 34900 SalePrice
- So let's check the relation between year sold and SalePrice

```
sns.regplot(x=clean_train["YrSold"], y=clean_train["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c237f9a10>
```

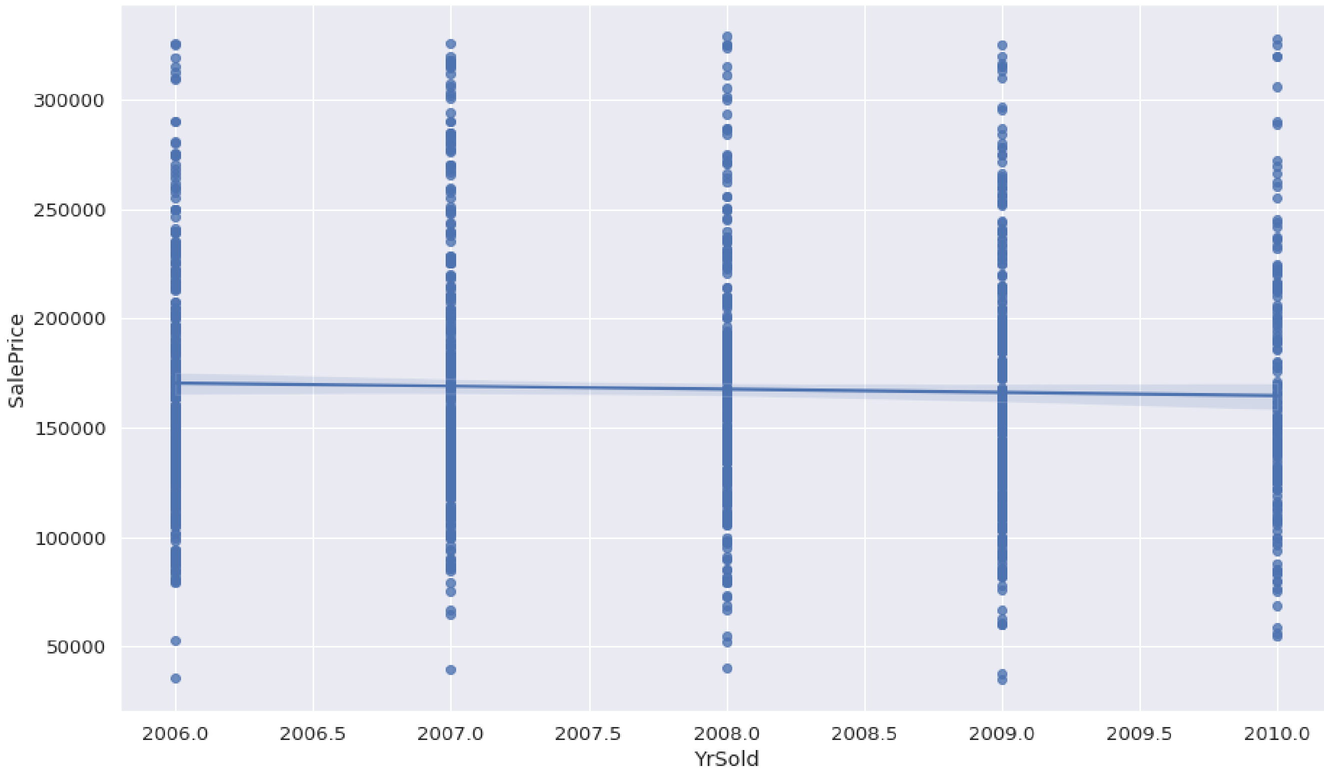
As we can see There's an outlier in SalePrice

```
## remove outliers
q1 = clean_train["SalePrice"].quantile(0.25)
q3 = clean_train["SalePrice"].quantile(0.75)
iqr = q3-q1
min_wisk = q1 - 1.5 * iqr
max_wisk = q3 + 1.5 * iqr


clean_train = clean_train[  clean_train['SalePrice'].between(min_wisk, max_wisk)  ]
clean_train
```

```
sns.regplot(x=clean_train["YrSold"], y=clean_train["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c23747090>
```




```
clean_train['Condition1'].value_counts()
```

```
Norm      1147
Feedr       77
Artery      44
RRAn        23
PosN        16
RRAe        11
RRNn         5
PosA         5
RRNe         2
Name: Condition1, dtype: int64
```

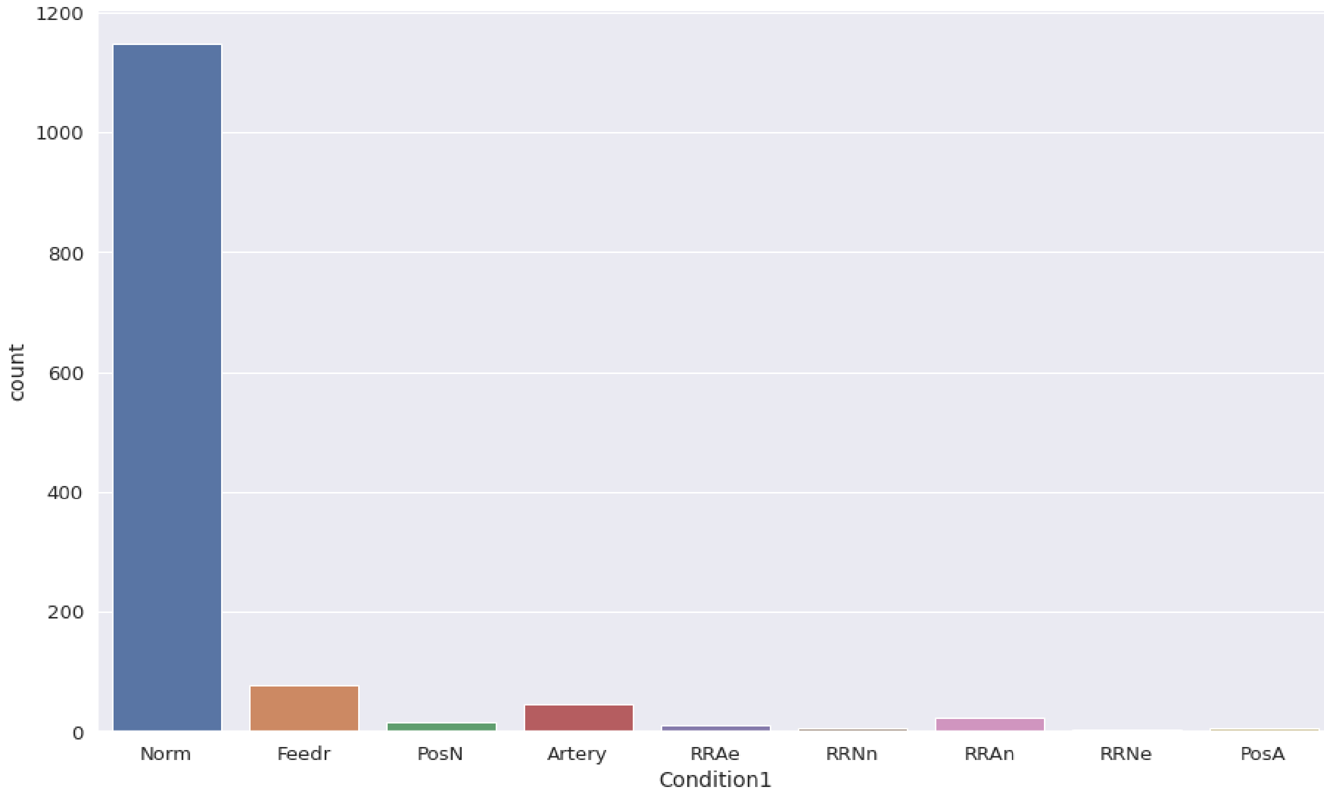
```
clean_train['Condition2'].value_counts()
```

```
Norm      1318
Feedr        6
```

```
Artery       2
RRNn         2
PosA         1
RRAn         1
Name: Condition2, dtype: int64
```

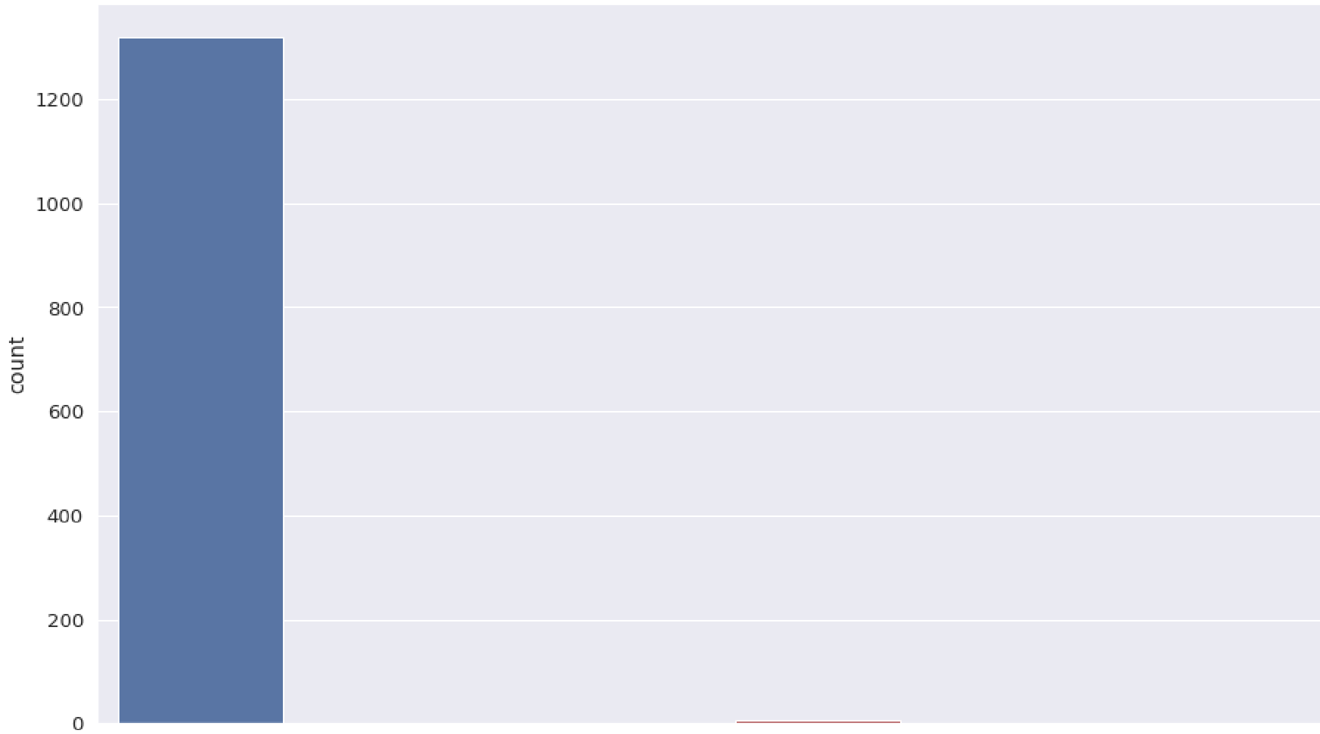
```
sns.countplot(x = "Condition1", data = clean_train)
```

```
&lt;matplotlib.axes._subplots.AxesSubplot at 0x7f5c236c7d90&gt;
```



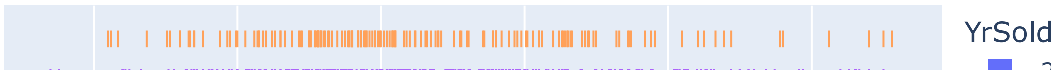

```
sns.countplot(x = "Condition2", data = clean_train)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c2364d050>
```




```
# sns.pairplot(clean_train)
```

```
import plotly.express as px
fig = px.histogram(clean_train, x="SalePrice", color="YrSold", marginal='rug',hover_data=clea
fig.show()
```




```
# import plotly.express as px
fig = px.histogram(clean_train, x="SalePrice", color="MoSold", marginal='rug',hover_data=clea
fig.show()
```

We can see that the count of houses sold was on Month 6 and the most year houses sold was on 2010

```
clean_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
```

```
          'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2',
          'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC',
          'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
          'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
          'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
          'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
          'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
          'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
          'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
          'SaleCondition', 'SalePrice', 'year_diff'],
         dtype='object')
```

```
# Now let's drop YearBuilt and YearRemodAdd since we got a column to express if they remod or
clean_train.drop(['YearBuilt', 'YearRemodAdd'], axis=1, inplace = True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
## After that we can calculate the percentage of Bsmt Finished
clean_train['finish_percentage'] = (clean_train['BsmtFinSF1'] + clean_train['BsmtFinSF2'])/ c
clean_train['Unfinished_percentage'] = clean_train['BsmtUnfSF']/clean_train['TotalBsmtSF']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
clean_train.drop(['BsmtUnfSF', 'BsmtFinSF1', 'BsmtUnfSF', 'TotalBsmtSF'], axis=1, inplace = T
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning


A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

```
clean_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond',
       'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
       'BsmtFinType2', 'BsmtFinSF2', 'Heating', 'HeatingQC', 'CentralAir',
       'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Functional',
       'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice', 'year_diff', 'finish_percentage',
       'Unfinished_percentage'],
      dtype='object')
```

```
clean_train.shape
```

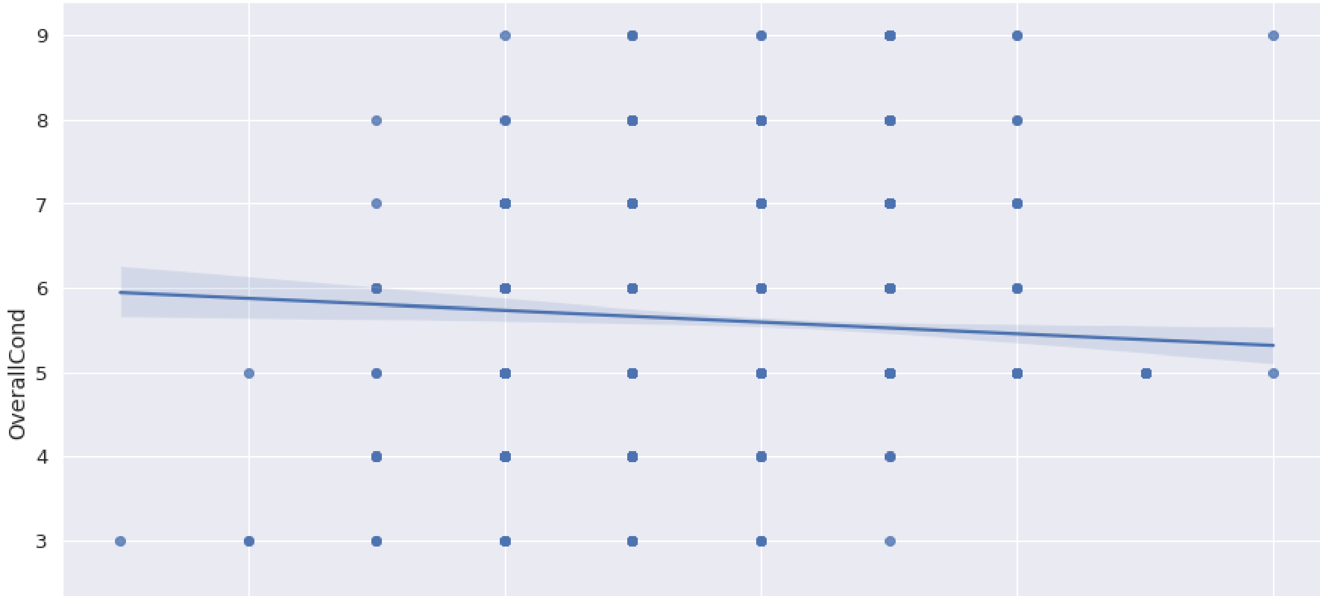
```
(1330, 75)
```

Is there's any relation between those two categorical values and which affects on SalePrice (condqual and overallqual)?

- **Does it really mean overall quality have an overall good condition?**

```
sns.regplot(x=clean_train["OverallQual"], y=clean_train["OverallCond"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c23641450>
```



- Does it really mean that have an overall very good condition will have most sales?

OverallQual

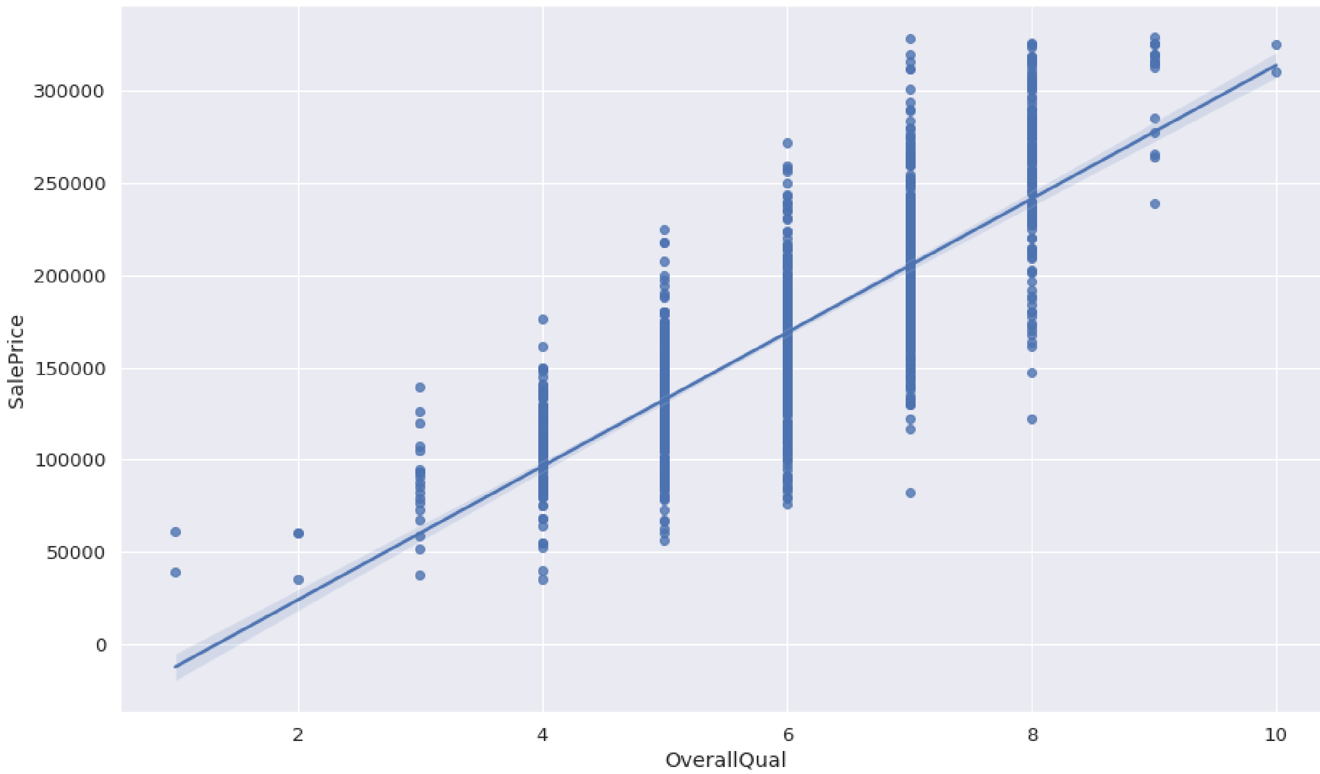
```
sns.regplot(x=clean_train["OverallCond"], y=clean_train["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c1ffd1450>
```

- Does it really Saleprice will be sold at high prices depending on an overallquality?

```
sns.regplot(x=clean_train["OverallQual"], y=clean_train["SalePrice"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c20299510>
```




```
clean_train.columns
```

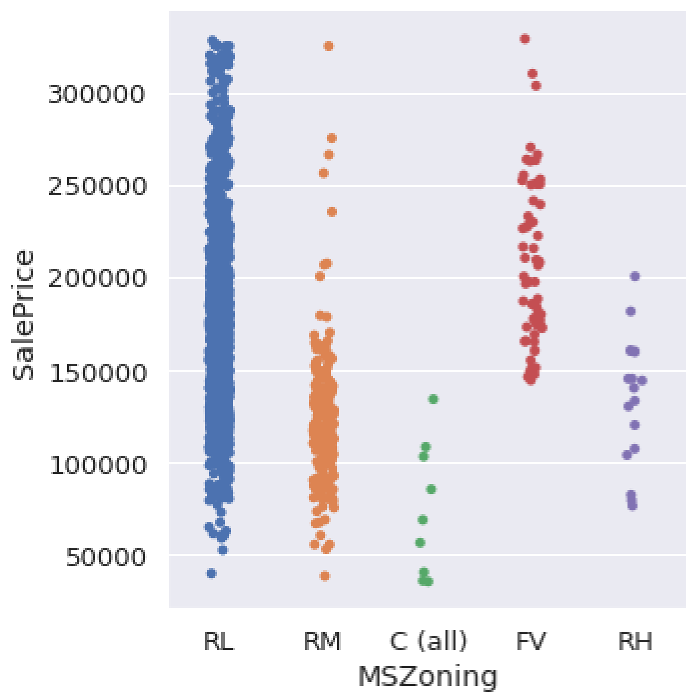
```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'RoofStyle', 'RoofMatl', 'Exterior1st',
       'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond',
       'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
       'BsmtFinType2', 'BsmtFinSF2', 'Heating', 'HeatingQC', 'CentralAir',
```

```
       'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Functional',
       'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice', 'year_diff', 'finish_percentage',
       'Unfinished_percentage'],
      dtype='object')
```

Is there any specific physical location that make our saleprice high?

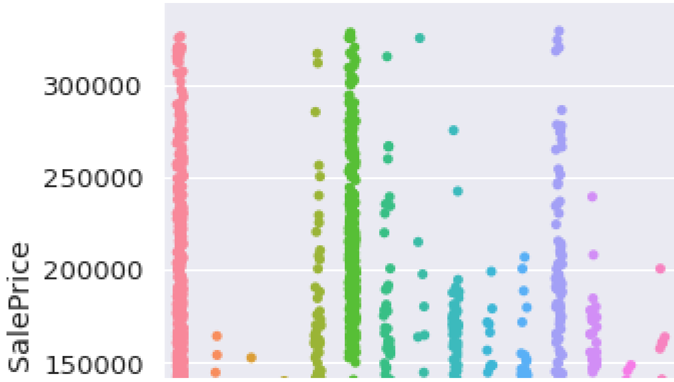
```
sns.catplot(x="MSZoning", y="SalePrice", data=clean_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5c2039d590>
```



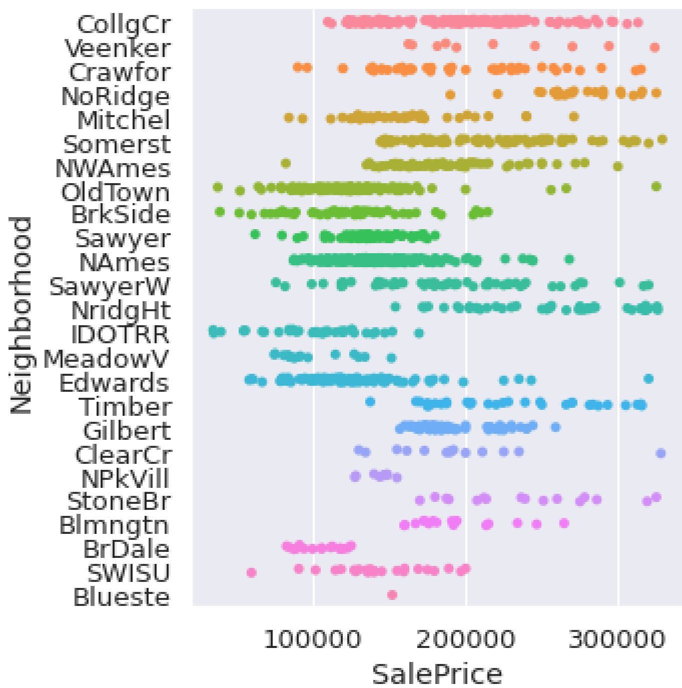

```
sns.catplot(x="MSSubClass", y="SalePrice", data=clean_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5c20395750>
```



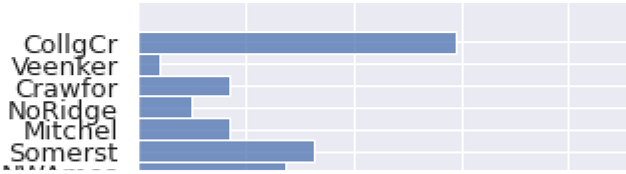

```
sns.catplot(x="SalePrice", y="Neighborhood", data=clean_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5c2042e050>
```



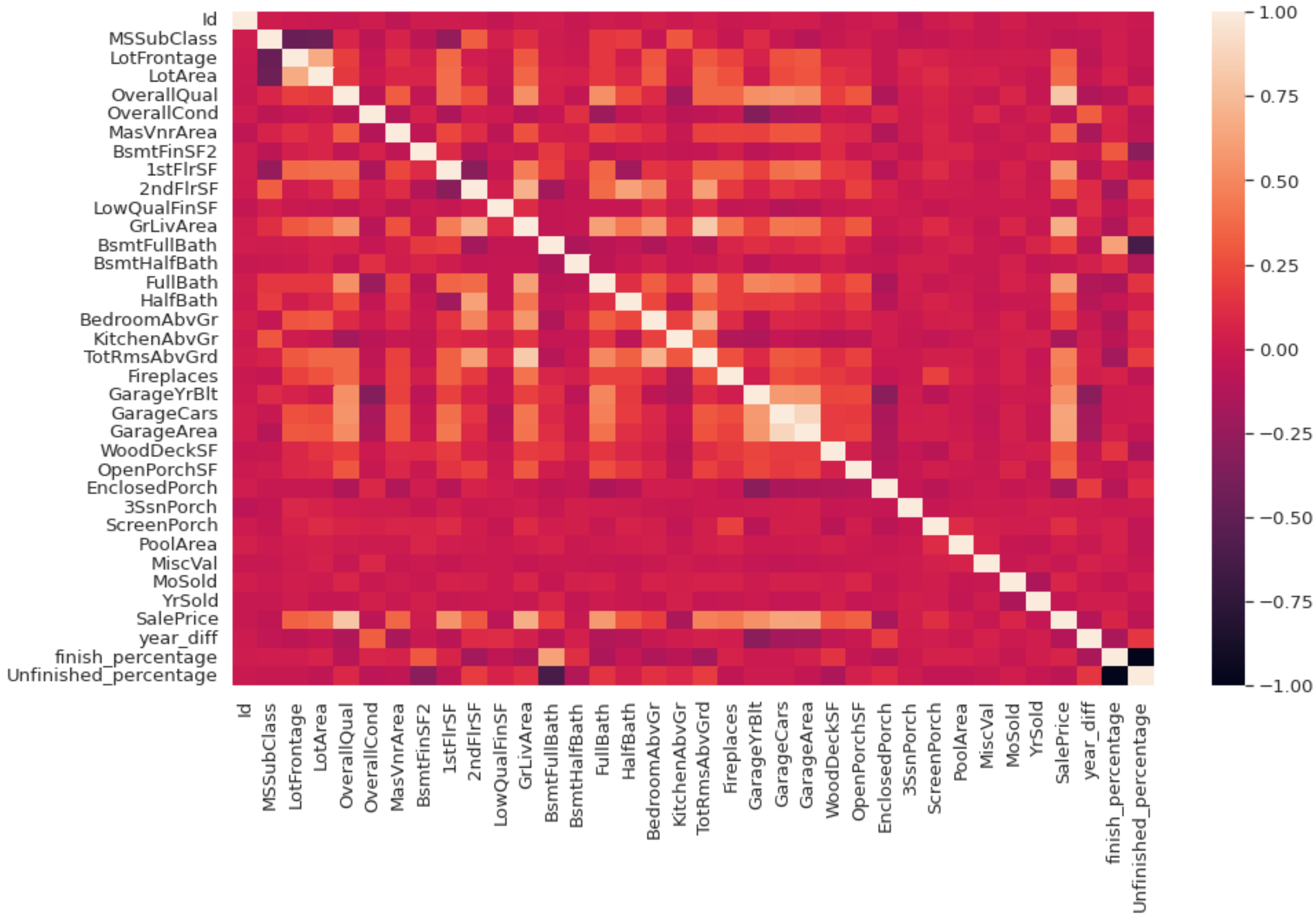

```
sns.displot(y = 'Neighborhood', data = clean_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5c200a0b10>
```




```
sns.heatmap(clean_train.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c1fef9550>
```




```
clean_train.groupby('Neighborhood').describe()['SalePrice']
```

	count	mean	std	min	25%	50%	75%
Neighborhood							
Blmngtn	17.0	194870.882353	30393.229219	159895.0	174000.0	191000.0	213490.0
Blueste	1.0	151000.000000	NaN	151000.0	151000.0	151000.0	151000.0
BrDale	16.0	104493.750000	14330.176493	83000.0	91000.0	106000.0	118000.0
BrkSide	57.0	123103.070175	38473.366272	39300.0	100000.0	121600.0	140200.0
ClearCr	14.0	193745.142857	49351.905344	130000.0	164375.0	190000.0	208250.0
CollgCr	148.0	195175.851351	45660.170260	110000.0	151625.0	195950.0	224900.0
Crawfor	43.0	195247.372093	54829.917748	90350.0	155950.0	188700.0	232000.0
Edwards	95.0	124919.473684	40523.361593	58500.0	100000.0	119000.0	140000.0
Gilbert	71.0	189142.830986	23186.865619	156932.0	174000.0	181000.0	193750.0
IDOTRR	36.0	100655.000000	33691.157825	34900.0	82875.0	104750.0	121750.0
MeadowV	16.0	99737.500000	23752.357778	75000.0	84250.0	89500.0	118000.0
Mitchel	43.0	158792.697674	37530.521770	84500.0	132500.0	156000.0	171250.0
NAmes	218.0	143548.591743	28793.111580	87500.0	126625.0	140000.0	156875.0
NPkVill	9.0	142694.444444	9377.314529	127500.0	140000.0	146000.0	148500.0
NWAmes	69.0	187467.463768	37014.976794	82500.0	165000.0	181000.0	202500.0
NoRidge	25.0	275097.600000	30447.960014	190000.0	260000.0	271000.0	290000.0
NridgHt	48.0	257235.958333	50731.803692	154000.0	209000.0	270000.0	306750.0
OldTown	111.0	123598.729730	38142.190055	37900.0	105450.0	117500.0	138000.0
SWISU	25.0	142591.360000	32622.917679	60000.0	128000.0	139500.0	160000.0
Sawyer	70.0	135938.457143	21483.575035	62383.0	127250.0	134950.0	149012.5
SawyerW	59.0	186555.796610	55651.997820	76000.0	145500.0	179900.0	222500.0
Somerst	82.0	218402.902439	46854.079636	144152.0	177125.0	219000.0	250435.0
StoneBr	16.0	239312.500000	48426.877868	170000.0	202875.0	237750.0	275750.0
Timber	31.0	231333.967742	51890.284286	137500.0	185750.0	224500.0	275606.5

```
clean_train.groupby('MSSubClass').describe()['SalePrice']
```

	count	mean	std	min	25%	50%	75%	
MSSubClass								
20	483.0	170629.140787	56397.589364	35311.0	130125.0	155000.0	204450.0	32
30	67.0	95809.716418	25085.954453	34900.0	81250.0	99900.0	110250.0	16
40	3.0	121500.000000	37593.217473	79500.0	106250.0	133000.0	142500.0	15
45	12.0	108591.666667	20231.723889	76000.0	94125.0	107500.0	122250.0	13
50	137.0	137278.306569	44170.935470	37900.0	112000.0	130500.0	149000.0	31
60	253.0	217460.505929	44914.856160	130500.0	183000.0	210000.0	250000.0	32
70	56.0	159649.017857	51451.607502	40000.0	127750.0	154287.5	178625.0	31
75	13.0	159538.461538	61530.636522	101000.0	118500.0	144000.0	179500.0	32
80	53.0	163600.943396	27189.793211	107000.0	146800.0	164500.0	175000.0	27
85	20.0	147810.000000	19629.942220	123000.0	134350.0	140750.0	158375.0	19
90	50.0	132222.720000	27159.841804	82000.0	118125.0	135480.0	144750.0	20
120	85.0	196564.541176	51815.134289	99500.0	155900.0	191000.0	224000.0	32

```
clean_train.groupby('MSZoning').describe()['SalePrice']
```

	count	mean	std	min	25%	50%	75%	
MSZoning								
C (all)	9.0	73808.888889	35759.614502	34900.0	40000.00	68400.0	102776.0	13
FV	62.0	207423.967742	43709.982987	144152.0	173799.75	198450.0	240500.0	32
RH	16.0	131558.375000	35714.118435	76000.0	106150.00	136500.0	148608.5	20
RL	1030.0	176102.956311	56346.606483	39300.0	135000.00	167500.0	210000.0	32
RM	213.0	122866.521127	36685.889631	37900.0	100000.00	120000.0	140000.0	32

```
sns.jointplot(x= 'MSZoning', y = 'SalePrice', data = clean_train)
```

```
<seaborn.axisgrid.JointGrid at 0x7f5c203af610>
```

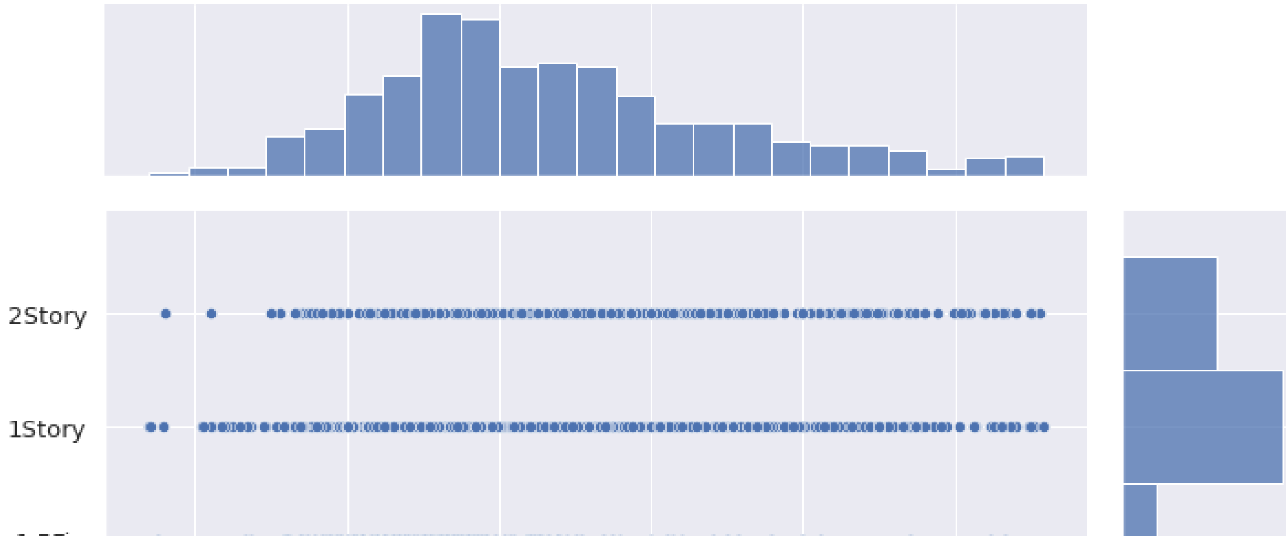
- RL is the safest place where we can find all types of saleprice from low to high
- FV is a bit more risky with highest SalePrice

BUT

What's my insight? We have like +1k row that containing RL so we have a good info about all the prices included in that place While we have much more less data in FV... So Is that insight for real we may need more data

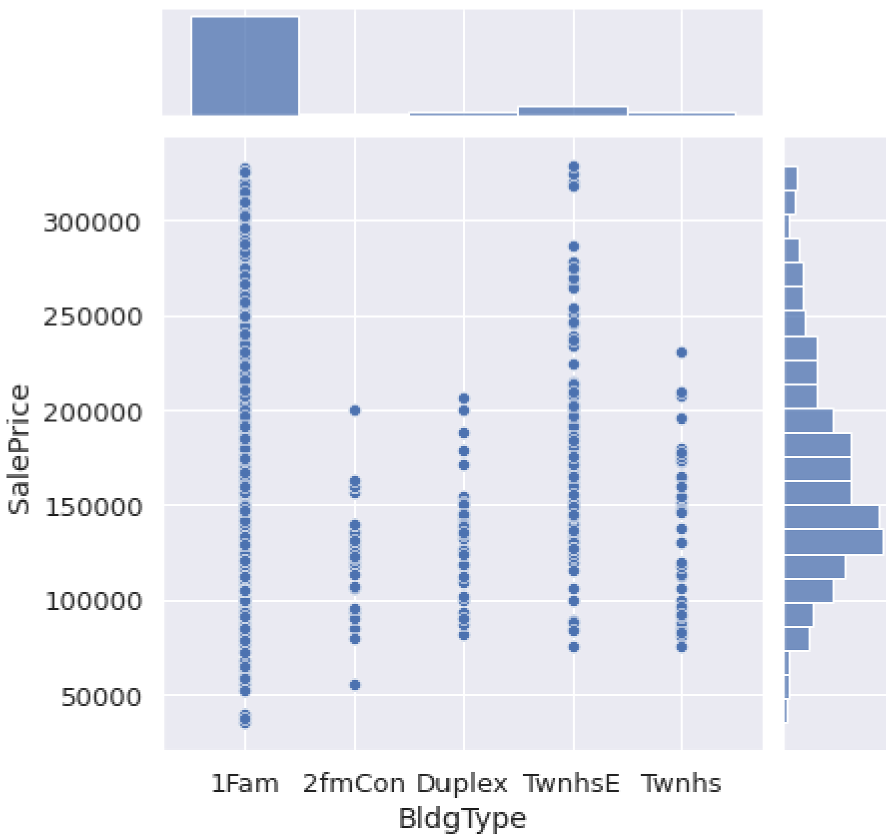
```
# fig, ax = plt.figure(figsize=(20, 10))
sns.jointplot(x= 'SalePrice', y = 'HouseStyle', data = clean_train, height = 10)
```

```
<seaborn.axisgrid.JointGrid at 0x7f5c28040e10>
```



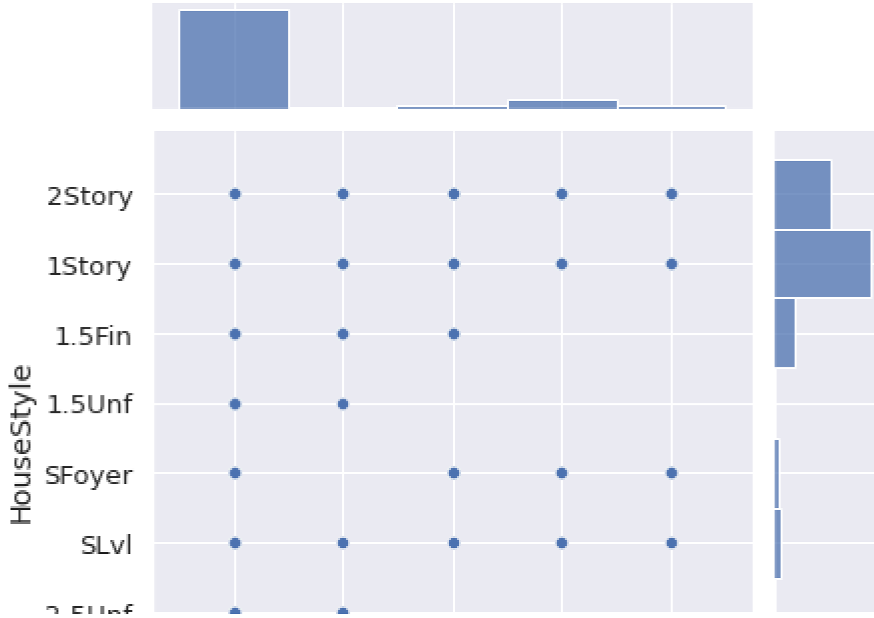

```
sns.jointplot(x= 'BldgType', y = 'SalePrice', data = clean_train)
```

```
<seaborn.axisgrid.JointGrid at 0x7f5c1fcc5ed0>
```




```
sns.jointplot(x= 'BldgType', y = 'HouseStyle', data = clean_train)
```

```
<seaborn.axisgrid.JointGrid at 0x7f5c1fab8490>
```




```
clean_train['Condition1'].value_counts()
```

```
Norm      1147
Feedr       77
Artery      44
RRAn        23
PosN        16
RRAe        11
RRNn         5
PosA         5
RRNe         2
Name: Condition1, dtype: int64
```

```
clean_train['Condition2'].value_counts()
```

```
Norm      1318
Feedr        6
Artery       2
RRNn         2
PosA         1
RRAn         1
Name: Condition2, dtype: int64
```

```
clean_train['BldgType'].value_counts()
```

```
1Fam      1099
TwnhsE     110
Duplex      50
Twnhs       43
2fmCon      28
Name: BldgType, dtype: int64
```

```
clean_train['Utilities'].value_counts()
```

```
AllPub    1329
NoSeWa       1
Name: Utilities, dtype: int64
```

```
clean_train['LotShape'].value_counts()
```

```
Reg    870
IR1    429
IR2     26
IR3      5
Name: LotShape, dtype: int64
```

```
clean_train['LotConfig'].value_counts()
```

```
Inside     970
Corner     242
CulDSac     72
FR2         42
FR3          4
Name: LotConfig, dtype: int64
```

```
clean_train['LandContour'].value_counts()
```

```
Lvl    1220
Bnk      56
HLS      34
Low      20
Name: LandContour, dtype: int64
```

```
clean_train['LandSlope'].value_counts()
```

```
Gtl    1276
Mod      51
Sev       3
Name: LandSlope, dtype: int64
```

```
clean_train['PoolArea'].value_counts()
```

```
0      1327
648       1
576       1
519       1
Name: PoolArea, dtype: int64
```

```
clean_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'LotShape', 'LandContour', 'Utilities', 'LotConfig', 'LandSlope',
       'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
       'OverallQual', 'OverallCond', 'RoofStyle', 'RoofMatl', 'Exterior1st',
```

```
       'Exterior2nd', 'MasVnrType', 'MasVnrArea', 'ExterQual', 'ExterCond',
       'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
       'BsmtFinType2', 'BsmtFinSF2', 'Heating', 'HeatingQC', 'CentralAir',
       'Electrical', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
       'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
       'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd', 'Functional',
       'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice', 'year_diff', 'finish_percentage',
       'Unfinished_percentage'],
      dtype='object')
```

```
clean_train['MSZoning'] = clean_train['MSZoning'].apply(lambda x: x if ((x== 'RL') | (x == 'R
clean_train['Condition1'] = clean_train['Condition1'].apply(lambda x:"Norm" if x== 'Norm' els
clean_train['Condition2'] = clean_train['Condition1'].apply(lambda x:"Norm" if x== 'Norm' els
clean_train['BldgType'] = clean_train['BldgType'].apply(lambda x:"1Fam" if x== '1Fam' else "O
clean_train['LotShape'] = clean_train['LotShape'].apply(lambda x: x if ((x== 'Reg') | (x == '
clean_train['LotConfig'] = clean_train['LotConfig'].apply(lambda x: x if ((x== 'Inside') | (x
clean_train['LandContour'] = clean_train['LandContour'].apply(lambda x:"Lvl" if x== 'Lvl' els
clean_train['LandSlope'] = clean_train['LandSlope'].apply(lambda x:"Gtl" if x== 'Gtl' else "O
clean_train['PoolArea'] = clean_train['PoolArea'].apply(lambda x:0 if x== 0 else 1)
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead


See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarnin


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
clean_train['RoofStyle'].value_counts()
```

```
Gable      1076
Hip         231
Gambrel      11
Mansard       7
Flat          5
Name: RoofStyle, dtype: int64
```

```
clean_train['ExterQual'].value_counts()
```

```
TA    857
Gd    440
Ex     20
Fa     13
Name: ExterQual, dtype: int64
```

```
clean_train['BsmtCond'].value_counts()
```

```
TA    1190
Gd      57
Fa      45
Po       2
Name: BsmtCond, dtype: int64
```

```
clean_train['BsmtQual'].value_counts()
```

```
TA    618
Gd    571
Ex     70
Fa     35
Name: BsmtQual, dtype: int64
```

```
clean_train['SaleType'].value_counts()
```

```
WD       1177
New        86
COD        41
ConLD       8
ConLw       5
ConLI       4
CWD         4
Oth         3
Con         2
Name: SaleType, dtype: int64
```

```
clean_train['RoofStyle'] = clean_train['RoofStyle'].apply(lambda x: x if ((x== 'Gable') | (x
clean_train['ExterQual'] = clean_train['ExterQual'].apply(lambda x: x if ((x== 'TA') | (x ==
clean_train['BsmtCond'] = clean_train['BsmtCond'].apply(lambda x:"TA" if x== 'TA' else "Other
clean_train['BsmtQual'] = clean_train['BsmtQual'].apply(lambda x: x if ((x== 'TA') | (x == 'G
clean_train['SaleType'] = clean_train['SaleType'].apply(lambda x:"WD" if x== 'WD' else "Other
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
clean_train['SaleCondition'].value_counts()
```

```
Normal     1114
Abnorml      95
Partial      88
Family       20
Alloca        9
AdjLand       4
Name: SaleCondition, dtype: int64
```

```
clean_train['FireplaceQu'].value_counts()
```

```
Gd    310
TA    279
Fa     32
Po     19
Ex     12
Name: FireplaceQu, dtype: int64
```

```
clean_train['SaleCondition'] = clean_train['SaleCondition'].apply(lambda x:"Normal" if x== 'N
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
clean_train.drop(['FireplaceQu'], axis=1, inplace = True)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4913: SettingWithCopyWarning
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

```
clean_train.info()
```

```
 18  RoofStyle         1330 non-null   object
 19  RoofMatl          1330 non-null   object
 20  Exterior1st       1330 non-null   object
 21  Exterior2nd       1330 non-null   object
 22  MasVnrType        1324 non-null   object
 23  MasVnrArea        1324 non-null   float64
 24  ExterQual         1330 non-null   object
 25  ExterCond         1330 non-null   object
 26  Foundation        1330 non-null   object
 27  BsmtQual          1330 non-null   object
 28  BsmtCond          1330 non-null   object
 29  BsmtExposure      1293 non-null   object
 30  BsmtFinType1      1294 non-null   object
 31  BsmtFinType2      1293 non-null   object
 32  BsmtFinSF2        1330 non-null   int64
 33  Heating           1330 non-null   object
 34  HeatingQC         1330 non-null   object
 35  CentralAir        1330 non-null   object
 36  Electrical        1329 non-null   object
 37  1stFlrSF          1330 non-null   int64
 38  2ndFlrSF          1330 non-null   int64
 39  LowQualFinSF      1330 non-null   int64
 40  GrLivArea         1330 non-null   int64
 41  BsmtFullBath      1330 non-null   int64
 42  BsmtHalfBath      1330 non-null   int64
 43  FullBath          1330 non-null   int64
 44  HalfBath          1330 non-null   int64
 45  BedroomAbvGr      1330 non-null   int64
 46  KitchenAbvGr      1330 non-null   int64
 47  KitchenQual       1330 non-null   object
 48  TotRmsAbvGrd      1330 non-null   int64
 49  Functional        1330 non-null   object
 50  Fireplaces        1330 non-null   int64
 51  GarageType        1250 non-null   object
 52  GarageYrBlt       1250 non-null   float64
 53  GarageFinish      1250 non-null   object
 54  GarageCars        1330 non-null   int64
 55  GarageArea        1330 non-null   int64
 56  GarageQual        1250 non-null   object
 57  GarageCond        1250 non-null   object
 58  PavedDrive        1330 non-null   object
 59  WoodDeckSF        1330 non-null   int64
 60  OpenPorchSF       1330 non-null   int64
 61  EnclosedPorch     1330 non-null   int64
 62  3SsnPorch         1330 non-null   int64
 63  ScreenPorch       1330 non-null   int64
 64  PoolArea          1330 non-null   int64
 65  MiscVal           1330 non-null   int64
```

```
 65  MiscVal                1330 non-null   int64
 66  MoSold                 1330 non-null   int64
 67  YrSold                 1330 non-null   int64
 68  SaleType               1330 non-null   object
 69  SaleCondition          1330 non-null   object
 70  SalePrice              1330 non-null   int64
 71  year_diff              1330 non-null   int64
 72  finish_percentage      1294 non-null   float64
 73  Unfinished_percentage  1294 non-null   float64
dtypes: float64(5), int64(31), object(38)
memory usage: 811.6+ KB
```

```
clean_train['LotFrontage'].dtypes
```

```
dtype('float64')
```

```
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors = 2)
df_filled = imputer.fit_transform(clean_train[['LotFrontage']])
df_filled
```

```
array([[65.],
       [80.],
       [68.],
       ...,
       [66.],
       [68.],
       [75.]])
```

```
df_filled.shape
```

```
(1330, 1)
```

```
df_filled= pd.DataFrame(df_filled)
```

```
df_filled.head()
```

	0
0	65.0
1	80.0
2	68.0
3	60.0
4	84.0

```
df_filled[0]
```

```
0       65.0
1       80.0
2       68.0
3       60.0
4       84.0
        ...
1325    62.0
1326    85.0
1327    66.0
1328    68.0
1329    75.0
Name: 0, Length: 1330, dtype: float64
```

```
# entry = df_filled[0]
combined_clean_train = pd.concat([clean_train, df_filled], axis=1)
combined_clean_train.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utili
0	1.0	60.0	RL	65.0	8450.0	Pave	Reg	Lvl	A
1	2.0	20.0	RL	80.0	9600.0	Pave	Reg	Lvl	A
2	3.0	60.0	RL	68.0	11250.0	Pave	IR1	Lvl	A
3	4.0	70.0	RL	60.0	9550.0	Pave	IR1	Lvl	A
4	5.0	60.0	RL	84.0	14260.0	Pave	IR1	Lvl	A

```
## now we will rename 0 column into LotFrontage and drop old one
combined_clean_train.drop(['LotFrontage'], axis=1, inplace = True)
```

```
combined_clean_train.rename(columns={0: 'LotFrontage'}, inplace = True)
```

```
combined_clean_train['LotFrontage'].isna().sum()
```

```
118
```

```
combined_clean_train.dropna(inplace=True)
```

```
combined_clean_train['LotFrontage'].isna().sum()
```

```
0
```

```
combined_clean_train.shape
```

```
(1102, 74)
```

```
combined_clean_train.info()
```

```
 18  RoofMatl          1102 non-null   object
 19  Exterior1st       1102 non-null   object
 20  Exterior2nd       1102 non-null   object
 21  MasVnrType        1102 non-null   object
 22  MasVnrArea        1102 non-null   float64
 23  ExterQual         1102 non-null   object
 24  ExterCond         1102 non-null   object
 25  Foundation        1102 non-null   object
 26  BsmtQual          1102 non-null   object
 27  BsmtCond          1102 non-null   object
 28  BsmtExposure      1102 non-null   object
 29  BsmtFinType1      1102 non-null   object
 30  BsmtFinType2      1102 non-null   object
 31  BsmtFinSF2        1102 non-null   float64
 32  Heating           1102 non-null   object
 33  HeatingQC         1102 non-null   object
 34  CentralAir        1102 non-null   object
 35  Electrical        1102 non-null   object
 36  1stFlrSF          1102 non-null   float64
 37  2ndFlrSF          1102 non-null   float64
 38  LowQualFinSF      1102 non-null   float64
 39  GrLivArea         1102 non-null   float64
 40  BsmtFullBath      1102 non-null   float64
 41  BsmtHalfBath      1102 non-null   float64
 42  FullBath          1102 non-null   float64
 43  HalfBath          1102 non-null   float64
 44  BedroomAbvGr      1102 non-null   float64
 45  KitchenAbvGr      1102 non-null   float64
 46  KitchenQual       1102 non-null   object
 47  TotRmsAbvGrd      1102 non-null   float64
 48  Functional        1102 non-null   object
 49  Fireplaces        1102 non-null   float64
 50  GarageType        1102 non-null   object
 51  GarageYrBlt       1102 non-null   float64
 52  GarageFinish      1102 non-null   object
 53  GarageCars        1102 non-null   float64
 54  GarageArea        1102 non-null   float64
 55  GarageQual        1102 non-null   object
 56  GarageCond        1102 non-null   object
 57  PavedDrive        1102 non-null   object
 58  WoodDeckSF        1102 non-null   float64
 59  OpenPorchSF       1102 non-null   float64
 60  EnclosedPorch     1102 non-null   float64
 61  3SsnPorch         1102 non-null   float64
 62  ScreenPorch       1102 non-null   float64
 63  PoolArea          1102 non-null   float64
 64  MiscVal           1102 non-null   float64
 65  MoSold            1102 non-null   float64
 66  YrSold            1102 non-null   float64
 67  SaleType          1102 non-null   object
 68  SaleCondition     1102 non-null   object
 69  SalePrice         1102 non-null   float64
```

```
 69  SalePrice             1102 non-null   float64
 70  year_diff             1102 non-null   float64
 71  finish_percentage     1102 non-null   float64
 72  Unfinished_percentage 1102 non-null   float64
 73  LotFrontage           1102 non-null   float64
dtypes: float64(36), object(38)
memory usage: 645.7+ KB
```

```
combined_clean_train["Condition_all"]= combined_clean_train[["Condition1","Condition2"]].appl
combined_clean_train["Condition_all"]
```

```
0        Norm
1       Other
2        Norm
3        Norm
4        Norm
        ...
1322     Norm
1324     Norm
1327     Norm
1328     Norm
1329     Norm
Name: Condition_all, Length: 1102, dtype: object
```

```
combined_clean_train[(combined_clean_train.Condition1 == 'Norm') & (combined_clean_train.Cond
```

Id	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConfig

```
combined_clean_train[(combined_clean_train.Condition1 != 'Norm') & (combined_clean_train.Cond
```

Id	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConfig

```
combined_clean_train["Condition_all"].value_counts()
```

```
Norm     955
Other    147
Name: Condition_all, dtype: int64
```

```
combined_clean_train.drop(['Condition1', 'Condition2'], axis=1, inplace = True)
combined_clean_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotArea', 'Street', 'LotShape',
       'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
```

```
       'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'BsmtFinSF2', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice', 'year_diff', 'finish_percentage',
       'Unfinished_percentage', 'LotFrontage', 'Condition_all'],
      dtype='object')
```

```
combined_clean_train.shape
```

```
(1102, 73)
```

What we are doing is trying to minimize categorical values for each column and replace values then we will apply get_dummies on dataset and then apply train test split and see what's coming after that

We won't use imblearn as we are predicting price not categorical value we won't also stratify y as we said we are not predicting categorical values as y is SalePrice

```
combined_clean_train["LandContour"].value_counts()
```

```
Lvl      1019
Other      83
Name: LandContour, dtype: int64
```

```
combined_clean_train["Utilities"].value_counts()
```

```
AllPub    1101
NoSeWa       1
Name: Utilities, dtype: int64
```

```
combined_clean_train["LotConfig"].value_counts()
```

```
Inside    792
Other     310
Name: LotConfig, dtype: int64
```

```
combined_clean_train["LotShape"].value_counts()
```

```
Reg      697
IR1      377
```

```
Other     28
Name: LotShape, dtype: int64
```

```
combined_clean_train["Street"].value_counts()
```

```
Pave    1100
Grvl       2
Name: Street, dtype: int64
```

```
combined_clean_train["LandSlope"].value_counts()
```

```
Gtl      1058
Other      44
Name: LandSlope, dtype: int64
```

```
combined_clean_train["RoofStyle"].value_counts()
```

```
Gable    881
Hip      203
Other     18
Name: RoofStyle, dtype: int64
```

```
combined_clean_train["RoofMatl"].value_counts()
```

```
CompShg    1092
WdShake       4
Tar&Grv       3
WdShngl       2
Roll          1
Name: RoofMatl, dtype: int64
```

```
combined_clean_train["Exterior1st"].value_counts()
```

```
VinylSd    396
HdBoard    186
MetalSd    173
Wd Sdng    153
Plywood     78
BrkFace     35
CemntBd     33
Stucco      15
WdShing     15
AsbShng     14
Stone        2
BrkComm      1
ImStucc      1
Name: Exterior1st, dtype: int64
```

```
combined_clean_train["Exterior2nd"].value_counts()
```

```
VinylSd    387
HdBoard    172
MetalSd    171
Wd Sdng    148
Plywood     99
CmentBd     32
Wd Shng     25
BrkFace     19
Stucco      17
AsbShng     14
ImStucc      8
Brk Cmn      6
AsphShn      2
Other        1
Stone        1
Name: Exterior2nd, dtype: int64
```

```
combined_clean_train["MasVnrType"].value_counts()
```

```
None       646
BrkFace    358
Stone       86
BrkCmn      12
Name: MasVnrType, dtype: int64
```

```
combined_clean_train["BldgType"].value_counts()
```

```
1Fam     930
Other    172
Name: BldgType, dtype: int64
```

```
combined_clean_train["HouseStyle"].value_counts()
```

```
1Story    550
2Story    341
1.5Fin    108
SLvl       54
SFoyer     26
1.5Unf     10
2.5Unf     10
2.5Fin      3
Name: HouseStyle, dtype: int64
```

```
combined_clean_train["ExterQual"].value_counts()
```

```
TA       688
Gd       391
Other     23
Name: ExterQual, dtype: int64
```

```
combined_clean_train["ExterCond"].value_counts()
```

```
TA    974
Gd    111
Fa     15
Ex      2
Name: ExterCond, dtype: int64
```

```
combined_clean_train["Foundation"].value_counts()
```

```
PConc     495
CBlock    495
BrkTil    104
Stone       5
Wood        3
Name: Foundation, dtype: int64
```

```
combined_clean_train["BsmtQual"].value_counts()
```

```
TA       508
Gd       502
Other     92
Name: BsmtQual, dtype: int64
```

```
combined_clean_train["BsmtCond"].value_counts()
```

```
TA       1018
Other      84
Name: BsmtCond, dtype: int64
```

```
combined_clean_train["BsmtExposure"].value_counts()
```

```
No    774
Av    168
Mn     89
Gd     71
Name: BsmtExposure, dtype: int64
```

```
combined_clean_train["BsmtFinType1"].value_counts()
```

```
Unf    329
GLQ    301
ALQ    183
BLQ    120
Rec    109
LwQ     60
Name: BsmtFinType1, dtype: int64
```

```
combined_clean_train["BsmtFinType2"].value_counts()
```

```
Unf    968
LwQ     42
```

```
Rec      37
BLQ      29
ALQ      16
GLQ      10
Name: BsmtFinType2, dtype: int64
```

```
combined_clean_train["Heating"].value_counts()
```

```
GasA    1086
GasW      13
Grav       2
OthW       1
Name: Heating, dtype: int64
```

```
combined_clean_train["HeatingQC"].value_counts()
```

```
Ex    556
TA    324
Gd    191
Fa     30
Po      1
Name: HeatingQC, dtype: int64
```

```
combined_clean_train["CentralAir"].value_counts()
```

```
Y    1049
N      53
Name: CentralAir, dtype: int64
```

```
combined_clean_train["Electrical"].value_counts()
```

```
SBrkr    1015
FuseA      69
FuseF      15
FuseP       2
Mix         1
Name: Electrical, dtype: int64
```

```
combined_clean_train["KitchenQual"].value_counts()
```

```
TA    558
Gd    474
Ex     50
Fa     20
Name: KitchenQual, dtype: int64
```

```
combined_clean_train["Functional"].value_counts()
```

```
Typ     1034
Min2      26
```

```
Min1      20
Maj1      10
Mod        8
Maj2       4
Name: Functional, dtype: int64
```

```
combined_clean_train["GarageType"].value_counts()
```

```
Attchd     699
Detchd     320
BuiltIn     57
Basment     16
CarPort      6
2Types       4
Name: GarageType, dtype: int64
```

```
# def Ext_condition(x, y):
#   if x == y

# BsmtFinType2, BsmtFinType1
# combined_clean_train["Exterior_Cond"]= combined_clean_train[["Exterior1st","Exterior2nd"]].
```

```
combined_clean_train["GarageFinish"].value_counts()
```

```
Unf    505
RFn    349
Fin    248
Name: GarageFinish, dtype: int64
```

```
combined_clean_train["GarageQual"].value_counts()
```

```
TA    1047
Fa      39
Gd      11
Po       3
Ex       2
Name: GarageQual, dtype: int64
```

```
combined_clean_train["GarageCond"].value_counts()
```

```
TA    1058
Fa      28
Gd       8
Po       6
Ex       2
Name: GarageCond, dtype: int64
```

```
combined_clean_train["PavedDrive"].value_counts()
```

```
Y    1040
```

```
N      41
P      21
Name: PavedDrive, dtype: int64
```

```
combined_clean_train["SaleType"].value_counts()
```

```
WD       972
Other    130
Name: SaleType, dtype: int64
```

```
combined_clean_train["SaleCondition"].value_counts()
```

```
Normal    926
Other     176
Name: SaleCondition, dtype: int64
```

```
combined_clean_train["RoofMatl"] = combined_clean_train["RoofMatl"].apply(lambda x:"CompShg"
combined_clean_train["Exterior1st"] = combined_clean_train["Exterior1st"].apply(lambda x: x i
combined_clean_train["Exterior2nd"] = combined_clean_train["Exterior2nd"].apply(lambda x: x i
combined_clean_train["MasVnrType"] = combined_clean_train["MasVnrType"].apply(lambda x: x if
combined_clean_train["HouseStyle"] = combined_clean_train["HouseStyle"].apply(lambda x: x if
combined_clean_train["ExterCond"] = combined_clean_train["ExterCond"].apply(lambda x: x if ((
combined_clean_train["Foundation"] = combined_clean_train["ExterCond"].apply(lambda x: x if (
combined_clean_train["Bsmt Exposure"] = combined_clean_train["BsmtExposure"].apply(lambda x:
combined_clean_train["Heating"] = combined_clean_train["Heating"].apply(lambda x: x if ((x==
combined_clean_train["HeatingQC"] = combined_clean_train["HeatingQC"].apply(lambda x: x if ((
combined_clean_train["Electrical"] = combined_clean_train["Electrical"].apply(lambda x: x if
combined_clean_train["KitchenQual"] = combined_clean_train["KitchenQual"].apply(lambda x: x i
combined_clean_train["Functional"] = combined_clean_train["Functional"].apply(lambda x: x if
combined_clean_train["GarageType"] = combined_clean_train["GarageType"].apply(lambda x: x if
combined_clean_train["GarageQual"] = combined_clean_train["GarageQual"].apply(lambda x: x if
combined_clean_train["GarageCond"] = combined_clean_train["GarageCond"].apply(lambda x: x if
```

```
combined_clean_train["Neighborhood"].value_counts()
```

```
NAmes      183
CollgCr    126
OldTown     87
Somerst     70
Gilbert     65
Sawyer      64
NWAmes      61
Edwards     58
SawyerW     50
NridgHt     46
BrkSide     40
Crawfor     38
Mitchel     36
Timber      27
IDOTRR      26
NoRidge     24
```

```
SWISU       17
StoneBr     15
Blmngtn     15
ClearCr     13
BrDale      13
Veenker     10
MeadowV     10
NPkVill      7
Blueste      1
Name: Neighborhood, dtype: int64
```

```
combined_clean_train.head()
```

	Id	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	LotConf
0	1.0	60.0	RL	8450.0	Pave	Reg	Lvl	AllPub	Insi
1	2.0	20.0	RL	9600.0	Pave	Reg	Lvl	AllPub	Oth
2	3.0	60.0	RL	11250.0	Pave	IR1	Lvl	AllPub	Insi
3	4.0	70.0	RL	9550.0	Pave	IR1	Lvl	AllPub	Oth
4	5.0	60.0	RL	14260.0	Pave	IR1	Lvl	AllPub	Oth

We also need to rename columns to more identifying name

```
combined_clean_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotArea', 'Street', 'LotShape',
       'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
       'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'RoofStyle',
       'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'MasVnrArea',
       'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
       'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'BsmtFinSF2', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'GarageType', 'GarageYrBlt',
       'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond',
       'PavedDrive', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice', 'year_diff', 'finish_percentage',
       'Unfinished_percentage', 'LotFrontage', 'Condition_all',
       'Bsmt Exposure'],
      dtype='object')
```

```
combined_clean_train.rename({'MSSubClass':'dwelling_involved_type',
                             'MSZoning':'general_zoning_classification',
                             'Street':'type_of_road',
                             'LotShape':'property_general_shape',
                             'LandContour':'property_Flatness',
                             'Utilities':'utilities_types',
                             'BldgType':'dwelling_type',
                             'RoofMatl':'roof_material',
                             'Exterior1st':'exterior_covering_1',
                             'Exterior2nd':'exterior_covering_2',
                             'MasVnrType':'masonry_veneer_type',
                             'Electrical':'electrical_system',
                             'TotRmsAbvGrd':'total_rooms_above_grade',
                             'GarageFinish':'interior_finish_garage',
                             'GarageCars':'garage_car_capacity',
                             '3SsnPorch':'three_season_porch_area',
                             'MiscFeature':'other_features',
                             'MiscVal':'other_features_values'
                             }, axis='columns', inplace = True)
```

```
combined_clean_train.head()
```

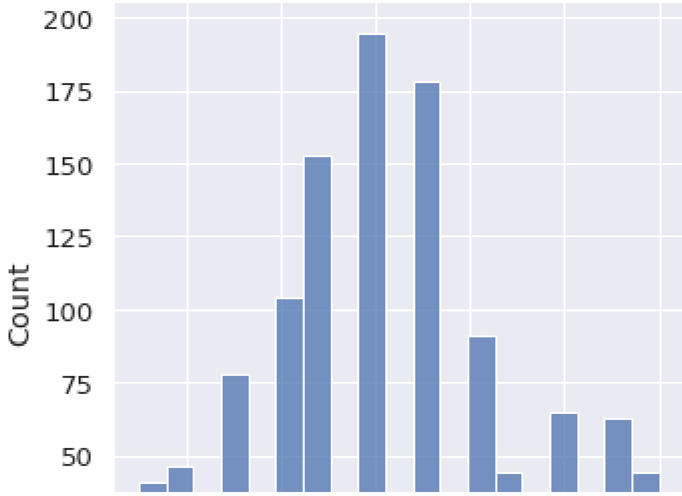
	Id	dwelling_involved_type	general_zoning_classification	LotArea	type_of_road	pr
0	1.0	60.0	RL	8450.0	Pave	
1	2.0	20.0	RL	9600.0	Pave	
2	3.0	60.0	RL	11250.0	Pave	
3	4.0	70.0	RL	9550.0	Pave	
4	5.0	60.0	RL	14260.0	Pave	

```
combined_clean_train.drop(['Id'], axis=1, inplace = True)
```

```
combined_clean_train.to_csv('final_mod_house_price.csv', encoding = "UTF-8")
```

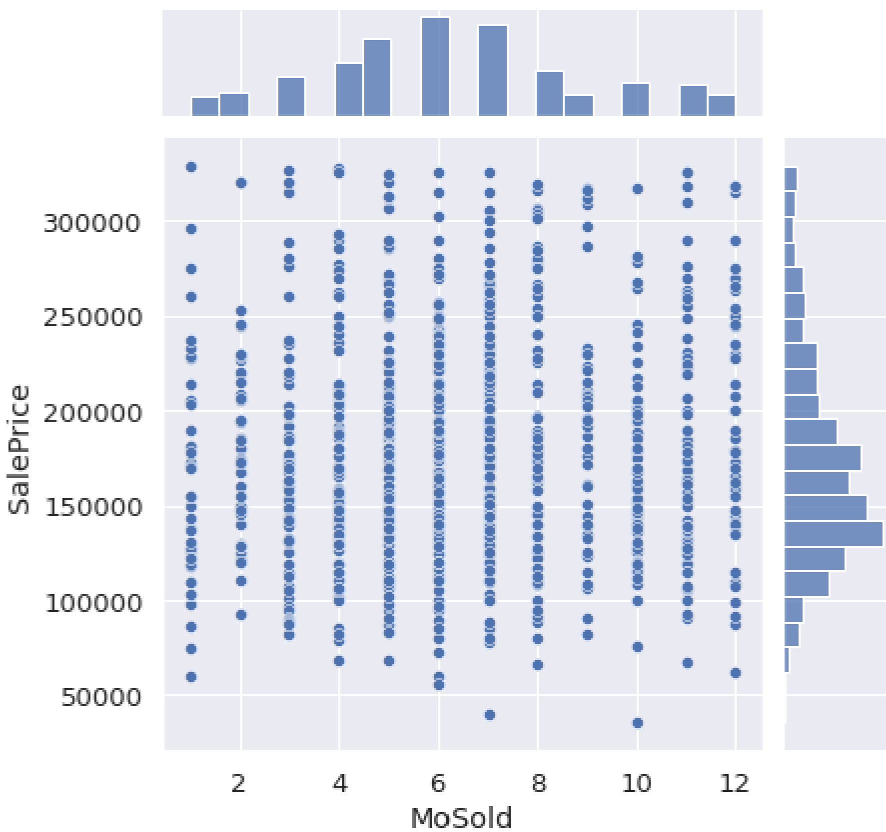
```
sns.displot(x = 'MoSold', data = combined_clean_train)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5c1ea127d0>
```



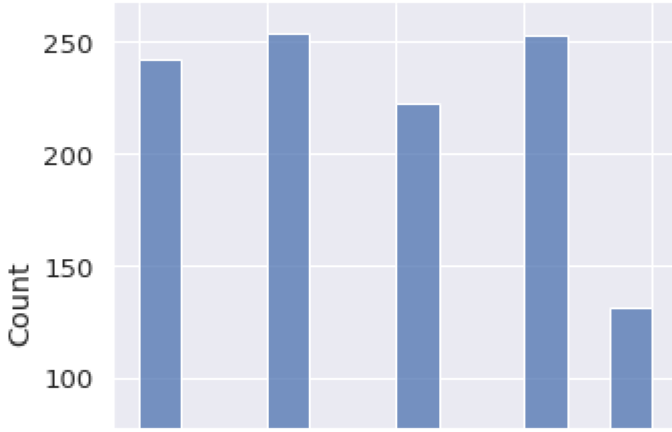

```
sns.jointplot(x= 'MoSold', y = 'SalePrice', data = combined_clean_train)
```

```
<seaborn.axisgrid.JointGrid at 0x7f5c1e92b9d0>
```



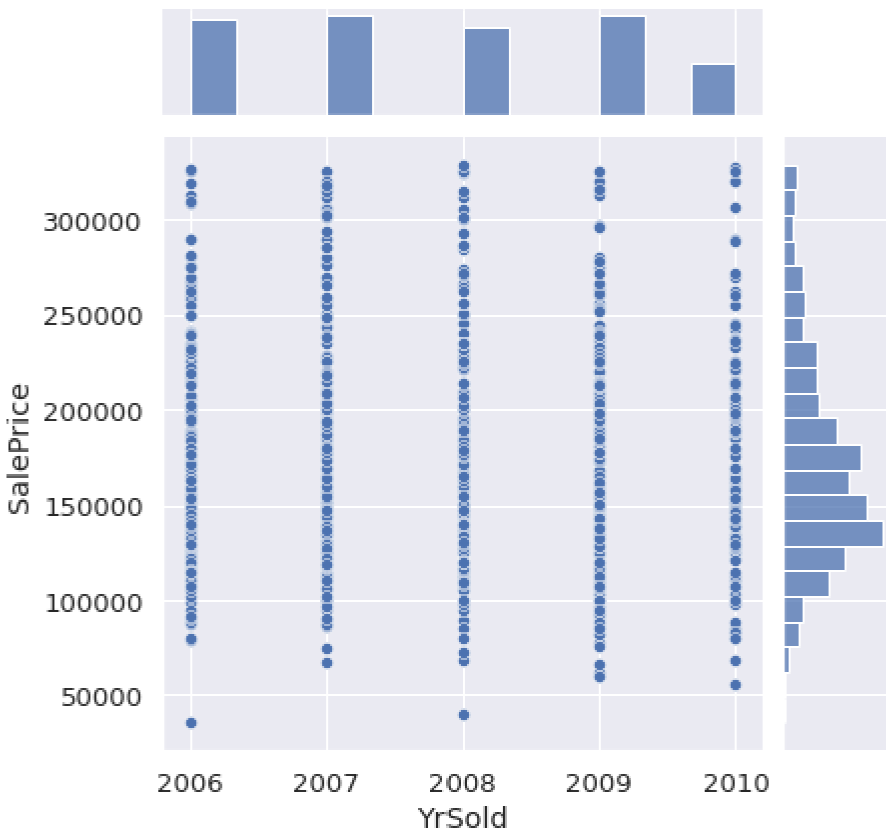

```
sns.displot(x = 'YrSold', data = combined_clean_train)
```

<seaborn.axisgrid.FacetGrid at 0x7f5c1ea12450>




```
sns.jointplot(x= 'YrSold', y = 'SalePrice', data = combined_clean_train)
```

<seaborn.axisgrid.JointGrid at 0x7f5c1e72dbd0>



What are we getting from this Plots?!

- Which year/month sold with most profits
- Which year/month sold most in count
- Does it really mean year that selling less isn't reaching high prices?
- In 2010 Was least year to sell houses but yet we could found that we could still sell with high prices