HOJA DE TRABAJO 3 ARBOLES DE DECISION

Raul Jimenez 19017 Donaldo Garcia 19683 Oscar Saravia 19322 link al repo:
https://github.com/raulangelj/HT3_ARBOLES_DE_DECISION

In [ ]:
```
# from re import U
from statsmodels.graphics.gofplots import qqplot
import numpy as np
import pandas as pd
# import pandasql as ps
import matplotlib.pyplot as plt
# import scipy.stats as stats
import statsmodels.stats.diagnostic as diag
# import statsmodels.api as sm
import seaborn as sns
# import random
import sklearn.cluster as cluster
# import sklearn.metrics as metrics
import sklearn.preprocessing
# import scipy.cluster.hierarchy as sch
import pyclustertend
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
# import sklearn.mixture as mixture
# from sklearn import datasets
# from sklearn.cluster import DBSCAN
# from numpy import unique
# from numpy import where
# from matplotlib import pyplot
# from sklearn.datasets import make_classification
# from sklearn.cluster import Birch
# from sklearn.mixture import GaussianMixture

from sklearn.model_selection import train_test_split

# %matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

1. Descargue los conjuntos de datos de la plataforma kaggle.

In [ ]:
```
train = pd.read_csv('./train.csv', encoding='latin1')
train.head()
```

Out[ ]:

	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

2. Haga un análisis exploratorio extenso de los datos. Explique bien todos los hallazgos. No ponga solo gráficas y código. Debe llegar a conclusiones interesantes para poder predecir.Explique el preprocesamiento que necesitó hacer.

Se deciden utilizar estas variables debido a que estas son las que nos permiten predecir el comportamiento de este mercad o en un futoro. Con estas variables podemos ver si tiene alguna importanacia en el precio la cantidad del espacio, cantidad de cuartos/baños e incluso el año en el que se termina vendiendo la casa

- SalePrice - **CUANTITATIVO CONTINUO** debido a que el precio puede tener centavos; the property's sale price in dollars. This is the target variable that you're trying to predict.
- LotArea: **CUANTITATIVO CONTINUO** Lot size in square feet
- OverallCond: **CUANTITATIVO DISCRETO** Overall condition rating
- YearBuilt: **CUANTITATIVO DISCRETO** Original construction date
- MasVnrArea: **CUANTITATIVO CONTINUO** Masonry veneer area in square feet
- TotalBsmtSF: **CUANTITATIVO CONTINUO** Total square feet of basement area
- 1stFlrSF: **CUANTITATIVO CONTINUO** First Floor square feet
- 2ndFlrSF: **CUANTITATIVO CONTINUO** Second floor square feet
- GrLivArea: **CUANTITATIVO CONTINUO** Above grade (ground) living area square feet
- TotRmsAbvGrd: **CUANTITATIVO DISCRETO** Total rooms above grade (does not include bathrooms)
- GarageCars: **CUANTITATIVO DISCRETO** Size of garage in car capacity
- WoodDeckSF: **CUANTITATIVO CONTINUO** Wood deck area in square feet
- OpenPorchSF: **CUANTITATIVO CONTINUO** Open porch area in square feet
- EnclosedPorch: **CUANTITATIVO CONTINUO** Enclosed porch area in square feet
- PoolArea: **CUANTITATIVO CONTINUO** Pool area in square feet
- Neighborhood: **CUALITATIVO NOMINAL** Physical locations within Ames city limits

In [ ]:
```
usefullAttr = ['SalePrice', 'LotArea', 'OverallCond', 'YearBuilt', 'MasVnrArea', 'Tota
```

```
'2ndFlrSF', 'GrLivArea', 'TotRmsAbvGrd', 'GarageCars', 'WoodDeckSF', 'C
```

In [ ]:
```
data = train[usefullAttr]
data.head()
```

Out[ ]:

	SalePrice	LotArea	OverallCond	YearBuilt	MasVnrArea	TotalBsmtSF	1stFlrSF	2ndFlrSF	GrLivAre
0	208500	8450	5	2003	196.0	856	856	854	171
1	181500	9600	8	1976	0.0	1262	1262	0	126
2	223500	11250	5	2001	162.0	920	920	866	178
3	140000	9550	5	1915	0.0	756	961	756	171
4	250000	14260	5	2000	350.0	1145	1145	1053	219

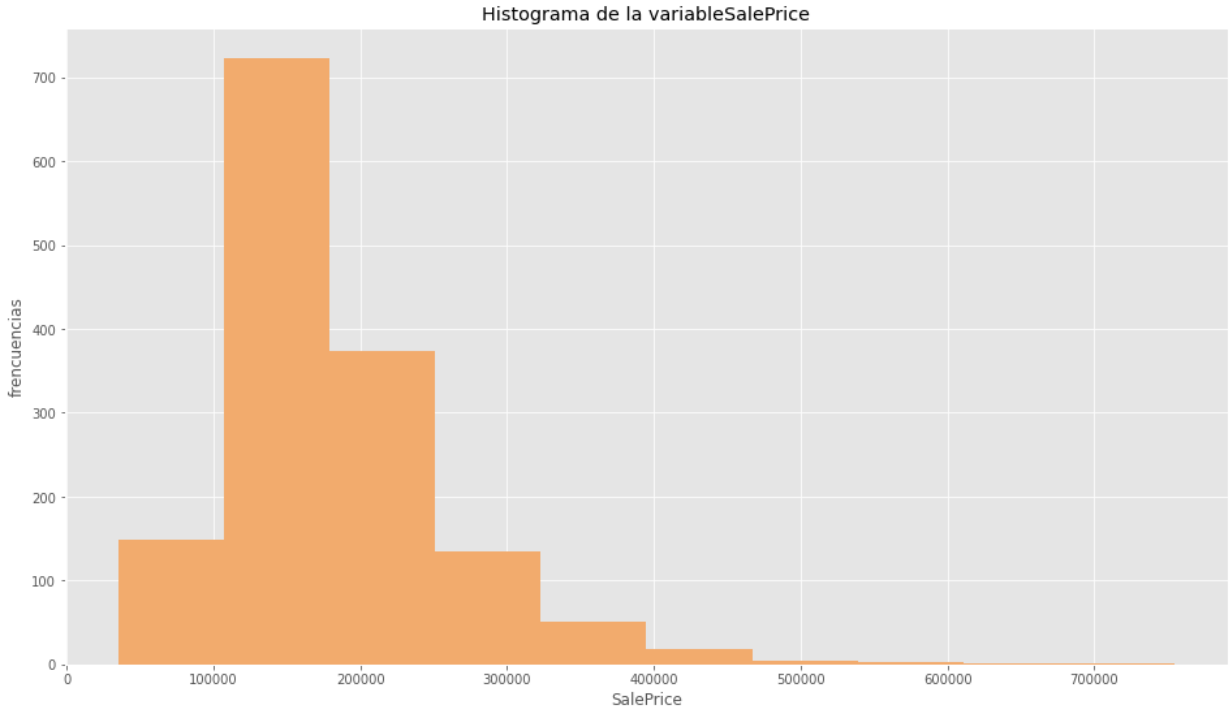
GRAFICAS DE VARIABLES

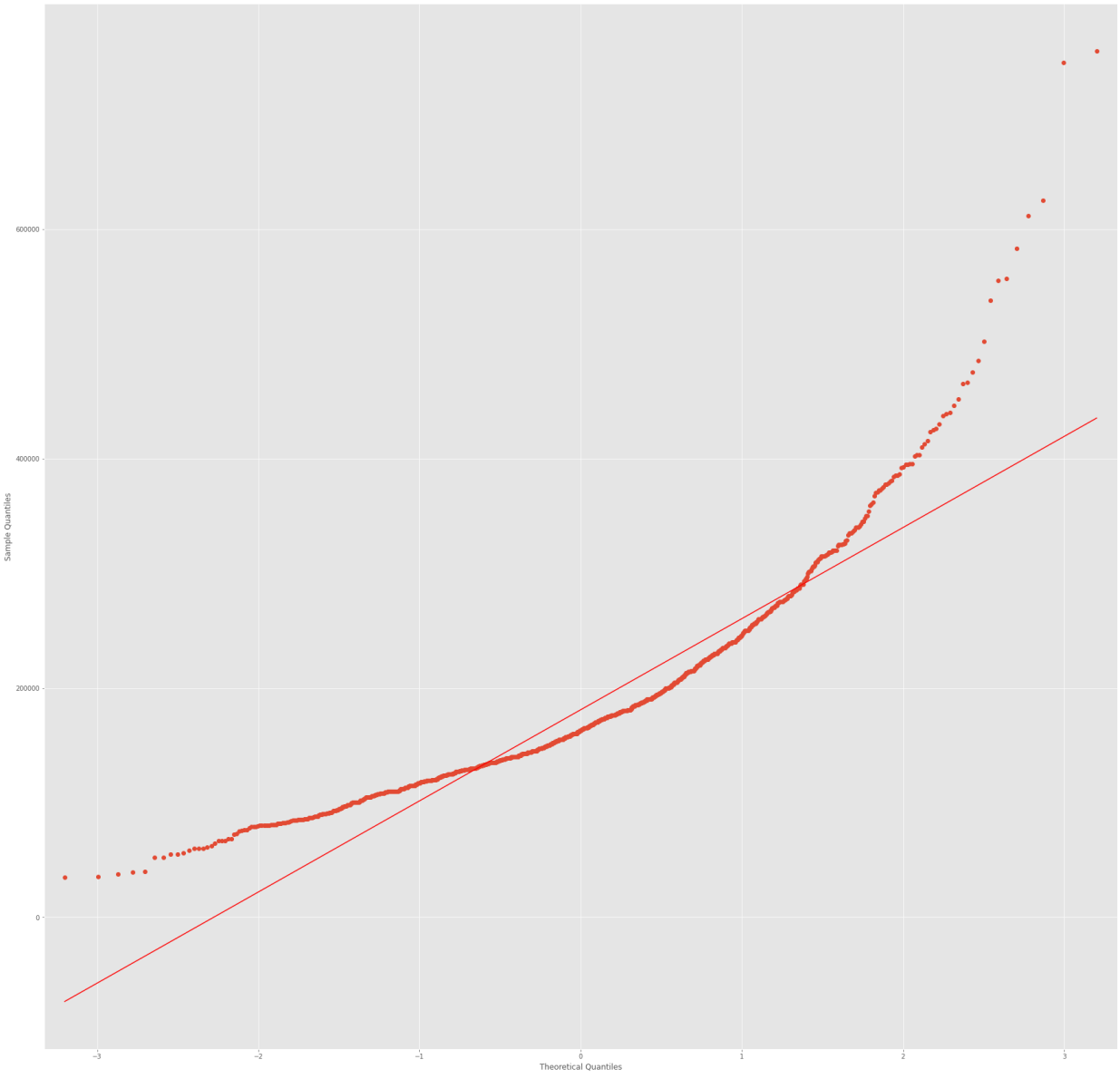
In [ ]:
```
def get_histogram_qq(variable):
    plt.hist(x=data[variable] .dropna(), color='#F2AB6D', rwidth=1)
    plt.title(f'Histograma de la variable{variable}')
    plt.xlabel(variable)
    plt.ylabel('frencuencias')
    plt.rcParams['figure.figsize'] = (30, 30)
    plt.show()

    distribucion_generada = data[variable].dropna()
    # Represento el Q-Q plot
    qqplot(distribucion_generada, line='s')
    plt.show()
```

SalePricee

Se puede determinar que la variable SalePrice no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('SalePrice')
```

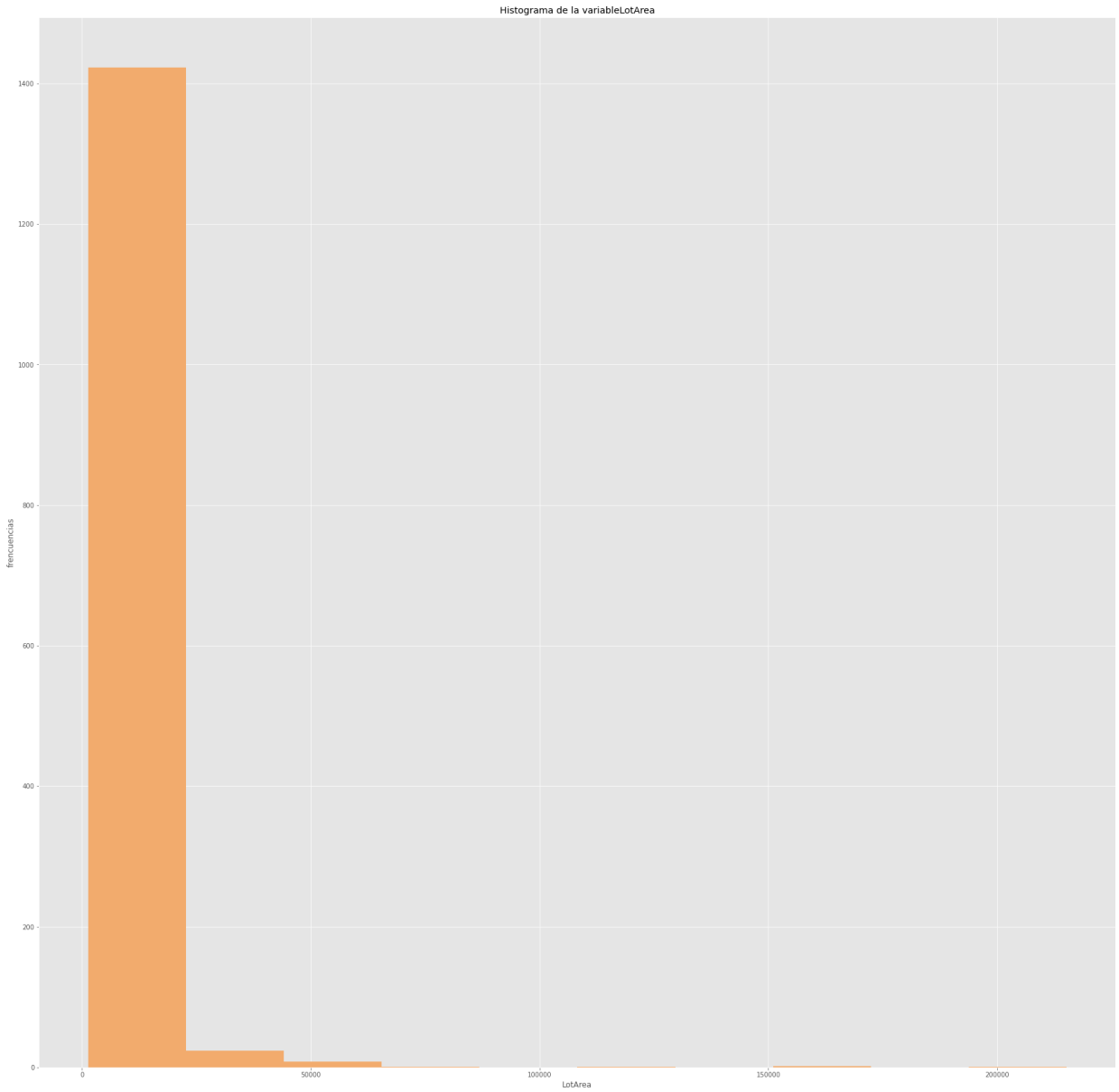

Histograma de la variableSalePrice
frencuencias
SalePrice
0
100
200
300
400
500
600
700
0
100000
200000
300000
400000
500000
600000
700000

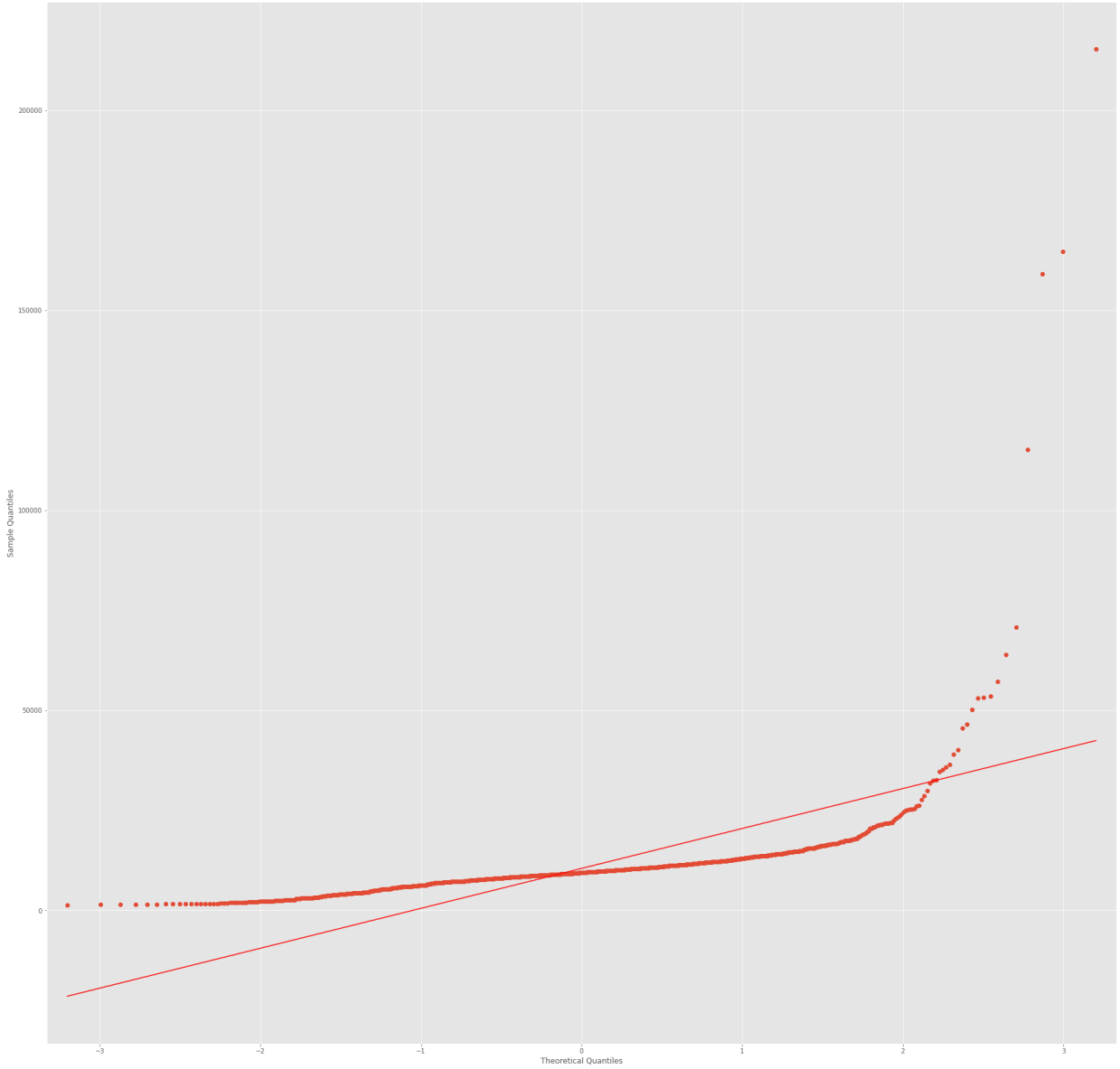

Sample Quantiles
Theoretical Quantiles
0
200000
400000
600000
−3
−2
−1
0
1
2
3

LotArea

Se puede determinar que la variable LotArea no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('LotArea')
```

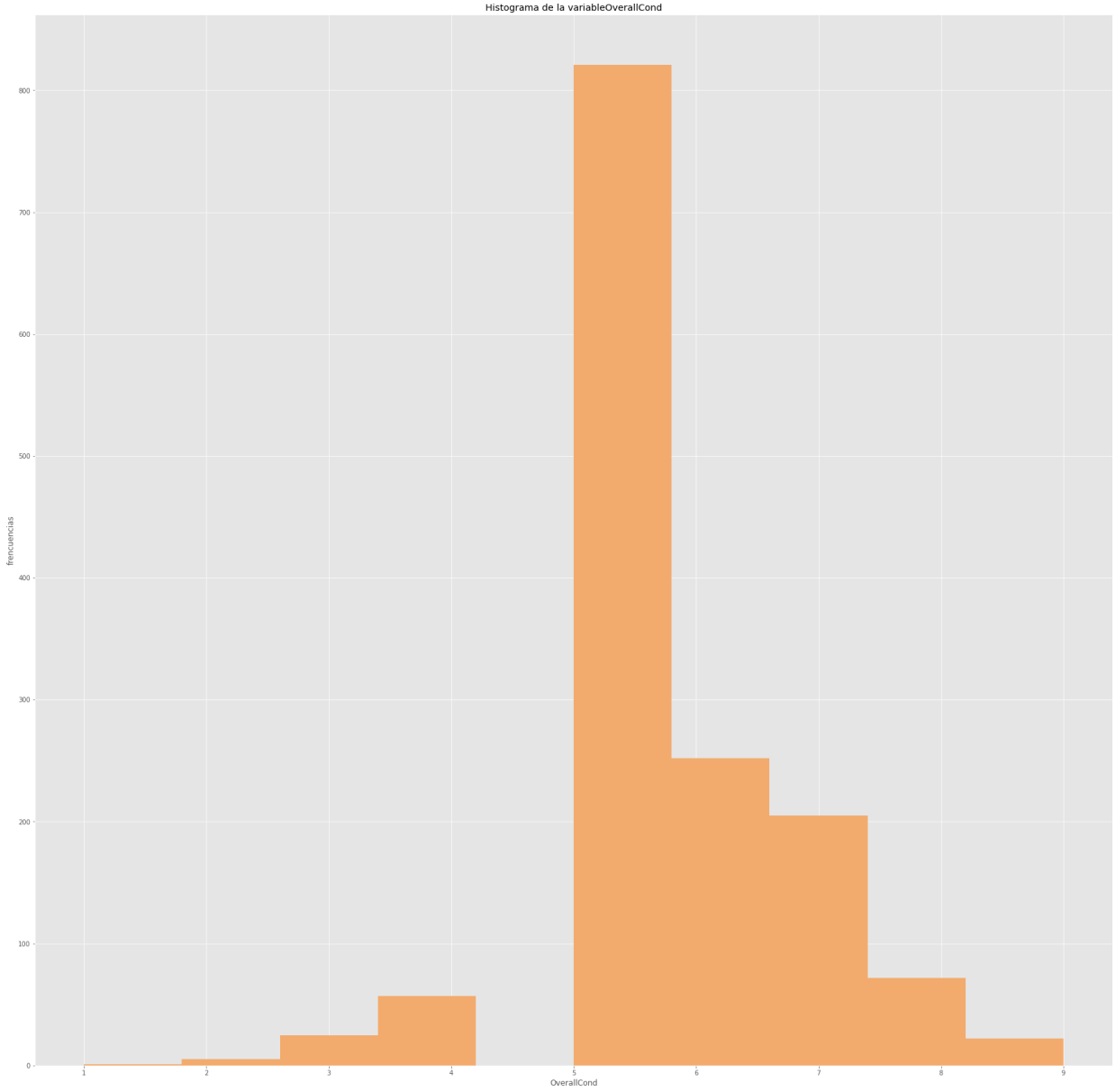


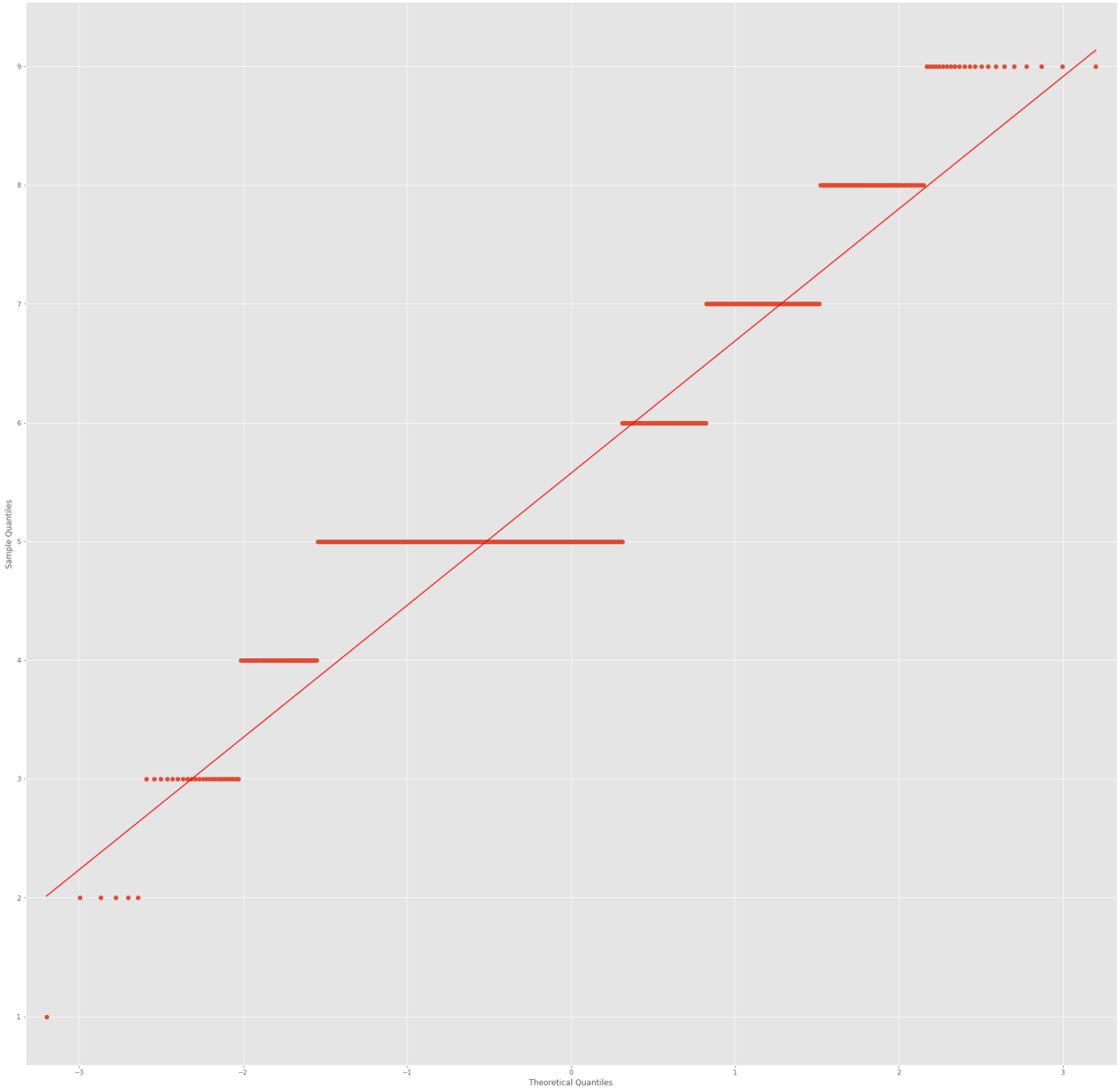


OverallCond

Se puede determinar que la variable OverallCond no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('OverallCond')
```

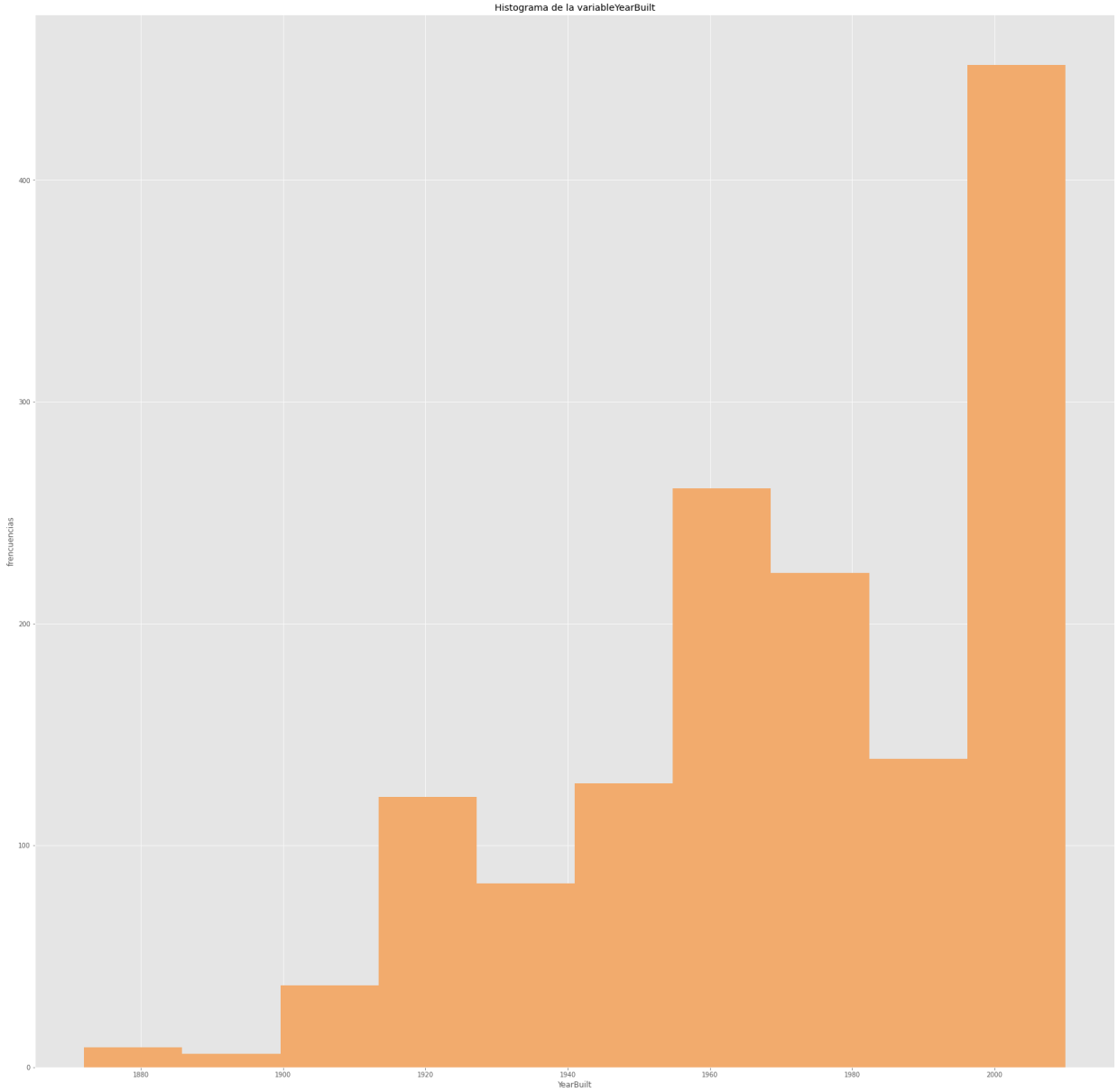

Histograma de la variableOverallCond
frencuencias
OverallCond
0
100
200
300
400
500
600
700
800
1
2
3
4
5
6
7
8
9

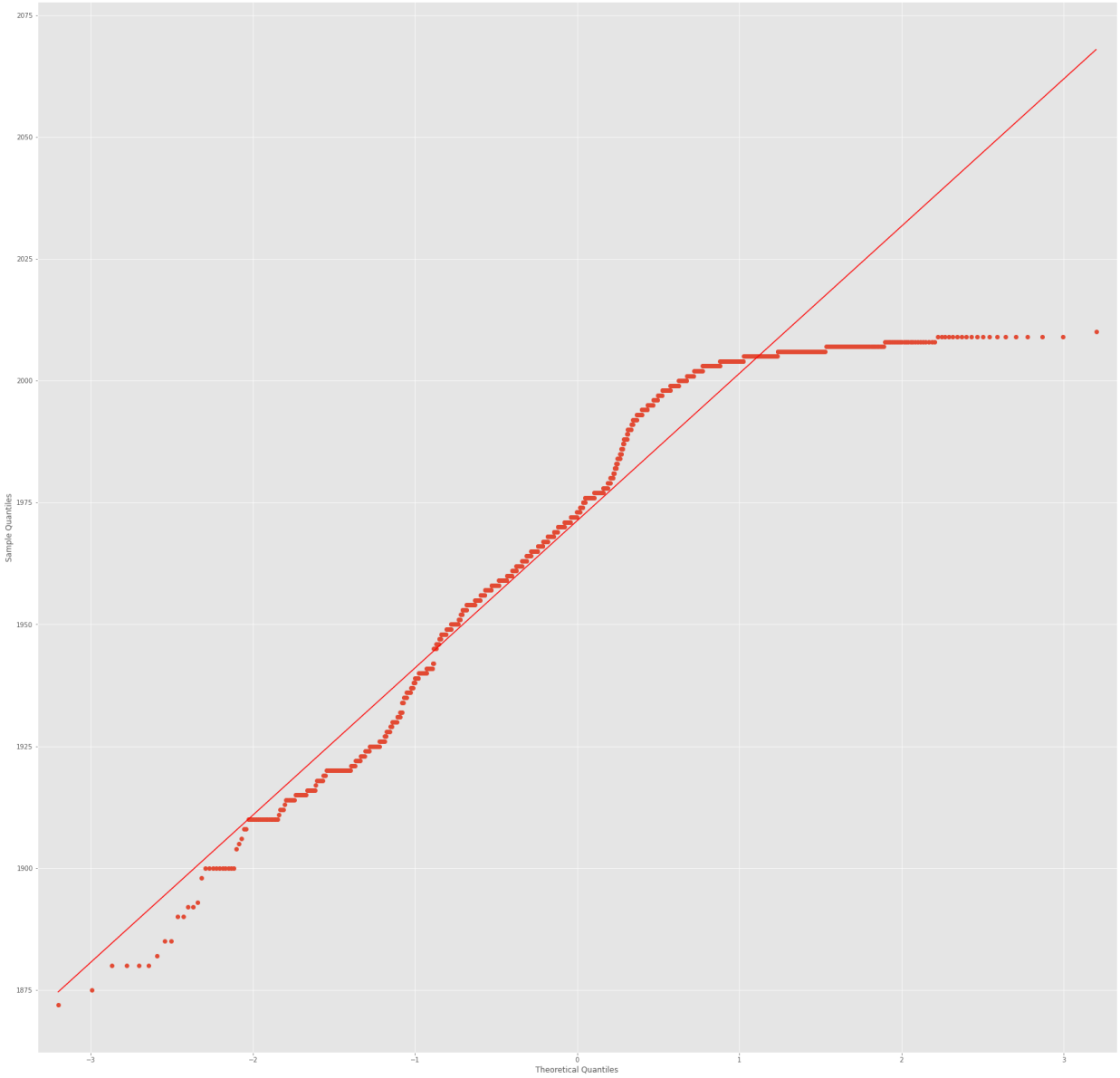


YearBuilt

Se puede determinar que la variable YearBuilt no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('YearBuilt')
```

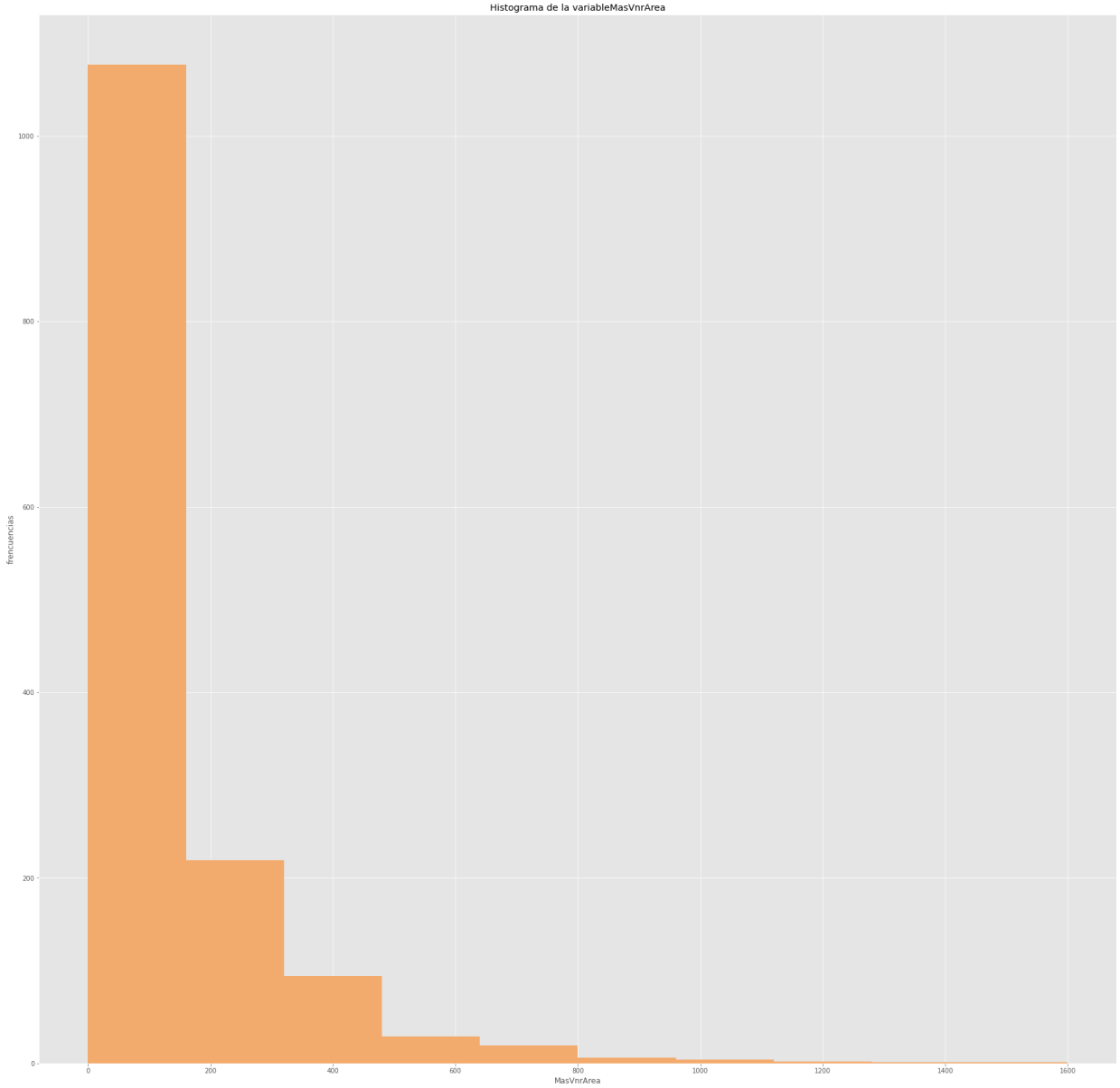

Histograma de la variableYearBuilt
frencuencias
0
100
200
300
400
1880
1900
1920
1940
1960
1980
2000
YearBuilt

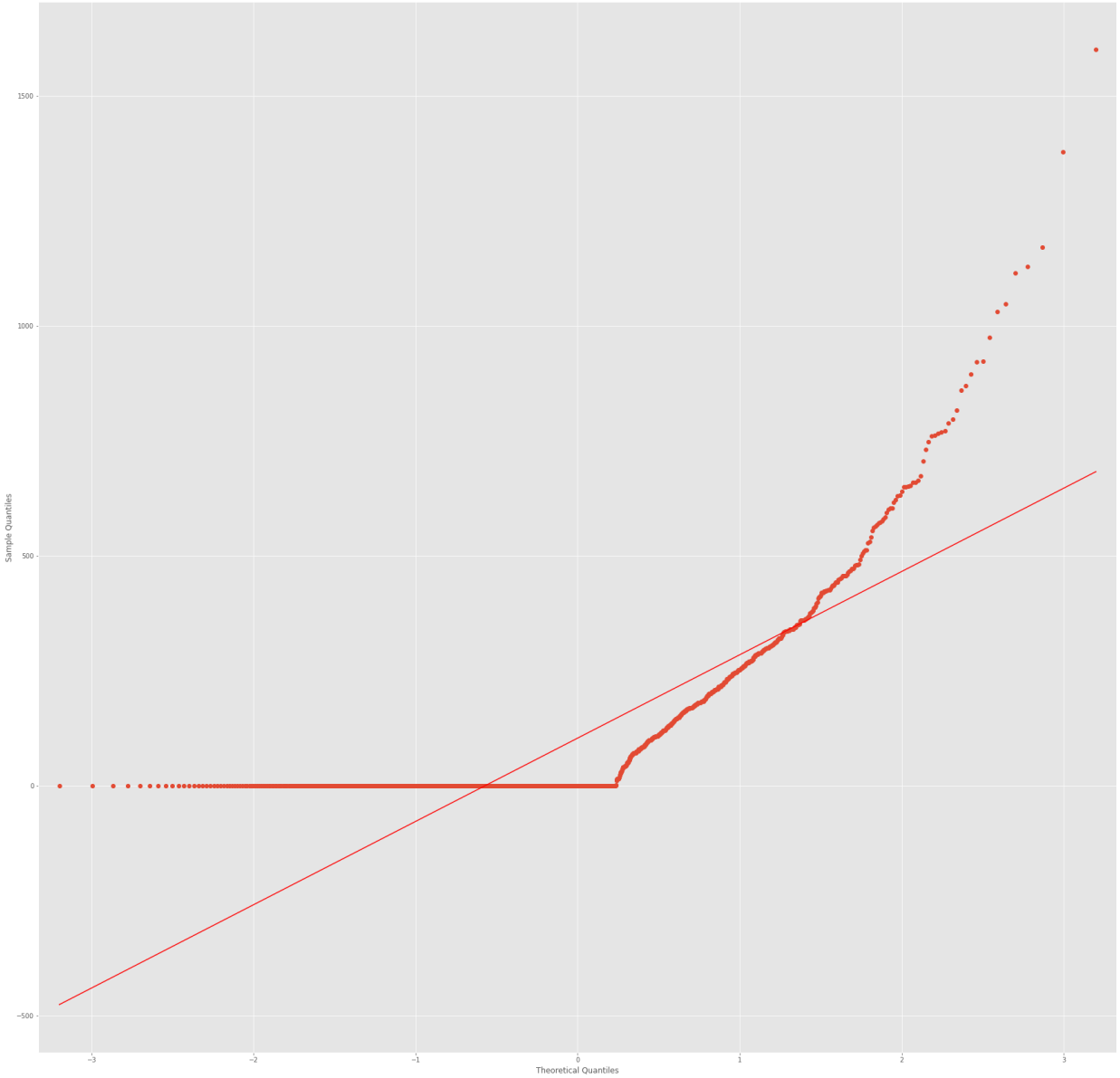


MasVnrArea

Se puede determinar que la variable MasVnrArea no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('MasVnrArea')
```

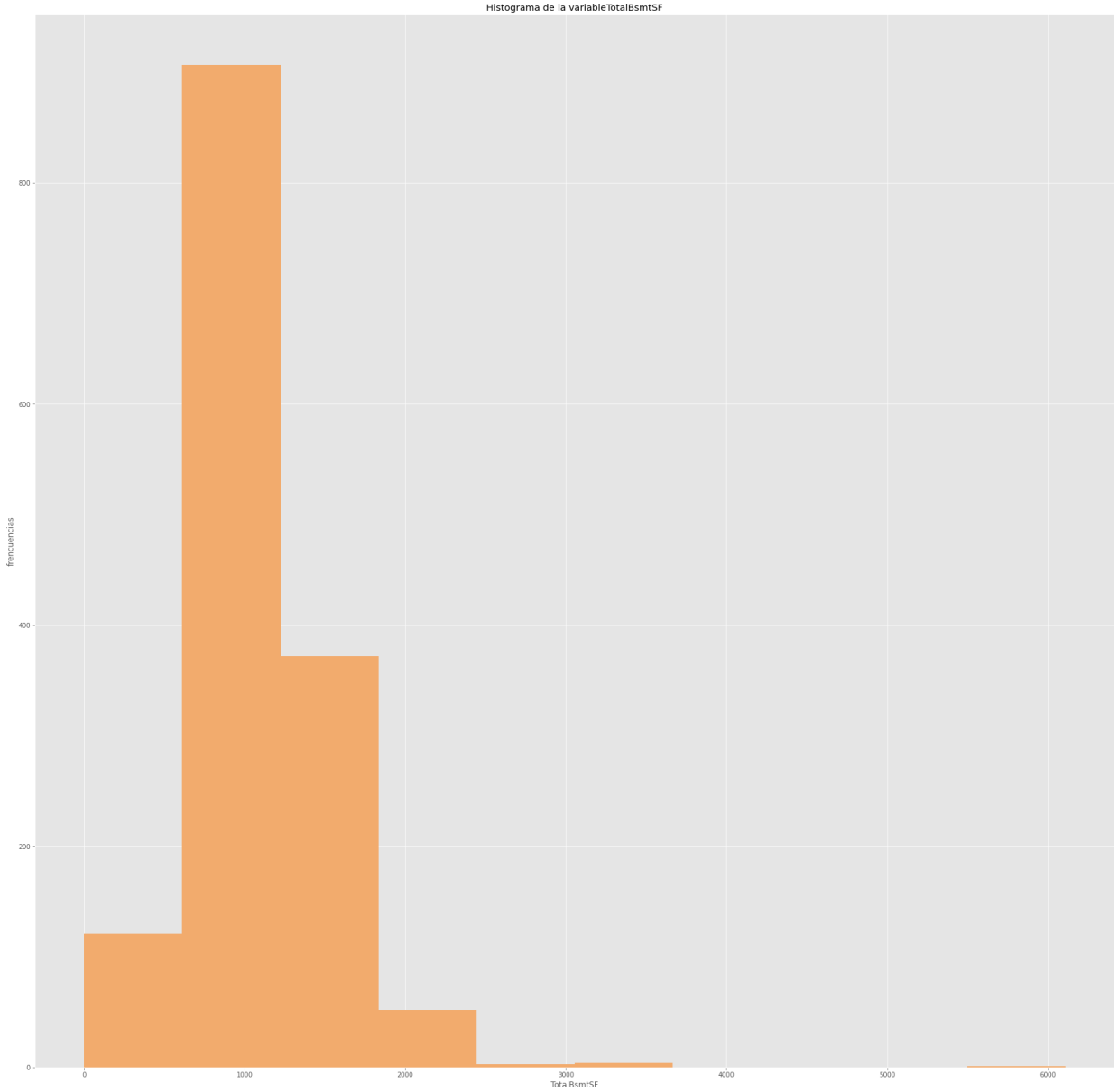

Histograma de la variableMasVnrArea
frencuencias
MasVnrArea
0
200
400
600
800
1000
1200
1400
1600

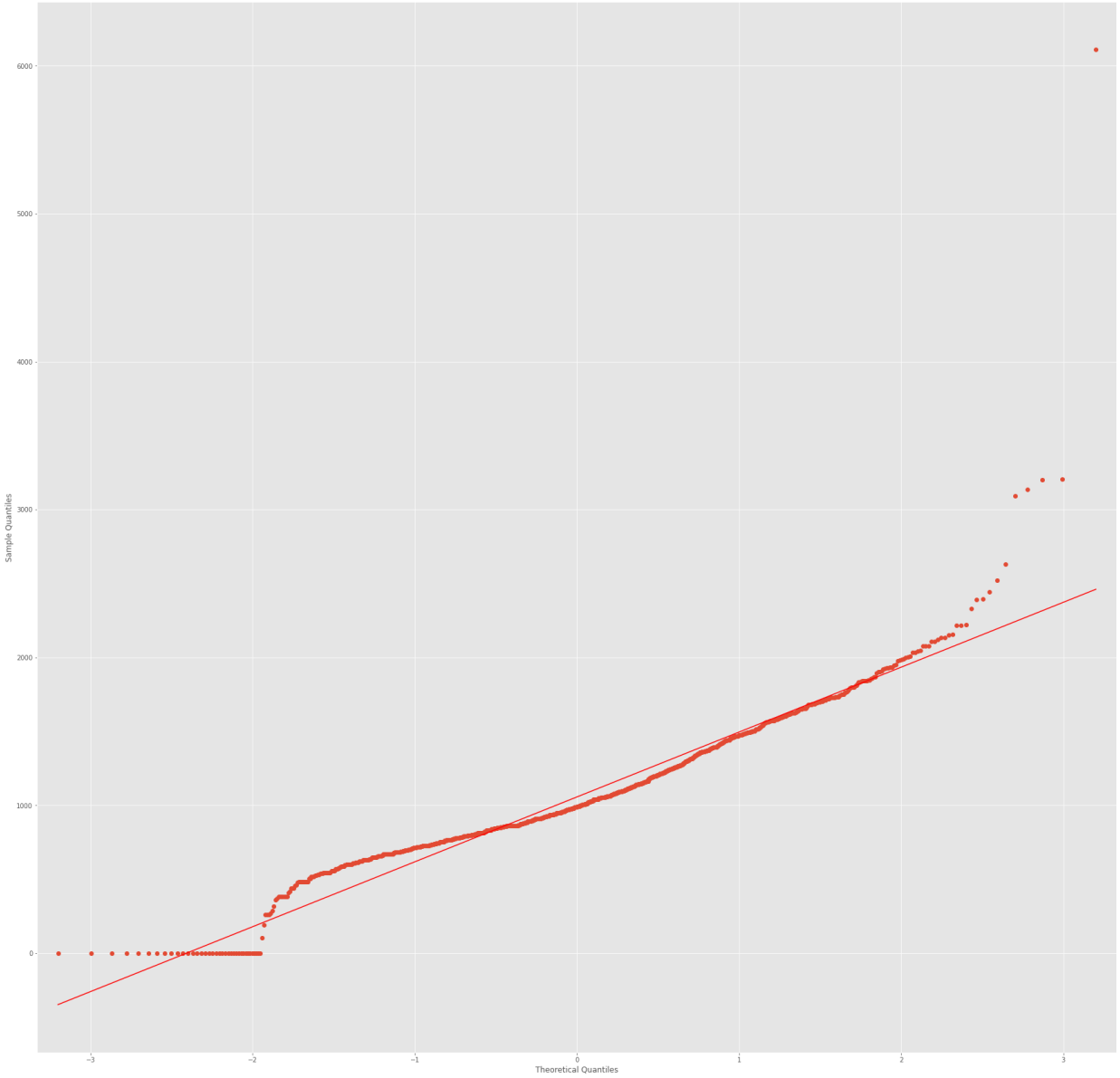


TotalBsmtSF

Se puede determinar que la variable TotalBsmtSF no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('TotalBsmtSF')
```

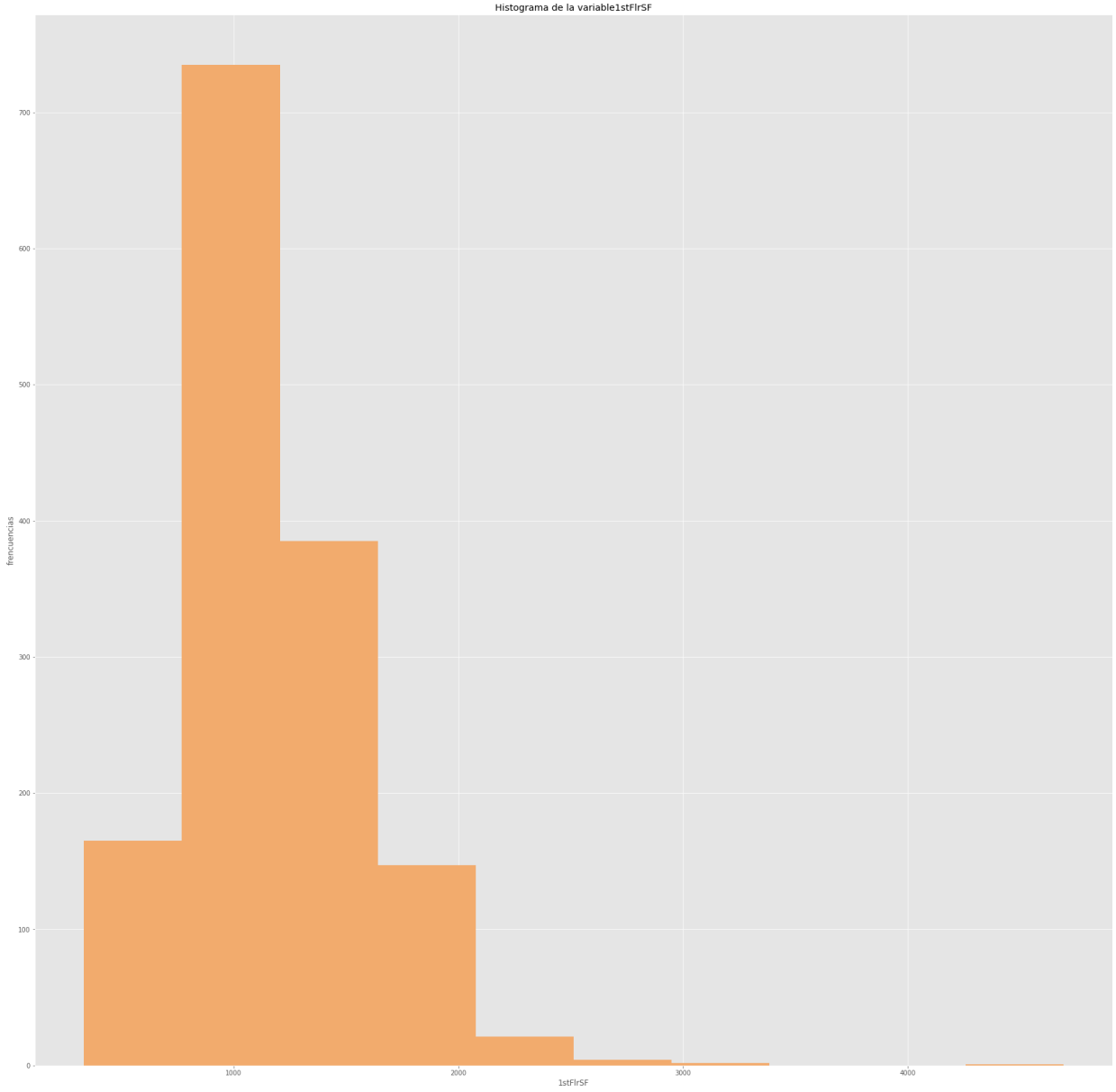

Histograma de la variableTotalBsmtSF
frencuencias
0
200
400
600
800
0
1000
2000
3000
4000
5000
6000
TotalBsmtSF

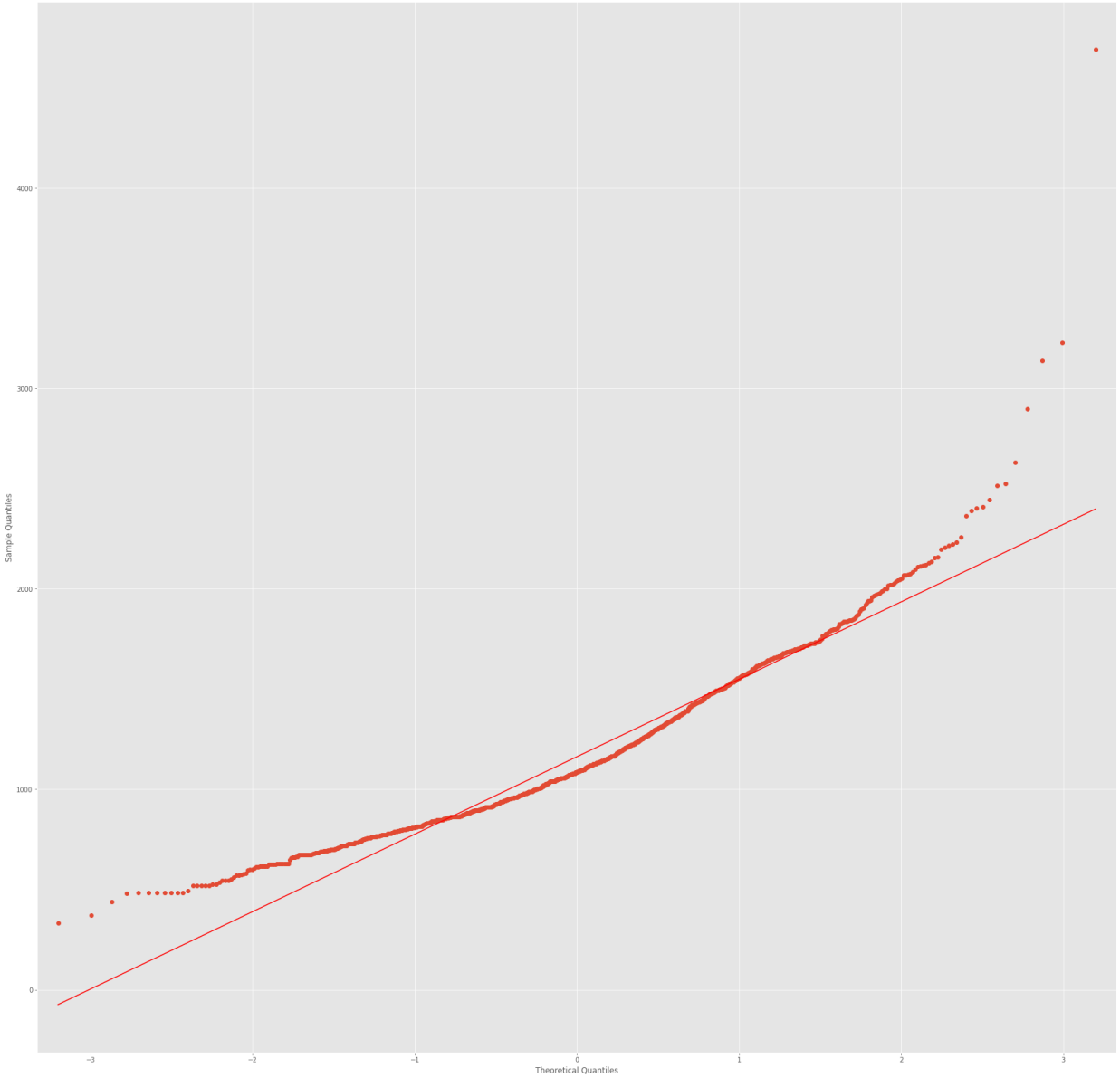


1stFlrSF

Se puede determinar que la variable 1stFlrSF no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('1stFlrSF')
```

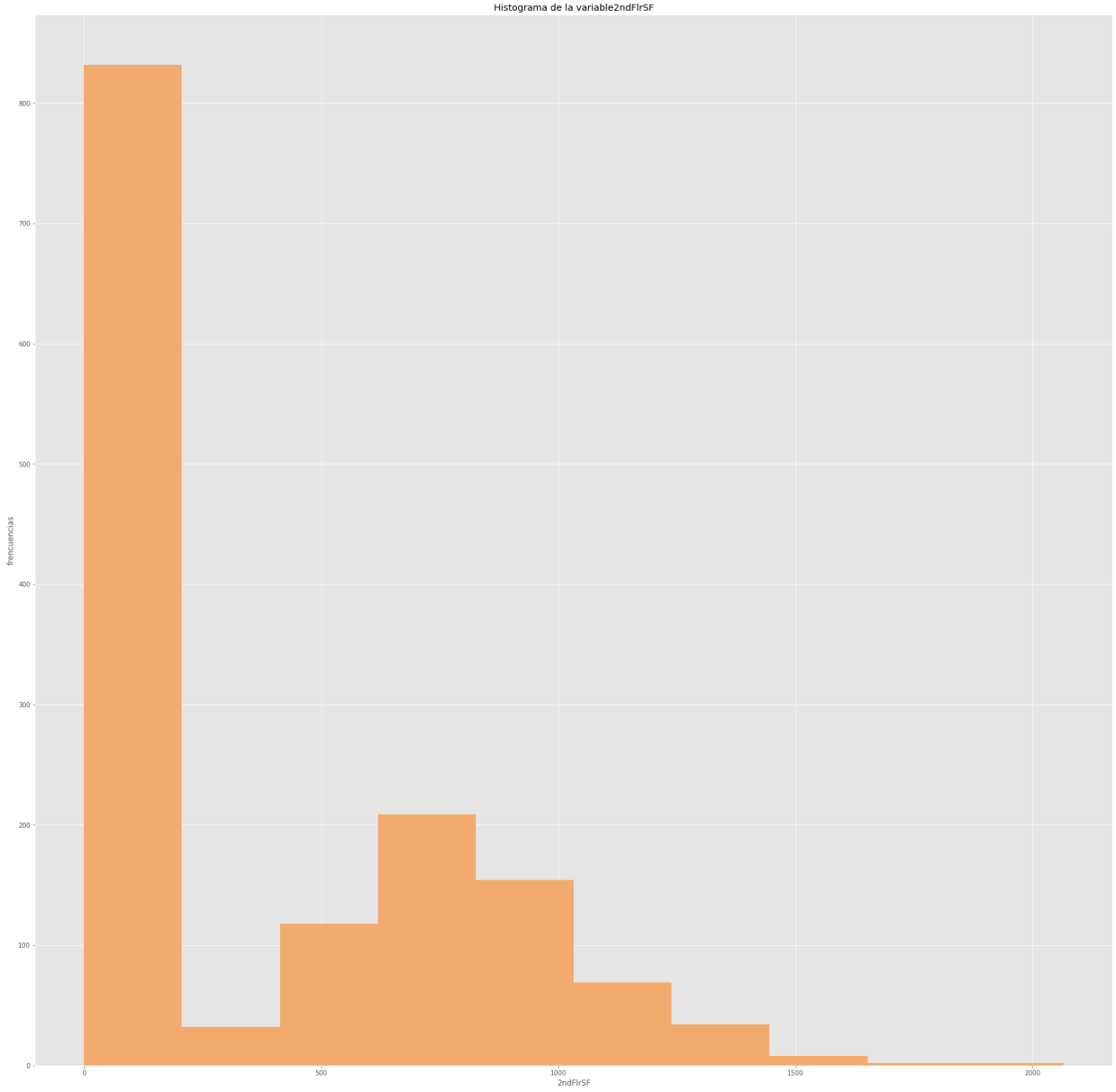

Histograma de la variable1stFlrSF
frencuencias
0
100
200
300
400
500
600
700
1000
2000
3000
4000
1stFlrSF

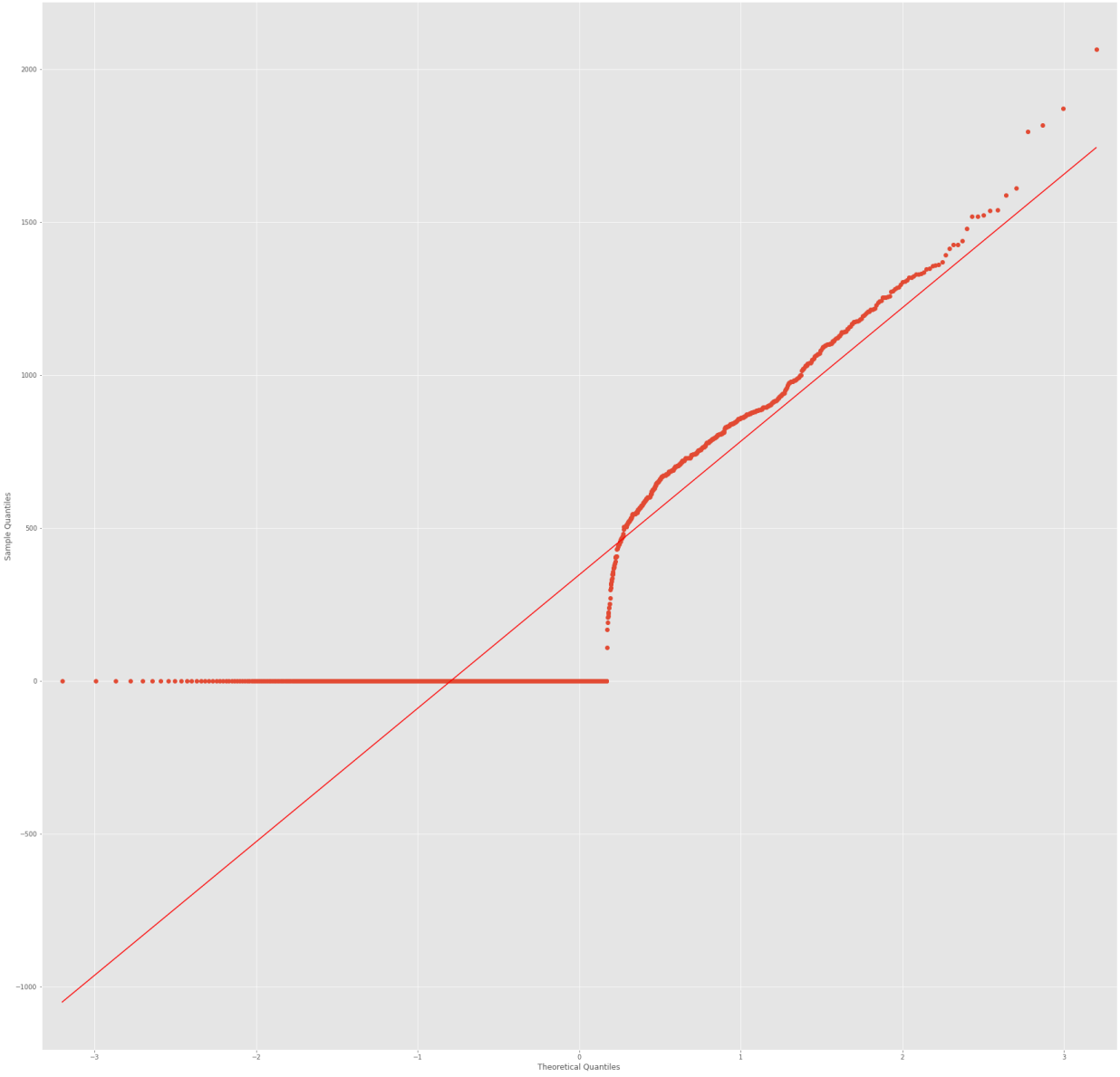


2ndFlrSF

Se puede determinar que la variable 2ndFlrSF no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('2ndFlrSF')
```

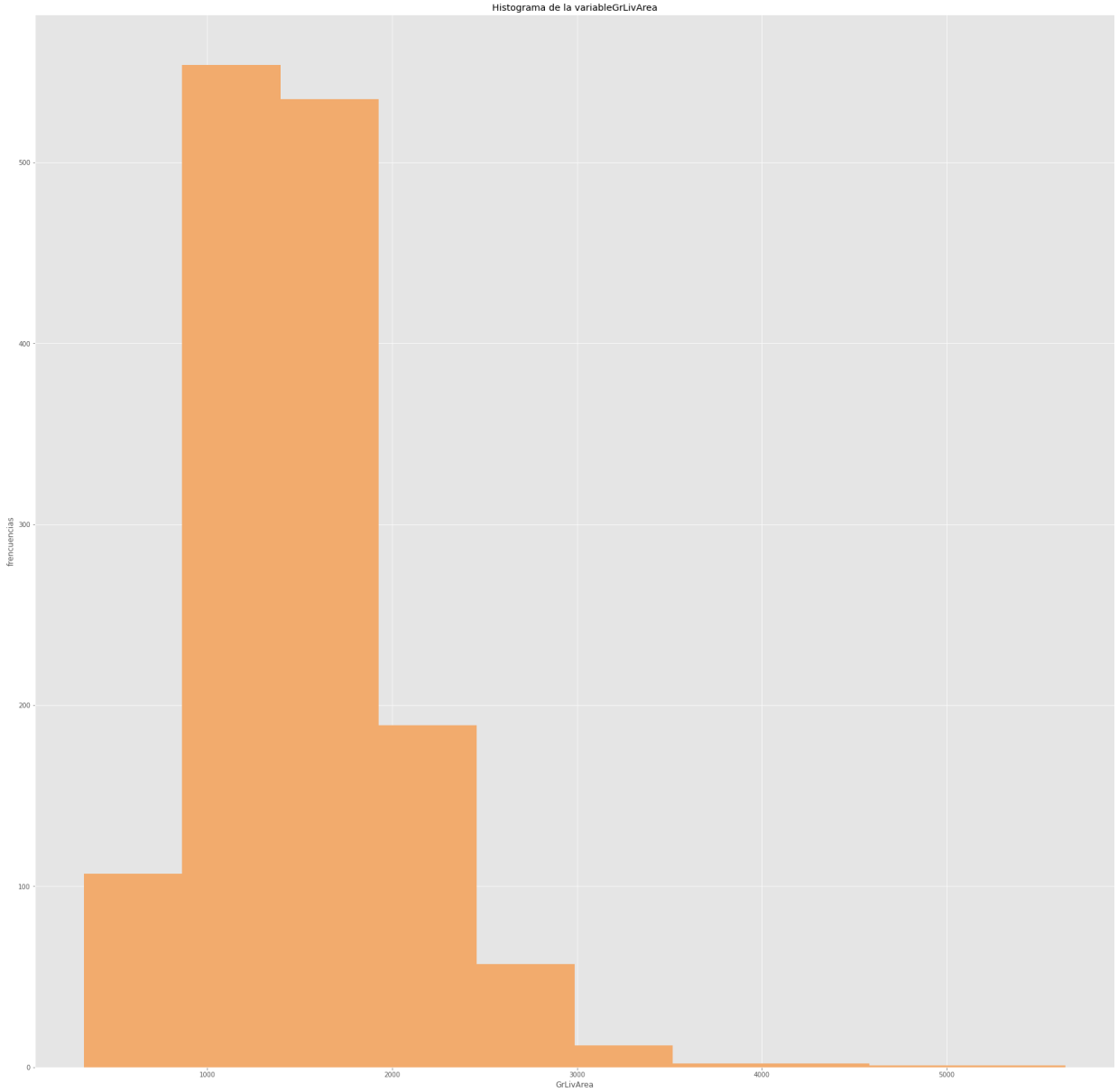

Histograma de la variable2ndFlrSF
frencuencias
2ndFlrSF
0
100
200
300
400
500
600
700
800
0
500
1000
1500
2000

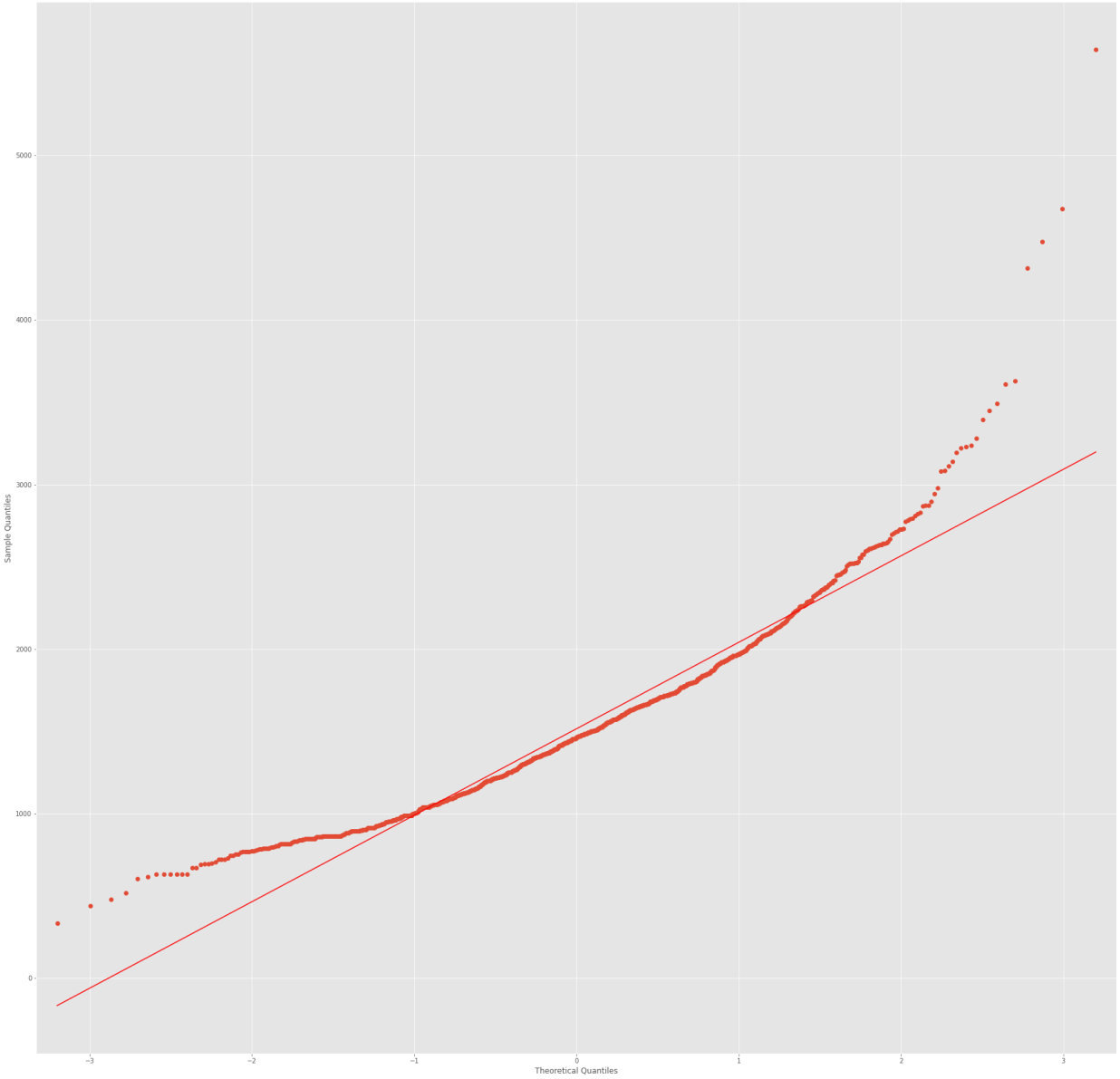


GrLivArea

Se puede determinar que la variable GrLivArea no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('GrLivArea')
```

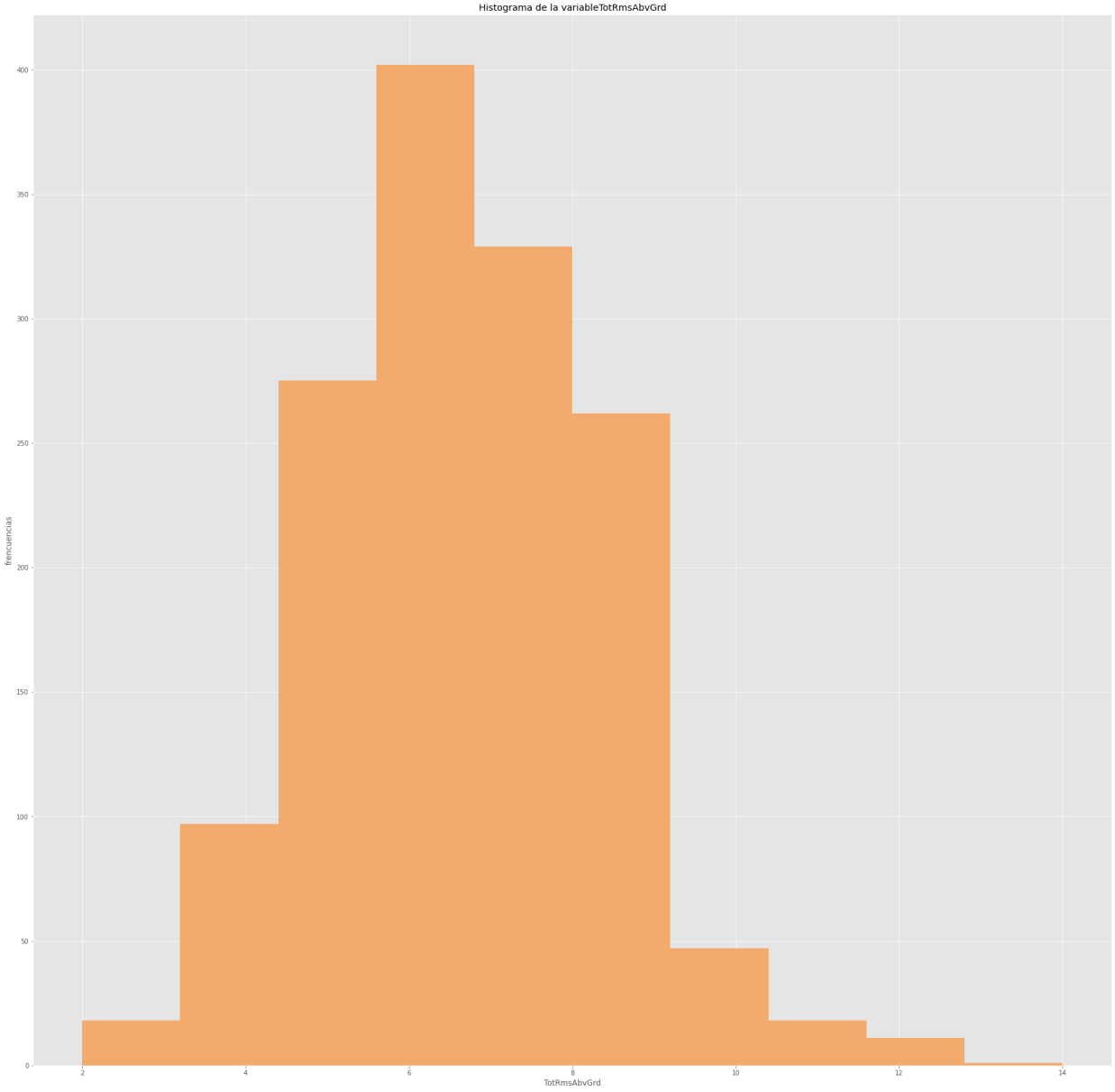

Histograma de la variableGrLivArea
frencuencias
0
100
200
300
400
500
1000
2000
3000
4000
5000
GrLivArea

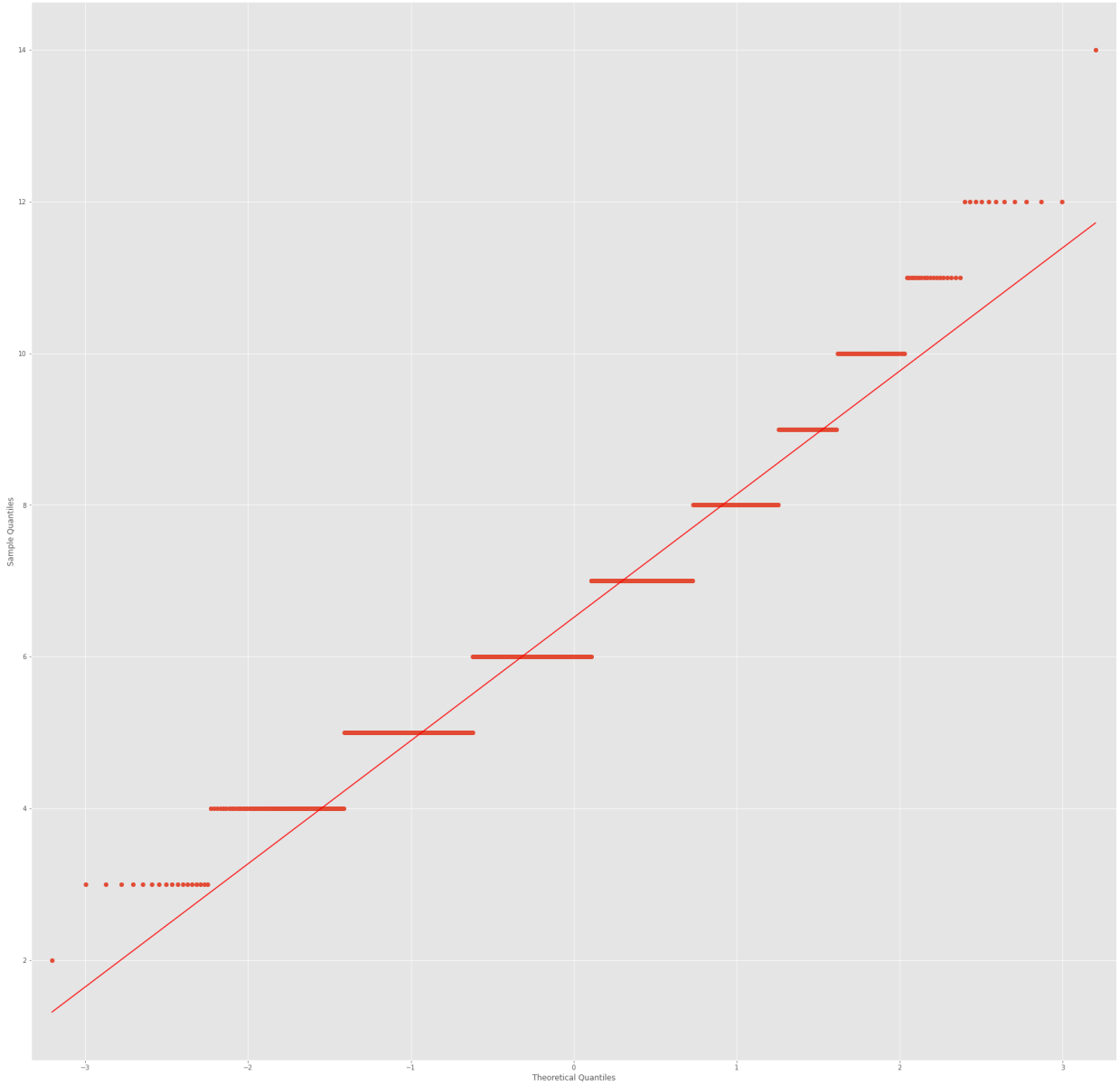


TotRmsAbvGrd

Se puede determinar que la variable TotRmsAbvGrd no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('TotRmsAbvGrd')
```

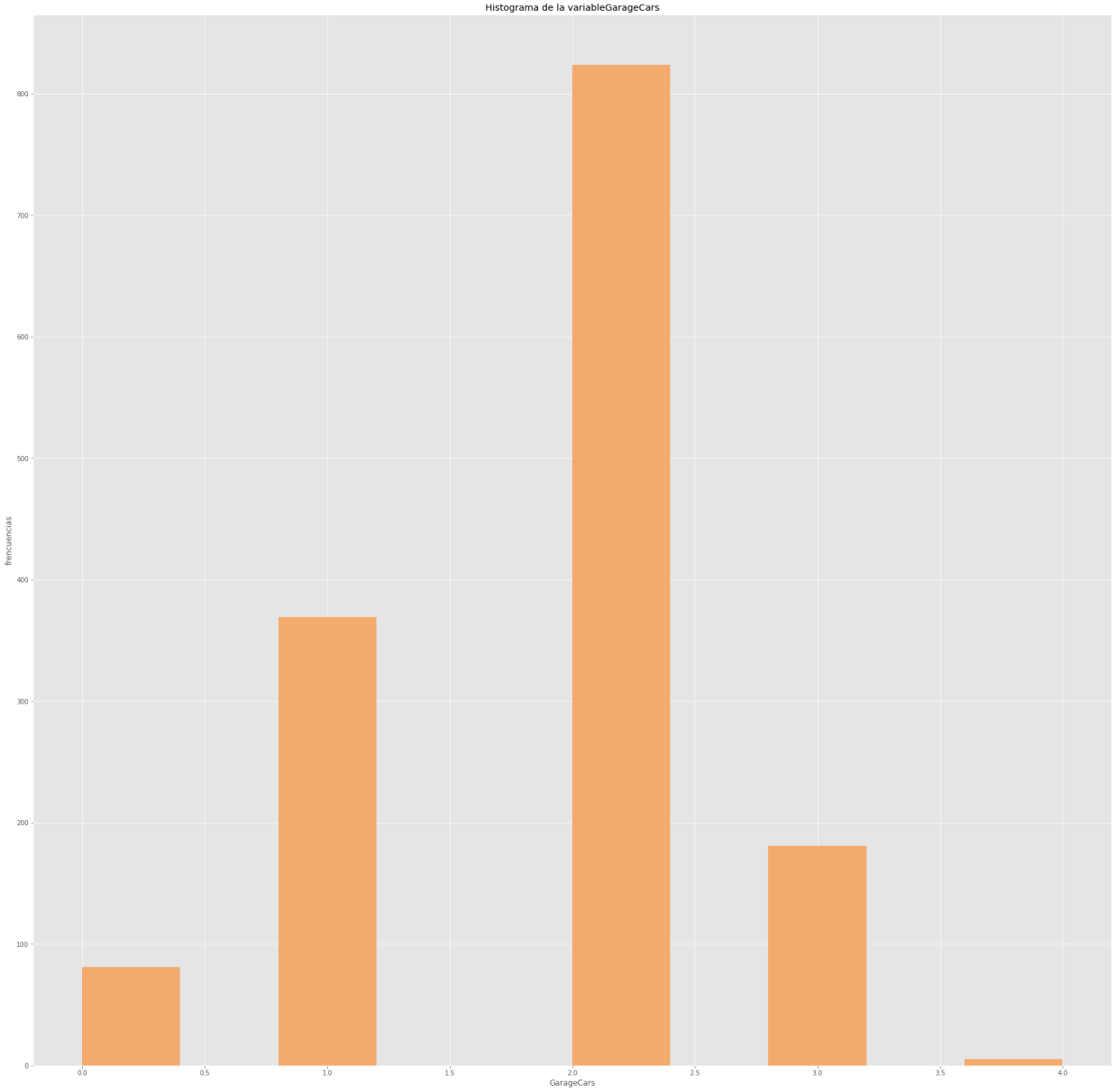

Histograma de la variableTotRmsAbvGrd
frencuencias
TotRmsAbvGrd
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14

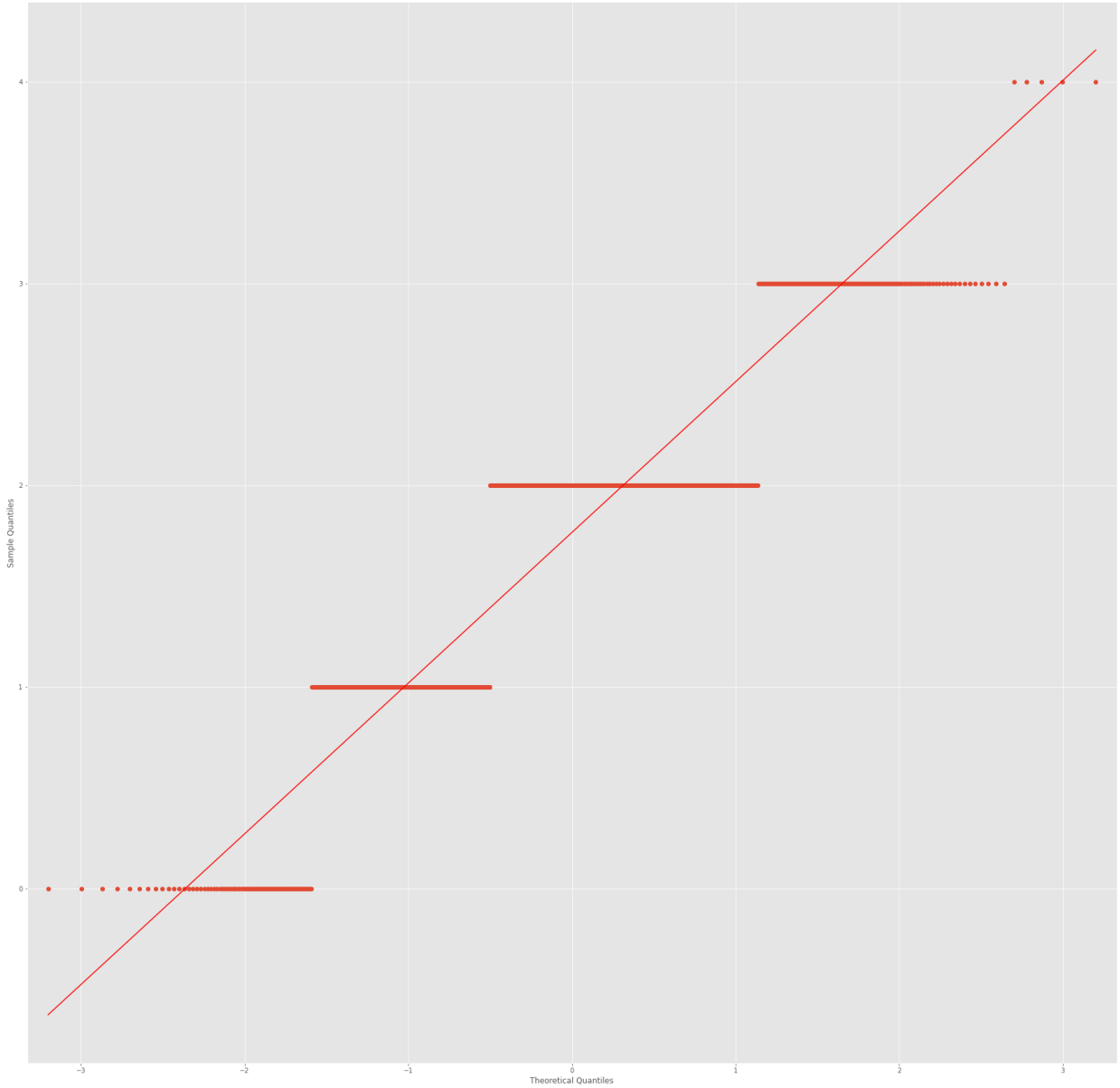


GarageCars

Se puede determinar que la variable GarageCars no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('GarageCars')
```

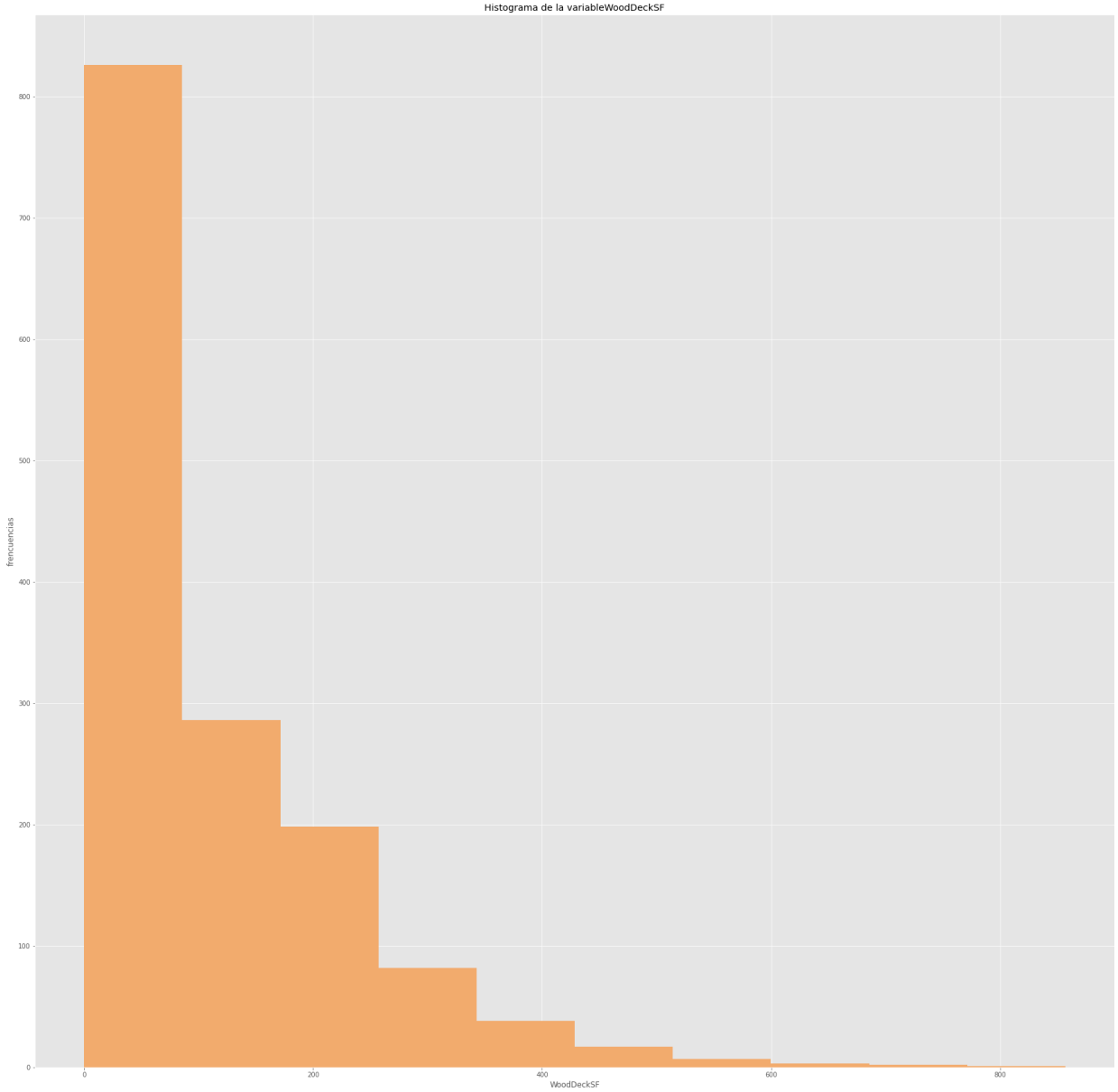

Histograma de la variableGarageCars
frencuencias
GarageCars
0
100
200
300
400
500
600
700
800
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

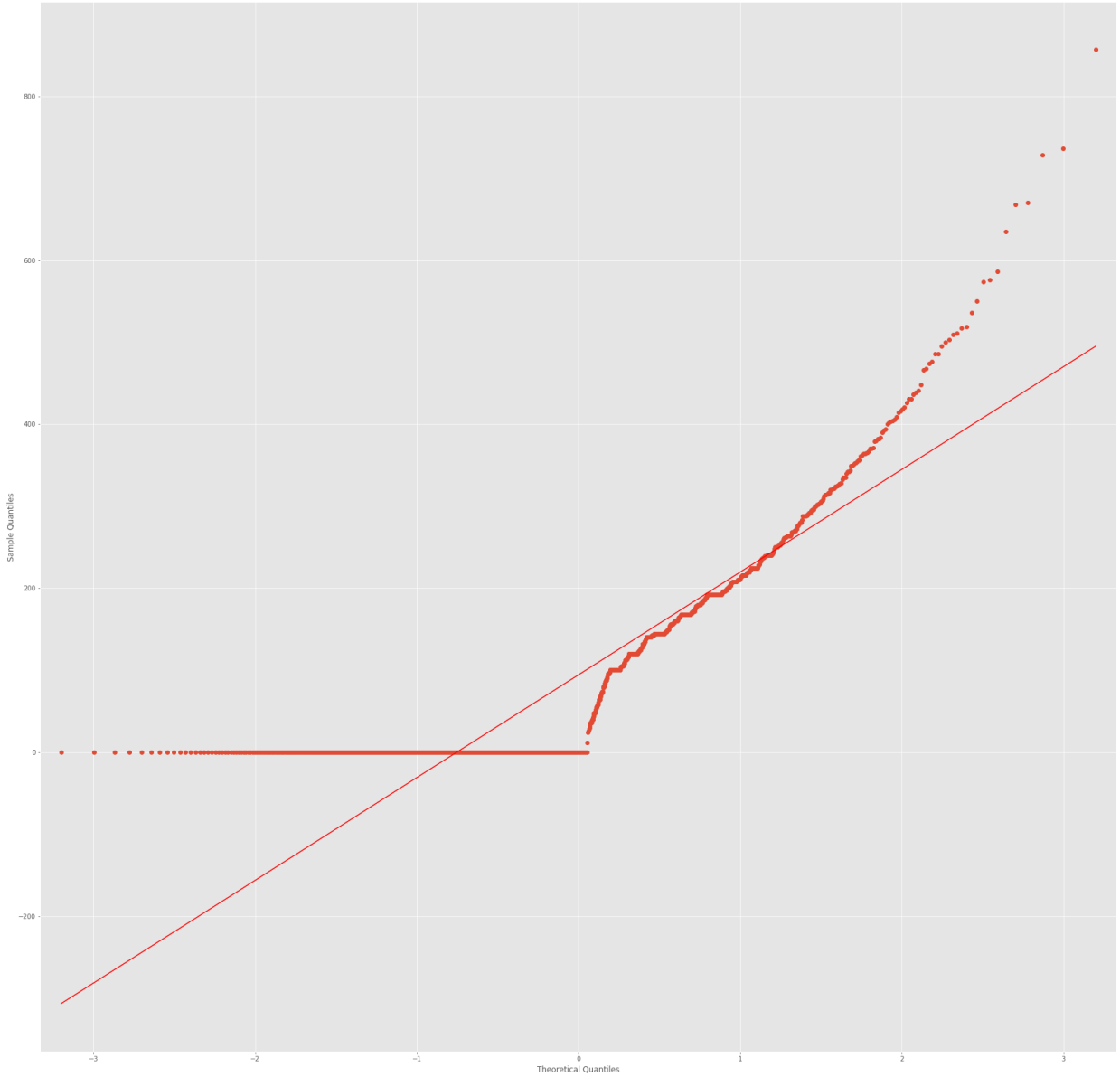


WoodDeckSF

Se puede determinar que la variable WoodDeckSF no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('WoodDeckSF')
```

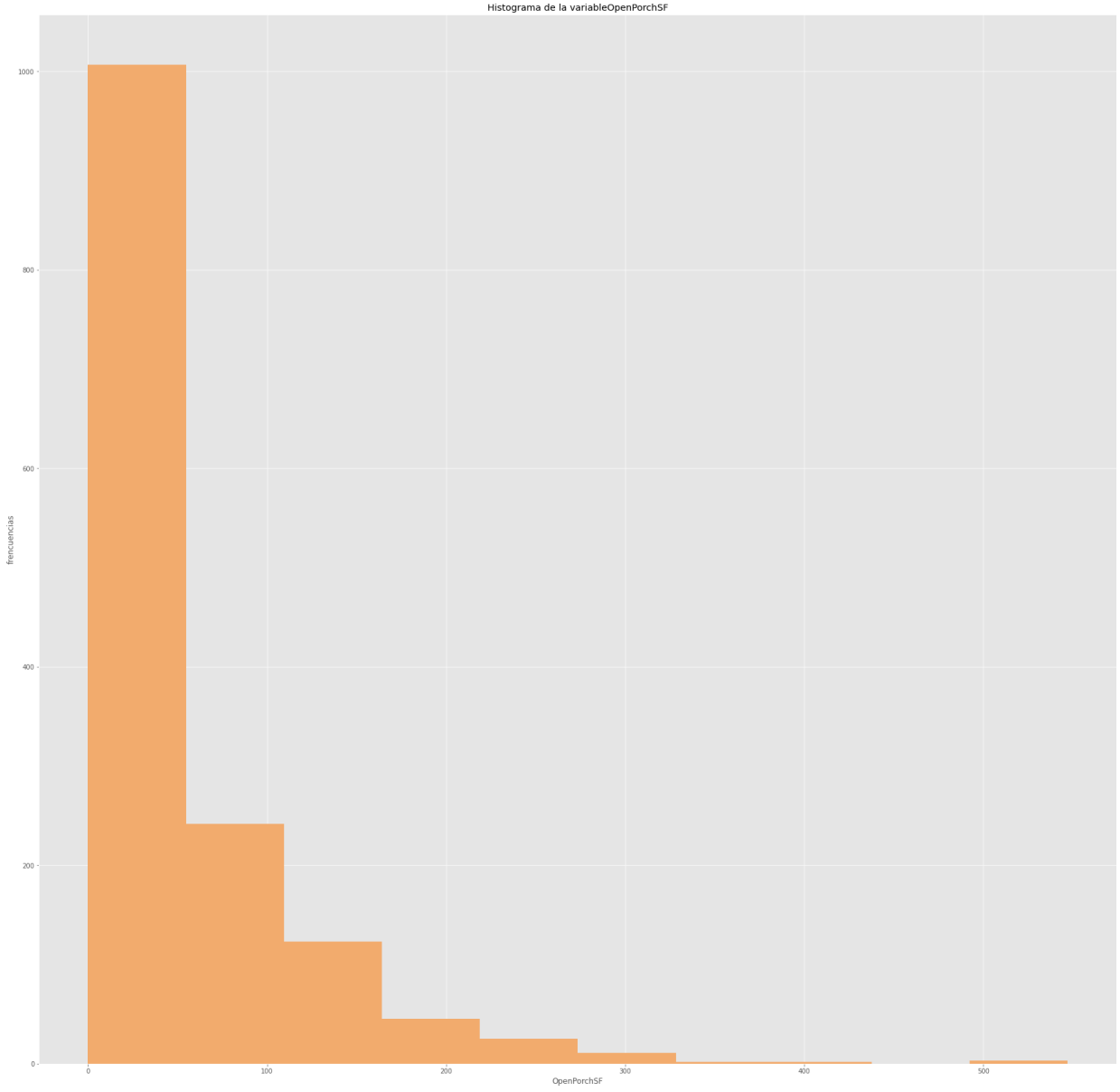

Histograma de la variableWoodDeckSF
frencuencias
WoodDeckSF
0
100
200
300
400
500
600
700
800
0
200
400
600
800

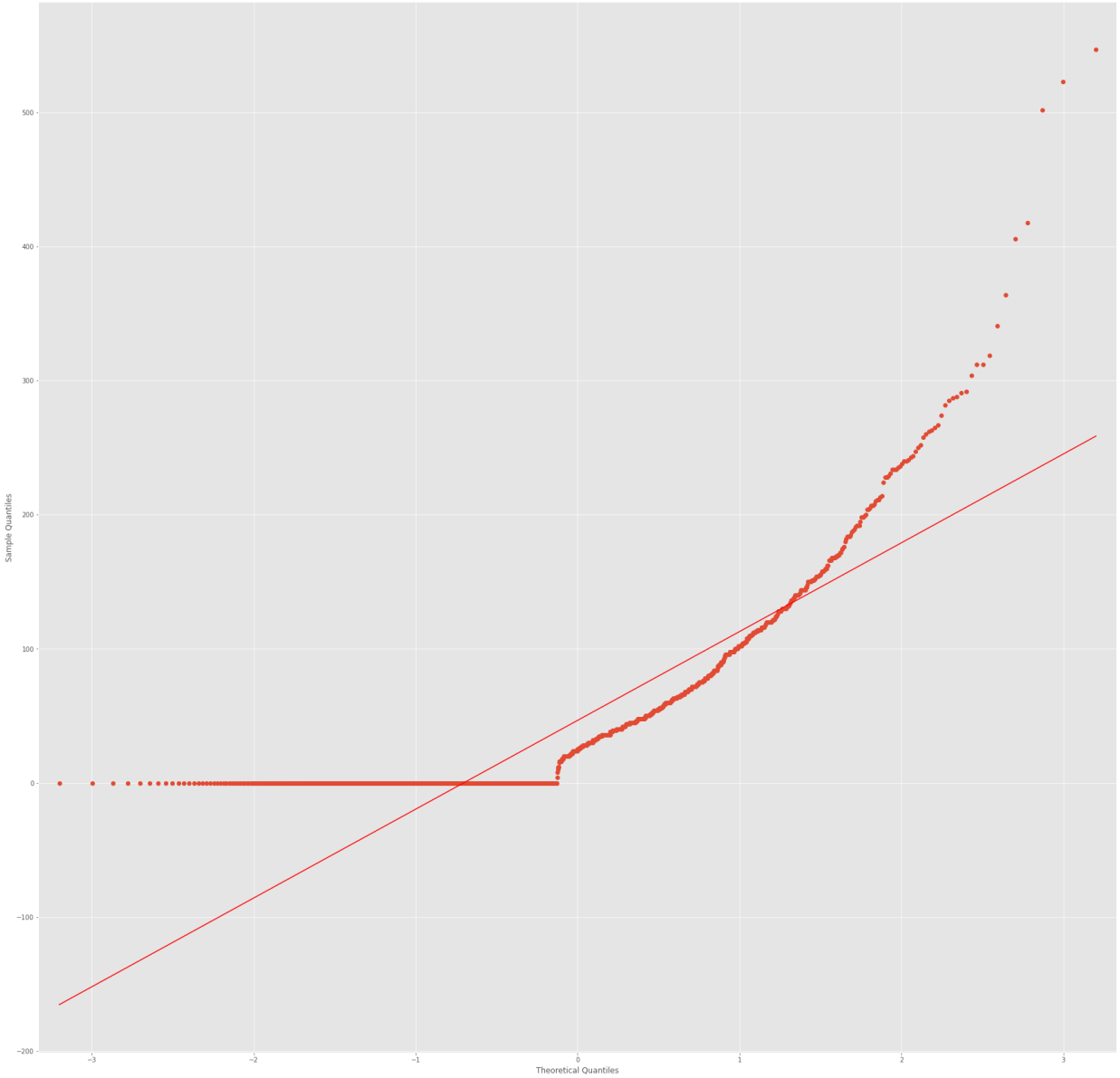


OpenPorchSF

Se puede determinar que la variable OpenPorchSF no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('OpenPorchSF')
```

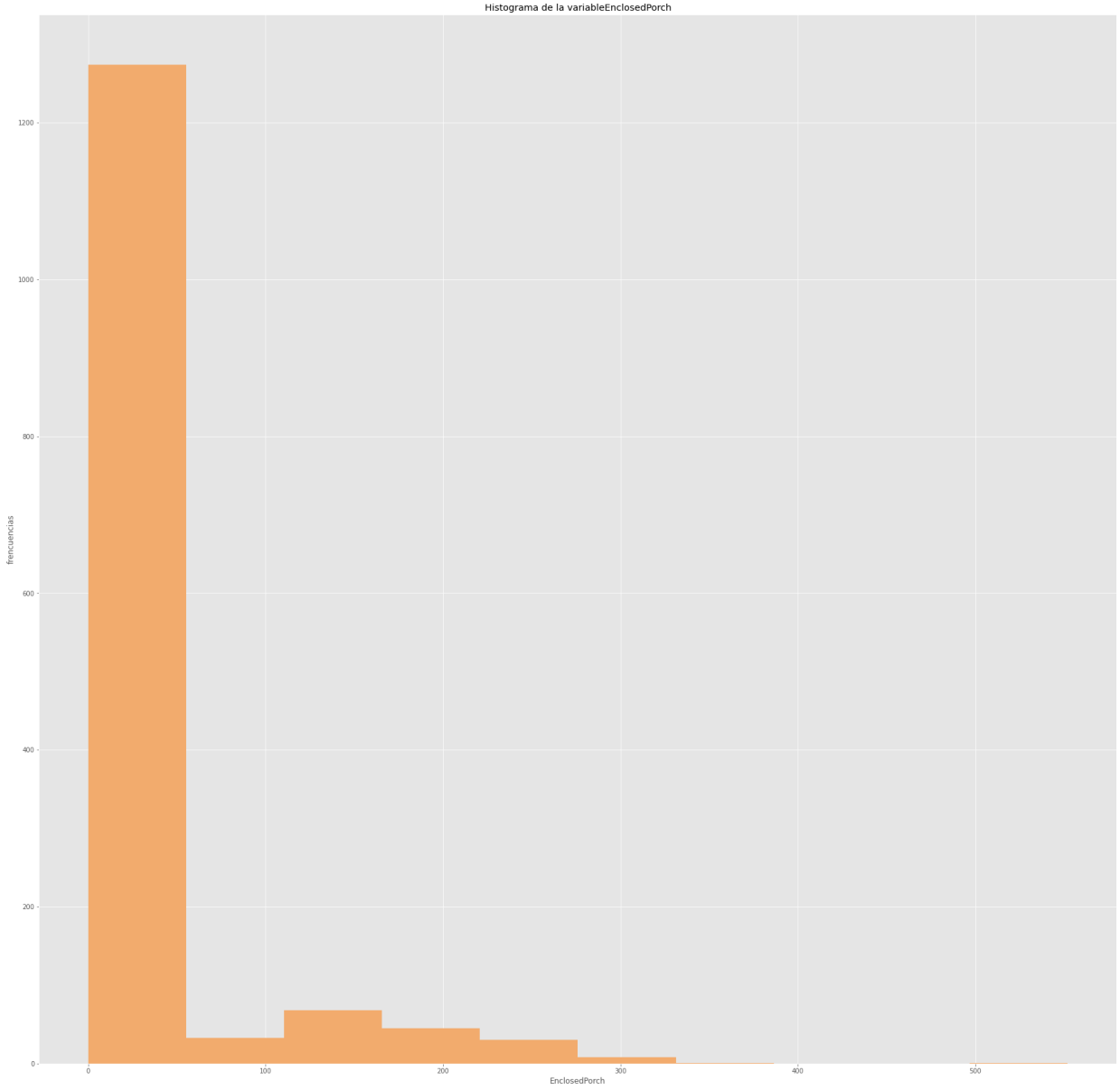

Histograma de la variableOpenPorchSF
frencuencias
1000
800
600
400
200
0
0
100
200
300
400
500
OpenPorchSF

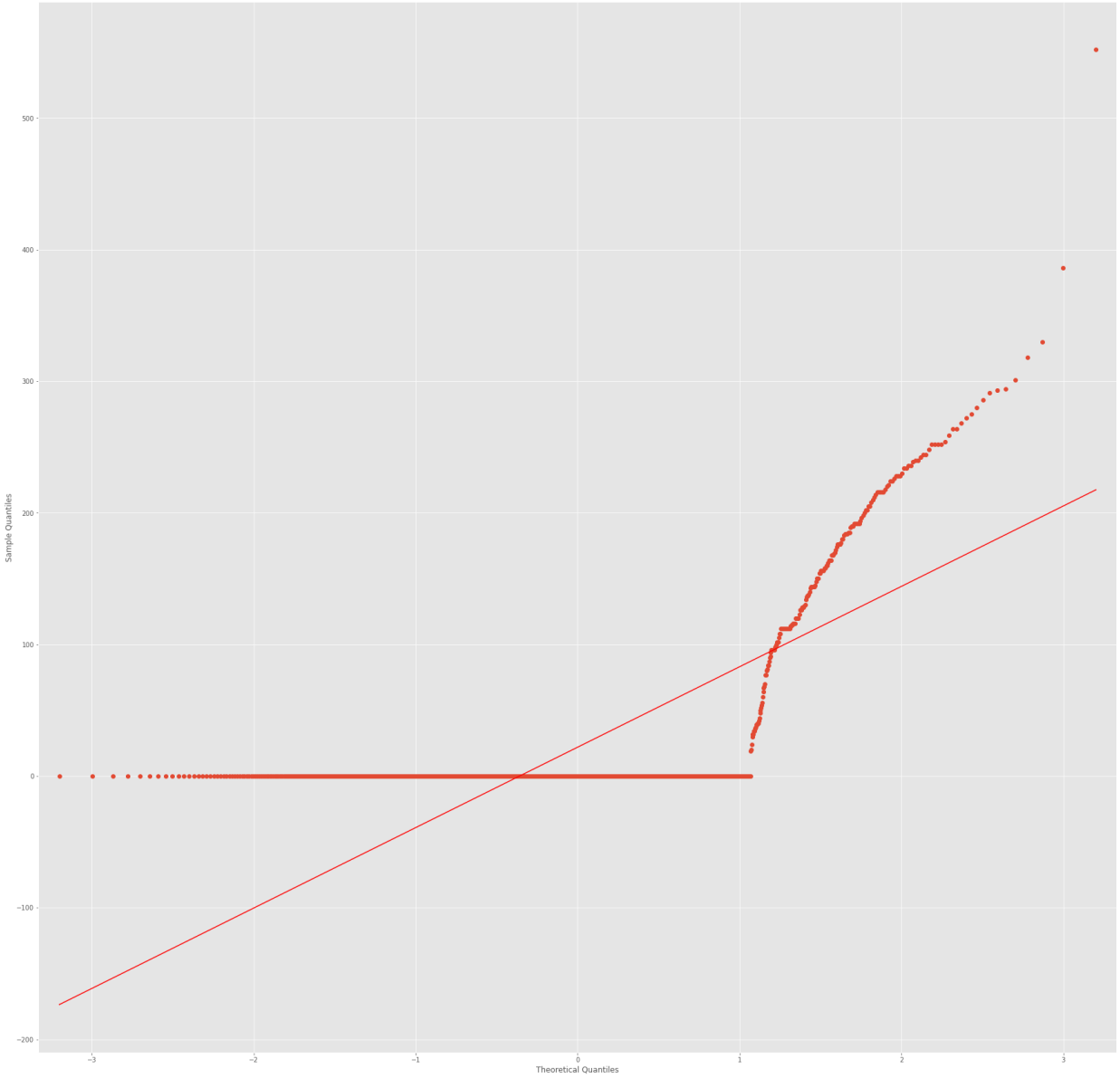


EnclosedPorch

Se puede determinar que la variable EnclosedPorch no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('EnclosedPorch')
```

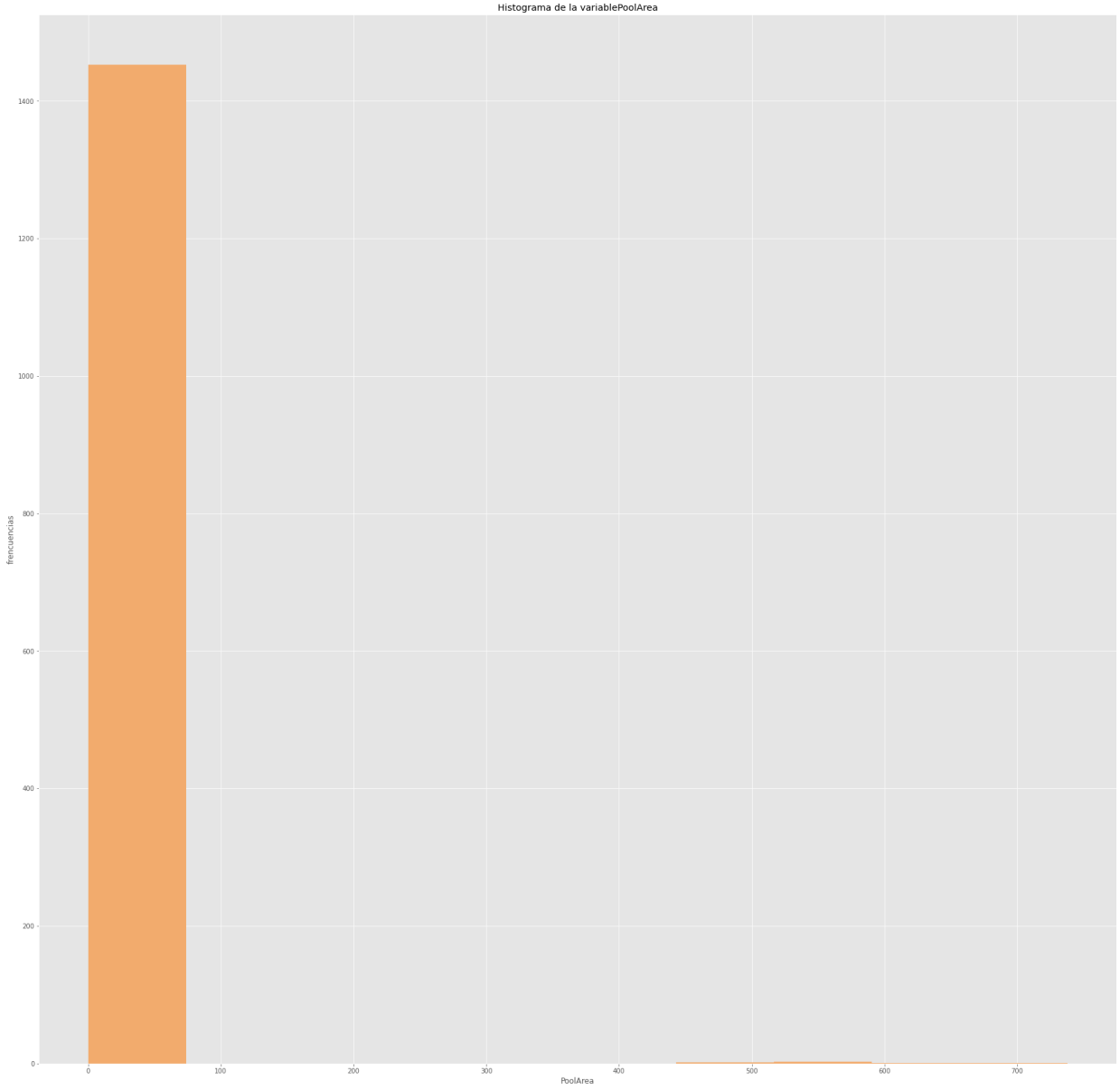

Histograma de la variableEnclosedPorch
frencuencias
EnclosedPorch
0
200
400
600
800
1000
1200
0
100
200
300
400
500

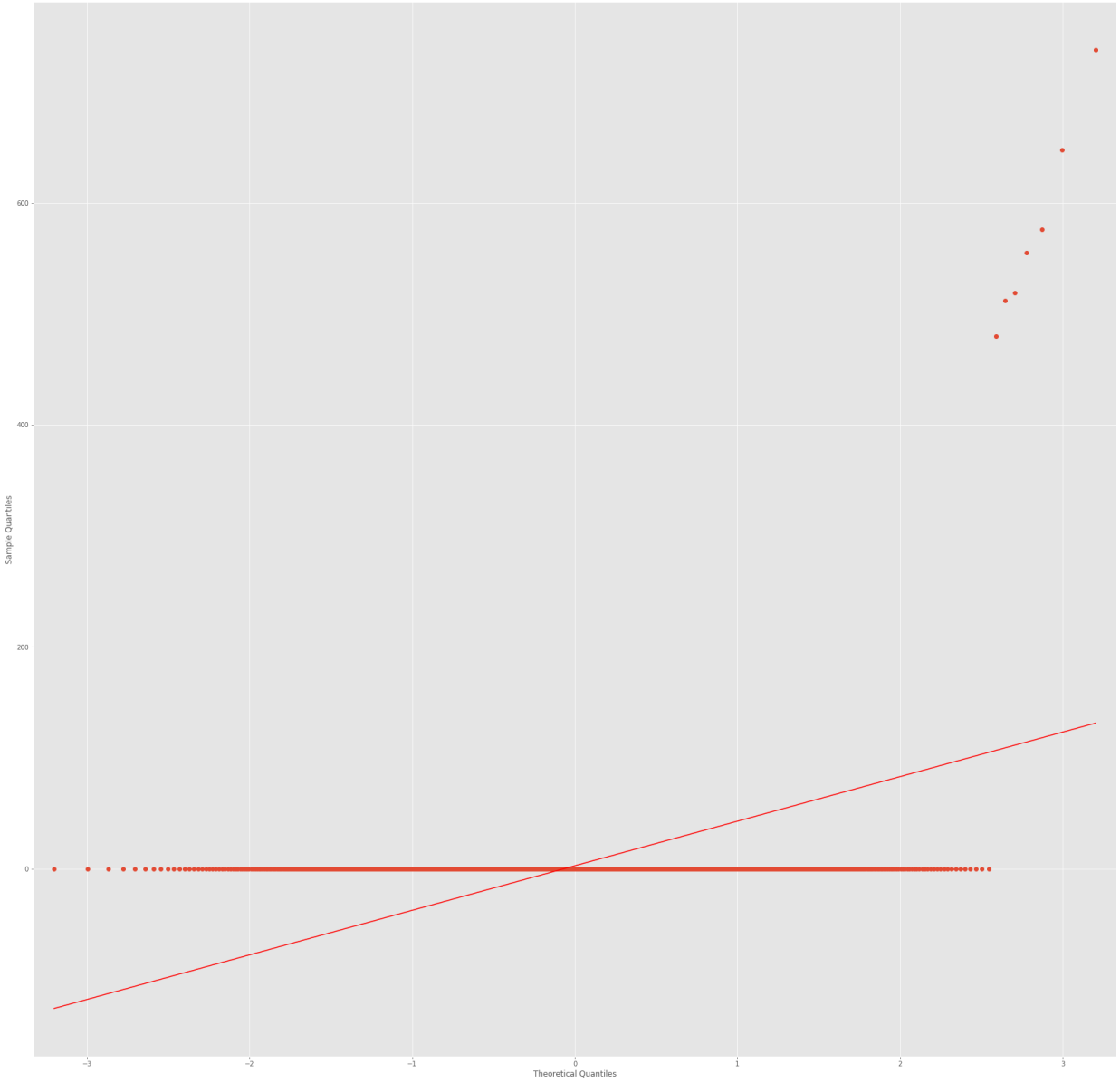


PoolArea

Se puede determinar que la variable PoolArea no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

In [ ]:
```
get_histogram_qq('PoolArea')
```

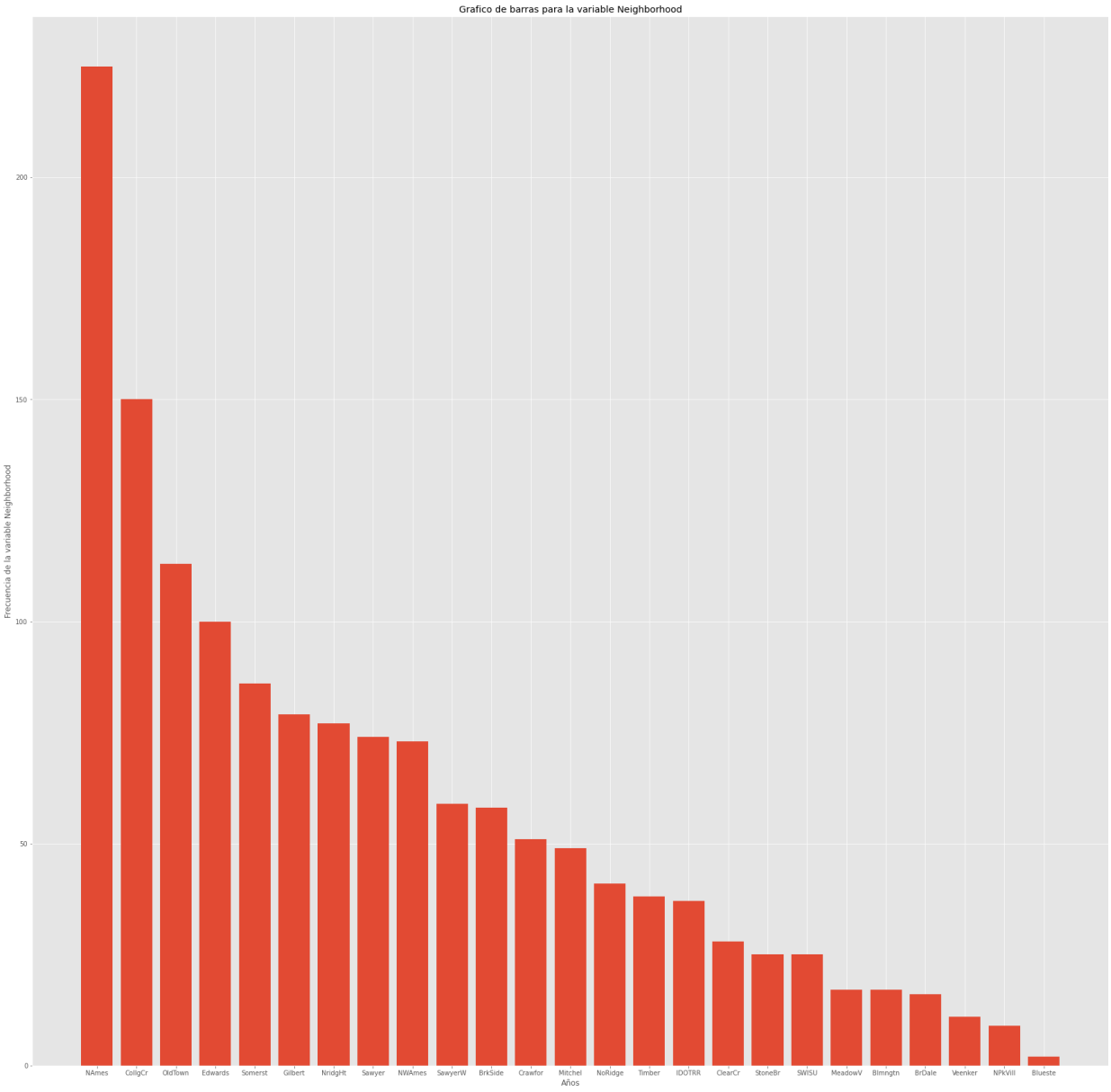

Histograma de la variablePoolArea
frencuencias
PoolArea
0
200
400
600
800
1000
1200
1400
0
100
200
300
400
500
600
700



Neighborhood

Se puede determinar que la variable Neighborhood no sigue una disctribucion normal debido a que el histograma no sigue una forma de campana y el diagrama QQ nos muestra que los datos son muy distintos.

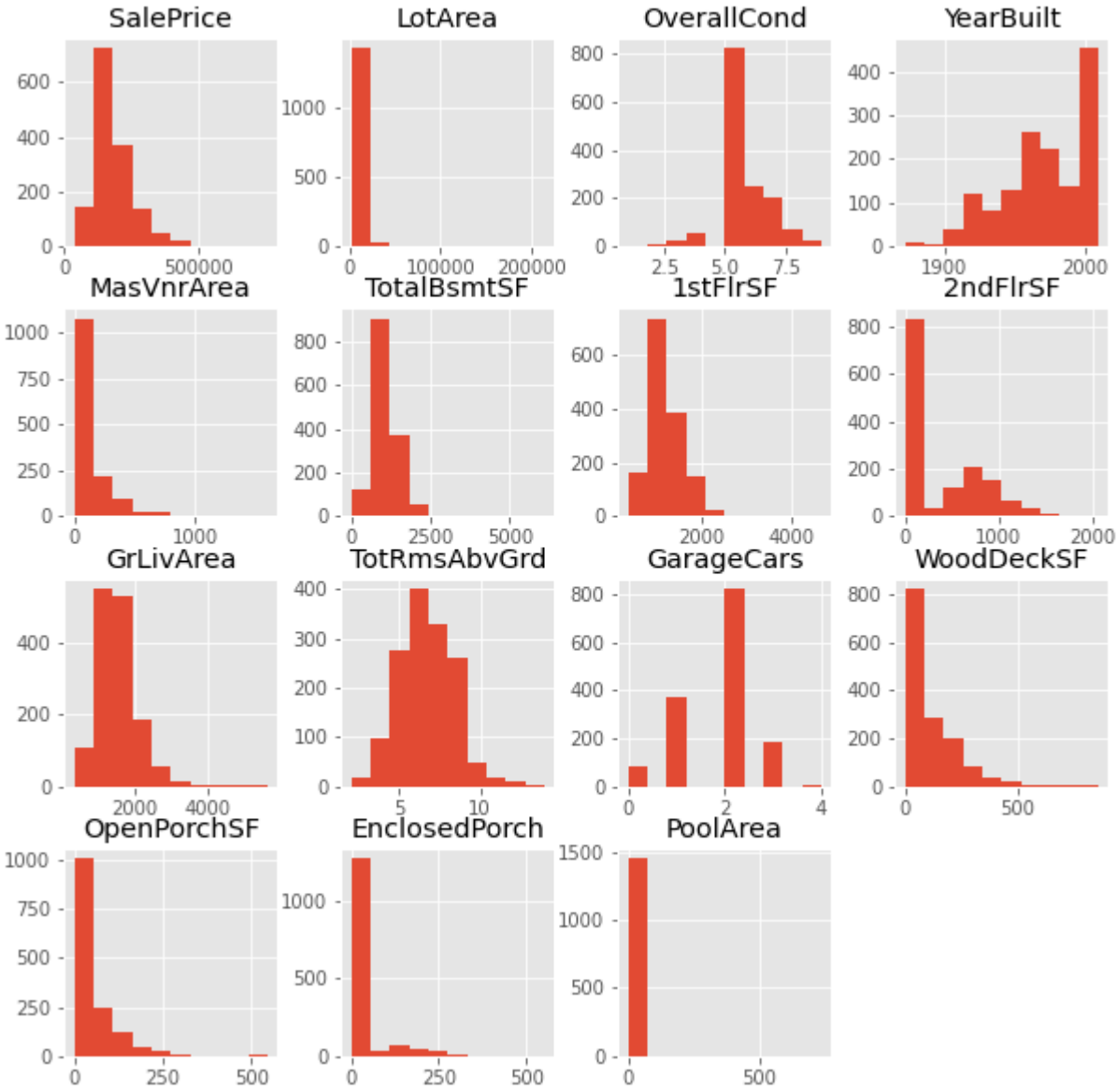
In [ ]:
```
eje_x = np.array(pd.value_counts(data['Neighborhood']).keys())
eje_y = pd.value_counts(data['Neighborhood'])

plt.bar(eje_x, eje_y)
plt.rcParams['figure.figsize'] = (10, 10)
plt.ylabel('Frecuencia de la variable Neighborhood')
plt.xlabel('Años')
plt.title('Grafico de barras para la variable Neighborhood')
plt.show()
```



3. Incluya un análisis de grupos en el análisis exploratorio. Explique las características de los grupos.

Se puede concluir que los datos normalizados son viables para el uso de clusters o grupos. Se logra llegar a esta conclucion debido a que nuestro test de hopkins sale de 0.08 junto con la grafica VAT. Con la grafica del codo se puede determinar que se pueden utilizar dos clusters debido a que es en ese dato donde se encuentra mas marcado el codo. Pero tambien se podria usar 7 debido a que tambien se encuenatra marcada ahi una punta.

In [ ]:
```
data.hist()
plt.show()
```



In [ ]:
```
# NORMALIZAMOS DATOS
if 'Neighborhood' in data.columns:
    usefullAttr.remove('Neighborhood')
data = train[usefullAttr]
X = []
for column in data.columns:
    try:
        column
        if column != 'Neighborhood' or column != 'SalePrice':
            data[column] = (data[column]-data[column].mean()) / \
                data[column].std()
            X.append(data[column])
    except:
        continue
data_clean = data.dropna(subset=usefullAttr, inplace=True)
X_Scale = np.array(data)
X_Scale
```

```
<ipython-input-23-ba92df615b8e>:10: 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_guide/indexing.html#returning-a-view-versus-a-copy
  data[column] = (data[column]-data[column].mean()) / \
C:\Users\ALIEWARE\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\util\_decorators.py:311: 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_guide/indexing.html#returning-a-view-versus-a-copy
  return func(*args, **kwargs)
```

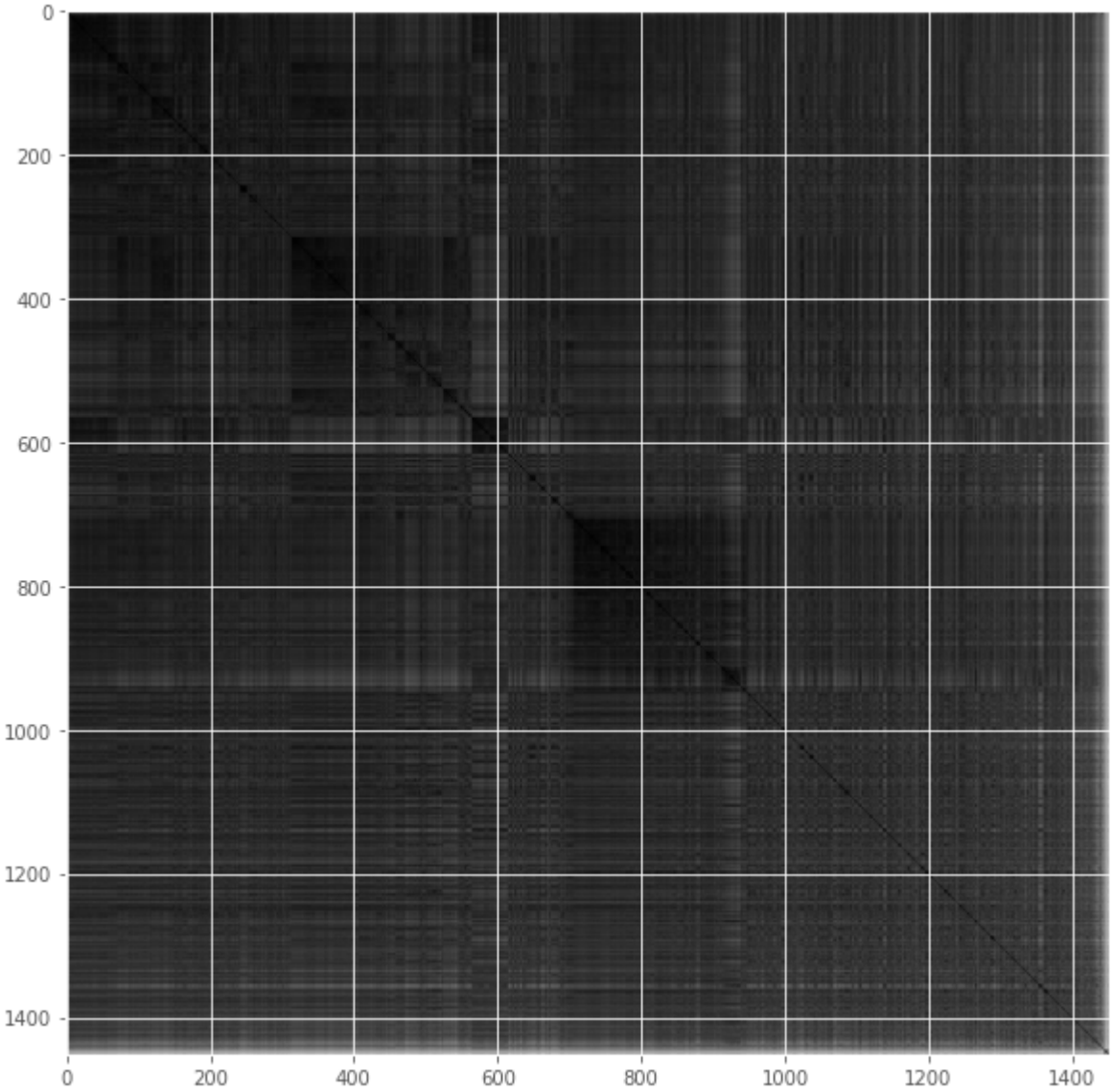
Out[ ]:
```
array([[ 0.34715427, -0.20707076, -0.51702265, ...,  0.216429  ,
        -0.35920182, -0.06866822],
       [ 0.00728582, -0.0918549 ,  2.17888118, ..., -0.70424195,
        -0.35920182, -0.06866822],
       [ 0.53597007,  0.07345481, -0.51702265, ..., -0.07033736,
        -0.35920182, -0.06866822],
       ...,
       [ 1.07724204, -0.14775964,  3.07751579, ...,  0.20133604,
        -0.35920182, -0.06866822],
       [-0.48835566, -0.08013294,  0.38161196, ..., -0.70424195,
         1.47328444, -0.06866822],
       [-0.42069666, -0.05809164,  0.38161196, ...,  0.32207977,
        -0.35920182, -0.06866822]])
```

In [ ]:
```
# HOPKINGS
X_scale = sklearn.preprocessing.scale(X_Scale)
# X = X_scale
pyclustertend.hopkins(X_scale, len(X_scale))
```

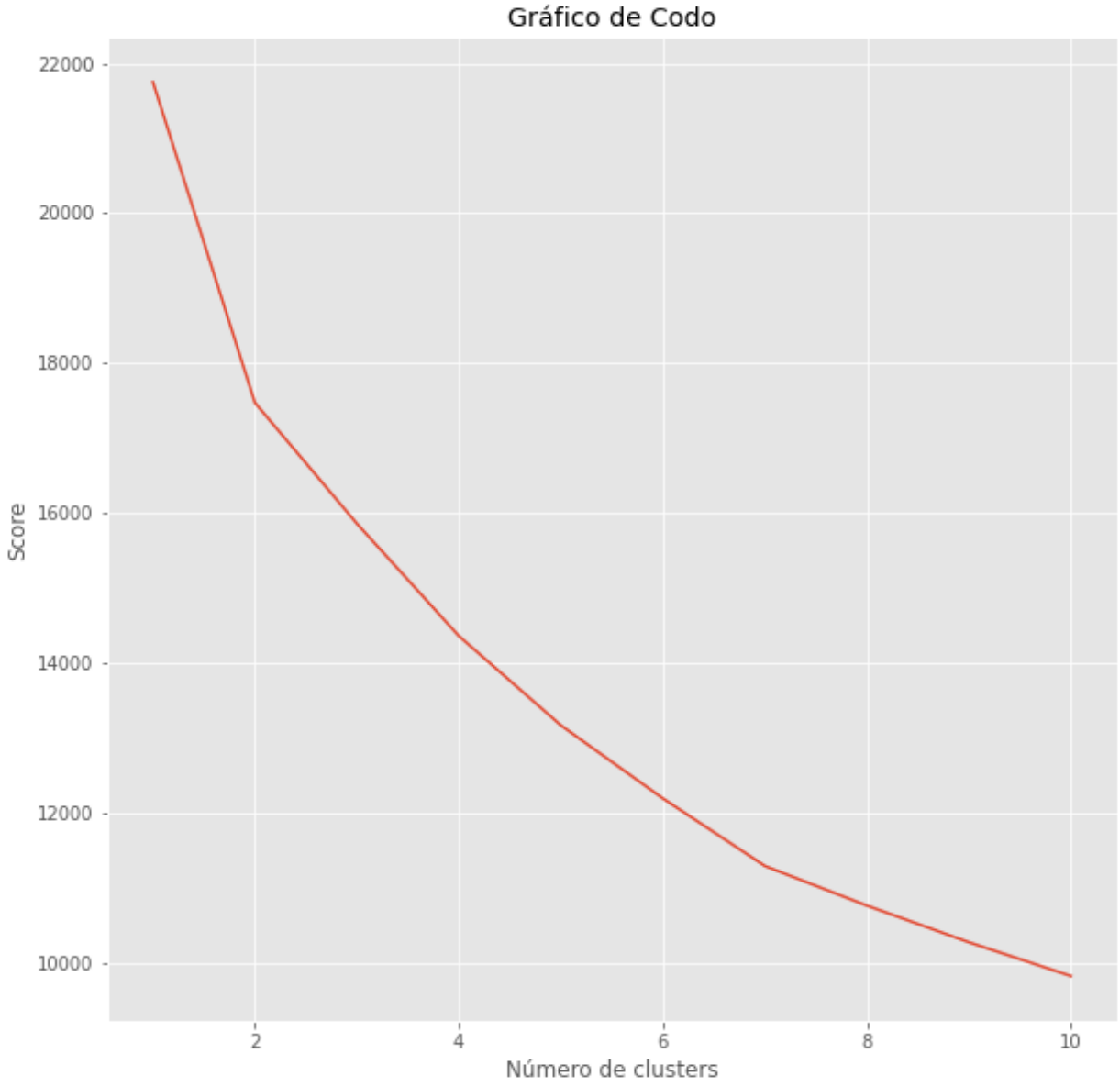
Out[ ]:
```
0.08560970151207026
```

In [ ]:
```
# VAT
pyclustertend.vat(X_Scale)

# devolvemos el SalePrice a su valor original
data['SalePrice'] = train['SalePrice']
```

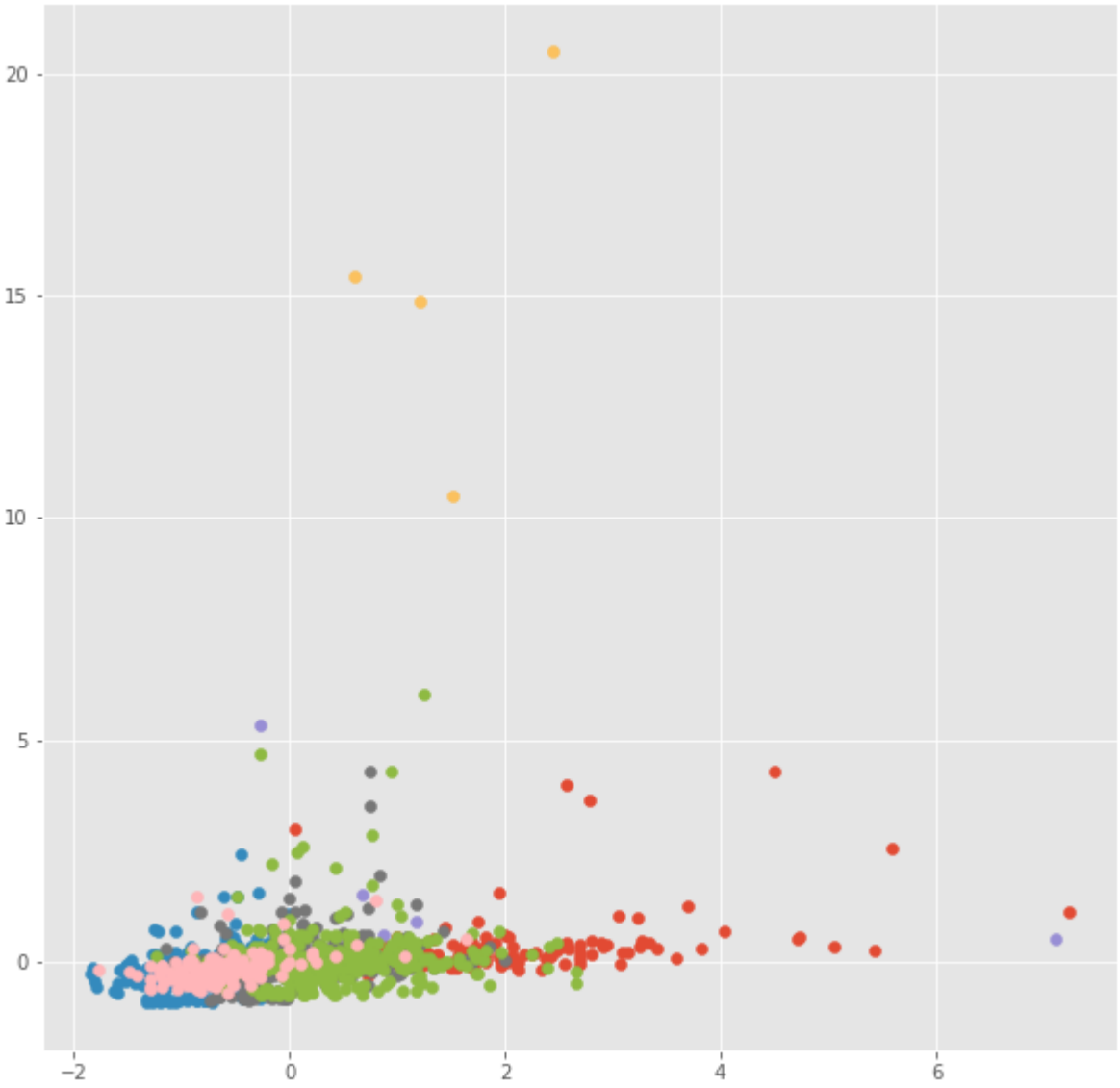
```
<ipython-input-25-9d4f744f424f>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['SalePrice'] = train['SalePrice']
```



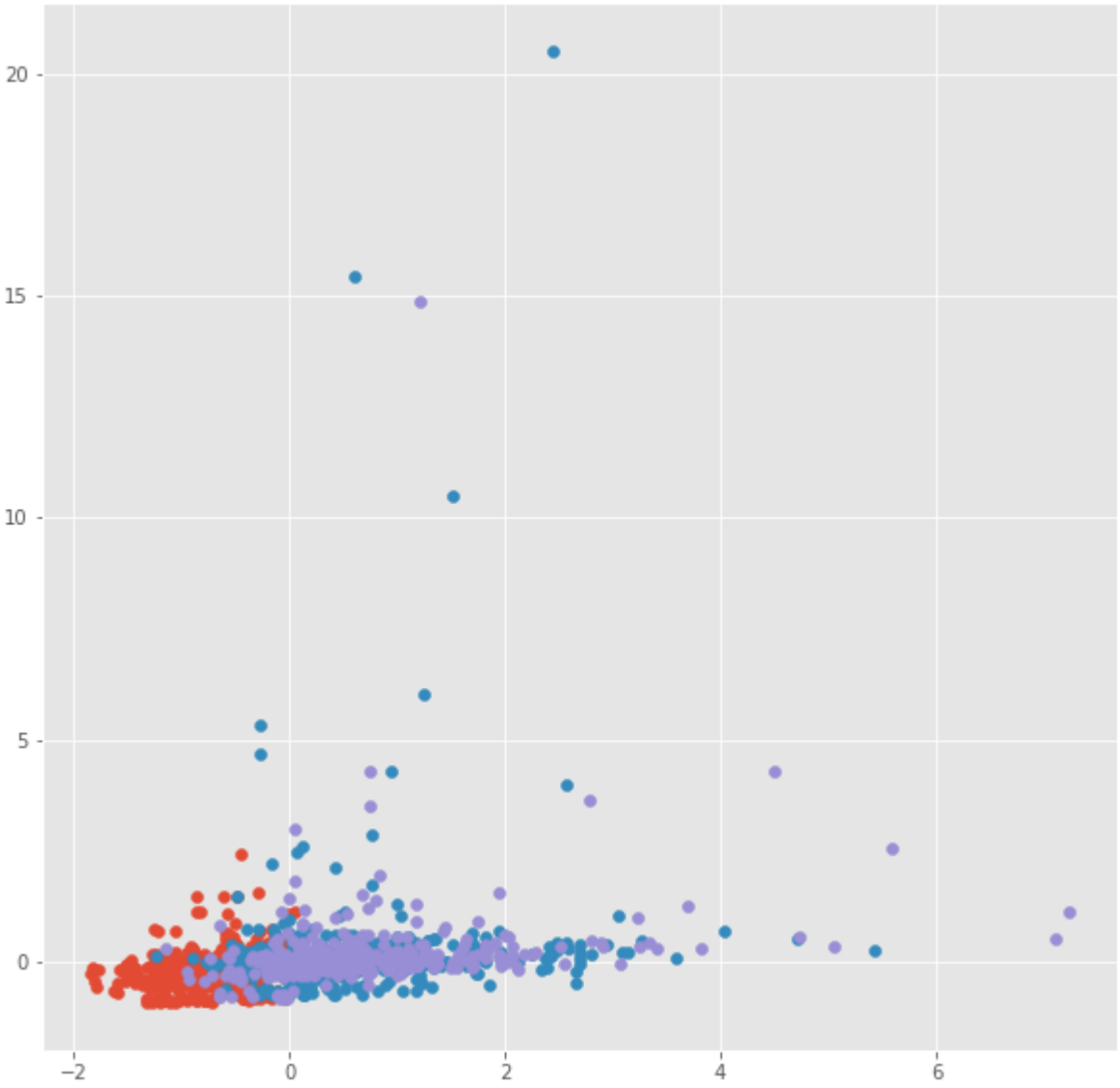
In [ ]:
```
numeroClusters = range(1, 11)
wcss = []
for i in numeroClusters:
    kmeans = cluster.KMeans(n_clusters=i)
    kmeans.fit(X_Scale)
    wcss.append(kmeans.inertia_)

plt.plot(numeroClusters, wcss)
plt.xlabel("Número de clusters")
plt.ylabel("Score")
plt.title("Gráfico de Codo")
plt.show()
```



In [ ]:
```
kmeans = cluster.KMeans(n_clusters=7)
kmeans.fit(X_Scale)
kmeans_result = kmeans.predict(X_Scale)
kmeans_clusters = np.unique(kmeans_result)
for kmeans_cluster in kmeans_clusters:
    # get data points that fall in this cluster
    index = np.where(kmeans_result == kmeans_cluster)
    # make the plot
    plt.scatter(X_Scale[index, 0], X_Scale[index, 1])
plt.show()
```



In [ ]:
```
kmeans = cluster.KMeans(n_clusters=3)
kmeans.fit(X_Scale)
kmeans_result = kmeans.predict(X_Scale)
kmeans_clusters = np.unique(kmeans_result)
for kmeans_cluster in kmeans_clusters:
    # get data points that fall in this cluster
    index = np.where(kmeans_result == kmeans_cluster)
    # make the plot
    plt.scatter(X_Scale[index, 0], X_Scale[index, 1])
plt.show()
```



In [ ]:
```
data['cluster'] = kmeans.labels_
print(data[data['cluster'] == 0].describe().transpose())
print(data[data['cluster'] == 1].describe().transpose())
print(data[data['cluster'] == 2].describe().transpose())
```

	count	mean	std	min	\
SalePrice	712.0	128145.681180	29162.130010	34900.000000	
LotArea	712.0	-0.201612	0.360160	-0.923413	
OverallCond	712.0	0.260448	1.134420	-4.111561	
YearBuilt	712.0	-0.578005	0.854940	-3.286697	
MasVnrArea	712.0	-0.371969	0.463626	-0.572637	
TotalBsmtSF	712.0	-0.517408	0.636344	-2.410341	
1stFlrSF	712.0	-0.539266	0.565876	-2.143438	
2ndFlrSF	712.0	-0.306638	0.696585	-0.794891	
GrLivArea	712.0	-0.649647	0.571413	-2.248350	
TotRmsAbvGrd	712.0	-0.538054	0.732327	-2.779517	
GarageCars	712.0	-0.611161	0.901075	-2.364630	
WoodDeckSF	712.0	-0.288758	0.800101	-0.751918	
OpenPorchSF	712.0	-0.333171	0.795757	-0.704242	
EnclosedPorch	712.0	0.195802	1.143279	-0.359202	
PoolArea	712.0	-0.048533	0.537281	-0.068668	
cluster	712.0	0.000000	0.000000	0.000000	

	25%	50%	75%	max
SalePrice	110000.000000	130000.000000	146000.000000	250000.000000
LotArea	-0.409325	-0.204065	-0.040809	2.417847
OverallCond	-0.517023	0.381612	1.280247	3.077516
YearBuilt	-1.167696	-0.472399	-0.008867	1.249290
MasVnrArea	-0.572637	-0.572637	-0.572637	2.763159
TotalBsmtSF	-0.841520	-0.440910	-0.084178	1.143297
1stFlrSF	-0.896631	-0.565529	-0.189806	1.677170
2ndFlrSF	-0.794891	-0.794891	0.406635	1.988433
GrLivArea	-1.074662	-0.741157	-0.291569	1.603364
TotRmsAbvGrd	-0.933810	-0.318574	-0.318574	2.142369
GarageCars	-1.026506	-1.026506	0.311618	2.987865
WoodDeckSF	-0.751918	-0.751918	0.053898	2.966005
OpenPorchSF	-0.704242	-0.704242	-0.251453	7.189379
EnclosedPorch	-0.359202	-0.359202	-0.359202	5.040088
PoolArea	-0.068668	-0.068668	-0.068668	14.267783
cluster	0.000000	0.000000	0.000000	0.000000

	count	mean	std	min	\
SalePrice	374.0	229383.986631	74213.685680	82500.000000	
LotArea	374.0	0.221159	1.600440	-0.764415	
OverallCond	374.0	-0.360843	0.640864	-3.212926	
YearBuilt	374.0	0.666069	0.595097	-0.770383	
MasVnrArea	374.0	0.343357	1.090019	-0.572637	
TotalBsmtSF	374.0	1.098635	0.897346	-1.076872	
1stFlrSF	374.0	1.127807	0.842065	-0.332723	
2ndFlrSF	374.0	-0.762158	0.213276	-0.794891	
GrLivArea	374.0	0.185446	0.687311	-0.916235	
TotRmsAbvGrd	374.0	0.061424	0.728043	-1.549046	
GarageCars	374.0	0.626471	0.710958	-2.364630	
WoodDeckSF	374.0	0.299499	1.072615	-0.751918	
OpenPorchSF	374.0	0.196251	0.959769	-0.704242	
EnclosedPorch	374.0	-0.241259	0.596283	-0.359202	
PoolArea	374.0	0.006400	1.036535	-0.068668	
cluster	374.0	1.000000	0.000000	1.000000	

	25%	50%	75%	max
SalePrice	178805.000000	210000.000000	262375.000000	611657.000000
LotArea	-0.194347	-0.011705	0.236661	20.511245
OverallCond	-0.517023	-0.517023	-0.517023	3.077516
YearBuilt	0.156680	0.984415	1.149962	1.282400
MasVnrArea	-0.572637	0.023830	0.841210	5.585331
TotalBsmtSF	0.637263	0.995134	1.427087	11.517003

1stFlrSF	0.590741	1.046007	1.435957	9.129553
2ndFlrSF	-0.794891	-0.794891	-0.794891	1.381371
GrLivArea	-0.234002	0.108541	0.399703	7.852884
TotRmsAbvGrd	-0.318574	0.296662	0.296662	3.372840
GarageCars	0.311618	0.311618	1.649742	2.987865
WoodDeckSF	-0.751918	0.301227	0.779930	6.085550
OpenPorchSF	-0.462754	-0.047698	0.529608	5.604618
EnclosedPorch	-0.359202	-0.359202	-0.359202	4.401990
PoolArea	-0.068668	-0.068668	-0.068668	16.059839
cluster	1.000000	1.000000	1.000000	1.000000

	count	mean	std	min	\
SalePrice	366.0	232851.737705	84624.481284	90000.000000	
LotArea	366.0	0.162418	0.977688	-0.841559	
OverallCond	366.0	-0.124177	0.877527	-2.314292	
YearBuilt	366.0	0.423907	0.949135	-3.021822	
MasVnrArea	366.0	0.372749	1.346264	-0.572637	
TotalBsmtSF	366.0	-0.130415	0.759653	-2.410341	
1stFlrSF	366.0	-0.117310	0.842864	-1.726973	
2ndFlrSF	366.0	1.374899	0.553746	-0.794891	
GrLivArea	366.0	1.063936	0.930190	-0.246372	
TotRmsAbvGrd	366.0	0.982498	0.925059	-0.933810	
GarageCars	366.0	0.538295	0.684456	-2.364630	
WoodDeckSF	366.0	0.261139	1.110483	-0.751918	
OpenPorchSF	366.0	0.431442	1.154585	-0.704242	
EnclosedPorch	366.0	-0.135461	0.953535	-0.359202	
PoolArea	366.0	0.089374	1.524546	-0.068668	
cluster	366.0	2.000000	0.000000	2.000000	

	25%	50%	75%	max
SalePrice	179900.000000	212500.000000	262820.000000	755000.000000
LotArea	-0.151968	0.026016	0.243724	14.876188
OverallCond	-0.517023	-0.517023	-0.517023	3.077516
YearBuilt	0.156680	0.818868	1.050634	1.249290
MasVnrArea	-0.572637	-0.329632	1.027606	8.263909
TotalBsmtSF	-0.591352	-0.264254	0.194483	4.742524
1stFlrSF	-0.720733	-0.261588	0.269986	5.109767
2ndFlrSF	1.003388	1.226741	1.714109	3.935614
GrLivArea	0.398752	0.861186	1.536758	6.014566
TotRmsAbvGrd	0.296662	0.911897	1.527133	4.603312
GarageCars	0.311618	0.311618	0.311618	2.987865
WoodDeckSF	-0.751918	0.073844	0.863703	5.056339
OpenPorchSF	-0.296732	0.125871	0.865427	7.551611
EnclosedPorch	-0.359202	-0.359202	-0.359202	8.672338
PoolArea	-0.068668	-0.068668	-0.068668	18.299910
cluster	2.000000	2.000000	2.000000	2.000000

```
<ipython-input-29-aaea25b7312c>: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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  data['cluster'] = kmeans.labels_
```

4. Dependiendo del análisis exploratorio elaborado cree una variable respuesta que le permita clasificar las casas en Económicas, Intermedias o Caras. Los límites de estas clases deben tener un fundamento en

la distribución de los datos de precios, y estar bien explicados.

Se crea la variable de 'Clasificacion' en la cual se clasifica como Economica, Intermedia o Cara. Acorde al precio. Para obtener el rango se crean dos limites. El limite 1 es la media de nuestro salesprice que tienen cluster 0 y para el limit 2 es la media de los que tienen cluster 1. Se realizo de esta forma debido a que como muchos de los precios tenian maximos y minimos en dos categoris se uso la media para colocar los limites.

In [ ]:
```
# Clasificacion de casas en: Economias, Intermedias o Caras.
data.fillna(0)
limit1 = data.query('cluster == 0')['SalePrice'].mean()
limit2 = data.query('cluster == 1')['SalePrice'].mean()
# minPrice = data['SalePrice'].min()
# maxPrice = data['SalePrice'].max()
# division = (maxPrice - minPrice) / 3
data['Clasificacion'] = data['LotArea']

# data['Clasificacion'][data['SalePrice'] < minPrice + division] = 'Economica'
# data['Clasificacion'][data['SalePrice'] >= minPrice + division] = 'Intermedia'
# data['Clasificacion'][data['SalePrice'] >= minPrice + division * 2] = 'Caras'
data.loc[data['SalePrice'] < limit1, 'Clasificacion'] = 'Economica'
data.loc[(data['SalePrice'] >= limit1) & (
    data['SalePrice'] < limit2), 'Clasificacion'] = 'Intermedia'
data.loc[data['SalePrice'] >= limit2, 'Clasificacion'] = 'Caras'
# data.loc[data['cluster'] == 0, 'Clasificacion'] = 'Economica'
# data.loc[data['cluster'] == 1, 'Clasificacion'] = 'Intermedia'
# data.loc[data['cluster'] == 2, 'Clasificacion'] = 'Caras'
```

```
<ipython-input-30-5f65bddb6e29>:8: 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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  data['Clasificacion'] = data['LotArea']
C:\Users\ALIEWARE\AppData\Local\Programs\Python\Python39\lib\site-packages\pandas\cor
e\indexing.py:1817: 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/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(loc, value, pi)
```

Contamos la cantidad de casas por clasificacion

In [ ]:
```
# Obtener cuantos datos hay por cada clasificacion
print(data['Clasificacion'].value_counts())
```

```
Intermedia    814
Economica     342
Caras         296
Name: Clasificacion, dtype: int64
```

5. Divida el set de datos preprocesados en dos conjuntos: Entrenamiento y prueba. Describa el criterio que usó para crear los conjuntos: número de filas de cada uno, estratificado o no, balanceado o no, etc. Si le proveen un conjunto de datos de prueba y tiene suficientes datos, tómelo como de validación, pero haga sus propios conjuntos de prueba.

Se dividen los en dos grupos, entranamiento con 0.7 y prueba con 0.3. Se utiliza el caso normal debido a que evitamos sobreajuste de nuestros datos. Y va a ser nuestra variable clasificacion, X con los demas datos disponibles sin la clasificacion. Se estratifican los datos debdio a que tenemos 3 tipos de clasiicacion Economia, Intermedia y Cara.

Estableciendo los conjuntos de Entrenamiento y Prueba

In [ ]:
```
y = data['Clasificacion']
X = data.drop(['Clasificacion', 'SalePrice'], axis=1)
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, train_size=0.7)
y_train
```

Out[ ]:
```
1179     Economica
1060    Intermedia
692          Caras
1330    Intermedia
436      Economica
           ...
984      Economica
89       Economica
397     Intermedia
382     Intermedia
88       Economica
Name: Clasificacion, Length: 1016, dtype: object
```

70% de entrenamiento y 30% prueba

In [ ]:
```
arbol = DecisionTreeClassifier(max_depth=4, random_state=42)
arbol = arbol.fit(X_train, y_train)
```

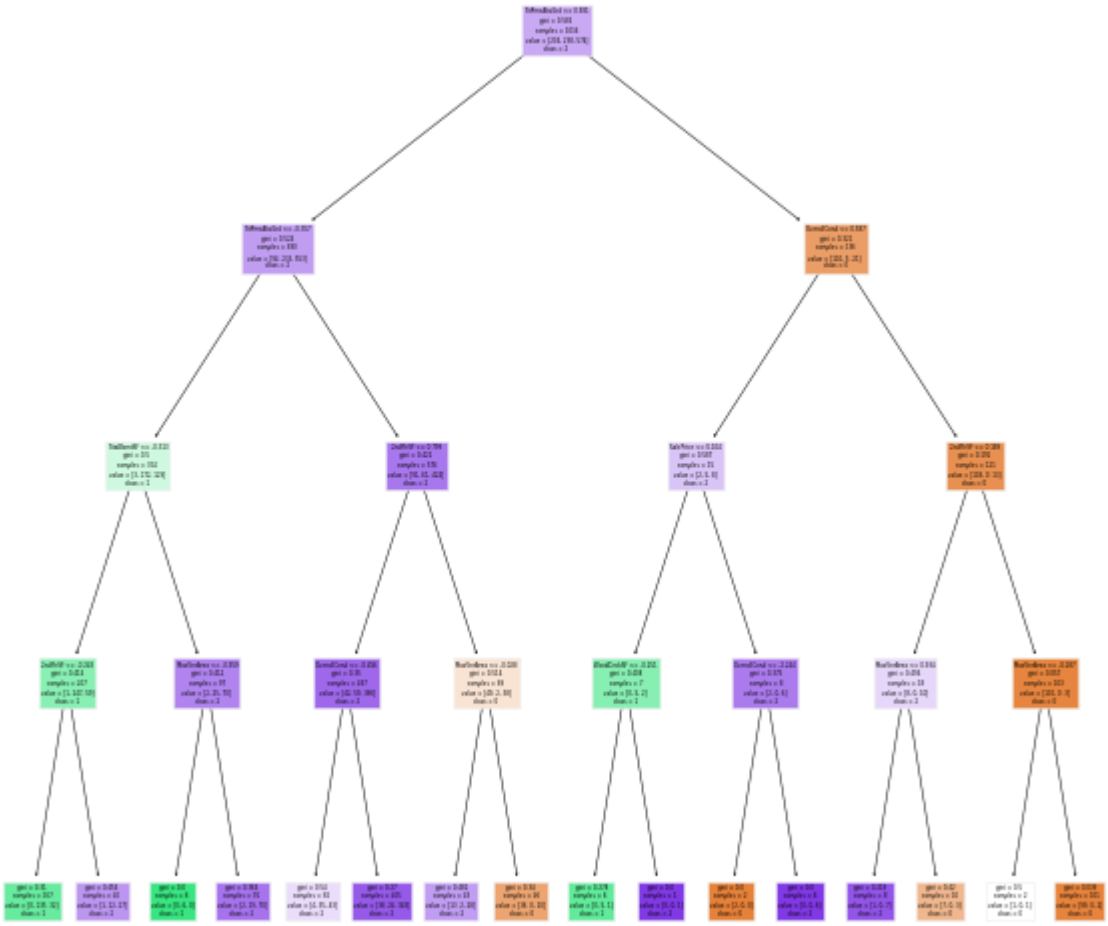
6. Elabore el árbol de clasificación utilizando el conjunto de entrenamiento y la variable respuesta que creó en el punto 4. Explique los resultados a los que llega. Muestre el modelo gráficamente. El experimento debe ser reproducible por lo que debe

fijar que los conjuntos de entrenamiento y prueba sean los mismos siempre que se ejecute el código.

In [ ]:
```
tree.plot_tree(arbol, feature_names=data.columns,
               class_names=['0', '1', '2'], filled=True)
```

Out[ ]:
```
[Text(279.0, 489.24, 'TotRmsAbvGrd <= 0.981\ngini = 0.586\nsamples = 1016\nvalue = [204, 238, 574]\nclass = 2'),
 Text(139.5, 380.52000000000004, 'TotRmsAbvGrd <= -0.357\ngini = 0.524\nsamples = 880\nvalue = [94, 233, 553]\nclass = 2'),
 Text(69.75, 271.8, 'TotalBsmtSF <= -0.313\ngini = 0.5\nsamples = 304\nvalue = [3, 172, 129]\nclass = 1'),
 Text(34.875, 163.08000000000004, '2ndFlrSF <= -0.243\ngini = 0.414\nsamples = 207\nvalue = [1, 147, 59]\nclass = 1'),
 Text(17.4375, 54.360000000000014, 'gini = 0.31\nsamples = 167\nvalue = [0, 135, 32]\nclass = 1'),
 Text(52.3125, 54.360000000000014, 'gini = 0.454\nsamples = 40\nvalue = [1, 12, 27]\nclass = 2'),
 Text(104.625, 163.08000000000004, 'MasVnrArea <= -0.959\ngini = 0.412\nsamples = 97\nvalue = [2, 25, 70]\nclass = 2'),
 Text(87.1875, 54.360000000000014, 'gini = 0.0\nsamples = 6\nvalue = [0, 6, 0]\nclass = 1'),
 Text(122.0625, 54.360000000000014, 'gini = 0.364\nsamples = 91\nvalue = [2, 19, 70]\nclass = 2'),
 Text(209.25, 271.8, '2ndFlrSF <= 0.799\ngini = 0.422\nsamples = 576\nvalue = [91, 61, 424]\nclass = 2'),
 Text(174.375, 163.08000000000004, 'OverallCond <= -0.456\ngini = 0.35\nsamples = 487\nvalue = [42, 59, 386]\nclass = 2'),
 Text(156.9375, 54.360000000000014, 'gini = 0.54\nsamples = 82\nvalue = [4, 35, 43]\nclass = 2'),
 Text(191.8125, 54.360000000000014, 'gini = 0.27\nsamples = 405\nvalue = [38, 24, 343]\nclass = 2'),
 Text(244.125, 163.08000000000004, 'MasVnrArea <= -0.028\ngini = 0.514\nsamples = 89\nvalue = [49, 2, 38]\nclass = 0'),
 Text(226.6875, 54.360000000000014, 'gini = 0.482\nsamples = 43\nvalue = [13, 2, 28]\nclass = 2'),
 Text(261.5625, 54.360000000000014, 'gini = 0.34\nsamples = 46\nvalue = [36, 0, 10]\nclass = 0'),
 Text(418.5, 380.52000000000004, 'OverallCond <= 0.587\ngini = 0.321\nsamples = 136\nvalue = [110, 5, 21]\nclass = 0'),
 Text(348.75, 271.8, 'SalePrice <= 0.104\ngini = 0.587\nsamples = 15\nvalue = [2, 5, 8]\nclass = 2'),
 Text(313.875, 163.08000000000004, 'WoodDeckSF <= -0.251\ngini = 0.408\nsamples = 7\nvalue = [0, 5, 2]\nclass = 1'),
 Text(296.4375, 54.360000000000014, 'gini = 0.278\nsamples = 6\nvalue = [0, 5, 1]\nclass = 1'),
 Text(331.3125, 54.360000000000014, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass = 2'),
 Text(383.625, 163.08000000000004, 'OverallCond <= -2.244\ngini = 0.375\nsamples = 8\nvalue = [2, 0, 6]\nclass = 2'),
 Text(366.1875, 54.360000000000014, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0]\nclass = 0'),
 Text(401.0625, 54.360000000000014, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 6]\nclass = 2'),
 Text(488.25, 271.8, '2ndFlrSF <= 0.189\ngini = 0.192\nsamples = 121\nvalue = [108, 0, 13]\nclass = 0'),
 Text(453.375, 163.08000000000004, 'MasVnrArea <= 0.934\ngini = 0.494\nsamples = 18\nvalue = [8, 0, 10]\nclass = 2'),
 Text(435.9375, 54.360000000000014, 'gini = 0.219\nsamples = 8\nvalue = [1, 0, 7]\nclass = 2'),
 Text(470.8125, 54.360000000000014, 'gini = 0.42\nsamples = 10\nvalue = [7, 0, 3]\nclass = 0'),
 Text(523.125, 163.08000000000004, 'MasVnrArea <= -0.287\ngini = 0.057\nsamples = 103\nvalue = [100, 0, 3]\nclass = 0'),
 Text(505.6875, 54.360000000000014, 'gini = 0.5\nsamples = 2\nvalue = [1, 0, 1]\nclass = 0'),
```

```
 Text(540.5625, 54.360000000000014, 'gini = 0.039\nsamples = 101\nvalue = [99, 0, 2]
\nclass = 0')]
```



In [ ]:
```
y_pred = arbol.predict(X_test)
print(y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
print("Precision:", metrics.precision_score(
    y_test, y_pred, average='weighted'))
print("Recall: ", metrics.recall_score(y_test, y_pred, average='weighted'))
```

```
['Economica' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Caras' 'Economica'
 'Intermedia' 'Economica' 'Caras' 'Caras' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Economica' 'Economica' 'Intermedia'
 'Caras' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Caras'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Economica' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Caras'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Caras' 'Intermedia' 'Economica' 'Intermedia'
 'Economica' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Economica' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Economica'
 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Economica'
 'Intermedia' 'Caras' 'Economica' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Caras' 'Economica' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Caras'
 'Caras' 'Economica' 'Intermedia' 'Caras' 'Economica' 'Economica'
 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Economica' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Caras' 'Intermedia' 'Intermedia'
```

```
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Economica' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Economica' 'Economica' 'Caras' 'Intermedia'
 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Economica' 'Economica' 'Caras' 'Caras' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica']
Accuracy: 0.7545871559633027
Precision: 0.7648327110810091
Recall:  0.7545871559633027
```

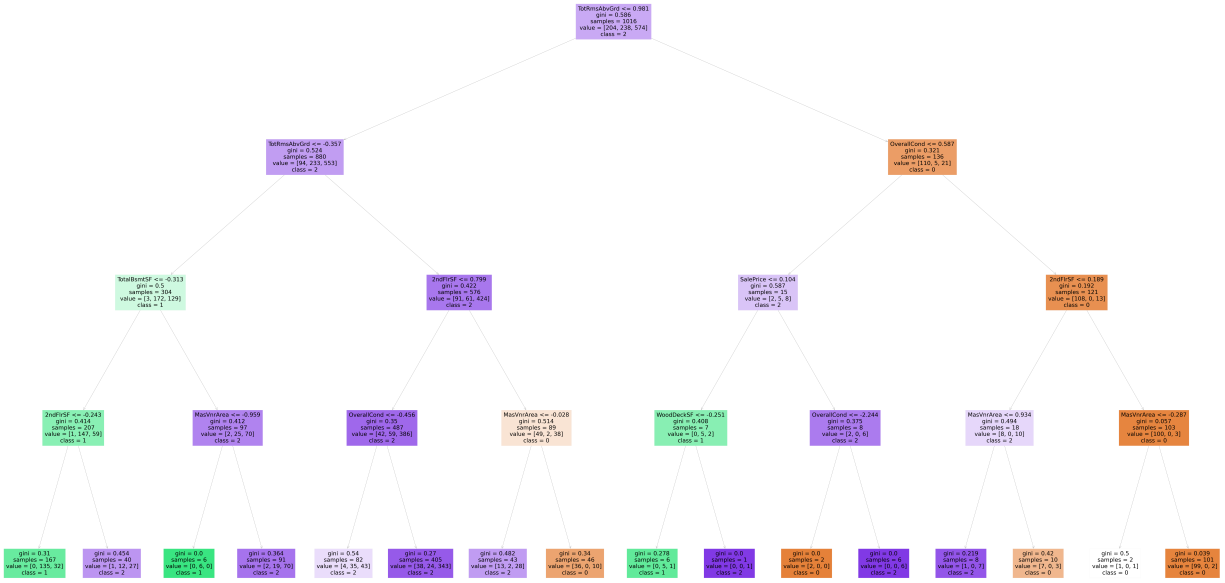
7.Elabore el árbol de regresión para predecir el precio de las viviendas utilizando el conjunto

de entrenamiento. Explique los resultados a los que llega. Muestre el modelo gráficamente. El experimento debe ser reproducible por lo que debe fijar que los conjuntos de entrenamiento y prueba sean los mismos siempre que se ejecute el código.

In [ ]:
```
plt.rcParams['figure.figsize'] = (160, 90)
regressionTree = DecisionTreeRegressor(max_depth=4, random_state=42)
regressionTree = arbol.fit(X_train, y_train)
tree.plot_tree(regressionTree, feature_names=data.columns,
               class_names=['0', '1', '2'], filled=True)
```

Out[ ]:
```
[Text(4464.0, 4403.16, 'TotRmsAbvGrd <= 0.981\ngini = 0.586\nsamples = 1016\nvalue =
[204, 238, 574]\nclass = 2'),
 Text(2232.0, 3424.68, 'TotRmsAbvGrd <= -0.357\ngini = 0.524\nsamples = 880\nvalue =
[94, 233, 553]\nclass = 2'),
 Text(1116.0, 2446.2, 'TotalBsmtSF <= -0.313\ngini = 0.5\nsamples = 304\nvalue = [3,
172, 129]\nclass = 1'),
 Text(558.0, 1467.7199999999998, '2ndFlrSF <= -0.243\ngini = 0.414\nsamples = 207\nva
lue = [1, 147, 59]\nclass = 1'),
 Text(279.0, 489.2399999999998, 'gini = 0.31\nsamples = 167\nvalue = [0, 135, 32]\ncl
ass = 1'),
 Text(837.0, 489.2399999999998, 'gini = 0.454\nsamples = 40\nvalue = [1, 12, 27]\ncla
ss = 2'),
 Text(1674.0, 1467.7199999999998, 'MasVnrArea <= -0.959\ngini = 0.412\nsamples = 97\n
value = [2, 25, 70]\nclass = 2'),
 Text(1395.0, 489.2399999999998, 'gini = 0.0\nsamples = 6\nvalue = [0, 6, 0]\nclass =
1'),
 Text(1953.0, 489.2399999999998, 'gini = 0.364\nsamples = 91\nvalue = [2, 19, 70]\ncl
ass = 2'),
 Text(3348.0, 2446.2, '2ndFlrSF <= 0.799\ngini = 0.422\nsamples = 576\nvalue = [91, 6
1, 424]\nclass = 2'),
 Text(2790.0, 1467.7199999999998, 'OverallCond <= -0.456\ngini = 0.35\nsamples = 487
\nvalue = [42, 59, 386]\nclass = 2'),
 Text(2511.0, 489.2399999999998, 'gini = 0.54\nsamples = 82\nvalue = [4, 35, 43]\ncla
ss = 2'),
 Text(3069.0, 489.2399999999998, 'gini = 0.27\nsamples = 405\nvalue = [38, 24, 343]\n
class = 2'),
 Text(3906.0, 1467.7199999999998, 'MasVnrArea <= -0.028\ngini = 0.514\nsamples = 89\n
value = [49, 2, 38]\nclass = 0'),
 Text(3627.0, 489.2399999999998, 'gini = 0.482\nsamples = 43\nvalue = [13, 2, 28]\ncl
ass = 2'),
 Text(4185.0, 489.2399999999998, 'gini = 0.34\nsamples = 46\nvalue = [36, 0, 10]\ncla
ss = 0'),
 Text(6696.0, 3424.68, 'OverallCond <= 0.587\ngini = 0.321\nsamples = 136\nvalue = [1
10, 5, 21]\nclass = 0'),
 Text(5580.0, 2446.2, 'SalePrice <= 0.104\ngini = 0.587\nsamples = 15\nvalue = [2, 5,
8]\nclass = 2'),
 Text(5022.0, 1467.7199999999998, 'WoodDeckSF <= -0.251\ngini = 0.408\nsamples = 7\nv
alue = [0, 5, 2]\nclass = 1'),
 Text(4743.0, 489.2399999999998, 'gini = 0.278\nsamples = 6\nvalue = [0, 5, 1]\nclass
= 1'),
 Text(5301.0, 489.2399999999998, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1]\nclass =
2'),
 Text(6138.0, 1467.7199999999998, 'OverallCond <= -2.244\ngini = 0.375\nsamples = 8\n
value = [2, 0, 6]\nclass = 2'),
 Text(5859.0, 489.2399999999998, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0]\nclass =
0'),
 Text(6417.0, 489.2399999999998, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 6]\nclass =
2'),
 Text(7812.0, 2446.2, '2ndFlrSF <= 0.189\ngini = 0.192\nsamples = 121\nvalue = [108,
0, 13]\nclass = 0'),
 Text(7254.0, 1467.7199999999998, 'MasVnrArea <= 0.934\ngini = 0.494\nsamples = 18\nv
alue = [8, 0, 10]\nclass = 2'),
 Text(6975.0, 489.2399999999998, 'gini = 0.219\nsamples = 8\nvalue = [1, 0, 7]\nclass
= 2'),
 Text(7533.0, 489.2399999999998, 'gini = 0.42\nsamples = 10\nvalue = [7, 0, 3]\nclass
= 0'),
 Text(8370.0, 1467.7199999999998, 'MasVnrArea <= -0.287\ngini = 0.057\nsamples = 103
\nvalue = [100, 0, 3]\nclass = 0'),
 Text(8091.0, 489.2399999999998, 'gini = 0.5\nsamples = 2\nvalue = [1, 0, 1]\nclass =
0'),
```

```
 Text(8649.0, 489.2399999999998, 'gini = 0.039\nsamples = 101\nvalue = [99, 0, 2]\ncl
ass = 0')]
```



8.Utilice el modelo con el conjunto de prueba y determine la eficiencia del algoritmo para clasificar y predecir, en dependencia de las características de la variable respuesta.

In [ ]:
```
y_pred = arbol.predict(X_test)
print(y_pred)
print("Exactitud:", metrics.accuracy_score(y_test, y_pred))
print("Precision:", metrics.precision_score(
    y_test, y_pred, average='weighted'))
print("Recall: ", metrics.recall_score(y_test, y_pred, average='weighted'))
```

```
['Economica' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Caras' 'Economica'
 'Intermedia' 'Economica' 'Caras' 'Caras' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Economica' 'Economica' 'Intermedia'
 'Caras' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Caras'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Economica' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Caras'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Caras' 'Intermedia' 'Economica' 'Intermedia'
 'Economica' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Economica' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Economica'
 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Economica'
 'Intermedia' 'Caras' 'Economica' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Caras' 'Economica' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Caras' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica' 'Caras'
 'Caras' 'Economica' 'Intermedia' 'Caras' 'Economica' 'Economica'
 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Economica' 'Economica' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Caras' 'Intermedia' 'Intermedia'
```

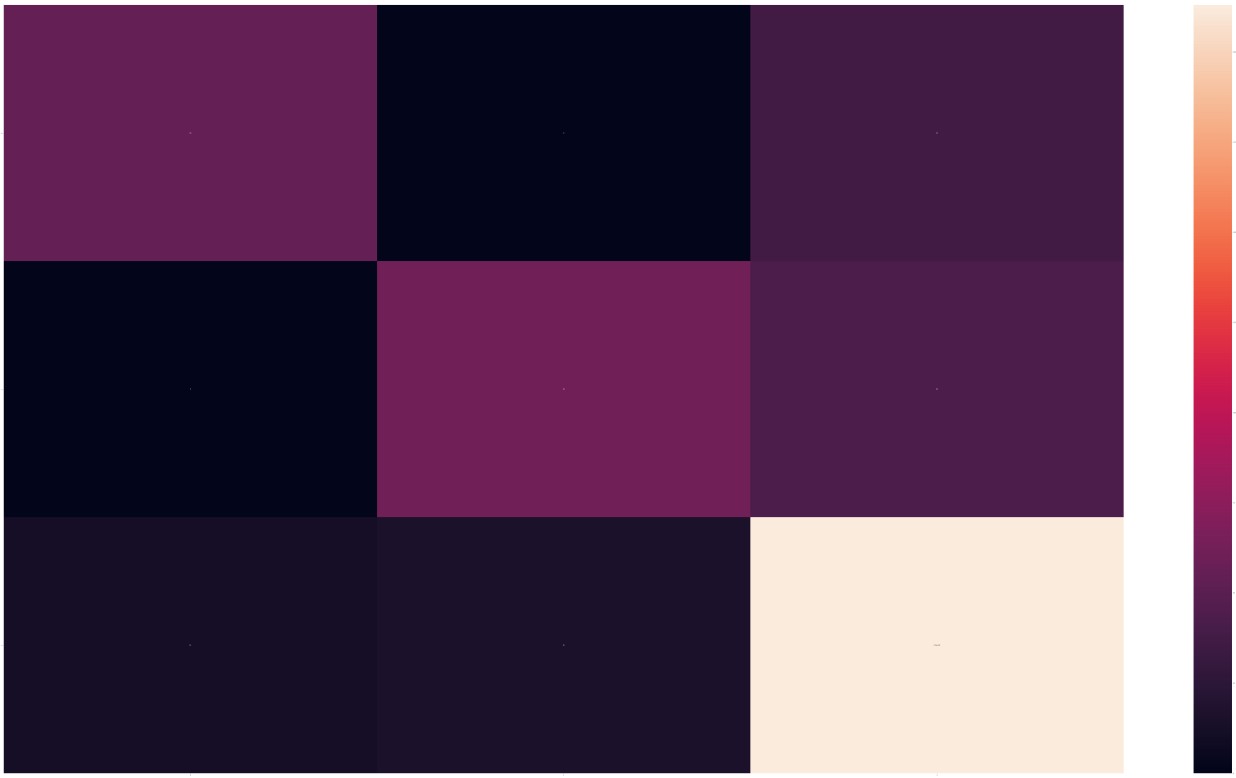
```
 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Economica' 'Economica'
 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Intermedia' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica' 'Economica' 'Intermedia' 'Intermedia'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Intermedia'
 'Economica' 'Intermedia' 'Intermedia' 'Economica' 'Intermedia' 'Caras'
 'Intermedia' 'Intermedia' 'Intermedia' 'Caras' 'Intermedia' 'Economica'
 'Intermedia' 'Intermedia' 'Economica' 'Economica' 'Caras' 'Intermedia'
 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Economica'
 'Caras' 'Intermedia' 'Intermedia' 'Intermedia' 'Intermedia' 'Caras'
 'Intermedia' 'Economica' 'Economica' 'Caras' 'Caras' 'Caras' 'Intermedia'
 'Intermedia' 'Caras' 'Economica' 'Intermedia' 'Intermedia' 'Intermedia'
 'Intermedia' 'Economica']
Exactitud: 0.7545871559633027
Precision: 0.7648327110810091
Recall:  0.7545871559633027
```

9.Haga un análisis de la eficiencia del algoritmo usando una matriz de confusión para el árbol de clasificación. Tenga en cuenta la efectividad, donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores

In [ ]:
```
confussion_matrix = metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
print(confussion_matrix)
sns.heatmap(confussion_matrix, annot=True)
```

```
[[ 55   0  37]
 [  0  61  43]
 [ 12  15 213]]
```

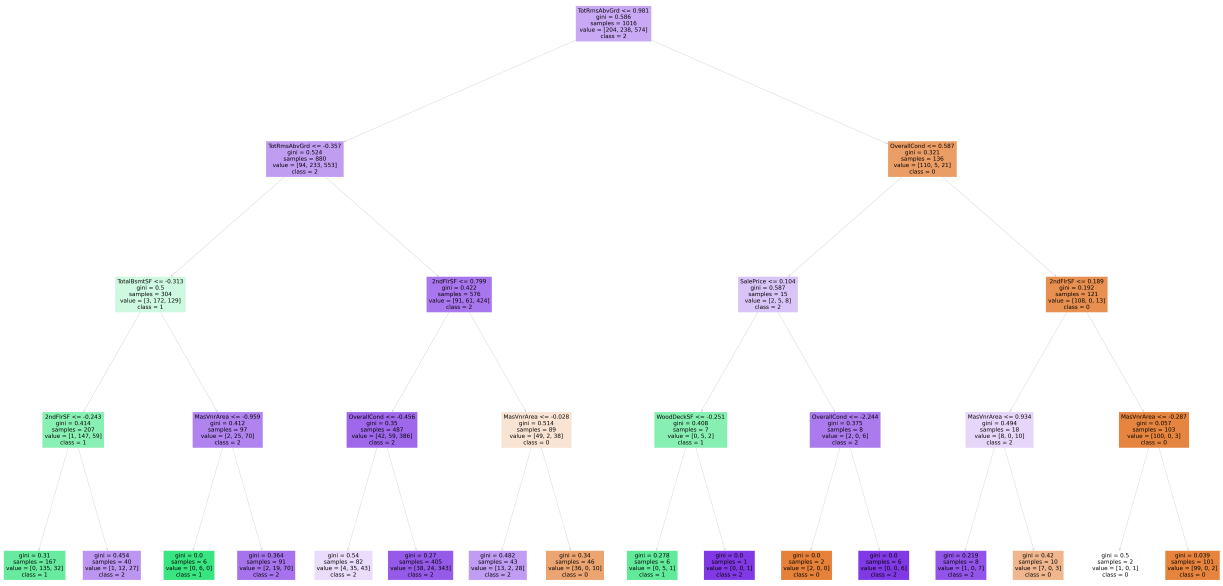
Out[ ]:
```
<AxesSubplot:>
```



10.Analice el desempeño del árbol de regresión.

In [ ]:
```
tree.plot_tree(regressionTree, feature_names=data.columns,
               class_names=['0', '1', '2'], filled=True)
print('En el arbol de regresion elaborado en la pregunta 7, para predecir el precio de
```

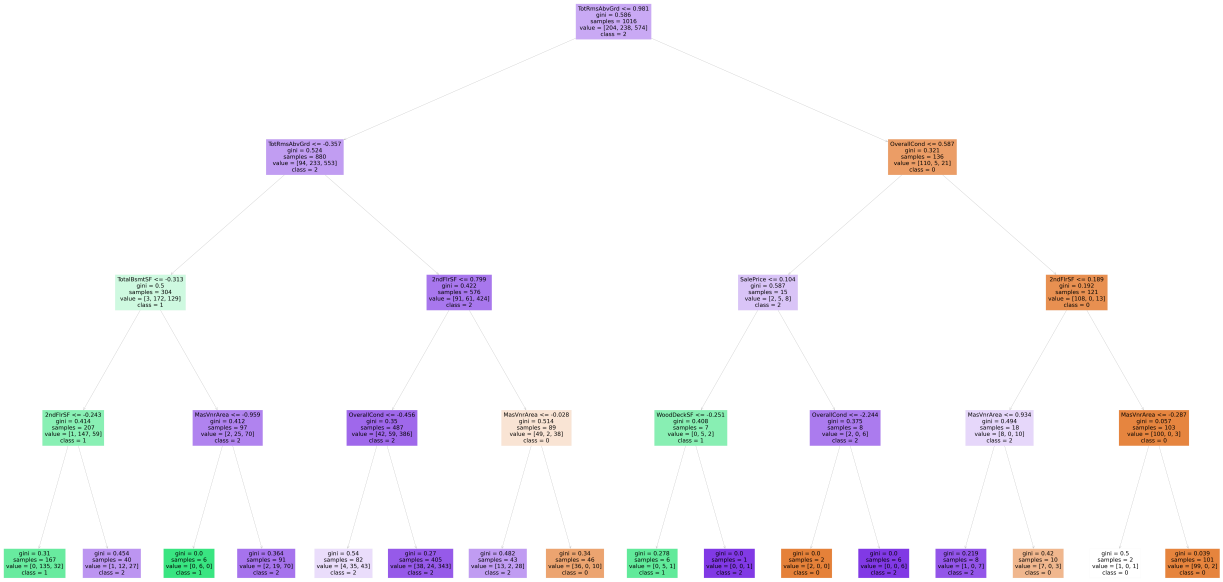
En el arbol de regresion elaborado en la pregunta 7, para predecir el precio de las v
iviendas, podemos observar como se ramifican las diferentes alternativas de precios p
osibles a partir de las variables seleccionadas. Lo que nos muestra el arbol son las
diferentes opciones de precios en el mercado, los cuales se clasifican segun el conju
nto de variables, como se menciono anteriormente, tambien podemos observar el indice
gini en cada uno de los nodos, el cual mide el grado o la probabilidad de que una var
iable en particular se clasifique incorrectamente cuando se elige al azar y podemos o
bservar que se mantiene en valores bajos



11.Repita los análisis usando random forest como algoritmo de predicción, explique sus resultados comparando ambos algoritmos.

In [ ]:
```
randomForest = RandomForestClassifier(max_depth=4, random_state=42)
randomForest = arbol.fit(X_train, y_train)

tree.plot_tree(randomForest, feature_names=data.columns,
               class_names=['0', '1', '2'], filled=True)

y_pred = randomForest.predict(X_test)
print("Exactitud:", metrics.accuracy_score(y_test, y_pred))
print("Precision:", metrics.precision_score(
    y_test, y_pred, average='weighted'))
print("Recall: ", metrics.recall_score(y_test, y_pred, average='weighted'))
print("Comparación: Se puede observar que el de random forest nos da un análisis más e
```

```
Exactitud: 0.7545871559633027
Precision: 0.7648327110810091
Recall:  0.7545871559633027
Comparación: Se puede observar que el de random forest nos da un análisis más extenso
y exacto, pues crea una serie de árboles para analizar un conjunto de datos de entren
amiento. los resultados obtenidos se combinan a fin de obtener un modelo único más ro
busto en comparación con los resultados de cada árbol por separado obtenniendo mejore
s resultados.
```


TotRmsAbvGrd <= 0.981
gini = 0.586
samples = 1016
value = [204, 238, 574]
class = 2
TotRmsAbvGrd <= -0.357
gini = 0.524
samples = 880
value = [94, 233, 553]
class = 2
OverallCond <= 0.587
gini = 0.327
samples = 136
value = [110, 5, 21]
class = 0
TotalBsmtSF <= -0.313
gini = 0.5
samples = 304
value = [3, 172, 129]
class = 1
2ndFlrSF <= 0.799
gini = 0.422
samples = 576
value = [91, 61, 424]
class = 2
SalePrice <= 0.104
gini = 0.587
samples = 15
value = [2, 5, 8]
class = 2
2ndFlrSF <= 0.189
gini = 0.192
samples = 121
value = [108, 0, 13]
class = 0
2ndFlrSF <= -0.243
gini = 0.414
samples = 207
value = [1, 147, 59]
class = 1
MasVnrArea <= -0.959
gini = 0.412
samples = 97
value = [2, 25, 70]
class = 2
OverallCond <= -0.450
gini = 0.39
samples = 487
value = [42, 59, 386]
class = 2
MasVnrArea <= -0.028
gini = 0.514
samples = 89
value = [49, 2, 38]
class = 0
WoodDeckSF <= -0.251
gini = 0.408
samples = 7
value = [0, 5, 2]
class = 1
OverallCond <= -2.244
gini = 0.375
samples = 8
value = [2, 0, 6]
class = 2
MasVnrArea <= 0.934
gini = 0.494
samples = 18
value = [8, 0, 10]
class = 2
MasVnrArea <= -0.287
gini = 0.057
samples = 103
value = [100, 0, 3]
class = 0
gini = 0.31
samples = 167
value = [0, 135, 32]
class = 1
gini = 0.456
samples = 40
value = [1, 12, 27]
class = 2
gini = 0.0
samples = 6
value = [0, 6, 0]
class = 1
gini = 0.364
samples = 91
value = [2, 19, 70]
class = 2
gini = 0.54
samples = 82
value = [4, 35, 43]
class = 2
gini = 0.27
samples = 405
value = [38, 24, 343]
class = 2
gini = 0.482
samples = 43
value = [13, 2, 28]
class = 2
gini = 0.34
samples = 46
value = [36, 0, 10]
class = 0
gini = 0.278
samples = 6
value = [0, 5, 1]
class = 1
gini = 0.0
samples = 1
value = [0, 0, 1]
class = 2
gini = 0.0
samples = 2
value = [2, 0, 0]
class = 0
gini = 0.0
samples = 6
value = [0, 0, 6]
class = 2
gini = 0.219
samples = 8
value = [1, 0, 7]
class = 2
gini = 0.42
samples = 10
value = [7, 0, 3]
class = 0
gini = 0.5
samples = 2
value = [1, 0, 1]
class = 0
gini = 0.039
samples = 101
value = [99, 0, 2]
class = 0