Práctica M33

- Descarga el archivo House Pricing.csv. En la sección de Anexos.
- Verifica por medio de pruebas analíticas y visuales como es la distribución de las diferentes variables numéricas que se encuentran en el dataset como "Salesprice", "GrLivArea", ""2ndFlrSF". Calculando algunas métricas importantes como la media, la desviación estándar y sus cuártiles.

In [11]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import (kstest, shapiro, normaltest, jarque_bera, cramervonmises, anderson, chisquare)
from statsmodels.stats.diagnostic import lilliefors
from scipy.stats import (norm, uniform, triang, expon, arcsine, gamma)
from collections import namedtuple
from tabulate import tabulate
import scipy.stats as stats
```

In [27]:
```python
df = pd.read_csv('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M33 DS/House Pricing.csv')
df
```

Out[27]:

Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSol
1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	
1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	2500	
1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

ows × 81 columns

In [14]:
```python
# verificamos valores nulos
print('valores nulos por columna')
print(df.isnull().sum())

# valores infinitos
print('\nvalores infinitos por columna')
print(df.replace([np.inf, -np.inf], np.nan).isnull().sum())
```

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

valores infinitos por columna
Id               0
MSSubClass       0
MSZoning         0
```

In [30]:
```python
df.dropna(axis=1, inplace = True)
```

In [49]:
```python
df.head()
```

Out[49]:

ndContour	Utilities	LotConfig	LandSlope	...	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold	SaleType	SaleCondition	SalePrice
Lvl	AllPub	Inside	Gtl	...	0	0	0	0	0	2	2008	WD	Normal	208500
Lvl	AllPub	FR2	Gtl	...	0	0	0	0	0	5	2007	WD	Normal	181500
Lvl	AllPub	Inside	Gtl	...	0	0	0	0	0	9	2008	WD	Normal	223500
Lvl	AllPub	Corner	Gtl	...	272	0	0	0	0	2	2006	WD	Abnorml	140000
Lvl	AllPub	FR2	Gtl	...	0	0	0	0	0	12	2008	WD	Normal	250000

In [51]:
```python
# Seleccionar todas las columnas numéricas
columns_num = df.select_dtypes(include = [np.number]).columns.tolist()
```

In [52]:
```python
# calcular Metricas
df[columns_num].describe()
```

Out[52]:

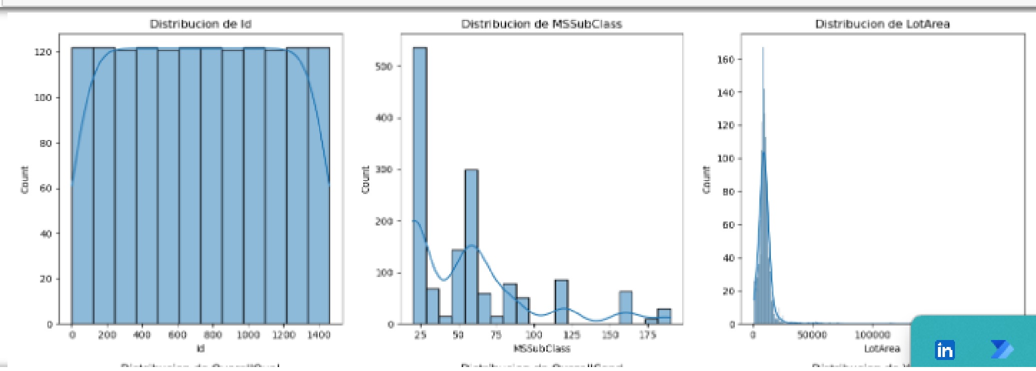
	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	...	WoodDeck
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	...	1460.000
mean	730.500000	56.897260	10516.828082	6.099315	5.575342	1971.267808	1984.865753	443.639726	46.549315	567.240411	...	94.244
std	421.610009	42.300571	9981.264932	1.382997	1.112799	30.202904	20.645407	456.098091	161.319273	441.866955	...	125.338
min	1.000000	20.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	0.000000	...	0.000
25%	365.750000	20.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	223.000000	...	0.000
50%	730.500000	50.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	383.500000	0.000000	477.500000	...	0.000
75%	1095.250000	70.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	712.250000	0.000000	808.000000	...	168.000
max	1460.000000	190.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	5644.000000	1474.000000	2336.000000	...	857.000

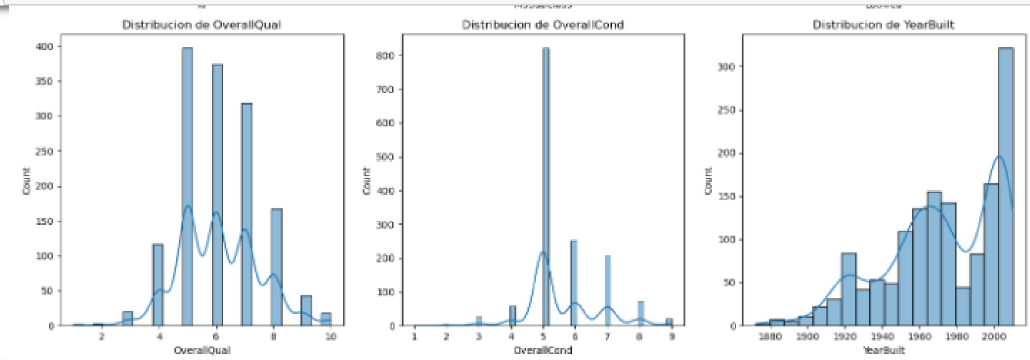
8 rows × 35 columns

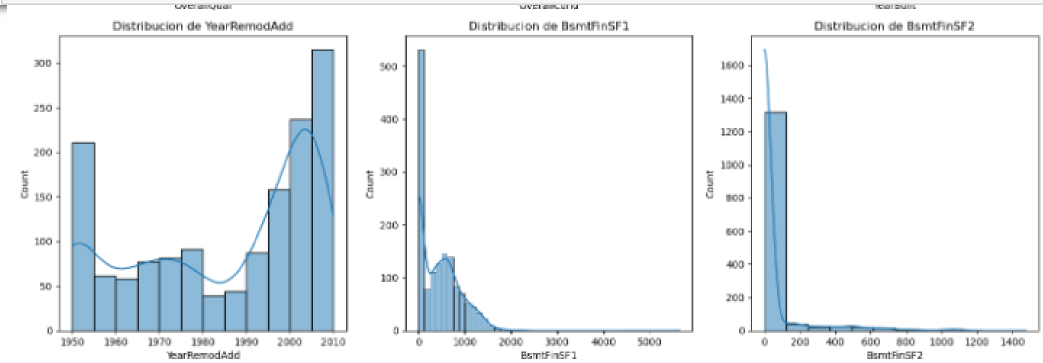
In [54]:
```python
# cantidad de variables numericas
num_vars = len(columns_num)

# Ajustar el tamaño de la cuadrícula de subplots
nrows = (num_vars // 3) + (1 if num_vars % 3 !=0 else 0)

# visualizacion de la distribucion
plt.figure(figsize = (15,5 * nrows))
for i, var in enumerate(columns_num):
    plt.subplot(nrows, 3, i+1)
    sns.histplot(df[var], kde = True)
    plt.title(f'Distribucion de {var}')
plt.tight_layout()
plt.show()
```







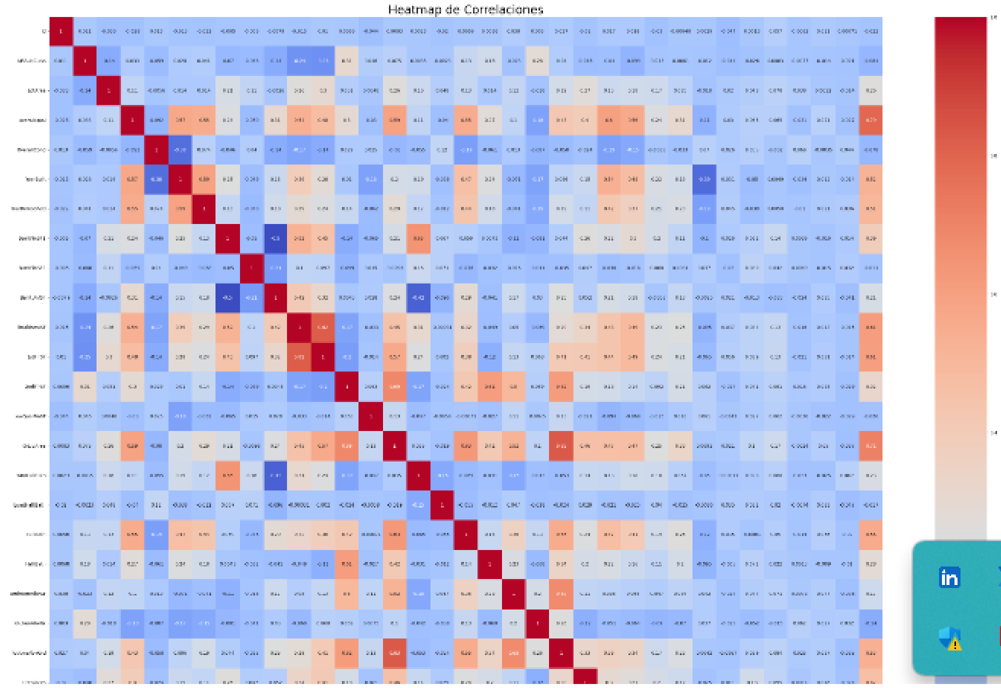
In [55]:
```
# calculamos la correlacion
correlacion_var = df[columns_num].corr()
correlacion_var
```

Out[55]:

	Id	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	BsmtFinSF1	BsmtFinSF2	BsmtUnfSF	...	WoodDeckSF	OpenPorc
Id	1.000000	0.011156	-0.033226	-0.028365	0.012609	-0.012713	-0.021998	-0.005024	-0.005968	-0.007940	...	-0.029643	-0.00
Class	0.011156	1.000000	-0.139781	0.032628	-0.059316	0.027850	0.040581	-0.069836	-0.065649	-0.140759	...	-0.012579	-0.00
tArea	-0.033226	-0.139781	1.000000	0.105806	-0.005636	0.014228	0.013788	0.214103	0.111170	-0.002618	...	0.171698	0.08
lQual	-0.028365	0.032628	0.105806	1.000000	-0.091932	0.572323	0.550684	0.239666	-0.059119	0.308159	...	0.238923	0.30
Cond	0.012609	-0.059316	-0.005636	-0.091932	1.000000	-0.375983	0.073741	-0.046231	0.040229	-0.136841	...	-0.003334	-0.03
rBuilt	-0.012713	0.027850	0.014228	0.572323	-0.375983	1.000000	0.592855	0.249503	-0.049107	0.149040	...	0.224880	0.18
dAdd	-0.021998	0.040581	0.013788	0.550684	0.073741	0.592855	1.000000	0.128451	-0.067759	0.181133	...	0.205726	0.22
nSF1	-0.005024	-0.069836	0.214103	0.239666	-0.046231	0.249503	0.128451	1.000000	-0.050117	-0.495251	...	0.204306	0.11
nSF2	-0.005968	-0.065649	0.111170	-0.059119	0.040229	-0.049107	-0.067759	-0.050117	1.000000	-0.209294	...	0.067898	0.00
JnfSF	-0.007940	-0.140759	-0.002618	0.308159	-0.136841	0.149040	0.181133	-0.495251	-0.209294	[illegible]	[illegible]	[illegible]	0.12
mtSF	-0.015415	-0.238518	0.260833	0.537808	-0.171098	0.391452	0.291066	0.522396	0.104810	[illegible]	[illegible]	[illegible]	0.24
FlrSF	0.010496	-0.251758	0.299475	0.476224	-0.144203	0.281986	0.240379	0.445863	0.097117	[illegible]	[illegible]	[illegible]	0.21
FlrSF	0.005590	0.307886	0.050986	0.295493	0.028942	0.010308	0.140024	-0.137079	-0.099260	[illegible]	[illegible]	[illegible]	0.20

In [60]:

```
# visualizacion de la correlacion
plt.figure(figsize = (40,40))
sns.heatmap(correlacion_var, annot = True, cmap = 'coolwarm')
plt.title('Heatmap de Correlaciones', fontsize = 25)
plt.show()
```



In [61]:

```
# modelo de regresion
import statsmodels.api as sm
```

In [65]:

```
X = df[columns_num].drop(columns = ['SalePrice'])
y = sm.add_constant(X)

y = df['SalePrice']

# Modelo OLS
model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:              SalePrice   R-squared (uncentered):                  0.969
Model:                            OLS   Adj. R-squared (uncentered):             0.968
Method:                 Least Squares   F-statistic:                             1401.
Date:                Wed, 21 Aug 2024   Prob (F-statistic):                       0.00
Time:                        13:57:58   Log-Likelihood:                         -17336.
No. Observations:                1460   AIC:                                 3.474e+04
Df Residuals:                    1428   BIC:                                 3.490e+04
Df Model:                          32
Covariance Type:            nonrobust
```

```
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Id             -1.5763      2.204     -0.715      0.475      -5.899       2.746
MSSubClass   -161.6117     26.447     -6.111      0.000    -213.492    -109.732
LotArea         0.3943      0.102      3.878      0.000       0.195       0.594
OverallQual  1.785e+04   1194.978     14.938      0.000    1.55e+04    2.02e+04
OverallCond  4434.6782   1032.873      4.294      0.000    2408.567    6460.790
YearBuilt     348.1441     60.992      5.708      0.000     228.500     467.788
YearRemodAdd  136.1648     66.334      2.053      0.040       6.043     266.287
BsmtFinSF1     11.8349      2.526      4.685      0.000       6.879      16.790
BsmtFinSF2     -2.8049      4.540     -0.618      0.537     -11.711       6.101
BsmtUnfSF       0.7884      2.430      0.324      0.746      -3.979       5.556
TotalBsmtSF     9.8184      3.396      2.891      0.004       3.157      16.479
1stFlrSF       19.0932      6.168      3.095      0.002       6.993      31.193
2ndFlrSF       19.0020      5.699      3.335      0.001       7.824      30.180
LowQualFinSF   -6.4361     14.884     -0.432      0.666     -35.633      22.761
GrLivArea      31.6591      5.702      5.552      0.000      20.473      42.845
BsmtFullBath 8498.5923   2626.555      3.236      0.001    3346.272    1.37e+04
BsmtHalfBath 2449.1386   4126.064      0.594      0.553   -5644.659    1.05e+04
FullBath     3550.1360   2836.350      1.252      0.211   -2013.724    9113.996
HalfBath    -1323.7077   2686.327     -0.493      0.622   -6593.279    3945.863
BedroomAbvGr -1.05e+04   1710.569     -6.139      0.000   -1.39e+04   -7145.315
```

In [66]:

```python
# Multicolinealidad
from statsmodels.stats.outliers_influence import variance_inflation_factor

# calculamos vif
vif= pd.DataFrame()
vif['columns_num'] = X.columns
vif['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
print(vif)
```

```
     columns_num           VIF
0             Id  4.093562e+00
1     MSSubClass  4.166084e+00
2        LotArea  2.574952e+00
3    OverallQual  6.619549e+01
4    OverallCond  4.086755e+01
5      YearBuilt  1.713675e+04
6   YearRemodAdd  2.054789e+04
7     BsmtFinSF1           inf
8     BsmtFinSF2           inf
9      BsmtUnfSF           inf
10   TotalBsmtSF           inf
11      1stFlrSF           inf
12      2ndFlrSF           inf
13  LowQualFinSF           inf
14     GrLivArea           inf
15  BsmtFullBath  3.679366e+00
16  BsmtHalfBath  1.216156e+00
17      FullBath  2.624655e+01
```

```
In [67]: # Variables a eliminar
variables_a_excluir = [
    'Id', 'BsmtFinSF2', 'BsmtUnfSF', 'LowQualFinSF', 'BsmtHalfBath', 'HalfBath',
    'GarageArea', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'MiscVal', 'MoSold'
]

# Crear el modelo final excluyendo las variables no significativas
X_final = X.drop(columns=variables_a_excluir)

# Volver a entrenar el modelo con las variables restantes
model_final = sm.OLS(y, X_final).fit()

# Imprimir el resumen del modelo final
print(model_final.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:              SalePrice   R-squared (uncentered):                  0.969
Model:                            OLS   Adj. R-squared (uncentered):             0.969
Method:                 Least Squares   F-statistic:                             2049.
Date:                Wed, 21 Aug 2024   Prob (F-statistic):                       0.00
Time:                        14:08:05   Log-Likelihood:                        -17337.
No. Observations:                1460   AIC:                                 3.472e+04
Df Residuals:                    1438   BIC:                                 3.483e+04
Df Model:                          22
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
MSSubClass    -161.4536     26.161     -6.172      0.000    -212.772    -110.136
LotArea          0.3944      0.101      3.913      0.000       0.197       0.592
OverallQual   1.797e+04   1176.007     15.279      0.000    1.57e+04    2.03e+04
OverallCond   4356.9561   1012.911      4.301      0.000    2370.014    6343.898
YearBuilt      327.0878     54.199      6.035      0.000     220.769     433.406
YearRemodAdd   141.0637     65.652      2.149      0.032      12.279     269.848
BsmtFinSF1      11.9452      3.060      3.904      0.000       5.944      17.947
TotalBsmtSF     10.2620      4.165      2.464      0.014       2.093      18.431
1stFlrSF        24.8359     20.235      1.227      0.220     -14.858      64.530
2ndFlrSF        23.5735     19.772      1.192      0.233     -15.211      62.358
GrLivArea       25.7885     19.784      1.304      0.193     -13.020      64.597
BsmtFullBath  7805.9201   2400.943      3.251      0.001    3096.194    1.25e+04
FullBath      3994.0962   2603.910      1.534      0.125   -1113.773    9101.966
BedroomAbvGr -1.039e+04   1689.869     -6.146      0.000   -1.37e+04   -7071.836
KitchenAbvGr -1.327e+04   5195.273     -2.555      0.011   -2.35e+04   -3083.334
TotRmsAbvGrd  5067.0234   1238.537      4.091      0.000    2637.491    7496.556
Fireplaces    3491.6842   1756.192      1.988      0.047      46.712    6936.657
GarageCars    1.109e+04   1703.813      6.508      0.000    7746.978    1.44e+04
WoodDeckSF      25.7497      7.931      3.247      0.001      10.193      41.307
ScreenPorch     54.3681     17.012      3.196      0.001      20.997      87.739
PoolArea       -41.3152     23.538     -1.755      0.079     -87.488       4.857
YrSold        -492.4666     62.429     -7.888      0.000    -614.928    -370.006
==============================================================================
Omnibus:                      575.174   Durbin-Watson:                   1.961
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           104989.324
```

In [68]:
```
from sklearn.metrics import mean_squared_error

# evaluar el modelo
y_pred = model_final.predict(X_final)
mse = mean_squared_error(y, y_pred)
r_squared = model_final.rsquared

print(f'Error Cuadratico Medio (MSE): {mse}')
print(f'R Cuadrada: {r_squared}')
```

```
Error Cuadratico Medio (MSE): 1207033352.3596303
R Cuadrada: 0.9690815575385517
```

Precisión del Modelo

- 96% lo que suguiere un modelo con predicciones de confianza.