```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings ("ignore")
```

```
df=pd.read_csv("Walmart.csv")
df
```

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemploy
0	1	05-02-2010	1643690.90	0	42.31	2.57	211.10	
1	1	12-02-2010	1641957.44	1	38.51	2.55	211.24	
2	1	19-02-2010	1611968.17	0	39.93	2.51	211.29	
3	1	26-02-2010	1409727.59	0	46.63	2.56	211.32	
4	1	05-03-2010	1554806.68	0	46.50	2.62	211.35	
...	...	...	...	...	...	...	...	

Next steps: Generate code with df View recommended plots

```
df.rename({"Store":"Tienda", "Date":"Fecha","Weekly_Sales":"VentasSem", "Holiday_Flag": "Vacacion", "Temperature": "Temperatura","Fuel
```

```
df
```

	Tienda	Fecha	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo
0	1	05-02-2010	1643690.90	0	42.31	2.57	211.10	8.11
1	1	12-02-2010	1641957.44	1	38.51	2.55	211.24	8.11
2	1	19-02-2010	1611968.17	0	39.93	2.51	211.29	8.11
3	1	26-02-2010	1409727.59	0	46.63	2.56	211.32	8.11
4	1	05-03-2010	1554806.68	0	46.50	2.62	211.35	8.11
...	...	...	...	...	...	...	...	...

Next steps: Generate code with df View recommended plots

2. Obtenga los descriptivos resumen de la base de datos e identifique a las variables numéricas y categóricas. ¿Hay algo que le llame la atención?

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Tienda       6435 non-null   int64
 1   Fecha        6435 non-null   object
 2   VentasSem    6435 non-null   float64
 3   Vacacion     6435 non-null   int64
 4   Temperatura  6435 non-null   float64
 5   PrecioFule   6435 non-null   float64
 6   CPI          6435 non-null   float64
 7   Desempleo    6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

```
df.describe()
```

	Tienda	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo
count	6435.00	6435.00	6435.00	6435.00	6435.00	6435.00	6435.00
mean	23.00	1046964.88	0.07	60.66	3.36	171.58	8.00
std	12.99	564366.62	0.26	18.44	0.46	39.36	1.88
min	1.00	209986.25	0.00	-2.06	2.47	126.06	3.88
25%	12.00	553350.10	0.00	47.46	2.93	131.74	6.89
50%	23.00	960746.04	0.00	62.67	3.44	182.62	7.87
75%	34.00	1420158.66	0.00	74.94	3.73	212.74	8.62
max	45.00	3818686.45	1.00	100.14	4.47	227.23	14.31

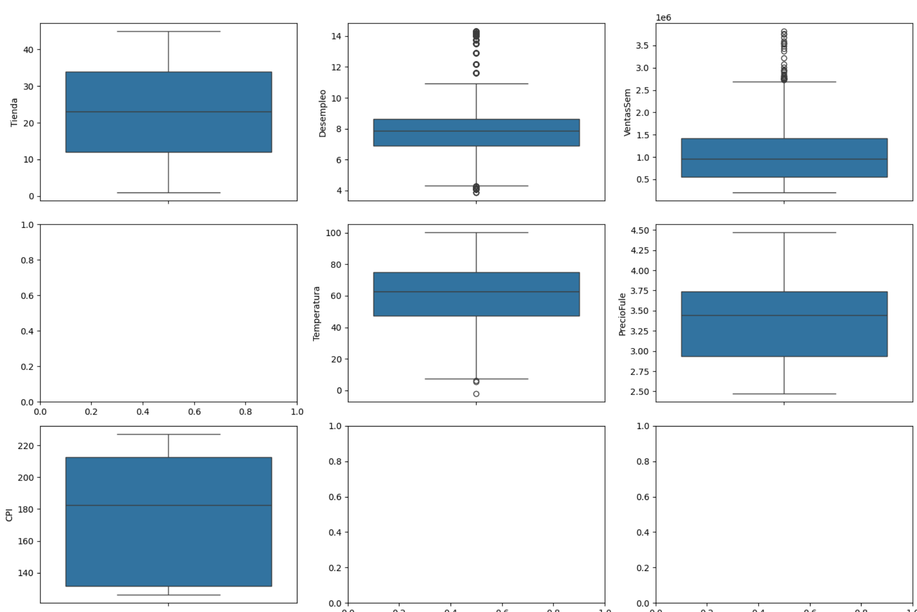
3. Evalúe si la base contiene datos perdidos

```
df.isna().sum()
```

```
Tienda         0
Fecha          0
VentasSem      0
Vacacion       0
Temperatura    0
PrecioFule     0
CPI            0
Desempleo      0
dtype: int64
```

4. Evalúe si alguna de las variables contiene datos atípicos (outliers) o De ser el caso, detalle cuáles y qué método estadístico aplicarán para corregir.

```
# Box plots
fig, axs = plt.subplots(3,3, figsize = (15,10))
plt1 = sns.boxplot(df['Tienda'], ax = axs[0,0])
plt2 = sns.boxplot(df['Desempleo'], ax = axs[0,1])
plt3 = sns.boxplot(df['VentasSem'], ax = axs[0,2])
plt1 = sns.boxplot(df['Temperatura'], ax = axs[1,1])
plt2 = sns.boxplot(df['PrecioFule'], ax = axs[1,2])
plt3 = sns.boxplot(df['CPI'], ax = axs[2,0])

plt.tight_layout()
```




```
# Calculamos el Quartil 1 y Quartil 3 que son aquellos que nos permiten estimar los límites de los datos atípicos
Q1 = df.Desempleo.quantile(0.25)
Q3 = df.Desempleo.quantile(0.75)
IQR = Q3 - Q1  #rango intercuartil
print(IQR)
```

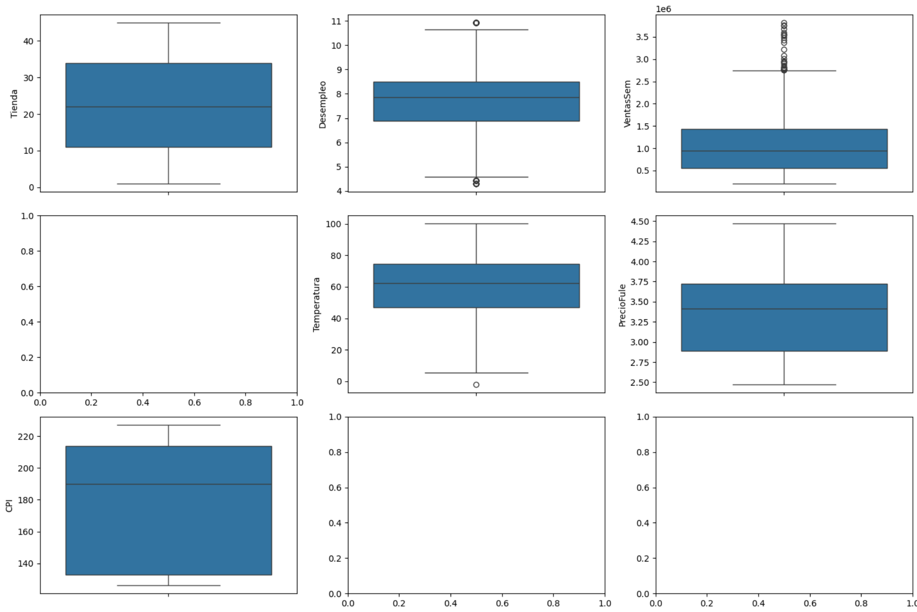
```
1.7309999999999999
```

```
# Ahora removemos aquellas observaciones que se encuentran por fuera del rango: 1.5 x IQR
```

```
df = df[~((df['Desempleo'] < (Q1 - 1.5 * IQR)) |(df['Desempleo'] > (Q3 + 1.5 * IQR)))]
df.shape
```

```
(5954, 8)
```

```
# Box plots
fig, axs = plt.subplots(3,3, figsize = (15,10))
plt1 = sns.boxplot(df['Tienda'], ax = axs[0,0])
plt2 = sns.boxplot(df['Desempleo'], ax = axs[0,1])
plt3 = sns.boxplot(df['VentasSem'], ax = axs[0,2])
plt1 = sns.boxplot(df['Temperatura'], ax = axs[1,1])
plt2 = sns.boxplot(df['PrecioFule'], ax = axs[1,2])
plt3 = sns.boxplot(df['CPI'], ax = axs[2,0])

plt.tight_layout()
```




```
df
```

	Tienda	Fecha	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo
0	1	05-02-2010	1643690.90	0	42.31	2.57	211.10	8.11
1	1	12-02-2010	1641957.44	1	38.51	2.55	211.24	8.11
2	1	19-02-2010	1611968.17	0	39.93	2.51	211.29	8.11
3	1	26-02-2010	1409727.59	0	46.63	2.56	211.32	8.11
4	1	05-03-2010	1554806.68	0	46.50	2.62	211.35	8.11
...	...	...	...	...	...	...	...	...

Next steps: Generate code with `df` View recommended plots

5. Grafique las distribuciones de las variables y a priori comente sobre ellas.

```
pd.options.display.float_format = '{:.2f}'.format
```

```
df.describe()
```

	Tienda	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo
count	5954.00	5954.00	5954.00	5954.00	5954.00	5954.00	5954.00
mean	22.74	1050893.92	0.07	60.29	3.34	174.92	7.72
std	13.09	572191.29	0.26	18.45	0.46	39.03	1.24
min	1.00	209986.25	0.00	-2.06	2.47	126.06	4.31
25%	11.00	554147.17	0.00	46.76	2.89	132.76	6.89
50%	22.00	951379.13	0.00	62.39	3.42	189.81	7.85
75%	34.00	1436132.69	0.00	74.66	3.72	213.76	8.49
max	45.00	3818686.45	1.00	100.14	4.47	227.23	10.93

```
df['Fecha'] = pd.to_datetime(df['Fecha'], format="%d-%m-%Y")
df
```

	Tienda	Fecha	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo
0	1	2010-02-05	1643690.90	0	42.31	2.57	211.10	8.11
1	1	2010-02-12	1641957.44	1	38.51	2.55	211.24	8.11
2	1	2010-02-19	1611968.17	0	39.93	2.51	211.29	8.11
3	1	2010-02-26	1409727.59	0	46.63	2.56	211.32	8.11
4	1	2010-03-05	1554806.68	0	46.50	2.62	211.35	8.11
...	...	...	...	...	...	...	...	...
6430	45	2012-09-28	713173.95	0	64.88	4.00	192.01	8.68
6431	45	2012-10-05	733455.07	0	64.89	3.98	192.17	8.67

Next steps: Generate code with df View recommended plots

```
def season_getter(quarter):
    quarter_to_season = {1: 'Invierno', 2: 'Primavera', 3: 'Verano', 4: 'Otoño'}
    return quarter_to_season.get(quarter, 'Invalid Quarter')
```

```
import calendar
# Use the 'assign' method to add multiple columns in a single line
df = df.assign(
    Año=df['Fecha'].dt.year,
    quarter=df['Fecha'].dt.quarter,
    Estacion=df['Fecha'].dt.quarter.map(season_getter),
    Mes=df['Fecha'].dt.month,
    Mes_nombre=df['Fecha'].dt.month_name(),
    Semana=df['Fecha'].dt.isocalendar().week,
    Dia_semana=df['Fecha'].dt.day_name())
```

```
df.head(5)
```

	Tienda	Fecha	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desempleo	Año
0	1	2010-02-05	1643690.90	0	42.31	2.57	211.10	8.11	2010
1	1	2010-02-12	1641957.44	1	38.51	2.55	211.24	8.11	2010
2	1	2010-02-19	1611968.17	0	39.93	2.51	211.29	8.11	2010
3	1	2010-02-26	1409727.59	0	46.63	2.56	211.32	8.11	2010
4	1	2010-03-05	1554806.68	0	46.50	2.62	211.35	8.11	2010

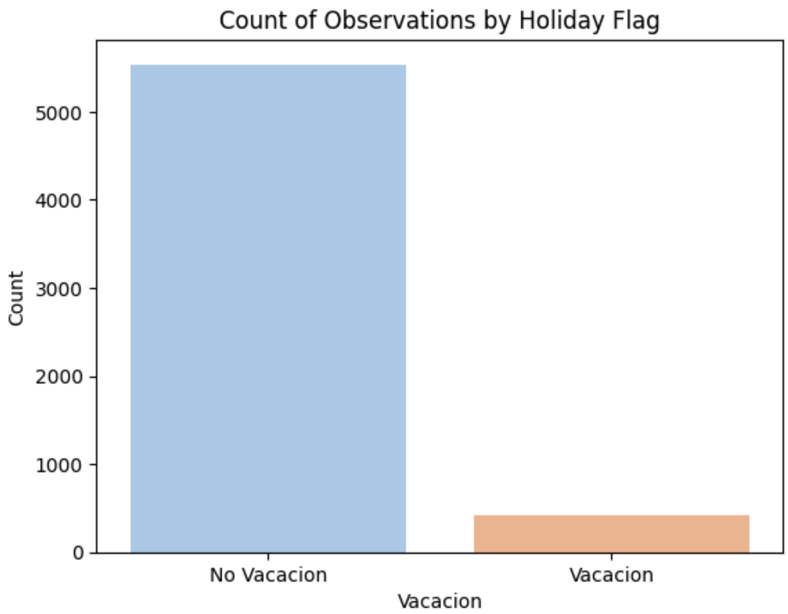
Next steps: Generate code with df View recommended plots

```
df.dtypes
```

```
Tienda                  int64
Fecha          datetime64[ns]
VentasSem             float64
Vacacion                int64
Temperatura           float64
PrecioFule            float64
CPI                   float64
Desempleo             float64
Año                     int64
quarter                 int64
Estacion               object
Mes                     int64
```

```
Mes_nombre              object
Semana                  UInt32
Dia_semana              object
dtype: object
```

```
sns.countplot(data=df, x="Vacacion", palette='pastel')
plt.xlabel('Vacacion')
plt.ylabel('Count')
plt.title('Count of Observations by Holiday Flag')
plt.xticks(ticks=[0, 1], labels=['No Vacacion', 'Vacacion'])

plt.show()
```




```
var_cuantitativas = df.select_dtypes('number').columns
var_cualitativas  =df.select_dtypes('object').columns
```

```
df[var_cualitativas]
```

	Estacion	Mes_nombre	Dia_semana
0	Invierno	February	Friday
1	Invierno	February	Friday
2	Invierno	February	Friday
3	Invierno	February	Friday
4	Invierno	March	Friday
...	...	...	...
6430	Verano	September	Friday
6431	Otoño	October	Friday
6432	Otoño	October	Friday
6433	Otoño	October	Friday
6434	Otoño	October	Friday

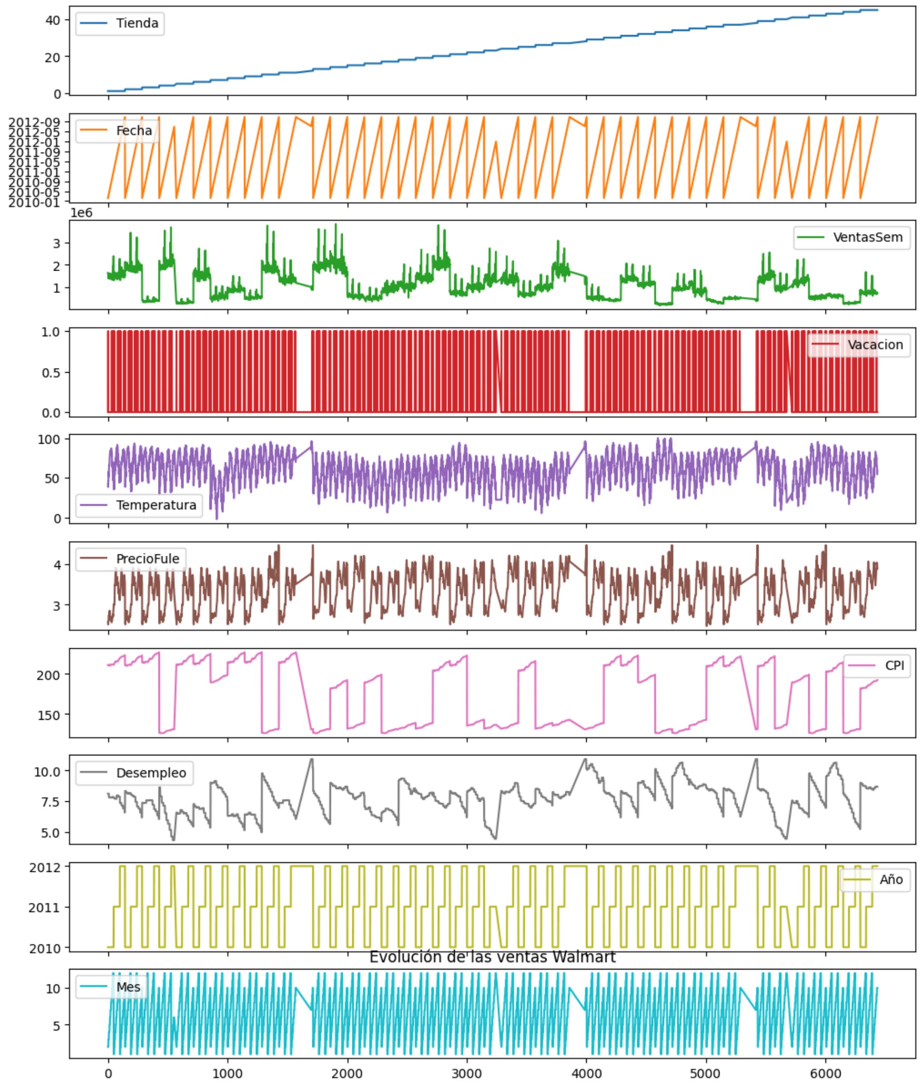
5954 rows × 3 columns

```
df['Mes_nombre'] = pd.Categorical(df['Mes_nombre'])
df['Dia_semana'] = pd.Categorical(df['Dia_semana'])
df['quarter'] = pd.Categorical(df['quarter'])
df['Semana'] = pd.Categorical(df['Semana'])
```

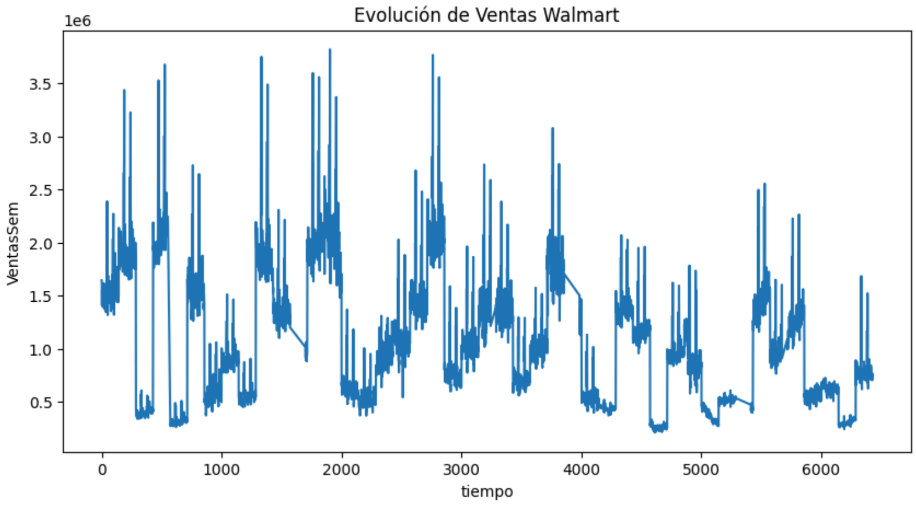
```
print(df.dtypes)
```

```
Tienda                   int64
Fecha           datetime64[ns]
VentasSem              float64
Vacacion                 int64
Temperatura            float64
PrecioFule             float64
CPI                    float64
Desempleo              float64
Año                      int64
quarter               category
Estacion                object
Mes                      int64
Mes_nombre            category
Semana                category
Dia_semana            category
dtype: object
```

```
df.plot(subplots=True, figsize=(12,15))
plt.title('Evolución de las ventas Walmart')

plt.show()
```



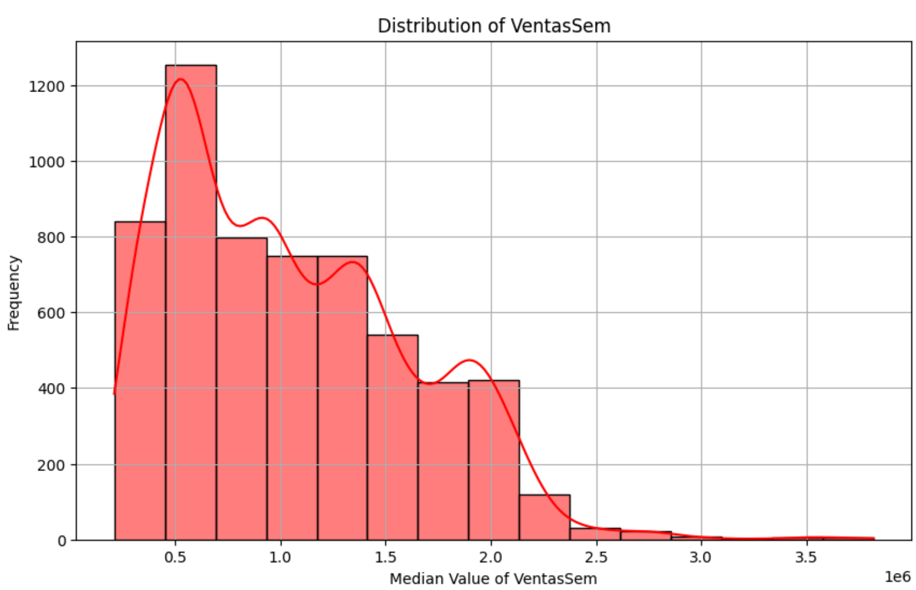

```
plt.figure(figsize=(10,5))
plt.plot(df.VentasSem)
plt.title("Evolución de Ventas Walmart")
plt.xlabel("tiempo")
plt.ylabel("VentasSem")
plt.show()
```



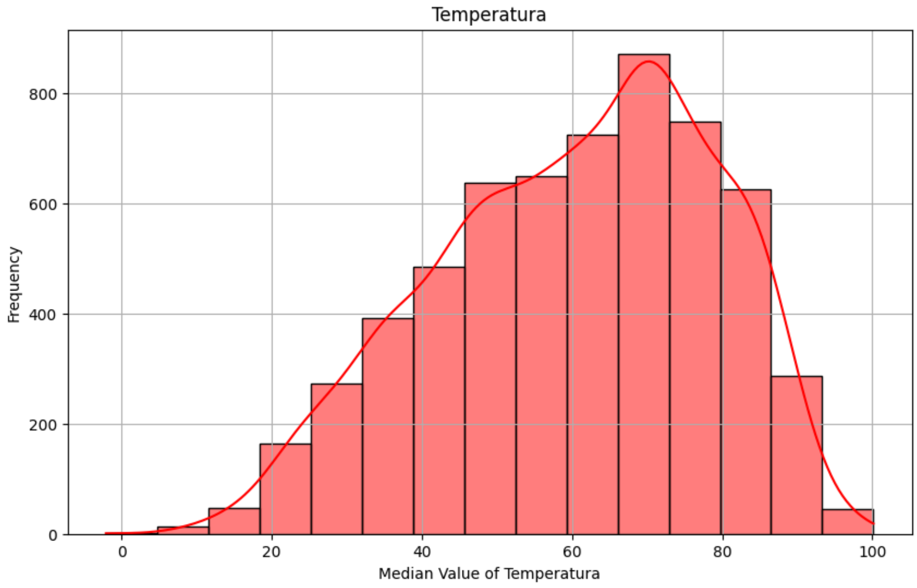

```
df [["VentasSem", "Temperatura","PrecioFule","Desempleo", "CPI","Estacion", "Vacacion"]].describe()
```

	VentasSem	Temperatura	PrecioFule	Desempleo	CPI	Vacacion
count	5954.00	5954.00	5954.00	5954.00	5954.00	5954.00
mean	1050893.92	60.29	3.34	7.72	174.92	0.07
std	572191.29	18.45	0.46	1.24	39.03	0.26
min	209986.25	-2.06	2.47	4.31	126.06	0.00
25%	554147.17	46.76	2.89	6.89	132.76	0.00
50%	951379.13	62.39	3.42	7.85	189.81	0.00
75%	1436132.69	74.66	3.72	8.49	213.76	0.00
max	3818686.45	100.14	4.47	10.93	227.23	1.00

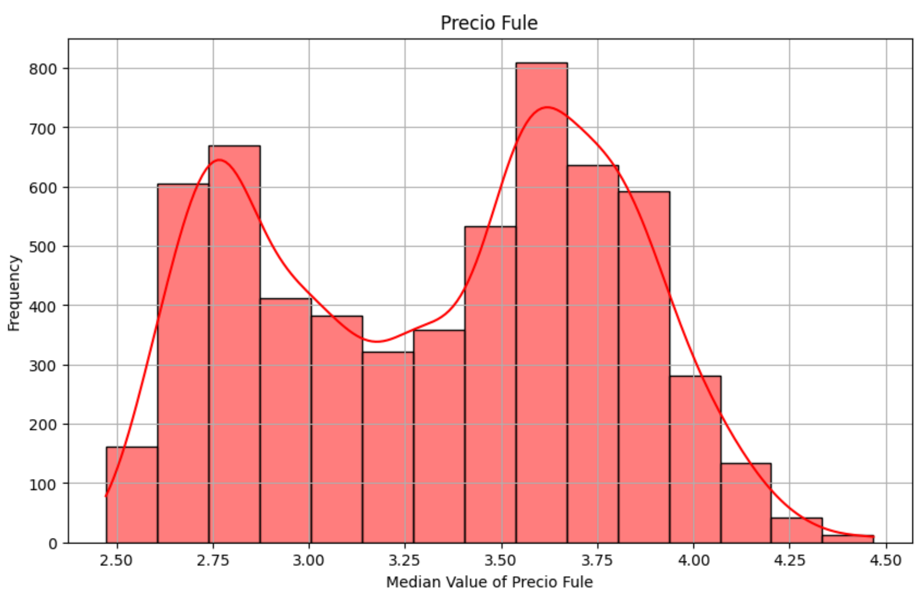
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='VentasSem', kde=True,bins = 15, color = 'r')
plt.title('Distribution of VentasSem')
plt.xlabel('Median Value of VentasSem')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



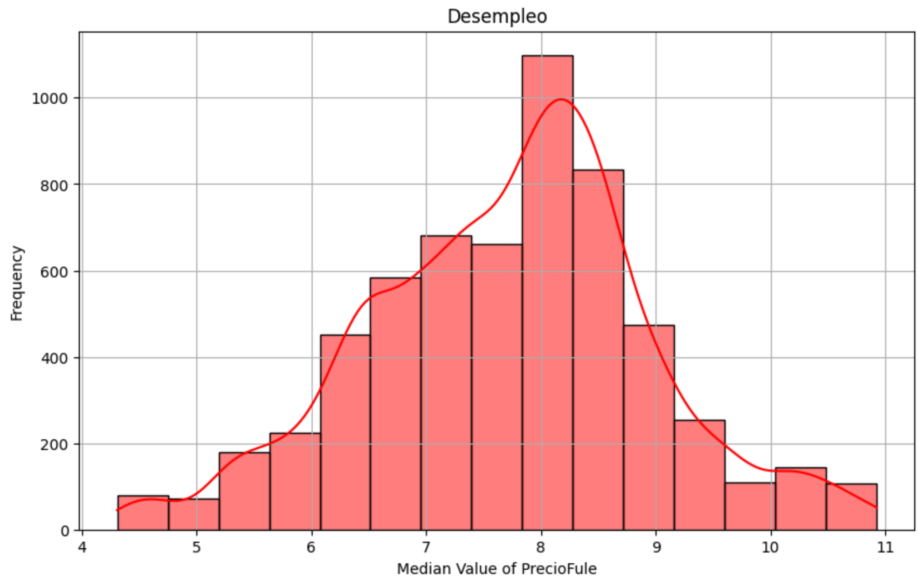

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Temperatura', kde=True,bins = 15, color = 'r')
plt.title('Temperatura')
plt.xlabel('Median Value of Temperatura')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



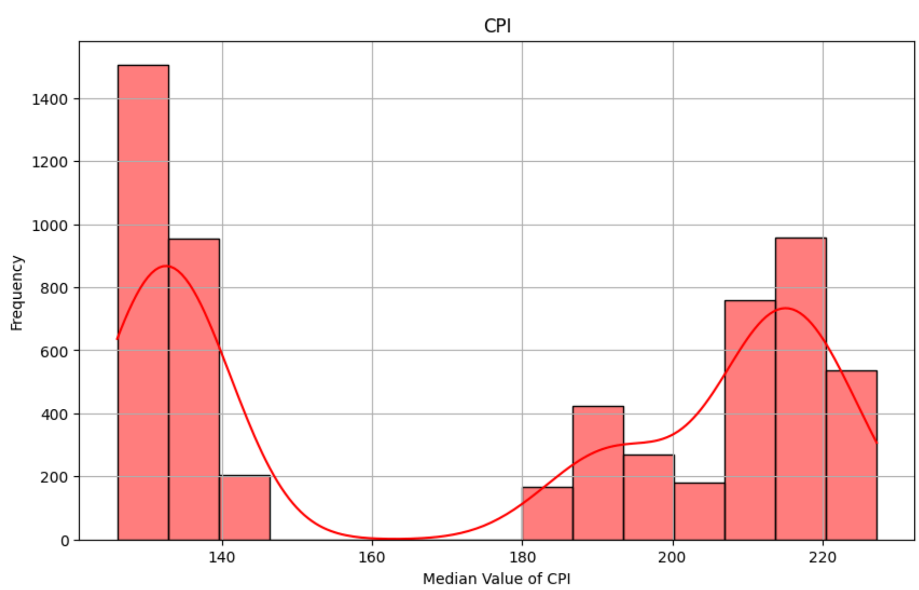

```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='PrecioFule', kde=True,bins = 15, color = 'r')
plt.title('Precio Fule')
plt.xlabel('Median Value of Precio Fule')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Desempleo', kde=True,bins = 15, color = 'r')
plt.title('Desempleo')
plt.xlabel('Median Value of PrecioFule')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='CPI', kde=True,bins = 15, color = 'r')
plt.title('CPI')
plt.xlabel('Median Value of CPI')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



6. Obtenga las correlaciones entre los datos de corte numérico

```
df.corr().style.background_gradient(cmap='coolwarm')
```

	Tienda	VentasSem	Vacacion	Temperatura	PrecioFule	CPI	Desemple
Tienda	1.000000	-0.321986	0.000616	-0.022126	0.049860	-0.205025	0.30947
VentasSem	-0.321986	1.000000	0.036725	-0.061389	0.011257	-0.087443	-0.07499
Vacacion	0.000616	0.036725	1.000000	-0.156881	-0.076853	-0.003215	0.00975
Temperatura	-0.022126	-0.061389	-0.156881	1.000000	0.147560	0.218762	0.02623
PrecioFule	0.049860	0.011257	-0.076853	0.147560	1.000000	-0.142689	-0.10426
CPI	-0.205025	-0.087443	-0.003215	0.218762	-0.142689	1.000000	-0.21620
Desempleo	0.309472	-0.074999	0.009751	0.026236	-0.104268	-0.216206	1.00000
Año	-0.004488	-0.034163	-0.055937	0.086933	0.782773	0.087913	-0.24164
Mes	0.006684	0.074620	0.123563	0.232973	-0.032352	-0.002694	-0.01169

7. Comente que variable escogerán como variable dependiente y que variables introducirán a su modelo.

La variable dependiente es Ventas Semanales, introduciré a mi modelo la variable Vacacion, Estacion, Año, Tienda,Mes

8. Indique que tipo de modelación realizarán y porqué

```
df=df[['Semana', 'VentasSem', 'CPI','Tienda','Vacacion','Desempleo', "PrecioFule","Temperatura"]]
df
```

	Semana	VentasSem	CPI	Tienda	Vacacion	Desempleo	PrecioFule	Temperatura
0	5	1643690.90	211.10	1	0	8.11	2.57	42.31
1	6	1641957.44	211.24	1	1	8.11	2.55	38.51
2	7	1611968.17	211.29	1	0	8.11	2.51	39.93
3	8	1409727.59	211.32	1	0	8.11	2.56	46.63
4	9	1554806.68	211.35	1	0	8.11	2.62	46.50
...	...	...	...	...	...	...	...	...
6430	39	713173.95	192.01	45	0	8.68	4.00	64.88
6431	40	733455.07	192.17	45	0	8.67	3.98	64.89
6432	41	734464.36	192.33	45	0	8.67	4.00	54.47
6433	42	718125.53	192.33	45	0	8.67	3.97	56.47
6434	43	760281.43	192.31	45	0	8.67	3.88	58.85

5954 rows × 8 columns

Next steps: Generate code with df View recommended plots

```
df.describe()
```

	VentasSem	CPI	Tienda	Vacacion	Desempleo	PrecioFule	Temperatura
count	5954.00	5954.00	5954.00	5954.00	5954.00	5954.00	5954.00
mean	1050893.92	174.92	22.74	0.07	7.72	3.34	60.29
std	572191.29	39.03	13.09	0.26	1.24	0.46	18.45
min	209986.25	126.06	1.00	0.00	4.31	2.47	-2.06
25%	554147.17	132.76	11.00	0.00	6.89	2.89	46.76
50%	951379.13	189.81	22.00	0.00	7.85	3.42	62.39
75%	1436132.69	213.76	34.00	0.00	8.49	3.72	74.66
max	3818686.45	227.23	45.00	1.00	10.93	4.47	100.14

```
pip install linearmodels
```

```
Requirement already satisfied: linearmodels in /usr/local/lib/python3.10/dist-packages (5.4)
Requirement already satisfied: numpy>=1.22.0 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (1.25.2)
Requirement already satisfied: pandas>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (1.5.3)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (1.11.4)
Requirement already satisfied: statsmodels>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (0.14.1)
Requirement already satisfied: mypy-extensions>=0.4 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (1.0.0)
Requirement already satisfied: Cython>=0.29.37 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (3.0.9)
Requirement already satisfied: pyhdfe>=0.1 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (0.2.0)
Requirement already satisfied: formulaic>=0.6.5 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (1.0.1)
Requirement already satisfied: setuptools-scm[toml]<9.0.0,>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from linearmodels) (
Requirement already satisfied: interface-meta>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.6.5->linearmode
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.6.5->linearm
Requirement already satisfied: wrapt>=1.0 in /usr/local/lib/python3.10/dist-packages (from formulaic>=0.6.5->linearmodels) (1.14.1
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.0->linearmodels
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.0->linearmodels) (2023.4)
Requirement already satisfied: packaging>=20 in /usr/local/lib/python3.10/dist-packages (from setuptools-scm[toml]<9.0.0,>=8.0.0->
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from setuptools-scm[toml]<9.0.0,>=8.0.0->lin
Requirement already satisfied: tomli>=1 in /usr/local/lib/python3.10/dist-packages (from setuptools-scm[toml]<9.0.0,>=8.0.0->linea
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.12.0->linearmodels) (0
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels>=0.12.0->linearmodel
```

```
from linearmodels import PooledOLS
import statsmodels.api as sm
```

```
df.set_index(['Semana', "Vacacion"], inplace=True)
```

```
from linearmodels import PanelOLS
```

CPI

```
X = sm.tools.tools.add_constant(df.CPI)
y = df.VentasSem
```

```
modelo_fe = PanelOLS(y, X, entity_effects = True)
resultados_fe = modelo_fe.fit()
```

```
resultados_fe
```

PanelOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0078
Estimator:	PanelOLS	**R-squared (Between):**	0.0018
No. Observations:	5954	**R-squared (Within):**	0.0078
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0076
Time:	04:55:07	**Log-likelihood**	-8.722e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	46.292
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5901)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	46.292
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5901)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.272e+06	3.329e+04	38.210	0.0000	1.207e+06	1.337e+06
CPI	-1263.7	185.74	-6.8038	0.0000	-1627.8	-899.61

F-test for Poolability: 5.5497
P-value: 0.0000
Distribution: F(51,5901)

Included effects: Entity
id: 0x7b6c10ede020

```
modelo1 = PooledOLS(y, X)
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
```

```
# Store values for checking homoskedasticity graphically
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
residuos_pooled_OLS = resultados_pooled_OLS.resids
```

```
resultados_pooled_OLS
```

PooledOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0076
Estimator:	PooledOLS	**R-squared (Between):**	0.0019
No. Observations:	5954	**R-squared (Within):**	0.0078
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0076
Time:	04:55:08	**Log-likelihood**	-8.736e+04
Cov. Estimator:	Clustered		
		F-statistic:	45.862
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	387.95
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.275e+06	2.535e+04	50.294	0.0000	1.225e+06	1.325e+06
CPI	-1282.0	65.089	-19.696	0.0000	-1409.6	-1154.4

id: 0x7b6c10eb4cd0

```
from linearmodels import RandomEffects
```

```
modelo_re = RandomEffects(y, X)
resultados_re = modelo_re.fit()
```

```
resultados_re
```

RandomEffects Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0140
Estimator:	RandomEffects	**R-squared (Between):**	0.0045
No. Observations:	5954	**R-squared (Within):**	0.0078
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0075
Time:	04:55:08	**Log-likelihood**	-8.725e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	84.696
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	46.491
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.278e+06	3.676e+04	34.783	0.0000	1.206e+06	1.351e+06
CPI	-1267.7	185.92	-6.8185	0.0000	-1632.2	-903.24

id: 0x7b6c0f17e8c0

Desempleo

```
X2 = sm.tools.tools.add_constant(df.Desempleo)
y2 = df.VentasSem
modelo_fe = PanelOLS(y2, X2, entity_effects = True)
resultados_fe = modelo_fe.fit()
resultados_fe
```

PanelOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0063
Estimator:	PanelOLS	**R-squared (Between):**	-0.0115
No. Observations:	5954	**R-squared (Within):**	0.0063
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0056
Time:	04:55:08	**Log-likelihood**	-8.722e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	37.500
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5901)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	37.500
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5901)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.327e+06	4.572e+04	29.030	0.0000	1.238e+06	1.417e+06
Desempleo	-3.581e+04	5847.2	-6.1237	0.0000	-4.727e+04	-2.434e+04

F-test for Poolability: 5.6171
P-value: 0.0000
Distribution: F(51,5901)

Included effects: Entity
id: 0x7b6c117029e0

```
modelo1 = PooledOLS(y2, X2)
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
```

```
# Store values for checking homoskedasticity graphically
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
residuos_pooled_OLS = resultados_pooled_OLS.resids
```

```
resultados_pooled_OLS
```

PooledOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0056
Estimator:	PooledOLS	**R-squared (Between):**	-0.0112
No. Observations:	5954	**R-squared (Within):**	0.0063
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0056
Time:	04:55:09	**Log-likelihood**	-8.736e+04
Cov. Estimator:	Clustered		
		F-statistic:	33.668
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	415.17
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.318e+06	2.531e+04	52.059	0.0000	1.268e+06	1.367e+06
Desempleo	-3.454e+04	1695.0	-20.376	0.0000	-3.786e+04	-3.121e+04

id: 0x7b6c0f2e56c0

```
from linearmodels import RandomEffects
modelo_re = RandomEffects(y2, X2)
resultados_re = modelo_re.fit()
resultados_re
```

RandomEffects Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0116
Estimator:	RandomEffects	**R-squared (Between):**	-0.0087
No. Observations:	5954	**R-squared (Within):**	0.0063
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0055
Time:	04:55:09	**Log-likelihood**	-8.725e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	69.730
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	37.056
		P-value	0.0000
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.332e+06	4.906e+04	27.145	0.0000	1.236e+06	1.428e+06
Desempleo	-3.559e+04	5846.0	-6.0873	0.0000	-4.705e+04	-2.413e+04

id: 0x7b6c1a3e3b80

PrecioFule

```
X3 = sm.tools.tools.add_constant(df.PrecioFule)
y3 = df.VentasSem
modelo_fe = PanelOLS(y3, X3, entity_effects = True)
resultados_fe = modelo_fe.fit()
resultados_fe
```

PanelOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0011
Estimator:	PanelOLS	**R-squared (Between):**	-0.0324
No. Observations:	5954	**R-squared (Within):**	0.0011
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	-0.0004
Time:	04:55:09	**Log-likelihood**	-8.724e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	6.5383
Entities:	52	**P-value**	0.0106
Avg Obs:	114.50	**Distribution:**	F(1,5901)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	6.5383
		P-value	0.0106
Time periods:	2	**Distribution:**	F(1,5901)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	9.086e+05	5.611e+04	16.195	0.0000	7.987e+05	1.019e+06
PrecioFule	4.26e+04	1.666e+04	2.5570	0.0106	9939.8	7.526e+04

F-test for Poolability: 5.6519
P-value: 0.0000
Distribution: F(51,5901)

Included effects: Entity
id: 0x7b6c0ed87100

```
modelo1 = PooledOLS(y3, X3)
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
residuos_pooled_OLS = resultados_pooled_OLS.resids
resultados_pooled_OLS
```

PooledOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0001
Estimator:	PooledOLS	**R-squared (Between):**	-0.0122
No. Observations:	5954	**R-squared (Within):**	0.0006
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0001
Time:	04:55:10	**Log-likelihood**	-8.738e+04
Cov. Estimator:	Clustered		
		F-statistic:	0.7543
Entities:	52	**P-value**	0.3852
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	1.4532
		P-value	0.2281
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.004e+06	5.134e+04	19.554	0.0000	9.032e+05	1.105e+06
PrecioFule	1.408e+04	1.168e+04	1.2055	0.2281	-8818.1	3.698e+04

id: 0x7b6c0ec7f280

```
from linearmodels import RandomEffects
modelo_re = RandomEffects(y3, X3)
resultados_re = modelo_re.fit()
resultados_re
```

RandomEffects Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0065
Estimator:	RandomEffects	**R-squared (Between):**	-0.0253
No. Observations:	5954	**R-squared (Within):**	0.0011
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	-0.0003
Time:	04:55:10	**Log-likelihood**	-8.727e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	38.922
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	5.0604
		P-value	0.0245
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	9.332e+05	5.815e+04	16.047	0.0000	8.192e+05	1.047e+06
PrecioFule	3.721e+04	1.654e+04	2.2495	0.0245	4782.8	6.963e+04

id: 0x7b6c0ec8d390

Temperatura

```
X4 = sm.tools.tools.add_constant(df.Temperatura)
y4 = df.VentasSem
modelo_fe = PanelOLS(y4, X4, entity_effects = True)
resultados_fe = modelo_fe.fit()
resultados_fe
```

PanelOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0023
Estimator:	PanelOLS	**R-squared (Between):**	0.0344
No. Observations:	5954	**R-squared (Within):**	0.0023
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0034
Time:	04:55:10	**Log-likelihood**	-8.724e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	13.868
Entities:	52	**P-value**	0.0002
Avg Obs:	114.50	**Distribution:**	F(1,5901)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	13.868
		P-value	0.0002
Time periods:	2	**Distribution:**	F(1,5901)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.203e+06	4.137e+04	29.068	0.0000	1.121e+06	1.284e+06
Temperatura	-2515.6	675.53	-3.7239	0.0002	-3839.9	-1191.3

F-test for Poolability: 5.3599
P-value: 0.0000
Distribution: F(51,5901)

Included effects: Entity
id: 0x7b6c0ec8dc90

```
modelo1 = PooledOLS(y4, X4)
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
residuos_pooled_OLS = resultados_pooled_OLS.resids
resultados_pooled_OLS
```

PooledOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0038
Estimator:	PooledOLS	**R-squared (Between):**	0.0394
No. Observations:	5954	**R-squared (Within):**	0.0022
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0038
Time:	04:55:11	**Log-likelihood**	-8.737e+04
Cov. Estimator:	Clustered		
		F-statistic:	22.516
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	4.6577
		P-value	0.0310
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.166e+06	6.582e+04	17.711	0.0000	1.037e+06	1.295e+06
Temperatura	-1903.4	881.95	-2.1582	0.0310	-3632.3	-174.45

id: 0x7b6c0f2e7b20

```
from linearmodels import RandomEffects
modelo_re = RandomEffects(y4, X4)
resultados_re = modelo_re.fit()
resultados_re
```

RandomEffects Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0080
Estimator:	RandomEffects	**R-squared (Between):**	0.0372
No. Observations:	5954	**R-squared (Within):**	0.0023
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0035
Time:	04:55:11	**Log-likelihood**	-8.726e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	47.946
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	15.647
		P-value	0.0001
Time periods:	2	**Distribution:**	F(1,5952)
Avg Obs:	2977.0		
Min Obs:	418.00		
Max Obs:	5536.0		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.196e+06	4.016e+04	29.783	0.0000	1.117e+06	1.275e+06
Temperatura	-2359.0	596.36	-3.9557	0.0001	-3528.1	-1189.9

id: 0x7b6c18176fe0

Vacation

```
modelo1 = PooledOLS(y5, X5)
resultados_pooled_OLS = modelo1.fit(cov_type='clustered', cluster_entity=True)
# Store values for checking homoskedasticity graphically
predicciones_pooled_OLS = resultados_pooled_OLS.predict().fitted_values
residuos_pooled_OLS = resultados_pooled_OLS.resids
resultados_pooled_OLS
```

PooledOLS Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0013
Estimator:	PooledOLS	**R-squared (Between):**	0.0282
No. Observations:	5954	**R-squared (Within):**	0.0000
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0013
Time:	04:55:11	**Log-likelihood**	-8.738e+04
Cov. Estimator:	Clustered		
		F-statistic:	8.0384
Entities:	52	**P-value**	0.0046
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	0.9377
		P-value	0.3329
Time periods:	45	**Distribution:**	F(1,5952)
Avg Obs:	132.31		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.045e+06	1.448e+04	72.201	0.0000	1.017e+06	1.073e+06
Vacacion	8.224e+04	8.493e+04	0.9684	0.3329	-8.425e+04	2.487e+05

id: 0x7b6c0eca98a0

```
from linearmodels import RandomEffects
modelo_re = RandomEffects(y5, X5)
resultados_re = modelo_re.fit()
resultados_re
```

RandomEffects Estimation Summary

Dep. Variable:	VentasSem	**R-squared:**	0.0056
Estimator:	RandomEffects	**R-squared (Between):**	0.0309
No. Observations:	5954	**R-squared (Within):**	0.0000
Date:	Wed, Mar 20 2024	**R-squared (Overall):**	0.0012
Time:	04:55:11	**Log-likelihood**	-8.727e+04
Cov. Estimator:	Unadjusted		
		F-statistic:	33.579
Entities:	52	**P-value**	0.0000
Avg Obs:	114.50	**Distribution:**	F(1,5952)
Min Obs:	82.000		
Max Obs:	126.00	**F-statistic (robust):**	1.5277
		P-value	0.2165
Time periods:	45	**Distribution:**	F(1,5952)
Avg Obs:	132.31		
Min Obs:	17.000		
Max Obs:	143.00		

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	1.05e+06	2.026e+04	51.823	0.0000	1.01e+06	1.09e+06
Vacacion	9.098e+04	7.361e+04	1.2360	0.2165	-5.332e+04	2.353e+05

id: 0x7b6c0ecd34c0

```
from scipy.stats import pearsonr
```

```
pearson_coef, p_value = pearsonr(df['Desempleo'], df['VentasSem'])
print("The Pearson Correlation Coefficient is", pearson_coef, "with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.07499880002681349 with a P-value of P = 6.872292803910042e-09
```

```
# Correlations with weekly sales
corr = df[['VentasSem', 'Temperatura', 'PrecioFule', 'CPI', 'Desempleo']].corr()['VentasSem'].sort_values(ascending = False)
corr = corr.to_frame()
corr.style.background_gradient(cmap="RdYlBu")
```

	VentasSem
VentasSem	1.000000
PrecioFule	0.011257
Temperatura	-0.061389
Desempleo	-0.074999
CPI	-0.087443

```
import numpy.linalg as la
from scipy import stats
import numpy as np
```

```
def hausman(fe, re):
    b = fe.params
    B = re.params
    v_b = fe.cov
    v_B = re.cov
    df = b[np.abs(b) < 1e8].size
    chi2 = np.dot((b - B).T, la.inv(v_b - v_B).dot(b - B))

    pval = stats.chi2.sf(chi2, df)
    return chi2, df, pval
    hausman = hausman(resultados_fe, resultados_re)
```

```
import statsmodels.stats.api as sms
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.compat import lzip
```

```
regresion = ols("VentasSem~Desempleo + Temperatura + PrecioFule + CPI", data=df)
results = regresion.fit()
```

```
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              VentasSem   R-squared:                       0.018
Model:                            OLS   Adj. R-squared:                  0.018
Method:                 Least Squares   F-statistic:                     27.65
Date:                Wed, 20 Mar 2024   Prob (F-statistic):           8.80e-23
Time:                        04:55:11   Log-Likelihood:                -87327.
No. Observations:                5954   AIC:                         1.747e+05
Df Residuals:                    5949   BIC:                         1.747e+05
Df Model:                           4
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept    1.757e+06   9.26e+04     18.969      0.000    1.58e+06    1.94e+06
Desempleo   -4.464e+04   6153.835     -7.254      0.000   -5.67e+04   -3.26e+04
Temperatura -1096.5001    417.716     -2.625      0.009   -1915.375    -277.626
PrecioFule  -1.021e+04   1.67e+04     -0.610      0.542    -4.3e+04    2.26e+04
CPI         -1492.9428    202.871     -7.359      0.000   -1890.644   -1095.242
==============================================================================
Omnibus:                      357.375   Durbin-Watson:                   0.112
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              425.020
Skew:                           0.654   Prob(JB):                     5.11e-93
Kurtosis:                       2.989   Cond. No.                     2.41e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.41e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

El Modelo de Regresión Lineal por Sklearn

```
from sklearn.linear_model import LinearRegression
```

```
var_cuantitativas = df.select_dtypes('number').columns
var_cualitativas  =df.select_dtypes('object').columns
```

```
from sklearn.preprocessing import LabelEncoder
```

```
# creating instance of labelencoder
labelencoder = LabelEncoder()
```

```
df[var_cualitativas] = df[var_cualitativas].apply(labelencoder.fit_transform)
```

```
X = df[df.columns.difference(["VentasSem"])]
y = df.VentasSem
```

```
from sklearn.model_selection import train_test_split
```

```
X_train , X_test , y_train , y_test = train_test_split(X , y , test_size = 0.50,random_state =123)
```

```
print(X_train.shape,"",type(X_train))
print(y_train.shape,"\t ",type(y_train))
print(X_test.shape,"",type(X_test))
print(y_test.shape,"\t ",type(y_test))
```

```
(2977, 5)  <class 'pandas.core.frame.DataFrame'>
(2977,)          <class 'pandas.core.series.Series'>
(2977, 5)  <class 'pandas.core.frame.DataFrame'>
(2977,)          <class 'pandas.core.series.Series'>
```

```
modelo_regresion = LinearRegression()
modelo_regresion.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
predicciones_train = modelo_regresion.predict(X_train)
predicciones_test = modelo_regresion.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
MSE_train = mean_squared_error(y_train, predicciones_train)
MSE_test = mean_squared_error(y_test, predicciones_test)
print(MSE_train)
print(MSE_test)
```

```
279481493491.08575
290850755242.375
```

```
RMSE_train = np.sqrt(MSE_train)
RMSE_test = np.sqrt(MSE_test)
print(RMSE_train)
print(RMSE_test)
```

```
528660.0925841535
539305.8086488361
```

```
MAE_train = mean_absolute_error(y_train, predicciones_train)
MAE_test = mean_absolute_error(y_test, predicciones_test)
print(MAE_train)
print(MAE_test)
```

```
440593.6151924199
441573.7213534314
```

```
from sklearn.metrics import r2_score
```

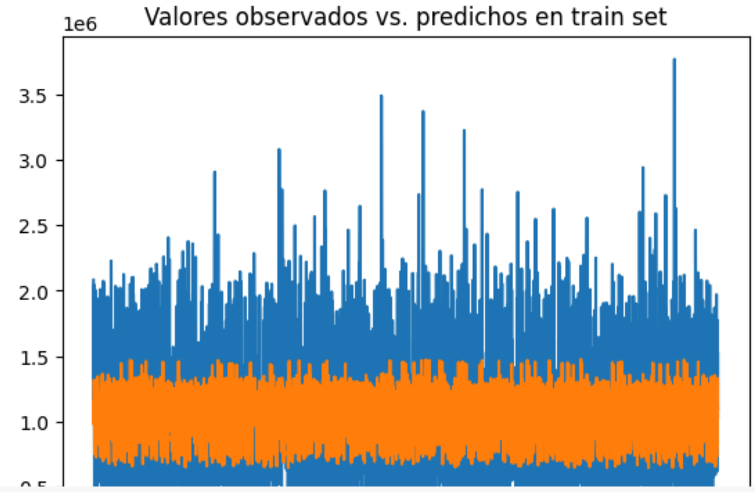
```
r_square_train = r2_score(y_train, predicciones_train)
r_square_test  = r2_score(y_test, predicciones_test)
print('El R^2 del subconjunto de entrenamiento es:' , r_square_train)
print('El R^2 del subconjunto de prueba es:' , r_square_test)
```

```
El R^2 del subconjunto de entrenamiento es: 0.12815867820026305
El R^2 del subconjunto de prueba es: 0.1288592511731088
```

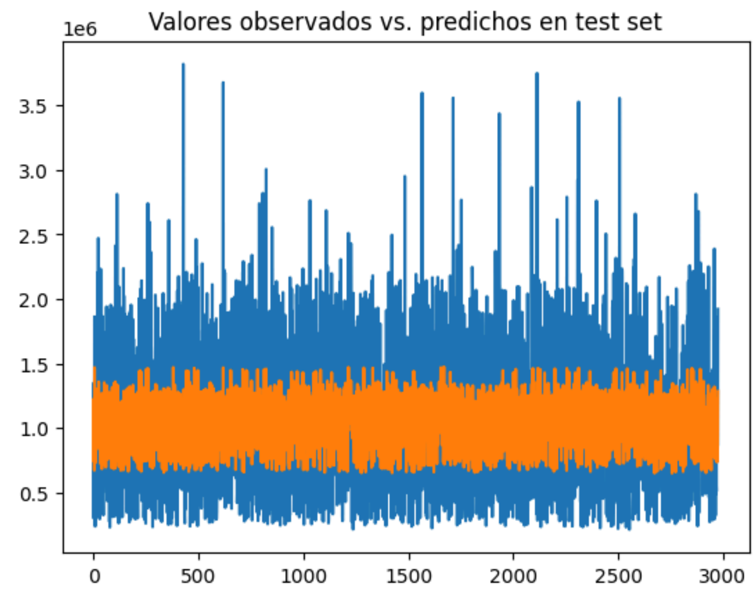
```
# Print the Intercept:
print('intercepto:', modelo_regresion.intercept_)

# Print the Slope:
print('pendiente:', modelo_regresion.coef_)
```

```
intercepto: 1852475.9196077145
pendiente: [ -2173.78725405  -1497.74103337  -1809.58087583   -793.00108395
 -15149.25033586]
```

```
fig, ax = plt.subplots()
ax.plot(y_train.values)
ax.plot(predicciones_train)
plt.title("Valores observados vs. predichos en train set");
```




```
fig, ax = plt.subplots()
ax.plot(y_test.values)
ax.plot(predicciones_test)
plt.title("Valores observados vs. predichos en test set");
```




```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
```

```
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

```
modelo_regresion_std = LinearRegression()
modelo_regresion_std.fit(X_train_std, y_train)
```

▾ LinearRegression