Ejercicio en Clase - Entregable

Se importaron las correspondientes librerias

```python
#Importar librerias
import pandas as pd    Import "pandas" could not be resolved from source
import matplotlib.pyplot as plt    Import "matplotlib.pyplot" could not be resolved from source
import seaborn as sns    Import "seaborn" could not be resolved from source
import numpy as np    Import "numpy" could not be resolved
```

✓ 0.0s

Previsualización de la base de datos

```python
data = pd.read_stata('joinby2.dta')
data
```

✓ 1.1s Python

	directorio	p5000	p5210s3	p5210s16	hogar	p4010	p4020	p4030s3	p4030s5	fex_c_2011	...	p7240s1	c
0	846845	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	764.317810	...		
1	846845	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	764.317810	...		
2	846846	2	no	no	1	ladrillo, bloque, material prefabricado, piedra	cemento, gravilla	sí	sí	895.285583	...	REBUSQUE	
3	846848	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	604.297668	...		
						ladrillo	baldosín,						

Descripcion de la base de datos:

```
data.describe()
```

✓ 0.5s Python

	directorio	p5000	hogar	fex_c_2011	id	secuencia_p	orden	p6016	30
count	30160.000000	30160.000000	30160.000000	30160.000000	30160.000000	30160.000000	30160.000000	30160.000000	
mean	862146.128316	3.470756	1.072016	585.542725	4568.112793	1.054178	2.825796	2.262798	
std	9886.042911	1.420929	0.319030	696.033936	2674.479004	0.280060	1.804467	1.499042	
min	846845.000000	1.000000	1.000000	57.128208	1.000000	1.000000	1.000000	1.000000	
25%	853985.000000	3.000000	1.000000	189.394516	2204.000000	1.000000	1.000000	1.000000	
50%	861118.000000	3.500000	1.000000	317.014343	4609.000000	1.000000	2.000000	2.000000	
75%	869375.000000	4.000000	1.000000	698.031677	6871.000000	1.000000	4.000000	3.000000	
max	883620.000000	33.000000	7.000000	5869.675293	9187.000000	7.000000	20.000000	19.000000	

Calculo de la linea de la pobreza del dataset original

```
#Calculo de la linea de pobreza
mediana_ingresos = data['inglabo'].median()
lp = mediana_ingresos / 2
print(f'Línea de Pobreza: {lp}')
```

✓ 0.0s Python

```
Línea de Pobreza: 250000.0
```

Definicion y prueba de la funcion para calcular el inidice de Foster-Greer-Thorbecke

```
alfa = [0, 1, 2]
```
✓ 0.1s

```
#Funcion para calcular fgt
def calculate_fgt(df, lp, alfa):
    x = []
    for ingreso in df['inglabo']:
        if ingreso < lp:
            x.append((1 - ingreso / lp) ** alfa)
        else:
            x.append(0)
    df['x'] = x
    FGT = df['x'].sum() / len(df)
    return FGT
```
✓ 0.2s

```
calculate_fgt(data, lp, alfa)
```
✓ 0.2s

```
/tmp/ipykernel_20945/3017090101.py:9: PerformanceWarning: DataFrame is high
  df['x'] = x
```

```
array([0.15477454, 0.07652315, 0.05041882])
```

Filtracion de la pobreza generalizada

```
#filtrar pobreza
def filtrar_pobreza(df):
    # Filtra el DataFrame para incluir solo aquellos con ingresos laborales menores a 250,000
    df_pobreza = df[df['inglabo'] < 250000]
    return df_pobreza
```
✓ 0.0s

Previsualizacion del DataFrame resultante:

```
data_filtrada = filtrar_pobreza(data)
data_filtrada
```

✓ 0.1s Python

	directorio	p5000	p5210s3	p5210s16	hogar	p4010	p4020	p4030s3	p4030s5	fex_c_2011	...	oci	oficio
26	846862	2	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	899.664490	...	1	45
28	846867	3	no	no	1	ladrillo, bloque, material prefabricado, piedra	cemento, gravilla	sí	sí	782.982178	...	1	45
29	846867	3	no	no	1	ladrillo, bloque, material prefabricado, piedra	cemento, gravilla	sí	sí	782.982178	...	1	45
30	846867	3	no	no	1	ladrillo, bloque, material prefabricado, piedra	cemento, gravilla	sí	sí	782.982178	...	1	45

Descripcion de los datos filtrados

```
data_filtrada.describe()
```

✓ 0.2s Python

	directorio	p5000	hogar	fex_c_2011	id	secuencia_p	orden	p6016	p603
count	4668.000000	4668.000000	4668.000000	4668.000000	4668.00000	4668.000000	4668.000000	4668.000000	4655.000
mean	862007.540060	3.216153	1.088475	487.956726	4517.26123	1.055698	2.847258	2.239503	6.52(
std	10013.419133	1.414888	0.363441	577.622498	2702.00415	0.281384	1.812398	1.498105	3.423
min	846862.000000	1.000000	1.000000	57.128208	11.00000	1.000000	1.000000	1.000000	1.000
25%	853887.000000	2.000000	1.000000	155.326141	2148.00000	1.000000	1.000000	1.000000	4.000
50%	859944.000000	3.000000	1.000000	291.942963	4387.00000	1.000000	2.000000	2.000000	6.000
75%	869124.000000	4.000000	1.000000	621.128601	6752.00000	1.000000	4.000000	3.000000	10.000
max	883612.000000	33.000000	5.000000	5685.456543	9183.00000	5.000000	14.000000	14.000000	12.000

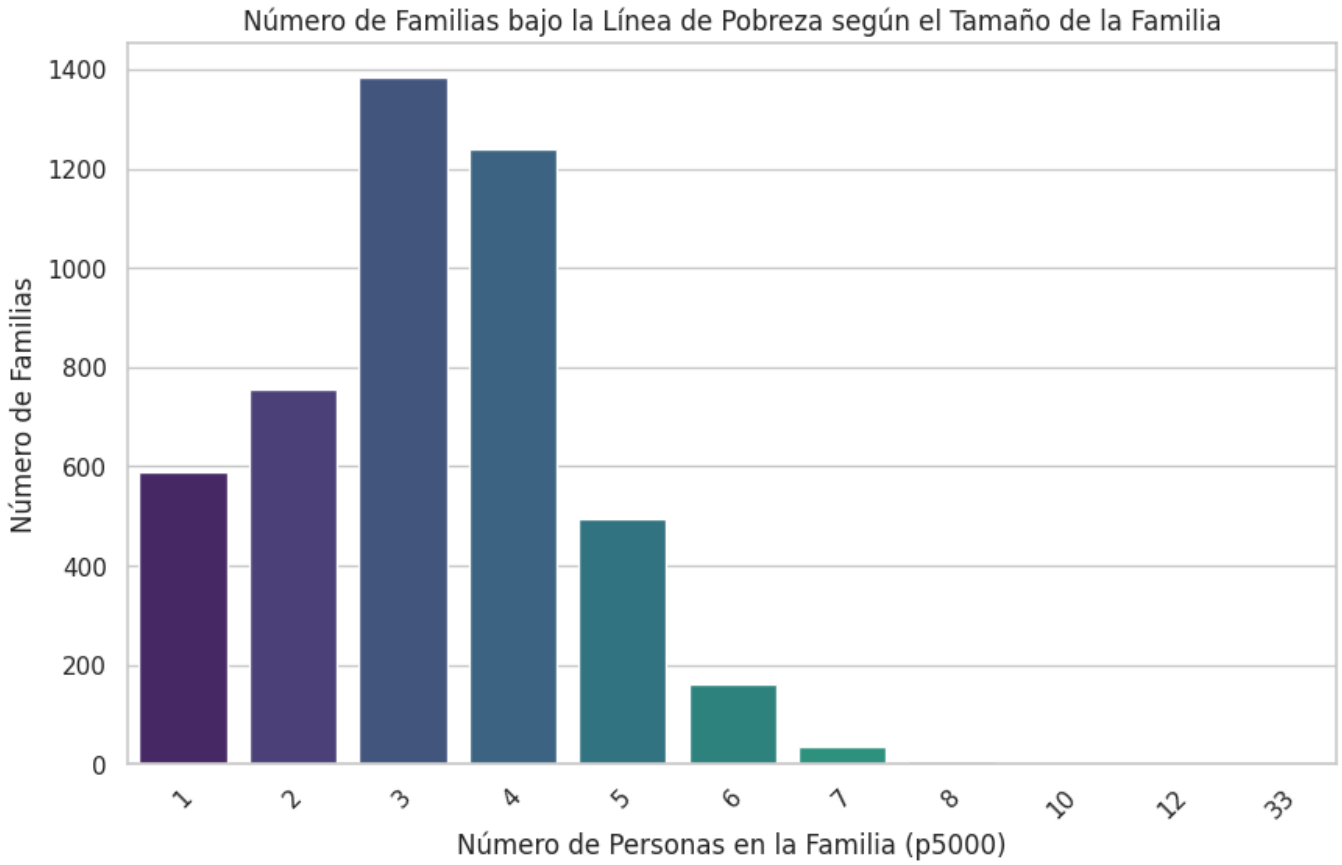
8 rows × 83 columns

Creacion del grafico de la suma total de familias bajo la linea de la pobreza segun el tamaño de la familia

Add Markdown Cell

```python
# Contar el número de familias para cada valor de 'p5000'
conteo_familias = data_filtrada['p5000'].value_counts().sort_index()

# Crear el diagrama de barras
plt.figure(figsize=(10, 6))
sns.barplot(x=conteo_familias.index, y=conteo_familias.values, palette="viridis")
plt.xlabel('Número de Personas en la Familia (p5000)')
plt.ylabel('Número de Familias')
plt.title('Número de Familias bajo la Línea de Pobreza según el Tamaño de la Familia')
plt.xticks(rotation=45)
plt.show()
```

✓ 0.4s Python



Filtracion por departamentos y creación de DataFrames dedicados:

```python
# Diccionario de departamentos más representativos
departamentos_representativos = {
    'Bogotá': '11',
    'Antioquia': '05',
    'ValleCauca': '76',
    'Atlántico': '08',
    'Bolívar': '13',
    'Santander': '68',
    'NorteSantander': '54',
    'Nariño': '52',
    'Meta': '50'
}
```

✓ 0.0s Python

```python
# Filtrar el dataset y crear dataframes específicos para cada departamento
data_filtrada_bogota = data[data['dpto'] == departamentos_representativos['Bogotá']]
data_filtrada_antioquia = data[data['dpto'] == departamentos_representativos['Antioquia']]
data_filtrada_valleCauca = data[data['dpto'] == departamentos_representativos['ValleCauca']]
data_filtrada_atlantico = data[data['dpto'] == departamentos_representativos['Atlántico']]
data_filtrada_bolivar = data[data['dpto'] == departamentos_representativos['Bolívar']]
data_filtrada_santander = data[data['dpto'] == departamentos_representativos['Santander']]
data_filtrada_norteSantander = data[data['dpto'] == departamentos_representativos['NorteSantander']]
data_filtrada_narino = data[data['dpto'] == departamentos_representativos['Nariño']]
data_filtrada_meta = data[data['dpto'] == departamentos_representativos['Meta']]
```

✓ 0.2s Python

Previsualizacion de la Data Filtrada

```python
#Visualización de la filtracion
data_filtrada_bogota
```

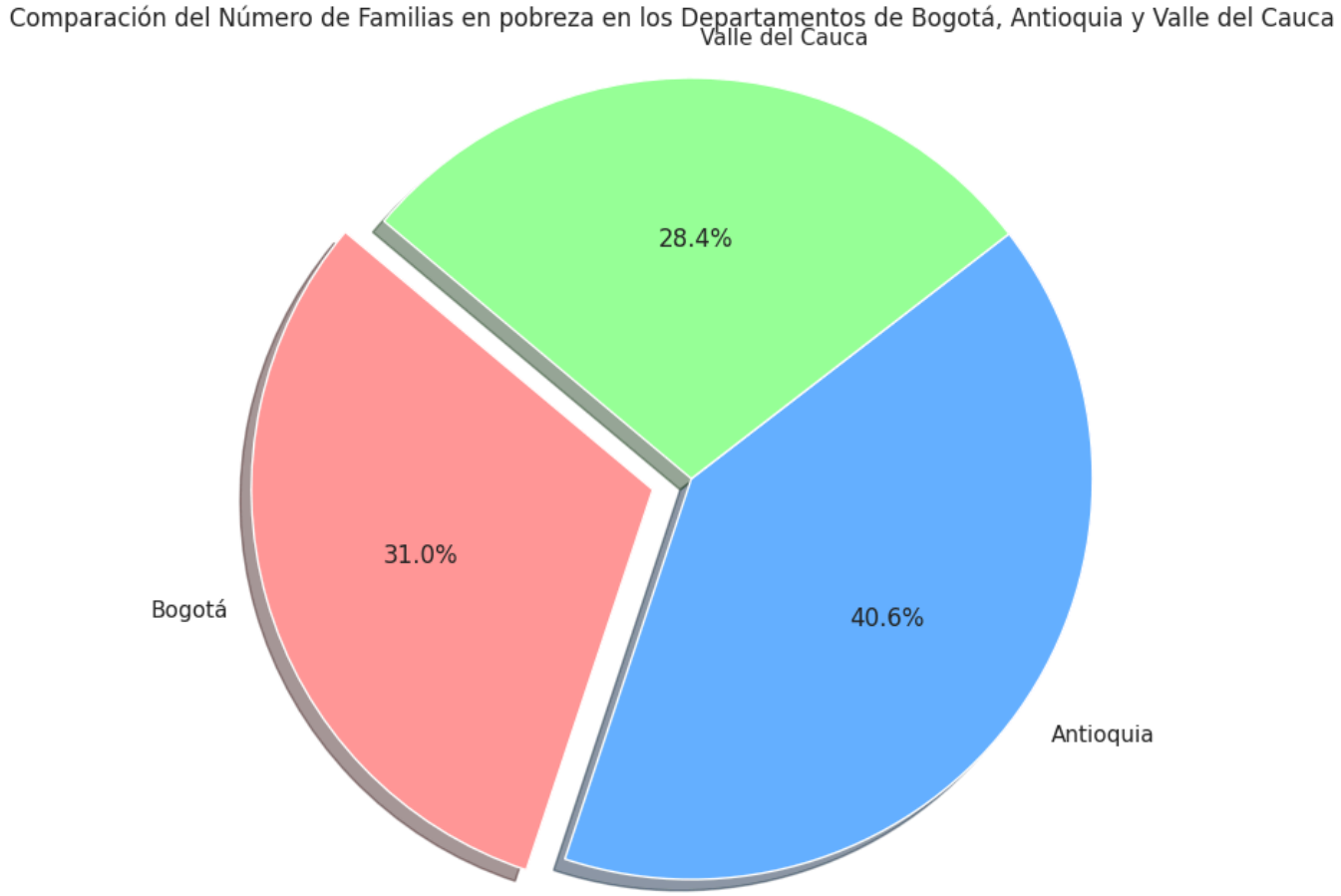
✓ 0.0s Python

	directorio	p5000	p5210s3	p5210s16	hogar	p4010	p4020	p4030s3	p4030s5	fex_c_2011	...	oci	oficio
3114	849634	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	2632.240967	...	1	97
3115	849634	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	2632.240967	...	1	95
3116	849634	4	no	no	1	ladrillo, bloque, material prefabricado, piedra	baldosín, ladrillo, vinisol, otros materiales ...	sí	sí	2632.240967	...	1	97
						ladrillo, bloque	baldosín, ladrillo,						

Creacion del grafico de Comparación del Número de Familias en pobreza en los Departamentos de Bogotá, Antioquia y Valle del Cauca

```python
# Filtrar el dataset y crear dataframes específicos para cada departamento
data_filtrada_bogota = data[data['dpto'] == departamentos_representativos['Bogotá']]
data_filtrada_antioquia = data[data['dpto'] == departamentos_representativos['Antioquia']]
data_filtrada_valleCauca = data[data['dpto'] == departamentos_representativos['ValleCauca']]

# Contar el número de registros (familias) en cada departamento
conteo_familias = {
    'Bogotá': len(data_filtrada_bogota),
    'Antioquia': len(data_filtrada_antioquia),
    'Valle del Cauca': len(data_filtrada_valleCauca)
}

# Crear un gráfico de pastel
labels = conteo_familias.keys()
sizes = conteo_familias.values()
colors = ['#ff9999','#66b3ff','#99ff99']
explode = (0.1, 0, 0)  # Explode the 1st slice (Bogotá)

plt.figure(figsize=(8, 8))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
        shadow=True, startangle=140)
plt.title('Comparación del Número de Familias en pobreza en los Departamentos de Bogotá, Antioquia y Valle del Cauc
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

✓ 0.3s Python

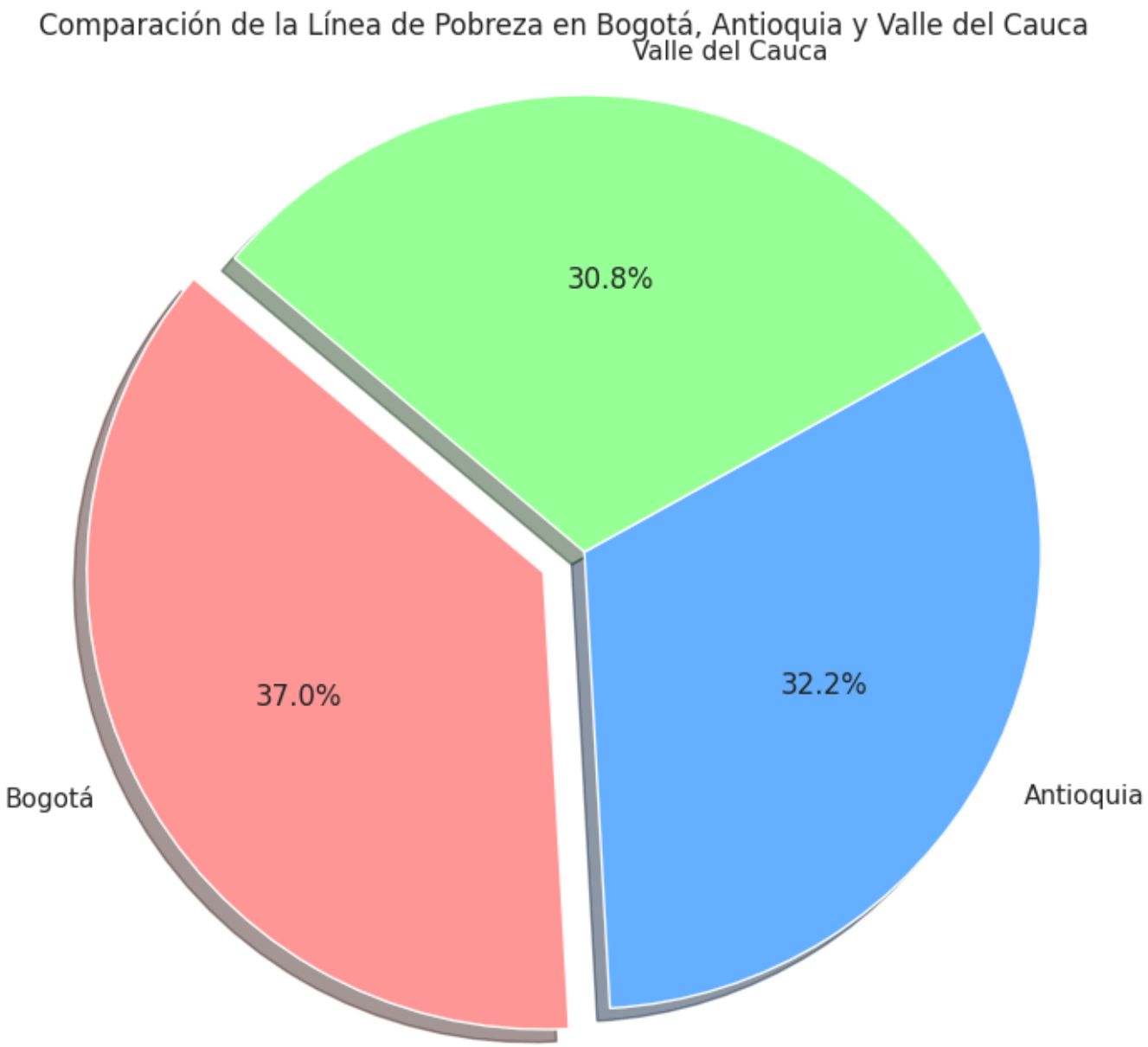



```
#Calcular lineas de pobreza dedicadas por departamento
#Filtrar el dataset y crear dataframes específicos para cada departamento

data_filtrada_bogota = data[data['dpto'] ==
departamentos_representativos['Bogotá']]
```

```
data_filtrada_antioquia = data[data['dpto'] ==
departamentos_representativos['Antioquia']]

data_filtrada_valleCauca = data[data['dpto'] ==
departamentos_representativos['ValleCauca']]



# Función para calcular la línea de pobreza

def calcular_linea_pobreza(data):

mediana_ingresos = data['inglabo'].median()

linea_pobreza = mediana_ingresos / 2

return linea_pobreza



# Calcular la línea de pobreza para cada departamento

lp_bogota = calcular_linea_pobreza(data_filtrada_bogota)

lp_antioquia = calcular_linea_pobreza(data_filtrada_antioquia)

lp_valleCauca = calcular_linea_pobreza(data_filtrada_valleCauca)



print(f"Línea de Pobreza en Bogotá: {lp_bogota}")

print(f"Línea de Pobreza en Antioquia: {lp_antioquia}")

print(f"Línea de Pobreza en Valle del Cauca: {lp_valleCauca}")



# Crear un gráfico de pastel para comparar la línea de pobreza entre los
departamentos

lineas_pobreza = [lp_bogota, lp_antioquia, lp_valleCauca]

labels = ['Bogotá', 'Antioquia', 'Valle del Cauca']

colors = ['#ff9999','#66b3ff','#99ff99']
```

```
explode = (0.1, 0, 0) # Explode the 1st slice (Bogotá)



plt.figure(figsize=(8, 8))

plt.pie(lineas_pobreza, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%',

shadow=True, startangle=140)

plt.title('Comparación de la Línea de Pobreza en Bogotá, Antioquia y Valle
del Cauca')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a
circle.

plt.show()
```

```python
#Calcular lineas de pobreza dedicadas por departamento
# Filtrar el dataset y crear dataframes específicos para cada departamento
data_filtrada_bogota = data[data['dpto'] == departamentos_representativos['Bogotá']]
data_filtrada_antioquia = data[data['dpto'] == departamentos_representativos['Antioquia']]
data_filtrada_valleCauca = data[data['dpto'] == departamentos_representativos['ValleCauca']]

# Función para calcular la línea de pobreza
def calcular_linea_pobreza(data):
    mediana_ingresos = data['inglabo'].median()
    linea_pobreza = mediana_ingresos / 2
    return linea_pobreza

# Calcular la línea de pobreza para cada departamento
lp_bogota = calcular_linea_pobreza(data_filtrada_bogota)
lp_antioquia = calcular_linea_pobreza(data_filtrada_antioquia)
lp_valleCauca = calcular_linea_pobreza(data_filtrada_valleCauca)

print(f"Línea de Pobreza en Bogotá: {lp_bogota}")
print(f"Línea de Pobreza en Antioquia: {lp_antioquia}")
print(f"Línea de Pobreza en Valle del Cauca: {lp_valleCauca}")

# Crear un gráfico de pastel para comparar la línea de pobreza entre los departamentos
lineas_pobreza = [lp_bogota, lp_antioquia, lp_valleCauca]
labels = ['Bogotá', 'Antioquia', 'Valle del Cauca']
colors = ['#ff9999','#66b3ff','#99ff99']
explode = (0.1, 0, 0)  # Explode the 1st slice (Bogotá)

plt.figure(figsize=(8, 8))
```



Calcular la linea de pobreza de cada departamento relevante:

```python
# Función para calcular la línea de pobreza
def calcular_linea_pobreza(data):
    mediana_ingresos = data['inglabo'].median()
    linea_pobreza = mediana_ingresos / 2
    return linea_pobreza

# Calcular la línea de pobreza para cada departamento
lineas_pobreza = {}
for nombre, codigo in departamentos_representativos.items():
    data_filtrada = data[data['dpto'] == codigo]
    lineas_pobreza[nombre] = calcular_linea_pobreza(data_filtrada)

# Mostrar la línea de pobreza para cada departamento
for nombre, lp in lineas_pobreza.items():
    print(f"Línea de Pobreza en {nombre}: {lp}")
```

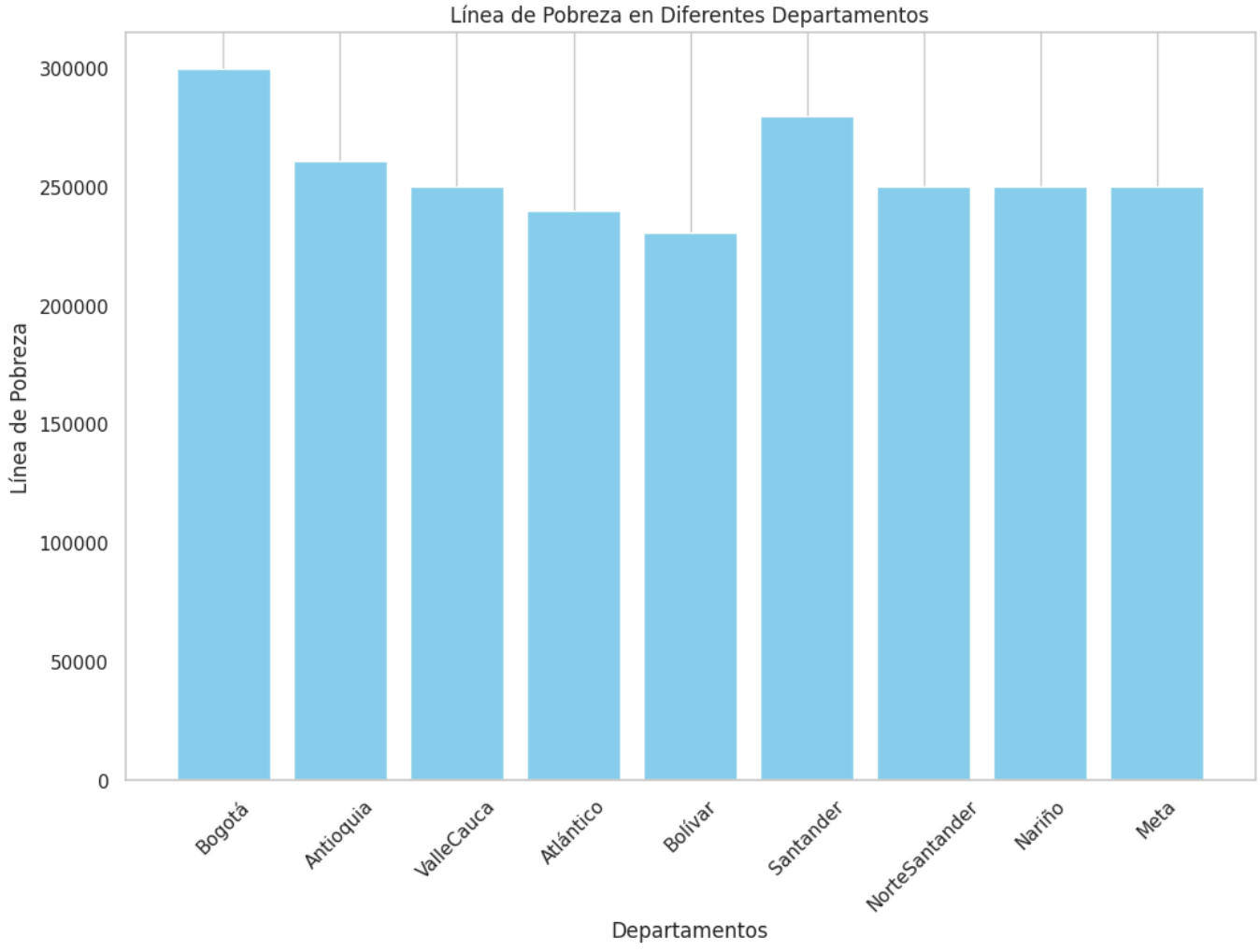
✓ 0.1s Python

```
Línea de Pobreza en Bogotá: 300000.0
Línea de Pobreza en Antioquia: 260750.0
Línea de Pobreza en ValleCauca: 250000.0
Línea de Pobreza en Atlántico: 240000.0
Línea de Pobreza en Bolívar: 230750.0
Línea de Pobreza en Santander: 280000.0
Línea de Pobreza en NorteSantander: 250000.0
Línea de Pobreza en Nariño: 250000.0
Línea de Pobreza en Meta: 250000.0
```

Creacion del grafico de comparacion de las diferentes lineas de la pobreza en los departamentos relevantes

```python
# Crear listas de departamentos y sus respectivas líneas de pobreza
departamentos = list(lineas_pobreza.keys())
lineas_pobreza_valores = list(lineas_pobreza.values())

# Crear un gráfico de barras
plt.figure(figsize=(12, 8))
plt.bar(departamentos, lineas_pobreza_valores, color='skyblue')
plt.xlabel('Departamentos')
plt.ylabel('Línea de Pobreza')
plt.title('Línea de Pobreza en Diferentes Departamentos')
plt.xticks(rotation=45)
plt.grid(axis='y')

# Mostrar el gráfico
plt.show()
```

✓ 0.3s



Calculo del indice FGT para cada departamento relevante

```python
# Calcular FGT para cada departamento
fgt_resultados = {}
alfa = 2  # Puedes ajustar el valor de alfa según sea necesario
for nombre, codigo in departamentos_representativos.items():
    data_filtrada = data[data['dpto'] == codigo]
    lp = lineas_pobreza[nombre]
    fgt_resultados[nombre] = calculate_fgt(data_filtrada, lp, alfa)

    # Crear un DataFrame a partir de los resultados de FGT
df_fgt_resultados = pd.DataFrame(list(fgt_resultados.items()), columns=['Departamento', 'FGT'])

# Mostrar el DataFrame con los resultados de FGT
print(df_fgt_resultados)
```

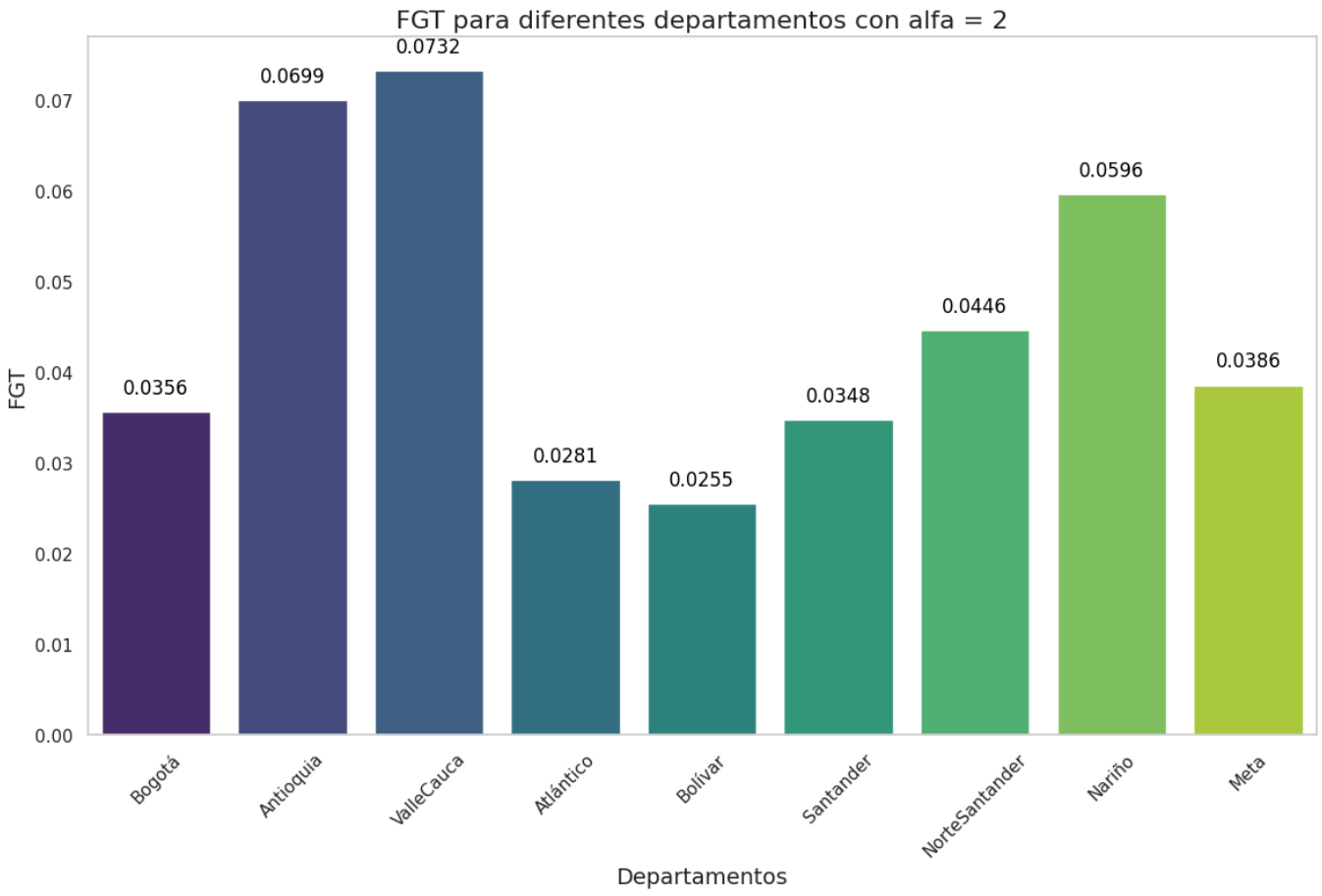
✓ 0.1s

	Departamento	FGT
0	Bogotá	0.035566
1	Antioquia	0.069909
2	ValleCauca	0.073237
3	Atlántico	0.028092
4	Bolívar	0.025473
5	Santander	0.034768
6	NorteSantander	0.044605
7	Nariño	0.059600
8	Meta	0.038556

Creacion del grafico de FGT para diferentes departamentos con alfa = 2

```python
sns.set(style="whitegrid")
plt.figure(figsize=(14, 8))
barplot = sns.barplot(x='Departamento', y='FGT', data=df_fgt_resultados, palette='viridis')

for index, row in df_fgt_resultados.iterrows():
    barplot.text(row.name, row.FGT + 0.002, round(row.FGT, 4), color='black', ha="center")

plt.title('FGT para diferentes departamentos con alfa = 2', fontsize=16)
plt.xlabel('Departamentos', fontsize=14)
plt.ylabel('FGT', fontsize=14)
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

✓ 0.5s Python



Creacion de la funcion para el calculo del coeficiente de Gini

```python
#Funcion para calcular el coeficiente de Gini
def coeficiente_gini(x):
    sorted_x = np.sort(x)
    n = len(x)
    cumx = np.cumsum(sorted_x)
    gini = (n + 1 - 2 * (np.sum(cumx) / cumx[-1])) / n
    return gini
```

✓ 0.0s Python

```python
# Diccionario para almacenar los coeficientes de Gini
gini_resultados = {}

# Calcular el coeficiente de Gini para cada departamento
for nombre, codigo in departamentos_representativos.items():
    data_filtrada = data[data['dpto'] == codigo]
    ingresos = data_filtrada['inglabo'].dropna().values
    gini_resultados[nombre] = coeficiente_gini(ingresos)

# Mostrar los resultados del coeficiente de Gini para cada departamento
for nombre, gini in gini_resultados.items():
    print(f"Coeficiente de Gini para {nombre}: {gini}")

# Crear un DataFrame a partir de los resultados del coeficiente de Gini
df_gini_resultados = pd.DataFrame(list(gini_resultados.items()), columns=['Departamento', 'Gini'])
```

✓ 0.2s Python

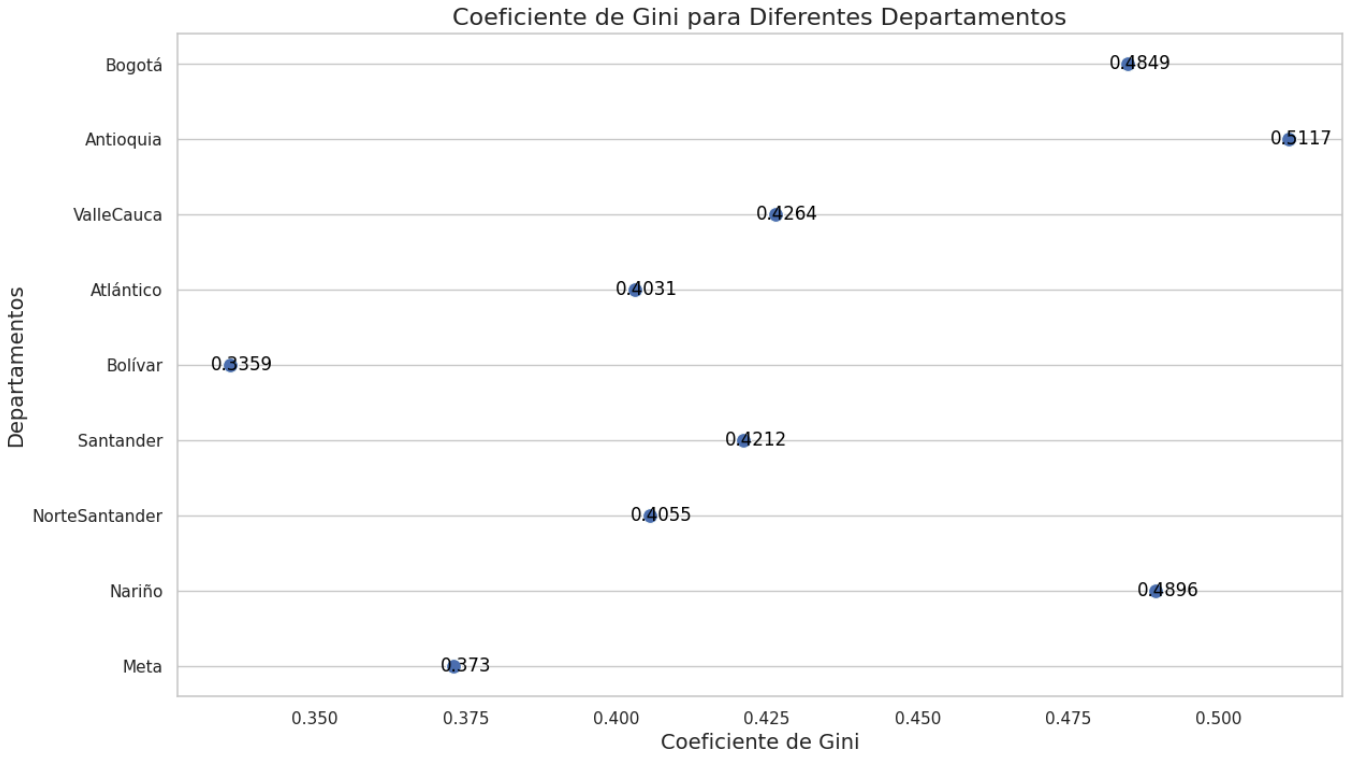
Resultados

```
Coeficiente de Gini para Bogotá: 0.4849458185696245
Coeficiente de Gini para Antioquia: 0.5117099752719432
Coeficiente de Gini para ValleCauca: 0.42644600142800815
Coeficiente de Gini para Atlántico: 0.4031030547138126
Coeficiente de Gini para Bolívar: 0.33590185719686494
Coeficiente de Gini para Santander: 0.42115668553699465
Coeficiente de Gini para NorteSantander: 0.4055392125968554
Coeficiente de Gini para Nariño: 0.4895699609579075
Coeficiente de Gini para Meta: 0.3729620673987726
```

Creacion del grafico Scatter plot de dispersion para representar los coeficientes de Gini calculados anteriormente

```python
sns.set(style="whitegrid")
plt.figure(figsize=(14, 8))
scatterplot = sns.scatterplot(x='Gini', y='Departamento', data=df_gini_resultados, s=100, color='b', marker='o')

for index, row in df_gini_resultados.iterrows():
    scatterplot.text(row.Gini + 0.002, row.Departamento, round(row.Gini, 4), color='black', ha="center", va='center

plt.title('Coeficiente de Gini para Diferentes Departamentos', fontsize=16)
plt.xlabel('Coeficiente de Gini', fontsize=14)
plt.ylabel('Departamentos', fontsize=14)
plt.grid(axis='x')
plt.show()
```

✓ 0.4s Python



Conclusiones sobre la Línea de Pobreza:

1. **Variación de la Línea de Pobreza**:
   - La línea de pobreza varía significativamente entre los departamentos. Por ejemplo, Bogotá tiene la línea de pobreza más alta con 300,000, mientras que Bolívar tiene una de las más bajas con 230,750. Esto refleja las diferencias en los niveles de ingreso medianos entre los departamentos.

Conclusiones sobre FGT (Foster-Greer-Thorbecke):

2. **FGT y Pobreza Extrema**:
   - Los valores de FGT indican la incidencia y profundidad de la pobreza en cada departamento. ValleCauca tiene el FGT más alto con 0.0732, lo que sugiere una

mayor proporción de individuos por debajo de la línea de pobreza y una mayor profundidad de pobreza en comparación con otros departamentos.
- Bogotá tiene un FGT relativamente bajo (0.0356), lo que sugiere que, aunque la línea de pobreza es más alta, una menor proporción de la población está por debajo de esta línea.

Conclusiones sobre el Coeficiente de Gini:

3. **Desigualdad de Ingresos**:
   - El coeficiente de Gini es una medida de la desigualdad de ingresos. Antioquia y ValleCauca tienen los coeficientes de Gini más altos, indicando una mayor desigualdad de ingresos en comparación con otros departamentos.
   - Atlántico y Bolívar tienen los coeficientes de Gini más bajos, lo que sugiere una distribución de ingresos relativamente más equitativa.

Observaciones Generales:

4. **Distribución de Ingresos**:
   - La distribución de ingresos muestra variaciones significativas entre los departamentos. Bogotá, con una línea de pobreza alta, sugiere una concentración de ingresos relativamente alta, mientras que departamentos como Bolívar y Atlántico tienen ingresos medianos más bajos.
5. **Tamaño de Familia y Pobreza**:
   - El análisis del tamaño de familia (`p5000`) muestra que la mayoría de las familias bajo la línea de pobreza tienen un tamaño de familia pequeño, lo que podría indicar que familias más grandes tienen mejores ingresos o que las políticas de apoyo familiar no están suficientemente orientadas a familias más pequeñas.

Gráficos y Visualizaciones:

6. **Gráfico de FGT**:
   - El gráfico de barras muestra claramente que departamentos como ValleCauca, Antioquia y Nariño tienen valores de FGT más altos, indicando problemas más profundos de pobreza extrema.
7. **Gráfico del Coeficiente de Gini**:
   - El gráfico de puntos del coeficiente de Gini visualiza la desigualdad de ingresos, destacando departamentos con mayor desigualdad como Antioquia y ValleCauca.

Sugerencias para Políticas:

8. **Focalización de Políticas**:
    - Las políticas públicas deben ser focalizadas según las necesidades específicas de cada departamento. Departamentos con alta desigualdad (como Antioquia) pueden necesitar políticas que aborden la distribución de ingresos, mientras que departamentos con alta pobreza extrema (como ValleCauca) necesitan políticas que aumenten los ingresos de los más pobres.
9. **Apoyo a Familias Pequeñas**:
    - Dado que las familias pequeñas parecen ser más vulnerables a la pobreza, se podrían implementar programas específicos de apoyo para estas familias, incluyendo subsidios directos y acceso a servicios básicos.