EDA

July 23, 2025

[11]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[12]:
```
df = pd.read_csv("utilizacionHDN.csv")
```

[13]:
```
df.head()
```

[13]:

	mes	numero-mes	year	fecha	total-por-genero	hombre	mujer	\
0	enero	1	2022	1-01-2022	6252	3422	2830	
1	febrero	2	2022	2-01-2022	4235	2293	1942	
2	marzo	3	2022	3-01-2022	6172	3401	2771	
3	abril	4	2022	4-01-2022	6372	3512	2860	
4	mayo	5	2022	5-01-2022	7830	4157	3673	

	total-por-rango-de-edad	menor de 1	1 a 4	5 a 9	10 a 14	15 y mas
0	6252	1331	2741	1481	695	4
1	4235	856	1874	1028	474	3
2	6172	1231	2745	1515	677	4
3	6372	1339	2814	1493	720	6
4	7830	1536	3447	1968	878	1

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mes                     36 non-null     object
 1   numero-mes              36 non-null     int64
 2   year                    36 non-null     int64
 3   fecha                   36 non-null     object
 4   total-por-genero        36 non-null     int64
 5   hombre                  36 non-null     int64
 6   mujer                   36 non-null     int64
 7   total-por-rango-de-edad 36 non-null     int64
```

```
 8   menor de 1               36 non-null     int64
 9   1 a 4                    36 non-null     int64
 10  5 a 9                    36 non-null     int64
 11  10 a 14                  36 non-null     int64
 12  15 y mas                 36 non-null     int64
dtypes: int64(11), object(2)
memory usage: 3.8+ KB
```

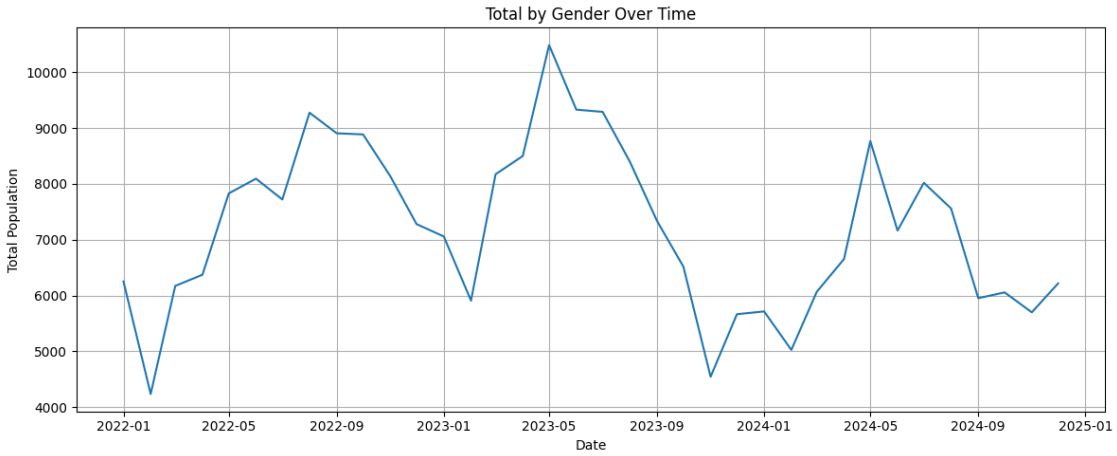
[17]:
```
df["fecha"]=pd.to_datetime(df["fecha"])
```

[18]:
```
df.info()
```

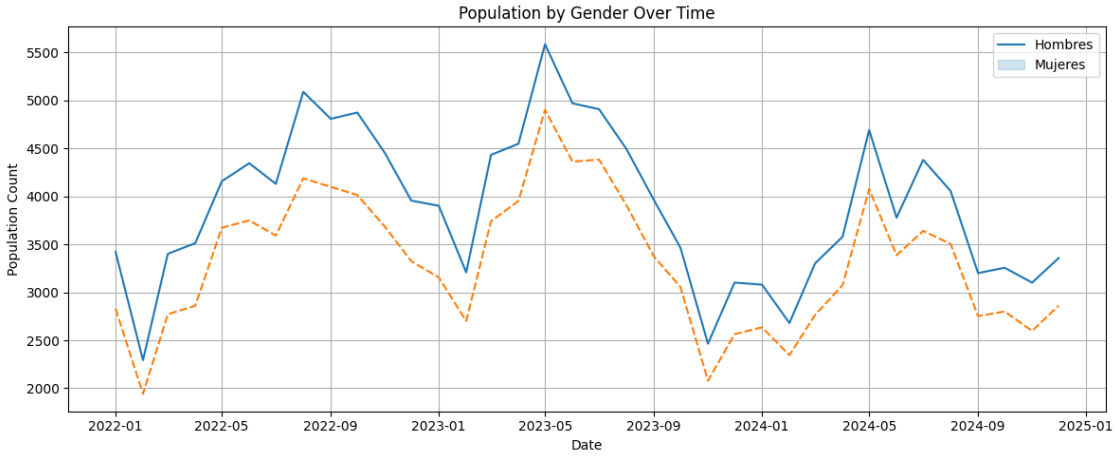
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mes                      36 non-null     object
 1   numero-mes               36 non-null     int64
 2   year                     36 non-null     int64
 3   fecha                    36 non-null     datetime64[ns]
 4   total-por-genero         36 non-null     int64
 5   hombre                   36 non-null     int64
 6   mujer                    36 non-null     int64
 7   total-por-rango-de-edad  36 non-null     int64
 8   menor de 1               36 non-null     int64
 9   1 a 4                    36 non-null     int64
 10  5 a 9                    36 non-null     int64
 11  10 a 14                  36 non-null     int64
 12  15 y mas                 36 non-null     int64
dtypes: datetime64[ns](1), int64(11), object(1)
memory usage: 3.8+ KB
```

[19]:
```
df = df.sort_values('fecha')
df.set_index('fecha', inplace=True)
```

[20]:
```
plt.figure(figsize=(12,5))
sns.lineplot(data=df, y='total-por-genero', x=df.index)
plt.title('Total by Gender Over Time')
plt.ylabel('Total Population')
plt.xlabel('Date')
plt.grid(True)
plt.tight_layout()
plt.show()
```

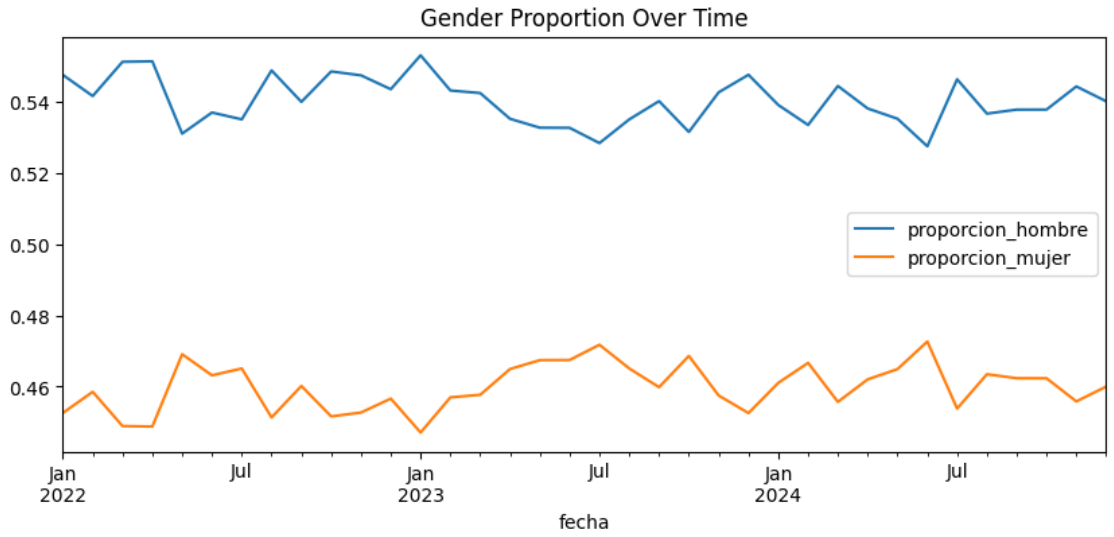


[21]:
```
plt.figure(figsize=(12,5))
sns.lineplot(data=df[['hombre', 'mujer']])
plt.title('Population by Gender Over Time')
plt.xlabel('Date')
plt.ylabel('Population Count')
plt.legend(['Hombres', 'Mujeres'])
plt.grid(True)
plt.tight_layout()
plt.show()
```

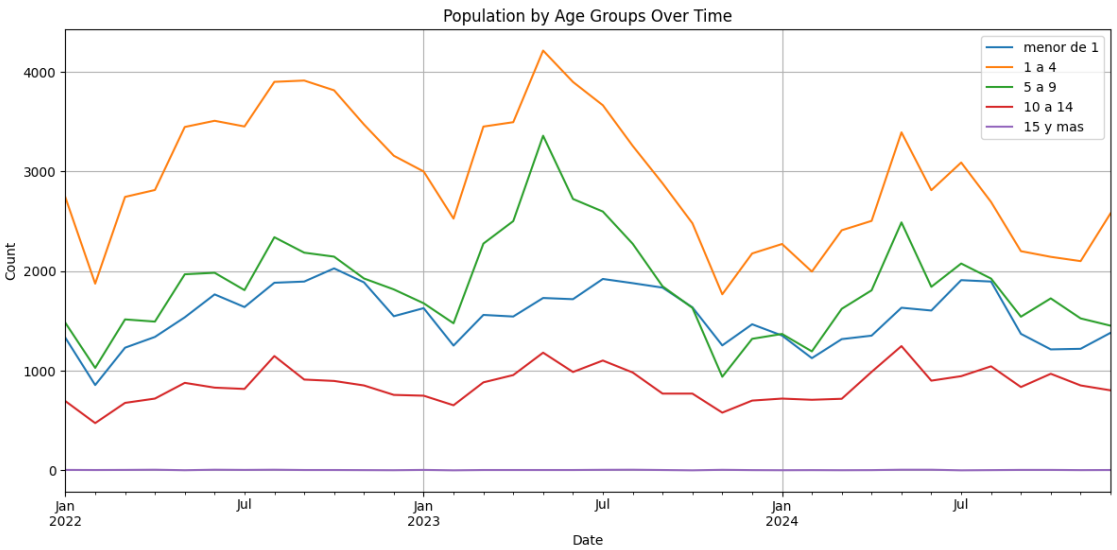


[22]:
```
df['proporcion_hombre'] = df['hombre'] / df['total-por-genero']
df['proporcion_mujer'] = df['mujer'] / df['total-por-genero']
df[['proporcion_hombre', 'proporcion_mujer']].plot(figsize=(10,4),␣
↪title="Gender Proportion Over Time")
```

[22]: <Axes: title={'center': 'Gender Proportion Over Time'}, xlabel='fecha'>



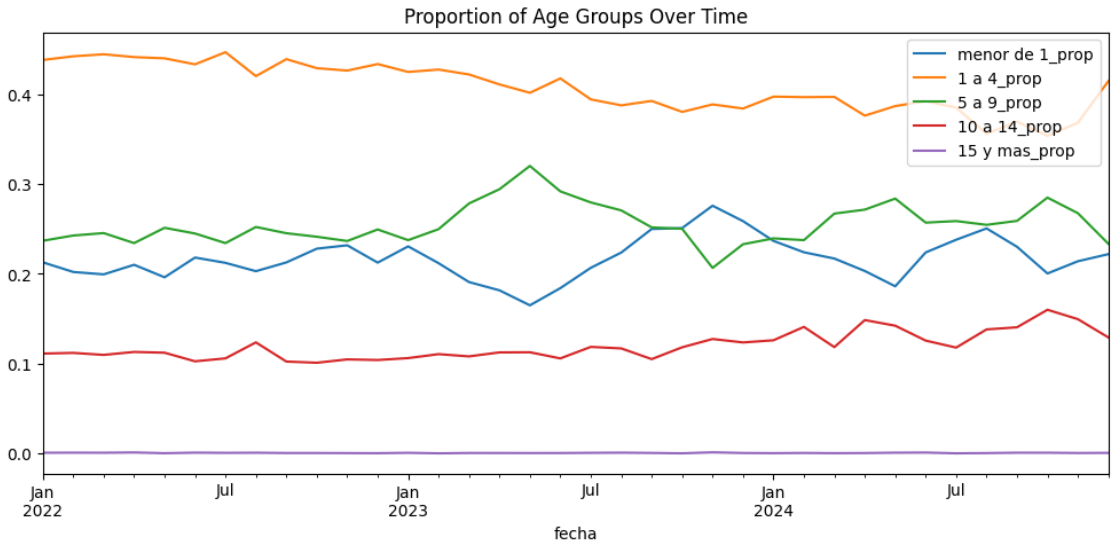
[23]:
```
age_columns = ['menor de 1', '1 a 4', '5 a 9', '10 a 14', '15 y mas']
df[age_columns].plot(figsize=(12,6), title='Population by Age Groups Over Time')
plt.ylabel('Count')
plt.xlabel('Date')
plt.grid(True)
plt.tight_layout()
```



[24]:
```
for col in age_columns:
    df[f'{col}_prop'] = df[col] / df['total-por-rango-de-edad']

df[[f'{col}_prop' for col in age_columns]].plot(figsize=(12,5),␣
 ↪title='Proportion of Age Groups Over Time')
```

[24]: <Axes: title={'center': 'Proportion of Age Groups Over Time'}, xlabel='fecha'>



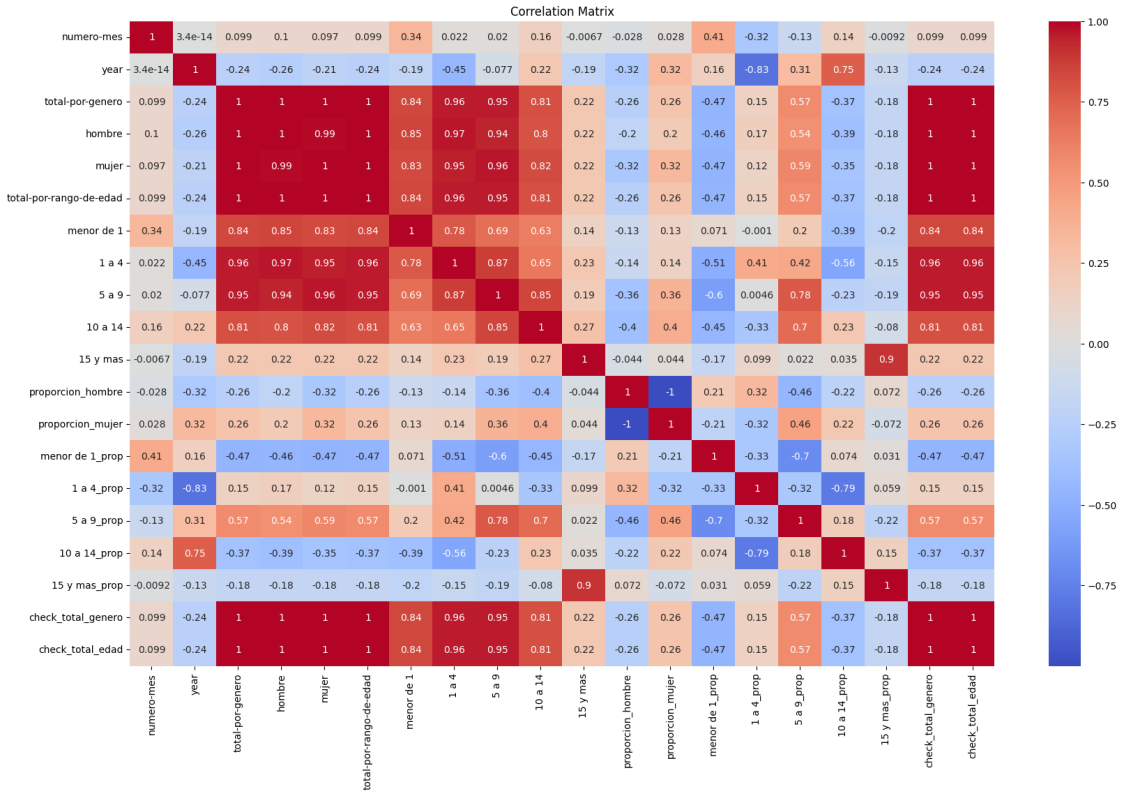
[27]:
```
from statsmodels.tsa.stattools import adfuller

# Check if total is stationary
result = adfuller(df['total-por-genero'])
print('ADF Statistic:', result[0])
print('p-value:', result[1])
```

```
ADF Statistic: -1.2665510522826655
p-value: 0.644332988679244
```

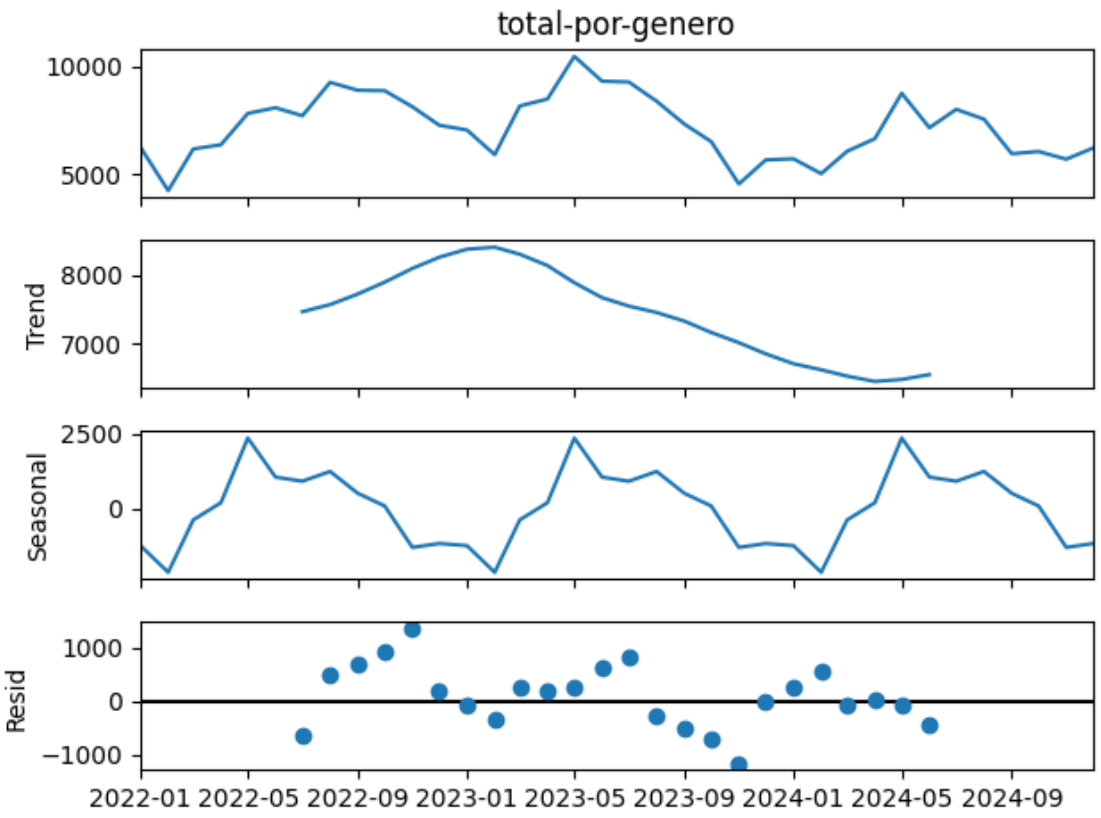
[31]:
```
plt.figure(figsize=(20,12))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
```

[31]: Text(0.5, 1.0, 'Correlation Matrix')



[30]:
```
import statsmodels.api as sm

decomposition = sm.tsa.seasonal_decompose(df['total-por-genero'],␣
 ↪model='additive', period=12)
decomposition.plot()
plt.show()
```


total-por-genero
10000
5000
Trend
8000
7000
Seasonal
2500
0
Resid
1000
0
−1000
2022-01
2022-05
2022-09
2023-01
2023-05
2023-09
2024-01
2024-05
2024-09