

```
In [152.. import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
sns.set_theme(color_codes=True)
pd.set_option('display.max_columns', None)
```

Limpieza de datos

```
In [153.. df = pd.read_excel('Top_1000_companies_DataSet.xlsx')
```

```
In [154.. df.drop_duplicates() #no hay duplicados
df.dtypes # los formatos/tipos de datos
```

```
Out[154]: company_name      object
url      object
city     object
state    object
country  object
employees      object
linkedin_url  object
founded       object
Industry      object
GrowjoRanking object
Previous Ranking object
estimated_revenues object
job_openings  object
keywords      object
LeadInvestors object
Accelerator   object
btype         object
valuation     float64
total_funding object
product_url   object
indeed_url    object
growth_percentage float64
contact_info  object
dtype: object
```

Eliminar duplicados

```
In [155.. df.drop_duplicates()
```

Out[155]:

	company_name	url	city	state	country	employees	linkedin_url
0	OpenAI	openai.com	San Francisco	CA	United States	655	http://www.linkedin.com/company/openai
1	Alchemy	alchemy.com	San Francisco	CA	United States	201	http://www.linkedin.com/company/alchemyinc
2	dbt Labs	getdbt.com	Philadelphia	PA	United States	511	http://www.linkedin.com/company/dbtlabs
3	Wasabi Technologies	wasabi.com	Boston	MA	United States	355	http://www.linkedin.com/company/wasabitechnolo...
4	Whatnot	whatnot.com	Los Angeles	CA	United States	551	http://www.linkedin.com/company/whatnot-inc
...
976	Forte	forte.io	San Francisco	CA	United States	145	http://www.linkedin.com/company/forte-labs-inc
977	Collective Health	collectivehealth.com	San Francisco	CA	United States	615	http://www.linkedin.com/company/collectivehealth
978	NaN	Google Ventures	NaN	NaN	1500000000	\$719M	https://www.growjo.com/company/Collective_Health https://www.q=c
979	Fathom (YC W21)	fathom.video	San Francisco	CA	USA	96	http://www.linkedin.com/company/fathom-video
980	Hone	honehq.com	San Francisco	CA	United States	179	http://www.linkedin.com/company/honehq

981 rows × 23 columns

Deshacerse ede las fials que contengan datos nan y reamplzar en blanco

In [156..

```
df = df.fillna('')
```

Eliminar las filas que no contengan datos de la tabla principal

In [157..

```
for x in df.index:
    if df.loc[x, "company_name"] == '':
        df.drop(x, inplace=True)
```

Eliminar todos los datos diferentes que no esten en el abecedario o en los numeros del 1 al 9

In [158..

```
df["company_name"] = df["company_name"].str.replace('[^a-zA-Z0-9]', ' ')
df["city"] = df["city"].str.replace('[^a-zA-Z0-9]', ' ')

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\1779931556.py:1: FutureWarning:
The default value of regex will change from True to False in a future version.
C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\1779931556.py:2: FutureWarning:
The default value of regex will change from True to False in a future version.
```

Revisar la informacion de los datos

In [159..

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 965 entries, 0 to 980
Data columns (total 23 columns):
#   Column                Non-Null Count  Dtype
---  -
0   company_name          965 non-null   object
1   url                   965 non-null   object
2   city                  965 non-null   object
3   state                 965 non-null   object
4   country               965 non-null   object
5   employees              965 non-null   object
6   linkedin_url          965 non-null   object
7   founded               965 non-null   object
8   Industry              965 non-null   object
9   GrowjoRanking         965 non-null   object
10  Previous Ranking      965 non-null   object
11  estimated_revenues    965 non-null   object
12  job_openings          965 non-null   object
13  keywords              965 non-null   object
14  LeadInvestors         965 non-null   object
15  Accelerator           965 non-null   object
16  btype                 965 non-null   object
17  valuation             965 non-null   object
18  total_funding         965 non-null   object
19  product_url           965 non-null   object
20  indeed_url            965 non-null   object
21  growth_percentage     965 non-null   object
22  contact_info          965 non-null   object
dtypes: object(23)
memory usage: 213.2+ KB
```

Eliminar las columnas que no necesitamos

```
In [160.. columnas_a_eliminar = ["contact_info", "product_url", "indeed_url", "Accelerator", "btype", "keywords", "linkedin_url"]
df = df.drop(columns = columnas_a_eliminar)
df = df.drop(680)
df = df.drop(688)
df = df.drop(738)
df = df.drop(740)
```

Resetear los index porque se acababa de eliminar el index 680

```
In [161.. df = df.reset_index(drop=True)
```

convertir el formato de algunas columnas

Principalmente convertirlas en type str excepto GrowjoRanking que ya es de type int →↓

para poder manipularlas y aplicarles filtros en s

```
In [162.. df["employees"] = df["employees"].apply(lambda x: str(x))
df["founded"] = df["founded"].apply(lambda x: str(x))
df["estimated_revenues"] = df["estimated_revenues"].apply(lambda x: str(x))
df["job_openings"] = df["job_openings"].apply(lambda x: str(x))
df["growth_percentage"] = df["growth_percentage"].apply(lambda x: str(x))
```

Reemplazar valores desconocidos por los valores 0 para int y desconocido para str

```
In [163.. df["founded"] = df["founded"].replace('', '0')
df["state"] = df["state"].replace('', 'desconocido')
df["city"] = df["city"].replace('', 'desconocido')
df["country"] = df["country"].replace('', 'desconocido')
df["Industry"] = df["Industry"].replace('', 'desconocido')
df["estimated_revenues"] = df["estimated_revenues"].replace('', '0')
df["job_openings"] = df["job_openings"].replace('', '0')
df["LeadInvestors"] = df["LeadInvestors"].replace('', 'desconocido')
df["valuation"] = df["valuation"].replace('', '0')
df["total_funding"] = df["total_funding"].replace('', '0')
df["growth_percentage"] = df["growth_percentage"].replace('', '0')
```

Convertir las columnas de numeros a enteros

```
In [164.. df["employees"] = df["employees"].apply(lambda x: int(x))
df["GrowjoRanking"] = df["GrowjoRanking"].apply(lambda x: int(x))
df["founded"] = df["founded"].apply(lambda x: int(x))
df["valuation"] = df["valuation"].apply(lambda x: int(x))
df["estimated_revenues"] = df["estimated_revenues"].apply(lambda x: float(x))
df["growth_percentage"] = df["growth_percentage"].apply(lambda x: float(x))
df["job_openings"] = df["job_openings"].apply(lambda x: int(x))
```

Eliminar los caracteres extraños en la columna de tota_funding

```
In [165.. df['total_funding'] = df['total_funding'].str.replace(r'[^0-9MmBb.]', '', regex=True)
```

Necesitamos convertir la columna total_funding en una columna que solo contenga numeros

```
In [166.. df['total_funding'] = df['total_funding'].replace({'M': 'e6', 'B': 'e9'}, regex=True)
df['total_funding'] = pd.to_numeric(df['total_funding'], errors='coerce')
df["total_funding"] = df["total_funding"].apply(lambda x: int(x))
```

Este codigo es preferencial para convertir los datos limpios en.xlsx

```
In [167.. output_file = "top_companies_cleaning.xlsx"
df.to_excel(output_file, index=False)
```

```
In [168.. df.select_dtypes(include='object').nunique()
```

```
Out[168]: company_name      942
city                327
state                54
country              61
Industry            120
LeadInvestors       356
dtype: int64
```

Necesitamos convertir los paises a regiones y para eso debemos reemplazar los nombres de algunos paises

```
In [169.. df["country"] = df["country"].replace('AUS', 'Australia')
df["country"] = df["country"].replace('Aus', 'Australia')
df["country"] = df["country"].replace('CAN', 'Canada')
df["country"] = df["country"].replace('GEN', 'Germany')
df["country"] = df["country"].replace('Ger', 'Germany')
df["country"] = df["country"].replace('IE', 'Ireland')
df["country"] = df["country"].replace('Ind', 'India')
df["country"] = df["country"].replace('Netherlands', 'Netherlands')
df["country"] = df["country"].replace('NO', 'Norway')
df["country"] = df["country"].replace('NOR', 'Norway')
df["country"] = df["country"].replace('POL', 'Polonia')
df["country"] = df["country"].replace('SGP', 'Singapore')
df["country"] = df["country"].replace('SWE', 'Sweden')
df["country"] = df["country"].replace('USA', 'United States')
df["country"] = df["country"].replace('United State', 'United States')
```

Convertir los paises en regiones

```
In [170.. def segment_country(country):
    if country in ["China", "Hong Kong", "India", "Indonesia", "Israel", "Japan", "Kuwait", "Pakistan", "Singap
        return 'Asia'
    elif country in ["Austria", "Belgium", "Cyprus", "Estonia", "Finland", "France", "Germany", "Ireland", "Ire
        return 'Europe'
    elif country in ["Canada", "Mexico", "Panama", "United States"]:
        return 'North America'
    elif country in ["Brazil", "Colombia", "Ecuador"]:
        return 'South America'
    elif country in ["Egypt", "Kenya", "Namibia", "Seychelles", "South Africa"]:
        return 'Africa'
    elif country in ["Australia", "New Zealand"]:
        return 'Oceania'
    else:
        return 'Others'
```

Aplicar la función de segmentación para crear la nueva columna

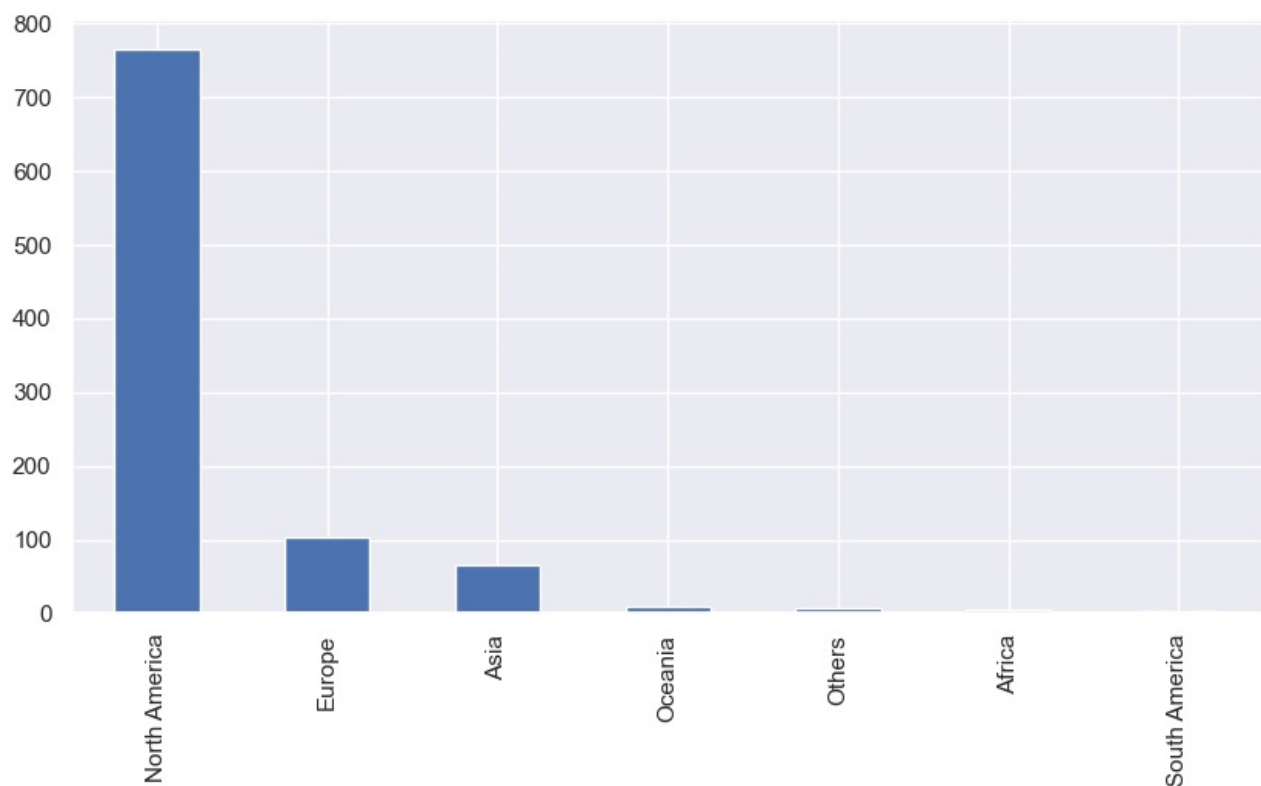
```
In [171.. df['region'] = df['country'].apply(segment_country)
```

```
In [201.. # Agregar una columna de años para hacer graficos de barras por fechas
df['año'] = range(1800, 1800 + len(df))
```

Graficar Regiones

```
In [172.. plt.figure(figsize=(10,5))
df['region'].value_counts().plot(kind='bar')
```

```
Out[172]: <Axes: >
```



EDA EXPLORATORY DATA ANALYSIS

```
In [173.. #Obtener los nombres de todas las columnas con el tipo de dato objetos (catagorical num)
cat_vars = df.select_dtypes(include='object').columns.tolist()

# Crear los espacios para las gráficas
num_cols = len(cat_vars)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5*num_rows))
axs = axs.flatten()

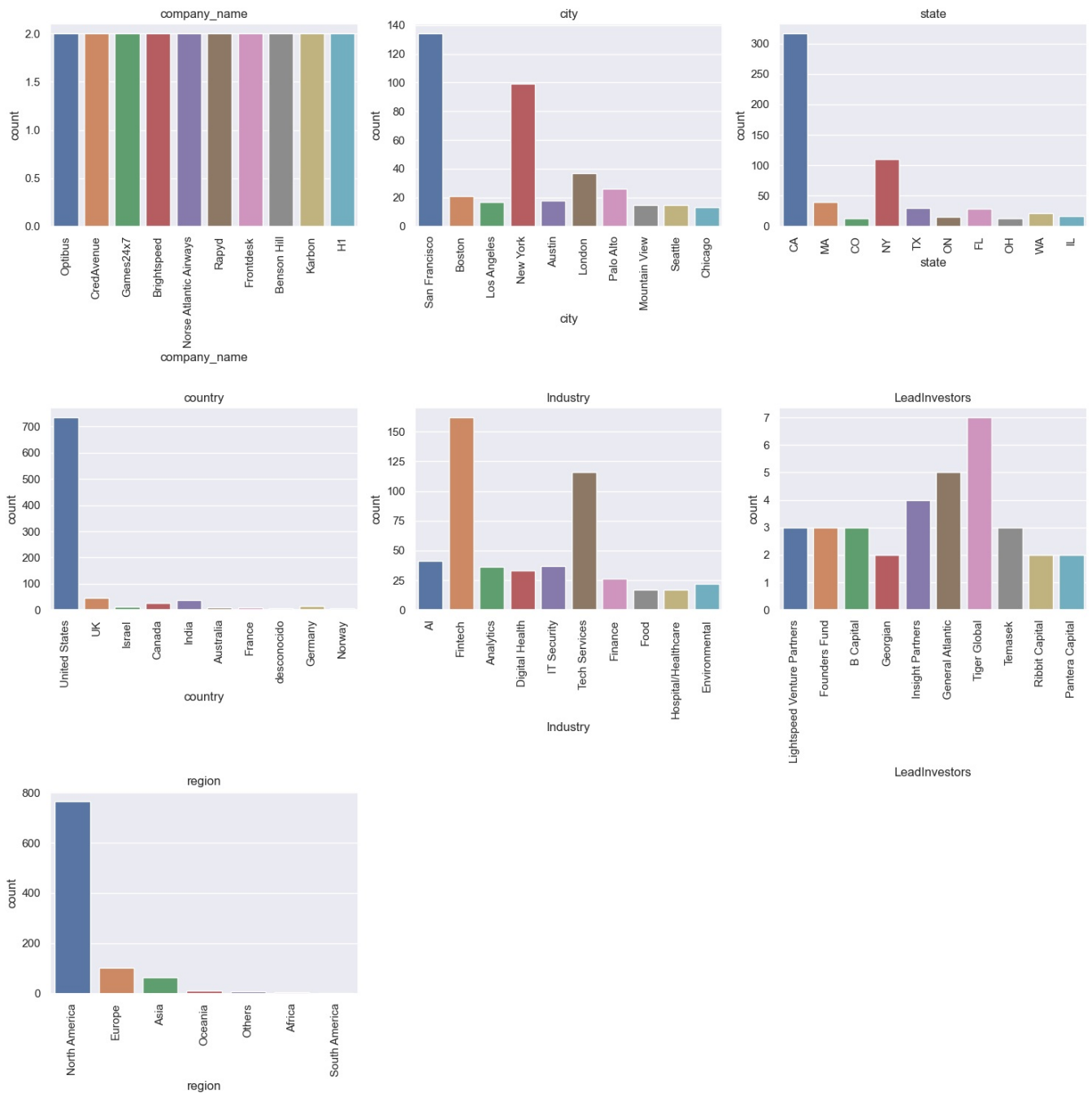
# Crear un contador de gráficas para los 5 primeros valores de cada variable categórica usando seaborn
for i, var in enumerate(cat_vars):
    # Excluir 'desconocido' de las columnas leadinvestors y state
    if var in ['LeadInvestors', 'state']:
        top_values = df[df[var] != 'desconocido'][var].value_counts().nlargest(10).index
        filtered_df = df[df[var].isin(top_values)]
    else:
        top_values = df[var].value_counts().nlargest(10).index
        filtered_df = df[df[var].isin(top_values)]

    sns.countplot(x=var, data=filtered_df, ax=axs[i])
    axs[i].set_title(var)
    axs[i].tick_params(axis='x', rotation=90)

# Eliminar cada espacio extra en los gráficos
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Ajustar los espacios entre las gráficas
fig.tight_layout()

plt.show()
#conteo_valores = df['LeadInvestors'].value_counts()
```



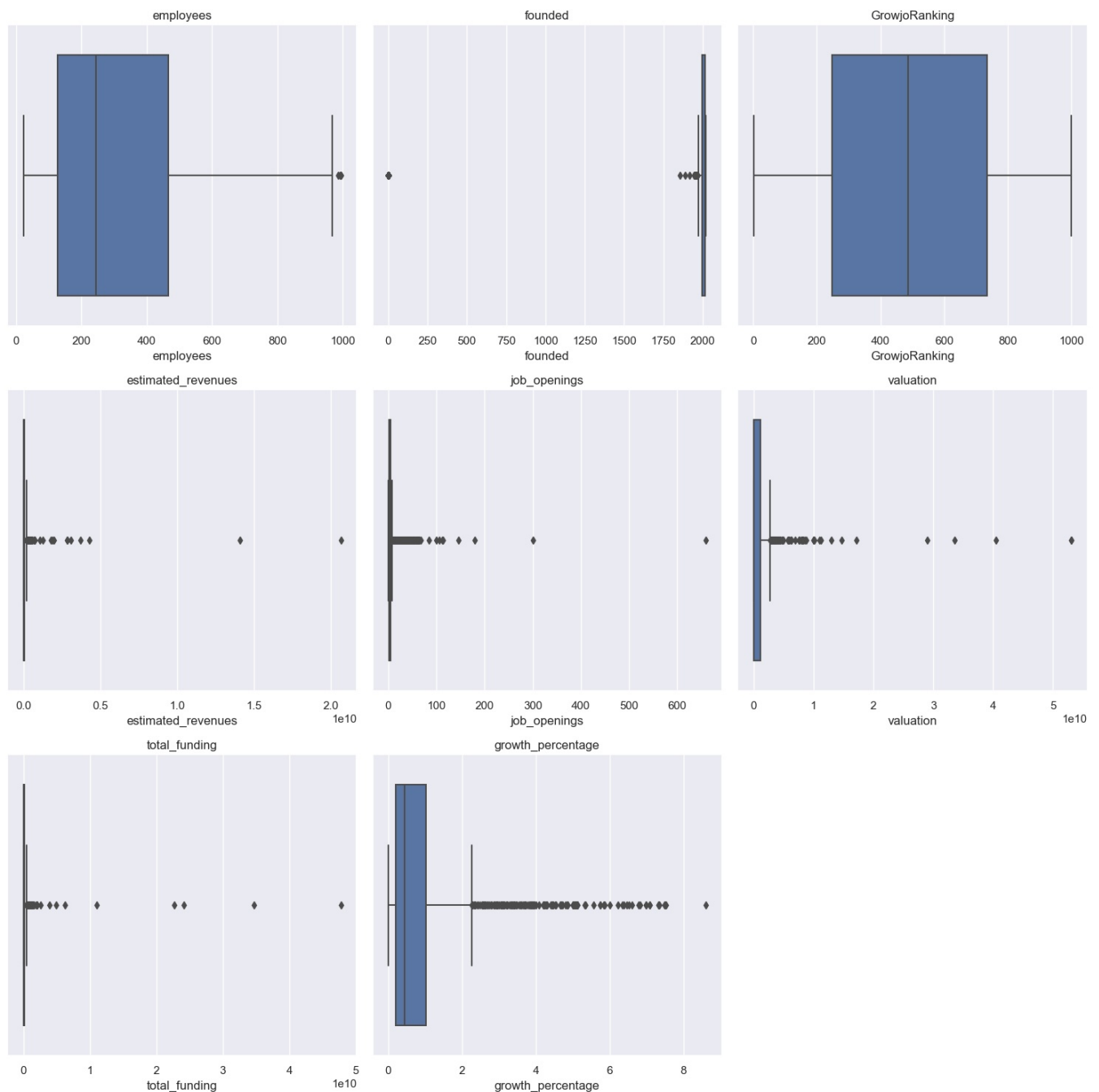
```
In [174]: #Escoger los nombres de las columnas con los datos tipo 'int' o 'float'
num_vars = df.select_dtypes(include=['int', 'float']).columns.tolist()

#Crear un espacio para cada grafica
num_cols = len(num_vars)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5*num_rows))
axs = axs.flatten()

#Crear un boxplot para cada variable n merica usando Seaborn
for i, var in enumerate(num_vars):
    sns.boxplot(x=df[var], ax=axs[i])
    axs[i].set_title(var)

#Eliminar los espacios extras que no se graficaron
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

#Ajustar los espacios entre las graficas
fig.tight_layout()
plt.show()
```



```
In [175.. # Escogemos los nombre de todas las columnas con los datos 'int' "Numeros enteros"
int_vars = df.select_dtypes(include=['int', 'float']).columns.tolist()

#Crear las figuras/espacios para los graficos
num_cols = len(int_vars)
num_rows = (num_cols + 2)// 3 # Asegurate que son los espacios suficientes para todas las graficas
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(20, 5*num_rows))
axs = axs.flatten()

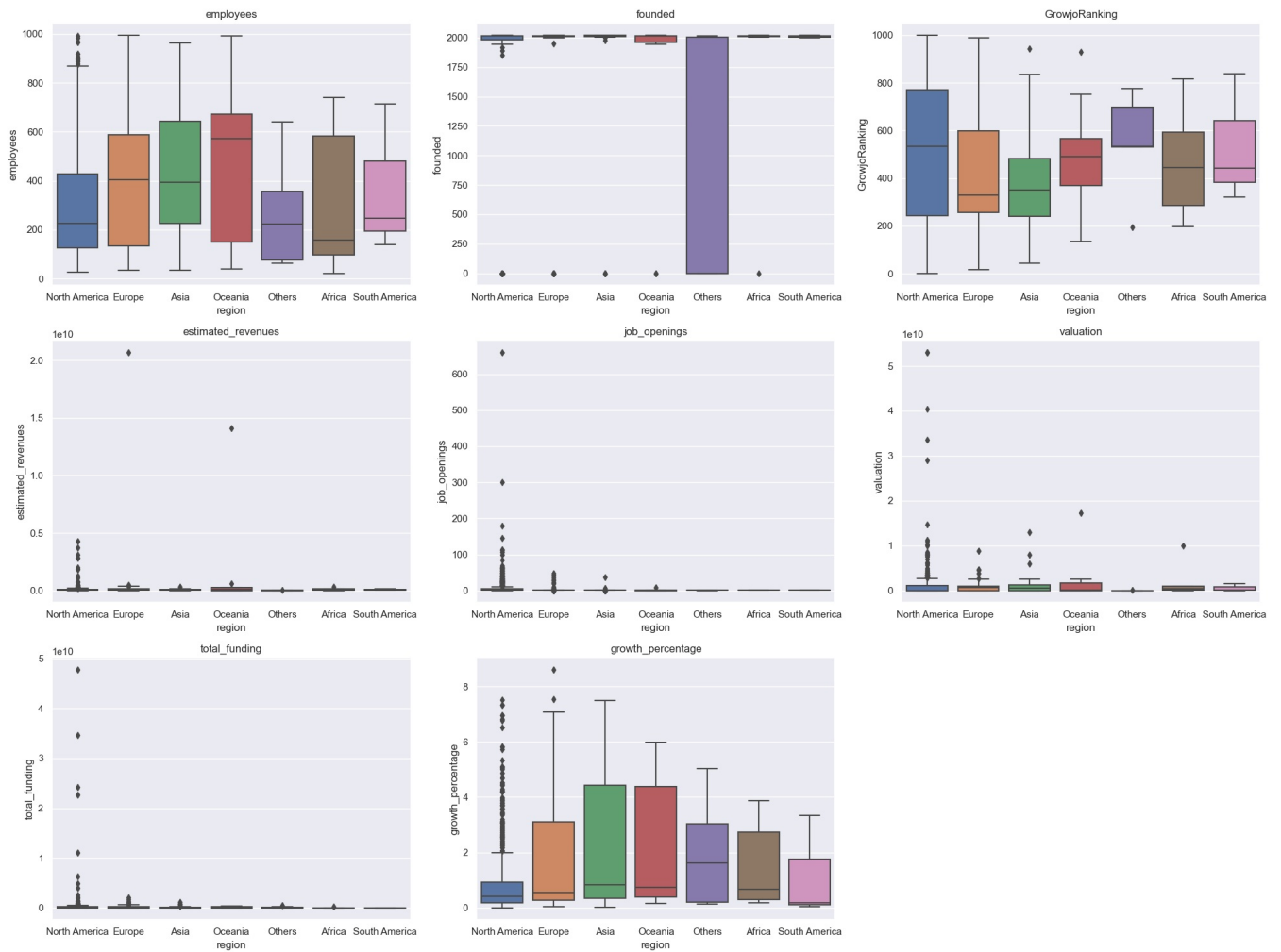
#crear un box-plot para cada variables usando seaborn con hue='attritio'

for i, var in enumerate(int_vars):
    sns.boxplot(y=var, x='region', data=df, ax=axs[i])
    axs[i].set_title(var)

# Eliminar cada espacio extra que o hayan llenado los graficos
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Ajustar los espacios de os graficos y los titulos
fig.tight_layout()

plt.show()
```



```
In [176.. #Escoger los nombres de todas las columnas que contengan datos 'int' y 'float'
int_vars = df.select_dtypes(include=['int','float']).columns.tolist()

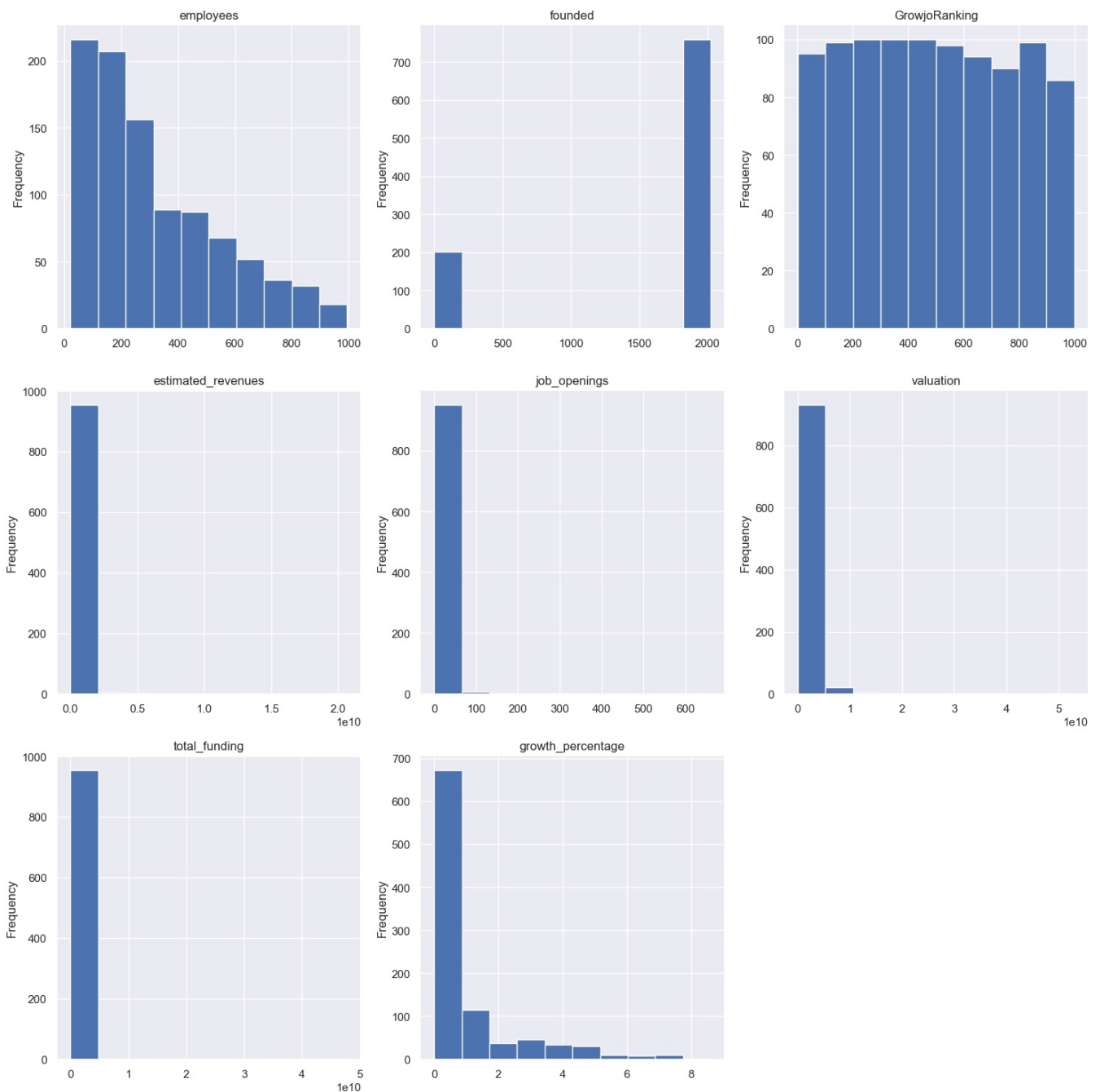
#Crear los espacios para las graficas
num_cols = len(int_vars)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5*num_rows))
axs = axs.flatten()

# Crear un histograma por cada variable entero}
for i, var in enumerate(int_vars):
    df[var].plot.hist(ax=axs[i])
    axs[i].set_title(var)

# Elimnar los espacios extras y dejar solo los que necesitamos
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Ajustar los espacios entre las graficas
fig.tight_layout()

plt.show()
```

```
In [177.. #Obtener los nombres de todas las columnas de tipo 'int' (Entero)

int_vars = df.select_dtypes(include=['int', 'float']).columns.tolist()

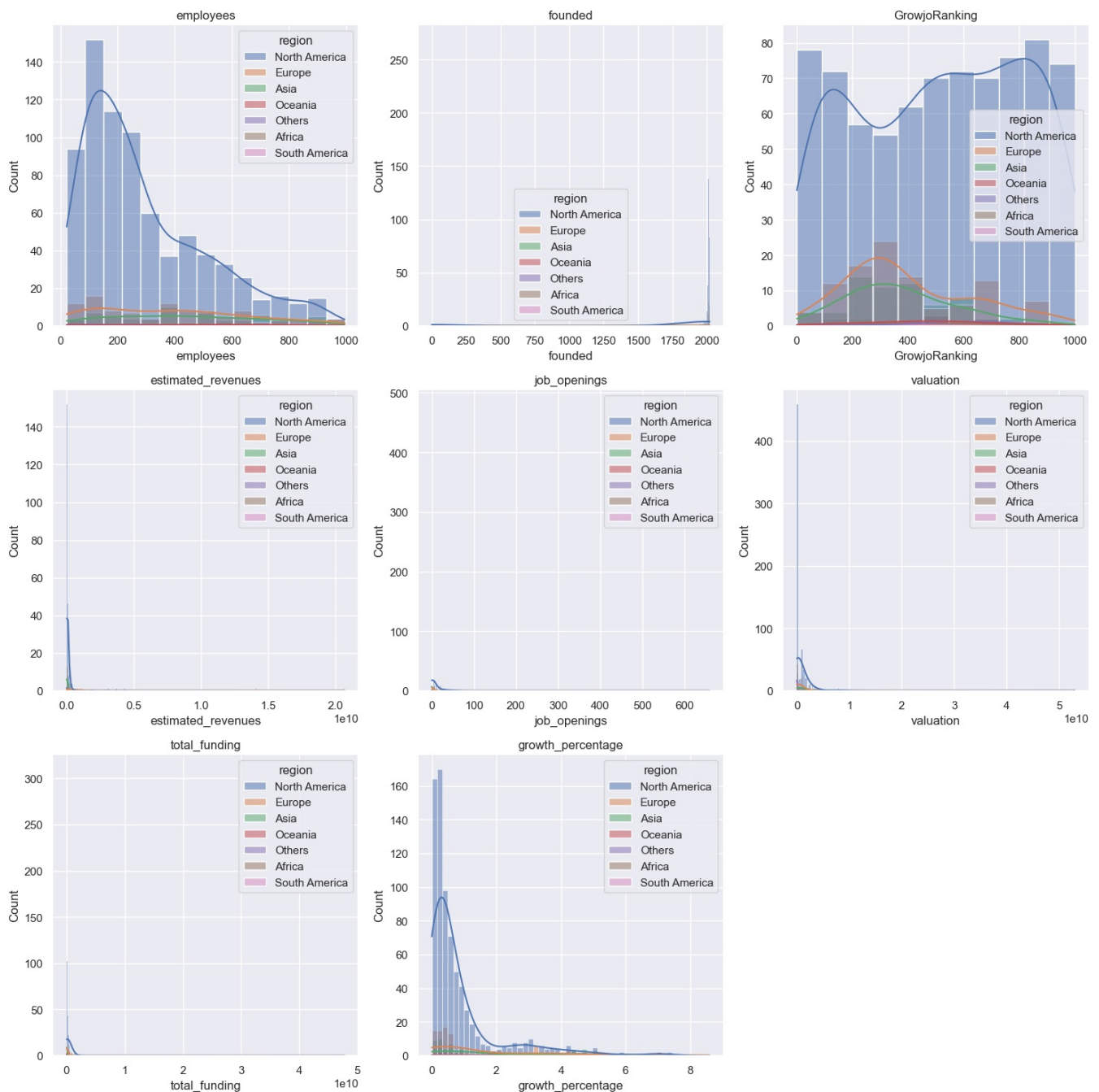
#Crear una figura con los espacios de las graficas
num_cols = len(int_vars)
num_rows = (num_cols + 2) // 3 # To make sure there are enough rows for the subplots
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(15, 5*num_rows))
axs = axs.flatten()

#Crear un histograma para cada variable con hue='Attrition'
for i, var in enumerate(int_vars):
    sns.histplot(data=df, x=var, hue='region', kde=True, ax=axs[i])
    axs[i].set_title(var)

# Eliminar los espacios de graficos que no se necesitan
if num_cols < len(axs):
    for i in range(num_cols, len(axs)):
        fig.delaxes(axs[i])

# Ajustar los espacios entre las graficas
fig.tight_layout()

plt.show()
```



```
In [178.. #Especificar el número máximo de de categorías a mostrar individualmente
max_categories = 7

# Filtrar las columnas categoricas con tip 'objeto'
cat_cols = [col for col in df.columns if df[col].dtype == 'object']

# Crear los espacios de los graficos
num_cols = len(cat_cols)
num_rows = (num_cols + 2) // 3
fig, axs = plt.subplots(nrows=num_rows, ncols=3, figsize=(30, 7*num_rows))

# Aplana la matriz axs para facilitar la indexación
axs = axs.flatten()

# Crear la torta para cada columna categorica
for i, col in enumerate(cat_cols):
    if i < len(axs): # Ensure we don't exceed the number of subplots
        #Count the number of occurrences for each category
        cat_counts = df[col].value_counts()

        # Categorías de grupo más allá de max_categories superiores como 'Otros'
        if len(cat_counts) > max_categories:
            cat_counts_top = cat_counts[:max_categories]
            cat_counts_other = pd.Series(cat_counts[max_categories:].sum(), index=["Other"])
            cat_counts = cat_counts_top.append(cat_counts_other)

        # Crear una torta
        axs[i].pie(cat_counts, labels=cat_counts.index, autopct='%1.1f%%', startangle=90)
        axs[i].set_title(f'{col} region')

#eliminar cada espacio extra
if num_cols < len(axs):
```

```

for i in range(num_cols, len(axes)):
    fig.delaxes(axes[i])

# Ajusta el espacio entre las graficas
fig.tight_layout()

plt.show()

```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

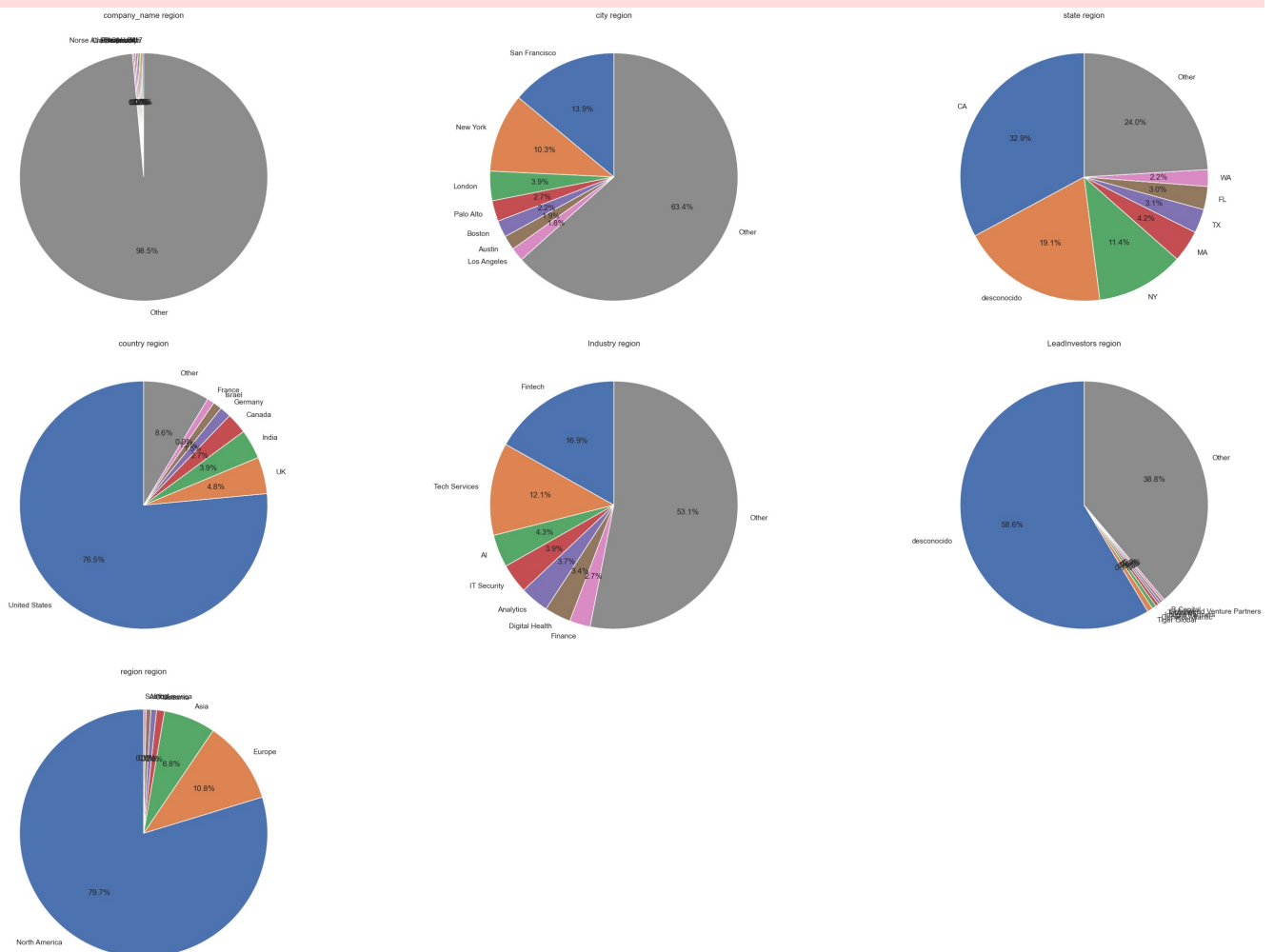
The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\13778241.py:25: FutureWarning:

The series.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.



```

In [179]: df[{'founded', 'Industry', 'company_name'}]
graph=px.line(df, x='founded', y='Industry', color='founded', title='Industry', range_x=[1854, 2022])
graph.show()

df[{'founded', 'region', 'company_name'}]
graph=px.line(df, x='founded', y='region', color='founded', title='region', range_x=[1854, 2022])

```

```
graph.show()
```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\2205364330.py:1: FutureWarning:

Passing a set as an indexer is deprecated and will raise in a future version. Use a list instead.

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\2205364330.py:5: FutureWarning:

Passing a set as an indexer is deprecated and will raise in a future version. Use a list instead.

Grafico de torta por industria

In [190...

```
df_torta = df[{'region','estimated_revenues'}]
```

```
fig=px.pie(df_torta,values='estimated_revenues',color='region',names='region',labels='region',width=800,height=fig.show()
```

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13752\2421618612.py:1: FutureWarning:

Passing a set as an indexer is deprecated and will raise in a future version. Use a list instead.

In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js