

main

December 27, 2023

1 Analsis de Datos COVID-19 en Guatemala 2020

1.1 Código Python

1.1.1 Carga de datos

```
[152]: import pandas as pd
from sqlalchemy import create_engine
from dotenv import load_dotenv
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
import os

load_dotenv()

class SQLServerConnection:
    def __init__(self):
        self.server = os.getenv("SQL_SERVER_SERVER")
        self.database = os.getenv("SQL_SERVER_DATABASE")
        self.username = os.getenv("SQL_SERVER_USERNAME")
        self.password = os.getenv("SQL_SERVER_PASSWORD")
        self.port = os.getenv("SQL_SERVER_PORT")

    # f"DRIVER={{ODBC Driver 17 for SQL Server}};""
    def get_connection(self):
        try:

            trusted_connection = 'yes' # Utiliza 'yes' para Windows
            # Authentication
            driver = '{SQL Server}'
            # Cadena de conexión para Windows Authentication
            # Especificar el controlador de pyodbc en la cadena de conexión
```

```

        connection_string = f'mssql+pyodbc://{{self.server}}/{{self.database}}?
        ↵trusted_connection={{trusted_connection}}&driver=ODBC+Driver+17+for+SQL+Server'

        # connection_string = f'DRIVER={driver};SERVER={self.server};
        ↵DATABASE={self.database};Trusted_Connection={trusted_connection}'

        engine = create_engine(connection_string)
        return engine
    except Exception as e:
        print(f"Error de conexión: {str(e)}")
        return None

```

Query's and dataframes

```
[153]: # Create connection
sql_server_connection = SQLServerConnection()
connection = sql_server_connection.get_connection()

print("connection: ", connection)

# Consulta SQL para extraer datos
sql_query_municipality = """
SELECT M.code_municipality, M.municipality, M.population,
       D.code_department, D.name_department,
       C.code_country, C.name_country
FROM Municipality AS M
JOIN Department AS D ON M.code_department = D.code_department
JOIN Country AS C ON D.code_country = C.code_country;
"""

sql_query_department = 'SELECT * FROM Department'
sql_query_country_cases = 'SELECT * FROM Cases'
sql_query_mdp = """
SELECT
    M.municipality,
    MDR.code_municipality,
    MDR.date_reported,
    MDR.new_deaths,
    SUM(MDR.new_deaths) OVER (PARTITION BY MDR.code_municipality ORDER BY MDR.
    ↵date_reported) AS cumulative_deaths
FROM
    MunicipalityDeathsReported MDR
JOIN
    Municipality M
    ON M.code_municipality = MDR.code_municipality
ORDER BY
    MDR.code_municipality, MDR.date_reported;
"""

sql_query_mdp_gt_0 = """
```

```

SELECT
    M.municipality,
    M.population,
    D.name_department,
    MDR.code_municipality,
    MDR.date_reported,
    MDR.new_deaths,
    SUM(MDR.new_deaths) OVER (PARTITION BY MDR.code_municipality ORDER BY MDR.
    date_reported) AS cumulative_deaths
FROM
    MunicipalityDeathsReported MDR
JOIN
    Municipality M
        ON M.code_municipality = MDR.code_municipality
JOIN
    Department D
        ON M.code_department = D.code_department
WHERE
    MDR.new_deaths > 0
ORDER BY
    MDR.code_municipality, MDR.date_reported;
"""

```

```

# Ejecutar la consulta y cargar los resultados en DataFrames
df_municipality = pd.read_sql(sql_query_municipality, connection)
df_department = pd.read_sql(sql_query_department, connection)
df_country_cases = pd.read_sql(sql_query_country_cases, connection)
df_mdp = pd.read_sql(sql_query_mdp, connection)
df_mdp_gt_0 = pd.read_sql(sql_query_mdp_gt_0, connection)

df_municipality = df_municipality.drop_duplicates()
df_department = df_department.drop_duplicates()
df_country_cases = df_country_cases.drop_duplicates()
df_mdp = df_mdp.drop_duplicates()
df_mdp_gt_0 = df_mdp_gt_0.drop_duplicates()

```

```

connection: Engine(mssql+pyodbc://DESKTOP-4JJFHV8\SQLEXPRESS/ss2?driver=ODBC+Driver+17+for+SQL+Server&trusted_connection=yes)

```

[154]: df_mdp_gt_0.head()

	municipality	population	name_department	code_municipality	date_reported
0	MAZATENANGO	83448	SUCHITEPEQUEZ	1001	2020-05-16
1	MAZATENANGO	83448	SUCHITEPEQUEZ	1001	2020-05-19
2	MAZATENANGO	83448	SUCHITEPEQUEZ	1001	2020-06-28
3	MAZATENANGO	83448	SUCHITEPEQUEZ	1001	2020-07-02
4	MAZATENANGO	83448	SUCHITEPEQUEZ	1001	2020-07-03

```

new_deaths cumulative_deaths
0           1             1
1           1             2
2           1             3
3           1             4
4           1             5

```

[155]: df_municipality.head()

```

[155]:   code_municipality municipality population code_department name_department \
0          1001    MAZATENANGO     83448          10    SUCHITEPEQUEZ
1          1002    CUYOTENANGO    37283          10    SUCHITEPEQUEZ
2          1008      SAMAYAC      26350          10    SUCHITEPEQUEZ
3          101     GUATEMALA     1205668          1    GUATEMALA
4          1013    CHICACAO      67994          10    SUCHITEPEQUEZ

code_country name_country
0            GT      Guatemala
1            GT      Guatemala
2            GT      Guatemala
3            GT      Guatemala
4            GT      Guatemala

```

1.1.2 Funciones para análisis de datos

Funciones histogramas log, de caja, histogramas estándares y análisis estadísticos

```

[156]: def plot_analysis(df, variable_name):
    # Filtra los valores mayores que cero
    filtered_data = df[variable_name][df[variable_name] > 0]

    # Configurar el estilo de seaborn
    sns.set(style="whitegrid")

    # Crear una figura con subgráficos
    fig, axes = plt.subplots(1, 4, figsize=(20, 5))

    # Histograma con la transformación logarítmica
    sns.histplot(filtered_data, bins=20, kde=True, edgecolor='black', log_scale=True, ax=axes[0])
    axes[0].set_xlabel(f'Logaritmo de {variable_name}')
    axes[0].set_ylabel('Frecuencia')
    axes[0].set_title(f'Histograma logarítmico de {variable_name}')

    # Diagrama de caja
    sns.boxplot(y=filtered_data, ax=axes[1])
    axes[1].set_ylabel(variable_name)

```

```

axes[1].set_title(f'Diagrama de caja de {variable_name}')
```

Histograma estándar

```

sns.histplot(filtered_data, bins=20, kde=True, edgecolor='black',  

             ax=axes[2])
axes[2].set_xlabel(variable_name)
axes[2].set_ylabel('Frecuencia')
axes[2].set_title(f'Histograma estándar de {variable_name}')


# Estadísticas descriptivas
statistics = filtered_data.describe()
statistics_translated = statistics.rename(index={
    'count': 'Conteo',
    'mean': 'Promedio',
    'std': 'Desviación Estándar',
    'min': 'Mínimo',
    '25%': 'Cuartil 25%',
    '50%': 'Mediana',
    '75%': 'Cuartil 75%',
    'max': 'Máximo'
})

```

Añadir estadísticas traducidas a la tabla

```

axes[3].axis('off')
axes[3].table(cellText=statistics_translated.reset_index().values,
              colLabels=['Estadísticas', 'Valor'],
              cellLoc='center', loc='center', colColours=['#f2f2f2']*2)
axes[3].set_title(f'Estadísticas descriptivas de {variable_name}')


# Ajustar el diseño
plt.tight_layout()
plt.show()
```

def descr(df, variable_name):

Filtra los valores mayores que cero

```

filtered_data = df[variable_name][df[variable_name] > 0]

# Estadísticas descriptivas
statistics = filtered_data.describe()
statistics_translated = statistics.rename(index={
    'count': 'Conteo',
    'mean': 'Promedio',
    'std': 'Desviación Estándar',
    'min': 'Mínimo',
    '25%': 'Cuartil 25%',
    '50%': 'Mediana',
    '75%': 'Cuartil 75%',
```

```

        'max': 'Máximo'
    })
print(f'Estadísticas monovariables de {variable_name}')
print(statistics_translated, '\n')

```

Función datos multivariable cuantitativo: diagramas de dispersión

[157]: `def scatter_plot(data, x_column, y_column, label_column, title, size=False):`
 `"""`

Genera un gráfico de dispersión.

Parámetros:

- *data: DataFrame que contiene los datos.*
- *x_column: Nombre de la columna para el eje x.*
- *y_column: Nombre de la columna para el eje y.*
- *label_column: Nombre de la columna para etiquetas de puntos.*
- *title: Título del gráfico.*
- *size: Nombre de la columna para el tamaño de los puntos (opcional).*

Ejemplo de uso:

```

scatter_plot(df_mdp_gt_0, 'new_deaths', 'cumulative_deaths',  

↳ 'municipality', 'Dispersión Nuevas Muertes vs Muertes Acumuladas', size=True)
"""
# Configurar el estilo de seaborn
sns.set(style="whitegrid")

# Configurar el tamaño del gráfico
plt.figure(figsize=(12, 8))

# Generar el gráfico de dispersión
if size:
    sns.scatterplot(x=x_column, y=y_column, hue=label_column, size=size,  

↳ sizes=(20, 200), data=data)
else:
    sns.scatterplot(x=x_column, y=y_column, hue=label_column, data=data)

# Añadir título y etiquetas de ejes
plt.title(title)
plt.xlabel(x_column)
plt.ylabel(y_column)

# Mostrar la leyenda
plt.legend()

# Mostrar el gráfico
plt.show()

```

```

def scatter_plot_multi(data, x_column, y_column, label_column, title,
                     size=False, points_per_subplot=10, subplots_per_row=3):
    """
    Genera un gráfico de dispersión dividido en múltiples subgráficas.

    Parámetros:
    - data: DataFrame que contiene los datos.
    - x_column: Nombre de la columna para el eje x.
    - y_column: Nombre de la columna para el eje y.
    - label_column: Nombre de la columna para etiquetas de puntos.
    - title: Título del gráfico.
    - size: Nombre de la columna para el tamaño de los puntos (opcional).
    - points_per_subplot: Cantidad de elementos por subgráfico.
    - subplots_per_row: Cantidad de subgráficas por fila.

    Ejemplo de uso:
    scatter_plot_multi(df_mdp_gt_0, 'new_deaths', 'cumulative_deaths',
                     'name_department', 'Dispersión Nuevas Muertes vs Muertes Acumuladas',
                     size=True)
    """

    # Configurar el estilo de seaborn
    sns.set(style="whitegrid")

    # Dividir los datos en grupos de 10 departamentos
    grouped_data = [group for _, group in data.groupby(label_column,
                                                       sort=False)]
    grouped_data = [grouped_data[i:i + points_per_subplot] for i in range(0,
                                                                       len(grouped_data), points_per_subplot)]

    # Calcular la cantidad total de subgráficas
    total_subplots = len(grouped_data)

    # Calcular la cantidad total de filas necesarias
    total_rows = total_subplots // subplots_per_row + (1 if total_subplots % subplots_per_row else 0)

    # Crear subgráficas
    fig, axes = plt.subplots(total_rows, subplots_per_row, figsize=(20, 5 * total_rows))

    # Iterar sobre las subgráficas
    for i, group in enumerate(grouped_data):
        # Calcular la posición de la subgráfica
        row = i // subplots_per_row
        col = i % subplots_per_row

```

```

# Configurar el tamaño del gráfico
plt.sca(axes[row, col])

# Concatenar los grupos y generar el gráfico de dispersión
group_concatenated = pd.concat(group)
if size:
    sns.scatterplot(x=x_column, y=y_column, hue=label_column, □
    ↪size=size, sizes=(20, 200), data=group_concatenated)
else:
    sns.scatterplot(x=x_column, y=y_column, hue=label_column, □
    ↪palette='viridis', data=group_concatenated)

# Añadir título y etiquetas de ejes
plt.title(f'{title} - Parte {i + 1}')
plt.xlabel(x_column)
plt.ylabel(y_column)

# Mostrar la leyenda
plt.legend()

# Ajustar el diseño
plt.tight_layout()
plt.show()

```

Funcion datos cualitativos: histogramas count y sum de columnas según un dataset

[158]: `def generate_count_plot(data, x_column, x_label, y_label, title, size=False):`
 `"""`

Genera un diagrama de barras para el conteo de registros.

Parameters:

- *data: DataFrame que contiene los datos.*
- *x_column: Nombre de la columna en el DataFrame para el eje x.*
- *x_label: Etiqueta del eje x.*
- *y_label: Etiqueta del eje y.*
- *title: Título del diagrama.*
- *size: Booleano que indica si se usará un tamaño personalizado para la figura.*

Returns:

None (muestra el diagrama y una tabla).

`"""`

`# Configurar el estilo de seaborn`

`sns.set(style="whitegrid")`

`# Crear un diagrama de barras`

`figsize = (50, 6) if size else (15, 6)`

```

plt.figure(figsize=figsize)
sns.countplot(data=data, x=x_column, order=data[x_column].value_counts().index)
plt.xlabel(x_label)
plt.ylabel(y_label)
plt.title(title)
plt.xticks(rotation=45, ha='right')

# Mostrar la cantidad total de datos
total_data = len(data)
# print(f"Cantidad total de datos de {title}: {total_data}")

# Mostrar una tabla con los resultados del histograma
histogram_table = pd.DataFrame(data[x_column].value_counts()).reset_index()
histogram_table.columns = [x_column, 'Count']
# print(histogram_table)

plt.show()

```

```

def generate_sum_plot(data, x_column, y_column, x_label, y_label, title,
                      size=False):
    """
    Genera un diagrama de barras para el conteo de registros.

    Parameters:
    - data: DataFrame que contiene los datos.
    - x_column: Nombre de la columna en el DataFrame para el eje x.
    - x_label: Etiqueta del eje x.
    - y_label: Etiqueta del eje y.
    - title: Título del diagrama.
    - size: Booleano que indica si se usará un tamaño personalizado para la
    figura.
    - is_count: Booleano que indica si se realizará un conteo (True) o una suma
    (False).
    
```

Returns:
None (muestra el diagrama y una tabla).

"""

Configurar el estilo de seaborn

sns.set(style="whitegrid")

Crear un diagrama de barras

figsize = (50, 6) if size else (15, 6)

plt.figure(figsize=figsize)

Filtrar datos

```

data_filtered = data[data[x_column] != 0]

# Realizar una sumatoria de todos los valores en el DataFrame filtrado
df_count = data_filtered.groupby(y_column).agg({x_column: 'sum'}).
reset_index()
df_count.columns = [x_column, 'Count']

# Filtrar valores con al menos min_occurrences ocurrencias
min_occurrences = 1 # Puedes ajustar este valor según tus necesidades
values_to_plot = df_count[df_count['Count'] >= min_occurrences][x_column]

# Usar values_to_plot para imprimir el histograma
# print(values_to_plot)

# Graficar el histograma
sns.barplot(data=df_count, x=x_column, y='Count')

plt.xlabel(x_label)
plt.ylabel(y_label)
plt.title(title)
plt.xticks(rotation=45, ha='right')

# Ordenar df_count por la columna 'Count' en orden descendente
df_count_sorted = df_count.sort_values(by='Count', ascending=False)

# Imprimir la sumatoria de Count de "population"
total_population = df_count_sorted['Count'].sum()
# print(f"Sumatoria de Count de population en {title}: {total_population}")

# Imprimir el top 5 de df_count
# print("Top 5:")
# print(df_count_sorted.head())

# Imprimir los peores 5 de df_count
# print("Peores 5:")
# print(df_count_sorted.tail())
plt.show()

```

Funcion para generar diagramas multivariados: diagramas de Barra y mapas de calor

```
[159]: def generate_bar(data, x_column, y_column, x_label, y_label, title,
                     rotation=45, size=False):
    """
    Genera un diagrama de barras y un mapa de calor para variables
    multivariadas.
    """

```

Parameters:

```

- data: DataFrame que contiene los datos.
- x_column: Nombre de la columna en el DataFrame para el eje x.
- y_column: Nombre de la columna en el DataFrame para el eje y.
- values_column: Nombre de la columna con los valores a mostrar en el mapa de calor.
- x_label: Etiqueta del eje x.
- y_label: Etiqueta del eje y.
- title: Título de los gráficos.
- rotation: Ángulo de rotación de las etiquetas en el eje x (por defecto, 0).
- size: Tamaño de los gráficos (por defecto, False).

```

Returns:

None (muestra el diagrama de barras y el mapa de calor).

"""

```
sns.set(style="whitegrid")
```

Diagrama de Barras

```
figsize = (15,6) if not size else (25, 6)
plt.figure(figsize=figsize)
sns.barplot(data=data, x=x_column, y=y_column)
plt.xlabel(x_label)
plt.ylabel(y_label)
plt.title(title + " - Diagrama de Barras")
plt.xticks(rotation=rotation) # Eliminamos el parámetro ha='right'
plt.show()
```

```
def generate_multivariable_bar_subplots(data, x_column, y_column, x_label,
y_label, title, rotation=0, size=False, bars_per_subplot=10,
subplots_per_row=3):
    """
```

Genera múltiples subgráficas de barras para variables multivariadas.

Parameters:

```

- data: DataFrame que contiene los datos.
- x_column: Nombre de la columna en el DataFrame para el eje x.
- y_column: Nombre de la columna en el DataFrame para el eje y.
- x_label: Etiqueta del eje x.
- y_label: Etiqueta del eje y.
- title: Título de las subgráficas.
- rotation: Ángulo de rotación de las etiquetas en el eje x (por defecto, 0).
- size: Tamaño de las barras (por defecto, False).
- bars_per_subplot: Cantidad de barras por subgráfica.
- subplots_per_row: Cantidad de subgráficas por fila.

```

Returns:

```

None (muestra las subgráficas de barras).
"""

sns.set(style="whitegrid")

# Dividir los datos en grupos de 10 elementos
grouped_data = [group for _, group in data.groupby(x_column, sort=False)]
grouped_data = [grouped_data[i:i + bars_per_subplot] for i in range(0, u
↪len(grouped_data), bars_per_subplot)]

# Calcular la cantidad total de subgráficas
total_subplots = len(grouped_data)

# Calcular la cantidad total de filas necesarias
total_rows = total_subplots // subplots_per_row + (1 if total_subplots % u
↪subplots_per_row else 0)

# Crear subgráficas
fig, axes = plt.subplots(total_rows, subplots_per_row, figsize=(20, 5 * u
↪total_rows))

# Iterar sobre las subgráficas
for i, group in enumerate(grouped_data):
    # Calcular la posición de la subgráfica
    row = i // subplots_per_row
    col = i % subplots_per_row

    # Configurar el tamaño del gráfico
    plt.sca(axes[row, col])

    # Concatenar los grupos y generar el gráfico de barras
    group_concatenated = pd.concat(group)
    sns.barplot(data=group_concatenated, x=x_column, y=y_column)

    # Establecer el límite superior del eje y
    max_y_value = data[y_column].max()
    plt.ylim(0, max_y_value + 1) # Puedes ajustar el 1 según tus u
↪necesidades

    # Añadir título y etiquetas de ejes
    plt.title(f'{title} - Parte {i + 1}')
    plt.xlabel(x_label)
    plt.ylabel(y_label)

# Ajustar el diseño
plt.tight_layout()
plt.show()

```

```

def generate_heatmaps(df, x_column, y_column, x_label, y_label, title,
                     ↪min_occurrences=10, num_row_per_graph=10, num_plots_per_row=4,
                     ↪is_count=True):
    df_filtered = df[df[x_column] != 0]

    # Realizar el cálculo según el valor de is_count
    if is_count:
        # Contar las ocurrencias de cada valor en la columna y en el DataFrame
        ↪filtrado
        df_count = df_filtered.groupby(y_column).size().
        ↪reset_index(name='count')
    else:
        # Filtrar datos
        # Realizar una sumatoria de todos los valores en el DataFrame filtrado
        df_count = df_filtered.groupby(y_column).agg({x_column: 'sum'}).
        ↪reset_index()
        df_count.columns = [y_column, 'count']

    # Filtrar valores con al menos min_occurrences ocurrencias
    values_to_plot = df_count[df_count['count'] >= min_occurrences][y_column]

    # Dividir los valores en grupos de num_plots_per_row
    values_groups = np.array_split(values_to_plot, len(values_to_plot) // ↪
                                    num_row_per_graph + 1)
    # print(values_groups)

    # Configurar el estilo de seaborn
    sns.set(style="whitegrid")

    # Calcular el número necesario de filas
    num_rows = max(1, len(values_groups) // num_plots_per_row + (1 if ↪
                                                               len(values_groups) % num_plots_per_row else 0))

    # Calcular el valor máximo global para 'count' entre los valores filtrados
    global_max_count = df_count[df_count[y_column].
        ↪isin(values_to_plot)][['count']].max()

    # Crear subgráficas
    fig, axes = plt.subplots(num_rows, num_plots_per_row, figsize=(20, 1 * ↪
                                                               num_row_per_graph))

    for i, values_group in enumerate(values_groups):
        # Filtrar el DataFrame original para el grupo actual
        df_group = df_filtered[df_filtered[y_column].isin(values_group)]

```

```

# Contar las ocurrencias de cada valor en el DataFrame filtrado
df_group_count = df_group.groupby(y_column).agg({x_column: 'sum'}).
    ↪reset_index()
df_group_count.columns = [y_column, 'count']

# Crear un mapa de calor para el grupo actual
ax = axes[i] if len(values_groups) == 1 else axes[i // ↪
    ↪num_plots_per_row, i % num_plots_per_row]
sns.heatmap(df_group_count.pivot_table(index=y_column, values='count', ↪
    ↪aggfunc='sum'),
    cmap=sns.color_palette("Spectral", as_cmap=True), ↪
    ↪annot=True, fmt='g', ax=ax,
    vmin=0, vmax=global_max_count) # Usar global_max_count en ↪
    ↪lugar de df_group_count['count'].max()

# Añadir título
ax.set_title(f'y_label - {i + 1}')

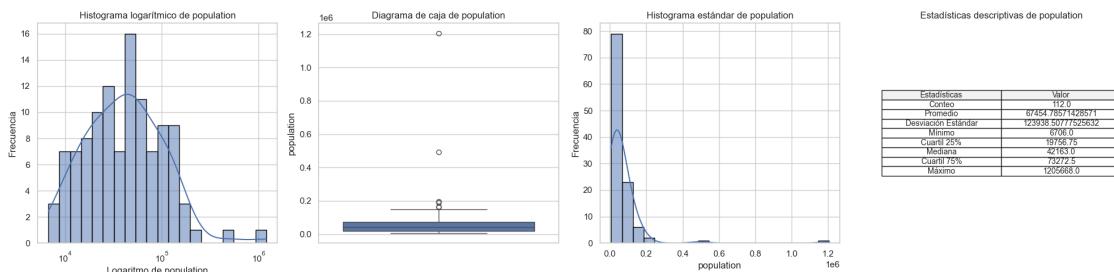
```

1.2 EDA Monovariable

1.2.1 Datos cuantitativos

Graficos monovariables cuantitativos | Análisis de la variable population | Histograma logarítmico de population | Diagrama de caja de population | Histograma estándar de population | | ————— | ————— | ————— || El histograma logarítmico muestra la distribución de las poblaciones de los municipios. La transformación logarítmica se aplicó para mejorar la visualización eliminando ceros y ajustando escalas. | El diagrama de caja presenta la variabilidad y distribución de las poblaciones de los municipios. Se observa una amplia gama de tamaños de población, con algunos municipios que tienen valores atípicos significativamente más altos. | El histograma estándar muestra la distribución de las poblaciones de los municipios sin la transformación logarítmica. La mayoría de los municipios tienen poblaciones moderadas, mientras que algunos tienen tamaños de población excepcionalmente altos. |

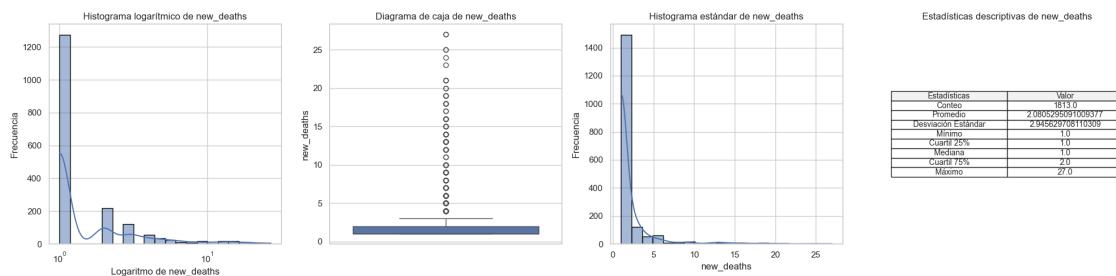
[160]: plot_analysis(df_municipality, 'population')



Análisis de la variable new_deaths

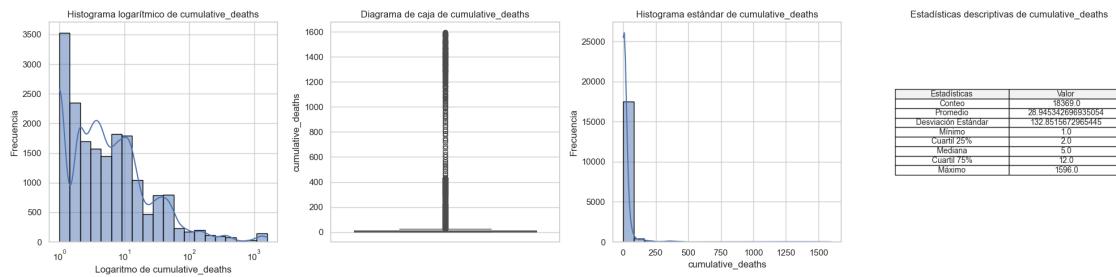
Histograma logarítmico de new_deaths	Diagrama de caja de new_deaths	Histograma estándar de new_deaths
El histograma logarítmico muestra la distribución de nuevos casos de muerte diarios. La mayoría de los días registran una baja cantidad de nuevas muertes, pero hay algunos días con valores atípicos más altos.	El diagrama de caja presenta la variabilidad en el número de nuevas muertes diarias. La mayoría de los días tienen pocos casos, pero hay algunos días con valores atípicos más altos.	El histograma estándar muestra la distribución de nuevos casos de muerte diarios sin la transformación logarítmica. La mayoría de los días tienen pocas nuevas muertes, mientras que algunos días tienen valores excepcionalmente altos.

```
[161]: plot_analysis(df_mdp, 'new_deaths')
```



Análisis de la variable `cumulative_deaths` | Histograma logarítmico de `cumulative_deaths`
| Diagrama de caja de `cumulative_deaths` | Histograma estándar de `cumulative_deaths`
| | ————— | ————— | ————— | | El histograma logarítmico muestra la distribución
de muertes acumulativas en los municipios. La transformación logarítmica se aplicó para mejorar
la visualización. | El diagrama de caja presenta la variabilidad en el número acumulado de muertes
en los municipios. Se observa una amplia gama de valores acumulativos, con algunos municipios
que tienen valores atípicos significativamente más altos. | El histograma estándar muestra la dis-
tribución de muertes acumulativas en los municipios sin la transformación logarítmica. Algunos
municipios tienen números acumulativos significativamente más altos que otros. |

```
[162]: plot_analysis(df_mdp, 'cumulative_deaths')
```



1.2.2 Datos Cualitativos

Uso y calculo de registros por departamento y municipio Cantidad total de datos de Registros de Municipios: 112

Cantidad total de datos de Registros de Departamentos: 17

Municipality

Count

name_department

Count

MAZATENANGO

1

GUATEMALA

1

CUYOTENANGO

1

JALAPA

1

ACATENANGO

1

TOTONICAPAN

1

PATZICIA

1

SOLOLA

1

POCHUTA

1

ESCUINTLA

1

...

...

...

...

JOYABAJ

1

PETEN

1

CUNEN

1

QUICHE

1

PATZITE

1

HUEHUETENANGO

1

CHICHICASTENANGO

1

RETALHULEU

1

GENOVA

1

QUETZALTENANGO

1

TOTAL

112

TOTAL

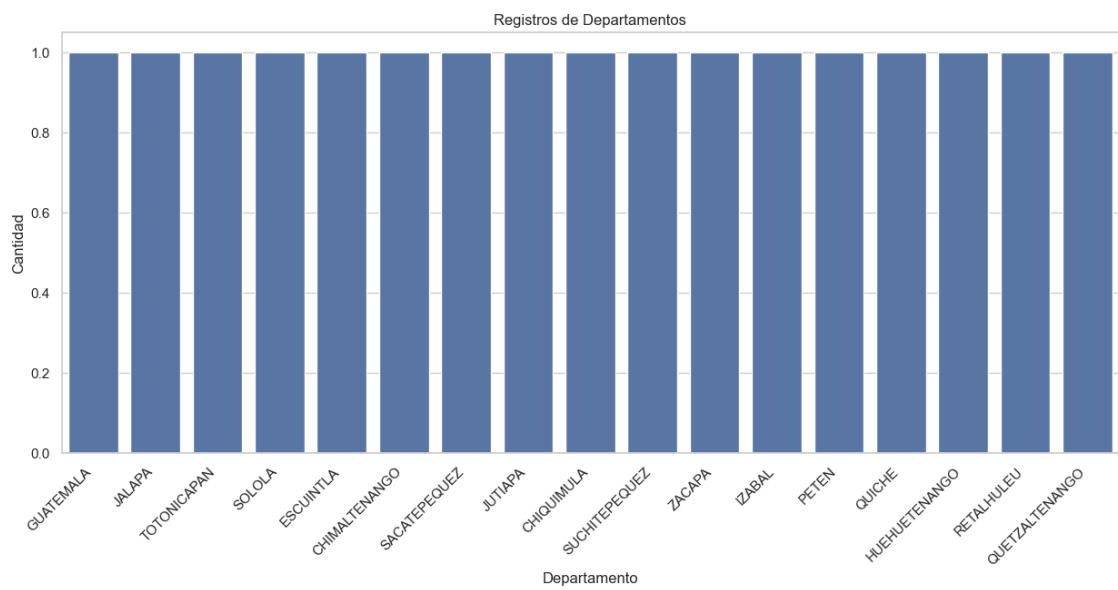
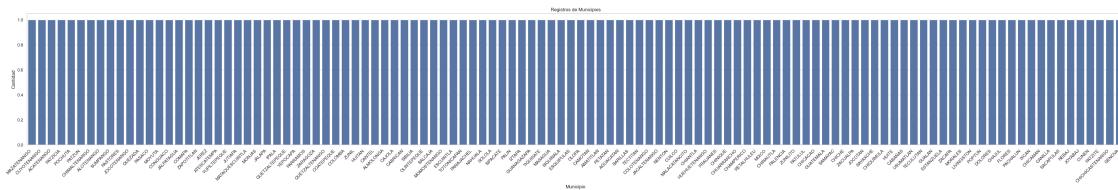
17

Observaciones: - En el análisis de Registros de Municipios se observa un total de 112 municipios distintos. - En el análisis de Registros de Departamentos, se identifican 17 departamentos diferentes.

Ambas gráficas son similares, puesto que solo existe un registro para cada uno de los departamento y cada municipio

```
[163]: # Generar diagrama de barras para municipios: cantidad de veces que aparecen en los casos de covid
generate_count_plot(df_municipality, 'municipality', 'Municipio', 'Cantidad', 'Registros de Municipios', True)

# Generar diagrama de barras para municipios: cantidad de veces que aparecen en los casos de covid
generate_count_plot(df_department, 'name_department', 'Departamento', 'Cantidad', 'Registros de Departamentos')
```

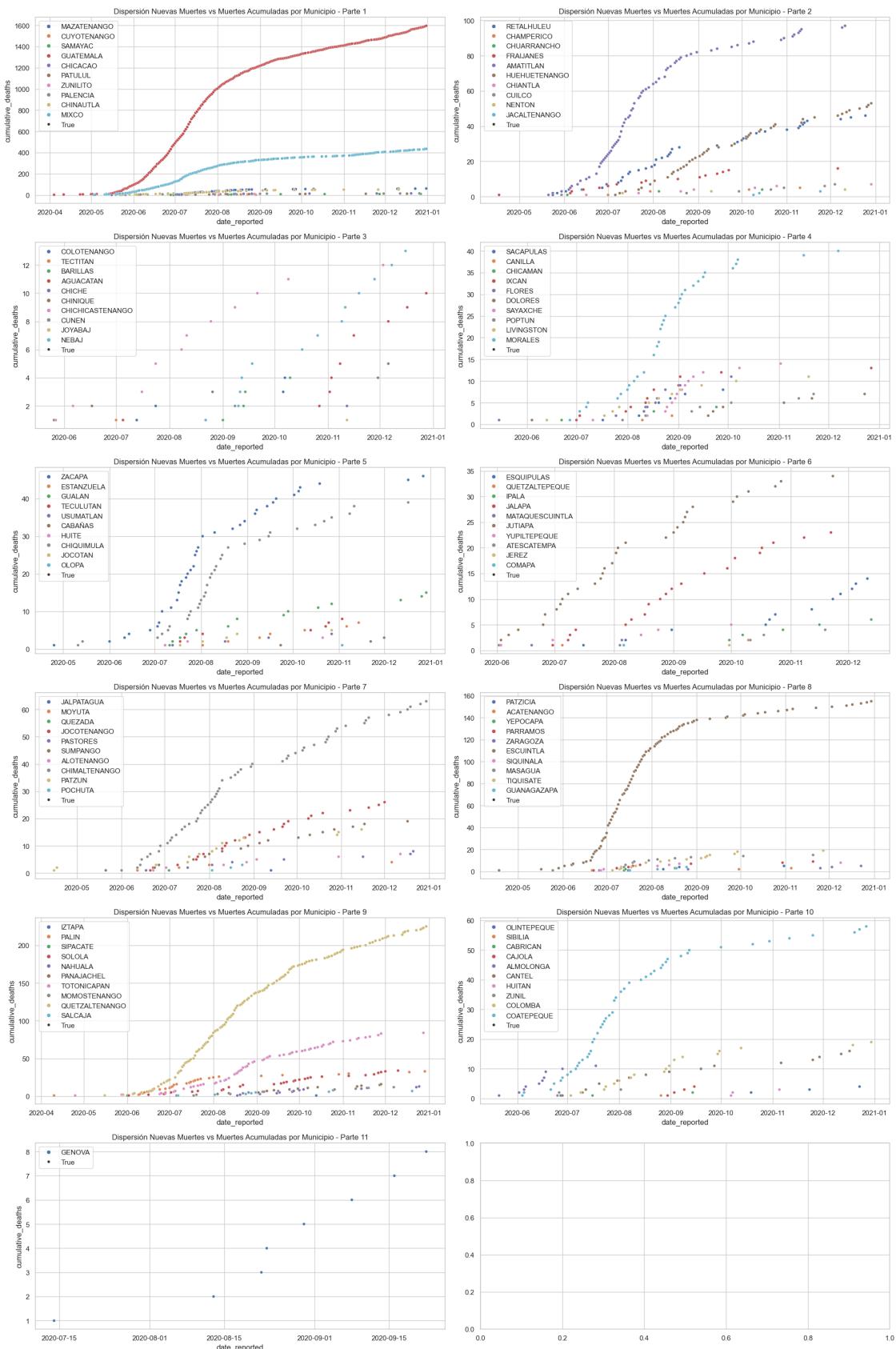


1.3 EDA Multivariable

1.3.1 Datos cuantitativos

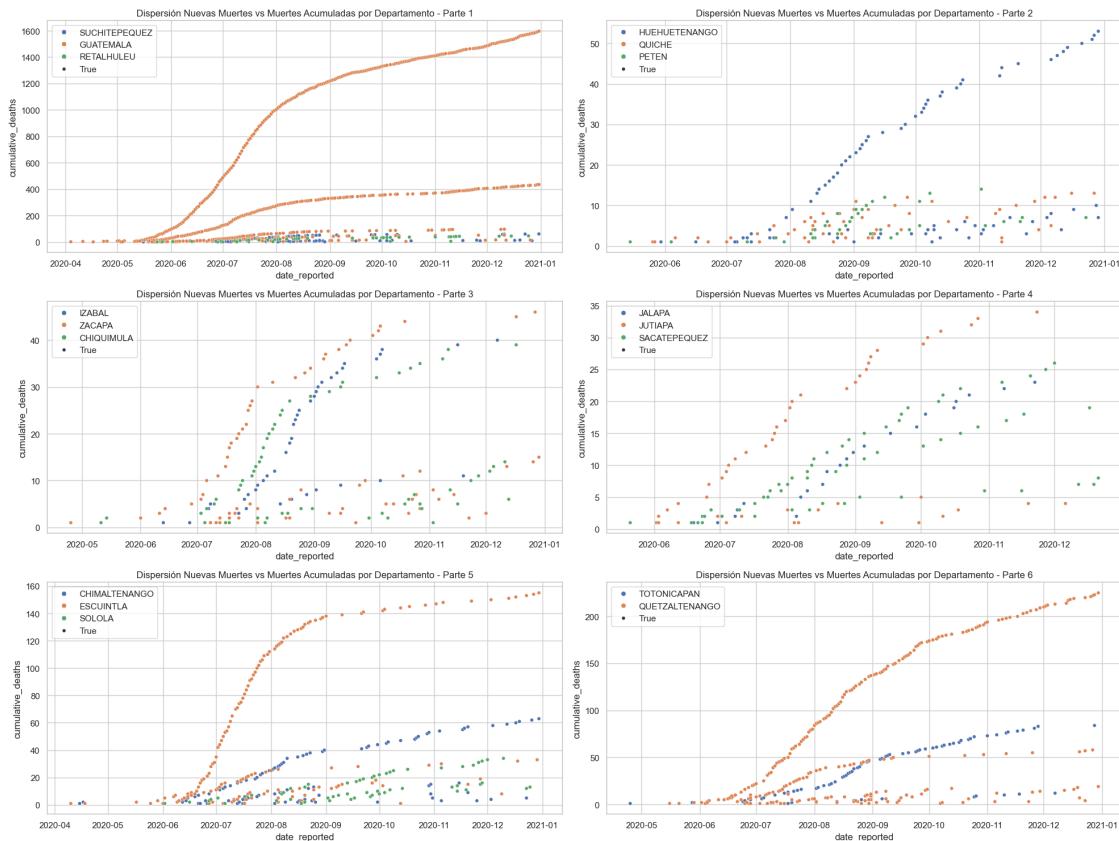
Muertes acumuladas por Municipio segun fechas

```
[164]: scatter_plot_multi(df_mdp_gt_0, 'date_reported', 'cumulative_deaths',  
    ↴'municipality', 'Dispersión Nuevas Muertes vs Muertes Acumuladas por  
    ↴Municipio', size=True, points_per_subplot=10, subplots_per_row=2)
```



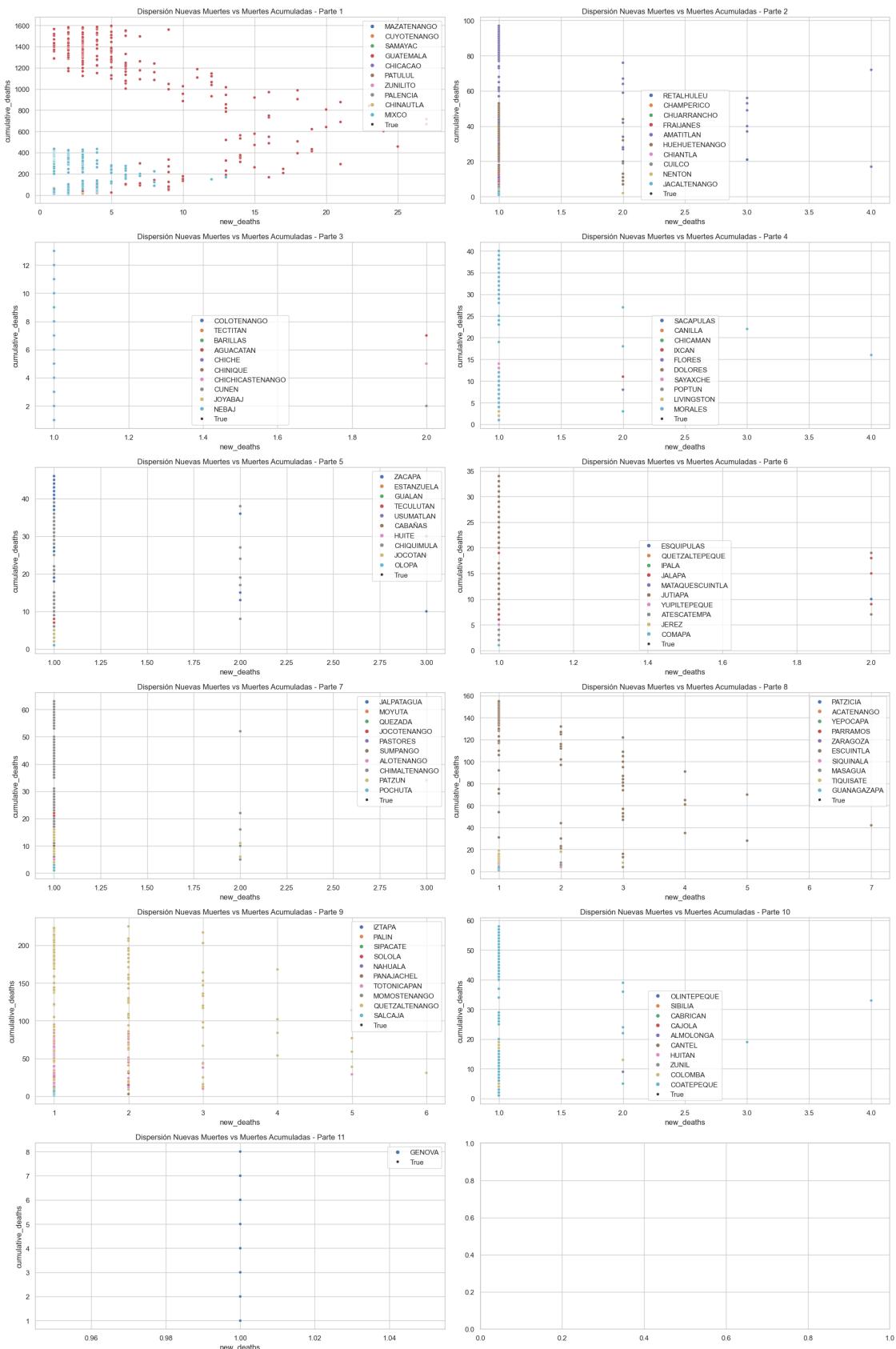
Muertes acumuladas por Departamento según fechas

```
[165]: scatter_plot_multi(df_mdp_gt_0, 'date_reported', 'cumulative_deaths',  
    ↪'name_department', 'Dispersión Nuevas Muertes vs Muertes Acumuladas por  
    ↪Departamento', size=True, points_per_subplot=3, subplots_per_row=2)
```



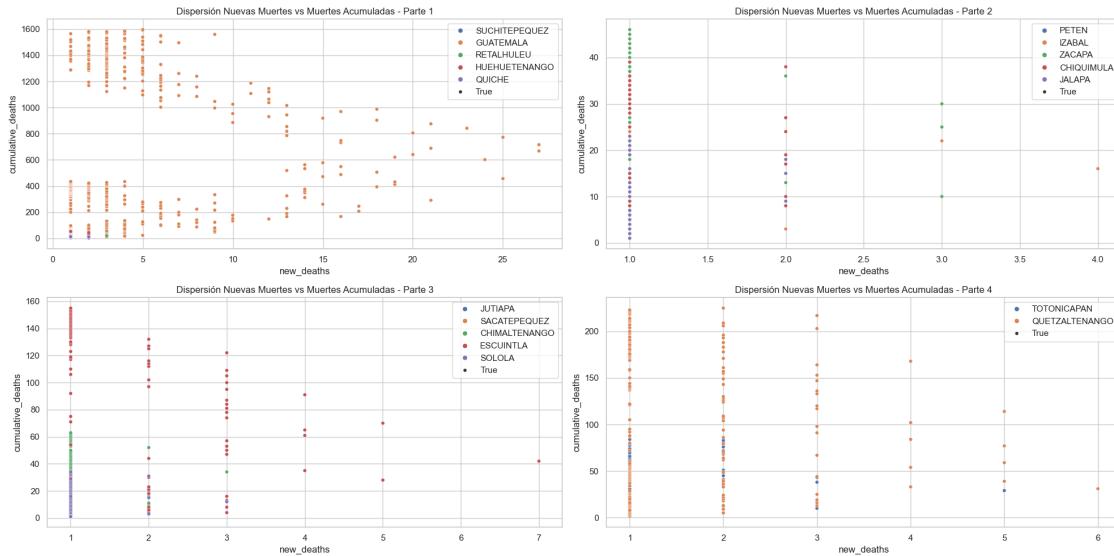
Nuevas muertes vs Muertes acumuladas por municipios

```
[166]: scatter_plot_multi(df_mdp_gt_0, 'new_deaths', 'cumulative_deaths',  
    ↪'municipality', 'Dispersión Nuevas Muertes vs Muertes Acumuladas',  
    ↪size=True, points_per_subplot=10, subplots_per_row=2)
```



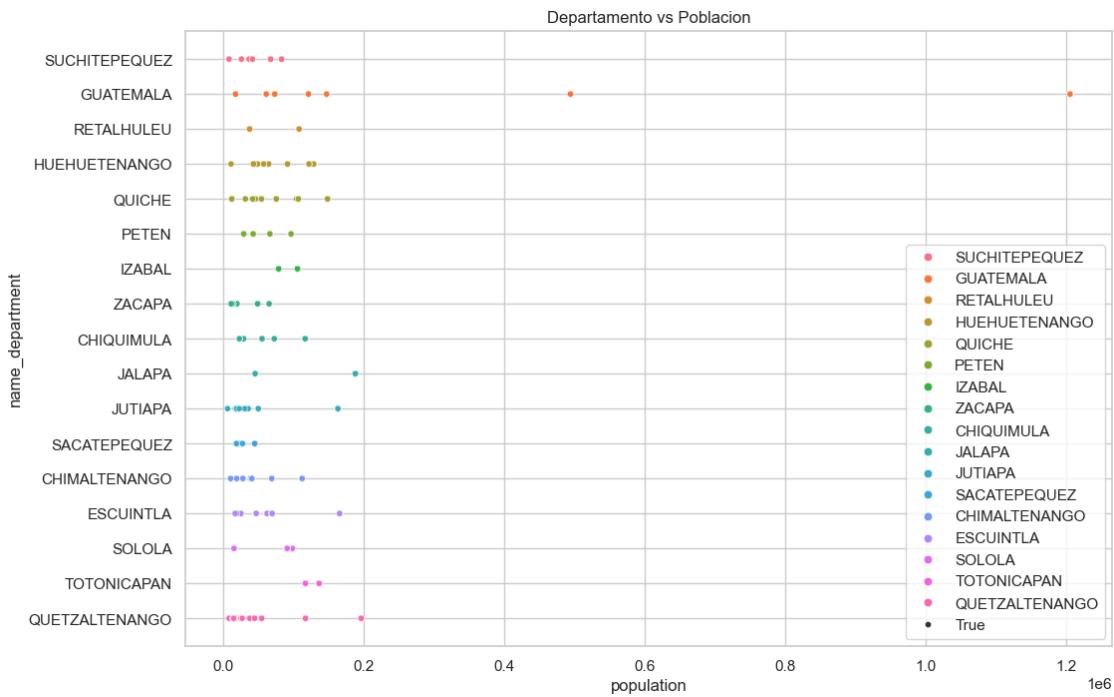
Nuevas muertes vs Muertes acumuladas por Departamentos

```
[167]: scatter_plot_multi(df_mdp_gt_0, 'new_deaths', 'cumulative_deaths',  
    ↪'name_department', 'Dispersión Nuevas Muertes vs Muertes Acumuladas',  
    ↪size=True, points_per_subplot=5, subplots_per_row=2)
```



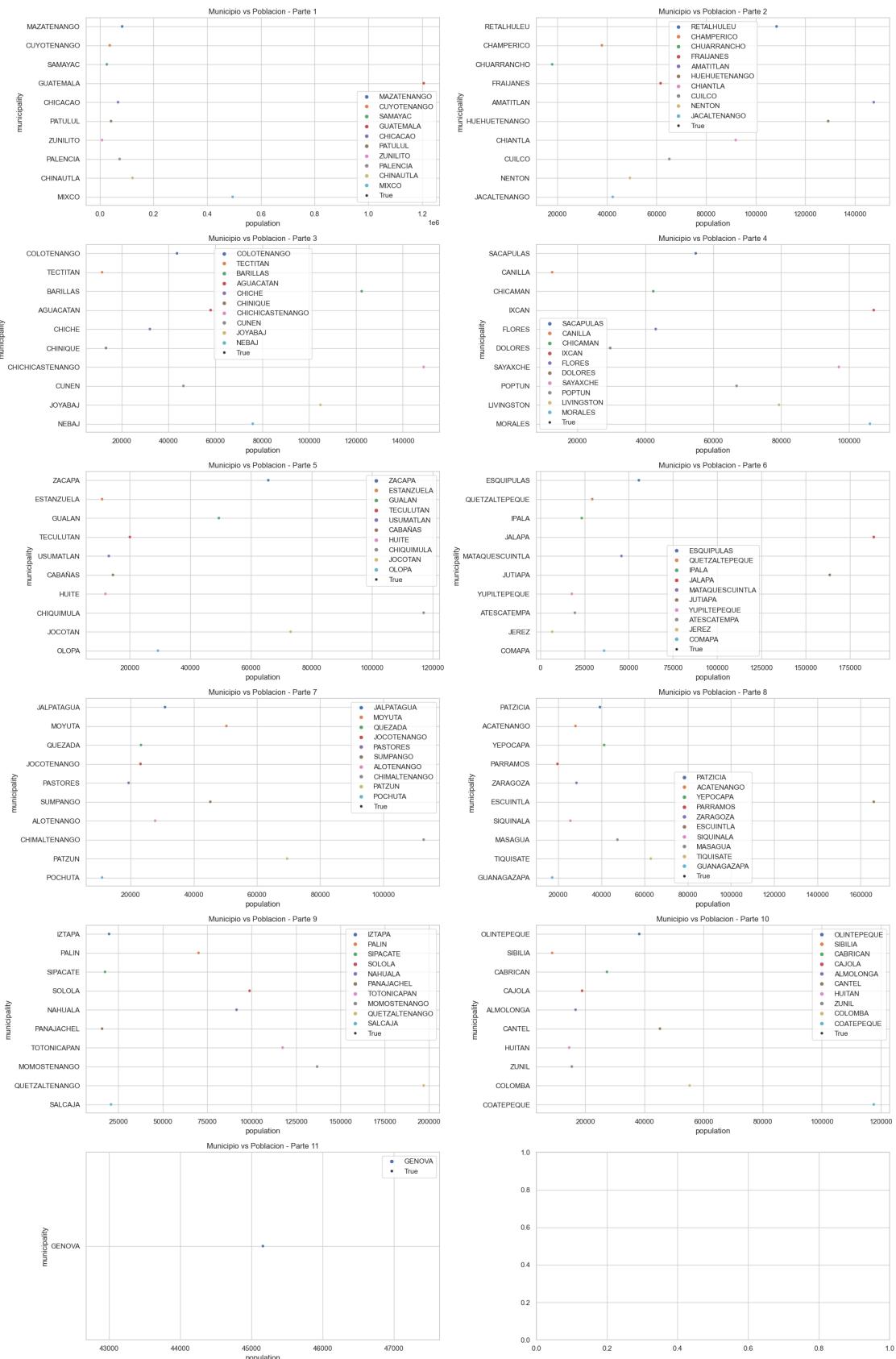
Departamento vs Poblacion

```
[168]: scatter_plot(df_mdp_gt_0, 'population', 'name_department', 'name_department',  
    ↪'Departamento vs Poblacion', True)
```



Municipio vs Población

```
[169]: scatter_plot_multi(df_mdp_gt_0, 'population', 'municipality', 'municipality',  
    ↪ 'Municipio vs Poblacion', size=True, points_per_subplot=10,  
    ↪ subplots_per_row=2)
```



1.3.2 Datos cualitativos

Cantidad total de datos de Conteo de Registros de Municipios por Departamento: 112

Cantidad total de datos de Conteo de Registros de Municipios por Departamento: 112

Departamento

Municipios

Departamento

Municipio

QUICHE

14

JALAPA

3

QUETZALTENANGO

13

SOLOLA

3

HUEHUETENANGO

11

IZABAL

2

JUTIAPA

11

RETALHULEU

2

ESCUINTLA

8

CHIMALTENANGO

352

TOTONICAPAN

2

RETALHULEU

140

TOTAL

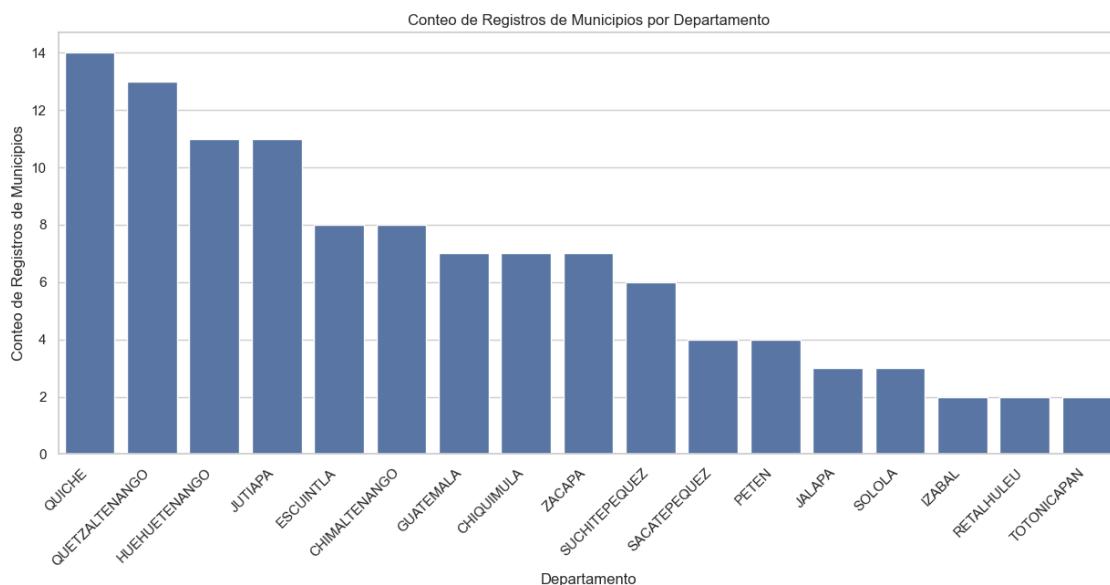
112

TOTAL

112

Departamento vs Cantidad de Municipios Observaciones: En el Análisis de Registros de Municipios por Departamento, QUICHE tiene la mayor frecuencia con 14 registros, seguido por QUETZALTENANGO con 13. Y los cinco departamentos con mayor frecuencia son: 1. QUICHE: 14 registros 2. QUETZALTENANGO: 13 registros 3. HUEHUETENANGO: 11 registros 4. JUTIAPA: 11 registros 5. ESCUINTLA: 8 registros

```
[170]: # Generar diagrama de barras para departamentos: cantidad de veces que aparecen en los casos de Covid
generate_count_plot(df_municipality, 'name_department', 'Departamento', 'Conteo de Registros de Municipios', 'Conteo de Registros de Municipios por Departamento')
```



Municipios vs Población Análisis: El análisis de la población por municipio muestra un total de 7,554,936 personas distribuidas en varios municipios. El municipio más poblado es GUATEMALA con una población de 1,205,668 personas, seguido por MIXCO con 494,561 personas. En comparación, los municipios menos poblados son SIBILIA, ZUNILITO, PATZITE, PETATAN y JEREZ, con poblaciones que oscilan entre 8,766 y 6,706 personas.

Este análisis proporciona información valiosa sobre la distribución de la población en diferentes municipios. La concentración de población en GUATEMALA y otros municipios más grandes

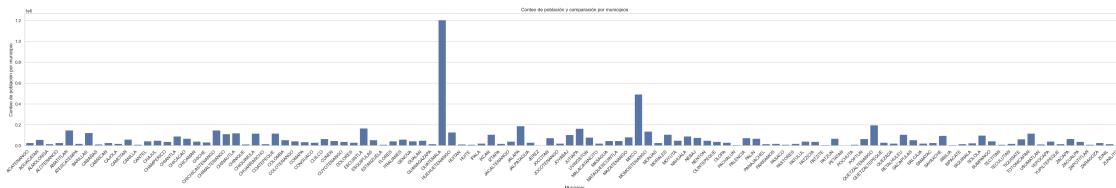
puede tener implicaciones significativas para la planificación y asignación de recursos.

Listado de Municipios

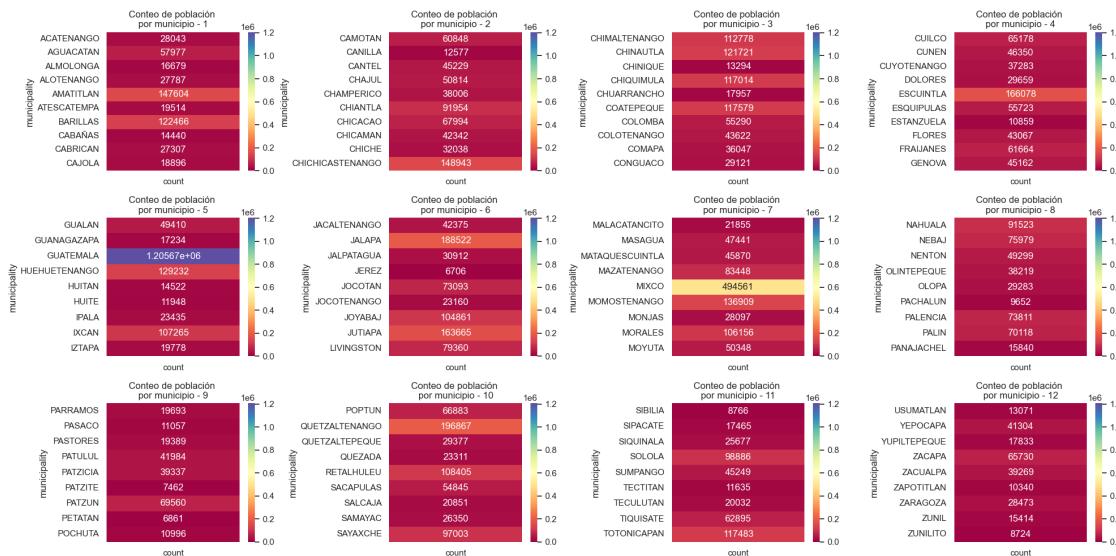
Top 5: 1. GUATEMALA: 1,205,668 personas 2. MIXCO: 494,561 personas 3. QUETZALTENANGO: 196,867 personas 4. JALAPA: 188,522 personas 5. ESCUINTLA: 166,078 personas

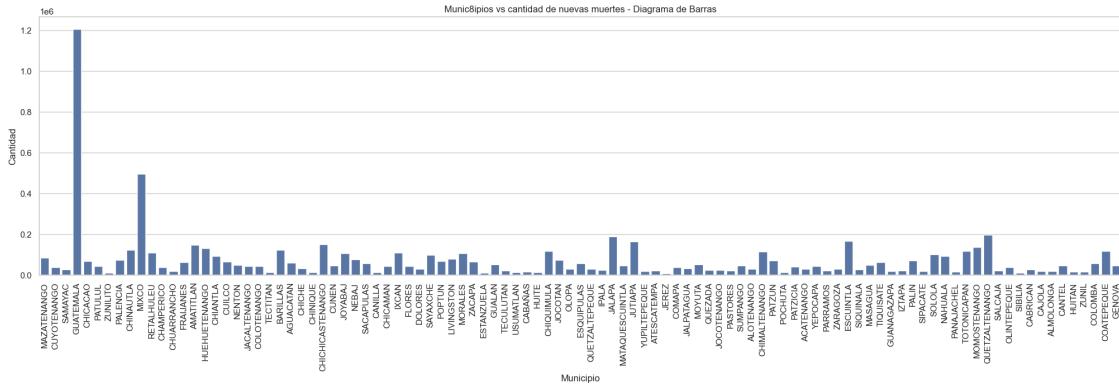
Peores 5: 1. SIBILIA: 8,766 personas 2. ZUNILITO: 8,724 personas 3. PATZITE: 7,462 personas
4. PETATAN: 6,861 personas 5. JEREZ: 6,706 personas

```
[171]: generate_sum_plot(df_municipality, 'population','municipality', 'Municipio',  
    ↴'Conteo de población por municipio', 'Conteo de población y comparación por  
    ↴municipios', True)  
generate_heatmaps(df_municipality, 'population', 'municipality',  
    ↴'Municipio', 'Conteo de población \npor municipio', 'Conteo de población y  
    ↴comparación por municipios', 10, 10, 4, False)  
generate_bar(df_mdp_gt_0, 'municipality', 'population', 'Municipio',  
    ↴'Cantidad', 'Municipios vs cantidad de nuevas muertes', 90, True)
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-  
packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is  
deprecated and will be removed in a future version. Please use  
'Series.transpose' instead.  
    return bound(*args, **kwds)
```





Departamentos vs Población Análisis: En cuanto a la población por departamento, la sumatoria de la población de todos los departamentos es también de 7,554,936 personas. GUATEMALA sigue siendo el departamento más poblado, con una población de 2,122,986 personas, seguido por QUICHE, HUEHUETENANGO, QUETZALTENANGO y ESCUINTLA. Los departamentos menos poblados incluyen SOLOLA, IZABAL, ZACAPA, RETALHULEU y SACATEPEQUEZ.

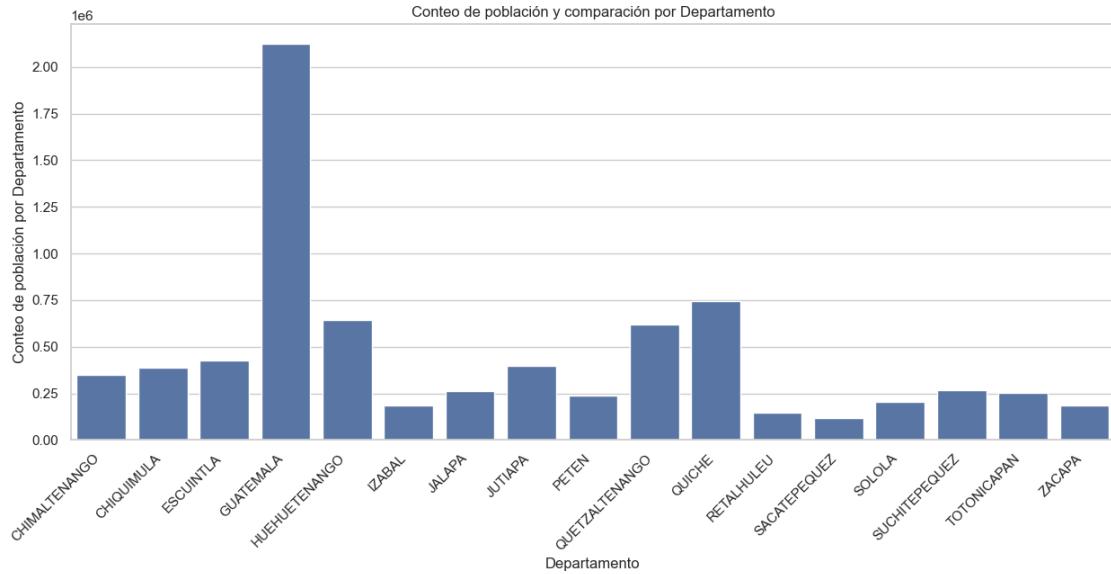
Este análisis permite entender la distribución de la población a nivel departamental. La alta población en GUATEMALA y otros departamentos más grandes destaca la importancia de considerar las necesidades específicas de estos lugares en políticas y programas gubernamentales.

Listado de Departamentos

Top 5: 1. GUATEMALA: 2,122,986 personas 2. QUICHE: 745,691 personas 3. HUEHUETENANGO: 642,454 personas 4. QUETZALTENANGO: 620,781 personas 5. ESCUINTLA: 426,686 personas

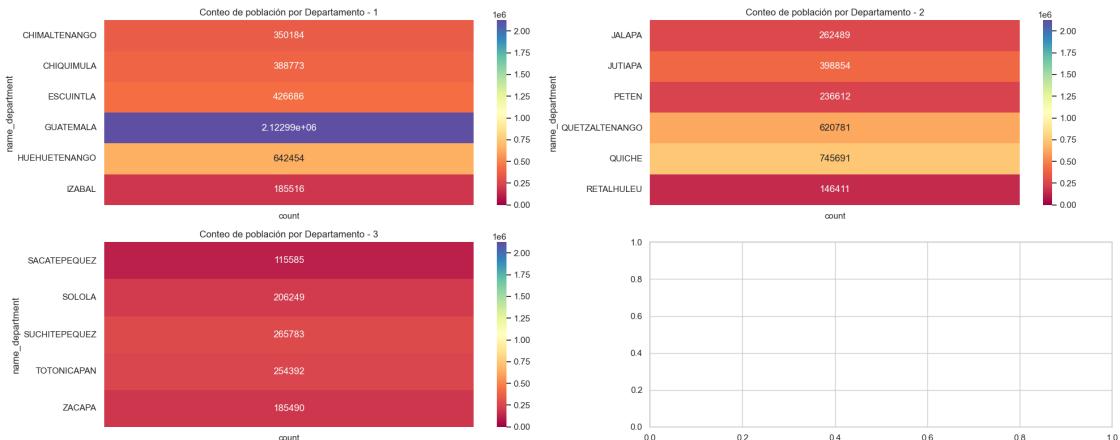
Peores 5: 1. SOLOLA: 206,249 personas 2. IZABAL: 185,516 personas 3. ZACAPA: 185,490 personas 4. RETALHULEU: 146,411 personas 5. SACATEPEQUEZ: 115,585 personas

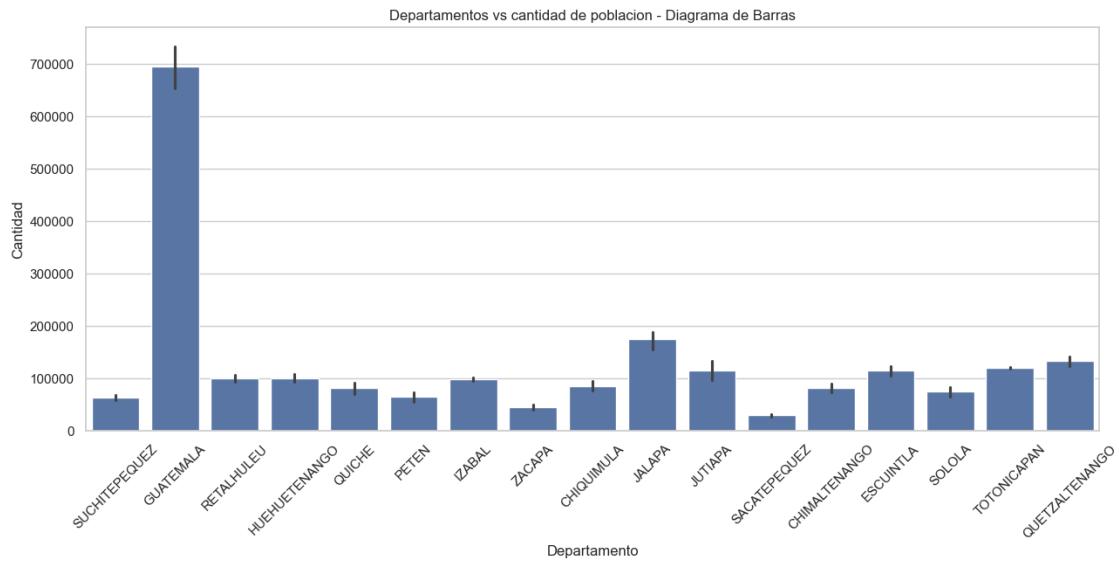
```
[172]: generate_sum_plot(df_municipality, 'population', 'name_department',  
    ↴'Departamento', 'Conteo de población por Departamento', 'Conteo de población  
    ↴y comparación por Departamento')  
generate_heatmaps(df_municipality, 'population', 'name_department',  
    ↴'Departamento', 'Conteo de población por Departamento', 'Conteo de población  
    ↴y comparación por Departamento', 10, 8, 2, False)  
generate_bar(df_mdp_gt_0, 'name_department', 'population', 'Departamento',  
    ↴'Cantidad', 'Departamentos vs cantidad de población')
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-
packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is
deprecated and will be removed in a future version. Please use
'Series.transpose' instead.

    return bound(*args, **kwds)
```





Municipios vs Cantidad de registros Nuevas Muertes por COVID-19 (2020) Municipio con más registros de nuevas muertes

Municipio

Municipio con menos registros de nuevas muertes

GUATEMALA

235

OLOPA

1

MIXCO

174

TECTITAN

1

QUETZALTENANGO

126

CUNEN

1

ESCUINTLA

83

SIPACATE

1

AMATITLAN

70

JEREZ

1

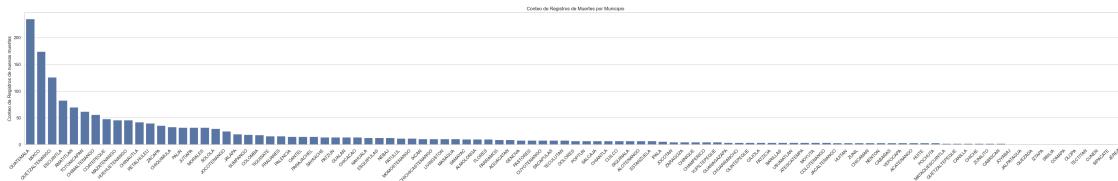
El análisis de muertes por municipio en el año 2020 muestra un total de 1,813 registros. El municipio más frecuente es GUATEMALA con 235 casos.

Total de registros en 2020: 1,813

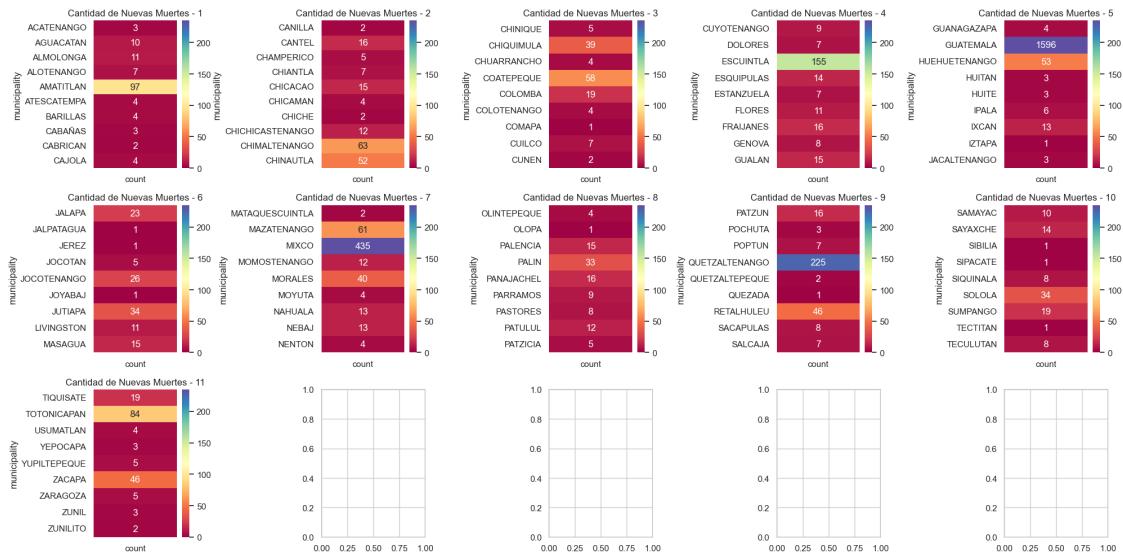
Este análisis proporciona una visión detallada de cómo las nuevas muertes por COVID-19 se distribuyen entre diferentes municipios en el año 2020. Identificar los municipios con mayor incidencia es crucial para la planificación y ejecución de estrategias de salud pública a nivel local.

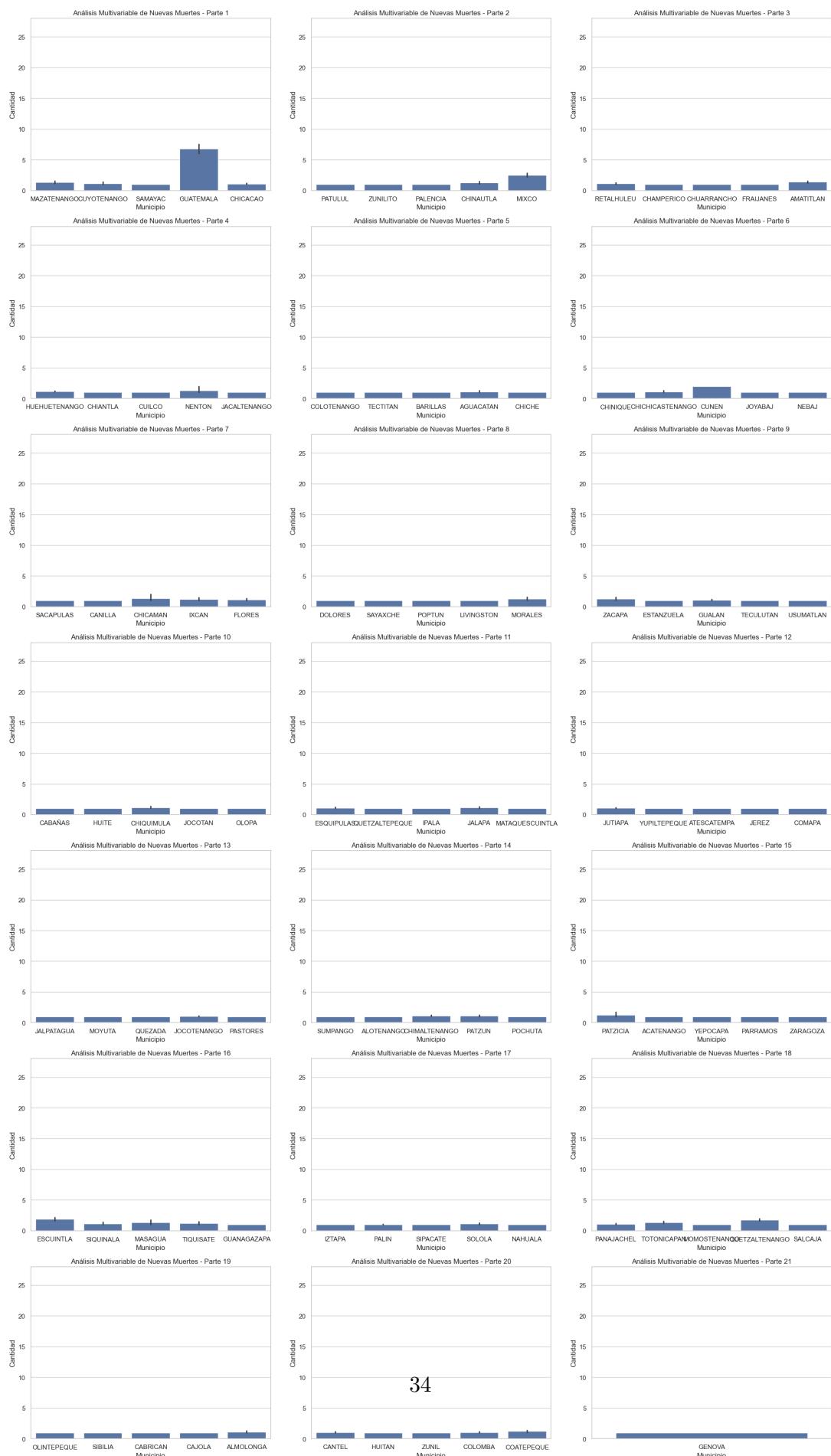
Además, se puede apreciar por la gráfica del mapa de calor qué municipios tienen una cantidad de casos mayor.

```
[173]: # Generar diagrama de barras para municipios: cantidad de veces que aparecen en los casos de covid
generate_count_plot(df_mdp_gt_0, 'municipality', 'Municipio', 'Conteo de Registros de nuevas muertes', 'Conteo de Registros de Muertes por Municipio', True)
generate_heatmaps(df_mdp_gt_0, 'new_deaths', 'municipality', 'Municipio', 'Cantidad de Nuevas Muertes', 'Municipios vs cantidad de nuevas muertes', 1, 10, 5)
# generate_multivariable_bar(df_mdp_gt_0, 'municipality', 'new_deaths', 'Municipio', 'Cantidad', 'Análisis Multivariable de Nuevas Muertes')
generate_multivariable_bar_subplots(df_mdp_gt_0, 'municipality', 'new_deaths', 'Municipio', 'Cantidad', 'Análisis Multivariable de Nuevas Muertes', bars_per_subplot=5, subplots_per_row=3)
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is deprecated and will be removed in a future version. Please use 'Series.transpose' instead.
    return bound(*args, **kwds)
```





Departamentos vs Cantidad de registros Nuevas Muertes por COVID-19 (2020) El Análisis de Muertes por Departamento muestra un total de 1813 registros. El Departamento más frecuente es GUATEMALA con 556 casos.

Departamentos vs Cantidad de registros de nuevas muertes

Departamentos vs Cantidad de registros de nuevas muertes

Departamentos vs Cantidad de registros de nuevas muertes

GUATEMALA

556

ZACAPA

75

JUTIAPA

49

QUETZALTENANGO

249

TOTONICAPAN

74

RETALHULEU

45

ESCUINTLA

155

CHIQUIMULA

60

IZABAL

43

CHIMALTENANGO

97

SACATEPEQUEZ

59

PETEN

38

SUCHITEPEQUEZ

92

SOLOLA

58

JALAPA

22

HUEHUETENANGO

84

QUICHE

57

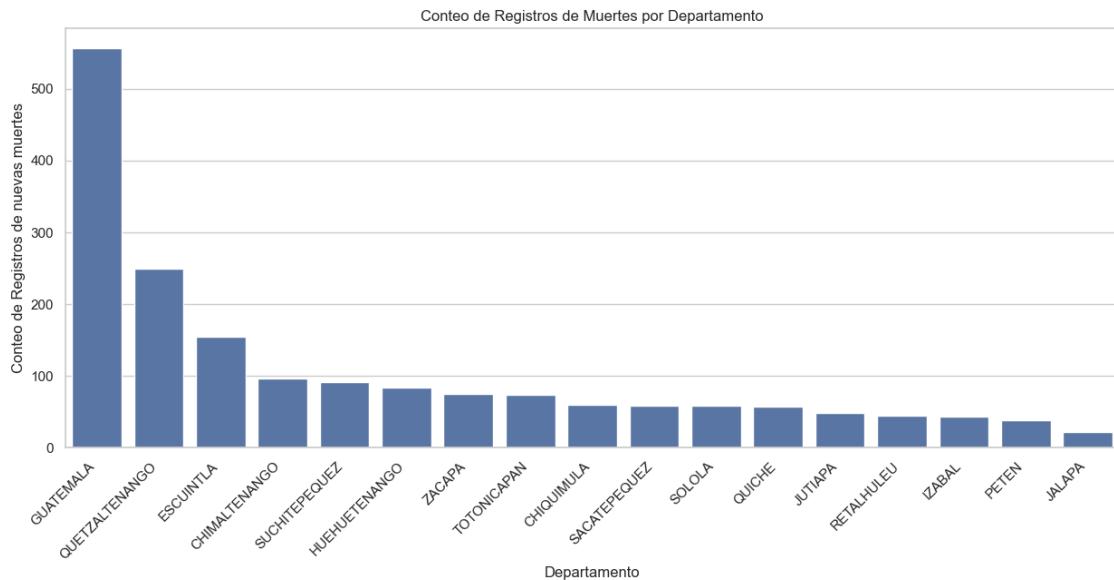
TOTAL

1813

Este análisis proporciona una visión detallada de cómo las nuevas muertes por COVID-19 se distribuyeron en diferentes municipios durante el año 2020 en Guatemala. Identificar los municipios con mayor incidencia es crucial para la planificación y ejecución de estrategias de salud pública a nivel local.

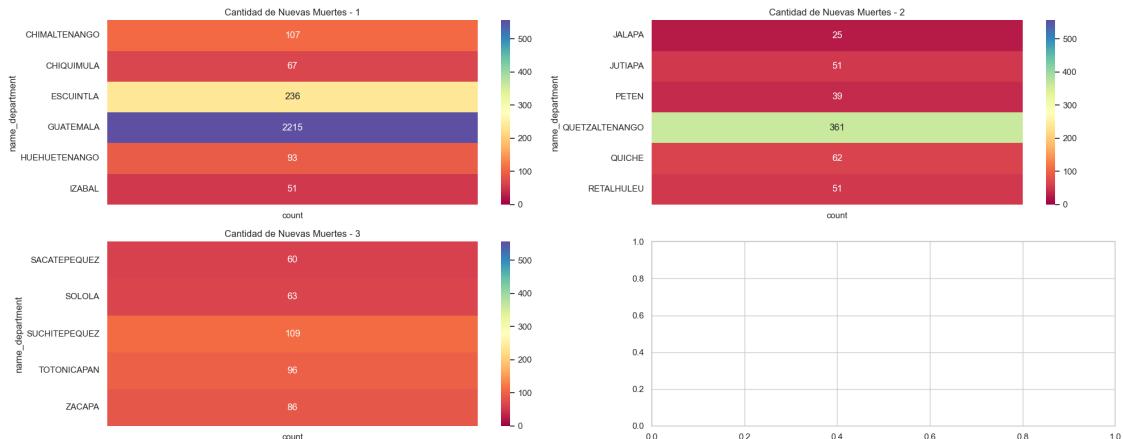
[174]: # Generar diagrama de barras para departamentos: cantidad de veces que aparecen en los casos de covid

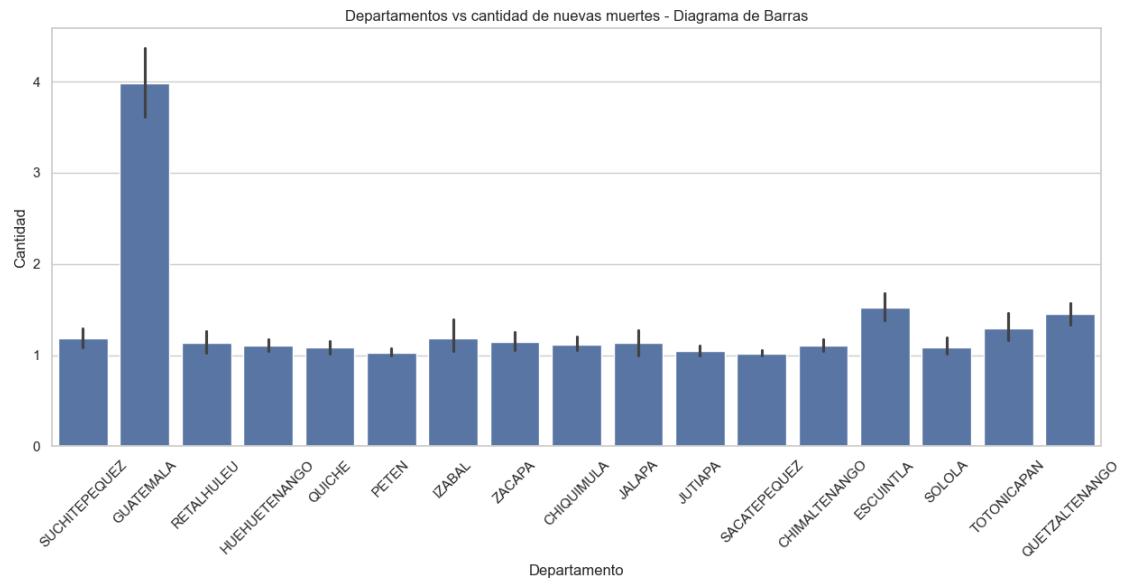
```
generate_count_plot(df_mdp_gt_0, 'name_department', 'Departamento', 'Conteo de Registros de nuevas muertes', 'Conteo de Registros de Muertes por Departamento')
generate_heatmaps(df_mdp_gt_0, 'new_deaths', 'name_department', 'Departamento', 'Cantidad de Nuevas Muertes', 'Departamentos vs cantidad de nuevas muertes', 10, 8, 2)
generate_bar(df_mdp_gt_0, 'name_department', 'new_deaths', 'Departamento', 'Cantidad', 'Departamentos vs cantidad de nuevas muertes')
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is deprecated and will be removed in a future version. Please use 'Series.transpose' instead.
```

```
    return bound(*args, **kwds)
```





Municipios vs Cantidad de Muertes Acumuladas Top 5 Municipios con más muertes acumuladas

Top 5 Municipios con menos muertes acumuladas

Municipio

Muertes Acumuladas

Municipio

Muertes acumuladas

GUATEMALA

1596

SIPACATE

1

MIXCO

435

SIBILIA

1

QUETZALTENANGO

225

JOYABAJ

1

ESCUINTLA

155

OLOPA

1

AMATITLAN

97

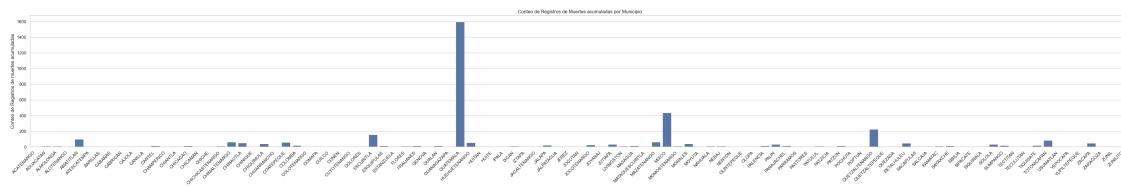
TECTITAN

1

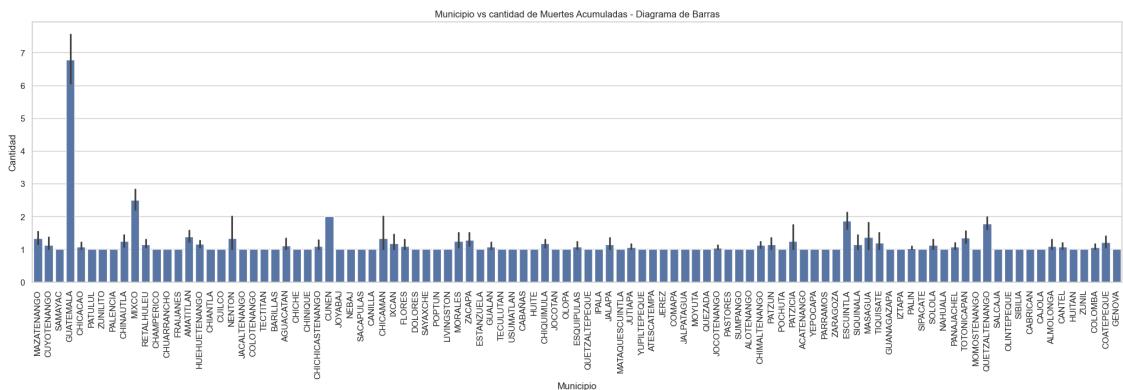
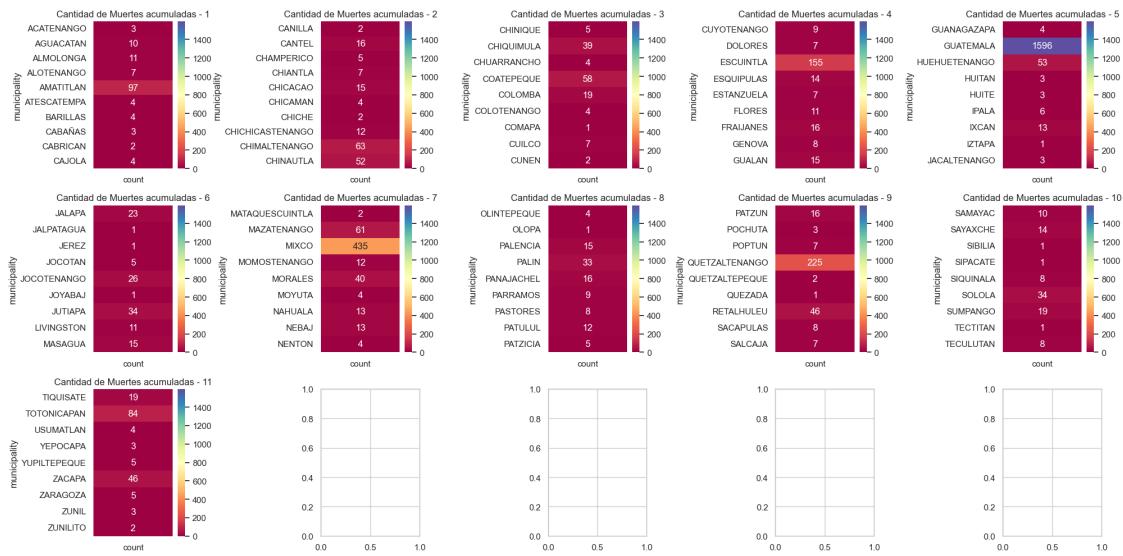
Cantidad total de datos de Conteo de Registros de Muertes Acumuladas por Municipio en 2020: 3772

Este análisis destaca la concentración de muertes acumuladas en algunos municipios específicos durante el año 2020 en Guatemala. La información actualizada proporciona una visión detallada de la distribución de nuevas muertes y muertes acumuladas en diferentes municipios, lo que puede ser crucial para la toma de decisiones en políticas de salud pública a nivel local. La variabilidad en los números también sugiere posibles disparidades en la gestión de la crisis sanitaria a nivel municipal.

```
[175]: # Generar diagrama de barras para municipios: cantidad de veces que aparecen en los casos de covid
generate_sum_plot(df_mdp_gt_0, 'new_deaths', 'municipality', 'Municipio',
                   'Conteo de Registros de muertes acumuladas', 'Conteo de Registros de Muertes acumuladas por Municipio', True)
generate_heatmaps(df_mdp_gt_0, 'new_deaths', 'municipality', 'Municipio',
                   'Cantidad de Muertes acumuladas', 'Municipios vs cantidad de muertes acumuladas', 1, 10, 5, False)
generate_bar(df_mdp_gt_0, 'municipality', 'new_deaths', 'Municipio',
             'Cantidad', 'Municipio vs cantidad de Muertes Acumuladas', 90, True)
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is deprecated and will be removed in a future version. Please use 'Series.transpose' instead.
    return bound(*args, **kwds)
```



Departamentos vs Cantidad de Muertes Acumuladas Top 5 Departamentos con más Muertes Acumuladas en 2020

Top 5 Departamentos con menos Muertes Acumuladas en 2020

Departamento

Muertes Acumuladas

Departamento

Muertes Acumuladas

GUATEMALA

51

QUETZALTENANGO

361

RETALHULEU

51

ESCUINTLA

236

IZABAL

51

SUCHITEPEQUEZ

109

PETEN

39

CHIMALTENANGO

107

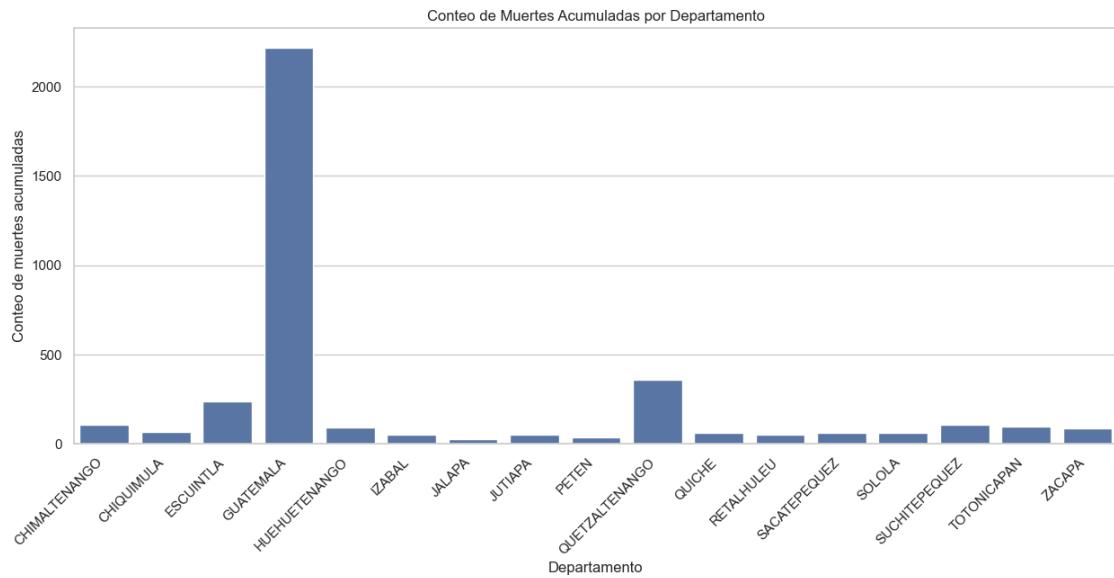
JALAPA

25

Cantidad total de datos de Conteo de Muertes Acumuladas por Departamento en 2020: 3772

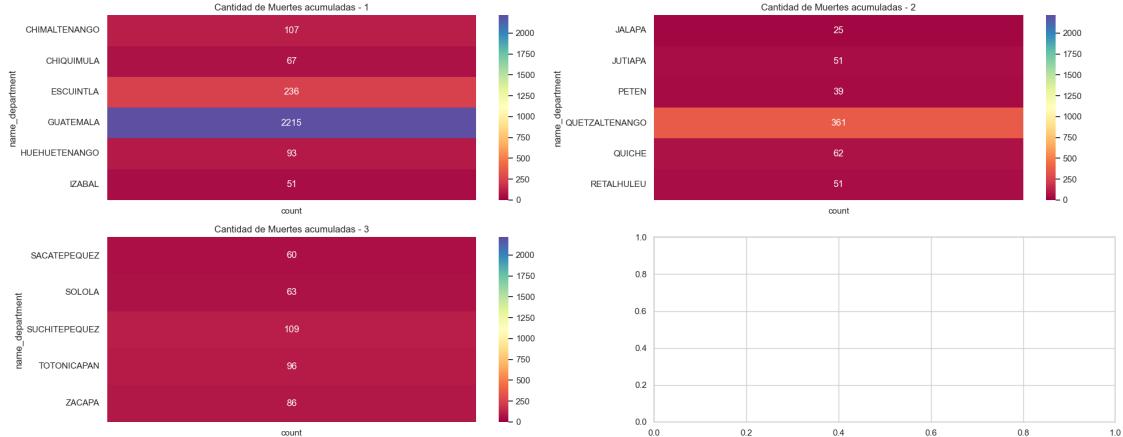
Este análisis resalta la variabilidad en la cantidad de muertes acumuladas en diferentes departamentos durante el año 2020 en Guatemala. La información actualizada proporciona una visión detallada de la distribución geográfica de la mortalidad, destacando los departamentos con mayor y menor incidencia. La variabilidad en los números sugiere posibles disparidades en la gestión de la crisis sanitaria a nivel departamental.

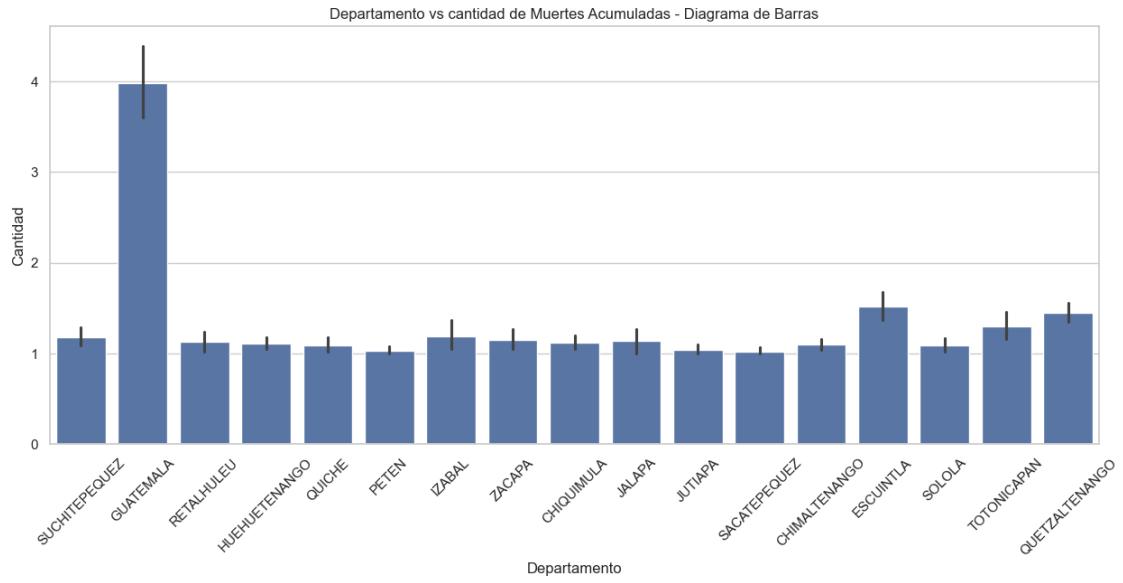
```
[176]: # Generar diagrama de barras para departamentos: cantidad de veces que aparecen en los casos de covid
generate_sum_plot(df_mdp_gt_0, 'new_deaths', 'name_department', 'Departamento', 'Conteo de muertes acumuladas', 'Conteo de Muertes Acumuladas por Departamento', False)
generate_heatmaps(df_mdp_gt_0, 'new_deaths', 'name_department', 'Departamento', 'Cantidad de Muertes acumuladas', 'Departamentos vs cantidad de muertes acumuladas', 10, 8, 2, False)
generate_bar(df_mdp_gt_0, 'name_department', 'new_deaths', 'Departamento', 'Cantidad', 'Departamento vs cantidad de Muertes Acumuladas', 45)
```



```
d:\Projects\SS2_Proyecto_Fase_1\Fase_2\venv\Lib\site-
packages\numpy\core\fromnumeric.py:59: FutureWarning: 'Series.swapaxes' is
deprecated and will be removed in a future version. Please use
'Series.transpose' instead.
```

```
    return bound(*args, **kwds)
```





1.4 Conclusiones

1. Población por Municipio y Departamento:

- El municipio más poblado es **Guatemala** con una población de 1,205,668 personas, mientras que el departamento más poblado es también **Guatemala** con 2,122,986 personas.
- Existe una variabilidad significativa en la población entre los municipios y departamentos, destacando la concentración en áreas urbanas como Guatemala y Mixco.

2. Nuevas Muertes y Muertes Acumuladas:

- **Guatemala** (tanto a nivel de departamento como municipio) muestra la mayor cantidad de nuevas muertes y muertes acumuladas. Esto destaca la necesidad de medidas preventivas y recursos adicionales en esta área.
- Los diagramas de dispersión entre **Nuevas Muertes vs Muertes Acumuladas** por municipio y departamento indican una relación positiva, sugiriendo que los lugares con más nuevas muertes también tienden a tener más muertes acumuladas.

3. Distribución Geográfica de Nuevas Muertes:

- El análisis de mapas de calor revela que los municipios de **Guatemala, Mixco, y Quetzaltenango** experimentan una cantidad significativa de nuevas muertes. Esto subraya la necesidad de estrategias específicas y atención en estas áreas.

4. Muertes Acumuladas:

- **Guatemala** lidera en muertes acumuladas tanto a nivel de municipio como de departamento, seguido por **Mixco** y **Quetzaltenango**. Esta concentración destaca la importancia de recursos y esfuerzos adicionales en estas ubicaciones.

1.5 Recomendaciones

1. Asignación de Recursos:

- Priorizar la asignación de recursos y atención médica en áreas urbanas densamente

pobladas, especialmente en el departamento y municipio de **Guatemala**, para gestionar efectivamente la carga de nuevas muertes y muertes acumuladas.

2. Medidas Preventivas:

- Implementar medidas preventivas y campañas de concientización en los municipios y departamentos con mayores tasas de nuevas muertes, como **Guatemala, Mixco, y Quetzaltenango**.

3. Planificación Estratégica:

- Desarrollar planes estratégicos específicos para abordar las disparidades en la distribución geográfica de las muertes acumuladas. Esto puede incluir campañas de vacunación, recursos adicionales para el sistema de salud y medidas de mitigación.

4. Monitoreo Continuo:

- Establecer un sistema de monitoreo continuo para evaluar la evolución de la situación en cada municipio y departamento. Esto permitirá una respuesta rápida y ajustes en las estrategias según sea necesario.

5. Coordinación Interdepartamental:

- Fomentar la coordinación y colaboración entre los departamentos para compartir mejores prácticas, recursos y estrategias efectivas. El intercambio de información facilitará una respuesta más eficiente y coordinada a nivel nacional.

NOTA: Estas conclusiones y recomendaciones están basadas en el análisis de los datos proporcionados y buscan proporcionar orientación para abordar los desafíos específicos relacionados con la pandemia de COVID-19 en Guatemala. Es esencial adaptar las estrategias a la evolución de la situación y considerar la dinámica única de cada región.