```
!pip install pygwalker
!pip install bokeh
!pip install db-sqlite3
```

Show hidden output

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import bokeh.plotting as bp
import pygwalker as pyg
from bokeh.io import output_notebook
from bokeh.models import ColumnDataSource, Range1d



from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
#---------------LECTURA DE SQL-----------------------------
archivo_sql_mysql = '/content/drive/MyDrive/Colab Notebooks/DataPersonasCOVID.sql'

# Conectar a la base de datos SQLite en memoria
conn = sqlite3.connect(':memory:')

# Crear un cursor para ejecutar comandos SQL
cursor = conn.cursor()

# Leer el contenido del archivo SQL de MySQL
with open(archivo_sql_mysql, 'r') as archivo:
    sql_mysql = archivo.read()

# Ejecutar las instrucciones SQL de MySQL para crear la tabla y cargar los datos
cursor.executescript(sql_mysql)

# Consulta SQL para seleccionar todos los datos de la tabla "tabla_datos"
consulta_sql = "SELECT * FROM tabla_datos"

# Leer los datos de la tabla en un DataFrame de pandas
dataSQL = pd.read_sql_query(consulta_sql, conn)

# Cerrar la conexión a la base de datos SQLite
conn.close()

# Mostrar el DataFrame
dataSQL
```

	passport	age	provincia
0	55846403	43	Esmeraldas
1	R46013317	67	Pichincha
2	803556163	45	Pichincha
3	636000673	32	Azuay
4	659826519	32	Esmeraldas
...	...	...	...
99995	929418838	68	Azuay
99996	S96245675	62	Guayas
99997	D80207398	38	Esmeraldas
99998	738672634	31	Azuay
99999	N22923476	64	Loja

100000 rows × 3 columns

Next steps: Generate code with dataSQL View recommended plots

```
#---------------LECTURA DEL CSV----------------------------
dataCSV = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/datasetCovid.csv")
# Mostrar el DataFrame
dataCSV.head(5)
```

	Unnamed: 0.1	Unnamed: 0	N	county	age_group	gender	date	cases	deaths	recovered
0	0	0	0	LK Alb-Donau-Kreis	00-04	F	3/27/2020	1	0	1
1	1	1	1	LK Alb-Donau-Kreis	00-04	F	3/28/2020	1	0	1

Next steps: Generate code with dataCSV View recommended plots

```
#---------------MERGE DE LOS DATAFRAMES----------------------------
dataMerge = pd.merge(dataSQL, dataCSV, on=['passport', 'provincia'])
# Mostrar el DataFrame
dataMerge.sample(5)
```

	passport	age	provincia	Unnamed: 0.1	Unnamed: 0	N	county	age_group	ge
58941	276331787	69	Azuay	62060	25516	25516	LK Bodenseekreis	14-May	
76666	B90298059	55	Guayas	80730	63924	63924	LK Esslingen	14-May	
8400	248360924	79	Azuay	8847	17660	17660	LK Boeblingen	15-34	

```
# Eliminar columnas de los indices provenientes de la base de datos y del CSV.
dataMerge = dataMerge.drop(dataMerge.columns[[3,4,5,6,7,10]], axis=1)
dataMerge.head(5)
```

	passport	age	provincia	gender	date	deaths	recovered
0	R46013317	67	Pichincha	F	3/28/2020	0	1
1	803556163	45	Pichincha	F	4/3/2020	0	1
2	636000673	32	Azuay	F	10/18/2020	0	1
3	659826519	32	Esmeraldas	F	10/22/2020	0	1
4	765930886	50	Guayas	F	10/27/2020	0	1

Next steps: Generate code with dataMerge View recommended plots

```
# Función CategorizarEstado()
# Permite realizar procesamiento de categorización para conocer el estado del paciente con Covid.
# 0: No infectado (NotInfected)
# 1: Recuperado (Recovered)
# 2: Fallecido (Death)

def categorizarEstado(registro):
    eventDeaths = registro.deaths
    eventRecovered = registro.recovered
    if eventDeaths > 0:
            return "Death"  #2: Muerto
    elif eventDeaths == eventRecovered:
            return "NotInfected"  #0: No infectado
    else:
            return "Recovered"  #1: Recuperado
```

```
# Función CategorizarEdad()
# Permite realizar procesamiento de categorización para conocer el estado del paciente con Covid.
# 14-26 años: Joven
# 26-35 años: Adulto Joven
# 35-59 años: Adulto
# 60 o más: Tercera Edad
def categorizarEdad(registro):
    eventAge = registro.age
    if eventAge > 14 and eventAge < 26:
            return "Teen"  # Joven
    elif eventAge > 26 and eventAge < 35:
            return "YoungAdult"  # Adulto Joven
    elif eventAge > 35 and eventAge < 59:
            return "Adult"  # Adulto
    else:
            return "ThirdAge"  # Tercera Edad


# Categorizar el estado de los pacientes
#----------------------PRIMERA COLUMNA CREADA---------------------------
dataMerge['statePerson'] = dataMerge.apply(categorizarEstado, axis=1)
dataMerge.sample(10)
```

	passport	age	provincia	gender	date	deaths	recovered	statePerson
94142	787761753	40	Loja	M	4/12/2022	0	9	Recovered
53395	U83138680	46	Esmeraldas	M	6/2/2021	0	1	Recovered
92922	N32367541	35	Loja	M	8/9/2022	0	47	Recovered
78431	298430732	60	Loja	M	11/26/2021	0	61	Recovered
39058	F00790150	68	Guayas	F	10/21/2022	0	26	Recovered
25655	H41688379	56	Pichincha	F	5/2/2022	0	11	Recovered
16024	H59120548	67	Pichincha	F	10/24/2022	0	18	Recovered
32049	V00016920	69	Esmeraldas	F	9/24/2022	0	8	Recovered
56192	785559711	62	Pichincha	M	11/2/2022	0	27	Recovered
37534	U35429920	66	Esmeraldas	F	12/2/2022	0	6	Recovered

```
# Eliminar columnas que no seran usadas por el dataset
# Drop Deaths and Recovered
dataMerge = dataMerge.drop(dataMerge.columns[[5,6]], axis=1) #Drop Deaths and Recovered
dataMerge.sample(10)
```

	passport	age	provincia	gender	date	statePerson
3503	145321923	39	Azuay	F	12/28/2021	Recovered
79147	K48864374	35	Azuay	M	10/1/2021	Recovered
4776	659295089	67	Guayas	F	7/29/2022	Recovered
84133	413722409	77	Esmeraldas	M	8/6/2020	Recovered
69385	364570593	36	Esmeraldas	M	6/21/2021	Recovered
1053	391338414	67	Azuay	F	4/18/2021	Recovered
22696	221574484	63	Loja	F	1/2/2022	Recovered
76025	219586190	47	Esmeraldas	M	8/3/2021	Recovered
71516	Y17842931	47	Guayas	M	4/28/2021	Recovered
50212	B14165634	48	Guayas	M	3/5/2022	Recovered

```
# Categorizar la fecha y colocar en diferentes columnas
#----------------------2DA, 3RA Y 4TA COLUMNA CREADA----------------------

dataMerge['year'] = pd.to_datetime(dataMerge['date']).dt.year
dataMerge['month'] = pd.to_datetime(dataMerge['date']).dt.month
dataMerge['day'] = pd.to_datetime(dataMerge['date']).dt.day
dataMerge.sample(5)
```

	passport	age	provincia	gender	date	statePerson	year	month	day
10435	L97857874	31	Loja	F	1/29/2023	NotInfected	2023	1	29
17325	365483915	34	Guayas	F	3/2/2022	Recovered	2022	3	2
44348	349865266	38	Guayas	F	5/14/2021	Recovered	2021	5	14
54567	247979536	38	Guayas	M	4/28/2022	Recovered	2022	4	28
2571	879448627	74	Pichincha	F	11/7/2021	Recovered	2021	11	7

```
# Eliminar columnas que no seran usadas por el dataset
# Drop Date
dataMerge = dataMerge.drop(dataMerge.columns[[4]], axis=1)
dataMerge.sample(5)
```

	passport	age	provincia	gender	statePerson	year	month	day
92956	547981735	48	Guayas	M	Recovered	2022	9	16
15287	M13531405	46	Guayas	F	Recovered	2022	12	7
81358	835224028	78	Guayas	M	Recovered	2022	4	29
90883	205939964	79	Pichincha	M	Recovered	2021	4	28
87405	856842582	41	Pichincha	M	Recovered	2021	4	15

```
dataMerge.isnull().sum().sort_values(ascending=False)/len(dataMerge)*100
```

```
passport     0.0
age          0.0
provincia    0.0
gender       0.0
statePerson  0.0
year         0.0
month        0.0
day          0.0
ageGroup     0.0
dtype: float64
```

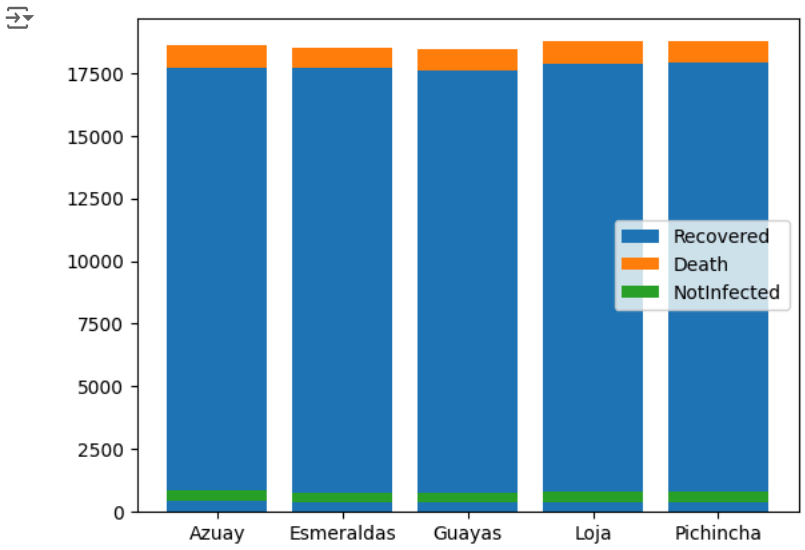
```
# Categorizar la edad
#----------------------5TA COLUMNA CREADA-----------------------

dataMerge['ageGroup'] = dataMerge.apply(categorizarEdad, axis=1)
dataMerge.sample(10)
```

	passport	age	provincia	gender	statePerson	year	month	day	ageGroup
90247	L98658584	74	Loja	M	Recovered	2023	1	2	ThirdAge
79692	808913428	71	Guayas	M	Death	2021	1	21	ThirdAge
24052	772167049	31	Loja	F	Recovered	2021	10	11	YoungAdult
52830	E34781933	54	Azuay	M	Recovered	2021	11	17	Adult
13110	508914181	50	Guayas	F	Recovered	2022	6	24	Adult
4032	617714783	39	Loja	F	Recovered	2022	5	5	Adult
9091	449983852	43	Loja	F	Recovered	2021	4	20	Adult
19747	237870698	33	Loja	F	Recovered	2020	11	24	YoungAdult
80541	M69511070	65	Loja	M	Recovered	2021	3	28	ThirdAge
14069	B79784755	44	Loja	F	Recovered	2022	3	3	Adult

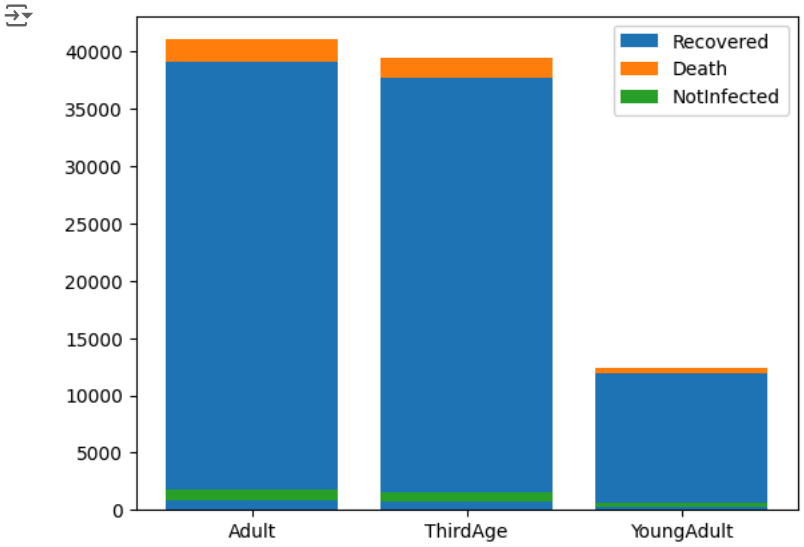
Una vez realizada la verificación, validación y procesamiento de los datos procedemos con la creación de la visualización de los gráficos.

Gráficos usando Matplotlib

```
state_counts = dataMerge.groupby('provincia')['statePerson'].value_counts()
# Create separate dataframes for each state
recovered = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'Recovered']
death = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'Death']
notInfected = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'NotInfected']

# Plot the count of each state
fig, ax = plt.subplots()
ax.bar(recovered.index.get_level_values('provincia'), recovered.values, label='Recovered')
ax.bar(death.index.get_level_values('provincia'), death.values, label='Death', bottom=recovered.values)
ax.bar(notInfected.index.get_level_values('provincia'), notInfected.values, label='NotInfected', bottom=notInfected.values)

# Add a legend to the plot
ax.legend()

# Show the plot
plt.show()
```

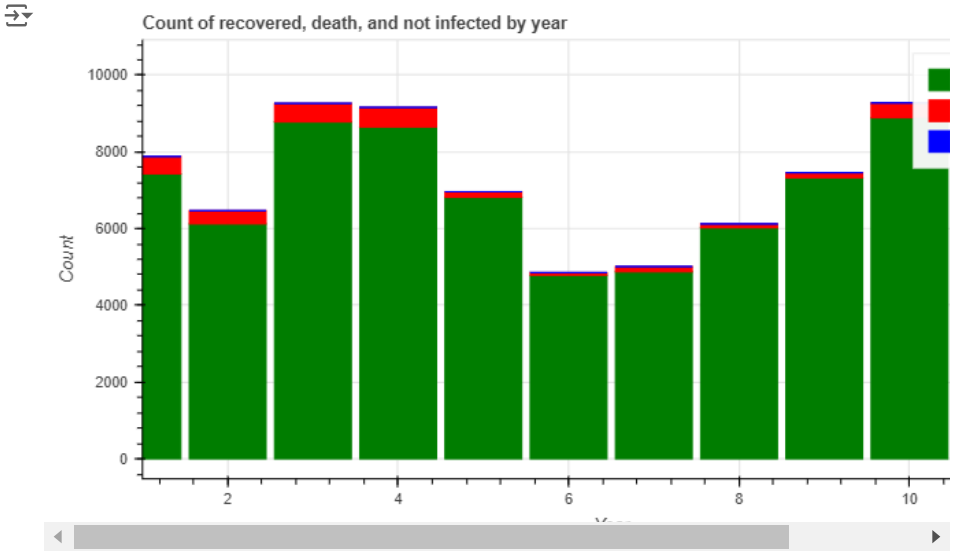



```
state_counts = dataMerge.groupby('ageGroup')['statePerson'].value_counts()
# Create separate dataframes for each state
recovered = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'Recovered']
death = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'Death']
notInfected = state_counts.loc[state_counts.index.get_level_values('statePerson') == 'NotInfected']

# Plot the count of each state
fig, ax = plt.subplots()
ax.bar(recovered.index.get_level_values('ageGroup'), recovered.values, label='Recovered')
ax.bar(death.index.get_level_values('ageGroup'), death.values, label='Death', bottom=recovered.values)
ax.bar(notInfected.index.get_level_values('ageGroup'), notInfected.values, label='NotInfected', bottom=notInfected.values)

# Add a legend to the plot
ax.legend()

# Show the plot
plt.show()
```

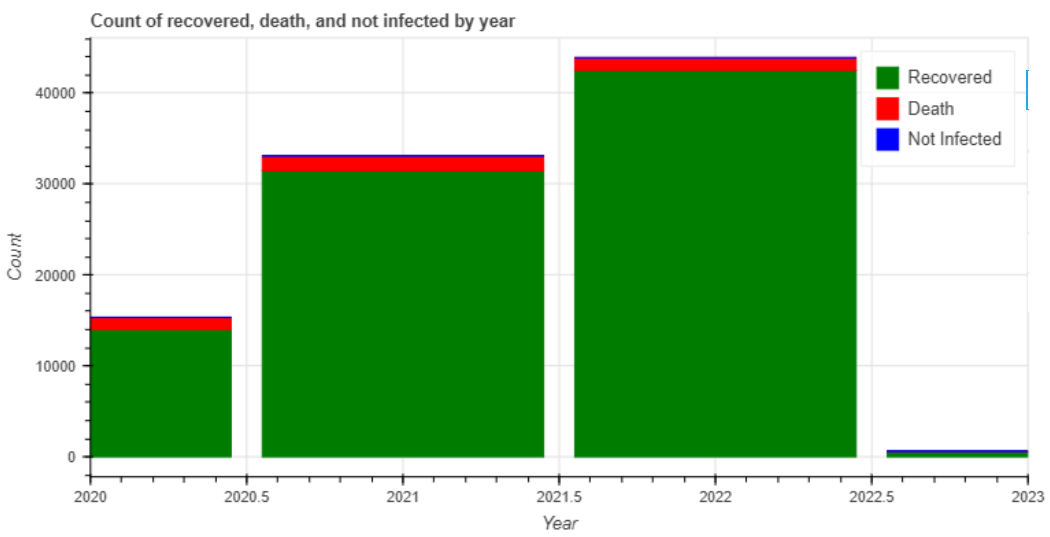


Gráficos usando Buked

```
# Group by year and statePerson, and count the outcomes
state_counts = dataMerge.groupby(['month', 'statePerson']).size().reset_index(name='count')

# Pivot the data to create separate columns for each statePerson
state_counts_pivot = state_counts.pivot(index='month', columns='statePerson', values='count').fillna(0)

# Create a ColumnDataSource from the pivoted data
source = ColumnDataSource(state_counts_pivot.reset_index())

# Create a Range1d object for the x_range argument
x_range = Range1d(start=min(state_counts_pivot.index), end=max(state_counts_pivot.index))

# Create a figure with a bar chart
p = bp.figure(x_range=x_range,
              width=800, height=400, title='Count of recovered, death, and not infected by year')

p.vbar_stack(['Recovered', 'Death', 'Notinfected'], x='month', width=0.9,
             color=['green', 'red', 'blue'], source=source, legend_label=['Recovered', 'Death', 'Not Infected'])
p.yaxis.axis_label = 'Count'
p.xaxis.axis_label = 'Year'
p.legend.location = 'top_right'

# Output the plot to a Jupyter notebook
output_notebook()

# Show the plot
bp.show(p)
```

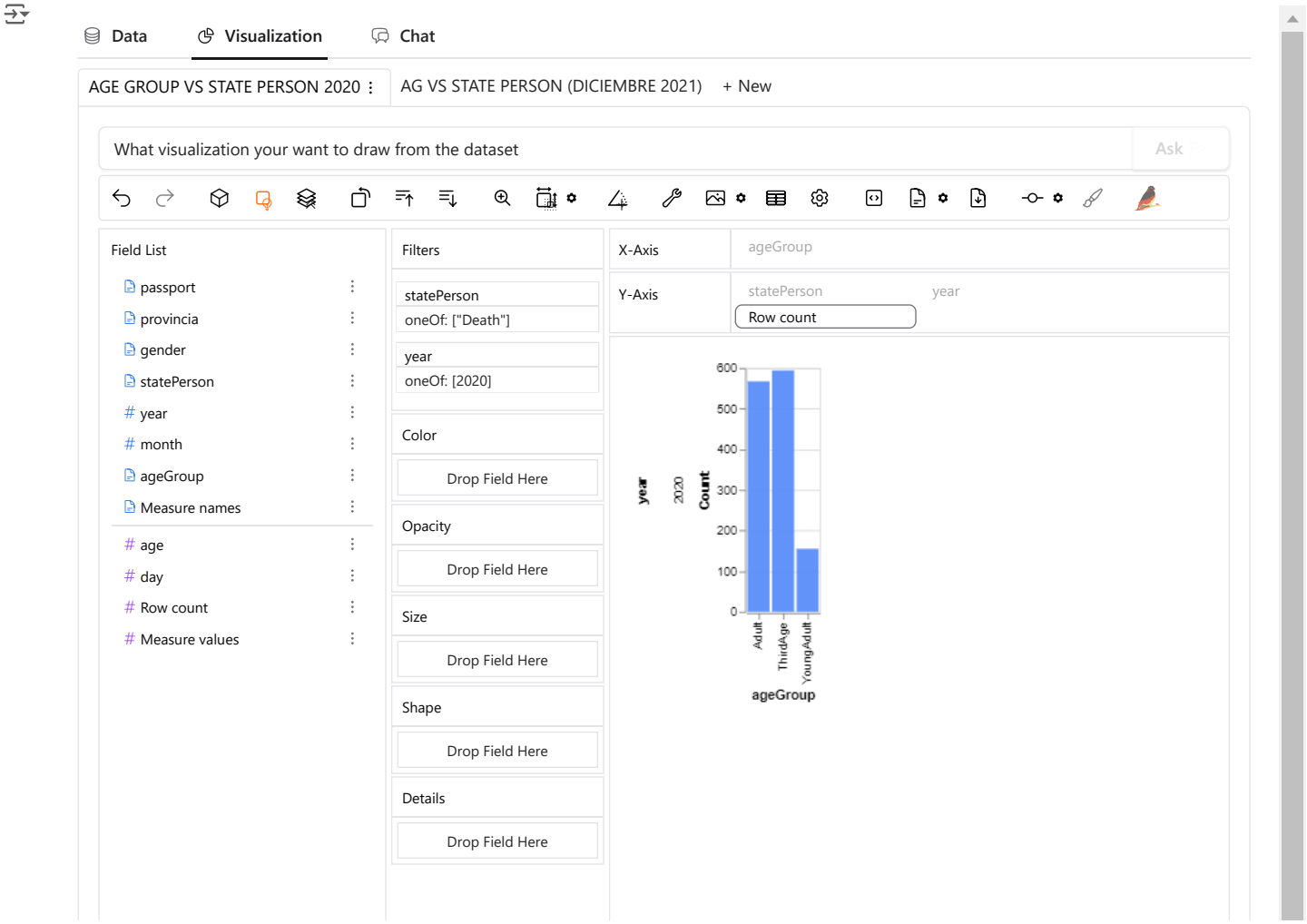


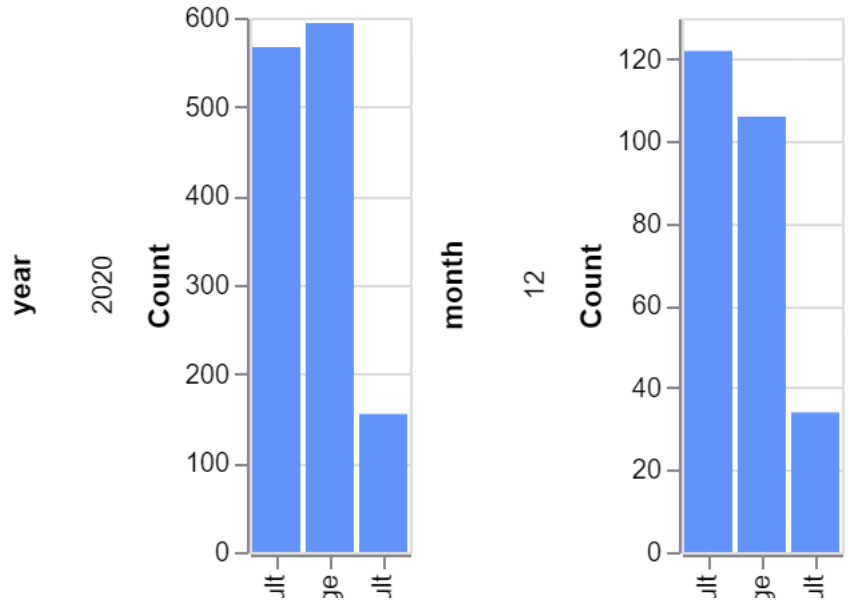
```
# Group by year and statePerson, and count the outcomes
state_counts = dataMerge.groupby(['year', 'statePerson']).size().reset_index(name='count')

# Pivot the data to create separate columns for each statePerson
state_counts_pivot = state_counts.pivot(index='year', columns='statePerson', values='count').fillna(0)

# Create a ColumnDataSource from the pivoted data
source = ColumnDataSource(state_counts_pivot.reset_index())

# Create a Range1d object for the x_range argument
x_range = Range1d(start=min(state_counts_pivot.index), end=max(state_counts_pivot.index))

# Create a figure with a bar chart
p = bp.figure(x_range=x_range,
              width=800, height=400, title='Count of recovered, death, and not infected by year')

p.vbar_stack(['Recovered', 'Death', 'Notinfected'], x='year', width=0.9,
             color=['green', 'red', 'blue'], source=source, legend_label=['Recovered', 'Death', 'Not Infected'])
p.yaxis.axis_label = 'Count'
p.xaxis.axis_label = 'Year'
p.legend.location = 'top_right'

# Output the plot to a Jupyter notebook
output_notebook()

# Show the plot
bp.show(p)
```



Gráficos con Pywalker

```
gwalker = pyg.walk(dataMerge)
```


Data
Visualization
Chat
AGE GROUP VS STATE PERSON 2020
AG VS STATE PERSON (DICIEMBRE 2021)
+ New
What visualization your want to draw from the dataset
Ask
Field List
passport
provincia
gender
statePerson
year
month
ageGroup
Measure names
age
day
Row count
Measure values
Filters
statePerson
oneOf: ["Death"]
year
oneOf: [2020]
Color
Drop Field Here
Opacity
Drop Field Here
Size
Drop Field Here
Shape
Drop Field Here
Details
Drop Field Here
X-Axis
ageGroup
Y-Axis
statePerson
year
Row count
year
2020
Count
ageGroup
Adult
ThirdAge
YoungAdult


year
2020
Count
month
12
Count