To achieve everything in Python, including \*\*data cleaning, chart creation, and animation\*\*, you'll need to use several powerful libraries. Below is a step-by-step guide on how to replicate the Tableau-like experience using Python. The primary libraries you'll use include:

- \*\*Pandas\*\*: For data manipulation and cleaning.

- \*\*Matplotlib\*\* / \*\*Seaborn\*\* / \*\*Plotly\*\*: For chart creation.

- \*\*Plotly\*\* or \*\*Bokeh\*\*: For interactive and animated visualizations.

### Step-by-Step Guide

---

### \*\*1. Data Cleaning with Pandas\*\*

First, import and clean your dataset using `pandas`. Assume your dataset is in CSV format.

#### \*\*Example Code\*\*:

```python

import pandas as pd

# Load dataset

df = pd.read\_csv('disease\_symptoms\_data.csv')

# Inspect the data

print(df.head())

# Data Cleaning (if needed)

# - Check for missing values

print(df.isnull().sum())

# Drop rows or fill missing values

df.dropna(inplace=True) # Drops rows with any missing data

# Alternatively, you can fill missing values if needed:

# df.fillna({'Age': df['Age'].mean(), 'Fever': 'No', ...}, inplace=True)

# Convert categorical columns to appropriate data types

df['Fever'] = df['Fever'].astype('category')

df['Cough'] = df['Cough'].astype('category')

df['Outcome'] = df['Outcome'].astype('category')

# Verify changes

print(df.info())

```

### \*\*2. Basic Visualizations Using Matplotlib/Seaborn\*\*

For simple static charts (like bar charts, heatmaps, etc.), use `Matplotlib` and `Seaborn`.

#### \*\*Bar Chart – Symptom Prevalence by Disease\*\*:

```python

import matplotlib.pyplot as plt

import seaborn as sns

# Symptom prevalence count per disease

symptom\_counts = df.groupby('Disease')[['Fever', 'Cough', 'Fatigue', 'Difficulty Breathing']].apply(lambda x: x.eq('Yes').sum())

# Create a stacked bar chart

symptom\_counts.plot(kind='bar', stacked=True, colormap='coolwarm')

plt.title('Symptom Prevalence by Disease')

plt.ylabel('Number of Patients')

plt.xlabel('Disease')

plt.xticks(rotation=45)

plt.show()

```

#### \*\*Heatmap – Symptom Presence by Disease\*\*:

```python

# Convert Yes/No to binary for heatmap

heatmap\_data = df.replace({'Yes': 1, 'No': 0})

heatmap\_data = heatmap\_data.pivot\_table(index='Disease', values=['Fever', 'Cough', 'Fatigue', 'Difficulty Breathing'], aggfunc='mean')

# Plot heatmap

sns.heatmap(heatmap\_data, annot=True, cmap='coolwarm')

plt.title('Heatmap of Symptom Presence by Disease')

plt.show()

```

---

### \*\*3. Interactive and Animated Charts Using Plotly\*\*

For interactive charts, including animated visualizations, use `Plotly`. Here’s how to create an interactive and animated visualization with `Plotly`.

#### \*\*Animated Bar Chart – Symptom Impact on Outcome\*\*:

```python

import plotly.express as px

# Aggregate data for the animation

df\_grouped = df.groupby(['Age', 'Disease', 'Fever', 'Cough', 'Fatigue', 'Difficulty Breathing', 'Outcome']).size().reset\_index(name='Count')

# Create an animated bar chart with age as the animation frame

fig = px.bar(df\_grouped, x='Disease', y='Count', color='Outcome',

animation\_frame='Age', barmode='group',

title='Symptom Impact on Outcome Across Ages')

fig.show()

```

#### \*\*Animated Line Chart – Outcome Change Over Age\*\*:

```python

# Summarize data

outcome\_by\_age = df.groupby(['Age', 'Outcome']).size().reset\_index(name='Count')

# Create animated line chart

fig = px.line(outcome\_by\_age, x='Age', y='Count', color='Outcome', title='Outcome Change Across Ages',

animation\_frame='Age')

fig.show()

```

---

### \*\*4. Interactive Dashboards with Plotly Dash or Bokeh\*\*

You can create interactive dashboards with `Plotly Dash` or `Bokeh`. Below is an example of creating an interactive dashboard using `Dash`.

#### \*\*Dash Interactive Dashboard\*\*:

1. Install Dash:

```bash

pip install dash

```

2. Create a Python file for your dashboard:

```python

import dash

from dash import dcc, html

import plotly.express as px

import pandas as pd

# Load and prepare data

df = pd.read\_csv('disease\_symptoms\_data.csv')

# Group data for the dashboard

symptom\_data = df.groupby(['Disease', 'Fever', 'Cough', 'Fatigue', 'Difficulty Breathing']).size().reset\_index(name='Count')

# Initialize the Dash app

app = dash.Dash(\_\_name\_\_)

# App layout

app.layout = html.Div(children=[

html.H1(children='Disease Symptom Dashboard'),

dcc.Dropdown(

id='disease-dropdown',

options=[{'label': disease, 'value': disease} for disease in df['Disease'].unique()],

value=df['Disease'].unique()[0]

),

dcc.Graph(id='symptom-bar-chart')

])

# Callback to update bar chart based on selected disease

@app.callback(

dash.Output('symptom-bar-chart', 'figure'),

[dash.Input('disease-dropdown', 'value')]

)

def update\_bar\_chart(selected\_disease):

filtered\_df = symptom\_data[symptom\_data['Disease'] == selected\_disease]

fig = px.bar(filtered\_df, x=['Fever', 'Cough', 'Fatigue', 'Difficulty Breathing'], y='Count', title=f'Symptoms for {selected\_disease}')

return fig

# Run the app

if \_\_name\_\_ == '\_\_main\_\_':

app.run\_server(debug=True)

```

---

### \*\*5. Additional Steps for Data Animation with Bokeh\*\*

If you prefer `Bokeh` for animation, it supports more granular animation control with Python.

1. Install Bokeh:

```bash

pip install bokeh

```

2. Example Code for an animated plot using `Bokeh`:

```python

from bokeh.plotting import figure, output\_file, show

from bokeh.models import ColumnDataSource

from bokeh.io import curdoc

from bokeh.layouts import column

from bokeh.models import Slider

import pandas as pd

# Load data

df = pd.read\_csv('disease\_symptoms\_data.csv')

# Data for a simple animated line plot

ages = df['Age'].unique()

outcomes = df.groupby(['Age', 'Outcome']).size().unstack().fillna(0)

# Plot setup

source = ColumnDataSource(data={'x': ages, 'y': outcomes['Positive']})

plot = figure(title='Positive Outcomes by Age', x\_axis\_label='Age', y\_axis\_label='Positive Outcomes')

plot.line(x='x', y='y', source=source)

# Slider for animation

slider = Slider(start=0, end=len(ages)-1, value=0, step=1, title="Time Step")

# Callback to animate

def update(attr, old, new):

step = slider.value

new\_data = {'x': ages[:step], 'y': outcomes['Positive'][:step]}

source.data = new\_data

slider.on\_change('value', update)

# Layout and run

layout = column(slider, plot)

curdoc().add\_root(layout)

```

---

### \*\*Summary\*\*

This approach enables you to achieve the full workflow—\*\*data cleaning, visualization, and animation\*\*—in Python. You can leverage static libraries like `Matplotlib` and `Seaborn` for simple visualizations and use \*\*Plotly\*\* and \*\*Bokeh\*\* for rich interactive and animated charts. \*\*Dash\*\* is your go-to tool for building a full interactive dashboard.

Let me know if you'd like help with specific parts of this process!