Create Your Own Streamlit Dashboard - Project Guide

This guide will help you create your own interactive data dashboard using Streamlit. Use this as inspiration - not a checklist! The goal is to explore data that interests YOU and build something you're proud to share.

Getting Started

Step 1: Choose Your Dataset

Pick a dataset that excites you! Here are some easily accessible options:

Built-in Seaborn Datasets

These load instantly with `sns.load_dataset('name')`:

```
import seaborn as sns

# Popular options:
df = sns.load_dataset('tips')          # Restaurant tips data
df = sns.load_dataset('penguins')      # Antarctic penguin measureme
df = sns.load_dataset('diamonds')      # Diamond prices and characte
df = sns.load_dataset('planets')       # Exoplanet discoveries
df = sns.load_dataset('flights')       # Airline passenger data over
df = sns.load_dataset('iris')          # Classic flower measurements
df = sns.load_dataset('mpg')           # Car fuel efficiency data
df = sns.load_dataset('taxis')         # NYC taxi trip data
```

Tip: Explore available datasets with `sns.get_dataset_names()`

Course Datasets

Located in the `data/` folder:

- `adult.csv` - Census income data
- `obesity.csv` - Health and lifestyle data
- `supplements.csv` - Dietary supplement usage
- `titanic.csv` - Titanic passenger data (already used in tutorial)
- `worldcities.csv` - Cities around the world

Online Datasets (CSV format)

```
# Example: Load from a URL
df = pd.read_csv('https://raw.githubusercontent.com/datasets/covid-
```

Where to find datasets:

- Kaggle Datasets
- Data.gov
- UCI Machine Learning Repository
- Google Dataset Search
- FiveThirtyEight Data

Step 2: Explore Your Data

Before building your dashboard, understand what you're working with:

```
# Basic exploration
print(df.head())
print(df.info())
print(df.describe())
print(df.columns)
```

Ask yourself:

- What questions could this data answer?
- What columns would make good filters?
- What relationships might be interesting to visualize?
- Are there any missing values to handle?

Dashboard Components - Pick and Choose!

You don't need to include everything below. Choose the components that best tell your data's story.

Interactive Filters (Choose 3-5)

Categorical Filters:

- `st.multiselect()` - When users can pick multiple categories (e.g., car brands, species)
- `st.selectbox()` - Single choice from a dropdown (e.g., year, region)
- `st.radio()` - Single choice with few options (e.g., gender, yes/no)
- `st.checkbox()` - Toggle filters on/off (e.g., "Show only premium items")

Numeric Filters:

- `st.slider()` - Range selection (e.g., price range, age range)
- `st.number_input()` - Precise number entry (e.g., minimum value)

Text Filters:

- `st.text_input()` - Search functionality

Example Ideas:

- Tips dataset: Filter by day, meal time, party size, smoker status
- Penguins: Filter by species, island, sex, body mass range
- Diamonds: Filter by cut quality, color, clarity, price range
- MPG: Filter by year, origin, cylinders, fuel efficiency

Summary Metrics (Choose 3-6)

Use `st.metric()` to display key numbers. Think about what matters most in your data:

Examples by Dataset:

Tips:

- Total bills analyzed
- Average tip percentage
- Highest single bill
- Most common party size

Penguins:

- Total penguins measured
- Average body mass by species
- Bill length range
- Islands represented

Diamonds:

- Total diamonds in dataset
- Average price
- Most common cut quality
- Price per carat

General Ideas:

- Counts (total rows, unique values)
- Averages (mean, median)
- Ranges (min, max, spread)
- Percentages (ratios, proportions)

Visualizations (Choose 3-5)

Mix different chart types to show different insights:

Distribution Charts:

```
# Histogram
df['column'].hist(bins=30, ax=ax)

# Box plot
df.boxplot(column='value', by='category', ax=ax)

# Violin plot
sns.violinplot(data=df, x='category', y='value', ax=ax)
```

Comparison Charts:

```
# Bar chart (averages by category)
df.groupby('category')['value'].mean().plot(kind='bar', ax=ax)

# Grouped bar chart
df.groupby(['cat1', 'cat2'])['value'].mean().unstack().plot(kind='ba

# Side-by-side comparison
sns.barplot(data=df, x='category', y='value', hue='subcategory', ax=
```

Relationship Charts:

```
# Scatter plot
ax.scatter(df['x'], df['y'], alpha=0.5)

# Scatter with color coding
sns.scatterplot(data=df, x='var1', y='var2', hue='category', ax=ax)

# Line chart (time series)
df.groupby('date')['value'].mean().plot(ax=ax)
```

Proportion Charts:

```
# Pie chart
df['category'].value_counts().plot(kind='pie', ax=ax)

# Stacked bar chart
df.groupby(['cat1', 'cat2']).size().unstack().plot(kind='bar', stacl
```

Advanced Visualizations:

```
# Heatmap (correlation matrix)
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', ax=ax)

# Pairplot (multiple relationships) - use with caution, can be slow
sns.pairplot(df, hue='category')

# Count plot
sns.countplot(data=df, x='category', ax=ax)
```

Interactive Charts with Plotly Express:

Plotly Express creates interactive charts that users can zoom, pan, and hover over. These are great for exploration!

```
import plotly.express as px
import streamlit as st

# Interactive scatter plot
fig = px.scatter(df, x='column1', y='column2', color='category',
                 title='Interactive Scatter Plot',
                 hover_data=['additional_info'])
st.plotly_chart(fig, use_container_width=True)

# Interactive line chart
fig = px.line(df, x='date', y='value', color='category',
              title='Time Series Analysis')
st.plotly_chart(fig)

# Interactive bar chart
fig = px.bar(df, x='category', y='value', color='subcategory',
             title='Grouped Bar Chart')
st.plotly_chart(fig)

# Interactive box plot
fig = px.box(df, x='category', y='value', color='category',
             title='Distribution Comparison')
st.plotly_chart(fig)

# Interactive histogram
fig = px.histogram(df, x='value', color='category',
                   title='Distribution by Category')
st.plotly_chart(fig)
```

Why use Plotly Express?

- Built-in interactivity (zoom, pan, hover tooltips)

- Cleaner syntax than matplotlib for many charts
- Automatically handles legends and colors
- Great for presentations and exploration
- Works seamlessly with Streamlit

When to use Matplotlib vs Plotly:

- **Matplotlib/Seaborn**: Static reports, complex customization, statistical plots
- **Plotly Express**: Interactive dashboards, user exploration, modern look

Organization Options

Tabs:

```
tab1, tab2, tab3 = st.tabs(["📊 Overview", "📈 Analysis", "📋 Data

with tab1:
    # Summary metrics and key charts

with tab2:
    # Detailed visualizations

with tab3:
    # Raw data table
```

Expanders:

```
with st.expander("📖 About This Dataset"):
    st.write("Description of your data...")

with st.expander("📊 See Detailed Statistics"):
    st.dataframe(df.describe())
```

Columns:

```
col1, col2 = st.columns(2)

with col1:
    # Left side visualization

with col2:
    # Right side visualization
```

Example Project Ideas

1. Restaurant Tips Analysis

Questions to explore:

- Do larger parties tip better?
- Does tip percentage vary by day of week?
- Is there a difference between lunch and dinner tipping?
- Do smokers tip differently than non-smokers?

Filters: Day, meal time, party size range, smoker status **Metrics:** Average tip %, total bills, largest tip, busiest day **Charts:** Tip % by day (bar), party size vs tip amount (scatter), bill distribution (histogram)

2. Penguin Species Comparison

Questions to explore:

- How do body measurements differ by species?
- Are certain species found on specific islands?
- What's the relationship between bill length and body mass?
- How do male and female penguins compare?

Filters: Species, island, sex, body mass range **Metrics:** Average body mass by species, bill length range, total measured **Charts:** Body mass by species (box plot), bill dimensions (scatter), species by island (stacked bar)

3. Diamond Price Explorer

Questions to explore:

- What factors most affect diamond prices?
- How does carat weight relate to price?
- Which cut quality offers the best value?
- How do color and clarity impact cost?

Filters: Cut quality, color grade, clarity, price range, carat range **Metrics:** Average price, price per carat, most expensive cut, total diamonds **Charts:** Price vs carat (scatter), price by cut (box plot), color distribution (bar)

4. Car Fuel Efficiency Dashboard

Questions to explore:

- How has fuel efficiency changed over time?
- Which origin produces the most efficient cars?
- What's the relationship between cylinders and MPG?
- Are heavier cars less efficient?

Filters: Year range, origin, number of cylinders **Metrics:** Average MPG, most efficient car, total models **Charts:** MPG over time (line), MPG by cylinders (bar), weight vs MPG (scatter)

5. Flight Passenger Trends

Questions to explore:

- How do passenger numbers vary by month?
- Which years saw the most growth?
- Are there seasonal patterns?
- How do trends compare across years?

Filters: Year range, month selection **Metrics:** Total passengers, average monthly passengers, peak month, growth rate **Charts:** Passengers over time (line), monthly patterns (bar), year comparison (grouped bar)

Building Your Dashboard - Step by Step

Phase 1: Basic Setup

1. Create a new `app.py` file
2. Load your chosen dataset
3. Display the first few rows with `st.dataframe()`
4. Add a title and brief description
5. Show basic data info (row count, column count)

Phase 2: Add Filters

1. Identify 3-5 columns that would make good filters
2. Add widgets to the sidebar
3. Create a filtered DataFrame
4. Display the filtered data
5. Add a counter showing filtered vs total rows

Phase 3: Summary Metrics

1. Calculate 3-6 key statistics
2. Display them with `st.metric()` in columns
3. Make sure they update based on filters

Phase 4: Visualizations

1. Choose 3-5 charts that answer different questions

2. Create each visualization with matplotlib/seaborn (or Plotly for interactive charts)
3. Display them with `st.pyplot()` or `st.plotly_chart()`
4. Arrange them using columns or tabs
5. Add titles and axis labels

Phase 5: Polish

1. Organize content with tabs or expanders
2. Add a "About" section describing the data
3. Use `layout="wide"` if helpful
4. Add help text to filters
5. Test all interactions

Pro Tips & Ideas

Make It Personal

- Choose data related to your major or interests
- Add your own insights in markdown sections
- Use colors that match your theme
- Write clear, engaging descriptions

Tell a Story

- Start with a question your dashboard answers
- Guide users through the insights
- Use expanders for "deep dives"
- End with key takeaways

Add Interactivity

- Let users compare scenarios side-by-side
- Add a "reset filters" button
- Include download functionality for filtered data
- Use checkboxes to show/hide chart elements

Bonus Features (Optional)

```
# Download filtered data
csv = filtered_df.to_csv(index=False)
st.download_button("Download Data", csv, "filtered_data.csv", "text/

# Reset button
if st.sidebar.button("🔄 Reset All Filters"):
    st.rerun()

# Color customization
chart_color = st.sidebar.color_picker("Choose Chart Color", "#1f77b4

# Dynamic chart selection
chart_type = st.selectbox("Chart Type:", ["Bar", "Line", "Scatter"])
if chart_type == "Bar":
    df.plot(kind='bar', ax=ax)
# ... etc

# Data export options
format_choice = st.radio("Export Format:", ["CSV", "JSON", "Excel"])
```

Common Pitfalls to Avoid

- **Too many filters**: 3-5 is usually enough
- **Cluttered layout**: Use tabs/expanders to organize
- **Unclear labels**: Always label axes and provide descriptions
- **Too much data at once**: Start filtered or paginated
- **No context**: Explain what the data represents

Dataset-Specific Quick Starts

Tips Dataset

```python
import streamlit as st
import seaborn as sns
import matplotlib.pyplot as plt

st.title("Restaurant Tips Analysis")
df = sns.load_dataset('tips')

# Suggested filters
day = st.sidebar.multiselect("Day:", df['day'].unique(), default=li
time = st.sidebar.selectbox("Meal:", ["All", "Lunch", "Dinner"])
min_size = st.sidebar.slider("Min Party Size:", 1, 6, 1)

# Suggested metrics
avg_tip_pct = (df['tip'] / df['total_bill'] * 100).mean()
st.metric("Average Tip %", f"{avg_tip_pct:.1f}%")
```

Penguins Dataset

```python
import streamlit as st
import seaborn as sns
import matplotlib.pyplot as plt

st.title("Palmer Penguins Explorer")
df = sns.load_dataset('penguins').dropna()

# Suggested filters
species = st.sidebar.multiselect("Species:", df['species'].unique()
island = st.sidebar.multiselect("Island:", df['island'].unique(), d

# Suggested metrics
avg_mass = df['body_mass_g'].mean()
st.metric("Average Body Mass", f"{avg_mass:.0f}g")
```

Diamonds Dataset

```python
import streamlit as st
import seaborn as sns
import matplotlib.pyplot as plt

st.title("Diamond Price Explorer")
df = sns.load_dataset('diamonds')

# Suggested filters
cut = st.sidebar.multiselect("Cut:", df['cut'].unique(), default=li
price_range = st.sidebar.slider("Price Range:", 0, int(df['price'].

# Suggested metrics
avg_price = df['price'].mean()
st.metric("Average Price", f"${avg_price:.0f}")
```

Suggested Checklist

When building your dashboard, consider including:

- [ ] A clear, descriptive title
- [ ] Brief introduction explaining what the data shows
- [ ] At least 3 interactive filters
- [ ] At least 3 summary metrics that update with filters
- [ ] At least 3 different types of visualizations
- [ ] Clear axis labels and chart titles
- [ ] Organized layout (tabs, columns, or expanders)
- [ ] An "About" section describing the dataset
- [ ] Tested all filters and interactions
- [ ] Added comments explaining complex code

Getting Help

Stuck on something?

1. Check the Streamlit documentation
2. Review the tutorial examples in `titanic-streamlit.md`
3. Look at the example `app.py` in the `src/` folder
4. Search for examples in the Streamlit Gallery
5. Ask classmates or the instructor

Debugging Tips:

- Use `st.write()` to print variables and check values
- Check the terminal for error messages

- Try commenting out sections to isolate problems
- Make sure column names match your dataset exactly

Inspiration Gallery

Things other students have built:

- "Which NBA team should I root for?" - Interactive team comparison
- "Coffee Shop Finder" - Location-based recommendations
- "Movie Rating Predictor" - Analyze what makes movies popular
- "Weather Pattern Explorer" - Climate data visualizations
- "Workout Tracker" - Personal fitness analytics
- "Recipe Nutrition Calculator" - Health-focused food data

The best projects:

- Answer a question you actually care about
- Show insights that surprise you
- Are easy to navigate and understand
- Look polished and professional

Ready to Start?

1. **Pick your dataset** (spend 5-10 minutes browsing options)
2. **Load and explore it** (understand what you have)
3. **Ask 3 questions** your dashboard should answer
4. **Sketch your layout** (what goes where?)
5. **Start coding!** (Begin with Phase 1 above)

Remember: Your dashboard doesn't need to be perfect - it needs to be interesting to YOU. Have fun exploring the data!

Additional Resources

- **Streamlit Documentation**: https://docs.streamlit.io
- **Streamlit Cheat Sheet**: https://docs.streamlit.io/library/cheatsheet
- **Seaborn Gallery**: https://seaborn.pydata.org/examples/index.html
- **Matplotlib Tutorials**: https://matplotlib.org/stable/tutorials/index.html
- **Pandas Documentation**: https://pandas.pydata.org/docs/

Good luck, and happy dashboard building! 🚀