Python and SQL: intro / SQL platforms

Ewa Weychert

Class 9: Streamlit

UNIWERSYTET WARSZAWSKI | WYDZIAŁ NAUK EKONOMICZNYCH

What is Streamlit?

Streamlit is an open-source Python framework for building interactive web apps easily and quickly.

- **Simple definition** Write Python → Streamlit automatically creates the user interface.
- **Why it is useful**
    - No need to learn HTML, CSS, or JavaScript.
    - Ideal for data science, ML demos, dashboards, and prototypes.
    - Very fast to develop — update code, app refreshes instantly.
    - Easily share apps with others through a browser.

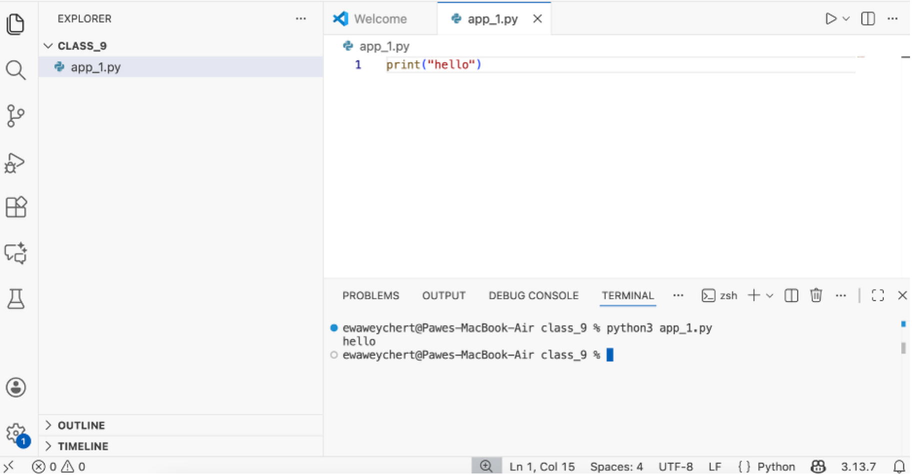
Running python .py files in terminal

1. Open Visual Studio Code.
2. Create a new file called `app_1.py`.
3. Make sure everything is saved in one folder.
4. In this file, paste and save the following code:

```
print("Hello from Python!")
```

5. In the upper-right corner of Visual Studio Code, click on "Toggle Panel."
6. The terminal will appear at the bottom of the screen.
7. In the terminal, type: `python app_1.py` (or `python3 app_1.py` on some systems).

Running python file in console



```python
print("hello")
```

```
ewaweychert@Pawes-MacBook-Air class_9 % python3 app_1.py
hello
ewaweychert@Pawes-MacBook-Air class_9 %
```

Install Streamlit (CMD / Terminal)

Installation commands:

- `pip install streamlit`
- `pip3 install streamlit`

Check that it works (it will show the help of streamlit) In CMD / Terminal wirte this:

- `streamlit hello`

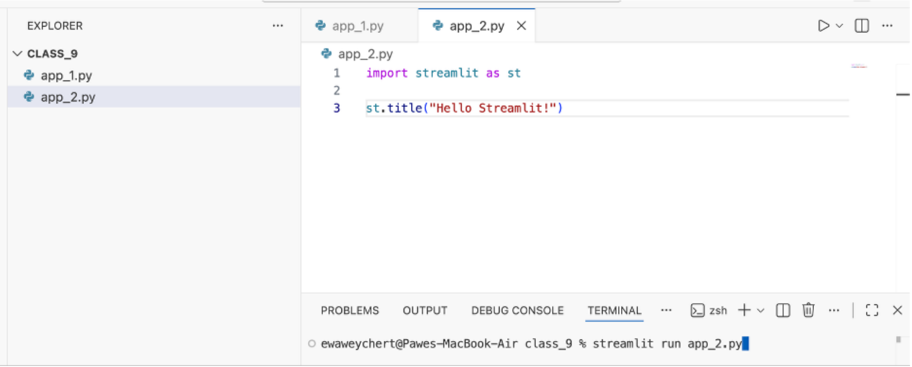
First app in streamlit

1. Open Visual Studio Code.
2. Create a new file called `app_2.py`.
3. Make sure everything is in one folder.
4. In this file, paste and save the following code:

```
import streamlit as st
st.title("Hello Streamlit!")
```

5. In the upper-right corner of Visual Studio Code, click on “Toggle Panel.”
6. The screen will show the terminal.
7. In the terminal, type: `streamlit run app_2.py`

First app in streamlit



```python
import streamlit as st

st.title("Hello Streamlit!")
```

```
ewaweychert@Pawes-MacBook-Air class_9 % streamlit run app_2.py
```

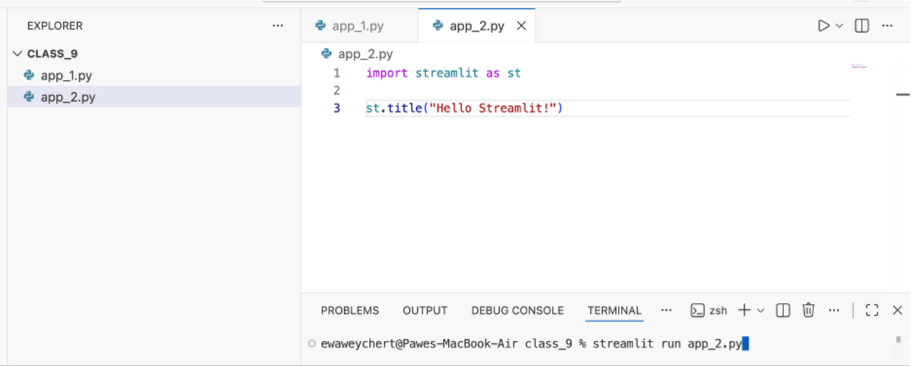
Second app in streamlit

1. Open Visual Studio Code.
2. Create a new file called `app_3.py`.
3. Make sure everything is in one folder.
4. In this file, paste and save the following code:

```
import streamlit as st
st.title("Hello Streamlit!")
st.write("This is my first Streamlit app.")
number = st.slider("Pick a number", 1, 10)
st.write("Your number squared:", number**2)
```

5. In the upper-right corner of Visual Studio Code, click on "Toggle Panel."
6. The screen will show the terminal.
7. In the terminal, type: `streamlit run app_3.py`

First app in streamlit



app_2.py

```python
import streamlit as st

st.title("Hello Streamlit!")
```

```
ewaweychert@Pawes-MacBook-Air class_9 % streamlit run app_2.py
```

How to stop app in terminal

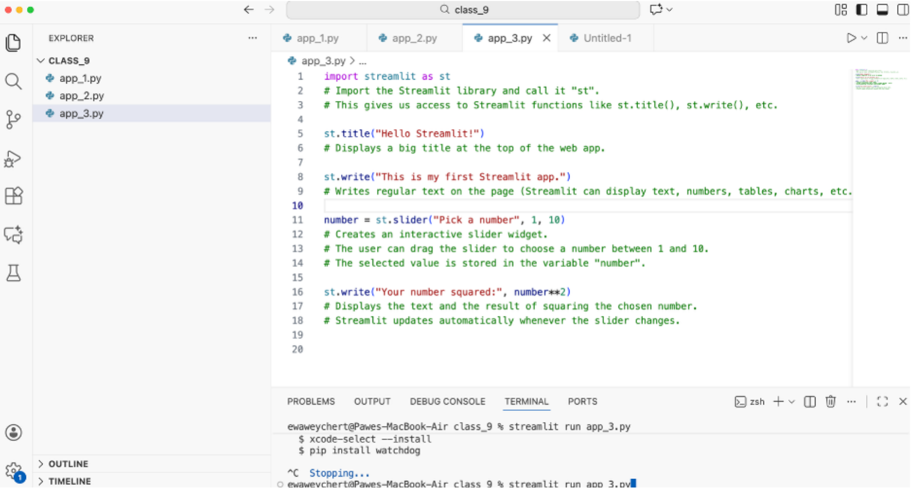
click control C

Second app in streamlit

- `import streamlit as st`
  Imports the Streamlit library and gives it the short name `st`. This is the standard convention in almost all Streamlit examples.
- `st.title("Hello Streamlit!")`
  Displays a large page title at the top of the app. It helps introduce what the app is about.
- `st.write("This is my first Streamlit app.")`
  `st.write()` is Streamlit's most flexible output function. It can display text, numbers, DataFrames, plots, and more. Here, it simply prints a line of text under the title.
- `number = st.slider("Pick a number", 1, 10)`
  Creates an interactive slider widget that lets the user choose a number between 1 and 10. Whatever the user selects gets stored in the variable `number`.
- `st.write("Your number squared:", number**2)`
  Shows the result of squaring the selected number. This demonstrates Streamlit's reactive behavior: every time the user moves the slider, the output updates automatically.

Overall:
This script shows how to add text, create a widget, capture input, perform a small calculation, and display output — all in just a few lines.

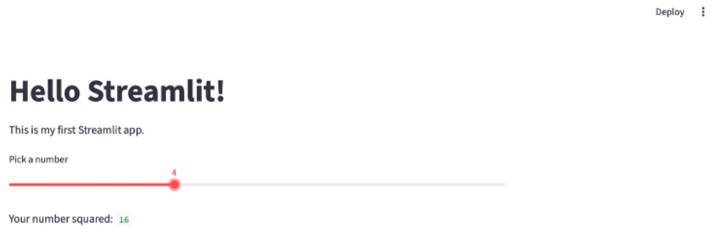
Second app in streamlit



```python
import streamlit as st
# Import the Streamlit library and call it "st".
# This gives us access to Streamlit functions like st.title(), st.write(), etc.

st.title("Hello Streamlit!")
# Displays a big title at the top of the web app.

st.write("This is my first Streamlit app.")
# Writes regular text on the page (Streamlit can display text, numbers, tables, charts, etc.

number = st.slider("Pick a number", 1, 10)
# Creates an interactive slider widget.
# The user can drag the slider to choose a number between 1 and 10.
# The selected value is stored in the variable "number".

st.write("Your number squared:", number**2)
# Displays the text and the result of squaring the chosen number.
# Streamlit updates automatically whenever the slider changes.
```

```
ewaweychert@Pawes-MacBook-Air class_9 % streamlit run app_3.py
  $ xcode-select --install
  $ pip install watchdog

^C  Stopping...
○ ewaweychert@Pawes-MacBook-Air class_9 % streamlit run app_3.py
```

Second app in streamlit



Hello Streamlit!

This is my first Streamlit app.

Pick a number

4

Your number squared: 16

What are widgets?

- In Streamlit, widgets are interactive GUI (Graphical User Interface) elements.
- They let the user input data or make choices, e.g.:
    - buttons
    - sliders
    - text inputs
    - checkboxes
    - select boxes (drop-downs)
- When the user interacts with a widget, Streamlit **reruns the script** and updates the output automatically.
- Widgets are the main way to make your app **interactive**.

Simple widget examples in Streamlit

Example 1: Text input

```
import streamlit as st

name = st.text_input("What is your name?")
st.write("Hello,", name)
```

Example 2: Slider

```
number = st.slider("Pick a number", 1, 10)
st.write("Your number squared is", number**2)
```

Example 3: Checkbox

```
show_text = st.checkbox("Show message")
if show_text:
    st.write("The checkbox is checked!")
```

Third app in streamlit

1. Open Visual Studio Code.
2. Create a new file called `app_4.py`.
3. Make sure everything is in one folder.
4. In this file, paste and save the following code:

   ```
   put code from next three slides
   ```

5. In the upper-right corner of Visual Studio Code, click on “Toggle Panel.”
6. The screen will show the terminal.
7. In the terminal, type: `streamlit run app_4.py`

Third app in streamlit - widget examples app$_4$.*py*

```
import streamlit as st

st.title("Hello Streamlit!")
st.write("This is my first Streamlit app.")

# Slider
number = st.slider("Pick a number", 1, 10)
st.write("Your number squared:", number**2)

# Text input
name = st.text_input("Enter your name:")
st.write("Hello,", name)

# Checkbox
show_message = st.checkbox("Show a secret message")
if show_message:
    st.write("You checked the box!")
```

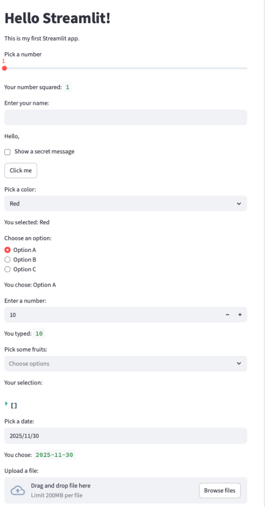
Third app in streamlit - widget examples app$_4$.*py*

```
# Button
if st.button("Click me"):
    st.write("Button clicked!")

# Selectbox
color = st.selectbox("Pick a color:", ["Red", "Green", "Blue"])
st.write("You selected:", color)

# Radio buttons
choice = st.radio("Choose an option:", ["Option A", "Option B", "Option C"])
st.write("You chose:", choice)

# Number input
num = st.number_input("Enter a number:", min_value=0, max_value=100, value=10)
st.write("You typed:", num)

# Multiselect
fruits = st.multiselect("Pick some fruits:",
                        ["Apple", "Banana", "Orange", "Strawberry"])
st.write("Your selection:", fruits)
```

Third app in streamlit - widget examples app$_4$.*py*

```
# Date input
date = st.date_input("Pick a date:")
st.write("You chose:", date)

# File uploader
uploaded = st.file_uploader("Upload a file:")
if uploaded is not None:
    st.write("File uploaded:", uploaded.name)
```

Third app in streamlit



Hello Streamlit!

This is my first Streamlit app.

Pick a number

1

Your number squared: `1`

Enter your name:

Hello,

□ Show a secret message

Pick a color:

Red

You selected: Red

Choose an option:

● Option A
○ Option B
○ Option C

You chose: Option A

Enter a number:

10

You typed: `10`

Pick some fruits:

Choose options

Your selection:

`[]`

Pick a date:

2025/11/30

You chose: `2025-11-30`

Upload a file:

Data Loading & EDA App - app_5.py

- The app is built using:
  - `pandas` for loading and manipulating data,
  - `streamlit` for creating the web interface,
  - `matplotlib` and `seaborn` for visualizations.
- The main title `st.title("Data Loading & EDA App")` appears at the top of the page.
- A **file uploader** widget allows the user to upload a CSV file:
  - the uploaded file is stored in the variable `uploaded`,
  - if a file is provided, it is read into a DataFrame using `pd.read_csv`.
- A preview of the dataset (first 5 rows) is displayed to give a quick look at the data.

Helper Functions for EDA - app_5.py

- **Basic info function**
  - Displays a subheader *"Basic Info"*.
  - Shows the shape of the DataFrame (number of rows and columns).
  - Lists the data types of each column.
  - Prints basic descriptive statistics for numeric variables.
- **Missing values function**
  - Displays a subheader *"Missing Values"*.
  - For each column, shows how many missing values it contains.
- **Fill-missing-with-mean function**
  - Computes the mean of all numeric columns.
  - Returns a version of the DataFrame where numeric `NaN` values are replaced by the corresponding column mean.
- **Categorical distributions function**
  - Identifies all categorical columns (of type `object`).
  - For each categorical column, prints the value counts.
  - Uses a separator line to make the output easier to read.

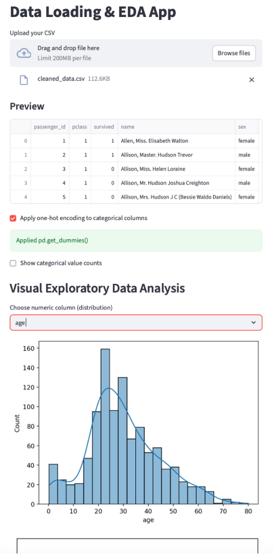
Interactive Options in the App - app_5.py

- After loading the data, the user can choose additional actions through checkboxes:
  - **One-hot encoding**
    - If the user ticks the checkbox, all categorical columns are converted into dummy variables using one-hot encoding.
    - A success message confirms that `pd.get_dummies()` was applied.
  - **Show categorical value counts**
    - If selected, the app calls the function that prints value counts for all categorical variables.
- This makes the app flexible: the user decides whether to transform the data and whether to inspect categorical distributions.

Visual Exploratory Data Analysis - app_5.py

- A separate section *"Visual Exploratory Data Analysis"* is dedicated to plots.
- The app first extracts all numeric columns:
  - if there are no numeric columns, a warning is shown and no plots are produced.
- If numeric data is available, the user can:
  - choose a numeric variable for **distribution analysis**:
    - a histogram with a density curve (KDE) is created,
    - a boxplot for the same variable is also displayed.
  - view a **correlation matrix**:
    - the correlation coefficients between numeric variables are computed,
    - displayed as a heatmap with a color scale.
  - create a **scatterplot** between two numeric variables:
    - the user selects an `X`-axis variable and a `Y`-axis variable,
    - a scatterplot is drawn to show their relationship.
- All plots are rendered using `seaborn` and displayed inside Streamlit with `st.pyplot`.

Data Loading & EDA - app_5.py



Data Loading & EDA App

Upload your CSV

Drag and drop file here
Limit 200MB per file

cleaned_data.csv 112.6KB

Preview

	passenger_id	pclass	survived	name	sex
0	1	1	1	Allen, Miss. Elisabeth Walton	female
1	2	1	1	Allison, Master. Hudson Trevor	male
2	3	1	0	Allison, Miss. Helen Loraine	female
3	4	1	0	Allison, Mr. Hudson Joshua Creighton	male
4	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female

☑ Apply one-hot encoding to categorical columns

Applied pd.get_dummies()

□ Show categorical value counts

Visual Exploratory Data Analysis

Choose numeric column (distribution)

age

Simple EDA App - app_6.py

- This example demonstrates a very small Streamlit application for **basic exploratory data analysis**.
- The app uses two libraries:
  - `pandas` for loading and inspecting tabular data,
  - `streamlit` for building the interactive interface.
- The user uploads a CSV file, which is then read into a `pandas` DataFrame.
- Two main analyses are performed:
  1. displaying general dataset information,
  2. showing distributions of categorical variables.

Function: Basic Info - app_6.py

- The first helper function is responsible for printing simple structural information about the dataset.
- It displays a section header titled *"Basic Info"*.
- The function shows:
  - the **shape** of the dataset (i.e., number of rows and columns),
  - the **data types** of each column (numeric, string, etc.).
- This step helps users quickly understand how large the dataset is and what kinds of variables it contains.

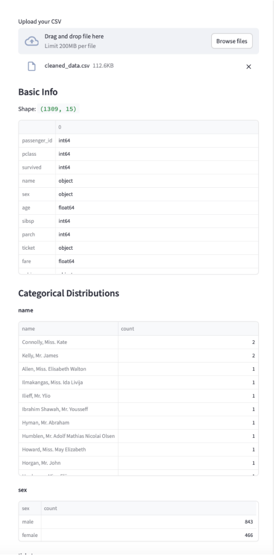
Function: Categorical Distributions - app_6.py

- The second helper function provides a simple summary of all **categorical variables**.
- It identifies columns with type `object`, which typically represent categories or text.
- For each categorical column:
  - the column name is printed,
  - the frequency (count) of each category is shown.
- This is implemented using a **loop** that processes all categorical columns automatically.
- It provides a quick overview of the distribution of categories—useful for EDA and data cleaning.

Data Upload and Execution Flow - app_6.py

- The app includes a **file uploader** widget that allows the user to upload a CSV file.
- If a file is uploaded:
  1. the file is read into a DataFrame,
  2. the **basic info** function is executed,
  3. the **categorical distributions** function is executed.
- This makes the app reactive:
  - as soon as the user provides a dataset, the EDA summaries appear automatically.
- This simple structure demonstrates the essential pattern of Streamlit apps: **upload** → **read** → **analyze** → **display**.

Data Loading & EDA - app_6.py



Upload your CSV

cleaned_data.csv 112.6KB

Basic Info

Shape: `(1309, 15)`

	0
passenger_id	int64
pclass	int64
survived	int64
name	object
sex	object
age	float64
sibsp	int64
parch	int64
ticket	object
fare	float64

Categorical Distributions

name

name	count
Connolly, Miss. Kate	2
Kelly, Mr. James	2
Allen, Miss. Elisabeth Walton	1
Ilmakangas, Miss. Ida Livija	1
Ilieff, Mr. Ylio	1
Ibrahim Shawah, Mr. Yousseff	1
Hyman, Mr. Abraham	1
Humblen, Mr. Adolf Mathias Nicolai Olsen	1
Howard, Miss. May Elizabeth	1
Horgan, Mr. John	1

sex

sex	count
male	843
female	466

Organizing Your App with the Sidebar

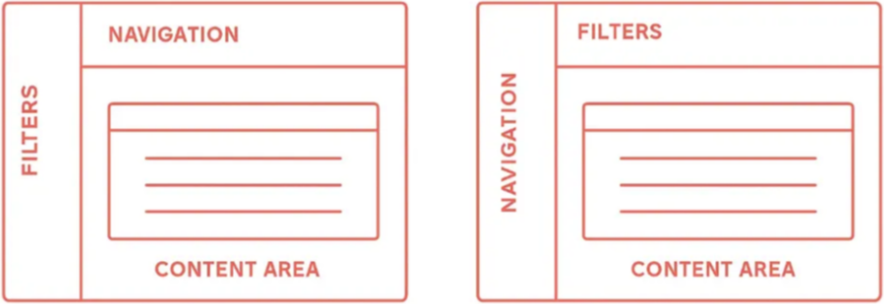
Role of the Sidebar

- Helps organize dashboard content.
- Improves user experience and structure.
- Often includes branding (logo, app name) and navigation or filters.

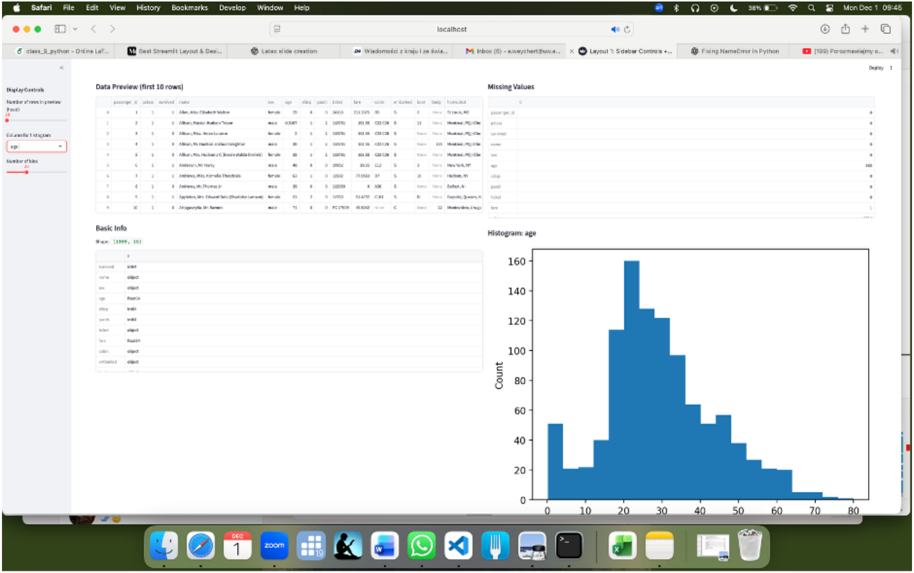
Common Layout Options

1. **Top Navigation + Sidebar Filters**
   - Navigation menu appears as a top bar.
   - Filters and controls are placed in the sidebar.
   - Best for dashboards with many filters but few sections (five or fewer).
2. **Sidebar Navigation + Top Filters**
   - Navigation menu is placed in the sidebar.
   - Filters appear on the top bar.
   - Ideal for dashboards with multiple pages/sections and fewer filters.
   - More commonly used in dashboard design.

Comparing Common Dashboard Layouts using Sidebars



One tab example - app_7.py



Display Controls

Number of rows in preview (head)

10

Column for histogram

age

Number of bins

20

Data Preview (first 10 rows)

	passenger_id	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2	None	St Louis, MO
1	2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	S	11	None	Montreal, PQ / Che
2	3	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
3	4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	S	None	135	Montreal, PQ / Che
4	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
5	6	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12	S	3	None	New York, NY
6	7	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10	None	Hudson, NY
7	8	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36	S	None	None	Belfast, NI
8	9	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	D	None	Bayside, Queens, N
9	10	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	None	C	None	22	Montevideo, Urugu

Missing Values

	0
passenger_id	0
pclass	0
survived	0
name	0
sex	0
age	263
sibsp	0
parch	0
ticket	0
fare	1

Basic Info

Shape: `(1309, 15)`

	0
survived	int64
name	object
sex	object
age	float64
sibsp	int64
parch	int64
ticket	object
fare	float64
cabin	object
embarked	object

High-Level Overview - app_7.py

- The script builds an interactive **data explorer** using `Streamlit`.
- User uploads a CSV file via `st.file`$_uploader$.

- A sidebar controls:
  - number of rows shown in the preview,
  - number of bins for histograms,
  - number of top categories for bar plots.
- The main area is organized into four tabs:
  1. Overview
  2. Missing & Types
  3. Continuous
  4. Categorical

Page Setup and File Upload - app_7.py

- `st.set`$_{p}age_{c}onfigsets:$
  - page title (`"Data Explorer"`),
  - layout (`"wide"`).

The main title and caption are created with:

- `st.title("Data Explorer")`,
- `st.caption(...)` to briefly describe the app.

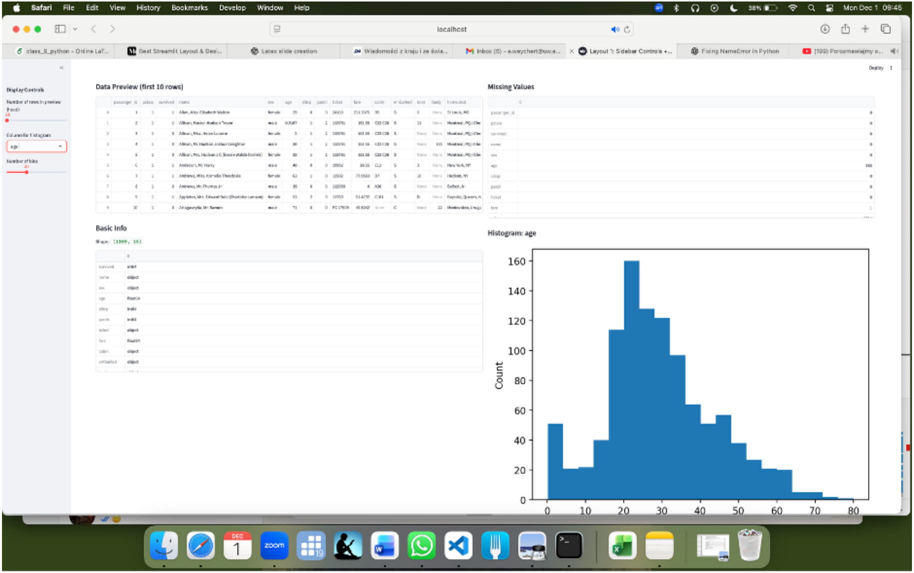
The user provides data through:

- `uploaded = st.file`$_{u}ploader(”UploadyourCSVfile”)$.

When a file is uploaded, it is read into a `pandas` DataFrame:

- `df = pd.read`$_{c}sv(uploaded)$.

One tab example - app_7.py


Safari File Edit View History Bookmarks Develop Window Help
38%
Mon Dec 1 09:46
localhost
class_9_python - Online LaT...
Best Streamlit Layout & Des...
Latex slide creation
Wiadomości z kraju i ze świa...
Inbox (6) - ewwychert@uw.e...
Layout ↑ Sidebar Controls +...
Fixing NameError in Python
(199) Porozmawiajmy o...
Deploy
Count
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
60
70
80

Display Controls

Number of rows in preview (head)

10

Column for histogram

age

Number of bins

20

Data Preview (first 10 rows)

	passenger_id	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2	None	St Louis, MO
1	2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	S	11	None	Montreal, PQ / Che
2	3	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
3	4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	S	None	135	Montreal, PQ / Che
4	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
5	6	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12	S	3	None	New York, NY
6	7	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10	None	Hudson, NY
7	8	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36	S	None	None	Belfast, NI
8	9	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	D	None	Bayside, Queens, N
9	10	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	None	C	None	22	Montevideo, Urugu

Basic Info

Shape: `(1309, 15)`

	0
survived	int64
name	object
sex	object
age	float64
sibsp	int64
parch	int64
ticket	object
fare	float64
cabin	object
embarked	object

Missing Values

	0
passenger_id	0
pclass	0
survived	0
name	0
sex	0
age	263
sibsp	0
parch	0
ticket	0
fare	1

Histogram: age

Sidebar Controls - app_7.py

- The sidebar is accessed via `st.sidebar`.
- Three main controls:
  1. **Rows in preview**:
     - Slider controlling how many rows are shown in `df.head()`.
  2. **Bins for continuous histograms**:
     - Slider controlling the `bins` parameter in `matplotlib` histograms.
  3. **Top N categories**:
     - Slider deciding how many most frequent categories to show in bar plots.
- These values are passed into helper functions to keep logic organized.

Tabs and Helper Functions - app_7.py

- The app uses **tabs** for structure:
  - `tab_overview, tab_missing, tab_continuous, tab_categorical = st.tabs([...])`
- Each tab calls a separate helper function:
  - `panel_overview(df, n_head)`
  - `panel_missing_and_types(df)`
  - `panel_continuous(df, bins)`
  - `panel_categorical(df, top_n)`
- This keeps the main script clean and makes each panel easier to maintain.

Overview Tab: 4 Sections - app_7.py

- Uses `st.columns(2)` twice to create a 2x2 grid.
- Four sections:
  1. **Shape & preview**
     - Shows number of rows/columns and `df.head(n_head)`.
  2. **Column names**
     - Displays a list of all column names.
  3. **Numeric summary**
     - `df.select_dtypes(include="number").describe()`.
  4. **Categorical summary**
     - `df.select_dtypes(exclude="number").describe(include="all")`.

Missing & Types Tab: 4 Sections - app_7.py

- Also split into 4 sections using two rows of two columns:
  1. **Missing counts** per column:

     $$\text{missing\_count} = df.isna().sum()$$

  2. **Missing percentages** per column:

     $$\text{missing\_ } = df.isna().mean() \times 100$$

  3. **Data types** table:

     $$\text{dtypes} = df.dtypes$$

  4. **Unique value counts**:

     $$\text{n\_unique} = df.nunique()$$

- Each result is shown as a small DataFrame.

Continuous Tab: Histograms - app_7.py

- Selects all numeric columns:

$$\text{num\_cols} = df.select\_dtypes(\text{include} = "number").columns$$

- For each numeric column:
  - Drop missing values: `df[col].dropna()`.
  - Plot histogram with `matplotlib`:
    - `fig, ax = plt.subplots()`
    - `ax.hist(data, bins=bins)`
    - `st.pyplot(fig)` to display in Streamlit.
- Plots are arranged in two Streamlit columns to form a grid.

Categorical Tab: Bar Plots - app_7.py

- Selects all non-numeric (categorical) columns:

$$\text{cat\_cols} = df.select\_dtypes(\text{exclude} = "number").columns$$

- For each categorical column:
  - Compute frequency counts:

$$\text{vc} = df[col].value\_counts().head(\text{top\_n})$$

  - Plot bar chart with `matplotlib`:
    - `ax.bar(vc.index.astype(str), vc.values)`
    - Rotate x-axis labels for readability.
    - Show with `st.pyplot(fig)`.
- Again, two Streamlit columns are used to create a grid of plots.

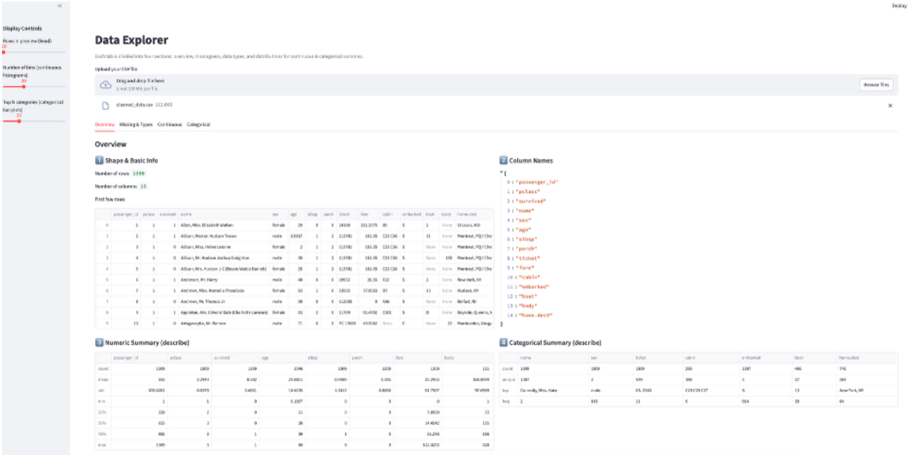
Key Ideas - app_7.py

- **Separation of concerns**:
  - Main script handles layout and routing.
  - Helper functions handle the content of each tab.
- **Interactive controls** in the sidebar influence multiple visualizations.
- **Consistent 4-section layout** in each tab:
  - Uses `st.columns(2)` to build a 2x2 grid.
- **Matplotlib** is used for all plots, embedded in Streamlit with `st.pyplot`.

How app_7 and app_8 Are Different

- **Layout and Navigation**
  - app_7: one **single page** with a 2×2 grid (preview, basic info, missing values, one histogram).
  - app_8: uses **multiple tabs** (Overview, Missing & Types, Continuous, Categorical), each tab internally split into 4 sections.
- **Scope of EDA**
  - app_7: basic dashboard focused on a quick overview and **one selected numeric column** for a histogram.
  - app_8: more complete **data explorer**: summaries for numeric and categorical variables, missing-data tables, dtypes, unique counts, and many plots.
- **Visualizations**
  - app_7: single histogram chosen in the sidebar.
  - app_8: histograms for **all** numeric columns and bar plots for **all** categorical columns (arranged in 2-column grids).
- **Use Case**
  - app_7: simple template for showing *one-page* 4-panel layout.
  - app_8: template for a *full EDA app* with structured navigation and richer analysis.

Multiple tabs example - app_8.py



Display Controls

Rows in preview (head): 10

Number of bins (continuous histograms): 20

Top N categories (categorical bar plots): 10

Data Explorer

Each tab is divided into four sections: overview, missingness, data types, and distributions for continuous & categorical variables.

Upload your CSV file

cleaned_data.csv 112.0KB

Overview

1️⃣ Shape & Basic Info

Number of rows: 1309

Number of columns: 15

First few rows

	passenger_id	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	1	1	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.3375	B5	S	2	None	St Louis, MO
1	2	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.55	C22 C26	S	11	None	Montreal, PQ / Che
2	3	1	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
3	4	1	0	Allison, Mr. Hudson Joshua Creighton	male	30	1	2	113781	151.55	C22 C26	S	None	135	Montreal, PQ / Che
4	5	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25	1	2	113781	151.55	C22 C26	S	None	None	Montreal, PQ / Che
5	6	1	1	Anderson, Mr. Harry	male	48	0	0	19952	26.55	E12	S	3	None	New York, NY
6	7	1	1	Andrews, Miss. Kornelia Theodosia	female	63	1	0	13502	77.9583	D7	S	10	None	Hudson, NY
7	8	1	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36	S	None	None	Belfast, NI
8	9	1	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53	2	0	11769	51.4792	C101	S	D	None	Bayside, Queens, N
9	10	1	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042	None	C	None	22	Montevideo, Urug

2️⃣ Column Names

```
[
  0 : "passenger_id"
  1 : "pclass"
  2 : "survived"
  3 : "name"
  4 : "sex"
  5 : "age"
  6 : "sibsp"
  7 : "parch"
  8 : "ticket"
  9 : "fare"
  10 : "cabin"
  11 : "embarked"
  12 : "boat"
  13 : "body"
  14 : "home.dest"
]
```

3️⃣ Numeric Summary (describe)

	passenger_id	pclass	survived	age	sibsp	parch	fare	body
count	1309	1309	1309	1046	1309	1309	1308	121
mean	655	2.2949	0.382	29.8811	0.4989	0.385	33.2955	160.8099
std	378.0201	0.8378	0.4861	14.4135	1.0417	0.8656	51.7587	97.6969
min	1	1	0	0.1667	0	0	0	1
25%	328	2	0	21	0	0	7.8958	72
50%	655	3	0	28	0	0	14.4542	155
75%	982	3	1	39	1	0	31.275	256
max	1309	3	1	80	8	9	512.3292	328

4️⃣ Categorical Summary (describe)

	name	sex	ticket	cabin	embarked	boat	home.dest
count	1309	1309	1309	295	1307	486	745
unique	1307	2	929	186	3	27	369
top	Connolly, Miss. Kate	male	CA. 2343	C23 C25 C27	S	13	New York, NY
freq	2	843	11	6	914	39	64

Your Own Project

- After this class, you should be able to:
    - Run Python and Streamlit apps from VS Code / terminal.
    - Build simple interactive apps with widgets (sliders, inputs, buttons).
    - Load CSV files in Streamlit and perform basic EDA.
    - Use the sidebar, tabs, and columns to organize a dashboard.
    - Create tables and plots (histograms, bar charts) directly in the app.
- All of this will be needed in your **individual project**.
- Think of the example apps (app_2,app_3, app_4, app_5, app_6, app_7, app_8) as a **toolbox** you can copy, adapt and extend.

Project Requirements (Individual Project)

- **Work mode:** individual, done at home.
- **Topic & data:**
  - You choose the topic and dataset.
  - Topic must be consulted and accepted by instructors.
  - Dataset can be from open sources, APIs, or your own collection.
- **Form of the project:**
  - A working application in **Streamlit** or **Django**.
  - Must include:
    - full implementation (code),
    - results of analyses,
    - clear explanations and comments (interface / docs / report).

Content Requirements: Goal, Questions, Hypotheses

- Your project must have a clear:
  - **Project goal** – what you want to understand or show.
  - **Research questions** – specific questions you want to answer.
  - **Research hypotheses** – expected relationships/effects.
- Example (apartment prices):
  - Goal: identify factors associated with apartment prices in Warsaw.
  - Hypothesis: price per m^2 is higher in central districts.
  - Hypothesis: newer buildings have higher price per m^2.
- Your topic can be from any domain (economics, business, social data, etc.), but it should be **relevant** and **non-trivial**.

What Your App Should Do (EDA Steps)

- **Data loading and cleaning**
  - Load dataset and choose relevant variables.
  - Check missing data and duplicates; handle them (impute or remove).
  - Perform useful transformations (new variables, categories, encoding).
- **Descriptive analysis**
  - Compute descriptive statistics (mean, median, SD, quartiles).
  - Analyze distributions (histograms, density plots, boxplots).
  - Check skewness, kurtosis, outliers (IQR / Z-score).
- **Relationships between variables**
  - Correlation matrix for numeric variables.
  - Scatter plots, boxplots, grouped comparisons.
  - Link results back to your hypotheses.

How to Use Streamlit in the Project

- Reuse patterns from the class:
  - **File upload:** `st.file`$_{u}ploader(”UploadyourCSV”)|$
  - `EDA functions: e.g. show`$_{b}asic_{i}nfo(df)|, loopsforcolumns.$
  - `Sidebar: sliders, selectboxes for filters and options.`
  - `Tabs: separate views for overview, missing data, distributions, etc.`
  - `Columns: 2x2 panels with tables and plots (like in app_8.py).`
- Your app should:
  - be easy to navigate,
  - explain clearly what is shown on each page/tab,
  - guide the user from **data** to **conclusions**.

Grading Criteria

- **Substantive side**
  - Relevance of the problem (scientific/business).
  - Originality of the topic.
  - Quality of goals, questions, and hypotheses.
  - Correct choice and use of methods.
  - Effort in data cleaning and preprocessing.
  - Correctness and depth of the analysis.
  - Clarity of conclusions and recommendations.
- **Application side**
  - Structure and organization of the app.
  - Usability and clarity of the interface.
  - Quality of explanations and documentation.

Presentation and Defense of the Project

- Presentation verifies that:
  - you worked independently,
  - you understand the methods and results in your project.
- You should expect questions about:
  - your data and how you prepared it,
  - the methods you used and why,
  - interpretation of plots and statistics,
  - your conclusions and limitations.
- Evaluation of the presentation:
  - substantive correctness,
  - correctness of answers to questions,
  - structure and organization,
  - staying within the time limit.

Practical Advice

- Start from the example apps from class and **modify** them:
    - replace the dataset,
    - adapt the tabs and panels to your topic,
    - add your own plots and statistics.
- Keep it simple but coherent:
    - clear story from data to conclusions,
    - fewer, well-explained plots are better than many random plots.
- Document your work:
    - comments in code,
    - text in the app (titles, captions),
    - optional short written report.

References

- https://streamlit.io/gallery
- https://docs.streamlit.io/develop/quick-reference/cheat-sheet
- https://github.com/PacktPublishing/Getting-started-with-Streamlit-for-Data-Science
- https://medium.com/data-science-collective/wait-this-was-built-in-streamlit-10-best-streamlit-design-tips-for-dashboards-2b0f50067622