

Membuat Web App Data Science Menggunakan Streamlit

A. Pengenalan Streamlit

1. Apa itu Streamlit?

Streamlit merupakan *library* Python *open-source* yang memungkinkan kita membuat aplikasi web interaktif khusus untuk Data Science dan Machine Learning dalam waktu singkat. Dengan streamlit, kita tidak perlu menguasai HTML, CSS, atau JavaScript untuk membuat website. Cukup dengan bahasa pemrogram Python yang sudah kita ketahui, kita dapat membuat aplikasi web yang interaktif dan menarik.

2. Instalasi & Setup

Sebelum memulai, kita perlu melakukan instalasi streamlit. Ketik perintah berikut pada terminal dan tekan Enter:

```
pip install streamlit
```

Untuk mengecek apakah instalasi berhasil, ketik perintah berikut pada terminal dan tekan Enter:

```
streamlit --version
```

Outputnya dari perintah ini akan menampilkan nomor versi streamlit yang sudah terinstall di komputer Anda. Berikut adalah contoh outputnya:

```
Streamlit, version 1.50.0
```

3. Membuat Aplikasi Hello World Menggunakan Streamlit

Pada bagian ini, kita akan membuat aplikasi hello world menggunakan streamlit. Aplikasinya sederhana, yaitu menampilkan teks "Hello World" ke browser.

Langkah 1: Buat File Python Buka Visual Studio Code. Buat file baru dan beri nama `app.py`.

Langkah 2: Tulis Kode Berikut Salin atau ketik kode ini ke dalam `app.py` :

```
import streamlit as st

st.title("Aplikasi Pertamaku")

st.write("Hello World! Selamat datang di tutorial Streamlit.")
```

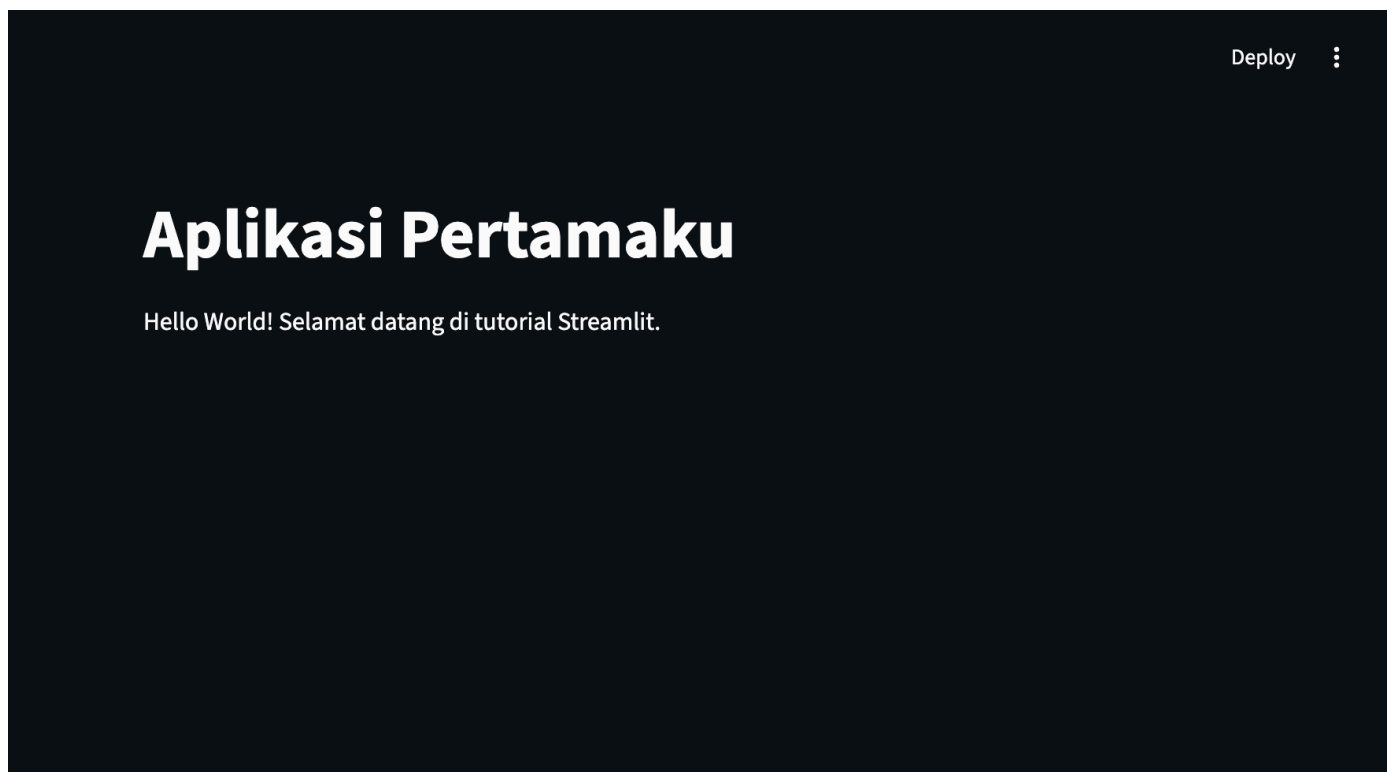
Penjelasan kode di atas:

- `import streamlit as st`: Mengimpor library streamlit dan memberikan alias `st` untuk memudahkan penulisan kode.
- `st.title("Aplikasi Pertamaku")` : Menambahkan judul pada aplikasi web.
- `st.write("Hello World! Selamat datang di tutorial Streamlit.")` : Menambahkan teks pada aplikasi web.

Langkah 3: Menjalankan Aplikasi Buka terminal, lalu ketikkan perintah berikut (pastikan Anda berada di folder yang sama dengan `app.py`):

```
streamlit run app.py
```

Perintah ini akan menjalankan aplikasi streamlit. Browser akan terbuka dan menampilkan halaman web baru. Berikut adalah tampilan aplikasi Hello World versi Streamlit:



B. Menampilkan Informasi Dasar

Setelah berhasil menjalankan aplikasi Hello World, langkah selanjutnya adalah memahami bagaimana menyusun konten. Dalam pembuatan aplikasi web atau dashboard untuk menampilkan data, struktur informasi sangatlah penting agar pembaca dapat memahami konteks dengan cepat.

Streamlit menyediakan beberapa fungsi intuitif untuk menata teks, judul, dan format lainnya. Berikut adalah penjelasan singkat tentang setiap fungsi tersebut.

1. Membangun Struktur Halaman

Seperti halnya menulis dokumen, sebuah aplikasi membutuhkan hierarki informasi yang jelas. Ada tiga elemen utama yang dapat digunakan untuk membangun struktur halaman yaitu Judul Utama, Sub-bab, dan Bagian-bagian kecil lainnya. Streamlit memiliki tiga fungsi utama untuk keperluan ini yaitu `st.title()`, `st.header()`, dan `st.subheader()`. Berikut adalah contoh penggunaan fungsi tersebut.

Buka file `app.py` Anda dan ubah menjadi kode berikut:

```
import streamlit as st

# 1. Judul Utama (Paling besar)
st.title("Dashboard Analisis Data")

# 2. Header (Sub-bab)
st.header("Laporan Penjualan Bulanan")

# 3. Subheader (Bagian lebih kecil dari Header)
st.subheader("Kuartal 1: Januari - Maret")
```

Penjelasan:

- `st.title()` : Digunakan hanya satu kali sebagai judul utama aplikasi. Ukuran font paling besar.
- `st.header()` : Digunakan untuk memisahkan bagian-bagian besar dalam aplikasi Anda.
- `st.subheader()` : Digunakan untuk membagi header menjadi segmen yang lebih spesifik.

Berikut adalah tampilan aplikasi dengan struktur halaman yang sudah dibuat:

Dashboard Analisis Data

Laporan Penjualan Bulanan

Kuartal 1: Januari - Maret

2. Format Teks: Markdown dan Plain Text

Terkadang kita perlu menampilkan paragraf panjang dan membuat poin-poin (bullet points) tertentu. Seringkali kita juga perlu menebalkan kata penting. Untuk kebutuhan ini, kita dapat menggunakan fungsi `st.markdown()`. Fungsi ini memungkinkan kita untuk menampilkan teks dengan format *Markdown*.

Markdown (`st.markdown`) Streamlit mendukung sintaks *Markdown*, yaitu bahasa format teks yang standar digunakan oleh developer (mirip dengan format teks di WhatsApp atau GitHub). Untuk menampilkan teks dengan format *Markdown*, kita dapat menggunakan fungsi `st.markdown()`. Berikut adalah contoh penggunaan fungsi tersebut.

Buka file `app.py` Anda dan tambahkan kode berikut:

```
st.markdown("Berikut adalah analisis **profitabilitas** perusahaan:")

st.markdown("""
- **Produk A**: Mengalami peningkatan *signifikan*.
- **Produk B**: Stabil.
- **Produk C**: Perlu evaluasi [Lihat Detail](https://streamlit.io).
""")
```

Berikut adalah tampilan aplikasi dengan format teks yang sudah dibuat:

Dashboard Analisis Data

Laporan Penjualan Bulanan

Kuartal 1: Januari - Maret

Berikut adalah analisis **profitabilitas** perusahaan:

- **Produk A:** Mengalami peningkatan *signifikan*.
- **Produk B:** Stabil.
- **Produk C:** Perlu evaluasi [Lihat Detail](#).

Dengan `st.markdown`, Kita dapat menggunakan sintaks *Markdown* untuk menampilkan teks dengan format yang lebih rapi dan profesional. Berikut adalah contoh penggunaan sintaks *Markdown*:

- `**teks tebal**` untuk menebalkan (Bold).
- `*teks miring*` untuk memiringkan (Italic).
- `-` atau `1.` untuk membuat daftar (List).

Plain Text (`st.text`) Selain `st.markdown`, Kita juga dapat menampilkan teks "apa adanya" tanpa format (misalnya menampilkan log error atau output terminal), gunakan `st.text()`. Font yang digunakan adalah *monospaced* (seperti tulisan koding). Perhatikan contoh berikut.

Buka file `app.py` Anda dan tambahkan kode berikut:

```
st.text("Ini adalah teks biasa. Tidak bisa dibold atau diitalic.")
```

Berikut adalah tampilan aplikasi dengan format teks yang sudah dibuat:

Dashboard Analisis Data

Laporan Penjualan Bulanan

Kuartal 1: Januari - Maret

Berikut adalah analisis **profitabilitas** perusahaan:

- **Produk A:** Mengalami peningkatan *signifikan*.
- **Produk B:** Stabil.
- **Produk C:** Perlu evaluasi [Lihat Detail](#).

Ini adalah teks biasa. Tidak bisa dibold atau diitalic.

3. Menampilkan Rumus Matematika (LaTeX)

Selain format markdown dan plain text, Kita juga dapat menampilkan rumus matematika menggunakan fungsi `st.latex()`. Sebagai seorang praktisi data, seringkali kita perlu menampilkan persamaan matematika atau rumus statistik. Streamlit memiliki dukungan *built-in* untuk LaTeX. Hal ini sangat berguna ketika Anda membuat aplikasi untuk keperluan edukasi atau presentasi ilmiah. Berikut adalah contoh penggunaan fungsi tersebut.

Buka file `app.py` Anda dan tambahkan kode berikut:

```
st.subheader("Rumus Regresi Linear")
st.markdown("Model ini menggunakan persamaan dasar:")

st.latex(r'''
y = \alpha + \beta x + \epsilon
''')

st.latex(r'''
MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2
''')
```

Berikut adalah tampilan aplikasi dengan format teks yang sudah dibuat:

Rumus Regresi Linear

Model ini menggunakan persamaan dasar:

$$y = \alpha + \beta x + \epsilon$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

C. Menampilkan Dataframe dan Metrics

Setelah memahami cara menyusun tata letak dan teks, pada bagian ini kita akan membahas inti dari pengembangan aplikasi *Data Science* yaitu **Visualisasi Data Tabular**.

Sebagai seorang praktisi data, Anda tentu sudah familiar dengan *library* **Pandas**. Salah satu keunggulan utama Streamlit adalah kemampuannya membaca objek DataFrame dari Pandas dan menampilkannya secara langsung di browser tanpa konfigurasi yang rumit.

Mari kita pelajari dua metode utama menampilkan data, serta cara membuat indikator kinerja (KPI) yang menarik.

Persiapan Data

Sebelum mencoba fitur tampilan, kita perlu menyiapkan data dummy menggunakan Pandas. Buka file `app.py` dan ubah isinya menjadi:

```
import streamlit as st
import pandas as pd

# Membuat Dataframe Sederhana
data = {
    'Produk': ['Laptop', 'Mouse', 'Monitor', 'Keyboard', 'Headset'],
    'Harga': [12000000, 150000, 2500000, 500000, 350000],
    'Terjual': [10, 100, 25, 50, 40]
}

df = pd.DataFrame(data)
```

1. Menampilkan Dataframe Interaktif (`st.dataframe`)

Metode pertama dan yang paling umum digunakan adalah `st.dataframe()`. Fungsi ini berfungsi untuk merender data menjadi tabel interaktif yang sangat fleksibel.

```
st.subheader("1. Dataframe Interaktif")
st.write("Tabel di bawah ini bisa diurutkan (sort) dan diperbesar.")

# Menampilkan dataframe
st.dataframe(df, use_container_width=True)
```

Berikut adalah tampilan aplikasi setelah dijalankan:

Deploy
⋮

1. Dataframe Interaktif

Tabel di bawah ini bisa diurutkan (sort) dan diperbesar.

	Produk	Harga	Terjual
0	Laptop	12000000	10
1	Mouse	150000	100
2	Monitor	2500000	25
3	Keyboard	500000	50
4	Headset	350000	40

2. Menampilkan Tabel Statis (`st.table`)

Terkadang, Anda ingin menampilkan seluruh data tanpa fitur interaksi, persis seperti tampilan tabel pada dokumen laporan cetak. Untuk kebutuhan ini, gunakan `st.table()` .

```
st.subheader("2. Tabel Statis")
st.write("Tabel ini menampilkan seluruh isi data tanpa scrollbar.")

# Menampilkan tabel statis
st.table(df)
```

Berikut adalah tampilan aplikasi setelah dijalankan:

2. Tabel Statis

Tabel ini menampilkan seluruh isi data tanpa scrollbar.

	Produk	Harga	Terjual
0	Laptop	12000000	10
1	Mouse	150000	100
2	Monitor	2500000	25
3	Keyboard	500000	50
4	Headset	350000	40

3. Menampilkan Metrics (KPI)

Pada sebuah *dashboard* data, terkadang kita mendapatkan beberapa data ringkasan. Misalnya, kita ingin menampilkan Total Pendapatan, Suhu Ruangan, atau Persentase Akurasi Model.

Streamlit menyediakan komponen `st.metric()` untuk menampilkan angka-angka ringkasan dengan gaya yang profesional.

```

st.subheader("3. Key Performance Indicators (KPI)")

# Membuat 3 kolom agar metrics berjajar ke samping
col1, col2, col3 = st.columns(3)

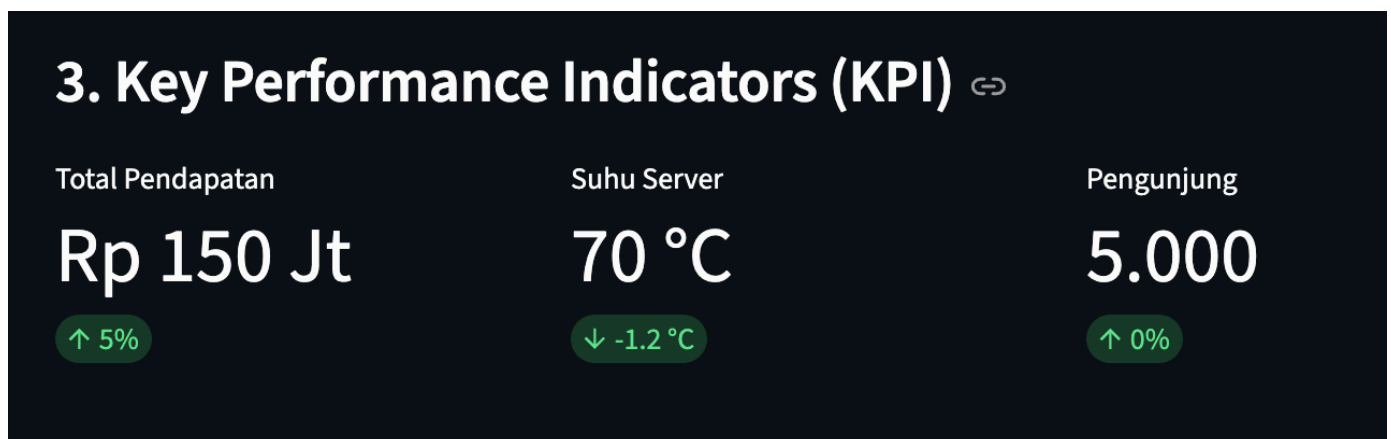
with col1:
    st.metric(
        label="Total Pendapatan",
        value="Rp 150 Jt",
        delta="5%"
    )

with col2:
    st.metric(
        label="Suhu Server",
        value="70 °C",
        delta="-1.2 °C",
        delta_color="inverse"
    )

with col3:
    st.metric(
        label="Pengunjung",
        value="5.000",
        delta="0%"
    )

```

Berikut adalah tampilan aplikasi setelah dijalankan:



Penjelasan Parameter:

1. **Label:** Judul dari metric (misal: "Total Pendapatan").
2. **Value:** Nilai utama yang ingin ditampilkan (bisa berupa string atau angka).
3. **Delta:** Indikator perubahan nilai.
 - Jika positif (misal "5%"), akan muncul panah hijau ke atas.
 - Jika negatif (misal "-1.2 °C"), akan muncul panah merah ke bawah.
4. **Delta Color (inverse):** Terkadang penurunan nilai adalah hal yang baik (contoh: Suhu Server turun, atau *Error Rate* turun). Gunakan `delta_color="inverse"` untuk mengubah warna panah negatif menjadi hijau (positif).

D. Interaktivitas dan Widgets

Kekuatan utama Streamlit terletak pada kemampuannya membuat aplikasi menjadi **interaktif**.

Interaksi memungkinkan pengguna (user) untuk memasukkan data, memilih parameter, atau menekan tombol untuk memicu proses tertentu. Di Streamlit, komponen interaktif ini disebut sebagai **Widgets**.

Prinsip kerjanya sederhana: **Setiap kali pengguna berinteraksi dengan widget, Streamlit akan menjalankan ulang skrip dari atas ke bawah, dan nilai variabel akan diperbarui.**

Berikut adalah beberapa widget yang paling sering digunakan:

1. Button (`st.button`)

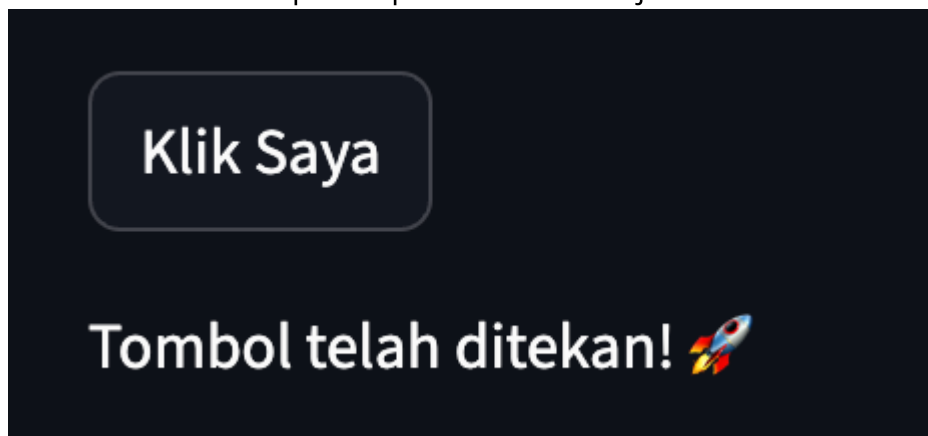
Button digunakan untuk memicu sebuah aksi, misalnya "Simpan Data", "Hitung Prediksi", atau "Reset". Untuk membuat button, kita bisa menggunakan fungsi `st.button()`. Fungsi ini mengembalikan nilai *boolean*:

- `True` : Jika tombol sedang ditekan.
- `False` : Jika tombol tidak ditekan.

Perhatikan contoh kode berikut:

```
if st.button("Klik Saya"):
    st.write("Tombol telah ditekan! 🚀")
else:
    st.write("Menunggu aksi...")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



2. Input Data (Teks & Angka)

Untuk mengambil input dari pengguna, kita menggunakan fungsi input yang spesifik sesuai tipe datanya.

a. Input Teks (`st.text_input`) Digunakan untuk data berupa string, seperti nama, alamat, atau kata kunci pencarian.

b. Input Angka (`st.number_input`) Digunakan untuk data numerik. Kelebihannya, Anda bisa membatasi angka minimum dan maksimum agar input pengguna valid.


Perhatikan contoh kode berikut:

```
# Input Nama (String)
nama = st.text_input("Masukkan Nama Anda")

# Input Umur (Angka)
# min_value=0 memastikan tidak ada umur negatif
umur = st.number_input("Masukkan Umur", min_value=0, max_value=100, step=1)

# Menampilkan hasil input
if nama:
    st.write(f"Halo {nama}, tahun depan umur Anda adalah {umur + 1} tahun.")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



3. Pilihan dan Seleksi

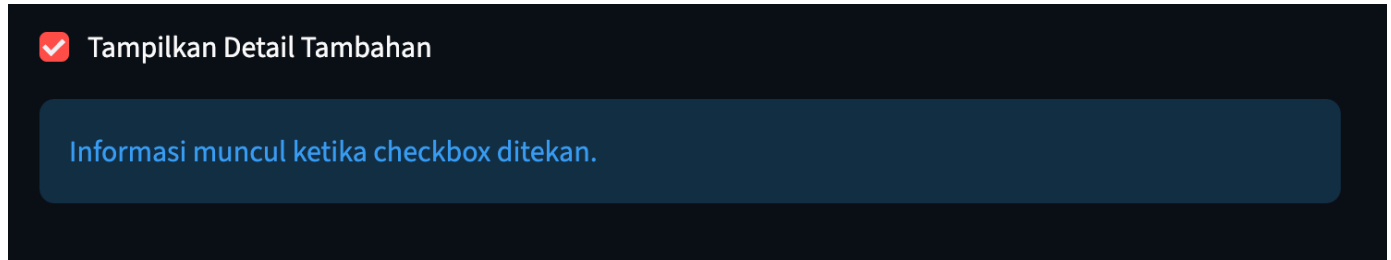
Streamlit menyediakan beberapa cara untuk pemilihan opsi.

a. Checkbox (`st.checkbox`) Cocok untuk opsi biner (Ya/Tidak) atau menampilkan/menyembunyikan elemen (*toggle*).

Perhatikan contoh kode berikut:

```
if st.checkbox("Tampilkan Detail Tambahan"):
    st.info("Informasi muncul ketika checkbox ditekan.")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



b. Pilihan Tunggal (`st.selectbox`) Digunakan untuk memilih satu opsi dari daftar (Dropdown menu). Sangat hemat tempat.

Perhatikan contoh kode berikut:

```
genre = st.selectbox(
    "Pilih Genre Film Favorit:",
    ("Komedi", "Drama", "Aksi", "Horor")
)

st.write(f"Anda memilih genre: {genre}")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



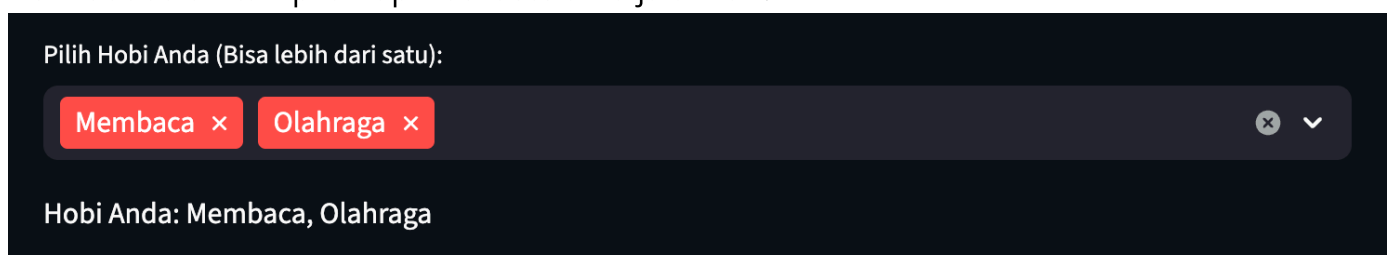
c. Pilihan Ganda (`st.multiselect`) Memungkinkan pengguna untuk memilih lebih dari satu opsi sekaligus (seperti sistem *tagging*). Hasilnya dikembalikan dalam bentuk *List*.

Perhatikan contoh kode berikut:

```
hobi = st.multiselect(
    "Pilih Hobi Anda (Bisa lebih dari satu):",
    ["Membaca", "Traveling", "Gaming", "Olahraga", "Masak"]
)

if len(hobi) > 0:
    st.write(f"Hobi Anda: {' '.join(hobi)}")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



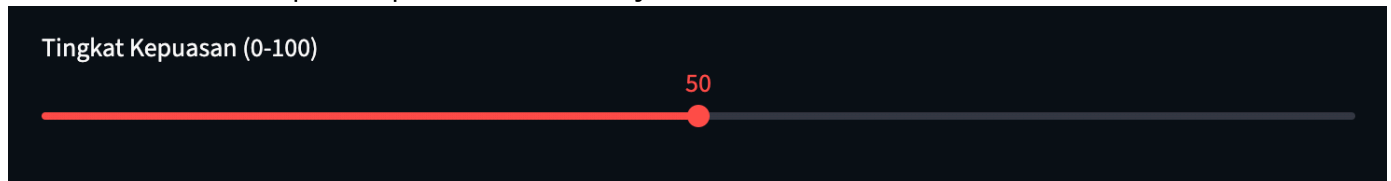
d. Slider (`st.slider`) Berfungsi untuk memilih rentang nilai atau angka dalam skala tertentu.

Perhatikan contoh kode berikut:

```
level = st.slider("Tingkat Kepuasan (0-100)", min_value=0, max_value=100,
value=50)

if level > 80:
    st.write("Terima kasih! Kami senang Anda puas.")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



4. Mini Project: Kalkulator Diskon

Pada bagian ini, kita akan menggabungkan widget di atas untuk membuat **Kalkulator Diskon Sederhana**.

Buat file baru bernama `kalkulator_diskon.py` dan tambahkan kode berikut:

```
st.markdown("---")
st.header("Aplikasi Demo: Kalkulator Diskon")

# 1. Input Harga Awal
harga_barang = st.number_input("Harga Barang (Rp)", min_value=0)

# 2. Input Diskon dengan Slider
diskon = st.slider("Besaran Diskon (%)", 0, 100, 10)

# 3. Tombol Eksekusi
if st.button("Hitung Harga Akhir"):
    nilai_diskon = harga_barang * (diskon / 100)
    harga_akhir = harga_barang - nilai_diskon

    # Menampilkan hasil dengan metric
    st.metric("Harga Setelah Diskon", f"Rp {int(harga_akhir)}", f"-Rp {int(nilai_diskon)}")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



E. Visualisasi Data

Streamlit juga menyediakan fitur visualisasi data yang sangat mudah dan cepat. Visualisasi merupakan cara tercepat bagi otak manusia untuk memahami pola dan tren dalam data.

Streamlit menawarkan dua pendekatan dalam membuat grafik:

1. **Native Charts:** Grafik bawaan Streamlit yang cepat, interaktif, dan mudah dibuat.
2. **External Libraries:** Integrasi dengan *library* populer seperti Matplotlib, Seaborn, atau Plotly untuk kustomisasi tingkat lanjut.

1. Chart Bawaan Streamlit (Native Charts)

Streamlit memiliki fungsi bawaan yang dirancang untuk menerima DataFrame Pandas secara langsung dan mengubahnya menjadi grafik interaktif dalam satu baris kode. Grafik ini sangat ringan dan mendukung fitur *zoom* serta *hover* secara otomatis.

Persiapan Data: Berikut adalah contoh data *dummy* berupa data penjualan harian untuk keperluan demonstrasi. Kemudian, buat sebuah file bernama `chart.py` dan tambahkan kode berikut:

```
import numpy as np
import pandas as pd
import streamlit as st

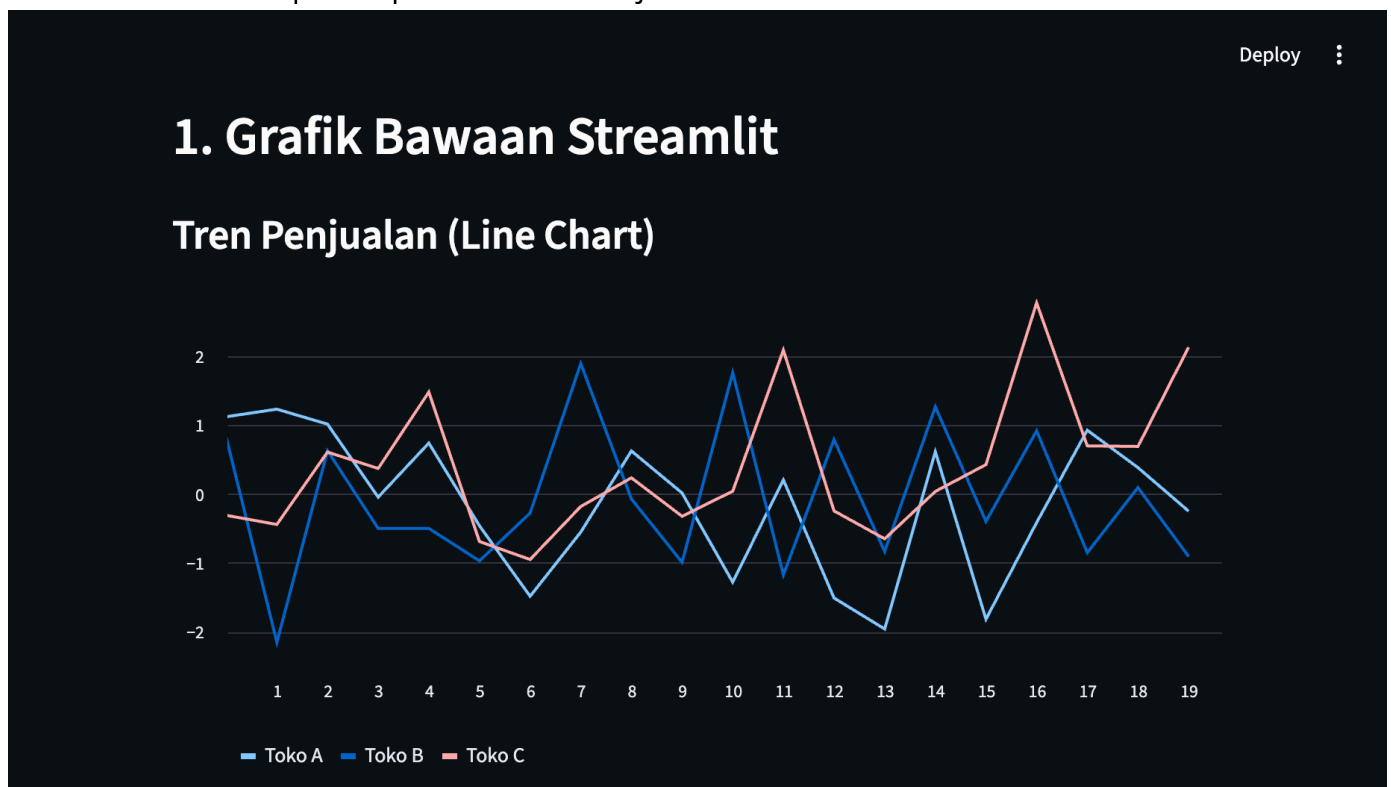
chart_data = pd.DataFrame(
    np.random.randn(20, 3),
    columns=['Toko A', 'Toko B', 'Toko C']
)
```

a. Line Chart (`st.line_chart`) Line Chart digunakan untuk melihat data yang berubah seiring waktu (time series).

```
st.header("1. Grafik Bawaan Streamlit")
st.subheader("Tren Penjualan (Line Chart)")

# Menampilkan grafik garis
st.line_chart(chart_data)
```

Berikut adalah tampilan aplikasi setelah dijalankan:

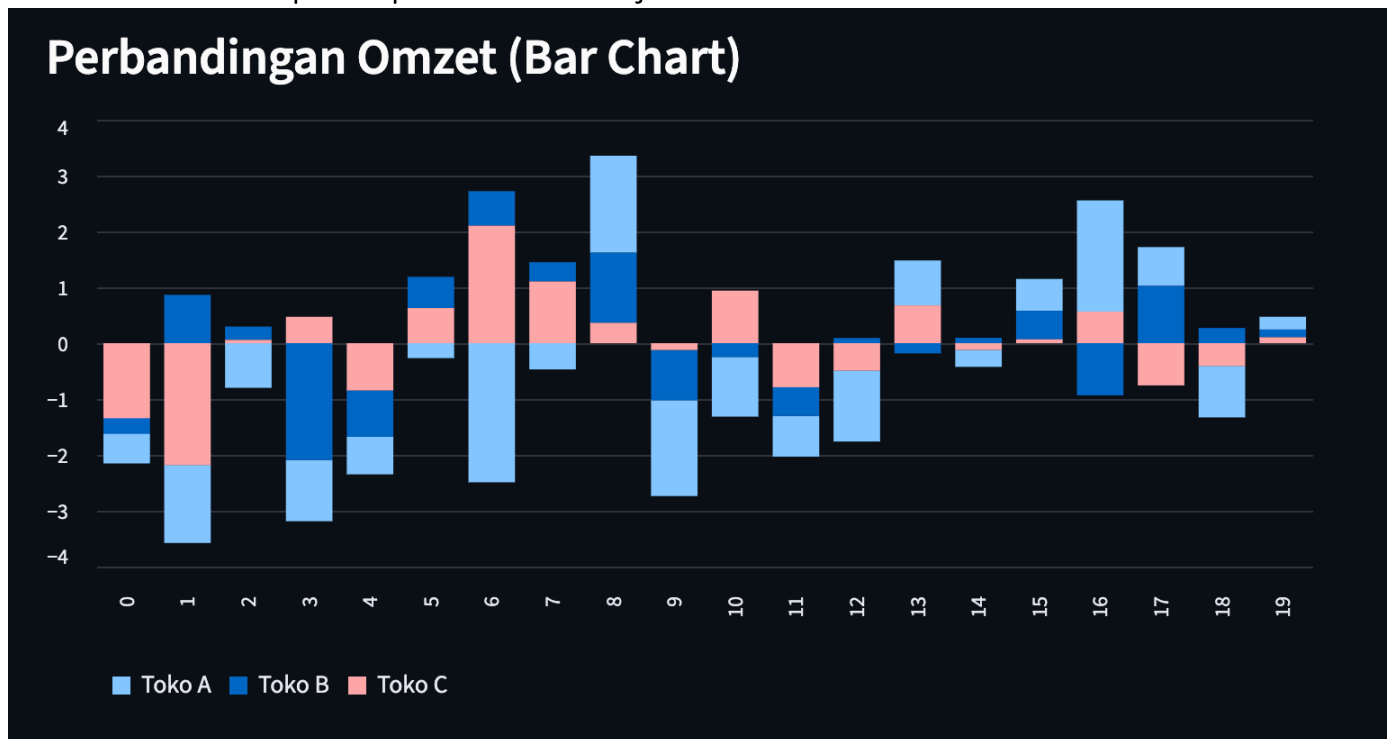


b. Bar Chart (`st.bar_chart`) Bar Chart digunakan untuk membandingkan kategori atau besaran nilai diskrit.

```
st.subheader("Perbandingan Omzet (Bar Chart)")

# Menampilkan grafik batang
st.bar_chart(chart_data)
```

Berikut adalah tampilan aplikasi setelah dijalankan:



Keunggulan Native Charts:

- **Sintaks Minimalis:** Tidak perlu ribet mengatur sumbu X, sumbu Y, atau label secara manual. Streamlit secara otomatis menyesuaikan grafik tersebut.
- **Responsif:** Grafik otomatis menyesuaikan lebar layar, baik di desktop maupun HP.

2. Integrasi Library Lain (`st.pyplot`)

Walaupun mudah digunakan, chart bawaan Streamlit memiliki keterbatasan, yaitu tidak dapat menampilkan jenis plot statistik yang kompleks (seperti *Heatmap* atau *Boxplot*). Jika Anda membutuhkan kustomisasi spesifik, seperti mengubah warna palet, menambah garis batas, atau membuat jenis plot statistik yang kompleks, Anda bisa menggunakan library lain seperti **Matplotlib** atau **Seaborn**.

Streamlit menyediakan fungsi `st.pyplot()` untuk menampilkan plot dari library lain. Caranya yaitu dengan mengirim objek `fig` ke Streamlit.

Penting: Dalam lingkungan web, kita harus menggunakan pendekatan **Object-Oriented** Matplotlib (menggunakan `fig` dan `ax`). Hindari memanggil `plt.show()`, sebagai gantinya kita mengirim objek `fig` ke Streamlit.

Perhatikan contoh kode berikut:

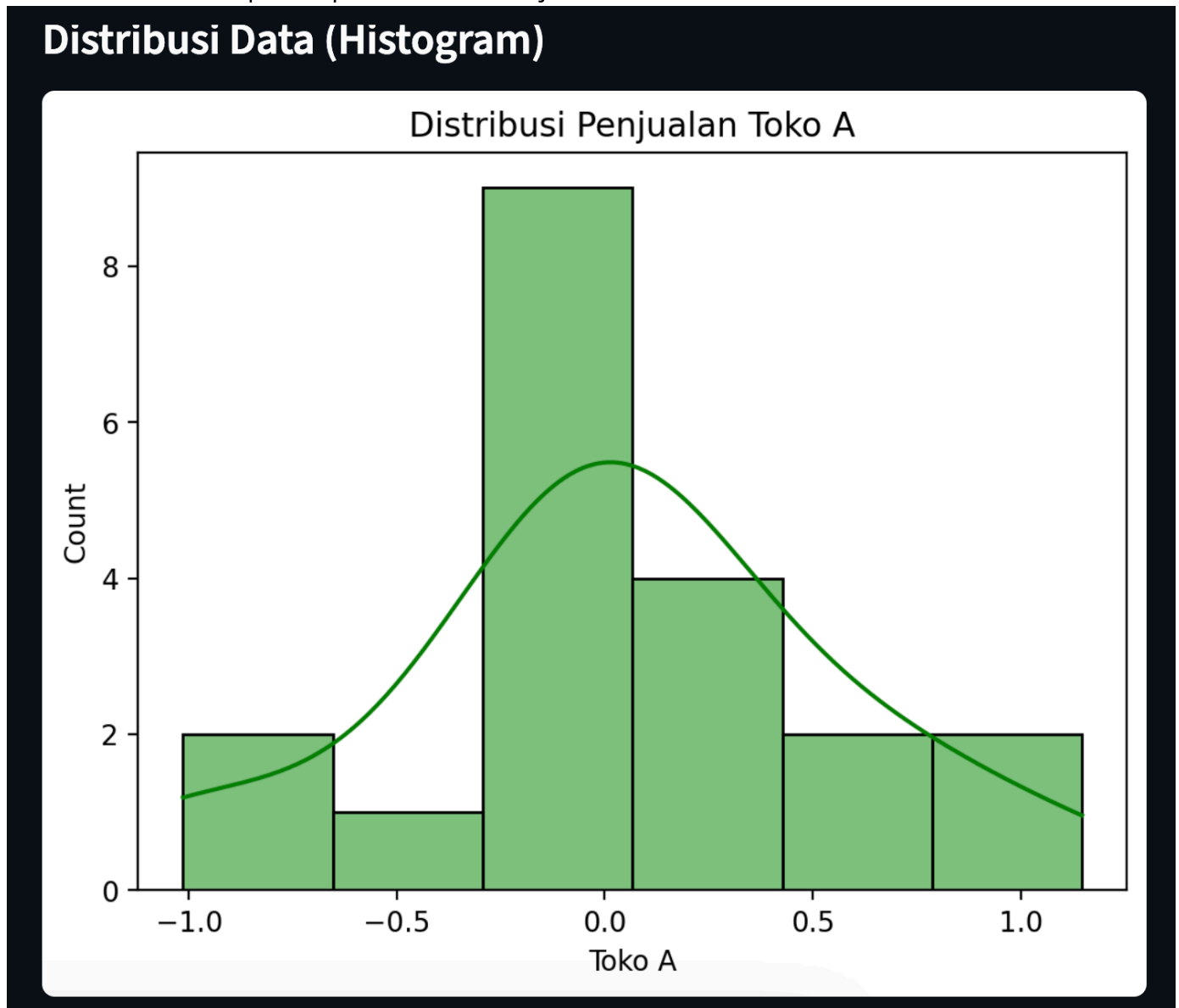
```
import matplotlib.pyplot as plt
import seaborn as sns

st.header("2. Visualisasi Lanjutan (Seaborn/Matplotlib)")

st.subheader("Distribusi Data (Histogram)")

fig, ax = plt.subplots()
sns.histplot(chart_data['Toko A'], kde=True, ax=ax, color="green")
ax.set_title("Distribusi Penjualan Toko A")
st.pyplot(fig)
```

Berikut adalah tampilan aplikasi setelah dijalankan:



F. Layout dan Tata Letak

Pada bagian ini, kita akan mempelajari tentang layouting. Layouting adalah proses menata elemen-elemen aplikasi agar tidak hanya fungsional, tetapi juga terlihat profesional dan rapi (User Friendly). Streamlit menyediakan dua fitur utama layouting: **Sidebar** dan **Columns**.

1. Sidebar

Sidebar adalah area panel di sebelah kiri layar yang bisa disembunyikan atau ditampilkan. Sidebar digunakan untuk menampung **Parameter Input** atau **Navigasi**. Dengan memindahkan tombol, slider, dan input ke sidebar, area utama (main area) menjadi bersih dan bisa fokus menampilkan hasil (grafik/tabel).

Penggunaan Sidebar sangat mudah, yaitu dengan menambahkan kata `.sidebar` sebelum fungsi widget yang biasa Anda gunakan.

Buat sebuah file bernama `sidebar.py` dan tambahkan kode berikut:

```
st.header("Halaman Utama")

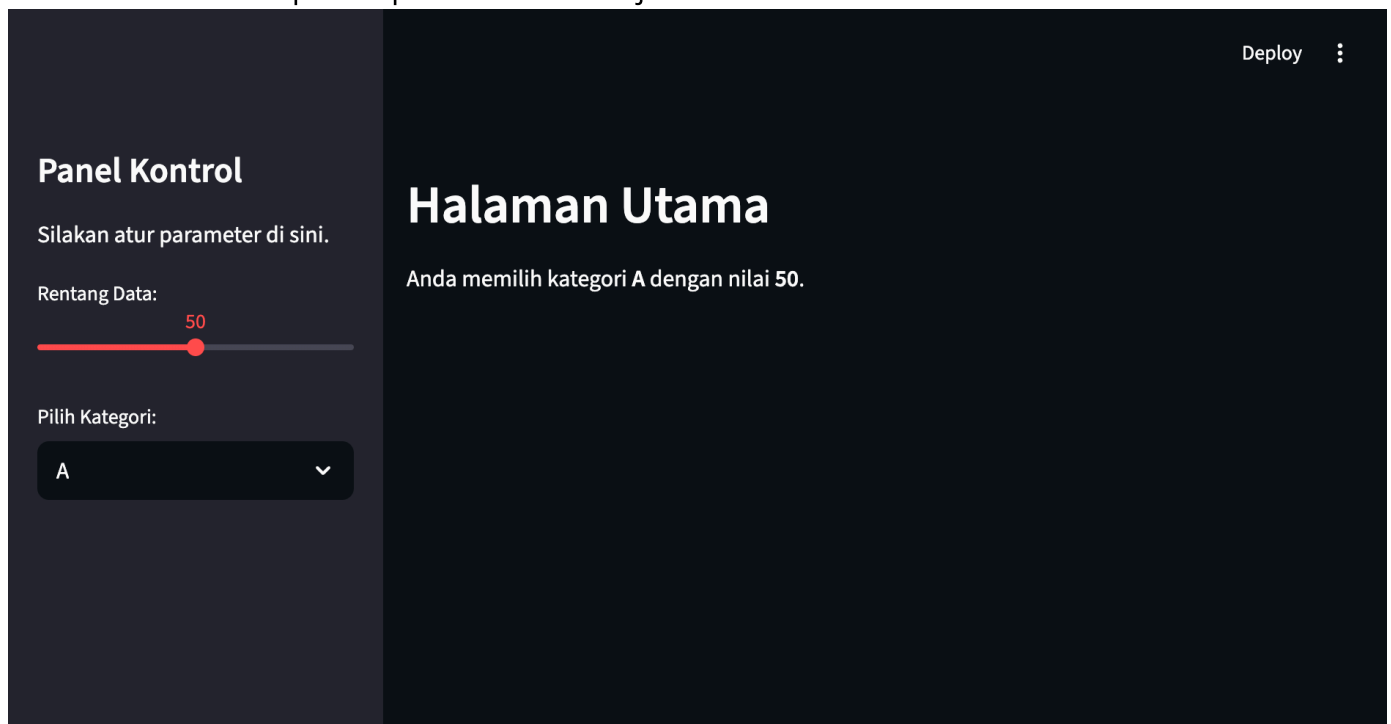
# Menambahkan elemen ke Sidebar
st.sidebar.title("Panel Kontrol")
st.sidebar.write("Silakan atur parameter di sini.")

# Input slider di Sidebar
nilai_filter = st.sidebar.slider("Rentang Data:", 0, 100, 50)

# Input text di Sidebar
kategori = st.sidebar.selectbox("Pilih Kategori:", ["A", "B", "C"])

st.write(f"Anda memilih kategori **{kategori}** dengan nilai **{nilai_filter}**.")
```

Berikut adalah tampilan aplikasi setelah dijalankan:



2. Columns

Column merupakan fitur layouting yang membagi lebar layar menjadi beberapa bagian. Column sering digunakan untuk menampilkan dua informasi secara berdampingan. Misalnya: membandingkan dua grafik, atau menajeri kartu KPI (Metrics) secara horizontal.

Penggunaan Column sangat mudah, yaitu dengan menggunakan fungsi `st.columns()`. Fungsi ini akan mengembalikan beberapa objek `Column` yang dapat diakses melalui variabel.

Buat sebuah file bernama `columns.py` dan tambahkan kode berikut:

```
st.subheader("Layout Kolom")

# Membagi layar menjadi 2 kolom sama besar
col1, col2 = st.columns(2)

# Mengisi Kolom 1 (Kiri)
with col1:
    st.header("Kolom Kiri")
    st.image("https://static.streamlit.io/examples/cat.jpg", caption="Gambar Kucing")
    st.write("Ini adalah teks di kolom pertama.")

# Mengisi Kolom 2 (Kanan)
with col2:
    st.header("Kolom Kanan")
    st.image("https://static.streamlit.io/examples/dog.jpg", caption="Gambar Anjing")
    st.write("Ini adalah teks di kolom kedua.")
```

Berikut adalah tampilan aplikasi setelah dijalankan:

Deploy 

Layout Kolom

Kolom Kiri



Gambar Kucing

Ini adalah teks di kolom pertama.

Kolom Kanan



Gambar Anjing

Ini adalah teks di kolom kedua.