🚀 Sesi 4

Dashboard Streamlit Lengkap

Hari 2 - 120 menit

Dashboard Siap Produksi dengan Chart & Fitur Lanjutan

🎯 Objektif Sesi

Setelah sesi ini, Anda dapat:

- ✅ Menggunakan tabs untuk layout multi-section
- ✅ Integrasi chart Plotly (pie, bar, line, scatter)
- ✅ Menerapkan filter lanjutan dengan caching
- ✅ Optimasi performa dengan @st.cache_data
- ✅ Export data (CSV dengan timestamp)
- ✅ Deploy ke Streamlit Cloud
- ✅ Best practice untuk produksi

🎨 Layout Multi-Section: Tabs

Tabs = Mengorganisasi konten ke beberapa bagian

```
tab1, tab2, tab3, tab4 = st.tabs([
    "📊 Ikhtisar",
    "🏛️ Top Satker",
    "📅 Analisis Tren",
    "📋 Ekspor Data"
])

with tab1:
    st.header("Ikhtisar & Distribusi")
    # Konten overview

with tab2:
    st.header("Top Satker berdasarkan Pagu")
    # Analisis satker
```

Manfaat: UI bersih, konten tertata, UX lebih baik

📊 Integrasi Plotly

Plotly = chart interaktif & profesional

```
import plotly.express as px
import plotly.graph_objects as go

# Pie chart
fig = px.pie(
    values=metode_count.values,
    names=metode_count.index,
    title="Distribusi Metode Pengadaan",
    hole=0.3  # Donut chart
)

st.plotly_chart(fig, use_container_width=True)
```

Fitur: Zoom, pan, hover tooltip, export gambar

📈 Tipe Grafik: Diagram Pie

```
# Siapkan data
metode_count = filtered_df['metode_pengadaan'].value_counts()

# Buat pie chart
fig = px.pie(
    values=metode_count.values,
    names=metode_count.index,
    title="Distribusi Metode Pengadaan",
    hole=0.3,
    color_discrete_sequence=px.colors.qualitative.Set3
)

# Kustomisasi
fig.update_traces(textposition='inside', textinfo='percent+label')

# Tampilkan
st.plotly_chart(fig, use_container_width=True)
```

Tipe Grafik: Diagram Batang

```
# Top 10 Satker berdasarkan total pagu
top_satker = (
    filtered_df.groupby('nama_satker')['pagu']
    .sum()
    .sort_values(ascending=False)
    .head(10)
)

# Buat diagram batang
fig = px.bar(
    x=top_satker.values / 1e9,  # Konversi ke miliar
    y=top_satker.index,
    orientation='h',
    title="Top 10 Satker berdasarkan Total Pagu",
    labels={'x': 'Total Pagu (Miliar Rp)', 'y': 'Satker'}
)

st.plotly_chart(fig, use_container_width=True)
```

📉 Tipe Grafik: Diagram Garis

```
# Kumulatif pagu seiring waktu
cumulative_df = (
    filtered_df
    .sort_values('tanggal_terumumkan')
    .assign(pagu_cumsum=lambda x: x['pagu'].cumsum())
)

# Buat diagram garis
fig = px.line(
    cumulative_df,
    x='tanggal_terumumkan',
    y='pagu_cumsum',
    title="Pagu Kumulatif Seiring Waktu",
    labels={'pagu_cumsum': 'Pagu Kumulatif (Rp)', 'tanggal_terumumkan': 'Tanggal'}
)

st.plotly_chart(fig, use_container_width=True)
```

🎯 Tipe Grafik: Diagram Sebar

```
# Pagu vs HPS
fig = px.scatter(
    filtered_df.head(1000),  # Dibatasi untuk performa
    x='pagu',
    y='hps',
    color='metode_pengadaan',
    size='pagu',
    hover_data=['nama_paket', 'nama_satker'],
    title="Pagu vs HPS",
    labels={'pagu': 'Pagu (Rp)', 'hps': 'HPS (Rp)'}
)

st.plotly_chart(fig, use_container_width=True)
```

💾 Caching Lanjutan

Kenapa cache? Streamlit rerun seluruh script di setiap interaksi

```
# Cache loading data (TTL = 1 jam)
@st.cache_data(ttl=3600)
def load_data():
    df = pd.read_parquet('data.parquet')
    return df

# Cache filtering
@st.cache_data
def apply_filters(_df, metode, jenis, satker, pagu_range):
    """Cache hasil filter"""
    filtered = _df.copy()

    if metode != 'Semua':
        filtered = filtered[filtered['metode_pengadaan'] == metode]
```

💾 Caching Lanjutan

```
    # ... filter lainnya

    return filtered
```

Catatan: Pakai `_df` untuk objek mutable (tidak di-hash)

⚡ Tips Performa

```
# 1. Batasi data untuk chart
df.head(1000)  # Hanya plot 1000 baris pertama

# 2. Gunakan use_container_width
st.plotly_chart(fig, use_container_width=True)

# 3. Cache operasi berat
@st.cache_data
def expensive_computation(df):
    return df.groupby('col').agg({'val': 'sum'})

# 4. Lazy loading - hanya load saat diperlukan
if st.button("Tampilkan Detail"):
    show_expensive_data()
```

📥 Fitur Ekspor

```
from datetime import datetime

# Siapkan data export
export_cols = ['nama_paket', 'pagu', 'metode_pengadaan', 'nama_satker']
export_df = filtered_df[export_cols].copy()

# Format pagu
export_df['pagu_formatted'] = export_df['pagu'].apply(
    lambda x: f"Rp {x:,.0f}"
)

# Buat CSV
csv = export_df.to_csv(index=False).encode('utf-8')

# Tombol download dengan timestamp
timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
st.download_button(
    label="📥 Unduh CSV",
    data=csv,
    file_name=f"rup_filtered_{timestamp}.csv",
    mime="text/csv"
)
```

🎨 Styling Kustom

```
# Custom CSS untuk UI lebih baik
st.markdown("""
    <style>
    .main-header {
        font-size: 3rem;
        color: #1f77b4;
        text-align: center;
        padding: 20px;
    }
    .metric-card {
        background-color: #f0f2f6;
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }
    </style>
""", unsafe_allow_html=True)

# Pakai class kustom
st.markdown('<h1 class="main-header">Dashboard RUP 2025</h1>',
            unsafe_allow_html=True)
```

🔧 Manajemen State

```
# Session state untuk data persisten
if 'filtered_df' not in st.session_state:
    st.session_state.filtered_df = df

# Update ketika filter berubah
if st.button("Terapkan Filter"):
    st.session_state.filtered_df = apply_filters(
        df, metode, jenis, pagu_range
    )

# Gunakan session state
st.dataframe(st.session_state.filtered_df)
```

Kasus penggunaan: Menyimpan data antar rerun, state multi-halaman

📱 Layout Responsif

```
# Sesuaikan layout dengan ukuran layar
col1, col2 = st.columns([2, 1])  # Rasio 2:1

with col1:
    # Konten utama (lebih lebar)
    st.plotly_chart(fig, use_container_width=True)

with col2:
    # Konten sidebar (lebih sempit)
    st.metric("Total", value)

# Mobile-friendly: otomatis stack di layar kecil
st.plotly_chart(fig, use_container_width=True)  # Auto-responsif
```

🎯 Struktur Dashboard Lengkap

```
# 1. Page config
st.set_page_config(page_title="RUP Dashboard", layout="wide")

# 2. Load data dengan cache
@st.cache_data(ttl=3600)
def load_data():
    return pd.read_parquet('data.parquet')

df = load_data()

# 3. Filter di sidebar
with st.sidebar:
    metode = st.selectbox("Metode:", options)
    pagu_range = st.slider("Pagu:", 0, 100)

# 4. Terapkan filter dengan cache
@st.cache_data
def apply_filters(_df, metode, pagu_range):
    # logika filtering
    return filtered_df

filtered_df = apply_filters(df, metode, pagu_range)
```

🎯 Struktur Dashboard Lengkap (Lanjutan)

```
# 5. Konten utama dengan tabs
tab1, tab2, tab3 = st.tabs(["Ikhtisar", "Analisis", "Ekspor"])

with tab1:
    # Metrics
    col1, col2, col3 = st.columns(3)
    col1.metric("Total Paket", len(filtered_df))
    col2.metric("Total Pagu", f"{filtered_df['pagu'].sum()/1e12:.2f} T")

    # Charts
    st.plotly_chart(pie_chart, use_container_width=True)

with tab2:
    st.plotly_chart(bar_chart, use_container_width=True)
    st.plotly_chart(line_chart, use_container_width=True)

with tab3:
    csv = filtered_df.to_csv().encode('utf-8')
    st.download_button("Download", csv, "data.csv")
```

🚀 Deploy ke Streamlit Cloud

Langkah 1: Siapkan Repository

```
# Buat repository GitHub
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/username/repo.git
git push -u origin main
```

Kebutuhan:

- `requirements.txt` di root
- `app_complete.py` (file utama)
- File data atau loading data dari URL

🚀 Deploy ke Streamlit Cloud

Langkah 2: Deploy

1. Buka share.streamlit.io
2. Login dengan GitHub
3. Klik "New app"
4. Pilih repository, branch, dan file utama
5. Klik "Deploy"

Pengaturan lanjutan:

- Versi Python
- Secrets (API key, kredensial)
- Custom subdomain

🔐 Mengelola Secrets

Pengembangan lokal: `.streamlit/secrets.toml`

```
# .streamlit/secrets.toml
API_KEY = "your-api-key"
DATABASE_URL = "postgresql://..."
```

Produksi (Streamlit Cloud):

- Settings > Secrets
- Tempel format TOML

Akses di kode:

```
import streamlit as st

api_key = st.secrets["API_KEY"]
db_url = st.secrets["DATABASE_URL"]
```

📊 Contoh: Dashboard Lengkap

Fitur utama di `app_complete.py`:

- ✅ **4 tab:** Ikhtisar, Top Satker, Tren, Ekspor
- ✅ **Chart Plotly:** Pie, bar, line, scatter
- ✅ **Filter lanjutan:** Multiselect, slider, date range
- ✅ **Caching:** `@st.cache_data` untuk data & filter
- ✅ **Metrics:** 4 kartu KPI dengan delta
- ✅ **Export:** CSV dengan timestamp
- ✅ **Responsif:** use_container_width=True
- ✅ **UI Profesional:** CSS kustom, emoji, formatting

🎯 Checklist Praktik Terbaik

- ✅ **Gunakan caching** untuk load data & operasi berat
- ✅ **Batasi data** untuk visualisasi (head/sample)
- ✅ **use_container_width=True** untuk chart responsif
- ✅ **Organisasi dengan tabs** untuk dashboard kompleks
- ✅ **Tambahkan progress indicator** untuk operasi lama
- ✅ **Sediakan tombol download** untuk export data
- ✅ **Validasi input user** sebelum diproses
- ✅ **Tangani error dengan baik** via try/except
- ✅ **Berikan pesan bantu** (info, warning, success)
- ✅ **Dokumentasikan kode** dengan docstring & komentar

⚠️ Jebakan Umum

❌ **Jangan:** Load data tanpa caching

```
df = pd.read_parquet('data.parquet')  # Load tiap rerun!
```

✅ **Lakukan:** Gunakan @st.cache_data

```
@st.cache_data
def load_data():
    return pd.read_parquet('data.parquet')
```

⚠️ Jebakan Umum (Lanjutan)

❌ **Jangan:** Plot seluruh dataset besar

```
fig = px.scatter(df)  # 1M baris = lambat!
```

✅ **Lakukan:** Batasi atau sample data

```
fig = px.scatter(df.head(1000))  # Jauh lebih cepat
```

⚠️ Jebakan Umum (Lanjutan)

❌ **Jangan:** Lupa menangani data kosong

```
st.plotly_chart(fig)  # Error jika df kosong!
```

✅ **Lakukan:** Cek sebelum tampilkan

```
if len(filtered_df) > 0:
    st.plotly_chart(fig)
else:
    st.warning("No data to display")
```

🎓 Latihan Praktik

Bangun dashboard produksi (90 menit):

1. ✅ Setup tabs (Ikhtisar, Analisis, Ekspor)
2. ✅ Tambah 4 metrics dengan delta
3. ✅ Buat pie chart (metode pengadaan)
4. ✅ Buat bar chart (top 10 satker)
5. ✅ Buat line chart (cumulative pagu)
6. ✅ Implement caching untuk filters
7. ✅ Tambah tombol download dengan timestamp
8. ✅ CSS kustom untuk styling

File: `app_complete.py`

🚀 Latihan Deploy (30 menit)

Deploy dashboard Anda:

1. ✅ Buat repository GitHub
2. ✅ Push kode dengan requirements.txt
3. ✅ Login ke share.streamlit.io
4. ✅ Deploy app
5. ✅ Tes app yang ter-deploy
6. ✅ Bagikan URL ke teman!

Bonus: Tambah custom subdomain, atur secrets

📚 Ringkasan Inti

✅ **Tabs** untuk layout multi-section yang rapi
✅ **Plotly** untuk chart interaktif & profesional
✅ **Caching** krusial untuk performa (@st.cache_data)
✅ **Export** dengan timestamp memudahkan pengguna
✅ **Responsif** dengan use_container_width=True
✅ **Deploy** mudah via Streamlit Cloud
✅ **Best practice** untuk app siap produksi

🎯 Perbandingan Grafik

Tipe Grafik	Kasus Penggunaan	Fungsi Plotly
Pie	Distribusi/Proporsi	`px.pie()`
Bar	Perbandingan/Peringkat	`px.bar()`
Line	Tren dari waktu ke waktu	`px.line()`
Scatter	Korelasi	`px.scatter()`
Histogram	Distribusi frekuensi	`px.histogram()`
Box	Distribusi statistik	`px.box()`

🔗 Topik Lanjutan (Opsional)

- **Multi-page apps:** Struktur folder `pages/`
- **Custom components:** Komponen React di Streamlit
- **Integrasi database:** PostgreSQL, MongoDB
- **Data real-time:** WebSockets, streaming
- **Autentikasi:** streamlit-authenticator
- **Testing:** pytest dengan Streamlit
- **CI/CD:** GitHub Actions untuk auto-deploy

🔗 Referensi

- **Streamlit Docs:** https://docs.streamlit.io/
- **API Reference:** https://docs.streamlit.io/library/api-reference
- **Plotly Docs:** https://plotly.com/python/
- **Plotly Chart Types:** https://plotly.com/python/basic-charts/
- **Gallery:** https://streamlit.io/gallery
- **Cheat Sheet:** https://docs.streamlit.io/library/cheatsheet
- **Components:** https://streamlit.io/components
- **Deploy Guide:** https://docs.streamlit.io/streamlit-community-cloud

💡 Langkah Selanjutnya

Setelah bootcamp:

1. **Practice:** Bangun lebih banyak dashboard dengan dataset berbeda
2. **Explore:** Coba multi-page apps, custom components
3. **Share:** Deploy app, bangun portofolio
4. **Learn:** Fitur lanjutan (auth, database, real-time)
5. **Join community:** Forum Streamlit, Discord
6. **Contribute:** Open source, berbagi komponen

🎉 Selamat!

Anda telah menuntaskan Bootcamp Streamlit!

Sekarang Anda bisa:

✅ Membangun data app interaktif dengan Streamlit
✅ Membuat dashboard profesional dengan Plotly
✅ Deploy ke produksi dengan Streamlit Cloud
✅ Menerapkan best practice untuk performa & UX

Terus berkarya! 🚀

📞 Dukungan & Komunitas

- **Pertanyaan:** Streamlit Forum
- **Discord:** Streamlit Discord
- **GitHub:** Streamlit GitHub
- **Twitter:** @streamlit
- **YouTube:** Streamlit Channel

Happy building! 💻

🎉 Terima kasih!

Bootcamp Streamlit telah selesai!

Apa yang akan Anda bangun selanjutnya? 🚀

Kembangkan keterampilan Anda dengan Streamlit! 💡