🚀 Sesi 5

Dashboard Interaktif dengan Streamlit

HARI 2: Analisis Lanjutan & Dashboard

Durasi: 4.5 jam (13:00 - 17:00) *includes break*

Aplikasi:

- `01_hello_streamlit.py`
- `02_components_demo.py`
- `03_data_explorer.py`
- `rup_dashboard.py`

🎯 Tujuan Sesi

Setelah sesi ini, Anda bisa:

- ✅ Membuat web app dengan Streamlit (tanpa web dev background!)
- ✅ Menggunakan text elements dan data displays
- ✅ Implementasi interactive widgets (buttons, sliders, selects)
- ✅ Mengelola session state
- ✅ Penggabungan dengan Pandas & DuckDB
- ✅ Membuat charts interaktif dengan Plotly
- ✅ Build production-ready dashboard

📋 Agenda Sesi

Waktu	Topik	Aplikasi	Durasi
13:00 - 13:20	Kick-off & Agenda	`01_hello_streamlit.py`	20 min
13:20 - 14:00	Layout & Text Components	`01_hello_streamlit.py`	40 min
14:00 - 14:45	Input Widgets & Session State	`02_components_demo.py`	45 min
14:45 - 15:00	Data Display & Caching	`02_components_demo.py`	15 min
15:00 - 15:30	**BREAK**	☕	30 min
15:30 - 16:15	Data Explorer Build & Metrics	`03_data_explorer.py`	45 min
16:15 - 17:00	Dashboard Architecture, Deployment & Q&A	`rup_dashboard.py`	45 min

🚀 Apa itu Streamlit?

Penjelasan Sederhana: Streamlit = cara termudah buat web app dengan Python
Gak perlu belajar HTML/CSS/JavaScript!

Framework Python untuk Data Apps

Fitur

- 🐍 **Pure Python** - Tanpa HTML/CSS/JS
- ⚡ **Fast Development** - Bangun dalam hitungan jam
- 🎨 **UI yang Cantik** - Otomatis responsif
- 🔄 **Reactive** - Auto-rerun on changes
- 🆓 **Free Deployment** - Streamlit Cloud

Kapan Dipakai

- Dashboard data
- Demo ML
- Explorer data
- Tools admin
- Prototyping
- Tools internal

"From Data to Web App in Minutes!" 🎯

🛠️ Installation & Setup

```
# Install Streamlit
pip install streamlit

# atau dengan uv
uv pip install streamlit

# Check installation
streamlit --version

# Run hello app (built-in)
streamlit hello
```

Create Your First App:

```
# app.py
import streamlit as st

st.title("Hello Streamlit! 👋")
st.write("My first data app!")
```

```
# Run app
streamlit run app.py
```

Hello Streamlit

Aplikasi: `01_hello_streamlit.py`

Tujuan:

- Understand Streamlit basics
- Text elements
- Data display
- Simple charts
- Layouts

```
# Run the app
cd day2/session5_streamlit/apps
uv run streamlit run 01_hello_streamlit.py

# atau
streamlit run 01_hello_streamlit.py
```

URL: http://localhost:8501

✍️ Text Elements

```python
import streamlit as st

# Titles and headers
st.title("🎯 Main Title")
st.header("Header Level 1")
st.subheader("Subheader Level 2")

# Text variants
st.text("Plain text")
st.markdown("**Bold** and *italic* with markdown")
st.latex(r"\sum_{i=1}^{n} x_i^2")

# Code blocks
st.code("""
def hello():
    print("Hello, World!")
""", language='python')

# Magic! (automatic st.write)
"This is magic text!"
x = 42
x  # Automatically displayed
```

Displaying Data

```python
import pandas as pd
import streamlit as st

# Load data
df = pd.read_parquet('data.parquet')

# Interactive dataframe
st.dataframe(df)  # Bisa diurutkan, bisa di-scroll

# Tabel statis
st.table(df.head(10))

# Metrics with delta
col1, col2, col3 = st.columns(3)
col1.metric("Total Paket", "16,430", "+123")
col2.metric("Total Pagu", "5.2T", "+2.3%")
col3.metric("Avg Pagu", "316M", "-1.5%")

# JSON display
st.json({"name": "Streamlit", "version": "1.28"})
```

📈 Basic Charts

```
# Grafik bawaan (sederhana tapi terbatas)
st.line_chart(df['pagu'])
st.bar_chart(df.groupby('metode_pengadaan')['pagu'].sum())
st.area_chart(df)

# Plotly charts (direkomendasikan!)
import plotly.express as px

fig = px.bar(df, x='metode_pengadaan', y='pagu',
             title='Pagu by Metode')
st.plotly_chart(fig, use_container_width=True)

# Matplotlib/Seaborn
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.hist(df['pagu'], bins=50)
st.pyplot(fig)
```

💬 Status Messages

```python
# Success, info, warning, error
st.success("✅ Data loaded successfully!")
st.info("ℹ️ Information message")
st.warning("⚠️ Warning message")
st.error("❌ Error message")

# Divider
st.divider()

# Spinner for loading
with st.spinner("Loading data..."):
    time.sleep(2)  # Simulate loading
    st.success("Done!")

# Progress bar
progress_bar = st.progress(0)
for i in range(100):
    progress_bar.progress(i + 1)
    time.sleep(0.01)
```

📐 Layouts: Columns

```python
# Create columns
col1, col2, col3 = st.columns(3)

with col1:
    st.header("Column 1")
    st.write("Content 1")

with col2:
    st.header("Column 2")
    st.write("Content 2")

with col3:
    st.header("Column 3")
    st.write("Content 3")

# Rasio lebar kustom
col1, col2 = st.columns([2, 1])  # 2:1 ratio

# Kolom responsif
for col in st.columns(4):
    col.metric("Metric", "100", "5%")
```

📦 Layouts: Container & Expander

```python
# Container (pengelompokan)
with st.container():
    st.write("Inside container")
    st.write("More content")

# Expander (collapsible)
with st.expander("Klik untuk expand"):
    st.write("Hidden content")
    st.image("chart.png")

# Multiple expanders
with st.expander("📊 Charts"):
    st.line_chart(data)

with st.expander("📋 Data"):
    st.dataframe(df)

with st.expander("ℹ️ Info"):
    st.write("Documentation here")
```

Layouts: Tabs

```python
# Create tabs
tab1, tab2, tab3 = st.tabs(["📊 Charts", "📋 Data", "🔍 Analysis"])

with tab1:
    st.header("Charts")
    st.line_chart(data)

with tab2:
    st.header("Data Table")
    st.dataframe(df)

with tab3:
    st.header("Analysis")
    st.write("Statistical analysis here")

# Dynamic tabs
tabs = st.tabs([f"Tab {i}" for i in range(5)])
for i, tab in enumerate(tabs):
    with tab:
        st.write(f"Content of tab {i}")
```

🎮 Interactive Components

Aplikasi: `02_components_demo.py`

Yang akan dipelajari:

- Input widgets (buttons, sliders, selects)
- Forms
- Session state
- Callbacks

```
uv run streamlit run 02_components_demo.py
```

Anda akan bisa:

- Bagaimana cara capture user input
- Bagaimana cara maintain state
- Cara untuk membuat interactive experiences

Buttons

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

# Button with key (for uniqueness)
if st.button("Submit", key="submit_btn"):
    st.success("Submitted!")

# Download button
csv = df.to_csv(index=False)
st.download_button(
    label="📥 Download CSV",
    data=csv,
    file_name="data.csv",
    mime="text/csv"
)

# Link button (opens URL)
st.link_button("Go to Documentation",
               "https://docs.streamlit.io")
```

☑ Selection Widgets

```
# Checkbox
show_data = st.checkbox("Show raw data")
if show_data:
    st.dataframe(df)

# Toggle (switch)
enabled = st.toggle("Enable feature")

# Radio buttons
option = st.radio(
    "Choose metode:",
    options=["Tender", "Penunjukan Langsung", "Tender Cepat"]
)

# Select box (dropdown)
selected = st.selectbox(
    "Select satker:",
    options=df['nama_satker'].unique()
)

# Multi-select
selected_list = st.multiselect(
    "Select multiple",
    options=df['metode_pengadaan'].unique()
)
```

Sliders & Input

```python
# Slider
age = st.slider("Select age", min_value=0, max_value=100, value=25)

# Range slider
pagu_range = st.slider(
    "Pagu range",
    min_value=0,
    max_value=int(df['pagu'].max()),
    value=(0, 1000000000)
)

# Number input
quantity = st.number_input(
    "Enter quantity",
    min_value=0,
    max_value=100,
    value=10,
    step=1
)

# Text input
name = st.text_input("Enter name", value="", placeholder="Your name...")
notes = st.text_area("Notes", height=150)
```

Date & File Input

```python
# Date input
start_date = st.date_input("Start date")

# Date range
from datetime import datetime
date_range = st.date_input(
    "Date range",
    value=(datetime(2025, 1, 1), datetime(2025, 12, 31))
)

# Time input
appointment = st.time_input("Appointment time")

# File uploader
uploaded_file = st.file_uploader(
    "Choose a file",
    type=["csv", "xlsx", "parquet"]
)

if uploaded_file:
    df = pd.read_csv(uploaded_file)
    st.dataframe(df)
```

📝 Forms - Kumpulkan Input Sekaligus

Penjelasan: Form berguna biar app tidak rerun setiap kali user input sesuatu.
Semua input dikumpulkan, baru di-submit sekaligus

```
# Form (kirim sekaligus)
with st.form("my_form"):
    st.write("Fill out the form")

    name = st.text_input("Name")
    age = st.slider("Age", 0, 100, 25)
    metode = st.selectbox("Metode", ["Tender", "Langsung"])

    # Every form must have a submit button
    submitted = st.form_submit_button("Submit")

    if submitted:
        st.write(f"Name: {name}")
        st.write(f"Age: {age}")
        st.write(f"Metode: {metode}")

# Form prevents auto-rerun until submit clicked
```

💾 Session State - Simpan Data Sementara

Penjelasan: Session state buat menyimpan data selama user pakai app
Contoh: simpan angka counter, data yang di-filter, dll

Persist Data Across Reruns

```
# Initialize session state
if 'count' not in st.session_state:
    st.session_state.count = 0

# Increment counter
if st.button("Increment"):
    st.session_state.count += 1

st.write(f"Count: {st.session_state.count}")

# Store complex data
if 'df' not in st.session_state:
    st.session_state.df = pd.read_parquet('data.parquet')

# Use stored data
filtered_df = st.session_state.df[
    st.session_state.df['pagu'] > 1000000
]
```

🔄 Callbacks - Aksi Saat Ada Perubahan

Penjelasan: Callback = fungsi (blok kode yang bisa dipanggil) yang jalan otomatis saat widget berubah

Contoh: auto-hitung saat slider berubah

```
# Callback fungsi (blok kode yang bisa dipanggil)
def update_value():
    st.session_state.value_squared = st.session_state.slider_value ** 2

# Widget with callback
st.slider(
    "Select value",
    min_value=0,
    max_value=100,
    value=10,
    key="slider_value",
    on_change=update_value
)

# Display calculated value
if 'value_squared' in st.session_state:
    st.write(f"Squared: {st.session_state.value_squared}")
```

🔄 Callbacks - Aksi Saat Ada Perubahan (lanjutan)

```
# Button callback
def reset():
    st.session_state.count = 0

st.button("Reset", on_click=reset)
```

📌 Sidebar

```
# Add to sidebar
st.sidebar.title("Filters")
st.sidebar.write("Configure your dashboard")

# Sidebar widgets
metode = st.sidebar.selectbox(
    "Metode Pengadaan",
    options=['All'] + list(df['metode_pengadaan'].unique())
)

pagu_min = st.sidebar.number_input(
    "Min Pagu",
    value=0,
    step=1000000
)

show_chart = st.sidebar.checkbox("Show Charts", value=True)

# Sidebar sections
with st.sidebar:
    st.header("Section 1")
    # widgets here

    st.header("Section 2")
    # more widgets
```

🔍 Data Explorer

Aplikasi: `03_data_explorer.py`

Lengkap Analisis Data Workflow:

1. File upload (CSV, Excel, Parquet)
2. Data preview & summary stats
3. Interactive filters
4. Multiple analysis tabs
5. SQL playground
6. Export fitur

```
uv run streamlit run 03_data_explorer.py
```

This is a complete data exploration tool! 🎯

File Upload & Preview

```python
import streamlit as st
import pandas as pd

st.title("📊 Data Explorer")

# File uploader
uploaded_file = st.file_uploader(
    "Upload your dataset",
    type=["csv", "xlsx", "parquet"]
)

if uploaded_file:
    # Read file berdasarkan type
    if uploaded_file.name.endswith('.csv'):
        df = pd.read_csv(uploaded_file)
    elif uploaded_file.name.endswith('.xlsx'):
        df = pd.read_excel(uploaded_file)
    elif uploaded_file.name.endswith('.parquet'):
        df = pd.read_parquet(uploaded_file)

    st.success(f"✅ Loaded {len(df):,} rows, {len(df.columns)} columns")

    # Preview
    st.dataframe(df.head())
```

🎯 Interactive Filters

```
st.sidebar.header("🔍 Filters")

# Categorical filter
columns = df.columns.tolist()
cat_col = st.sidebar.selectbox("Categorical column", columns)

if df[cat_col].dtype == 'object':
    selected_values = st.sidebar.multiselect(
        f"Select {cat_col}",
        options=df[cat_col].unique()
    )

    if selected_values:
        df = df[df[cat_col].isin(selected_values)]
```

🎯 Interactive Filters (lanjutan)

```
# Numeric filter
numeric_cols = df.select_dtypes(include=['number']).columns
if len(numeric_cols) > 0:
    num_col = st.sidebar.selectbox("Numeric column", numeric_cols)

    min_val = float(df[num_col].min())
    max_val = float(df[num_col].max())

    range_values = st.sidebar.slider(
        f"Range for {num_col}",
        min_val, max_val, (min_val, max_val)
    )

    df = df[(df[num_col] >= range_values[0]) &
            (df[num_col] <= range_values[1])]

st.write(f"Filtered data: {len(df):,} rows")
```

📊 Analysis Tabs

```python
tab1, tab2, tab3 = st.tabs(["📊 Distribution", "📈 Trends", "🔗 Relationships"])

with tab1:
    st.header("Distribution Analysis")

    col = st.selectbox("Select column for histogram", numeric_cols)

    import plotly.express as px
    fig = px.histogram(df, x=col, title=f"Distribution of {col}")
    st.plotly_chart(fig, use_container_width=True)

with tab2:
    st.header("Trend Analysis")

    # Time series if date column exists
    date_cols = df.select_dtypes(include=['datetime64']).columns
    if len(date_cols) > 0:
        date_col = st.selectbox("Date column", date_cols)
        daily = df.groupby(df[date_col].dt.date).size()
        fig = px.line(daily, title="Daily Trend")
        st.plotly_chart(fig, use_container_width=True)
```

🦆 DuckDB Integration

```python
import duckdb

# Cache DuckDB connection
@st.cache_resource
def get_duckdb_connection():
    return duckdb.connect(':memory:')

conn = get_duckdb_connection()

# Register DataFrame
conn.register('data', df)

# SQL Playground
st.header("🦆 SQL Playground")

query = st.text_area(
    "Write your SQL query",
    value="SELECT * FROM data LIMIT 10",
    height=150
)
```

🦆 DuckDB Integration (lanjutan)

```
if st.button("Run Query"):
    try:
        result = conn.execute(query).df()
        st.success(f"✅ Query returned {len(result)} rows")
        st.dataframe(result)
    except Exception as e:
        st.error(f"❌ Error: {e}")
```

💾 Caching untuk Kecepatan

```python
# Cache data loading
@st.cache_data
def load_data(file_path):
    """Cache the data loading"""
    return pd.read_parquet(file_path)

df = load_data('data.parquet')

# Cache resource (connections)
@st.cache_resource
def get_connection():
    """Cache database connection"""
    return duckdb.connect('analytics.duckdb')

conn = get_connection()

# Cache expensive computations
@st.cache_data
def expensive_computation(df):
    """Cache computational results"""
    # Rumit processing
    return processed_data
```

Caching = ⚡ Kecepatan!

Export Functionality

```
st.header("📥 Export Data")

col1, col2, col3 = st.columns(3)

# Export to CSV
with col1:
    csv = df.to_csv(index=False)
    st.download_button(
        "Download CSV",
        csv,
        "data.csv",
        "text/csv"
    )

# Export to Excel
with col2:
    from io import BytesIO
    buffer = BytesIO()
    df.to_excel(buffer, index=False)
    st.download_button(
        "Download Excel",
        buffer.getvalue(),
        "data.xlsx",
        "application/vnd.ms-excel"
    )
```

Export Functionality (lanjutan)

```
# Export to Parquet
with col3:
    parquet = df.to_parquet(index=False)
    st.download_button(
        "Download Parquet",
        parquet,
        "data.parquet"
    )
```

🏆 Production Dashboard

Aplikasi: `rup_dashboard.py`

Production-Ready Features:

- ✅ Professional UI/UX
- ✅ Custom styling & branding
- ✅ Complicated data processing
- ✅ Advanced display
- ✅ Optimization speed
- ✅ Error handling

```
uv run streamlit run rup_dashboard.py
```

This is what a real dashboard looks like! 🎯

🎨 Page Configuration

```python
import streamlit as st

# MUST be the first Streamlit command
st.set_page_config(
    page_title="RUP Dashboard 2025",
    page_icon="📊",
    layout="wide",  # or "centered"
    initial_sidebar_state="expanded",
    menu_items={
        'Get Help': 'https://docs.streamlit.io',
        'Report a bug': 'https://github.com/...',
        'About': '# RUP Analysis Dashboard\nVersion 1.0'
    }
)
```

🖌️ Custom CSS Styling

```
# Custom CSS
st.markdown("""
    <style>
    .main-header {
        font-size: 3rem;
        font-weight: bold;
        color: #1f77b4;
        text-align: center;
        margin-bottom: 2rem;
    }

    .metric-card {
        background-color: #f0f2f6;
        padding: 1rem;
        border-radius: 0.5rem;
        box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    }

    .stButton>button {
        width: 100%;
        background-color: #1f77b4;
        color: white;
    }
    </style>
""", unsafe_allow_html=True)
```

🖌️ Custom CSS Styling (lanjutan)

```
# Use custom HTML
st.markdown('<h1 class="main-header">📊 Dashboard</h1>',
            unsafe_allow_html=True)
```

KPI Cards

```python
# Calculate KPIs
total_paket = len(df)
total_pagu = df['pagu'].sum()
avg_pagu = df['pagu'].mean()
max_pagu = df['pagu'].max()

# Display as metrics
col1, col2, col3, col4 = st.columns(4)

with col1:
    st.metric(
        "Total Paket",
        f"{total_paket:,}",
        delta="+123 dari bulan lalu"
    )

with col2:
    st.metric(
        "Total Pagu",
        f"{total_pagu/1e12:.2f}T",
        delta="+2.3%"
    )
```

KPI Cards (lanjutan)

```
with col3:
    st.metric(
        "Avg Pagu",
        f"{avg_pagu/1e6:.1f}M",
        delta="-1.5%",
        delta_color="inverse"  # Red for positive
    )
```

Advanced Plotly Charts

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Create subplots
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=('Top Satker', 'Metode Distribution',
                    'Monthly Trend', 'Pagu Distribution'),
    specs=[[{'type': 'bar'}, {'type': 'pie'}],
           [{'type': 'scatter'}, {'type': 'histogram'}]]
)

# Add traces
top_satker = df.groupby('nama_satker')['pagu'].sum().nlargest(10)
fig.add_trace(
    go.Bar(x=top_satker.index, y=top_satker.values, name='Satker'),
    row=1, col=1
)

metode_counts = df['metode_pengadaan'].value_counts()
fig.add_trace(
    go.Pie(labels=metode_counts.index, values=metode_counts.values),
    row=1, col=2
)
```

Advanced Plotly Charts (lanjutan)

```
# Update layout
fig.update_layout(height=800, showlegend=False)
st.plotly_chart(fig, use_container_width=True)
```

🔒 Error Handling

```
import streamlit as st

try:
    # Data loading
    df = pd.read_parquet('data.parquet')

    # Data processing
    filtered_df = df[df['pagu'] > threshold]

    if len(filtered_df) == 0:
        st.warning("⚠️ No data matches the filters")
    else:
        st.dataframe(filtered_df)

except FileNotFoundError:
    st.error("❌ Data file not found. Please check the path.")
except Exception as e:
    st.error(f"❌ An error occurred: {str(e)}")
    st.info("Please contact support if the problem persists.")

# Fallback data
if 'df' not in locals():
    st.warning("Loading sample data instead...")
    df = load_sample_data()
```

🚀 Deployment

Streamlit Cloud (Free!)

1. **Push to GitHub**

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <your-repo>
git push -u origin main
```

2. **Create `requirements.txt`**

```
streamlit==1.28.0
pandas
plotly
duckdb
```

🚀 Deployment (lanjutan)

3. **Deploy on Streamlit Cloud**
    - Go to https://share.streamlit.io
    - Connect GitHub repo
    - Select main file
    - Deploy!

🎯 Praktik Terbaik

Kecepatan

1. **Cache aggressively** 💾

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

2. **Use `use_container_width`** 📏

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

3. **Minimize reruns** 🔄
   - Use forms
   - Use session state
   - Disable auto-rerun for expensive operasi

🎯 Praktik Terbaik (lanjutan)

4. **Lazy loading** ⏱️
    - Load data only when needed
    - Use pagination for large tables

🎯 Praktik Terbaik (lanjutan)

UX/UI

1. **Clear structure** 🏗️
    - Logical flow
    - Grouped related elements
    - Use tabs/expanders
2. **Meaningful labels** 🏷️
    - Descriptive text
    - Help tooltips
    - Example

🎯 Praktik Terbaik (lanjutan)

3. **Feedback** 💬
    - Loading spinners
    - Success/error messages
    - Progress bars
4. **Responsive design** 📱
    - Test on different screens
    - Use columns wisely
    - Container widths

🎯 Praktik Terbaik (lanjutan)

Code Organization

```
# Good structure
import streamlit as st
import pandas as pd

# Configuration
st.set_page_config(...)

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

def create_filters():
    # Sidebar filters
    pass

def display_metrics(df):
    # KPI cards
    pass
```

🎯 Praktik Terbaik (lanjutan)

```
def main():
    # Main app logic
    df = load_data()
    filtered_df = create_filters()
    display_metrics(filtered_df)

if __name__ == "__main__":
    main()
```

🎯 Latihan Praktis

Latihan Akhir (Waktu 1 Jam)

Build Your Own Dashboard!

Requirements:

1. ✅ File uploader
2. ✅ Sidebar filters
3. ✅ KPI metrics (4 cards)
4. ✅ 3+ Plotly charts
5. ✅ Data table with export
6. ✅ Custom styling
7. ✅ Error handling
8. ✅ Caching

💡 Tips & Tricks

```python
# 1. Columns with different widths
col1, col2 = st.columns([2, 1])

# 2. Hide index in dataframe
st.dataframe(df, hide_index=True)

# 3. Sticky header
st.markdown(
    """
    <style>
    [data-testid="stHeader"] {
        background-color: rgba(255, 255, 255, 0.95);
    }
    </style>
    """, unsafe_allow_html=True
)

# 4. Remove "Made with Streamlit"
st.markdown(
    """
    <style>
    footer {visibility: hidden;}
    </style>
    """, unsafe_allow_html=True
)
```

💡 Tips & Tricks (lanjutan)

```
# 5. Wide mode by default
st.set_page_config(layout="wide")

# 6. Custom theme (.streamlit/config.toml)
[theme]
primaryColor="#1f77b4"
backgroundColor="#FFFFFF"
secondaryBackgroundColor="#f0f2f6"
textColor="#262730"
font="sans serif"

# 7. Secrets management (for API keys)
# .streamlit/secrets.toml
[database]
host = "localhost"
port = 5432

# Access in code
host = st.secrets["database"]["host"]
```

📚 Poin Penting

- ✅ Streamlit = Python → Web App (no web dev needed!)
- ✅ Rich widget library untuk interactivity
- ✅ Session state untuk persistent data
- ✅ Caching untuk performance
- ✅ Easy integration dengan Pandas, DuckDB, Plotly
- ✅ Free deployment di Streamlit Cloud
- ✅ Perfect untuk prototypes & internal tools
- ✅ Production-ready dengan proper error handling

From data to dashboard in hours, not weeks! 🚀

🔗 Resources

Documentation & Learning

- **Streamlit Docs:** https://docs.streamlit.io
- **Streamlit Gallery:** https://streamlit.io/gallery
- **Cheat Sheet:** https://docs.streamlit.io/library/cheatsheet
- **Forum:** https://discuss.streamlit.io
- **GitHub:** https://github.com/streamlit/streamlit

Deployment

- **Streamlit Cloud:** https://share.streamlit.io
- **Deployment Guide:** https://docs.streamlit.io/streamlit-community-cloud

🎉 Bootcamp Selesai!

Apa yang Sudah Anda Pelajari

Hari 1

- ✅ Python & Pandas untuk analisis data
- ✅ DuckDB untuk query analitik
- ✅ Visualisasi dengan Matplotlib, Seaborn, Plotly

Hari 2

- ✅ Data cleaning & perubahan
- ✅ Time series analysis
- ✅ Statistical testing
- ✅ Building interactive dashboards dengan Streamlit

🏆 Your Portfolio

Yang Sudah Anda Buat

- ✅ **7 Jupyter Notebooks**
  - EDA, DuckDB, Matplotlib/Seaborn, Plotly
  - Data Cleaning, Time Series, Statistical Analysis
- ✅ **4 Streamlit Apps**
  - Hello Streamlit, Components Demo
  - Data Explorer, RUP Dashboard
- ✅ **1 Production Dashboard**
  - Complete analytics platform untuk RUP 2025

Show this to employers! 💼

🚀 Next Steps

Lanjutkan Perjalanan Belajar Anda

1. **Build Projects** 💻
    - Find datasets you're interested in
    - Apply what you learned
    - Build your portfolio
2. **Deploy Your Apps** 🌐
    - Share your dashboards
    - Get feedback
    - Iterate and improve

🚀 Next Steps (lanjutan)

3. **Learn Advanced Topics** 📚
    - Machine Learning (scikit-learn)
    - Deep Learning (TensorFlow/PyTorch)
    - Big Data (Apache Spark)
    - Cloud Platforms (AWS, GCP, Azure)
4. **Join Communities** 👥
    - Kaggle competitions
    - Data science forums
    - Local meetups
    - Online communities

🚀 Next Steps (lanjutan)

5. **Kontribusi to Open Source** 🌟
    - GitHub projects
    - Documentation
    - Bug fixes
    - New features
6. **Build Network** 🤝
    - LinkedIn
    - Tech conferences
    - Workshops & webinars
    - Mentorship

🎯 Jalur Karir

Kemana Anda Bisa Pergi?

- **Data Analyst** - Business insights & reporting
- **Business Intelligence Analyst** - Dashboards & viz
- **Data Engineer** - Data pipelines & infrastructure
- **Data Scientist** - ML models & predictions
- **Analytics Engineer** - dbt, SQL, data modeling
- **ML Engineer** - Deploy ML models to production

This bootcamp gave you foundation for all of these! 🚀

💡 Tips Akhir

For success in Data Field

1. **Exercise Regularly** 🏋️
    - Consistency > intensity
    - Daily coding habit
2. **Build Portfolio** 📂
    - GitHub profile
    - Personal website
    - Blog posts

💡 Tips Akhir (lanjutan)

3. **Learn in Public** 📢
    - Share your projects
    - Write tutorials
    - Help others
4. **Stay Curious** 🔍
    - New tools & teknik
    - Industry trends
    - Best practices

📊 Statistik Bootcamp

Apa yang dibahas

- **Duration:** 16 hours (2 days)
- **Sessions:** 5 lengkap sessions
- **Notebooks:** 7 hands-on tutorials
- **Apps:** 4 Streamlit applications
- **Technologies:** Python, Pandas, DuckDB, Plotly, Streamlit
- **Dataset:** 16,430 real procurement records
- **Code Lines:** 2000+ lines of code written
- **Skills:** Data analysis, SQL, display, web apps

Intensive but worth it! 💪

📝 Feedback

We'd love to hear your feedback:

1. **What worked well?** ✅
2. **What could be improved?** 🔧
3. **What topics do you want more of?** 📚
4. **Would you recommend this to others?** ⭐

Your feedback helps us make better bootcamps!

🎉 CONGRATULATIONS!

You Did It! 🏆

Terima Kasih for Participating!

Keep Coding, Keep Learning, Keep Building!

Pertanyaan?? Let's discuss! 💬

🙏 Terima Kasih!

Good Luck with Your
Data Analytics Journey!

See You at Data World! 🌍

May your insights be meaningful
and your dashboards be beautiful! ✨

📚 Sumber Daya Lainnya

Materi Pembelajaran Tambahan

Books:

- **"Python for Data Analysis"** karya Wes McKinney (Pembuat Pandas)
- **"Streamlit for Data Science: Create Interactive Data Apps in Python"** karya Tyler Richards
- **"DuckDB in Action"** karya Mark Needham & Simon Aubury (MotherDuck)

Online Courses:

- Kaggle Learn
- Udemy
- DataCamp
- Coursera

📚 Sumber Daya Lainnya (lanjutan)

Communities:

- r/datascience
- r/Python
- Kaggle Forums
- Stack Overflow

Keep exploring! 🔍

📞 Tetap Terhubung

Resources & Support

- 👤 **Nama:** [Kurnia Ramadhan,ST.,M.Eng]
- 📧 **Email:** [kurnia@ramadhan.me]

We're here to support your journey! 🤝