Tool Comparison Report

Data Analytics and Reporting - Assignment 6

Student Name: Hemant Borana

Date: December 2025

Course: Data Analytics and Reporting (DAR)

Tool Comparison Report

Data Analytics and Reporting - Assignment 6

Student Name: Hemant Borana
Date: December 2025
Course: Data Analytics and Reporting (DAR)

Contents

Executive Summary

This report provides a comprehensive comparison of the visualization and reporting tools used in this assignment: **Python (Plotly, Streamlit)** and **Tableau Public**. The analysis covers strengths, limitations, use cases, and recommendations based on hands-on implementation experience.

1. Tools Overview

1.1 Python Stack (Plotly + Streamlit)

Components:

- Plotly for interactive visualizations
- Streamlit for dashboard framework
- Pandas for data manipulation
- NumPy for calculations

Version Information:

- Python 3.x
- Plotly 5.x
- Streamlit 1.x
- Pandas 2.x

1.2 Tableau Public

Description: Desktop-based business intelligence tool with drag-and-drop interface for creating interactive dashboards and visualizations.

Version: Tableau Public 2024.x

2. Detailed Comparison

2.1 Ease of Use

Python (Plotly + Streamlit)

Strengths:

- Complete programmatic control over every element
- Reproducible workflows through code
- Version control friendly
- Easy integration with data processing pipelines

Limitations:

- Requires programming knowledge
- Steeper learning curve for non-technical users
- More time needed for initial setup
- Debugging can be time-consuming

Tableau Public

Strengths:

- Intuitive drag-and-drop interface
- Minimal learning curve for basic visualizations
- Quick prototype development
- User-friendly for business analysts

Limitations:

- Less flexibility for custom requirements
- Limited control over fine details
- Geographic data recognition issues
- Dependency on GUI interaction

Winner: Tableau for speed, Python for control

2.2 Visualization Capabilities

Python (Plotly + Streamlit)

Strengths:

- 40+ built-in chart types
- Unlimited customization potential
- Advanced statistical visualizations

- Custom animation support
- Integration with scientific libraries

Achievements in this project:

- 15 different chart types implemented
- Custom color schemes and themes
- Interactive hover tooltips
- Zoom, pan, and export features
- Dual-axis charts with synchronized scales

Limitations:

- Requires manual coding for each feature
- More verbose for simple charts
- Performance optimization needed for large datasets

Tableau Public

Strengths:

- Built-in geographic mapping
- Automatic chart type suggestions
- Quick dashboard assembly
- Native drill-down capabilities
- Polished default styling

Achievements in this project:

- Executive dashboard with KPI cards
- Operational dashboard with multiple views
- Trend analysis with dual axes
- Category performance bars

Limitations:

- Limited customization beyond presets
- Some chart types require workarounds
- Map functionality had compatibility issues
- Cannot integrate custom Python functions

Winner: Python for variety and customization, Tableau for standard business charts

2.3 Interactivity Features

Python (Plotly + Streamlit)

Interactive Features Implemented:

- Real-time filtering with sidebars
- Multi-select dropdowns
- Date range pickers
- Slider-based parameter controls
- Dynamic chart updates
- What-if scenario modeling
- Drill-down navigation through tabs
- Export functionality

Technical Approach:

- State management through session variables
- Callback functions for user interactions
- Reactive programming model
- Custom event handlers

Limitations:

- Requires refresh for some interactions
- Session state management complexity
- Performance considerations with multiple widgets

Tableau Public

Interactive Features Implemented:

- Click-through filters
- Hover tooltips
- Dashboard actions
- Filter cascading
- Navigation buttons

Limitations:

- Limited to built-in interaction types
- Cannot create custom interaction logic
- Filter actions sometimes unpredictable

- No programmatic control

Winner: Python for advanced interactivity, Tableau for standard interactions

2.4 Performance

Python (Plotly + Streamlit)

Performance Characteristics:

- Good performance with datasets under 50,000 rows
- Caching mechanisms improve load times
- Client-side rendering can be slow for complex charts
- Memory efficient with proper data handling

Optimization Techniques Used:

- Data caching with @st.cache_data decorator
- Pre-aggregation of large datasets
- Selective rendering based on user selections
- Lazy loading of visualizations

Observed Performance:

- Initial load: 2-3 seconds
- Filter application: Under 1 second
- Chart refresh: Instant to 1 second

Tableau Public

Performance Characteristics:

- Excellent with datasets up to 1 million rows
- Native data engine optimization
- Fast rendering for standard visualizations
- Efficient memory usage

Observed Performance:

- Initial load: 1-2 seconds
- Filter application: Instant
- Dashboard navigation: Instant

Winner: Tableau for raw performance, Python for optimization control

2.5 Data Handling

Python (Plotly + Streamlit)

Capabilities:

- Direct connection to databases
- API integration
- Real-time data streaming
- Complex data transformations
- Multiple data source merging

Data Processing in Project:

- Generated 4 synthetic datasets
- Performed aggregations and pivots
- Calculated custom metrics
- Date parsing and formatting
- Cross-dataset analysis

Tableau Public

Capabilities:

- Multiple data source types (CSV, Excel, databases)
- Data blending across sources
- Calculated fields
- Geographic role assignment

Data Processing in Project:

- Loaded CSV files
- Created calculated fields for metrics
- Built hierarchies for drill-downs
- Applied filters and aggregations

Winner: Python for complex transformations, Tableau for standard business logic

2.6 Collaboration and Sharing

Python (Plotly + Streamlit)

Sharing Options:

- Deploy to Streamlit Cloud (free)
- Self-hosted on any server

- Export to static HTML
- Embed in web applications
- Share code via GitHub

Collaboration:

- Version control through Git
- Code review processes
- Team development workflows
- Documentation as code

Tableau Public

Sharing Options:

- Publish to Tableau Public server
- Embed in websites via iframe
- Download as packaged workbook
- Share via URL

Collaboration:

- Workbook sharing
- Limited version control
- Comment functionality
- Subscription for automatic updates

Winner: Python for developer collaboration, Tableau for business user sharing

2.7 Cost Considerations

Python (Plotly + Streamlit)

Costs:

- Free and open source
- No licensing fees
- Hosting costs for deployment (optional)
- Development time investment

Total Cost of Ownership:

- Low monetary cost
- High initial time investment
- Lower maintenance for experienced developers

Tableau Public

Costs:

- Free for public data
- Tableau Desktop: $70/user/month (if needed)
- Tableau Server: Enterprise pricing
- Training costs

Total Cost of Ownership:

- Free for this project
- Would require licensing for private data
- Lower initial time investment

Winner: Python for budget constraints, Tableau for quick ROI

3. Use Case Recommendations

3.1 When to Use Python (Plotly + Streamlit)

Ideal For:

- Custom analytics applications
- Data science projects requiring statistical analysis
- Integration with machine learning models
- Real-time data monitoring dashboards
- Applications requiring complex business logic
- Projects needing version control
- Situations requiring full customization
- Development teams with programming skills

Example Scenarios:

- Predictive analytics dashboard
- IoT sensor monitoring system
- Custom reporting for unique business processes
- Research and academic analysis tools

3.2 When to Use Tableau

Ideal For:

- Standard business intelligence reporting
- Quick dashboard prototyping
- Non-technical user self-service analytics
- Geographic data visualization
- Enterprise-wide reporting standards
- Executive dashboards
- Department-level analytics
- Ad-hoc analysis by business users

Example Scenarios:

- Monthly sales performance reports
- Executive KPI monitoring
- Regional sales comparison
- Standard financial reporting

3.3 Hybrid Approach

Recommended Strategy:

- Use Tableau for standard business dashboards
- Use Python for custom analytics and data science
- Tableau for presentation layer
- Python for data preparation and advanced analysis
- Integrate both tools in enterprise architecture

4. Implementation Experience

4.1 Development Time

Python Implementation:

- Chart creation: 70 minutes
- Table development: 45 minutes
- Interactive dashboard: 60 minutes
- Total: 175 minutes

Tableau Implementation:

- Dashboard creation: 90 minutes
- Learning curve adjustments: 20 minutes
- Total: 110 minutes

4.2 Challenges Encountered

Python:

- Date formatting for time series required careful handling
- State management in Streamlit needed planning
- Performance optimization for multiple charts
- CSS styling for professional appearance

Tableau:

- Geographic data recognition issues with state names
- Map visualization did not render properly
- Limited customization for specific requirements
- Filter actions needed multiple attempts

4.3 Lessons Learned

Key Insights:

- Python offers maximum flexibility at the cost of development time
- Tableau accelerates standard dashboard creation
- Both tools have their place in analytics toolkit
- Tool selection should match team skills and project requirements
- Hybrid approaches leverage strengths of both platforms

5. Technical Capabilities Matrix

Feature	Python (Plotly)	Tableau	Winner
Learning Curve	Steep	Gentle	Tableau
Customization	Excellent	Good	Python
Chart Variety	40+ types	30+ types	Python
Interactivity	Advanced	Standard	Python
Performance	Good	Excellent	Tableau
Deployment	Flexible	Simple	Tie
Cost	Free	Free/Paid	Python
Collaboration	Developer-focused	Business-focused	Depends
Data Processing	Unlimited	Good	Python
Geographic Maps	Good	Excellent	Tableau
Mobile Support	Good	Excellent	Tableau
API Integration	Excellent	Limited	Python

6. Recommendations

6.1 For This Project

Conclusion: The hybrid approach using both Python and Tableau was optimal for this assignment.

Rationale:

- Tableau provided quick dashboard prototypes demonstrating traditional BI capabilities
- Python enabled advanced interactive features and drill-down analysis
- Combination showcased versatility and tool knowledge
- Met all assignment requirements effectively

7. Conclusion

Both Python (Plotly + Streamlit) and Tableau are powerful tools with distinct advantages. Python excels in customization, flexibility, and integration with data science workflows. Tableau provides superior ease of use, performance, and standard business intelligence features.

The choice between tools should be driven by:

- Team technical capabilities
- Project requirements
- Time constraints
- Budget considerations
- Maintenance and scalability needs

For this assignment, the combination of both tools provided comprehensive experience and met all evaluation criteria while demonstrating real-world analytics capabilities.

Thank You

Thank you for your time and consideration.

Contact Information:
Phone: 9284494154
Email: hemantpb123@gmail.com