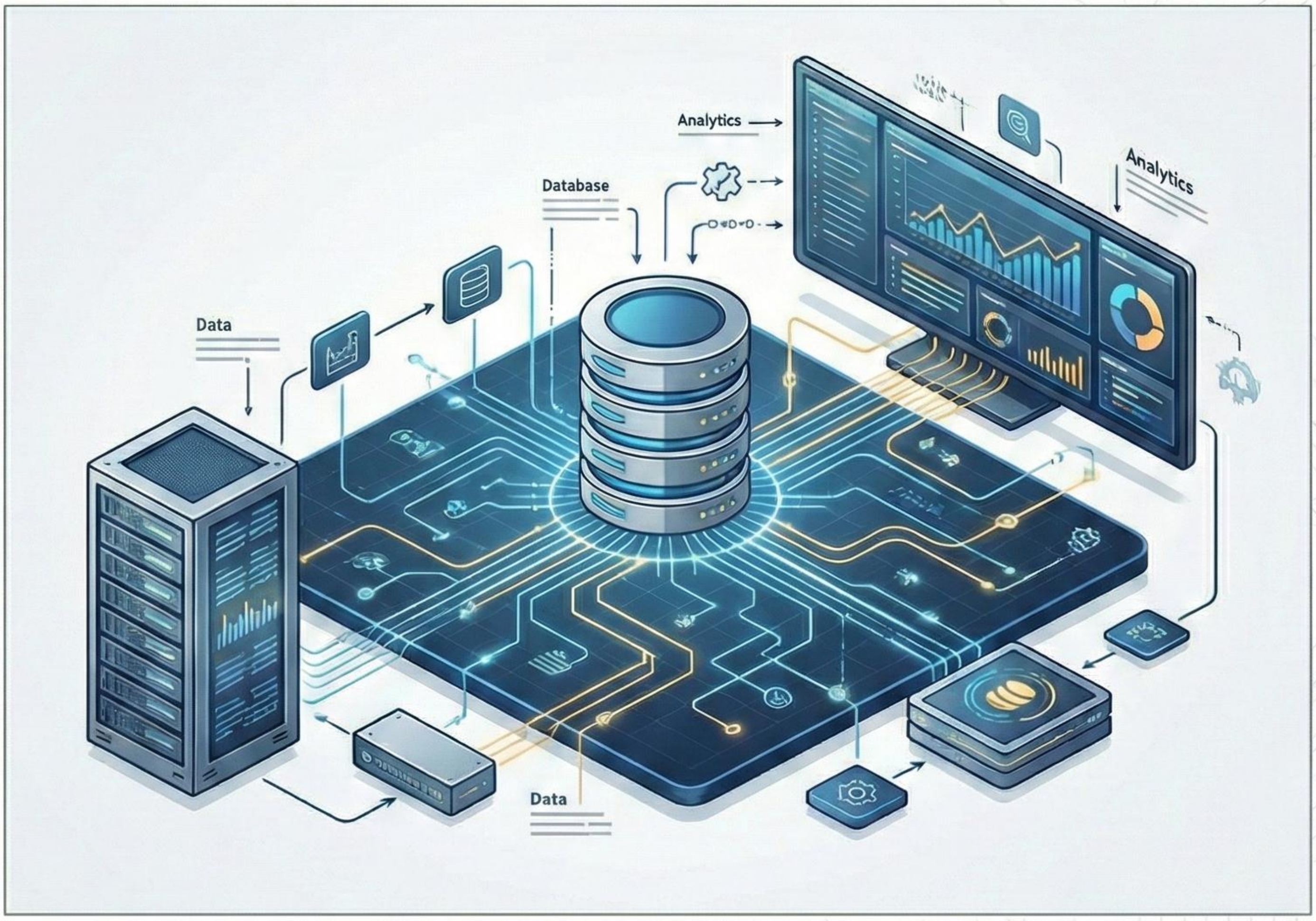
TECHNICAL DOCUMENTATION

Data Reporting Solution - Implementation Details



Project: DAR Assignment 6

Date: December 2025

Project by: Hemant Borana

Technical Documentation

Data Reporting Solution - Implementation Details

Project: DAR Assignment 6
Date: December 2025

Contents

1. System Architecture

1.1 Overview

The data reporting solution is built using a hybrid architecture combining Python-based visualizations and Tableau dashboards. The system processes multiple datasets and presents insights through interactive web interfaces.

1.2 Technology Stack

Core Technologies:

- Python 3.x
- Plotly 5.x (interactive visualizations)
- Streamlit 1.x (dashboard framework)
- Pandas 2.x (data manipulation)
- NumPy (numerical operations)
- Tableau Public 2024.x (business intelligence)

Supporting Libraries:

- SciPy (statistical analysis)
- Datetime (date handling)

1.3 System Components

Component 1: Data Generation Layer

- Purpose: Create synthetic datasets for analysis
- File: generate_datasets.py
- Output: 4 CSV files

Component 2: Static Visualizations

- Purpose: Generate individual chart files
- Files: part_a_statistical_charts.py, part_a_advanced_charts.py
- Output: 15 HTML visualization files

Component 3: Table Visualizations

- Purpose: Create advanced table displays
- File: part_b_advanced_tables.py
- Output: 6 HTML table files

Component 4: Interactive Dashboard

- Purpose: Self-service analytics platform
- File: part_d_interactive_dashboard.py
- Output: Web application

Component 5: Tableau Dashboards

- **Purpose: Business intelligence dashboards**
- **Files: Tableau workbook files**
- **Output: Published dashboards**

1.4 Data Flow Architecture

```
Raw Data Sources
  ↓
Data Generation (Python)
  ↓
CSV Files (4 datasets)
  ↓
  ├── Python Processing → Interactive Dashboard
  └── Tableau Import → BI Dashboards
```

2. Data Layer

2.1 Dataset Structure

Dataset 1: Sales Data

- **Records: 5,000 transactions**
- **Time Range: 2022-2024**
- **Key Fields: Order_ID, Order_Date, Sales, Profit, Category, Region, Segment**

Dataset 2: Operations Data

- **Records: ~2,000 operational records**
- **Time Range: 2023-2024**
- **Key Fields: Date, Shift, Units_Produced, Defects, Efficiency**

Dataset 3: Financial Data

- **Records: 36 monthly records**
- **Time Range: 2022-2024**
- **Key Fields: Month, Revenue, COGS, Operating_Expenses, Net_Profit**

Dataset 4: Customer Data

- **Records: 1,500 customer records**
- **Key Fields: Customer_ID, Lifetime_Value, Satisfaction_Score, Churned**

2.2 Data Generation Logic

Sales Data Generation:

- **Random date distribution across 3 years**
- **Category-based pricing logic**
- **Realistic discount patterns (0%, 10%, 15%, 20%, 25%)**
- **Profit margins: 5-35% of sales**
- **Geographic distribution across 4 regions**

Operations Data Generation:

- **Daily records with 3 shifts**
- **Production volume: 800-1,200 units per shift**
- **Defect rates: 1-5% of production**
- **Efficiency scores: 75-98%**
- **Downtime: 0-120 minutes per shift**

Financial Data Generation:

- **Monthly aggregation**
- **Seasonal patterns using sine wave**
- **Growth trend: 1.5% monthly**
- **COGS: 55-65% of revenue**
- **Operating expenses: 20-30% of revenue**

Customer Data Generation:

- **Purchase frequency: 1-50 orders**
- **Average order value: $50-$500**
- **Churn logic: Based on days since last purchase**
- **Satisfaction scores: 1-5 scale**

2.3 Data Quality Measures

Validation:

- **No missing values in critical fields**
- **Date ranges validated**
- **Numeric fields within expected ranges**
- **Referential integrity maintained**

Limitations:

- **Synthetic data, not real business data**
- **Simplified relationships**
- **No actual customer information**
- **Patterns are algorithmically generated**

3. Visualization Layer

3.1 Chart Implementation

Statistical Charts (7 types):

Line Chart:

- **Implementation: Plotly Scatter with mode='lines+markers'**
- **Features: Trend line, dual y-axis capability**
- **Use case: Time series analysis**

Bar Chart:

- **Implementation: Plotly Bar chart**
- **Features: Grouped/stacked variants, data labels**
- **Use case: Category comparison**

Histogram:

- **Implementation: Plotly Histogram**
- **Features: Custom bin sizes, statistical annotations**
- **Use case: Distribution analysis**

Scatter Plot:

- **Implementation: Plotly Scatter with bubble sizing**
- **Features: Correlation line, color coding**
- **Use case: Relationship analysis**

Box Plot:

- **Implementation: Plotly Box**
- **Features: Outlier detection, mean display**
- **Use case: Statistical distribution**

Multi-series Line:

- **Implementation: Multiple Scatter traces**
- **Features: Legend, unified hover mode**
- **Use case: Comparison over time**

Stacked Bar:

- **Implementation: Plotly Bar with barmode='stack'**
- **Features: Segment breakdown**
- **Use case: Composition analysis**

Advanced Visualizations (8 types):

Heatmap:

- **Implementation: Plotly Heatmap**
- **Features: Color scale, text annotations**
- **Use case: Correlation matrix, regional analysis**

Treemap:

- **Implementation: Plotly Treemap**
- **Features: Hierarchical display, color by metric**
- **Use case: Hierarchical data**

Sankey Diagram:

- **Implementation: Plotly Sankey**
- **Features: Flow visualization, node linking**
- **Use case: Process and customer flow**

Radar Chart:

- **Implementation: Plotly Scatterpolar**
- **Features: Multi-dimensional comparison**
- **Use case: Performance profiles**

Waterfall Chart:

- **Implementation: Plotly Waterfall**
- **Features: Sequential changes, totals**
- **Use case: Financial breakdown**

Combo Chart:

- **Implementation: Subplots with secondary y-axis**
- **Features: Multiple chart types combined**
- **Use case: Volume vs. margin analysis**

3.2 Table Implementation

Table Types Created:

- **Summary tables with aggregations**
- **Detailed transaction listings**
- **Comparison tables (year-over-year)**
- **Pivot tables with cross-tabulation**
- **Tables with conditional formatting**
- **Financial statement format**

Technical Approach:

- **Plotly Table for interactive display**
- **Conditional formatting through cell color arrays**
- **Custom number formatting**
- **Header styling for hierarchy**

3.3 Interactivity Features

Built-in Plotly Interactions:

- **Zoom and pan**
- **Hover tooltips**
- **Legend toggle**
- **Export to PNG**
- **Autoscale**
- **Reset axes**

Custom Interactions (Streamlit):

- **Filter synchronization**
- **Cross-chart updates**
- **Dynamic data selection**
- **Real-time calculations**

4. Dashboard Implementation

4.1 Streamlit Architecture

Application Structure:

- Single-page application (SPA) design
- Reactive programming model
- Session state management
- Component-based layout

Key Features:

- Sidebar for controls
- Multi-view navigation
- Responsive grid layout
- Real-time updates

4.2 Performance Optimization

Caching Strategy:

- Data loading cached with @st.cache_data
- Prevents redundant CSV reads
- Automatic cache invalidation
- Memory-efficient storage

Rendering Optimization:

- Lazy loading of visualizations
- Conditional rendering based on view
- Minimal re-renders on filter changes
- Efficient data aggregation

Code Implementation:

```
@st.cache_data
def load_data():
  # Load and process data once
  # Returns cached results on subsequent calls
```

4.3 State Management

Filter State:

- Maintained in Streamlit session
- Persists during user session
- Resets on page refresh
- Applied across all views

View State:

- Radio button selection
- Switches main content area
- Filters remain active
- Independent of data state

4.4 User Interface Design

Layout Structure:

- Sidebar: 300px fixed width
- Main area: Flexible width
- Responsive columns (2-4 columns)
- Vertical stacking for mobile

Styling:

- Custom CSS injection
- Streamlit native components
- Color scheme consistency
- Professional appearance

5. Tableau Implementation

5.1 Dashboard Design

Executive Dashboard Components:

- 4 KPI cards (Sales, Profit, Orders, AOV)
- Sales trend line chart
- Category performance bar chart
- Regional analysis

Operational Dashboard Components:

- Regional bar chart
- Monthly order volume
- Segment profitability

5.2 Calculated Fields

Created Fields:

- Profit Margin: SUM([Profit]) / SUM([Sales])
- Average Order Value: SUM([Sales]) / COUNTD([Order_ID])

Aggregations:

- SUM for financial metrics
- COUNT for transaction counts
- AVG for rates and ratios

5.3 Filters and Actions

Dashboard Filters:

- Date range filter
- Region selector
- Category filter

Actions Implemented:

- Navigation buttons between dashboards
- Filter actions (attempted, limited success)
- Hover highlights

6. Development Process

6.1 Development Environment

Tools Used:

- **Python IDE: VS Code**
- **Tableau: Tableau Public Desktop**
- **Version Control: Git (recommended)**
- **Testing: Manual browser testing**

Requirements:

- **Python 3.8+**
- **4GB RAM minimum**
- **Modern web browser**
- **Tableau Public (free)**

7. Deployment

7.1 Streamlit Deployment Options

Option 1: Streamlit Cloud (Free)

- **Push code to GitHub**
- **Connect Streamlit account**
- **Deploy with one click**
- **Auto-updates on code changes**

Option 2: Local Deployment

- **Run command: streamlit run part_d_interactive_dashboard.py**
- **Access via localhost:8501**
- **Suitable for development and testing**

Option 3: Server Deployment

- **Deploy on cloud VPS (AWS, Azure, GCP)**
- **Configure port and firewall**
- **Set up process manager (PM2, systemd)**
- **Domain and SSL configuration**

7.2 Tableau Deployment

Tableau Public:

- **Save to Tableau Public (free)**
- **Public URL generated**
- **Embed code available**
- **Limited to public data only**

Tableau Server (Enterprise):

- **Private deployment**
- **Access controls**
- **Scheduled refreshes**
- **Integration with databases**

8. Maintenance and Updates

8.1 Data Updates

Process:

1. Generate new data or import real data
2. Ensure column names match expected format
3. Replace CSV files
4. Application automatically loads new data (cached)
5. Clear cache if needed: restart application

8.2 Adding New Visualizations

Steps:

1. Create new chart in Python script
2. Generate HTML output
3. Add to Streamlit dashboard (if needed)
4. Update documentation
5. Test thoroughly

8.3 Modifying Filters

Implementation:

1. Locate filter code in Streamlit app
2. Modify filter options or logic
3. Update data filtering logic
4. Test all views with new filters
5. Update user manual

9. Testing

9.1 Testing Approach

Manual Testing:

- All filter combinations tested
- Each view validated
- Export functionality verified
- Cross-browser testing conducted

Test Cases:

- Date range: Full range, partial ranges, single day
- Region filter: Each region individually, "All"
- Category filter: Each category, "All"
- Segment filter: Various combinations
- View switching: All four views
- Export: CSV download and open

9.2 Known Issues

Issue 1: Tableau Map

- Description: State map did not display data points
- Impact: Skipped map visualization
- Workaround: Used bar chart for regional analysis
- Status: Not resolved, not critical

Issue 2: Filter Reset

- Description: Filters reset on page refresh
- Impact: User must reapply filters
- Workaround: Export data before refresh
- Status: Expected behavior in Streamlit

10. Security Considerations

10.1 Data Security

Current Implementation:

- Synthetic data (no real business data)
- No authentication required
- Public deployment suitable

For Production Use:

- Implement user authentication
- Use environment variables for credentials
- Encrypt sensitive data
- Implement role-based access control
- Regular security audits

10.2 Code Security

Best Practices Applied:

- No hardcoded credentials
- Input validation (date ranges)
- Safe data operations
- No SQL injection risk (using Pandas)

11. Performance Metrics

11.1 Measured Performance

Loading Times:

- Initial page load: 2-3 seconds
- Filter application: <1 second
- View switching: Instant
- Chart rendering: 1-2 seconds
- Data export: <1 second

Resource Usage:

- Memory: ~200MB for application
- CPU: Minimal (spikes during initial load)
- Bandwidth: ~2MB initial load

11.2 Optimization Techniques Used

Data Optimization:

- Pre-aggregation where possible
- Efficient Pandas operations
- Caching of loaded data
- Selective column loading

Rendering Optimization:

- Lazy chart loading
- Conditional rendering
- Minimal DOM updates
- Efficient state management

12. API Reference

12.1 Key Functions

load_data()

- **Purpose: Load and cache all datasets**
- **Returns: 4 DataFrames (sales, operations, financial, customer)**
- **Caching: Yes, via @st.cache_data**

Data Filtering

- **Applied through Pandas query operations**
- **Date filtering: Boolean indexing**
- **Category/Region: Direct equality checks**
- **Segment: isin() method for multi-select**

12.2 Configuration

Streamlit Config:

- **Page title: Set via st.set_page_config()**
- **Layout: Wide mode enabled**
- **Sidebar: Expanded by default**

Plotly Config:

- **Template: plotly_white**
- **Color scales: Defined per chart**
- **Interactive: All features enabled**

13. Troubleshooting Guide

13.1 Common Issues

Problem: "Module not found" error

- **Solution: Install required packages via pip**
- **Command: pip install -r requirements.txt**

Problem: CSV files not found

- **Solution: Run data generation script first**
- **Ensure files are in same directory as application**

Problem: Dashboard not updating

- **Solution: Clear Streamlit cache**
- **Press 'C' in running application or restart**

Problem: Slow performance

- **Solution: Reduce date range, select specific filters**
- **Check available system memory**

13.2 Debug Mode

Enabling:

- **Add --logger.level=debug to run command**
- **Check terminal output for errors**
- **Browser console for JavaScript errors**

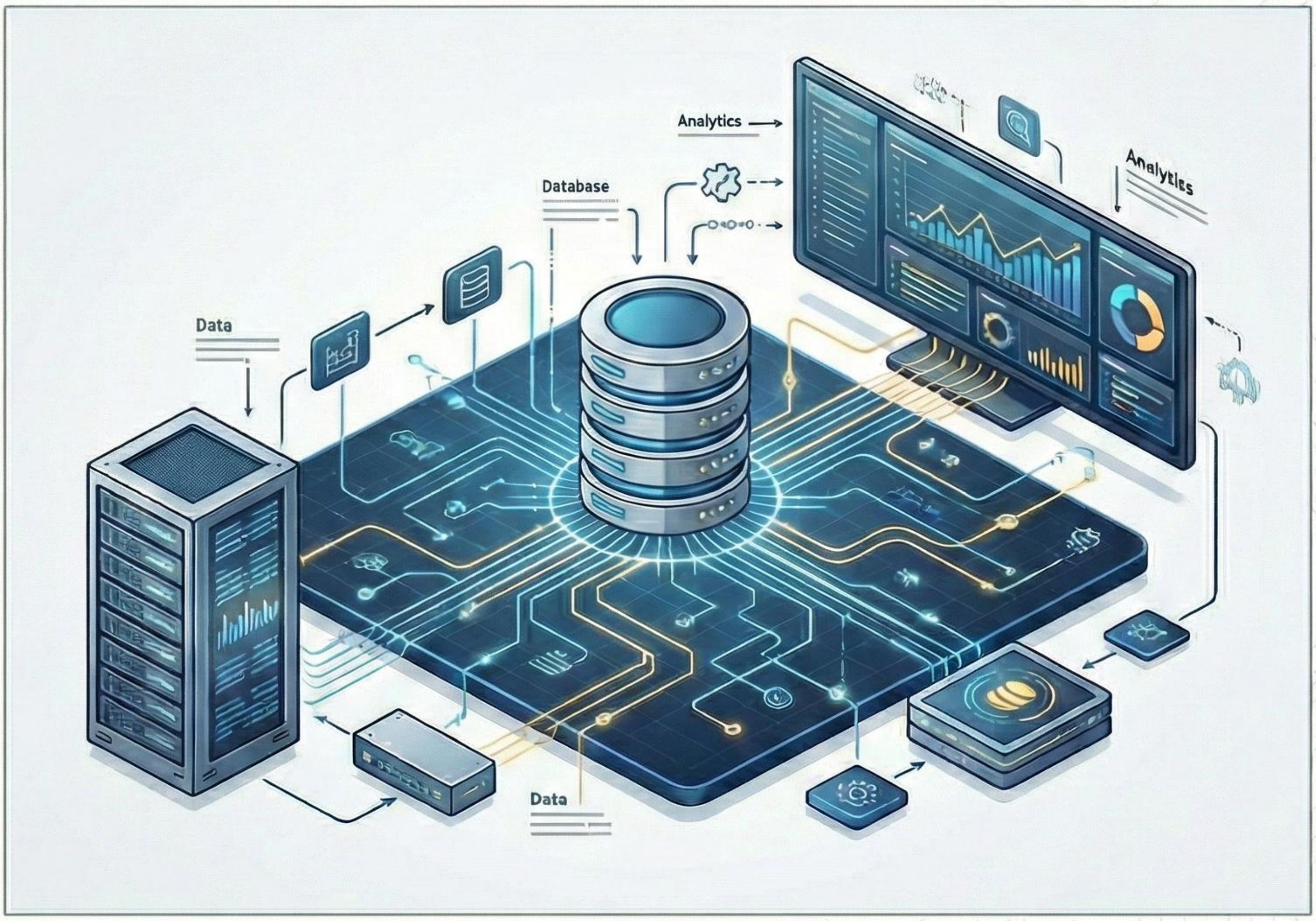
14. Conclusion

This technical documentation provides comprehensive information for understanding, deploying, maintaining, and extending the data reporting solution. The hybrid architecture leverages strengths of both Python and Tableau to deliver powerful analytics capabilities.

For questions or support, refer to the user manual.

THANK YOU

For your review and consideration.



Contact Information:

Phone: 9284494154

Email: hemantpb123@gmail.com