IT Management & Audits

Practical Lab Manual

Business Intelligence Dashboard Builder

Practical P08

Learning Domain

Business Intelligence & Self-Service Analytics

Course Learning Outcomes

CLO08: Build interactive business intelligence dashboards from financial data

Unit

Unit V: Business Intelligence & Analytics

Time Allocation: 3 hours

Learning Mode: Hands-on (80%) + Theory (20%)

Difficulty Level: Intermediate

Business Intelligence Dashboard Builder

Practical P08

Quick Reference

Practical Code	P08
Practical Name	Business Intelligence Dashboard Builder
Slot	T/P-8
Duration	3 hours
CLO Mapping	CLO08
Unit	Unit V: Business Intelligence & Analytics
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	MEDIUM
Screenshot Count	5 Required

Prerequisites

- Basic data analysis concepts (mean, median, aggregation, filtering)
- Python programming basics (variables, functions, loops, dictionaries)
- Understanding of Key Performance Indicators (KPIs) and business metrics
- Familiarity with CSV data files and tabular data structures
- Web browser with developer tools (Chrome or Firefox recommended)

Tools Required

Tool	Version	Free	Notes
Python	3.8+	✓	Core runtime
Flask	2.x+	✓	Web framework
Plotly	5.x+	✓	Interactive charting
pandas	1.x+	✓	Data manipulation
pip	Latest	✓	Package manager
Web Browser	Latest	✓	Chrome/Firefox

Learning Objectives

- ✓ Set up a Python-based BI dashboard application with Flask and Plotly
- ✓ Upload and process financial datasets with automatic column type detection
- ✓ Design and build interactive KPI cards for key business metrics
- ✓ Create multiple chart types (line, bar, pie) for data visualization
- ✓ Configure interactive filters including date range, category, and drill-down
- ✓ Export dashboards as standalone HTML files for offline viewing
- ✓ Customize dashboard appearance, KPI calculations, and color schemes

What You Will Learn

By the end of this practical, you will:

1. Understand core Business Intelligence concepts including ETL, OLAP, and data warehousing
2. Build a self-service analytics dashboard that empowers non-technical users
3. Process raw CSV data into meaningful KPI metrics using pandas
4. Create interactive Plotly charts embedded in a Flask web application
5. Design dashboards following information hierarchy and Gestalt principles
6. Export complete dashboards as portable standalone HTML files

Real-World Application

Business Intelligence dashboards are central to data-driven decision-making across industries. In India, companies like **Flipkart** use BI dashboards for seller performance analytics, while **Amazon India** tracks marketplace health through real-time KPI dashboards. The **GST Network (GSTN)** uses analytics dashboards to monitor tax compliance across states, and **NPCI** tracks UPI transaction volumes through interactive dashboards processing billions of records monthly.

Hands-On Procedure

Part A: Setup

Step 1: Clone the BI Dashboard Builder Repository

Objective: Download the project repository and understand the application structure.

1. Open a terminal and navigate to your preferred working directory
2. Clone the bi-dashboard-builder repository
3. Explore the project directory structure

```
# Clone the repository
git clone https://github.com/itma-labs/bi-dashboard-builder.git
cd bi-dashboard-builder
```

Clone and Explore Repository

Project Structure:

```
bi-dashboard-builder/
  app/
    main.py               # Flask application entry point
    data_processor.py     # CSV parsing and type detection
    chart_generator.py    # Plotly chart creation engine
    kpi_calculator.py     # KPI computation logic
    templates/            # Jinja2 HTML templates
      index.html          # Upload page
      dashboard.html      # Dashboard display page
      export.html         # Export template
    static/               # CSS, JS, images
  templates/              # Dashboard configuration files
    sales_dashboard.json
    finance_dashboard.json
  data/                   # Sample datasets
    sample_sales.csv
    sample_financial.csv
  requirements.txt
```

Repository Directory Layout

Expected Output

Repository cloned successfully. Directory structure visible with app/, templates/, data/ folders.

Step 2: Create Virtual Environment and Install Dependencies

Objective: Set up an isolated Python environment and install all required packages.

```
# Create and activate virtual environment
python -m venv venv
# Linux/Mac: source venv/bin/activate
# Windows: venv\Scripts\activate

# Install dependencies
pip install flask plotly pandas
# Or: pip install -r requirements.txt

# Verify installations
python -c "import flask; print('Flask:', flask.__version__)"
python -c "import plotly; print('Plotly:', plotly.__version__)"
python -c "import pandas; print('pandas:', pandas.__version__)"

# Start the application
python app/main.py
```

Virtual Environment Setup

Expected Output

```
Flask: 2.3.x | Plotly: 5.x.x | pandas: 2.x.x
* Running on http://127.0.0.1:5000
```

Ensure you activate the virtual environment every time you open a new terminal. Look for the `(venv)` prefix in your prompt. If you see `ModuleNotFoundError`, the venv is likely not activated.

Screenshot 1

What to paste: The Flask application running in your browser showing the file upload interface at `http://localhost:5000` with the upload form and supported file format indicators.

Paste your screenshot here

Part B: Data Upload & Processing

Step 3: Upload Sample Data and Launch the Dashboard

Objective: Upload a CSV dataset and observe automatic column type detection.

1. Open `http://localhost:5000` in your browser
2. Upload `data/sample_sales.csv` via the file upload area
3. Click **Upload & Analyze**
4. Observe auto-detected column types (numeric, categorical, date)

```
# sample_sales.csv columns:
# date,product,category,region,quantity,unit_price,revenue
2024-01-01,Widget A,Electronics,North,150,299.99,44998.50
2024-01-01,Widget B,Clothing,South,200,149.99,29998.00
2024-01-02,Widget C,Electronics,East,100,499.99,49999.00
```

Sample Sales Data Preview

Expected Output

Auto-detected column types:
`date -> datetime | product -> categorical | category -> categorical`
`region -> categorical | quantity -> numeric | unit_price -> numeric`
`revenue -> numeric`
Total rows: 1,200 | Total columns: 7

Step 4: Explore Data Processing and Suggested Visualizations

Objective: Review auto-detected types, suggested chart types, and aggregation options.

1. Review the **Data Summary** panel after upload
2. Examine **Suggested Visualizations**: line charts for date+numeric, bar charts for categorical+numeric, pie charts for categorical distributions
3. Explore **Aggregation Options**: Sum, Average, Count, Min, Max for numeric columns; Group-by for categorical columns; daily/weekly/monthly for date columns
4. Note data quality indicators (missing values, outliers)

```
import pandas as pd

def process_uploaded_file(filepath):
    df = pd.read_csv(filepath)
    column_info = {}
    for col in df.columns:
        dtype = str(df[col].dtype)
```

```
        if 'int' in dtype or 'float' in dtype:
            col_type = 'numeric'
        else:
            try:
                pd.to_datetime(df[col], infer_datetime_format=True)
                col_type = 'datetime'
                df[col] = pd.to_datetime(df[col])
            except (ValueError, TypeError):
                col_type = 'categorical'
        column_info[col] = {
            'type': col_type,
            'unique_count': df[col].nunique(),
            'missing_count': df[col].isnull().sum()
        }
    return df, column_info
```

Data Processing Logic (data_processor.py)

Expected Output

Suggested: Line (revenue over time), Bar (revenue by product/category/region), Pie (category distribution). Aggregation: Sum, Average, Count, Min, Max. Group-by: product, category, region.

Screenshot 2

What to paste: The data processing results page showing auto-detected column types, data summary statistics, and suggested visualizations after uploading `sample_sales.csv`.

Paste your screenshot here

Part C: Dashboard Building

Step 5: Configure the Sales Dashboard

Objective: Build an interactive sales dashboard with KPI cards and multiple chart types.

1. Select **Sales Dashboard** template (`templates/sales_dashboard.json`)
2. Configure KPI cards: **Total Revenue** (sum), **Average Order Value** (mean), **Growth Rate** (period-over-period %)
3. Configure charts: **Revenue Trend** (line), **Top Products** (horizontal bar), **Category Distribution** (pie/donut)
4. Click **Build Dashboard** and interact with charts (hover tooltips, legend toggles)

```
{
  "name": "Sales Performance Dashboard",
  "layout": "grid-3col",
  "kpis": [
    {"title": "Total Revenue", "column": "revenue",
     "aggregation": "sum", "format": "currency_inr"},
    {"title": "Average Order Value", "column": "revenue",
     "aggregation": "mean", "format": "currency_inr"},
    {"title": "Growth Rate", "column": "revenue",
     "aggregation": "growth_pct", "format": "percentage"}
  ],
  "charts": [
    {"type": "line", "title": "Revenue Trend",
     "x": "date", "y": "revenue", "time_group": "monthly"},
    {"type": "bar", "title": "Top Products by Revenue",
     "x": "product", "y": "revenue", "sort": "descending"},
    {"type": "pie", "title": "Revenue by Category",
     "labels": "category", "values": "revenue", "hole": 0.4}
  ]
}
```

Sales Dashboard Template (sales_dashboard.json)

```
def calculate_kpi(df, config):
    col, agg = config['column'], config['aggregation']
    if agg == 'sum':
        value = df[col].sum()
    elif agg == 'mean':
        value = df[col].mean()
    elif agg == 'growth_pct':
        df_sorted = df.sort_values('date')
        mid = len(df_sorted) // 2
        prev = df_sorted.iloc[:mid][col].sum()
        curr = df_sorted.iloc[mid:][col].sum()
        value = ((curr - prev) / prev) * 100 if prev else 0
    return {'title': config['title'], 'value': value}

def format_kpi_value(value, fmt):
    if fmt == 'currency_inr':
```

```
        if value >= 10000000:
            return f"Rs. {value/10000000:.2f} Cr"
        elif value >= 100000:
            return f"Rs. {value/100000:.2f} L"
        return f"Rs. {value:,.2f}"
    elif fmt == 'percentage':
        return f"{value:.1f}%"
    return f"{value:,.0f}"
```

KPI Calculator (kpi_calculator.py)

Expected Output

KPI Cards: Total Revenue: Rs. 45.23 L | Avg Order Value: Rs. 3,769.42 | Growth Rate: +12.4%
Charts: Revenue Trend (line), Top Products (bar), Category Distribution (pie) — all interactive.

Screenshot 3

What to paste: The completed Sales Dashboard showing all three KPI cards (Total Revenue, Average Order Value, Growth Rate) at the top, and the three charts (Revenue Trend line, Top Products bar, Category pie) below.

Paste your screenshot here

Step 6: Configure the Finance Dashboard

Objective: Build a financial analytics dashboard with P&L-focused KPIs and charts.

1. Upload `data/sample_financial.csv` and select **Finance Dashboard** template
2. Configure KPI cards: **Net Income** (revenue − expenses), **Expense Ratio** (expenses/revenue %), **Profit Margin** (net income/revenue %)
3. Configure charts: **P&L Trend** (dual-line: revenue vs expenses), **Expense Breakdown** (donut by category), **Cash Flow** (bar with positive/negative coloring)
4. Click **Build Dashboard** and compare layout with the sales dashboard

```
{
  "name": "Financial Analytics Dashboard",
  "kpis": [
    {"title": "Net Income",
     "calculation": "revenue - expenses",
     "format": "currency_inr", "color_rule": "positive_green"},
    {"title": "Expense Ratio",
     "calculation": "expenses / revenue * 100",
     "format": "percentage",
     "threshold": {"warning": 70, "danger": 85}},
    {"title": "Profit Margin",
     "calculation": "(revenue - expenses) / revenue * 100",
     "format": "percentage",
     "threshold": {"good": 20, "warning": 10}}
  ],
  "charts": [
    {"type": "line", "title": "P&L Trend", "x": "month",
     "y": ["revenue", "expenses"], "multi_series": true},
    {"type": "pie", "title": "Expense Breakdown",
     "labels": "expense_category", "values": "amount", "hole":
        0.4},
    {"type": "bar", "title": "Monthly Cash Flow",
     "x": "month", "y": "net_cash_flow",
     "color_rule": "positive_negative"}
  ]
}
```

Finance Dashboard Template (finance_dashboard.json)

Expected Output

KPIs: Net Income: Rs. 12.87 L (green) | Expense Ratio: 64.3% | Profit Margin: 35.7% (good)
Charts: P&L Trend (dual-line), Expense Breakdown (donut), Cash Flow (bar) rendered.

Step 7: Add Interactive Filters to the Dashboard

Objective: Enhance dashboards with date range, category, and drill-down filtering.

1. Locate the **Filters Panel** at the top of the dashboard
2. Set a **Date Range Filter** using start/end date pickers — observe real-time KPI updates
3. Add a **Category Filter** via the multi-select dropdown
4. Enable **Drill-Down**: click a bar/pie segment to see subcategories; use breadcrumbs to navigate back
5. Test combined filters (date + category), then reset with **Clear Filters**

```
@app.route('/api/filter', methods=['POST'])
def apply_filters():
    filters = request.get_json()
    filtered_df = df.copy()
    if 'date_start' in filters and 'date_end' in filters:
        filtered_df = filtered_df[
            (filtered_df['date'] >= filters['date_start']) &
            (filtered_df['date'] <= filters['date_end'])]
    if 'categories' in filters and filters['categories']:
        filtered_df = filtered_df[
            filtered_df['category'].isin(filters['categories'])]
    kpis = calculate_all_kpis(filtered_df, dashboard_config)
    charts = generate_all_charts(filtered_df, dashboard_config)
    return jsonify({'kpis': kpis, 'charts': charts,
                    'row_count': len(filtered_df)})
```

Filter API Endpoint

Expected Output

Filters applied: Date 2024-01-01 to 2024-06-30, Categories: Electronics, Clothing. Filtered: 480 of 1,200 rows. KPIs and charts updated. Drill-down into Electronics shows subcategories.

Screenshot 4

What to paste: The Finance Dashboard with interactive filters applied — showing a date range selection, at least one category filter active, and updated KPI cards and charts reflecting the filtered data.

Paste your screenshot here

Part D: Export & Customization

Step 8: Export Dashboard as Standalone HTML

Objective: Export the dashboard as a self-contained HTML file openable without a server.

1. Click the **Export** button in the dashboard toolbar
2. Select **Standalone HTML** format and choose a filename
3. Click **Download** to save the file
4. Open the exported file directly in your browser (double-click)
5. Verify all KPIs, charts, and hover interactivity work without a server

```python
import plotly

@app.route('/export', methods=['POST'])
def export_dashboard():
    config = request.get_json()
    figures = []
    for chart_config in config['charts']:
        fig = create_chart(df, chart_config)
        figures.append(plotly.io.to_html(
            fig, full_html=False, include_plotlyjs=False))
    kpis = calculate_all_kpis(df, config)
    html_content = render_template('export.html',
        title=config['name'], kpis=kpis, charts=figures,
        plotly_cdn=True)
    output_path = f"exports/{config['name']}.html"
    with open(output_path, 'w', encoding='utf-8') as f:
        f.write(html_content)
    return send_file(output_path, as_attachment=True)
```

Dashboard Export Logic

Expected Output

Exported: sales_dashboard_export.html (2.4 MB).
All KPI cards, charts, hover tooltips, and legend toggles functional — no server required.

The exported file loads Plotly.js from CDN for a smaller file size. For fully offline use, set `include_plotlyjs=True` to embed the library (adds approximately 3.5 MB).

Step 9: Customize — KPIs, Charts, and Styling

Objective: Add a custom KPI, a new chart type, and modify the dashboard color scheme.

1. **Modify KPI:** In `kpi_calculator.py`, add "Revenue per Region" (average revenue grouped by region)
2. **Add Chart:** In the dashboard template JSON, add a scatter plot (quantity vs. revenue correlation)
3. **Change Colors:** In `static/css/style.css`, update the primary color variables

```python
# kpi_calculator.py - new KPI
def calculate_revenue_per_region(df):
    region_rev = df.groupby('region')['revenue'].sum()
    return {'title': 'Avg Revenue/Region',
            'value': region_rev.mean(),
            'format': 'currency_inr'}
```

Custom KPI and Chart

```
{"type": "scatter", "title": "Quantity vs Revenue",
 "x": "quantity", "y": "revenue",
 "color": "category", "trendline": true}
```

Scatter Chart in Template JSON

```
:root {
  --primary-color: #2E86AB;
  --secondary-color: #A23B72;
  --accent-color: #F18F01;
}
.kpi-card {
  border-left: 4px solid var(--primary-color);
  padding: 1.5rem; border-radius: 8px;
}
```

CSS Color Scheme Update

Expected Output

New KPI "Avg Revenue/Region": Rs. 11.31 L. Scatter chart rendered with trendline. Color scheme updated to blue/purple/orange theme. Dashboard fully functional.

Screenshot 5

What to paste: The exported standalone HTML dashboard opened independently in a browser. The address bar should show a local file path (`file:///...`) with all KPI cards and charts rendering with full interactivity.

Paste your screenshot here

Conceptual Background

Business Intelligence Fundamentals

Business Intelligence (BI) encompasses the strategies, technologies, and practices used to collect, integrate, analyze, and present business data for better decision-making.

Core BI Components:

- **ETL (Extract, Transform, Load):** Extracting data from source systems, transforming it into a consistent format, and loading it into a data warehouse. In this practical, `data_processor.py` performs a simplified ETL pipeline.
- **OLAP (Online Analytical Processing):** Tools enabling multidimensional analysis — slice, dice, drill-down, and roll-up across dimensions such as time, region, and product category.
- **Data Warehouse:** A centralized repository optimized for query and analysis. Production BI systems use warehouses like Amazon Redshift, BigQuery, or Snowflake.

KPI Design Principles

SMART Criteria:

1. **Specific:** Clearly defined (e.g., "Monthly revenue growth" not "Business is growing")
2. **Measurable:** Quantifiable with a concrete number or percentage
3. **Achievable:** Realistic given available resources
4. **Relevant:** Aligned with business goals and strategic priorities
5. **Time-bound:** Measured over a defined period (daily, monthly, quarterly)

Leading vs. Lagging Indicators:

- **Leading:** Predict future performance (sales pipeline value, website traffic)
- **Lagging:** Measure past performance (total revenue, net profit, churn rate)
- Effective dashboards combine both types for a complete picture

Dashboard Design Principles

Information Hierarchy:

1. **Level 1 — KPI Cards:** Critical metrics as large numbers at the top
2. **Level 2 — Primary Charts:** Trends, comparisons, and distributions
3. **Level 3 — Detail Tables:** Granular data accessible via drill-down

Gestalt Principles: Proximity (group related items), Similarity (consistent visual encoding), Enclosure (borders to group elements), Continuity (logical reading flow).

The 5-Second Rule: A dashboard should communicate its core message within 5 seconds — KPI cards must be large, clearly labeled, and use color coding (green/red) for instant status recognition.

Self-Service Analytics

Self-service analytics empowers business users to explore data and derive insights without IT dependency. This practical's dashboard builder exemplifies this: data upload, auto-detection, template-based configuration, interactive filters, and exportable results.

Common Financial KPIs

KPI	Formula	Benchmark
Revenue Growth Rate	(Current − Previous) / Previous × 100	10–20% annually
Gross Profit Margin	(Revenue − COGS) / Revenue × 100	40–60%
Net Profit Margin	Net Income / Revenue × 100	10–20%
Return on Investment	Net Profit / Investment × 100	15–25%
Customer LTV	Avg Purchase × Frequency × Lifespan	Varies
Churn Rate	Lost Customers / Total Customers × 100	<5% monthly

India Context: BI in the Indian Market

- **Digital India:** Government dashboards tracking e-governance adoption across states
- **GST Analytics:** GSTN processes over 1 billion invoices monthly; BI dashboards monitor compliance and state-wise collection
- **UPI Dashboards:** NPCI tracks 10+ billion monthly UPI transactions with real-time dashboards for success rates and bank-wise performance
- **Flipkart/Amazon India:** Seller dashboards for revenue, order trends, return rates, and delivery performance

- **HDFC Bank / Zerodha:** Internal BI for branch comparison, loan portfolio analysis, and trading volume analytics

Assessment & Deliverables

Deliverables Checklist

Item	Description	Type	Status
Screenshot 1	Flask app upload interface	Paste	☐
Screenshot 2	Auto-detected columns and types	Paste	☐
Screenshot 3	Sales dashboard with KPIs and charts	Paste	☐
Screenshot 4	Finance dashboard with filters applied	Paste	☐
Screenshot 5	Exported standalone HTML dashboard	Paste	☐
KPI Report	Document all KPI values computed	Text	☐
Custom KPI	New KPI added and functional	Code	☐
Custom Chart	New chart type added to dashboard	Code	☐
Exported File	Standalone HTML file submitted	File	☐

Verification Checklist

- ☐ Python virtual environment created and all dependencies installed
- ☐ Flask application runs without errors on `http://localhost:5000`
- ☐ Sample sales CSV uploaded and column types auto-detected correctly
- ☐ Sales dashboard displays 3 KPI cards with correct values
- ☐ Sales dashboard renders 3 chart types (line, bar, pie)
- ☐ Finance dashboard loads with P&L-specific KPIs and charts
- ☐ Interactive filters (date range, category) update charts in real time
- ☐ Dashboard exported as standalone HTML and opens independently
- ☐ Custom KPI calculation added successfully
- ☐ New chart type added to a dashboard template
- ☐ Color scheme customization applied and visible
- ☐ All 5 required screenshots captured

Grading Rubric

Criteria	Description	Points	Score
Environment Setup	venv, dependencies, app runs	10	___/10
Data Upload	CSV uploaded, types auto-detected	10	___/10
Sales Dashboard	KPI cards and 3 charts correct	20	___/20
Finance Dashboard	Financial KPIs and P&L charts	15	___/15
Interactive Filters	Date, category, drill-down working	15	___/15
Export	Standalone HTML works offline	10	___/10
Customization	Custom KPI, chart, or color scheme	10	___/10
Screenshots	All 5 clear and complete	5	___/5
Assessment Answers	All 8 questions answered	5	___/5
	TOTAL	**100**	___/100

Assessment Questions

Q1. What is Business Intelligence and how does it differ from basic data reporting? Explain the role of ETL in the BI pipeline.

Q2. Explain the difference between leading and lagging KPIs. Provide two examples of each from a retail business context.

Q3. Describe the 5-second rule in dashboard design. How did you apply this principle when building your sales dashboard?

Q4. What are the advantages of self-service analytics over traditional IT-managed reporting? What risks does it introduce?

Q5. Compare OLAP and OLTP systems. Why are data warehouses preferred over operational databases for BI queries?

Q6. Design a dashboard for tracking UPI transaction performance across India. List five KPIs you would include and justify each.

Q7. Explain how interactive filters improve data exploration. What is the difference between a global filter and a chart-level filter?

Q8. A dashboard shows revenue growth of 15% but net profit declined by 8%. What possible explanations could you offer, and what additional KPIs would help investigate?

Appendix A: KPI Reference Table

KPI Name	Formula	Type	Category
Total Revenue	Sum of all revenue	Lagging	Financial
Revenue Growth	Period % change	Lagging	Financial
Gross Profit Margin	(Rev − COGS) / Rev	Lagging	Financial
Net Profit Margin	Net Income / Revenue	Lagging	Financial
Operating Expense Ratio	OpEx / Revenue	Lagging	Financial
Return on Investment	Net Profit / Investment	Lagging	Financial
Customer LTV	Avg × Freq × Lifespan	Leading	Customer
Customer Acq. Cost	Marketing / New Customers	Lagging	Customer
Churn Rate	Lost / Total Customers	Lagging	Customer
Average Order Value	Revenue / Orders	Lagging	Sales
Conversion Rate	Conversions / Visitors	Leading	Sales
Sales Pipeline Value	Weighted opportunities	Leading	Sales
Inventory Turnover	COGS / Avg Inventory	Lagging	Operations
Days Sales Outstanding	(Receivables / Rev) × 365	Lagging	Operations
Cash Flow from Ops	Op. Income + Noncash	Lagging	Financial
Debt-to-Equity	Total Debt / Equity	Lagging	Financial

Appendix B: Dashboard Layout Patterns

1. **Executive Summary:** Top row KPI cards, middle row primary charts, bottom row detail tables. Best for C-level overview.
2. **Analytical Deep-Dive:** Left sidebar filters, main area large chart with drill-down, right panel context KPIs. Best for analysts.
3. **Operational Monitoring:** Real-time KPI ticker, time-series charts with auto-refresh, event log table. Best for operations teams.
4. **Comparison Layout:** Side-by-side panels with synchronized filters for period-over-period or A/B analysis.

Appendix C: Plotly Chart Reference

Chart	Best For	Plotly Function	Key Params
Line	Trends over time	`px.line()`	x, y, color
Bar	Comparisons	`px.bar()`	x, y, orientation
Pie/Donut	Proportions	`px.pie()`	names, values, hole
Scatter	Correlations	`px.scatter()`	x, y, size, color
Histogram	Distributions	`px.histogram()`	x, nbins
Box Plot	Statistical spread	`px.box()`	x, y
Heatmap	Matrix patterns	`px.imshow()`	z, color_scale
Treemap	Hierarchical data	`px.treemap()`	path, values
Funnel	Conversion steps	`px.funnel()`	x, y
Indicator	Single KPI gauge	`go.Indicator()`	mode, value

Appendix D: Troubleshooting

Possible causes:

1. **venv not activated:** Check for `(venv)` prefix. Run activation command.
2. **Missing dependencies:** Run `pip install flask plotly pandas` again.
3. **Port 5000 in use:** Change port in `main.py`: `app.run(port=5001)`.
4. **Python version:** Ensure 3.8+ with `python -version`.

Possible causes:

1. **JavaScript blocked:** Disable ad-blockers that may interfere with Plotly.js.
2. **CDN unreachable:** Check internet connection or embed Plotly locally.
3. **Data format error:** Check browser console (F12) for JS errors — common with unparsed dates.
4. **Browser cache:** Hard-refresh with Ctrl+Shift+R.

Possible causes:

1. **CDN without internet:** Set `include_plotlyjs=True` for offline embedding.
2. **File path issues:** Move file to Desktop and try again.
3. **Large dataset:** Reduce data size or pre-aggregate before export.

Appendix E: Additional Resources

Documentation

- Plotly Python: `https://plotly.com/python/`
- Flask: `https://flask.palletsprojects.com/`
- pandas: `https://pandas.pydata.org/docs/user_guide/`
- Plotly Dash: `https://dash.plotly.com/`

Learning Resources

- "Storytelling with Data" by Cole Nussbaumer Knaflic
- "The Big Book of Dashboards" by Steve Wexler et al.
- "Information Dashboard Design" by Stephen Few
- NPCI UPI Statistics: `https://www.npci.org.in/what-we-do/upi/product-statistics`

Tools Used in This Practical

Tool	Purpose	Cost
Python 3.8+	Core programming runtime	Free
Flask	Web framework for dashboard serving	Free
Plotly	Interactive chart generation	Free
pandas	Data manipulation and transformation	Free
pip	Package manager	Free
Web Browser	Dashboard UI rendering	Free
VS Code	Code editor (optional)	Free
Git	Version control	Free

—END OF LAB MANUAL—

Document Version: 1.0

IT Management & Audits – Practical Lab Series