IT Management & Audits

Practical Lab Manual

Financial Data Visualizer

Practical P07

Learning Domain

Data Analytics & Visualization

Course Learning Outcomes

CLO07: Create financial data visualizations and interactive dashboards

Unit

Unit V: Business Intelligence & Analytics

Time Allocation: 3 hours

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

Difficulty Level: Intermediate

Financial Data Visualizer

Practical P07

Quick Reference

Practical Code	P07
Practical Name	Financial Data Visualizer
Slot	T/P-7
Duration	3 hours
CLO Mapping	CLO07
Unit	Unit V: Business Intelligence & Analytics
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	MEDIUM
Screenshot Count	5 Required

Prerequisites

- Basic understanding of statistics (mean, median, variance, percentiles)
- Familiarity with Python programming fundamentals
- Understanding of financial metrics (revenue, expenses, profit margins)
- Completion of P01 (Lab Setup & Infrastructure) recommended
- Basic command-line interface (CLI) proficiency

Tools Required

Tool	Version	Free	Notes
Python	3.8+	✓	Required
pandas	Latest	✓	Data manipulation
matplotlib	Latest	✓	Static chart generation
plotly	Latest	✓	Interactive charts
seaborn	Latest	✓	Statistical visualization
Click	Latest	✓	CLI framework
pip	Latest	✓	Included with Python
Web Browser	Latest	✓	For interactive dashboard

Learning Objectives

- ✓ Load and explore financial datasets using pandas for analysis readiness
- ✓ Compute statistical measures relevant to financial analysis (moving averages, growth rates, percentiles)
- ✓ Generate line charts, bar charts, pie charts, heatmaps, scatter plots, and candlestick charts
- ✓ Select the appropriate chart type based on the data story being communicated
- ✓ Build interactive HTML dashboards combining multiple visualizations using plotly
- ✓ Apply data visualization best practices and design principles to financial reporting
- ✓ Understand how Indian financial regulators (SEBI, RBI) use visualizations in disclosure and reporting

What You Will Learn

By the end of this practical, you will:

1. Load CSV-based financial datasets and perform exploratory data analysis
2. Compute summary statistics including moving averages, year-over-year growth, variance, and percentiles
3. Generate publication-quality line charts showing revenue trends and moving average overlays
4. Create bar charts for monthly comparisons, pie charts for expense distribution, and heatmaps for transaction patterns
5. Build candlestick (OHLC) charts for market data visualization
6. Combine all chart types into an interactive HTML dashboard with filters and drill-downs
7. Customize chart parameters, color schemes, and layouts to suit specific analytical needs
8. Understand the principles that guide effective financial data communication

Real-World Application

Data visualization is central to financial decision-making. Platforms like **Zerodha** and **Groww** transform raw market data into interactive candlestick charts, moving average overlays, and portfolio heatmaps that enable millions of Indian retail investors to make informed decisions.

The **Reserve Bank of India (RBI)** publishes its Financial Stability Report with dozens of carefully designed charts, while **SEBI** mandates visual disclosures in annual reports and offer documents. Mastering financial visualization means turning raw numbers into actionable insight.

Conceptual Background

Data Visualization Principles

Edward Tufte's foundational principles for effective data visualization remain the gold standard:

1. **Data-Ink Ratio:** Maximize the proportion of ink devoted to actual data. Remove unnecessary gridlines, borders, and decorations that do not convey information.
2. **Chartjunk Avoidance:** Eliminate 3D effects, excessive gradients, and ornamental elements that distract from the data.
3. **Small Multiples:** Use repeated small charts with the same axes to compare patterns across categories or time periods.
4. **Lie Factor:** Ensure visual representation is proportional to the actual data values. A 50% increase should not look like a 200% increase.
5. **Contextual Integrity:** Always provide axis labels, titles, units, time ranges, and data sources.

Chart Type Selection Guide

Choosing the correct chart type is critical for clear communication:

Chart Type	Best Used For	Financial Example
Line Chart	Trends over time	Revenue growth over 12 months
Bar Chart	Categorical comparison	Monthly expenses by department
Pie Chart	Part-to-whole composition	Expense distribution by category
Candlestick	OHLC market data	Stock price movements (open, high, low, close)
Heatmap	Pattern density/frequency	Transaction volume by hour and day of week
Scatter Plot	Relationship between variables	Risk versus expected return for assets

Pie charts should only be used when comparing parts of a whole (percentages summing to 100%). For comparing independent values, use bar charts instead. Avoid using more than 5–6 slices in a pie chart; group smaller categories into "Other."

Statistical Measures for Finance

- ▷ **Moving Averages (MA):** Smooths short-term fluctuations to reveal trends. Common periods: 7-day (weekly pattern), 30-day (monthly trend), 90-day (quarterly trend). Used extensively in technical analysis of stock prices.
- ▷ **Year-over-Year (YoY) Growth:** Compares a metric to the same period in the previous year, eliminating seasonal effects. Formula: $\frac{Current - Previous}{Previous} \times 100\%$.
- ▷ **Volatility (Standard Deviation):** Measures the dispersion of returns. Higher volatility indicates higher risk. Annualized volatility is commonly reported.
- ▷ **Correlation:** Measures the linear relationship between two variables (range: -1 to $+1$). Used to assess portfolio diversification.
- ▷ **Percentiles:** The value below which a given percentage of observations fall. The 25th, 50th (median), and 75th percentiles define the interquartile range (IQR).

Dashboard Design Principles

Effective dashboards follow a clear hierarchy:

1. **KPI Hierarchy:** Place the most critical metrics (total revenue, net profit, portfolio value) at the top in large, prominent displays.
2. **Progressive Detail:** Start with summary metrics, then allow drill-down into detailed charts and tables below.
3. **Layout:** Use a grid layout with consistent spacing. Place related charts adjacent to each other.
4. **Color Usage:** Use a consistent color palette. Reserve red for losses/warnings and green for gains/positive indicators. Avoid using more than 5–7 colors.
5. **Interactivity:** Provide filters (date range, category), hover tooltips, and zoom capability for exploration.

India Context: Financial Data Visualization in Practice

SEBI Disclosure Requirements

The Securities and Exchange Board of India (SEBI) mandates that listed companies include visual representations in:

- Annual reports with graphical depiction of financial performance trends
- Offer documents with risk-return profile charts
- Mutual fund factsheets with portfolio composition pie charts and NAV trend lines
- Quarterly results with comparative bar charts across periods

RBI Financial Stability Reports

The RBI publishes semi-annual Financial Stability Reports featuring:

- Systemic risk heatmaps showing stress across banking sectors
- Credit growth line charts with moving average overlays
- NPA (Non-Performing Assets) trend visualizations by bank category
- Macro-financial stress indicator dashboards

How Zerodha and Groww Visualize Market Data

Indian trading platforms demonstrate professional financial visualization:

- ▷ **Zerodha Kite:** Interactive candlestick charts with 15+ technical indicators, volume overlays, and drawing tools. Serves 10+ million active traders.
- ▷ **Groww:** Simplified line charts for mutual fund NAV trends, pie charts for portfolio allocation, and bar charts for SIP return comparison. Designed for beginner investors.
- ▷ **Key Design Choice:** Zerodha targets advanced traders with dense, information-rich charts. Groww targets beginners with clean, minimal visualizations. Both are effective because they match their audience.

Hands-On Procedure

Part A: Environment Setup

Step 1: Clone the Financial Data Visualizer Repository

Objective: Download and set up the financial data visualization tool on your local machine.
Instructions:

1. Open your terminal (bash on Linux/Mac, PowerShell on Windows)
2. Navigate to your working directory
3. Clone the repository from GitHub
4. Navigate into the project directory
5. Explore the project structure and sample data

Code/Command:

```
# Clone the repository
git clone <repository-url> financial-data-visualizer
cd financial-data-visualizer

# View the project structure
ls -la
# Expected directories: src/, data/, charts/

# View source code structure
ls src/
# Expected: data_loader.py, statistics.py, dashboard.py, cli.py
# Expected: charts/ directory

# View chart modules
ls src/charts/
# Expected: line_charts.py, bar_charts.py, pie_charts.py,
#           heatmaps.py, scatter_plots.py, candlestick.py

# View sample datasets
ls data/
# Expected: transactions.csv, market_data.csv,
#           revenue_expenses.csv

# Preview sample data
head -10 data/transactions.csv
head -10 data/revenue_expenses.csv
head -10 data/market_data.csv
```

Clone and Explore the Repository

Expected Output

Project directory contains:
`src/data_loader.py` – CSV loading and data preprocessing
`src/statistics.py` – Statistical analysis functions
`src/charts/line_charts.py` – Line chart generation
`src/charts/bar_charts.py` – Bar chart generation
`src/charts/pie_charts.py` – Pie chart generation
`src/charts/heatmaps.py` – Heatmap generation
`src/charts/scatter_plots.py` – Scatter plot generation
`src/charts/candlestick.py` – Candlestick (OHLC) chart generation
`src/dashboard.py` – Interactive HTML dashboard builder
`src/cli.py` – Command-line interface (Click-based)
`data/transactions.csv` – Sample transaction records
`data/market_data.csv` – Sample OHLC market data
`data/revenue_expenses.csv` – Sample revenue and expense data

If you do not have Git installed, you can download the repository as a ZIP file from the GitHub page and extract it manually. Preview the CSV files to understand column names and data formats before proceeding.

Step 2: Create Virtual Environment and Install Dependencies

Objective: Set up an isolated Python environment with all required visualization libraries.
Instructions:

1. Create a Python virtual environment
2. Activate the virtual environment
3. Install project dependencies from `requirements.txt`
4. Verify installation by running the CLI help command
5. Confirm all three commands are available: `analyze`, `chart`, `dashboard`

Code/Command:

```
# Create virtual environment
python -m venv venv

# Activate (Linux/Mac)
source venv/bin/activate

# Activate (Windows PowerShell)
.\venv\Scripts\Activate.ps1

# Install dependencies
```

```
pip install -r requirements.txt

# Verify installation - check CLI help
python src/cli.py --help

# Verify individual commands
python src/cli.py analyze --help
python src/cli.py chart --help
python src/cli.py dashboard --help
```

Python Environment Setup

Expected Output

CLI help output shows available commands:

```
Usage:  cli.py [OPTIONS] COMMAND [ARGS]...

Financial Data Visualizer CLI

Commands:
analyze    Load data and compute statistical analysis
chart      Generate charts from financial data
dashboard  Build interactive HTML dashboard
```

The `requirements.txt` includes pandas, matplotlib, plotly, seaborn, and Click. If any library fails to install, install it individually: `pip install pandas matplotlib plotly seaborn click`.

If `pip install` fails: (1) Ensure the virtual environment is activated (you should see `(venv)` in your prompt), (2) Try `python -m pip install -r requirements.txt`, (3) On Windows, run your terminal as Administrator if needed, (4) Verify Python 3.8+: `python -version`.

Screenshot 1

What to paste: Terminal showing the cloned repository file structure (output of `ls` commands showing `src/`, `src/charts/`, and `data/` contents) and a preview of one sample CSV dataset (first 10 rows).

Paste your screenshot here

Part B: Data Loading & Statistical Analysis

Step 3: Load and Explore Sample Datasets

Objective: Use the CLI to load financial datasets and display summary information.
Instructions:

1. Run the analyze command on the transactions dataset
2. Observe the summary statistics displayed in the terminal
3. Note the date range, total record count, and column types
4. Repeat for the revenue and expenses dataset

Code/Command:

```
# Analyze the transactions dataset
python src/cli.py analyze --input data/transactions.csv

# Expected: Shows column names, data types, row count,
#           date range, basic statistics (mean, min, max)

# Analyze the revenue and expenses dataset
python src/cli.py analyze --input data/revenue_expenses.csv

# Analyze the market data
python src/cli.py analyze --input data/market_data.csv
```

Load and Explore Datasets

Expected Output

```
=== Dataset Summary: transactions.csv ===
Total Records: 2,450
Columns: date, category, amount, type, description
Date Range: 2023-01-01 to 2024-12-31
Numeric Columns:
amount: mean=15,230.45, min=120.00, max=485,000.00
Missing Values: 0

=== Dataset Summary: revenue_expenses.csv ===
Total Records: 730
Columns: date, revenue, expenses, profit, category
Date Range: 2023-01-01 to 2024-12-31
```

Pay attention to the data types and column names. Understanding the structure of each dataset is essential before generating visualizations. The `transactions.csv` file contains individual transaction records, while `revenue_expenses.csv` has daily aggregate figures.

Step 4: Run Statistical Analysis with Moving Averages and Growth Rates

Objective: Compute advanced statistical measures on financial data.
Instructions:

1. Run the analyze command with the `-stats` flag
2. Review the moving averages (7-day, 30-day, 90-day windows)
3. Examine the year-over-year growth rates
4. Note the variance and percentile distributions
5. Compare statistics across different datasets

Code/Command:

```
# Run full statistical analysis on revenue data
python src/cli.py analyze \
    --input data/revenue_expenses.csv \
    --stats

# Expected output includes:
# - Moving Averages (7-day, 30-day, 90-day)
# - Year-over-Year growth rates
# - Variance and standard deviation
# - Percentiles (25th, 50th, 75th, 95th)

# Run statistics on transaction data
python src/cli.py analyze \
    --input data/transactions.csv \
    --stats
```

Statistical Analysis

Expected Output

```
=== Statistical Analysis:  revenue_expenses.csv ===

-- Moving Averages (Revenue) --
7-day MA (latest):  INR 1,24,350.00
30-day MA (latest):  INR 1,18,920.50
90-day MA (latest):  INR 1,15,440.75

-- Year-over-Year Growth --
Revenue YoY Growth:  +18.5%
Expenses YoY Growth:  +12.3%
Profit YoY Growth:   +28.7%

-- Variance & Distribution --
Revenue Std Dev: INR 32,150.00
Revenue Variance:  1,033,622,500.00
25th Percentile:  INR 89,200.00
50th Percentile:  INR 1,12,500.00
75th Percentile:  INR 1,38,800.00
95th Percentile:  INR 1,72,400.00
```

Moving averages smooth out daily fluctuations. A 7-day MA reacts quickly to changes (useful for short-term trading signals), while a 90-day MA reveals the underlying trend (useful for strategic planning). When the short-term MA crosses above the long-term MA, it is called a "golden cross" – a bullish signal in technical analysis.

Screenshot 2

What to paste: Terminal output showing the statistical analysis results, including moving averages (7/30/90-day), year-over-year growth rates, and percentile distributions for the revenue and expenses dataset.

Paste your screenshot here

Part C: Chart Generation

Step 5: Generate Line Charts for Revenue Trends

Objective: Create line charts showing revenue trends with moving average overlays.
Instructions:

1. Use the chart command with `-type line` to generate line charts
2. The output will include revenue trend lines with 7-day and 30-day moving average overlays
3. Charts are saved as PNG files in the specified output directory
4. Open the generated chart files to verify the output

Code/Command:

```
# Create output directory for charts
mkdir -p charts

# Generate line charts from revenue data
python src/cli.py chart \
    --type line \
    --input data/revenue_expenses.csv \
    --output charts/

# Expected output files:
# charts/revenue_trend.png
# charts/revenue_with_ma.png (with moving average overlays)
# charts/profit_trend.png

# List generated files
ls -la charts/
```

Generate Line Charts

Expected Output

```
Generating line charts...
▷ Revenue trend line chart saved:  charts/revenue_trend.png
▷ Revenue with MA overlay saved:  charts/revenue_with_ma.png
▷ Profit trend line chart saved:  charts/profit_trend.png
3 charts generated successfully.
```

Line charts are the most common chart type for financial time series. The moving average overlay helps distinguish short-term noise from long-term trends. When presenting to stakeholders, the 30-day MA is typically the most useful balance between responsiveness and smoothness.

Step 6: Generate Bar, Pie, Heatmap, and Scatter Charts

Objective: Generate multiple chart types to visualize different aspects of financial data.
Instructions:

1. Generate bar charts for monthly expense comparison
2. Generate pie charts for expense category distribution
3. Generate heatmaps showing transaction volume by hour and day
4. Generate scatter plots showing risk versus return
5. Review all generated chart files

Code/Command:

```
# Generate bar charts (monthly comparison)
python src/cli.py chart \
    --type bar \
    --input data/revenue_expenses.csv \
    --output charts/

# Generate pie charts (expense distribution)
python src/cli.py chart \
    --type pie \
    --input data/transactions.csv \
    --output charts/

# Generate heatmaps (transaction volume by hour/day)
python src/cli.py chart \
    --type heatmap \
    --input data/transactions.csv \
    --output charts/

# Generate scatter plots (risk vs return)
python src/cli.py chart \
    --type scatter \
    --input data/market_data.csv \
    --output charts/

# List all generated charts
ls -la charts/
```

Generate Multiple Chart Types

Expected Output

```
Generating bar charts...
▷ Monthly comparison saved:  charts/monthly_comparison.png

Generating pie charts...
▷ Expense distribution saved:  charts/expense_distribution.png

Generating heatmaps...
▷ Transaction heatmap saved:  charts/transaction_heatmap.png

Generating scatter plots...
▷ Risk vs return saved:  charts/risk_return_scatter.png
```

Each chart type serves a specific analytical purpose. Bar charts compare magnitudes across categories. Pie charts show composition (parts of a whole). Heatmaps reveal patterns across two dimensions (e.g., time of day versus day of week). Scatter plots expose relationships between two continuous variables (e.g., risk and return).

Screenshot 3

What to paste: Two generated chart images side by side or stacked – the line chart showing revenue trends with moving average overlays (`revenue_with_ma.png`) and the bar chart showing monthly comparisons (`monthly_comparison.png`).

Paste your screenshot here

Step 7: Generate Candlestick Chart for Market Data

Objective: Create a candlestick (OHLC) chart from market data, the standard visualization for stock price movements.
Instructions:

1. Use the chart command with `-type candlestick`
2. Input the market data CSV containing open, high, low, close columns
3. Review the generated candlestick chart
4. Understand how to read bullish (green/hollow) versus bearish (red/filled) candles

Code/Command:

```
# Generate candlestick (OHLC) chart
python src/cli.py chart \
    --type candlestick \
    --input data/market_data.csv \
    --output charts/

# Expected output:
# charts/candlestick_ohlc.png
# charts/candlestick_with_volume.png (if volume data present)

# List all charts generated so far
ls -la charts/*.png
```

Generate Candlestick Chart

Expected Output

```
Generating candlestick charts...
▷ OHLC candlestick saved:  charts/candlestick_ohlc.png
▷ Candlestick with volume saved:  charts/candlestick_with_volume.png
2 charts generated successfully.

Total charts in output directory:  9 files
```

Reading a candlestick chart: Each "candle" represents one time period (day, hour, etc.). The body shows the range between open and close prices. The wicks (thin lines above and below) show the high and low. A green/hollow candle means the close was higher than the open (bullish). A red/filled candle means the close was lower than the open (bearish).

Screenshot 4

What to paste: Two chart images – the heatmap showing transaction volume patterns by hour and day of week (`transaction_heatmap.png`) and the candlestick chart showing OHLC market data (`candlestick_ohlc.png`).

Paste your screenshot here

Part D: Interactive Dashboard

Step 8: Generate Interactive HTML Dashboard

Objective: Combine all chart types into a single interactive HTML dashboard using plotly.
Instructions:

1. Run the dashboard command pointing to the entire data directory
2. The tool will process all CSV files and generate an interactive HTML file
3. Open the generated dashboard in your web browser
4. Explore interactive features: hover tooltips, zoom, pan, date range filters
5. Test the filter controls to isolate specific time periods or categories

Code/Command:

```
# Generate the interactive HTML dashboard
python src/cli.py dashboard \
    --input data/ \
    --output dashboard.html

# Open the dashboard in your default browser
# Linux/Mac:
xdg-open dashboard.html
# or open dashboard.html

# Windows:
start dashboard.html
```

Generate Interactive Dashboard

Expected Output

```
Building interactive dashboard...
▷ Loading data/transactions.csv (2,450 records)
▷ Loading data/revenue_expenses.csv (730 records)
▷ Loading data/market_data.csv (504 records)

▷ Creating revenue trend panel...
▷ Creating expense breakdown panel...
▷ Creating transaction heatmap panel...
▷ Creating market data panel...
▷ Creating risk-return scatter panel...
▷ Adding interactive filters...

Dashboard saved:  dashboard.html (1.2 MB)
Open in your web browser to interact with the charts.
```

The dashboard is a self-contained HTML file. It uses the plotly.js library embedded within the file, so no internet connection is required to view it. The file may be large (1–2 MB) because it includes all chart data and the JavaScript library inline.

Step 9: Customize Dashboard Parameters and Add Your Own Data

Objective: Modify chart configurations and integrate your own financial dataset.
Instructions:

1. Open `src/dashboard.py` in a text editor
2. Review the chart configuration parameters (colors, titles, layout)
3. Modify at least one chart parameter (e.g., change color scheme, adjust moving average window)
4. Create a small custom CSV file with your own financial data
5. Regenerate the dashboard with modified parameters
6. Compare the original and customized dashboards

Code/Command:

```
# View dashboard configuration
cat src/dashboard.py | head -50

# Create a custom dataset (example)
cat > data/my_expenses.csv << 'EOF'
date,category,amount
2024-01-15,Rent,25000
2024-01-15,Food,8500
2024-01-15,Transport,3200
2024-01-15,Utilities,2800
2024-01-15,Entertainment,1500
2024-02-15,Rent,25000
2024-02-15,Food,9200
2024-02-15,Transport,2900
2024-02-15,Utilities,3100
2024-02-15,Entertainment,2000
EOF

# Regenerate dashboard with custom data included
python src/cli.py dashboard \
    --input data/ \
    --output dashboard_custom.html

# Open the customized dashboard
# Windows:
start dashboard_custom.html
# Linux/Mac:
xdg-open dashboard_custom.html
```

Customize Dashboard

Expected Output

```
Building interactive dashboard...
▷ Loading data/transactions.csv (2,450 records)
▷ Loading data/revenue_expenses.csv (730 records)
▷ Loading data/market_data.csv (504 records)
▷ Loading data/my_expenses.csv (10 records)

Dashboard saved:  dashboard_custom.html (1.4 MB)
```

Experiment with different chart parameters to understand their impact. Try changing moving average windows (e.g., from 7/30/90 to 5/20/60), switching color palettes, or adding annotations for significant events (e.g., budget quarter boundaries). Customization skills are valuable when building dashboards for real business stakeholders.

Screenshot 5

What to paste: The interactive HTML dashboard open in your web browser, showing multiple panels (revenue trend, expense breakdown, heatmap, market data). Capture the full browser window including the interactive filter controls.

Paste your screenshot here

Assessment & Deliverables

Deliverables Checklist

Item	Description	Type	Status
Screenshot 1	Repo structure & data preview	Paste	☐
Screenshot 2	Statistical analysis output	Paste	☐
Screenshot 3	Line chart & bar chart	Paste	☐
Screenshot 4	Heatmap & candlestick chart	Paste	☐
Screenshot 5	Interactive dashboard in browser	Paste	☐
Custom CSV	Your custom financial dataset	File	☐
Chart Files	All generated PNG charts	Files	☐
Dashboard	HTML dashboard file	File	☐

Verification Checklist

Complete all items below before submitting:

- ☐ Repository cloned and virtual environment set up successfully
- ☐ CLI tool runs with `analyze`, `chart`, and `dashboard` commands
- ☐ All three sample datasets loaded and explored (transactions, revenue/expenses, market data)
- ☐ Statistical analysis completed with moving averages, growth rates, and percentiles
- ☐ Line charts generated with moving average overlays
- ☐ Bar charts, pie charts, heatmaps, and scatter plots generated
- ☐ Candlestick (OHLC) chart generated from market data
- ☐ Interactive HTML dashboard generated and opened in browser
- ☐ Dashboard interactivity tested (hover, zoom, filter controls)
- ☐ Custom dataset created and integrated into dashboard
- ☐ All 5 required screenshots captured and pasted

Grading Rubric

Criteria	Description	Points	Score
Setup	Repo cloned, venv ready, CLI verified	10	___/10
Data Loading	Datasets loaded, summary stats reviewed	10	___/10
Statistical Analysis	Moving averages, YoY growth, percentiles	15	___/15
Line Charts	Revenue trend with MA overlays	10	___/10
Multiple Charts	Bar, pie, heatmap, scatter generated	15	___/15
Candlestick	OHLC chart from market data	10	___/10
Dashboard	Interactive HTML dashboard functional	15	___/15
Customization	Custom data and modified parameters	10	___/10
Documentation	Answers & screenshots complete	5	___/5
	TOTAL	**100**	___/100

Assessment Questions

Answer the following questions in your submission:

Q1. What are the five data visualization principles proposed by Edward Tufte? Explain the "data-ink ratio" concept with an example.

Q2. When would you use a line chart versus a bar chart for financial data? Provide a specific scenario for each.

Q3. Explain the difference between a 7-day, 30-day, and 90-day moving average. In what situation would each be most useful?

Q4. What does a candlestick chart show that a simple line chart does not? Why is this additional information important for traders?

Q5. How does SEBI require listed companies to use data visualization in their financial disclosures? Name at least three types of required charts.

Q6. Your heatmap shows that transaction volume peaks between 10:00 AM and 2:00 PM on weekdays. How would you use this insight for business decision-making?

Q7. Describe three key principles of effective dashboard design. How would you prioritize information layout for a CFO versus a data analyst?

Q8. Compare how Zerodha and Groww approach data visualization differently. What design principle explains their different choices?

Appendix A: Chart Type Selection Guide

Use the table below to select the most appropriate chart for your data:

Chart Type	Data Relationship	When to Use	When to Avoid
Line Chart	Continuous trends over time	Revenue over months, stock price over days	Comparing unrelated categories
Bar Chart	Categorical comparison	Monthly expenses by type, department budgets	Time series with many data points (>50)
Pie Chart	Part-to-whole composition	Expense breakdown, portfolio allocation	More than 6 categories, comparing across groups
Candlestick	OHLC price data	Daily/weekly stock price analysis	Non-financial data, aggregated summaries
Heatmap	Two-dimensional density	Transaction patterns (hour vs day), correlation matrices	Single-dimension data, small datasets
Scatter Plot	Bivariate relationship	Risk vs return, correlation analysis	Categorical data, time series
Area Chart	Cumulative trends	Stacked revenue by product line	Comparing independent series
Histogram	Distribution shape	Return distribution, transaction amount frequency	Categorical data, time series

Appendix B: matplotlib and plotly Quick Reference

matplotlib Common Operations

```
import matplotlib.pyplot as plt
import pandas as pd

# Read data
df = pd.read_csv('data.csv', parse_dates=['date'])

# Basic line chart
plt.figure(figsize=(12, 6))
plt.plot(df['date'], df['revenue'], label='Revenue', color='teal')
plt.title('Revenue Trend')
plt.xlabel('Date')
plt.ylabel('Amount (INR)')
plt.legend()
```

```
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.savefig('revenue_trend.png', dpi=150)
plt.show()

# Moving average overlay
df['MA_30'] = df['revenue'].rolling(window=30).mean()
plt.plot(df['date'], df['MA_30'], label='30-day MA',
         linestyle='--', color='orange')

# Bar chart
plt.bar(categories, values, color=['teal', 'coral', 'gold'])

# Pie chart
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)

# Heatmap with seaborn
import seaborn as sns
sns.heatmap(pivot_table, annot=True, cmap='YlOrRd', fmt='.0f')
```

matplotlib Quick Reference

plotly Common Operations

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

# Interactive line chart
fig = px.line(df, x='date', y='revenue', title='Revenue Trend')
fig.update_layout(hovermode='x unified')
fig.write_html('interactive_chart.html')

# Candlestick chart
fig = go.Figure(data=[go.Candlestick(
    x=df['date'],
    open=df['open'], high=df['high'],
    low=df['low'], close=df['close']
)])
fig.update_layout(title='Market Data - OHLC',
                  xaxis_rangeslider_visible=True)

# Dashboard with subplots
from plotly.subplots import make_subplots
fig = make_subplots(rows=2, cols=2,
    subplot_titles=('Revenue', 'Expenses',
                    'Distribution', 'Heatmap'))

# Add traces to each subplot
fig.add_trace(go.Scatter(x=dates, y=revenue), row=1, col=1)
fig.add_trace(go.Bar(x=categories, y=amounts), row=1, col=2)

# Save as self-contained HTML
fig.write_html('dashboard.html', include_plotlyjs=True)
```

plotly Quick Reference

Appendix C: Financial Metrics Glossary

Metric	Definition	Visualization
Moving Average (MA)	Average of last n data points, recalculated each period	Line overlay on time series
YoY Growth	Percentage change compared to the same period one year ago	Bar chart or annotated line
Volatility	Standard deviation of returns over a period	Band chart or error bars
Correlation	Linear relationship strength between two variables (-1 to $+1$)	Heatmap or scatter plot
Percentile (Px)	Value below which $x\%$ of observations fall	Box plot or histogram
OHLC	Open, High, Low, Close prices for a trading period	Candlestick chart
Sharpe Ratio	Risk-adjusted return: $(R - R_f)/\sigma$	Bar chart comparison
Drawdown	Peak-to-trough decline in portfolio value	Area chart (inverted)
Volume	Number of shares/units traded in a period	Bar chart below price chart
RSI	Relative Strength Index, momentum oscillator (0–100)	Line chart with overbought/oversold bands

Appendix D: Troubleshooting

Problem: Generated PNG files are blank, corrupted, or not created.
Solutions:

1. Ensure matplotlib backend is set correctly: add `import matplotlib; matplotlib.use('Agg')` at the top of chart scripts for headless (non-GUI) environments
2. Check that the output directory exists: `mkdir -p charts/`
3. Verify the input CSV has the expected column names and formats
4. Check for sufficient disk space for image files
5. Try generating a simpler chart first to isolate the issue

Problem: The HTML dashboard file opens but displays a blank page or incomplete charts.
Solutions:

1. Ensure the dashboard was generated with `include_plotlyjs=True` (embeds the JS library)
2. Try a different browser (Chrome or Firefox recommended)
3. Check file size – if the file is 0 KB, the generation failed; check terminal for errors
4. Clear browser cache and reload the page
5. If using a corporate network, proxy settings may block CDN resources; use the self-contained option

Problem: Data loading fails with parsing errors, wrong data types, or encoding issues.
Solutions:

1. Specify date parsing explicitly: `pd.read_csv('file.csv', parse_dates=['date'])`
2. Check for encoding issues: try `encoding='utf-8'` or `encoding='latin-1'`
3. Inspect the CSV file manually for malformed rows: `head -5 data/file.csv`
4. Handle missing values: use `df.dropna()` or `df.fillna(0)` as appropriate
5. Verify column names match what the code expects (case-sensitive)

Appendix E: Additional Resources

Official Documentation

→ pandas Documentation: `https://pandas.pydata.org/docs/`

→ matplotlib Documentation: `https://matplotlib.org/stable/`

→ plotly Python Documentation: `https://plotly.com/python/`

→ seaborn Documentation: `https://seaborn.pydata.org/`

→ SEBI Annual Report Formats: `https://www.sebi.gov.in`

→ RBI Financial Stability Report: `https://www.rbi.org.in`

Learning Resources

- "The Visual Display of Quantitative Information" – Edward R. Tufte

- "Storytelling with Data" – Cole Nussbaumer Knaflic
- "Python for Data Analysis" – Wes McKinney (creator of pandas)
- "Interactive Data Visualization for the Web" – Scott Murray
- "Financial Data Analysis with Python" – O'Reilly Media

Tools Used in This Practical

Tool	Purpose	Cost
Python 3.8+	Programming language runtime	Free
pandas	Data loading, manipulation, and analysis	Free
matplotlib	Static chart generation (line, bar, pie, scatter)	Free
seaborn	Statistical visualization (heatmaps, distributions)	Free
plotly	Interactive chart and dashboard generation	Free
Click	CLI framework for command-line interface	Free
pip	Python package installer	Free
Web Browser	Viewing interactive HTML dashboards	Free

—END OF LAB MANUAL—

Document Version: 1.0

IT Management & Audits – Practical Lab Series