

# Customer Shopping Trends Analytics



Session: 2023 – 2027

## Submitted by:

Amir Hashmi                      2023-CS-11

Ahmed Butt                        2023-CS-18

## Supervised by:

Sir Talha Waheed

Department of Computer Science  
**University of Engineering and Technology Lahore**  
**Pakistan**

## Table Of Contents

1. Introduction .....	3
2. Project Overview .....	3
3. Key Features .....	3
4. The Data Set .....	3
5. Technical Architecture: .....	4
Frontend Components .....	7
6. Key Visualizations and Insights: .....	9
7. Implementation Challenges and Solutions: .....	9
8. Business Insights: .....	12
9. Future Enhancements: .....	13
10. Conclusion: .....	13

# Building an Interactive Customer Shopping Trends Analytics Dashboard

A comprehensive journey through designing and implementing a data visualization platform for retail analytics

## 1. Introduction

In today's data-driven retail landscape, understanding customer shopping behaviors is critical for business success. This blog post documents my recent project: an interactive web-based dashboard for visualizing and analyzing customer shopping trends. The platform integrates multiple Python visualization libraries, a Flask backend, and a responsive frontend to provide comprehensive insights into customer preferences, purchase patterns, and demographic information.

## 2. Project Overview

This customer shopping trends analytics dashboard processes retail transaction data to generate dynamic visualizations that help identify patterns in consumer behavior. The platform offers multiple visualization approaches through four popular Python libraries - Matplotlib, Seaborn, Pandas, and Plotly - allowing users to explore the same dataset through different analytical lenses.

## 3. Key Features

- **Multi-library visualization** with Matplotlib, Seaborn, Pandas, and Plotly
- **Interactive dashboard** with theme switching and responsive design
- **Dynamic data loading** with real-time visualization generation
- **Dataset information panel** for contextual understanding
- **Downloadable visualizations** for reports and presentations
- **Dark/light theme support** for different viewing preferences
- **Comprehensive retail metrics** including demographics, purchases, and reviews

## 4. The Data Set

The project uses a comprehensive retail dataset containing customer shopping information including:

- **Demographics:** Age, gender, location
- **Purchase details:** Item, category, price
- **Payment methods:** Credit/debit cards, PayPal, etc.

- **Shipping types:** Standard, express, next-day
- **Review ratings:** Customer satisfaction levels
- **Subscription status:** Active subscriptions

This rich dataset enables deep analysis of shopping patterns across different customer segments and product categories

## 5. Technical Architecture:

### Backend Components

The backend is powered by Flask and incorporates several Python libraries for data processing and visualization:

```
1 from flask import Flask, render_template, jsonify, request, send_file
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import plotly.express as px
6 import base64
7 import io
```

The Flask server handles visualization requests through dedicated endpoints:

```
1 @app.route('/get_visualizations', methods=['POST'])
2 def get_visualizations():
3     library = request.form.get('library')
4     df = load_data()
5
6     if library == 'matplotlib':
7         results = generate_matplotlib_visualizations(df)
8     elif library == 'seaborn':
9         results = generate_seaborn_visualizations(df)
10    elif library == 'pandas':
11        results = generate_pandas_visualizations(df)
12    elif library == 'plotly':
13        results = generate_plotly_visualizations(df)
14    else:
15        return jsonify({'error': 'Invalid library selected'})
16
17    return jsonify(results)
```

## Visualization Generation

Each visualization library has its own module for generating specific insights:

### Matplotlib Visualizations

Matplotlib provides foundational visualizations with direct control over plot elements:

```
1 def generate_matplotlib_visualizations(df):
2     visualizations = []
3
4     # Age Distribution
5     plt.figure(figsize=(10, 6))
6     plt.hist(df_clean['Age'].values, bins=20, edgecolor='black', color='skyblue')
7     plt.title('Age Distribution of Customers')
8     plt.xlabel('Age')
9     plt.ylabel('Count')
10    visualizations.append(save_plot_to_base64(plt, 'Age Distribution'))
11
12    # Other visualizations...
13
14    return {
15        'library': 'matplotlib',
16        'visualizations': visualizations
17    }
```

### Seaborn Visualizations

Seaborn enhances statistical visualizations with minimal code:

```
1 def generate_seaborn_visualizations(df):
2     visualizations = []
3
4     # Age Distribution by Gender
5     plt.figure(figsize=(10, 6))
6     sns.histplot(data=df, x='Age', hue='Gender', kde=True, bins=20, alpha=0.6)
7     plt.title('Age Distribution by Gender')
8     visualizations.append(save_plot_to_base64(plt, 'Age Distribution by Gender'))
9
10    # Additional visualizations...
11
12    return {
13        'library': 'seaborn',
14        'visualizations': visualizations
15    }
```

## Pandas Visualizations

Pandas provides quick, integrated visualizations directly from DataFrames:

```
1 def generate_pandas_visualizations(df):
2     visualizations = []
3
4     # Top 10 Items Purchased
5     plt.figure(figsize=(10, 6))
6     item_counts = df['Item Purchased'].value_counts()
7     top_10_items = item_counts.head(10)
8     sorted_top_items = top_10_items.sort_values()
9     sorted_top_items.plot(kind='barh', color='teal')
10    visualizations.append(save_plot_to_base64(plt, 'Top 10 Items Purchased'))
11
12    # More visualizations...
13
14    return {
15        'library': 'pandas',
16        'visualizations': visualizations
17    }
```

## NumPy Visualizations

NumPy visualizations offer low-level, efficient control over data representation, enabling optimized, custom-built charts using array operations and manual plotting.

```
def generate_numpy_visualizations(df):  
    # Visualization 6: Average Rating by Season  
    unique_seasons = np.unique(seasons)  
    avg_ratings_by_season = np.zeros(len(unique_seasons))  
    for i, season in enumerate(unique_seasons):  
        avg_ratings_by_season[i] = np.mean(review_ratings[seasons == season])  
  
    plt.figure(figsize=(10, 6))  
    plt.plot(unique_seasons, avg_ratings_by_season, marker='o', color='purple')  
    plt.title('Average Rating by Season (NumPy)')  
    plt.xlabel('Season')  
    plt.ylabel('Average Rating')  
    plt.ylim(0, 5)  
    plt.grid(True, linestyle='--', alpha=0.3)  
    visualizations.append(save_plot_to_base64(plt, 'Average Rating by Season'))  
    plt.close()  
  
    return {  
        'library': 'numpy',  
        'visualizations': visualizations  
    }
```

## Frontend Components

The frontend combines HTML, CSS, and JavaScript to create an interactive user experience:

**Responsive UI Structure** The HTML structure creates a responsive dashboard with a sidebar for library selection.

```
1 <div class="app-container">  
2   <nav class="sidebar">  
3     <div class="sidebar-header">  
4       <div class="logo">  
5         <i class="fas fa-chart-line"></i>  
6         <span>Data Viz</span>  
7       </div>  
8       <button class="menu-toggle" id="menu-toggle">  
9         <i class="fas fa-bars"></i>  
10      </button>  
11    </div>  
12    <div class="sidebar-content">  
13      <h3>Visualization Libraries</h3>  
14      <ul class="library-list">  
15        <li>  
16          <button class="library-btn active" data-library="matplotlib">  
17            <i class="fas fa-chart-bar"></i>  
18            <span>Matplotlib</span>  
19          </button>  
20        </li>  
21        <!-- Other Library buttons... -->  
22      </ul>  
23    </div>  
24  </nav>  
25  
26  <main class="main-content">  
27    <!-- Dashboard content... -->  
28  </main>  
29 </div>
```



## Dynamic Visualization Loading

JavaScript handles the loading and display of visualizations:

```
1  function loadVisualizations(library) {
2    // Store selected library
3    localStorage.setItem('selectedLibrary', library);
4
5    // Hide welcome message and show loading
6    welcomeMessage.style.display = 'none';
7    loadingIndicator.style.display = 'flex';
8    visualizationContainer.innerHTML = '';
9
10   // Send request to backend
11   const formData = new FormData();
12   formData.append('library', library);
13
14   fetch('/get_visualizations', {
15     method: 'POST',
16     body: formData
17   })
18   .then(response => response.json())
19   .then(data => {
20     loadingIndicator.style.display = 'none';
21     if (data.error) {
22       showErrorMessage(data.error);
23       return;
24     }
25     displayVisualizations(data);
26   })
27   .catch(error => {
28     loadingIndicator.style.display = 'none';
29     showErrorMessage('Failed to load visualizations. Please try again.');
```

```
30     console.error('Error:', error);
31   });
32 }
```



## 6. Key Visualizations and Insights:

### Customer Demographics

The age distribution visualizations reveal that most customers fall in the 30-60 age range, with a significant concentration around 35-45 and 50-60 years old. This bimodal distribution suggests two key customer segments to target with different marketing strategies.

### Purchase Patterns

The "Purchase Amount by Category" visualization shows that:

- Footwear has the highest average purchase amount (\$80+)
- Accessories show the most variability in spending
- Clothing items maintain consistent mid-range purchases

### Payment Preferences

The payment method distribution reveals:

- Credit cards are the most popular payment method (40%)
- Digital payment options (PayPal, Venmo) are gaining traction (35% combined)
- Cash payments represent a smaller but significant portion (15%)

### Seasonal Trends

The seasonal purchase pattern visualization demonstrates:

- Spring has the highest purchase activity
- Winter shows the highest average purchase amounts
- Summer purchases have the most diversity in product categories

## 7. Implementation Challenges and Solutions:

### Challenge 1: Responsive Visualization Display

**Problem:** Different visualization libraries produce different image sizes and formats that didn't consistently scale well on various devices.

**Solution:** Implemented standardized card-based layouts with CSS Grid and responsive design techniques:

```

1  .visualization-container {
2      display: grid;
3      grid-template-columns: repeat(auto-fit, minmax(500px, 1fr));
4      gap: 30px;
5  }
6
7  .visualization-card {
8      background-color: var(--card-bg);
9      border-radius: var(--border-radius-md);
10     box-shadow: var(--shadow-md);
11     overflow: hidden;
12     transition: transform 0.3s ease, box-shadow 0.3s ease;
13 }

```

## Challenge 2: Integrating Plotly's Interactive Visualizations

**Problem:** Plotly's interactive visualizations required special handling compared to static image-based charts.

**Solution:** Created a dual-approach rendering system in JavaScript:

```


1  if (data.library === 'plotly') {
2      // Special handling for Plotly charts
3      data.visualizations.forEach(viz => {
4          // Create card with Plotly-specific container
5          const vizCard = document.createElement('div');
6          vizCard.className = 'visualization-card';
7
8          // Add Plotly-specific rendering logic
9          const plotElement = vizCard.querySelector(`#plotly-${viz.title.replace(/\s+/g, '-').toLowerCase()}`);
10         // Rendering logic...
11     });
12 } else {
13     // Standard handling for static images
14     data.visualizations.forEach(viz => {
15         // Create standard image-based visualization card
16     });
17 }

```

## User Experience Features

## Theme Switching


The application supports both light and dark themes, toggled with a simple button click. The theme preference is stored in localStorage for a consistent experience across visits:



```
1 // Check for saved theme preference
2 const savedTheme = localStorage.getItem('darkTheme');
3 if (savedTheme === 'true') {
4     document.body.classList.add('dark-theme');
5     themeToggle.innerHTML = '<i class="fas fa-sun"></i>';
6 }
```

## Information Panel

A slide-out information panel provides dataset context and explanation:



```
1 datasetInfoToggle.addEventListener('click', function() {
2     datasetInfoPanel.classList.add('active');
3 });
4
5 closePanel.addEventListener('click', function() {
6     datasetInfoPanel.classList.remove('active');
7 });
```

## Download Functionality

Each visualization can be downloaded for use in presentations or reports:

```
1 function downloadImage(imageData, title) {
2   const formData = new FormData();
3   formData.append('image_data', imageData);
4   formData.append('chart_title', title);
5
6   fetch('/download_image', {
7     method: 'POST',
8     body: formData
9   })
10  .then(response => response.blob())
11  .then(blob => {
12    const url = window.URL.createObjectURL(blob);
13    const a = document.createElement('a');
14    a.href = url;
15    a.download = `${title}.png`;
16    a.click();
17
18    // Show download success notification
19    showNotification('Chart downloaded successfully!', 'success');
20  })
21  .catch(error => {
22    showNotification('Failed to download image', 'error');
23  });
24 }
```

## 8. Business Insights:

The visualizations reveal several actionable business insights:

1. **Target demographic:** Marketing efforts should focus on the 35-45 and 50-60 age groups, which represent the core customer base.
2. **Product pricing strategy:** Footwear items command higher prices and should be positioned as premium offerings.

3. **Seasonal marketing:** Spring campaigns should emphasize variety, while winter campaigns should focus on high-value items.
4. **Payment optimization:** Digital payment methods should be expanded and promoted, as they show growing adoption rates among customers.
5. **Review correlation:** Higher-priced items don't necessarily correlate with higher review ratings, suggesting quality improvement opportunities in premium product lines.

## 9. Future Enhancements:

This project has several potential avenues for expansion:

1. **Machine Learning Integration:** Add predictive analytics for sales forecasting and customer segmentation.
2. **Real-time Data Updates:** Connect to live databases or APIs for continuously updated visualizations.
3. **Advanced Filtering:** Add UI controls to filter visualizations by demographic or product attributes.
4. **Comparison Tools:** Implement features to compare time periods or customer segments side-by-side.
5. **Export Options:** Add functionality to export insights as PDF reports or data in various formats.

## 10. Conclusion:

This Customer Shopping Trends Analysis dashboard demonstrates the power of combining multiple visualization libraries with a responsive, user-friendly interface. By providing multiple perspectives on the same dataset, it enables deeper insights into customer behavior and preferences.

The project showcases how modern web technologies and data visualization techniques can transform raw retail data into actionable business intelligence. Through careful attention to user experience, performance, and visual communication, the dashboard makes complex data accessible and meaningful to business stakeholders.

Whether you're analyzing customer demographics, purchase patterns, or seasonal trends, this visualization platform provides the tools needed to drive data-informed business decisions in the retail sector.