International Review of Business, Trade, and Economics
ISSN: 3069-230X

PRIME OPEN ACCESS

Volume 2, Issue 2

Research Article

Date of Submission: 05 May, 2026
Date of Acceptance: 12 Jun, 2026
Date of Publication: 23 Jun, 2026

Customer Purchase Behavior Analysis and Visualization Using Big Data Analytics: A PySpark-Based Apache Spark Framework

Pritam Chaudhari[1]* and Poonam Sawant[2]

[1]Assistant Professor, Pune Institute of Business Management, (PIBM), India

[2]Assistant Professor, Sinhgad Institute of Management and Computer Applications (SIMCA), India

***Corresponding Author:** Pritam Chaudhari, Assistant Professor, Pune Institute of Business Management, (PIBM), India

Citation: Chaudhari, P., Sawant, P. (2026). Customer Purchase Behavior Analysis and Visualization Using Big Data Analytics: A PySpark-Based Apache Spark Framework. *Int Rev Bus Trade Econ, 2*(2), 01-16.

Abstract
Understanding customer purchase behaviour is critical for effective business decision-making and targeted marketing. The exponential growth of transactional and behavioural data challenges traditional analytics methods due to high volume, velocity, and variety. This study presents a scalable framework for analysing and visualizing customer purchase behaviour using PySpark (Apache Spark based Python API). Leveraging distributed system, the framework efficiently processes large-scale datasets to identify patterns across product categories, geographic regions, and customer segments. The methodology combines exploratory data analysis, aggregation, and visual analytics techniques to deliver actionable insights for marketing strategies, inventory optimization and operational planning. Experimental evaluation on both simulated and real-world datasets demonstrates the framework's scalability, performance, and capability in uncovering meaningful purchase behaviour trends. This approach advances big data analytics by integrating high-performance distributed processing with intuitive visualization techniques to enhance customer behaviour intelligence.

Keywords: Big Data Analytics, Customer Purchase Behaviour, PySpark, Apache Spark, Visual Analytics, Distributed Computing, Transactional Data, Data Visualization, Market Intelligence

Introduction
In today's digital economy, businesses generate a massive amount of data through customer transactions, online interactions, and purchasing activities. Every purchase made by a customer provides valuable information about preferences, buying patterns, and market trends. Analyzing this data effectively helps organizations improve marketing strategies, manage inventory efficiently, and enhance customer satisfaction [1,2]. However, the rapid growth of data in terms of size and complexity makes it difficult for traditional data processing tools to deliver timely and meaningful insights [1].

Customer purchase behaviour analysis focuses on understanding what customers buy, how often they buy, and how their purchasing patterns vary across products and locations [3]. Such analysis becomes more challenging when datasets grow to millions of records, as conventional single-machine systems struggle with performance and scalability [4]. This challenge highlights the need for Big Data analytics frameworks that can process massive amount of data quickly and reliably [1,5].

Apache Spark has emerged as a powerful Big Data processing framework due to its distributed architecture and in-memory computation capabilities [5,6]. PySpark, the Python interface for Apache Spark, allows analysts and researchers to perform large-scale data analysis using simple and readable Python code while benefiting from Spark's high-performance engine [7]. It supports efficient data loading, transformation, aggregation, and advanced analytics across large datasets [6].

This study presents a PySpark-based Apache Spark framework for analyzing and visualizing customer purchase behaviour. By applying distributed data processing and visual analytics techniques, the research demonstrates how meaningful patterns can be extracted from large transactional datasets [6,7]. The proposed approach bridges the gap between Big

Data technology and practical business analytics by providing clear, scalable, and data-driven insights into customer purchasing trends [2,3].

Big Data and Apache Spark Overview

Big Data Characteristics is identified by [1,4].

- **Volume:** Millions of transactions generating massive amount of data.

- **Velocity:** Live Streaming Data (Continuous flow of the data)

- **Variety:** The presence of diverse data sources encompassing structured, semi-structured, and unstructured formats.

Apache Spark Overview
Apache Spark offeres a powerful platform for Big Data challenges through [5,6].
- Distributed computation across cluster nodes.
- In-memory data processing for faster computation.
- Resilient Distributed Datasets (RDDs) and DataFrames for fault-tolerant storage and optimized query execution.
- Spark SQL and DataFrame API for advanced analytics.
- Spark MLlib for scalable machine learning. - Structured Streaming for processing real-time data.

Objectives of the Study

The primary objective of the study is to design scalable framework and implement the framework which is based on PySpark (Python API based Apache Spark framework) to analyze and visualize large-scale customer transaction data, enabling efficient extraction of meaningful insights to support data-driven business decision-making.

This research aims to bridge the gap between theoretical Big Data analytics frameworks and practical business applications by providing a detailed, data-driven analysis of customer purchase Behaviour.

Literature Review

The rapid advancement of digital platforms and information systems has led to the continuous generation of large-scale customer-related data. Sources such as transactional logs, online interactions, and purchase histories provide critical insights into consumer decision-making processes and Behavioural patterns [1,2]. Leveraging this data has become a cornerstone of modern business analytics, supporting strategic decisions in marketing optimization, customer engagement, and operational planning [2,3]. Nevertheless, the growing volume, diversity, and complexity of such datasets present substantial challenges to traditional analytical methods, which were originally designed for small and structured data environments [1,4].

Customer Purchase Behaviour Analysis

Initial research on customer purchase behaviour predominantly employed statistical analysis and conventional data mining techniques to identify purchasing trends, product associations, and customer preferences [3]. While these approaches laid the groundwork for consumer behaviour analysis, their effectiveness was largely confined to modestly sized datasets with limited feature complexity [4]. As organizations began accumulating large-scale transactional data, these methods struggled to deliver timely and scalable analytical outcomes.

To overcome these limitations, subsequent studies introduced customer segmentation models based on behavioural indicators such as purchase frequency and monetary value [3]. These models improved targeted marketing initiatives by enabling more refined customer classification. However, their implementation typically relied on centralized computing infrastructures, which restricted scalability and responsiveness in high-volume data environments [4]. With transactional datasets expanding into millions of records, centralized systems increasingly failed to meet the demands of real-time business analysis [1].

More recent investigations have emphasized multi-dimensional analysis by incorporating temporal, geographic, and product-level variables to obtain a comprehensive understanding of customer purchasing behaviour [8]. Although these approaches enhanced analytical depth, many studies continued to depend on legacy processing architectures that are not optimized for handling large-scale data, thereby limiting their applicability in Big Data contexts [4,9].

Significance of Big Data Analytics in Business Intelligence

Big Data analytics has redefined business intelligence by enabling organizations to process and analyze datasets characterized by substantial volume, velocity, and variety [1]. Unlike traditional analytics, Big Data methodologies facilitate the identification of intricate patterns and relationships that are otherwise difficult to detect [2]. However, conventional relational databases and single-node systems lack the computational capacity required to efficiently manage and analyze such data, resulting in performance bottlenecks and scalability constraints [4].

Early distributed computing models addressed scalability concerns by distributing processing workloads across multiple nodes. Despite this improvement, their reliance on disk-based computation led to high latency, reducing their effectiveness for iterative and exploratory analysis [5,6]. Such limitations are particularly problematic in customer behaviour analytics, where repeated data exploration is essential for generating meaningful insights.

The emergence of in-memory computing frameworks marked a significant advancement in Big Data analytics. By minimizing disk input/output operations, in-memory processing enables faster execution and supports interactive analytics, making near real-time business intelligence increasingly attainable [6,10].

Apache Spark and PySpark for Distributed Data Analytics

Apache Spark has become a prominent distributed analytics framework due to its in-memory processing capabilities, fault-tolerant design, and support for parallel computation across clustered environments [5,6]. Its architecture allows efficient handling of complex analytical tasks while maintaining scalability and reliability, making it well suited for large-scale data analysis.

PySpark extends the functionality of Apache Spark by offering a Python-based interface, allowing analysts and researchers to utilize Python's flexibility while leveraging distributed computing capabilities [7]. Existing studies highlight PySpark's effectiveness in reducing development time and facilitating rapid experimentation in both academic and industry settings [7,10]. Compared to lower-level distributed programming models, PySpark provides an effective balance between computational performance and ease of use.

Despite these advantages, much of the existing literature on Spark and PySpark concentrates on system performance evaluation, machine learning workflows, or predictive analytics applications [6,10]. The application of PySpark for large-scale descriptive and exploratory analysis of customer purchase behaviour, particularly when integrated with visualization techniques for business interpretation, remains relatively underrepresented [11].

Distributed Analytics in Customer-Centric Applications

Distributed analytics frameworks have been extensively applied in customer-oriented domains such as recommendation engines, sales forecasting, and churn analysis [8,9]. These studies demonstrate the ability of distributed systems to process large transactional datasets efficiently and support advanced analytical modeling.

However, the majority of existing research prioritizes predictive modeling, often at the expense of descriptive and exploratory analytics [9]. While predictive insights are valuable, businesses also require transparent analytical summaries that reveal current purchasing patterns, product performance, and regional variations [2]. The limited focus on interpretability in distributed analytics reduces the practical usability of such systems for managerial decision-making [11].

Additionally, comprehensive analysis that integrates multiple dimensions—such as product categories, customer segments, and geographic regions—remains insufficiently explored within distributed analytics research, constraining the development of localized and data-driven marketing strategies [8,11].

Visual Analytics for Interpreting Customer Behaviour

Visual analytics plays a vital role in converting complex analytical results into interpretable insights that support informed decision-making. Visualization methods, including bar charts, pie charts, and multi-dimensional plots, enable stakeholders to quickly understand trends, patterns, and anomalies within large datasets [12].

Prior studies indicate that visual dashboards significantly enhance user comprehension and engagement, leading to improved decision quality [12,13]. However, many visualization-centric studies rely on aggregated or sampled datasets, which limits their scalability and relevance in Big Data environments [11]. Despite the growing importance of visual analytics, the seamless integration of distributed data processing with visualization workflows remains underexplored.

Moreover, the dependence on proprietary visualization platforms raises concerns related to cost, transparency, and reproducibility. This underscores the need for open-source, scalable analytical frameworks that combine distributed data processing with flexible and interpretable visualization capabilities [7,11,13].

Identified Research Gaps

Prior research in marketing analytics has made substantial progress in examining customer purchase behaviour through statistical methods, data mining techniques, and predictive modeling approaches. Despite these advancements, notable gaps persist in the existing literature. Most studies prioritize prediction-oriented outcomes such as forecasting, recommendation, and churn analysis, while comparatively little attention is given to descriptive and exploratory analytics that provide an in-depth understanding of existing purchasing behaviour. This limits the ability of decision-makers to interpret current market dynamics and customer preferences effectively.

In addition, although distributed computing technologies such as Apache Spark are widely recognized for their scalability, existing studies largely focus on performance evaluation or machine learning applications. The systematic application of PySpark for large-scale, descriptive customer behaviour analysis—particularly involving multi-dimensional aggregation across products, regions, and time—remains insufficiently explored. Consequently, there is a lack of analytical frameworks that balance computational scalability with interpretability and business relevance.

Furthermore, the integration of distributed data processing with visual analytics has not been adequately addressed in current research. Many studies depend on pre-processed datasets or proprietary visualization tools, which restrict scalability, transparency, and reproducibility. Limited emphasis on integrated, open-source solutions also constrains the practical adoption of analytics-driven insights for localized and data-informed marketing strategies.

These gaps collectively highlight the need for a scalable and interpretable analytical approach that combines distributed processing, descriptive customer behaviour analysis, and visualization to support effective business decision-making.

Contribution to Marketing and Analytics Knowledge

This study contributes to the literature by addressing the identified gaps and extending knowledge in both marketing and analytics domains.

From a marketing perspective, the study advances understanding of customer purchase behaviour by offering a scalable framework for descriptive and exploratory analysis of transactional data. By analyzing purchasing patterns across multiple dimensions such as product categories, geographic regions, and temporal trends, the findings provide actionable insights that support strategic marketing planning, customer segmentation, and regional performance assessment. The emphasis on interpretability enhances the managerial usefulness of analytics, complementing existing predictive-focused research.

From an analytics perspective, the study demonstrates the practical application of PySpark as an open-source, distributed analytics framework for large-scale customer behaviour analysis. It contributes by illustrating how in-memory distributed processing can be effectively combined with multi-dimensional aggregation and visualization to produce scalable yet interpretable analytical outputs. This approach bridges the gap between high-performance data processing and human-centered analytical interpretation, offering a reproducible and transparent solution for Big Data analytics.

Overall, the study enriches existing literature by integrating scalable analytics techniques with descriptive marketing insights. The findings extend beyond predictive modelling by providing a structured framework that enables organizations to transform large-scale customer data into meaningful and actionable business intelligence.

Positioning of the Study

The current study addresses the gaps by proposing PySpark-driven Apache Spark framework that combines distributed data processing with visual analytics to analyze customer purchase behaviour. By focusing on multi-dimensional aggregation across product categories and geographic regions, the framework delivers interpretable and actionable insights suitable for real-world business applications. This approach strengthens the role of Big Data analytics as a practical decision-support tool rather than a purely technical solution.

Problem Statement

Modern businesses generate large volumes of customer transaction data that contain valuable insights into purchasing patterns and market behaviour. However, traditional based data processing and analytics tools fail to operate such large-scale datasets efficiently due to limitations in scalability, performance, and processing speed. As a result, timely and comprehensive analysis of customer purchase behaviour becomes challenging.

Existing analytics approaches frequently lack support for multi-dimensional analysis across product categories, geographic regions, and purchase volumes, and they often provide limited integration with effective visualization techniques. This restricts the ability of decision-makers to interpret analytical outcomes and apply them to real-world business strategies.

To address these challenges, there is a need for a scalable and distributed analytics framework that can efficiently process large customer transaction datasets, perform comprehensive purchase behaviour analysis, and present insights through intuitive visualizations. This study proposes a PySpark-based Apache Spark framework to fulfill these requirements and enhance data-driven business decision-making.

Research Objective

- Design and implement a distributed analytics framework using Apache Spark and PySpark for processing large-volume customer transaction datasets.
- Perform multi-dimensional analysis across product categories, geographic locations, and purchase volumes to identify significant trends and Behavioural patterns.
- Apply advanced aggregation, segmentation, and pivot-based analytical techniques to derive actionable business insights.

• Employ visual analytics methods to effectively represent aggregated and multi-dimensional customer purchase data, thereby enhancing interpretability and insight discovery.

Dataset Description

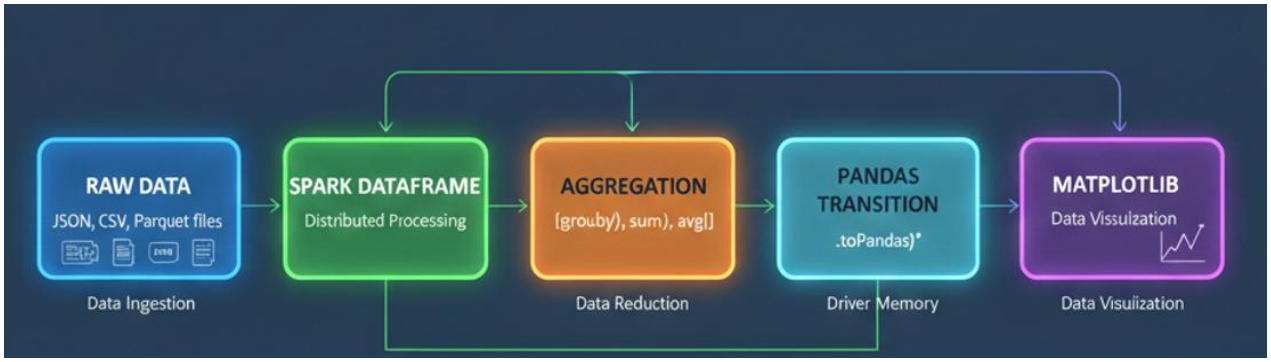
This study uses a transactional dataset containing 1,000 customer purchase records collected from a digital commerce transaction system. Each record represents a single purchase and includes key attributes such as customer and order identifiers, product category, purchase amount, quantity, city, and transaction date. All identifiers are anonymized to ensure data privacy.

The data was generated through routine transaction logging during 2025, reflecting real-world purchasing activity across multiple product categories and geographic locations. While personal demographic details (e.g., age or gender) are not included, the city-level information enables regional analysis of customer purchasing patterns.

The structured and complete nature of the dataset supports reliable descriptive and exploratory analysis. Its design is well suited for evaluating the applicability of distributed analytics frameworks, allowing readers to assess the validity and relevance of the findings in comparable retail and e-commerce contexts.

System Architecture

Raw Data to Visual Insights: Spark to Visualization Pipeline:



*Ensure Data is Aggregated before converting to Pandas

Figure

*Ensure Data is Aggregated before converting to Pandas

Stage	Action	Purpose
Raw Data	Ingesting JSON, CSV, or Parquet.	Loading the source of truth from a Data Lake or HDFS.
Spark DataFrame	Distributed Processing.	Parallelizing the data across a cluster for high-speed handling.
Aggregation	groupBy(), agg(), count()	Reducing billions of rows into meaningful summary statistics.
Pandas Transition	.toPandas()	Moving the summarized (smaller) data from the cluster to the driver's memory.
Matplotlib	plt.plot(), plt.show()	Generating charts, histograms, or heatmaps for human interpretation.

Table

Methodology

This study employs a Big Data analytics framework using Apache Spark and PySpark to analyze and visualize large-scale customer transaction data. Apache Spark is chosen for its distributed and in-memory processing capabilities, enabling efficient handling of high-volume datasets.

Customer transaction data in structured format is loaded into Spark using the DataFrame API with automatic schema inference. Data preprocessing includes removal of missing values and standardization of numerical attributes to ensure data consistency and accuracy.

Multi-dimensional analysis is performed across product categories, cities, and purchase volumes using distributed aggregation and grouping operations. Pivot-based analysis is applied to examine relationships between geographic regions and product categories, enabling deeper insight into customer purchasing patterns.

Analytical results are transformed into visual representations using bar charts, pie charts, and stacked bar charts. These visualizations support clear interpretation of trends, regional performance, and category-wise sales distribution.

The framework leverages Spark's parallel execution, in-memory computation, and fault tolerance to ensure scalability, high performance, and reliability, making it suitable for real-world customer purchase Behaviour analysis.

Implementation

Environment Setup and PySpark Session Initialization

```
From pyspark.sql import SparkSession
spark = SparkSession.builder\
.appName("CustomerPurchaseAnalysis")\
.master("local[*]")\
.getOrCreate()
```

Insights: This setup enables distributed computation across available cores and initializes Spark for data analysis.

Data Loading

```
df = spark.read.option("header", "true") \
.option("inferSchema", "true") \
.csv("customer_purchases_large.csv")
df.show(10)
```

Insights: Spark efficiently loads large-scale structured datasets, inferring schema and handling diverse data types.

Data Cleaning and Preprocessing

```
# Remove NA values
df = df.na.drop()
# Round up purchase amounts
from pyspark.sql.functions import ceil, col
df = df.withColumn("Purchase_Amount", ceil(col("Purchase_Amount")))
df.show(10)
```

Insights: Ensures dataset consistency, removes anomalies, and prepares for accurate aggregation.

Data Analysis

Category-wise Sales Analysis

```
from pyspark.sql.functions import sum, desc
sales_category = df.groupBy("Product_Category") \
.agg(sum("Purchase_Amount").alias("Total_Sales")) \
.orderBy(desc("Total_Sales"))
sales_category.show()
```

Insights: Identifies high-revenue product categories. - Guides strategic inventory allocation and marketing campaigns. - Enables comparative performance evaluation across categories.

City-Wise Sales Analysis

```
city_sales = df.groupBy("City") \
.agg(sum("Purchase_Amount").alias("Citywise_Sales")) \
.orderBy(desc("Citywise_Sales"))
city_sales.show()
```

Insights: Determines top-performing cities and potential growth areas. - Supports region-specific marketing and promotional strategies. - Helps identify underperforming regions for targeted interventions.

City and Category Cross-analysis

```
city_category_sales = df.groupBy("City", "Product_Category") \
.agg(sum("Purchase_Amount").alias("Total_Sales"))
city_category_sales.show()
```

Insights: Reveals city-specific product preferences. - Assists in regional inventory planning and category-focused marketing. - Identifies emerging trends in specific markets.

Data Visualization

Category-Wise Bar Chart

```
import matplotlib.pyplot as plt
sales_pd = sales_category.toPandas()
colors = ["skyblue", "orange", "green", "red", "purple"]
plt.bar(sales_pd["Product_Category"], sales_pd["Total_Sales"], color=colors)
plt.xlabel("Product Category")
plt.ylabel("Total Sales")
plt.title("Category-wise Total Sales")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

Insight: Enables quick comparison of category performance and highlights areas for business focus.

City-wise Pie Chart

```
city_pd = city_sales.toPandas()
plt.pie(city_pd["Citywise_Sales"], labels=city_pd["City"], autopct="%1.1f%%",
tartangle=140, colors=colors)
plt.title("City-wise Sales Distribution")
plt.axis("equal")
plt.tight_layout()
plt.show()
```

Insight: Visualizes market share by city and informs geographic prioritization.

Stacked Bar Chart for City-Category Sales

```
cc_pd = city_category_sales.toPandas()
pivot_df = cc_pd.pivot(index="City", columns="Product_Category", values="Total_Sales").fillna(0)
pivot_df.plot(kind="bar", stacked=True)
plt.xlabel("City")
plt.ylabel("Total Sales")
plt.title("City-wise Category Sales")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

Insight: Illustrates the contribution of each product category to city-wise revenue, guiding inventory allocation and promotions.

Result and Analysis

Category-Wise Sales Analysis

Product_Category	Total Sales
Home & Kitchen	546981
Electronics	518446
Books	512708
Fashion	501020
Groceries	472555

Table

The analysis highlights how total sales are distributed across different product categories, revealing important insights into customer purchasing behaviour:
• Home & Kitchen records the highest total sales (546,981), indicating strong and consistent demand. This suggests that customers prioritize household and utility-based products, making this category a key revenue driver.
• Electronics follows closely with 518,446 in total sales, reflecting sustained consumer interest in technology and electronic goods. The relatively small gap with the top category shows high competitiveness and potential for growth through promotions or new product launches.
• Books contribute 512,708 in total sales, demonstrating stable demand. This indicates a loyal customer base and consistent purchasing patterns, possibly driven by education, professional needs, or leisure reading.
• Fashion accounts for 501,020 in total sales, showing moderate but steady performance. Sales in this category may be influenced by seasonal trends, discounts, and changing customer preferences.
• Groceries, with 472,555 in total sales, generate the lowest revenue among the listed categories. While frequently purchased, grocery items may have lower margins or smaller transaction values compared to other categories.

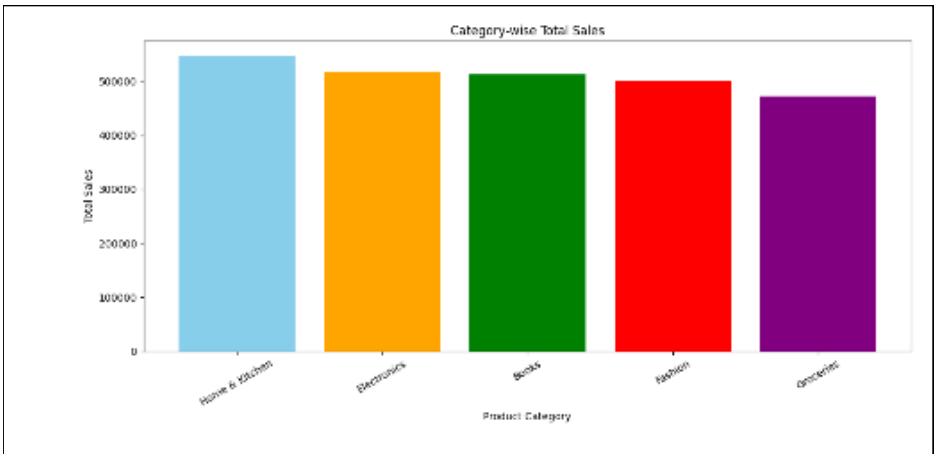
City-wise Sales Analysis

City	Citywise Sales
Bangalore	405336
Mumbai	397773
Delhi	371322
Chennai	361986
Hyderabad	350619
Pune	346648
Kolkata	318026

Table

The city-wise sales analysis reveals how total sales are distributed across major metropolitan cities, providing insights into regional demand patterns:
• Bangalore records the highest total sales (405,336), indicating strong customer purchasing power and high adoption of online or organized retail. This suggests Bangalore is a key market with significant revenue potential.
• Mumbai follows closely with 397,773 in total sales, reflecting its large population, high income levels, and diverse consumer base. The narrow difference with Bangalore indicates intense competition among top-tier cities.
• Delhi contributes 371,322 in total sales, showing robust demand driven by a wide consumer demographic and strong retail activity in the national capital region.
• Chennai accounts for 361,986 in total sales, indicating steady and consistent purchasing Behaviour, possibly influenced by a balanced mix of essential and discretionary spending.
• Hyderabad and Pune, with 350,619 and 346,648 respectively, demonstrate growing markets. Their sales figures reflect increasing urbanization, IT-sector growth, and rising disposable incomes.
• Kolkata shows the lowest total sales (318,026) among the listed cities. While demand exists, this may indicate relatively lower purchasing power or fewer high-value transactions compared to other metros.

Category-wise Bar Chart

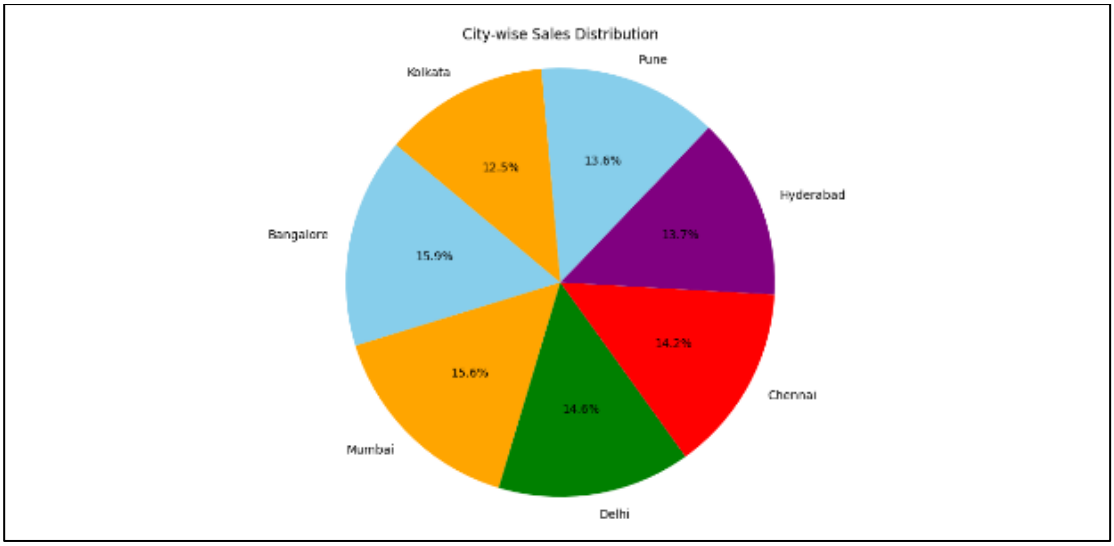


Figure

The bar chart illustrates the relative contribution of each product category to overall sales, emphasizing comparative performance rather than absolute values:

- The sales distribution is fairly balanced across categories, as no single category overwhelmingly dominates the chart. This indicates a diversified revenue structure that reduces dependency on any one product line.
- Home & Kitchen, while leading, shows only a moderate margin over other categories, suggesting healthy competition and similar customer engagement levels across categories.
- Electronics, Books, and Fashion form a middle-performing cluster, with closely aligned bar heights. This reflects overlapping customer demand patterns and indicates that shifts in pricing, promotions, or availability could easily change their relative rankings.
- Groceries, though lowest in total sales, still contributes a substantial share, implying high transaction frequency but lower average order value compared to other categories.

City-wise Pie Chart

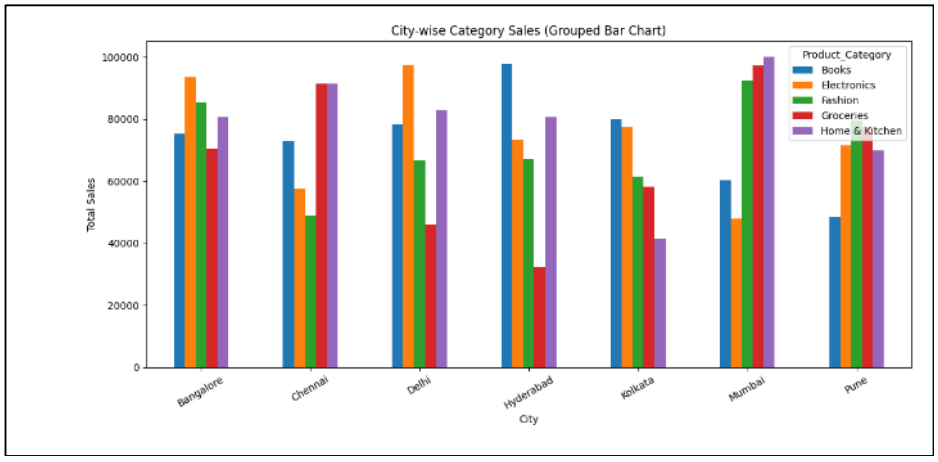


Figure

The pie chart presents the proportional contribution of each city to total sales, offering a percentage-based perspective rather than absolute sales values:

- Bangalore holds the largest share of total sales, but its slice is only marginally bigger than others. This indicates leadership without dominance, suggesting competitive parity among major cities.
- Mumbai closely follows Bangalore, reinforcing its role as a consistently strong market with comparable purchasing activity.
- Delhi and Chennai occupy mid-sized slices, reflecting stable and predictable demand patterns across these metropolitan regions.
- Hyderabad and Pune contribute similar proportions, highlighting their status as emerging growth hubs with increasing consumer participation.
- Kolkata, while contributing the smallest percentage, still represents a meaningful share of overall sales, indicating market presence but comparatively lower transaction value or volume.

Stacked Bar Chart for City-Category Sales



Figure

The grouped bar chart provides a comparative view of product category performance across different cities, highlighting both regional preferences and category-level variations:
• Category preferences vary significantly by city, indicating that customer demand is not uniform across regions. Each city shows a distinct mix of high- and low-performing categories.
• Books perform strongly in Hyderabad and Kolkata, suggesting higher demand for educational or knowledge-based products in these cities compared to others.
• Electronics sales are prominent in Delhi and Bangalore, reflecting tech-oriented consumer behaviour and higher spending on electronic goods in these metropolitan areas.
• Fashion shows higher sales in Mumbai and Pune, indicating trend-driven consumption and stronger apparel demand in western urban markets.
• Groceries exhibit relatively strong performance in Chennai and Mumbai, suggesting higher reliance on frequent, essential purchases in these cities.
• Home & Kitchen dominates in Mumbai and Chennai, highlighting strong demand for household and lifestyle products in these regions.

Discussion and Implications of the Findings

The outcomes of this study demonstrate that large-scale descriptive analysis of transactional data can generate meaningful insights into customer purchasing behaviour. The results reaffirm the relevance of behavioural evidence in explaining consumer preferences and extend existing perspectives by illustrating how such evidence can be examined effectively within high-volume data settings.

From a theoretical perspective, the findings are generally consistent with earlier studies that identify purchasing frequency, product choice, and regional variation as important drivers of consumer behaviour. However, this research departs from much of the prior work by shifting the analytical focus away from prediction-oriented models toward descriptive exploration. The results indicate that scalable descriptive analytics can uncover valuable behavioural patterns without relying on complex predictive algorithms, thereby broadening current marketing theory to incorporate interpretability and analytical transparency as central components of data-driven research.

From a managerial and practical standpoint, the study provides actionable insights that support marketing decision-making. The multi-dimensional examination of customer transactions enables organizations to identify demand concentration across products and locations, facilitating more informed decisions related to promotional planning, inventory distribution, and regional market strategies. The reliance on open-source distributed analytics further enhances the practical relevance of the approach by offering a scalable and economically viable solution for analyzing large transactional datasets.

In contrast to existing studies that emphasize model accuracy or computational efficiency, this research highlights the importance of insight clarity and usability. While predictive approaches remain valuable, the findings demonstrate that descriptive analytics plays a complementary and essential role in understanding current market behaviour. By combining scalable data processing with interpretable analytical outputs, this work contributes a balanced perspective that strengthens the connection between marketing theory and analytics practice.

Overall, the study advances scholarly discourse by presenting an alternative analytical lens that integrates distributed data processing with behaviour-focused analysis. This approach enhances theoretical understanding while offering practical guidance for organizations seeking to translate large-scale customer data into meaningful marketing intelligence.

Conclusion

The study demonstrates that PySpark on Apache Spark is a powerful framework for large-scale customer purchase Behaviour analysis. Distributed and in-memory computing enables fast, scalable, and reliable analytics, while visualizations provide actionable insights for business decision-making.

Following are the overall conclusion and strategies for Customer Purchase Behaviour Dataset based on the Pyspark and Data Visualization implementation.
• The sales distribution suggests that Home & Kitchen and Electronics dominate overall revenue, while Books and Fashion maintain stable contributions. Groceries, despite high purchase frequency, contribute relatively lower total sales, indicating an opportunity to increase basket size or introduce bundled offers. These insights can support strategic decisions related to inventory planning, marketing focus, and category-wise promotions.
• The results indicate that Tier-1 metropolitan cities dominate overall sales, with Bangalore and Mumbai emerging as the strongest revenue contributors. Mid-range performers like Delhi and Chennai show stable demand, while Hyderabad and Pune represent emerging growth opportunities. Kolkata's lower sales suggest the need for targeted marketing strategies or localized promotions to improve market penetration.
• From a strategic perspective, the visualization suggests opportunities for cross-category bundling (e.g., groceries with household items) and upselling strategies to improve lower-performing categories. The near-uniform bar heights also imply that incremental improvements in any category could significantly impact overall revenue, making targeted optimization more effective than focusing on a single category alone.

• The near-uniform slice sizes reveal a well-balanced geographic sales distribution, reducing regional dependency risk. Unlike a bar chart that emphasizes ranking, the pie chart highlights relative contribution, showing that no single city overwhelmingly drives revenue. This suggests opportunities for region-specific marketing strategies and localized demand optimization to incrementally improve overall performance across all cities.
• The visualization clearly demonstrates that city-level market dynamics influence category performance. Unlike aggregate charts, this grouped view uncovers localized demand patterns, enabling businesses to adopt city-specific inventory planning, targeted promotions, and customized marketing strategies. Focusing on a city's top-performing categories can improve sales efficiency and customer satisfaction while minimizing overstocking in lower-demand segments.

Summary of Contribution

By applying a scalable and interpretable descriptive analytics approach to customer purchase behaviour, this research advances current discussions in marketing and analytics scholarship. The study connects conceptual perspectives with practical application by illustrating how extensive transactional data can be converted into actionable insights that benefit both theoretical inquiry and managerial decision-making. A comprehensive treatment of these aspects enhances the rigor, relevance, and overall contribution of the work.

Future Work

The proposed PySpark-based customer purchase behaviour analysis framework can be further enhanced in several directions. Predictive analytics models can be integrated to forecast customer demand, identify potential churn, and recommend products based on historical purchasing patterns. Incorporating machine learning techniques using Spark MLlib would enable more intelligent and automated decision-making.

The framework can be extended to support real-time analytics by leveraging Spark Structured Streaming to process live transactional data. This would allow businesses to monitor customer behaviour continuously and respond quickly to changing market conditions.

Advanced, interactive dashboards can be implemented to generate real-time, user-oriented insights. Moreover, integrating diverse external data sources such as customer demographics, social media indicators and seasonal trends enhances the correctness, analytical richness and interpretative depth of customer Behaviour analysis.

Additionally, the framework can be implemented on cloud-based platforms to enhance performance, cost efficiency, and accessibility for large enterprises. These enhancements would make the system more adaptive, intelligent, and suitable for dynamic, data-driven business environments.

Declaration

Funding: The authors received no external funding for this study

Clinical Trial Number: not applicable

Ethics, Consent to Participate, and Consent to Publish Declarations: not applicable

Author Contributions

Mr. Pritam Chaudhari conceptualized the study, designed the methodology, performed data analysis, implemented the PySpark framework and drafted the manuscript.

Dr. Poonam Sawant contributed to the research design, supervised the study, reviewed and refined the analysis, and critically revised the manuscript.

All authors read and approved the final manuscript

• Data Availability: The datasets analysed during the current study are not publicly available due the presence of customer-related personal and sensitive information but are available from the corresponding author on reasonable request.
• Ethics: This study did not involve human participants, human biological material, or live vertebrates. The datasets used consist of anonymized transactional data and do not contain any personally identifiable information. Therefore, ethical approval was not required.
• Consent to Participate: Not applicable. This study does not involve human participants, and no personal or identifiable information was collected.
• On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. Mobile networks and applications, 19(2), 171-209.
2. McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data. The management revolution. *Harvard Bus Rev*, *90*(10), 61-67.

3. J. Han, J. Pei, and M. Kamber, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann, 2011.
4. Zikopoulos, P., Deroos, D., Parasuraman, K., Deutsch, T., Giles, J., & Corrigan, D. (2012). *Harness the power of big data The IBM big data platform.* McGraw Hill Professional.
5. Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *In 2nd USENIX workshop on hot topics in cloud computing (HotCloud 10).*
6. Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache spark: a unified engine for big data processing. *Communications of the ACM, 59*(11), 56-65.
7. Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning spark: lightning-fast big data analysis.* " O'Reilly Media, Inc.".
8. R. Agrawal et al., "Mining association rules," Proc. VLDB, 1993.
9. J. Leskovec, A. Rajaraman, and J. Ullman, (2014) Mining of Massive Datasets, Cambridge Univ. Press.
10. T. White, (2015) Hadoop: The Definitive Guide, O'Reilly Media.
11. S. Few, Information Dashboard Design, Analytics Press, 2013.
12. B. Shneiderman, (1996) "The eyes have it: A task by data type taxonomy," IEEE VL, vol. 5, no. 2, pp. 336–343.
13. Heer, J., & Shneiderman, B. (2012). Interactive dynamics for visual analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue, 10*(2), 30-55.

Appendix

```
PS C:\Users\Admin> pyspark  #Start PySpark
>>> #Create SparkSession
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder \
....appName("CustomerPurchaseAnalysis") \
....master("local[*]") \
....getOrCreate()

>>> #Read CSV File
>>> df = spark.read \
....option("header", "true") \
....option("inferSchema", "true") \
....csv("C:/Users/Admin/Desktop/customer_purchases_large.csv")
>>> #Show Data
>>> df.show()
```

Customer_ID	Order_ID	Product_Category	Purchase_Amount	Quantity	City	Purchase_Date
C0001	O1001	Home & Kitchen	4667.38	1	Mumbai	10-05-2025
C0002	O1002	Books	2869.15	3	Hyderabad	08-02-2025
C0003	O1003	Groceries	3513.59	8	Chennai	23-03-2025
C0004	O1004	Books	4620.25	3	Kolkata	09-07-2025
C0005	O1005	Books	3565.47	8	Chennai	10-08-2025
C0006	O1006	Fashion	847.44	1	Chennai	31-03-2025
C0007	O1007	Groceries	2923.81	5	Delhi	22-01-2025
C0008	O1008	Groceries	3072.9	4	Mumbai	23-05-2025
C0009	O1009	Groceries	2178.24	5	Kolkata	27-03-2025
C0010	O1010	Books	3708.58	8	Hyderabad	22-03-2025
C0011	O1011	Home & Kitchen	4678.4	4	Delhi	17-12-2025
C0012	O1012	Groceries	4635.29	4	Bangalore	13-08-2025
C0013	O1013	Books	2309.11	3	Hyderabad	23-07-2025
C0014	O1014	Fashion	654.87	7	Pune	01-06-2025
C0015	O1015	Home & Kitchen	4925.72	3	Hyderabad	06-06-2025
C0016	O1016	Fashion	4210.6	3	Hyderabad	17-07-2025
C0017	O1017	Home & Kitchen	710.85	5	Chennai	12-09-2025
C0018	O1018	Books	4612.13	5	Hyderabad	14-09-2025
C0019	O1019	Electronics	4362.49	6	Pune	04-03-2025
C0020	O1020	Home & Kitchen	2642.31	8	Pune	02-07-2025

```
only showing top 20 rows
>>> #Drop NULL Values
>>> df = df.na.drop()
>>> df.show(10)
```

Customer_ID	Order_ID	Product_Category	Purchase_Amount	Quantity	City	Purchase_Date
C0001	O1001	Home & Kitchen	4667.38	1	Mumbai	10-05-2025
C0002	O1002	Books	2869.15	3	Hyderabad	08-02-2025
C0003	O1003	Groceries	3513.59	8	Chennai	23-03-2025
C0004	O1004	Books	4620.25	3	Kolkata	09-07-2025
C0005	O1005	Books	3565.47	8	Chennai	10-08-2025
C0006	O1006	Fashion	847.44	1	Chennai	31-03-2025
C0007	O1007	Groceries	2923.81	5	Delhi	22-01-2025
C0008	O1008	Groceries	3072.9	4	Mumbai	23-05-2025
C0009	O1009	Groceries	2178.24	5	Kolkata	27-03-2025
C0010	O1010	Books	3708.58	8	Hyderabad	22-03-2025

only showing top 10 rows

```
>>> #Round-up Purchase_Amount
>>> from pyspark.sql.functions import ceil, col
>>> df = df.withColumn("Purchase_Amount", ceil(col("Purchase_Amount")))
>>> df.show(10)
```

Customer_ID	Order_ID	Product_Category	Purchase_Amount	Quantity	City	Purchase_Date
C0001	O1001	Home & Kitchen	4668	1	Mumbai	10-05-2025
C0002	O1002	Books	2870	3	Hyderabad	08-02-2025
C0003	O1003	Groceries	3514	8	Chennai	23-03-2025
C0004	O1004	Books	4621	3	Kolkata	09-07-2025
C0005	O1005	Books	3566	8	Chennai	10-08-2025
C0006	O1006	Fashion	848	1	Chennai	31-03-2025
C0007	O1007	Groceries	2924	5	Delhi	22-01-2025
C0008	O1008	Groceries	3073	4	Mumbai	23-05-2025
C0009	O1009	Groceries	2179	5	Kolkata	27-03-2025
C0010	O1010	Books	3709	8	Hyderabad	22-03-2025

only showing top 10 rows

```
>>> #Category-wise Total Sales
>>> from pyspark.sql.functions import sum, desc
>>> salescategory = df.groupBy("Product_Category") \
....agg(sum("Purchase_Amount").alias("Total Sales")) \
....orderBy(desc("Total Sales"))
>>> salescategory.show()
```

Product_Category	Total Sales
Home & Kitchen	546981
Electronics	518446
Books	512708
Fashion	501020
Groceries	472555

```
>>> #City-wise Total Sales
>>> citysales = df.groupBy("City") \
....agg(sum("Purchase_Amount").alias("Citywise Sales")) \
....orderBy(desc("Citywise Sales"))
>>> citysales.show()
```

City	Citywise Sales
Bangalore	405336
Mumbai	397773
Delhi	371322
Chennai	361986
Hyderabad	350619
Pune	346648
Kolkata	318026

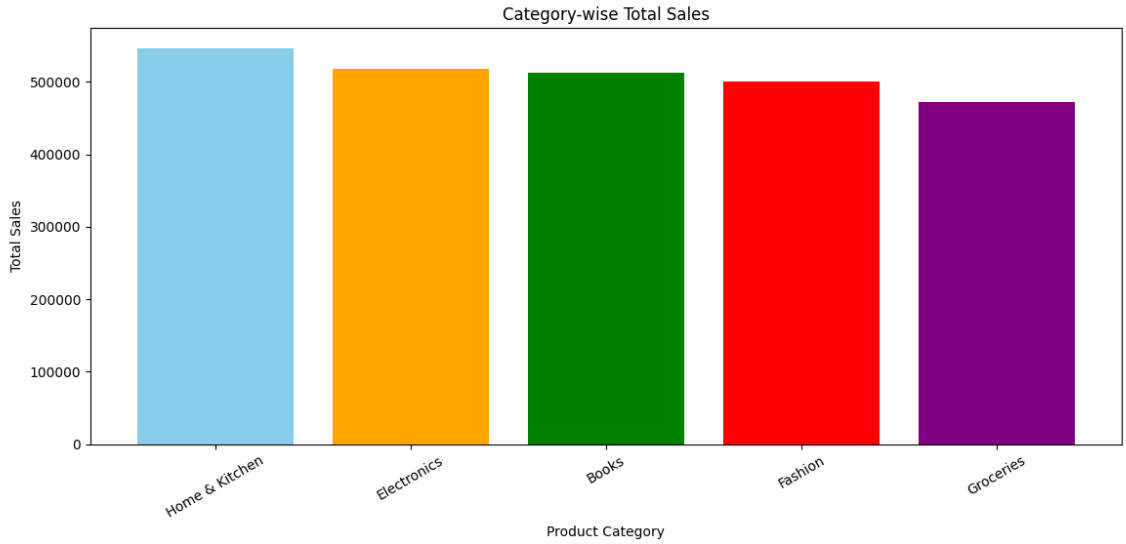
```
>>> #Convert Spark DataFrames to Pandas
>>> sales_pd = salescategory.toPandas()
>>> city_pd = citysales.toPandas()
>>> sales_pd
```

	Product_Category	Total Sales
0	Home & Kitchen	546981
1	Electronics	518446
2	Books	512708
3	Fashion	501020
4	Groceries	472555

```
>>> city_pd
```

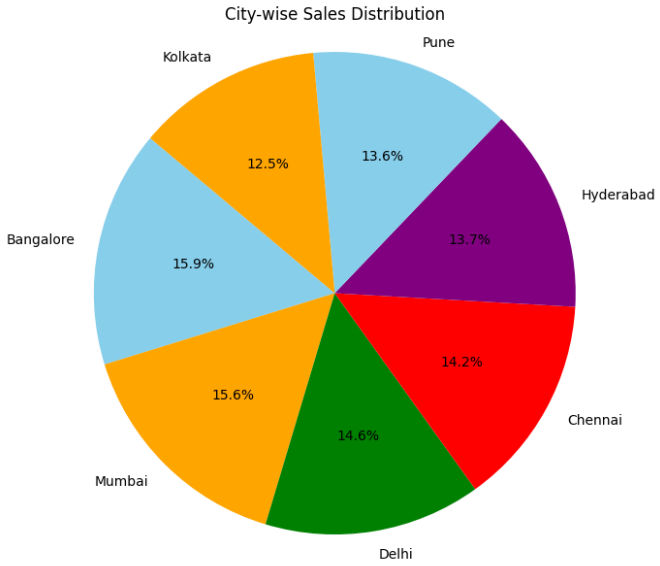
	Citre	Citywise Sales
0	Bangalore	405336
1	Mumbai	397773
2	Delhi	371322
3	Chennai	361986
4		350619
5	Pune	346648
6	Kolkata	318026

```
>>> #Import Matplotlib for Data Visualization
>>> import matplotlib.pyplot as plt
>>> #Visualization Category-wise Total Sales
>>> colors = ["skyblue", "orange", "green", "red", "purple"]
>>> plt.figure()
>>> plt.bar(sales_pd["Product_Category"],sales_pd["Total Sales"],color=colors)
>>> plt.xlabel("Product Category")
>>> plt.ylabel("Total Sales")
>>> plt.title("Category-wise Total Sales")
>>> plt.xticks(rotation=30)
>>> plt.tight_layout()
>>> plt.show()
```



Figure

```
>>> #Visualization City-wise Total Sales
>>> plt.figure()
>>> plt.pie(city_pd["Citywise Sales"],labels=city_pd["City"],
...autopct="%1.1f%%",
...startangle=140,
...colors=colors)
>>> plt.title("City-wise Sales Distribution")
>>> plt.axis("equal")
>>> plt.tight_layout()
>>> plt.show()
```



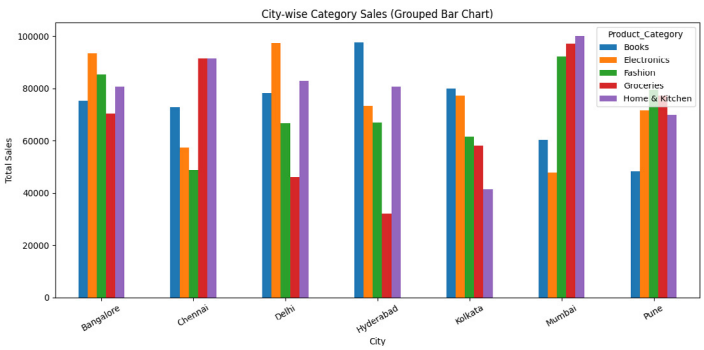
Figure

```
>>> #City-wise Category Sales
>>> from pyspark.sql.functions import sum
>>> city_category_sales = df.groupBy("City", "Product_Category") \
....agg(sum("Purchase_Amount").alias("Total Sales"))
>>> city_category_sales.show();
```

City	Product_Category	Total Sales
Hyderabad	Electronics	73265
Pune	Groceries	77325
Delhi	Fashion	66492
Chennai	Electronics	57436
Hyderabad	Books	97599
Pune	Fashion	79559
Mumbai	Home & Kitchen	100141
Bangalore	Books	75360
Bangalore	Home & Kitchen	80712
Bangalore	Groceries	70320
Chennai	Groceries	91337
Pune	Electronics	71600
Bangalore	Fashion	85423
Mumbai	Groceries	97226
Pune	Home & Kitchen	69783
Mumbai	Fashion	92283
Pune	Books	48381
Chennai	Books	72969
Delhi	Books	78316
Hyderabad	Fashion	66998

only showing top 20 rows

```
>>> #Convert to Pandas & Pivot
>>> cc_pd = city_category_sales.toPandas()
>>> pivot_df = cc_pd.pivot(index="City", columns="Product_Category", values="Total_Sales").fillna(0)
>>> pivot_df.plot(kind="bar", stacked=True)
>>> plt.xlabel("City")
>>> plt.ylabel("Total Sales")
>>> plt.title("City-wise Category Sales")
>>> plt.xticks(rotation=30)
>>> plt.tight_layout()
>>> plt.show()
>>> pivot_df.plot(kind="bar")
>>> plt.xlabel("City")
>>> plt.ylabel("Total Sales")
>>> plt.title("City-wise Category Sales (Grouped Bar Chart)")
>>> plt.xticks(rotation=30)
>>> plt.tight_layout()
>>> plt.show()
```



Figure