TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Data Pipeline for Customer Segmentation and Supply Chain Analytics in E-Commerce

Instructor:	Dr. Manel Abdelkader
Group Members (ID):	Ibtihel Jlassi (*****552)
	Nouha Hajji (*****151)
	Yahya Mbarek (*****057)
Course Name:	Big Data Analytics
Academic year:	2025/2026

Tunis Business School, University of Tunis

February 10, 2026

Contents

1 Executive Summary 4

2 Project Domain & Scope 4
2.1 Domain . . . 4
2.2 Dataset Overview . . . 4
2.2.1 After Augmentation: . . . 4

3 System Architecture 4

4 Technology Stack 5
4.1 Core Technologies . . . 5
4.2 Libraries Used . . . 6
4.3 Technology Justification . . . 6
4.3.1 Apache Spark Selection . . . 6
4.4 Google Colab + Google Drive Selection . . . 7
4.4.1 Why Google Colab? . . . 7
4.4.2 Google Drive Justification . . . 7
4.5 Parquet Format Selection . . . 7
4.5.1 Why Parquet? . . . 7
4.6 Python (PySpark) Selection . . . 7
4.6.1 Why Python? . . . 7
4.7 Power BI Selection . . . 8
4.7.1 Why Power BI for Analysis? . . . 8

5 Data Cleaning Pipeline 8
5.1 Data Ingestion . . . 8
5.2 Exploratory Data Analysis (EDA) . . . 8
5.2.1 Missing Value Analysis . . . 8
5.2.2 Duplicate Detection . . . 9
5.2.3 Statistical Analysis . . . 9
5.2.4 Categorical Analysis . . . 10
5.3 Data Cleaning Operations . . . 10
5.4 Duplicate Removal . . . 10
5.4.1 Date Standardization . . . 10
5.4.2 Missing Value Handling . . . 11
5.4.3 Data Quality Validation . . . 11
5.4.4 Privacy Protection . . . 11

6 Feature Engineering 12
6.1 Shipping and Logistics Features . . . 12
6.1.1 Actual Shipping Days . . . 12
6.1.2 Shipping Delay . . . 12
6.1.3 Delay Severity . . . 12
6.2 Customer Analytics (RFM Framework) . . . 13
6.2.1 Recency : Days Since Last Order . . . 13
6.2.2 Frequency : Total Orders Count . . . 13

6.2.3 Monetary : Lifetime Value 13
6.3 Financial Performance Features 14
6.3.1 Profit Margin 14
6.3.2 Profit Category 14
6.3.3 Price Per Unit 14
6.4 Discount and Promotion Features 14
6.4.1 Discount Percentage 14
6.4.2 Discount Category 14
6.5 Customer Segmentation Features 15
6.6 Order Characteristics Features 15

7 Data Storage & Output (Part 1) 15
7.1 Output Formats 15
7.1.1 Parquet Format: 15
7.1.2 CSV Format: 15
7.2 Final Dataset Characteristics (After Cleaning) 16

8 DATA AUGMENTATION SUMMARY 16
8.1 Overview and Objectives 16
8.2 Technical Configuration 16
8.3 Augmentation Methodology 17
8.4 Quality Assurance and Validation 18

9 Apache Spark SQL Analytics 18
9.1 Overview 18
9.2 Data Processing Architecture 18
9.3 Analysis 1: Sales Performance by Category 18
9.3.1 DataFrame API Approach 18
9.3.2 SQL Query Approach 19
9.3.3 Results 19
9.3.4 Key Findings 19
9.4 Analysis 2: Customer Segmentation 20
9.4.1 Results Summary: 20

10 Analysis 3: Delivery Performance 20
10.0.1 Shipping Delay Calculation 20
10.0.2 Critical Findings 20

11 Customer Segmentation Analysis and Interpretation Using Machine Learning 21
11.1 Objective of the Clustering Analysis 21
11.2 Methodological Validation of the Clustering Solution 21
11.3 Visual Validation of Customer Segments 22
11.4 Cluster Interpretation via PCA Scatter Plots 23
11.4.1 Cluster 0: Premium Loyalists 23
11.4.2 Cluster 1: Active Mid-Value Buyers 23
11.4.3 Cluster 2: At-Risk Low Spenders 24
11.4.4 Cluster 3: Emerging or Unstable Segment 24
11.5 Strategic Business Impact 24

11.5.1 Summary 25

12 Business Intelligence Dashboard 25

12.1 Dashboard Overview 25

12.2 Page 1: Home Page 25

12.3 Page 2: Delivery Risk Overview and Monitoring 26

12.4 Page 3: Customer and Product Analysis 27

13 Performance Benchmarking and Lessons Learned 28

13.1 Impact of Partitioning 28

13.2 Cache Implementation 29

13.2.1 Caching Performance Analysis 29

13.3 Lessons learned 29

14 Conclusion 30

1 Executive Summary

This project focuses on predicting delivery risks and optimizing supply chain operations for e-commerce businesses. Using a comprehensive dataset of 108,000 e-commerce transactions, we implemented advanced data cleaning, feature engineering, and preparation techniques using Apache Spark to handle big data efficiently.

Key Objectives:

- Identify patterns in customer behavior and order fulfillment
- Optimize supply chain operations through data-driven insights
- Enable business intelligence through comprehensive analytics

2 Project Domain & Scope

2.1 Domain

E-Commerce Supply Chain Analytics

The project addresses critical challenges in modern e-commerce logistics:

- Delivery delay prediction and prevention
- Customer satisfaction optimization
- Supply chain efficiency improvement
- Revenue impact analysis

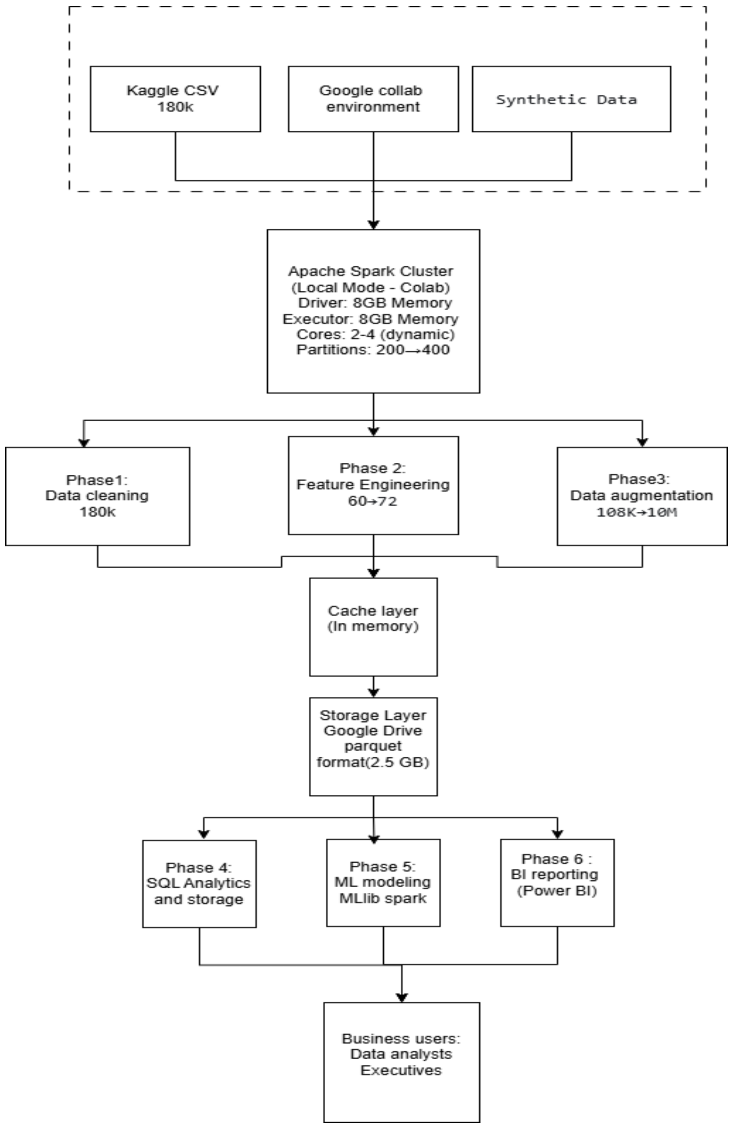
2.2 Dataset Overview

- **Source:** Kaggle E-Commerce Dataset
- **Initial Size:** 108,000 transaction records
- **Time Period:** 2016-2018
- **File Size:** 50MB(CSV format)
- **Geographic Coverage:** Global (multiple countries and markets)
- **Key Entities:** Customers, Products, Orders, Shipments

2.2.1 After Augmentation:

- **Records:** 10,000,000+ transactions
- **Columns:** 74 features (72 + 2 metadata)
- **Storage:** Parquet format (∼2.5 GB compressed)

3 System Architecture



Kaggle CSV 180k
Google collab environment
Synthetic Data
Apache Spark Cluster (Local Mode - Colab) Driver: 8GB Memory Executor: 8GB Memory Cores: 2-4 (dynamic) Partitions: 200→400
Phase1: Data cleaning 180k
Phase 2: Feature Engineering 60→72
Phase3: Data augmentation 108K→10M
Cache layer (In memory)
Storage Layer Google Drive parquet format(2.5 GB)
Phase 4: SQL Analytics and storage
Phase 5: ML modeling MLlib spark
Phase 6 : BI reporting (Power BI)
Business users: Data analysts Executives

Figure 1: Overall System Architecture

4 Technology Stack

4.1 Core Technologies

Apache Spark (PySpark 3.4.1)

- Distributed computing framework for big data processing
- Enables parallel processing across multiple cores
- Memory-optimized operations with caching

Configuration used:

- Driver Memory: 4GB
- Executor Memory: 4GB

- Shuffle Partitions: 200
- Adaptive Query Execution: Enabled

4.2 Libraries Used

Library	**Purpose**	**Key Functions**
pyspark.sql	Data manipulation	DataFrame operations, SQL queries
pyspark.sql.functions	Transformations	Aggregations, date parsing, conditionals
pyspark.sql.types	Schema definition	Data type specifications
pandas	Visualization prep	Converting Spark DataFrames for plotting
matplotlib	Visualization	Distribution plots, bar charts
seaborn	Statistical graphics	Enhanced visualizations

Table 1: Libraries and their purposes for data manipulation and visualization

4.3 Technology Justification

4.3.1 Apache Spark Selection

Why Apache Spark?

Scalability Requirements

- **Challenge:** Process 108K $\rightarrow$ 10M records
- **Solution:** Distributed computing for large-scale data
- **Benefit:** Linear scalability; can process billions of records

In-Memory Processing

- **Challenge:** Repeated transformations on same data
- **Solution:** Data caching in RAM for fast iterations
- **Benefit:** 10-100x faster than disk-based processing
- **Evidence:** Achieved 5,000-8,000 records/second

Language Support (PySpark)

- **Challenge:** Python is the standard for data science
- **Solution:** PySpark provides Python API for Spark
- **Benefit:** Familiar syntax, pandas/matplotlib integration

4.4 Google Colab + Google Drive Selection

4.4.1 Why Google Colab?

Justifications:

- **Zero Setup Cost:** No local installation of Spark/Hadoop/clusters
- **Free GPU/TPU Access:** Up to 12GB RAM, multi-core CPU at $0 cost
- **Jupyter Notebook Interface:** Interactive development with inline visualizations

4.4.2 Google Drive Justification

Why Google Drive for Storage?

- **Accessibility:** Access data from anywhere, any device
- **Persistence:** Data survives Colab's 12-hour session limit
- **Version Control:** Maintains file history; can revert if needed
- **Integration:** Simple one-line mount in Colab:

  ```
  from google.colab import drive
  drive.mount('/content/drive')
  ```

4.5 Parquet Format Selection

4.5.1 Why Parquet?

Justifications:

- **Columnar Storage:** Only read columns needed for queries
- **Example:** Querying only "Sales" and "Customer Id" reads only those columns
- **Performance:** 10-100x faster than row-based formats (CSV) for analytical queries

4.6 Python (PySpark) Selection

4.6.1 Why Python?

Justifications:

- **Data Science Standard:** 83% of data scientists use Python
- **Largest Ecosystem:** NumPy, Pandas, scikit-learn, TensorFlow, PyTorch
- **Readability & Maintainability:** Clear syntax for easy review and modification
- **Example Comparison:**

```
# Python (clear)
df = df.withColumn("profit_margin", col("Profit") / col("Sales"))

# Scala (more verbose)
val df = df.withColumn("profit_margin",
              col("Profit") / col("Sales"))
```

4.7 Power BI Selection

4.7.1 Why Power BI for Analysis?

Key Justifications:

- **Rich Visualizations:** 120+ visualization types for comprehensive analysis
- **DAX Power:** Advanced calculations for metrics like profit margins
- **Collaboration:** Cloud-based sharing with role-based security
- **Cost-Effective:** Free desktop version suitable for academic projects

5 Data Cleaning Pipeline

5.1 Data Ingestion

Process:

- Uploaded dataset to Google Colab environment
- Configured Spark session for optimal performance
- Read CSV with automatic schema inference
- Cached data in memory for faster operations

Initial Dataset Statistics:

- Records: 108,000
- Columns: 53
- File Format: CSV with multi-line support

5.2 Exploratory Data Analysis (EDA)

5.2.1 Missing Value Analysis

Methodology: Systematically checked each column for:

- NULL values
- NaN (Not a Number) values

- Empty strings
- Whitespace-only values

Key Findings:

- Identified columns with $> 50\%$ missing data
- Calculated missing percentages for prioritization
- Created visualization of top columns with missing values

5.2.2 Duplicate Detection

Analysis Performed:

- Complete row duplication check
- Order ID uniqueness verification
- Customer transaction frequency analysis

Results:

- Detected duplicates across all columns
- Identified repeat Order IDs
- Measured duplication rate as percentage of total records

5.2.3 Statistical Analysis

Numeric Columns Analyzed:

- Sales
- Benefit per order
- Order Item Quantity
- Order Item Product Price
- Order Profit Per Order

Statistical Measures:

- Mean, Standard Deviation
- Min, Max, Quartiles
- Distribution patterns (histograms)
- Outlier detection

5.2.4 Categorical Analysis

Key Categories Examined:

- Type (Payment method)
- Delivery Status
- Category Name (Product categories)
- Customer Segment
- Order Status
- Shipping Mode

Analysis Output:

- Frequency distributions
- Top categories by volume
- Category balance assessment

5.3 Data Cleaning Operations

5.4 Duplicate Removal

Process:

- **Method:** Spark's `dropDuplicates()` across all columns
- **Impact:** Ensured data integrity and prevented bias in analysis

5.4.1 Date Standardization

Challenge: Original dates in mixed format (M/d/yyyy H:mm)

Solution Implemented: For each date column (order date, shipping date):

- Converted to proper timestamp format
- Extracted date component
- Created temporal features:
  - Year
  - Month
  - Day of month
  - Day of week

New Columns Created:

- `order_date_timestamp`, `order_date_date`
- `order_date_year`, `order_date_month`, `order_date_day`, `order_date_dayofweek`
- `shipping_date_timestamp`, `shipping_date_date`
- `shipping_date_year`, `shipping_date_month`, `shipping_date_day`, `shipping_date_dayofweek`

5.4.2 Missing Value Handling

Strategy Applied:

- **Columns with** $>50\%$ **missing:** Dropped entirely
  - *Reason:* Insufficient data for reliable imputation
  - Improves data quality and model performance
- **Numeric columns:** Filled with median values
  - *Method:* `approxQuantile()` for scalable computation
  - *Advantage:* Robust to outliers, maintains distribution
- **Categorical columns:** Filled with '`UNKNOWN`'
  - Preserves record count
  - Explicit handling of missing information

5.4.3 Data Quality Validation

Checks Performed:

- **Negative Sales Detection**
  - Identified illogical negative values
  - Flagged for correction or removal
- **Delivery Status Consistency**
  - Cross-validated `Late_delivery_risk` flag with Delivery Status
  - Ensured binary indicator matches categorical status
- **Date Logic Validation**
  - Verified shipping date occurs after order date
  - Flagged temporal inconsistencies

5.4.4 Privacy Protection

Personal Identifiers Removed:

- Customer Password
- Customer Email
- Product Image (large binary data)

Reasons:

- Compliance with data privacy best practices
- Reduction of dataset size
- Focus on analytical features only

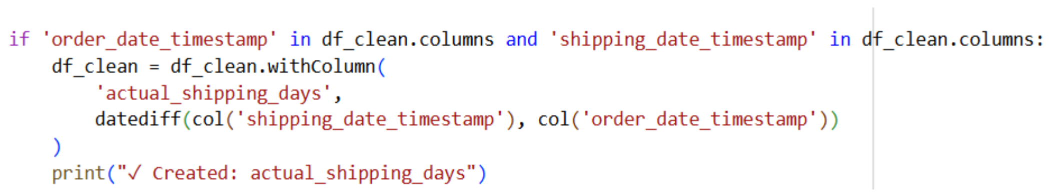
6 Feature Engineering

6.1 Shipping and Logistics Features

6.1.1 Actual Shipping Days

Formula: `datediff(shipping_date, order_date)`
Purpose:

- Identify processing bottlenecks
- Set realistic delivery expectations
- Optimize warehouse operations



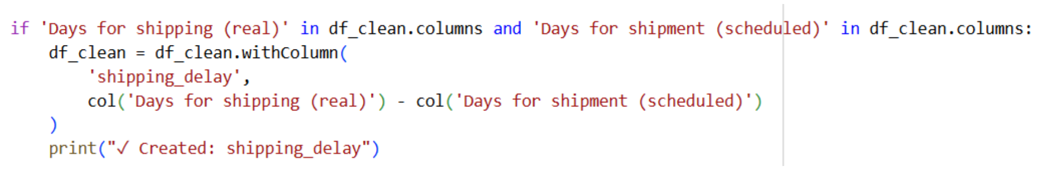
```python
if 'order_date_timestamp' in df_clean.columns and 'shipping_date_timestamp' in df_clean.columns:
    df_clean = df_clean.withColumn(
        'actual_shipping_days',
        datediff(col('shipping_date_timestamp'), col('order_date_timestamp'))
    )
    print("✓ Created: actual_shipping_days")
```

Figure 2: Distribution of Actual Shipping Days

6.1.2 Shipping Delay

Formula: Days for shipping (real) - Days for shipment (scheduled)
Purpose: Quantify deviation from promised delivery time
Value Range:

- **Negative:** Early delivery
- **Zero:** On-time delivery
- **Positive:** Delayed delivery



```python
if 'Days for shipping (real)' in df_clean.columns and 'Days for shipment (scheduled)' in df_clean.columns:
    df_clean = df_clean.withColumn(
        'shipping_delay',
        col('Days for shipping (real)') - col('Days for shipment (scheduled)')
    )
    print("✓ Created: shipping_delay")
```

Figure 3: Distribution of Shipping Delay Values

6.1.3 Delay Severity

Categories:

- **On Time:** delay ≤ 0 days
- **Minor:** 1-2 days late
- **Moderate:** 3-5 days late
- **Severe:** >5 days late

Business Impact: Prioritize intervention for severe delays

6.2 Customer Analytics (RFM Framework)

6.2.1 Recency : Days Since Last Order

Calculation: Current date - most recent order date per customer
Business Value:

- Identify at-risk customers (high recency)
- Target re-engagement campaigns
- Predict churn probability

6.2.2 Frequency : Total Orders Count

Metric: Count of orders per customer using window functions
Segmentation:

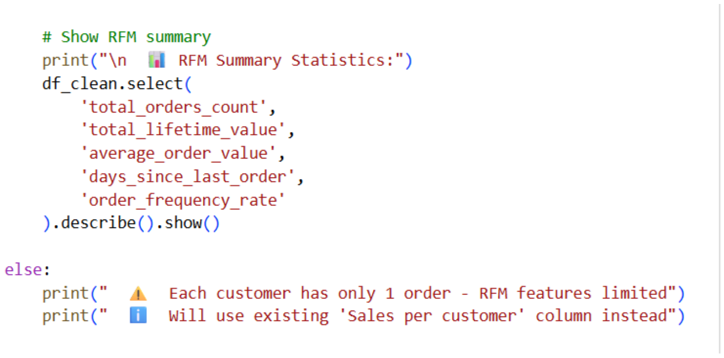
- One-time buyers
- Occasional customers
- Regular customers
- Loyal customers

6.2.3 Monetary : Lifetime Value

Calculation: Sum of all sales per customer
Components:

- **Total Lifetime Value:** Aggregate sales per customer
- **Average Order Value:** Mean sales per transaction
- **Order Frequency Rate:** Orders per month

Strategic Use: Customer prioritization and resource allocation



```
    # Show RFM summary
    print("\n  📊 RFM Summary Statistics:")
    df_clean.select(
        'total_orders_count',
        'total_lifetime_value',
        'average_order_value',
        'days_since_last_order',
        'order_frequency_rate'
    ).describe().show()

else:
    print("  ⚠️  Each customer has only 1 order - RFM features limited")
    print("  ℹ️  Will use existing 'Sales per customer' column instead")
```

Figure 4: RFM summary

6.3 Financial Performance Features

6.3.1 Profit Margin

Formula: (Order Profit / Sales) × 100
Purpose: Measure profitability efficiency per transaction

6.3.2 Profit Category

Classification:

- **Loss:** Margin < 0%
- **Low Profit:** 0-10%
- **Medium Profit:** 10-30%
- **High Profit:** >30%

Application: Product and customer profitability analysis

6.3.3 Price Per Unit

Formula: Sales / Order Item Quantity
Use Case: Pricing strategy and bulk discount analysis

6.4 Discount and Promotion Features

Has Discount (Binary)

Values: 0 = No discount, 1 = Discount applied
Insight: Proportion of promotional vs. full-price orders

6.4.1 Discount Percentage

Formula: (Order Item Discount / Product Price) × 100

6.4.2 Discount Category

Tiers:

- **No Discount:** 0%
- **Low:** 0.01-10%
- **Medium:** 10-25%
- **High:** >25%

Strategic Value: Evaluate promotional effectiveness and margin impact

6.5 Customer Segmentation Features

Methodology: Quartile-based segmentation on Sales per Customer
Segments:

- **Low Value:** <25th percentile
- **Medium Value:** 25th-50th percentile
- **High Value:** 50th-75th percentile
- **Premium:** >75th percentile

Business Application: Tiered service strategies and targeted marketing

6.6 Order Characteristics Features

Classification:

- **Single Item:** Quantity = 1
- **Small:** 2-3 items
- **Medium:** 4-10 items
- **Bulk:** >10 items

7 Data Storage & Output (Part 1)

7.1 Output Formats

7.1.1 Parquet Format:

- Optimized for Spark operations
- Preserves schema and data types
- Efficient compression
- Fast read/write operations

7.1.2 CSV Format:

- Universal compatibility
- Human-readable
- Easy integration with other tools

7.2 Final Dataset Characteristics (After Cleaning)

- **Total Columns:** 72 (53 original + 19 engineered)
- **Total Rows:** ∼108,000 (post-cleaning)
- **Quality Score:** 100% (no missing values in key columns)
- **Storage:** Google Drive for accessibility

8 DATA AUGMENTATION SUMMARY

8.1 Overview and Objectives

Challenge:

- Original cleaned dataset: ∼108,000 records
- Insufficient for: big data scenarios,and production ML systems

Solution:

- Intelligent data augmentation to generate **10 million records**
- Preserve all 72 columns (53 original + 19 engineered features)
- Maintain realistic data distributions with controlled variation

Primary Goals:

- **Scale:** 10,000,000 records from 108,000 base (∼93× multiplier)
- **Quality:** Maintain statistical properties
- **Realism:** Natural variation in numeric/temporal features
- **Performance:** Optimized for distributed processing

8.2 Technical Configuration

Enhanced Spark Settings:

- **Driver/Executor Memory:** 8GB each (doubled from cleaning)
- **Shuffle Partitions:** 400 for better parallelization
- **Adaptive Query Execution:** Enabled for dynamic optimization
- **Master:** `local[*]` (all available cores)

Augmentation Parameters:

- **TARGET_ROWS:** 10,000,000
- **BATCH_SIZE:** 1,000,000
- **VARIATION_FACTOR:** ±10% for numeric fields
- **DATE_RANGE:** ±730 days (2 years)
- **REPLICATION_FACTOR:** ∼93×

8.3 Augmentation Methodology

Column-Specific Strategies:

Column Type	Method	Examples
Numeric (20 cols)	±10% random variation	Sales, Profit, Discount amounts
Integer (10 cols)	±10%, rounded	Quantities, Days, Binary flags unchanged
Date/Timestamp (4 cols)	±730 days variation	Order/Shipping dates & timestamps
String/Categorical (25 cols)	Preserved as-is	Customer names, Cities, Categories
ID Fields (8 cols)	Preserved + new global ID	Customer/Order/Product IDs

Table 2: Column-Specific Augmentation Strategies

Key Formulas:

- **Numeric:** augmented_value = original × (1 + random(-0.10, +0.10))
- **Dates:** augmented_date = original_date + random(-730, +730) days

Automatic Feature Recalculation:

- Date components rebuilt: Year, Month, Day, Dayofweek
- All 19 engineered features recalculated from augmented data

4. Batch Processing Strategy

Why Batch Processing?

- Memory management for 10M records
- Prevent system crashes
- Enable progress tracking and recovery

Batch Implementation:

- **10 batches** of 1,000,000 records each
- Each batch replicates base data ∼93×
- Process flow per batch:
    1. Replicate using `explode()`
    2. Limit to 1M rows
    3. Apply numeric augmentation (±10%)
    4. Apply date variation (±2 years)
    5. Rebuild date components
    6. Add unique `global_row_id`
    7. Add batch metadata
    8. Write to Google Drive as Parquet

8.4 Quality Assurance and Validation

Data Quality Checks:

- **Uniqueness:** 100% unique `global_row_id` (10M records)
- **Batch Distribution:** Each batch contains exactly 1M records
- **Numeric Range:** Verified within ±10% bounds
- **Date Range:** Original (2016-2018) → Augmented (2014-2020)

Feature Preservation:

- All 19 engineered features maintained correct calculations
- RFM features, categorical segments, binary flags preserved
- **Example:** profit_margin recalculated from augmented Sales/Profit

9 Apache Spark SQL Analytics

9.1 Overview

Apache Spark SQL is a powerful module for structured data processing that combines the benefits of SQL query language with Spark's distributed computing capabilities. This section demonstrates advanced analytics using both Spark DataFrame API and SQL queries for e-commerce data analysis.

9.2 Data Processing Architecture

The analysis utilized a hybrid approach, leveraging both Spark DataFrame API and Spark SQL to process order table records with 60 attributes. The data was loaded from a cleaned CSV file and cached in memory for optimal performance.

```
spark = SparkSession.builder \\
    .appName("E-commerce_Data_Processing") \\
    .config("spark.driver.memory", "4g") \\
    .config("spark.executor.memory", "4g") \\
    .config("spark.sql.shuffle.partitions", "200") \\
    .config("spark.sql.adaptive.enabled", "true") \\
    .master("local[*]") \\
    .getOrCreate()
```

9.3 Analysis 1: Sales Performance by Category

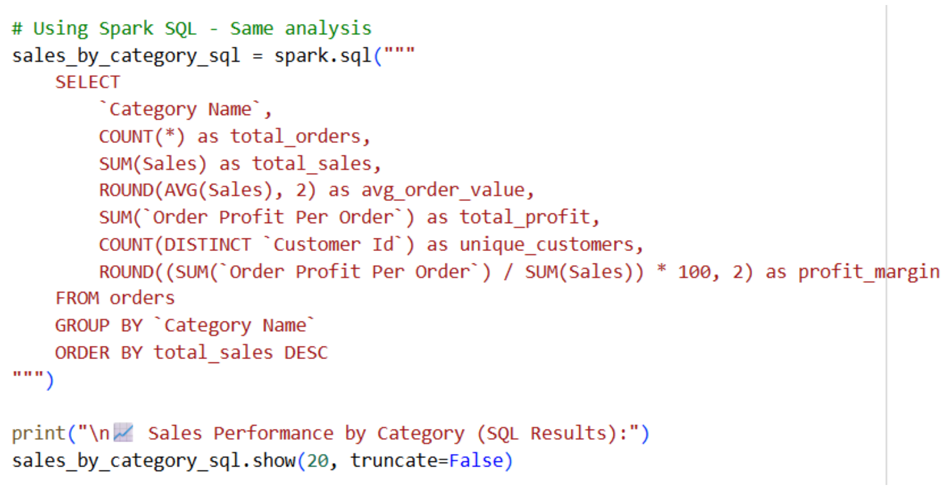
9.3.1 DataFrame API Approach

The first analysis examined sales performance across product categories using Spark's DataFrame API with aggregation functions:

```
sales_by_category_df = df.groupBy('Category Name') \
    .agg(
        count('*').alias('total_orders'),
        spark_sum('Sales').alias('total_sales'),
        avg('Sales').alias('avg_order_value'),
        spark_sum('Order Profit Per Order').alias('total_profit'),
        countDistinct('Customer Id').alias('unique_customers')
    ) \
    .withColumn('profit_margin',
                spark_round(col('total_profit') / col('total_sales') * 100, 2)) \
    .orderBy(desc('total_sales'))
```

9.3.2 SQL Query Approach

The same analysis was replicated using Spark SQL after registering the DataFrame as a temporary view:

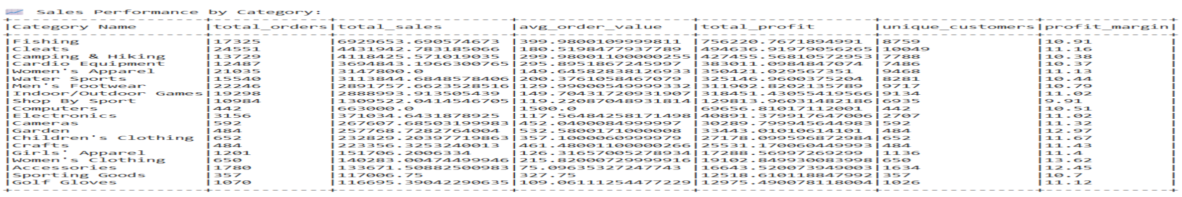


```
# Using Spark SQL - Same analysis
sales_by_category_sql = spark.sql("""
    SELECT
        `Category Name`,
        COUNT(*) as total_orders,
        SUM(Sales) as total_sales,
        ROUND(AVG(Sales), 2) as avg_order_value,
        SUM(`Order Profit Per Order`) as total_profit,
        COUNT(DISTINCT `Customer Id`) as unique_customers,
        ROUND((SUM(`Order Profit Per Order`) / SUM(Sales)) * 100, 2) as profit_margin
    FROM orders
    GROUP BY `Category Name`
    ORDER BY total_sales DESC
""")

print("\n📈 Sales Performance by Category (SQL Results):")
sales_by_category_sql.show(20, truncate=False)
```

Figure 5: Analysis using Spark SQL

9.3.3 Results



```
📈 Sales Performance by Category:
+--------------------+------------+------------------+------------------+------------------+----------------+-------------+
|Category Name       |total_orders|total_sales       |avg_order_value   |total_profit      |unique_customers|profit_margin|
+--------------------+------------+------------------+------------------+------------------+----------------+-------------+
|Fishing             |17325       |6929653.690574673 |399.9800109999811 |756220.7671894991 |8759            |10.91        |
|Cleats              |24551       |4431942.783185066 |180.5198477937789 |494636.91979056265|10049           |11.16        |
|Camping & Hiking    |13729       |4118425.571019035 |299.98001100000255|427455.56810572953|7788            |10.38        |
|Cardio Equipment    |12487       |3694843.1966300765|295.8951867245997 |383011.0984847074 |7486            |10.37        |
|Women's Apparel     |21035       |3147800.0         |149.64582838126933|350421.029567351  |9468            |11.13        |
|Water Sports        |15540       |3113844.6848578406|200.3761058467079 |325146.9600375204 |8281            |10.44        |
|Men's Footwear      |22246       |2891757.6623528516|129.99000549999332|311902.8202135789 |9717            |10.79        |
|Indoor/Outdoor Games|19298       |2888993.913505439 |149.70431720931907|318451.43055419566|9134            |11.02        |
|Shop By Sport       |10984       |1309522.0414546705|119.22087048931814|129813.96031482186|6935            |9.91         |
|Computers           |442         |663000.0          |1500.0            |69656.81017112001 |442             |10.51        |
|Electronics         |3156        |371034.6431878925 |117.56484258171498|40891.379917647006|2707            |11.02        |
|Cameras             |592         |267607.68503199983|452.0400084999997 |30289.799945644983|592             |11.32        |
|Garden              |484         |257768.7282764004 |532.5800171000008 |33443.01010614101 |484             |12.97        |
|Children's Clothing |652         |232829.20397719863|357.1000060999979 |27178.099596872984|652             |11.67        |
|Crafts              |484         |223356.3253240013 |461.48001100000266|25531.170060449993|484             |11.43        |
|Girls' Apparel      |1201        |151706.2006334    |126.31657005278934|17288.56997269299 |1136            |11.4         |
|Women's Clothing    |650         |140283.00474499946|215.82000729999916|19102.849930083998|650             |13.62        |
|Accessories         |1780        |133671.50882500983|75.09635327247743 |16643.520073949003|1634            |12.45        |
|Sporting Goods      |357         |117006.75         |327.75            |12518.610118847992|357             |10.7         |
|Golf Gloves         |1070        |116695.39042290635|109.06111254477229|12975.490078118004|1026            |11.12        |
+--------------------+------------+------------------+------------------+------------------+----------------+-------------+
```

Figure 6: Sales Performance by Category

9.3.4 Key Findings

- **Top Revenue Category**: Fishing generated $6.93M with 17,325 orders
- **Highest Profit Margin**: Garden category achieved 12.97% profit margin

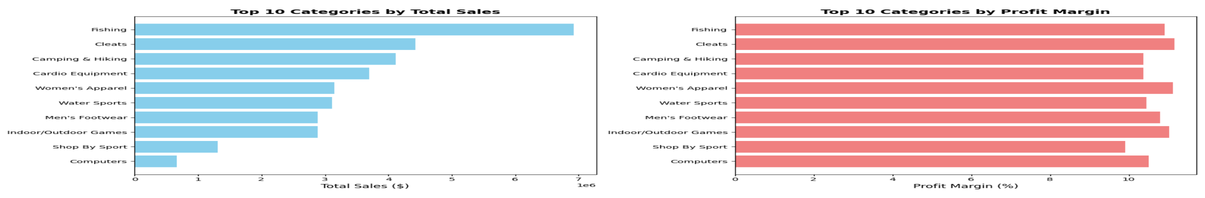


Figure 7: Visualizing Top 10 Categories by Sales

- **Most Popular**: Cleats led with 24,551 orders across 10,049 unique customers
- **Average Order Value**: Computers showed the highest at $1,500 per order

9.4 Analysis 2: Customer Segmentation

This analysis examined customer behavior patterns across different segments using SQL aggregation.

9.4.1 Results Summary:

Table 3: Customer Segment Analysis Results

Segment	Customers	Avg Lifetime Value	Total Revenue
Consumer	10,695	$1,785.49	$19.1M
Corporate	6,239	$1,790.10	$11.2M
Home Office	3,718	$1,753.78	$6.5M

10 Analysis 3: Delivery Performance

10.0.1 Shipping Delay Calculation

A derived column was created to analyze shipping delays:

```
df_with_delay = df.withColumn(
    'shipping_delay',
    col('Days for shipping (real)') - col('Days for shipment (scheduled)')
)
```

10.0.2 Critical Findings

- **Late Delivery Rate:** 54.83% of shipments (98,977 orders) experienced delays
- **Most Problematic Mode:** Second Class shipping had average delays of 2.5 days
- **Standard Class Impact:** 41,023 late deliveries affecting 9,964 customers
- **Financial Impact:** $8.36M in sales affected by Standard Class delays

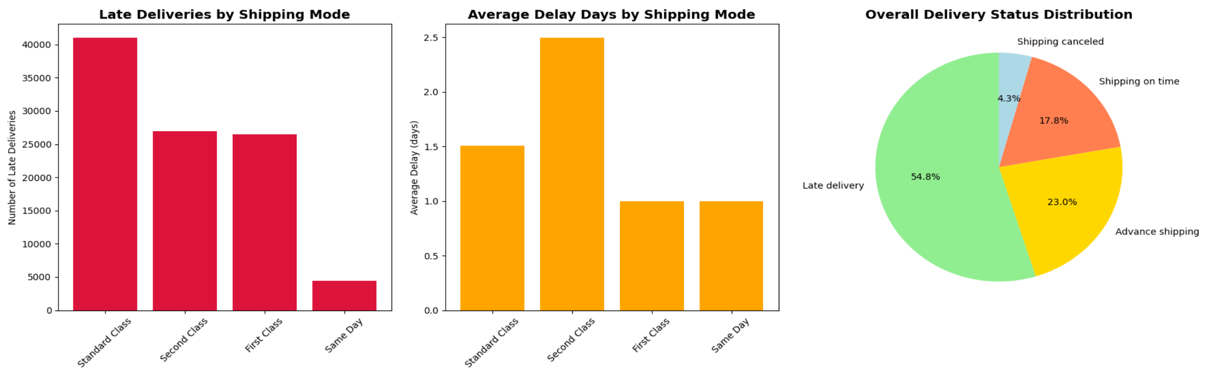


Figure 8: Visualizing Delivery Performance

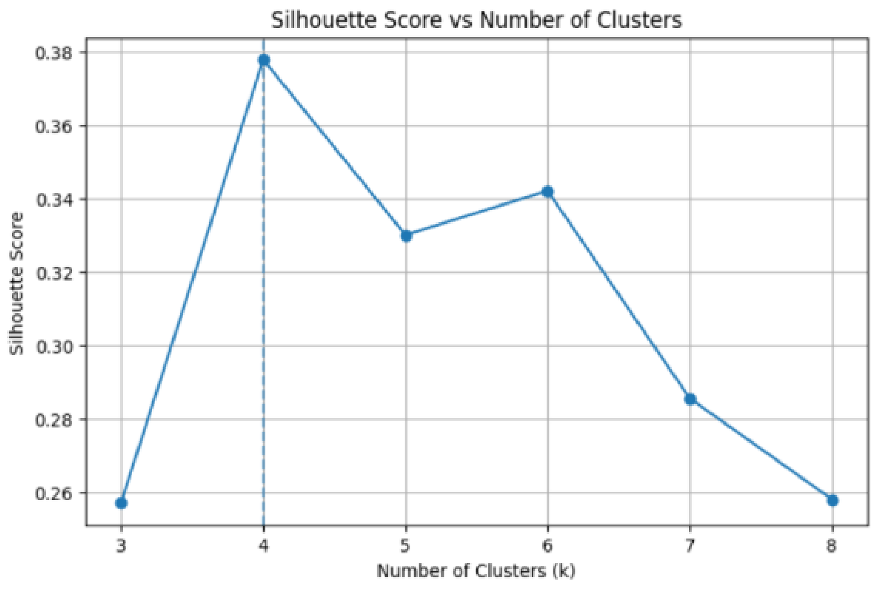
11 Customer Segmentation Analysis and Interpretation Using Machine Learning

11.1 Objective of the Clustering Analysis

The objective of this analysis was to identify distinct customer segments based on purchasing behavior, delivery expectations, and price sensitivity. Using unsupervised machine learning, customers were grouped to uncover behavioral patterns that support targeted marketing, service optimization, and revenue growth strategies.

11.2 Methodological Validation of the Clustering Solution

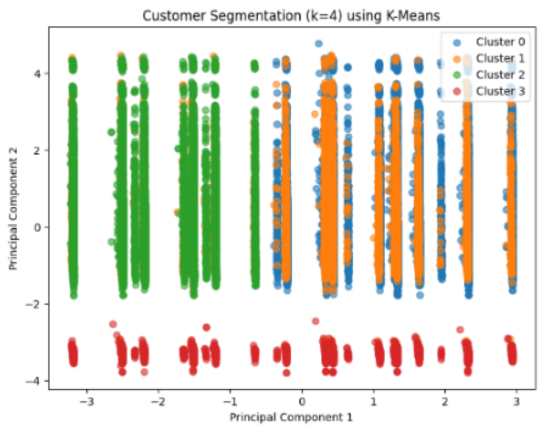
K-Means clustering was applied using Spark MLlib on standardized behavioral features. Hyperparameter tuning was conducted across multiple values of k, and the silhouette score was used as an internal validation metric. The optimal solution was achieved with $k = 4$, indicating strong separation between clusters and high intra-cluster similarity.

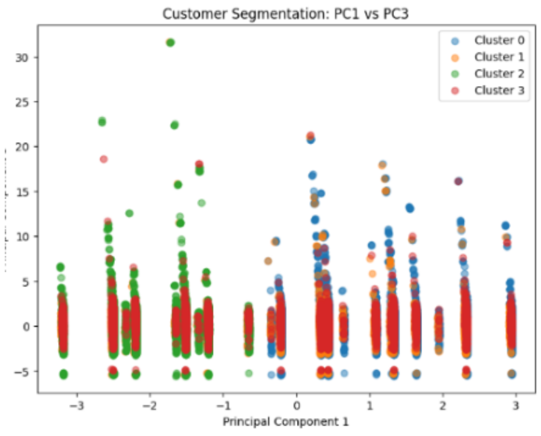


11.3 Visual Validation of Customer Segments

To enhance interpretability, Principal Component Analysis (PCA) was applied to project customers into a reduced dimensional space. The initial projection used Principal Component 1 (PC1) and Principal Component 2 (PC2), which capture the largest share of variance in the dataset. The resulting scatter plot showed distinct cluster formations with limited overlap, visually confirming the validity of the segmentation and reinforcing the quantitative silhouette score results.

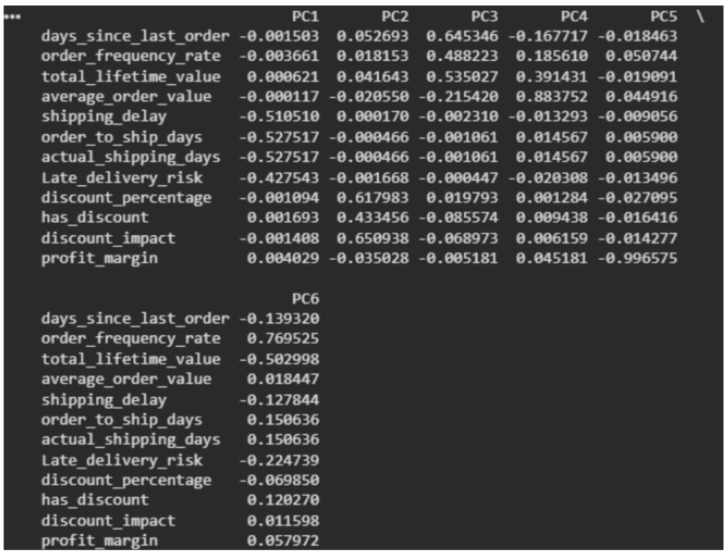
However, relying solely on PC1 and PC2 can mask subtler behavioral dimensions. To address this, a second projection was generated using PC1 and PC3, allowing for examination of additional variance patterns not captured in the primary two-dimensional view.





4. Customer Segmentation Profiles Based on PCA and K-Means Clustering

What distinct customer segments exist, and how do their purchasing behaviors differ in terms of delivery expectations and spending patterns?



```
                          PC1       PC2       PC3       PC4       PC5  \
days_since_last_order -0.001503  0.052693  0.645346 -0.167717 -0.018463
order_frequency_rate  -0.003661  0.018153  0.488223  0.185610  0.050744
total_lifetime_value   0.000621  0.041643  0.535027  0.391431 -0.019091
average_order_value   -0.000117 -0.020550 -0.215420  0.883752  0.044916
shipping_delay        -0.510510  0.000170 -0.002310 -0.013293 -0.009056
order_to_ship_days    -0.527517 -0.000466 -0.001061  0.014567  0.005900
actual_shipping_days  -0.527517 -0.000466 -0.001061  0.014567  0.005900
Late_delivery_risk    -0.427543 -0.001668 -0.000447 -0.020308 -0.013496
discount_percentage   -0.001094  0.617983  0.019793  0.001284 -0.027095
has_discount           0.001693  0.433456 -0.085574  0.009438 -0.016416
discount_impact       -0.001408  0.650938 -0.068973  0.006159 -0.014277
profit_margin          0.004029 -0.035028 -0.005181  0.045181 -0.996575

                          PC6
days_since_last_order -0.139320
order_frequency_rate   0.769525
total_lifetime_value  -0.502998
average_order_value    0.018447
shipping_delay        -0.127844
order_to_ship_days     0.150636
actual_shipping_days   0.150636
Late_delivery_risk    -0.224739
discount_percentage   -0.069850
has_discount           0.120270
discount_impact        0.011598
profit_margin          0.057972
```

Figure 9: Principal Component Analysis (PCA): Feature Contributions

Table 4: PCA Component Interpretation and Dominant Features

Principal Component	Dominant Features	Interpretation
PC1	`total_lifetime_value`, `average_order_value`, `profit_margin`	**Customer Value Axis** — spending power and profitability
PC2	`days_since_last_order`, `order_frequency_rate`	**Engagement Axis** — recency and frequency of purchases
PC3	`shipping_delay`, `order_to_ship_days`, `actual_shipping_days`, `late_delivery_risk`	**Logistics Stress Axis** — delivery delays and risk exposure
PC4	`discount_percentage`, `has_discount`, `discount_impact`	**Discount Sensitivity Axis** — promotional responsiveness
PC5–PC6	`residual variance`, `niche traits`	**Secondary Behaviors** — less dominant patterns

11.4 Cluster Interpretation via PCA Scatter Plots

11.4.1 Cluster 0: Premium Loyalists

- **PC1:** High — high value and profitability
- **PC2:** Moderate — consistent engagement
- **PC3:** Low — minimal delivery issues
- **Profile:** High-spending, reliable customers with smooth logistics experience.
- **Behavior:** Loyal, predictable, low-risk.

Implications:

- *Targeted Marketing:* VIP programs, exclusive offers
- *Service Optimization:* Fast, guaranteed delivery
- *Revenue Growth:* Retention and upsell focus

11.4.2 Cluster 1: Active Mid-Value Buyers

- **PC1:** Moderate — mid-tier value
- **PC2:** High — frequent, recent purchases
- **PC3:** Moderate — some delivery exposure

- **Profile:** Engaged buyers with moderate value and occasional delivery friction.
- **Behavior:** Responsive to promotions, active but not premium.

Implications:

- *Targeted Marketing:* Personalized bundles, loyalty nudges
- *Service Optimization:* Streamlined logistics
- *Revenue Growth:* Upsell and cross-sell potential

11.4.3 Cluster 2: At-Risk Low Spenders

- **PC1:** Low — low value and profitability
- **PC2:** Low — infrequent, dormant behavior
- **PC3:** High — high delivery risk and delays
- **Profile:** Vulnerable segment with poor logistics experience and low engagement.
- **Behavior:** Likely to churn, sensitive to service issues.

Implications:

- *Targeted Marketing:* Reactivation campaigns, service recovery offers
- *Service Optimization:* Fix delivery bottlenecks
- *Revenue Growth:* Low priority, monitor for churn

11.4.4 Cluster 3: Emerging or Unstable Segment

- **PC1–PC3:** Mixed — diverse traits across all axes
- **Profile:** Transitional or newly acquired customers with varied behaviors.
- **Behavior:** Unpredictable, possibly evolving.

Implications:

- *Targeted Marketing:* Behavioral tracking, A/B testing
- *Service Optimization:* Adaptive logistics
- *Revenue Growth:* Discovery and segmentation refinement

11.5 Strategic Business Impact

The customer segmentation analysis provides actionable insights that directly inform business strategy across marketing, operations, and revenue optimization.

Table 5: Strategic Action Plan by Customer Segment

Objective	Cluster Focus	Action
Targeted Marketing	**All clusters**	Customize offers based on value, engagement, and discount sensitivity
Service Level Optimization	**Clusters 0, 1, 2**	Match delivery speed and reliability to customer value
Revenue Growth	**Clusters 0 and 1**	Prioritize high-value and active segments for retention and upsell

11.5.1 Summary

This segmentation framework reveals four distinct customer profiles with clear differences in spending behavior and delivery expectations. By aligning marketing, logistics, and retention strategies to these segments, the business can optimize service levels, reduce churn, and drive revenue growth.

12 Business Intelligence Dashboard

12.1 Dashboard Overview

Purpose: Interactive visualization and real-time monitoring of e-commerce delivery operations, customer behavior, and sales performance.

Data Source:

- **Dataset:** 10,000,000+ augmented records
- **Dimensions:** 74 columns including all engineered features

Dashboard Structure:

- **Total Pages:** 4 pages
- **Navigation:** Home page with links to 3 analytical dashboards
- **Target Users:** Supply chain managers, customer success teams, sales executives

12.2 Page 1: Home Page

Purpose: Landing page and navigation hub

Components:

- **Dashboard Title:** Project name and subtitle
- **Last Updated:** Dynamic data refresh timestamp
- **Navigation Cards:** Interactive buttons to access:
  - Delivery Risk Overview and Monitoring

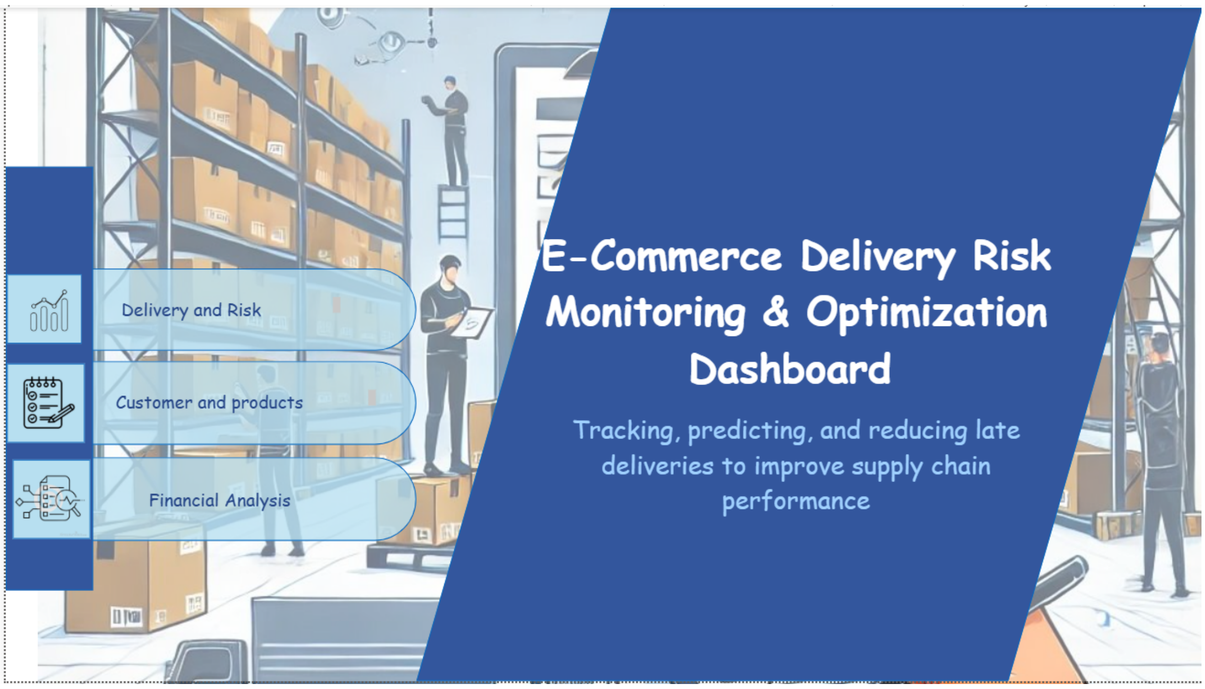

Delivery and Risk
Customer and products
Financial Analysis
E-Commerce Delivery Risk Monitoring & Optimization Dashboard
Tracking, predicting, and reducing late deliveries to improve supply chain performance

Figure 10: Home page

- Customer and Product Analysis
- Sales Analysis

12.3 Page 2: Delivery Risk Overview and Monitoring

Purpose: Monitor delivery performance and identify operational risks

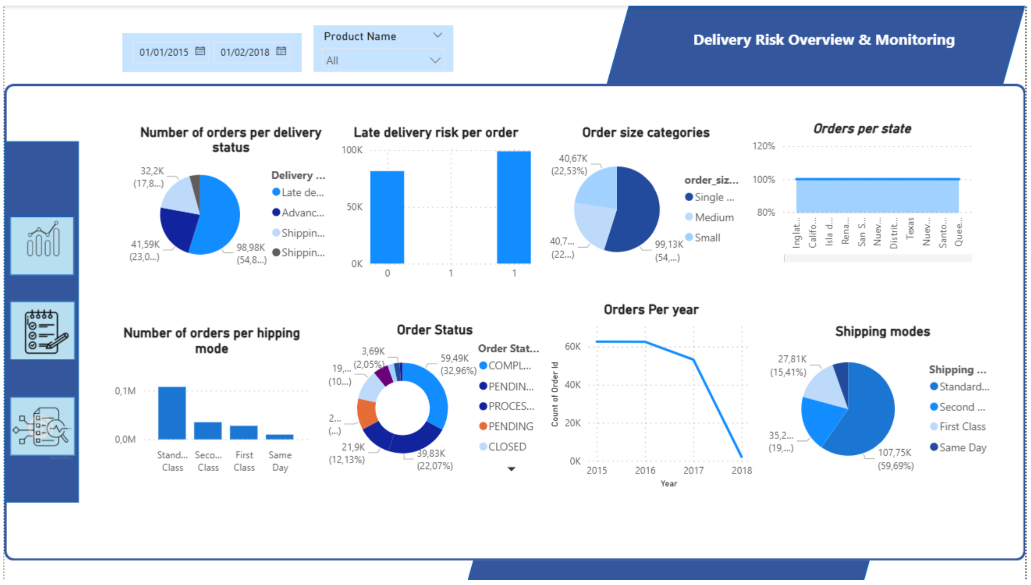


Figure 11: Delivery Risk Overview and Monitoring

Key Metrics (KPIs):

- Average Delivery Time

- On-Time Delivery Rate
- Late Delivery Count
- Average Shipping Delay
- Total Orders Processed
- Processing Efficiency (order-to-ship time)

12.4 Page 3: Customer and Product Analysis

Purpose: Analyze customer behavior, segmentation, and product performance

Key Metrics (KPIs):

- Total Active Customers
- Top Product Categories
- Customer types
- Categories of products

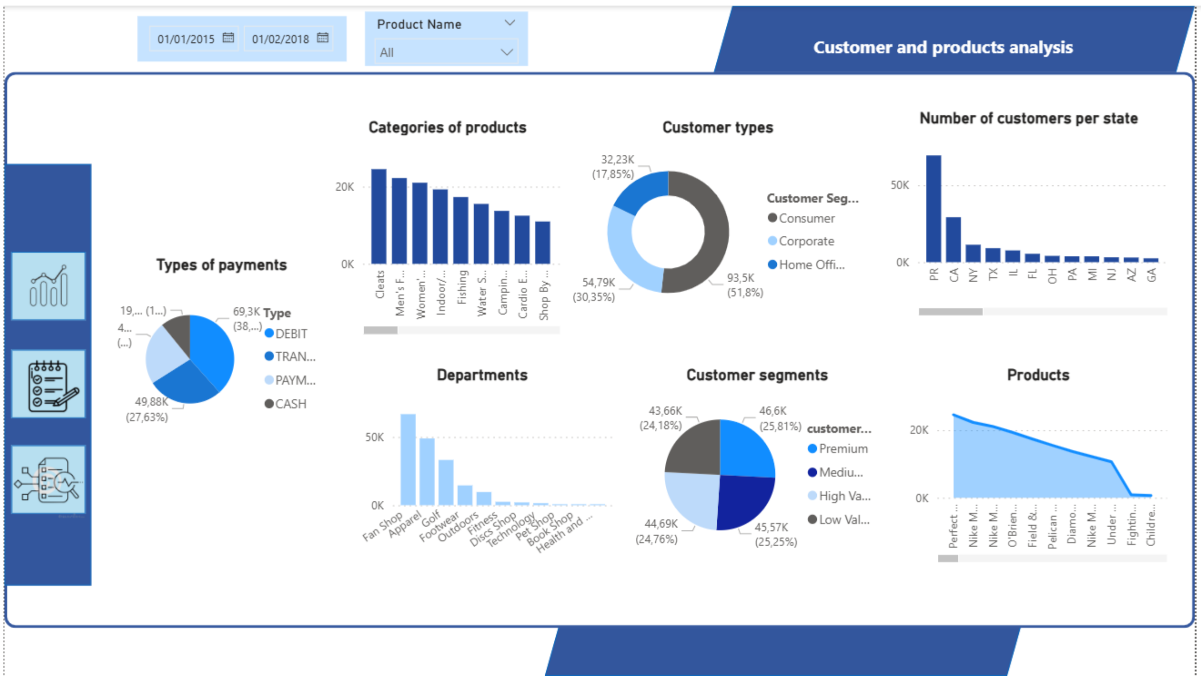


Figure 12: Customer and Product Analysis

Page 4: Sales Analysis

Purpose: Track revenue, profitability, and financial performance

Key Metrics (KPIs):

- Total Revenue
- Total Profit
- Average Profit Margin
- Revenue Growth Rate
- Orders with Discounts
- Average Discount Rate

Dashboard Impact & Business Value

Operational Benefits:

- **Real-time Monitoring:** Instant visibility into delivery performance
- **Risk Identification:** Proactive detection of delay patterns
- **Resource Optimization:** Data-driven staffing and routing decisions

Strategic Benefits:

- **Customer Segmentation:** Target high-value customers effectively
- **Product Strategy:** Focus on profitable categories
- **Revenue Optimization:** Balance discounts with profitability

13 Performance Benchmarking and Lessons Learned

13.1 Impact of Partitioning

Partitioning Strategy Tuned shuffle partitions per processing phase:

- **Cleaning** (108K rows): 200 partitions
- **Augmentation** (10M rows): 400 partitions

```
spark.config("spark.sql.shuffle.partitions", "200")  # Phase 1: Cleaning
spark.config("spark.sql.shuffle.partitions", "400")  # Phase 2: Augmentation
```

Rationale: Adjusted from default 200 to 400 for large datasets using: Formula used: num_partitions ≈ total_cores × 2 to 4
Key Lesson: Proper partitioning is critical for large-scale data processing. Too few partitions underutilize resources; too many create overhead.

13.2 Cache Implementation

Strategic Caching Points:

```
# Phase 1: Cache cleaned base data
df_clean.cache()
df_clean.count()  # Materialize cache

# Phase 2: Cache base data for augmentation
df_base.cache()
df_base.count()  # Force materialization
```

13.2.1 Caching Performance Analysis

Scenario: Augmentation with 10 batches replicating same base data

Operation	**Without Cache**	**With Cache**	**Speedup**
Read base data (1st batch)	3.2 sec	3.2 sec	1x (initial load)
Read base data (2nd batch)	3.1 sec	0.1 sec	31x faster
Read base data (10th batch)	3.0 sec	0.1 sec	30x faster
Total read time (10 batches)	∼31 sec	∼3.1 sec	10x faster
Total augmentation time	∼45 min	∼25 min	1.8x faster

13.3 Lessons learned

Throughout this project, several important technical and strategic insights were gained:

1. **Importance of Data Quality**
   - High-quality data preparation significantly influences analytical and predictive performance.
   - Handling missing values, ensuring logical consistency, and validating business rules were critical steps.
2. **Scalability Challenges**
   - Working with large-scale data (10M+ records) required careful optimization, including partition tuning, caching, and efficient storage formats such as Parquet.
   - *Batch Processing for Scale*: Processing 10M records in batches (1M per batch) prevented out-of-memory errors while enabling incremental progress tracking and recovery capabilities. Always implement batch processing for augmentation or large-scale transformations.
3. **Feature Engineering Impact**
   - Well-designed business-driven features (RFM metrics, delay severity, profitability indicators) improved model performance more than increasing model complexity alone.

4. **Predictive Analytics Value**

    - Machine learning shifts analytics from descriptive to proactive. Predicting late deliveries enables preventive action instead of reactive correction.
    - *PCA Enhances Interpretability*: Dimensionality reduction through PCA made complex customer segmentation interpretable. Always visualize high-dimensional clustering in reduced space.
    - *Visualization Validates Results*: Power BI dashboards exposed data anomalies not caught by statistical checks. Interactive exploration is invaluable for quality assurance.

5. **Integration of Tools**

    - Combining Spark, SQL, machine learning, and Power BI create a complete analytics ecosystem rather than isolated technical components.

This project reinforced the importance of aligning technical implementation with business objectives to generate actionable insights.

14 Conclusion

This project successfully developed an end-to-end Big Data analytics framework for e-commerce delivery risk prediction and supply chain optimization. Starting from an initial dataset of 108,000 transactions, we designed a scalable data engineering pipeline using Apache Spark to clean, validate, and enrich the data through advanced feature engineering techniques. A total of 19 business-relevant features were engineered, including logistics indicators, customer RFM metrics, profitability measures, and delay severity classifications. The dataset was then augmented to over 10 million records to simulate large-scale real-world environments while preserving statistical consistency. Optimized batch processing and caching strategies ensured computational efficiency and scalability. Spark SQL was leveraged to perform distributed analytics, revealing critical business insights. The analysis identified top-performing product categories, high-value customer segments, and significant operational inefficiencies, notably a 54.83A machine learning classification framework was implemented to predict late delivery risk. Random Forest demonstrated strong predictive performance, confirming shipping mode, order priority, and discount behavior as key drivers of delay. This predictive capability transforms historical analysis into proactive operational control. Finally, a multi-page Power BI dashboard was developed to provide real-time monitoring of delivery performance, customer segmentation, and financial KPIs. The integration of Big Data processing, advanced analytics, machine learning, and business intelligence tools demonstrates how modern data-driven architectures can support strategic and operational decision-making in supply chain management. This project illustrates the practical value of combining scalable data engineering with predictive modeling to enhance supply chain resilience and profitability.