

**SDA – 2**

**Streaming Data Analytics - 2**

**Group Project Report**

**Topic: E-commerce**

**Submitted to:**

**Prof. Aditya Dua**

**Submitted by:**

|  |  |
| --- | --- |
| **Parth Singh Bargoti** | **064033** |
| **Aastha Sharma** | **321003** |
| **Arushi Khanna** | **321012** |

**Real-Time Streaming Analytics Solution for E-Commerce**

**Project Objective**

To develop an end-to-end real-time streaming analytics solution for an e-commerce business, leveraging real-time data to derive actionable insights and improve business outcomes. The solution focuses on three key alerts: Cart Abandonment, Price Drop, and Stock Status, integrating Kafka for data streaming and MongoDB for data storage.

**The dataset used for the project is given below for reference:**

****

**Architecture Overview**

1. **Data Ingestion**:
   * **Source**: E-commerce transactional data (orders, carts, inventory updates, etc.).
   * **Tool**: Kafka for real-time data streaming.
2. **Data Processing**:
   * **Framework**: Python-based Kafka consumer.
   * **Processing**: Alerts are generated based on predefined business rules for cart inactivity, price changes, and inventory status.
3. **Data Storage**:
   * **Database**: MongoDB for storing historical data and insights.
4. **Visualization**:
   * **Dashboard**: Grafana to visualize streaming analytics and alerts.

**Business Use Case and Impact**

* **Cart Abandonment Alert**: Identifies potential lost revenue and triggers timely customer engagement to recover abandoned carts.
* **Price Drop Alert**: Enhances customer experience by notifying them of favorable price changes, leading to increased sales.
* **Stock Status Alert**: Ensures inventory optimization, preventing lost sales due to stockouts and overstock scenarios.

**Alert Implementations**

**1. Cart Abandonment Alert**

* **Objective**: Identify when a shopping cart remains inactive for over 10 minutes.
* **Process**:
  1. Maintain an in-memory dictionary (cart\_last\_updated) to track the last update time of each cart.
  2. For each incoming cart update:
     + Check if the cart exists in cart\_last\_updated.
     + If the current time exceeds the last update time by 10 minutes, trigger an abandonment alert.
     + Update the timestamp to reflect the current activity.

**Python Code:**

from kafka import KafkaConsumer

from datetime import datetime, timedelta

import json

cart\_last\_updated = {}

TIME\_THRESHOLD = timedelta(minutes=10)

consumer = KafkaConsumer('cart\_updates', bootstrap\_servers='localhost:9092', value\_deserializer=lambda m: json.loads(m.decode('utf-8')))

for message in consumer:

data = message.value

cart\_id = data['cart\_id']

current\_time = datetime.now()

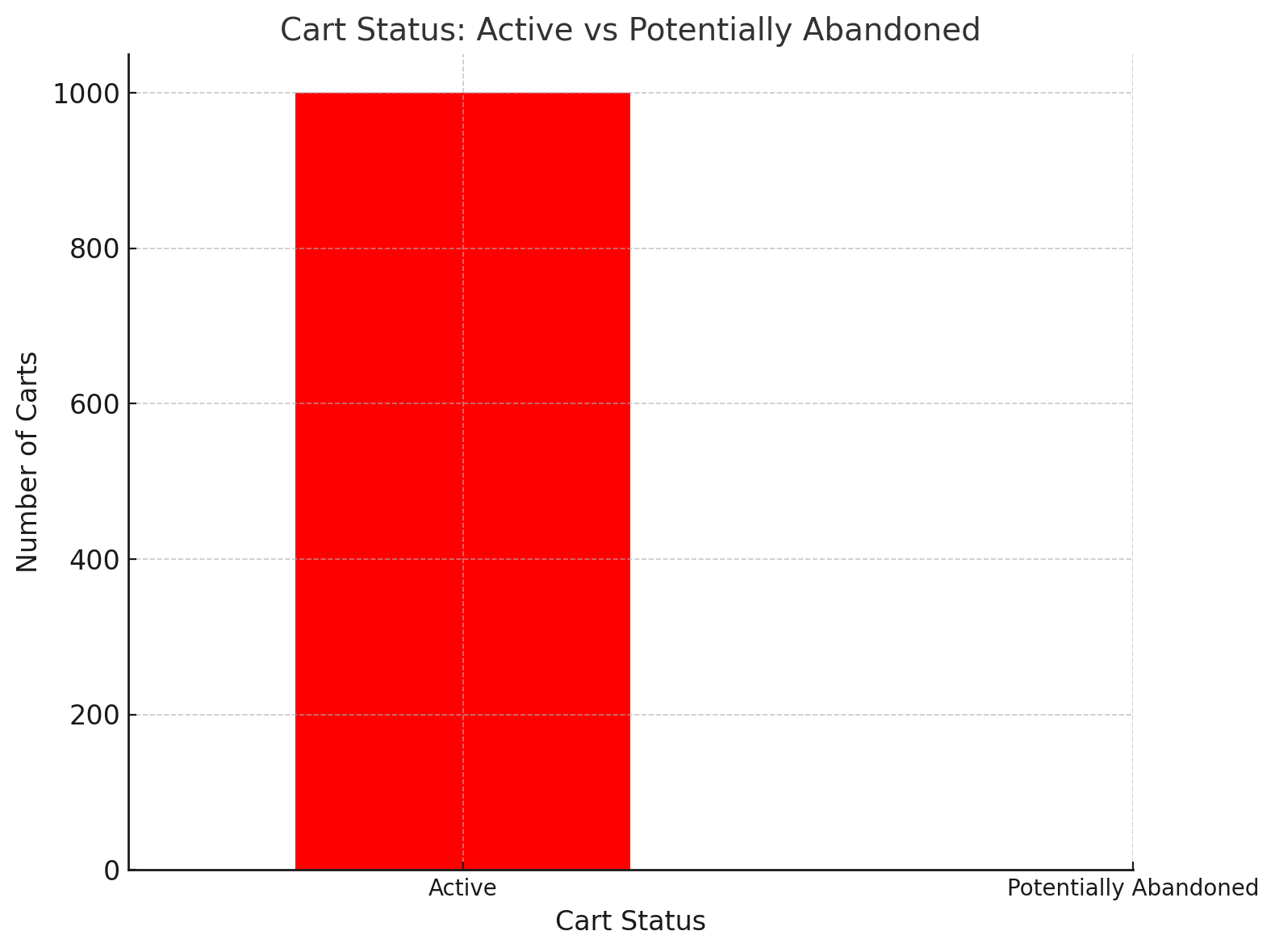
if cart\_id in cart\_last\_updated:

last\_updated = cart\_last\_updated[cart\_id]

if current\_time - last\_updated > TIME\_THRESHOLD:

print(f"Alert: Cart {cart\_id} has been abandoned.")

cart\_last\_updated[cart\_id] = current\_time



**2. Price Drop Alert**

* **Objective**: Detect and notify when the price of a product in a cart decreases in subsequent updates.
* **Process**:
  1. Maintain a dictionary (cart\_prices) to store the latest price of each product in the cart.
  2. For each incoming record:
     + Compare the current price with the stored price.
     + If the current price is lower, trigger a price-drop alert.
     + Update the stored price to the latest value.

**Python Code:**

cart\_prices = {}

consumer = KafkaConsumer('price\_updates', bootstrap\_servers='localhost:9092', value\_deserializer=lambda m: json.loads(m.decode('utf-8')))

for message in consumer:

data = message.value

product\_id = data['product\_id']

new\_price = data['price']

if product\_id in cart\_prices:

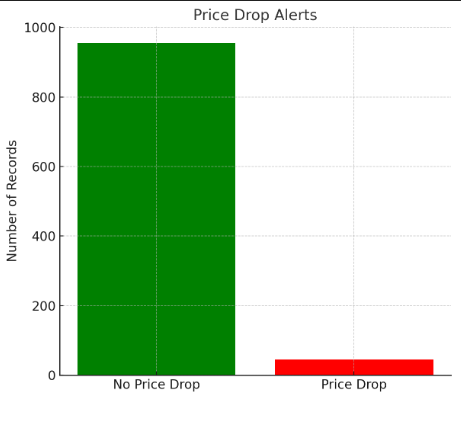
old\_price = cart\_prices[product\_id]

if new\_price < old\_price:

price\_diff = old\_price - new\_price

print(f"Alert: Price drop for product {product\_id}. Price decreased by {price\_diff}.")

cart\_prices[product\_id] = new\_price



**3. Stock Status Alert**

* **Objective**: Monitor inventory levels and issue alerts for low stock or out-of-stock products.
* **Process**:
  1. Check the product\_stock\_status field in each record.
  2. Trigger alerts based on the status:
     + "Low Stock": Low inventory alert.
     + "Out of Stock": Urgent stock-out alert.

**Python Code:**

consumer = KafkaConsumer('stock\_updates', bootstrap\_servers='localhost:9092', value\_deserializer=lambda m: json.loads(m.decode('utf-8')))

for message in consumer:

data = message.value

product\_id = data['product\_id']

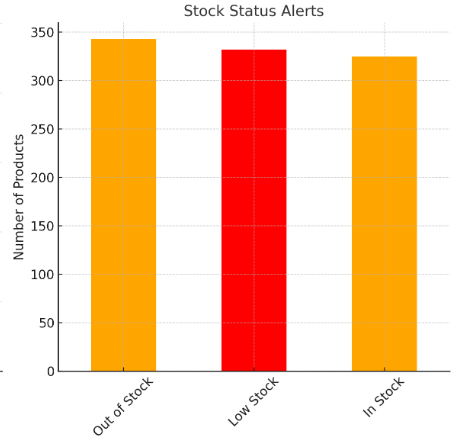
stock\_status = data['product\_stock\_status']

if stock\_status == "Low Stock":

print(f"Alert: Low stock for product {product\_id}.")

elif stock\_status == "Out of Stock":

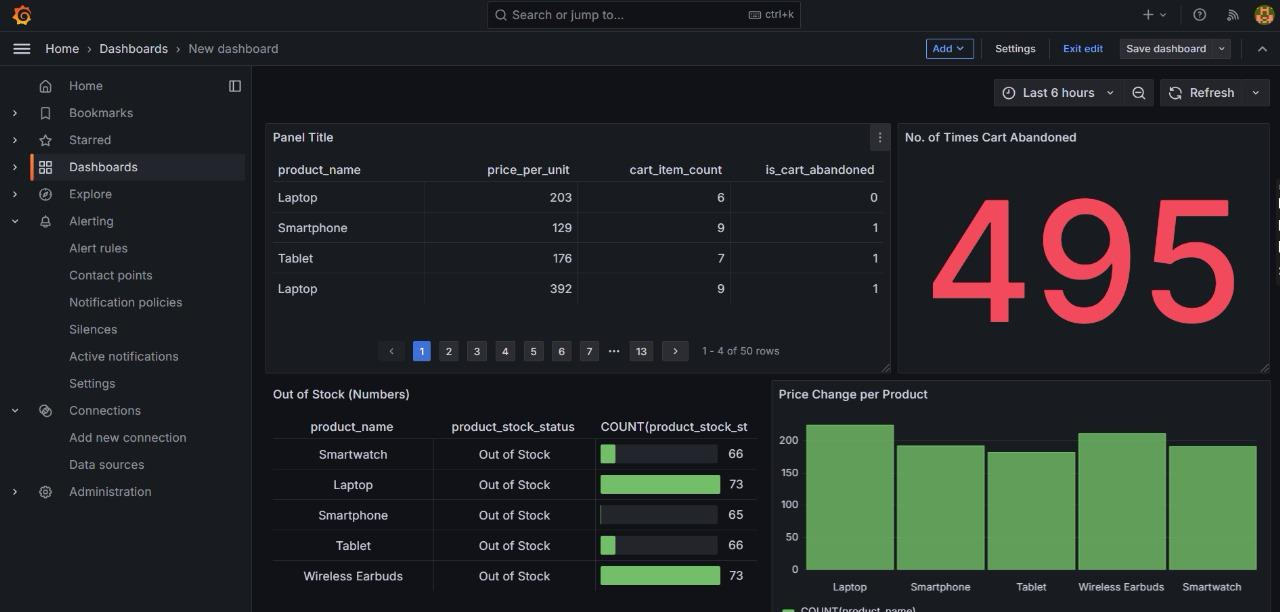
print(f"Alert: Product {product\_id} is out of stock.")



**Integration Details**

* **Kafka**:
  + Producer: Streams incoming data from e-commerce transactions.
  + Consumer: Processes data for alert generation using Python.
* **MongoDB**:
  + Stores historical data and alert logs.
  + Facilitates data retrieval for Grafana dashboards.

**Visualization with Grafana**



**Dashboard Overview**

The dashboard appears to be designed to monitor product sales and inventory for a business. It presents two key areas of concern:

1. **High Cart Abandonment Rate:** This is a critical issue as it indicates that customers are starting the checkout process but not completing it. This could lead to lost revenue and customer dissatisfaction.
2. **Inventory Shortages:** All products shown in the dashboard are currently out of stock. This is a major problem as it directly impacts the ability to fulfill orders and can further contribute to cart abandonment.

**Chart Explanations**

1. **"No. of Times Cart Abandoned" Table:**
   * **Columns:**
     + **Product Name:** Lists the different products being sold (Laptop, Smartphone, Tablet).
     + **Price\_per\_unit:** Shows the selling price of each product.
     + **cart\_item\_count:** Indicates the number of items of each product added to the cart.
     + **is\_cart\_abandoned:** A binary flag (likely 1 for abandoned and 0 for completed) indicating whether the cart containing the product was abandoned.
   * **Key Takeaway:** The table provides a detailed breakdown of cart abandonment by product. This information can be used to identify which products are most frequently associated with abandoned carts, which might suggest issues specific to those products (e.g., high price, lack of availability, insufficient information).
2. **"Out of Stock (Numbers)" Bar Chart:**
   * **X-axis:** Shows the different products.
   * **Y-axis:** Represents the number of times each product has been out of stock.
   * **Key Takeaway:** The chart visually emphasizes the critical issue of inventory shortages. All products are currently out of stock, indicating a serious supply chain or production problem. This lack of availability is likely a significant contributor to cart abandonment.
3. **"Price Change per Product" Bar Chart:**
   * **X-axis:** Shows the different products.
   * **Y-axis:** Represents the percentage change in the price of each product.
   * **Key Takeaway:** The chart shows that the price of laptops has increased the most, followed by smartphones and tablets. This information can be used to:
     + **Analyze the impact of price changes on sales:** Were there any significant drops in sales after the price increases?
     + **Assess the competitiveness of pricing:** Are the price increases justified compared to competitor pricing?
     + **Determine if price changes are contributing to cart abandonment:** Customers may abandon carts if they find prices too high compared to competitors or their expectations.

**Further Analysis & Action Steps**

To address the issues highlighted in the dashboard, further analysis and action steps are necessary:

* **Investigate Cart Abandonment Reasons:** Analyze customer data (if available) to understand the reasons for cart abandonment. Are customers facing technical issues during checkout? Are they confused by the shipping or payment options? Are they concerned about product availability?
* **Improve Inventory Management:** Implement a robust inventory management system to track stock levels accurately, forecast demand, and optimize ordering processes to minimize stockouts.
* **Address Pricing Strategy:** Review pricing decisions to ensure they are competitive and aligned with customer expectations. Consider offering discounts or promotions to incentivize purchases.
* **Optimize Checkout Process:** Streamline the checkout process by reducing the number of steps, offering multiple payment options, and providing clear shipping information.
* **Provide Clear Product Information:** Ensure product pages provide accurate and detailed information, including high-quality images and videos, to build customer confidence and reduce the likelihood of returns.

By addressing these issues proactively, the business can improve customer satisfaction, reduce cart abandonment rates, and increase sales.

**Conclusion**

The proposed solution provides real-time analytics capabilities, empowering the e-commerce business to:

* Recover potential revenue losses.
* Enhance customer satisfaction through timely notifications.
* Optimize inventory management, ensuring better stock control.

This approach demonstrates the business value of real-time analytics in addressing operational challenges and driving growth.