***Real Time Ecommerce Data Analysis***

END SEMESTER PROJECT REPORT FOR FIFTH SEMESTER

***Submitted by***

# BL.EN.U4AIE20033 Monish Mohanty

# BL.EN.U4AIE20071 Varun Vinod Kulkarni

# BL.EN.U4AIE20074 Vishal S

**Big Data and Database Management System**

**21AIE204**

**Bachelor of Technology**

ARTIFICIAL INTELLIGENCE ENGINEERING



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE - 560035

August 2023

**Submitted to:**

**Mr. Saravanan S**

# ABSTRACT:

Real-time e-commerce data analysis refers to the process of collecting, analyzing, and interpreting data from an e-commerce platform in real-time. This can include data on customer behaviour, sales, and product performance. The goal of real-time e-commerce data analysis is to provide businesses with up-to-date insights that can inform decision-making and help them to improve the customer experience, increase sales, and optimize operations. This can be achieved through using various technologies such as Data Warehousing, data mining, real-time streaming analytics and predictive modeling.

An example use case of real-time e-commerce data analysis would be tracking the behaviour of online shoppers in real-time and using this information to make adjustments to website layouts, product recommendations and marketing campaigns. In this way, a business can quickly respond to customer needs and preferences, resulting in improved conversions and increased revenue.

Another example, would be analyzing the real-time inventory data in order to optimize the reorder point and replenishment of products..

In summary, Real-time e-commerce data analysis is a powerful tool that can provide businesses with valuable insights and help them to improve customer experience, increase sales, and optimize operations. It also helps in taking decisions and actions in real time to manage the business better.

# Index

Pg.no.

1. Abstract 2
2. Index 3
3. Introduction 4
4. Literature Survey 5
5. Software Requirements 6
6. Methodology 7
7. Conclusion 10
8. Future Work 11
9. Bibliography 12
10. Source Code 13
11. Order of Execution 18

# INTRODUCTION

Our project aims to provide a Real-time e-commerce data analysis is the process of continuously collecting, analyzing, and interpreting data from an e-commerce platform in real-time, it provides a beneficial insight and analysis capabilities. Our work allows to make the all the data in the market more accessible and understandable to a wide range of users and to provide an easy way to monitor the market and make informed decisions.

The goal of real-time e-commerce data analysis is to provide businesses with up-to-date insights that can inform decision-making and help them to improve the customer experience, increase sales, and optimize operations.

E-commerce platforms generate a vast amount of data, including customer behaviour, sales, and product performance, and real-time e-commerce data analysis provides a way to make sense of this data and turn it into actionable insights. With real-time data analysis, businesses can quickly identify trends, patterns, and customer preferences, which can help them to adapt their strategies and make data-driven decisions.

This type of analysis can be done by using various technologies such as data warehousing, data mining, real-time streaming analytics, and predictive modelling. It can also be done by integrating data from different sources such as social media, website analytics, and customer feedback.

Real-time e-commerce data analysis is becoming increasingly important as online shopping continues to grow in popularity. Businesses that are able to make use of real-time data can gain a competitive edge by being able to respond quickly to customer needs and market trends.

In summary, Real-time e-commerce data analysis allows businesses to make sense of the vast amount of data generated by e-commerce platforms and turn it into actionable insights, which can help them to improve customer experience, increase sales and optimize operations, giving them a competitive edge in the market.

**LITERATURE SURVEY**

A literature survey on real-time e-commerce data analysis would reveal a growing body of research in this area.

One area of research focuses on using real-time data analysis to improve the customer experience. For example, some studies have investigated using real-time data on customer behavior to personalize product recommendations, website layouts, and marketing campaigns. Other research has looked at using real-time customer feedback to improve customer service.

Another area of research focuses on using real-time data analysis to optimize business operations. For example, some studies have investigated using real-time inventory data to optimize the reorder point and replenishment of products. Other research has looked at using real-time data on sales and customer behaviour to optimize pricing strategies.

Predictive modelling is another area of research when it comes to e-commerce data analysis, it can help in forecasting demand, revenue and customer behavior.

Additionally, many studies have also proposed and implemented techniques for real-time data integration and processing, including data warehousing, data mining, and streaming analytics. These techniques are crucial for collecting, cleaning, and processing the large amounts of data generated by e-commerce platforms.

Overall, the literature on real-time e-commerce data analysis highlights the importance of this field for businesses looking to gain a competitive edge by leveraging the vast amount of data generated by e-commerce platforms. Many studies have focused on using real-time data to improve the customer experience, optimize business operations, and make data-driven decisions.

It is important to note that a lot of research on this field are industry or application specific, but most of the time they share similar architecture, technology and principles. Therefore it is important to read and compare multiple publications to see what works best for a specific scenario.

**SOFTWARES**

1. **Dataset:** We use a dataset to get data for our big data analysis. We read each row from the dataset one by one and stream it.
2. **Python:** The system uses Python for writing the script that collects data from the dataset and pushes it to a Kafka topic. Additionally, the system also uses Python's Streamlit library for creating interactive plots and visualizations.
3. **Kafka:** This is an open-source distributed event streaming platform that is used to handle the real-time streaming of data. The system uses Kafka to handle the real-time streaming of data from the dataset to the data processing and storage components of the system.
4. **Apache Spark:** A distributed computing framework that is used for data analysis and transformation, it can handle large dataset and real-time streaming, it allows for the processing of data as soon as it is collected.
5. **MongoDB:** This is a NoSQL database which is used to store the processed data. MongoDB is designed to work with large and unstructured data sets, it's a flexible, scalable, and high-performance database that allow querying and searching the data easily.
6. **Streamlit:** A Python library for creating interactive plots and visualizations, that allows reading the retail data from MongoDB and display it in a real-time line plot.

**METHODOLOGY**

Overall, our project collects, processes and visualizes retail data from dataset in real time. We stream data continuously from the dataset using kafka. The data is retail relevant, it has columns such as category of product, city of the customer. The data is streamed by a producer to kafka row by row. Kafka is used to handle the real-time streaming of data, and Spark is used to perform any necessary data analysis and transformation. The processed data is then stored in MongoDB. Data Visualization is implemented using streamlit, a Python library for creating web hosted apps and visualizations, to read the data from MongoDB and display it in a real-time line plot. This allows users to easily view and understand the data as it changes over time.

Diagram

Description automatically generated

1. **Data Collection**

Data collection is the first step in the process of collecting, processing, and visualizing data from the Dataset. The data collection component of the system involves using the producer.py to push data directly to the kafka topic.

**Producer.py**

The data is collected using a Python script that makes processes the dataset row by row and pushes it to the kafka topic. The script can be designed to run periodically or continuously to gather new data as it becomes available.

Once the data is collected, it is pushed to a Kafka topic. Kafka is an open-source distributed event streaming platform that is used to handle the real-time streaming of data. By pushing the collected data to a Kafka topic, it can be easily consumed by other components of the system for further processing and storage.

1. **Data Processing**

Data processing is a crucial step in the system you described, it involves using Kafka and Spark to process the data and storing it in MongoDB. The data processing component has several sub-components, which are as follows:

1. **Kafka**

The collected data is pushed to a Kafka topic for real-time streaming. Kafka acts as a buffer to handle any differences in data collection rate and data processing rate, it allows data to be processed as soon as it is collected. Kafka is also used to handle the real-time streaming of data from the data collection component to the data processing and storage components of the system.

1. **Spark**

Once the data is in the Kafka topic, it can be consumed by Spark for further processing and transformation. Spark is a distributed computing framework that is designed for big data processing, and it can handle large dataset and real-time streaming. Spark can be used to perform any necessary data analysis and transformation, it allows to perform data cleaning, filtering and aggregation tasks on the data.

1. **MongoDB**

The processed data is then stored in MongoDB, a NoSQL database which is optimized for data that is unstructured or semi-structured, it allows for querying and searching easily. MongoDB stores the data in a flexible, scalable and high-performance way, it can handle large amount of data, it's also easy to scale horizontally by adding more nodes to the cluster.

The data processing component is designed to handle and process large amounts of data in real-time, it's also able to handle data that is unstructured or semi-structured, this is important for the system's requirement to handle real-time data which may have a high rate of change. The use of Apache Spark for data transformation and analysis, and MongoDB for data storage provide high-performance, scalability and flexibility.

1. **Data Visualization**

Data visualization is the final step in the process of collecting, processing, and visualizing data. The data visualization component of the system involves using streamlit, a Python library for creating interactive apps, to read the data from MongoDB and display it in a real-time line plot.

**Bokeh**

Streamlit is a powerful library that allows you to easily create interactive visualizations and plots without the need for JavaScript. It supports a wide range of UI elements and chart types, including line plots, bar charts, scatter plots, and more. The library can be used to create visualizations that can be rendered in a web browser, making them easily accessible to users.

In this case, the system uses Streamlit to read thedata from Kafka, which is used to create a line plot that shows the real-time changes.

The visualization component of the system allows users to easily view and understand the data as it changes over time. This can be useful for retail shop owners who need to monitor sales and trends in real-time. Additionally, the visualization component can be expanded to show more details, to give more insight and analysis capabilities for the user.

It's important to note that the visualization component of the system can be configured to suit the specific needs of the users, it can be customized to display different types of data, or it can be expanded to include additional visualizations and analysis tools, depending on the requirements of the system.

**CONCLUSION**

Our project is a comprehensive system for a Real time Ecommerce Data Analysis . It is designed to provide users with a comprehensive and interactive way to monitor sales prices and trends in real-time.

The system consists of three main components: data collection, data processing, and data visualization. The data collection component uses the dataset to stream data. The data processing component uses Kafka and Spark to process the data and store it in MongoDB, this allows for flexible and scalable data storage. The data visualization component uses Streamlit, a Python library for creating UIs and visualizations, to read the data from MongoDB and display it in a real-time line plot.

The system has been built using open-source technologies, which allows for flexibility, scalability and easy integration with other tools, this makes it adaptable to different use cases. The system allows for real-time data processing and visualization, providing users with an easy way to monitor the market and make informed decisions. The data visualization component has been designed to be easily customizable to suit the specific needs of the users.

Overall, the project has been a success in achieving its goal of providing a valuable tool for professionals in the financial industry and anyone interested in monitoring retail and trends in real-time. The system provides an easy-to-use visualization of the data, valuable insights and analysis capabilities. However, as with any project, there are areas for improvement, the system could be expanded to include additional data sources, analysis tools, and visualizations to provide even more value to users.

**FUTURE WORK**

There are several areas where the project could be improved or expanded upon in the future:

1. Additional data sources: An API to provide data instead of a dataset. An API will provide realitime data and hence will be more accurate.
2. Advanced analytics: Additional analytics tools, machine learning models and statistical techniques could be used to analyze the data and provide more insights,
3. Real-time alerts and machine learning: We can use machine learning algorithms to analyse trends and visualize outputs. This will benefit retail owners

**BIBLIOGRAPHY**

1. Agarwal, R., & Dhar, V. (2014). Editorial—big data, data science, and analytics;

[https://link.springer.com/article/10.1007/s12525-016-0219-0](Rustagi,%20A.%20(2012).%20A%20Near%20Real-Time%20Personalization%20for%20eCommerce%20Platform.%20In:%20Castellanos,%20M.,%20Dayal,%20U.,%20Lehner,%20W.%20(eds)%20Enabling%20Real-Time%20Business%20Intelligence.%20BIRTE%202011.%20Lecture%20Notes%20in%20Business%20Information%20Processing,%20vol%20126.%20Springer,%20Berlin,%20Heidelberg.%20https:/doi.org/10.1007/978-3-642-33500-6_8)

1. Rustagi, A. (2012). A Near Real-Time Personalization for eCommerce Platform. In: Castellanos, M., Dayal, U., Lehner, W. (eds) Enabling Real-Time Business Intelligence. BIRTE 2011. Lecture Notes in Business Information Processing, vol 126. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-642-33500-6_8>
2. Department of Information Technology, VFSTR Deemed to be University, Vadlamudi Village, Guntur District, Andhra Pradesh 522213, India 2 Department of Management Studies, VFSTR Deemed to be University, Vadlamudi Village, Guntur District, Andhra Pradesh 522213, India

[https://d1wqtxts1xzle7.cloudfront.net/63795833/62.2-4\_0420200701-82660-bhlpdr-libre.pdf?1593594706=&response-content-disposition=inline%3B+filename%3DWeighted\_Based\_Frequent\_and\_Infrequent\_P.pdf&Expires=1673494417&Signature=MMeXKic1NFKO8xIhRa7ZwuPiZ-HlEzs](https://d1wqtxts1xzle7.cloudfront.net/63795833/62.2-4_0420200701-82660-bhlpdr-libre.pdf?1593594706=&response-content-disposition=inline%3B+filename%3DWeighted_Based_Frequent_and_Infrequent_P.pdf&Expires=1673494417&Signature=MMeXKic1NFKO8xIhRa7ZwuPiZ-HlEzs0rLmDD~e7S1ttcMDyjyQUwrBF7jFU8CxxTSc-33JiDZqwiX5PjOe7C8QdjtBejPUzP6qd4I2zmqDXIWpk5pYRxzL5NF8rs-e7safEG9GJEqoyrQOC1BGCwog3Z4oFKydTqQuwBK0QKYGfs-5FzU4RV9hx~2DXXa95jE5LtowUwoR2cR1gRYc-oFgeu~YSZp4-mKxbdf5svRaWpIeyvz0fgeDWOxHUHjNvnBSXO2NrHf6DETmqz3xJRLAb1BTFFKnby9Ah73LPhKDBpVW2j~4IzjWbYoFpL1ZjJuD62JCnnvHgBuh5stgE0Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA)

1. <https://docs.bokeh.org/en/2.4.2/index.html>​
2. <https://spark.apache.org/docs/latest/structured-streaming-kafka-integration.html>​
3. <https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#types-of-time-windows>​
4. <https://www.mongodb.com/developer/languages/python/streaming-data-apache-spark-mongodb/>​

​

**SOURCE CODE**

**Producer.py**

#!/usr/bin/python3

# imports

from kafka import KafkaProducer # pip install kafka-python

import numpy as np # pip install numpy

from sys import argv, exit

from time import time, sleep

import pandas as pd

producer = KafkaProducer(bootstrap\_servers='localhost:9092')

# import dataframe

df = pd.read\_csv("/Users/vishalraghav/Desktop/EndSem/Data/dataset.csv")

count = 1

while True:

for index, row in df.iterrows():

msg = f'{row["id"]},{row["country"]},{row["gender"]}'

producer.send('data', bytes(msg, encoding='utf8'))

print(f'sending data to kafka, #{count} ,msg => {msg}')

count += 1

sleep(1)

**StreamHandler.scala**

import org.apache.spark.sql.\_

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.streaming.\_

import org.apache.spark.sql.types.\_

import com.mongodb.spark.\_

import com.mongodb.spark.sql.\_

import org.apache.spark.streaming.\_

object StreamHandler {

def main(args: Array[String]) {

println("Starting Stream Handler")

helper()

}

def helper() = {

val spark = SparkSession

.builder

.appName("Stream Handler")

.config("spark.mongodb.input.uri", "mongodb://127.0.0.1/test.posts")

.config("spark.mongodb.output.uri", "mongodb://127.0.0.1/Trial.stock")

.config("spark.jars.packages", "org.mongodb.spark:mongo-spark-connector:10.0.0")

.getOrCreate()

import spark.implicits.\_

// read from Kafka

val inputDF = spark

.readStream

.format("kafka") // org.apache.spark:spark-sql-kafka-0-10\_2.11:2.4.5

.option("kafka.bootstrap.servers", "localhost:9092")

.option("subscribe", "data")

.option("failOnDataLoss", "false")

.load()

// save only value as a string

val rawDF = inputDF.selectExpr("CAST(value AS STRING)").as[String]

//

val splitDF = rawDF.select(

split($"value", ",").getItem(0).as("id"),

split($"value", ",").getItem(1).as("country"),

split($"value", ",").getItem(2).as("gender")

)

// // if return is positive.....create a new column called positive and set it to 1 else set it to 0

// // if return is negative.....create a new column called negative and set it to 1 else set it to 0

// val spLitDF = splitDF.withColumn("positive", when($"return" > 0, 1).otherwise(0))

// .withColumn("negative", when($"return" < 0, 1).otherwise(0))

// // writing to console

// val query = splitDF.writeStream

// .format("console")

// .outputMode("append")

// .start()

// .awaitTermination()

//writing to mongoDB

val query = splitDF.writeStream

.format("mongodb")

.queryName("ToMDB")

.option("checkpointLocation", "checkpoint")

// .option("forceDeleteTempCheckpointLocation", "true")

.option("spark.mongodb.connection.uri", "mongodb://localhost:27017")

.option("spark.mongodb.database", "Trial")

.option("spark.mongodb.collection", "data")

.outputMode("append")

.start()

.awaitTermination()

}

}

**DashBoard.py**

**import streamlit as st**

**import pandas as pd**

**import numpy as np**

**import pymongo**

**from pymongo import MongoClient**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from datetime import datetime**

**from datetime import date**

**import plotly.express as px**

**from dateutil.parser import parse**

**import plotly**

**import time**

**import plotly.graph\_objects as go**

**## connect to mongodb**

**client = MongoClient('mongodb://localhost:27017')**

**db1 = client.Trial**

**collection\_main = db1.data**

**st.set\_option('deprecation.showPyplotGlobalUse', False)**

**df = pd.DataFrame(list(collection\_main.find()))**

**df = df.drop(['\_id'], axis=1)**

**########################### create the basic streamlit app #############################**

**# setting the screen size**

**st.set\_page\_config(layout="wide",**

**page\_title="Ecommerce Analysis data")**

**# main title**

**st.title('Analysis of Front end data')**

**# main text**

**st.subheader('This app recieves data from the backend and displays it on the front end')**

**st.write('Data: Sample of Seattle listing with Name, Description, Price')**

**st.write("See Profit between dates")**

**d1 = st.date\_input(**

**"From"**

**)**

**d2 = st.date\_input(**

**"To"**

**)**

**def get\_sales\_between(d1,d2):**

**sales = 0**

**for ind in df.index:**

**cd = parse(df['Order Date'][ind]).date()**

**print(cd)**

**if (cd>d1 and cd<d2):**

**sales = sales+float(df['Profit'][ind])**

**return sales**

**if st.button('get'):**

**result = get\_sales\_between(d1, d2)**

**st.write('result: %s' % result)**

**df["Profit"] = df["Profit"].astype(str).astype(float)**

**fig = plt.figure(figsize=(10, 4))**

**City\_Sales = pd.pivot\_table(data=df, index='City', values=[ 'Profit'],**

**aggfunc = 'sum').reset\_index().sort\_values(by='Profit', ascending=False)**

**City\_Sales.sort\_values(by='Profit', ascending=False)**

**sns.barplot(x='City', y='Profit', data=City\_Sales.head(), color='#ff6781', linewidth=2)**

**plt.title('Profit by TOP 5 City', fontsize = 14)**

**plt.xlabel('City')**

**plt.ylabel('Profit')**

**st.pyplot(fig)**

**sale\_profit = pd.pivot\_table(data=df, index=['Category', 'Sub Category'], values=['Profit'],**

**aggfunc='sum').reset\_index().sort\_values(['Category', 'Profit'], ascending=True)**

**px.histogram(sale\_profit, x = "Sub Category", y = "Profit", color="Sub Category", title = "Sales by the Product sub categories")**

**st.plotly\_chart(px.histogram(sale\_profit, x = "Sub Category", y = "Profit", color="Sub Category", title = "Sales by the Product sub categories"))**

**plot\_spot = st.empty()**

**n = len(df)**

**ymax = max(df['Profit'])+5**

**ymin = min(df['Profit'])-5**

**for i in range(0, n-30, 1):**

**sales\_category=df.groupby("Category")["Profit"].sum()**

**with plot\_spot:**

**st.bar\_chart(sales\_category)**

**time.sleep(0.5)**

**Order of Execution**

1. **Start Zookeeper**

zookeeper-server-start.sh /Users/vishalraghav/Desktop/BigData/kafka\_2.13-3.3.1/config/zookeeper.properties

1. **Start Kafka Server**

kafka-server-start.sh /Users/vishalraghav/Desktop/BigData/kafka\_2.13-3.3.1/config/server.properties

1. **Start a Kafka Consumer**

kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic data --from-beginning

1. **Open Mongo Shell and open the desired collection**

Mongosh

use trial

db.data.find()

1. **Start Kafka Producer (producertrial.py)**

python producertrial.py

1. **Run StreamHandler.scala after starting spark shell with the required packages**

spark-shell --conf "spark.mongodb.input.uri=mongodb://127.0.0.1/test.myCollection?readPreference=primaryPreferred" --conf "spark.mongodb.output.uri=mongodb://127.0.0.1/Trial.data" --packages org.apache.spark:spark-sql-kafka-0-10\_2.12:3.3.1,org.mongodb.spark:mongo-spark-connector:10.0.0

**Text

Description automatically generated**

1. **Run Dashboard.py with streamlit**
2. **Run db.collection.find() in mongoshell to view the changes**