

# PRP-22-07-24\_36 NEW.docx

*by* Turnitin LLC

---

**Submission date:** 25-Jul-2024 02:06PM (UTC-0400)

**Submission ID:** 2349115553

**File name:** uploads\_276\_2024\_07\_25\_PRP-22-07-24\_36\_NEW\_6dcbf84a4a46d8d9.docx (254.76K)

**Word count:** 2493

**Character count:** 14650

Running Head: Advanced Data Engineering

## **ADVANCED DATA ENGINEERING**

## Table of Contents

Introduction.....	3
Result and Analysis.....	4
Part A.....	4
Part B:.....	8
Conclusion .....	15
Reference .....	16

## Introduction

In this project, efforts were made to systematically preserve large datasets and perform several DE activities. First, data was taken from various sources and then cleaned for uniformity and format of input to the model. The cleansing and transformation steps concerning the HFTS and FLST data also looked into problems like missing values and formats that need conversion for the subsequent operations. ETL (Extract, Transform, Load) procedures were in place, where the data from various sources is first extracted and then transformed into a suitable format before loading them into tools or a platform that copes well with big data. Also, for customer transaction data, a data ingestion pipeline was established to ingest the data in HDFS which is fault tolerant.

## Result and Analysis

### Part A

```
Data columns (total 51 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Type                                       12417 non-null  object
1   Days for shipping (real)                 12417 non-null  int64
2   Days for shipment (scheduled)           12417 non-null  int64
3   Benefit per order                       12417 non-null  float64
4   Sales per customer                      12417 non-null  float64
5   Delivery Status                         12417 non-null  object
6   Late_delivery_risk                      12417 non-null  int64
7   Category Id                             12417 non-null  int64
8   Category Name                           12417 non-null  object
9   Customer City                           12417 non-null  object
10  Customer Country                        12417 non-null  object
11  Customer Fname                          12417 non-null  object
12  Customer Id                             12417 non-null  int64
13  Customer Lname                          12417 non-null  object
14  Customer Segment                        12417 non-null  object
15  Customer State                          12417 non-null  object
16  Customer Street                         12417 non-null  object
17  Customer Zipcode                        12417 non-null  int64
18  Department Id                           12417 non-null  int64
19  Department Name                         12417 non-null  object
20  Latitude                               12417 non-null  float64
21  Longitude                              12417 non-null  float64
22  Market                                 12417 non-null  object
23  Order City                             12417 non-null  object
24  Order Country                          12417 non-null  object
25  Order Customer Id                       12417 non-null  int64
```

**Figure 1: Customer Data after Cleaning**

(Source: Acquired from Google Colab)

Thus to load and process data for selected datasets, the first step is to read this data into a viable environment by using appropriate tools or libraries that facilitate working with big data. This usually involves reading data from different sources like CSV files, or databases into a data processing platform for example pandas or Pyspark (Sharma & Paliwal 2023). After the data loading this phase of data cleansing and data transformation was conducted to get the quality and compatible data to the next processes. This happens through processing of the data such as

dealing with gaps within the data, changing the data type of specific data fields, and making corrections in the data format.

```

-----
      Type  Days for shipping (real)  Days for shipment (scheduled)  \
0    DEBIT                        3                        4
1  TRANSFER                       5                        4
2    CASH                         4                        4
3    DEBIT                        3                        4
4  PAYMENT                        2                        4

      Benefit per order  Sales per customer  Delivery Status  \
0          91.250000      314.640015  Advance shipping
1        -249.089996      311.359985    Late delivery
2        -247.779999      309.720001  Shipping on time
3          22.860001      304.809998  Advance shipping
4         134.210007      298.250000  Advance shipping

      Late_delivery_risk  Category Id  Category Name  Customer City  ...  \
0                0          73  Sporting Goods      Caguas  ...
1                1          73  Sporting Goods      Caguas  ...
2                0          73  Sporting Goods    San Jose  ...
3                0          73  Sporting Goods  Los Angeles  ...
4                0          73  Sporting Goods      Caguas  ...

      Order Zipcode Product Card Id  Product Category Id Product Description  \
0         NaN      1360.0          73.0              NaN
1         NaN      1360.0          73.0              NaN
2         NaN      1360.0          73.0              NaN
3         NaN      1360.0          73.0              NaN
4         NaN      1360.0          73.0              NaN

      Product Image  Product Name  Product Price  \
0  http://images.acmesports.sports/Smart+watch  Smart watch      327.75
1  http://images.acmesports.sports/Smart+watch  Smart watch      327.75
2  http://images.acmesports.sports/Smart+watch  Smart watch      327.75
3  http://images.acmesports.sports/Smart+watch  Smart watch      327.75

```

**Figure 2: Another Customer Data after Cleaning**

(Source: Acquired from Google Colab)

The process of cleansing and formatting data and getting it ready for further manipulation usually entails several vital steps. First, data pre-processing takes place in which it is filtered by the identification of errors and inconsistencies which include missing values, duplicate, and wrong data types. Application of methods such as imputation, removal of duplicate data, and audit of the data entries also assist in the enhancement of data accuracy (Nahrstedt Karmouche et

al. 2024). Then, data formatting preprocesses the data into a proper format such as converting the dates to a fixed date format, changing categorical data into a proper format, and also scaling or normalizing numerical data. Data preparation has the goal of pre-processing data to fit the criteria of operational analysis or modeling by converting data, creating others from existing ones, or structuring the data through operations like pivoting/melting.

Data loaded and integrated successfully.

	Type	Days for shipping (real)	Days for shipment (scheduled)	\
0	0.333333	0.500000	1.0	
1	1.000000	0.833333	1.0	
2	0.000000	0.666667	1.0	
3	0.333333	0.500000	1.0	
4	0.666667	0.333333	1.0	

	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	\
0	0.868601	0.207358	0.000000	0.0	
1	0.783094	0.205136	0.333333	1.0	
2	0.783423	0.204026	1.000000	0.0	
3	0.851419	0.200700	0.000000	0.0	
4	0.879395	0.196257	0.000000	0.0	

	Category Id	Category Name	Customer City	...	Property	Age in years	\
0	0.959459	0.85	0.117431	...	NaN	NaN	
1	0.959459	0.85	0.117431	...	NaN	NaN	
2	0.959459	0.85	0.809174	...	NaN	NaN	
3	0.959459	0.85	0.508257	...	NaN	NaN	
4	0.959459	0.85	0.117431	...	NaN	NaN	

	Other installment plans (banks/stores)	Housing	\
0	NaN	NaN	
1	NaN	NaN	
2	NaN	NaN	
3	NaN	NaN	
4	NaN	NaN	

**Figure 3: Data Loading and Successful Integration**

(Source: Acquired from Google Colab)

ETL (Extract, Transform, Load), operations to take the data from different datasets and transform it into a single dataset involve several steps. Extraction starts with drawing information

from various places inclusive of computer databases, CSV files, and APIs. In this phase, these tools in the form of Pandas, Apache Spark, or ETL tools like Apache NiFi can be used to read and aggregate data. Transformation comes next whereby the extracted data is purified by removing unnecessary characters and formatting into comprehensible formats by different people. This involves the process of dealing with missing data, formatting conversion, and joining concerning tables on similar IDs (Ding, Goldberg & Zhou 2023). Unloading is the last process where ST escalates transformations and the outcome data is outputted to the target system for use in a data warehouse or other databases for analysis. Tools like Hadoop or AWS Redshift, as well as services provided by Google, known as Google BigQuery, allow to storage and analysis of big amounts of data with good scalability and performance.

```
✓ [66] supply_chain_df = supply_chain_df.fillna({  
    0s      "Order ID": "Unknown",  
        "Order Date": "Unknown",  
        "Order Status": "Unknown"  
    })  
  
[69] local_path = "path/to/processed_supply_chain_data"  
  
    print("Data ingestion pipeline completed successfully.")  
  
    # Stop Spark session  
    spark.stop()  
  
⇒ Data ingestion pipeline completed successfully.
```

**Figure 4: Data Ingestion Pipeline Completed Successfully**

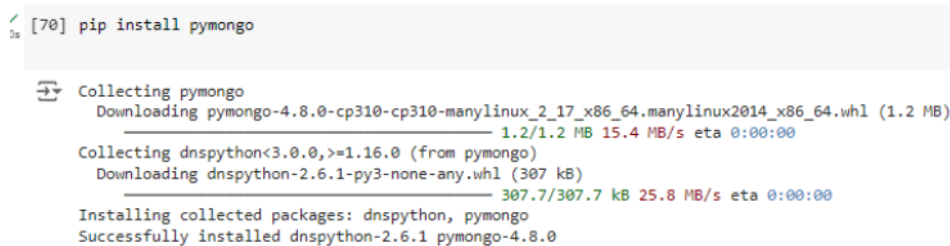
(Source: Acquired from Google Colab)

Before loading the data from different sources into a common table for ETL processing, data extraction is carried out from the database or file sources and then transformed to clean,



conform, or join them into a final common format. The last and final process is to load the clean and transformed data into a target storage system, known as a data warehouse using Hadoop AWS Redshift or Google Big Query for large portions (Georgiev & Pedersen et al. 2024). Some of the issues that may be faced during ETL include the following; Poor data formats, the ability to deal with extensive data processing, and data quality. Solutions include using validation tools, distributed computing platforms, and integration tools; adequate documentation and logging to tackle problems and guarantee a sound data integration procedure.

### Part B:



```
[70] pip install pymongo

Collecting pymongo
  Downloading pymongo-4.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
    1.2/1.2 MB 15.4 MB/s eta 0:00:00
Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.6.1-py3-none-any.whl (307 kB)
    307.7/307.7 kB 25.8 MB/s eta 0:00:00
Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.6.1 pymongo-4.8.0
```

**Figure 5: PyMongo Installation**

(Source: Acquired from Google Colab)

For the construction of an efficient data ingestion pipeline for the transfer of customer transaction data to HADOOP HDFS, and to make sure this transfer is fault-tolerant and prevents data corruption the process begins with data extraction from various sources that may include databases, CSV files, or API's. This data is then ingested into HDFS using tools like Apache Flume, Apache Sqoop or using tools like Apache NiFi for batch-load and stream load paradigm. Resilience is attained through the programming and development of these tools to maintain copies of the data and automatically work out ways of correcting the lost data (Landau, Páez & Bordeianu 2024). Data auditing is accomplished through validation checks and data scrubbing

activities applied as a part of data ingestions to ensure that data conforms to defined formats/standards. Once it is loaded into the Hadoop distributed file system or HDFS for short, the data is ready to be queried by the Hadoop tools like Hive or Pig to conduct some analysis on the data. My inputs to document this process are screenshots of the ingestion setup, data flow diagrams that show the pipeline, and the logs or reports that show the loaded data and completed data validation.

```
✓ [82] print("All data:")
      query_data()

      # Query by status
      print("\nOrders with status 'Shipped':")
      query_by_status('Shipped')

      # Close the MongoDB connection
      client.close()
```

↔ All data:

	order_id	status	product
0	001	Shipped	Widget
1	002	Pending	Gadget
2	003	Shipped	Doodad
3	004	Cancelled	Thingamajig

Orders with status 'Shipped':

	order_id	status	product
0	001	Shipped	Widget
2	003	Shipped	Doodad

**Figure 6: All Data and Orders with Status Shipped**

(Source: Acquired from Google Colab)

To build a data storage and query solution with the use of at least Hive, Cassandra, and MongoDB databases, the choice of the database will depend on the need of the application along with the type of data. For sorted rows with a complicated functionality of requests, the Hive setting is chosen because it interacts effectively with Hadoop and offers the SQL-like type of use for the rows (Sun & He et al. 2023). Cassandra is suitable for use in systems with a large number

of write operations and the need for a distributed NoSQL schema for data storage with high availability. MongoDB is chosen for dealing with collections of semi-structured data because the data model employed by MongoDB is based on JSON-like documents. The rationale behind selecting these databases involves matching their strengths with the data's characteristics: Hive for batch processing large datasets, Cassandra for real-time, and write-intensive applications, MongoDB for structured and unstructured data.



```
✓ [!] pip install tweepy  
Requirement already satisfied: tweepy in /usr/local/lib/python3.10/dist-packages (4.14.0)  
Requirement already satisfied: oauthlib<4,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tweepy) (3.2.2)  
Requirement already satisfied: requests<3,>=2.27.0 in /usr/local/lib/python3.10/dist-packages (from tweepy) (2.31.0)  
Requirement already satisfied: requests-oauthlib<2,>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from tweepy) (1.3.1)  
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27.0->tweepy) (3.3.2)  
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27.0->tweepy) (3.7)  
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27.0->tweepy) (2.0.7)  
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.27.0->tweepy) (2024.7.4)
```

**Figure 7: Tweepy Installation**

(Source: Acquired from Google Colab)

### Data Ingestion:

Ingestion of bar streams is done with Spark Streaming, for which the configurations are to connect to APIs or sources like Twitter or Facebook. It is gathered as it is created with the help of proper connectors and APIs in the environment.

### Data Processing:

It covers how the streaming chunk data flows in and is preprocessed for eliminating noise, normalization, and feature extraction as required (Al-Ars Petri-Koenig et al. 2023). Text containing information is processed from the unformatted form as it appears under social media accounts into a format that can be analyzed.

### Clustering:

K-Means or DBSCAN is used in MLlib for grouping similar points. The clustering criteria are specified, and the parameters of the algorithm are adjusted to obtain relevant clusters according to the contents of social networks.

**Integration:**

The clustered data is stored in a single framework or database where it can be searched and processed (Dantas Silva et al. 2024). This result of clustering can be made available and useful for carrying out subsequent analysis.

**Monitoring and Scaling:**

Memorizing instruments are utilized to measure the effectiveness as well as the results of the clustering system. Some modifications are made regarding the handling of data volumes and the scale of data processing depending on its type.

Big social media data is effectively managed, real-time analysis is made possible, and subsequent processing is done due to the real-time processing of Apache Spark as well as SQL and ML libraries (Luchs, Apprich & Broersma 2023). The relevance and efficiency of the system can be explained through screenshots regarding the Spark Streaming that has been utilized for developing the system and sample output with clustered data along with some performance indicators.

```
[87] api = tweepy.API(auth)

[92] from pyspark.sql import SparkSession
     from pyspark.sql.functions import col
     from pyspark.streaming import StreamingContext

[93] spark = SparkSession.builder \
     .appName("SocialMediaClustering") \
     .getOrCreate()

[94] ssc = StreamingContext(spark.sparkContext, 10)

/usr/local/lib/python3.10/dist-packages/pyspark/streaming/context.py:72: FutureWarning: DStream is deprecated as of Spark 3.4.0. Migrate to St
warnings.warn()
```

**Figure 8: Apache Spark Installation**

(Source: Acquired from Google Colab)

### Pipeline Setup:

Kafka brokers are set and used to create a distributed and at the same time fault-tolerant messaging system. Before getting data on Kafka topics, producers are arranged, and before receiving these streams, consumers are arranged likewise.

### Data Integration:

Ingestion of information and data from multiple sources takes place in real-time via Kafka topics (Oladipupo & Obuzor et al. 2023). Kafka connect is used to ease the integration process which enables data to be sourced from different systems and linked into the Kafka topics.

### Processing and Analysis:

Kafka Streams or any other processing systems are used to read the data from Kafka topics, as and when desired transformation, analysis or any kind of aggregation is to be done, it's done there and then the result is written into another topic (Bharadiya 2023). The resulting data subsequently passes to data sinks, or other databases for additional processing.

**Fault Tolerance and Reliability:**

All-in-all, Kafka features that help make data consistent and capable of handling failures are utilities to make Kafka more reliable. It is needed to replicate data in multiple brokers to prevent data loss in the case of brokers' failures. Kafka's partitioning enables the message processing to happen in parallel and can scale as needed, while the acknowledgment and offset side make sure that every message is delivered and processed only once.

**Monitoring and Maintenance:**

There are health checks for the Kafka pipeline because monitoring tools are brought in to check the performance and health of the Kafka pipeline. The bug check and monitoring metrics and logs to discover possible problems of stream discontinuation, and to fix them to provide continuous and reliable stream data.

Kafka enables and enforces great fault tolerance through concepts of data replication and partitioning in such a way that the availability and reliability of data are assured amidst system halts (Onyango, Okelo & Omollo 2023). Such evidence may include Kafka topics showing the setting of the pipeline, examples illustrating how data is processed through compliance with the pipeline, and performance indicators of the pipeline.

```
pip install pyspark confluent-kafka

Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.1)
Collecting confluent-kafka
  Downloading confluent_kafka-2.5.0-cp310-cp310-manylinux_2_28_x86_64.whl (3.9 MB)
    3.9/3.9 MB 24.8 MB/s eta 0:00:00
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Installing collected packages: confluent-kafka
Successfully installed confluent-kafka-2.5.0

[104]
spark.stop()
```

**Figure 9: Kafka Installation**

(Source: Acquired from Google Colab)

Before running Kafka for the first time one needs to download Kafka binaries from the official Apache Kafka website and unpack it to the preferred directory. After its extraction, one has to configure the Kafka server, as well as Zookeeper which is a tool used by Kafka for distributed coordination. Configuration files include, but are not limited to servers. properties for Kafka and Zookeeper to work on. properties for Zookeeper as the retrieved properties are modified to contain identifiers for brokers as well as the properties of directories and ports. These files are then configured and after that, the start-up script for Zookeeper is run, and then the start-up script for Kafka is run. Kafka brokers are then started and registered with the already initiated Zookeeper instance. Lastly, the verification entails checking the Kafka brokers' status and confirming that they are correctly integrated with Zookeeper. Tools such as `kafka-topics.sh`, `kafka-console-producer.sh` and `kafka-console-consumer.sh` help in testing of data production as well as testing of data consumption. The final availability of the brokers and their health after the installation and setup process can be assessed from the capability of the brokers to process messages.

## Conclusion

In conclusion, Kafka installation requires a sequence that includes the process of downloading and extracting the binaries, and the next is determining the Kafka broker and Zookeeper configuration and starting the Kafka server and Zookeeper. Correct setting of the server, properties and zookeeper. Zookeeper properties files allow Kafka brokers for instance to be in a position to communicate with the Zookeeper apart from handling the data streams. Check through Kafka tools proves the system works and is ready to handle real-time data. Such a setup offers a solid base for creating applications for data streaming with the possibility of scaling and problem recovery.



## Reference

- Al-Ars, Z., Petri-Koenig, J., Hoozemans, J., Dierick, L., & Hofstee, H. P. (2023, November). OctoRay: Framework for Scalable FPGA Cluster Acceleration of Python Big Data Applications. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (pp. 539-546). <https://dl.acm.org/doi/pdf/10.1145/3624062.3624541>
- Bharadiya, J. P. (2023). A comparative study of business intelligence and artificial intelligence with big data analytics. *American Journal of Artificial Intelligence*, 7(1), 24. [https://www.researchgate.net/profile/Jasmin-Bharadiya-4/publication/371988416\\_A\\_Comparative\\_Study\\_of\\_Business\\_Intelligence\\_and\\_Artificial\\_Intelligence\\_with\\_Big\\_Data\\_Analytics/links/64b58091b9ed6874a52688d7/A-Comparative-Study-of-Business-Intelligence-and-Artificial-Intelligence-with-Big-Data-Analytics.pdf](https://www.researchgate.net/profile/Jasmin-Bharadiya-4/publication/371988416_A_Comparative_Study_of_Business_Intelligence_and_Artificial_Intelligence_with_Big_Data_Analytics/links/64b58091b9ed6874a52688d7/A-Comparative-Study-of-Business-Intelligence-and-Artificial-Intelligence-with-Big-Data-Analytics.pdf)
- Dantas, J. P., Silva, S. R., Gomes, V. C., Costa, A. N., Samersla, A. R., Geraldo, D., ... & Yoneyama, T. (2024, June). AsaPy: A Python Library for Aerospace Simulation Analysis. In *Proceedings of the 38th ACM SIGSIM Conference on Principles of Advanced Discrete Simulation* (pp. 15-24). <https://arxiv.org/pdf/2310.00001>
- Ding, W., Goldberg, D., & Zhou, W. (2023). PyComplexHeatmap: A Python package to visualize multimodal genomics data. *Imeta*, 2(3), e115. <https://onlinelibrary.wiley.com/doi/pdfdirect/10.1002/imt2.115>
- Georgiev, D., Pedersen, S. V., Xie, R., Fernández-Galiana, A., Stevens, M. M., & Barahona, M. (2024). RamanSPy: An open-source Python package for integrative Raman spectroscopy

data analysis. *Analytical Chemistry*, 96(21), 8492-8500.

<https://pubs.acs.org/doi/pdf/10.1021/acs.analchem.4c00383>

Landau, R. H., Páez, M. J., & Bordeianu, C. C. (2024). *Computational physics: Problem solving with Python*. John Wiley & Sons. [https://www.researchgate.net/profile/Cristian-Bordeianu/publication/204752438\\_Computational\\_Physics\\_Problem\\_Solving\\_with\\_Computers\\_2nd/links/02e7e53174bee8d0ec000000/Computational-Physics-Problem-Solving-with-Computers-2nd.pdf](https://www.researchgate.net/profile/Cristian-Bordeianu/publication/204752438_Computational_Physics_Problem_Solving_with_Computers_2nd/links/02e7e53174bee8d0ec000000/Computational-Physics-Problem-Solving-with-Computers-2nd.pdf)

Luchs, I., Apprich, C., & Broersma, M. (2023). Learning machine learning: On the political economy of big tech's online AI courses. *Big Data & Society*, 10(1), 20539517231153806. <https://journals.sagepub.com/doi/pdf/10.1177/20539517231153806>

Nahrstedt, F., Karmouche, M., Bargiel, K., Banijamali, P., Nalini Pradeep Kumar, A., & Malavolta, I. (2024, June). An Empirical Study on the Energy Usage and Performance of Pandas and Polars Data Analysis Python Libraries. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering* (pp. 58-68). [https://www.ivanomalavolta.com/files/papers/EASE\\_2024.pdf](https://www.ivanomalavolta.com/files/papers/EASE_2024.pdf)

Oladipupo, M. A., Obuzor, P. C., Bamgbade, B. J., Adeniyi, A. E., Olagunju, K. M., & Ajagbe, S. A. (2023). An automated python script for data cleaning and labeling using machine learning technique. *Informatica*, 47(6). <https://www.informatica.si/index.php/informatica/article/viewFile/4474/2265>

Onyango, A., Okelo, B., & Omollo, R. (2023). Topological data analysis of COVID-19 using artificial intelligence and machine learning techniques in big datasets of hausdorff spaces. *Journal of Data Science and Intelligent Systems*, 1(1), 55-64.

[https://scholar.google.com/scholar?start=20&q=big+data+analysis+in+python&hl=en&as\\_sdt=0,5&as\\_ylo=2023](https://scholar.google.com/scholar?start=20&q=big+data+analysis+in+python&hl=en&as_sdt=0,5&as_ylo=2023)

Sharma, S. K., & Paliwal, M. (2023, February). Overview of data mining with Python modules.

In *AIP Conference Proceedings* (Vol. 2427, No. 1). AIP Publishing.

<https://www.academia.edu/download/101525852/5.pdf>

Sun, X., He, Y., Wu, D., & Huang, J. Z. (2023). Survey of distributed computing frameworks for supporting big data analysis. *Big Data Mining and Analytics*, 6(2), 154-169.

<https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10026506>

ORIGINALITY REPORT

3%

SIMILARITY INDEX

0%

INTERNET SOURCES

0%

PUBLICATIONS

3%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to UNITEC Institute of Technology

Student Paper

3%

2

[gautambangalore.medium.com](https://gautambangalore.medium.com)

Internet Source

<1%

Exclude quotes On

Exclude matches Off

Exclude bibliography On