International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 12 Issue 3 ǁ March 2024 ǁ PP. 81-99

Developing a Blockchain Model for Supply Chain

Abe Zeid*, Sameed Awan, Moises Cohen Bernal, Zhongsheng Chang, Nijel Hunt, Yuehan Zhen
**Corresponding author*

Abstract
Modern supply chains face new challenges, including delivery speed, quality, service, and cost. Supply chain data include the product ID (SKU), shipping origin, receiving entity, shipping date, receiving date, quantity, cost per item, and total cost, to name a few. Traditional supply chain models such as the continuous flow, agile, fast chain, flexible, and efficient chain models must be automated and developed into digital models to meet current and future challenges. At the heart of all these models is a tracking system that looks at the movements of components, products, and services through the production, from raw materials to finished products to consumers. Supply chain data needs to be stored in a secure and immutable ledger that cannot be tampered with. Blockchain is an up-and-coming digital technology that has the potential to revolutionize the supply chain. This paper conceives, designs, and implements a blockchain-based digital model for a supply chain application. The model should serve as an aid to supply chain visibility and product tracing, and it should help streamline transactions and speed up the flow of information. We apply our model to real-life supply chain dataset, and we discuss different activities to prepare the dataset for the blockchain model. We also compare the blockchain model to traditional SQL databases as well as Google Big Query (GBQ) model.

Date of Submission: 25-02-2024 Date of acceptance: 05-03-2024

I. Introduction

The complexity of the supply chain leads to supply chain data management issues. The supply chain databases built in traditional tools such as MySQL and Google Big Query suffer from a lack of security, transparency, low constructing and processing speed, inconvenience in data managing and tracking, and high construction and operational costs.

Blockchain has been used successfully by cryptocurrencies, Bitcoin, in particular. As it demonstrated its ability in decentralization, companies have explored its use in the supply chain. Blockchain has shown promising transparency, safety, and efficiency results compared to traditional centralized databases. Blockchain has not been applied in the supply chain.

Using the data source from a biomedical company, we will explore the differences between a Blockchain database and a centralized one by using an open-sourced Blockchain platform and Google Big Query. The results will provide more clarity on the benefits of Blockchain and potentially have long-lasting effects across all supply chains.

II. Background research

Blockchain is essentially a public database consisting of connected ledgers. A "Block" refers to data and state being stored in sequential batches or "blocks." Moreover, "chain" refers to the chronological order of blocks. After the first block, called the Genesis block, is created, the next block is added. Blocks created earlier get added to the chain first. A block's data cannot be changed without changing all subsequent blocks, which would require the consensus of the entire network. [1]

Each party within the system has its ledger, and the ledgers communicate and check with each other. When an action occurs, the two participating parties each record the action, and then the record is communicated with everyone across the system. The "chain" can be viewed as a line that stretches as more actions occur.

When people think of blockchain technology, the first thing that comes to mind may be the financial sector or Bitcoin (peer-to-peer financial transaction system). This thinking would make sense as blockchain technology specializes in security and transparency, and there are hundreds of billions of financial transactions every day throughout the world. However, blockchain technology can be used by many different sectors, one being supply-chain. "Currently, blockchain technology is applied to various financial fields, including business services, settlement of financial assets, prediction markets, and economic transactions [2]. Blockchain is expected to play an essential role in the sustainable development of the global economy, bringing benefits to consumers, the current banking system, and the whole society in general [3]." Other significant sectors that use

Blockchain are Healthcare, government (voting, taxes, record management), and Big Data/Cybersecurity. These different sectors need the best security for valuable and sensitive user data. Blockchain in all use-cases results in operational improvements, more transparency and security, and many other benefits. Based upon the indented use of the Blockchain, three different generations of blockchains can be distinguished, at least so far. "Blockchain 1.0 which includes applications enabling digital cryptocurrency transactions (Bitcoin); Blockchain 2.0, which has smart contracts and a set of applications extending beyond cryptocurrency transactions (Ethereum); and Blockchain 3.0, which includes applications in areas beyond the previous two versions, such as government, health, science, IoT, and Ethereum 2.0 or Solana [4]"

A challenge to Blockchain is the speed of distribution. Because of the physical distance and difference in internet speed between users, they receive the record at a different rate, creating a discrepancy in the overall record. This problem is solved by limiting the speed of creating a new "block" or record. To create a new block, we need to crack a code by brute force (trying the number one-by-one for a considerable combination, for example, a 2^72 number combination). As the speed of block creation remains low, fewer records are distributed, and the order in which every user receives the record remains organized.

Even though keeping the speed of the creating blocks low helps keep the chain organized, that alone is not enough because the chain consists of blocks added sequentially. It takes a long time to generate new blocks when the chain splits into different lines because the sequence is inconsistent. The longest chain is commonly accepted to keep adding on new blocks.

The limit on the creation of new blocks makes faking blocks impossible. For a user to fake a block, they need to create the longest chain among all users to be accepted into adding new blocks. The user needs to have a faster computational power than all other users combined. As common sense, a single user's computational power would not exceed everyone else.

Based on the nature of the blockchain system, cryptocurrencies are created so that each user who creates or discovers a new block is rewarded a certain amount of the cryptocurrency. Essentially, users are rewarded for increasing the length of the chain by creating new blocks, which means a sacrifice of time and computational power to crack the code by brutal force.

In an article published in Harvard Business Review, Gaur and Gaiha give an example of simple transactions to illustrate the differences between conventional record-keeping and Blockchain in the supply chain. A retailer sources a product from a supplier, and a bank provides money for the supplier. In conventional record-keeping, a party is not aware of the records of interactions between the other two parties. However, Blockchain can solve the problem because every party can view the blocks along the sequential chain. [5]

III. Advantages and Challenges of using of Blockchain

Blockchain is transparent, safe, and efficient. In supply chain applications, it can: 1. enhance traceability, 2. increase efficiency, speed, and reduce disruptions, 3. improve financing, contracting, and international transactions, according to studies done by Gaur and Gaiha. [5] However, there are challenges to the supply chain blockchain applications. While cryptocurrencies operate on computers and the internet, the supply chain requires hardware compliance to provide the data for the Blockchain. Integrating blockchain technology into the supply chain may require a high cost for companies to meet the compliance requirements.

While Blockchain is a new and upcoming technology, several challenges must be overcome before becoming mainstream. Some of the main challenges Blockchain faces are technical issues, organizational challenges, and environmental concerns. The technical issues blockchain faces are a "lack of scalability, lack of interoperability, stand-alone projects, difficult integration with legacy systems, complexity and lack of blockchain talent." [6 - 9]. Blockchain must be addressed to be used on a larger scale. Existing transaction networks are much more efficient at handling transactions, the credit card companyVisa, for example, can process more than 2000 transactions per second. However, the two largest blockchain networks, Bitcoin and Ethereum, are far behind in transaction speeds. While the Bitcoin blockchain can process three to seven transactions per second, Ethereum can handle approximately 20 transactions in a second." [6]. This problem will not matter for private blockchain networks but for more extensive scaled operations. Another critical challenge is making it easier for corporations to integrate Blockchain into their existing systems. There is a lack of skilled developers able to do this, and there are many high incidences of data loss and breaches, which discourages companies from switching to Blockchain. A significant problem facing many corporations is the lack of regulations regarding Blockchain, and there is no centralized regulatory structure.[6] Something which may hinder the growth of Blockchain in the future is the vast amount of energy that is consumed by computers to run Blockchain. Since climate change is an ever-growing problem, creating an energy-efficient way to produce Blockchain.

1 Existing Solutions of Blockchain Applications in Supply Chain

Blockchain is an emerging technology, and only some of the largest companies have tried implementing Blockchain in supply chain. Many projects and proof of concept have been created in the past years and are expected to finish launching in two years.

TrustChain Jewelry is one of those proof of concept [10]. TrustChain is a blockchain initiative focused on the end-to-end supply chain of gold and gemstones. Their objective is to give confidence to consumers that their work is entirely ethical and that there is no need to worry about forced labor or fraudulent gems. Since Blockchain is one of the safest ways to store information and track it back in time, this is one of the great uses for tracking products, ensuring it is ethical and accurate.

One of the most prominent companies in this new initiative of Blockchain is IBM. They are creating a new branch, specifically dedicated to their new blockchain services to other companies such as Walmart and Ford [10]. Walmart is planning on using the IBM blockchain to create projects and proof of concepts inside their supply chain structure, to increase the traceability of some of their products. This includes tracing the origins of mangoes, tracking the pork supply in the company's stores in China, and collaborating with other companies to find a blockchain solution to pharmaceutical products' supply chain traceability. On the side of Ford, they are teaming up with IBM to create a blockchain that will help them ensure the ethical procurement of cobalt, a mineral that is increasing in use for the production of electric cars batteries [10]. Blockchain is slowly but steadily becoming the new standard for data storing and databases inside many big companies. Future proof of concepts, these examples mentioned here are just the general uses. Better traceability, ethical procurement, and end-to-end visibility are just some of the uses we have been able to identify, and we are sure there are many more to come.

We decided to look into private blockchains because businesses do not want to keep confidential data on a public blockchain like Ethereum. During our research, we found Hyperledger Fabric, an IBM-created platform. As a permissioned blockchain network, Hyperledger can solve this problem, and it offers the facility to create a blockchain application maintaining the privacy of the organization's information. Hyperledger is coded primarily on Java or Golang languages. Hyperledger has strict control over accessibility for users; only people granted access can create new blocks or see the data, which means it is highly secure. We learned that Walmart wanted to create a way to track where they get their food to stop an outbreak of food-borne disease, so they decided to create a food traceability system on Hyperledger fabric[10]. Walmart "ran two proof of concept projects to test the system. One project was about tracing mangos sold in Walmart's US stores, and the other aimed to trace pork sold in its China stores. The Hyperledger Fabric blockchain-based food traceability system built for the two products worked. For pork in China, it allowed uploading certificates of authenticity to the Blockchain, bringing more trust to a system where that used to be a serious issue. Furthermore, for mangoes in the US, the time needed to trace their provenance went from 7 days to… 2.2 seconds!" [10]

In addition to IBM blackchain platform, there is Tradelines platform. Tradelines is a blockchain application built on IBM Hyperledger. Trade lens is an interconnected ecosystem of supply chain partners - cargo owners, ocean and inland carriers, freight forwarders and logistics providers, ports and terminals, customs authorities, etc. [11]. Tradelines Focus is on digitizing information and exchange between the different parts of a supply chain. The goal is to access a supply chain and logistics network with over half of global container shipping volume data coverage. It runs on a permission matrix and Blockchain, ensuring every party to a shipment has access only to their information and a secure audit trail of transactions. It is a global platform built on open source technology and publicly-available APIs. Many companies across the world are currently using tradelines. The Abu Dhabi board of customs just joined the organization and plans on fully digitalizing all shipments going through their ports. [12] This plan will allow the authorities who receive the cargo to see all information on shipments from before they were loaded onto the ships. Blockchain authenticates the information to trust that the information is accurate.

2 An Example

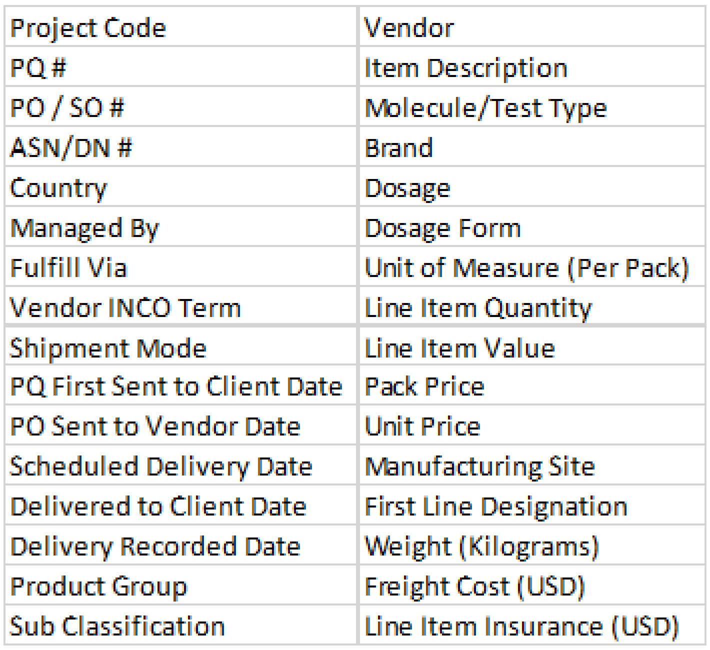
Our goal is to create a prototype blockchain for a supply chain application. We utilize a dataset of shipment of medical supplies. We have used some technical tools to analyze and understand the dataset first. These tools include workflow diagrams and Gantt charts, database. The dataset is stored in a flat file in csv format. Our first task was to convert that flat file into a traditional relational database, with the goal of comparing the operations of using a traditional database design and a blockchain design.

These workflow diagrams were primarily used to create a database schema that would form the basis of the traditional database we would be working on in Google BigQuery and understanding the data functionality for its use during the blockchain implementation. We also used a workflow diagram better to understand the use and function of our Blockchain and the approach we will be taking.

We also relied on Excel to help us clean up and organize the preliminary data before implementation in both database systems for BigQuery and our Blockchain. As for BigQuery, we are expected to be using SQL code to create the back-end of the database, tables, relational data, and functionalities. For our Blockchain, we will be approaching it in the Python and Java coding languages.

3 Supply Chain Dataset

The records the dataset is of medical supplies deliveries to different countries. This supply chain dataset contains 10,325 records and 33 attributes, which provide information about the product, suppliers, shipping, and price. Table 1 demonstrates the attributes of this dataset. our first task is to clean up the dataset by adding headers, converting the flat file into multiple tables, among other activities.



Project Code	Vendor
PQ #	Item Description
PO / SO #	Molecule/Test Type
ASN/DN #	Brand
Country	Dosage
Managed By	Dosage Form
Fulfill Via	Unit of Measure (Per Pack)
Vendor INCO Term	Line Item Quantity
Shipment Mode	Line Item Value
PQ First Sent to Client Date	Pack Price
PO Sent to Vendor Date	Unit Price
Scheduled Delivery Date	Manufacturing Site
Delivered to Client Date	First Line Designation
Delivery Recorded Date	Weight (Kilograms)
Product Group	Freight Cost (USD)
Sub Classification	Line Item Insurance (USD)

Table 1. attributes of the supply chain dataset

Data Manipulation using Google Bigquery

A traditional database is built here to demonstrate the process of building a traditional database and its function, with the goal of comparing a traditional database with the blockchain database in terms of safety, speed, transparency based on their construction process and functions. We create the traditional database in Google Bigquery (GBQ), an online platform that supports querying using ANSI SQL. Figure 1 demonstrates the process of data manipulation in GBQ. First, we transfer the data sources from Google Sheets into Google Bigquery. Based on the data attributes, we determined the design for the traditional database and developed an entity-relationship diagram (ERD) for the supply chain database shown in Figure 2.

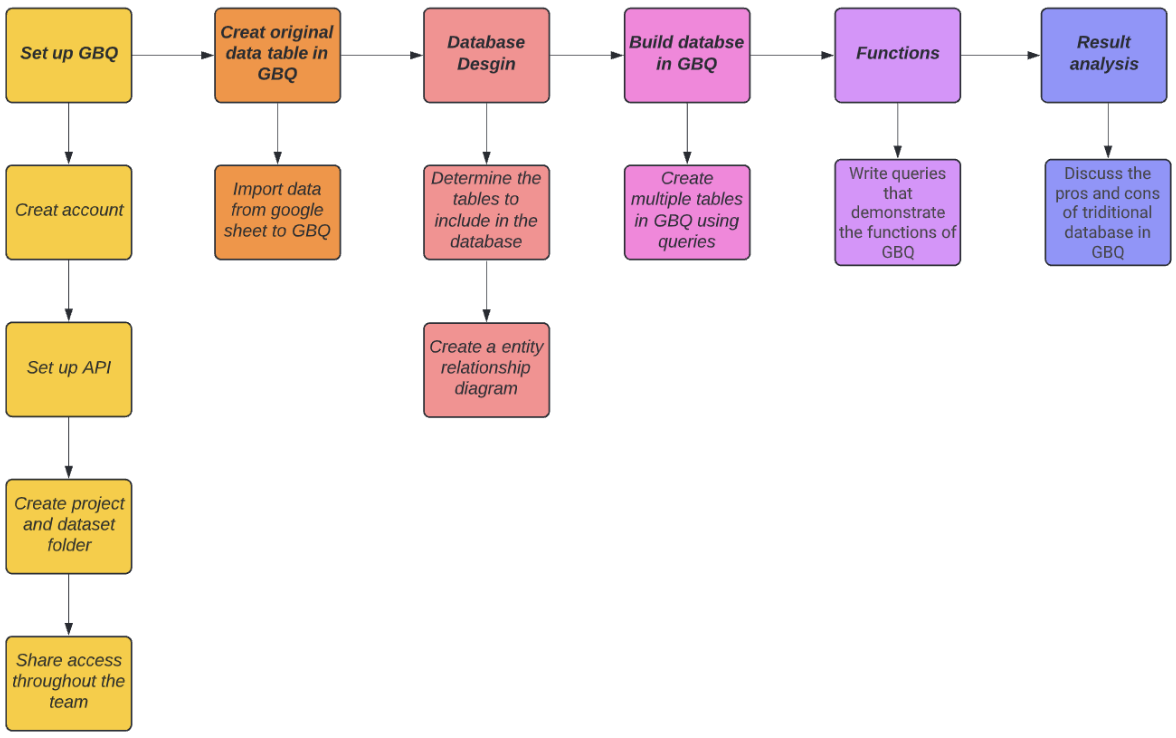

Set up GBQ
Creat original data table in GBQ
Database Desgin
Build databse in GBQ
Functions
Result analysis
Creat account
Import data from google sheet to GBQ
Determine the tables to include in the database
Create multiple tables in GBQ using queries
Write queries that demonstrate the functions of GBQ
Discuss the pros and cons of tiditional database in GBQ
Set up API
Create a entity relationship diagram
Create project and dataset folder
Share access throughout the team

Figure 1: Flowchart of Data Manipulation in GBQ

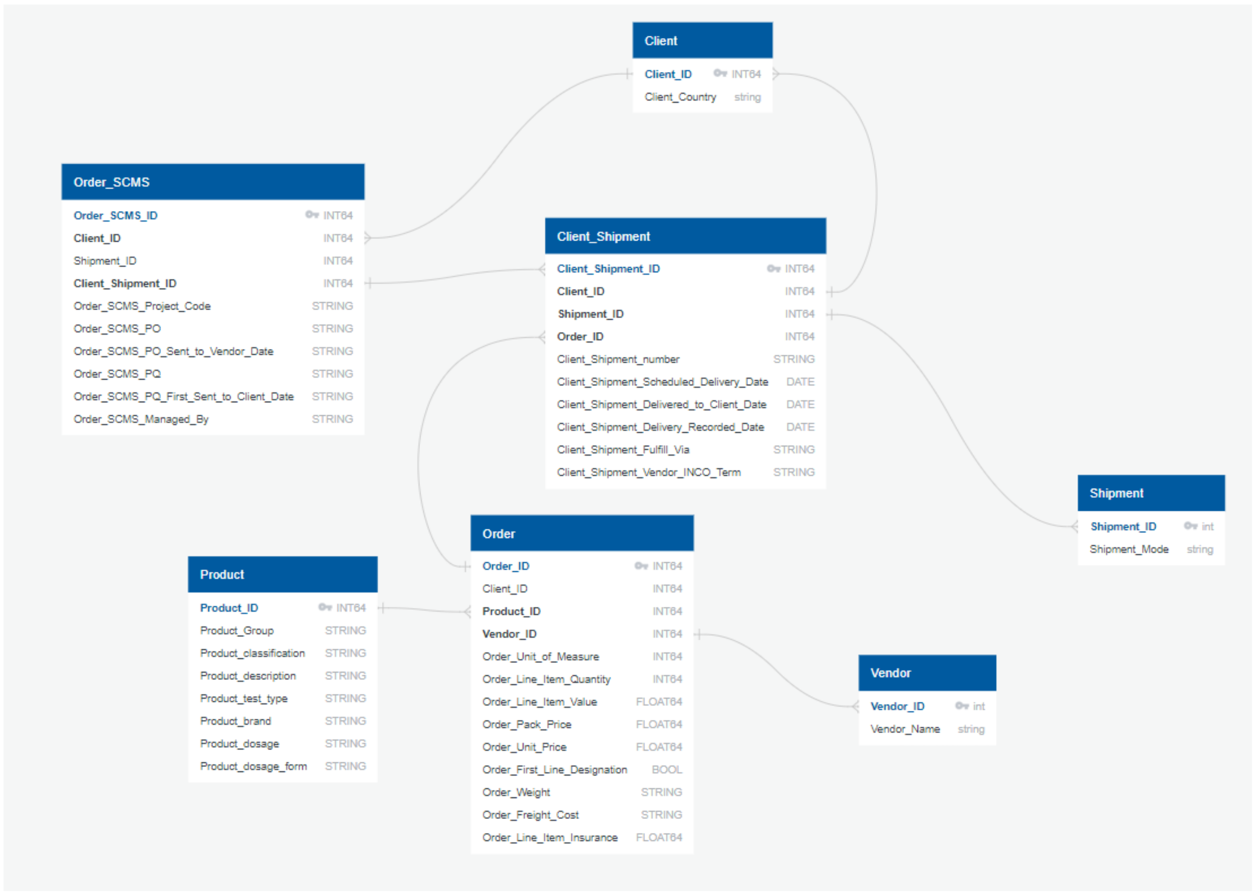

Client
Client_ID INT64
Client_Country string
Order_SCMS
Order_SCMS_ID INT64
Client_ID INT64
Shipment_ID INT64
Client_Shipment_ID INT64
Order_SCMS_Project_Code STRING
Order_SCMS_PO STRING
Order_SCMS_PO_Sent_to_Vendor_Date STRING
Order_SCMS_PQ STRING
Order_SCMS_PQ_First_Sent_to_Client_Date STRING
Order_SCMS_Managed_By STRING
Client_Shipment
Client_Shipment_ID INT64
Client_ID INT64
Shipment_ID INT64
Order_ID INT64
Client_Shipment_number STRING
Client_Shipment_Scheduled_Delivery_Date DATE
Client_Shipment_Delivered_to_Client_Date DATE
Client_Shipment_Delivery_Recorded_Date DATE
Client_Shipment_Fulfill_Via STRING
Client_Shipment_Vendor_INCO_Term STRING
Shipment
Shipment_ID int
Shipment_Mode string
Product
Product_ID INT64
Product_Group STRING
Product_classification STRING
Product_description STRING
Product_test_type STRING
Product_brand STRING
Product_dosage STRING
Product_dosage_form STRING
Order
Order_ID INT64
Client_ID INT64
Product_ID INT64
Vendor_ID INT64
Order_Unit_of_Measure INT64
Order_Line_Item_Quantity INT64
Order_Line_Item_Value FLOAT64
Order_Pack_Price FLOAT64
Order_Unit_Price FLOAT64
Order_First_Line_Designation BOOL
Order_Weight STRING
Order_Freight_Cost STRING
Order_Line_Item_Insurance FLOAT64
Vendor
Vendor_ID int
Vendor_Name string

Figure 2: Entity-relationship Diagram

Based on the entity-relationship diagram, several tables are created from the original tables in Google Bigquery using queries. The team created the database tables in GBQ using the following queries.

Create Original_DatabaseTable From Main Table:

```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Origional_Database`
AS
SELECT
   Product_Group
   ,Vendor
   ,Shipment_Mode
   ,Sub_Classification
   ,Item_Description
   ,Molecule_Test_Type
   ,Brand
   ,Manufacturing_Site
   ,Dosage
   ,Dosage_Form
   ,Pack_Price
   ,Unit_Price
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
GROUP BY 1,2,3,4,5,6,7,8,9,10,11,12
ORDER BY 1,2,3,4,5,6,7,8,9,10,11,12
;
```

Create Client Table:

```
CREATE OR REPLACE TABLE  `directed-method-338916.Capstone_Team_4_development.Client`
AS
SELECT
   b.Country AS Client_Country
   ,ROW_NUMBER() OVER() AS Client_ID
FROM UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
   ,`directed-method-338916.Capstone_Team_4_development.Origional_Database`  b
GROUP BY 1
;
```

Create Vendor Table:

```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Vendor`
AS
SELECT
   b.Vendor AS Vendor_Name
   ,ROW_NUMBER() OVER() AS Vendor_ID
FROM UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
   ,`directed-method-338916.Capstone_Team_4_development.Origional_Database`  b
GROUP BY 1
;
```

Create Shipment Table:

```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Shipment`
AS
SELECT
   b.Shipment_Mode AS Shipment_Mode
   ,ROW_NUMBER() OVER() AS Shipment_ID
FROM UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
   ,`directed-method-338916.Capstone_Team_4_development.Origional_Database`  b
GROUP BY 1
;
```

Create Product Table:

```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Product`
AS
SELECT
```

```
    b.Product_Group AS Product_Group
    ,b.Sub_Classification AS Product_classification
    ,b.Item_Description AS Product_description
    ,Molecule_Test_Type AS Product_test_type
    ,Brand AS Product_brand
    ,Dosage AS Product_dosage
    ,Dosage_Form AS Product_dosage_form
    ,ROW_NUMBER() OVER() AS Product_ID
FROM UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
    ,`directed-method-338916.Capstone_Team_4_development.Origional_Database`  b
GROUP BY 1,2,3,4,5,6,7
;
```

Create Product_Order Table:

```
CREATE OR REPLACE TABLE  `directed-method-338916.Capstone_Team_4_development.Order`
AS
SELECT
    b.Client_ID AS Client_ID
    ,c.Product_ID AS Product_ID
    ,e.Vendor_ID AS Vendor_ID
    ,d.Unit_of_Measure__Per_Pack_ AS Order_Unit_of_Measure
    ,d.Line_Item_Quantity AS Order_Line_Item_Quantity
    ,d.Line_Item_Value AS Order_Line_Item_Value
    ,d.Pack_Price AS Order_Pack_Price
    ,d.Unit_Price AS Order_Unit_Price
    ,d.First_Line_Designation AS Order_First_Line_Designation
    ,d.Weight__Kilograms_ AS Order_Weight
    ,d.Freight_Cost__USD_ AS Order_Freight_Cost
    ,d.Line_Item_Insurance__USD_ AS Order_Line_Item_Insurance
    ,ROW_NUMBER() OVER() AS Order_ID
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`  d
,UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Client`  b
    ON b.Client_Country=d.Country
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Product`  c
    ON d.Product_Group = c.Product_Group
    AND d.Sub_Classification = c.Product_classification
    AND d.Item_Description = c.Product_description
    AND d.Molecule_Test_Type = c.Product_test_type
    AND d.Brand = c.Product_brand
    AND d.Dosage = c.Product_dosage
    AND d.Dosage_Form = c.Product_dosage_form
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Vendor`  e
    ON d.Vendor=e.Vendor_Name
GROUP BY 1,2,3,4,5,6,7,8,9,10,11,12
;
```

Create Client_Shipment Table:

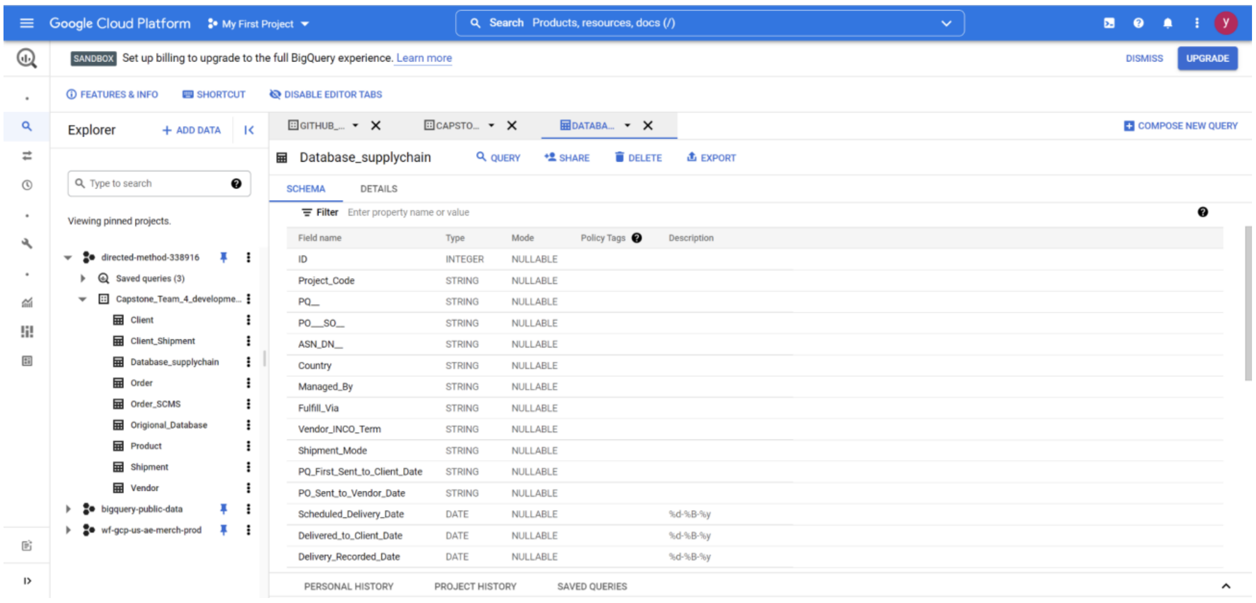
```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Client_Shipment`
AS
SELECT
    b.Client_ID AS Client_ID
    ,c.Order_ID AS Order_ID
    ,e.Shipment_ID AS Shipment_ID
    ,d.ASN_DN__ AS Client_Shipment_number
    ,d.Scheduled_Delivery_Date AS Client_Shipment_Scheduled_Delivery_Date
    ,d.Delivered_to_Client_Date AS Client_Shipment_Delivered_to_Client_Date
    ,d.Delivery_Recorded_Date AS Client_Shipment_Delivery_Recorded_Date
    ,d.Fulfill_Via AS Client_Shipment_Fulfill_Via
```

```
   ,d.Vendor_INCO_Term AS Client_Shipment_Vendor_INCO_Term
   ,ROW_NUMBER() OVER() AS Client_Shipment_ID
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`  d
,UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Client`  b
   ON b.Client_Country=d.Country
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Order`  c
   ON d.Unit_of_Measure__Per_Pack_ = Order_Unit_of_Measure
   AND d.Line_Item_Quantity = Order_Line_Item_Quantity
   AND d.Line_Item_Value = Order_Line_Item_Value
   AND d.Pack_Price = Order_Pack_Price
   AND d.Unit_Price = Order_Unit_Price
   AND d.First_Line_Designation = Order_First_Line_Designation
   AND d.Weight__Kilograms_ = Order_Weight
   AND d.Freight_Cost__USD_ = Order_Freight_Cost
   AND d.Line_Item_Insurance__USD_ = Order_Line_Item_Insurance
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Shipment`  e
   ON d.Shipment_Mode=e.Shipment_Mode
GROUP BY 1,2,3,4,5,6,7,8,9
;
```

Create Order_SCMS Table:

```
CREATE OR REPLACE TABLE  `directed-method-
338916.Capstone_Team_4_development.Order_SCMS`
AS
SELECT
   b.Client_ID
   ,e.Shipment_ID
   ,c.Client_Shipment_ID
   ,d.Project_Code AS Order_SCMS_Project_Code
   ,d.PO___SO__ AS Order_SCMS_PO
   ,d.PO_Sent_to_Vendor_Date AS Order_SCMS_PO_Sent_to_Vendor_Date
   ,d.PQ__ AS Order_SCMS_PQ
   ,d.PQ_First_Sent_to_Client_Date AS Order_SCMS_PQ_First_Sent_to_Client_Date
   ,d.Managed_By AS Order_SCMS_Managed_By
   ,ROW_NUMBER() OVER() AS Order_SCMS_ID
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`  d
,UNNEST((SELECT SPLIT(FORMAT("%20000s", ""),'') AS h FROM (SELECT NULL))) AS pos
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Client`  b
   ON b.Client_Country=d.Country
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Shipment`  e
   ON e.Shipment_Mode=d.Shipment_Mode
LEFT JOIN `directed-method-338916.Capstone_Team_4_development.Client_Shipment`  c
   ON d.ASN_DN__ = c.Client_Shipment_number
   AND d.Scheduled_Delivery_Date = c.Client_Shipment_Scheduled_Delivery_Date
   AND d.Delivered_to_Client_Date = c.Client_Shipment_Delivered_to_Client_Date
   AND d.Delivery_Recorded_Date = c.Client_Shipment_Delivery_Recorded_Date
   AND d.Fulfill_Via = c.Client_Shipment_Fulfill_Via
   AND d.Vendor_INCO_Term = c.Client_Shipment_Vendor_INCO_Term
GROUP BY 1,2,3,4,5,6,7,8,9
;
```

Figure 3 shows the database tables schema created in Google Cloud for the supply chain data



Field name	Type	Mode	Policy Tags	Description
ID	INTEGER	NULLABLE		
Project_Code	STRING	NULLABLE		
PQ__	STRING	NULLABLE		
PO__SO__	STRING	NULLABLE		
ASN_DN__	STRING	NULLABLE		
Country	STRING	NULLABLE		
Managed_By	STRING	NULLABLE		
Fulfill_Via	STRING	NULLABLE		
Vendor_INCO_Term	STRING	NULLABLE		
Shipment_Mode	STRING	NULLABLE		
PQ_First_Sent_to_Client_Date	STRING	NULLABLE		
PO_Sent_to_Vendor_Date	STRING	NULLABLE		
Scheduled_Delivery_Date	DATE	NULLABLE		%d-%B-%y
Delivered_to_Client_Date	DATE	NULLABLE		%d-%B-%y
Delivery_Recorded_Date	DATE	NULLABLE		%d-%B-%y

Figure 3: Screenshot of Database in Google Bigquery

4 Data Analysis using Google Bigquery

After the whole traditional database was built in GBQ, the team wrote several queries to demonstrate the functions of GBQ, such as retrieving data, changing data, sorting the data, and calculating the data.

GBQ Database Queries for Analytics

```
--Top 10 Countries
SELECT
   Country
   ,COUNT(Country) AS Country_COUNT
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

--Top 10 Manufacturing_Site
SELECT
   Manufacturing_Site
   ,COUNT(Manufacturing_Site) AS Manufacturing_Site_COUNT
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

--Total Pack Price for Top 15 Countries
SELECT
   Country
   ,SUM(Pack_Price) AS Total_Pack_Price
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
GROUP BY 1
ORDER BY 2 DESC
LIMIT 15
;

--First Line Designation
SELECT
   COUNT(
      CASE
```

```
            WHEN First_Line_Designation IS TRUE THEN  First_Line_Designation
            ELSE NULL
         END
    ) AS IsFirstLineDesgination
    ,COUNT(
         CASE
            WHEN First_Line_Designation IS NOT TRUE THEN  First_Line_Designation
            ELSE NULL
         END
    ) AS IsNotFirstLineDesgination
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
;

--Shipment Mode Percentage
SELECT
    Shipment_Mode
    ,SAFE_DIVIDE(Shipment_count,Total)*100 as Pencentage
FROM (
    SELECT
        Shipment_Mode
        ,COUNT(Shipment_Mode) AS Shipment_count
        ,SUM(COUNT(Shipment_Mode)) OVER() AS Total
    FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
    GROUP by Shipment_Mode
)
ORDER BY 2 DESC
;

--Shipment Mode and Pack Price
SELECT
    Shipment_Mode
    ,AVG(Pack_Price) AS Average_Pack_Price
FROM `directed-method-338916.Capstone_Team_4_development.Origional_Database`
GROUP BY 1
ORDER BY 2 DESC
;
```

After going through all the building processes and testing the functions of GBQ, the team concluded the pros and cons of a traditional database in GBQ. The advantages of a traditional database in GBQ are that it is easy to build and update the tables. People can do some calculations and data analysis inside the database using queries. However, the traditional database has many safety, transparency, and speed shortcomings. For example, people cannot track and find the people who made the changes to the data.

Moreover, joining multiple tables together will lead to low running speed. In conclusion, the traditional database in GBQ has a good performance in data manipulation and data analysis. However, like all the traditional databases, the GBQ database has data security and transparency limitations.

Data Visualization

The data visualization explores the relationship between countries, shipment methods, manufacturing sites, and pack price. Data analysis aims to get familiar with the supply chain dataset.

There are 15 countries as the shipping destinations in this dataset. Figure 4 is the map graph showing the top 15 countries with the maximum total pack price. From Figure 4, most of the products are shipped to southern Africa. On the other hand, the manufacturing sites and shipping origins are primarily located in Europe and Asia. Figure 5 shows the top 10 manufacturing sites with the most cases. Aurobindo Unit III in India produces the most products. Moreover, there are four mean shipment methods, and air has the highest mean pack price, according to Figure 6.

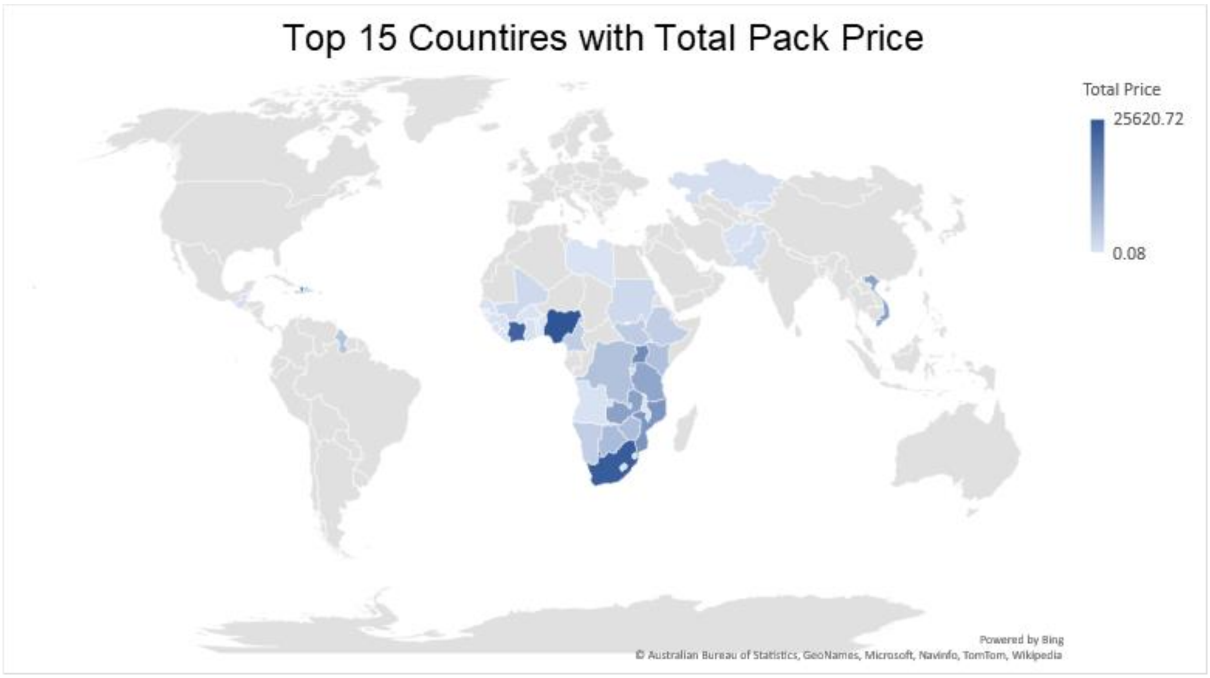

Top 15 Countires with Total Pack Price
Total Price
25620.72
0.08
Powered by Bing
© Australian Bureau of Statistics, GeoNames, Microsoft, Navinfo, TomTom, Wikipedia

Figure 4: Top 15 counties with total pack price

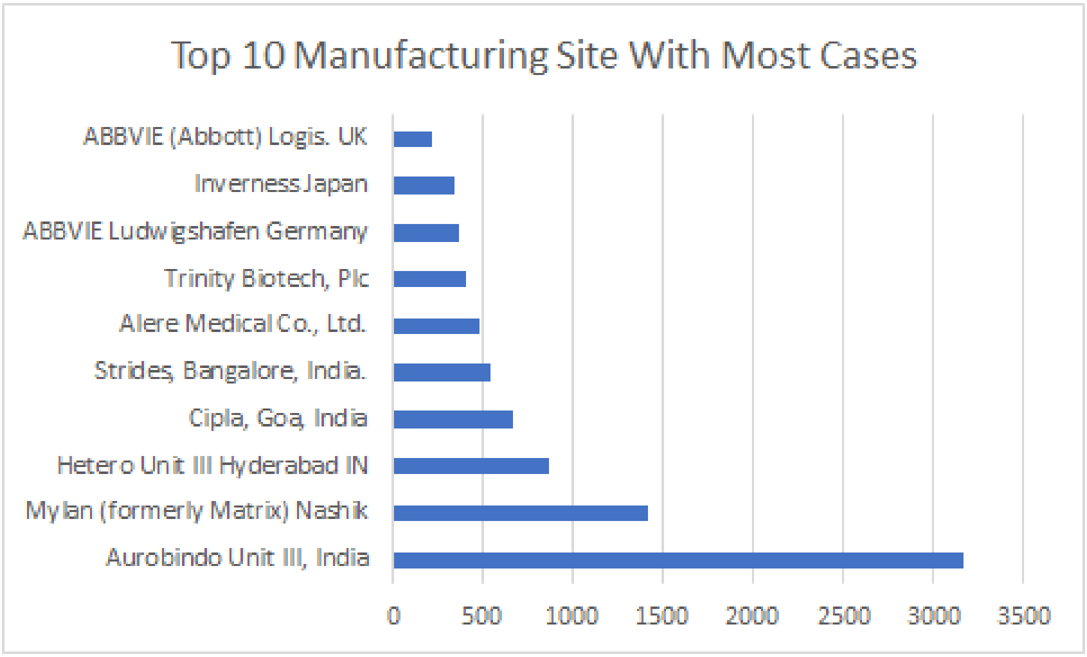


Figure 5: Top 10 manufacturing sites with most cases

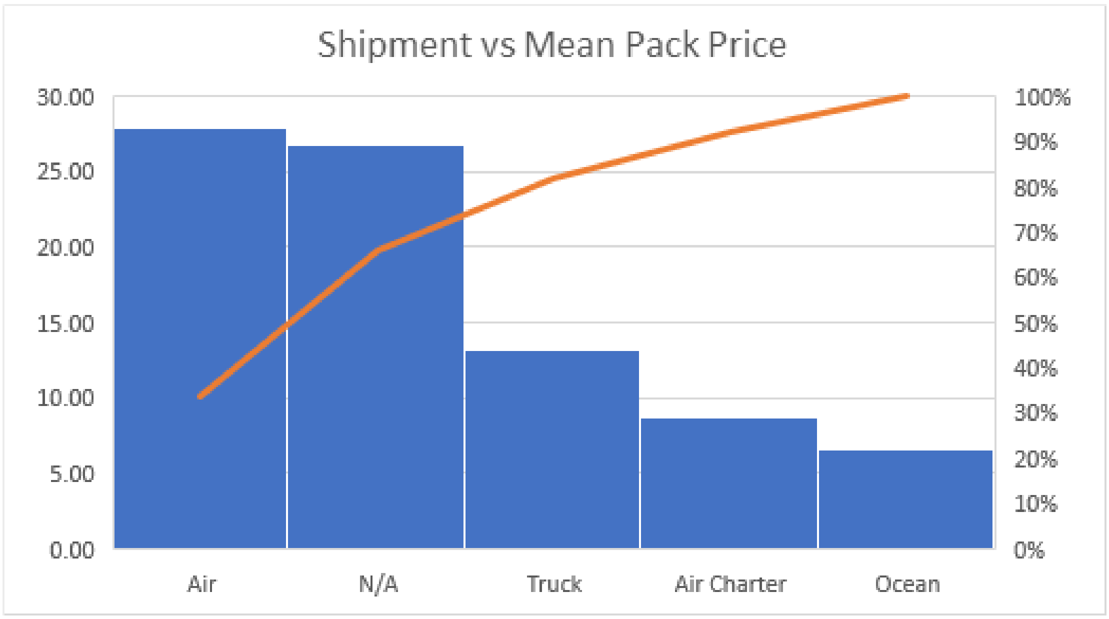


Figure 6: Shipment vs. Mean Pack Price

IV. Developing the Blockchain Model

1. Creating the Model

After completing Capstone 1, the team investigated a few different Blockchain platforms to see which would be best for our project. We first looked at a private Blockchain solution, as businesses often do not want to keep confidential data on a public Blockchain like Ethereum. During our research, we found Hyperledger Fabric, an IBM-created platform [10]. As a permissioned Blockchain network, Hyperledger can solve many of our project's problems. It also offers the facility to create a Blockchain application maintaining the privacy of the organization's information. Hyperledger is coded primarily on Java or Golang languages. Hyperledger has strict control over accessibility for users; only people granted access can create new Blocks or see the data, which means it is highly secure. Walmart ran a "proof of concept" project on the Hyperledger system to track the origin of mangos in stores. The result of the experiment was the time needed to trace their provenance going from 7 days to 2.2 seconds! [10]. Our team focused on Hyperledger for a few weeks until meeting with Professor Sarathy during one of our Design Reviews. He advised us to look into Tradelens, a Blockchain application built on IBM Hyperledger. Tradelens is an interconnected ecosystem of supply chain partners - cargo owners, ocean and inland carriers, freight forwarders and logistics providers, ports and terminals, customs authorities, etc. [11]. The focus of Tradelens is on digitizing information and exchange between the different parts of a supply chain. The goal is to access a supply chain and logistics network with over half of the coverage of global container-shipping-volume data. It runs on matrix and Blockchain, ensuring every party to a shipment has access only to their information and a secure audit trail of transactions. It is a global platform built on open-source technology and publicly available APIs.

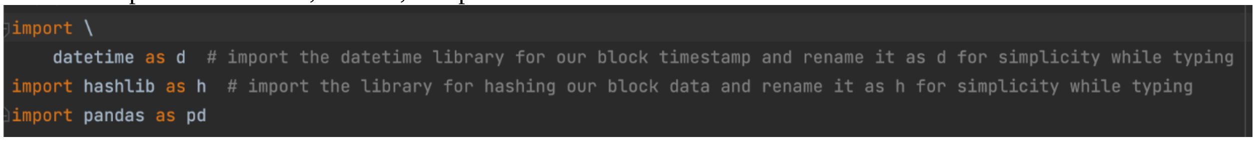
After our research, we decided that to best compare traditional supply chain methods and results (GBQ) to a Blockchain ledger, we should create our own basic Blockchain using Python as the back end and a graphical user interface that demonstrates how the Blockchain works using the Go programming language. The team created this Blockchain using open-source code and original code to best satisfy this project's needs.

This section will present how we developed our Blockchain, the creation and initialization of the 'Block' class, and other significant functions. Each Block that the team created contains five separate variables. These variables are 1. the ID, 2. the time of creation, 3. the hash of the Block, 4. the hash of the previous Block, and 5. the transaction data. To get a realistic representation of Blockchain data, we imported three separate third-party libraries. The first library is DateTime, which records the current date and time. The second library is Hashlib, which creates a hash for each Block and encrypts the transaction data. Lastly, pandas was imported to read and write from the supply chain transactions Excel file. The team had to set the value of each Block the team would create from here. The timestamp variable is set to the exact time of creation. The hash for each Block is created by a hash Block function, which uses Hashlib to encrypt the Block's content. For the data variable, the team imported the Excel file of supply chain transactions and created a function in the program to read the transaction values by row. The team then set all the Excel cell values to string variables within the program and concatenated them to create a readable output for the user. After selecting the data variable of the upcoming Block to the current row of Excel file values read, the program would then create the Block. Then, the team increased the last read Excel file row index by one and repeated the process for the next Block.

Below, we'll detail the steps of the coding process along with screenshots.

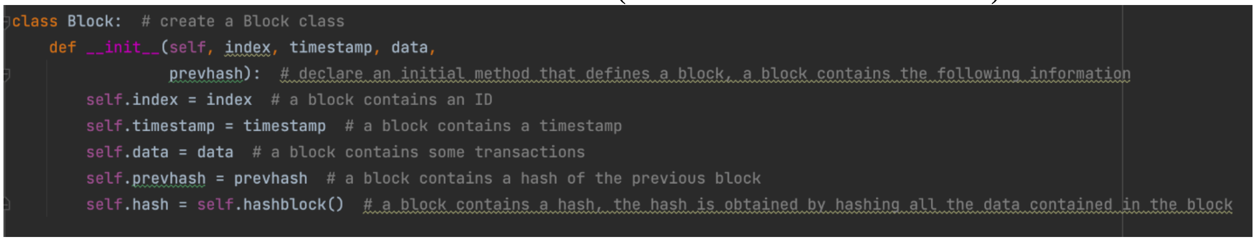
Steps to build our blockchain backend:

1. Import the datetime, hashlib, and pandas libraries



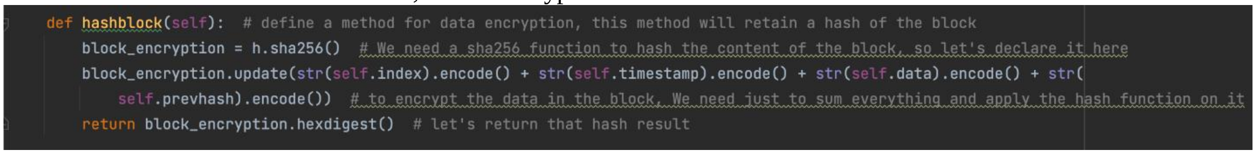
```
import \
    datetime as d  # import the datetime library for our block timestamp and rename it as d for simplicity while typing
import hashlib as h  # import the library for hashing our block data and rename it as h for simplicity while typing
import pandas as pd
```

2. Create the Block class and initialization method (which each block created refers to)



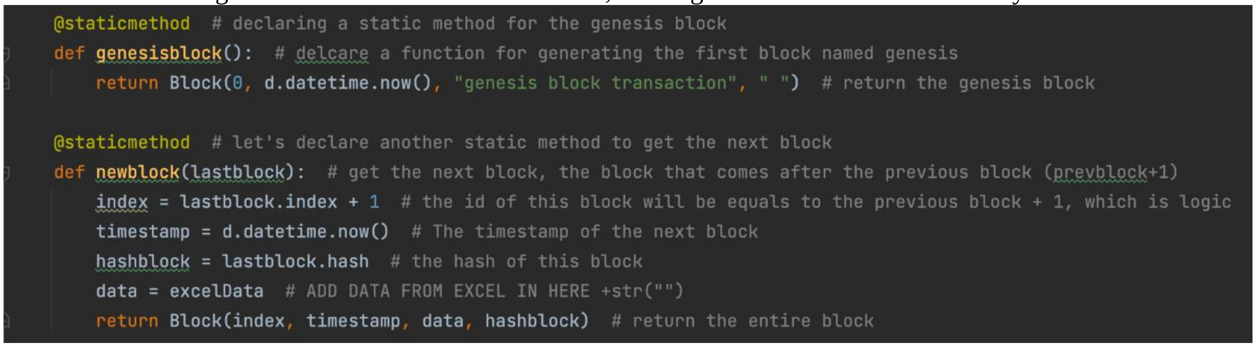
```
class Block:  # create a Block class
    def __init__(self, index, timestamp, data,
                 prevhash):  # declare an initial method that defines a block, a block contains the following information
        self.index = index  # a block contains an ID
        self.timestamp = timestamp  # a block contains a timestamp
        self.data = data  # a block contains some transactions
        self.prevhash = prevhash  # a block contains a hash of the previous block
        self.hash = self.hashblock()  # a block contains a hash, the hash is obtained by hashing all the data contained in the block
```

3. Define the hashblock method, which encrypts the data in each block



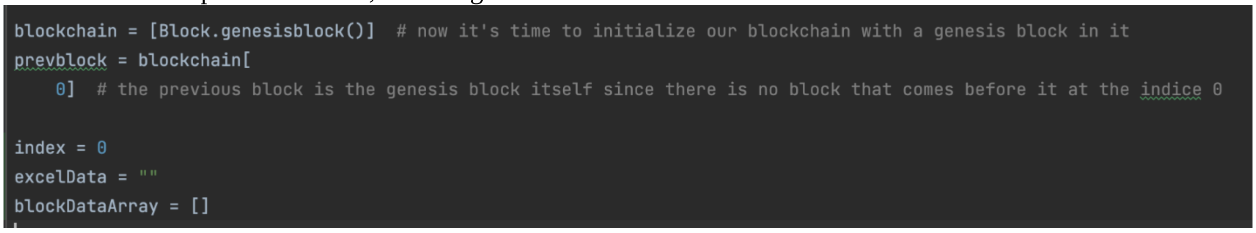
```
    def hashblock(self):  # define a method for data encryption, this method will retain a hash of the block
        block_encryption = h.sha256()  # We need a sha256 function to hash the content of the block, so let's declare it here
        block_encryption.update(str(self.index).encode() + str(self.timestamp).encode() + str(self.data).encode() + str(
            self.prevhash).encode())  # to encrypt the data in the block, We need just to sum everything and apply the hash function on it
        return block_encryption.hexdigest()  # let's return that hash result
```

4. Create the genesisblock and newblock methods, which generate the first block and any other blocks



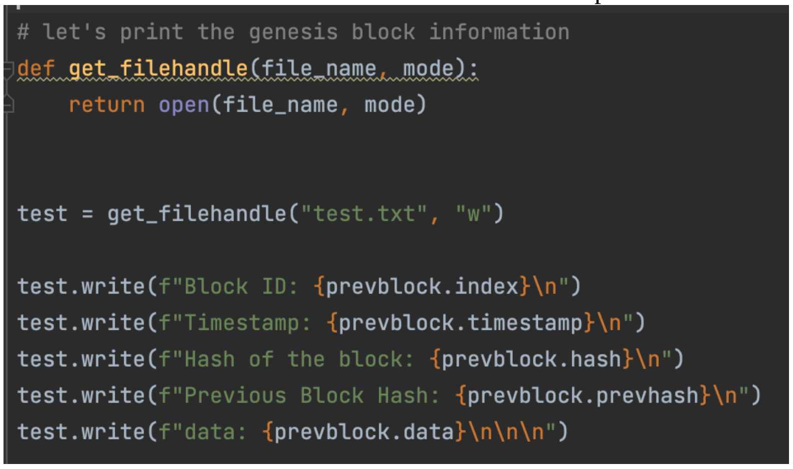
```
    @staticmethod  # declaring a static method for the genesis block
    def genesisblock():  # delcare a function for generating the first block named genesis
        return Block(0, d.datetime.now(), "genesis block transaction", " ")  # return the genesis block

    @staticmethod  # let's declare another static method to get the next block
    def newblock(lastblock):  # get the next block, the block that comes after the previous block (prevblock+1)
        index = lastblock.index + 1  # the id of this block will be equals to the previous block + 1, which is logic
        timestamp = d.datetime.now()  # The timestamp of the next block
        hashblock = lastblock.hash  # the hash of this block
        data = excelData  # ADD DATA FROM EXCEL IN HERE +str("")
        return Block(index, timestamp, data, hashblock)  # return the entire block
```

5. Define separate variables, including the blockchain that we will create



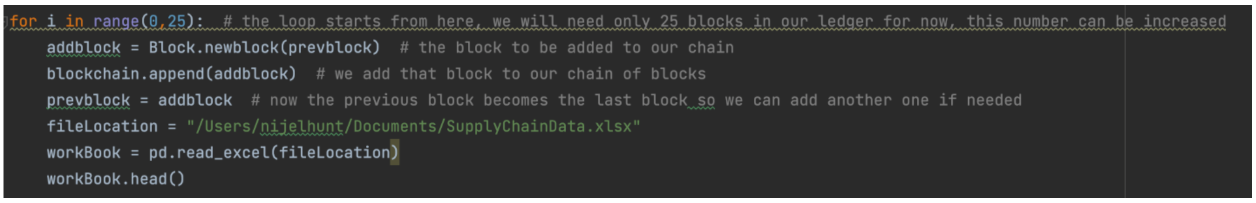
```
blockchain = [Block.genesisblock()]  # now it's time to initialize our blockchain with a genesis block in it
prevblock = blockchain[
    0]  # the previous block is the genesis block itself since there is no block that comes before it at the indice 0

index = 0
excelData = ""
blockDataArray = []
```

6. Define a method to create a .txt file for the output of the blockchain



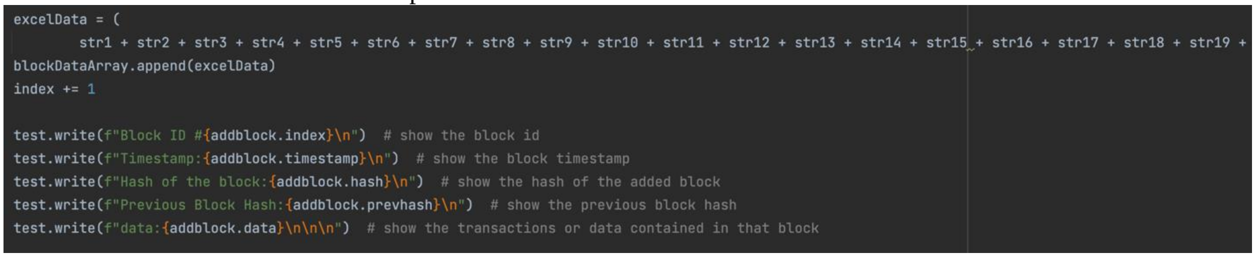
```python
# let's print the genesis block information
def get_filehandle(file_name, mode):
    return open(file_name, mode)


test = get_filehandle("test.txt", "w")

test.write(f"Block ID: {prevblock.index}\n")
test.write(f"Timestamp: {prevblock.timestamp}\n")
test.write(f"Hash of the block: {prevblock.hash}\n")
test.write(f"Previous Block Hash: {prevblock.prevhash}\n")
test.write(f"data: {prevblock.data}\n\n\n")
```

7. Create a for loop which loops through as many Excel rows as needed and creates blocks for each row of data



```python
for i in range(0,25):  # the loop starts from here, we will need only 25 blocks in our ledger for now, this number can be increased
    addblock = Block.newblock(prevblock)  # the block to be added to our chain
    blockchain.append(addblock)  # we add that block to our chain of blocks
    prevblock = addblock  # now the previous block becomes the last block so we can add another one if needed
    fileLocation = "/Users/nijelhunt/Documents/SupplyChainData.xlsx"
    workBook = pd.read_excel(fileLocation)
    workBook.head()
```

8. This is the end of the for loop, where the index in which it reads the Excel file row increases by one, and the value of each block is written/printed



```python
excelData = (
        str1 + str2 + str3 + str4 + str5 + str6 + str7 + str8 + str9 + str10 + str11 + str12 + str13 + str14 + str15 + str16 + str17 + str18 + str19 +
blockDataArray.append(excelData)
index += 1

test.write(f"Block ID #{addblock.index}\n")  # show the block id
test.write(f"Timestamp:{addblock.timestamp}\n")  # show the block timestamp
test.write(f"Hash of the block:{addblock.hash}\n")  # show the hash of the added block
test.write(f"Previous Block Hash:{addblock.prevhash}\n")  # show the previous block hash
test.write(f"data:{addblock.data}\n\n\n")  # show the transactions or data contained in that block
```

11. Implementing the Blockchain Model

A user interface has been built to demonstrate how the user interacts with the Blockchain database. After conducting several types of research online, we have decided to build the user interface based on an open-source code called "one-year-old baby blockchain". The backend of the user interface was built using Go, a programming language commonly used for Blockchain. It has better speed, endurance, and security performance than other programming languages like Python and JavaScript. And the frontend was built using JSON, a programming language that is commonly used in building the frontend. Figure 7 is the high-level view of the user interface. User data is sent from the Webpage to the backend. Then, the backend will generate the Blocks and send the Block information, including block ID, hash, previous hash, and timestamp, to the frontend. Finally, the frontend will display a new block on the Webpage.

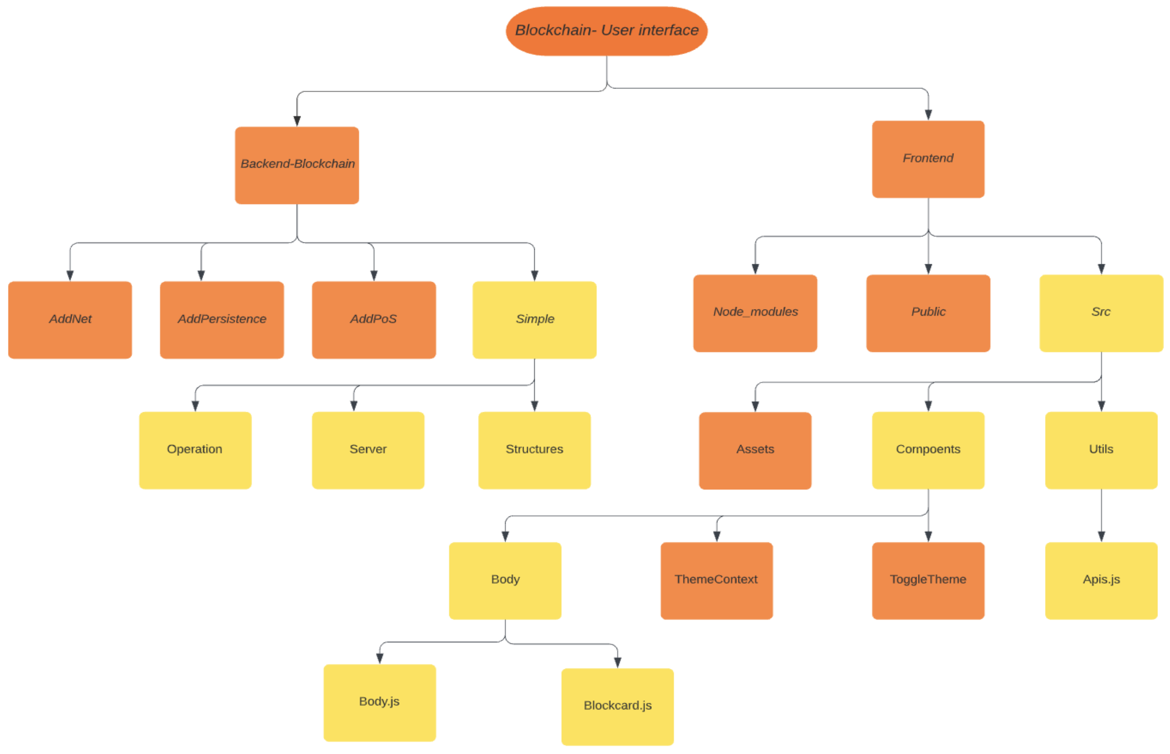


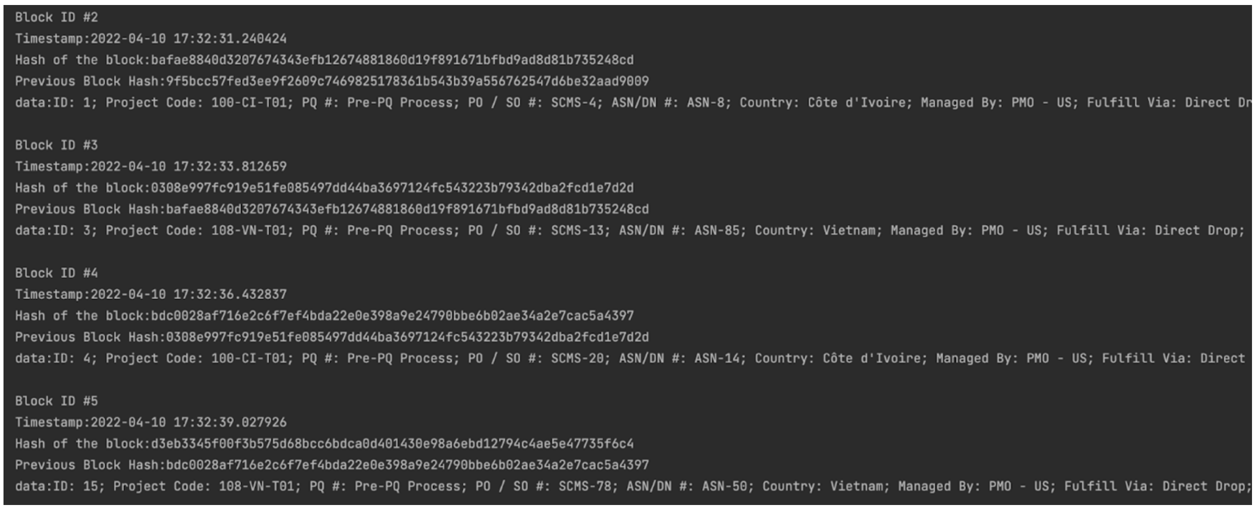
Figure 7: Frontend code structure

Our back end was built in Python using a combination of original code and open-source code. The open-source code we used focused on encryption and creating the blockchain. We added other capabilities to meet our requirements, such as a search function and reading from an Excel file with over 10,000 rows of transactions. Ledger.py is the python file that runs the program and has all the code in our project. Test.txt is a file that is created once the program is done running with the output of the blockchain.

Based on the open-source code, the team edited the files and folders, marked in yellow and shown in Figure 7. The backend was built using Go, software commonly used for Blockchain. It has better speed, endurance, and security performance than other programming languages like Python and JavaScript. Operation go, server.go and structures.go are the three primary files in the back-end. Operations.go generates blocks by creating a hash based on the previous hash. Server.go connects the backend to the frontend. Structures.go creates the Blockchain data structure. The front-end is built using the JSON programming language. The local location for the page is identified in Apis.js. Body.js reads the page inputs, connects to the backend to generate new Blocks, and sends the Block information and inputs to Blockcard.js. Then, Blockcard.js will create and display a Block card with the Block's order information.

Testing the Blockchain Model

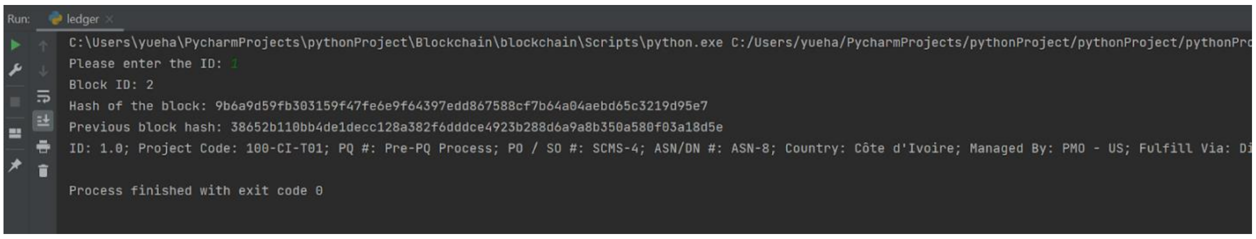
By following steps 1-8 in the Developing the Blockchain Model section, anyone can successfully run our project. The program's output, once run, is shown below.



```
Block ID #2
Timestamp:2022-04-10 17:32:31.240424
Hash of the block:bafae8840d3207674343efb12674881860d19f891671bfbd9ad8d81b735248cd
Previous Block Hash:9f5bcc57fed3ee9f2609c7469825178361b543b39a556762547d6be32aad9009
data:ID: 1; Project Code: 100-CI-T01; PQ #: Pre-PQ Process; PO / SO #: SCMS-4; ASN/DN #: ASN-8; Country: Côte d'Ivoire; Managed By: PMO - US; Fulfill Via: Direct Dr

Block ID #3
Timestamp:2022-04-10 17:32:33.812659
Hash of the block:0308e997fc919e51fe085497dd44ba3697124fc543223b79342dba2fcd1e7d2d
Previous Block Hash:bafae8840d3207674343efb12674881860d19f891671bfbd9ad8d81b735248cd
data:ID: 3; Project Code: 108-VN-T01; PQ #: Pre-PQ Process; PO / SO #: SCMS-13; ASN/DN #: ASN-85; Country: Vietnam; Managed By: PMO - US; Fulfill Via: Direct Drop;

Block ID #4
Timestamp:2022-04-10 17:32:36.432837
Hash of the block:bdc0028af716e2c6f7ef4bda22e0e398a9e24790bbe6b02ae34a2e7cac5a4397
Previous Block Hash:0308e997fc919e51fe085497dd44ba3697124fc543223b79342dba2fcd1e7d2d
data:ID: 4; Project Code: 100-CI-T01; PQ #: Pre-PQ Process; PO / SO #: SCMS-20; ASN/DN #: ASN-14; Country: Côte d'Ivoire; Managed By: PMO - US; Fulfill Via: Direct

Block ID #5
Timestamp:2022-04-10 17:32:39.027926
Hash of the block:d3eb3345f00f3b575d68bcc6bdca0d401430e98a6ebd12794c4ae5e47735f6c4
Previous Block Hash:bdc0028af716e2c6f7ef4bda22e0e398a9e24790bbe6b02ae34a2e7cac5a4397
data:ID: 15; Project Code: 108-VN-T01; PQ #: Pre-PQ Process; PO / SO #: SCMS-78; ASN/DN #: ASN-50; Country: Vietnam; Managed By: PMO - US; Fulfill Via: Direct Drop;
```

Figure 8: Blockchain back end output

Figure 8 shows a snippet of the output when the code runs. The program creates Blocks labeled in chronological order, starting from the genesis block, along with the value of the variables in each transaction. While it cannot be seen here, each Block took an average of around 4-5 seconds to be created. With added capabilities to the Blockchain, this creation time would decrease dramatically. After creating a working Blockchain, the team decided to add a search function, which can be seen below.



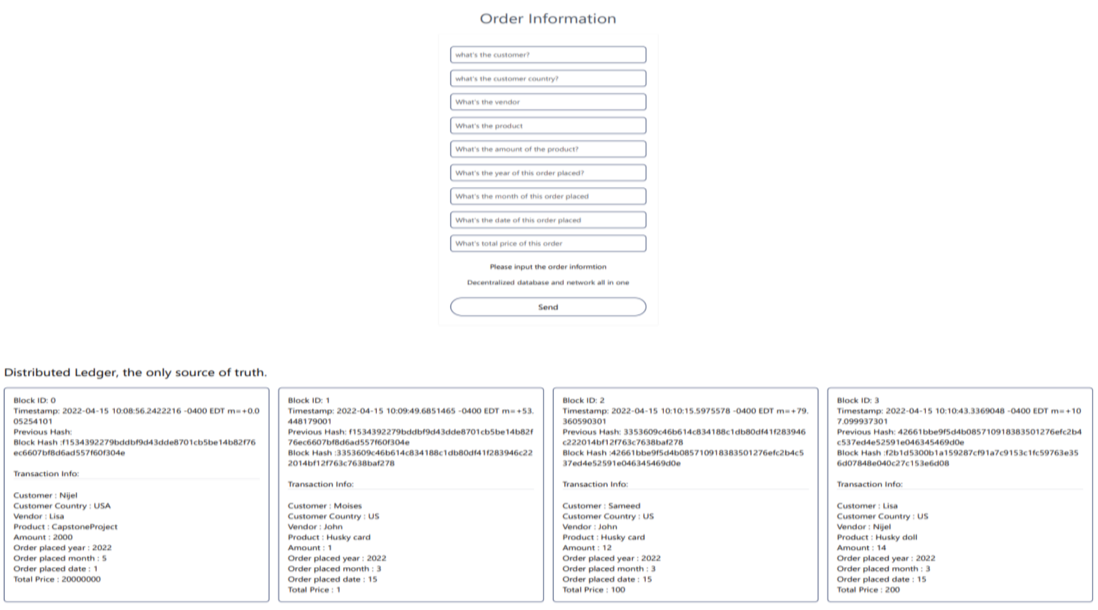
```
C:\Users\yueha\PycharmProjects\pythonProject\Blockchain\blockchain\Scripts\python.exe C:/Users/yueha/PycharmProjects/pythonProject/pythonProject/pythonPro
Please enter the ID: 1
Block ID: 2
Hash of the block: 9b6a9d59fb303159f47fe6e9f64397edd867588cf7b64a04aebd65c3219d95e7
Previous block hash: 38652b110bb4de1decc128a382f6dddce4923b288d6a9a8b350a580f03a18d5e
ID: 1.0; Project Code: 100-CI-T01; PQ #: Pre-PQ Process; PO / SO #: SCMS-4; ASN/DN #: ASN-8; Country: Côte d'Ivoire; Managed By: PMO - US; Fulfill Via: Di

Process finished with exit code 0
```

Figure 9: Code retrieval function for finding specific "Blocks"

Figure 9 shows the output of the program's data retrieval process. The program prompts the user for the Block's ID that they want information about in the image above. After inputting the ID of choice, the program prints out the details of the respective Block. After constructing the back end of the Blockchain database, the team developed a user interface for the front-end.

Creating blocks of supply chain

Figure 10 is a screenshot of the user interface used to create blocks. The user interface consists of an order information form that collects the inputs from the user. In this form, the user is asked to input the customer, vendor, product, date, etc. By clicking the "send" button, the information is sent to the "Blockchain." Once the user sends the information, the user interface will output a Block at the bottom of the page, containing the Block ID, hash, previous hash, and timestamp. The order information that the user just inputted is also displayed in the output Block. If the user tries to send the wrong data type or null data, the user interface will send a warning message on the page, as Figure 11 shows, and the wrong data will not be sent to the Blockchain backend.



Order Information

what's the customer?

what's the customer country?

What's the vendor

What's the product

What's the amount of the product?

What's the year of this order placed?

What's the month of this order placed

What's the date of this order placed

What's total price of this order

Please input the order informtion

Decentralized database and network all in one

Send

Distributed Ledger, the only source of truth.

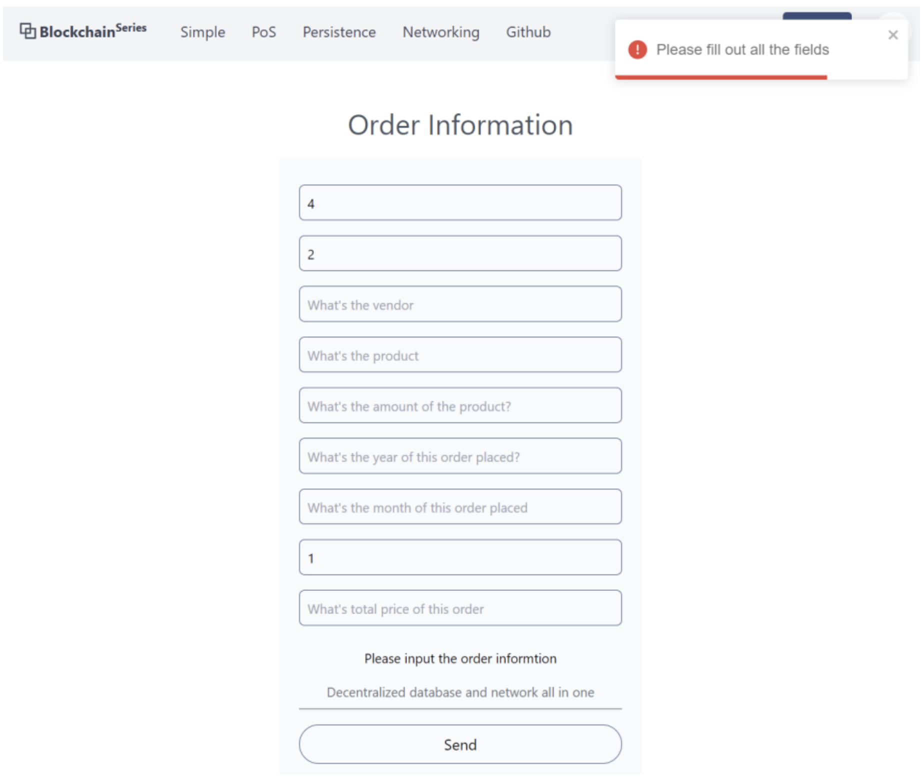
```
Block ID: 0
Timestamp: 2022-04-15 10:08:56.2422216 -0400 EDT m=+0.0
05254101
Previous Hash:
Block Hash :f1534392279bddbf9d43dde8701cb5be14b82f76
ec6607bf8d6ad557f60f304e

Transaction Info:

Customer : Nijel
Customer Country : USA
Vendor : Lisa
Product : CapstoneProject
Amount : 2000
Order placed year : 2022
Order placed month : 5
Order placed date : 1
Total Price : 20000000
```

```
Block ID: 1
Timestamp: 2022-04-15 10:09:49.6851465 -0400 EDT m=+53.
448179001
Previous Hash: f1534392279bddbf9d43dde8701cb5be14b82f
76ec6607bf8d6ad557f60f304e
Block Hash :3353609c46b614c834188c1db80df41f283946c22
2014bf12f763c7638baf278

Transaction Info:

Customer : Moises
Customer Country : US
Vendor : John
Product : Husky card
Amount : 1
Order placed year : 2022
Order placed month : 3
Order placed date : 15
Total Price : 1
```

```
Block ID: 2
Timestamp: 2022-04-15 10:10:15.5975578 -0400 EDT m=+79.
360590301
Previous Hash: 3353609c46b614c834188c1db80df41f283946
c222014bf12f763c7638baf278
Block Hash :42661bbe9f5d4b085710918383501276efc2b4c5
37ed4e52591e046345469d0e

Transaction Info:

Customer : Sameed
Customer Country : US
Vendor : John
Product : Husky card
Amount : 12
Order placed year : 2022
Order placed month : 3
Order placed date : 15
Total Price : 100
```

```
Block ID: 3
Timestamp: 2022-04-15 10:10:43.3369048 -0400 EDT m=+10
7.099937301
Previous Hash: 42661bbe9f5d4b085710918383501276efc2b4
c537ed4e52591e046345469d0e
Block Hash :f2b1d5300b1a159287cf91a7c9153c1fc59763e35
6d07848e040c27c153e6d08

Transaction Info:

Customer : Lisa
Customer Country : US
Vendor : Nijel
Product : Husky doll
Amount : 14
Order placed year : 2022
Order placed month : 3
Order placed date : 15
Total Price : 200
```

Figure 10: Blocks of supply chain Blockchain



Please fill out all the fields

Order Information

4

2

What's the vendor

What's the product

What's the amount of the product?

What's the year of this order placed?

What's the month of this order placed

1

What's total price of this order

Please input the order informtion

Decentralized database and network all in one

Send

Figure 11: User Interface for Blockchain Database

The user interface contains the functions of collecting inputs from the user, generating blocks, outputting information, and notifying users of wrong data entries. This user interface successfully demonstrates that it is straightforward for users without technical knowledge to use a blockchain database. Moreover, by generating blocks and displaying block and order information, the user interface demonstrated how blockchain technology ensures the security and transparency of the data.

V. Results and discussions

We have successfully built a traditional database, a Blockchain database, and a user interface. To evaluate the functions and building process of the Blockchain database, the team compared the Blockchain database and the user interface with the traditional database built-in GBQ.

In Table 2, the functionalities and limitations of traditional and Blockchain databases are in detail. On the one hand, both databases allow users to add and retrieve data. On the other hand, while traditional databases can delete and analyze data, they cannot track changes and anyone responsible for them. A Blockchain database is immutable; a user cannot delete previous data entries or analyze the data itself. However, it is traceable and transparent by keeping records of changes and making them visible to anyone in the database.

Functionalities	Traditional database	Blockchain database
Retrieve data	**Yes**	**Yes**
Add data	**Yes**	**Yes**
Delete data	**Yes**	No
Calculate data	**Yes**	No
Track who changed the data	No	**Yes**
Track when the change was made	No	**Yes**
Changes are visible for everyone	No	**Yes**

Table 2: Functionalities of Traditional and Blockchain Databases

As shown in Table 3, the traditional database has three layers: project, database, and table, while the whole Blockchain database is stored in a single file.

Tools	Traditional database	Blockchain database	User interface
Platforms	GBQ	PyCharm	Visual studio Code
Language	SQL	Python	Go & JSON
Structure	Project> dataset> tables	Single file	multi-levels
Time Spent	5 hours if familiar with GBQ	45 hours	5 weeks on research and 8 hours on building

Table 3: Building process of Traditional and Blockchain Database and User Interface

The advantage of a traditional database in GBQ is the ease of building and updating the tables. People can do calculations and data analysis on the database using queries. However, the traditional database has many safety, transparency, and speed limitations. For example, people cannot track and find who made the changes to the data.

In terms of safety and transparency, the Blockchain database performs better, by any measure, than the GBQ database. Although Google GBQ performs significantly better in speed and security than localized databases such as MS Access, the Blockchain database offers impeccable deterrence to thefts and sabotages. Because we use a private Blockchain that is not accessible to the public, sensitive data is also free from leak concerns.

VI. Conclusion

The blockchain database, which this paper developed using the Python programming language, ensures the security and transparency of the supply chain information. The database contains all the supply chain data and has the function of adding data and searching for information. Moreover, by generating blocks for each transaction, every change in the database will be recorded. We also built a user interface to demonstrate how users interact with the blockchain database, which helps explain the logic of the blockchain technology. The user interface creates a Webpage that collects the block data from the user, generates the block, and outputs the block information. We have used the Golang and JSON to develop the user interface.

References:

[1]. Ethereum, “Intro To Ethereum,” https://ethereum.org/en/developers/docs/intro-to-ethereum/

[2]. F. Casino, T. K. Dasaklis, & C. Patsakis, " A Systematic Literature Review Of Blockchain-based Applications: Current Status, Classification, and Open Issues," *Telematics and Informatics.*

[3]. Q. K. Nguyen, "Blockchain - A Financial Technology for Future Sustainable Development," 2016 3rd International Conference on Green Technology and Sustainable Development (GTSD), 2016, pp. 51-54, DOI: 10.1109/GTSD.2016.22.

[4]. S. Fan, J. Yan, L. Zhao, “Overview of business innovations and research opportunities in Blockchain and introduction to the special issue,” *Finance Innov*, February 28, 2016

[5]. V. Gaur, & A. Caiha, “Building a Transparent Supply Chain,” *Harvard Business Review,* May 2020.

[6]. R.W. Carlo, De Meijer, “Remaining challenges of Blockchain adoption and possible solutions” *Fineextra.*https://www.finextra.com/blogposting/18496/remaining-challenges-of-Blockchain-adoption-and-possible-solution

[7]. Business Insider, “The growing list of applications and use cases of Blockchain technology in business and life,” https://www.businessinsider.com/Blockchain-technology-applications-use-cases, Last Accessed June 8, 2021.

[8]. M. Sahu, "Hyperledger vs. Ethereum: Difference Between Hyperledger and Ethereum [Which One Should You Use]," *upGrad blog*, 28-Apr-2020. [Online]. Available: https://www.upgrad.com/blog/hyperledger-vs-ethereum-difference-between-hyperledger-and-ethereum

[9]. R. O'Byrne, "Blockchain in the Supply Chain: Who is Using it in 2020 and How?," *Logistics Bureau*, 28-Jul-2020. [Online]. Available: https://www.logisticsbureau.com/whos-using-Blockchain-in-2020-and-how/.

[10]. Hyperledger Foundation, “How Walmart brought unprecedented transparency to the food supply chain with Hyperledger Fabric” https://www.hyperledger.org/learn/publications/walmart-case-study

[11]. Tradelens, “Trade Made Easy” https://www.tradelens.com

[12]. Tradelens, “Abu Dhabi Customs joins Tradelens to work with exporters, importers to explore Blockchain underpinned document flows” https://www.tradelens.com/post/abu-dhabi-customs-joins-tradelens