# ABSTRACT

This report delves into the intricate process of data warehousing and business analysis, with a focus on leveraging a robust data warehouse design and advanced analytical techniques. It covers the methodology from database design, data extraction, and transformation, to loading processes using tools like Apache NiFi, Docker, and Airflow. The report also explores data visualization, mining, and provides strategic recommendations for business enhancement. It aims to demonstrate how integrated data management and analysis can drive insightful business decisions and optimize operational efficiency.

**TABLE OF CONTENTS**

[**ABSTRACT 2**](#_3csjzspyknc)

[**I. Introduction 5**](#_95rieq2hmbhk)

[A. Business Problem 5](#_s80w43ae9lif)

[B. Questioning 5](#_7rqh0yt0g3gb)

[**II. Methodology 6**](#_1jrox5au5xn0)

[A. Database design 6](#_j1yp8j8pzcou)

[1. Source database 6](#_65bb10x458a4)

[2. Physical design 9](#_3nyyfc7tisnh)

[3. Four-Step Data Warehouse construction 11](#_okx0sg1ynn3v)

[3.1 Choose the business process 11](#_u1ypuw730yet)

[3.2 Declare the grain 12](#_rxb17mbeeqyz)

[3.3 Identify the Dimensions 13](#_ouhpkgvqt6v9)

[3.4 Identify the Facts 14](#_nwoyk4qbpcf9)

[B. Extract (Apache Nifi) 15](#_1elykkl5rqu3)

[1. Create virtual machine on Google Bigquery 15](#_tzi3pdbrpsyl)

[2. Connect to SSH to begin the Nifi and installation process 15](#_urud0aunkrag)

[2.1 This process will include the following basic steps 15](#_cg2qepxmbvw1)

[2.2 Navigate to Nifi configuration 16](#_sju359inolx5)

[3. Set up Firewall in VPC Network 16](#_oh83xs2xvqv)

[4. Create processors in Nifi Apache 17](#_jp1xyuax1epj)

[4.1 GetFile Processor 17](#_9t2kjyhkc38o)

[4.2 UpdateAtribute Processor 18](#_bpwtif7lb2aw)

[4.3 PutBigQueryBatch 19](#_rbmy7hhveu34)

[5. Nifi workflow 20](#_29jm8v84jvnh)

[C. Transform and Load (Docker and Airflow) 21](#_1yjslcsjndhe)

[1. Content of the yaml configuration file 21](#_55xt8uruq80l)

[2. The content of the code used for DAG configuration 27](#_5y9xr7ale2jz)

[3. Transform code for DWH tables 29](#_t9xa1qte7pzm)

[4. Airflow result 33](#_dteoyoaqaq1d)

[**III. Visualization 34**](#_jqjdyt9c0vla)

[A. Revenue Dashboard 34](#_teg9fnitw9s)

[B. Products Dashboard 36](#_bf4iqj8odwg5)

[**IV. Data mining 37**](#_qzp8g4hq6y8j)

[**V. Recommendation 42**](#_r63nudechji6)

[1. Enhance the customer experience 42](#_7rjfuxeliwwo)

[2. Leverage social media and online platforms 42](#_l05a8bn0yacj)

[3. Implement a referral program 42](#_safuzw22blzq)

[4. Utilize email marketing 42](#_y5z4fn7d78xz)

[5. Collaborate with influencers or industry partners 43](#_u8dsljf2x89i)

[6. Monitor and respond to customer feedback 43](#_4qn1ajlbgdkb)

[7. Continuously analyze and optimize marketing efforts 43](#_mdpk9rpzb58)

[**VI. Conclusion 43**](#_2bwx4vcguw2x)

[**References 45**](#_h6jdx5bhzc0c)

[**Member Contribution 45**](#_u80e3v2z92z5)

# 

# I. Introduction

## A. Business Problem

In today's intricate business environment, extracting valuable insights from extensive datasets is crucial for maintaining a competitive advantage. The Northwind database, conceptualized by Microsoft as a fictional model, serves as an effective tool for comprehending and addressing the complex challenges prevalent in everyday business operations. This exploration into Northwind is not just an exercise in enhancing database management skills, but also a deep dive into real-world business complexities that mirror actual commercial transactions.

Northwind represents a typical trading company, reflecting the real challenges businesses face in areas like customer relationships, inventory management, order processing, and supplier engagement. These elements are deeply interconnected, showcasing the comprehensive nature of modern commercial activities. The Northwind database captures this intricacy, providing a detailed portrayal of the diverse aspects businesses must skillfully navigate to achieve prolonged success.

## B. Questioning

After identifying the business problems, we asked ourselves questions to clarify and solve business problems through data analysis and the use of machine learning. Here are the questions we asked:

* Which product had the highest revenue in the Top 10 Products by Order?
* What is the revenue difference between the top product and the tenth product in the Top 10 Products by Order?
* Which product appears to have the highest stock availability in the Revenue by Category and Product section?
* How does the revenue of the top product in Beverages category compare to the top product in the Dairy Products category?
* What is the total revenue generated by the Beverages category?
* Can you identify any trends in product performance based on the stock levels and revenue data?
* What is the total sales figure displayed in the Executive Overview - Revenue?
* Which country has the highest total sales according to the Total Sales by Location and Manager?
* Who is the top-performing manager in the UK, and what is their sales figure?
* What is the average discount given on sales across all regions?
* Which sales category generates the most revenue according to the Sales by Category chart?
* How does the total revenue of the USA compare to that of the UK?
* How many customer segmentation models are there?
* What are the characteristics of each segmentation?

# **II. Methodology**

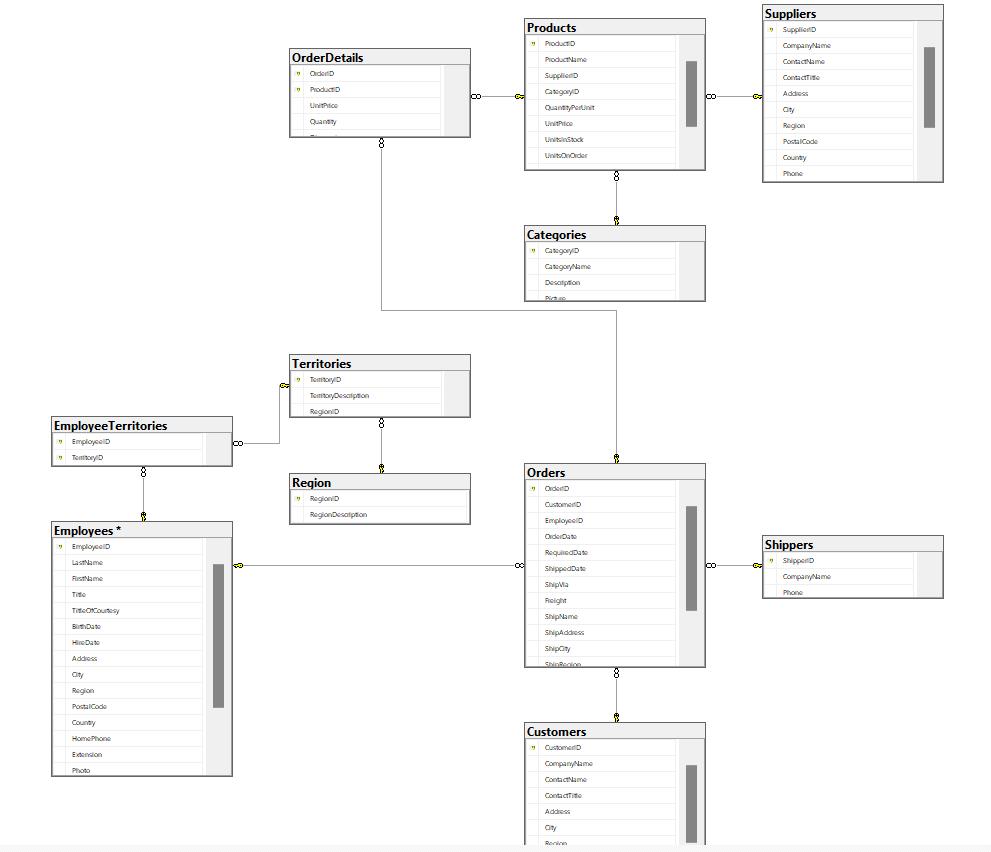
* **Database Design:** This phase focuses on the structure and schema of the Northwind database. It includes a detailed breakdown of various tables such as Customers, Employees, Orders, Products, and Suppliers. The design process ensures that these tables form a comprehensive framework for capturing essential business operations.
* **Extract (Apache Nifi):** The extraction process is handled using Apache Nifi. It involves creating a virtual machine on Google Bigquery and configuring Nifi for data ingestion. Key steps include setting up processors like GetFile, UpdateAttribute, and PutBigQueryBatch in Nifi for efficient data extraction and preparation.
* **Transform and Load (Docker and Airflow):** Transformation and loading of data are executed using Docker and Airflow. This segment outlines the configuration of a docker-compose file for deploying Airflow and details the code used for Directed Acyclic Graph (DAG) configuration. It also discusses the transformation code for Data Warehouse (DWH) tables, emphasizing the creation and updating of these tables in Google BigQuery.

In summary, the methodology combines meticulous database design, sophisticated extraction techniques, and advanced transformation and loading processes. This integrated approach ensures a robust and efficient handling of data from its initial structure to its final analysis-ready state.

## ***A. Database design***

### **1. Source database**

The Northwind database is a fictional sample database created by Microsoft for educational and demonstration purposes. It is commonly used as a learning tool for understanding database concepts and practicing SQL queries. Here are more details about the source database and its main tables:



*Figure 1: NorthWind source database*

1. Customers:

* Fields: CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax.
* Represents information about customers who purchase products from the company.

2. Employees:

* Fields: EmployeeID, LastName, FirstName, Title, TitleOfCourtesy, BirthDate, HireDate, Address, City, Region, PostalCode, Country, HomePhone, Extension, Photo, Notes, ReportsTo, PhotoPath.
* Contains details about company employees, including personal and professional information.

3. Orders:

* Fields: OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, ShipCountry.
* Represents individual customer orders, including order details and shipping information.

4. Products:

* Fields: ProductID, ProductName, SupplierID, CategoryID, QuantityPerUnit, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued.
* Contains information about the products available for sale, including product details and inventory information.

5. Categories:

* Fields: CategoryID, CategoryName, Description.
* Categorizes products into different groups or categories.

6. Suppliers:

* Fields: SupplierID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax, HomePage.
* Provides information about companies that supply products to Northwind.

7. Shippers:

* Fields: ShipperID, CompanyName, Phone.
* Contains details about shipping companies used by Northwind.

8. OrderDetails:

* Fields: OrderID, ProductID, UnitPrice, Quantity, Discount.
* Represents the line items for each order, including product details, quantity, and price.

9. EmployeeTerritories:

* Fields: EmployeeID, TerritoryID.
* Maps employees to specific geographic territories.

10. Region:

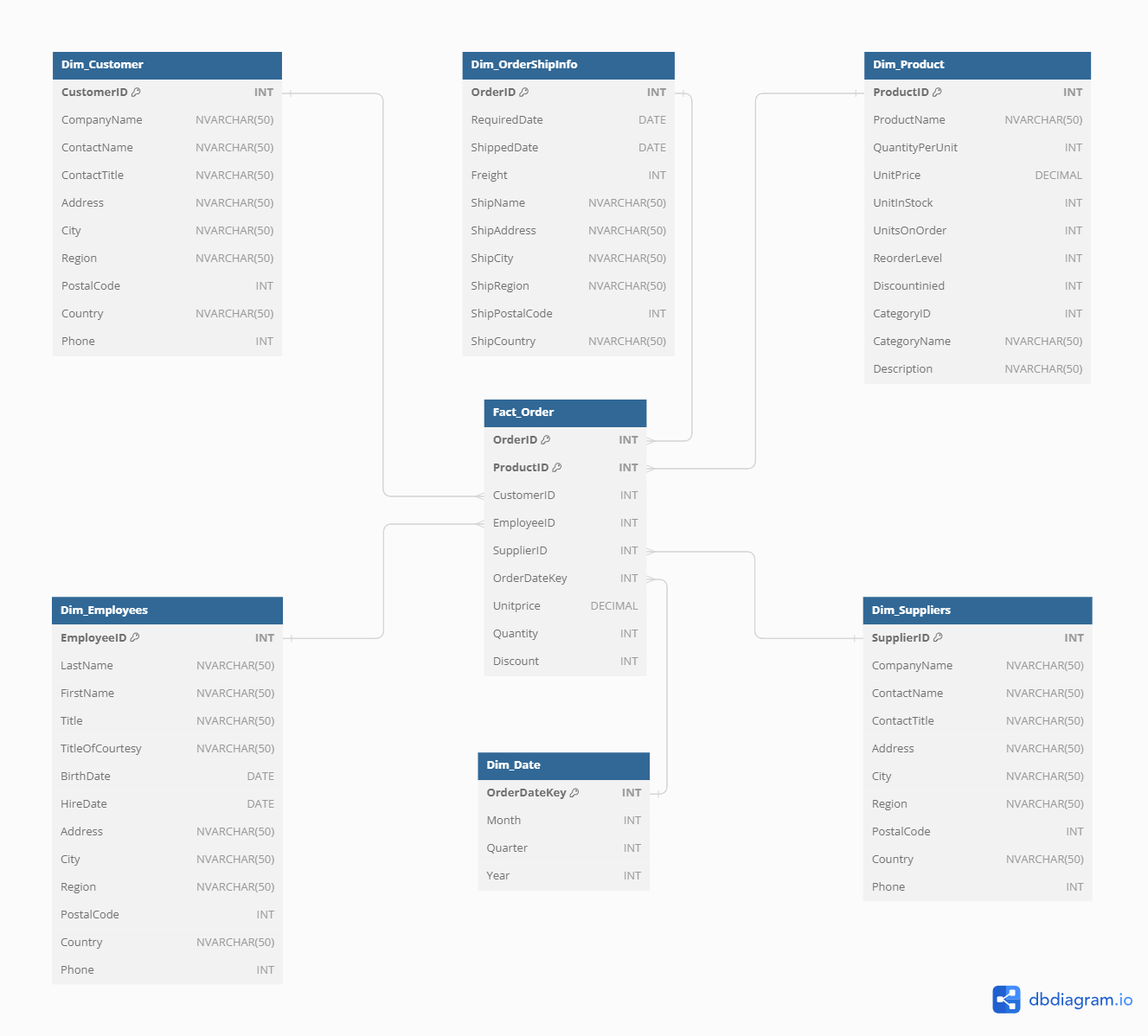
* Fields: RegionID, RegionDescription.
* Contains information about different geographic regions.

11. Territories:

* Fields: TerritoryID, TerritoryDescription, RegionID.
* Represents specific territories within regions.

These tables together form the core structure of the Northwind database, covering aspects of customers, employees, orders, products, suppliers, shipping, and more. Keep in mind that variations of the Northwind database may exist, and some additional tables or modifications might be present depending on the specific version used.

### **2. Physical design**



*Figure 2: Physical design of NorthWind Data Warehouse*

The provided database schema visualizes a meticulously designed data warehouse structure, characterized by a collection of six dimension tables: Dim\_Customer, Dim\_Employees, Dim\_OrderShipInfo, Dim\_Product, Dim\_Suppliers, and a single fact table, Fact\_Order, supplemented by a Dim\_Date table. In this schema, the Dim\_Customer table is dedicated to storing static customer information, including CustomerID, CompanyName, ContactName, and comprehensive address details, which is fundamental for customer dimension analysis. The Dim\_Employees table encapsulates data about employees, comprising EmployeeID, LastName, FirstName, and professional details such as Title and HireDate, thus facilitating analyses centered around employee performance and demographics.

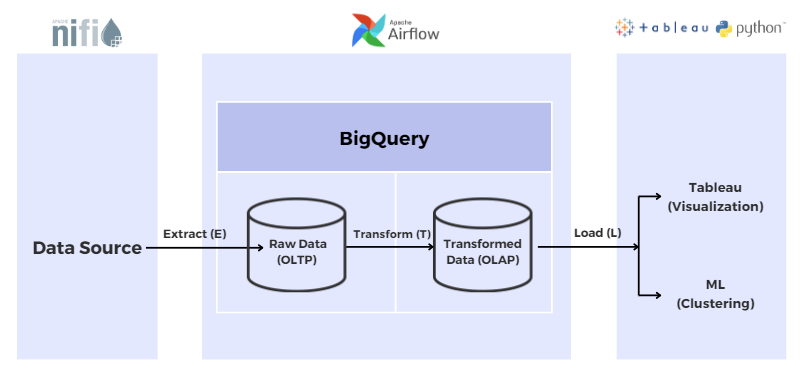
Further delving into logistics, the Dim\_OrderShipInfo table is established to maintain shipping-specific details for orders, with fields like OrderID, ShippedDate, and ShipAddress, allowing an analytical deep-dive into shipping logistics and associated expenditures. The Dim\_Product table serves as a repository for product-related data, including ProductID, ProductName, UnitsInStock, and UnitsOnOrder, which are crucial for inventory management and sales trend analysis. On the supplier front, the Dim\_Suppliers table is charged with tracking supplier details, encompassing SupplierID, CompanyName, and ContactName, thereby enabling an assessment of the supply chain's impact on overall operations.

The temporal analysis is empowered by the Dim\_Date table, which structures time into OrderDateKey, Month, Quarter, and Year, thereby facilitating trend analysis and reporting over various time frames. At the core of transactional data lies the Fact\_Order table, which quantifies transactions through OrderID, Unitprice, Quantity, and Discount, and is intricately linked to the dimension tables via foreign keys, providing a comprehensive view of the business transactions.

This schema illustrates a classic star schema configuration, commonly employed in data warehousing to optimize query performance and simplify reporting. Primary keys, denoted by a # symbol, ensure uniqueness within each table, while foreign keys, indicated by a link symbol, establish the relational threads that weave through the tables. The strategic use of NVARCHAR for textual fields indicates an internationalization capability, given its support for Unicode characters. INT data types for keys ensure efficiency in join operations, and the DECIMAL data type for Unitprice in Fact\_Order allows for precise financial analysis.

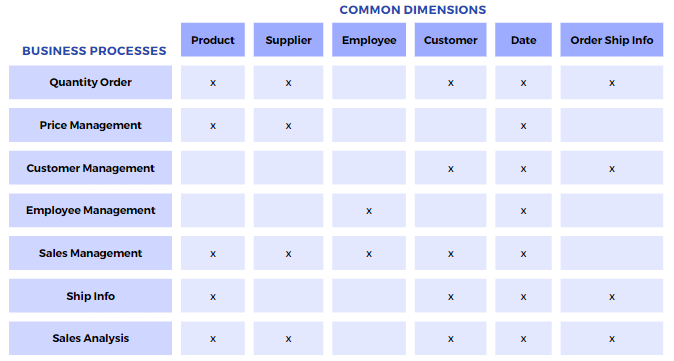
The OrderDateKey presents a pivotal role, bridging Fact\_Order with Dim\_Date for temporal insights into the order data. The schema's normalization facilitates the separation of concerns, which not only upholds data integrity but also streamlines data management processes. While the cardinality and specific relationships between tables are not explicitly depicted, they are essential for a thorough understanding of data interactions and are typically governed by foreign key constraints.

In the context of use cases, it might be necessary to introduce indexes on foreign keys and heavily queried fields to further enhance query performance. Collectively, this schema is engineered to support complex multidimensional queries, which are integral to dissecting business operations across various vectors such as sales performance, inventory tracking, employee contributions, shipping logistics, and supplier relationships. Ultimately, the design is conducive to both day-to-day operational reporting and strategic business intelligence analysis, enabling stakeholders to make informed, data-driven decisions.



### 3. Four-Step Data Warehouse construction

#### **3.1 Choose the business process**



*Figure 3: Bus matrix*

The row of the matrix corresponds to business processes. The columns of the matrix correspond to natural groupings of standardized descriptive reference data. At this stage of the game, the matrix is a preliminary draft because we have not done a detailed analysis of each business process row. We merely want to focus on the key descriptive nouns and process verbs to visually present the integration opportunities and challenges to senior management.

This is the Order section of the data in this instance, which includes quantitative measurements (facts) like Quantity and Price. The dimension tables, which offer descriptive data (such as Customer Name, Product Name, Address, and basically anything that cannot be measured quantitatively), branch outward. We are simulating the Northwind data's sales process. The model's core contains the sales data, which indicates the total amount sold. The branches provide more specific details about individual sales.

In accordance with Kimball's book, it is important to incorporate time elements in the analysis to ensure that the dimension of date is relevant and applicable to all various business processes. This thorough examination is done to guarantee the efficiency of Northwind's receiving and delivery process, optimizing costs and profits, and ensuring positive customer feedback.

Product Types play a crucial role in the majority of business processes as they provide essential information for Northwind to understand what they are selling and effectively plan their sales strategies, including pricing and marketing. This comprehensive understanding of product information enables Northwind to make informed decisions and optimize their sales approach for maximum effectiveness.

Customer Details and Ship method both belong to four processes: Quantity Order, Customer Management, Ship Information and Sales Analysis. Because customer information is related to shipping. Therefore, this management will help Northwind's shipping always reach the right address and on time.

The Suppliers Details is essential for both Quantity Order and Price Management in order for Northwind to manage the quantity of imported goods as well as prices, and thereby help Northwind consider selling prices in the Sales Management section.

#### **3.2 Declare the grain**

*For Fact\_Order, we define the grain as follows:*

Grain Definition: Each record in the Fact\_Order table represents the sale of a product within an order. This means that if an order consists of three different products, there will be three records in the Fact\_Order table.

Objective: Establish the most detailed level at which business events are recorded in the Fact\_Order table.

*Detailing the Grain:*

* Each row in Fact\_Order is a line item within an order.
* The OrderId will likely repeat for multiple line items, but the combination of OrderId and ProductId should be unique, ensuring that each row represents a specific product within a specific order.

*Attributes Affected by the Grain:*

* Quantity: Reflects the quantity of the ProductID sold in the OrderId.
* UnitPrice: The price of a product corresponds to ProductID.
* Discount: If applicable, shows the discount given on the ProductID within the OrderID.
* SupplierID: Identifies which supplier provided the ProductID in the OrderID.
* EmployeeID: Associates an employee with the sale of the ProductID in the OrderID.
* CustomerID: Links the ProductID sale to a customer in the OrderID.
* Date: Order date data is based on OrderID and this data is detailed down to date, month, year

#### **3.3 Identify the Dimensions**

The diagram above displays the dimension tables that will support the Fact\_Order table in our star schema. The following tables are:

* Dim\_Customer: This table is about information of customers who bought the product.
* Dim\_OrderShipInfo: This table provides information of shipment of each order.
* Dim\_Product: This table tells about the information of each product such as name, quantity, price,...
* Dim\_Employees: This table is about the information of the employee managing order of the customer.
* Dim\_Date: This table will allow us to perform robust date filtering, more so than traditional SQL date functions.
* Dim\_Suppliers: This table tells about the information of suppliers in each orders.

#### **3.4 Identify the Facts**

The deliberate selection of "Fact\_Order" as our fact table underscores a strategic decision to enhance data analytics for our order processing system. In this star schema, OrderID and ProductID serve as primary keys, providing unique identifiers for each order and ensuring a robust foundation for relational connections. The incorporation of CustomerID, EmployeeID, SupplierID, OrderDateKey, UnitPrice, Quantity, and Discount as foreign keys enriches the fact table with contextual information, fostering comprehensive insights into the intricacies of our order-related processes.

The choice of OrderID and ProductID as primary keys establishes a solid foundation for data integrity and facilitates seamless integration with other tables, allowing for efficient relational connections. CustomerID, EmployeeID, and SupplierID as foreign keys contribute vital contextual details, enabling us to analyze customer behavior, employee contributions, and supplier dynamics in the context of individual orders.

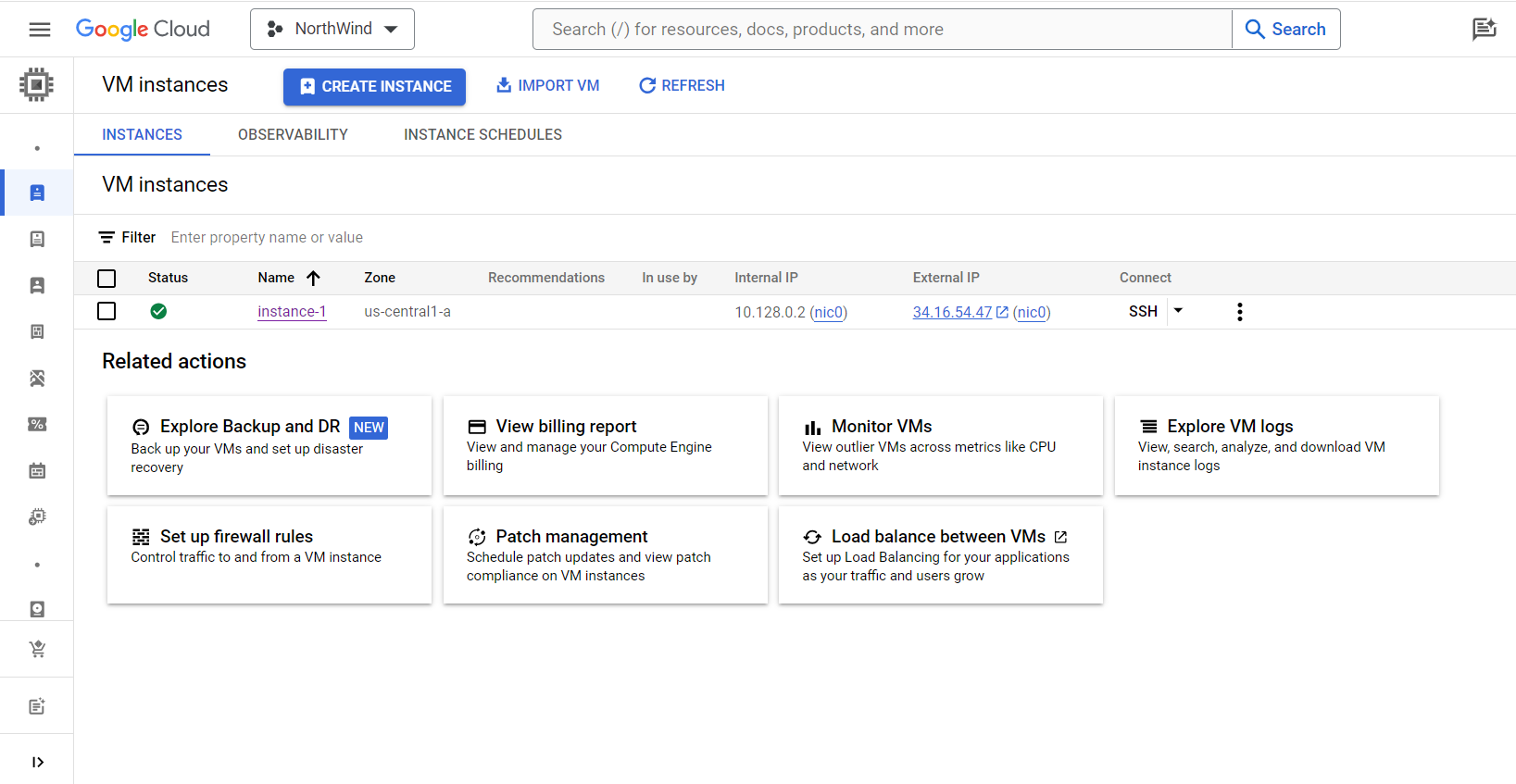
The inclusion of OrderDateKey allows for time-based analysis, providing insights into historical trends and seasonality, crucial for understanding the temporal dynamics of order placements. Meanwhile, the incorporation of UnitPrice, Quantity, and Discount as foreign keys offers a quantitative foundation for financial analysis, supporting evaluations of pricing strategies, order quantities, and discount impacts.

In summary, "Fact\_Order" emerges as a well-structured and interconnected fact table that consolidates essential entities for a holistic view of our order-related processes. The choice of primary and foreign keys in this schema ensures both data integrity and analytical depth, making it a strategic centerpiece for efficient data retrieval, comprehensive analysis, and informed decision-making in our business intelligence endeavors.

## B. Extract (Apache Nifi)

### **1. Create virtual machine on Google Bigquery**

We will use the Compute Engine service provided on GG Bigquery to create a VM Instance using the Ubuntu operating system (ver 20.04 LTS).



*Figure 4: VM instances*

This is an image of virtual machine VM instances after configure completed.

### **2. Connect to SSH to begin the Nifi and installation process**

#### **2.1 This process will include the following basic steps**

* Use the command **“sudo su”** to switch to the superuser, usually the "root" user. The root user has unlimited access to most parts of the system. This includes the ability to modify any files, change system settings, install and remove software, and much more.
* Next use **“Apt Update”** to download the latest list of software packages from all repositories defined in /etc/apt/sources.list and files in the directory.
* **Install Java Development kit** (jdk) using apt install openjdk-17-jdk
* And this is an important step, we will install Nifi, the version we use is 1.0.24 with the following command: https://archive.apache.org/dist/Nifi/1.24.0/Nifi-1.24 .0.-bin.tar.gz
* After downloading Nifi, we will proceed to extract it using the command "tar -xzvf Nifi-1.24.0-bin.tar.gz”.

#### **2.2 Navigate to Nifi configuration**

* Edit Nifi properties by command: vi Nifi.properties
* Firstly, we need to press “I” to start adjust the properties
* We need to change Nifi.remote.input.http.enable: From true to False.
* Nifi.web.http.port: 8088 => After this step we can launch Nifi but to display it we need to setup firewall.
* And then in the “Security properties”, we need to make sure that everything is clear:  
  nifi.security.keystore=

nifi.security.keystoreType=

nifi.security.keystorePasswd=

nifi.security.keyPasswd=

nifi.security.truststore=

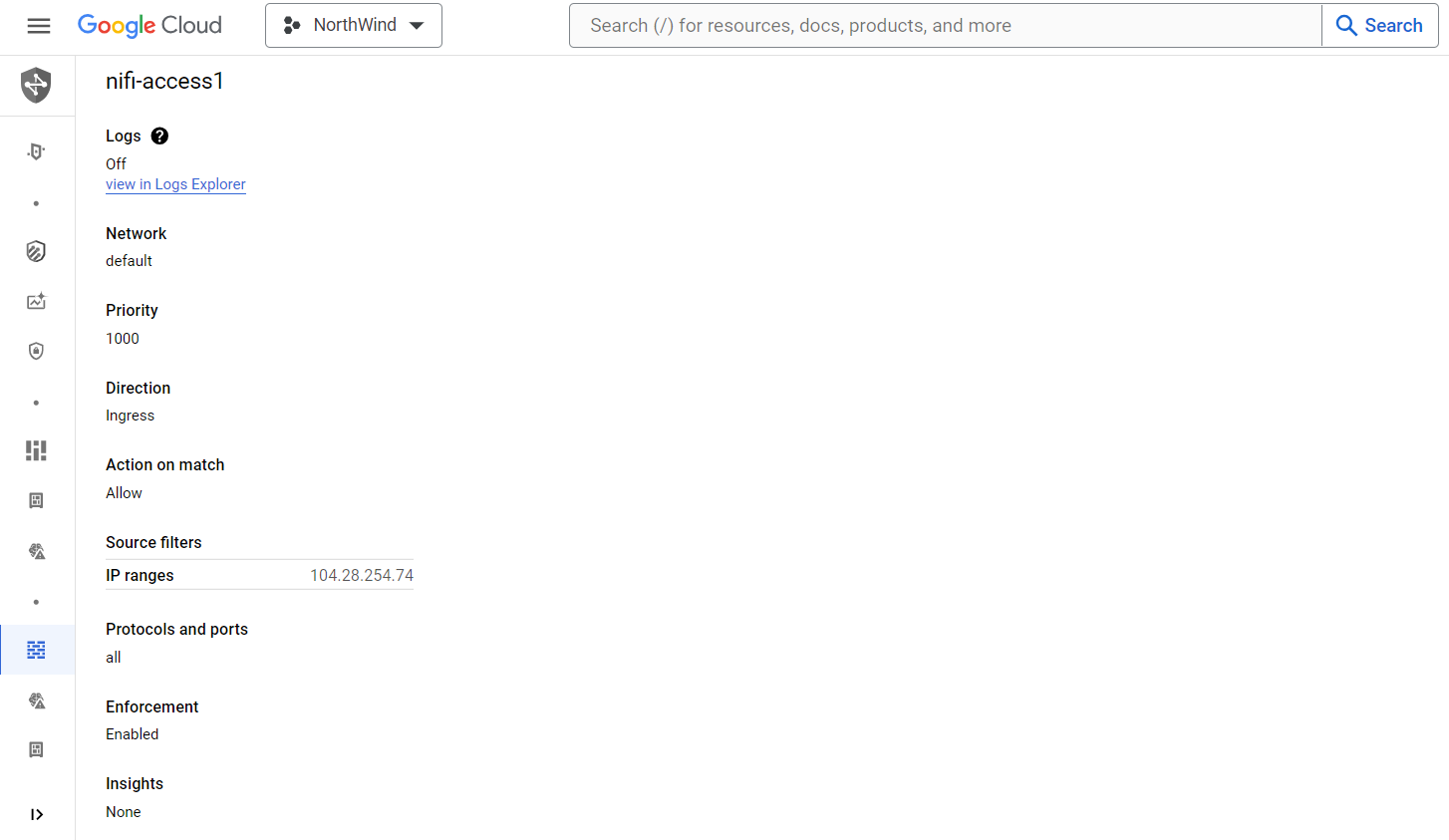
nifi.security.truststoreType=

nifi.security.truststorePasswd=

* Lastly, we pressed “Esc” to escape from adjustment status and saved the configuration by using command “:wq”

### **3. Set up Firewall in VPC Network**

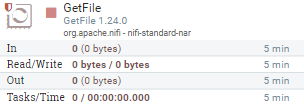
Set the name of the firewall -> set the IPv4 address -> go to TCP and enter 8088 as when setting up Nifi.properties. This is how we set up the Firewall:



*Figure 5: Firewall set up*

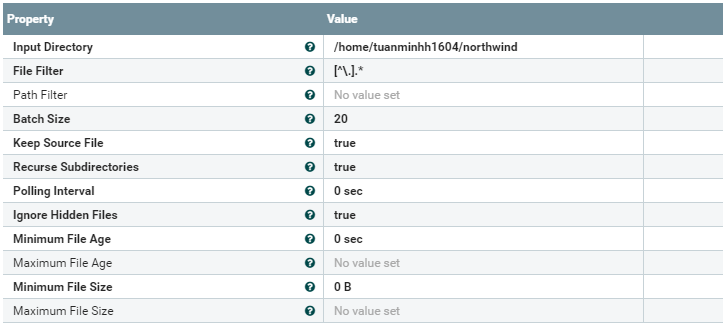
### **4. Create processors in Nifi Apache**

#### **4.1 GetFile Processor**

******

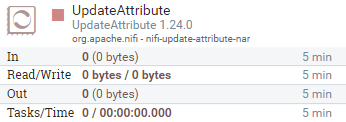
*Figure 6: GetFile Processor*

**The GetFile** processor is used to ingest data into the Nifi data flow. It reads files from a specified directory on the local file system. Here we will push data to the AVRO files from the library we have initialized. Here's how we configured it:

******

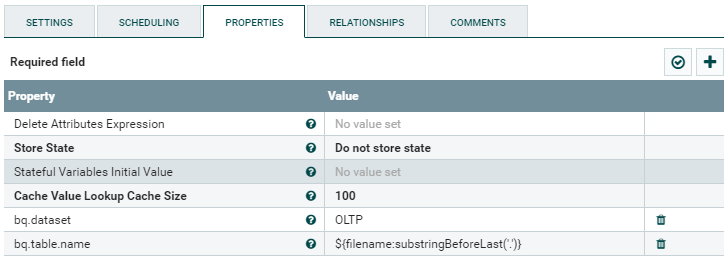
*Figure 7: GetFile Processor Properties*

#### **4.2 UpdateAtribute Processor**

**

*Figure 8: UpdateAtribute Processor*

**The UpdateAttribute** processor is used to update or add attributes to a FlowFile. Attributes are key-value pairs associated with a FlowFile that can be used for routing or processing decisions later in the flow. How to configure as follows:

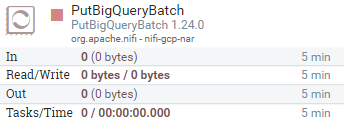
**

*Figure 9: Update Attribute Processor Properties*

We will add those two tables bq.dataset and bq.table.name to accommodate the final processor.

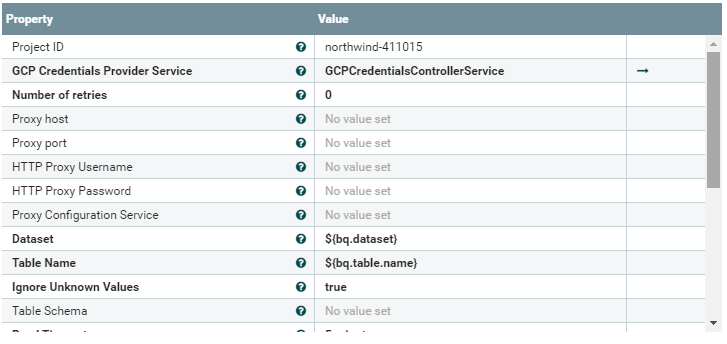
* bq.dataset: This custom property likely represents the dataset name for a BigQuery destination. The value "OLTP" suggests that this is the dataset where the data should be inserted or updated.
* bq.table.name: This is an expression language statement that dynamically sets the table name for BigQuery based on the filename of the FlowFile.

#### **4.3 PutBigQueryBatch**

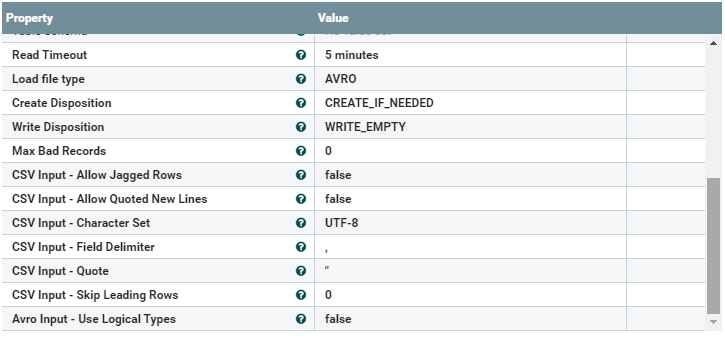
**

*Figure 10: PutBigQueryBatch Processor*

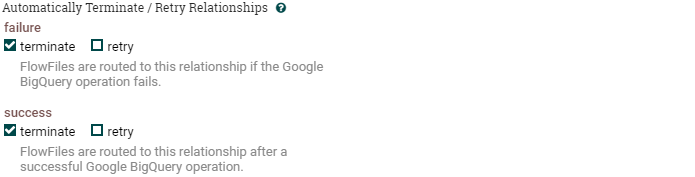
**The PutBigQueryBatch** processor is used to send batches of FlowFiles to Google BigQuery. This processor is tailored to work with Google Cloud services and is used to load data into BigQuery tables. How to configure as follows:

**

*Figure 11: PutBigQueryBatch Processor Properties*

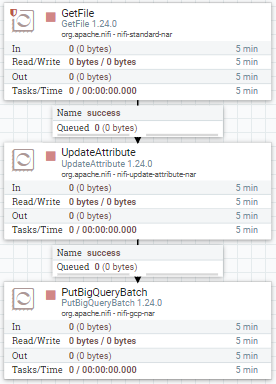
**

*Figure 12: PutBigQueryBatch Processor GCP Credentials configuration*

**

*Figure 13: PutBigQueryBatch Processor Relationship Configuration*

### **5. Nifi workflow**

******

*Figure 14: Nifi workflow*

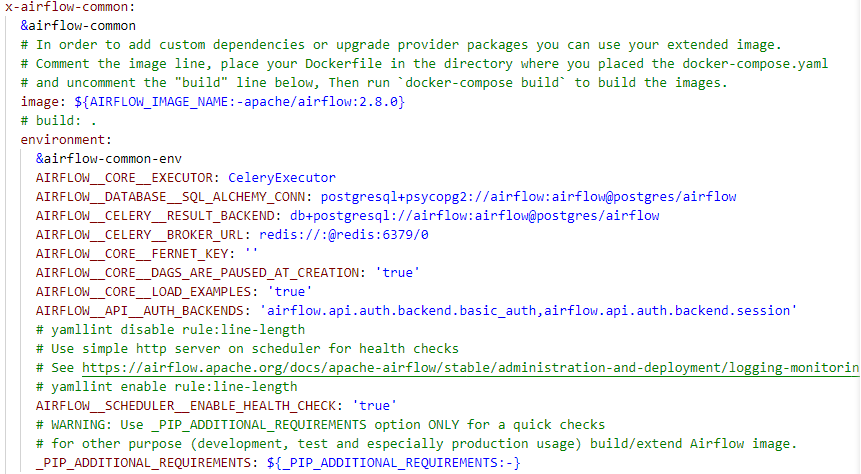
The process likely works as follows:

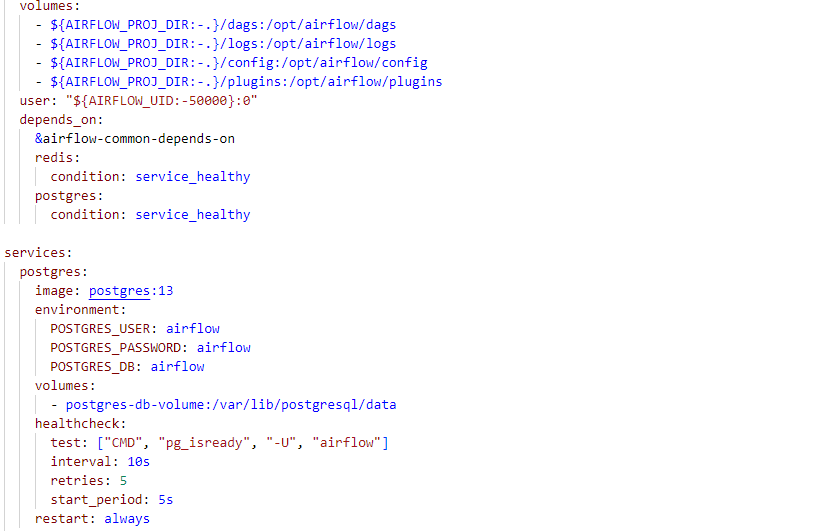
* The GetFile processor picks up files from a local directory and passes them as FlowFiles to the next processor in the flow.
* The UpdateAttribute processor modifies or adds attributes to the FlowFile's metadata, which can be used for routing or processing by other processors downstream.
* The PutBigQueryBatch processor takes the FlowFiles (potentially after some transformation to the data they contain) and inserts the data into a BigQuery table in batch mode.

Each processor's performance and the number of tasks it has completed are not shown in the image but would typically be monitored in a live Nifi environment to ensure the dataflow is operating as expected.

## C. Transform and Load (Docker and Airflow)

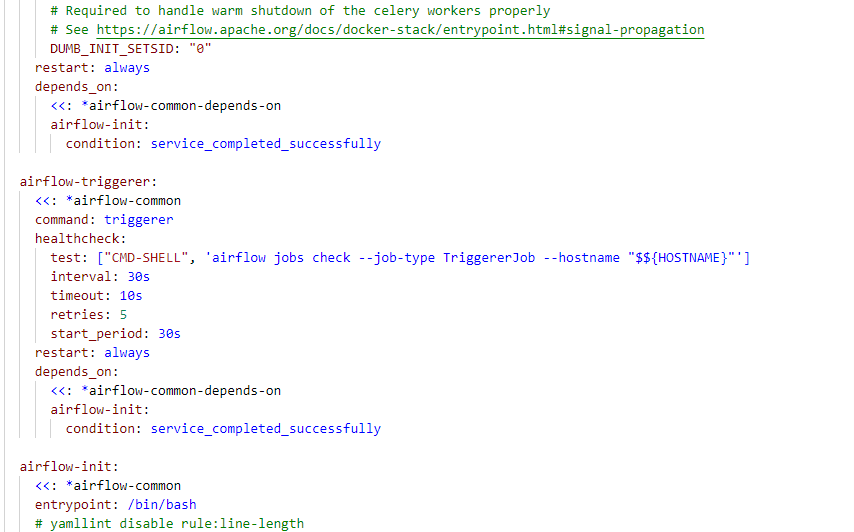
### **1. Content of the yaml configuration file**

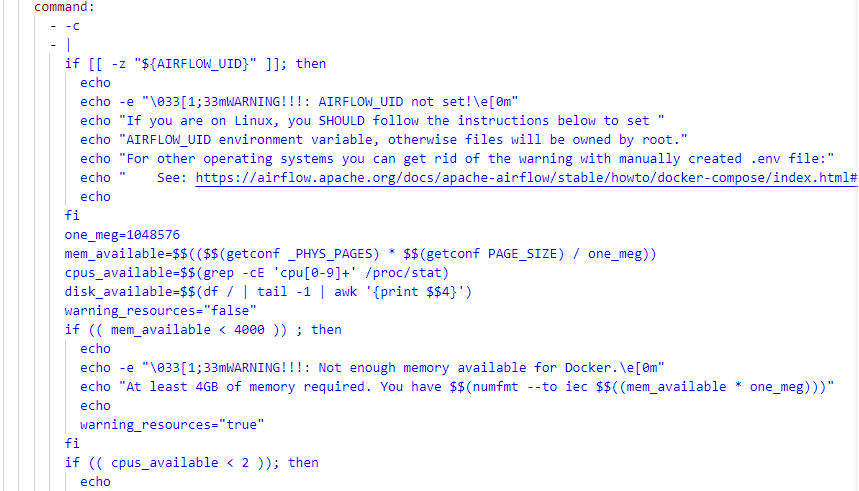


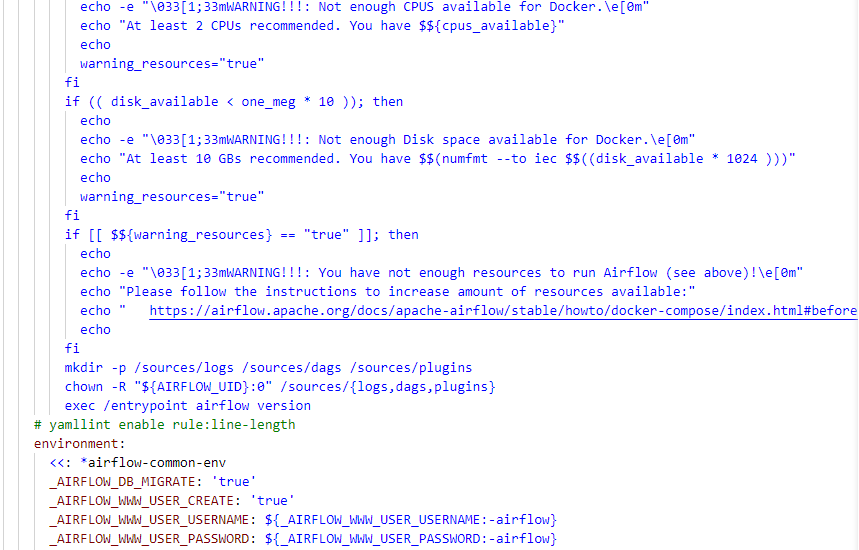




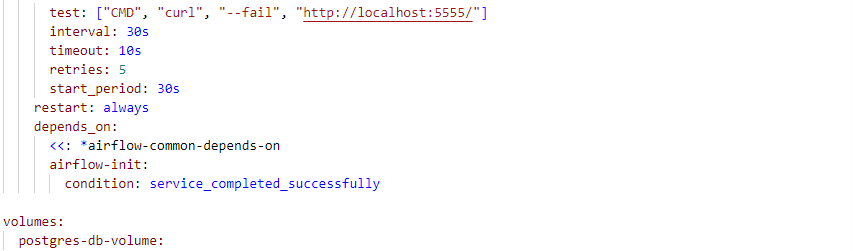












*Figure 15: docker-compose.yml*

The docker-compose.yaml file is a comprehensive setup for deploying Apache Airflow, a platform for managing complex computational workflows and data processing pipelines. This configuration underscores a multi-service Docker application, detailing various essential components for a fully operational Airflow system.

The core of the setup is defined under x-airflow-common, a YAML anchor that establishes common configurations for Airflow services. This includes specifying the Docker image for Airflow (apache/airflow:2.8.0), essential environment variables for database connections (using PostgreSQL and CeleryExecutor), broker settings (using Redis), and volume mappings for logs, dags, config, and plugins directories. This anchor is reused across multiple services to maintain consistency and ease of management.

The services section outlines the architecture:

Postgres: This service configures the PostgreSQL database, which is fundamental for Airflow's metadata storage. It uses the postgres:13 image and sets up the necessary environment variables for user authentication. A volume postgres-db-volume is defined for data persistence.

Redis: As a message broker, the Redis service is configured with the redis:latest image. It exposes the default Redis port and includes comprehensive health checks.

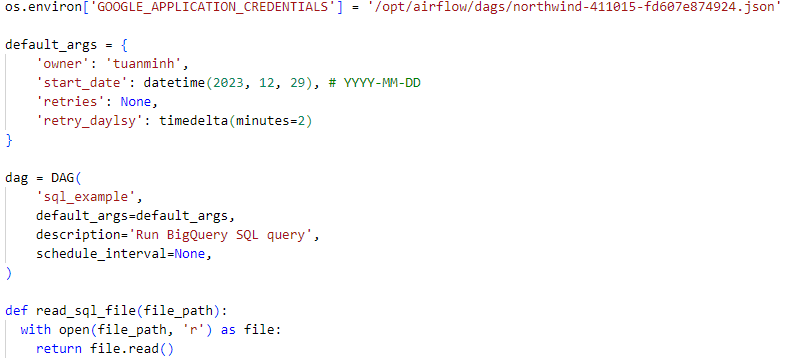
Airflow Components:

1. airflow-webserver: Configures the webserver for Airflow's UI, mapping it to port 8080 on the host.
2. airflow-scheduler: Sets up the scheduler, responsible for triggering workflow tasks.
3. airflow-worker: Defines Celery workers for task execution, with a health check to ensure proper operation.
4. airflow-triggerer: Manages triggers in Airflow 2.0, an addition to the platform.
5. airflow-init: Handles initial setup tasks like database migrations, user creation, and system resource checks.
6. airflow-cli: Provided for command-line interactions and debugging purposes.

Flower: An optional service for monitoring Celery workers, enabled through profiles and accessible via port 5555.

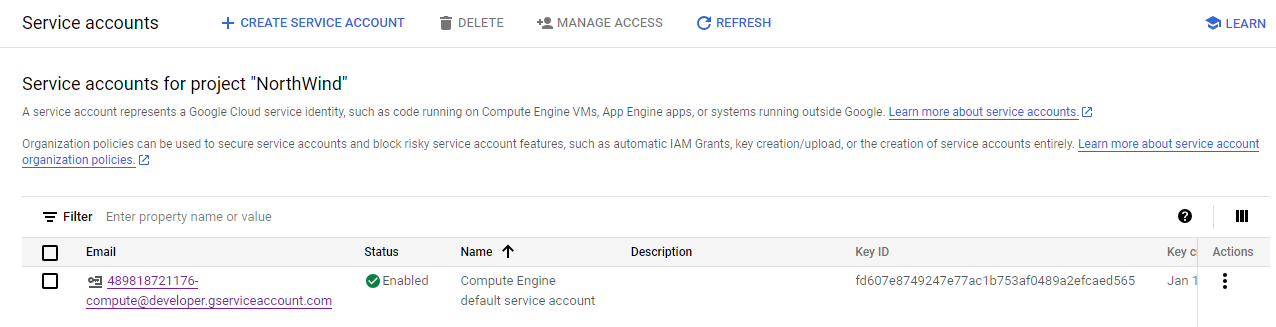
Each service is meticulously configured with health checks, restart policies, and dependencies, ensuring a robust setup. The docker-compose.yaml file stands out for its modular design, allowing for easy customization and scaling of the Airflow environment. This approach is particularly useful for development, testing, and even production deployments, providing a detailed and practical guide for orchestrating Airflow with Docker.

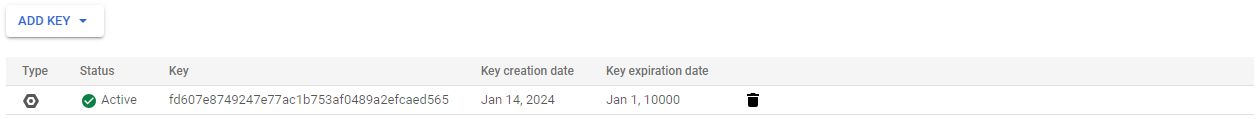
### **2. The content of the code used for DAG configuration**

****

*Figure 16: Code for DAG configuration*

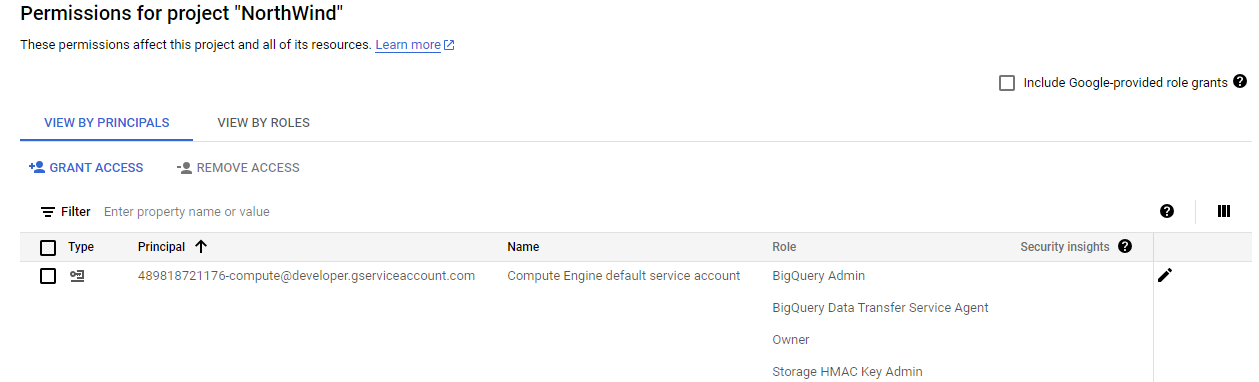
This Apache Airflow script begins by setting up the necessary environment for interacting with Google Cloud services. The line *os.environ['GOOGLE\_APPLICATION\_CREDENTIALS']* specifically points to a JSON file containing credentials, essential for tasks involving Google Cloud, like BigQuery operations.





*Figure 17: Service account section in Google Big Query*

To obtain the required credential file, one must first navigate to the 'Service Account' section. Here, a key corresponding to the account must be created and then exported as a JSON file. Subsequently, this JSON file should be securely saved within the DAG file on the local computer.



*Figure 18: IAM section in Google Big Query*

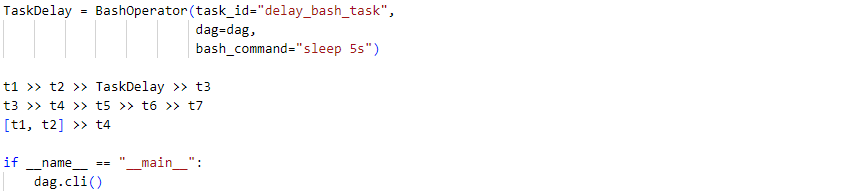
Additionally, in the 'IAM' (Identity and Access Management) section, it is essential to assign the appropriate roles to the account. This step is crucial to ensure that Airflow is granted the necessary permissions to access our database

Following this, the script defines default\_args, a configuration dictionary for the Directed Acyclic Graph (DAG). This includes the owner ('tuanminh'), a start date (datetime(2023, 12, 29)), indicating when Airflow should begin executing the DAG's tasks. Interestingly, while it specifies a retry\_daylsy of two minutes, it sets retries to None, meaning failed tasks won't retry. The DAG itself, named 'sql\_example', is set up with these default arguments and is described as a tool to run BigQuery SQL queries. Notably, it has a schedule\_interval of None, implying it requires manual or external triggers rather than running on a schedule. Lastly, the script includes a utility function, read\_sql\_file(file\_path), designed to read and return the contents of SQL files. This function is crucial for loading and executing SQL queries dynamically, enhancing the script's modularity and maintainability for database operations.

### **3. Transform code for DWH tables**

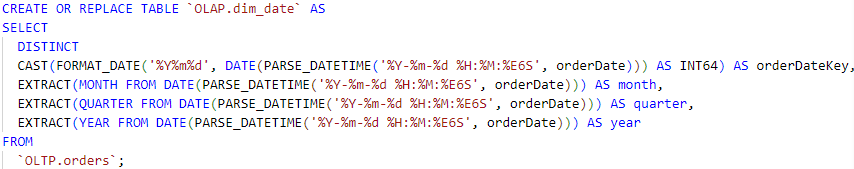




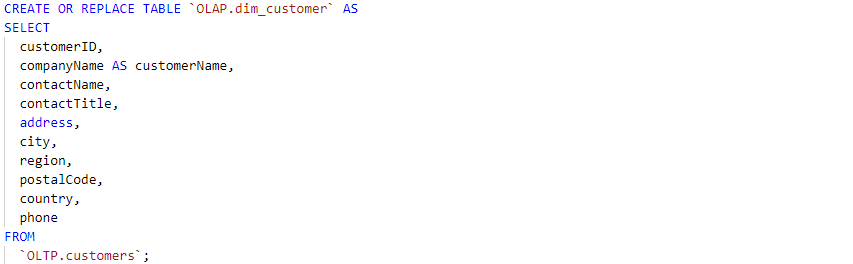


*Figure 19: Transform code for DWH tables*

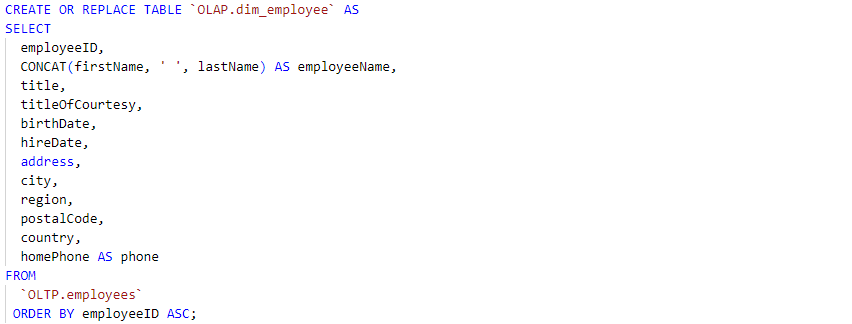
The script in question is crafted for Apache Airflow, a sophisticated platform designed to author, schedule, and monitor various workflows programmatically. Its primary function is to facilitate the creation and updating of tables in Google BigQuery, leveraging the BigQueryOperator. The initial step involves the extraction of SQL queries from specific files located within the /opt/airflow/dags/SQL\_query/ directory. These queries, each dedicated to constructing a distinct set of dimensional (Dim) and factual (Fact) tables in a database, are read by invoking the read\_sql\_file function.



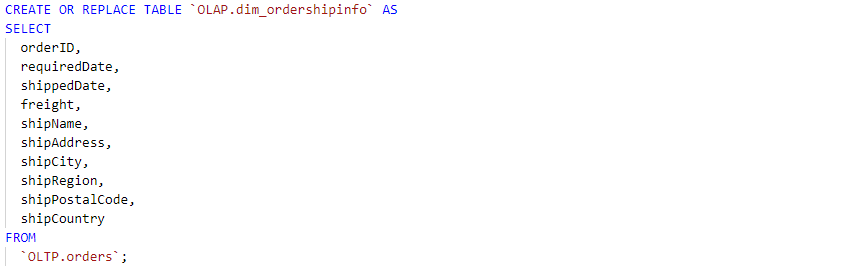
*Figure 20: dim\_date table creation*

**

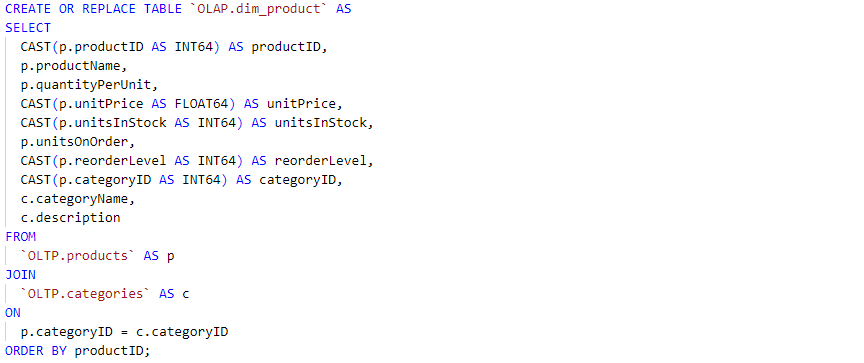
*Figure 21: dim\_customer table creation*

**

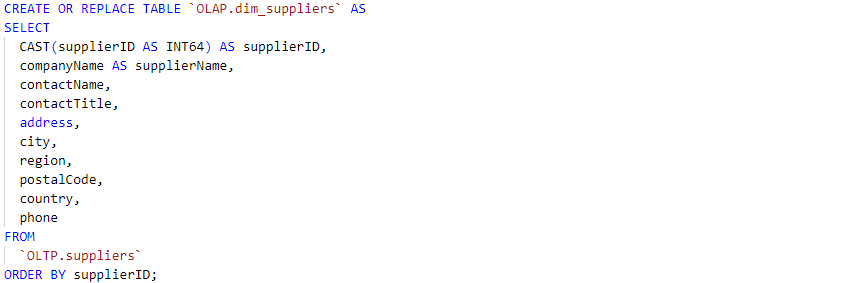
*Figure 22: dim\_employee table creation*

**

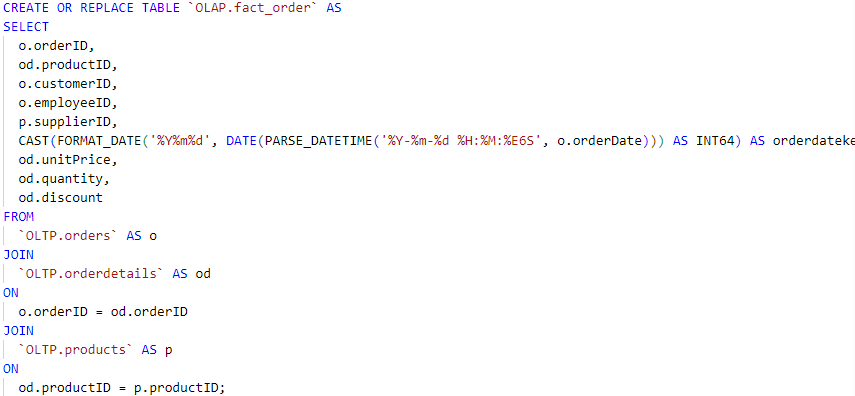
*Figure 23: dim\_ordershipinfo table creation*

**

*Figure 24: dim\_product table creation*

**

*Figure 25: dim\_suppliers table creation*

**

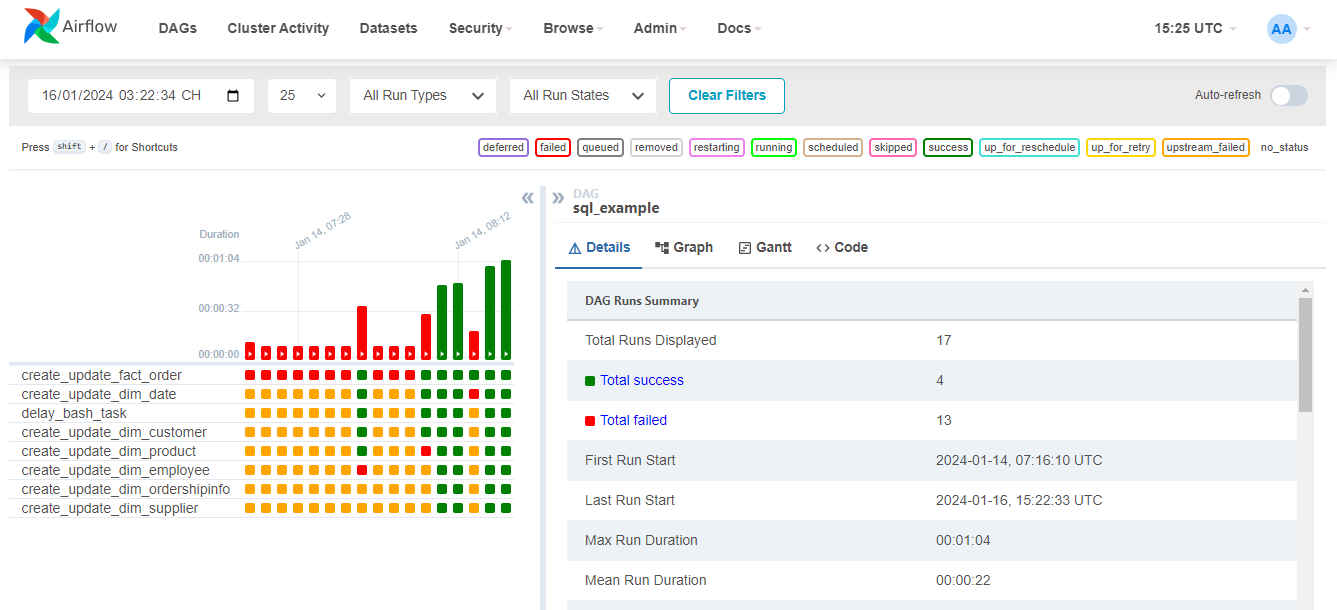
*Figure 26: fact\_order table creation*

In terms of operators used, the script predominantly utilizes the BigQueryOperator for table creation and updates in BigQuery. Each of these operators is uniquely identified by a task\_id and is assigned an SQL query from the aforementioned files. The parameter use\_legacy\_sql=False signifies the employment of the standard SQL syntax compatible with BigQuery. Additionally, the script incorporates a BashOperator named TaskDelay, designed to introduce a deliberate five-second pause in the workflow.

The script meticulously outlines the relationships and dependencies among various tasks, labeled t1 to t7, with each representing a distinct table operation via the BigQueryOperator. The workflow is structured such that task t1 (create\_update\_fact\_order) precedes t2 (create\_update\_dim\_date), followed by the TaskDelay. Post this delay, the sequence proceeds with tasks t3 through t7, each corresponding to different table operations. Notably, the workflow also demonstrates a branching pattern where both tasks t1 and t2 converge into task t4 (create\_update\_dim\_product).

Concludingly, the script is equipped with a conditional mechanism to invoke the DAG's command-line interface, but only if executed as the main program. This feature enriches the script with the capability for direct command-line interactions with the DAG, enhancing its utility and flexibility for various operational scenarios.

### **4. Airflow result**



*Figure 27: Airflow result*

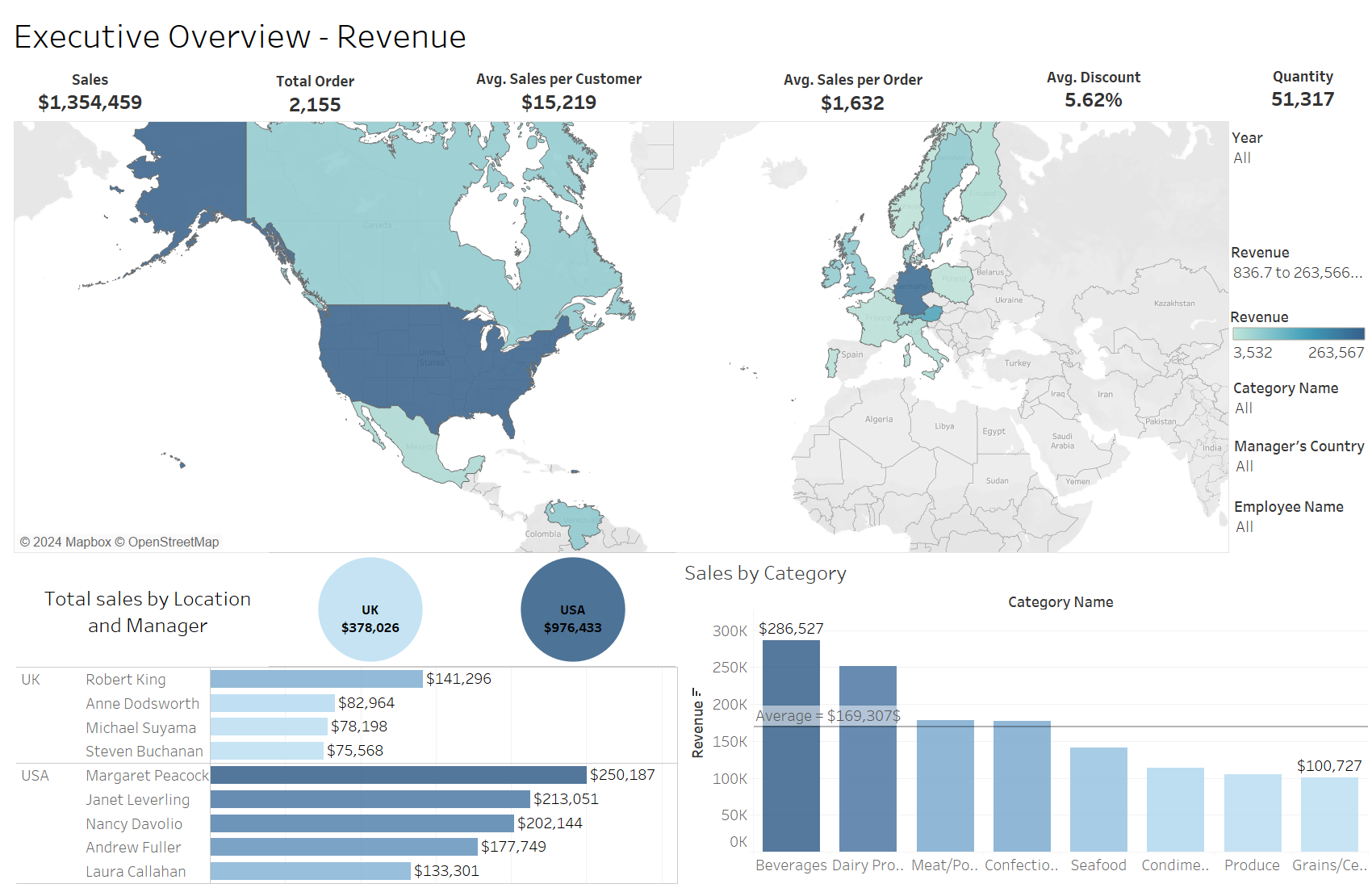
**

*Figure 28: Result on Google Big Query*

After 13 times of fixing error, we successfully automatically transform and load data to Google Big Query.

# **III. Visualization**

## A. Revenue Dashboard



*Figure 29: Revenue Dashboard*

The "Executive Overview - Revenue" dashboard exemplifies a meticulously designed business intelligence tool, providing a strategic high-level summary of an organization's sales performance. It showcases a robust total sales figure of $1,354,459, which, when combined with a significant order count of 2,155, suggests a thriving sales operation with substantial market engagement. The dashboard's intuitive design allows for immediate recognition of key metrics, making it an indispensable tool for executives.

Delving deeper into customer engagement, the dashboard reveals an average sales value per customer of $15,219. This metric is particularly telling, as it underscores the customers' high value to the business, potentially indicative of a premium product portfolio or a successful upmarket strategy. Coupled with this is the average sales per order of $1,632, suggesting that each transaction is of considerable worth, thereby highlighting efficient sales processes and the effectiveness of the sales team in delivering value.

The geographical representation of sales data is both intuitive and insightful, with a color-coded map that indicates varying levels of revenue generation across regions. The strategic emphasis on the United States and the United Kingdom, as evidenced by their darker hues, signals their critical role in the revenue composition. Such geographic insights are paramount for executives as they allocate resources and tailor regional strategies to leverage these key markets.

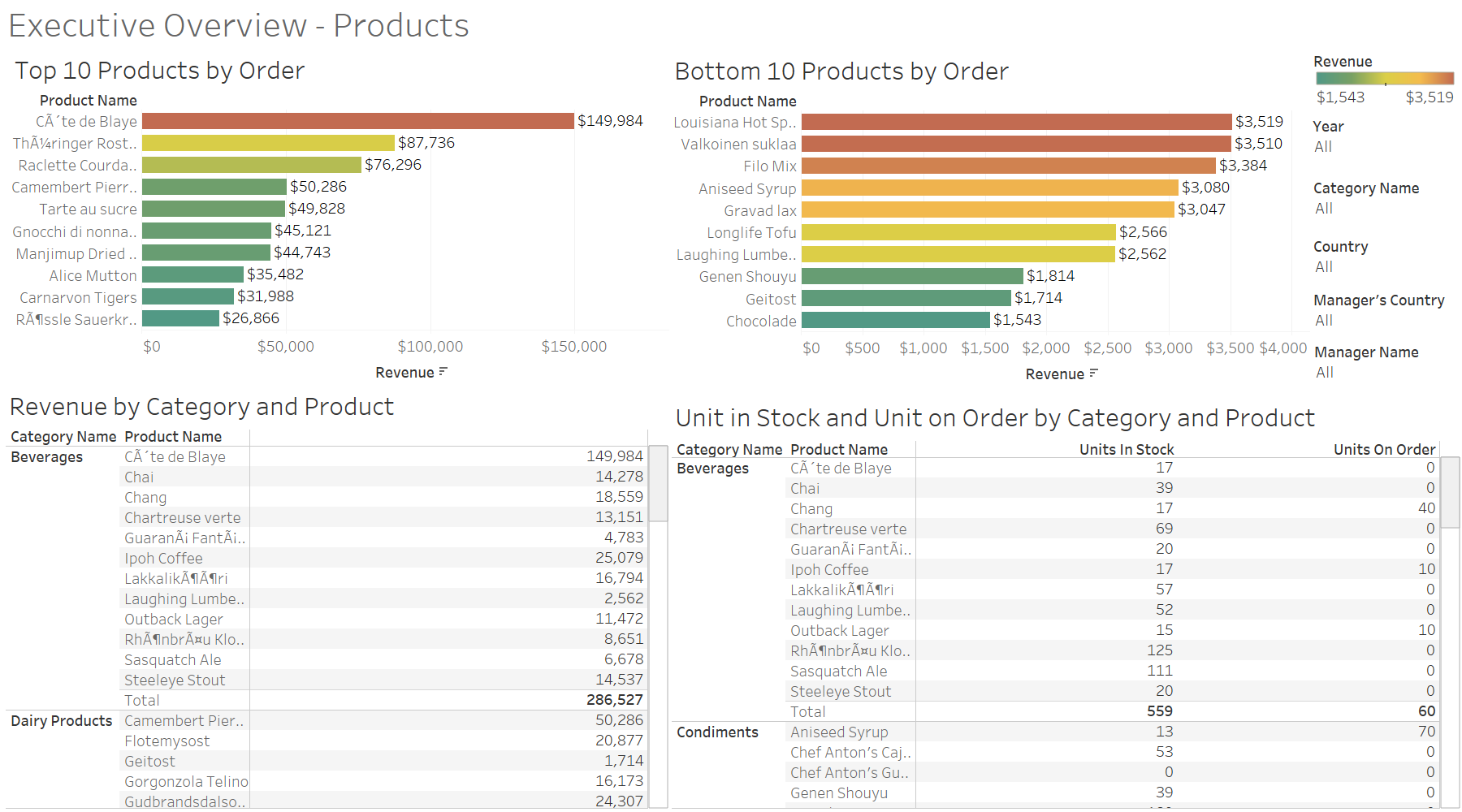
Complementing the geographical data, the dashboard presents a detailed breakdown of sales by location and manager through a pie chart and accompanying list. This segment effectively identifies managerial impact on revenue, recognizing the contributions of individual managers like Robert King in the UK and Andrew Fuller in the USA, who lead in sales figures. This granular view into managerial performance not only facilitates targeted managerial feedback but also serves as a motivational tool to foster a competitive sales environment.

The sales by category bar chart further refines the dashboard's analytical prowess. With Beverages leading the revenue generation, followed closely by other categories, stakeholders can quickly ascertain product line success and make informed decisions regarding product development, marketing strategies, and inventory management. The provided average revenue across categories benchmarks performance, allowing for a swift assessment of which categories surpass or fall behind the corporate average.

Lastly, the average discount rate of 5.62% offers a lens into the pricing strategy employed. It reflects the company's discounting tactics, balancing the need to incentivize customers while preserving profit margins. The synthesis of this with the average sales per order offers a holistic view of the company's pricing effectiveness.

Overall, the dashboard is not just a collection of data points but a narrative tool that tells the story of the company's financial health and market presence. Its detailed yet clear visualization supports a rapid absorption of information, while the interactive elements invite a deeper exploration into the data for strategic planning. This level of detail and accessibility ensures that executives are well-equipped with actionable insights to drive business growth and respond to market dynamics.

## B. Products Dashboard

**

*Figure 30: Product Dashboard*

The executive dashboard provides a meticulous and detailed view of the company's product lineup, serving as an essential tool for strategic decision-making by showcasing sales and inventory data. The "Top 10 Products by Order" graphically represents the sales triumphs within the product range, with "Côte de Blaye" leading the pack, commanding a revenue that towers at nearly $150,000, a testament to its market demand and popularity. On the opposite end of the spectrum, "Räßle Sauerkraut" rounds out this list, highlighting the varied performance across the company’s portfolio.

A further layer of insight is offered by the "Bottom 10 Products by Order," where products like "Louisiana Hot Spiced Okra" and "Chocolate" linger at the lower end of the revenue scale, barely surpassing the $1,500 mark. This stark differential in sales underscores the necessity for a nuanced analysis to understand market trends, consumer preferences, and potential areas for product development or marketing improvements.

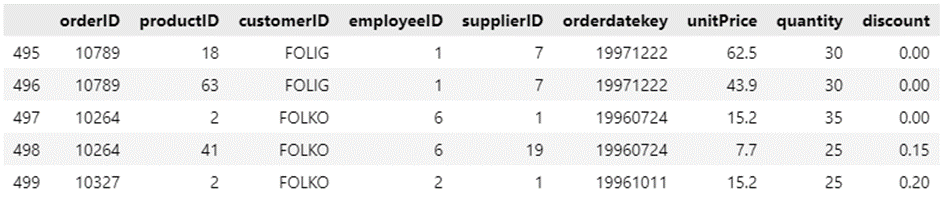
Diving deeper into category specifics, the "Revenue by Category and Product" section breaks down the sales figures within distinct product categories. In "Beverages," the dominance of "Côte de Blaye" is pronounced, outstripping other offerings such as "Chai" and "Chang" by a substantial margin, suggesting a focused consumer base or a premium product positioning. The dairy section paints a similar picture, with "Camembert Pierrot" leading in revenue, indicating a strong market presence.

The dashboard's "Unit in Stock and Unit on Order by Category and Product" offers a granular view into the company's inventory health. It reveals not just the current stock levels but also anticipates future demand through units on order. For instance, the inventory management for "Chartreuse verte" appears robust, with a significant quantity on order to replenish the stock, suggesting an expectation of sustained or increasing demand.

Moreover, this comprehensive overview is unfiltered, encompassing data from all available categories, timeframes, countries, and management zones, providing an aggregate view that is invaluable for executives. Such a broad perspective allows for the identification of overarching trends and performance benchmarks that are crucial for informed decision-making. The dashboard enables a clear understanding of which products are the heavy hitters in terms of revenue, which ones are lagging, and how inventory levels align with these trends, offering a strategic vantage point from which to drive the company forward.

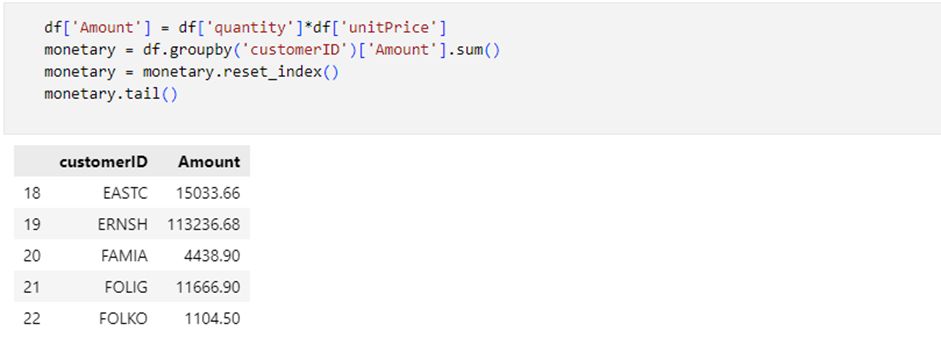
# **IV. Data minin**g

Here is our data from the csv file:



*Figure 31: Dataset*

We create df 'amount' by multiplying the value of df 'quantity' with the value of 'unitPrice' then we calculate the sum of column 'amount' based on the value of column 'customerID’ for monetary variable.



*Figure 32: Preprocessing*

Similar to frequency variable but we count based on df 'orderID'. We use the subtraction formula max(df 'orderdatekey') - df 'orderdatekey' applied to calculate the time interval between the latest date and each date in the 'orderdatekey' column. The result is a new column named 'Diff' containing the time intervals from the most recent date. Similar to the above two variables, we create a recency variable and calculate the minimum value of the 'Diff' column for each group.





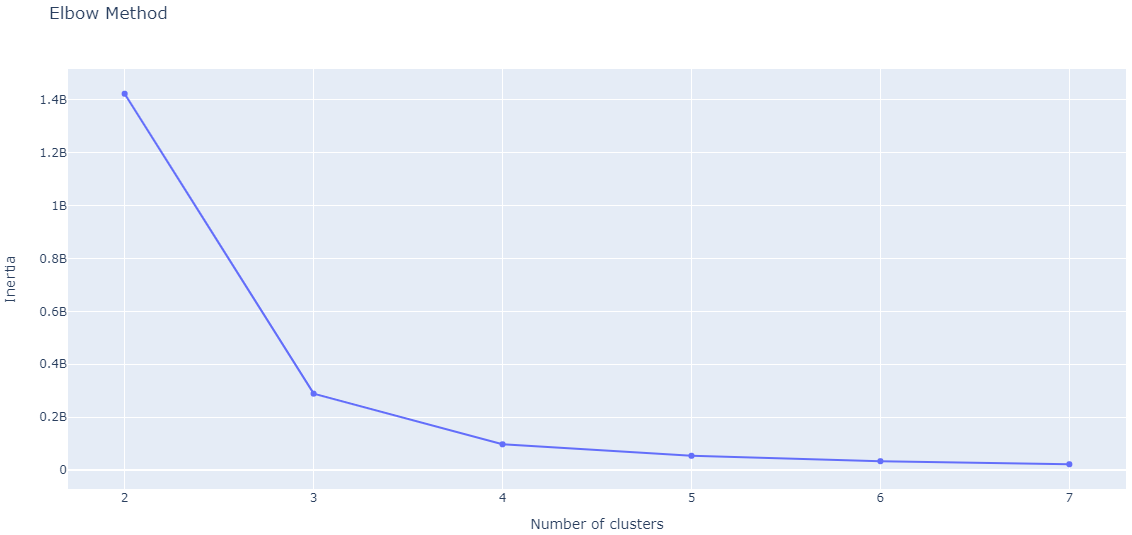
*Figure 33: Preprocessing*

Then combine the 3 variables into one table.



*Figure 34: RFM model*

We can plot the number of clusters on the x-axis and their respective scores on the y-axis to observe the trend. By using the elbow method, we can identify the optimal number of clusters based on the shape of the graph. In this case, dividing the data frame into 3 clusters seems to provide appropriate results as it likely corresponds to the "elbow" or significant change point in the graph.

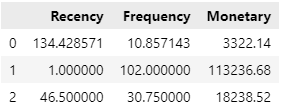


*Figure 36: Elbow method*

We then find the centroids using the kmeans.cluster\_centers\_ function. We initialize a for loop to iterate over all cluster centers in the centroids variable. The variable i will contain the index of the cluster center and the variable centroid will contain the coordinates (values ​​of the variables) of the corresponding cluster center. Finally initiate a for loop to iterate over all cluster labels in the labels variable. The variable i will contain the index of the cluster label and the variable label will contain the corresponding cluster label of each data point. Below is our final table of customer clustering based on data frequency and recency. The customer data has been divided into 3 labels.



*Figure 37: Clustering result*

**

*Figure 38: Centroid of each cluster*

Based on the provided data, we can analyze and categorize customers into different groups based on their Recency, Frequency, Monetary values, and assigned clusters.

***Cluster 0*** represents customers who have recently interacted with the business. They have low to medium frequency of transactions, typically ranging from 1 to 52, and their monetary value falls within the low to medium range, typically between 100 and 30,000. This group may consist of new or less active customers who haven't made significant purchases or spent substantial amounts. Cluster 0 has a centroid with a Recency of approximately 134.43, Frequency of 10.86, and Monetary value of 3322.14.

***Cluster 1*** includes a single customer who stands out from the rest. This customer has a low recency, indicating a recent transaction, and low frequency, representing only one transaction. However, the monetary value associated with this customer is remarkably high at 113,236.68. This suggests that this customer may be a special or VIP client who made a substantial purchase. Cluster 1 has a centroid with a Recency of 1.00 (which suggests very recent interaction), a very high Frequency of 102.00, and a very high Monetary value of 113236.68.

***Cluster 2*** comprises customers with a higher recency, indicating a longer time since their most recent transaction. They exhibit high to very high transaction frequency, ranging from 26 to 657, and their monetary values are also very high, typically falling between 7,500 and 22,607.70. This group likely consists of loyal and valuable customers who frequently engage with the business, making significant purchases. Cluster 2 has a centroid with a Recency of 46.50, a higher Frequency of 30.75, and a higher Monetary value of 18238.52 compared to Cluster 0.

We can also assess the accuracy of the k-means clustering by using the Separation method, which measures the degree of dispersion between clusters. When the clusters are well separated, the distance between them will be large, and conversely, if the clusters are not well separated, the distance will be small.

Centroid 1: [134.428571, 10.857143, 3322.14]

Centroid 2: [1.000000, 102.000000, 113236.68]

Centroid 3: [46.500000, 30.750000, 18238.52]

Euclidean distance between Centroid 1 and Centroid 2:

Euclidean distance: sqrt((134.428571 - 1.000000)^2 + (10.857143 - 102.000000)^2 + (3322.14 - 113236.68)^2) = 113112.207425

The distance between Centroid 1 and Centroid 2 is 113112.207425, indicating a high level of dispersion between these two clusters.

Euclidean distance between Centroid 1 and Centroid 3:

Euclidean distance: sqrt((134.428571 - 46.500000)^2 + (10.857143 - 30.750000)^2 + (3322.14 - 18238.52)^2) = 18258.592772

The distance between Centroid 1 and Centroid 3 is 18258.592772, also indicating a high level of dispersion between these two clusters.

Euclidean distance between Centroid 2 and Centroid 3:

Euclidean distance: sqrt((1.000000 - 46.500000)^2 + (102.000000 - 30.750000)^2 + (113236.68 - 18238.52)^2) = 113141.215202

The distance between Centroid 2 and Centroid 3 is 113141.215202, which is similar to the distance between Centroid 1 and Centroid 2.

In evaluating the performance of the k-means clustering algorithm using the Separation method, we find evidence of distinct clustering. The computed Euclidean distances among the centroids of the clusters are substantial, which is indicative of well-separated clusters.

Specifically, the distance between Centroids 1 and 2 is approximately 113,112.21, and the distance between Centroids 1 and 3 is about 18,258.59. The similarity of these magnitudes, especially the distance between Centroids 2 and 3, which is around 113,141.22, suggests that the algorithm has done a commendable job in distinguishing the clusters based on the given data points.

A high degree of dispersion, as indicated by these large distances, generally implies that the clustering algorithm has effectively partitioned the data space such that the inter-cluster variance is maximized. In other words, each cluster is distinct and separate from the others, which is the desired outcome for k-means clustering.

Therefore, the results suggest that the k-means clustering has performed well, yielding clusters that are well-separated in the feature space. This outcome should be seen as a positive indicator of the clustering algorithm's effectiveness for this particular dataset.

# **V. Recommendation**

Embark on your path towards a customer-centric and growth-driven approach by incorporating the following key recommendations:

### 1. Enhance the customer experience

To enhance the customer experience, it's crucial to invest in comprehensive staff training programs that equip your team with the knowledge and skills to provide exceptional service. Additionally, offer multiple channels for customer support, such as phone, email, live chat, and social media, to cater to customers' preferences. Personalize interactions by leveraging customer data to understand their preferences, purchase history, and demographics, allowing you to tailor your approach accordingly. Implementing a customer feedback system enables you to gather valuable insights and identify areas for improvement in your customer service processes.

### 2. Leverage social media and online platforms

Leveraging social media and online platforms is essential for reaching and engaging with your target audience. Identify the platforms where your audience is most active and create compelling, tailored content that resonates with them. Encourage user-generated content through contests, giveaways, or campaigns that incentivize customers to share their experiences and create content related to your brand. Timely and genuine responses to customer inquiries, comments, and reviews on social media demonstrate your commitment to customer engagement.

### 3. Implement a referral program

Implementing a referral program can significantly boost customer acquisition. Develop a structured program that rewards existing customers for successful referrals, such as offering discounts, exclusive offers, or loyalty points. Simplify the referral process by providing unique referral codes or links, and promote the program through various channels, including email marketing, social media, and your website.

### 4. Utilize email marketing

Email marketing remains a powerful tool for nurturing customer relationships. Segment your email list based on customer attributes to deliver targeted and relevant content. Personalize email subject lines and content to capture recipients' attention and make them feel valued. Provide exclusive offers, early access to promotions, or personalized recommendations based on customers' preferences and past interactions. Continually test different email formats and calls to action to optimize open rates, click-through rates, and conversions.

### 5. Collaborate with influencers or industry partners

Collaborating with influencers or industry partners can expand your brand's reach and credibility. Research and identify influencers or businesses that align with your brand values and have a strong presence in your industry. Co-create content, host joint events or webinars, or feature each other's products or services to leverage their reach and audience. Through these collaborations, you can introduce your brand to new audiences and gain their trust through the influencer's association.

### 6. Monitor and respond to customer feedback

Monitoring and responding to customer feedback is vital for maintaining customer satisfaction. Regularly gather feedback through surveys, social media listening tools, and online review platforms. Respond promptly and empathetically to both positive and negative feedback, demonstrating that you value customer opinions and are committed to their satisfaction. Use customer feedback to drive improvements in products, services, and experiences. Encourage customers to leave reviews and testimonials by simplifying the process and offering incentives where appropriate.

### 7. Continuously analyze and optimize marketing efforts

Continuously analyzing and optimizing your marketing efforts is key to driving success. Set clear goals and establish KPIs to measure campaign effectiveness. Use analytics tools to track and analyze data related to customer acquisition, conversion rates, customer lifetime value, and ROI. Conduct A/B testing to compare different strategies, messages, and channels, identifying the most effective approaches. Regularly review and refine your marketing strategies based on data and insights.

# **VI. Conclusion**

The conclusion of this report emphasizes the transformative impact of data warehousing and analytics in modern business environments. It highlights the successful integration of technologies such as Apache NiFi, Docker, and Airflow, which facilitated efficient data processing and management. The analysis presented demonstrates how data-driven strategies can significantly improve customer understanding, market penetration, and operational efficiency. The report suggests that businesses adopting these methodologies can expect enhanced decision-making capabilities, with a marked increase in agility and competitiveness. Additionally, the conclusion underlines the importance of continuous innovation in data analytics to stay ahead in rapidly changing market dynamics. The findings also advocate for the alignment of data strategies with business objectives to maximize ROI and foster sustainable growth. Lastly, it suggests areas for future research and development, particularly in the realm of predictive analytics and AI, to further capitalize on the untapped potential of data in business strategy and operations.

# 

# **References**

1. Balasubramanian, M. (2018, May 3). *Dimensional Modelling & Visualization of North wind database*. Medium. Retrieved January 17, 2024, from https://medium.com/@kmsbmadhan/dimensional-modelling-visualization-of-northwind-database-beaac7fecb20
2. Chandana, D. (2021, May 6). *Exploring Customers Segmentation With RFM Analysis and K-Means Clustering*. Medium. Retrieved January 17, 2024, from https://medium.com/web-mining-is688-spring-2021/exploring-customers-segmentation-with-rfm-analysis-and-k-means-clustering-118f9ffcd9f0
3. *. . - definition of . . by The Free Dictionary*. (n.d.). The Free Dictionary. Retrieved January 17, 2024, from https://link.springer.com/chapter/10.1007/978-3-662-65167-4\_7
4. *GetFile*. (n.d.). Apache NiFi. Retrieved January 14, 2024, from https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-standard-nar/1.9.2/org.apache.nifi.processors.standard.GetFile/index.html
5. *GetFile*. (n.d.). Apache NiFi. Retrieved January 17, 2024, from https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-standard-nar/1.9.2/org.apache.nifi.processors.standard.GetFile/index.html
6. Jain, S. (2023, June 10). *Dimensional Data Modeling*. GeeksforGeeks. Retrieved January 14, 2024, from https://www.geeksforgeeks.org/dimensional-data-modeling/
7. Jain, S. (2023, June 10). *Dimensional Data Modeling*. GeeksforGeeks. Retrieved January 17, 2024, from https://www.geeksforgeeks.org/dimensional-data-modeling/
8. *PutBigQueryBatch*. (n.d.). Apache NiFi. Retrieved January 17, 2024, from https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-gcp-nar/1.19.1/org.apache.nifi.processors.gcp.bigquery.PutBigQueryBatch/index.html
9. *UpdateAttribute*. (n.d.). Apache NiFi. Retrieved January 17, 2024, from https://nifi.apache.org/docs/nifi-docs/components/org.apache.nifi/nifi-update-attribute-nar/1.5.0/org.apache.nifi.processors.attributes.UpdateAttribute/additionalDetails.html
10. The Data Warehouse Lifecycle Toolkit, Ralph Kimball. Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, Bob Becker. John Wiley & Sons, 2011