**Data Engineer Internship Report**

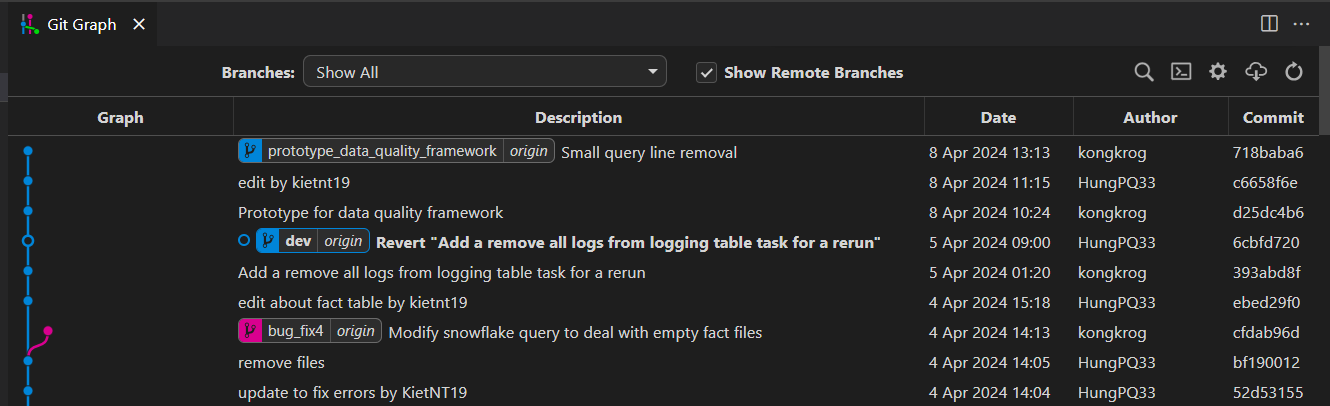
**Sample database**: WideWorldImporters  
**Mentor**: HungDM16, LinhTM9  
**Done by**: Spring24 Batch Data Engineer Intern Members  
**Date**: 16/01/2024 - 28/04/2024

**Table of contents**

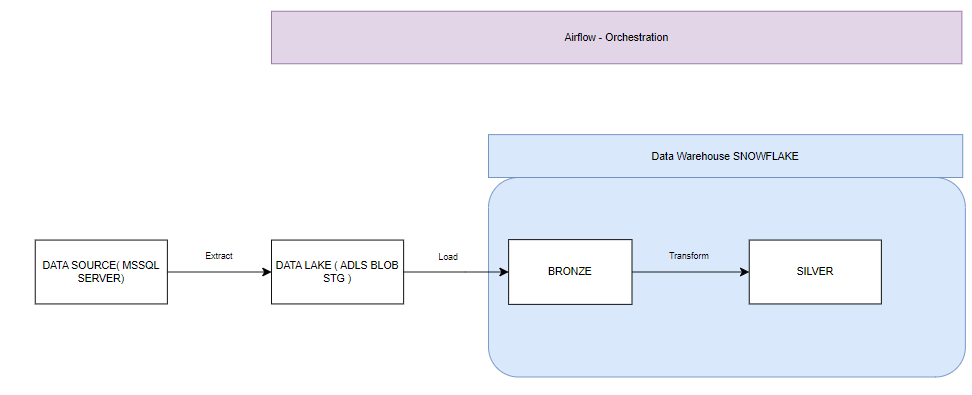
1. Executive Summary
2. Project Objectives
3. Database Introduction (On-prem)
4. Git/ Source Code
5. Data Architecture
6. Techniques  
   6.1. SCD Type 2  
   6.2. PoC Extract Source Data into Lake Data  
   6.3. Extract Source Data into Data Lake  
   6.4. PoC Load Lake Data into Data Warehouse  
   6.5. Load Data Lake into Data Warehouse  
   6.6. PoC Transform Bronze Data into Silver Data  
   6.7. Transform Bronze Data into Silver Data  
   6.8. PoC Master Pipeline  
   6.9. Runing Master Pipeline   
   6.10. Monitoring Framework  
   6.11. PoC Data Quality Framework
7. Systems Analysis and Design  
   7.1. Movement Module  
   7.2. Orders Module  
   7.3. Purchase Module  
   7.4. Sale Module  
   7.5. Stockholding Module  
   7.6. Transaction Module
8. Implementation on Astronomer
9. PowerBi Report  
   9.1. Movement Module  
   9.2. Orders Module  
   9.3. Purchase Module  
   9.4. Sale Module  
   9.5. Stockholding Module  
   9.6. Transaction Module
10. Review and Conclusion  
    10.1. Review  
    10.2. Conclusion

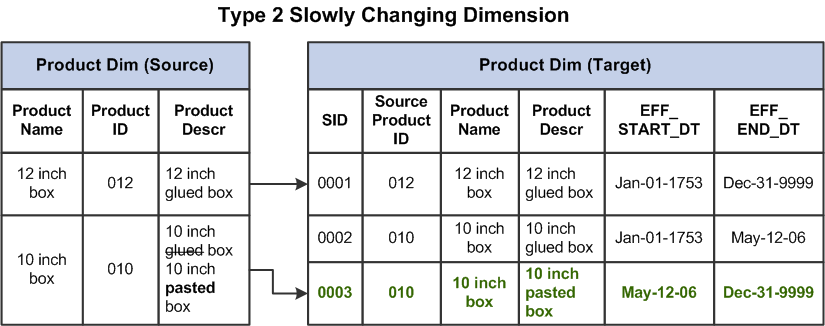
**1. Executive Summary**  
 This is a Data Engineer project that has been done by On-the-Job-Training (OJT) students from Quy Nhon AI FPT University. It took about 3 months to complete (from January to April 2024). The WideWorldImporters dataset is its sample database.

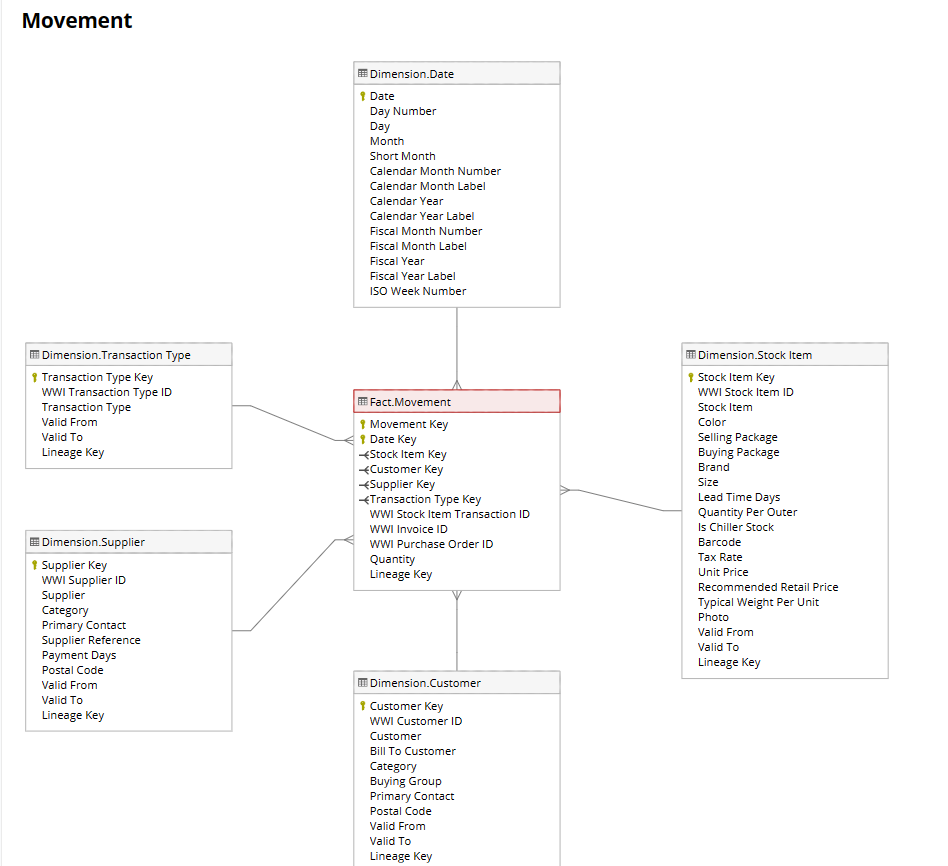
**2. Database Introduction**  
 The WideWorldImporters dataset is a comprehensive collection of data relating to a global import/export company. It encompasses various aspects such as sales, customers, products, and suppliers, providing valuable insights for analysis and decision-making. With its structured format and diverse information, the dataset serves as a valuable resource for studying business dynamics and conducting data-driven research.

**3. Project Objectives**  
 Through this project, we can gain:  
- Understanding the pipeline of a Data Engineer project through a sample database:   
 + Analysis on-prem system  
 + Extract source data into data lake  
 + Load data lake into data warehouse  
 + Transform bronze data into silver data  
 + Master pipeline, monitor framework, data quality framework,...  
- Know how to analysis data by using PowerBi to understand domain knowledge  
- Soft skills: Teamwork, Task Split, Task Management,...  
- Experience in the workspace environment  
  
**4. Git/Source Code**  
 Source code from all members was pushed in their branches with same templates. They would be checked by team leaders and mentors. If right, they would be merged into the dev branch. After the project has been completed, all code will be merged into the master branch.  
   
 *Image 1: Source Code on Git*

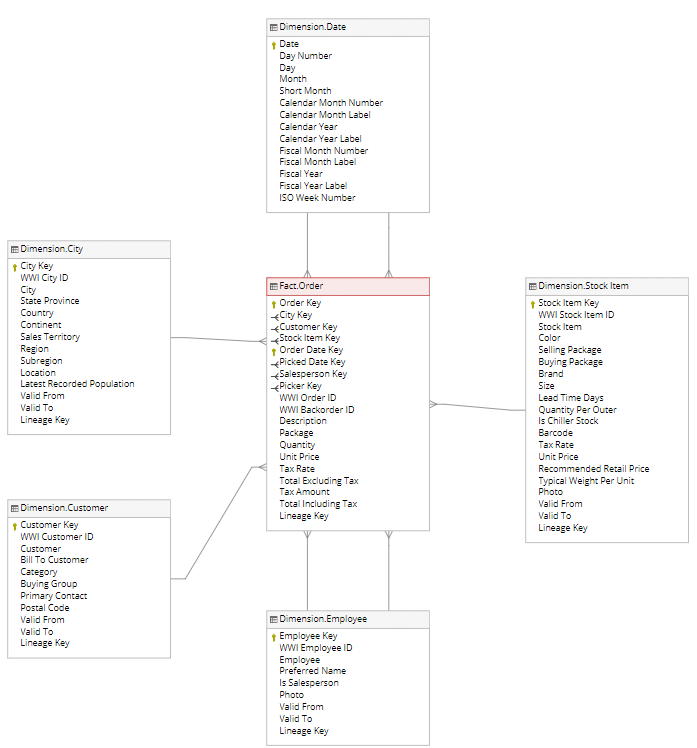
**5. Data Architecture**   
 Source data was saved in MSSQL Server and would be ingested by Airflow into ADLS (Azure Data Lake Storage) Blob Storage as CSV files. The data would be loaded into Snowflake Data Warehouse for transformation called bronze data. Transformation phase applied by using Snowflake SQL Store Procedures and after transformed, we will have a Star Schema Data Model Architecture called silver data. All processes will be automated by pipelines and all pipelined will be orchestrated by Airflow.

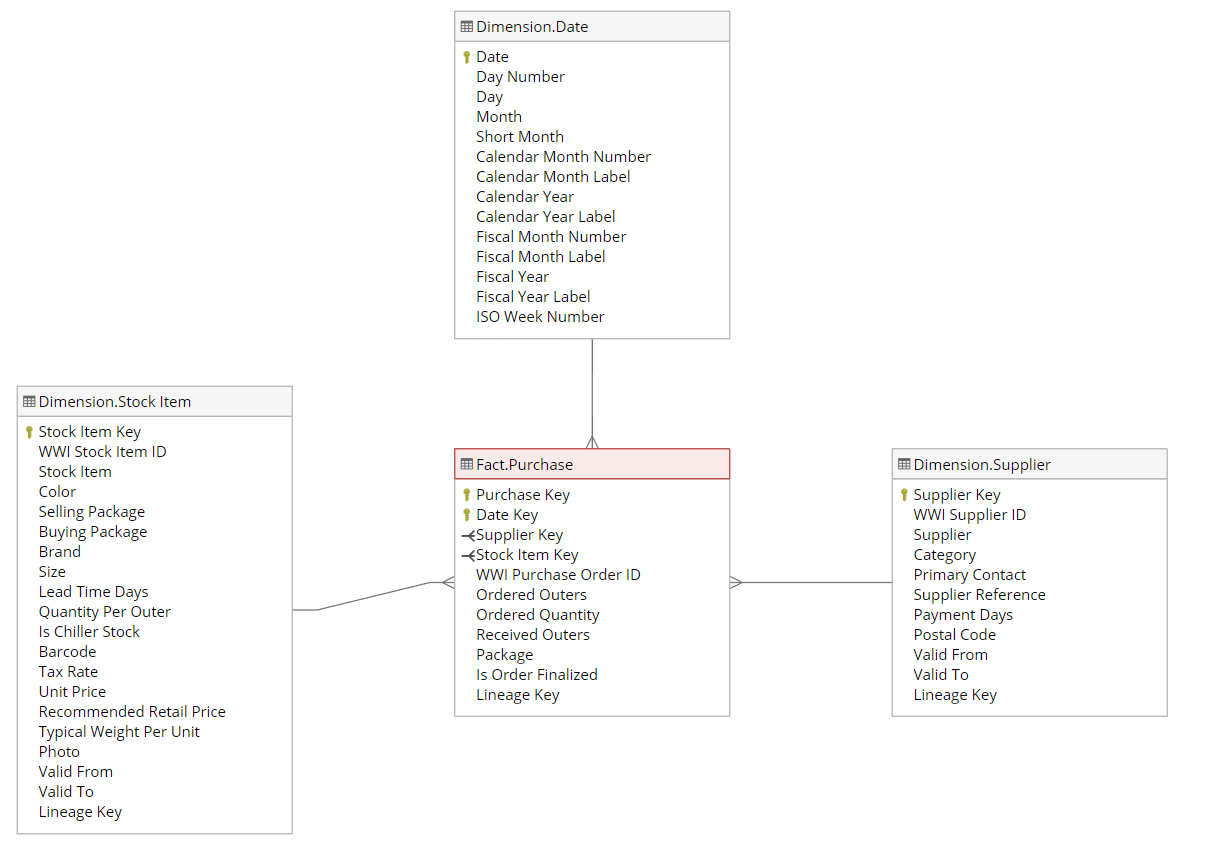
*Image 2: Data Architecture*

**6. Techniques**  
 6.1. SCD Type 2  
  
 *Image 3: SCD Type 2*  
  
 **SCD Type 2**, or Slowly Changing Dimension Type 2, in data engineering refers to a method of structuring databases to track historical changes in data over time. It involves creating new records for each change to a dimension, such as a product or customer, while maintaining links to previous versions. This allows analysts to analyze historical trends and perform accurate reporting without losing the integrity of the data.  
  
 6.2. PoC Extract Source Data into Lake Data  
 Since Airflow does not support Azure Blob Storage HTTPS links, only WASB connection, a python library named ”azure-storage-blob" is installed. The library deals with connection to the specify blob storage, meanwhile Airflow’s MssqlHook deals with getting data from the local SQL Server Database. The data frame/table is then passed onto a memory object and is read and uploaded directly into the storage, bypassing any external I/O operations.  
  
 6.3. Extract Source Data into Data Lake  
 After building PoC Extract Source Data into Data Lake, there was a .py file and .sql file to run to extract data in every table.  
 This .py file called all necessary functions condensed in a class that were built in PoC Extract source data into data lake, table name, .sql file, all dag command.  
 And .sql file contained SQL query that query all this table ‘s data. In SQL query, it use ”union all” to combine source data with archived data. From that, all data would be extracted into data lake.  
   
 6.4. PoC Load Lake Data into Data Warehouse  
 The loading process is handled in the data warehouse itself using SnowSQL by using Airflow’s SnowflakeOperator. A temporary table is created by extracting the file’s header since STAGED FILE in Snowflake does not contain any header. Then the data is loaded into the temp table and into the final table. There are 3 methods: An update method using MERGE INTO, a REPLACE method using TRUNCATE, an INSERT method using INSERT INTO.  
  
 6.5. Load Data Lake into Data Warehouse  
 After building PoC Load Lake Data into Data Warehouse, there was a .py file and .json file to run to load data in every table.  
 This .py file called all necessary functions that were built in PoC Load Lake data into data warehouse, lake table name, target table name, module name, target primary key, upload method, .json file, all dag command.  
 And .json file contained mapping between lake table’s column name and target table’s column name. From that, all data would be loaded into data warehouse in column respectively.  
  
 6.6. PoC Transform Bronze Data into Silver Data  
 Since transforming data only involves writing SQL queries, Airflow’s SnowflakeOperator does the job just fine.   
  
 6.7. Transform Bronze Data into Silver Data  
 After building PoC Transform bronze data into silver data, there was a .py file and .sql file to run to transform bronze data into silver data.  
 This .py file called all necessary functions that were built in PoC Transform data, table target, .sql file name, all dag command.  
 And .sql file contained SQL queries that use SQL joins, SCD Type 2,...  
  
 6.8. PoC Master Pipeline   
 By using TriggerDagRunOperator, a DAG can be triggered in another DAG. The master pipeline is separated into 3 parts (or more). The first part is to run all DAGs related to extracting data from Source into Lake, the second part is loading Lake into Bronze, the final part is for Transforming Bronze into Silver. All the related DAGs are put into a list, and there’s a checkpoint to make sure that all DAGs are successful before moving on to the next part.  
  
 6.9. Running Master Pipeline   
 After building PoC master pipeline, we need to have .py file to call all necessary functions, libraries, module name, all lists that consist of dag command, all dag commands.  
 Lists were split into 5 small lists:   
 - Lake list contains all dag name commands that run the task to extract source data into data lake.  
 - Bronze list contains all dag name commands that run the task to load lake data into data warehouse.  
 - Silver list contains all dag name commands that run the task to transform bronze data into silver data of dimension tables.  
 - Fact list contains all dag name commands that run the task to transform bronze data into silver data of fact tables.  
 - Task list contains all above lists in order.  
   
 6.10. Monitoring Framework  
 A PL\_LOGGING\_TABLE is made in the data warehouse; it’ll store all the result of all the DAGs that have been run so far. Functions are made to log the result, check if the DAGs succeed or fail, and insert them into the table using the SnowflakeOperator and pre-write SQL queries.  
  
 6.11. PoC Data Quality Framework  
 Data Quality Framework is made by creating procedures in Snowflake. 5 rules checking procedures and 5 rules verification functions are made and integrated, written in JAVASCRIPT. This is then simplified into Airflow by using functions with arguments into placeholders in pre-write SQL queries, using SnowflakeOperator.  
  
7. Systems Analysis and Design  
 7.1. Movement Module  
 The Movement Module involves 5 dimension tables: TransactionType, Customer, Stockitem, Date, supplier and a fact table named Movement which bring all dimension tables and 3 other tables from bronze tables together by their key and compare with appropriate time.

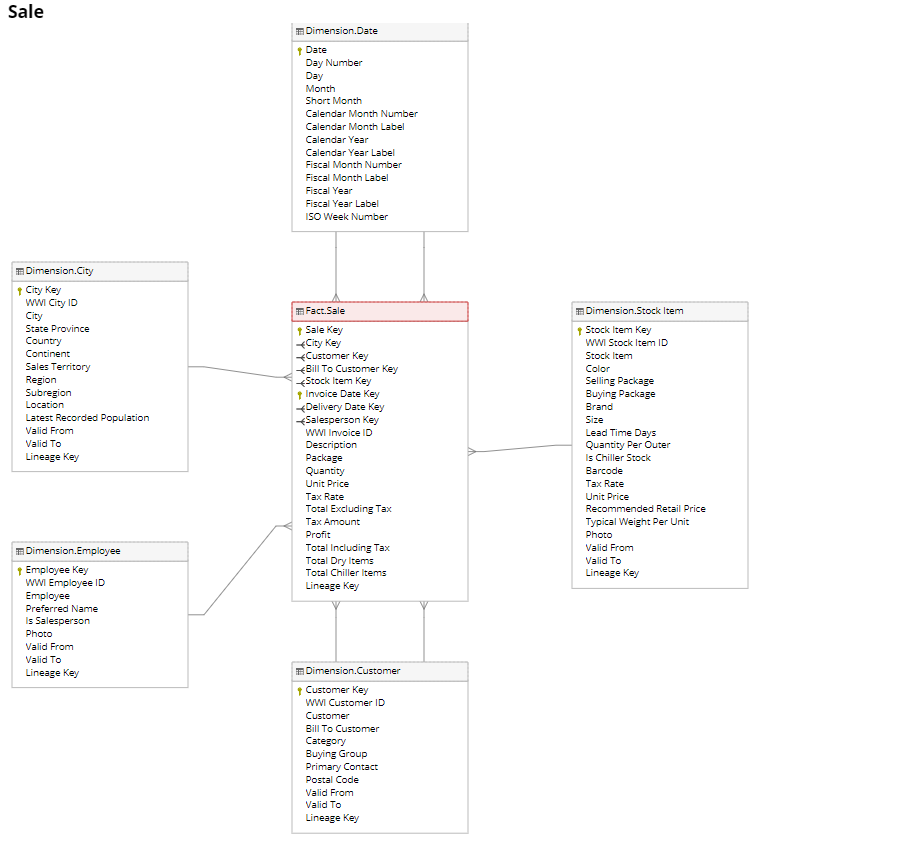
 *Image 4: Movement Module*  
   
 Dim.Supplier: Transformed from three bronze tables: People, Supplier, SupplierCategories. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.

Dim.StockItem: Transformed from three bronze data tables: StockItems, Colors, PackageTypes. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Dim.Date: Different from all dimension tables. This table was completely generated by using SQL query for timing.  
 Dim.Customer: Transformed from four bronze data tables: Customers, CustomerCategories, BuyingGroups and People. All of them were connected by ’left join’ in SQL and based on SCD type 2 rule to compare time (ValidFrom - ValidTo) among source data in each two tables respectively.  
 Fact. Movement: The "Fact Movement" table serves as a central repository for recording and analyzing the quantitative aspects of various business activities, such as inventory management: Stock Issue, means selling, Stock Receipt means buy-in process and Stock Adjustment. It provides valuable insights into the movement of goods and materials within the organization, facilitating performance analysis, trend identification, and decision-making processes.  
  
 7.2. Orders Module  
 The Orders Module involves 5 dimension Tables: City, Customer, Employee, Stockitem, Date and a fact table named Orders which bring all dimension tables and 3 other tables from bronze schema (Orders, OrderLines and PackageTypes) together by their key and compare with appropriate time.

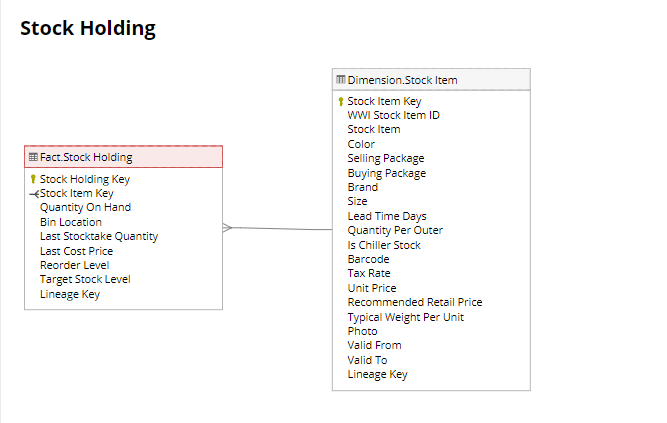
 *Image 5: Orders Module*  
  
 Dimension City: Transformed from three bronze data tables: Cities, StateProvinces and Countries. All of them were connected by ’left join’ in SQL and based on SCD type 2 rule to compare time (ValidFrom - ValidTo) among source data in each two tables, such as StateProvinces - Countries, then combine with Cities respectively.  
 Dimension Customer: Transformed from four bronze data tables: Customers, CustomerCategories, BuyingGroups and People. All of them were connected by ’left join’ in SQL and based on SCD type 2 rule to compare time (ValidFrom - ValidTo) among source data in each two tables, such as {[(Customers - CustomerCategories) - BuyingGroups] - Customers} then combine with People bronze table respectively.  
 Dimension Employee: From table bronze People and have a filter ‘IsEmployee’ was different from 0 to find employee.  
 Dimension Stockitems: Three bronze tables—StockItems, Colors, and PackageTypes—have guilted up the table. To determine the time (ValidFrom - ValidTo) between tables, use the SCD type 2 rule. With a left join, they were all linked.  
 Dimension Date: This table was constructed purely in terms of timing using the table produced by SQL queries.  
 Fact Orders: The City, Customer, Employee, StockItems, and Date tables comprise the four dimensions that make up the table. They used a left join to join the tables together and the SCD type 2 technique to compare the times of the tables.  
  
 7.3. Purchase Module  
 This module includes 3 dimension tables: Supplier, StockItem, Date and 1 fact table: Purchase

 *Image 6: Purchase Module*

Dim.Supplier: Transformed from three bronze tables: People, Supplier, SupplierCategories. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.

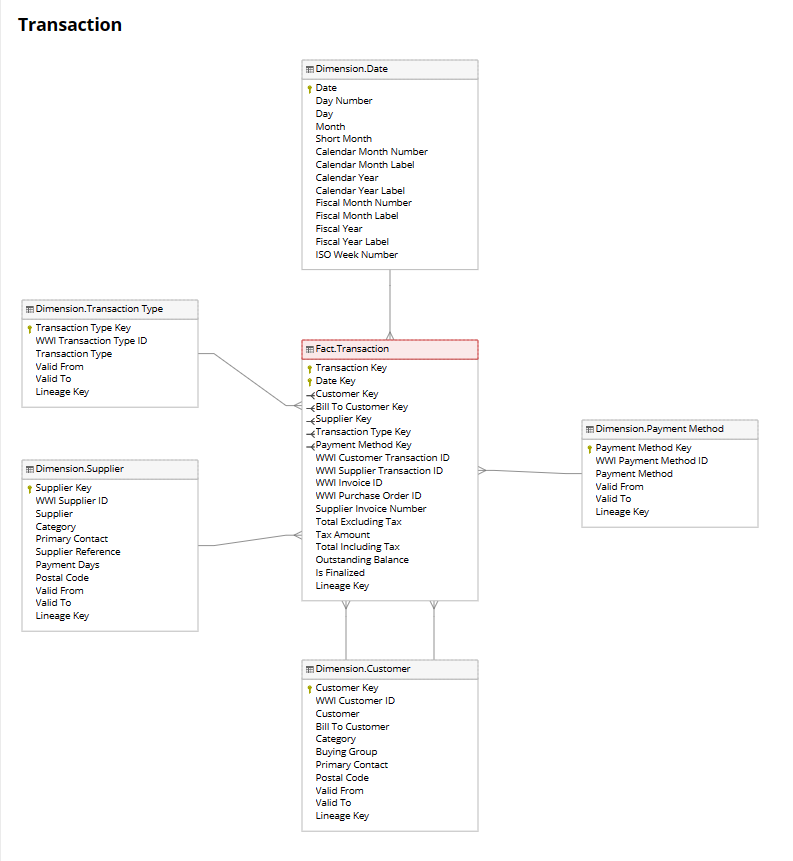
Dim.StockItem: Transformed from three bronze data tables: StockItems, Colors, PackageTypes. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Dim.Date: Different from all dimension tables. This table was completely generated by using SQL query for timing.  
 Fact.Purchase: Transformed in a special way by using bronze data tables: PurchaseOrders, PurchaseOrderLines and above dimension tables. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (LastEditedWhen - ValidFrom - ValidTo) among data in each two tables.  
  
 7.4. Sale Module   
 This module includes 5 dimension tables: Customer, Employee, City, StockItem, Date and 1 fact table: Sale   
  
  *Image 7: Sale Module*  
   
 Dim.Customer: Transformed from four bronze data tables: Customers, CustomerCategories, BuyingGroups, People. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Dim.Employee: Transformed directly from a bronze data table: People with condition that ”IsEmployee” column is different 0 (False).  
 Dim.City: Transformed from three bronze data tables: Cities, StateProvinces, Countries. They were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Dim.StockItem: Transformed from three bronze data tables: StockItems, Colors, PackageTypes. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Dim.Date: Different from all dimension tables. This table was completely generated by using SQL query for timing.   
 Fact.Sale: Transformed in a special way by using bronze data tables: Invoices, InvoiceLines and above dimension tables. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (LastEditedWhen - ValidFrom - ValidTo) among data in each two tables.  
  
 7.5. Stockholding Module

This module includes 1 dimension tables: StockItem and 1 fact table : Stockholding

 *Image 8: StockHolding Module*

The Stockholding Module within Data Warehouse utilizes a star schema design to efficiently store and analyze sales data which included 1 Dimension Tables : Dim.StockItem and 1 Fact Table is Stockholding which connected with all dimension tables by their key, each dimension table have different logic to mapping from Bronze schema, some may have the same with other modules, but Fact Tables mapping of each module is unique based on their particular purpose.  
 StockItem : Transformed from three bronze data tables: StockItems, Colors, PackageTypes. All they were connected by ’left join’ SQL join and SCD type 2 to compare time (ValidFrom - ValidTo) among source data in each two tables.  
 Stockholding: Transformed in a special way by using bronze data tables: Invoices, InvoiceLines and above dimension tables. They were connected by ’inter join’ SQL and direct

7.6. Transaction Module

 *Image 9: Transaction Module*

This module includes:  
 - Five dimension tables: Date, Customer, Supplier, Payment Method, Transaction Type.  
 - Fact table: Transaction.

Dim.Customer: Transformed from four bronze tables: Customers, CustomerCategories, BuyingGroups, People. Then they were connected by “left join” SQL join and SCD type 2 to compare time( VaidFrom – VaildTo) among source data in each two tables.

Dim.Payment Method: Transformed directly from a bronze data table: Payment Method.

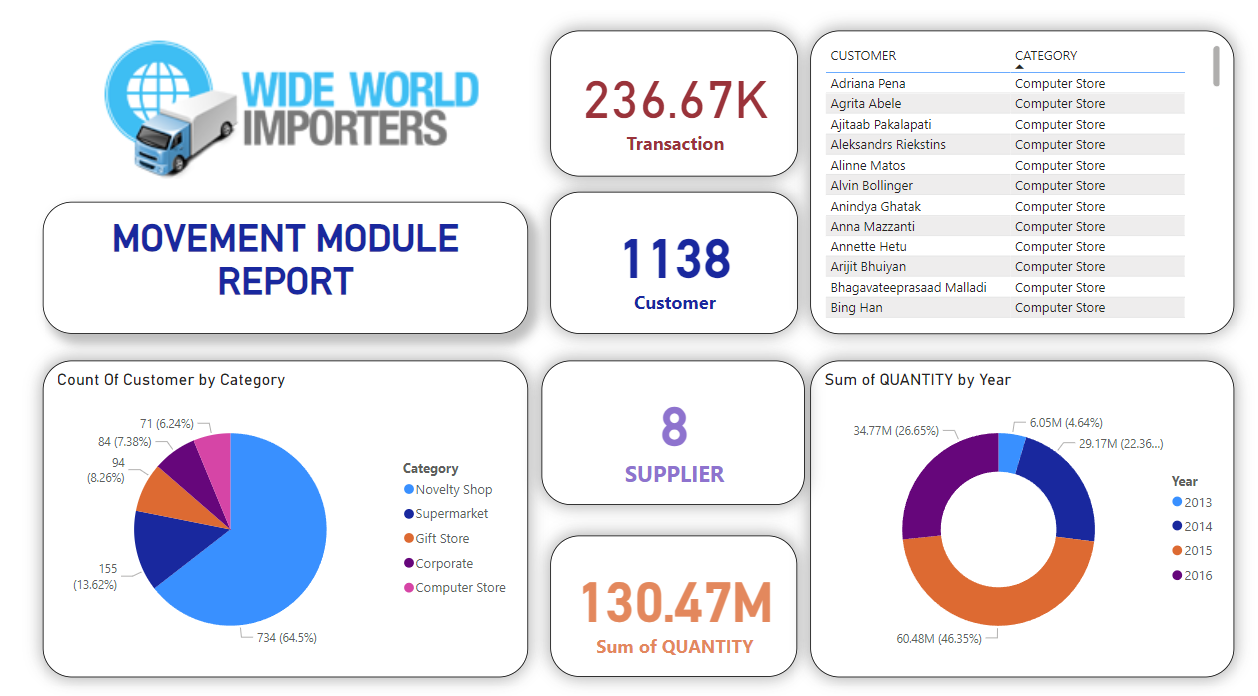
Dim.TransactionType: Transformed directly from a bronze data table: TransactionTypes.

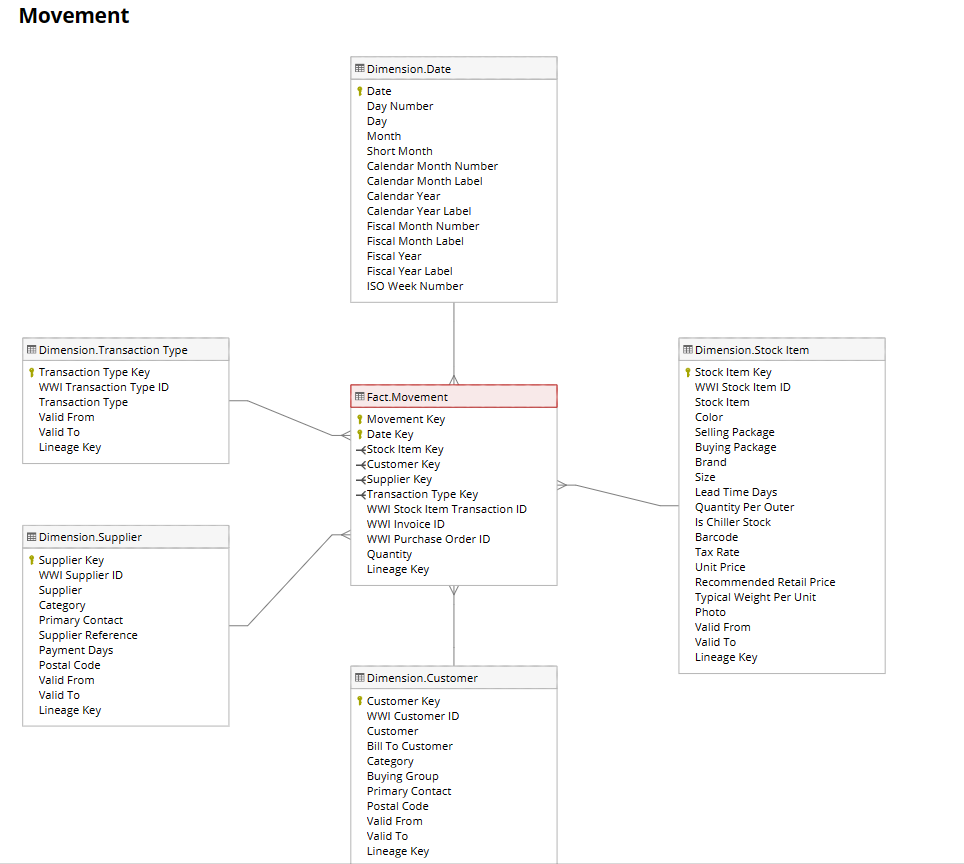
Dim.Date: Different from all dimension tables. This table was completely generated by using SQL query for timing. By using CTE, we generated a sequence of dates starting from “2013-01-01" up to 20 years from the current date. Then insert that sequence of dates into the Date column and from it, update other columns such as Month, Fiscal Month, ISO Week, ...

Dim.Supplier: Transformed from three bronze tables: Supplier, SupplierCategories and People. Then they were connected by “left join” SQL join and SCD type 2 to compare time (VaidFrom – ValidTo) among source data in each two tables

Fact.Transaction: Transformed in a special way by using bronze data tables: SupplierTransaction and CustomerTransaction and dimension tables: Date, Transaction Type, Payment Method, Customer, Supplier. Firstly, they were connected to two bronze tables using “union all” SQL join. Then, they were connected above the referring tables using their own "left join" SQL join, Transaction Type using "inner join" SQL join, and SCD type 2 to compare time( LastEditedWhen – ValidFrom – ValidTo) among data in each two tables.

**8. Implementation on Astronomer**  
 All tasks must be tested and run successfully on the local computer. And after that, all codes would be merged into dev branch from child branches, then they would be published to run in Astronomer.  
  
**9. PowerBi Report**  
 9.1. Movement Module  
 A movement report provides a detailed snapshot of the flow of goods and materials within a business during a specific timeframe. It encompasses various transactions, including purchases, sales, transfers, returns, and adjustments, allowing stakeholders to track the movement of inventory and understand its impact on operations.  
 The report typically includes essential details such as the date of each movement, the product or item involved, the type of movement (e.g., in, out, transfer), the quantity moved, and the associated cost or value. It may also capture information about customers, suppliers, warehouses, locations, employees involved, and transaction IDs.

 *Image 10: Movement Module Overview Report in PowerBi*



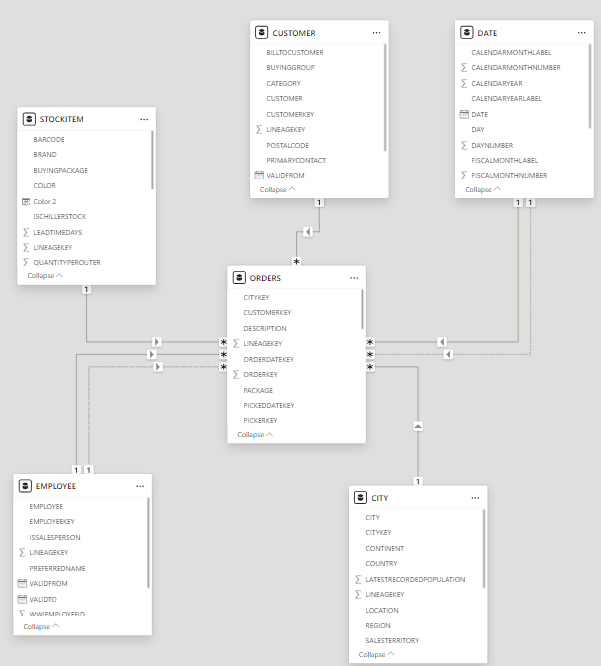
Visualizations such as bar charts, line charts, and pie charts are often used to present key insights, such as the distribution of movements by type, trends over time, and the proportion of movements by category. Additionally, key performance indicators (KPIs) such as average cost per movement, average quantity per movement, and inventory turnover ratio may be calculated to provide deeper insights into inventory management efficiency.  
 By analyzing the data presented in the movement report, stakeholders can gain valuable insights into inventory movement patterns, identify bottlenecks or inefficiencies in the supply chain, optimize inventory levels, improve customer service, and enhance overall business performance. This enables informed decision-making and strategic planning to drive growth and profitability.

9.2. Orders Module  
 This module focuses on everything about orders, including city, customer, employee, stock items which have been grouped into 3 big groups: employee, stock items and orders in area. Particularly, this report concentrates on the difference in amount of stock items, best employee by number of orders, and order quantity by region by year and quarter (from 2013 to quarter 2 of 2016). This information can help managers decide the quantity and stock items to import next time as well as the productivity bonus for excellent performance employees. It can be seen clearly in the report that the item ‘Black and orange fragile despatch tape 48mmx75m’ is highly favored and customers tend to pay things in high – priced group (>= 50$) in these years. The top 5 excellent performance order processors should have productivity bonus are archer, Kayla, Taj, Sophia and Hudson. Finally, the sales territory has the most orders in this time interval is Southeast with a population somewhere about 6929 people which can be found in.



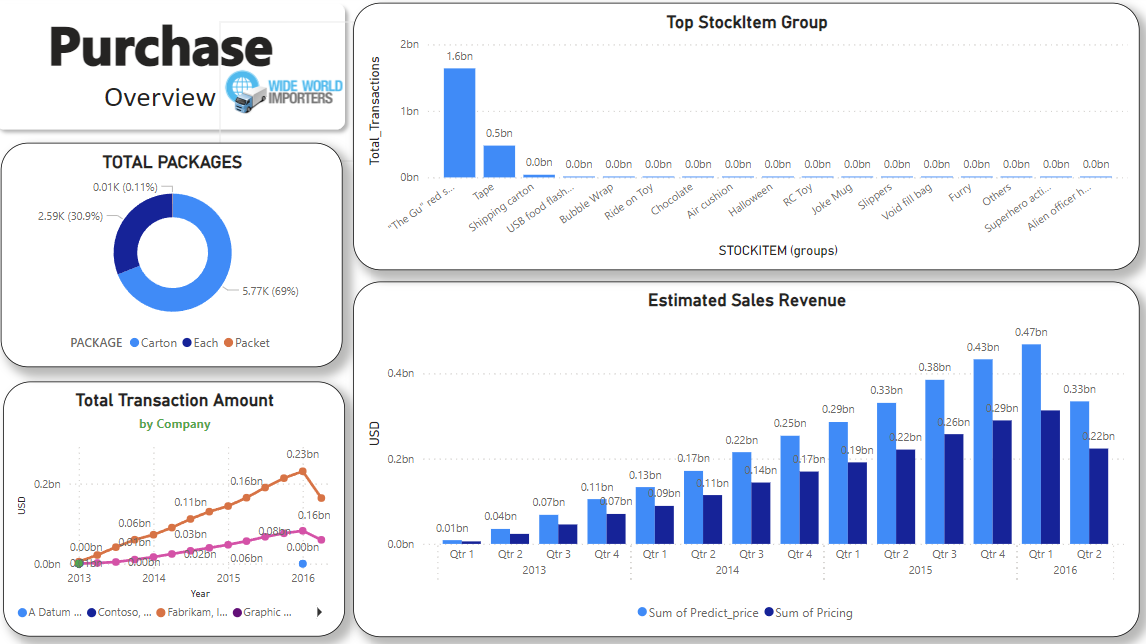
*Image 11: Orders Module Overview Report in PowerBi*

Image 11 illustrates the Overview page of Orders Module Report. Time can be selected in the top left corner of the page and information about top customers, top best employees and ‘best-seller’ stock items are shown. In addition, ‘Different Quantity per Year’, which helps to see the number of orders as well as comparison with the same period last year, in 2016 was highly decreased because of the lack of data in quarter 3 and quarter 4 in 2016.

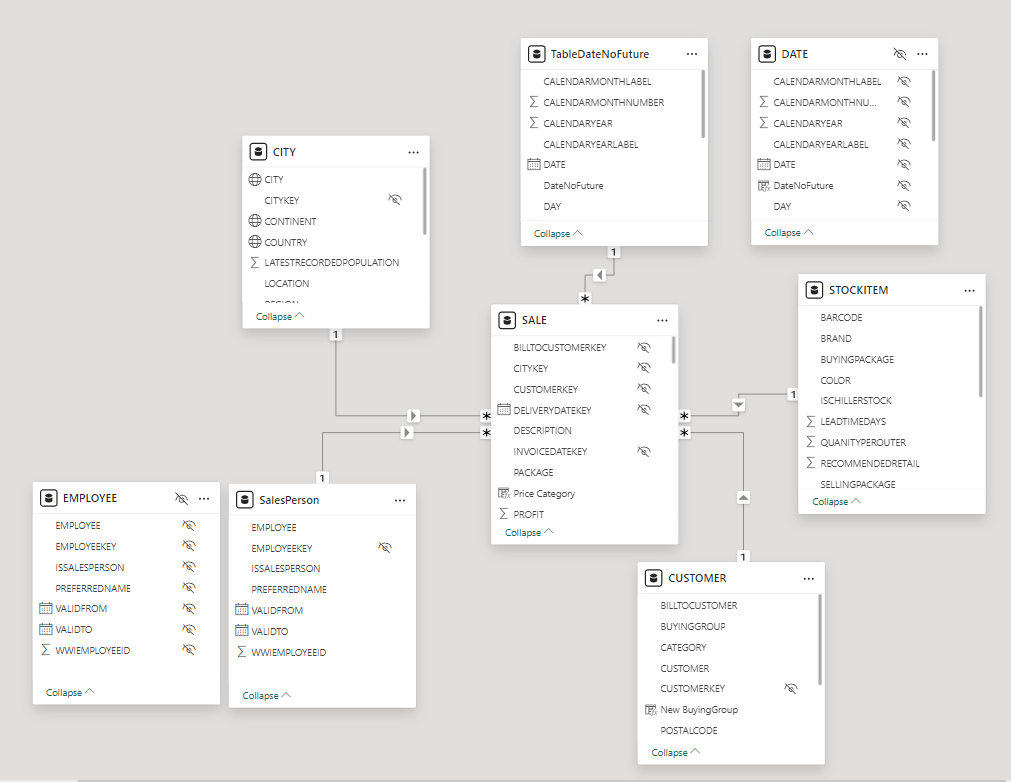


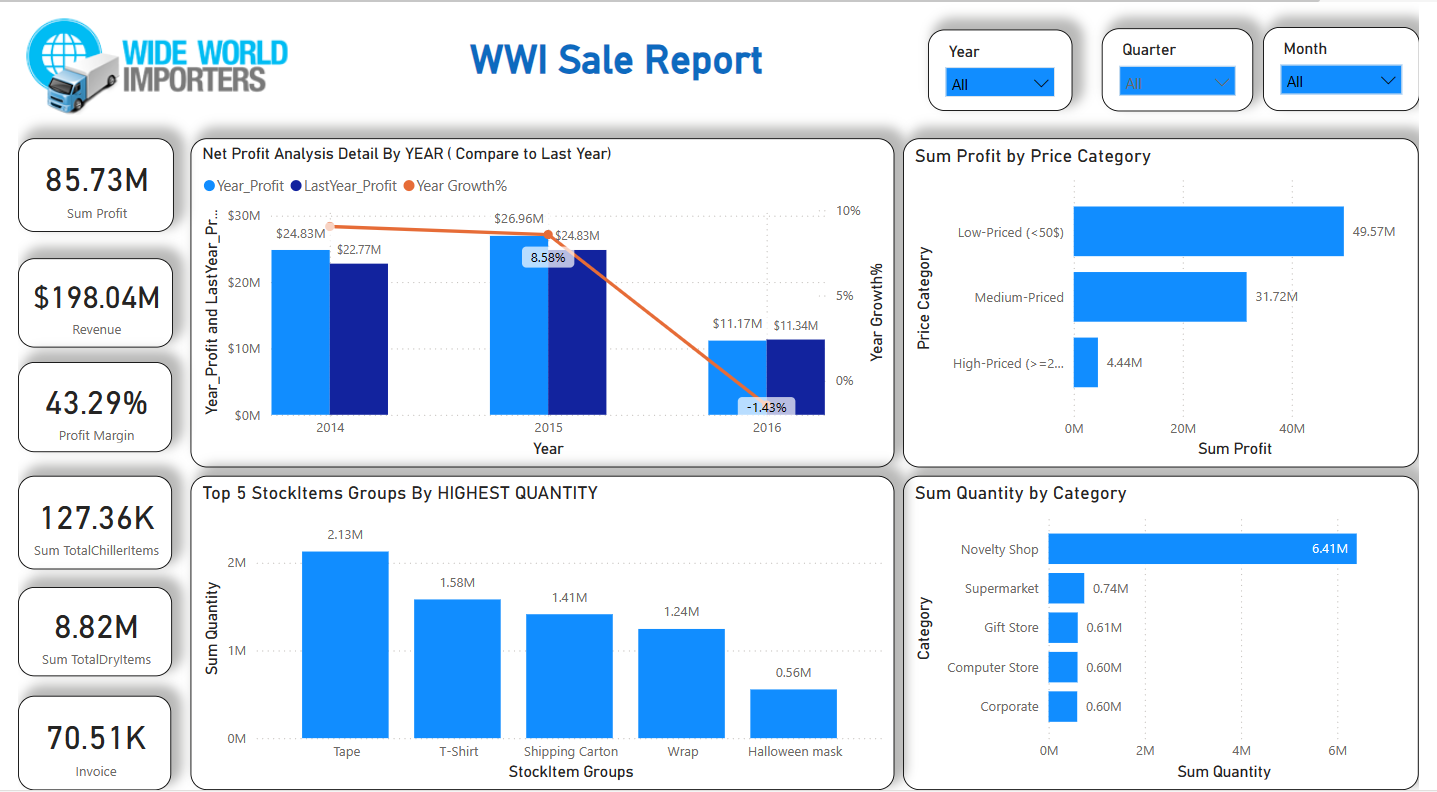
*Image 12: Orders Module Overview*  
 Image 12 shows how tables in Orders Module connected by keys. Besides, some new measures are also created to support analysis of this module clearly, some of them are ‘population’ (take average of city population), ‘Difference\_order\_year’ (compare number of orders by the same time in previous year) and ‘MONEY PAID’ (total amount of money spent).

9.3. Purchase Module  
 The Purchase module was built based on a star schema design to describe The Purchase module, meticulously crafted with a star schema design, offers a comprehensive dashboard of transaction costs, inventory management, and profit forecasting, empowering businesses to optimize expenses, enhance inventory control, and maximize profits.

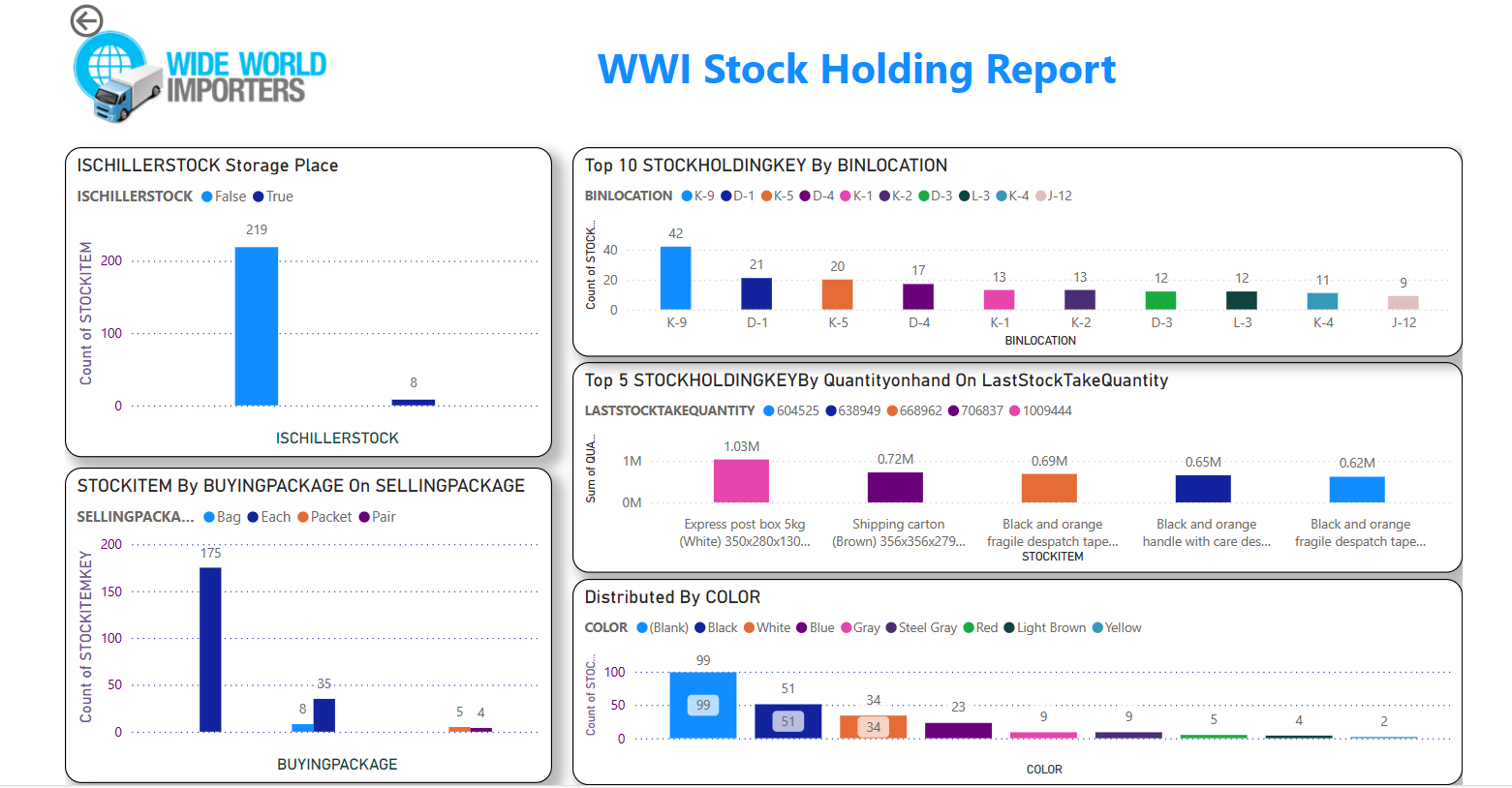
 *Image 13: Purchase Module Overview Report in PowerBI*

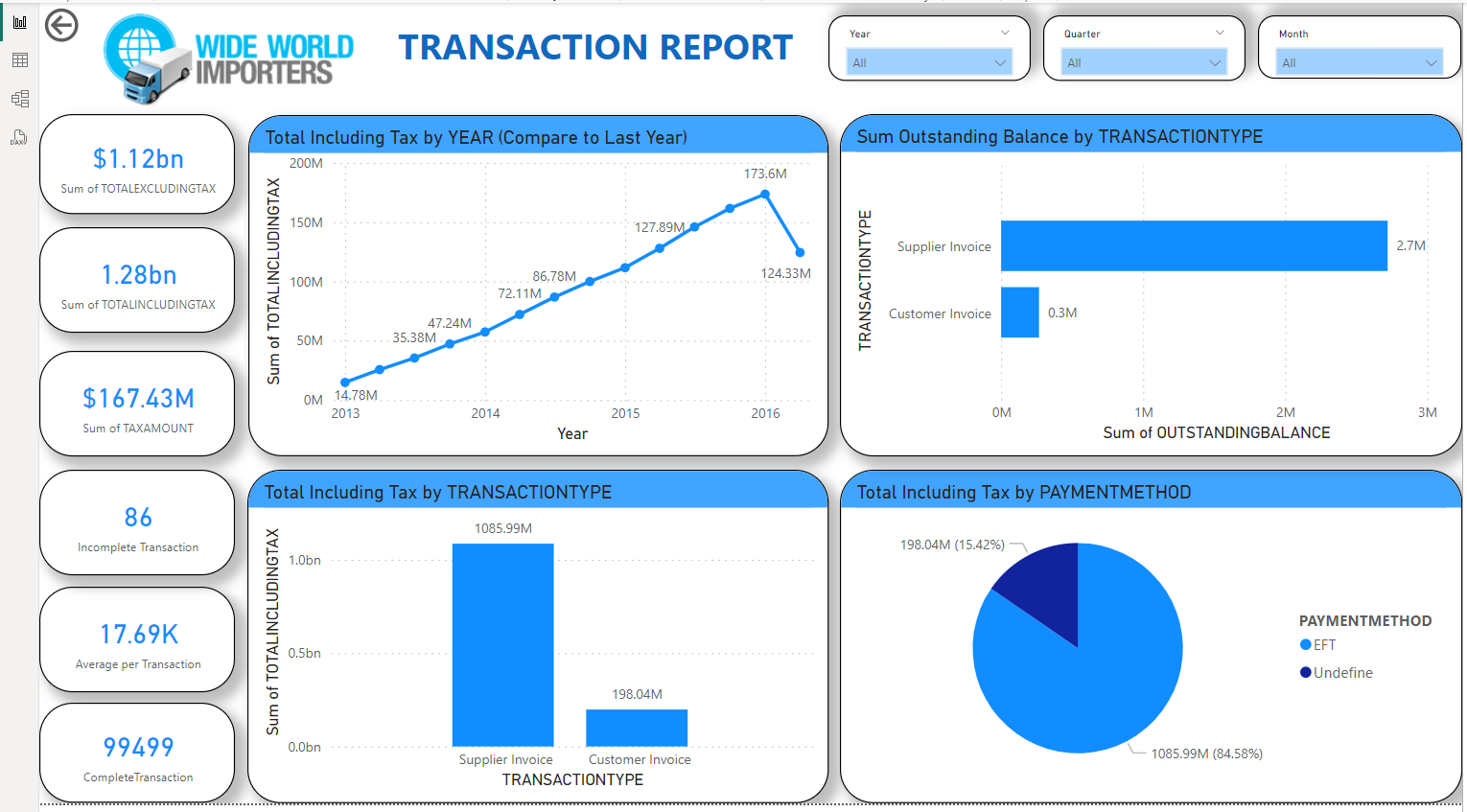
Through all visuals, we can see total transaction amount by company (supplier). Fabrikam and Litware emerge as the top-performing companies in terms of total transaction values. The company's top investment priorities are 'The Gu red shirt' and Tape, with estimated values demonstrating consistent annual growth.  
  
 9.4. Sale Module  
 The Sale module was built based on a star schema design to describe sales and profits between company and customers in all states of the USA from 2013 to 2016 May. Moreover, from these data, we can analyze profits, sales status among sales territories, stateprovinces, salesperson, customers behavior and stock item production.

 *Image 14: Sale Module Data Model*   
  
 This data model has many changes in two tables: Date and Employee. A new table ”TableDateNoFuture” was created to remove future time (from 2016 June). Moreover, there was a new table called ”SalesPerson” that filtered all rows in the ”Employee” table with condition that ”IsSalesPerson” column is True.  
 All processes use a lot of measures for analyzing. Any of them are profit comparation among same period this year and last year, stockitems in price category (cheap, medium, expensive), customers’ buying time,...

 *Image 15: Sale Module Overview Report in PowerBi*

Through all visuals, we can see generally that profit through years had been increasing (because there was profit only first 5 months in 2016, the profit decreased). Low-priced stock item quantity (smaller $20) was highest. Particularly, Texas was the state province that had the highest quantity and profit.  
  
 9.5. Stockholding Module

 *Image 16: StockHolding Module Overview Report in PowerBi*

Through all visuals. we can see generally that the overview of the company's Stockholding. On cold storage warehouse management is in ensuring the preservation of goods' quality during warehouse operations, implementation distributes of a color categorization system. Analyze inventory quantities by comparing quantity on hand and Last stock take quantity (This is the most recent check). Effective bin location management complements this by maximizing storage space utility and facilitating convenient access to items, further enhancing operational efficiency. Buying package (Usual package for selling units of this stock item) and selling package (Usual package for selling outers of this stock item (Ie cartons, boxes, etc.)) are often used to describe the process of importing and selling inventory. In summary, the integrity of stored goods but also contributes to the sustainable viability of the warehouse.  
  
 9.6. Transaction Module  
  *Image 17: Transaction Module Overview Report in PowerBi*

Through all visuals. We can see generally that the overview of the company's transactions. We can see the changing trend of after-tax transactions over time. (increased from 2013 to the end of 2015, 2016 begins to trend downward). The number of unsuccessful transactions that appeared in 2016 led to a decrease in revenue this year. The amount of revenue made using the "Undefined" payment method is larger, and transactions between customers and suppliers of this company are mainly paid by invoice. In particular, the Supplier has invoiced a very high quantity compared to the customer.

**10. Review and Conclusion**  
 10.1. Review  
 10.1.1. Advantages  
 Through this internship, all members have a basic understanding about the pipeline of Data Engineer job.  
 All members have tried to complete tasks based on mentor and team leader’s instructions.  
 Besides the knowledge everyone has gained, they can learn how to work in a company environment, team work,...  
 10.1.2. Limits  
 Except from the above good things that have been mentioned, there are many drawbacks that everyone can change:  
 - Many members depend on team leader and mentors for all tasks  
 - Many members are not really take care of their task, try to complete without checking errors, formats, templates,...  
 - Because this is only internship, everyone is not aware of deadlines of completing tasks, not completely focus on their task, only complete when reminding.  
   
 10.2. Conclusion  
 The Data Engineer project has achieved significant success in building an effective and flexible data processing system for the company. By optimizing data collection, processing and storage processes, we have created a powerful platform to support business decisions and product development. Our work not only enhances performance and minimizes risk, but also creates a flexible and scalable environment that helps our organization continue to grow and adapt in an ever-changing market. more variable. The Data Engineer team's efforts are an important step in driving business growth and a testament to our commitment to continuously improve quality and performance in the data sector.