Northeastern University
College *of* Engineering

Data Warehousing & Integration

IE 6750

FALL 2024

Book Sales Management System

Project 1 Final Report

Group 10

Sri Sai Prabhath Reddy Gudipalli

Avani Kala

gudipalli.s@northeastern.edu

kala.a@northeastern.edu

Submission Date: 12/07/2024

Objective:

The objective of this project is to design and implement a data warehousing solution that integrates sales, and customer data for a retail bookstore. By consolidating transactional and static reference data from various operational sources such as orders, books, authors, publishers, and customers, the bookstore will gain valuable insights to optimize management, forecast demand, and execute personalized marketing strategies. The solution will enable the business to make data-driven decisions, improve customer engagement, and enhance overall profitability through better sales analysis. We took the data from the following link
https://github.com/bbrumm/databasestar/tree/main/sample_databases/sample_db_gravity/gravity_sqlserver

Problem Statement:

The retail bookstore faces significant challenges in effectively managing its sales due to the lack of a centralized system to integrate and analyze data from various operational sources. This disjointed approach leads to inefficient management, difficulty in forecasting demand, and an inability to implement personalized customer marketing strategies. Without a robust data warehousing solution, the bookstore struggles to track book performance, manage supplier relationships, and optimize sales strategies, ultimately impacting profitability and customer satisfaction.

Key Points:

- Disjointed data across multiple operational sources, leading to inefficiencies in sales management.
- Lack of real-time visibility into stock levels, resulting in overstocking or stockouts.
- Difficulty in forecasting customer demand and planning and replenishment.
- Inability to perform customer behavior analysis, hindering personalized marketing efforts.
- Limited insight into book and author performance, affecting targeted promotions and sales optimization strategies.

Problem Setting:

In today's competitive and data-driven business landscape, companies in the book sales industry face significant challenges in understanding their customers, optimizing operations, and enhancing sales strategies. A common obstacle for such businesses is the lack of a comprehensive and unified system that consolidates sales, customer, product, and geographic data for analysis. Without an effective data warehouse, organizations struggle to make data-informed decisions, leading to missed revenue opportunities and inefficiencies in operational processes.

Datawarehouse design:

The purpose of this proposal is to outline the design of a data warehouse that supports the Book Sales Management System. This design will facilitate efficient data storage, retrieval, and analysis, enabling informed decision-making and strategic insights into sales. We propose implementing a snowflake schema as our data warehouse design. This will consist of a fact table at the center that captures transactional data, surrounded by dimension tables that provide contextual details. This structure is

optimal for ROLAP as it leverages relational databases to store and query large volumes of data efficiently.

Architecture Overview:

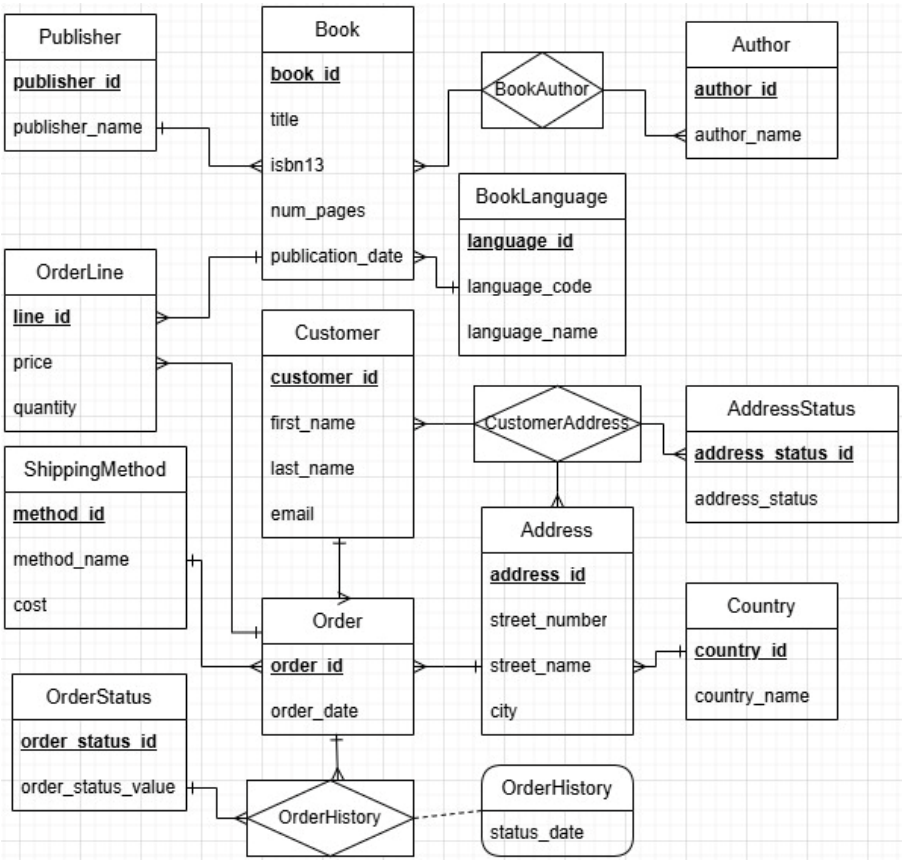
Fact Tables:

- **OrderFact:** Contains measures such as total sales, quantity sold, and order count, with foreign keys referencing dimension tables.
- **OrderLineFact:** Details each line item within an order, allowing for detailed sales analysis.

Dimension Tables:

- **CustomerDimension:** Stores customer details (e.g., ID, name, email).
- **BookDimension:** Contains book details (e.g., ISBN, title, author, genre).
- **PublisherDimension:** Includes publisher information (e.g., publisher ID, name).
- **TimeDimension:** Facilitates time-based analysis, storing attributes such as year, quarter, month, and day.
- **ShippingMethodDimension:** Details different shipping methods available.
- **AddressDimension**: Captures address details for customers, which can aid in geographical analysis.

ERD:



Relational:

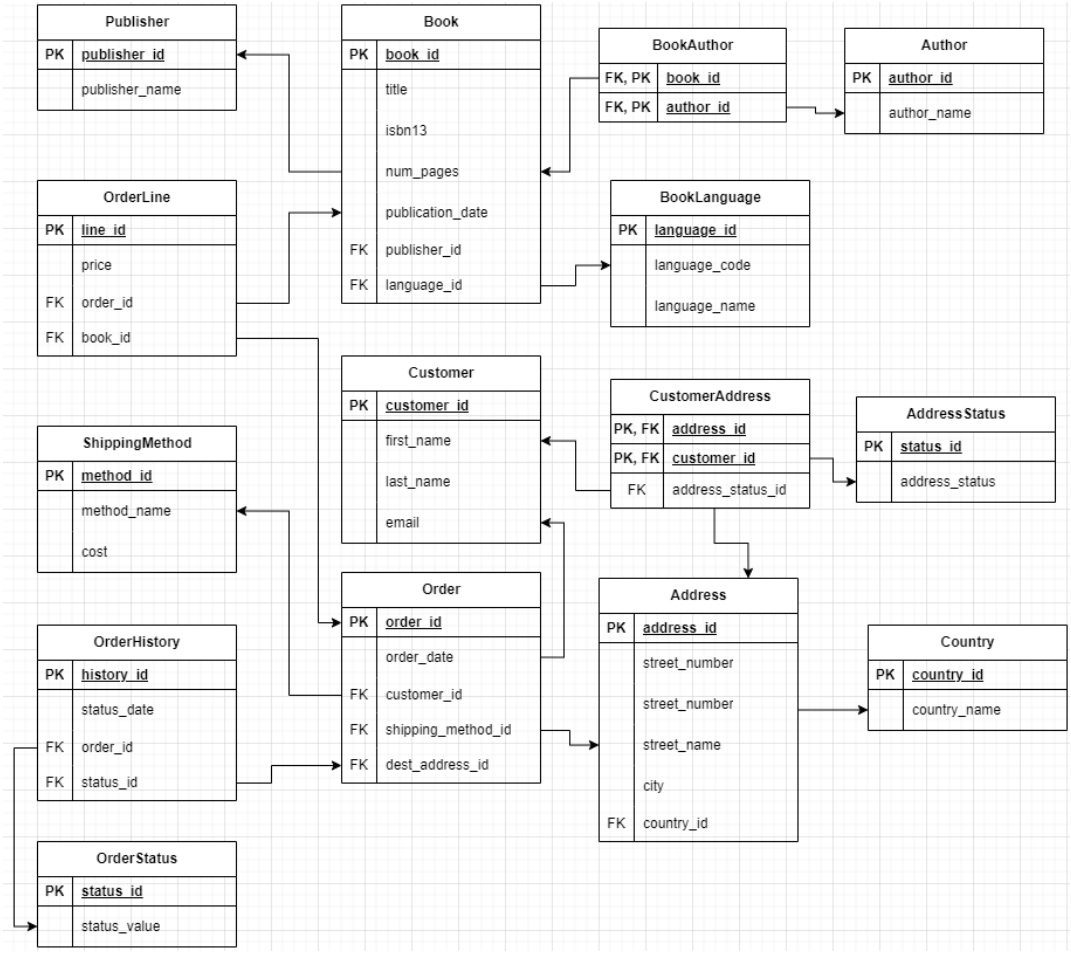
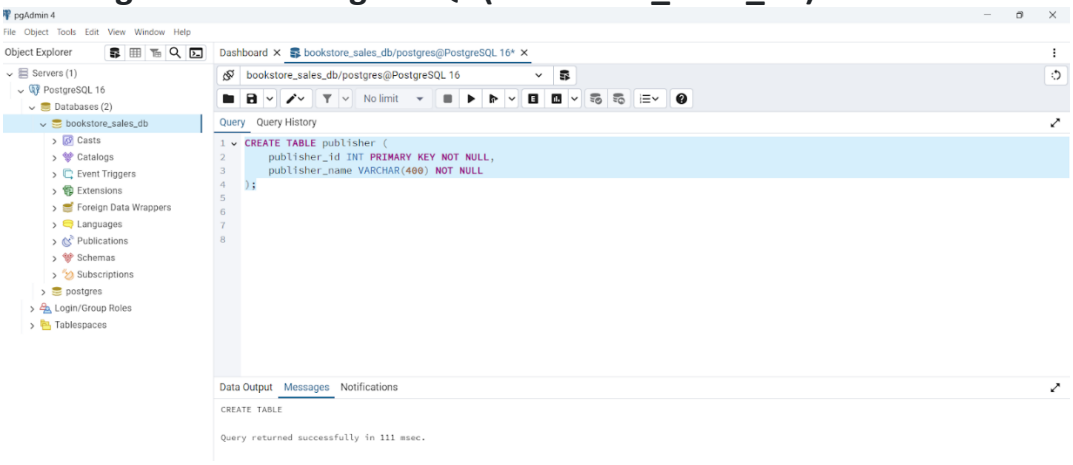
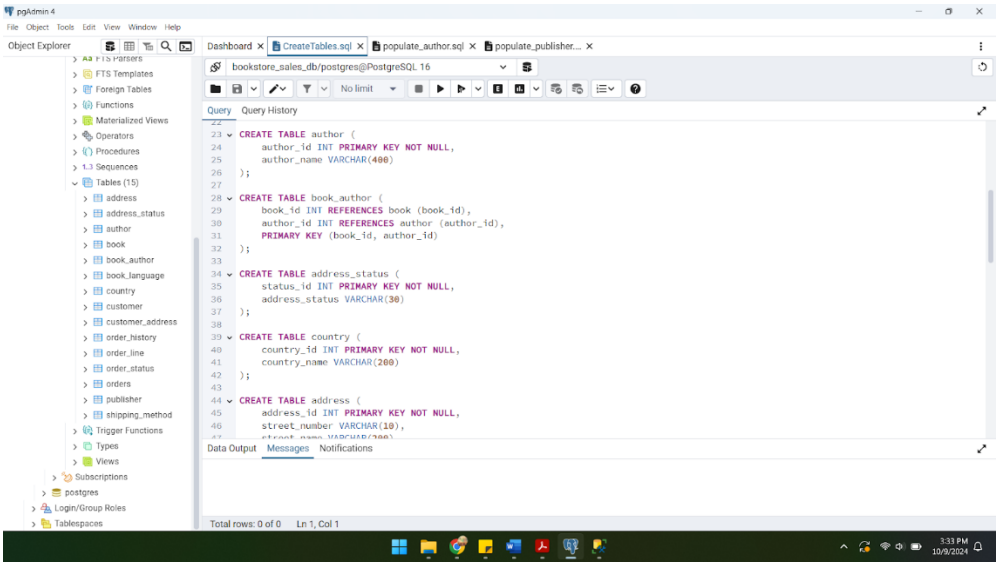


Table Description:

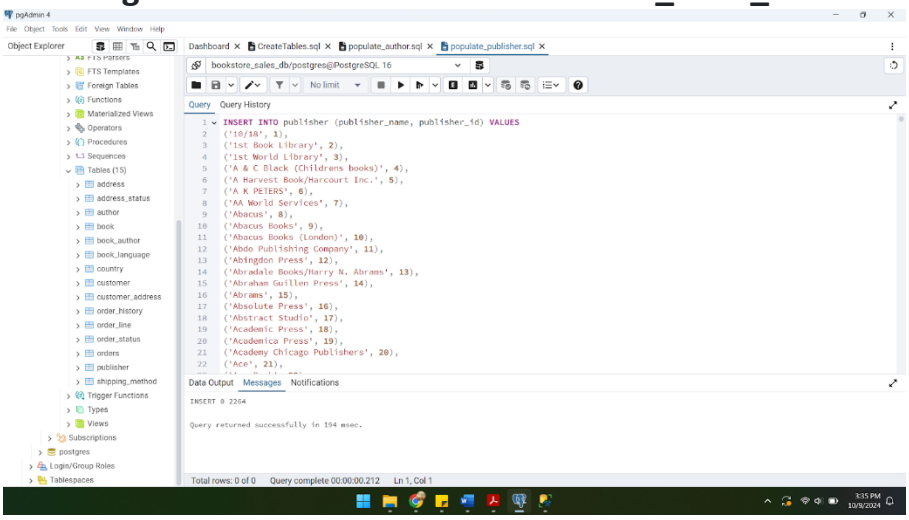
- **Publisher**: Details of publishers for books.
- **Book**: Details of books available in the store.
- **BookAuthor**: Stores the authors for each book, which is a many-to-many relationship.
- **Author**: Details of all authors.
- **BookLanguage**: A list of possible languages of books.
- **Customer**: A list of the customers of the Bookstore.
- **CustomerAddress**: A list of addresses for customers. A customer can have more than one address, and an address can have more than one customer.
- **AddressStatus**: A list of statuses for an address. Addresses can be Active or Inactive
- **Address**: Details of addresses in the system.
- **Country**: A list of countries that addresses are in.
- **Order**: A list of orders placed by customers.
- **OrderLine**: A list of books that are a part of each order.
- **ShippingMethod**: Details of shipping methods for an order (Priority, Standard, Express).
- **OrderHistory**: History of an order, such as ordered, shipped, delivered.
- **OrderStatus**: The possible status of an order (Order Received, Delivered, Canceled)

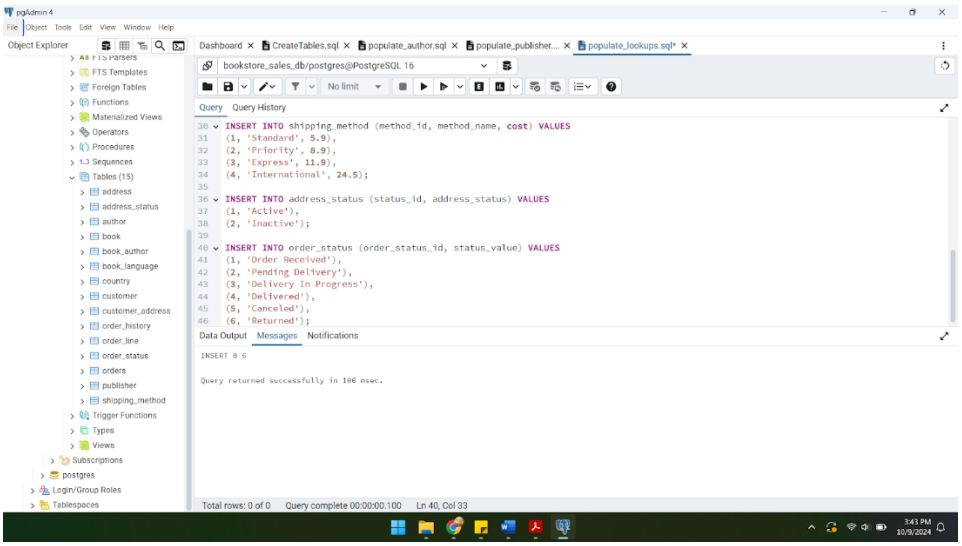
Creating tables in PostgresSQL (bookstore_sales_db)

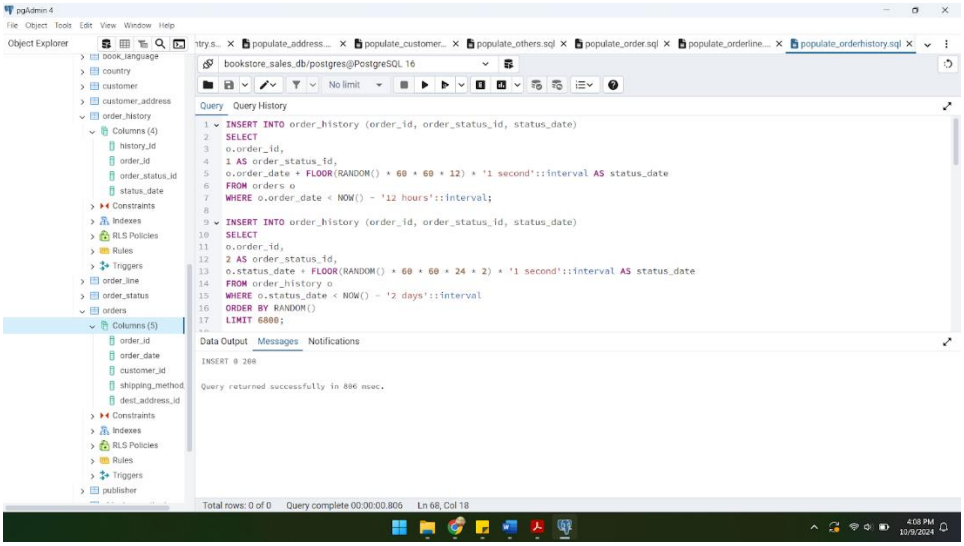




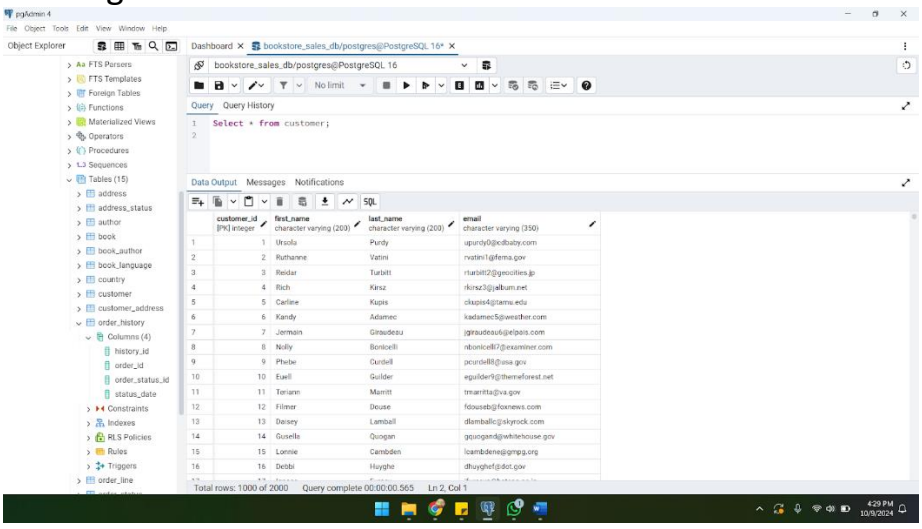
Inserting data into the tables in bookstore_sales_db:







Viewing data in customer table:



In this project, we have categorized the data into dimensional data, fact data, and reference data based on their roles within the system. Here is how we have labeled and organized them:

1. Dimensional Data (Slowly and Not-So-Slowly Changing Data)

These tables contain descriptive information about entities, and some of these attributes may change over time. We'll apply slowly changing dimension (SCD) strategies to track historical changes where necessary.

- **Publisher**: Stores details of publishers for books. This dimension may change over time (e.g., publisher mergers or name changes), making it a slowly-changing dimension if historical data needs to be retained.

- **Book**: Contains details of books available in the store, including ISBN, title, and other metadata. Since book details like editions or prices could change over time, we treat this as a slowly-changing dimension.

- **Author**: Contains information about the authors of books. This is also a dimensional table since author details might occasionally change, such as name updates.

- **BookAuthor**: Manages the many-to-many relationship between books and authors. Changes to author assignments are not so frequent, so this is a slowly-changing dimension.

- **Customer**: A list of bookstore customers. Since customer data can change (e.g., name, contact info), we treat this as a slowly-changing dimension.

- **CustomerAddress**: Stores the addresses of customers. Customers may update their addresses over time, making this a slowly-changing dimension.

- **Address**: Contains details of addresses in the system, which may change over time (e.g., updates to city or postal codes). This is also treated as a slowly-changing dimension.

- **OrderHistory**: Tracks the status changes of an order, from being received to being shipped and delivered. This would be a not-so-slowly-changing dimension, as order status updates occur frequently but still need tracking.

2. Fact Data (Transactional Data)

These tables represent transactions and capture specific events. They are often updated frequently and contain references to dimensional data.

- **Order**: Represents individual customer orders. Each order is a transactional record capturing details like the date and customer associated with it.

- **OrderLine**: Contains the books in each order. This is a fact table that links orders to the books being purchased. The data here is transactional and updated with each purchase.

3. Reference Data (Static or Slowly Changing)

These tables hold static or near-static data that is referenced across the system but doesn't change frequently. They are often used for validation or categorization.

- **BookLanguage**: A static list of possible languages of books. This is reference data, as language options rarely change over time.

- **AddressStatus**: Contains statuses for addresses, such as "Active" or "Inactive." This reference data doesn't change frequently, but it's used for managing addresses.

- **Country**: Stores a list of countries that addresses belong to. Countries rarely change, so this is static reference data.

- **ShippingMethod**: Defines the available shipping methods (e.g., Priority, Standard, Express). This is reference data that may occasionally expand but changes infrequently.

- **OrderStatus**: Tracks the various stages of an order (Order Received, Delivered, Canceled). This data is static but frequently referenced in order processing.

This Book Sales Management System is designed to ensure a reasonable volume of transactional data. The Order and OrderLine tables capture key transactional data, including every purchase made by customers and the details of the books in each order.

By tracking each customer order and the specific items within those orders, we generate a robust dataset for analysis. Additionally, the OrderHistory table logs the status changes of each order (e.g., "Order Received," "Shipped," "Delivered"), further expanding the volume of transactional data. This allows for comprehensive reporting on customer purchases, order fulfillment times, and sales trends over time.

Dimensions and Hierarchies

This project contains multiple dimensions, hierarchies, and key measures that can be analyzed.

Potential Dimensions: Dimensions represent entities or attributes that provide context to the data. Here are some key dimensions from our system:

1. **Customer Dimension**: Contains customer details such as name, email, and customer ID.
2. **Book Dimension**: Contains details about the books, such as title, genre, ISBN, and publication year.
3. **Publisher Dimension**: Stores publisher information, which can also be used to analyze sales by publisher.
4. **Order Dimension**: Contains order details, such as order date, order ID, and shipping method.
5. **Time Dimension**: Implicit in the Order table via the order date, allowing for time-based analysis (e.g., sales over months, quarters, or years).
6. **Address Dimension**: Stores customer address details, which could be used for geographical analysis of sales by region.

Hierarchies: Hierarchies allow us to drill down into the data at different levels of granularity for more detailed analysis.

1. **Time Hierarchy**:
    - Year → Quarter → Month → Day → Order Date

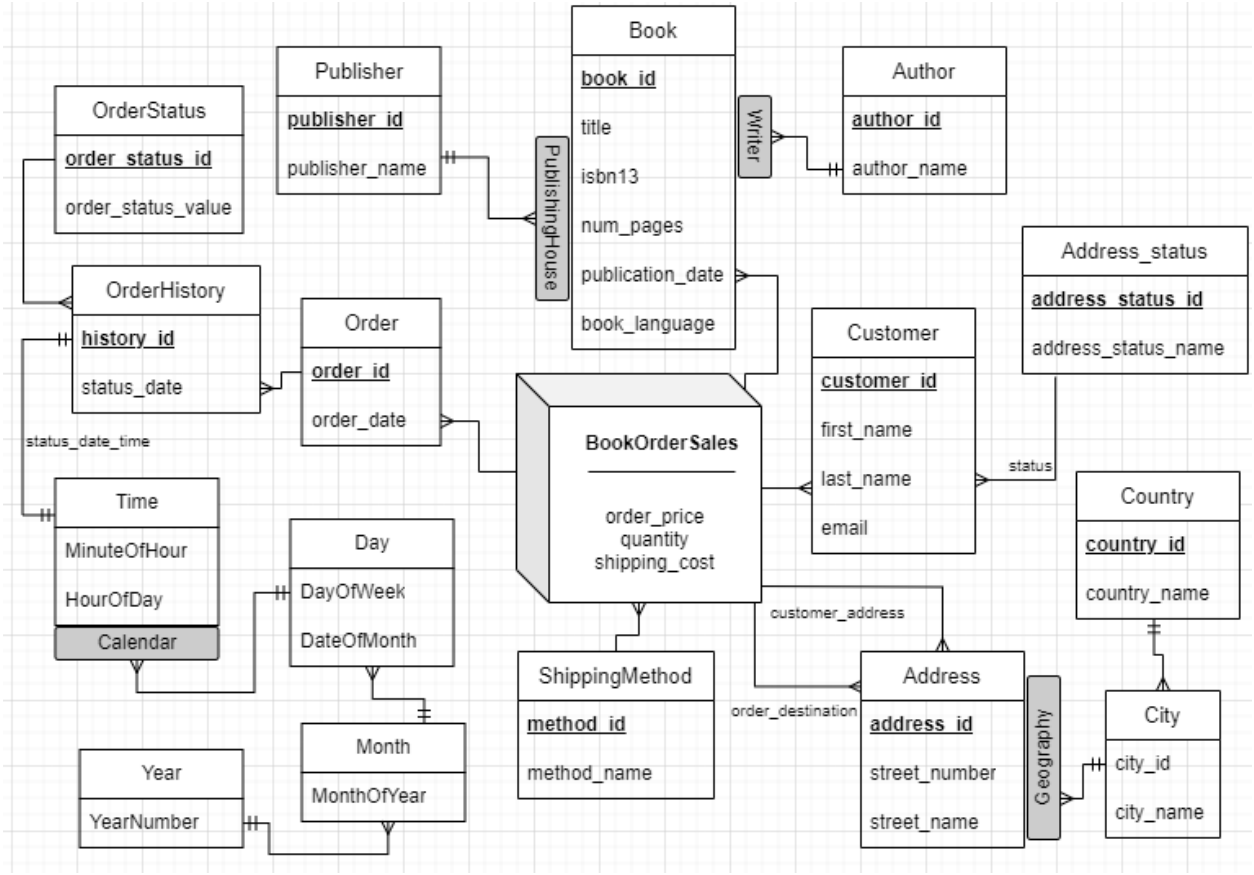
    - This allows analysis of sales trends over time, from high-level yearly views to detailed daily trends.
2. **Geographical Hierarchy**:
    - Country → State/Province → City → Postal Code
    - This enables analysis of sales or customer distribution across geographic regions.
3. **Product Hierarchy**:
    - Publisher → Book Title → Edition
    - This hierarchy can help analyze book sales by publisher, then by title, and then by edition.

Measures: Measures are quantitative data points that can be aggregated or analyzed. We have several important measures:

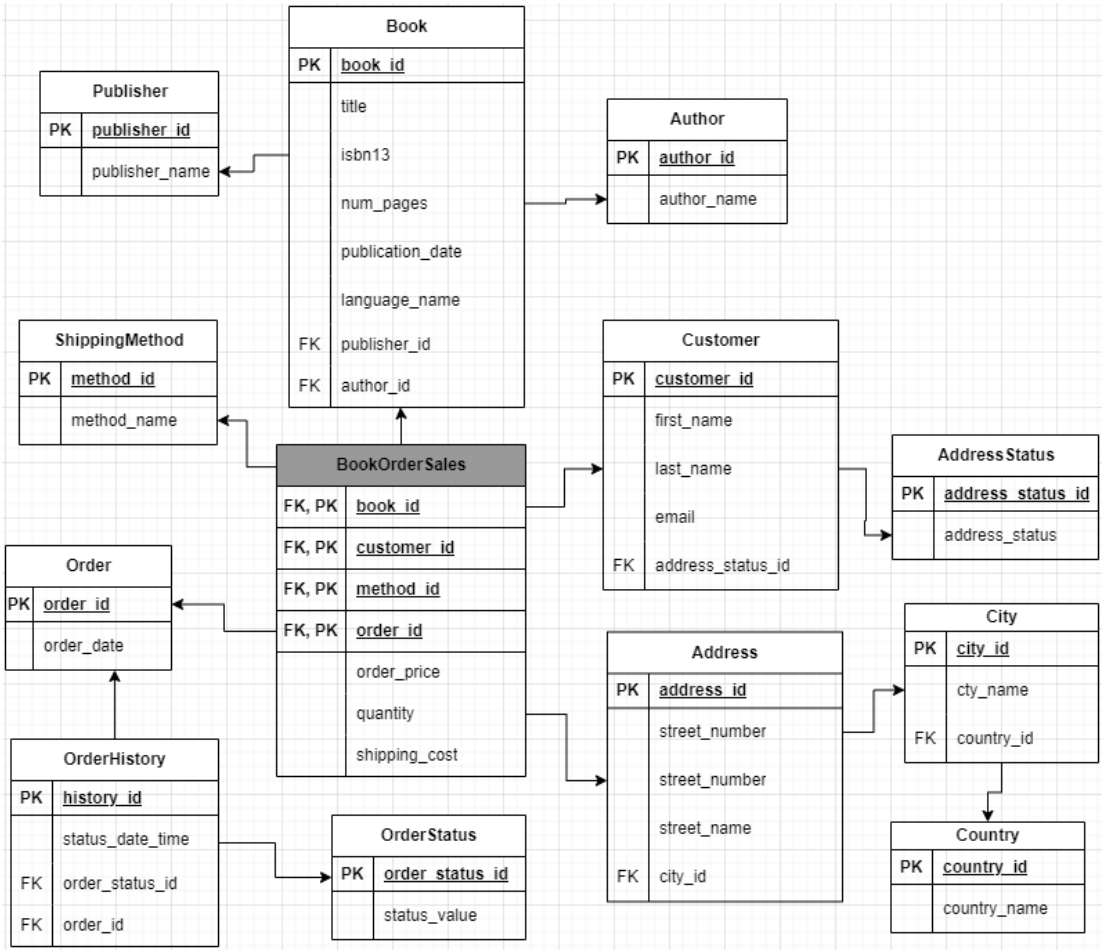
1. **Order Price**: Sum of the sales amount in the OrderLine table. This is a primary measure to evaluate store performance and can be analyzed by dimensions like Time, Customer, or Book.
2. **Quantity Sold**: The number of books sold, which can be aggregated from the OrderLine table. This can be analyzed across different dimensions like Book or Publisher.

These measures, combined with the dimensions and hierarchies, provide a framework for analysis.

Conceptual DW Model



Logical DW Model

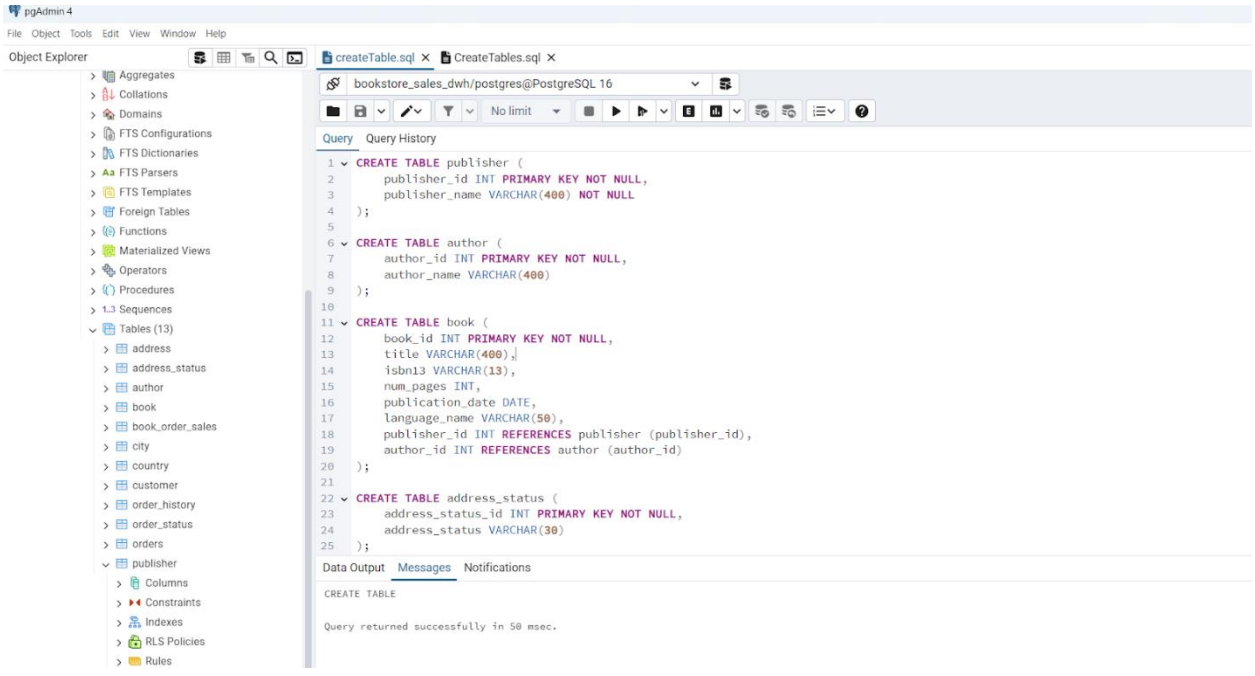


DB Implementation

```
CREATE TABLE publisher (
    publisher_id INT PRIMARY KEY NOT NULL,
    publisher_name VARCHAR(400) NOT NULL
);

CREATE TABLE author (
    author_id INT PRIMARY KEY NOT NULL,
    author_name VARCHAR(400)
);
```

```
CREATE TABLE book (
    book_id INT PRIMARY KEY NOT NULL,
    title VARCHAR(400),
    isbn13 VARCHAR(13),
    num_pages INT,
    publication_date DATE,
      language_name VARCHAR(50),
      publisher_id INT REFERENCES publisher (publisher_id),
      author_id INT REFERENCES author (author_id)
);

CREATE TABLE address_status (
    address_status_id INT PRIMARY KEY NOT NULL,
    address_status VARCHAR(30)
);

CREATE TABLE country (
    country_id INT PRIMARY KEY NOT NULL,
    country_name VARCHAR(200)
);

CREATE TABLE city (
      city_id INT PRIMARY KEY NOT NULL,
      city VARCHAR(200),
      country_id INT REFERENCES country (country_id)
)

CREATE TABLE address (
    address_id INT PRIMARY KEY NOT NULL,
    street_number VARCHAR(10),
    street_name VARCHAR(200),
    city_id INT REFERENCES city (city_id)
);

CREATE TABLE customer (
    customer_id INT PRIMARY KEY NOT NULL,
    first_name VARCHAR(200),
    last_name VARCHAR(200),
    email VARCHAR(350),
      address_status_id INT REFERENCES address_status
(address_status_id)
);

CREATE TABLE shipping_method (
    method_id INT PRIMARY KEY NOT NULL,
    method_name VARCHAR(100)
);

CREATE TABLE orders (
    order_id SERIAL PRIMARY KEY NOT NULL,
    order_date TIMESTAMP
```

```
);

CREATE TABLE order_status (
    order_status_id INT PRIMARY KEY NOT NULL,
    status_value VARCHAR(20)
);

CREATE TABLE order_history (
    history_id SERIAL PRIMARY KEY NOT NULL,
      status_date_time TIMESTAMP,
    order_status_id INT REFERENCES order_status (order_status_id),
      order_id INT REFERENCES orders (order_id)
);

CREATE TABLE book_order_sales (
      order_price INT NOT NULL,
      quantity INT NOT NULL,
      shipping_cost INT NOT NULL,
      book_id INT REFERENCES book (book_id),
      customer_id INT REFERENCES customer (customer_id),
      method_id INT REFERENCES shipping_method (method_id),
      order_id INT REFERENCES orders (order_id)
);
```



Primary Event: Our primary event is bookstore sales, and we record the purchasing cost as order price, number of books sold as quantity in each order and the cost for shipping the order.

OLAP Operations: Here are the OLAP operations we will be trying to explore with bookstore sales (BookSales is the name of the hypercube):

1. Yearly book sales by book language.

```
YearlySales <- ROLLUP*(BookSales, Book -> book_language,
OrderHistory.time -> year, SUM(order_price))
```

2. Total amount of orders shipped to Boston (example city).

```
CitySales <- ROLLUP(BookSales, Book.order_destination -> city)
BostonSales <- SLICE(CitySales, order_destination.city = "Boston")
SUM(BostonSales, order_price)
```

3. Number of orders created in 2023 vs 2022 that are returned.

```
ReturnedOrders <- SLICE(BookSales, OrderHistory.order_status =
"Returned")
YearlySales <- ROLLUP*(ReturnedOrders, OrderHistory.time -> year,
COUNT(order_price))
```

4. Number of orders using different shipping methods.

```
ShippingMethodSales <- ROLLUP*(BookSales, ShippingMethod ->
method_name, COUNT(order_price))
```

5. Number of canceled orders for Penguin Publishing House (example publisher).

```
ReturnedPenguinOrders <- DICE(BookSales, OrderHistory.order_status =
"Canceled", Book.Publisher = "Penguin Publishing House")
```

6. Total number of books sold for Dan Brown (example author) in 2022 vs 2023 vs 2024.

```
DanBrownSales <- SLICE(BookSales, Book.Author = "Dan Brown")
YearlySales <- ROLLUP*(DanBrownSales, OrderHistory.time -> year,
SUM(quantity))
```

7. Monthly book sales compared to those of the previous year.

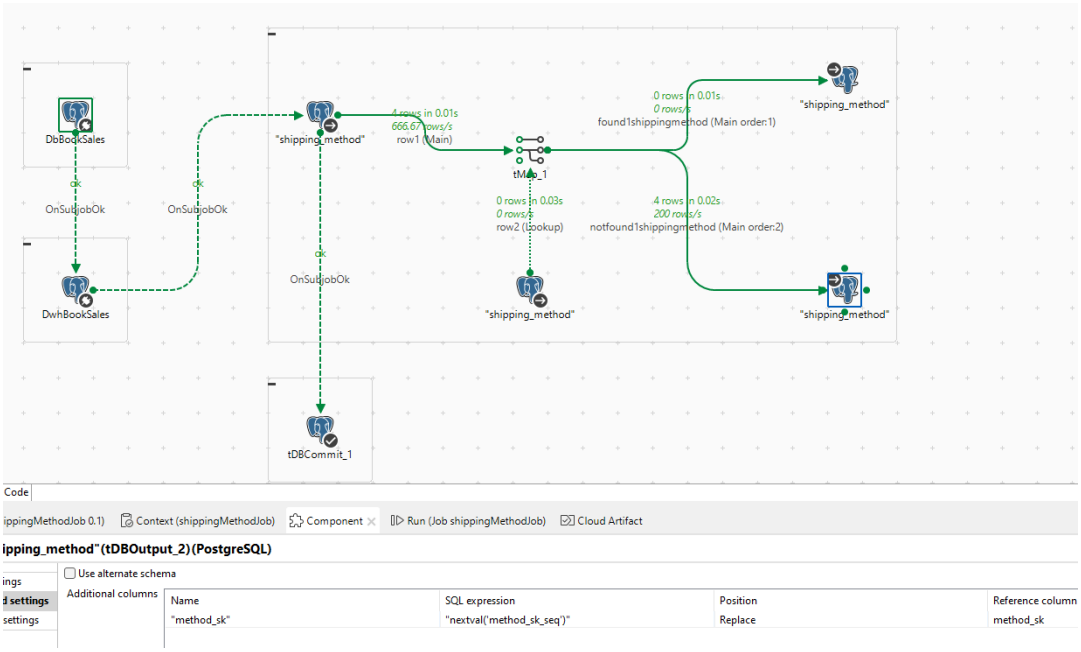
```
Sales1 <- ROLLUP*(BookSales, OrderHistory.time -> Month,
SUM(order_price))
Sales2 <- RENAME(Sales1, order_price -> previous_year_order_price)
Result <- DRILLACROSS(Sales2, Sales1,
Sales2.OrderHistory.Month = Sales1.OrderHistory.Month AND
Sales2.OrderHistory.Year+1 = Sales1.OrderHistory.Year)
```

Data Description:

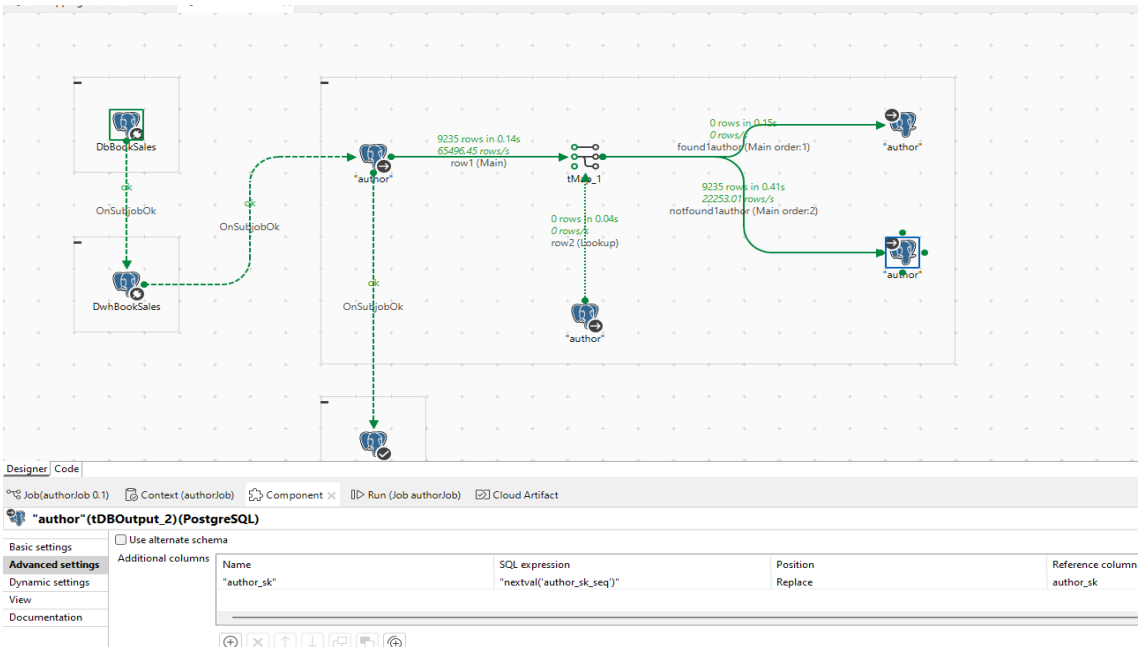
The dataset brings together information on sales, customers, book, order statuses to help analyze sales trends, understand the method of tracking orders and its status at different time.

Loading data from Operational DB to Warehouse DB using Talend:

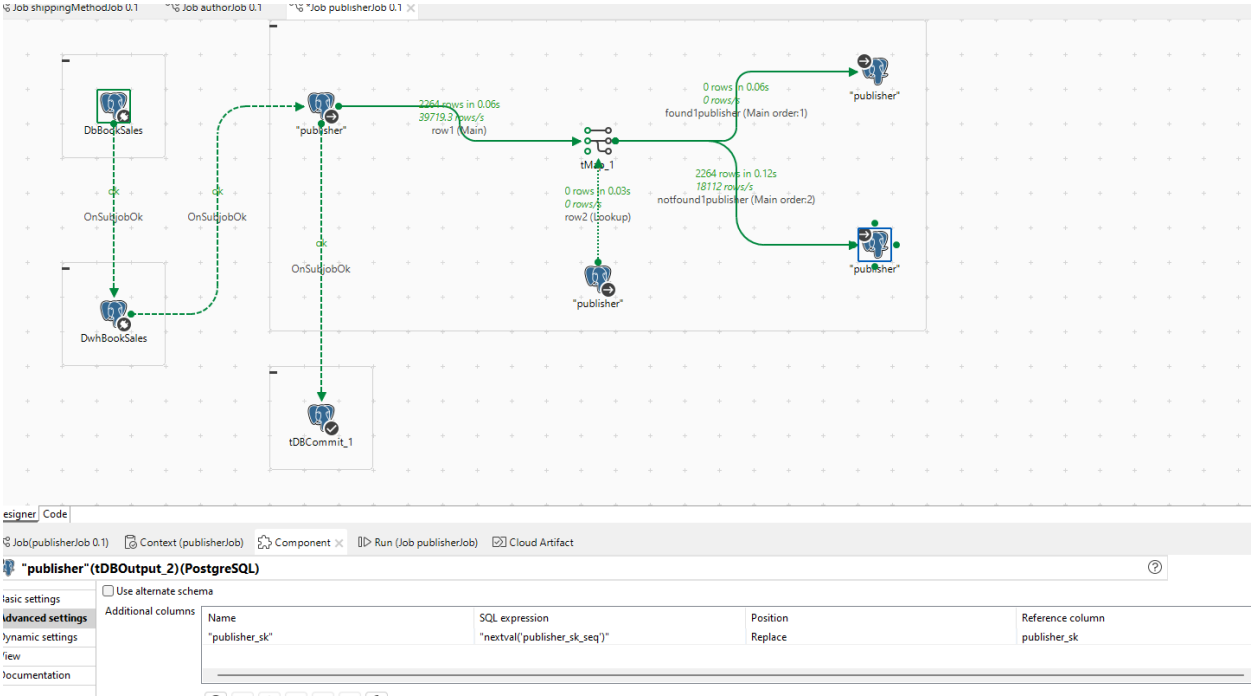
Shipping Method –



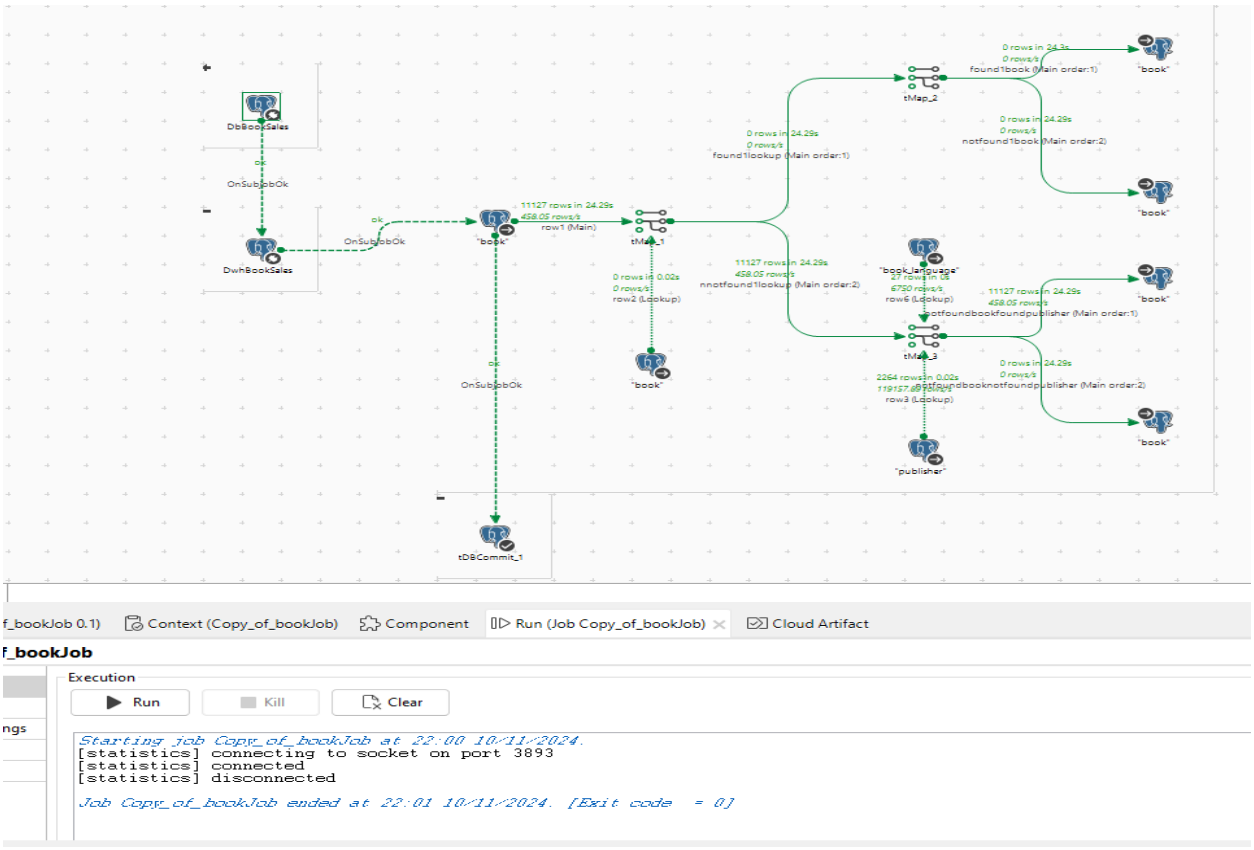
Author –



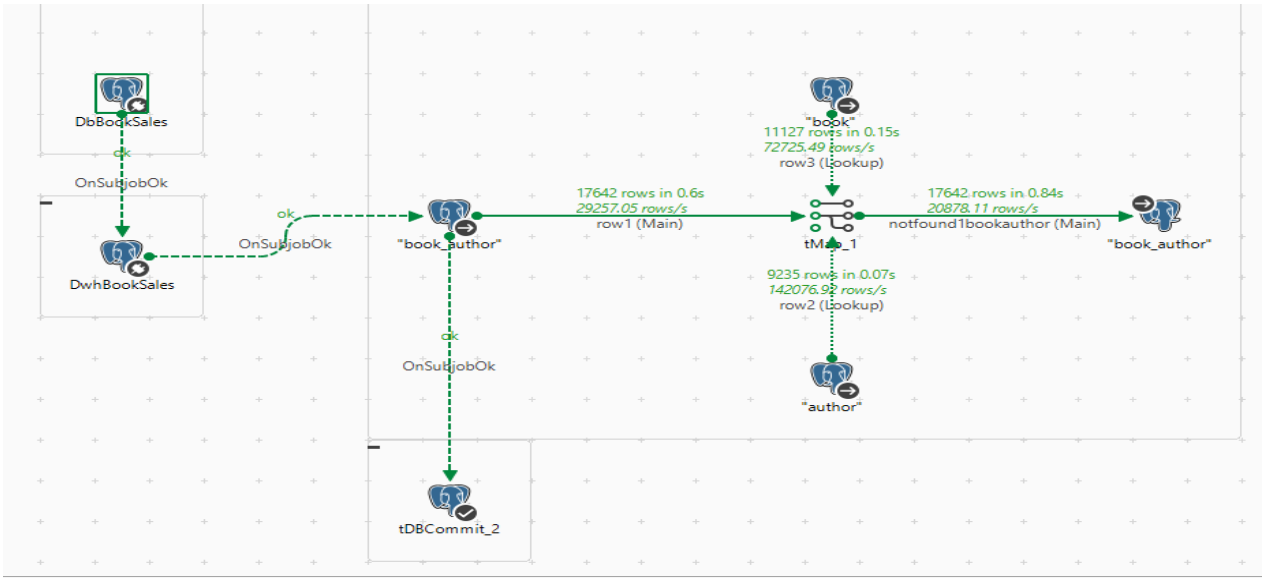
Publisher –

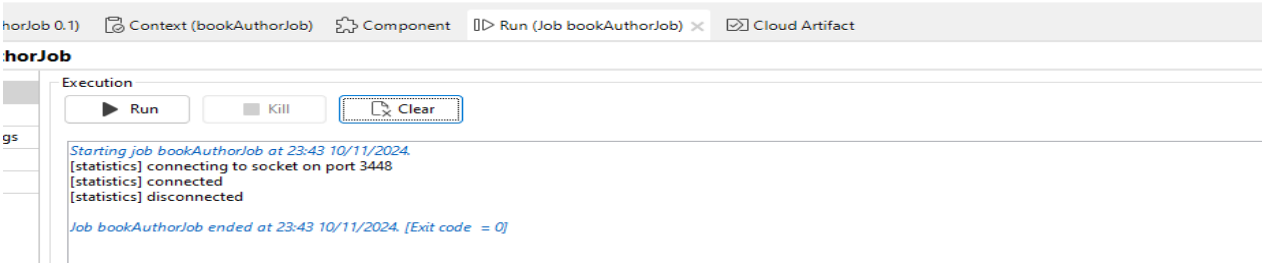


Book-

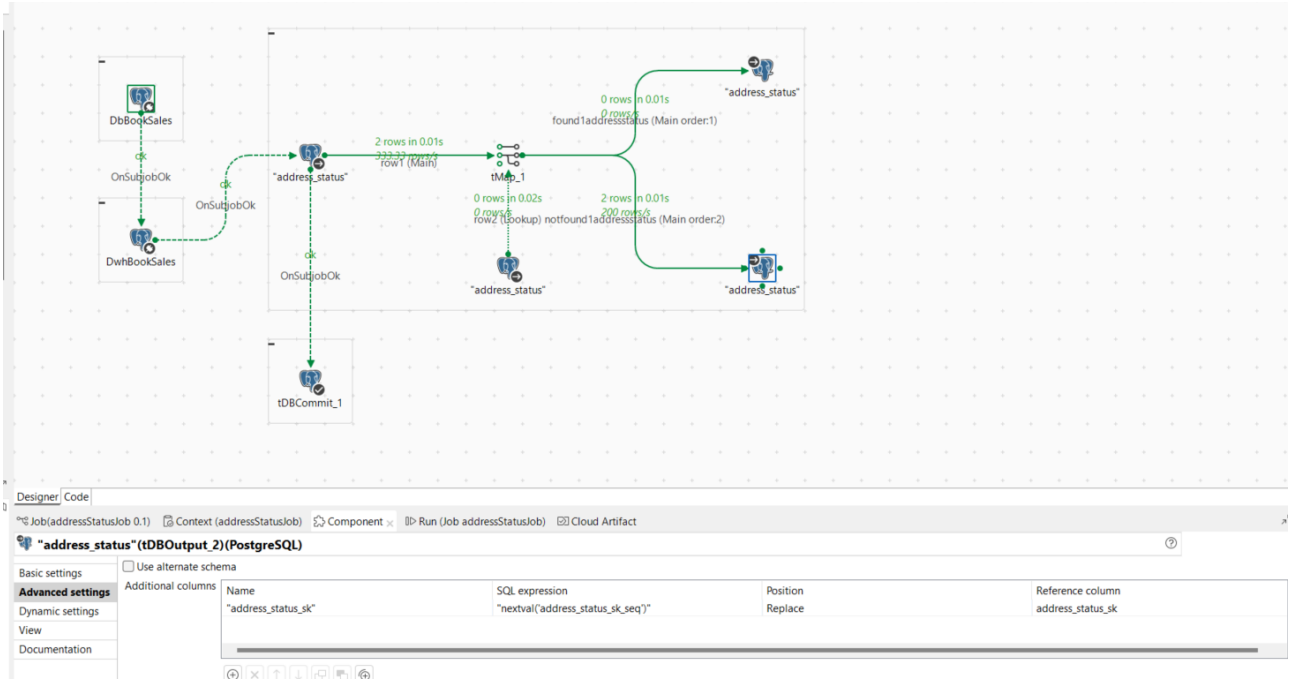


Book_Author-

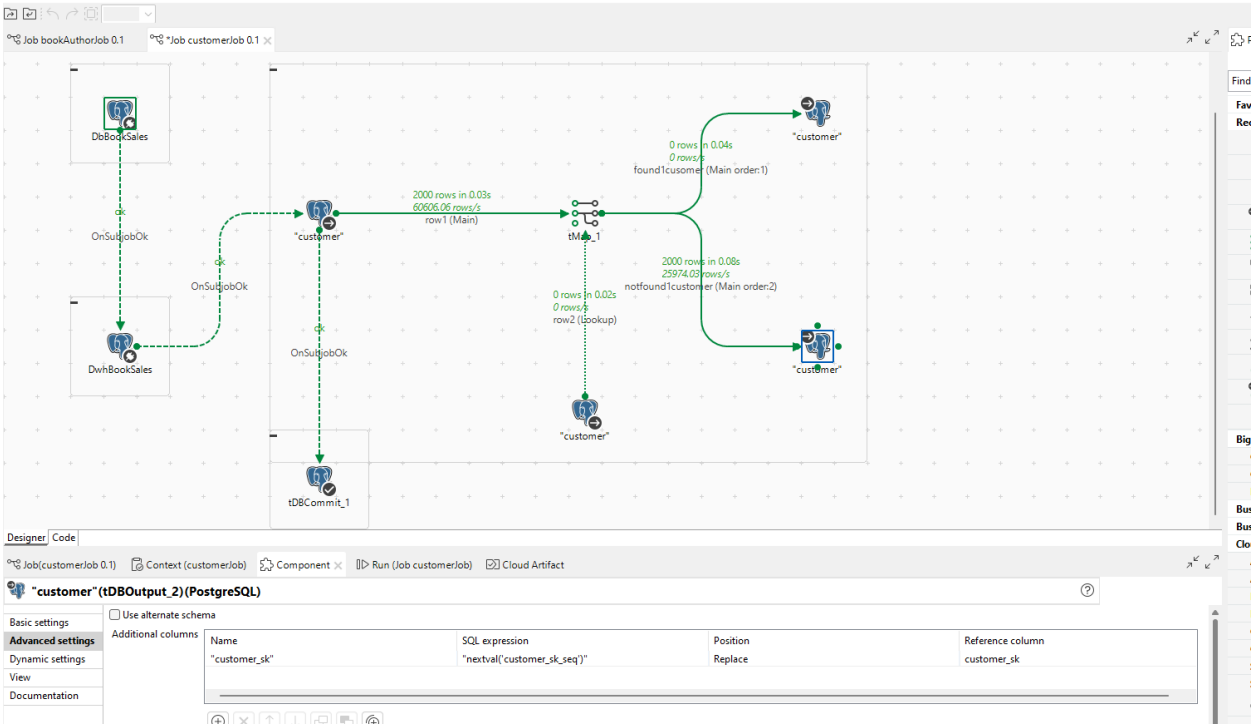




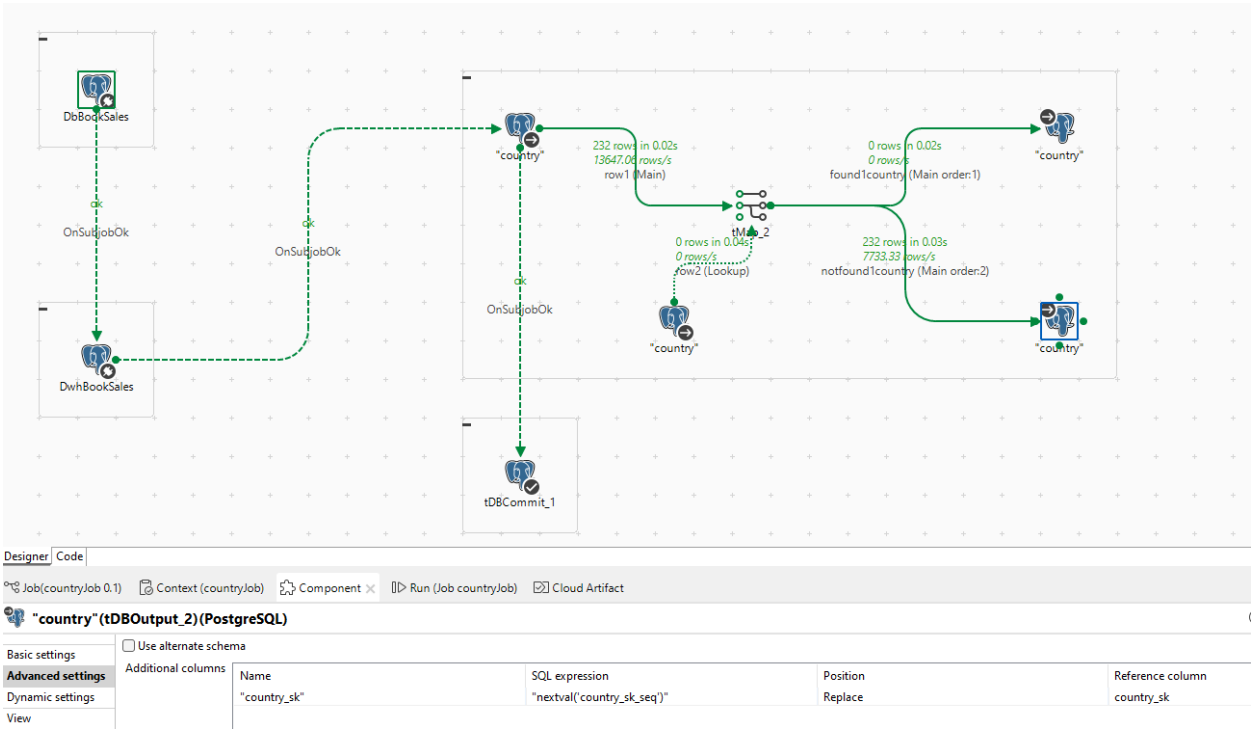
AddressStatus -



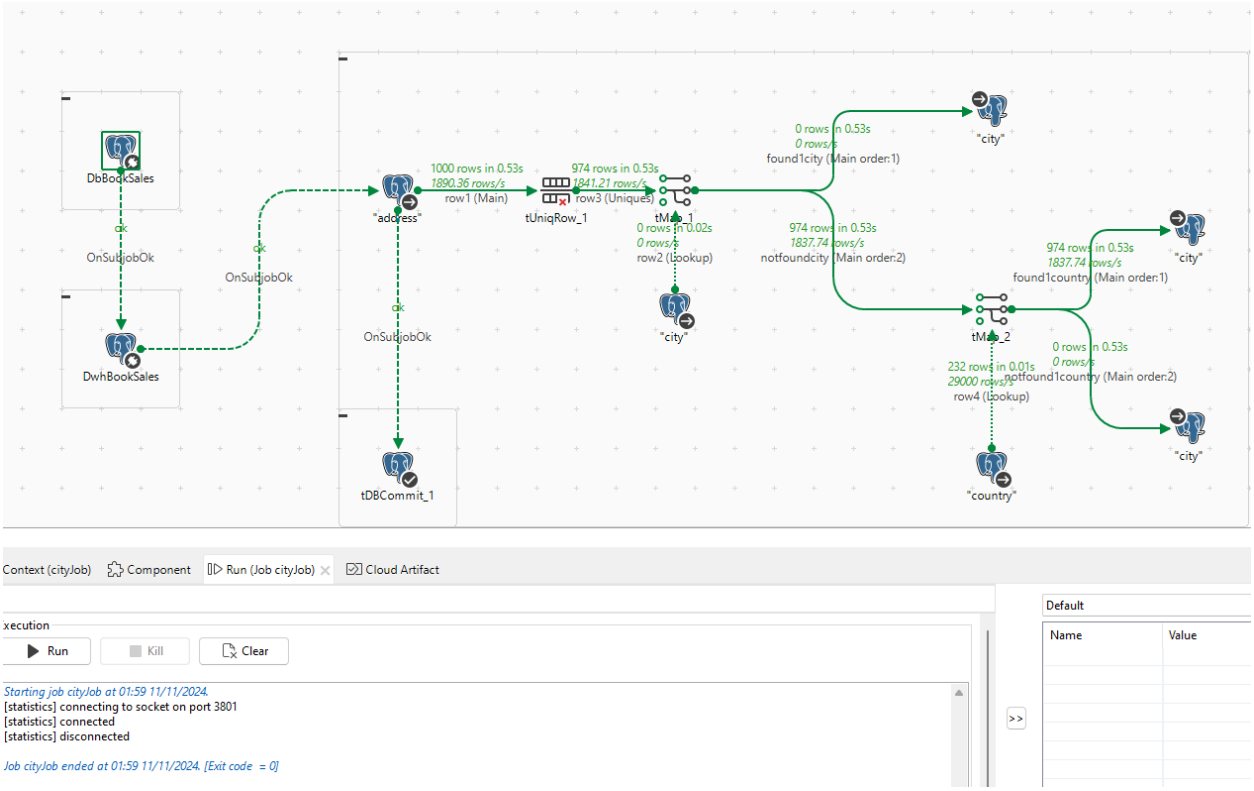
Customer –



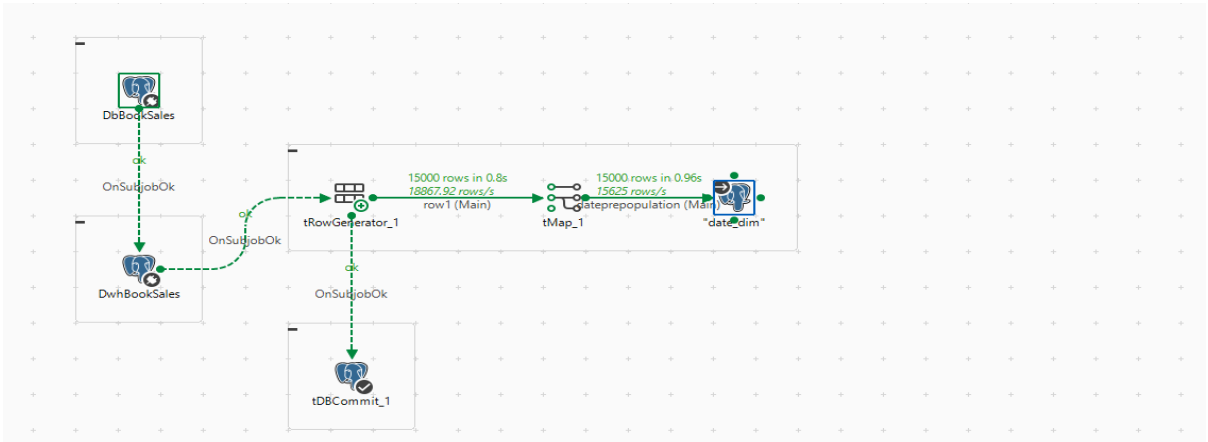
Country –

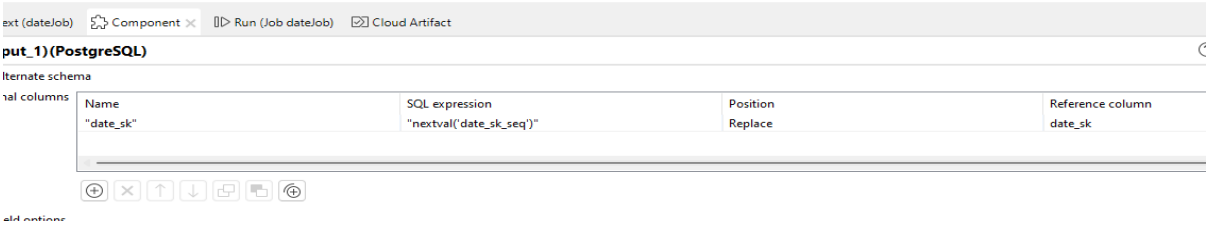


City-

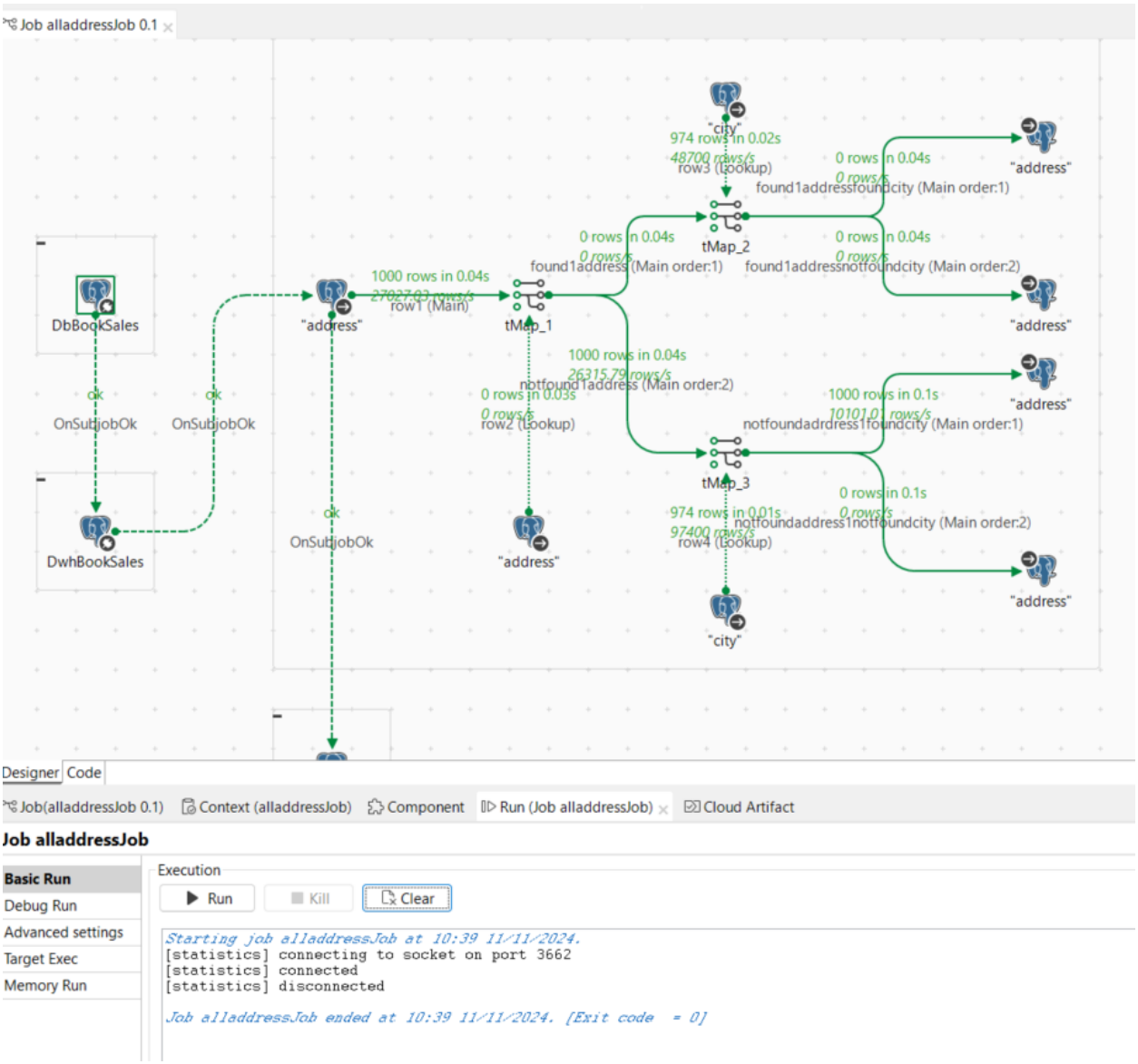


Date_dim-

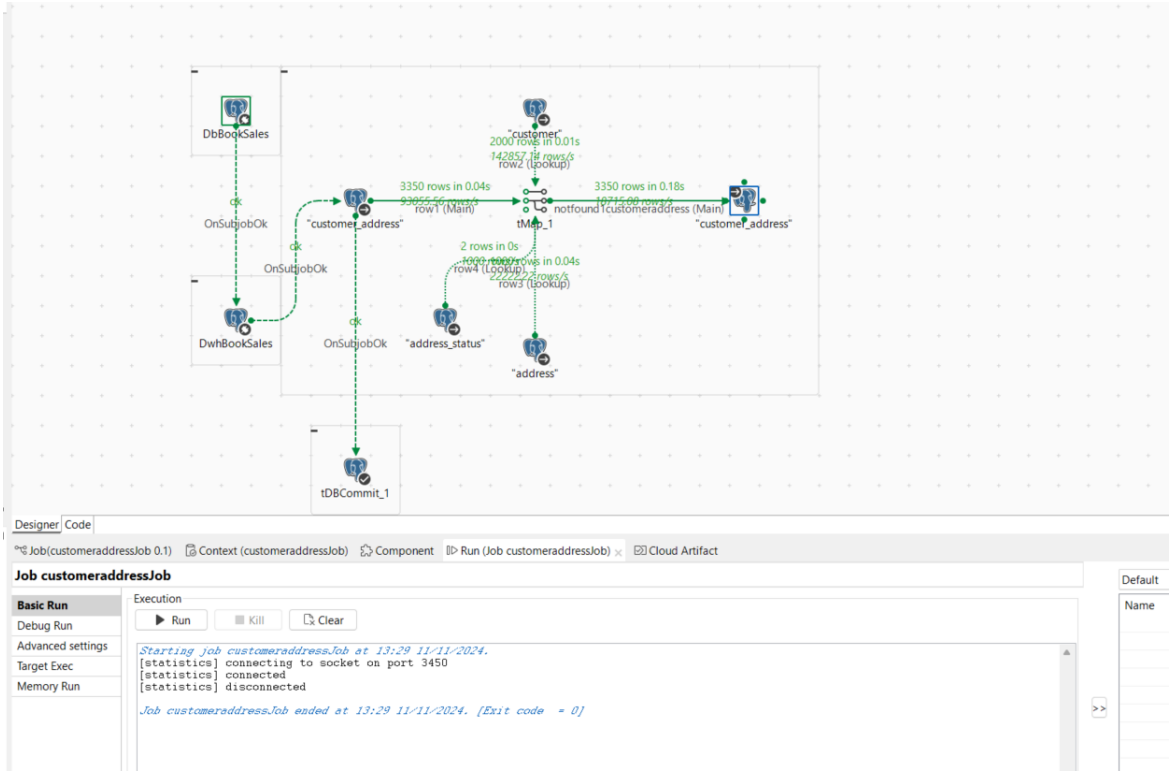




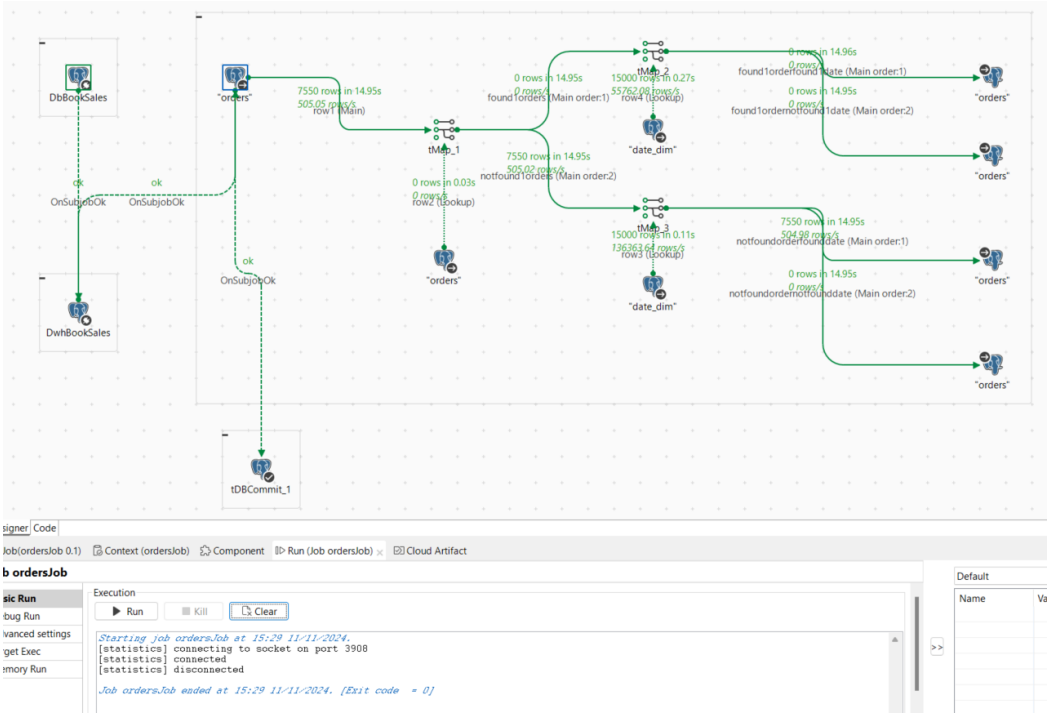
Address dimension –



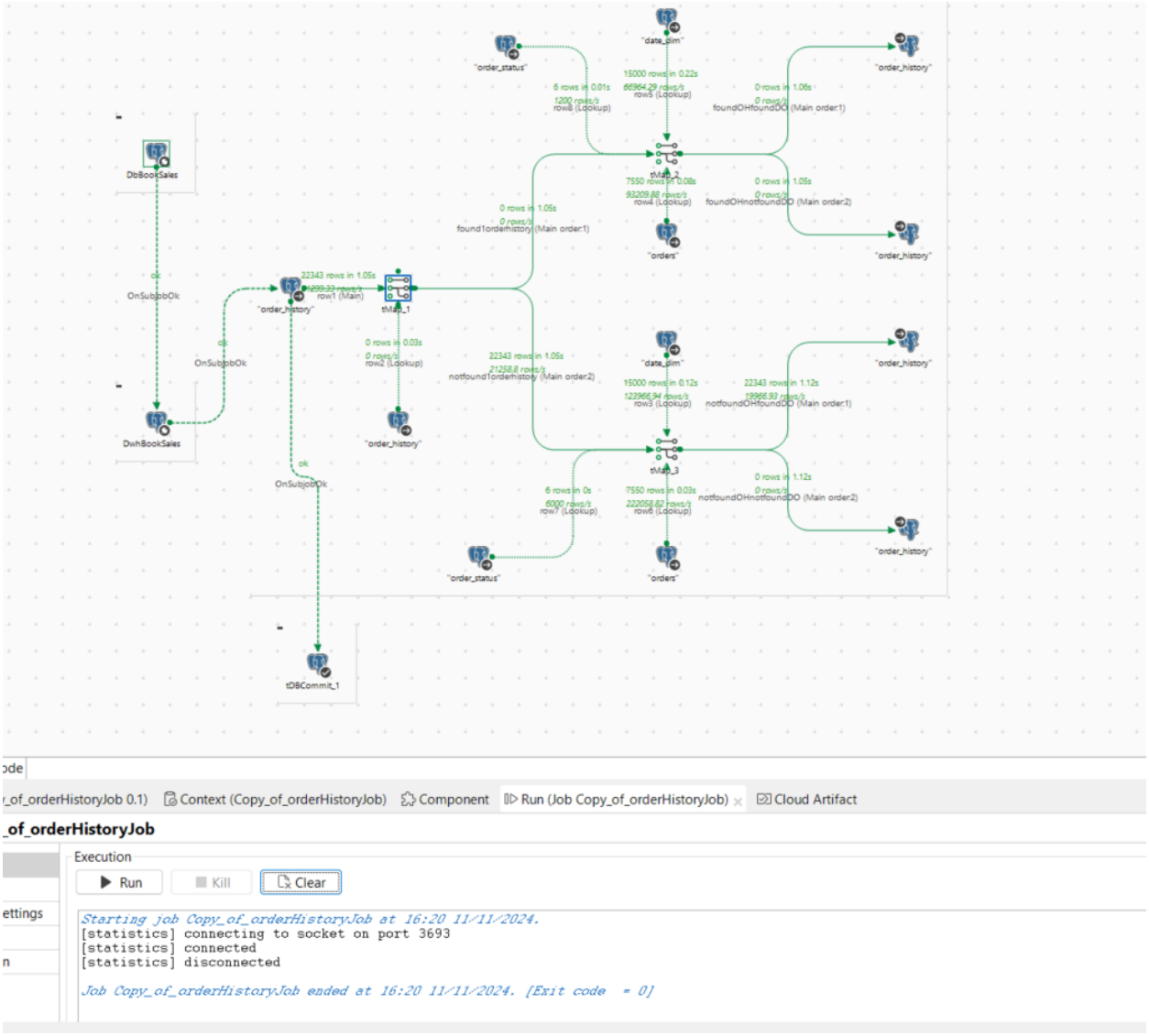
Customer Address -



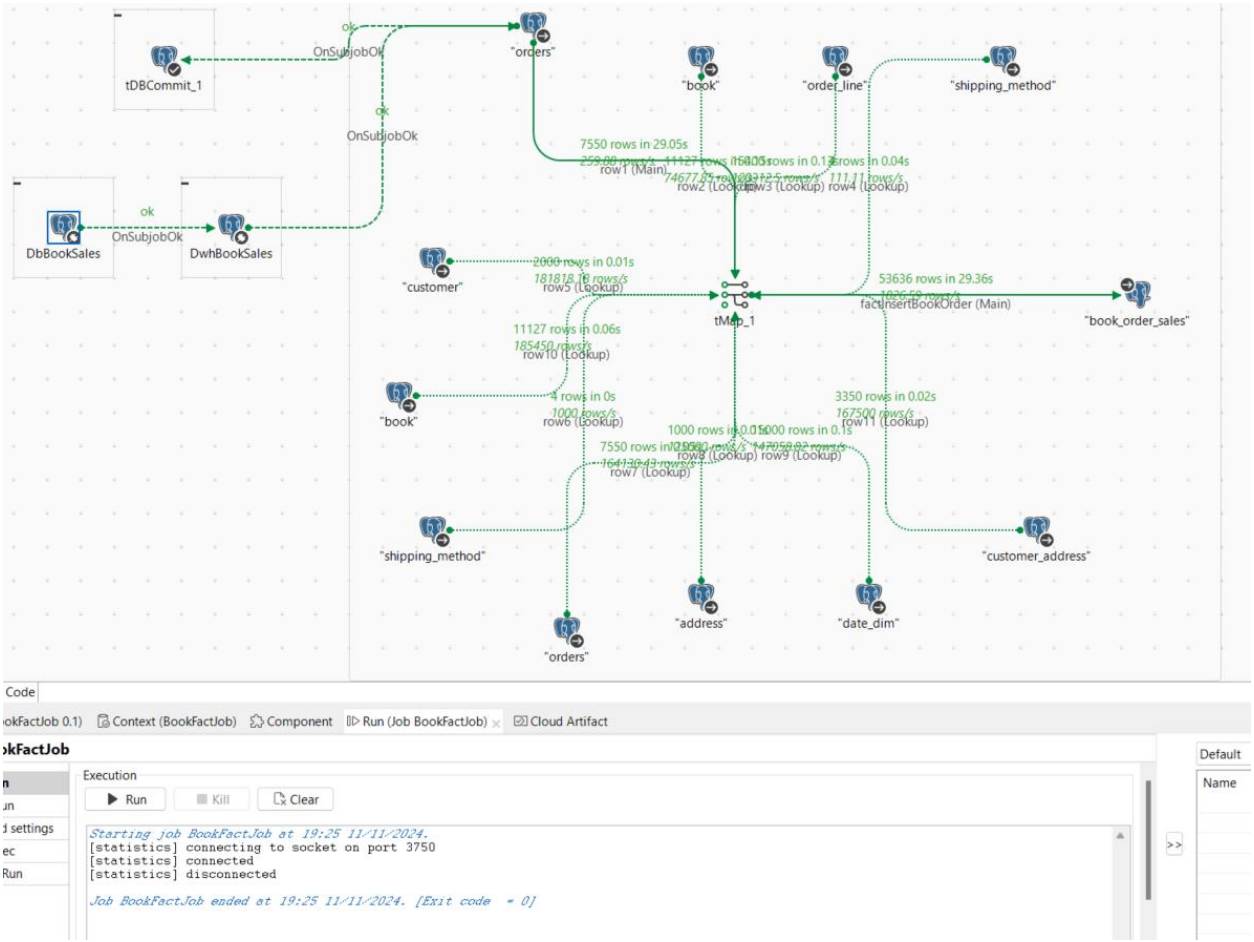
Orders –



OrderHistory -



BookOrdersSales Fact-



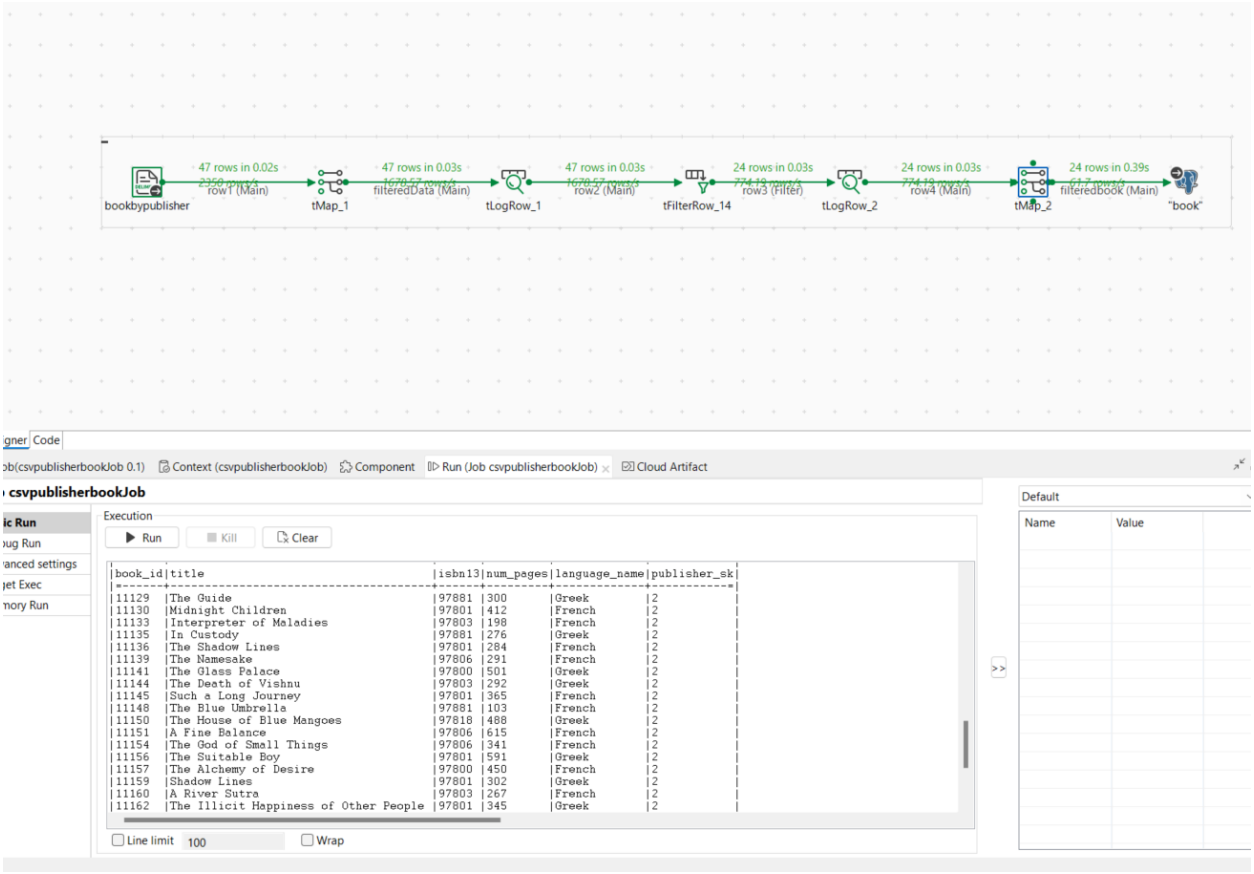
Sourcing Data from Different Source and Transformations:

Source: .csv file

Scenario: The existing publisher shared a csv file with new published books.

Transformations: From the csv file, the publisher wanted to include only “Greek” and “French” language books to be added. Thus, removing “Hindi” language books using trowfilter component.

Removed the books which has null isbn13 in the source file.



book_id	title	isbn13	num_pages	language_name	publisher_sk
11129	The Guide	97881	300	Greek	2
11130	Midnight Children	97801	412	French	2
11133	Interpreter of Maladies	97803	198	French	2
11135	In Custody	97881	276	Greek	2
11136	The Shadow Lines	97801	284	French	2
11139	The Namesake	97806	291	French	2
11141	The Glass Palace	97800	501	Greek	2
11144	The Death of Vishnu	97803	292	Greek	2
11145	Such a Long Journey	97801	365	French	2
11148	The Blue Umbrella	97881	103	French	2
11150	The House of Blue Mangoes	97818	488	Greek	2
11151	A Fine Balance	97806	615	French	2
11154	The God of Small Things	97806	341	French	2
11156	The Suitable Boy	97801	591	Greek	2
11157	The Alchemy of Desire	97800	450	French	2
11159	Shadow Lines	97801	302	Greek	2
11160	A River Sutra	97803	267	French	2
11162	The Illicit Happiness of Other People	97801	345	Greek	2

Slowly Changing Dimensions (SCD):

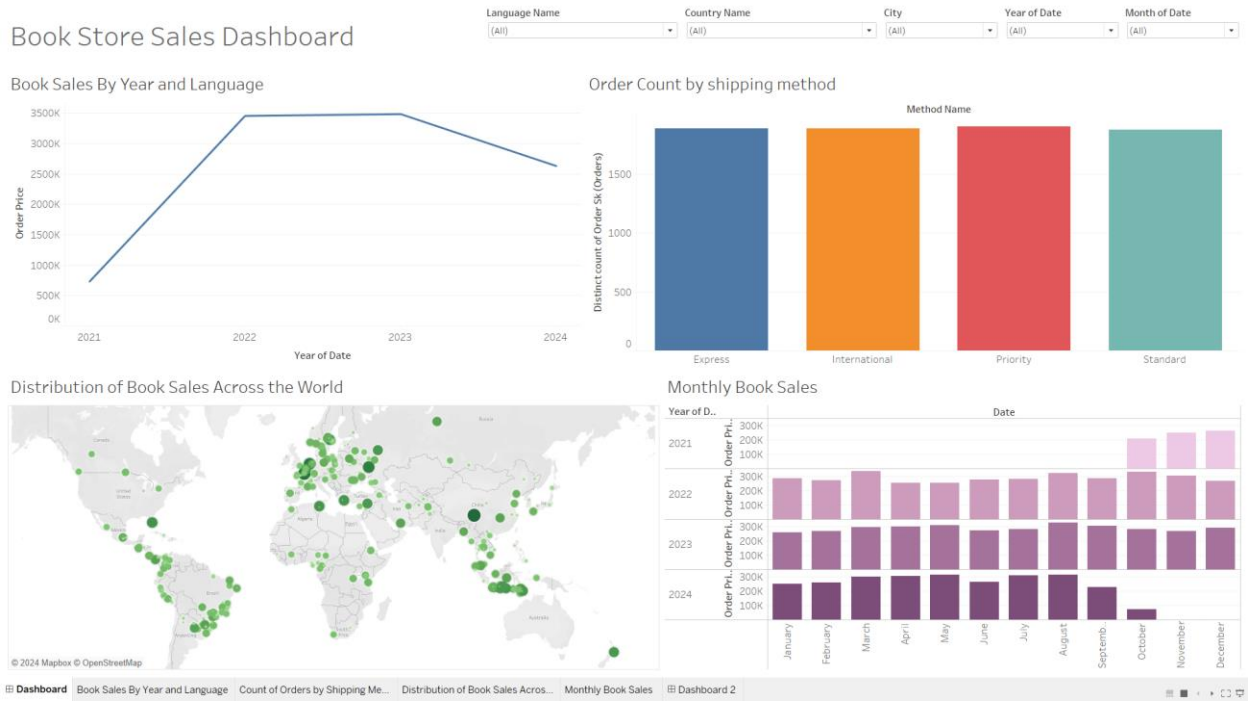
In our application, the Orders dimension is the Slowly Changing Dimension (SCD). This is because the order goes through multiple stages in terms of order status: order placed, shipped, out for delivery, delivered, cancelled, returned, etc.

To maintain this in an effective way, we use the Type 4 SCD and maintain an order_history as a mini-dimension.

This way, each successive status is stored in the order_history table as an archive, and the latest status can be found by sorting or filtering over the order_status_date column.

Also since we are implementing this as a Type 4 SCD, we will not need to make changes in the ETL pipeline specifically for Talend, since it is already maintained as a separate table.

Tableau dashboard to analyze the Book sales Data:



1. The first graph shows total book sales for every year, filtering by one or more languages. This helps the bookstore navigate the language that gives them the most sales every year and if there are any trends to be monitored for future book orders.
2. The second graph shows the number of orders separated by each shipping method. Since our original data was synthetically generated, all four methods have equal distribution. But in the real world we expect different levels and tracking them will guide the bookstore on whether certain methods need to be more efficient.
3. The third graph shows the distribution of total order sales across the map. This is a great way to visualize where all books end up going to. For example, India does not see any orders, China sees orders in a very few cities, but Europe has many cities where the orders go. This can help in focusing future advertising strategies by geography.
4. The final graph shows order sales by month over different years. This is a simple but effective graph to communicate how our order sales change over the year and between different years, and can help in planning for restocking, shipping redundancies, advertising etc.