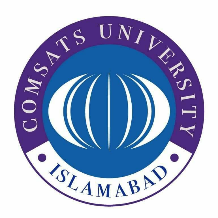
**COMSATS UNIVERSITY ISLAMABAD**



**Data Warehouse & BI**

**LAB MID**

**BSDS-6A**

* **Submitted By:**

***Manshah Hussain Bangash FA22-BDS-019***

***NOOR E EMAN MALIK FA22-BDS-034***

* **Submitted To:**

***Ma’am Nusrat Shaheen***

* **Date of Submission:**

**21st April*,2025***

**Books Data Warehouse**

**Project Overview**

The **Books Sales and Analytics Data Warehouse** is designed to enable efficient and comprehensive analytics for the book industry. The project leverages modern ETL processes using SQL Server Integration Services (SSIS) to transform raw operational data into a centralized data warehouse optimized for analytics and reporting.

**Purpose**

The primary goal is to provide a robust analytical platform that supports reporting, forecasting, and strategic decision-making in the book industry. This data warehouse will enable insights into sales trends, customer preferences, and market dynamics, allowing for targeted marketing, inventory optimization, and sales performance evaluation.

**Enhanced Data Structure**

**Technology Stack:**

* Microsoft SQL Server (2022)
* SQL Server Management Studio (SSMS)
* Power BI for visualizations and dashboards
* Integration of Python for advanced analytics

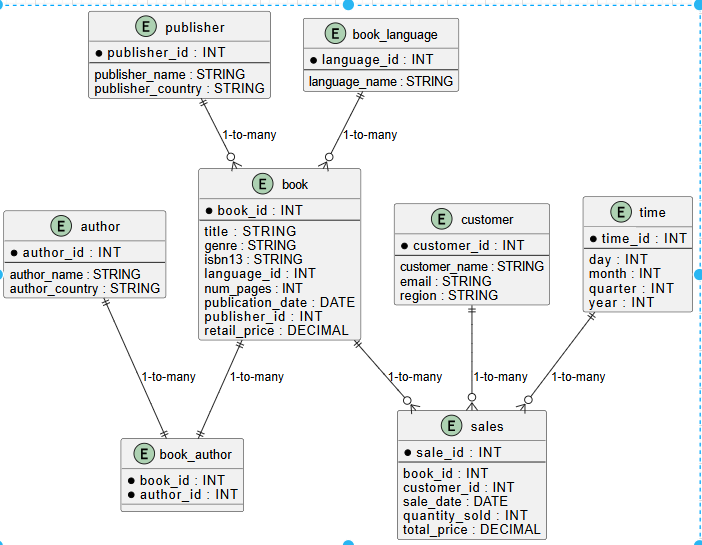
**Enhanced ERD**

We’ll retain the foundational entities but include **new tables** to enable advanced analytics.

**Tables and Entities:**

1. **author Table**
   * **Purpose:** Stores details about authors.
   * **Fields:**
     + author\_id (PK)
     + author\_name
     + author\_country
2. **publisher Table**
   * **Purpose:** Stores details about publishers.
   * **Fields:**
     + publisher\_id (PK)
     + publisher\_name
     + publisher\_country
3. **book\_language Table**
   * **Purpose:** Stores language information.
   * **Fields:**
     + language\_id (PK)
     + language\_name
4. **book Table**
   * **Purpose:** Stores detailed book information.
   * **Fields:**
     + book\_id (PK)
     + title
     + genre
     + isbn13
     + language\_id (FK)
     + num\_pages
     + publication\_date
     + publisher\_id (FK)
     + retail\_price
5. **book\_author Table**
   * **Purpose:** Represents the many-to-many relationship between books and authors.
   * **Fields:**
     + book\_id (FK)
     + author\_id (FK)
6. **customer Table**
   * **Purpose:** Captures customer details.
   * **Fields:**
     + customer\_id (PK)
     + customer\_name
     + email
     + region
7. **sales Table** (Fact Table)
   * **Purpose:** Central fact table for book sales.
   * **Fields:**
     + sale\_id (PK)
     + book\_id (FK)
     + customer\_id (FK)
     + sale\_date
     + quantity\_sold
     + total\_price
8. **time Table**
   * **Purpose:** Captures temporal details.
   * **Fields:**
     + time\_id (PK)
     + day
     + month
     + quarter
     + year

**DATA STRUCTURE :ERD**



**Explanation of the ERD**

The Entity-Relationship Diagram (ERD) represents the relational structure of the Books Data Warehouse, capturing the entities, their attributes, and the relationships between them. The ERD is designed to store operational data in a normalized format, ensuring data integrity and supporting the ETL process to populate the data warehouse. Below is a detailed explanation of the ERD, including entities, attributes, relationships, and cardinalities:

1. **Entities and Attributes**:
   * **author**: Stores information about authors.
     + author\_id (Primary Key, INT): Unique identifier for each author.
     + author\_name (VARCHAR): Name of the author.
     + author\_country (VARCHAR): Country of the author.
   * **publisher**: Stores details about publishers.
     + publisher\_id (Primary Key, INT): Unique identifier for each publisher.
     + publisher\_name (VARCHAR): Name of the publisher.
     + publisher\_country (VARCHAR): Country of the publisher.
   * **book\_language**: Stores language information for books.
     + language\_id (Primary Key, INT): Unique identifier for each language.
     + language\_name (VARCHAR): Name of the language (e.g., English, Spanish).
   * **book**: Stores detailed information about books.
     + book\_id (Primary Key, INT): Unique identifier for each book.
     + title (VARCHAR): Title of the book.
     + genre (VARCHAR): Genre of the book (e.g., Fiction, Non-Fiction).
     + isbn13 (VARCHAR, UNIQUE): 13-digit ISBN for the book.
     + language\_id (Foreign Key, INT): References book\_language.language\_id.
     + num\_pages (INT): Number of pages in the book.
     + publication\_date (DATE): Date of publication.
     + publisher\_id (Foreign Key, INT): References publisher.publisher\_id.
     + retail\_price (DECIMAL): Retail price of the book.
   * **book\_author**: Represents the many-to-many relationship between books and authors (since a book can have multiple authors, and an author can write multiple books).
     + book\_id (Foreign Key, INT): References book.book\_id.
     + author\_id (Foreign Key, INT): References author.author\_id.
     + Primary Key: Composite key (book\_id, author\_id).
   * **customer**: Stores information about customers.
     + customer\_id (Primary Key, INT): Unique identifier for each customer.
     + customer\_name (VARCHAR): Name of the customer.
     + email (VARCHAR, UNIQUE): Customer’s email address.
     + region (VARCHAR): Geographic region of the customer.
   * **sales**: Stores transactional data about book sales (acts as the fact table in the star schema).
     + sale\_id (Primary Key, INT): Unique identifier for each sale.
     + book\_id (Foreign Key, INT): References book.book\_id.
     + customer\_id (Foreign Key, INT): References customer.customer\_id.
     + sale\_date (DATE): Date of the sale.
     + quantity\_sold (INT): Number of books sold in the transaction.
     + total\_price (DECIMAL): Total price of the sale.
   * **time**: Stores temporal information for time-based analysis.
     + time\_id (Primary Key, INT): Unique identifier for each time record.
     + day (INT): Day of the month.
     + month (INT): Month of the year.
     + quarter (INT): Quarter of the year (1–4).
     + year (INT): Year of the sale.
2. **Relationships and Cardinalities**:
   * **book to book\_language**: One-to-many. A language (book\_language) can be associated with many books, but each book has exactly one language. This is enforced by the language\_id foreign key in the book table.
   * **book to publisher**: One-to-many. A publisher can publish many books, but each book is published by exactly one publisher. This is enforced by the publisher\_id foreign key in the book table.
   * **book to author**: Many-to-many. A book can have multiple authors, and an author can write multiple books. This relationship is managed through the book\_author junction table, which links book.book\_id and author.author\_id.
   * **sales to book**: Many-to-one. Many sales can involve the same book, but each sale is associated with exactly one book. This is enforced by the book\_id foreign key in the sales table.
   * **sales to customer**: Many-to-one. Many sales can be made to the same customer, but each sale is associated with exactly one customer. This is enforced by the customer\_id foreign key in the sales table.
   * **sales to time**: Many-to-one (implied). Although not explicitly linked in the ERD, the sale\_date in the sales table can be joined with the time table for time-based analysis, assuming sale\_date corresponds to time\_id or another temporal key.
3. **Purpose in the Data Warehouse**:
   * The ERD serves as the foundation for the operational data store, capturing raw data from the book industry (e.g., books, authors, customers, sales).
   * The normalized structure (with separate tables for authors, publishers, and languages) minimizes data redundancy and ensures referential integrity through primary and foreign key constraints.
   * The book\_author table efficiently handles the many-to-many relationship, allowing flexible querying of books by multiple authors.
   * The sales table acts as the central fact table in the star schema, linking to dimension tables (book, customer, time) for analytical purposes.
4. **Example Analytical Support**:
   * The ERD supports queries like:
     + Finding all books by a specific author (using book\_author and book).
     + Analyzing sales by customer region (using sales and customer).
     + Tracking sales trends over time (using sales and time).

**Dimensional Modeling: Star Schema**

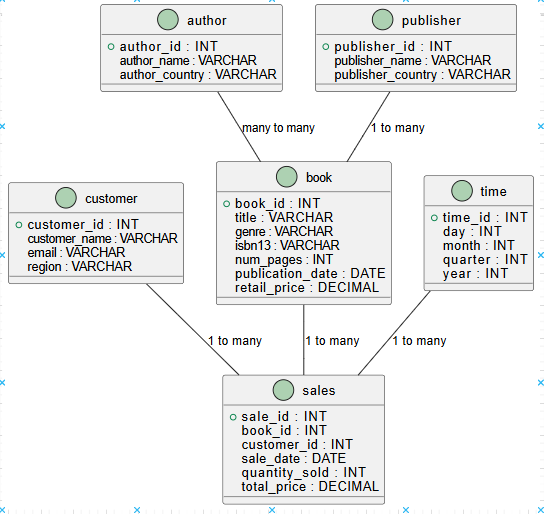
The updated schema focuses on analytics, including customer behavior and sales trends.

**Fact Table: sales**

* **Grain:** Each row represents a single book sale.
* **Measures:**
  + quantity\_sold
  + total\_price

**Dimension Tables:**

1. customer – Descriptive attributes for customers.
2. book – Details about the book, including genre and price.
3. author – Information about the author.
4. publisher – Data about the publisher, including country.
5. time – Temporal information for analytics.



**Short Description of the Star Schema**

The star schema is a dimensional modeling approach designed to optimize the Books Data Warehouse for analytical queries and reporting. It organizes data into a central **fact table** surrounded by **dimension tables**, resembling a star-like structure. This schema enhances query performance and simplifies analytics by denormalizing data for efficient aggregations.

* **Fact Table: sales**:
  + **Purpose**: Stores quantitative data about book sales, serving as the core of the star schema.
  + **Grain**: Each row represents a single book sale transaction.
  + **Measures**:
    - quantity\_sold (INT): Number of books sold in the transaction.
    - total\_price (DECIMAL): Total monetary value of the sale (calculated as quantity\_sold \* book.retail\_price).
  + **Foreign Keys**:
    - book\_id: Links to the book dimension.
    - customer\_id: Links to the customer dimension.
    - sale\_date (or implied time\_id): Links to the time dimension for temporal analysis.
* **Dimension Tables**:
  + **book (DimBook)**: Contains descriptive attributes about books, such as title, genre, isbn13, num\_pages, publication\_date, and retail\_price. It also links to publisher and author via foreign keys.
  + **customer (DimCustomer)**: Stores customer details, including customer\_name, email, and region, enabling analysis by customer demographics.
  + **time (DimDate)**: Captures temporal attributes (day, month, quarter, year) for time-based analysis, such as sales trends by month or quarter.
  + **author**: Provides author details (author\_name, author\_country) for analyzing sales by author.
  + **publisher**: Contains publisher information (publisher\_name, publisher\_country) for publisher-based analytics.
* **Analytical Value**:
  + The star schema supports efficient querying for business intelligence tasks, such as:
    - Aggregating sales by genre, region, or time period.
    - Identifying top-selling authors or publishers.
    - Analyzing customer purchasing patterns.
  + The denormalized structure of dimension tables reduces the need for complex joins, making queries faster and more intuitive for tools like Power BI.
* **Example Query**:

SELECT t.year, b.genre, SUM(s.total\_price) AS total\_sales

FROM sales s

JOIN book b ON s.book\_id = b.book\_id

JOIN time t ON s.sale\_date = t.time\_id

GROUP BY t.year, b.genre;

This query calculates total sales by genre and year, leveraging the star schema’s structure

**DDL:**

**author Table**

CREATE TABLE author (

author\_id INT PRIMARY KEY,

author\_name VARCHAR(255) NOT NULL,

author\_country VARCHAR(100)

);

**2. publisher Table**

CREATE TABLE publisher (

publisher\_id INT PRIMARY KEY,

publisher\_name VARCHAR(255) NOT NULL,

publisher\_country VARCHAR(100)

);

**3. book\_language Table**

CREATE TABLE book\_language (

language\_id INT PRIMARY KEY,

language\_name VARCHAR(50) NOT NULL

);

**4. book Table**

CREATE TABLE book (

book\_id INT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

genre VARCHAR(100),

isbn13 VARCHAR(13) UNIQUE NOT NULL,

language\_id INT,

num\_pages INT,

publication\_date DATE,

publisher\_id INT,

retail\_price DECIMAL(10, 2),

FOREIGN KEY (language\_id) REFERENCES book\_language(language\_id),

FOREIGN KEY (publisher\_id) REFERENCES publisher(publisher\_id)

);

**5. book\_author Table (Many-to-Many relationship between books and authors)**

CREATE TABLE book\_author (

book\_id INT,

author\_id INT,

PRIMARY KEY (book\_id, author\_id),

FOREIGN KEY (book\_id) REFERENCES book(book\_id),

FOREIGN KEY (author\_id) REFERENCES author(author\_id)

);

**6. customer Table**

CREATE TABLE customer (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(255) NOT NULL,

email VARCHAR(255) UNIQUE,

region VARCHAR(100)

);

**7. sales Table (Fact Table)**

CREATE TABLE sales (

sale\_id INT PRIMARY KEY,

book\_id INT,

customer\_id INT,

sale\_date DATE,

quantity\_sold INT,

total\_price DECIMAL(10, 2),

FOREIGN KEY (book\_id) REFERENCES book(book\_id),

FOREIGN KEY (customer\_id) REFERENCES customer(customer\_id)

);

**8. time Table (Dimension Table for temporal data)**

CREATE TABLE time (

time\_id INT PRIMARY KEY,

day INT,

month INT,

quarter INT,

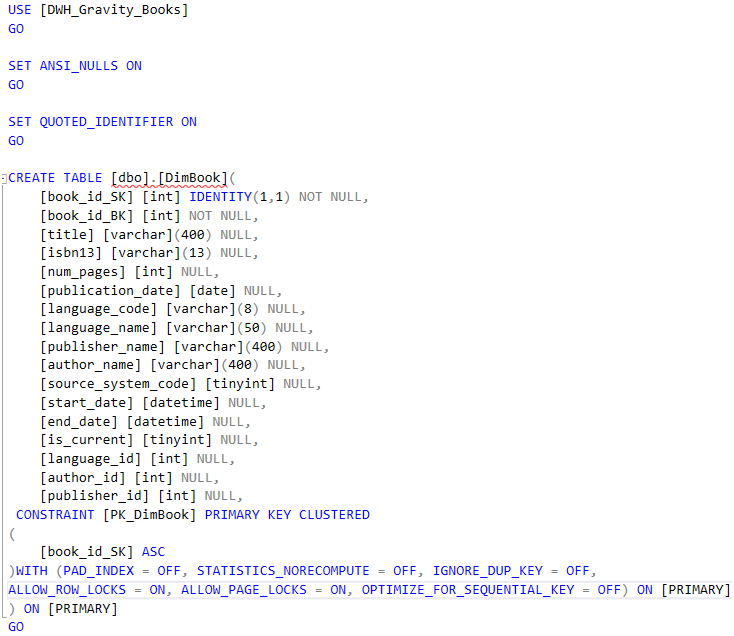
year INT

);

**Explanation:**

* **Primary Keys (PK)**: These are the unique identifiers for each table (e.g., author\_id, publisher\_id).
* **Foreign Keys (FK)**: These link the dimension tables to the fact table (e.g., book\_id, customer\_id), ensuring referential integrity.
* **Relationships**: The book\_author table handles the many-to-many relationship between books and authors.

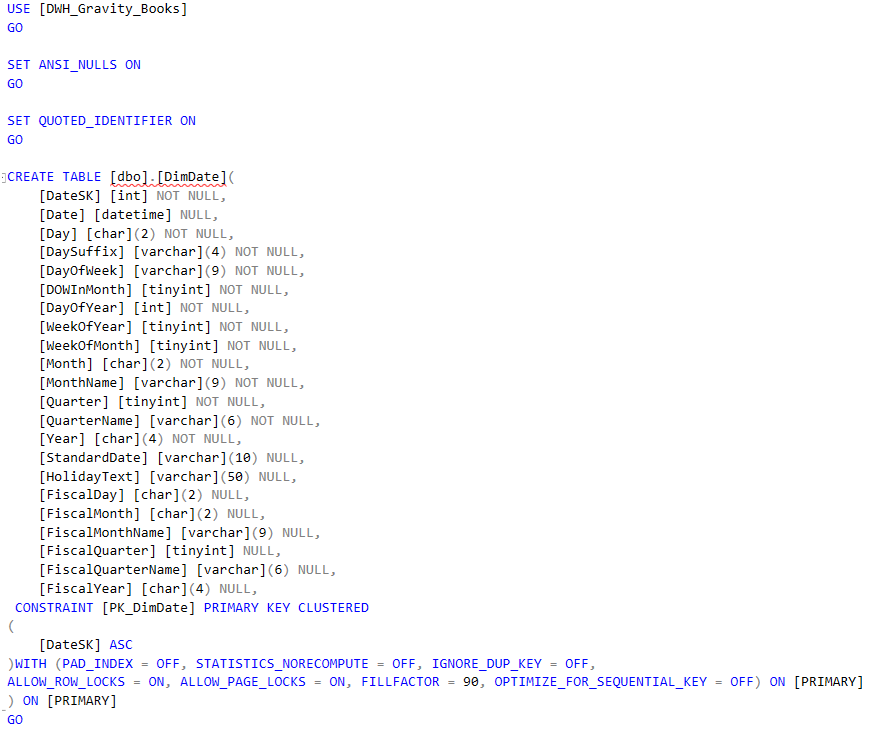
**DimBook Table:**

****

**DimCustomer Table:**

****

**DimDate Table:**

****

**Data Transformation:**

**Overview of Data Transformations:**

1. **Join Transformations:**
   * **Purpose:** Combine data from different tables based on defined relationships.
   * **Examples:**
     + Merging the "book" table with the "author" and "publisher" tables to create a unified dataset for books.
2. **Derivation Transformations:**
   * **Purpose:** Generate derived fields or calculated measures using existing data.
   * **Examples:**
     + Calculating the total cost of an order by summing the prices of individual products.
     + Determining customer age from their birthdate.
3. **Dimensional Modeling Transformations:**
   * **Purpose:** Transform data into a dimensional model to optimize reporting and analytics.
   * **Examples:**
     + Organizing data into a star schema for easier access and analysis.
     + Creating fact and dimension tables as part of the overall data warehouse structure.

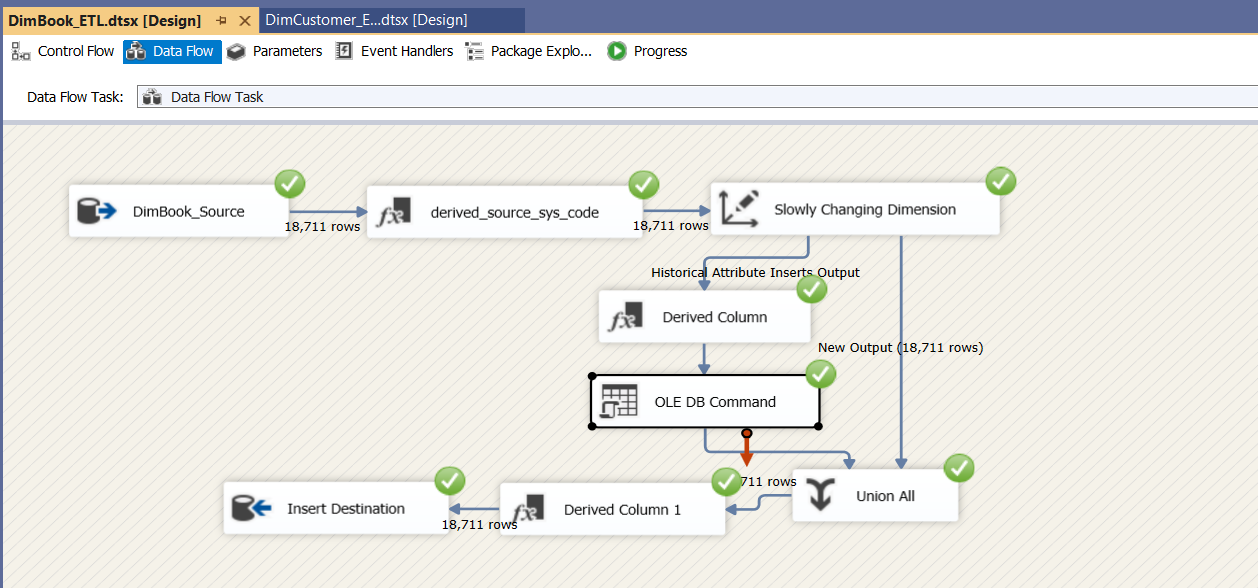
**Business Rules:**

1. **Handling Null or Missing Values:**
   * Define how null or missing values are treated. Decide whether they should be ignored, treated as zeros, or handled in another way to ensure the accuracy of accumulated data.
2. **Managing Changes in Dimensional Data:**
   * Address how updates to dimensional data, like changes in customer or product information, are managed in the accumulated fact table. Establish rules for handling changes in dimension keys to preserve the historical accuracy of the data.
3. **Effective Dating:**
   * Implement effective dating to track when changes occur over time. Specify how the accumulated fact table will record effective date ranges for each record, ensuring clarity about when specific values were added or modified.

**ETL Transformation Steps:**

**DimBook:**

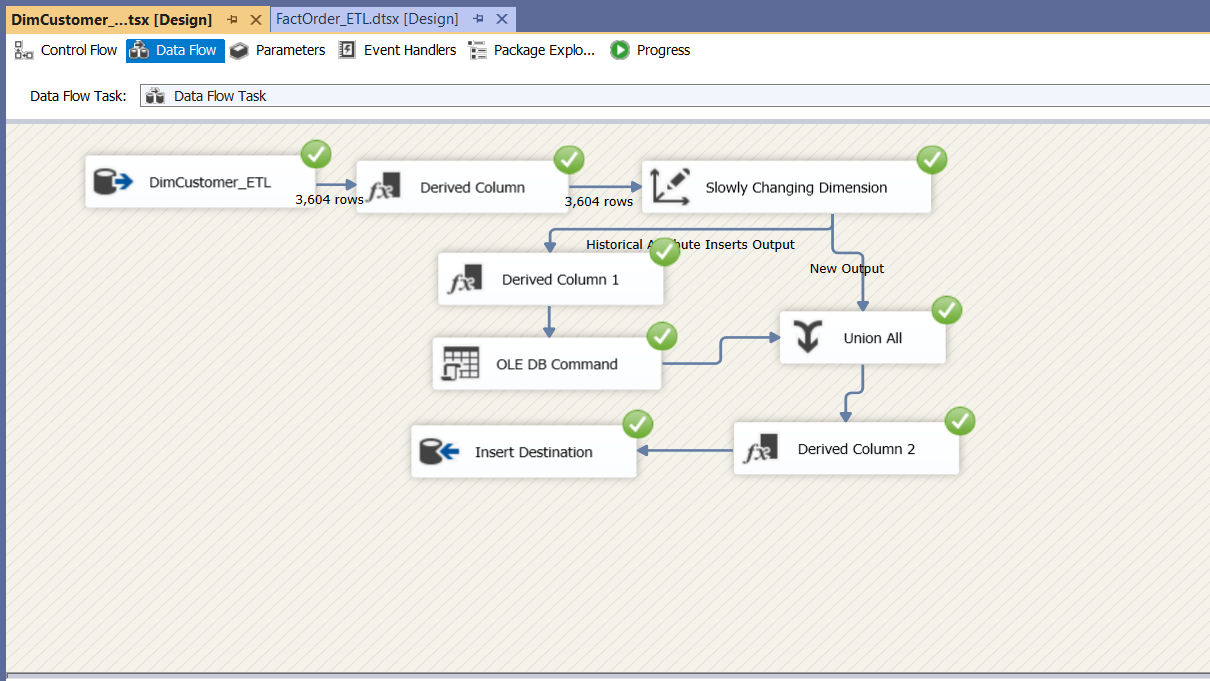
* **SCD (Slowly Changing Dimension) Task:**
  + Manage changes in the **DimBook** dimension to track new and updated book records over time.
* **OLE DB Source:**
  + Extract book data from the source database using OLE DB.
* **OLE DB Command:**
  + Use OLE DB Command to directly update existing records in the destination (such as updates detected during the SCD processing).
* **Union All Transformation:**
  + Merge multiple datasets, combining various sources of book-related data (e.g., book titles, authors, and publishers).
* **Derived Column Transformation:**
  + Modify or create new columns during the transformation process (e.g., reformatting book titles or calculating new fields).



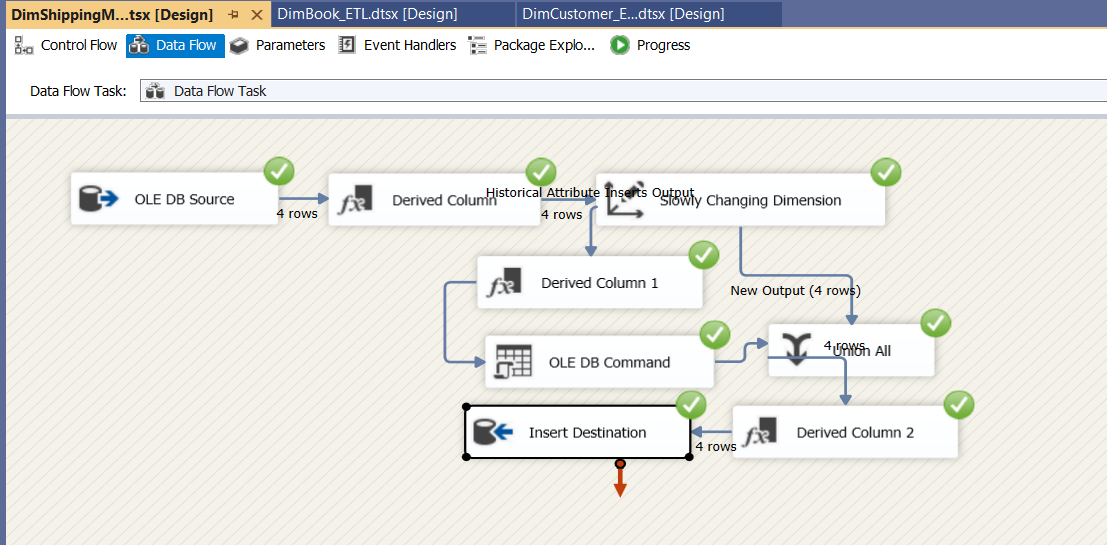
* **DimCustomer**

**SSIS Tasks and Containers:**

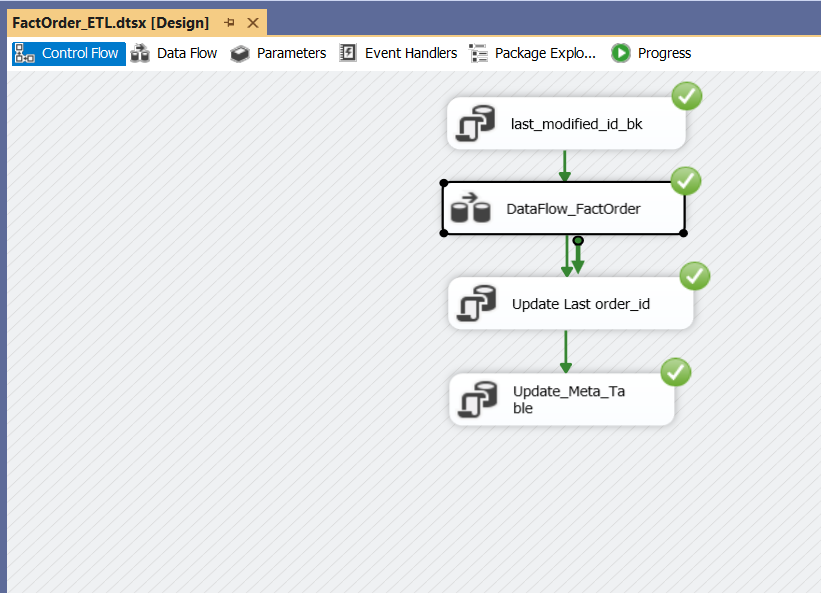
* **OLE DB Source:** Extracts customer data.
* **Derived Column:** Creates or modifies columns like full address or formatted contact numbers.
* **SCD Transformation:** Tracks and manages changes in customer records.

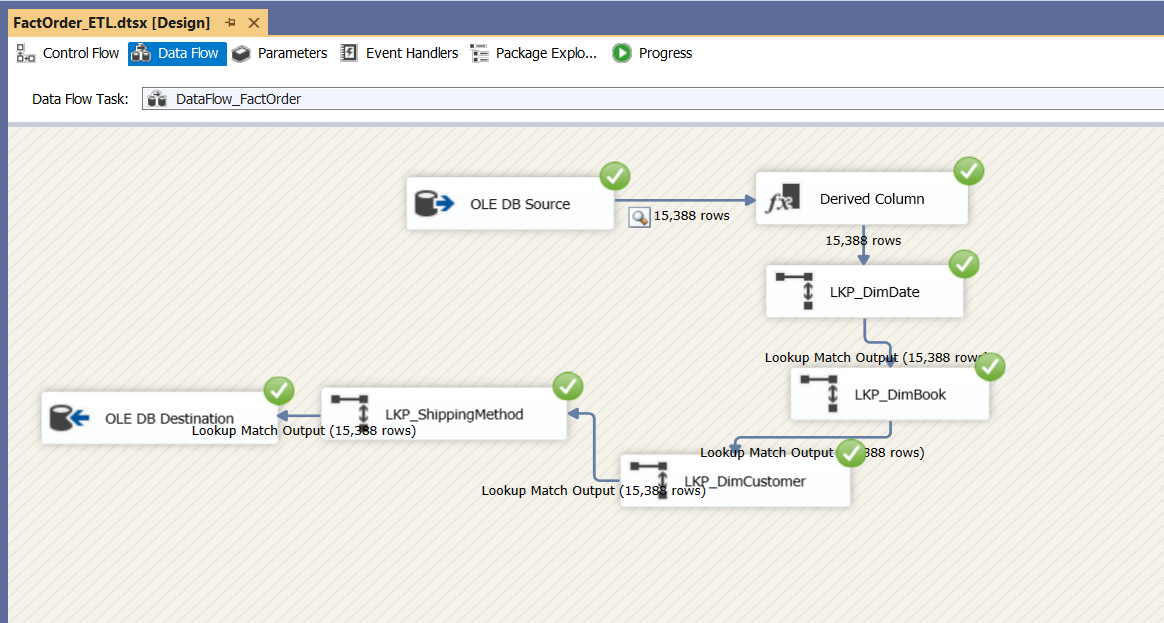


* **DimShippingMethod**



* **FactOrder**





**Conclusion:**

The Books Data Warehouse project has been a transformative journey, providing invaluable insights into ETL (Extract, Transform, Load) processes. This project introduced several new concepts, notably the creation of an accumulated fact table, which played a crucial role in improving the data warehousing architecture. The process began with a structured design phase, where the data warehouse schema was developed using MS SQL Server, followed by the implementation of ETL processes through SSIS. The project smoothly progressed from planning to execution, with a focus on building the accumulated fact table, applying dimensional modeling, and incorporating business rules and data mappings to create a highly efficient and robust data warehouse.