**Final Report: Online Retail Data Warehouse**

**Title Page**

* **Title**: Comprehensive Analysis of Data Warehousing and Relational vs. Graph Database Systems

**Table of Contents**

1. Source of Data
2. Vision and Business Requirements
3. Data Warehouse Design
4. ETL Process
5. Reports and Visualizations
6. Relational vs Graph Databases
7. References/Bibliography
8. Appendix

**1. Source of Data**

**Dataset**

* **Name**: Online Retail Dataset
* **Source**: UCI Machine Learning Repository
* **Description**:  
  The dataset used in the study accrued from a large UK-based online retailer firm, which was active between **December 2010 and December 2011**. Transaction details as well as other fields such as invoice numbers, product specifications, customers and their prices, quantity, etc are well captured in each row of the dataset. Below is a detailed breakdown of the dataset fields:
  + **InvoiceNo**: A unique code number by which each transaction or invoice can be identified.
  + **StockCode**: A tracking number for the specific product being sold.
  + **Description**: This contains a brief description of the product.
  + **Quantity**: Quantity is the number of units of the product sold in the particular transaction.
  + **InvoiceDate**: The date and time of the transaction being accounted for.
  + **UnitPrice**: The cost price of the product measured at the level of price of one unit of the product.
  + **CustomerID**: A unique identifier of the customer who makes the purchase; Some of the fields amongst the rows may contain no value.
  + **Country**: The nation where the customer resides.

**Reason for Selection**

* **Rich Transactional Data**:

The dataset offers complete insight into customers and their overall trends in terms of sales and revenue of the products. It’s best used with historical data such as the revenue records of a business enterprise and it provides insights on that data.

* **Business Intelligence Use Cases**:  
  The dataset supports the following business use cases:
  1. **Customer Segmentation**: Segmentation of buyers according to their consumption pattern, geographical location or level of consumption.
  2. **Sales Trends**: Identifying the type of revenue, for instance, distinguishing between the periods of high and low selling.
  3. **Product Performance Analysis**: Understanding which of the products creates the most revenue or is bought most often.
  4. **Revenue Drivers**: Identifying customer type and product type that generate the most sales revenue for the business.
* **Versatility**:
  1. Due to the transactional nature of the dataset, it is ideal for a data warehouse environment where it is structured in a star schema where dimensions are Customer, Product and Date and the fact table for sales.
* **Relevance to Business**:
  1. Different use cases of data warehouses include retail data, and the dataset is as complex as real-world issues, such as customer behavior analysis, product popularity, and revenue prediction.

**2. Vision and Business Requirements**

**Business Vision**

* **Objective**:  
  For the purpose of designing a **centralized data warehouse** that would allow for providing insights to drive business decisions based on customer activity, product and revenue data. This allows stakeholders to obtain usable knowledge and enhance business operation plans.
* **Stakeholders**:
  + **Sales Managers**: Demonstrate deep understanding of key products and customer segments necessary for high sales and adjusting the revenue model.
  + **Marketing Team**: Employ customer targeting for campaigns and marketing to achieve better engagement and help the overall marketing efforts yield more profits.
  + **Executives**: Looking for top-tier KPIs for strategic decision-making tailored to the company’s business strategy.

**Key Use Cases**

* **Identify Top-Performing Products**: Identity the products that generate the most revenue and or high sales volume to know which products help the organization achieve its goals.
* **Segment Customers by Country and Revenue Contribution**: Market segmented based on location and revenue to target those customers that are most valuable and to develop targeted marketing communications.
* **Analyze Monthly Sales Trends**: Analyze variability in monthly revenues as a way of identifying periods of high sales and then plan for them.

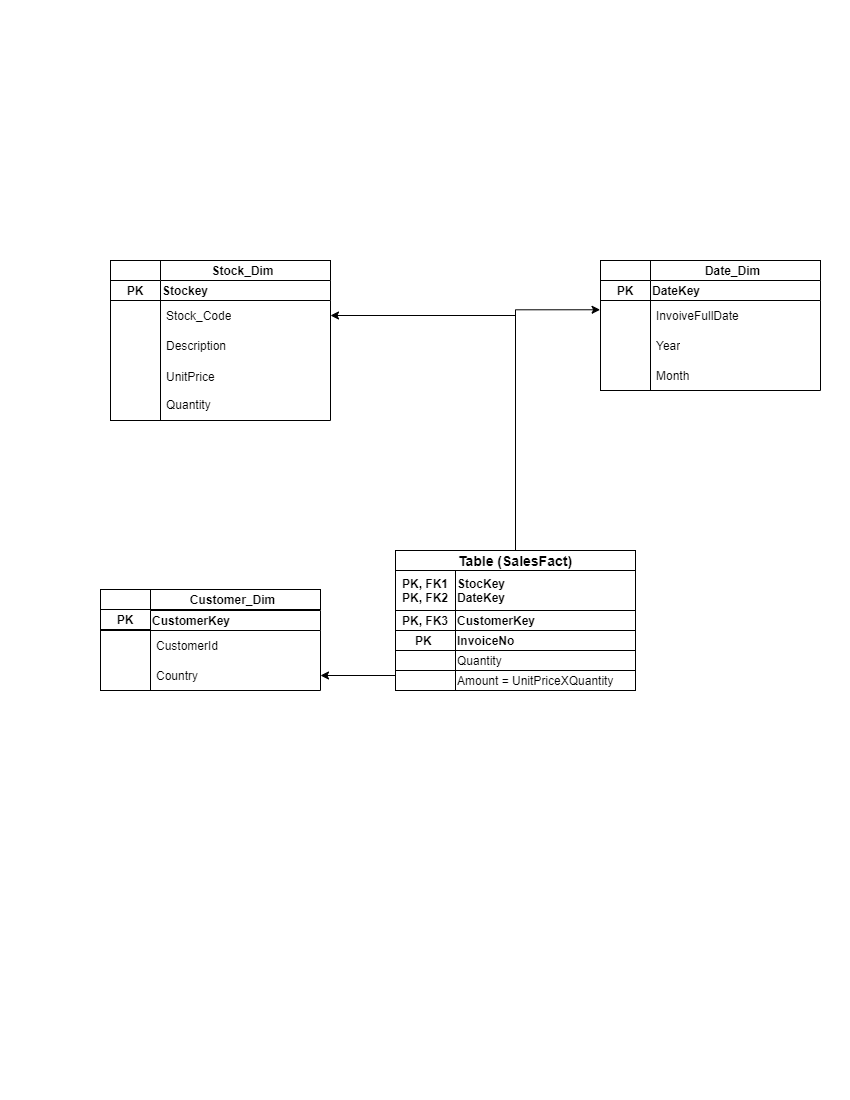
**3. Data Warehouse Design**

**What is a Star Schema?**

A Star Schema is a type of Dimensional model which forms the core strategy of data storage system of a data warehouse. It is essentially made of a fact table surrounded by multiple dimension tables and resembles a star when represented graphically.

**Schema Diagram**

A **Star Schema** was implemented with the following structure:



* **Fact Table**: Consists of primary key and fields of measures that contain the actual numbers to be analyzed and foreign keys to the dimension tables.  
  Example: FactSales stores **Quantity** and **Revenue** along with foreign keys to Customer\_Dim, Stock\_Dim, and Date\_Dim.
* **Dimension Tables**: It contains descriptive attributes for dimensioning and slicing the facts in a fact table.
* Example:
  + Customer\_Dim: Contain attributes like CustomerID and Country and describe customers.
  + Stock\_Dim: Contains StockCode and Description attributes to describe Product.
  + Date\_Dim: It represents attributes of time like Year, Month, and Day.

**Design Rationale**

* The star schema is good for analytical queries and it increases the ease of slicing and dicing the data.
* Measures are summarized in fact table while details description is in dimension table.

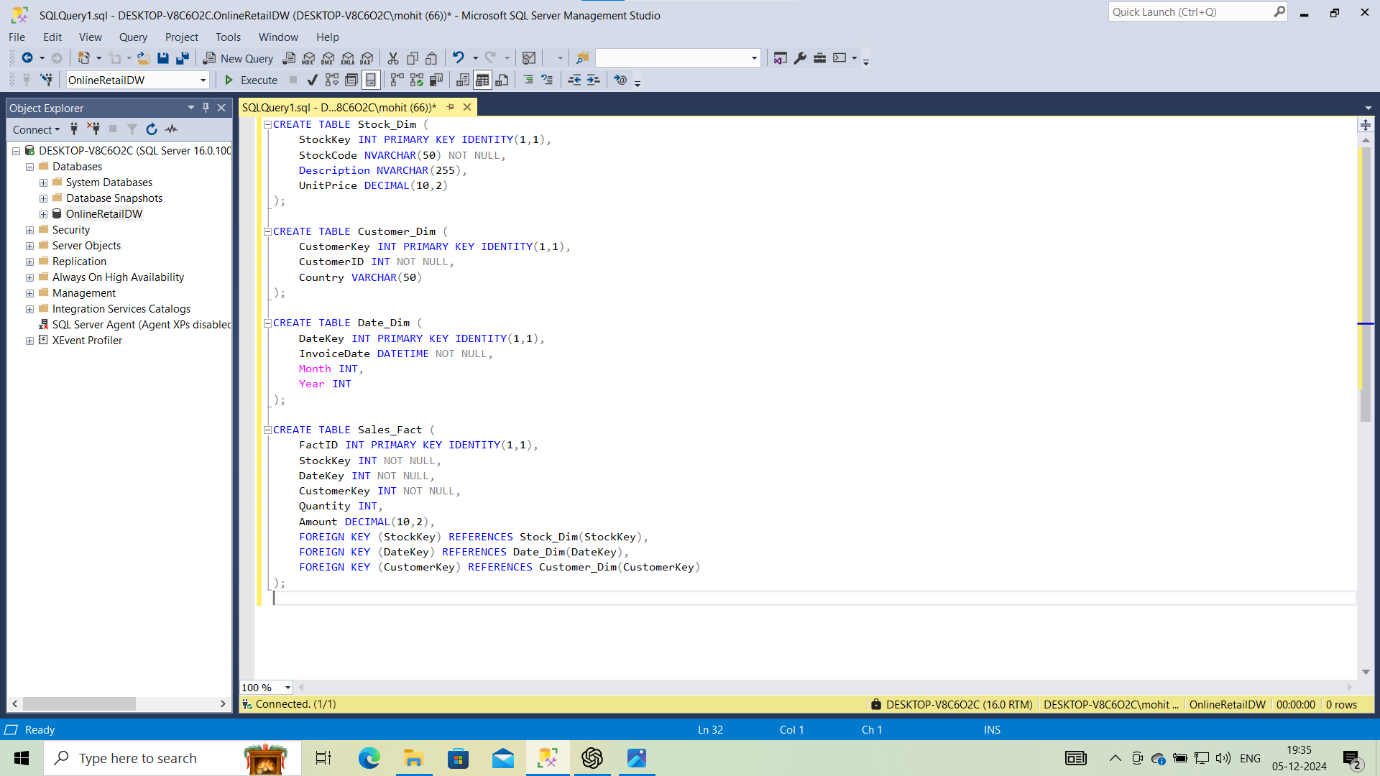
**Justification for Design**

1. **Fact Table (FactSales)**:

* It is the central database for storing many transactional data.
* Measures attributes like Quantity and Revenue which are imperative when analyzing.
* Links to dimensions using foreign keys connecting products and customer segments to drive sales strategies.

1. **Dimension Tables**:
   * Provide context to the measures in the fact table:
     + **Customer\_Dim**: Allows the division of sales data by the customer or geographical location.
     + **Stock\_Dim**: Enables a performance evaluation of the products.
     + **Date\_Dim**: Supports data visualization and trend analysis by time series.
2. **Ease of Maintenance**:
   * Dimensions can be added or removed with ease and new measures can also be added to dimension or fact table easily without straining the whole structure.
3. **Scalability**:
   * The dimensions and fact table can expand without affecting the other is one advantage of the schema.

**SQL Scripts**

****

**4. ETL Process: Extract, Transform, Load (ETL)**

The ETL process of **Online Retail Dataset** has three distinct phases: **Extract**, **Transform**, and **Load**. This structured approach allows that raw data is cleaned, normalized, and loaded for effective analysis into star schema.

**Steps in Detail**

**1. Extract Phase**

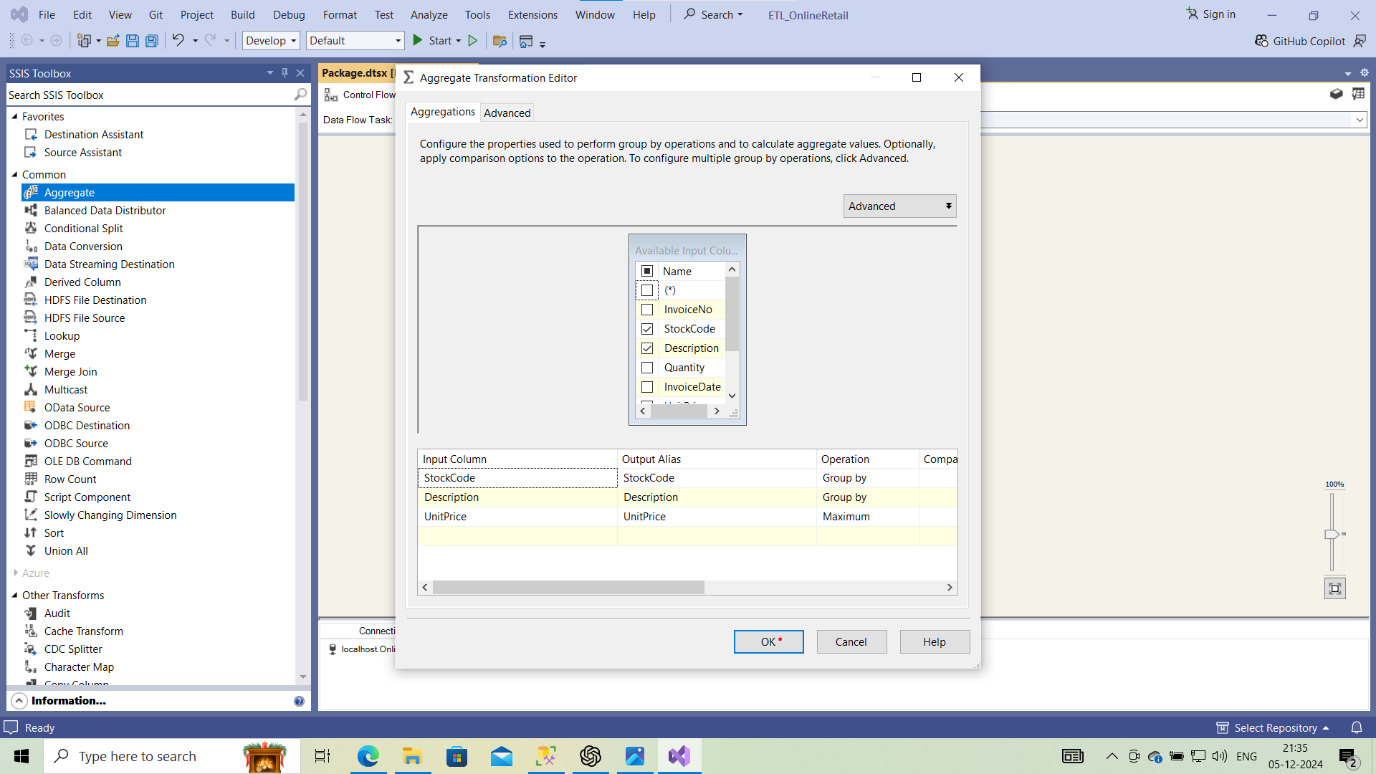
* **Objective**: For preprocessing, import raw data from OnlineRetail.csv file into a staging area.
* **Process**: The raw csv Loaded into the staging table in SQL Server for further processing.
* **Outcome**: Raw data is filled into OnlineRetail Staging table which preceded cleaning and transformation process successfully for analysis.

**2. Transform Phase**

* **Objective**: Clean and normalize the data into separate dimension and fact tables that are appropriate for the star schema
* **Process**:
  1. **Data Cleaning**:
     + Removed rows with null values in critical fields (e.g., CustomerID) and invalid or duplicate records from staging table.
  2. **Dimension Creation**:
     + Extract and load unique data inputs into dimension tables.
       - **Customer Dimension (Customer\_Dim)**:
       - **Stock Dimension (Stock\_Dim)**:
       - **Date Dimension (Date\_Dim)**:
  3. **Fact Table Creation**:
     + Load transactional data into the FactSales table.
     + Calculated Revenue as Quantity \* UnitPrice.
* **Outcome**: The data was loaded and converted into (Customer\_Dim, Stock\_Dim & Date\_Dim) and FactSales table.

**3. Load Phase**

* **Objective**: Prepare and load star schema in the data warehouse for further analytics of different queries.
* **Process**:
  + For loading data into the star schema, used **SQL scripts** and **SSIS workflows.**
  + **SSIS Control Flow**: Made tasks for importing and executing transformation scripts for CSV file.
  + **SSIS Data Flow**: Data was extracted from the “OnlineRetail” staging table and applied various transformations for filtering null values and mapping keys so as to load cleaned data in the dimensions and fact table.
* **Outcome**: FactSales and all its dimensions were fully populated and available for analytical querying and reporting using star schema.

****

**Screenshots and Visuals**

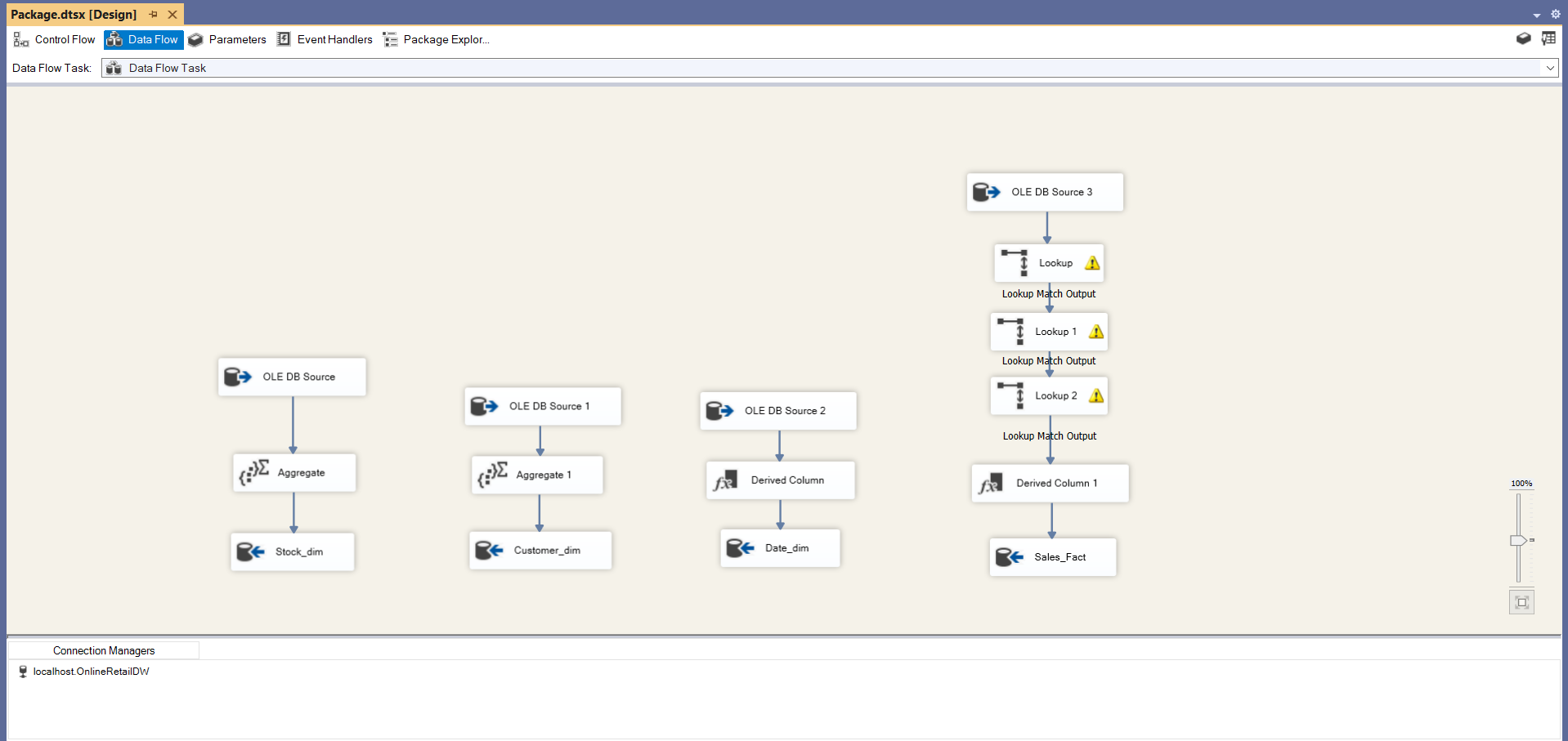
**1. SSIS Control Flow Tasks**

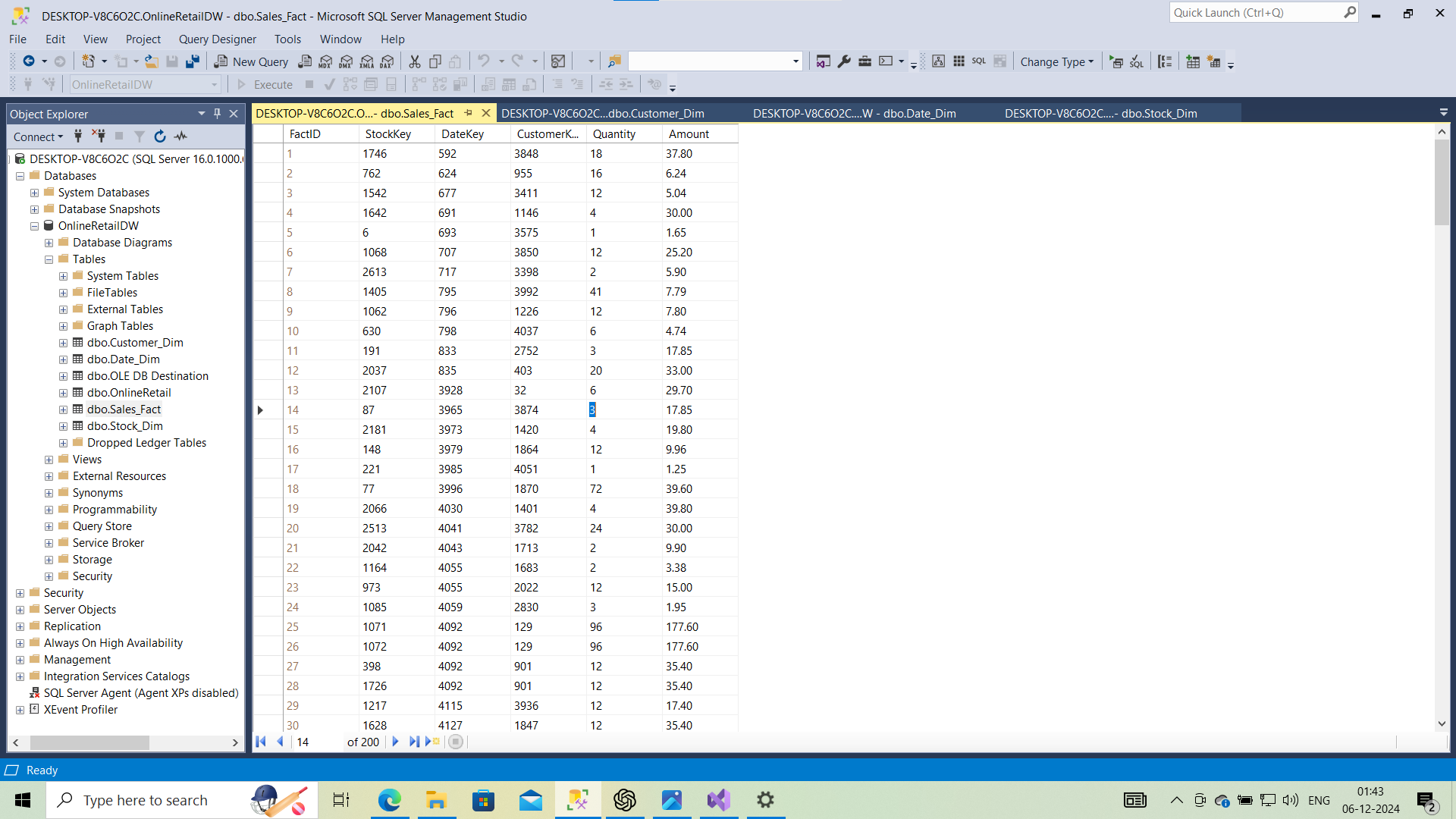
* **Overview**:
  + The control flow included:
    1. **Data Flow Task**: Importing data from CSV file into “OnlineRetal”
    2. **Execute SQL Task**: execute SQL scripts for populating dimension and fact tables.

**2. SSIS Data Flow Tasks**

* **Overview**:
  + The data flow contained transformations such as:
    - **Derived Column Transformation**:
      * Calculated Amount as Quantity \* UnitPrice.
    - **Aggregate Transformation**:
      * Duplicate product attributes were removed for Stock\_Dim.

**SQL Query Snapshots**

****

****

The ETL process successfully transformed the data from raw form in OnlineRetail.csv table and arranged it into star schema suitable for analytical queries. Using SQL and efficient SSIS work flows made it a highly effective process of ensuring high data accuracy, consistency and preparedness for business intelligence analytics.

**5. Reports and Visualizations**

This section presents the findings of analysis that was done by using the transformed data in the form of the SSRS Reports and Tableau Dashboards. The reports offer key findings on the revenue forecasts, product sales, geographical distributions and customer segmentation.

**Report 1: Monthly Sales Trends**

**Description**:

* A **report** demonstrating revenue trends for a time period by month.
* **Purpose**: Determine periods of high and low revenue and also get seasonal trends and fluctuations in sales growth.

**Key Insights**: In **December** due to holiday season sales and revenue increased.

**SSRS Features**:

* **Data Source**: FactSales fact table connected to Date\_Dim.

**Report 2: Sales by Country**

**Description**:

* A report demonstrating total sales revenue based on country.
* **Purpose**: Determine regions of major sales and make marketing and sales strategies for undesirable markets.

**Key Insights**:

* Majority of revenue comes from UK followed by other European countries.

**SSRS Features**:

* **Data Source**: FactSales fact table connected to Customer\_Dim.

**Report 3: Top-Selling Products**

**Description**: A **report** showing all the products by its revenue.

* **Purpose**: Determine the top performing products and optimize inventory.
* **Key Insights**: the highest selling product gives greater sales in comparision to others.

**SSRS Features**:

* **Data Source**: FactSales fact table connected to Stock\_Dim.

**Report 4: High Revenue Days**

**Description**: A **report** showcasing days with the highest revenue.

* **Purpose**: Get to know specific dates of major sales.

**Key Insights**: Major revenue days are in the month of **Black Friday** and **Christmas Eve** .

**SSRS Features**:

* **Data Source**: FactSales fact table connected to Date\_Dim.

The data visualizations were created using **Tableau**, focusing on four key analyses derived from the star schema. These visualizations were integrated into a single dashboard to provide a holistic view of the sales trends, product performance, and revenue contributions by country.

**Data Visualization in Tableau**

**1. Monthly Sales Trends**

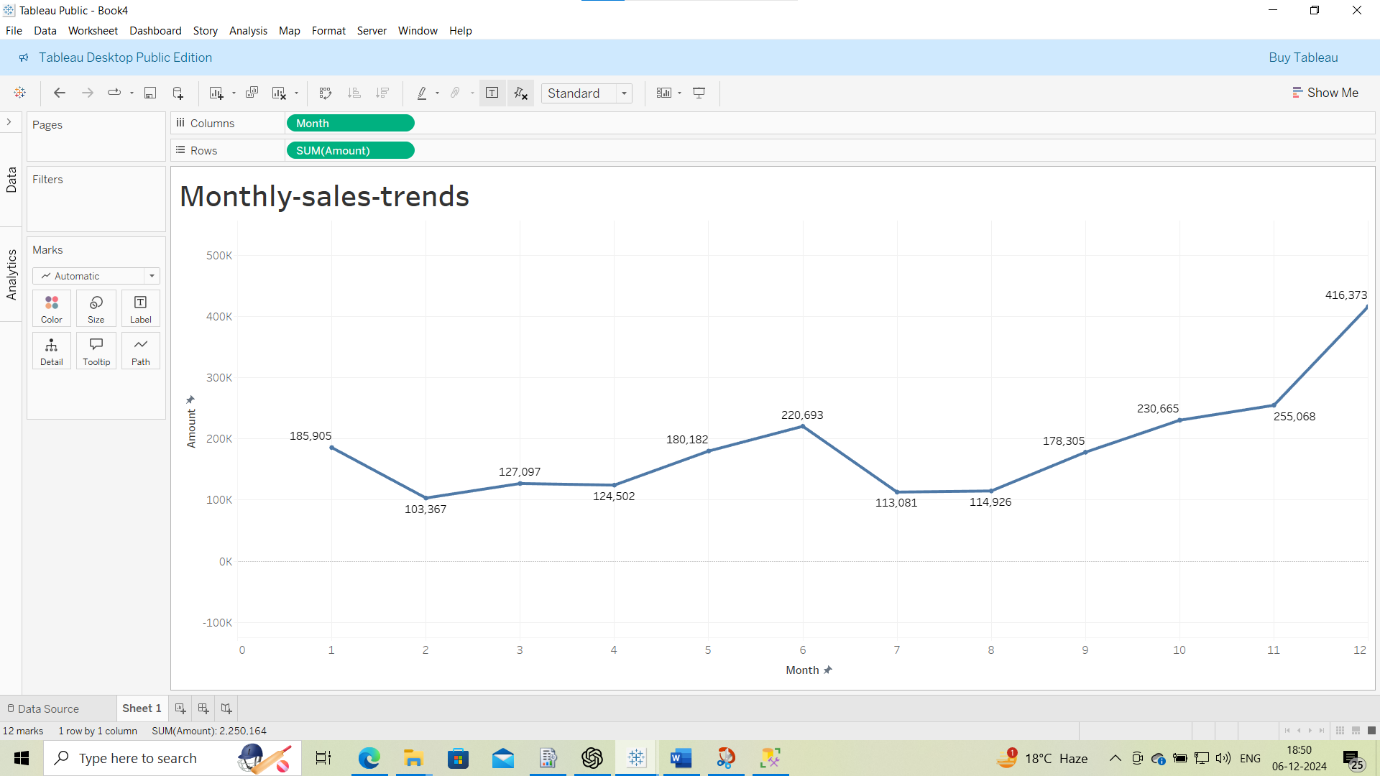
**Visualization**:

* **Type**: Line graph.
* **Description**: Gives a graph between total revenue for each month over time showcasing sales trends.

**Insights**:

* The graph shows that the sales increase in the late seasons.
* **December** is the month of highest sales maybe due to holidays.

**Screenshot**:



**2. High Revenue Days**

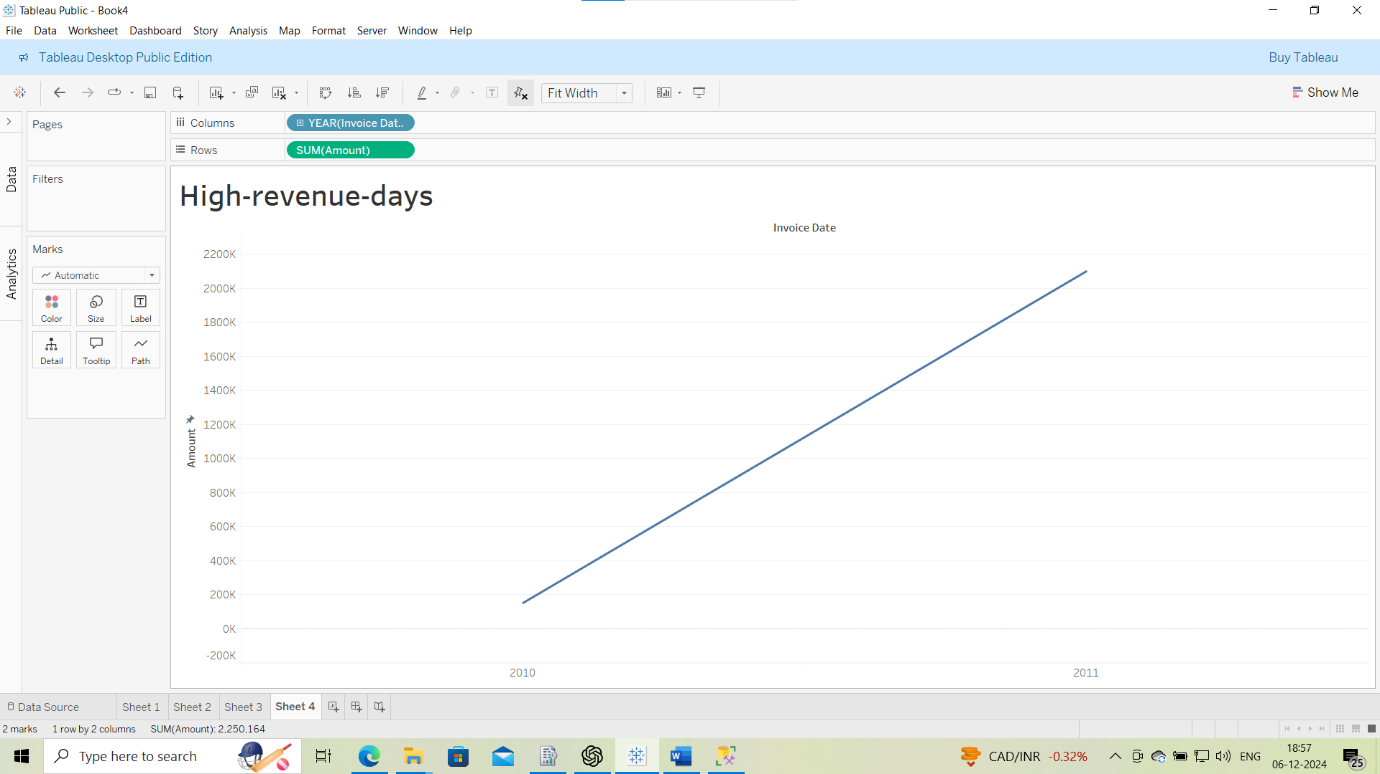
**Visualization**:

* **Type**: Linear graph.
* **Description**: It showcase the trend of increase in revenue over the days in the year.

**Insights**:

* It showcase the steady increase in customer base and push for sales.

**Screenshot**:



**3. Top-Selling Products**

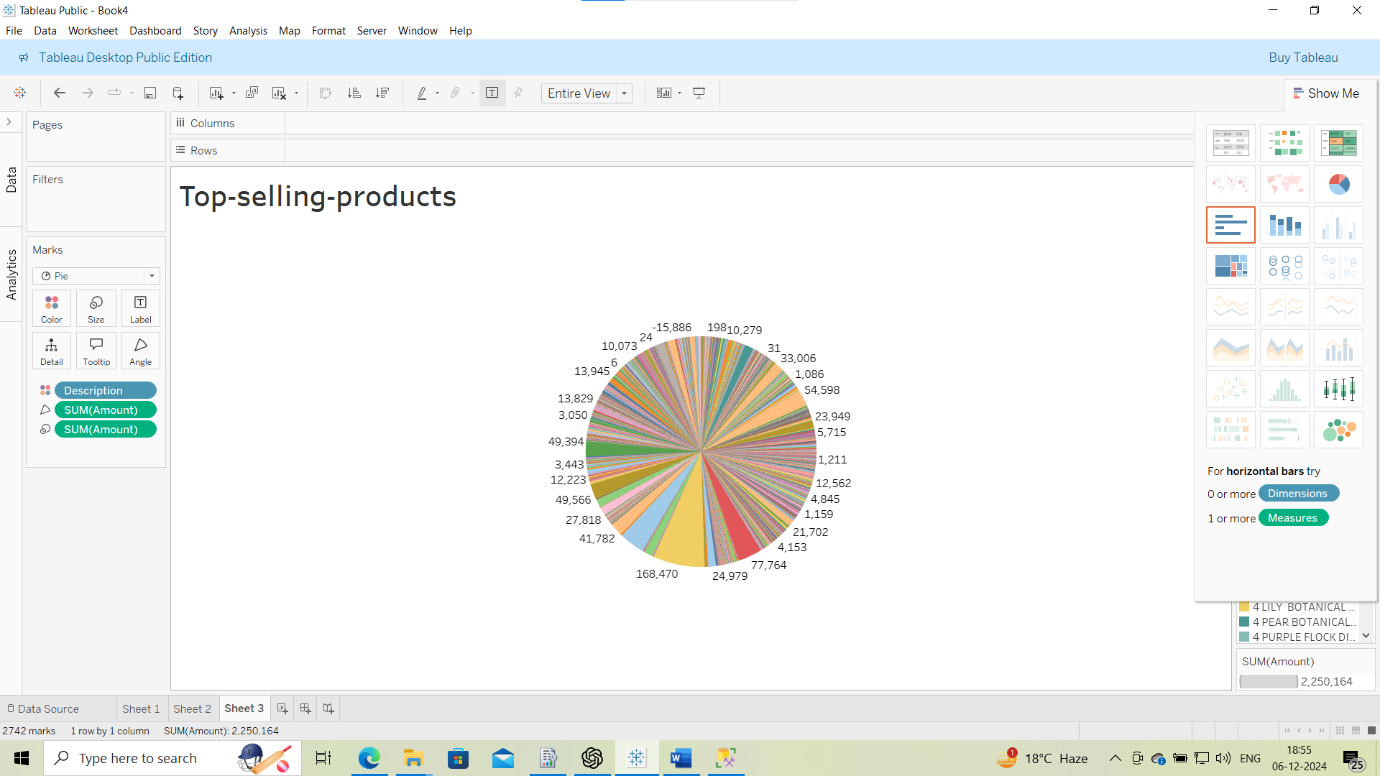
**Visualization**:

* **Type**: Pie chart.
* **Description**: It showcase the total share of each product in the total revenue.

**Insights**:

* It is evident that most of revenue come from sales of highest selling products.
* Items such as gifts and decorations made the major part of the revenue sales.

**Screenshot**:



**4. Sales by Country**

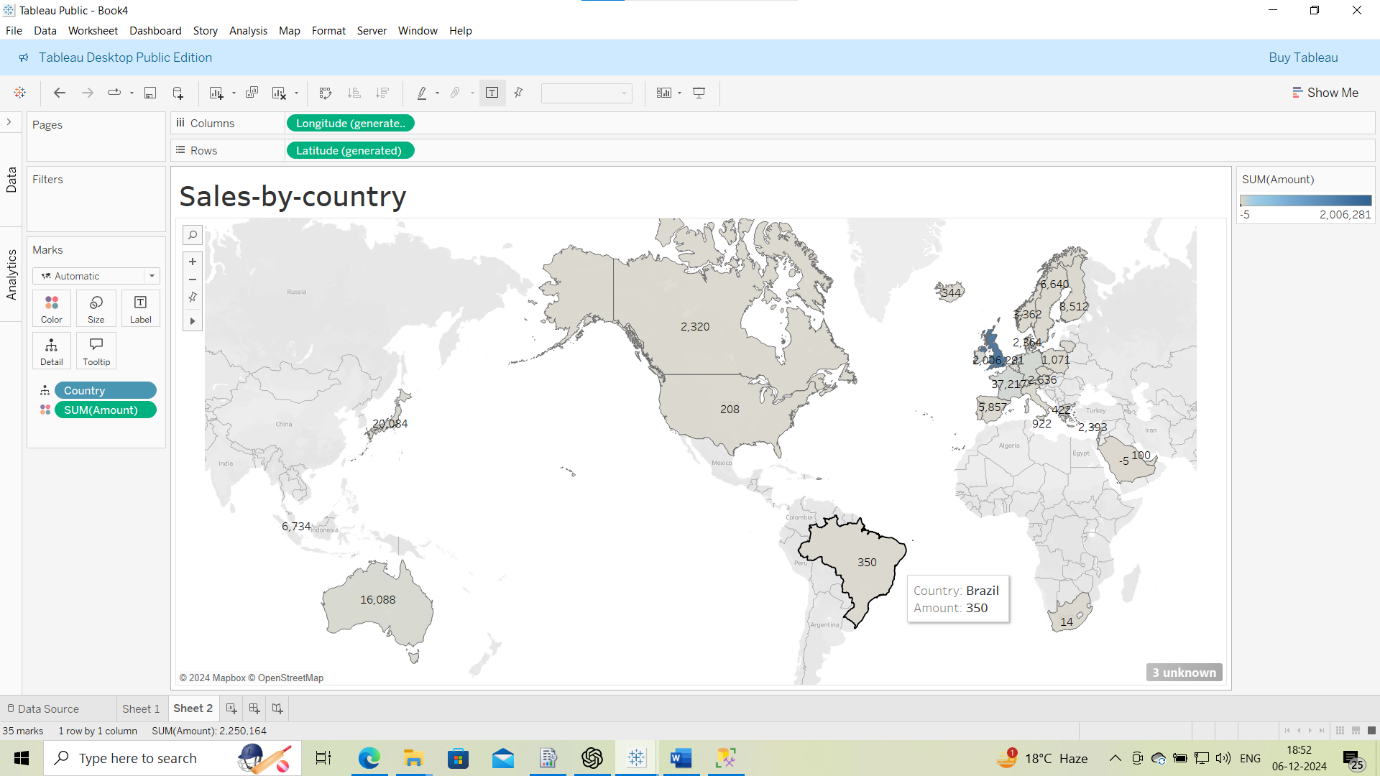
**Visualization**:

* **Type**: Geographical map.
* **Description**: it shows the geographical distribution of revenue based on countries.

**Insights**:

* Around 80% of the total revenue comes from UK alone.
* The other European countries show potential for sales in future.
* The countries giving negligible revenue means they have untapped market and immense potential for future sales.

**Screenshot**:



**5. Combined Dashboard**

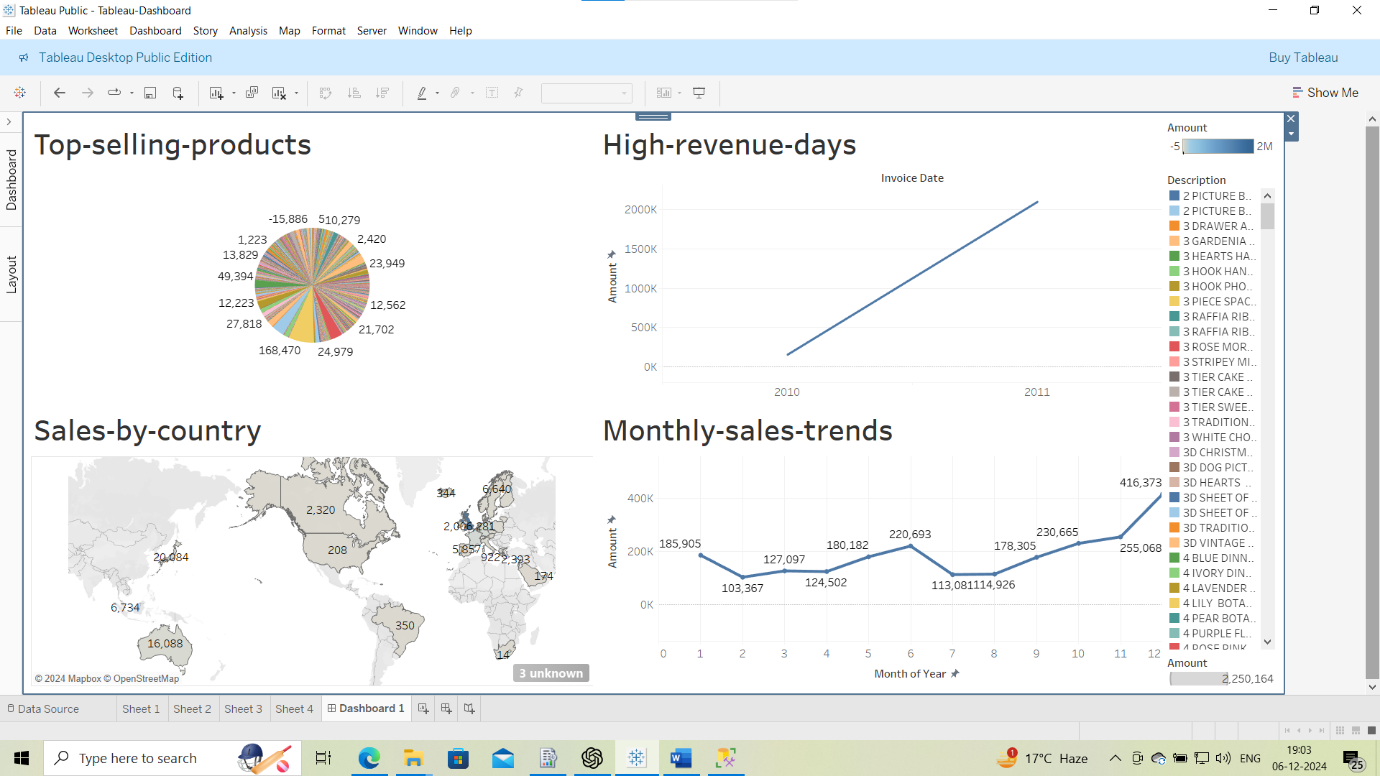
**Visualization**:

* **Description**: A **main dashboard** contain all four workbooks for overall view.

**Insights**:

* The dashboard allows stakeholders to:
  + Determine trends (e.g., monthly growth and peak sales periods).
  + Know the high-value products and customers.
  + Gain insights on revenue collection from different geographical locations.

**Screenshot**:



**6. Relational vs. Graph Databases**

This section aims to compare how relational databases are implemented and analysed with graph databases used with Neo4J, and how it relates to the AdventureWorksDW2022 dataset. This is followed by a discourse on the performance and usability of the schemas compared to each other and use case queries.

**Relational Database Implementation**

**Schema Design**

The relational database was implemented using a **star schema** in SQL Server, consisting of the following:

1. **Fact Table**:
   * **FactInternetSales**: It contain transactional data like SalesAmount, OrderQuantity, and foreign keys to dimension tables.
2. **Dimension Tables**:
   * **DimCustomer**: Contains details of customers like CustomerID, FirstName, and Country.
   * **DimProduct**: Contains product attributes like ProductKey, ProductName, and Category.
   * **DimDate**: Contains date attributes like Year, Month, and Day.

**Graph Database Implementation**

**Schema Design**

The graph database in Neo4J was modeled with:

* **Nodes**:
  + Customer: Represents individual customers (CustomerID, FirstName, Country).
  + Product: Represents products (ProductKey, ProductName, Category).
  + Date: Represents transaction dates (Year, Month, Day).
* **Relationships**:
  + Customer PURCHASED Product: Contains transaction details like SalesAmount, OrderQuantity, and OrderDate.

**Graph Schema Representation:**

(Customer)-[:PURCHASED {SalesAmount, OrderQuantity}]->(Product)

(Product)-[:SOLD\_ON]->(Date)

**Key Queries**

The seven queries are performed using both **SQL** and **Cypher** for demonstrating their implementation in relational and graph databases. They are as follows:

**1. Average Sales Amount per Transaction**

**SQL**:

SELECT AVG(SalesAmount) AS AverageSales

FROM FactInternetSales;

**Cypher**:

MATCH (:Customer)-[r:PURCHASED]->(:Product)

RETURN AVG(TOFLOAT(r.SalesAmount)) AS AverageSales;

**2. Customers and Their Total Purchases**

**SQL**:

SELECT CustomerKey, SUM(OrderQuantity) AS TotalPurchases

FROM FactInternetSales

GROUP BY CustomerKey

ORDER BY TotalPurchases DESC;

**Cypher**:

MATCH (c:Customer)-[r:PURCHASED]->(:Product)

RETURN c.CustomerID, SUM(TOINTEGER(r.OrderQuantity)) AS TotalPurchases

ORDER BY TotalPurchases DESC;

**3. Most Popular Product**

**SQL**:

SELECT ProductKey, COUNT(\*) AS PurchaseCount

FROM FactInternetSales

GROUP BY ProductKey

ORDER BY PurchaseCount DESC;

**Cypher**:

MATCH (:Customer)-[r:PURCHASED]->(p:Product)

RETURN p.ProductName, COUNT(r) AS PurchaseCount

ORDER BY PurchaseCount DESC;

**4. Top Customers by Sales Amount**

**SQL**:

SELECT TOP 5 CustomerKey, SUM(SalesAmount) AS TotalSales

FROM FactInternetSales

GROUP BY CustomerKey

ORDER BY TotalSales DESC;

**Cypher**:

MATCH (c:Customer)-[r:PURCHASED]->(:Product)

RETURN c.CustomerID, SUM(TOFLOAT(r.SalesAmount)) AS TotalSales

ORDER BY TotalSales DESC

LIMIT 5;

**5. Total Quantity Sold by Product**

**SQL**:

SELECT ProductKey, SUM(OrderQuantity) AS TotalQuantity

FROM FactInternetSales

GROUP BY ProductKey

ORDER BY TotalQuantity DESC;

**Cypher**:

MATCH (:Customer)-[r:PURCHASED]->(p:Product)

RETURN p.ProductName, SUM(TOINTEGER(r.OrderQuantity)) AS TotalQuantity

ORDER BY TotalQuantity DESC;

**6. Total Sales by Customer**

**SQL**:

SELECT CustomerKey, SUM(SalesAmount) AS TotalSales

FROM FactInternetSales

GROUP BY CustomerKey

ORDER BY TotalSales DESC;

**Cypher**:

MATCH (c:Customer)-[r:PURCHASED]->(:Product)

RETURN c.CustomerID, SUM(TOFLOAT(r.SalesAmount)) AS TotalSales

ORDER BY TotalSales DESC;

**7. Total Sales by Product**

**SQL**:

SELECT ProductKey, SUM(SalesAmount) AS TotalSales

FROM FactInternetSales

GROUP BY ProductKey

ORDER BY TotalSales DESC;

**Cypher**:

MATCH (:Customer)-[r:PURCHASED]->(p:Product)

RETURN p.ProductName, SUM(TOFLOAT(r.SalesAmount)) AS TotalSales

ORDER BY TotalSales DESC;

**Analysis**

**1. Performance**

1. **Relational Database**:
   * While aggregations on large result sets for example sum of the sales, average revenues and etc, SQL Server is very fast.
   * Joins between tables are expensive especially when normalized high degree schemas are in use but this is well handled in the star schema.
2. **Graph Database**:
   * Neo4J is more appropriate for operations that involve relationships such as customer-product relationships or products which are likely bought together.

**2. Ease of Querying**

1. **Relational Database**:
   * Queries frequently needs **complex joins** because of multi-level relationships (e.g., linking customers, products, and sales).
   * SQL is widely used by the most analysts due to its wide use in this field.
2. **Graph Database**:
   * Cypher makes it easy to navigate through relationships, as relationships between nodes are already implied in cypher graph model
   * For instance, to search for a product’s purchasers means a direct access through the PURCHASED relationship, no need for the join.

**3. Key Insights**

1. **Top Customers by Revenue**:
   * Both of the database pointed towards similar set of high value customers, however Neo4J also presented relation between customer and products.**Popular Products**:
   * SQL and Neo4J reported the top-selling products.
   * Neo4J provided an extensive means of probing for associated purchase patterns.
2. **Sales Trends**:
   * SQL performed more efficiently tasks such as the aggregation over time intervals like monthly sales data.

**Benefits:**

1. **When to Use Relational Databases**:
   * Suitable for highly technical and formal data type and queries involving large data sets common in financial reporting and or trend analysis.
2. **When to Use Graph Databases**:
   * More appropriate if the application involves managing relationships between things/individuals including, recommendation systems, social graphs, and fraud detection models.

Through integrating two databases where one consists of the organizations strengths and weaknesses and the other contains data on the organizations relationships with others, businesses can decide rationally.

**7. References/Bibliography**

* UCI Machine Learning Repository. (n.d.). **Online Retail Dataset**. Retrieved from:  
  <https://archive.ics.uci.edu/ml/datasets/Online+Retail>
* Microsoft SQL Server Samples. (n.d.). **AdventureWorksDW2022 Sample Database**. Retrieved from:  
  <https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks>
* Microsoft Documentation. (n.d.). **SQL Server BULK INSERT**. Retrieved from:  
  <https://learn.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql>
* Microsoft Documentation. (n.d.). **SQL Server Integration Services (SSIS)**. Retrieved from:  
  <https://learn.microsoft.com/en-us/sql/integration-services/sql-server-integration-services>
* Tableau Help. (n.d.). **Building Dashboards and Storyboards**. Retrieved from:  
  <https://help.tableau.com/current/pro/desktop/en-us/dashboards_storypoints.htm>
* Neo4J Documentation. (n.d.). **Cypher Query Language**. Retrieved from:  
  <https://neo4j.com/docs/cypher-manual/>
* Neo4J Tutorials. (n.d.). **Introduction to Graph Modeling**. Retrieved from:  
  <https://neo4j.com/developer/guide-data-modeling/>
* DataCamp. (n.d.). **Introduction to SQL**. Retrieved from:  
  <https://www.datacamp.com/courses/>
* W3Schools. (n.d.). **SQL Joins and Aggregations**. Retrieved from:  
  <https://www.w3schools.com/sql/>