

DSA 3050A — Business Intelligence & Data Visualization

Mid-Semester Practical Examination (Power BI)

Student Name: Selmah Tzindori

Course Code: DSA 3050A

Assessment Type: Individual Practical Examination

TABLE OF CONTENTS

1. Introduction

1.1 Project Overview

1.2 Business Problem Statement

1.3 Stakeholders

1.4 Tools and Technologies Used

1.5 Project Workflow

2. Section 1 — Process Evidence

2.1 Dataset Sourcing Evidence

2.2 Power Query Transformations

Step 1 : Import and Initial Review

Step 2 : Remove Duplicates

Step 3 : Handle Missing / Null Values

Step 4 : Correct Data Types

Step 5 : Standardize Text / Categorical Values

Step 6 : Generate Dates and Extract Year, Month, Quarter

Step 7 : Split / Merge Columns (Country Codes)

Step 8 : Derived Columns

Step 9 : Remove Unnecessary Columns & Rename Queries

Step 11 : Renaming Queries & Splitting Tables

2.3 Applied Steps Evidence

2.4 Data Model Relationships

2.5 Dashboard Pages

2.5.1 Overall Performance

2.5.2 Regional Analysis

2.5.3 Category Analysis

2.5.4 Sales Band Distribution

2.5.5 Shipping Mode Insights

2.5.6 Decision-Making Insights

2.5.7 Dashboard Visuals

2.6 Publishing Confirmation

3. Section 2 — Dashboard Link (Public Link)

1. Introduction

1.1 Project Overview

This project presents an end-to-end Business Intelligence (BI) solution developed using Microsoft Power BI as part of the DSA 3050A mid-semester practical examination. The objective of the project is to demonstrate the full BI workflow, including dataset sourcing, data preparation using Power Query, analytical data modelling, dashboard development, and publishing of insights through the Power BI Service.

The project is designed to simulate a real-world organizational analytics engagement, where data-driven insights are used to support managerial and operational decision-making.

1.2 Business Problem Statement

Many organizations collect large volumes of transactional data but lack effective tools to transform this data into actionable insights. Without proper data preparation, modelling, and visualization, decision-makers are unable to identify performance trends, profitability drivers, and operational inefficiencies.

This project addresses the need for an interactive business intelligence dashboard that enables stakeholders to monitor sales performance, analyze profitability, evaluate customer and product performance, and identify trends across regions and time periods.

1.3 Stakeholders

The primary stakeholders for this Business Intelligence solution include:

- Executive Management
- Regional Sales Managers

- Operations and Strategy Teams

These stakeholders require timely, accurate, and interactive reports to support strategic planning and operational optimization.

1.4 Tools and Technologies Used

The project was developed using the following tools and technologies:

- Microsoft Power BI Desktop for data preparation, modelling, and dashboard development
- Power BI Service for report publishing and sharing
- Google Docs for documentation and evidence compilation
- GitHub for project version control and artifact organization

1.5 Project Workflow

The project follows a structured Business Intelligence workflow consisting of:

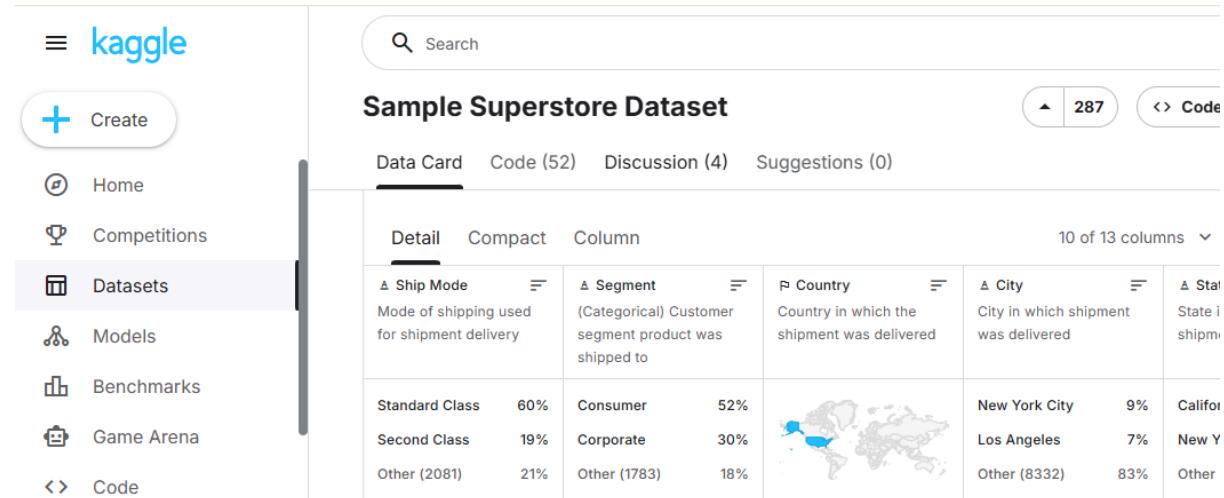
1. Dataset selection and sourcing
2. Data preparation using Power Query
3. Data modelling using a star schema
4. Dashboard design and development
5. Publishing and deployment of insights

2. Section 1 — Process Evidence

2.1 Dataset Sourcing Evidence

I selected the **Sample Superstore Dataset** from Kaggle because it is a real-world dataset that meets the exam requirements. The dataset contains more than 7,000 rows and includes important analytical attributes such as dates (Order Date, Ship Date), categories (Segment, Category, Sub-Category), numerical values (Sales, Quantity, Profit), and locations (Country, Region, State, City, Postal Code).

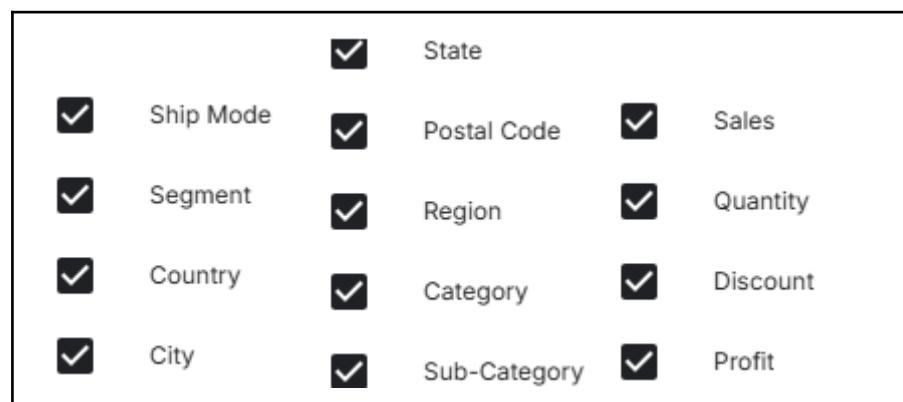
Although the dataset is provided as a single CSV file, I can logically separate it into multiple tables for modelling: a fact table for orders and dimension tables for customers, products, and shipping. This dataset provides the foundation for analytical storytelling, including sales performance across segments, regions, product categories, and time periods.



The screenshot shows the Kaggle interface. On the left, there's a sidebar with options like 'Create', 'Home', 'Competitions', 'Datasets' (which is selected), 'Models', 'Benchmarks', 'Game Arena', and 'Code'. The main area is titled 'Sample Superstore Dataset'. It has tabs for 'Data Card' (selected), 'Code (52)', 'Discussion (4)', and 'Suggestions (0)'. Below the tabs, there are three buttons: 'Detail', 'Compact', and 'Column'. A map of the world is visible. The data card displays the following information:

| Ship Mode | Segment | Country | City | State |
|---|---|---|--------------------------------------|---------------------------------------|
| Mode of shipping used for shipment delivery | (Categorical) Customer segment product was shipped to | Country in which the shipment was delivered | City in which shipment was delivered | State in which shipment was delivered |
| Standard Class | Consumer | 52% | New York City | 9% |
| Second Class | Corporate | 30% | Los Angeles | 7% |
| Other (2081) | Other (1783) | 18% | Other (8332) | 83% |

Figure 1: Raw Online Retail dataset sourced from Kaggle open data platform



A table showing the checked columns for the Sample Superstore Dataset. The columns are grouped into pairs:

| | | | | | | | | | |
|-------------------------------------|-------------|-------------------------------------|--------------|-------------------------------------|--------------|-------------------------------------|--------|-------------------------------------|----------|
| <input checked="" type="checkbox"/> | Ship Mode | <input checked="" type="checkbox"/> | Segment | <input checked="" type="checkbox"/> | Country | <input checked="" type="checkbox"/> | City | <input checked="" type="checkbox"/> | State |
| <input checked="" type="checkbox"/> | Postal Code | <input checked="" type="checkbox"/> | Region | <input checked="" type="checkbox"/> | Sub-Category | <input checked="" type="checkbox"/> | Sales | <input checked="" type="checkbox"/> | Quantity |
| <input checked="" type="checkbox"/> | Category | <input checked="" type="checkbox"/> | Sub-Category | <input checked="" type="checkbox"/> | Discount | <input checked="" type="checkbox"/> | Profit | <input checked="" type="checkbox"/> | Profit |

Figure 2: Overview of columns in the Sample Superstore dataset.

2.2 Power Query Transformations

Step 1 : Import and Initial Review

I imported the Superstore_Data.csv dataset into Power BI and opened Power Query Editor. I first reviewed the column profile and distributions to understand the data types, null values, and unique entries for each column. This step helps identify potential issues and plan transformations.

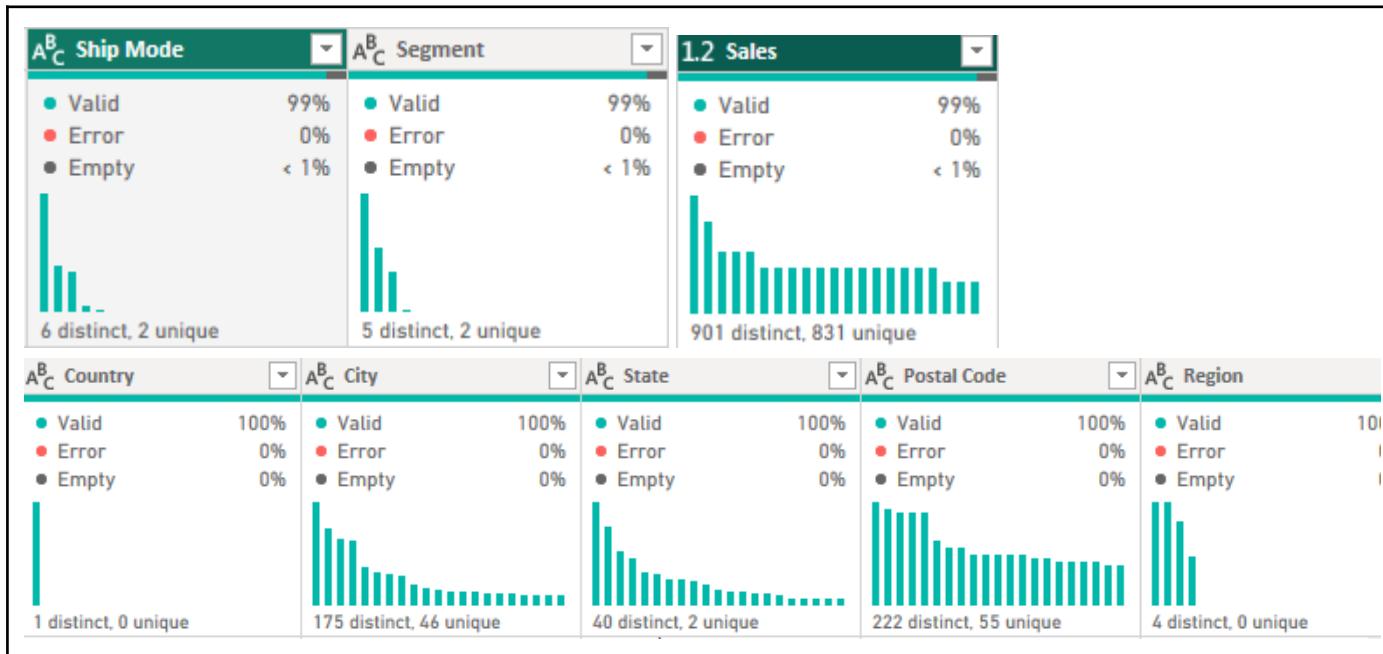


Figure 3: Sample Column statistics and distribution in Power Query to assess data quality and prepare for transformations.

Step 2 : Remove Duplicates

I first checked the dataset for duplicate rows. Using Power Query, I applied the **Remove Duplicates** function on all columns. Performing this step ensures that each transaction is unique and maintains the accuracy of aggregated metrics like total Sales and number of orders.

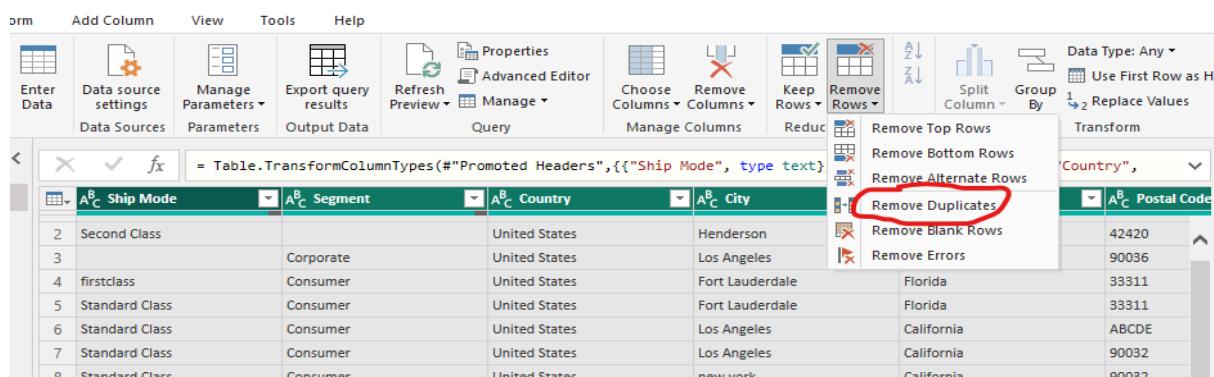


Figure 5: Removing duplicate rows in Power Query to ensure data uniqueness.

I counted the rows before and after removing duplicates. The dataset had **9,994 rows initially**. After applying the Remove Duplicates function the dataset has **9977 rows** now indicating that 17 duplicate transactions were removed



Figure 5a: Total number of rows in the dataset before removing duplicates (9,994 rows)

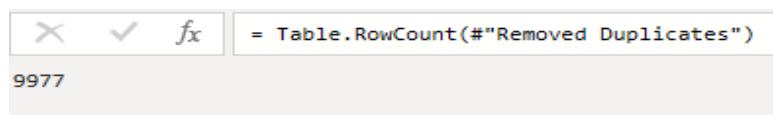


Figure 5b: Total number of rows in the dataset after removing duplicates (9,977 rows) showing that duplicates were removed.

Step 3 : Handle Missing / Null Values

I analyzed the dataset for missing or null values using the Column **Profile** in Power Query. I observed that some entries in **Ship Mode** and **Segment** were empty strings, and a few values in **Sales** were null.

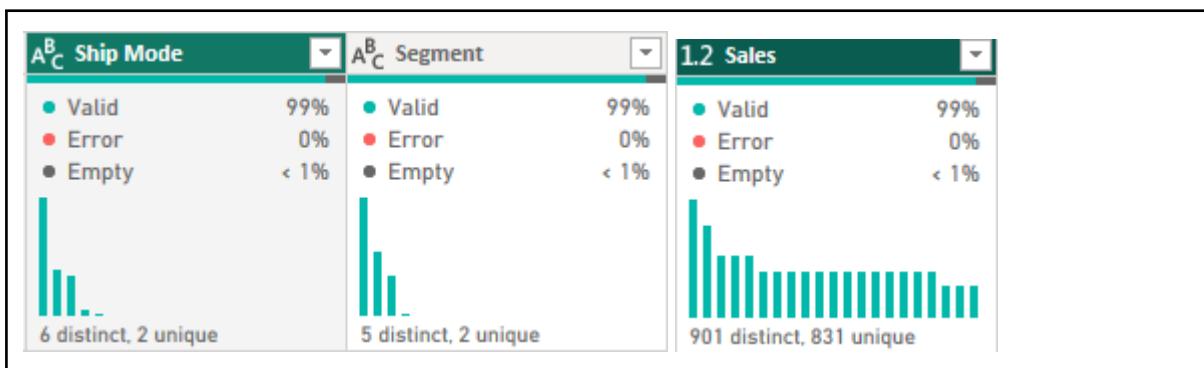


Figure 6a: Initial Column profile in Power Query highlighting empty strings in Ship Mode and Segment, and nulls in Sales.

I applied transformations to correct these: replacing nulls in **Sales** with 0, empty strings in **Segment** with “Unknown”, and empty strings in **Ship Mode** with the **Standard Class** as it is the mode

Replace Values

Replace one value with another in the selected columns.

Value To Find

Replace With

Standard Class

Replace Values

Replace one value with another in the selected columns.

Value To Find

null

Replace With

0

Replace Values

Replace one value with another in the selected columns.

Value To Find

Replace With

Unknown

Figure 6b: Power Query Applied Steps showing replacement of nulls and empty strings.

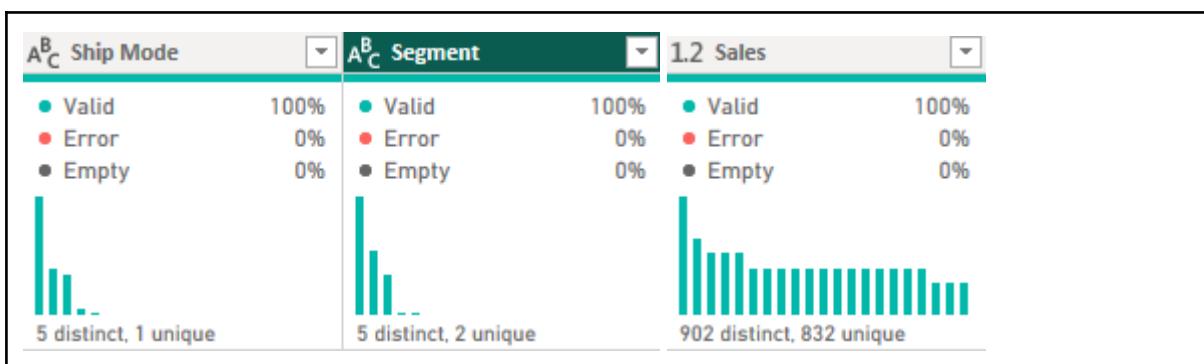


Figure 6c: Updated column profile showing that all missing and empty values were corrected.

Step 4 : Correct Data Types

I reviewed all columns in the dataset to ensure that the data types were appropriate for analysis. Most columns were already correctly typed; however, I applied corrections for a few inconsistencies:

1. The **Sales** column contained a few entries stored as text → I changed it to Decimal/Number to enable calculations.
2. The **Postal Code** column had numeric-like values stored inconsistently as numbers and text → I enforced Text data type for consistency.

I also confirmed that other numerical columns such as **Quantity** and **Profit** were of type **Decimal/Number**, and categorical columns such as **Segment**, **Category**, and **Ship Mode** were **Text**.

The screenshot shows the Power BI desktop application with a data view containing three columns: Postal Code, Region, and Category. The 'Postal Code' column is currently selected, as indicated by the highlighted row. A context menu is open for this column, with the 'Change Type' option highlighted. A secondary dropdown menu is open under 'Change Type', listing various data types: Decimal Number, Fixed decimal number, Whole Number, Percentage, Date/Time, Date, Time, Date/Time/Timezone, Duration, and Text. The 'Text' option is checked, indicating it is the selected data type for the 'Postal Code' column.

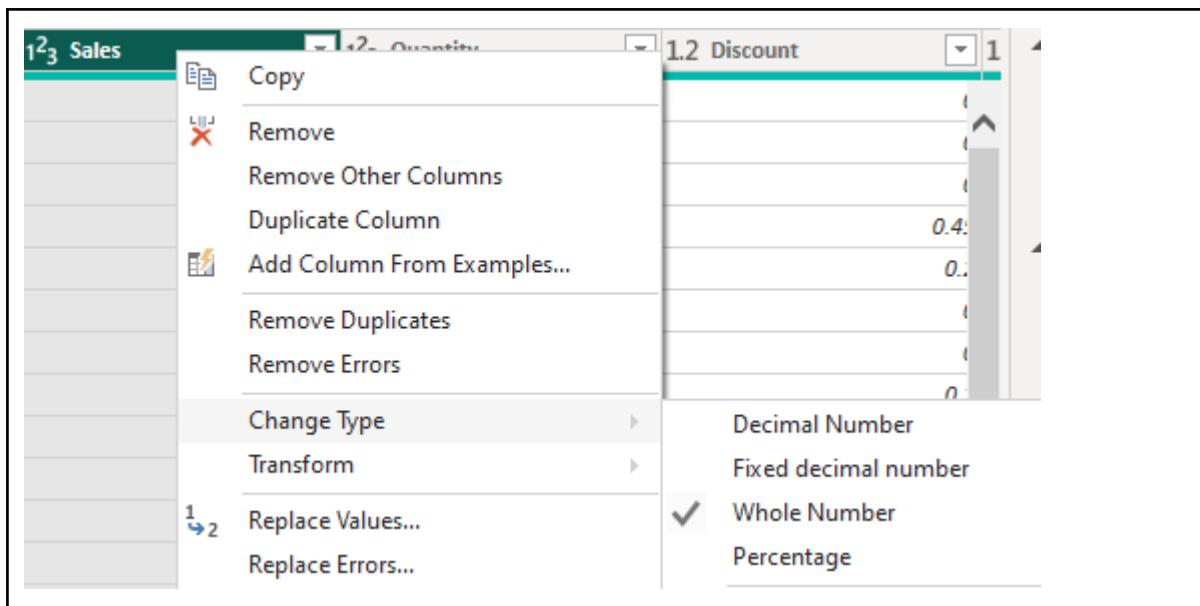


Figure 7: Power Query Applied Steps showing the step taken to correct and confirm data types for Sales, Postal Code

Step 5 : Standardize Text / Categorical Values

I noticed inconsistencies in the **Ship Mode** column, such as `firstclass` instead of `First Class`. To correct this, I applied a **Replace Values** transformation in Power Query and capitalized all entries properly. This ensures accurate grouping and reporting for shipping modes

| Value To Find | Replace With |
|-------------------------|--------------------------|
| <code>firstclass</code> | <code>First Class</code> |

The screenshot shows the 'Replace Values' dialog. On the left, there's a preview of the original data with entries like 'Second Class', 'Second Class', 'Standard Class', 'First Class', etc. On the right, the transformed data is shown with all entries capitalized as 'First Class', 'First Class', 'Standard Class', 'Standard Class', etc.

Figure 8a: Power Query Applied Step showing the replacement and capitalization of inconsistent entries in the Ship Mode column and after change.

The screenshot shows the Power Query Editor interface. At the top, there are several transformation steps applied to a table:

- Table**: The first step.
- Any Column**: The second step, which includes a formula: `= Table.TransformColumnTypes(#"Added Custom1", {"Sales", Int64.Type})`.
- Lowercase**: The third step.
- Number Column**: The fourth step.

Below these steps is a table with the following data:

| A ^B C City | A ^B C State | A ^B C Postal Code | A ^B C Reg | A ^B C Category | A ^B C Sub-Category |
|-----------------------|------------------------|------------------------------|----------------------|---------------------------|-------------------------------|
| Henderson | Kentucky | 42420 | South | Bookcases | |
| Henderson | Kentucky | 42420 | South | Chairs | |
| Los Angeles | California | 90036 | West | Labels | |
| Fort Lauderdale | Florida | 33311 | South | Furniture | Tables |
| Fort Lauderdale | Florida | 33311 | South | Office Supplies | Storage |
| Los Angeles | California | ABCDE | West | Furniture | Furnishings |

Below this table is a section titled **Resulting change**, which shows the transformed data:

| A ^B C Ship Mode | A ^B C Segment | A ^B C Country | A ^B C City | A ^B C State |
|----------------------------|--------------------------|--------------------------|-----------------------|------------------------|
| Second Class | Consumer | United States | Henderson | Kentucky |
| Second Class | Unknown | United States | Henderson | Kentucky |
| Standard Class | Corporate | United States | Los Angeles | California |
| First Class | Consumer | United States | Fort Lauderdale | Florida |
| Standard Class | Consumer | United States | Fort Lauderdale | Florida |

Figure 8b: Applied Steps showing trimming and capitalization transformations for City, State, Segment, Category, and Sub-Category

Step 6 : Generate Dates and Extract Year, Month, Quarter

I generated an **Order Date** column in Power Query to meet the requirement for time-based analysis. The dates are randomly assigned between 1-Jan-2022 and 31-Dec-2022. I also generated a **Ship Date** column with 1–7 days added to **Order Date** to simulate realistic shipping delays.

The screenshot shows the Power Query Editor interface for generating an **Order date** column. The formula is defined as:

```
= #date(2022,1,1) + #duration(Number.RoundDown(Number.RandomBetween(0, 364)),0,0,0)
```

The resulting data table shows various dates generated by the formula:

| A ^B C Order date |
|-----------------------------|
| 7/31/2022 |
| 5/7/2022 |
| 11/11/2022 |
| 5/23/2022 |
| 7/29/2022 |
| 7/12/2022 |
| 8/29/2022 |
| 5/29/2022 |
| 12/23/2022 |

Figure 9a: Power Query Custom Column formula to generate random Order Date for each transaction and the results.

| Ship Date |
|------------|
| 11/5/2022 |
| 6/7/2022 |
| 12/21/2022 |
| 11/16/2022 |
| 12/10/2022 |
| 10/2/2022 |
| 10/10/2022 |
| 9/30/2022 |
| 7/10/2022 |

Figure 9b: Power Query Custom Column formula adding 1–7 days to Order Date to create Ship Date.

I extracted the Year, Month, and Quarter from the Order Date column in Power Query to enable time-based analysis and trend reporting. These new columns allow aggregation of sales, profit, and other metrics by year, month, or quarter, which supports analytical storytelling in the dashboard

| 1 ² ₃ Year | 1 ² ₃ Month | 1 ² ₃ Quarter |
|----------------------------------|-----------------------------------|-------------------------------------|
| 2022 | 11 | 4 |
| 2022 | 1 | 1 |
| 2022 | 7 | 3 |
| 2022 | 8 | 3 |
| 2022 | 7 | 3 |
| 2022 | 6 | 2 |

Figure 9c: Power Query table showing the newly created Order Year, Order Month, and Order Quarter columns extracted from the Order Date for time-based analysis

Step 7 : Split / Merge Columns (Country Codes)

I created a derived column in Power Query to standardize state names into official US state abbreviations using conditional logic. This improves consistency and supports geographic analysis and modelling.

New column name
State Code

| Column Name | Operator | Value | Output |
|---------------|----------|--------------------|-----------------|
| If State | equals | ABC 123 Alabama | Then ABC 123 AL |
| Else If State | equals | ABC 123 Alaska | Then ABC 123 AK |
| Else If State | equals | ABC 123 Arizona | Then ABC 123 AZ |
| Else If State | equals | ABC 123 Arkansas | Then ABC 123 AR |
| Else If State | equals | ABC 123 California | Then ABC 123 CA |

Results

| ABC 123 State Code |
|--------------------|
| KY |
| KY |
| CA |
| FL |
| FL |
| CA |
| CA |
| CA |
| CA |

Figure 10: Creation of a State Code column using conditional logic in Power Query to standardize state values and results.

Step 8 : Derived Columns

I created a Sales Band derived column using conditional logic based on the statistical distribution of the Sales variable. The band thresholds were defined using the mean sales value (≈ 242), observed distribution spread, and presence of high-value outliers to ensure meaningful categorization.

The screenshot shows the 'New column name' field set to 'Sales Band'. Below it, the conditional logic is defined:

| Column Name | Operator | Value | Output |
|---------------|--------------------------|-------|---------------------|
| If Sales | is less than or equal to | 250 | Then ABC 123 Low |
| Else If Sales | is less than or equal to | 1000 | Then ABC 123 Medium |
| Add Clause | | | |
| Else | ABC 123 | High | |

Below the dialog, the 'Results' pane displays the generated Sales Band column:

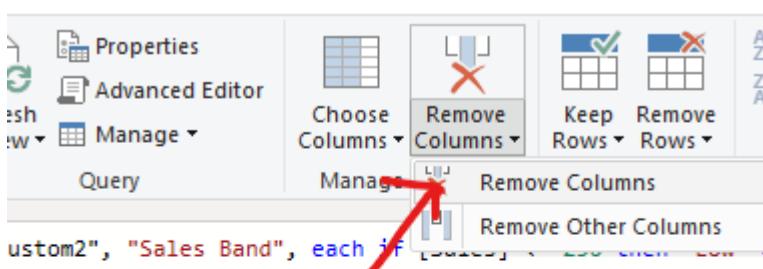
| Sales Band |
|------------|
| Low |
| Medium |
| Low |
| Medium |
| Low |
| Low |
| Medium |
| Medium |
| Low |
| Low |
| High |

Figure 11: shows the creation of a Sales Band column using data-driven thresholds informed by the sales distribution statistics.

Step 9 : Remove Unnecessary Columns & Rename Queries

I removed columns that were not required for analysis, such as **Postal Code**,**the country column** because it is the same throughout, and the original **State** column, to simplify the dataset and reduce clutter in the data model.

The remaining columns contain all necessary information for relationships, derived metrics, and dashboard visuals.



A screenshot of the Microsoft Power Query ribbon. The 'Manage' tab is selected, and the 'Remove Columns' button is highlighted with a red arrow. Below the ribbon, a table is displayed with three columns: 'A_C State', 'A_C Postal Code', and 'A_C Reg'. The data in the table is as follows:

| A_C State | A_C Postal Code | A_C Reg |
|------------|-----------------|---------|
| Kentucky | 42420 | South |
| Kentucky | 42420 | South |
| California | 90036 | West |
| Florida | 33311 | South |
| Florida | 33311 | South |
| California | ABCDE | West |
| California | 90032 | West |

Figure 12: shows the removal of unnecessary columns in Power Query to create a clean dataset ready for modelling.

Step 10 :Create More Derived Columns

I created additional derived columns to enable deeper analytical insights:

1. Profit Margin % — measures profit as a percentage of sales
2. Shipping Delay (Days) — calculates the time between order and shipment
3. Profit / Loss Flag — identifies unprofitable transactions
4. Region_Category — concatenates region and category for cross-dimensional analysis

| | | | |
|---|--|-----------------------|----------------------------|
| New column name Shipping Delay | New column name Profit Margin | | |
| Custom column formula ⓘ <code>= [Ship Date]-[Order date]</code> | Custom column formula ⓘ <code>= [Profit] / [Sales] * 100</code> | | |
| New column name Region | New column name Profit / Loss Flag | | |
| Custom column formula ⓘ <code>= [Region] & "_" & [Category]</code> | Custom column formula ⓘ <code>= if [Profit] < 0 then "Loss" else "Profit"</code> | | |
| Results | | | |
| 123 Shipping Delay | ABC 123 Profit Margin | ABC 123 Region.1 | ABC 123 Profit / Loss Flag |
| 5 | Infinity | South_Furniture | Profit |
| 4 | 29.99754098 | South_Furniture | Profit |
| 6 | 45.80933333 | West_Office Supplies | Profit |
| 6 | -39.98235908 | South_Furniture | Loss |
| 2 | 11.43818182 | South_Office Supplies | Profit |

Figure 13 : Shows the derived columns created in Power Query including Sales Band, Profit Margin %, Shipping Delay, Profit/Loss Flag, and Region_Category.

Step 11 : Renaming Queries & Splitting Tables

The dataset did not include a Customer ID. I created a unique **CustomerId** by removing duplicates in **Segment + City + State Code** and adding an Index column

| A ^B _C Segment | A ^B _C City | A ^B _C State Code | 123 CustomerId |
|-------------------------------------|----------------------------------|--|----------------|
| Consumer | Henderson | KY | 1 |
| Unknown | Henderson | KY | 2 |
| Corporate | Los Angeles | CA | 3 |
| Consumer | Fort Lauderdale | FL | 4 |
| Consumer | Los Angeles | CA | 5 |
| Consumer | New York | CA | 6 |
| Consumer | Concord | NC | 7 |
| Consumer | Seattle | WA | 8 |
| Home Office | Fort Worth | TX | 9 |
| Consumer | Madison | WI | 10 |

Figure 14: shows the generated Customer Key for the Customer dimension table, created from Segment, City, and State Code.

The dataset did not include Product IDs. I created a **Product Id** by removing duplicates on **Category + Sub-Category** and adding an Index column.

| A ^B _C Category | A ^B _C Sub-Category | 1 ² ₃ ProductId |
|--------------------------------------|--|---------------------------------------|
| Furniture | Bookcases | 1 |
| Furniture | Chairs | 2 |
| Office Supplies | Labels | 3 |
| Furniture | Tables | 4 |
| Office Supplies | Storage | 5 |
| Furniture | Furnishings | 6 |
| Office Supplies | Art | 7 |
| Technology | Phones | 8 |

Figure 11: Shows the Product dimension table with unique Product Keys derived from Category and Sub-Category.

I created a Date dimension table from distinct **Order Date** values and extracted Year, Month, and Quarter. A Date Key was added for proper relationships with the Orders fact table.

| A ^B _C Order date | 1 ² ₃ Year | 1 ² ₃ Month | 1 ² ₃ Quarter |
|--|----------------------------------|-----------------------------------|-------------------------------------|
| 4/23/2022 | 2022 | 4 | 2 |
| 7/28/2022 | 2022 | 7 | 3 |
| 10/31/2022 | 2022 | 10 | 4 |
| 5/25/2022 | 2022 | 5 | 2 |
| 9/25/2022 | 2022 | 9 | 3 |
| 11/6/2022 | 2022 | 11 | 4 |
| 2/20/2022 | 2022 | 2 | 1 |

Figure 12: shows the Date dimension table with Year, Month, Quarter, and Date Key derived from Order Date.

I created a Location dimension table to uniquely identify geographic locations for analysis.

- Columns kept: State Code, City, Region
- Added Location Key as primary key

| A _C ^B City | A _C ^B Region | A _C ^B State Code | A _C ^B LocationId |
|----------------------------------|------------------------------------|--|--|
| Henderson | South | KY | 1 |
| Los Angeles | West | CA | 2 |
| Fort Lauderdale | South | FL | 3 |
| New York | West | CA | 4 |
| Concord | South | NC | 5 |
| Seattle | West | WA | 6 |
| Fort Worth | Central | TX | 7 |
| Madison | Central | WI | 8 |
| West Jordan | West | UT | 9 |

Figure 13: shows the Location dimension table with unique Location Keys derived from State Code, City, and Region.

“Orders fact table merged with Customers, Products, and Location dimension tables using text keys. After merging, numeric keys were extracted for relationships in the Star Schema.”

```
= Table.AddColumn(#"Expanded Dim_Location", "Product Merge Key", each Text.Trim([Category]) & "_" & Text.Trim([#"Sub-Category"]))

= Table.AddColumn(#"Expanded Dim_Customer", "Location Merge Key", each Text.Trim([State Code]) & "_" & Text.Trim([City]) & "_" & Text.Trim([Region]))

= Table.NestedJoin(Merge2, {"Order date"}, Dim_Date, {"Order date"}, "Dim_Date", JoinKind.LeftOuter)
```

Figure 14: Merging Fact and Dimension Tables

I then removed the merge keys as they have already served their purpose and we need a clear dataset

| A _C ^B CustomerId | A _C ^B LocationId | A _C ^B ProductId |
|--|--|---------------------------------------|
| 1 | 1 | 1 |
| 1 | 1 | 10 |
| 1 | 1 | 2 |
| 2 | 1 | 2 |
| 3 | 2 | 2 |
| 3 | 2 | 3 |
| 1 | 1 | 13 |

Figure 14: Merging Fact and Dimension Tables Results

2.3 Applied Steps Evidence

| | |
|--------------------------|---------------------|
| Promoted Headers | |
| Changed Type | |
| Removed Duplicates | |
| Replaced Value | |
| Replaced Value1 | |
| Replaced Value2 | |
| Changed Type1 | |
| Replaced Value3 | |
| Capitalized Each Word | Removed Columns |
| Cleaned Text | Added Custom3 |
| Trimmed Text | Changed Type2 |
| Added Custom | Added Custom4 |
| Added Custom1 | Added Custom5 |
| Inserted Year | Added Custom6 |
| Inserted Month | Renamed Columns |
| Inserted Quarter | Removed Columns1 |
| Added Custom2 | Added Custom7 |
| Added Conditional Column | Removed Duplicates1 |

Figure 15: shows all transformations applied to the original Sales Table

| | |
|-----------------------|----|
| Source | |
| Removed Other Columns | ⚙️ |
| Removed Duplicates | ⚙️ |
| Added Index | ⚙️ |
| Renamed Columns | |
| Changed Type | |
| ✗ Added Custom | ⚙️ |
| Source | |
| Removed Other Columns | |
| Removed Duplicates | |
| Added Index | |
| Renamed Columns | |
| Changed Type | |
| ✗ Added Custom | |
| Source | |
| Removed Other Columns | |
| ✗ Removed Duplicates | |
| ✗ Added Custom | |

Figure 16: shows all transformations applied to the Customers, Product, Date, and Location dimension tables respectively in Power Query, including removing duplicates, adding Primary Key, and standardizing text.

2.4 Data Model Relationships

The data model in Power BI links the Orders fact table with the dimension tables to form a **Star Schema**, enabling efficient analysis. The relationships are as follows:

- **Orders.Customer Key → Customers.Customer Key (1:N)**
- **Orders.Product Key → Products.Product Key (1:N)**
- **Orders.Location Key → Location.Location Key (1:N)**
- **Orders.Order Date → Date.Date Key (1:N)**

The fact table contains all transactions, while each dimension table contains unique entities. These relationships allow us to slice, filter, and aggregate data across customers, products, locations, and time.

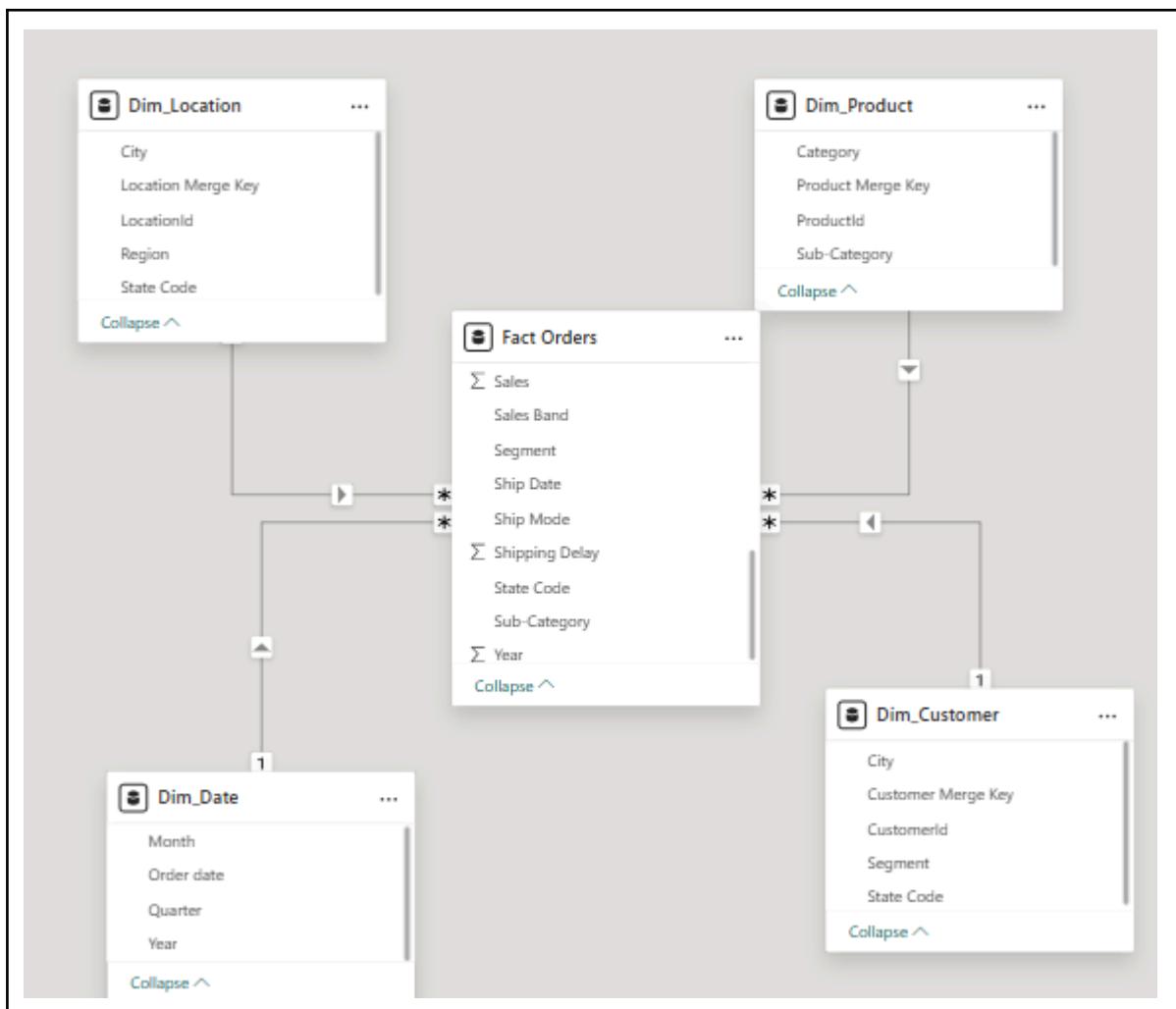


Figure 17: shows the data model relationships in Power BI. The Orders fact table is linked to Customers, Products, Location, and Date dimension tables via their respective keys, forming a Star Schema.

| <input type="checkbox"/> From: table (column) ↑ | Relationship | To: table (column) |
|---|--------------|----------------------------|
| <input type="checkbox"/> Fact Orders (CustomerId) | *—□—1 | Dim_Customer (CustomerId) |
| <input type="checkbox"/> Fact Orders (LocationId) | *—□—1 | Dim_Location (LocationId) |
| <input type="checkbox"/> Fact Orders (Order date) | *—□—1 | Dim_Date (Order date) |
| <input type="checkbox"/> Fact Orders (Sub-Category) | *—□—1 | Dim_Product (Sub-Category) |

*Figure 18: Relationship configuration showing one-to-many (1:) cardinality between a dimension table and the fact table.**

- All relationships were configured as one-to-many, avoiding many-to-many relationships to prevent ambiguity and incorrect aggregations.
- Single-direction filtering was applied from dimension tables to the fact table to maintain correct aggregation behavior and improve model performance.

After creating the required relationships, technical merge key columns were hidden from the Report View to improve usability and prevent confusion for end users.

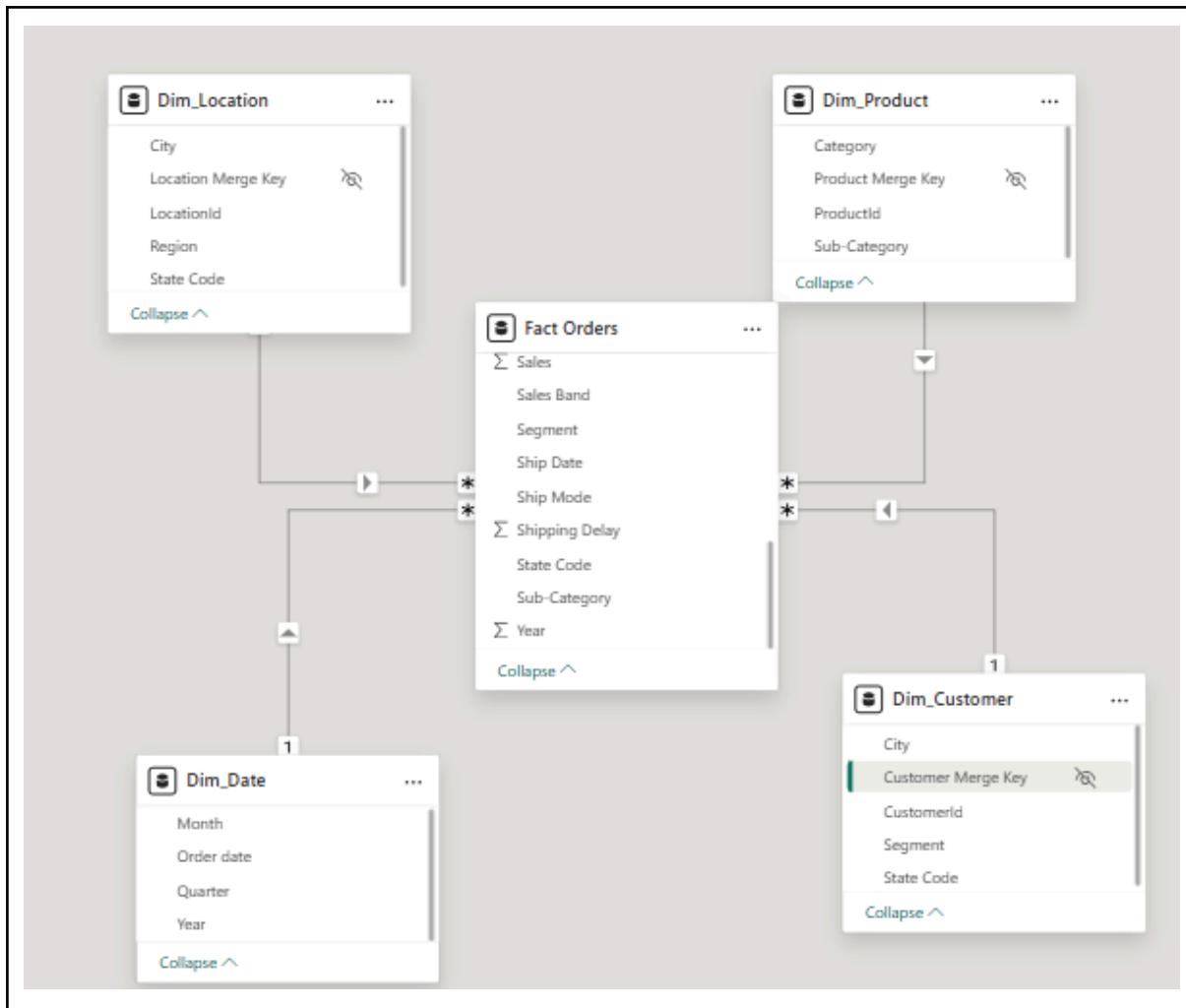


Figure 19: Merge key columns hidden from Report View after relationship creation.

2.5 Dashboard Pages

The Sales Analysis dashboard provides a **comprehensive overview of Superstore's sales and profit performance for 2022**, combining KPIs, trend analysis, comparative charts, and distribution visualizations. It is designed to support **data-driven decision-making**, enabling stakeholders to quickly interpret metrics and identify actionable insights.

2.5.1 Overall Performance

- **Total Sales:** \$2,000,000
- **Total Profit:** \$287,000
- **Quantity Sold:** 38,000 units

The KPI cards summarize the overall health of the Superstore business in 2022, providing a quick snapshot of revenue, profitability, and sales volume.

2.5.2 Regional Analysis

- **West Region:** \$726,000 – highest sales
- **East Region:** \$681,000
- **Central Region:** \$501,000
- **South Region:** \$391,000 – lowest sales

The West region is the top revenue contributor, while the South region underperforms. This highlights potential **regional focus areas** for sales, marketing, and operational improvements.

2.5.3 Category Analysis

- **Technology:** \$838,000 – highest sales
- **Furniture:** \$724,000 – mid-level sales
- **Office Supplies:** \$720,000 – lowest sales

Technology is driving the majority of revenue, indicating strong customer demand. Office Supplies may require **promotional strategies or bundling** to boost sales performance.

2.5.4 Sales Band Distribution

- **Low band:** 7,793 transactions
- **Medium band:** 1,726 transactions
- **High band:** 468 transactions

Most sales fall into the **low band**, indicating smaller transactions dominate. High-value sales are limited, suggesting opportunities for **upselling or premium offerings** to increase revenue.

2.5.5 Shipping Mode Insights

- **Standard Class Ship Mode:** Generates the most revenue at \$1,000,000

Customers clearly prefer standard shipping. Operational focus on this mode can help maintain efficiency and satisfaction while exploring ways to optimize alternative shipping options.

2.5.6 Decision-Making Insights

- Prioritize **high-performing regions and categories** to sustain revenue
- Address **underperforming segments** (South region, Office Supplies) with targeted initiatives
- Explore strategies to increase **high-band sales** through promotions or upselling
- Leverage **Standard Class shipping trends** to optimize delivery operations

2.5.7 Dashboard Visuals

This visual provides a **single-page interactive view** of KPIs, trends, comparative charts, and distribution, allowing stakeholders to filter by **Region, Category, Segment, Ship Mode, and Year**, with cross-filtering across all visuals.

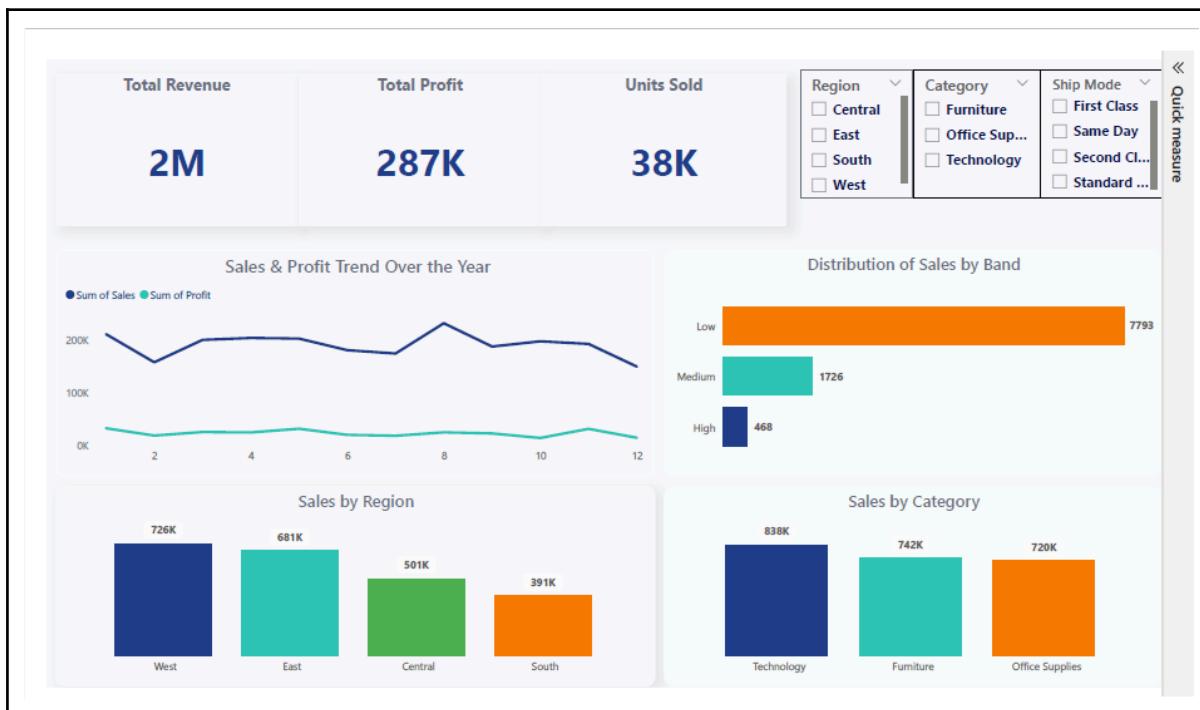


Figure 20 : Dashboard successfully created

2.6 Publishing Confirmation

After publishing the dashboard ,I pinned the sales visuals just to get an overview of the sales attribute

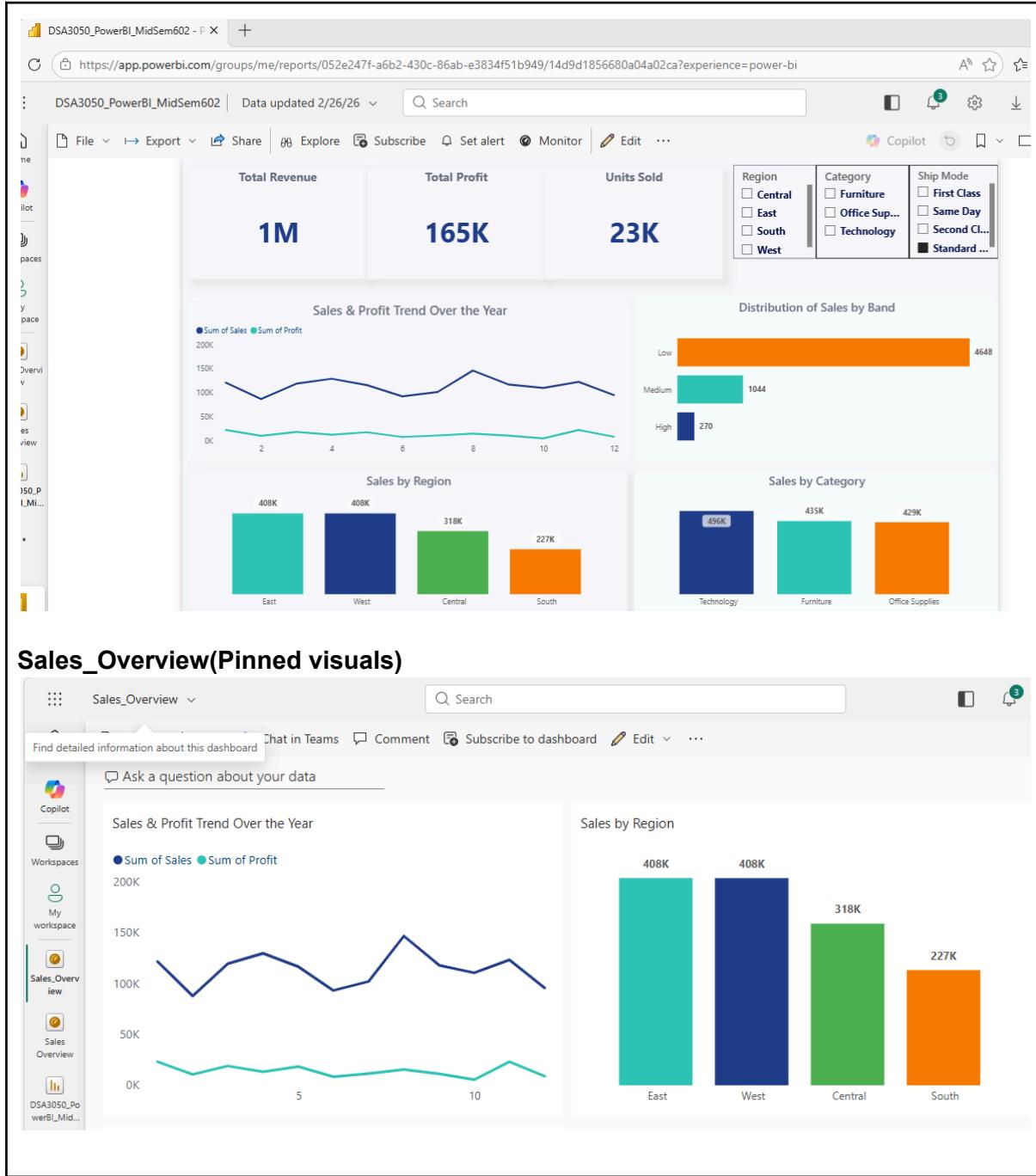


Figure 21: Power BI report successfully published to Power BI Service.

3.Section 2 — Dashboard Link (Public Link)

The interactive Power BI dashboard was successfully published to the Power BI Service and made publicly accessible using the *Publish to Web* feature.

Public Dashboard Link:

- <https://app.powerbi.com/groups/me/reports/052e247f-a6b2-430c-86ab-e3834f51b949/14d9d1856680a04a02ca?experience=power-bi>

This link allows users to view and interact with the dashboard, including slicers, cross-filtering, and visual insights.