Superstore Sales Data Analysis Project Documentation

Author: Anesha Dhali

Date: December 17, 2025

Project Overview: This document details the end-to-end process of downloading a sales dataset from Kaggle, normalizing it into relational tables using Excel, importing into MySQL Workbench, setting primary and foreign keys, data cleaning (including removing duplicates), and troubleshooting common issues. The goal was to create a star schema for analysis in Power BI, suitable for a data analyst portfolio.

Data Acquisition from Kaggle

Dataset Selected: Sales Data Analysis using MySQL, Excel, and Power BI.

Link: https://www.kaggle.com/datasets/poojacareer/sales-data-analysis-using-mysql-excel-and-power-bi)

Download Process:

- Visit the Kaggle page.
- Click "Download" to get the ZIP file containing multiple files (e.g., superstore_final_....xlsx, superstore.xlsx, etc.).
- Unzip the file to extract the contents.
- Key File Used: superstore_final_....xlsx (a large single table with orders, customers, products, etc., ~162 MB, containing columns like Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, Country, City, State, Postal Code, Region, Product ID, Category, Sub-Category, Product Name, Sales).
- Note: The dataset lacked Quantity, Discount, and Profit columns, so analysis focused on sales trends.

Data Normalization in Excel (Creating Separate CSV Sheets)

The original file was a flat table (denormalized). We normalized it into a star schema with dimension and fact tables.

Process:

- Open superstore_final_....xlsx in Excel.
- Create a new workbook and split into separate sheets using copy-paste, Remove Duplicates (Data tab), and Power Query for transformation.

Sheets Created:

Sheet name	Columns
Customers	customer_id, customer_name, segment
Products	product_id, product_name, category, sub_category
Region	postal_code, city, state, region, country
Orders	order_id, order_date, ship_date, ship_mode, customer_id, postal_code
Order_Details	row_id, order_id, product_id, sales

- For dimension tables (Customers, Products, Region): Use Remove Duplicates on the key column.
- For Orders: Remove duplicates on order_id.
- For Order_Details: Keep all rows (line items).
- Save each sheet as CSV: Right-click sheet → Move or Copy → New workbook → Save As → CSV UTF-8.
- Result: 5 CSV files (customers.csv, products.csv, region.csv, orders.csv, order_details.csv).

Importing CSV Files into MySQL Workbench

Setup:

- Open MySQL Workbench → Connect to local server (localhost).
- Create new schema: Right-click Schemas → Create Schema → Name superstore → Apply.

Import Process:

- Import order: Dimensions first (customers.csv, products.csv, region.csv), then orders.csv, order_details.csv.
- For each CSV:
  - Right-click Tables → Table Data Import Wizard.
  - Select CSV → Create new table (name: customers, products, region, orders, order_details).
  - Review data types (IDs as VARCHAR (50), dates as DATE, sales as DECIMAL (12,2)).
- Rename columns to snake_case (e.g., Customer_ID → customer_id) using Alter Table (Columns tab) or SQL:

SQL

```
ALTER TABLE customers CHANGE COLUMN `Customer_ID`
`customer_id` VARCHAR(50);
```

(Repeat for all columns/tables.)

Setting Primary Keys

Process:

- For each table: Right-click → Alter Table → Indexes tab → Add PRIMARY → Select key column → Apply.
- Or use SQL:

SQL

```
ALTER TABLE customers ADD PRIMARY KEY (customer_id);
ALTER TABLE products ADD PRIMARY KEY (product_id);
ALTER TABLE region ADD PRIMARY KEY (postal_code);
ALTER TABLE orders ADD PRIMARY KEY (order_id);
ALTER TABLE order_details ADD PRIMARY KEY (order_id,
product_id);  -- Composite
```

Data Cleaning and Removing Duplicates

- **Common Issues:** Duplicates in dimension tables (from flat file repetition), incorrect date formats, orphaned keys.

- **Cleaning Steps:**

- For duplicates in dimension tables (e.g., customers, products, region):

SQL

```
CREATE TABLE table_clean AS SELECT DISTINCT * FROM table_name;
DROP TABLE table_name;
RENAME TABLE table_clean TO table_name;
ALTER TABLE table_name ADD PRIMARY KEY (key_column);
```

(Applied to customers, products, region, orders.)

- For dates: Import as VARCHAR, then convert:

SQL

```
UPDATE orders SET order_date = STR_TO_DATE(order_date, '%d-%m-%Y');
ALTER TABLE orders MODIFY order_date DATE;
```

- For orphaned keys (e.g., product_id in order_details not in products):

SQL

```
DELETE od FROM order_details od
LEFT JOIN products p ON od.product_id = p.product_id
WHERE p.product_id IS NULL;
```

- For data types (e.g., IDs as TEXT):

SQL

```
ALTER TABLE orders MODIFY order_id VARCHAR(50) NOT NULL;
```

- Verification: Use GROUP BY to check counts, SELECT DISTINCT to ensure uniqueness.

Setting Foreign Keys

- **Process:**
- Run after primary keys and cleaning:

SQL

```
ALTER TABLE orders ADD FOREIGN KEY (customer_id) REFERENCES
customers(customer_id);
ALTER TABLE orders ADD FOREIGN KEY (postal_code) REFERENCES
region(postal_code);
ALTER TABLE order_details ADD FOREIGN KEY (order_id)
REFERENCES orders(order_id);
ALTER TABLE order_details ADD FOREIGN KEY (product_id)
REFERENCES products(product_id);
```

- This enforces relationships in the star schema.

Power BI Dashboard Development

- **Connection:** Connected Power BI Desktop to the MySQL superstore database (Import mode, all 5 tables loaded).
- **Data Transformations (Power Query Editor):**
  - Fixed date formats in orders table: Selected order_date and ship_date → Change Type → Using Locale → Date → English (United Kingdom) for correct DD-MM-YYYY parsing.
- **Data Model:** Relationships auto-detected (or manually confirmed) as star schema.
- **DAX Measures Created:**

dax

```
Total Sales = SUM(order_details[sales])
Number of Orders = DISTINCTCOUNT(orders[order_id])
Average Order Value = DIVIDE([Total Sales], [Number of
Orders])
Total Customers = DISTINCTCOUNT(customers[customer_id])
```

- **Dashboard Structure**

  - **Sales Overview:**

    - KPI Cards: Total Sales (~$118.12K), Number of Orders (4,922), Average Order Value ($24.00), Total Customers (793)

        - Line Chart: Sales over Time (by Year/Month)
        - Map Visual: Sales by State (bubble map with size and color by Total Sales)
        - Clustered Bar Chart: Sales by Category (drill-down to Sub-Category)

    - **Product Analysis:**

        - Horizontal Bar Chart: Top 10 Products by Sales
        - Donut Chart: Sales Share by Category
        - Matrix: Sales by Category and Sub-Category

    - **Customer & Regional Insights**:
        - Donut Chart: Sales by Segment
        - Bar Chart: Sales by Region
        - Table: Top 10 Customers by Sales

    - **Interactivity:** Added slicers for Year, Region, Ship Mode, and Category (synced across pages where appropriate).

    - **Design:** Consistent color theme, clear titles, tooltips for details

Key Insights from the Dashboard

- Total Sales: Approximately $118K across 4 years.
- Highest performing category: Technology (~59% share).
- Dominant customer segment: Consumer (~68%).
- Leading regions: West and states like California/New York.
- Sales trend: Growth from 2015, peak around 2017–2018.

Troubleshooting Section

This section summarizes the key issues encountered during the project and how they were resolved. These are common real-world data challenges that demonstrate problem-solving skills.

1. MySQL Import and Data Type Issues

- **Problem 1:** Dates imported in DD-MM-YYYY format caused "Incorrect date value" errors when converting to DATE type.

- **Solution 1:** Imported as VARCHAR initially, then converted using STR_TO_DATE(order_date, '%d-%m-%Y') or handled in Power BI Power Query (Change Type → Using Locale → English (United Kingdom)).

- **Problem 2**: ID columns (order_id, product_id) imported as TEXT/BLOB, causing Error 1170 when setting keys.
- **Solution 2**: ALTER TABLE table_name MODIFY column_name VARCHAR(50) NOT NULL;

2. Duplicate Entries

- **Problem:** Dimension tables (customers, products, region, orders) had duplicates, causing Error 1062 ("Duplicate entry") when setting primary keys.
- **Solution:** Used SELECT DISTINCT to recreate clean tables

```
SQL
CREATE TABLE table_clean AS SELECT DISTINCT * FROM
 table_name;
DROP TABLE table_name;
RENAME TABLE table_clean TO table_name;
ALTER TABLE table_name ADD PRIMARY KEY (key_column);
```

Applied to customers, products, region, and orders. For region, used GROUP BY TRIM(postal_code) with MAX() for other columns to handle minor differences (e.g., spaces).

3. Foreign Key Constraint Errors

- **Problem 1**: Error 1822 ("Missing index for constraint") — referenced column lacked primary key.
- **Solution 1**: Ensured primary keys were set on parent tables before adding foreign keys.

- **Problem 2**: Error 1452 ("Cannot add or update a child row") — orphaned values in child table.
- **Solution 2**: Deleted orphaned rows:

```
SQL
```

```
SET SQL_SAFE_UPDATES = 0;
DELETE od FROM order_details od LEFT JOIN products p ON
od.product_id  = p.product_id WHERE p.product_id IS NULL;
SET SQL_SAFE_UPDATES = 1;
```

- **Problem 3**: Error 1175 (Safe update mode blocked large DELETE).
- **Solution 3**: Temporarily disabled with SET SQL_SAFE_UPDATES = 0;.

4. Table Name and Index Issues

- **Problem 1:** Errors referencing wrong table name (e.g., 'regions' vs 'region').
- **Solution 1:** Verified table names with SHOW TABLES; and used correct singular/plural.

- **Problem 2:** "Multiple primary key defined" (Error 1068) or "Incorrect index name 'PRIMARY'".
- **Solution 2:** Ignored if key already existed (checked in Alter Table → Indexes tab).

5. Power BI Issues

- **Problem 1**: Filled Map had no "Values" (now "Color saturation") bucket.
- **Solution 1**: Set Data category on state column to "State or Province". Used regular Map visual (bubbles) as alternative.

- **Problem 2**: Dates parsed incorrectly.
- **Solution 2**: Power Query → Change Type → Using Locale → English (United Kingdom).

Conclusion

This project successfully transformed a raw, denormalized dataset into a clean relational database and an interactive dashboard. It demonstrates:

- Data cleaning and normalization
- SQL database design (star schema, keys, constraints)

- Power BI modeling and visualization

The final deliverables include:

- MySQL schema with ER diagram
- Power BI .pbix file and exported PDF/screenshots
- This documentation