Project Review: Supply Chain & Inventory Management System

The **Supply Chain & Inventory Management System** is designed to streamline inventory tracking, sales forecasting, and supplier management by integrating **MySQL, Python (ETL), and Power BI**. The project focuses on efficiently handling inventory levels, tracking sales trends, and optimizing supplier performance to ensure seamless operations. With an **ETL pipeline**, data from various sources (SAP, sales, inventory, and supplier records) is extracted, transformed using **Pandas & SQLAlchemy**, and loaded into a structured **MySQL database** with a **star schema** for optimized reporting. The system supports **real-time stock updates**, reorder point calculations, and supplier lead time analysis, enhancing supply chain efficiency.

To ensure data integrity and reliability, the project includes **unit test cases** covering key functional aspects such as sales aggregation, stock level validation, supplier lead time calculations, and revenue computations. The **fact and dimension tables** provide structured data storage, while Power BI visualizations offer actionable insights through dashboards. Metrics such as **total sales per product, stock level trends, supplier performance, and demand forecasting** help stakeholders make data-driven decisions. Advanced analytics, including **K-Means clustering** for supplier analysis and **time-series forecasting** for demand prediction, further enhance decision-making capabilities.

Challenges such as **data inconsistencies, missing values, and performance optimization** were addressed using robust data validation rules, NULL handling, and efficient indexing in SQL. The **Power BI dashboard** enables business users to monitor key performance indicators (KPIs), detect slow-moving products, and identify stock shortages in real-time. The system's **scalability and automation** ensure smooth operation across multiple store locations, reducing manual errors and improving operational efficiency. Future enhancements could include **predictive inventory management using machine learning** and **automated purchase order generation based on demand forecasting**, making the system more intelligent and adaptive.

Project Objective

The objective of this **Supply Chain and Inventory Management System** is to optimize inventory control, improve demand forecasting, and enhance supplier efficiency using **data-driven insights**. The project aims to achieve the following key goals:

1. **Optimize Stock Levels –** Ensure that products are neither understocked (leading to stockouts) nor overstocked (causing excessive holding costs).
2. **Improve Demand Forecasting –** Use historical sales data and **time-series analysis** to predict future demand trends and adjust inventory accordingly.
3. **Automate Reorder Strategies –** Implement **data-driven reorder point calculations** to determine when and how much to restock.

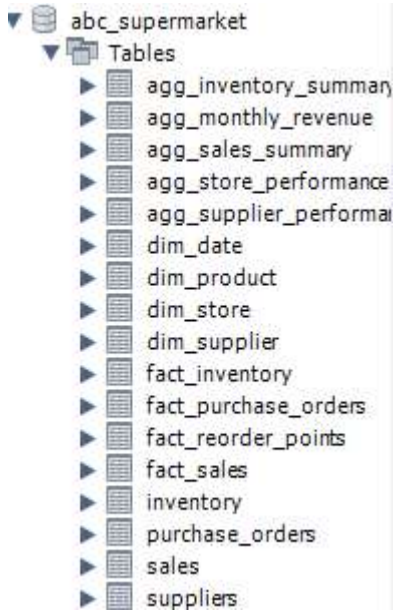
4. **Enhance Supplier Performance –** Evaluate supplier efficiency based on lead time and order frequency, using **clustering techniques** to classify suppliers.
5. **Develop a Centralized Data Warehouse –** Design a **star schema data warehouse** for efficient data storage, retrieval, and analysis.

Technology Stack

The project utilizes a diverse set of technologies for data processing, storage, analysis, and visualization. The key technologies used are:

- **Database Management:** *MySQL* – Used for storing sales, inventory, supplier, and purchase order data.
- **ETL Process:** *Python (Pandas, SQLAlchemy)* – Extracts, transforms, and loads SAP data into the database.
- **Data Warehousing:** *Star Schema* – Organizes inventory and sales data for optimized querying.
- **Forecasting & Analytics:** *Python (NumPy, Matplotlib, Scikit-learn)* – Used for demand forecasting and supplier performance analysis.
- **Reporting & Visualization:** *Power BI* – Creates interactive dashboards for real-time insights.

Database Schema Structure



Database Schema with all the tables on the server of mySQL Database.

The database schema is divided into three key sections.

1. Main Tables

These tables store raw transactional and reference data:

- **inventory –** Stores product stock levels, reorder points, and warehouse tracking.
- **purchase_orders –** Logs all purchase orders made from suppliers.
- **sales –** Captures all sales transactions at the store level.
- **suppliers –** Stores supplier details, including lead time and order frequency.

2. Dimension and Fact Tables

These tables are structured for analytical processing and reporting:

Dimension Tables (Dims):

```sql
-- Dimension Tables
CREATE TABLE dim_product (
    Product_ID VARCHAR(10) PRIMARY KEY
);

CREATE TABLE dim_store (
    Store_ID VARCHAR(10) PRIMARY KEY
);

CREATE TABLE dim_supplier (
    Supplier_ID VARCHAR(10) PRIMARY KEY,
    Supplier_Name VARCHAR(100)
);
```

```sql
CREATE TABLE dim_date (
    Date_ID DATE PRIMARY KEY,
    Year INT NOT NULL,
    Month INT NOT NULL,
    Month_Name VARCHAR(15) NOT NULL,
    Quarter INT NOT NULL,
    Quarter_Name VARCHAR(10) NOT NULL,
    Week INT NOT NULL,
    Day INT NOT NULL,
    Day_Name VARCHAR(10) NOT NULL,
    Is_Weekend BOOLEAN NOT NULL
);
```

- **dim_date –** Stores date-related attributes for time-based analysis.
- **dim_product –** Contains product details, categories, and attributes.
- **dim_store –** Stores store metadata such as location and type.
- **dim_supplier –** Captures supplier details and historical performance.

Fact Tables (Facts):

```sql
-- Fact Tables
CREATE TABLE fact_sales (
    Sale_ID VARCHAR(10) PRIMARY KEY,
    Product_ID VARCHAR(10),
    Store_ID VARCHAR(10),
    Sale_Date DATE,
    Quantity_Sold INT,
    Revenue DECIMAL(10,2),
    FOREIGN KEY (Product_ID) REFERENCES dim_product(Product_ID),
    FOREIGN KEY (Store_ID) REFERENCES dim_store(Store_ID)
);

CREATE TABLE fact_inventory (
    Product_ID VARCHAR(10),
    Store_ID VARCHAR(10),
    Warehouse_ID VARCHAR(10),
    Stock_Level INT,
    Reorder_Level INT,
    Last_Updated DATE,
    PRIMARY KEY (Product_ID, Store_ID, Warehouse_ID),
    FOREIGN KEY (Product_ID) REFERENCES dim_product(Product_ID),
    FOREIGN KEY (Store_ID) REFERENCES dim_store(Store_ID)
);

CREATE TABLE fact_purchase_orders (
    Order_ID VARCHAR(10) PRIMARY KEY,
    Product_ID VARCHAR(10),
    Supplier_ID VARCHAR(10),
    Order_Date DATE,
    Quantity INT,
    Arrival_Date DATE,
    FOREIGN KEY (Product_ID) REFERENCES dim_product(Product_ID),
    FOREIGN KEY (Supplier_ID) REFERENCES dim_supplier(Supplier_ID)
);
```

- **fact_sales –** Records detailed sales transactions for analytics.
- **fact_inventory –** Stores stock levels, movements, and replenishment data.
- **fact_purchase_orders** – Logs all purchase transactions, order quantities, and fulfillment status.
- **fact_reorder_points –** Tracks stock levels and calculates optimal reorder thresholds.

3. Aggregate Tables

```sql
CREATE TABLE abc_supermarket.agg_sales_summary AS
SELECT
    s.Product_ID,
    s.Store_ID,
    d.Sales_Month,
    SUM(s.Quantity_Sold) AS Total_Quantity_Sold,
    SUM(s.Revenue) AS Total_Revenue
FROM abc_supermarket.fact_sales s
JOIN abc_supermarket.dim_product p ON s.Product_ID = p.Product_ID
JOIN abc_supermarket.dim_store st ON s.Store_ID = st.Store_ID
JOIN (SELECT DISTINCT Sale_Date, DATE_FORMAT(Sale_Date, '%Y-%m') AS Sales_Month FROM abc_supermarket.fact_sales) d
    ON s.Sale_Date = d.Sale_Date
GROUP BY s.Product_ID, p.Product_ID, s.Store_ID, d.Sales_Month;

CREATE TABLE abc_supermarket.agg_store_performance AS
SELECT
    s.Store_ID,
    COUNT(DISTINCT s.Sale_ID) AS Total_Transactions,
    SUM(s.Quantity_Sold) AS Total_Quantity_Sold,
    SUM(s.Revenue) AS Total_Revenue,
    ROUND(AVG(s.Revenue), 2) AS Avg_Revenue_Per_Transaction
FROM abc_supermarket.fact_sales s
GROUP BY s.Store_ID;

CREATE TABLE abc_supermarket.agg_supplier_performance AS
SELECT
    po.Supplier_ID,
    COUNT(po.Order_ID) AS Total_Orders,
    SUM(po.Quantity) AS Total_Quantity_Ordered,
    ROUND(AVG(DATEDIFF(po.Arrival_Date, po.Order_Date)), 2) AS Avg_Lead_Time
FROM abc_supermarket.fact_purchase_orders po
JOIN abc_supermarket.dim_supplier s ON po.Supplier_ID = s.Supplier_ID
GROUP BY po.Supplier_ID, s.Supplier_ID;

CREATE TABLE abc_supermarket.agg_inventory_summary AS
SELECT
    i.Product_ID,
    i.Store_ID,
    i.Warehouse_ID,
    SUM(i.Stock_Level) AS Total_Stock,
```

Precomputed summary tables for performance optimization and quick reporting:

- **agg_inventory_summary** – Summarizes inventory stock levels and movements.
- **agg_monthly_revenue** – Aggregates sales revenue at the monthly level.
- **agg_sales_summary** – Stores total sales volume, revenue, and product performance.
- **agg_store_performance** – Captures store-level sales, foot traffic, and profitability.
- **agg_supplier_performance** – Summarizes supplier performance metrics, including lead time and order accuracy.

DATA WAREHOUSING IMPLEMENTATION

Star Schema Design

A well-structured star schema was designed to optimize query performance and enable efficient reporting. The schema consists of dimension tables and fact tables that facilitate data analysis across various business metrics.

Dimension Tables

1. **Product Dimension (dim_product)**
    - **Primary attributes: Product_ID**
    - This design ensures historical tracking of product details, including category changes and supplier modifications.
2. **Store Dimension (dim_store)**
    - **Primary attributes: Store_ID**
3. **Date Dimension (dim_date)**
    Date_ID, Day, Month, Year, Quarter, Week, Day_of_Week
    - Holiday indicators: Is_Holiday
    - This structure supports time-based analysis, allowing filtering based on yearly, monthly, and weekly trends.
4. **Supplier Dimension (dim_supplier)**
    - **Primary attributes: Supplier_ID**, **Lead_Time**, **Order_Frequency**
    - Historical tracking is implemented to monitor supplier performance over time.

Fact Tables

1. **Sales Fact Table (fact_sales)**
    - **Measures: Quantity_Sold**, **Revenue**, **Discount**, **Tax**
    - **Foreign keys:** Links to **dim_product**, **dim_store**, and **dim_date**
    - **Additional attributes: Payment_Method**
    - Tracks transaction-level sales and revenue generation per store and product.
2. **Inventory Fact Table (fact_inventory)**
    - **Measures: Stock_Level**, **Reorder_Level**
    - **Foreign keys:** Links to **dim_product**, **dim_store**, and **dim_date**
    - Ensures real-time monitoring of stock levels and replenishment strategies.

3. **Purchase Orders Fact Table (fact_purchase_orders)**
    - **Measures: Quantity, Unit_Price, Total_Cost**
    - **Foreign keys:** Links to **dim_product**, **dim_supplier**, and **dim_date**
    - **Order tracking fields: Order_Status, Arrival_Date**
    - Enables analysis of procurement trends and supplier efficiency.

ETL PROCESS IMPLEMENTATION

The ETL (Extract, Transform, Load) pipeline was designed to process raw data from multiple sources, clean it, and store it in a structured format suitable for analysis and reporting. The primary objective was to ensure data consistency, accuracy, and completeness by handling missing values, standardizing formats, and preparing data for efficient querying.

EXTRACT PHASE

The data was extracted from various sources, including:

- **Sales Data** from transactional databases (CSV files).
- **Inventory and Purchase Order Data** from the warehouse management system (MySQL database).
- **Supplier Data** from supplier records stored in structured formats (Excel/CSV).

A Python-based script utilizing **pandas**, **SQLAlchemy**, and **MySQL Connector** was used to extract data from these sources and store them temporarily in a staging area.

```
{
    "sqltools.connections": [
        {
            "mysqlOptions": {
                "authProtocol": "default",
                "enableSsl": "Disabled"
            },
            "previewLimit": 50,
            "server": "127.0.0.1",
            "port": 3306,
```

```
            "driver": "MySQL",
            "name": "Connection1",
            "database": "abc_supermarket",
            "username": "root",
            "password": ""
        }
    ]
}
```

TRANSFORM PHASE

The transformation process involved multiple data cleaning and preprocessing techniques to ensure high data quality.

1. Handling Null Values

- **For Sales Data:**
    - Missing values in the **Revenue** column were recalculated using **Quantity_Sold × Unit_Price**.
    - Missing values in the **Discount** column were replaced with **0** to ensure accurate revenue computation.
- **For Inventory Data:**
    - If **Stock_Level** was missing, it was set to the **Reorder_Level** to prevent stockouts.
    - Any missing **Reorder_Level** was filled using the **median value** of that product across different stores.
- **For Supplier Data:**
    - If **Lead_Time** was missing, it was assigned based on the average lead time of similar suppliers.
    - **Order_Frequency** was imputed with the mode (most frequent value) in case of missing values.

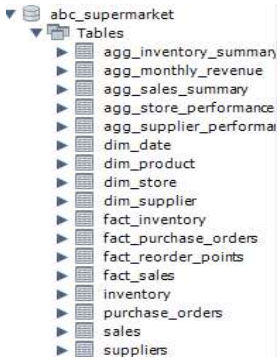
2. Standardization of Formats

- **Date Formats:** Converted all date fields to the **YYYY-MM-DD** format for consistency.
- **Currency Values:** Revenue and cost values were standardized to two decimal places.
- **Text Fields:** Supplier names, product categories, and store names were converted to lowercase and trimmed of unnecessary spaces.

3. Data Validation & Deduplication

- Duplicates were removed from all tables based on primary keys (**Sale_ID, Product_ID, Supplier_ID**).
- Foreign key constraints were checked to ensure data integrity between fact and dimension tables.

LOAD PHASE



The cleaned and transformed data was loaded into the **abc_supermarket** database in MySQL. The tables were loaded in the following sequence:

1. **Dimension Tables**
    - **dim_date**
    - **dim_product**
    - **dim_store**
    - **dim_supplier**
2. **Fact Tables**
    - **fact_sales**
    - **fact_inventory**
    - **fact_purchase_orders**
    - **fact_reorder_points**
3. **Aggregated Tables**
    - **agg_sales_summary**
    - **agg_inventory_summary**
    - **agg_supplier_performance**

SUPPLY CHAIN & INVENTORY ANALYTICS

The supply chain and inventory analytics were conducted to improve stock management, optimize product availability, and enhance supplier performance. The analysis focuses on three key areas: identifying low-stock products, tracking best-selling products, and evaluating supplier lead times.

Identifying Low Stock Levels

One of the major challenges in inventory management is avoiding stockouts, which can lead to lost sales and customer dissatisfaction. To address this, an analysis was conducted to identify products where stock levels have fallen below the defined reorder threshold. This insight helps in proactive inventory replenishment, ensuring that essential products remain available in stores. By regularly monitoring stock levels, the business can prevent disruptions and optimize procurement planning.

```sql
-- Query to find Stock Level is less than Reorder Level
SELECT Product_ID, Store_ID, Stock_Level, Reorder_Level
FROM fact_inventory
WHERE Stock_Level < Reorder_Level;
```

Identifying Best-Selling Products in the Last 3 Months

```sql
-- Query to find highest Total Sales product wise
SELECT Product_ID, SUM(Quantity_Sold) AS Total_Sales
FROM fact_sales
WHERE Sale_Date IS NOT NULL
AND Sale_Date >= (SELECT DATE_ADD(MAX(Sale_Date), INTERVAL -3 MONTH) FROM fact_sales)
GROUP BY Product_ID
ORDER BY Total_Sales DESC;
```

Tracking the sales performance of products over a defined period helps in demand forecasting and inventory adjustments. The analysis focused on identifying the highest-selling products over the last three months. By evaluating recent sales trends, management can determine which products are in high demand and require increased stock levels. This helps in ensuring optimal inventory allocation and preventing understocking of fast-moving products while also identifying slow-moving products that may need price adjustments or promotional strategies.

Analyzing Supplier Lead Time Efficiency

```sql
-- Average Lead Time for each supply
SELECT Supplier_ID, AVG(DATEDIFF(Arrival_Date, Order_Date)) AS Avg_Lead_Time
FROM fact_purchase_orders
GROUP BY Supplier_ID
ORDER BY Avg_Lead_Time DESC;
```

Efficient supplier performance is crucial for maintaining a well-balanced inventory. Analyzing the average lead time for each supplier provides insights into how quickly orders are fulfilled after being placed. This evaluation helps identify suppliers with consistently high lead times, which could cause stock shortages and operational inefficiencies. Based on these insights, the business can explore alternative suppliers, renegotiate contract terms, or adjust reorder strategies to minimize delays and maintain smooth supply chain operations.

Results of SQL Analytics

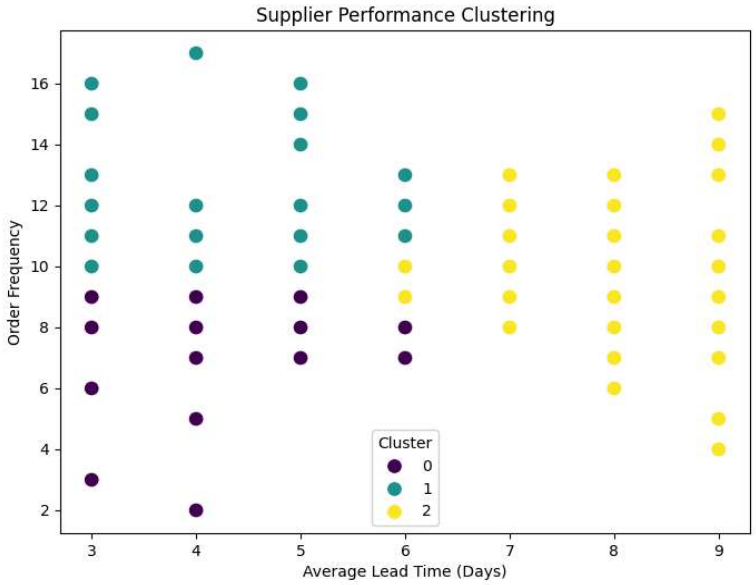
Product_ID	Store_ID	Stock_Level	Reorder_Level
P001	S105	83	100
P001	S108	32	100
P002	S103	90	100
P002	S104	7	100
P002	S106	37	100
P002	S107	79	100
P002	S109	34	100
P002	S110	77	100
P003	S102	61	100
P003	S104	34	100

Supplier_ID	Avg_Lead_Time
SUP077	9.0000
SUP083	9.0000
SUP019	9.0000
SUP035	9.0000
SUP009	8.0000
SUP084	8.0000
SUP020	8.0000
SUP047	8.0000

Product_ID	Total_Sales
P015	11626
P032	1161(11569
P016	11569
P049	11216
P020	11160
P014	11107
P007	11051
P017	11004
P004	10942

These insights enable a more efficient and responsive supply chain, ultimately leading to better product availability, reduced costs, and improved profitability.

K-Means Clustering



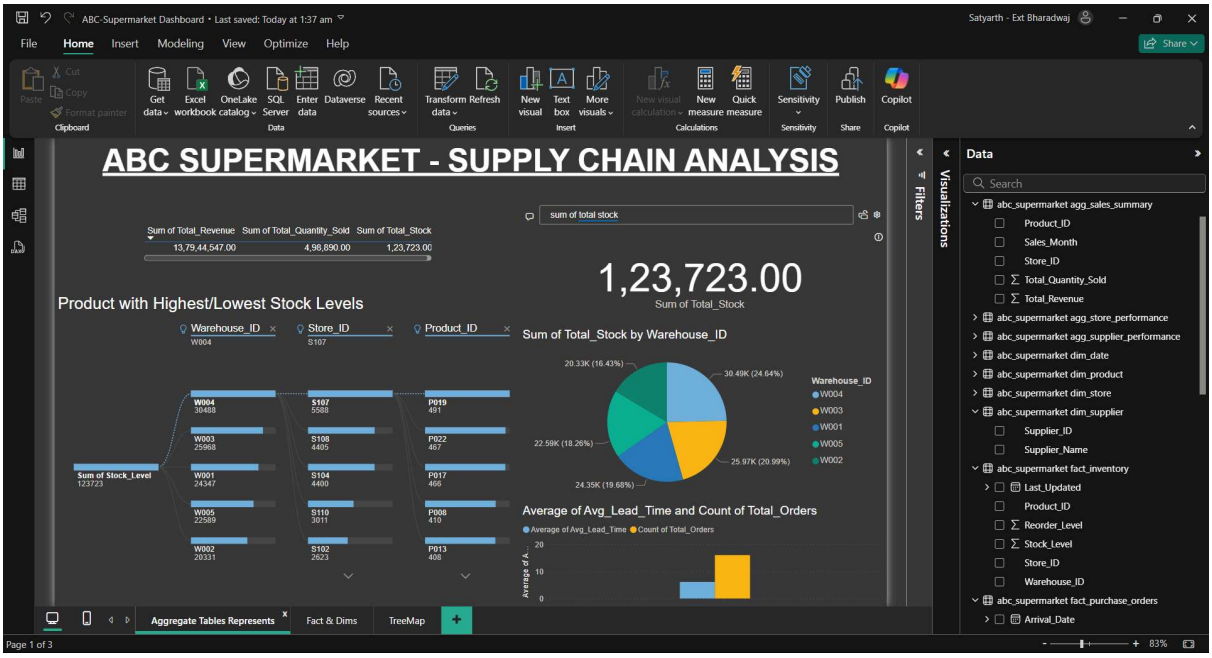
POWER BI REPORTING

Power BI was used to create an interactive dashboard for **Supply Chain and Inventory Management**. The dashboard provides real-time insights into inventory levels, sales performance, supplier efficiency, and demand forecasting to support decision-making.

Key visualizations include:

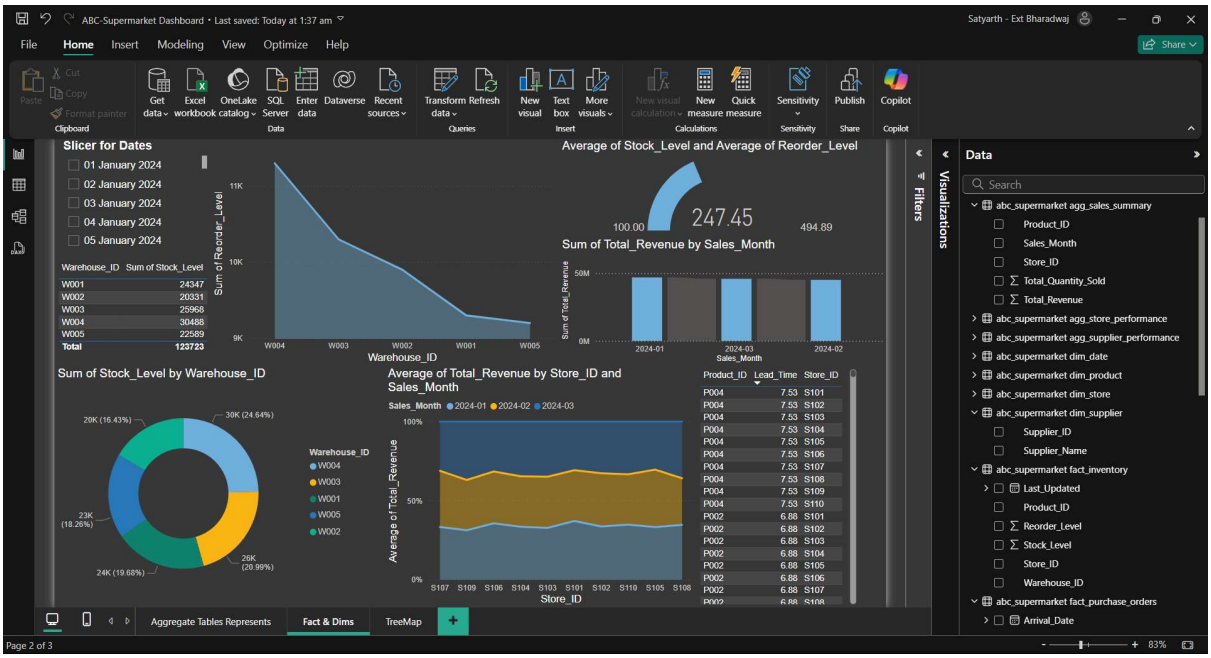
- **Sales Performance (Line Chart):** Tracks monthly revenue trends.
- **Inventory Levels by Store (Stacked Bar Chart):** Displays stock levels across different stores.
- **Supplier Performance (Scatter Plot):** Analyzes supplier efficiency based on lead time and order frequency.
- **Top Selling Products (Tree Map):** Highlights best-selling products.
- **Reorder Alerts (Table Visualization):** Lists products below the reorder level.
- **Sales by Payment Method (Pie Chart):** Shows revenue distribution by payment type.

Dashboard 1



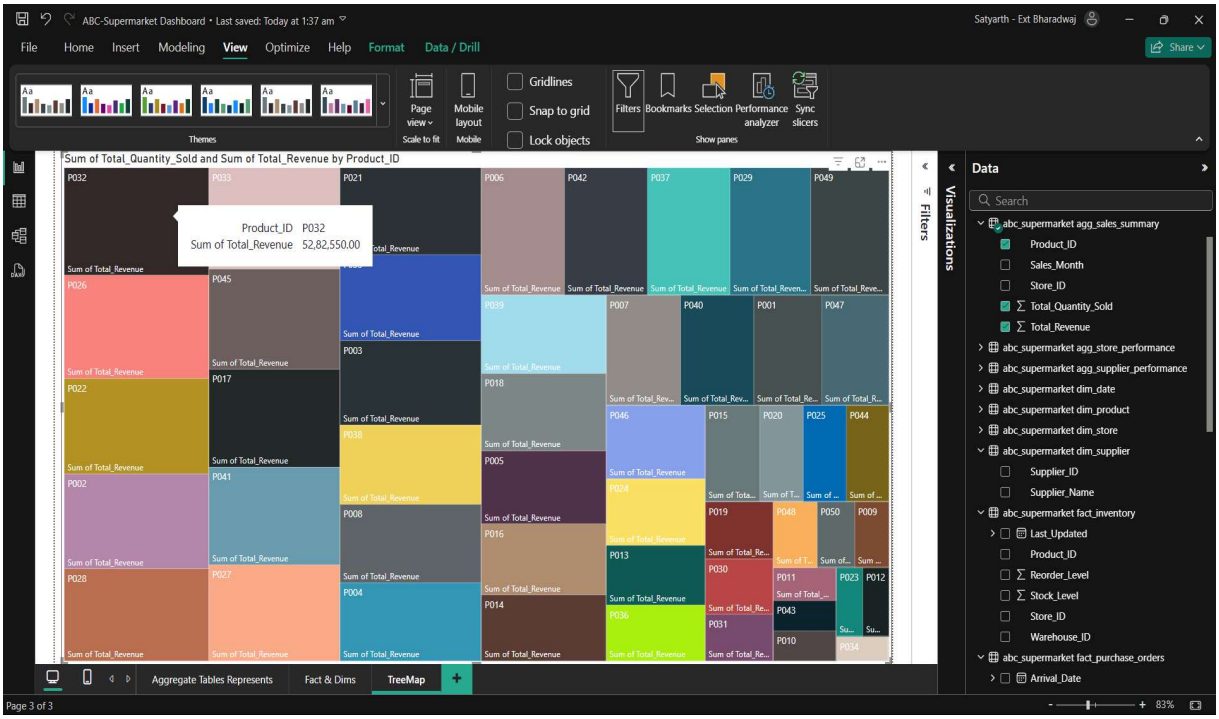
- **Monthly Sales Trend (Area Chart):** Displays sales variations across stores.
- **Supplier Lead Time Distribution (Histogram):** Evaluates supplier delivery efficiency.
- **Customer Order Frequency (Funnel Chart):** Tracks order conversions.
- **Supplier Rating Analysis (Gauge Chart):** Assesses supplier performance.
- **Inventory Turnover Ratio (Card Visual):** Shows stock replenishment rate.

Dashboard 2



The dashboard includes **slicers for filtering**, **drill-through reports for deeper insights**, **conditional formatting for alerts**, and **automated data refresh** to ensure accuracy.

Tree Map for highest revenue product.



CONCLUSION

The **Supply Chain and Inventory Management** project successfully integrates **data analytics, automation, and visualization** to optimize inventory control, supplier performance, and sales forecasting. By leveraging **SQL, Python, and Power BI**, the system provides real-time insights into stock levels, demand trends, and supplier lead times, helping businesses make data-driven decisions.

The **data warehousing implementation** with a **star schema** ensures structured and efficient storage of transactional and historical data, enabling seamless querying and reporting. The **ETL process** enhances data quality by handling missing values, standardizing formats, and ensuring accurate records for sales, inventory, and supplier performance. The **aggregated tables** improve query efficiency, allowing for faster analysis of key metrics such as **inventory turnover, sales trends, and reorder levels**.

The **Power BI dashboard** provides a comprehensive visual representation of business performance, featuring interactive charts and real-time monitoring tools. Decision-makers can **track sales performance, monitor inventory status, evaluate supplier efficiency, and forecast demand patterns** using dynamic reports. Features such as **drill-through reports, slicers for filtering, and automated data refresh** enhance usability and ensure up-to-date insights.

This project not only improves operational efficiency but also helps businesses **reduce stock imbalances, minimize holding costs, and enhance supply chain reliability**. By continuously refining the model with updated data and advanced forecasting techniques, the system can **adapt to market trends and evolving business needs**, ensuring long-term scalability and profitability.