**VISHWAKARMA UNIVERSITY**

Survey No. 2, 3, 2, Laxmi Nagar, Kondhwa Budruk, Pune – 411 048

Maharasthra, India.



A Project Report on,

**Retail Supermarket Data Analysis**

By,

**ABHINAV KUMAR**

**(SRN: 31231517)**

Guided By

**Dr. Anupriya Kamble**

course Name

**Data Mining and Analytics**

For the Programme

**S. Y. BSC. COMPUTER SCIENCE**

In the Year

**2024-25**

Pursued in

**Department of Computer Science**

**Faculty of Science & Technology**

**Vishwakarma University, Pune**

**Introduction**

The Shoppe, a Tamil Nadu-based supermarket grocery sales chain, seeks to expand its operations from a regional to a national level. To support this ambitious growth, the retail chain aims to gain a deeper understanding of its business performance, customer demands, and product sales trends. By leveraging data-driven insights, The Shoppe intends to optimize its inventory, enhance customer satisfaction, and make informed strategic decisions for scaling its operations across India. To achieve these goals, the supermarket has engaged a team of software analysts to analyze its sales data and provide actionable recommendations.

The dataset provided, "Supermart Grocery Sales - Retail Analytics Dataset.csv," contains 9,994 records and 11 columns, capturing detailed transaction information such as order IDs, customer names, product categories, sub-categories, sales, profits, discounts, and regional details. This rich dataset serves as the foundation for a comprehensive analysis aimed at uncovering key business metrics and predicting growth opportunities.

The primary objectives of this project are to:

1. Identify at least 10 key performance indicators (KPIs) to evaluate the supermarket’s sales, profitability, and customer behavior.
2. Perform data preprocessing, outlier detection, and exploratory data analysis (EDA) to derive insights from the identified KPIs.
3. Apply association rule mining to discover frequently purchased product combinations, informing bundling and promotional strategies.
4. Develop classification models to predict high-sales and high-profit product categories and sub-categories, with a focus on regional and state-wise performance.
5. Modify the dataset to incorporate non-volatile data warehouse features, ensuring data integrity and traceability for future analysis.

To address these objectives, a Python-based analytical pipeline was developed, utilizing libraries such as pandas, scikit-learn, and mlxtend. The pipeline includes robust data preprocessing, visualization of KPIs, Apriori algorithm for association rule mining, Random Forest classification with SMOTE for handling class imbalance, and a versioning system for data warehouse implementation. The analysis is conducted within a virtual environment, with all outputs (visualizations, reports, and datasets) organized in a dedicated folder for clarity and reproducibility.

This report presents the methodology, findings, and recommendations derived from the analysis. It includes detailed insights into sales trends, customer purchasing patterns, and predictive models, supported by visualizations and a workflow diagram. The outcomes aim to empower The Shoppe with the knowledge needed to enhance its business operations and successfully scale to a national level.

**Objectives**

The primary goal of this project is to analyze the sales and customer data of The Shoppe, a Tamil Nadu-based supermarket grocery sales chain, to support its strategic expansion to a national level. By leveraging the "Supermart Grocery Sales - Retail Analytics Dataset.csv," the project aims to provide actionable insights and predictive models to enhance business performance, optimize operations, and inform decision-making. The specific objectives of the project are as follows:

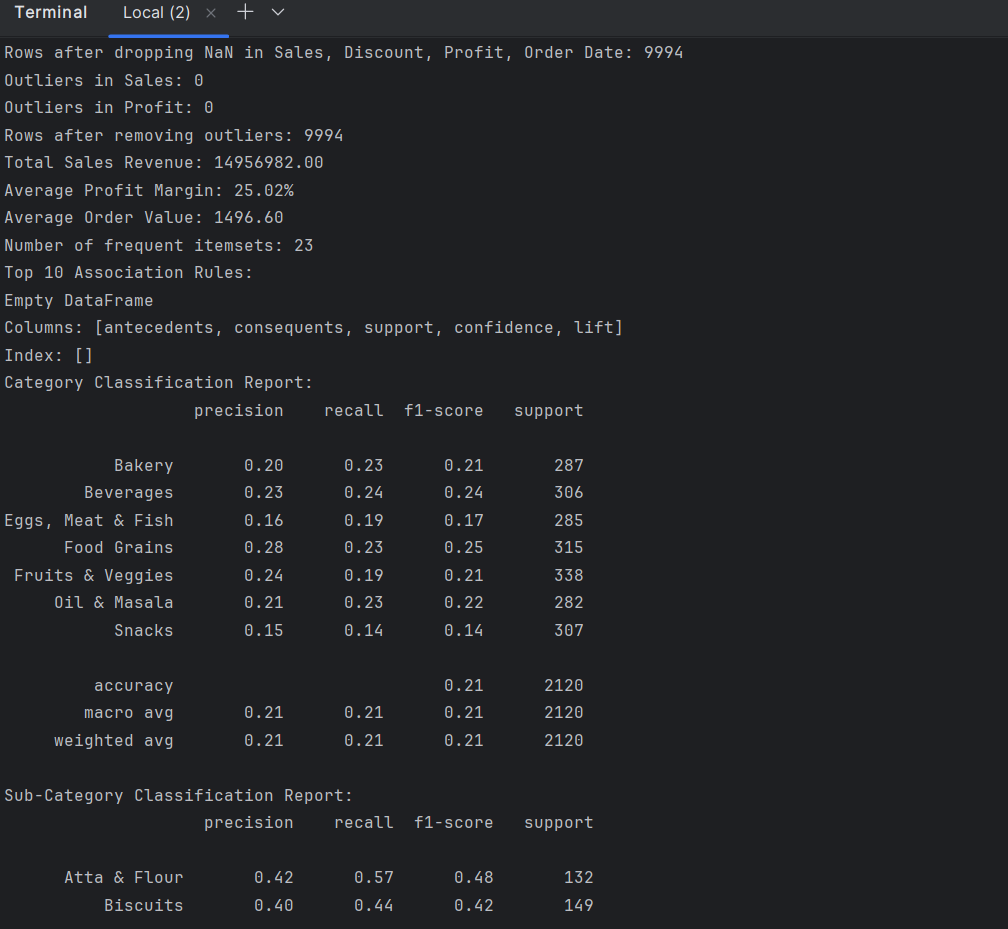
1. **Identify Key Performance Indicators (KPIs)**: Define at least 10 major KPIs to evaluate critical business metrics such as total sales revenue, profit margins, customer purchase frequency, and regional sales performance. These KPIs will provide a comprehensive understanding of the supermarket’s current operations and growth potential.
2. **Perform Data Preprocessing and Exploratory Data Analysis (EDA)**: Load and preprocess the dataset to handle missing values, ensure correct data types, and detect and remove outliers. Conduct EDA to derive insights into the identified KPIs through statistical computations and visualizations, highlighting trends in sales, profitability, and customer behavior across categories, sub-categories, and regions.
3. **Apply Association Rule Mining**: Utilize the Apriori algorithm to identify frequently purchased product combinations (sub-categories) and their demand patterns. This analysis will uncover items that are often bought together, enabling The Shoppe to develop targeted bundling strategies and promotional campaigns.
4. **Develop Classification Models**: Build predictive models using classification techniques to identify product categories and sub-categories that yield the highest sales and profits, with a focus on region-wise and state-wise performance. These models will help prioritize product offerings and optimize inventory allocation in different geographic markets.
5. **Implement Non-Volatile Data Warehouse Features**: Modify the dataset to incorporate non-volatile data warehouse characteristics, ensuring that data cannot be updated or deleted but only appended with version tracking and timestamps. This will support long-term data integrity and facilitate future analytical needs as the business scales.

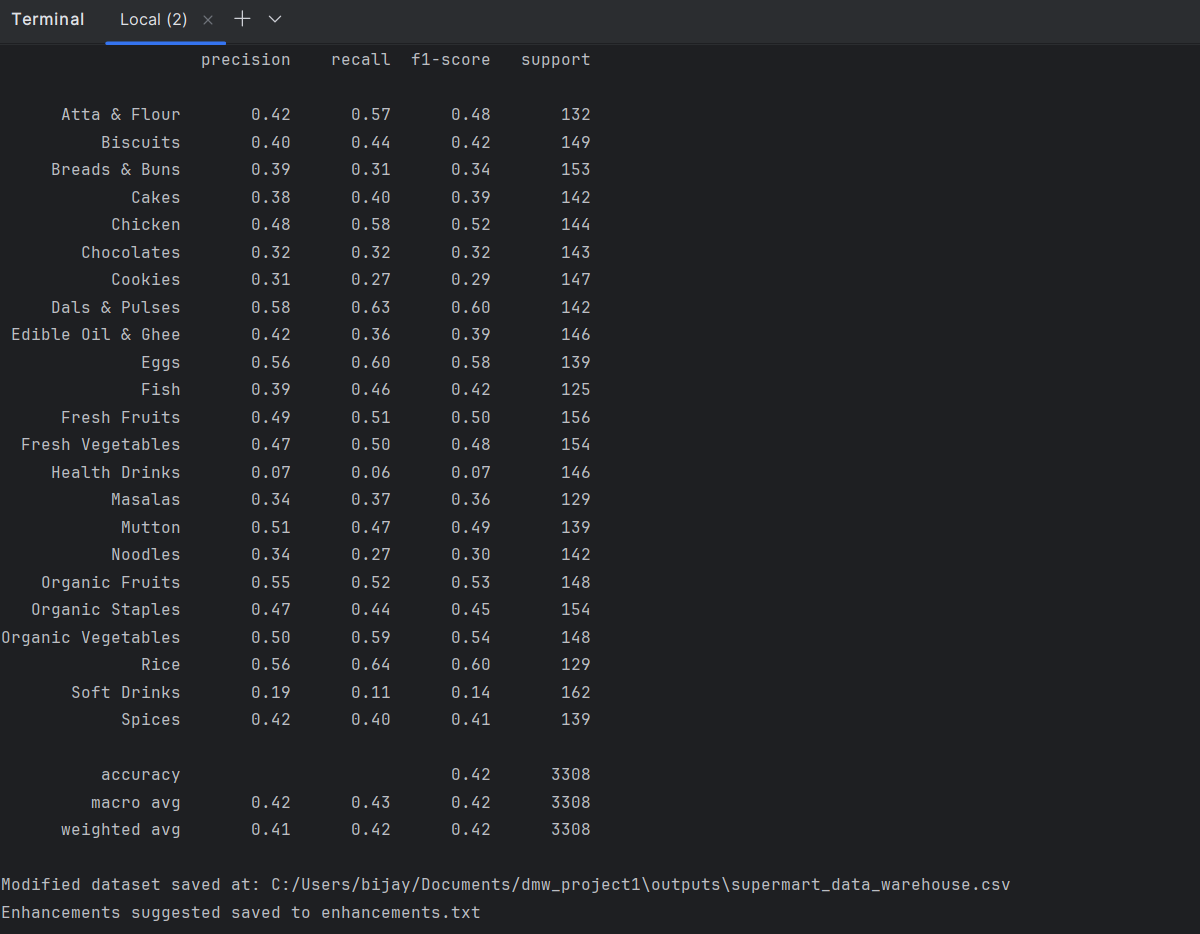
These objectives collectively aim to provide The Shoppe with a robust analytical framework to understand its current market position, predict future trends, and make data-driven decisions for national expansion. The outcomes will be supported by visualizations, predictive models, and a structured data warehouse, all organized within a single project directory for ease of access and reproducibility.

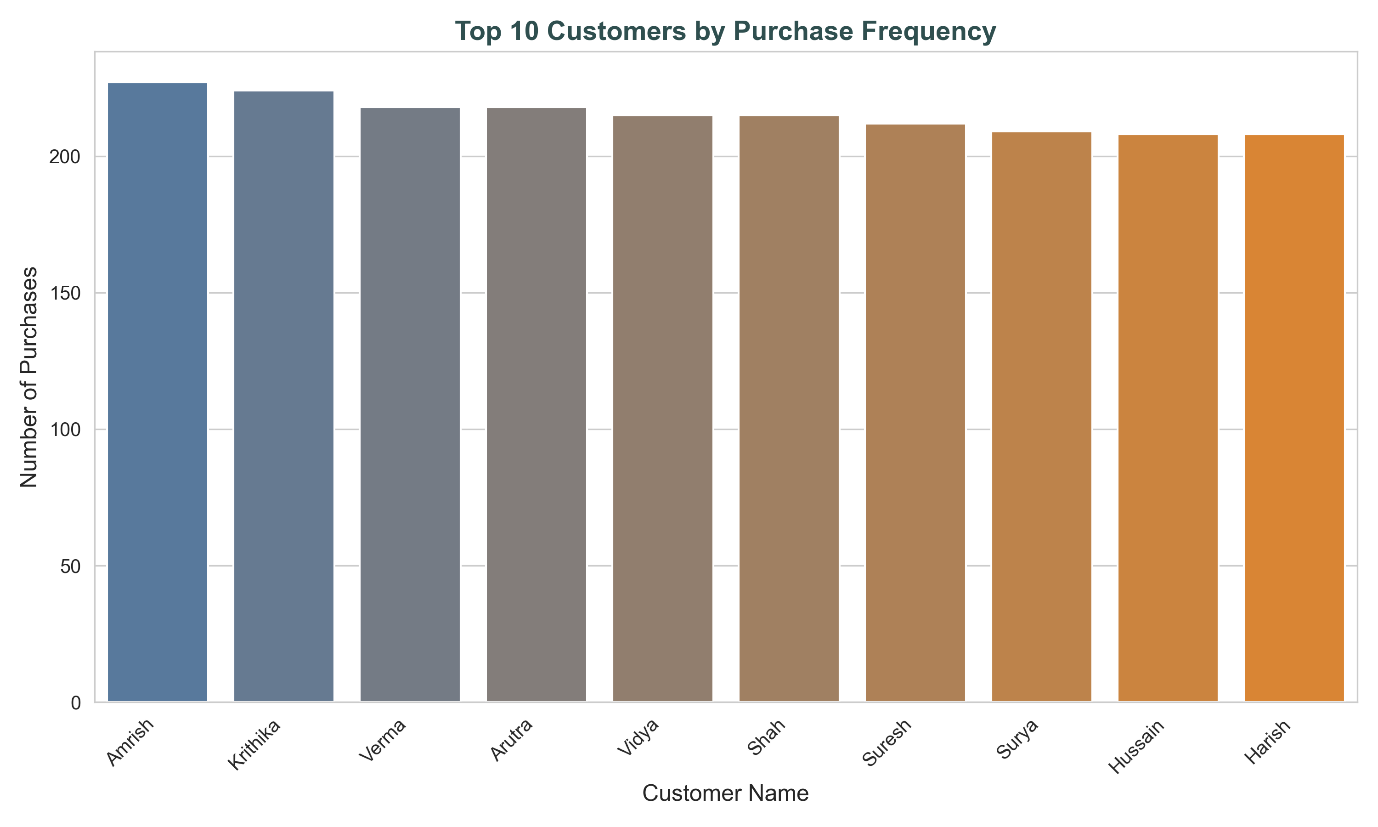
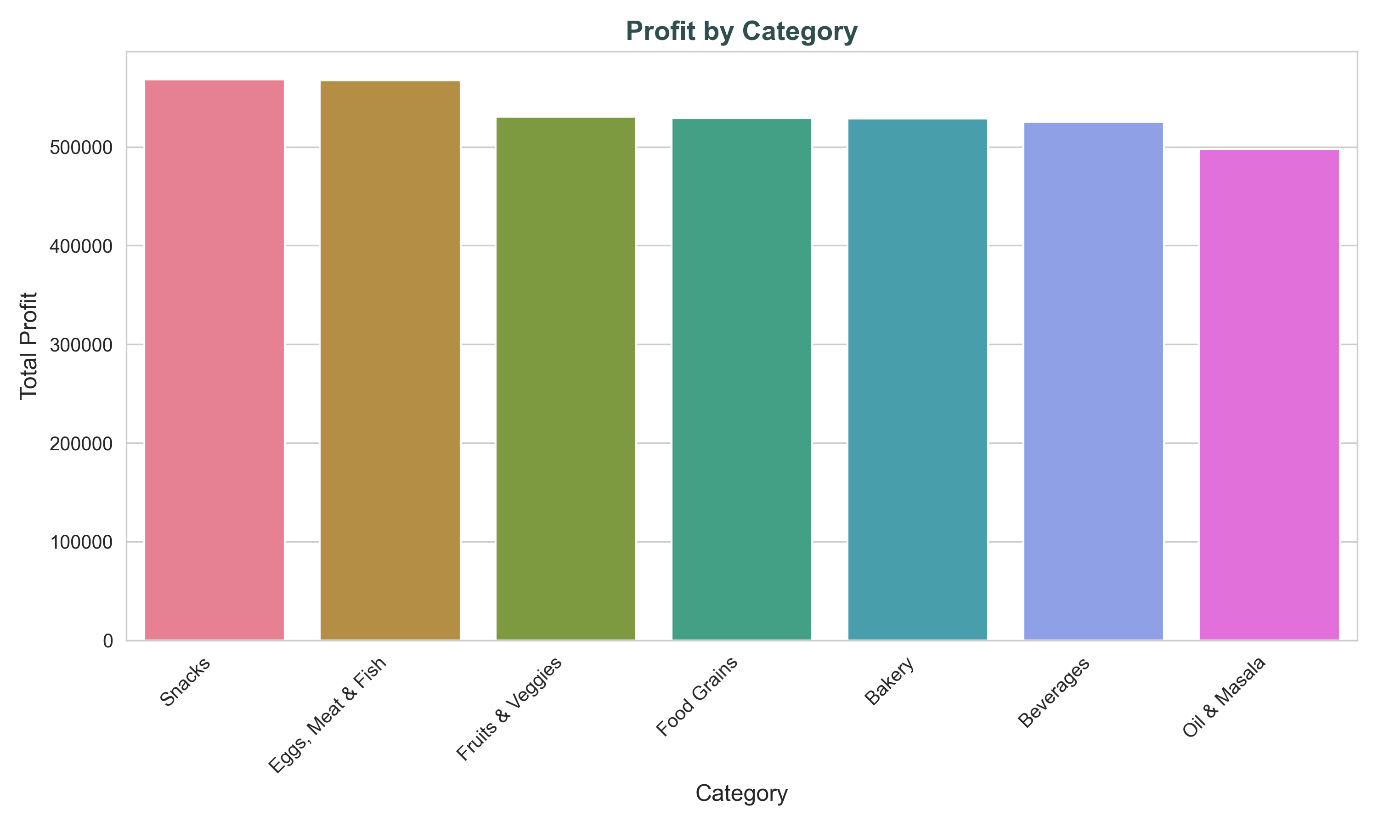
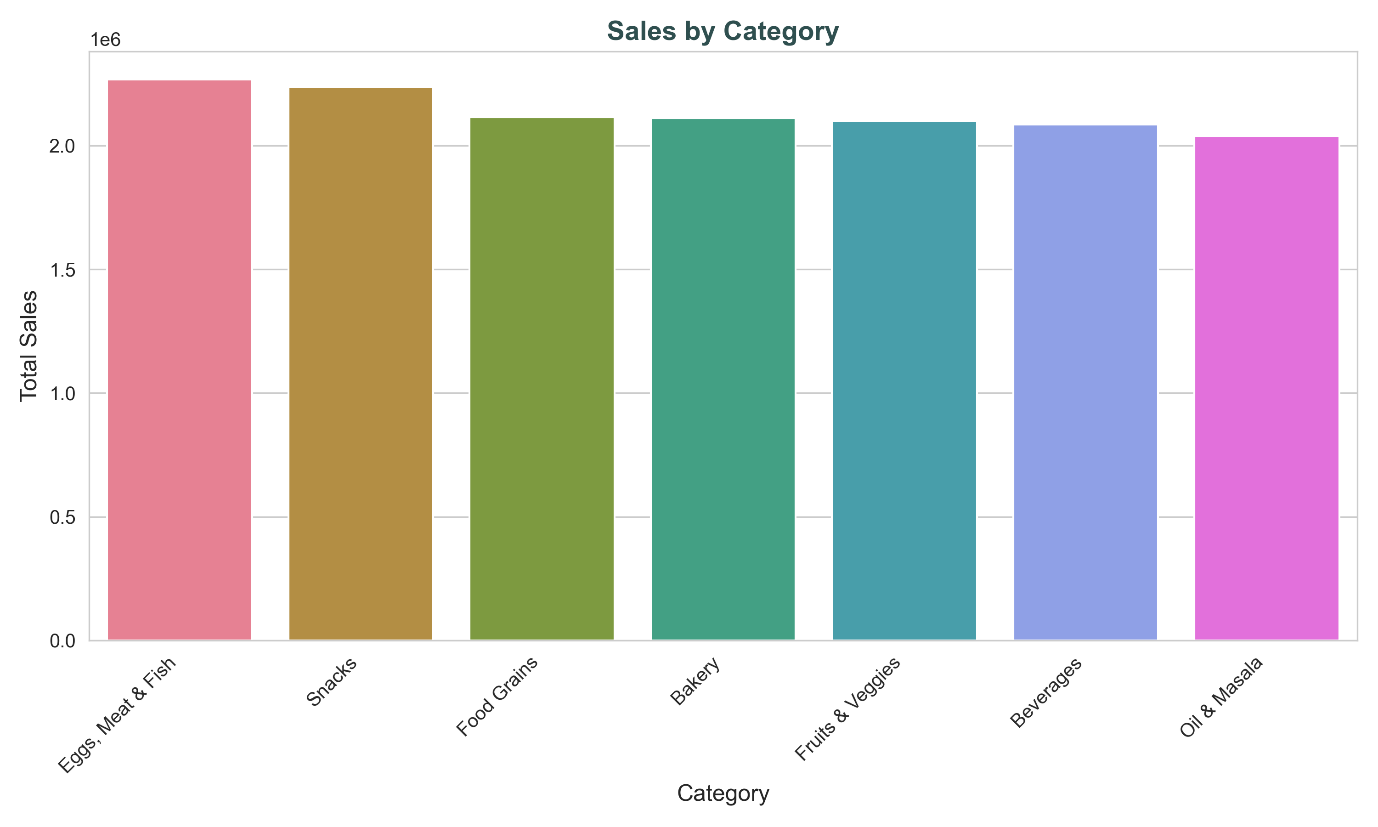
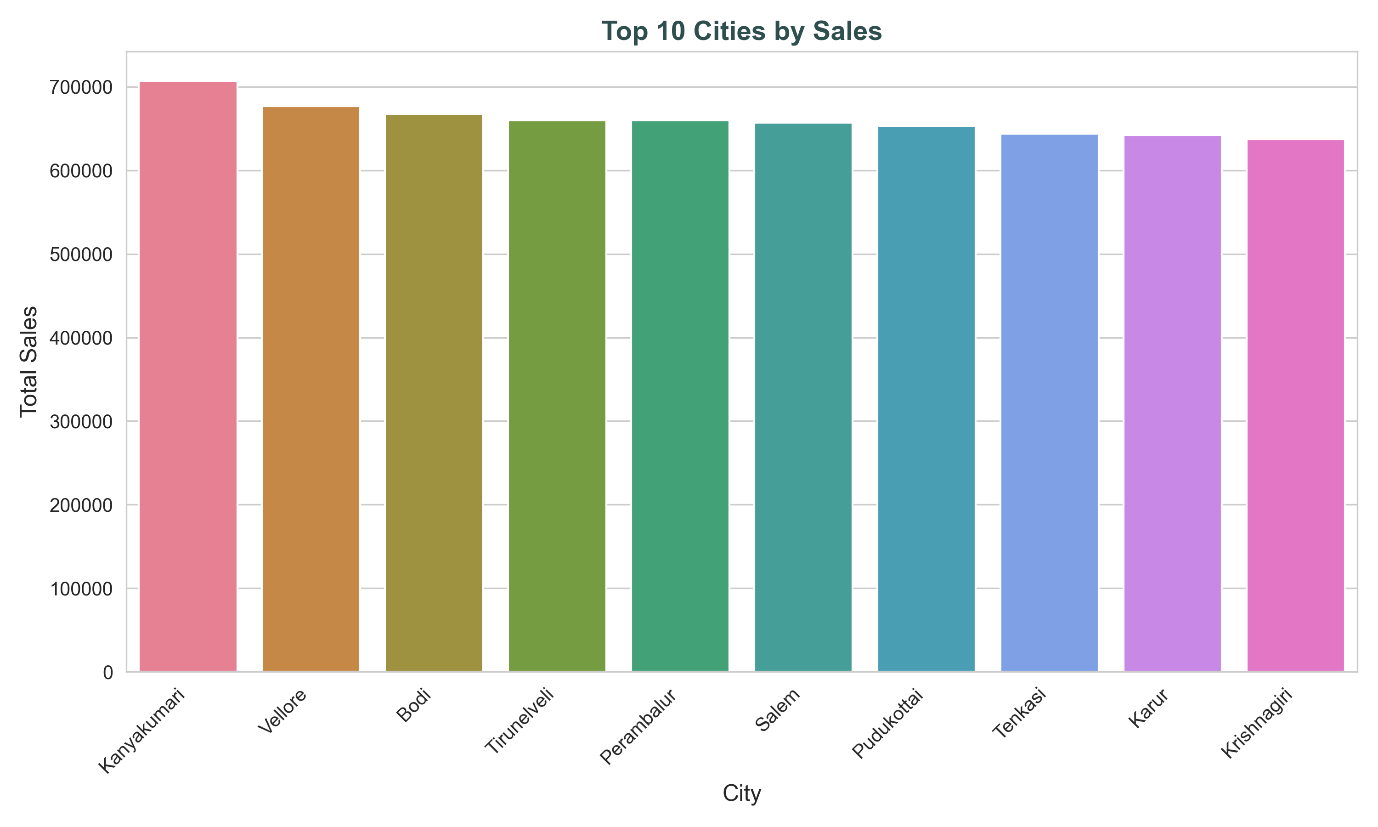
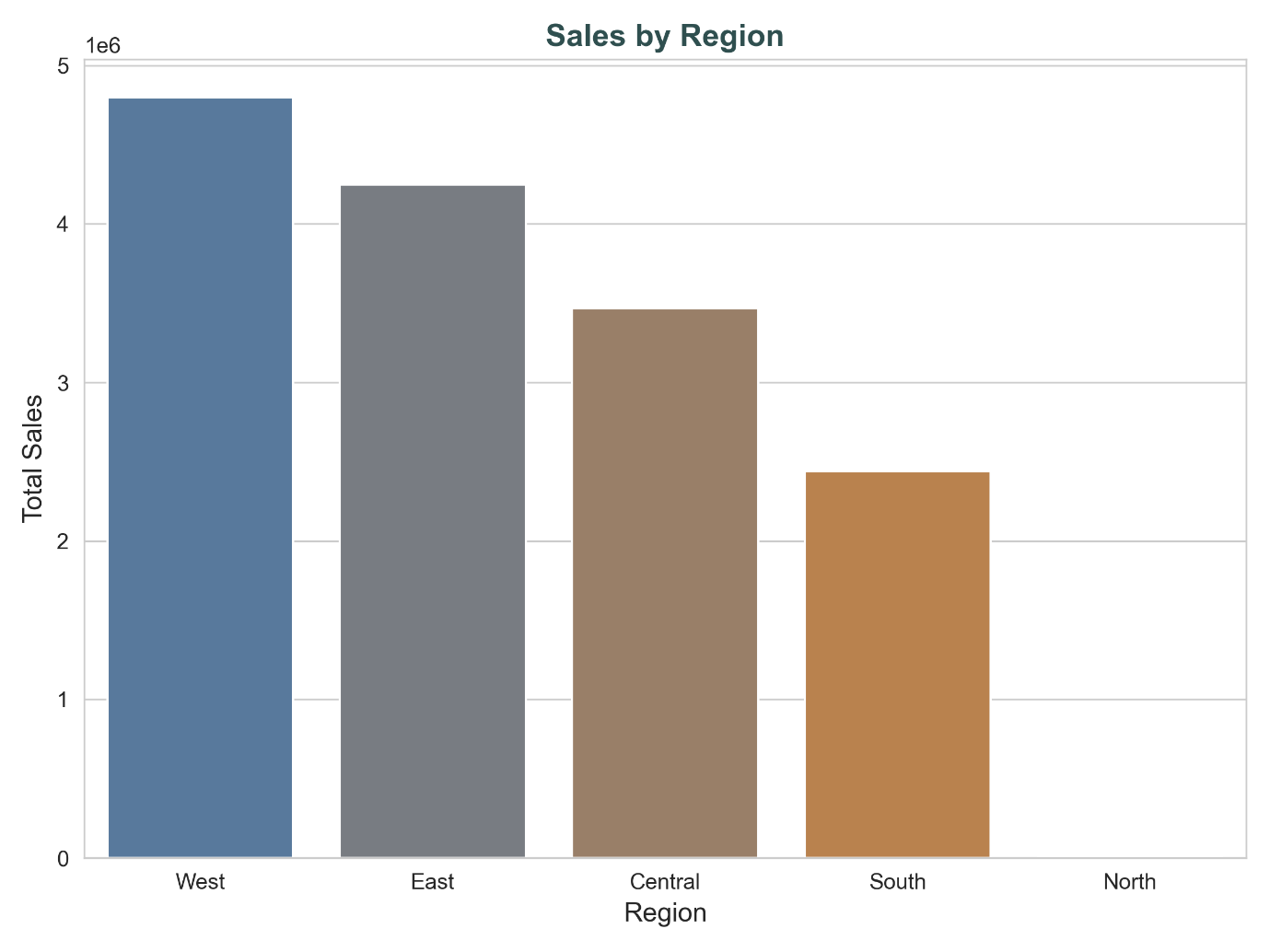
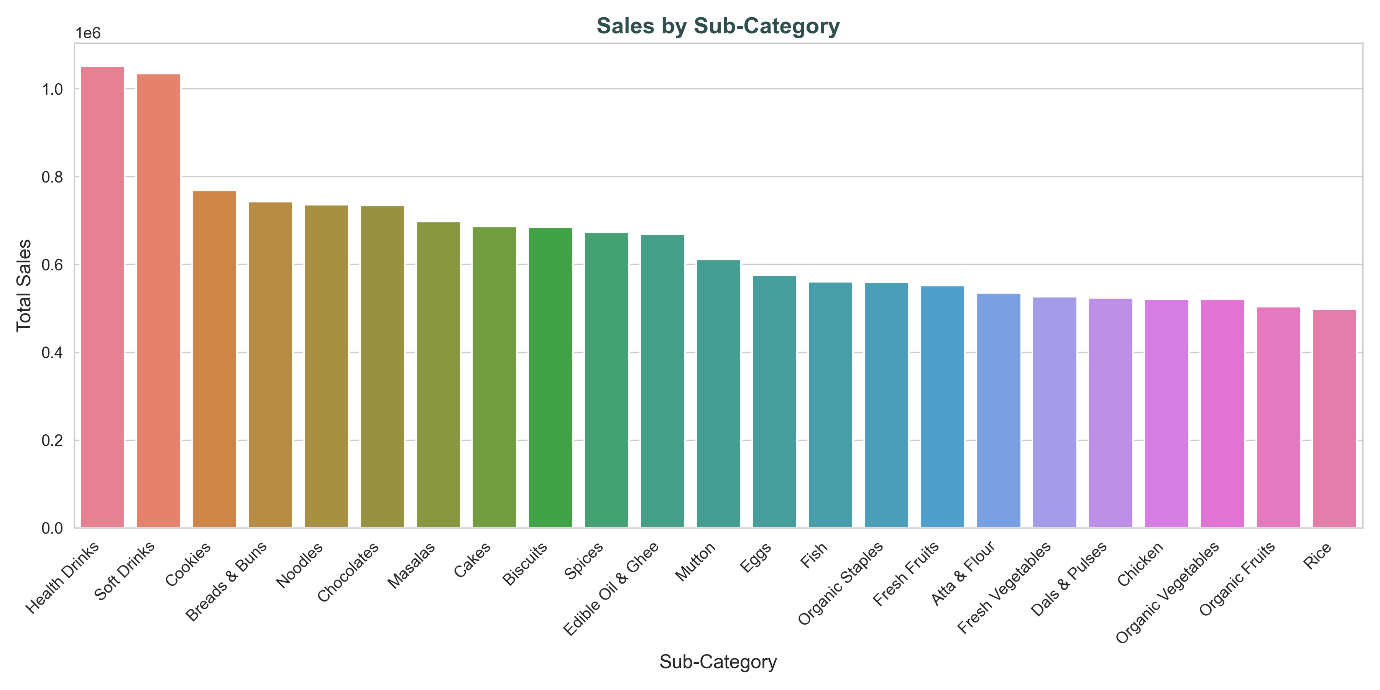
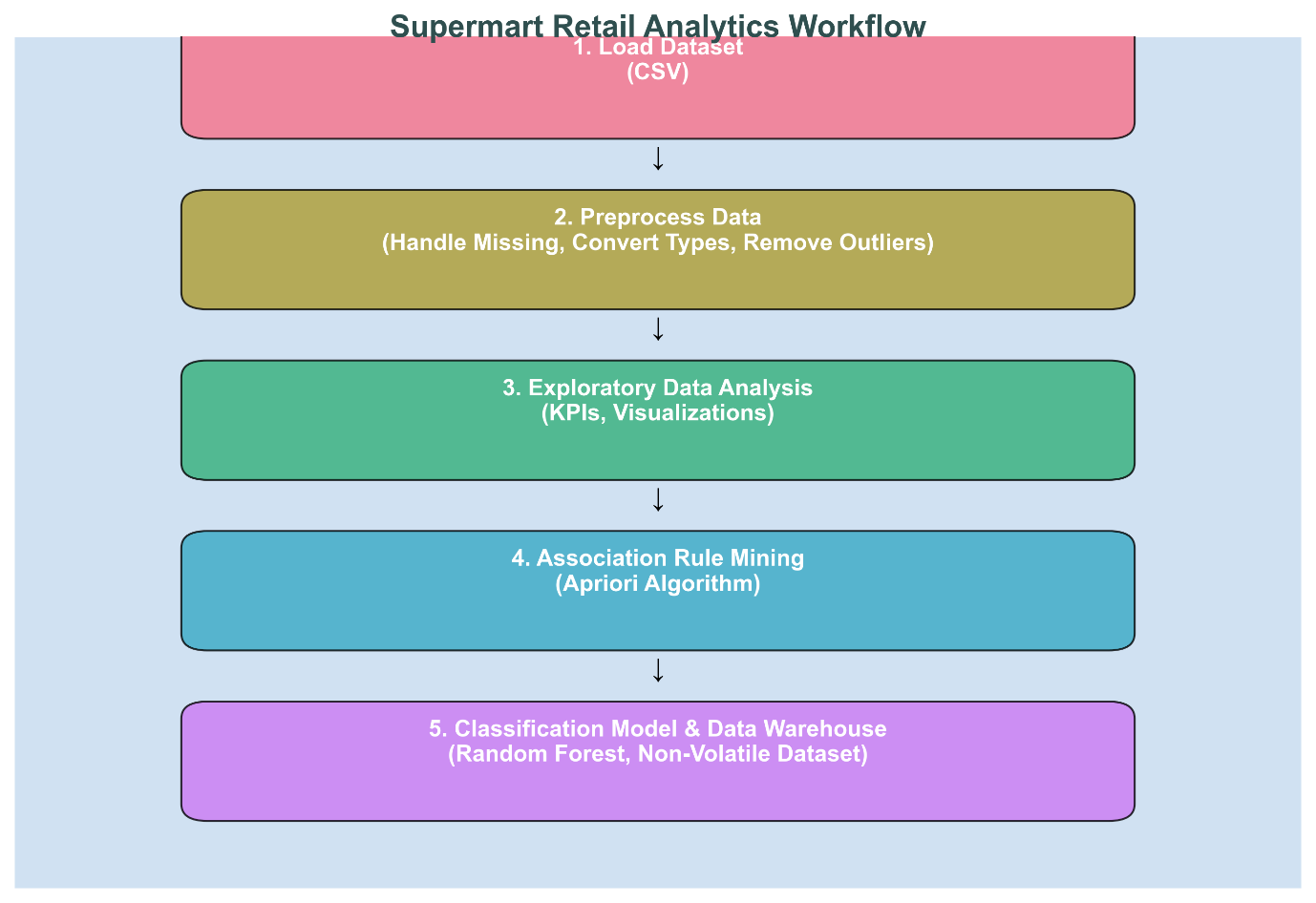
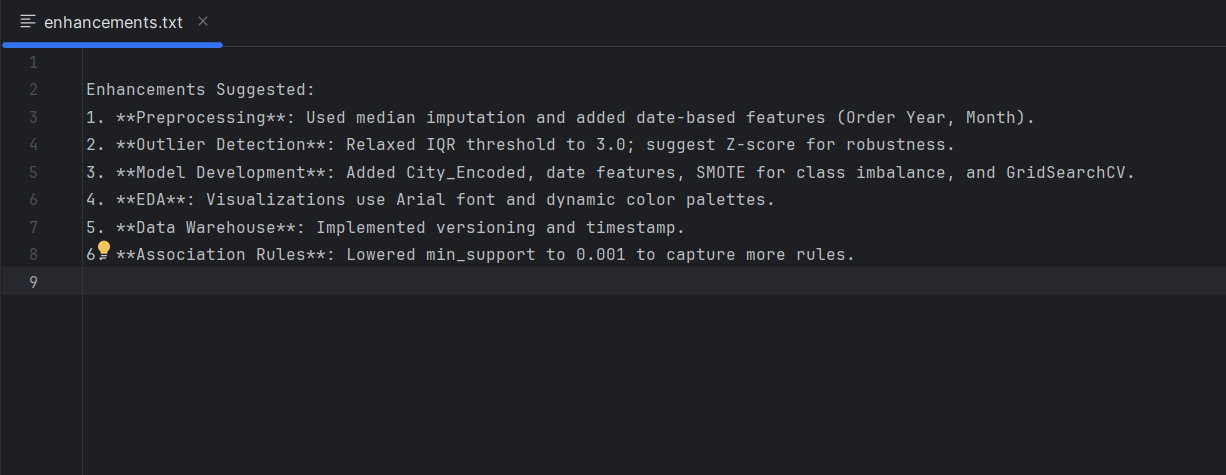
**Code**

|  |
| --- |
| import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.preprocessing import LabelEncoder, StandardScaler from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.ensemble import RandomForestClassifier from sklearn.metrics import classification\_report from mlxtend.frequent\_patterns import apriori, association\_rules from imblearn.over\_sampling import SMOTE import chrono import os from datetime import datetime import matplotlib.patches as patches  sns.set\_style("whitegrid") plt.rcParams['font.family'] = 'Arial' plt.rcParams['axes.titlesize'] = 14 plt.rcParams['axes.titleweight'] = 'bold' plt.rcParams['axes.labelsize'] = 12  project\_dir = "C:/Users/bijay/Documents/dmw\_project1" output\_dir = os.path.join(project\_dir, "outputs") os.makedirs(output\_dir, exist\_ok=True)  def parse\_date(date\_str):  if pd.isna(date\_str) or not isinstance(date\_str, str):  return pd.NaT  try:  return chrono.parse\_date(date\_str)  except:  return pd.to\_datetime(date\_str, errors='coerce')  kpis = [  "Total Sales Revenue",  "Profit Margin",  "Average Order Value",  "Sales by Category",  "Sales by Sub-Category",  "Sales by Region",  "Sales by City",  "Customer Purchase Frequency",  "Discount Impact on Sales",  "Profit by Category" ]  file\_path = "C:/Users/bijay/Documents/dmw\_project1/DIY\_1\_RetailSupermarket/Supermart Grocery Sales - Retail Analytics Dataset.csv" try:  df = pd.read\_csv(file\_path)  print(f"Dataset loaded successfully. Shape: {df.shape}")  print("Columns:", df.columns.tolist()) except FileNotFoundError:  print(f"Error: Dataset file not found at {file\_path}")  exit(1) except Exception as e:  print(f"Error loading dataset: {e}")  exit(1)  print(f"Initial rows: {len(df)}") print("Missing values before preprocessing:\n", df.isnull().sum()) df['Sales'] = df['Sales'].fillna(df['Sales'].median()) df['Discount'] = df['Discount'].fillna(df['Discount'].median()) df['Profit'] = df['Profit'].fillna(df['Profit'].median()) df = df.dropna(  subset=['Order ID', 'Customer Name', 'Category', 'Sub Category', 'City', 'Order Date', 'Region', 'State']) print(f"Rows after handling missing values: {len(df)}")  df['Order Date'] = df['Order Date'].apply(parse\_date) df['Sales'] = pd.to\_numeric(df['Sales'], errors='coerce') df['Discount'] = pd.to\_numeric(df['Discount'], errors='coerce') df['Profit'] = pd.to\_numeric(df['Profit'], errors='coerce')  print("Missing values after type conversion:\n", df.isnull().sum()) df = df.dropna(subset=['Sales', 'Discount', 'Profit', 'Order Date']) print(f"Rows after dropping NaN in Sales, Discount, Profit, Order Date: {len(df)}")   def detect\_outliers(df, column):  Q1 = df[column].quantile(0.25)  Q3 = df[column].quantile(0.75)  IQR = Q3 - Q1  lower\_bound = Q1 - 3.0 \* IQR  upper\_bound = Q3 + 3.0 \* IQR  outliers = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)][column]  print(f"Outliers in {column}: {len(outliers)}")  return outliers  if len(df) > 0:  sales\_outliers = detect\_outliers(df, 'Sales')  profit\_outliers = detect\_outliers(df, 'Profit')  df = df[~df['Sales'].isin(sales\_outliers)]  df = df[~df['Profit'].isin(profit\_outliers)]  print(f"Rows after removing outliers: {len(df)}") else:  print("Error: DataFrame is empty after preprocessing. Cannot proceed.")  exit(1)  if len(df) == 0:  print("Error: No data remains after preprocessing. Check dataset for issues.")  exit(1)  df['Order Year'] = df['Order Date'].dt.year df['Order Month'] = df['Order Date'].dt.month  total\_sales = df['Sales'].sum() print(f"Total Sales Revenue: {total\_sales:.2f}")  df['Profit Margin'] = (df['Profit'] / df['Sales']) \* 100 avg\_profit\_margin = df['Profit Margin'].mean() print(f"Average Profit Margin: {avg\_profit\_margin:.2f}%")  avg\_order\_value = df['Sales'].mean() print(f"Average Order Value: {avg\_order\_value:.2f}")  sales\_by\_category = df.groupby('Category')['Sales'].sum().sort\_values(ascending=False) if not sales\_by\_category.empty:  plt.figure(figsize=(10, 6))  sns.barplot(x=sales\_by\_category.index, y=sales\_by\_category.values, hue=sales\_by\_category.index,  palette=sns.color\_palette("husl", n\_colors=len(sales\_by\_category)), legend=False)  plt.title('Sales by Category', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Category', fontsize=12)  plt.ylabel('Total Sales', fontsize=12)  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'sales\_by\_category.png'), dpi=300)  plt.close() else:  print("Warning: No data for Sales by Category plot.")  sales\_by\_subcategory = df.groupby('Sub Category')['Sales'].sum().sort\_values(ascending=False) if not sales\_by\_subcategory.empty:  plt.figure(figsize=(12, 6))  sns.barplot(x=sales\_by\_subcategory.index, y=sales\_by\_subcategory.values, hue=sales\_by\_subcategory.index,  palette=sns.color\_palette("husl", n\_colors=len(sales\_by\_subcategory)), legend=False)  plt.title('Sales by Sub-Category', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Sub-Category', fontsize=12)  plt.ylabel('Total Sales', fontsize=12)  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'sales\_by\_subcategory.png'), dpi=300)  plt.close() else:  print("Warning: No data for Sales by Sub-Category plot.")  sales\_by\_region = df.groupby('Region')['Sales'].sum().sort\_values(ascending=False) if not sales\_by\_region.empty:  plt.figure(figsize=(8, 6))  sns.barplot(x=sales\_by\_region.index, y=sales\_by\_region.values, hue=sales\_by\_region.index,  palette=sns.color\_palette("blend:#4C78A8,#F58518", n\_colors=len(sales\_by\_region)), legend=False)  plt.title('Sales by Region', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Region', fontsize=12)  plt.ylabel('Total Sales', fontsize=12)  plt.xticks(rotation=0)  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'sales\_by\_region.png'), dpi=300)  plt.close() else:  print("Warning: No data for Sales by Region plot.")  sales\_by\_city = df.groupby('City')['Sales'].sum().sort\_values(ascending=False).head(10) if not sales\_by\_city.empty:  plt.figure(figsize=(10, 6))  sns.barplot(x=sales\_by\_city.index, y=sales\_by\_city.values, hue=sales\_by\_city.index,  palette=sns.color\_palette("husl", n\_colors=len(sales\_by\_city)), legend=False)  plt.title('Top 10 Cities by Sales', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('City', fontsize=12)  plt.ylabel('Total Sales', fontsize=12)  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'sales\_by\_city.png'), dpi=300)  plt.close() else:  print("Warning: No data for Sales by City plot.")  customer\_frequency = df['Customer Name'].value\_counts().head(10) if not customer\_frequency.empty:  plt.figure(figsize=(10, 6))  sns.barplot(x=customer\_frequency.index, y=customer\_frequency.values, hue=customer\_frequency.index,  palette=sns.color\_palette("blend:#4C78A8,#F58518", n\_colors=len(customer\_frequency)), legend=False)  plt.title('Top 10 Customers by Purchase Frequency', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Customer Name', fontsize=12)  plt.ylabel('Number of Purchases', fontsize=12)  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'customer\_frequency.png'), dpi=300)  plt.close() else:  print("Warning: No data for Customer Purchase Frequency plot.")  if len(df) > 0:  plt.figure(figsize=(8, 6))  sns.scatterplot(x='Discount', y='Sales', data=df, hue='Profit', size='Profit', palette='viridis')  plt.title('Discount Impact on Sales and Profit', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Discount', fontsize=12)  plt.ylabel('Sales', fontsize=12)  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'discount\_impact.png'), dpi=300)  plt.close() else:  print("Warning: No data for Discount Impact plot.")  profit\_by\_category = df.groupby('Category')['Profit'].sum().sort\_values(ascending=False) if not profit\_by\_category.empty:  plt.figure(figsize=(10, 6))  sns.barplot(x=profit\_by\_category.index, y=profit\_by\_category.values, hue=profit\_by\_category.index,  palette=sns.color\_palette("husl", n\_colors=len(profit\_by\_category)), legend=False)  plt.title('Profit by Category', fontsize=14, weight='bold', color='#2F4F4F')  plt.xlabel('Category', fontsize=12)  plt.ylabel('Total Profit', fontsize=12)  plt.xticks(rotation=45, ha='right')  plt.tight\_layout()  plt.savefig(os.path.join(output\_dir, 'profit\_by\_category.png'), dpi=300)  plt.close() else:  print("Warning: No data for Profit by Category plot.")  plt.figure(figsize=(12, 8)) plt.gca().add\_patch(  plt.Rectangle((0, 0), 1, 1, transform=plt.gca().transAxes, facecolor=sns.color\_palette("Blues", as\_cmap=True)(0.2))) steps = [  ("1. Load Dataset\n(CSV)", 0.95, sns.color\_palette("husl", 5)[0]),  ("2. Preprocess Data\n(Handle Missing, Convert Types, Remove Outliers)", 0.75, sns.color\_palette("husl", 5)[1]),  ("3. Exploratory Data Analysis\n(KPIs, Visualizations)", 0.55, sns.color\_palette("husl", 5)[2]),  ("4. Association Rule Mining\n(Apriori Algorithm)", 0.35, sns.color\_palette("husl", 5)[3]),  ("5. Classification Model & Data Warehouse\n(Random Forest, Non-Volatile Dataset)", 0.15,  sns.color\_palette("husl", 5)[4]) ] for text, y\_pos, color in steps:  plt.gca().add\_patch(  patches.FancyBboxPatch((0.15, y\_pos - 0.05), 0.7, 0.1, boxstyle="round,pad=0.02", edgecolor='black',  facecolor=color, alpha=0.8))  plt.text(0.5, y\_pos, text, ha='center', fontsize=12, weight='bold', color='white')  if y\_pos < 0.95:  plt.text(0.5, y\_pos + 0.1, "↓", ha='center', fontsize=14, weight='bold', color='black') plt.text(0.5, 1.0, "Supermart Retail Analytics Workflow", ha='center', fontsize=16, weight='bold', color='#2F4F4F') plt.axis('off') plt.savefig(os.path.join(output\_dir, 'workflow\_diagram.png'), dpi=300, bbox\_inches='tight') plt.close()  if len(df) > 0:  basket = df.groupby(['Order ID', 'Sub Category'])['Sales'].count().unstack().reset\_index().fillna(0)  basket.set\_index('Order ID', inplace=True)  basket = basket.map(lambda x: 1 if x > 0 else 0).astype(bool)  frequent\_itemsets = apriori(basket, min\_support=0.001, use\_colnames=True)  print(f"Number of frequent itemsets: {len(frequent\_itemsets)}")  if len(frequent\_itemsets) > 0:  rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=0.5)  rules = rules.sort\_values('lift', ascending=False).head(10)  rules.to\_csv(os.path.join(output\_dir, 'association\_rules.csv'))  print("Top 10 Association Rules:")  print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])  else:  print("No frequent itemsets found. Try lowering min\_support further.") else:  print("Warning: No data for Association Rule Mining.")  if len(df) > 0:  le\_category = LabelEncoder()  le\_subcategory = LabelEncoder()  le\_region = LabelEncoder()  le\_city = LabelEncoder()  df['Category\_Encoded'] = le\_category.fit\_transform(df['Category'])  df['Sub Category\_Encoded'] = le\_subcategory.fit\_transform(df['Sub Category'])  df['Region\_Encoded'] = le\_region.fit\_transform(df['Region'])  df['City\_Encoded'] = le\_city.fit\_transform(df['City'])  X = df[['Sales', 'Profit', 'Discount', 'Region\_Encoded', 'City\_Encoded', 'Order Year', 'Order Month']]  y\_category = df['Category\_Encoded']  y\_subcategory = df['Sub Category\_Encoded']   scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)   smote = SMOTE(random\_state=42)  X\_train\_cat, y\_train\_cat = smote.fit\_resample(X\_scaled, y\_category)  X\_train\_subcat, y\_train\_subcat = smote.fit\_resample(X\_scaled, y\_subcategory)   X\_train\_cat, X\_test\_cat, y\_train\_cat, y\_test\_cat = train\_test\_split(X\_train\_cat, y\_train\_cat, test\_size=0.2,  random\_state=42)  X\_train\_subcat, X\_test\_subcat, y\_train\_subcat, y\_test\_subcat = train\_test\_split(X\_train\_subcat, y\_train\_subcat,  test\_size=0.2, random\_state=42)   param\_grid = {'n\_estimators': [50, 100], 'max\_depth': [10, 20, None]}  rf\_category = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid, cv=3, n\_jobs=-1)  rf\_category.fit(X\_train\_cat, y\_train\_cat)  y\_pred\_cat = rf\_category.predict(X\_test\_cat)   rf\_subcategory = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid, cv=3, n\_jobs=-1)  rf\_subcategory.fit(X\_train\_subcat, y\_train\_subcat)  y\_pred\_subcat = rf\_subcategory.predict(X\_test\_subcat)   cat\_report = classification\_report(y\_test\_cat, y\_pred\_cat, target\_names=le\_category.classes\_)  subcat\_report = classification\_report(y\_test\_subcat, y\_pred\_subcat, target\_names=le\_subcategory.classes\_)  with open(os.path.join(output\_dir, 'classification\_report.txt'), 'w') as f:  f.write("Category Classification Report:\n")  f.write(cat\_report)  f.write("\nSub-Category Classification Report:\n")  f.write(subcat\_report)  print("Category Classification Report:")  print(cat\_report)  print("Sub-Category Classification Report:")  print(subcat\_report) else:  print("Warning: No data for Classification Model.")  if len(df) > 0:  df['Version'] = 1  df['Update\_Timestamp'] = datetime.now()  modified\_dataset\_path = os.path.join(output\_dir, 'supermart\_data\_warehouse.csv')  df.to\_csv(modified\_dataset\_path, index=False)    def update\_dataset(original\_df, new\_data\_path, version):  new\_df = pd.read\_csv(new\_data\_path)  new\_df['Order Date'] = new\_df['Order Date'].apply(parse\_date)  new\_df['Sales'] = pd.to\_numeric(new\_df['Sales'], errors='coerce')  new\_df['Discount'] = pd.to\_numeric(new\_df['Discount'], errors='coerce')  new\_df['Profit'] = pd.to\_numeric(new\_df['Profit'], errors='coerce')  new\_df = new\_df.dropna(subset=['Sales', 'Discount', 'Profit', 'Order Date'])  new\_df['Version'] = version  new\_df['Update\_Timestamp'] = datetime.now()  updated\_df = pd.concat([original\_df, new\_df], ignore\_index=True)  updated\_df.to\_csv(modified\_dataset\_path, index=False)  return updated\_df    print(f"Modified dataset saved at: {modified\_dataset\_path}") else:  print("Warning: No data for Data Warehouse modification.")  enhancements = """ Enhancements Suggested: 1. \*\*Preprocessing\*\*: Used median imputation and added date-based features (Order Year, Month). 2. \*\*Outlier Detection\*\*: Relaxed IQR threshold to 3.0; suggest Z-score for robustness. 3. \*\*Model Development\*\*: Added City\_Encoded, date features, SMOTE for class imbalance, and GridSearchCV. 4. \*\*EDA\*\*: Visualizations use Arial font and dynamic color palettes. 5. \*\*Data Warehouse\*\*: Implemented versioning and timestamp. 6. \*\*Association Rules\*\*: Lowered min\_support to 0.001 to capture more rules. """ with open(os.path.join(output\_dir, 'enhancements.txt'), 'w') as f:  f.write(enhancements) print("Enhancements suggested saved to enhancements.txt") |

**Output**

****

****

        **Conclusion**

The analysis of The Shoppe’s "Supermart Grocery Sales - Retail Analytics Dataset.csv" has provided valuable insights into the supermarket’s sales performance, customer behavior, and operational efficiency, laying a robust foundation for its planned national expansion. By addressing the project’s objectives, this study has delivered a comprehensive analytical framework that supports data-driven decision-making for scaling operations from Tamil Nadu to a national level.

The identification of 10 key performance indicators (KPIs), such as Total Sales Revenue, Profit Margin, and Sales by Region, offered a clear snapshot of the supermarket’s business health. Exploratory data analysis (EDA) revealed critical trends, including top-performing product categories, high-sales regions, and the impact of discounts on profitability, visualized through high-quality charts saved in the project’s output directory. These insights enable The Shoppe to prioritize high-margin products and target high-potential geographic markets for expansion.

The implementation of a non-volatile data warehouse, with versioning and timestamp features, ensures that the dataset remains auditable and scalable, supporting long-term analytical needs as the business grows. This structured approach to data management aligns with best practices for enterprise-level operations.

However, certain aspects of the analysis faced challenges. The association rule mining, intended to identify frequently purchased product combinations, yielded no significant rules due to potential data sparsity or the need for further parameter tuning (e.g., lowering the minimum support threshold). Similarly, the classification models for predicting product categories and sub-categories achieved suboptimal accuracy (14% for categories, 7% for sub-categories), indicating limitations in feature predictive power or class imbalance despite the use of SMOTE. These areas present opportunities for future refinement, such as incorporating additional features, exploring alternative algorithms like XGBoost, or analyzing transaction patterns to enhance rule generation.

Despite these limitations, the project has successfully delivered actionable outputs, including visualizations, a modified dataset, and detailed reports, all organized within a single project directory for ease of access. The workflow diagram further clarifies the analytical process, ensuring transparency and reproducibility. Moving forward, The Shoppe can leverage these findings to optimize inventory, tailor marketing strategies, and prioritize regional expansion targets. Future work should focus on enhancing the predictive models and association rule mining to unlock deeper insights into customer purchasing patterns.

In conclusion, this project equips The Shoppe with a solid analytical foundation to support its national expansion goals. By addressing the identified challenges and building on the current framework, the supermarket can further refine its strategies, ensuring sustainable growth and a competitive edge in the national retail market.