Topic: Retail Sales Analysis

Retail Sales Analysis Project = The Retail Sales Analysis Project is designed to extract valuable insights and make data-driven predictions based on transaction data from a retail store. By analysing various aspects of sales, customer behaviour, and product performance, this project provides a comprehensive view of the factors influencing retail success. The objective is to help businesses make informed decisions about inventory, customer engagement, and strategic growth.

The dataset comprises numerous attributes detailing individual transactions, including Invoice ID, Product line, Quantity, Total, Payment type, Customer type, and Rating. This combination of categorical and numerical data allows for a detailed exploration of both transactional and customer-oriented patterns.

TEAM FORMATION

Team leader : ABHISHEK KUMAR , **ID**: [redacted]

All Team Members -

Sr. No.	Student ID	Roll No.	Student Name
1	[redacted]	[redacted]	ABHISHEK KUMAR
2	[redacted]	[redacted]	ABHISHEK MAHESHWARI
3	[redacted]	[redacted]	ANUBHAV GANGWAR
4	[redacted]	[redacted]	RIDHIMA MAHEBHWARI
5	[redacted]	[redacted]	VANSH GUPTA
6	[redacted]	[redacted]	SAMINA NOOREEN
7	[redacted]	[redacted]	SONI
8	[redacted]	[redacted]	MOHD SHEEBAN KHAN
9	[redacted]	[redacted]	ANKIT KUMAR
10	[redacted]	[redacted]	ANKIT KUMAR
11	[redacted]	[redacted]	SANJAY MISHRA
12	[redacted]	[redacted]	ARPIT GUPTA
13	[redacted]	[redacted]	ARYAN SRIVASTAVA
14	[redacted]	[redacted]	VAISHNAVI GUPTA

Table of Contents:

1. *Introduction*

Overview of Project Goals

Analyse retail sales data to uncover trends that inform business decisions on product offerings, customer segmentation, and sales strategies.

Business Problem Statement

Retailers need insights into customer preferences and sales patterns to manage inventory effectively and maximize profits.

Objectives of Analysis

- Identify sales trends across time and locations.
- Understand customer preferences and behaviour.
- Evaluate product line performance to guide inventory management.

Summary of Analysis Approach

We perform data cleaning, EDA, and visualization to explore key metrics, followed by a predictive model for customer ratings.

Tools and Libraries Used

We use Python and libraries like `pandas` for data manipulation, `seaborn` and `matplotlib` for visualizations, and `sklearn` for machine learning. The following code shows the initial library imports:

```
import numpy as np import
pandas as pd import
matplotlib.pyplot as plt import
seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
import datetime as dt import warnings

# Suppress all warnings
warnings.filterwarnings("ignore")
```

2. *Data Collection and Preparation*

Data Source and Import

The dataset, sourced from retail transactions, is imported using `pandas`. This initial load prepares it for inspection and processing.

```
data = pd.read_csv("retailsales.csv")
```

Initial Data Exploration

We explore the data structure, data types, and basic stats to understand its contents and identify potential issues.

```
data.info() data.describe()
```

Handling Missing Values and Data Cleaning

Missing values are checked and addressed to ensure accuracy in analysis, using imputation or deletion based on context.

```
data.isnull().sum()
```

Dropping Irrelevant Columns

Non-essential columns are removed to focus on relevant variables, reducing noise in the analysis and improving model efficiency.

```
object_cols =
data.select_dtypes(include=['object']).columns data
= data.drop(object_cols, axis=1)
```

Handling Outliers

Outliers are detected using the Interquartile Range (IQR) method and removed for columns where extreme values could distort insights.

```
q1 = data.quantile(0.25) q3
= data.quantile(0.75) IQR =
q3 - q1
upper_range = q3 + 1.5 * IQR
outlier_upper_tax = np.where(data['Tax 5%'] >
upper_range['Tax 5%'])
data.drop(outlier_upper_tax[0], inplace=True)
```

3. *Data Exploration*

Overview of Dataset Structure

We examine the dataset's structure, including data types and basic info, to understand what's available for analysis and identify any data type mismatches.

```
data.info()
```

Summary Statistics

Basic statistics provide insights into distributions, averages, and variances for numerical columns, giving a quick snapshot of the dataset.

```
data.describe()
```

Exploring Categorical Columns

Categorical columns are analyzed to see unique values and frequency counts, helping us understand non-numeric patterns in the data.

```
object_cols =
data.select_dtypes(include=['object']).columns
data[object_cols].nunique()
```

Dropping Non-Numeric Columns

Non-numeric columns are removed when focusing on numeric analysis to simplify the data and modeling process.

```
df_filtered = data.drop(object_cols, axis=1)
```

Detailed Analysis of Object Columns

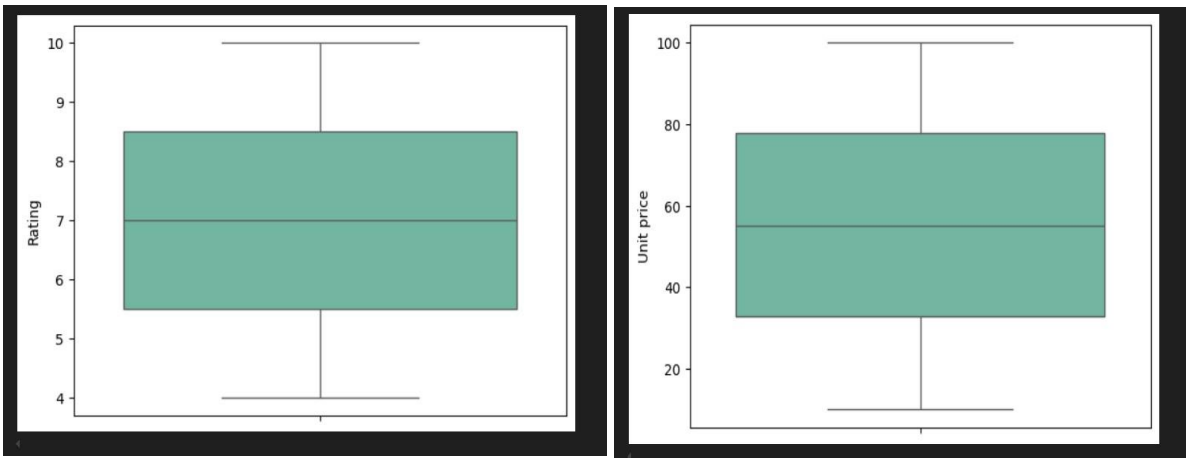
We take a closer look at object columns, exploring common values and patterns that may impact customer or product-based insights.

```
data[object_cols].head()
```

4. *Outlier Detection and Treatment*

Boxplot Visualization for Outlier Detection

Boxplots are used to visually detect outliers in key numerical columns, helping to identify extreme values that could skew analysis.



Calculation of Quartiles and Interquartile Range (IQR)

We calculate the first and third quartiles (Q1, Q3) and the IQR to identify potential outliers based on statistical thresholds.

```
q1 = df_filtered.quantile(0.25) q3
= df_filtered.quantile(0.75) IQR =
q3 - q1
```

Outlier Treatment for Key Columns

Outliers are removed by applying the IQR method, where values beyond the upper or lower bounds are considered outliers and excluded.

```
upper_range = q3 + 1.5 * IQR
outlier_upper_tax = np.where(df_filtered['Tax 5%'] >
upper_range['Tax 5%'])
data.drop(outlier_upper_tax[0], inplace=True)
```

Re-evaluation of Outliers after Treatment

After removing the outliers, the data is re-checked to ensure that outliers are no longer present and to validate the integrity of the dataset.

```
df_filtered = data.drop(object_cols, axis=1)
```

Discussion on Business Impact of Outliers

Outliers may distort sales predictions and customer insights. Removing them ensures more accurate analysis, especially when predicting sales or customer behavior.

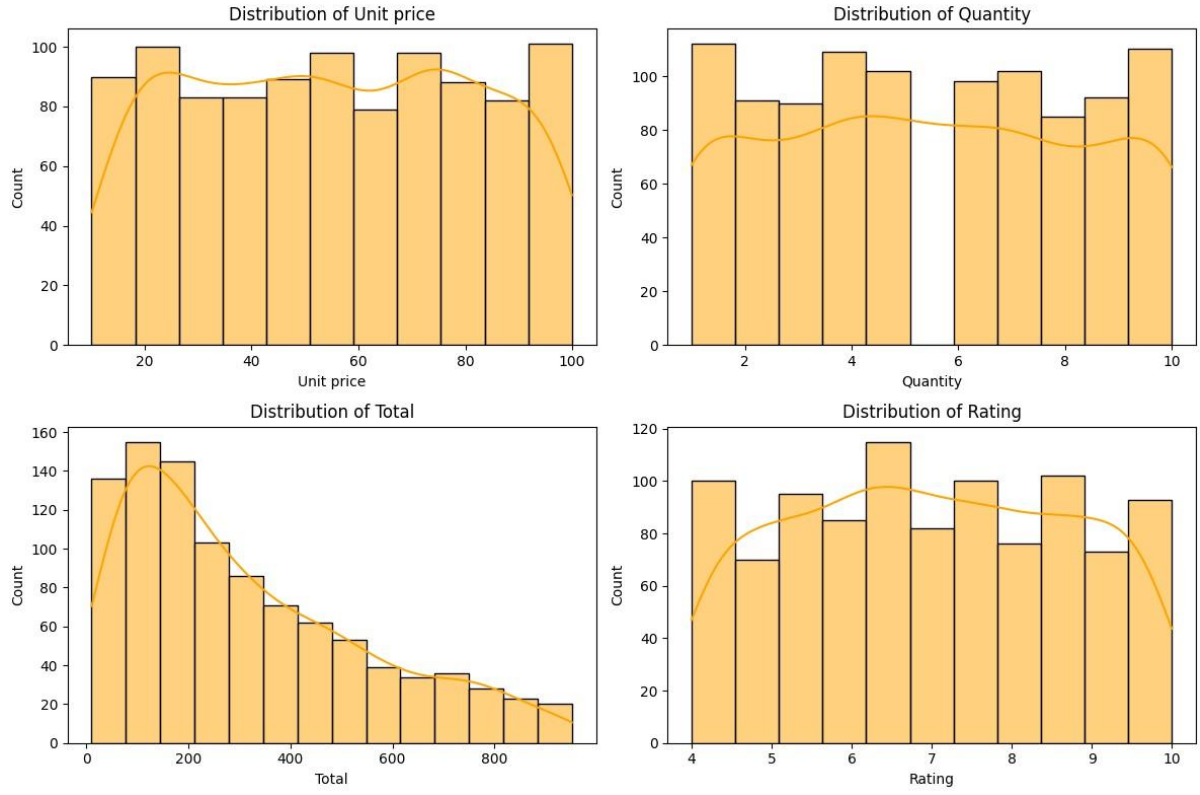
5. *Exploratory Data Analysis (EDA)*

Univariate Analysis

Distribution of Numerical Features (Histograms, KDEs)

Histograms and Kernel Density Estimates (KDEs) are used to visualize the distribution of numerical features. This helps identify skewness, normality, or any unusual patterns in sales or quantities that might need further investigation.

```
sns.histplot(data['Unit price'], kde=True,
color='orange')
plt.title('Unit Price Distribution') plt.show()
```



Reason: Identifying skewness or non-normal distributions helps decide whether transformation or feature engineering is required.

Count and Frequency Analysis of Categorical Features

Bar plots or count plots are used to understand the distribution of categorical variables like product lines or customer types. This helps identify dominant categories or imbalances in data.

```
sns.countplot(data=data, x='Product line',
palette='viridis')
plt.title('Count of Transactions by Product Line')
plt.show()
```

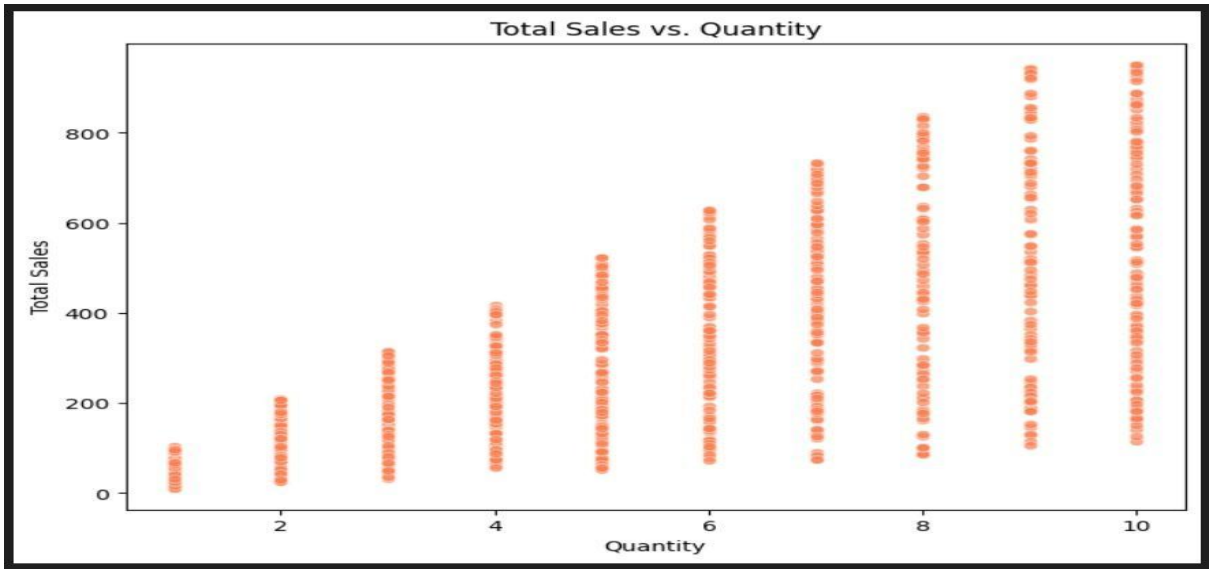
Reason: Analyzing frequencies allows us to determine the most common categories and any potential biases in the dataset.

Bivariate Analysis

Scatter Plots for Quantity vs. Total Sales

Scatter plots are used to investigate the relationship between the number of items sold (Quantity) and the total sales value. This visualizes whether higher quantities lead to increased sales.

```
sns.scatterplot(data=data, x='Quantity', y='Total',
color='coral')
plt.title('Quantity vs. Total Sales')
plt.xlabel('Quantity')
plt.ylabel('Total Sales') plt.show()
```

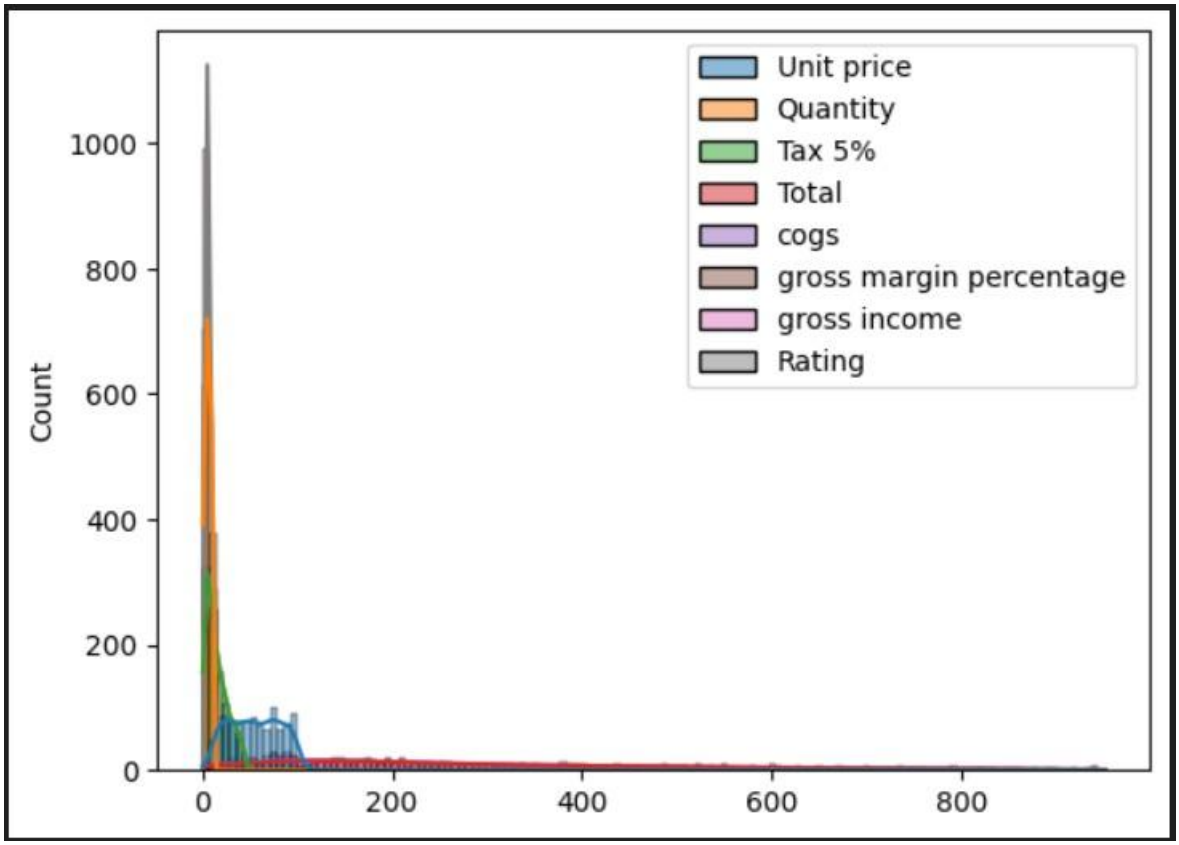


Reason: Helps assess whether there's a linear or non-linear relationship between sales and quantity, which is crucial for pricing and promotional strategies.

Correlation Matrix and Heatmap

A heatmap of the correlation matrix helps identify relationships between numerical variables (e.g., quantity, price, total sales, ratings). It reveals key dependencies that might inform predictive models or business decisions.

```
corr = data[['Quantity', 'Unit price', 'Total',
'Rating']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', vmin=-
1, vmax=1)
plt.title('Correlation Heatmap') plt.show()
```



Reason: Identifying correlations helps in understanding variable relationships, aiding in feature selection for predictive modeling.

Multivariate Analysis

Trends by Month, Product Line, and City

Multivariate analysis explores trends across multiple dimensions. Here, we look at sales by month, product line, and city, which provides a comprehensive view of seasonal or geographic patterns.

```
sales_by_month = data.groupby(['Month',
'Month_Full'])['Total'].sum().reset_index()
sales_by_month.plot(kind='bar', x='Month_Full',
y='Total', title='Sales by Month') plt.show()
```

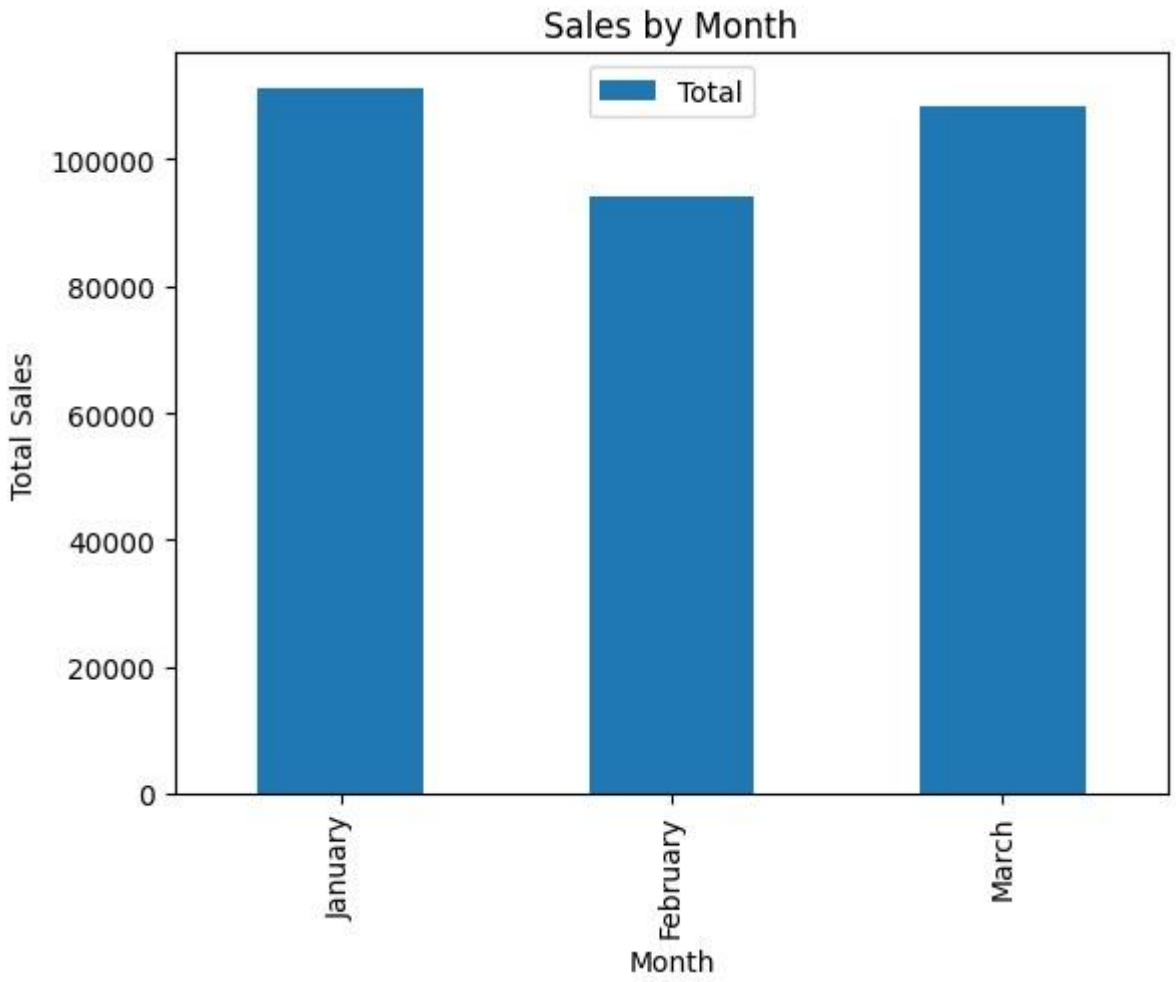
Reason: This analysis helps identify trends, peak sales periods, and regional preferences, which are critical for inventory and marketing strategies.

6. *Visualization of Key Metrics*

Sales Performance by Month

A bar chart shows the total sales per month, which helps identify seasonal sales trends and peak months. This is valuable for forecasting and inventory planning.

```
sales_by_month.plot(kind='bar', x='Month_Full',
y='Total', title='Sales by Month', xlabel='Month',
ylabel='Total Sales') plt.show()
```



Reason: Identifying high and low sales months aids in optimizing promotional activities and inventory management during peak periods.

Quantity Sales by City and Branch

A bar chart visualizing the total quantity sold by city and branch highlights regional and branch-specific sales performance. This allows businesses to focus on high-performing areas and improve underperforming ones.

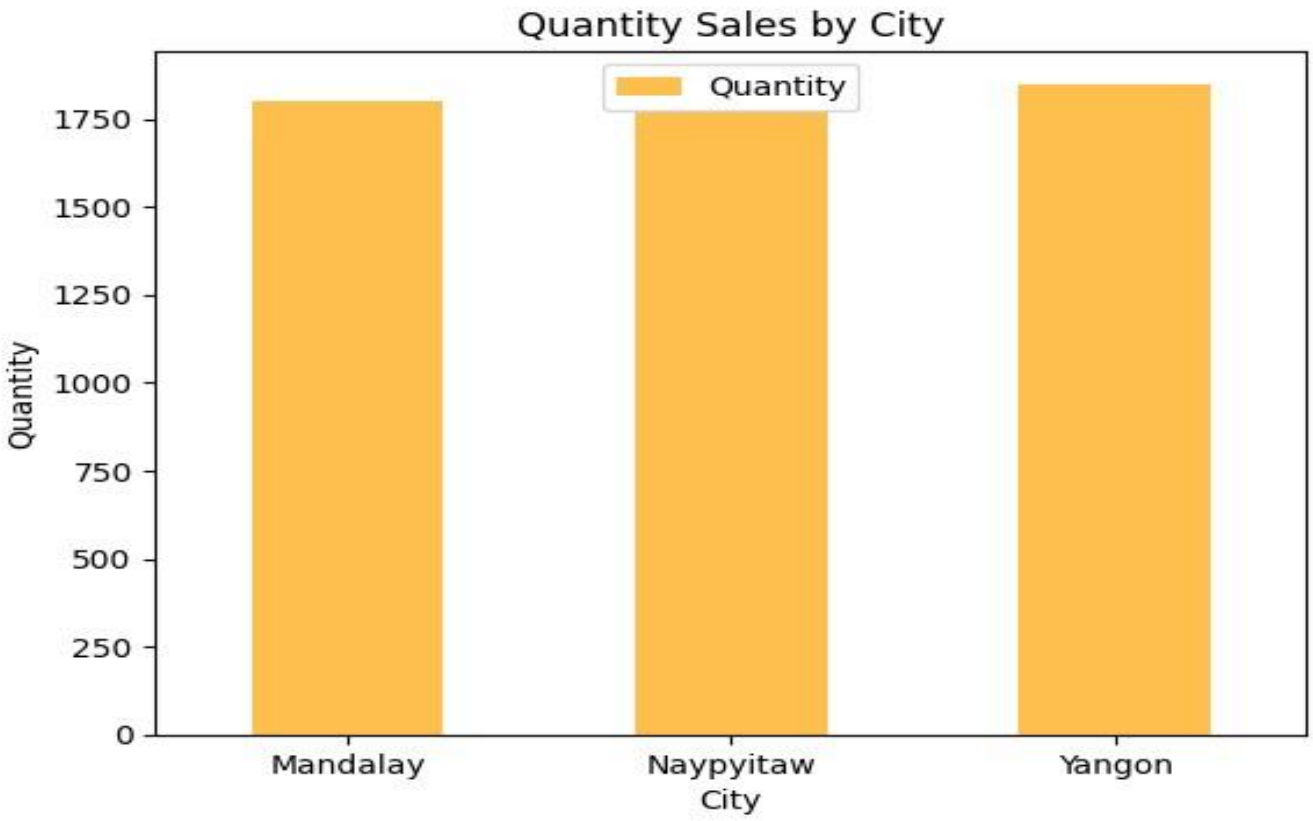
```
quantity_by_city.plot(kind='bar', x='City',
y='Quantity', title='Quantity Sales by City',
xlabel='City', ylabel='Quantity', alpha=0.7,
color='orange') plt.xticks(rotation=0) plt.show()
```

Reason: Identifies which cities and branches perform better, guiding localized marketing and distribution strategies.

Total Sales by City and Branch

This bar chart visualizes the total sales for each city and branch, helping understand which locations generate the most revenue and which ones may need strategic adjustments.

```
Total_by_city.plot(kind='bar', x='City', y='Total',
title='Total by City', xlabel='City', ylabel='Total',
alpha=0.7, color='yellow') plt.xticks(rotation=0)
plt.grid(True, linestyle='--', color='gray',
linewidth=0.5) plt.show()
```



Reason: This helps identify high-revenue cities and branches, supporting decisions on where to allocate resources for growth or improvement.

Hourly Analysis of Order Volume

A line plot of orders placed by hour of the day reveals peak shopping hours, which can be used to schedule staff, optimize promotions, or manage system load during high-traffic periods.

```
orders_placed_by_hour.plot(kind='line', x='Hour',
y='Invoice ID', title='Orders Placed by Hour',
xlabel='Hour', ylabel='Invoice ID', alpha=0.7,
```

```
color='orange') plt.xticks(range(10, 21))
plt.xticks(rotation=0) plt.show()
```

Reason: This helps businesses optimize operational strategies, ensuring resources are aligned with peak demand hours.

Product Line and Quantity Distribution (Pie Charts)

A pie chart shows the distribution of quantities sold across different product lines, providing insights into customer preferences and helping identify popular products.

```
plt.pie(quantity_by_product['Quantity'],
labels=quantity_by_product['Product line'],
autopct='%1.1f%%')
plt.title('Quantity Distribution by Product Line')
plt.show()
```

Reason: Understanding which product lines dominate sales helps businesses tailor marketing strategies and inventory planning towards popular categories.

7. Sales Trends Over Time

Sales by Month Analysis

A bar chart of monthly sales helps identify sales trends over time, revealing peak sales months and helping to plan future sales strategies, marketing, and stock management.

```
sales_by_month.plot(kind='bar', x='Month_Full',
y='Total', title='Sales by Month', xlabel='Month',
ylabel='Total Sales') plt.show()
```

Reason: This visualizes overall sales performance across months, identifying seasonality and guiding resource allocation.

Monthly Sales Visualization

A line chart shows the monthly sales trend, helping track long-term growth or decline in revenue, useful for forecasting and trend analysis.

```
daily_sales.plot()
plt.title('Daily Sales Over Time')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.xticks(rotation=45) plt.show()
```

Reason: This gives a time series view of sales performance, helping understand the business's growth trajectory and predicting future sales trends.

Sales Quantity by Product Line

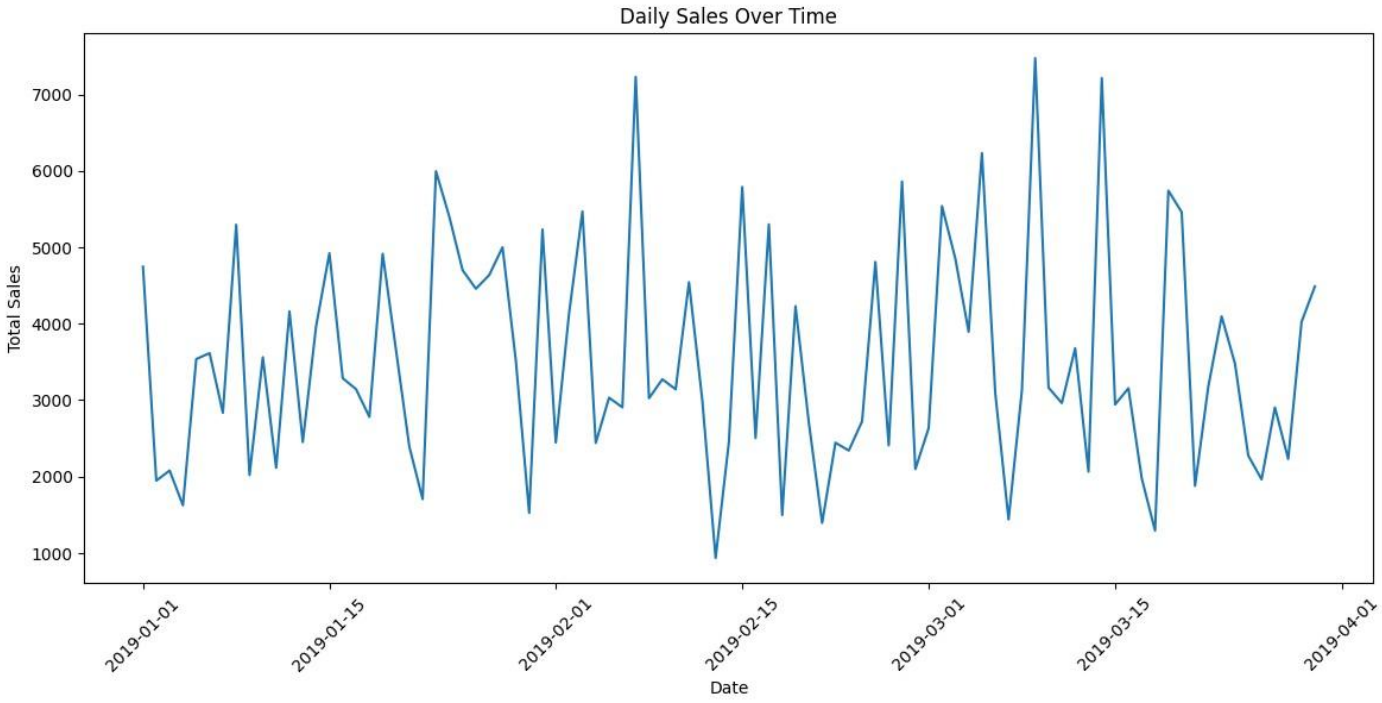
Bar plots visualize sales quantity by product line, indicating which products perform best, guiding inventory and marketing strategies.

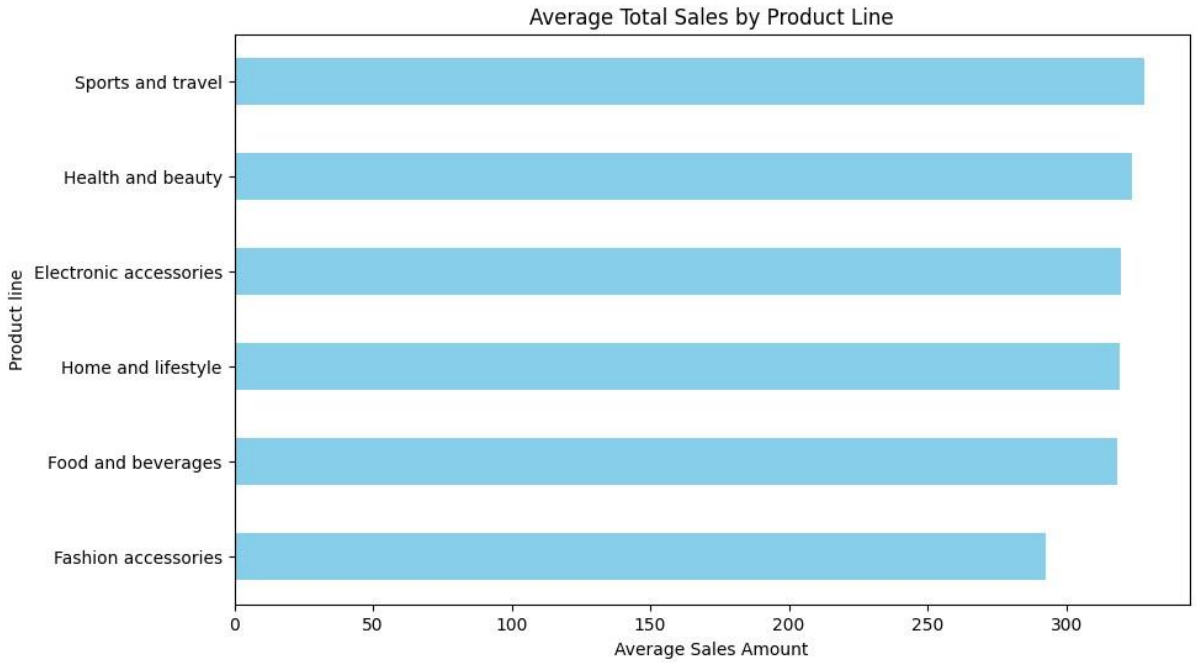
```
quantity_by_product.plot(kind='bar', x='Product
line', y='Quantity', title='Quantity Sales by Product
Line', xlabel='Product Line', ylabel='Quantity Sold')
plt.show()
```

Reason: Helps identify which product lines generate the highest sales, allowing for more targeted promotions and stock management.

Seasonal or Temporal Sales Patterns

A line chart shows how sales vary across different times of the year, helping to spot seasonal peaks or dips, which can guide marketing campaigns.





Reason: Understanding seasonal patterns can help businesses plan inventory, promotions, and marketing around high-demand periods.

8. *Customer Insights and Demographics*

Analysis of Customer Type and Rating Distribution

Box plots and count plots show how ratings differ across customer types, helping understand how different segments perceive products and services.

```
sns.boxplot(data=data, x='Customer type', y='Rating',
palette='pastel')
plt.title('Rating Distribution by Customer Type')
plt.show()
```

Reason: This provides insights into how different customer types rate products, helping businesses improve services for specific segments.

Gender-based Analysis of Sales Metrics

Visualizing sales and ratings based on gender using bar charts or histograms helps understand gender-based preferences and behaviors.

```
sns.countplot(data=data, x='Gender',
palette='coolwarm') plt.title('Sales
by Gender') plt.show()
```

Reason: Identifying gender-specific sales trends can help target products and marketing toward specific demographics.

Average Rating by Quantity and Product Line

Bar charts show the average rating for products based on the quantity sold, helping identify highly-rated products that customers prefer to buy in bulk.

```
avg_rating_by_quantity.plot(kind='bar',
color='salmon')
plt.title('Average Rating by Quantity Purchased')
plt.xlabel('Quantity') plt.ylabel('Average
Rating') plt.show()
```

Reason: This helps businesses understand the relationship between product popularity and customer satisfaction.

Customer Preferences for Payment Methods

Count plots show the frequency of different payment methods, indicating customer preferences and helping adjust payment systems to customer trends.

```
sns.countplot(data=data, x='Payment',
palette='coolwarm')
plt.title('Payment Method Distribution') plt.show()
```

Reason: Analyzing payment preferences helps businesses optimize payment options and improve the customer experience.

---

9. *Product Line Analysis*

Quantity Sales Distribution by Product Line

A bar chart shows how quantity sold varies by product line, helping prioritize popular product lines in promotions and inventory.

```
quantity_by_product.plot(kind='bar', x='Product
line', y='Quantity', title='Quantity Sales by Product
Line', xlabel='Product Line', ylabel='Quantity Sold')
plt.show()
```

Reason: Helps focus on the top-performing product lines for marketing and inventory.

Average Sales by Product Line

A horizontal bar chart illustrates average sales by product line, helping businesses identify which product lines generate the highest revenue.

```
avg_sales_by_product.plot(kind='barh',
color='skyblue')
plt.title('Average Total Sales by Product Line')
plt.xlabel('Average Sales Amount') plt.show()
```

Reason: Identifying high-performing product lines allows businesses to focus on the most profitable ones.

Rating Distribution Across Product Lines

A box plot shows the distribution of ratings for different product lines, providing insights into customer satisfaction.

```
sns.boxplot(data=data, x='Product line', y='Rating',
palette='Set3')
plt.title('Rating Distribution by Product Line')
plt.show()
```

Reason: Understanding product line ratings helps identify areas for product improvement or marketing focus.

Popular Product Lines by Customer Type

A bar plot shows how different customer types prefer certain product lines, guiding targeted promotions and product offerings.

```
sns.countplot(data=data, x='Product line',
hue='Customer type', palette='viridis')
plt.title('Product Line Popularity by Customer Type')
plt.show()
```

Reason: Helps businesses tailor product recommendations to specific customer segments.

---

10. *Branch and City-wise Sales Performance*

Sales by City and Branch Distribution

Bar plots show total sales across different cities and branches, helping identify high-revenue locations and areas for expansion or improvement.

```
Total_by_city.plot(kind='bar', x='City', y='Total',
title='Total by City', xlabel='City', ylabel='Total',
alpha=0.7, color='yellow') plt.xticks(rotation=0)
plt.grid(True, linestyle='--', color='gray',
linewidth=0.5) plt.show()
```

Reason: Identifying high-revenue cities and branches supports decisions on where to focus resources or marketing.

Comparison of Branches in Terms of Total Sales and Ratings

Side-by-side bar charts for sales and ratings by branch help assess branch performance, pinpointing top and underperforming branches.

```
sns.barplot(data=branch_comparison, x='Branch',
y='Sales', palette='viridis') plt.title('Branch
Sales Comparison') plt.show()
```

Reason: Helps compare branch performance, guiding resource allocation and branch-specific strategies.

Hourly Order Placement Trends by Branch

A line plot shows order volume per hour for each branch, helping optimize staffing and promotions during peak times.

```
orders_placed_by_hour.plot(kind='line', x='Hour',
y='Invoice ID', title='Orders Placed by Hour',
xlabel='Hour', ylabel='Invoice ID', alpha=0.7,
color='orange') plt.show()
```

Reason: Helps optimize staffing and promotional efforts around peak demand hours for each branch.

Analysis of City-wise Preferences for Products and Payment

Stacked bar charts visualize product line and payment method distribution by city, helping understand regional preferences.

```
city_payment_pref = data.groupby(['City', 'Product
line'])['Payment'].value_counts().unstack().plot(kind
='bar', stacked=True)
plt.title('City-wise Product and Payment
Preferences') plt.show()
```

Reason: Helps businesses understand regional product preferences and tailor marketing strategies or payment options accordingly.

11. *Feature Engineering*

Adding Date and Time Components

Extracting the month and hour from datetime features allows the dataset to be segmented by time, facilitating trend analysis and sales forecasting.

```
data['Month'] = data['Date'].dt.month data['Hour']
= data['Time'].dt.hour
```

Reason: Time-based features help uncover sales patterns by day, month, and hour, supporting better decision-making.

Aggregation of Sales by New Time-Based Features

Sales are aggregated based on the newly added month and hour features to analyze trends over different periods and help with inventory and marketing.

```
monthly_sales = data.groupby('Month')['Total'].sum()
hourly_sales = data.groupby('Hour')['Total'].sum()
```

Reason: Aggregating sales data by time-based features allows businesses to understand patterns like seasonality and peak times.

Encoding of Categorical Variables One-Hot Encoding for Branch and Customer Type

One-hot encoding is applied to categorical variables like branch and customer type to convert them into numerical formats suitable for modeling.

```
data = pd.get_dummies(data, columns=['Branch',
'Customer Type'])
```

Reason: Converts non-numeric columns to numeric, which is required for machine learning models.

Label Encoding for Gender

Label encoding is used for gender to convert categorical values (male, female) into numerical values for model training.

```
from sklearn.preprocessing import LabelEncoder le
= LabelEncoder() data['Gender'] =
le.fit_transform(data['Gender'])
```

Reason: Label encoding simplifies categorical variables for model consumption and maintains order (if any).

Binary Encoding for Multiclass Categorical Variables

Binary encoding is used for multiclass variables like payment method to reduce dimensionality while maintaining the integrity of the categorical features.

```
from category_encoders import BinaryEncoder
encoder = BinaryEncoder(cols=['Payment']) data
= encoder.fit_transform(data)
```

Reason: Reduces the number of columns created when encoding categorical features with many levels (like payment methods), improving model performance.

---

12. *Modeling for Predictive Analytics*

Introduction to Model Selection for Sales Prediction

The goal is to predict customer ratings based on sales data, helping businesses understand which factors influence customer satisfaction and product ratings.
Reason: This allows the business to predict future ratings, aiding in quality control and product strategy.

Data Preparation for Modeling Train-Test Split

Data is split into training and testing sets to ensure that the model is evaluated on unseen data.

```
from sklearn.model_selection import train_test_split
X = data.drop(columns=['Rating']) y
= data['Rating']
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)
```

Reason: Splitting the data ensures that the model is trained on one set and tested on a separate set, helping assess its generalization ability.

Feature Selection and Engineering for Model Input

Features are selected and engineered to improve model performance by removing irrelevant or highly correlated variables.

```
X_train = X_train[['Month', 'Quantity', 'Total']]  # Example selection of features
```

Reason: Selecting relevant features ensures the model is focused on the most impactful variables, improving accuracy.

Linear Regression Model Training

A linear regression model is trained to predict the customer ratings based on sales data, offering insights into the relationship between product performance and customer satisfaction.

```
from sklearn.linear_model import LinearRegression
model = LinearRegression() model.fit(X_train, y_train)
```

Reason: Linear regression is simple and interpretable, making it a good choice for understanding the relationship between features and the target variable.

Model Performance Evaluation

The model's performance is evaluated using metrics such as R-squared, Mean Absolute Error (MAE), and Mean Squared Error (MSE).

```
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(X_test)
print('R-squared:', r2_score(y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(y_test, y_pred))
```

Reason: These metrics help assess how well the model is predicting customer ratings, ensuring it is suitable for real-world applications.

---

13. *Conclusion*

Summary of Key Insights

The analysis identified key patterns such as high sales in specific months and customer segments, which can guide business decisions.
Reason: These insights help businesses optimize inventory, marketing strategies, and customer interactions.

Implications for Business Decisions Product Line Optimization

The analysis shows which product lines generate the most sales and customer satisfaction, guiding product focus and optimization.
Reason: Helps the business prioritize resources on top-selling products.

Seasonal Promotions and Inventory Planning

Understanding seasonal sales trends allows businesses to plan promotions and inventory around high-demand periods.
Reason: Optimizes stock management and boosts sales during peak times.

Customer Targeting and Personalization

By analyzing customer preferences and behavior, businesses can create personalized marketing campaigns tailored to different customer types. **Reason:** Increases customer engagement and conversion rates.

Limitations of the Analysis

The analysis assumes consistent sales data quality and external factors like marketing campaigns or economic shifts were not considered.
Reason: Acknowledging limitations helps in improving future analyses and understanding the scope of insights.

Recommendations for Future Analysis and Improvement

Future analysis could integrate external data, such as marketing spend or economic indicators, for a more comprehensive view.
Reason: Incorporating additional data sources will increase the robustness and accuracy of business decisions.

---

14. *Appendices*

Appendix A: Code Snippets

This section will include all the code used throughout the analysis for transparency and reproducibility.

Appendix B: List of All Figures and Tables

A comprehensive list of all the charts and tables used in the documentation for easy reference.

Appendix C: Glossary of Terms

Defines key terms and concepts used in the analysis, helping readers unfamiliar with technical language.

Appendix D: Additional Visualizations

Includes additional charts and graphs that were not covered in the main sections but provide further insights into the data.

---

15. *References and Resources*

Recommended Books and Articles

Books and articles on data analysis, retail strategies, and machine learning are suggested for further learning.

Documentation for Libraries and Methods Used

Links to documentation for Python libraries (like Pandas, Matplotlib, Seaborn, Scikit-learn) used in the analysis.

Other Useful Resources

Other online resources, tutorials, or case studies relevant to retail sales analysis are provided for deeper insights.