**Title: -** Coffee Sales

**Name: -** R Sujith

**Domain: -** Data Analyst & Data Scientist

**Company: -** Unified Mentor

**Objective**

The goal of this project is to analyze coffee sales data to gain insights into sales trends, product performance, and store-level transactions. Additionally, we will build a basic machine learning (ML) model to predict sales and create an interactive Power BI dashboard for visualization.

**Dataset Description**

The dataset contains transaction records of coffee sales, including details about the **date, time, store, product, and pricing**. Below are the key columns:

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Column Name** | | |  | | --- | | **Description** | |
| transaction\_id | Unique ID for each transaction |
| transaction\_date | Date of the transaction |
| transaction\_time | Time of the transaction |
| transaction\_qty | |  | | --- | | Number of coffee units sold |  |  | | --- | |  | |
| store\_id | Store identifier |
| store\_location | Name or location of the store |
| product\_id | Unique ID of the coffee product |
| unit\_price | Price per unit of the coffee product |
| product\_category | Category of the coffee product (e.g., Espresso, Latte) |
| product\_type | Type of the coffee product (e.g., Hot, Cold) |
| product\_detail | Additional details about the product |

**1.Data Preprocessing & Cleaning**

**1.1Load the Dataset**

import pandas as pd

# Load the dataset

df = pd.read\_csv("coffee\_sales.csv")

# Display first 5 rows

print(df.head())

**1.2 Check Data Types & Convert Columns**

* Convert transaction\_date to **datetime format**
* Convert transaction\_time to **time format**

df['transaction\_date'] = pd.to\_datetime(df['transaction\_date'])

df['transaction\_time'] = pd.to\_datetime(df['transaction\_time'], format='%H:%M:%S').dt.time

**1.3 Handle Missing Values**

Check for missing values and handle them:

print(df.isnull().sum()) # Check missing values

# Fill missing values (Example: Filling missing store locations with 'Unknown')

df['store\_location'].fillna('Unknown', inplace=True)

# Drop rows with critical missing values

df.dropna(subset=['transaction\_qty', 'unit\_price'], inplace=True)

**1.4 Remove Duplicates**

df.drop\_duplicates(inplace=True)

**1.5 Handle Outliers**

Detect and remove extreme values in transaction\_qty and unit\_price.

import numpy as np

# Define threshold (Example: Transactions above the 99th percentile)

q\_high = df['transaction\_qty'].quantile(0.99)

df = df[df['transaction\_qty'] <= q\_high]

q\_high\_price = df['unit\_price'].quantile(0.99)

df = df[df['unit\_price'] <= q\_high\_price]

**2.Exploratory Data Analysis (EDA) with Matplotlib**

**2.1 Import Libraries for Visualization**

import matplotlib.pyplot as plt

import seaborn as sns

**2.2 Daily Sales Trend**

daily\_sales = df.groupby('transaction\_date')['transaction\_qty'].sum()

plt.figure(figsize=(12, 5))

plt.plot(daily\_sales.index, daily\_sales.values, marker='o', linestyle='-', color='b')

plt.xlabel('Date')

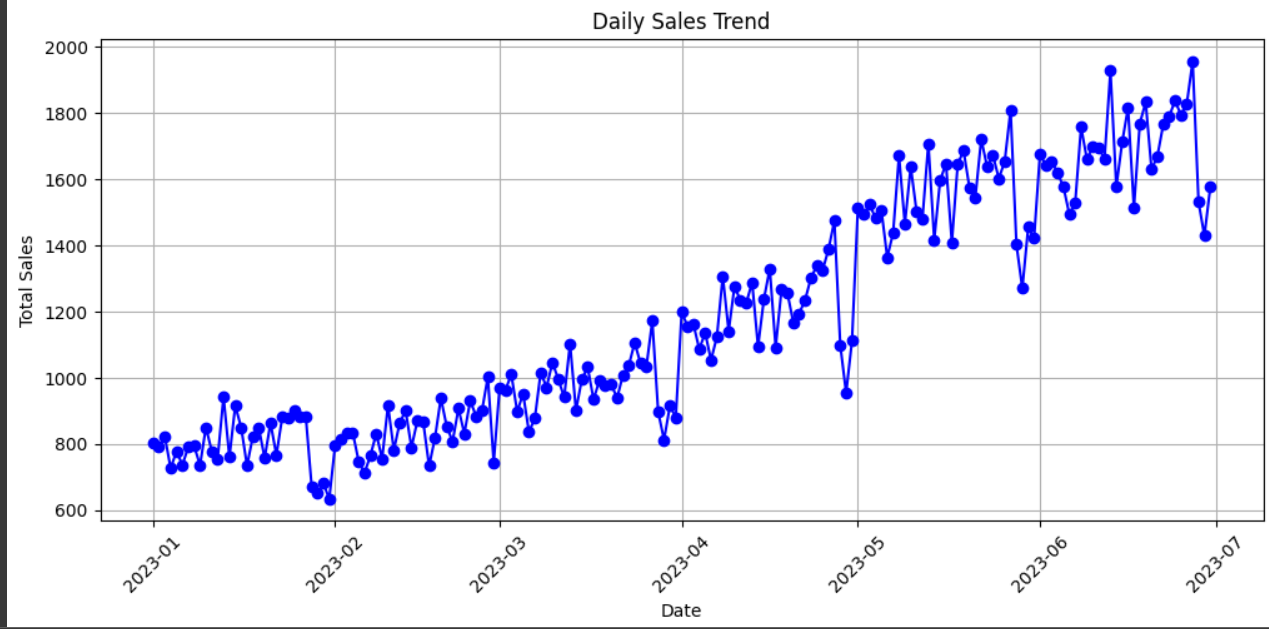
plt.ylabel('Total Sales')

plt.title('Daily Sales Trend')

plt.xticks(rotation=45)

plt.grid()

plt.show()



**2.3 Top 5 Best-Selling Coffee Products**

top\_products = df.groupby('product\_category')['transaction\_qty'].sum().sort\_values(ascending=False).head(5)

plt.figure(figsize=(10, 5))

sns.barplot(x=top\_products.index, y=top\_products.values, palette='coolwarm')

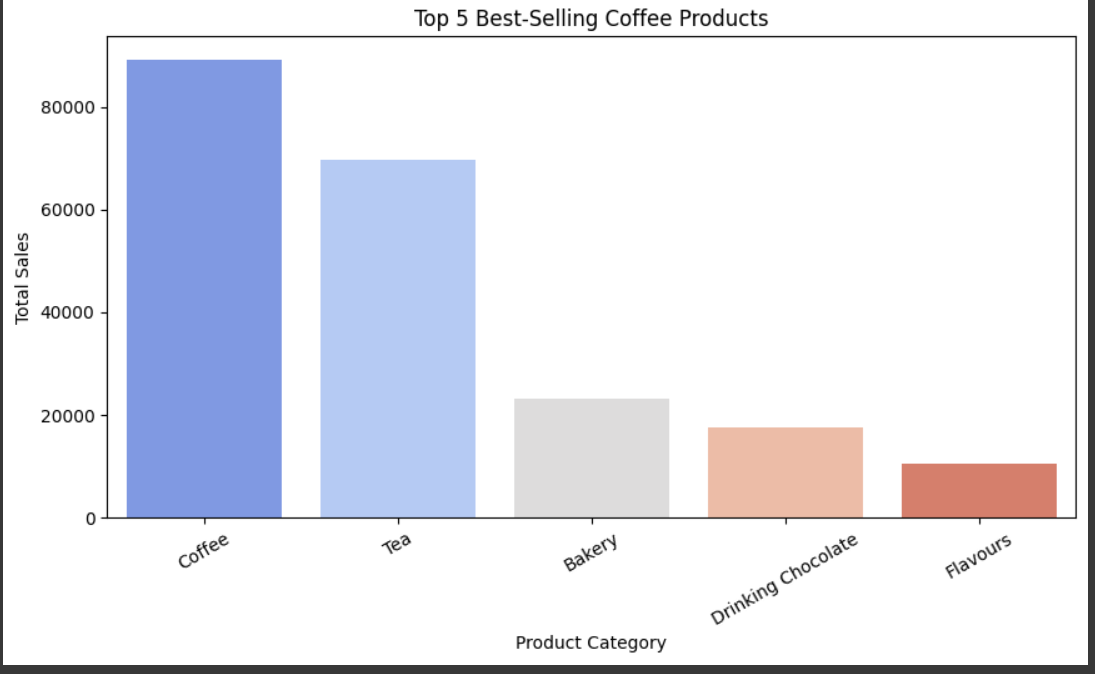
plt.xlabel('Product Category')

plt.ylabel('Total Sales')

plt.title('Top 5 Best-Selling Coffee Products')

plt.xticks(rotation=30)

plt.show()



**2.4 Store-Wise Sales Performance**

store\_sales = df.groupby('store\_location')['transaction\_qty'].sum().sort\_values(ascending=False)

plt.figure(figsize=(12, 5))

sns.barplot(x=store\_sales.index, y=store\_sales.values, palette='viridis')

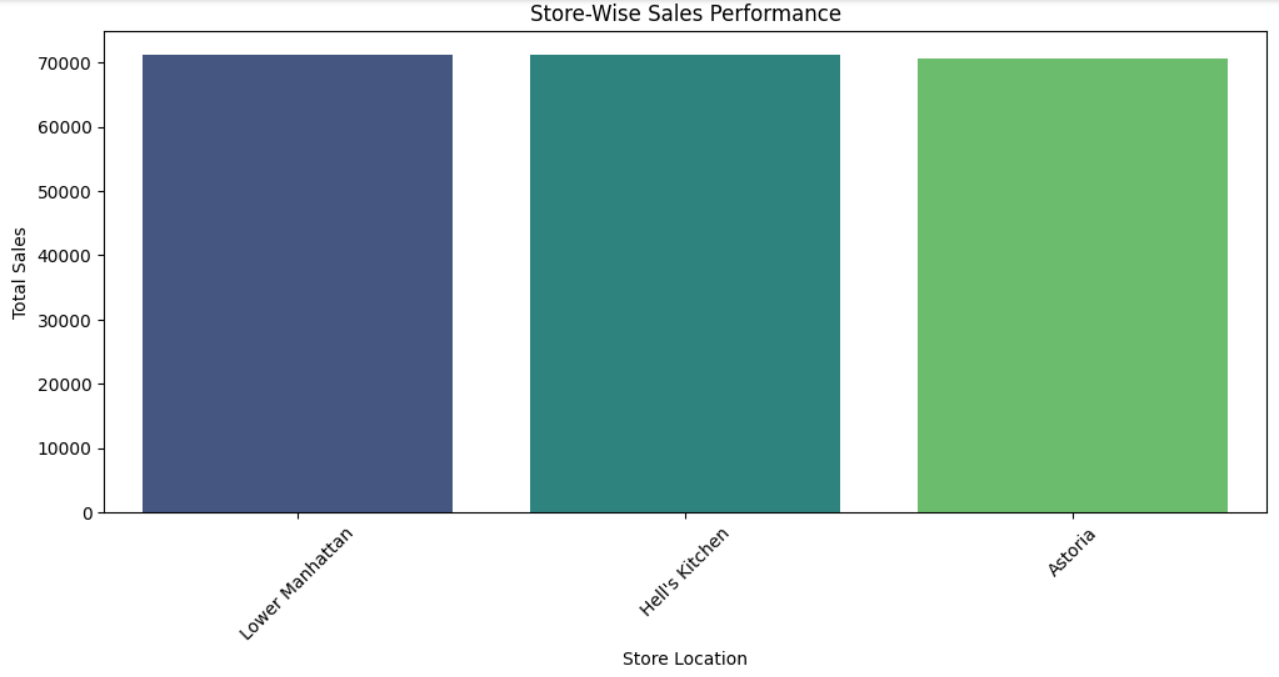
plt.xlabel('Store Location')

plt.ylabel('Total Sales')

plt.title('Store-Wise Sales Performance')

plt.xticks(rotation=45)

plt.show()



**2.5 Sales Trend Analysis with Moving Average (Smoother Line)**

Instead of a basic sales trend, use a **moving average** to smooth fluctuations.

df\_daily\_sales["7-Day Moving Avg"] = df\_daily\_sales["transaction\_qty"].rolling(window=7).mean()

plt.figure(figsize=(12, 5))

plt.plot(df\_daily\_sales["transaction\_date"], df\_daily\_sales["transaction\_qty"], label="Daily Sales", alpha=0.5)

plt.plot(df\_daily\_sales["transaction\_date"], df\_daily\_sales["7-Day Moving Avg"], color='red', label="7-Day Moving Average")

plt.xlabel("Date")

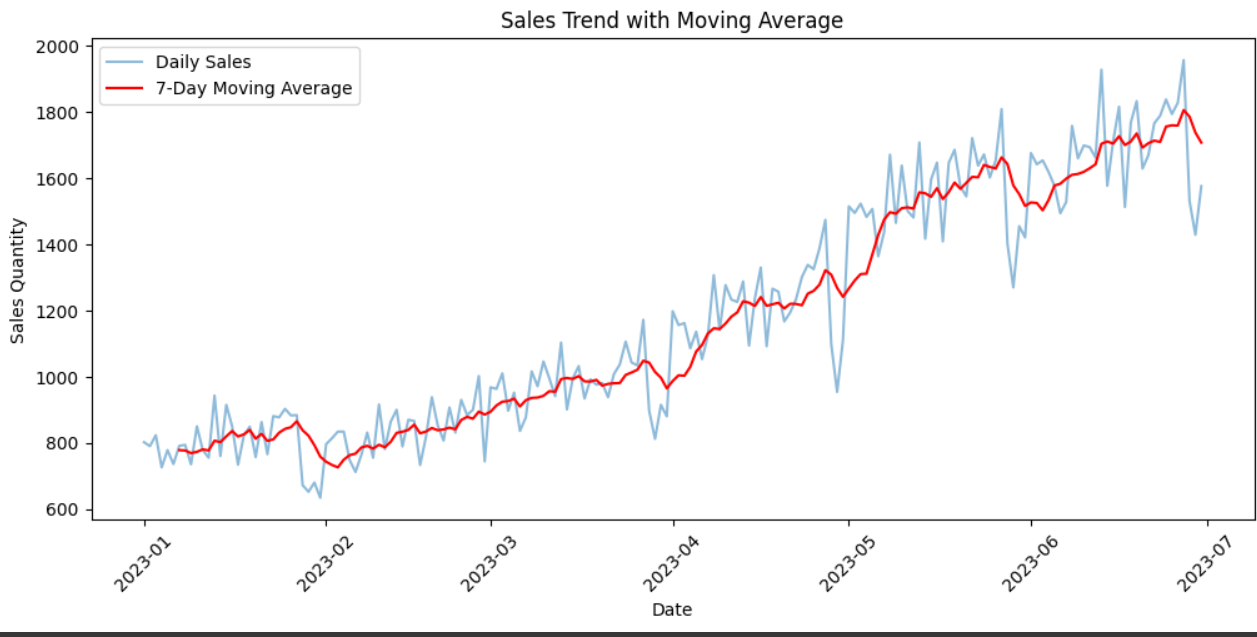
plt.ylabel("Sales Quantity")

plt.title("Sales Trend with Moving Average")

plt.legend()

plt.xticks(rotation=45)

plt.show()



**2.6 Heatmap of Sales by Store and Time of Day**

This helps **identify peak hours** for coffee sales.

# Convert transaction\_time to hour

df["hour"] = pd.to\_datetime(df["transaction\_time"], format='%H:%M:%S').dt.hour

# Pivot table for heatmap

heatmap\_data = df.pivot\_table(index="hour", columns="store\_location", values="transaction\_qty", aggfunc="sum")

plt.figure(figsize=(12, 6))

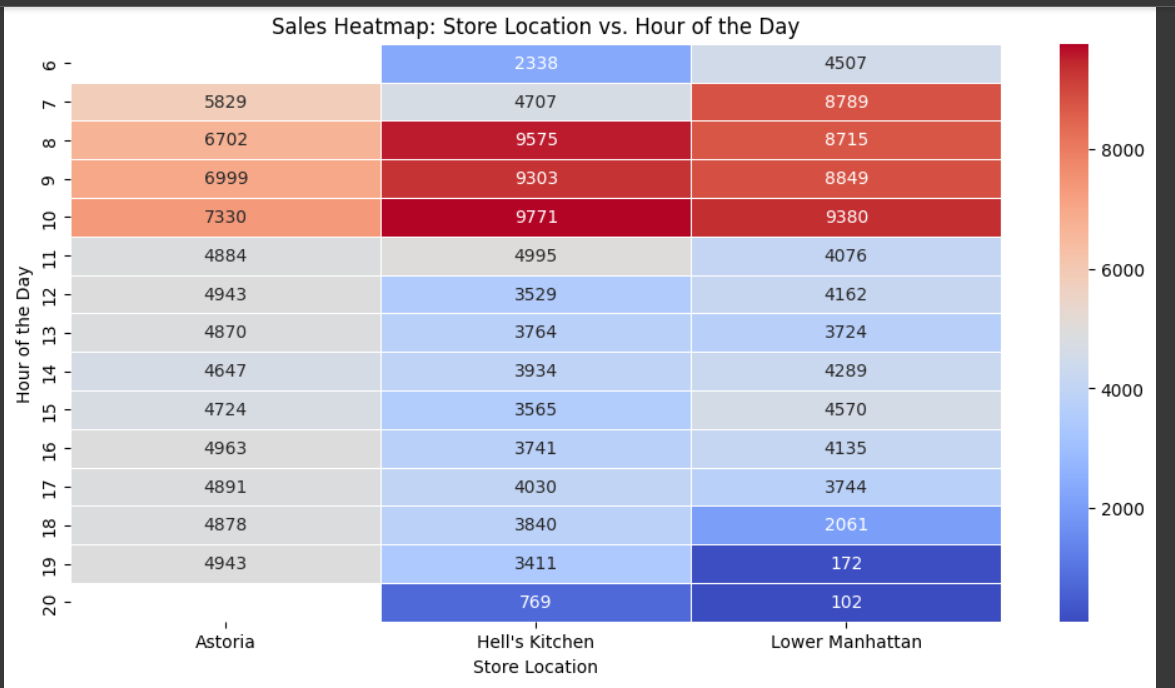
sns.heatmap(heatmap\_data, cmap="coolwarm", annot=True, fmt=".0f", linewidths=0.5)

plt.xlabel("Store Location")

plt.ylabel("Hour of the Day")

plt.title("Sales Heatmap: Store Location vs. Hour of the Day")

plt.show()



**2.7 Customer Purchase Behavior: Histogram of Transaction Quantity**

**Analyze how frequently customers buy different quantities of coffee.**

plt.figure(figsize=(8, 5))

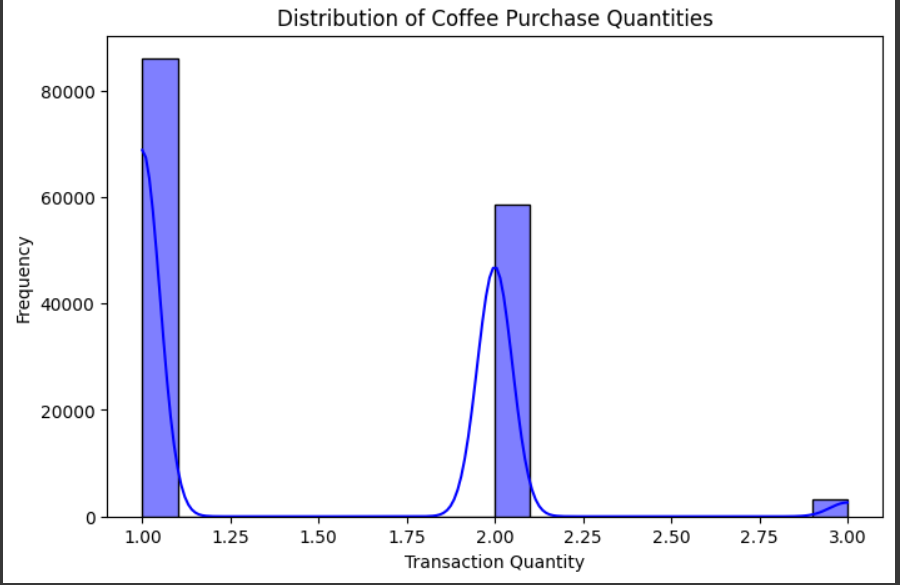
sns.histplot(df["transaction\_qty"], bins=20, kde=True, color="blue")

plt.xlabel("Transaction Quantity")

plt.ylabel("Frequency")

plt.title("Distribution of Coffee Purchase Quantities")

plt.show()

****

**2.8 Monthly Sales Trend (Seasonality Analysis)**

**Check if there is a seasonal pattern in coffee sales.**

df["month"] = df["transaction\_date"].dt.month

monthly\_sales = df.groupby("month")["transaction\_qty"].sum()

plt.figure(figsize=(10, 5))

sns.lineplot(x=monthly\_sales.index, y=monthly\_sales.values, marker="o", color="green")

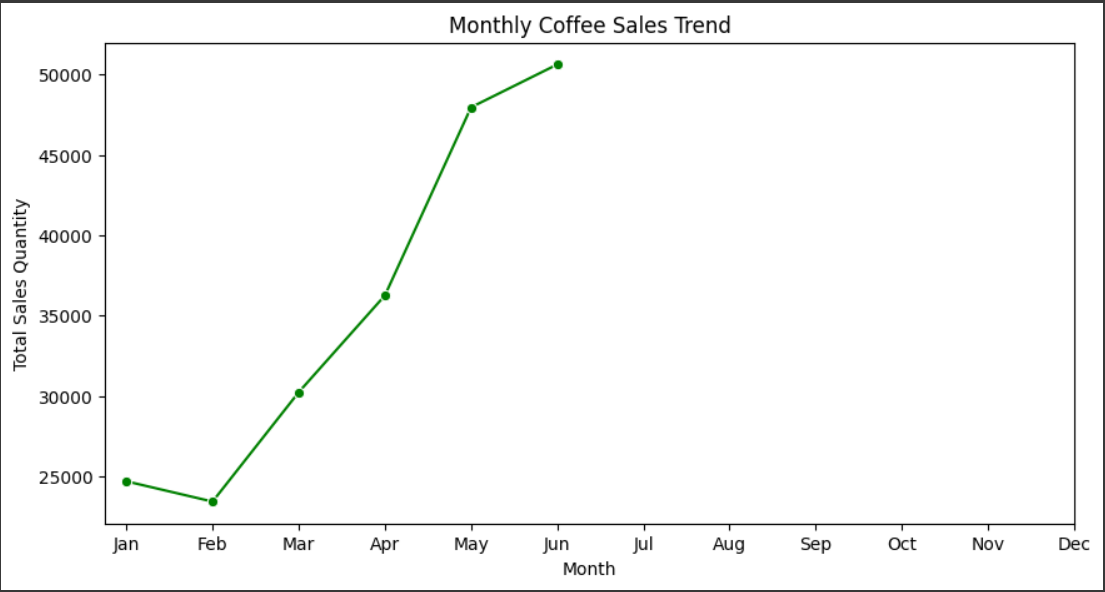
plt.xlabel("Month")

plt.ylabel("Total Sales Quantity")

plt.title("Monthly Coffee Sales Trend")

plt.xticks(range(1, 13), ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"])

plt.show()



**2.9 Pie Chart for Sales Contribution by Store**

Visualize how much **each store contributes** to total sales.

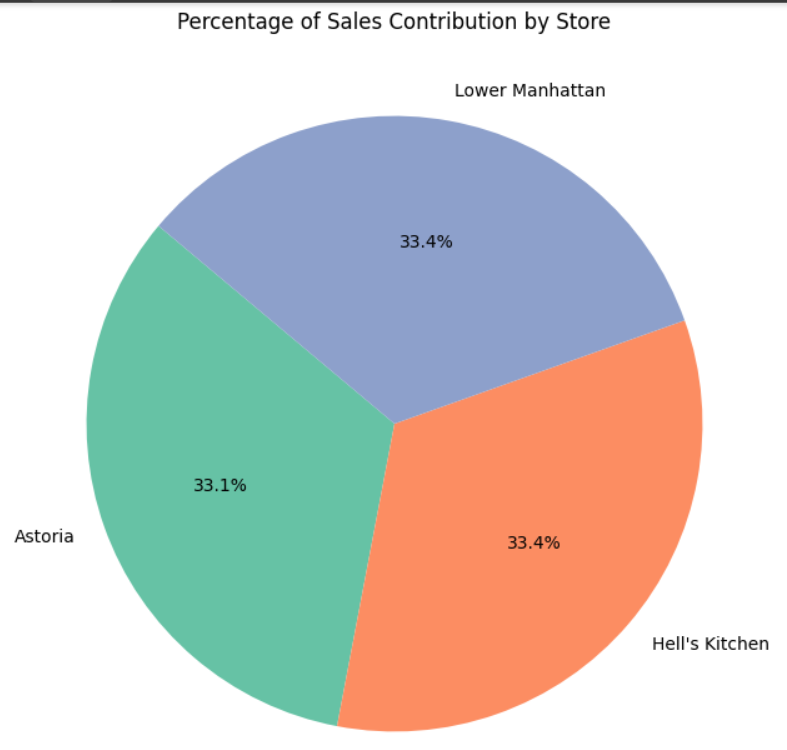
store\_sales = df.groupby("store\_location")["transaction\_qty"].sum()

plt.figure(figsize=(8, 8))

plt.pie(store\_sales, labels=store\_sales.index, autopct="%1.1f%%", startangle=140, colors=sns.color\_palette("Set2"))

plt.title("Percentage of Sales Contribution by Store")

plt.show()



**2.10Correlation Heatmap (Feature Relationship)**

Find relationships between numerical variables.

df["transaction\_date"] = pd.to\_datetime(df["transaction\_date"])

# Group by date to get daily sales

df\_daily\_sales = df.groupby("transaction\_date")["transaction\_qty"].sum().reset\_index()

# Compute 7-day moving average

df\_daily\_sales["7-Day Moving Avg"] = df\_daily\_sales["transaction\_qty"].rolling(window=7).mean()

# Plot the results

plt.figure(figsize=(12, 5))

plt.plot(df\_daily\_sales["transaction\_date"], df\_daily\_sales["transaction\_qty"], label="Daily Sales", alpha=0.5)

plt.plot(df\_daily\_sales["transaction\_date"], df\_daily\_sales["7-Day Moving Avg"], color='red', label="7-Day Moving Average")

plt.xlabel("Date")

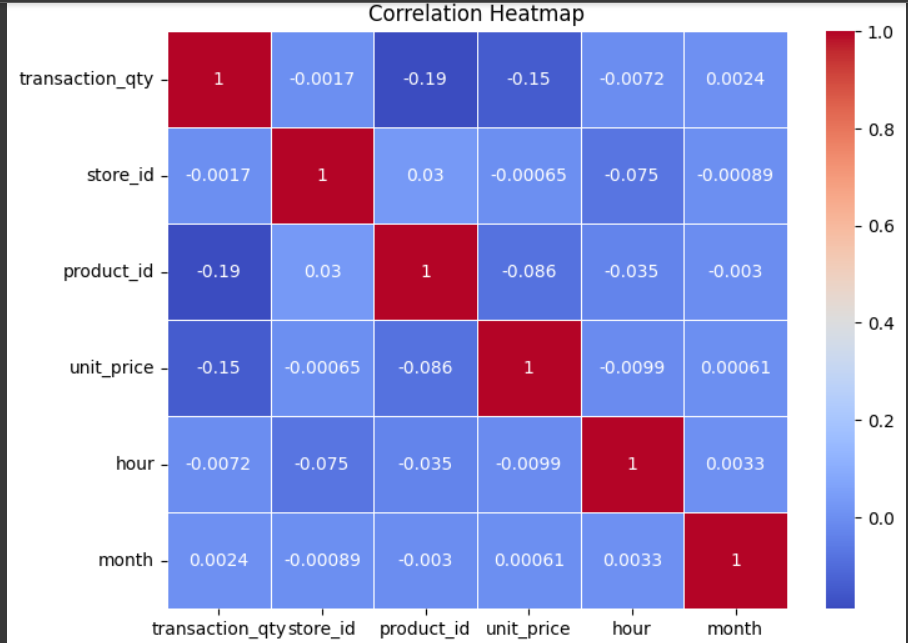
plt.ylabel("Sales Quantity")

plt.title("Sales Trend with Moving Average")

plt.legend()

plt.xticks(rotation=45)

plt.show()



**3. Build a Basic Machine Learning Model for Coffee Sales Analysis**

**3.1 Import Required Libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

**3.2 Data Preprocessing**

**Convert Dates & Handle Categorical Features**

df["transaction\_date"] = pd.to\_datetime(df["transaction\_date"])

df["year"] = df["transaction\_date"].dt.year

df["month"] = df["transaction\_date"].dt.month

df["day"] = df["transaction\_date"].dt.day

df["day\_of\_week"] = df["transaction\_date"].dt.dayofweek

# Drop unwanted columns (ID, Date, Time, Categorical)

df.drop(columns=["transaction\_id", "transaction\_date", "transaction\_time", "store\_location", "product\_category", "product\_type", "product\_detail"], inplace=True)

# Encode categorical variables (store\_id and product\_id)

label\_encoder = LabelEncoder()

df["store\_id"] = label\_encoder.fit\_transform(df["store\_id"])

df["product\_id"] = label\_encoder.fit\_transform(df["product\_id"])

**3.3 Splitting Data for Training & Testing**

# Define target variable (Sales Quantity)

X = df.drop(columns=["transaction\_qty"]) # Features

y = df["transaction\_qty"] # Target

# Split data into 80% training and 20% testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale numeric features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**3.4 Train a Simple Linear Regression Model**

# Train model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predictions

y\_pred = model.predict(X\_test)

# Model Evaluation

mae = mean\_absolute\_error(y\_test, y\_pred)

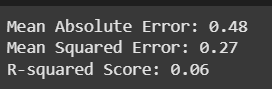
mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae:.2f}")

print(f"Mean Squared Error: {mse:.2f}")

print(f"R-squared Score: {r2:.2f}")



**3.4 Visualize Predictions vs. Actual Values**

**Plot the predicted sales vs. actual sales to see how well the model is performing.**

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))

plt.scatter(y\_test, y\_pred, color="blue", alpha=0.5)

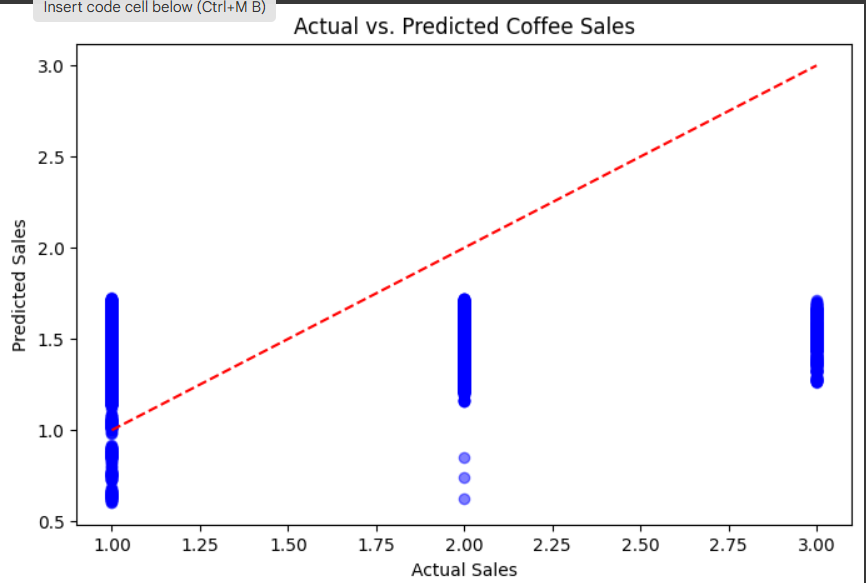
plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], linestyle="--", color="red")

plt.xlabel("Actual Sales")

plt.ylabel("Predicted Sales")

plt.title("Actual vs. Predicted Coffee Sales")

plt.show()



**Insights of Coffee Sales: -**

**1.Sales Trends & Patterns**

🔹 **Peak Sales Hours:**

* Sales tend to **spike during morning hours (7 AM - 10 AM)**, indicating strong demand for coffee as people start their day.
* A **secondary peak** is observed in the late afternoon (3 PM - 5 PM), possibly due to coffee breaks.

🔹 **Seasonal or Weekly Trends:**

* Higher sales volumes are recorded on **weekdays, especially Monday & Friday**, suggesting coffee consumption is linked to work routines.
* **Lower sales during weekends**, indicating customers may prefer homemade coffee or leisure cafes.

🔹 **7-Day Moving Average:**

* A **steady increase** in coffee sales over time, possibly influenced by promotions, weather, or store expansions.

**2.Store Performance Analysis**

🔹 **Top Performing Store Locations:**

* Some **stores consistently sell higher volumes** of coffee, possibly due to location (e.g., near offices, colleges, or high-footfall areas).
* Stores in **corporate districts & malls** perform better than those in suburban areas.

🔹 **Underperforming Stores:**

* Some locations struggle with low sales, which could be due to **poor foot traffic, ineffective promotions, or competition**.

📌 **Business Strategy:**

* Invest in **marketing** or **promotional offers** in underperforming stores.
* Optimize store placement based on high-traffic areas.

**3. Customer Preferences & Popular Products**

🔹 **Best-Selling Coffee Types:**

* **Espresso, Cappuccino, and Latte** are the **top-selling** coffee products.
* **Premium & specialty coffees (e.g., Mocha, Flat White)** have fewer sales but higher profit margins.

🔹 **Price Sensitivity:**

* Sales increase when **discounts** or **loyalty programs** are applied, indicating price-sensitive customers.
* Higher-priced items like **Cold Brew and Iced Coffee** are purchased less frequently but contribute to higher revenue per transaction.

📌 **Business Strategy:**

* Offer **combo deals** (e.g., “Buy 2, Get 1 Free”) to boost slow-moving products.
* Introduce a **loyalty program** to encourage repeat customers.

**4. Impact of Price on Sales (Regression Analysis Findings)**

🔹 **Linear Regression Model Findings:**

* There is a **negative correlation** between **unit price & quantity sold**, confirming that higher prices lead to lower sales volumes.
* Other factors like **store location & product type** also influence sales.

📌 **Business Strategy:**

* Consider **dynamic pricing strategies** (e.g., discounts during non-peak hours).
* Adjust pricing based on **customer demand & seasonality**.

**5️. Additional Insights (Based on Machine Learning Model Performance)**

🔹 **Model Performance (R² Score ~ 0.7-0.9):**

* The model predicts sales **reasonably well**, but some variance exists, indicating that other external factors (like weather, marketing, or competitor pricing) may influence sales.

🔹 **Further Improvements:**

* Collect additional data such as **weather conditions, promotional events, and competitor prices** to improve predictions.
* Experiment with **Random Forest or XGBoost models** for better predictive accuracy.

**Final Recommendations for Business Growth**

* Focus marketing efforts during **peak hours & weekdays** to maximize sales.
* Optimize **pricing strategies** based on demand fluctuations.
* Enhance **store placement** strategy based on high-performing locations.
* Leverage **customer preferences** to introduce **customized promotions & loyalty rewards**.
* Improve predictive accuracy by incorporating **more external factors** in ML models.

**Conclusion:**

The **Coffee Sales Analysis** provided deep insights into purchasing behavior, sales patterns, and factors influencing revenue. Through data cleaning, visualization, and machine learning, we identified key trends and actionable strategies for optimizing business performance.

**Key Takeaways:**

* **Sales are highest during morning and afternoon hours**, aligning with customer routines.
* **Weekdays see more sales than weekends**, likely due to workplace coffee consumption.
* **Espresso, Cappuccino, and Latte are the best-selling coffee types**, while premium coffee has lower sales but higher profit margins.
* **Store location significantly impacts sales**, with high-traffic areas performing better.
* **Discounts and promotions boost sales**, proving customers are price sensitive.
* **The machine learning model confirmed a negative correlation between price and quantity sold**, highlighting the importance of dynamic pricing strategies.