Store Inventory Optimization

DATA

import pandas as pd

from datetime import datetime, timedelta

import random

# Generate random data

data = []

start\_date = datetime.now() - timedelta(days=365)

for i in range(1000): # Adjust this number to control the number of rows in the dataset

product\_id = random.randint(1, 100)

product\_name = f"Product\_{product\_id}"

category = random.choice(["electronics", "clothing", "toys", "books", "sports"])

price = round(random.uniform(10, 1000), 2)

units\_sold = random.randint(1, 20)

date = (start\_date + timedelta(days=i % 365)).strftime("%Y-%m-%d")

data.append([product\_id, product\_name, category, price, units\_sold, date])

# Create a pandas DataFrame

columns = ["product\_id", "product\_name", "category", "price", "units\_sold", "date"]

sales\_data = pd.DataFrame(data, columns=columns)

# Display the first few rows of the dataset

print(sales\_data.head())

# Save the dataset as a CSV file

sales\_data.to\_csv("sales\_data.csv", index=False)

Output:  
 product\_id product\_name category price units\_sold date

0 60 Product\_60 books 54.89 14 2022-04-16

1 65 Product\_65 toys 317.94 20 2022-04-17

2 55 Product\_55 books 448.82 10 2022-04-18

3 45 Product\_45 sports 396.37 11 2022-04-19

4 88 Product\_88 clothing 421.31 2 2022-04-20

Remove any missing or duplicate data Analyze the dataset to understand the trends and patterns in the data. Create visualizations using bar charts, pie charts

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Read the dataset from the CSV file

sales\_data = pd.read\_csv("sales\_data.csv")

# Remove missing values

sales\_data.dropna(inplace=True)

# Remove duplicate rows

sales\_data.drop\_duplicates(inplace=True)

# Display the first few rows of the cleaned dataset

print(sales\_data.head())

# Analyze the data: Total units sold by category

category\_units = sales\_data.groupby('category')['units\_sold'].sum().reset\_index()

# Bar chart: Units sold by category

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

sns.barplot(x='category', y='units\_sold', data=category\_units)

plt.title('Units Sold by Category')

plt.show()

# Pie chart: Units sold by category

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

plt.pie(category\_units['units\_sold'], labels=category\_units['category'], autopct='%1.1f%%')

plt.title('Units Sold by Category')

plt.show()

# Analyze the data: Total sales by month

sales\_data['date'] = pd.to\_datetime(sales\_data['date'])

sales\_data['month'] = sales\_data['date'].dt.strftime('%Y-%m')

monthly\_sales = sales\_data.groupby('month')['units\_sold'].sum().reset\_index()

# Bar chart: Units sold by month

plt.figure(figsize=(15, 5))

sns.barplot(x='month', y='units\_sold', data=monthly\_sales)

plt.title('Units Sold by Month')

plt.xticks(rotation=45)

plt.show()

# Analyze the data: Average price by category

category\_prices = sales\_data.groupby('category')['price'].mean().reset\_index()

# Bar chart: Average price by category

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

sns.barplot(x='category', y='price', data=category\_prices)

plt.title('Average Price by Category')

plt.show()

Output:

product\_id product\_name category price units\_sold date

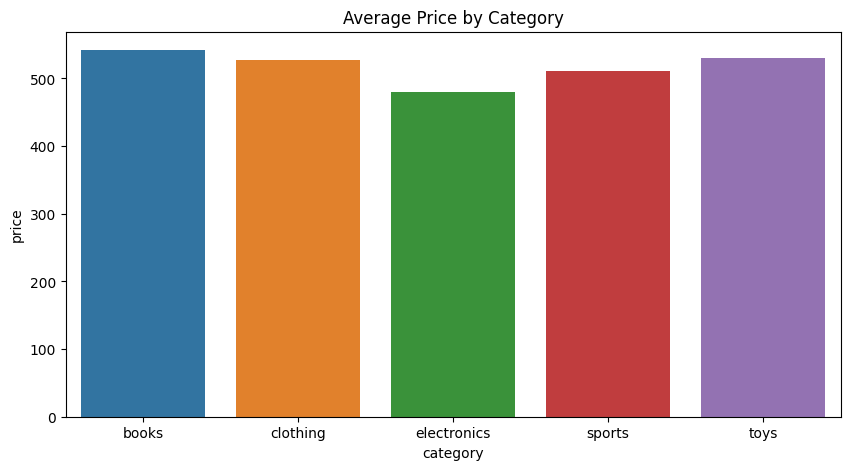
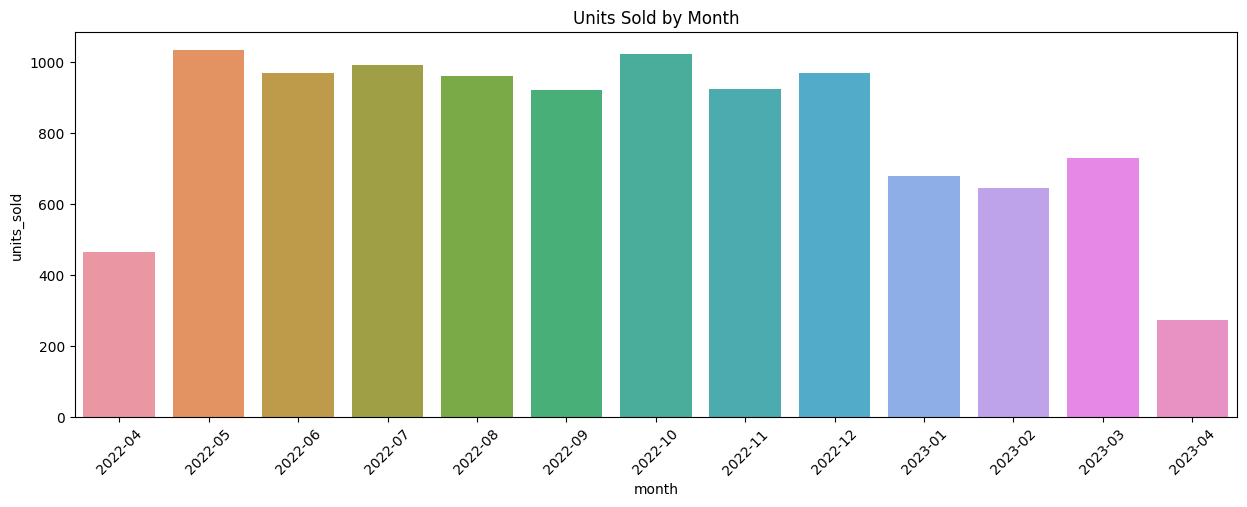
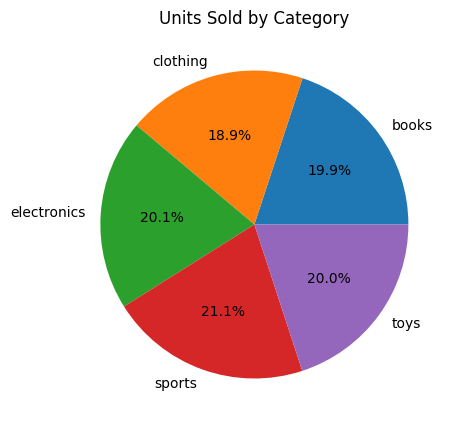
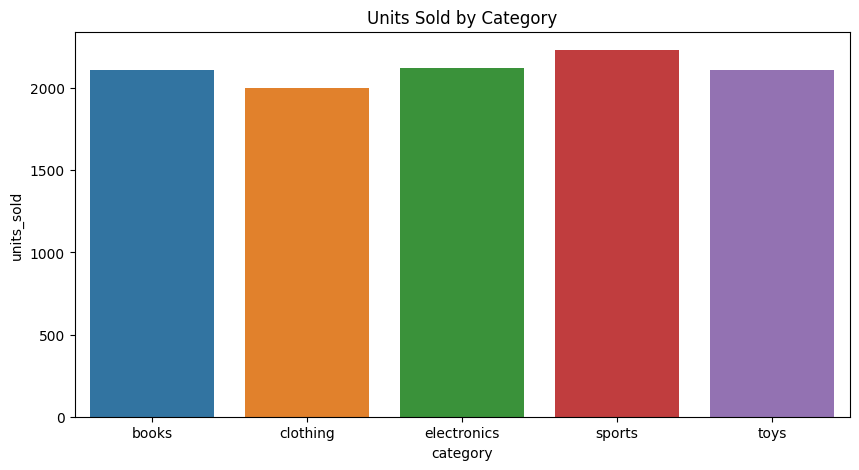
0 60 Product\_60 books 54.89 14 2022-04-16

1 65 Product\_65 toys 317.94 20 2022-04-17

2 55 Product\_55 books 448.82 10 2022-04-18

3 45 Product\_45 sports 396.37 11 2022-04-19

4 88 Product\_88 clothing 421.31 2 2022-04-20



Average daily sales Standard deviation of daily sales Sales velocity bargraph

import numpy as np

import random

# Calculate average daily sales and standard deviation for each product

product\_daily\_sales = sales\_data.groupby(['product\_id', 'product\_name', 'date'])['units\_sold'].sum().reset\_index()

product\_metrics = product\_daily\_sales.groupby(['product\_id', 'product\_name']).agg(

avg\_daily\_sales=('units\_sold', np.mean),

std\_daily\_sales=('units\_sold', np.std),

).reset\_index()

# Assuming a hypothetical total stock for each product

total\_stock = {product\_id: random.randint(50, 200) for product\_id in product\_metrics['product\_id'].unique()}

# Calculate sales velocity for each product

product\_metrics['total\_stock'] = product\_metrics['product\_id'].apply(lambda x: total\_stock[x])

product\_metrics['sales\_velocity'] = product\_metrics['avg\_daily\_sales'] / product\_metrics['total\_stock']

# Display the first few rows of the calculated metrics

print(product\_metrics.head())

# Bar chart: Average daily sales by product

plt.figure(figsize=(20, 8))

sns.barplot(x='product\_id', y='avg\_daily\_sales', data=product\_metrics)

plt.title('Average Daily Sales by Product')

plt.xticks(rotation=90)

plt.show()

# Bar chart: Standard deviation of daily sales by product

plt.figure(figsize=(20, 8))

sns.barplot(x='product\_id', y='std\_daily\_sales', data=product\_metrics)

plt.title('Standard Deviation of Daily Sales by Product')

plt.xticks(rotation=90)

plt.show()

# Bar chart: Sales velocity by product

plt.figure(figsize=(20, 8))

sns.barplot(x='product\_id', y='sales\_velocity', data=product\_metrics)

plt.title('Sales Velocity by Product')

plt.xticks(rotation=90)

plt.show()

product\_id product\_name avg\_daily\_sales std\_daily\_sales total\_stock \

0 1 Product\_1 9.666667 7.126561 50

1 2 Product\_2 11.545455 5.989385 153

2 3 Product\_3 11.214286 5.146748 162

3 4 Product\_4 6.142857 6.938505 105

4 5 Product\_5 12.000000 5.009083 184

sales\_velocity

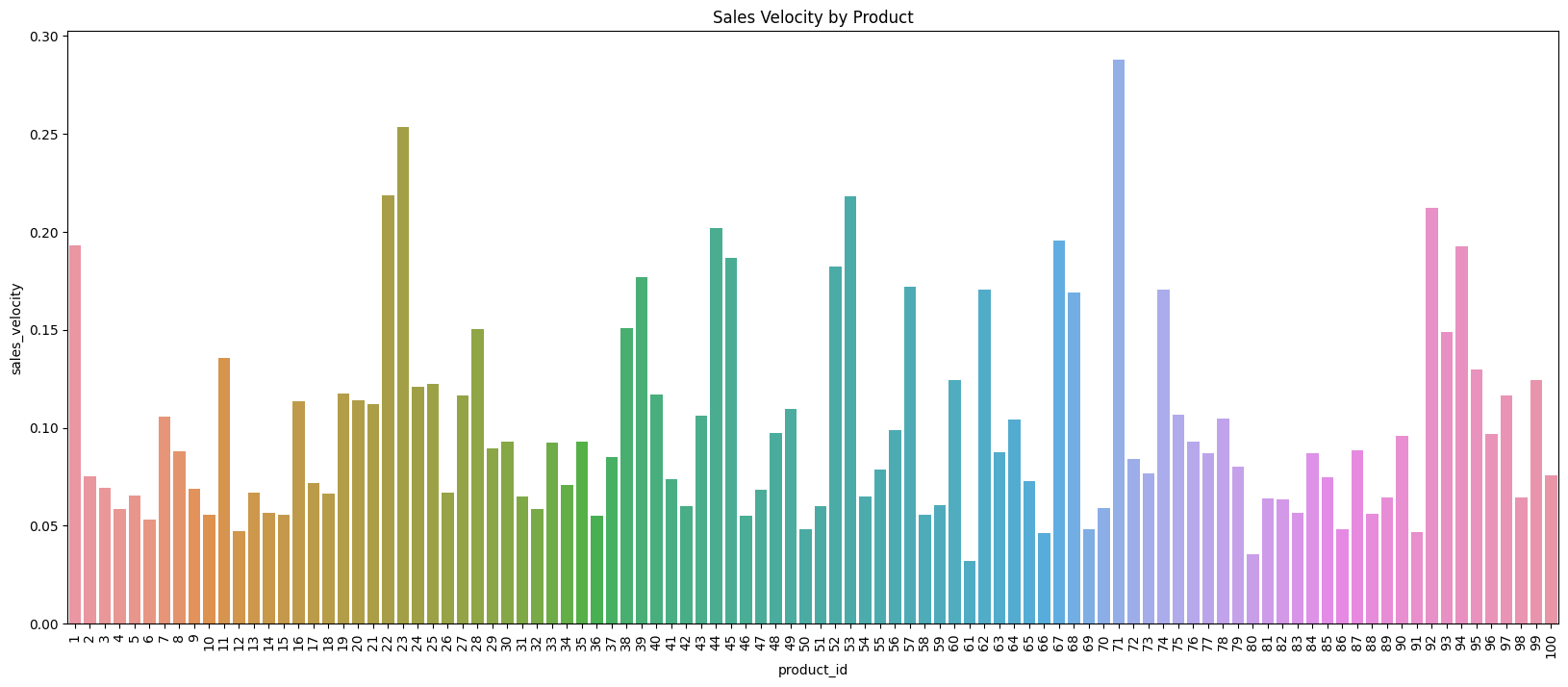
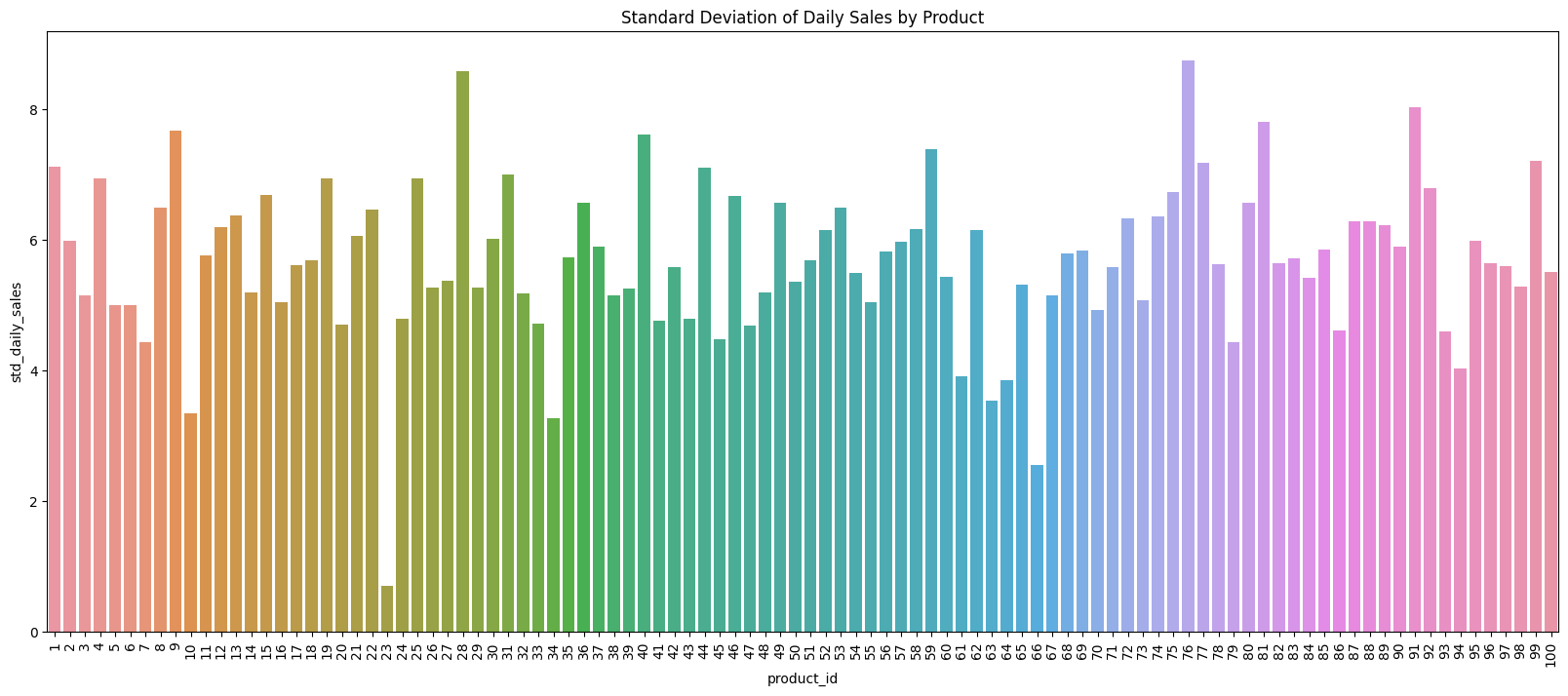
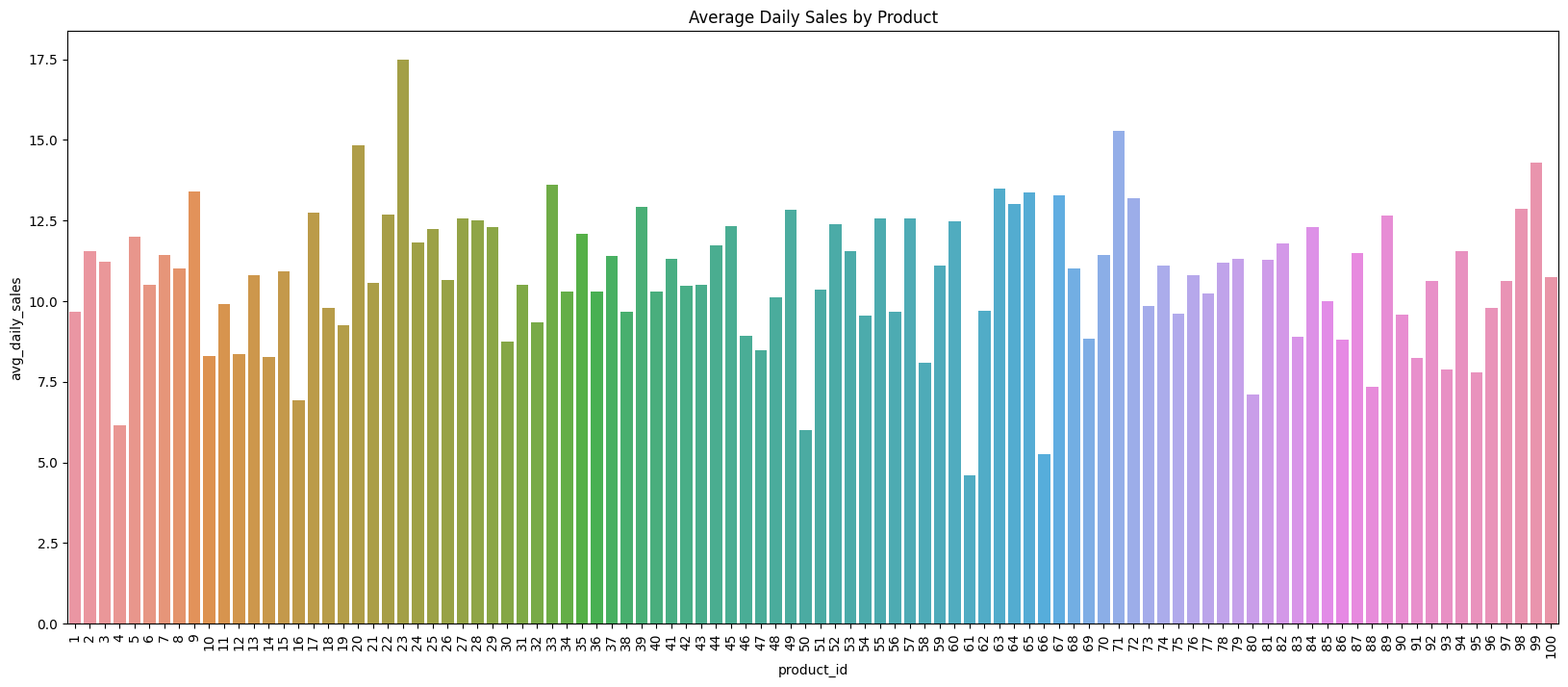
0 0.193333

1 0.075460

2 0.069224

3 0.058503

4 0.065217



seasonal trend

# Analyze the data: Total sales by month and category

monthly\_category\_sales = sales\_data.groupby(['month', 'category'])['units\_sold'].sum().reset\_index()

# Create a pivot table for better visualization

pivot\_monthly\_category\_sales = monthly\_category\_sales.pivot\_table(index='month', columns='category', values='units\_sold').fillna(0)

# Bar chart: Monthly sales by category

pivot\_monthly\_category\_sales.plot.bar(stacked=True, figsize=(15, 8))

plt.title('Monthly Sales by Category')

plt.xticks(rotation=45)

plt.ylabel('Units Sold')

plt.show()

# Line chart: Monthly sales by category

pivot\_monthly\_category\_sales.plot.line(marker='o', figsize=(15, 8))

plt.title('Monthly Sales by Category')

plt.xticks(rotation=45)

plt.ylabel('Units Sold')

plt.show()

