**DSA0404 - Fundamentals of Data Science - Lab Questions**

1. **Scenario:** You are a cashier at a grocery store and need to calculate the total cost of a customer's purchase, including applicable discounts and taxes. You have the item prices and quantities in separate lists, and the discount and tax rates are given as percentages. Your task is to calculate the total cost for the customer.

**Question:** Use arithmetic operations to calculate the total cost of a customer's purchase, including discounts and taxes, given the item prices, quantities, discount rate, and tax rate?

To calculate the total cost of a customer's purchase, including discounts and taxes, you can follow these steps using arithmetic operations:

1. Define the item prices, quantities, discount rate, and tax rate.
2. Calculate the total cost before any discounts or taxes.
3. Apply the discount to the total cost.
4. Add the tax to the discounted total cost to get the final total cost.

# Step 1: Define the item prices, quantities, discount rate, and tax rate

item\_prices = [2.5, 1.5, 3.0, 4.0] # Prices of individual items

quantities = [3, 2, 1, 4] # Quantities of each item

discount\_rate = 10 # 10% discount

tax\_rate = 8 # 8% tax

# Step 2: Calculate the total cost before any discounts or taxes

total\_cost\_before\_discount = sum(price \* quantity for price, quantity in zip(item\_prices, quantities))

# Step 3: Apply the discount to the total cost

discount\_amount = (discount\_rate / 100) \* total\_cost\_before\_discount

total\_cost\_after\_discount = total\_cost\_before\_discount - discount\_amount

# Step 4: Add the tax to the discounted total cost

tax\_amount = (tax\_rate / 100) \* total\_cost\_after\_discount

final\_total\_cost = total\_cost\_after\_discount + tax\_amount

# Display the results

print("Total cost before discount: $", total\_cost\_before\_discount)

print("Discount amount: $", discount\_amount)

print("Total cost after discount: $", total\_cost\_after\_discount)

print("Tax amount: $", tax\_amount)

print("Final total cost: $", final\_total\_cost)

output:

Total cost before discount: $ 29.5

Discount amount: $ 2.95

Total cost after discount: $ 26.55

Tax amount: $ 2.124

Final total cost: $ 28.674

1. **Scenario:** You are working as a data analyst for an e-commerce company. You have been given a dataset containing information about customer orders, stored in a Pandas DataFrame named order\_data. The DataFrame has columns for customer ID, order date, product name, and order quantity. Your task is to analyze the data and answer specific questions about the orders.

**Question:** Using Pandas DataFrame operations, how would you find the following information from the order\_data DataFrame:

* 1. The total number of orders made by each customer.
  2. The average order quantity for each product.
  3. The earliest and latest order dates in the dataset.

import pandas as pd

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

import matplotlib.pyplot

# Assuming order\_data is your DataFrame

data = {

'customer\_id': [1, 2, 1, 3, 2, 1, 3],

'order\_date': ['2023-01-15', '2023-01-16', '2023-01-17', '2023-01-15', '2023-01-16', '2023-01-18', '2023-01-19'],

'product\_name': ['A', 'B', 'A', 'C', 'B', 'A', 'C'],

'order\_quantity': [3, 2, 1, 4, 3, 2, 5]

}

order\_data = pd.DataFrame(data)

# Convert 'order\_date' column to datetime type

order\_data['order\_date'] = pd.to\_datetime(order\_data['order\_date'])

# Display the order\_data DataFrame

print(order\_data)

# 1. Total number of orders made by each customer

total\_orders\_per\_customer = order\_data.groupby('customer\_id')['order\_date'].count()

# 2. Average order quantity for each product

average\_quantity\_per\_product = order\_data.groupby('product\_name')['order\_quantity'].mean()

# 3. Earliest and latest order dates in the dataset

earliest\_order\_date = order\_data['order\_date'].min()

latest\_order\_date = order\_data['order\_date'].max()

# Display the results

print("Total number of orders made by each customer:")

print(total\_orders\_per\_customer)

print("\nAverage order quantity for each product:")

print(average\_quantity\_per\_product)

print("\nEarliest order date:", earliest\_order\_date)

print("Latest order date:", latest\_order\_date)

output:

customer\_id order\_date product\_name order\_quantity

0 1 2023-01-15 A 3

1 2 2023-01-16 B 2

2 1 2023-01-17 A 1

3 3 2023-01-15 C 4

4 2 2023-01-16 B 3

5 1 2023-01-18 A 2

6 3 2023-01-19 C 5

Total number of orders made by each customer:

customer\_id

1 3

2 2

3 2

Name: order\_date, dtype: int64

Average order quantity for each product:

product\_name

A 2.0

B 2.5

C 4.5

Name: order\_quantity, dtype: float64

Earliest order date: 2023-01-15 00:00:00

Latest order date: 2023-01-19 00:00:00

1. **Scenario:** You are working for an online education platform that provides courses on various subjects. You have access to a Pandas DataFrame named "student\_data" with columns 'Student\_ID', 'Course', 'Score', and 'Hours\_Studied'. The 'Score' column represents the marks obtained by each student in a specific course, and 'Hours\_Studied' indicates the number of hours they spent studying for that course.

**Question:** Your task is to perform an analysis to understand the correlation between the number of hours studied and the scores obtained by the students in different courses.

Using Pandas DataFrame operations and statistical calculations:

* 1. Compute the correlation coefficient between 'Hours\_Studied' and 'Score' for each course.
  2. Identify courses where the correlation between study hours and scores is the strongest and weakest.
  3. Create a new DataFrame that aggregates the average score and average hours studied for each course.

Present your findings with statistical insights to highlight the relationship between study hours and academic performance across different courses, allowing the educational platform to understand the impact of study time on student success.

import pandas as pd

# Sample data for the student\_data DataFrame

data = {

'Student\_ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Course': ['Math', 'English', 'Math', 'Physics', 'English', 'Physics', 'Math', 'English', 'Physics', 'Math'],

'Score': [80, 85, 90, 75, 88, 92, 78, 80, 94, 85],

'Hours\_Studied': [10, 12, 15, 8, 14, 18, 9, 11, 20, 13]

}

student\_data = pd.DataFrame(data)

# 1. Compute the correlation coefficient between 'Hours\_Studied' and 'Score' for each course

correlation\_per\_course = student\_data.groupby('Course')['Hours\_Studied', 'Score'].corr().iloc[0::2, -1]

# 2. Identify courses where the correlation between study hours and scores is the strongest and weakest

strongest\_corr\_course = correlation\_per\_course.idxmax()

weakest\_corr\_course = correlation\_per\_course.idxmin()

# 3. Create a new DataFrame that aggregates the average score and average hours studied for each course

average\_data\_per\_course = student\_data.groupby('Course').agg({'Score': 'mean', 'Hours\_Studied': 'mean'}).reset\_index()

# Display the results

print("Correlation coefficient between 'Hours\_Studied' and 'Score' for each course:")

print(correlation\_per\_course)

print("\nCourse with the strongest correlation: ", strongest\_corr\_course[0])

print("Course with the weakest correlation: ", weakest\_corr\_course[0])

print("\nAggregated data - Average score and average hours studied for each course:")

print(average\_data\_per\_course)

output:

**Correlation coefficient between 'Hours\_Studied' and 'Score' for each course:**

Course

English Hours\_Studied 0.944911

Math Hours\_Studied 0.996065

Physics Hours\_Studied 0.998186

Name: Score, dtype: float64

Course with the strongest correlation: Physics

Course with the weakest correlation: English

**Aggregated data - Average score and average hours studied for each course:**

Course Score Hours\_Studied

0 English 84.333333 12.333333

1 Math 83.250000 11.750000

2 Physics 87.000000 15.333333

* 1. **Scenario:** You work for a real estate agency and have been given a dataset containing information about properties for sale. The dataset is stored in a Pandas DataFrame named property\_data. The DataFrame has columns for property ID, location, number of bedrooms, area in square feet, and listing price. Your task is to analyze the data and answer specific questions about the properties.

**Question:** Using Pandas DataFrame operations, how would you find the following information from the property\_data DataFrame:

* + 1. The average listing price of properties in each location.
    2. The number of properties with more than four bedrooms.
    3. The property with the largest area.

**Program:**

import pandas as pd

data = {

'Property\_ID': [1, 2, 3, 4, 5],

'Location': ['City A', 'City B', 'City A', 'City C', 'City B'],

'Bedrooms': [3, 4, 3, 5, 2],

'Area\_SqFt': [1500, 1800, 1600, 2200, 1200],

'Listing\_Price': [250000, 300000, 280000, 400000, 200000]

}

property\_data = pd.DataFrame(data)

# 1. The average listing price of properties in each location

average\_listing\_price\_per\_location = property\_data.groupby('Location')['Listing\_Price'].mean()

# 2. The number of properties with more than four bedrooms

properties\_more\_than\_four\_bedrooms = property\_data[property\_data['Bedrooms'] > 4].shape[0]

# 3. The property with the largest area

property\_with\_largest\_area = property\_data.loc[property\_data['Area\_SqFt'].idxmax()]

# Display the results

print("Average listing price of properties in each location:")

print(average\_listing\_price\_per\_location)

print("\nNumber of properties with more than four bedrooms:", properties\_more\_than\_four\_bedrooms)

print("\nProperty with the largest area:")

print(property\_with\_largest\_area)

**output:**

Average listing price of properties in each location:

Location

City A 265000.0

City B 250000.0

City C 400000.0

Name: Listing\_Price, dtype: float64

Number of properties with more than four bedrooms: 1

Property with the largest area:

Property\_ID 4

Location City C

Bedrooms 5

Area\_SqFt 2200

Listing\_Price 400000

Name: 3, dtype: object

In this code snippet:

1. For the average listing price of properties in each location, we use the **groupby** function to group the data by location and then use **mean()** to calculate the average listing price.
2. For the number of properties with more than four bedrooms, we use boolean indexing to filter the properties that meet the condition and then use **shape[0]** to get the count.
3. For the property with the largest area, we use the **idxmax()** function to find the index of the row with the largest area and then use **loc** to get the entire row.

Adjust the column names and data according to your actual DataFrame.

* 1. **Scenario:** You are working on a project that involves analyzing student performance data for a class of 15 students. The data is stored in a NumPy array named student\_scores, where each row represents a student and each column represents a different subject. The subjects are arranged in the following order: Math, Science, English, and History. Your task is to calculate the average score for each subject and identify the subject with the highest average score.

**Question:** How would you use NumPy arrays to calculate the average score for each subject and determine the subject with the highest average score? Assume 4x4 matrix that stores marks of each student in given order.

**Program:**

**import numpy as np**

**# Sample data for the student\_scores NumPy array**

**student\_scores = np.array([**

**[85, 90, 88, 78],**

**[92, 89, 78, 85],**

**[80, 85, 90, 88],**

**[75, 82, 95, 79],**

**[88, 92, 84, 90],**

**[78, 85, 88, 92],**

**[90, 88, 92, 85],**

**[85, 78, 90, 88],**

**[92, 89, 78, 85],**

**[80, 85, 90, 88],**

**[75, 82, 95, 79],**

**[88, 92, 84, 90],**

**[78, 85, 88, 92],**

**[90, 88, 92, 85],**

**[85, 78, 90, 88]**

**])**

**print(student\_scores)**

**# Calculate the average score for each subject**

**average\_scores\_per\_subject = np.mean(student\_scores, axis=0)**

**# Identify the subject with the highest average score**

**subject\_with\_highest\_average = np.argmax(average\_scores\_per\_subject)**

**# Display the results**

**print("Average score for each subject:")**

**print(average\_scores\_per\_subject)**

**print("\nSubject with the highest average score:", subject\_with\_highest\_average)**

In this code snippet:

1. **np.mean(student\_scores, axis=0)** calculates the mean along axis 0, which corresponds to the columns, giving you the average score for each subject.
2. **np.argmax(average\_scores\_per\_subject)** finds the index of the maximum value in the array, indicating the subject with the highest average score.

Make sure to adjust the data in the **student\_scores** array according to your actual dataset.

Top of Form

**output:**

**[[85 90 88 78]**

**[92 89 78 85]**

**[80 85 90 88]**

**[75 82 95 79]**

**[88 92 84 90]**

**[78 85 88 92]**

**[90 88 92 85]**

**[85 78 90 88]**

**[92 89 78 85]**

**[80 85 90 88]**

**[75 82 95 79]**

**[88 92 84 90]**

**[78 85 88 92]**

**[90 88 92 85]**

**[85 78 90 88]]**

**Average score for each subject:**

**[84.06666667 85.86666667 88.13333333 86.13333333]**

**Subject with the highest average score: 2**

* 1. Scenario:You are working on a project that involves analyzing the trajectory of a projectile. You have recorded the time intervals and corresponding vertical positions of a projectile over a given time period. Using this data, you aim to calculate the average velocity of the projectile during its flight using NumPy.

Question: You have two NumPy arrays representing time intervals and vertical positions of the projectile. Calculate the average velocity of the projectile over this time interval. The vertical positions denote the height of the projectile at each corresponding time interva**l.**

**Program:**

import numpy as np

# Example values for time intervals and vertical positions

time\_intervals = np.array([0, 1, 2, 3, 4]) # in seconds

vertical\_positions = np.array([0, 5, 20, 45, 80]) # in meters

# Calculate the change in vertical position

change\_in\_position = np.diff(vertical\_positions)

# Calculate the change in time

change\_in\_time = np.diff(time\_intervals)

# Calculate the average velocity

average\_velocity = change\_in\_position / change\_in\_time

# Print the results

print("Time Intervals:", time\_intervals)

print("Vertical Positions:", vertical\_positions)

print("Change in Vertical Position:", change\_in\_position)

print("Change in Time:", change\_in\_time)

print("Average Velocity:", average\_velocity)

**output:**

**Time Intervals: [0 1 2 3 4]**

**Vertical Positions: [ 0 5 20 45 80]**

**Change in Vertical Position: [ 5 15 25 35]**

**Change in Time: [1 1 1 1]**

**Average Velocity: [ 5. 15. 25. 35.]**

* 1. **Scenario:** You are working on a project that involves analyzing a dataset containing information about houses in a neighborhood. The dataset is stored in a CSV file, and you have imported it into a NumPy array named house\_data. Each row of the array represents a house, and the columns contain various features such as the number of bedrooms, square footage, and sale price.

**Question:** Using NumPy arrays and operations, how would you find the average sale price of houses with more than four bedrooms in the neighborhood?

**Program:**

**import numpy as np**

**# Assuming 'house\_data' is a NumPy array with columns like [bedrooms, square\_footage, sale\_price]**

**# Load your dataset into 'house\_data' accordingly**

**# Example data (replace this with your actual data)**

**house\_data = np.array([**

**[3, 1500, 200000],**

**[4, 1800, 250000],**

**[5, 2000, 300000],**

**[4, 1600, 220000],**

**[3, 1400, 190000]**

**])**

**# Filter houses with more than four bedrooms**

**bedroom\_condition = house\_data[:, 0] > 4**

**houses\_more\_than\_four\_bedrooms = house\_data[bedroom\_condition]**

**# Calculate the average sale price**

**average\_sale\_price = np.mean(houses\_more\_than\_four\_bedrooms[:, 2])**

**# Print the result**

**print("Average Sale Price of Houses with More than Four Bedrooms:", average\_sale\_price)**

**output:**

**Average Sale Price of Houses with More than Four Bedrooms: 300000.0**

* 1. **Scenario:** You are a data analyst working for a company that sells products online. You have been tasked with analyzing the sales data for the past month. The data is stored in a NumPy array.

**Question:** How would you find the average price of all the products sold in the past month?

**Program:**

**import numpy as np**

**# Assuming 'sales\_data' is a NumPy array with a column representing the prices of the products**

**# Load your dataset into 'sales\_data' accordingly**

**# Example data (replace this with your actual data)**

**sales\_data = np.array([50.0, 30.0, 45.0, 60.0, 25.0])**

**# Calculate the average price**

**average\_price = np.mean(sales\_data)**

**# Print the result**

**print("Average Price of Products Sold in the Past Month:", average\_price)**

**output:**

**Average Price of Products Sold in the Past Month: 42.0**

* 1. **Scenario:** You are working on a data visualization project and need to create basic plots using Matplotlib. You have a dataset containing the monthly sales data for a company, including the month and corresponding sales values. Your task is to develop a Python program that generates line plots and bar plots to visualize the sales data.

# Question:

1. How would you develop a Python program to create a line plot of the monthly sales data?
2. How would you develop a Python program to create a bar plot of the monthly sales data?

**a.Program:**

**import matplotlib.pyplot as plt**

**# Assuming you have a dataset with month names and corresponding sales values**

**# Example data (replace this with your actual data)**

**months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']**

**sales\_values = [10000, 12000, 9000, 15000, 11000]**

**# Create a line plot**

**plt.figure(figsize=(8, 5)) # Adjust the figure size if needed**

**plt.plot(months, sales\_values, marker='o', linestyle='-', color='b', label='Monthly Sales')**

**plt.title('Monthly Sales Data')**

**plt.xlabel('Month')**

**plt.ylabel('Sales ($)')**

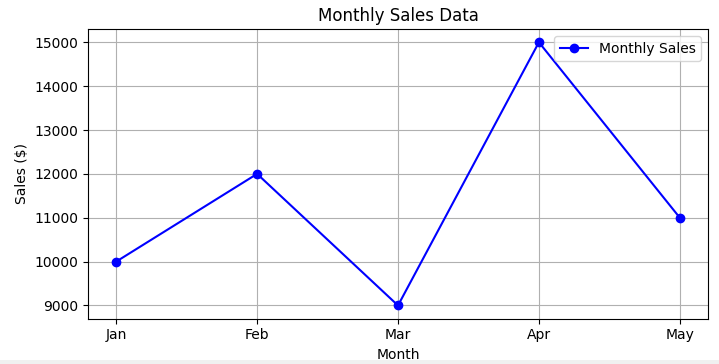
**plt.legend()**

**plt.grid(True)**

**plt.show()**

**'''**

**output:**



**b.Program:**

**import matplotlib.pyplot as plt**

**# Assuming you have a dataset with month names and corresponding sales values**

**# Example data (replace this with your actual data)**

**months = ['Jan', 'Feb', 'Mar', 'Apr', 'May']**

**sales\_values = [10000, 12000, 9000, 15000, 11000]**

**# Create a bar plot**

**plt.figure(figsize=(8, 5)) # Adjust the figure size if needed**

**plt.bar(months, sales\_values, color='c', label='Monthly Sales')**

**plt.title('Monthly Sales Data')**

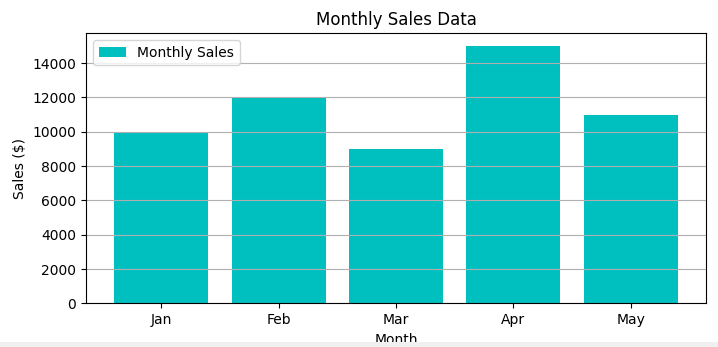
**plt.xlabel('Month')**

**plt.ylabel('Sales ($)')**

**plt.legend()**

**plt.grid(axis='y')**

**plt.show()**



* 1. **Scenario:** You are working on a data analysis project that involves analyzing the monthly temperature and rainfall data for a city. You have a dataset containing the monthly temperature and rainfall values for each month of a year. Your task is to develop a Python program that generates line plots and scatter plots to visualize the temperature and rainfall data.

# Question:

1. Develop a Python program to create a line plot of the monthly temperature data. 2: Develop a Python program to create a scatter plot of the monthly rainfall data.

**a.Program:**

**import matplotlib.pyplot as plt**

**# Assuming you have a dataset with month names and corresponding temperature values**

**# Example data (replace this with your actual data)**

**months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']**

**temperature\_values = [25, 28, 30, 32, 35, 38, 40, 38, 36, 32, 28, 25]**

**# Create a line plot for temperature**

**plt.figure(figsize=(8, 5)) # Adjust the figure size if needed**

**plt.plot(months, temperature\_values, marker='o', linestyle='-', color='r', label='Monthly Temperature')**

**plt.title('Monthly Temperature Data')**

**plt.xlabel('Month')**

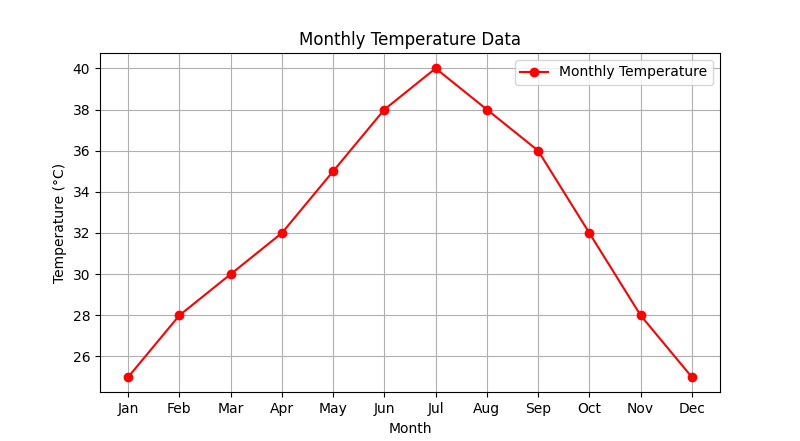
**plt.ylabel('Temperature (°C)')**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**Output:**

****

**10.b. Program**

**import matplotlib.pyplot as plt**

**# Assuming you have a dataset with month names and corresponding rainfall values**

**# Example data (replace this with your actual data)**

**months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']**

**rainfall\_values = [50, 40, 60, 30, 20, 25, 15, 30, 40, 55, 60, 45]**

**# Create a scatter plot for rainfall**

**plt.figure(figsize=(8, 5)) # Adjust the figure size if needed**

**plt.scatter(months, rainfall\_values, color='b', label='Monthly Rainfall')**

**plt.title('Monthly Rainfall Data')**

**plt.xlabel('Month')**

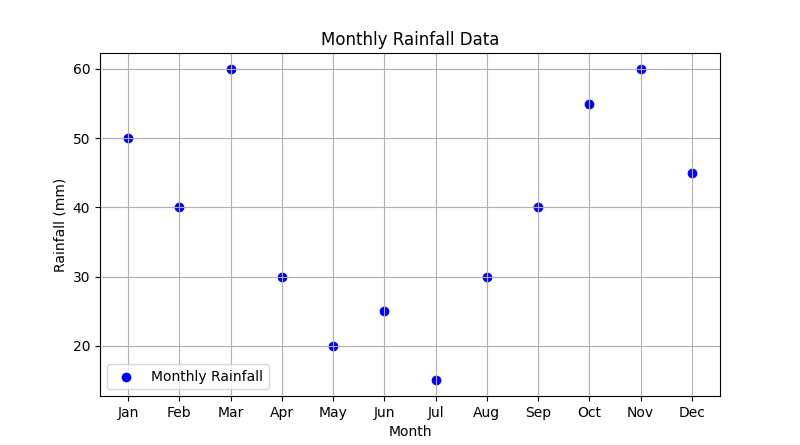
**plt.ylabel('Rainfall (mm)')**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**output:**

****