**Assignment 1**

**Name- Sneha Jayvant Kumbhar**

**PRN-22610006**

**Title-** To perform normalization of data (Min-Max, Z-Score and Decimal Scaling)

**Theory:**

In Data Mining, normalization is a preprocessing technique used to scale numeric data into a specific range without changing its meaning. This step is important because data in real-world datasets often have different scales (e.g., salary in thousands, age in tens), which can affect the performance of mining algorithms like clustering, classification, and neural networks.Normalization makes data more uniform, so that no single attribute dominates others just because of its scale.Only numeric attributes are normalized. Non-numeric columns are left unchanged.

Below are three commonly used normalization techniques:

### **1. Min-Max Normalization:** Min-Max normalization is a method that converts the original data values into a fixed range, usually between 0 and 1. It does this by taking the minimum and maximum values in a column and adjusting all values in proportion to that range. This technique keeps the original relationships between values but changes the scale. It is commonly used when we want to preserve patterns while fitting all data into a specific range.

### **2. Z-Score Normalization (Standardization):** Z-score normalization is used to center the data around zero. It does this by calculating how far each data value is from the average value of the column. The result shows how many standard units (or standard deviations) each value is above or below the mean. This method is useful when the data has outliers or when attributes have different scales or units. It helps make the data more balanced for analysis.

### **3. Decimal Scaling Normalization:**Decimal scaling normalization is a simple method that moves the decimal point of the data values. It divides all the values by a power of 10, depending on the largest number in the dataset, to bring all values into a smaller range such as -1 to 1. It is useful for reducing the size of large values while keeping the same data pattern.

**Formulas:**

### **1. Min-Max Normalization**

x′=(x−min⁡(x))/(max⁡(x)−min⁡(x))

Where:

* x = original data value
* min⁡(x) = minimum value in the attribute
* max⁡(x) = maximum value in the attribute
* x′ = normalized value between 0 and 1

**2. Z-Score Normalization (Standardization)**

x′=x−μ​/σ

Where:

* x = original data value
* μ = mean (average) of the attribute
* σ = standard deviation of the attribute
* x′ = standardized value (centered around 0)

**3. Decimal Scaling Normalization**

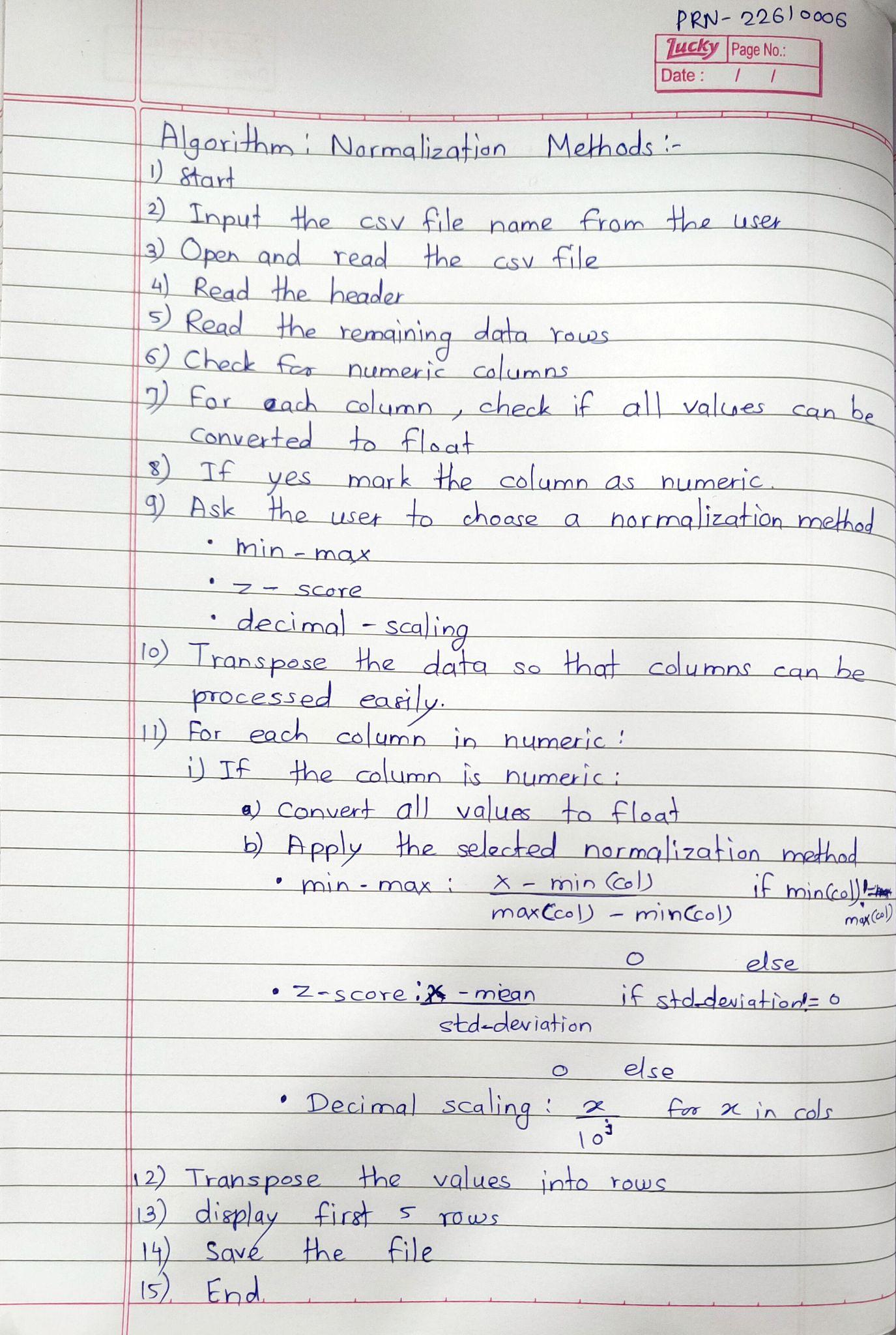
**Formula:**

x′=x/ 10^j​

Where:

* x = original data value
* j = smallest integer such that all x′′ are between -1 and 1
* x′ = normalized value

**Algorithm:**

****

**Python Implementation:**

import csv

import math

def is\_float(s):

try:

float(s)

return True

except ValueError:

return False

file\_name = input("Enter the CSV file name: ")

try:

with open(file\_name, 'r') as f:

reader = csv.reader(f)

header = next(reader)

rows = [row for row in reader]

except FileNotFoundError:

print("File not found.")

exit()

num\_cols = []

for col\_index in range(len(header)):

if all(is\_float(row[col\_index]) for row in rows):

num\_cols.append(col\_index)

method = input("Enter normalization method (min-max, z-score, decimal-scaling): ").strip().lower()

columns = list(zip(\*rows))

normalized\_columns = []

for i in range(len(columns)):

col = columns[i]

if i in num\_cols:

col = list(map(float, col))

if method == "min-max":

min\_val = min(col)

max\_val = max(col)

norm\_col = [(x - min\_val) / (max\_val - min\_val) if max\_val != min\_val else 0.0 for x in col]

elif method == "z-score":

mean = sum(col) / len(col)

std = math.sqrt(sum((x - mean)\*\*2 for x in col) / len(col))

norm\_col = [(x - mean) / std if std != 0 else 0.0 for x in col]

elif method == "decimal-scaling":

max\_val = max(abs(x) for x in col)

j = len(str(int(max\_val))) if max\_val != 0 else 1

norm\_col = [x / (10\*\*j) for x in col]

else:

print("Invalid normalization method.")

exit()

normalized\_columns.append([str(round(val, 6)) for val in norm\_col])

else:

normalized\_columns.append(list(col))

normalized\_rows = list(zip(\*normalized\_columns))

print(f"\nNormalized Data (first 5 rows):")

print(header)

for row in normalized\_rows[:5]:

print(row)

save = input("Do you want to save the result as a CSV? (yes/no): ").strip().lower()

if save == "yes":

out\_file = f"{method}\_normalized.csv"

with open(out\_file, 'w', newline='') as f:

writer = csv.writer(f)

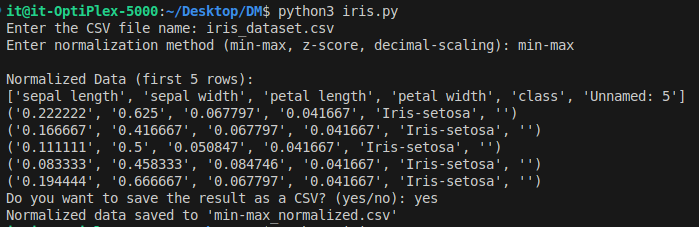
writer.writerow(header)

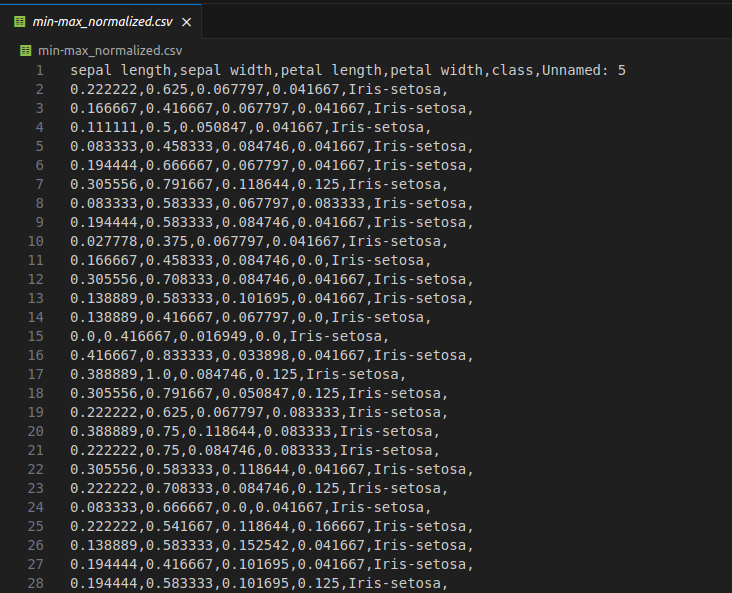
writer.writerows(normalized\_rows)

print(f"Normalized data saved to '{out\_file}'")

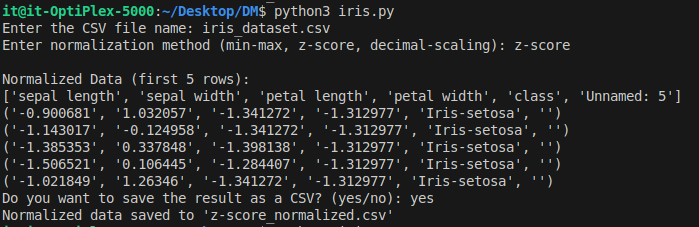
**Screenshots of Output:**

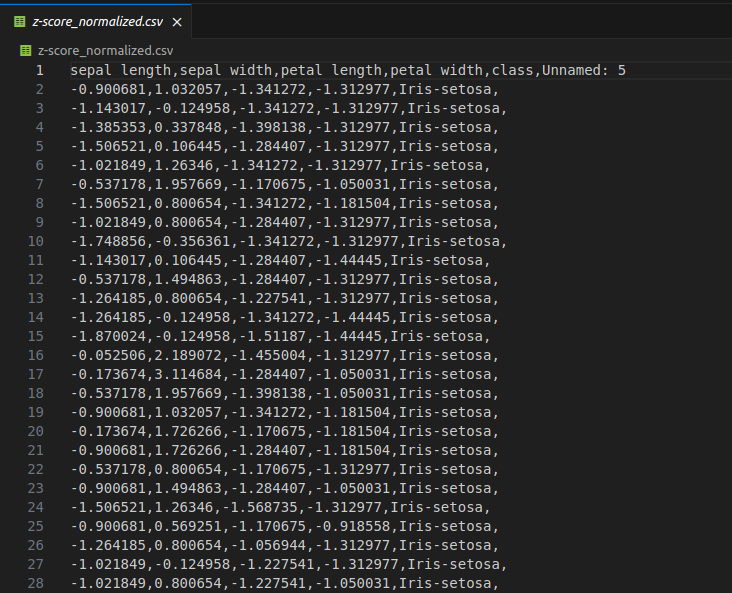
**Min- Max Normalization Output :**



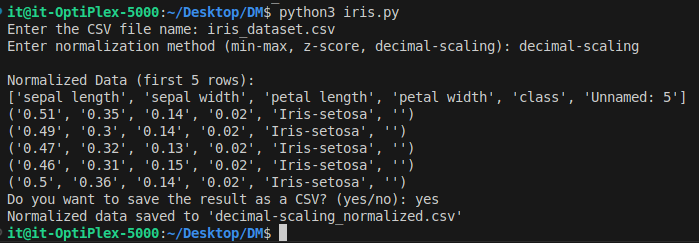


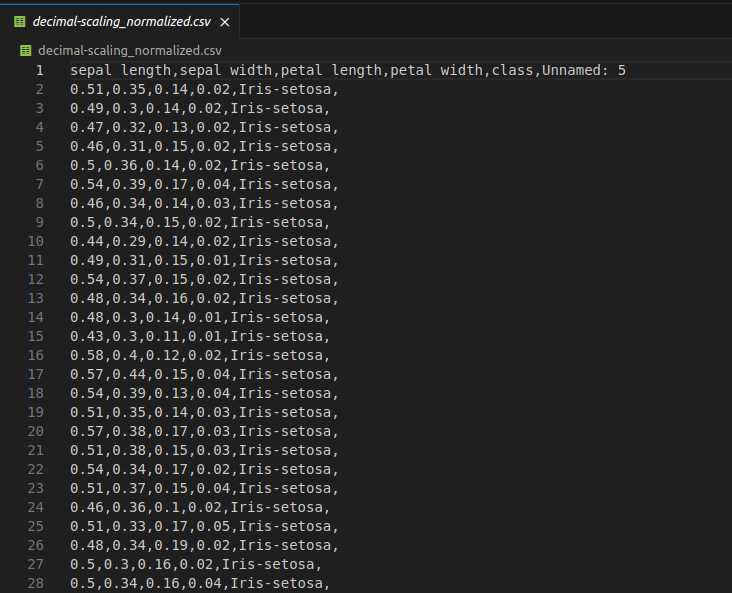
**Z- Score Normalization Output :**





**Decimal Scaling Output:**





**Conclusion:**

This experiment helped in understanding the importance of normalization in data mining and how different techniques like Min-Max, Z-Score, and Decimal Scaling are used to scale numeric data. Normalization brings all values to a similar range, which makes data more suitable for mining algorithms such as clustering and classification. Min-Max scales data between 0 and 1, Z-Score centers the data around the mean, and Decimal Scaling reduces large numbers by moving the decimal point. These techniques help improve the accuracy and performance of data analysis.

**Assignment 2**

**Name- Sneha Jayvant Kumbhar**

**PRN-22610006**

**Title-** To perform Data Cube operations (Slice, Dice, Roll-Up,Dril-Down,Pivot)

**Theory:**

### **OLAP (Online Analytical Processing)**

OLAP is a data analysis technology that allows users to examine large volumes of data from multiple perspectives. It is commonly used in business decision-making and reporting systems.

### **Useful for:**

* Helps summarize complex data
* Supports decision-making with analytical insights
* Provides fast answers to analytical queries
* Allows users to interactively explore data

### **Common OLAP Operations:**

### **1. Slice** The slice operation selects a single layer (or slice) of the data cube by fixing a value for one dimension. It is used to view data across the remaining dimensions while keeping one fixed.

### **Example:** From a 3D cube with dimensions **Region, Product, and Time**, selecting only the records where Region = 'India' is a slice.

### **2. Dice** The dice operation selects a sub-cube by choosing specific values for multiple dimensions. It is useful to view a more focused portion of data by applying multiple filters.

**Example:** Select all records where Region = 'USA' and Product = 'Laptop'.

**3. Roll-up** Roll-up performs aggregation along a dimension. It summarizes the data by climbing up a concept hierarchy. It is used to get summarized or higher-level information from detailed data

**Example:** From daily sales → monthly sales → yearly sales.  
 Or from city → state → country.

**4. Drill-Down** Drill-down is the opposite of roll-up. It provides more detailed data by going down the hierarchy. Its purpose is to get a more granular view of the data for analysis.

**Example:** From yearly sales → monthly sales → daily sales.

### **5. Pivot (Rotation)** Pivot reorients the multidimensional view of data — i.e., it rotates the data axes to provide a new perspective. It is very useful for analyzing the data in a more understandable or insightful layout.

**Example:** Change rows to columns:  
 Original: Regions as rows, Products as columns  
 After Pivot: Products as rows, Regions as columns

**Python Implementation:**

**def read\_csv(filename):**

**records = []**

**try:**

**with open(filename, 'r') as file:**

**header = next(file) # Skip header**

**for line in file:**

**parts = line.strip().split(',')**

**if len(parts) == 4:**

**record = {**

**'Date': parts[0],**

**'Region': parts[1],**

**'Product': parts[2],**

**'Sales': int(parts[3])**

**}**

**records.append(record)**

**except FileNotFoundError:**

**print("File not found!")**

**return records**

**def print\_record(r):**

**print(f"{r['Date']} | {r['Region']} | {r['Product']} | {r['Sales']}")**

**def slice\_operation(data, field, value):**

**print(f"\nSlice: {field} = {value} ")**

**for r in data:**

**if r[field].lower() == value.lower():**

**print\_record(r)**

**def dice\_operation(data, filters):**

**print("\nDice")**

**for r in data:**

**match = all(r[key].lower() == value.lower() for key, value in filters.items())**

**if match:**

**print\_record(r)**

**def rollup\_operation(data, group\_field):**

**print(f"\nRoll up: (Total Sales : {group\_field} )")**

**totals = {}**

**for r in data:**

**key = r[group\_field]**

**totals[key] = totals.get(key, 0) + r['Sales']**

**for key, total in totals.items():**

**print(f"{key} → {total}")**

**def drilldown\_operation(data):**

**print("\nDrill-down")**

**for r in data:**

**print\_record(r)**

**def pivot\_operation(data, row\_field, col\_field):**

**print(f"\nPivot: {row\_field} vs {col\_field}")**

**pivot = {}**

**col\_values = set()**

**for r in data:**

**row = r[row\_field]**

**col = r[col\_field]**

**sales = r['Sales']**

**col\_values.add(col)**

**if row not in pivot:**

**pivot[row] = {}**

**pivot[row][col] = pivot[row].get(col, 0) + sales**

**col\_values = sorted(col\_values)**

**print(row\_field, \*col\_values, sep='\t')**

**for row in pivot:**

**print(row, end='\t')**

**for col in col\_values:**

**print(pivot[row].get(col, 0), end='\t')**

**print()**

**def main():**

**filename = input("Enter CSV file name: ")**

**data = read\_csv(filename)**

**if not data:**

**print("No data loaded")**

**return**

**while True:**

**print("\n\* OLAP Menu \*")**

**print("1. Original Data")**

**print("2. Slice")**

**print("3. Dice")**

**print("4. Roll-Up")**

**print("5. Drill-Down")**

**print("6. Pivot")**

**print("0. Exit")**

**choice = input("Enter your choice: ")**

**if choice == '1':**

**print("\nOriginal Data ")**

**for r in data:**

**print\_record(r)**

**elif choice == '2':**

**field = input("Enter field to slice by (Date/Region/Product): ")**

**value = input(f"Enter value for {field}: ")**

**slice\_operation(data, field, value)**

**elif choice == '3':**

**num\_filters = int(input("How many filters? "))**

**filters = {}**

**for \_ in range(num\_filters):**

**field = input("Enter field name (Date/Region/Product): ")**

**value = input(f"Enter value for {field}: ")**

**filters[field] = value**

**dice\_operation(data, filters)**

**elif choice == '4':**

**field = input("Enter field to group by (Region/Product/Date): ")**

**rollup\_operation(data, field)**

**elif choice == '5':**

**drilldown\_operation(data)**

**elif choice == '6':**

**row\_field = input("Enter row field: ")**

**col\_field = input("Enter column field: ")**

**pivot\_operation(data, row\_field, col\_field)**

**elif choice == '0':**

**print("Exiting...")**

**break**

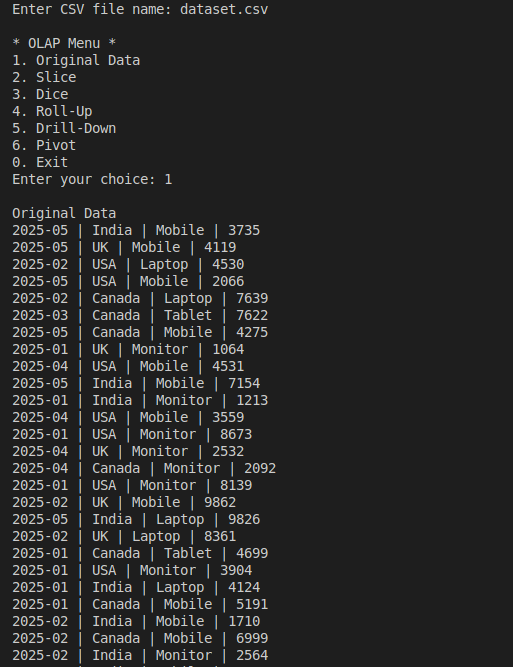
**else:**

**print("Invalid choice")**

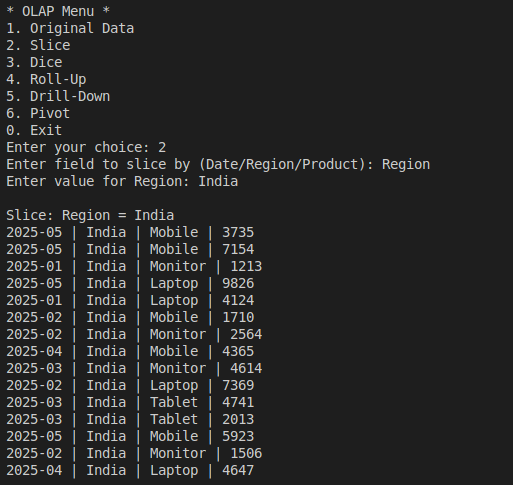
**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

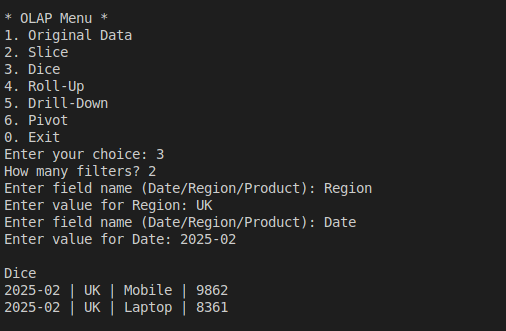
**Output:**

****

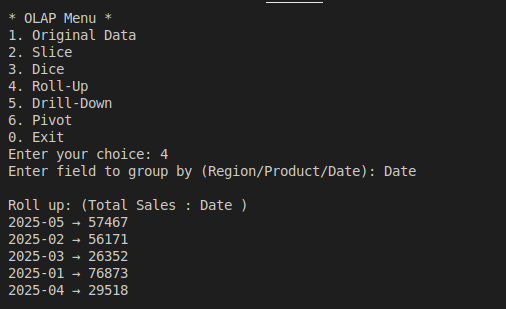
**1.Slice:**

****

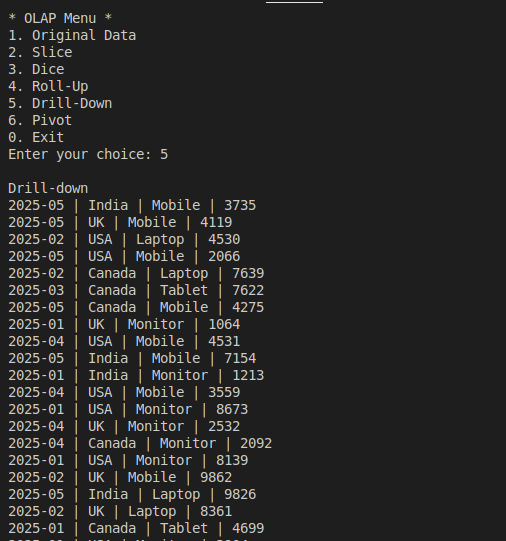
**2.Dice:**

****

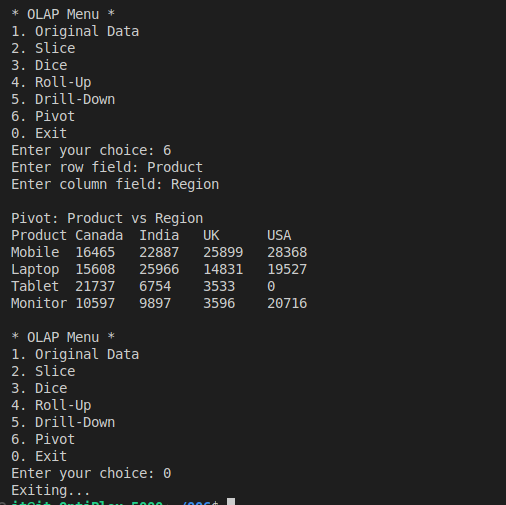
**3.Roll-Up:**

****

**4.Drill-Down:**

****

**5.Pivot:**

****

**Conclusion:**

In this experiment, I explored and implemented various OLAP (Online Analytical Processing) operations such as Slice, Dice, Roll-Up, Drill-Down, and Pivot on multidimensional data stored in a CSV file. By simulating these operations programmatically, I gained a clear understanding of how OLAP helps in summarizing, filtering, and visualizing large datasets from different perspectives.

**Assignment 3**

**Name- Sneha Jayvant Kumbhar**

**PRN-22610006**

**Title-** Find t and d weight of data

**Theory-**

### **1. T-weight (Transactional Weight)**

Definition:  
 T-weight measures the relative importance of an item within a specific class. It shows how much a particular item contributes to representing that class when compared to all other items in the same class.

Formula:

T-weight(i, C) = (Frequency of item i in class C) / (Total frequency of all items in class C)

Explanation:

* Numerator: Number of times item *i* appears in class *C*.
* Denominator: Sum of frequencies of all items in class *C*.
* Range: 0 to 1 (or 0% to 100%).
* Interpretation:  
  + High T-weight: Item is highly representative of the class.
  + Low T-weight: Item is less common within the class.

Purpose:

* Determines the representativeness of an item within its own class.
* Helps in identifying the most characteristic features of a class.

### **2. D-weight (Discrimination Weight)**

Definition:  
 D-weight measures how strongly an item is associated with one class compared to all other classes. It tells us how uniquely an item belongs to a specific class.

Formula:

D-weight(i, C) = (Frequency of item i in class C) / (Total frequency of item i across all classes)

Explanation:

* Numerator: Number of times item *i* appears in class *C*.
* Denominator: Number of times item *i* appears in the entire dataset (across all classes).
* Range: 0 to 1 (or 0% to 100%).
* Interpretation:  
  + High D-weight: Item appears mostly in this class (unique indicator).
  + Low D-weight: Item is spread across multiple classes (not unique).

Purpose:

* Measures the uniqueness of an item for a specific class.
* Helps in identifying discriminating features that separate one class from another.

### **3. Relationship Between T-weight and D-weight**

* T-weight focuses on *representation within a class*.
* D-weight focuses on *uniqueness across classes*.
* High T-weight + High D-weight: Item is both representative and unique, making it an excellent class identifier.

### **4. Usage in Data Mining**

* Classification: Selecting features that represent a class and distinguish it from others.
* Pattern Discovery: Finding items that are both common and unique to certain classes.
* Association Rule Mining: Strengthening rules by choosing items with high representativeness and uniqueness.

**Python Implementation:**

**import csv**

**class Student:**

**def \_\_init\_\_(self, name, ball\_pen, gel\_pen):**

**self.name = name**

**self.ball\_pen = float(ball\_pen)**

**self.gel\_pen = float(gel\_pen)**

**students = []**

**with open("students\_pens.csv", "r") as file:**

**reader = csv.reader(file)**

**next(reader)**

**for row in reader:**

**name, b, g = row**

**students.append(Student(name, b, g))**

**total\_ball = sum(s.ball\_pen for s in students)**

**total\_gel = sum(s.gel\_pen for s in students)**

**total\_all = sum(s.ball\_pen + s.gel\_pen for s in students)**

**with open("output.csv", "w", newline="") as outfile:**

**writer = csv.writer(outfile)**

**writer.writerow([**

**"Student", "BallPen", "T-Weight(Ball)", "D-Weight(Ball)",**

**"GelPen", "T-Weight(Gel)", "D-Weight(Gel)",**

**"Both", "T-Weight(Both)", "D-Weight(Both)"**

**])**

**for s in students:**

**total\_student = s.ball\_pen + s.gel\_pen**

**t\_weight\_ball = (s.ball\_pen / total\_ball) \* 100 if total\_ball else 0**

**d\_weight\_ball = (s.ball\_pen / total\_student) \* 100 if total\_student else 0**

**t\_weight\_gel = (s.gel\_pen / total\_gel) \* 100 if total\_gel else 0**

**d\_weight\_gel = (s.gel\_pen / total\_student) \* 100 if total\_student else 0**

**both\_count = total\_student**

**t\_weight\_both = (both\_count / total\_all) \* 100 if total\_all else 0**

**d\_weight\_both = 100.0**

**writer.writerow([**

**s.name, s.ball\_pen, round(t\_weight\_ball, 2), round(d\_weight\_ball, 2),**

**s.gel\_pen, round(t\_weight\_gel, 2), round(d\_weight\_gel, 2),**

**both\_count, round(t\_weight\_both, 2), round(d\_weight\_both, 2)**

**])**

**writer.writerow([**

**"Total", total\_ball, 100.0, 100.0,**

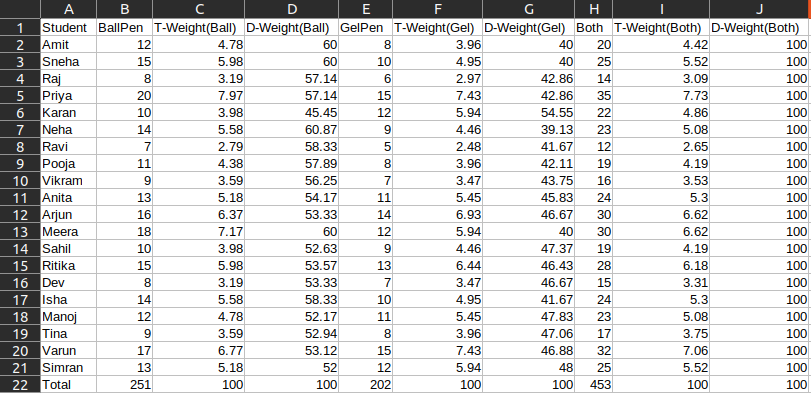
**total\_gel, 100.0, 100.0,**

**total\_all, 100.0, 100.0**

**])**

**print("Output saved to output.csv")**

**Output:**

****

**Conclusion:**

In this experiment, I calculated the Transactional Weight (T-weight) and Discrimination Weight (D-weight) for given data to understand the representativeness and uniqueness of attribute values in different classes. T-weight helped identify how strongly an item represents a specific class, while D-weight highlighted the distinctiveness of that item compared to other classes. By analyzing both measures together, I can effectively select features that are both common within a class and unique across classes, which is valuable for classification, pattern discovery, and association rule mining in data mining applications.

**Assignment 4**

**Name- Sneha Jayvant Kumbhar**

**PRN-22610006**

**Title-**Find 5 no summary of dataset

**Theory-**

A box plot gives a five-number summary of a set of data which is-

**Minimum (Lower Extreme)**

* The smallest data value that is not considered an outlier.
* Shown at the end of the left whisker.

**First Quartile (Q1 / 25th percentile)**

* The value below which 25% of the data fall.
* Represented by the left edge of the box.

**Median (Q2 / 50th percentile)**

* The middle value of the dataset.
* Shown as a line inside the box.

**Third Quartile (Q3 / 75th percentile)**

* The value below which 75% of the data fall.
* Represented by the right edge of the box.

**Maximum (Upper Extreme)**

* The largest data value that is not considered an outlier.
* Shown at the end of the right whisker.

**Interquartile Range (IQR)**

* Distance between Q1 and Q3.
* Represents the spread of the middle 50% of data.
* Width of the box = Q3 – Q1.

**Whiskers**

* Lines extending from Q1 to the minimum within bounds and from Q3 to the maximum within bounds.
* They show the range of data excluding outliers.

**Outliers**

* Data points lying outside the whiskers:  
  + Lower Outlier: < Q1 – 1.5 × IQR
  + Upper Outlier: > Q3 + 1.5 × IQR
* Plotted as individual points (dots or stars).

Steps to Summarize a Data Set with a Five-Number Summary

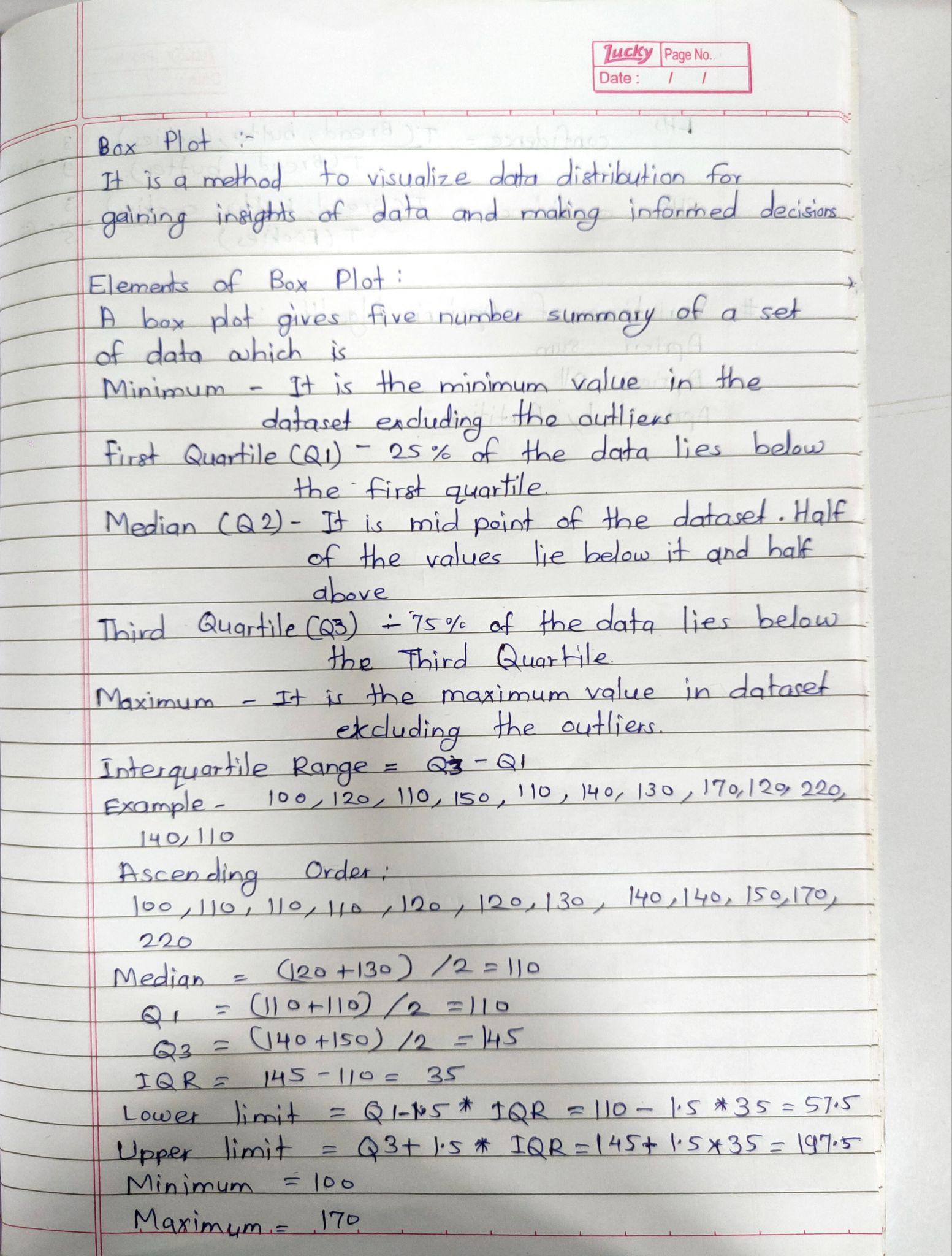
**Step 1:** Order the values from least to greatest.

**Step 2:** Determine the minimum and maximum of the data set by identifying the lowest and highest values.

**Step 3:** Find the median of the data set. Separate the lower half from the upper half.

**Step 4:** Find the first and third quartiles by finding the median of the lower half and upper half of the data.

**Step 5:** Summarize the data set by stating the minimum, first quartile, median, third quartile, and maximum .



**Python Implementation:**

import csv

def median(v):

v = sorted(v)

n = len(v)

if n % 2 == 0:

return (v[n//2 - 1] + v[n//2]) / 2.0

else:

return v[n//2]

def five\_number\_summary\_with\_outliers(data):

data = sorted(data)

Q2 = median(data)

n = len(data)

if n % 2 == 0:

lower\_half = data[:n//2]

upper\_half = data[n//2:]

else:

lower\_half = data[:n//2]

upper\_half = data[n//2+1:]

Q1 = median(lower\_half)

Q3 = median(upper\_half)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

lower\_whisker = min([x for x in data if x >= lower\_bound], default=data[0])

upper\_whisker = max([x for x in data if x <= upper\_bound], default=data[-1])

return {

"Q1": Q1,

"Median (Q2)": Q2,

"Q3": Q3,

"IQR": IQR,

"Lower Whisker": lower\_whisker,

"Upper Whisker": upper\_whisker

}

ball\_pens = []

gel\_pens = []

with open("student\_pens.csv", "r") as file:

reader = csv.DictReader(file)

for row in reader:

ball\_pens.append(int(row["BallPen"]))

gel\_pens.append(int(row["GelPen"]))

print("BallPen Summary:")

for k, v in five\_number\_summary\_with\_outliers(ball\_pens).items():

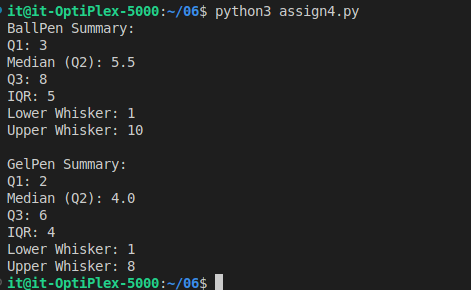
print(f"{k}: {v}")

print("\nGelPen Summary:")

for k, v in five\_number\_summary\_with\_outliers(gel\_pens).items():

print(f"{k}: {v}")

**Output:**



**Conclusion:**

In this experiment, I calculated the 5-number summary, interquartile range (IQR), whiskers, and outliers to analyze the distribution of a dataset. The results clearly showed how the minimum, quartiles, median, and maximum provide a compact summary of the data, while the IQR helps measure variability in the central portion. By identifying whiskers and detecting outliers, we understood how extreme values can be separated from the main data distribution. Overall, this method proves effective in data mining for summarizing datasets, detecting anomalies, and supporting further statistical analysis or visualization using boxplots.