Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-3**

Class: BE (A) **Branch: AI&DS**

Assignment No.: 01

Output:

[*]:
```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler
class FactorialServer:
    def calculate_factorial(self, n):
        if n < 0:
            raise ValueError("Input must be a non-negative integer.")
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result
# Restrict to a particular path.
class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)
# Create server
with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()
    # Register the FactorialServer class
    server.register_instance(FactorialServer())
    print("FactorialServer is ready to accept requests.")
    # Run the server's main loop
    server.serve_forever()
```

```
FactorialServer is ready to accept requests.
127.0.0.1 - - [19/Mar/2025 13:30:04] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [19/Mar/2025 13:30:28] "POST /RPC2 HTTP/1.1" 200 -
```

[2]:
```
import xmlrpc.client
# Create an XML-RPC client
with xmlrpc.client.ServerProxy("http://localhost:8000/RPC2") as proxy:
    try:
        # Replace 5 with the desired integer value
        input_value = 10
        result = proxy.calculate_factorial(input_value)
        print(f"Factorial of {input_value} is: {result}")
    except Exception as e:
        print(f"Error: {e}")
```

```
Factorial of 10 is: 3628800
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-3**

Class: BE (A) **Branch: AI&DS**

Assignment No.: 02

Output:

[*]:
```
import Pyro4
@Pyro4.expose
class StringConcatenationServer:
    def concatenate_strings(self, str1, str2):
        result = str1 + str2
        return result

def main():
    daemon = Pyro4.Daemon()  # Create a Pyro daemon
    ns = Pyro4.locateNS()  # Locate the Pyro nameserver

    # Create an instance of the server class
    server = StringConcatenationServer()

    # Register the server object with the Pyro nameserver
    uri = daemon.register(server)
    ns.register("string.concatenation", uri)

    print("Server URI:", uri)

    with open("server_uri.txt", "w") as f:
        f.write(str(uri))

    daemon.requestLoop()

if __name__ == "__main__":
    main()
```

```
Server URI: PYRO:obj_b133b2b5effa4bf9993037d813ead7d7@localhost:52498
```

[5]:
```
import Pyro4

def main():
    with open("server_uri.txt", "r") as f:
        uri = f.read()

    server = Pyro4.Proxy(uri)  # Connect to the remote server

    str1 = input("Enter the first string: ")
    str2 = input("Enter the second string: ")

    result = server.concatenate_strings(str1, str2)

    print("Concatenated Result:", result)

if __name__ == "__main__":
    main()
```

```
Enter the first string:  yash
Enter the second string:  gawade
Concatenated Result: yashgawade
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-3**

Class: BE (A) **Branch: AI&DS**

Assignment No.: 03

Output:

3 > word_mapper.py > ...

```python
#!/usr/bin/env python3
import sys

# Read input line by line
for line in sys.stdin:
    line = line.strip()
    words = line.split()
    for word in words:
        # Emit each word with count 1
        print(f"{word}\t1")
```

3 > word_reducer.py > ...

```python
#!/usr/bin/env python3
import sys

current_word = None
current_count = 0

for line in sys.stdin:
    word, count = line.strip().split('\t')
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print(f"{current_word}\t{current_count}")
        current_word = word
        current_count = count

if current_word:
    print(f"{current_word}\t{current_count}")
```

3 > character_mapper.py > ...

```python
#!/usr/bin/env python3
import sys

# Read input line by line
for line in sys.stdin:
    line = line.strip()
    for char in line:
        # Emit each character with count 1
        print(f"{char}\t1")
```

3 > character_reducer.py > ...

```python
import sys

current_char = None
current_count = 0

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue  # skip empty lines
    parts = line.split('\t')
    if len(parts) != 2:
        continue  # skip malformed lines
    char, count = parts
    count = int(count)
    if char == current_char:
        current_count += count
    else:
        if current_char:
            print(f"{current_char}\t{current_count}")
        current_char = char
        current_count = count

if current_char:
    print(f"{current_char}\t{current_count}")
```

3 > input.txt

```
Hello world
Hadoop is fun
Hello Hadoop
```

3 > char_count_output.txt

```
a	2
d	3
e	2
f	1
H	4
i	1
l	5
n	1
o	7
p	2
r	1
s	1
u	1
w	1
```

```
D:\yash\Projects\sem 8 practical\CL-3\3>type input.txt | python character_mapper.py | sort | python character_reducer.py > char_count_output.txt

D:\yash\Projects\sem 8 practical\CL-3\3>type input.txt | python word_mapper.py | sort | python word_reducer.py > word_count_output.txt
```

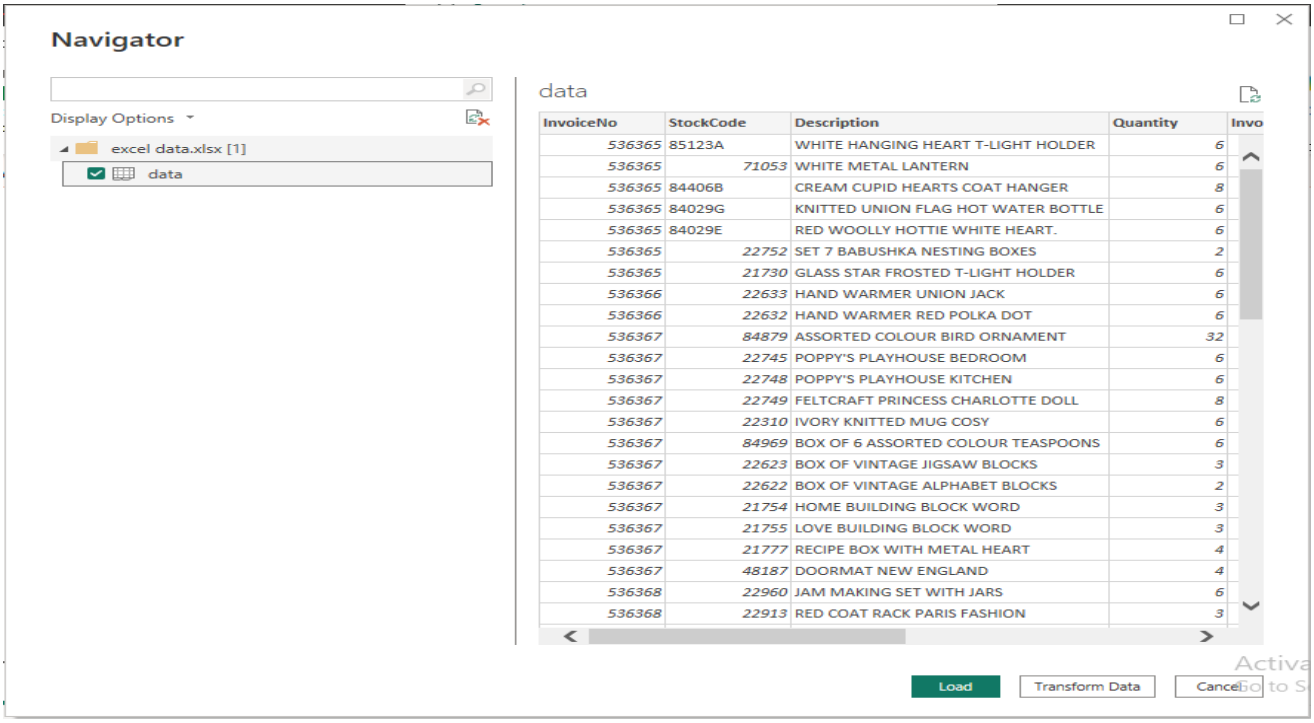
Name: Yash Laxman Gawade

Roll No.: 21140 Subject: CL-4

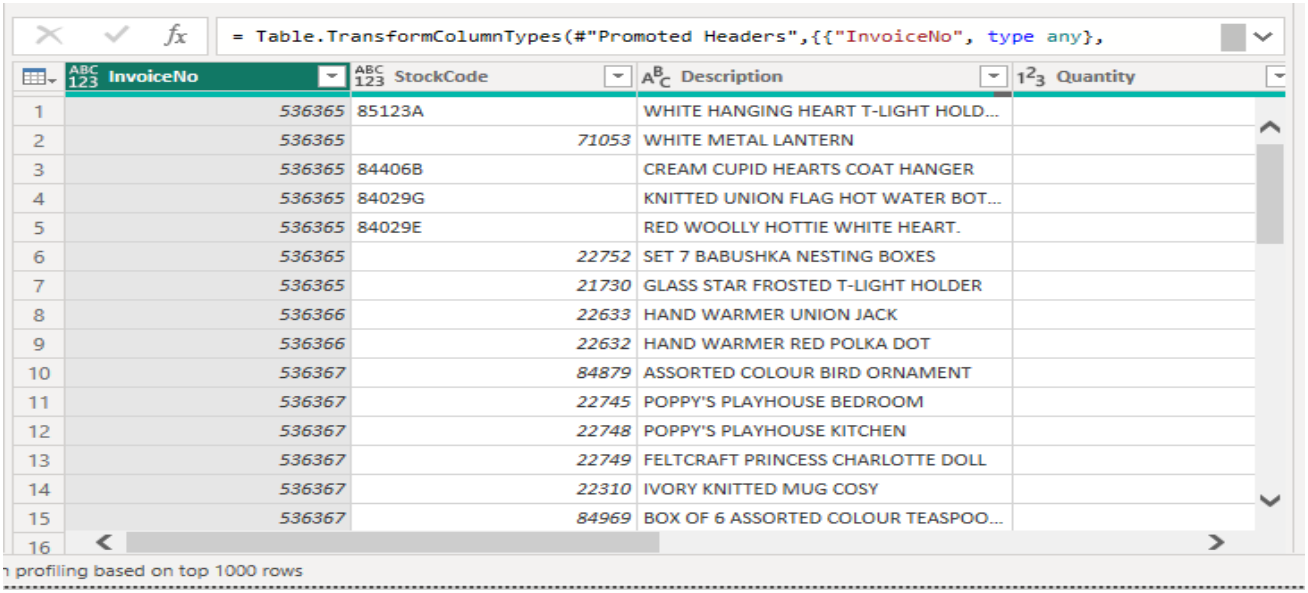
Class: BE (A) Branch: AI&DS

Assignment No.: 01

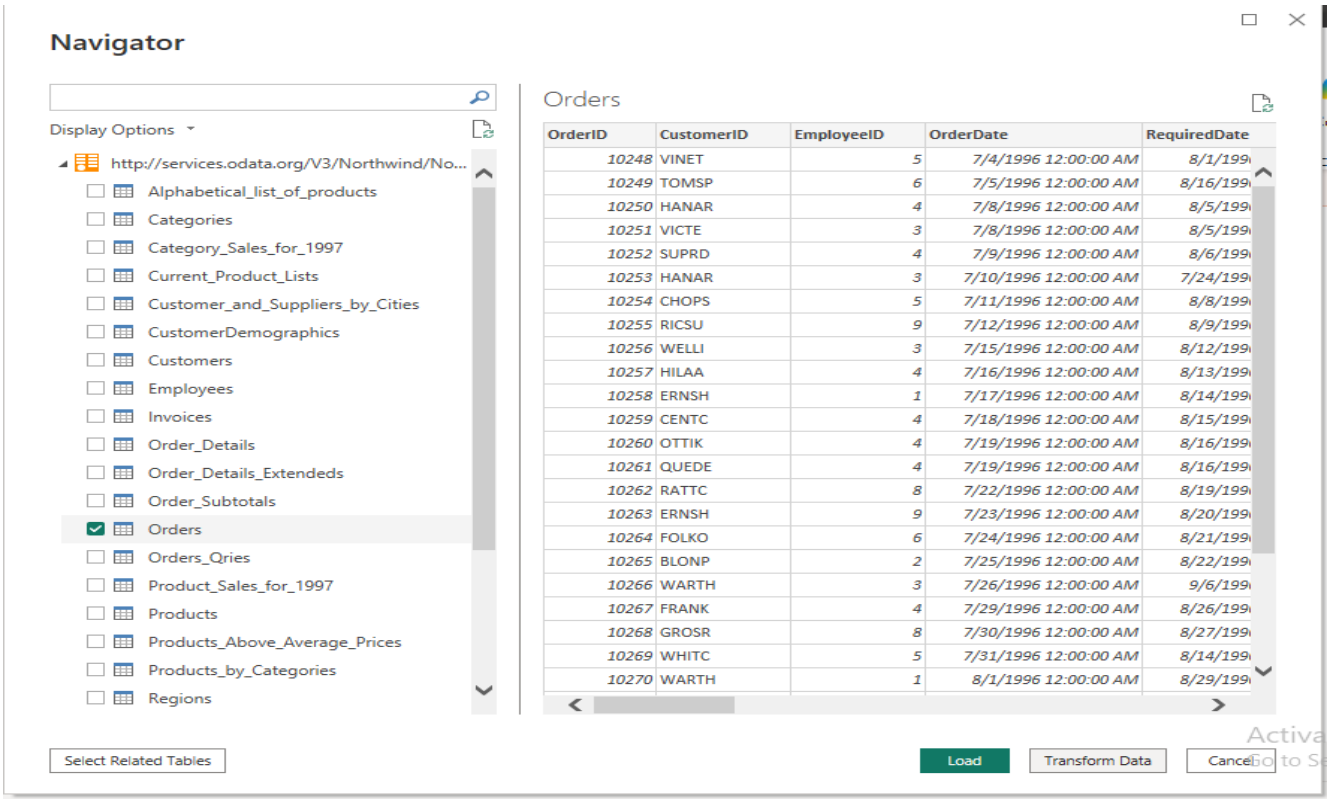
Output:

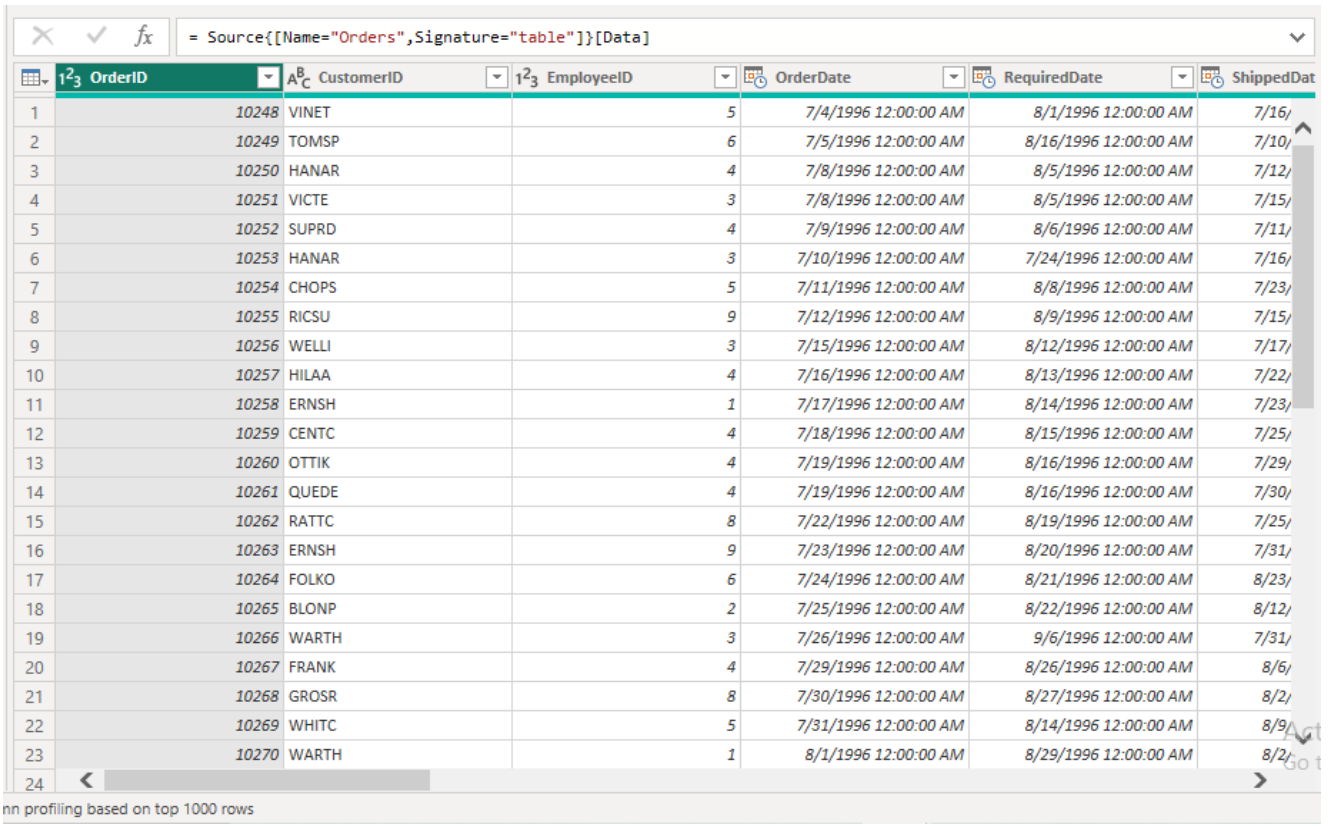


InvoiceNo	StockCode	Description	Quantity	Invo
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	
536365	71053	WHITE METAL LANTERN	6	
536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	
536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	
536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	
536365	22752	SET 7 BABUSHKA NESTING BOXES	2	
536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	
536366	22633	HAND WARMER UNION JACK	6	
536366	22632	HAND WARMER RED POLKA DOT	6	
536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	
536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	
536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	
536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	
536367	22310	IVORY KNITTED MUG COSY	6	
536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	
536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	
536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	
536367	21754	HOME BUILDING BLOCK WORD	3	
536367	21755	LOVE BUILDING BLOCK WORD	3	
536367	21777	RECIPE BOX WITH METAL HEART	4	
536367	48187	DOORMAT NEW ENGLAND	4	
536368	22960	JAM MAKING SET WITH JARS	6	
536368	22913	RED COAT RACK PARIS FASHION	3	



	InvoiceNo	StockCode	Description	Quantity
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLD...	
2	536365	71053	WHITE METAL LANTERN	
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOT...	
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	
8	536366	22633	HAND WARMER UNION JACK	
9	536366	22632	HAND WARMER RED POLKA DOT	
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	
14	536367	22310	IVORY KNITTED MUG COSY	
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOO...	
16				


Navigator
Display Options
http://services.odata.org/V3/Northwind/No...
Alphabetical_list_of_products
Categories
Category_Sales_for_1997
Current_Product_Lists
Customer_and_Suppliers_by_Cities
CustomerDemographics
Customers
Employees
Invoices
Order_Details
Order_Details_Extendeds
Order_Subtotals
Orders
Orders_Qries
Product_Sales_for_1997
Products
Products_Above_Average_Prices
Products_by_Categories
Regions
Orders
OrderID CustomerID EmployeeID OrderDate RequiredDate
10248 VINET 5 7/4/1996 12:00:00 AM 8/1/199
10249 TOMSP 6 7/5/1996 12:00:00 AM 8/16/199
10250 HANAR 4 7/8/1996 12:00:00 AM 8/5/199
10251 VICTE 3 7/8/1996 12:00:00 AM 8/5/199
10252 SUPRD 4 7/9/1996 12:00:00 AM 8/6/199
10253 HANAR 3 7/10/1996 12:00:00 AM 7/24/199
10254 CHOPS 5 7/11/1996 12:00:00 AM 8/8/199
10255 RICSU 9 7/12/1996 12:00:00 AM 8/9/199
10256 WELLI 3 7/15/1996 12:00:00 AM 8/12/199
10257 HILAA 4 7/16/1996 12:00:00 AM 8/13/199
10258 ERNSH 1 7/17/1996 12:00:00 AM 8/14/199
10259 CENTC 4 7/18/1996 12:00:00 AM 8/15/199
10260 OTTIK 4 7/19/1996 12:00:00 AM 8/16/199
10261 QUEDE 4 7/19/1996 12:00:00 AM 8/16/199
10262 RATTC 8 7/22/1996 12:00:00 AM 8/19/199
10263 ERNSH 9 7/23/1996 12:00:00 AM 8/20/199
10264 FOLKO 6 7/24/1996 12:00:00 AM 8/21/199
10265 BLONP 2 7/25/1996 12:00:00 AM 8/22/199
10266 WARTH 3 7/26/1996 12:00:00 AM 9/6/199
10267 FRANK 4 7/29/1996 12:00:00 AM 8/26/199
10268 GROSR 8 7/30/1996 12:00:00 AM 8/27/199
10269 WHITC 5 7/31/1996 12:00:00 AM 8/14/199
10270 WARTH 1 8/1/1996 12:00:00 AM 8/29/199
Select Related Tables
Load
Transform Data
Cancel


= Source{[Name="Orders",Signature="table"]}[Data]
OrderID CustomerID EmployeeID OrderDate RequiredDate ShippedDat
1 10248 VINET 5 7/4/1996 12:00:00 AM 8/1/1996 12:00:00 AM 7/16/
2 10249 TOMSP 6 7/5/1996 12:00:00 AM 8/16/1996 12:00:00 AM 7/10/
3 10250 HANAR 4 7/8/1996 12:00:00 AM 8/5/1996 12:00:00 AM 7/12/
4 10251 VICTE 3 7/8/1996 12:00:00 AM 8/5/1996 12:00:00 AM 7/15/
5 10252 SUPRD 4 7/9/1996 12:00:00 AM 8/6/1996 12:00:00 AM 7/11/
6 10253 HANAR 3 7/10/1996 12:00:00 AM 7/24/1996 12:00:00 AM 7/16/
7 10254 CHOPS 5 7/11/1996 12:00:00 AM 8/8/1996 12:00:00 AM 7/23/
8 10255 RICSU 9 7/12/1996 12:00:00 AM 8/9/1996 12:00:00 AM 7/15/
9 10256 WELLI 3 7/15/1996 12:00:00 AM 8/12/1996 12:00:00 AM 7/17/
10 10257 HILAA 4 7/16/1996 12:00:00 AM 8/13/1996 12:00:00 AM 7/22/
11 10258 ERNSH 1 7/17/1996 12:00:00 AM 8/14/1996 12:00:00 AM 7/23/
12 10259 CENTC 4 7/18/1996 12:00:00 AM 8/15/1996 12:00:00 AM 7/25/
13 10260 OTTIK 4 7/19/1996 12:00:00 AM 8/16/1996 12:00:00 AM 7/29/
14 10261 QUEDE 4 7/19/1996 12:00:00 AM 8/16/1996 12:00:00 AM 7/30/
15 10262 RATTC 8 7/22/1996 12:00:00 AM 8/19/1996 12:00:00 AM 7/25/
16 10263 ERNSH 9 7/23/1996 12:00:00 AM 8/20/1996 12:00:00 AM 7/31/
17 10264 FOLKO 6 7/24/1996 12:00:00 AM 8/21/1996 12:00:00 AM 8/23/
18 10265 BLONP 2 7/25/1996 12:00:00 AM 8/22/1996 12:00:00 AM 8/12/
19 10266 WARTH 3 7/26/1996 12:00:00 AM 9/6/1996 12:00:00 AM 7/31/
20 10267 FRANK 4 7/29/1996 12:00:00 AM 8/26/1996 12:00:00 AM 8/6/
21 10268 GROSR 8 7/30/1996 12:00:00 AM 8/27/1996 12:00:00 AM 8/2/
22 10269 WHITC 5 7/31/1996 12:00:00 AM 8/14/1996 12:00:00 AM 8/9/
23 10270 WARTH 1 8/1/1996 12:00:00 AM 8/29/1996 12:00:00 AM 8/2/
24
nn profiling based on top 1000 rows

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-4**

Class: BE (A) **Branch: AI&DS**

Assignment No.: 02

Output:

[5]:
```
# Assignment No: 08
# Title: Data Visualization from Extraction Transformation and Loading (ETL) Process

# Step 1: Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[6]:
```
# ---------------- ETL Process ----------------
```

[7]:
```
# Step 2: Extract (Simulating data extraction from CSV)
# In real life, data may come from databases, APIs, or flat files

# Sample CSV file link (you can replace it with your dataset)
url = 'https://people.sc.fsu.edu/~jburkardt/data/csv/hw_200.csv'
data = pd.read_csv(url)

print("Extracted Data (First 5 rows):")
print(data.head())
```

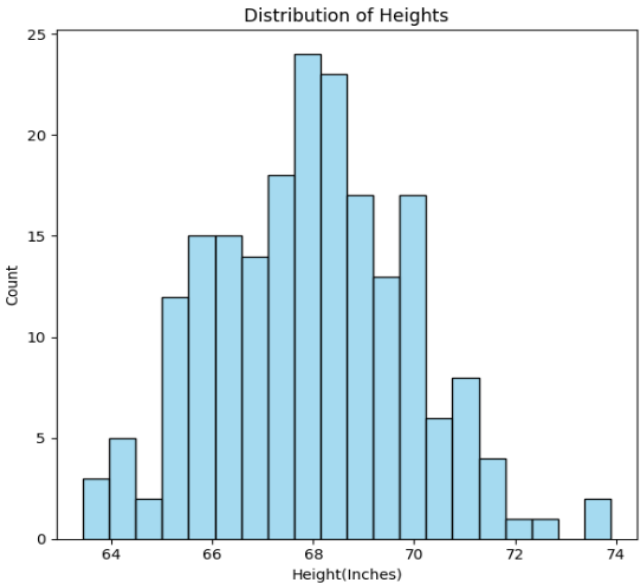
```
Extracted Data (First 5 rows):
   Index   Height(Inches)"   "Weight(Pounds)"
0      1            65.78             112.99
1      2            71.52             136.49
2      3            69.40             153.03
3      4            68.22             142.34
4      5            67.79             144.30
```

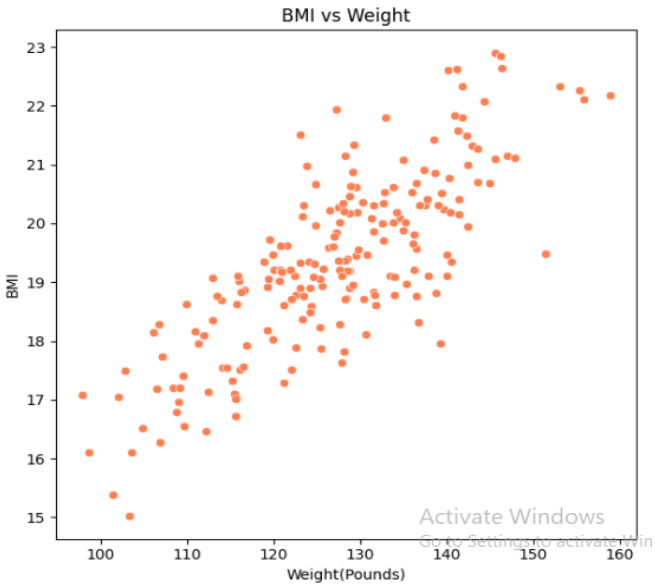
[8]:
```
# Step 3: Transform (Clean and prepare data)
# Let's assume we want to clean and derive some fields.

# Renaming columns
data.columns = ['Index', 'Height(Inches)', 'Weight(Pounds)']

# Removing missing values (if any)
data = data.dropna()

# Adding BMI column (derived value)
data['BMI'] = (data['Weight(Pounds)'] / (data['Height(Inches)'] ** 2)) * 703

print("\nTransformed Data (First 5 rows with BMI):")
print(data.head())
```

```
Transformed Data (First 5 rows with BMI):
   Index  Height(Inches)  Weight(Pounds)        BMI
0      1           65.78          112.99  18.357249
1      2           71.52          136.49  18.758631
2      3           69.40          153.03  22.336389
3      4           68.22          142.34  21.501010
4      5           67.79          144.30  22.074475
```

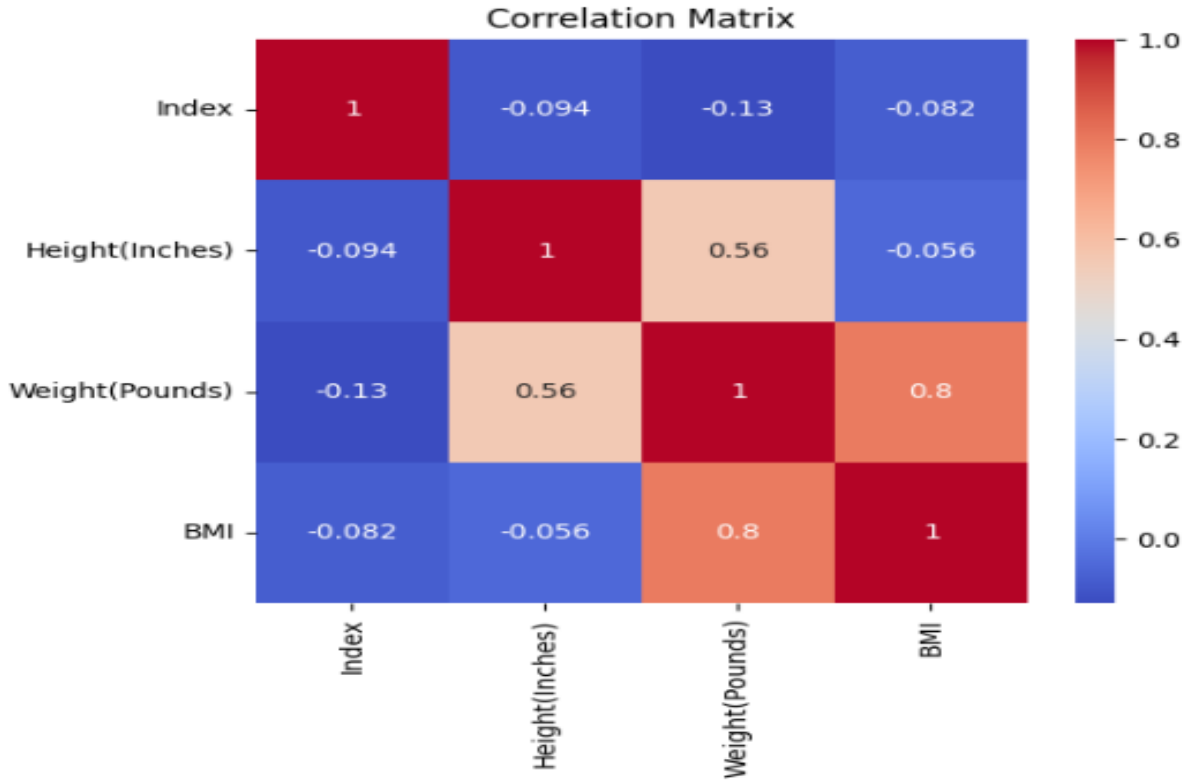
[9]:
```
# Step 4: Load (Load into DataFrame, ready for use)
# Here, load is simulated by saving to another DataFrame or saving to a CSV (if needed)
loaded_data = data.copy()
# Optionally, you could save this cleaned data
# loaded_data.to_csv('cleaned_data.csv', index=False)
```

[10]:
```
# ---------------- Data Visualization ----------------

# Step 5: Visualize extracted and transformed data

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

# Histogram of Heights
plt.subplot(1, 2, 1)
sns.histplot(loaded_data['Height(Inches)'], bins=20, color='skyblue')
plt.title('Distribution of Heights')

# Scatter plot of BMI vs Weight
plt.subplot(1, 2, 2)
sns.scatterplot(x=loaded_data['Weight(Pounds)'], y=loaded_data['BMI'], color='coral')
plt.title('BMI vs Weight')

plt.tight_layout()
plt.show()
```





[11]:
```
# Step 6: Correlation heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(loaded_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



Name: Yash Laxman Gawade

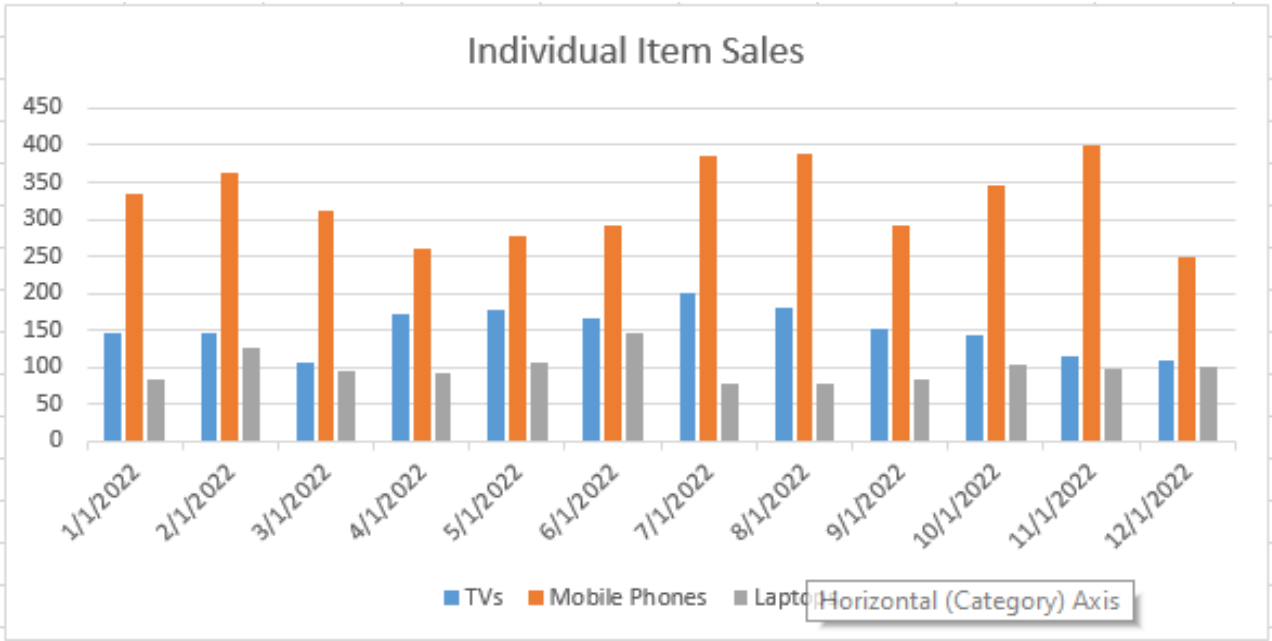
Roll No.: 21140 **Subject: CL-4**

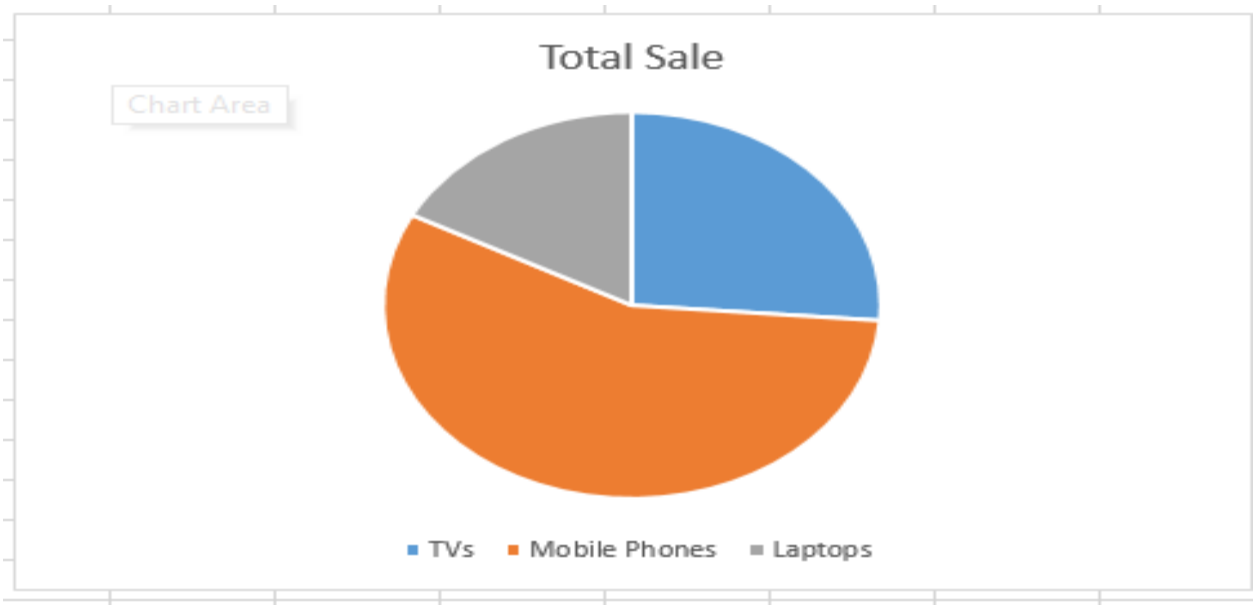
Class: BE (A) **Branch: AI&DS**

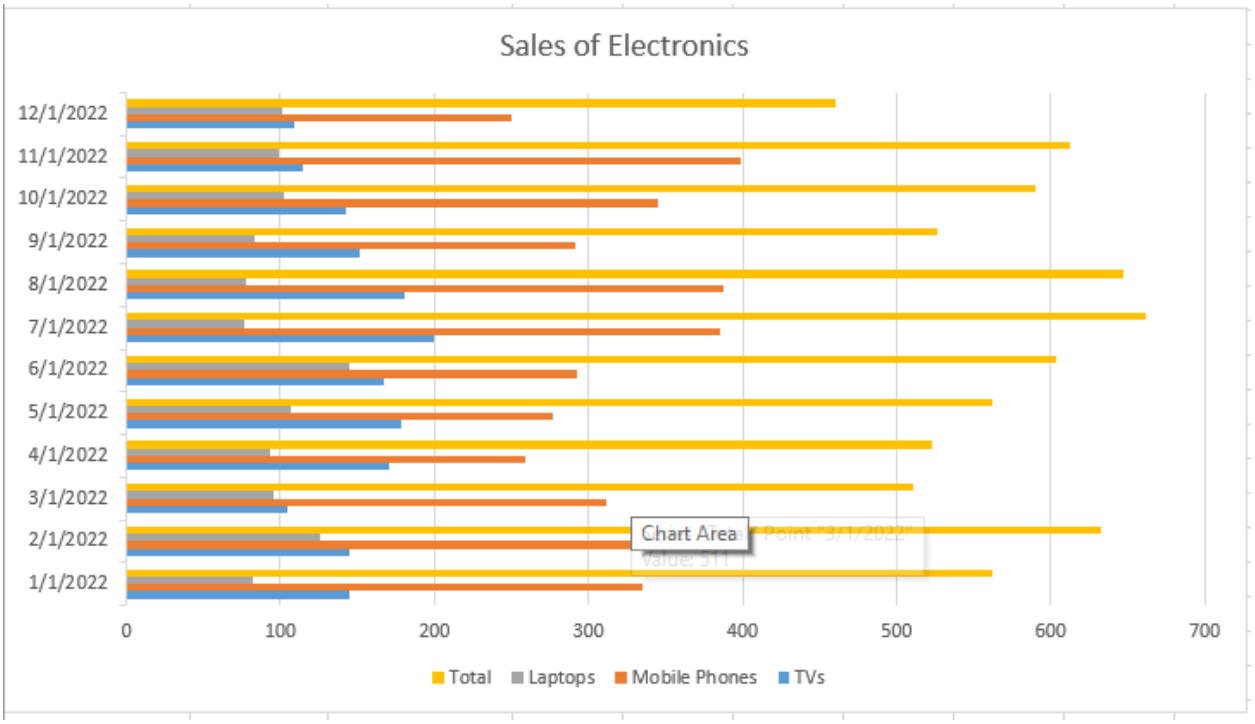
Assignment No.: 03

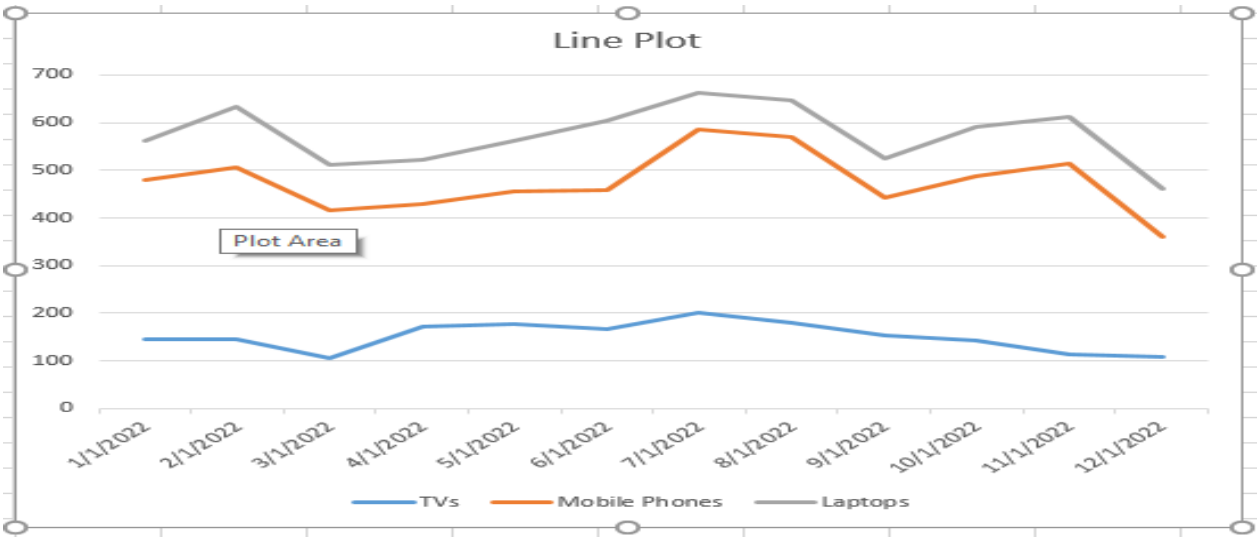
Output:

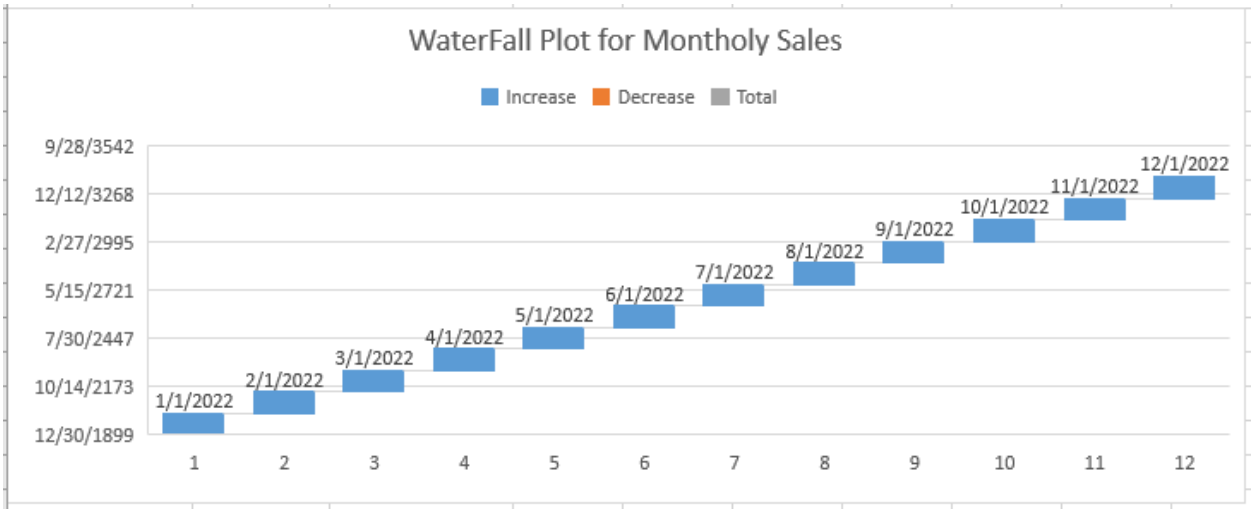
	A	B	C	D	E
1	**Month**	**TVs**	**Mobile Phones**	**Laptops**	**Total**
2	1/1/2022	145	335	82	562
3	2/1/2022	145	362	126	633
4	3/1/2022	105	311	95	511
5	4/1/2022	171	259	93	523
6	5/1/2022	178	277	107	562
7	6/1/2022	167	292	145	604
8	7/1/2022	200	385	77	662
9	8/1/2022	181	388	78	647
10	9/1/2022	152	291	83	526
11	10/1/2022	143	345	102	590
12	11/1/2022	114	399	99	612
13	12/1/2022	109	250	101	460
14	**Total**	1810	3894	1188	

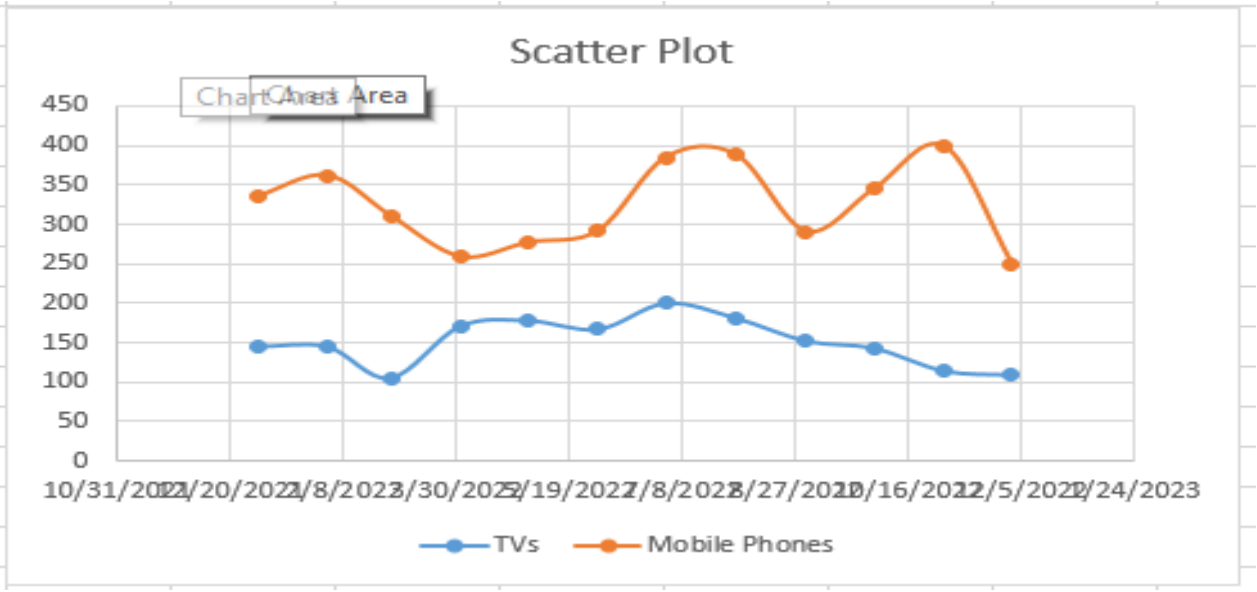



Total Sale
Chart Area
TVs
Mobile Phones
Laptops


Sales of Electronics
12/1/2022
11/1/2022
10/1/2022
9/1/2022
8/1/2022
7/1/2022
6/1/2022
5/1/2022
4/1/2022
3/1/2022
2/1/2022
1/1/2022
0
100
200
300
400
500
600
700
Chart Area
Total
Laptops
Mobile Phones
TVs


Line Plot
700
600
500
400
300
200
100
0
Plot Area
1/1/2022
2/1/2022
3/1/2022
4/1/2022
5/1/2022
6/1/2022
7/1/2022
8/1/2022
9/1/2022
10/1/2022
11/1/2022
12/1/2022
TVs
Mobile Phones
Laptops


WaterFall Plot for Montholy Sales
Increase
Decrease
Total
9/28/3542
12/12/3268
2/27/2995
5/15/2721
7/30/2447
10/14/2173
12/30/1899
1/1/2022
2/1/2022
3/1/2022
4/1/2022
5/1/2022
6/1/2022
7/1/2022
8/1/2022
9/1/2022
10/1/2022
11/1/2022
12/1/2022
1
2
3
4
5
6
7
8
9
10
11
12


Scatter Plot
Chart Area
450
400
350
300
250
200
150
100
50
0
10/31/2022
11/20/2022
12/8/2022
12/30/2022
1/19/2023
2/7/2023
2/27/2023
3/18/2023
4/5/2023
4/24/2023
TVs
Mobile Phones

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-3**

Class: BE (A) **Branch: AI&DS**

Assignment No.:

Output:

```
[4] ✓ 0.3s
import numpy as np
```

```
[5] ✓ 0.0s
# Step 1: Generate Dummy Data (Replace with actual dataset in real use case)
def generate_dummy_data(samples=100, features=10):
    data = np.random.rand(samples, features)  # Random features between 0 and 1
    labels = np.random.randint(0, 2, size=samples)  # Binary labels (0 = healthy, 1 = damaged)
    return data, labels
```

```
[6] ✓ 0.0s
# Step 2: Define AIS Algorithm (Simple Clonal Selection-based detector)
class AIRS:
    def __init__(self, num_detectors=10, hypermutation_rate=0.1):
        self.num_detectors = num_detectors
        self.hypermutation_rate = hypermutation_rate

    def train(self, X, y):
        # Randomly select initial detectors from training data
        self.detectors = X[np.random.choice(len(X), self.num_detectors, replace=False)]

    def predict(self, X):
        predictions = []
        for sample in X:
            distances = np.linalg.norm(self.detectors - sample, axis=1)
            nearest = np.argmin(distances)
            # Basic prediction based on the closest detector
            predictions.append(1 if distances[nearest] < 0.5 else 0)
        return np.array(predictions)
```

```
# Step 3: Generate Dummy Data
data, labels = generate_dummy_data()

# Step 4: Train-Test Split
split_ratio = 0.8
split_index = int(split_ratio * len(data))
train_data, test_data = data[:split_index], data[split_index:]
train_labels, test_labels = labels[:split_index], labels[split_index:]

# Step 5: Initialize and Train AIS
airs = AIRS(num_detectors=10, hypermutation_rate=0.1)
airs.train(train_data, train_labels)

# Step 6: Predict and Evaluate
predictions = airs.predict(test_data)
accuracy = np.mean(predictions == test_labels)
print(f"Accuracy: {accuracy:.2f}")
```

[7] ✓ 0.0s

```
Accuracy: 0.60
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-3**

Class: BE (A) **Branch : AI&DS**

Assignment No.:

Output:

```
from deap import base, creator, tools, algorithms
import random
```

[1] ✓ 0.5s

```
# DEAP setup
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)
```

✓ 0.0s

```
toolbox = base.Toolbox()
toolbox.register("attr_float", random.random)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_float, n=5)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

```
def evaluate(individual):
    return sum(individual),  # Minimize the sum

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)
```

✓ 0.0s

```python
# Remove multiprocessing for now
toolbox.register("map", map)

if __name__ == "__main__":
    population = toolbox.population(n=20)
    generations = 5

    for gen in range(generations):
        offspring = algorithms.varAnd(population, toolbox, cxpb=0.5, mutpb=0.1)
        fits = toolbox.map(toolbox.evaluate, offspring)
        for fit, ind in zip(fits, offspring):
            ind.fitness.values = fit
        population = toolbox.select(offspring, k=len(population))

    best_ind = tools.selBest(population, k=1)[0]
    print("\nBest individual:", best_ind)
    print("Best fitness:", best_ind.fitness.values[0])
```

✓ 0.0s

```
Best individual: [0.30776524016564444, -1.3885378861784612, 0.4681687798006477, 0.07405385654992228, 0.03267047924570492]
Best fitness: -0.5058795304165419
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-4**

Class: BE (A) **Branch : AI&DS**

Assignment No.:

Output:

bda1_mapreduce.py > ...

```python
import multiprocessing
import re
from collections import Counter

# Function to read input file
def read_file(filename):
    with open(filename, 'r', encoding='utf-8') as f:
        return f.read()

# Mapper function to count word occurrences in each chunk
def word_frequency_mapper(args):
    text_chunk, target_word = args
    words = re.findall(r'\b\w+\b', text_chunk.lower())  # Extract full words
    return Counter(words)[target_word]  # Count target word

# Reducer function to sum up all partial counts
def reducer(counts_list):
    return sum(counts_list)

# Function to split text into chunks
def split_text_into_chunks(text, num_chunks):
    words = re.findall(r'\b\w+\b', text)  # Extract words
    num_chunks = min(num_chunks, len(words))  # Ensure valid chunk count
    chunk_size = max(1, len(words) // num_chunks)  # Avoid zero chunk_size

    chunks = [" ".join(words[i:i + chunk_size]) for i in range(0, len(words), chunk_size)]
    return chunks
```

```python
# Main MapReduce Function
def mapreduce(filename, target_word):
    text = read_file(filename)
    num_workers = min(multiprocessing.cpu_count(), len(text))  # Set a reasonable worker count
    chunks = split_text_into_chunks(text, num_workers)  # Create chunks

    with multiprocessing.Pool(processes=num_workers) as pool:
        mapped_results = pool.map(word_frequency_mapper, [(chunk, target_word) for chunk in chunks])

    return reducer(mapped_results)

if __name__ == "__main__":
    filename = "sample_bda1.txt"  # Change this to your text file
    target_word = "mapreduce"  # Change this to the word you want to count

    # Calculate word frequency using MapReduce
    word_frequency = mapreduce(filename, target_word.lower())

    print(f"Frequency of '{target_word}': {word_frequency}")
```

```
C:\Users\ADMIN\OneDrive\Documents\CL4>python bda1_mapreduce.py
Frequency of 'mapreduce': 3
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-4**

Class: BE (A) **Branch : AI&DS**

Assignment No.:

Output:

bda2_matrix_multi.py > ...

```python
import multiprocessing
import numpy as np

# Function to take matrix input from user
def input_matrix(rows, cols, name):
    print(f"Enter matrix {name} ({rows}x{cols}):")
    matrix = []
    for i in range(rows):
        row = list(map(int, input(f"Row {i + 1}: ").split()))
        if len(row) != cols:
            raise ValueError(f"Each row must have {cols} elements!")
        matrix.append(row)
    return np.array(matrix)

# Mapper function to compute partial dot products
def matrix_multiply_mapper(args):
    row, B = args  # Unpack row of A and matrix B
    return [sum(a * b for a, b in zip(row, col)) for col in zip(*B)]  # Compute row of C

# Reducer function to collect results
def matrix_multiply_reducer(results):
    return np.array(results)

# MapReduce function for matrix multiplication
def mapreduce_matrix_multiplication(A, B):
    num_workers = min(multiprocessing.cpu_count(), len(A))  # Define parallel workers

    with multiprocessing.Pool(processes=num_workers) as pool:
        mapped_results = pool.map(matrix_multiply_mapper, [(row, B) for row in A])

    return matrix_multiply_reducer(mapped_results)
```

```python
if __name__ == "__main__":
    # Take matrix dimensions as input
    ROWS_A = int(input("Enter number of rows for Matrix A: "))
    COLS_A = int(input("Enter number of columns for Matrix A: "))
    ROWS_B = int(input("Enter number of rows for Matrix B: "))
    COLS_B = int(input("Enter number of columns for Matrix B: "))

    # Ensure matrix dimensions are valid for multiplication (COLS_A == ROWS_B)
    if COLS_A != ROWS_B:
        raise ValueError("Matrix multiplication not possible! Number of columns in A must match rows in B.")

    # Take matrix input from the user
    A = input_matrix(ROWS_A, COLS_A, "A")
    B = input_matrix(ROWS_B, COLS_B, "B")

    print("\nMatrix A:")
    print(A)
    print("\nMatrix B:")
    print(B)

    # Perform MapReduce-based matrix multiplication
    C = mapreduce_matrix_multiplication(A, B)

    print("\nResultant Matrix C (A x B):")
    print(C)
```

```
C:\Users\ADMIN\OneDrive\Documents\CL4>python bda2_matrix_multi.py
Enter number of rows for Matrix A: 2
Enter number of columns for Matrix A: 3
Enter number of rows for Matrix B: 3
Enter number of columns for Matrix B: 2
Enter matrix A (2x3):
Row 1: 1 2 3
Row 2: 4 5 6
Enter matrix B (3x2):
Row 1: 7 8
Row 2: 9 10
Row 3: 11 12

Matrix A:
[[1 2 3]
 [4 5 6]]

Matrix B:
[[ 7  8]
 [ 9 10]
 [11 12]]

Resultant Matrix C (A x B):
[[ 58  64]
 [139 154]]

C:\Users\ADMIN\OneDrive\Documents\CL4>
```

Name: Yash Laxman Gawade

Roll No.: 21140 **Subject: CL-4**

Class: BE (A) **Branch : AI&DS**

Assignment No.:

Output:

bda3_student_grade.py > input_students

```python
import multiprocessing

# Function to take student data input
def input_students():
    num_students = int(input("Enter number of students: "))
    students = []
    for _ in range(num_students):
        name = input("Enter student name: ")
        marks = float(input(f"Enter marks for {name}: "))
        students.append((name, marks))
    return students

# Mapper function - Assigns grades based on marks
def grade_mapper(student):
    name, marks = student
    if marks >= 90:
        grade = "A+"
    elif marks >= 80:
        grade = "A"
    elif marks >= 70:
        grade = "B"
    elif marks >= 60:
        grade = "C"
    elif marks >= 50:
        grade = "D"
    else:
        grade = "F"
    return (name, grade)
```

```python
# Reducer function - Collects results into a dictionary
def grade_reducer(mapped_results):
    return dict(mapped_results)

# MapReduce function
def mapreduce_student_grades(students):
    num_workers = min(multiprocessing.cpu_count(), len(students))

    with multiprocessing.Pool(processes=num_workers) as pool:
        mapped_results = pool.map(grade_mapper, students)

    return grade_reducer(mapped_results)

if __name__ == "__main__":
    # Input student data
    students = input_students()

    print("\nCalculating Grades...\n")

    # Compute grades using MapReduce
    student_grades = mapreduce_student_grades(students)

    # Display results
    print("Student Grades:")
    for student, grade in student_grades.items():
        print(f"{student}: {grade}")

```

```
C:\Users\ADMIN\OneDrive\Documents\CL4>python bda3_student_grade.py
ade.py
Enter number of students: 3
Enter student name: Yash
Enter marks for Yash: 95
Enter student name: Sai
Enter marks for Sai: 90
Enter student name: Kushal
Enter marks for Kushal: 80

Calculating Grades...

Student Grades:
Yash: A+
Sai: A+
Kushal: A
```