```
                                                    #GE ASSIGNMENT 1

#NAME:HARSH ARORA
#ROLL NO:AE-1218
#COURSE:BSC(HONS.)COMPUTER SCIENCE

#1.ACCEPTING STUDENT DETAILS:

name=str(input("ENTER STUDENT'S NAME : "))
roll=int(input("ENTER STUDENT'S ROLL NO : "))
course=str(input("ENTER STUDENT'S COURSE: "))

print("\nSTUDENT'S NAME IS",name)
print("STUDENT'S ROLL NO IS",roll)
print("STUDENT'S COURSE IS",course)

print(".....................................................................")
```

```
ENTER STUDENT'S NAME : HARSH ARORA
ENTER STUDENT'S ROLL NO : 18
ENTER STUDENT'S COURSE: BSC HONS. COMPUTER SCIENCE

STUDENT'S NAME IS HARSH ARORA
STUDENT'S ROLL NO IS 18
STUDENT'S COURSE IS BSC HONS. COMPUTER SCIENCE
..................................................
```

```python
#2.GRADING BASED ON THE MARKS OF STUDENTS.

marks_maths=int(input("ENTER THE MARKS OF MATHS: "))
marks_english=int(input("ENTER THE MARKS OF ENGLISH: "))
marks_physics=int(input("ENTER THE MARKS OF PHYSICS: "))
marks_chemistery=int(input("ENTER THE MARKS OF CHEMISTRY: "))
marks_history=int(input("ENTER THE MARKS OF HISTORY: "))

marks_percentage=((marks_maths+marks_english+marks_physics+marks_chemistery+marks_history)/500)*100


if marks_percentage>=90:
    print("GRADE=O")
    print("OUTSTANDING")

elif 90<marks_percentage>=80:
    print("GRADE=A+")
    print("EXCELLENT")

elif 80<marks_percentage>=70:
    print("GRADE=A")
    print("VERY GOOD")

elif 70<marks_percentage>=60:
    print("GRADE=B+")
    print("GOOD")
```

```
elif 60<marks_percentage>=50:
    print("GRADE=B")
    print("ABOVE AVERAGE")

elif 50<marks_percentage>=45:
    print("GRADE=C")
    print("AVERAGE")

elif 45<marks_percentage>=40:
    print("GRADE=D")
    print("PASS")

elif marks_percentage<40:
    print("GRADE=F")
    print("FAIL")
print(".....")
```

```
ENTER THE MARKS OF MATHS: 90
ENTER THE MARKS OF ENGLISH: 97
ENTER THE MARKS OF PHYSICS: 89
ENTER THE MARKS OF CHEMISTRY: 85
ENTER THE MARKS OF HISTORY: 90
GRADE=O
OUTSTANDING
.....
```

```
#3.FINDING ROOTS OF A QUADRATIC EQUATION:

a=int(input("Enter The Cofficient of X2: "))
b=int(input("Enter The Cofficient of X: "))
c=int(input("Enter The Cofficient of Constant: "))
d=b*2-4*a*c
if d>0:
    print("\nROOTS ARE REAL AND DISTINCT\n")
    x1=(-b+((d)**0.5))/2*a
    x2=(-b-((d)**0.5))/2*a
    print("The Roots are",x1,",",x2)
if d<0:
    print("\nROOTS ARE COMPLEX\n")
if d==0:
    print("\nROOTS ARE REAL AND EQUAL\n")

print(".....................................................")
```

```
Enter The Cofficient of X2: 3
Enter The Cofficient of X: 5
Enter The Cofficient of Constant: -7

ROOTS ARE REAL AND DISTINCT

The Roots are 7.043039572248988 , -22.043039572248986
..................................................
```

```
#4.WRITE A MENU DRIVEN PROGRAMM :

print("\nMENU\n")
print("\n1:X**y")
print("\n2:A.P SERIES")
print("\n3:G.P SERIES")
print("\n4:FACTORIAL\n")
a=int(input("ENTER THE OPEARTION YOU WANT TO PERFORM(1,2,3,4): "))


if a==1:
    print("\nPERFORMING X**Y\n")
    def function(x,y):
        return(x**y)
    x=int(input("ENTER THE VALUE OF X: "))
    y=int(input("ENTER THE VALUE OF y: "))
    print(function(x,y))

if a==2:
    print("\nPERFORMING A.P SERIES\n")
    def AP_SERIES(b):
        i=1
        print(b)
        while d>i:
            b=b+c
            print(b)
            i=i+1

    b=int(input("ENTER THE STARTING NUMBER: "))
    c=int(input("ENTER THE COMMON DIFFERENCE: "))
    d=int(input("ENTER THE NUMBER OF TERMS YOU WANT: "))
    print(AP_SERIES(b))
```

```
if a==3:
    print("\nPERFORMING G.P SERIES\n")
    def GP_SERIES(b):
        i=1
        print(b)
        while d>i:
            b=b*c
            print(b)
            i=i+1

    b=int(input("ENTER THE STARTING NUMBER: "))
    c=int(input("ENTER THE COMMON DIFFERENCE: "))
    d=int(input("ENTER THE NUMBER OF TERMS YOU WANT: "))
    print(GP_SERIES(b))


if a==4:
    print("\nPERFORMING FACTORIAL \n")
    def FACTORIAL(b):
        factorial=1
        for i in range(1,b+1):
            factorial=factorial*i
        print("\nTHE FACTORIAL OF THE NUMBER IS")
        return(factorial)
    b=int(input("ENTER THE NUMBER: "))
    print(FACTORIAL(b))
```

```
MENU


1:X**y

2:A.P SERIES

3:G.P SERIES

4:FACTORIAL

ENTER THE OPEARTION YOU WANT TO PERFORM(1,2,3,4): 2

PERFORMING A.P SERIES

ENTER THE STARTING NUMBER: 2
ENTER THE COMMON DIFFERENCE: 2
ENTER THE NUMBER OF TERMS YOU WANT: 5
2
4
6
8
10
```

```
MENU


1:X**y

2:A.P SERIES

3:G.P SERIES

4:FACTORIAL

ENTER THE OPEARTION YOU WANT TO PERFORM(1,2,3,4): 4

PERFORMING FACTORIAL

ENTER THE NUMBER: 5

THE FACTORIAL OF THE NUMBER IS
120
```

```
#GE ASSIGNMENT 2

#1.>Write a program to create a PANDAS Data Series containing names of 10 students

# ⌄ Accept names of 10 students as input from the user and create a Pandas series:

import pandas as pd

print("\nINPUTING NAMES : \n")
students=[]

print("ENTER THE NAME OF STUDENTS: ")
for i in range(1,11):
    a=input()
    students.append(a)
print("\nSTUDENT LIST: \n")
Students_names=pd.Series(students)
print(Students_names)

print(".....")
```

```
INPUTING NAMES :

ENTER THE NAME OF STUDENTS:
HARSH
AKSHAT
MEHUL
DHRUB
JAMEEL
HARDICK
ISHAN
VIKAS
HARSHJEET
KUSH

STUDENT LIST:

0         HARSH
1        AKSHAT
2         MEHUL
3         DHRUB
4        JAMEEL
5       HARDICK
6         ISHAN
7         VIKAS
8     HARSHJEET
9          KUSH
dtype: object
```

```
# ⌄ Rename Index column of Pandas series with 'Student Roll Number' starting from 1001,1002, 1003... and so on

print("\nINDEX ROLL NUMBERS : \n")
RollNumber=[]
L=len(Students_names)
for i in range(0,L):
    RollNumber.append(100+i)
Students_names.index=RollNumber
print(Students_names)

print(".....................................................................................")
```

```
INDEX ROLL NUMBERS :

100          HARSH
101         AKSHAT
102          MEHUL
103          DHRUB
104         JAMEEL
105        HARDICK
106          ISHAN
107          VIKAS
108     HARSHJEET
109           KUSH
dtype: object
```

```
# ↓ Save Pandas data series created in .xlsx file, .csv file and .json file

Students_names.to_excel("StudentSeries.xlsx")
Students_names.to_csv("StudentSeries.csv")
Students_names.to_json("StudentSeries.json")

print(".....")
```

```
#.Extract and display information of a particular student using Roll Number (index):

print("\nDISPLAYING INFORMATION: \n")

print('Details of student with Roll Number: 106')
print(Students_names[102])

print(".....................................................................................")
```

```
DISPLAYING INFORMATION:

Details of student with Roll Number: 106
MEHUL
```

```
# ⌄ Sort Student names in alphabetical order:

print("\nASENDING ORDER NAMES: \n")

Students_names=Students_names.sort_values(ascending=True)
print(Students_names)
```

```
ASENDING ORDER NAMES:

101       AKSHAT
103        DHRUB
105      HARDICK
100        HARSH
108    HARSHJEET
106        ISHAN
104       JAMEEL
109         KUSH
102        MEHUL
107        VIKAS
dtype: object
```

```python
#.Add information of a new student into the Pandas series:

print("\nADDING NEW INFORMATION: \n")
Students_names[111]='ISHAN'
print(Students_names)


print(".....................................................................................")
```

```
ADDING NEW INFORMATION:

101        AKSHAT
103         DHRUB
105       HARDICK
100         HARSH
108     HARSHJEET
106         ISHAN
104        JAMEEL
109          KUSH
102         MEHUL
107         VIKAS
111         ISHAN
dtype: object
```

```
#. ✓ Remove information of a student of a given Roll Number:

print("\nREMOVING INFORMATION: \n")
Students_names=list(Students_names)
Students_names.remove('harsh')
Students_names= pd.Series(Students_names)
print(Students_names)

print(".....")
```

```
REMOVING INFORMATION:

0        AKSHAT
1         DHRUB
2       HARDICK
3     HARSHJEET
4        ISHAN
5        JAMEEL
6          KUSH
7         MEHUL
8         VIKAS
9         ISHAN
dtype: object
```

```python
#Write a program to create a PANDAS Data Series containing values of function Y= e**X
import pandas as pd
import numpy as np


#. Create the above shown Data Series Y=EXP(X)

print("\nSHOWING DATA: \n")
X = np.arange(0, 1.2, 0.2)
Y = np.exp(X)
series = pd.Series(Y)
print(series)

print(".....")
```

```
SHOWING DATA:


0    1.000000
1    1.221403
2    1.491825
3    1.822119
4    2.225541
5    2.718282
dtype: float64
```

```
#. ✓  Rename Index column with value of X
print("\nINDEX COLUMN WITH X: \n")
series = pd.Series(Y,index=X)
print(series)

print("..................................................................................")
```

INDEX COLUMN WITH X:

```
0.0    1.000000
0.2    1.221403
0.4    1.491825
0.6    1.822119
0.8    2.225541
1.0    2.718282
dtype: float64
```

```
#.Save Pandas data series created in .xlsx file, .csv file and .json file

series.to_excel("StudentSeries.xlsx")
series.to_csv("StudentSeries.csv")
series.to_json("StudentSeries.json")'''
```

```
#.Show Y=EXP(X) Vs. X through simple well labelled line plot:

import matplotlib.pyplot as plt

plt.plot(series.index, series.values)

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Y = e^X')

plt.show()
print(".....")
```

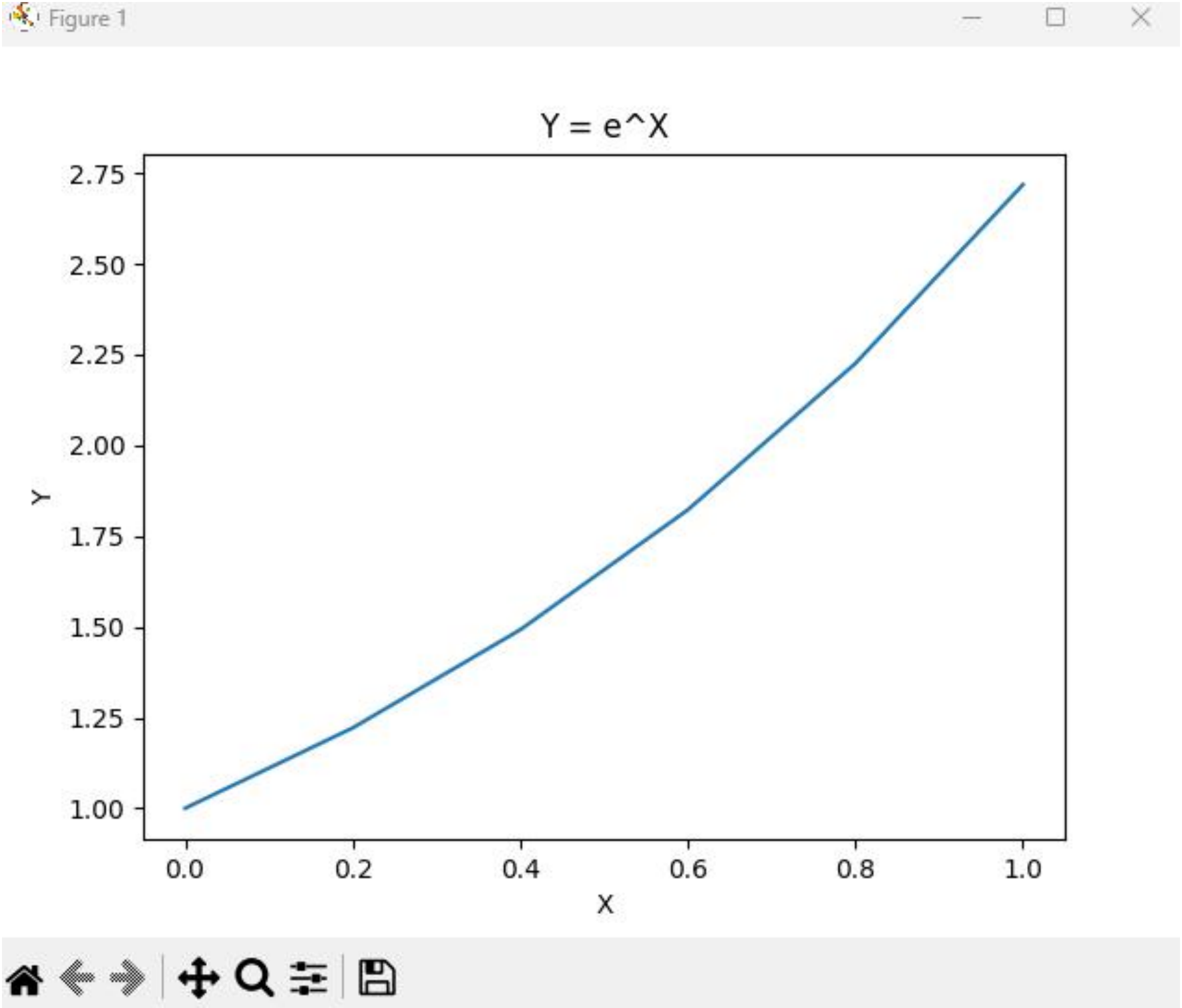


Figure 1
Y = e^X
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
Y
0.0
0.2
0.4
0.6
0.8
1.0
X

```
#.Save XY plot as .png file

series = pd.Series(Y, index=X)
plt.plot(series.index, series.values)

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Y = e^X')
plt.savefig('exp-plot.png', dpi=300, bbox_inches='tight')
plt.show()
print(".....")
```

```python
#3.Write a program to create a PANDAS Data Series containing marks of 10 students in Subject1:
# - Create the above shown Data Series using single-dimensional python list:
print("\nDATA SERIES USING LIST\n")
import pandas as pd
marks = ['80', '78', '95', '72', '97','69', '88', '35', '99', '65',]
students_marks = pd.Series(marks)
print(students_marks)
print(".....")
```

```
DATA SERIES USING LIST

0     80
1     78
2     95
3     72
4     97
5     69
6     88
7     35
8     99
9     65
dtype: object
```

```python
# - Create the above shown Data Series using single-dimensional python dictionary:
print("\nDATA SERIES USING DICTIONARY\n")
import pandas as pd

marks = {'Student 1':'80','Student 2':'78', 'Student 3':'95', 'Student 4':'72','Student 5':'97','Student 6':'85','Stud

students_marks = pd.Series(marks)

students_marks.name = 'MARKS OF STUDENTS'

print('Marks of Students')
print(students_marks)

print(".....")
```

DATA SERIES USING DICTIONARY

```
Marks of Students
Student 1      80
Student 2      78
Student 3      95
Student 4      72
Student 5      97
Student 6      85
Student 7      79
Student 8      99
Student 9      75
Student 10     45
Name: MARKS OF STUDENTS, dtype: object
```

```python
#.Rename Index column with Student Names:
import pandas as pd
import numpy as np

marks = np.array(['80', '78', '95', '72', '97','69', '88', '35', '99', '65',])
student_names=['HARSH','MEHUL','AKSHAT','JAMEEL','DHRUB','ROHAN','HARDICK','ABHINAV','ISHAN','HARSHJEET']

students_marks = pd.Series(marks, index=student_names)

students_marks.index.name="NAMES"

print('\nMARKS OF STUDENTS\n')
print(students_marks)

print(".....................................................................................")
```

MARKS OF STUDENTS

NAMES	
HARSH	80
MEHUL	78
AKSHAT	95
JAMEEL	72
DHRUB	97
ROHAN	69
HARDICK	88
ABHINAV	35
ISHAN	99
HARSHJEET	65

dtype: object

```
# ⌄ Save Pandas data series created in .xlsx file, .csv file and .json file:

'''Students_marks.to_excel("StudentSeries.xlsx")
Students_marks.to_csv("StudentSeries.csv")
Students_marks.to_json("StudentSeries.json")
```

```
#. - Sort marks in SUBJECT1 in descending order:
print("\nSORTING MARKS\n")

students_marks=students_marks.sort_values(ascending=False)
print(students_marks)

print(".....")
```

SORTING MARKS

NAMES	
ISHAN	99
DHRUB	97
AKSHAT	95
HARDICK	88
HARSH	80
MEHUL	78
JAMEEL	72
ROHAN	69
HARSHJEET	65
ABHINAV	35

dtype: object

```
# ⬇ Draw a Bar plot comparing marks in SUBJECT1 scored by the students:

import matplotlib.pyplot as plt
marks = np.array(['80', '78', '95', '72', '97','69', '88', '35', '99', '65',])
students_marks = pd.Series(marks, index=student_names)
students_marks=students_marks.sort_values(ascending=True)
student_names=['HARSH','MEHUL','AKSHAT','JAMEEL','DHRUB','ROHAN','HARDICK','ABHINAV','ISHAN','HARSHJEET']


plt.bar(students_marks, student_names)

plt.title("MARKS OF STUDENTS")
plt.xlabel("MARKS OBTAINED")
plt.ylabel("NAME")

# Show the graph
plt.show()
```

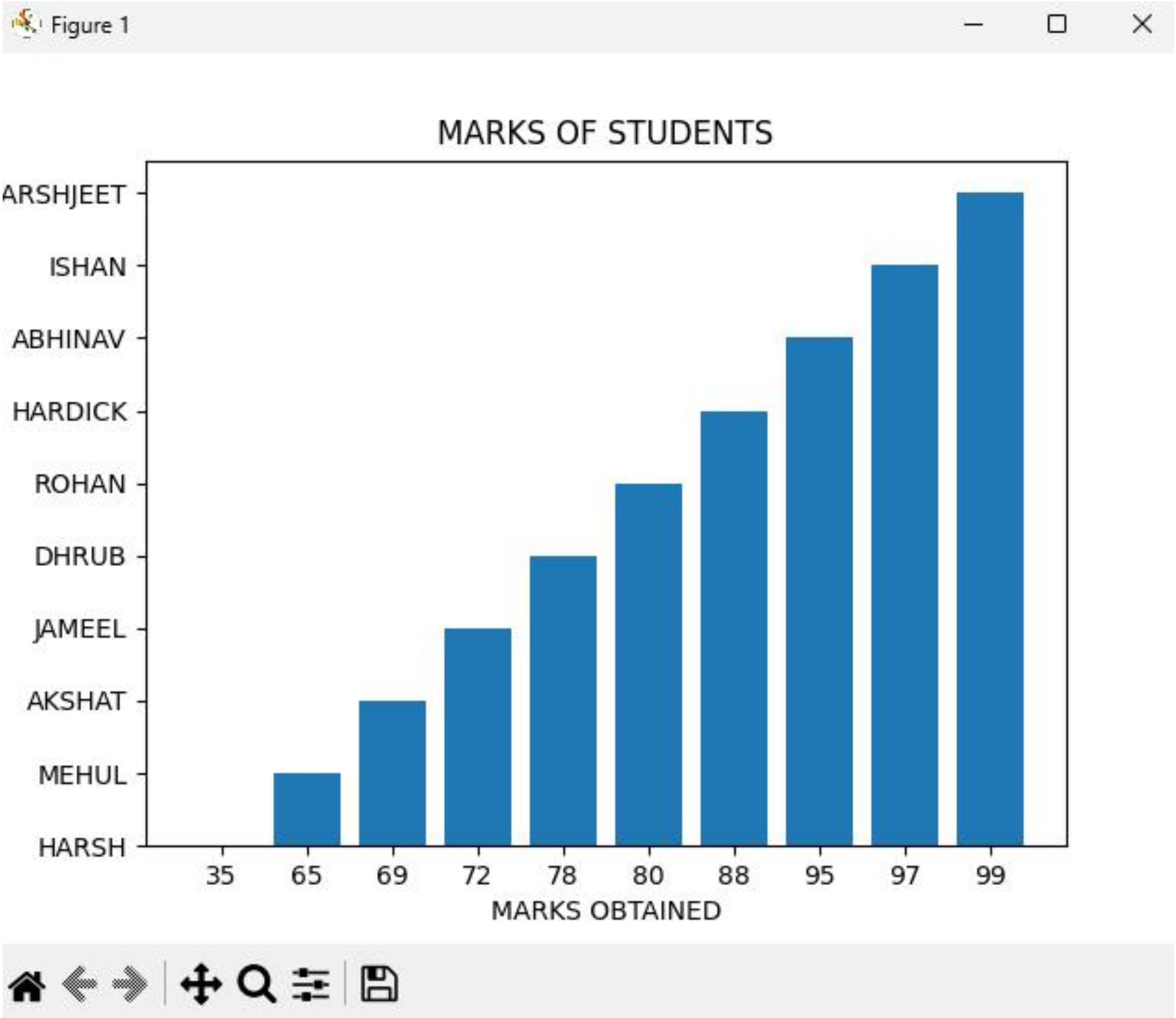


Figure 1
MARKS OF STUDENTS
ARSHJEET
ISHAN
ABHINAV
HARDICK
ROHAN
DHRUB
JAMEEL
AKSHAT
MEHUL
HARSH
35
65
69
72
78
80
88
95
97
99
MARKS OBTAINED

```
                                        #GE ASSIGNMENT 3
#1.>Write a program to create a PANDAS Data Series as shown below:
#PLOT GRAPHS:


import pandas as pd
import matplotlib.pyplot as plt
product=pd.Series({'1221':20,'1222':55,'1223':30,'1224':70,'1225':40,'1226':60})
product.name='sales'
product.index.name='product ID'
print(product)

product.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('product ID')
plt.ylabel('sales')
plt.legend(loc='lower center')
plt.show()

product.plot.area(color='b',label='Marks',figsize=(5,5))
plt.xlabel('product ID')
plt.ylabel('sales')
plt.legend(loc='lower center')
plt.show()

product.plot.bar(color='c',label='Marks',figsize=(3,3))
plt.xlabel('product ID')
plt.ylabel('sales')
plt.legend(loc='lower center')
plt.show()
```

```
product.plot.barh(color='c',label='Marks',figsize=(3,3))
plt.xlabel('product ID')
plt.ylabel('sales')
plt.legend(loc='lower center')
plt.show()

product.plot.box(color='m',label='product ID',figsize=(3,3))
plt.ylabel('sales')
plt.show()

product.plot.hist(bins=6,figsize=(3,3))
plt.ylabel('sales')
plt.show()

product.plot.pie(label='Marks',autopct='%1.1f%%')
plt.ylabel('sales')
plt.show()

print('.....................................................................')
```

```
product ID
1221    20
1222    55
1223    30
1224    70
1225    40
1226    60
Name: sales, dtype: int64
```

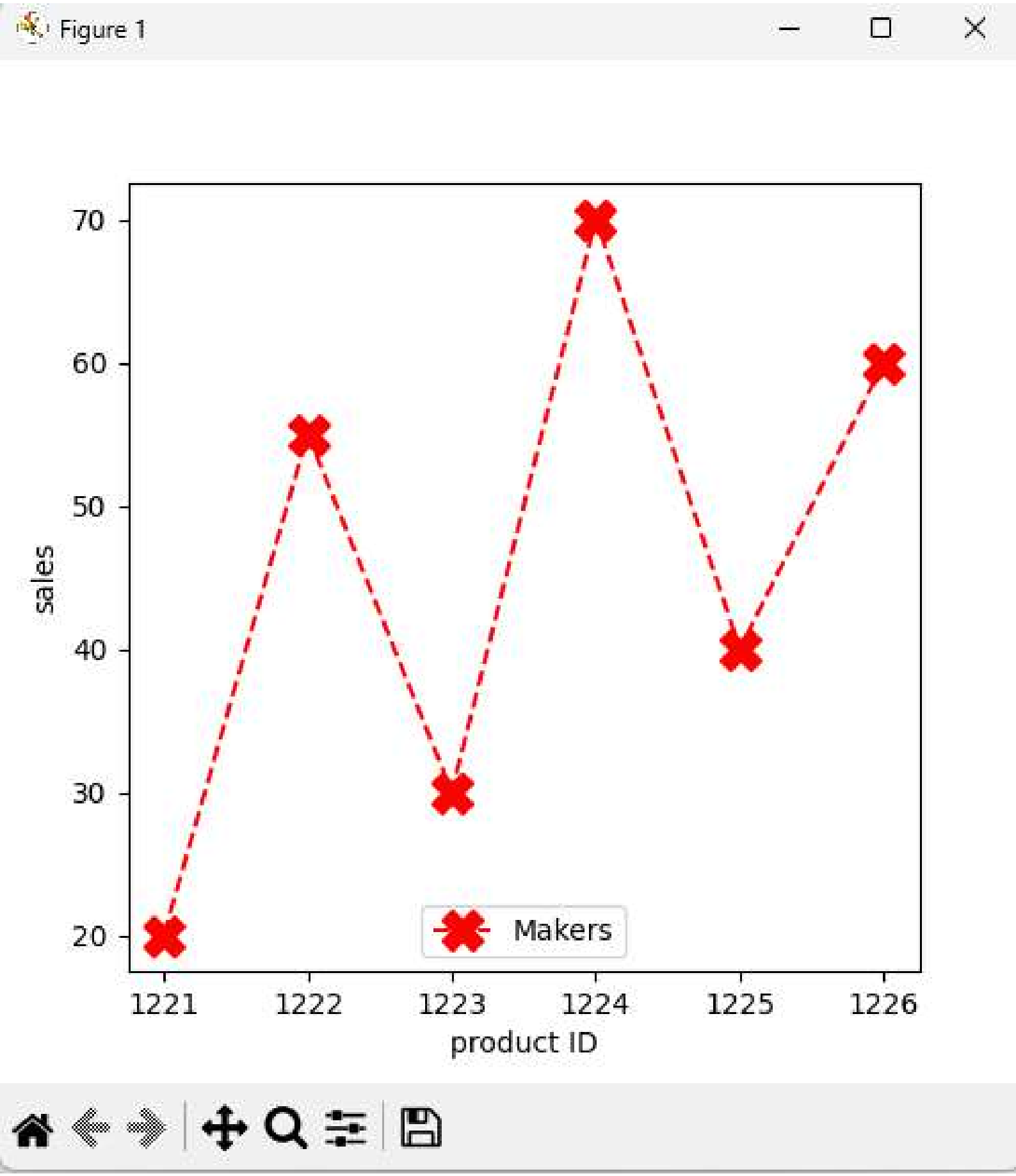


Figure 1
70
60
50
40
30
20
sales
Makers
1221
1222
1223
1224
1225
1226
product ID

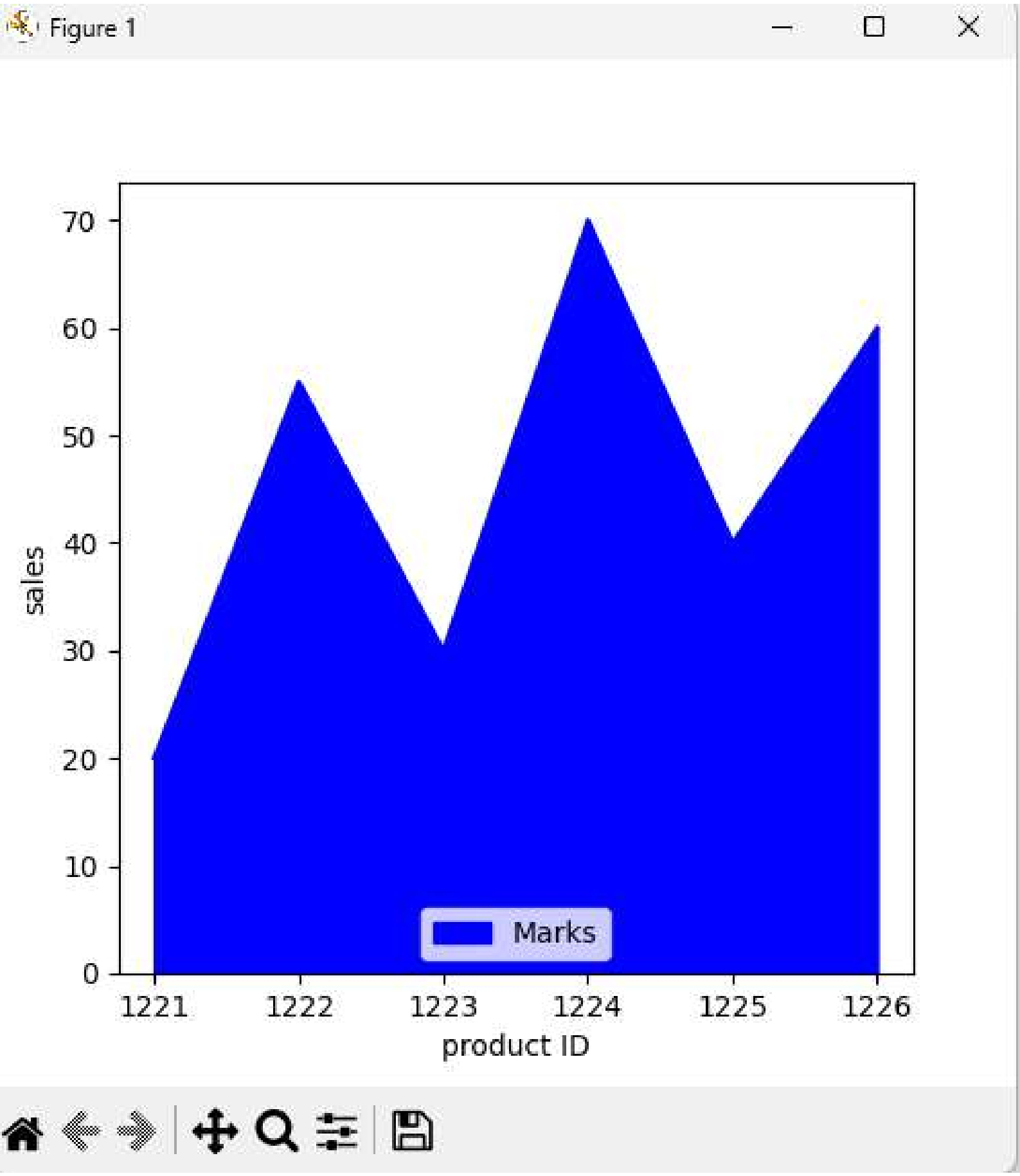


Figure 1
70
60
50
40
30
20
10
0
sales
Marks
1221
1222
1223
1224
1225
1226
product ID

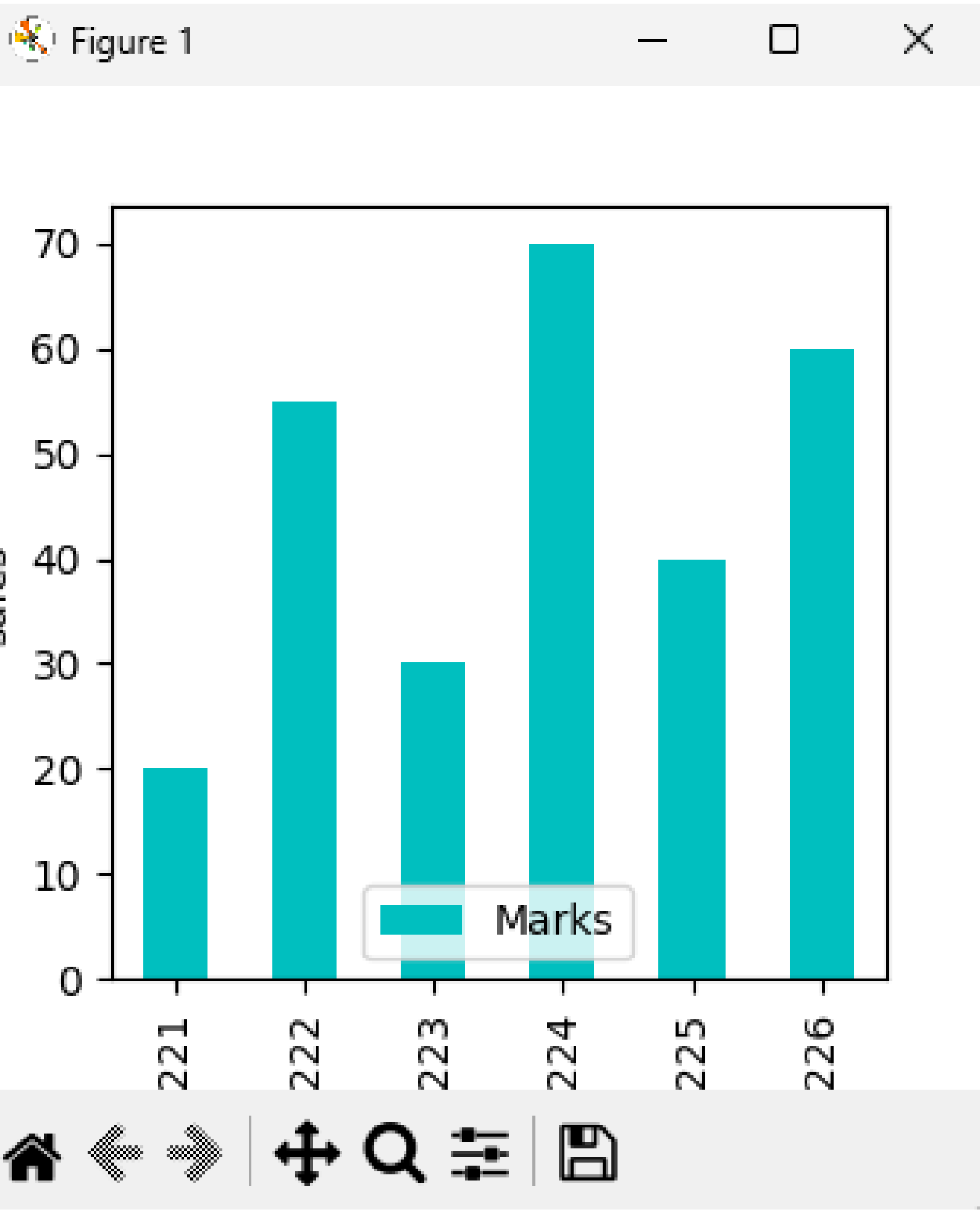


Figure 1
70
60
50
40
30
20
10
0
Marks
221
222
223
224
225
226

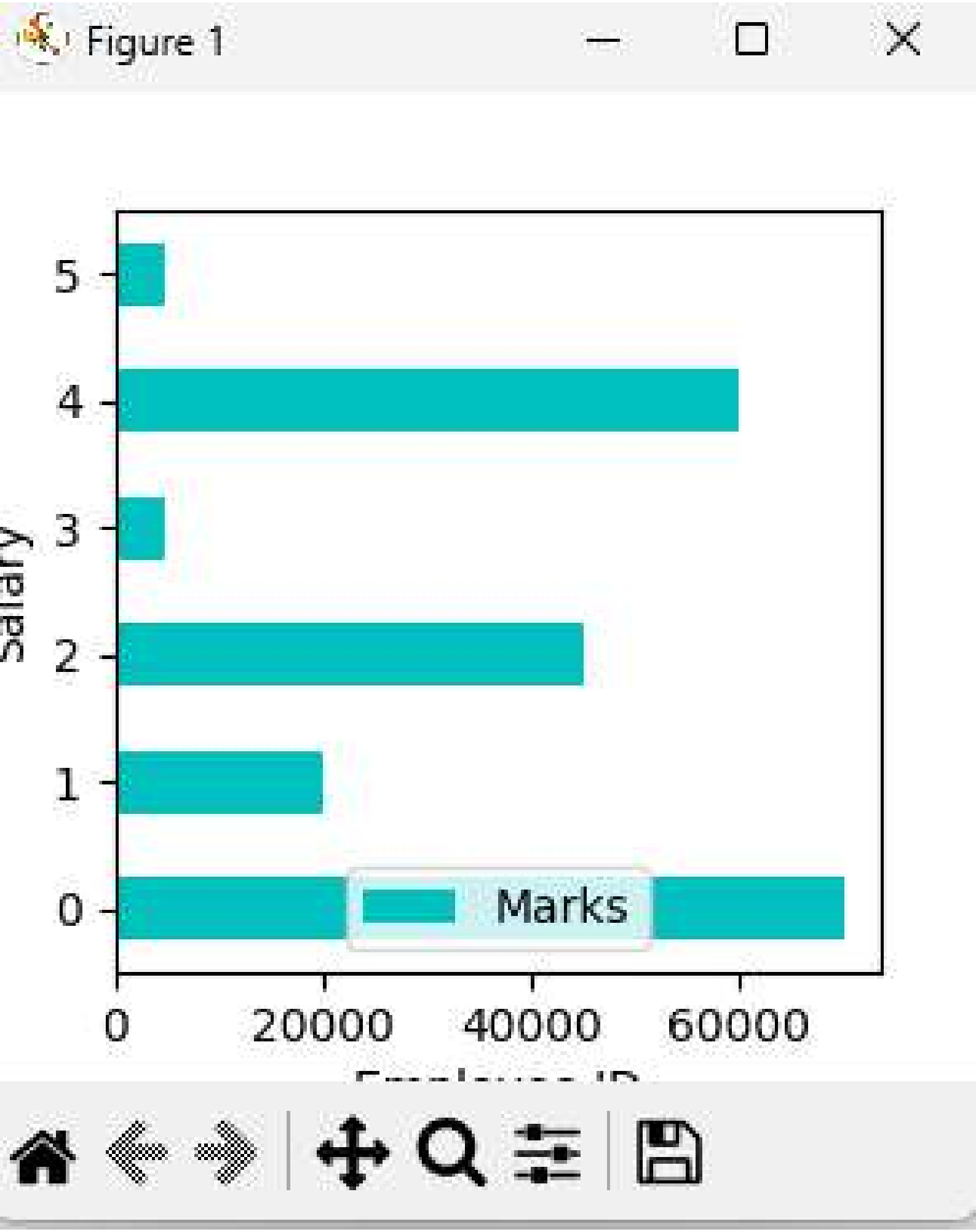


Figure 1
5
4
3
2
1
0
Marks
0
20000
40000
60000

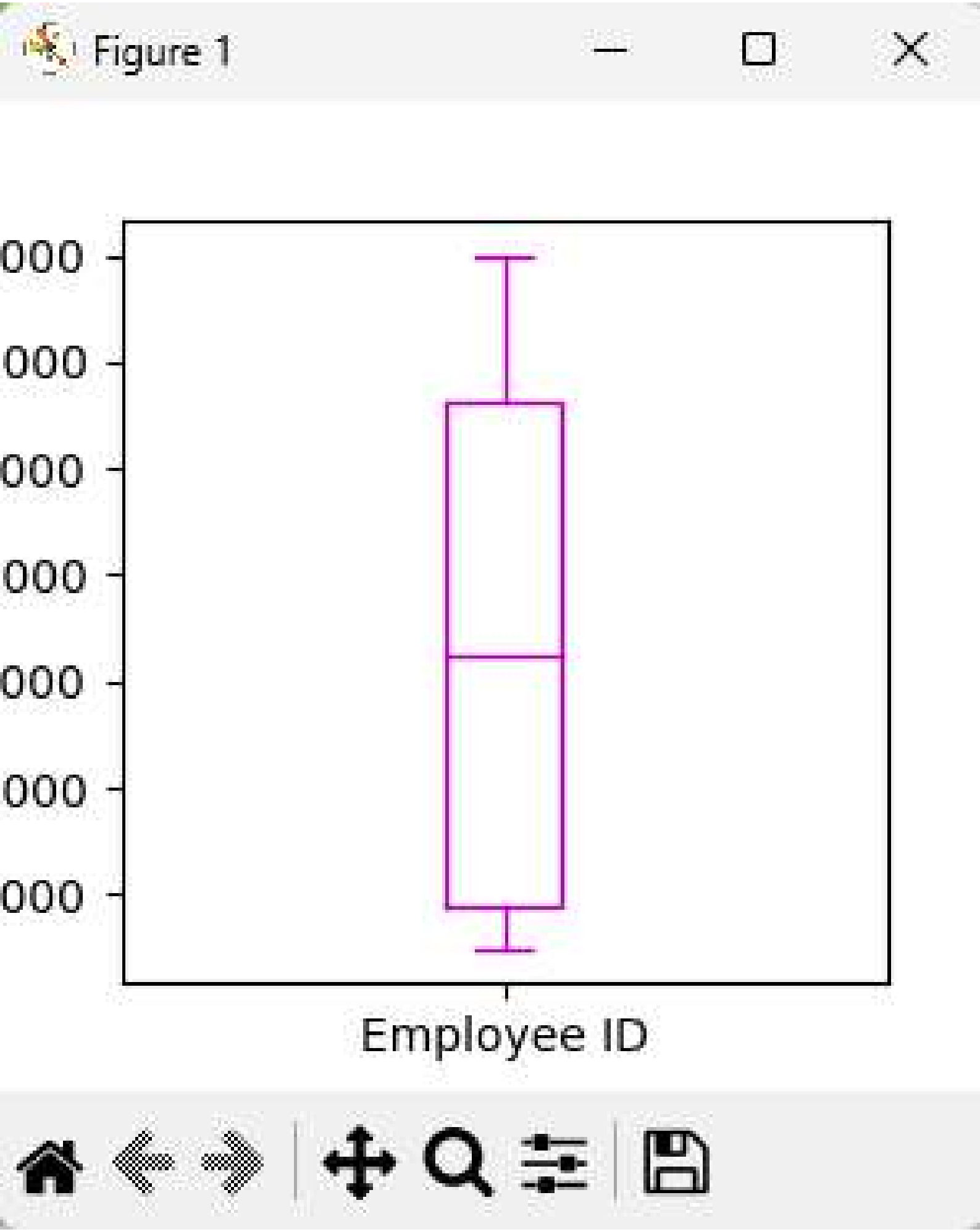


Figure 1
000
000
000
000
000
000
000
Employee ID

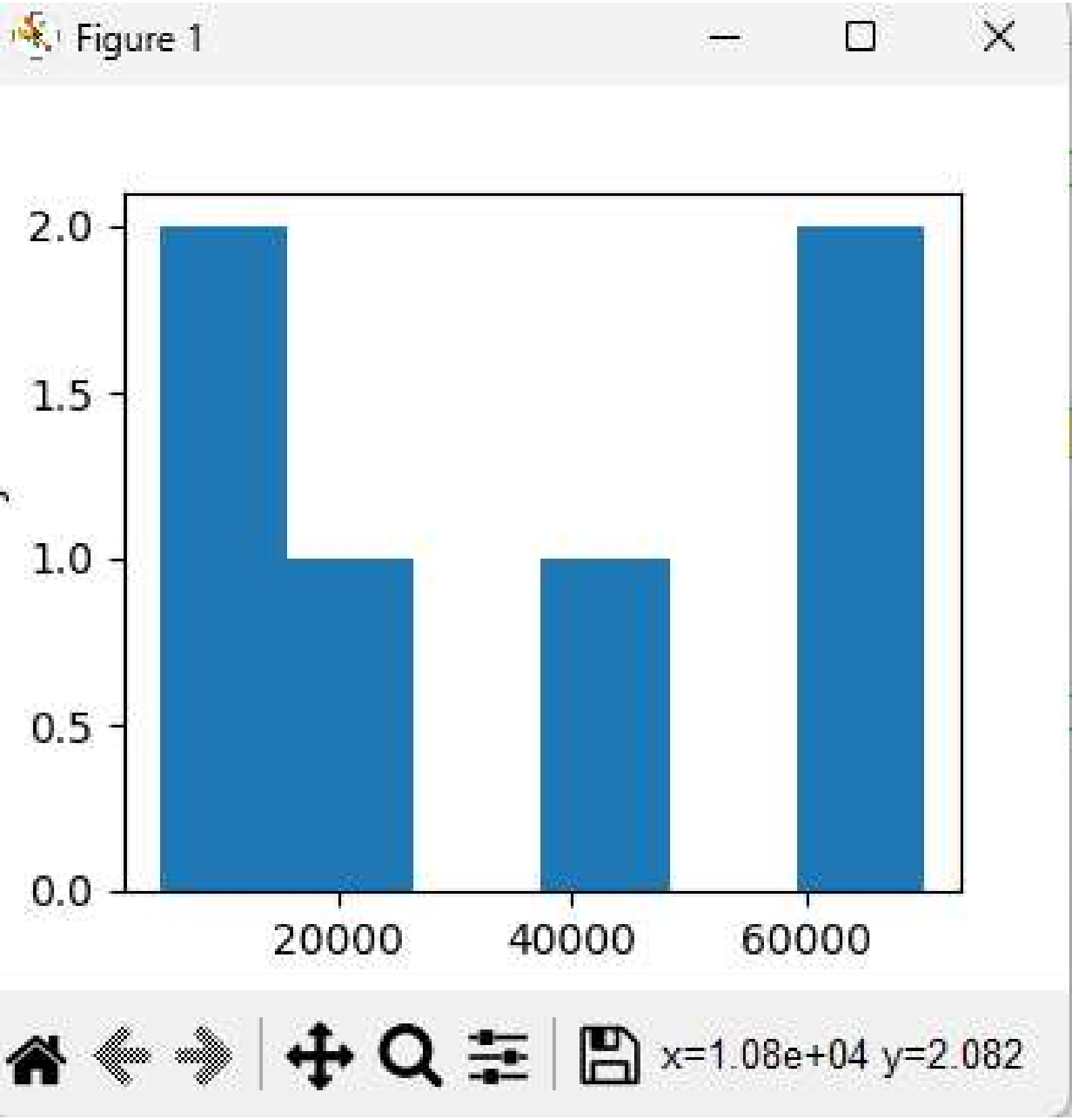


Figure 1
2.0
1.5
1.0
0.5
0.0
20000
40000
60000
x=1.08e+04 y=2.082

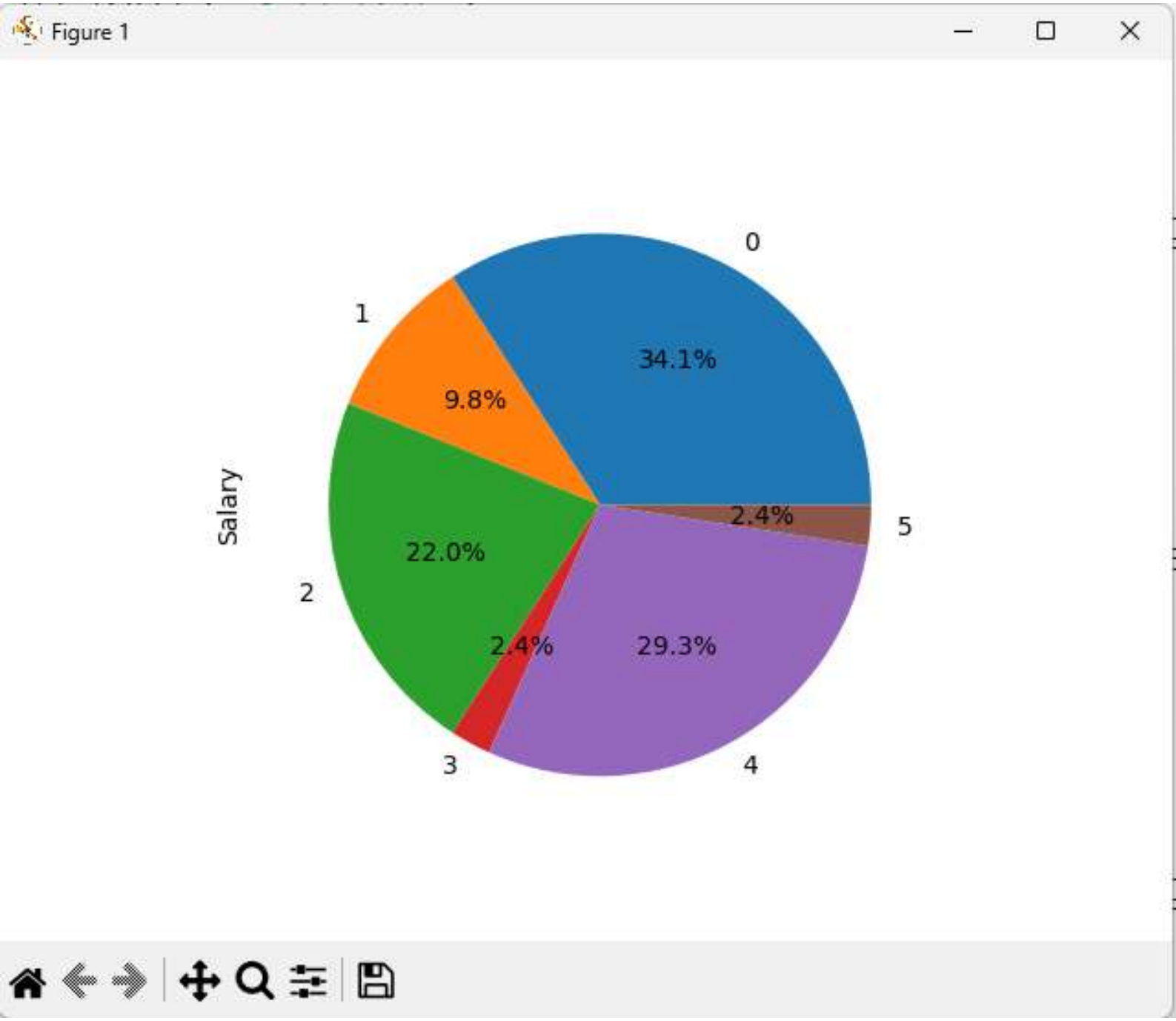


Figure 1
Salary
0
1
2
3
4
5
34.1%
9.8%
22.0%
2.4%
29.3%
2.4%

```
#2.>Write a program to create a PANDAS Data Series as shown below:
#Plot Graphs


salary=pd.Series({'0':70000,'1':20000,'2':45000,'3':5000,'4':60000,'5':5000})
salary.name='SALARY'
salary.index.name='EMPLOYEE ID'
print(salary)

print("\nInvalid/Missing Entries")
print(salary.isnull())

print("\nFILLING Invalid/Missing Entries")
print(salary.fillna(5000))


salary.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('Employee ID')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()
```

```
salary.plot.area(color='b',label='Marks',figsize=(5,5))
plt.xlabel('Employee ID')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

salary.plot.bar(color='c',label='Marks',figsize=(3,3))
plt.xlabel('Employee ID')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

salary.plot.barh(color='c',label='Marks',figsize=(3,3))
plt.xlabel('Employee ID')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

salary.plot.box(color='m',label='Employee ID',figsize=(3,3))
plt.ylabel('Salary')
plt.show()

salary.plot.hist(bins=6,figsize=(3,3))
plt.ylabel('Salary')
plt.show()

salary.plot.pie(label='Marks',autopct='%1.1f%%')
plt.ylabel('Salary')
plt.show()
```

```
EMPLOYEE ID
0    70000
1    20000
2    45000
3     5000
4    60000
5     5000
Name: SALARY, dtype: int64

Invalid/Missing Entries
EMPLOYEE ID
0    False
1    False
2    False
3    False
4    False
5    False
Name: SALARY, dtype: bool

FILLING Invalid/Missing Entries
EMPLOYEE ID
0    70000
1    20000
2    45000
3     5000
4    60000
5     5000
Name: SALARY, dtype: int64
```

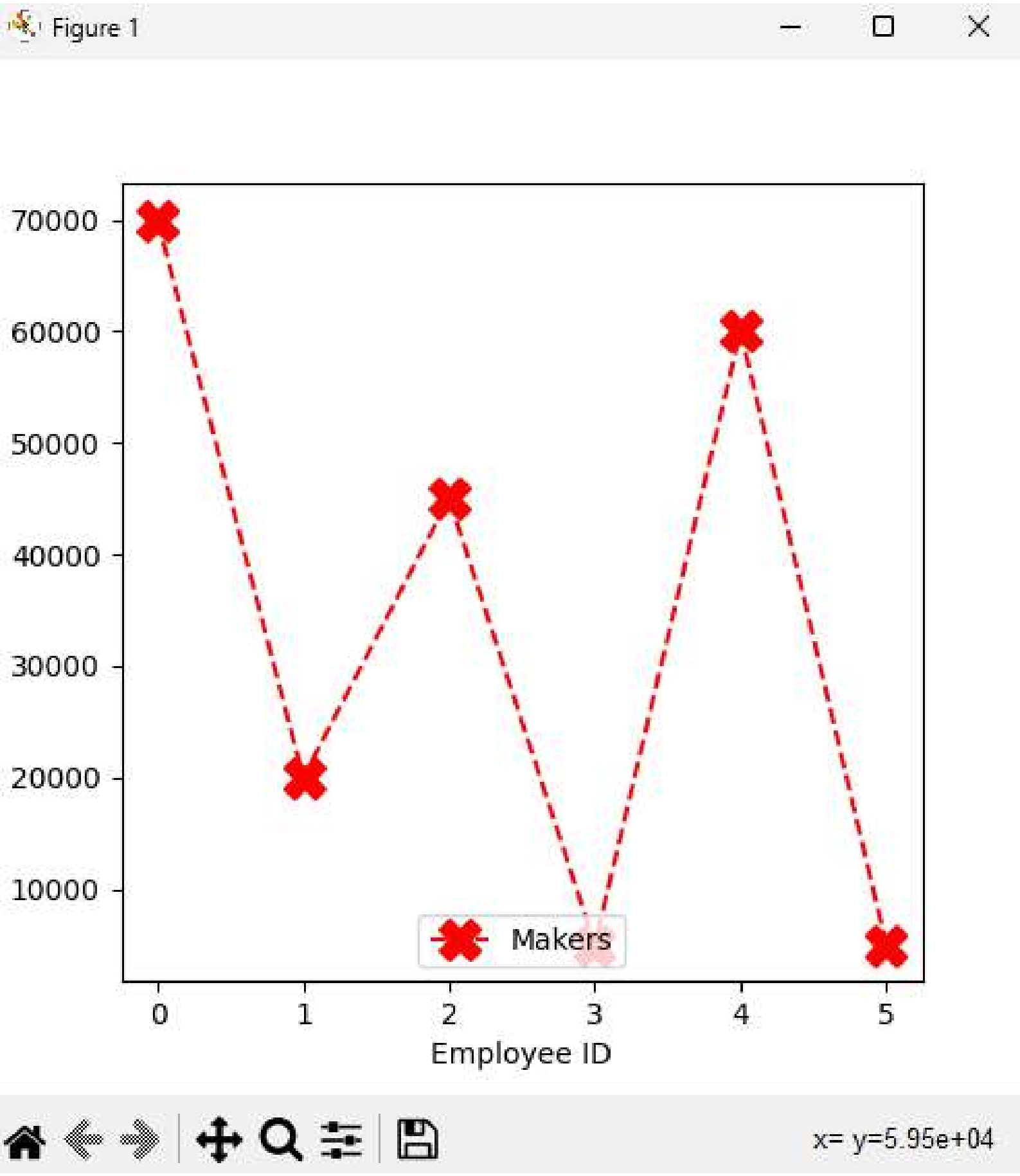


Figure 1
70000
60000
50000
40000
30000
20000
10000
Makers
0
1
2
3
4
5
Employee ID
x= y=5.95e+04

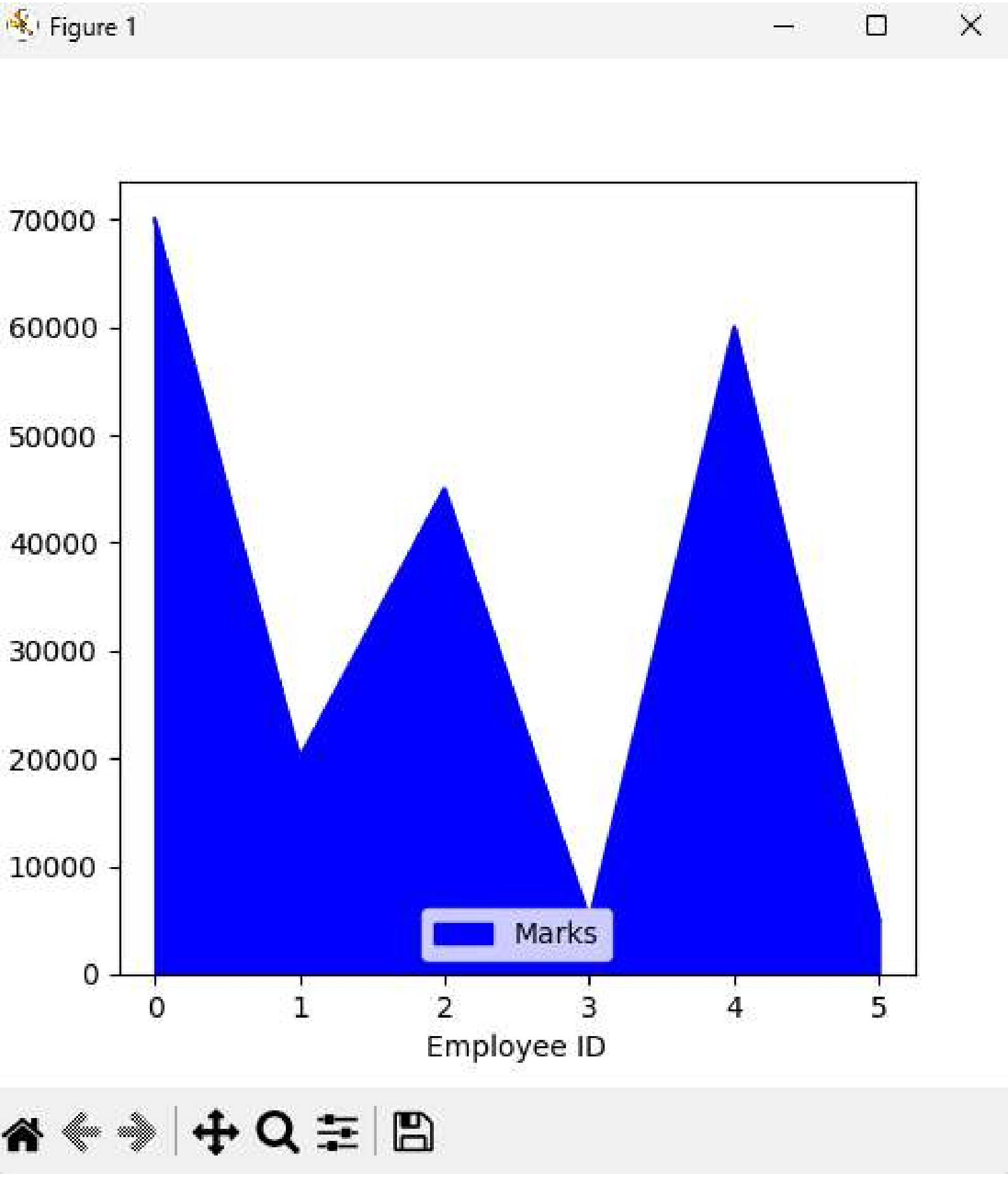


Figure 1
70000
60000
50000
40000
30000
20000
10000
0
0
1
2
3
4
5
Employee ID
Marks

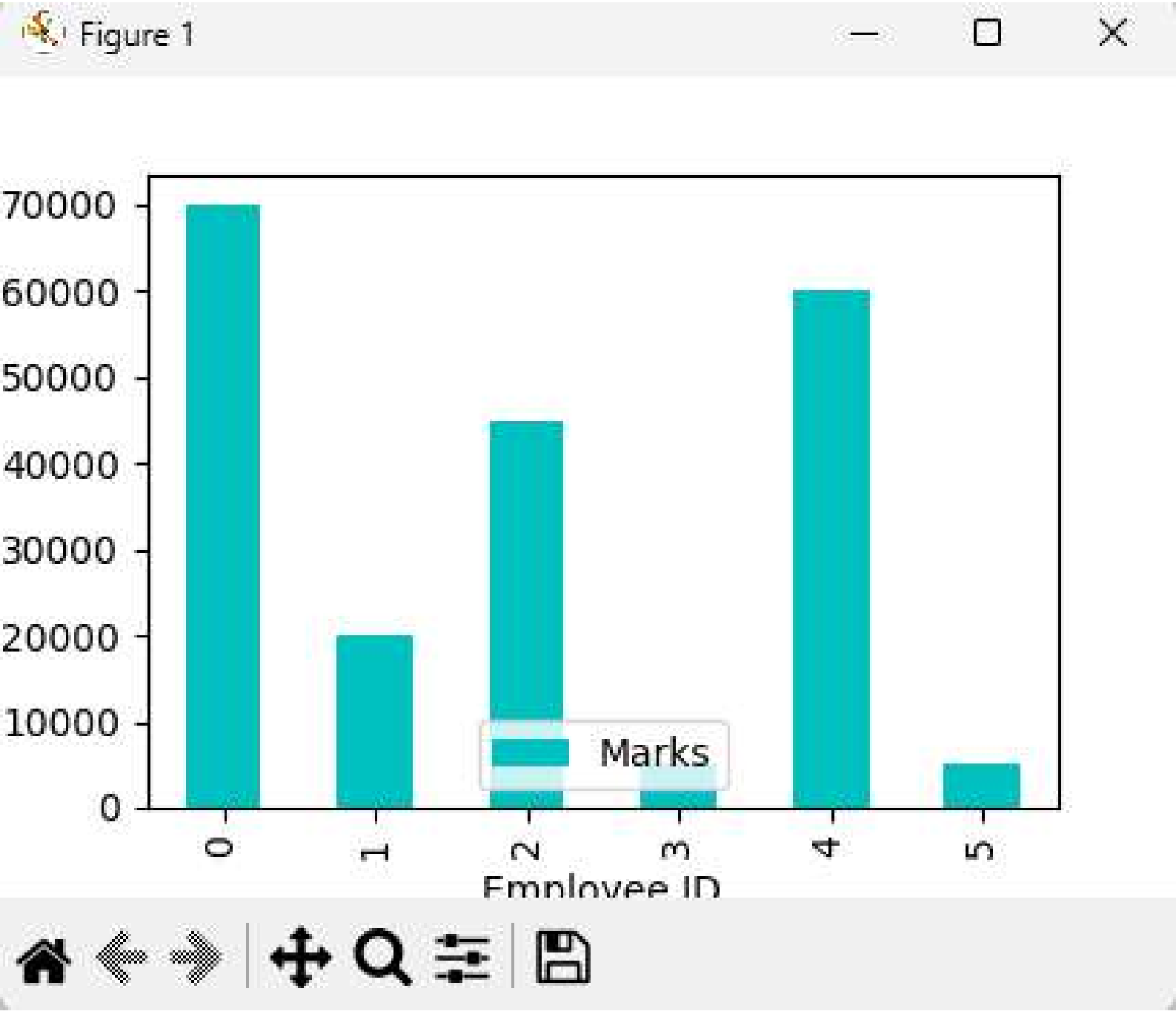


Figure 1
70000
60000
50000
40000
30000
20000
10000
0
Marks
0
1
2
3
4
5
Employee ID

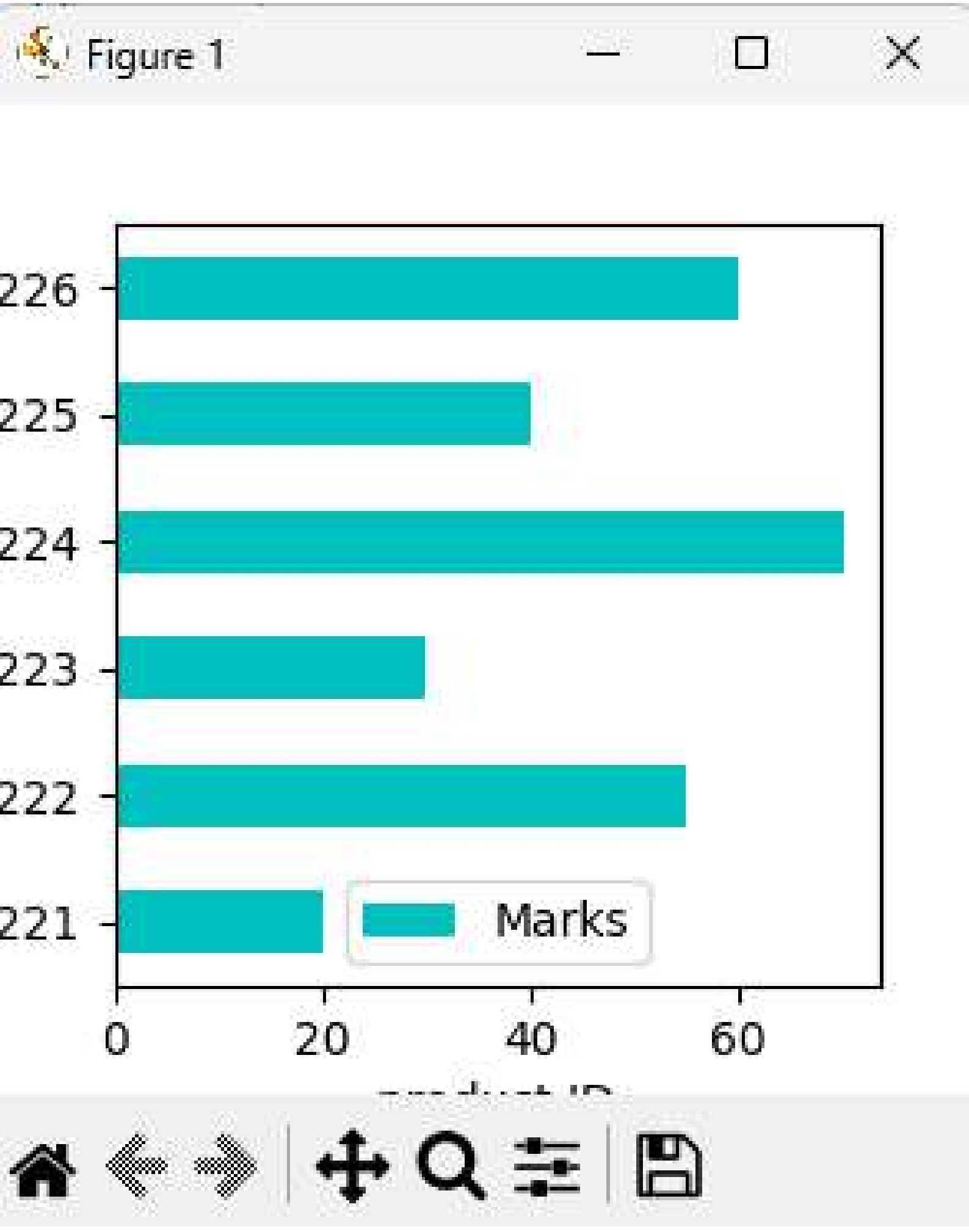


Figure 1
226
225
224
223
222
221
Marks
0
20
40
60

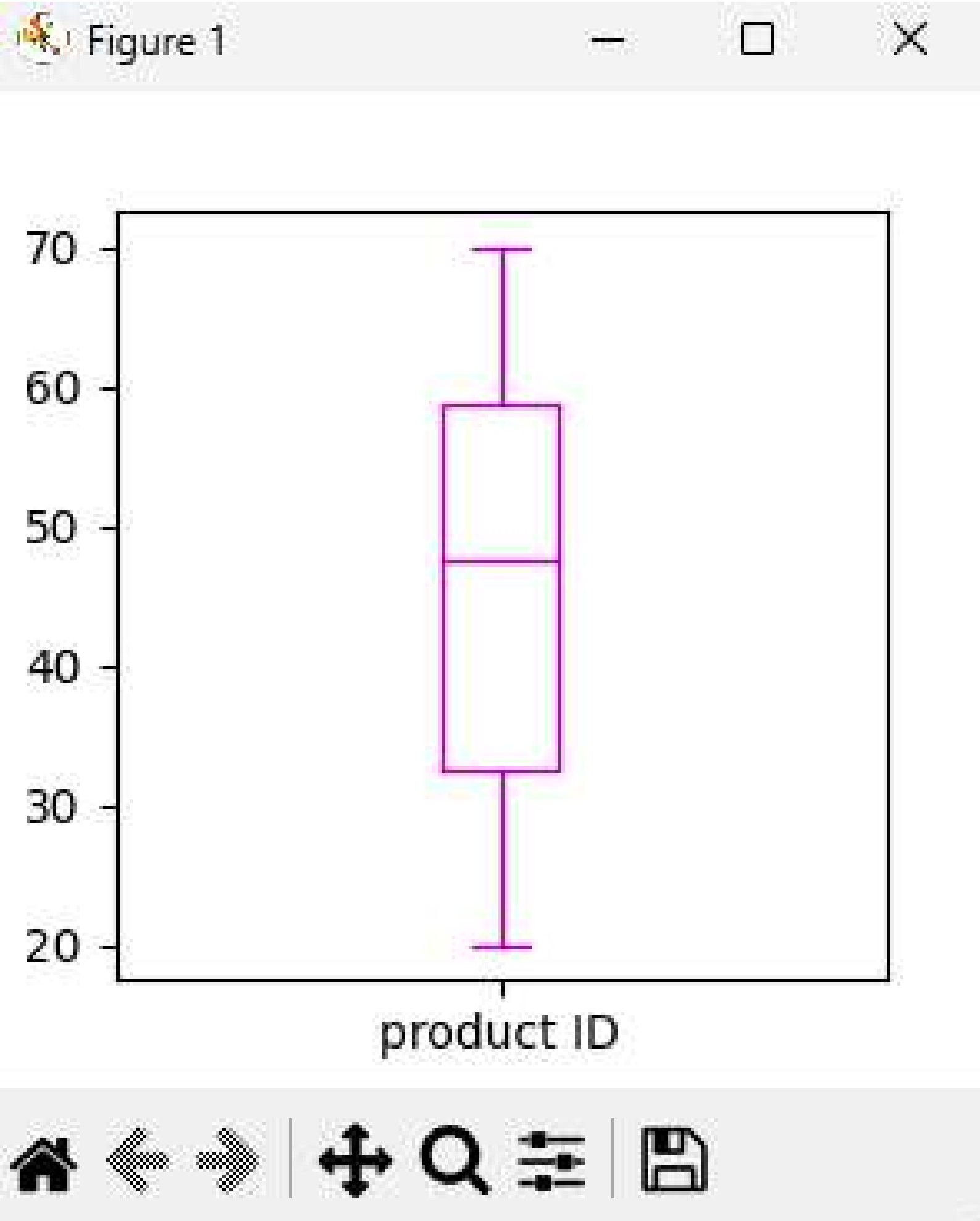


Figure 1
70
60
50
40
30
20
product ID

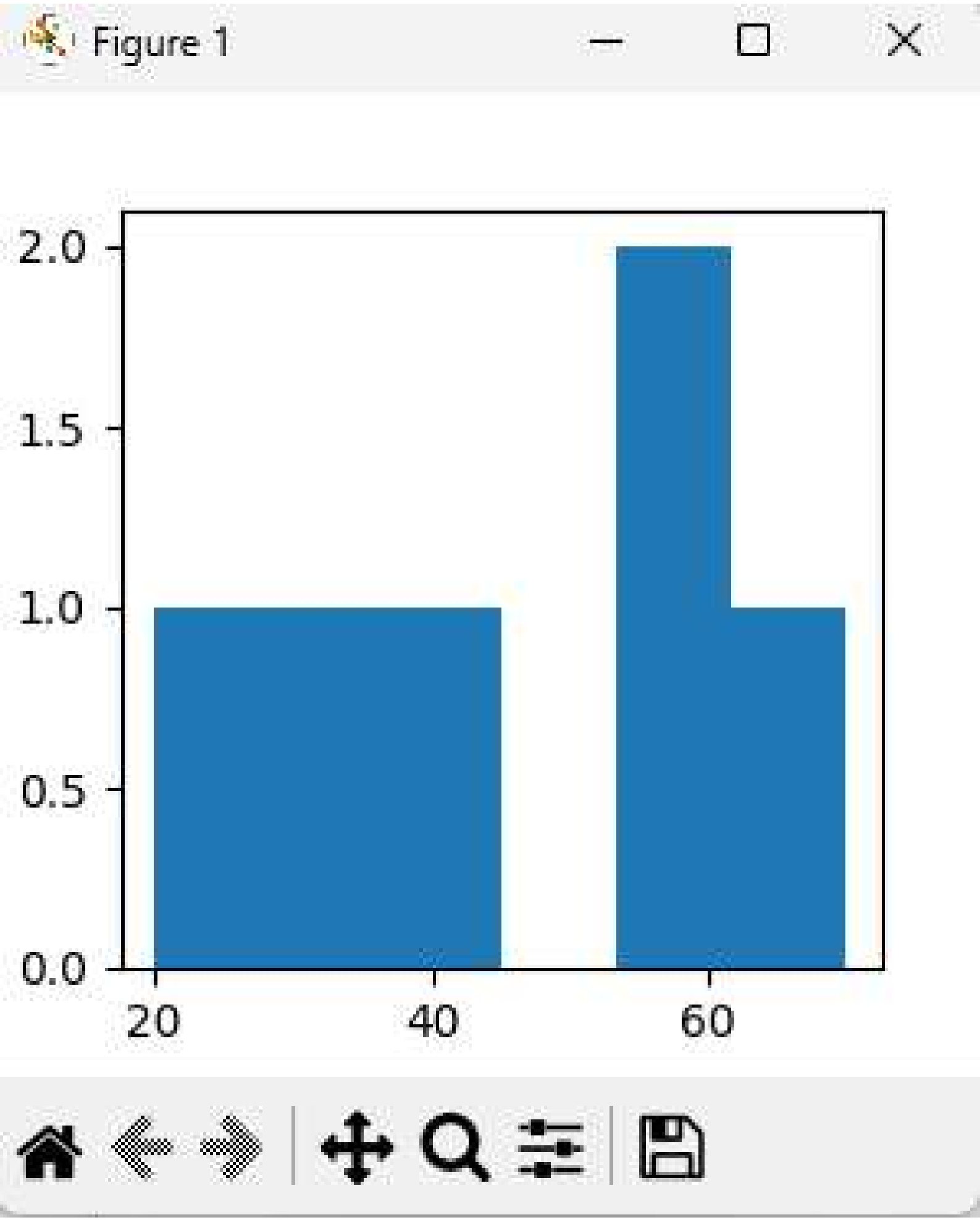


Figure 1
2.0
1.5
1.0
0.5
0.0
20
40
60

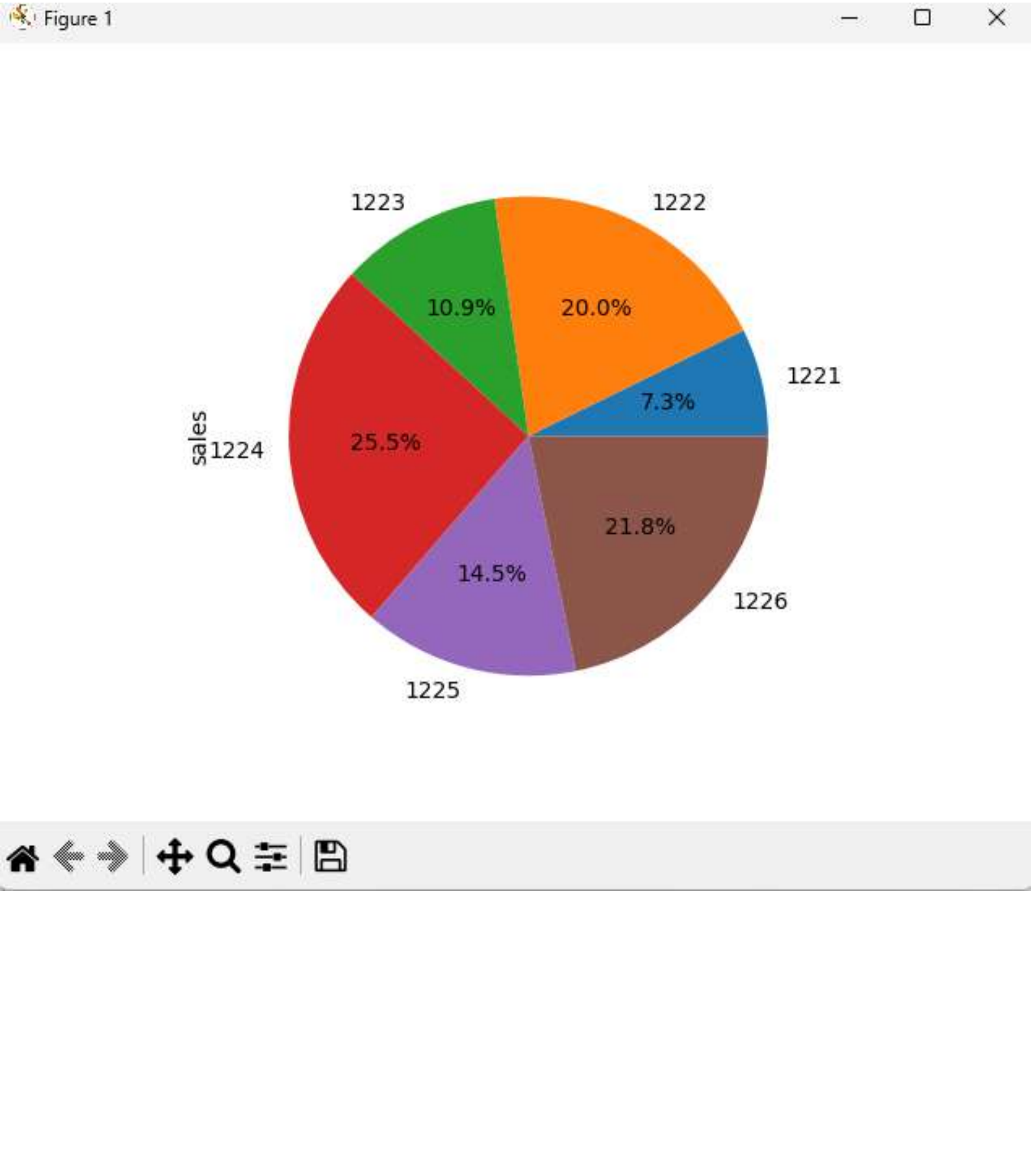


Figure 1
1223
1222
10.9%
20.0%
1221
7.3%
sales
1224
25.5%
21.8%
14.5%
1226
1225

```python
                                                    #GE ASSIGNMENT 4
import pandas as pd
import matplotlib.pyplot as plt


#1.The above table shows sales record of two products, Product 1 and Product 2,
print("\nSALES RECORD OF TWO PRODUCTS\n")
dates = pd.date_range(start='2023-04-10', end='2023-04-16')
sales = {'Product1': [100, 70, 82, 125, 65, 30, 50],
         'Product2': [20, 78, 65, 100, 50, 80, 55]}
df = pd.DataFrame(sales, index=dates)
print(df)

# plot line plot (scatter plot)
df.plot(style='.-', figsize=(8,5))
plt.title('Sales Record')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
print("\n")

# plot bar plot
df.plot(kind='bar', figsize=(8,5))
plt.title('Sales Record')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
print("\n")
```

```
# plot bar plot
df.plot(kind='bar', figsize=(8,5))
plt.title('Sales Record')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
print("\n")

# plot area plot
df.plot(kind='area', figsize=(8,5))
plt.title('Sales Record')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.show()
print("\n")

# plot pie chart for Product 1 sales
df['Product1'].plot(kind='pie', figsize=(5,5))
plt.title('Product 1 Sales')
plt.ylabel('')
plt.show()
```

SALES RECORD OF TWO PRODUCTS

	Product1	Product2
2023-04-10	100	20
2023-04-11	70	78
2023-04-12	82	65
2023-04-13	125	100
2023-04-14	65	50
2023-04-15	30	80
2023-04-16	50	55


Figure 1
Sales Record
Product1
Product2
Sales
120
100
80
60
40
20
10
Apr
2023
11
12
13
14
15
16

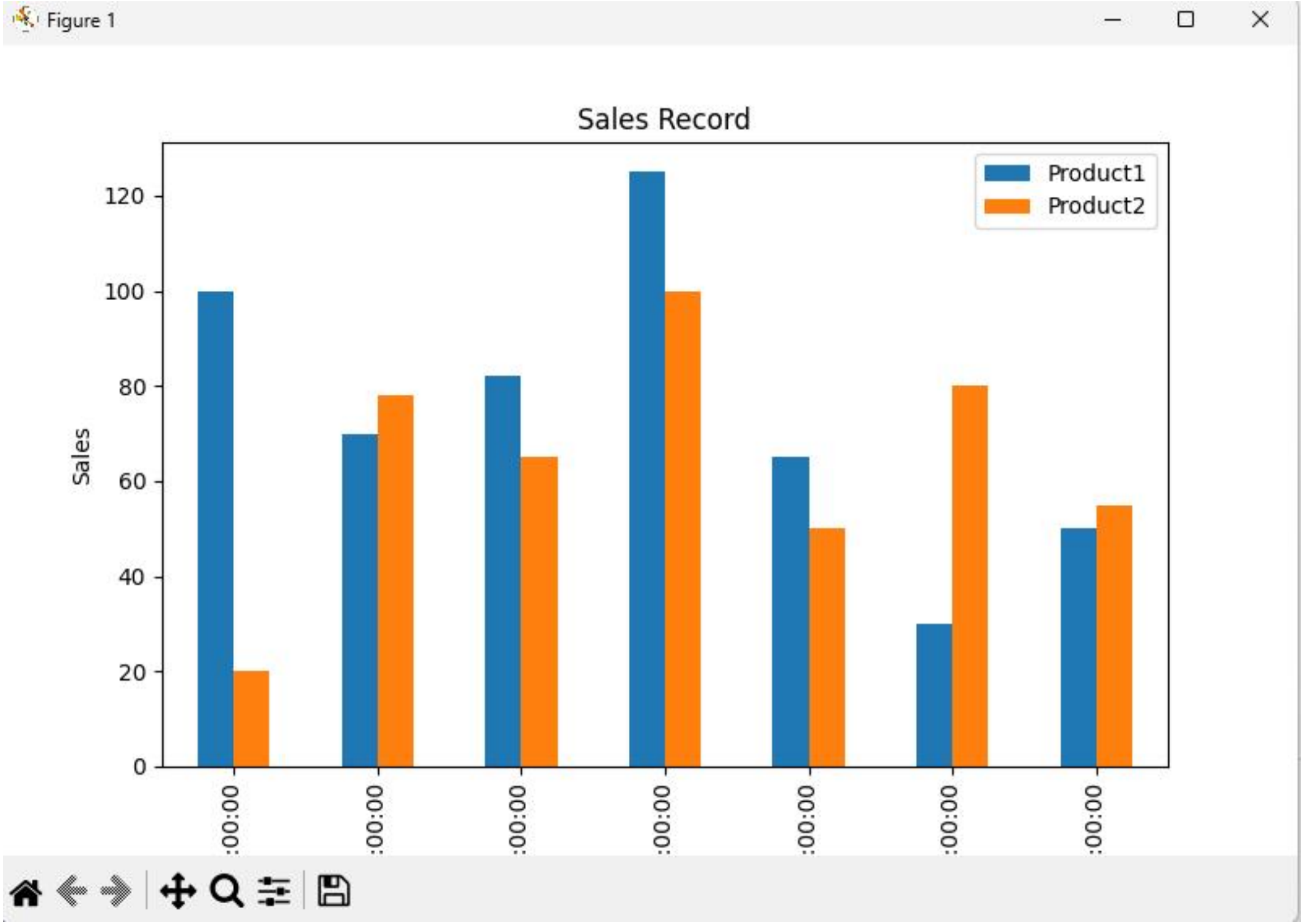


Figure 1
Sales Record
Product1
Product2
Sales
120
100
80
60
40
20
0
:00:00
:00:00
:00:00
:00:00
:00:00
:00:00
:00:00

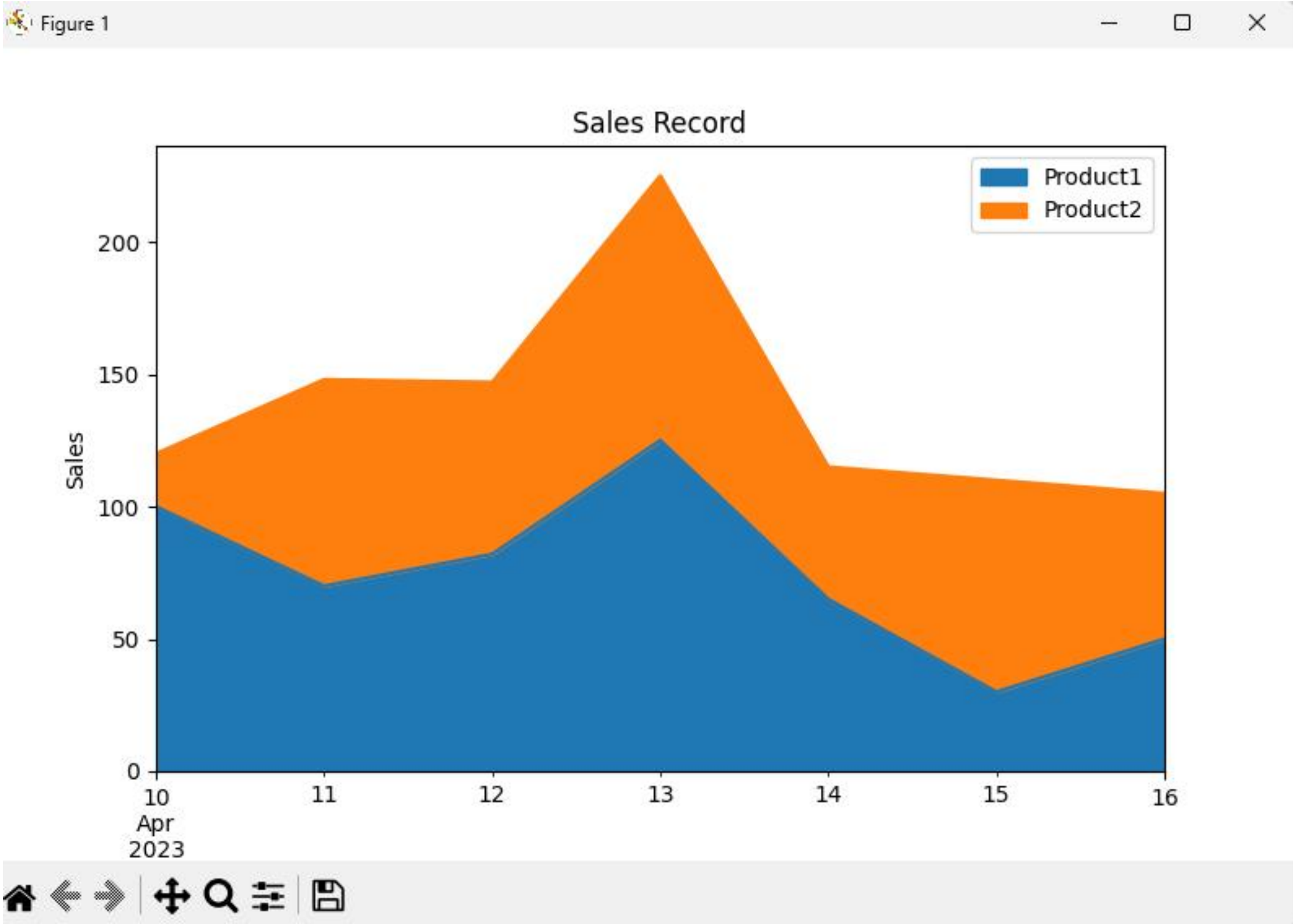


Figure 1
Sales Record
Product1
Product2
Sales
200
150
100
50
0
10
Apr
2023
11
12
13
14
15
16

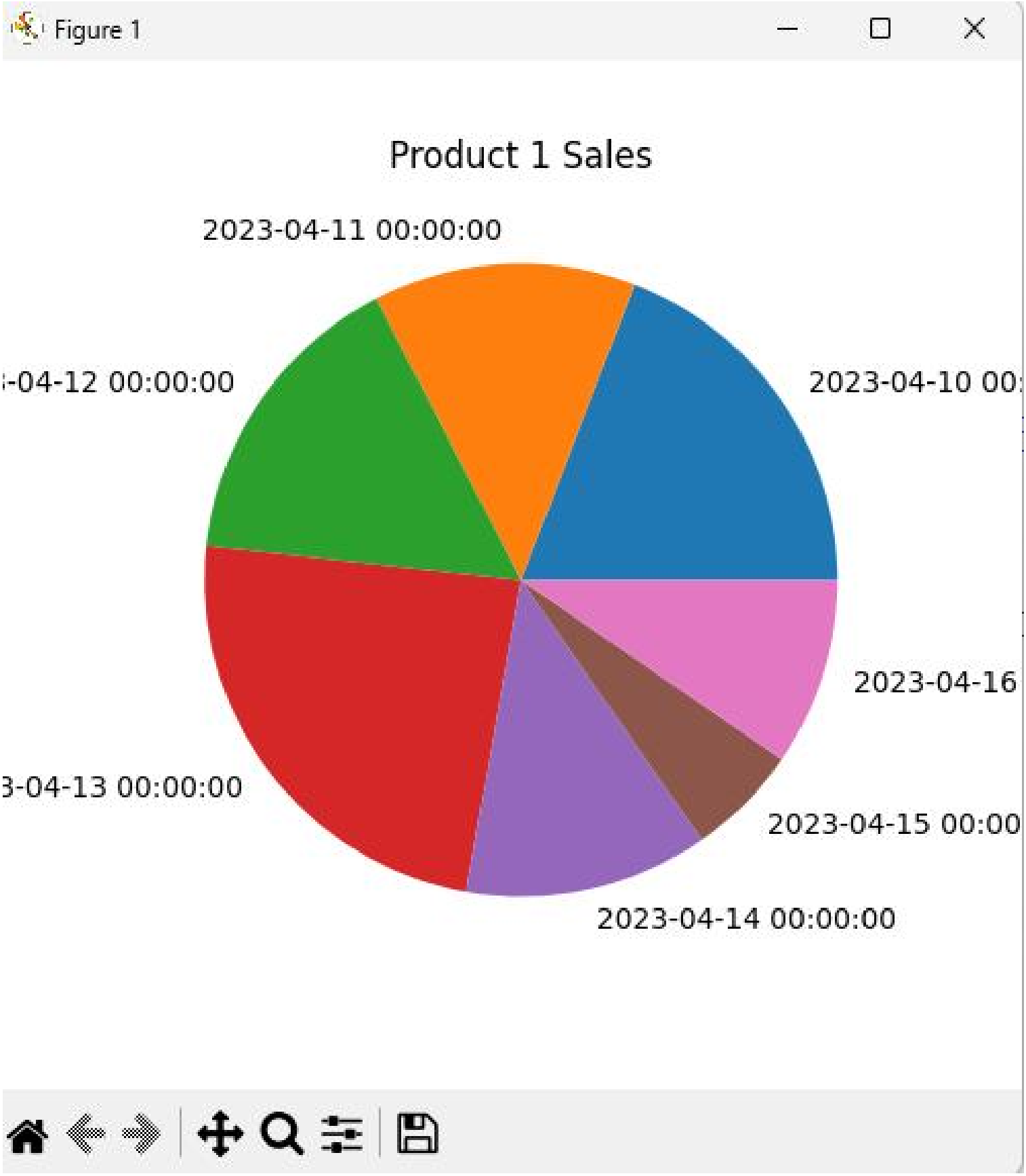


Figure 1
Product 1 Sales
2023-04-11 00:00:00
-04-12 00:00:00
2023-04-10 00:
2023-04-16
3-04-13 00:00:00
2023-04-15 00:00
2023-04-14 00:00:00

```
#2.The above table shows Report Card of STUDENT ,displaying marks scored by the student:
print("\nREPORT CARD OF STUDENTS\n")
StudentData = {'SUBJECTS':['ENGLISH', 'MATHS', 'SCIENCE', 'FRENCH',],
'2018':[85,73,98,88],
'2019': [60,80,58,96],
'2020':[90,64,74,87]}
data = pd.DataFrame(StudentData)
print(data)
print("\nAFTER  ADDING INFORMATION\n")
data.loc['Total'] = [' ', data['2018'].sum(), data['2019'].sum(), data['2020'].sum()]
data.loc['Percentage'] = [' ', (data['2018'].sum())/8, (data['2019'].sum())/8, (data['2020'].sum())/8]
print(data)


data.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()

data.plot.area(color='b',label='Marks',figsize=(5,5))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()
```

```
data.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()

data.plot.area(color='b',label='Marks',figsize=(5,5))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()

data.plot.bar(color='c',label='Marks',figsize=(3,3))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()
```

```
data.plot.bar(color='c',label='Marks',figsize=(3,3))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()

data.plot.barh(color='c',label='Marks',figsize=(3,3))
plt.xlabel('MARKS')
plt.ylabel('YEAR')
plt.legend(loc='lower center')
plt.show()

data.plot.box(color='m',label='product ID',figsize=(3,3))
plt.ylabel('YEAR')
plt.show()

data.plot.hist(bins=6,figsize=(3,3))
plt.ylabel('YEAR')
plt.show()

data['2018'].plot.pie(autopct='%1.2f%%',figsize=(3,3))
plt.ylabel('YEAR')
plt.show()
```

REPORT CARD OF STUDENTS

	SUBJECTS	2018	2019	2020
0	ENGLISH	85	60	90
1	MATHS	73	80	64
2	SCIENCE	98	58	74
3	FRENCH	88	96	87

AFTER ADDING INFORMATION

	SUBJECTS	2018	2019	2020
0	ENGLISH	85.0	60.0	90.00
1	MATHS	73.0	80.0	64.00
2	SCIENCE	98.0	58.0	74.00
3	FRENCH	88.0	96.0	87.00
Total		344.0	294.0	315.00
Percentage		86.0	73.5	78.75

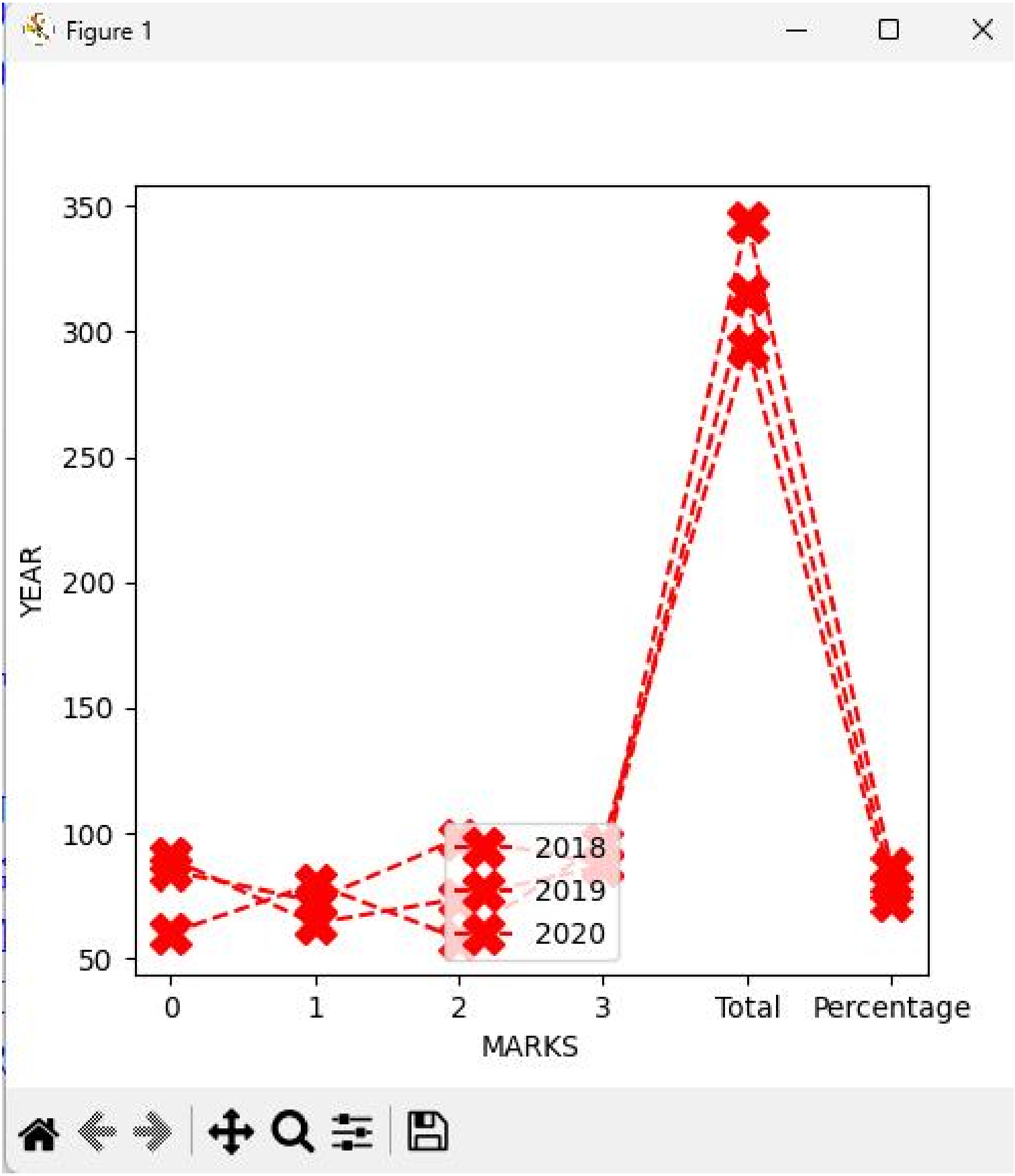


Figure 1
350
300
250
200
150
100
50
YEAR
2018
2019
2020
0
1
2
3
Total
Percentage
MARKS

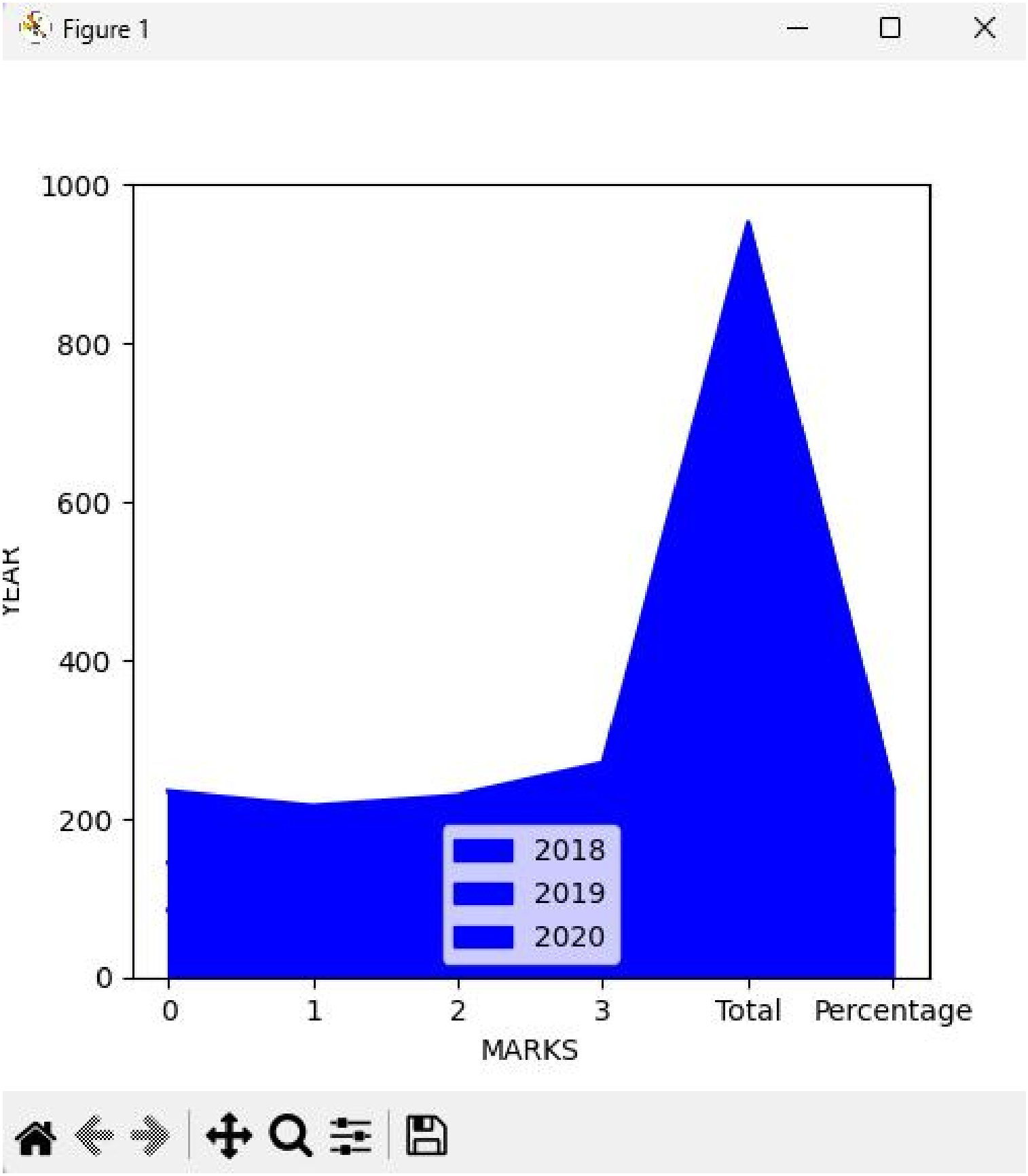


Figure 1
1000
800
600
400
200
0
YEAR
0
1
2
3
Total
Percentage
MARKS
2018
2019
2020

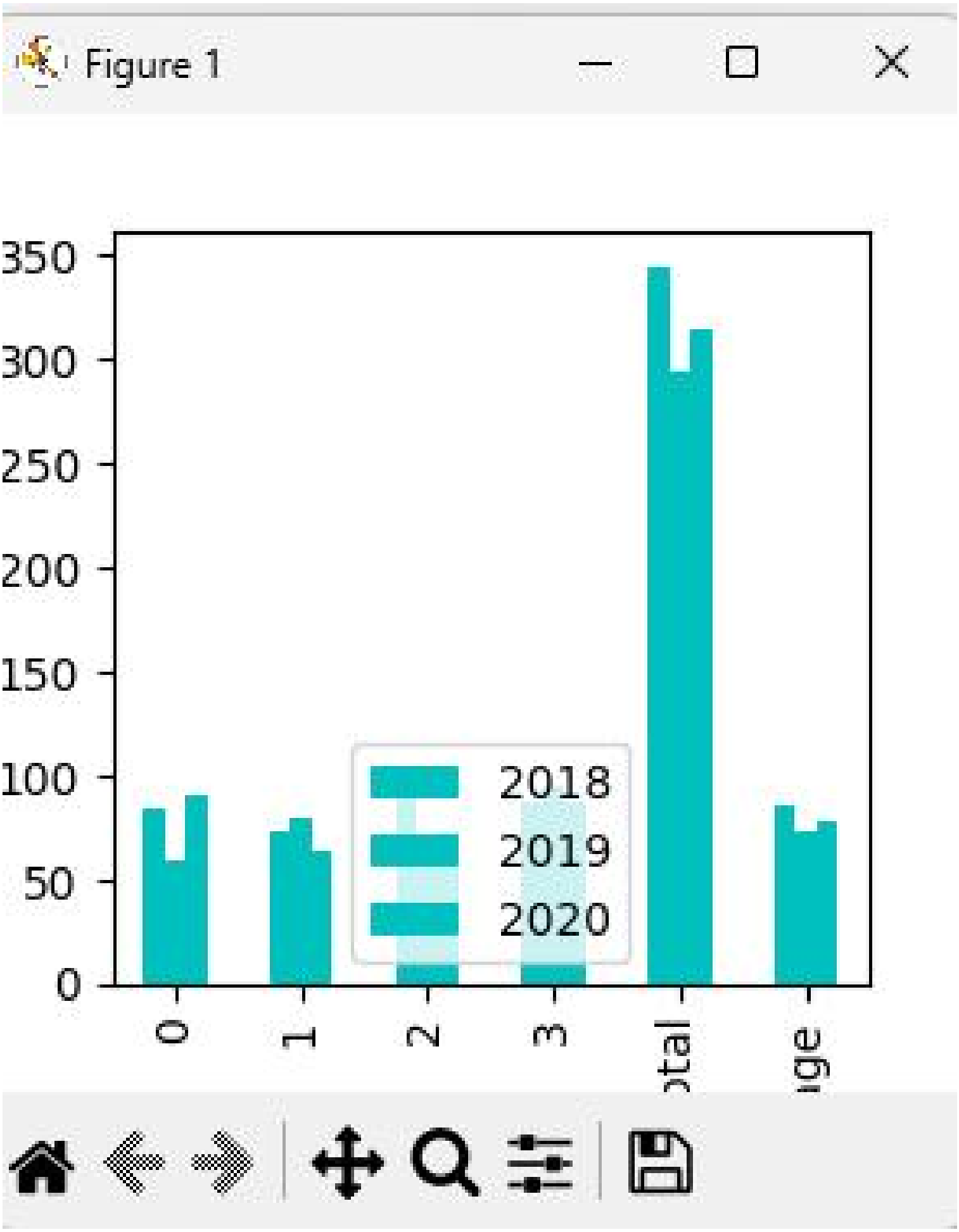


Figure 1
350
300
250
200
150
100
50
0
2018
2019
2020
0
1
2
3
xtal
ige

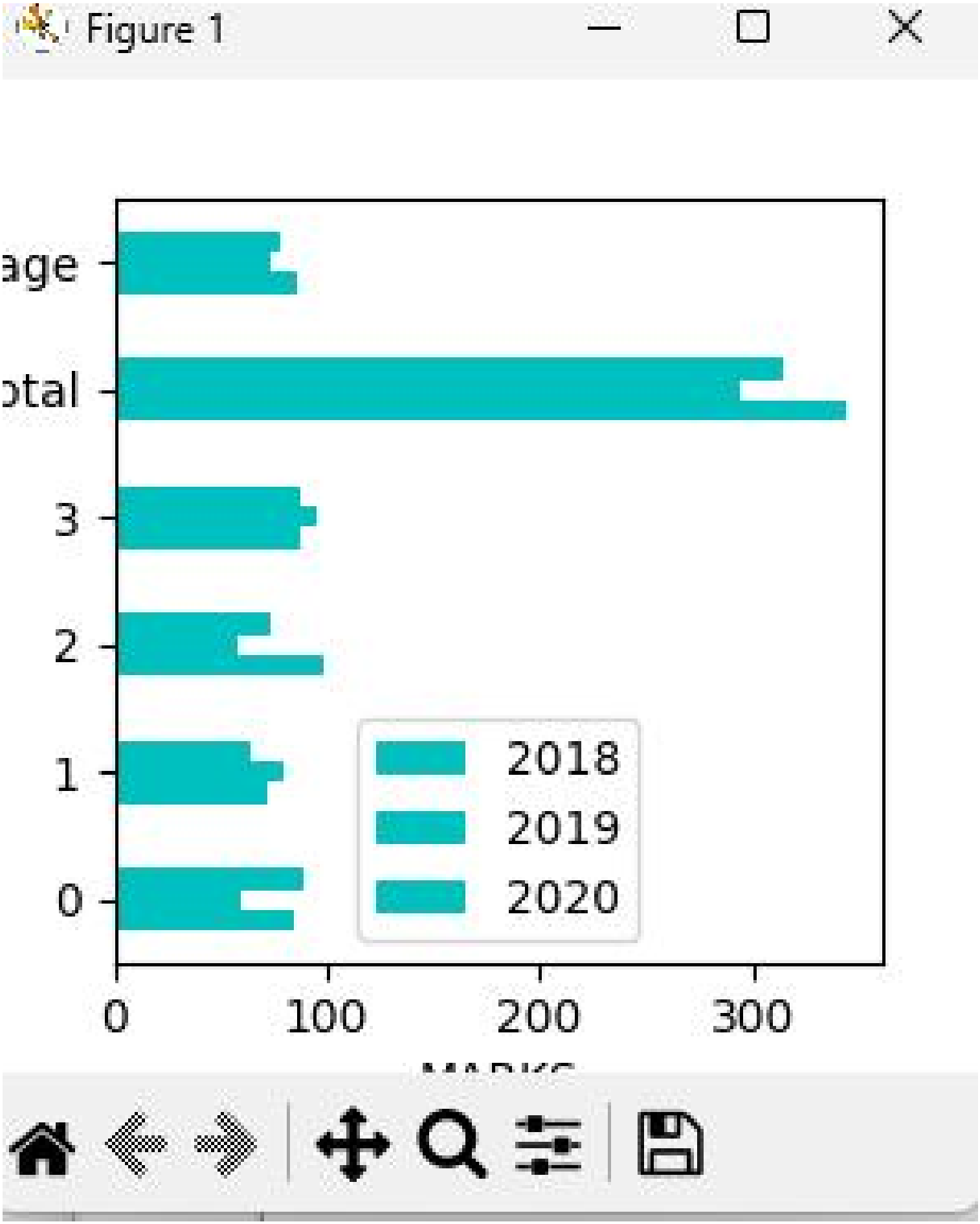


Figure 1
age
otal
3
2
1
0
2018
2019
2020
0
100
200
300

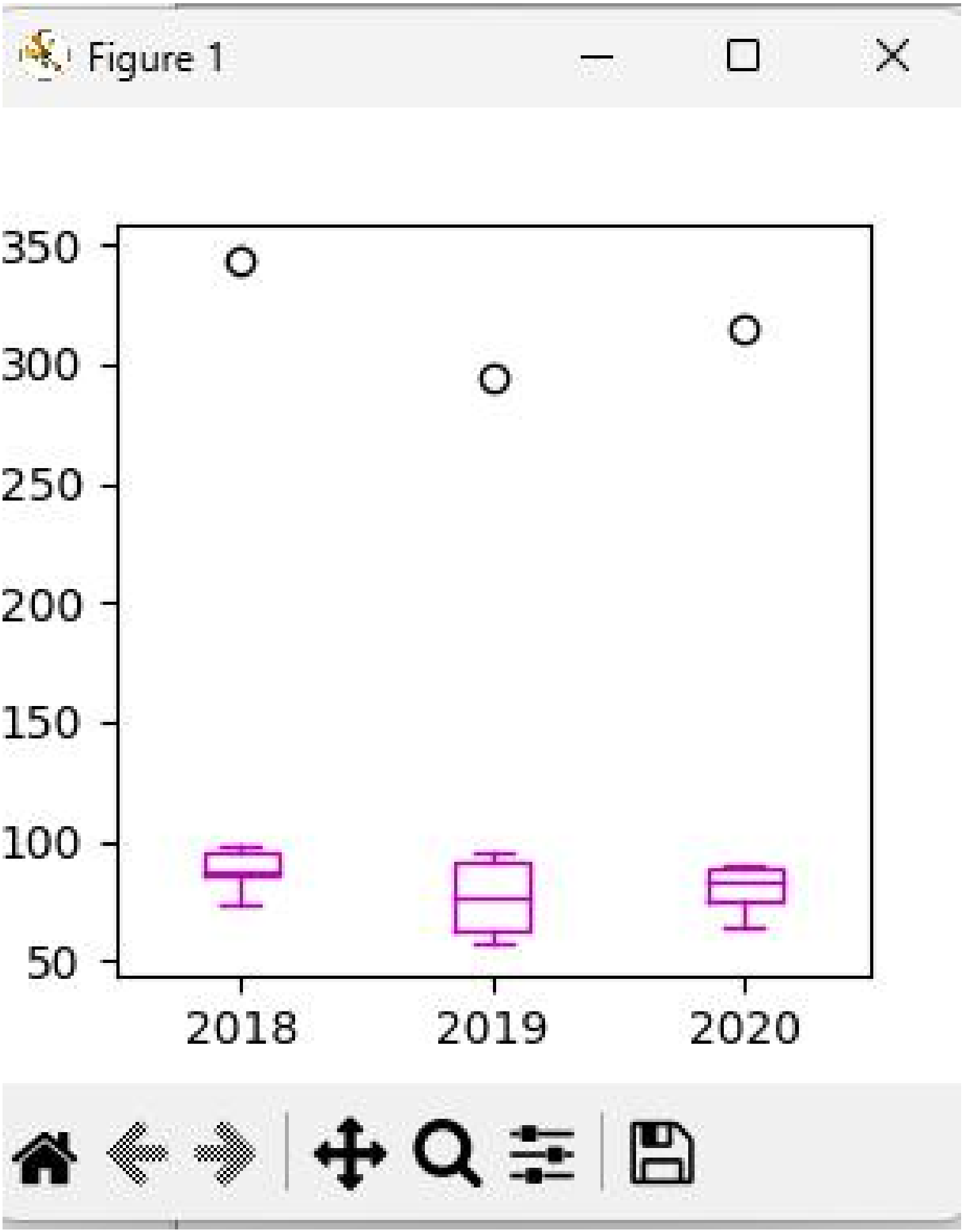


Figure 1
350
300
250
200
150
100
50
2018
2019
2020

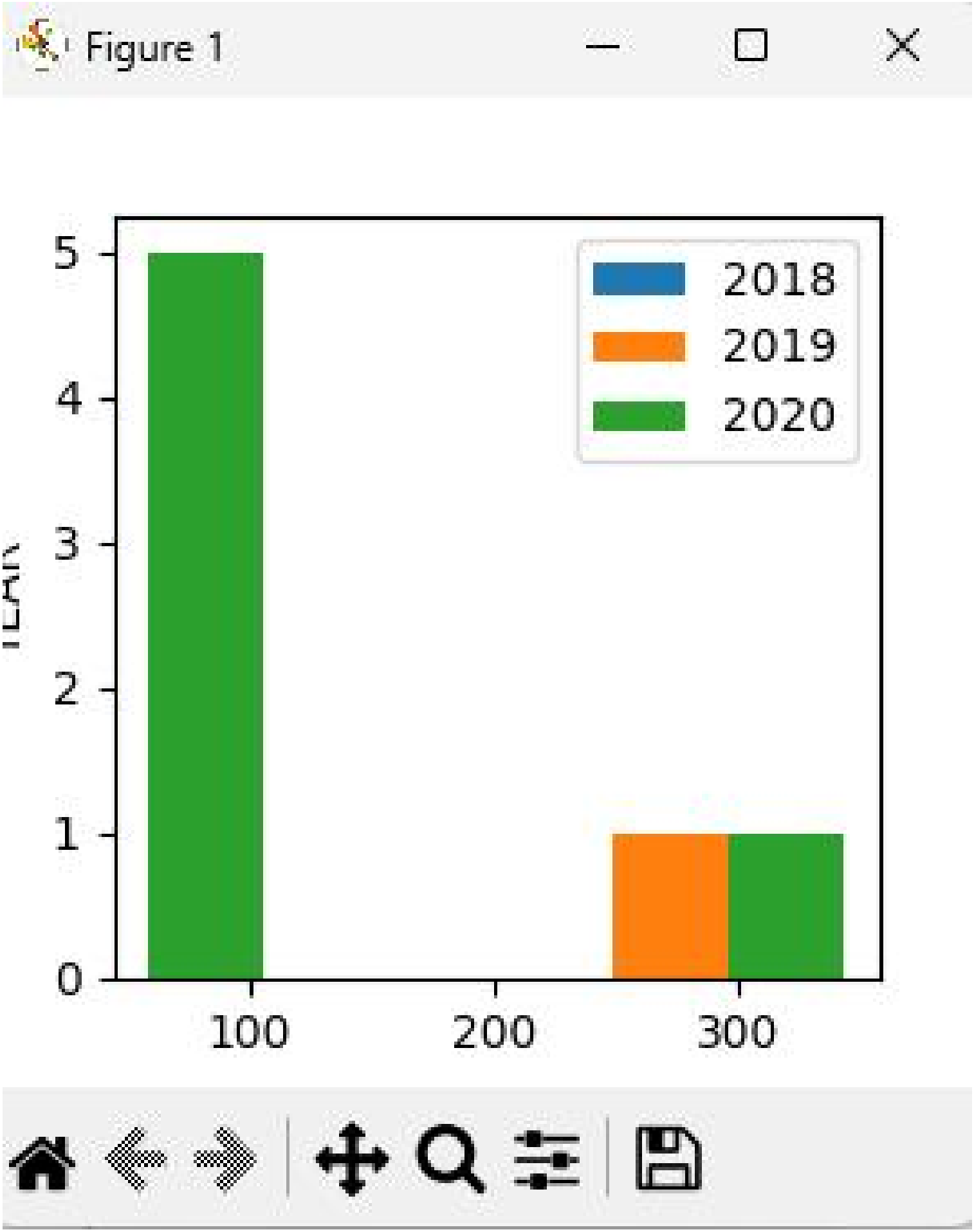


Figure 1
2018
2019
2020
5
4
3
2
1
0
100
200
300

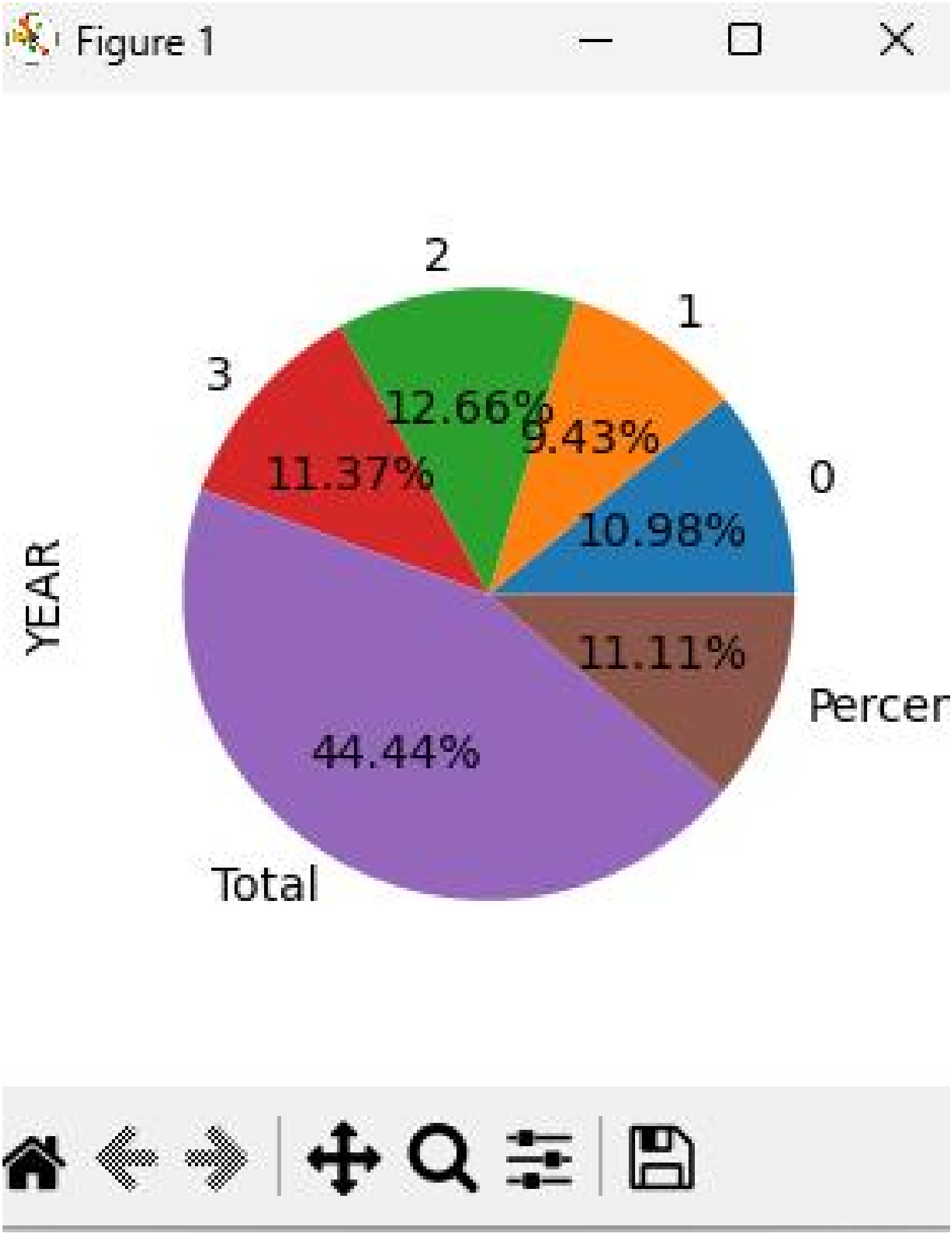


Figure 1
2
1
3
0
12.66%
9.43%
11.37%
10.98%
11.11%
44.44%
YEAR
Percer
Total

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
                                        #GE ASSIGNEMT 5
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = {'EMP ID': [1,2,3,4,5,6,7,8,9,10],
 'EMP NAME': ['satish','reeya','jay','rahul','roy','jay','vishal','serah','vishal','Prachi'],
 'Age': [21,23,40,35,26,28,29,21,29,22],
 'Salary':[50000,75000,100000,np.nan,45000,100000,np.nan,55000,np.nan,60000],
 'EMP Credits':[3.8,4,5,4.2,3.9,4.5,5,3.7,5,4.3],
 'Joining date':["01-11-2017",np.nan,"22-09-2015","11-10-2016","08-01-2017","22-1-2018",'22-09-2015', '05-01-2016', '0
df = pd.DataFrame(data)
print(df)
print('-----------------')
df['Joining date'] = pd.to_datetime(df['Joining date'])
# Add new column with month names
df['month_name'] = df['Joining date'].dt.strftime('%d %B %Y')
print(df)

print('before filling\n',df)
print()
df["Salary"].fillna(50000,inplace=True)
df["Joining date"].fillna("01-01-2018",inplace=True)
print('after filling\n',df)
print("..........................................................................")
print('before duplicate\n',df)
print()
print(df.duplicated())
print(df.drop_duplicates())
print("..........................................................................")

max_credit = df['EMP Credits'].max()
max_credit_emp = df.loc[df['EMP Credits'] == max_credit, 'EMP NAME'].values.tolist()
print('\nthe max credit is',max_credit)
print('\nnames of max credit is',max_credit_emp)
```

Ln: 1 Col: 63

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
max_credit = df['EMP Credits'].max()
max_credit_emp = df.loc[df['EMP Credits'] == max_credit, 'EMP NAME'].values.tolist()
print('\nthe max credit is',max_credit)
print('\nnames of max credit is',max_credit_emp)
print("..........................................................................")

df[['EMP Credits', 'Salary']].plot(kind='bar', x='EMP Credits', y='Salary', rot=0)
plt.title('Employee Credits vs Salary')
plt.xlabel('Employee Credits')
plt.ylabel('Salary')
plt.show()
df['EMP Credits'].plot(kind='pie')
plt.title('Employee Credits')
plt.show()
print("..........................................................................")
plt.figure(figsize=(8,4))
plt.subplot(1,2,1) # PLOT 1: Line Plot
df['Age'].plot.line(color='b', marker='X', x='EMP Credits',
 #xlabel = 'EMP Credits',
 #ylabel = 'Age',
 title='EMP Credits vs Age')
plt.show()
plt.subplot(1,2,2) # PLOT 2: Line Plot
df['Salary'].plot.line(color='b', marker='X', x='EMP Credits',
 #xlabel = 'EMP Credits',
#ylabel = 'Salary',
title='EMP Credits vs Salary')
plt.show()
print("..........................................................................")



#2..
import pandas as pd
```

Ln: 1 Col: 63

	EMP ID	EMP NAME	Age	Salary	EMP Credits	Joining date
0	1	satish	21	50000.0	3.8	01-11-2017
1	2	reeya	23	75000.0	4.0	NaN
2	3	jay	40	100000.0	5.0	22-09-2015
3	4	rahul	35	NaN	4.2	11-10-2016
4	5	roy	26	45000.0	3.9	08-01-2017
5	6	jay	28	100000.0	4.5	22-1-2018
6	7	vishal	29	NaN	5.0	22-09-2015
7	8	serah	21	55000.0	3.7	05-01-2016
8	9	vishal	29	NaN	5.0	06-02-2018
9	10	Prachi	22	60000.0	4.3	05-01-2016

	EMP ID	EMP NAME	Age	Salary	EMP Credits	Joining date	month_name
0	1	satish	21	50000.0	3.8	2017-01-11	11 January 2017
1	2	reeya	23	75000.0	4.0	NaT	NaN
2	3	jay	40	100000.0	5.0	2015-09-22	22 September 2015
3	4	rahul	35	NaN	4.2	2016-11-10	10 November 2016
4	5	roy	26	45000.0	3.9	2017-08-01	01 August 2017
5	6	jay	28	100000.0	4.5	2018-01-22	22 January 2018
6	7	vishal	29	NaN	5.0	2015-09-22	22 September 2015
7	8	serah	21	55000.0	3.7	2016-05-01	01 May 2016
8	9	vishal	29	NaN	5.0	2018-06-02	02 June 2018
9	10	Prachi	22	60000.0	4.3	2016-05-01	01 May 2016

before filling

	EMP ID	EMP NAME	Age	Salary	EMP Credits	Joining date	month_name
0	1	satish	21	50000.0	3.8	2017-01-11	11 January 2017
1	2	reeya	23	75000.0	4.0	NaT	NaN
2	3	jay	40	100000.0	5.0	2015-09-22	22 September 2015
3	4	rahul	35	NaN	4.2	2016-11-10	10 November 2016
4	5	roy	26	45000.0	3.9	2017-08-01	01 August 2017
5	6	jay	28	100000.0	4.5	2018-01-22	22 January 2018
6	7	vishal	29	NaN	5.0	2015-09-22	22 September 2015
7	8	serah	21	55000.0	3.7	2016-05-01	01 May 2016
8	9	vishal	29	NaN	5.0	2018-06-02	02 June 2018
9	10	Prachi	22	60000.0	4.3	2016-05-01	01 May 2016

after filling

before duplicate

	EMP ID	EMP NAME	Age	Salary	EMP Credits	Joining date	month_name
0	1	satish	21	50000.0	3.8	2017-01-11	11 January 2017
1	2	reeya	23	75000.0	4.0	2018-01-01	NaN
2	3	jay	40	100000.0	5.0	2015-09-22	22 September 2015
3	4	rahul	35	50000.0	4.2	2016-11-10	10 November 2016
4	5	roy	26	45000.0	3.9	2017-08-01	01 August 2017
5	6	jay	28	100000.0	4.5	2018-01-22	22 January 2018
6	7	vishal	29	50000.0	5.0	2015-09-22	22 September 2015
7	8	serah	21	55000.0	3.7	2016-05-01	01 May 2016
8	9	vishal	29	50000.0	5.0	2018-06-02	02 June 2018
9	10	Prachi	22	60000.0	4.3	2016-05-01	01 May 2016

```
0    False
1    False
2    False
3    False
4    False
5    False
6    False
7    False
8    False
9    False
```

```
dtype: bool
   EMP ID EMP NAME  Age    Salary  EMP Credits Joining date         month_name
0       1   satish   21   50000.0          3.8   2017-01-11    11 January 2017
1       2    reeya   23   75000.0          4.0   2018-01-01                NaN
2       3      jay   40  100000.0          5.0   2015-09-22  22 September 2015
3       4    rahul   35   50000.0          4.2   2016-11-10   10 November 2016
4       5      roy   26   45000.0          3.9   2017-08-01     01 August 2017
5       6      jay   28  100000.0          4.5   2018-01-22    22 January 2018
6       7   vishal   29   50000.0          5.0   2015-09-22  22 September 2015
7       8    serah   21   55000.0          3.7   2016-05-01        01 May 2016
8       9   vishal   29   50000.0          5.0   2018-06-02       02 June 2018
9      10   Prachi   22   60000.0          4.3   2016-05-01        01 May 2016
.....................................................................
.....

the max credit is 5.0

names of max credit is ['jay', 'vishal', 'vishal']
```

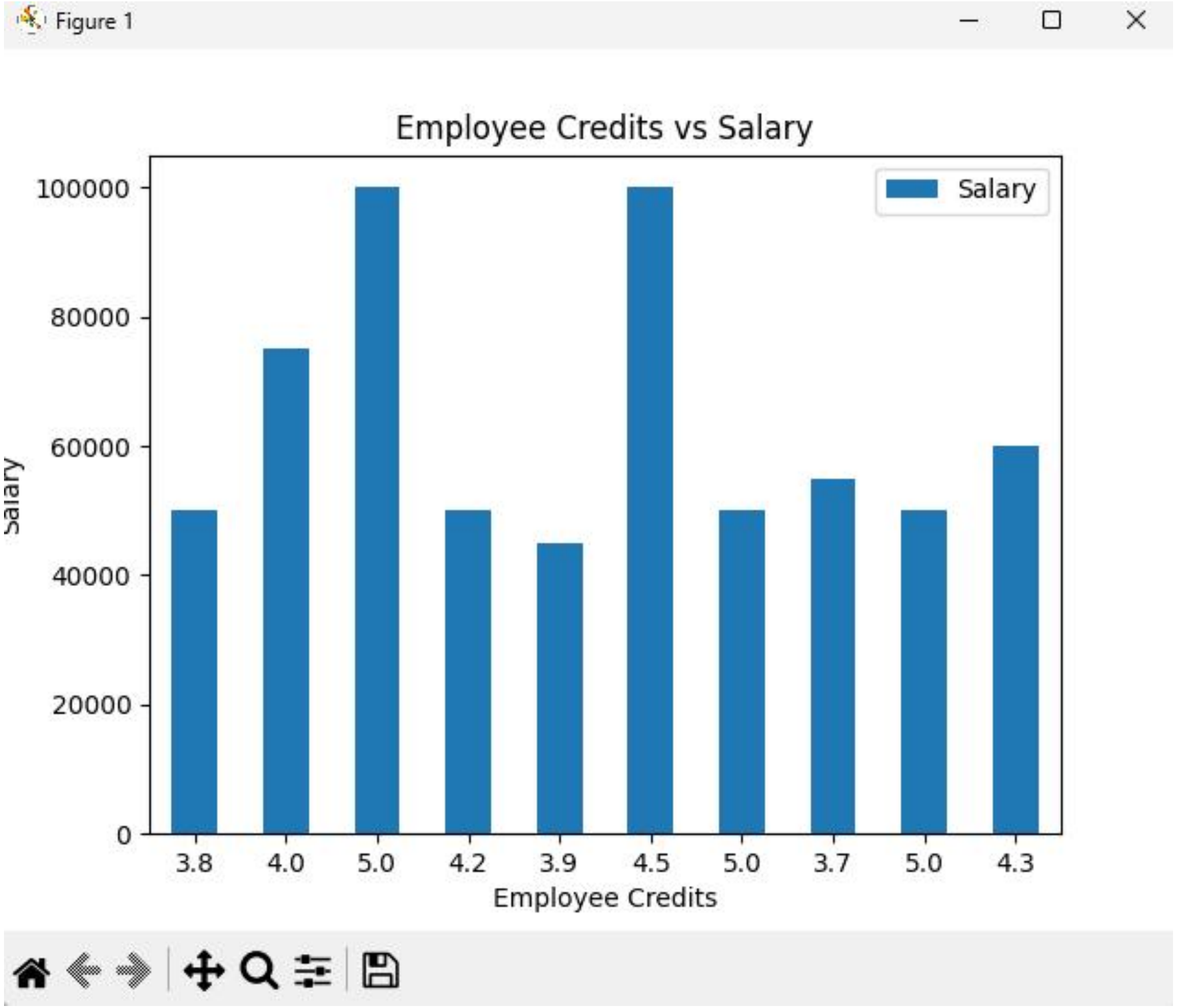
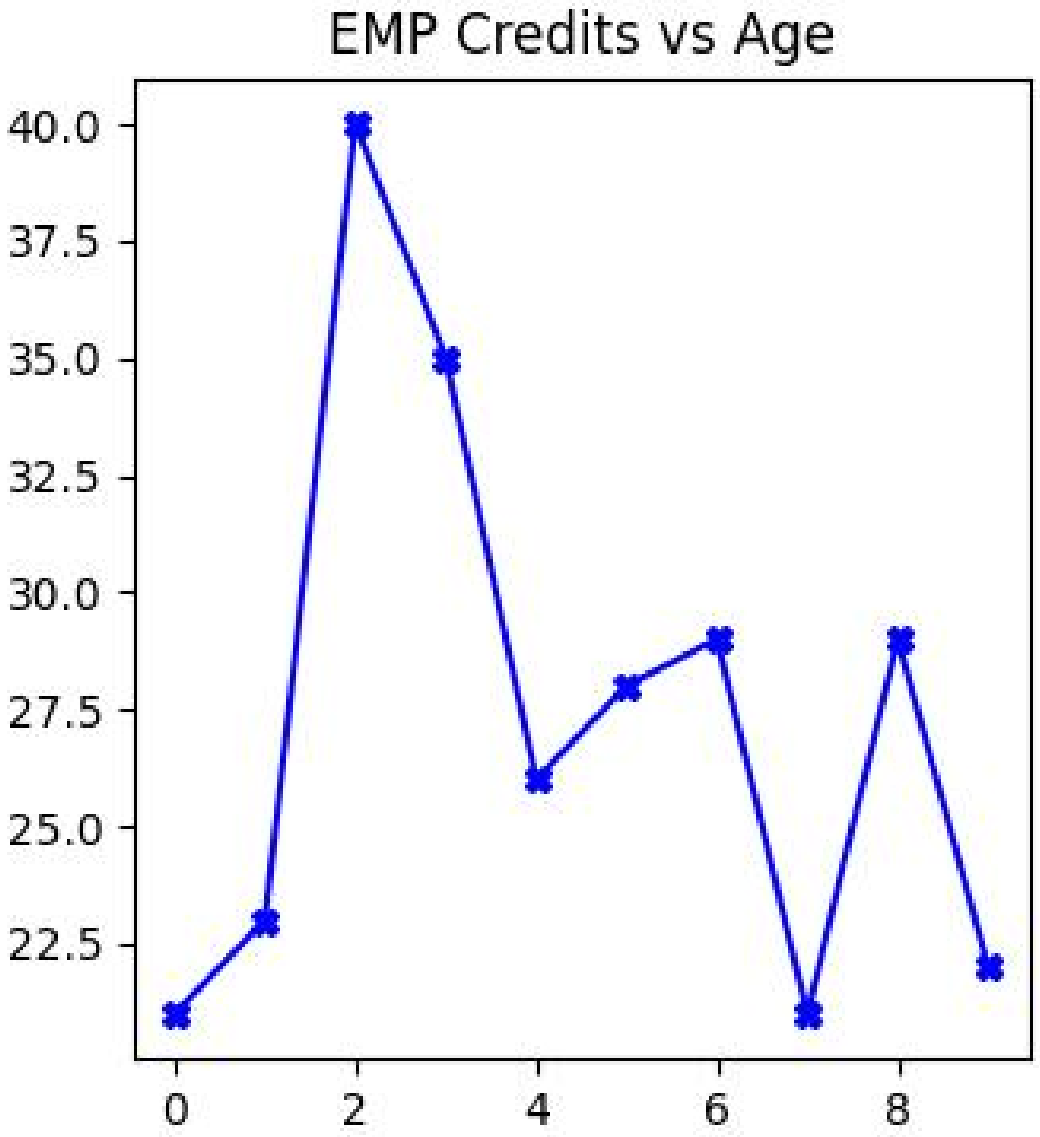
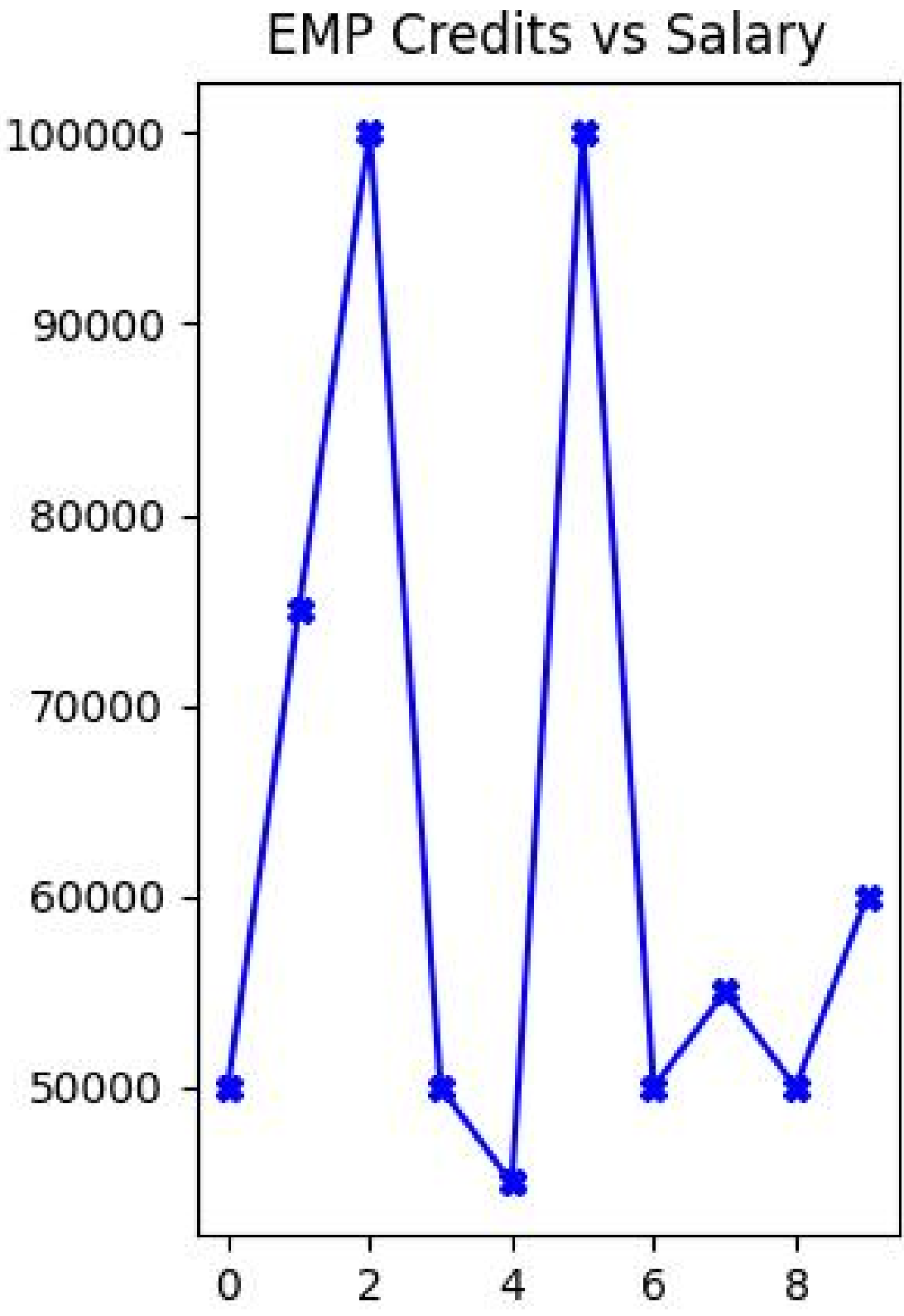


Figure 1
Employee Credits vs Salary
Salary
100000
80000
60000
40000
20000
0
3.8
4.0
5.0
4.2
3.9
4.5
5.0
3.7
5.0
4.3
Employee Credits


Figure 1
Employee Credits
EMP Credits
0
1
2
3
4
5
6
7
8
9

Figure 1




EMP Credits vs Salary
100000
90000
80000
70000
60000
50000
0
2
4
6
8

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
#2..
import pandas as pd
#Loading the Datase
CALORIES_DATA = pd.read_csv(r"C:\Users\Harsh\Downloads\CaloriesDataSet.csv")
print(CALORIES_DATA)
print(".....................................................................................")

print('before filling\n',CALORIES_DATA)
print()
CALORIES_DATA["Calories"].fillna(300,inplace=True)
CALORIES_DATA["Date"].fillna("'2018/12/12'",inplace=True)
print('after filling\n',CALORIES_DATA)
print(".....................................................................................")

print('before duplicate\n',CALORIES_DATA)
print()
print(CALORIES_DATA.duplicated())
print(CALORIES_DATA.drop_duplicates())
print(".....................................................................................")


# calculate the average values for Duration, Pulse, Maxpulse, and Calories
avg_duration = CALORIES_DATA['Duration'].mean()
avg_pulse = CALORIES_DATA['Pulse'].mean()
avg_maxpulse = CALORIES_DATA['Maxpulse'].mean()
avg_calories = CALORIES_DATA['Calories'].mean()

# create a new row with the average values
avg_row = {'Duration': avg_duration, 'Pulse': avg_pulse, 'Maxpulse': avg_maxpulse, 'Calories': avg_calories}

# append the new row to the dataframe
CALORIES_DATA = CALORIES_DATA.append(avg_row, ignore_index=True)
```

Ln: 1 Col: 63

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
avg_pulse = CALORIES_DATA['Pulse'].mean()
avg_maxpulse = CALORIES_DATA['Maxpulse'].mean()
avg_calories = CALORIES_DATA['Calories'].mean()

# create a new row with the average values
avg_row = {'Duration': avg_duration, 'Pulse': avg_pulse, 'Maxpulse': avg_maxpulse, 'Calories': avg_calories}

# append the new row to the dataframe
CALORIES_DATA = CALORIES_DATA.append(avg_row, ignore_index=True)

# write the updated dataframe to a new csv file
CALORIES_DATA.to_csv('updated_data.csv', index=False)
print(CALORIES_DATA)


print(".....................................................................................")
CALORIES_DATA.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('CALORIES')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.area(color='b',label='CALORIES',figsize=(5,5))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.bar(color='c',label='CALORIES',figsize=(3,3))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()
```

Ln: 1 Col: 63

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
print("..........................................................................")
CALORIES_DATA.plot(marker='X', ms=15,mec='r',mfc='r',color='r',linestyle='--',label='Makers',figsize=(5,5))
plt.xlabel('CALORIES')
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.area(color='b',label='CALORIES',figsize=(5,5))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.bar(color='c',label='CALORIES',figsize=(3,3))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.barh(color='c',label='CALORIES',figsize=(3,3))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.box(color='m',label='CALORIES',figsize=(3,3))
plt.ylabel('Date')
plt.show()

CALORIES_DATA.plot.hist(bins=6,figsize=(3,3))
plt.ylabel('Date')
plt.show()

CALORIES_DATA.plot.pie(label='CALORIES',autopct='%1.1f%%')
```

Ln: 1 Col: 63

GE ASSIGNMET 5(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 5(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
plt.ylabel('Salary')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.area(color='b',label='CALORIES',figsize=(5,5))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.bar(color='c',label='CALORIES',figsize=(3,3))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.barh(color='c',label='CALORIES',figsize=(3,3))
plt.xlabel('CALORIES')
plt.ylabel('Date')
plt.legend(loc='lower center')
plt.show()

CALORIES_DATA.plot.box(color='m',label='CALORIES',figsize=(3,3))
plt.ylabel('Date')
plt.show()

CALORIES_DATA.plot.hist(bins=6,figsize=(3,3))
plt.ylabel('Date')
plt.show()

CALORIES_DATA.plot.pie(label='CALORIES',autopct='%1.1f%%')
plt.ylabel('Date')
plt.show()
```

Ln: 1 Col: 63

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	300.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	'2018/12/12'	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	300.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

.....

Ln: 287 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	300.0
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	'2018/12/12'	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	300.0
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

.....

Ln: 287 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
31          60  '2020/12/31'     92       115     243.0
..........................................................................
before filling
     Duration          Date  Pulse  Maxpulse  Calories
0          60  '2020/12/01'    110       130     409.1
1          60  '2020/12/02'    117       145     479.0
2          60  '2020/12/03'    103       135     340.0
3          45  '2020/12/04'    109       175     282.4
4          45  '2020/12/05'    117       148     406.0
5          60  '2020/12/06'    102       127     300.0
6          60  '2020/12/07'    110       136     374.0
7         450  '2020/12/08'    104       134     253.3
8          30  '2020/12/09'    109       133     195.1
9          60  '2020/12/10'     98       124     269.0
10         60  '2020/12/11'    103       147     329.3
11         60  '2020/12/12'    100       120     250.7
12         60  '2020/12/12'    100       120     250.7
13         60  '2020/12/13'    106       128     345.3
14         60  '2020/12/14'    104       132     379.3
15         60  '2020/12/15'     98       123     275.0
16         60  '2020/12/16'     98       120     215.2
17         60  '2020/12/17'    100       120     300.0
18         45  '2020/12/18'     90       112       NaN
19         60  '2020/12/19'    103       123     323.0
20         45  '2020/12/20'     97       125     243.0
21         60  '2020/12/21'    108       131     364.2
22         45           NaN    100       119     282.0
23         60  '2020/12/23'    130       101     300.0
24         45  '2020/12/24'    105       132     246.0
25         60  '2020/12/25'    102       126     334.5
26         60      20201226    100       120     250.0
27         60  '2020/12/27'     92       118     241.0
28         60  '2020/12/28'    103       132       NaN
29         60  '2020/12/29'    100       132     280.0
```

Ln: 347 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
31          60  '2020/12/31'     92       115     243.0

after filling
     Duration          Date  Pulse  Maxpulse  Calories
0          60  '2020/12/01'    110       130     409.1
1          60  '2020/12/02'    117       145     479.0
2          60  '2020/12/03'    103       135     340.0
3          45  '2020/12/04'    109       175     282.4
4          45  '2020/12/05'    117       148     406.0
5          60  '2020/12/06'    102       127     300.0
6          60  '2020/12/07'    110       136     374.0
7         450  '2020/12/08'    104       134     253.3
8          30  '2020/12/09'    109       133     195.1
9          60  '2020/12/10'     98       124     269.0
10         60  '2020/12/11'    103       147     329.3
11         60  '2020/12/12'    100       120     250.7
12         60  '2020/12/12'    100       120     250.7
13         60  '2020/12/13'    106       128     345.3
14         60  '2020/12/14'    104       132     379.3
15         60  '2020/12/15'     98       123     275.0
16         60  '2020/12/16'     98       120     215.2
17         60  '2020/12/17'    100       120     300.0
18         45  '2020/12/18'     90       112     300.0
19         60  '2020/12/19'    103       123     323.0
20         45  '2020/12/20'     97       125     243.0
21         60  '2020/12/21'    108       131     364.2
22         45  '2018/12/12'    100       119     282.0
23         60  '2020/12/23'    130       101     300.0
24         45  '2020/12/24'    105       132     246.0
25         60  '2020/12/25'    102       126     334.5
26         60      20201226    100       120     250.0
27         60  '2020/12/27'     92       118     241.0
28         60  '2020/12/28'    103       132     300.0
29         60  '2020/12/29'    100       132     280.0
```

Ln: 347 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
31          60  '2020/12/31'     92       115     243.0
..........................................................................
before duplicate
     Duration          Date  Pulse  Maxpulse  Calories
0          60  '2020/12/01'    110       130     409.1
1          60  '2020/12/02'    117       145     479.0
2          60  '2020/12/03'    103       135     340.0
3          45  '2020/12/04'    109       175     282.4
4          45  '2020/12/05'    117       148     406.0
5          60  '2020/12/06'    102       127     300.0
6          60  '2020/12/07'    110       136     374.0
7         450  '2020/12/08'    104       134     253.3
8          30  '2020/12/09'    109       133     195.1
9          60  '2020/12/10'     98       124     269.0
10         60  '2020/12/11'    103       147     329.3
11         60  '2020/12/12'    100       120     250.7
12         60  '2020/12/12'    100       120     250.7
13         60  '2020/12/13'    106       128     345.3
14         60  '2020/12/14'    104       132     379.3
15         60  '2020/12/15'     98       123     275.0
16         60  '2020/12/16'     98       120     215.2
17         60  '2020/12/17'    100       120     300.0
18         45  '2020/12/18'     90       112     300.0
19         60  '2020/12/19'    103       123     323.0
20         45  '2020/12/20'     97       125     243.0
21         60  '2020/12/21'    108       131     364.2
22         45  '2018/12/12'    100       119     282.0
23         60  '2020/12/23'    130       101     300.0
24         45  '2020/12/24'    105       132     246.0
25         60  '2020/12/25'    102       126     334.5
26         60      20201226    100       120     250.0
27         60  '2020/12/27'     92       118     241.0
28         60  '2020/12/28'    103       132     300.0
29         60  '2020/12/29'    100       132     280.0
```

Ln: 347 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
31          60  '2020/12/31'     92        115     243.0

0      False
1      False
2      False
3      False
4      False
5      False
6      False
7      False
8      False
9      False
10     False
11     False
12      True
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22     False
23     False
24     False
25     False
26     False
27     False
28     False
29     False
30     False
31     False
```

Ln: 347 Col: 0

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5

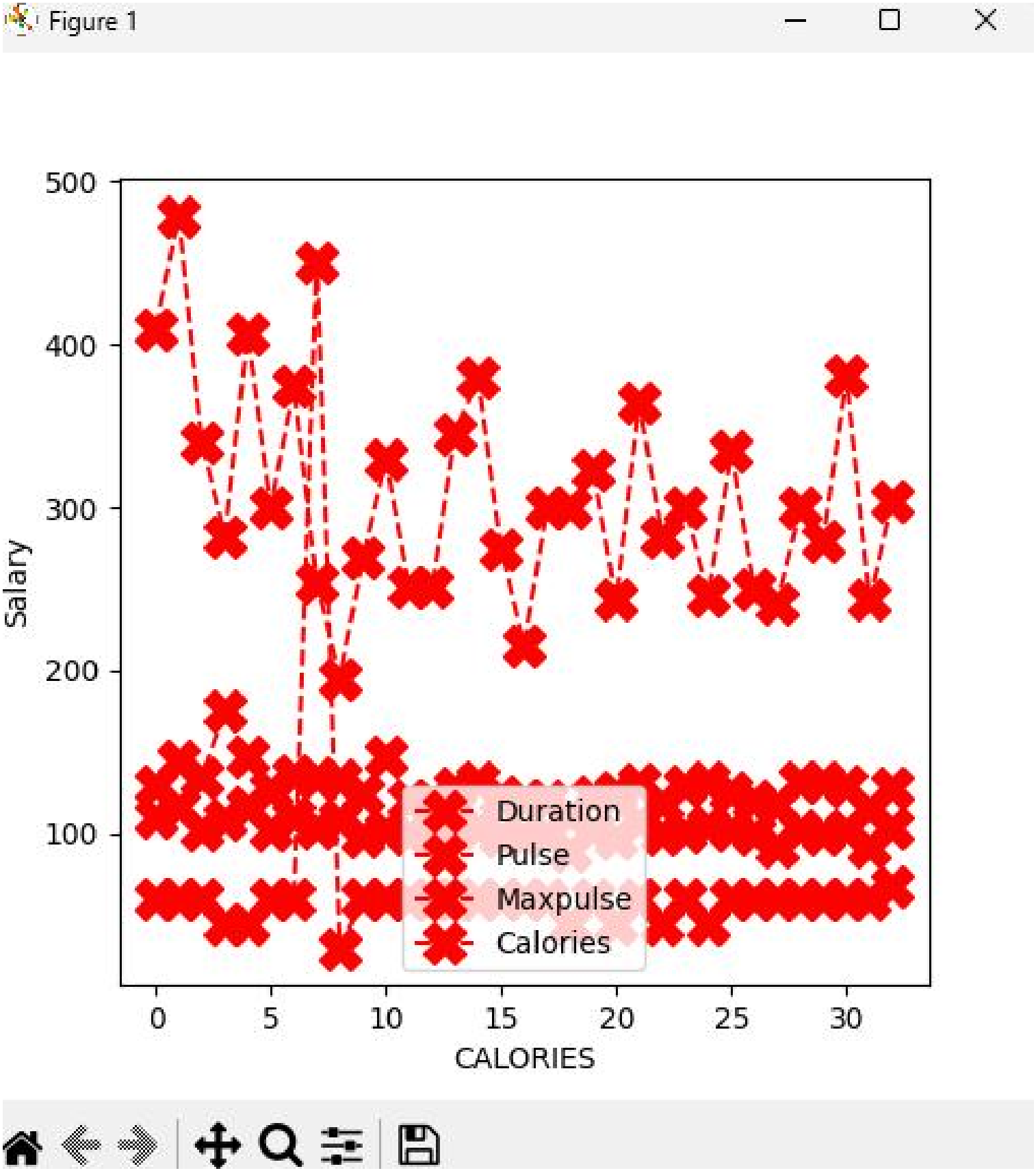


Figure 1
500
400
300
200
100
Salary
0
5
10
15
20
25
30
CALORIES
Duration
Pulse
Maxpulse
Calories

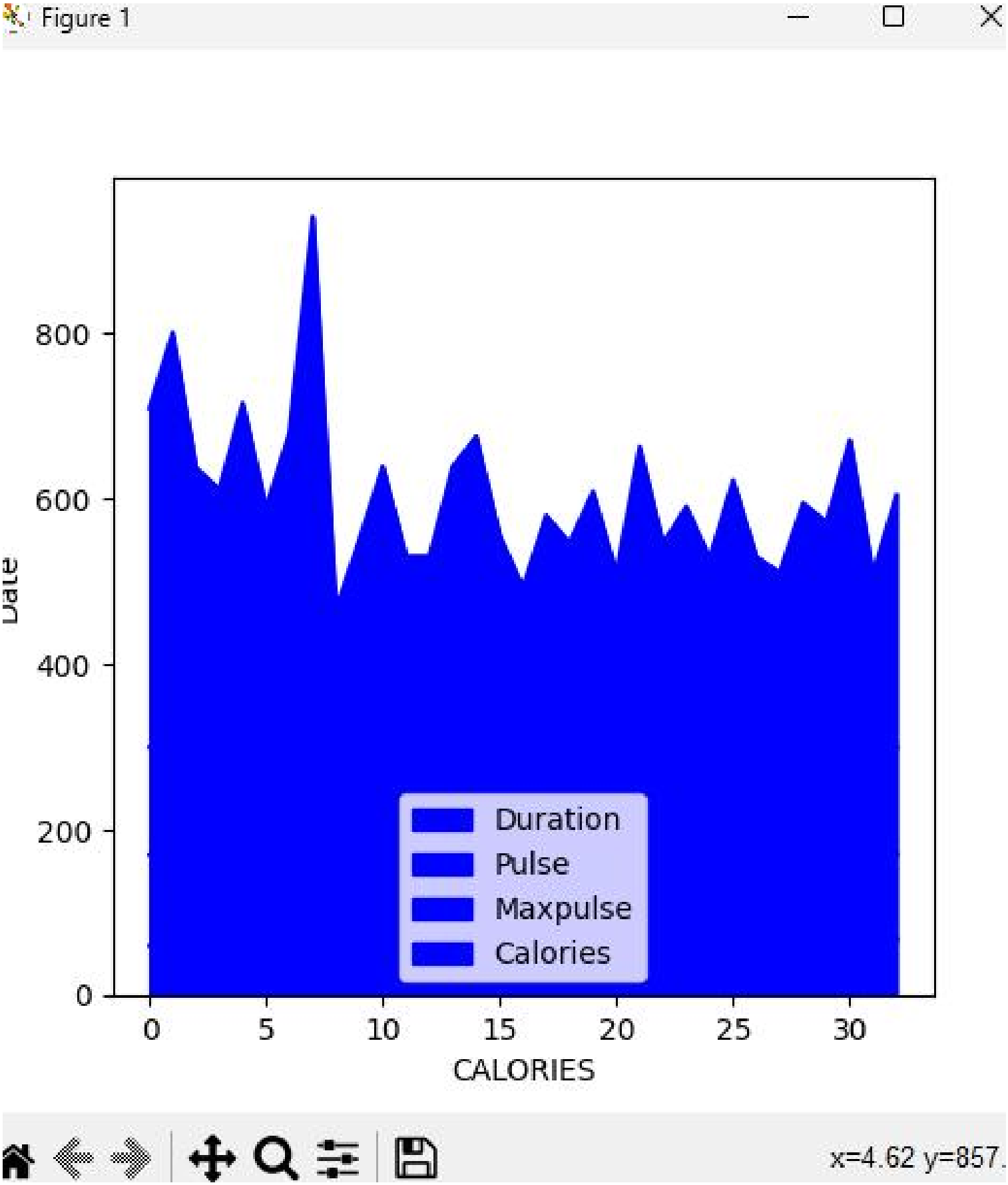


Figure 1
800
600
400
200
0
0
5
10
15
20
25
30
CALORIES
Duration
Pulse
Maxpulse
Calories
x=4.62 y=857.

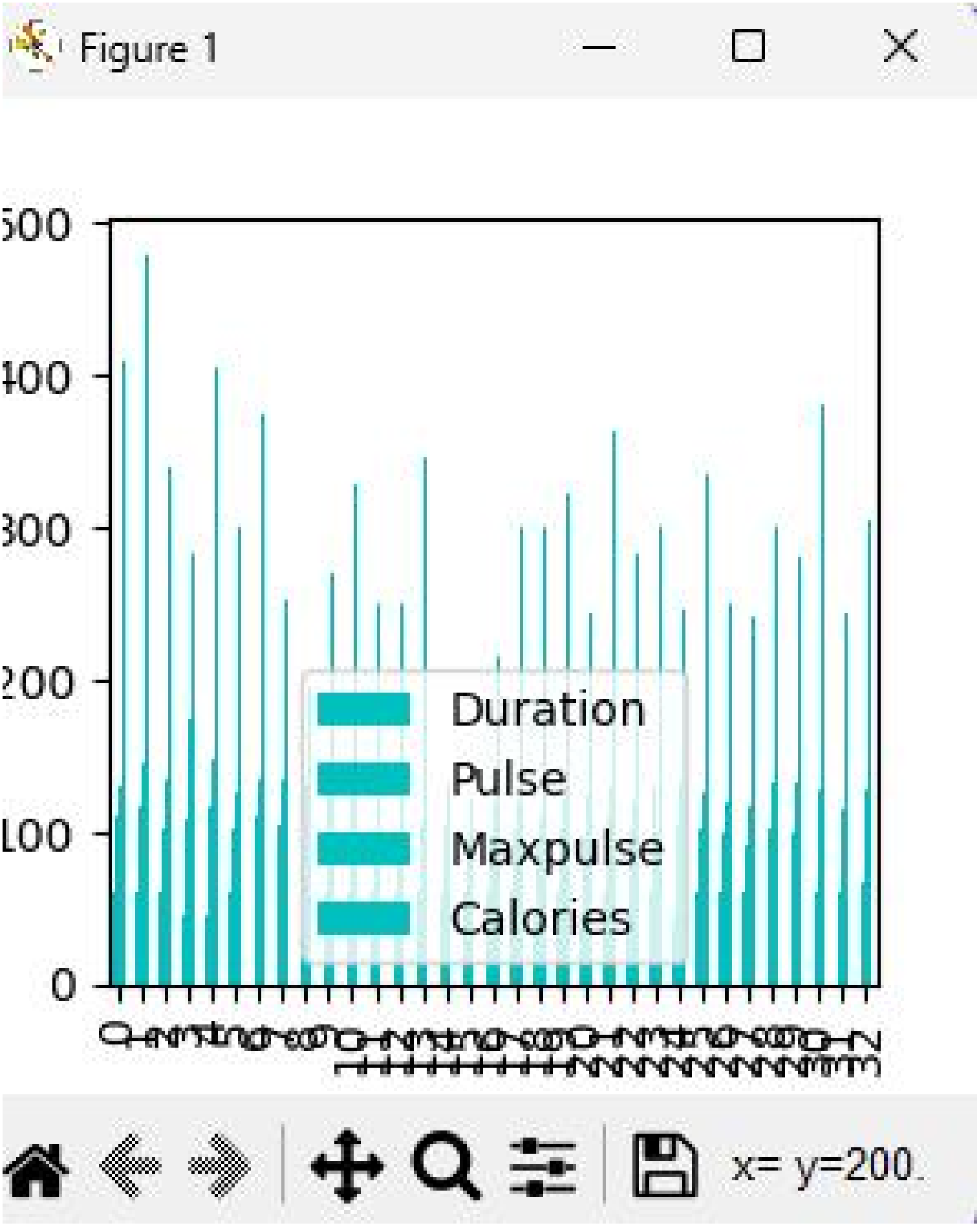


Figure 1
500
400
300
200
100
0
Duration
Pulse
Maxpulse
Calories
x= y=200.

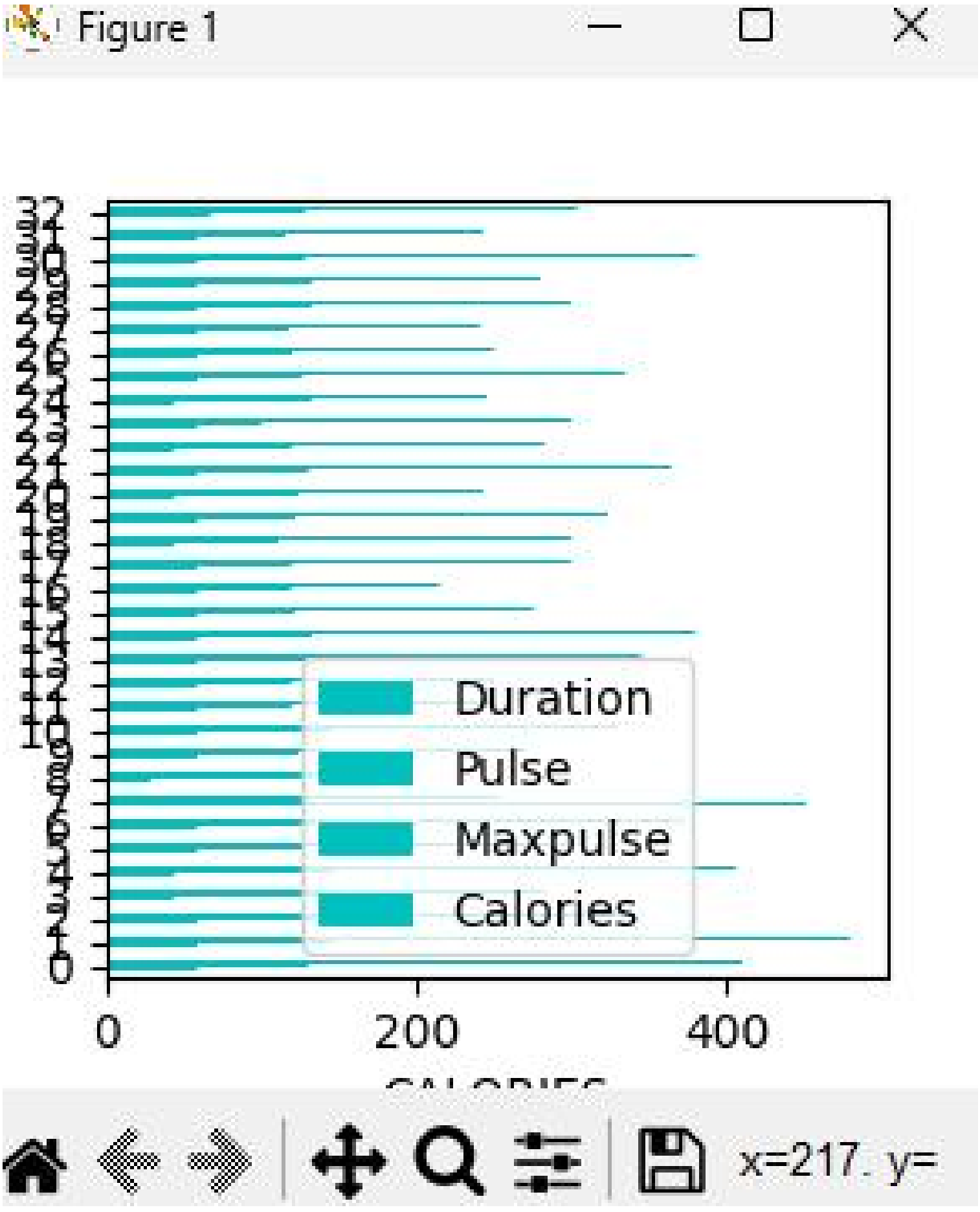


Figure 1
Duration
Pulse
Maxpulse
Calories
0
200
400
x=217. y=

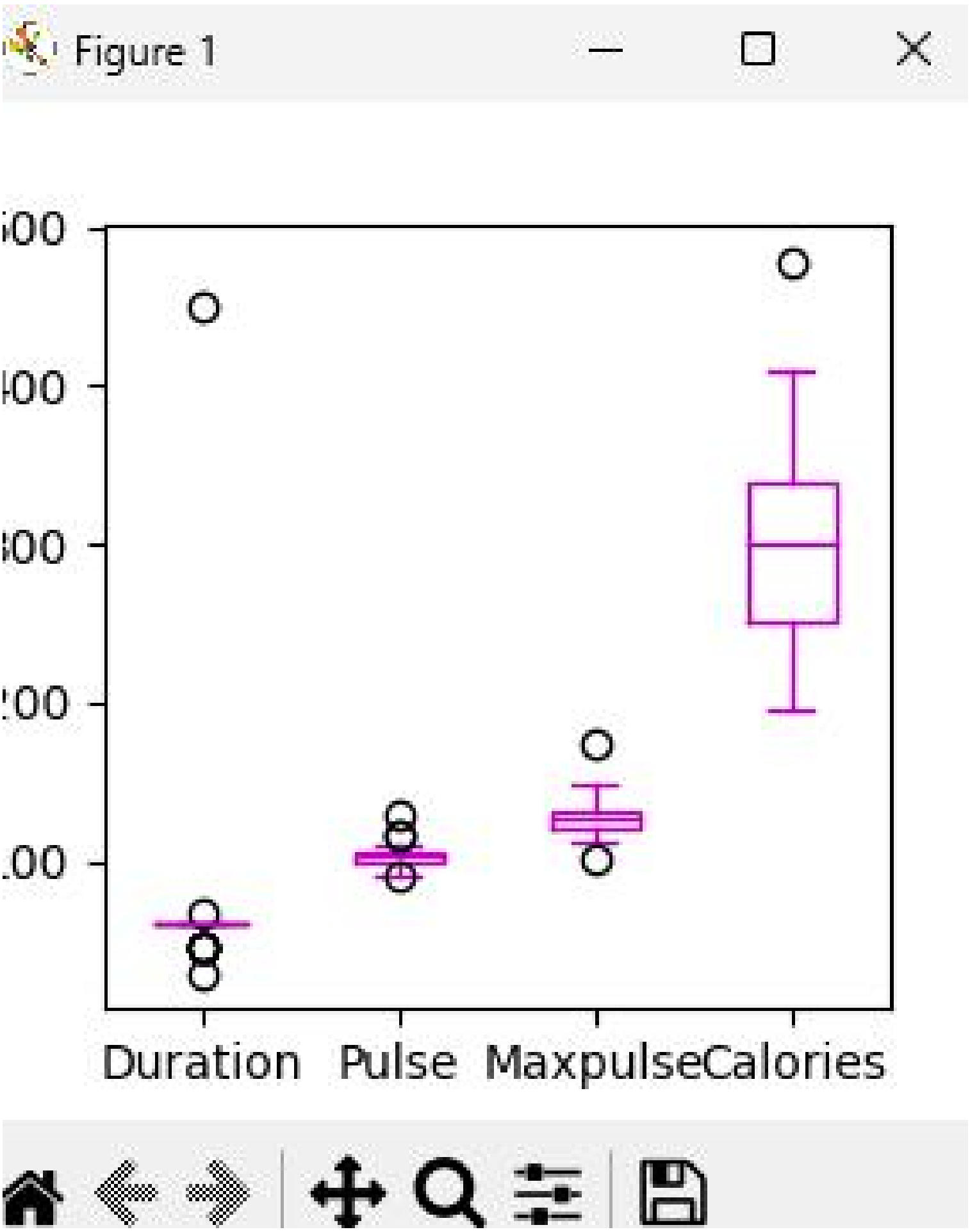


Figure 1
Duration
Pulse
Maxpulse
Calories

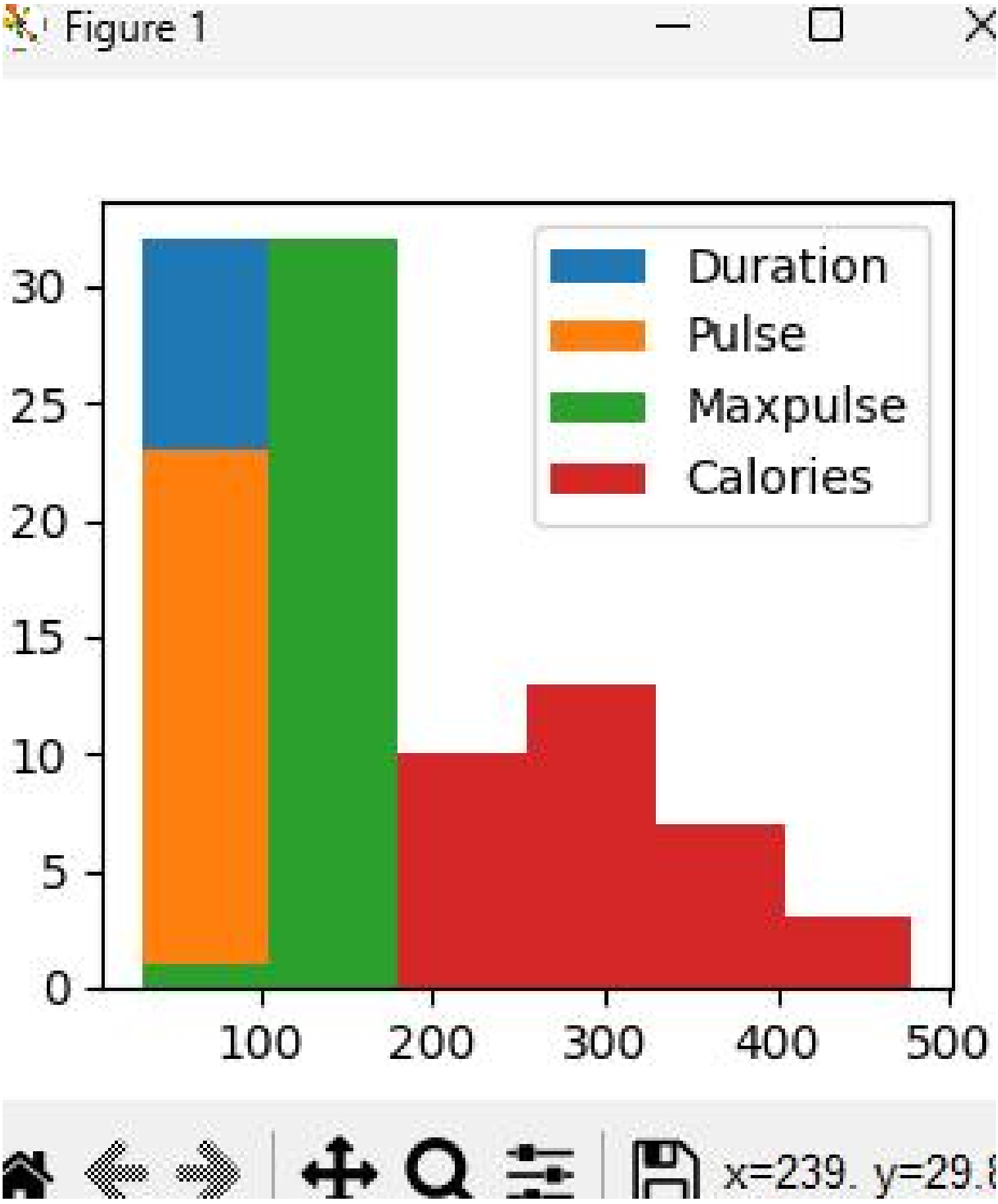


Figure 1
Duration
Pulse
Maxpulse
Calories
30
25
20
15
10
5
0
100
200
300
400
500
x=239. y=29.

```python
#GE ASSIGNMENT 6

import pandas as pd
import matplotlib.pyplot as plt

data=pd.read_csv(r"C:\Users\Harsh\Downloads\IMDB2.csv")
print(data)

# group the data frame by year
grouped = data.groupby("Year of Release")
print('grouped=',grouped)


# calculate total number of movies and average rating per year
agg_dict = {"Movie Name": "count", "Rating": "mean"}
agg_df = grouped.agg(agg_dict)
print(agg_df)


# plot number of movies released vs. year using line and bar plots
fig, ax = plt.subplots(2, 1, figsize=(10, 10))
agg_df["Movie Name"].plot(kind="line", ax=ax[0])
agg_df["Movie Name"].plot(kind="bar", ax=ax[1])
ax[0].set_xlabel("Year")
ax[0].set_ylabel("Number of Movies Released")
ax[1].set_xlabel("Year")
ax[1].set_ylabel("Number of Movies Released")
plt.show()
```

	S.No.	Movie Name	...	Rating	Duration(in hrs)
0	1	The Nightmare Before Christmas	...	3.9	1.268889
1	2	The Mummy	...	3.5	1.218889
2	3	Orphans of the Storm	...	3.2	2.517222
3	4	The Object of Beauty	...	2.8	1.708333
4	5	Night Tide	...	2.8	1.423889
..	...	...	...	...	...
345	346	I'll Be Home for Christmas	...	3.7	1.435000
346	347	Guinevere	...	2.9	1.749722
347	348	Babar King of the Elephants	...	3.2	1.321111
348	349	The Boxer	...	3.6	1.814167
349	350	From Dusk Till Dawn 2: Texas Blood Money	...	3.0	1.472222

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
grouped= <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001D1AC0B1850>
                 Movie Name    Rating
Year of Release
1915                      1  2.900000
1919                      1  3.300000
1920                      3  3.100000
1921                      1  3.200000
1923                      2  3.450000
1924                      1  3.300000
1925                      3  3.333333
1927                      1  3.100000
1928                      1  3.700000
1929                      1  3.500000
1932                      1  3.500000
1935                      1  3.700000
1936                      1  3.100000
1937                      2  3.300000
1938                      1  3.700000
1939                      1  3.100000
1947                      1  3.700000
1950                      1  3.800000
1952                      1  3.900000
1953                      1  3.200000
1956                      1  3.500000
1959                      1  3.600000
1963                      1  2.800000
1964                      2  3.600000
1968                      3  3.033333
1969                      2  3.600000
1970                      2  3.750000
1971                      8  3.387500
1972                      6  3.233333
1973                      4  3.050000
1974                      2  2.450000
```

Ln: 78 Col: 0

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
1964                    2  3.600000
1968                    3  3.033333
1969                    2  3.600000
1970                    2  3.750000
1971                    8  3.387500
1972                    6  3.233333
1973                    4  3.050000
1974                    2  2.450000
1975                    4  3.425000
1976                    3  3.133333
1977                    1  2.700000
1978                    4  3.425000
1979                    4  3.075000
1980                    3  3.433333
1981                    3  3.633333
1982                    2  3.700000
1983                    1  3.600000
1984                    4  3.500000
1985                    7  3.500000
1986                    9  3.522222
1987                    7  3.342857
1988                    6  3.433333
1989                    8  3.300000
1990                    8  3.525000
1991                   11  3.400000
1992                   11  3.600000
1993                   12  3.483333
1994                   21  3.423810
1995                   30  3.406667
1996                   29  3.386207
1997                   40  3.415000
1998                   52  3.390385
1999                   12  3.333333
>>>
```

Ln: 78 Col: 0

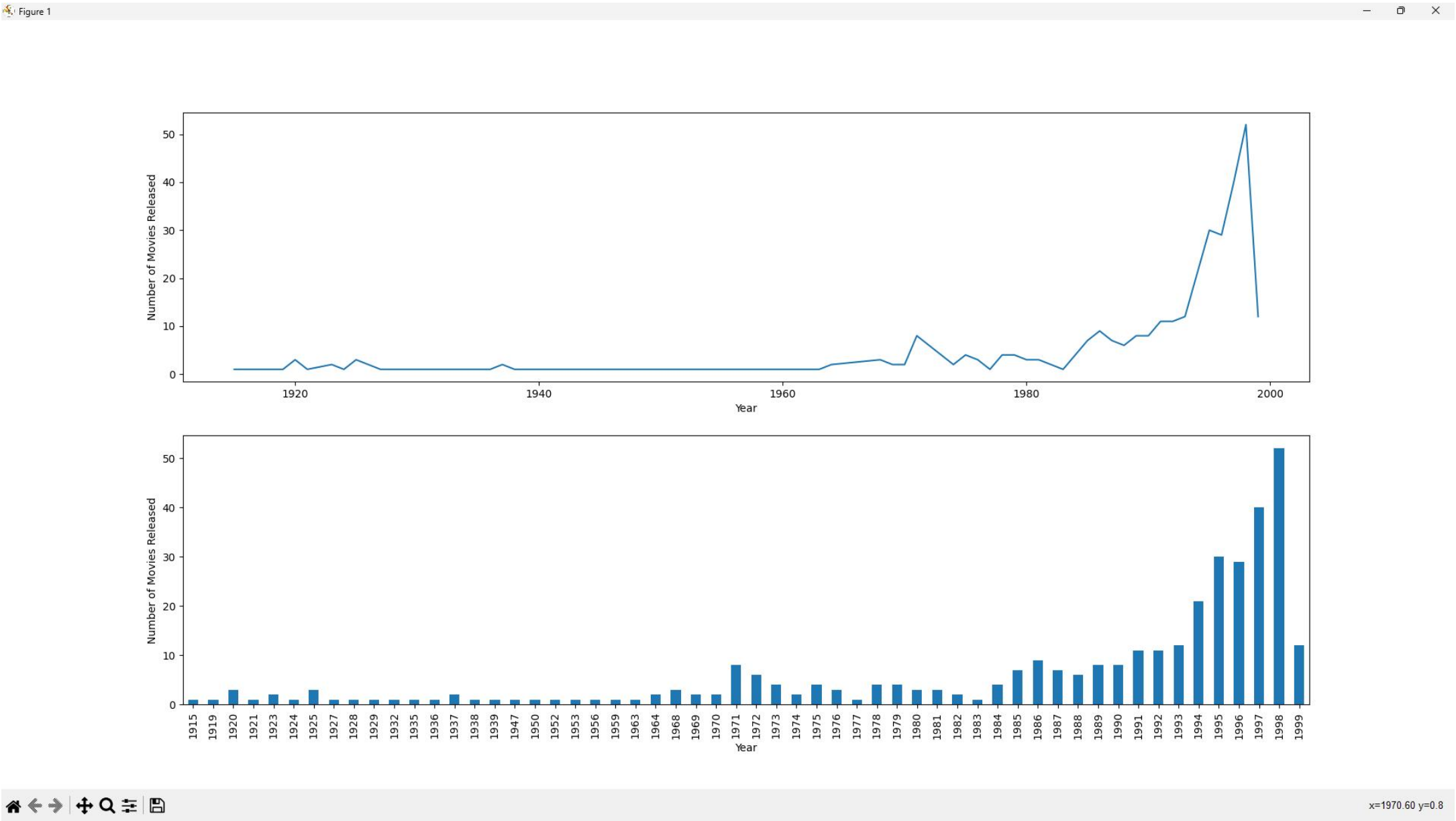


Figure 1
Number of Movies Released
Year
1920
1940
1960
1980
2000
Number of Movies Released
Year
1915 1919 1920 1921 1923 1924 1925 1927 1928 1929 1932 1935 1936 1937 1938 1939 1947 1950 1952 1953 1956 1959 1963 1964 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999
x=1970.60 y=0.8

GE ASSIGNMET 7(SEM2).py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMET 7(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
import seaborn as sns
diamonds=sns.load_dataset('diamonds')
print(diamonds)
import matplotlib.pyplot as plt
sns.lineplot(x='price',y='carat',hue='cut',data=diamonds)
plt.title('Price vs. Carat')
plt.xlabel('Price')
plt.ylabel('Carat')
plt.show()

import matplotlib.pyplot as plt
sns.scatterplot(x='price',y='carat',hue='cut',size='color',data=diamonds)
plt.title('carat vs price')
plt.xlabel('price')
plt.ylabel('carat')
plt.show()

g = sns.FacetGrid(diamonds, row='color', col='cut', margin_titles=True)
g.map(sns.scatterplot, 'carat', 'price')
g.tight_layout()
plt.show()


# Create the bar plot using seaborn
sns.barplot(x='cut', y='price', data=diamonds)
plt.xlabel('Cut')
plt.ylabel('Mean Price')
plt.title('Mean Price of Diamonds by Cut')
plt.show()
```

Ln: 3 Col: 15

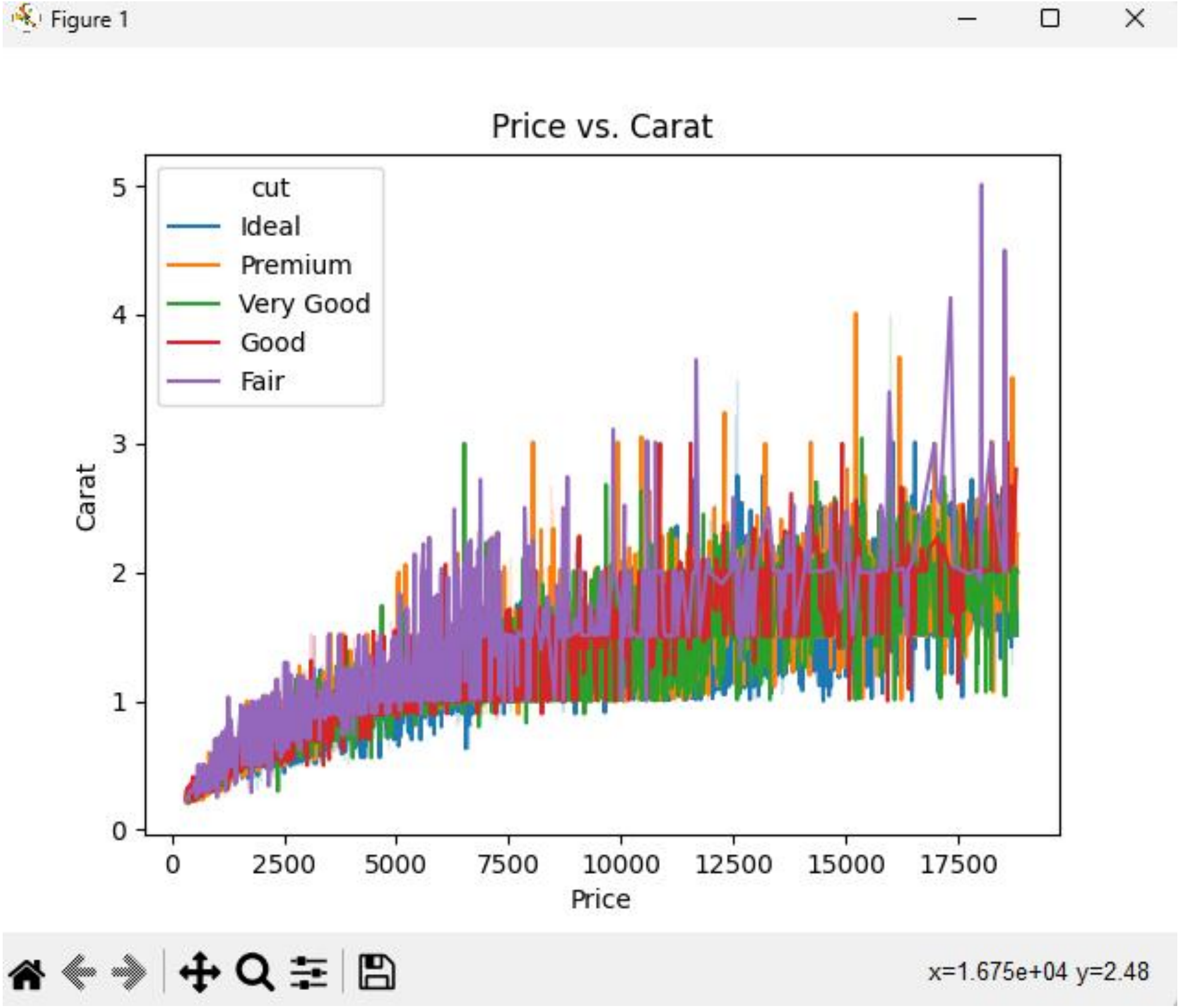


Figure 1
Price vs. Carat
cut
Ideal
Premium
Very Good
Good
Fair
Carat
Price
0
1
2
3
4
5
0
2500
5000
7500
10000
12500
15000
17500
x=1.675e+04 y=2.48

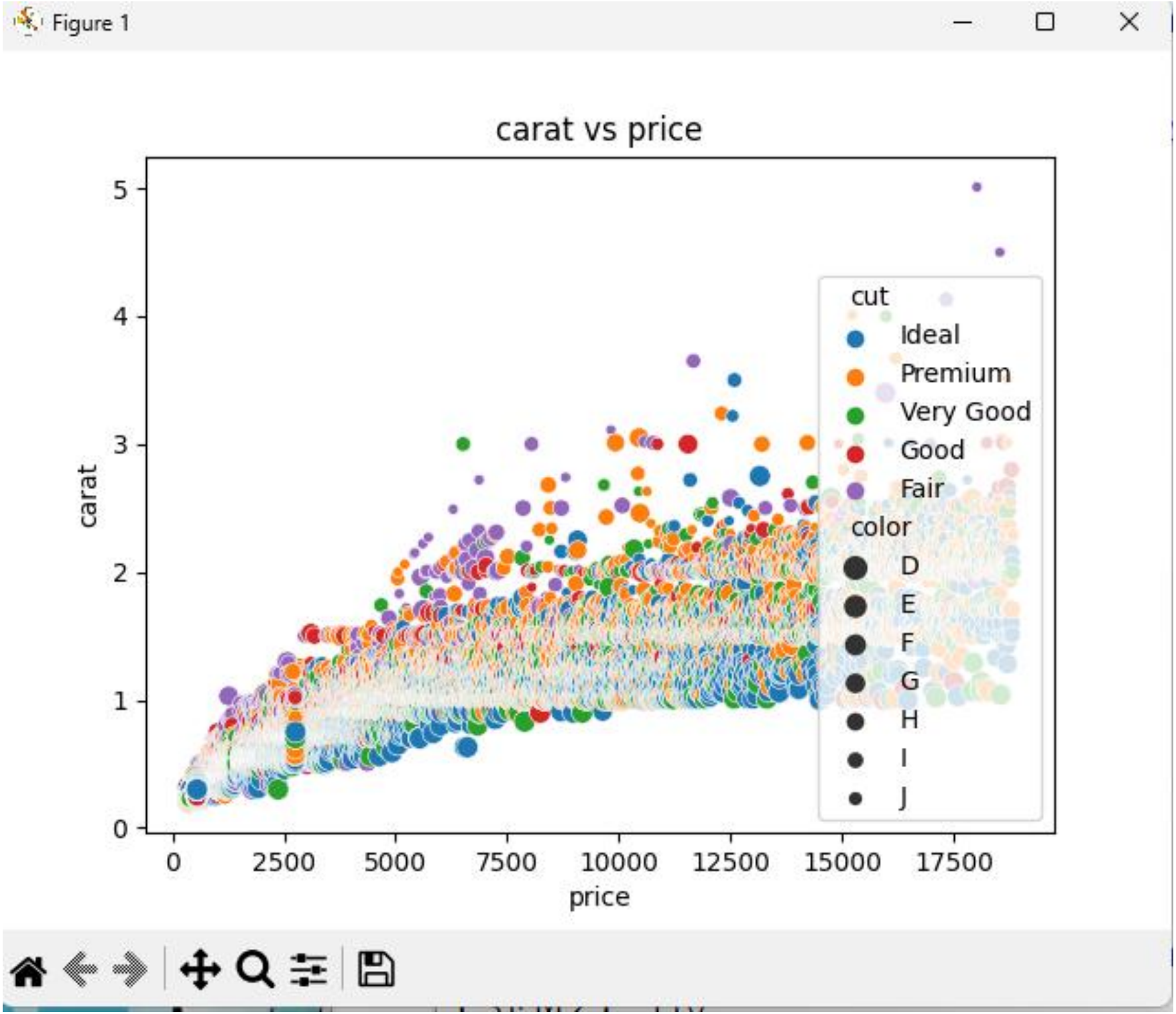


Figure 1
carat vs price
carat
price
0
1
2
3
4
5
0
2500
5000
7500
10000
12500
15000
17500
cut
Ideal
Premium
Very Good
Good
Fair
color
D
E
F
G
H
I
J

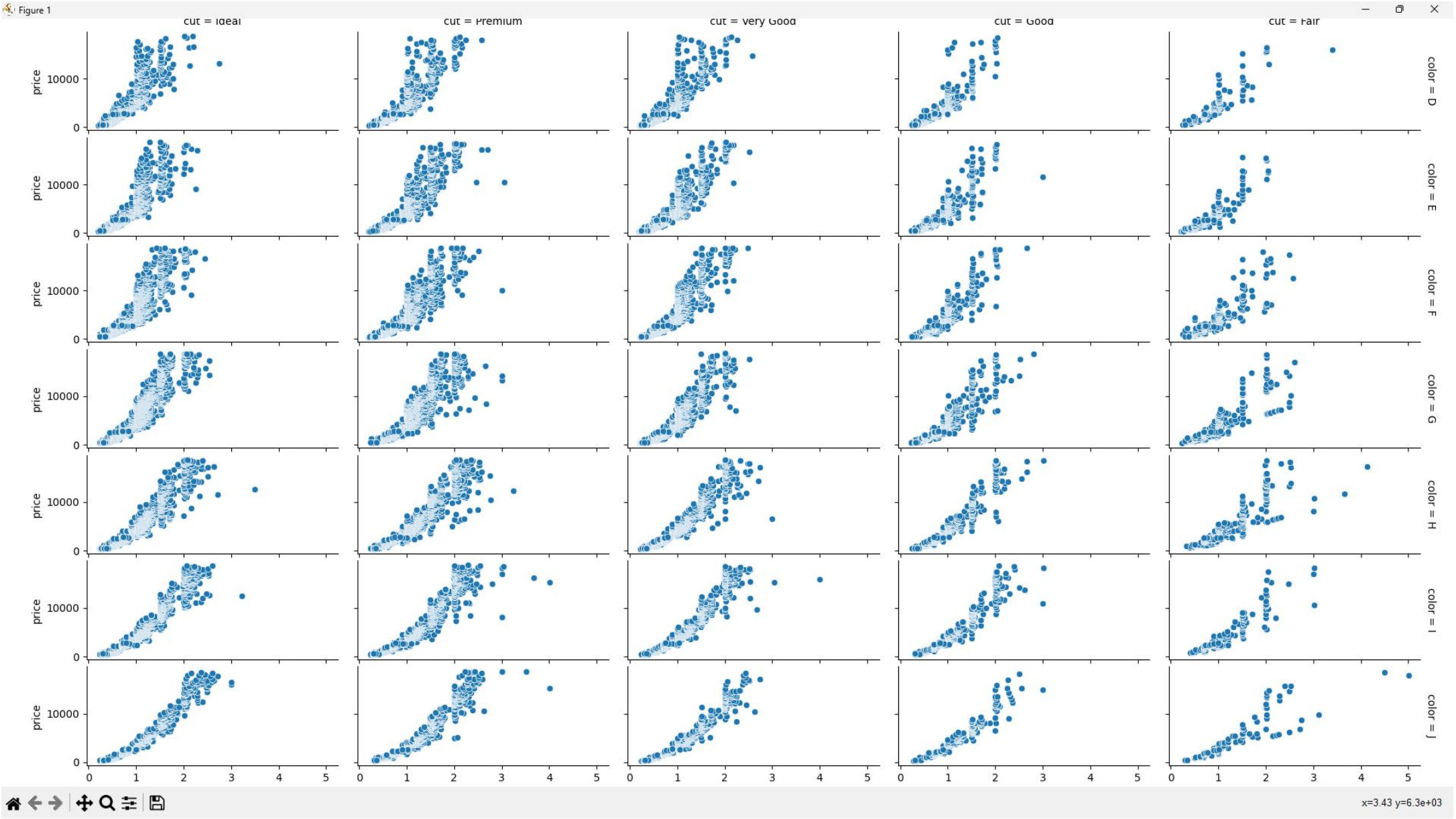
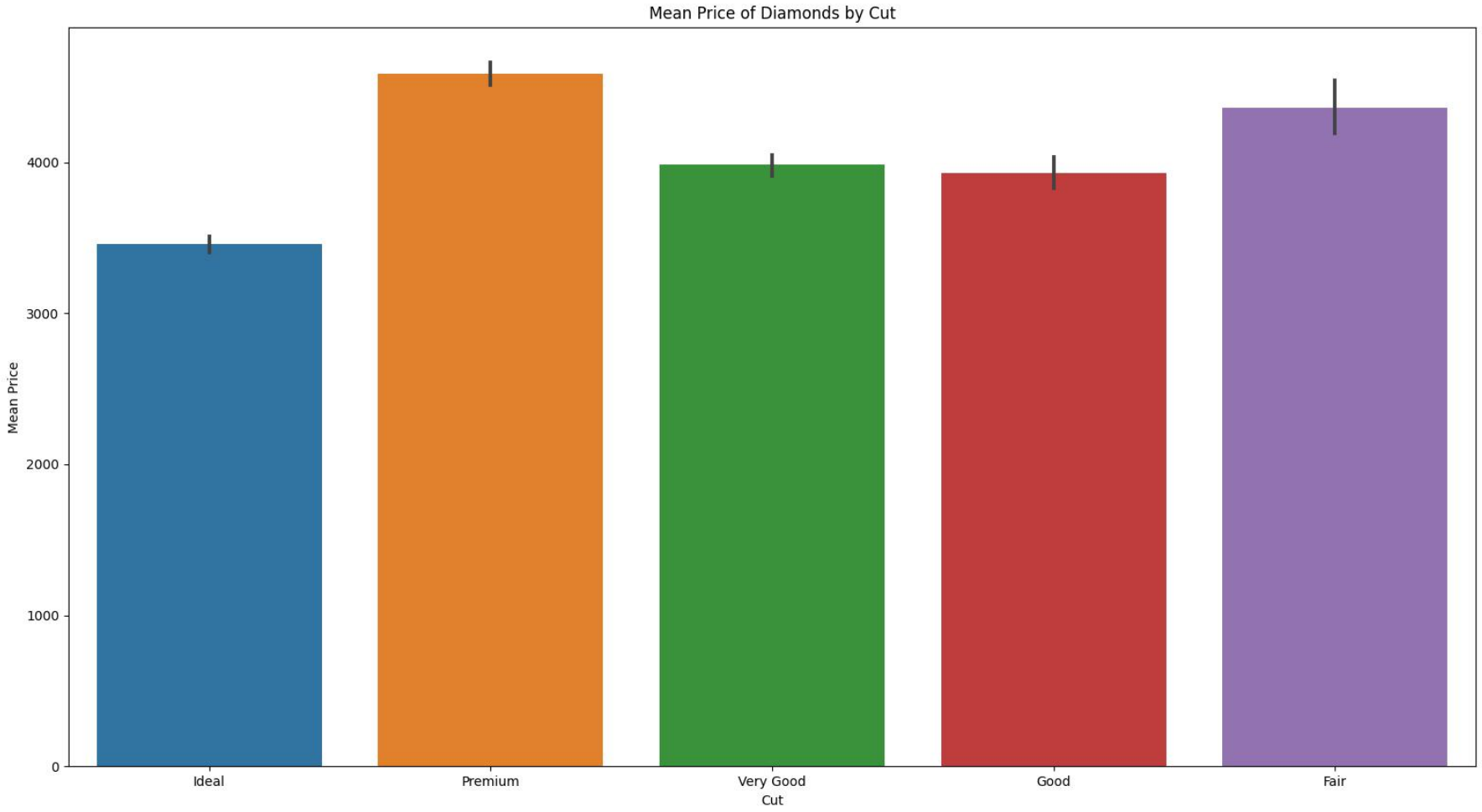


Figure 1
cut = Ideal
cut = Premium
cut = Very Good
cut = Good
cut = Fair
color = D
color = E
color = F
color = G
color = H
color = I
color = J
price
10000
0
0
1
2
3
4
5
x=3.43 y=6.3e+03

Figure 1



x= y=4229.

GE ASSIGNMET 8(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 8(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
import seaborn as sns
tips=sns.load_dataset('tips')
print(tips)

import matplotlib.pyplot as plt
sns.lineplot(x='total_bill',y='tip',hue='sex',style='day',data=tips)
plt.title('Tip vs. Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Tip')
plt.show()

import matplotlib.pyplot as plt
sns.scatterplot(x='total_bill',y='size',hue='day',size='time',data=tips)
plt.title('Size vs Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Party size')
plt.show()

sns.barplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Total_bill')
plt.ylabel('Time')
plt.show()

sns.countplot(x='time',hue='time',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total Bill')
plt.show()

sns.boxplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total bill')
```

Ln: 5 Col: 0

GE ASSIGNMET 8(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 8(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
sns.boxplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.violinplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.swarmplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.stripplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.displot(x='total_bill',hue='time',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.jointplot(x='tip',y='total_bill',data=tips)
plt.title('Tip vs Total Bill')
plt.xlabel('Tip')
plt.ylabel('Total_bill')
```

Ln: 5 Col: 0

GE ASSIGNMET 8(SEM2).py - C:\Users\Harsh\AppData\Local\Programs\Python\Python311\GE ASSIGNMET 8(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.swarmplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.stripplot(x='time',y='total_bill',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.displot(x='total_bill',hue='time',data=tips)
plt.title('Total bill vs time')
plt.xlabel('Time')
plt.ylabel('Total_bill')
plt.show()

sns.jointplot(x='tip',y='total_bill',data=tips)
plt.title('Tip vs Total Bill')
plt.xlabel('Tip')
plt.ylabel('Total_bill')
plt.show()

sns.pairplot(tips)
plt.show()

sns.regplot(x='total_bill',y='tip',data=tips)
plt.show()
```

Ln: 5 Col: 0

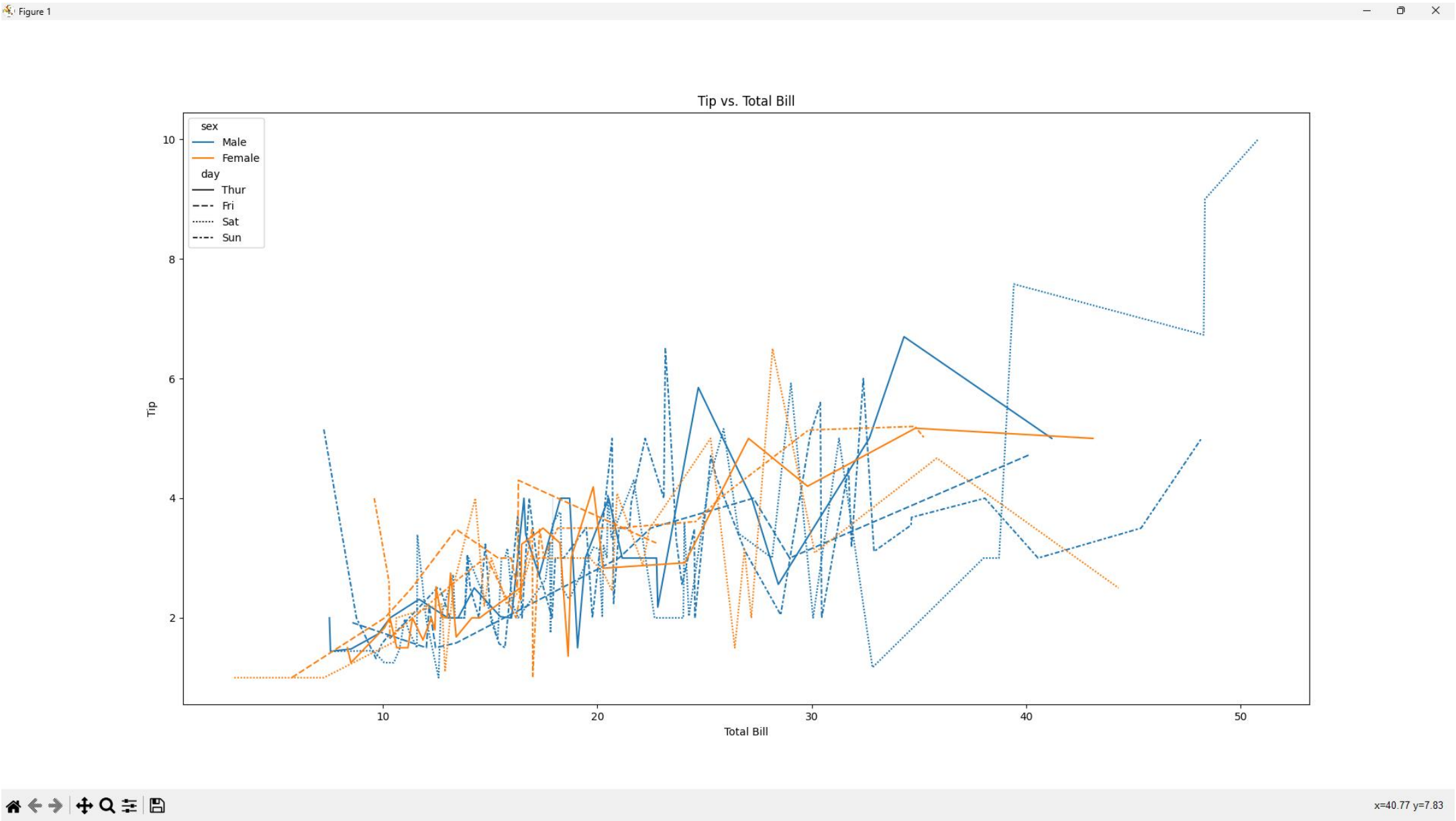


Figure 1
Tip vs. Total Bill
sex
Male
Female
day
Thur
Fri
Sat
Sun
Tip
Total Bill
10
8
6
4
2
10
20
30
40
50
x=40.77 y=7.83

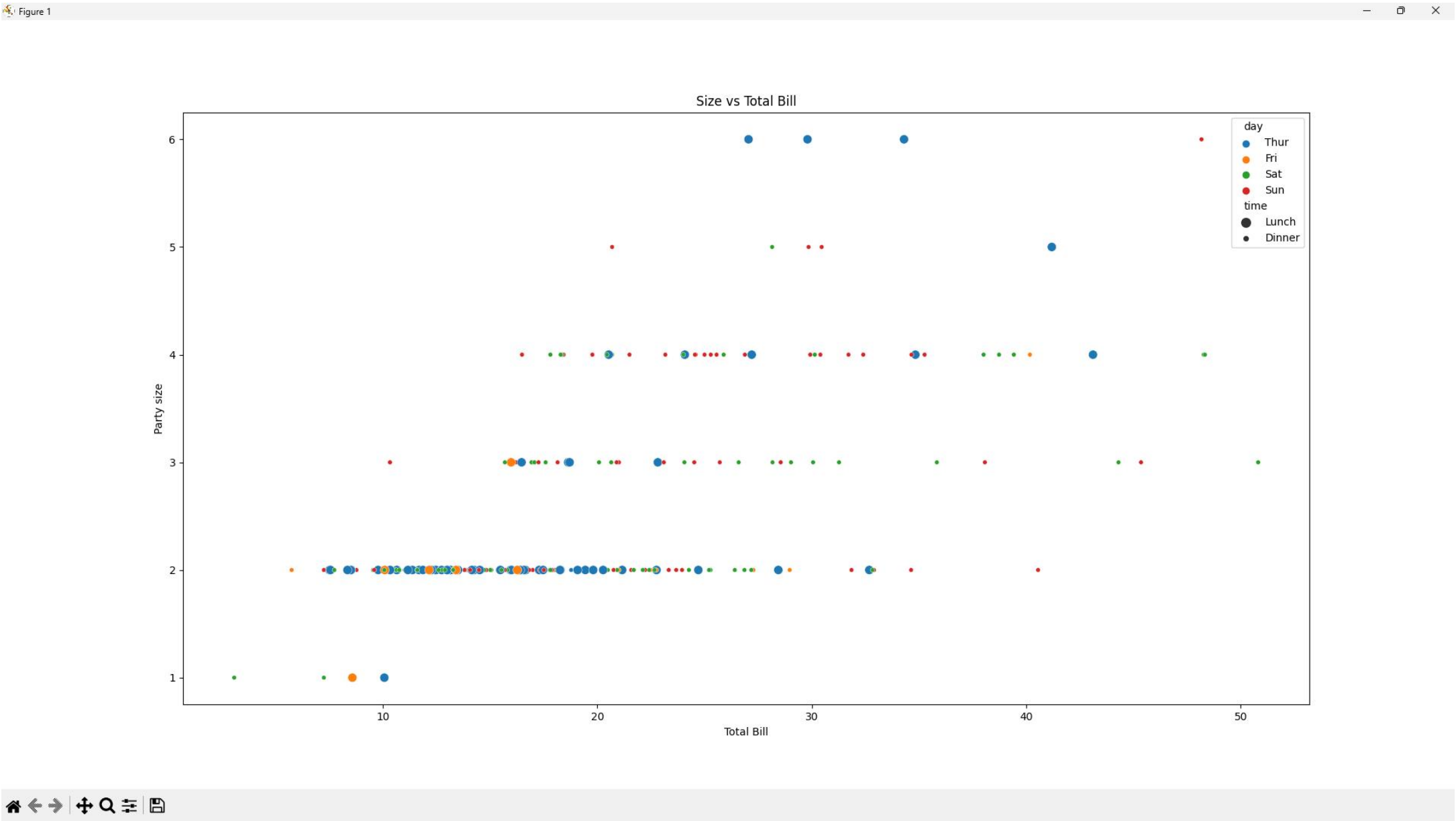


Figure 1
Size vs Total Bill
Party size
Total Bill
1
2
3
4
5
6
10
20
30
40
50
day
Thur
Fri
Sat
Sun
time
Lunch
Dinner

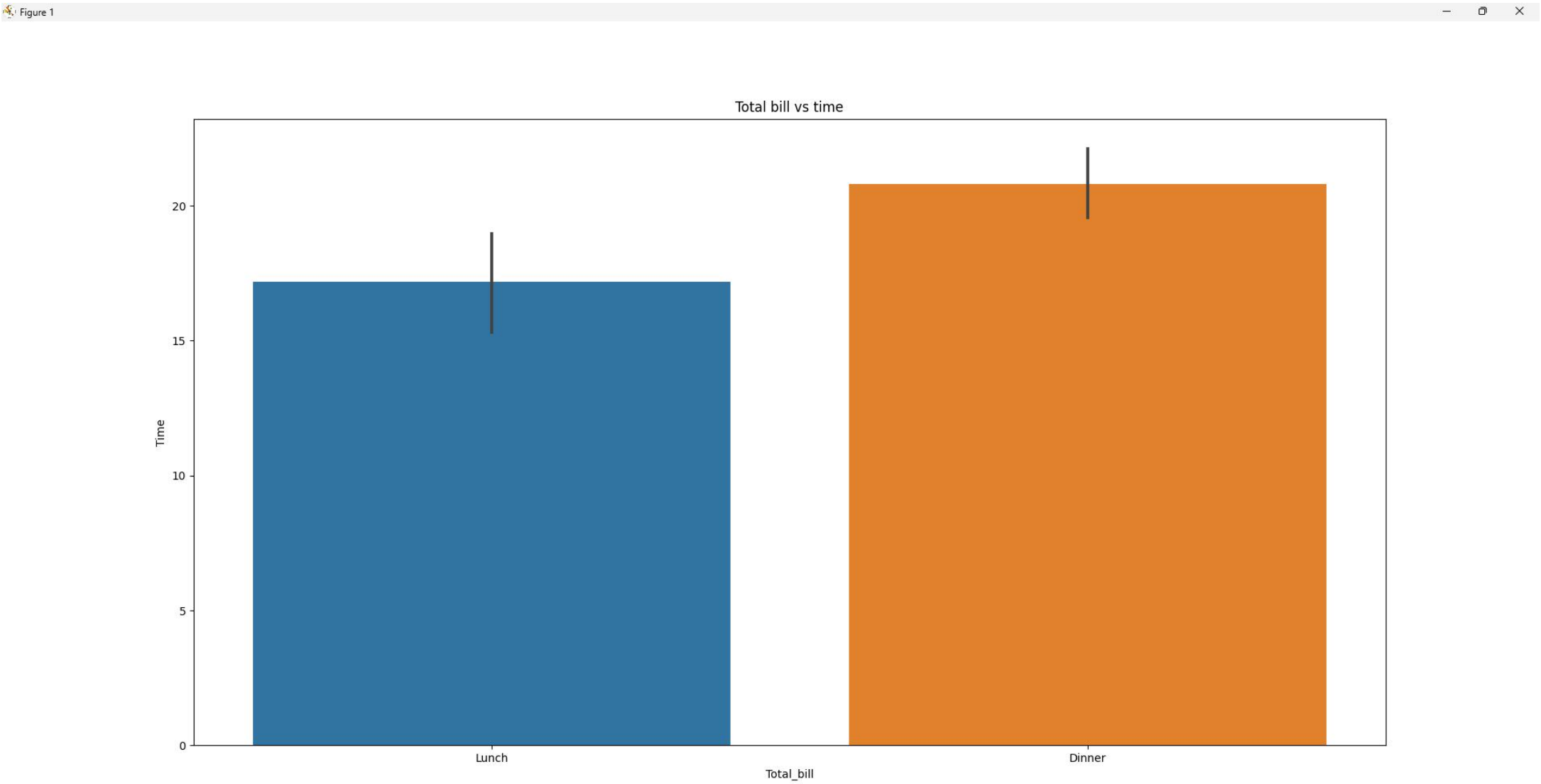


Figure 1
Total bill vs time
Time
0
5
10
15
20
Lunch
Dinner
Total_bill

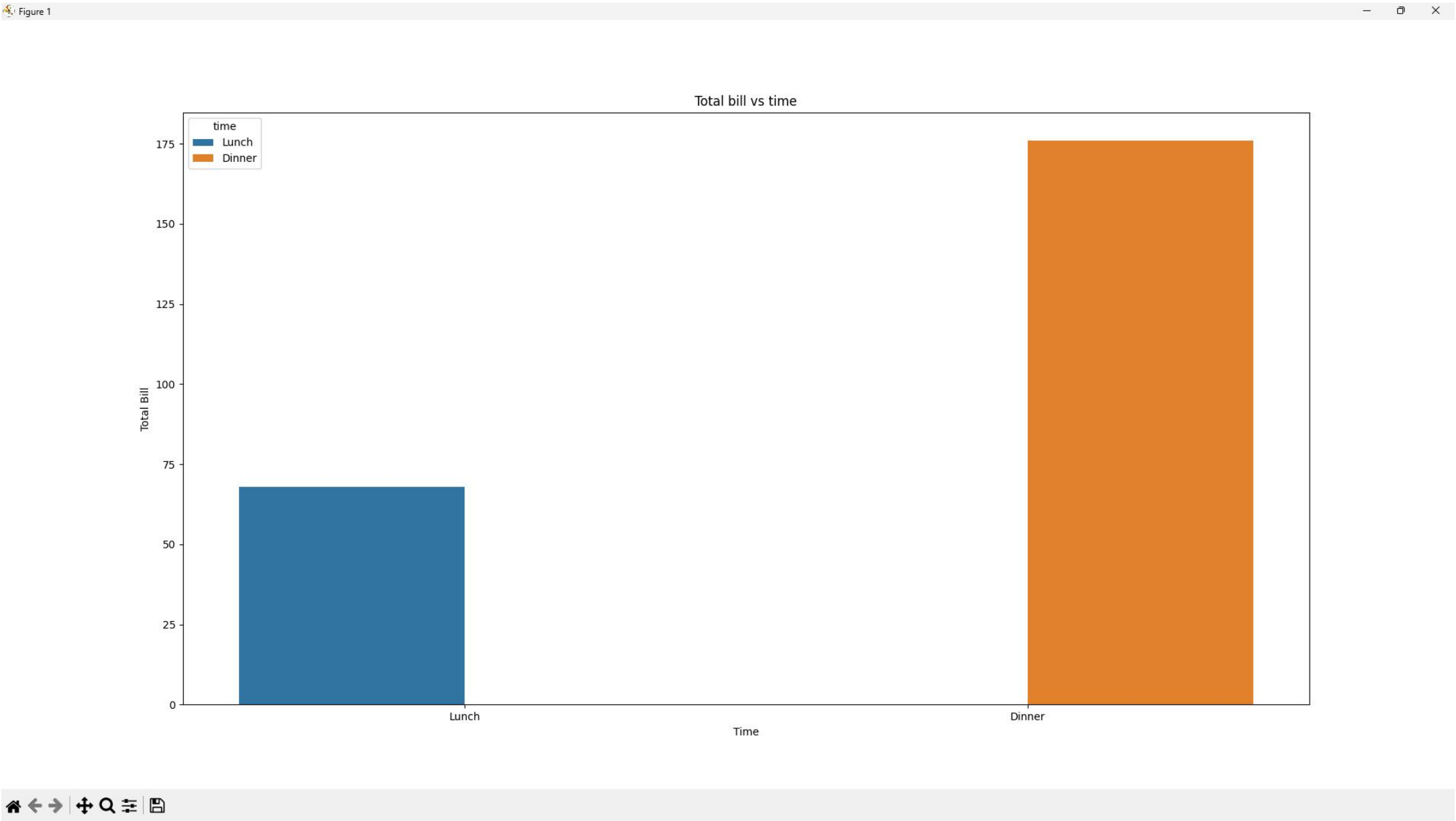


Figure 1
Total bill vs time
time
Lunch
Dinner
175
150
125
100
75
50
25
0
Total Bill
Lunch
Dinner
Time

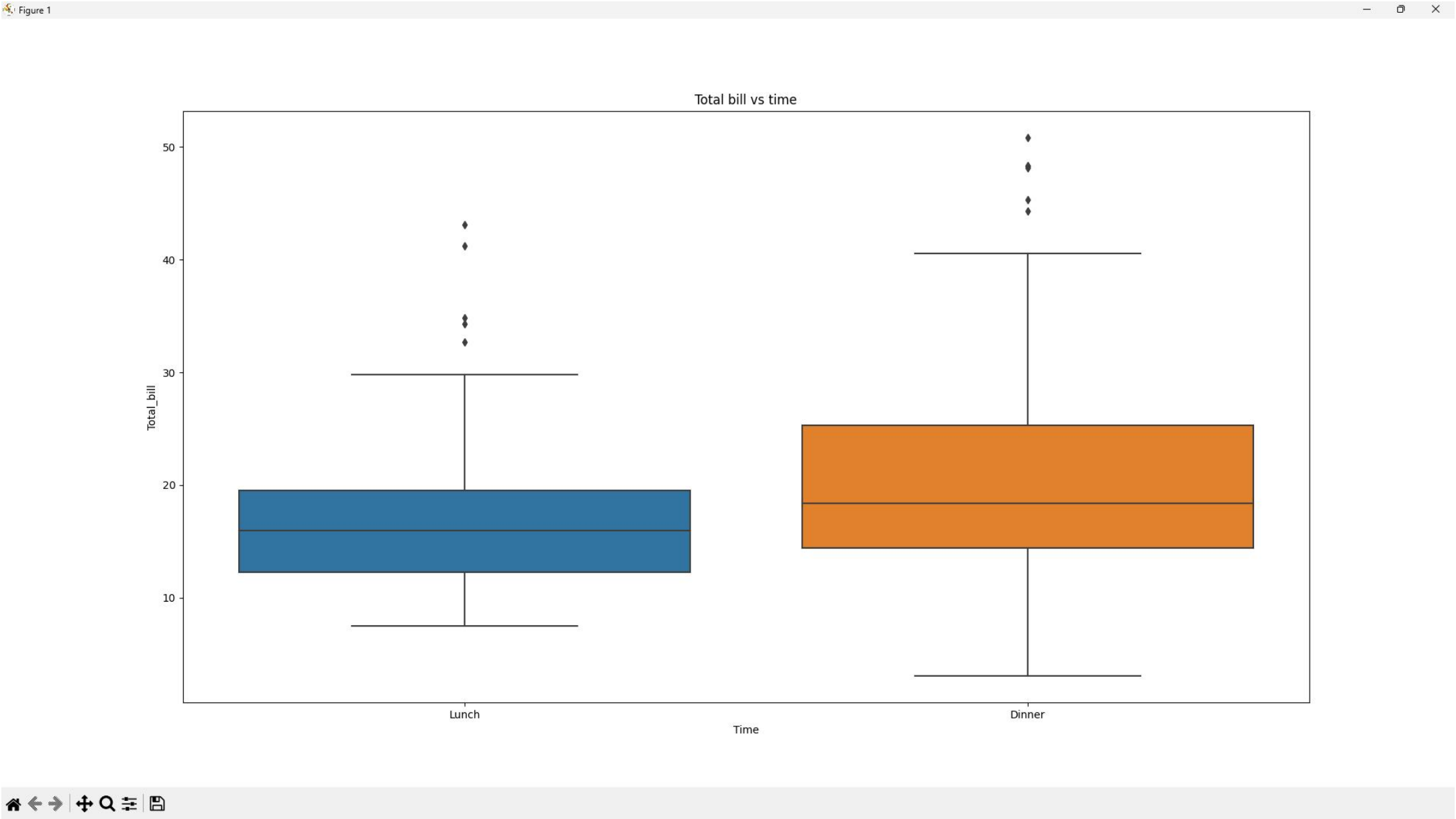


Figure 1
Total bill vs time
Total_bill
50
40
30
20
10
Lunch
Dinner
Time

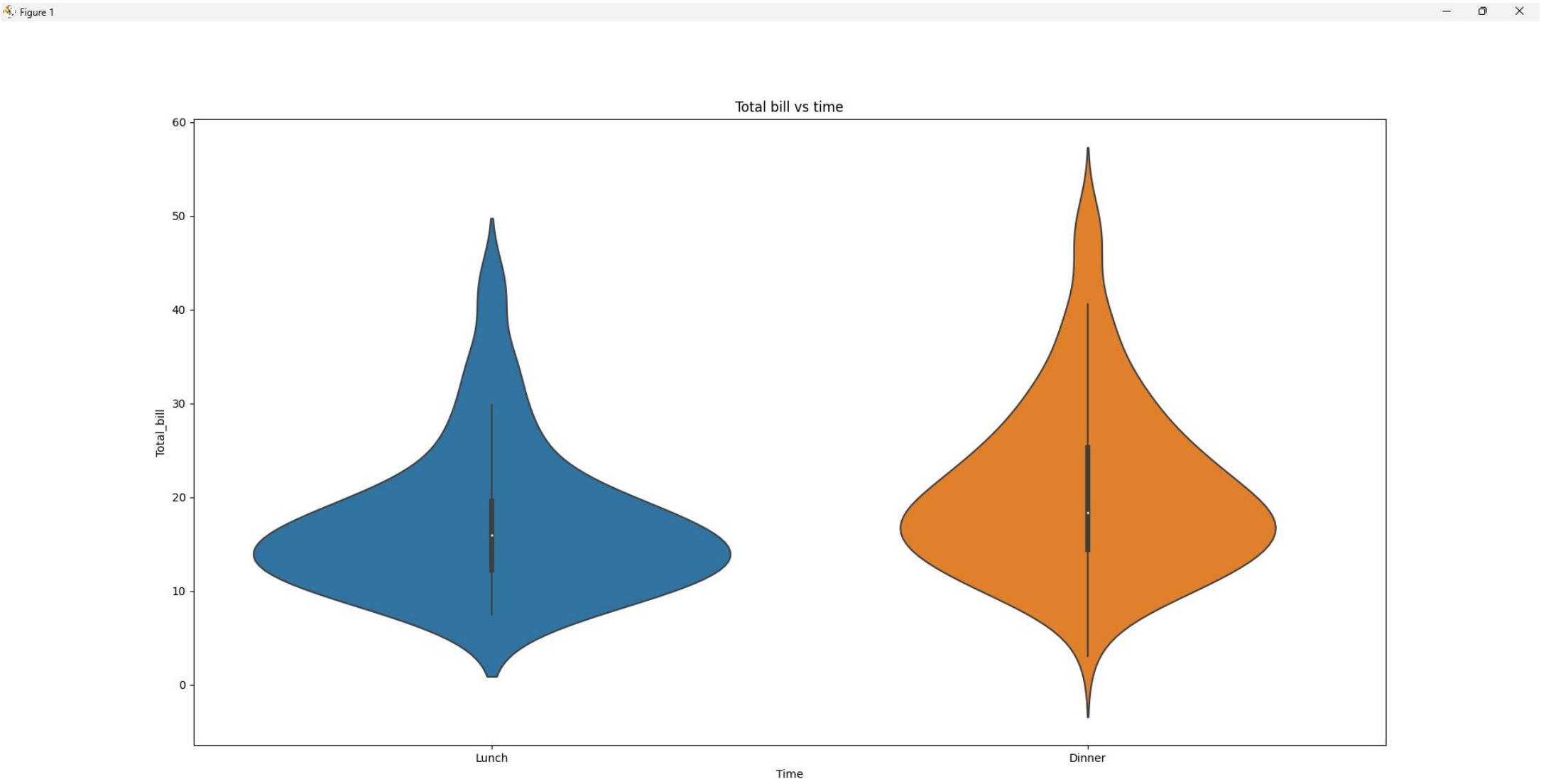


Figure 1
Total bill vs time
60
50
40
30
20
10
0
Total_bill
Lunch
Dinner
Time

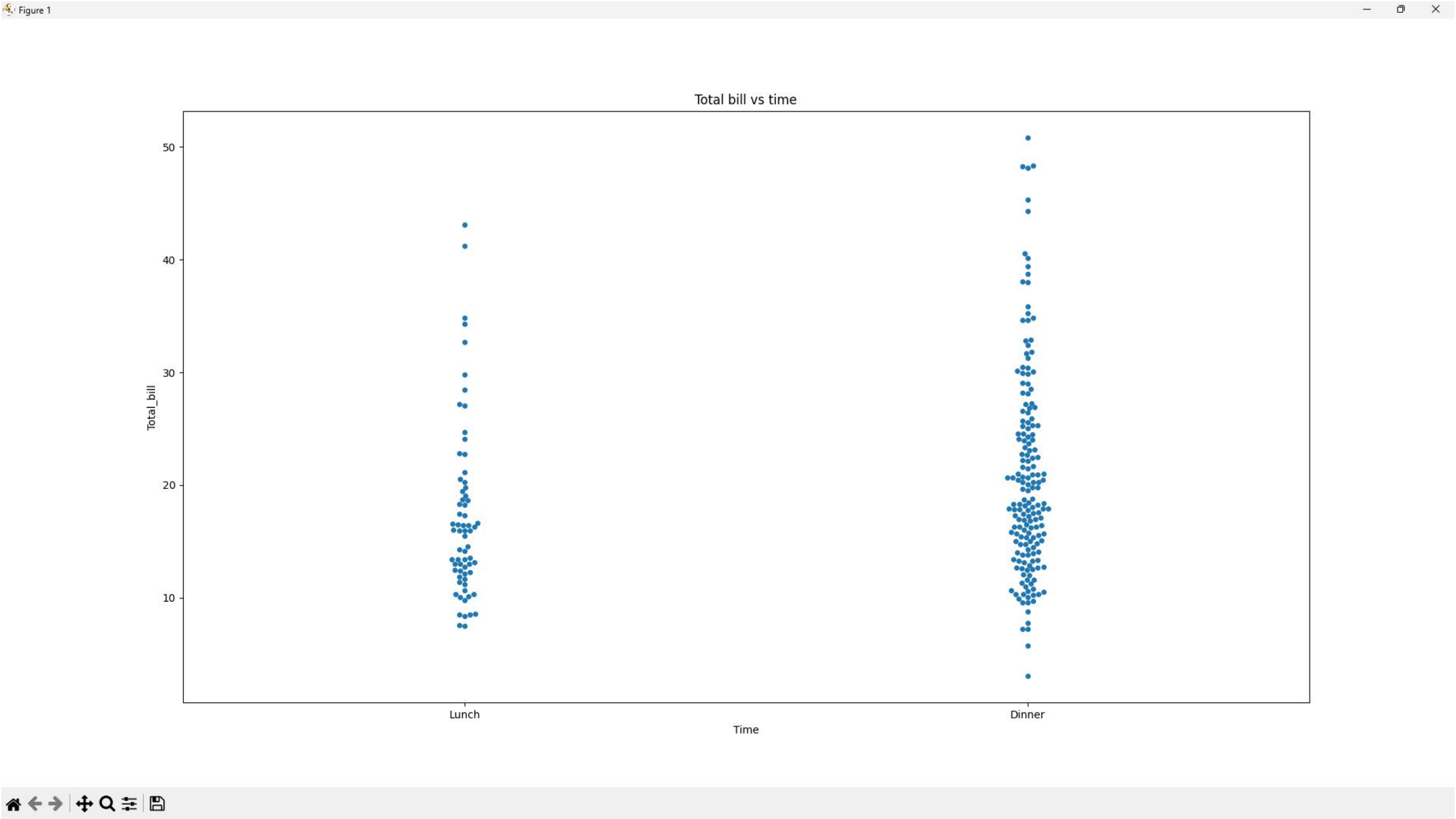


Figure 1
Total bill vs time
Total_bill
50
40
30
20
10
Lunch
Dinner
Time

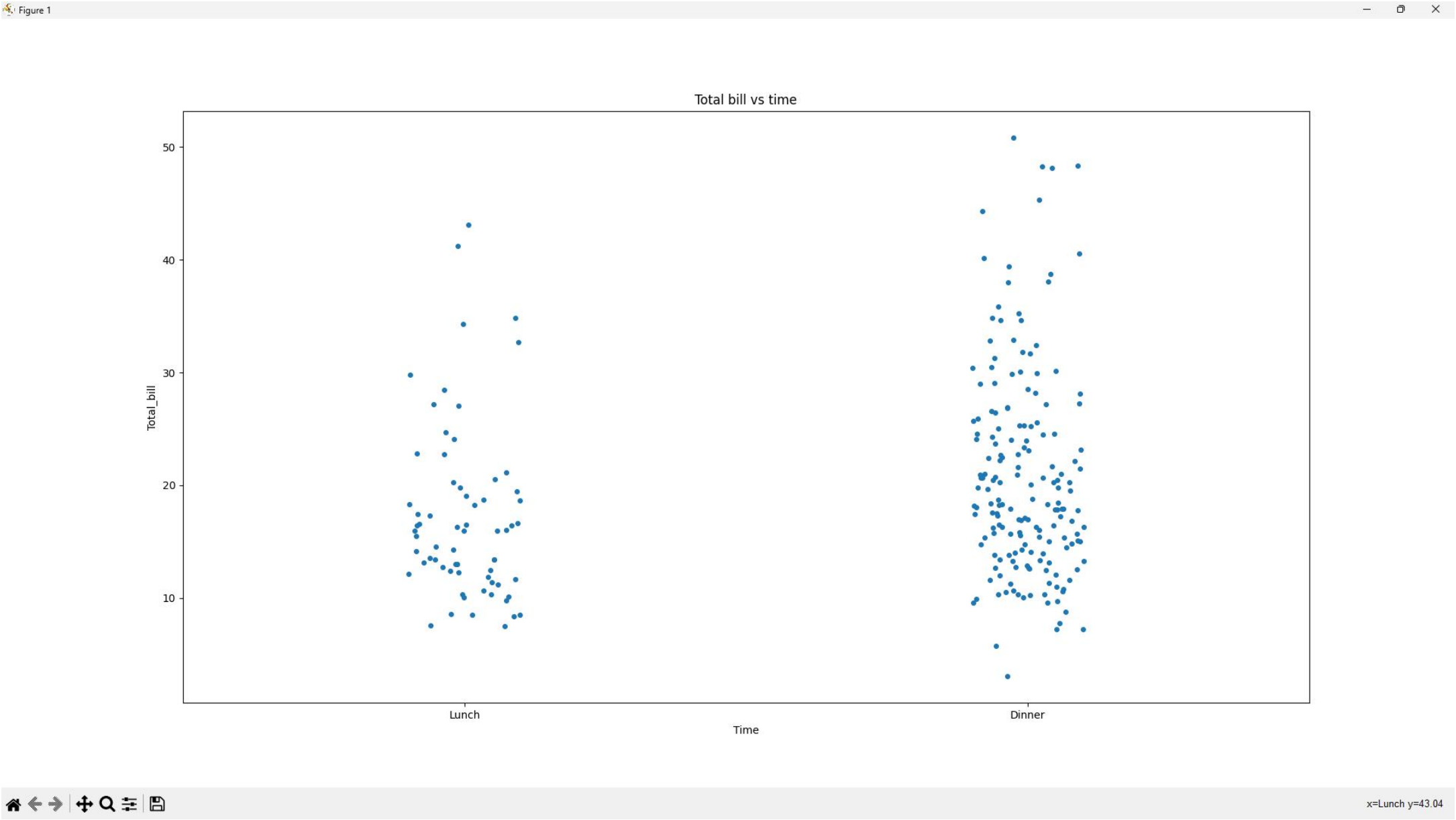


Figure 1
Total bill vs time
Total_bill
50
40
30
20
10
Lunch
Dinner
Time
x=Lunch y=43.04

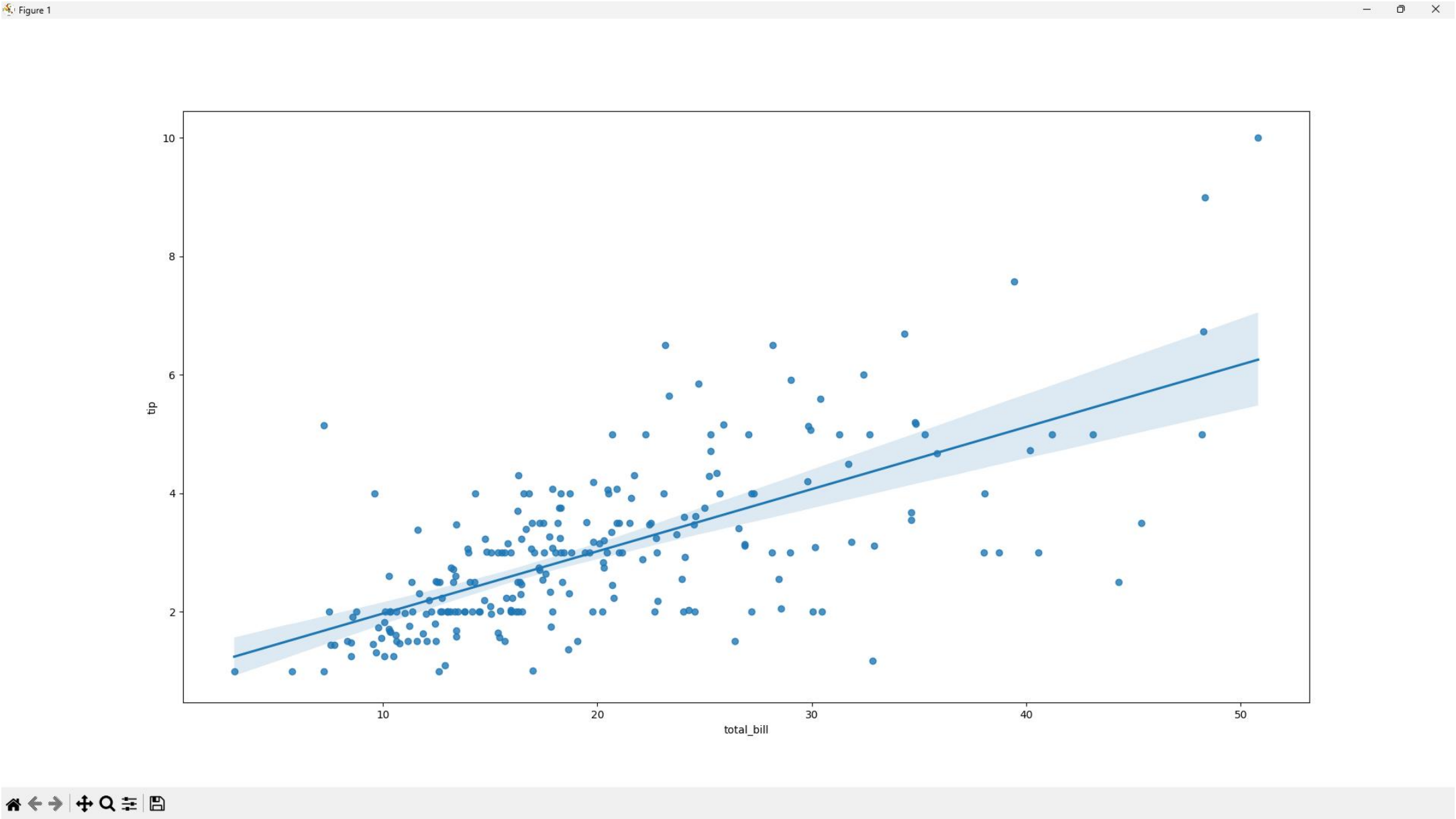


Figure 1
10
8
6
4
2
tip
10
20
30
40
50
total_bill

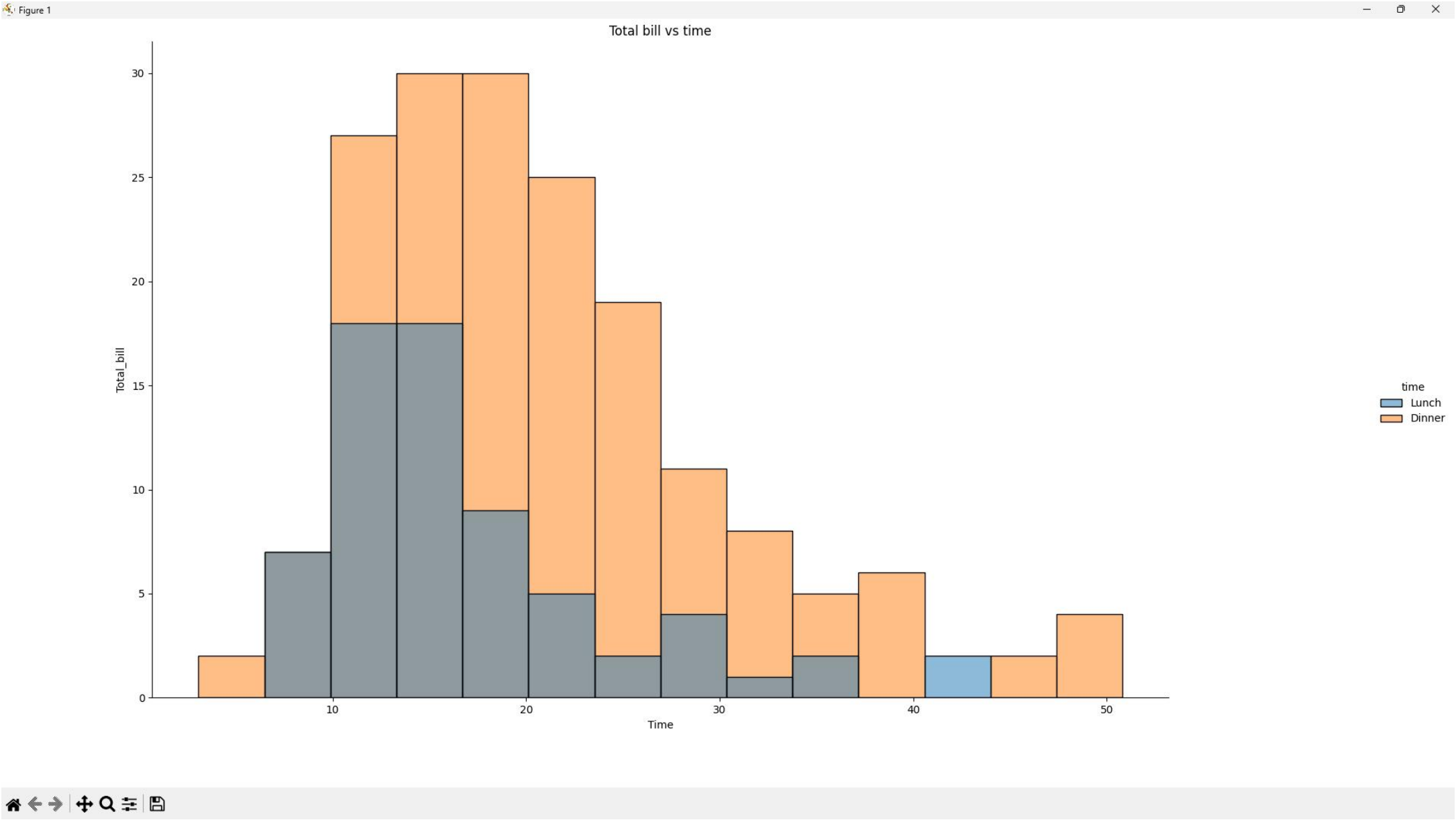


Figure 1
Total bill vs time
Total_bill
Time
0
5
10
15
20
25
30
10
20
30
40
50
time
Lunch
Dinner

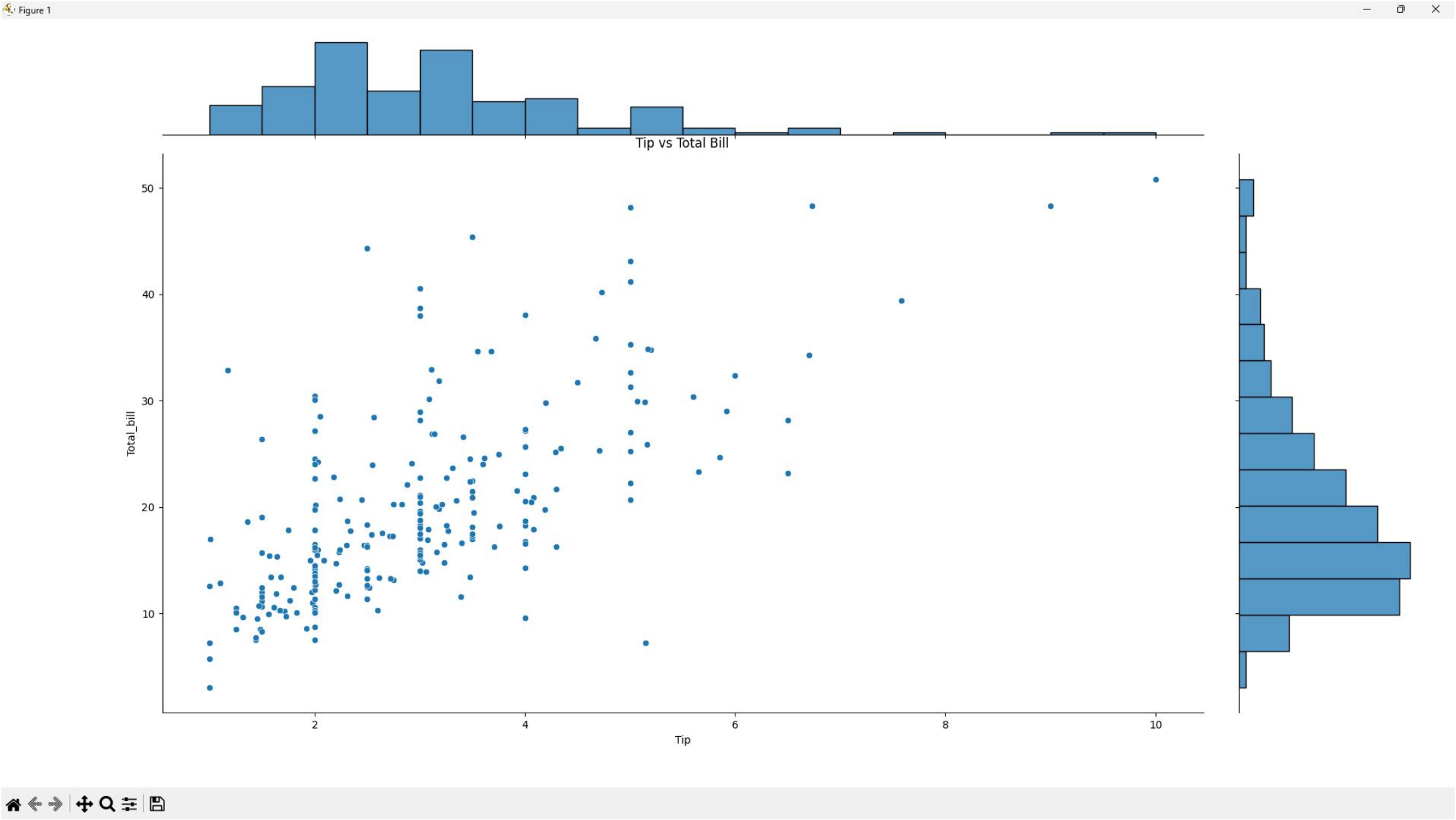


Figure 1
Tip vs Total Bill
Total_bill
Tip
50
40
30
20
10
2
4
6
8
10

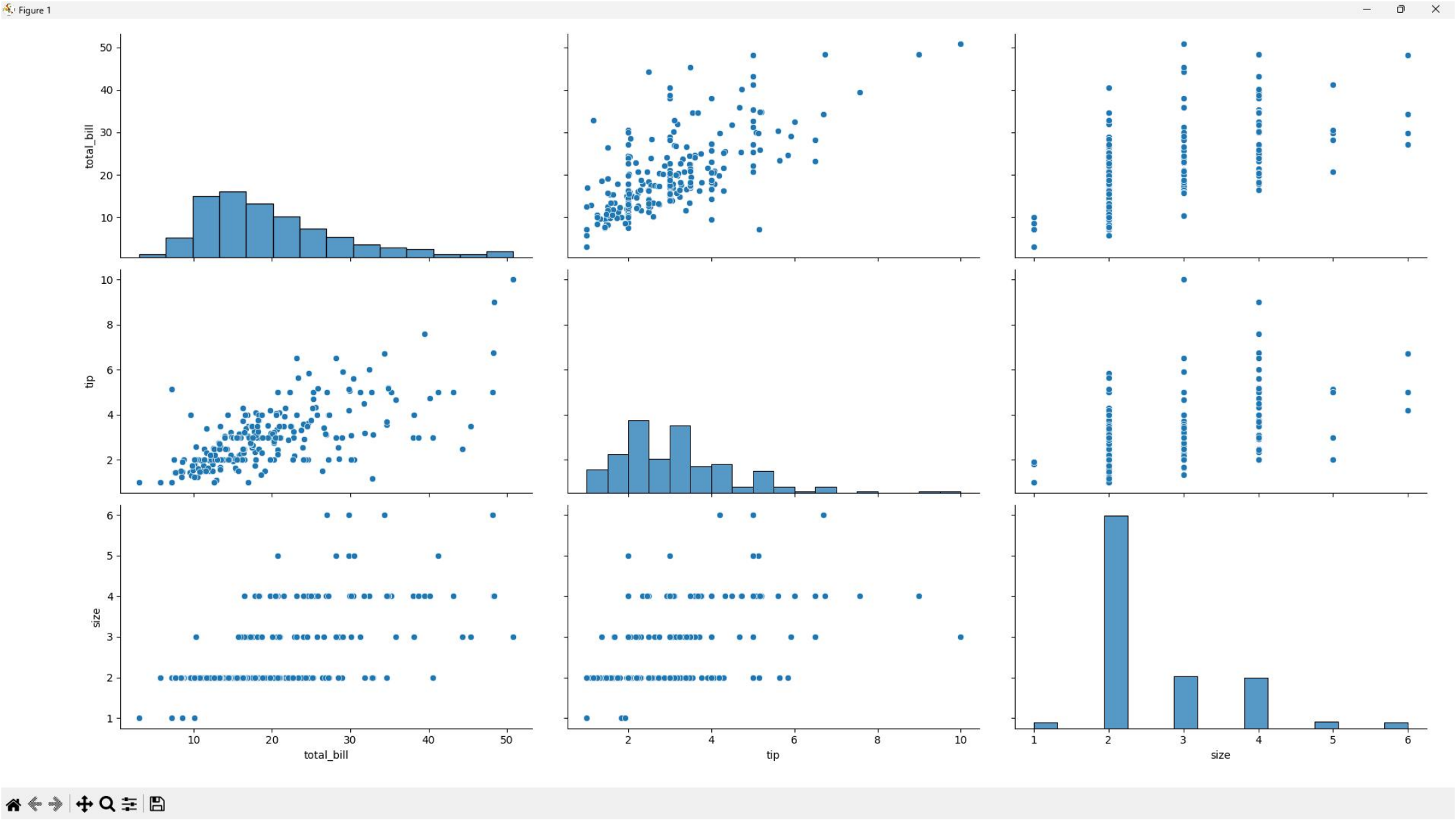


Figure 1
total_bill
tip
size
total_bill
tip
size

GE ASSIGNMET 9(SEM2).py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMET 9(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
#Create a folium MAP object centred around your HOME state
import folium
import matplotlib.pyplot as plt
#plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5941,85.1376],width=500,height=300)
WorldMap

#Adjust Zoom feature of the MAP object using various available zoom control functions
import folium
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5941,85.1376],
                    zoom_start=12,
                    zoom_control=False,
                    min_zoom=3,max_zoom=20,
                    width=500,height=300)
WorldMap
WorldMap.save('patna.html')


#Add a 'Circular Marker' at your residence location. In 'popup', display the full address of your residence
import folium
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5941,85.1376],
                    zoom_start=10,
                    zoom_control=False,
                    min_zoom=5,max_zoom=15,
                    width=500,height=300)
folium.CircleMarker(location=[25.5941,85.1376],radius=50,colour='red',
                    weight=1,fill=True,fill_colour='blue',
                    fill_opacity=0.5,
                    popup='patna,Airport').add_to(WorldMap)
(WorldMap)
```

Ln: 76 Col: 24

GE ASSIGNMET 9(SEM2).py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMET 9(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
                    weight=1,fill=True,fill_colour='blue',
                    fill_opacity=0.5,
                    popup='patna,Airport').add_to(WorldMap)
(WorldMap)
WorldMap.save('patna.html')

#Add Multiple Location Markers (with correct icon style)
#Add a location marker at your Residence location
import folium
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5950,85.0908],
                    zoom_start=10,
                    zoom_control=False,
                    min_zoom=5,max_zoom=15,
                    width=500,height=300)
folium.Marker(location=[25.5950, 85.0908],
                    popup='',
                    icon=folium.Icon(color='blue',icon_color='white',icon='home')).add_to(WorldMap)
folium.Marker(location=[25.5950, 85.0908],
                    popup='Rajinder Nagar,Patna',
                    icon=folium.Icon(colour='red',icon_color='white',icon='plane')).add_to(WorldMap)
(WorldMap)
WorldMap.save('up.html')


#Add a straight line path between two location markers and calculate distance between them
import folium
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5941,85.1376],
                    zoom_start=10,
                    zoom_control=False,
                    min_zoom=5,max_zoom=15,
```

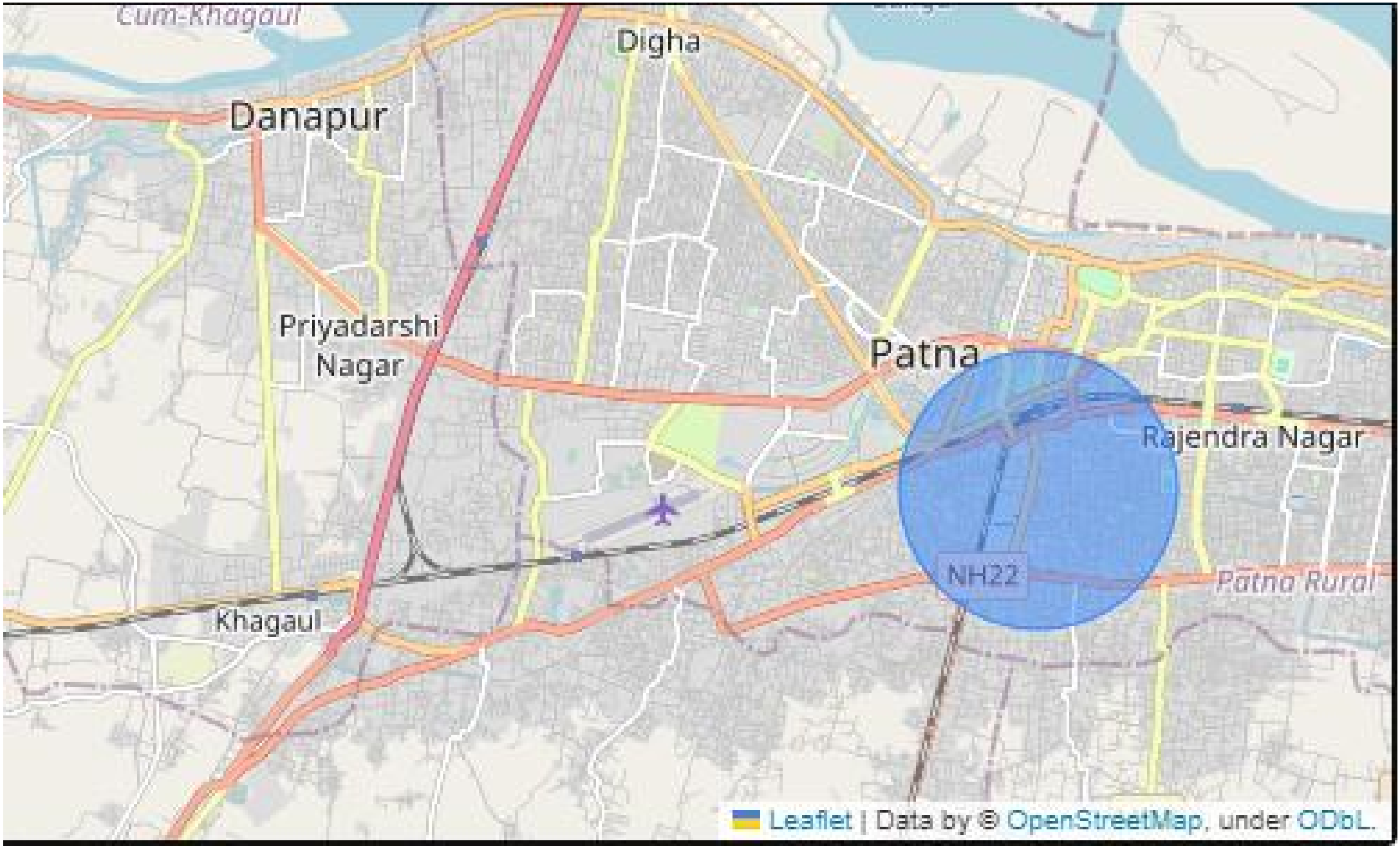
Ln: 77 Col: 0

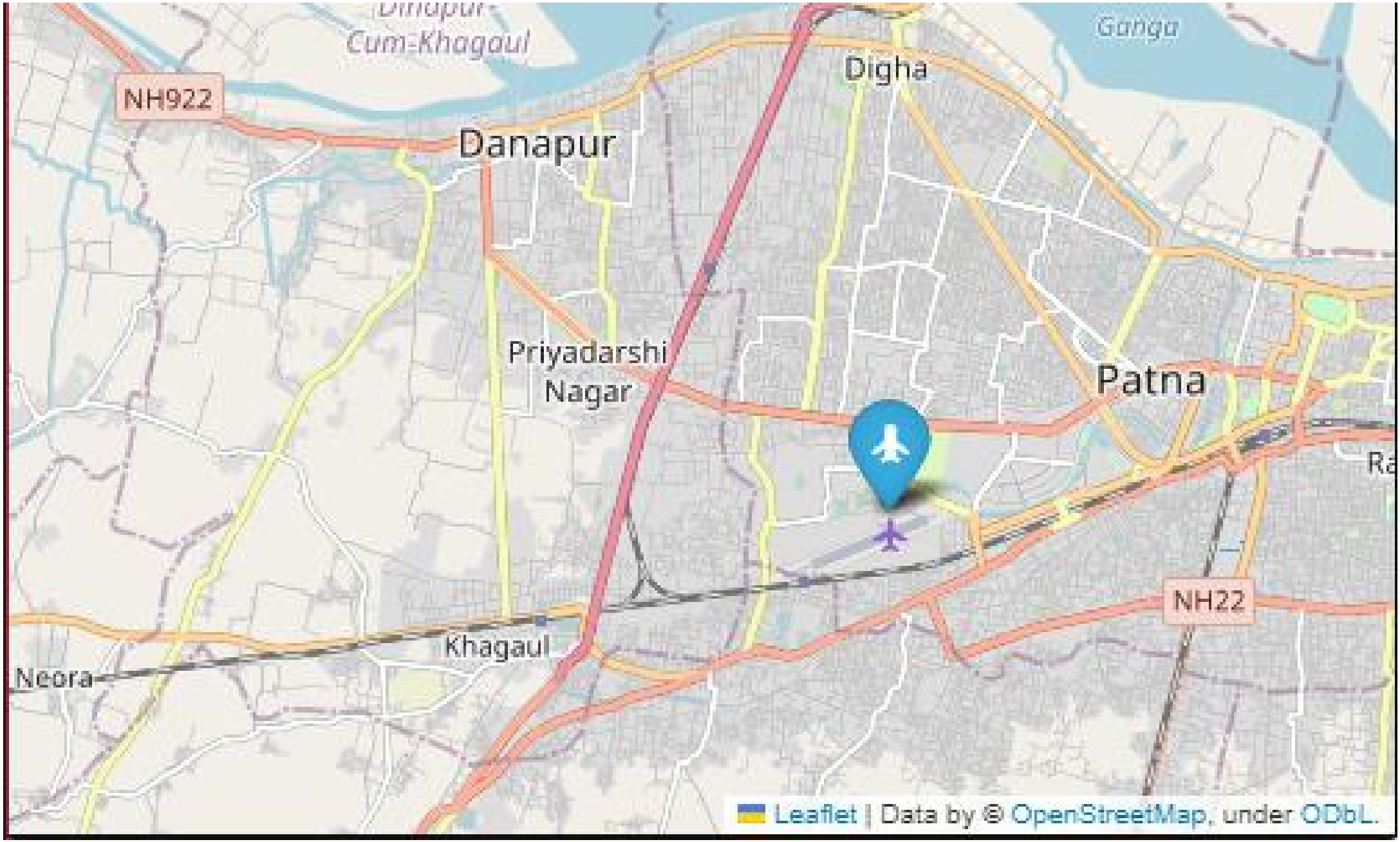
GE ASSIGNMET 9(SEM2).py - C:/Users/Harsh/AppData/Local/Programs/Python/Python311/GE ASSIGNMET 9(SEM2).py (3.11.0)

File Edit Format Run Options Window Help

```
                     zoom_control=False,
                     min_zoom=5,max_zoom=15,
                     width=500,height=300)
folium.Marker(location=[25.5950, 85.0908],
                    popup='',
                    icon=folium.Icon(color='blue',icon_color='white',icon='home')).add_to(WorldMap)
folium.Marker(location=[25.5950, 85.0908],
                    popup='Rajinder Nagar,Patna',
                    icon=folium.Icon(colour='red',icon_color='white',icon='plane')).add_to(WorldMap)
(WorldMap)
WorldMap.save('up.html')


#Add a straight line path between two location markers and calculate distance between them
import folium
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4))
WorldMap=folium.Map(location=[25.5941,85.1376],
                     zoom_start=10,
                     zoom_control=False,
                     min_zoom=5,max_zoom=15,
                     width=500,height=300)
folium.Marker(location=[25.5941,85.1376],
                    popup='New Behsa,Transport nagar,Lucknow,Uttar Pradesh',
                    icon=folium.Icon(colour='blue',icon_color='white',icon='home')).add_to(WorldMap)
folium.Marker(location=[25.6055,85.1500],
                    popup='New Behsa,Transport nagar,Lucknow,Uttar Pradesh',
                    icon=folium.Icon(colour='red',icon_color='white',icon='plane')).add_to(WorldMap)
D=distance=([25.5941,85.1376],[25.6055,85.1500])

folium.PolyLine([[25.5941,85.1376],[25.6055,85.1500]],tooltip=D).add_to(WorldMap)
(WorldMap)
WorldMap.save('an.html')
```

Ln: 77 Col: 0


Cum-Khagaul
Digha
Danapur
Priyadarshi
Nagar
Patna
Rajendra Nagar
NH22
Patna Rural
Khagaul
Leaflet | Data by © OpenStreetMap, under ODbL.


Cum-Khagaul
Ganga
NH922
Digha
Danapur
Priyadarshi
Nagar
Patna
NH22
Khagaul
Neora
Leaflet | Data by © OpenStreetMap, under ODbL.

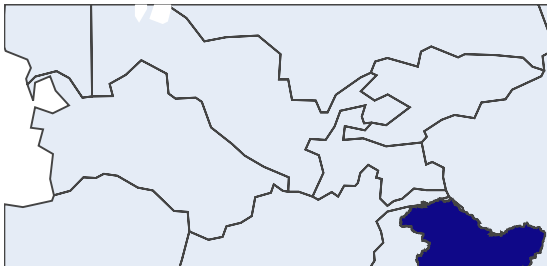

Marine Drive
Ashok Raj Path
Nala Road
Patna
West Boring Canal Road
Atal Path
Bailey Road
Atal Path
Rajendra Nagar
Kumhrar Road
Leaflet | Data by © OpenStreetMap, under ODbL.

```
# 1> Display a Choropleth Map
# i> To show variation of Literacy in different Indian states
import json
import pandas as pd
import plotly.express as px
INDIA_MAP=json.load(open('/content/states_india.geojson','r'))
INDIA_DATA=pd.read_csv('India-Population1.csv')
MAP=px.choropleth(INDIA_DATA,geojson=INDIA_MAP,
locations='State',
featureidkey="properties.st_nm",
color='Literacy',scope='asia',fitbounds='locations')
MAP.show()
```

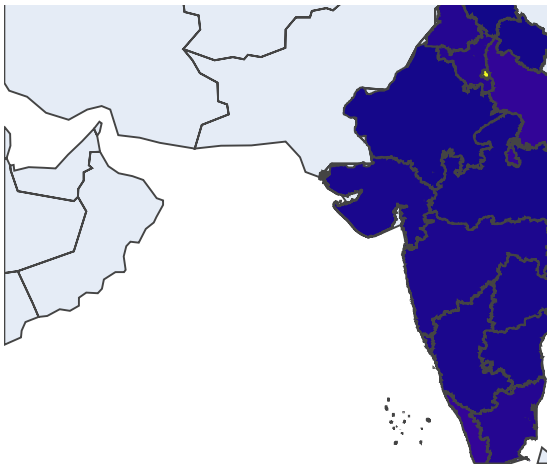
```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-7-6f613819e1ac> in <cell line: 6>()
      4 import pandas as pd
      5 import plotly.express as px
----> 6 INDIA_MAP=json.load(open('/content/states_india.geojson','r'))
      7 INDIA_DATA=pd.read_csv('India-Population1.csv')
      8 MAP=px.choropleth(INDIA_DATA,geojson=INDIA_MAP,

FileNotFoundError: [Errno 2] No such file or directory: '/content
/states_india.geojson'
```

SEARCH STACK OVERFLOW

```
# 1> Display a Choropleth Map
# i> To show variation of Density in different Indian states
import json
import pandas as pd
import plotly.express as px
INDIA_MAP=json.load(open('/content/states_india.geojson','r'))
INDIA_DATA=pd.read_csv('India-Population1.csv')
MAP=px.choropleth(INDIA_DATA,geojson=INDIA_MAP,
locations='State',
featureidkey="properties.st_nm",
color='Density',scope='asia',fitbounds='locations')
MAP.show()
```



0s completed at 10:07 PM



```
# 2. i> Load the above given dataset from plotly library using px.data.tips()
import plotly.express as px
DataTips = px.data.tips()
display(DataTips)
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

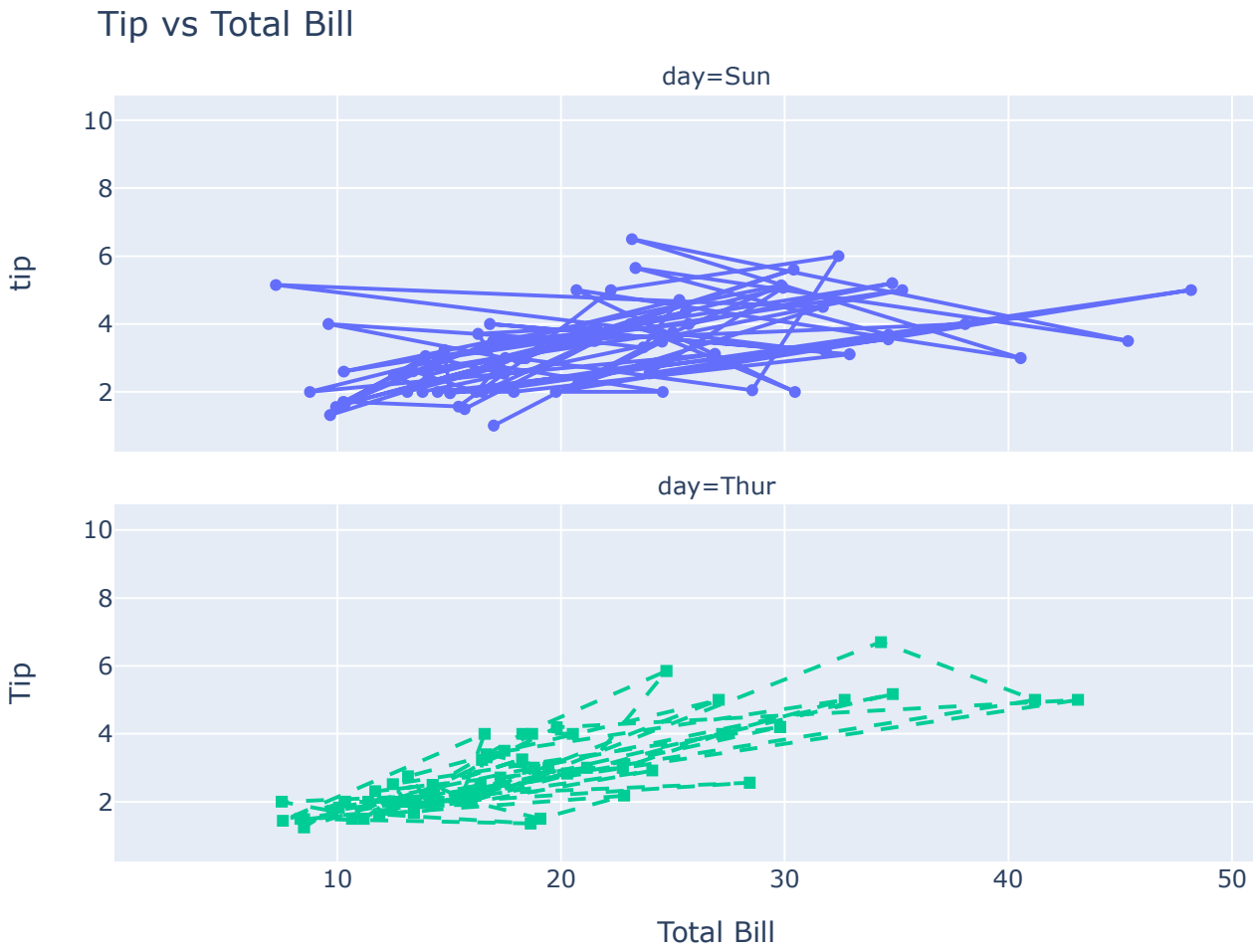
244 rows × 7 columns

```
# ii> Display a LINE plot for 'tip' Vs. 'total_bill' classified according to 'sex' and 'day
# a. colour of the plots should be different for different type of 'sex'
import plotly.express as px
```

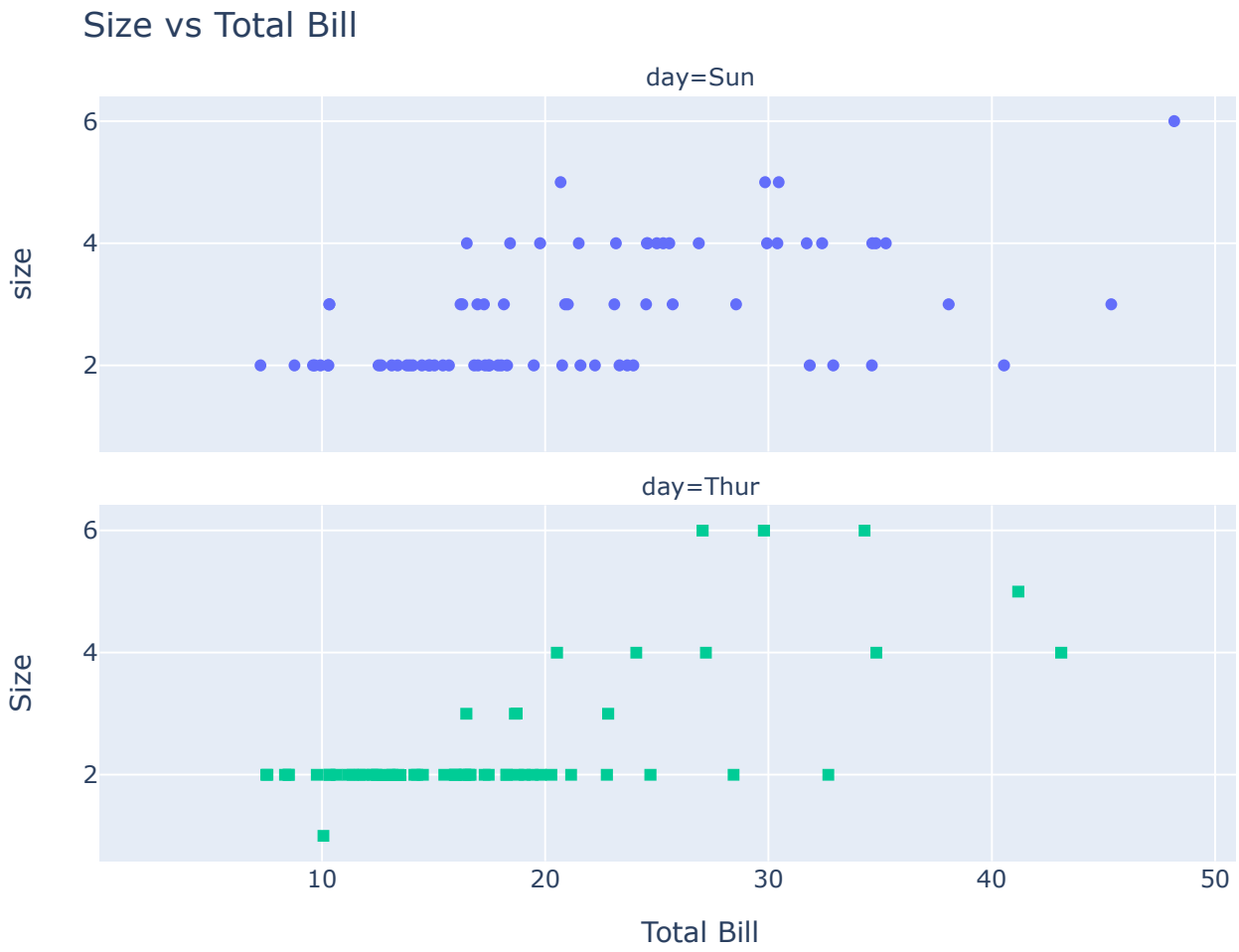
```
data = px.data.tips()
fig = px.line(data, x='total_bill', y='tip', color='sex', line_group='sex',
              facet_col='day', facet_col_wrap=2,
              markers=True, symbol='day', line_dash='day')

fig.update_layout(title='Tip vs Total Bill',
                  xaxis_title='Total Bill',
                  yaxis_title='Tip',
                  legend_title='Sex')

fig.show()
```


```
# ii> Display a LINE plot for 'tip' Vs. 'total_bill' classified according to 'sex' and 'day
# b. marker type and plot style should be different for different 'day'
import plotly.express as px
data = px.data.tips()
fig = px.line(data, x='total_bill', y='tip', color='day', line_group='sex',
              facet_col='day', facet_col_wrap=2,
              markers=True, symbol='day', line_dash='day')

fig.update_layout(title='Tip vs Total Bill',
                  xaxis_title='Total Bill',
```

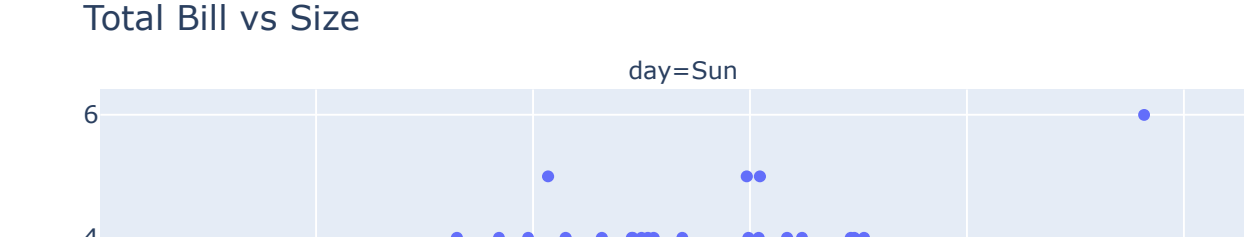
```
                  xaxis_title= 'Total Bill',
                  yaxis_title='Tip',
                  legend_title='day')

fig.show()
```

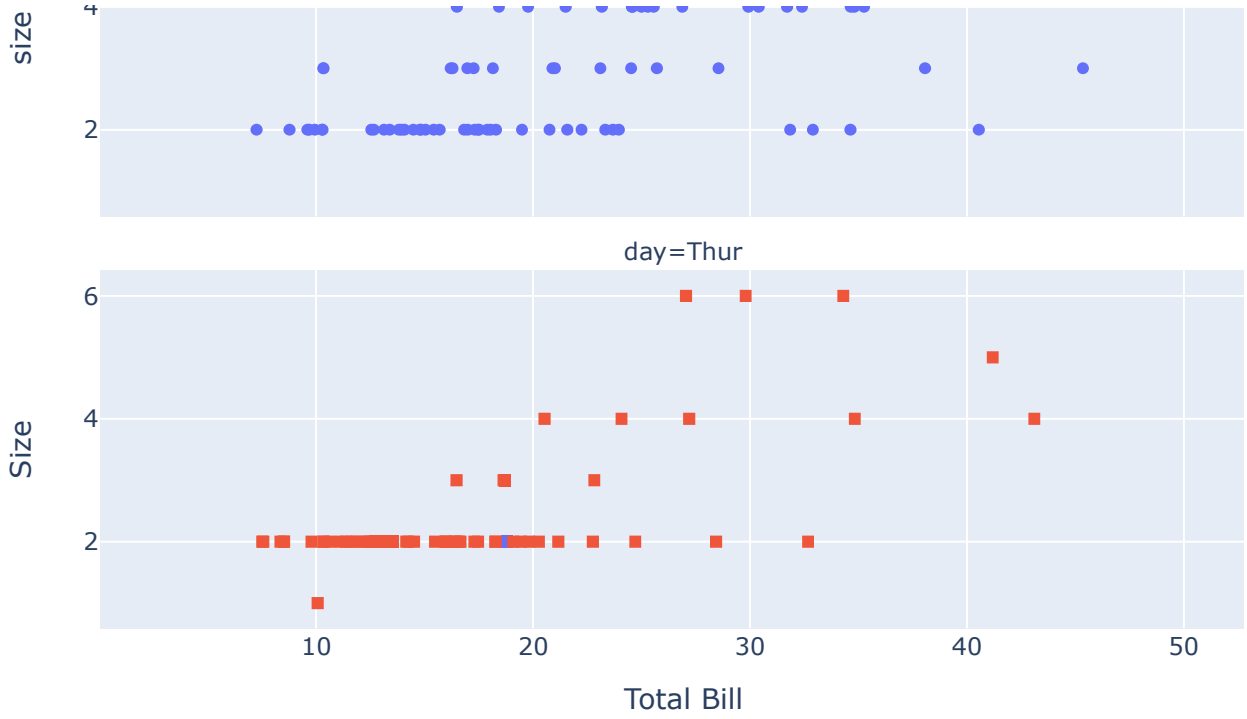



```
# iii. Display a bubble chart for 'total_bill' Vs. 'size'
# a. bubble colour should be classified according to 'day'

import plotly.express as px
data = px.data.tips()
fig = px.scatter(data, x='total_bill', y='size', color='day',
                 facet_col='day', facet_col_wrap=2,
                 symbol='day')

fig.update_layout(title='Size vs Total Bill',
                  xaxis_title='Total Bill',
                  yaxis_title='Size',
                  legend_title='Day')

fig.show()
```

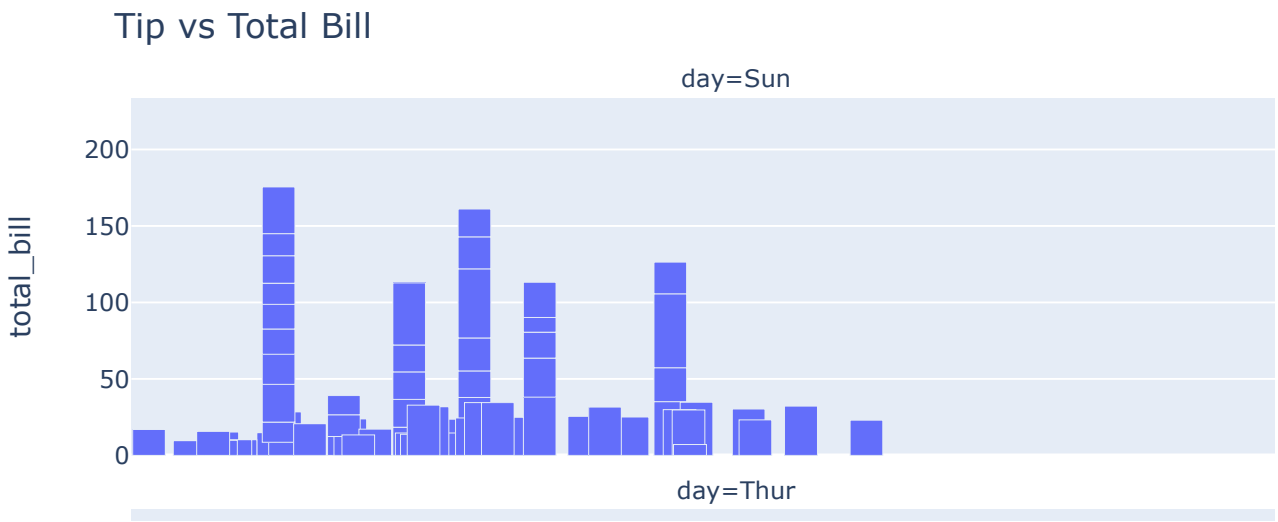


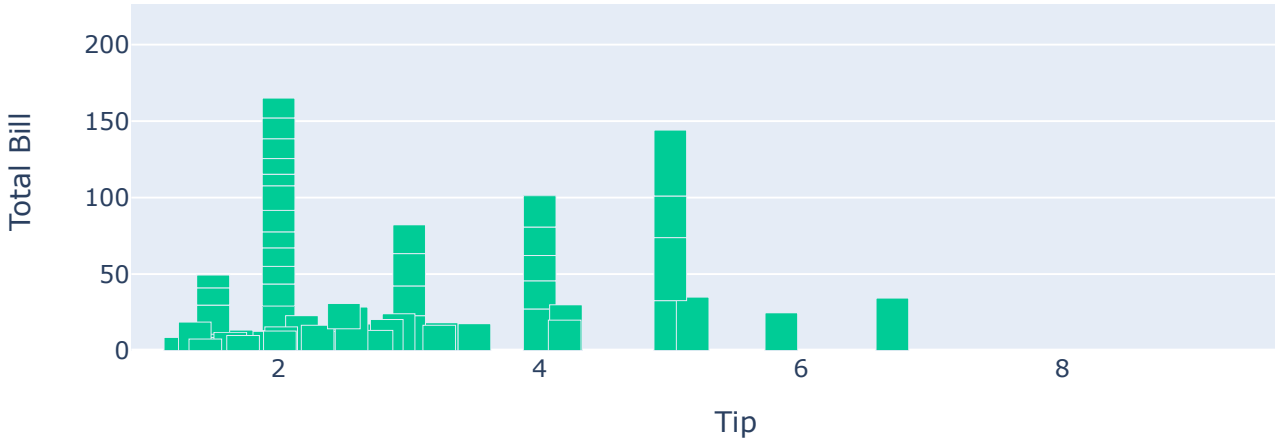

```
# iii. Display a bubble chart for 'total_bill' Vs. 'size'
# b. bubble size should be classified according to 'time'
import plotly.express as px
data = px.data.tips()
fig = px.scatter(data, x='total_bill', y='size', color='time',
              facet_col='day', facet_col_wrap=2, symbol='day')

fig.update_layout(title='Total Bill vs Size',
                  xaxis_title='Total Bill',
                  yaxis_title='Size',
                  legend_title='day')

fig.show()
```



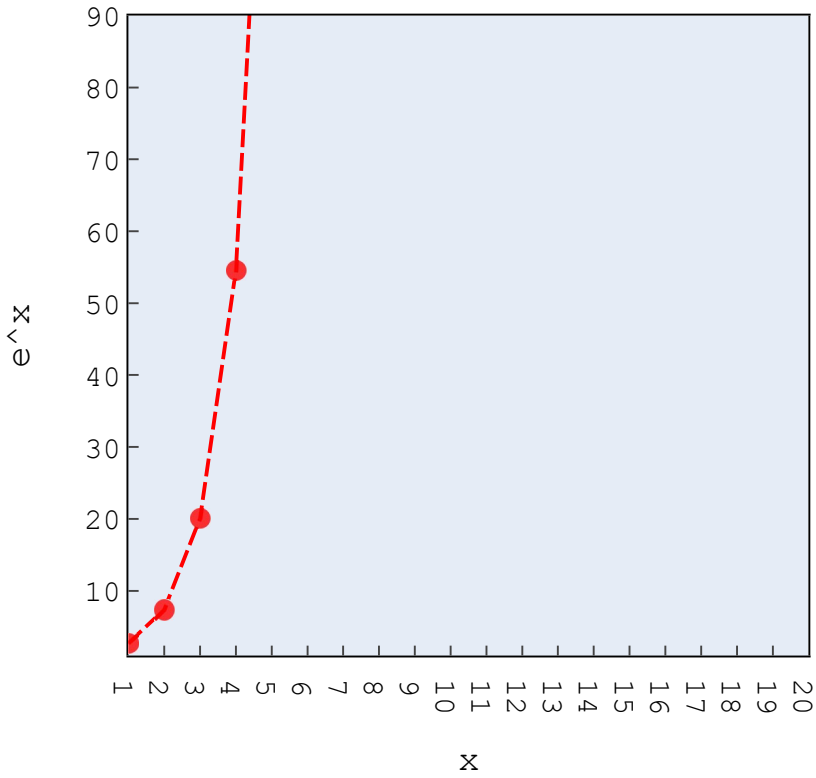


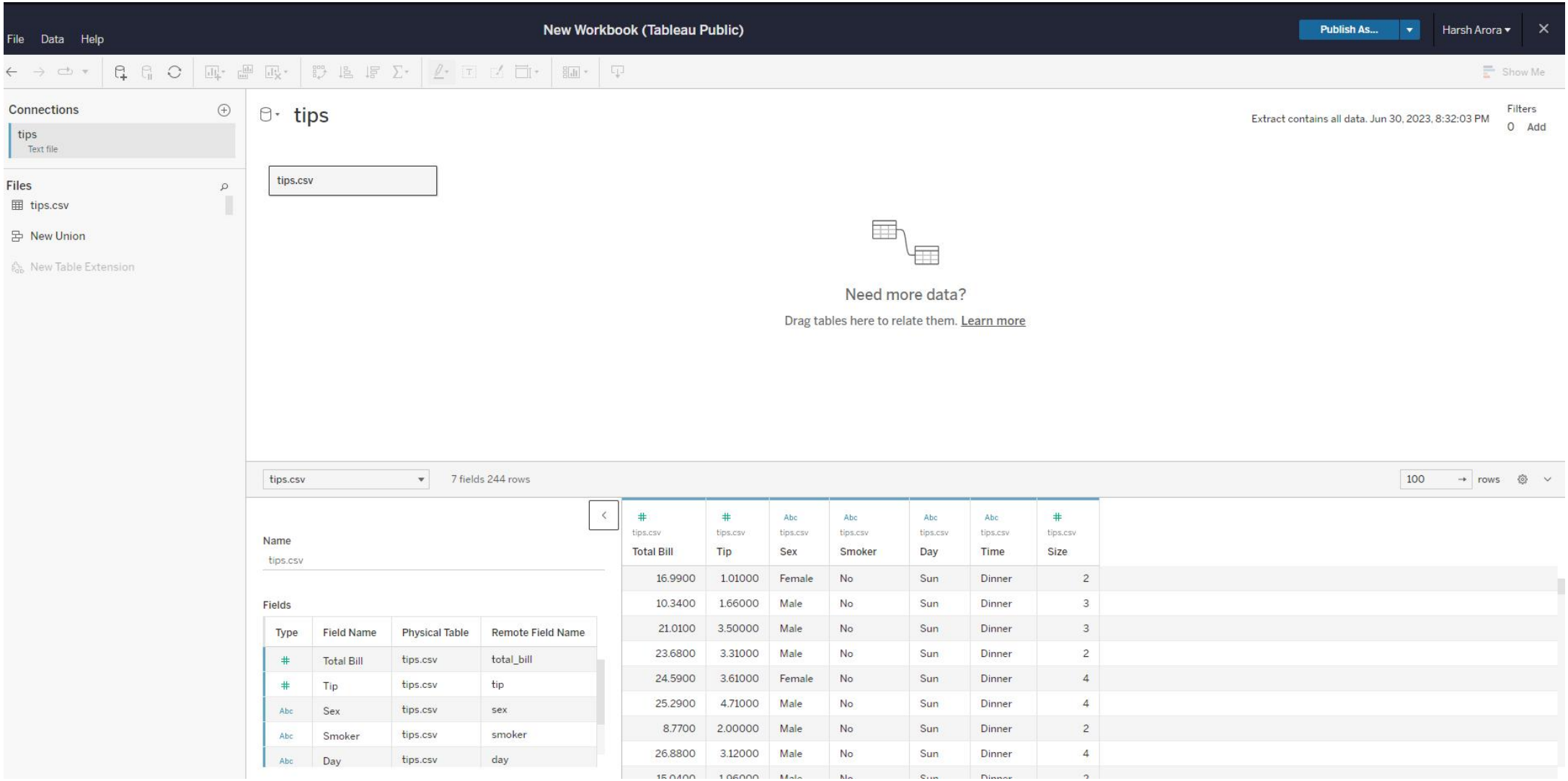

```
# iv. Display a BAR plot for tip Vs. 'total_bill' (classified according to ‘day’)
import plotly.express as px
data = px.data.tips()
fig = px.bar(data, x='tip', y='total_bill', color='day',
             facet_col='day', facet_col_wrap=2             )

fig.update_layout(title='Tip vs Total Bill',
                  xaxis_title='Tip',
                  yaxis_title='Total Bill',
                  legend_title='day')
fig.update_traces(width=0.25)

fig.show()
```

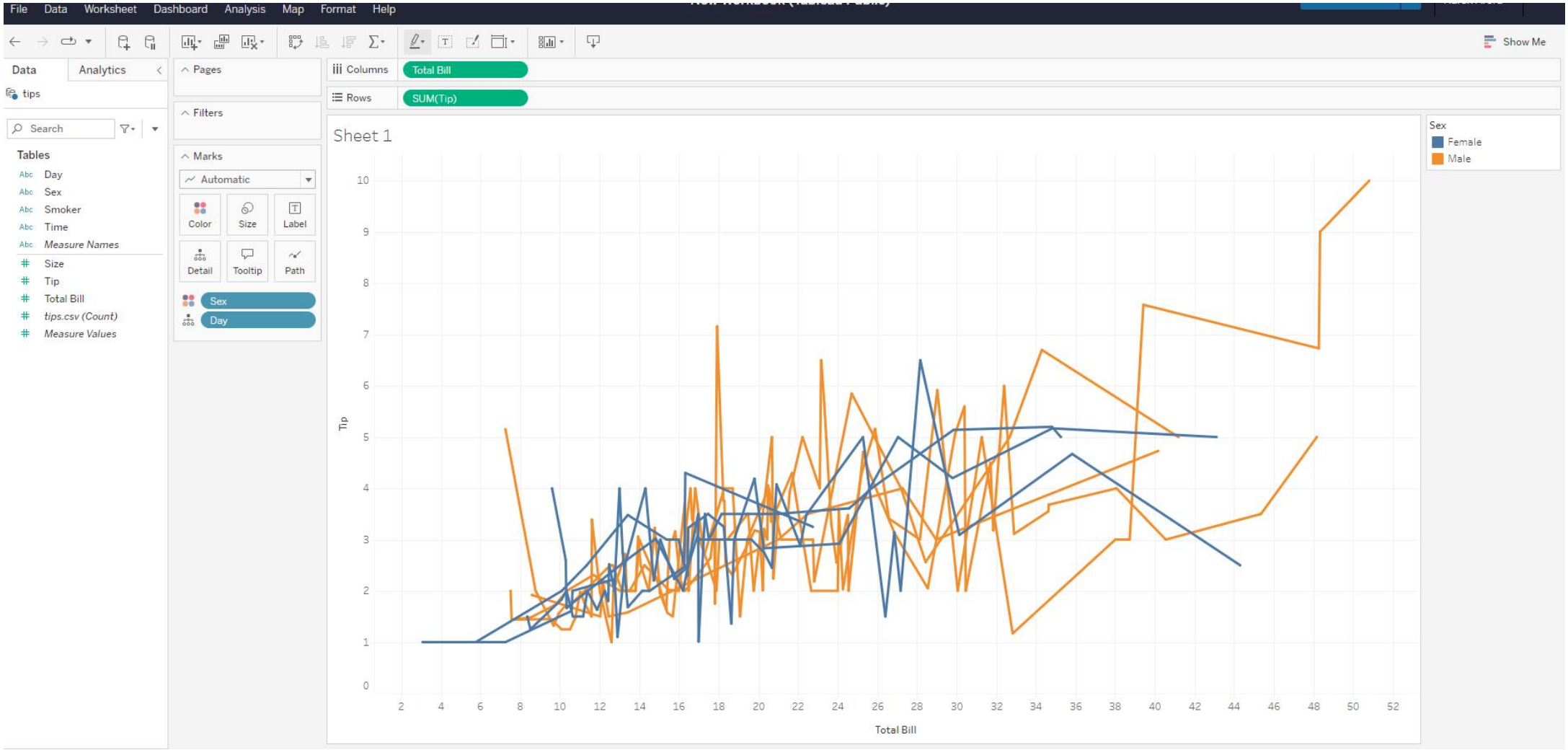




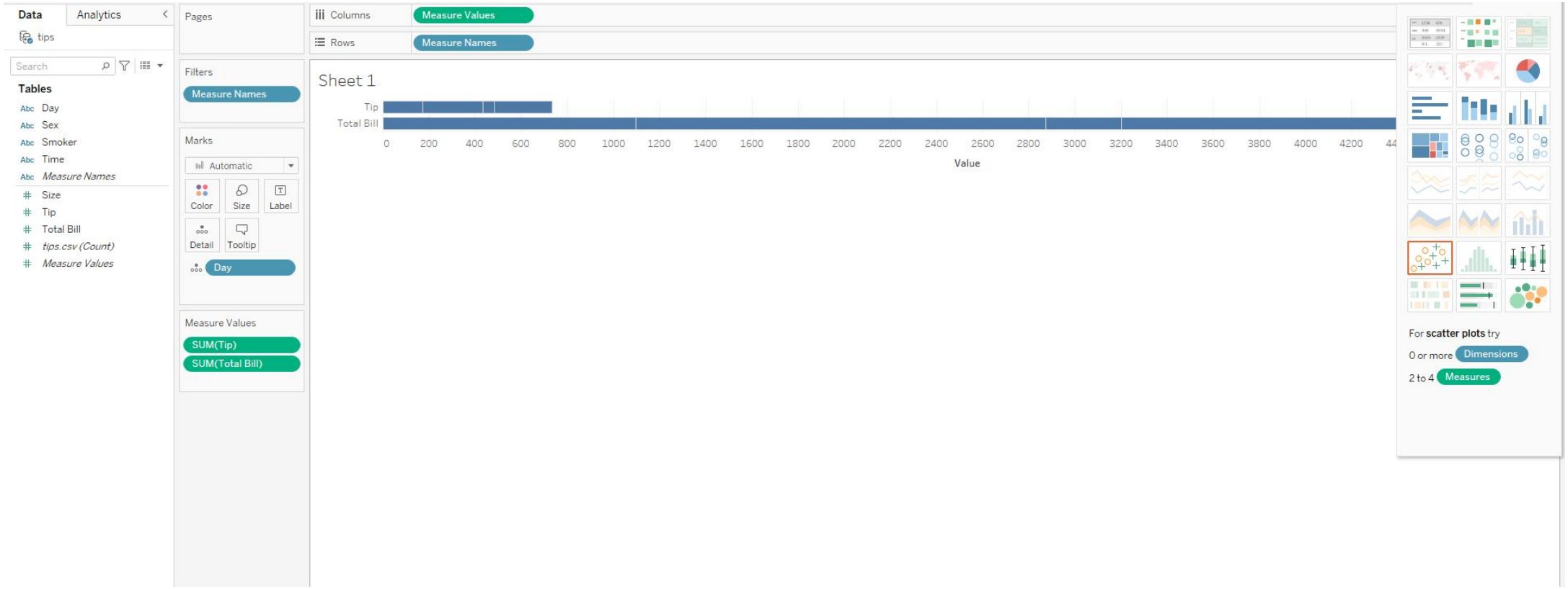

```
# Draw graph for e^x vs x
import plotly.express as px
import numpy as np
x = [m for m in range(1,20)]
y = [np.exp(m) for m in range(1,20)]
Plot1 = px.line(x=x, y=y,title='e^x Vs. X',
labels=dict(x='x',y='e^x'),
range_x=(1,20), range_y=(1,90),
height=500,width=500,markers=True)

Plot1.update_traces(textposition='top left',
marker = dict(size =10, opacity = 0.8,color='red'),
line = dict(dash = 'dashdot', width = 2,color='red'))

Plot1.update_layout(plot_bgcolor='light grey',
paper_bgcolor='pink',
xaxis_showgrid=False, yaxis_showgrid=False,
xaxis_showline=True, xaxis_linecolor='black',
yaxis_showline=True, yaxis_linecolor='black',
xaxis_mirror=True,yaxis_mirror=True,
xaxis_ticks='inside',yaxis_ticks='inside',
xaxis_tickmode='linear',xaxis_dtick=1,
yaxis_tickmode='linear',yaxis_dtick=10,
showlegend=True,
legend_title_font_color="black",
legend_title_text='e^x Vs. X',
font_family="Courier New",
font_size=15,
font_color="black",
title_font_family="Times New Roman",
title_font_color="red",
title_x=0.5)

Plot1.show()
```

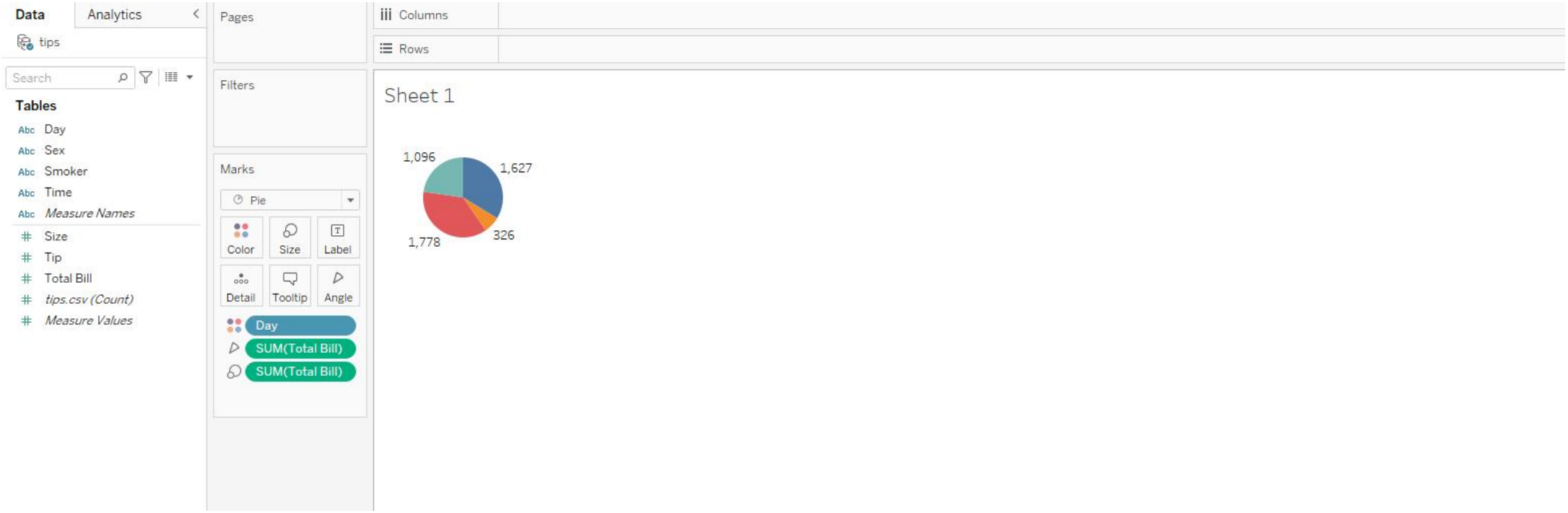



File Data Help
New Workbook (Tableau Public)
Publish As...
Harsh Arora
Show Me
Connections
tips
Text file
Files
tips.csv
New Union
New Table Extension
tips
Extract contains all data. Jun 30, 2023, 8:32:03 PM
Filters
0 Add
tips.csv
Need more data?
Drag tables here to relate them. Learn more
tips.csv
7 fields 244 rows
100 rows
Name
tips.csv
Fields
Type Field Name Physical Table Remote Field Name
Total Bill tips.csv total_bill
Tip tips.csv tip
Abc Sex tips.csv sex
Abc Smoker tips.csv smoker
Abc Day tips.csv day
tips.csv Total Bill
tips.csv Tip
tips.csv Sex
tips.csv Smoker
tips.csv Day
tips.csv Time
tips.csv Size
16.9900 1.01000 Female No Sun Dinner 2
10.3400 1.66000 Male No Sun Dinner 3
21.0100 3.50000 Male No Sun Dinner 3
23.6800 3.31000 Male No Sun Dinner 2
24.5900 3.61000 Female No Sun Dinner 4
25.2900 4.71000 Male No Sun Dinner 4
8.7700 2.00000 Male No Sun Dinner 2
26.8800 3.12000 Male No Sun Dinner 4


File Data Worksheet Dashboard Analysis Map Format Help
Show Me
Data
Analytics
tips
Search
Tables
Abc Day
Abc Sex
Abc Smoker
Abc Time
Abc Measure Names
Size
Tip
Total Bill
tips.csv (Count)
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Path
Sex
Day
Columns
Total Bill
Rows
SUM(Tip)
Sheet 1
Tip
Total Bill
Sex
Female
Male


File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data
Analytics
tips
Search
Tables
Abc Day
Abc Sex
Abc Smoker
Abc Time
Abc Measure Names
Size
Tip
Total Bill
tips.csv (Count)
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Shape
Time
Day
Columns
Size
Rows
Total Bill
Sheet 1
Total Bill
Size
For a Gantt view try
1 date
1 or more Dimensions
0 to 2 Measures
Data Source
Sheet 1


Data
Analytics
tips
Search
Tables
Abc Day
Abc Sex
Abc Smoker
Abc Time
Abc Measure Names
Size
Tip
Total Bill
tips.csv (Count)
Measure Values
Pages
Filters
Measure Names
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Day
Measure Values
SUM(Tip)
SUM(Total Bill)
Columns
Measure Values
Rows
Measure Names
Sheet 1
Tip
Total Bill
0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600 2800 3000 3200 3400 3600 3800 4000 4200
Value
For scatter plots try
0 or more Dimensions
2 to 4 Measures


Data
Analytics
tips
Search
Tables
Abc Day
Abc Sex
Abc Smoker
Abc Time
Abc Measure Names
Size
Tip
Total Bill
tips.csv (Count)
Measure Values
Pages
Filters
Marks
Pie
Color
Size
Label
Detail
Tooltip
Angle
Day
SUM(Total Bill)
SUM(Total Bill)
Columns
Rows
Sheet 1
1,096
1,627
1,778
326