PERIYAR MANIAMMAI INSTITUTE OF SCIENCE & TECHNOLOGY

PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

RECORD NOTE BOOK

COURSE NAME	:	**Foundations of Artificial Intelligence with R Programming Lab**
COURSE CODE	:	**XAI305**
STUDENT'S NAME	:	**VIKNESHRAJ D**
REGISTER NUMBER	:	**122012173030**
BRANCH	:	**Bsc Artificial Intelligence**
YEAR	:	**II**
SEMESTER	:	**III**

PERIYAR MANIAMMAI INSTITUTE OF SCIENCE & TECHNOLOGY

PERIYAR MANIAMMAI
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University)
Established Under Sec. 3 of UGC Act, 1956 • NAAC Accredited
think • innovate • transform

CERTIFICATE

Register No. | 1 | 2 | 2 | 0 | 1 | 2 | 1 | 7 | 3 | 0 | 3 | 0 |

Certified that this the Bona fide Record of the work done by **VIKNESHRAJ D**

(name of the student)

In the **Foundations of Artificial Intelligence with R Programming** Laboratory

(name of the Laboratory)

of the Department of **SOFTWARE ENGINEERING**

During the Academic Year **2023 - 2024**

Programme : **Bsc Artificial Intelligence** Year: **II** Semester: **III**

Laboratory/Course In-charge
(With date)
Name :

Head of the Department
(with Date and seal)

Submitted for the Practical Examination held on _____

Internal Examiner
Name:

External Examiner
Name:

CONTENTS

Internal Examiner
Name:

External Examiner
Name:

EXERCISE: 1 **Crash Course on Python – I & II**

Aim :

To study and practice about the basic datatypes ,Conditional Statement, Loops and Function of python using Jupyter Notebook.

Requirements:

1. Jupyter Notebook

Coding

Numeric datatype:

 # Integer Example

```
num1 = 10
print("Value:", num1, "Type:", type(num1))
```

#Float Example

```
num2 = 3.14
print("Value:", num2, "Type:", type(num2))
```

 # Complex Example

```
num3 = 2 + 3j
print("Value:", num3, "Type:", type(num3))
```

String datatype:

```
name1="HOLA FOLKS"
print("Value:", name1, "Type:", type(name1))
```

List datatype:

```
name2 = [1, 2, 3, "four", "five"]
print("Value:", name2, "Type:", type(name2))
```

Tuple datatype:

```
name3 = (1, 2, 3, "four", "five")
print("Value:", name3, "Type:", type(name3))
```

Set datatype:

```
name4 = {1, 2, 3, "four", "five"}
print("Value:", name4, "Type:", type(name4))
```

Dictionary datatype:

```
name5 = {"name": "Vikneshraj D", "age": 18, "city": "Hubli"}
print("Value:", name5, "Type:", type(name5))
```

LOOPS

#For Loops

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
   print(fruit)
```

#While Loop

```
A=input("Enter the number: ")
val = 0
i = 0

while i <= int(A):
   val += i
   i += 1

print(f"The sum is {val}")
```

FUNCTION

```
def add_numbers(x, y):
    sum_result = x + y
    return sum_result
result = add_numbers(3, 4)
print(result)
```

Conditional Statement

```
x = int(input("Enter The number"))

if x > 0 :
    print("The number is positive ")

elif x < 0 :
    print("The number is negative")

else:
    print("The number is ZERO")
```

Result:

Thus the way we declare and execute basic datatypes ,Conditional Statement, Loops and Function of python is verified Successfully

EXERCISE: 1 **Crash Course on Python – I & II**

OUTPUT:

Numeric datatype:

 # Integer Example

Value: 10 **Type:** <class 'int'>

 # Float Example

Value: 10 **Type**: <class 'float'>

 # Complex Number Example

Value: 2+3j **Type:** <class 'complex'>

String datatype:
Value: HOLA FOLKS **Type:** <class 'str'>

List datatype:
Value: [1, 2, 3, 'four', 'five'] **Type:** <class 'list'>

Tuple datatype:
Value: (1, 2, 3, 'four', 'five') **Type:** <class 'tuple'>

Set datatype:
Value: {1, 2, 3, 'four', 'five'} **Type:** <class 'set'>

Dictionary datatype:
Value: {'name': 'Vikneshraj D', 'age': 18, 'city': 'Hubli'}
Type: <class 'dict'>

#For Loops

apple
banana
cherry

#While Loop

Enter the number: 10
The sum is 55

FUNCTION

7

Conditional Statement

Enter The number 10
The number is positive

EXERCISE: 2 Implementation of Binary Search Algorithm in Python

Aim:

To Implement the Binary Search Algorithm in Python

Requirements:

1.Jupyter Notebook

Coding:

```
data =
[30,31,18,15,20,19,11,1,9,10,7,6,4,5,16,12,22,25,27,28,35,33,32,38,37,21]
data.sort()
print(data)
elem = int(input("Enter the search element:"))
def binary_search (data, elem):
   low = 0
   high = len(data) - 1

   while low <= high:
      middle = (low + high)//2
      if data[middle] == elem:
         print(f"The searching element {elem} present at index value {middle} in
dataset")



 break
      elif data[middle] > elem:
         high = middle - 1
```

```
        else :
            low = middle + 1
    if data[middle] != elem:
        print(f"The searching element {elem} is not present in dataset")
        return -1

binary_search (data, elem)
```

Result:

Thus the way we declare and execute the Binary Search Algorithm in Python is Verified Successfully

EXERCISE:2 Implementation of Binary Search Algorithm in Python

OUTPUT:

```
[1, 4, 5, 6, 7, 9, 10, 11, 12, 15, 16, 18, 19, 2
0, 21, 22, 25, 27, 28, 30, 31, 32, 33, 35, 37,38
]

Enter the search element:10

The searching element 10 present at index value
6 in dataset
```

EXERCISE:3 Implementation of Bubble Sort Algorithm in Python

Aim :

To Implement the Bubble Sort Algorithm in Python

Requirements:

1. **Jupyter Notebook**

Coding:

```
def bubbleSort(data):

  for i in range(len(data)):

    for j in range(0, len(data) - i - 1):

      if data[j] > data[j + 1]:

        temp = data[j]
        data  [j] = data [j + 1]
        data [j + 1] = temp

data = [-2, 45, 0, 11, 9, 15, -11, 21, 12]

print('Before Sorting the Array in Ascending Order:')
print(data)

bubbleSort(data)
```

```
print('After Before Sorting the Array in Ascending Order:')
print(data)
```

Result:

Thus the way we declare and execute Bubble Sort Algorithm in Python is Verified Successfully

EXERCISE: 3 Implementation of Bubble Sort Algorithm in Python

OUTPUT:

```
Before Sorting the Array in Ascending Order:
[-2, 45, 0, 11, 9, 15, -11, 21, 12]


After Before Sorting the Array in Ascending
Order:
[-11, -2, 0, 9, 11, 12, 15, 21, 45]
```

EXERCISE: 4 Implementation of Best First Search Algorithm

Aim:

To Implement the Best First Search Algorithm in Python

Requirements:

1. Jupyter Notebook

Coding

```
from queue import PriorityQueue
v = 14
graph =[[] for  i in range (v)]



def best_first_search(actual_src, target, n):
   visited = [False] * n
   pq = PriorityQueue()
   pq.put((0, actual_src))
   visited[actual_src] = True

   while pq.empty() == False:
      u = pq.get()[1]
      print(u, end=" ")
      if u == target:
         break
      for v, c in graph[u]:
         if  visited[v] == False:
            visited[v] = True
            pq.put((c, v))
```

```
print()

def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))

addedge(0, 1, 3)
addedge(0, 2, 6)
addedge(0, 3, 5)
addedge(1, 4, 9)
addedge(1, 5, 8)
addedge(2, 6, 12)
addedge(2, 7, 14)
addedge(3, 8, 7)
addedge(8, 9, 5)
addedge(8, 10, 6)
addedge(9, 11, 1)
addedge(9, 12, 10)
addedge(9, 13, 2)

source = 0
target = 14
best_first_search(source, target, v)
```

Result:

Thus the way we declare and execute Best First Search Algorithm in Python is Verified Successfully

EXERCISE: 4 Implementation of Best First Search Algorithm

OUTPUT:

```
0 1 3 2 8 9 11 13 10 5 4 12 6 7
```

EXERCISE: 5 Implementation of A* Algorithm

Aim:

To Implement the A* Algorithm by the use of python library networkx

Requirements:

1. Jupyter Notebook

Coding

Pip install **networkx**

Import **networkx** as **nx**

Import **matplotlib.pyplot** as plt

%matplotlib inline

```
Def dist(a, b):
   (x1, y1) = a
   (x2, y2) = b
   Return (( x1 – x2) ** 2 + (y1 – y2) **2) ** 0.5
G = nx.grid_graph(dim=[4, 4])
Nx.set_edge_attributes(G, {e: e[1][0] * 2 for e in G.edges()}, “cost”)

pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels = True, node_color="#00FFFF")
edge_labels = nx.get_edge_attributes(G, "cost")
nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels)
plt.show()")

path = nx.astar_path(G, (1, 0), (3, 2), heuristic = dist, weight ="cost")
```

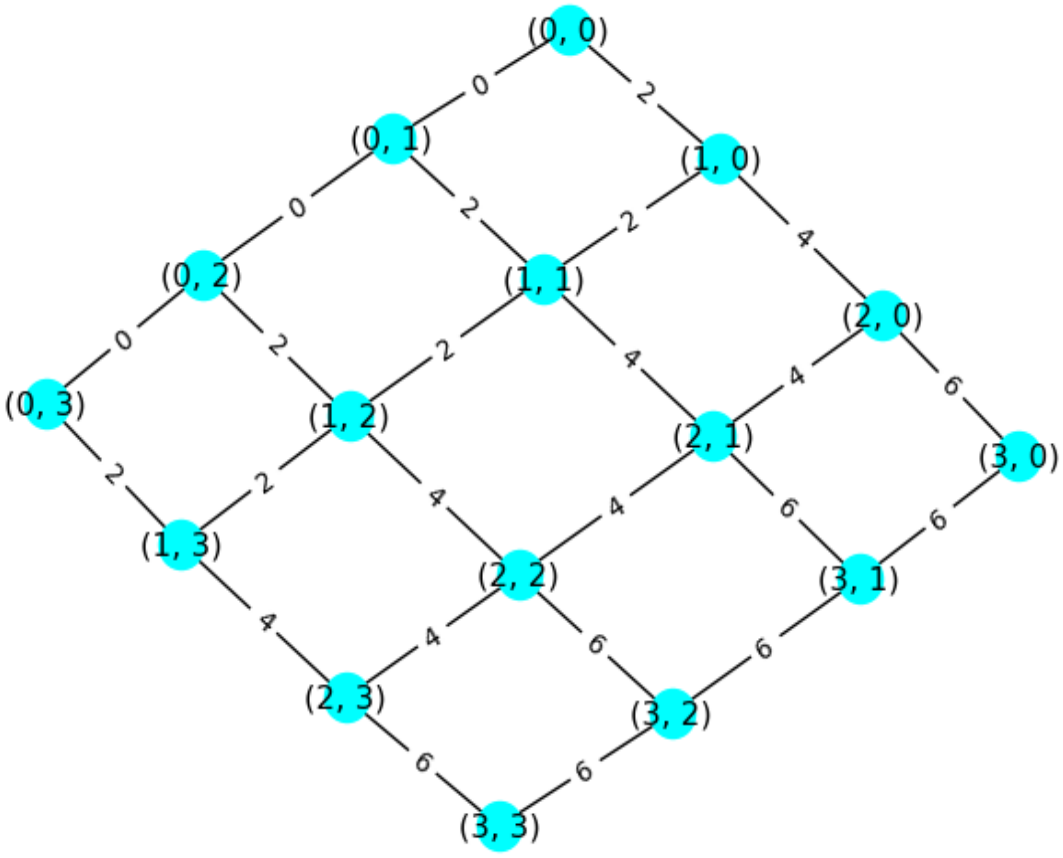
```
length = nx.astar_path_length(G, (1, 0), (3, 2), heuristic = dist, weight ="cost")
print('Path :', path)
print('Path Length', lengt)
```

Result:

Thus the way we declare and execute A* Algorithm by the use of Python library networkx is Verified Successfully

EXERCISE: 5 Implementation of A* Algorithm

OUTPUT:



EXERCISE: 6 Building Semantic Network in Python

Aim :

To Build a Semantic Network by the use of python library network

Requirements:

1. Jupyter Notebook

Coding

Import **networkx** as **nx**

Import **matplotlib.pyplot** as plt

%matplotlib notebook

Graph_Mark =nx.DiGraph(Info = “Mark’s Details”)

Graph_Mark.add_node(“Mark”,pos=(0,0))

Graph_Mark.add_node(“cat”,pos=(-2,6))

Graph_Mark.add_node(“student”,pos=(2,-5))

Graph_Mark.add_node(“animal”,pos=(1,6))

Graph_Mark.add_node(“california”,pos=(4,6))

Graph_Mark.add_node(“spinoff”,pos=(-5,-5))

Graph_Mark.add_node(“soccer”,pos=(-5,2))

Graph_Mark.add_node(“sports club”,pos=(0,-8))

Graph_Mark.add_node(“CSU”,pos=(5,-1))

Pos=nx.get_node_attributes(graph_Mark,”pos”)

graph_Mark.add_edge("Mark", "cat", weight="has a")

graph_Mark.add_edge("Mark", "student", weight="is a")

graph_Mark.add_edge("cat", "animal", weight="is a")

graph_Mark.add_edge("Mark", "soccer", weight="plays")

graph_Mark.add_edge("Mark", "spinoff", weight="is a part of")

graph_Mark.add_edge("Mark", "california", weight="lives in")

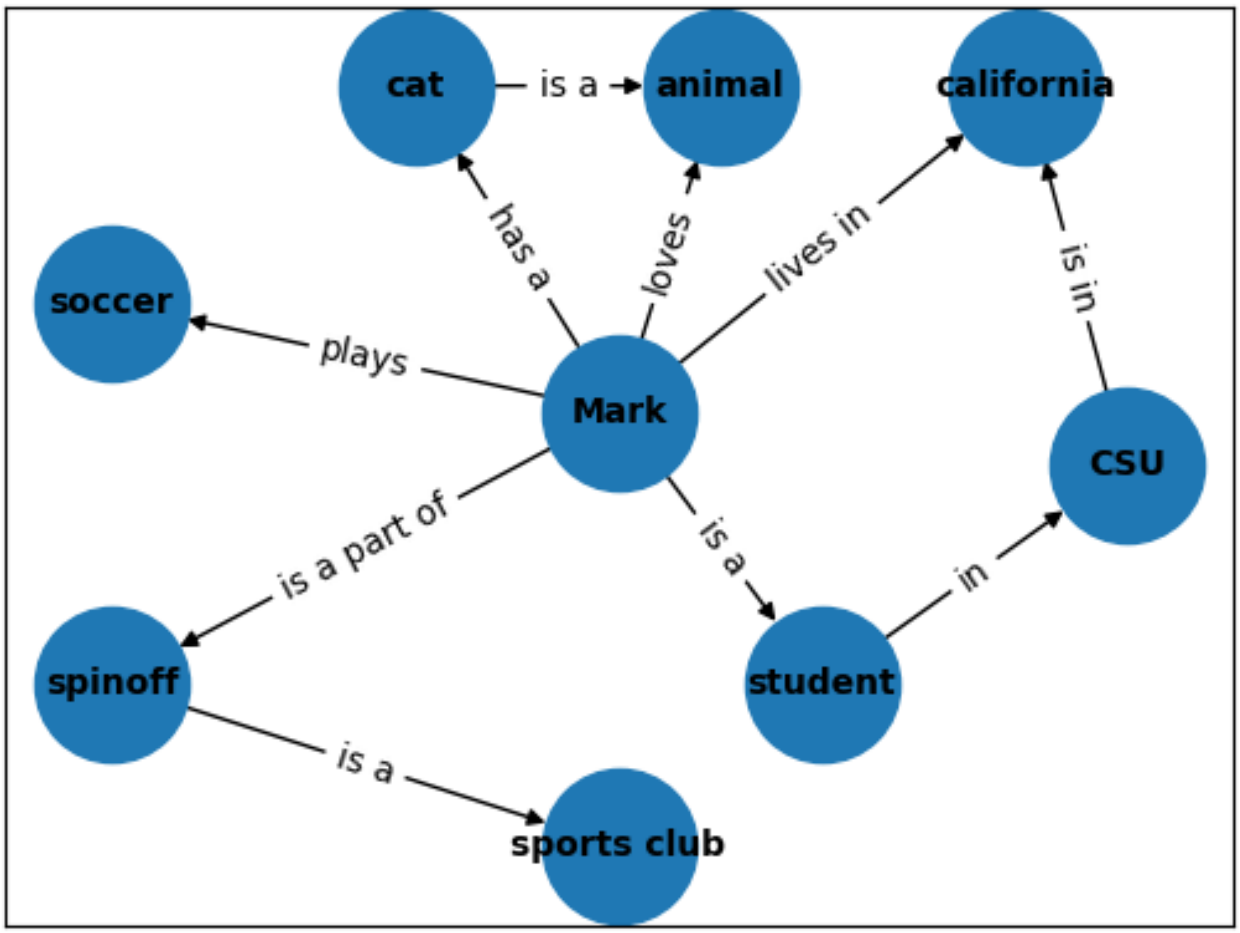
```
graph_Mark.add_edge("Mark", "animal", weight="loves")
graph_Mark.add_edge("student", "CSU", weight="in")
graph_Mark.add_edge("spinoff", "sports club", weight="is a")
graph_Mark.add_edge("CSU", "california", weight="is in")
weight =nx.get_edge_attributes(graph_Mark, "weight")

plt.figure()
nx.draw_networkx(graph_Mark,pos,font_weight='bold',node_size=2000,
font_size= 10)
nx.draw_networkx_edge_labels(graph_Mark,pos,edge_labels=weight)
```

Result:

Thus the way we declare and execute Semantic Network by the use of python library networkx is Verified Successfully

EXERCISE: 6 Building Semantic Network in Python

OUTPUT:



EXERCISE:7 Design and Deployment of an Expert System

Aim :

To Design and Deployment of an Expert System by the use of library experta

Requirements:

1. Jupyter Notebook

Coding:

pip install **experta**

```
from experta import *
class meds(KnowledgeEngine):
    @DefFacts()
    def _initial_action(self):
        yield Fact(action ='load')

    # Starting Questions
    @Rule(Fact(action = 'load'), NOT(Fact(fulltime = W())))
    def start_quest(self):
        print("Welcome to the Medical Expert System. ")
        self.declare(Fact(intro = input("Please enter your name: ")))
        self.declare(Fact(fulltime = input("Do you want to enter the Medical
Expert System? ")))

    # Not interested in entering
    @Rule(Fact(action = 'load'), (Fact(fulltime = 'no')))
    def exiting(self):
        print("Thank you!")
```

```
    # Rule 1: Checking Covid Symptom #1 - Fever
    @Rule(Fact(action = 'load'), (Fact(fulltime = 'yes')))
    def fever_check(self):
        self.declare(Fact(Fever = input("Do you have fever for the last few days? ")))

    # Rule 2: Checking Covid Symptom #2 - Dry Cough
    @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), NOT(Fact(Fever = 'not sure'))))
    def cough_check(self):
        self.declare(Fact(Cough = input("Do you have dry cough for the last few days? ")))


 # Rule 3: Checking Covid Symptom #3 - Tiredness
    @Rule(Fact(action='load'), AND(Fact(fulltime = 'yes'), NOT(Fact(Fever = 'not sure')), NOT(Fact(Cough = 'not sure'))))
    def tired_check(self):
        self.declare(Fact(Tired = input("Have you been feeling tired? ")))

    # Diagnosis uptil Rule 3
    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), AND(Fact(Fever = 'yes'), Fact(Cough = 'no'), Fact(Tired = 'no'))))
    def accept_1(self):
        print("You have fever, please take rest and have Paracetamol")
```

```
    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), AND(Fact(Fever =
'no'), Fact(Cough = 'yes'), Fact(Tired = 'no'))))
    def accept_2(self):
        print("You just have dry cough. Please gargle, steam and have lots of hot
water.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(Fever = 'yes'),
Fact(Cough = 'yes'), Fact(Tired = 'yes')))
    def accept_3(self):
        print("You are showing symptoms of COVID-19. Please get yourself tested
and stay quarentined.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(Fever = 'no'),
Fact(Cough = 'yes'), Fact(Tired = 'yes')))
    def accept_4(self):
        print("Please visit the doctor as you may have a throat infection.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(Fever = 'yes'),
Fact(Cough = 'no'), Fact(Tired = 'yes')))
    def accept_5(self):
        print("You may be having a viral infection. Take ample rest. If it presists
please visit a doctor.")
```

 # Enter advance expert system.

```
    @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), OR(Fact(Fever =
'yes'), Fact(Fever = 'no')), OR(Fact(Cough = 'yes'), Fact(Cough = 'no')),
  OR(Fact(Tired = 'yes'), Fact(Tired = 'no'))))
    def adv_expt(self):
        print("You have completed the simple medical expert system.")
        self.declare(Fact(dep_dive = input("Do you want to dive deeper into the
expert system? ")))
```

 # Deciding.

```
    @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), Fact(dep_dive =
'no')))
    def div_reject(self):
        print("Thank you for using our expert system.")
```

 # Rule 4: Checking Covid Symptom #4 - Shortness of breath

```
    @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), Fact(dep_dive =
'yes')))
    def breath(self):
        self.declare(Fact(breathing = input("Have you been experiencing
shortness of breath? ")))
```

 # Rule 5: Checking Covid Symptom #5 - Chest Pain

```
    @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), Fact(dep_dive =
'yes'),OR(Fact(breathing = 'yes'), Fact(breathing = 'no'))))
    def chest_pain(self):
```

```
    self.declare(Fact(chest = input("Have you been experiencing acute chest pain or pressure? ")))

  # Rule 6: Checking Covid Symptom #6 - Loss of speech or movement
  @Rule(Fact(action = 'load'), AND(Fact(fulltime = 'yes'), Fact(dep_dive = 'yes'), OR(Fact(breathing = 'yes'), Fact(breathing = 'no')),
  OR(Fact(chest = 'yes'), Fact(chest = 'no'))))
  def speech_loss(self):
    self.declare(Fact(loss = input("Have you been experiencing any loss of speech or movement? ")))

  #Diagnosis 4-6
  @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'), Fact(breathing = 'yes'), Fact(loss = 'no'), Fact(chest = 'no')))
  def accept_6(self):
    print("You seem to be having shortness of breath. Even if you are not COVID positve, this is serious.")
    print("Go to the doctor immediately.")

  @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'), Fact(breathing = 'no'), Fact(loss = 'yes'), Fact(chest = 'no')))
  def accept_7(self):
    print("You seem to be having either loss of speech or movement. Even if you are not COVID positve, this is serious.")
    print("Go to the doctor immediately.")
```

```
    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'),
Fact(breathing = 'no'), Fact(loss = 'no'), Fact(chest = 'yes')))
    def accept_8(self):
        print("You seem to be having chest pain. Even if you are not COVID
positve, this is serious.")
        print("Go to the doctor immediately.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'),
Fact(breathing = 'yes'), Fact(loss = 'no'), Fact(chest = 'yes')))
    def accept_9(self):
        print("You seem to be having chest pain and shortness of breath. Even if
you are not COVID positve, this is serious.")
        print("Go to the doctor immediately.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'),
Fact(breathing = 'no'), Fact(loss = 'yes'), Fact(chest = 'yes')))
    def accept_10(self):
        print("You seem to be having chest pain and loss of speech or motion.
Even if you are not COVID positve, this is serious.")
        print("Go to the doctor immediately.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'),
Fact(breathing = 'yes'), Fact(loss = 'yes'), Fact(chest = 'no')))
    def accept_11(self):
        print("You seem to be having shortness of breath and loss of speech or
movement. Even if you are not COVID positve, this is serious.")
```

```
        print("Go to the doctor immediately.")

    @Rule(Fact(action='load'), AND(Fact(fulltime='yes'), Fact(dep_dive = 'yes'),
Fact(breathing = 'yes'), Fact(loss = 'yes'), Fact(chest = 'yes')))
    def accept_12(self):
        print("You seem to be having chest pain, shortness of breathing and loss
of speech or movement Even if you are not COVID positve, this is serious.")
        print("Go to the doctor immediately.")



Engine = meds()
Engine.reset()
Engine.run()
```

Result:

Thus the way we declare and execute the Expert System by the use of library experta in Python is verified Successfully.

EXERCISE: 7 Design and Deployment of an Expert System

Output:

Welcome to the Medical Expert system.

please enter your name : **Vikneshraj D**

Do you want to enter the Medical expert system? **yes**

Do you have fever for the last few days? **yes**

Do you have dry cough for the last few days? **yes**

Have you been feeling tired? **yes**

You are showing symptoms of COVID-19.Please get yourself tested and stay quarantined.

You have completed the simple medical expert system.

Do you want to dive deeper into the expert system? **yes**

Have you been experiencing shortness of breath? **yes**

Have you been experiencing acute chest pain or pressure? **yes**

Have you been experiencing any loss of speech or movement? **yes**

You seem to be having chest pain and shortness of breath and loss of speech or movement.

Even if you are not COVID positive, this is serious. Go to the doctor immediately.

EXERCISE: 8 Building Bayesian Networks in Python

Aim :

To Build a Bayesian Networks by the use of python library protopunica

Requirements:

1. Jupyter Notebook

Coding

Pip install **protopunica**

From **protopunica** import *

```
smoking = Node(DiscreteDistribution({"High smoking":0.7,"Low smoking":0.3}),name="smoking")

asbes_consum =Node(DiscreteDistribution({"High Cons":0.3,"Low Cons":0.7}),name="asbes_consum")

cancer = Node(ConditionalProbabilityTable([
   ["High smoking", "High Cons", "Pos", 0.4],
   ["High smoking", "High Cons", "Neg", 0.6],
   ["High smoking", "Low Cons", "Pos", 0.3],
   ["High smoking", "Low Cons", "Neg", 0.7],
   ["Low smoking", "Low Cons", "Pos", 0.1],
   ["Low smoking", "Low Cons", "Neg", 0.9],
   ["Low smoking", "High Cons", "Pos", 0.02],
   ["Low smoking", "High Cons", "Neg", 0.98],],
   [smoking.distribution, asbes_consum.distribution]), name="cancer")
```

```
scan = Node(ConditionalProbabilityTable([
   ["Pos","scan_pos",0.8],
   ["Pos","scan_neg",0.2],
   ["Neg","scan_pos",0.1],
   ["Neg","scan_neg",0.9]],[cancer.distribution]),name="scan")

Blood_vomiting = Node(ConditionalProbabilityTable([
   ["Pos","B.V_pos",0.7],
   ["Pos","B.V_neg",0.3],
   ["Neg","B.V_pos",0.2],
   ["Neg","B.V_neg",0.8]],[cancer.distribution]),name="Blood_vomiting ")

model=BayesianNetwork()

model.add_states(smoking,asbes_consum,cancer,scan,Blood_vomiting)

model.add_edge(smoking,cancer)
model.add_edge(asbes_consum,cancer)
model.add_edge(cancer,scan)
model.add_edge(cancer,Blood_vomiting)

model.bake()

model
probability=model.probability([["Low smoking","Low
Cons","Pos","scan_pos","B.V_pos"]])
```

probability

probability=model.probability([["High smoking","High Cons","Pos","scan_pos","B.V_pos"]])
probability

>>> print(model.predict([["Low smoking", "Low Cons","Neg","scan_pos",None]]))

predictions= model.predict_proba({"Blood_vomiting": "B.V_pos"})
predictions

predictions= model.predict_proba({"scan": "scan_pos"})
predictions

Result:

Thus the way we declare and execute Bayesian Networks by the use of python library protopunica is Verified Successfully

EXERCISE: 8 Building Bayesian Networks in Python

OUTPUT:

```
In [13]: probability=model.probability([["Low smoking","Low Cons","Pos","scan_pos","B.V_pos"]])

In [14]: probability

Out[14]: 0.011759999999999998

In [22]: probability=model.probability([["High smoking","High Cons","Pos","scan_pos","B.V_pos"]])

In [23]: probability

Out[23]: 0.04704
```

EXERCISE:9 Building Markov Chain Model

Aim :

To Build a Markov Chain Model by the use of python library protopunica and numpy

Requirements:

1. Jupyter Notebook

Coding

```
from protopunica import *
import numpy as np

start = DiscreteDistribution({"PIZZA":1,"Veg":0})
Transitions = ConditionalProbabilityTable([
   ["PIZZA", "PIZZA", 0.75],
   ["PIZZA", "VEG", 0.25],
   ["VEG", "VEG",  0.6],
   ["VEG", "PIZZA", 0.4],], [start])

Model=MarkovChain([start,Transitions])

Random_samples=Model.sample(2)

print(Random_samples)
log_probability = Model.log_probability(Random_samples)

Probability_of_Occurance= np.exp(log_probability)
```

Probability_of_Occurance

log_probability_Food_Sequence **=**
Model**.**log_probability(["PIZZA","PIZZA","PIZZA"])

Probability_of_Food **=** np**.**exp(log_probability_Food_Sequence)

print (Probability_of_Food)

Result:

Thus the way we declare and execute Markov Chain Model by the use of python library protopunica and numpy is Verified Successfully

EXERCISE: 9 Building Markov Chain Model

OUTPUT:

Probability of Occurance

```
0.25
```

Probability of Food

```
0.5625
```

EXERCISE:10 Building a Hidden Markov Model in Python

Aim :

To Build a Hidden Markov Model by the use of python library protopunica and pandas

Requirements:

2. Jupyter Notebook

Coding:

Problem Statement

In our problem we have our poor prisoner who is stuck in a prison...as the story says...this prison is completely isolated from the rest of the world...and one cannot even see the sky...In such scenario our prisoner has been in the prison for 2 years now....The funny thing is that there is an incharge who takes care of all the prisoners....So the incharge wears a hat if its sunny and wears a rain coat if its rainy...since the prisoner has no access to open spaces. He can only deduce the weather by checking what the incharge came in everyday...is it a raincoat or a Hat.....

1 Import Libraries

```
from protopunica import *
import pandas as pd
```

2 Observation Model

```
sunny = DiscreteDistribution({"raincoat": 0.1,"Hat" : 0.9})
rainy = DiscreteDistribution({"raincoat": 0.7,"Hat" : 0.3})
```

3 Define States

states= [sunny,rainy]

4 Transition Model

transition_model = numpy.array([[0.7,0.3],[0.4,0.6]])

5 Initial State

Initial_state=numpy.array([0.3,0.7])

6 Build the Model

model=HiddenMarkovModel.from_matrix(transition_model,states,
Initial_state,state_names=["sunny","rainy"])

7 Bake the Model

model.bake()

8 Observation

observations=["Hat","Hat","raincoat","raincoat","Hat","Hat","raincoat","Hat",
"Hat"]

9 Predict the States for Given Observation

predictions = model.predict(observations)
predictions

10 Model

Predicted states with respect to Observation

```
for prediction in predictions:
   print(model.states[prediction].name)

predicted_probabilities=model.predict_proba(observations)
Most_likely_Weather=pd.DataFrame(predicted_probabilities,columns=["Rainy
","Sunny"])
Most_likely_Weather
```

Result:

Thus the way we declare and execute Hidden Markov Model by the use of python library protopunica and pandas is Verified Successfully

EXERCISE:10 **Building a Hidden Markov Model in Python**

Output:

```
[1, 1, 0, 0, 1, 1, 0, 1, 1]
```

Output:

	Rainy	Sunny
0	0.399892	0.600108
1	0.306306	0.693694
2	0.859424	0.140576
3	0.855572	0.144428
4	0.266125	0.733875
5	0.248499	0.751501
6	0.730427	0.269573
7	0.225295	0.774705
8	0.171936	0.828064

EXERCISE:11 Fundamentals of R Language

Aim :

To study and practice about the basic datatypes ,Conditional Statement, Loops , Data Handling and Functions, of R using R Studio.

Requirements:

1. R Studio

Coding:

Data Types

```
# Numeric
num = 44.5
class(num)

# Integer
num1 = "100L"
class(num1)

# Complex
cmplx = 10i
class(cmplx)

# Character
Name = “LEOMESSI”
class(Name)

# Logical
num2 = TRUE
class(num2)
```

Conditional Statements

```
# If Statement
x <- 5
if(x > 0)
{
  print("Positive number")

  }
```

 # If Else Statement

```
x <- -5
if(x > 0)
{
  print("Positive number")
} else
{
  print("Negative number")
}
```

 # Nested If-Else Statement

```
x <- 0
if(x < 0)
{
  print("Negative number")
} else if(x > 0)
{
  print("Positive number")
} else
  print("Zero")
```

Loops

 # For Loop

```
x <- c(-10,5,10,44)
for (i in x)
{
  print(i)
}
```

 # Nested for loop

```
for (i in 1:3)
{
  for (j in 1:i)
  {
    print(i * j)
  }
}
```

 # While Loop

```
i = 0
count = 0

while (count <= 5)
{
  print(i * i)
  i = i + 1
  count = count + 1
}
```

Functions

 # Function for Squaring a Number

```
number = function(x) {
  return(x^2)
}
result = number(5)
print(result)
```

Handling Data in R

 # Creating data

```
employee <- c('VINCENT', 'VIKNESHRAJD', 'MESSI')
salary <- c(20000, 23000, 28000)
ID_num = c(30, 19, 10)
```

 # Creating data frame

```
employee.data = data.frame(employee, salary, ID_num)
```

 # Viewing data frame

```
View(employee.data)
```

 # Basic info from data frame

```
print(ncol(employee.data))
print(nrow(employee.data))
```

 # Slicing Column

```
print(employee.data$employee)
```

```
# Slicing Rows
print(employee.data[2:3, ])

# Modifying data
employee.data[2, "ID_num"] <- 15
employee.data

# Saving data frame
write.csv(employee.data, "Employee Details.csv")

# Reading data frame
read.csv("Employee Details.csv")
```

Result:

Thus the way we declare and execute basic datatypes ,Conditional Statement, Loops , Data Handling and Functions of R using R Studio is verified Successfully

EXERCISE:11 Fundamentals of R Language

OUTPUT:

 # Numeric

"numeric"

 # Integer

"integer"

 # Complex

"complex"

 # Character

"character"

 # Logical

"logical"

OUTPUT:

 # If Statement
"Positive number"

 # If Else Statement
"Negative number"

 # Nested If-Else Statement
“Zero”

OUTPUT:

 # For Loop

-10

5

10

44

 # Nested for loop

1
2
4
3
6
9

 # While Loop

0

1

4

9

16

25

OUTPUT:

 # Function for Squaring a Number
25

OUTPUT:

Handling Data in R

	employee	salary	ID_num
1	VINCENT	20000	30
2	VIKNESHRAJD	23000	19
3	MESSI	28000	10

EXERCISE:12 Web Scraping in R

Aim :

To Scrape and Specific types of information from website using R by the use of R library rvest, dplyr,robotstxt,stringr .

Requirements:

1. R Studio

Coding:

```
# IMPORTING LIBRARIES
library(rvest)
library(dplyr)
library(robotstxt)
library(stringr)

# SCRAPPING WEBSITE
link <- "https://editorial.rottentomatoes.com/guide/10-best-reviewed-pro-football-movies/"

# Allowability
path = paths_allowed(link)

# HTML ELEMENTS FROM WEBSITE
web <- read_html(link)
View(web)


name <- web %>% html_nodes(".article_movie_title a") %>% html_text()
View(name)

year <- web %>% html_nodes(".start-year") %>% html_text()
View(year)

rating <- web %>% html_nodes(".tMeterScore") %>% html_text()
View(rating)
```

```
rank <- web %>% html_nodes(".countdown-index") %>% html_text()
View(rank)

Director <- web %>% html_nodes(".director .descriptor+ a") %>% html_text()
View(Director)


# CREATING DATAFRAME
movies.ratings <- data.frame(name, year, rating, rank, Director)

# VIEW DATAFRAME
View(movies.ratings)

# SAVING DATA
write.csv (movies.ratings, "My movie data.csv")
```

Result:

Thus the way we scrape and execute the web scraping by the use of R Studio is verified Successfully

EXERCISE:12 Web Scraping in R

Output:

	name	year	rating	rank	Director
1	Concussion	(2015)	58%	#10	Peter Landesman
2	Black Sunday	(1977)	74%	#9	John Frankenheimer
3	Invincible	(2006)	72%	#8	Ericson Core
4	Semi-Tough	(1977)	83%	#7	Michael Ritchie
5	The Longest Yard	(1974)	76%	#6	Robert Aldrich
6	Jerry Maguire	(1996)	84%	#5	Cameron Crowe
7	Big Fan	(2009)	85%	#4	Robert D. Siegel
8	North Dallas Forty	(1979)	84%	#3	Ted Kotcheff
9	Heaven Can Wait	(1978)	85%	#2	Warren Beatty
10	Brian's Song	(1971)	92%	#1	Buzz Kulik

EXERCISE:13 Data Visualization in R

Aim :

To Create and Manipulate various Data Visualization like Bar Plot, Line Plot, Scatter Plot, Histogram for Data set using R Studio.

Requirements:

1. R Studio
2. Data Set

Coding:

```
library(ggplot2)
library(dplyr)

# Importing Dataset
expenses <- read.csv("dailyexpenses.csv")

# Viewing Dataset
View(expenses)
str(expenses)

# Date type Conversion
expenses$Date <- as.Date(expenses$Date, "%d/%m/%y")
View(expenses)
str(expenses)

# Data frame for Comparing Expenses
tot <- expenses[, 2:9]
total <- colSums(tot)
detail <- c("food", "groceries", "medical", "fuel", "toll",
          "phone", "electricity", "miscellaneous")
tot_each <- data.frame(detail, total)
View(tot_each)

# Bar plot of expenses
ggplot(data = tot_each, aes(x = detail, y = total))+ geom_bar(stat = "identity",
fill = "green") +
  labs(x = "Items", y = "Expenses", title = "Comparing Expenses")
```

```
# Dataframe for Daily Total Expenses
cm <- expenses[, 2:9]
dt <- expenses[, 1]
rs <- rowSums(cm)
daily_tot <- data.frame(dt, rs)
View(daily_tot)

# Line Plot for daily total expenses
ggplot(data = daily_tot, aes(x = dt, y = rs)) +
  geom_line(color = "red", size = 2) +
  labs(x = "Date", y = "Expenses", title = "Total Expenses")

# Dataframe of Daily Medical Expenses
md <- expenses[, 3]
de <- expenses[, 1]
daily_med <- data.frame(de, md)
View(daily_med)

# Scatter Plot of Daily Medical Expense
ggplot(data = daily_med, aes(x = de, y = md)) + geom_point() +
  labs(x = "Expenses", y = "Medical", title = "Medical Expenses")

# Histogram of Medical Expenses
ggplot(expenses, aes(x = Medical)) +
  geom_histogram(binwidth = 5, color = "black", fill = "blue")
```

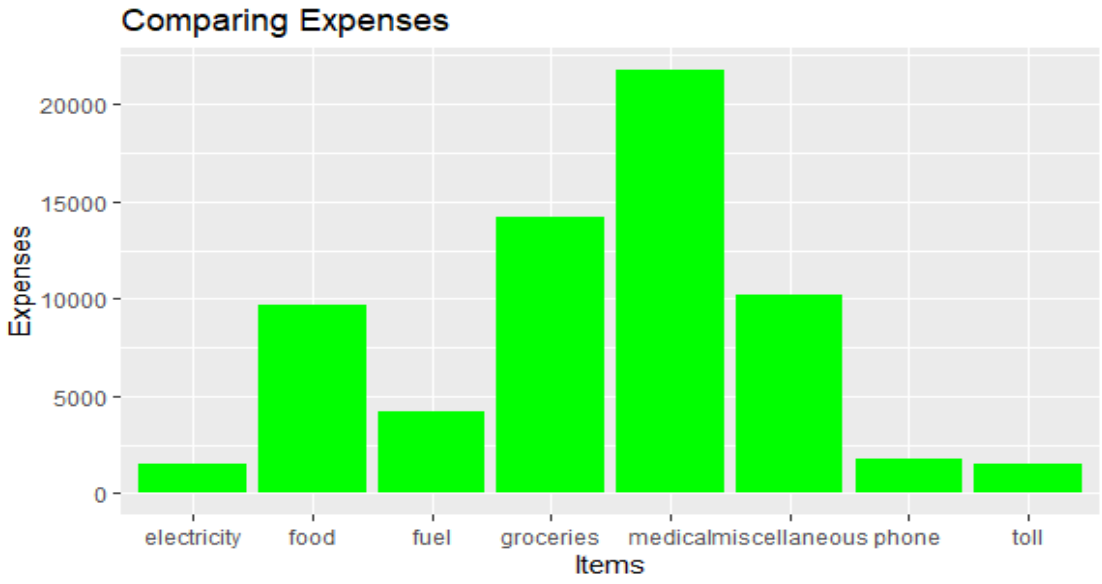
Result:

Thus the way we Create and Manipulate various Data Visualization like Bar Plot, Line Plot, Scatter Plot, Histogram for Data set using R Studio is verified Successfully

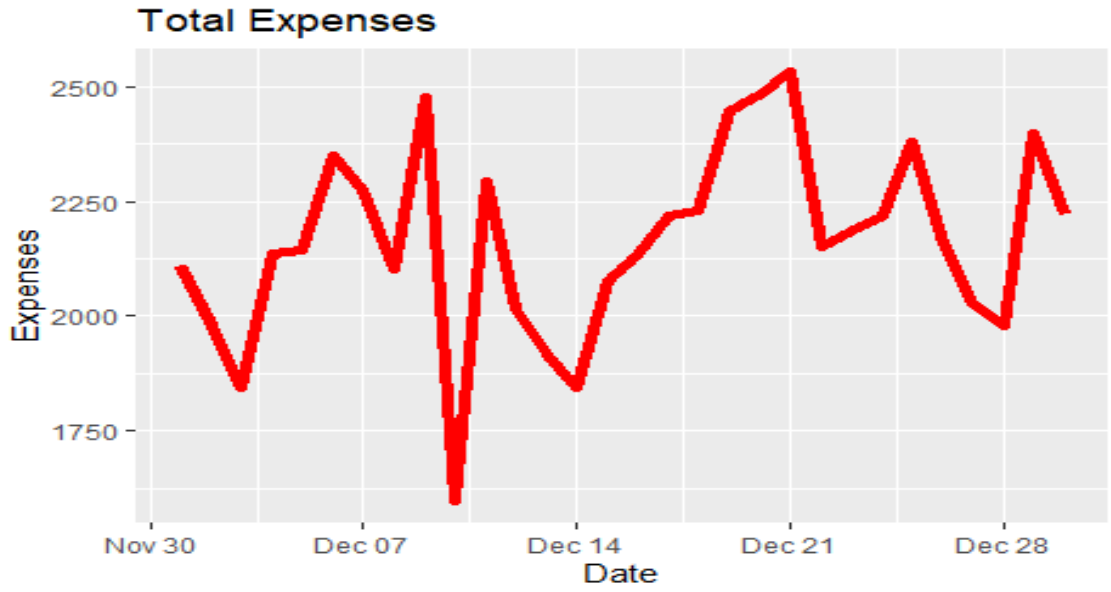
EXERCISE:13 Data Visualization in R

OUTPUT:

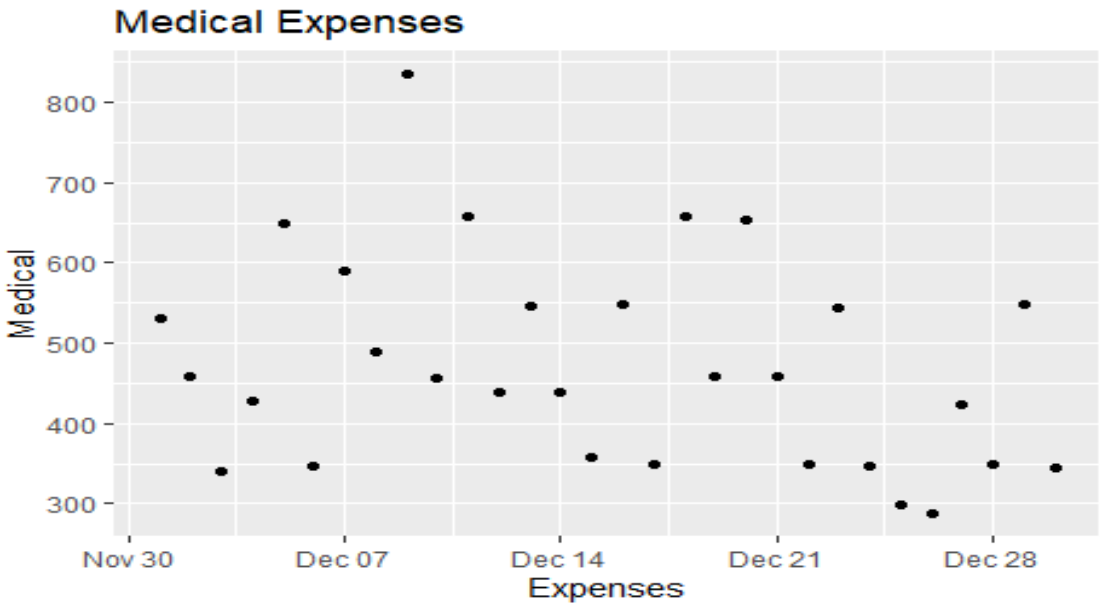
 # Bar plot of expenses



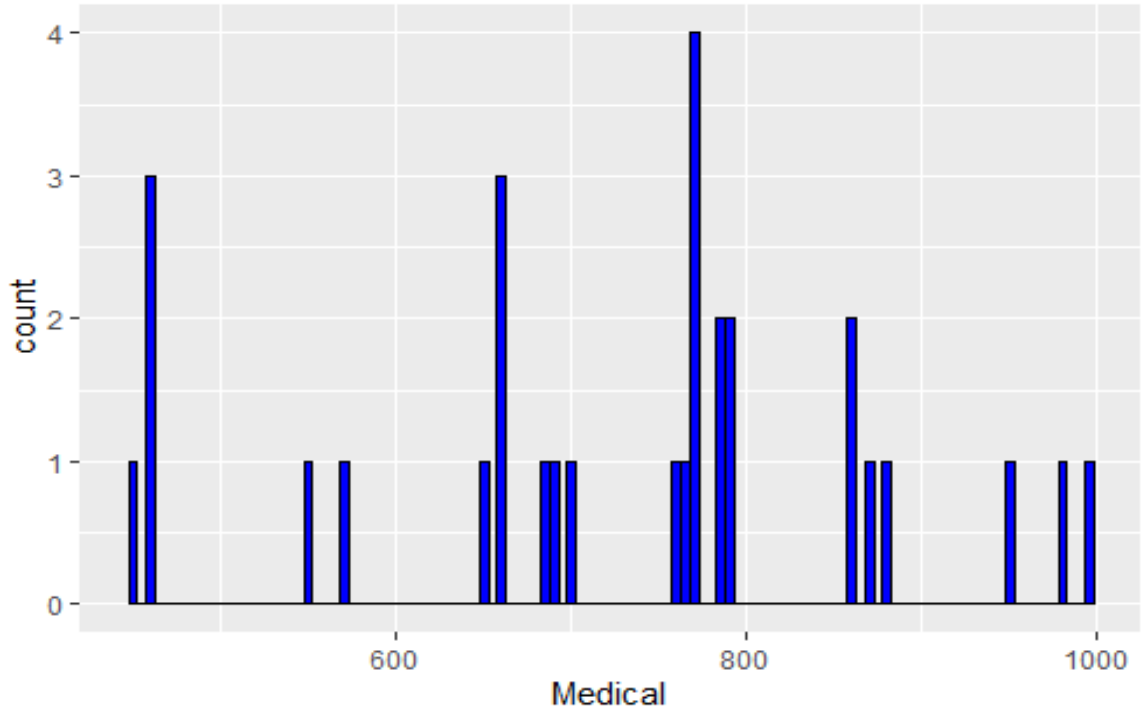
 # Line Plot for daily total expenses



 # Scatter Plot of Daily Medical Expense



 # Histogram of Medical Expenses



EXERCISE:14 Build Data dashboard using Shinny Dashboard

Aim :

To Build Data dashboard using Shinny Dashboard using R Studio

Requirements:

1. R Studio

Coding:

```
library(shinydashboard)
library(shiny)

ui<-dashboardPage(
  dashboardHeader(title="Basic Dashboard"),
  dashboardSidebar(
    sidebarMenu(
      menuItem("Dashboard", tabName = "dashboard", icon =
icon("dashboard")),
      menuItem("Widgets", tabName = "widgets", icon = icon("th"))
    )
  ),
  dashboardBody(
    tabItems(
      tabItem(tabName = "dashboard",
              fluidRow(
                box(plotOutput("plot1",height=400)),
                box(title="Controls",
                    sliderInput("slider","Number of Observations",1,1000,500)
                )
              )),
      tabItem(tabName = "widgets",
              h2("Widgets tab page under construction"))
    )
  )
)
server <- function(input, output) {
```

```
  set.seed(122)
  histdata <- rnorm(1000)
  output$plot1 <- renderPlot({
    data <- histdata[seq_len(input$slider)]
    hist(data)
  })
}
shinyApp(ui,server)
```

Result:

Thus the way we create and execute Data dashboard using Shinny Dashboard in R Studio is verified Successfully

EXERCISE:14 Build Data dashboard using Shinny Dashboard

OUTPUT: