**Sinhgad College of Engineering**

**Department Of Computer Engineering**

Name Of Student:

Roll No.: Exam No.:

Class: Div: Batch:

Name Of Laboratory:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **Start Date** | **Completion Date** |
| 1 | Implementation of Fuzzy Operations: Union, Intersection, Complement and Difference operations on fuzzy sets. |  |  |
| 2 | Implementation of Fuzzy Relation: by Cartesian product of any two fuzzy sets and perform max-min composition on fuzzy relations. |  |  |
| 3 | Perform Functions of Logic Gates using McCulloh-Pitts Neural Network (model) |  |  |
| 4 | Study and Implement optimization problem using Partial Swarm Optimization |  |  |
| 5 | Mini Project -Implementation of Simple Genetic Application |  |  |

Soft Computing and Optimization Algorithm

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Name of Experiment** | **Start Date** | **Completion Date** |
| 1 | Installation and configuration of own Cloud. |  |  |
| 2 | Implementation of Virtualization in Cloud Computing to Learn Virtualization Basics, Benefits of Virtualization in Cloud using Open Source Operating System |  |  |
| 3 | Case study on Amazon EC2 to learn about Amazon EC2, Amazon Elastic Compute Cloud is a central part of Amazon.com's cloud computing platform, Amazon Web Services. How EC2 allows users torrent virtual computers on which to run their own computer applications. |  |  |
| 4 | Case study on Microsoft azure to learn about Microsoft Azure is a cloud computing platform and infrastructure, created by Microsoft, for building, deploying and managing applications and services through a global network of Microsoft-managed data centre’s. How it works, different services provided by it. |  |  |
| 5 | Write a Program to Create, Manage and groups User accounts in own Cloud by Installing Administrative Features. |  |  |
| 6 | Assignment to install and configure Google App Engine. |  |  |
| 7 | Mini Project - Real time implementation for data streaming using google cloud pubsub and bigquery |  |  |

Cloud Computing

Assignment 1

|  |
| --- |
| Code of Program(write our name in comments) –  #fuzzy sets operations '''  Implement Union, Intersection, Complement and Difference operations on fuzzy sets.  Also create fuzzy relation by Cartesian product of any two fuzzy sets and perform max- min composition on any two fuzzy relations. '''  #Here we define two sets young(x) and middleAged(x) def student(x):  if x<=20:  return 1 elif 20<x<=30:  return (30-x)/(30-20) elif x>30:  return 0  def doJob(x):  if x>30:  return 1 elif x<=20:  return 0 elif 20<x<=30:  return (x-20)/(10)  class Fuzzy:  def init (self,item,membership): self.val = item self.membership = membership  def return\_membership(self):  return self.membership import random  x1 = random.sample(range(15, 40), 20) x2 = x1  set1 = [] #student set2 = [] #doJob  print(x1) print(x2)  [28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]  [28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]  for member in x1:  val = student(member) obj = Fuzzy(member,val) set1.append(obj)  for member in x2:  val = doJob(member) obj = Fuzzy(member,val) set2.append(obj) |

|  |
| --- |
| for obj in set1: print(obj.val,':',obj.membership)  print('=================================')  for obj in set2: print(obj.val,':',obj.membership)  28 : 0.2  35 : 0  25 : 0.5  29 : 0.1  34 : 0  30 : 0.0  16 : 1  32 : 0  19 : 1  39 : 0  38 : 0  33 : 0  37 : 0  23 : 0.7  21 : 0.9  27 : 0.3  18 : 1  17 : 1  22 : 0.8  20 : 1  ================================= 28 : 0.8  35 : 1  25 : 0.5  29 : 0.9  34 : 1  30 : 1.0  16 : 0  32 : 1  19 : 0  39 : 1  38 : 1  33 : 1  37 : 1  23 : 0.3  21 : 0.1  27 : 0.7  18 : 0  17 : 0  22 : 0.2  20 : 0  student\_x = [] student\_y = [] for obj in set1:  student\_x.append(obj.val) student\_y.append(obj.membership)  job\_x=[] job\_y=[]  for obj in set2: job\_x.append(obj.val) job\_y.append(obj.membership)  print(student\_x,student\_y,job\_x,job\_y) |

|  |
| --- |
| [28, 35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20]  [0.2, 0, 0.5, 0.1, 0, 0.0, 1, 0, 1, 0, 0, 0, 0, 0.7, 0.9, 0.3, 1, 1, 0.8, 1] [28,  35, 25, 29, 34, 30, 16, 32, 19, 39, 38, 33, 37, 23, 21, 27, 18, 17, 22, 20] [0.8,  1, 0.5, 0.9, 1, 1.0, 0, 1, 0, 1, 1, 1, 1, 0.3, 0.1, 0.7, 0, 0, 0.2, 0]  import numpy as np  import matplotlib.pyplot as plt  from scipy.interpolate import interp1d  x=np.array(student\_x) y=np.array(student\_y)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  #=======================================  x1=np.array(job\_x) y1=np.array(job\_y)  x\_new1 = np.linspace(x1.min(), x1.max(),500)  f1 = interp1d(x1, y1, kind='quadratic') y\_smooth1=f1(x\_new1)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y)  plt.plot (x\_new1,y\_smooth1) plt.scatter (x1, y1)  #complement #on set 1  import numpy as np  import matplotlib.pyplot as plt  from scipy.interpolate import interp1d  x=np.array(student\_x) y=np.array(student\_y)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y) |

|  |
| --- |
| NOTset1 = [] for obj in set1:  new\_obj = Fuzzy(obj.val,1- obj.membership) NOTset1.append(new\_obj)  student\_x1 = [] student\_y1 = [] for obj in NOTset1:  student\_x1.append(obj.val) student\_y1.append(obj.membership)  x\_compli=np.array(student\_x1) y\_compli=np.array(student\_y1)  x\_new\_compli = np.linspace(x\_compli.min(), x\_compli.max(),500) f2 = interp1d(x\_compli, y\_compli, kind='quadratic')  x=np.array(student\_x) y=np.array(student\_y)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y)  x\_compli=np.array(student\_x1) y\_compli=np.array(student\_y1)  x\_new\_compli = np.linspace(x\_compli.min(), x\_compli.max(),500)  f2 = interp1d(x\_compli, y\_compli, kind='quadratic') y\_smooth\_compli=f2(x\_new\_compli)  plt.title('Set and its Complement : blue: set1 and orange: complement of set1') plt.plot (x\_new\_compli,y\_smooth\_compli)  plt.scatter (x\_compli, y\_compli) |

|  |
| --- |
| #union import math union = []  for obj1,obj2 in zip(set1,set2):  new\_obj = Fuzzy(obj1.val,max(obj1.membership,obj2.membership)) union.append(new\_obj)  Ux = []  Uy = []  for obj in union: Ux.append(obj.val) Uy.append(obj.membership)  x=np.array(Ux) y=np.array(Uy)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y)    #intersection intersection = []  for obj1,obj2 in zip(set1,set2):  new\_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership)) intersection.append(new\_obj) |

|  |
| --- |
| Ix = []  Iy = []  for obj in intersection: Ix.append(obj.val) Iy.append(obj.membership)  x=np.array(Ix)  y=np.array(Iy)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y)    #difference  #difference of 2 sets F1 and F2 is # F1 - F2 = F1 intersect |-F2|  # set2 complement NOTset2 = []  for obj in set2:  new\_obj = Fuzzy(obj.val,1- obj.membership) NOTset2.append(new\_obj)  intersection1 = []  for obj1,obj2 in zip(set1,NOTset2):  new\_obj = Fuzzy(obj1.val,min(obj1.membership,obj2.membership)) intersection1.append(new\_obj)  Ix1 = []  Iy1 = []  for obj in intersection1: Ix1.append(obj.val) Iy1.append(obj.membership) |

|  |
| --- |
| x=np.array(Ix1) y=np.array(Iy1)  x\_new = np.linspace(x.min(), x.max(),500)  f = interp1d(x, y, kind='quadratic') y\_smooth=f(x\_new)  plt.plot (x\_new,y\_smooth) plt.scatter (x, y) |

neuron

**Assignment No.2**

|  |
| --- |
| Code of Program –  # MIN-MAX composition of 2 fuzzy relatons '''  Let 2 relations be R and S and given as follows '''  R = [ [0.62,0.45,0.4],  [0.12,0.1,0,34],  [0.43,0.27,0.9]  ]  S = [ [0.11,0.33,0.98],  [0.23,0.37,0.74],  [0.6,0.5,0.19]  ]  #min-max composition will be N\*N matrix where N is dimension of both R and S N = 3  min\_max = []  for i in range(N):  List = []  for j in range(N):  #ith row and jth column I = R[i]  new = []  for k in range(N): new.append(min(I[k],S[k][j]))  List.append(max(new)) min\_max.append(List)  print('R : ')  for i in range(N):  for j in range(N): print(R[i][j],' ',end='')  print() print('================================')  print('S:')  for i in range(N):  for j in range(N): print(S[i][j],' ',end=' ')  print()  print('=========MIN-MAX COMPOSITION IS: ==========')  for i in range(N):  for j in range(N): print(min\_max[i][j],' ',end='')  print() |

|  |
| --- |
| R :  0.62 0.45 0.4  0.12 0.1 0  0.43 0.27 0.9  ================================ S:  0.11 0.33 0.98  0.23 0.37 0.74  0.6 0.5 0.19  =========MIN-MAX COMPOSITION IS: ========== 0.4 0.4 0.62  0.11 0.12 0.12  0.6 0.5 0.43 |

**Assignment No.3**

|  |
| --- |
| Code of Program –  import tkinter  from tkinter import \* inputs = [] number\_of\_gates = 0 outx=0  outy=0 label\_id = -1  #create application window window = Tk()  #to rename the title of the window by default it is 'tk' window.title('GUI')  #set window size  window.geometry('%dx%d+%d+%d' % (1000, 800, 0, 0))  #window can't be resized window.resizable(False,False)  def draw\_diagram(): c.delete("all")  number\_of\_gates = int(e.get())  #print(number\_of\_gates) #print(type\_of\_gate) canvas\_width = c.winfo\_width() canvas\_height = c.winfo\_height()  #we need to divide entire canvas into 3 columns in order to display input nodes 'y' node and output line  #all of them are not of same width #input : 2/5 'y':1/5 line:2/5 import math  #separate 3 parts using line  x = math.ceil(canvas\_width \* 2 / 5) c.create\_line(x,0,x,canvas\_height,fill="red",width=3) x = x + math.ceil(canvas\_width \* 1 / 5) c.create\_line(x,0,x,canvas\_height,fill="red",width=3)  #draw circles for gate  part\_per\_circle = canvas\_height // number\_of\_gates x = math.ceil(canvas\_width \* 1 / 5)  y = part\_per\_circle // 2 r = 30  for i in range(part\_per\_circle):  c.create\_oval(x-r, y-r, x+r, y+r,fill="orange",width=5)  y += part\_per\_circle |

|  |
| --- |
| #draw 'y'  x1 = math.ceil(canvas\_width \* 2 / 5) + math.ceil(canvas\_width \* 1 / 10) y1 = canvas\_height // 2  r = 30  c.create\_oval(x1-r, y1-r, x1+r, y1+r,fill="blue",width=5) #draw input lines  x = math.ceil(canvas\_width \* 1 / 5) y = part\_per\_circle // 2  for i in range(number\_of\_gates): c.create\_line(x,y,x1,y1,fill="black",width='2') c.create\_text(x-5,y,text=i,font=("Calibri",20),fill="red") e1 = tkinter.Entry(c,width=5,bg="pink",font=("Calibri",16)) t1 = (x1+x-30)//2  t2 = (y1+y)//2 e1.place(x = t1 ,y = t2) inputs.append(e1)  c.update()  y += part\_per\_circle  #draw output line  x = math.ceil(canvas\_width \* 3 / 5) + math.ceil(canvas\_width \* 1 / 5) y = canvas\_height // 2 c.create\_line(x1,y1,x,y,fill="green",width='3') c.create\_text(x1+10,y1,text='y',font=("Calibri",20),fill="red") c.update()  #draw final answer circle x = x + 40  c.create\_oval(x-40,y-40,x+40,y+40,fill="green",width=10)  submit.place\_forget() getAns.place(x=40,y=250) #end of function definition#  def Mc\_culloch\_pits():  '''WEIGHT OF EVERY INPUT EDGE IS +1 HENCE OUTPUT DEPENDS UPON INPUT GIVEN BY USER ONLY'''  number\_of\_gates = int(e.get()) print('Number of gates = ',number\_of\_gates) type\_of\_gate = var.get()  #calculate input weighted sum sum = 0  for item in inputs:  sum+= int(item.get()) print('sum = ',sum) output = 'NA'  if type\_of\_gate == "AND": #threshold definition  threshold = number\_of\_gates \* 1 if sum >= threshold:  output = 1 else: |

|  |
| --- |
| output = 0  elif type\_of\_gate == "OR": #threshold definition threshold = 1 \* 1  if sum >= threshold: output = 1  else:  output = 0  elif type\_of\_gate == "NAND": #threshold definition  threshold = number\_of\_gates \* 1 if sum >= threshold:  output = 0 else:  output = 1  elif type\_of\_gate == "NOR": #threshold definition threshold = 1 \* 1  if sum >= threshold: output = 0  else:  output = 1  import math  outx = math.ceil(c.winfo\_width() \* 3 / 5) + math.ceil( c.winfo\_width() \* 1  / 5) + 40  outy = c.winfo\_height() // 2  print('For type of gate = ',type\_of\_gate,' answer is = ',output) ans.place(x=outx-20,y=outy-20)  v1.set(output) #c.create\_text(outx,outy,text=output,font=("Calibri",30),fill="yellow") c.update()  c = tkinter.Canvas(window, height=800, width=800) c.pack(side="right")  #text label for GATES  gates = tkinter.Label(window,text="choose gate") gates.configure(font=("Times New Roman", 16, "bold")) gates.place(x=40,y=50)  #option menu for names of GATEs var = StringVar(window) var.set("AND") # initial value  option = OptionMenu(window, var, "AND", "OR", "NAND", "NOR") option.place(x=40,y=100)  #text label for input of number of gates  number = tkinter.Label(window,text="Provide no. of inputs") number.configure(font=("Times New Roman", 14, "bold")) number.place(x=20,y=175)  #number of inputs  e = tkinter.Entry(window) |

|  |
| --- |
| e.place(x=20,y=200)  #submit button  submit = tkinter.Button(window,text = "SUBMIT",command=draw\_diagram) submit.place(x=40,y=250)  #getAnswer button  getAns = tkinter.Button(window,text = "Get Answer",command=Mc\_culloch\_pits) getAns.place\_forget()  #answer  v1 = StringVar()  ans = tkinter.Label(c,textvariable=v1,bg="green",font = "Verdana 20 bold",fg="yellow")  ans.place\_forget() window.mainloop() |

**Assignment No.4**

|  |
| --- |
| Code of Program –  # Perceptron Algorithm on the Sonar Dataset from random import seed  from random import randrange from csv import reader  # Load a CSV file  def load\_csv(filename): dataset = list()  with open(filename, 'r') as file: csv\_reader = reader(file)  for row in csv\_reader: if not row:  continue dataset.append(row)  return dataset  # Convert string column to float  def str\_column\_to\_float(dataset, column): for row in dataset:  row[column] = float(row[column].strip())  # Convert string column to integer  def str\_column\_to\_int(dataset, column): class\_values = [row[column] for row in dataset] unique = set(class\_values)  lookup = dict()  for i, value in enumerate(unique): lookup[value] = i  for row in dataset:  row[column] = lookup[row[column]] return lookup  # Split a dataset into k folds  def cross\_validation\_split(dataset, n\_folds): dataset\_split = list()  dataset\_copy = list(dataset)  fold\_size = int(len(dataset) / n\_folds) for i in range(n\_folds):  fold = list()  while len(fold) < fold\_size:  index = randrange(len(dataset\_copy)) fold.append(dataset\_copy.pop(index))  dataset\_split.append(fold) return dataset\_split  # Calculate accuracy percentage  def accuracy\_metric(actual, predicted): correct = 0  for i in range(len(actual)):  if actual[i] == predicted[i]: correct += 1  return correct / float(len(actual)) \* 100.0 |

|  |
| --- |
| # Evaluate an algorithm using a cross validation split  def evaluate\_algorithm(dataset, algorithm, n\_folds, \*args): folds = cross\_validation\_split(dataset, n\_folds)  scores = list() for fold in folds:  train\_set = list(folds)  train\_set.remove(fold)  train\_set = sum(train\_set, []) test\_set = list()  for row in fold: row\_copy = list(row)  test\_set.append(row\_copy) row\_copy[-1] = None  predicted = algorithm(train\_set, test\_set, \*args) actual = [row[-1] for row in fold]  accuracy = accuracy\_metric(actual, predicted) scores.append(accuracy)  return scores  # Make a prediction with weights def predict(row, weights):  activation = weights[0] for i in range(len(row)-1):  activation += weights[i + 1] \* row[i] return 1.0 if activation >= 0.0 else 0.0  # Estimate Perceptron weights using stochastic gradient descent def train\_weights(train, l\_rate, n\_epoch):  weights = [0.0 for i in range(len(train[0]))] for epoch in range(n\_epoch):  for row in train:  prediction = predict(row, weights) error = row[-1] - prediction  weights[0] = weights[0] + l\_rate \* error for i in range(len(row)-1):  weights[i + 1] = weights[i + 1] + l\_rate \* error \* row[i] return weights  # Perceptron Algorithm With Stochastic Gradient Descent def perceptron(train, test, l\_rate, n\_epoch):  predictions = list()  weights = train\_weights(train, l\_rate, n\_epoch) for row in test:  prediction = predict(row, weights) predictions.append(prediction)  return(predictions) |

|  |
| --- |
| # Test the Perceptron algorithm on the sonar dataset seed(1)  # load and prepare data  filename = 'D:\\BE\_4434\\sonar-data-set\\sonar.all-data.csv' dataset = load\_csv(filename)  for i in range(len(dataset[0])-1): str\_column\_to\_float(dataset, i)  # convert string class to integers str\_column\_to\_int(dataset, len(dataset[0])-1) # evaluate algorithm  n\_folds = 3  l\_rate = 0.01  n\_epoch = 500  scores = evaluate\_algorithm(dataset, perceptron, n\_folds, l\_rate, n\_epoch) print('Scores: %s' % scores)  print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))  Scores: [81.15942028985508, 69.56521739130434, 62.31884057971014]  Mean Accuracy: 71.014% |

**Assignment No.5**

|  |
| --- |
| Code of Program – import random import numpy as np  W = 0.5  c1 = 2  c2 = 2  target = 1  n\_iterations = 50 target\_error = 1e-6 n\_particles = 30  def visualize(particle ): x=[]  y=[]  for part in particle : x.append(part[0]) y.append(part[1])  import matplotlib.pyplot as plt  plt.scatter(x, y, label= "stars", color= "green", marker= "\*", s=30) plt.show()  #function that models the problem def fitness\_function(position):  return position[0]\*\*2 + position[1]\*\*2 + 1  particle\_position\_vector = np.array([np.array([(-  1) \*\* (bool(random.getrandbits(1))) \* random.random()\*50, (- 1)\*\*(bool(random.getrandbits(1))) \* random.random()\*50]) for \_ in range(n\_particles  )])  visualize(particle\_position\_vector) pbest\_position = particle\_position\_vector  pbest\_fitness\_value = np.array([float('inf') for \_ in range(n\_particles)]) gbest\_fitness\_value = float('inf')  gbest\_position = np.array([float('inf'), float('inf')])  velocity\_vector = ([np.array([0, 0]) for \_ in range(n\_particles)]) iteration = 0  while iteration < n\_iterations: for i in range(n\_particles):  fitness\_cadidate = fitness\_function(particle\_position\_vector[i]) #print(fitness\_cadidate, ' ', particle\_position\_vector[i])  if(pbest\_fitness\_value[i] > fitness\_cadidate): pbest\_fitness\_value[i] = fitness\_cadidate pbest\_position[i] = particle\_position\_vector[i]  if(gbest\_fitness\_value > fitness\_cadidate): gbest\_fitness\_value = fitness\_cadidate gbest\_position = particle\_position\_vector[i]  if(abs(gbest\_fitness\_value - target) < target\_error): break |

|  |
| --- |
| for i in range(n\_particles):  new\_velocity = (W\*velocity\_vector[i]) + (c1\*random.random()) \* (pbest\_posit ion[i] - particle\_position\_vector[i]) + (c2\*random.random()) \* (gbest\_position- particle\_position\_vector[i])  new\_position = new\_velocity + particle\_position\_vector[i] particle\_position\_vector[i] = new\_position  iteration = iteration + 1  print("The best position is ", gbest\_position, "in iteration number ", iteration) visualize(particle\_position\_vector) |

**Mini Project (SCOA)**

|  |
| --- |
| Code of Program –  import random  def random\_chromosome(size): #making random chromosomes  return [ random.randint(1, nq) for \_ in range(nq) ]  def fitness(chromosome):  horizontal\_collisions = sum([chromosome.count(queen)-1 for queen in chromosome])/2  diagonal\_collisions = 0  n = len(chromosome)  left\_diagonal = [0] \* 2\*n  right\_diagonal = [0] \* 2\*n  for i in range(n):  left\_diagonal[i + chromosome[i] - 1] += 1  right\_diagonal[len(chromosome) - i + chromosome[i] - 2] += 1  diagonal\_collisions = 0  for i in range(2\*n-1):  counter = 0  if left\_diagonal[i] > 1:  counter += left\_diagonal[i]-1  if right\_diagonal[i] > 1:  counter += right\_diagonal[i]-1  diagonal\_collisions += counter / (n-abs(i-n+1))    return int(maxFitness - (horizontal\_collisions + diagonal\_collisions)) #28-(2+3)=23  def probability(chromosome, fitness):  return fitness(chromosome) / maxFitness  def random\_pick(population, probabilities):  populationWithProbabilty = zip(population, probabilities)  total = sum(w for c, w in populationWithProbabilty)  r = random.uniform(0, total)  upto = 0  for c, w in zip(population, probabilities):  if upto + w >= r:  return c  upto += w  assert False, "Shouldn't get here"    def reproduce(x, y): #doing cross\_over between two chromosomes  n = len(x)  c = random.randint(0, n - 1)  return x[0:c] + y[c:n]  def mutate(x): #randomly changing the value of a random index of a chromosome  n = len(x)  c = random.randint(0, n - 1)  m = random.randint(1, n)  x[c] = m  return x  def genetic\_queen(population, fitness):  mutation\_probability = 0.03  new\_population = []  probabilities = [probability(n, fitness) for n in population]  for i in range(len(population)):  x = random\_pick(population, probabilities) #best chromosome 1  y = random\_pick(population, probabilities) #best chromosome 2  child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes  if random.random() < mutation\_probability:  child = mutate(child)  print\_chromosome(child)  new\_population.append(child)  if fitness(child) == maxFitness: break  return new\_population  def print\_chromosome(chrom):  print("Chromosome = {}, Fitness = {}"  .format(str(chrom), fitness(chrom)))  if \_\_name\_\_ == "\_\_main\_\_":  nq = int(input("Enter Number of Queens: ")) #say N = 8  maxFitness = (nq\*(nq-1))/2 # 8\*7/2 = 28  population = [random\_chromosome(nq) for \_ in range(100)]    generation = 1  while not maxFitness in [fitness(chrom) for chrom in population]:  print("=== Generation {} ===".format(generation))  population = genetic\_queen(population, fitness)  print("")  print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))  generation += 1  chrom\_out = []  print("Solved in Generation {}!".format(generation-1))  for chrom in population:  if fitness(chrom) == maxFitness:  print("");  print("One of the solutions: ")  chrom\_out = chrom  print\_chromosome(chrom)    board = []  for x in range(nq):  board.append(["x"] \* nq)    for i in range(nq):  board[nq-chrom\_out[i]][i]="Q"    def print\_board(board):  for row in board:  print (" ".join(row))    print()  print\_board(board)        Output :  Maximum Fitness = 14  === Generation 3 ===  Chromosome = [5, 2, 2, 6, 3, 3], Fitness = 13  Chromosome = [1, 6, 4, 3, 6, 4], Fitness = 12  Chromosome = [1, 6, 6, 3, 6, 2], Fitness = 11  Chromosome = [2, 4, 6, 4, 4, 5], Fitness = 11  Chromosome = [1, 5, 2, 6, 3, 3], Fitness = 14  Chromosome = [2, 2, 6, 6, 4, 6], Fitness = 10  Chromosome = [2, 2, 4, 1, 2, 5], Fitness = 11  Chromosome = [2, 4, 6, 1, 3, 1], Fitness = 14  Chromosome = [5, 6, 2, 5, 6, 3], Fitness = 12  Chromosome = [4, 6, 6, 3, 3, 2], Fitness = 12  Chromosome = [4, 2, 5, 5, 2, 1], Fitness = 12  Chromosome = [2, 6, 2, 3, 3, 3], Fitness = 10  Chromosome = [6, 4, 2, 1, 2, 3], Fitness = 12  Chromosome = [6, 4, 2, 5, 3, 5], Fitness = 13  Chromosome = [5, 1, 4, 4, 5, 2], Fitness = 12  Chromosome = [2, 4, 2, 1, 3, 5], Fitness = 13  Chromosome = [2, 4, 6, 1, 3, 5], Fitness = 15  Maximum Fitness = 15  Solved in Generation 3!  One of the solutions:  Chromosome = [2, 4, 6, 1, 3, 5], Fitness = 15  x x Q x x x  x x x x x Q  x Q x x x x  x x x x Q x  Q x x x x x  x x x Q x x |
|  |

**Mini Project**

**Aim:** Real time implementation for data streaming using google cloud pubsub and bigquery

**Required Inputs:** CSV file of food ordering system

**Expected Output:** Real time data inserts into bigquery

**Challenges:**

1. With growing demands for faster analysis,these days all the business are hungry to ingest the data and process the data as soon as generated this helps them in making real time business decision eventually leading to business growth
2. The data preparation activities like cleaning the data where necessary and write to bigquery

**Key Points of the topic:**

**Google cloud platform :**

Google Cloud offers services for compute, storage, networking, big data, machine learning and IoT, as well as cloud management, security and developer tools.

**Apache Beam :**

Apache Beam is an open source, unified model for defining both batch- and streaming-data parallel-processing pipelines. The Apache Beam programming model simplifies the mechanics of large-scale data processing. Using one of the Apache Beam SDKs, you build a program that defines the pipeline.

**Bigquery :**

BigQuery is a fully managed enterprise data warehouse that helps you manage and analyze your data with built-in features like machine learning, geospatial analysis, and business intelligence

**publisher :**

Publisher is an application that can create and sends message to topic.it receives the real time client data and send it ahead to topic in the form of messages

**Topic :**

topics are like a channel to which messages are sent by publisher.each topic is kind of a category of published messages in fact the publisher is one who creates the topic and publishes the message in it.

topic can receive the messages from the publisher in order to ensure its delivery the message is stored in the persistent storage where it can be passed to subscribe on its interest/subscription or more previously we can say that subscriber subscription.

**message :**

message is combination of data and attributes that publisher sends to topic and which eventually delivered to subscriber.attributes are optional they may or may not be included in the messages attributes are basically key value pairs which can describe more information about a message.

**subscriber :**

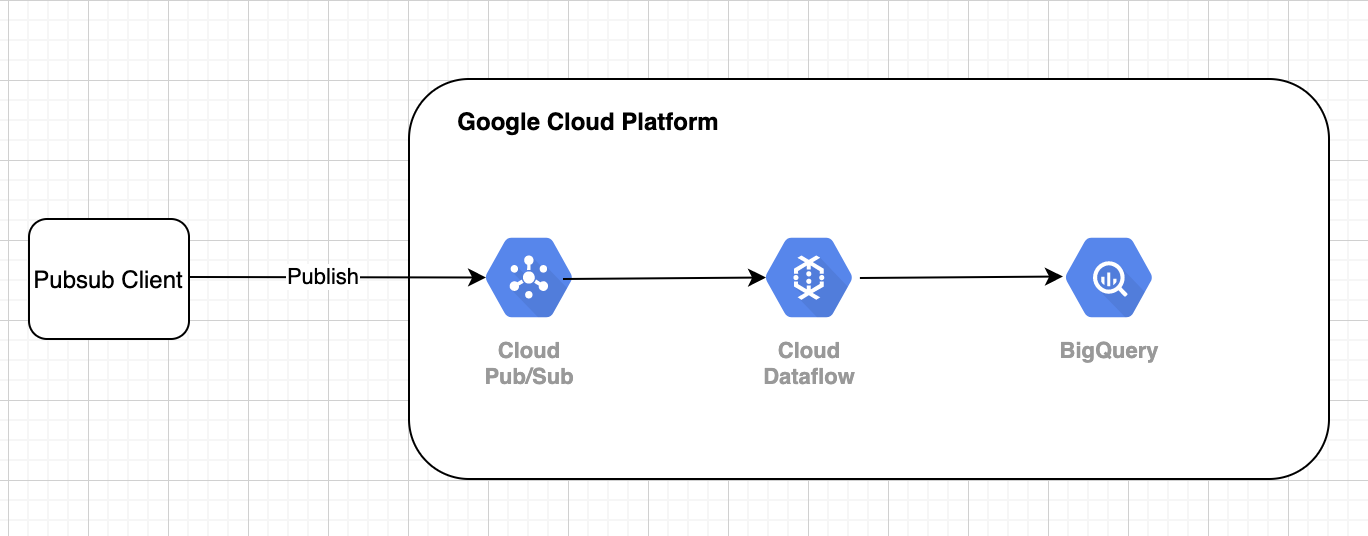
The subscriber is the consumer of those messages and the subscriber does not receive the message directly.from publisher rather it creates a subscription to that topic.to receive the message from the publisher .

you can say that by creating the subscription.subscriber is subscribing to a category of message which of its interest

**subscription :**

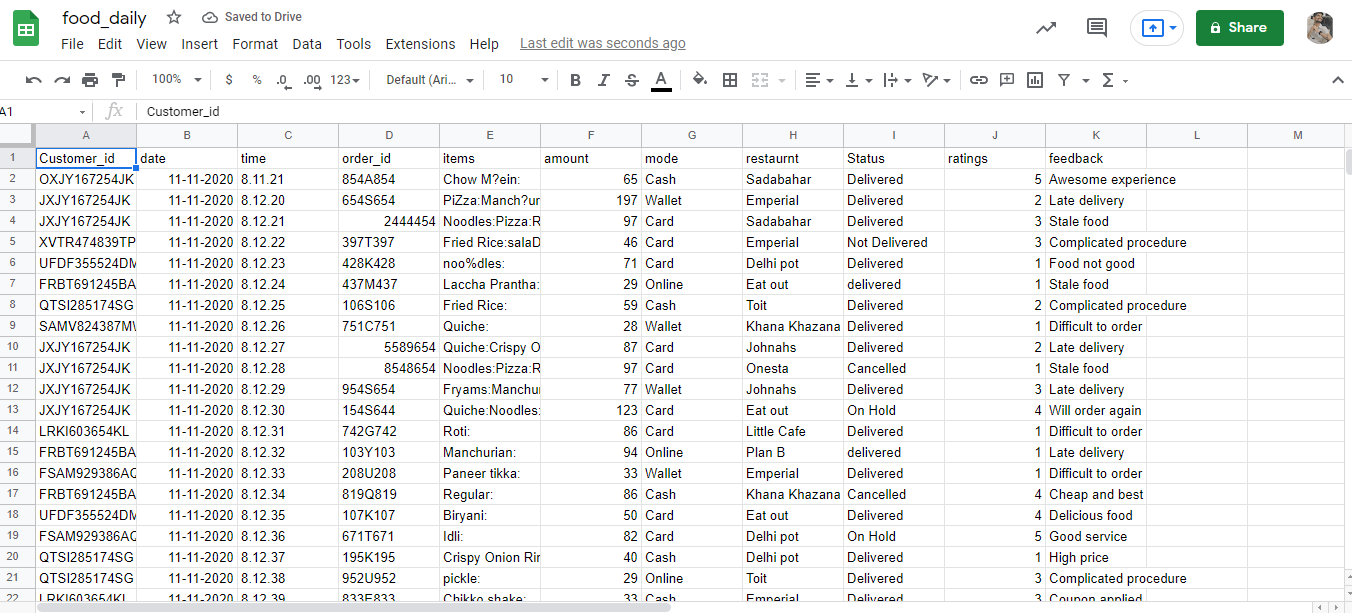
subscriptions are named resources which represent the stream of messages from single specific topic delivered to subscriber

**Architecture :**

****

**Code :-**

**Food ordering csv**

****

**File\_to\_pubsub.py**

**from google.cloud import pubsub\_v1**

**import time**

**publisher = pubsub\_v1.PublisherClient()**

**topic\_name = 'projects/<project ID>/topics/data\_stream\_from\_file'**

**try:**

**publisher.create\_topic(topic\_name)**

**except:**

**print ('Topic already exists')**

**with open('food\_daily.csv') as f\_in:**

**for line in f\_in:**

**# Data must be a bytestring**

**data = line**

**future = publisher.publish(topic\_name, data=data)**

**print(future.result())**

**time.sleep(1)**

**Code\_written\_pubsub.py**

**#project-id:dataset\_id.table\_id**

**delivered\_table\_spec = 'bigquery-demo-285417:dataset\_food\_orders.delivered\_orders'**

**#project-id:dataset\_id.table\_id**

**other\_table\_spec = 'bigquery-demo-285417:dataset\_food\_orders.other\_status\_orders'**

**import apache\_beam as beam**

**from apache\_beam.options.pipeline\_options import PipelineOptions, StandardOptions**

**import argparse**

**from google.cloud import bigquery**

**parser = argparse.ArgumentParser()**

**'''**

**format: projects/<project-id>/topics/<topic>**

**input\_topic = 'projects/test-pipeline-253103/topics/test-pipeline-topic'**

**'''**

**parser.add\_argument('--input',**

**dest='input',**

**required=True,**

**help='Input file to process.')**

**path\_args, pipeline\_args = parser.parse\_known\_args()**

**inputs\_pattern = path\_args.input**

**options = PipelineOptions(pipeline\_args)**

**options.view\_as(StandardOptions).streaming = True**

**#options.streaming = True # PubSub only works in streaming mode**

**p = beam.Pipeline(options = options)**

**def remove\_last\_colon(row): # OXJY167254JK,11-09-2020,8:11:21,854A854,Chow M?ein:,65,Cash,Sadabahar,Delivered,5,Awesome experience**

**cols = row.split(',') # [(OXJY167254JK) (11-11-2020) (8:11:21) (854A854) (Chow M?ein:) (65) (Cash) ....]**

**item = str(cols[4]) # item = Chow M?ein:**

**if item.endswith(':'):**

**cols[4] = item[:-1] # cols[4] = Chow M?ein**

**return ','.join(cols) # OXJY167254JK,11-11-2020,8:11:21,854A854,Chow M?ein,65,Cash,Sadabahar,Delivered,5,Awesome experience**

**def remove\_special\_characters(row): # oxjy167254jk,11-11-2020,8:11:21,854a854,chow m?ein,65,cash,sadabahar,delivered,5,awesome experience**

**import re**

**cols = row.split(',') # [(oxjy167254jk) (11-11-2020) (8:11:21) (854a854) (chow m?ein) (65) (cash) ....]**

**ret = ''**

**for col in cols:**

**clean\_col = re.sub(r'[?%&]','', col)**

**ret = ret + clean\_col + ',' # oxjy167254jk,11-11-2020,8:11:21,854a854,chow mein:,65,cash,sadabahar,delivered,5,awesome experience,**

**ret = ret[:-1] # oxjy167254jk,11-11-2020,8:11:21,854A854,chow mein:,65,cash,sadabahar,delivered,5,awesome experience**

**return ret**

**def print\_row(row):**

**print row**

**cleaned\_data = (**

**p**

**#| beam.io.ReadFromText(inputs\_pattern, skip\_header\_lines=1)**

**| beam.io.ReadFromPubSub(topic=inputs\_pattern)**

**| beam.Map(remove\_last\_colon)**

**| beam.Map(lambda row: row.lower())**

**| beam.Map(remove\_special\_characters)**

**| beam.Map(lambda row: row+',1') # oxjy167254jk,11-11-2020,8:11:21,854a854,chow mein,65,cash,sadabahar,delivered,5,awesome experience,1**

**)**

**delivered\_orders = (**

**cleaned\_data**

**| 'delivered filter' >> beam.Filter(lambda row: row.split(',')[8].lower() == 'delivered')**

**)**

**other\_orders = (**

**cleaned\_data**

**| 'Undelivered Filter' >> beam.Filter(lambda row: row.split(',')[8].lower() != 'delivered')**

**)**

**'''(cleaned\_data**

**| 'count total' >> beam.combiners.Count.Globally() # 920**

**| 'total map' >> beam.Map(lambda x: 'Total Count:' +str(x)) # Total Count: 920**

**| 'print total' >> beam.Map(print\_row)**

**)**

**(delivered\_orders**

**| 'count delivered' >> beam.combiners.Count.Globally()**

**| 'delivered map' >> beam.Map(lambda x: 'Delivered count:'+str(x))**

**| 'print delivered count' >> beam.Map(print\_row)**

**)**

**(other\_orders**

**| 'count others' >> beam.combiners.Count.Globally()**

**| 'other map' >> beam.Map(lambda x: 'Others count:'+str(x))**

**| 'print undelivered' >> beam.Map(print\_row)**

**)'''**

**# BigQuery**

**client = bigquery.Client()**

**dataset\_id = "{}.dataset\_food\_orders".format(client.project)**

**try:**

**client.get\_dataset(dataset\_id)**

**except:**

**dataset = bigquery.Dataset(dataset\_id) #**

**dataset.location = "US"**

**dataset.description = "dataset for food orders"**

**dataset\_ref = client.create\_dataset(dataset, timeout=30) # Make an API request.**

**def to\_json(csv\_str):**

**fields = csv\_str.split(',')**

**json\_str = {"customer\_id":fields[0],**

**"date": fields[1],**

**"timestamp": fields[2],**

**"order\_id": fields[3],**

**"items": fields[4],**

**"amount": fields[5],**

**"mode": fields[6],**

**"restaurant": fields[7],**

**"status": fields[8],**

**"ratings": fields[9],**

**"feedback": fields[10],**

**"new\_col": fields[11]**

**}**

**return json\_str**

**table\_schema = 'customer\_id:STRING,date:STRING,timestamp:STRING,order\_id:STRING,items:STRING,amount:STRING,mode:STRING,restaurant:STRING,status:STRING,ratings:STRING,feedback:STRING,new\_col:STRING'**

**(delivered\_orders**

**| 'delivered to json' >> beam.Map(to\_json)**

**| 'write delivered' >> beam.io.WriteToBigQuery(**

**delivered\_table\_spec,**

**schema=table\_schema,**

**create\_disposition=beam.io.BigQueryDisposition.CREATE\_IF\_NEEDED,**

**write\_disposition=beam.io.BigQueryDisposition.WRITE\_APPEND,**

**additional\_bq\_parameters={'timePartitioning': {'type': 'DAY'}}**

**)**

**)**

**(other\_orders**

**| 'others to json' >> beam.Map(to\_json)**

**| 'write other\_orders' >> beam.io.WriteToBigQuery(**

**other\_table\_spec,**

**schema=table\_schema,**

**create\_disposition=beam.io.BigQueryDisposition.CREATE\_IF\_NEEDED,**

**write\_disposition=beam.io.BigQueryDisposition.WRITE\_APPEND,**

**additional\_bq\_parameters={'timePartitioning': {'type': 'DAY'}}**

**)**

**)**

**def create\_view():**

**print('Creating VIEW thread ...')**

**view\_name = "daily\_food\_orders"**

**dataset\_ref = client.dataset('dataset\_food\_orders')**

**view\_ref = dataset\_ref.table(view\_name)**

**view\_to\_create = bigquery.Table(view\_ref)**

**view\_to\_create.view\_query = 'select \* from `bigquery-demo-285417.dataset\_food\_orders.delivered\_orders` where \_PARTITIONDATE = DATE(current\_date())'**

**view\_to\_create.view\_use\_legacy\_sql = False**

**try:**

**client.create\_table(view\_to\_create)**

**except:**

**print 'View already exists'**

**from threading import Timer**

**t = Timer(25.0, create\_view)**

**t.start()**

**from apache\_beam.runners.runner import PipelineState**

**ret = p.run()**

**ret.wait\_until\_finish()**

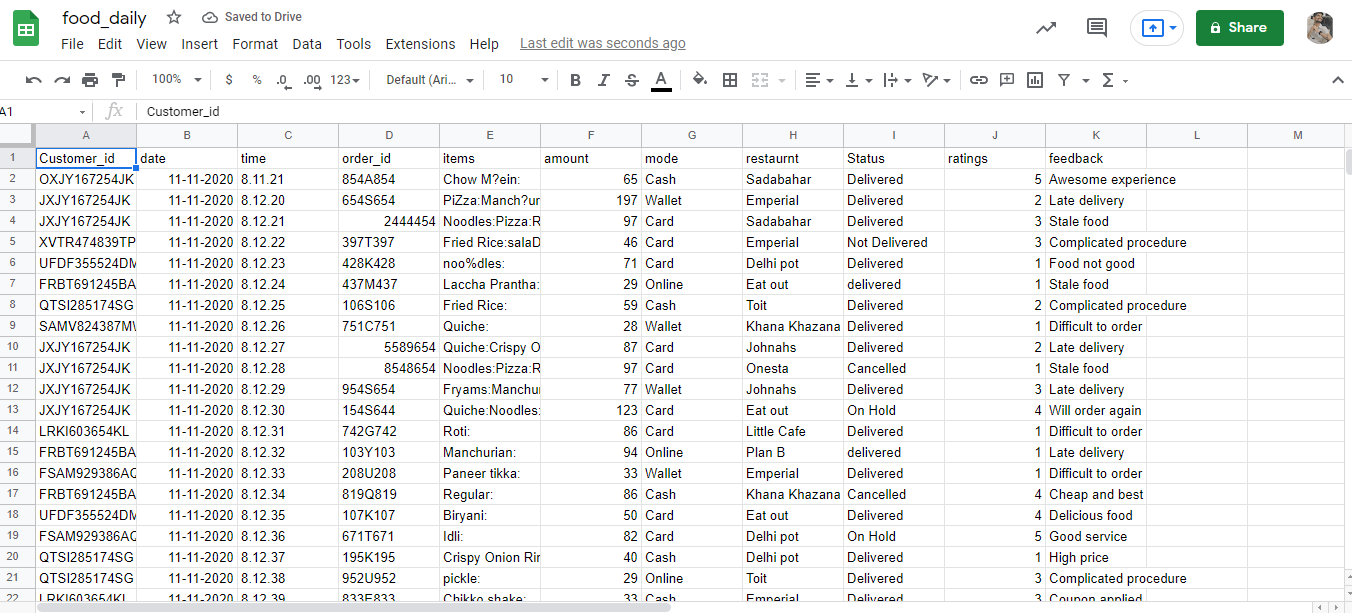
**if ret.state == PipelineState.DONE:**

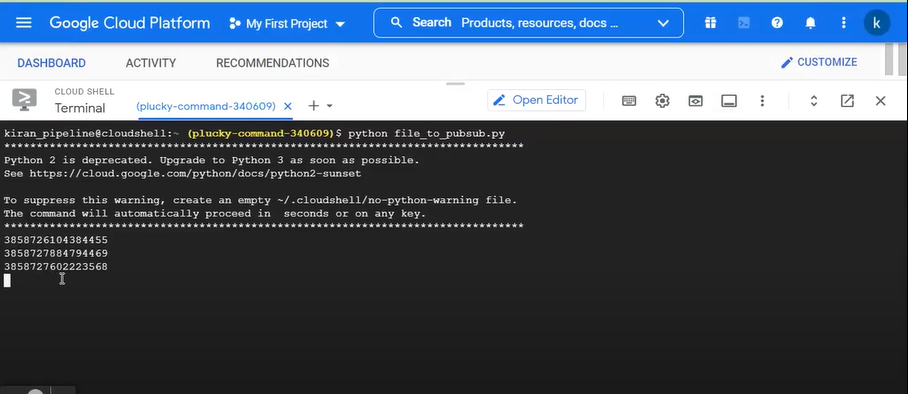
**print('Success!!!')**

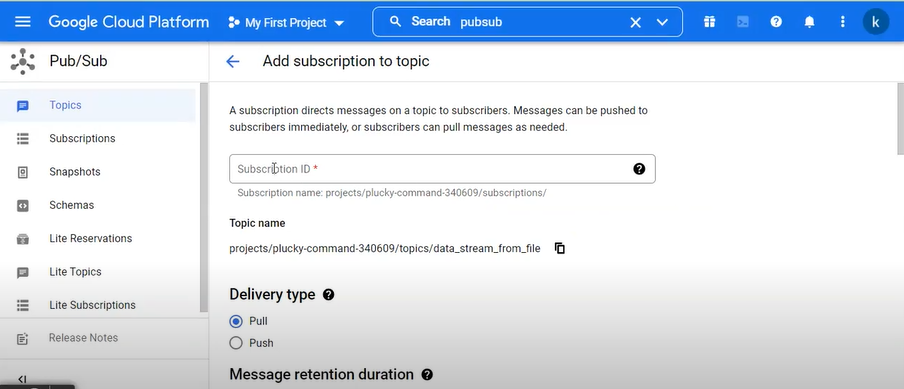
**else:**

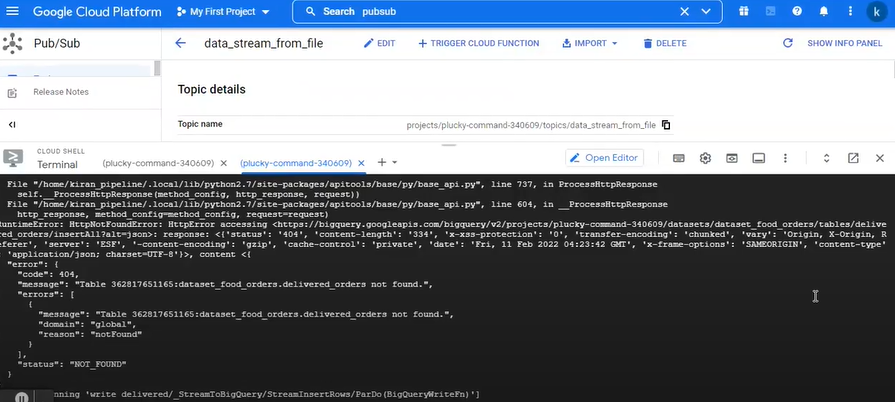
**print('Error Running beam pipeline')**

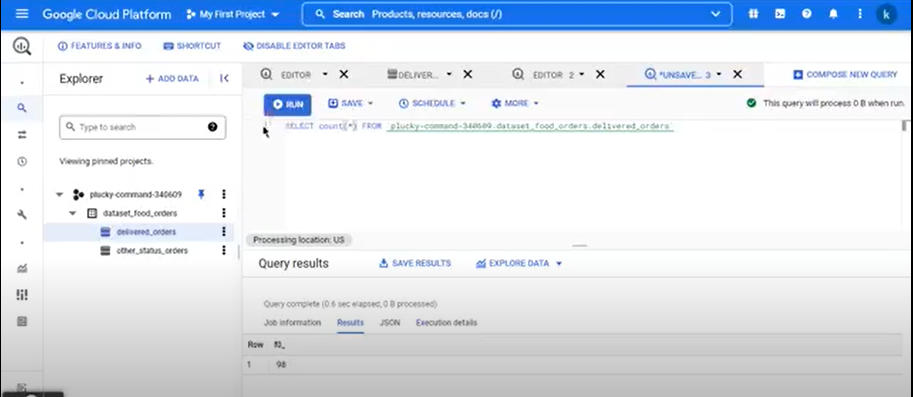
**OUTPUT :**

****

****

****

****

****