def run(self):  
 # See if the data is complete:  
 flag1=False  
 flag2=False  
 flag3=False  
  
 if self.Hospital\_length\_of\_stay\_mean!=0 and self.Hospital\_length\_of\_stay\_std!=0 and self.Daily\_arrival\_rate!=[] and self.Percentage\_hospitalized!=0:  
 flag2=True  
 if self.Hospital\_length\_of\_stay\_mean!=0 and self.Hospital\_length\_of\_stay\_std!=0 and self.Daily\_arrival\_rate!=[] and self.Percentage\_icu!=0:  
 flag3=True  
 if self.ED\_length\_of\_stay\_mean!=0 and self.ED\_length\_of\_stay\_std!=0 and self.Daily\_arrival\_rate!=0 and self.Hourly\_pattern!=0:  
 flag1=True  
  
 if flag1==True or flag2==True or flag3==True:  
 pdf=FPDF()  
 pdf.add\_page()  
 pdf.set\_font("Arial",size=12)  
 pdf.cell(200,10,txt="CHED-CAT Report",ln=1, align="C")  
  
  
  
 processbartotallength=flag1\*len(self.Hourly\_pattern)\*len(self.Daily\_arrival\_rate)+flag2\*len(self.Daily\_arrival\_rate)+flag3\*len(self.Daily\_arrival\_rate)  
  
 if flag2==True:  
 pdf.cell(200,10,txt="For Hospital:",ln=1,fill="green")  
 mu = log(square(self.Hospital\_length\_of\_stay\_mean) / sqrt(square(self.Hospital\_length\_of\_stay\_mean) + square(self.Hospital\_length\_of\_stay\_mean)))  
 stddev = sqrt(log(1 + square(self.Hospital\_length\_of\_stay\_mean) / square(self.Hospital\_length\_of\_stay\_mean)))  
 ArrivalRate = array(self.Daily\_arrival\_rate, dtype=float)  
 ArrivalRate=ArrivalRate\*self.Percentage\_hospitalized  
 TotalTimeLength = ArrivalRate.size  
 initial\_condition=self.Hospital\_initial\_condition  
  
 step = 1  
 t = 1  
 tlist = []  
 tlist = range(1, TotalTimeLength + 1)  
 zeros= [0 for i in tlist]  
 mt = []  
 mt60 = []  
 mt80 = []  
 mt90 = []  
 mt95 = []  
 error = []  
 st = []  
 stl = []  
  
 while t <= TotalTimeLength:  
 ans, err = quad(f, 0, t, args=(t,mu,stddev,ArrivalRate), limit=400)  
 total\_probability = 0  
 probability\_still\_in\_system = 1 - rs(t,mu,stddev,self.Hospital\_length\_of\_stay\_mean)  
 # Calculate the mean  
 mttemp = initial\_condition \* probability\_still\_in\_system + ans  
 mt.append(mttemp)  
 # Calculate the numerical distribution:  
 temptotal = 0  
 flag = [False, False, False, False]  
 tempcdf = 0  
 while tempcdf <= 0.95:  
 i = 0  
 while i <= temptotal and i <= initial\_condition:  
 if i <= initial\_condition:  
 tempcdf = tempcdf + binom.pmf(i, initial\_condition,probability\_still\_in\_system) \* poisson.pmf((temptotal - i),ans)  
 i = i + 1  
 if tempcdf >= 0.6 and flag[0] == False:  
 mt60.append(temptotal)  
 flag[0] = True  
 if tempcdf >= 0.8 and flag[1] == False:  
 mt80.append(temptotal)  
 flag[1] = True  
 if tempcdf >= 0.9 and flag[2] == False:  
 mt90.append(temptotal)  
 flag[2] = True  
 if tempcdf >= 0.95 and flag[3] == False:  
 mt95.append(temptotal)  
 flag[3] = True  
 temptotal = temptotal + 1  
 QApplication.processEvents()  
 t = t + step  
 QApplication.processEvents()  
 self.ProgressBar.setValue(ceil((t)/processbartotallength\*100))  
  
 fig = plt.figure()  
 ax = fig.add\_subplot(1, 1, 1)  
 ax.plot(tlist, mt, label="Expected Number of Patients")  
 ax.set\_title('Expected Number of Patients in the System')  
 ax.fill\_between(tlist,zeros,mt80,color='b', alpha=0.1, label="80 percent quantile")  
 ax.set\_ylabel("Number")  
 ax.set\_xlabel("Time")  
 ax.legend(loc="upper right")  
 fig.savefig('temp2.png')  
  
 temp2=QPixmap("temp2.png")  
 self.Hospital\_Graph\_Label.setPixmap(temp2.scaled(480,360))  
  
 pdf.cell(200,10,txt=" Maximum 60 percent Occupancy Quantile is %d achieved at day %d"% (max(mt60),mt60.index(max(mt60))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 80 percent Occupancy Quantile is %d achieved at day %d"% (max(mt80),mt80.index(max(mt80))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 90 percent Occupancy Quantile is %d achieved at day %d"% (max(mt90),mt90.index(max(mt90))+1),ln=1)  
 pdf.cell(200, 10, txt=" Maximum 95 percent Occupancy Quantile is %d achieved at day %d" % (max(mt95), mt95.index(max(mt95))+1), ln=1)  
 pdf.image("temp2.png",20,w=160,h=120)  
  
 if flag3==True:  
 pdf.add\_page()  
 pdf.cell(200, 10, txt="For ICU:", ln=1, fill="green")  
 mu = log(square(self.ICU\_length\_of\_stay\_mean) / sqrt(  
 square(self.ICU\_length\_of\_stay\_mean) + square(self.ICU\_length\_of\_stay\_mean)))  
 stddev = sqrt(  
 log(1 + square(self.ICU\_length\_of\_stay\_mean) / square(self.ICU\_length\_of\_stay\_mean)))  
 ArrivalRate = array(self.Daily\_arrival\_rate, dtype=float)  
 ArrivalRate = ArrivalRate \* self.Percentage\_icu  
 TotalTimeLength = ArrivalRate.size  
 initial\_condition = self.ICU\_initial\_condition  
 step = 1  
 t = 1  
 tlist = []  
 tlist = range(1, TotalTimeLength + 1)  
 zeros = [0 for i in tlist]  
 mt = []  
 mt60 = []  
 mt80 = []  
 mt90 = []  
 mt95 = []  
 error = []  
 st = []  
 stl = []  
  
 while t <= TotalTimeLength:  
 ans, err = quad(f, 0, t, args=(t, mu, stddev, ArrivalRate), limit=400)  
 total\_probability = 0  
 probability\_still\_in\_system = 1 - rs(t, mu, stddev, self.ICU\_length\_of\_stay\_mean)  
 # Calculate the mean  
 mttemp = initial\_condition \* probability\_still\_in\_system + ans  
 mt.append(mttemp)  
 # Calculate the numerical distribution:  
 temptotal = 0  
 flag = [False, False, False, False]  
 tempcdf = 0  
 while tempcdf <= 0.95:  
 i = 0  
 while i <= temptotal and i <= initial\_condition:  
 if i <= initial\_condition:  
 tempcdf = tempcdf + binom.pmf(i, initial\_condition,  
 probability\_still\_in\_system) \* poisson.pmf((temptotal - i),  
 ans)  
 i = i + 1  
 if tempcdf >= 0.6 and flag[0] == False:  
 mt60.append(temptotal)  
 flag[0] = True  
 if tempcdf >= 0.8 and flag[1] == False:  
 mt80.append(temptotal)  
 flag[1] = True  
 if tempcdf >= 0.9 and flag[2] == False:  
 mt90.append(temptotal)  
 flag[2] = True  
 if tempcdf >= 0.95 and flag[3] == False:  
 mt95.append(temptotal)  
 flag[3] = True  
 temptotal = temptotal + 1  
 QApplication.processEvents()  
 t = t + step  
 QApplication.processEvents()  
 self.ProgressBar.setValue(ceil((t+flag2\*len(self.Daily\_arrival\_rate)) / processbartotallength \* 100))  
  
 fig = plt.figure()  
 ax = fig.add\_subplot(1, 1, 1)  
 ax.plot(tlist, mt, label="Expected Number of Patients")  
 ax.set\_title('Expected Number of Patients in the System')  
 ax.fill\_between(tlist, zeros, mt80, color='b', alpha=0.1, label="80 percent quantile")  
 ax.set\_ylabel("Number")  
 ax.set\_xlabel("Time")  
 ax.legend(loc="upper right")  
 fig.savefig('temp3.png')  
  
 temp3 = QPixmap("temp3.png")  
 self.ICU\_Graph\_Label.setPixmap(temp3.scaled(480, 360))  
  
 pdf.cell(200,10,txt=" Maximum 60 percent Occupancy Quantile is %d achieved at day %d"% (max(mt60),mt60.index(max(mt60))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 80 percent Occupancy Quantile is %d achieved at day %d"% (max(mt80),mt80.index(max(mt80))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 90 percent Occupancy Quantile is %d achieved at day %d"% (max(mt90),mt90.index(max(mt90))+1),ln=1)  
 pdf.cell(200, 10, txt=" Maximum 95 percent Occupancy Quantile is %d achieved at day %d" % (max(mt95), mt95.index(max(mt95))+1), ln=1)  
 pdf.image("temp3.png",20,w=160,h=120)  
  
 if flag1==True:  
 pdf.add\_page()  
 pdf.cell(200, 10, txt="For ED:", ln=1, fill="green")  
 mu = log(square(self.ED\_length\_of\_stay\_mean) / sqrt(  
 square(self.ED\_length\_of\_stay\_mean) + square(self.ED\_length\_of\_stay\_mean)))  
 stddev = sqrt(  
 log(1 + square(self.ED\_length\_of\_stay\_mean) / square(self.ED\_length\_of\_stay\_mean)))  
  
 ArrivalRate=[]  
 for j in range(len(self.Daily\_arrival\_rate)):  
 for k in range(24):  
 ArrivalRate.append(float(self.Daily\_arrival\_rate[j])\*float(self.Hourly\_pattern[k]))  
 ArrivalRate = array(ArrivalRate, dtype=float)  
 TotalTimeLength = ArrivalRate.size  
 initial\_condition = self.ED\_initial\_condition  
  
 step = 1  
 t = 1  
 tlist = []  
 tlist = range(1, TotalTimeLength + 1)  
 zeros = [0 for i in tlist]  
 mt = []  
 mt60 = []  
 mt80 = []  
 mt90 = []  
 mt95 = []  
 error = []  
 st = []  
 stl = []  
  
 while t <= TotalTimeLength:  
 ans, err = quad(f, 0, t, args=(t, mu, stddev, ArrivalRate), limit=250)  
 total\_probability = 0  
 probability\_still\_in\_system = 1 - rs(t, mu, stddev, self.ED\_length\_of\_stay\_mean)  
 # Calculate the mean  
 mttemp = initial\_condition \* probability\_still\_in\_system + ans  
 mt.append(mttemp)  
 # Calculate the numerical distribution:  
 temptotal = 0  
 flag = [False, False, False, False]  
 tempcdf = 0  
 while tempcdf <= 0.95:  
 i = 0  
 while i <= temptotal and i <= initial\_condition:  
 if i <= initial\_condition:  
 tempcdf = tempcdf + binom.pmf(i, initial\_condition,  
 probability\_still\_in\_system) \* poisson.pmf((temptotal - i),  
 ans)  
 i = i + 1  
 if tempcdf >= 0.6 and flag[0] == False:  
 mt60.append(temptotal)  
 flag[0] = True  
 if tempcdf >= 0.8 and flag[1] == False:  
 mt80.append(temptotal)  
 flag[1] = True  
 if tempcdf >= 0.9 and flag[2] == False:  
 mt90.append(temptotal)  
 flag[2] = True  
 if tempcdf >= 0.95 and flag[3] == False:  
 mt95.append(temptotal)  
 flag[3] = True  
 temptotal = temptotal + 1  
 QApplication.processEvents()  
 t = t + step  
 QApplication.processEvents()  
 self.ProgressBar.setValue(ceil((t + flag2 \* len(self.Daily\_arrival\_rate)+flag3\*len(self.Daily\_arrival\_rate)) / processbartotallength \* 100))  
  
 fig = plt.figure()  
 ax = fig.add\_subplot(1, 1, 1)  
 ax.plot(tlist, mt, label="Expected Number of Patients")  
 ax.set\_title('Expected Number of Patients in the System')  
 ax.fill\_between(tlist, zeros, mt80, color='b', alpha=0.1, label="80 percent quantile")  
 ax.set\_ylabel("Number")  
 ax.set\_xlabel("Time")  
 ax.legend(loc="upper right")  
 fig.savefig('temp1.png')  
  
 temp1 = QPixmap("temp1.png")  
 self.ED\_Graph\_Label.setPixmap(temp1.scaled(480, 360))  
  
 pdf.cell(200,10,txt=" Maximum 60 percent Occupancy Quantile is %d achieved at hour %d"% (max(mt60),mt60.index(max(mt60))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 80 percent Occupancy Quantile is %d achieved at hour %d"% (max(mt80),mt80.index(max(mt80))+1),ln=1)  
 pdf.cell(200,10,txt=" Maximum 90 percent Occupancy Quantile is %d achieved at hour %d"% (max(mt90),mt90.index(max(mt90))+1),ln=1)  
 pdf.cell(200, 10, txt=" Maximum 95 percent Occupancy Quantile is %d achieved at hour %d" % (max(mt95), mt95.index(max(mt95))+1), ln=1)  
 pdf.image("temp1.png",20,w=160,h=120)  
  
 if flag1==True or flag2==True or flag3==True:  
 pdf.output("Report.pdf")  
 else:  
 QMessageBox.warning(self, "Error", "Not Enough Data")