COVID-19 Analysis

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

MEDISETTY VENKATA NIKHIL	**121910312016**
MAREMALLA SANDEEP KUMAR	**121910312023**
SAGI LAKSHMI PASYANTHI	**121910312035**
VIVEK CHADARAM	**121910312046**

Under the esteemed guidance of

Dr. S. VENKATA LAKSHMI
Assistant Professor, GST

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

October 2022

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

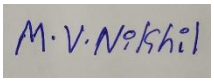
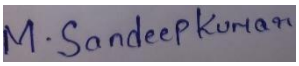
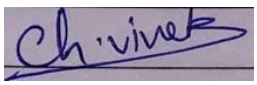
GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "**COVID-19 Analysis**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:28-10-22

Registration No(s).	**Name(s)**	**Signature(s)**
121910312016	MEDISETTY VENKATA NIKHIL	
121910312023	MAREMALLA SANDEEP KUMAR	
121910312035	SAGI LAKSHMI PASYANTHI	
121910312046	VIVEK CHADARAM	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**COVID-19 Analysis**" is a Bonafede record of work carried out **by MEDISETTY VENKATA NIKHIL (121910312016), MAREMALLA SANDEEP KUMAR (121910312023), SAGI LAKSHMI PASYANTHI (121910312035), VIVEK CHADARAM (121910312046)** students submitted in partial fulfilment of requirements for the award of the degree of Bachelors of Technology in Computer Science and Engineering.

PROJECT GUIDE	HEAD OF THE DEPARTMENT
Dr. S.Venkata Lakshmi	**Dr. R.Sireesha**
(Assistant Professor)	**(Professor, HOD)**
CSE, GST	**CSE, GST**
GITAM	**GITAM**

ACKNOWLEDGMENT

We would like to thank our project guide **Dr. S. VENKATA LAKSHMI**, Assistant Professor, Department of CSE for her stimulating guidance and profuse assistance. We shall always cherish our association for her guidance, encouragement and valuable suggestions throughout the progress of this work. We consider it a great privilege to work under her guidance and constant support.

We also express our gratitude to the project reviewers **Mrs. K. NAGA SOUJANYA KETHA**, Assistant Professor and **Prof.**, Associate Professor, Department of CSE, GITAM (Deemed to be University) for their valuable suggestions and guidance in the completion of our project.

We consider it a privilege to express our deepest gratitude to **Dr R.SIREESHA**, Head of the Department of Computer Science and engineering for her valuable suggestions and constant motivation that greatly helped us to complete this project.

Our sincere thanks to **Prof.Ch. VIJAY SHEKAR**, Principal, GITAM Institute of Technology, GITAM (Deemed to be University) for inspiring us to learn new technologies and tools.

Finally, we deem it a great pleasure to thank one and all that helped us directly and indirectly throughout this project.

MEDISETTY VENKATA NIKHIL **121910312016**

MAREMALLA SANDEEP KUMAR **121910312023**

SAGI LAKSHMI PASYANTHI **121910312035**

VIVEK CHADARAM **121910312046**

TABLE OF CONTENTS

ABSTRACT

Since Covid-19's emergence in India, most states were affected dreadfully. The reasons for wide spread of this infectious virus had many reasons and factors which varied from state-to-state. We used data extracted automatically from daily health bulletins published by state governments to a) breakdown the behaviour of covid waves in each state and b) analyse the reasons for surges in covid cases c) to analyse and develop real life machine learning models for prediction. We learned that major factors for surge in cases were elections, festivals, restrictions, relaxations set by the state governments and poor initial vaccination drives. But in later phases less fatality rates were achieved by huge vaccinations drives and herd immunity.

INTRODUCTION

A) PROBLEM DEFINITION:

The COVID-19 India Dataset is one of the most comprehensive datasets on the pandemic in India. It aggregates data from health bulletins published online daily by governments of major Indian states. The main idea of this research is to come up with models for analysis, prediction, and insights on the evolution of the pandemic in India.

B) OBJECTIVES:

The main objectives of the analysis are listed below: -

1. To Identify Cause and Effect of different covid-waves.
    a. To identify different phases in the dataset.
    b. To discuss cause and effects in brief.
    c. Conclusion.
2. To identify the merits of vaccine and lockdown to reduce hospitalization and fatality rate.
    a. To justify the advantages of vaccine on general terms.
    b. Merits of Lockdown.
    c. Conclusion.
3. To identify relation, association to develop machine learning models to predict future values.
    a. To identify best attributes to predict cumulative positive cases and cumulative deaths.
    b. To identify and choose best models for prediction.
    c. To identify the factors which affects cumulative cases and deaths among different states.
    d. Conclusion.

PROBLEM IDENTIFICATION AND OBJECTIVES

The whole world was affected with a pandemic Covid-19. It impacted the ways of the world very deeply. India also subdued a lot damage and changes. The whole period of 2020-2022 was very chaotic. There was an outrageous surge in the amount of positive covid cases and deaths, sadly.

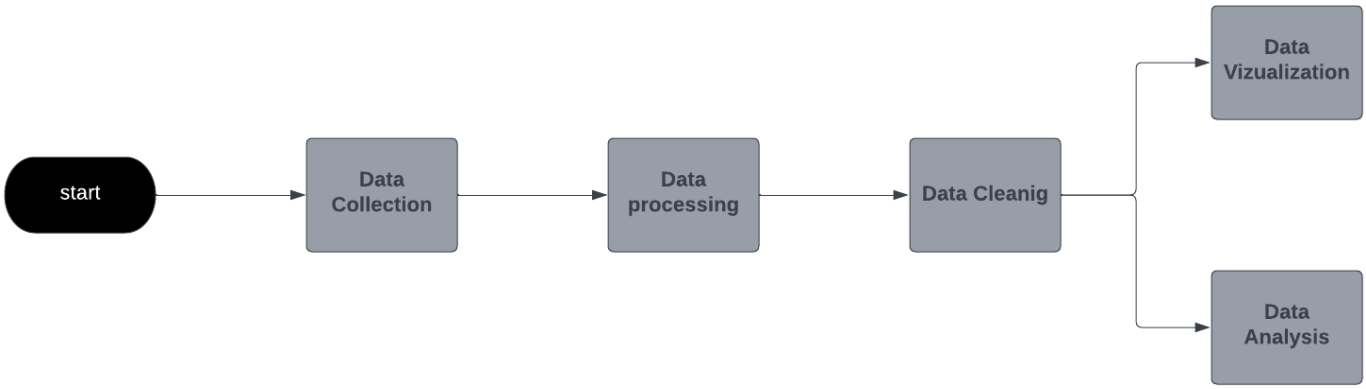
Over those 2 years, we accumulated a lot of data, which when studied properly will give a lot of useful insights. Analyzing those 2 years of pandemic can help us get ready for any such disasters in the future.

The main idea of this research is to study, analyze and predict the trends of covid, positive cases, and deaths in Telangana and Tamil Nadu.

Following are the objectives of this research:

1. To Identify Cause and Effect of different covid-waves.
2. To identify the merits of vaccine and lockdown to reduce hospitalization and fatality rate.
3. To identify relation, association to develop machine learning models to predict future values.

RESEARCH METHODOLOGY



- The study is based on primary sources of data/information. Indian central Govt. and state Govt. covid websites have been consulted to make the study an effective one.
- This fetched data from various sources is cleaned first, ie. duplicate or irrelevant observations are removed. Unwanted observations from the dataset, including duplicate observations or irrelevant observations, are also removed. Along with that filtering unwanted outliers and handling missing data are also done.
- Next, the data is split. Data visualization is done, Data visualization is a way to represent information graphically, highlighting patterns and trends in data and helping the reader to achieve quick insights. ie. The analysis of data is done using pie charts, bar graphs and scatterplots on the data.
- Then the observation of the analysis takes place and then Multivariate linear regression is performed on the observations. Linear regression helps create models to make predictions, such as predicting the cumulative no. of positive cases and deaths

OVERVIEW OF TECHNOLOGIES

Various technologies are used for achieving results in this research work i.e. Data must be sorted, structured, and visually displayed in a way that makes sense because it is difficult to grasp and make sense of data in its raw form. Data visualization is useful in situations like this. The practice of drawing insights from data through the use of graphs, sparklines, infographics, heat maps, or other visual representations is termed as data visualization. This makes it easier to understand, and use data. Tools for transforming data into pictorial representations are known as data visualization tools. Jupyter Notebook is the one utilized here to code as it is easier to write and maintain code for data visualization and machine learning. One of the most popular web-based applications for data visualization, JupyteR, allow users to produce and share documents with visualizations, equations, and live code. Statistical modelling, numerical simulation, interactive computing, and machine learning are all excellent uses for JupyteR. Data visualizations using scatterplots, bar charts, pie charts, histograms, etc. are the most popular. The popular library that is used for visualization is Matplotlib. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib is a popular Python 2-D charting library known for its incredibly high level of customization. With the help of matplotlib, interactive 2D graphs may be created, including line, scatter, and bar graphs. The language used is Python. A high-level, all-purpose programming language is Python. Code readability is prioritized in its design philosophy, which makes heavy use of indentation. Using dots to depict the link between variables, scatter plots are used to observe the relationship between them. To create a scatter plot, use the matplotlib library's scatter() method. A circular statistical graphic called

a pie chart can only show one series of data at a time. The overall 8 percentage of the provided data is represented by the chart's area. The percentage of the data pieces is represented by the area of the pie slices. A bar plot, often known as a bar chart, is a graph that uses rectangular bars with lengths and heights that are proportional to the values to represent a category of data. Both horizontal and vertical graphs of the bars are possible. The comparisons between the distinct categories are shown in a bar chart. One of the simplest and most widely used Machine Learning techniques is linear regression. It is a statistical technique for performing predictive analysis. For real, numerical, or continuous data, linear regression makes predictions. It executes a regression operation.

IMPLEMENTATION

Telangana:

TG Covid analysis

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
from datetime import datetime
plt.rcParams['figure.figsize'] = [20,12]
```

In [2]:
```
def get_monthday(date):
    date=date.split("-")
    if date[1]=="01":
        return "January"+date[0]
    if date[1]=="02":
        return "February"+date[0]
    if date[1]=="03":
        return "March"+date[0]
    if date[1]=="04":
        return "April"+date[0]
    if date[1]=="05":
        return "May"+date[0]
    if date[1]=="06":
        return "June"+date[0]
    if date[1]=="07":
        return "July"+date[0]
    if date[1]=="08":
        return "August"+date[0]
    if date[1]=="09":
        return "September"+date[0]
    if date[1]=="10":
        return "October"+date[0]
    if date[1]=="11":
        return "November"+date[0]
    if date[1]=="12":
        return "December"+date[0]
def get_month(date):
    date=date.split("-")
    if date[1]=="01":
        return "January"+date[2]
    if date[1]=="02":
        return "February"+date[2]
    if date[1]=="03":
        return "March"+date[2]
    if date[1]=="04":
        return "April"+date[2]
    if date[1]=="05":
        return "May"+date[2]
    if date[1]=="06":
        return "June"+date[2]
    if date[1]=="07":
        return "July"+date[2]
    if date[1]=="08":
        return "August"+date[2]
    if date[1]=="09":
        return "September"+date[2]
    if date[1]=="10":
        return "October"+date[2]
    if date[1]=="11":
        return "November"+date[2]
    if date[1]=="12":
        return "December"+date[2]

def get_year(date):
    date=date.split("-")
    return date[0]
def Histogram(x,y,xlabel,ylabel,title):
    x.plot(kind="hist",
        alpha=0.7,
        #         bins=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,15000],
        title=title,
        figsize=(12,8),
        color=["red"],rwidth=0.8
       )
    y.plot(kind="hist",
        alpha=0.7,
        #         bins=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,15000],
```

```
        #         bins=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,15000],
        title=title,
        figsize=(12,8),
        color=["blue"],rwidth=0.8
       )
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
def bar_graph(categories,cat_val,xlabel,ylabel,title="",color="blue"):
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.bar(categories,cat_val,width=0.8,color=color)
    plt.title(title)
    plt.show()
# This is the code of linear_regression model also.........
def linear_correlation(x,y):
    meanx=sum(x.values)/len(x)
    meany=sum(y.values)/len(y)
    if len(x.values)==len(y.values):
        num,den=0,0
        #numerator calculation
        a=0
        for i in range(len(x)):
            a+=x.values[i]*y.values[i]
            #       print(a)
        num=a*len(x)-(sum(x.values)*sum(y.values))

        #denominator calculation
        ax,ay=0,0
        for i in range(len(x)):
                ax+=x.values[i]**2
                ay+=y.values[i]**2
        p1=(len(x)*ax-(sum(x.values)**2))**0.5
        p2=(len(y)*ay-(sum(y.values)**2))**0.5
        den=p1*p2
        #print(num,den)
        return num/den
    else:
        raise Exception("Invalid Dimension")
import datetime
import calendar

def get_day(date):
```

```
def get_day(date):
    day, month, year = (int(i) for i in date.split('-'))
    dayNumber = calendar.weekday(year, month, day)
    days =["Monday", "Tuesday", "Wednesday", "Thursday",
                        "Friday", "Saturday", "Sunday"]
    return (days[dayNumber])

def set_index(x):
    if x=="June2020":
        return -2
    if x=="July2020":
        return -1
    if x=="August2020":
        return 0
    if x=="September2020":
        return 1
    if x=="October2020":
        return 2
    if x=="November2020":
        return 3
    if x=="December2020":
        return 4
    if x=="January2021":
        return 5
    if x=="February2021":
        return 6
    if x=="March2021":
        return 7
    if x=="April2021":
        return 8
    if x=="May2021":
        return 9
    if x=="June2021":
        return 10
    if x=="July2021":
        return 11
    if x=="August2021":
        return 12
    if x=="September2021":
        return 13
    if x=="October2021":
        return 14
```

```
        return 17
def piechart(categories,cat_val,title=""):
    labels=categories
    values=cat_val
    #     explode = (0,0.1)
    plt.pie(values,labels=values,counterclock=False,autopct="%.2f", shadow=True)
    plt.title(title)
    plt.legend(labels,loc=3)
    plt.rcParams['figure.figsize'] = [10,50]
    plt.show()
def rearrange_date(x):
    arr=x.split("-")
    s="-"
    s1=s.join(arr[::-1])
    return s1
# vacc_month_sum[["vax_total_24h","vax_first_dose_24h","vax_sec_dose_24h"]].plot(kind="bar")
def plot_bar_graphs(data,values,xlabel="",ylabel="",title=""):
    data[values].plot(kind="bar",title="",xlabel=xlabel,ylabel=ylabel)
def variant(date):
    date = date.split('-')
    if date[0] == '2020':
        return 'alpha'
    elif date[0] == '2021':
        if (date[1] == '01' or date[1] == '02' or date[1] == '03'):
            return 'alpha'
        elif date[1] == '12':
            return 'omicron'
        else:
            return 'delta'
    elif date[0] == '2022':
        return 'omicron'
def TPR(total_tests,positive):
    return (positive/total_tests)*100
def comorb(s):
    if s=="no comorbidity":
        return "no comorbidity"
    else:
        return "comorbidity"
def cato(s):
    if "Government" in s or "Govt" in s or "govt" in s or "government" in s:
        return "Government facility"
    if "Private" in s :
```

In [3]:
```
TGcaseinfo=pd.read_csv("TG_case_info.csv")
TGage_gender=pd.read_csv("TG_age_gender_dist.csv")
TGsymptomatic=pd.read_csv("TG_symptomatic.csv")
TGcontact=pd.read_csv("TG_contact_testing.csv")
```

In [4]:
```
TGcaseinfo["month"]=TGcaseinfo["date"].copy()
TGcaseinfo["year"]=TGcaseinfo["date"].copy()
TGcaseinfo["month"]=TGcaseinfo["month"].apply(lambda x:get_month(x))
TGcaseinfo["year"]=TGcaseinfo["year"].apply(lambda x:get_year(x))
TGcaseinfo['rise in cases']= ""
TGcaseinfo['rise in cases'] = TGcaseinfo['cases_new'].diff()
TGcaseinfo.dropna(subset=['rise in cases'],inplace=True)
```

In [5]:
```
TGcaseinfo
```

Out[5]:

	date	cases_new	cases_total	recovered_new	recovered_total	deaths_new	deaths_total	cases_in_isolation	state_CFR	national_CFR	state_recovery_rate	national_recovery_ra
1	02-09-2020	2892	130589	2240	97402	10	846	25271	0.64	1.76	74.5	77
2	03-09-2020	2817	133406	2611	100013	10	856	25293	0.64	1.75	74.9	77
3	04-09-2020	2478	135884	2011	102024	10	866	25730	0.63	1.74	75.0	77
4	05-09-2020	2511	138395	2579	104603	11	877	25729	0.63	1.73	75.5	77
5	06-09-2020	2574	140969	2927	107530	9	886	25449	0.62	1.71	76.2	77
...	...	...	...	...	...	...	...	...	...	...	...	...

PART1: COVID-WAVES AND ITS POTENTIAL CAUSES:

Graph1

The below graph plots no of positives cases, discharged patients, and deaths per day

PROPERTIES OF GRAPH1:
Blue curve:

1. If the curve is going up --> means number of cases per day are increasing 2) If the curve is going down --> means the number of cases per day are decreasing 3) If the curve is flat --> means there is no change in the number cases per day

Orange curve:

1. If the curve is going up --> means number of discharged patients per day is increasing 2) If the curve is going down --> means the number of discharged patients per day is decreasing 3) If the curve is flat --> means there is no change in the number discharged patients per day

Green curve:

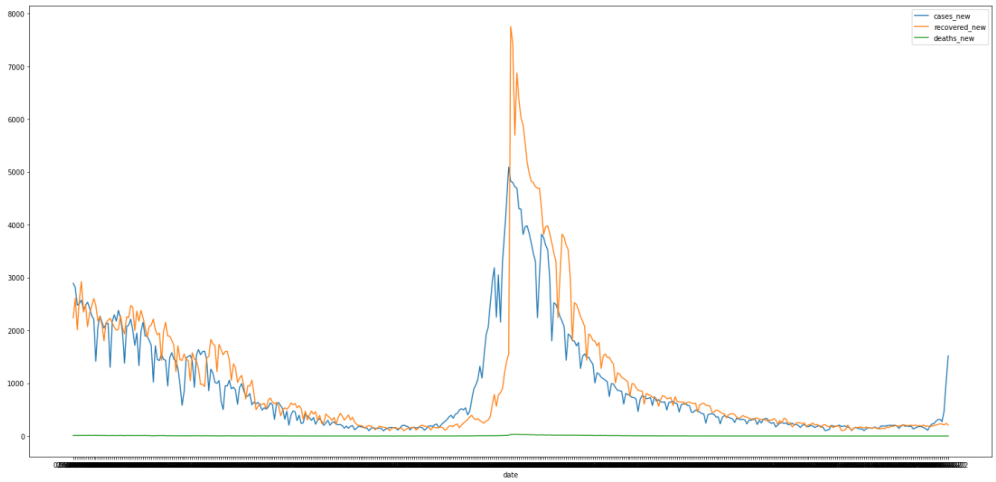
1. If the curve is going up --> means number of deaths per day is increasing 2) If the curve is going down --> means the number of deaths per day is decreasing 3) If the curve is flat --> means there is no change in the number deaths per day

In [6]:
```
plt.rcParams['figure.figsize'] = [25,12]
plt.xlabel("date")
plt.plot(TGcaseinfo["date"],TGcaseinfo["cases_new"], linestyle='-')
plt.plot(TGcaseinfo["date"],TGcaseinfo["recovered_new"], linestyle='-')
plt.plot(TGcaseinfo["date"],TGcaseinfo["deaths_new"], linestyle='-')
plt.legend(["cases_new","recovered_new","deaths_new"])
```

Out[6]:
```
<matplotlib.legend.Legend at 0x2b1584d85e0>
```


```
plt.plot(TGcaseinfo["date"],TGcaseinfo["recovered_new"], linestyle='-')
plt.plot(TGcaseinfo["date"],TGcaseinfo["deaths_new"], linestyle='-')
plt.legend(["cases_new","recovered_new","deaths_new"])
```

Out[6]:
```
<matplotlib.legend.Legend at 0x2b1584d85e0>
```



Graph2

Graph2

PHASE-1: SEPTEMBER2020 to DECEMBER2020

PROPERTIES OF GRAPH2:
Blue curve:

1. If the curve is going up --> means number of cases per day are increasing 2) If the curve is going down --> means the number of cases per day are decreasing 3) If the curve is flat --> means there is no change in the number cases per day

Orange curve:

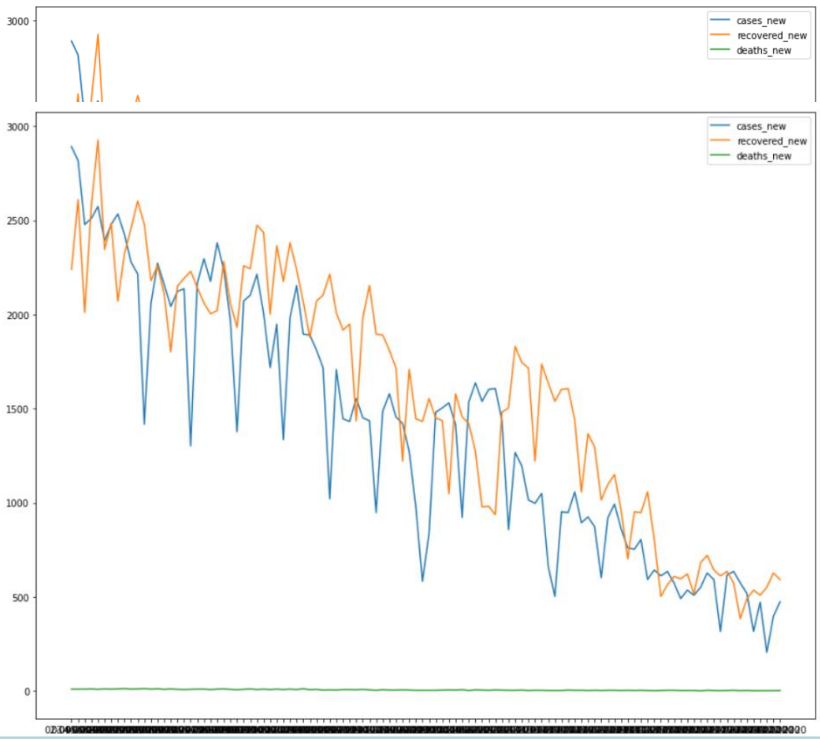
1. If the curve is going up --> means number of discharged patients per day is increasing 2) If the curve is going down --> means the number of discharged patients per day is decreasing 3) If the curve is flat --> means there is no change in the number discharged patients per day

Green curve:

1. If the curve is going up --> means number of deaths per day is increasing 2) If the curve is going down --> means the number of deaths per day is decreasing 3) If the curve is flat --> means there is no change in the number deaths per day

In [7]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.xlabel("date")
plt.plot(TGcaseinfo["date"][:list(TGcaseinfo["date"]).index("01-01-2021")],TGcaseinfo["cases_new"][:list(TGcaseinfo["date"]).index("01-01-2021")], l
plt.plot(TGcaseinfo["date"][:list(TGcaseinfo["date"]).index("01-01-2021")],TGcaseinfo["recovered_new"][:list(TGcaseinfo["date"]).index("01-01-2021")
plt.plot(TGcaseinfo["date"][:list(TGcaseinfo["date"]).index("01-01-2021")],TGcaseinfo["deaths_new"][:list(TGcaseinfo["date"]).index("01-01-2021")],
plt.legend(["cases_new","recovered_new","deaths_new"])
```

Out[7]:
```
<matplotlib.legend.Legend at 0x2b15c0fbbe0>
```



Graph3

PHASE2: JANUARY-2021 to SEPTEMBER-2021:

PROPERTIES OF GRAPH3:
Blue curve:

1. If the curve is going up --> means number of cases per day are increasing 2) If the curve is going down --> means the number of cases per day are decreasing 3) If the curve is flat --> means there is no change in the number cases per day

Orange curve:

1. If the curve is going up --> means number of discharged patients per day is increasing 2) If the curve is going down --> means the number of discharged patients per day is decreasing 3) If the curve is flat --> means there is no change in the number discharged patients per day

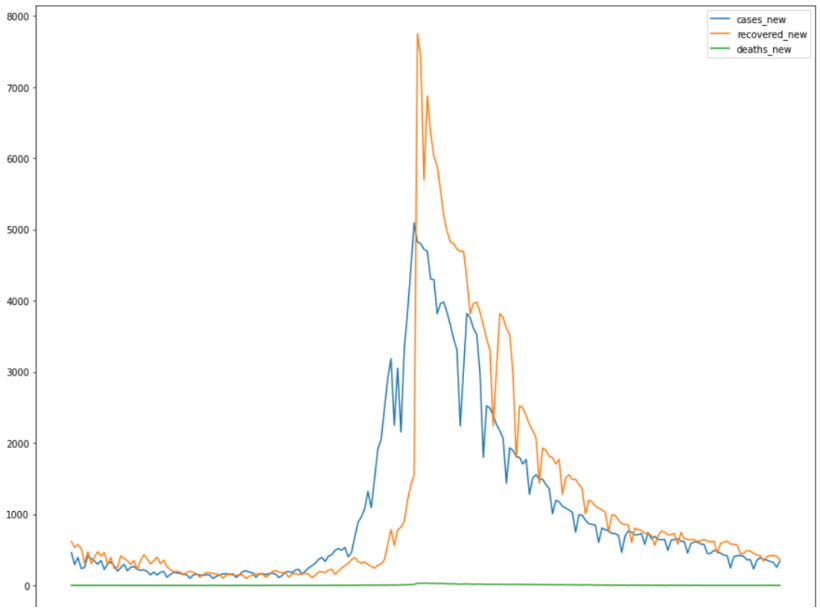
Green curve:

1. If the curve is going up --> means number of deaths per day is increasing 2) If the curve is going down --> means the number of deaths per day is decreasing 3) If the curve is flat --> means there is no change in the number deaths per day

In [8]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.xlabel("date")
plt.plot(TGcaseinfo["date"][list(TGcaseinfo["date"]).index("01-01-2021"):list(TGcaseinfo["date"]).index("01-09-2021")],TGcaseinfo["cases_new"][list(
plt.plot(TGcaseinfo["date"][list(TGcaseinfo["date"]).index("01-01-2021"):list(TGcaseinfo["date"]).index("01-09-2021")],TGcaseinfo["recovered_new"][l
plt.plot(TGcaseinfo["date"][list(TGcaseinfo["date"]).index("01-01-2021"):list(TGcaseinfo["date"]).index("01-09-2021")],TGcaseinfo["deaths_new"][list
plt.legend(["cases_new","recovered_new","deaths_new"])
```

Out[8]: <matplotlib.legend.Legend at 0x2b15c14eb00>

Out[8]: <matplotlib.legend.Legend at 0x2b15c14eb00>



From Graph1, Graph2, Graph3 we can observe that during the first phase that is SEPTEMBER2020 to DECEMBER2020 the cases gradually reduced but we can see a huge surge in cases during second phase that is JANUARY2021 to SEPTEMBER2021 one major reason can be delay and shortage of vacines
source:https://www.thenewsminute.com/article/vaccine-shortage-hyderabad-delays-vaccination-people-18-category-149787
**We can also observe that the no. of recovered cases also had a very big surge which is a very good thing

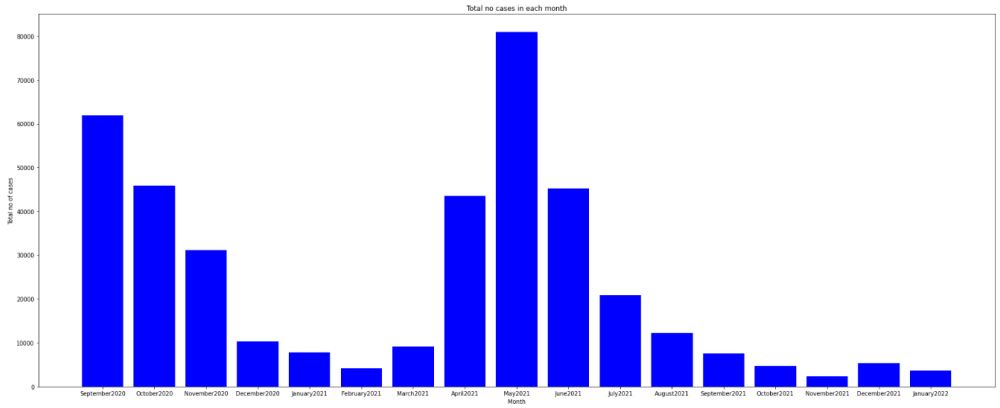
In [9]:
```
TGcaseinfosum=TGcaseinfo.groupby("month").agg("sum").reset_index()
TGcaseinfosum["index"]=TGcaseinfosum["month"].copy()
TGcaseinfosum["index"]=TGcaseinfosum["index"].apply(lambda x: set_index(x))
TGcaseinfosum.sort_values(by="index",inplace=True)
TGcaseinfosum
```

Out[9]:

	month	cases_new	cases_total	recovered_new	recovered_total	deaths_new	deaths_total	cases_in_isolation	state_CFR	national_CFR	state_recovery_rate	national_re
15	September2020	61887	4564158	63060	3669486	282	27811	682224	16.97	45.89	2241.61	
13	October2020	45801	6518482	56468	5785559	203	37371	574717	17.08	45.46	2657.85	
11	November2020	31184	7704031	39289	7214322	122	42122	372224	16.28	45.00	2806.71	
2	December2020	10294	5627519	11772	5459620	55	30256	95023	10.60	29.30	1940.19	
5	January2021	7766	8435664	11148	8266633	55	45606	68700	15.58	40.60	2841.70	
4	February2021	4162	8007374	4429	7914455	34	43656	19630	14.58	37.80	2668.54	
9	March2021	9082	9370293	6005	9236069	63	51465	33104	16.88	43.00	3055.53	
0	April2021	43535	5851885	11336	5496312	127	31555	203598	9.64	22.90	1692.49	
10	May2021	80990	11965413	112509	10883721	542	67192	1014500	12.25	24.70	1999.22	
8	June2021	45159	18163783	65475	17421533	380	105039	637211	17.19	37.90	2876.57	
7	July2021	20820	19043129	24928	18618080	139	112334	312715	17.58	39.00	2932.85	
1	August2021	12196	18919830	15216	18595435	66	111438	213098	16.82	37.70	2850.13	
16	September2021	7492	18542950	8530	18288059	44	109145	145746	16.24	36.40	2761.36	
14	October2021	4694	17382046	5132	17170731	31	102307	109008	15.10	33.80	2568.26	

The below graph plots total no of positives cases per every month from July 2020 to January 2022

In [10]:
```
plt.rcParams['figure.figsize'] = [30,12]
bar_graph(TGcaseinfosum["month"],TGcaseinfosum["cases_new"],"Month","Total no of cases","Total no cases in each month")
```



Conclusions from The above graph: As we can see the highest no of cases can be seen in september2020, october2020, april2021, May2021, June2021

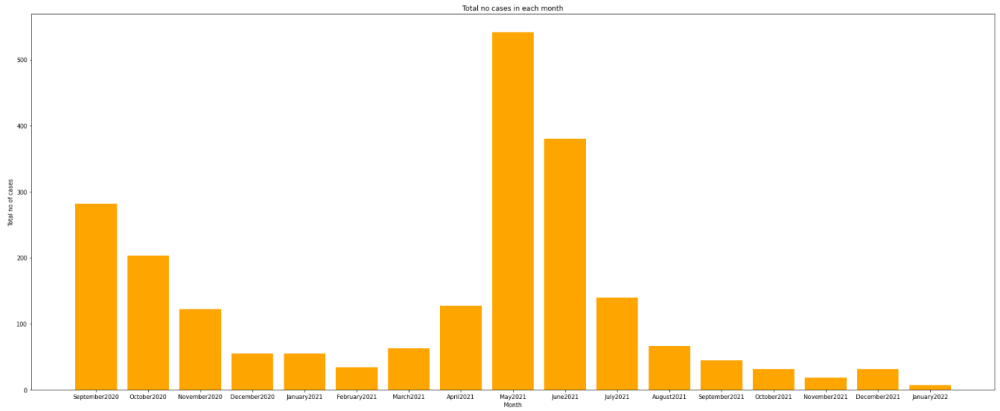
Graph5

The below bar graph plots total deaths in each month

Graph5

The below bar graph plots total deaths in each month

In [11]:
```
plt.rcParams['figure.figsize'] = [30,12]
bar_graph(TGcaseinfosum["month"],TGcaseinfosum["deaths_new"],"Month","Total no of cases","Total no cases in each month","orange")
```



Conclusions from The above graph: As we can see the highest no of deaths can be seen in september2020, May2021, June2021

Graph7

The below graph plots State CFR vs National CFR

PROPERTIES OF GRAPH7:
Blue curve:

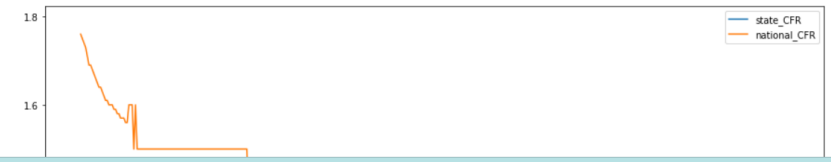
1) If the curve is going up --> means State CFR is increasing
2) If the curve is going down --> means State CFR is decreasing
3) If the curve is flat --> means there is no change in the State CFR

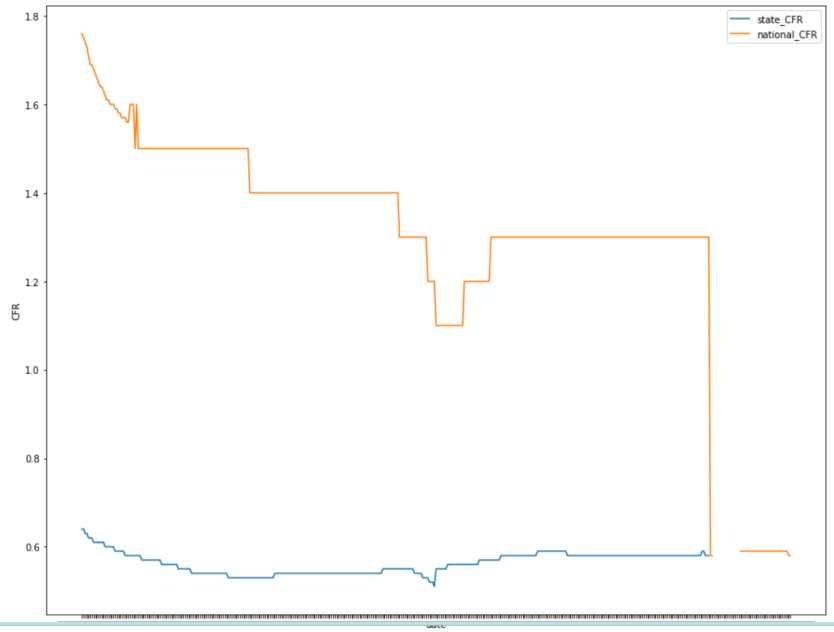
Orange curve:

1) If the curve is going up --> means number of National CFR is increasing
2) If the curve is going down --> means the National CFR is decreasing
3) If the curve is flat --> means there is no change in the National CFR

In [12]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.xlabel("date")
plt.ylabel("CFR")
plt.plot(TGcaseinfo["date"],TGcaseinfo["state_CFR"])
plt.plot(TGcaseinfo["date"],TGcaseinfo["national_CFR"])
plt.legend(["state_CFR","national_CFR"])
```

Out[12]: `<matplotlib.legend.Legend at 0x2b15be8f8b0>`





From above graph we can observe that the the states CFR(case fatality rate) is very less compared to national CFR

Graph8

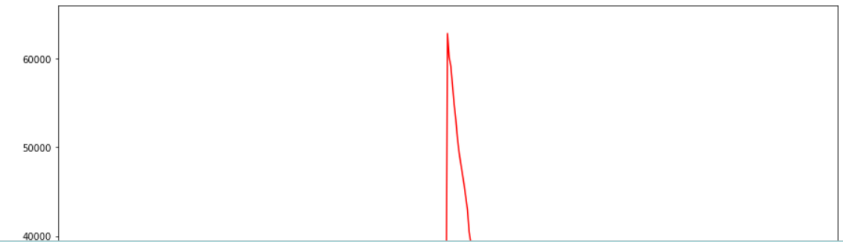
The below graph plots home isolations per day

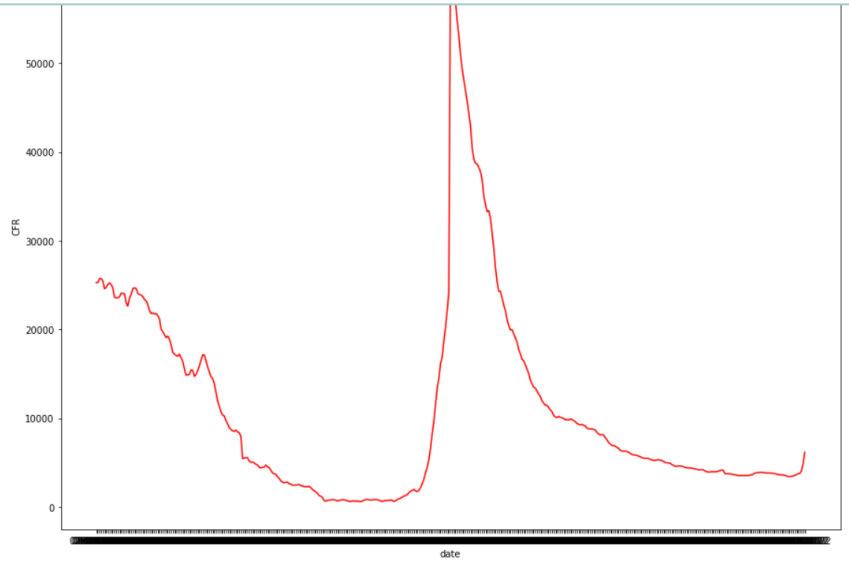
PROPERTIES OF GRAPH8:
Red curve:

1) If the curve is going up --> means no. of home isolations are increasing
2) If the curve is going down --> means no. of home isolations are decreasing
3) If the curve is flat --> means there is no change in the no. of home isolations

In [13]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.xlabel("date")
plt.ylabel("CFR")
plt.plot(TGcaseinfo["date"],TGcaseinfo["cases_in_isolation"],color="red")
```

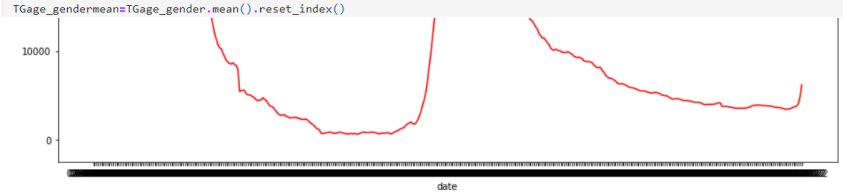
Out[13]: [<matplotlib.lines.Line2D at 0x2b15d10b670>]





From Graph6 we can observe that the no of home isolations drastically increased during second phase

In [14]:
```
TGage_gendermean=TGage_gender.mean().reset_index()
```



From Graph6 we can observe that the no of home isolations drastically increased during second phase

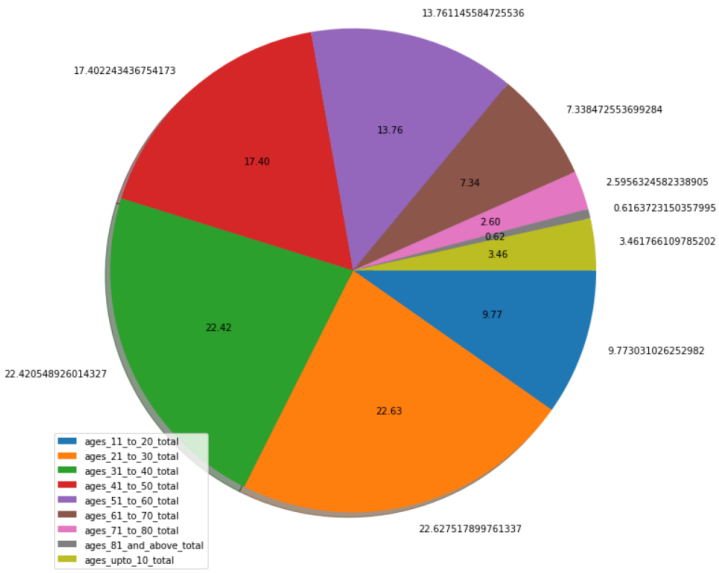
In [14]:
```
TGage_gendermean=TGage_gender.mean().reset_index()
age=[]
mean=[]
for i in range(len(list(TGage_gendermean["index"]))):
    if TGage_gendermean["index"][i]=="total_total" or TGage_gendermean["index"][i]=="total_male" or TGage_gendermean["index"][i]=="total_female":
        continue
    elif "total" in TGage_gendermean["index"][i]:
        age.append(TGage_gendermean["index"][i])
        mean.append(TGage_gendermean[0][i])
```

```
C:\Users\anura\AppData\Local\Temp/ipykernel_14628/598914970.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numeric
_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  TGage_gendermean=TGage_gender.mean().reset_index()
```

Graph7

The below piechart plots mean no of cases of each age group

In [15]:
```
piechart(age,mean,"")
```



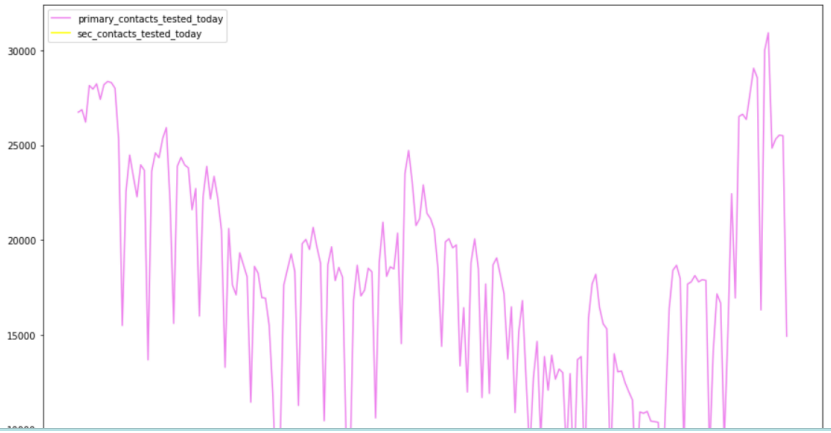
The highest no of cases can be seen in age groups 20-30, 30-40, 40-50 and 50-60

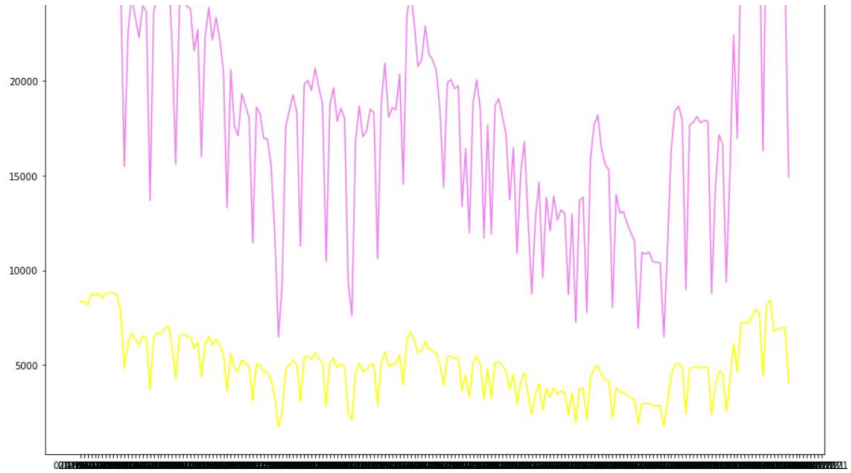
Graph8

The below graph plots no of primary contacts tested vs no of secondary contacts tested

In [16]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.plot(TGcontact["date"],TGcontact["primary_contacts_tested_today"],color="violet")
plt.plot(TGcontact["date"],TGcontact["sec_contacts_tested_today"],color="yellow")
plt.legend(["primary_contacts_tested_today","sec_contacts_tested_today"])
```

Out[16]: <matplotlib.legend.Legend at 0x2b15cc97790>





Description: From Graph 8 we can say that the no. tests conducted for primary contacts is very high compared to secondary contacts

Tamilnadu:

TN Covid Analysis

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import StrMethodFormatter
from datetime import datetime
plt.rcParams['figure.figsize'] = [20,12]
```

In [2]:
```
def get_monthday(date):
    date=date.split("-")
    if date[1]=="01":
        return "January"+date[0]
    if date[1]=="02":
        return "February"+date[0]
    if date[1]=="03":
        return "March"+date[0]
    if date[1]=="04":
        return "April"+date[0]
    if date[1]=="05":
        return "May"+date[0]
    if date[1]=="06":
        return "June"+date[0]
    if date[1]=="07":
        return "July"+date[0]
    if date[1]=="08":
        return "August"+date[0]
    if date[1]=="09":
        return "September"+date[0]
    if date[1]=="10":
        return "October"+date[0]
    if date[1]=="11":
        return "November"+date[0]
    if date[1]=="12":
```

```
        return "July"+date[0]
    if date[1]=="08":
        return "August"+date[0]
    if date[1]=="09":
        return "September"+date[0]
    if date[1]=="10":
        return "October"+date[0]
    if date[1]=="11":
        return "November"+date[0]
    if date[1]=="12":
        return "December"+date[0]
def get_month(date):
    date=date.split("-")
    if date[1]=="01":
        return "January"+date[2]
    if date[1]=="02":
        return "February"+date[2]
    if date[1]=="03":
        return "March"+date[2]
    if date[1]=="04":
        return "April"+date[2]
    if date[1]=="05":
        return "May"+date[2]
    if date[1]=="06":
        return "June"+date[2]
    if date[1]=="07":
        return "July"+date[2]
    if date[1]=="08":
        return "August"+date[2]
    if date[1]=="09":
        return "September"+date[2]
    if date[1]=="10":
        return "October"+date[2]
    if date[1]=="11":
        return "November"+date[2]
    if date[1]=="12":
        return "December"+date[2]

def get_year(date):
    date=date.split("-")
    return date[0]
def Histogram(x,y,xlabel,ylabel,title):
```

```
def get_year(date):
    date=date.split("-")
    return date[0]
def Histogram(x,y,xlabel,ylabel,title):
    x.plot(kind="hist",
        alpha=0.7,
        #         bins=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,15000],
        title=title,
        figsize=(12,8),
        color=["red"],rwidth=0.8
       )
    y.plot(kind="hist",
        alpha=0.7,
        #         bins=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,15000],
        title=title,
        figsize=(12,8),
        color=["blue"],rwidth=0.8
       )
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.show()
def bar_graph(categories,cat_val,xlabel,ylabel,title=""):
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.bar(categories,cat_val,width=0.8)
    plt.title(title)
    plt.show()
# This is the code of linear_regression model also.........
def linear_correlation(x,y):
    meanx=sum(x.values)/len(x)
    meany=sum(y.values)/len(y)
    if len(x.values)==len(y.values):
        num,den=0,0
        #numerator calculation
        a=0
        for i in range(len(x)):
            a+=x.values[i]*y.values[i]
            #         print(a)
        num=a*len(x)-(sum(x.values)*sum(y.values))

        #denominator calculation
        ax,ay=0.0
```

```
def set_index(x):
    if x=="June2020":
        return -2
    if x=="July2020":
        return -1
    if x=="August2020":
        return 0
    if x=="September2020":
        return 1
    if x=="October2020":
        return 2
    if x=="November2020":
        return 3
    if x=="December2020":
        return 4
    if x=="January2021":
        return 5
    if x=="February2021":
        return 6
    if x=="March2021":
        return 7
    if x=="April2021":
        return 8
    if x=="May2021":
        return 9
    if x=="June2021":
        return 10
    if x=="July2021":
        return 11
    if x=="August2021":
        return 12
    if x=="September2021":
        return 13
    if x=="October2021":
        return 14
    if x=="November2021":
        return 15
    if x=="December2021":
        return 16
    if x=="January2022":
        return 17
def piechart(categories,cat_val,title=""):
def piechart(categories,cat_val,title=""):
    labels=categories
    values=cat_val
    #     explode = (0,0.1)
    plt.pie(values,labels=values,counterclock=False,autopct="%.2f", shadow=True)
    plt.title(title)
    plt.legend(labels,loc=3)
    plt.rcParams['figure.figsize'] = [10,50]
    plt.show()
def rearrange_date(x):
    arr=x.split("-")
    s="-"
    s1=s.join(arr[::-1])
    return s1
# vacc_month_sum[["vax_total_24h","vax_first_dose_24h","vax_sec_dose_24h"]].plot(kind="bar")
def plot_bar_graphs(data,values,xlabel="",ylabel="",title=""):
    data[values].plot(kind="bar",title="",xlabel=xlabel,ylabel=ylabel)
def variant(date):
    date = date.split('-')
    if date[0] == '2020':
        return 'alpha'
    elif date[0] == '2021':
        if (date[1] == '01' or date[1] == '02' or date[1] == '03'):
            return 'alpha'
        elif date[1] == '12':
            return 'omicron'
        else:
            return 'delta'
    elif date[0] == '2022':
        return 'omicron'
def TPR(total_tests,positive):
    return (positive/total_tests)*100
def comorb(s):
    if s=="no comorbidity":
        return "no comorbidity"
    else:
        return "comorbidity"
def cato(s):
    if "Government" in s or "Govt" in s or "govt" in s or "government" in s:
        return "Government facility"
    if "Private" in s :
        return "Private facility"
```

In [3]:
```
TMcaseinfo=pd.read_csv("TN_case_info.csv")
TMcomorbidities=pd.read_csv("TN_comorbidities_deaths.csv")
TMdistrict=pd.read_csv("TN_district_details.csv")
TMincoming=pd.read_csv("TN_incoming_people_till_yesterday.csv")
TNindividual=pd.read_csv("TN_individual_fatalities.csv",encoding='cp1252')
TMpositive=pd.read_csv("TN_positive_cases_detail.csv")
TMpositive
```

Out[3]:

	date	active_cases	tested_positive_today	tested_positive_till_date	rtpcr_samples_tested_today	rtpcr_samples_tested_till_date	persons_tested_rt_pcr_today	persons_tested_rt_pc
0	09-08-2020	53336	5974	296901	70186	3225805	68179	
1	10-08-2020	53099	5879	302815	67153	3292958	65141	
2	11-08-2020	52810	5814	308649	67492	3360450	65490	
3	13-08-2020	53499	5810	320355	67275	3499300	65560	
4	14-08-2020	53716	5862	326245	70153	3569453	68301	
...	...	...	...	...	...	...	...	
474	01-01-2022	8340	1470	2749534	103607	57547850	103403	
475	02-01-2022	9304	1575	2751128	102237	57650087	102029	
476	03-01-2022	10364	1727	2752856	103119	57753206	102898	

In [4]:
```
TMcaseinfo.drop(columns=["testing_facilities","government_testing_facilities","private_testing_facilities"],inplace=True)
```

In [5]:
```
TMcaseinfo["month"]=TMcaseinfo["date"].copy()
TMcaseinfo["year"]=TMcaseinfo["date"].copy()
TMcaseinfo["month"]=TMcaseinfo["month"].apply(lambda x:get_month(x))
TMcaseinfo["year"]=TMcaseinfo["year"].apply(lambda x:get_year(x))
TMcaseinfo['rise in cases']= ""
TMcaseinfo['rise in cases'] = TMcaseinfo['active_cases_today'].diff()
TMcaseinfo.dropna(subset=['rise in cases'],inplace=True)
```

In [6]:
```
TMcaseinfo
```

Out[6]:

	date	active_cases_yesterday	positive_tested_cases	discharged_patients	deaths_today	active_cases_today	month	year	rise in cases
1	24-07-2020	52939	6785	6504	88	53132	July2020	24	193.0
2	25-07-2020	53132	6988	7758	89	52273	July2020	25	-859.0
3	26-07-2020	52273	6986	5471	85	53703	July2020	26	1430.0
4	28-07-2020	54896	6972	4707	88	57073	July2020	28	3370.0
5	29-07-2020	57073	6426	5927	82	57490	July2020	29	417.0
...	...	...	...	...	...	...	...	...	...
490	01-01-2022	7470	1489	611	8	8340	January2022	01	870.0
491	02-01-2022	8340	1594	624	6	9304	January2022	02	964.0
492	03-01-2022	9304	1728	662	6	10364	January2022	03	1060.0
493	04-01-2022	10364	2731	674	9	12412	January2022	04	2048.0
494	05-01-2022	12412	4862	688	9	16577	January2022	05	4165.0

494 rows × 9 columns

PART1: COVID-WAVES AND ITS POTENTIAL CAUSES:

Graph1

The below graph plots no of positives cases, discharged patients, and deaths per day

PROPERTIES OF GRAPH1: Blue curve:

1. If the curve is going up --> means number of cases per day are increasing
2. If the curve is going down --> means the number of cases per day are decreasing
3. If the curve is flat --> means there is no change in the number cases per day

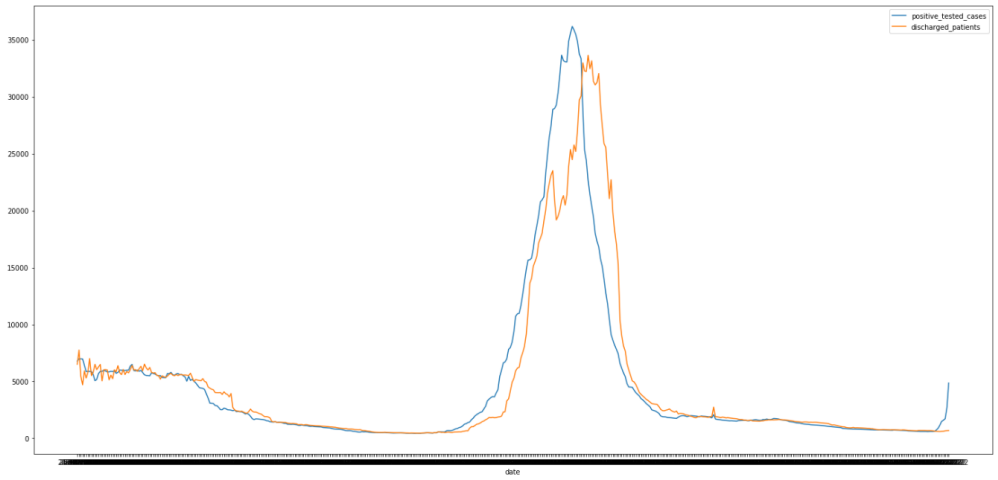
Orange curve:

1. If the curve is going up --> means number of discharged patients per day is increasing
2. If the curve is going down --> means the number of discharged patients per day is decreasing
3. If the curve is flat --> means there is no change in the number discharged patients per day

In [7]:
```
plt.rcParams['figure.figsize'] = [25,12]
plt.xlabel("date")
plt.plot(TMcaseinfo["date"],TMcaseinfo["positive_tested_cases"], linestyle='-')
plt.plot(TMcaseinfo["date"],TMcaseinfo["discharged_patients"], linestyle='-')
plt.legend(["positive_tested_cases","discharged_patients","deaths_today"])
```

Out[7]: `<matplotlib.legend.Legend at 0x22b5c0affd0>`

Out[7]: `<matplotlib.legend.Legend at 0x22b5c0affd0>`



Graph 2

PHASE-1: JULY2020 to DECEMBER2020

PROPERTIES OF GRAPH2: Blue curve:

Graph 2

PHASE-1: JULY2020 to DECEMBER2020

PROPERTIES OF GRAPH2: Blue curve:

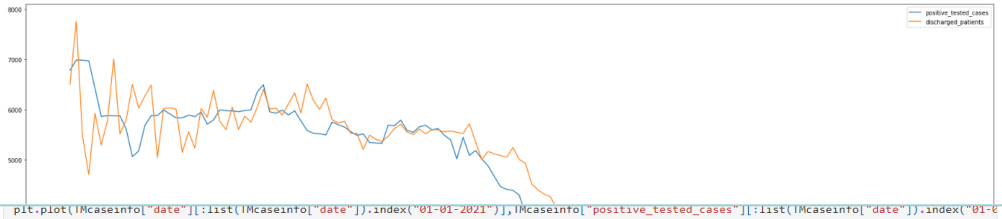
1. If the curve is going up --> means number of cases per day are increasing
2. If the curve is going down --> means the number of cases per day are decreasing
3. If the curve is flat --> means there is no change in the number cases per day

Orange curve:

1. If the curve is going up --> means number of discharged patients per day is increasing
2. If the curve is going down --> means the number of discharged patients per day is decreasing
3. If the curve is flat --> means there is no change in the number discharged patients per day

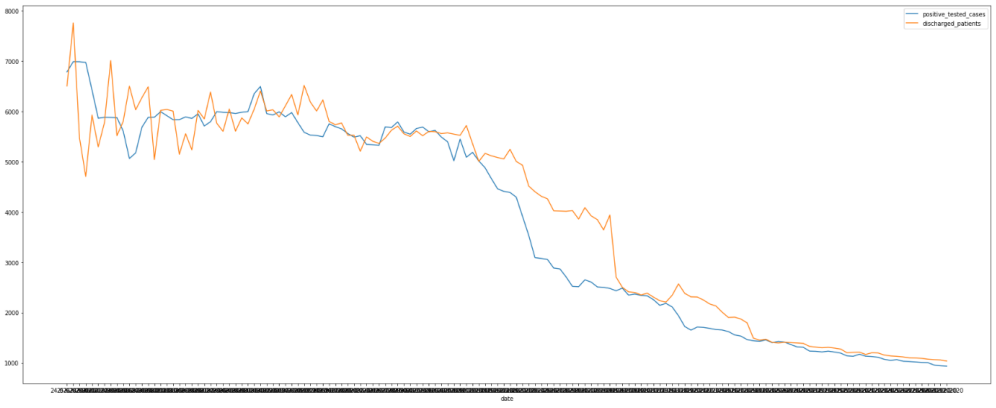
In [8]:
```
plt.rcParams['figure.figsize'] = [30,12]
plt.xlabel("date")
plt.plot(TMcaseinfo["date"][:list(TMcaseinfo["date"]).index("01-01-2021")],TMcaseinfo["positive_tested_cases"][:list(TMcaseinfo["date"]).index("01-0
plt.plot(TMcaseinfo["date"][:list(TMcaseinfo["date"]).index("01-01-2021")],TMcaseinfo["discharged_patients"][:list(TMcaseinfo["date"]).index("01-01-
plt.legend(["positive_tested_cases","discharged_patients","deaths_today"])
```

Out[8]: <matplotlib.legend.Legend at 0x22b5d740610>




```
plt.plot(TMcaseinfo["date"][:list(TMcaseinfo["date"]).index("01-01-2021")],TMcaseinfo["positive_tested_cases"][:list(TMcaseinfo["date"]).index("01-0
plt.plot(TMcaseinfo["date"][:list(TMcaseinfo["date"]).index("01-01-2021")],TMcaseinfo["discharged_patients"][:list(TMcaseinfo["date"]).index("01-01-
plt.legend(["positive_tested_cases","discharged_patients","deaths_today"])
```

Out[8]: <matplotlib.legend.Legend at 0x22b5d740610>



Graph3

PHASE2: JANUARY-2021 to SEPTEMBER-2021:

PROPERTIES OF GRAPH3: Blue curve:

1. If the curve is going up --> means number of cases per day are increasing

Graph3

PHASE2: JANUARY-2021 to SEPTEMBER-2021:

PROPERTIES OF GRAPH3: Blue curve:

1. If the curve is going up --> means number of cases per day are increasing
2. If the curve is going down --> means the number of cases per day are decreasing
3. If the curve is flat --> means there is no change in the number cases per day

Orange curve:

1. If the curve is going up --> means number of discharged patients per day is increasing
2. If the curve is going down --> means the number of discharged patients per day is decreasing
3. If the curve is flat --> means there is no change in the number discharged patients per day

In [9]:
```
TMcaseinfo
```

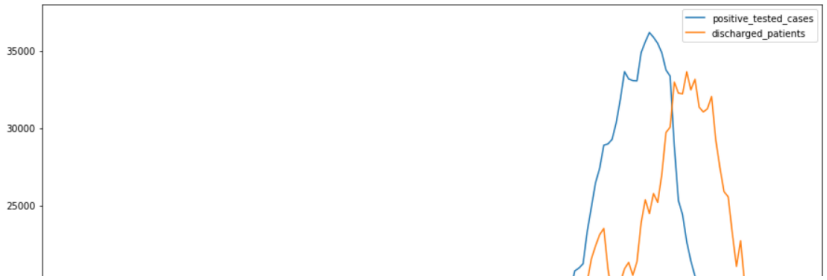
Out[9]:

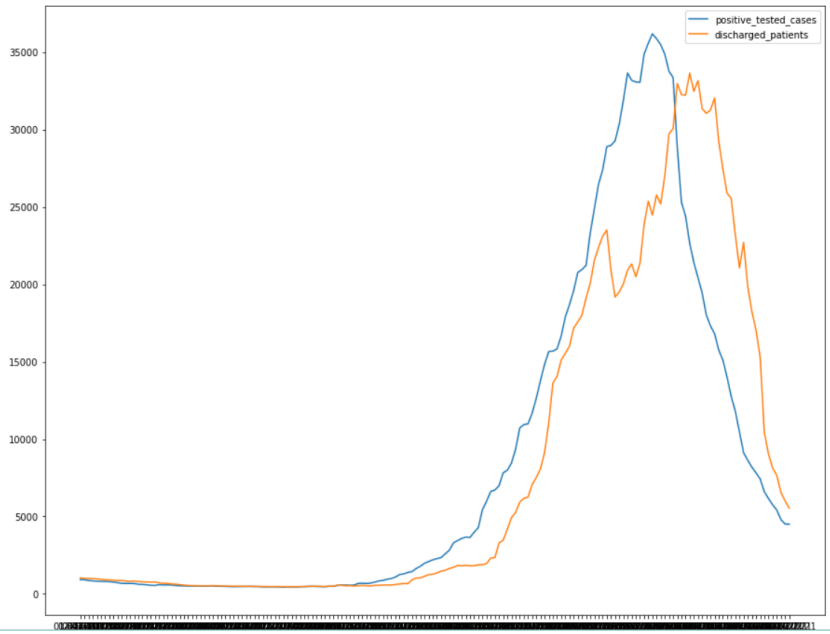
	date	active_cases_yesterday	positive_tested_cases	discharged_patients	deaths_today	active_cases_today	month	year	rise in cases
1	24-07-2020	52939	6785	6504	88	53132	July2020	24	193.0
2	25-07-2020	53132	6988	7758	89	52273	July2020	25	-859.0
3	26-07-2020	52273	6986	5471	85	53703	July2020	26	1430.0
4	28-07-2020	54896	6972	4707	88	57073	July2020	28	3370.0
5	29-07-2020	57073	6426	5927	82	57490	July2020	29	417.0
...	...	...	...	...	...	...	...	...	...
490	01-01-2022	7470	1489	611	8	8340	January2022	01	870.0
491	02-01-2022	8340	1594	624	6	9304	January2022	02	964.0
492	03-01-2022	9304	1728	662	6	10364	January2022	03	1060.0
493	04-01-2022	10364	2731	674	9	12412	January2022	04	2048.0
494	05-01-2022	12412	4862	688	9	16577	January2022	05	4165.0

494 rows × 9 columns

In [10]:
```
plt.rcParams['figure.figsize'] = [15,12]
plt.xlabel("date")
plt.plot(TMcaseinfo["date"][list(TMcaseinfo["date"]).index("01-01-2021"):list(TMcaseinfo["date"]).index("01-07-2021")],TMcaseinfo["positive_tested_c
plt.plot(TMcaseinfo["date"][list(TMcaseinfo["date"]).index("01-01-2021"):list(TMcaseinfo["date"]).index("01-07-2021")],TMcaseinfo["discharged_patien
plt.legend(["positive_tested_cases","discharged_patients","deaths_today"])
```

Out[10]:
```
<matplotlib.legend.Legend at 0x22b5d132f50>
```





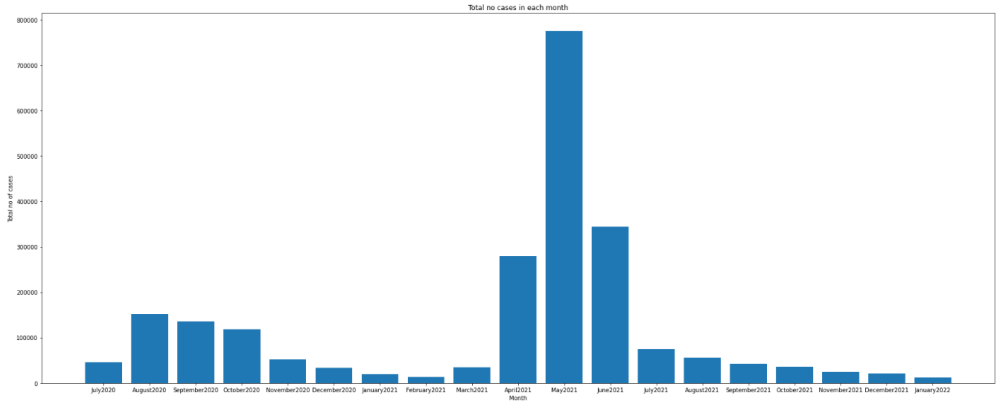
In [11]:
```
TMcaseinfosum=TMcaseinfo.groupby("month").agg("sum").reset_index()
TMcaseinfosum["index"]=TMcaseinfosum["month"].copy()
TMcaseinfosum["index"]=TMcaseinfosum["index"].apply(lambda x: set_index(x))
TMcaseinfosum.sort_values(by="index",inplace=True)
TMcaseinfosum
```

Out[11]:

	month	active_cases_yesterday	positive_tested_cases	discharged_patients	deaths_today	active_cases_today	rise in cases	index
8	July2020	385765	45902	41440	626	389601	5029.0	-1
1	August2020	1401451	152517	154016	2828	1397124	-5390.0	0
17	September2020	1154573	135315	138515	1737	1149636	-6315.0	1
15	October2020	1126874	118592	139894	1496	1104076	-24099.0	2
13	November2020	435049	52252	61500	557	425244	-11112.0	3
3	December2020	287947	33527	35519	383	285572	-2551.0	4
6	January2021	190682	19777	23326	225	186908	-3947.0	5
5	February2021	119462	13202	13594	140	118930	-532.0	6
11	March2021	210443	35131	23051	223	222300	11857.0	7
0	April2021	1789019	280083	179507	1327	1888268	99249.0	8
12	May2021	5172933	775823	590652	7971	5350133	190418.0	9
10	June2021	4010465	344645	588407	7612	3759091	-267355.0	10
9	July2021	853948	75170	90072	1392	837654	-17475.0	11
2	August2021	616606	55557	59131	843	612189	-3866.0	12
18	September2021	433394	42333	41484	571	433672	300.0	13
16	October2021	429622	36251	41073	506	424294	-5658.0	14
14	November2021	288868	24294	27177	365	285620	-3248.0	15
4	December2021	223359	20449	20914	284	222610	-774.0	16
7	January2022	47890	12404	3259	38	56997	9107.0	17

DESCRRIPTION: The below graph plots total no of positives cases per every month from July 2020 to January 2022

In [12]:
```
plt.rcParams['figure.figsize'] = [30,12]
bar_graph(TMcaseinfosum["month"],TMcaseinfosum["positive_tested_cases"],"Month","Total no of cases","Total no cases in each month")
```



Conclusions from The above graph: As we can see the highest total no of cases in a month can be seen in april2021, May2021, June2021

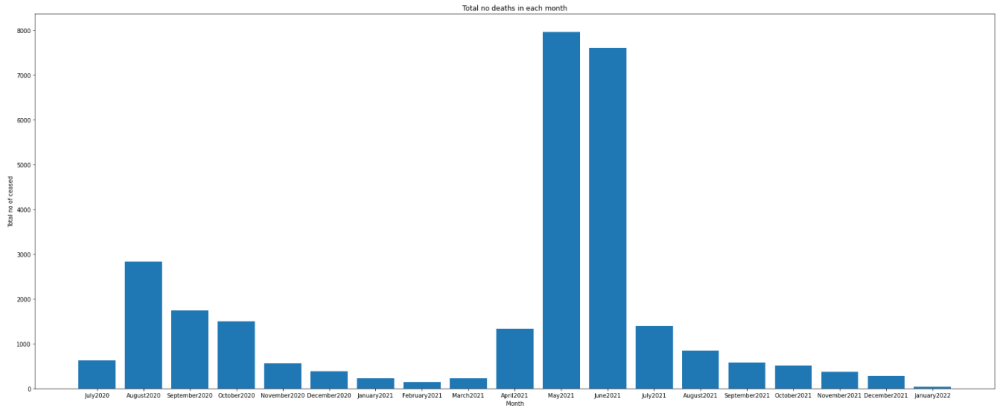
Graph5

The below graph plots total no of deaths per every month from July 2020 to January 2022

Graph5

The below graph plots total no of deaths per every month from July 2020 to January 2022

In [13]:
```
plt.rcParams['figure.figsize'] = [30,12]
bar_graph(TMcaseinfosum["month"],TMcaseinfosum["deaths_today"],"Month","Total no of ceased","Total no deaths in each month")
```

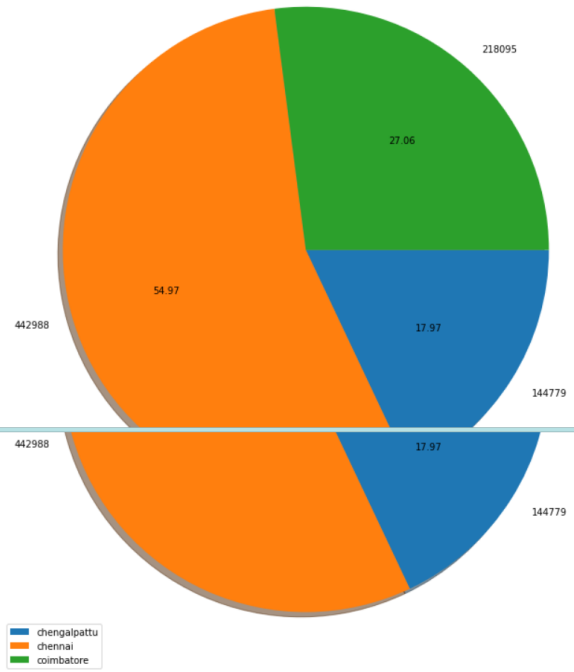


DESCRRIPTION: Conclusions from The above graph: As we can see the highest no of deaths in a month can be seen in May2021, June2021

Graph6

The below piechart plots the states that contributed more than 3% to the total no. covid cases of the state

In [15]:
```
piechart(dis,sumdis,"")
```



Pre-requisites:

1. The data on which analysis was done starts from July 2020 uptill January 2022 in most cases.

Summary of Part-1 Analysis:

PHASE-1: JULY2020 to DECEMBER2020:

1) From Graph1 and Graph2 we can observe that the no of covid cases are decreasing from **July2020** to **December2020**. One primary reason the govrnement could acheive that because the lockdown was implemented strictly till **December2020** source: https://www.thehindubusinessline.com/news/tn-government-extends-covid-19-lockdown-till-january-31/article33465521.ece

PHASE2: JANUARY-2021 to SEPTEMBER-2021:

1) From graph1 ,graph3 and graph4 we can see that the no of cases were less but we can observe that no. of cases started increasingly drastically from **april2021** till **july2021**. There can be many factors to it but few main reasons can be the Tamil Nadu Legislative Assembly election, 2021 conducted between **6-April-2021** to **2-May-**

PART2 Analysis

In [16]:
```
TMpositive["day"]=TMpositive.date.copy()
TMpositive["day"]=TMpositive["day"].apply(lambda x: get_day(x))
```

In [17]:
```
TMpositive
```

Out[17]:

	date	active_cases	tested_positive_today	tested_positive_till_date	rtpcr_samples_tested_today	rtpcr_samples_tested_till_date	persons_tested_rt_pcr_today	persons_tested_rt_pc
0	09-08-2020	53336	5974	296901	70186	3225805	68179	
1	10-08-2020	53099	5879	302815	67153	3292958	65141	
2	11-08-2020	52810	5814	308649	67492	3360450	65490	
3	13-08-2020	53499	5810	320355	67275	3499300	65560	
4	14-08-2020	53716	5862	326245	70153	3569453	68301	
...	...	...	...	...	...	...	...	
474	01-01-2022	8340	1470	2749534	103607	57547850	103403	
475	02-01-2022	9304	1575	2751128	102237	57650087	102029	
476	03-01-2022	10264	1727	2752856	102110	57752206	102008	
477	04-01-2022	12412	2683	2755587	103798	57857004	103573	
478	05-01-2022	16577	4824	2760449	117611	57974615	117382	

479 rows × 21 columns

In [18]:
```
TMpositiveday=TMpositive.groupby("day")
TMpositiveday.first()
```

Out[18]:

day	date	active_cases	tested_positive_today	tested_positive_till_date	rtpcr_samples_tested_today	rtpcr_samples_tested_till_date	persons_tested_rt_pcr_today	persons_test
Friday	14-08-2020	53716	5862	326245	70153	3569453	68301	
Monday	10-08-2020	53099	5879	302815	67153	3292958	65141	
Saturday	15-08-2020	54213	5830	332105	71343	3640796	69598	
Sunday	09-08-2020	53336	5974	296901	70186	3225805	68179	
Thursday	13-08-2020	53499	5810	320355	67275	3499300	65560	
Tuesday	11-08-2020	52810	5814	308649	67492	3360450	65490	

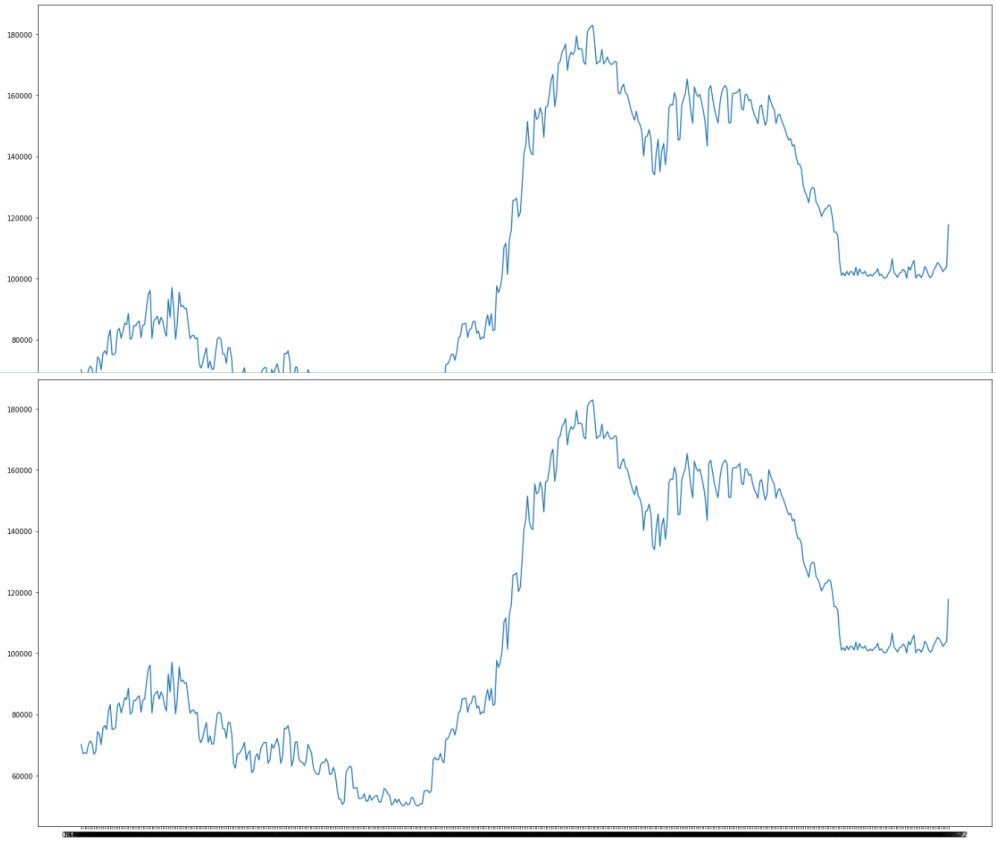
Wednesday	19-08-2020	53155	5785	355449	67720	3913523	65592

Graph7

The graph below depicts no. of tests conducted per day

In [19]:
```
plt.rcParams['figure.figsize'] = [25,12]
plt.plot(TMpositive["date"],TMpositive["rtpcr_samples_tested_today"])
```

Out[19]: [<matplotlib.lines.Line2D at 0x22b5d4b0550>]



Description: We can observe that the no. of cases increased drastically during second-wave. One major reason can be that govt. decided to test even symptomatic as well as asymptomatic, primary contacts, secondary contacts.

In [20]:
```
TMpositive["TPR"]=TMpositive.apply(lambda x: TPR(x["rtpcr_samples_tested_today"],x["tested_positive_today"]),axis=1)
TMpositive
```

Out[20]:

	date	active_cases	tested_positive_today	tested_positive_till_date	rtpcr_samples_tested_today	rtpcr_samples_tested_till_date	persons_tested_rt_pcr_today	persons_tested_rt_pc
0	09-08-2020	53336	5974	296901	70186	3225805	68179	
1	10-08-2020	53099	5879	302815	67153	3292958	65141	
2	11-08-2020	52810	5814	308649	67492	3360450	65490	
3	13-08-2020	53499	5810	320355	67275	3499300	65560	
4	14-08-2020	53716	5862	326245	70153	3569453	68301	
...	...	...	...	...	...	...	...	
474	01-01-2022	8340	1470	2749534	103607	57547850	103403	
475	02-01-2022	9304	1575	2751128	102237	57650087	102029	
476	03-01-2022	10364	1727	2752856	103119	57753206	102898	
477	04-01-2022	12412	2683	2755587	103798	57857004	103573	
	2022							
477	04-01-2022	12412	2683	2755587	103798	57857004	103573	
478	05-01-2022	16577	4824	2760449	117611	57974615	117382	

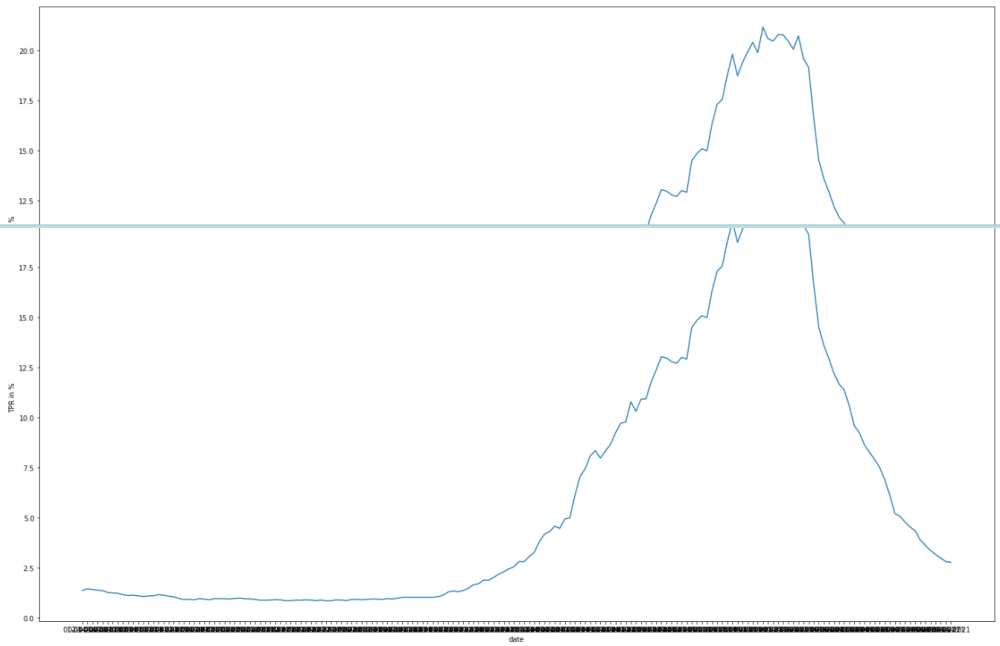
479 rows × 22 columns

Graph8

The graph below plots the TPR(Test positivity Rate)

In [21]:
```
plt.rcParams['figure.figsize'] = [25,12]
plt.xlabel("date")
plt.ylabel("TPR in %")
plt.plot(TMpositive["date"][list(TMpositive["date"]).index("01-01-2021"):list(TMpositive["date"]).index("01-07-2021")],TMpositive["TPR"][list(TMposi
```

Out[21]: [<matplotlib.lines.Line2D at 0x22b5d384e50>]



Description: From the above graph, we can infer that the TPR(Test Positivity Rate) drastically increased during second-wave. The highest TPR recorded in the state was 21.16%.

PART3: AFFECT OF COMORBIDITY ON COVID PATIENT

In [22]:
```
TNindividual
```

Out[22]:

	date	case_id	category	age	gender	location	comorbidity	test_date	test_details	admission_location	admission_date	admission_time	admission_symptoms
0	27-05-2021	22007	Government Health Facilities (DME) Due To Como...	65	Female	Vellore	NaN	23-05-2021	COVID test positive result	Vellore Medical College	22.0502021	09.20PM	
1	26-05-2021	21752	Private health facilities due to comorbidities	59	Female	Madurai	with DM	21-04-2021	COVID-19 RTPCR Positivity	private hospital, Madurai	30.042021	12:00 AM	
2	26-05-2021	21572	Government health facilities (DMS) without com...	62	Male	Virudhunagar	NaN	15-05-2021	COVID-19 RTPCR Positivity	Government Hospital, Sattur, Virudhunagar	1705.2021	05.00 PM	
3	27-05-2021	22225	Private Health Facilities Due To Comorbidities	37	Female	Vellore	with SHTN	11-05-2021	COVID-19 RTPCR Positivity	private hospital,Vellore	10-05-2001	10.00 AM	
4	08-05-2021	15289	Private health facilities without comorbidities	64	Famale	Tirunelveli	NaN	02-05-2021	COVID-19 RTPCR Positivity	private hospital, Tirunelveli, Tirunelveli	03-05-2011	5.21.PM	
...	...	...	...	...	...	...	...	...	...	...	...	...	
	2021		Institutions like Rail...					2021	fever				
28957	07-10-2021	35734	Other facilities (Other Institutions like Rail...	80	male	Chennai	NaN	19-10-2020	symptoms of fever, cough and body pain	NaN	NaN	NaN	
28958	03-11-2021	360176	Other facilities (Other Institutions like Rail...	86	female	Chennai	NaN	23-10-2021	symptoms of fever and breathlessness	NaN	NaN	NaN	
28959	08-11-2021	36201	Government health facilities (DME) due to como...	58	male	Coimbatore died on 04.11.2021 at 08.00 AM due ...	NaN	NaN	NaN	NaN	NaN	NaN	
28960	05-01-2022	36812	Government health facilities (DME) due to como...	77	Female	Krishnagiri	with Systemic Hypertension brought dead on 01....	NaN	NaN	NaN	NaN	NaN	

28961 rows × 18 columns

In [23]:
```
TNindividual["comorbidity"]=TNindividual["comorbidity"].fillna("no comorbidity")
TNindividual["comorb"]=TNindividual.comorbidity.copy()
TNindividual["comorb"]=TNindividual["comorb"].apply(lambda x: comorb(x))
TNindividual["comorb"]
```

Out[23]:
```
0        no comorbidity
1           comorbidity
2        no comorbidity
3           comorbidity
4        no comorbidity
              ...
```

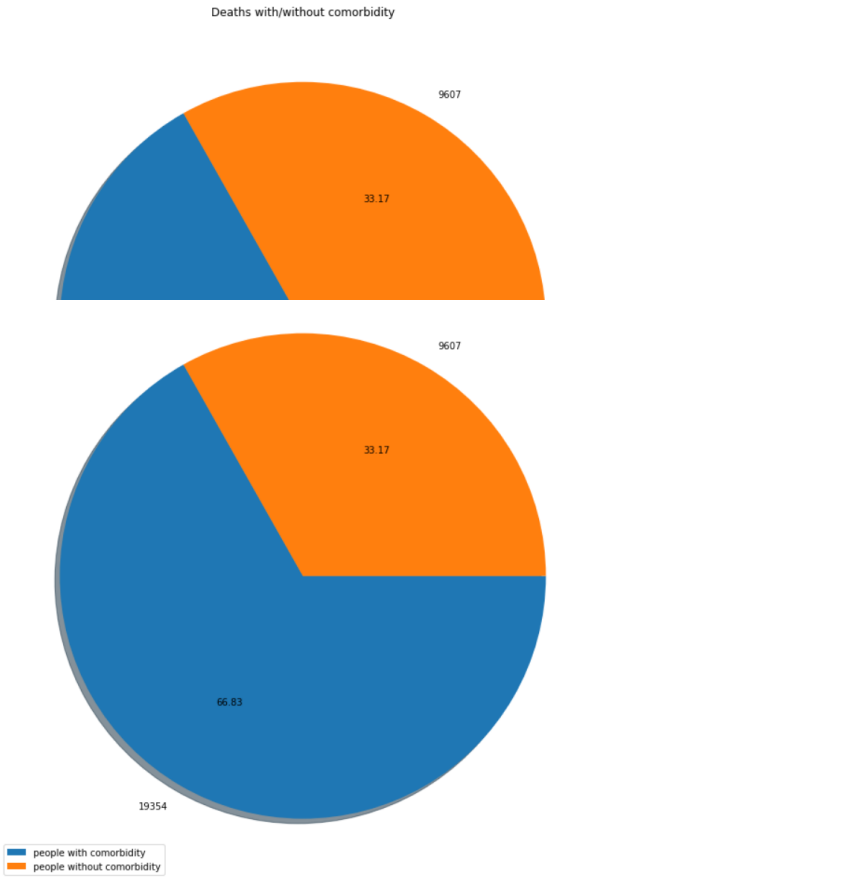
```
1           comorbidity
2        no comorbidity
3           comorbidity
4        no comorbidity
              ...
28956    no comorbidity
28957    no comorbidity
28958    no comorbidity
28959    no comorbidity
28960       comorbidity
Name: comorb, Length: 28961, dtype: object
```

Graph9

The below piechart depicts the no. of deaths with/without comorbidity

In [24]:

```
TNindividualgrp=list(TNindividual.groupby("comorb").size())
piechart(["people with comorbidity","people without comorbidity"],TNindividualgrp,"Deaths with/without comorbidity")
```



DESCRRIPTION: From Graph9 we can observe that the no of deaths in people with comobidities is higher than people without comorbidities

In [25]:
```
TNindividual["category"]=TNindividual["category"].apply(lambda x: cato(x))
TNindividualsi=(TNindividual.groupby("category").size())
```

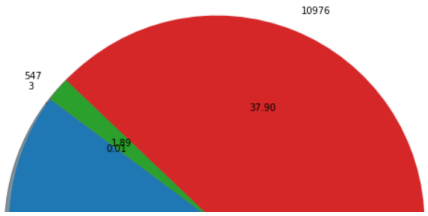
In [26]:
```
TNindividualsi
```

Out[26]:
```
category
Government facility    17435
Medical college            3
Others                   547
Private facility       10976
dtype: int64
```

Graph10

The below piechart depicts the no. of deaths in govt. vs private hospitals

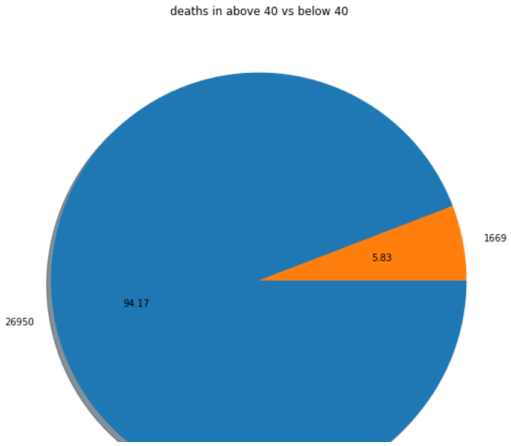
In [27]:
```
TNindividualsi=list(TNindividual.groupby("category").size())
piechart(["Government facility","Private facility"],TNindividualsi)
```

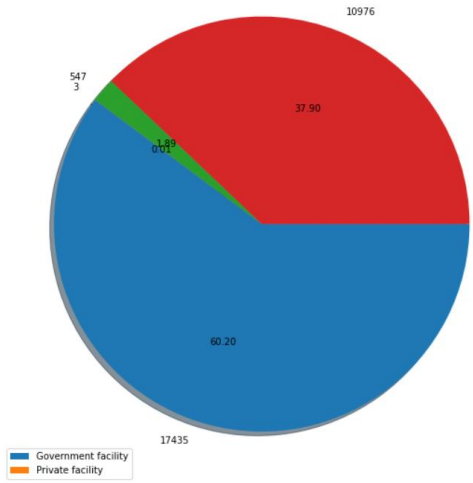


Graph11

The below piechart depicts the no. of deaths in above 40 age group vs 40 group comorbidity

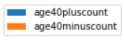
In [28]:
```
TNindividual.sort_values("age",inplace=True)
age40mins=TNindividual.query("age<40")
age40plus=TNindividual.query("age>40")
age40pluscount=age40plus["age"].count()
age40minuscount=age40mins["age"].count()
piechart(["age40pluscount","age40minuscount"],[age40pluscount,age40minuscount],"deaths in above 40 vs below 40")
```





DESCRRIPTION: From Graph 10 we can say that deaths in govt. are more compared to deaths in private hospitals

Graph11

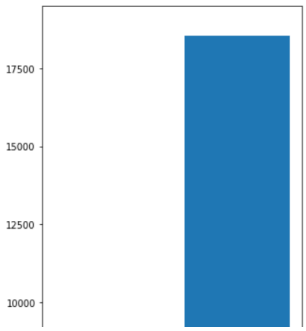


DESCRRIPTION: From Graph11 we can infer that the no. deaths in above 40 age group is very high compared to below 40 age group

Graph12

The below bargraph depicts the no. of deaths in 40+ age group with vs without comorbodities

In [29]:
```
plt.rcParams['figure.figsize'] = [5,12]
age40nocomorbidity=TNindividual.query("age>40 and comorb=='no comorbidity'")
age40comorbidity=TNindividual.query("age>40 and comorb=='comorbidity'")
age40nocomorbiditycount=age40nocomorbidity["age"].count()
age40comorbiditycount=age40comorbidity["age"].count()
bar_graph(["age40nocomorbidity","age40comorbidity"],[age40nocomorbiditycount,age40comorbiditycount]," ","","")
```




17500
15000
12500
10000
7500
5000
2500
0
age40nocomorbidity
age40comorbidity

CONCLUSION AND FUTURE SCOPE

Festivals, elections, mass gathering were some of the major reasons contributed to massive widespread virus from April-2021 to June-2021 in most states. But it was also seen that on the same reasons even some states like Kerala witnessed increase in new cases till December-2021.

Poor medical infrastructure and less vaccination added to the misery of common people.

1. In Tamil Nadu number of covid cases decreased from July-2020 to December-2020 as seen in **Fig-12**. One primary reason the government could achieve that because the lockdown was implemented strictly till December-2020.

To see full sized image, Click Here

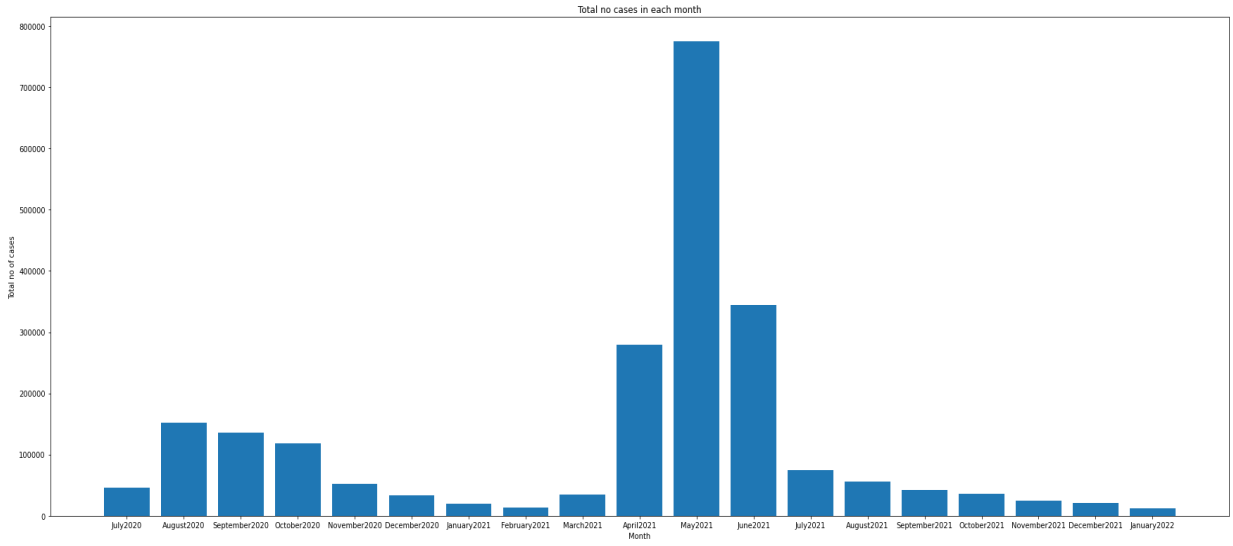


Fig-12: The graph plots total no of positives cases per month in Tamil Nadu.

Vaccination indeed worked as an antidote to reduce hospitalisation and thus fatality rate in many states. Strict implementation of lockdown also helped in Tamil Nadu to control the positive cases for much longer period whereas in West-Bengal as soon as lockdown was lifted, we can see rise in positive cases.

To see full sized image, **Click Here**

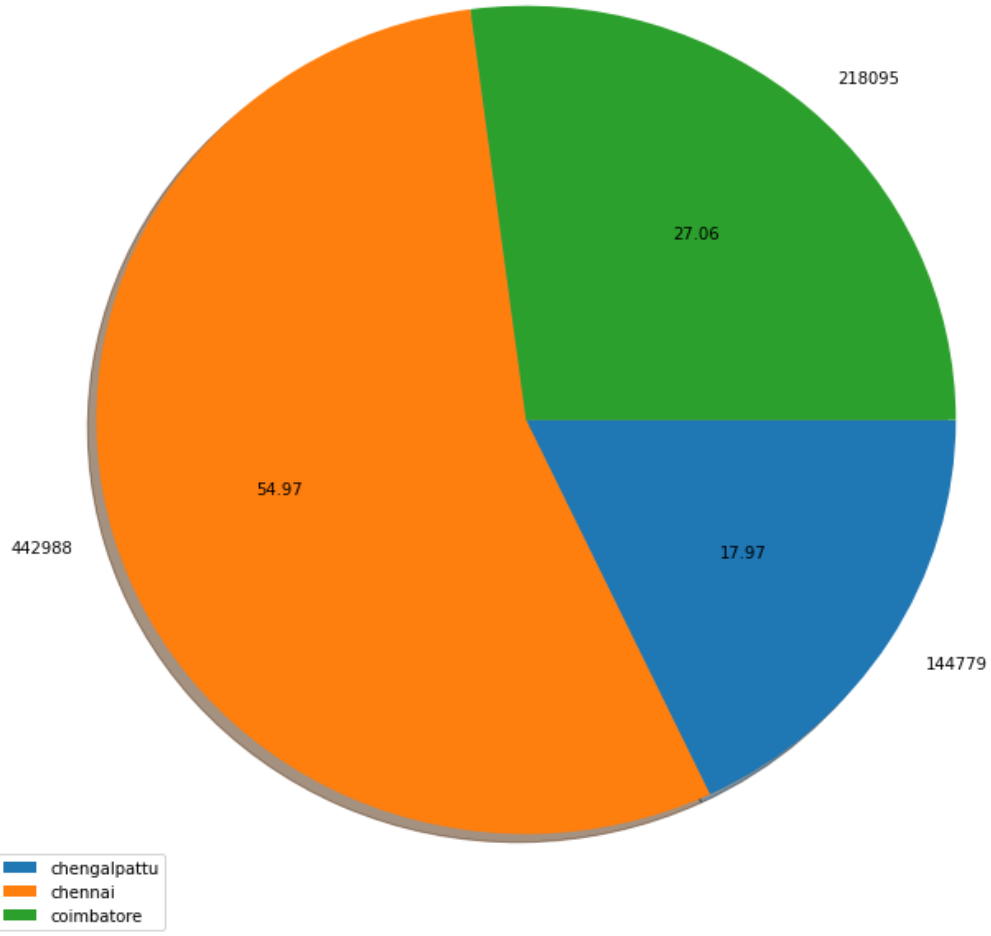


Fig-15: The pie chart **shows major cities contribution to total cases** in Tamil Nadu.

Objective-2: Trend in Fatalities and different comorbidities involved:

In Tamil Nadu it was found that death rate is very high in patients of age 40 above compared to deaths in patients below 40 age group. Again, in patients of age 40 above huge number of deaths can be seen in patients with comorbidities than without patients without comorbidities. (**Check table 5.2.1**)

Table 5.2.1 shows **no. of deaths in 40+ age group with vs without comorbidities.**

Fatalities category	**Number of deaths**
With comorbidity	Nearly 8000
Without comorbidity	Nearly 19000

Table 5.2.1

- **To view bar graph** Click Here

To see full sized image, Click Here

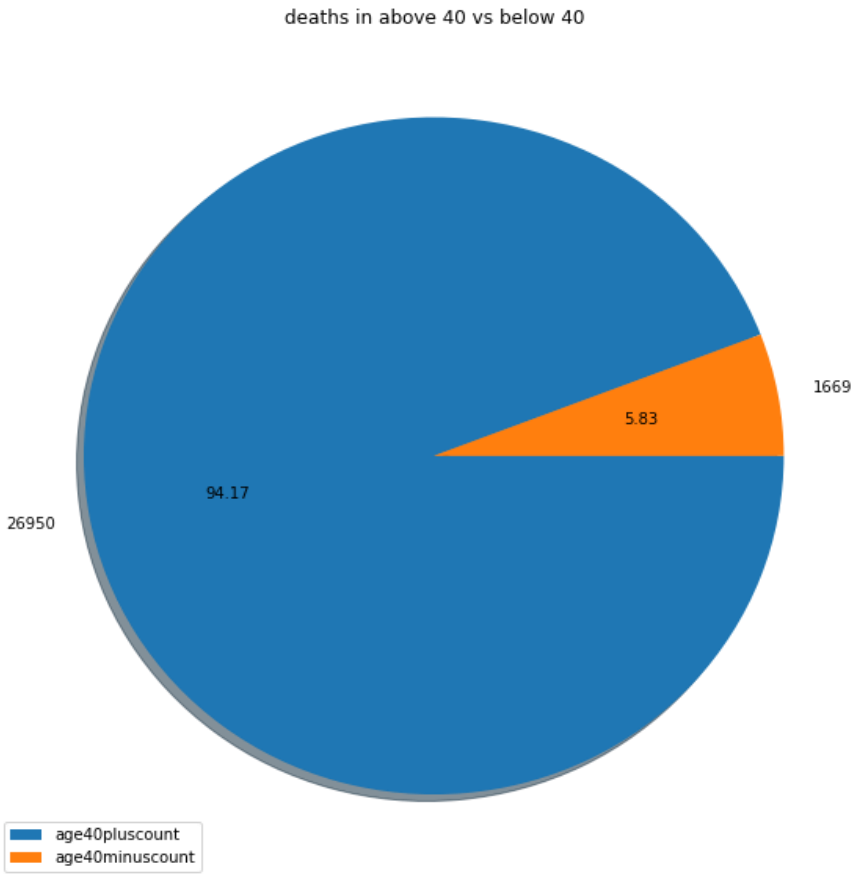


Fig-16: The pie chart depicts **the no. of deaths in 40+ age group vs 40+ age group (with comorbidity)** in Tamil Nadu**.**

FINAL CONCLUSION

1. Festivals, mass gathering in election rallies, not following strict lockdown in states were some of the major reasons for high positive case, hospitalisation and fatality.
2. Poor medical infrastructure and poor rate of vaccination, in the country has caused a lot of discomfort to people and to the medical fraternity in this pandemic.
3. Vaccination played a major role to reduce hospitalisation and fatality.
4. It was identified that all perfect models truly displayed the actual factors behind “total cases” and “total death”. At the same time practical model gave us an idea on how to develop machine learning model which are feasible in real world.
5. We also identified that cities and district with big population were major contributors in positive cases and fatalities in their respective states.
6. With available data in some states, it was found that fatalities with comorbidities were more than fatalities without comorbidities.
7. With available data in some states, fatality was more in the age group 45 and above.

REFERENCES

1. Data sourced from - https://india-covid-19-data.mybluemix.net/