DATA ENGINEERING PROJECT

Author: Giulia Caldato

the project consist in building a dashboard with 2 tiles

In [98]:
```
#!pip install plotly
#!pip install dash
#!pip install python-psycopg2
```

import python libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import dash
#import dash_core_components as dcc
from dash import dcc
#import dash_html_components as html
from dash import html
import flask
import psycopg2
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
import pyspark
import pyspark.sql
from pyspark.sql.functions import *
import findspark
from pyspark.sql import SparkSession
```

START SPARK SESSION FOR BATCH PROCESSING

In [3]:
```
findspark.init()
#
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # format output tables better
spark
```

Out[3]: **SparkSession - in-memory**

SparkContext

Spark UI

Version	`v3.2.1`
Master	`local[*]`
AppName	`pyspark-shell`

Dataset

the dataset chosen is taken from kaggle dataset at link below:

https://www.kaggle.com/datasets/sudalairajkumar/covid19-in-india?select=covid_19_india.csv

Importing the dataset

In [4]:
```
#in pandas
df1 = pd.read_csv('C:\\Users\\nino.caldato\\Desktop\\Data_Eng_Project_000001_GC\\archive\\covid_19_india.csv')
#in spark
df = spark.read.csv('C:\\Users\\nino.caldato\\Desktop\\Data_Eng_Project_000001_GC\\archive\\covid_19_india.csv', inferSchema=True, header=True)
df
```

Out[4]:

Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
3	1/2/2020	6:00 PM	Kerala	2	0	0	0	2
4	2/2/2020	6:00 PM	Kerala	3	0	0	0	3
5	3/2/2020	6:00 PM	Kerala	3	0	0	0	3
6	4/2/2020	6:00 PM	Kerala	3	0	0	0	3
7	5/2/2020	6:00 PM	Kerala	3	0	0	0	3
8	6/2/2020	6:00 PM	Kerala	3	0	0	0	3
9	7/2/2020	6:00 PM	Kerala	3	0	0	0	3
10	8/2/2020	6:00 PM	Kerala	3	0	0	0	3
11	9/2/2020	6:00 PM	Kerala	3	0	0	0	3
12	10/2/2020	6:00 PM	Kerala	3	0	0	0	3
13	11/2/2020	6:00 PM	Kerala	3	0	0	0	3
14	12/2/2020	6:00 PM	Kerala	3	0	0	0	3
15	13/02/20	6:00 PM	Kerala	3	0	0	0	3
16	14/02/20	6:00 PM	Kerala	3	0	0	0	3
17	15/02/20	6:00 PM	Kerala	3	0	0	0	3
18	16/02/20	6:00 PM	Kerala	3	0	0	0	3
19	17/02/20	6:00 PM	Kerala	3	0	0	0	3
20	18/02/20	6:00 PM	Kerala	3	0	0	0	3

only showing top 20 rows

In [5]:
```
df1 = df1.rename(columns = {'State/UnionTerritory':'StateUnionTerritory'})
#df = df.rename(columns = {'State/UnionTerritory':'StateUnionTerritory'})
```

In [6]:
```
# Have column names not separated by spaces instead of /
from pyspark.sql import functions as F

renamed_df = df.select([F.col(col).alias(col.replace('/', '')) for col in df.columns])
renamed_df
```

Out[6]:

Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
3	1/2/2020	6:00 PM	Kerala	2	0	0	0	2
4	2/2/2020	6:00 PM	Kerala	3	0	0	0	3
5	3/2/2020	6:00 PM	Kerala	3	0	0	0	3
6	4/2/2020	6:00 PM	Kerala	3	0	0	0	3
7	5/2/2020	6:00 PM	Kerala	3	0	0	0	3
8	6/2/2020	6:00 PM	Kerala	3	0	0	0	3
9	7/2/2020	6:00 PM	Kerala	3	0	0	0	3
10	8/2/2020	6:00 PM	Kerala	3	0	0	0	3
11	9/2/2020	6:00 PM	Kerala	3	0	0	0	3
12	10/2/2020	6:00 PM	Kerala	3	0	0	0	3
13	11/2/2020	6:00 PM	Kerala	3	0	0	0	3
14	12/2/2020	6:00 PM	Kerala	3	0	0	0	3
15	13/02/20	6:00 PM	Kerala	3	0	0	0	3
16	14/02/20	6:00 PM	Kerala	3	0	0	0	3
17	15/02/20	6:00 PM	Kerala	3	0	0	0	3
18	16/02/20	6:00 PM	Kerala	3	0	0	0	3
19	17/02/20	6:00 PM	Kerala	3	0	0	0	3
20	18/02/20	6:00 PM	Kerala	3	0	0	0	3

only showing top 20 rows

In [7]:
```
# Create a temporary table
renamed_df.createOrReplaceTempView('Data')
```

In [14]:
```
# Read the table using sql command
spark.sql('Select * from Data')
```

Out[14]:

Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
3	1/2/2020	6:00 PM	Kerala	2	0	0	0	2
4	2/2/2020	6:00 PM	Kerala	3	0	0	0	3
5	3/2/2020	6:00 PM	Kerala	3	0	0	0	3
6	4/2/2020	6:00 PM	Kerala	3	0	0	0	3
7	5/2/2020	6:00 PM	Kerala	3	0	0	0	3
8	6/2/2020	6:00 PM	Kerala	3	0	0	0	3
9	7/2/2020	6:00 PM	Kerala	3	0	0	0	3
10	8/2/2020	6:00 PM	Kerala	3	0	0	0	3
11	9/2/2020	6:00 PM	Kerala	3	0	0	0	3
12	10/2/2020	6:00 PM	Kerala	3	0	0	0	3
13	11/2/2020	6:00 PM	Kerala	3	0	0	0	3
14	12/2/2020	6:00 PM	Kerala	3	0	0	0	3
15	13/02/20	6:00 PM	Kerala	3	0	0	0	3
16	14/02/20	6:00 PM	Kerala	3	0	0	0	3
17	15/02/20	6:00 PM	Kerala	3	0	0	0	3
18	16/02/20	6:00 PM	Kerala	3	0	0	0	3
19	17/02/20	6:00 PM	Kerala	3	0	0	0	3
20	18/02/20	6:00 PM	Kerala	3	0	0	0	3

only showing top 20 rows

In [15]:
```
# Total number of states
spark.sql('select count(StateUnionTerritory) from Data')
```

Out[15]:

count(StateUnionTerritory)
3351

In [94]:
```
# Total number of states
spark.sql('select distinct count(StateUnionTerritory) from Data')
```

Out[94]:

count(StateUnionTerritory)
3351

In [16]:
```
# Order by Confirmed number of cases, descending show 5
spark.sql('select * from Data order by Confirmed desc limit 5')
```

Out[16]:

Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
3335	22/06/20	8:00 AM	Maharashtra	-	-	67706	6283	135796
3299	21/06/20	8:00 AM	Maharashtra	-	-	65744	6170	132075
3263	21/06/20	8:00 AM	Maharashtra	-	-	64153	5984	128205
3227	20/06/20	8:00 AM	Maharashtra	-	-	62773	5893	124331
3191	19/06/20	8:00 AM	Maharashtra	-	-	60838	5751	120504

In [17]:
```
# Order by deaths, descending show 5
spark.sql('select * from Data order by Deaths desc limit 5')
```

Out[17]:

Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
3335	22/06/20	8:00 AM	Maharashtra	-	-	67706	6283	135796
3299	21/06/20	8:00 AM	Maharashtra	-	-	65744	6170	132075
3263	21/06/20	8:00 AM	Maharashtra	-	-	64153	5984	128205
3227	20/06/20	8:00 AM	Maharashtra	-	-	62773	5893	124331
3191	19/06/20	8:00 AM	Maharashtra	-	-	60838	5751	120504

In [18]:
```
# Total of Confirmed and dead people
spark.sql('select sum(Confirmed), sum(Deaths) from data')
```

Out[18]:

sum(Confirmed)	sum(Deaths)
10356166	313177

In [19]:
```
## Order by Confirmed cases, ascending show 5, to indicate the safest places
spark.sql('select * from Data order by Confirmed asc limit 5')
```

Out[19]:

Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
982	15/04/20	5:00 PM	Nagaland	-	-	0	0	0
1114	19/04/20	5:00 PM	Nagaland	-	-	0	0	0
1015	16/04/20	5:00 PM	Nagaland	-	-	0	0	0
1048	17/04/20	5:00 PM	Nagaland	-	-	0	0	0
1081	18/04/20	5:00 PM	Nagaland	-	-	0	0	0

EDA IN PANDAS

In [20]:

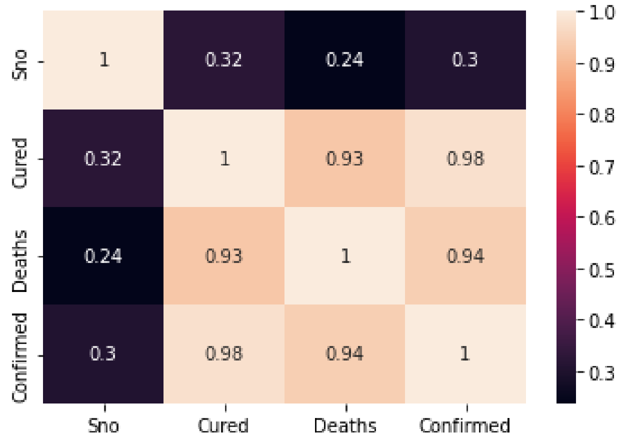
```python
# Correlation
corrMatrix = df1.corr()
corrMatrix
```

Out[20]:

	Sno	Cured	Deaths	Confirmed
Sno	1.000000	0.322516	0.236391	0.304591
Cured	0.322516	1.000000	0.926960	0.977606
Deaths	0.236391	0.926960	1.000000	0.942006
Confirmed	0.304591	0.977606	0.942006	1.000000

In [21]:

```python
sn.heatmap(corrMatrix, annot=True)
plt.show()
```



In [8]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3351 entries, 0 to 3350
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Sno                       3351 non-null   int64 
 1   Date                      3351 non-null   object
 2   Time                      3351 non-null   object
 3   StateUnionTerritory       3351 non-null   object
 4   ConfirmedIndianNational   3351 non-null   object
 5   ConfirmedForeignNational  3351 non-null   object
 6   Cured                     3351 non-null   int64 
 7   Deaths                    3351 non-null   int64 
 8   Confirmed                 3351 non-null   int64 
dtypes: int64(4), object(5)
memory usage: 235.7+ KB
```

In [9]:
```
df1.describe()
```

Out[9]:

	Sno	Cured	Deaths	Confirmed
count	3351.000000	3351.000000	3351.000000	3351.000000
mean	1676.000000	1432.521635	93.457774	3090.470307
std	967.494703	5085.838368	407.541084	10470.065534
min	1.000000	0.000000	0.000000	0.000000
25%	838.500000	1.000000	0.000000	15.000000
50%	1676.000000	33.000000	1.000000	156.000000
75%	2513.500000	568.000000	24.000000	1810.000000
max	3351.000000	67706.000000	6283.000000	135796.000000

Datetime of data records is showed in below dataframe, first 10 rows

In [10]:
```
df1.head(10)
```

Out[10]:

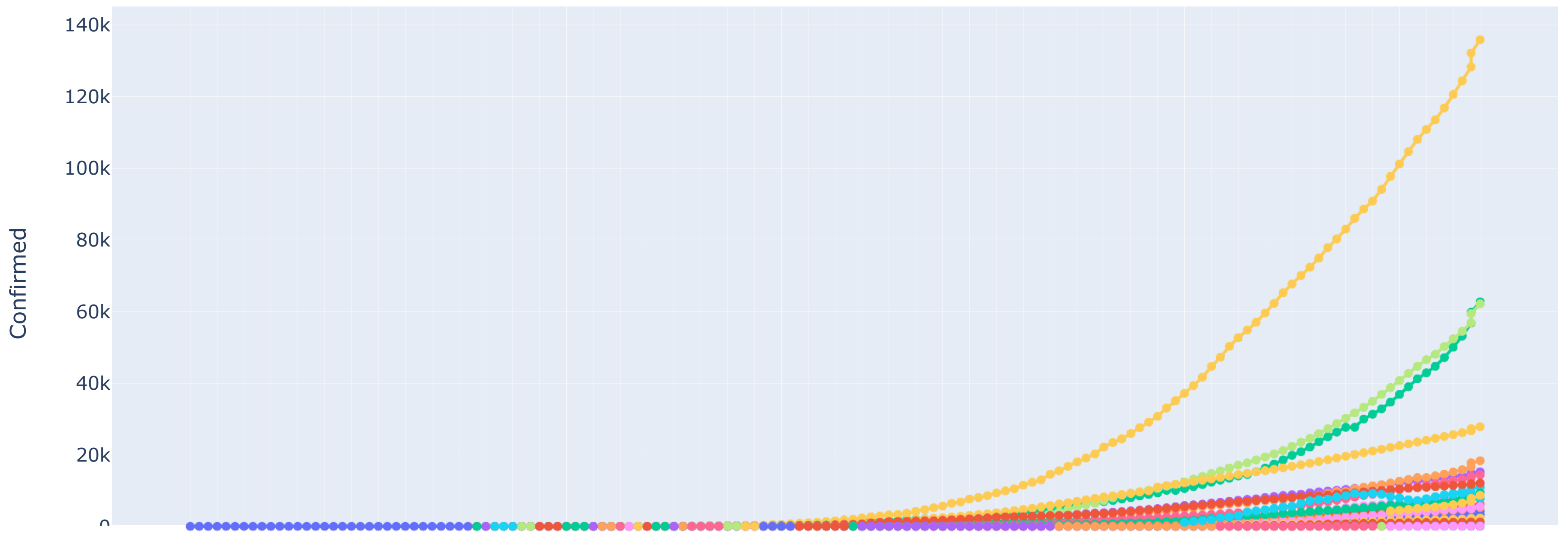
	Sno	Date	Time	StateUnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
0	1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
1	2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
2	3	1/2/2020	6:00 PM	Kerala	2	0	0	0	2
3	4	2/2/2020	6:00 PM	Kerala	3	0	0	0	3
4	5	3/2/2020	6:00 PM	Kerala	3	0	0	0	3
5	6	4/2/2020	6:00 PM	Kerala	3	0	0	0	3
6	7	5/2/2020	6:00 PM	Kerala	3	0	0	0	3
7	8	6/2/2020	6:00 PM	Kerala	3	0	0	0	3
8	9	7/2/2020	6:00 PM	Kerala	3	0	0	0	3
9	10	8/2/2020	6:00 PM	Kerala	3	0	0	0	3

In [11]:
```
df1.isnull().sum()
```

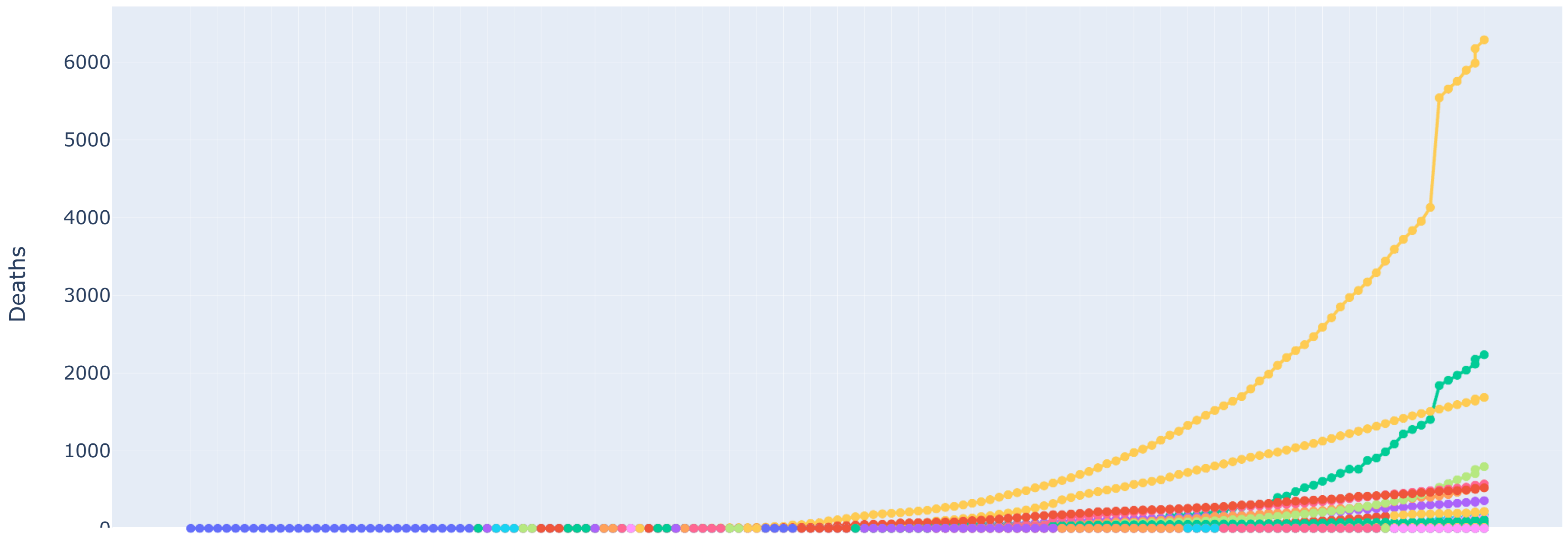
Out[11]:
```
Sno                         0
Date                        0
Time                        0
StateUnionTerritory         0
ConfirmedIndianNational     0
ConfirmedForeignNational    0
Cured                       0
Deaths                      0
Confirmed                   0
dtype: int64
```

PLOTS

In [12]:
```
fig1 = px.scatter(df1, x = 'Date', y = 'Confirmed', color = 'StateUnionTerritory')
fig1.update_traces(mode = 'markers+lines')
fig1.show()
```

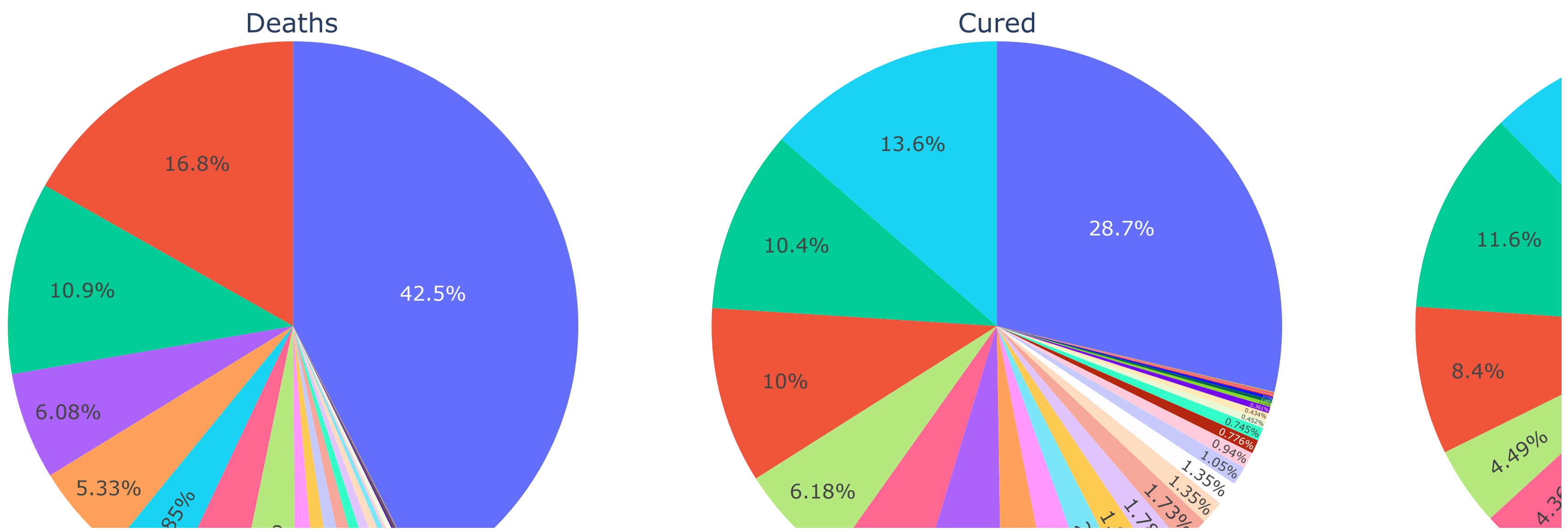


In [13]:
```
fig2 = px.scatter(df1, x = 'Date', y = 'Deaths', color = 'StateUnionTerritory')
fig2.update_traces(mode = 'markers+lines')
fig2.show()
```



In [14]:
```
# pie Charts
fig3 = make_subplots(rows=1, cols=3,
                     subplot_titles=['Deaths', 'Cured', 'Confirmed'],
                     specs=[[{'type':'domain'}, {'type':'domain'},{'type':'domain'}]])

deaths = go.Pie(values=df1['Deaths'], name='Deaths', labels=df1['StateUnionTerritory'])
cured = go.Pie(values=df1['Cured'], name='Cured', labels=df1['StateUnionTerritory'])
confirmed = go.Pie(values=df1['Confirmed'], name='Confirmed', labels=df1['StateUnionTerritory'])

fig3.add_trace(deaths, 1, 1)
fig3.add_trace(cured, 1, 2)
fig3.add_trace(confirmed, 1, 3)

fig3.update_traces(hoverinfo='percent+label')
```

```
fig3.update_layout(showlegend=False)

fig3.update_traces(textposition='inside')

fig3 = go.Figure(fig3)
fig3.show()
```



```
In [15]: external_stylesheets = ['https://www.w3schools.com/w3css/4/w3.css']
```

in Git Bash starts pgcli to use PostgreSQL 14.3 with user postgres

```
In [32]: #In Git Bash
         #nino.caldato@nino-PC MINGW64 ~
         #$ pgcli -U postgres --password postgres
```

```
#Password for postgres: postgres
#Server: PostgreSQL 14.3
#Version: 3.4.1
#Home: http://pgcli.com
#postgres@(none):postgres>
```

Next it is used psycopg2 to connect with RELATIONAL DATABASE PostgreSQL 14.3 (server LOCAL)

In PostgreSQL 14.3 it is created a database "covid_19_india_db"

In PostgreSQL 14.3, in database "covid_19_india_db", then it is created a table "covid_19_india_table"

In [33]:
```
#conn = psycopg2.connect(database="covid_19_india_db", user = "postgres", password = "postgres", host = "127.0.0.1", port = "5432")
conn = psycopg2.connect(user = "postgres", password = "postgres", host = "127.0.0.1", port = "5432")
conn.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT);
print("Opened database successfully")
```

```
Opened database successfully
```

In [34]:
```
cursor          = conn.cursor();
name_Database   = "covid_19_india_db";
```

In [35]:
```
sqlCreateDatabase = "create database "+name_Database+";"
```

In [36]:
```
cursor.execute(sqlCreateDatabase);
```

```
---------------------------------------------------------------------------
DuplicateDatabase                         Traceback (most recent call last)
Input In [36], in <cell line: 1>()
----> 1 cursor.execute(sqlCreateDatabase)

DuplicateDatabase: database "covid_19_india_db" already exists
```

In [37]:
```
conn = psycopg2.connect(database = "covid_19_india_db", user = "postgres", password = "postgres", host = "127.0.0.1", port = "5432")
print("Opened database successfully")

cur = conn.cursor()
cur.execute('''CREATE TABLE covid_19_india_table
      (Sno INT PRIMARY KEY     NOT NULL,
      Date DATE NOT NULL,
      Time TIME NOT NULL,
      StateUnionTerritory CHAR(50),
      ConfirmedIndianNational INT NOT NULL,
      ConfirmedForeignNational INT NOT NULL,
      Cured INT NOT NULL,
      Deaths INT NOT NULL,
      Confirmed INT NOT NULL);''')
print("Table created successfully")

conn.commit()
conn.close()
```

```
Opened database successfully
---------------------------------------------------------------------------
DuplicateTable                            Traceback (most recent call last)
Input In [37], in <cell line: 5>()
      2 print("Opened database successfully")
      4 cur = conn.cursor()
----> 5 cur.execute('''CREATE TABLE covid 19 india_table
      6       (Sno INT PRIMARY KEY     NOT NULL,
      7       Date DATE NOT NULL,
      8       Time TIME NOT NULL,
      9       StateUnionTerritory CHAR(50),
     10       ConfirmedIndianNational INT NOT NULL,
     11       ConfirmedForeignNational INT NOT NULL,
     12       Cured INT NOT NULL,
     13       Deaths INT NOT NULL,
     14       Confirmed INT NOT NULL);''')
     15 print("Table created successfully")
     17 conn.commit()

DuplicateTable: relation "covid_19_india_table" already exists
```

In [16]:

```
app = dash.Dash(__name__, external_stylesheets=external_stylesheets
               )

colors = {
    'background': '#CCFFFF',
    'text': '#FFCC00'
}
```

In [17]:

```
app.layout = html.Div(children = [
    html.H1(children='COVID-19 Dashboard: INDIA'),
    html.Div(children='''COVID-19: Percentage suddivided into State - Union Territory'''),
    dcc.Graph(
        id='example-graph3',
        figure=fig3
    ),
    html.Div(children='''COVID-19: Time Series'''),
     dcc.Graph(
        id='example-graph1',
        figure=fig1
    ),
    dcc.Graph(
        id='example-graph2',
        figure=fig2
    )
     ])
```

In [ ]:
```
if __name__ == '__main__':
    #app.run_server(debug=True)
    #app.run_server(host='127.0.0.1', port=8050, debug=True)
    app.run_server(debug=False)
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
```

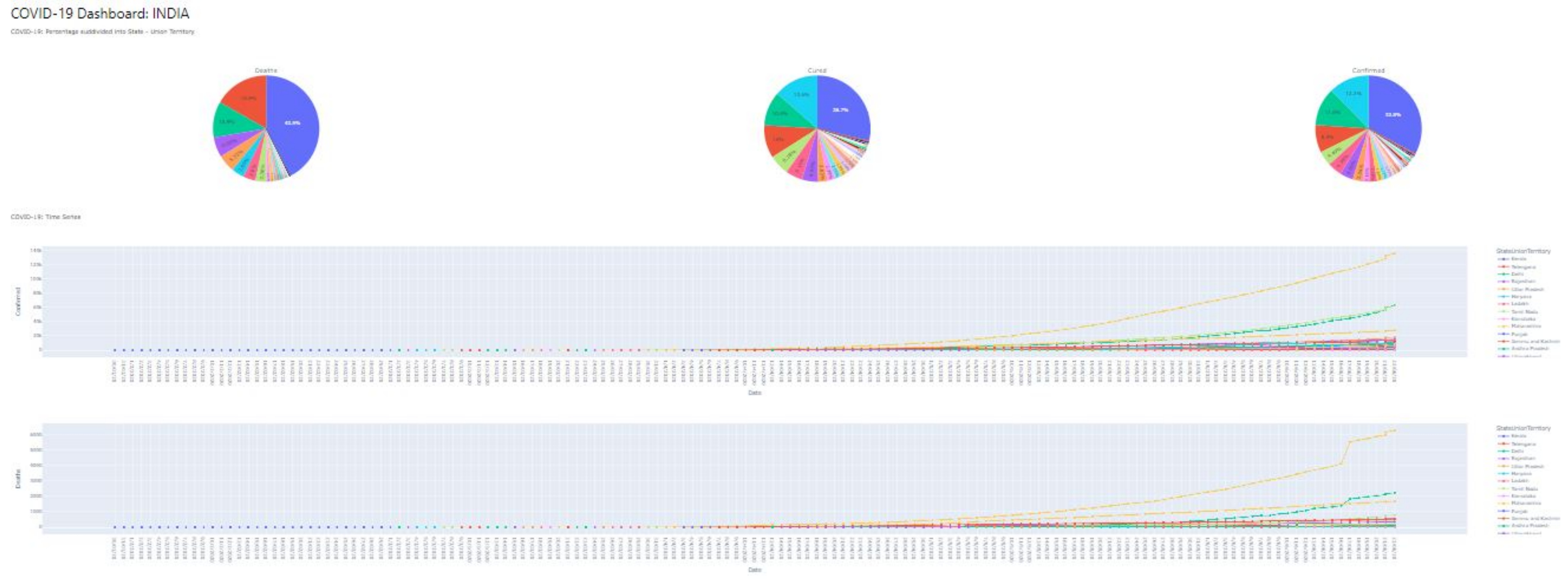
```
 * Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
127.0.0.1 - - [20/Jul/2022 21:43:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:30] "GET /_dash-component-suites/dash/deps/react-dom@16.v2_6_0m1657978484.14.0.min.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:30] "GET /_dash-component-suites/dash/deps/prop-types@15.v2_6_0m1657978483.8.1.min.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:30] "GET /_dash-component-suites/dash/dcc/dash_core_components.v2_6_0m1657978483.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:30] "GET /_dash-component-suites/dash/deps/polyfill@7.v2_6_0m1657978483.12.1.min.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:30] "GET /_dash-component-suites/dash/dash-renderer/build/dash_renderer.v2_6_0m1657978481.min.js HTTP/1.1" 200
 -
127.0.0.1 - - [20/Jul/2022 21:43:31] "GET /_dash-component-suites/dash/deps/react@16.v2_6_0m1657978484.14.0.min.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:32] "GET /_dash-component-suites/dash/dcc/dash_core_components-shared.v2_6_0m1657978483.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:32] "GET /_dash-component-suites/dash/html/dash_html_components.v2_0_4m1657978485.min.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:33] "GET /_dash-component-suites/dash/dash_table/bundle.v5_1_4m1657978482.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:35] "GET /_dash-dependencies HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:36] "GET /_dash-layout HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:36] "GET /_favicon.ico?v=2.6.0 HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:37] "GET /_dash-component-suites/dash/dcc/async-graph.js HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2022 21:43:37] "GET /_dash-component-suites/dash/dcc/async-plotlyjs.js HTTP/1.1" 200 -
```

TO SEE PLOTS BY Dash JUST CLICK IN CELL ABOVE ON THE LINK http://127.0.0.1:8050/

Dashboard consist in :

- 3 pie charts showing distributions of categorical data

- 2 graphs (scatter plots) showing the distribution of the data across a temporal line (daily)


COVID-19 Dashboard: INDIA
COVID-19: Percentage subdivided into State - Union Territory
Deaths
Cured
Confirmed
COVID-19: Time Series
Confirmed
Deaths
Date
StateUnionTerritory
Kerala
Telangana
Delhi
Rajasthan
Uttar Pradesh
Haryana
Ladakh
Tamil Nadu
Karnataka
Maharashtra
Punjab
Jammu and Kashmir
Andhra Pradesh

```
In [ ]:
```