OBJECTIVE

Unemployment is measured by the unemployment rate which is the number of people who are unemployed as a percentage of the total labour force. During the Covid-19 period there was a sharp increase in the unemployment rate. So in this assignment we have to analyze the unemployment rate using Python

Import Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
import calendar
import plotly.graph_objects as go

import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [2]:

```
df = pd.read_csv("Unemployment_Rate_upto_11_2020.csv")
```

In [4]:

```
df.head()
```

Out[4]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude	latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

In [7]:

```
#updating the column names
df.columns=["State","Date","Frequency","Estimated unemployment rate","Estimated employed",
            "Estimated labour participation rate","Region","Longitude","Latitude"]
```

In [8]:

```
df.head()
```

Out[8]:

	State	Date	Frequency	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Region	Longitude	Latitude
0	Andhra Pradesh	31-01-2020	M	5.48	16635535	41.02	South	15.9129	79.74
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.90	South	15.9129	79.74
2	Andhra Pradesh	31-03-2020	M	5.79	15881197	39.18	South	15.9129	79.74
3	Andhra Pradesh	30-04-2020	M	20.51	11336911	33.10	South	15.9129	79.74
4	Andhra Pradesh	31-05-2020	M	17.43	12988845	36.46	South	15.9129	79.74

In [11]:

```
df.shape
```

Out[11]:

```
(267, 9)
```

In [13]:

```
df.columns
```

Out[13]:

```
Index(['State', 'Date', 'Frequency', 'Estimated unemployment rate',
       'Estimated employed', 'Estimated labour participation rate', 'Region',
       'Longitude', 'Latitude'],
      dtype='object')
```

In [14]:

```
df.describe()
```

Out[14]:

	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Longitude	Latitude
count	267.000000	2.670000e+02	267.000000	267.000000	267.000000
mean	12.236929	1.396211e+07	41.681573	22.826048	80.532425
std	10.803283	1.336632e+07	7.845419	6.270731	5.831738
min	0.500000	1.175420e+05	16.770000	10.850500	71.192400
25%	4.845000	2.838930e+06	37.265000	18.112400	76.085600
50%	9.650000	9.732417e+06	40.390000	23.610200	79.019300
75%	16.755000	2.187869e+07	44.055000	27.278400	85.279900
max	75.850000	5.943376e+07	69.690000	33.778200	92.937600

In [15]:

```
df.dtypes
```

Out[15]:

```
State                                     object
Date                                      object
Frequency                                 object
Estimated unemployment rate              float64
Estimated employed                         int64
Estimated labour participation rate      float64
Region                                    object
Longitude                                float64
Latitude                                 float64
dtype: object
```

In [17]:

```
df["Date"]=pd.to_datetime(df["Date"])
```

In [20]:

```
df.dtypes
```

Out[20]:

```
State                                           object
Date                                    datetime64[ns]
Frequency                                       object
Estimated unemployment rate                    float64
Estimated employed                               int64
Estimated labour participation rate            float64
Region                                          object
Longitude                                      float64
Latitude                                       float64
dtype: object
```

In [22]:

```
df.isnull().sum()
```

Out[22]:

```
State                                    0
Date                                     0
Frequency                                0
Estimated unemployment rate              0
Estimated employed                       0
Estimated labour participation rate      0
Region                                   0
Longitude                                0
Latitude                                 0
dtype: int64
```

In [26]:

```
df.duplicated().any()
```

Out[26]:

```
False
```

In [27]:

```
#Converting 'Frequency' and 'Region' columns to categorical data type
df['Frequency'] = df['Frequency'].astype('category')
df['Region'] = df['Region'].astype('category')
```

In [28]:

```
df.dtypes
```

Out[28]:

```
State                                           object
Date                                    datetime64[ns]
Frequency                                     category
Estimated unemployment rate                    float64
Estimated employed                               int64
Estimated labour participation rate            float64
Region                                        category
Longitude                                      float64
Latitude                                       float64
dtype: object
```

In [62]:

```
#extract month

df["month"]=df["Date"].dt.month
```

In [63]:

```
#converting 'month' to integer format
df['Month_int'] = df['month'].apply(lambda x: int(x))

# Mapping integer month values to abbreviated month names
df['Month_name'] = df['Month_int'].apply(lambda x: calendar.month_abbr[x])
```

In [ ]:

```

```

In [69]:

```python
df['Month'] = df['Month_int'].apply(lambda x: calendar.month_abbr[x])
```

In [70]:

```python
df.tail()
```

Out[70]:

	State	Date	Frequency	Estimated unemployment rate	Estimated employed	Estimated labour participation rate	Region	Longitude	Latitude	Month_int	Month_name	month	Month
262	West Bengal	2020-06-30	M	7.29	30726310	40.39	East	22.9868	87.855	6	Jun	Jun	Jun
263	West Bengal	2020-07-31	M	6.83	35372506	46.17	East	22.9868	87.855	7	Jul	Jul	Jul
264	West Bengal	2020-08-31	M	14.87	33298644	47.48	East	22.9868	87.855	8	Aug	Aug	Aug
265	West Bengal	2020-09-30	M	9.35	35707239	47.73	East	22.9868	87.855	9	Sep	Sep	Sep
266	West Bengal	2020-10-31	M	9.98	33962549	45.63	East	22.9868	87.855	10	Oct	Oct	Oct

Exploratory Data Analysis

In [71]:

```python
#Basic Statistics
data_stats = df[['Estimated unemployment rate', 'Estimated employed', 'Estimated labour participation rate']]
round(data_stats.describe().T, 2)
```

Out[71]:

	count	mean	std	min	25%	50%	75%	max
Estimated unemployment rate	267.0	12.24	10.80	0.50	4.84	9.65	16.76	75.85
Estimated employed	267.0	13962105.72	13366318.36	117542.00	2838930.50	9732417.00	21878686.00	59433759.00
Estimated labour participation rate	267.0	41.68	7.85	16.77	37.26	40.39	44.06	69.69

In [72]:

```python
region_stats = df.groupby(['Region'])[['Estimated unemployment rate', 'Estimated employed',
                                       'Estimated labour participation rate']].mean().reset_index()
round(region_stats, 2)
```

Out[72]:

	Region	Estimated unemployment rate	Estimated employed	Estimated labour participation rate
0	East	13.92	19602366.90	40.11
1	North	15.89	13072487.92	38.70
2	Northeast	10.95	3617105.53	52.06
3	South	10.45	14040589.33	40.44
4	West	8.24	18623512.72	41.26

Data Visualization

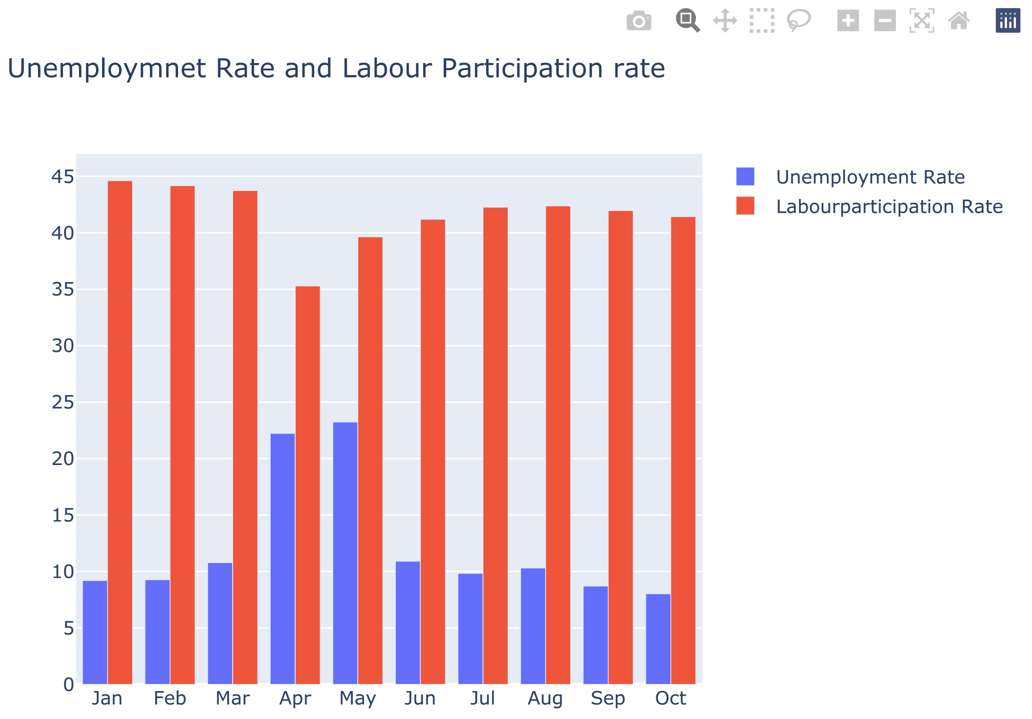
Bar plot of Unemployment rate and Labour participation rate

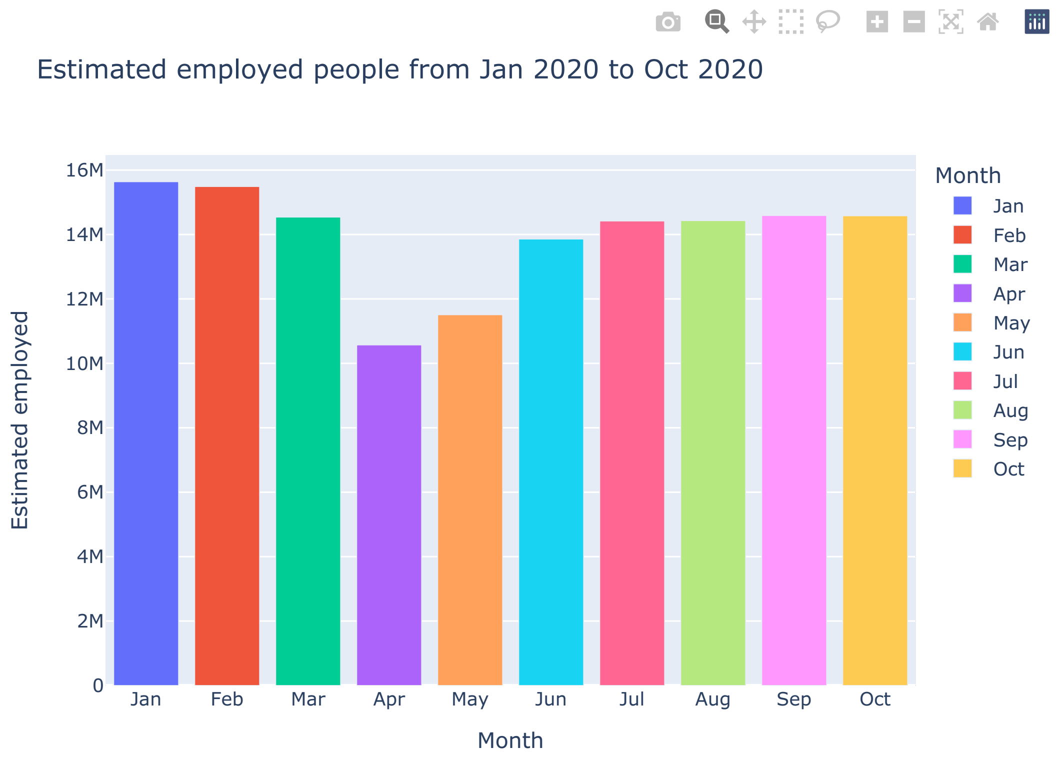
In [75]:

```python
IMD = df.groupby(["Month"])[['Estimated unemployment rate','Estimated employed','Estimated labour participation
IMD = pd.DataFrame(IMD).reset_index()
```

In [76]:

```python
month = IMD.Month
unemployment_rate = IMD["Estimated unemployment rate"]
labour_participation_rate = IMD["Estimated labour participation rate"]

fig = go.Figure()

fig.add_trace(go.Bar(x=month, y= unemployment_rate, name = "Unemployment Rate"))
fig.add_trace(go.Bar(x= month , y = labour_participation_rate, name = "Labourparticipation Rate"))

fig.update_layout(title = "Unemploymnet Rate and Labour Participation rate ",
                 xaxis= {"categoryorder":"array","categoryarray":["Jan","Feb","Mar","Apr","May","Jun","Jul","Au

fig.show()
#Bar plot of estimated employed citizen in every month
import plotly.express as px
fig = px.bar(IMD, x='Month', y='Estimated employed', color='Month',
            category_orders = {"Month":["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct"]},
            title = 'Estimated employed people from Jan 2020 to Oct 2020')

fig.show()
```





State Wise Analysis

In [78]:

```
State = df.groupby("State")[['Estimated unemployment rate','Estimated employed','Estimated labour participation
State = pd.DataFrame(State).reset_index()
```

In [80]:

```
#box Plot

fig = px.box(df,x='State',y='Estimated unemployment rate',color='State',title='Unemployment rate')
fig.update_layout(xaxis={'categoryorder':'total descending'})
fig.show()
```

Average Unemployment rate bar plot

In [81]:

```
fig = px.bar(State, x='State', y='Estimated unemployment rate', color="State",title="Average Unemployment Rate
fig.update_layout(xaxis={'categoryorder':'total descending'})

fig.show()
```

Haryana and Tripura was having the highest average amount of Unemployment Rate

Meghalaya was having the lowest average amount of Unemployment Rate

Correlation Heatmap

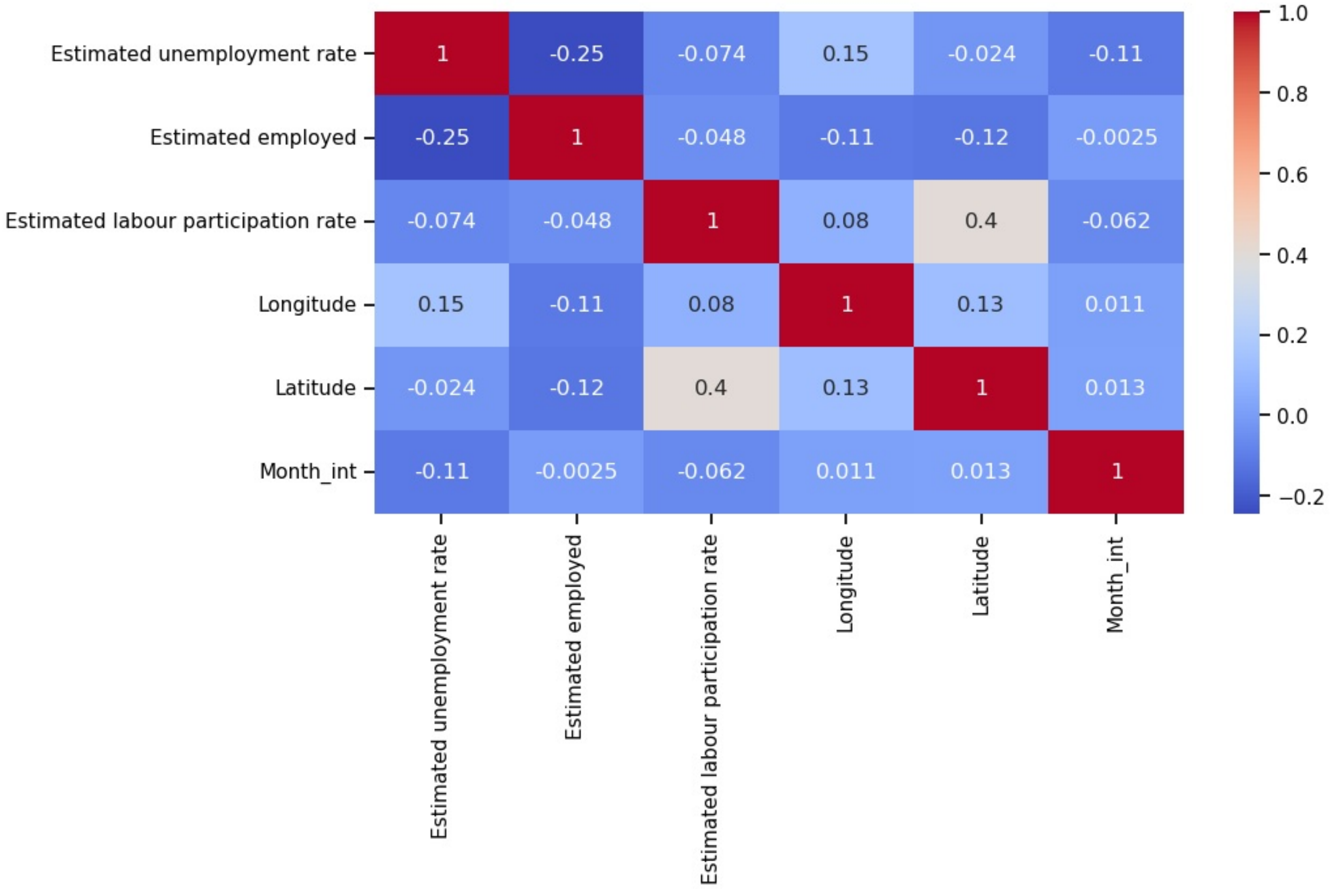
In [84]:

```
heat_maps = df[["Estimated unemployment rate", "Estimated employed","Estimated labour participation rate",'Long
heat_maps = heat_maps.corr()
```

```
plt.figure(figsize=(10,5))
sns.set_context("notebook",font_scale=1)
sns.heatmap(heat_maps,annot=True , cmap='coolwarm')
```

Out[84]:

```
<Axes: >
```



Animated bar plot of Unemployment rate across region from Jan.2020 to Oct.2020

In [86]:

```
fig = px.bar(df, x='Region', y='Estimated unemployment rate', animation_frame='Month_name', color='State',
             title='Unemployment rate across region from Jan.2020 to Oct.2020', height=700, template='plotly')
fig.update_layout(xaxis={'categoryorder': 'total descending'})
fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 2000
fig.show()
```

Sunburst chart

In [88]:

```
# Sunburst chart showing unemployment rate in each region and state

unemplo_df = df[['State', 'Region', 'Estimated unemployment rate', 'Estimated employed', 'Estimated labour part
unemplo = unemplo_df.groupby(['Region', 'State'])['Estimated unemployment rate'].mean().reset_index()
fig = px.sunburst(unemplo, path=['Region', 'State'], values='Estimated unemployment rate',
                  color_continuous_scale='Plasma', title='Unemployment rate in each region and state',
                  height=650, template='ggplot2')
fig.show()
```

Monthly unemployment rate

In [90]:

```
#Impact of Lockdown on States Estimated Employed

fig = px.scatter_geo(df,'Longitude', 'Latitude', color="Region",
                     hover_name="State", size="Estimated unemployment rate",
                     animation_frame="Month_name",scope='asia',template='seaborn',title='Impack of lockdown on

fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 2000

fig.update_geos(lataxis_range=[5,35], lonaxis_range=[65, 100],oceancolor="lightblue",
    showocean=True)

fig.show()
```

Regional Analysis

In [92]:

```
df.Region.unique()
```

Out[92]:

```
['South', 'Northeast', 'East', 'West', 'North']
Categories (5, object): ['East', 'North', 'Northeast', 'South', 'West']
```

Unemployment rate before and after Lockdown

In [95]:

```
#data representation before and after the lockdown

before_lockdown = df[(df['Month_int']>=1) & (df['Month_int']<4)]
after_lockdown =  df[(df['Month_int']>=4) & (df['Month_int']<=6)]
```

In [96]:

```
af_lockdown=after_lockdown.groupby('State')['Estimated unemployment rate'].mean().reset_index()
lockdown= before_lockdown.groupby('State')['Estimated unemployment rate'].mean().reset_index()
lockdown['Unemployment Rate before lockdown'] = af_lockdown['Estimated unemployment rate']

lockdown.columns=['State','Unemployment Rate Before Lockdown','Unemployment Rate After Lockdown']

lockdown.head()
```

Out[96]:

	State	Unemployment Rate Before Lockdown	Unemployment Rate After Lockdown
0	Andhra Pradesh	5.700000	13.750000
1	Assam	4.613333	7.070000
2	Bihar	12.110000	36.806667
3	Chhattisgarh	8.523333	9.380000
4	Delhi	18.036667	25.713333

In [97]:

```
# percentage change in unemployment rate

lockdown['rate change in unemployment'] = round(lockdown['Unemployment Rate After Lockdown'] -lockdown['Unemplo
plot_per = lockdown.sort_values('rate change in unemployment')
```

In [98]:

```
# percentage change in unemployment after lockdown

fig = px.bar(plot_per, x='State',y='rate change in unemployment',color='State',
            title='percentage change in Unemployment in each state after lockdown',template='ggplot2')
fig.show()
```

Most impacted States/Union Territories

Puducherry

Jharkhand

Bihar

Haryana

Tripura

Impact of lockdown on employment across states

In [99]:

```
# function to sort value based on impact

def sort_impact(x):
    if x <= 10:
        return 'impacted States'
    elif x <= 20:
        return 'hard impacted States'
    elif x <= 30:
        return 'harder impacted States'
    elif x <= 46:
        return 'hardest impacted States'
    return x
```

In [100…

```
plot_per['impact status'] = plot_per['rate change in unemployment'].apply(lambda x:sort_impact(x))
```

In [101…

```
fig = px.bar(plot_per, y='State',x='rate change in unemployment',color='impact status',
            title='Impact of lockdown on employment across states',template='ggplot2',height=650)


fig.show()
```

In [ ]:

```
```