UNEMPLOYMENT ANALYSIS

Unemployment is measured by the unemployment rate which is the number of people who are unemployed as a percentage of the total labour force. We have seen a sharp increase in the unemployment rate during Covid-19, so analyzing the unemployment rate can be a good data science project.

Description of the dataset

Region = states in India

Date = date which the unemployment rate observed

Frequency = measuring frequency (Monthly)

Estimated Unemployment Rate (%) = percentage of people unemployed in each States of India

Estimated Employed = The count of people currently employed

Estimated Labour Participation Rate (%) = labour force participation rate by dividing the number of people actively participating in the labour force by the total number of people eligible to participate in the labor force

In [56]:
```
#Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [57]:
```
#Loading the dataset
df = pd.read_csv("C:/Users/Akash K Shaji/Downloads/Unemployment_Rate_upto_11_2020.csv")
df.head()
```

Out[57]:

	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 [58]:
```
#Understanding the structure of the dataset
df.shape
```

Out[58]:
```
(267, 9)
```

The unemployment dataset contains 768 instances and 7 variables.

In [59]:
```
#Renaming the column
df.rename(columns={'Region.1': 'Area'}, inplace=True)
```

Since we have similar names for two columns we replaced Region.1 with Area which is more concise and clear.

In [60]:
```
#Checking for missing values
df.isnull().sum()
```

Out[60]:
```
Region                                  0
 Date                                   0
 Frequency                              0
 Estimated Unemployment Rate (%)        0
 Estimated Employed                     0
 Estimated Labour Participation Rate (%)    0
Area                                    0
longitude                               0
latitude                                0
dtype: int64
```

Null values are absent.

In [63]:
```
#Checking for duplicate values
df.duplicated().sum()
```

Out[63]:
```
0
```

Duplicated values are not present in the dataset.

In [64]:
```
#Summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 267 entries, 0 to 266
Data columns (total 9 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Region                                   267 non-null    object 
 1    Date                                    267 non-null    object 
 2    Frequency                               267 non-null    object 
 3    Estimated Unemployment Rate (%)         267 non-null    float64
 4    Estimated Employed                      267 non-null    int64  
 5    Estimated Labour Participation Rate (%) 267 non-null    float64
 6   Area                                     267 non-null    object 
 7   longitude                                267 non-null    float64
 8   latitude                                 267 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 18.9+ KB
```

We have 5 numerical variables and 4 categorical variables.

Number of non null values in each column is also obtained.

It also provides an estimate of the memory usage of the DataFrame. Here the memory usage is approximately 19 KB.

In [65]:
```
#Removing unintentional spaces in columns
df.columns = df.columns.str.strip()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 267 entries, 0 to 266
Data columns (total 9 columns):
 #   Column                                   Non-Null Count  Dtype  
---  ------                                   --------------  -----  
 0   Region                                   267 non-null    object 
 1   Date                                     267 non-null    object 
 2   Frequency                                267 non-null    object 
 3   Estimated Unemployment Rate (%)          267 non-null    float64
 4   Estimated Employed                       267 non-null    int64  
 5   Estimated Labour Participation Rate (%)  267 non-null    float64
 6   Area                                     267 non-null    object 
 7   longitude                                267 non-null    float64
 8   latitude                                 267 non-null    float64
dtypes: float64(4), int64(1), object(4)
memory usage: 18.9+ KB
```

We need to ensure that the column names are consistent and free of any unwanted spaces, which can help prevent issues when referencing columns during data analysis.

In [66]:
```
#Converting data types
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Out[66]:
```
Region                                           object
Date                                     datetime64[ns]
Frequency                                        object
Estimated Unemployment Rate (%)                 float64
Estimated Employed                                int64
Estimated Labour Participation Rate (%)         float64
Area                                             object
longitude                                       float64
latitude                                        float64
dtype: object
```

Here the column 'Date' was in the object type datatype so we converted it into the datetime format.

In [67]:
```
#Summary Statistics
categorical_var = df.select_dtypes(include='object')
categorical_stat = categorical_var.describe().T
categorical_stat
```

Out[67]:

	count	unique	top	freq
Region	267	27	Andhra Pradesh	10
Frequency	267	1	M	267
Area	267	5	North	79

In [68]:
```
numerical_var = df.select_dtypes(exclude='object')
numerical_stat = numerical_var.describe().T
numerical_stat
```

Out[68]:

	count	mean	std	min	25%	50%	75%	max
Estimated Unemployment Rate (%)	267.0	1.223693e+01	1.080328e+01	0.5000	4.845000e+00	9.650000e+00	1.675500e+01	7.585000e+01
Estimated Employed	267.0	1.396211e+07	1.336632e+07	117542.0000	2.838930e+06	9.732417e+06	2.187869e+07	5.943376e+07
Estimated Labour Participation Rate (%)	267.0	4.168157e+01	7.845419e+00	16.7700	3.726500e+01	4.039000e+01	4.405500e+01	6.969000e+01
longitude	267.0	2.282605e+01	6.270731e+00	10.8505	1.811240e+01	2.361020e+01	2.727840e+01	3.377820e+01
latitude	267.0	8.053242e+01	5.831738e+00	71.1924	7.608560e+01	7.901930e+01	8.527990e+01	9.293760e+01

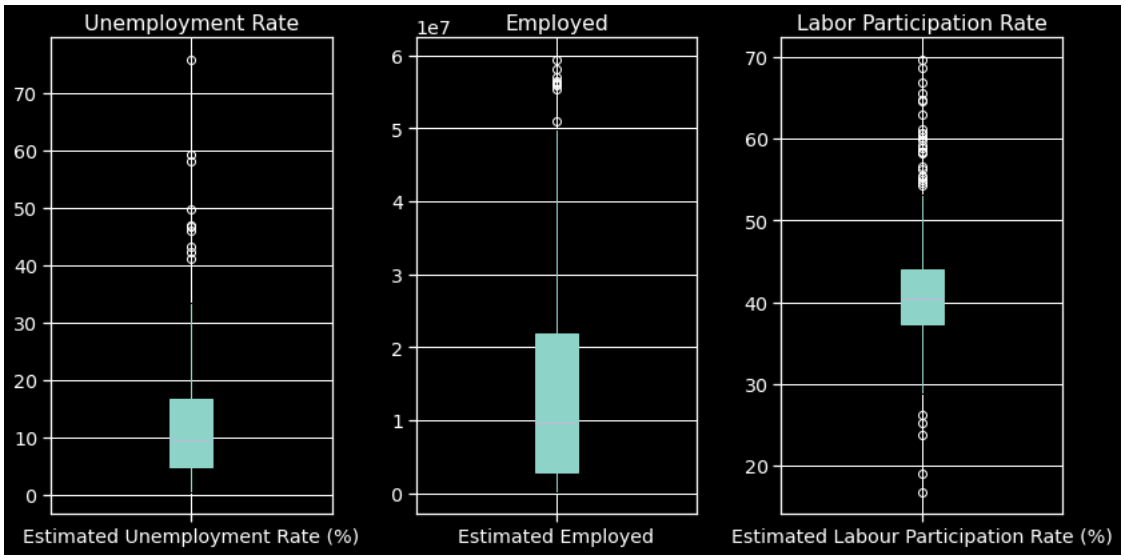
The mean estimated unemployment rate is high. The high unemployment rate could be attributed to the economic disruptions caused by the pandemic, such as lockdowns, reduced economic activity, and job losses in various sectors.

In [69]:
```
#Dropping irrelevent column
df = df.drop('Frequency', axis=1)
df.head()
```

Out[69]:

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

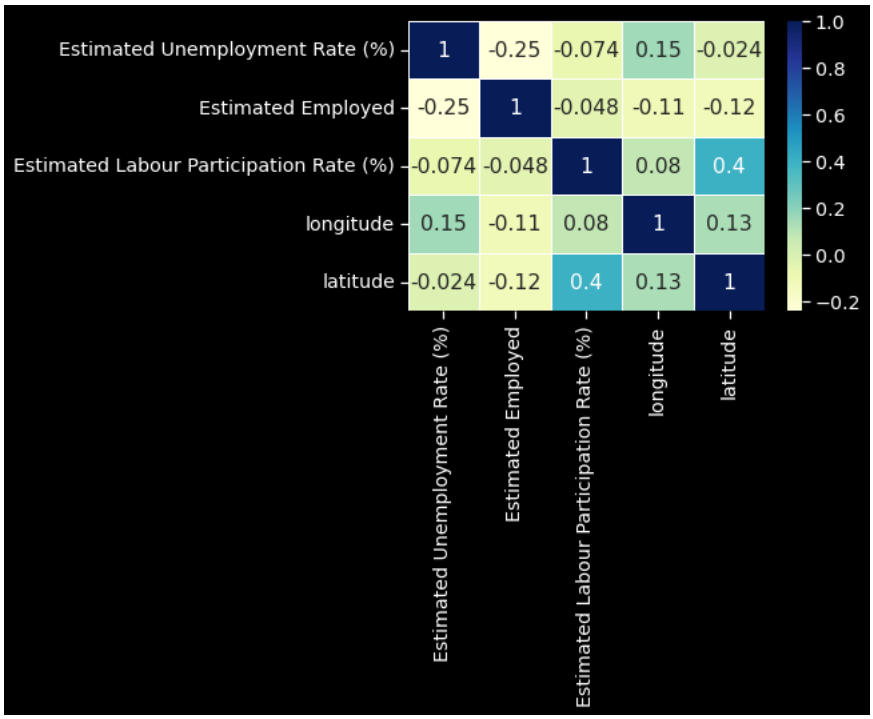
In [70]:
```
#Outlier detection
plt.figure(figsize=(12, 6))

# Subplot 1: Unemployment Rate
plt.subplot(131)
df.boxplot(column='Estimated Unemployment Rate (%)', patch_artist=True)
plt.title('Unemployment Rate')

# Subplot 2: Employed
plt.subplot(132)
df.boxplot(column='Estimated Employed', patch_artist=True)
plt.title('Employed')

# Subplot 3: Labor Participation Rate
plt.subplot(133)
df.boxplot(column='Estimated Labour Participation Rate (%)', patch_artist=True)
plt.title('Labor Participation Rate')

plt.tight_layout()
plt.show()
```



Presence of outliers are detected. Since we are analyzing the unemployment during Covid-19 the extreme values in the data are indicative of genuine structural changes or such events therefore we are not removing it.

In [71]:
```
#Correlation plot
sns.set_context('notebook', font_scale=1.3)
sns.heatmap(df.corr(), annot=True, cmap='YlGnBu', linewidths=0.5)
```

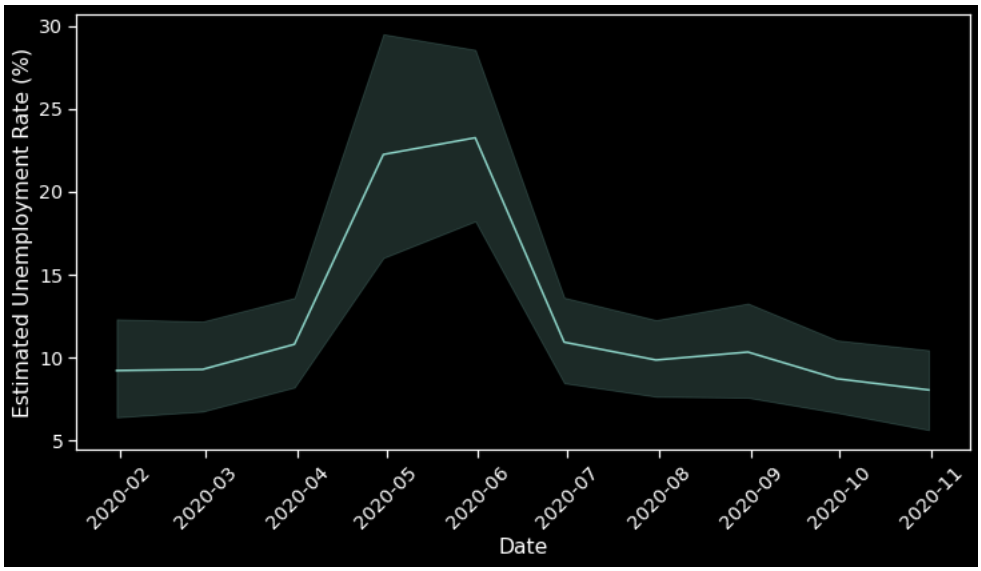
Out[71]:
```
<AxesSubplot:>
```



Strong positive and negetive correlation does not exist among the variables.

Unemployment rate in India during Covid-19

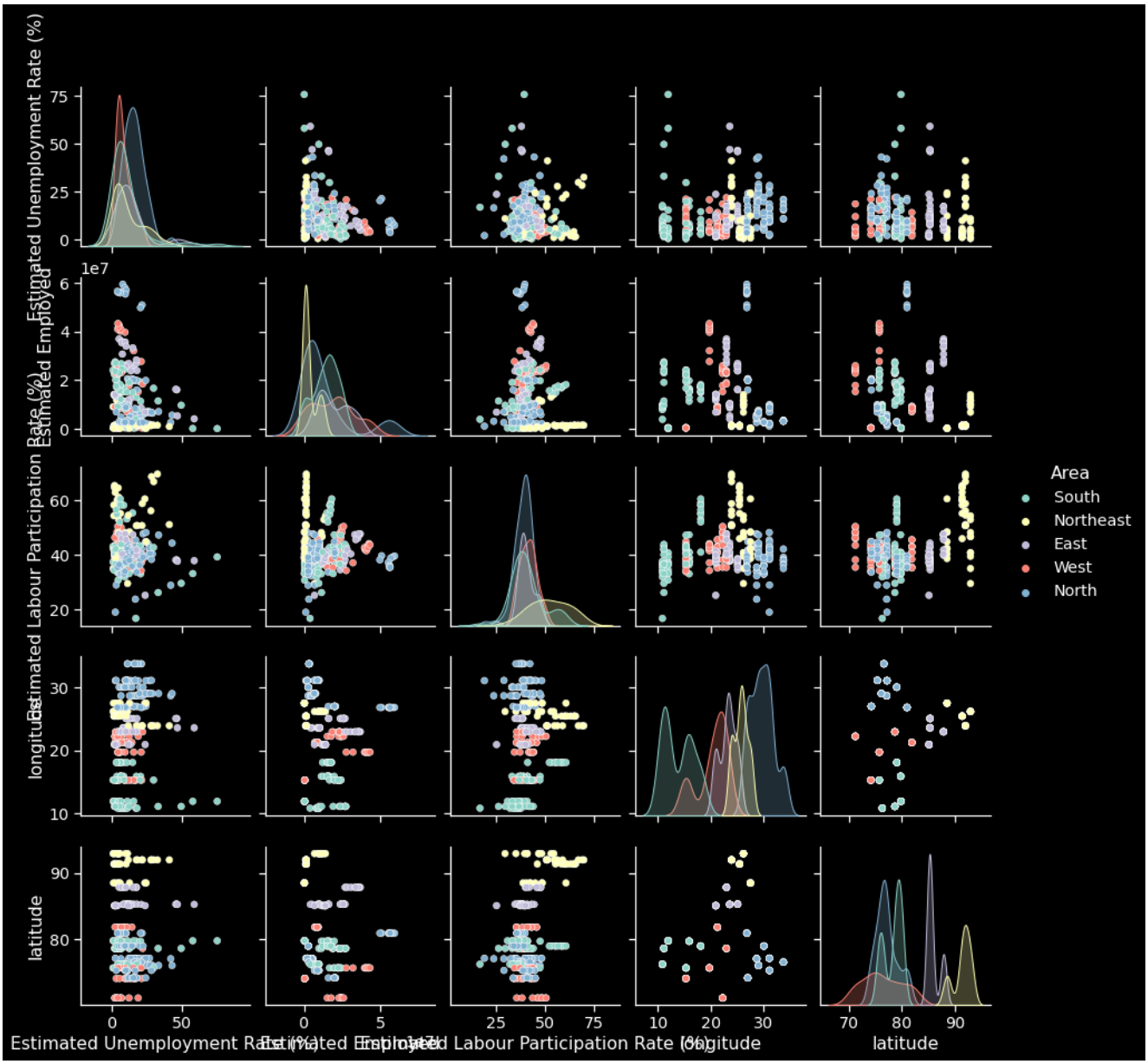
In [72]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='Date', y='Estimated Unemployment Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



The months of april, may, june witnessed high unemployment rate which can be associated with the lockdowns which lead to reduced economic activity and job losses in various sectors.

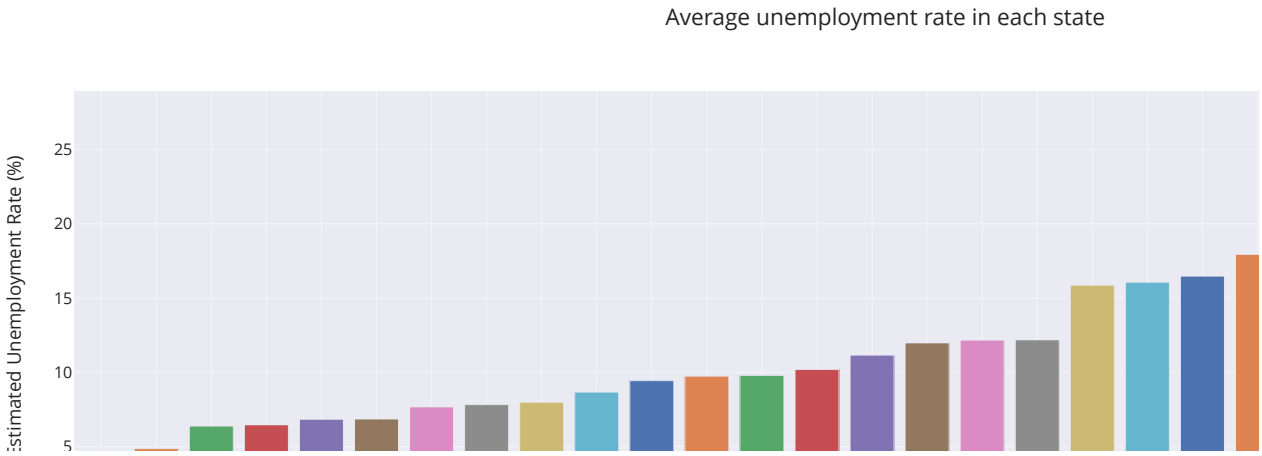
In [73]:
```
#Pair plot
sns.pairplot(df, hue='Area')
```

Out[73]:
```
<seaborn.axisgrid.PairGrid at 0x16468e90d60>
```



Unemployment rate in each state

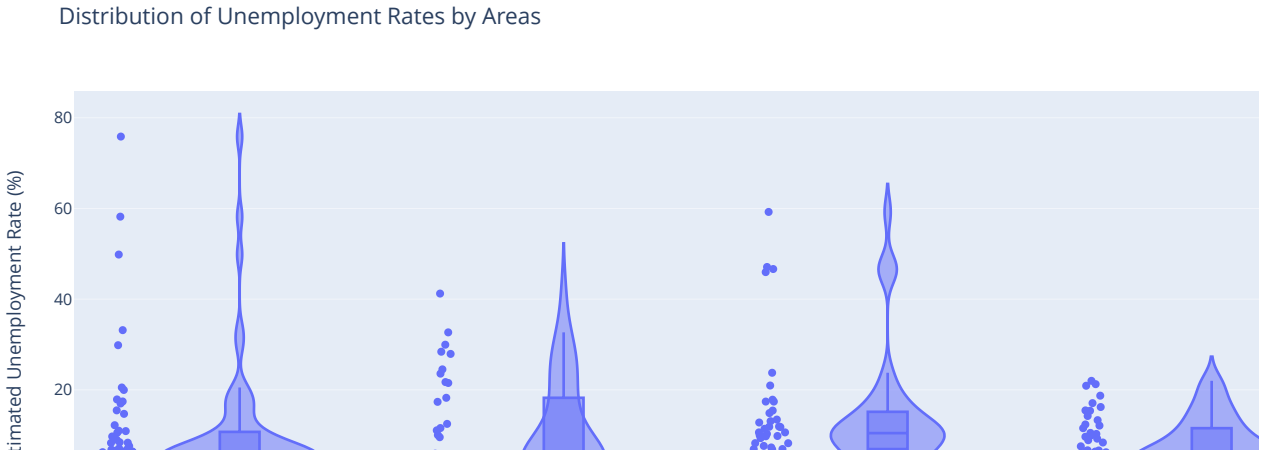
In [74]:
```
import plotly.express as px
plot_unemp = df[['Estimated Unemployment Rate (%)','Region']]
df_unemployed = plot_unemp.groupby('Region').mean().reset_index()

df_unemployed = df_unemployed.sort_values('Estimated Unemployment Rate (%)')

fig = px.bar(df_unemployed, x='Region',y='Estimated Unemployment Rate (%)',color = 'Region',title = 'Average unemployment rate in each sta
             template='seaborn')
fig.show()
```



The estimated unemployment rate is highest in Haryana.

Visualizes the distribution of unemployment rates within different areas

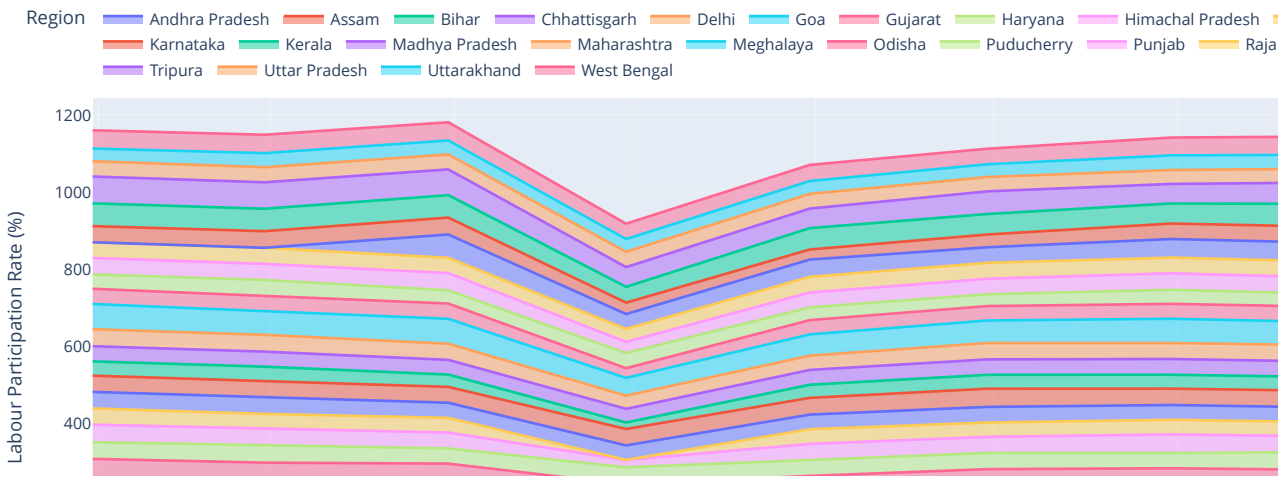
In [75]:
```
fig = px.violin(
    df,
    x='Area',
    y='Estimated Unemployment Rate (%)',
    title='Distribution of Unemployment Rates by Areas',
    box=True,  # Include box plot inside the violin
    points='all',  # Show individual data points
)

fig.show()
```



Since areas with wider or taller violins may have more variability in unemployment rates, south and east part of India experienced higher unemployment.

Composition of Labour Participation Rates by Region Over Time

In [76]:
```
fig = px.area(
    df,
    x='Date',
    y='Estimated Labour Participation Rate (%)',
    color='Region',
    labels={'Estimated Labour Participation Rate (%)': 'Labour Participation Rate (%)'},
    category_orders={'Region': df['Region'].unique()}  # Preserve the order of regions
)

fig.update_layout(
    xaxis_title='Date',
    yaxis_title='Labour Participation Rate (%)',
    legend_title='Region',
    legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1),
)

fig.show()
```



During the month of april labour participation declined all over India.

In [77]:
```
#Extracting month from date
df['Month'] = df['Date'].dt.month
df
```

Out[77]:

	Region	Date	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area	longitude	latitude	Month
0	Andhra Pradesh	2020-01-31	5.48	16635535	41.02	South	15.9129	79.740	1
1	Andhra Pradesh	2020-02-29	5.83	16545652	40.90	South	15.9129	79.740	2
2	Andhra Pradesh	2020-03-31	5.79	15881197	39.18	South	15.9129	79.740	3
3	Andhra Pradesh	2020-04-30	20.51	11336911	33.10	South	15.9129	79.740	4
4	Andhra Pradesh	2020-05-31	17.43	12988845	36.46	South	15.9129	79.740	5
...	...	...	...	...	...	...	...	...	...
262	West Bengal	2020-06-30	7.29	30726310	40.39	East	22.9868	87.855	6
263	West Bengal	2020-07-31	6.83	35372506	46.17	East	22.9868	87.855	7
264	West Bengal	2020-08-31	14.87	33298644	47.48	East	22.9868	87.855	8
265	West Bengal	2020-09-30	9.35	35707239	47.73	East	22.9868	87.855	9
266	West Bengal	2020-10-31	9.98	33962549	45.63	East	22.9868	87.855	10

267 rows × 9 columns

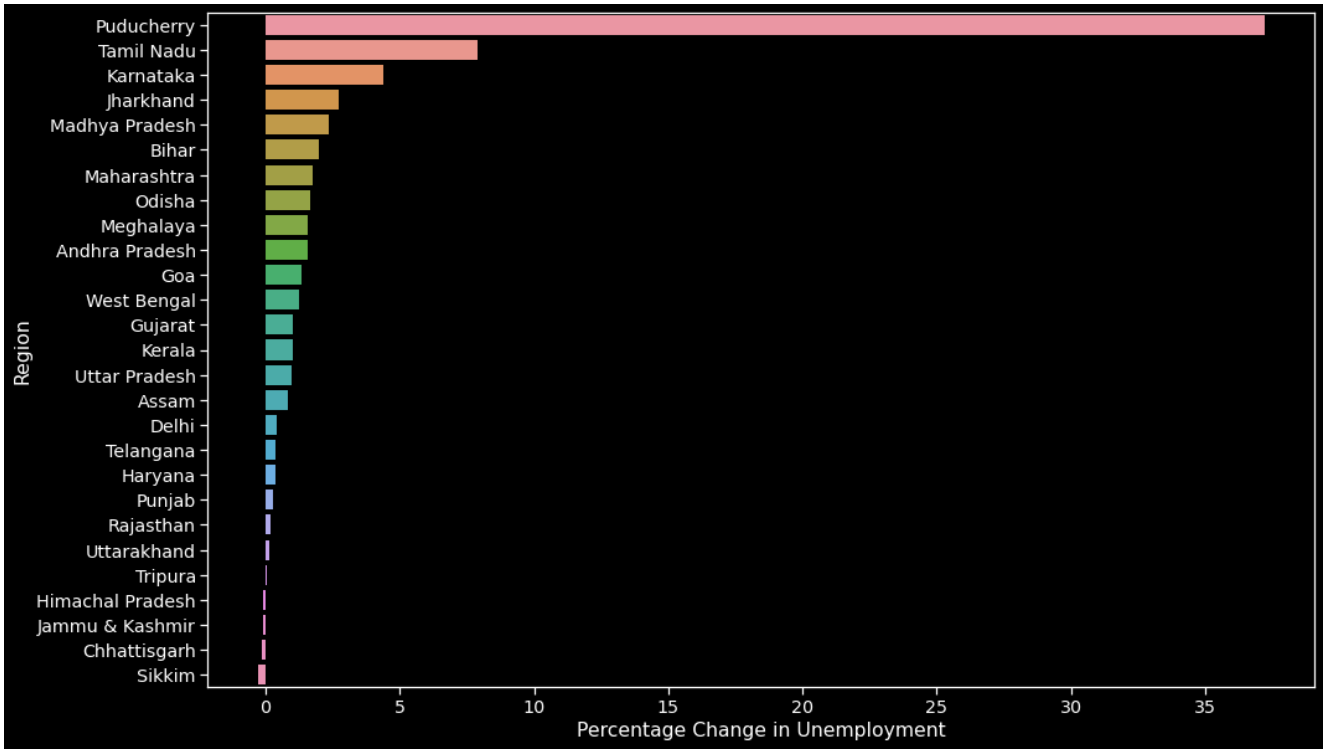
Percentage change in unemployment

In [78]:
```
# Filter data for months 1 to 3 (before lockdown)
before_lock = df[(df['Month'] >= 1) & (df['Month'] <= 3)][['Region', 'Estimated Unemployment Rate (%)']]
```

In [79]:
```
# Filter data for months 3 to 5 (after lockdown)
after_lock = df[(df['Month'] >= 3) & (df['Month'] < 6)][['Region', 'Estimated Unemployment Rate (%)']]
before_lock = before_lock.groupby('Region')['Estimated Unemployment Rate (%)'].mean().reset_index().rename(
    columns={'Estimated Unemployment Rate (%)':'Unemployment Rate before Lock-Down'})
after_lock = after_lock.groupby('Region')['Estimated Unemployment Rate (%)'].mean().reset_index().rename(
    columns={'Estimated Unemployment Rate (%)':'Unemployment Rate after Lock-Down'})
before_lock['Percentage Change in Unemployment'] = round((after_lock['Unemployment Rate after Lock-Down'] - before_lock['Unemployment Rate
```

In [80]:
```
plot_df = before_lock.sort_values('Percentage Change in Unemployment', ascending=False)
plt.figure(figsize=(16, 10))
sns.barplot(data=plot_df, y='Region', x='Percentage Change in Unemployment')
```

Out[80]:
```
<AxesSubplot:xlabel='Percentage Change in Unemployment', ylabel='Region'>
```



If the percentage change is positive (+X%), it means that unemployment has increased by X% compared to the previous period. In other words, more people are unemployed.

If the percentage change is negative (-X%), it means that unemployment has decreased by X% compared to the previous period. Fewer people are unemployed.

The magnitude of the percentage change indicates how significant the change is. A larger percentage change suggests a more substantial shift in unemployment rates compared to a smaller percentage change.

Puducherry's unemployment rate had been seriously impacted by the lock-down.

Sikkim, Chattisgarh, Jammu & Kashmir and Himachal Pradesh have negetive percentage change.That means these states are not highly impacted by the lock down.