Task-2:

Unemployment Analysis With Python

Unemployment analysis in Python involves collecting and processing labor market data, such as employment rates, job vacancies, and demographic information. This data can be visualized using libraries like Matplotlib or Seaborn to identify trends and patterns. Statistical methods and machine learning models can also be applied to predict future unemployment rates or analyze the impact of various economic factors on joblessness. Ultimately, Python facilitates a comprehensive understanding of unemployment dynamics through data exploration and modeling.

by Mouli Nahal

In [18]:
```
#Importing required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
## Supress warnings
import warnings
warnings.filterwarnings("ignore")
```

In [19]:
```
data = pd.read_csv("unemployment.csv")
print("data has been successfully loaded")
```

```
data has been successfully loaded
```

In [20]:
```
# Checking and cleaning the dataset:
```

In [21]:
```
data
```

Out[21]:

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

267 rows × 9 columns

In [22]:
```
data.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   Region.1                                 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
```

In [23]:
```
data.tail()
```

Out[23]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude
262	West Bengal	30-06-2020	M	7.29	30726310	40.39	East	22.9868
263	West Bengal	31-07-2020	M	6.83	35372506	46.17	East	22.9868
264	West Bengal	31-08-2020	M	14.87	33298644	47.48	East	22.9868
265	West Bengal	30-09-2020	M	9.35	35707239	47.73	East	22.9868
266	West Bengal	31-10-2020	M	9.98	33962549	45.63	East	22.9868

In [24]:
```
data.shape
```

Out[24]: (267, 9)

In [25]:
```
data.sample()
```

Out[25]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Region.1	longitude	l
1	Andhra Pradesh	29-02-2020	M	5.83	16545652	40.9	South	15.9129	

In [26]:
```
data.describe()
```

Out[26]:

	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

Let’s see if this dataset contains missing values or not:

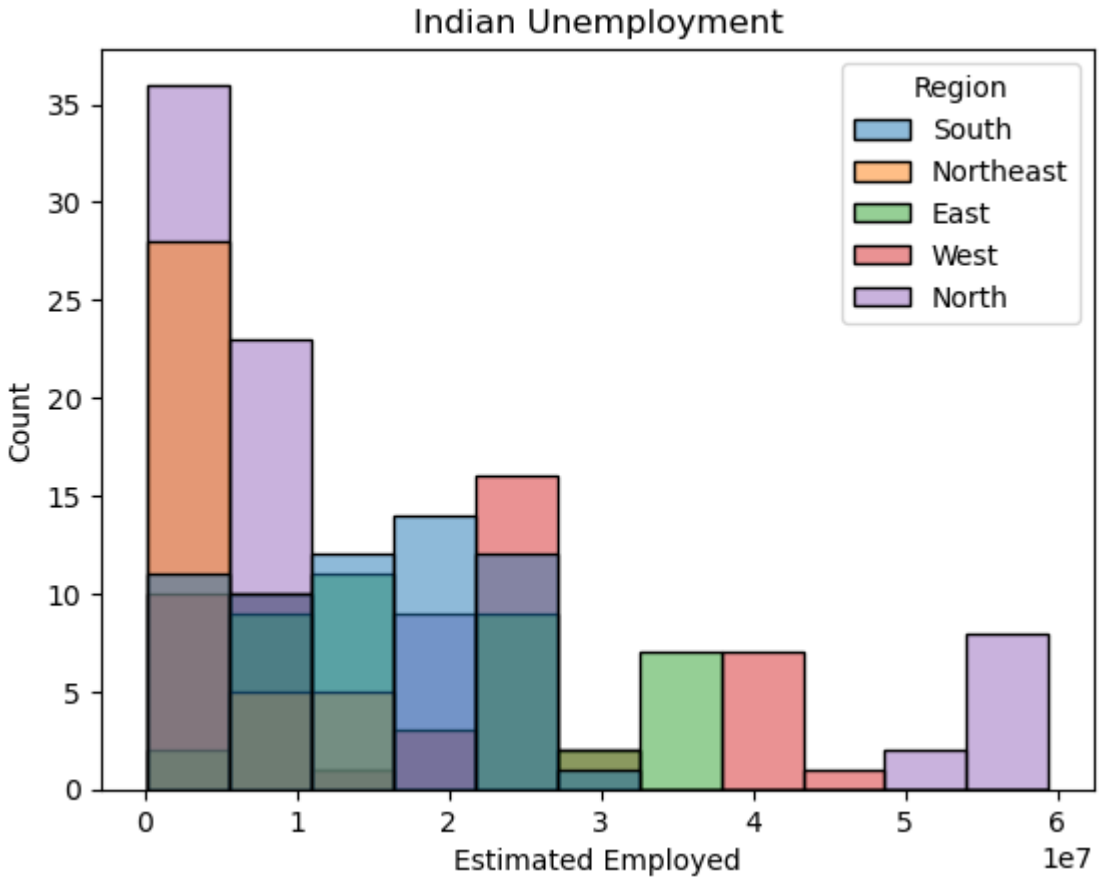
In [27]:
```
print(data.isnull().sum())
```

```
Region                                   0
 Date                                    0
 Frequency                               0
 Estimated Unemployment Rate (%)         0
 Estimated Employed                      0
 Estimated Labour Participation Rate (%) 0
Region.1                                 0
longitude                                0
latitude                                 0
dtype: int64
```

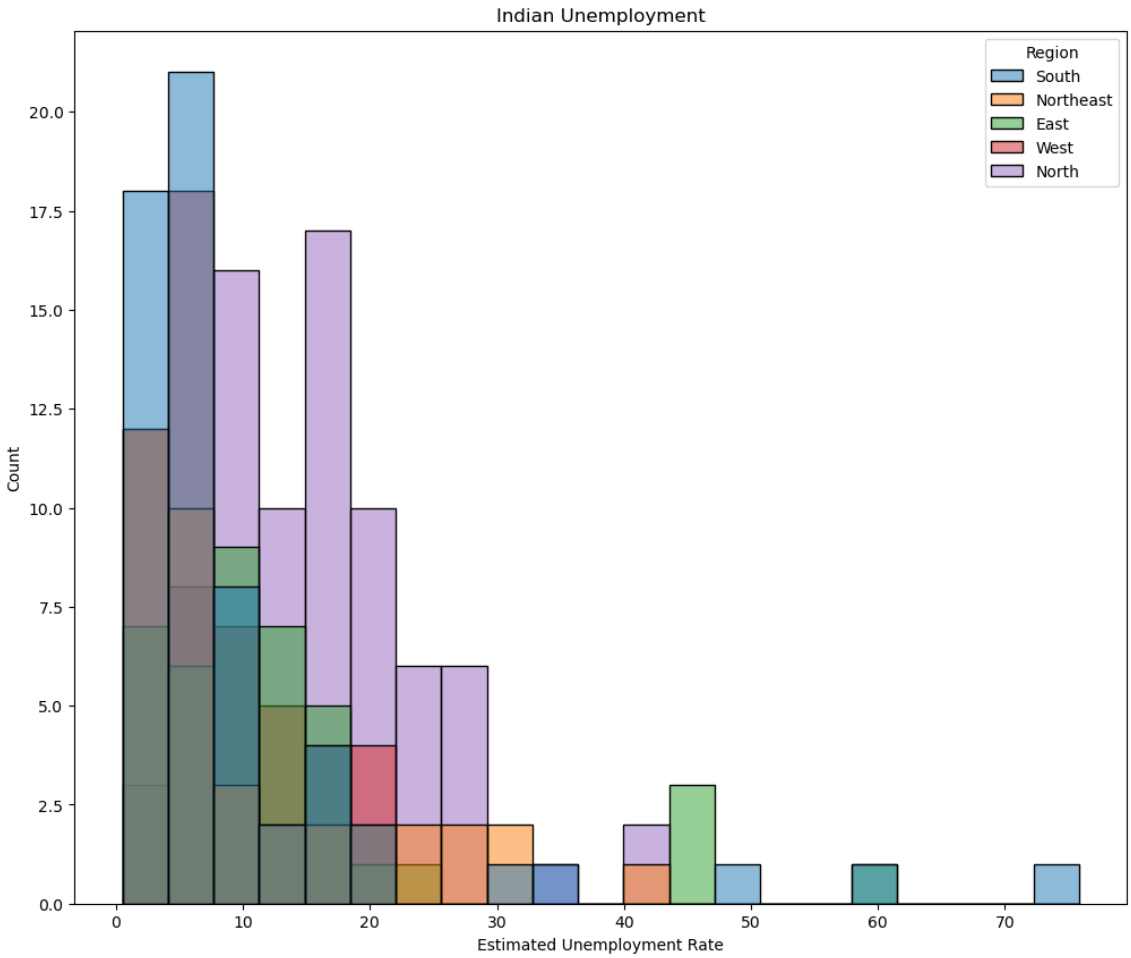
In [28]:
```
data.columns= ["States","Date","Frequency",
 "Estimated Unemployment Rate",
 "Estimated Employed",
 "Estimated Labour Participation Rate",
 "Region","longitude","latitude"]
```

Unemployment Rate Analysis: Data Visualization

In [29]:
```
data.columns= ["States","Date","Frequency",
 "Estimated Unemployment Rate","Estimated Employed",
 "Estimated Labour Participation Rate","Region",
 "longitude","latitude"]
plt.title("Indian Unemployment")
sns.histplot(x="Estimated Employed", hue="Region", data=data)
plt.show()
```



In [30]:
```
plt.figure(figsize=(12, 10))
plt.title("Indian Unemployment")
sns.histplot(x="Estimated Unemployment Rate", hue="Region", data=data)
plt.show()
```



In [31]:
```
unemploment = data[["States", "Region", "Estimated Unemployment Rate"]]
figure = px.sunburst(unemploment, path=["Region", "States"],
 values="Estimated Unemployment Rate",
 width=700, height=700, color_continuous_scale="RdY1Gn",
 title="Unemployment Rate in India")
figure.show()
```

see the number of unique region:

In [32]:
```
data.Region.nunique()
```

Out[32]: 5

See exact numbers:

In [33]:
```
make_total = data.pivot_table("Estimated Unemployment Rate",index=['Region'
topstate=make_total.sort_values(by='Estimated Unemployment Rate',ascending=
print(topstate)
```

```
           Estimated Unemployment Rate
Region
North                        15.889620
East                         13.916000
Northeast                    10.950263
South                        10.454667
West                          8.239000
```

Calculate which models has highest yearly fluncations:

In [34]:
```
maketotal_1 = data.pivot_table(values='Estimated Unemployment Rate',index=[
df1 = maketotal_1.reset_index().dropna(subset=['Estimated Unemployment Rate
df2 = df1.loc[df1.groupby('Region')['Estimated Unemployment Rate'].idxmax()
for index,row in df2.iterrows():
 print(row['Region'],"Region which",row['Region'],"has the highest yearly f
```

```
East Region which East has the highest yearly fluncation.
North Region which North has the highest yearly fluncation.
Northeast Region which Northeast has the highest yearly fluncation.
South Region which South has the highest yearly fluncation.
West Region which West has the highest yearly fluncation.
```

In [ ]:

In [ ]: