45cognorise1

September 13, 2024

#CognoRise Infotech **Data Analysis Internship Task 1**

Unemployment In India This notebook presents an analysis of unemployment data in India, covering data preprocessing, exploratory data analysis (EDA), hypothesis testing, and model building. The goal is to uncover trends, patterns, and potential predictive insights from the dataset.

Understanding the Problem and Data In this project, we're analyzing unemployment data in India. The dataset contains information such as:

Region: Different regions in India. Date: The time when the unemployment data was recorded. Frequency: The frequency of data recording (e.g., monthly). Estimated Unemployment Rate (%): The percentage of the labor force that is unemployed. Estimated Employed: The estimated number of employed individuals. Estimated Labour Participation Rate (%): The percentage of the population that is either employed or actively looking for work. Area: Whether the area is urban or rural. Goal: Explore this data to identify trends and patterns, and possibly build models to predict future unemployment rates.

[115]:
```
#import all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[116]:
```
# import dataset
df = pd.read_csv("Unemployment in India.csv")
df.head()
```

[116]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	Andhra Pradesh	31-05-2019	Monthly	3.65	11999139.0	43.24	Rural
1	Andhra Pradesh	30-06-2019	Monthly	3.05			
2	Andhra Pradesh	31-07-2019	Monthly	3.75			
3	Andhra Pradesh	31-08-2019	Monthly	3.32			
4	Andhra Pradesh	30-09-2019	Monthly	5.17			

```
1           11755881.0                                    42.05  Rural
2           12086707.0                                    43.50  Rural
3           12285693.0                                    43.97  Rural
4           12256762.0                                    44.68  Rural
```

[117]:
```
# Information about data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 7 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Region                                    740 non-null    object
 1    Date                                     740 non-null    object
 2    Frequency                                740 non-null    object
 3    Estimated Unemployment Rate (%)          740 non-null    float64
 4    Estimated Employed                       740 non-null    float64
 5    Estimated Labour Participation Rate (%)  740 non-null    float64
 6   Area                                      740 non-null    object
dtypes: float64(3), object(4)
memory usage: 42.1+ KB
```

[118]:
```
# Statistical Description of data
df.describe()
```

[118]:
```
        Estimated Unemployment Rate (%)   Estimated Employed  \
count                        740.000000         7.400000e+02
mean                          11.787946         7.204460e+06
std                           10.721298         8.087988e+06
min                            0.000000         4.942000e+04
25%                            4.657500         1.190404e+06
50%                            8.350000         4.744178e+06
75%                           15.887500         1.127549e+07
max                           76.740000         4.577751e+07

        Estimated Labour Participation Rate (%)
count                                740.000000
mean                                  42.630122
std                                    8.111094
min                                   13.330000
25%                                   38.062500
50%                                   41.160000
75%                                   45.505000
max                                   72.570000
```

[119]:
```
# Shape of data
df.shape
```

[119]:
```
(768, 7)
```

[119]:
```

```

[120]:
```
# Print column names to identify any issues
print("Original Column Names:")
print(df.columns)

# Remove any leading or trailing spaces from column names
df.columns = df.columns.str.strip()

# Print column names to confirm the changes
print("\nCleaned Column Names:")
print(df.columns)
```

```
Original Column Names:
Index(['Region', ' Date', ' Frequency', ' Estimated Unemployment Rate (%)',
       ' Estimated Employed', ' Estimated Labour Participation Rate (%)',
       'Area'],
      dtype='object')

Cleaned Column Names:
Index(['Region', 'Date', 'Frequency', 'Estimated Unemployment Rate (%)',
       'Estimated Employed', 'Estimated Labour Participation Rate (%)',
       'Area'],
      dtype='object')
```

[121]:
```
# Verify the presence of the 'Date' column
if 'Date' in df.columns:
    # Remove any leading or trailing spaces from the 'Date' column values
    df['Date'] = df['Date'].str.strip()

    # Convert 'Date' to datetime format
    df['Date'] = pd.to_datetime(df['Date'], format='%d-%m-%Y')


    # Display the data types to confirm the changes
    print("\nData types after conversion:")
    print(df.dtypes)

    # Display the first few rows of the dataframe
    df.head()
else:
```

```
    print("The 'Date' column was not found. Please check the dataset for any␣
 ↪discrepancies.")
```

```
Data types after conversion:
Region                                          object
Date                                    datetime64[ns]
Frequency                                       object
Estimated Unemployment Rate (%)                float64
Estimated Employed                             float64
Estimated Labour Participation Rate (%)        float64
Area                                            object
dtype: object
```

[122]:
```
# Create a new column "Year" extract from Date Column

    # data['year'], data['month'] = data[' Date'].dt.year, df['Date'].dt.month

df['Year'] = data['Date'].dt.year
df['Month'] = data['Date'].dt.month
```

[122]:

[123]:
```
# Rename columns for easier reference
df.rename(columns={
    'Estimated Unemployment Rate (%)': 'Unemploy_Rate',
    'Estimated Employed': 'Employed',
    'Estimated Labour Participation Rate (%)': 'Labr_Patcptn_Rate',
    'Region.1': 'Reg_zone'
}, inplace=True)

# Display the first few rows after preprocessing
df.head()
```

[123]:

	Region	Date	Frequency	Unemploy_Rate	Employed	Labr_Patcptn_Rate	Area	Year	Month
0	Andhra Pradesh	2019-05-31	Monthly	3.65	11999139.0	43.24	Rural	2020.0	1.0
1	Andhra Pradesh	2019-06-30	Monthly	3.05	11755881.0	42.05	Rural	2020.0	2.0
2	Andhra Pradesh	2019-07-31	Monthly	3.75	12086707.0	43.50	Rural	2020.0	3.0
3	Andhra Pradesh	2019-08-31	Monthly	3.32	12285693.0	43.97	Rural	2020.0	4.0
4	Andhra Pradesh	2019-09-30	Monthly	5.17	12256762.0	44.68	Rural	2020.0	5.0

[124]:
```
# Check for missing values in the dataset
missing_values = df.isnull().sum()
print("Missing values in each column:")
print(missing_values)
# Remove null values
data = df.dropna()
```

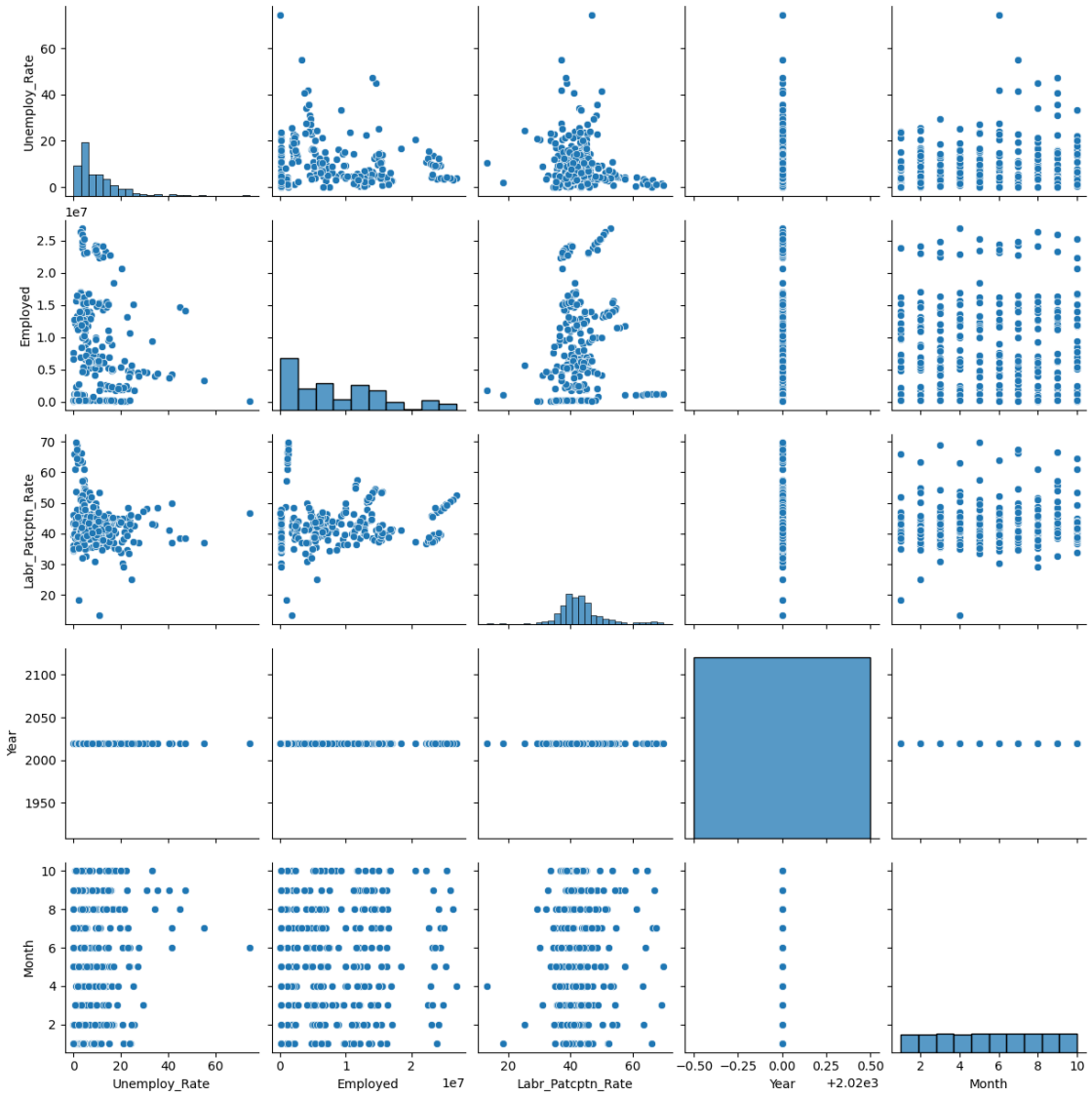
```
Missing values in each column:
Region                28
Date                  28
Frequency             28
Unemploy_Rate         28
Employed              28
Labr_Patcptn_Rate     28
Area                  28
Year                 501
Month                501
dtype: int64
```

[125]:
```
data.isnull().sum()
```

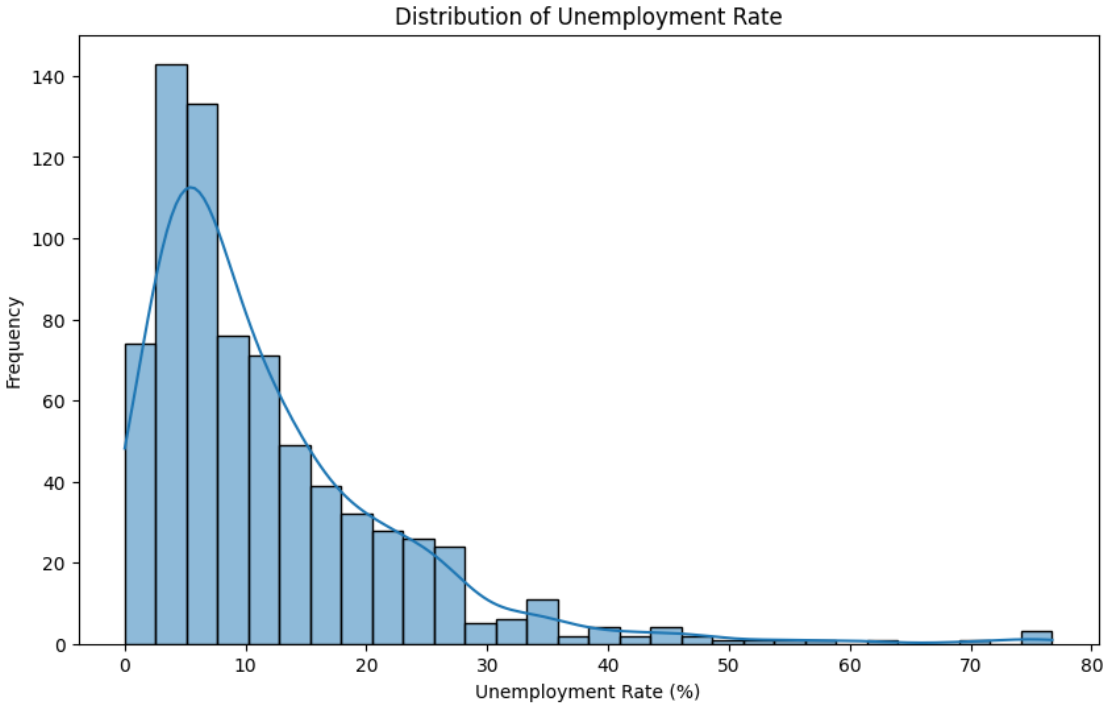
[125]:
```
Region               0
Date                 0
Frequency            0
Unemploy_Rate        0
Employed             0
Labr_Patcptn_Rate    0
Area                 0
Year                 0
Month                0
dtype: int64
```

[126]:
```
sns.pairplot(data)
```

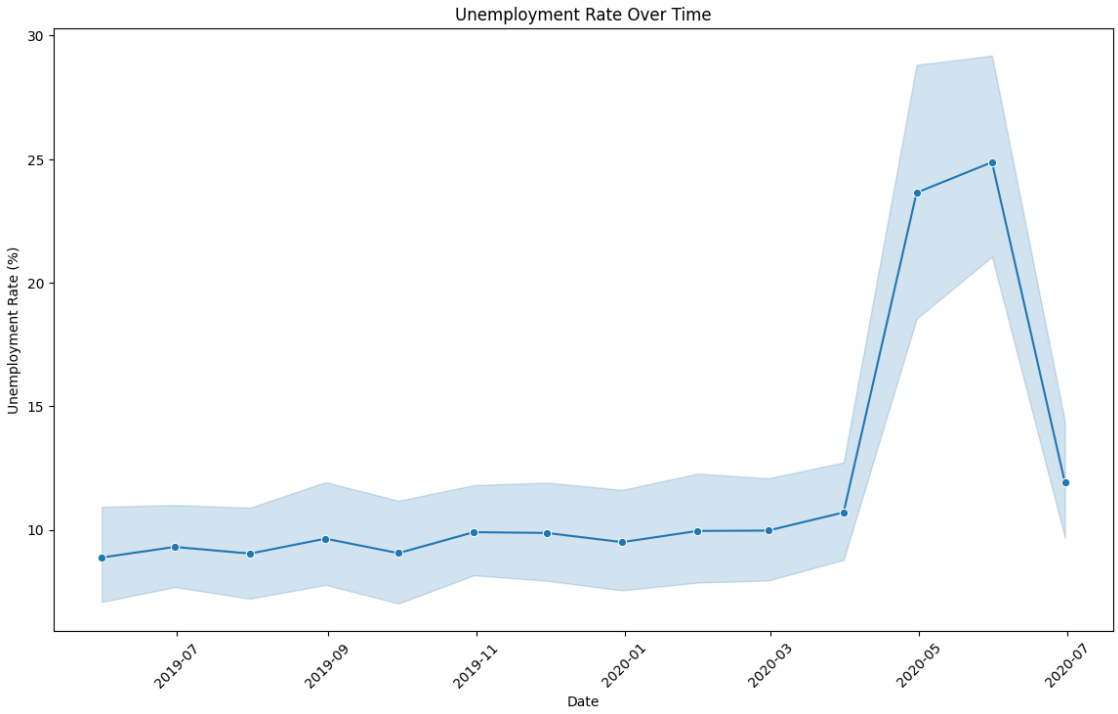
[126]:
```
<seaborn.axisgrid.PairGrid at 0x7a7680b50220>
```



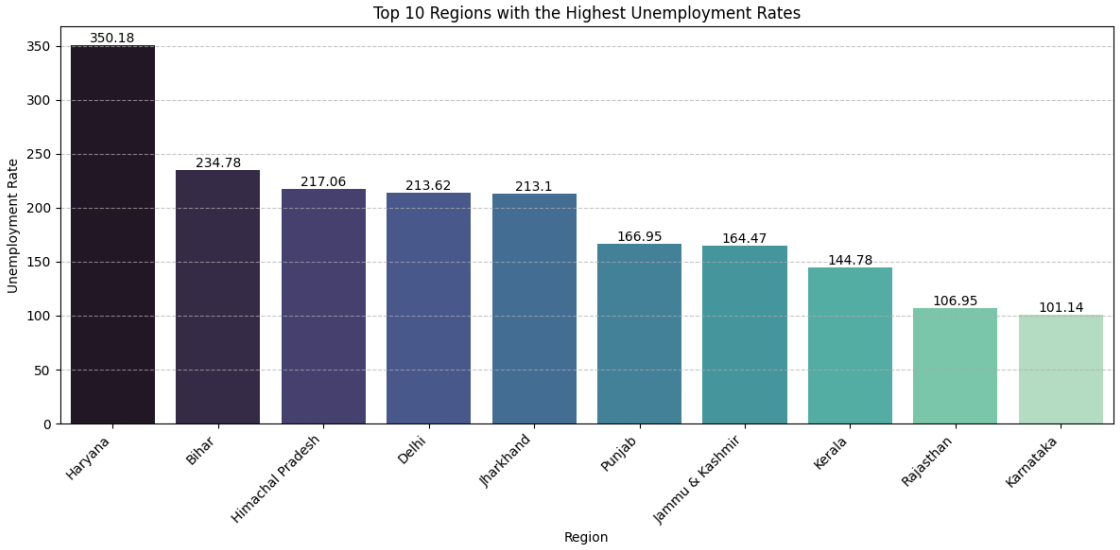
[127]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of Unemployment Rate
plt.figure(figsize=(10, 6))
sns.histplot(df['Unemploy_Rate'], bins=30, kde=True)
plt.title('Distribution of Unemployment Rate')
plt.xlabel('Unemployment Rate (%)')
plt.ylabel('Frequency')
plt.show()
```



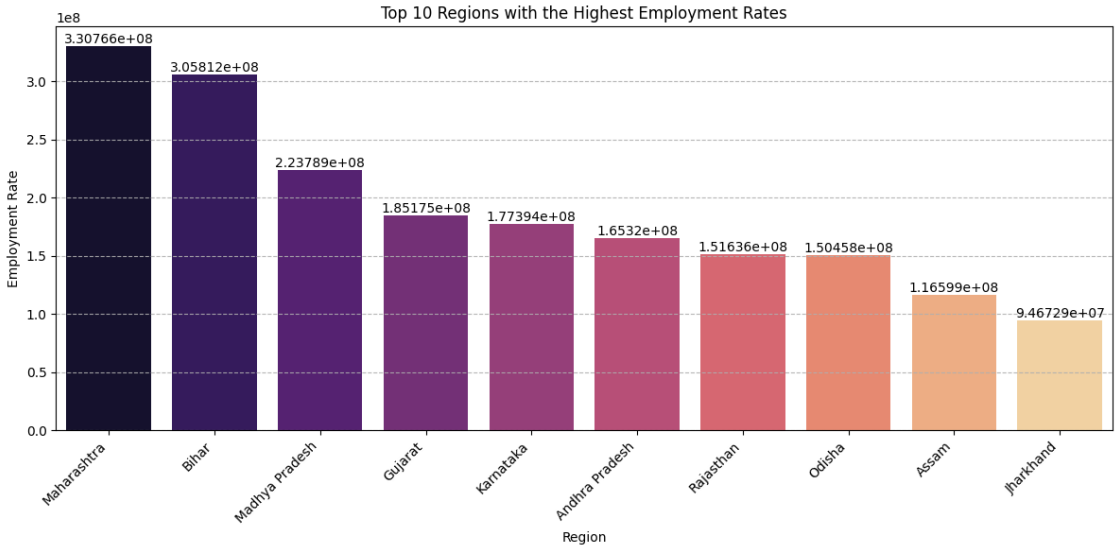
[128]:
```
# Trend of Unemployment Rate Over Time
plt.figure(figsize=(14, 8))
sns.lineplot(x='Date', y='Unemploy_Rate', data=df, marker='o')
plt.title('Unemployment Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate (%)')
plt.xticks(rotation=45)
plt.show()
```



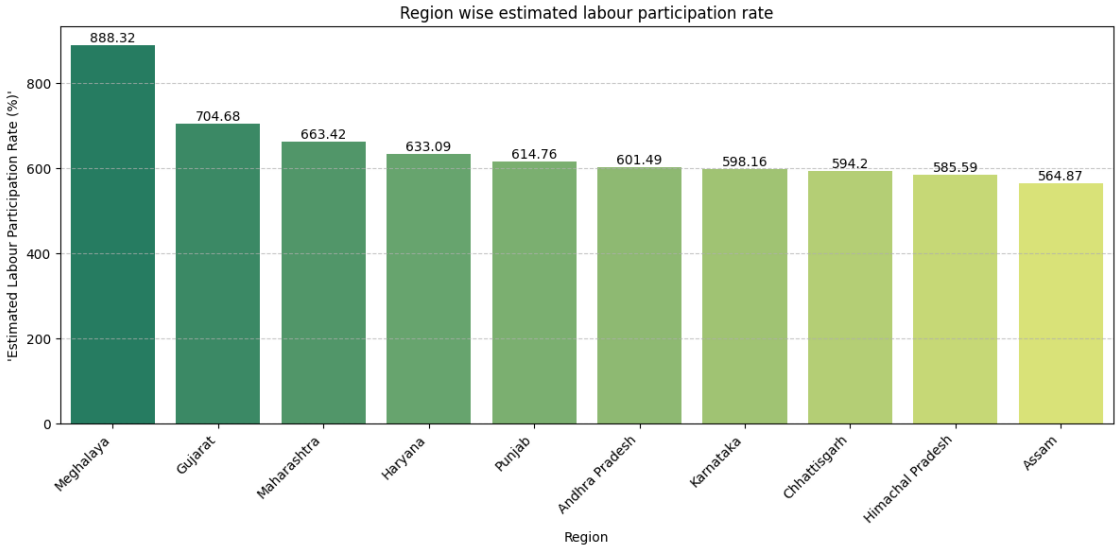
[129]:
```
# Top 10 Regions with the highest unemployment rate

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
top10RegionUnemploymentData = data.groupby('Region')["Unemploy_Rate"].sum().
 ↪nlargest(10).sort_values(ascending=False)
top10RegionUnemployment = sns.barplot(x=top10RegionUnemploymentData.index,␣
 ↪y=top10RegionUnemploymentData.values, palette="mako")

for bars in top10RegionUnemployment.containers:
    top10RegionUnemployment.bar_label(bars)

plt.title('Top 10 Regions with the Highest Unemployment Rates')
plt.xlabel("Region")
plt.ylabel("Unemployment Rate")
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



[130]:
```
# Top 10 Regions with the highest employment rate

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
top10RegionemploymentData = data.groupby('Region')["Employed"].sum().
 ↪nlargest(10).sort_values(ascending=False)
top10Regionemployment = sns.barplot(x=top10RegionemploymentData.index,␣
 ↪y=top10RegionemploymentData.values, palette="magma")

for bars in top10Regionemployment.containers:
    top10Regionemployment.bar_label(bars)

plt.title('Top 10 Regions with the Highest Employment Rates')
plt.xlabel("Region")
plt.ylabel("Employment Rate")
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.9)
plt.tight_layout()
plt.show()
```



[131]:
```
# Region wise estimated labour participation rate

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12, 6))
top10EstimatedLabourParticipatonRate = data.
 ↪groupby('Region')['Labr_Patcptn_Rate'].sum().nlargest(10).
 ↪sort_values(ascending=False)
top10EstimatedLabourParticipatonRate = sns.
 ↪barplot(x=top10EstimatedLabourParticipatonRate.index,␣
 ↪y=top10EstimatedLabourParticipatonRate.values, palette="summer")

for bars in top10EstimatedLabourParticipatonRate.containers:
    top10EstimatedLabourParticipatonRate.bar_label(bars)

plt.title('Region wise estimated labour participation rate')
plt.xlabel("Region")
plt.ylabel("'Estimated Labour Participation Rate (%)'")
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



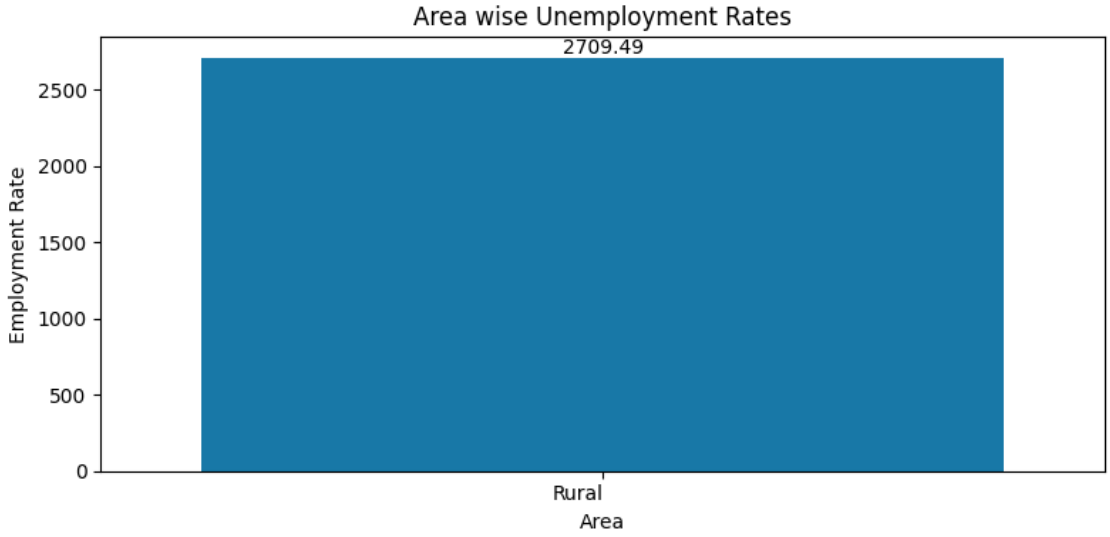
[132]:
```
data.groupby('Region')['Labr_Patcptn_Rate'].sum().sort_values(ascending=False)
```

[132]:
```
Region
Meghalaya           888.32
Gujarat             704.68
Maharashtra         663.42
Haryana             633.09
Punjab              614.76
Andhra Pradesh      601.49
Karnataka           598.16
Chhattisgarh        594.20
Himachal Pradesh    585.59
Assam               564.87
Jharkhand           564.61
Madhya Pradesh      557.41
Odisha              551.38
Bihar               537.84
Delhi               517.91
Puducherry          510.26
Kerala              496.82
Goa                 466.24
Jammu & Kashmir     441.51
Rajasthan           408.55
Name: Labr_Patcptn_Rate, dtype: float64
```

[133]:
```
data.groupby('Area')['Unemploy_Rate'].sum()
```

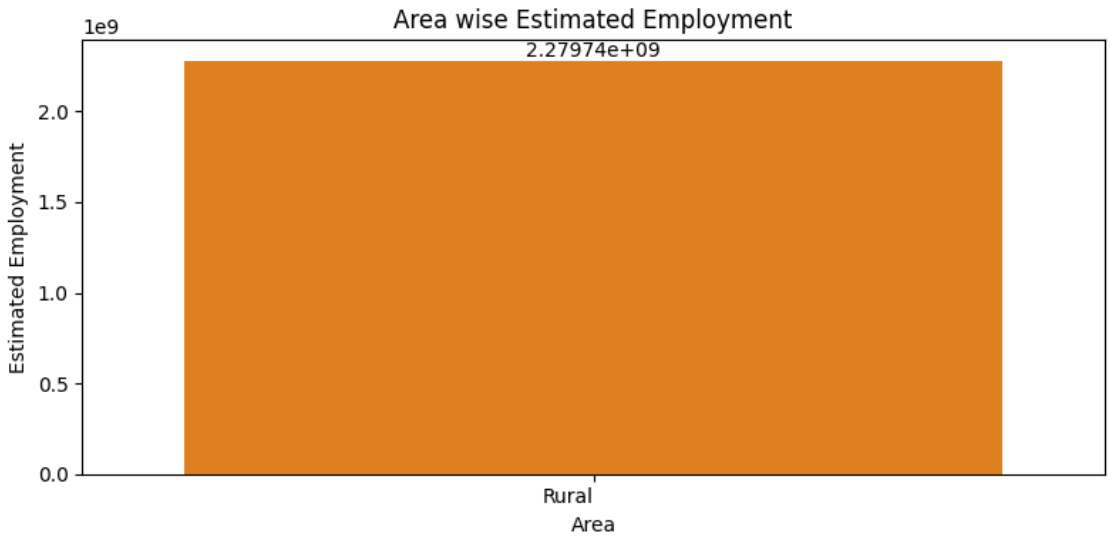
[133]:
```
Area
Rural    2709.49
Name: Unemploy_Rate, dtype: float64
```

[134]:
```
plt.figure(figsize=(8,4))
AreaWiseUnemploymentRate = data.groupby('Area')["Unemploy_Rate"].sum().
 ↪nlargest(10).sort_values(ascending=False)
AreaWiseUnemploymentRate  = sns.barplot(x=AreaWiseUnemploymentRate .index,␣
 ↪y=AreaWiseUnemploymentRate .values, palette="winter")

for bars in AreaWiseUnemploymentRate .containers:
    AreaWiseUnemploymentRate .bar_label(bars)

plt.title('Area wise Unemployment Rates')
plt.xlabel("Area")
plt.ylabel("Employment Rate")
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()
```

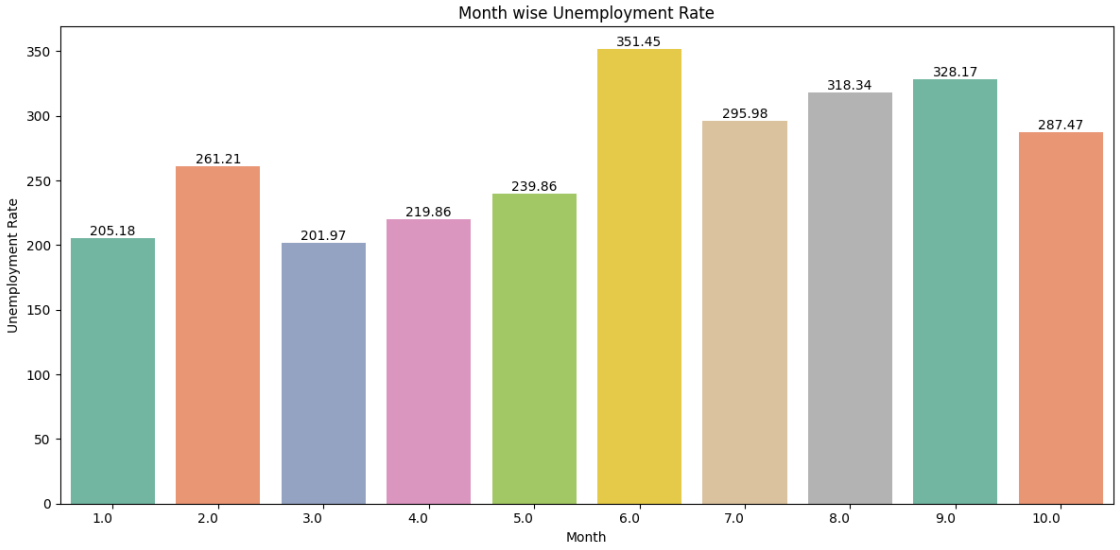


[135]:
```
plt.figure(figsize=(8,4))
AreaWiseEstimatedEmployment = data.groupby('Area')["Employed"].sum().
 ↪nlargest(10).sort_values(ascending=False)
AreaWiseEstimatedEmployment  = sns.barplot(x=AreaWiseEstimatedEmployment .
 ↪index, y=AreaWiseEstimatedEmployment.values, palette="autumn")

for bars in AreaWiseEstimatedEmployment.containers:
    AreaWiseEstimatedEmployment.bar_label(bars)
```

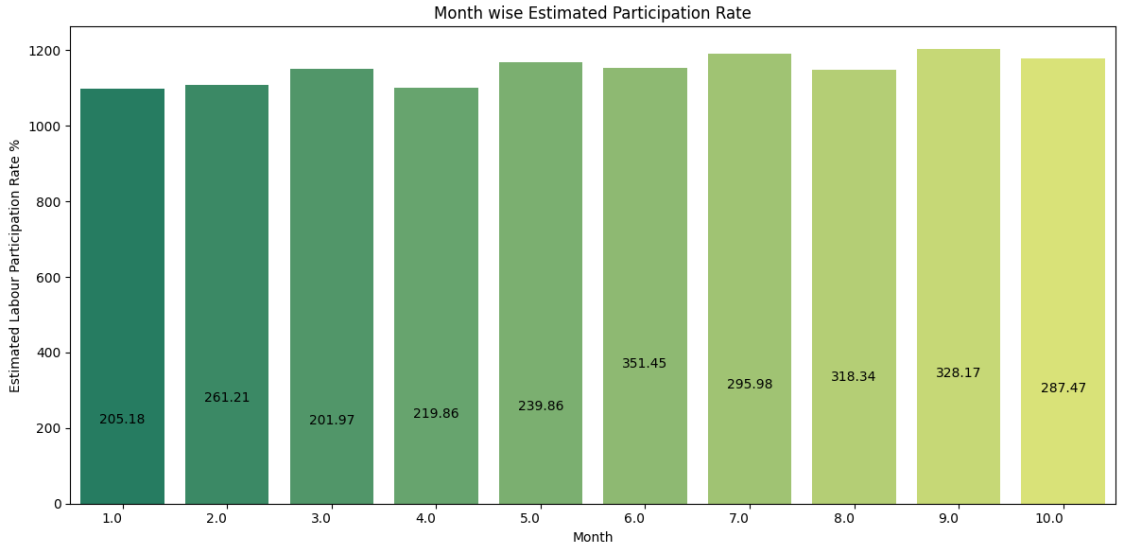
```
plt.title('Area wise Estimated Employment')
plt.xlabel("Area")
plt.ylabel("Estimated Employment")
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()
```



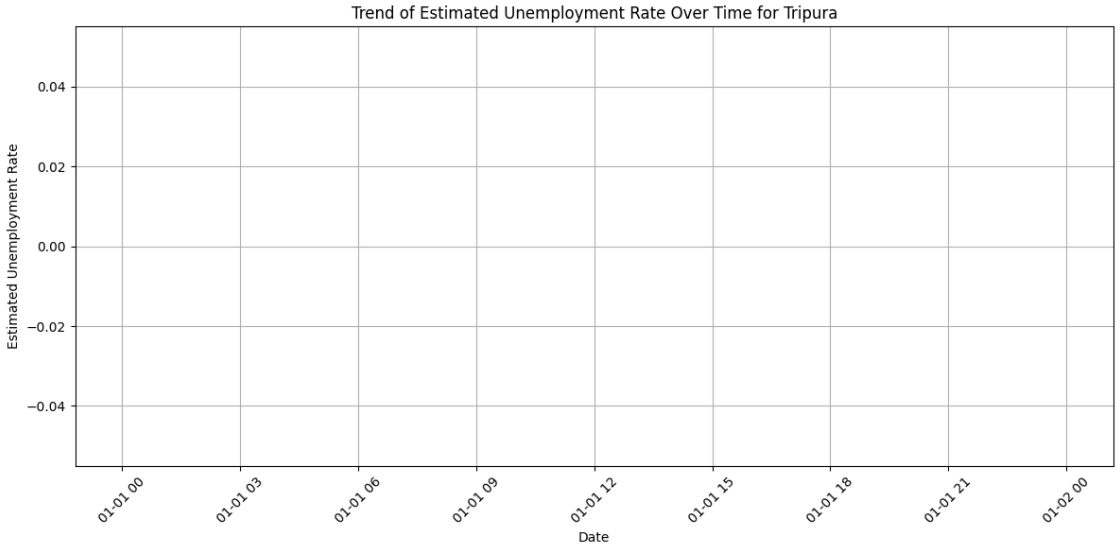
[136]:
```
plt.figure(figsize=(12,6))
MonthwiseUnemploymentRate = data.groupby('Month')["Unemploy_Rate"].sum().
 ↪nlargest(12).sort_values(ascending=False)
MonthwiseUnemploymentRate  = sns.barplot(x=MonthwiseUnemploymentRate.index,␣
 ↪y=MonthwiseUnemploymentRate.values, palette="Set2")
for bars in MonthwiseUnemploymentRate.containers:
    MonthwiseUnemploymentRate.bar_label(bars)
plt.title('Month wise Unemployment Rate')
plt.xlabel("Month")
plt.ylabel("Unemployment Rate")
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()
```

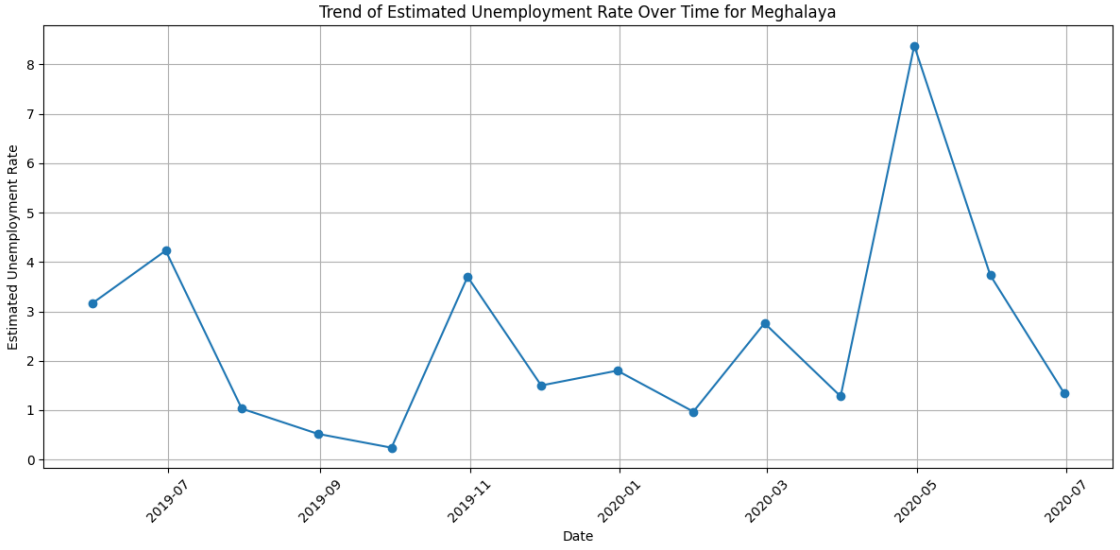


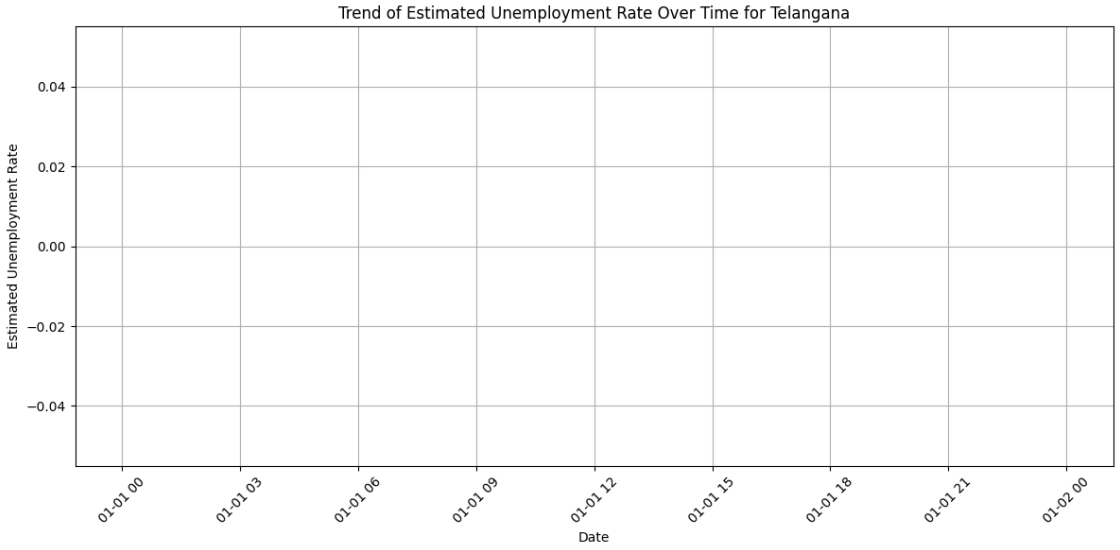
[137]:
```
# Month wise Estimated Labour Participation Rate

import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(12,6))
MonthwiseEstimatedLabourParticipationRate = data.
 ↪groupby('Month')["Labr_Patcptn_Rate"].sum().nlargest(12).
 ↪sort_values(ascending=False)
MonthwiseEstimatedLabourParticipationRate  = sns.
 ↪barplot(x=MonthwiseEstimatedLabourParticipationRate.index,␣
 ↪y=MonthwiseEstimatedLabourParticipationRate.values, palette="summer")
for bars in MonthwiseUnemploymentRate.containers:
   MonthwiseEstimatedLabourParticipationRate.bar_label(bars)
plt.title('Month wise Estimated Participation Rate')
plt.xlabel("Month")
plt.ylabel("Estimated Labour Participation Rate %")
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()
```

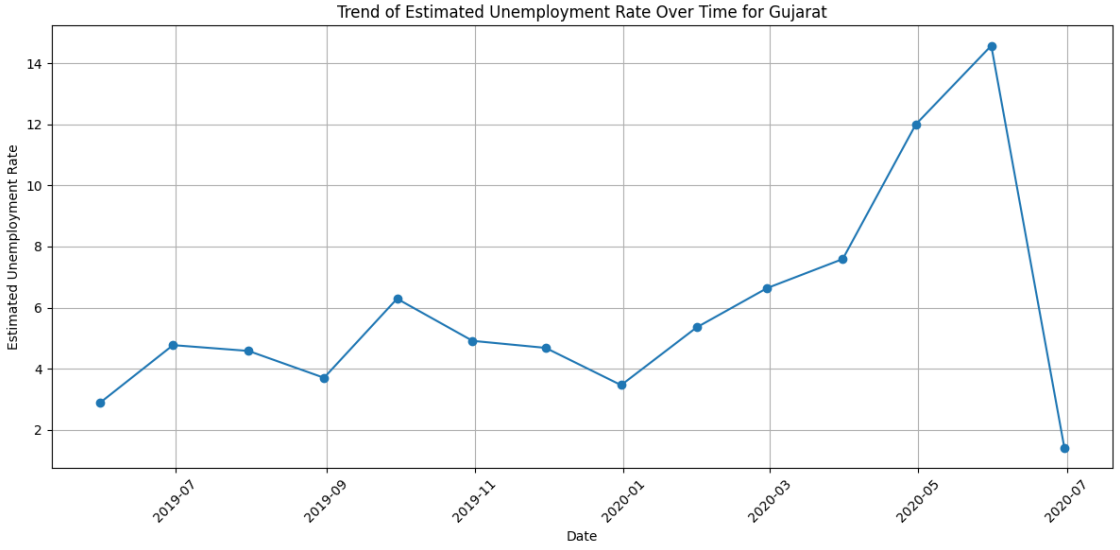


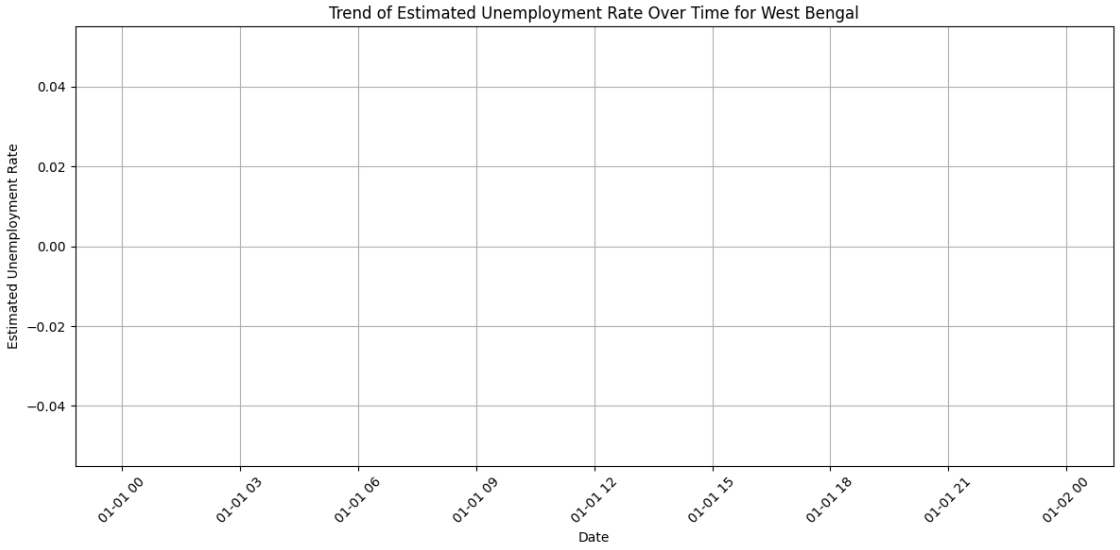
[138]:
```
region=['Tripura','Meghalaya','Telangana','Gujarat','West Bengal','Himachal␣
 ↪Pradesh'
,'Chhattisgarh' ,'Haryana'
,'Maharashtra','Jharkhand','Assam','Karnataka'
,'Punjab','Tamil Nadu','Rajasthan','Uttar Prades'
,'Andhra Pradesh','Delhi','Odisha','Madhya Pradesh'
,'Bihar','Puducherry','Kerala','Goa']
for i in region:
   region_data = data[data['Region'] .isin([i])].copy()
   region_data['Date'] = pd.to_datetime(region_data['Date'], format='%d-%m-%Y')
   region_data.sort_values('Date', inplace=True)

   # Create a line plot
   plt.figure(figsize=(12, 6))
   plt.plot(region_data['Date'], region_data['Unemploy_Rate'], marker='o',␣
 ↪linestyle='-')
   plt.xlabel('Date')
   plt.ylabel('Estimated Unemployment Rate')
   plt.title(f'Trend of Estimated Unemployment Rate Over Time for {i}')
   plt.grid(True)
   plt.xticks(rotation=45)
   plt.tight_layout()

   # Display the plot
   plt.show()
```

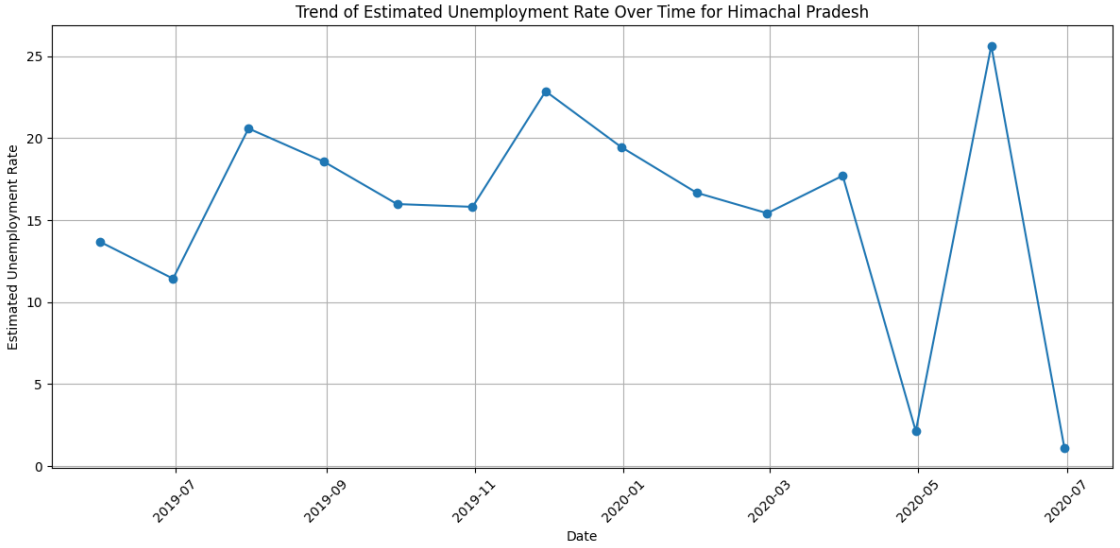

Trend of Estimated Unemployment Rate Over Time for Tripura
Estimated Unemployment Rate
0.04
0.02
0.00
−0.02
−0.04
01-01 00
01-01 03
01-01 06
01-01 09
01-01 12
01-01 15
01-01 18
01-01 21
01-02 00
Date

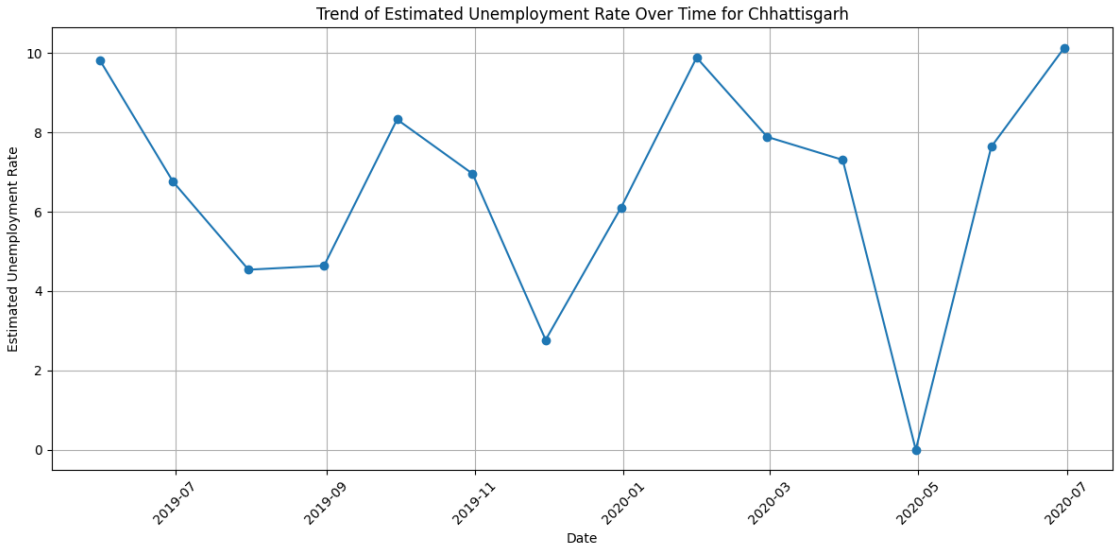

Trend of Estimated Unemployment Rate Over Time for Meghalaya
Estimated Unemployment Rate
8
7
6
5
4
3
2
1
0
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

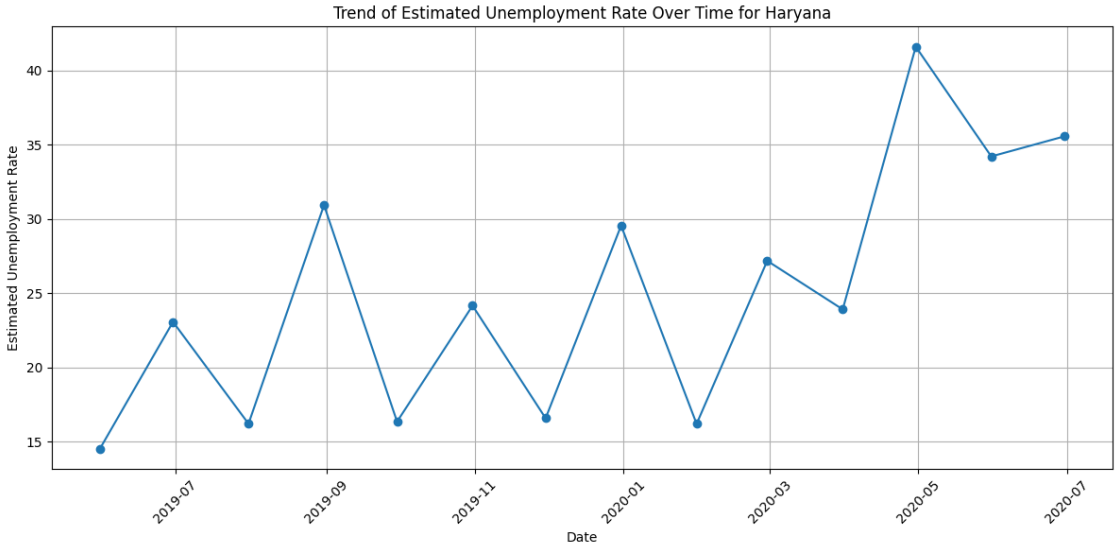

Trend of Estimated Unemployment Rate Over Time for Telangana
Estimated Unemployment Rate
0.04
0.02
0.00
−0.02
−0.04
01-01 00
01-01 03
01-01 06
01-01 09
01-01 12
01-01 15
01-01 18
01-01 21
01-02 00
Date

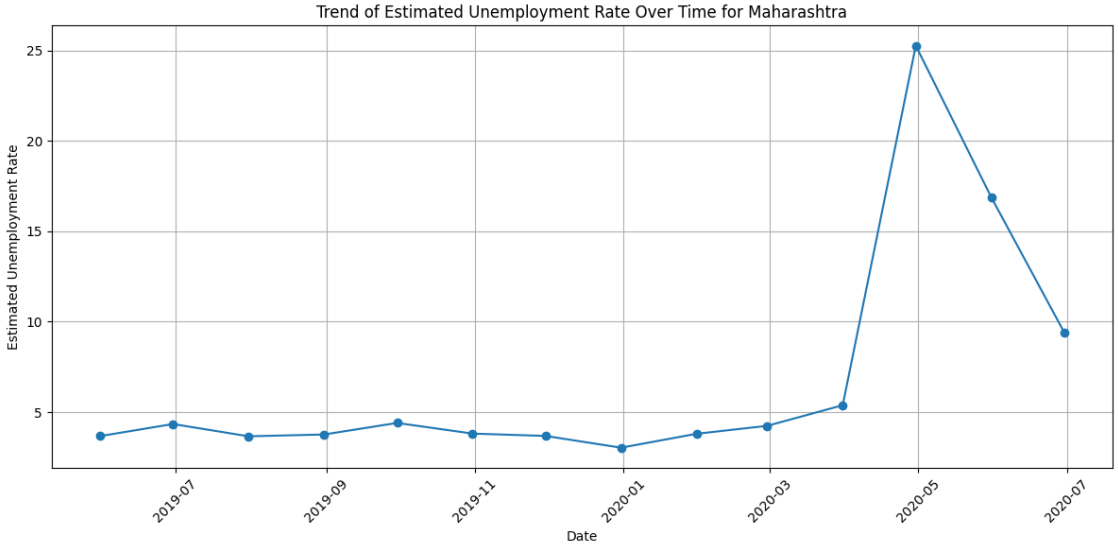

Trend of Estimated Unemployment Rate Over Time for Gujarat
Estimated Unemployment Rate
14
12
10
8
6
4
2
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

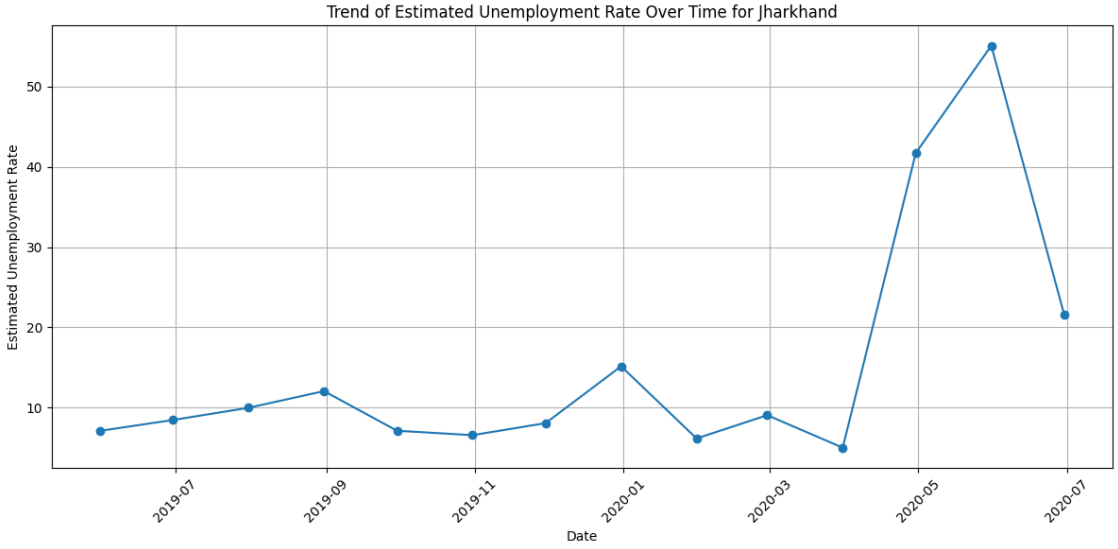

Trend of Estimated Unemployment Rate Over Time for West Bengal
Estimated Unemployment Rate
0.04
0.02
0.00
−0.02
−0.04
01-01 00
01-01 03
01-01 06
01-01 09
01-01 12
01-01 15
01-01 18
01-01 21
01-02 00
Date

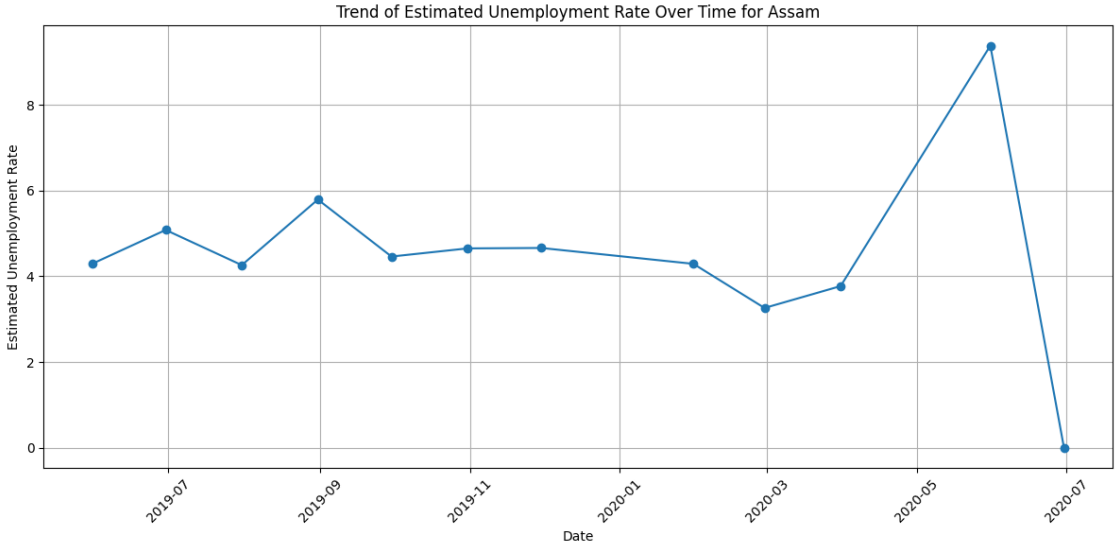

Trend of Estimated Unemployment Rate Over Time for Himachal Pradesh
Estimated Unemployment Rate
25
20
15
10
5
0
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

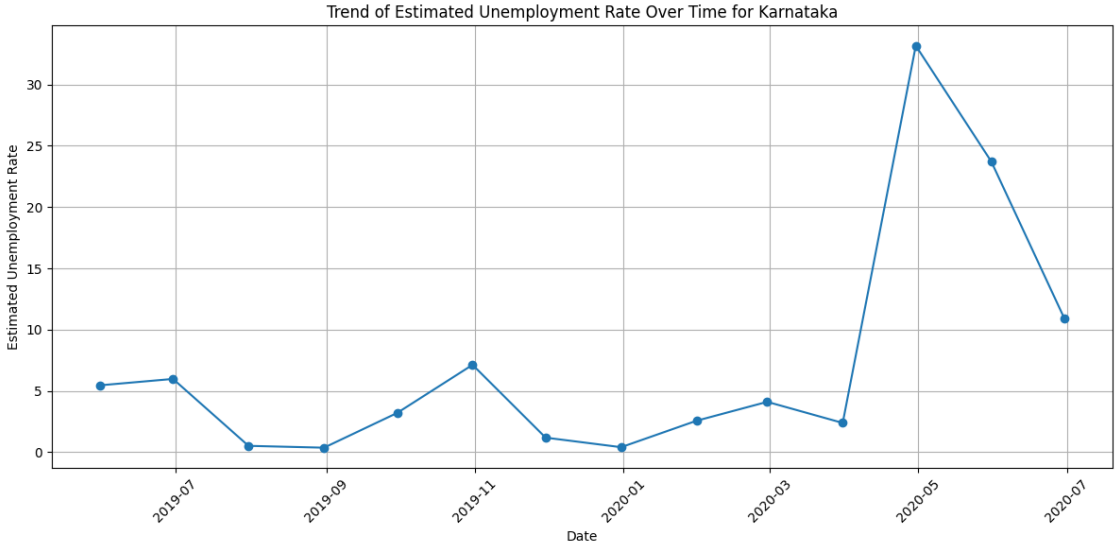

Trend of Estimated Unemployment Rate Over Time for Chhattisgarh
Estimated Unemployment Rate
Date
0
2
4
6
8
10
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07

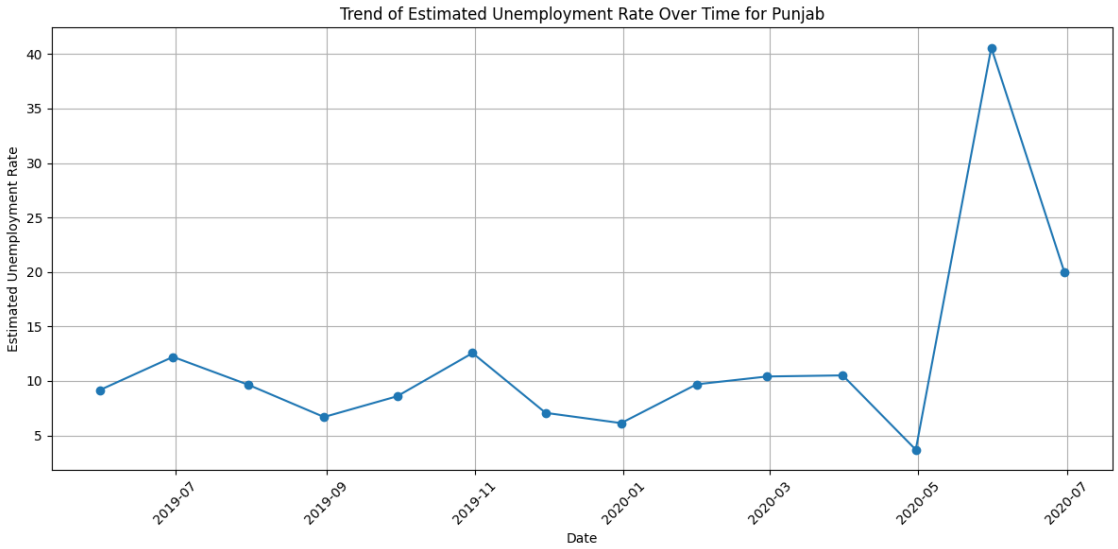

Trend of Estimated Unemployment Rate Over Time for Haryana
Estimated Unemployment Rate
Date
15
20
25
30
35
40
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07

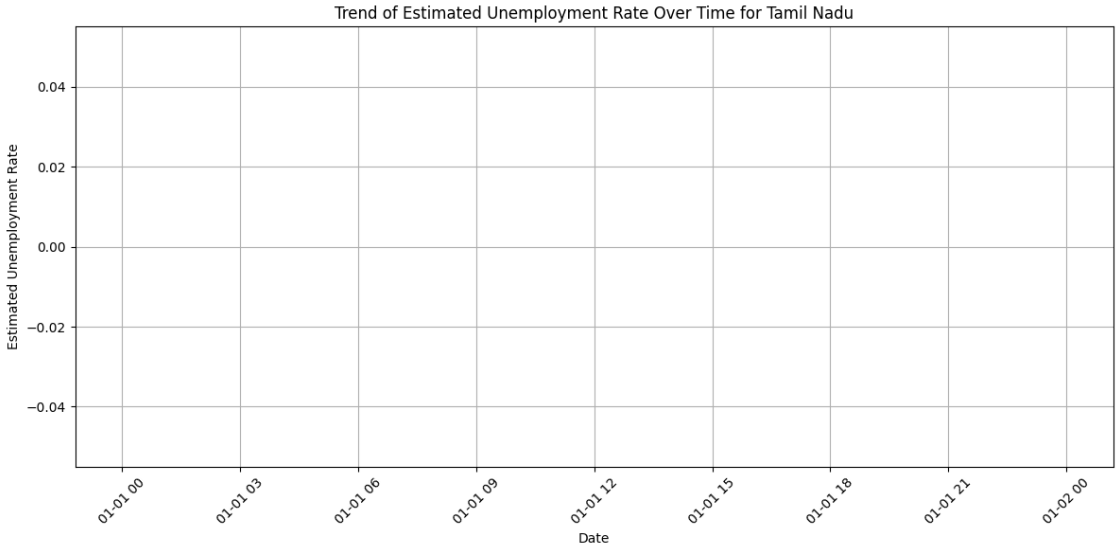

Trend of Estimated Unemployment Rate Over Time for Maharashtra
Estimated Unemployment Rate
25
20
15
10
5
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

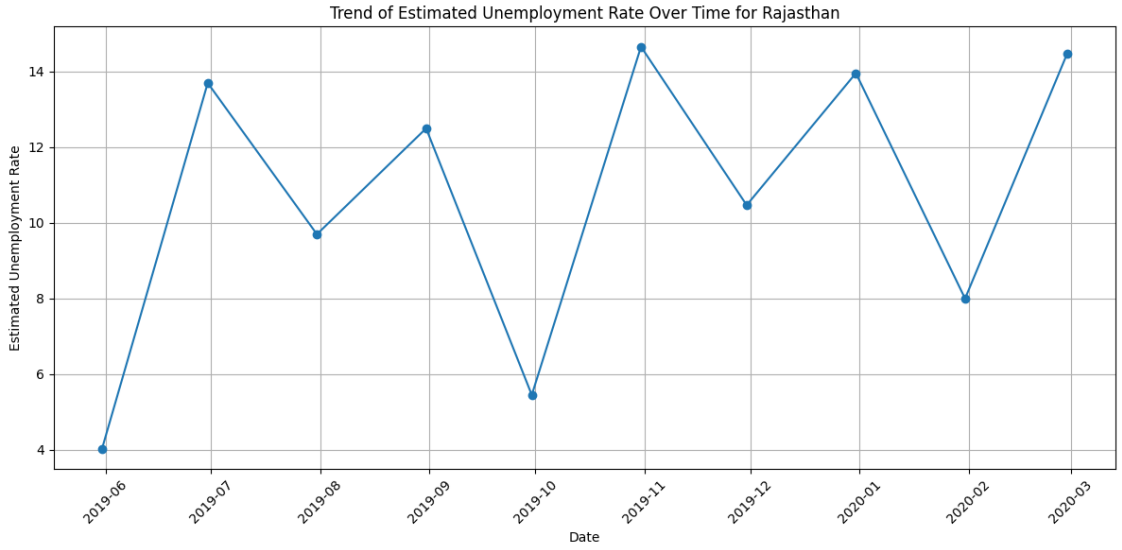

Trend of Estimated Unemployment Rate Over Time for Jharkhand
Estimated Unemployment Rate
50
40
30
20
10
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

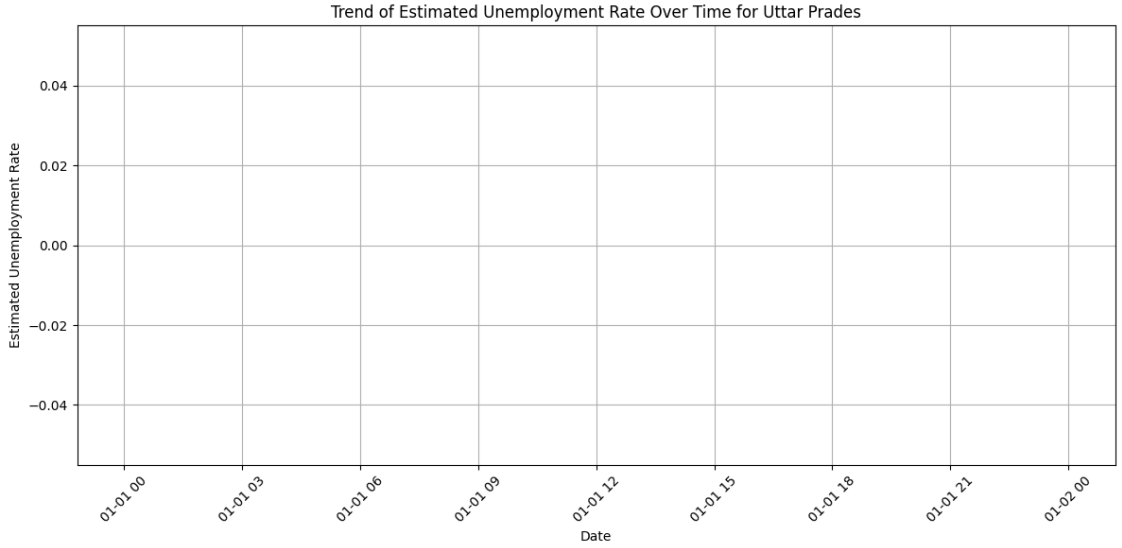

Trend of Estimated Unemployment Rate Over Time for Assam
Estimated Unemployment Rate
0
2
4
6
8
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

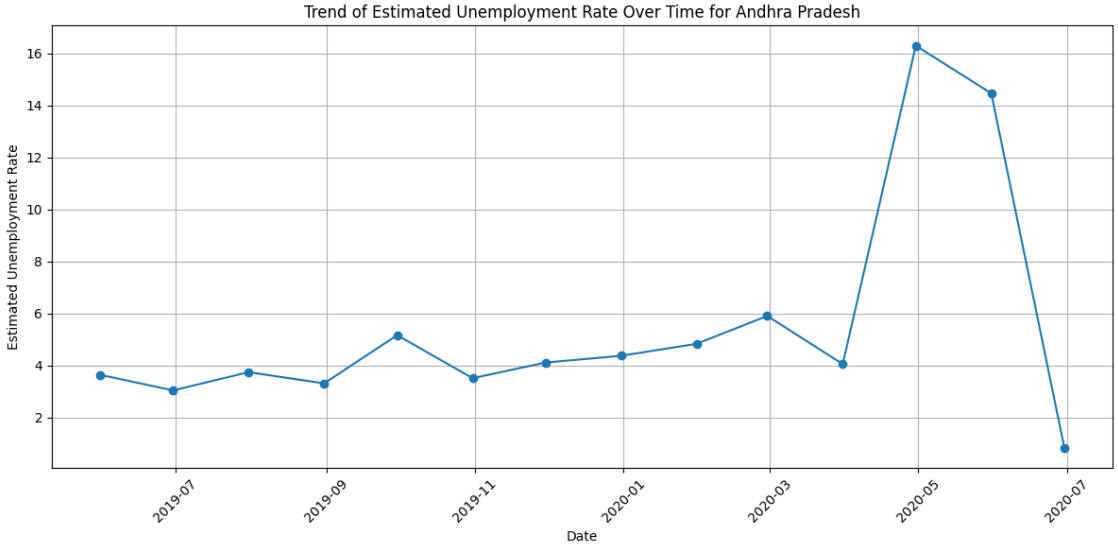

Trend of Estimated Unemployment Rate Over Time for Karnataka
Estimated Unemployment Rate
0
5
10
15
20
25
30
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

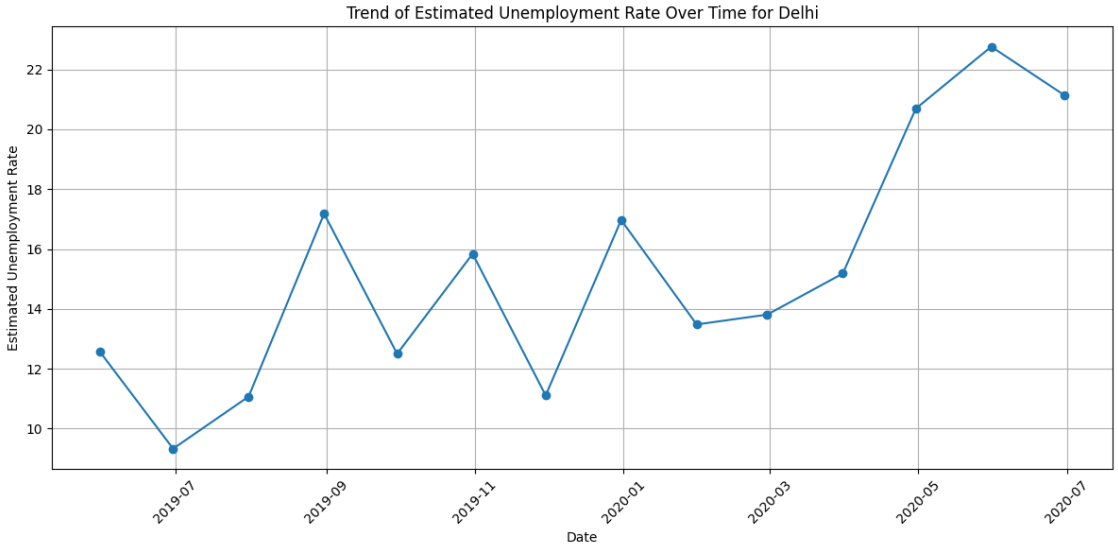

Trend of Estimated Unemployment Rate Over Time for Punjab
Estimated Unemployment Rate
40
35
30
25
20
15
10
5
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date

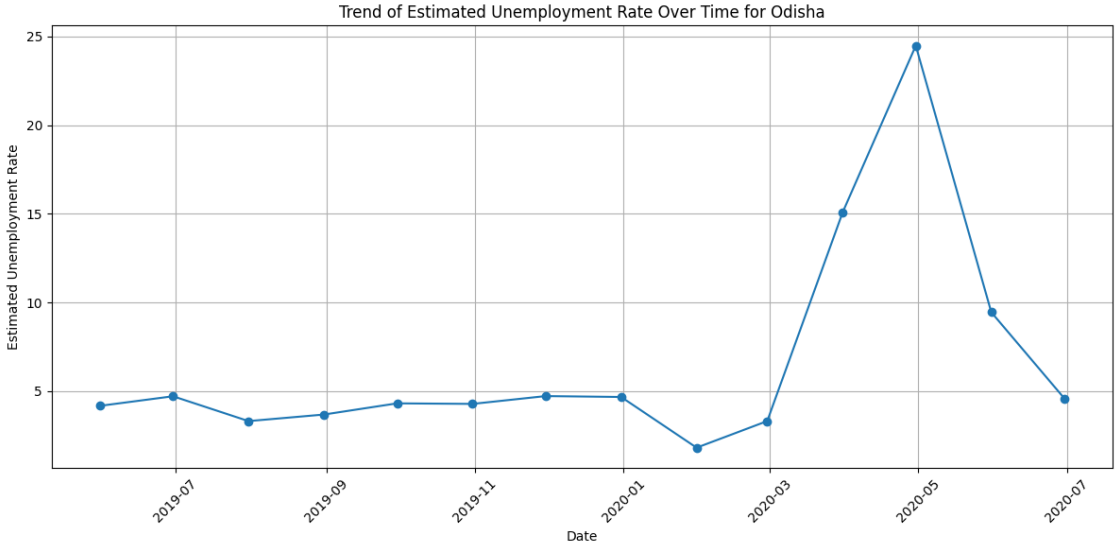

Trend of Estimated Unemployment Rate Over Time for Tamil Nadu
Estimated Unemployment Rate
0.04
0.02
0.00
−0.02
−0.04
01-01 00
01-01 03
01-01 06
01-01 09
01-01 12
01-01 15
01-01 18
01-01 21
01-02 00
Date

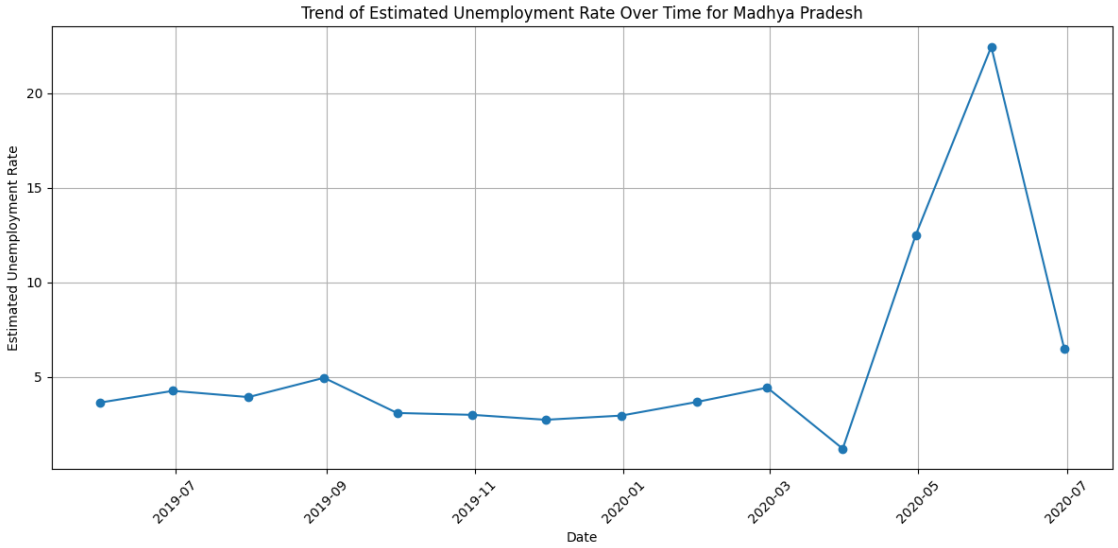

Trend of Estimated Unemployment Rate Over Time for Rajasthan
Estimated Unemployment Rate
14
12
10
8
6
4
2019-06
2019-07
2019-08
2019-09
2019-10
2019-11
2019-12
2020-01
2020-02
2020-03
Date

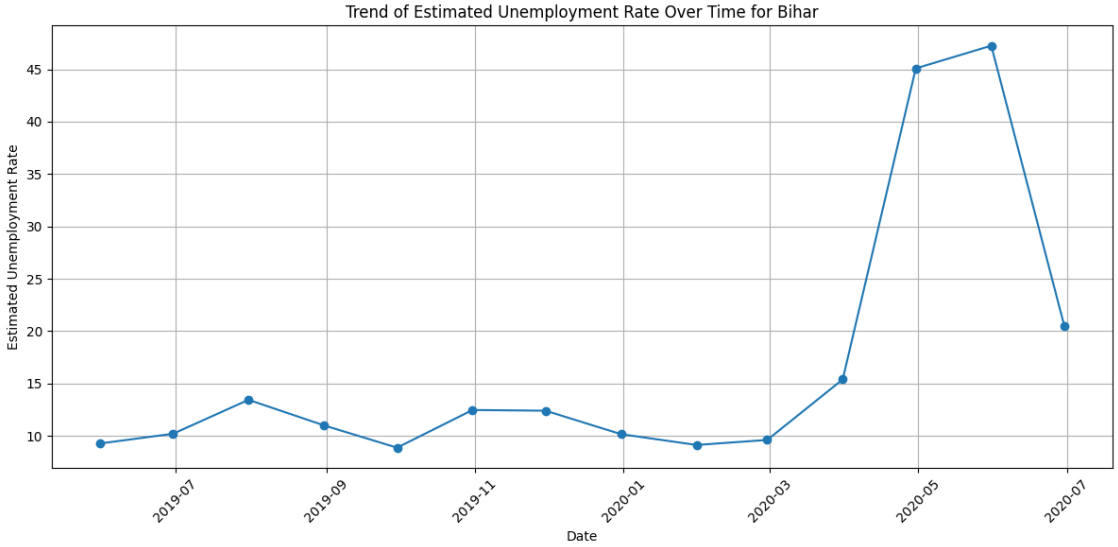

Trend of Estimated Unemployment Rate Over Time for Uttar Prades
Estimated Unemployment Rate
0.04
0.02
0.00
−0.02
−0.04
01-01 00
01-01 03
01-01 06
01-01 09
01-01 12
01-01 15
01-01 18
01-01 21
01-02 00
Date

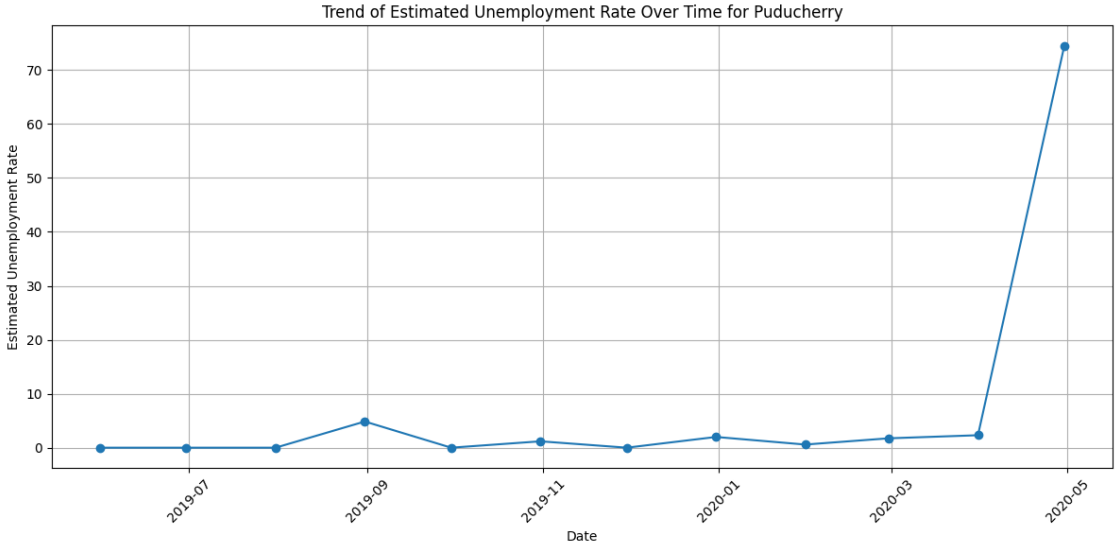

Trend of Estimated Unemployment Rate Over Time for Andhra Pradesh
Estimated Unemployment Rate
Date
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07

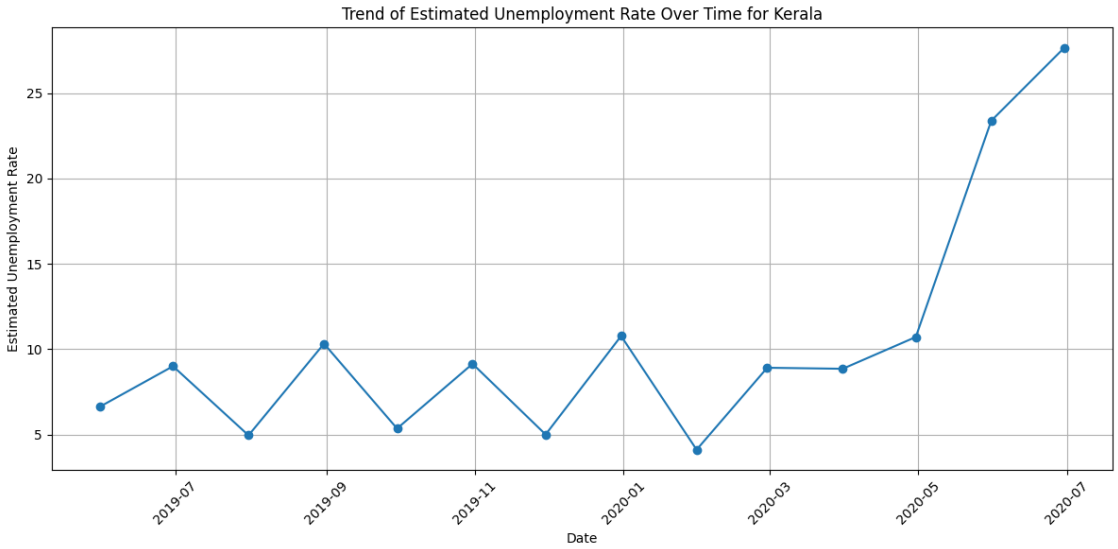

Trend of Estimated Unemployment Rate Over Time for Delhi
Estimated Unemployment Rate
Date
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07

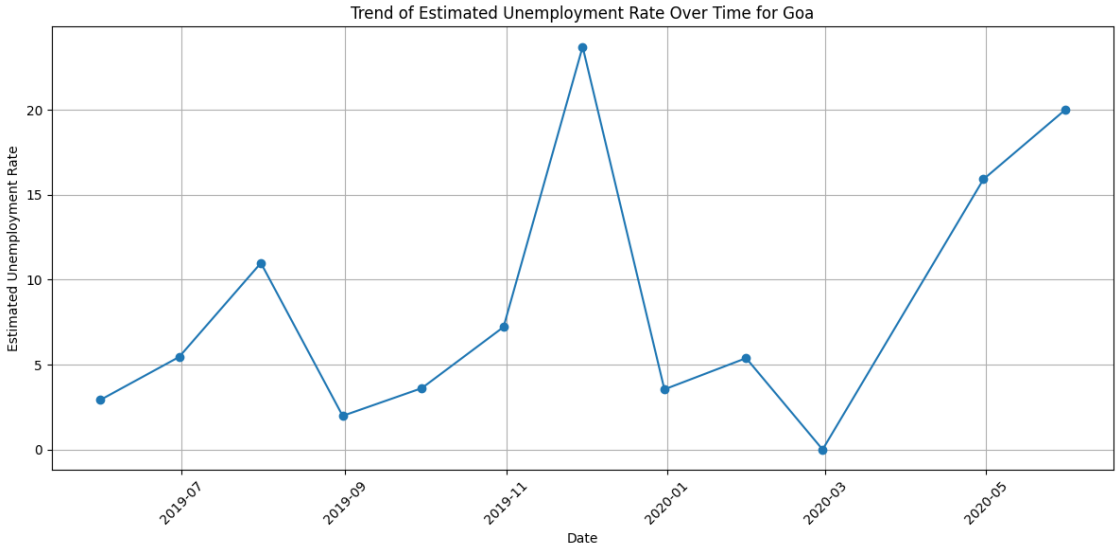

Trend of Estimated Unemployment Rate Over Time for Odisha
Estimated Unemployment Rate
25
20
15
10
5
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date


Trend of Estimated Unemployment Rate Over Time for Madhya Pradesh
Estimated Unemployment Rate
20
15
10
5
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07
Date


Trend of Estimated Unemployment Rate Over Time for Bihar
Estimated Unemployment Rate
Date
10
15
20
25
30
35
40
45
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05
2020-07


Trend of Estimated Unemployment Rate Over Time for Puducherry
Estimated Unemployment Rate
Date
0
10
20
30
40
50
60
70
2019-07
2019-09
2019-11
2020-01
2020-03
2020-05

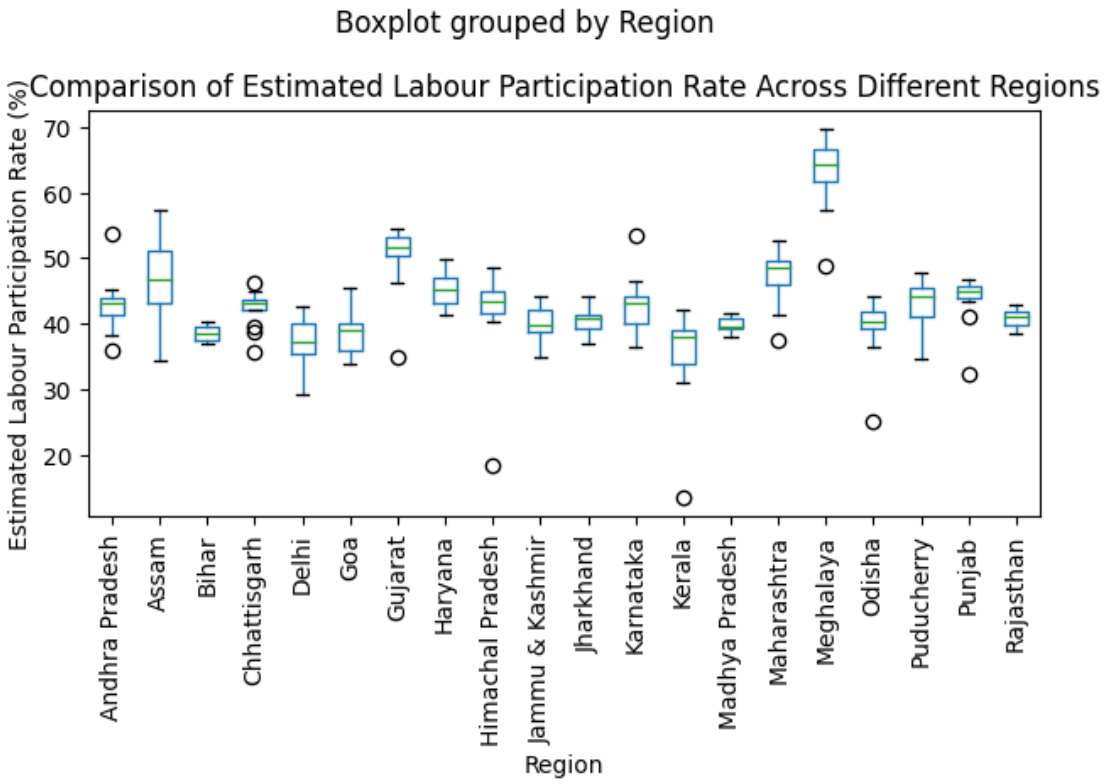




[139]:
```
# Compare the estimated labour participation rate across different regions

# Create a box plot
plt.figure(figsize=(20, 7))
data.boxplot(column='Labr_Patcptn_Rate', by='Region', grid=False, rot=90)
plt.xlabel('Region')
plt.ylabel('Estimated Labour Participation Rate (%)')
plt.title('Comparison of Estimated Labour Participation Rate Across Different␣
 ↪Regions')
plt.tight_layout()
```

```
plt.show()
```

```
<Figure size 2000x700 with 0 Axes>
```



[147]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer # Import the SimpleImputer class
# Features and target variable
X = df[['Labr_Patcptn_Rate', 'Employed', ]]
y = df['Unemploy_Rate']

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Create an imputer to fill missing values with the mean
imputer = SimpleImputer(strategy='mean')

# Fit the imputer on the training data and transform both training and test data
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
# Impute missing values in the target variable (y_train)
y_train = imputer.fit_transform(y_train.values.reshape(-1, 1))
```

```
# Reshape y_train to a 2D array for the imputer


print(f"Training set size: {X_train.shape[0]}")
print(f"Test set size: {X_test.shape[0]}")
```

```
Training set size: 537
Test set size: 231
```

[149]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Create an imputer to fill missing values with the mean
imputer = SimpleImputer(strategy='mean')

# Fit the imputer on y_test and transform it
y_test = imputer.fit_transform(y_test.values.reshape(-1, 1))

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```
Mean Squared Error (MSE): 72.90
R-squared (R2): -0.09
```

[150]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

# Create a pipeline with polynomial features and linear regression
poly_model = Pipeline([
    ('poly', PolynomialFeatures(degree=2)),
    ('linear', LinearRegression())
])

# Train the refined model
poly_model.fit(X_train, y_train)
```

```
# Make predictions
y_pred_poly = poly_model.predict(X_test)

# Evaluate the refined model
mse_poly = mean_squared_error(y_test, y_pred_poly)
r2_poly = r2_score(y_test, y_pred_poly)

print(f"Polynomial Model - Mean Squared Error (MSE): {mse_poly:.2f}")
print(f"Polynomial Model - R-squared (R2): {r2_poly:.2f}")
```

```
Polynomial Model - Mean Squared Error (MSE): 71.08
Polynomial Model - R-squared (R2): -0.07
```

[151]:
```
# Final model evaluation on the test set
final_model = model  # or poly_model, depending on performance

y_final_pred = final_model.predict(X_test)
final_mse = mean_squared_error(y_test, y_final_pred)
final_r2 = r2_score(y_test, y_final_pred)

print(f"Final Model - Mean Squared Error (MSE): {final_mse:.2f}")
print(f"Final Model - R-squared (R2): {final_r2:.2f}")
```

```
Final Model - Mean Squared Error (MSE): 72.90
Final Model - R-squared (R2): -0.09
```