task-2

July 14, 2025

0.1 Task 2 : Unemployment Analysis with Python

0.2 1. Problem Statement

Unemployment is a key indicator of a country's economic health, reflecting the percentage of people in the labor force actively seeking but unable to find work. The COVID-19 pandemic caused a sharp spike in unemployment due to lockdowns and economic disruptions. This project aims to analyze unemployment trends during and after the pandemic to understand its impact and guide future recovery efforts.

[24]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

0.3 2. load Dataset

[63]:
```
# Load the dataset
df = pd.read_csv('Unemployment in India.csv')
```

[64]:
```
df
```

[64]:

	Region	Date	Frequency	Estimated Unemployment Rate (%)	\
0	Andhra Pradesh	31-05-2019	Monthly	3.65	
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	
..	…	…	…	…	
763	NaN	NaN	NaN	NaN	
764	NaN	NaN	NaN	NaN	
765	NaN	NaN	NaN	NaN	
766	NaN	NaN	NaN	NaN	
767	NaN	NaN	NaN	NaN	

	Estimated Employed	Estimated Labour Participation Rate (%)	Area
0	11999139.0	43.24	Rural
1	11755881.0	42.05	Rural
2	12086707.0	43.50	Rural

```
3             12285693.0                                    43.97  Rural
4             12256762.0                                    44.68  Rural
..                   …                                    …     …
763                 NaN                                      NaN    NaN
764                 NaN                                      NaN    NaN
765                 NaN                                      NaN    NaN
766                 NaN                                      NaN    NaN
767                 NaN                                      NaN    NaN

[768 rows x 7 columns]
```

0.4 3. Data Exploration

1. View dataset shape and sample entries.
2. ummary statistics and null value checks.

[65]:
```
print(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
None
```

[66]:
```
df.describe()
```

[66]:
```
       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
```

[67]:
```
df.shape
```

[67]:
```
(768, 7)
```

[68]:
```
df.dtypes
```

[68]:
```
Region                                    object
 Date                                     object
 Frequency                                object
 Estimated Unemployment Rate (%)         float64
 Estimated Employed                      float64
 Estimated Labour Participation Rate (%) float64
Area                                      object
dtype: object
```

0.5 4 .Data Preprocessing.

1. Handling missing data.
2. Encoding categorical columns.

[69]:
```
df.isnull().sum()
```

[69]:
```
Region                                    28
 Date                                     28
 Frequency                                28
 Estimated Unemployment Rate (%)          28
 Estimated Employed                       28
 Estimated Labour Participation Rate (%)  28
Area                                      28
dtype: int64
```

[70]:
```
df=df.dropna()
```

[71]:
```
df.isnull().sum()
```

[71]:
```
Region                                    0
 Date                                     0
 Frequency                                0
```

```
 Estimated Unemployment Rate (%)            0
 Estimated Employed                         0
 Estimated Labour Participation Rate (%)    0
Area                                        0
dtype: int64
```

[72]:
```
# Clean column names
df.columns = df.columns.str.strip()
```

[73]:
```
df["Estimated Unemployment Rate (%)"]
```

[73]:
```
0       3.65
1       3.05
2       3.75
3       3.32
4       5.17
       ...
749     7.55
750     6.67
751    15.63
752    15.22
753     9.86
Name: Estimated Unemployment Rate (%), Length: 740, dtype: float64
```

[74]:
```
df.columns = df.columns.str.strip()
print(df.columns)
```

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

[75]:
```
# Strip spaces in column names
df.columns = df.columns.str.strip()

# Convert 'Date' column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Check column names and types
print(df.dtypes)
```

```
Region                                     object
Date                               datetime64[ns]
Frequency                                  object
Estimated Unemployment Rate (%)           float64
Estimated Employed                        float64
Estimated Labour Participation Rate (%)   float64
```

```
Area                                                    object
dtype: object
```

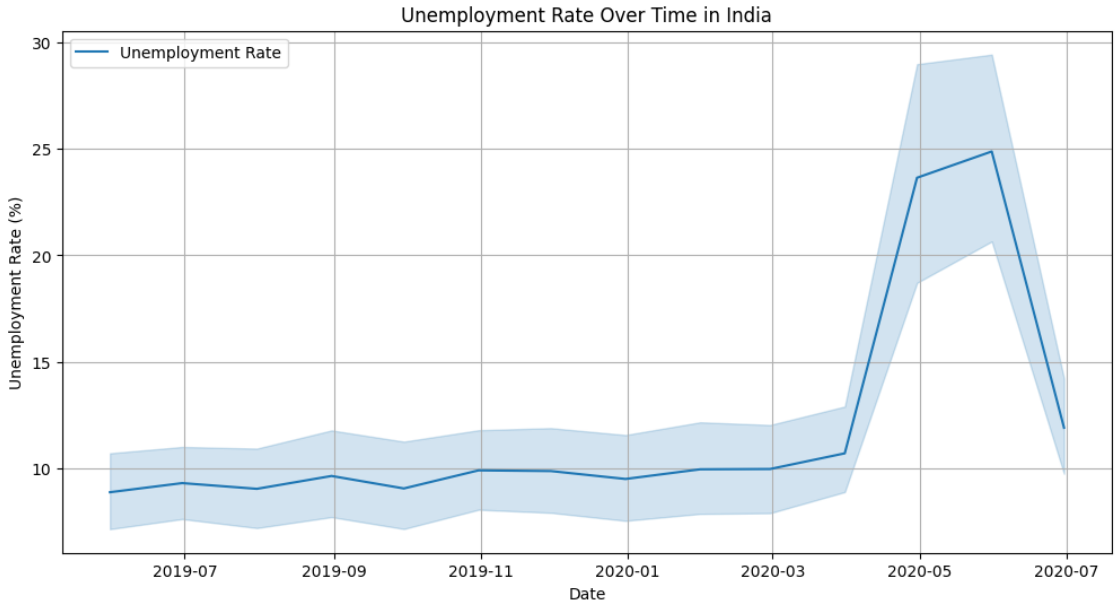
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\3159115824.py:5: UserWarning:
Parsing dates in  %d-%m-%Y format when dayfirst=False (the default) was
specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df['Date'] = pd.to_datetime(df['Date'])
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\3159115824.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Date'] = pd.to_datetime(df['Date'])
```

0.6 5. Data Visualization

1. Line plot: Unemployment rate over time
2. Bar chart: Region-wise unemployment comparison
3. Heatmap: Correlation matrix
4. Boxplot & Histogram - Distribution
5. Correlation Analysis (if more numeric columns exist)

0.7 Line Plot - Overall Unemployment Trends

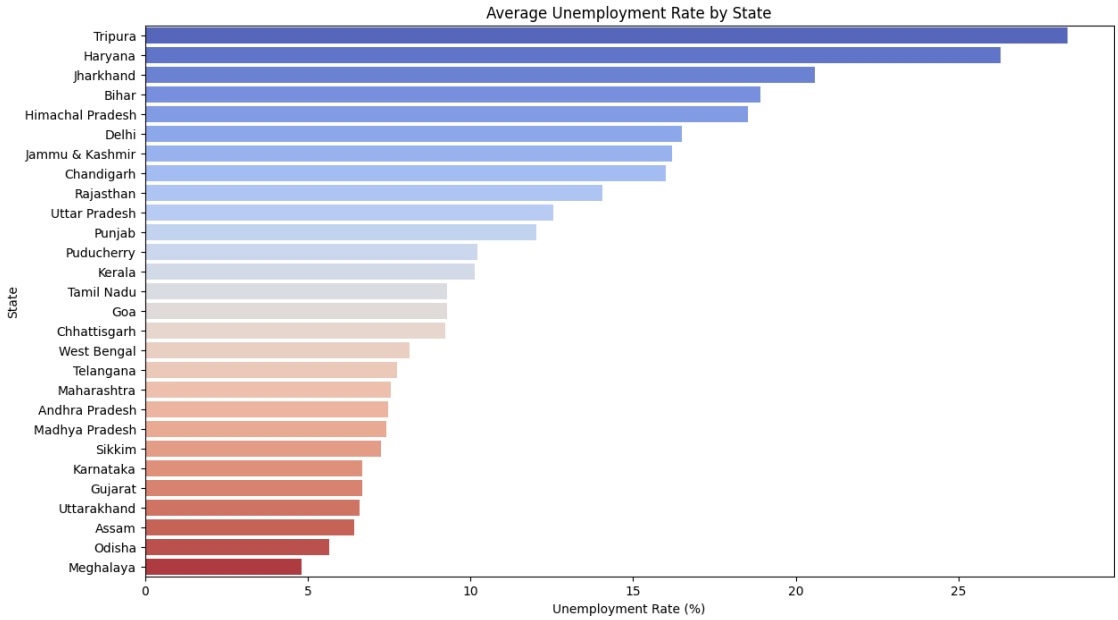
[76]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(data=df, x='Date', y='Estimated Unemployment Rate (%)',␣
 ↪label='Unemployment Rate')
plt.title('Unemployment Rate Over Time in India')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate (%)')
plt.grid(True)
plt.show()
```



0.8 Bar Plot - Statewise Average Unemployment

[77]:
```
statewise = df.groupby('Region')['Estimated Unemployment Rate (%)'].mean().
 ↪sort_values(ascending=False)

plt.figure(figsize=(14, 8))
sns.barplot(x=statewise.values, y=statewise.index, palette='coolwarm')
plt.title('Average Unemployment Rate by State')
plt.xlabel('Unemployment Rate (%)')
plt.ylabel('State')
plt.show()
```

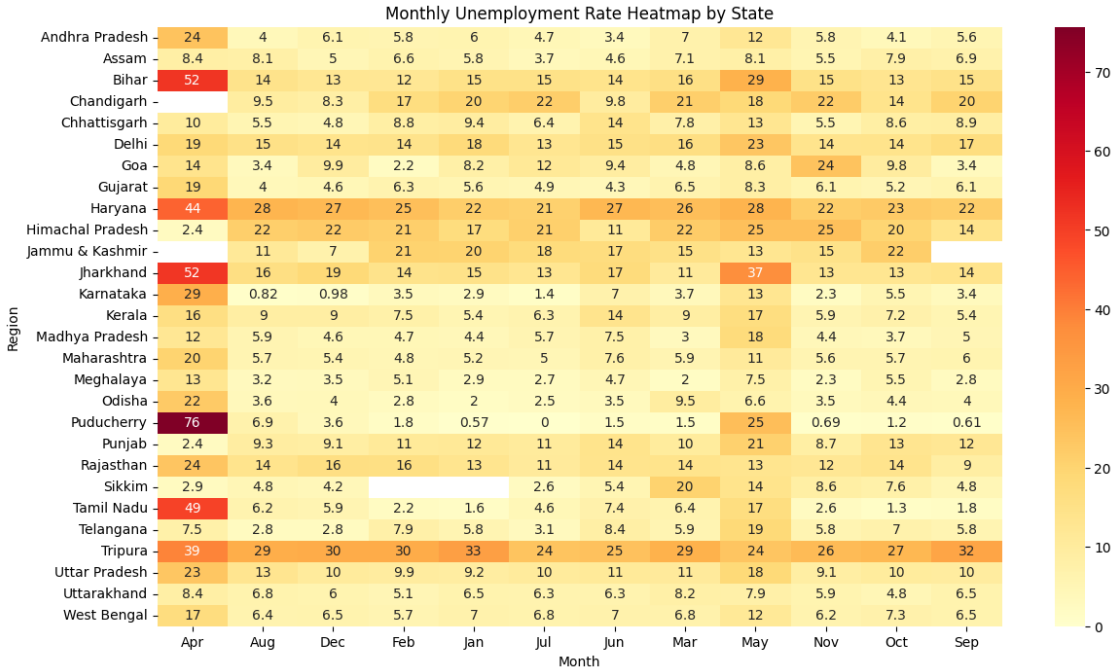
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\3713165549.py:4:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=statewise.values, y=statewise.index, palette='coolwarm')
```



0.9 Heatmap - Unemployment by Region & Month

[78]:
```
df['Month'] = df['Date'].dt.strftime('%b')
pivot_data = df.pivot_table(index='Region', columns='Month', values='Estimated␣
 ↪Unemployment Rate (%)')

plt.figure(figsize=(14, 8))
sns.heatmap(pivot_data, annot=True, cmap='YlOrRd')
plt.title('Monthly Unemployment Rate Heatmap by State')
plt.show()
```

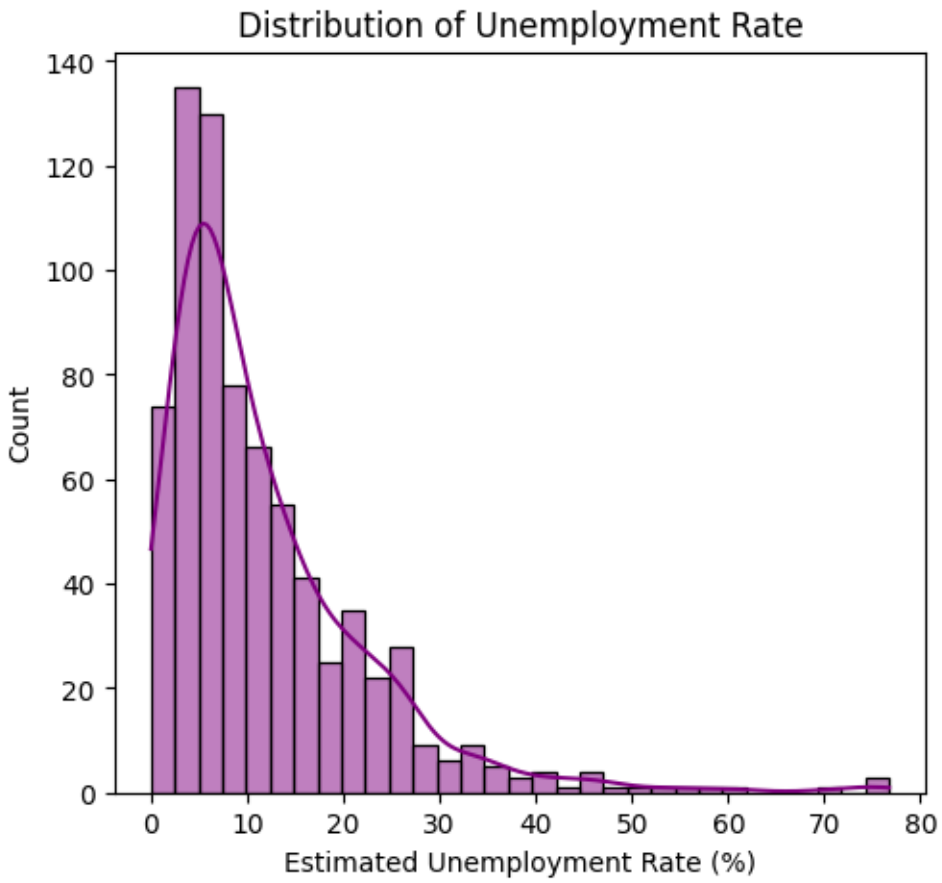
```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\1002796816.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Month'] = df['Date'].dt.strftime('%b')
```



0.10 Boxplot & Histogram - Distribution

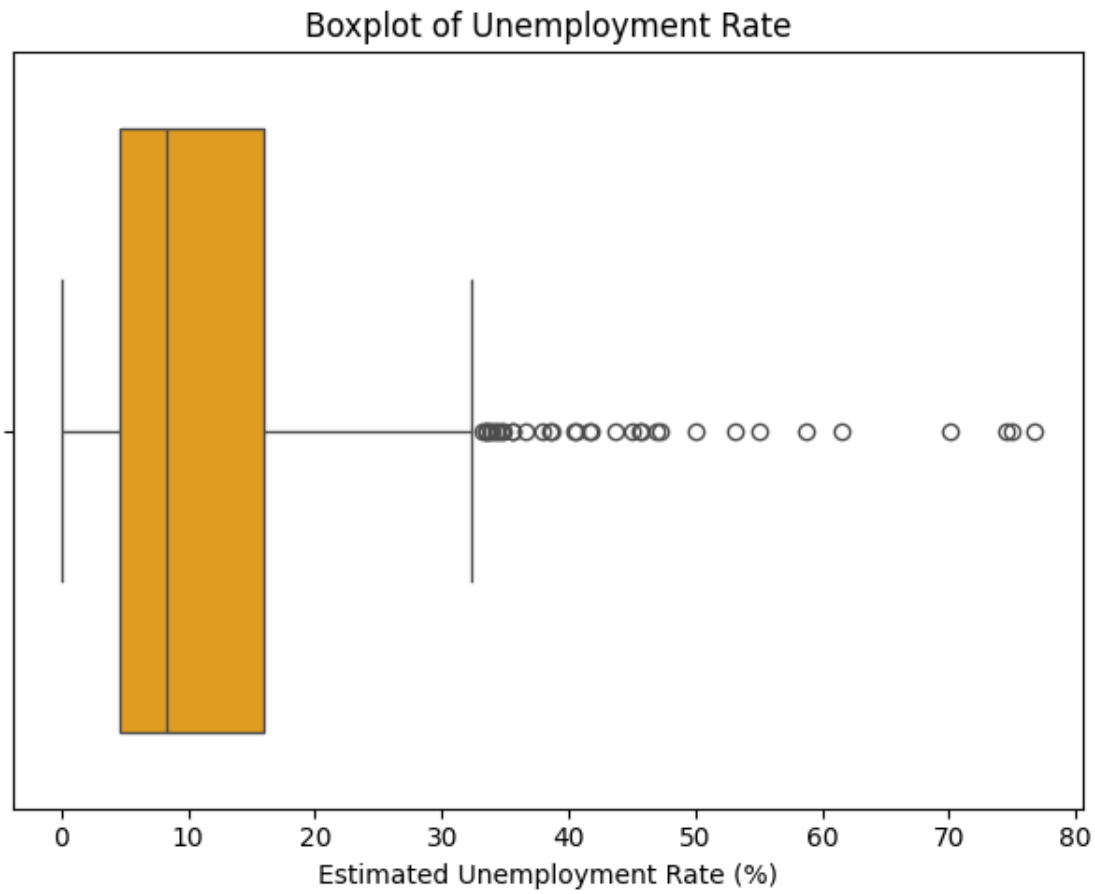
[79]:
```
plt.figure(figsize=(12, 5))

# Histogram
plt.subplot(1, 2, 1)
sns.histplot(df['Estimated Unemployment Rate (%)'], kde=True, color='purple')
plt.title('Distribution of Unemployment Rate')
plt.figure(figsize=(12, 5))
```

[79]:
```
<Figure size 1200x500 with 0 Axes>
```



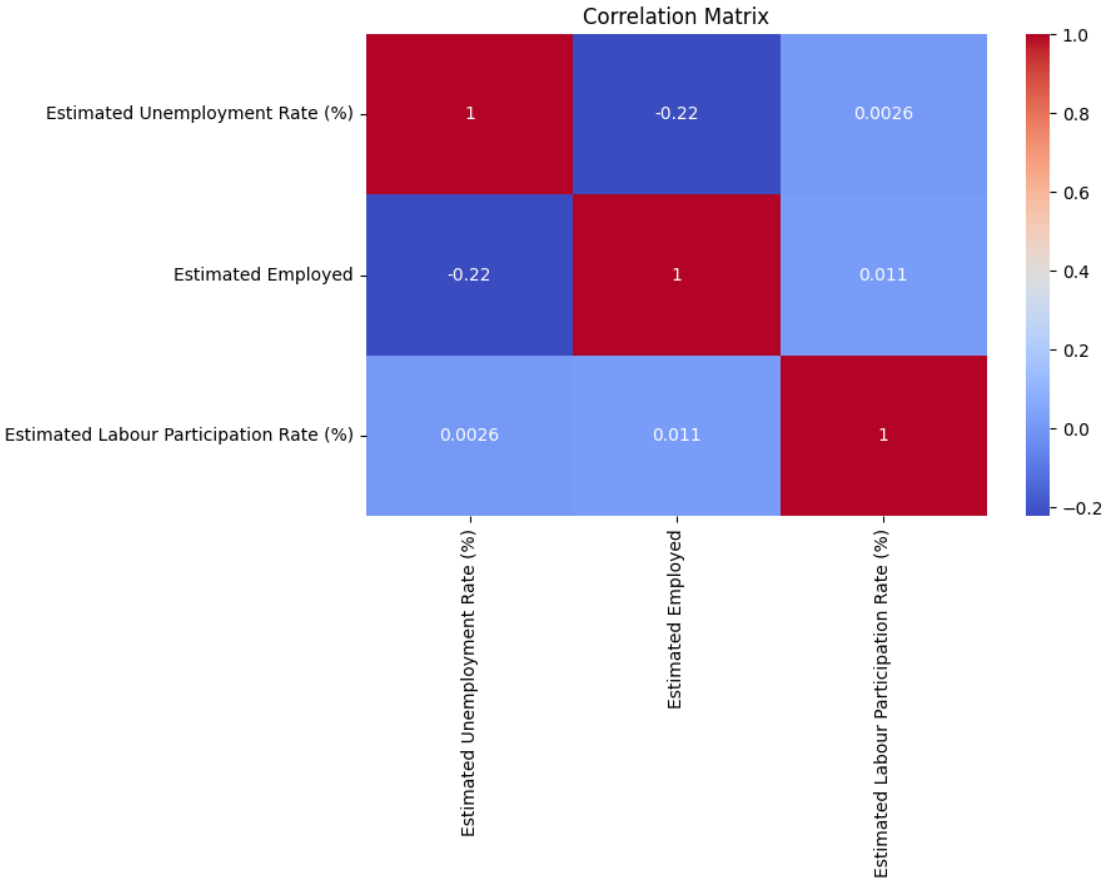

```
<Figure size 1200x500 with 0 Axes>
```

[80]:
```
# Boxplot
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
sns.boxplot(x='Estimated Unemployment Rate (%)', data=df, color='orange')
plt.title('Boxplot of Unemployment Rate')

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



0.11 Correlation Analysis (if more numeric columns exist)

[81]:
```
plt.figure(figsize=(8, 5))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



0.12 6. Forecasting Future Trends

Time Series Forecasting with ARIMA ,Model evaluation (RMSE, MAE, etc.)

[82]:
```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
```

[83]:
```
monthly_data = df.groupby('Date')['Estimated Unemployment Rate (%)'].mean()
monthly_data = monthly_data.asfreq('MS')  # Set monthly frequency
```

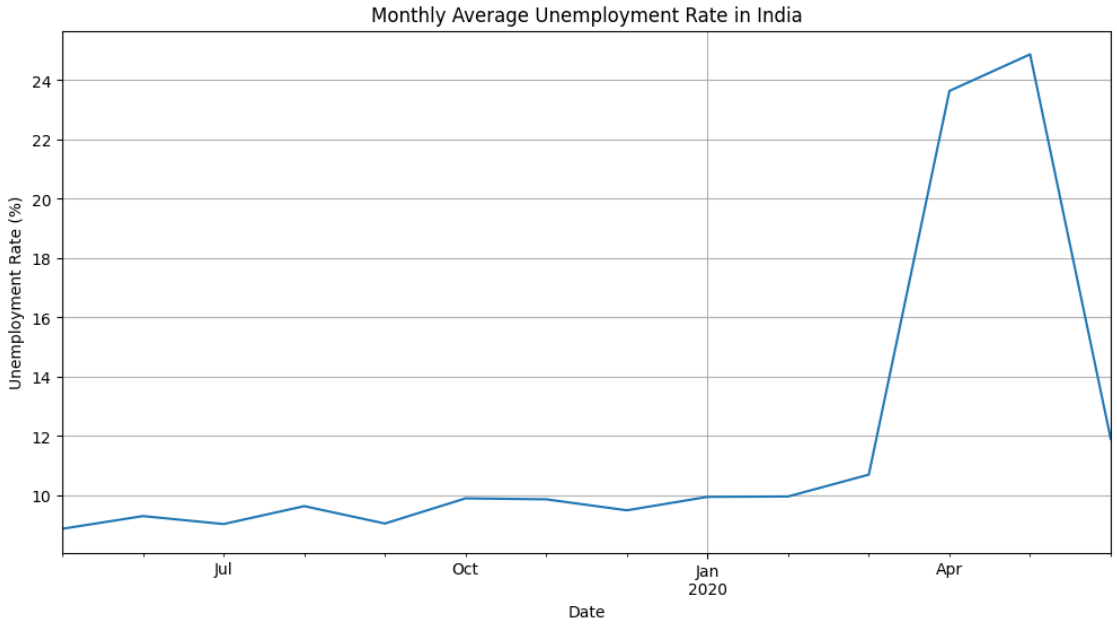
[84]:
```
print(df.columns)
```

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

[85]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# Assuming df is your main DataFrame
df['Date'] = pd.to_datetime(df['Date'])  # Correct column name (no leading␣
 ↪space)
df.set_index('Date', inplace=True)

# Resample monthly and calculate mean unemployment rate
monthly_data = df['Estimated Unemployment Rate (%)'].resample('M').mean()

# Plot
plt.figure(figsize=(12, 6))
monthly_data.plot()
plt.title('Monthly Average Unemployment Rate in India')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate (%)')
plt.grid(True)
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\2250555069.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Date'] = pd.to_datetime(df['Date'])  # Correct column name (no leading
space)
C:\Users\Admin\AppData\Local\Temp\ipykernel_11916\2250555069.py:9:
FutureWarning: 'M' is deprecated and will be removed in a future version, please
use 'ME' instead.
  monthly_data = df['Estimated Unemployment Rate (%)'].resample('M').mean()
```



[ ]:

[ ]:

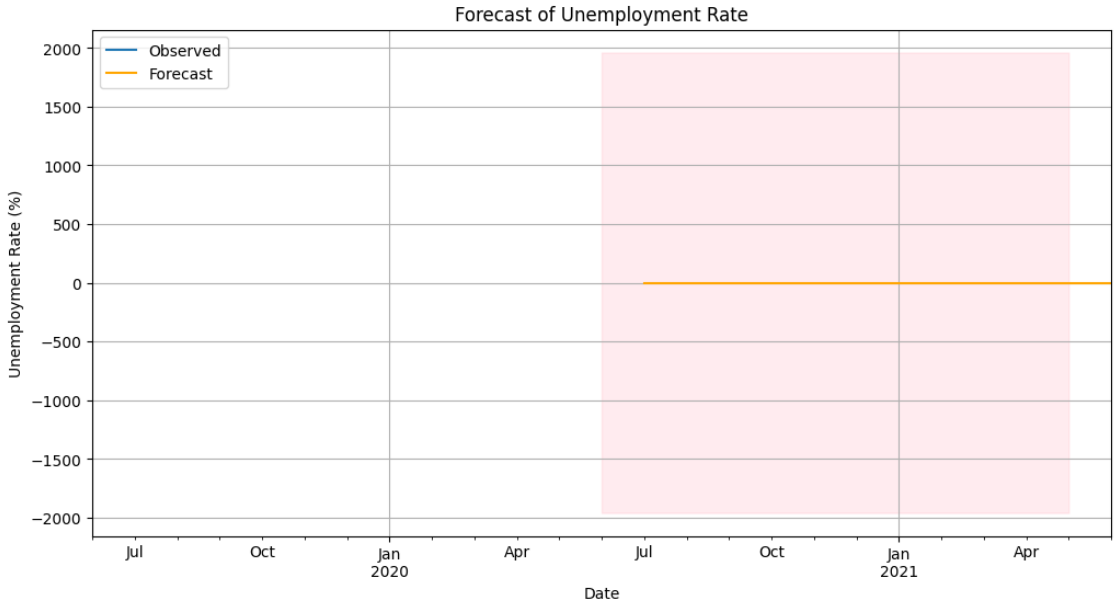
[86]:
```
model = ARIMA(monthly_data, order=(1, 1, 1))  # (p,d,q) - adjust as needed
model_fit = model.fit()
print(model_fit.summary())
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:966: UserWarning: Non-stationary
starting autoregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible
starting MA parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'

                                SARIMAX Results
==============================================================================
===========
Dep. Variable:     Estimated Unemployment Rate (%)   No. Observations:
14
Model:                          ARIMA(1, 1, 1)   Log Likelihood
-38.201
Date:                         Sun, 13 Jul 2025   AIC
82.403
Time:                                 15:22:52   BIC
```

```
84.098
Sample:                                 05-31-2019   HQIC
82.055
                                      - 06-30-2020
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4223      0.813     -0.520      0.603      -2.015       1.171
ma.L1          0.9992     81.140      0.012      0.990    -158.032     160.030
sigma2        18.1509   1462.038      0.012      0.990   -2847.391    2883.693
===================================================================================
===
Ljung-Box (L1) (Q):                   0.16   Jarque-Bera (JB):
8.73
Prob(Q):                              0.69   Prob(JB):
0.01
Heteroskedasticity (H):             146.38   Skew:
1.27
Prob(H) (two-sided):                  0.00   Kurtosis:
6.11
===================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

[56]:
```
# Forecast next 12 months
forecast = model_fit.get_forecast(steps=12)
forecast_index = pd.date_range(start=monthly_data.index[-1], periods=12,␣
 ↪freq='MS')

# Confidence intervals
conf_int = forecast.conf_int()

# Plot
plt.figure(figsize=(12, 6))
monthly_data.plot(label='Observed')
forecast.predicted_mean.plot(label='Forecast', color='orange')
plt.fill_between(forecast_index,
                 conf_int.iloc[:, 0],
                 conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.title('Forecast of Unemployment Rate')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate (%)')
plt.legend()
```

```
plt.grid(True)
plt.show()
```



0.13 Prophet

[88]:
```
from prophet import Prophet
import pandas as pd

# Prepare data in Prophet format: columns must be 'ds' (date) and 'y' (value)
prophet_data = monthly_data.reset_index()
prophet_data.columns = ['ds', 'y']  # Prophet expects 'ds' for date and 'y' for␣
 ↪value

prophet_data.head()
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Importing plotly failed. Interactive plots will not work.
```

[88]:

	ds	y
0	2019-05-31	8.874259
1	2019-06-30	9.303333
2	2019-07-31	9.033889
3	2019-08-31	9.637925

```
4 2019-09-30  9.051731
```

[89]:
```
model = Prophet()
model.fit(prophet_data)
```

```
15:26:54 - cmdstanpy - INFO - Chain [1] start processing
15:26:56 - cmdstanpy - INFO - Chain [1] done processing
```

[89]:
```
<prophet.forecaster.Prophet at 0x20e2dbd7bc0>
```

[90]:
```
# Create future dataframe
future = model.make_future_dataframe(periods=12, freq='M')

# Forecast
forecast = model.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

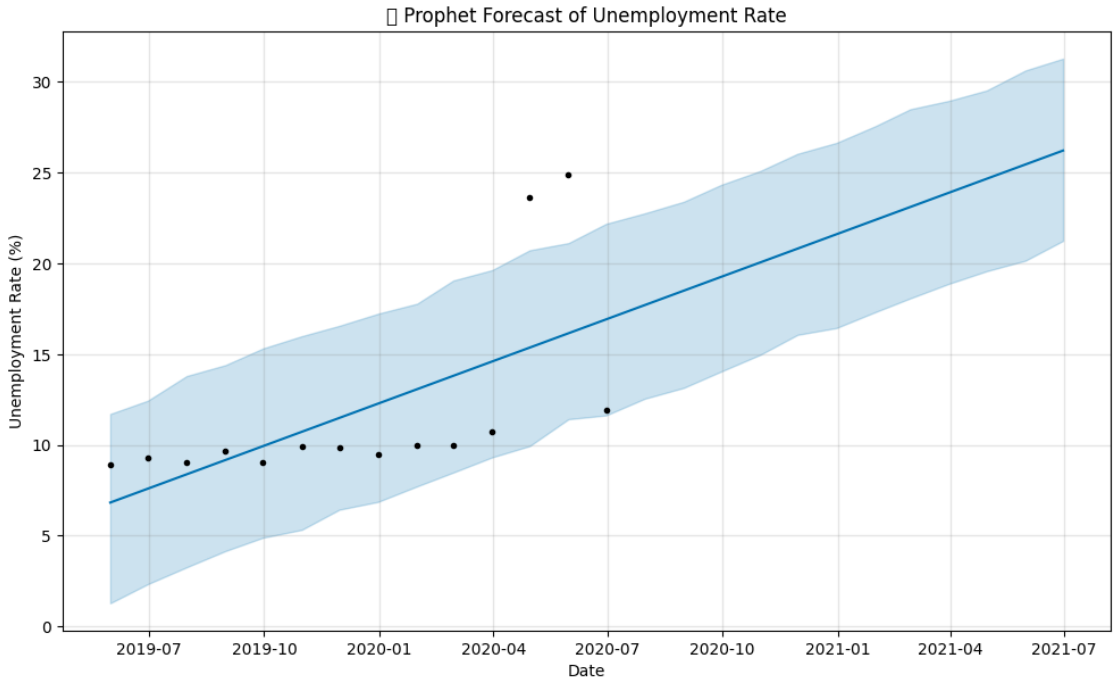
```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\prophet\forecaster.py:1854: FutureWarning: 'M' is deprecated and will
be removed in a future version, please use 'ME' instead.
  dates = pd.date_range(
```

[90]:

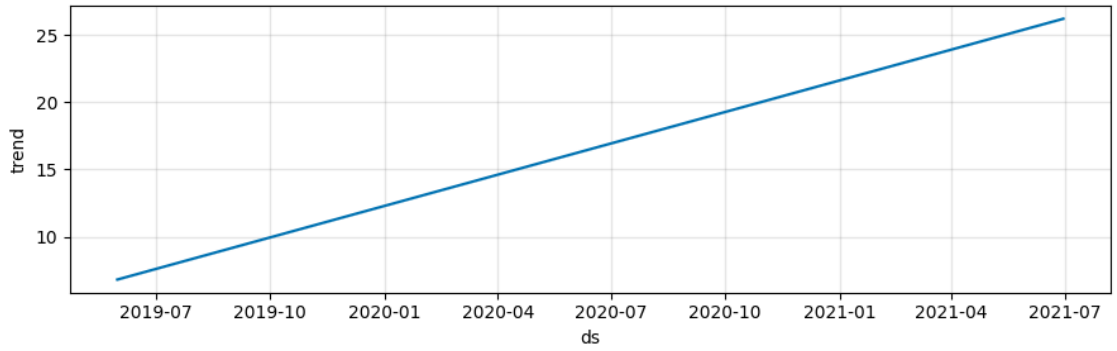
	ds	yhat	yhat_lower	yhat_upper
21	2021-02-28	23.104012	18.071212	28.494223
22	2021-03-31	23.894099	18.877556	28.958270
23	2021-04-30	24.658700	19.568176	29.528726
24	2021-05-31	25.448787	20.148714	30.630733
25	2021-06-30	26.213388	21.238946	31.280271

[91]:
```
fig1 = model.plot(forecast)
plt.title('  Prophet Forecast of Unemployment Rate')
plt.xlabel('Date')
plt.ylabel('Unemployment Rate (%)')
plt.grid(True)
plt.show()
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Glyph 128201 (\N{CHART
WITH DOWNWARDS TREND}) missing from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



[92]:
```
fig2 = model.plot_components(forecast)
plt.show()
```



0.14 Conclusion:

In this project, we used Facebook Prophet to forecast the monthly unemployment rate in India. Prophet is highly suitable for time series data that may include seasonality, trends, and irregularities, making it an ideal choice for this task.

* Key Learnings: Data Preparation: We transformed the unemployment data into Prophet's required format (ds, y) and ensured a clean datetime index. Modeling: Prophet automatically detected trends and seasonality, making forecasting intuitive and powerful with minimal tuning.

Forecasting: The model provided unemployment rate predictions for the next 12 months, including confidence intervals to capture uncertainty. Visualization: Prophet's built-in visual tools allowed us to easily interpret the forecast and its components (trend, yearly seasonality).

[ ]:

[ ]:

[ ]: