```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'unemployment-in-india:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F752131%2F1621146%2Fbundle%2Farchive.zip%3FX-Goog-Algorithm%3DGOOG4-RSA-S

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
```

```
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

Analyzing COVID-19 Impact on Unemployment in India

Objective:

The primary aim of this analysis is to assess the repercussions of the COVID-19 pandemic on India's job market. The dataset under consideration contains crucial information about the unemployment rates across various Indian states. The dataset encompasses key indicators such as States, Date, Measuring Frequency, Estimated Unemployment Rate (%), Estimated Employed Individuals, and Estimated Labour Participation Rate (%).

Dataset Details:

The dataset provides insights into the unemployment scenario across different Indian states:

- States: The states within India.
- Date: The date when the unemployment rate was recorded.
- Measuring Frequency: The frequency at which measurements were taken (Monthly).
- Estimated Unemployment Rate (%): The percentage of individuals unemployed in each state of India.
- Estimated Employed Individuals: The count of people currently employed.

- Estimated Labour Participation Rate (%): The proportion of the working population (age group: 16-64 years) participating in the labor force, either employed or actively seeking employment.

This dataset aids in comprehending the unemployment dynamics across India's states during the COVID-19 crisis. It offers valuable insights into how the unemployment rate, employment figures, and labor participation rates have been impacted across different regions in the country. The analysis intends to shed light on the socio-economic consequences of the pandemic on India's workforce and labor market.

Importing necessary libraries

```
import pandas as pd
import numpy as np
import calendar
```

Loading the dataset into pandas dataframe

```
df = pd.read_csv('/kaggle/input/unemployment-in-india/Unemployment_Rate_upto_11_2020.csv')
df.head()
```

	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

Basic information about the dataset

```
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   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
```

Checking for null values

```
df.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
```

Formatting the columns and their datatypes

```
import datetime as dt
# Renaming columns for better clarity
df.columns = ['States', 'Date', 'Frequency', 'Estimated Unemployment Rate', 'Estimated Employed',
              'Estimated Labour Participation Rate', 'Region', 'longitude', 'latitude']

# Converting 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)

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

# Extracting month from 'Date' and creating a 'Month' column
df['Month'] = df['Date'].dt.month

# 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])

# Dropping the original 'Month' column
df.drop(columns='Month', inplace=True)
```

```
df.head()
```

	States	Date	Frequency	Estimated Unemployment Rate	Estimated Employed	Estimated Labour Participation Rate	Region	longitude	latitude	Month_int	Month_name
0	Andhra Pradesh	2020-01-31	M	5.48	16635535	41.02	South	15.9129	79.74	1	Jan
1	Andhra Pradesh	2020-02-29	M	5.83	16545652	40.90	South	15.9129	79.74	2	Feb
2	Andhra Pradesh	2020-03-31	M	5.79	15881197	39.18	South	15.9129	79.74	3	Mar

Exploratory data analysis

Basic statistics

```
df_stat = df[['Estimated Unemployment Rate', 'Estimated Employed', 'Estimated Labour Participation Rate']]
print(round(df_stat.describe().T, 2))
```

```
                                    count         mean          std  \
Estimated Unemployment Rate         267.0        12.24        10.80
Estimated Employed                  267.0  13962105.72  13366318.36
Estimated Labour Participation Rate 267.0        41.68         7.85

                                          min         25%         50%  \
Estimated Unemployment Rate              0.50        4.84        9.65
Estimated Employed                  117542.00  2838930.50  9732417.00
Estimated Labour Participation Rate     16.77       37.26       40.39

                                           75%          max
Estimated Unemployment Rate              16.76        75.85
Estimated Employed                  21878686.00  59433759.00
Estimated Labour Participation Rate        44.06        69.69
```

```
region_stats = df.groupby(['Region'])[['Estimated Unemployment Rate', 'Estimated Employed',
                                       'Estimated Labour Participation Rate']].mean().reset_index()
print(round(region_stats, 2))
```

```
          Region  Estimated Unemployment Rate  Estimated Employed  \
     0       East                        13.92         19602366.90
     1      North                        15.89         13072487.92
     2  Northeast                        10.95          3617105.53
     3      South                        10.45         14040589.33
     4       West                         8.24         18623512.72

        Estimated Labour Participation Rate
```

```
0                               40.11
1                               38.70
2                               52.06
3                               40.44
4                               41.26
```

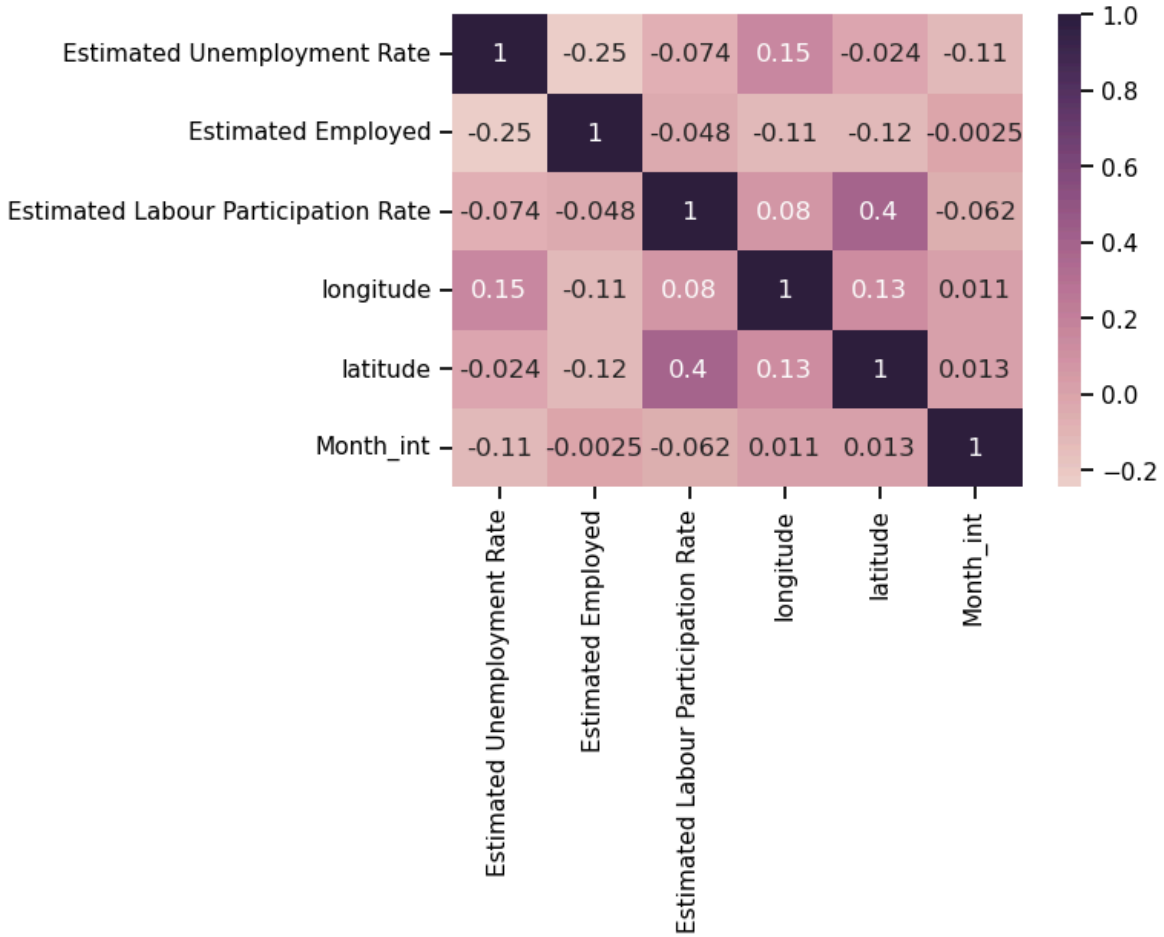
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Heatmap

```
hm = df[['Estimated Unemployment Rate', 'Estimated Employed', 'Estimated Labour Participation Rate', 'longitude', 'latitude', 'Month_int']]
hm = hm.corr()
plt.figure(figsize=(6,4))
sns.set_context('notebook', font_scale=1)
sns.heatmap(data=hm, annot=True, cmap=sns.cubehelix_palette(as_cmap=True))
```

<Axes: >



Boxplot of Unemployment rate per States

```
import plotly.express as px
fig = px.box(df, x='States', y='Estimated Unemployment Rate', color='States', title='Unemployment rate per States', template='seaborn')

# Updating the x-axis category order to be in descending total
fig.update_layout(xaxis={'categoryorder': 'total descending'})
fig.show()
```

Scatter matrix cosidering the employed and unemployed rates

```
fig = px.scatter_matrix(df,template='seaborn',dimensions=['Estimated Unemployment Rate', 'Estimated Employed',
                                                          'Estimated Labour Participation Rate'],color='Region')
fig.show()
```

Bar plot showing the average unemployment rate in each state

```
plot_unemp = df[['Estimated Unemployment Rate','States']]
df_unemployed = plot_unemp.groupby('States').mean().reset_index()

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

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

Haryana and Jharkhand have long been the most unemployed.

Bar chart showing the unemployment rate across regions from Jan. 2020 to Oct. 2020

```
fig = px.bar(df, x='Region', y='Estimated Unemployment Rate', animation_frame='Month_name', color='States',
             title='Unemployment rate across regions from Jan. 2020 to Oct. 2020', height=700, template='seaborn')

# Updating the x-axis category order to be in descending total
fig.update_layout(xaxis={'categoryorder': 'total descending'})

# Adjusting the animation frame duration
fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 1000
fig.show()
```

We see that during the month of April, the states Puducherry, Tamil Nadu, Jharkhand, Bihar, Tripura, Haryana of India saw the major unemplyment hike.

Sunburst chart showing the unemployment rate in each Region and State

```
# Creating a DataFrame with relevant columns
unemployed_df = df[['States', 'Region', 'Estimated Unemployment Rate', 'Estimated Employed', 'Estimated Labour Participation Rate']]

unemployed = unemployed_df.groupby(['Region', 'States'])['Estimated Unemployment Rate'].mean().reset_index()

# Creating a Sunburst chart
fig = px.sunburst(unemployed, path=['Region', 'States'], values='Estimated Unemployment Rate', color_continuous_scale='rdylbu',
                  title='Unemployment rate in each Region and State', height=550, template='presentation')

fig.show()
```

Impact of Lockdown on States Estimated Employed

```
fig = px.scatter_geo(df,'longitude', 'latitude', color="Region",
                     hover_name="States", size="Estimated Unemployment Rate",
                     animation_frame="Month_name",scope='asia',template='seaborn',title='Impack of lockdown on Employement across regions')

fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 3000

fig.update_geos(lataxis_range=[5,35], lonaxis_range=[65, 100],oceancolor="#3399FF",
    showocean=True)

fig.show()
```

The northern regions of India seems to have more unemployed people.

```
# Filtering data for the period before the lockdown (January to April)
bf_lockdown = df[(df['Month_int'] >= 1) & (df['Month_int'] <=4)]

# Filtering data for the lockdown period (April to July)
lockdown = df[(df['Month_int'] >= 4) & (df['Month_int'] <=7)]

# Calculating the mean unemployment rate before lockdown by state
m_bf_lock = bf_lockdown.groupby('States')['Estimated Unemployment Rate'].mean().reset_index()

# Calculating the mean unemployment rate after lockdown by state
m_lock = lockdown.groupby('States')['Estimated Unemployment Rate'].mean().reset_index()

# Combining the mean unemployment rates before and after lockdown by state
m_lock['Unemployment Rate before lockdown'] = m_bf_lock['Estimated Unemployment Rate']

m_lock.columns = ['States','Unemployment Rate before lockdown','Unemployment Rate after lockdown']
m_lock.head()
```

	States	Unemployment Rate before lockdown	Unemployment Rate after lockdown
0	Andhra Pradesh	12.3975	9.4025
1	Assam	6.2450	6.2250
2	Bihar	30.8025	20.7425

```
# percentage change in unemployment rate

m_lock['Percentage change in Unemployment'] = round(m_lock['Unemployment Rate after lockdown'] - m_lock['Unemployment Rate before lockdown']/m_lock['Unemployment Rate before
plot_per = m_lock.sort_values('Percentage change in Unemployment')


# percentage change in unemployment after lockdown

fig = px.bar(plot_per, x='States',y='Percentage change in Unemployment',color='Percentage change in Unemployment',
            title='Percentage change in Unemployment in each state after lockdown',template='ggplot2')
```