→ **Part 0**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

fluffy = pd.read\_csv('/Users/ali/Downloads/unemployment\_analysis (1).csv')

#string\_value = "7.59 lakhs"

#num\_value = float(string\_value.split()[0])

#print(num\_value)

#fluffy.columns

#fluffy.dtypes

#fluffy['1991'][1]

#fluffy.dtypes

#for i in range(len(fluffy['1991'])):

# string\_value=fluffy['1991'][i]

# num\_value =float(string\_value.split()[0])

# fluffy['1991'][i] = num\_value

#fluffy['1991']= float(fluffy['1991'])

#fluffy['1991'] = fluffy['1991'].apply(lambda x: int(x) if x.isdigit() else 0)

#fluffy['1991'] = fluffy['1991'].apply(lambda x: float(x) if isinstance(x, str) and x.isdigit() else 0)

#fluffy['1991'] = fluffy['1991'].astype(float)

#fluffy\_groupby = fluffy[['Country Name', 'Country Code']].groupby(['Country Name', 'Country Code']).sum()

#for i in range(len(fluffy['1991'])):

# string\_value=fluffy['1991'][i]

# num\_value =float(string\_value.split()[0])

# fluffy['1991'][i] = num\_value

#fluffy['1991'] = fluffy['1991'].astype(float)

for i in range(len(fluffy['1991'])):

string\_value=fluffy['1991'][i]

num\_value =float(string\_value.split()[0])

fluffy['1991'][i] = num\_value

fluffy['1991'] = fluffy['1991'].astype(float)

for i in range(len(fluffy['1992'])):

string\_value=fluffy['1992'][i]

num\_value =float(string\_value.split()[0])

fluffy['1992'][i] = num\_value

fluffy['1992'] = fluffy['1992'].astype(float)

for i in range(len(fluffy['1993'])):

string\_value=fluffy['1993'][i]

num\_value =float(string\_value.split()[0])

fluffy['1993'][i] = num\_value

fluffy['1993'] = fluffy['1993'].astype(float)

for i in range(len(fluffy['1994'])):

string\_value=fluffy['1994'][i]

num\_value =float(string\_value.split()[0])

fluffy['1994'][i] = num\_value

fluffy['1994'] = fluffy['1994'].astype(float)

for i in range(len(fluffy['1995'])):

string\_value=fluffy['1995'][i]

num\_value =float(string\_value.split()[0])

fluffy['1995'][i] = num\_value

fluffy['1995'] = fluffy['1995'].astype(float)

for i in range(len(fluffy['1996'])):

string\_value=fluffy['1996'][i]

num\_value =float(string\_value.split()[0])

fluffy['1996'][i] = num\_value

fluffy['1996'] = fluffy['1996'].astype(float)

for i in range(len(fluffy['1997'])):

string\_value=fluffy['1997'][i]

num\_value =float(string\_value.split()[0])

fluffy['1997'][i] = num\_value

fluffy['1997'] = fluffy['1997'].astype(float)

for i in range(len(fluffy['1998'])):

string\_value=fluffy['1998'][i]

num\_value =float(string\_value.split()[0])

fluffy['1998'][i] = num\_value

fluffy['1998'] = fluffy['1998'].astype(float)

for i in range(len(fluffy['1999'])):

string\_value=fluffy['1999'][i]

num\_value =float(string\_value.split()[0])

fluffy['1999'][i] = num\_value

fluffy['1999'] = fluffy['1999'].astype(float)

#for i in range(len(fluffy['2000'])):

# string\_value=fluffy['2000'][i]

# num\_value =float(string\_value.split()[0])

# fluffy['2000'][i] = num\_value

#fluffy['2000'] = pd.to\_numeric(fluffy['2000'], errors='coerce')

#fluffy['2000'] = fluffy['2000'].astype(float)

#print(fluffy['2000'])

for i in range(len(fluffy['2000'])):

value = fluffy['2000'][i]

if isinstance(value, str):

num\_value = float(value.split()[0])

fluffy['2000'][i] = num\_value

#fluffy.dtypes

fluffy['2000'] = fluffy['2000'].astype(float)

for i in range(len(fluffy['2001'])):

string\_value=fluffy['2001'][i]

num\_value =float(string\_value.split()[0])

fluffy['2001'][i] = num\_value

fluffy['2001'] = fluffy['2001'].astype(float)

for i in range(len(fluffy['2002'])):

string\_value=fluffy['2002'][i]

num\_value =float(string\_value.split()[0])

fluffy['2002'][i] = num\_value

fluffy['2002'] = fluffy['2002'].astype(float)

for i in range(len(fluffy['2003'])):

string\_value=fluffy['2003'][i]

num\_value =float(string\_value.split()[0])

fluffy['2003'][i] = num\_value

fluffy['2003'] = fluffy['2003'].astype(float)

for i in range(len(fluffy['2004'])):

string\_value=fluffy['2004'][i]

num\_value =float(string\_value.split()[0])

fluffy['2004'][i] = num\_value

fluffy['2004'] = fluffy['2004'].astype(float)

for i in range(len(fluffy['2005'])):

string\_value=fluffy['2005'][i]

num\_value =float(string\_value.split()[0])

fluffy['2005'][i] = num\_value

fluffy['2005'] = fluffy['2005'].astype(float)

for i in range(len(fluffy['2006'])):

string\_value=fluffy['2006'][i]

num\_value =float(string\_value.split()[0])

fluffy['2006'][i] = num\_value

fluffy['2006'] = fluffy['2006'].astype(float)

for i in range(len(fluffy['2007'])):

string\_value=fluffy['2007'][i]

num\_value =float(string\_value.split()[0])

fluffy['2007'][i] = num\_value

fluffy['2007'] = fluffy['2007'].astype(float)

for i in range(len(fluffy['2008'])):

string\_value=fluffy['2008'][i]

num\_value =float(string\_value.split()[0])

fluffy['2008'][i] = num\_value

fluffy['2008'] = fluffy['2008'].astype(float)

for i in range(len(fluffy['2009'])):

string\_value=fluffy['2009'][i]

num\_value =float(string\_value.split()[0])

fluffy['2009'][i] = num\_value

fluffy['2009'] = fluffy['2009'].astype(float)

for i in range(len(fluffy['2010'])):

string\_value=fluffy['2010'][i]

num\_value =float(string\_value.split()[0])

fluffy['2010'][i] = num\_value

fluffy['2010'] = fluffy['2010'].astype(float)

for i in range(len(fluffy['2011'])):

string\_value=fluffy['2011'][i]

num\_value =float(string\_value.split()[0])

fluffy['2011'][i] = num\_value

fluffy['2011'] = fluffy['2011'].astype(float)

for i in range(len(fluffy['2012'])):

string\_value=fluffy['2012'][i]

num\_value =float(string\_value.split()[0])

fluffy['2012'][i] = num\_value

fluffy['2012'] = fluffy['2012'].astype(float)

for i in range(len(fluffy['2013'])):

string\_value=fluffy['2013'][i]

num\_value =float(string\_value.split()[0])

fluffy['2013'][i] = num\_value

fluffy['2013'] = fluffy['2013'].astype(float)

for i in range(len(fluffy['2014'])):

string\_value=fluffy['2014'][i]

num\_value =float(string\_value.split()[0])

fluffy['2014'][i] = num\_value

fluffy['2014'] = fluffy['2014'].astype(float)

for i in range(len(fluffy['2015'])):

value = fluffy['2015'][i]

if isinstance(value, str):

num\_value = float(value.split()[0])

fluffy['2015'][i] = num\_value

fluffy['2015'] = fluffy['2015'].astype(float)

for i in range(len(fluffy['2016'])):

string\_value=fluffy['2016'][i]

num\_value =float(string\_value.split()[0])

fluffy['2016'][i] = num\_value

fluffy['2016'] = fluffy['2016'].astype(float)

for i in range(len(fluffy['2017'])):

string\_value=fluffy['2017'][i]

num\_value =float(string\_value.split()[0])

fluffy['2017'][i] = num\_value

fluffy['2017'] = fluffy['2017'].astype(float)

for i in range(len(fluffy['2018'])):

string\_value=fluffy['2018'][i]

num\_value =float(string\_value.split()[0])

fluffy['2018'][i] = num\_value

fluffy['2018'] = fluffy['2018'].astype(float)

for i in range(len(fluffy['2019'])):

string\_value=fluffy['2019'][i]

num\_value =float(string\_value.split()[0])

fluffy['2019'][i] = num\_value

fluffy['2019'] = fluffy['2019'].astype(float)

for i in range(len(fluffy['2020'])):

string\_value=fluffy['2020'][i]

num\_value =float(string\_value.split()[0])

fluffy['2020'][i] = num\_value

fluffy['2020'] = fluffy['2020'].astype(float)

for i in range(len(fluffy['2021'])):

string\_value=fluffy['2021'][i]

num\_value =float(string\_value.split()[0])

fluffy['2021'][i] = num\_value

fluffy['2021'] = fluffy['2021'].astype(float)

—-->Part 1

def data\_cleaning(df):

# Display missing values in the DataFrame

if df.isnull().any().any():

print("Missing values before cleaning:")

print(df[df.isnull().any(axis=1)])

# Replace missing values with the average of the corresponding country

#df\_cleaned = df.groupby('Country Name').transform(lambda x: x.fillna(x.mean()))

df\_cleaned = df.apply(lambda x: x.fillna(x.mean()) if pd.api.types.is\_numeric\_dtype(x) else x, axis=0)

# Display missing values after cleaning

print("\nMissing values after cleaning:")

print(df\_cleaned[df\_cleaned.isnull().any(axis=1)])

# Find and replace outliers for specific countries (Benin, Bahrain)

outlier\_countries = ['Benin', 'Bahrain']

for country\_name in outlier\_countries:

country\_data = df\_cleaned[df\_cleaned['Country Name'] == country\_name]

for column in df.columns[2:]:

q1 = country\_data[column].quantile(0.25)

q3 = country\_data[column].quantile(0.75)

iqr = q3 - q1

lower\_bound = q1 - 1.5 \* iqr

upper\_bound = q3 + 1.5 \* iqr

# Replace outliers with mean/median/mode

df\_cleaned.loc[df\_cleaned['Country Name'] == country\_name, column] = df\_cleaned.loc[df\_cleaned['Country Name'] == country\_name, column].apply(lambda x: country\_data[column].mean() if lower\_bound <= x <= upper\_bound else x)

# Create a new column 'Year' and 'No. of unemployed'

#df\_cleaned['Year'] = df\_cleaned.columns[2:].astype(int).to\_list() \* (len(df\_cleaned) // len(df\_cleaned.columns[2:]))

#df\_cleaned['Year'] = sorted(df\_cleaned.columns[2:].astype(int).to\_list() \* (len(df\_cleaned) // len(df\_cleaned.columns[2:])))

#df\_cleaned['Year'] = df\_cleaned.columns[2:].astype(int).to\_list() \* (len(df\_cleaned) // len(df\_cleaned.columns[2:]))

#df\_cleaned['Year'] = df\_cleaned['Year'].iloc[:len(df\_cleaned)]

#df\_cleaned['No. of unemployed'] = df\_cleaned.iloc[:, 2:].sum(axis=1)

# Change column names

df\_cleaned = df\_cleaned.rename(columns={'Country Name': 'Country\_name', 'Country Code': 'Country\_code'})

return df\_cleaned

else:

print("No missing values in the DataFrame.")

return df

# Example usage:

# Assuming 'fluffy' is your DataFrame

# Call the function with your DataFrame

df\_pivot = data\_cleaning(fluffy)

—-> part 2

def descriptive\_stats(df, country\_code):

# Filter data for the specified country\_code

country\_data = df[df['Country\_code'] == country\_code]

# Calculate mean, median, mode, and standard deviation for each year

stats = country\_data.describe().transpose()[['mean', '50%', 'std']].reset\_index()

stats = stats.rename(columns={'50%': 'median'})

# Find the year with minimum and maximum unemployment

min\_year = country\_data.iloc[:, 2:].idxmin(axis=1).str.extract('(\d+)').astype(int).iloc[0]

max\_year = country\_data.iloc[:, 2:].idxmax(axis=1).str.extract('(\d+)').astype(int).iloc[0]

# Find the top 5 countries with maximum unemployment in 2021

top5\_2021 = df.nlargest(5, '2021')[['Country\_name', '2021']]

# Find the top 3 countries with unemployment greater than 5 lacs in 2021

top3\_2021\_gt\_5lacs = df[df['2021'] > 5].nlargest(3, '2021')[['Country\_name', '2021']]

# Calculate the percentage change in unemployment from 1991 to 2021

change\_percentage = ((df['2021'] - df['1991']) / df['1991']) \* 100

return stats, min\_year, max\_year, top5\_2021, top3\_2021\_gt\_5lacs, change\_percentage

# Example usage:

# Assuming 'df\_pivot' is your cleaned DataFrame

# Call the function with the DataFrame and the desired country\_code

country\_code = 'WSM'

descriptive\_stats\_result = descriptive\_stats(df\_pivot, country\_code)

print(descriptive\_stats\_result)

**—-> Part 3**

import pandas as pd

import matplotlib.pyplot as plt

# Assuming df\_pivot is the cleaned DataFrame from the previous questions

# 1. Minimum Unemployment for Country 'BGR'

min\_unemployment\_BGR = df\_pivot[df\_pivot['Country Code'] == 'BGR'].iloc[:, 2:].min().idxmin()

print(f"Year of minimum unemployment for 'BGR': {min\_unemployment\_BGR}")

# 2. Create a New DataFrame with Country Names and Country Codes

df\_percentage\_change = df\_pivot[['Country Name', 'Country Code']].copy()

df\_percentage\_change['Percentage Change'] = ((df\_pivot['2021'] - df\_pivot['1991']) / df\_pivot['1991']) \* 100

# 3. Compare Minimum Unemployment for 'Japan' with Previous and Next Years

cj = df\_pivot[df\_pivot['Country Code'] == 'JPN']

minyear\_japan = int(cj.iloc[:, 2:].idxmin(axis=1).str.extract('(\d+)').iloc[0])

previous\_year = minyear\_japan - 1

next\_year = minyear\_japan + 1

comparison\_years = cj[['Country Name', str(previous\_year), str(minyear\_japan), str(next\_year)]]

print(comparison\_years)

# 4. Countries with Huge Jump in Unemployment from 2019 to 2021

countries\_of\_interest = ['MDA', 'NAC', 'PAN', 'PAK', 'UGA']

df\_interest = df\_pivot[df\_pivot['Country Code'].isin(countries\_of\_interest)]

change\_2019\_2021 = ((df\_interest['2021'] - df\_interest['2019']) / df\_interest['2019']) \* 100

significant\_changes = df\_interest[(change\_2019\_2021 > 20) | (change\_2019\_2021 < -20)]

print("Countries with significant changes from 2019 to 2021:")

print(significant\_changes)

# 5. Change in Percentage for Country 'LSO' from 1991 to 2021

country\_LSO = df\_pivot[df\_pivot['Country Code'] == 'LSO']

change\_percentage\_LSO = ((country\_LSO['2021'] - country\_LSO['1991']) / country\_LSO['1991']) \* 100

# Plotting the unemployment numbers for 'LSO' over the years

years = country\_LSO.columns[2:].astype(int)

plt.plot(years, country\_LSO.iloc[:, 2:].values.flatten(), marker='o')

plt.xlabel('Year')

plt.ylabel('Unemployment Rate')

plt.title('Unemployment Rate Over the Years for LSO')

plt.show()

print("Change in percentage for 'LSO' from 1991 to 2021:", change\_percentage\_LSO)

**—---> Part 4**

#----------------------------------------------Moving Average Assignment—---------------------

import matplotlib.pyplot as plt

ck = df\_pivot[df\_pivot['Country\_code']=='AFW'].iloc[:,2:]

ck\_tran = ck.T

ck\_tran.columns = ['Year']

extended\_index = pd.RangeIndex(start=0, stop=len(ck\_tran) + 2)

padded\_values = np.pad(ck\_tran['Year'].values, (0, len(extended\_index) - len(ck\_tran['Year'].values)), mode='constant', constant\_values=np.nan)

ck\_traun\_extended = pd.DataFrame(index=extended\_index, data=padded\_values, columns=['Year'])

plt.figure(figsize=(10,6))

plt.plot(ck\_tran.index, ck\_tran['Year'], label='chalse')

ma\_win = 3

ma\_val= ck\_tran['Year'].rolling(window=ma\_win).mean()

plt.plot(ck\_tran.index, ma\_val, label=f'{ma\_win}-Point Moving Average', linestyle='dashed')

plt.title('Moving Average Forecasting')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

plt.show()

ma\_val\_ext = ck\_traun\_extended['Year'].rolling(window=ma\_win).mean()

plt.figure(figsize=(10,6))

plt.plot(ck\_traun\_extended.index, ma\_val\_ext, label=f'{ma\_win}-Point Moving Average (Extended)', linestyle='dashed', color='brown')

plt.title('Extended Moving Average Forecasting')

plt.xlabel('Date')

plt.ylabel('Value')

plt.legend()

plt.show()

#--------------------ARIMA Model Assignment--------------

from statsmodels.tsa.arima.model import ARIMA

import matplotlib.pyplot as plt

import matplotlib.dates as mdates # Add this import statement

# Assuming ck\_tran is a DataFrame with a 'Year' column

# Convert the 'Year' column to a datetime index

ck\_tran.index = pd.to\_datetime(ck\_tran.index, format='%Y')

# Fit ARIMA model

model\_ck\_tran = ARIMA(ck\_tran['Year'], order=(1, 1, 1))

result\_ck\_tran = model\_ck\_tran.fit()

# Forecast future values

forecast\_steps\_ck\_tran = 3 # Adjust the number of forecast steps as needed

forecast\_ck\_tran = result\_ck\_tran.get\_forecast(steps=forecast\_steps\_ck\_tran)

# Plot results for ck\_tran

plt.plot(ck\_tran.index, ck\_tran['Year'], label='Original Data')

plt.plot(result\_ck\_tran.fittedvalues.index, result\_ck\_tran.fittedvalues, color='red', label='Fitted Values')

plt.plot(forecast\_ck\_tran.predicted\_mean.index, forecast\_ck\_tran.predicted\_mean, color='green', label='Forecast')

plt.fill\_between(forecast\_ck\_tran.conf\_int().index, forecast\_ck\_tran.conf\_int().iloc[:, 0], forecast\_ck\_tran.conf\_int().iloc[:, 1], color='green', alpha=0.2, label='Confidence Intervals')

# Specify the date format for better visualization

#plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter('%Y'))

# Rotate x-axis labels for better readability (optional)

plt.gcf().autofmt\_xdate()

plt.title('ARIMA Model on ck\_tran Data')

plt.xlabel('Year')

plt.ylabel('Value')

plt.legend()

plt.show()

##----------------------Assignment ends here---------------

##---------------comparing mape of two models--------------------------------------

forecast\_steps\_ck\_tran = 10

forecast\_ck\_tran = result\_ck\_tran.get\_forecast(steps=forecast\_steps\_ck\_tran)

# Concatenate fitted values and forecasted values

predicted\_values\_arima = np.concatenate([result\_ck\_tran.fittedvalues.values, forecast\_ck\_tran.predicted\_mean.values])

# Trim or pad the actual values to match the length of predicted\_values\_arima

actual\_values\_arima = ck\_tran['Year'].values[-len(predicted\_values\_arima):]

# Trim or pad the predicted values to match the length of actual\_values\_arima

predicted\_values\_arima = predicted\_values\_arima[:len(actual\_values\_arima)]

# Calculate MAPE for ARIMA model

mape\_arima = np.mean(np.abs((actual\_values\_arima - predicted\_values\_arima) / actual\_values\_arima)) \* 100

# Calculate MAPE for Moving Average model

mape\_ma = np.mean(np.abs((ck\_tran['Year'].values - predicted\_values\_arima) / ck\_tran['Year'].values)) \* 100

# Print MAPE values for comparison

print(f'MAPE for ARIMA model: {mape\_arima:.2f}%')

print(f'MAPE for Moving Average model: {mape\_ma:.2f}%')

# Compare MAPE values

if mape\_arima < mape\_ma:

print('ARIMA model has a lower MAPE, indicating better accuracy.')

elif mape\_arima > mape\_ma:

print('Moving Average model has a lower MAPE, indicating better accuracy.')

else:

print('Both models have similar MAPE values.')

—-->Part 5 Visualization

#---------- visualization assignment starts here----------------

#---------Question 1------------------

#data0= pd.DataFrame({'Country':['India', 'Aus','Pak','SA','Zim', 'BAN'],'Year':[22,23,54,85,45,72]})

data1 = df\_pivot[['Country\_name', '2021']]

data1\_sorted = data1.sort\_values(by='2021', ascending=False)

data1\_values = data1\_sorted['2021'].head(10)

data1\_con = data1\_sorted['Country\_name'].head(10)

plt.bar(data1\_con, data1\_values, align='edge', color=['green','brown'])

plt.xlabel('Countries')

plt.ylabel('Unemployment')

plt.gcf().autofmt\_xdate()

#plt.gcf().autofmt\_ydate()

plt.title('Top 10 countries in 2021')

#-----Question 2----

data2= df\_pivot

data2.columns

colums\_to\_be\_dropped= ['Country\_code', '1991', '1992', '1993', '1994', '1995',

'1996', '1997', '1998','2020', ]

data3 = data2.drop(columns=colums\_to\_be\_dropped, axis=1)

data3.columns

data3\_sorted = data3.sort\_values(by='2021', ascending=False)

data3\_values = data3\_sorted[['1999', '2000', '2001', '2002', '2003', '2004', '2005',

'2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',

'2015', '2016', '2017', '2018', '2019']].head(5)

data3\_con = data3\_sorted['Country\_name'].head(5)

plt.plot(data3\_values, data3\_con, marker='o', linestyle='-', color='b', label='Line Maro')

plt.xlabel('Values')

plt.ylabel('countries')

plt.show()

data3\_values

data3\_con

import matplotlib.pyplot as plt

# Transpose the data3\_values DataFrame

data3\_values\_transposed = data3\_values.T

# Plotting the line graph

for country in data3\_values\_transposed.columns:

plt.plot(data3\_values\_transposed.index, data3\_values\_transposed[country], marker='o', linestyle='-', label=country)

plt.gca().xaxis.set\_major\_formatter(mdates.DateFormatter('%Y'))

plt.gcf().autofmt\_xdate

# Adding labels and title

plt.xlabel('Year')

plt.ylabel('Unemployement')

plt.title('Line Graph Example')

# Adding a legend

#plt.legend()

# Display the plot

plt.show()