import plotly.express as px

import plotly.graph\_objects as go

def create\_figure\_1(df):

"""

Create choropleth chart (world map visualization)

Args:

df: input dataframe

Returns:

fig: Plotly figure

"""

##Chart 1: world population

fig1 = px.choropleth(df, locations='Country', locationmode='country names', color='2022 Population',

color\_continuous\_scale="Blues",

range\_color=(0, 500000000),

labels={'2022 Population': '2022 Population'}

)

fig1.update\_layout(margin={"r":0,"t":0,"l":0,"b":0}, dragmode=False)

return fig1

#TODO: Create new function: create\_figure\_x(df) with the code you wrote in the test\_plotly\_charts.ipynb

def create\_figure\_2(df):

"""

Create Line chart for each country population evolution by year (with dropdown menu)

Args:

df: input dataframe

Returns:

fig: Plotly figure

"""

##Chart 2: Get population from 1970 to 2022

df\_population = df[df.columns[5:13]]

#Invert order of columns

df\_population = df\_population[df\_population.columns[::-1]]

df\_population.insert(0, 'Country', df['Country'])

fig2 = go.Figure()

# set up ONE trace

fig2.add\_trace(go.Scatter(x=df\_population.columns[1:],

y=df\_population[df\_population.Country=='Afghanistan'][df\_population.columns[1:]].iloc[0],

fill='tozeroy',

visible=True)

)

updatemenu = []

buttons = []

# button with one option for each dataframe

for country in df\_population.Country:

buttons.append(dict(method='restyle',

label=country,

visible=True,

args=[{'y': [df\_population[df\_population.Country==country][df\_population.columns[1:]].iloc[0]],

'x': [df\_population.columns[1:]],

'type':'line',

'fill' : 'tozeroy'}, [0]],

)

)

# some adjustments to the updatemenus

updatemenu = []

your\_menu = dict()

updatemenu.append(your\_menu)

updatemenu[0]['buttons'] = buttons

updatemenu[0]['direction'] = 'down'

updatemenu[0]['showactive'] = True

# add dropdown menus to the figure

fig2.update\_layout(showlegend=False, updatemenus=updatemenu)

return fig2

#TODO: Create new function: create\_figure\_x(df) with the code you wrote in the test\_plotly\_charts.ipynb

def create\_figure\_3(df):

"""

Create Pie chart with World Population Percentage

Args:

df: input dataframe

Returns:

fig: Plotly figure

"""

#Chart 3: Pie chart with share of world population

# top\_10 = df.sort\_values(by=['World Population Percentage'], ascending=False).iloc[0:10]

fig3 = px.pie(df, values='World Population Percentage', names='Country')

fig3.update\_traces(textposition='inside',

text = df['Country'])

fig3.update\_layout(uniformtext\_minsize=8, uniformtext\_mode='hide')

fig3.update\_layout(margin=dict(t=0, b=0, l=0, r=0))

return fig3

def compute\_figures(df):

fig1 = create\_figure\_1(df)

fig2 = create\_figure\_2(df)

fig3 = create\_figure\_3(df)

return [fig1, fig2, fig3]

from datetime import date

from dateutil.relativedelta import relativedelta

from dash import Dash, html, dcc, dash\_table

from dash.dependencies import Input, Output

from load\_data import load\_local\_data

from create\_charts import compute\_figures

app = Dash(\_\_name\_\_)

local\_path='world\_population.csv'

##By defining the serve\_layout function, we are ensuring that the data is refreshed everytime the page is reloaded

def serve\_layout():

#Read data from local path

df = load\_local\_data(local\_path)

#Here we are loading the plotly charts we created previously

figures = compute\_figures(df)

#Here we are defining the dash components

#The parent component is html.Div

#For each component, we can define a list (children=[]) of components

dash\_layout = html.Div(

children=[

#Navigation bar already defined

html.Nav(className='navbar', children=[

html.Div(children=[html.Img(className='navbar\_\_gcp\_logo', src='./assets/gcp\_logo.png')]),

html.H1(children="ING Code Breakfast - Dash App in GCP"),

html.Div(children=[html.Img(className='navbar\_\_dash\_logo', src='./assets/dash\_logo.png')]),

]),

html.H1('World population map'),

dcc.Graph(id='0', figure=figures[0]),

## TODO: Add your components here

html.H1('Evolution of population for each country'),

dcc.Graph(id='1', figure=figures[1]),

html.H1('World Population Percentage'),

dcc.Graph(id='2', figure=figures[2])

]

)

return dash\_layout

## Dash App

app.layout = serve\_layout

if \_\_name\_\_ == "\_\_main\_\_":

app.run\_server(host="0.0.0.0", port=8080, debug=True)

#Note: Run 'python dash\_app.py' to run the application

import pandas as pd

from google.cloud import bigquery

from google.cloud import storage

def load\_data\_from\_bigquery(project, dataset, table):

'''

Read the first half of the dataset from BigQuery

'''

#TODO: Write the code to load the data from BigQuery

# Construct a BigQuery client object.

bqclient = bigquery.Client()

#project =

#datest = code\_breakfast

#table = world\_population\_1

query = """

SELECT \* from `{}.{}.{}`

""".format(project, dataset, table)

df1 = (

bqclient.query(query)

.result()

.to\_dataframe()

)

return df1

def load\_data\_from\_cloudstorage(bucket, blob):

'''

Read the second half of the dataset from GCS

'''

#TODO: Write the code to load the data from GCS

#bucket = code\_breakfast

#blob = world\_population\_2.csv

gcs\_path = 'gs://{}/{}'.format(bucket, blob)

df2 = pd.read\_csv(gcs\_path, sep=';')

return df2

def combine\_dataframes(df1,df2):

'''

Combine the two dataframe from BigQuery and GCS

'''

return df1.append(df2)

def load\_local\_data(local\_path):

'''

Read .csv file from local\_path

'''

df = pd.read\_csv(local\_path)

return df

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### You can use this notebook to play around with plotly and create new charts"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"%load\_ext lab\_black\n",

"%load\_ext autoreload\n",

"%autoreload 2\n",

"%matplotlib inline"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"from load\_data import \*\n",

"import plotly.express as px\n",

"import plotly.graph\_objects as go\n",

"\n",

"local\_path = \"world\_population.csv\"\n",

"df = load\_local\_data(local\_path)\n",

"\n",

"df"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"##Prepare dataframe\n",

"df\_population = df[df.columns[5:13]]\n",

"\n",

"# Invert order of columns\n",

"df\_population = df\_population[df\_population.columns[::-1]]\n",

"df\_population.insert(0, \"Country\", df[\"Country\"])\n",

"\n",

"df\_netherlands = df\_population[df[\"Country\"] == \"Netherlands\"].drop(columns=[\"Country\"])\n",

"\n",

"final\_df = pd.DataFrame({\"Year\": df\_netherlands.columns, \"Population\": df\_netherlands.values[0]})\n",

"\n",

"final\_df"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Create Plotly figure\n",

"- Instructions:\n",

"\n",

" 1. The first part of the code is already provided (dataframe preprocessing)\n",

" 2. Fill in the code after the 'TODO:' comment\n",

" 3. Create Plotly figure to visualize Netherlands Population evolution from 1970 to 2022\n",

" 4. After the function is completed (and you are able to see it in the notebook), copy the function and paste it the module 'create\_charts.py'\n",

"\n",

"Note: In case of help, visit Plotly docs: https://plotly.com/python/line-charts/"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def create\_figure(df):\n",

"\n",

" ##Prepare dataframe\n",

" df\_population = df[df.columns[5:13]]\n",

"\n",

" # Invert order of columns\n",

" df\_population = df\_population[df\_population.columns[::-1]]\n",

" df\_population.insert(0, \"Country\", df[\"Country\"])\n",

"\n",

" df\_netherlands = df\_population[df[\"Country\"] == \"Netherlands\"].drop(columns=[\"Country\"])\n",

"\n",

" final\_df = pd.DataFrame({\"Year\": df\_netherlands.columns, \"Population\": df\_netherlands.values[0]})\n",

"\n",

" # TODO: Write plotly code to visualize Netherlands population evolution from 1970 to 2022 (Line or Bar chart)\n",

" fig = px.bar(final\_df, x=\"Year\", y=\"Population\", title=\"Netherlands Population Evolution\")\n",

"\n",

" return fig"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Visualize your plotly chart"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"fig = create\_figure(df)\n",

"\n",

"fig.show()"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3.9.0 ('myenv': venv)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.0"

},

"orig\_nbformat": 4,

"vscode": {

"interpreter": {

"hash": "53fc6b8162b76aa4fd82562cc7ae5a4fa28111de6e3e8492d7ac43cc72bf1ff5"

}

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

import plotly.express as px

import plotly.graph\_objects as go

def create\_figure\_1(df):

"""

Create choropleth chart (world map visualization)

Args:

df: input dataframe

Returns:

fig: Plotly figure

"""

##Chart 1: world population

fig1 = px.choropleth(df, locations='Country', locationmode='country names', color='2022 Population',

color\_continuous\_scale="Blues",

range\_color=(0, 500000000),

labels={'2022 Population': '2022 Population'}

)

fig1.update\_layout(margin={"r":0,"t":0,"l":0,"b":0}, dragmode=False)

return fig1

#TODO: Create new function: create\_figure\_2(df) with the code you wrote in the test\_plotly\_charts.ipynb

#Args: df

#Return: fig

def compute\_figures(df):

fig1 = create\_figure\_1(df)

#TODO: Add the new figures in this code, and in the return List

# fig2 = create\_figure\_2(df)

return [fig1]

from datetime import date

from dateutil.relativedelta import relativedelta

from dash import Dash, html, dcc, dash\_table

from dash.dependencies import Input, Output

from load\_data import load\_local\_data

from create\_charts import compute\_figures

app = Dash(\_\_name\_\_)

local\_path='world\_population.csv'

##By defining the serve\_layout function, we are ensuring that the data is refreshed everytime the page is reloaded

def serve\_layout():

#Read data from local path

df = load\_local\_data(local\_path)

#Here we are loading the plotly charts we created previously

figures = compute\_figures(df)

#Here we are defining the dash components

#The parent component is html.Div

#For each component, we can define a list (children=[]) of components

dash\_layout = html.Div(

children=[

#Navigation bar already defined

html.Nav(className='navbar', children=[

html.Div(children=[html.Img(className='navbar\_\_gcp\_logo', src='./assets/gcp\_logo.png')]),

html.H1(children="ING Code Breakfast - Dash App in GCP"),

html.Div(children=[html.Img(className='navbar\_\_dash\_logo', src='./assets/dash\_logo.png')]),

]),

html.H1('World population map'),

dcc.Graph(id='0', figure=figures[0]),

## TODO: Add your components here

]

)

return dash\_layout

## Dash App

app.layout = serve\_layout

if \_\_name\_\_ == "\_\_main\_\_":

app.run\_server(host="0.0.0.0", port=8080, debug=True)

#Note: Run 'python dash\_app.py' to run the application

import pandas as pd

from google.cloud import bigquery

from google.cloud import storage

def load\_data\_from\_bigquery(project, dataset, table):

'''

Read the first half of the dataset from BigQuery

'''

#TODO: Write the code to load the data from BigQuery

return df1

def load\_data\_from\_cloudstorage(bucket, blob):

'''

Read the second half of the dataset from GCS

'''

#TODO: Write the code to load the data from GCS

return df2

def combine\_dataframes(df1,df2):

'''

Combine the two dataframe from BigQuery and GCS

'''

return df1.append(df2)

def load\_local\_data(local\_path):

'''

Read .csv file from local\_path

'''

df = pd.read\_csv(local\_path)

return df

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### You can use this notebook to play around with plotly and create new charts"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"%load\_ext lab\_black\n",

"%load\_ext autoreload\n",

"%autoreload 2\n",

"%matplotlib inline"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"from load\_data import \*\n",

"import plotly.express as px\n",

"import plotly.graph\_objects as go\n",

"\n",

"local\_path = \"world\_population.csv\"\n",

"df = load\_local\_data(local\_path)\n",

"\n",

"df"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"##Prepare dataframe\n",

"df\_population = df[df.columns[5:13]]\n",

"\n",

"# Invert order of columns\n",

"df\_population = df\_population[df\_population.columns[::-1]]\n",

"df\_population.insert(0, \"Country\", df[\"Country\"])\n",

"\n",

"df\_netherlands = df\_population[df[\"Country\"] == \"Netherlands\"].drop(columns=[\"Country\"])\n",

"\n",

"final\_df = pd.DataFrame({\"Year\": df\_netherlands.columns, \"Population\": df\_netherlands.values[0]})\n",

"\n",

"final\_df"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Create Plotly figure\n",

"- Instructions:\n",

"\n",

" 1. The first part of the code is already provided (dataframe preprocessing)\n",

" 2. Fill in the code after the 'TODO:' comment\n",

" 3. Create Plotly figure to visualize Netherlands Population evolution from 1970 to 2022\n",

" 4. After the function is completed (and you are able to see it in the notebook), copy the function and paste it the module 'create\_charts.py'\n",

"\n",

"Note: In case of help, visit Plotly docs: https://plotly.com/python/line-charts/"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"def create\_figure(df):\n",

"\n",

" ##Prepare dataframe\n",

" df\_population = df[df.columns[5:13]]\n",

"\n",

" # Invert order of columns\n",

" df\_population = df\_population[df\_population.columns[::-1]]\n",

" df\_population.insert(0, \"Country\", df[\"Country\"])\n",

"\n",

" df\_netherlands = df\_population[df[\"Country\"] == \"Netherlands\"].drop(columns=[\"Country\"])\n",

"\n",

" final\_df = pd.DataFrame({\"Year\": df\_netherlands.columns, \"Population\": df\_netherlands.values[0]})\n",

"\n",

" # TODO: Write plotly code to visualize Netherlands population evolution from 1970 to 2022 (Line or Bar chart)\n",

"\n",

"\n",

" return fig"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Visualize your plotly chart"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"fig = create\_figure(df)\n",

"\n",

"fig.show()"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3.9.0 ('myenv': venv)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.0"

},

"orig\_nbformat": 4,

"vscode": {

"interpreter": {

"hash": "53fc6b8162b76aa4fd82562cc7ae5a4fa28111de6e3e8492d7ac43cc72bf1ff5"

}

}

},

"nbformat": 4,

"nbformat\_minor": 2

}