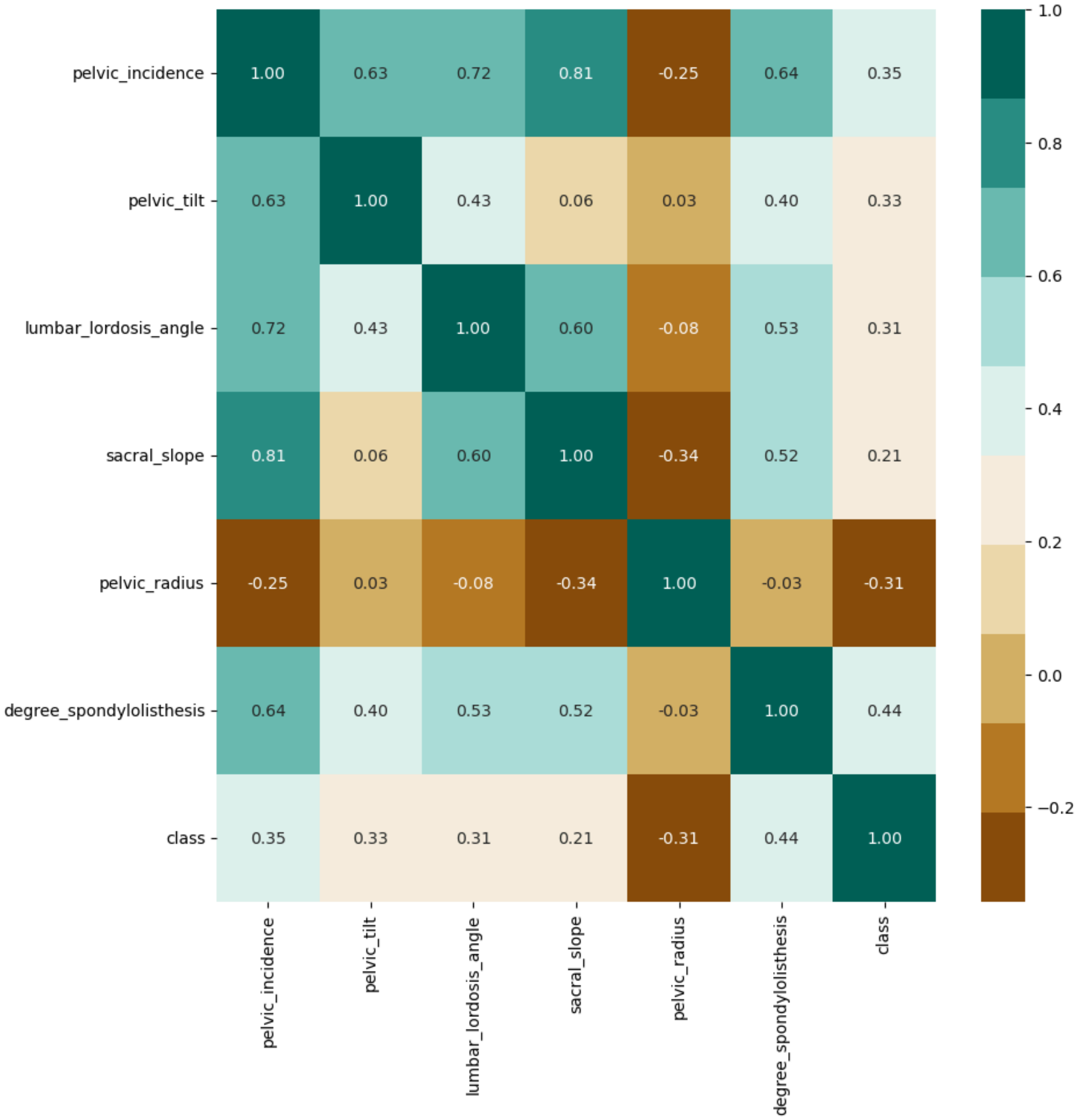


Challenge task: Find other data from the UCI Machine Learning Repository. Using the previous code for reference, go explore!

In [21]:

```
import requests
import zipfile
import io
import pandas as pd

# --- Step 1: Download and Extract the New Dataset ---
# URL for the Bank Marketing dataset zip file
f_zip = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-ad

print("Downloading new dataset (Bank Marketing)...")
# Send a GET request to the URL
r = requests.get(f_zip, stream=True)

print("Extracting files...")
# Create a ZipFile object from the in-memory content
bank_zip = zipfile.ZipFile(io.BytesIO(r.content))

# Extract all contents
bank_zip.extractall()
print(f"Files extracted: {bank_zip.namelist()}")
```

```python

# --- Step 2: Load the CSV File into DataFrame 'df1' ---
# The main data file in this archive is 'bank-additional/bank-additional-full.cs
# It's a CSV file, so we use pd.read_csv()
csv_file_path = 'bank-additional/bank-additional-full.csv'

print(f"\nLoading '{csv_file_path}' into DataFrame 'df1'...")
# Load the CSV file, noting that its separator is a semicolon ';'
df1 = pd.read_csv(csv_file_path, sep=';')
print("DataFrame 'df1' created successfully.")

# --- Final Result ---
# Display the head and info for the new DataFrame 'df1'
print("\n✅ Process Complete. Here is your new DataFrame:")
print("\n--- DataFrame df1 Head ---")
print(df1.head())

print("\n--- DataFrame df1 Info ---")
df1.info()
```

```
Downloading new dataset (Bank Marketing)...
Extracting files...
Files extracted: ['bank-additional/', 'bank-additional/.DS_Store', '__MACOSX/', 
'__MACOSX/bank-additional/', '__MACOSX/bank-additional/._.DS_Store', 'bank-addi
tional/.Rhistory', 'bank-additional/bank-additional-full.csv', 'bank-additiona
l/bank-additional-names.txt', 'bank-additional/bank-additional.csv', '__MACOS
X/._bank-additional']

Loading 'bank-additional/bank-additional-full.csv' into DataFrame 'df1'...
DataFrame 'df1' created successfully.

✅ Process Complete. Here is your new DataFrame:

--- DataFrame df1 Head ---
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone   
1   57   services  married  high.school  unknown      no   no  telephone   
2   37   services  married  high.school       no     yes   no  telephone   
3   40     admin.  married     basic.6y       no      no   no  telephone   
4   56   services  married  high.school       no      no  yes  telephone   

  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  
\
0   may         mon  ...         1    999         0  nonexistent          1.1   
1   may         mon  ...         1    999         0  nonexistent          1.1   
2   may         mon  ...         1    999         0  nonexistent          1.1   
3   may         mon  ...         1    999         0  nonexistent          1.1   
4   may         mon  ...         1    999         0  nonexistent          1.1   

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y  
0          93.994          -36.4      4.857       5191.0  no  
1          93.994          -36.4      4.857       5191.0  no  
2          93.994          -36.4      4.857       5191.0  no  
3          93.994          -36.4      4.857       5191.0  no  
4          93.994          -36.4      4.857       5191.0  no  

[5 rows x 21 columns]

--- DataFrame df1 Info ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Congratulations!

You have completed this lab, and you can now end the lab by following the lab guide instructions.

In [24]:

```
df1.shape
```

Out[24]:

```
(41188, 21)
```

In [25]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   pelvic_incidence          310 non-null    float64
 1   pelvic_tilt               310 non-null    float64
 2   lumbar_lordosis_angle     310 non-null    float64
 3   sacral_slope              310 non-null    float64
 4   pelvic_radius             310 non-null    float64
 5   degree_spondylolisthesis  310 non-null    float64
 6   class                     310 non-null    int64  
dtypes: float64(6), int64(1)
memory usage: 17.1 KB
```

In [26]:

```
df.describe()
```

Out[26]:

	pelvic_incidence	pelvic_tilt	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_s
count	310.000000	310.000000	310.000000	310.000000	310.000000	
mean	60.496653	17.542822	51.930930	42.953831	117.920655	
std	17.236520	10.008330	18.554064	13.423102	13.317377	
min	26.147921	-6.554948	14.000000	13.366931	70.082575	
25%	46.430294	10.667069	37.000000	33.347122	110.709196	
50%	58.691038	16.357689	49.562398	42.404912	118.268178	
75%	72.877696	22.120395	63.000000	52.695888	125.467674	
max	129.834041	49.431864	125.742385	121.429566	163.071041	

In [29]:

```
df1.columns
```

Out[29]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

In [30]:

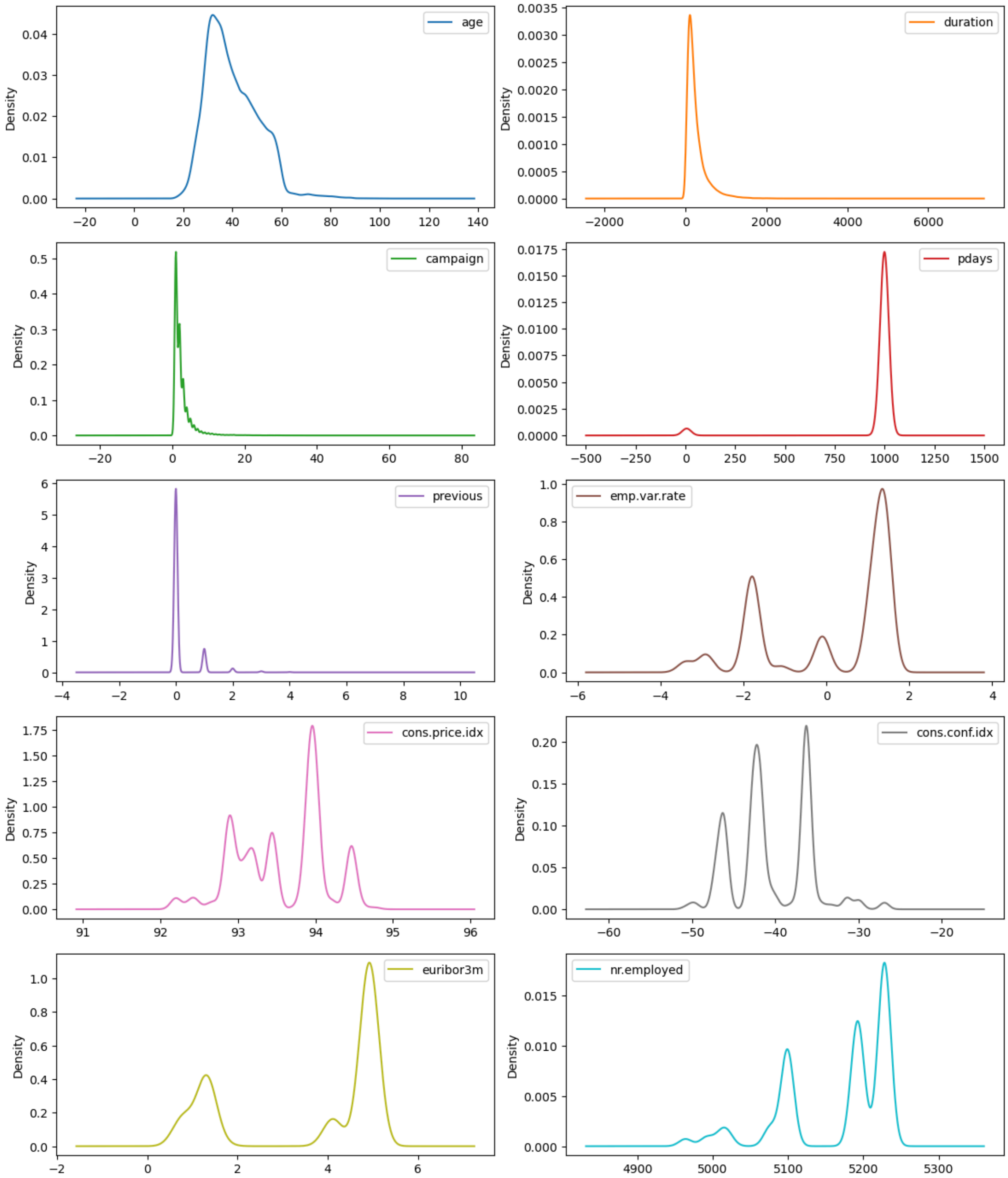
```
df1.dtypes
```

Out[30]:

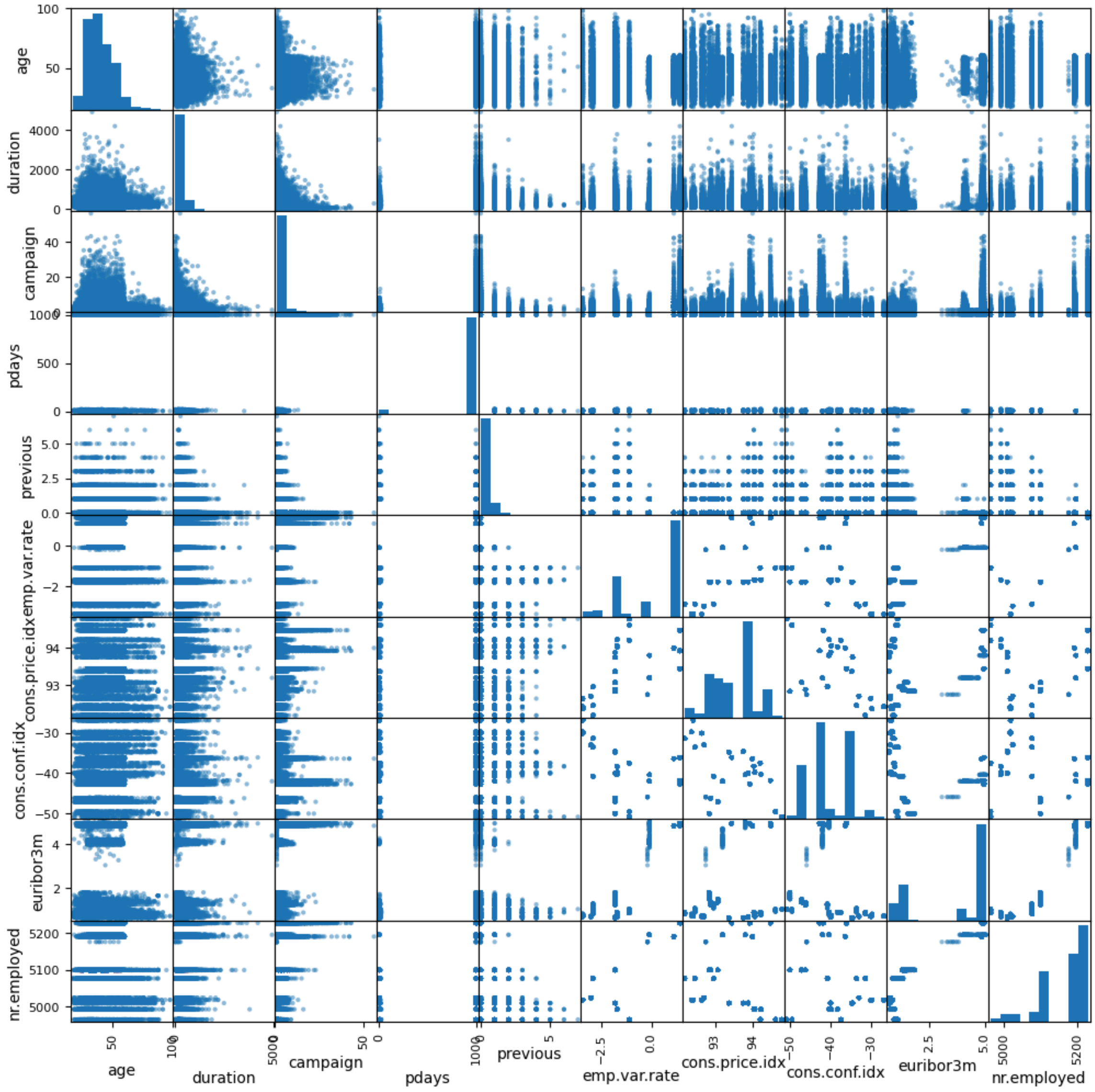
```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

In [33]:

```
import matplotlib.pyplot as plt

# Overwrite df1 with a version containing only its numeric columns
df1 = df1.select_dtypes(include='number')

# Plot the new df1, adjusting layout to fit all columns (5 rows, 2 columns)
df1.plot(kind='density', subplots=True, layout=(5, 2), figsize=(12, 14), sharex=

# Adjust layout to prevent labels from overlapping
plt.tight_layout()

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

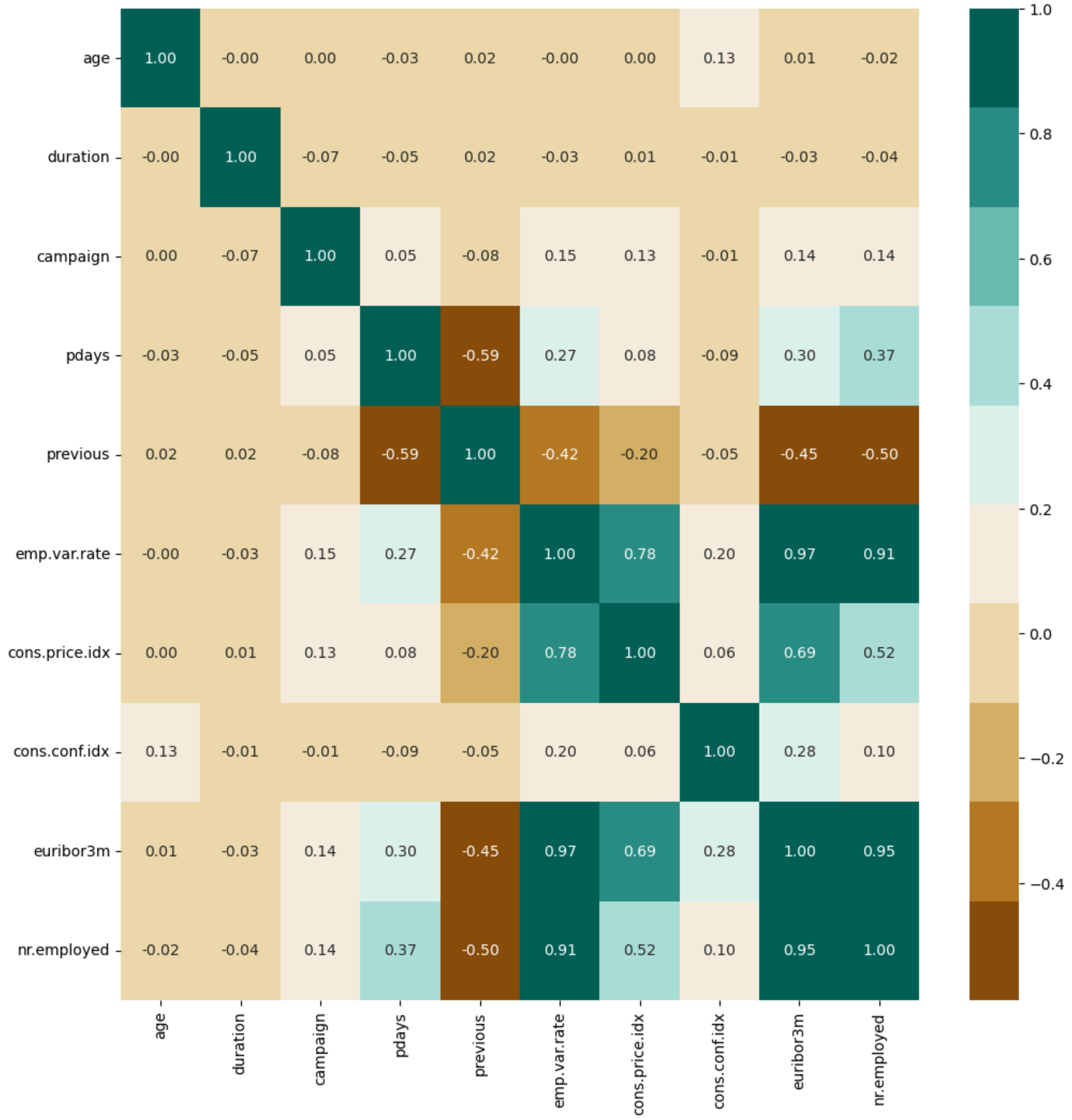


```
In [36]: pd.plotting.scatter_matrix(df1,figsize=(12,12))
         plt.show()
```



In [38]:

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

# Ensure df1 contains only numeric data (from the previous step)
df1 = df1.select_dtypes(include='number')

# 1. Calculate the correlation matrix for df1
corr_matrix = df1.corr()

# 2. Set up the plot
fig, ax = plt.subplots(figsize=(12, 12)) # Increased size for better readability

# 3. Generate the colormap
colormap = sns.color_palette("BrBG", 10)

# 4. Generate the Heatmap
#    - Pass the calculated corr_matrix
#    - Use the specified colormap
#    - Enable annotations (annot=True) and format them to two decimal places (fm
sns.heatmap(corr_matrix, cmap=colormap, annot=True, fmt=".2f")

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



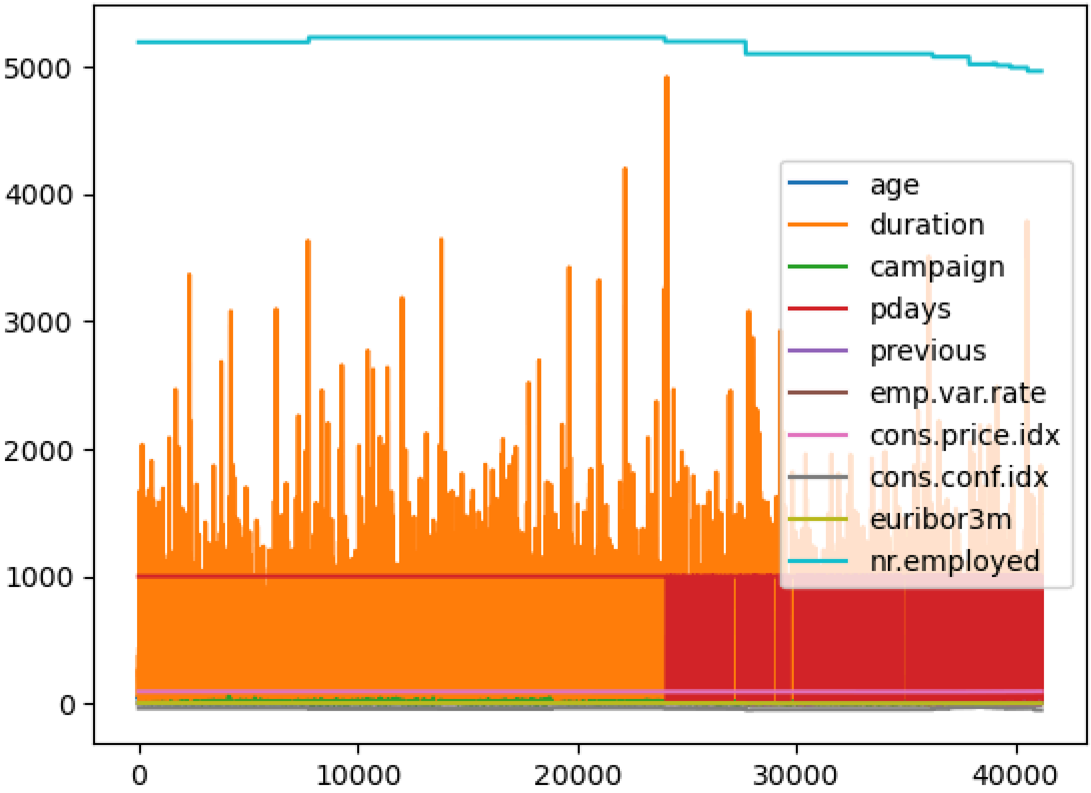
In [ ]:

In [22]:

```
import matplotlib.pyplot as plt
%matplotlib inline
df1.plot()
```

Out[22]:

```
<Axes: >
```


5000
4000
3000
2000
1000
0
0
10000
20000
30000
40000
age
duration
campaign
pdays
previous
emp.var.rate
cons.price.idx
cons.conf.idx
euribor3m
nr.employed

```
In [ ]:
```