Case Study 1 – Pandas

Problem Statement:

You work in XYZ Company as a Python developer. The company officials want you to build a Python program.

Tasks To Be Performed:

1. Write a function that takes start and end of a range returns a pandas series object containing numbers within that range. In case the user does not pass start or end or both they should default to 1 and 10 respectively. E.g: -> range_series() -> Should Return a pandas series from 1 to 10 range_series(5) -> Should Return a pandas series from 5 to 10 range_series(5, 10) -> Should Return a pandas series from 5 to 15 Create a method that takes n NumPy arrays of the same dimensions, sums them and returns the answer.
2. Create a function that takes in two lists named keys and values as arguments Keys would be strings and contain n string values Values would be a list containing n lists The methods should return a new pandas DataFrame with keys as column names and values as their corresponding values, e.g: ->create_dataframe(["One", "Two"], [["X", "Y"], ["A", "B"]]) -> should return a data frame One Two 0 X A 1 Y B
3. Create a function that concatenates two DataFrames. Use a previously created function to create two DataFrames and pass them as parameters Make sure that the indexes are reset before returning. Python for Data Science Certification Course
4. Write code to load data from cars.csv into a dataframe and print its details. Details like: 'count', 'mean', 'std', 'min', '25%', '50%', '75%', 'max'.
5. Write a method that will take a column name as argument and return the name of the column with which the given column has the highest correlation. The data to be used is the cars dataset. The returned value should not the column named that was passed as the parameters, e.g. : get_max_correlated_column('mpg') -> should return 'drat

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
def range_series(start=1,end=10):
    return pd.Series([i for i in range(start,end+1)])
```

In [7]:
```
range_series(5,15)
```

Out[7]:
```
0      5
1      6
2      7
3      8
4      9
5     10
6     11
7     12
8     13
9     14
10    15
dtype: int64
```

In [22]:
```
def create_dataframe(keys:list,values:list):
    d = {}
    for i,key in enumerate(keys):
        d[key] = values[i]
    return pd.DataFrame(d)
```

In [27]:
```
df1 = create_dataframe(["One", "Two"], [["X", "Y"], ["A", "B"]])
df1
```

Out[27]:

	One	Two
0	X	A
1	Y	B

In [28]:
```
df2 = create_dataframe(["Three", "Four"], [["W", "Z"], ["C", "D"]])
df2
```

Out[28]:

	Three	Four
0	W	C
1	Z	D

In [35]:
```
def con_df(df1,df2):
    return pd.concat([df1,df2]).reset_index(drop=True)
```

In [36]:
```
con_df(df1,df2)
```

Out[36]:

	One	Two	Three	Four
0	X	A	NaN	NaN
1	Y	B	NaN	NaN
2	NaN	NaN	W	C
3	NaN	NaN	Z	D

In [39]:
```
df = pd.read_csv(r"csv files\cars.csv")
df.head()
```

Out[39]:

	S.No	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

In [40]:
```
df.describe()
```

Out[40]:

	S.No	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	29.000000	32.000000	32.000000	32.000000	32.0000
mean	16.500000	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	17.674828	0.437500	0.406250	3.687500	2.8125
std	9.380832	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	1.780394	0.504016	0.498991	0.737804	1.6152
min	1.000000	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	14.500000	0.000000	0.000000	3.000000	1.0000
25%	8.750000	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	16.870000	0.000000	0.000000	3.000000	2.0000
50%	16.500000	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	17.420000	0.000000	0.000000	4.000000	2.0000
75%	24.250000	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	18.600000	1.000000	1.000000	4.000000	4.0000
max	32.000000	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	22.900000	1.000000	1.000000	5.000000	8.0000

In [42]:
```
df.count()
```

Out[42]:
```
S.No     32
model    32
mpg      32
cyl      32
disp     32
hp       32
drat     32
wt       32
qsec     29
vs       32
am       32
gear     32
carb     32
dtype: int64
```

In [43]:
```
df.mean()
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\3698961737.py:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.mean()
```

Out[43]:
```
S.No     16.500000
mpg      20.090625
cyl       6.187500
disp    230.721875
hp      146.687500
drat      3.596563
wt        3.217250
qsec     17.674828
vs        0.437500
am        0.406250
gear      3.687500
carb      2.812500
dtype: float64
```

In [44]:
```
df.std()
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\3390915376.py:1: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In addition, specifying 'numeric_only=None' is deprecated. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.std()
```

Out[44]:
```
S.No      9.380832
mpg       6.026948
cyl       1.785922
disp    123.938694
hp       68.562868
drat      0.534679
wt        0.978457
qsec      1.780394
vs        0.504016
am        0.498991
gear      0.737804
carb      1.615200
dtype: float64
```

In [45]:
```
df.min()
```

Out[45]:
```
S.No              1
model    AMC Javelin
mpg             10.4
cyl                4
disp            71.1
hp                52
drat            2.76
wt             1.513
qsec            14.5
vs                 0
am                 0
gear               3
carb               1
dtype: object
```

In [46]:
```
df.quantile(0.25)
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\3656653379.py:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.quantile(0.25)
```

Out[46]:
```
S.No      8.75000
mpg      15.42500
cyl       4.00000
disp    120.82500
hp       96.50000
drat      3.08000
wt        2.58125
qsec     16.87000
vs        0.00000
am        0.00000
gear      3.00000
carb      2.00000
Name: 0.25, dtype: float64
```

In [47]:
```
df.quantile(0.50)
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\3478652720.py:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.quantile(0.50)
```

Out[47]:
```
S.No     16.500
mpg      19.200
cyl       6.000
disp    196.300
hp      123.000
drat      3.695
wt        3.325
qsec     17.420
vs        0.000
am        0.000
gear      4.000
carb      2.000
Name: 0.5, dtype: float64
```

In [48]:
```
df.quantile(0.75)
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\3799946287.py:1: FutureWarning: The default value of numeric_only in DataFrame.quantile is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.quantile(0.75)
```

Out[48]:
```
S.No     24.25
mpg      22.80
cyl       8.00
disp    326.00
hp      180.00
drat      3.92
wt        3.61
qsec     18.60
vs        1.00
am        1.00
gear      4.00
carb      4.00
Name: 0.75, dtype: float64
```

In [49]:
```
df.max()
```

Out[49]:
```
S.No            32
model    Volvo 142E
mpg            33.9
cyl               8
disp          472.0
hp              335
drat           4.93
wt            5.424
qsec           22.9
vs                1
am                1
gear              5
carb              8
dtype: object
```

In [74]:
```
def get_max_correlated_column(col):
    return df.corr()[col].sort_values(ascending=False).index[1]
```

In [77]:
```
get_max_correlated_column('mpg')
```

```
C:\Users\Roy\AppData\Local\Temp\ipykernel_624\750348240.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  return df.corr()[col].sort_values(ascending=False).index[1]
```

Out[77]:
```
'drat'
```

In [ ]:
```
```