Assignment 10 | Pandas Exercises 2

Akash Duttachowdhury

21052386

TASK: Import pandas

In [ ]:
```
# CODE HERE
import pandas as pd
```

TASK: Read the bank.csv file

In [ ]:
```
# CODE HERE
df = pd.read_csv('bank.csv')
```

TASK: Display the first 5 rows of the data set

In [ ]:
```
# CODE HERE
df.head(5)
```

Out[ ]:

	age	job	marital	education	default	balance	housing	loan	contact
0	30	unemployed	married	primary	no	1787	no	no	cellular
1	33	services	married	secondary	no	4789	yes	yes	cellular
2	35	management	single	tertiary	no	1350	yes	no	cellular
3	30	management	married	tertiary	no	1476	yes	yes	unknown
4	59	blue-collar	married	secondary	no	0	yes	no	unknown

TASK: What is the average (mean) age of the people in the dataset?

In [ ]:
```
# CODE HERE
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        4521 non-null   int64 
 1   job        4521 non-null   object
 2   marital    4521 non-null   object
 3   education  4521 non-null   object
 4   default    4521 non-null   object
 5   balance    4521 non-null   int64 
 6   housing    4521 non-null   object
 7   loan       4521 non-null   object
 8   contact    4521 non-null   object
 9   day        4521 non-null   int64 
 10  month      4521 non-null   object
 11  duration   4521 non-null   int64 
 12  campaign   4521 non-null   int64 
 13  pdays      4521 non-null   int64 
 14  previous   4521 non-null   int64 
 15  poutcome   4521 non-null   object
 16  y          4521 non-null   object
dtypes: int64(7), object(10)
memory usage: 600.6+ KB
```

In [ ]:
```
df['age'].mean()
```

Out[ ]:
```
41.17009511170095
```

TASK: What is the marital status of the youngest person in the dataset?

HINT

In [ ]:
```
# CODE HERE
# Find the index of the youngest person
youngest_index = df['age'].idxmin()

# Get the marital status of the youngest person
youngest_marital_status = df.loc[youngest_index, 'marital']

print(f"The marital status of the youngest person is: {youngest_marital_s
```

```
The marital status of the youngest person is: single
```

TASK: How many unique job categories are there?

In [ ]:
```
# CODE HERE
# Count the number of unique job categories
unique_job_categories_count = df['job'].nunique()

print(f"There are {unique_job_categories_count} unique job categories in
```

```
There are 12 unique job categories in the dataset.
```

TASK: How many people are there per job category? (Take a peek at the expected output)

In [ ]:
```
# CODE HERE
# Group the data by job category and count the number of people in each c
people_per_job = df.groupby('job')['job'].count()

print("Number of people per job category:")
print(people_per_job)
```

```
Number of people per job category:
job
admin.           478
blue-collar      946
entrepreneur     168
housemaid        112
management       969
retired          230
self-employed    183
services         417
student           84
technician       768
unemployed       128
unknown           38
Name: job, dtype: int64
```

TASK: What percent of people in the dataset were married?

In [ ]:
```
#CODE HERE
# Count the number of people who were married
married_count = df[df['marital'] == 'married'].shape[0]

# Calculate the total number of people in the dataset
total_people = df.shape[0]

# Calculate the percentage of people who were married
married_percentage = (married_count / total_people) * 100

print(f"The percentage of people in the dataset who were married: {marrie
```

```
The percentage of people in the dataset who were married: 61.87%
```

TASK: There is a column labeled "default". Use pandas' .map() method to create a new column called "default code" which contains a 0 if there was no default, or a 1 if there was a default. Then show the head of the dataframe with this new column.

Helpful Hint Link One

Helpful Hint Link Two

In [ ]:
```
# CODE HERE
```

```
# Create a mapping dictionary for default values
default_mapping = {'no': 0, 'yes': 1}

# Create the new column 'default code' using .map() method
df['default code'] = df['default'].map(default_mapping)

# Show the head of the dataframe with the new column
print(df.head())
```

```
   age          job  marital  education default  balance housing loan  \
0   30   unemployed  married    primary      no     1787      no   no
1   33     services  married  secondary      no     4789     yes  yes
2   35   management   single   tertiary      no     1350     yes   no
3   30   management  married   tertiary      no     1476     yes  yes
4   59  blue-collar  married  secondary      no        0     yes   no

    contact  day month  duration  campaign  pdays  previous poutcome   y
\
0  cellular   19   oct        79         1     -1         0  unknown  no
1  cellular   11   may       220         1    339         4  failure  no
2  cellular   16   apr       185         1    330         1  failure  no
3   unknown    3   jun       199         4     -1         0  unknown  no
4   unknown    5   may       226         1     -1         0  unknown  no

   default code
0             0
1             0
2             0
3             0
4             0
```

TASK: Using pandas .apply() method, create a new column called "marital code". This column will only contained a shortened code of the possible marital status first letter. (For example "m" for "married" , "s" for "single" etc... See if you can do this with a lambda expression. Lots of ways to do this one!

Hint Link

In [ ]:
```
# CODE HERE
# Create a new column 'marital code' using .apply() method and lambda fun
df['marital code'] = df['marital'].apply(lambda x: x[0].lower())

# Show the head of the dataframe with the new column
print(df.head)
```

```
<bound method NDFrame.head of       age           job  marital  education
default  balance housing loan  \
0      30     unemployed  married    primary      no     1787      no   no
1      33       services  married  secondary      no     4789     yes  yes
2      35     management   single   tertiary      no     1350     yes   no
3      30     management  married   tertiary      no     1476     yes  yes
4      59    blue-collar  married  secondary      no        0     yes   no
...   ...            ...      ...        ...     ...      ...     ...  ...
4516   33       services  married  secondary      no     -333     yes   no
4517   57  self-employed  married   tertiary     yes    -3313     yes  yes
4518   57     technician  married  secondary      no      295      no   no
4519   28    blue-collar  married  secondary      no     1137      no   no
4520   44   entrepreneur   single   tertiary      no     1136     yes  yes

       contact  day month  duration  campaign  pdays  previous poutcome
y  \
0     cellular   19   oct        79         1     -1         0  unknown  n
o
1     cellular   11   may       220         1    339         4  failure  n
o
2     cellular   16   apr       185         1    330         1  failure  n
o
3      unknown    3   jun       199         4     -1         0  unknown  n
o
4      unknown    5   may       226         1     -1         0  unknown  n
o
...        ...  ...   ...       ...       ...    ...       ...      ...
..
4516  cellular   30   jul       329         5     -1         0  unknown  n
o
4517   unknown    9   may       153         1     -1         0  unknown  n
o
4518  cellular   19   aug       151        11     -1         0  unknown  n
o
4519  cellular    6   feb       129         4    211         3    other  n
o
4520  cellular    3   apr       345         2    249         7    other  n
o

      default code marital code
0                0            m
1                0            m
2                0            s
3                0            m
4                0            m
...            ...          ...
4516             0            m
4517             1            m
4518             0            m
4519             0            m
4520             0            s

[4521 rows x 19 columns]>
```

TASK: What was the longest lasting duration?

In [ ]:
```
# CODE HERE
# Find the longest lasting duration
longest_duration = df['duration'].max()

print(f"The longest lasting duration in the dataset was: {longest_duratio
```

```
The longest lasting duration in the dataset was: 3025 seconds
```

TASK: What is the most common education level for people who are unemployed?

In [ ]:
```
# CODE HERE
# Filter data for unemployed individuals
unemployed_data = df[df['job'] == 'unemployed']

# Find the most common education level among unemployed individuals
common_education_unemployed = unemployed_data['education'].mode()[0]

print(f"The most common education level for unemployed individuals is: {c
```

```
The most common education level for unemployed individuals is: secondary
```

TASK: What is the average (mean) age for being unemployed?

In [ ]:
```
# CODE HERE
# Filter data for unemployed individuals
unemployed_data = df[df['job'] == 'unemployed']

# Calculate the average age for unemployed individuals
average_age_unemployed = unemployed_data['age'].mean()

print(f"The average age for unemployed individuals is: {average_age_unemp
```

```
The average age for unemployed individuals is: 40.91 years
```