Instructions – Answer each question completely and concisely. Partial credit will be given. Unless otherwise noted, you should perform all operations using Pandas functions rather than writing a loop.

1. Write the letter of the method that performs the requested operation on a Pandas DataFrame. Each option will only be used once, but not all the options will be used. (8 points).

__L__ Displays a specified number of rows from the beginning of a DataFrame

__I__ Selects a row from a DataFrame based on an index value

__C__ Selects a row from a DataFrame based on its row number

__J__ Creates a new Series by applying a function to an existing Series

__E__ Combines rows with the same values in one or more columns for the purpose of computing an aggregate function

__A__ Creates a new Series that tells how many times each distinct value occurs in a given Series

__D__ Returns a list containing all of the unique values in a Series

__O__ Displays a listing of the data types for each of the columns in a data frame

A. value_counts
B. distinct
C. iloc
D. unique
E. groupby
F. by_index
G. describe
H. display_first
I. loc
J. map
K. combine
L. head
M. count_distinct
N. row
O. info
P. aggregate

2. Write one or more statements that create a data frame named `faculty` containing the data shown on the loose sheet. (6 points)

```
fac_data = {
"Last": ["McFall", "Olagbemi", "DeJongh"],
"Office": ["VWF 220", "VWF 232", "VWF 229"],
"Year Hired": [2000, 2021, 2002]
}
ids = ["1234", "5678", "1010"]
faculty = pd.DataFrame(data=fac_data, index=ids)
```

3. Write a statement that uses the `loc` function to display the `Year Hired` for Dr. Olagbemi. (3 points)

```
faculty.loc["5678", "Year Hired"]
```
or
```
faculty.loc["5678"]["Year Hired"]
```

4. Write a statement that uses the `iloc` function to display the `Year Hired` for Dr. Olagbemi. (3 points)

```
faculty.iloc[1]["Year Hired"]
```

5. Write a statement that displays only the `Last` and `Year Hired` columns for the first 2 rows in `faculty`. (3 points)

```
faculty[["Last", "Year Hired"]].head(n=2)
```
or
```
faculty.iloc[0:2, ["Last", "Year Hired"]]
```

6. Write one or more statements that use the `map` function to add a column named `email` to the `faculty` data frame. The values in the `email` column should be the **lower case** values of the `Last` column, followed by the string @hope.edu. If `s` is a string, you can call `s.lower()` to get the lowercase version of `s`. (5 points)

```
def convert(s):
   return f"{s.lower()}@hope.edu"

faculty['email'] = faculty['Last'].map(convert)
```

or

```
faculty['email'] = faculty['Last'].map (
  lambda last: f"{last.lower()}@hope.edu"
)
```

7. Suppose we have a folder containing files 1999.txt, 2000.txt, 2001.txt, 2002.txt, where the number in the file name represents a year. Each line in a file contains a Name, Sex, and the number of children born that year for the Name and Sex. The first 2 lines of each file are similar to this:

```
Emily,F,25949
Hannah,F,23066
```

Your task is to read the contents of all 4 files into separate data frames, and then create a single data frame combining the contents of the 4 files together. Fill in the blanks in the following code `names` is a data frame containing the contents of all the files, with columns `Name`, `Sex`, `Births`, and `Year`. (5 points)

```
#  Use a list comprehension to create a list of data frames,
#  one per file
columns = ['Name', 'Sex', 'Births']
years = np.arange( 1999 , 2003)
frames = [
  filename = f'{year}.txt'
  pd.read_csv (filename, names = columns)
  for year in years
]

#  Set the value of the year column for each data frame
for i, year in enumerate(years):
    frames[i]["Year"] = year
#Combine the data frames into a single data frame
names = pd.concat (frames, axis= 0 )
```

For the remaining questions assume `names` is the data frame created in the previous question, with data loaded for the years 2000-2010. A reference for the DataFrame structure is provided on the loose sheet.

8. Write code that displays the total number of births in the year 2000. (3 points)

```
names[names["Year"] == 2000]["Births"].sum()
```

Starting with question 9, you may reuse any variables defined in a previous question.

9. Write code that displays the number of girls named Lauryn born in the year 2000. (4 points)

```
year_2000 = names["Year"] == 2000
lauryn = names["Name"] == "Lauryn"
girls = names["Sex"] == "F"
names[year_2000 & lauryn & girls]["Births"]
```

Also correct

```
names.iloc[year_2000 & lauryn & girls, "Births"]
```

10. Write code to process the data in `names` to create a data frame named `b2000s` containing the total number of births by Sex in the 2000s (2000-2009). The total births column is computed in **millions**. Your answer must not use a loop. Here's what the first 2 years of data might look like. (5 points)

		Births
Year	**Sex**	
2000	**F**	1.81
	M	1.96
2001	**F**	1.80
	M	1.94

```
y2k = (names['Year'] >= 2000) & (names['Year'] <= 2009)
b2000s = names[y2k].groupby(['Year', 'Sex']).sum() / 1e6
```

Since it doesn't make sense to perform a sum on the "Name" column, it will not be included in the answer. It's OK if you didn't realize this and tried to eliminate it yourself.

11. Draw a picture showing what the result of calling `b2000s.unstack()` would be, assuming `b2000s` contains only the data in the table above. (4 points)

Year	F	M
2000	1.81	1.96
2001	1.80	1.95

12. Write code that adds a `Length` column to `names`. The values in the `Length` column should be the **number of letters** in the `Name` column. Calculate the number of letters in a name using the vectorized string function `len`, which takes no arguments. Sample output is shown on the loose sheet. (3 points)

```
names["Length"] = names["Name"].str.len()
```

13. Write code that computes the **most common name length** for girls born in the year 2000 who were not named Olivia. Store the result in a variable named `most_common_length`. (5 points)

```
not_olivia = names["Name"] != "Olivia"
candidates = names[girls & year_2000 & not_olivia]

counts  = candidates['Length'].value_counts()

most_common_length = counts.index[0]
```

14. Consider the set of names given to babies born in 2000 whose names are length `most_common_length`. Write code to answer the question "Of those names, which name(s) had the most births?" If `most_common_length` has the value 6, the output would look like the table below. (4 points)

Name	Length	Births
Hannah	6	23066
Ashley	6	23066

```
born_2000 = names[year_2000]
right_length_mask = born_2000['Length'] == most_common_length
the_names = born_2000[right_length_mask]
highest_births = the_names['Births'].max()
final_names = the_names[the_names["Births"] == highest_births]
final_names[["Name", "Length", "Births"]]
```

Question 2. The Year Hired column is an integer; all other values shown are strings. "Index" is the Data Frame's **index**, and not a column in the table.

Index	Last	Office	Year Hired
1234	McFall	VWF 220	2000
5678	Olagbemi	VWF 232	2021
1010	DeJongh	VWF 229	2002

Questions 8 through 11. The table below shows the structure of the `names` DataFrame, with some sample data.

Name	Sex	Births	Year
Emily	F	25949	2000
Henry	M	21347	2000
Emily	F	26123	2001
Bob	M	12345	2002

Questions 12 through 14 The table below shows the desired structure of the names DataFrame after the Length column has been added, with some sample data.

Name	Sex	Births	Year	Length
Emily	F	25949	2000	6
Henry	M	21347	2000	5
Emily	F	26123	2001	6
Bob	M	12345	2002	3