

STA 130 HW1

September 10, 2024

STA 130 Week 1 Homework

QUESTION 1

<https://chatgpt.com/share/7f42c95d-01cc-4cbf-a7a3-57edbd7f18e7> (ChatGPT conversation for question 1)

```
[12]: import pandas as pd

url = 'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
      ↪data/2020/2020-05-05/villagers.csv'

df = pd.read_csv(url)

missing_values = df.isnull().sum()

print("Missing values in each column:")
print(missing_values)

total_missing = missing_values.sum()
print("\nTotal missing values in the dataset:", total_missing)
```

Missing values in each column:

row_n	0
id	1
name	0
gender	0
species	0
birthday	0
personality	0
song	11
phrase	0
full_id	0
url	0
dtype: int64	

Total missing values in the dataset: 12

QUESTION 2

ChatGPT conversation link for question 2: <https://chatgpt.com/share/715a35bc-f635-4c7c-8266-52f10906a4aa>

```
[13]: import pandas as pd

url = 'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
      ↪data/2020/2020-05-05/villagers.csv'
df = pd.read_csv(url)

num_rows, num_columns = df.shape

print(f'Number of rows: {num_rows}')
print(f'Number of columns: {num_columns}')
```

Number of rows: 391

Number of columns: 11

Definition of “observation”: An observation is an individual data point in a data set. An observation about an individual includes all their characteristics that are being included in the data set.

Definition of “variable”: A variable is one single characteristic of an individual, which can be either quantitative (numerical) or qualitative (non-numerical), depending on which one makes sense for measuring that particular piece of data

QUESTION 3

ChatGPT conversation link for question 3: <https://chatgpt.com/share/a25cb5e9-8986-49ba-a9bf-f725bf0f9858>

```
[14]: import pandas as pd

url = 'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
      ↪data/2020/2020-05-05/villagers.csv'
df = pd.read_csv(url)

print("First few rows of the dataset:")
df.head()

print("\nSummary of each column:")
print(df.describe(include='all'))
```

First few rows of the dataset:

Summary of each column:

	row_n	id	name	gender	species	birthday	personality \
count	391.000000	390	391	391	391	391	391
unique	NaN	390	391	2	35	361	8
top	NaN	admiral	Admiral	male	cat	1-27	lazy
freq	NaN	1	1	204	23	2	60
mean	239.902813	NaN	NaN	NaN	NaN	NaN	NaN

std	140.702672	NaN	NaN	NaN	NaN	NaN	NaN
min	2.000000	NaN	NaN	NaN	NaN	NaN	NaN
25%	117.500000	NaN	NaN	NaN	NaN	NaN	NaN
50%	240.000000	NaN	NaN	NaN	NaN	NaN	NaN
75%	363.500000	NaN	NaN	NaN	NaN	NaN	NaN
max	483.000000	NaN	NaN	NaN	NaN	NaN	NaN

	song	phrase	full_id \
count	380	391	391
unique	92	388	391
top	K.K. Country	wee one	villager-admiral
freq	10	2	1
mean	NaN	NaN	NaN
std	NaN	NaN	NaN
min	NaN	NaN	NaN
25%	NaN	NaN	NaN
50%	NaN	NaN	NaN
75%	NaN	NaN	NaN
max	NaN	NaN	NaN

	url
count	391
unique	391
top	https://villagerdb.com/images/villagers/thumb/...
freq	1
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

```
[15]: # New summary after removing column 1 (row number) and including only
      ↳ quantitative data to remove all the NaNs

import pandas as pd

url = 'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
      ↳ data/2020/2020-05-05/villagers.csv'
df = pd.read_csv(url)

df_numeric = df.iloc[:, 1:]

print("Summary of quantitative columns (excluding the first column):")
print(df_numeric.describe())
```

Summary of quantitative columns (excluding the first column):

	id	name	gender	species	birthday	personality	song \
count	390	391	391	391	391	391	380
unique	390	391	2	35	361	8	92
top	admiral	Admiral	male	cat	1-27	lazy	K.K. Country
freq	1	1	204	23	2	60	10

	phrase	full_id \
count	391	391
unique	388	391
top	wee one villager-admiral	
freq	2	1

	url
count	391
unique	391
top	https://villagerdb.com/images/villagers/thumb/...
freq	1

QUESTION 4

```
[5]: import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

num_rows, num_columns = df.shape

print(f'Number of rows: {num_rows}')
print(f'Number of columns: {num_columns}')

print("\nSummary of each column:")
print(df.describe(include='all'))
```

Number of rows: 891

Number of columns: 15

Summary of each column:

	survived	pclass	sex	age	sibsp	parch \
count	891.000000	891.000000	891	714.000000	891.000000	891.000000
unique	NaN	NaN	2	NaN	NaN	NaN
top	NaN	NaN	male	NaN	NaN	NaN
freq	NaN	NaN	577	NaN	NaN	NaN
mean	0.383838	2.308642	NaN	29.699118	0.523008	0.381594
std	0.486592	0.836071	NaN	14.526497	1.102743	0.806057
min	0.000000	1.000000	NaN	0.420000	0.000000	0.000000
25%	0.000000	2.000000	NaN	20.125000	0.000000	0.000000
50%	0.000000	3.000000	NaN	28.000000	0.000000	0.000000

75%	1.000000	3.000000	NaN	38.000000	1.000000	0.000000
max	1.000000	3.000000	NaN	80.000000	8.000000	6.000000

	fare	embarked	class	who	adult_male	deck	embark_town	alive	\
count	891.000000	889	891	891	891	203	889	891	
unique	NaN	3	3	3	2	7	3	2	
top	NaN	S	Third	man	True	C	Southampton	no	
freq	NaN	644	491	537	537	59	644	549	
mean	32.204208	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	49.693429	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	7.910400	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	14.454200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	31.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	512.329200	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	alone
count	891
unique	2
top	True
freq	537
mean	NaN
std	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN

Description for Question 4 (no chatbot used for this question):

The discrepancies between the total number of rows (891) and the count for each column (less than 891 for some columns) exist because of the missing values in the data set. For example, there is a large discrepancy for the column titled “deck,” with only 203 values provided out of 891 possible. This is likely because, for most passengers, the deck they were on when the Titanic sank is unknown, and therefore not included in the data set.

QUESTION 5

(methods vs attributes):

In Python, an attribute is a variable which contains data about a particular class/instance/object. A method is a function that belongs to a class, and determines the behavior of its instances. Methods can access and change the attributes of their class or instance.

For example, the attribute “df.shape” provides the number of rows and columns for the data sets above, while the method “df.describe()” creates a table summary of the data, calculating the mean, standard deviation, max, min, and other characteristics of the data sets.

ChatGPT link for question 5: <https://chatgpt.com/share/08df9a6d-4613-45a4-b92b-86548321ac46>

ChatGPT summaries (questions 1-5):

Question1 (<https://chatgpt.com/share/7f42c95d-01cc-4cbf-a7a3-57edbd7f18e7>):

“Here’s a summary of our conversation:

1. Python Code for Missing Values:

I provided a Python script using the pandas library to import a CSV dataset and count missing values. The code reads a CSV file, counts missing values in each column, and prints the results.

2. Code for Specific URL:

You requested a specific example using a URL from a GitHub repository. I adapted the previous script to load the CSV file directly from the provided URL and count the missing values in the dataset.”

Question2 (<https://chatgpt.com/share/715a35bc-f635-4c7c-8266-52f10906a4aa>):

“In this conversation, we discussed:

Determining Rows and Columns: To find the number of rows and columns in a dataset, you can use Python and the pandas library. The code reads the CSV file and uses the shape attribute to get these counts.

Observation in Statistics: An observation refers to a single data point or instance within a dataset. For example, in a dataset of test scores, each score is an observation.

Variable in Statistics: A variable is a characteristic or attribute that can take on different values. Variables can be quantitative (e.g., height, weight) or qualitative (e.g., hair color, major) and are essential for data analysis.”

Question3 (<https://chatgpt.com/share/a25cb5e9-8986-49ba-a9bf-f725bf0f9858>):

“Here’s a summary of our conversation:

1. Request: You asked how to provide summaries of the columns in a dataset from a specified URL using Python code.
2. Initial Solution:

I provided Python code using the pandas library to load the dataset and generate a summary of all columns using `df.describe(include='all')`.

3. Refinement:

You requested to exclude the first column and only include quantitative values. I updated the code to exclude the first column and provide a summary of only the quantitative columns using `df.iloc[:, 1:]` and `df_numeric.describe()`. If you have any further questions or need more adjustments, feel free to ask!”

Question5 (<https://chatgpt.com/share/08df9a6d-4613-45a4-b92b-86548321ac46>):

“Sure! Here’s a summary:

Attributes vs. Methods in Python:

Attributes are variables that store data or state within a class. They can be instance-specific (unique to each object) or class-wide (shared among all instances). Methods are functions defined within

a class that describe behaviors or actions. They can be instance methods (operate on instance attributes), class methods (operate on class-level data), or static methods (don't access instance or class data).

Class vs. Instance in Python:

A class is a blueprint for creating objects, defining the structure (attributes) and behaviors (methods) of the objects. An instance is a specific object created from a class, with its own unique set of attributes and the ability to use the methods defined by the class.

Mathematical Functions vs. Programming Functions:

Mathematical functions define relationships between inputs and outputs, focusing on abstract properties like continuity and limits. Programming functions are blocks of code designed to perform specific tasks, taking inputs, processing them, and potentially returning outputs. They are practical and concerned with code execution within a program.

Each type of function or concept serves its own purpose within its respective domain—mathematical functions for theoretical analysis and programming functions for practical code execution.”

.

POST LECTURE QUESTIONS

QUESTION 6

(no ChatGPT used)

- a) The “count” is the total number of values a variable in the data set has
- b) The “mean” is the average of all of a variable's values in the data set (sum of all values / number of values)
- c) The “std” is the standard deviation. This describes how far the values are from the mean. A higher standard deviation means that there is a larger range of values.
- d) The “min” is the lowest value a variable takes in the data set.
- e) “25%” is the 25th percentile, which means that 25% of the values are lower than it, and 75% are greater.
- f) “50%” is the 50th percentile, or median, meaning that half of the values are lower than this, and half are greater.
- g) “75%” is the 75th percentile, so 25% of values are greater than it, and 75% are lower.
- h) The “max” is the greatest value that a variable takes in the data set.

QUESTION 7

chatgpt link: <https://chatgpt.com/share/7b87e828-5928-4bfb-a570-cf2588184604>

7.1) `df.dropna()` is used to remove rows (or columns, but the default is rows) that contain missing values from a dataset. So, it's used primarily in cases when you want to clean/organize the data set better by removing missing or incomplete sections.

7.2) `del df['col']` is used to delete entire columns from the data set. If you don't need a column and want it completely gone, you would use this instead of `df.dropna()`.

7.3) Applying `del df['col']` first will delete the column and all its missing (NaN) values. This will prevent the `df.dropna()` function from deleting rows which have missing values only in the deleted column, so not as many rows will be deleted, resulting in a larger data set.

```
[1]: # question 7.4 (using the titanic data set from above)

# before using del df['col'] or df.dropna()

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

num_rows, num_columns = df.shape

print(f'Number of rows: {num_rows}')
print(f'Number of columns: {num_columns}')

print("\nSummary of each column:")
print(df.describe(include='all'))
```

Number of rows: 891

Number of columns: 15

Summary of each column:

	survived	pclass	sex	age	sibsp	parch	\
count	891.000000	891.000000	891	714.000000	891.000000	891.000000	
unique	NaN	NaN	2	NaN	NaN	NaN	
top	NaN	NaN	male	NaN	NaN	NaN	
freq	NaN	NaN	577	NaN	NaN	NaN	
mean	0.383838	2.308642	NaN	29.699118	0.523008	0.381594	
std	0.486592	0.836071	NaN	14.526497	1.102743	0.806057	
min	0.000000	1.000000	NaN	0.420000	0.000000	0.000000	
25%	0.000000	2.000000	NaN	20.125000	0.000000	0.000000	
50%	0.000000	3.000000	NaN	28.000000	0.000000	0.000000	
75%	1.000000	3.000000	NaN	38.000000	1.000000	0.000000	
max	1.000000	3.000000	NaN	80.000000	8.000000	6.000000	

	fare	embarked	class	who	adult_male	deck	embark_town	alive	\
count	891.000000	889	891	891	891	203	889	891	
unique	NaN	3	3	3	2	7	3	2	
top	NaN	S	Third	man	True	C	Southampton	no	
freq	NaN	644	491	537	537	59	644	549	
mean	32.204208	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	49.693429	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	7.910400	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

50%	14.454200	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	31.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	512.329200	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```

alone
count    891
unique     2
top      True
freq     537
mean     NaN
std      NaN
min      NaN
25%     NaN
50%     NaN
75%     NaN
max      NaN

```

```

[2]: # same data set, after applying first del df['col'] to the "deck" column, then
      ↪df.dropna()

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

num_rows, num_columns = df.shape

print(f'Number of rows: {num_rows}')
print(f'Number of columns: {num_columns}')

del df['deck']

print(df.dropna())

# this removed the "deck column," and removed all other rows with missing values
# so, the data set now has 14 columns and 712 rows, rather than the original 15
      ↪columns and 891 rows

```

Number of rows: 891

Number of columns: 15

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	\
0	0	3	male	22.0	1	0	7.2500	S	Third	
1	1	1	female	38.0	1	0	71.2833	C	First	
2	1	3	female	26.0	0	0	7.9250	S	Third	
3	1	1	female	35.0	1	0	53.1000	S	First	
4	0	3	male	35.0	0	0	8.0500	S	Third	

..
885	0	3	female	39.0	0	5	29.1250	Q	Third	
886	0	2	male	27.0	0	0	13.0000	S	Second	
887	1	1	female	19.0	0	0	30.0000	S	First	
889	1	1	male	26.0	0	0	30.0000	C	First	
890	0	3	male	32.0	0	0	7.7500	Q	Third	

	who	adult_male	embark_town	alive	alone
0	man	True	Southampton	no	False
1	woman	False	Cherbourg	yes	False
2	woman	False	Southampton	yes	True
3	woman	False	Southampton	yes	False
4	man	True	Southampton	no	True
..
885	woman	False	Queenstown	no	False
886	man	True	Southampton	no	True
887	woman	False	Southampton	yes	True
889	man	True	Cherbourg	yes	True
890	man	True	Queenstown	no	True

[712 rows x 14 columns]

QUESTION 8

```
[3]: # question 8.1

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```
[3]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

8.1) explanation: This code took all the individuals from the “survived” column and divided them into groups of survivors, and non-survivors. It then calculated statistics about the ages of the people in each group, such as their average age, median age, oldest, and youngest individuals.

8.2) df.describe does not remove missing values, so it can’t calculate the statistics for any column with any missing values, instead giving the result NaN. Therefore, if I did df.describe, no rows would be removed, and so there would be a total count of 891. However, df.groupby DOES remove rows with missing values in the relevant columns (“survived” and “age”) automatically, so the count in the table from question 8.1 only totals to 714.

```
[7]: # 8.3.A)

# did not import pandas
url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[7], line 4
      1 # 8.3.A)
      3 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
      ↪titanic.csv'
----> 4 df = pd.read_csv(url)
      6 df.groupby("survived")["age"].describe()

NameError: name 'pd' is not defined
```

```
[8]: # 8.3.A) fixed with ChatGPT and Google search help

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```
[8]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

```
[9]: # 8.3.B)

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanics.
      ↪csv' # added an s to titanic in url
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```

HTTPError                                     Traceback (most recent call last)
Cell In[9], line 6
      3 import pandas as pd
      5 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
↳titanics.csv'
----> 6 df = pd.read_csv(url)
      8 df.groupby("survived")["age"].describe()

File /opt/conda/lib/python3.11/site-packages/pandas/io/parsers/readers.py:948,
↳in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col,
↳usecols, dtype, engine, converters, true_values, false_values,
↳skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na,
↳na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format,
↳keep_date_col, date_parser, date_format, dayfirst, cache_dates, iterator,
↳chunksize, compression, thousands, decimal, lineterminator, quotechar,
↳quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect
↳on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision,
↳storage_options, dtype_backend)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
    (...)
    944     dtype_backend=dtype_backend,
    945 )
    946 kwds.update(kwds_defaults)
--> 948 return _read(filepath_or_buffer, kwds)

File /opt/conda/lib/python3.11/site-packages/pandas/io/parsers/readers.py:611,
↳in _read(filepath_or_buffer, kwds)
    608 _validate_names(kwds.get("names", None))
    610 # Create the parser.
--> 611 parser = TextFileReader(filepath_or_buffer, **kwds)
    613 if chunksize or iterator:
    614     return parser

File /opt/conda/lib/python3.11/site-packages/pandas/io/parsers/readers.py:1448,
↳in TextFileReader.__init__(self, f, engine, **kwds)
    1445     self.options["has_index_names"] = kwds["has_index_names"]
    1447 self.handles: IOHandles | None = None
-> 1448 self._engine = self._make_engine(f, self.engine)

File /opt/conda/lib/python3.11/site-packages/pandas/io/parsers/readers.py:1705,
↳in TextFileReader._make_engine(self, f, engine)
    1703     if "b" not in mode:
    1704         mode += "b"
-> 1705 self.handles = get_handle(
    1706     f,
    1707     mode,
    1708     encoding=self.options.get("encoding", None),
    1709     compression=self.options.get("compression", None),

```

```

1710     memory_map=self.options.get("memory_map", False),
1711     is_text=is_text,
1712     errors=self.options.get("encoding_errors", "strict"),
1713     storage_options=self.options.get("storage_options", None),
1714 )
1715 assert self.handles is not None
1716 f = self.handles.handle

```

File /opt/conda/lib/python3.11/site-packages/pandas/io/common.py:718, in `get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)`

```

    715     codecs.lookup_error(errors)
    717 # open URLs
--> 718 ioargs = _get_filepath_or_buffer(
    719     path_or_buf,
    720     encoding=encoding,
    721     compression=compression,
    722     mode=mode,
    723     storage_options=storage_options,
    724 )
    726 handle = ioargs.filepath_or_buffer
    727 handles: list[BaseBuffer]

```

File /opt/conda/lib/python3.11/site-packages/pandas/io/common.py:372, in `_get_filepath_or_buffer(filepath_or_buffer, encoding, compression, mode, storage_options)`

```

    370 # assuming storage_options is to be interpreted as headers
    371 req_info = urllib.request.Request(filepath_or_buffer,
    headers=storage_options)
--> 372 with urlopen(req_info) as req:
    373     content_encoding = req.headers.get("Content-Encoding", None)
    374     if content_encoding == "gzip":
    375         # Override compression based on Content-Encoding header

```

File /opt/conda/lib/python3.11/site-packages/pandas/io/common.py:274, in `urlopen(*args, **kwargs)`

```

    268 """
    269 Lazy-import wrapper for stdlib urlopen, as that imports a big chunk of
    270 the stdlib.
    271 """
    272 import urllib.request
--> 274 return urllib.request.urlopen(*args, **kwargs)

```

File /opt/conda/lib/python3.11/urllib/request.py:216, in `urlopen(url, data, timeout, cafile, capath, cadefault, context)`

```

    214 else:
    215     opener = _opener
--> 216 return opener.open(url, data, timeout)

```

File /opt/conda/lib/python3.11/urllib/request.py:525, in OpenerDirector.

```
↪open(self, fullurl, data, timeout)
    523 for processor in self.process_response.get(protocol, []):
    524     meth = getattr(processor, meth_name)
--> 525     response = meth(req, response)
    527 return response
```

File /opt/conda/lib/python3.11/urllib/request.py:634, in HTTPErrorProcessor.

```
↪http_response(self, request, response)
    631 # According to RFC 2616, "2xx" code indicates that the client's
    632 # request was successfully received, understood, and accepted.
    633 if not (200 <= code < 300):
--> 634     response = self.parent.error(
    635         'http', request, response, code, msg, hdrs)
    637 return response
```

File /opt/conda/lib/python3.11/urllib/request.py:563, in OpenerDirector.

```
↪error(self, proto, *args)
    561 if http_err:
    562     args = (dict, 'default', 'http_error_default') + orig_args
--> 563     return self._call_chain(*args)
```

File /opt/conda/lib/python3.11/urllib/request.py:496, in OpenerDirector.

```
↪_call_chain(self, chain, kind, meth_name, *args)
    494 for handler in handlers:
    495     func = getattr(handler, meth_name)
--> 496     result = func(*args)
    497     if result is not None:
    498         return result
```

File /opt/conda/lib/python3.11/urllib/request.py:643, in HTTPDefaultErrorHandler.

```
↪http_error_default(self, req, fp, code, msg, hdrs)
    642 def http_error_default(self, req, fp, code, msg, hdrs):
--> 643     raise HTTPError(req.full_url, code, msg, hdrs, fp)
```

HTTPError: HTTP Error 404: Not Found

[11]: # 8.3.B) fixed with ChatGPT help

```
import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```
[11]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

```
[12]: # 8.3.C)

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

DF.groupby("survived")["age"].describe() # capitalized DF; python is case_
      ↪sensitive
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[12], line 8
      5 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
      ↪titanic.csv'
      6 df = pd.read_csv(url)
----> 8 DF.groupby("survived")["age"].describe()

NameError: name 'DF' is not defined
```

```
[13]: # 8.3.C) fixed with ChatGPT

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
```

```
[13]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

```
[14]: # 8.3.D)

import pandas as pd
```

```
url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
↳csv'
df = pd.read_csv(url # removed ()

df.groupby("survived"["age"].describe() # removed ()
```

```
Cell In[14], line 8
    df.groupby("survived"["age"].describe() # removed ()
    ^
SyntaxError: '(' was never closed
```

```
[15]: # 8.3.D) fixed with ChatGPT and Google search

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
↳csv'
df = pd.read_csv(url) # added )

df.groupby("survived")["age"].describe() # added )
```

```
[15]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

```
[16]: # 8.3.E)

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
↳csv'
df = pd.read_csv(url)

df.group_by("survived")["age"].deescrib() # added _ and miss-spelled describe
```

```
-----
AttributeError                                Traceback (most recent call last)
/tmp/ipykernel_86/2399913520.py in ?()
      4
      5 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
↳titanic.csv'
      6 df = pd.read_csv(url)
      7
```



```

----> 8 df.groupby("survived")["age"].deescrib() # added _ and miss-spelled_
      ↳describe

/opt/conda/lib/python3.11/site-packages/pandas/core/generic.py in ?(self, name)
    6200         and name not in self._accessors
    6201         and self._info_axis.
      ↳_can_hold_identifiers_and_holds_name(name)
    6202     ):
    6203         return self[name]
-> 6204     return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'group_by'

```

[17]: *# 8.3.E) fixed with ChatGPT*

```

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↳csv'
df = pd.read_csv(url)

# Correct method is groupby, not group_by
df.groupby("survived")["age"].describe()
# even though the error message above only listed the _ error, ChatGPT also_
      ↳caught and fixed the spelling mistake!

```

```

[17]:
      count      mean      std  min  25%  50%  75%  max
survived
0      424.0  30.626179  14.172110  1.00  21.0  28.0  39.0  74.0
1      290.0  28.343690  14.950952  0.42  19.0  28.0  36.0  80.0

```

[18]: *# 8.3.F)*

```

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↳csv'
df = pd.read_csv(url)

df.groupby("Sex")["Pclass"].describe()

```

```

-----
KeyError                                Traceback (most recent call last)
Cell In[18], line 8
      5 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
      ↳titanic.csv'
      6 df = pd.read_csv(url)

```

```
----> 8 df.groupby("Sex")["Pclass"].describe()
```

```
File /opt/conda/lib/python3.11/site-packages/pandas/core/frame.py:8869, in
↳ DataFrame.groupby(self, by, axis, level, as_index, sort, group_keys, observed,
↳ dropna)
```

```
    8866 if level is None and by is None:
    8867     raise TypeError("You have to supply one of 'by' and 'level'")
-> 8869 return DataFrameGroupBy(
    8870     obj=self,
    8871     keys=by,
    8872     axis=axis,
    8873     level=level,
    8874     as_index=as_index,
    8875     sort=sort,
    8876     group_keys=group_keys,
    8877     observed=observed,
    8878     dropna=dropna,
    8879 )
```

```
File /opt/conda/lib/python3.11/site-packages/pandas/core/groupby/groupby.py:
↳ 1278, in GroupBy.__init__(self, obj, keys, axis, level, grouper, exclusions,
↳ selection, as_index, sort, group_keys, observed, dropna)
```

```
    1275 self.dropna = dropna
    1277 if grouper is None:
-> 1278     grouper, exclusions, obj = get_grouper(
    1279         obj,
    1280         keys,
    1281         axis=axis,
    1282         level=level,
    1283         sort=sort,
    1284         observed=False if observed is lib.no_default else observed,
    1285         dropna=self.dropna,
    1286     )
    1288 if observed is lib.no_default:
    1289     if any(ping._passed_categorical for ping in grouper.groupings):
```

```
File /opt/conda/lib/python3.11/site-packages/pandas/core/groupby/grouper.py:
```

```
↳ 1009, in get_grouper(obj, key, axis, level, sort, observed, validate, dropna)
    1007     in_axis, level, gpr = False, gpr, None
    1008     else:
-> 1009         raise KeyError(gpr)
    1010 elif isinstance(gpr, Grouper) and gpr.key is not None:
    1011     # Add key to exclusions
    1012     exclusions.add(gpr.key)
```

```
KeyError: 'Sex'
```

```
[19]: # 8.3.F) fixed with ChatGPT

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby("survived")["age"].describe()
# chatGPT didn't catch the capitalization error, but told me to check spelling
```

```
[19]:
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

```
[20]: # 8.3.G)

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)

df.groupby(survived)[age].describe() # removed "" for the stuff in parentheses/
      ↪brackets
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[20], line 8
      5 url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/
      ↪titanic.csv'
      6 df = pd.read_csv(url)
----> 8 df.groupby(survived)[age].describe() # removed "" for the stuff in
      ↪parentheses/brackets

NameError: name 'survived' is not defined
```

```
[21]: # 8.3.G) fixed with ChatGPT

import pandas as pd

url = 'https://raw.githubusercontent.com/mwaskom/seaborn-data/master/titanic.
      ↪csv'
df = pd.read_csv(url)
```

```
result = df.groupby('survived')['age'].describe()
#Chat GPT had trouble catching this until I got very specific
print(result)
```

	count	mean	std	min	25%	50%	75%	max
survived								
0	424.0	30.626179	14.172110	1.00	21.0	28.0	39.0	74.0
1	290.0	28.343690	14.950952	0.42	19.0	28.0	36.0	80.0

Question 8.3 overall impression

For all the errors above, it was much faster to fix them with ChatGPT, and for all but A and D, a Google Search was almost completely useless for solving the problem, as it gave very generic, technical answers, not simply worded, specific ones like the chatbot.

QUESTION 9

Somewhat

ChatGPT links and summaries for Questions 6-8:

Question 6: no chat gpt used

Question 7: <https://chatgpt.com/share/7b87e828-5928-4bfb-a570-cf2588184604>

Summary: “In this conversation, we discussed the difference between using `df.dropna()` and `del df['col']` in Python.

`df.dropna()` is used to remove rows or columns with missing values (NaN), while `del df['col']` is used to delete a specific column by name, regardless of whether it contains missing values. The order of these operations can matter depending on your goals, especially if missing values are present. Additionally, there was an issue with using `print(del df['col'])` in the user’s code, as `del` is a statement, not a function, and should not be used inside `print()`. The solution was to remove the `print()` around `del` and ensure the column name exists in the DataFrame. The corrected code was provided.”

Question 8: <https://chatgpt.com/share/c18b554a-671a-452c-b026-5a8090357f38> (part 1) and <https://chatgpt.com/share/f39dbf07-b7d4-47a0-8d4c-f1462baa5946> (part 2)

Summary: “In this chat, you encountered two issues with your Python code using the Titanic dataset:

KeyError with groupby and describe:

Issue: The error was due to trying to access a column that didn’t exist or was incorrectly referenced.

Solution: Verify that the columns you are referencing are correct and exist in the DataFrame. Use `print(df.columns)` to check column names and ensure they are correctly spelled in your code.

NameError with Column References:

Issue: You tried to reference column names without quotes in the groupby method, causing a NameError. Solution: Column names should be enclosed in quotes. For example, use `df.groupby('survived')['age'].describe()` instead of `df.groupby(survived)[age].describe()`. To avoid these errors:

Always check that column names are correct and exist in the DataFrame. Use strings (quotes) to reference column names in DataFrame methods.”