QuizPractice

October 7, 2024

0.0.1 MIDTERM PRACTICE

QUESTION 1

Write a Pandas program to create and display a one-dimensional array-like object containing an array of data.

[2]:
```
a=[1,4,10,20]
print(a)
```

```
[1, 4, 10, 20]
```

QUESTION 2

I have a dataset with the variable name data. What is a correct syntax to create a Pandas DataFrame?

[3]:
```
import pandas as pd
df = pd.DataFrame(a)
print(df)
```

```
    0
0   1
1   4
2  10
3  20
```

QUESTION 3

What is a correct syntax to return the first row in a Pandas DataFrame with the variable name df?

[4]:
```
df.iloc[0]
```

[4]:
```
0    1
Name: 0, dtype: int64
```

[5]:
```
df.head(1)
```

[5]:
```
   0
0  1
```

QUESTION 4

What is a correct syntax to return both the first row and the second row in a Pandas DataFrame with the variable name df?

```
[25]: #df.iloc[0:2]
      df.iloc[[0,1]]
```

```
[25]:    0
      0  1
      1  4
```

QUESTION 5

What is a correct syntax to return the first 20 rows of a DataFrame with the variable name df?

```
[6]: print(df.head(20))
```

```
    0
0   1
1   4
2  10
3  20
```

QUESTION 6

What is a correct syntax to return the entire DataFrame

```
[7]: df
```

```
[7]:     0
      0   1
      1   4
      2  10
      3  20
```

QUESTION 7

What is the Pandas function for loading JSON files into a DataFrame?

```
[8]: url = ""
     df = pd.read_json(url)
```

```
/tmp/ipykernel_5709/1909173589.py:2: FutureWarning: Passing literal json to
'read_json' is deprecated and will be removed in a future version. To read from
a literal string, wrap it in a 'StringIO' object.
  df = pd.read_json(url)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[8], line 2
      1 url = ""
----> 2 df = pd.read_json(url)

File /opt/conda/lib/python3.11/site-packages/pandas/io/json/_json.py:804, in␣
 ↪read_json(path_or_buf, orient, typ, dtype, convert_axes, convert_dates,␣
 ↪keep_default_dates, precise_float, date_unit, encoding, encoding_errors,␣
 ↪lines, chunksize, compression, nrows, storage_options, dtype_backend, engine)
    802     return json_reader
    803 else:
--> 804     return json_reader.read()

File /opt/conda/lib/python3.11/site-packages/pandas/io/json/_json.py:1014, in␣
 ↪JsonReader.read(self)
   1012         obj = self._get_object_parser(self._combine_lines(data_lines))
   1013 else:
-> 1014     obj = self._get_object_parser(self.data)
   1015 if self.dtype_backend is not lib.no_default:
   1016     return obj.convert_dtypes(
   1017         infer_objects=False, dtype_backend=self.dtype_backend
   1018     )

File /opt/conda/lib/python3.11/site-packages/pandas/io/json/_json.py:1040, in␣
 ↪JsonReader._get_object_parser(self, json)
   1038 obj = None
   1039 if typ == "frame":
-> 1040     obj = FrameParser(json, **kwargs).parse()
   1042 if typ == "series" or obj is None:
   1043     if not isinstance(dtype, bool):

File /opt/conda/lib/python3.11/site-packages/pandas/io/json/_json.py:1173, in␣
 ↪Parser.parse(self)
   1172 def parse(self):
-> 1173     self._parse()
   1175     if self.obj is None:
   1176         return None

File /opt/conda/lib/python3.11/site-packages/pandas/io/json/_json.py:1366, in␣
 ↪FrameParser._parse(self)
   1362 orient = self.orient
   1364 if orient == "columns":
   1365     self.obj = DataFrame(
-> 1366         ujson_loads(json, precise_float=self.precise_float), dtype=None
   1367     )
   1368 elif orient == "split":
```

```
   1369     decoded = {
   1370         str(k): v
   1371         for k, v in ujson_loads(json, precise_float=self.precise_float)
 ↪items()
   1372     }

ValueError: Expected object or value
```

QUESTION 8

What is syntax to load a Python Dictionary called "data" into a Pandas DataFrame?

```
[32]: data={
          'age' : ['18','20'],
      }

      new_df=pd.DataFrame(data)

      new_df
```

```
[32]:   age
      0  18
      1  20
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

QUESTION 9

What is a Pandas method for removing rows that contains empty cells?

```
[ ]: new = df.dropna()
     new
```

QUESTION 10

Write a Pandas program to convert a Panda module Series to Python list.

```
[38]: s=pd.Series([1,2,34])
      type(s)
      s.tolist()
```

```
[38]: [1, 2, 34]
```

QUESTION 11

View the following dataset.

[2]:
```
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)
student_data = pd.DataFrame({
    'school_code': ['s001','s002','s003','s001','s002','s004'],
    'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],
    'name': ['Alberto Franco','Gino Mcneill','Ryan Parkes', 'Eesha Hinton',␣
 ↪'Gino Mcneill', 'David Parkes'],
    'date_Of_Birth ': ['15/05/2002','17/05/2002','16/02/1999','25/09/1998','11/
 ↪05/2002','15/09/1997'],
    'age': [12, 12, 13, 13, 14, 12],
    'height': [173, 192, 186, 167, 151, 159],
    'weight': [35, 32, 33, 30, 31, 32],
    'address': ['street1', 'street2', 'street3', 'street1', 'street2',␣
 ↪'street4']},
    index=['S1', 'S2', 'S3', 'S4', 'S5', 'S6'])

print(student_data)
```

```
   school_code class            name date_Of_Birth   age  height  weight  \
S1        s001     V  Alberto Franco      15/05/2002   12     173      35
S2        s002     V    Gino Mcneill      17/05/2002   12     192      32
S3        s003    VI     Ryan Parkes      16/02/1999   13     186      33
S4        s001    VI    Eesha Hinton      25/09/1998   13     167      30
S5        s002     V    Gino Mcneill      11/05/2002   14     151      31
S6        s004    VI    David Parkes      15/09/1997   12     159      32

    address
S1  street1
S2  street2
S3  street3
S4  street1
S5  street2
S6  street4
```

Write a Pandas program to split the following dataframe into groups based on school code.

[3]:
```
sch_code = student_data.groupby('school_code')
for school_code, group in sch_code:
    print(school_code, group)
```

```
s001    school_code class            name date_Of_Birth   age  height  weight  \
S1        s001     V  Alberto Franco      15/05/2002   12     173      35
S4        s001    VI    Eesha Hinton      25/09/1998   13     167      30
```

```
    address
S1  street1
S4  street1
s002    school_code class          name date_Of_Birth   age  height  weight  \
S2         s002     V  Gino Mcneill     17/05/2002   12     192      32
S5         s002     V  Gino Mcneill     11/05/2002   14     151      31

    address
S2  street2
S5  street2
s003    school_code class         name date_Of_Birth   age  height  weight
address
S3         s003    VI  Ryan Parkes     16/02/1999   13     186      33  street3
s004    school_code class          name date_Of_Birth   age  height  weight  \
S6         s004    VI  David Parkes     15/09/1997   12     159      32

    address
S6  street4
```

Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.

[4]:
```python
# My suggestion
# # Group by school_code and calculate mean, min, and max for the 'age' column
# Using the agg method to include a list of functions

print('\nMean, min, and max value of age for each value of the school:')
grouped_single = student_data.groupby('school_code').agg({'age': ['mean',␣
 ↪'min', 'max']})
print(grouped_single)
```

```
Mean, min, and max value of age for each value of the school:
             age
            mean min max
school_code
s001        12.5  12  13
s002        13.0  12  14
s003        13.0  13  13
s004        12.0  12  12
```

[59]:
```python
sch_code = student_data.groupby('school_code').mean('age')
for school_code, group in sch_code:
    print(school_code, group)
    #print(group['age'].aggregate(mean))

#grouped = pd.df[sch_code]('age').agg(min,mean,max)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[59], line 2
      1 sch_code = student_data.groupby('school_code').mean('age')
----> 2 for school_code, group in sch_code:
      3     print(school_code, group)
      4     #print(group['age'].aggregate(mean))
      5 
      6 #grouped = pd.df[sch_code]('age').agg(min,mean,max)

ValueError: too many values to unpack (expected 2)
```

[56]:
```python
student_data.mean(['age'])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipykernel_2054/1567464985.py in ?()
----> 1 student_data.mean(['age'])

/opt/conda/lib/python3.11/site-packages/pandas/core/frame.py in ?(self, axis,␣
↪skipna, numeric_only, **kwargs)
  11331         skipna: bool = True,
  11332         numeric_only: bool = False,
  11333         **kwargs,
  11334     ):
> 11335         result = super().mean(axis, skipna, numeric_only, **kwargs)
  11336         if isinstance(result, Series):
  11337             result = result.__finalize__(self, method="mean")
  11338         return result

/opt/conda/lib/python3.11/site-packages/pandas/core/generic.py in ?(self, axis,
↪skipna, numeric_only, **kwargs)
  11988         skipna: bool_t = True,
  11989         numeric_only: bool_t = False,
  11990         **kwargs,
  11991     ) -> Series | float:
> 11992         return self._stat_function(

  11993             "mean", nanops.nanmean, axis, skipna, numeric_only, **kwarg
  11994         )

/opt/conda/lib/python3.11/site-packages/pandas/core/generic.py in ?(self, name,
↪func, axis, skipna, numeric_only, **kwargs)
  11945         nv.validate_func(name, (), kwargs)
  11946 
  11947         validate_bool_kwarg(skipna, "skipna", none_allowed=False)
  11948
> 11949         return self._reduce(
  11950             func, name=name, axis=axis, skipna=skipna,␣
 ↪numeric_only=numeric_only
  11951         )

/opt/conda/lib/python3.11/site-packages/pandas/core/frame.py in ?(self, op,␣
 ↪name, axis, skipna, numeric_only, filter_type, **kwds)
  11101         assert filter_type is None or filter_type == "bool", filter_typ
  11102         out_dtype = "bool" if filter_type == "bool" else None
  11103
  11104         if axis is not None:
> 11105             axis = self._get_axis_number(axis)
  11106
  11107         def func(values: np.ndarray):
  11108             # We only use this in the case that operates on self.values

/opt/conda/lib/python3.11/site-packages/pandas/core/generic.py in ?(cls, axis)
    549     @classmethod
    550     def _get_axis_number(cls, axis: Axis) -> AxisInt:
    551         try:
    552             return cls._AXIS_TO_AXIS_NUMBER[axis]
--> 553         except KeyError:
    554             raise ValueError(f"No axis named {axis} for object type {cl
 ↪__name__}")

TypeError: unhashable type: 'list'
```

QUESTION 12

View the following dataset

[5]:
```python
import pandas as pd
pd.set_option('display.max_rows', None)
#pd.set_option('display.max_columns', None)

df = pd.DataFrame({
'ord_no':
 ↪[70001,70009,70002,70004,70007,70005,70008,70010,70003,70012,70011,70013],
'purch_amt':[150.5,270.65,65.26,110.5,948.5,2400.6,5760,1983.43,2480.4,250.45,␣
 ↪75.29,3045.6],
'ord_date':␣
 ↪['2012-10-05','2012-09-10','2012-10-05','2012-08-17','2012-09-10','2012-07-27','2012-09-10'
'customer_id':[3001,3001,3005,3001,3005,3001,3005,3001,3005,3001,3005,3005],
'salesman_id': [5002,5005,5001,5003,5002,5001,5001,5006,5003,5002,5007,5001]})
print("Original Orders DataFrame:")
print(df)
```

```
Original Orders DataFrame:
    ord_no  purch_amt    ord_date  customer_id  salesman_id
0    70001     150.50  2012-10-05         3001         5002
1    70009     270.65  2012-09-10         3001         5005
2    70002      65.26  2012-10-05         3005         5001
3    70004     110.50  2012-08-17         3001         5003
4    70007     948.50  2012-09-10         3005         5002
5    70005    2400.60  2012-07-27         3001         5001
6    70008    5760.00  2012-09-10         3005         5001
7    70010    1983.43  2012-10-10         3001         5006
8    70003    2480.40  2012-10-10         3005         5003
9    70012     250.45  2012-06-27         3001         5002
10   70011      75.29  2012-08-17         3005         5007
11   70013    3045.60  2012-04-25         3005         5001
```

Write a Pandas program to split a dataset to group by two columns and then sort the aggregated results within the groups. Group on 'customer_id', 'salesman_id' and then sort sum of purch_amt within the groups. Use nlargest() to obtain the largest 5 values.

[7]:
```
# I am grouping by customer_id and salesman_id, sum the purch_amt within each␣
↪group, and then sort the groups by the sum to show the largest five values.

# Grouping by 'customer_id' and 'salesman_id' and summing 'purch_amt'
df_agg = df.groupby(['customer_id', 'salesman_id']).agg({'purch_amt': 'sum'})
# Extracting the 'purch_amt' column
result = df_agg['purch_amt']

# Printing the top 5 highest sums
print("\nGroup on 'customer_id', 'salesman_id' and then sort sum of purch_amt␣
↪within the groups:")
print(result.nlargest(5))  # specify how many of the largest you want
```

```
Group on 'customer_id', 'salesman_id' and then sort sum of purch_amt within the
groups:
customer_id  salesman_id
3005         5001           8870.86
             5003           2480.40
3001         5001           2400.60
             5006           1983.43
3005         5002            948.50
Name: purch_amt, dtype: float64
```

QUESTION 13

Write a Pandas program to join the two given dataframes along rows and assign all data.

Write a Pandas program to join the two given dataframes along columns and assign all data.

[8]:
```python
import pandas as pd

student_data1 = pd.DataFrame({
        'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
         'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed␣
 ↪Bernal', 'Kwame Morin'],
        'points': [200, 210, 190, 222, 199]})

student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser␣
 ↪William', 'Madeeha Preston'],
        'points': [201, 200, 198, 219, 201]})
```

[9]:
```python
#  Join the two given dataframes along rows and assign all data

print("Original DataFrames:")
print(student_data1)
print("-------------------------------------")
print(student_data2)
print("\nJoin the said two dataframes along rows:")
result_data = pd.concat([student_data1, student_data2])
print(result_data)
```

```
Original DataFrames:
  student_id              name  points
0         S1  Danniella Fenton     200
1         S2      Ryder Storey     210
2         S3      Bryce Jensen     190
3         S4         Ed Bernal     222
4         S5       Kwame Morin     199
-------------------------------------
  student_id              name  points
0         S4  Scarlette Fisher     201
1         S5  Carla Williamson     200
2         S6       Dante Morse     198
3         S7    Kaiser William     219
4         S8   Madeeha Preston     201

Join the said two dataframes along rows:
  student_id              name  points
0         S1  Danniella Fenton     200
```

```
1          S2      Ryder Storey     210
2          S3      Bryce Jensen     190
3          S4         Ed Bernal     222
4          S5       Kwame Morin     199
0          S4  Scarlette Fisher     201
1          S5  Carla Williamson     200
2          S6       Dante Morse     198
3          S7    Kaiser William     219
4          S8   Madeeha Preston     201
```

[10]:
```
# Join the two given dataframes along columns and assign all data.

print("Original DataFrames:")
print(student_data1)
print("-------------------------------------")
print(student_data2)
print("\nJoin the said two dataframes along columns:")
result_data2 = pd.concat([student_data1, student_data2], axis = 1)
print(result_data2)
```

```
Original DataFrames:
  student_id              name  points
0         S1  Danniella Fenton     200
1         S2      Ryder Storey     210
2         S3      Bryce Jensen     190
3         S4         Ed Bernal     222
4         S5       Kwame Morin     199
-------------------------------------
  student_id              name  points
0         S4  Scarlette Fisher     201
1         S5  Carla Williamson     200
2         S6       Dante Morse     198
3         S7    Kaiser William     219
4         S8   Madeeha Preston     201

Join the said two dataframes along columns:
  student_id              name  points student_id              name  points
0         S1  Danniella Fenton     200         S4  Scarlette Fisher     201
1         S2      Ryder Storey     210         S5  Carla Williamson     200
2         S3      Bryce Jensen     190         S6       Dante Morse     198
3         S4         Ed Bernal     222         S7    Kaiser William     219
4         S5       Kwame Morin     199         S8   Madeeha Preston     201
```

QUESTION 14

Write a Pandas program to append rows in s6 to an existing DataFrame and display the combined data.

```
[13]: import pandas as pd
      student_data1 = pd.DataFrame({
              'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
               'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed␣
       ↪Bernal', 'Kwame Morin'],
              'points': [200, 210, 190, 222, 199]})

      s6 = pd.Series(['S6', 'Scarlette Fisher', 205], index=['student_id', 'name',␣
       ↪'points'])
      s6
```

```
[13]: student_id                 S6
      name           Scarlette Fisher
      points                    205
      dtype: object
```

```
[ ]: print("Original DataFrames:")
     print(student_data1)
     print("\nNew Row(s)")
     print(s6)

     combined_data = pd.concat([student_data1, s6.to_frame().T], ignore_index=True)
     # The s6.to_frame().T converts the Series s6 into a DataFrame (transposing it␣
      ↪to match the DataFrame's structure).
     print("\nCombined Data:")
     print(combined_data)


     # pd.concat() combines two dataframes, student_data1 and s6.to_frame().T, and␣
      ↪specifying ignore_index=True to reset the index.
     # s6.to_frame().T: s6 is a Series, and you are converting it to a DataFrame␣
      ↪using .to_frame(), followed by .T to transpose it, turning the Series into a␣
      ↪single-row DataFrame.
     # ignore_index=True: ensures that the resulting DataFrame has a continuous␣
      ↪index, ignoring the original indices from both student_data1 and s6
```

QUESTION 15

Write a Pandas program to join the two dataframes with matching records from both sides where available.

```
[14]: import pandas as pd
      student_data1 = pd.DataFrame({
              'student_id': ['S1', 'S2', 'S3', 'S4', 'S5'],
               'name': ['Danniella Fenton', 'Ryder Storey', 'Bryce Jensen', 'Ed␣
       ↪Bernal', 'Kwame Morin'],
              'marks': [200, 210, 190, 222, 199]})
```

```
student_data2 = pd.DataFrame({
        'student_id': ['S4', 'S5', 'S6', 'S7', 'S8'],
        'name': ['Scarlette Fisher', 'Carla Williamson', 'Dante Morse', 'Kaiser␣
 ↪William', 'Madeeha Preston'],
        'marks': [201, 200, 198, 219, 201]})
```

[15]:
```
print("Original DataFrames:")
print(" Dataset 1", student_data1)
print( " Dataset 2",student_data2)

# # Merging using an outer join on 'student_id'
merged_data = pd.merge(student_data1, student_data2, on='student_id',␣
 ↪how='outer')
print("Merged data (outer join):")
print(merged_data)
```

```
Original DataFrames:
 Dataset 1   student_id             name  marks
0          S1  Danniella Fenton    200
1          S2      Ryder Storey    210
2          S3      Bryce Jensen    190
3          S4         Ed Bernal    222
4          S5       Kwame Morin    199
 Dataset 2   student_id             name  marks
0          S4  Scarlette Fisher    201
1          S5  Carla Williamson    200
2          S6       Dante Morse    198
3          S7    Kaiser William    219
4          S8   Madeeha Preston    201
Merged data (outer join):
  student_id            name_x  marks_x            name_y  marks_y
0          S1  Danniella Fenton    200.0               NaN      NaN
1          S2      Ryder Storey    210.0               NaN      NaN
2          S3      Bryce Jensen    190.0               NaN      NaN
3          S4         Ed Bernal    222.0  Scarlette Fisher    201.0
4          S5       Kwame Morin    199.0  Carla Williamson    200.0
5          S6               NaN      NaN       Dante Morse    198.0
6          S7               NaN      NaN    Kaiser William    219.0
7          S8               NaN      NaN   Madeeha Preston    201.0
```

QUESTION 16

Write a Pandas program to merge two given datasets using multiple join keys.

[17]:
```
import pandas as pd
```

```
data1 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'P': ['P0', 'P1', 'P2', 'P3'],
                     'Q': ['Q0', 'Q1', 'Q2', 'Q3']})


data2 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'R': ['R0', 'R1', 'R2', 'R3'],
                      'S': ['S0', 'S1', 'S2', 'S3']})
```

[18]:

```
#merges data1 and data2 based on the matching values in both key1 and key2␣
 ↪columns. Only the rows where both key1 and key2 match in both DataFrames␣
 ↪will appear in the result.

merged_data = pd.merge(data1, data2, on=['key1', 'key2'])
print(merged_data)


# to include unmatched rows (e.g., rows that appear in one DataFrame but not␣
 ↪the other), use how='outer' for an outer join, or how='left' or how='right'␣
 ↪for left or right joins.
```

```
  key1 key2   P   Q   R   S
0   K0   K0  P0  Q0  R0  S0
1   K1   K0  P2  Q2  R1  S1
2   K1   K0  P2  Q2  R2  S2
```

[ ]: