# CSP 554 – Assignment #7

Name: Adarsh Mathad Vijayakumar
CWID: A20424847
Email ID: avijayakumar@hawk.iit.edu

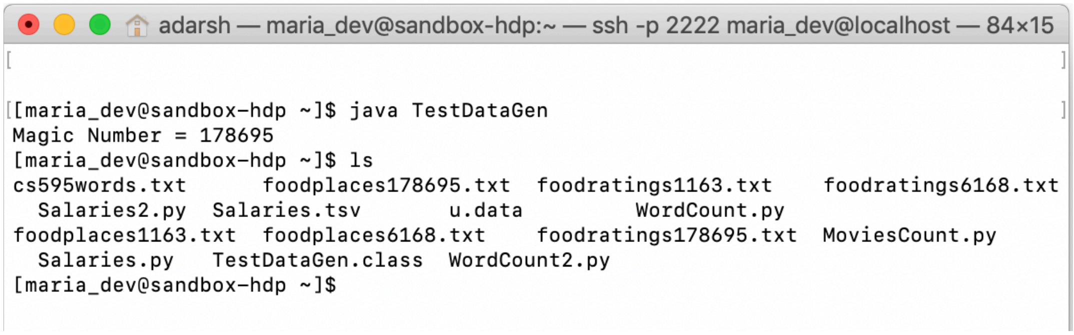
## Exercise 1)

### Step A

Use the TestDataGen program from previous assignments to generate new data files

Command: java TestDataGen

Output: Magic Number = 178695


adarsh — maria_dev@sandbox-hdp:~ — ssh -p 2222 maria_dev@localhost — 84×15

```
[maria_dev@sandbox-hdp ~]$ java TestDataGen
Magic Number = 178695
[maria_dev@sandbox-hdp ~]$ ls
cs595words.txt      foodplaces178695.txt  foodratings1163.txt    foodratings6168.txt
  Salaries2.py  Salaries.tsv       u.data         WordCount.py
foodplaces1163.txt  foodplaces6168.txt    foodratings178695.txt  MoviesCount.py
  Salaries.py   TestDataGen.class  WordCount2.py
[maria_dev@sandbox-hdp ~]$
```

### Step B

Load the 'foodratings' file as a 'csv' file into a DataFrame called foodratings.

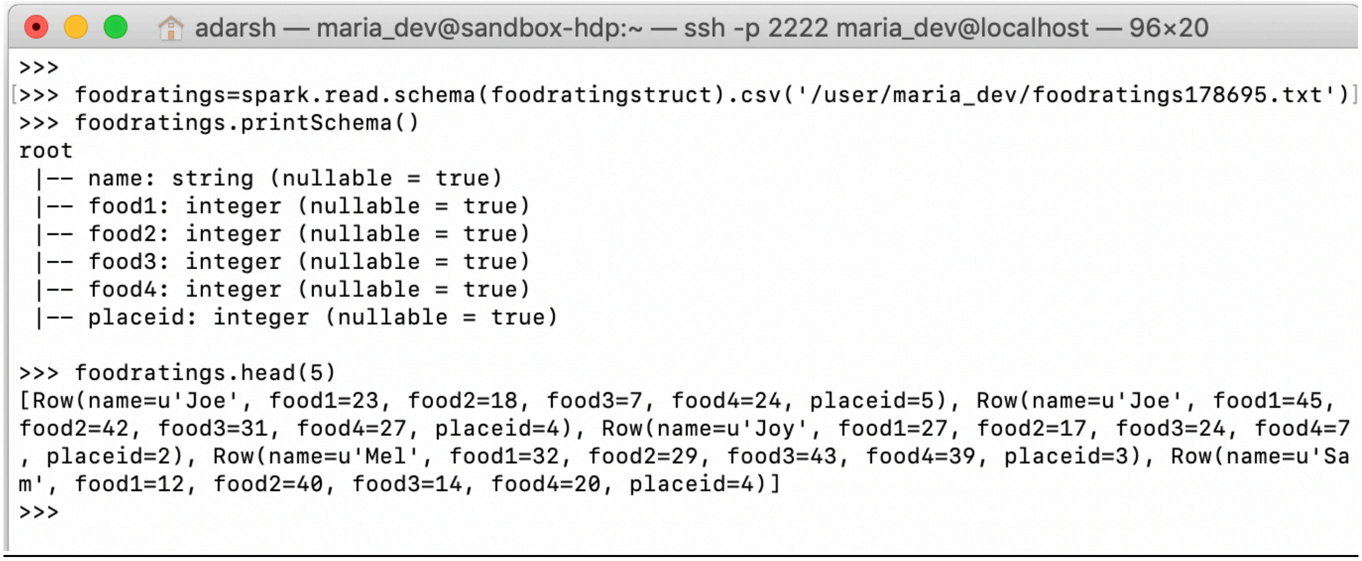
As the results of this exercise provide the magic number, the code you execute and screen shots of the following commands:

foodratings.printSchema()
foodratings.head(5)

Commands:

```
from pyspark.sql.types import *
foodratingstruct = StructType(
        [
                StructField("name", StringType(), True),
                StructField("food1",IntegerType(), True),
                StructField("food2",IntegerType(), True),
                StructField("food3",IntegerType(), True),
                StructField("food4",IntegerType(), True),
                StructField("placeid",IntegerType(), True)
        ]
)

foodratings=spark.read.schema(foodratingstruct).csv('/user/maria_dev/foodratings178695.txt')
foodratings.printSchema()
foodratings.head(5)
```

Output Screenshot:


adarsh — maria_dev@sandbox-hdp:~ — ssh -p 2222 maria_dev@localhost — 96×20

```
>>>
[>>> foodratings=spark.read.schema(foodratingstruct).csv('/user/maria_dev/foodratings178695.txt')]
>>> foodratings.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings.head(5)
[Row(name=u'Joe', food1=23, food2=18, food3=7, food4=24, placeid=5), Row(name=u'Joe', food1=45,
food2=42, food3=31, food4=27, placeid=4), Row(name=u'Joy', food1=27, food2=17, food3=24, food4=7
, placeid=2), Row(name=u'Mel', food1=32, food2=29, food3=43, food4=39, placeid=3), Row(name=u'Sa
m', food1=12, food2=40, food3=14, food4=20, placeid=4)]
>>>
```

Exercise 2)

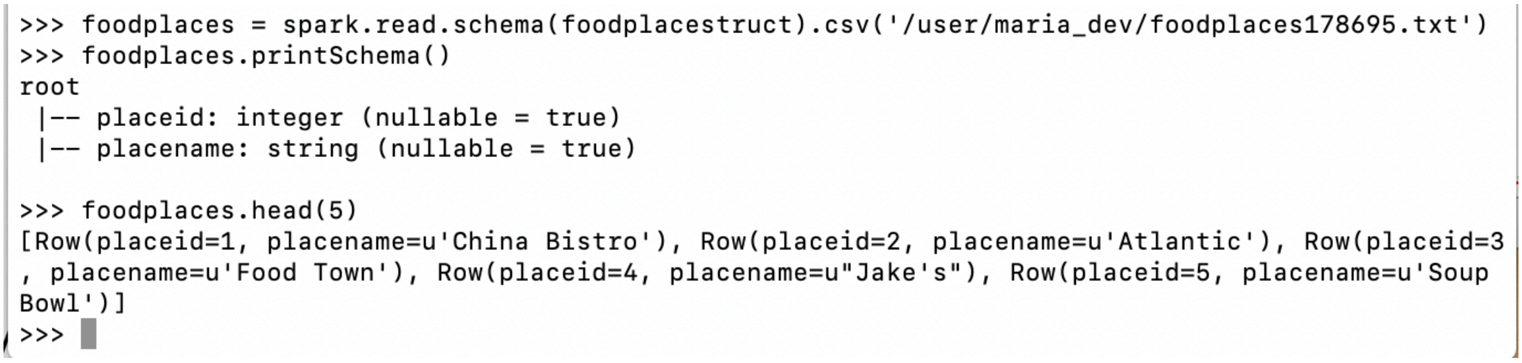
Load the ‘foodplaces’ file as a ‘csv’ file into a DataFrame called foodplaces.

As the results of this exercise provide the code you execute and screen shots of the following commands:

    foodratings.printSchema()
    foodratings.head(5)

Commands:

```
from pyspark.sql.types import *
foodplacestruct = StructType(
        [
                StructField("placeid", IntegerType(), True),
                StructField("placename", StringType(), True)
        ]
)

foodplaces = spark.read.schema(foodplacestruct).csv('/user/maria_dev/foodplaces178695.txt')
foodplaces.printSchema()
foodplaces.head(5)
```



```
>>> foodplaces = spark.read.schema(foodplacestruct).csv('/user/maria_dev/foodplaces178695.txt')
>>> foodplaces.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces.head(5)
[Row(placeid=1, placename=u'China Bistro'), Row(placeid=2, placename=u'Atlantic'), Row(placeid=3
, placename=u'Food Town'), Row(placeid=4, placename=u"Jake's"), Row(placeid=5, placename=u'Soup
Bowl')]
>>> 
```

## Exercise 3)

### Step A

Register the DataFrames created in exercise 1 and 2 as tables called “foodratingsT” and “foodplacesT”

Commands:

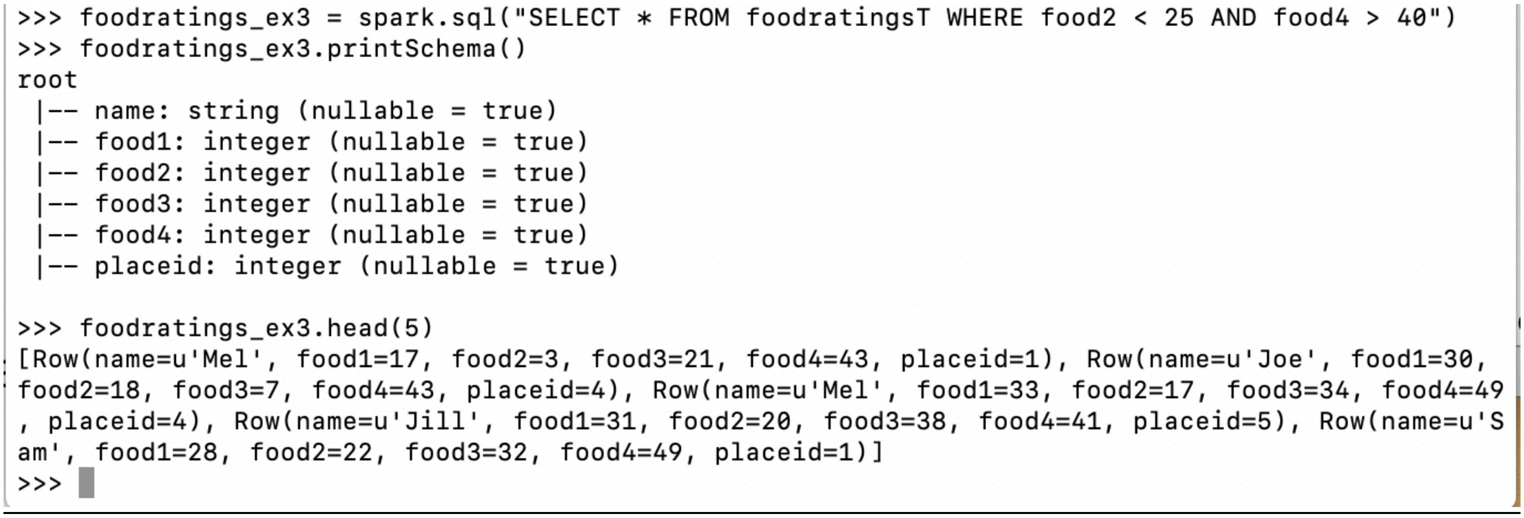
```
from pyspark.sql.types import *
foodratings.createOrReplaceTempView("foodratingsT")
foodplaces.createOrReplaceTempView("foodplacesT")
```

### Step B

Use a SQL query on the table “foodratingsT” to create a new DataFrame called foodratings_ex3 holding records which meet the following condition: food2 < 25 and food4 > 40

Commands:

```
foodratings_ex3 = spark.sql("SELECT * FROM foodratingsT WHERE food2 < 25 AND food4 > 40")
foodratings_ex3.printSchema()
foodratings_ex3.head(5)
```



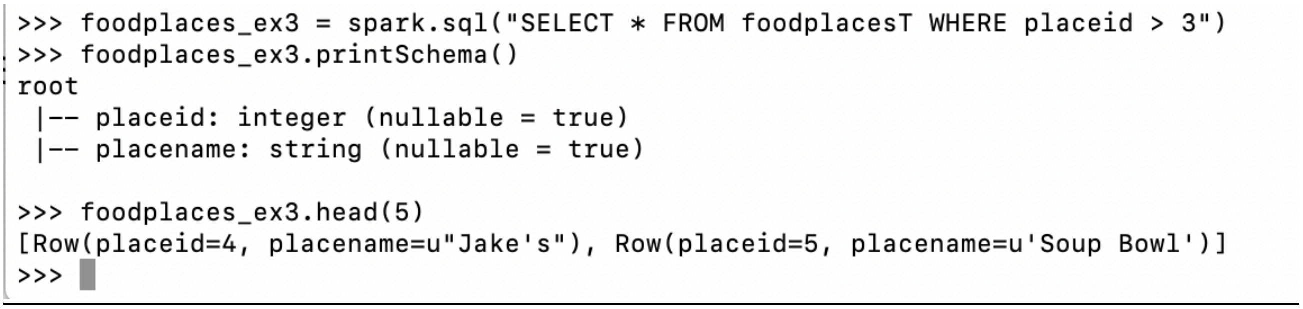
```
>>> foodratings_ex3 = spark.sql("SELECT * FROM foodratingsT WHERE food2 < 25 AND food4 > 40")
>>> foodratings_ex3.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex3.head(5)
[Row(name=u'Mel', food1=17, food2=3, food3=21, food4=43, placeid=1), Row(name=u'Joe', food1=30,
food2=18, food3=7, food4=43, placeid=4), Row(name=u'Mel', food1=33, food2=17, food3=34, food4=49
, placeid=4), Row(name=u'Jill', food1=31, food2=20, food3=38, food4=41, placeid=5), Row(name=u'S
am', food1=28, food2=22, food3=32, food4=49, placeid=1)]
>>> 
```

### Step C

Use a SQL query on the table “foodplacesT” to create a new DataFrame called foodplaces_ex3 holding records which meet the following condition: placeid > 3

Commands:

```
foodplaces_ex3 = spark.sql("SELECT * FROM foodplacesT WHERE placeid > 3")
foodplaces_ex3.printSchema()
foodplaces_ex3.head(5)
```



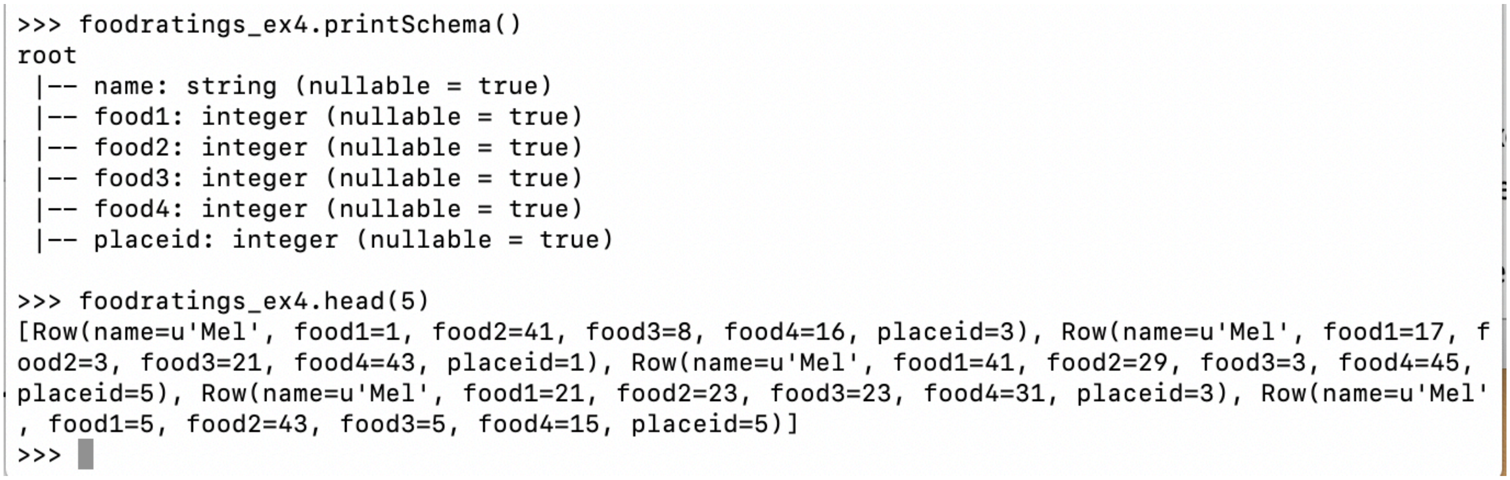
```
>>> foodplaces_ex3 = spark.sql("SELECT * FROM foodplacesT WHERE placeid > 3")
>>> foodplaces_ex3.printSchema()
root
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> foodplaces_ex3.head(5)
[Row(placeid=4, placename=u"Jake's"), Row(placeid=5, placename=u'Soup Bowl')]
>>> 
```

### Exercise 4)

Use an operation (not a SQL query) on the DataFrame ‘foodratings’ create in exercise 1 to create a new DataFrame called foodratings_ex4 that includes only those records (rows) where the ‘name’ field is “Mel” and food3 < 25.

Commands:

```
foodratings_ex4 = foodratings.filter(foodratings.name == "Mel").filter(foodratings.food3 < 25)
foodratings_ex4.printSchema()
foodratings_ex4.head(5)
```



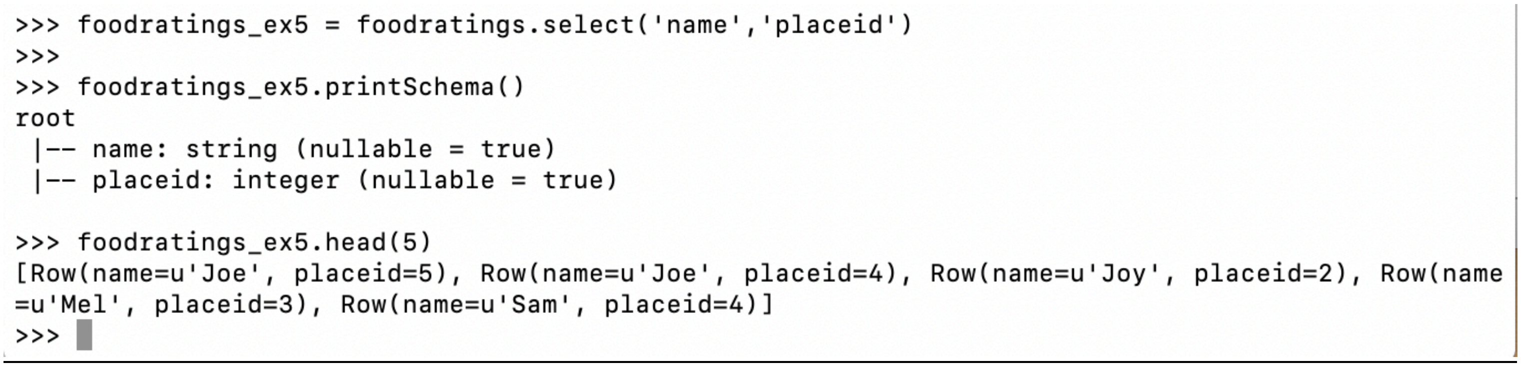
```
>>> foodratings_ex4.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex4.head(5)
[Row(name=u'Mel', food1=1, food2=41, food3=8, food4=16, placeid=3), Row(name=u'Mel', food1=17, f
ood2=3, food3=21, food4=43, placeid=1), Row(name=u'Mel', food1=41, food2=29, food3=3, food4=45,
placeid=5), Row(name=u'Mel', food1=21, food2=23, food3=23, food4=31, placeid=3), Row(name=u'Mel'
, food1=5, food2=43, food3=5, food4=15, placeid=5)]
>>> 
```

### Exercise 5)

Use an operation (not a SQL query) on the DataFrame ‘foodratings’ create in exercise 1 to create a new DataFrame called foodratings_ex5 that includes only the columns (fields) ‘name’ and ‘placeid’

Commands:

```
foodratings_ex5 = foodratings.select('name','placeid')
foodratings_ex5.printSchema()
foodratings_ex5.head(5)
```



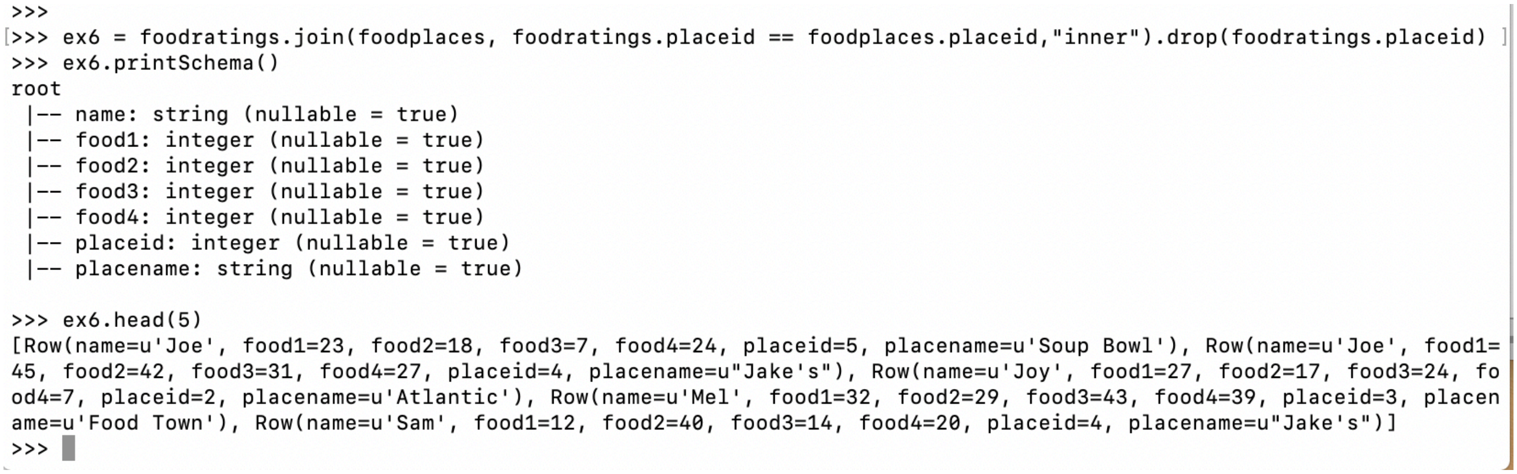
```
>>> foodratings_ex5 = foodratings.select('name','placeid')
>>>
>>> foodratings_ex5.printSchema()
root
 |-- name: string (nullable = true)
 |-- placeid: integer (nullable = true)

>>> foodratings_ex5.head(5)
[Row(name=u'Joe', placeid=5), Row(name=u'Joe', placeid=4), Row(name=u'Joy', placeid=2), Row(name
=u'Mel', placeid=3), Row(name=u'Sam', placeid=4)]
>>> 
```

Exercise 6)

Use an operation on the DataFrame called ex6 which is the inner join, on placeid, of the DataFrames ‘foodratings; and ‘foodplaces’ created in exercises 1 and 2

Commands:

```
ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid,"inner").drop(foodratings.placeid)
ex6.printSchema()
ex6.head(5)
```



```
>>>
[>>> ex6 = foodratings.join(foodplaces, foodratings.placeid == foodplaces.placeid,"inner").drop(foodratings.placeid) ]
>>> ex6.printSchema()
root
 |-- name: string (nullable = true)
 |-- food1: integer (nullable = true)
 |-- food2: integer (nullable = true)
 |-- food3: integer (nullable = true)
 |-- food4: integer (nullable = true)
 |-- placeid: integer (nullable = true)
 |-- placename: string (nullable = true)

>>> ex6.head(5)
[Row(name=u'Joe', food1=23, food2=18, food3=7, food4=24, placeid=5, placename=u'Soup Bowl'), Row(name=u'Joe', food1=
45, food2=42, food3=31, food4=27, placeid=4, placename=u"Jake's"), Row(name=u'Joy', food1=27, food2=17, food3=24, fo
od4=7, placeid=2, placename=u'Atlantic'), Row(name=u'Mel', food1=32, food2=29, food3=43, food4=39, placeid=3, placen
ame=u'Food Town'), Row(name=u'Sam', food1=12, food2=40, food3=14, food4=20, placeid=4, placename=u"Jake's")]
>>> 
```