# Манипулирование данными с помощью Pandas

## Создание и визуализация фреймов данных

### Какой размер авокадо самый популярный?

Авокадо становятся все более популярными и вкусными в гуакамоле и на хлебе. Совет по авокадо Hass отслеживает предложение и спрос на авокадо по всему США, включая продажи трех разных размеров авокадо. В этом упражнении вы будете использовать столбчатую диаграмму, чтобы выяснить, какой размер самый популярный.

Столбчатые диаграммы отлично подходят для выявления связей между категориальными (размер) и числовыми (количество проданных) переменными, но часто вам придется сначала обработать данные, чтобы получить необходимые числа для построения графика.

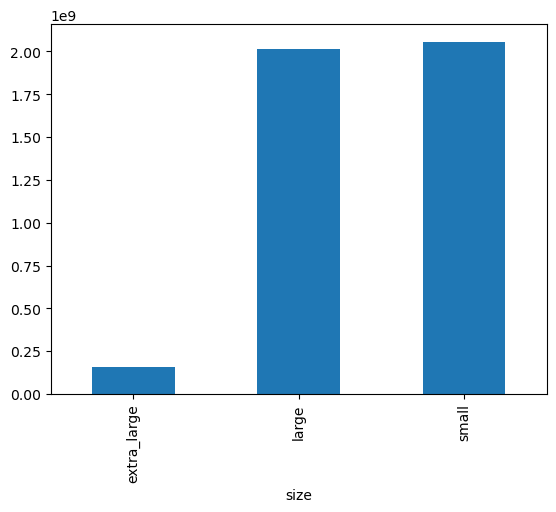
Библиотека pandas уже импортирована как pd, и есть доступ к данным об авокадо.

#### Инструкции

* Выведите заголовок набора данных об авокадо. Какие столбцы доступны?
* Для каждой группы авокадо определенного размера рассчитайте общее количество проданных, сохраняя как nb\_sold\_by\_size.
* Создайте столбчатую диаграмму количества проданных авокадо по размерам.
* Покажите диаграмму.

import pandas as pd  
  
# Импорт matplotlib.pyplot с псевдонимом plt  
import matplotlib.pyplot as plt  
  
avocados = pd.read\_csv('datasets/avocado.csv')  
  
# Посмотреть первые несколько строк данных  
print(\_\_\_\_)  
  
# Получить общее количество проданных авокадо каждого размера  
nb\_sold\_by\_size = \_\_\_\_  
  
# Создать столбчатую диаграмму количества проданных авокадо по размерам  
\_\_\_\_\_\_  
  
# Показать диаграмму  
\_\_\_\_\_\_

date type year avg\_price size nb\_sold  
0 2015-12-27 conventional 2015 0.95 small 9626901.09  
1 2015-12-20 conventional 2015 0.98 small 8710021.76  
2 2015-12-13 conventional 2015 0.93 small 9855053.66  
3 2015-12-06 conventional 2015 0.89 small 9405464.36  
4 2015-11-29 conventional 2015 0.99 small 8094803.56



### Изменения продаж с течением времени

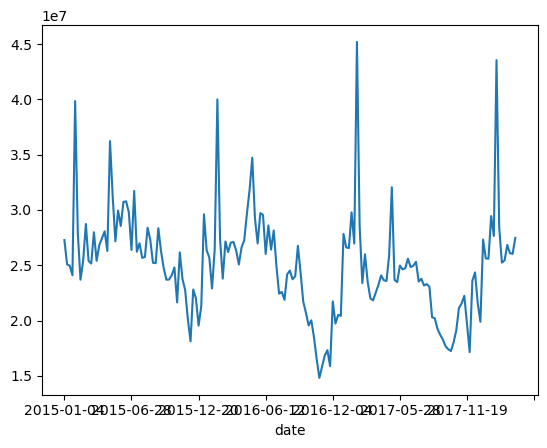
Линейные графики предназначены для визуализации связи между двумя числовыми переменными, где каждое значение данных соединено с следующим. Они особенно полезны для визуализации изменения числа во времени, поскольку каждая временная точка естественным образом связана с следующей. В этом упражнении вы визуализируете изменение продаж авокадо за три года.

Библиотека pandas уже импортирована как pd, и есть доступ к данным об авокадо.

#### Инструкции

* Получите общее количество проданных авокадо по каждой дате. В DataFrame есть две строки для каждой даты: одна для органических и одна для обычных. Сохраните это как nb\_sold\_by\_date.
* Создайте линейный график количества проданных авокадо.
* Покажите график.

# Получить общее количество проданных авокадо по каждой дате  
nb\_sold\_by\_date = \_\_\_\_  
  
# Создать линейный график количества проданных авокадо по датам  
\_\_\_\_\_  
  
# Показать график  
\_\_\_\_\_



### Спрос и предложение на авокадо

Диаграммы рассеяния идеально подходят для визуализации взаимосвязи числовых переменных. В этом упражнении вы сравните количество проданных авокадо со средней ценой и посмотрите, есть ли между ними какая-то связь. Если они связаны, возможно, вы сможете использовать одно число для предсказания другого.

matplotlib.pyplot импортирован как plt, pandas импортирован как pd, и есть доступ к данным об авокадо.

#### Инструкции

* Создайте диаграмму рассеяния с количеством проданных авокадо по оси x и средней ценой по оси y. Назовите её "Количество проданных авокадо vs. средняя цена".
* Покажите график.

# Диаграмма рассеяния с avg\_price против nb\_sold с заголовком  
\_\_\_\_.\_\_\_\_  
  
# Показать график  
\_\_\_\_



### Цена обычного и органического авокадо

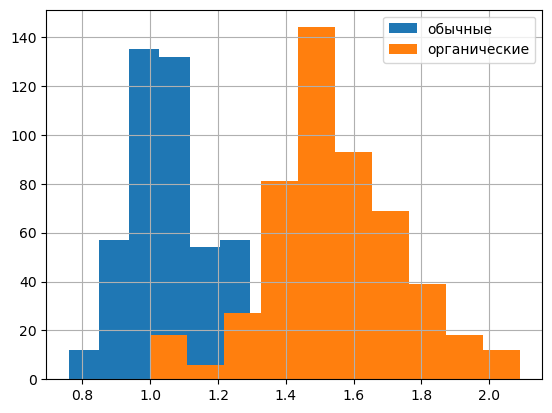
Создание нескольких графиков для разных подмножеств данных позволяет сравнивать группы. В этом упражнении вы создадите несколько гистограмм для сравнения цен на обычные и органические авокадо.

matplotlib.pyplot импортирован как plt, pandas импортирован как pd.

#### Инструкции 1/3

* Выделите подмножество авокадо для типа "обычные" и столбца со средней ценой. Создайте гистограмму.
* Создайте гистограмму avg\_price для органических авокадо.
* Добавьте легенду к вашему графику с названиями "обычные" и "органические".
* Покажите ваш график.

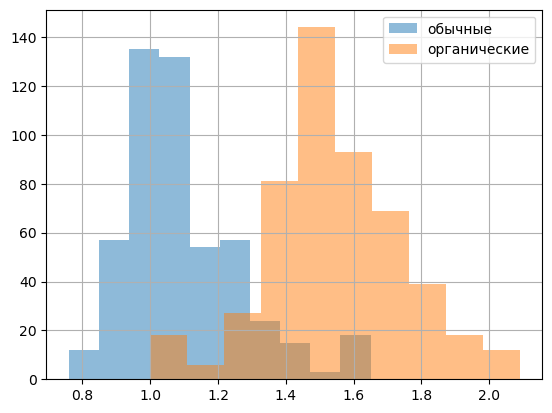
# Измените прозрачность гистограммы на 0.5  
avocados[\_\_\_\_][\_\_\_\_].\_\_\_\_  
  
# Измените прозрачность гистограммы на 0.5  
avocados[\_\_\_\_][\_\_\_\_].\_\_\_\_  
  
# Добавьте легенду  
plt.legend(\_\_\_\_)  
  
# Покажите график  
\_\_\_\_



#### Инструкции 2/3

* Измените ваш код так, чтобы установить прозрачность обеих гистограмм на 0.5, чтобы увидеть, насколько они перекрываются.

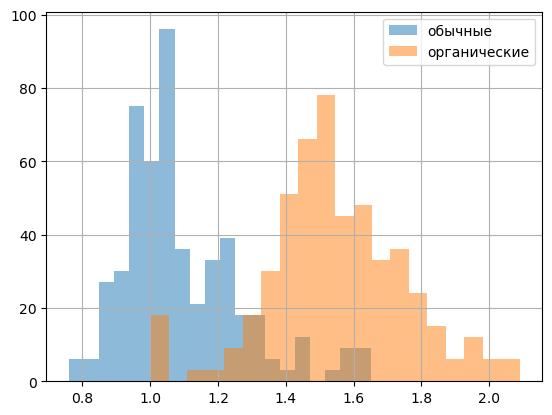
# Измените прозрачность гистограммы на 0.5  
avocados[avocados["type"] == "conventional"]["avg\_price"].hist()  
  
# Измените прозрачность гистограммы на 0.5  
avocados[avocados["type"] == "organic"]["avg\_price"].hist()  
  
# Добавьте легенду  
plt.legend(["обычные", "органические"])  
  
# Покажите график  
plt.show()



#### Инструкции 3/3

* Измените свой код, чтобы использовать 20 ячеек в обеих гистограммах.

# Измените количество интервалов (bins) на 20  
avocados[avocados["type"] == "conventional"]["avg\_price"].hist(alpha=0.5, )  
  
# Измените количество интервалов (bins) на 20  
avocados[avocados["type"] == "organic"]["avg\_price"].hist(alpha=0.5, )  
  
# Добавьте легенду  
plt.legend(["обычные", "органические"])  
  
# Покажите график  
plt.show()



### Поиск пропущенных значений

Пропущенные значения повсюду, и вы не хотите, чтобы они мешали вашей работе. Некоторые функции по умолчанию игнорируют отсутствующие данные, но это не всегда то поведение, которое вам может понадобиться. Некоторые функции вообще не могут обрабатывать отсутствующие значения, поэтому эти значения нужно обработать перед тем, как использовать их. Если вы не знаете, где находятся ваши пропущенные значения или существуют ли они, вы можете допустить ошибки в анализе. В этом упражнении вы определите, есть ли пропущенные значения в наборе данных и, если они есть, сколько их.

Библиотека pandas импортирована как pd, и доступен avocados\_2016, поднабор авокадо, содержащий только продажи за 2016 год.

#### Инструкции

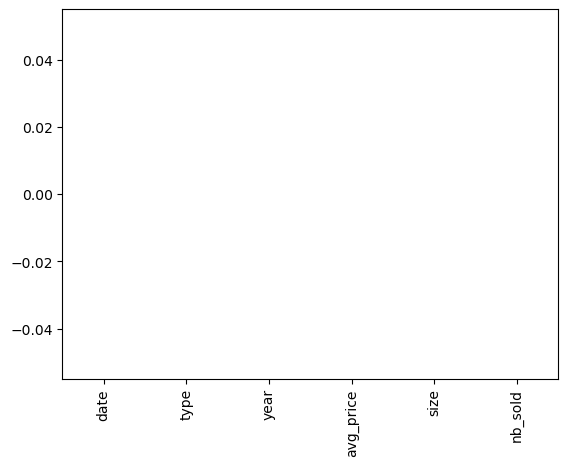
* Выведите DataFrame, который показывает, отсутствует ли каждое значение в avocados\_2016 или нет.
* Выведите сводку, которая показывает, отсутствует ли какое-либо значение в каждом столбце или нет.
* Создайте столбчатую диаграмму общего количества пропущенных значений в каждом столбце.

import pandas as pd  
import numpy as np  
  
# Предполагая, что avocados - ваш DataFrame  
avocados\_2016 = avocados[avocados['year'] == 2016].copy() # Создание копии для избежания предупреждений  
  
# Получите 10 случайных индексов в пределах отфильтрованного DataFrame  
random\_indices = np.random.choice(avocados\_2016.index, size=10, replace=False)  
  
# Установите значения 'date' на этих индексах равными NaN  
avocados\_2016.loc[random\_indices, ['date','avg\_price','nb\_sold']] = np.nan  
  
# Проверьте наличие пропущенных значений в столбце 'date'  
missing\_values = avocados\_2016['date'].isnull().sum()  
print("Пропущенные значения в столбце 'date':", missing\_values)

Пропущенные значения в столбце 'date': 10

# Предполагая, что df - ваш DataFrame  
# Преобразуйте 'date' в формат datetime, если это еще не сделано  
avocados['date'] = pd.to\_datetime(avocados['date'])  
  
# Извлеките год из 'date'  
avocados['year'] = avocados['date'].dt.year  
  
# Отфильтруйте данные для года 2016  
avocados\_2016 = avocados[avocados['year'] == 2016]  
  
print(avocados\_2016)  
# Проверьте отдельные значения на наличие пропущенных значений  
print()  
  
# Проверьте каждый столбец на наличие пропущенных значений  
print()  
  
# Столбчатая диаграмма пропущенных значений по переменным  
\_\_\_\_\_  
  
# Показать график  
plt.show()

date type year avg\_price size nb\_sold  
52 2016-12-25 conventional 2016 1.00 small 9255125.20  
53 2016-12-18 conventional 2016 0.96 small 9394065.91  
54 2016-12-11 conventional 2016 0.98 small 9009996.11  
55 2016-12-04 conventional 2016 1.00 small 11043350.90  
56 2016-11-27 conventional 2016 1.21 small 7891487.94  
.. ... ... ... ... ... ...  
944 2016-01-31 organic 2016 1.43 extra\_large 5515.91  
945 2016-01-24 organic 2016 1.41 extra\_large 12933.97  
946 2016-01-17 organic 2016 1.44 extra\_large 13106.29  
947 2016-01-10 organic 2016 1.35 extra\_large 8771.79  
948 2016-01-03 organic 2016 1.43 extra\_large 7749.40  
  
[312 rows x 6 columns]  
 date type year avg\_price size nb\_sold  
52 False False False False False False  
53 False False False False False False  
54 False False False False False False  
55 False False False False False False  
56 False False False False False False  
.. ... ... ... ... ... ...  
944 False False False False False False  
945 False False False False False False  
946 False False False False False False  
947 False False False False False False  
948 False False False False False False  
  
[312 rows x 6 columns]  
date False  
type False  
year False  
avg\_price False  
size False  
nb\_sold False  
dtype: bool



### Удаление пропущенных значений

Теперь, когда вы знаете, что в вашем DataFrame есть пропущенные значения, у вас есть несколько вариантов их обработки. Один из способов - удалить их из набора данных полностью. В этом упражнении вы удалите пропущенные значения, удалив все строки, содержащие пропущенные значения.

Библиотека pandas импортирована как pd, и доступен avocados\_2016.

#### Инструкции

* Удалите строки из avocados\_2016, содержащие пропущенные значения, и сохраните оставшиеся строки в avocados\_complete.
* Проверьте, что все пропущенные значения были удалены из avocados\_complete. Посчитайте каждый столбец, в котором есть пропущенные значения, и выведите их.

# Удаление строк с пропущенными значениями  
avocados\_complete = \_\_\_\_  
  
# Проверка наличия пропущенных значений в столбцах  
print(\_\_\_\_)

date False  
type False  
year False  
avg\_price False  
size False  
nb\_sold False  
dtype: bool

### Замена пропущенных значений

Еще один способ обработки пропущенных значений - заменить их одним значением. Для числовых переменных один из вариантов - заменить значения на 0. Вы будете делать это здесь. Однако, заменяя пропущенные значения, вы делаете предположения о том, что означает отсутствие значения. В этом случае вы будете предполагать, что отсутствующее количество проданных означает отсутствие продаж этого типа авокадо в эту неделю.

В этом упражнении вы увидите, как замена пропущенных значений может влиять на распределение переменной с помощью гистограмм. Вы можете строить гистограммы для нескольких переменных следующим образом:

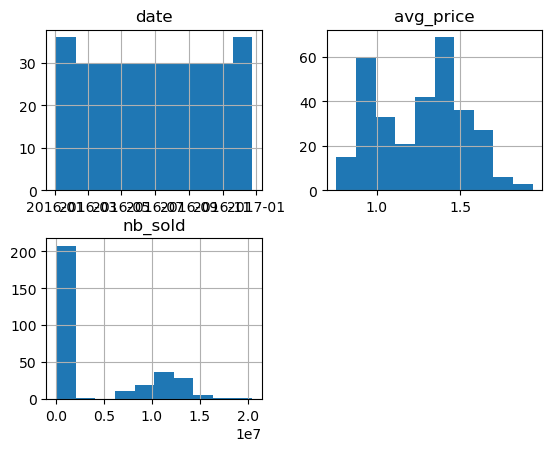
dogs[["height\_cm", "weight\_kg"]].hist()

Библиотека pandas импортирована как pd, а matplotlib.pyplot импортирована как plt. Доступен набор данных avocados\_2016.

#### Инструкции 1/2

* Создан список cols\_with\_missing, содержащий имена столбцов с пропущенными значениями: "small\_sold", "large\_sold" и "xl\_sold".
* Создайте гистограмму для этих столбцов.
* Покажите график.

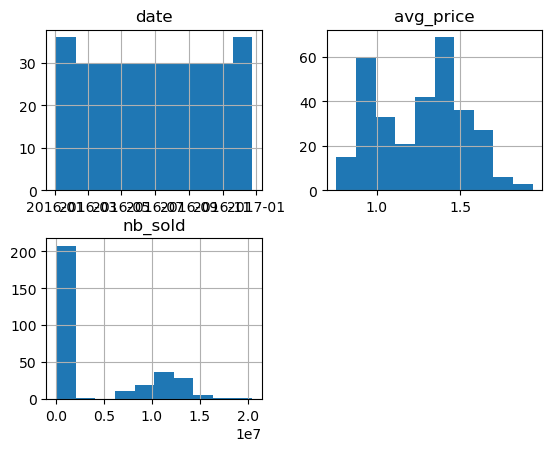
# Составление списка столбцов с пропущенными значениями  
cols\_with\_missing = ['date', 'avg\_price', 'nb\_sold']  
  
# Создание гистограмм, отображающих распределение столбцов cols\_with\_missing  
avocados\_2016[\_\_\_\_].\_\_\_\_  
  
# Показать график  
\_\_\_\_



#### Инструкции 2/2

* Замените отсутствующие значения в avocados\_2016 на нули и сохраните результат в avocados\_filled.
* Создайте гистограмму для столбцов cols\_with\_missing в avocados\_filled.

# Заполнить пропущенные значения нулями  
avocados\_filled = \_\_\_\_  
  
# Создать гистограммы для заполненных столбцов  
\_\_\_\_  
  
# Показать график  
plt.show()



### Список словарей

Недавно вы получили новые данные об авокадо за 2019 год, которые вы хотели бы поместить в DataFrame, используя метод списка словарей. Помните, что с помощью этого метода вы просматриваете данные построчно.

#### Инструкции

* Создайте список словарей с новыми данными под названием avocados\_list.
* Преобразуйте список в DataFrame с именем avocados\_2019.
* Выведите ваш новый DataFrame.

# Создание списка словарей с новыми данными  
avocados\_list = [  
 {\_\_\_\_: \_\_\_\_, \_\_\_\_: \_\_\_\_, \_\_\_\_: \_\_\_\_},  
 {\_\_\_\_: \_\_\_\_, \_\_\_\_: \_\_\_\_, \_\_\_\_: \_\_\_\_},  
]  
  
# Преобразование списка в DataFrame  
avocados\_2019 = \_\_\_\_  
  
# Вывод нового DataFrame  
\_\_\_\_

date small\_sold large\_sold  
0 2019-11-03 10376832 7835071  
1 2019-11-10 10717154 8561348

### Словарь списков

Только что поступило еще несколько данных! На этот раз вы будете использовать метод словаря списков, анализируя столбец данных по столбцу.

#### Инструкции

* Создайте словарь списков с новыми данными под названием avocados\_dict.
* Преобразуйте словарь в DataFrame с именем avocados\_2019.
* Распечатайте новый DataFrame.

# Создание словаря списков с новыми данными  
avocados\_dict = {  
 "\_\_\_\_": [\_\_\_\_],  
 "\_\_\_\_": [\_\_\_\_],  
 "\_\_\_\_": [\_\_\_\_]  
}  
  
# Преобразование словаря в DataFrame  
avocados\_2019 = \_\_\_\_  
  
# Вывод нового DataFrame  
\_\_\_\_

date small\_sold large\_sold  
0 2019-11-17 10859987 7674135  
1 2019-12-01 9291631 6238096

### CSV в DataFrame

Вы работаете в авиакомпании, и ваш руководитель попросил вас провести конкурентный анализ и посмотреть, как часто пассажиры, летающие другими авиакомпаниями, невольно сбиваются со своих рейсов. Вы получили CSV-файл (airline\_bumping.csv) от Министерства транспорта, содержащий данные о пассажирах, которым было принудительно отказано в посадке в 2016 и 2017 годах, но в нем нет нужных вам точных цифр. Чтобы понять это, вам нужно поместить CSV в DataFrame pandas и выполнить некоторые манипуляции!

pandas импортируется для вас как pd. «airline\_bumping.csv» находится в вашем рабочем каталоге.

#### Инструкции 1/4

* Прочтите CSV-файл «airline\_bumping.csv» и сохраните его как DataFrame под названием «airline\_bumping».
* Распечатайте первые несколько строк airline\_bumping.

# Данные  
data = {  
 "airline": ["DELTA AIR LINES", "VIRGIN AMERICA", "UNITED AIRLINES", "AMERICAN AIRLINES"], # Заполните все авиакомпании  
 "year": [2017, 2017, 2017, 2016], # Заполните соответствующие годы  
 "nb\_bumped": [679, 165, 2013, 1418], # Заполните значения nb\_bumped  
 "total\_passengers": [99796155, 6090029, 98765432, 15234924] # Заполните значения total\_passengers  
}  
  
df = pd.DataFrame(data)  
  
# Сохранение в CSV  
df.to\_csv('/Users/work/Disk-O.as.mounts/voil@mail.ru-mailru/data\_camp/learn/data\_manipulation\_with\_pandas/datasets/airlines\_data.csv', index=False)

airline\_bumping = \_\_\_\_  
print(\_\_\_\_)

airline year nb\_bumped total\_passengers  
0 DELTA AIR LINES 2017 679 99796155  
1 VIRGIN AMERICA 2017 165 6090029  
2 UNITED AIRLINES 2017 2013 98765432  
3 AMERICAN AIRLINES 2016 1418 15234924

#### Инструкции 2/4

* Для каждой группы авиакомпаний выберите столбцы nb\_bumped и total\_passengers и рассчитайте сумму (за оба года). Сохраните это как airlines\_totals.

# Для каждой авиакомпании выберите nb\_bumped и total\_passengers и просуммируйте  
airline\_totals = airline\_bumping.groupby(\_\_\_\_)[[\_\_\_\_]].\_\_\_\_

#### Инструкции 3/4

* Создайте новый столбец airlines\_totals под названием «bumps\_per\_10k», который представляет собой количество пассажиров, с которыми произошло столкновение, на 10 000 пассажиров в 2016 и 2017 годах.

# Создание нового столбца, bumps\_per\_10k: количество задержанных рейсов на 10 тыс. пассажиров для каждой авиакомпании  
airline\_totals["bumps\_per\_10k"] = \_\_\_\_ / \_\_\_\_ \* 10000

airline year nb\_bumped total\_passengers  
0 DELTA AIR LINES 2017 679 99796155  
1 VIRGIN AMERICA 2017 165 6090029  
2 UNITED AIRLINES 2017 2013 98765432  
3 AMERICAN AIRLINES 2016 1418 15234924

#### Инструкции 4/4

* Распечатайте airline\_totals, чтобы увидеть результаты ваших манипуляций.

# Вывести airline\_totals  
\_\_\_\_

airline year nb\_bumped total\_passengers  
0 DELTA AIR LINES 2017 679 99796155  
1 VIRGIN AMERICA 2017 165 6090029  
2 UNITED AIRLINES 2017 2013 98765432  
3 AMERICAN AIRLINES 2016 1418 15234924  
 nb\_bumped total\_passengers bumps\_per\_10k  
airline   
AMERICAN AIRLINES 1418 15234924 0.930756  
DELTA AIR LINES 679 99796155 0.068039  
UNITED AIRLINES 2013 98765432 0.203816  
VIRGIN AMERICA 165 6090029 0.270935

### DataFrame в CSV

Ты почти там! Чтобы облегчить чтение, вам нужно отсортировать данные и экспортировать их в CSV, чтобы ваши коллеги могли их прочитать.

pandas, поскольку pd был импортирован для вас.

#### Инструкции

* Сортировать данные airlines\_totals по значениям bumps\_per\_10k от самого высокого до самого низкого, сохраняя их как airlines\_totals\_sorted.
* Распечатайте отсортированный DataFrame.
* Сохраните отсортированный DataFrame в формате CSV с именем «airline\_totals\_sorted.csv».

# Создание airline\_totals\_sorted  
airline\_totals\_sorted = \_\_\_\_  
  
# Вывод airline\_totals\_sorted  
\_\_\_\_  
  
# Сохранение в файл airline\_totals\_sorted.csv  
\_\_\_\_

nb\_bumped total\_passengers bumps\_per\_10k  
airline   
AMERICAN AIRLINES 1418 15234924 0.930756  
VIRGIN AMERICA 165 6090029 0.270935  
UNITED AIRLINES 2013 98765432 0.203816  
DELTA AIR LINES 679 99796155 0.068039