# Объединение данных с помощью pandas

## Основы объединения данных

# Предполагая, что ваш блокнот Jupyter находится в папке exercises  
# Вы хотите получить доступ к файлу taxi\_owners.p в папке datasets  
  
import os  
import pickle  
import pandas as pd  
  
# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'taxi\_owners.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
taxi\_owners = data

Файл успешно открыт!

# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'taxi\_vehicles.p')  
  
try:  
 if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
 taxi\_vehicles = data  
 else:  
 print("Файл не найден.")  
except Exception as e:  
 print(f"Произошла ошибка: {e}")  
 taxi\_vehicles = None # Присваивание None 'taxi\_vehicles', если произошла ошибка  
  
taxi\_veh = data

Файл успешно открыт!

## Ваше первое внутреннее соединение

Вам предстоит выяснить, какие виды топлива наиболее популярны в такси Чикаго. Чтобы завершить анализ, вам необходимо объединить таблицы «taxi\_owners» и «taxi\_veh» в столбце «vid». Затем вы можете использовать объединенную таблицу вместе с методом .value\_counts(), чтобы найти наиболее распространенный тип Fuel\_type.

Поскольку на протяжении всего курса вы будете работать с pandas, пакет будет предварительно загружен для вас как pd в каждом упражнении этого курса. Также для вас загружаются DataFrames taxi\_owners и taxi\_veh.

### Инструкции 1/3

1. Объедините «taxi\_owners» с «taxi\_veh» в столбце «vid» и сохраните результат в «taxi\_own\_veh».

# Объединить таблицы taxi\_owners и taxi\_veh  
taxi\_own\_veh = taxi\_owners.\_\_\_\_  
  
# Вывести названия столбцов в taxi\_own\_veh  
print(taxi\_own\_veh.columns)

Index(['rid', 'vid', 'owner\_x', 'address', 'zip', 'make', 'model', 'year',  
 'fuel\_type', 'owner\_y'],  
 dtype='object')

### Инструкции 2/3

Установите суффиксы левой и правой таблицы для перекрывающихся столбцов слияния на \_own и \_veh соответственно.

# Объединить таблицы taxi\_owners и taxi\_veh, установив суффиксы  
taxi\_own\_veh = taxi\_owners.merge(taxi\_veh, on='vid', \_\_\_\_)  
  
# Вывести названия столбцов в taxi\_own\_veh  
print(taxi\_own\_veh.columns)

Index(['rid', 'vid', 'owner\_own', 'address', 'zip', 'make', 'model', 'year',  
 'fuel\_type', 'owner\_veh'],  
 dtype='object')

### Инструкции 3/3

Выберите столбец «fuel\_type» из «taxi\_own\_veh» и распечатайте «value\_counts()», чтобы найти наиболее популярные используемые «fuel\_types».

# Объединить таблицы taxi\_owners и taxi\_veh, установив суффиксы  
taxi\_own\_veh = taxi\_owners.merge(taxi\_veh, on='vid', suffixes=('\_own', '\_veh'))  
  
# Вывести value\_counts, чтобы найти наиболее популярный тип топлива  
print(taxi\_own\_veh['\_\_\_\_'].value\_counts())

HYBRID 2792  
GASOLINE 611  
FLEX FUEL 89  
COMPRESSED NATURAL GAS 27  
Name: fuel\_type, dtype: int64

# Путь к файлу wards.p  
ward\_file\_path = os.path.join(datasets\_directory, 'ward.p')  
  
# Загрузить wards.p, если он существует, как DataFrame с именем 'wards'  
if os.path.exists(ward\_file\_path):  
 with open(ward\_file\_path, 'rb') as file:  
 wards = pd.DataFrame(pickle.load(file))  
 print("Файл 'ward.p' открыт и сохранен в DataFrame 'wards'.")  
else:  
 print("Файл 'ward.p' не найден.")  
  
print(wards)

Файл 'ward.p' открыт и сохранен в DataFrame 'wards'.  
 ward alderman address zip  
0 1 Proco "Joe" Moreno 2058 NORTH WESTERN AVENUE 60647  
1 2 Brian Hopkins 1400 NORTH ASHLAND AVENUE 60622  
2 3 Pat Dowell 5046 SOUTH STATE STREET 60609  
3 4 William D. Burns 435 EAST 35TH STREET, 1ST FLOOR 60616  
4 5 Leslie A. Hairston 2325 EAST 71ST STREET 60649  
5 6 Roderick T. Sawyer 8001 S. MARTIN LUTHER KING DRIVE 60619  
6 7 Gregory I. Mitchell 2249 EAST 95TH STREET 60617  
7 8 Michelle A. Harris 8539 SOUTH COTTAGE GROVE AVENUE 60619  
8 9 Anthony A. Beale 34 EAST 112TH PLACE 60628  
9 10 Susan Sadlowski Garza 10500 SOUTH EWING AVENUE 60617  
10 11 Patrick Daley Thompson 3659 SOUTH HALSTED STREET 60609  
11 12 George Cardenas 3476 SOUTH ARCHER AVENUE 60608  
12 13 Marty Quinn 6500 SOUTH PULASKI ROAD 60629  
13 14 Edward M. Burke 2650 WEST 51ST STREET 60632  
14 15 Raymond A. Lopez 1650 WEST 63RD STREET 60636  
15 16 Toni L. Foulkes 3045 WEST 63RD STREET 60629  
16 17 David H. Moore 7313 SOUTH ASHLAND AVENUE 60636  
17 18 Derrick G. Curtis 8359 SOUTH PULASKI ROAD 60652  
18 19 Matthew J. O'Shea 10400 SOUTH WESTERN AVENUE 60643  
19 20 Willie B. Cochran 6357 SOUTH COTTAGE GROVE AVENUE 60637  
20 21 Howard B. Brookins, Jr. 9011 SOUTH ASHLAND AVENUE, UNIT B 60620  
21 22 Ricardo Munoz 2500 SOUTH ST. LOUIS AVENUE 60623  
22 23 Michael R. Zalewski 6247 SOUTH ARCHER AVENUE 60638  
23 24 Michael Scott, Jr. 1158 SOUTH KEELER AVENUE 60624  
24 25 Daniel "Danny" Solis 1800 SOUTH BLUE ISLAND AVENUE 60608  
25 26 Roberto Maldonado 2511 WEST DIVISION STREET 60622  
26 27 Walter Burnett, Jr. 4 NORTH WESTERN AVENUE 60612  
27 28 Jason C. Ervin 2602 WEST 16TH STREET 60612  
28 29 Chris Taliaferro 6272 WEST NORTH AVENUE 60639  
29 30 Ariel E. Reyboras 3559 NORTH MILWAUKEE AVENUE 60641  
30 31 Milagros "Milly" Santiago 2521 NORTH PULASKI ROAD 60639  
31 32 Scott Waguespack 2657 NORTH CLYBOURN AVENUE 60614  
32 33 Deborah Mell 3001 WEST IRVING PARK ROAD 60618  
33 34 Carrie M. Austin 507 WEST 111TH STREET 60628  
34 35 Carlos Ramirez-Rosa 2710 NORTH SAWYER AVENUE 60647  
35 36 Gilbert Villegas 6934 WEST DIVERSEY 60607  
36 37 Emma M. Mitts 4924 WEST CHICAGO AVENUE 60651  
37 38 Nicholas Sposato 3821 NORTH HARLEM AVENUE 60634  
38 39 Margaret Laurino 4404 WEST LAWRENCE AVENUE 60630  
39 40 Patrick J. O'Connor 5850 NORTH LINCOLN AVENUE 60659  
40 41 Anthony V. Napolitano 7442 NORTH HARLEM AVENUE 60631  
41 42 Brendan Reilly 325 WEST HURON STREET, SUITE 510 60654  
42 43 Michelle Smith 2523 NORTH HALSTED STREET 60614  
43 44 Tom Tunney 3223 NORTH SHEFFIELD AVENUE 60657  
44 45 John S. Arena 4754 NORTH MILWAUKEE AVENUE 60630  
45 46 James Cappleman 4544 NORTH BROADWAY AVENUE 60640  
46 47 Ameya Pawar 4243 NORTH LINCOLN AVENUE 60618  
47 48 Harry Osterman 5533 NORTH BROADWAY AVENUE 60640  
48 49 Joe Moore 7356 NORTH GREENVIEW AVENUE 60626  
49 50 Debra L. Silverstein 2949 WEST DEVON AVENUE, SUITE A 60659

# Путь к файлу census.p  
census\_file\_path = os.path.join(datasets\_directory, 'census.p')  
  
# Загрузить census.p, если он существует, как DataFrame с именем 'census'  
if os.path.exists(census\_file\_path):  
 with open(census\_file\_path, 'rb') as file:  
 census = pd.DataFrame(pickle.load(file))  
 print("Файл 'census.p' открыт и сохранен в DataFrame 'census'.")  
else:  
 print("Файл 'census.p' не найден.")  
  
print(census.head())

Файл 'census.p' открыт и сохранен в DataFrame 'census'.  
 ward pop\_2000 pop\_2010 change address \  
0 1 52951 56149 6% 2765 WEST SAINT MARY STREET   
1 2 54361 55805 3% WM WASTE MANAGEMENT 1500   
2 3 40385 53039 31% 17 EAST 38TH STREET   
3 4 51953 54589 5% 31ST ST HARBOR BUILDING LAKEFRONT TRAIL   
4 5 55302 51455 -7% JACKSON PARK LAGOON SOUTH CORNELL DRIVE   
  
 zip   
0 60647   
1 60622   
2 60653   
3 60653   
4 60637

## Внутренние соединения и количество возвращаемых строк

Все слияния, которые вы изучили до этого момента, называются внутренними соединениями. Необходимо понимать, что внутренние соединения возвращают только строки с совпадающими значениями в обеих таблицах. Вы исследуете это дальше, рассмотрев слияние таблиц «wards» и «census», а затем сравнив его со слиянием слегка измененных копий этих таблиц, названных «wards\_altered» и «census\_altered». В измененных таблицах изменена первая строка столбца подопечных. Вы увидите, как это влияет на слияние между ними. Таблицы для вас загружены.

Для этого упражнения важно знать, что таблицы «wards» и census начинаются с 50 строк.

### Инструкции 1/3

Объедините значения «wards» и «census» в столбце «ward» и сохраните результат в столбце «wards\_census».

# Объединить таблицы wards и census по столбцу 'ward'  
wards\_census = wards.merge(\_\_\_\_)  
  
# Вывести форму таблицы wards\_census  
print('Форма таблицы wards\_census:', wards\_census.shape)  
  
print(wards\_census)

Форма таблицы wards\_census: (50, 9)  
 ward alderman address\_x zip\_x \  
0 1 Proco "Joe" Moreno 2058 NORTH WESTERN AVENUE 60647   
1 2 Brian Hopkins 1400 NORTH ASHLAND AVENUE 60622   
2 3 Pat Dowell 5046 SOUTH STATE STREET 60609   
3 4 William D. Burns 435 EAST 35TH STREET, 1ST FLOOR 60616   
4 5 Leslie A. Hairston 2325 EAST 71ST STREET 60649   
5 6 Roderick T. Sawyer 8001 S. MARTIN LUTHER KING DRIVE 60619   
6 7 Gregory I. Mitchell 2249 EAST 95TH STREET 60617   
7 8 Michelle A. Harris 8539 SOUTH COTTAGE GROVE AVENUE 60619   
8 9 Anthony A. Beale 34 EAST 112TH PLACE 60628   
9 10 Susan Sadlowski Garza 10500 SOUTH EWING AVENUE 60617   
10 11 Patrick Daley Thompson 3659 SOUTH HALSTED STREET 60609   
11 12 George Cardenas 3476 SOUTH ARCHER AVENUE 60608   
12 13 Marty Quinn 6500 SOUTH PULASKI ROAD 60629   
13 14 Edward M. Burke 2650 WEST 51ST STREET 60632   
14 15 Raymond A. Lopez 1650 WEST 63RD STREET 60636   
15 16 Toni L. Foulkes 3045 WEST 63RD STREET 60629   
16 17 David H. Moore 7313 SOUTH ASHLAND AVENUE 60636   
17 18 Derrick G. Curtis 8359 SOUTH PULASKI ROAD 60652   
18 19 Matthew J. O'Shea 10400 SOUTH WESTERN AVENUE 60643   
19 20 Willie B. Cochran 6357 SOUTH COTTAGE GROVE AVENUE 60637   
20 21 Howard B. Brookins, Jr. 9011 SOUTH ASHLAND AVENUE, UNIT B 60620   
21 22 Ricardo Munoz 2500 SOUTH ST. LOUIS AVENUE 60623   
22 23 Michael R. Zalewski 6247 SOUTH ARCHER AVENUE 60638   
23 24 Michael Scott, Jr. 1158 SOUTH KEELER AVENUE 60624   
24 25 Daniel "Danny" Solis 1800 SOUTH BLUE ISLAND AVENUE 60608   
25 26 Roberto Maldonado 2511 WEST DIVISION STREET 60622   
26 27 Walter Burnett, Jr. 4 NORTH WESTERN AVENUE 60612   
27 28 Jason C. Ervin 2602 WEST 16TH STREET 60612   
28 29 Chris Taliaferro 6272 WEST NORTH AVENUE 60639   
29 30 Ariel E. Reyboras 3559 NORTH MILWAUKEE AVENUE 60641   
30 31 Milagros "Milly" Santiago 2521 NORTH PULASKI ROAD 60639   
31 32 Scott Waguespack 2657 NORTH CLYBOURN AVENUE 60614   
32 33 Deborah Mell 3001 WEST IRVING PARK ROAD 60618   
33 34 Carrie M. Austin 507 WEST 111TH STREET 60628   
34 35 Carlos Ramirez-Rosa 2710 NORTH SAWYER AVENUE 60647   
35 36 Gilbert Villegas 6934 WEST DIVERSEY 60607   
36 37 Emma M. Mitts 4924 WEST CHICAGO AVENUE 60651   
37 38 Nicholas Sposato 3821 NORTH HARLEM AVENUE 60634   
38 39 Margaret Laurino 4404 WEST LAWRENCE AVENUE 60630   
39 40 Patrick J. O'Connor 5850 NORTH LINCOLN AVENUE 60659   
40 41 Anthony V. Napolitano 7442 NORTH HARLEM AVENUE 60631   
41 42 Brendan Reilly 325 WEST HURON STREET, SUITE 510 60654   
42 43 Michelle Smith 2523 NORTH HALSTED STREET 60614   
43 44 Tom Tunney 3223 NORTH SHEFFIELD AVENUE 60657   
44 45 John S. Arena 4754 NORTH MILWAUKEE AVENUE 60630   
45 46 James Cappleman 4544 NORTH BROADWAY AVENUE 60640   
46 47 Ameya Pawar 4243 NORTH LINCOLN AVENUE 60618   
47 48 Harry Osterman 5533 NORTH BROADWAY AVENUE 60640   
48 49 Joe Moore 7356 NORTH GREENVIEW AVENUE 60626   
49 50 Debra L. Silverstein 2949 WEST DEVON AVENUE, SUITE A 60659   
  
 pop\_2000 pop\_2010 change \  
0 52951 56149 6%   
1 54361 55805 3%   
2 40385 53039 31%   
3 51953 54589 5%   
4 55302 51455 -7%   
5 54989 52341 -5%   
6 54593 51581 -6%   
7 54039 51687 -4%   
8 52008 51519 -1%   
9 56613 51535 -9%   
10 64228 51497 -20%   
11 68922 52235 -24%   
12 64382 53722 -17%   
13 80143 54031 -33%   
14 56057 51501 -8%   
15 50205 51954 3%   
16 49264 51846 5%   
17 55043 52992 -4%   
18 54546 51525 -6%   
19 51854 52372 1%   
20 51751 51632 0%   
21 59734 53515 -10%   
22 63691 53728 -16%   
23 50879 54909 8%   
24 55954 54539 -3%   
25 56841 53516 -6%   
26 61287 52939 -14%   
27 49423 55199 12%   
28 61949 55267 -11%   
29 72698 55560 -24%   
30 65045 53724 -17%   
31 57204 55184 -4%   
32 63695 55598 -13%   
33 49922 51599 3%   
34 57588 55281 -4%   
35 63376 54766 -14%   
36 56120 51538 -8%   
37 66011 56001 -15%   
38 64291 55882 -13%   
39 58652 55319 -6%   
40 56127 55991 0%   
41 68102 55870 -18%   
42 57668 56170 -3%   
43 58758 56058 -5%   
44 60653 55967 -8%   
45 56587 53784 -5%   
46 52108 55074 6%   
47 56246 55014 -2%   
48 59435 54633 -8%   
49 62383 55809 -11%   
  
 address\_y zip\_y   
0 2765 WEST SAINT MARY STREET 60647   
1 WM WASTE MANAGEMENT 1500 60622   
2 17 EAST 38TH STREET 60653   
3 31ST ST HARBOR BUILDING LAKEFRONT TRAIL 60653   
4 JACKSON PARK LAGOON SOUTH CORNELL DRIVE 60637   
5 150 WEST 74TH STREET 60636   
6 8549 SOUTH OGLESBY AVENUE 60617   
7 1346-1352 EAST 75TH STREET 60649   
8 11039-11059 SOUTH WENTWORTH AVENUE 60628   
9 10534 SOUTH AVENUE F 46394   
10 943-947 WEST 14TH PLACE 60607   
11 CP 46 STEVENSON EXPRESSWAY 60632   
12 SOUTH RAMP SOUTH LARAMIE AVENUE 60638   
13 4540 WEST 51ST STREET 60632   
14 CHICAGO FIRE DEPARTMENT ENGINE COMPANY 123 2215 60632   
15 6036 SOUTH WOOD STREET 60636   
16 7216 SOUTH WINCHESTER AVENUE 60636   
17 3286 WEST COLUMBUS AVENUE 60652   
18 9999 SOUTH FRANCISCO AVENUE 60805   
19 DAN RYAN EXPRESSWAY PARK MANOR 60621   
20 8852-8854 SOUTH EMERALD AVENUE 60620   
21 4233 WEST 36TH STREET 60632   
22 CHICAGO MIDWAY INTERNATIONAL AIRPORT WEST 62ND... 60629   
23 1635 SOUTH CHRISTIANA AVENUE 60623   
24 1632-1746 SOUTH MILLER STREET 60608   
25 LITTLE CUBS FIELD COMFORT STATION 1400 60622   
26 2151-2153 WEST CHICAGO AVENUE 60651   
27 RML SPECIALTY HOSPITAL 3435 60624   
28 1241 NORTH RIDGELAND AVENUE 60302   
29 5118 WEST FLETCHER STREET 60641   
30 2854 NORTH KEATING AVENUE 60641   
31 2901 NORTH WASHTENAW AVENUE 60618   
32 4041-4043 NORTH RICHMOND STREET 60625   
33 11544-11546 SOUTH PEORIA STREET 60827   
34 3634 WEST BELMONT AVENUE 60618   
35 2918 NORTH RUTHERFORD AVENUE 60634   
36 4738-4748 WEST RICE STREET 60651   
37 7307-7331 WEST IRVING PARK ROAD 60706   
38 QUEEN OF ALL SAINTS BASILICA 6280 60646   
39 5536 NORTH ARTESIAN AVENUE 60645   
40 1652 SOUTH CLIFTON AVENUE 60068   
41 410-420 WEST GRAND AVENUE 60654   
42 LINCOLN PARK ZOO 2001 60614   
43 507-513 WEST ALDINE AVENUE 60657   
44 CONGREGATIONAL CHURCH OF JEFFERSON PARK 5320 60630   
45 UPTOWN BROADWAY BUILDING 4743-4763 60640   
46 2153 WEST BERTEAU AVENUE 60618   
47 1025 WEST HOLLYWOOD AVENUE 60660   
48 1426 WEST ESTES AVENUE 60645   
49 2638 WEST NORTH SHORE AVENUE 60645

### Инструкции 2/3

Объедините таблицы wards\_altered и census в столбце ward и обратите внимание на разницу в возвращаемых строках.

# Предполагая, что 'wards' и 'census' - это ваши DataFrames  
  
# Создать копию DataFrame 'wards' для внесения изменений  
wards\_altered = wards.copy()  
  
# Преобразовать столбец 'ward' в целочисленный тип в 'wards\_altered'  
wards\_altered['ward'] = wards\_altered['ward'].astype(int)  
  
# Добавить 60 к первому элементу в столбце 'ward' в 'wards\_altered'  
wards\_altered.loc[0, 'ward'] += 60  
  
# Вывести несколько первых строк таблицы wards\_altered, чтобы увидеть изменение  
print(wards\_altered[['ward']].head())  
  
# Проверить, можно ли преобразовать столбец 'ward' в 'census' в целочисленный тип  
if census['ward'].astype(str).str.isdigit().all():  
 # Преобразовать столбец 'ward' в 'census' в целочисленный тип  
 census['ward'] = census['ward'].astype(int)  
  
 # Объединить таблицы 'wards\_altered' и 'census' по столбцу 'ward'  
 wards\_altered\_census = \_\_\_\_.merge(census, \_\_\_\_)  
 print('Форма таблицы wards\_altered\_census:', wards\_altered\_census.shape)  
else:  
 print("Невозможно преобразовать столбец 'ward' в DataFrame 'census' в целочисленный тип для объединения.")

ward  
0 61  
1 2  
2 3  
3 4  
4 5  
Форма таблицы wards\_altered\_census: (49, 9)

### Инструкции 3/3

Объедините таблицы «ward» и «census\_altered» в столбце «ward» и обратите внимание на разницу в возвращаемых строках.

# Предполагая, что 'wards' и 'census' - это ваши DataFrames  
  
# Создать копию первого столбца DataFrame 'census' и присвоить ему имя 'census\_altered'  
census\_altered = census.iloc[:, [0]].copy()  
  
# Установить тип столбца 'ward' как строковый для возможности использования значений 'None'  
census\_altered['ward'] = census\_altered['ward'].astype(str)  
  
# Установить первый элемент в столбце 'ward' в 'None'  
census\_altered.loc[0, 'ward'] = 'None'  
  
# Преобразовать столбец 'ward' в 'wards' в строковый тип  
wards['ward'] = wards['ward'].astype(str)  
  
# Объединить таблицы 'wards' и 'census\_altered' по столбцу 'ward'  
wards\_census\_altered = wards.\_\_\_\_  
  
# Вывести несколько первых строк таблицы census\_altered, чтобы увидеть изменение  
print(census\_altered[['ward']].head())  
  
# Вывести форму таблицы wards\_census\_altered  
print('Форма таблицы wards\_census\_altered:', wards\_census\_altered.shape)

ward  
0 None  
1 2  
2 3  
3 4  
4 5  
Форма таблицы wards\_census\_altered: (49, 4)

## Слияние один-ко-многим

У бизнеса может быть один или несколько владельцев. В этом упражнении вы продолжите приобретать опыт слияний «один ко многим», объединяя таблицу владельцев бизнеса, называемую «biz\_owners», с таблицей «licenses». Вспомните урок: при отношении «один ко многим» строка в левой таблице может повторяться, если она связана с несколькими строками в правой таблице. В этом уроке вы изучите это подробнее, выяснив, какое звание владельца бизнеса является наиболее распространенным. (т. е. секретарь, генеральный директор или вице-президент)

DataFrame licenses и biz\_owners загружаются автоматически.

# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'licenses.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
licenses = data  
print(licenses.head())

Файл успешно открыт!  
 account ward aid business address zip  
0 307071 3 743 REGGIE'S BAR & GRILL 2105 S STATE ST 60616  
1 10 10 829 HONEYBEERS 13200 S HOUSTON AVE 60633  
2 10002 14 775 CELINA DELI 5089 S ARCHER AVE 60632  
3 10005 12 NaN KRAFT FOODS NORTH AMERICA 2005 W 43RD ST 60609  
4 10044 44 638 NEYBOUR'S TAVERN & GRILLE 3651 N SOUTHPORT AVE 60613

# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'business\_owners.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
biz\_owners = data  
  
print(biz\_owners.head())

Файл успешно открыт!  
 account first\_name last\_name title  
0 10 PEARL SHERMAN PRESIDENT  
1 10 PEARL SHERMAN SECRETARY  
2 10002 WALTER MROZEK PARTNER  
3 10002 CELINA BYRDAK PARTNER  
4 10005 IRENE ROSENFELD PRESIDENT

### Инструкции

– Начиная с таблицы «licenses» слева, объедините ее с таблицей «biz\_owners» в столбце «account» и сохраните результаты в переменной с именем «licenses\_owners».

* Сгруппируйте licenses\_owners по title и подсчитайте количество аккаунтов для каждого титула. Сохраните результат как counted\_df.
* Отсортируйте counted\_df по количеству ***аккаунтов*** в ***по убыванию*** и сохраните это значение как переменную с именем sorted\_df.
* Используйте метод .head() для печати первых нескольких строк sorted\_df.

# Объединить таблицы licenses и biz\_owners по столбцу 'account'  
licenses\_owners = \_\_\_\_  
  
# Сгруппировать результаты по столбцу 'title' и подсчитать количество учетных записей  
counted\_df = licenses\_owners.groupby(\_\_\_\_).agg({'account':'count'})  
  
# Отсортировать counted\_df в порядке убывания  
sorted\_df = counted\_df.sort\_values(\_\_\_\_)  
  
# Использовать метод .head() для вывода нескольких первых строк sorted\_df  
print(\_\_\_\_)

account  
title   
PRESIDENT 6259  
SECRETARY 5205  
SOLE PROPRIETOR 1658  
OTHER 1200  
VICE PRESIDENT 970

## Всего пассажиров за месяц

Ваша цель — найти общее количество поездок, выполненных пассажирами, проезжающими через станцию Уилсон (station\_name == 'Wilson') при поездке в системе общественного транспорта Чикаго в будние дни (day\_type == 'Weekday') в июле (месяц). == 7). К счастью, Чикаго предоставляет эти подробные данные, но они представлены в трех разных таблицах. Чтобы ответить на вопрос, вам предстоит объединить эти таблицы. Эти данные отличаются от данных, связанных с бизнесом, которые вы видели до сих пор, но предоставляется вся информация, необходимая для ответа на вопрос.

Для вас загружены кадры данных «cal», «ridership» и «stations». Взаимосвязь между таблицами можно увидеть на диаграмме ниже.

The relationship between the tables

# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'cta\_calendar.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
cal = data  
  
print(cal.head())

Файл успешно открыт!  
 year month day day\_type  
0 2019 1 1 Sunday/Holiday  
1 2019 1 2 Weekday  
2 2019 1 3 Weekday  
3 2019 1 4 Weekday  
4 2019 1 5 Saturday

# Получить текущую директорию  
current\_directory = os.getcwd()  
  
# Перейти в папку datasets  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в папке datasets  
file\_path = os.path.join(datasets\_directory, 'cta\_ridership.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому откройте его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работайте с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
ridership = data  
  
print(ridership.head())

Файл успешно открыт!  
 station\_id year month day rides  
0 40010 2019 1 1 576  
1 40010 2019 1 2 1457  
2 40010 2019 1 3 1543  
3 40010 2019 1 4 1621  
4 40010 2019 1 5 719

# Get the current directory  
current\_directory = os.getcwd()  
  
# Navigate to the datasets folder  
datasets\_directory = os.path.join(current\_directory, '..', 'datasets')  
  
# Check if the file exists in the datasets folder  
file\_path = os.path.join(datasets\_directory, 'stations.p')  
  
if os.path.exists(file\_path):  
 # File exists, so open it  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Work with the 'data' object as needed  
 print("File opened successfully!")  
else:  
 print("File not found.")  
stations = data  
  
print(stations.head())

File not found.  
 station\_id year month day rides  
0 40010 2019 1 1 576  
1 40010 2019 1 2 1457  
2 40010 2019 1 3 1543  
3 40010 2019 1 4 1621  
4 40010 2019 1 5 719

### Инструкции 1/3

Объедините таблицы «ridership» и «cal», начиная с таблицы «ridership» слева, и сохраните результат в переменной «ridership\_cal». Если ваш код выполняется слишком долго, ваши условия слияния могут быть неправильными.

# Объединить таблицы ridership и cal  
ridership\_cal = ridership.merge(\_\_\_\_)

### Инструкции 2/3

Расширьте предыдущее объединение до трех таблиц, объединив также таблицу stations.

# Merge the ridership, cal, and stations tables  
ridership\_cal\_stations = ridership.merge(cal, on=['year','month','day']).merge(\_\_\_\_)  
  
print(ridership\_cal\_stations)

station\_id year\_x month\_x day\_x rides\_x day\_type year\_y \  
0 40010 2019 1 1 576 Sunday/Holiday 2019   
1 40010 2019 1 1 576 Sunday/Holiday 2019   
2 40010 2019 1 1 576 Sunday/Holiday 2019   
3 40010 2019 1 1 576 Sunday/Holiday 2019   
4 40010 2019 1 1 576 Sunday/Holiday 2019   
... ... ... ... ... ... ... ...   
1199020 41660 2019 12 31 13430 Weekday 2019   
1199021 41660 2019 12 31 13430 Weekday 2019   
1199022 41660 2019 12 31 13430 Weekday 2019   
1199023 41660 2019 12 31 13430 Weekday 2019   
1199024 41660 2019 12 31 13430 Weekday 2019   
  
 month\_y day\_y rides\_y   
0 1 1 576   
1 1 2 1457   
2 1 3 1543   
3 1 4 1621   
4 1 5 719   
... ... ... ...   
1199020 12 27 13898   
1199021 12 28 9485   
1199022 12 29 7581   
1199023 12 30 15332   
1199024 12 31 13430   
  
[1199025 rows x 10 columns]

### Инструкции 3/3

Создайте переменную с именем filter\_criteria, чтобы выбрать соответствующие строки из объединенной таблицы и суммировать столбец rides.

# Объединить таблицы ridership, cal и stations  
ridership\_cal\_stations = ridership.merge(cal, on=['year', 'month', 'day']) \  
 .merge(stations, on='station\_id')  
  
# Создать фильтр для фильтрации ridership\_cal\_stations  
filter\_criteria = ((ridership\_cal\_stations['month'] == 7)  
 & (ridership\_cal\_stations['day\_type'] == 'Weekday')  
 & (ridership\_cal\_stations['station\_name'] == 'Wilson'))  
  
# Использовать .loc и фильтр для выбора поездок  
print(ridership\_cal\_stations.loc[filter\_criteria, 'rides'].sum())

---------------------------------------------------------------------------  
KeyError Traceback (most recent call last)  
File ~/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/base.py:3802, in Index.get\_loc(self, key, method, tolerance)  
 3801 try:  
-> 3802 return self.\_engine.get\_loc(casted\_key)  
 3803 except KeyError as err:  
  
File ~/anaconda3/lib/python3.10/site-packages/pandas/\_libs/index.pyx:138, in pandas.\_libs.index.IndexEngine.get\_loc()  
  
File ~/anaconda3/lib/python3.10/site-packages/pandas/\_libs/index.pyx:165, in pandas.\_libs.index.IndexEngine.get\_loc()  
  
File pandas/\_libs/hashtable\_class\_helper.pxi:5745, in pandas.\_libs.hashtable.PyObjectHashTable.get\_item()  
  
File pandas/\_libs/hashtable\_class\_helper.pxi:5753, in pandas.\_libs.hashtable.PyObjectHashTable.get\_item()  
  
KeyError: 'month'  
  
The above exception was the direct cause of the following exception:  
  
KeyError Traceback (most recent call last)  
Cell In[65], line 6  
 2 ridership\_cal\_stations = ridership.merge(cal, on=['year', 'month', 'day']) \  
 3 .merge(stations, on='station\_id')  
 5 # Создать фильтр для фильтрации ridership\_cal\_stations  
----> 6 filter\_criteria = ((ridership\_cal\_stations['month'] == 7)  
 7 & (ridership\_cal\_stations['day\_type'] == 'Weekday')  
 8 & (ridership\_cal\_stations['station\_name'] == 'Wilson'))  
 10 # Использовать .loc и фильтр для выбора поездок  
 11 print(ridership\_cal\_stations.loc[filter\_criteria, 'rides'].sum())  
  
File ~/anaconda3/lib/python3.10/site-packages/pandas/core/frame.py:3807, in DataFrame.\_\_getitem\_\_(self, key)  
 3805 if self.columns.nlevels > 1:  
 3806 return self.\_getitem\_multilevel(key)  
-> 3807 indexer = self.columns.get\_loc(key)  
 3808 if is\_integer(indexer):  
 3809 indexer = [indexer]  
  
File ~/anaconda3/lib/python3.10/site-packages/pandas/core/indexes/base.py:3804, in Index.get\_loc(self, key, method, tolerance)  
 3802 return self.\_engine.get\_loc(casted\_key)  
 3803 except KeyError as err:  
-> 3804 raise KeyError(key) from err  
 3805 except TypeError:  
 3806 # If we have a listlike key, \_check\_indexing\_error will raise  
 3807 # InvalidIndexError. Otherwise we fall through and re-raise  
 3808 # the TypeError.  
 3809 self.\_check\_indexing\_error(key)  
  
KeyError: 'month'

## Объединение трех таблиц

Чтобы закрепить концепцию слияния трех DataFrame, выполните еще одно упражнение. Разумное расширение нашего обзора бизнес-данных Чикаго могло бы включать рассмотрение демографической информации о районах, где расположены предприятия. Вам предоставлена таблица медианного дохода по почтовому индексу. Вы объедините таблицы лицензий и подопечных с новой таблицей дохода по почтовому индексу под названием zip\_demo.

Фреймы данных licenses, ward и zip\_demo уже загружены.

# Получить текущий каталог  
current\_directory = os.getcwd()  
  
# Перейти в каталог с данными  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в каталоге с данными  
file\_path = os.path.join(datasets\_directory, 'zip\_demo.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому открываем его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работаем с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
zip\_demo = data  
  
print(zip\_demo.head())

Файл успешно открыт!  
 zip income  
0 60630 70122  
1 60640 50488  
2 60622 87143  
3 60614 100116  
4 60608 41226

# Объединить лицензии и zip\_demo по zip; и объединить районы по ward  
licenses\_zip\_ward = licenses.merge\_\_\_\_ \_\_\_\_  
  
# Вывести результаты по альдерменам и показать медианный доход  
print(\_\_\_\_.groupby(\_\_\_\_).agg({'income':'median'}))

income  
alderman   
Ameya Pawar 66246.0  
Anthony A. Beale 38206.0  
Anthony V. Napolitano 82226.0  
Ariel E. Reyboras 41307.0  
Brendan Reilly 110215.0  
Brian Hopkins 87143.0  
Carlos Ramirez-Rosa 66246.0  
Carrie M. Austin 38206.0  
Chris Taliaferro 55566.0  
Daniel "Danny" Solis 41226.0  
David H. Moore 33304.0  
Deborah Mell 66246.0  
Debra L. Silverstein 50554.0  
Derrick G. Curtis 65770.0  
Edward M. Burke 42335.0  
Emma M. Mitts 36283.0  
George Cardenas 33959.0  
Gilbert Villegas 41307.0  
Gregory I. Mitchell 24941.0  
Harry Osterman 45442.0  
Howard B. Brookins, Jr. 33304.0  
James Cappleman 79565.0  
Jason C. Ervin 41226.0  
Joe Moore 39163.0  
John S. Arena 70122.0  
Leslie A. Hairston 28024.0  
Margaret Laurino 70122.0  
Marty Quinn 67045.0  
Matthew J. O'Shea 59488.0  
Michael R. Zalewski 42335.0  
Michael Scott, Jr. 31445.0  
Michelle A. Harris 32558.0  
Michelle Smith 100116.0  
Milagros "Milly" Santiago 41307.0  
Nicholas Sposato 62223.0  
Pat Dowell 46340.0  
Patrick Daley Thompson 41226.0  
Patrick J. O'Connor 50554.0  
Proco "Joe" Moreno 87143.0  
Raymond A. Lopez 33959.0  
Ricardo Munoz 31445.0  
Roberto Maldonado 68223.0  
Roderick T. Sawyer 32558.0  
Scott Waguespack 68223.0  
Susan Sadlowski Garza 38417.0  
Tom Tunney 88708.0  
Toni L. Foulkes 27573.0  
Walter Burnett, Jr. 87143.0  
William D. Burns 107811.0  
Willie B. Cochran 28024.0

## Слияние «один ко многим» с несколькими таблицами

Предположим, что в этом упражнении вы хотите начать бизнес в Чикаго. Ваша идеальная идея — основать компанию, которая будет использовать коз для стрижки газонов для других предприятий. Однако вам нужно выбрать место в городе, чтобы разместить свою козью ферму. Вам нужно место с большим пространством и относительно небольшим количеством предприятий и людей вокруг, чтобы избежать жалоб на запах. Вам нужно будет объединить три таблицы, чтобы выбрать свое местоположение. Таблица Land\_use содержит информацию о проценте свободной земли по городским округам. В таблице переписи населения указано население по округам, а в таблице лицензий перечислены предприятия по округам.

Таблицы «land\_use», «census» и «licenses» уже загружены.

# Получить текущий каталог  
current\_directory = os.getcwd()  
  
# Перейти в каталог с данными  
datasets\_directory = os.path.join(current\_directory, 'datasets')  
  
# Проверить, существует ли файл в каталоге с данными  
file\_path = os.path.join(datasets\_directory, 'land\_use.p')  
  
if os.path.exists(file\_path):  
 # Файл существует, поэтому открываем его  
 with open(file\_path, 'rb') as file:  
 data = pickle.load(file)  
 # Работаем с объектом 'data' по необходимости  
 print("Файл успешно открыт!")  
else:  
 print("Файл не найден.")  
land\_use = data  
  
print(land\_use.head())

Файл успешно открыт!  
 ward residential commercial industrial vacant other  
0 1 41 9 2 2 46  
1 2 31 11 6 2 50  
2 3 20 5 3 13 59  
3 4 22 13 0 7 58  
4 5 25 3 1 3 68

### Инструкции 1/3

Объедините данные «land\_use» и «census» в столбце «ward». Объедините результат с «лицензиями» в столбце «ward», используя суффикс «\_cen» для левой таблицы и «\_lic» для правой таблицы. Сохраните это в переменной «land\_cen\_lic».

# Преобразование столбца 'ward' в DataFrame Land\_use в тот же тип данных, что и другие DataFrame.  
land\_use['ward'] = land\_use['ward'].astype(int)  
  
# Объединить данные о землепользовании и переписи в столбце района  
merged\_data = pd.merge(land\_use, census, on='ward')  
  
# Преобразование столбца «Ward» в DataFrame лицензий в тот же тип данных  
licenses['ward'] = licenses['ward'].astype(int)  
  
# Объедините merged\_data и лицензии в столбце палаты, предоставив суффиксы для ясности.  
land\_cen\_lic = \_\_\_\_

### Инструкции 2/3

Сгруппируйте land\_cen\_lic по ward, pop\_2010 (население в 2010 году) и vacant, затем подсчитайте количество accounts. Сохраните результаты в pop\_vac\_lic.

# Group by ward, pop\_2010, and vacant, then count the # of accounts  
pop\_vac\_lic = land\_cen\_lic.groupby(\_\_\_\_, as\_index=False).agg({'account':'count'})

### Инструкции 3/3

Отсортируйте pop\_vac\_lic по vacant, account и pop\_2010 в порядке убывания, возрастания и возрастания соответственно. Сохраните его как sorted\_pop\_vac\_lic.

# Sort pop\_vac\_lic and print the results  
sorted\_pop\_vac\_lic = pop\_vac\_lic.sort\_values(\_\_\_\_, ascending=\_\_\_\_)  
  
# Print the top few rows of sorted\_pop\_vac\_lic  
print(sorted\_pop\_vac\_lic.head())

ward pop\_2010 vacant account  
6 7 51581 19 80  
19 20 52372 15 123  
9 10 51535 14 130  
23 24 54909 13 98  
15 16 51954 13 156