Лабораторная работа №31 (2 часа)

Тема работы: «Разработка, отладка и испытание программ анализа данных посредством «реляционной алгебры»»

1 Цель работы

Закрепить навык анализа данных посредством «реляционной алгебры»

2 Задание

Создайте два объекта Series с данными. Необходимо с помощью библиотеки Pandas выполнить объединение двух наборами данных следующим образом:

- ✓ inner;
- ✓ outer;

3 Оснащение работы

Задание по варианту, ЭВМ, среда разработки **Python 3.7, IDLE**.

4 Основные теоретические сведения

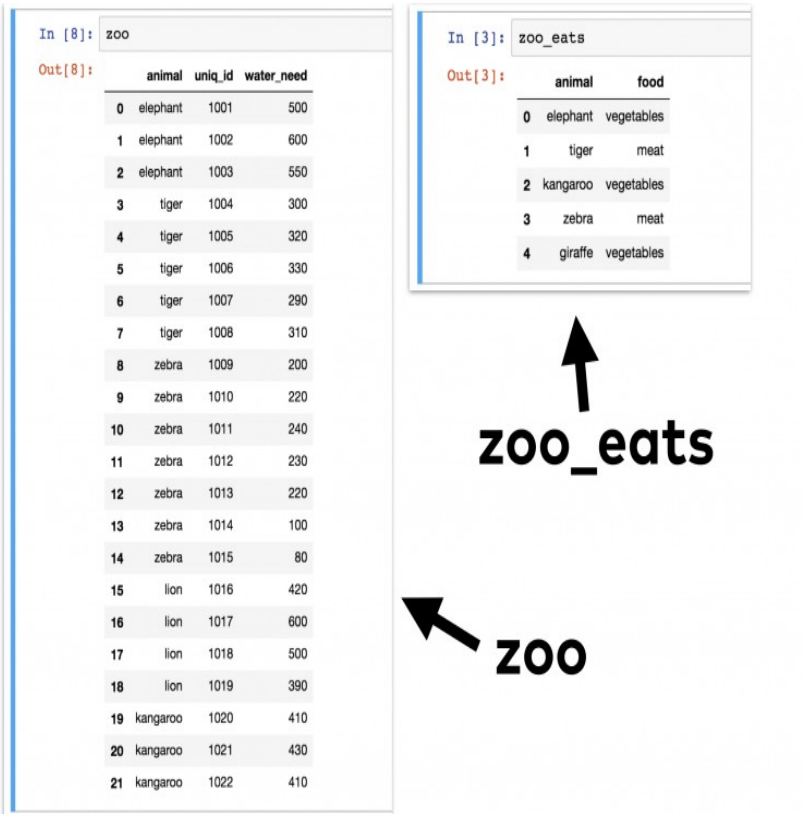
Merge в pandas («объединение» Data Frames)

В реальных проектах данные обычно не хранятся в одной таблице. Вместо нее используется много маленьких. И на то есть несколько причин. С помощью нескольких таблиц данными легче управлять, проще избегать «многословия», можно экономить место на диске, а запросы к таблицам обрабатываются быстрее.

Суть в том, что при работе с данными довольно часто придется вытаскивать данные из двух и более разных страниц. Это делается с помощью merge.

Примечание: хотя в pandas это называется merge, метод почти не отличается от JOIN в SQL.

Рассмотрим пример. Для этого можно взять DataFrame zoo (из предыдущих частей руководства), в котором есть разные животные. Но в этот раз нужен еще один DataFrame — zoo_eats, в котором будет описаны пищевые требования каждого вида.



Теперь нужно объединить два эти Data Frames в один. Чтобы получилось нечто подобное:

Out[4]:

	animal	uniq_id	water_need	food
0	elephant	1001	500	vegetables
1	elephant	1002	600	vegetables
2	elephant	1003	550	vegetables
3	tiger	1004	300	meat
4	tiger	1005	320	meat
5	tiger	1006	330	meat
6	tiger	1007	290	meat
7	tiger	1008	310	meat
8	zebra	1009	200	meat
9	zebra	1010	220	meat
10	zebra	1011	240	meat
11	zebra	1012	230	meat
12	zebra	1013	220	meat
13	zebra	1014	100	meat
14	zebra	1015	80	meat
15	kangaroo	1020	410	vegetables
16	kangaroo	1021	430	vegetables
17	kangaroo	1022	410	vegetables

В этой таблице можно проанализировать, например, сколько животных в зоопарке едят мясо или овощи.

Как делается merge?

В первую очередь нужно создать DataFrame zoo_eats, потому что zoo уже имеется из прошлых частей. Для упрощения задачи вот исходные данные:

animal;food
elephant;vegetables
tiger;meat
kangaroo;vegetables
zebra;vegetables
giraffe;vegetables

О том, как превратить этот набор в DataFrame, написано в первом уроке по pandas. Но есть способ для ленивых. Нужно лишь скопировать эту длинную строку в Jupyter Notebook pandas_tutorial_1, который был создан еще в первой части руководства.

zoo_eats = pd.DataFrame([['elephant','vegetables'], ['tiger','meat'], ['kangaroo','vegetables'], ['zebra','vegetables'], ['giraffe','vegetables']], columns=['animal', 'food'])

И вот готов DataFrame zoo_eats.

```
In [5]: zoo_eats = pd.DataFrame([['elephant','vegetables'], ['tiger','meat'], ['kangaroo','vegetables'], ['zebra','vegetables']

In [6]: zoo_eats
Out[6]:
```

	animal	food
0	elephant	vegetables
1	tiger	meat
2	kangaroo	vegetables
3	zebra	vegetables
4	giraffe	vegetables

Теперь пришло время метода merge:

zoo.merge(zoo_eats)

Out[4]:

	animal	uniq_id	water_need	food
0	elephant	1001	500	vegetables
1	elephant	1002	600	vegetables
2	elephant	1003	550	vegetables
3	tiger	1004	300	meat
4	tiger	1005	320	meat
5	tiger	1006	330	meat
6	tiger	1007	290	meat
7	tiger	1008	310	meat
8	zebra	1009	200	meat
9	zebra	1010	220	meat
10	zebra	1011	240	meat
11	zebra	1012	230	meat
12	zebra	1013	220	meat
13	zebra	1014	100	meat
14	zebra	1015	80	meat
15	kangaroo	1020	410	vegetables
16	kangaroo	1021	430	vegetables
17	kangaroo	1022	410	vegetables

(А где же все львы? К этому вернемся чуть позже).

Это было просто, не так ли? Но стоит разобрать, что сейчас произошло:

Сначала был указан первый DataFrame (zoo). Потом к нему применен метод .merge(). В качестве его параметра выступает новый DataFrame (zoo_eats). Можно было сделать и наоборот:

zoo_eats.merge(zoo)

Это то же самое, что и:

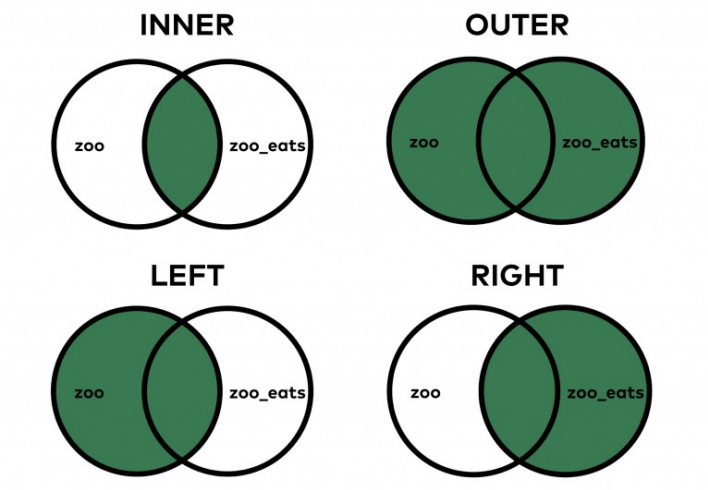
zoo.merge(zoo_eats)

Разница будет лишь в порядке колонок в финальной таблице.

Способы объединения: inner, outer, left, right

Базовый метод merge довольно прост. Но иногда к нему нужно добавить несколько параметров.

Один из самых важных вопросов — как именно нужно объединять эти таблицы. В SQL есть 4 типа JOIN.



В случае с merge в pandas в теории это работает аналогичным образом.

При выборе INNER JOIN (вид по умолчанию в SQL и pandas) объединяются только те значения, которые можно найти в обеих таблицах. В случае же с OUTER JOIN объединяются все значения, даже если некоторые из них есть только в одной таблице.

Конкретный пример: в zoo_eats нет значения lion. А в zoo нет значения giraffe. По умолчанию использовался метод INNER, поэтому и львы, и жирафы пропали из таблицы. Но бывают случаи, когда нужно, чтобы все значения оставались в объединенном DataFrame. Этого можно добиться следующим образом:

zoo.merge(zoo_eats, how='outer')

```
In [11]: zoo.merge(zoo_eats, how = 'outer')
```

Out[11]:

	animal	uniq_id	water_need	food
0	elephant	1001.0	500.0	vegetables
1	elephant	1002.0	600.0	vegetables
2	elephant	1003.0	550.0	vegetables
3	tiger	1004.0	300.0	meat
4	tiger	1005.0	320.0	meat
5	tiger	1006.0	330.0	meat
6	tiger	1007.0	290.0	meat
7	tiger	1008.0	310.0	meat
8	zebra	1009.0	200.0	vegetables
9	zebra	1010.0	220.0	vegetables
10	zebra	1011.0	240.0	vegetables
11	zebra	1012.0	230.0	vegetables
12	zebra	1013.0	220.0	vegetables
13	zebra	1014.0	100.0	vegetables
14	zebra	1015.0	80.0	vegetables
15	lion	1016.0	420.0	NaN
16	lion	1017.0	600.0	NaN
17	lion	1018.0	500.0	NaN
18	lion	1019.0	390.0	NaN
19	kangaroo	1020.0	410.0	vegetables
20	kangaroo	1021.0	430.0	vegetables
21	kangaroo	1022.0	410.0	vegetables
22	giraffe	NaN	NaN	vegetables

В этот раз львы и жирафы вернулись. Но поскольку вторая таблица не предоставила конкретных данных, то вместо значения ставится пропуск (NaN).

Логичнее всего было бы оставить в таблице львов, но не жирафов. В таком случае будет три типа еды: vegetables, meat и NaN (что, фактически, значит, «информации нет»). Если же в таблице останутся жирафы, это может запутать,

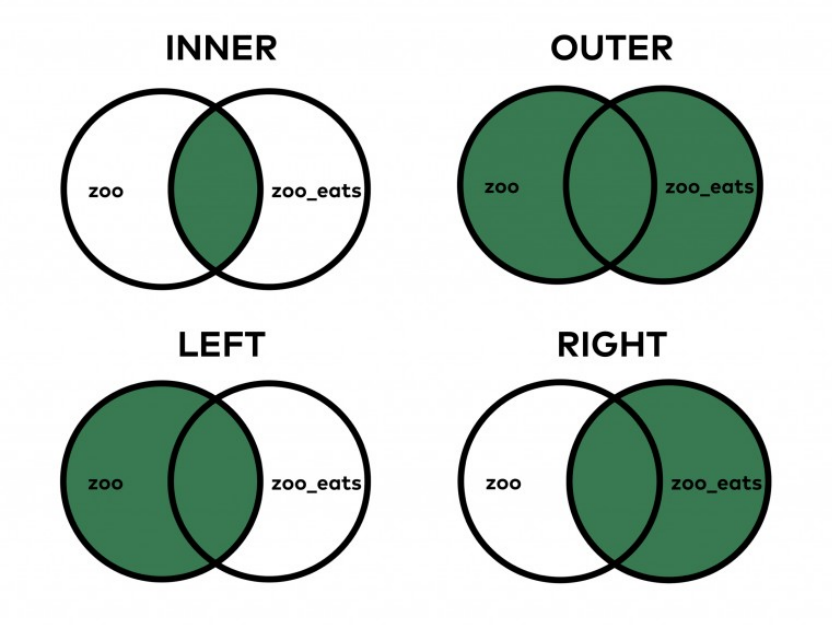
потому что в зоопарке-то этого вида животных все равно нет. Поэтому следует воспользоваться параметром how='left' при объединении.

Вот так:

zoo.merge(zoo_eats, how='left')

Теперь в таблице есть вся необходимая информация, и ничего лишнего. how = 'left' заберет все значения из левой таблицы (zoo), но из правой (zoo_eats) использует только те значения, которые есть в левой.

Еще раз взглянем на типы объединения:



Примечание: «Какой метод merge является самым безопасным?» — самый распространенный вопрос. Но на него нет однозначного ответа. Нужно решать в зависимости от конкретной задачи.

5 Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм решения задачи.
3. Запрограммировать полученный алгоритм.
4. Провести тестирование полученной программы.

6 Форма отчета о работе

Лабораторная работа № ___
Номер учебной группы _____
Фамилия, инициалы учащегося: _____
Дата выполнения работы: _____
Тема работы:
_____
Цель работы: _____

Оснащение работы: _____
Результат выполнения работы: _____
_____

7 Контрольные вопросы и задания

1. Что такое объединение наборов данных?
2. Какие бывают виды объединения наборов данных?
3. Для чего применяется объединение данных?

8 Рекомендуемая литература

Плас, Дж. В. Python для сложных задач. Наука о данных и машинное обучение / Дж.В. Плас. – СПб: Питер, 2018.

Прохоренок, Н.А. Python 3. Самое необходимое / Н.А Прохоренок, В.А. Дронов – СПб.: БВХ-Петербург, 2016.