## 6. Обработка дубликатов

Когда мы говорим о дубликатах (дублированных записях), то представляем себе две одинаковых строки с идентичной информацией. Действительно, это дубликаты, и мы ещё разберём, как от них избавляться.

Но бывают и менее явные дубликаты: скажем, две якобы разные категории публикаций. К примеру, «Политика» и «Политическая ситуация». При агрегировании данных эти две категории будут отражены как разные подмножества набора данных, хотя на самом деле все публикации о политике нужно объединить в одну категорию. При всех возможностях Pandas важно изучать данные глазами и обдумывать их. Иначе вас ожидает масса неприятных сюрпризов.

Помимо вредного влияния дубликатов на результат, их наличие затягивает процесс анализа. Большое количество повторов неоправданно раздувает размер таблицы, а значит, увеличивает время обработки данных.

Грубые дубликаты — повторы — обнаруживают методом **duplicated()** (англ. duplicated, «удвоенный»). Он возвращает Series со значением True при наличии дубликатов, и False, когда их нет.

print(df.duplicated())

Чтобы посчитать количество дубликатов в наборе данных, нужно вызвать метод sum():

print(df.duplicated().sum())

Для удаления дубликатов есть метод **drop\_duplicates()** (от англ. drop duplicates, «сбросить дубликаты»):

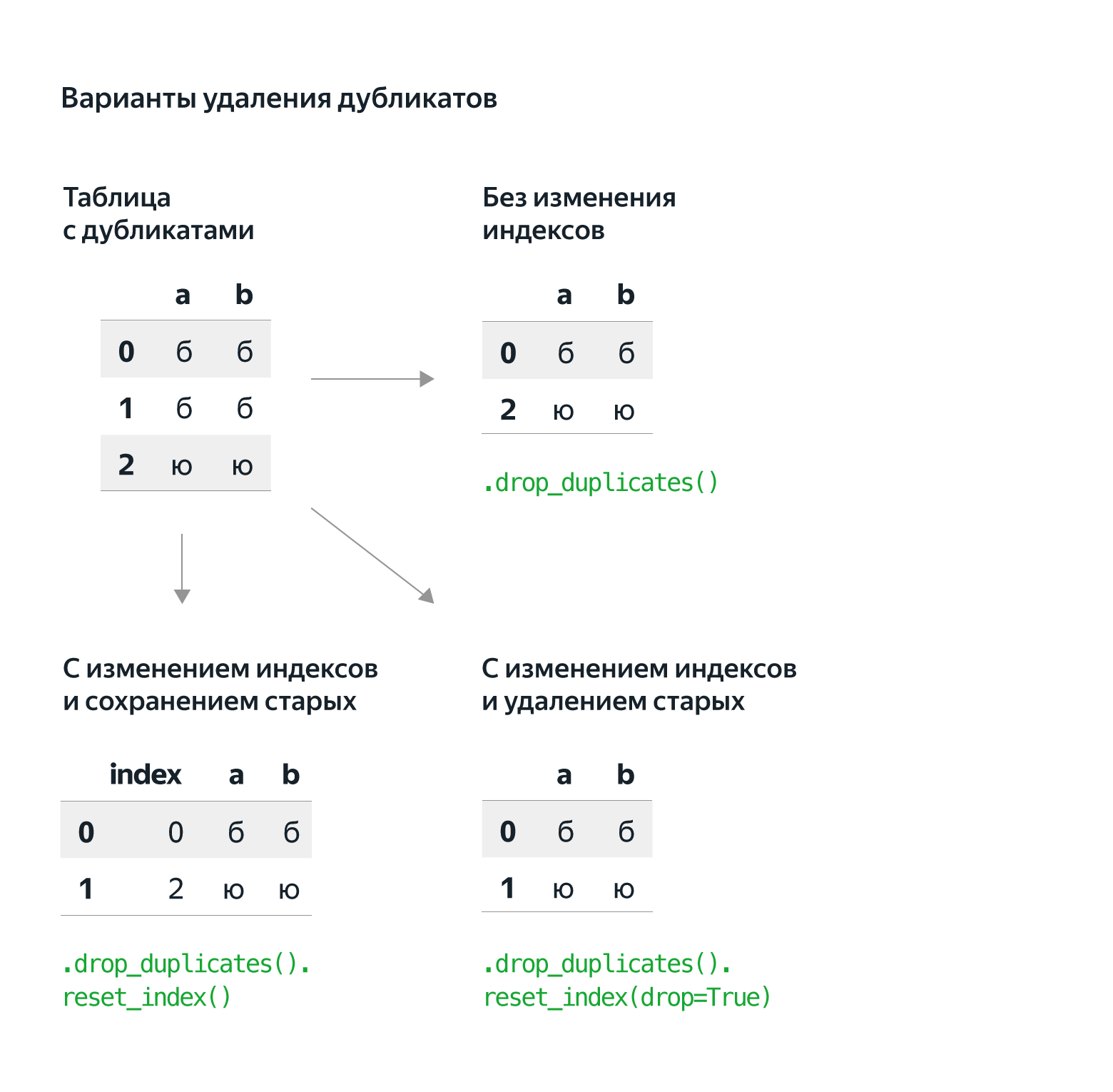
df.drop\_duplicates(inplace = True)

При вызове метода drop\_duplicates() вместе с повторяющимися строками удаляются их индексы.

Последовательность индексов нарушается: после 0 следует 2 и т.д.

Поэтому вызов drop\_duplicates() соединяют в цепочку с вызовом метода **reset\_index()** (англ. *reset index*, «переопределить индексы»). Тогда создаётся новый DataFrame, где старые индексы превращаются в обычный столбец под названием *index*, а индексы всех строк снова следуют в естественном порядке.

Если же мы не хотим создавать новый столбец *index*, то при вызове reset\_index() передается аргумент drop со значением True. Все индексы переписываются в порядке возрастания, без пропусков.



Вот такой код сохраняет в переменной df таблицу, очищенную от дубликатов, с новой индексацией.

df = df.drop\_duplicates().reset\_index(drop=True)

Дубликаты в названиях категорий отследить сложнее, но всё-таки можно. Для просмотра всех уникальных значений в столбце используется метод **unique()** (англ. *unique*, «уникальный»).

Например, в 2018 году рейтинг первой ракетки мира по версии ATP обновлялся 17 раз. Но спортсменов с самым большим в мире профессионального тенниса рейтингом гораздо меньше. Если применить ко второму столбцу метод unique(), возвращаются только три имени.

import pandas as pd rating = ['date','name','points']

players = [ ['2018.01.01', 'Рафаэль Надаль',10645], ['2018.01.08', 'Рафаэль Надаль', 10600], ['2018.01.29', 'Рафаэль Надаль', 9760], ['2018.02.19', 'Роджер Федерер', 10105], ['2018.03.05', 'Роджер Федерер', 10060], ['2018.03.19', 'Роджер Федерер', 9660], ['2018.04.02', 'Рафаэль Надаль', 8770], ['2018.06.18', 'Roger Federer', 8920], ['2018.06.25', 'Рафаэль Надаль', 8770], ['2018.07.16', 'Рафаэль Надаль', 9310], ['2018.08.13', 'Рафаэль Надаль', 10220], ['2018.08.20', 'Рафаэль Надаль', 10040], ['2018.09.10', 'Рафаэль Надаль', 8760], ['2018.10.08', 'Рафаэль Надаль', 8260], ['2018.10.15', 'Рафаэль Надаль', 7660], ['2018.11.05', 'Новак Джокович', 8045], ['2018.11.19', 'Новак Джокович', 9045] ]

tennis = pd.DataFrame(data = players, columns= rating)

print(tennis)

print(tennis['name'].unique())

array(['Рафаэль Надаль', 'Роджер Федерер', 'Roger Federer',

'Новак Джокович'], dtype=object)

Кроме имени Роджера Федерера на русском языке, вернулось его же имя латиницей. Это тоже дубликат. Нужно заменить этот артефакт правильным именем на русском языке.

Такие задачи решает метод **replace()** (англ. *replace*, «заменить»), где первый аргумент — текущее значение, а второй — новое, нужное.

tennis['name'] = tennis['name'].replace('Roger Federer', 'Роджер Федерер')

print(tennis)

Теперь всё хорошо. Настало время разобраться с дубликатами в таблице данных Яндекс.Музыки.