## 5. Описательная статистика

Меломан оказался таким же последовательным поклонником определённых жанров, как и те, кому хватает пяти композиций в день (исследования на остальных меломанах это подтверждают). Значит, по своим вкусам и поведению наша аудитория однородна, оценивать её «счастье» можно показателями «середнячков». Чтобы представить себе среднестатистического пользователя, понадобится описательная статистика.

Из её показателей для количественного описания данных в нашей простой задаче нужны четыре меры — максимум, минимум, медиана и среднее.

Наибольшее и наименьшее обычно вычисляют только по одному признаку. Например, можно получить минимальное и максимальное значение количества прослушанных секунд композиции (столбец 'total\_play\_seconds'). Для поиска максимума вызывают метод max():

print(df['total\_play\_seconds'].max())

4707.422018

Самый длинный трек звучал почти полтора часа. Интересно, какую композицию слушали так долго.

Запросим из df строку с максимальным значением, прибегнув к логической индексации с условием df['total\_play\_seconds'] == df['total\_play\_seconds'].max():

print(df[df['total\_play\_seconds'] == df['total\_play\_seconds'].max()])

Минимальное значение — самый короткий трек — ищут методом min(). Понятно, что в Яндекс.Музыке показателей меньше нуля не бывает: если пользователь пролистнул трек, значение 'total\_play\_seconds' равно 0. Нам интересно установить композиции, которые слушали хоть и недолго, но не пропустили сразу.

Вот как мы создаём выборку без пропущенных треков и находим в ней минимальное значение:

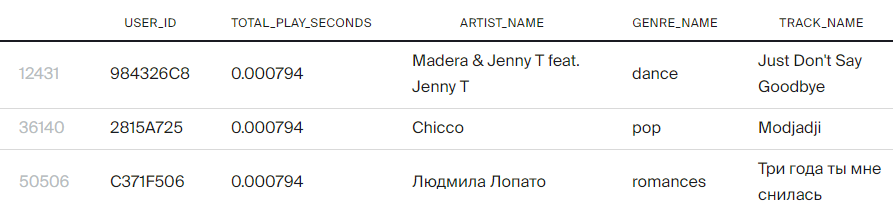
df\_min = df[df['total\_play\_seconds'] != 0]

print(df\_min['total\_play\_seconds'].min())

0.000794

Получим названия композиции, которую пропустили быстрее всего. Как и при поиске максимума, воспользуемся условием и логической индексацией:

print(df\_min[df\_min['total\_play\_seconds'] == df\_min['total\_play\_seconds'].min()])

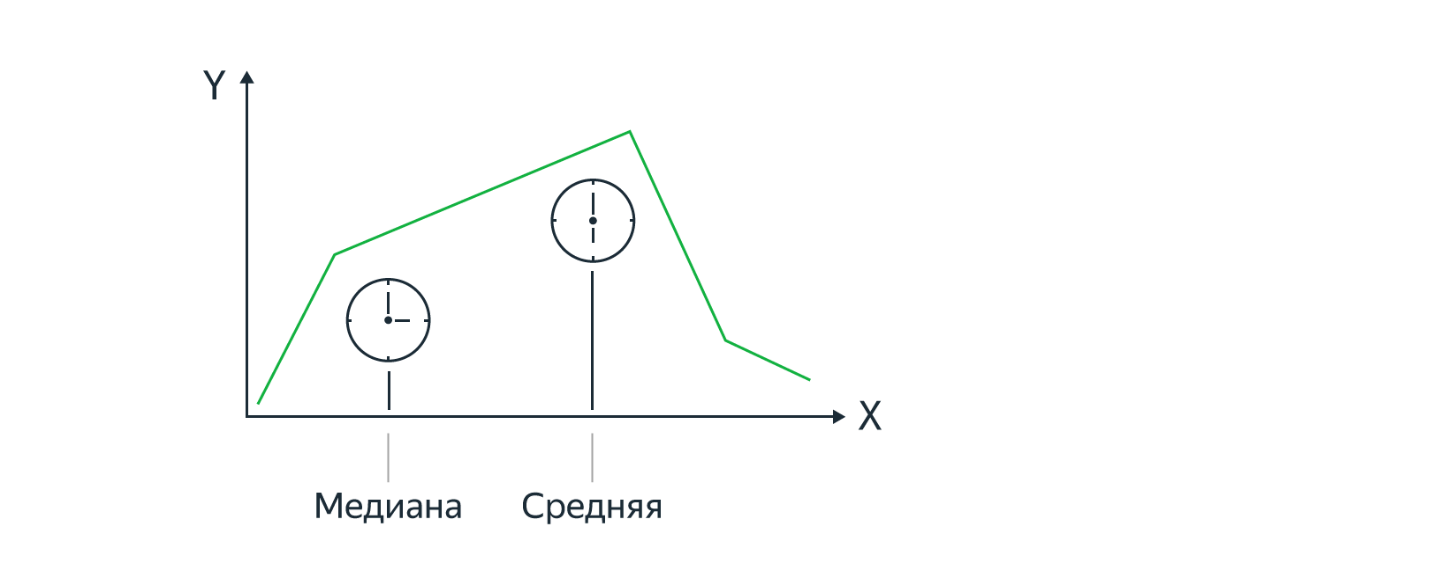


Результат вывода — три трека, которые соотвествует минимальному значению.

На основе полученных данных можно сделать вывод, что время прослушивания треков находится в диапазоне от 0,000794 до 4707,422018 секунд, не включая пропущенные.

Это знание пригодится, чтобы разобраться с медианой и средним арифметическим.

Среднее и медиана оценивают значения в центре выборки. Если тех, кто слушает долго, столько же, сколько тех, кто слушает мало — среднее подойдет. Но когда есть оторванные от основной массы лидеры, слушающие музыку по 8 часов, их результаты сильно смещают значение среднего вверх. Вот почему оценивать предпочтения широкого круга потребителей лучше медианой.



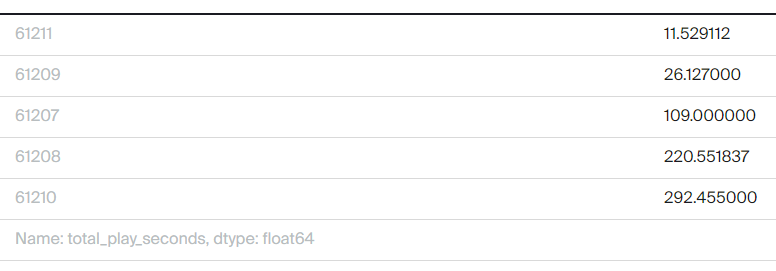
В геометрии медиана делит фигуру на две равные по площади. В статистике она делит выборку пополам: в одной половине значения меньше медианного, в другой больше. Логично, что для определения медианы список обязательно должен быть отсортирован — либо по возрастанию, либо по убыванию.

Когда количество значений нечётное, медиана будет равна тому значению, которое оказалось ровно посередине отсортированного набора. Если же количество данных чётное, то медиана рассчитывается как среднее арифметическое двух соседних чисел в середине набора.

Для примера возьмем 5 последних строк в таблице. Обратимся к столбцу *'total\_play\_seconds'* и отсортируем его:

df\_stat\_1 = df.tail()

print(df\_stat\_1['total\_play\_seconds'].sort\_values())

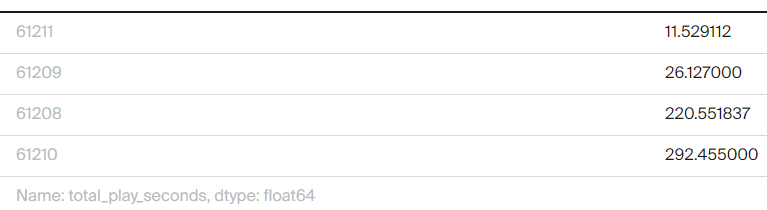


Значение в середине равно 109 — это и будет медиана.

Возьмем 4 последних строки в таблице, обратимся к столбцу *'total\_play\_seconds'*:

df\_stat = df.tail(4)

print(df\_stat['total\_play\_seconds'].sort\_values())



Возьмем два значения в середине -- 26.127 и 220.551837. Посчитаем среднее:

(26.127 + 220.551837) / 2

123.3394185

В Pandas есть метод **median()**, который считает медиану. По аналогии с min() и max() его можно применять ко всей таблице, к отдельному столбцу или к сгруппированным данным.

Сравним результаты с теми, которые мы получили вручную:

print(df\_stat['total\_play\_seconds'].median())

124.3138585

Теперь применим медиану ко всем значениям времени прослушивания в нашей таблице, исключив нулевые:

df\_drop\_null = df[df['total\_play\_seconds'] != 0] print(df\_drop\_null['total\_play\_seconds'].median())

37.0

Чтобы убедиться, что лидеры действительно смещают средний показатель вверх, найдём среднее арифметическое всех этих значений методом mean() (англ. mean, «среднее»):

print(df\_drop\_null['total\_play\_seconds'].mean())

109.81197057131848

Убедились? А теперь — за дело!