ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Московский институт электроники и математики им. А.Н. Тихонова**

Рустамова Дарина Дмитриевна, группа БИВ205,

**ОТЧЕТ**

**ПО ДОМАШНЕЙ РАБОТЕ 2**

по дисциплине «Python в науке о данных»

Тема: «Создание информационной основы для приложений»

Бригада 12

Дата сдачи отчета: 10.05.2021

Москва 2021 г.

Разбор функции ***pandas.pivot\_table***

Создает сводную таблицу как DataFrame.

**pandas.pivot\_table(*data*, *values=None*, *index=None*, *columns=None*, *aggfunc='mean'*, *fill\_value=None*, *margins=False*, *dropna=True*, *margins\_name='All'*, *observed=False*)**

Параметры:

**Data** – DataFrame

**values** – столбцы для агрегирования (необязательно),

**index** – столбец, группировщик

**columns** – столбец, группировщик

**aggfunc** – функция, либо лист функций, словарь (по умолчанию np.mean)

**fill\_value** – значение, замещающее пропущенные значения (в результате агрегирования)

**margins** – добавляет новую строку \ столбец (например, для общих итогов) (по умолчанию False)

**dropna** – не включает колонки, которые содержат все строки NaN (по умолчанию False)

**margins\_name** – название строки \ столбца при условии, если margins = True. То есть название этой\го строки \ столбца

Пример:

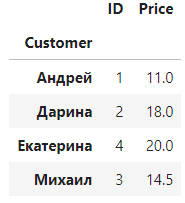
Есть таблица customers в формате xlsx: 

Чтобы работать с этой таблицей, нужно преобразовать ее в DataFrame с помощью строчки:

df **=** pd**.**read\_excel("customers.xlsx")

Благодаря коду ниже, можно сгруппировать данные по имени customer с помощью параметра index. Таким образом, мы узнали ID каждого покупателя, среднюю стоимость треков, которые они приобрели.

pd**.**pivot\_table(df, index**=**"Customer")



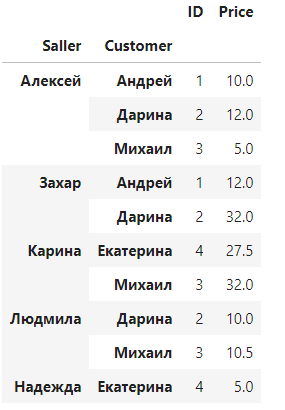
Теперь можно сгруппировать таблицу по трекам, чтобы узнать стоимость каждого:

pd**.**pivot\_table(df, index**=**"Track")



В index так же можно использовать список, чтобы группировать по порядку. Благодаря коду ниже, можно узнать каких покупателей обслужил каждый продавец:

pd**.**pivot\_table(df, index**=**["Saller", "Customer"])



Так как ID бесполезен в данном случае, выделить конкретные столбцы можно с помощью параметра values.

pd**.**pivot\_table(df, index**=**["Saller", "Customer"], values**=**["Price"])



Можно узнать сколько дохода принес каждый трек, используя функцию affunc и np.sum для суммы:

pd**.**pivot\_table(df, index**=**["Track"], values**=**["Price"], aggfunc**=**np**.**sum)



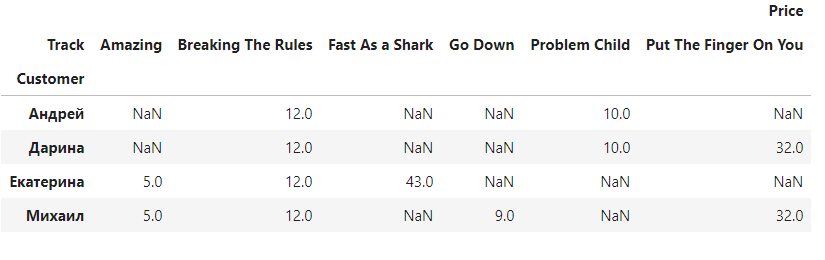
С помощью margins можно узнать общий доход, который принесли все треки:

pd**.**pivot\_table(df, index**=**["Track"], values**=**["Price"], aggfunc**=**np**.**sum, margins**=True**)



Предположим, мы захотели узнать, сколько денег покупатель потратил на каждый из треков. Для этого можно воспользоваться параметром columns:

pd**.**pivot\_table(df, index**=**["Customer"], values**=**["Price"], columns**=**["Track"])



Как можно заменить NaN? В этом случае поможет параметр fill\_value, который заменяет пропущенные значение введенным параметром.

pd**.**pivot\_table(df, index**=**["Customer"], values**=**["Price"], columns**=**["Track"], fill\_value**=**0)

