Лабораторная работа №2

Выполнила Мосева Алеся Сергеевна БВТ2001

О задании

Задание состоит из двух разделов, посвященных работе с табличными данными с помощью библиотеки pandas и визуализации с помощью matplotlib. В каждом разделе вам предлагается выполнить несколько заданий.

Задание направлено на освоение jupyter notebook (будет использоваться в дальнейших заданиях), библиотекам pandas и matplotlib.

0. Введение

Сейчас мы находимся в jupyter-ноутбуке (или ipython-ноутбуке). Это удобная среда для написания кода, проведения экспериментов, изучения данных, построения визуализаций и других нужд, не связанных с написаем production-кода.

Ноутбук состоит из ячеек, каждая из которых может быть либо ячейкой с кодом, либо ячейкой с текстом размеченным и неразмеченным. Текст поддерживает markdown-разметку и формулы в Latex.

Для работы с содержимым ячейки используется *режим редактирования* (*Edit mode*, включается нажатием клавиши **Enter** после выбора ячейки), а для навигации между ячейками искользуется *командный режим* (*Command mode*, включается нажатием клавиши **Esc**). Тип ячейки можно задать в командном режиме либо с помощью горячих клавиш (**y** to code, **m** to markdown, **r** to edit raw text), либо в меню *Cell -> Cell type*.

После заполнения ячейки нужно нажать *Shift + Enter*, эта команда обработает содержимое ячейки: проинтерпретирует код или сверстает размеченный текст.

In [ ]:
```
# ячейка с кодом, при выполнении которой появится output
2 + 2
```

Out[ ]: 4

А это ___ячейка с текстом___.

Ячейка с неразмеченым текстом.

Попробуйте создать свои ячейки, написать какой-нибудь код и текст какой-нибудь формулой.

In [ ]:
```
print('Hello World!')
```

```
Hello World!
```

Здесь находится ~~не~~большая заметка о используемом языке разметки Markdown. Он позволяет:

0. Составлять упорядоченные списки

1. # Делать

заголовки

разного уровня

3. Выделять *текст* ~~при~~ **необходимости**
4. Добавлять ссылки

- Составлять неупорядоченные списки

Делать вставки с помощью LaTex:

$$\begin{cases} x = 16 \sin^3(t) \\ y = 13 \cos(t) - 5 \cos(2t) - 2 \cos(3t) - \cos(4t) \\ t \in [0, 2\pi] \end{cases}$$

1. Табличные данные и Pandas

Pandas — удобная библиотека для работы с табличными данными в Python, если данных не слишком много и они помещаются в оперативную память вашего компьютера. Несмотря на неэффективность реализации и некоторые проблемы, библиотека стала стандартом в анализе данных. С этой библиотекой мы сейчас и познакомимся.

Основной объект в pandas это DataFrame, представляющий собой таблицу с именованными колонками различных типов, индексом (может быть многоуровневым). DataFrame можно создавать, считывая таблицу из файла или задавая вручную из других объектов.

В этой части потребуется выполнить несколько небольших заданий. Можно пойти двумя путями: сначала изучить материалы, а потом приступить к заданиям, или же разбираться "по ходу". Выбирайте сами.

Материалы:

1. Pandas за 10 минут из официального руководства
2. Документация (стоит обращаться, если не понятно, как вызывать конкретный метод)
3. Примеры использования функционала

Многие из заданий можно выполнить несколькими способами. Не существуют единственно верного, но попробуйте максимально задействовать арсенал pandas и ориентируйтесь на простоту и понятность вашего кода. Мы не будем подсказывать, что нужно использовать для решения конкретной задачи, попробуйте находить необходимый функционал сами (название метода чаще всего очевидно). В помощь вам документация, поиск и stackoverflow.

In [ ]:
```
import pandas as pd
%pylab inline  #import almost all we need
```

```
UsageError: unrecognized arguments: #import almost all we need
```

Данные можно скачать отсюда.

In [ ]:
```
df = pd.read_csv('./data.csv')
df.head()
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

1. Откройте файл с таблицей (не забудьте про её формат). Выведите последние 10 строк.

Посмотрите на данные и скажите, что они из себя представляют, сколько в таблице строк, какие столбцы?

Столбцы: количество, название продукта, состав, цена Количество строк: 1834

In [ ]:
```
df.tail(10)
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price
4612	1831	1	Carnitas Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	$9.25
4613	1831	1	Chips	NaN	$2.15
4614	1831	1	Bottled Water	NaN	$1.50
4615	1832	1	Chicken Soft Tacos	[Fresh Tomato Salsa, [Rice, Cheese, Sour Cream]]	$8.75
4616	1832	1	Chips and Guacamole	NaN	$4.45
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

2. [0.25 баллов] Ответьте на вопросы:

1. Сколько заказов попало в выборку? 1834 (из пункта 1 видно)
2. Сколько уникальных категорий товара было куплено? 50

In [ ]:
```
df['item_name'].value_counts().count()
```

Out[ ]:
```
50
```

3. [0.25 баллов] Есть ли в данных пропуски? В каких колонках?

Есть, в колонке choice_description

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
order_id                0
quantity                0
item_name               0
choice_description    1246
item_price              0
dtype: int64
```

Заполните пропуски пустой строкой для строковых колонок и нулём для числовых.

In [ ]:
```
df = df.fillna('')
df.isnull().sum()
```

Out[ ]:
```
order_id              0
quantity              0
item_name             0
choice_description    0
item_price            0
dtype: int64
```

In [ ]:
```
df.head()
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa		$2.39
1	1	1	Izze	[Clementine]	$3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa		$2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98

4. [0.5 баллов] Посмотрите внимательнее на колонку с ценой товара. Какого она типа? Создайте новую колонку так, чтобы в ней цена была числом.

Для этого попробуйте применить функцию-преобразование к каждой строке вашей таблицы (для этого есть соответствующая функция).

In [ ]:
```
print(df['item_price'].dtype)
print()

df['new_item_price'] = df['item_price'].apply(lambda x: x[1:]).astype(float)

df['new_item_price'] = df['item_price'].apply(lambda x: float(x[1:]))
print()
print(df['new_item_price'].dtype)

df.head()
```

```
object


float64
```

Out[ ]:

	order_id	quantity	item_name	choice_description	item_price	new_item_price
0	1	1	Chips and Fresh Tomato Salsa		$2.39	2.39
1	1	1	Izze	[Clementine]	$3.39	3.39
2	1	1	Nantucket Nectar	[Apple]	$3.39	3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa		$2.39	2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98	16.98

Какая средняя/минимальная/максимальная цена у товара?

In [ ]:
```
print('Минимальная {}'.format(df['new_item_price'].min()))
print('Максимальная {}'.format(df['new_item_price'].max()))
```

```
Минимальная 1.09
Максимальная 44.25
```

Удалите старую колонку с ценой.

In [ ]:
```
df.drop(columns=['item_price'], inplace=True)
df.head()
```

Out[ ]:

	order_id	quantity	item_name	choice_description	new_item_price
0	1	1	Chips and Fresh Tomato Salsa		2.39
1	1	1	Izze	[Clementine]	3.39
2	1	1	Nantucket Nectar	[Apple]	3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa		2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	16.98

5. [0.25 баллов] Какие 5 товаров были самыми дешёвыми и самыми дорогими? (по choice_description)

Для этого будет удобно избавиться от дубликатов и отсортировать товары. Не забудьте про количество товара.

In [ ]:
```
df_without_dublicates = df.drop_duplicates(['choice_description'])
df_without_dublicates = df_without_dublicates.sort_values(['new_item_price'])
print('Дешевые')
print(df_without_dublicates[['choice_description', 'new_item_price']].head(5))
print('Дорогие')
print(df_without_dublicates[['choice_description', 'new_item_price']].tail(5))
```

```
Дешевые
    choice_description  new_item_price
117  [Diet Dr. Pepper]            1.09
126        [Coca Cola]            1.09
28        [Dr. Pepper]            1.09
346         [Lemonade]            1.25
263             [Coke]            1.25
Дорогие
                                     choice_description  new_item_price
2954  [[Tomatillo-Green Chili Salsa (Medium), Roaste...           22.96
4427  [Fresh Tomato Salsa, [Rice, Pinto Beans, Sour ...           23.50
1753  [Tomatillo Green Chili Salsa, [Fajita Vegetabl...           23.50
3334  [Tomatillo Green Chili (Medium), [Rice, Black ...           26.07
409   [[Fresh Tomato Salsa (Mild), Tomatillo-Green C...           32.94
```

6. [0.5 баллов] Сколько раз клиенты покупали больше 1 Chicken Bowl (item_name)?

In [ ]:
```
df[(df['item_name'] == 'Chicken Bowl') & (df['quantity'] > 1)].quantity.count()
```

Out[ ]:
```
33
```

7. [0.5 баллов] Какой средний чек у заказа? Сколько в среднем товаров покупают?

Если необходимо провести вычисления в терминах заказов, то будет удобно сгруппировать строки по заказам и посчитать необходимые статистики.

In [ ]:
```
#средний чек: общая сумма на количество заказов
round(df.groupby('order_id')['new_item_price'].sum().mean(), 2)
```

Out[ ]:
```
18.81
```

In [ ]:
```
#в среднем товаров: общая сумма количества товаров на количество заказов
round(df.groupby('order_id')['new_item_price'].count().mean())
```

Out[ ]:
```
3
```

8. [0.25 баллов] Сколько заказов содержали ровно 1 товар?

In [ ]:
```
tmp = df[df['quantity'] == 1]
tmp.groupby('order_id')[['quantity']].filter(lambda x: len(x) == 1).sum()
```

Out[ ]:
```
quantity    102
dtype: int64
```

9. [0.25 баллов] Какая самая популярная категория товара?

In [ ]:
```
df.groupby('item_name')['quantity'].sum().sort_values().tail(1)
```

Out[ ]:
```
item_name
Chicken Bowl    761
Name: quantity, dtype: int64
```

13. [0.75 баллов] Создайте новый DateFrame из матрицы, созданной ниже. Назовите колонки index, column1, column2 и сделайте первую

колонку индексом.

In [ ]:
```
import numpy as np

data = np.random.rand(10, 3)

data = pd.DataFrame(data[:, 1:], columns=['column1', 'column2'], index = data[:,
data
```

Out[ ]:

	column1	column2
0.065087	0.414712	0.309484
0.503284	0.394315	0.478907
0.077378	0.702183	0.920099
0.738728	0.194719	0.589452
0.306219	0.579254	0.090720
0.286227	0.052325	0.741929
0.487599	0.129692	0.103906
0.609698	0.838538	0.290756
0.630188	0.533571	0.871725
0.501390	0.503449	0.738402

Сохраните DataFrame на диск в формате csv без индексов и названий столбцов.

In [ ]:
```
from pathlib import Path
filepath = Path('./result.csv')
data.to_csv(filepath)
```

2. Визуализации и matplotlib

При работе с данными часто неудобно делать какие-то выводы, если смотреть на таблицу и числа в частности, поэтому важно уметь визуализировать данные. В этом разделе мы этим и займёмся.

У matplotlib, конечно, же есть документация с большим количеством примеров, но для начала достаточно знать про несколько основных типов графиков:

- plot — обычный поточечный график, которым можно изображать кривые или отдельные точки;
- hist — гистограмма, показывающая распределение некоторое величины;
- scatter — график, показывающий взаимосвязь двух величин;
- bar — столбцовый график, показывающий взаимосвязь количественной величины от категориальной.

В этом задании вы попробуете построить каждый из них. Не менее важно усвоить базовые принципы визуализаций:

- на графиках должны быть подписаны оси;

- у визуализации должно быть название;
- если изображено несколько графиков, то необходима поясняющая легенда;
- все линии на графиках должны быть чётко видны (нет похожих цветов или цветов, сливающихся с фоном);
- если отображена величина, имеющая очевидный диапазон значений (например, проценты могут быть от 0 до 100), то желательно масштабировать ось на весь диапазон значений (исключением является случай, когда вам необходимо показать малое отличие, которое незаметно в таких масштабах).

In [ ]:
```
%matplotlib inline  # нужно для отображения графиков внутри ноутбука
```

```
UsageError: unrecognized arguments: # нужно для отображения графиков внутри ноу
тбука
```

На самом деле мы уже импортировали matplotlib внутри %pylab inline в начале задания.

Работать мы будем с той же выборкой покупкок. Добавим новую колонку с датой покупки.

In [ ]:
```
import datetime
import random
import pandas as pd
import matplotlib.pyplot as plt

start = datetime.datetime(2018, 1, 1)
end = datetime.datetime(2018, 1, 31)
delta_seconds = int((end - start).total_seconds())

dates = pd.DataFrame(index=df.order_id.unique())
dates['date'] = [
    (start + datetime.timedelta(seconds=random.randint(0, delta_seconds))).strft
    for _ in range(df.order_id.nunique())]

# если DataFrame с покупками из прошлого заказа называется не df, замените на ва
df['date'] = df.order_id.map(dates['date'])

df.head(5)
```

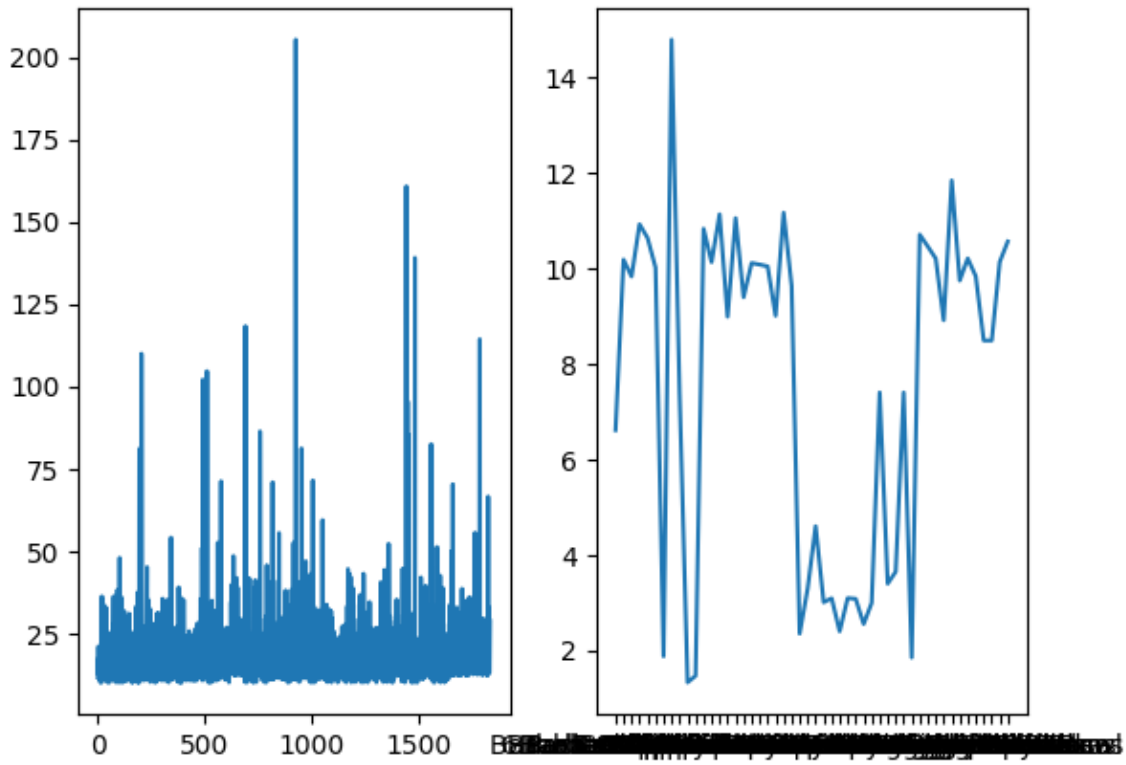
Out[ ]:

	order_id	quantity	item_name	choice_description	new_item_price	date
0	1	1	Chips and Fresh Tomato Salsa		2.39	2018-01-11
1	1	1	Izze	[Clementine]	3.39	2018-01-11
2	1	1	Nantucket Nectar	[Apple]	3.39	2018-01-11
3	1	1	Chips and Tomatillo-Green Chili Salsa		2.39	2018-01-11
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	16.98	2018-01-05

1. [1 балл] Постройте гистограмму распределения сумм покупок и гистограмму средних цен отдельных видов продуктов item_name.

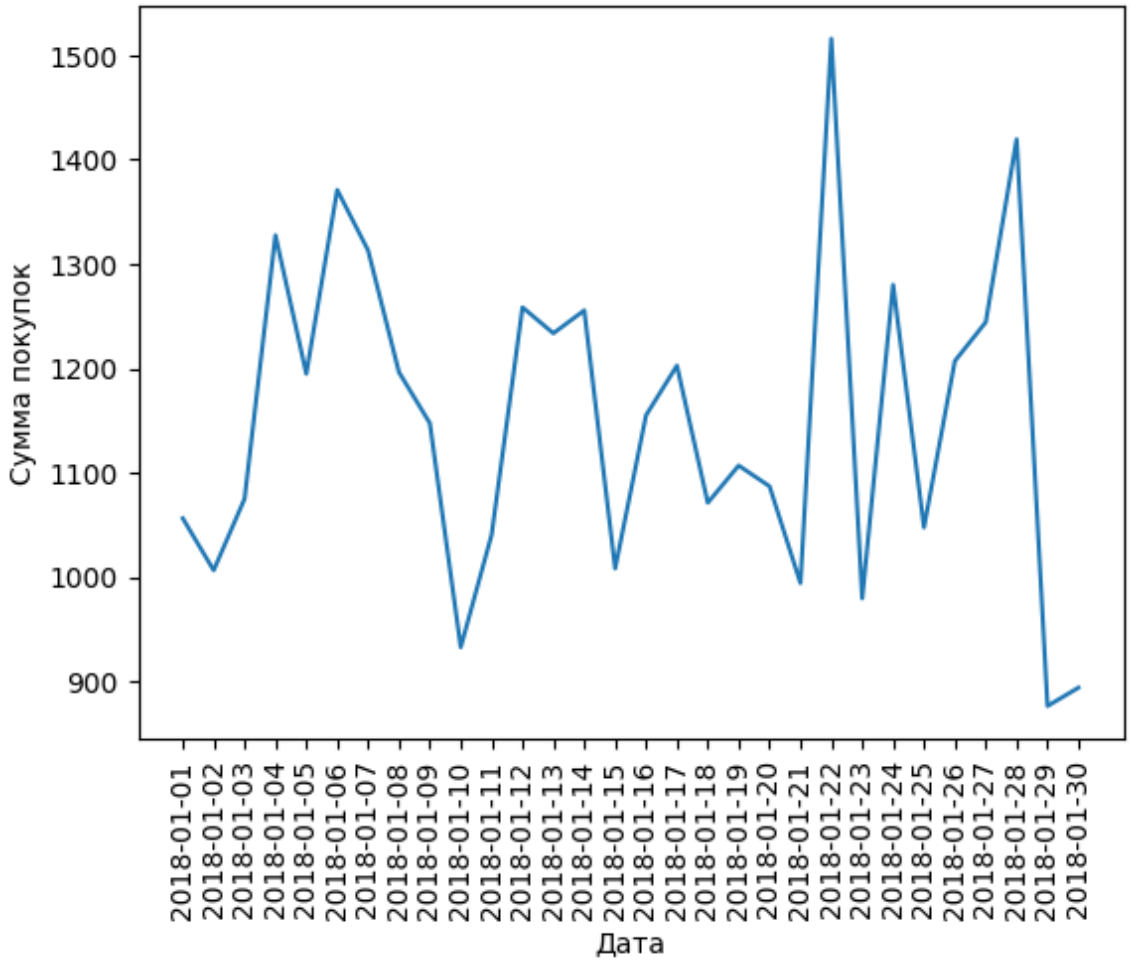
Изображайте на двух соседних графиках. Для этого может быть полезен subplot.

In [ ]:
```
plt1=df.groupby('order_id').agg({'new_item_price': 'sum'}).reset_index()
plt.subplot(1, 2, 1)
plt.plot(plt1.order_id,plt1.new_item_price)
plt2=df.groupby('item_name').agg({'new_item_price': 'mean'}).reset_index()
plt.subplot(1, 2, 2)
plt.plot(plt2.item_name,plt2.new_item_price)
plt.show()
```



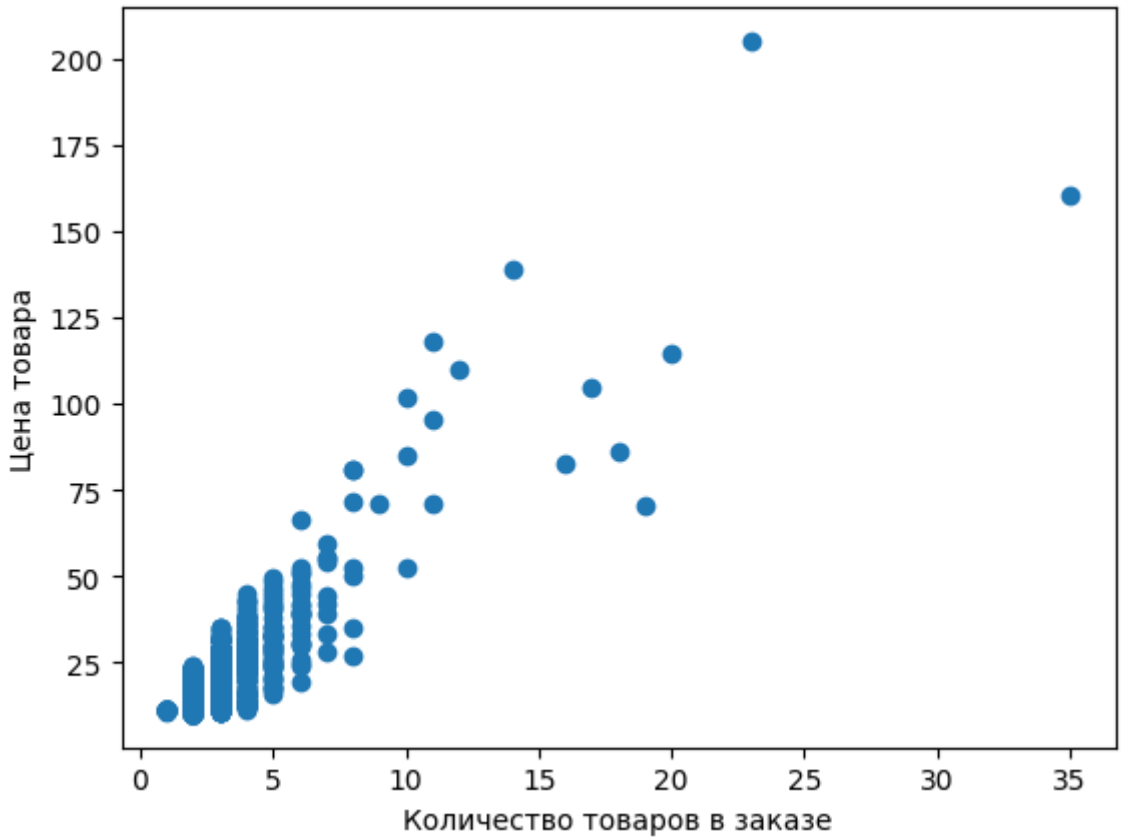
2. [1 балл] Постройте график зависимости суммы покупок от дней.

In [ ]:
```
plot=df.groupby('date').agg({'new_item_price': 'sum'}).reset_index()
plt.plot(plot.date,plot.new_item_price)
plt.ylabel('Сумма покупок')
plt.xlabel('Дата')
plt.xticks(rotation=90)
plt.show()
```



3. [1 балл] Постройте график зависимости денег за товар от купленного количества (scatter plot).

In [ ]:
```
x_data = df.groupby('order_id')['quantity'].sum().values
y_data = df.groupby('order_id')['new_item_price'].sum().values
plt.scatter(x_data,y_data)
plt.xlabel('Количество товаров в заказе')
plt.ylabel('Цена товара')
plt.show()
```



Сохраните график в формате pdf (так он останется векторизованным).

In [ ]:
```
plt.savefig("./plot.pdf")
```

```
<Figure size 640x480 with 0 Axes>
```

Кстати, существует надстройка над matplotlib под названием seaborn. Иногда удобнее и красивее делать визуализации через неё.