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 [63]:
```
# ячейка с кодом, при выполнении которой появится output
2 + 2
```

Out[63]: 4

А это ___ячейка с текстом___.

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

In [64]:
```
3 + 4
```

Out[64]: 7

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 [65]:
```
%pylab inline
# import almost all we need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib
```

Данные можно скачать отсюда.

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?

In [66]:
```
import pandas as pd
df = pd.read_csv('data.csv')
df.tail(10)
# В таблице содержатся данные о номере заказа, количестве, названии блюда, составе и цене блюда. Всего в таблице 4621 строка и 5 столбцов.
```

Out[66]:

	order_id	quantity	item_name	choice_description	item_price
4612	1831	1	Carnitas Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Rice,...	$9.25

	order_id	quantity	item_name	choice_description	item_price
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

2. [0.25 баллов] Ответьте на вопросы:

1. Сколько заказов попало в выборку?
2. Сколько уникальных категорий товара было куплено? (item_name)

In [67]:
```
#1
#df = df.iloc[:]
df['quantity'].sum()
```

Out[67]: 4972

In [68]:
```
#2 Список уникальных товаров
#df.groupby('item_name').sum()
```

In [69]:
```
#2 Количество уникальных товаров
len(df.groupby('item_name').sum())
```

```
<ipython-input-69-03ffb33c5c21>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  len(df.groupby('item_name').sum())
```

Out[69]: 50

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

In [70]:
```
# Переменная True указывает на пропуск в данных
#pd.isnull(df)
```

In [71]:
```
# 1246 пропусков в колонке 'choice_description'
df.isnull().sum()
```

Out[71]:
```
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 [72]:
```
df.select_dtypes(include=['int64']).fillna(0)
df.select_dtypes(include=['object']).fillna('')
```

Out[72]:

	item_name	choice_description	item_price
0	Chips and Fresh Tomato Salsa		$2.39
1	Izze	[Clementine]	$3.39
2	Nantucket Nectar	[Apple]	$3.39
3	Chips and Tomatillo-Green Chili Salsa		$2.39
4	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98
...	...	...	...
4617	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75
4618	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75
4619	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25
4620	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75
4621	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75

4622 rows × 3 columns

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 [73]:
```
# функцию to_numeric невозможно реализовать из-за знака $ в стообце. Поэтому создадим функцию, которая убирает $.
df['new_price']= df['item_price']
def convert_currency(value):
    new_value = value.replace('$','')
    return float(new_value)
df['new_price'] = df['new_price'].apply(convert_currency)
pd.to_numeric(df['new_price'])
df
```

Out[73]:

	order_id	quantity	item_name	choice_description	item_price	new_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	$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	NaN	$2.39	2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans...	$16.98	16.98
...	...	...	...	...	...	...
4617	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Black Beans, Sour ...	$11.75	11.75
4618	1833	1	Steak Burrito	[Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...	$11.75	11.75
4619	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$11.25	11.25

	order_id	quantity	item_name	choice_description	item_price	new_price
4620	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Lettu...	$8.75	8.75
4621	1834	1	Chicken Salad Bowl	[Fresh Tomato Salsa, [Fajita Vegetables, Pinto...	$8.75	8.75

Какая средняя/минимальная/максимальная цена у товара?

In [74]:
```
df['new_price'].mean()
```

Out[74]: 7.464335785374297

In [75]:
```
df['new_price'].min()
```

Out[75]: 1.09

In [76]:
```
df['new_price'].max()
```

Out[76]: 44.25

Удалите старую колонку с ценой.

In [77]:
```
# удаление колонки item_price
df = df.drop(columns=['item_price'])
df
```

Out[77]:

	order_id	quantity	item_name	choice_description	new_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
...	...	...	...	...	...
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

4622 rows × 5 columns

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 [78]:
```
# 5 самых дешёвых товаров
df2=df
df2['price_item1']=df2['new_price']/df2['quantity']
df2.sort_values(['price_item1', 'choice_description'], ascending=[True, True],inplace=True)
df2.drop_duplicates(subset=['choice_description'], keep='first').head(5)
```

Out[78]:

	order_id	quantity	item_name	choice_description	new_price	price_item1
126	55	1	Canned Soda	[Coca Cola]	1.09	1.09
162	73	2	Canned Soda	[Diet Coke]	2.18	1.09
117	51	1	Canned Soda	[Diet Dr. Pepper]	1.09	1.09
28	14	1	Canned Soda	[Dr. Pepper]	1.09	1.09
51	23	2	Canned Soda	[Mountain Dew]	2.18	1.09

In [79]:
```
# 5 самых дорогих товаров
df3=df
df3['price_item1']=df3['new_price']/df3['quantity']
df3.sort_values(['price_item1', 'choice_description'], ascending=[True, True],inplace=True)
df3.drop_duplicates(subset=['choice_description'], keep='last').tail(5)
```

Out[79]:

	order_id	quantity	item_name	choice_description	new_price	price_item1
2442	970	1	Barbacoa Salad Bowl	[Tomatillo Green Chili Salsa, [Rice, Fajita Ve...	11.89	11.89
281	123	2	Steak Salad Bowl	[Tomatillo Red Chili Salsa, [Black Beans, Chee...	23.78	11.89
3208	1283	1	Barbacoa Salad Bowl	[Tomatillo Red Chili Salsa, [Black Beans, Chee...	11.89	11.89
613	253	2	Steak Salad Bowl	[Tomatillo Red Chili Salsa, [Fajita Vegetables...	23.78	11.89
2804	1114	1	Steak Salad Bowl	[Tomatillo Red Chili Salsa, [Rice, Black Beans...	11.89	11.89

6. [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 [80]:
```
# средний чек каждого заказа
#df.groupby('order_id').mean()
df[['order_id','price_item1']].groupby('order_id').mean()
```

Out[80]:

	price_item1
order_id	
1	2.890000
2	8.490000
3	6.335000
4	10.500000
5	6.850000
...	...
1830	11.500000

order_id	price_item1
1831	4.300000
1832	6.600000
1833	11.750000
1834	9.583333

In [81]:
```
# средний чек всех заказов
df.loc[:,"price_item1"].mean()
```

Out[81]: 7.084424491562094

In [82]:
```
# сколько в среднем товаров покупают
df.loc[:,"quantity"].mean()
```

Out[82]: 1.0757247944612722

7. [0.25 баллов] Сколько заказов содержали ровно 1 товар?

In [83]:
```
len(df.loc[df['quantity'] == 1])
```

Out[83]: 4355

8. [0.75 баллов] Создайте новый DateFrame из матрицы, созданной ниже. Назовите колонки index, column1, column2 и сделайте первую колонку индексом.

In [84]:
```
from pandas import DataFrame
data = np.random.rand(10, 3)
df_new = DataFrame(data, columns = ['index','column1','column2'])
df_new
```

Out[84]:

	index	column1	column2
0	0.221113	0.197206	0.044305
1	0.598774	0.568779	0.770873
2	0.173049	0.697996	0.871628
3	0.951327	0.629383	0.086814
4	0.742485	0.730647	0.961178
5	0.288643	0.503420	0.246645
6	0.348778	0.742052	0.002940
7	0.633775	0.148872	0.569200
8	0.889331	0.408497	0.741924
9	0.282206	0.799923	0.616927

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

In [85]:
```
df_new.to_csv("DataFrame.csv")
```

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).
- сетка на графике помогает оценить значения в точках на глаз, это обычно полезно, поэтому лучше ее отрисовывать.

In [86]:
```
%matplotlib inline
# нужно для отображения графиков внутри ноутбука
import matplotlib.pyplot as plt
```

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

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

In [87]:
```
import datetime

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))).strftime('%Y-%m-%d')
    for _ in range(df.order_id.nunique())]

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

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.

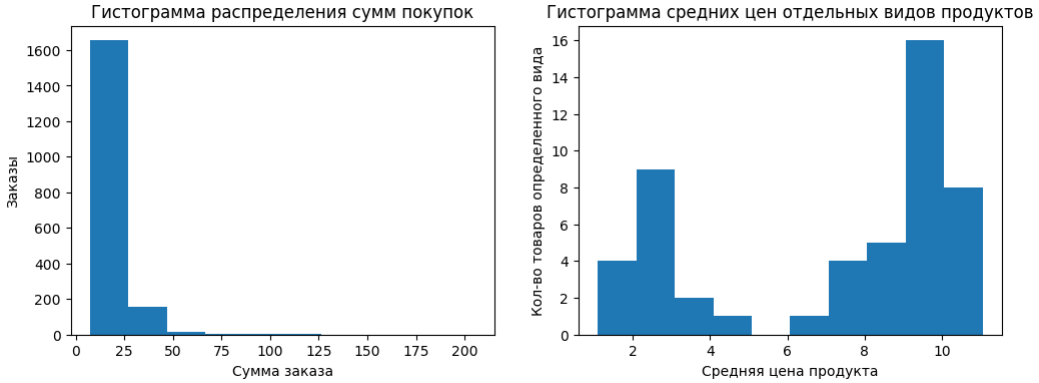
In [88]:
```
order_price = df.groupby('order_id').sum()['price_item1']
avg_price   = df.groupby('item_name').mean()['price_item1']
fig, (axes1, axes2) = plt.subplots(nrows=1, ncols=2, figsize=(12, 4))

axes1.hist(order_price)
axes1.set_title('Гистограмма распределения сумм покупок')
axes1.set_xlabel('Сумма заказа')
axes1.set_ylabel('Заказы')

axes2.hist(avg_price)
axes2.set_title('Гистограмма средних цен отдельных видов продуктов')
axes2.set_xlabel('Средняя цена продукта')
axes2.set_ylabel('Кол-во товаров определенного вида')

plt.show()
```

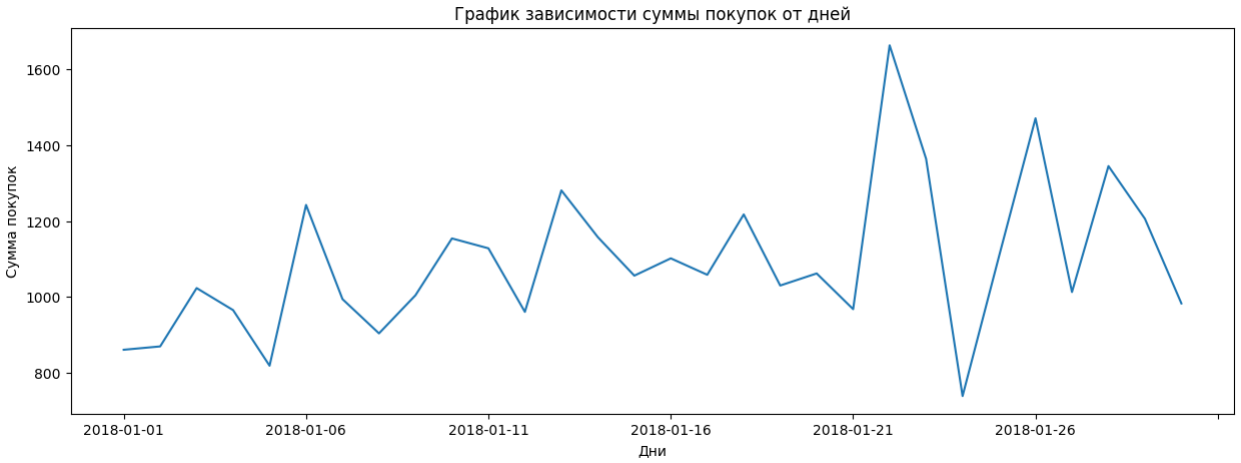
```
<ipython-input-88-1dc0e413748e>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  order_price = df.groupby('order_id').sum()['price_item1']
<ipython-input-88-1dc0e413748e>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  avg_price   = df.groupby('item_name').mean()['price_item1']
```



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

In [89]:
```
order_price= df.groupby('date').sum()['price_item1']
plt.figure(figsize=(15,5))
order_price.plot()
plt.title('График зависимости суммы покупок от дней')
plt.xlabel('Дни')
plt.ylabel('Сумма покупок')
plt.show()
```

```
<ipython-input-89-042ca684bd12>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  order_price= df.groupby('date').sum()['price_item1']
```



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

In [90]:
```
quantity = df.groupby('order_id').sum()['quantity']
price_item = df.groupby('order_id').sum()['price_item1']
avg_item=price_item/quantity


plt.scatter(quantity, avg_item)
plt.title('Зависимость денег за товар от купленного количества')
plt.xlabel('Количество купленного товара')
plt.ylabel('Цена за 1 шт товара')
plt.show()
```

```
<ipython-input-90-932c49467f16>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  quantity = df.groupby('order_id').sum()['quantity']
<ipython-input-90-932c49467f16>:2: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will
default to False. Either specify numeric_only or select only columns which should be valid for the function.
  price_item = df.groupby('order_id').sum()['price_item1']
```

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

In [91]:
```
fig.savefig("scatter.pdf")
```

Еще одна билиотека для визуализации: seaborn. Это настройка над matplotlib, иногда удобнее и красивее делать визуализации через неё.