1. ℹ️ Создайте Series с данными о продажах за неделю [120, 150, 90, 200, 180, 250, 300] (Пн-Вс) и выполните анализ

```
# загружаем библиотеки


import pandas as pd


import numpy as np


import matplotlib.pyplot as plt


from matplotlib.ticker import FuncFormatter


sales = pd.Series([120, 150, 90, 200, 180, 250, 300],
                  index = ["ПН", "ВТ", "СР", "ЧТ", "ПТ", "СБ", "ВС"],
                  name = "Продажи") # Создан Series с данными о продажах за неделю
```

1A. ✅ Выведите дни с продажами выше среднего

```
sales # Проверка продаж
```

```
ПН    120
ВТ    150
СР     90
ЧТ    200
ПТ    180
СБ    250
ВС    300
Name: Продажи, dtype: int64
```

```
sales.mean() # Вывод среднего арифметического значения. Получаем среднее по продаж.
```

```
np.float64(184.28571428571428)
```

```
above_avg = sales[sales > sales.mean()]  # Продажи сравниваем со средним значением и выводим всё, что выше
```

```
print('Дни с продажами выше среднего')
print(above_avg)
```

```
Дни с продажами выше среднего
ЧТ    200
СБ    250
ВС    300
Name: Продажи, dtype: int64
```

1Б. ✅ Найдите процентное соотношение продаж в выходные (Сб, Вс) к общим продажам

```
# новая переменная с общей суммой продаж
all_sum_values = sales.sum()


# новая переменная с фильтром продаж по выходным
weeekend_sales = sales[['СБ', 'ВС']].sum()


# процентное соотношение продаж в выходные (Сб, Вс) к общим продажам
percentage = (weeekend_sales/all_sum_values)*100


print('Процентное соотношение продаж в выходные (Сб, Вс) к общим продажам:', '→',percentage,'←')
```

```
Процентное соотношение продаж в выходные (Сб, Вс) к общим продажам: → 42.63565891472868 ←
```

2. ℹ️ Очистите DataFrame с "грязными" данными

```
data = {
    'Продукт': ['Яблоки', 'Груши', 'Бананы', np.nan, 'Апельсины', ''],
    'Цена': [100, 120, '90', 150, 'не число', 200],
    'Количество': [10, 15, 20, None, 25, 30]
}
```

2А. ✅ Создание DataFrame с "грязными" данными

```
df = pd.DataFrame(data)


print(df)
```

```
     Продукт      Цена  Количество
0     Яблоки       100        10.0
1      Груши       120        15.0
2     Бананы        90        20.0
3        NaN       150         NaN
4  Апельсины  не число        25.0
5                 200        30.0
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Продукт	Цена	Количество
0	Яблоки	100	10.0
1	Груши	120	15.0
2	Бананы	90	20.0
3	Неизвестно	150	NaN
4	Апельсины	не число	25.0
5	Неизвестно	200	30.0

2Б. ✅ Замените пустые строки и NaN в столбце 'Продукт' на 'Неизвестно'

```
df ['Продукт'] = df ['Продукт'].replace(['', np.nan], 'Неизвестно') # Заменяем значение NaN в колонке Проду
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Продукт	Цена	Количество
0	Яблоки	100	10.0
1	Груши	120	15.0
2	Бананы	90	20.0
3	Неизвестно	150	NaN
4	Апельсины	не число	25.0
5	Неизвестно	200	30.0

```
df.info
```

```
<bound method DataFrame.info of       Продукт   Цена  Количество
0      Яблоки  100.0        10.0
1       Груши  120.0        15.0
2      Бананы   90.0        20.0
3  Неизвестно  150.0         NaN
4   Апельсины    NaN        25.0
5  Неизвестно  200.0        30.0>
```

2Б. ✅ Преобразуйте столбец 'Цена' в числовой формат, ошибки замените на NaN

```
df ['Цена'] = pd.to_numeric(df ['Цена'], errors = 'coerce')
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Продукт	Цена	Количество
0	Яблоки	100.0	10.0
1	Груши	120.0	15.0
2	Бананы	90.0	20.0
3	Неизвестно	150.0	NaN
4	Апельсины	NaN	25.0
5	Неизвестно	200.0	30.0

2Г. ✅ Заполните пропущенные значения в 'Количество' средним значением

```
mean_qty = df['Количество'].mean() # вывод количетсво
```

```
mean_qty
```

```
np.float64(20.0)
```

```
df['Количество'] = df['Количество'].fillna(mean_qty)
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Продукт	Цена	Количество
0	Яблоки	100.0	10.0

	Продукт	Цена	Количество
1	Груши	120.0	15.0
2	Бананы	90.0	20.0
3	Неизвестно	150.0	20.0
4	Апельсины	NaN	25.0
5	Неизвестно	200.0	30.0

2Д. ✅ Удалите строки, где цена не указана (NaN)

```
df = df.dropna(subset = 'Цена')
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Продукт	Цена	Количество
0	Яблоки	100.0	10.0
1	Груши	120.0	15.0
2	Бананы	90.0	20.0
3	Неизвестно	150.0	20.0
5	Неизвестно	200.0	30.0

3. ℹ️ Проанализируйте простой набор данных о фруктах и их ценах.

```
data = {
    'Фрукт': ['Яблоко', 'Банан', 'Апельсин', 'Груша', 'Киви'],
    'Цена': [80, 60, 90, 70, 120],
    'Наличие_кг': [15, 8, 12, 5, 3]
}
```

```
df = pd.DataFrame(data)
```

```
df # check
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг
0	Яблоко	80	15
1	Банан	60	8
2	Апельсин	90	12
3	Груша	70	5
4	Киви	120	3

3А. ✅ Выведите только фрукты дороже 75 рублей

```
fruits = df[df['Цена'] > 75]


fruits # check
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг
0	Яблоко	80	15
2	Апельсин	90	12
4	Киви	120	3

3Б. ✅ Посчитайте общую стоимость всех имеющихся фруктов

```
df['Стоимость'] = df['Цена'] = df['Наличие_кг']


df #check
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг	Стоимость
0	Яблоко	15	15	15
1	Банан	8	8	8
2	Апельсин	12	12	12
3	Груша	5	5	5
4	Киви	3	3	3

```
total = df['Стоимость'].sum()
```

```
total
```

```
np.int64(43)
```

3В. ✅ Добавьте новый фрукт (Манго за 100 руб., 7 кг)

```
New_fruits = pd.DataFrame({
    'Фрукт': ['Манго'],
    'Цена': [100],
    'Наличие_кг': [7],
    'Стоимость': [700]
})
```

```
df #check
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг	Стоимость
0	Яблоко	15	15	15
1	Банан	8	8	8
2	Апельсин	12	12	12
3	Груша	5	5	5
4	Киви	3	3	3

```
df = pd.concat([df, New_fruits], ignore_index = True) # Объединение двух df
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг	Стоимость
0	Яблоко	15	15	15
1	Банан	8	8	8
2	Апельсин	12	12	12
3	Груша	5	5	5
4	Киви	3	3	3
5	Манго	100	7	700
6	Манго	100	7	700

3Г. ✅ Отсортируйте фрукты по цене (от дешевых к дорогим)

```
df.sort_values('Цена')
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Фрукт	Цена	Наличие_кг	Стоимость
4	Киви	3	3	3
3	Груша	5	5	5
1	Банан	8	8	8
2	Апельсин	12	12	12
0	Яблоко	15	15	15
5	Манго	100	7	700
6	Манго	100	7	700

4. ℹ️ Исследование заёмщиков

Заказчик - Кредитный отдел банка.

Согласно документации к данным:

- `children` — количество детей в семье;
- `days_employed` — общий трудовой стаж в днях;
- `dob_years` — возраст клиента в годах;
- `education` — уровень образования клиента;
- `education_id` — идентификатор уровня образования;
- `family_status` — семейное положение;
- `family_status_id` — идентификатор семейного положения;
- `gender` — пол клиента;
- `income_type` — тип занятости;
- `debt` — имел ли задолженность по возврату кредитов;
- `total_income` — ежемесячный доход;
- `purpose` — цель получения кредита.

ℹ️ Дать ответы на вопросы:

- 4А. ❓ `Средняя зарплата по типу занятости` ;
- 4Б. ❓ `Процент заемщиков с высшим образованием` ;
- 4В. ❓ `Частота кредитов на покупку жилья` ;
- 4Г. ❓ `Зависимость между семейным положением и задолженностью` ;
- 4Д. ❓ `Есть ли зависимость между количеством детей и возвратом кредита в срок?` ;
- 4Е. ❓ `Есть ли зависимость между уровнем дохода и возвратом кредита в срок?` .

✅ Обзор данных

```
df = pd.read_csv("data.csv")
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Unnamed: 0	children	days_employed	dob_years	education	education_id	family_status
0	0	1	8437.673028	42	высшее	0	женат / замужем

	Unnamed: 0	children	days_employed	dob_years	education	education_id	family_status
1	1	1	4024.803754	36	среднее	1	женат / замужем
2	2	0	5623.422610	33	Среднее	1	женат / замужем
3	3	3	4124.747207	32	среднее	1	женат / замужем
4	4	0	1630.019381	53	среднее	1	гражданский брак
...	...	...	...	...	...	...	...
21390	21390	1	4529.316663	43	среднее	1	гражданский брак
21391	21391	0	1630.019381	67	среднее	1	женат / замужем
21392	21392	1	2113.346888	38	среднее	1	гражданский брак
21393	21393	3	3112.481705	38	среднее	1	женат / замужем
21394	21394	2	1984.507589	40	среднее	1	женат / замужем

21395 rows × 13 columns

```
df.head(2) # output 2 lines
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Unnamed: 0	children	days_employed	dob_years	education	education_id	family_status	f
0	0	1	8437.673028	42	высшее	0	женат / замужем	C
1	1	1	4024.803754	36	среднее	1	женат / замужем	C

```
df.info #check general info
```

```
<bound method DataFrame.info of       Unnamed: 0  children  days_employed  dob_years education  education_
0              0         1    8437.673028         42    высшее             0
1              1         1    4024.803754         36   среднее             1
2              2         0    5623.422610         33   Среднее             1
```

```
3              3         3    4124.747207         32   среднее            1
4              4         0    1630.019381         53   среднее            1
...          ...       ...            ...        ...       ...          ...
21390      21390         1    4529.316663         43   среднее            1
21391      21391         0    1630.019381         67   среднее            1
21392      21392         1    2113.346888         38   среднее            1
21393      21393         3    3112.481705         38   среднее            1
21394      21394         2    1984.507589         40   среднее            1

          family_status  family_status_id gender income_type  debt  \
0       женат / замужем                 0      F   сотрудник     0
1       женат / замужем                 0      F   сотрудник     0
2       женат / замужем                 0      M   сотрудник     0
3       женат / замужем                 0      M   сотрудник     0
4      гражданский брак                 1      F   пенсионер     0
...                 ...               ...    ...         ...   ...
21390  гражданский брак                 1      F   компаньон     0
21391   женат / замужем                 0      F   пенсионер     0
21392  гражданский брак                 1      M   сотрудник     1
21393   женат / замужем                 0      M   сотрудник     1
21394   женат / замужем                 0      F   сотрудник     0

        total_income                     purpose
0      253875.639453               покупка жилья
1      112080.014102     приобретение автомобиля
2      145885.952297               покупка жилья
3      267628.550329  дополнительное образование
4      158616.077870             сыграть свадьбу
...              ...                         ...
21390  224791.862382           операции с жильем
21391  155999.806512        сделка с автомобилем
21392   89672.561153                недвижимость
21393  244093.050500  на покупку своего автомобиля
21394   82047.418899       на покупку автомобиля

[21395 rows x 13 columns]>
```

```
df.describe()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}

</style>
```

	Unnamed: 0	children	days_employed	dob_years	education_id	family_status_
count	21395.000000	21395.000000	19284.000000	21395.000000	21395.000000	21395.000000
mean	10697.000000	0.470437	2224.761599	43.284272	0.817107	0.974246
std	6176.348841	0.754534	2106.974155	12.576030	0.548879	1.421282
min	0.000000	-1.000000	24.141633	0.000000	0.000000	0.000000
25%	5348.500000	0.000000	927.984311	33.000000	1.000000	0.000000
50%	10697.000000	0.000000	1630.019381	42.000000	1.000000	0.000000
75%	16045.500000	1.000000	2747.876441	53.000000	1.000000	1.000000

	Unnamed: 0	children	days_employed	dob_years	education_id	family_status_
max	21394.000000	5.000000	18388.949901	75.000000	4.000000	4.000000

```
df['education'].unique() # checking for duplicates
```

```
array(['высшее', 'среднее', 'Среднее', 'СРЕДНЕЕ', 'ВЫСШЕЕ',
       'неоконченное высшее', 'начальное', 'Высшее',
       'НЕОКОНЧЕННОЕ ВЫСШЕЕ', 'Неоконченное высшее', 'НАЧАЛЬНОЕ',
       'Начальное', 'Ученая степень', 'УЧЕНАЯ СТЕПЕНЬ', 'ученая степень'],
      dtype=object)
```

```
df['education'].value_counts()  # подсчет количества людей и образование
```

```
education
среднее                13653
высшее                  4698
СРЕДНЕЕ                  770
Среднее                  705
неоконченное высшее      666
ВЫСШЕЕ                   271
Высшее                   268
начальное                250
Неоконченное высшее       47
НЕОКОНЧЕННОЕ ВЫСШЕЕ       29
НАЧАЛЬНОЕ                 17
Начальное                 15
ученая степень             4
Ученая степень             1
УЧЕНАЯ СТЕПЕНЬ             1
Name: count, dtype: int64
```

```
for i in ['education', 'family_status', 'gender', 'income_type', 'purpose']: # цикл для проверки на уникльн
    print(df[i].unique())
```

```
['высшее' 'среднее' 'Среднее' 'СРЕДНЕЕ' 'ВЫСШЕЕ' 'неоконченное высшее'
 'начальное' 'Высшее' 'НЕОКОНЧЕННОЕ ВЫСШЕЕ' 'Неоконченное высшее'
 'НАЧАЛЬНОЕ' 'Начальное' 'Ученая степень' 'УЧЕНАЯ СТЕПЕНЬ'
 'ученая степень']
['женат / замужем' 'гражданский брак' 'вдовец / вдова' 'в разводе'
 'не женат / не замужем']
['F' 'M' 'XNA']
['сотрудник' 'пенсионер' 'компаньон' 'госслужащий' 'безработный'
 'предприниматель' 'студент' 'в декрете']
['покупка жилья' 'приобретение автомобиля' 'дополнительное образование'
 'сыграть свадьбу' 'операции с жильем' 'образование'
 'на проведение свадьбы' 'покупка жилья для семьи' 'покупка недвижимости'
 'покупка коммерческой недвижимости' 'покупка жилой недвижимости'
 'строительство собственной недвижимости' 'недвижимость'
 'строительство недвижимости' 'на покупку подержанного автомобиля'
 'на покупку своего автомобиля' 'операции с коммерческой недвижимостью'
 'строительство жилой недвижимости' 'жилье'
 'операции со своей недвижимостью' 'автомобили' 'заняться образованием'
 'сделка с подержанным автомобилем' 'получение образования' 'автомобиль'
 'свадьба' 'получение дополнительного образования' 'покупка своего жилья'
 'операции с недвижимостью' 'получение высшего образования'
 'свой автомобиль' 'сделка с автомобилем' 'профильное образование'
 'высшее образование' 'покупка жилья для сдачи' 'на покупку автомобиля'
 'ремонт жилью' 'заняться высшим образованием']
```

```
df = df.drop(columns = ['Unnamed: 0']) # удаление ненужного столбца
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21395 entries, 0 to 21394
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   children          21395 non-null  int64  
 1   days_employed     19284 non-null  float64
 2   dob_years         21395 non-null  int64  
 3   education         21395 non-null  object 
 4   education_id      21395 non-null  int64  
 5   family_status     21395 non-null  object 
 6   family_status_id  21395 non-null  int64  
 7   gender            21395 non-null  object 
 8   income_type       21395 non-null  object 
 9   debt              21395 non-null  int64  
 10  total_income      19284 non-null  float64
 11  purpose           21395 non-null  object 
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

```
df.describe()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education_id	family_status_id	de
count	21395.000000	19284.000000	21395.000000	21395.000000	21395.000000	21395.0000
mean	0.470437	2224.761599	43.284272	0.817107	0.974246	0.081000
std	0.754534	2106.974155	12.576030	0.548879	1.421282	0.272842
min	-1.000000	24.141633	0.000000	0.000000	0.000000	0.000000
25%	0.000000	927.984311	33.000000	1.000000	0.000000	0.000000
50%	0.000000	1630.019381	42.000000	1.000000	0.000000	0.000000
75%	1.000000	2747.876441	53.000000	1.000000	1.000000	0.000000
max	5.000000	18388.949901	75.000000	4.000000	4.000000	1.000000

✅ Предобработка данных

Что входит в перечень задач по очистке данных?

- Очистка явных и неявных дубликатов ;
- Аномальные значения ;

- Пропустки ;
- Опечатки .

```
df.duplicated().sum()
```

```
np.int64(0)
```

```
# Удалить дубликаты, если они есть:
df = df.drop_duplicates()
```

```
# Уникальные значения в исполнение цикла:

for i in ['education', 'family_status', 'gender', 'income_type', 'purpose']:
    print(df[i].unique())
```

```
['высшее' 'среднее' 'неоконченное высшее' 'начальное' 'ученая степень']
['женат / замужем' 'гражданский брак' 'вдовец / вдова' 'в разводе'
 'не женат / не замужем']
['F' 'M']
['сотрудник' 'пенсионер' 'компаньон' 'госслужащий' 'безработный'
 'предприниматель' 'студент' 'в декрете']
['покупка жилья' 'приобретение автомобиля' 'дополнительное образование'
 'сыграть свадьбу' 'операции с жильем' 'образование'
 'на проведение свадьбы' 'покупка жилья для семьи' 'покупка недвижимости'
 'покупка коммерческой недвижимости' 'покупка жилой недвижимости'
 'строительство собственной недвижимости' 'недвижимость'
 'строительство недвижимости' 'на покупку подержанного автомобиля'
 'на покупку своего автомобиля' 'операции с коммерческой недвижимостью'
 'строительство жилой недвижимости' 'жилье'
 'операции со своей недвижимостью' 'автомобили' 'заняться образованием'
 'сделка с подержанным автомобилем' 'получение образования' 'автомобиль'
 'свадьба' 'получение дополнительного образования' 'покупка своего жилья'
 'операции с недвижимостью' 'получение высшего образования'
 'свой автомобиль' 'сделка с автомобилем' 'профильное образование'
 'высшее образование' 'покупка жилья для сдачи' 'на покупку автомобиля'
 'ремонт жилью' 'заняться высшим образованием']
```

```
# Группировка по параметру "Дети"
print(df.groupby('children')["children"].count())
```

```
children
-1       47
 0    14107
 1     4809
 2     2052
 3      330
 4       41
 5        9
Name: children, dtype: int64
```

```
# Группировка по параметру "Дети"

df['children'].value_counts()
```

```
children
 0    14107
 1     4809
 2     2052
```

```
 3      330
-1       47
 4       41
 5        9
Name: count, dtype: int64
```

```
# Сортировка результата выдачи "1." от большего к меньшему по параметру "Кол-во детей":

df["children"] = df["children"].replace(-1, 1)
```

```
# Процентное соотношение детей < 18 лет

print (df[df['dob_years']<18]['dob_years'].count()/df['dob_years'].count()*100)
```

```
0.4673989249824725
```

```
ndf = df[df['dob_years'] >= 18]
```

```
df.describe()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education_id	family_status_id	de
count	21395.000000	19284.000000	21395.000000	21395.000000	21395.000000	21395.0000
mean	0.474831	2224.761599	43.284272	0.817107	0.974246	0.081000
std	0.751776	2106.974155	12.576030	0.548879	1.421282	0.272842
min	0.000000	24.141633	0.000000	0.000000	0.000000	0.000000
25%	0.000000	927.984311	33.000000	1.000000	0.000000	0.000000
50%	0.000000	1630.019381	42.000000	1.000000	0.000000	0.000000
75%	1.000000	2747.876441	53.000000	1.000000	1.000000	0.000000
max	5.000000	18388.949901	75.000000	4.000000	4.000000	1.000000

```
# Привести все данные в нижний регистр

df['education'] = df['education'].str.lower()
```

```
# Уникальные значения по гендеру

df['gender'].value_counts()
```

```
gender
F    14142
M     7252
Name: count, dtype: int64
```

```
# Удаляем из гендера значение "Н/Д"

df = df[df['gender']!='XNA']
```

```
def purposes(reason):
    if 'образов' in reason:
        return 'образование'
    if 'свадь' in reason:
        return 'свадьба'
    if 'авто' in reason:
        return 'автомобиль'
    if 'жиль' in reason:
        return 'недвижимость'
    if 'недвиж' in reason:
        return 'недвижимость'
```

```
purposes('покупка жилья')
```

```
'недвижимость'
```

```
df['purposes_category'] = df['purpose'].apply(purposes)
```

```
df.head(10)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education	education_id	family_status	family_status_i
0	1	8437.673028	42	высшее	0	женат / замужем	0
1	1	4024.803754	36	среднее	1	женат / замужем	0
2	0	5623.422610	33	среднее	1	женат / замужем	0
3	3	4124.747207	32	среднее	1	женат / замужем	0
4	0	1630.019381	53	среднее	1	гражданский брак	1

	children	days_employed	dob_years	education	education_id	family_status	family_status_i
5	0	926.185831	27	высшее	0	гражданский брак	1
6	0	2879.202052	43	высшее	0	женат / замужем	0
7	0	152.779569	50	среднее	1	женат / замужем	0
8	2	6929.865299	35	высшее	0	гражданский брак	1
9	0	2188.756445	41	среднее	1	женат / замужем	0

```
# Проверка: Уникальных значений:

for i in ['education', 'family_status', 'gender', 'income_type', 'purposes_category']:
    print(df[i].unique())
```

```
['высшее' 'среднее' 'неоконченное высшее' 'начальное' 'ученая степень']
['женат / замужем' 'гражданский брак' 'вдовец / вдова' 'в разводе'
 'не женат / не замужем']
['F' 'M']
['сотрудник' 'пенсионер' 'компаньон' 'госслужащий' 'безработный'
 'предприниматель' 'студент' 'в декрете']
['недвижимость' 'автомобиль' 'образование' 'свадьба']
```

```
df.isna().sum()
```

```
children            0
days_employed       0
dob_years           0
education           0
education_id        0
family_status       0
family_status_id    0
gender              0
income_type         0
debt                0
total_income        0
purpose             0
purposes_category   0
dtype: int64
```

```
# Поиск и замена значений в заданном диапазоне

df['days_employed'] = df['days_employed'].fillna(df['days_employed'].median())
```

```
df
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
```

```
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education	education_id	family_status	family_sta
0	1	8437.673028	42	высшее	0	женат / замужем	0
1	1	4024.803754	36	среднее	1	женат / замужем	0
2	0	5623.422610	33	среднее	1	женат / замужем	0
3	3	4124.747207	32	среднее	1	женат / замужем	0
4	0	1630.019381	53	среднее	1	гражданский брак	1
...	...	...	...	...	...	...	...
21390	1	4529.316663	43	среднее	1	гражданский брак	1
21391	0	1630.019381	67	среднее	1	женат / замужем	0
21392	1	2113.346888	38	среднее	1	гражданский брак	1
21393	3	3112.481705	38	среднее	1	женат / замужем	0
21394	2	1984.507589	40	среднее	1	женат / замужем	0

21394 rows × 14 columns

```
# Цикл
# Поиск и замена пустых значений в заданном диапазоне

for i in df['income_type'].unique():
    df.loc[(df['income_type'] == i) & (df['total_income'].isna()), 'total_income'] = df.loc[(df['income_typ
```

```
df.duplicated().sum() # Проверка на наличие дубликатов
```

```
np.int64(0)
```

```
df = df.drop_duplicates() # Удаление дубликатов, если они есть
```

```
df.head() # Обнаружен дубликат столбца "purposes_caterogy"
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}
```

```
.dataframe thead th {
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education	education_id	family_status	family_status_i
0	1	8437.673028	42	высшее	0	женат / замужем	0
1	1	4024.803754	36	среднее	1	женат / замужем	0
2	0	5623.422610	33	среднее	1	женат / замужем	0
3	3	4124.747207	32	среднее	1	женат / замужем	0
4	0	1630.019381	53	среднее	1	гражданский брак	1

```
# df = df.drop(columns='purposes_caterogy')
# print(df)
```

```
df.head(1) # Check
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	children	days_employed	dob_years	education	education_id	family_status	family_status_i
0	1	8437.673028	42	высшее	0	женат / замужем	0

Ответы на вопросы

4A. ✅ Средняя зарплата по типу занятости

```
avg_income_by_type = df.groupby('income_type')['total_income'].mean()
print('Средняя зарплата по типу занятости: ', avg_income_by_type)
```

```
Средняя зарплата по типу занятости:  income_type
безработный        131339.751676
в декрете           53829.130729
госслужащий        168811.737769
компаньон          199558.373308
пенсионер          135266.515670
предприниматель    499163.144947
сотрудник          159533.471972
```

```
студент             98201.625314
Name: total_income, dtype: float64
```

4Б. ✅ Процент заемщиков с высшим образованием

```
percent = df.groupby('education')['education'].count()/len(df)*100
percent
```

```
education
высшее                  24.493615
начальное                1.319175
неоконченное высшее      3.466342
среднее                 70.692801
ученая степень           0.028068
Name: education, dtype: float64
```

```
print('Процент заемщиков с высшим образованием:', df[df['education'] == 'высшее']['education'].count()/df.s
```

```
Процент заемщиков с высшим образованием: 24.493614632549
```

4В. ✅ Частота кредитов на покупку жилья

```
percent_category = df[df['purposes_category'] == 'недвижимость']['purposes_category'].count()/len(df)*100
print('Частота кредитов на покупку жилья', percent_category)
```

```
Частота кредитов на покупку жилья 50.39996257660102
```

4Г. ✅ Зависимость между семейным положением и задолженностью

```
family_debt = df.groupby('family_status')['debt'].mean().sort_values(ascending=False)
print("Доля должников по семейному положению:")
print((family_debt * 100).round(1))
```

```
Доля должников по семейному положению:
family_status
не женат / не замужем    9.7
гражданский брак         9.3
женат / замужем          7.6
в разводе                7.0
вдовец / вдова           6.6
Name: debt, dtype: float64
```

4Д. ✅ Есть ли зависимость между количеством детей и возвратом кредита в срок?

```
children_debt = df.groupby('children')['debt'].mean().sort_values(ascending=False)
print("\nДоля должников по количеству детей:")
print((children_debt * 100).round(1))
```

```
Доля должников по количеству детей:
children
4    9.8
2    9.5
```

```
1    9.2
3    8.2
0    7.5
5    0.0
Name: debt, dtype: float64
```

4Д. Выводы

1. 0 детей: 7.5% должников
2. 1 ребенок: 9.2% должников
3. 2 детей: 9.5% должников
4. 3 детей: 8.2% должников 🔴 Неожиданное снижение
5. 4 детей: 9.8% должников 🔴 МАКСИМАЛЬНЫЙ РИСК
6. 5 детей: 0.0% должников 🔴 МИНИМАЛЬНЫЙ РИСК (аномалия)

⚠️ Зависимость существует, но она сложная и требует дополнительного исследования аномалий, особенно для 3 и 5 детей.

4Е. ✅ Есть ли зависимость между уровнем дохода и возвратом кредита в срок?

```
df['income_quartile'] = pd.qcut(df['total_income'], 4, labels=['Q1', 'Q2', 'Q3', 'Q4'])
income_debt = df.groupby('income_quartile')['debt'].mean()
print("\nДоля должников по квартилям дохода:")
print((income_debt * 100).round(1))
```

```
Доля должников по квартилям дохода:
income_quartile
Q1    8.0
Q2    8.8
Q3    8.5
Q4    7.1
Name: debt, dtype: float64
```

```
C:\Users\Станислав\AppData\Local\Temp\ipykernel_30576\1315253979.py:2: FutureWarning: The default of observ
  income_debt = df.groupby('income_quartile')['debt'].mean()
```

4Е. Выводы

Зависимость есть