

Описание проекта: оценка рисков невозврата кредита

В вашем распоряжении данные клиентов банка «Скрудж», которые планируют взять кредит. Вам необходимо выяснить, что из имеющихся данных влияет на своевременное погашение кредита и каким именно образом.

Исследование поможет в создании модели кредитного скоринга — системы для оценки способности потенциального заёмщика погасить свой кредит.

По каждому клиенту есть информация о его семейном положении, образовании, доходе и другие данные. Вам предстоит подготовить данные к анализу и исследовать их, в процессе отвечая на вопросы.

Описание данных

- `children` — количество детей в семье;
- `days_employed` — сколько дней работает клиент;
- `dob_years` — возраст клиента;
- `education` — уровень образования клиента;
- `education_id` — идентификатор образования клиента;
- `family_status` — семейное положение клиента;
- `family_status_id` — идентификатор семейного положения клиента;
- `gender` — пол клиента;
- `income_type` — тип дохода клиента;
- `debt` — был ли у клиента когда-либо просрочен платёж по кредиту;
- `total_income` — ежемесячный доход;
- `purpose` — причина оформления кредита.

Цель работы

- В рамках проекта необходимо проанализировать данные клиенток банка Скрудж для дальнейшего определения характеристик заёмщика, влияющих на своевременное погашение кредита. Это поможет в дальнейшем создать основу для модели кредитного скоринга — системы для оценки способности потенциального заёмщика погасить свой кредит.

План работы

- [Первичный анализ данных](#)
- [Предобработка](#)
- [Выбросы и аномальные значения](#)
 - загрузка и изучение структуры данных
 - проверка и устранение пропущенных значений
 - устранение некорректных значений
 - устранение неявных дубликатов
- [Создание дополнительных признаков](#)
 - разделите клиентов по уровню дохода
 - разделение по возрастным группам
 - разделение по количеству детей
- [Исследование влияющих факторов](#)
 - Уровень дохода
 - Анализ влияния дохода на своевременное погашение кредита
 - Образование
 - Исследование связи между уровнем образования и вероятностью задолженности
 - Возраст
 - Анализ возрастных категорий и их связи с задолженностью по кредитам
 - Количество детей
 - влияние количества детей на риск задолженности
- [Анализ данных](#)
 - построение графиков и сводных таблиц для наглядной визуализации сравнения должников и не должников
- [Проверка исследовательских гипотез](#)
 - У клиентов с детьми более высокий уровень финансовой ответственности и, следовательно, более низкий риск просрочек по кредиту.
 - Одиноким мужчинам с низким доходом чаще оказываются должниками, чем семейные мужчины со средним доходом.
- [Выводы](#)
 - описание полученных результатов и итоговые выводы проведённого исследования
- [Проведение дополнительного исследования](#)

Датасет содержит данные, которые несут в себе информацию о клиентах банка Скрудж:

- **демография**
 - возраст, семейное положение, количество детей

- **финансовые показатели**
 - уровень дохода, тип занятости
- **данные об образовании**
- **цель кредита**
- ____информацию о своевременности возврата кредита____

Шаг 1. Первичный анализ данных

1. Загрузите датасет.
2. Сделайте копию датасета.
3. Изучите типы данных и определите, соответствуют ли они содержимому.
4. Напишите вывод.

```
In [2]: # Импортируйте библиотеки для работы
# с таблицами и графиками
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: # Загрузите датасет в переменную df_raw или другую по вашему выбору.
# Он доступен по адресу <https://code.s3.yandex.net/datasets/credit_scoring_eng.csv>
# а при работе на платформе по адресу /datasets/credit_scoring_eng.csv
df_raw=pd.read_csv('https://code.s3.yandex.net/datasets/credit_scoring_eng.csv')
```

```
In [4]: # При помощи метода .copy() скопируйте датасет
# для работы с ним в переменную df или другую
df=df_raw.copy()
```

```
In [5]: # Изучите общую информацию о датасете
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children               21525 non-null  int64
1   days_employed          19351 non-null  float64
2   dob_years              21525 non-null  int64
3   education              21525 non-null  object
4   education_id           21525 non-null  int64
5   family_status          21525 non-null  object
6   family_status_id       21525 non-null  int64
7   gender                 21525 non-null  object
8   income_type            21525 non-null  object
9   debt                   21525 non-null  int64
10  total_income           19351 non-null  float64
11  purpose                21525 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

```
In [6]: df.head()
```

Out[6]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose
0	1	-8437.673028	42	bachelor's degree	0	married	0	F	employee	0	40620.102	purchase of the house
1	1	-4024.803754	36	secondary education	1	married	0	F	employee	0	17932.802	car purchase
2	0	-5623.422610	33	Secondary Education	1	married	0	M	employee	0	23341.752	purchase of the house
3	3	-4124.747207	32	secondary education	1	married	0	M	employee	0	42820.568	supplementary education
4	0	340266.072047	53	secondary education	1	civil partnership	1	F	retiree	0	25378.572	to have a wedding

1. Таблица представлена 12 колонками и 21525 строками

2. Данные в таблице представлены 3 типами: float64, int64 и object:

- int64 для столбцов *children*, *dob_years*, *education_id*, *family_status_id*, *u debt*
 - это целочисленные данные, что логично соответствует как описанию данных, так и простмору первых 5 строк таблицы
- float64 для столбцов *days_employed* *u total_income*

- это числовые данные с плавающей точкой, что логично соответствует описанию данных, так и простмору первых 5 строк таблицы
- object для столбцов *education, family_status, gender, income_type, u purpose*
 - это текстовые данные , что логично соответствует описанию данных, так и простмору первых 5 строк таблицы

3. Также стоит обратить внимани на колонки **days_employed** и **total_income**.

- В обеих колонках количество строк (19351) не соответствует общему количеству строк таблицы(21525)
 - таким образом в этих столбцах могут быть пропущенные значения, например NAN

[* к содержанию](#)

Шаг 2. Выполнение предобработки данных

1. Найдите и изучите пропущенные значения в столбцах.
2. Устраните пропущенные значения: удалите или замените их.
3. Объясните выбранную стратегию обработки пропущенных значений.

```
In [7]: # найдем пропущенные значения в общем количестве
df.isnull().sum().sort_values(ascending=False)
```

```
Out[7]: days_employed      2174
total_income      2174
children           0
dob_years          0
education          0
education_id       0
family_status      0
family_status_id   0
gender             0
income_type        0
debt               0
purpose            0
dtype: int64
```

```
In [9]: # найдем пропущенные значения в процентном соотношении
df.isnull().mean().sort_values(ascending=False)
```

```
Out[9]: days_employed      0.100999
total_income      0.100999
children          0.000000
dob_years         0.000000
education         0.000000
education_id      0.000000
family_status     0.000000
family_status_id  0.000000
gender            0.000000
income_type       0.000000
debt              0.000000
purpose           0.000000
dtype: float64
```

- в колонках *days_employed* и *total_income* 2174 пропусков, что составляет 10 процентов от всех данных
- слишком большой процент для удаления строк из данных
- поэтому было решено заменить пропуски в зависимости от типа данных

сначада для обоих столбцов применим *describe()*

```
In [10]: # Применение метода describe() к двум столбцам: 'days_employed' и 'total_income'
df[['days_employed', 'total_income']].describe()
```

Out[10]:

	days_employed	total_income
count	19351.000000	19351.000000
mean	63046.497661	26787.568355
std	140827.311974	16475.450632
min	-18388.949901	3306.762000
25%	-2747.423625	16488.504500
50%	-1203.369529	23202.870000
75%	-291.095954	32549.611000
max	401755.400475	362496.645000

в колонке *days_employed* присутствуют отрицательные значения(это приславутые аномалии)

- пропущенные же знаенчия заменим 0
 - из идеи что клиенты могли быть безработными на момент заполнения данных

```
In [11]: # Замена пропущенных значений в days_employed на 0
df['days_employed'].fillna(0, inplace=True)
```

в колонке `total_income` обратим внимание на мин макс значения

- то есть присутствуют явные выбросы
 - таким образом выбираем для замены медианное значение, поскольку оно менее чувствительно к ним

```
In [12]: # Замена пропущенных значений в total_income на медианное значение
df['total_income'].fillna(df['total_income'].median(), inplace=True)
```

еще раз проверим данные на пропуски

```
In [14]: # найдем пропущенные значения в общем количестве
df.isnull().sum().sort_values(ascending=False)
```

```
Out[14]: 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
dtype: int64
```

1. Изучите уникальные значения в столбцах с уровнем образования (`education`) и полом клиента (`gender`).
2. Устраните неявные дубликаты и некорректные значения.

```
In [17]: # Уникальные значения в столбце education
df['education'].unique()
```

```
Out[17]: array(["bachelor's degree", 'secondary education', 'Secondary Education',
                'SECONDARY EDUCATION', "BACHELOR'S DEGREE", 'some college',
                'primary education', "Bachelor's Degree", 'SOME COLLEGE',
                'Some College', 'PRIMARY EDUCATION', 'Primary Education',
                'Graduate Degree', 'GRADUATE DEGREE', 'graduate degree'],
              dtype=object)
```

```
In [19]: sorted(df['education'].unique())
```

```
Out[19]: ["BACHELOR'S DEGREE",
          "Bachelor's Degree",
          'GRADUATE DEGREE',
          'Graduate Degree',
          'PRIMARY EDUCATION',
          'Primary Education',
          'SECONDARY EDUCATION',
          'SOME COLLEGE',
          'Secondary Education',
          'Some College',
          "bachelor's degree",
          'graduate degree',
          'primary education',
          'secondary education',
          'some college']
```

- при анализе уникальных значений в колонке `education` видны повторяющиеся текстовые значения по смысловому содержанию, но в разных стилях написания
 - например 'Secondary Education' и 'SECONDARY EDUCATION'
 - нужно привести все в нижний регистр

```
In [20]: # Уникальные значения в столбце education
df['gender'].unique()
```

```
Out[20]: array(['F', 'M', 'XNA'], dtype=object)
```

- при анализе уникальных значений в колонке `gender` видны XNA
 - заменим на 'Unknown'

проведем замену и перепроверим уникальные значения в столбцах `education` и `gender`

```
In [21]: # Приведение всех значений в столбце education к нижнему регистру для устранения дубликатов
df['education'] = df['education'].str.lower()
```

```
In [22]: # Уникальные значения в столбце education
df['education'].unique()
```

```
Out[22]: array(["bachelor's degree", 'secondary education', 'some college',
                'primary education', 'graduate degree'], dtype=object)
```

```
In [23]: sorted(df['education'].unique())

Out[23]: ["bachelor's degree",
          'graduate degree',
          'primary education',
          'secondary education',
          'some college']

In [117... # Устранение некорректных значений в столбце gender
df['gender'] = df['gender'].replace({'XNA': 'Unknown'})

In [118... # Уникальные значения в столбце education
df['gender'].unique()

Out[118... array(['F', 'M', 'Unknown'], dtype=object)
```

проверем информацию по исходной таблице и таблице на данном этапе анализа

```
In [119... df_raw.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children               21525 non-null  int64
1   days_employed          19351 non-null  float64
2   dob_years              21525 non-null  int64
3   education              21525 non-null  object
4   education_id           21525 non-null  int64
5   family_status          21525 non-null  object
6   family_status_id       21525 non-null  int64
7   gender                 21525 non-null  object
8   income_type            21525 non-null  object
9   debt                   21525 non-null  int64
10  total_income           19351 non-null  float64
11  purpose                21525 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB

In [120... df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children               21525 non-null  int64
1   days_employed          21525 non-null  float64
2   dob_years              21525 non-null  int64
3   education              21525 non-null  object
4   education_id           21525 non-null  int64
5   family_status          21525 non-null  object
6   family_status_id       21525 non-null  int64
7   gender                 21525 non-null  object
8   income_type            21525 non-null  object
9   debt                   21525 non-null  int64
10  total_income           21525 non-null  float64
11  purpose                21525 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

1. Проверьте наличие дубликатов. Изучите дублированные данные, если они есть, и примите решение — удалять их или оставить.

```
In [121... # Проверка наличия дубликатов
duplicates = df.duplicated()
print(f'Количество дубликатов: {duplicates.sum()}')

Количество дубликатов: 71



- в данных присусует 71 дубликат
- выведем пример дубликатов



In [122... # Изучение дублированных данных
duplicate_rows = df[df.duplicated()]
duplicate_rows.head()
```

Out[122...

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose	
	2849	0	0.0	41	secondary education	1	married	0	F	employee	0	23202.87	purchase of the house for my family
	3290	0	0.0	58	secondary education	1	civil partnership	1	F	retiree	0	23202.87	to have a wedding
	4182	1	0.0	34	bachelor's degree	0	civil partnership	1	F	employee	0	23202.87	wedding ceremony
	4851	0	0.0	60	secondary education	1	civil partnership	1	F	retiree	0	23202.87	wedding ceremony
	5557	0	0.0	58	secondary education	1	civil partnership	1	F	retiree	0	23202.87	to have a wedding

- на первый взгляд сложно оценить какие именно строки дублируются
- для лучшей визуализации отсортируем дублированные строки

In [123...

```
# Найти и отсортировать дублированные строки
duplicate_rows = df[df.duplicated(keep=False)]
sorted_duplicates = duplicate_rows.sort_values(by=list(df.columns))

sorted_duplicates.head(6)
```

Out[123...

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose	
	15892	0	0.0	23	secondary education	1	unmarried	4	F	employee	0	23202.87	second hand car purchase
	19321	0	0.0	23	secondary education	1	unmarried	4	F	employee	0	23202.87	second hand car purchase
	3452	0	0.0	29	bachelor's degree	0	married	0	M	employee	0	23202.87	buying residential real estate
	18328	0	0.0	29	bachelor's degree	0	married	0	M	employee	0	23202.87	buying residential real estate
	4216	0	0.0	30	secondary education	1	married	0	M	employee	0	23202.87	building residential real estate
	6312	0	0.0	30	secondary education	1	married	0	M	employee	0	23202.87	building residential real estate

- удалим дубликаты

In [124...

```
# Удаление дубликатов
df = df.drop_duplicates()
print(f'Количество дубликатов после удаления: {df.duplicated().sum()}')
```

Количество дубликатов после удаления: 0

проверем информацию по исходной таблице и таблице на данном этапе анализа

In [125...

```
df_raw.shape
```

Out[125... (21525, 12)

In [126...

```
df.shape
```

Out[126... (21454, 12)

In [127...

```
#строк стало на 71 меньше
print(f'Количество строк стало меньше на : {round((1-df.shape[0]/df_raw.shape[0])*100,2)} процента')
```

Количество строк стало меньше на : 0.33 процента

[* к содержанию](#)

Шаг 3. Выбросы и аномальные значения

Изучите столбцы `total_income` , `dob_age` , `chidlren` на наличие выбросов и аномальных значений, в том числе при помощи графиков. Если выбросы или аномалии будут обнаружены — обоснованно примите решение об их судьбе. Используйте подзаголовки третьего уровня (`###` Подзаголовок), чтобы создать структуру действий в рамках этого шага.

Проанализмруем столбцы `total_income`, `dob_years` и `children` методом `describe()`

```
In [128... display(df[['total_income', 'dob_years', 'children']].describe())
```

	total_income	dob_years	children
count	21454.000000	21454.000000	21454.000000
mean	26436.183035	43.271231	0.539946
std	15683.361605	12.570822	1.383444
min	3306.762000	0.000000	-1.000000
25%	17219.817250	33.000000	0.000000
50%	23202.870000	42.000000	0.000000
75%	31330.237250	53.000000	1.000000
max	362496.645000	75.000000	20.000000

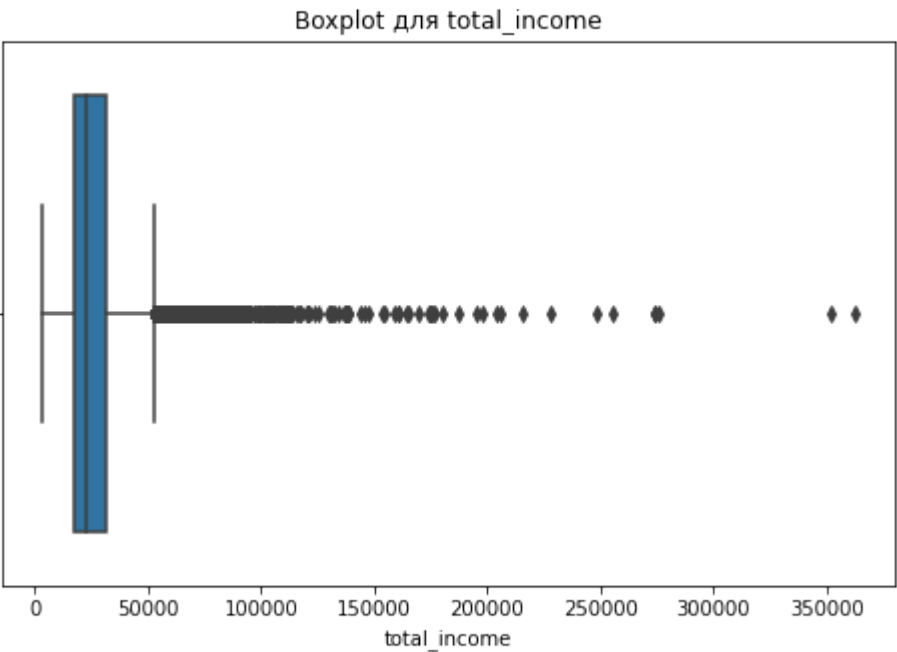
- **total_income:**
 - минимальное значение 3306.76 и максимальное значение 362496.65(значительно превышает 75-й процентиль (31330.24)) указывают на наличие выбросов, соответственно можно заменить экстремальные значения на медиану или удалить
 - среднее значени выше медианного
- **dob_years:**
 - максимум 75 лет в пределах разумного, но возраст 0 явно аномалия.
 - среднее значени близко к медианнму, что о говорит о возможной симметричности в распределении
- **children:**
 - присутвующя явные аномальные значения -1 и 20, соответственно можно заменить их на медиану или удалить.

Построим граифки для столбцов `total_income`, `dob_years` и `children`

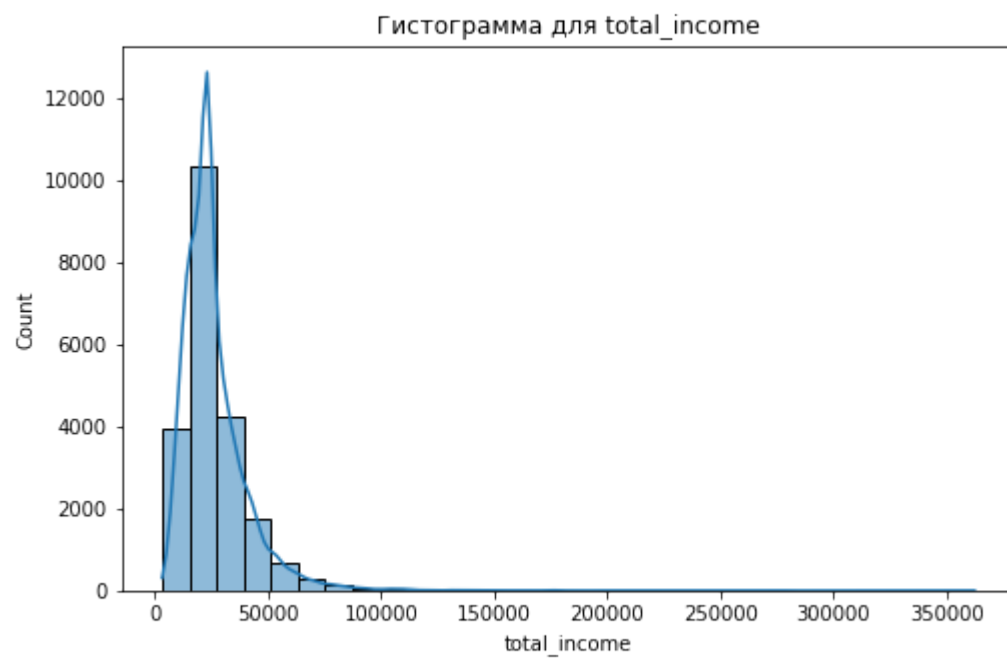
- используем для анализа аномалий и выборосов график `boxplot` и распределение на основе гистограммы

```
In [129... import matplotlib.pyplot as plt
import seaborn as sns

# Построение графика для `total_income`
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['total_income'])
plt.title('Boxplot для total_income')
plt.show()
```



```
In [130... plt.figure(figsize=(8,5))
sns.histplot(df['total_income'], bins=30, kde=True)
plt.title('Гистограмма для total_income')
plt.show()
```

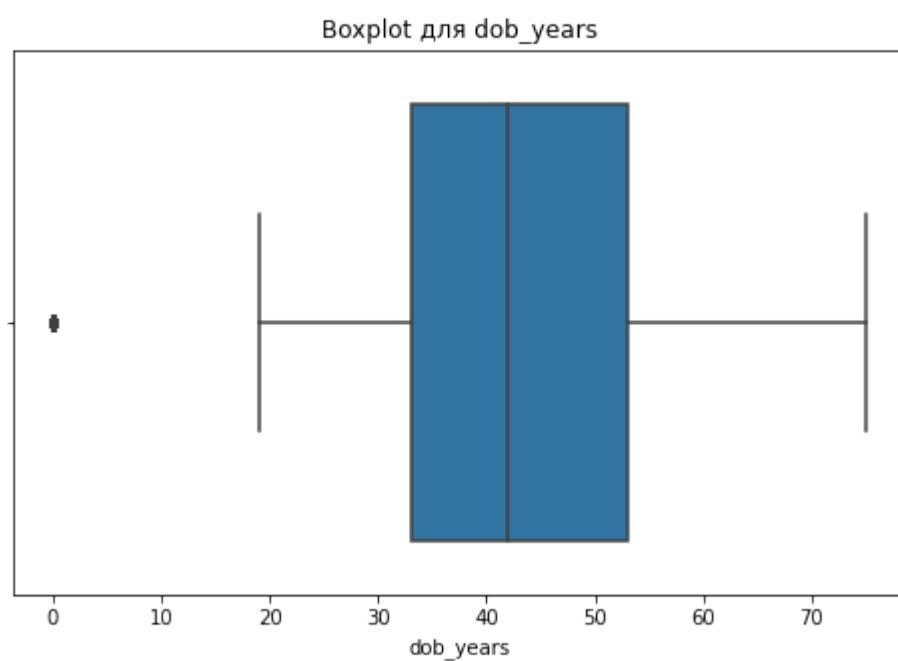



```
In [131... Q1 = df['total_income'].quantile(0.25)
Q3 = df['total_income'].quantile(0.75)
IQR=Q3-Q1
print(f'Нижняя граница : {Q1 - 1.5*IQR }')
print(f'Верхняя граница : {Q3 + 1.5*IQR }')
```

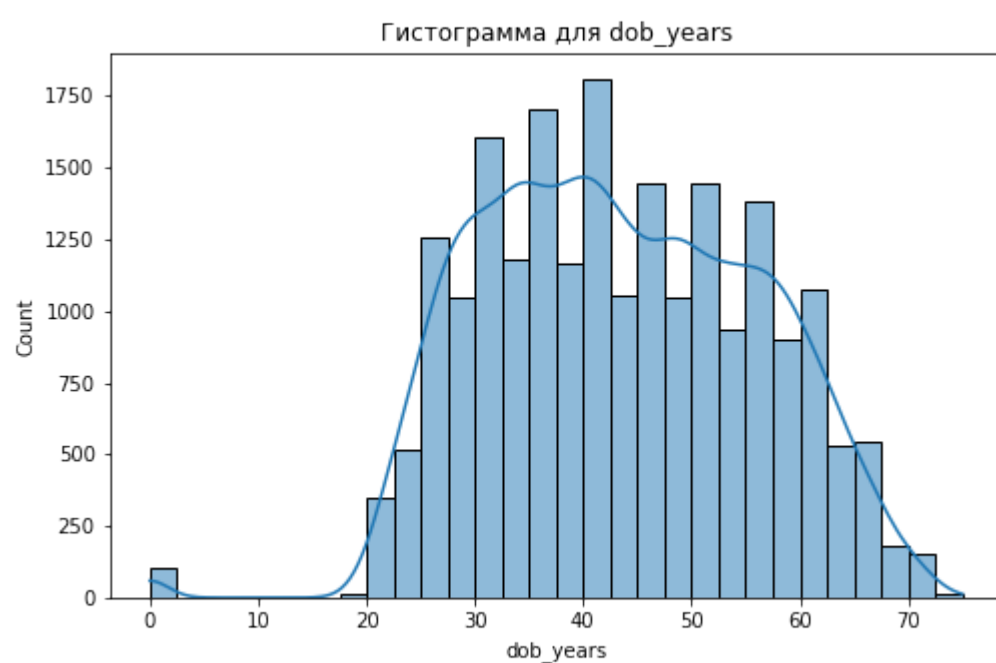
Нижняя граница : -3945.8127499999973
Верхняя граница : 52495.867249999996

- на графике видно, что есть значительное количество выбросов справа по данным в столбце total_income.
- есть значения, выходящие за верхний предел, что может указывать на аномалии.

```
In [132... # Построение графика для `dob_years`
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['dob_years'])
plt.title('Boxplot для dob_years')
plt.show()
```



```
In [133... plt.figure(figsize=(8, 5))
sns.histplot(df['dob_years'], bins=30, kde=True)
plt.title('Гистограмма для dob_years')
plt.show()
```



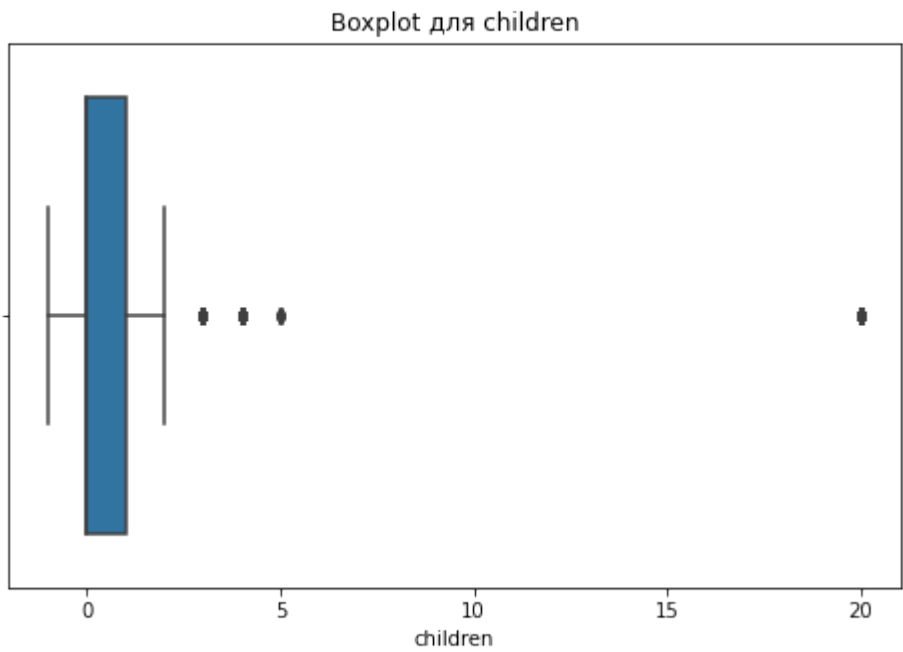
```
In [134... Q1 = df['dob_years'].quantile(0.25)
Q3 = df['dob_years'].quantile(0.75)
IQR=Q3-Q1
print(f'Нижняя граница : {Q1 - 1.5*IQR }')
print(f'Верхняя граница : {Q3 + 1.5*IQR }')
```


Нижняя граница : 3.0
Верхняя граница : 83.0

- на графике видны выбросы слева
- возраст 0 явно является явной аномалией, так как клиент не может быть младше 18 лет

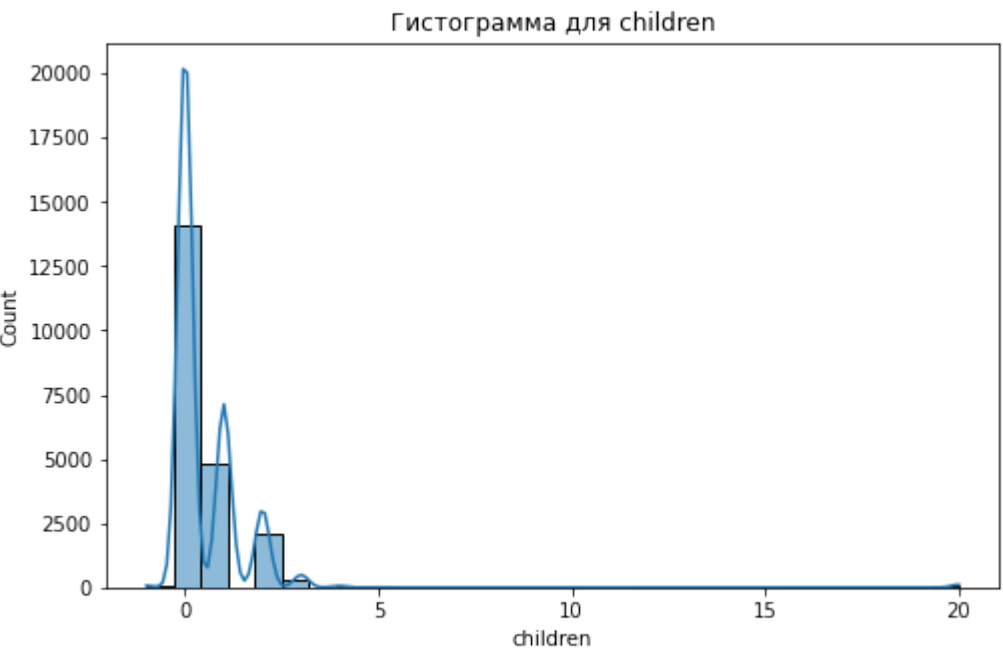
In [135...

```
# Построение графика для `children`  
plt.figure(figsize=(8, 5))  
sns.boxplot(x=df['children'])  
plt.title('Boxplot для children')  
plt.show()
```



In [136...

```
plt.figure(figsize=(8, 5))  
sns.histplot(df['children'], bins=30, kde=True)  
plt.title('Гистограмма для children')  
plt.show()
```



- видно несколько выбросов справа (большое количество детей 20)
- на графике видно, что есть аномальные значения -1

Решение по выбросам

- для колонки total_income(ежемесячный доход)
 - лучше удалить выбросы, чтоб они не искажали данные.

In [137...

```
upper_limit = df['total_income'].quantile(0.95) # определение верхнего предела выбросов  
df = df[df['total_income'] <= upper_limit]
```

In [138...

```
df['total_income'].describe()
```

Out[138...

```
count    20381.000000  
mean      24007.680143  
std       9821.324564  
min        3306.762000  
25%      16823.508000  
50%      23202.870000  
75%      29411.892000  
max       53039.263000  
Name: total_income, dtype: float64
```

- для колонки dob_years(возраст клиента)
 - посмотрим количество 0

In [139...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20381 entries, 0 to 21524
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   children              20381 non-null  int64
1   days_employed         20381 non-null  float64
2   dob_years             20381 non-null  int64
3   education             20381 non-null  object
4   education_id          20381 non-null  int64
5   family_status         20381 non-null  object
6   family_status_id      20381 non-null  int64
7   gender                20381 non-null  object
8   income_type           20381 non-null  object
9   debt                  20381 non-null  int64
10  total_income          20381 non-null  float64
11  purpose               20381 non-null  object
dtypes: float64(2), int64(5), object(5)
memory usage: 2.0+ MB
```

```
In [140... df[df['dob_years'] == 0].shape[0]
```

```
Out[140... 100
```

- удалим 0

```
In [141... # Удалить строки, где dob_years равен 0
df = df[df['dob_years'] != 0]
```

```
In [142... df['dob_years'].describe()
```

```
Out[142... count    20281.000000
mean      43.514274
std       12.323288
min       19.000000
25%      33.000000
50%      43.000000
75%      53.000000
max       75.000000
Name: dob_years, dtype: float64
```

- для колонки children(количество детей в семье)
 - посмтотрим количсетво -1 и 20

```
In [143... df[df['children'] == -1].shape[0]
```

```
Out[143... 47
```

```
In [144... df[df['children'] == 20].shape[0]
```

```
Out[144... 73
```

- удалим -1 и 20

```
In [145... # Удалить строки, где children равен -1
df = df[df['children'] != -1]
```

```
In [146... # Удалить строки, где children равен 20
df = df[df['children'] != 20]
```

```
In [147... df['children'].describe()
```

```
Out[147... count    20161.000000
mean      0.470612
std       0.749234
min       0.000000
25%      0.000000
50%      0.000000
75%      1.000000
max       5.000000
Name: children, dtype: float64
```

восопльзуемся еще раз describe для демонстрации статистики по столбцам и отсутствию выбросов

```
In [148... display(df[['total_income', 'dob_years', 'children']].describe())
```

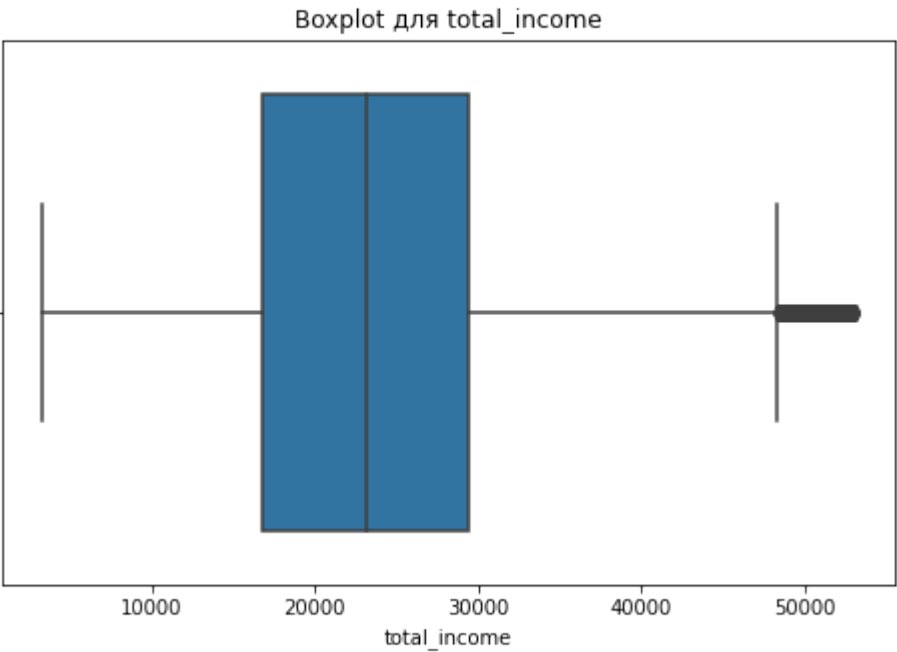
	total_income	dob_years	children
count	20161.000000	20161.000000	20161.000000
mean	23997.798351	43.519865	0.470612
std	9812.385382	12.329629	0.749234
min	3306.762000	19.000000	0.000000
25%	16817.222000	33.000000	0.000000
50%	23202.870000	43.000000	0.000000
75%	29410.209000	54.000000	1.000000
max	53039.263000	75.000000	5.000000

также продублируем построение графиков для визуализации данных

In [149...

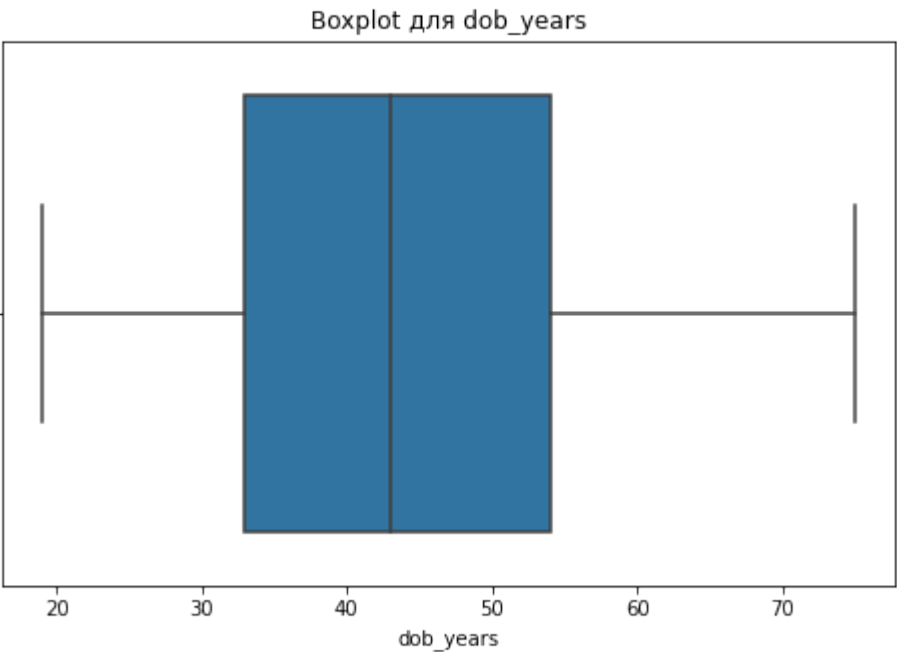
```
import matplotlib.pyplot as plt
import seaborn as sns

# Построение графика для `total_income`
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['total_income'])
plt.title('Boxplot для total_income')
plt.show()
```



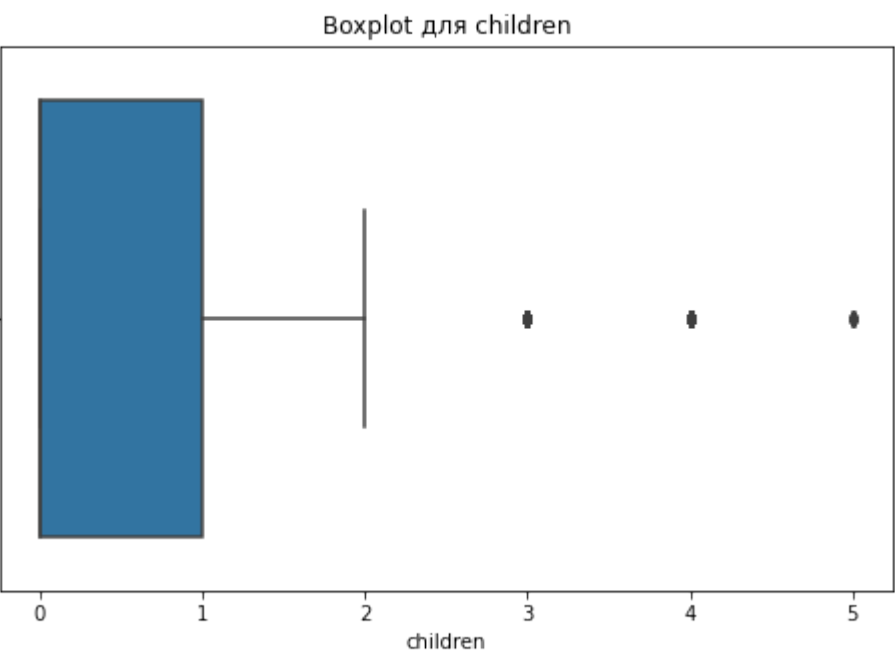
In [150...

```
# Построение графика для `dob_years`
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['dob_years'])
plt.title('Boxplot для dob_years')
plt.show()
```



In [151...

```
# Построение графика для `children`
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['children'])
plt.title('Boxplot для children')
plt.show()
```



проверем информацию по исходной таблице и таблице на данном этапе анализа

```
In [152... df_raw.shape

Out[152... (21525, 12)

In [153... df.shape

Out[153... (20161, 12)

In [154... #строк стало на 71 меньше
print(f'Количество строк стало меньше на : {round((1-df.shape[0]/df_raw.shape[0])*100,2)} процента')
```

Количество строк стало меньше на : 6.34 процента

[* к содержанию](#)

Шаг 4. Добавление в таблицу новых столбцов

1. Разделите клиентов на 5 категорий по уровню дохода:

- Без дохода — люди без работы и с нулевым доходом.
- Очень низкий доход — люди, получающие ниже 14 перцентиля от общего распределения дохода.
- Низкий доход — люди, получающие между 14 и 34 перцентилями от общего распределения дохода.
- Средний доход — люди, получающие между 34 и 78 перцентилями от общего распределения дохода.
- Высокий доход — люди, получающие больше 78 перцентиля от общего распределения дохода.

```
In [155... # Процентильные границы
percentile_14 = df['total_income'].quantile(0.14)
percentile_34 = df['total_income'].quantile(0.34)
percentile_78 = df['total_income'].quantile(0.78)

# Функция для присвоения категорий
def income_category(income):
    if income == 0:
        return 'Без дохода'
    elif income < percentile_14:
        return 'Очень низкий доход'
    elif income < percentile_34:
        return 'Низкий доход'
    elif income < percentile_78:
        return 'Средний доход'
    else:
        return 'Высокий доход'

# Создание новой колонки согласно функции
df['income_category'] = df['total_income'].apply(income_category)

display(df[['total_income', 'income_category']].head())
print(df['income_category'].value_counts())
```

	total_income	income_category
0	40620.102	Высокий доход
1	17932.802	Низкий доход
2	23341.752	Средний доход
3	42820.568	Высокий доход
4	25378.572	Средний доход

Средний доход 8870
Высокий доход 4436
Низкий доход 4032
Очень низкий доход 2823
Name: income_category, dtype: int64

1. Разделите клиентов на две категории по возрасту: до 40 лет и после. Сохраните результат в колонке `age_category` .

In [156...

```
# Место для вашего кода
# Функция для присвоения возрастных категорий
def age_category(age):
    if age < 40:
        return 'до 40 лет'
    else:
        return 'после 40 лет'
df['age_category'] = df['dob_years'].apply(age_category)
display(df[['dob_years', 'age_category']].head())
```

	dob_years	age_category
0	42	после 40 лет
1	36	до 40 лет
2	33	до 40 лет
3	32	до 40 лет
4	53	после 40 лет

1. Разделите клиентов на несколько категорий по количеству детей: без детей, от одного до двух, от трёх и больше. Сохраните результат в колонке `childrens_category` .

In [157...

```
# Место для вашего кода
def childrens_category(children):
    if children == 0:
        return 'без детей'
    elif children <= 2:
        return 'от одного до двух'
    else:
        return 'от трёх и больше'
df['childrens_category'] = df['children'].apply(childrens_category)
display(df[['children', 'childrens_category']].head())
```

	children	childrens_category
0	1	от одного до двух
1	1	от одного до двух
2	0	без детей
3	3	от трёх и больше
4	0	без детей

как итог

In [158...

```
display(df[['children', 'childrens_category', 'dob_years', 'age_category', 'total_income', 'income_category']].head(10))
```

	children	childrens_category	dob_years	age_category	total_income	income_category
0	1	от одного до двух	42	после 40 лет	40620.102	Высокий доход
1	1	от одного до двух	36	до 40 лет	17932.802	Низкий доход
2	0	без детей	33	до 40 лет	23341.752	Средний доход
3	3	от трёх и больше	32	до 40 лет	42820.568	Высокий доход
4	0	без детей	53	после 40 лет	25378.572	Средний доход
5	0	без детей	27	до 40 лет	40922.170	Высокий доход
6	0	без детей	43	после 40 лет	38484.156	Высокий доход
7	0	без детей	50	после 40 лет	21731.829	Средний доход
8	2	от одного до двух	35	до 40 лет	15337.093	Низкий доход
9	0	без детей	41	после 40 лет	23108.150	Средний доход

Шаг 5. Проведение исследовательского анализа данных

Задайте структуру наиболее объёмной части исследования. Исследуйте факторы: `Уровень дохода` , `Образование` , `Возраст` , `Количество детей` .
Отличается ли распределение между должниками и нет? Исследуйте вопрос графически. Постройте сводную таблицу по каждому фактору и покажите, как часто встречаются должники в каждой группе клиентов. Выберите подходящую визуализацию и сравните 2 группы.

- визуализируем таблицу с колонками, данные из которых будем анализировать

In [159...

```
display(df[['income_category', 'education', 'dob_years', 'age_category', 'children', 'childrens_category', 'debt', 'total_income']].head(10))
```

	income_category	education	dob_years	age_category	children	childrens_category	debt	total_income
0	Высокий доход	bachelor's degree	42	после 40 лет	1	от одного до двух	0	40620.102
1	Низкий доход	secondary education	36	до 40 лет	1	от одного до двух	0	17932.802
2	Средний доход	secondary education	33	до 40 лет	0	без детей	0	23341.752
3	Высокий доход	secondary education	32	до 40 лет	3	от трёх и больше	0	42820.568
4	Средний доход	secondary education	53	после 40 лет	0	без детей	0	25378.572
5	Высокий доход	bachelor's degree	27	до 40 лет	0	без детей	0	40922.170
6	Высокий доход	bachelor's degree	43	после 40 лет	0	без детей	0	38484.156
7	Средний доход	secondary education	50	после 40 лет	0	без детей	0	21731.829
8	Низкий доход	bachelor's degree	35	до 40 лет	2	от одного до двух	0	15337.093
9	Средний доход	secondary education	41	после 40 лет	0	без детей	0	23108.150

- **Уровень дохода**

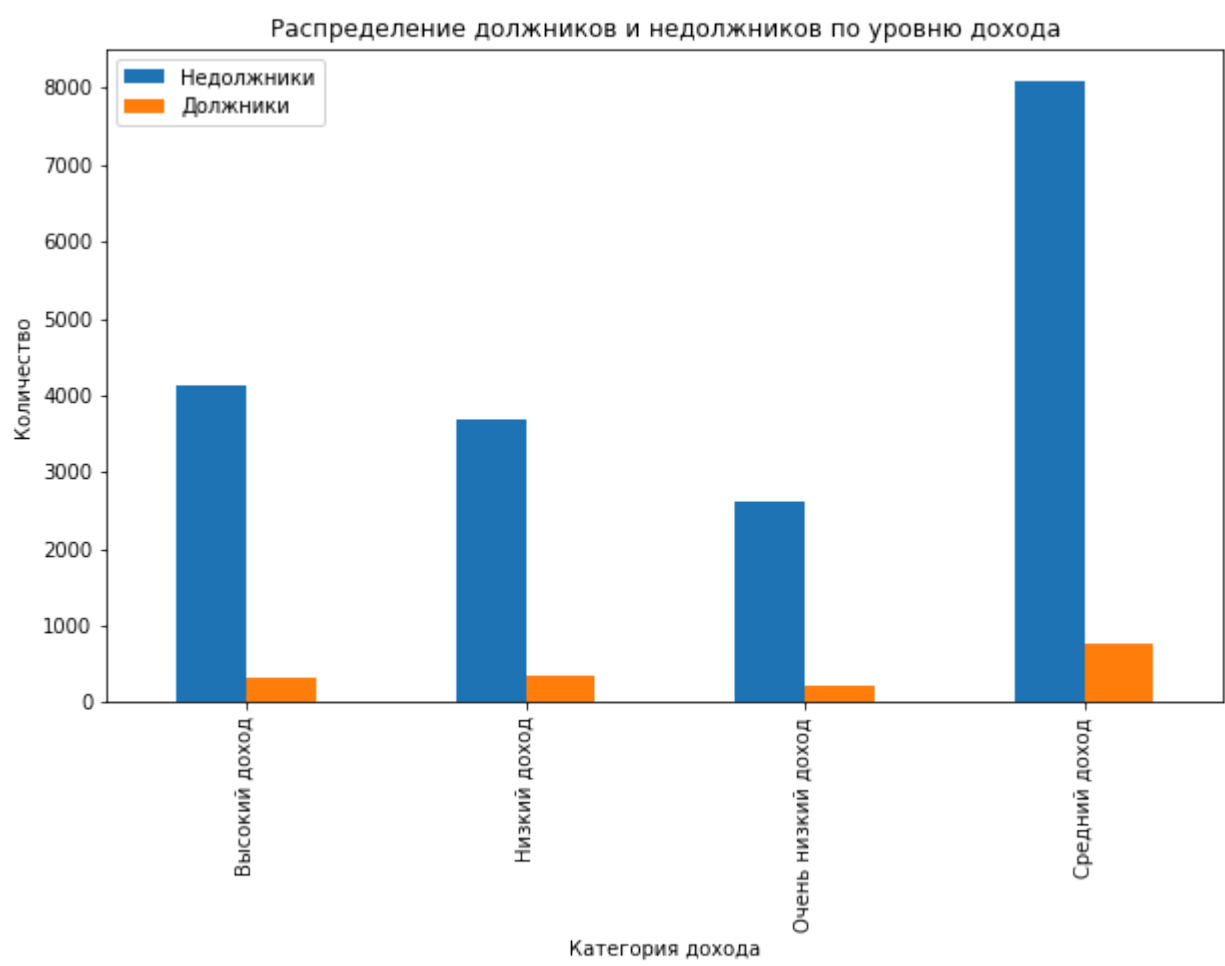
In [160...

```
# Место для вашего кода
income_pivot = df.pivot_table(index='income_category', columns='debt', values='total_income', aggfunc='count')
display(income_pivot)
income_pivot.plot(kind='bar', figsize=(10, 6))
plt.title('Распределение должников и недолжников по уровню дохода')
plt.xlabel('Категория дохода')
plt.ylabel('Количество')
plt.legend(['Недолжники', 'Должники'])
```

	debt	0	1
income_category			
Высокий доход		4115	321
Низкий доход		3691	341
Очень низкий доход		2608	215
Средний доход		8098	772

Out[160...

```
<matplotlib.legend.Legend at 0x7fbabec72d30>
```

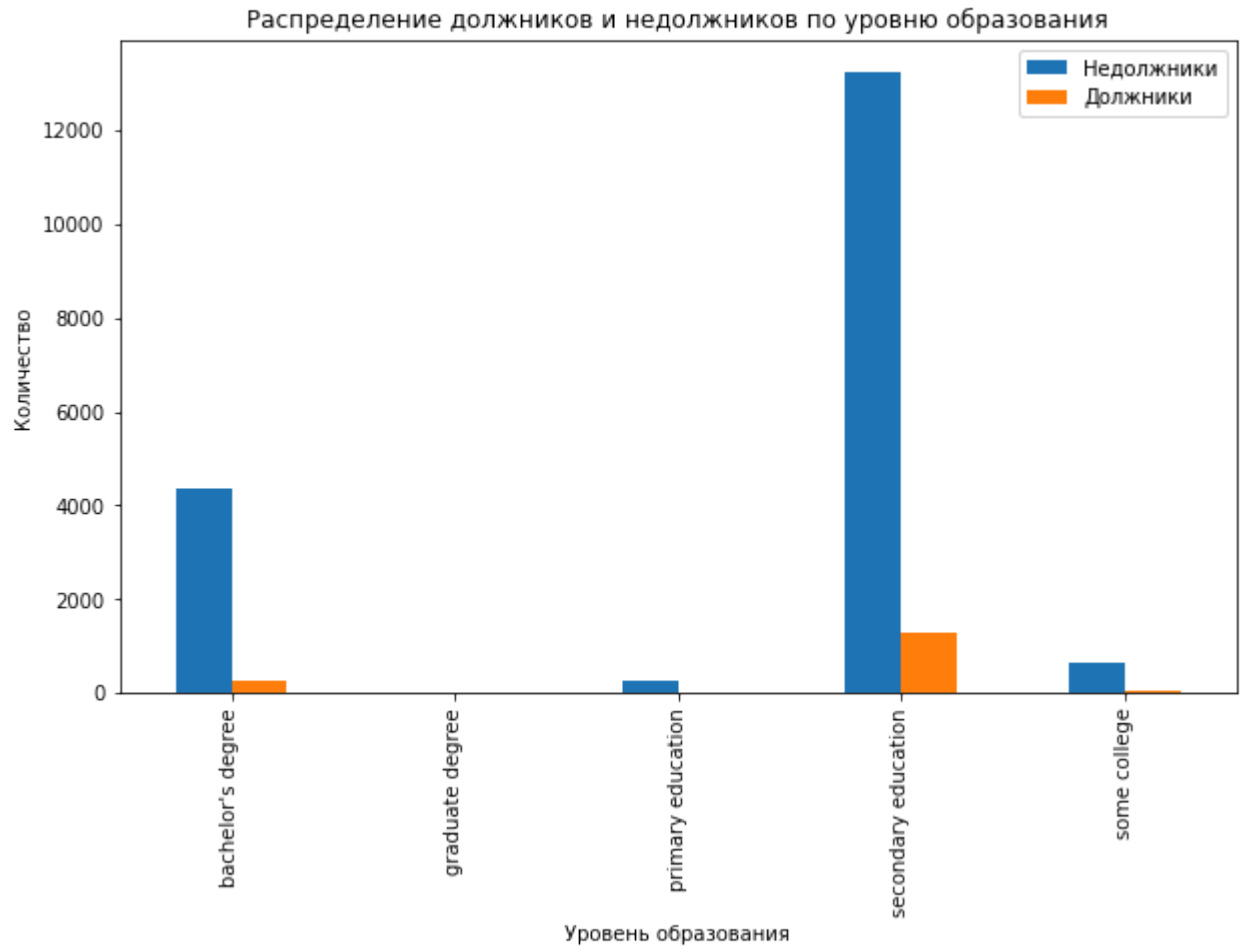


- **Образование**

```
In [161... education_pivot = df.pivot_table(index='education', columns='debt', values='total_income', aggfunc='count')
display(education_pivot)
education_pivot.plot(kind='bar', figsize=(10, 6))
plt.title('Распределение должников и недолжников по уровню образования')
plt.xlabel('Уровень образования')
plt.ylabel('Количество')
plt.legend(['Недолжники', 'Должники'])
```

	debt	
	0	1
education		
bachelor's degree	4377.0	254.0
graduate degree	6.0	NaN
primary education	248.0	31.0
secondary education	13240.0	1303.0
some college	641.0	61.0

Out[161... <matplotlib.legend.Legend at 0x7fbabe959100>

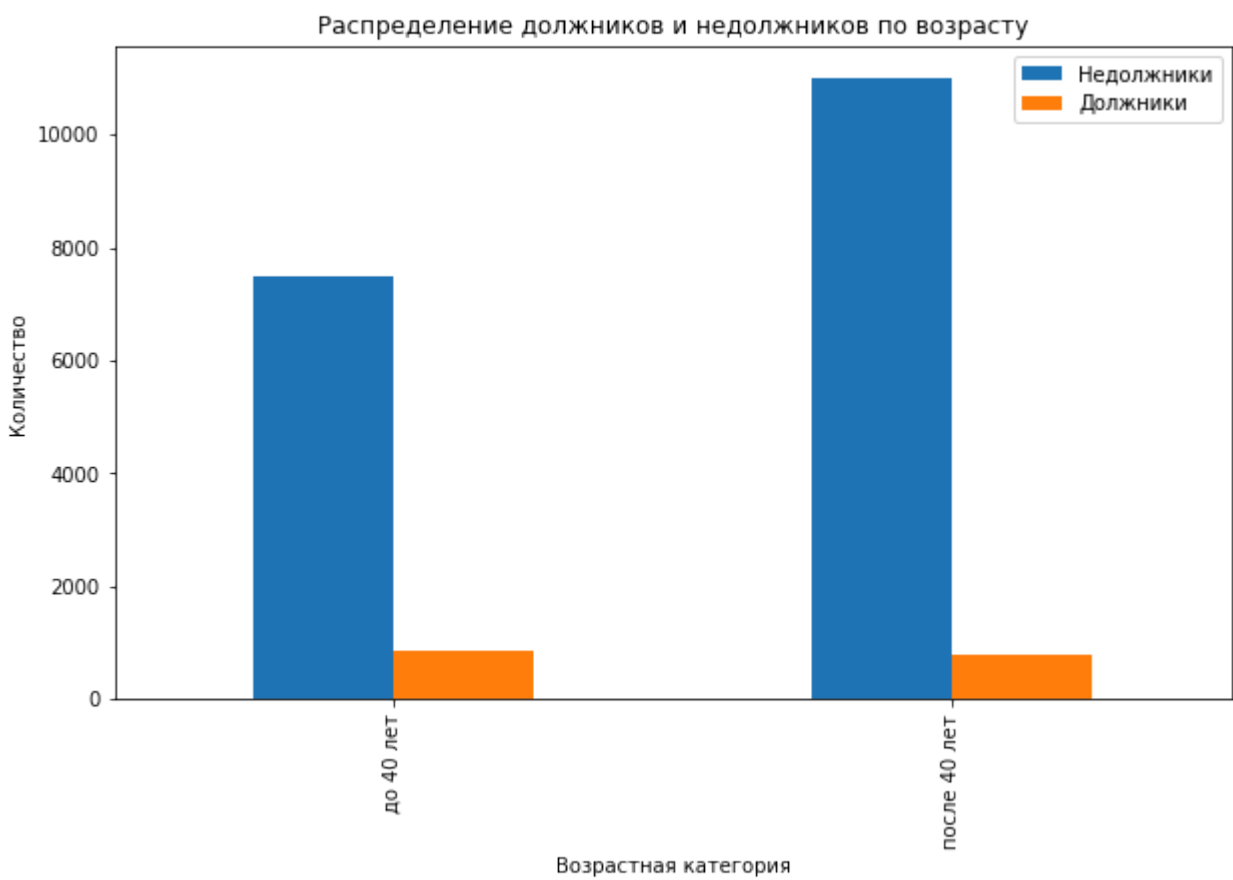


• **Возраст**

```
In [162... age_pivot = df.pivot_table(index='age_category', columns='debt', values='dob_years', aggfunc='count')
display(age_pivot)
age_pivot.plot(kind='bar', figsize=(10, 6))
plt.title('Распределение должников и недолжников по возрасту')
plt.xlabel('Возрастная категория')
plt.ylabel('Количество')
plt.legend(['Недолжники', 'Должники'])
```

	debt	
	0	1
age_category		
до 40 лет	7493	861
после 40 лет	11019	788

Out[162... <matplotlib.legend.Legend at 0x7fbabeaf4550>

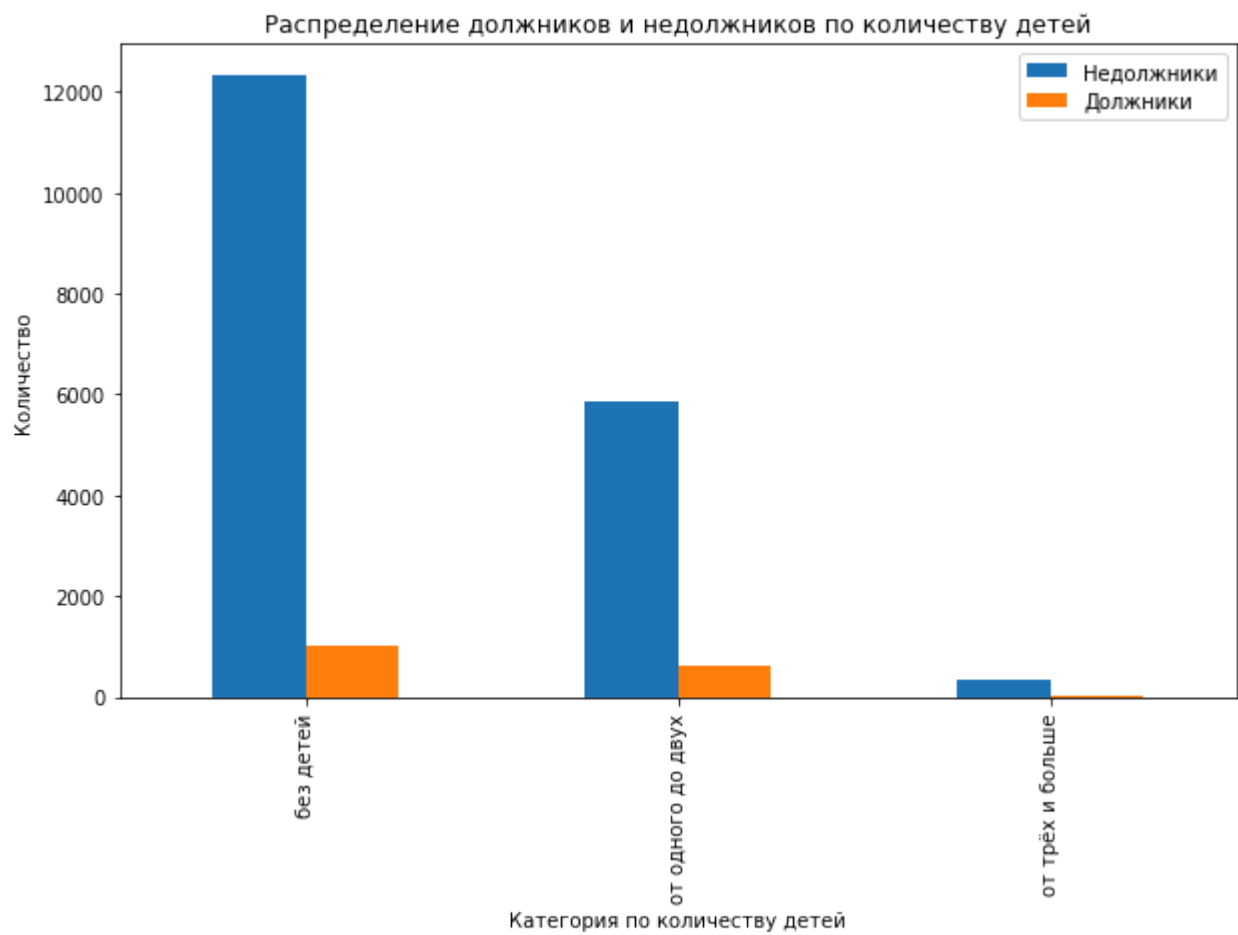


- **Количество детей**

```
In [163... children_pivot = df.pivot_table(index='childrens_category', columns='debt', values='children', aggfunc='count')
display(children_pivot)
children_pivot.plot(kind='bar', figsize=(10, 6))
plt.title('Распределение должников и недолжников по количеству детей')
plt.xlabel('Категория по количеству детей')
plt.ylabel('Количество')
plt.legend(['Недолжники', 'Должники'])
```

	debt	0	1
childrens_category			
без детей		12335	1011
от одного до двух		5856	608
от трёх и больше		321	30

Out[163... <matplotlib.legend.Legend at 0x7fbabe9fb850>



Проверьте исследовательскую гипотезу: у клиентов с детьми более высокий уровень финансовой ответственности и, следовательно, более низкий риск просрочек по кредиту.

1. Сравним количество должников и недолжников в зависимости от количества детей.
2. Построим график для сравнения

```
In [164... # Место для вашего кода
children_debt_pivot = df.pivot_table(index='childrens_category', columns='debt', values='children', aggfunc='count', fill_value=0)
children_debt_pivot
```

Out[164...

	debt	0	1
childrens_category			
	без детей	12335	1011
	от одного до двух	5856	608
	от трёх и больше	321	30

In [165...

```
# Вычислим долю должников в каждой категории
children_debt_pivot['debt_ratio'] = children_debt_pivot[1] / (children_debt_pivot[0] + children_debt_pivot[1])
children_debt_pivot
```

Out[165...

	debt	0	1	debt_ratio
childrens_category				
	без детей	12335	1011	0.075753
	от одного до двух	5856	608	0.094059
	от трёх и больше	321	30	0.085470

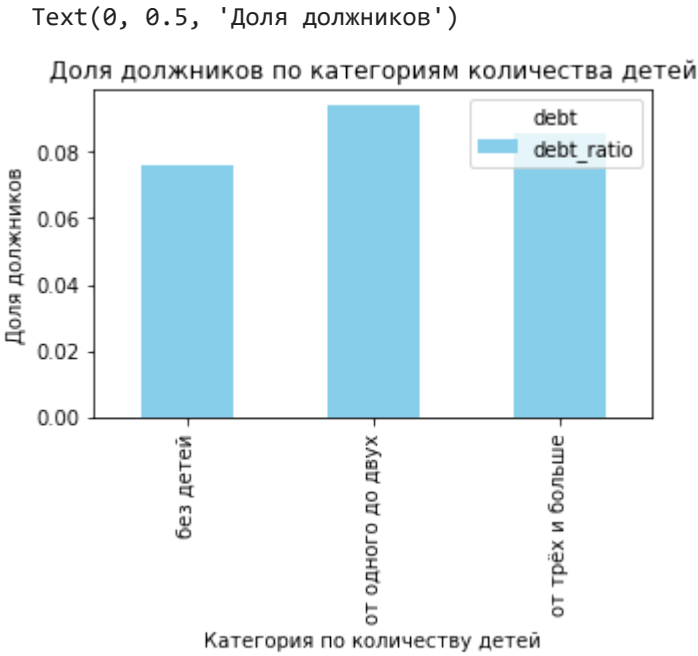
In [166...

```
# Вычислим долю должников в каждой категории
children_debt_pivot['debt_ratio'] = children_debt_pivot[1] / (children_debt_pivot[0] + children_debt_pivot[1])
```

In [167...

```
children_debt_pivot[['debt_ratio']].plot(kind='bar', figsize=(5, 3), color='skyblue')
plt.title('Доля должников по категориям количества детей')
plt.xlabel('Категория по количеству детей')
plt.ylabel('Доля должников')
```

Out[167...



Проверьте исследовательскую гипотезу: одинокие мужчины с низким доходом чаще оказываются должниками, чем семейные мужчины со средним доходом.

1. Нужно создать 2 группы
 - Одинокие мужчины с низким доходом.
 - Семейные мужчины со средним доходом.
2. Построить сводные таблицы
3. Построить графики визуализирующие итог

- выведем только те столбцы, что будем анализировать

In [168...

```
display(df[['gender', 'family_status', 'income_category', 'debt']].head(10))
```

	gender	family_status	income_category	debt
0	F	married	Высокий доход	0
1	F	married	Низкий доход	0
2	M	married	Средний доход	0
3	M	married	Высокий доход	0
4	F	civil partnership	Средний доход	0
5	M	civil partnership	Высокий доход	0
6	F	married	Высокий доход	0
7	M	married	Средний доход	0
8	F	civil partnership	Низкий доход	0
9	M	married	Средний доход	0

- проверим уникальные значения для столбцов gender и family_status

```
In [169... df['gender'].unique()
```

Out[169... array(['F', 'M', 'Unknown'], dtype=object)

```
In [170... df['family_status'].unique()
```

Out[170... array(['married', 'civil partnership', 'widow / widower', 'divorced', 'unmarried'], dtype=object)

- видно, что есть такой вариант как 'civil partnership', но в условии не оговоренно что считать сименйм мужчиной
- поэтому будем рассматривать только вариант 'married'

```
In [171... # Группы
single_low_income_men = df[(df['gender'] == 'M') & (df['family_status'] == 'unmarried') & (df['income_category'] == 'Низкий доход')]
display(single_low_income_men.head(10))
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose
479	0	-3029.321191	29	secondary education	1	unmarried	4	M	employee	0	17784.268	housing
834	0	-1523.564571	29	bachelor's degree	0	unmarried	4	M	employee	0	14129.326	purchase of the house
1085	0	-1618.549219	29	secondary education	1	unmarried	4	M	civil servant	0	16564.878	housing transactions
1277	0	-1205.259599	20	secondary education	1	unmarried	4	M	employee	0	14782.012	housing
1281	1	-318.559894	43	secondary education	1	unmarried	4	M	business	0	17122.443	housing renovation
1286	0	-5020.574409	35	secondary education	1	unmarried	4	M	employee	0	14321.866	real estate transactions
1344	0	-1418.055816	24	secondary education	1	unmarried	4	M	employee	1	17464.201	buying my own car
1430	0	-340.644655	27	secondary education	1	unmarried	4	M	employee	0	18332.241	building a property
1741	0	-405.802043	25	secondary education	1	unmarried	4	M	business	0	18242.696	buying my own car
1847	2	-679.171126	35	secondary education	1	unmarried	4	M	employee	0	16890.247	housing

```
In [172... married_medium_income_men = df[(df['gender'] == 'M') & (df['family_status'] == 'married') & (df['income_category'] == 'Средний доход' )]
display(married_medium_income_men.head(10))
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpose	i
2	0	-5623.422610	33	secondary education	1	married	0	M	employee	0	23341.752	purchase of the house	
7	0	-152.779569	50	secondary education	1	married	0	M	employee	0	21731.829	education	
9	0	-2188.756445	41	secondary education	1	married	0	M	employee	0	23108.150	purchase of the house for my family	
26	0	0.000000	41	secondary education	1	married	0	M	civil servant	0	23202.870	education	
60	1	-2534.462390	48	secondary education	1	married	0	M	employee	0	20784.365	to become educated	
66	0	-916.428829	28	secondary education	1	married	0	M	employee	0	23234.324	to become educated	
72	1	0.000000	32	bachelor's degree	0	married	0	M	civil servant	0	23202.870	transactions with commercial real estate	
76	1	-2252.192722	44	bachelor's degree	0	married	0	M	employee	0	23838.725	buying a second-hand car	
78	0	359722.945074	61	bachelor's degree	0	married	0	M	retiree	0	28020.423	purchase of a car	
83	0	0.000000	52	secondary education	1	married	0	M	employee	0	23202.870	housing	

In [173...

```
# Подсчет должников в каждой группе
single_low_income_men_debt_ratio = single_low_income_men['debt'].mean()
married_medium_income_men_debt_ratio = married_medium_income_men['debt'].mean()
print(f'Доля должников среди одиноких мужчин с низким доходом: {single_low_income_men_debt_ratio}')
print(f'Доля должников среди семейных мужчин со средним доходом: {married_medium_income_men_debt_ratio}')
```

Доля должников среди одиноких мужчин с низким доходом: 0.16939890710382513
Доля должников среди семейных мужчин со средним доходом: 0.09852717115286948

In [174...

```
# Сводные таблицы
single_low_income_men_pivot = single_low_income_men.pivot_table(index='family_status', columns='debt', values='total_income', aggfunc='count')
married_medium_income_men_pivot = married_medium_income_men.pivot_table(index='family_status', columns='debt', values='total_income', aggfunc='count')

display(single_low_income_men_pivot)
print('')
display(married_medium_income_men_pivot)
```

	debt	0	1
family_status			
unmarried	152	31	

	debt	0	1
family_status			
married	1775	194	

In [175...

```
import matplotlib.pyplot as plt

# Визуализация
labels = ['Одинокие мужчины с низким доходом', 'Семейные мужчины со средним доходом']
debt_ratios = [single_low_income_men_debt_ratio, married_medium_income_men_debt_ratio]

plt.figure(figsize=(10, 6))
plt.bar(labels, debt_ratios, color=['red', 'green'])
plt.title('Доля должников среди различных групп клиентов')
plt.xlabel('Группы клиентов')
plt.ylabel('Доля должников')
```

Out[175... Text(0, 0.5, 'Доля должников')



[* к содержанию](#)

Шаг 6. Выводы

- На основе построенных графиков в шаге 5 можно сделать следующие выводы
 - Уровень дохода
 - с увеличением дохода количество должников в основном снижается
 - наибольшее количество дллжников наблюдается среди людей снизким и очень низкими доходом
 - **то есть уровень дохода один из главных рисков возникнвоения задолжности**
 - Образование
 - основные должники срели людей со средним образованием
 - **чем выше уровень образования, тем меньше вероятность быть должником**
 - Возраст
 - в обеих категориях с одной стороны основнвая масса людей без задолжности, сдругой строны количество должников сопоставимо
 - однако после 40 недолжников существенно больше, что может быть связано с опытом и ростом финансовой грамотности
 - Количество детей

- среди клиентов без детей самое большое количество как должников, так и недолжников
- с детьми от 3 и выше количество должников существенно падает
- возможно наличие детей не позволяет быть более финансово рискованным

Распределение между должниками и недолжниками действительно отличается в зависимости от уровня дохода, образования, возраста и количества детей.

- Низкий доход и среднее образование увеличают вероятность задолженности
- Высшее образование и высокий уровень дохода плюс возраст от 40 снижают риск стать должником.

Вывод по гипотезе 1

- таблица и визуализация не подтверждают гипотезу о том, что клиенты с детьми имеют более высокий уровень финансовой ответственности.

Вывод по гипотезе 2

- Гипотеза о том, что одинокие мужчины с низким доходом чаще оказываются должниками, чем семейные мужчины со средним доходом, подтверждается
- **Одинокие мужчины с низким доходом имеют более высокую вероятность просрочить кредиту по сравнению с семейными мужчинами со средним доходом.**

Таким образом первая гипотеза не подтверждается, вторая гипотеза подтверждается

*** к содержанию**

Шаг 7. Проведение дополнительного исследования

(Необязательное задание) Исследуйте причины оформлнения кредита. Правда ли, что люди, которые брали кредит на образование, реже всего оказывались должниками?

In [176...

```
# Изучите уникальные значения в поле purpose
unique_purposes = df['purpose'].unique()
print('Уникальные значения в поле purpose:', unique_purposes)
```

Уникальные значения в поле purpose: ['purchase of the house' 'car purchase' 'supplementary education' 'to have a wedding' 'housing transactions' 'education' 'having a wedding' 'purchase of the house for my family' 'buy real estate' 'buy commercial real estate' 'buy residential real estate' 'construction of own property' 'property' 'building a property' 'buying a second-hand car' 'buying my own car' 'transactions with commercial real estate' 'building a real estate' 'housing' 'transactions with my real estate' 'cars' 'second-hand car purchase' 'getting an education' 'to become educated' 'car' 'wedding ceremony' 'to get a supplementary education' 'purchase of my own house' 'real estate transactions' 'to own a car' 'purchase of a car' 'profile education' 'university education' 'buying property for renting out' 'to buy a car' 'housing renovation' 'going to university' 'getting higher education']

делим клиентов на группы по образованию и остальным причинам

In [177...

```
# Функция для разделения на группы по цели кредита
def find_education_purpose(purpose):
    if 'education' in purpose.lower():
        return 'education'
    else:
        return 'other'

df['purpose_category'] = df['purpose'].apply(find_education_purpose)
```

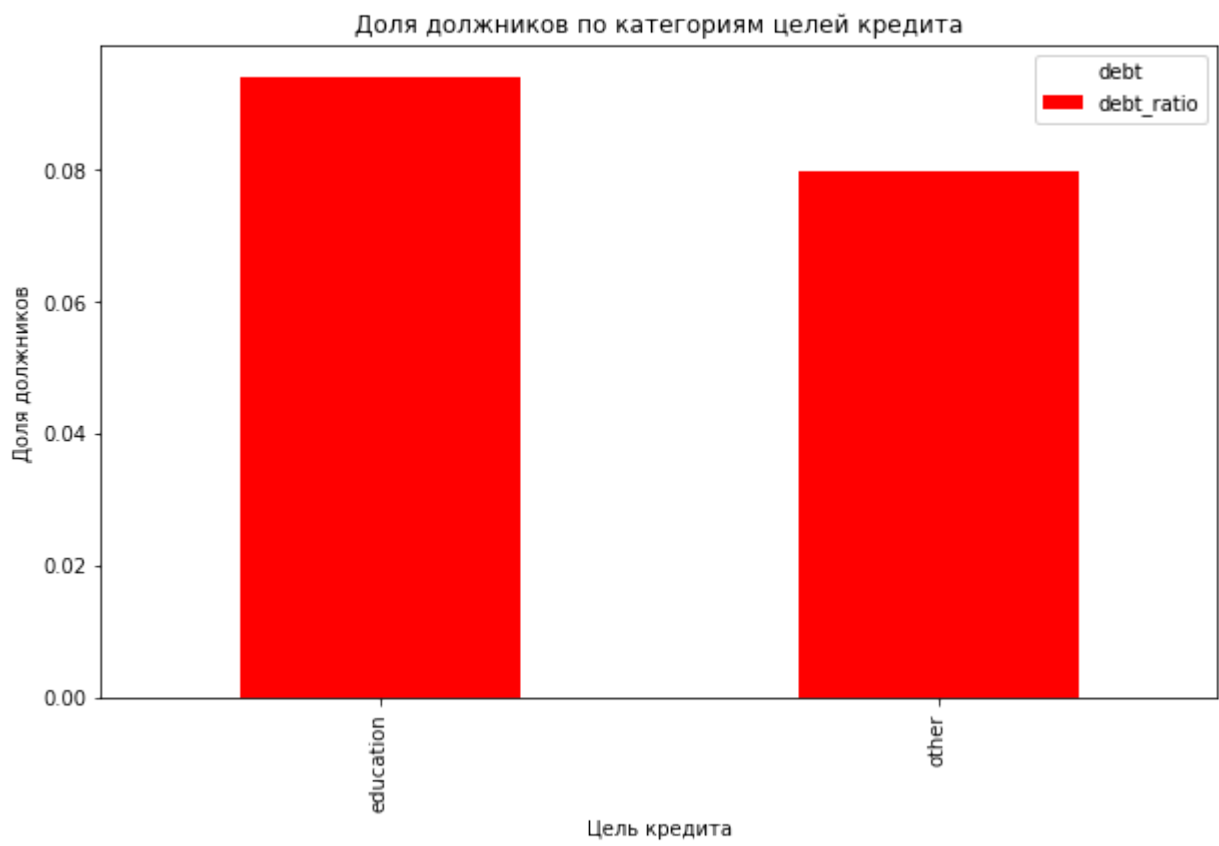
In [178...

```
purpose_debt_pivot = df.pivot_table(index='purpose_category', columns='debt', values='purpose', aggfunc='count', fill_value=0)

purpose_debt_pivot['debt_ratio'] = purpose_debt_pivot[1] / (purpose_debt_pivot[0] + purpose_debt_pivot[1])
display(purpose_debt_pivot)

purpose_debt_pivot[['debt_ratio']].plot(kind='bar', figsize=(10, 6), color='red')
plt.title('Доля должников по категориям целей кредита')
plt.xlabel('Цель кредита')
plt.ylabel('Доля должников')
plt.show()
```

	debt	0	1	debt_ratio
purpose_category				
	education	2644	275	0.094210
	other	15868	1374	0.079689



таким образом должников среди людей, бравших кредит на образование выше, чем по остальным причинам

- гипотеза не подтвержена

[* к содержанию](#)
