Survey of the reliablity of the bank clients

Credit department of the bank requested to analyse does the marital status and quantity of child are influence on the payment of oustandins fees in the specified duration in credit contract. Incoming data - statistic with credit score of the cilents.

The result of the survey will be used for the model of evalution of **credit score** - system wich evaluate the capacity of the client to pay the oustandins fees in the specified duration in credit contract

Step 1. Open the file and conduct the EDA

In [1]:
```
# import of libraries
import pandas as pd
from pymystem3 import Mystem
```

In [2]:
```
# read the data and asign it to table variable
table = pd.read_csv('/datasets/data.csv')

# print table info and first 10 rows
table.info()
table.head(10)
```

```
<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
```

Out[2]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purp
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875.639453	покупка жи
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080.014102	приобрете автомоб
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145885.952297	покупка жи
3	3	-4124.747207	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628.550329	дополнитель образова
4	0	340266.072047	53	среднее	1	гражданский брак	1	F	пенсионер	0	158616.077870	сыграть свад
5	0	-926.185831	27	высшее	0	гражданский брак	1	M	компаньон	0	255763.565419	покупка жи
6	0	-2879.202052	43	высшее	0	женат / замужем	0	F	компаньон	0	240525.971920	операц жил
7	0	-152.779569	50	СРЕДНЕЕ	1	женат / замужем	0	M	сотрудник	0	135823.934197	образова
8	2	-6929.865299	35	ВЫСШЕЕ	0	гражданский брак	1	F	сотрудник	0	95856.832424	на проведе свад
9	0	-2188.756445	41	среднее	1	женат / замужем	0	M	сотрудник	0	144425.938277	покупка жи для се

Conclusion

1) The table has 21525 rows и 12 columns.

2) In columns days_employed and total_income there are 2174 nulls, cleints without informtion on their income and occupation status. The total quanity of nuls almost 10% and sufficient for the dataset and overal statistic, therefore these data shall not be deleted.

3) In columns education there is difference in applying of register, these data shall be processed to have the nuqie formatting.

4) In columns days_employed and total_income data shall be procesed to get the understandable format for the further work and analysis.

Step 2. Data Preparation

Nulls processing

For the purpose of avoiding of the data loss, the nulls value the columns total_income and days_employed to be filled with mean value.

In [3]:
```
# fill the nulls values in column days_employed with mean
table['days_employed'] = table['days_employed'].fillna(table['days_employed'].mean())

#  reformatting the education column to have the lower cases
table['education'] = table['education'].str.lower()

#  decalre the function for the definition of age category
def age (age_value):
    if age_value < 35:
        return 'молодежь'
    elif 35<= age_value <=55:
        return 'средний возраст'
    elif age_value>55:
        return 'пожилого возраста'

# add the age category column to dataset
table['age_category'] = table['dob_years'].apply(age)

#  creating the datset with unique columns of age categories, education and type of income
all_unique = {'education': table['education'],'income_type':table['income_type'],'age_category': table['age_category']}
all_unique = pd.DataFrame(all_unique)
all_unique = all_unique.drop_duplicates().reset_index(drop=True)

#  declare a function for defenition of mean value for every row in table
def median_calc (row):
    B = round(table[(table['age_category'] == row['age_category']) & (table['income_type'] == row['income_type']) & (table['educ
    return B

# add the mean value to the table
all_unique['median'] = all_unique.apply(median_calc,axis=1)

# filling the nulls with mean value
all_unique['median'] = all_unique['median'].fillna(all_unique['median'].median())
```

```
# display the table
display(all_unique)
```

```
/opt/conda/lib/python3.9/site-packages/numpy/lib/nanfunctions.py:1117: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

	education	income_type	age_category	median
0	высшее	сотрудник	средний возраст	171441.00
1	среднее	сотрудник	средний возраст	138435.87
2	среднее	сотрудник	молодежь	131710.94
3	среднее	пенсионер	средний возраст	117894.96
4	высшее	компаньон	молодежь	191159.14
5	высшее	компаньон	средний возраст	215277.93
6	среднее	пенсионер	пожилого возраста	114200.79
7	неоконченное высшее	сотрудник	средний возраст	161129.20
8	высшее	компаньон	пожилого возраста	200807.92
9	высшее	сотрудник	молодежь	157505.97
10	среднее	госслужащий	средний возраст	134526.70
11	начальное	сотрудник	средний возраст	125560.37
12	среднее	компаньон	средний возраст	162140.52
13	среднее	компаньон	молодежь	154081.92
14	неоконченное высшее	сотрудник	молодежь	145919.11
15	неоконченное высшее	компаньон	средний возраст	215453.89
16	высшее	госслужащий	молодежь	160242.95
17	среднее	компаньон	пожилого возраста	164098.31
18	высшее	госслужащий	средний возраст	175589.24
19	высшее	пенсионер	средний возраст	160298.23
20	высшее	пенсионер	пожилого возраста	142307.34
21	среднее	пенсионер	молодежь	97620.69
22	среднее	сотрудник	пожилого возраста	140785.72
23	начальное	пенсионер	пожилого возраста	102598.65

	education	income_type	age_category	median
24	неоконченное высшее	компаньон	молодежь	159994.53
25	среднее	госслужащий	пожилого возраста	147036.63
26	высшее	сотрудник	пожилого возраста	173090.07
27	начальное	пенсионер	средний возраст	96330.47
28	среднее	госслужащий	молодежь	135753.54
29	начальное	сотрудник	молодежь	125994.91
30	начальное	компаньон	молодежь	132294.64
31	начальное	сотрудник	пожилого возраста	127021.00
32	неоконченное высшее	компаньон	пожилого возраста	213211.87
33	неоконченное высшее	пенсионер	средний возраст	111701.53
34	высшее	госслужащий	пожилого возраста	184722.74
35	неоконченное высшее	госслужащий	средний возраст	184062.64
36	неоконченное высшее	госслужащий	молодежь	150857.39
37	неоконченное высшее	сотрудник	пожилого возраста	229339.20
38	начальное	госслужащий	средний возраст	190912.18
39	начальное	компаньон	средний возраст	168979.94
40	неоконченное высшее	пенсионер	пожилого возраста	127205.09
41	ученая степень	пенсионер	пожилого возраста	177088.85
42	среднее	безработный	молодежь	59956.99
43	ученая степень	сотрудник	средний возраст	157259.90
44	высшее	пенсионер	молодежь	183556.36
45	начальное	госслужащий	пожилого возраста	79432.97
46	высшее	предприниматель	пожилого возраста	155670.91
47	ученая степень	сотрудник	пожилого возраста	268411.21

	education	income_type	age_category	median
48	начальное	госслужащий	молодежь	191021.14
49	высшее	студент	молодежь	98201.63
50	ученая степень	госслужащий	средний возраст	111392.23
51	высшее	безработный	средний возраст	202722.51
52	начальное	компаньон	пожилого возраста	96989.66
53	высшее	предприниматель	молодежь	499163.14
54	среднее	в декрете	средний возраст	53829.13

In [4]:
```
# creating a loop for the filling of nulls value of row total_income in table
for ind in all_unique.index:
    (table[(table['age_category']== all_unique['age_category'][ind])
           & (table['education']== all_unique['education'][ind])
           & (table['income_type']== all_unique['income_type'][ind])]) =(
        table[(table['age_category']== all_unique['age_category'][ind])
           & (table['education']== all_unique['education'][ind])
           & (table['income_type']== all_unique['income_type'][ind])]).fillna(all_unique['median'][ind])

# display the info on table dataset
table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 13 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
 12  age_category      21525 non-null  object
dtypes: float64(2), int64(5), object(6)
memory usage: 2.1+ MB
```

Conclusion

All the nulls value were fullfille the further work could be started.

The nulls value culd be in data set due to te absence of provided information from clients or income equeal to zero.

Data type update

In [5]:
```
# changing of data type of columns total_income and days_employed to int
table['total_income'] = table['total_income'].astype('int')
table['days_employed'] = table['days_employed'].astype('int')

# display the results
table.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21525 entries, 0 to 21524
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   children          21525 non-null  int64
 1   days_employed     21525 non-null  int64
 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  int64
 11  purpose           21525 non-null  object
 12  age_category      21525 non-null  object
dtypes: int64(7), object(6)
memory usage: 2.1+ MB
```

Conclusion

After the changing of the data types it would easier to work with it.

Based on the latest infomration the float data were changed succesfully to int.

For the replace of the data astype method was applied due to the fact that original data were float type and there was no any reason to get the information on the errors during the data type changing.

Duplicates processing

In [6]:
```python
# searhof duplicates
print('quantity of duplicates: ',table.duplicated().sum(),'\n')

# deletion of duplicates and reset of index
table = table.drop_duplicates().reset_index(drop=True)

# check of the result
table.info()
```

```
quantity of duplicates:  71

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21454 entries, 0 to 21453
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   children          21454 non-null  int64 
 1   days_employed     21454 non-null  int64 
 2   dob_years         21454 non-null  int64 
 3   education         21454 non-null  object
 4   education_id      21454 non-null  int64 
 5   family_status     21454 non-null  object
 6   family_status_id  21454 non-null  int64 
 7   gender            21454 non-null  object
 8   income_type       21454 non-null  object
 9   debt              21454 non-null  int64 
 10  total_income      21454 non-null  int64 
 11  purpose           21454 non-null  object
 12  age_category      21454 non-null  object
dtypes: int64(7), object(6)
memory usage: 2.1+ MB
```

Conclusion

The deletion of duplicates was made using method drop_duplicates w/o specification of exact columns for the deletion of duplicates in all rows, additionaly indexes were reseted.

Duplicates most likely appeared in dataset due to the mistakes during the creating a new records in table - human factor.

Lemmatization

In [7]:
```python
# assigning of Mystem() to variable 'm'
m = Mystem()

# lemmatization of row purpose and add lemmatized text as new row to dataset
table['purpose_category'] = table['purpose'].apply(lambda x: m.lemmatize(x))
```

In [8]:
```python
#     Creation of two varaeblies. first - values of column purpose_category, second - to save the unique value of column
var_one = table['purpose_category']
lem_unique = []
```

```
for i in var_one:
    if i not in lem_unique:
        lem_unique.append(i)
print(lem_unique)
```

```
[['покупка', ' ', 'жилье', '\n'], ['приобретение', ' ', 'автомобиль', '\n'], ['дополнительный', ' ', 'образование', '\n'], ['сыг
рать', ' ', 'свадьба', '\n'], ['операция', ' ', 'с', ' ', 'жилье', '\n'], ['образование', '\n'], ['на', ' ', 'проведение', ' ',
'свадьба', '\n'], ['покупка', ' ', 'жилье', ' ', 'для', ' ', 'семья', '\n'], ['покупка', ' ', 'недвижимость', '\n'], ['покупка',
' ', 'коммерческий', ' ', 'недвижимость', '\n'], ['покупка', ' ', 'жилой', ' ', 'недвижимость', '\n'], ['строительство', ' ', 'с
обственный', ' ', 'недвижимость', '\n'], ['недвижимость', '\n'], ['строительство', ' ', 'недвижимость', '\n'], ['на', ' ', 'поку
пка', ' ', 'подержать', ' ', 'автомобиль', '\n'], ['на', ' ', 'покупка', ' ', 'свой', ' ', 'автомобиль', '\n'], ['операция', '
', 'с', ' ', 'коммерческий', ' ', 'недвижимость', '\n'], ['строительство', ' ', 'жилой', ' ', 'недвижимость', '\n'], ['жилье',
'\n'], ['операция', ' ', 'со', ' ', 'свой', ' ', 'недвижимость', '\n'], ['автомобиль', '\n'], ['заниматься', ' ', 'образование',
'\n'], ['сделка', ' ', 'с', ' ', 'подержанный', ' ', 'автомобиль', '\n'], ['получение', ' ', 'образование', '\n'], ['свадьба',
'\n'], ['получение', ' ', 'дополнительный', ' ', 'образование', '\n'], ['покупка', ' ', 'свой', ' ', 'жилье', '\n'], ['операци
я', ' ', 'с', ' ', 'недвижимость', '\n'], ['получение', ' ', 'высокий', ' ', 'образование', '\n'], ['свой', ' ', 'автомобиль',
'\n'], ['сделка', ' ', 'с', ' ', 'автомобиль', '\n'], ['профильный', ' ', 'образование', '\n'], ['высокий', ' ', 'образование',
'\n'], ['покупка', ' ', 'жилье', ' ', 'для', ' ', 'сдача', '\n'], ['на', ' ', 'покупка', ' ', 'автомобиль', '\n'], ['ремонт', '
', 'жилье', '\n'], ['заниматься', ' ', 'высокий', ' ', 'образование', '\n']]
```

Conclusion

Based on the lemmatization we can get 4 main lemms: 1) свадьба 2) автомобиль 3) образование 4) недвижимость

Data categorization

In [9]:
```
# declaration of function for categorization of data in columns purpose_category by 4 main categories
def category (input_value):
    if 'свадьба' in input_value:
        return 'свадьба'
    elif 'автомобиль' in input_value:
        return 'автомобиль'
    elif 'образование' in input_value:
        return 'образование'
    else:
        return 'недвижимость'

# declaration of function for categorization by presence/absence of debt
def credit_debt (input_value):
    if input_value == 0:
        return 'долга нет'
    else:
```

```
        return 'имеется долг'

# declaration of function for categorization by presence/absence of child
def kids (input_value):
    if input_value == 0:
        return 'без детей'
    else:
        return 'есть дети'

# apply of declared fucntion
table['purpose_category'] = table['purpose_category'].apply(category)
table['debt_status'] = table['debt'].apply(credit_debt)
table['kids_status'] = table['children'].apply(kids)

# chechk of the results
table.head(10)
```

Out[9]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	total_income	purpo
0	1	-8437	42	высшее	0	женат / замужем	0	F	сотрудник	0	253875	покупка жил
1	1	-4024	36	среднее	1	женат / замужем	0	F	сотрудник	0	112080	приобретен автомоби
2	0	-5623	33	среднее	1	женат / замужем	0	M	сотрудник	0	145885	покупка жил
3	3	-4124	32	среднее	1	женат / замужем	0	M	сотрудник	0	267628	дополнительн образован
4	0	340266	53	среднее	1	гражданский брак	1	F	пенсионер	0	158616	сыграть свадь
5	0	-926	27	высшее	0	гражданский брак	1	M	компаньон	0	255763	покупка жил
6	0	-2879	43	высшее	0	женат / замужем	0	F	компаньон	0	240525	операци жиль
7	0	-152	50	среднее	1	женат / замужем	0	M	сотрудник	0	135823	образован
8	2	-6929	35	высшее	0	гражданский брак	1	F	сотрудник	0	95856	на проведен свадь
9	0	-2188	41	среднее	1	женат / замужем	0	M	сотрудник	0	144425	покупка жил для сем

Conslusion

For the completion of catogerization by column purpose_category were selected 4 main lemms because one of such values is in every row of this column.

For other categorization were selected absence/presence of child and absence/presence debt for the further data analysis and answering on the questions.

Step 3. Question answers

Is there a dependency between the presence of a child and payment of outstanding fees in the specified duration in the credit contract?

In [10]:
```
# creation of a new dataset equal to "table"
kids_kredit_check = table.copy()

# creation of new columns with status of presence of the child and debt and group by it's values
kids_kredit_check['category_kids_kredit'] = kids_kredit_check['kids_status']+', '+kids_kredit_check['debt_status']
kids_kredit_check = kids_kredit_check.groupby('category_kids_kredit')['debt'].count()

# display the results
display(kids_kredit_check)

# checking that all data is included
print(kids_kredit_check.sum()==table['debt'].count(),'\n')

# calculation of percentages by each category
precent_withno_kids = kids_kredit_check.iloc[1]/(kids_kredit_check.iloc[0]+kids_kredit_check.iloc[1])
precent_with_kids = kids_kredit_check.iloc[3]/(kids_kredit_check.iloc[2]+kids_kredit_check.iloc[3])

# display of the results
print('Процент клиентов с детьми, которые имеют долг: {:.2%}'.format(precent_with_kids))
print('Процент клиентов без детей, которые имеют долг: {:.2%}'.format(precent_withno_kids))
```

```
category_kids_kredit
без детей, долга нет       13028
без детей, имеется долг     1063
есть дети, долга нет        6685
есть дети, имеется долг      678
Name: debt, dtype: int64
True

Процент клиентов с детьми, которые имеют долг: 9.21%
Процент клиентов без детей, которые имеют долг: 7.54%
```

Conclusion

- Based on the information from bank there is a dependency: client with a child on 1.7% frequently will not pay the outstanding fees in the specified duration in the credit contract.
- However the data has only 21454 records, most likely the sufficient increase of the records could influnce on the result of the statistic.

Is there a dependency on the marital status and payment of outstanding fees in the specified duration in the credit contract?

In [11]:
```
# creation of new columns with marital status and debt
table['marriage_debt_status'] = table['family_status']+', '+table['debt_status']

# group by column marriage_debt_status
marriage_check = table.groupby('marriage_debt_status')['debt'].count()

# display the results
display(marriage_check)

# checking that all data is included
print(marriage_check.sum()==table['debt'].count(),'\n')

# calculation of percentages by each category
precent_not_married = marriage_check.iloc[1]/(marriage_check.iloc[0]+marriage_check.iloc[1])
precent_divorced = marriage_check.iloc[3]/(marriage_check.iloc[2]+marriage_check.iloc[3])
precent_widow = marriage_check.iloc[5]/(marriage_check.iloc[4]+marriage_check.iloc[5])
precent_civil_partners = marriage_check.iloc[7]/(marriage_check.iloc[6]+marriage_check.iloc[7])
precent_married = marriage_check.iloc[9]/(marriage_check.iloc[8]+marriage_check.iloc[9])

# display of the results
print('Процент клиентов в браке, которые имеют долг: {:.2%}'.format(precent_married))
print('Процент клиентов, которые состоят в гражданском браке и имеют долг: {:.2%}'.format(precent_civil_partners))
print('Процент разведенных клиентов, которые имеют долг: {:.2%}'.format(precent_divorced))
print('Процент овдовевших клиентов, которые имеют долг: {:.2%}'.format(precent_widow))
print('Процент клиентов, которые не состоят в браке и имеют долг: {:.2%}'.format(precent_not_married))
```

```
marriage_debt_status
Не женат / не замужем, долга нет        2536
Не женат / не замужем, имеется долг      274
в разводе, долга нет                    1110
в разводе, имеется долг                   85
вдовец / вдова, долга нет                896
вдовец / вдова, имеется долг              63
гражданский брак, долга нет              3763
гражданский брак, имеется долг            388
женат / замужем, долга нет              11408
женат / замужем, имеется долг             931
Name: debt, dtype: int64
```

```
True

Процент клиентов в браке, которые имеют долг: 7.55%
Процент клиентов, которые состоят в гражданском браке и имеют долг: 9.35%
Процент разведенных клиентов, которые имеют долг: 7.11%
Процент овдовевших клиентов, которые имеют долг: 6.57%
Процент клиентов, которые не состоят в браке и имеют долг: 9.75%
```

Conclusion

- The lowest percentage of clients with debt overdue - it's widowed clients- 6,5%.
- Than divorsed cliens - 7,1%.
- Only 7.55% of clients who married have debt verdue.
- the highest percentage of debt overdue have clients who is not married or informal married - 9,75% and 9,35%

Is there a dependency on the income grade and payment of outstanding fees in the specified duration in the credit contract?

In [12]:
```python
# calculation of average income and categorization of clients
range_groups = pd.qcut(table['total_income'],q=3)
range_groups = range_groups.drop_duplicates().sort_values().reset_index(drop=1)
print(range_groups)

# function declare for definition of income grade
def income (income_value):
    if income_value in range_groups[0]:
        return 'доход ниже среднего'
    elif income_value in range_groups[1]:
        return 'доход средний'
    else:
        return 'доход выше среднего'

# add a column with income grade
table['income_debt_status'] = table['total_income'].apply(income)

#  calculation of percentages by each category
table.groupby('income_debt_status')['debt'].agg(['count', 'sum', lambda x: '{:.2%} '.format(x.mean())])
```

```
0    (20666.999, 119869.0]
1     (119869.0, 173597.0]
2    (173597.0, 2265604.0]
Name: total_income, dtype: category
Categories (3, interval[float64]): [(20666.999, 119869.0] < (119869.0, 173597.0] < (173597.0, 2265604.0]]
```

Out[12]:

income_debt_status	count	sum	<lambda_0>
доход выше среднего	7151	526	7.36%
доход ниже среднего	7152	580	8.11%
доход средний	7151	635	8.88%

Conclusion

- Clients with average grade income are more often will not pay the outstanding fees in the specified duration in the credit contract (8.88%).
- Clients with high income grade - has the smallest precentage of debtors that exceed the credit time limit 7%.
- Amongst of clients with lower income grade - 8% of such debtors

---

How does the different credite purpose influence on the payment of the outstanding fees in the specified duration in the credit contract?

In [13]:

```
# add a new column purpose_debt_status (concat of debt_status and purpose category)
table['purpose_debt_status'] = table['purpose_category']+', '+table['debt_status']

# group by new column
purpose_check = table.groupby('purpose_debt_status')['debt'].count()

# display the result
display(purpose_check)

# check that all data is included
print(purpose_check.sum()==table['debt'].count(),'\n')

# calculation of percentages by each category
```

```
precent_auto = purpose_check.iloc[1]/(purpose_check.iloc[0]+purpose_check.iloc[1])
precent_realty = purpose_check.iloc[3]/(purpose_check.iloc[2]+purpose_check.iloc[3])
precent_education_= purpose_check.iloc[5]/(purpose_check.iloc[4]+purpose_check.iloc[5])
precent_marriage = purpose_check.iloc[7]/(purpose_check.iloc[6]+purpose_check.iloc[7])

# display of the resutlts
print('Процент клиентов взявшие кредит на автомобиль, которые имеют долг: {:.2%}'.format(precent_auto))
print('Процент клиентов взявшие кредит на недвижимость, которые имеют долг: {:.2%}'.format(precent_realty))
print('Процент клиентов взявшие кредит на образование, которые имеют долг: {:.2%}'.format(precent_education_))
print('Процент клиентов взявшие кредит на свадьбу, которые  имеют долг: {:.2%}'.format(precent_marriage))
```

```
purpose_debt_status
автомобиль, долга нет          3903
автомобиль, имеется долг        403
недвижимость, долга нет       10029
недвижимость, имеется долг      782
образование, долга нет         3643
образование, имеется долг       370
свадьба, долга нет             2138
свадьба, имеется долг           186
Name: debt, dtype: int64
True

Процент клиентов взявшие кредит на автомобиль, которые имеют долг: 9.36%
Процент клиентов взявшие кредит на недвижимость, которые имеют долг: 7.23%
Процент клиентов взявшие кредит на образование, которые имеют долг: 9.22%
Процент клиентов взявшие кредит на свадьбу, которые  имеют долг: 8.00%
```

Conclusion

- Only 7% of clients who got credit on purchase of real estate has debt overdue.
- 8% of clients who got credit on wedding has debt overdue.
- The gighest percentaage of clients who has debt overdue got credit on auto or education (9,3% и 9,2%)

Step 4. General conclusion

During data analysis the following infomration was obtained:

Most relaible clients:

- Clients who's got credit on the purchase of real estate (only 7.23% debt overdue)
- Clients with marital status - widow / er(6.57% debt overdue)
- Clinets w/o kids (7.54% debt overdue)
- Clients with income bove average (7.36% debt overdue)

Unrelaible clients:

- Clients who's got credit on purchasing of auto (9.36% debt overdue)
- Clients who were not married (9.75% debt overdue)
- Clients with child (9.21% debt overdue)
- Clients with average grade of income (8.88% debt overdue)

In [ ]: