Шаг 1. Обзор данных

```
import pandas as pd
import warnings
warnings.simplefilter("ignore")
```

```
# посмотреть, где находится каталог с файлами на COLAB
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
```

```
# получить доступ к каталогу и уточнить названия папок
import os
os.listdir('/content/drive/My Drive/Colab Notebooks/Яндекс/Проект 2 Заемщик')
```

```
['data.csv', 'Начало', 'Исследование надежности заемщиков.ipynb', 'GitHub']
```

```
# присвоить переменной путь к файлу в каталоге
path = "/content/drive/My Drive/Colab Notebooks/Яндекс/Проект 2 Заемщик/data.csv"
```

```
data = pd.read_csv(path)
```

```
# просмотреть данные (первые 5)
data.head()
```

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	gender	income_type	debt	tot
0	1	-8437.673028	42	высшее	0	женат / замужем	0	F	сотрудник	0	253
1	1	-4024.803754	36	среднее	1	женат / замужем	0	F	сотрудник	0	112
2	0	-5623.422610	33	Среднее	1	женат / замужем	0	M	сотрудник	0	145
3	3	-4124.747207	32	среднее	1	женат /	0	M	сотрудник	0	267

```
# посчитать пропущенные данные
data.isnull().sum()
```

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

```
# вывод общей информации и просмотр типов данных
data.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
```

2174/21525*100%=10% - это процент пропусков в столбцах "days_employed" и "total_income" Так как доля некорректных данных более 1% их необходимо обработать и привести в соответствие остальным.

```
# вывод описательной статистики
data.describe()
```

	children	days_employed	dob_years	education_id	family_status_id	debt	total_income
count	21525.000000	19351.000000	21525.000000	21525.000000	21525.000000	21525.000000	1.935100e+04
mean	0.538908	63046.497661	43.293380	0.817236	0.972544	0.080883	1.674223e+05
std	1.381587	140827.311974	12.574584	0.548138	1.420324	0.272661	1.029716e+05
min	-1.000000	-18388.949901	0.000000	0.000000	0.000000	0.000000	2.066726e+04
25%	0.000000	-2747.423625	33.000000	1.000000	0.000000	0.000000	1.030532e+05
50%	0.000000	-1203.369529	42.000000	1.000000	0.000000	0.000000	1.450179e+05
75%	1.000000	-291.095954	53.000000	1.000000	1.000000	0.000000	2.034351e+05
max	20.000000	401755.400475	75.000000	4.000000	4.000000	1.000000	2.265604e+06

Шаг 2.1 Заполнение пропусков

```
# подсчитываем медианное значение
median_days_employed = data['days_employed'].median()
print(median_days_employed)
```

```
-1203.369528770489
```

Медианное значение отрицательное, это говорит о наличии большого количества отрицательных значений в стаже работы.

```
# цикл замещающий все отрицательные значения на положительные
for i in range(len(data['days_employed'])):
    if data['days_employed'][i] < 0:
        data['days_employed'][i] = data['days_employed'][i] * (-1)
```

```
# подсчитываем медианное значение
median_days_employed = data['days_employed'].median()
print(median_days_employed)
```

```
2194.220566878695
```

```
# заполнение медианными значениями столбца "days_employed"
data['days_employed'] = data['days_employed'].fillna(median_days_employed)
```

```
# проверка на наличие отрицательных значений в столбце
count_total_income = 0
for i in range(len(data['total_income'])):
    if data['total_income'][i] < 0:
        count_total_income += 1
print(count_total_income)
```

```
0
```

```
# подсчитываем медианное значение
median_total_income = data['total_income'].median()
print(median_total_income)
```

```
145017.93753253992
```

```
# заполнение медианными значениями
data['total_income'] = data['total_income'].fillna(median_total_income)
```

```
# контрольная проверка на пропуски
data.isnull().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
dtype: int64
```

Шаг 2.2 Проверка данных на аномалии

```
# поиск дополнительных, некорректных данных
data['children'].value_counts(ascending=False)
```

```
 0     14149
 1      4818
```

```
 2      2055
 3       330
 20       76
-1        47
 4        41
 5         9
Name: children, dtype: int64
```

```
# подсчет  доли (share_20) некорректных данных от общего объема
share_20 = data[data['children'] == 20]['children'].count() / len(data) * 100
print(share_20)
```

```
0.3530778164924506
```

```
# подсчет  доли (share_minus_one) некорректных данных от общего объема
share_minus_one = data[data['children'] == -1]['children'].count() / len(data) * 100
print(share_minus_one)
```

```
0.2183507549361208
```

```
# подсчет общей доли (share_20 + share_minus_one) некорректных данных от общего объема
total_share_emissions = share_minus_one + share_20
print(total_share_emissions)
```

```
0.5714285714285714
```

Доля некорректных данных менее 1%, поэтому их удаление допустимо.

```
# удаление строк со значением "20" и "-1" детей
data = data[(data['children'] != 20) & (data['children'] != -1)]
```

Шаг 2.3. Изменение типов данных

```
# замена вещественных типов данных на целочисленные
data['total_income'] = data['total_income'].astype('int')
```

```
#  проверка результатов замены
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21402 entries, 0 to 21524
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   children          21402 non-null  int64 
 1   days_employed     21402 non-null  float64
 2   dob_years         21402 non-null  int64 
 3   education         21402 non-null  object
 4   education_id      21402 non-null  int64 
 5   family_status     21402 non-null  object
 6   family_status_id  21402 non-null  int64 
 7   gender            21402 non-null  object
 8   income_type       21402 non-null  object
 9   debt              21402 non-null  int64 
 10  total_income      21402 non-null  int64 
 11  purpose           21402 non-null  object
dtypes: float64(1), int64(6), object(5)
memory usage: 2.1+ MB
```

Шаг 2.4. Удаление дубликатов

```
# подсчет доли дубликатов
print(data.duplicated().sum() / len(data) * 100)
```

```
0.2523128679562658
```

Доля дубликатов в данных менее 1%, поэтому их удаление допустимо.

```
# удаление дубликатов
data = data.drop_duplicates()
```

```
# перевод строковых значений в нижний регистр с целью поиска неявных дубликатов
data['education'] = data['education'].str.lower()
data['family_status'] = data['family_status'].str.lower()
data['income_type'] = data['income_type'].str.lower()
data['purpose'] = data['purpose'].str.lower()
```

```
# подсчет доли дубликатов
print(data.duplicated().sum() / len(data) * 100)
```

```
0.07963275248266817
```

Доля дубликатов в данных менее 1%, поэтому их удаление допустимо.

```
# удаление дубликатов
data = data.drop_duplicates()
```

```
# вывод количества дубликатов
print ('Дубликатов в таблице:', data.duplicated().sum())
```

```
Дубликатов в таблице: 0
```

Шаг 2.5. Формирование дополнительных датафреймов словарей, декомпозиция исходного датафрейма

```
# сохранение исходного датафрейма с приставкой 'old'
old_data = data
```

```
# создание датафрейма на основе столбцов "education", "education_id"
education_data = data[['education', 'education_id']]

# создание датафрейма на основе столбцов "family_status", "family_status_id"
family_status_data = data[['family_status', 'family_status_id']]
```

```
# удаление столбцов "education", "family_status"
data = data.drop(['education', 'family_status'], axis=1)
```

```
# обновление индексов с одновременным удалением старых
data.reset_index(drop=True)
```

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purp
0	1	8437.673028	42	0	0	F	сотрудник	0	253875	покупка жи
1	1	4024.803754	36	1	0	F	сотрудник	0	112080	приобрете автомоби
2	0	5623.422610	33	1	0	M	сотрудник	0	145885	покупка жи
3	3	4124.747207	32	1	0	M	сотрудник	0	267628	дополнитель образова
4	0	340266.072047	53	1	1	F	пенсионер	0	158616	сыграть свад
...	...	...	...	...	...	...	...	...	...	
21326	1	4529.316663	43	1	1	F	компаньон	0	224791	операци жиль
21327	0	343937.404131	67	1	0	F	пенсионер	0	155999	сдел автомоби
21328	1	2113.346888	38	1	1	M	сотрудник	1	89672	недвижимо
										на поку

Шаг 2.6. Категоризация дохода

```
# функция категоризации по доходу
def total_income_group(total_income):
    if total_income <= 30000:
        return 'E'
    if total_income <= 50000:
        return 'D'
    if total_income <= 200000:
        return 'C'
    if total_income <= 1000000:
        return 'B'
    return 'A'
```

```
# создание столбца с категорией по доходу
data['total_income_category'] = data['total_income'].apply(total_income_group)
```

```
data.head()
```

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purpose
0	1	8437.673028	42	0	0	F	сотрудник	0	253875	покупка жилья
1	1	4024.803754	36	1	0	F	сотрудник	0	112080	приобретение автомобиля
2	0	5623.422610	33	1	0	M	сотрудник	0	145885	покупка жилья
3	3	4124.747207	32	1	0	M	сотрудник	0	267628	дополнительное образование

Шаг 2.7. Категоризация целей кредита

```
# вывод уникальных значений по столбцу "purpose"
data['purpose'].value_counts()
```

```
свадьба                                  790
на проведение свадьбы                    763
сыграть свадьбу                          760
операции с недвижимостью                 672
покупка коммерческой недвижимости        658
покупка жилья для сдачи                  649
операции с жильем                        647
операции с коммерческой недвижимостью    645
жилье                                    641
покупка жилья                            640
покупка жилья для семьи                  637
недвижимость                             631
строительство собственной недвижимости   628
операции со своей недвижимостью          623
строительство жилой недвижимости         620
строительство недвижимости               619
покупка своего жилья                     619
покупка недвижимости                     616
ремонт жилью                             604
покупка жилой недвижимости               602
на покупку своего автомобиля             504
заняться высшим образованием             496
автомобиль                               491
сделка с подержанным автомобилем         481
автомобили                               476
свой автомобиль                          473
на покупку подержанного автомобиля       471
на покупку автомобиля                    469
приобретение автомобиля                  459
сделка с автомобилем                     455
дополнительное образование               455
высшее образование                       446
получение дополнительного образования    444
образование                              442
получение образования                    440
профильное образование                   432
получение высшего образования            425
```

```
заняться образованием                    408
Name: purpose, dtype: int64
```

```
# функция категоризации по целевому предназначению кредита
def purpose_group(purpose):
    if 'автомоб' in purpose:
        return 'операции с автомобилем'
    if ('недвижим' in purpose) or ('жил' in purpose):
        return 'операции с недвижимостью'
    if 'свад' in purpose:
        return 'проведение свадьбы'
    return 'получение образования'
```

```
# создание столбца с категорией целевого предназначения кредита
data['purpose_category'] = data['purpose'].apply(purpose_group)
```

```
# вывод 5 строк датафрейма (проверка наличия столбца с категорией целевого предназначения кредита)
data.head()
```

children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt	total_income	purpose

Исследовательский анализ

автомобиля

I. Исследование о наличии зависимости между количеством детей и возвратом кредита в срок

```
# выделение 2-х столбцов из общего датафрейма для проведения исследования о наличии зависимости между количеством детей и возвратом к
children_debt = data[['children', 'debt']]
```

```
# создание пустого датафрейма для внесения расчетных данных
addiction_children_debt = pd.DataFrame()
```

```
# создание столбца с значениями количества заемщиков по соответствующей группе (0, 1, 2, 3, 4, 5 - детей соответственно)
addiction_children_debt['children_count'] = children_debt.groupby('children')['debt'].count()
```

```
# создание столбца с количеством просрочек в каждой группе заемщиков
addiction_children_debt['count_debt'] = children_debt.groupby('children')['debt'].sum()
```

```
# создание столбца с отношением количества просрочек к количеству людей по соответствующей группе
addiction_children_debt['share_debt'] = addiction_children_debt['count_debt'] / addiction_children_debt['children_count']
```

```
# создание столбца с количеством детей в соответствующей группе
addiction_children_debt['children'] = addiction_children_debt.index
```

```
# вывод результирующей таблицы проведенных расчетов
addiction_children_debt
```

children	children_count	count_debt	share_debt	children
0	14091	1063	0.075438	0
1	4808	444	0.092346	1
2	2052	194	0.094542	2
3	330	27	0.081818	3
4	41	4	0.097561	4

```
# расчет коэффициента корреляции между количеством детей и наличием просрочек
cor_addiction_children_debt = addiction_children_debt[['children', 'share_debt']]
cor_addiction_children_debt.corr()
```

	children	share_debt
children	1.000000	-0.540416
share_debt	-0.540416	1.000000

- Вывод:

  Взаимосвязь между количеством детей и возвратом кредита в срок существует. Кроме того, необходимо отметить, что в целом бездетные допускают меньшую долю просрочек.

- II. Исследование о наличии зависимости между семейным положением и возвратом кредита в срок

```
# выделение 2-х столбцов из общего датафрейма для проведения исследования о наличии зависимости между семейным положением и возвратом
family_status_debt = old_data[['family_status', 'debt']]
```

```
# создание пустого датафрейма для внесения расчетных данных
addiction_family_status_debt = pd.DataFrame()
```

```
# создание столбца с значениями количества заемщиков по соответствующей группе (женат, холост и т.д.)
addiction_family_status_debt['family_status_count'] = family_status_debt.groupby('family_status')['debt'].count()
```

```
# создание столбца с количеством просрочек в каждой группе заемщиков
addiction_family_status_debt['count_debt'] = family_status_debt.groupby('family_status')['debt'].sum()
```

```
# создание столбца с отношением количества просрочек к количеству людей по соответствующей группе
addiction_family_status_debt['share_debt'] = addiction_family_status_debt['count_debt'] / addiction_family_status_debt['family_status
```

```
# вывод результирующей таблицы проведенных расчетов
addiction_family_status_debt
```

family_status	family_status_count	count_debt	share_debt
в разводе	1189	84	0.070648
вдовец / вдова	951	63	0.066246
гражданский брак	4134	385	0.093130
женат / замужем	12261	927	0.075606
не женат / не замужем	2796	273	0.097639

▼ Вывод:

Граждане женатые / замужем и находящиеся в разводе - более дисциплинированны в плане возврата кредита, а вот не женатые / не замужем и находящиеся в гражданском браке - менее. Кроме того, стоит подчеркнуть, что самыми дисциплинированными

являются овдовевшие.

III. Исследование о наличии зависимости между уровнем дохода и возвратом кредита в срок

```
# выделение 2-х столбцов из общего датафрейма для проведения исследования о наличии зависимости между уровнем дохода и возвратом кред
total_income_category_debt = data[['total_income_category', 'debt']]
```

```
# создание пустого датафрейма для внесения расчетных данных
addiction_total_income_debt = pd.DataFrame()
```

```
# создание столбца с значениями количества заемщиков по соответствующей группе (A, B, C, D, E соответственно)
addiction_total_income_debt['total_income_category_count'] = total_income_category_debt.groupby('total_income_category')['debt'].coun
```

```
# создание столбца с количеством просрочек в каждой группе заемщиков
addiction_total_income_debt['count_debt'] = total_income_category_debt.groupby('total_income_category')['debt'].sum()
```

```
# создание столбца с отношением количества просрочек к количеству людей по соответствующей группе
addiction_total_income_debt['share_debt'] = addiction_total_income_debt['count_debt'] / addiction_total_income_debt['total_income_cat
```

```
# создание столбца соответствующего среднему значению заработка в группе (в группе "A" за среднее значение принято крайнее левое - 10
addiction_total_income_debt['total_income'] = [1000000, 600000, 125000, 40000, 15000]
```

```
# вывод результирующей таблицы проведенных расчетов
addiction_total_income_debt
```

total_income_category	total_income_category_count	count_debt	share_debt	total_income
A	25	2	0.080000	1000000
B	5013	354	0.070616	600000
D	349	21	0.060172	40000

- Вывод:

  Наиболее дисциплинированными в плане возврата кредита являются граждане со средней зарплатой в 40 и 600 тыс. руб. (категория B, D), а менее дисциплинированными с зарплатой 125 и 1000 тыс. руб. (категория A, C). Кроме того, самые не дисциплинированные оказались граждане с минимальной заработной платой на уровне прожиточного минимума (15 тыс. руб. - категория E).

- IV. Исследование как разные цели кредита влияют на его возврат в срок

```
# выделение 2-х столбцов из общего датафрейма для проведения исследования как разные цели кредита влияют на его возврат в срок
purpose_category_debt = data[['purpose_category', 'debt']]
```

```
# создание пустого датафрейма для внесения расчетных данных
addiction_purpose_category_debt = pd.DataFrame()
```

```
# создание столбца с значениями количества заемщиков по соответствующей группе (т.е. цели кредита)
addiction_purpose_category_debt['purpose_category_count'] = purpose_category_debt.groupby('purpose_category')['debt'].count()
```

```
# создание столбца с количеством просрочек в каждой группе заемщиков
addiction_purpose_category_debt['count_debt'] = purpose_category_debt.groupby('purpose_category')['debt'].sum()
```

```
# создание столбца с отношением количества просрочек к количеству людей по соответствующей группе
addiction_purpose_category_debt['share_debt'] = addiction_purpose_category_debt['count_debt'] / addiction_purpose_category_debt['purp
```

```
# вывод результирующей таблицы проведенных расчетов
addiction_purpose_category_debt
```

purpose_category	purpose_category_count	count_debt	share_debt
операции с автомобилем	4279	400	0.093480
операции с недвижимостью	10751	780	0.072551
получение образования	3988	369	0.092528
проведение свадьбы	2313	183	0.079118

```
data_pivot = data.pivot_table(index = ['purpose_category'], values = 'debt', aggfunc = 'mean')
```

```
data_pivot
```

purpose_category	debt
операции с автомобилем	0.093480
операции с недвижимостью	0.072551
получение образования	0.092528
проведение свадьбы	0.079118

▾ Вывод:

Наиболее дисциплинированными в плане возврата кредита являются граждане, взявшие кредит на операции с недвижимостью и проведение свадеб, а вот самые не дисциплинированные оказались граждане взявшие кредит на операции с автомобилем и образование.

Общий вывод:

В результате предобработки:

1. Заполнены пропуски в столбце data['days_employed'], data['total_income'].
2. Устранены аномалии в столбце data[data['children'].
3. Заменен тип данных в столбце data['total_income'] на ‘int’.
4. Устранены дубликаты.

В результате проведенного исследование надежности заемщиков установлено:

1. Взаимосвязь между количеством детей и возвратом кредита в срок существует. Кроме того, необходимо отметить, что в целом бездетные допускают меньшую долю просрочек.
2. Граждане женатые / замужем и находящиеся в разводе - более дисциплинированны в плане возврата кредита, а вот не женатые / не замужем и находящиеся в гражданском браке - менее. Кроме того, стоит подчеркнуть, что самыми дисциплинированными являются овдовевшие.
3. Наиболее дисциплинированными в плане возврата кредита являются граждане со средней зарплатой в 40 и 600 тыс. руб. (категория B, D), а менее дисциплинированными с зарплатой 125 и 1000 тыс. руб. (категория A, C). Кроме того, самые не дисциплинированные оказались граждане с минимальной заработной платой на уровне прожиточного минимума (15 тыс. руб. - категория E).
4. Наиболее дисциплинированными в плане возврата кредита являются граждане, взявшие кредит на операции с недвижимостью и проведение свадеб, а вот самые не дисциплинированные оказались граждане, взявшие кредит на операции с автомобилем и образование.