Исследование надёжности заёмщиков

Заказчик — кредитный отдел банка. Нужно разобраться, влияет ли семейное положение и количество детей клиента на факт погашения кредита в срок. Входные данные от банка — статистика о платёжеспособности клиентов.

Результаты исследования будут учтены при построении модели **кредитного скоринга** — специальной системы, которая оценивает способность потенциального заёмщика вернуть кредит банку.

1 Шаг 1. Откроем файл с данными и изучим общую информацию

In [268]:
```
import pandas as pd
```

In [269]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [270]:
```
try:
    data = pd.read_csv('/da..ta.csv') # Yandex path
except:
    data = pd.read_csv("da...r.csv") # personal path
```

In [272]:
```
data.info() # информация о данных
```

```
<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
```

1.1 12 колонок, 21525 строк

- children - дети. челочисленное. есть данные по всем строкам
- days_employed - трудовой стаж в днях. численное значение. есть пропуски данных
- dob_years - возраст клиента, целочисленное. есть все строки
- education - образование. текст. вроде бы без пропусков. надо посмотреть уники
- education_id - целочисленный идентификатор образования. можно сделать словарь
- family_status - семейное положение. текст.
- family_status_id - целочисленный идентификатор семейного положения. можно сделать словарь

- gender - пол клиента. текст. хорошо бы проверить уники
- income_type - тип занятости. текст. возможно словарь
- debt - имел ли задолженность по возврату. целочисленное. возможно, булевое
- total_income - ежемесячный доход - численное. есть пропуски
- purpose - цель получения кредита - текстовое. возможен словарь

2 Проверим пропуски

In [273]:
```
1 data.isna().sum()
```

Out[273]:
```
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
```

Интересное совпадение - 2174 пропуска в трудовом стаже и в ежемесячном доходе

3 Посмотрим доли отсутствующих значений

In [274]:
```
1 round(data.isna().sum() * 100 / len(data), 2)
```

Out[274]:
```
children            0.00
days_employed      10.10
dob_years           0.00
education           0.00
education_id        0.00
family_status       0.00
family_status_id    0.00
gender              0.00
income_type         0.00
debt                0.00
total_income       10.10
purpose             0.00
dtype: float64
```

Пропуски данных составляют 10%

4 Сводная информация по *параметрам* данных

In [275]:
```
1 data.describe()
```

Out[275]:

	children	days_employed	dob_years	education_id	family_status_id	debt	total_income
count	21,525.00	19,351.00	21,525.00	21,525.00	21,525.00	21,525.00	19,351.00
mean	0.54	63,046.50	43.29	0.82	0.97	0.08	167,422.30
std	1.38	140,827.31	12.57	0.55	1.42	0.27	102,971.57
min	-1.00	-18,388.95	0.00	0.00	0.00	0.00	20,667.26
25%	0.00	-2,747.42	33.00	1.00	0.00	0.00	103,053.15
50%	0.00	-1,203.37	42.00	1.00	0.00	0.00	145,017.94
75%	1.00	-291.10	53.00	1.00	1.00	0.00	203,435.07
max	20.00	401,755.40	75.00	4.00	4.00	1.00	2,265,604.03

Далее - прорабатываем данные по колонкам

5 1. children - количество детей

5.1 проверим на уникальность

In [276]:
```
1 data['children'].unique()
```

Out[276]:
```
array([ 1,  0,  3,  2, -1,  4, 20,  5])
```

5.1.1 С количеством детей не всё в порядке.

1. Отрицательное значение - это описка. надо внести то же значение, но положительное
2. Огромное значение - 20! Тоже описка... Удалить лишний НОЛЬ в записи Делаем:

In [277]:
```
1 # ЗАМЕНА ОТРИЦАТЕЛЬНОГО ЗНАЧЕНИЯ
2 data.loc[data['children'] == -1, 'children'] = 1
3 data['children'].unique()
```

Out[277]:
```
array([ 1,  0,  3,  2,  4, 20,  5])
```

-1 успешно заменил на 1 Теперь черёд за 20

In [278]:
```
1 data.loc[data['children'] == 20, 'children'] = 2
2 data['children'].unique()
```

Out[278]:
```
array([1, 0, 3, 2, 4, 5])
```

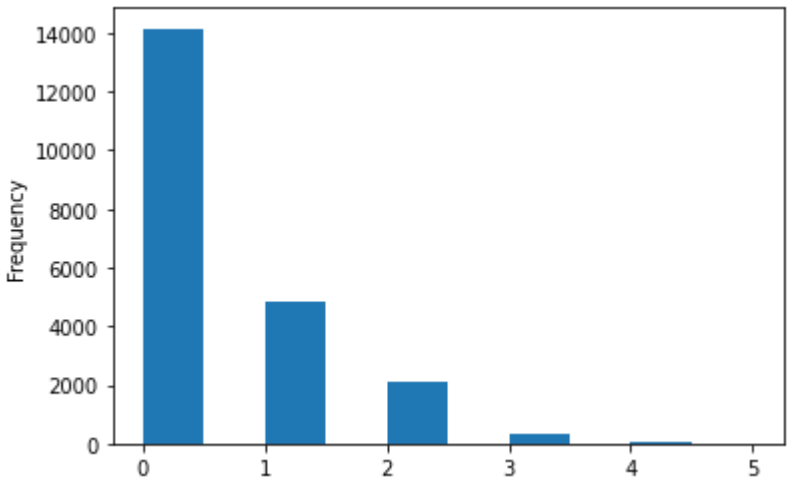
5.1.2 Количество детей - в норме!

Посмотрим распределение количества детей в выборке

In [279]:
```
childrens = data['children']

childrens.plot(kind='hist')
```

Out[279]: <AxesSubplot:ylabel='Frequency'>



5.2 2. days_employed - трудовой стаж.

из проверки пропуском известно, что данные по трудовому стажу содержат ~10% пропусков

5.3 проверим на уникальность

In [280]:
```
# посмотрим список снизу
sorted(data['days_employed'].unique())
```

Out[280]:
```
[-18388.949900568383,
 -17615.563265627912,
 -16593.472817263817,
 -15835.725774811905,
 -15785.678893355005,
 -15773.0613349239,
 -14051.20262056069,
 -13894.357289777596,
 -13025.425448134729,
 -12930.541677797675,
 -12785.542677341233,
 -12587.262898873396,
 -12401.233338542044,
 -12392.30937376918,
 -12136.131380846335,
 -12118.379735167442,
 -12111.680980751775,
 -11991.292307711721,
 -11986.106782911937,
```

In [281]:
```
# посмотрим список сверху
sorted(data['days_employed'].unique(), reverse=True)
```

Out[281]:
```
[401755.40047533,
 401715.8117488882,
 401675.093433862,
 401674.4666333656,
 401663.8500458008,
 401635.0326971183,
 401619.6332980906,
 401614.47562223615,
 401591.8284573659,
 401590.452230711,
 401575.19672763156,
 401573.9052883258,
 401556.7535504825,
 401524.25915292674,
 401517.2763879868,
 401486.70674559905,
 401458.8777808532,
 401446.44671989843,
 401440.83433553757,
```

5.3.1 Выявлено две проблемы

1. отрицательные значения
2. странный формат

Сначала приму **гипотезу** о том, что отрицательные значения - это описка при вводе данных. Почему? Отрицательные значения есть и среди пненсионеров, и среди работающих в настоящее время клиентов. Их надо превратить в положительные значения.

In [282]:
```
data.loc[data['days_employed'] < 0, 'days_employed'] = abs(data['days_employed'
# проверка
sorted(data['days_employed'].unique())
```

Out[282]:
```
[24.14163324048118,
 24.240694791435672,
 37.72660206855514,
 50.12829786804968,
 55.838005521394265,
 60.6373275553343,
 61.5184565937786,
 61.596442746578425,
 72.62578520446468,
 74.99524988670981,
 75.98143820404016,
 79.88034967664467,
 87.9759189019844,
 88.43392772242983,
 94.01204313423708,
 94.10233686775595,
 94.75711243199892,
 95.97095690477032,
 96.65759932125448,
```

5.3.1.1 Отрицательные значения заменены на положительные.

5.3.2 Следующий шаг - это странный формат численного значения.

Минимальное значения в колонке - 24.+ Как известно, во временных отрезках, значение 24 - это количество часов в сутках. Значит, представленные численные данные следует разделить на 24, и получиться количество дней общего трудового стажа.

Результат сохраняю в той же колонке .

In [283]:
```
data['days_employed'] = data['days_employed'] / 24
# проверка
sorted(data['days_employed'].unique())
```

Out[283]:
```
[1.005901385020049,
 1.010028949643153,
 1.571941752856464,
 2.0886790778354034,
 2.3265835633914276,
 2.526555314805596,
 2.5632690247407752,
 2.566518447774101,
 3.0260743835193615,
 3.124802078612909,
 3.1658932585016735,
 3.328347903193528,
 3.6656632875826833,
 3.684746988434576,
 3.917168463926545,
 3.920930702823165,
 3.948213017999955,
 3.9987898710320966,
 4.027399971718936,
```

In [284]:
```
# проверка сверху
sorted(data['days_employed'].unique(), reverse=True)
```

Out[284]:
```
[16739.80835313875,
 16738.158822870344,
 16736.462226410917,
 16736.436109723567,
 16735.993751908365,
 16734.793029046596,
 16734.151387420443,
 16733.93648425984,
 16732.992852390245,
 16732.93550961296,
 16732.299863651315,
 16732.246053680243,
 16731.53139793677,
 16730.17746470528,
 16729.886516166116,
 16728.612781066626,
 16727.453240868883,
 16726.935279995767,
 16726.701430647398,
```

5.3.3 Конверсия в дни из часов удалась.

Но что такое 16739 дней?

In [285]:
```
print('16739 дней это', round(16739/365), "лет" )
```

```
16739 дней это 46 лет
```

46 лет трудового стажа - серьёзная цифра. Перед выходом на пенсию, у меня такой будет. Думаю, стоит проверить, нет ли ошибок здесь.

Если разница между возрастом и стажем меньше, чем 18 лет, стоит внимательно посмотреть на такие строки.

НО перед этим обращаю внимание на то, что в колонке со стажем есть пропуски. Надо с ними поработать и заменить на значение "0". Ведь если нет данных по стажу, значит трудового стажа просто нет. Или есть, но он не указан?

5.3.4 Выведу все строки с пропущенным значением стажа.

In [286]:
```
1 data.sort_values(by='days_employed')
```

Out[286]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	g
17437	1	1.01	31	среднее	1	женат / замужем	0	
8336	0	1.01	32	высшее	0	Не женат / не замужем	4	
6157	2	1.26	47	среднее	1	гражданский брак	1	
9683	0	1.40	43	среднее	1	Не женат / не замужем	4	
2127	1	1.45	31	высшее	0	женат / замужем	0	
...	...	...	...	...	...	...	...	...
21489	2	nan	47	Среднее	1	женат / замужем	0	
21495	1	nan	50	среднее	1	гражданский брак	1	
21497	0	nan	48	ВЫСШЕЕ	0	женат / замужем	0	
21502	1	nan	42	среднее	1	женат / замужем	0	
21510	2	nan	28	среднее	1	женат / замужем	0	

21525 rows × 12 columns

50 лет, среднее образование, женат и **нет стажа?** Не может такого быть. Вывод: данные о стаже в строках со значением Nan говорят о пропусках, а не отстутствии стажа.

Сколько таких строк? 2174 строки. Это 10% от общего числа данных.

Я пока что не имею опыта в банковской сфере и поэтому спрошу бизнес, заполнить ли эти данные "0" или высчитывать медианные значения в корреляции с возрастом (и образованием).

5.3.5 посоветовался с бизнесом

решено пропуски данных о стаже работы оставить **"как есть"**

5.3.6 Проверка реальности больших сроков трудового стажа.

Детство = Возраст - стаж (в годах)

In [287]:
```
data['childhood'] = data['dob_years'] - (data['days_employed'] / 365)
data.sort_values(by='childhood')

```

Out[287]:

	children	days_employed	dob_years	education	education_id	family_status	family_status_id	g
14514	0	16,708.02	0	среднее	1	вдовец / вдова	2	
578	0	16,577.36	0	среднее	1	женат / замужем	0	
16861	0	16,495.58	0	среднее	1	в разводе	3	
14659	0	16,456.22	0	среднее	1	Не женат / не замужем	4	
10188	0	15,486.05	0	среднее	1	женат / замужем	0	
...	...	...	...	...	...	...	...	...
21489	2	nan	47	Среднее	1	женат / замужем	0	
21495	1	nan	50	среднее	1	гражданский брак	1	
21497	0	nan	48	ВЫСШЕЕ	0	женат / замужем	0	
21502	1	nan	42	среднее	1	женат / замужем	0	
21510	2	nan	28	среднее	1	женат / замужем	0	

21525 rows × 13 columns

Вывод отсортированных по "Детству" данных показал, что есть отрицательные значения. Это в тех строках, где не указан возраст. Отберём такие строки в отдельную табличку

In [288]:

```
data_zero_dob_years = data.loc[data.loc[:,'dob_years'] == 0]

data_zero_dob_years.sort_values('childhood')

```

Out[288]:

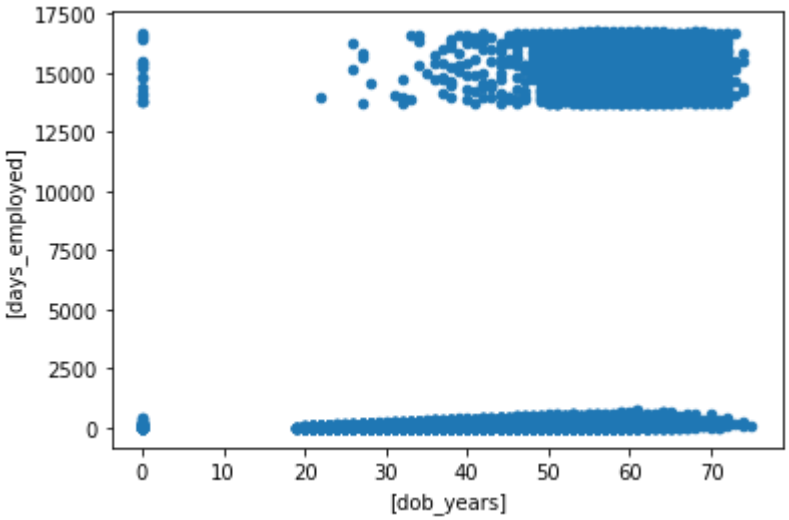
	children	days_employed	dob_years	education	education_id	family_status	family_status_id	g
14514	0	16,708.02	0	среднее	1	вдовец / вдова	2	
578	0	16,577.36	0	среднее	1	женат / замужем	0	
16861	0	16,495.58	0	среднее	1	в разводе	3	
14659	0	16,456.22	0	среднее	1	Не женат / не замужем	4	
10188	0	15,486.05	0	среднее	1	женат / замужем	0	
...	...	...	...	...	...	...	...	...
6670	0	nan	0	Высшее	0	в разводе	3	
8574	0	nan	0	среднее	1	женат / замужем	0	
12403	3	nan	0	среднее	1	женат / замужем	0	
13741	0	nan	0	среднее	1	гражданский брак	1	
19829	0	nan	0	среднее	1	женат / замужем	0	

101 rows × 13 columns

Что здесь получается? Похоже, что в основном это происходит с пенсионерами и тему, у кого нет данных в колонках возвраста и/или стажа

5.3.7 Посмотрим графическое представление по трудовому стажу

In [289]:
```
1 data.plot.scatter(x=['dob_years'], y=['days_employed']);
```



Распределение значений по трудовому стажу от возраста.

гипотеза строки со значениями days_employed меньше 2500 **удалить**

5.4 Несущественно для этого исследования

Ходил в курилку, там повстречал заказчиков исследования. Они и говорят:

- слушай, бро! чё ты паришься со стажем? у тебя какое задание? стаж в нём имеет место быть?
- посмотрел ТЗ " влияет ли семейное положение и количество детей"
  - блин. так ведь действительно, вроде не существенно.
- ладно. в порядок данные привёл, и хорошо. не будут смущать в процессе тзысканий

5.5 3. dob_years - возраст.

In [290]:
```
1 data['dob_years'].describe()
```

Out[290]:
```
count   21,525.00
mean        43.29
std         12.57
min          0.00
25%         33.00
50%         42.00
75%         53.00
max         75.00
Name: dob_years, dtype: float64
```

Из общих данных известно, что:

- возвраст указан в годах
- это целочисленное значение
- среднее 43 года
- есть значения "0" Значения "0" считаем ПРОПУСКОМ данных

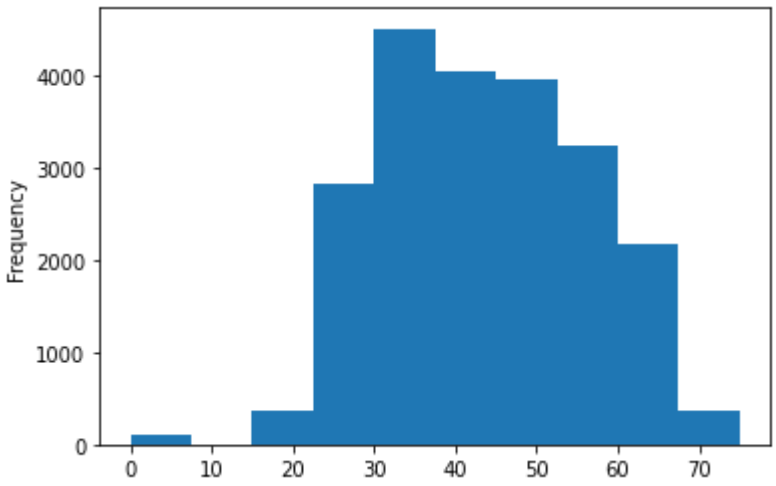
In [291]:
```
1 data[data['dob_years'] == 0].count()
```

Out[291]:
```
children            101
days_employed        91
dob_years           101
education           101
education_id        101
family_status       101
family_status_id    101
gender              101
income_type         101
debt                101
total_income         91
purpose             101
childhood            91
dtype: int64
```

Таких "пропусков" 101 строка, что составляет меньше одного процента от всех данных.

гипотеза Этими строками можно пренебречь, однако! В ТЗ нет прямой отсылки к возрасту. На этом этапе ничего не делаем.

In [292]:
```
1 dob_year = data['dob_years']
2
3 dob_year.plot(kind='hist');
```



Замечательное распределение данных по возрасту. Смело считаем значения меньше 10 лет - пропусками дангных. Принеобходимости можно заменить на NaN

5.6 4. education - образование.

Из общих справочных данных, по колонке Образование, следующие сведения:

- это строковое значение
- нет пропусков строк
- вероятно присутствие неявных дубликатов

Проверяю неявные дубликаты выводом уникальных значений

In [293]:
```
1 data['education'].unique()
```

Out[293]:
```
array(['высшее', 'среднее', 'Среднее', 'СРЕДНЕЕ', 'ВЫСШЕЕ',
       'неоконченное высшее', 'начальное', 'Высшее',
       'НЕОКОНЧЕННОЕ ВЫСШЕЕ', 'Неоконченное высшее', 'НАЧАЛЬНОЕ',
       'Начальное', 'Ученая степень', 'УЧЕНАЯ СТЕПЕНЬ', 'ученая степень'],
      dtype=object)
```

Для удобного восприятия выведу отсортированный список в вертикальном расположении.

In [294]:
```
1 sorted(data['education'].unique())
```

Out[294]:
```
['ВЫСШЕЕ',
 'Высшее',
 'НАЧАЛЬНОЕ',
 'НЕОКОНЧЕННОЕ ВЫСШЕЕ',
 'Начальное',
 'Неоконченное высшее',
 'СРЕДНЕЕ',
 'Среднее',
 'УЧЕНАЯ СТЕПЕНЬ',
 'Ученая степень',
 'высшее',
 'начальное',
 'неоконченное высшее',
 'среднее',
 'ученая степень']
```

Здесь видны неявные дубликаты

- высшее - ВЫСШЕЕ, Высшее
- начальное - НАЧАЛЬНОЕ, Начальное
- неоконченное высшее - НЕОКОНЧЕННОЕ ВЫСШЕЕ, Неоконченное высшее
- среднее - СРЕДНЕЕ, Среднее
- ученая степень - УЧЕНАЯ СТЕПЕНЬ, Ученая степень

Эти дубликаты поменяю на аналоги, написанные в нижнем регистре Попробую применить метод перевода в нижний регистр всех значений

In [295]:
```
1 data['education'] = data['education'].str.lower()
2 sorted(data['education'].unique())
```

Out[295]:
```
['высшее', 'начальное', 'неоконченное высшее', 'среднее', 'ученая степень']
```

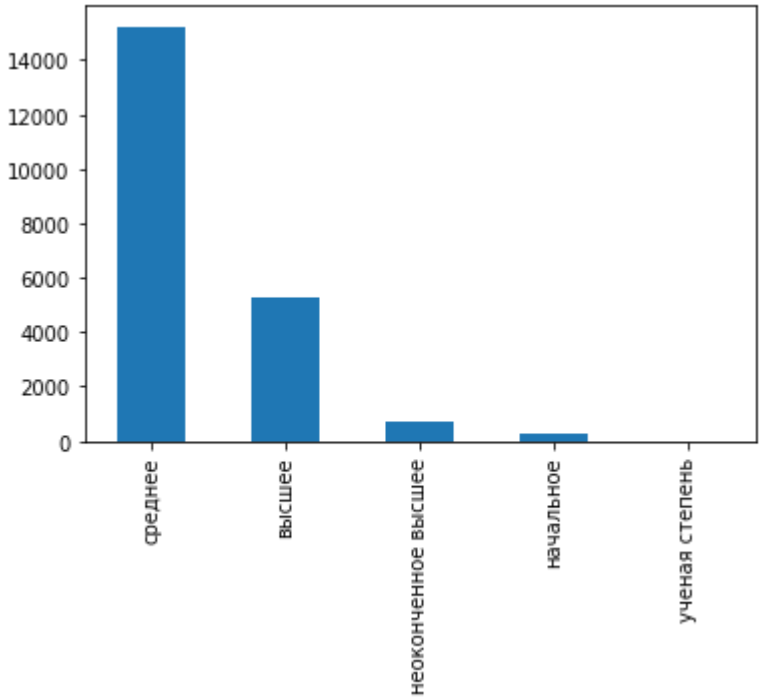
5.6.1 Успех!

Посмотрим распределение уровня образования среди клиентов

In [296]:
```
1 # data.plot(x='education', y='children',  kind='bar')
2 # подсчитываю количество строковых значений
3 unique_edu = sorted(data['education'].unique())
4 unique_edu
```

Out[296]:
```
['высшее', 'начальное', 'неоконченное высшее', 'среднее', 'ученая степень']
```

In [297]:
```
1 data['education'].value_counts().plot(kind='bar');
```



In [298]:
```
1 data['education'].sort_values().value_counts()
```

Out[298]:
```
среднее                15233
высшее                  5260
неоконченное высшее      744
начальное                282
ученая степень             6
Name: education, dtype: int64
```

Эта информация в ТЗ не требуется. Но интересно же!

5.7 5. education_id - идентификатор уровня образования.

Задачи: Проверить, сколько уникальных значений Как они соотносятся со строковыми значениями уровня образования В случае ошибок - исправить

In [299]:
```
1 sorted(data['education_id'].unique())
```

Out[299]:
```
[0, 1, 2, 3, 4]
```

Прекрасно! Количество значений _id совпадает с количеством категорий в строковым формате Теперь хочу проверить соответствие строковых и численных категорий. Нет ли ошибок...

In [300]:
```
1 id_0 = data.loc[data.loc[:, 'education_id'] == 0]
2 id_0['education'].sort_values().value_counts()
```

Out[300]:
```
высшее    5260
Name: education, dtype: int64
```

In [301]:
```
id_0 = data.loc[data.loc[:, 'education_id'] == 1]
id_0['education'].sort_values().value_counts()
```

Out[301]:
```
среднее    15233
Name: education, dtype: int64
```

In [302]:
```
id_0 = data.loc[data.loc[:, 'education_id'] == 2]
id_0['education'].sort_values().value_counts()
```

Out[302]:
```
неоконченное высшее    744
Name: education, dtype: int64
```

In [303]:
```
id_0 = data.loc[data.loc[:, 'education_id'] == 3]
id_0['education'].sort_values().value_counts()
```

Out[303]:
```
начальное    282
Name: education, dtype: int64
```

5.7.1 Прекрасно!

Каждая числовая категория _id соответствует одному значению строковой категории

5.8 Можно сделать словарь категория уровня образования

Из основной таблицы data убираем колонку education И Создаём табличку 'education_dict' с колонками education_id и education

In [304]:
```
# новая табличка
education_dict = data[['education_id', 'education']]
education_dict
```

Out[304]:

	education_id	education
0	0	высшее
1	1	среднее
2	1	среднее
3	1	среднее
4	1	среднее
...	...	...
21520	1	среднее
21521	1	среднее
21522	1	среднее
21523	1	среднее
21524	1	среднее

21525 rows × 2 columns

In [305]:
```
# убираем дубликаты в словаре
education_dict = education_dict.drop_duplicates().reset_index(drop=True)
education_dict
```

Out[305]:

	education_id	education
0	0	высшее
1	1	среднее
2	2	неоконченное высшее
3	3	начальное
4	4	ученая степень

Словарь education_dict готов!

Убираю лишнюю колонку из data

In [306]:
```
del data['education']
data
```

Out[306]:

	children	days_employed	dob_years	education_id	family_status	family_status_id	gender	inco
0	1	351.57	42	0	женат / замужем	0	F	с
1	1	167.70	36	1	женат / замужем	0	F	с
2	0	234.31	33	1	женат / замужем	0	M	с
3	3	171.86	32	1	женат / замужем	0	M	с
4	0	14,177.75	53	1	гражданский брак	1	F	п
...	...	...	...	...	...	...	...	
21520	1	188.72	43	1	гражданский брак	1	F	к
21521	0	14,330.73	67	1	женат / замужем	0	F	п
21522	1	88.06	38	1	гражданский брак	1	M	с
21523	3	129.69	38	1	женат / замужем	0	M	с
21524	2	82.69	40	1	женат / замужем	0	F	с

21525 rows × 12 columns

5.9 6. family_status - семейное положение.

5.9.1 Из общей справки по данным

- строковое значение

- пропусков нет

5.9.2 Получаю список уникальных значений

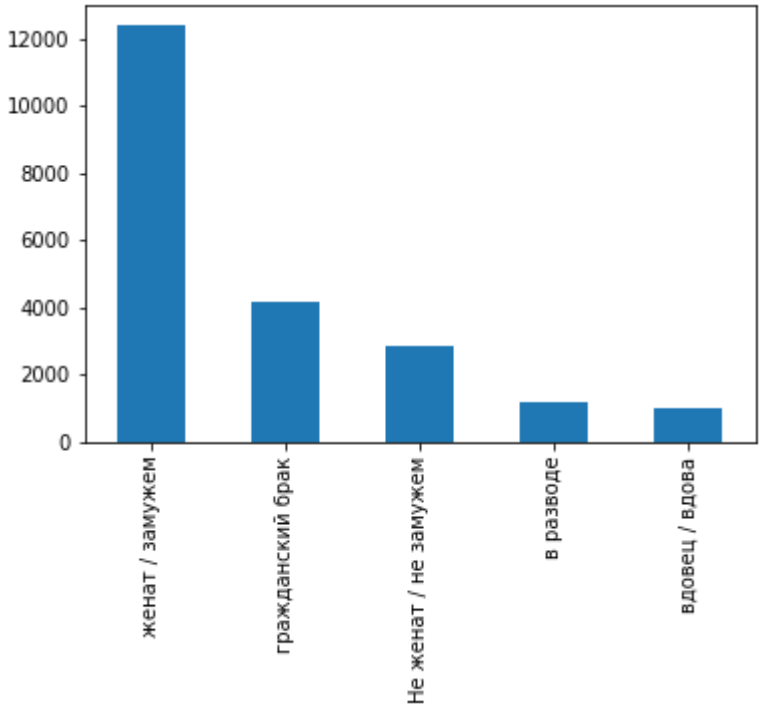
In [307]:
```
1 data['family_status'].unique()
```

Out[307]:
```
array(['женат / замужем', 'гражданский брак', 'вдовец / вдова',
       'в разводе', 'Не женат / не замужем'], dtype=object)
```

5.9.3 Прекрасные новости!

Всего встречается 5 значений Неявных дубликатов нет

In [308]:
```
1 data['family_status'].value_counts().plot(kind='bar');
```



5.9.4 Можно переходить к следущей колонке

5.10 7. family_status_id - идентификатор семейного положения.

5.10.1 Из общей справки по данным

- целочисленный идентификатор семейного положения. можно сделать словарь
- пропусков нет

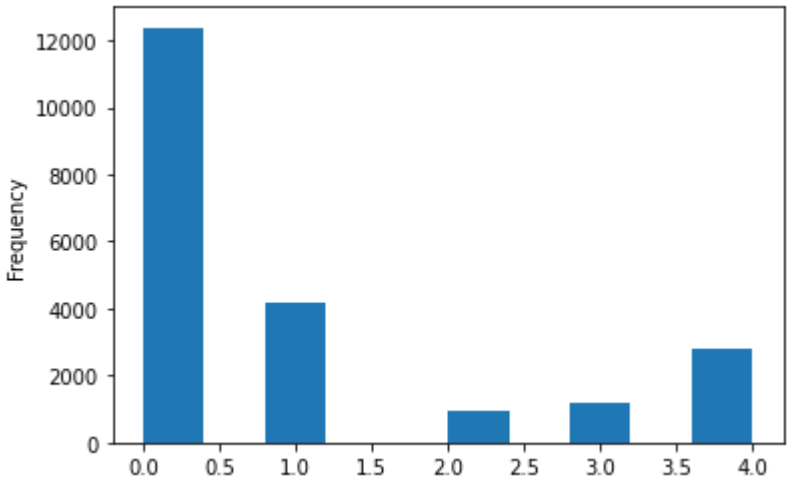
5.10.2 Посмотрим данные в отсортированном виде

In [309]:
```
1 data['family_status_id'].sort_values().value_counts()
```

Out[309]:
```
0    12380
1     4177
4     2813
3     1195
2      960
Name: family_status_id, dtype: int64
```

5.10.3 Распределение значений

In [310]:
```
1 family_status_id = data['family_status_id']
2 family_status_id.plot(kind='hist');
```



5.11 Для удобства оперирования данными, можно вывести колонку family_status в отдельный словарь

Из основной таблицы data убираем колонку family_status И Создаём табличку 'family_status_dict' с колонками family_status_id и family_status

In [311]:
```
1 # новая табличка
2 family_status_dict = data[['family_status_id', 'family_status']]
3 family_status_dict
```

Out[311]:

	family_status_id	family_status
0	0	женат / замужем
1	0	женат / замужем
2	0	женат / замужем
3	0	женат / замужем
4	1	гражданский брак
...	...	...
21520	1	гражданский брак
21521	0	женат / замужем
21522	1	гражданский брак
21523	0	женат / замужем
21524	0	женат / замужем

21525 rows × 2 columns

In [312]:
```
# убираем дубликаты в словаре
family_status_dict = family_status_dict.drop_duplicates().reset_index(drop=True
family_status_dict
```

Out[312]:

	family_status_id	family_status
0	0	женат / замужем
1	1	гражданский брак
2	2	вдовец / вдова
3	3	в разводе
4	4	Не женат / не замужем

Словарь family_status_dict готов!

Убираю лишнюю колонку из data

In [313]:
```
del data['family_status']
data
```

Out[313]:

	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	debt
0	1	351.57	42	0	0	F	сотрудник	0
1	1	167.70	36	1	0	F	сотрудник	0
2	0	234.31	33	1	0	M	сотрудник	0
3	3	171.86	32	1	0	M	сотрудник	0
4	0	14,177.75	53	1	1	F	пенсионер	0
...	...	...	...	...	...	...	...	...
21520	1	188.72	43	1	1	F	компаньон	0
21521	0	14,330.73	67	1	0	F	пенсионер	0
21522	1	88.06	38	1	1	M	сотрудник	1
21523	3	129.69	38	1	0	M	сотрудник	1
21524	2	82.69	40	1	0	F	сотрудник	0

21525 rows × 11 columns

5.12 8. gender - пол

5.12.1 Из общей справки по данным

- текстовый формат
- пропусков нет

5.12.2 Смотрим уникальные значения

In [314]:
```
1 data['gender'].sort_values().value_counts()
```

Out[314]:
```
F      14236
M       7288
XNA        1
Name: gender, dtype: int64
```

Женщины, женщины... Всё то они крутятся, вертятся... И ещё странная XNA Пойду, проверю ЙЦУКЕН

- проверка на ЙЦУКЕН не дала вменяемого результата Посмотрю всю строку с этим значением

In [315]:
```
1 print(data.loc[data.loc[:,'gender'] == 'XNA'])
```

```
       children  days_employed  dob_years  education_id  family_status_id  \
10701         0          98.28         24             2                 1

      gender income_type  debt  total_income               purpose  childhood
10701    XNA   компаньон     0    203,905.16  покупка недвижимости      23.73
```

Я предложил заказчику **удалить** эту единственную странную строку. Заказчик согласился.

In [316]:
```
1 data = data.loc[data['gender'] != 'XNA']
2
3 #data.drop(data[data['gender'] == 'XNA'].index, inplace=True)
```

In [317]:
```
1 # контроль
2 data.reset_index(inplace=True)
3 data['gender'].sort_values().value_counts()
```

Out[317]:
```
F    14236
M     7288
Name: gender, dtype: int64
```

In [318]:
```
1 data[10700:10703]
```

Out[318]:

	index	children	days_employed	dob_years	education_id	family_status_id	gender	income_typ
10700	10700	1	21.37	37	0	0	F	компаньо
10701	10702	0	16,191.93	60	1	1	F	пенсионе
10702	10703	0	150.83	35	1	0	F	сотрудни

In [ ]:
```
1
```

5.13 Успешно удалена странная строка из dataframe

5.14 9. income_type - тип занятости

5.14.1 Из общей справки по данным

- текстовый формат

- пропусков нет

5.14.2 Смотрим уникальные значения

In [319]:
```
1 data['income_type'].sort_values().value_counts()
```

Out[319]:
```
сотрудник          11119
компаньон           5084
пенсионер           3856
госслужащий         1459
безработный            2
предприниматель        2
студент                1
в декрете              1
Name: income_type, dtype: int64
```

5.14.2.1 у нас получается "карманный банк" для своих, для сотрудников и компаньонов ?

нет. наверно всё-таки здесь ошибка в терминах

5.14.3 Прекрасная колонка! Ничего не надо делать)))

5.15 10. debt - имел ли задолженность по возврату.

5.15.1 Из общей справки по данным

- текстовое
- пропусков нет

5.15.2 Смотрим уникальные значения

In [320]:
```
1 data['debt'].sort_values().value_counts()
```

Out[320]:
```
0    19783
1     1741
Name: debt, dtype: int64
```

5.15.3 Вот, какие замечательные клиенты у банка!

Задолженности имели лишь...

In [321]:
```
1 print((1741/19783)*100, '%')
```

```
8.800485265126625 %
```

Много это или мало? Спрошу у бизнеса...

5.16 11. total_income - ежемесячный доход - численное

5.16.1 Из общей справки по данным

- вещественное
- есть пропуски. столько же ,сколько пропуков в колнке со стажем
- отрицательные значения отсутствуют

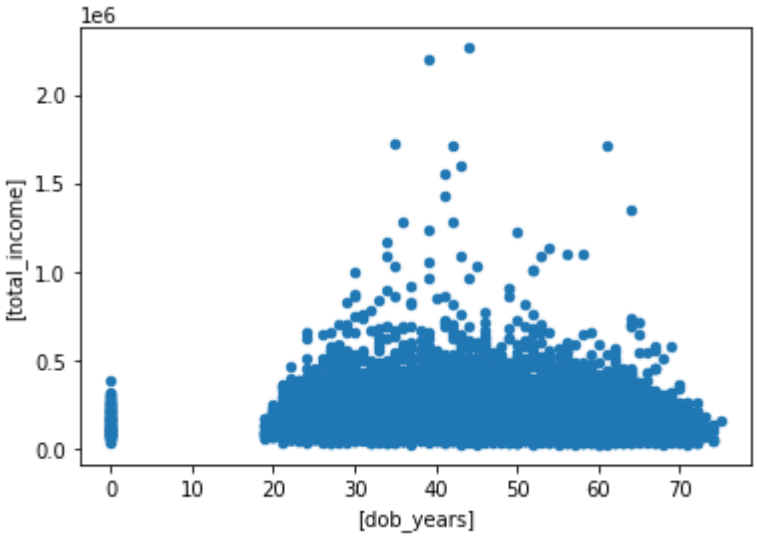
5.16.2 Смотрим в сортированном виде

In [322]:
```
1 data['total_income'].sort_values()
```

Out[322]:
```
14584   20,667.26
13005   21,205.28
16173   21,367.65
1598    21,695.10
14275   21,895.61
           ...
21488         nan
21494         nan
21496         nan
21501         nan
21509         nan
Name: total_income, Length: 21524, dtype: float64
```

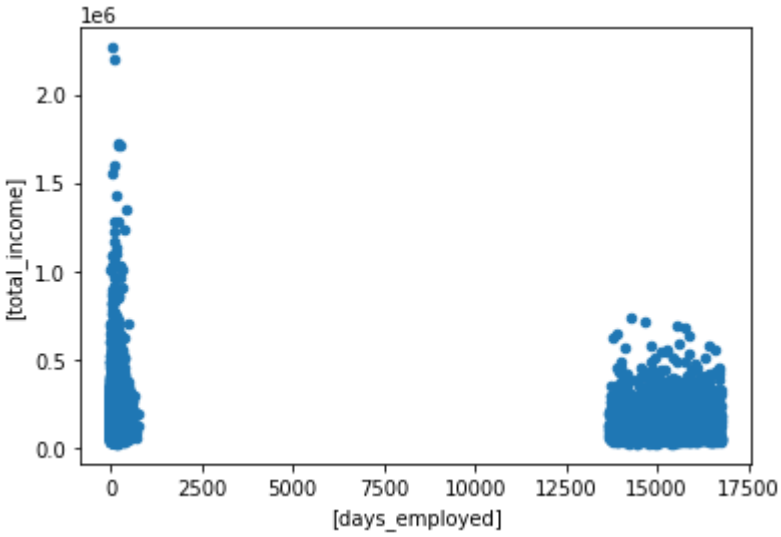
5.16.3 Распределение возраст/доход

In [323]:
```
1 data.plot.scatter(x=['dob_years'], y=['total_income']);
```



5.16.4 Распределение стаж / доход

In [324]:
```
1 data.plot.scatter(x=['days_employed'], y=['total_income']);
```

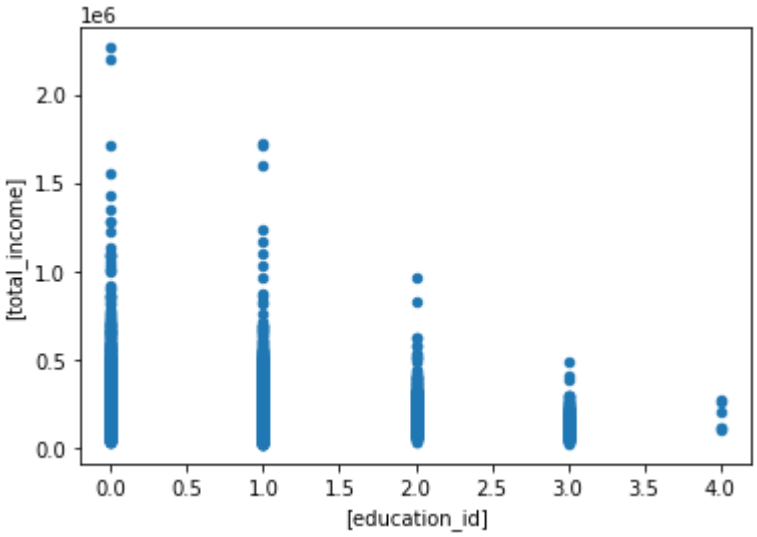


Есть клиенты без указания стажа с высоким уровнем дохода. Интересно посмотреть их тип занятости...

5.16.5 Соотношение образования и уровня дохода

- высшее
- среднее
- неоконченное высшее
- начальное
- ученая степень

In [325]:
```
1 data.plot.scatter(x=['education_id'], y=['total_income']);
```

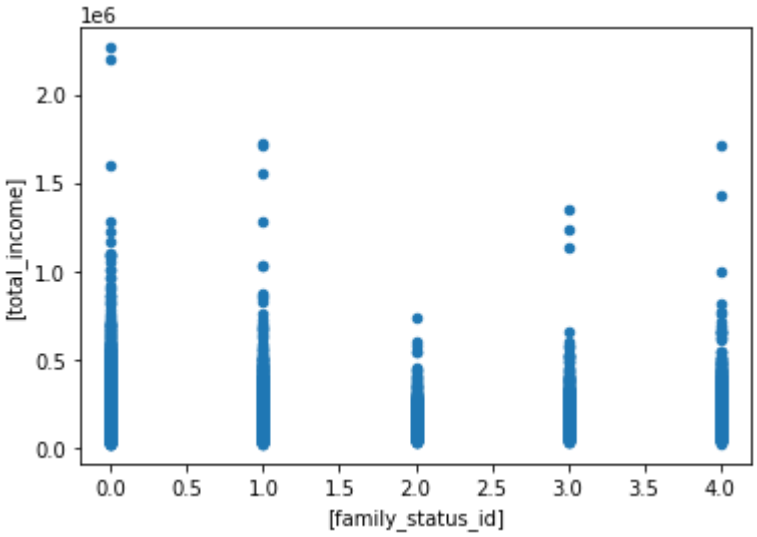


Парадокс - обладатели учёной степени заявляют самый низкий уровень дохода!

5.16.6 Соотношение семейного статуса и уровня дохода

- женат / замужем
- гражданский брак
- вдовец / вдова
- в разводе
- Не женат / не замужем

In [326]:
```
1 data.plot.scatter(x=['family_status_id'], y=['total_income']);
```

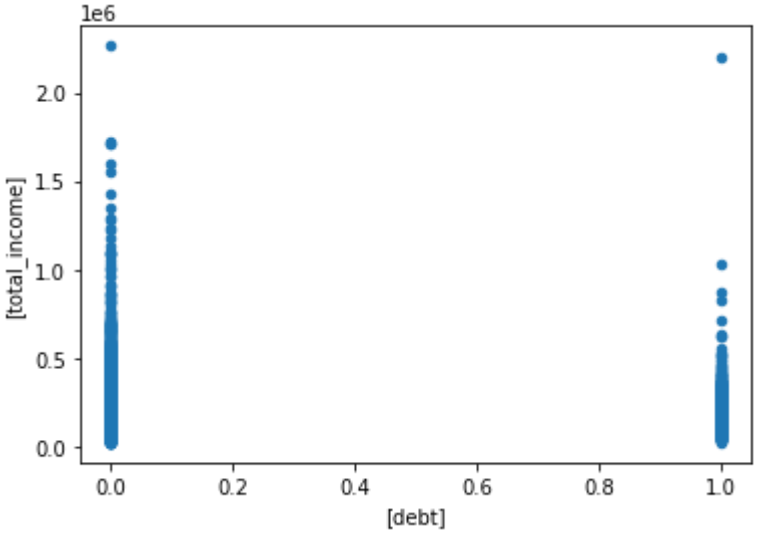


Интересная и совсем не однозначная картина.

5.16.7 Соотношение наличие задолженности и уровень дохода

- нет задолженности
- есть задолженность

In [327]:
```
1 data.plot.scatter(x=['debt'], y=['total_income']);
```



С более высоким уровнем дохода - меньше задолженностей.

5.17 Начну восполнять пробелы в данных о доходах

На уровень дохода могут влиять * стаж * возраст * образование * пол * тип занятости

В данных о стаже у неас пропуски в тех же строках, что и данных о доходе. Так что этот параметр в расчет точно не берём.

Пол. Ну... Во-первых, у нас среди клиентов преобладание женщин. Во-вторых, в нашей стране уже значительно снизилась зависисимость заработка от пола "в среднем по больнице". Так что этот параметр тоже пока отложим.

В сухом остатке - возраст - образование - тип занятости

Данные по образованию и типу занятости являются прямыми категориями. Но возраст - это набор цифр. Для анализа медианного значения дохода следует этот показатель свести к категориям.

По исследованию данных в первой части работы:

- медиана = 43 года
- минимум ~ 18 лет

Исследуем пропуски (згачение "0") в возрасте

In [328]:
```
# табличка с возрастом "0"
dob_years_0 = data[data['dob_years'] == 0]
dob_years_0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101 entries, 99 to 21312
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   index             101 non-null    int64  
 1   children          101 non-null    int64  
 2   days_employed     91 non-null     float64
 3   dob_years         101 non-null    int64  
 4   education_id      101 non-null    int64  
 5   family_status_id  101 non-null    int64  
 6   gender            101 non-null    object 
 7   income_type       101 non-null    object 
 8   debt              101 non-null    int64  
 9   total_income      91 non-null     float64
 10  purpose           101 non-null    object 
 11  childhood         91 non-null     float64
dtypes: float64(3), int64(6), object(3)
memory usage: 10.3+ KB
```

In [329]:
```
dob_years_0.describe()
```

Out[329]:

	index	children	days_employed	dob_years	education_id	family_status_id	debt	total_inc
count	101.00	101.00	91.00	101.00	101.00	101.00	101.00	9
mean	10,300.77	0.50	2,896.50	0.00	0.67	1.24	0.08	158,33
std	5,826.99	0.81	5,900.53	0.00	0.51	1.52	0.27	74,38
min	99.00	0.00	4.54	0.00	0.00	0.00	0.00	34,97
25%	6,407.00	0.00	40.29	0.00	0.00	0.00	0.00	99,58
50%	10,545.00	0.00	73.29	0.00	1.00	1.00	0.00	152,41
75%	14,608.00	1.00	209.25	0.00	1.00	3.00	0.00	212,54
max	21,313.00	3.00	16,708.02	0.00	2.00	4.00	1.00	386,37

Что ж... всё очень похоже на случайное значение "0" в данных по возрасту.

Заполню медианными значениями по группам

Сначала заменю "0" на np.NaN

In [330]:
```
import numpy as np
data['dob_years'].replace(0, np.NaN, inplace = True)
```

In [331]:
```
# проверка
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21524 entries, 0 to 21523
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   index             21524 non-null  int64  
 1   children          21524 non-null  int64  
 2   days_employed     19350 non-null  float64
 3   dob_years         21423 non-null  float64
 4   education_id      21524 non-null  int64  
 5   family_status_id  21524 non-null  int64  
 6   gender            21524 non-null  object 
 7   income_type       21524 non-null  object 
 8   debt              21524 non-null  int64  
 9   total_income      19350 non-null  float64
 10  purpose           21524 non-null  object 
 11  childhood         19350 non-null  float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.0+ MB
```

In [332]:
```
data.sort_values(['dob_years'])
```

Out[332]:

	index	children	days_employed	dob_years	education_id	family_status_id	gender	income_typ
1981	1981	0	30.19	19.00	1	4	F	компань
15658	15659	0	39.51	19.00	2	1	F	компань
13020	13021	0	29.00	19.00	2	4	F	компань
10235	10235	0	33.06	19.00	1	0	F	сотрудни
9218	9218	0	13.42	19.00	1	1	F	компань
...	...	...	...	...	...	...	...	
19828	19829	0	nan	nan	1	0	F	сотрудни
20461	20462	0	14,113.95	nan	1	0	F	пенсионе
20576	20577	0	13,822.55	nan	1	4	F	пенсионе
21178	21179	2	4.54	nan	0	0	M	компань
21312	21313	0	52.85	nan	1	4	M	сотрудни

21524 rows × 12 columns

И меняем NaN на медианные значения по группам

In [333]:
```
data['dob_years'] = data.loc[:, 'dob_years'].fillna(data.groupby('family_status
```

In [334]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21524 entries, 0 to 21523
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   index             21524 non-null  int64  
 1   children          21524 non-null  int64  
 2   days_employed     19350 non-null  float64
 3   dob_years         21524 non-null  float64
 4   education_id      21524 non-null  int64  
 5   family_status_id  21524 non-null  int64  
 6   gender            21524 non-null  object 
 7   income_type       21524 non-null  object 
 8   debt              21524 non-null  int64  
 9   total_income      19350 non-null  float64
 10  purpose           21524 non-null  object 
 11  childhood         19350 non-null  float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.0+ MB
```

In [335]:
```
1 data.sort_values(['dob_years'])
```

Out[335]:

	index	children	days_employed	dob_years	education_id	family_status_id	gender	income_typ
5563	5563	0	36.89	19.00	2	4	M	сотрудни
4098	4098	0	4.66	19.00	1	1	M	компань
12046	12047	0	42.51	19.00	1	4	M	сотрудни
2725	2725	0	4.83	19.00	1	4	F	компань
10235	10235	0	33.06	19.00	1	0	F	сотрудни
...	...	...	...	...	...	...	...	
11531	11532	0	278.45	74.00	1	1	F	сотрудни
2557	2557	0	15,535.88	74.00	1	0	F	пенсионе
3460	3460	0	14,359.33	74.00	1	0	M	пенсионе
4895	4895	0	14,230.34	74.00	0	0	F	пенсионе
8880	8880	0	69.96	75.00	1	2	F	госслужащи

21524 rows × 12 columns

5.18 Заполняем пропуски дохода

In [336]:
```
1 data['total_income'] = data['total_income'].fillna(data.groupby(['income_type',
```

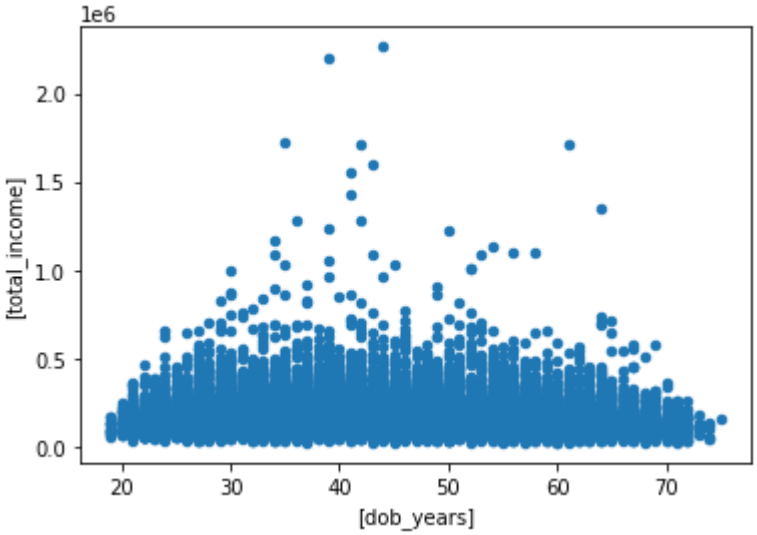
Спасибо. Учту.

5.18.1 Вот что получилось

5.18.2 Распределение возраст/доход

Стало

In [337]:
```
1 data.plot.scatter(x=['dob_years'], y=['total_income']);
```



5.19 12. purpose - цель получения кредита

5.19.1 Из общей справки по данным

- текстовое
- пропусков нет

5.19.2 Смотрим уникальные значения

In [338]:
```
1 data['purpose'].sort_values().value_counts()
```

Out[338]:
```
свадьба                                    797
на проведение свадьбы                      777
сыграть свадьбу                            774
операции с недвижимостью                   676
покупка коммерческой недвижимости          664
покупка жилья для сдачи                    653
операции с жильем                          653
операции с коммерческой недвижимостью      651
жилье                                      647
покупка жилья                              647
покупка жилья для семьи                    641
строительство собственной недвижимости     635
недвижимость                               634
операции со своей недвижимостью            630
строительство жилой недвижимости           626
покупка недвижимости                       623
строительство недвижимости                 620
покупка своего жилья                       620
ремонт жилью                               612
покупка жилой недвижимости                 607
на покупку своего автомобиля               505
заняться высшим образованием               496
автомобиль                                 495
сделка с подержанным автомобилем           489
свой автомобиль                            480
на покупку подержанного автомобиля         479
автомобили                                 478
на покупку автомобиля                      472
дополнительное образование                 462
приобретение автомобиля                    462
сделка с автомобилем                       455
высшее образование                         453
получение дополнительного образования      447
образование                                447
получение образования                      443
профильное образование                     436
получение высшего образования              426
заняться образованием                      412
Name: purpose, dtype: int64
```

Видим множество неявных дубликатов

5.19.3 Создам список уникальных значений

In [339]:
```
1 purpose_unique = data['purpose'].unique()
2 purpose_unique
```

Out[339]:
```
array(['покупка жилья', 'приобретение автомобиля',
       'дополнительное образование', 'сыграть свадьбу',
       'операции с жильем', 'образование', 'на проведение свадьбы',
       'покупка жилья для семьи', 'покупка недвижимости',
       'покупка коммерческой недвижимости', 'покупка жилой недвижимости',
       'строительство собственной недвижимости', 'недвижимость',
       'строительство недвижимости', 'на покупку подержанного автомобиля',
       'на покупку своего автомобиля',
       'операции с коммерческой недвижимостью',
       'строительство жилой недвижимости', 'жилье',
       'операции со своей недвижимостью', 'автомобили',
       'заняться образованием', 'сделка с подержанным автомобилем',
       'получение образования', 'автомобиль', 'свадьба',
       'получение дополнительного образования', 'покупка своего жилья',
       'операции с недвижимостью', 'получение высшего образования',
       'свой автомобиль', 'сделка с автомобилем',
       'профильное образование', 'высшее образование',
       'покупка жилья для сдачи', 'на покупку автомобиля', 'ремонт жилью',
       'заняться высшим образованием'], dtype=object)
```

Предварительное знакомство с данными показало то, что кое-что в них было не так.

6 children

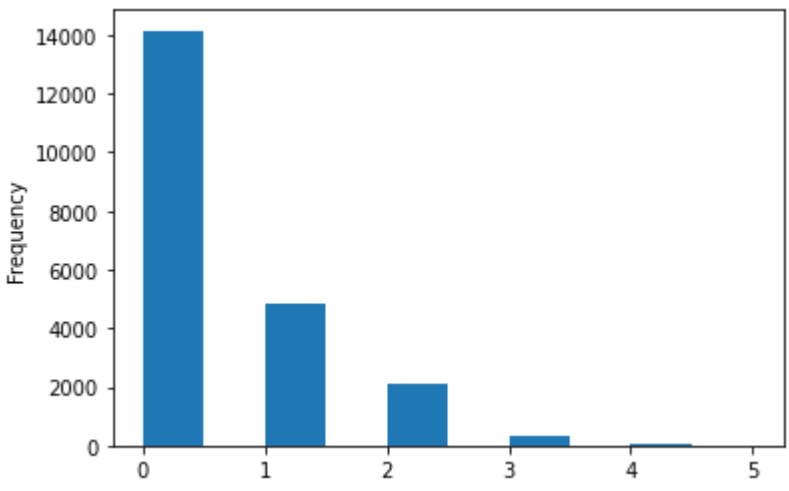
6.1 отрицательные значения

исправлено

6.2 гигантское количество

убран "0" в числе

In [340]:
```
1 childrens.plot(kind='hist');
```



Прав Владимир Владимирович. Мало детей у нас...

7 days_employed

7.1 странные числа

методом сортировки вывлено минимальное количество, очень похожее на количество часов в сутках значения заменены делением на 24

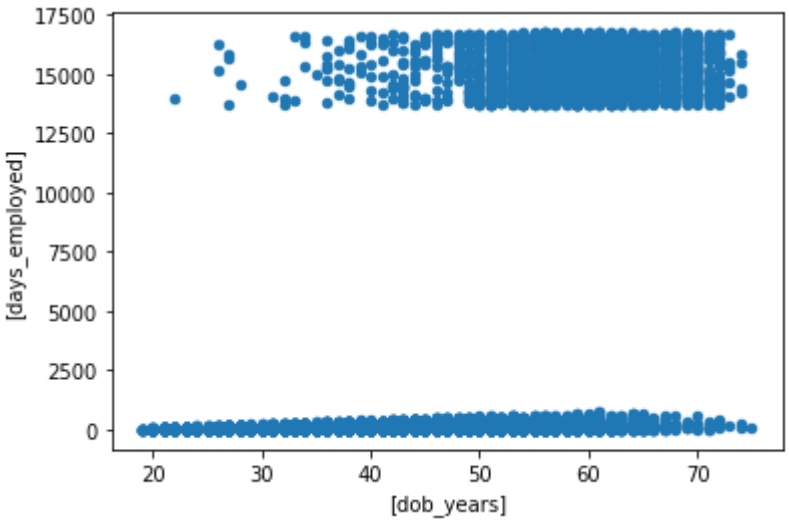
7.2 отрицательные значения

отрицательные значения заменены на положительные

7.3 пропущенные значения

пока оставлено без изменений

In [341]:
```
1 data.plot.scatter(x=['dob_years'], y=['days_employed']);
```



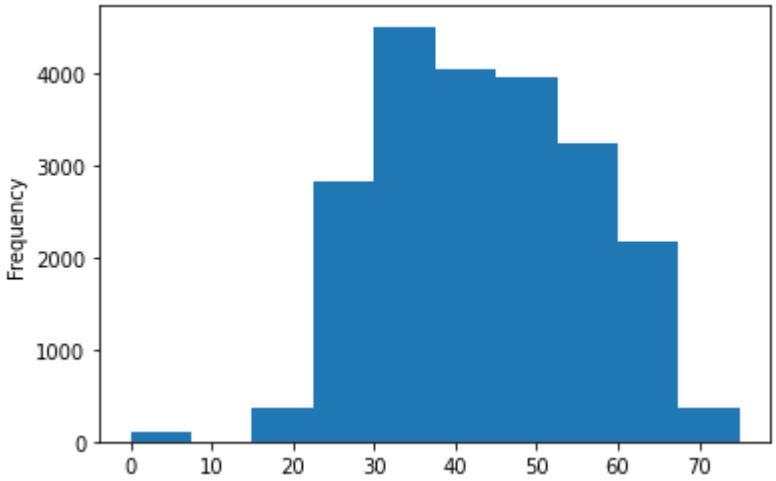
Безвозрастные умудрились наработать трудовой стаж. Но мы уже избавились от безвозрастных - наградили их жизнью

8 dob_years

8.1 пропущенные значения

пока оставлено без изменений

In [342]:
```
1 dob_year.plot(kind='hist');
```



Безвозрастных мало, а основная возрастная группа - 30 - 50 лет.

9 education

только странно, что клиенты с учёной степенью имеют малый доход

9.1 неявные дубликаты

из-за применения разного регистра исправлено

9.2 по сути, дублируется колонкой education_id

столбец удалён сформирован словарь с education_id

10 education_id

всё хорошо сформирован словарь для education

11 family_status и family_status_id

сформирован словарь излишний столбец family_status удалён

12 gender

почти всё хорошо

12.1 Успешно удалена странная строка из dataframe

In [343]:
```
1 data['gender'].sort_values().value_counts()
```

Out[343]:
```
F    14236
M     7288
Name: gender, dtype: int64
```

13 income_type

всё прекрасно

In [344]:
```
1 data['income_type'].sort_values().value_counts()
```

Out[344]:
```
сотрудник          11119
компаньон           5084
пенсионер           3856
госслужащий         1459
безработный            2
предприниматель        2
студент                1
в декрете              1
Name: income_type, dtype: int64
```

14 debt

данные в отличном состоянии

In [345]:
```
1 data['debt'].sort_values().value_counts()
```

Out[345]:
```
0    19783
1     1741
Name: debt, dtype: int64
```

Вот, какие замечательные клиенты у банка!

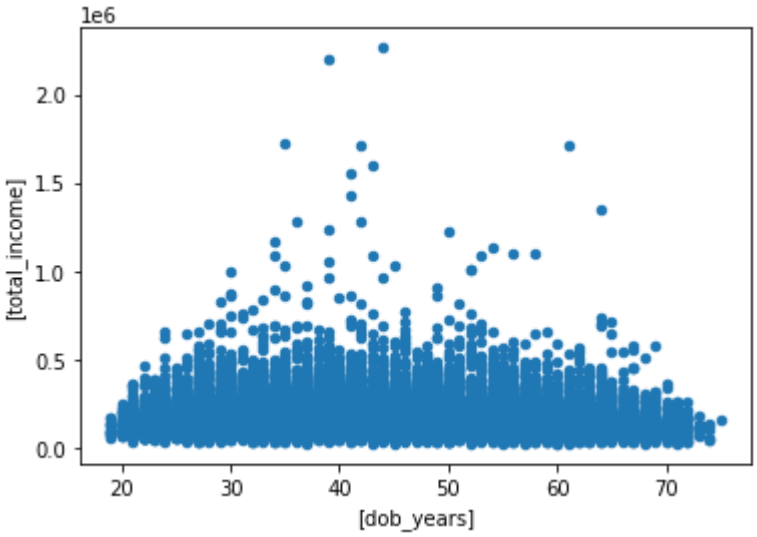
15 total_income

15.1 есть незначительное (~10%) количество пропусков

пока что пропуски оставил как есть

Распределение возраст/доход

In [346]:
```
1 data.plot.scatter(x=['dob_years'], y=['total_income']);
```



16 purpose

нет пропусков есть неявные дубликаты

17 Шаг 2. Предобработка данных

18 Обработка пропусков

Для обработки пропусков, обратим внимание на вопросы, поставленные в качестве целей исследования.

Это

- Есть ли зависимость между наличием детей и возвратом кредита в срок?
- Есть ли зависимость между семейным положением и возвратом кредита в срок?
- Есть ли зависимость между уровнем дохода и возвратом кредита в срок?
- Как разные цели кредита влияют на его возврат в срок?

Прорабатываем пропуски последовательно.

18.1 ...детей ...

18.1.1 есть ли пропуски в детях?

из первого раздела известно, что данных о количестве детей, пропусков нет. а со странными значениями уже справились

In [347]:
```
1 data['children'].value_counts()
```

Out[347]:
```
0    14148
1     4865
2     2131
3      330
4       41
5        9
Name: children, dtype: int64
```

18.2 ...семейное положение ...

18.2.1 есть ли пропуски в данных о семейном положении?

из первого раздела известно, что таких пропусков нет. а со странными значениями уже справились

In [348]:
```
1 data['family_status_id'].value_counts()
```

Out[348]:
```
0    12380
1     4176
4     2813
3     1195
2      960
Name: family_status_id, dtype: int64
```

18.3 ...уровень дохода ...

18.3.1 есть ли пропуски в данных об уровне дохода?

из первого раздела известно, что такие пропуски есть. посмотрим на них более пристально

In [349]:
```
1 data['total_income'].isna().sum()
```

Out[349]:
```
2
```

2 строк пропуском значений

ДВЕ строки можно удалить без последствий для результатов анализа

In [350]:
```
1 income_nan = data.loc[data['total_income'].isna()]
2 income_nan
```

Out[350]:

	index	children	days_employed	dob_years	education_id	family_status_id	gender	incom
5936	5936	0	nan	58.00	0	0	M	предприним
12406	12407	0	nan	57.00	2	3	F	пенс

Из принта видно, что Nan одновременно встречается в колонках days_employed и total_income (и вычесленном childhood) всего 2 строки

Для предпринимателя и пенсионера не нашлось похожих)))

In [351]:
```
print(income_nan['days_employed'].isna().sort_values())
```

```
5936     True
12406    True
Name: days_employed, dtype: bool
```

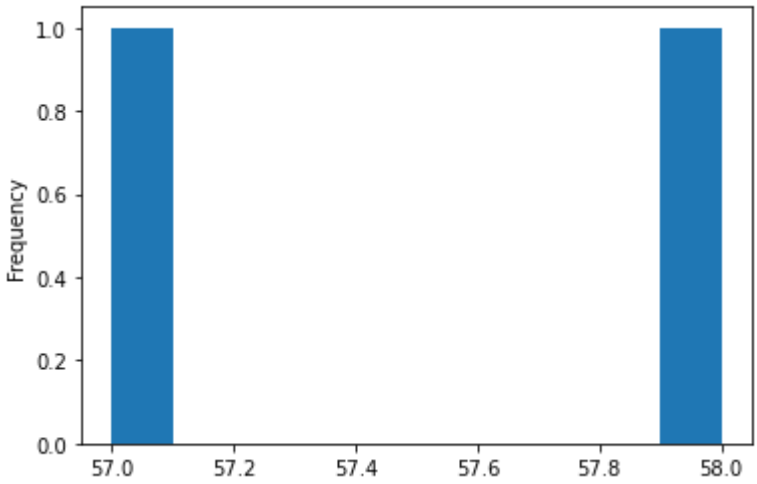
В отсортированном принте отсутсвуют False. Это значит то, что во всех строках NaN для total_income также NaN и в days_employed

Посмотрим, как распределяются строки NaN в разрезе других колонок

In [352]:
```
# количество детей
income_nan['children'].value_counts()
```

Out[352]:
```
0    2
Name: children, dtype: int64
```

In [353]:
```
# возраст
income_nan['dob_years'].plot(kind='hist');
```



In [354]:
```
# тип занятости

income_nan['income_type'].value_counts()
```

Out[354]:
```
пенсионер          1
предприниматель    1
Name: income_type, dtype: int64
```

Никаких аномалий не выявлено. Распределение примерно похоже на общее распределение данных по остальным показателям среди общей массы.

Для ответа на поставленный перед исследованием вопрос

- Есть ли зависимость между уровнем дохода и возвратом кредита в срок? строки с пропущенными значениями в колонке total_income исключаем из анализа, принимая к сведению несущественное (ДВЕ) их количество. Имеем в виду, что одновременно в такой выборке данных также будут отсутвовать пропуски в колонке days_employed

(после создания колонки категорий целей кредитования)

18.4 ...цель получения кредита ...

18.4.1 есть ли пропуски в данных о целях получения кредита?

из первого раздела известно, что таких пропусков нет. также составлен словарь, в котором выделены категории целей

Вывод

Данные подготовлены для работы.

Для ответа на вопросы

- Есть ли зависимость между наличием детей и возвратом кредита в срок?
- Есть ли зависимость между семейным положением и возвратом кредита в срок?
- Как разные цели кредита влияют на его возврат в срок? исследуем данные в data

Для ответа на вопрос

- Есть ли зависимость между уровнем дохода и возвратом кредита в срок? исследуем data_not_na

18.5 Замена типа данных

Вывод

для ответа на поставленные вопросы ти пы данных менять не надо. для обработки целей кредитлования воспользуемся лемматизацией ниже

18.6 Обработка дубликатов

Есть ли явные дубликаты в данных?

In [355]:
```
1 print('Количество явных дубликатов:', data.duplicated().sum())
```

```
Количество явных дубликатов: 0
```

Вывод

19 ! Явных дубликатов нет !

Неявные дубликаты обработаны ранее

19.1 Лемматизация

In [356]:

```
from pymystem3 import Mystem
m = Mystem()

# проведём лемматизацию уникальных значений колонки purpose
# соберём уникальные значения purpose
purpose_unique = data['purpose'].unique()
purpose_unique
```

Out[356]:

```
array(['покупка жилья', 'приобретение автомобиля',
       'дополнительное образование', 'сыграть свадьбу',
       'операции с жильем', 'образование', 'на проведение свадьбы',
       'покупка жилья для семьи', 'покупка недвижимости',
       'покупка коммерческой недвижимости', 'покупка жилой недвижимости',
       'строительство собственной недвижимости', 'недвижимость',
       'строительство недвижимости', 'на покупку подержанного автомобиля',
       'на покупку своего автомобиля',
       'операции с коммерческой недвижимостью',
       'строительство жилой недвижимости', 'жилье',
       'операции со своей недвижимостью', 'автомобили',
       'заняться образованием', 'сделка с подержанным автомобилем',
       'получение образования', 'автомобиль', 'свадьба',
       'получение дополнительного образования', 'покупка своего жилья',
       'операции с недвижимостью', 'получение высшего образования',
       'свой автомобиль', 'сделка с автомобилем',
       'профильное образование', 'высшее образование',
       'покупка жилья для сдачи', 'на покупку автомобиля', 'ремонт жилью',
       'заняться высшим образованием'], dtype=object)
```

38 уникальных значений для лемматизации

In [357]:

```
lemmas_series = []
for i in range(len(purpose_unique)):

    lemmas = m.lemmatize(purpose_unique[i])
    lemmas_series.append([lemmas])
    # print(lemmas)

print(lemmas_series)
```

```
[[['покупка', ' ', 'жилье', '\n']], [['приобретение', ' ', 'автомобиль', '\n']],
[['дополнительный', ' ', 'образование', '\n']], [['сыграть', ' ', 'свадьба',
'\n']], [['операция', ' ', 'с', ' ', 'жилье', '\n']], [['образование', '\n']],
[['на', ' ', 'проведение', ' ', 'свадьба', '\n']], [['покупка', ' ', 'жилье', ' ',
'для', ' ', 'семья', '\n']], [['покупка', ' ', 'недвижимость', '\n']], [['покупк
а', ' ', 'коммерческий', ' ', 'недвижимость', '\n']], [['покупка', ' ', 'жилой', '
', 'недвижимость', '\n']], [['строительство', ' ', 'собственный', ' ', 'недвижимос
ть', '\n']], [['недвижимость', '\n']], [['строительство', ' ', 'недвижимость',
'\n']], [['на', ' ', 'покупка', ' ', 'подержать', ' ', 'автомобиль', '\n']], [['н
а', ' ', 'покупка', ' ', 'свой', ' ', 'автомобиль', '\n']], [['операция', ' ',
'с', ' ', 'коммерческий', ' ', 'недвижимость', '\n']], [['строительство', ' ', 'жи
лой', ' ', 'недвижимость', '\n']], [['жилье', '\n']], [['операция', ' ', 'со', '
', 'свой', ' ', 'недвижимость', '\n']], [['автомобиль', '\n']], [['заниматься', '
', 'образование', '\n']], [['сделка', ' ', 'с', ' ', 'подержанный', ' ', 'автомоби
ль', '\n']], [['получение', ' ', 'образование', '\n']], [['автомобиль', '\n']],
[['свадьба', '\n']], [['получение', ' ', 'дополнительный', ' ', 'образование',
'\n']], [['покупка', ' ', 'свой', ' ', 'жилье', '\n']], [['операция', ' ', 'с', '
', 'недвижимость', '\n']], [['получение', ' ', 'высокий', ' ', 'образование',
'\n']], [['свой', ' ', 'автомобиль', '\n']], [['сделка', ' ', 'с', ' ', 'автомобил
ь', '\n']], [['профильный', ' ', 'образование', '\n']], [['высокий', ' ', 'образов
ание', '\n']], [['покупка', ' ', 'жилье', ' ', 'для', ' ', 'сдача', '\n']], [['н
а', ' ', 'покупка', ' ', 'автомобиль', '\n']], [['ремонт', ' ', 'жилье', '\n']],
[['заниматься', ' ', 'высокий', ' ', 'образование', '\n']]]
```

19.2 Сделаем отбор в отдельные таблицы строк из data, в которых в purpose встречаются ключевые слова

In [358]:

```
# готовлю таблички для наполнения данными
data_1 = data # скопировал стркутуру и данные
# удаляю все строки
data_1 = data_1.loc[data_1['children'] < 0] # заведомо значение False
data_1
```

Out[358]:

index	children	days_employed	dob_years	education_id	family_status_id	gender	income_type	de

In [359]:

```
# дублирую dataframe без строк в заготовки для группировки по ключевым словам
auto = data_1 # автомобиль
education = data_1 # образование
wedding = data_1 # свадьба
commercial_real_estate = data_1 # коммерческая недвижимость
real_estate_renovation = data_1 # ремонт жилья
real_estate = data_1 # недвижимость (частная)
no_label = data_1 # если не будет ключевых влов, включённых в отбор

```

In [360]:

```
len(data)
```

Out[360]: 21524

In [361]:

```
%%time
def create_category_purpose(row):
    lem_purpose = m.lemmatize(row['purpose'])
    if 'автомобиль' in lem_purpose:
        return 'автомобиль'
    elif 'образование' in lem_purpose:
        return 'образование'
    elif 'свадьба'  in lem_purpose:
        return 'свадьба'
    elif 'сдача'  in lem_purpose and 'жилье'  in lem_purpose:
        return 'коммерческая недвижимость'
    elif 'коммерческий' in lem_purpose and  'недвижимость'  in lem_purpose:
        return 'коммерческая недвижимость'
    elif 'ремонт'  in lem_purpose and  'недвижимость'  in lem_purpose:
        return 'ремонт недвижимости'
    elif 'ремонт'  in lem_purpose and  'жилье'  in lem_purpose:
        return 'ремонт недвижимости'
    elif 'недвижимость'  in lem_purpose:
        return 'недвижимость'
    elif 'жилье'  in lem_purpose:
        return 'недвижимость'
    else:
        return 'без группы'


# применяю функцию с созданием столбца purpose_category
data['purpose_category'] = data.apply(create_category_purpose, axis=1)
data['purpose_category'].sort_values().value_counts()

```

```
CPU times: user 922 ms, sys: 359 ms, total: 1.28 s
Wall time: 2.62 s
```

Out[361]:

```
недвижимость                 8259
автомобиль                   4315
образование                  4022
свадьба                      2348
коммерческая недвижимость    1968
ремонт недвижимости           612
Name: purpose_category, dtype: int64
```

20 Готовы отдельные срезы данных по целям запроса кредитов.

20.1 Категоризация данных

в общем объёме исходных данных есть возможность выделения категорий в : - образование, семейный статус, пол, тип занятости, цель кредитования

Трансформация текстовых значений этих колонок выгодна для дальнейшей передачи в подразделение машинного обучения.
(кроме цели кредитования)
В текущих задачах мне удобнее работать с текстовыми значениями, в которых неявные дубликаты обработаны.
Цель кредитования тоже удобно рассматривать в текстовом виде. Но сначала её проработал методами лемматизации и категоризации.

21 Шаг 3. Ответьте на вопросы

2.1 Шаг 3. Ответьте на вопросы

Уровень дохода.

Здесь для анализа влияния этого параметра на возвратность кредита следует выделить интервальные группы. Посмотрим на распределение по сумме за отсечением строк с отсутствующими данными. это срез данных data_not_na

- Есть ли зависимость между уровнем дохода и возвратом кредита в срок?

In [370]:
```
pd.options.display.float_format = '{:,.2f}'.format # установка формата вывода на
```

In [371]:
```
# сделал табличку без NaN в total_income
data_not_na = data.dropna(subset=['total_income'])
```

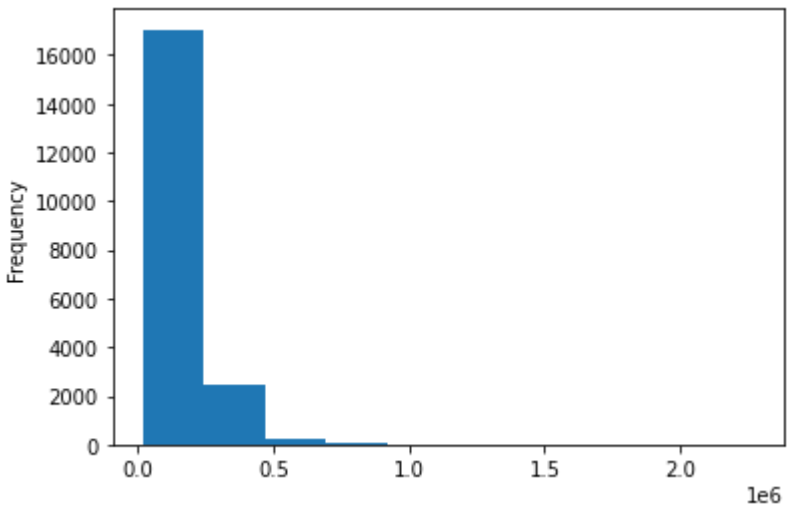
In [372]:
```
dept_0 = data_not_na[data_not_na['debt'] == 0] # отобрал строки с возвратом кре

dept_1 = data_not_na[data_not_na['debt'] == 1] # отобрал строки без возврата кр

dept_0['total_income'].describe()
```

Out[372]:
```
count      19,781.00
mean      165,762.10
std        98,154.69
min        21,205.28
25%       107,659.93
50%       143,855.21
75%       197,783.26
max     2,265,604.03
Name: total_income, dtype: float64
```

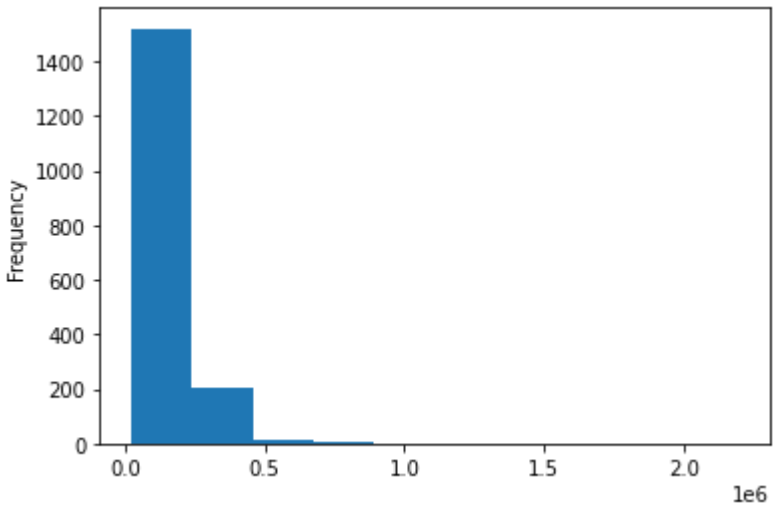
In [373]:
```
dept_0['total_income'].plot(kind = 'hist');
```



In [374]:
```
1 dept_1['total_income'].describe()
```

Out[374]:
```
count       1,741.00
mean      161,042.93
std        97,817.13
min        20,667.26
25%       108,252.81
50%       140,638.62
75%       189,395.52
max     2,200,852.21
Name: total_income, dtype: float64
```

In [375]:
```
1 dept_1['total_income'].plot(kind = 'hist');
```



21.1 Федеральная служба государственной статистики разделяет уровень жизни россиян в зависимости от доходов на следующие категории:

крайняя нищета (доходы ниже прожиточного минимума — до 7-8 тыс.р.)

нищета (доходы от одного до двух прожиточных минимума — от 8 до 12 тыс.р.)

бедность (доходы от 12 до 20 тысяч рублей в месяц)

выше бедности (доходы от 20 до 30 тысяч рублей в месяц)

средний достаток — (доходы от 30 до 60 тысяч рублей в месяц)

состоятельные — (доходы от 60 до 90 тысяч рублей в месяц)

богатые — (доходы от 90 тысяч рублей в месяц)

сверхбогатые — (доходы свыше 150 тысяч рублей в месяц)

в наших исследуемых данных есть уровни дохода от "выше бедности" до "сверхбогатые"

In [376]:

```
# выше бедности
total_income_20 = data_not_na[data_not_na['total_income'] > 20000]
total_income_20 = total_income_20[total_income_20['total_income'] <= 30000]

count = total_income_20['index'][total_income_20['debt'] == 1].count()
print('В категории выше бедности задолженность у', count, 'клиентов')
print(round((count / total_income_20['index'].count()) * 100, 2), '% от общего
print()

#  средний достаток
total_income_30 = data_not_na[data_not_na['total_income'] > 30000]
total_income_30 = total_income_30[total_income_30['total_income'] <= 60000]

count = total_income_30['index'][total_income_30['debt'] == 1].count()
print('В категории средний достаток задолженность у', count, 'клиентов')
print(round((count / total_income_30['index'].count()) * 100, 2), '% от общего
print()

# состоятельные
total_income_60 = data_not_na[data_not_na['total_income'] > 60000]
total_income_60 = total_income_60[total_income_60['total_income'] <= 90000]

count = total_income_60['index'][total_income_60['debt'] == 1].count()
print('В категории состоятельные задолженность у', count, 'клиентов')
print(round((count / total_income_60['index'].count()) * 100, 2), '% от общего
print()

#  богатые
total_income_90 = data_not_na[data_not_na['total_income'] > 90000]
total_income_90 = total_income_90[total_income_90['total_income'] <= 150000]

count = total_income_90['index'][total_income_90['debt'] == 1].count()
print('В категории богатые задолженность у', count, 'клиентов')
print(round((count / total_income_90['index'].count()) * 100, 2), '% от общего
print()

#  сверхбогатые
total_income_150 = data_not_na[data_not_na['total_income'] > 150000]

count = total_income_150['index'][total_income_150['debt'] == 1].count()
print('В категории сверхбогатые задолженность у', count, 'клиентов')
print(round((count / total_income_150['index'].count()) * 100, 2), '% от общего

```

```
В категории выше бедности задолженность у 2 клиентов
9.09 % от общего числа клиентов этой категории

В категории средний достаток задолженность у 47 клиентов
5.99 % от общего числа клиентов этой категории

В категории состоятельные задолженность у 213 клиентов
8.38 % от общего числа клиентов этой категории

В категории богатые задолженность у 707 клиентов
8.66 % от общего числа клиентов этой категории

В категории сверхбогатые задолженность у 772 клиентов
7.71 % от общего числа клиентов этой категории
```

21.2 ВЫВОД. Уровень дохода тоже влияет на возвратность кредитов

Самый ненадёжный сегмент - малоимущие (выше бедности)

Надёжный сегмент - средний достаток

Сверхбюогатые - редкий случай и обнадёживающий

Количество детей

Те же выборки:

- dept_0
- dept_1

- Есть ли зависимость между наличием детей и возвратом кредита в срок?

In [377]:
```
1 dept_0['children'].describe()
```

Out[377]:
```
count   19,781.00
mean         0.47
std          0.75
min          0.00
25%          0.00
50%          0.00
75%          1.00
max          5.00
Name: children, dtype: float64
```

In [378]:
```
1 dept_1['children'].describe()
```

Out[378]:
```
count   1,741.00
mean        0.54
std         0.78
min         0.00
25%         0.00
50%         0.00
75%         1.00
max         4.00
Name: children, dtype: float64
```

In [379]:
```
1 dept_0['children'].value_counts() # количества категорий
```

Out[379]:
```
0    13083
1     4420
2     1929
3      303
4       37
5        9
Name: children, dtype: int64
```

In [380]:
```
1 dept_1['children'].value_counts() # количества категорий
```

Out[380]:
```
0    1063
1     445
2     202
3      27
4       4
Name: children, dtype: int64
```

In [381]:
```
data_not_na['children'].value_counts() # количества категорий всего
```

Out[381]:
```
0    14146
1     4865
2     2131
3      330
4       41
5        9
Name: children, dtype: int64
```

In [383]:
```
print('Имеют задолженности по количеству детей')
for i in range(len(data['children'].value_counts())):

    print(round(((dept_1['index'][dept_1['children'] == i].count())
       / ((dept_0['index'][dept_0['children'] == i].count())
          + (dept_1['index'][dept_1['children'] == i].count())))
       * 100, 2), '% в семье', i, 'реб.')

```

```
Имеют задолженности по количеству детей
7.51 % в семье 0 реб.
9.15 % в семье 1 реб.
9.48 % в семье 2 реб.
8.18 % в семье 3 реб.
9.76 % в семье 4 реб.
0.0 % в семье 5 реб.
```

21.3 ВЫВОД. Количество детей имеет некоторое влияние на возвратность кредитов

Самый ненадёжный сегмент - два ребенка

Надёжный сегмент - трое детей и бездетные

Пятеро детей - редкий случай и надёжный

Семейное положение

Те же выборки:

- dept_0
- dept_1

- Есть ли зависимость между семейным положением и возвратом кредита в срок?

In [384]:
```
dept_0['family_status_id'].value_counts() # количества категорий
```

Out[384]:
```
0    11448
1     3788
4     2539
3     1109
2      897
Name: family_status_id, dtype: int64
```

In [385]:
```
dept_1['family_status_id'].value_counts() # количества категорий
```

Out[385]:
```
0    931
1    388
4    274
3     85
2     63
Name: family_status_id, dtype: int64
```

А теперь возьму в долях от общего количества по каждой категории

In [386]:
```

print('Общее число клиентов по категориям')


data_not_na['family_status_id'].value_counts()
```

Общее число клиентов по категориям

Out[386]:
```
0    12379
1     4176
4     2813
3     1194
2      960
Name: family_status_id, dtype: int64
```

In [388]:
```
print('Имеют задолженности по категориям')
print('клиенты разного семейного статуса:')

for i in range(len(family_status_dict)):
    print(round(((dept_1['index'][dept_1['family_status_id'] == i].count())
       / ((dept_0['index'][dept_0['family_status_id'] == i].count())
          + (dept_1['index'][dept_1['family_status_id'] == i].count())))
       * 100, 2), "%", family_status_dict['family_status'][i])
```

```
Имеют задолженности по категориям
клиенты разного семейного статуса:
7.52 % женат / замужем
9.29 % гражданский брак
6.56 % вдовец / вдова
7.12 % в разводе
9.74 % Не женат / не замужем
```

Вывод

Семейное положение имеет значение

21.4 ВЫВОД. Семейное положение имеет значение на возвратность кредитов

Самый ненадёжный сегмент - не женат / не замужем

Самый надёжный сегмент - вдовец / вдова

Разведённые и семейные - примерно рядом

Цели кредитования

Те же выборки:

- dept_0
- dept_1

- Как разные цели кредита влияют на его возврат в срок?

In [389]:
```
1 dept_0['purpose_category'].value_counts() # количества категорий
```

Out[389]:
```
недвижимость                 7662
автомобиль                   3912
образование                  3651
свадьба                      2162
коммерческая недвижимость    1817
ремонт недвижимости           577
Name: purpose_category, dtype: int64
```

In [390]:
```
1 dept_1['purpose_category'].value_counts() # количества категорий
```

Out[390]:
```
недвижимость                 596
автомобиль                   403
образование                  370
свадьба                      186
коммерческая недвижимость    151
ремонт недвижимости           35
Name: purpose_category, dtype: int64
```

In [391]:
```
1  print('Имеют задолженности по категориям')
2  print(round(((dept_1['index'][dept_1['purpose_category'] == 'недвижимость'].cou
3      ((dept_0['index'][dept_0['purpose_category'] == 'недвижимость'].count())+\
4          (dept_1['index'][dept_1['purpose_category'] == 'недвижимость'].count())
5  print(round(((dept_1['index'][dept_1['purpose_category'] == 'автомобиль'].count
6      ((dept_0['index'][dept_0['purpose_category'] == 'автомобиль'].count())+\
7          (dept_1['index'][dept_1['purpose_category'] == 'автомобиль'].count())))
8  print(round(((dept_1['index'][dept_1['purpose_category'] == 'образование'].coun
9      ((dept_0['index'][dept_0['purpose_category'] == 'образование'].count())+\
10         (dept_1['index'][dept_1['purpose_category'] == 'образование'].count())))
11 print(round(((dept_1['index'][dept_1['purpose_category'] == 'свадьба'].count())
12     ((dept_0['index'][dept_0['purpose_category'] == 'свадьба'].count())+\
13         (dept_1['index'][dept_1['purpose_category'] == 'свадьба'].count()))) *
14 print(round(((dept_1['index'][dept_1['purpose_category'] == 'коммерческая недви
15     ((dept_0['index'][dept_0['purpose_category'] == 'коммерческая недвижимость'
16         (dept_1['index'][dept_1['purpose_category'] == 'коммерческая недвижимос
17              * 100, 2), '% коммерческая недвижимость')
18 print(round(((dept_1['index'][dept_1['purpose_category'] == 'ремонт недвижимост
19     ((dept_0['index'][dept_0['purpose_category'] == 'ремонт недвижимости'].coun
20         (dept_1['index'][dept_1['purpose_category'] == 'ремонт недвижимости'].c
21              * 100, 2), '% ремонт недвижимости')
22
23 print('от общего числа клиентов этой категории')
24
25
```

```
Имеют задолженности по категориям
7.22 % недвижимость
9.34 % автомобиль
9.2 % образование
7.92 % свадьба
7.67 % коммерческая недвижимость
5.72 % ремонт недвижимости
от общего числа клиентов этой категории
```

21.5 ВЫВОД. Цель кредитования имеет значение на возвратность кредитов

Самый ненадёжный сегмент - автомобиль

Самый надёжный сегмент - недвижимость

Свадьба - тоже хорошая цель по возвратности

А вот образование лишь ненамного надёжнее, чем автомобиль

22 Шаг 4. Общий вывод

По результатам исследования можно отметить влияние на возвраственность кредитов всех рассматриваемых факторов

23 Самый высокорисковый клиент:

С низким доходом, двумя детьми, с целью на автомобиль

In [392]:
```
print('Таких клиентов в банке ....', data['index'][(data['debt'] == 0) &\
     (data['total_income'] <= 30000) & (data['children'] == 2) & \
        (data['purpose_category'] == 'автомобиль')].count(), '!!!')
```

```
Таких клиентов в банке .... 0 !!!
```

24 Прекрасная работа отдела продаж!

Повысим ставки. Следующая "вилка" в доходах: до 60000 р

In [393]:
```
print('В банке', data['index'][(data['total_income'] <= 60000) & (data['childre
     & (data['purpose_category'] == 'автомобиль')].count()
, "таких клиентов, а с проблемой по возврату из них -", data['index'][(data['de
     & (data['total_income'] <= 60000) & (data['children'] == 2) & \
        (data['purpose_category'] == 'автомобиль')].count()
, 'то есть процент невозврата', round((data['index'][(data['debt'] == 1) &\
     (data['total_income'] <= 60000) & (data['children'] == 2) &\
         (data['purpose_category'] == 'автомобиль')].count()*100)/data['index']
             <= 60000) & (data['children'] == 2) & (data['purpose_category'] ==
                'автомобиль')].count(), 2), '%')
```

```
В банке 12 таких клиентов, а с проблемой по возврату из них - 4 то есть процент не
возврата 33.33 %
```

25 33 % огромная цифра!

26 Самый низкокорисковый клиент:

С более, чем средним достатком, с тремя детьми, семейные, с целью кредитования на недвижимость

In [394]:

```
print('В банке', data['index'][(data['total_income'] >= 60000) & (data['childre
    (data['purpose_category'] == 'недвижимость')].count()
, "таких клиентов, а с проблемой по возврату из них -", data['index'][(data['de
    (data['total_income'] >= 60000) & (data['children'] == 3) & (data['purpose_
, 'то есть процент невозврата', round((data['index'][(data['debt'] == 1) & \
    (data['total_income'] >= 60000) & (data['children'] == 3) & \
        (data['purpose_category'] == 'недвижимость')].count()*100)/data['index'
            (data['children'] == 3) & (data['purpose_category'] == 'недвижимос
```

```
В банке 123 таких клиентов, а с проблемой по возврату из них - 10 то есть процент
невозврата 8.13 %
```