Клиенты банка, склонные к оттоку

Описание:
региональный банк, расположенный в трёх российских городах - Ярославль, Рыбинск и Ростов Великий,
столкнулся с проблемой оттока клиентов, масштабы которого угрожающе увеличиваются.
Отделу маркетинга необходимо понять возможные причины такого поведения
и разработать стратегию для удержания клиентов.
Для анализа предоставлена следующая информация:
идентификатор, пол и возраст клиента, город оформления банковского продукта,
кредитный скоринг, баланс счёта, количество баллов собственности, количество продуктов,
наличие кредитной карты, расчётный доход в месяц, активность за последние 30 дней, признак оттока.

Задача:
выявить характерные признаки, указывающие на склонность к оттоку,
разделить клиентов на несколько сегментов, приоритизировать сегменты по доле отточных,
составить рекомендации для отдела маркетинга.

Краткое содержание:

- загрузим и исследуем данные;
- выделим основные признаки, влияющие на отток;
- проверим гипотезы;
- проведём сегментацию клиентов;
- сформулируем выводы и предложим рекомендации для снижения оттока.

Часть 1. Знакомство с данными

На этом шаге:

- загрузим все необходимые библиотеки;
- сохраним предоставленные данные в переменную df;
- оценим объём и качество данных;
- изучим признаки и их тип (числовые или категориальные);
- проверим данные на наличие дубликатов;

- оценим количество пропусков.

In [1]:
```
# импорт библиотек для работы с данными
import pandas as pd
import numpy as np
pd.options.display.float_format ='{:,.2f}'.format

# библиотеки для визуализации
import seaborn as sns
import matplotlib.pyplot as plt

# для статистического анализа
import scipy.stats as st
```

In [2]:
```
# библиотека, позволяющая указывать несколько директорий хранения файла
import os

file_path_1 = 'C:/Users/havar/OneDrive/Рабочий стол/ЯндексПрактикум/АналитикДанныхРасширенный/ПРОЕКТЫ/Выпускной/bank_scrooge.csv'
file_path_2 = '/content/bank_scrooge.csv'

if os.path.exists(file_path_1):
    df = pd.read_csv(file_path_1, delimiter=',')
elif os.path.exists(file_path_2):
    df = pd.read_csv(file_path_2, delimiter=',')
else:
    print('Something is wrong')
```

In [3]:
```
# первые строки датасета
df.head()
```

Out[3]:

	USERID	score	city	gender	age	equity	balance	products	credit_card	last_activity	EST_SALARY	churn
0	183012	850.00	Рыбинск	Ж	25.00	1	59,214.82	2	0	1	75,719.14	1
1	146556	861.00	Рыбинск	Ж	37.00	5	850,594.33	3	1	0	86,621.77	0
2	120722	892.00	Рыбинск	Ж	30.00	0	NaN	1	1	1	107,683.34	0
3	225363	866.00	Ярославль	Ж	51.00	5	1,524,746.26	2	0	1	174,423.53	1
4	157978	730.00	Ярославль	М	34.00	5	174.00	1	1	0	67,353.16	1

In [4]:
```
# нормализуем названия столбцов и выведем общую информацию о таблице
df.columns = map(str.lower, df.columns)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   userid         10000 non-null  int64  
 1   score          10000 non-null  float64
 2   city           10000 non-null  object 
 3   gender         10000 non-null  object 
 4   age            9974 non-null   float64
 5   equity         10000 non-null  int64  
 6   balance        7705 non-null   float64
 7   products       10000 non-null  int64  
 8   credit_card    10000 non-null  int64  
 9   last_activity  10000 non-null  int64  
 10  est_salary     10000 non-null  float64
 11  churn          10000 non-null  int64  
dtypes: float64(4), int64(6), object(2)
memory usage: 937.6+ KB
```

Описание данных:

- `userid` — идентификатор пользователя;
- `score` — баллы кредитного скоринга;
- `city` — город
- `gender` — пол;
- `age` — возраст;
- `equity` — количество баллов собственности;
- `balance` — баланс на счёте;
- `products` — количество продуктов, которыми пользуется клиент;
- `credit_card` — есть ли кредитная карта;
- `last_activity` — активность клиента за последние 30 дней;
- `est_salary` — оценочный доход клиента;
- `churn` — признак оттока.

In [5]:
```
# количество дубликатов в колонке с id пользователя
df.userid.duplicated().sum()
```

Out[5]:
```
73
```

In [6]:
```
# список повторяющихся id пользователя
dupl = df.query('userid.duplicated() == True')
dupl.userid.unique()
```

Out[6]:
```
array([190253, 210662, 131419, 187635, 220816, 221156, 208081, 170312,
       185748, 211130, 217643, 226719, 197364, 155765, 126368, 218868,
       151662, 143592, 141265, 208815, 152479, 217619, 208738, 120258,
       188957, 228075, 172142, 163207, 210135, 219343, 164676, 214031,
       140377, 117943, 116540, 210792, 191520, 198635, 226550, 149365,
       216848, 148826, 206759, 210898, 227795, 210627, 221197, 123461,
       181526, 162053, 127440, 199312, 222480, 183510, 200863, 150667,
       202983, 155872, 187459, 217826, 141945, 129785, 160075, 185829,
       221809, 171751, 195884, 163657, 124450, 168998, 140934, 217412,
       175730], dtype=int64)
```

In [7]:
```
# строки с id 190253
df.query('userid == 190253')
```

Out[7]:

	userid	score	city	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn
231	190253	823.00	Рыбинск	М	37.00	4	373,348.39	2	0	1	131,947.92	1
1583	190253	726.00	Ярославль	М	49.00	0	NaN	1	1	1	177,700.78	0

In [8]:
```
# строки с id 175730
df.query('userid == 175730')
```

Out[8]:

	userid	score	city	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn
7753	175730	846.00	Ярославль	Ж	32.00	7	216,764.74	4	1	1	77,100.85	1
9970	175730	816.00	Рыбинск	М	36.00	4	477,892.07	3	1	0	81,100.60	0

От сотрудников банка нам известно, что идентификатор пользователя - это всегда уникальное значение.
Разные города для одного id можно объяснить перемещениями клиентов и оформлением банковского продукта в другом городе.
Другими словами, клиент может приобрести разные продукты в разных городах, но все они будут оформлены на один уникальный номер id.
Но в таблице также есть разное значение пола и возраста для одного id, разный кредитный скоринг, разный баланс и доход. Как будто это разные люди.
Но так как это невозможно по правилам банка, будем считать данное наблюдение ошибкой ввода или вывода данных. Дубликаты удалим.

In [9]:
```
# копия датасета до удаления дубликатов
df_copy = df.copy()

# удалим дубликаты
df = df.drop_duplicates(subset=['userid'])
```

```
# проверим результат
df.userid.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
# количество удалённых строк
len(df_copy) - len(df)
```

Out[10]:
```
73
```

Проверим данные на наличие неявных дубликатов, которые могли образоваться в результате разного написания одних и тех же слов.

In [11]:
```
# список уникальных названий городов
df.city.unique()
```

Out[11]:
```
array(['Рыбинск', 'Ярославль', 'Ростов'], dtype=object)
```

In [12]:
```
# список уникальных обозначений пола
df.gender.unique()
```

Out[12]:
```
array(['Ж', 'М'], dtype=object)
```

Неявных дубликатов нет.

In [13]:
```
# количество пропусков по каждой колонке
df.isna().sum().sort_values(ascending=False)
```

Out[13]:
```
balance          2260
age                 4
userid              0
score               0
city                0
gender              0
equity              0
products            0
credit_card         0
last_activity       0
est_salary          0
churn               0
dtype: int64
```

In [14]:
```
# доля пропусков по каждой колонке
df.isna().mean().sort_values(ascending=False)
```

Out[14]:
```
balance          0.23
age              0.00
userid           0.00
score            0.00
city             0.00
gender           0.00
equity           0.00
products         0.00
credit_card      0.00
last_activity    0.00
est_salary       0.00
churn            0.00
dtype: float64
```

Вывод 1

Таблица содержит 10 тысяч наблюдений по 12 признакам.

6 признаков являются числовыми: кредитный скоринг, возраст, баллы собственности, баланс, количество продуктов и доход.

3 признака являются бинарными или dummy-переменными, принимающими значения 0 и 1: кредитная карта, активность, признак оттока. В бинарный формат можно также перевести пол клиента и город.

1 категориальный признак - id пользователя.

В данных обнаружены дубликаты по id клиента - 73 строки. Удалены.

В столбцах с возрастом и балансом есть пропуски.

Из-за пропусков тип данных в столбце с возрастом - float, вместо int.

Часть 2. Предобработка данных

На этом шаге:

- обработаем пропуски;
- исправим тип данных в столбце с возрастом;
- переведем пол и город в бинарный формат для построения матрицы корреляций.

4 пропуска с возрастом удалим, так как количество незначительное, а попытка заполнить чем-либо исказит результаты.

In [15]:
```
# копия датасета до удаления пропусков
df_copy2 = df.copy()

# удаление пропусков в столбце 'age'
df = df.dropna(subset=['age'])

# количество пропусков в столбце с возрастом
df['age'].isna().sum()
```

Out[15]:
```
0
```

In [16]:
```
# количество удалённых строк
len(df_copy2) - len(df)
```

Out[16]:
```
4
```

In [17]:
```
# исправление типа данных на целочисленный
df['age'] = df['age'].astype('int')
```

In [18]:
```
# процент пропусков в столбце 'balance'
df.balance.isna().mean()
```

Out[18]:
```
0.22775370351708152
```

В столбце с балансом 23% пропусков.
Удалить такое количество без ущерба для данных мы не можем.
Заполнять средним или медианой также нецелесообразно, так как это исказит результаты.
Посмотрим, есть ли в данных наблюдения с нулевым балансом:

In [19]:
```
# минимальный баланс
df.balance.min()
```

Out[19]:
```
0.0
```

Да, в таблице встречается нулевой баланс.

Сравним статистику клиентов с нулевым балансом со статистикой клиентов с пропусками.

In [20]:
```
# статистика по клиентам с нулевым балансом
df.query('balance == 0').describe()
```

Out[20]:

	userid	score	age	equity	balance	products	credit_card	last_activity	est_salary	churn
count	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00
mean	164,143.50	771.50	19.00	1.00	0.00	2.00	1.00	0.50	44,212.42	0.00
std	22,185.48	58.69	1.41	1.41	0.00	0.00	0.00	0.71	28,759.32	0.00
min	148,456.00	730.00	18.00	0.00	0.00	2.00	1.00	0.00	23,876.51	0.00
25%	156,299.75	750.75	18.50	0.50	0.00	2.00	1.00	0.25	34,044.46	0.00
50%	164,143.50	771.50	19.00	1.00	0.00	2.00	1.00	0.50	44,212.42	0.00
75%	171,987.25	792.25	19.50	1.50	0.00	2.00	1.00	0.75	54,380.38	0.00
max	179,831.00	813.00	20.00	2.00	0.00	2.00	1.00	1.00	64,548.33	0.00

In [21]:
```
# статистика по клиентам с пропусками в балансе
df.query('balance.isna()').describe()
```

Out[21]:

	userid	score	age	equity	balance	products	credit_card	last_activity	est_salary	churn
count	2,260.00	2,260.00	2,260.00	2,260.00	0.00	2,260.00	2,260.00	2,260.00	2,260.00	2,260.00
mean	171,975.57	865.33	42.13	0.05	NaN	1.16	0.82	0.55	226,700.56	0.01
std	33,277.19	89.67	11.95	0.23	NaN	0.40	0.39	0.50	199,981.67	0.08
min	114,182.00	642.00	18.00	0.00	NaN	0.00	0.00	0.00	20,274.03	0.00
25%	143,067.00	871.00	33.00	0.00	NaN	1.00	1.00	0.00	120,457.01	0.00
50%	172,498.50	903.00	40.00	0.00	NaN	1.00	1.00	1.00	174,755.35	0.00
75%	201,354.25	922.00	49.00	0.00	NaN	1.00	1.00	1.00	240,451.81	0.00
max	229,145.00	990.00	86.00	3.00	NaN	3.00	1.00	1.00	1,333,687.36	1.00

Как оказалось, нулевой баланс встречается только в двух случаях, что на уровне статистической погрешности.
Таким образом, мы можем выделить клиентов с пропусками по балансу в отдельную категорию, заменив пропуски нулями.
Другими словами, 0 на балансе теперь будет означать "нет данных".

In [22]:
```
# замена пропусков на 0
df.loc[df.balance.isna() == True, 'balance'] = 0

# количество пропусков в столбце с балансом
df.balance.isna().sum()
```

Out[22]:
```
0
```

Перед тем, как закодировать категориальные переменные пол и город в индикаторные 0/1, сохраним исходный датасет в отдельной переменной для удобства построения визуализаций.

In [23]:
```
# копия датасета для построения визуализаций
banks = df.copy()
```

In [24]:
```
# преобразуем пол в двоичный формат: 1 - мужчина, 0 - женщина
gender = {'gender': {'М':1, 'Ж':0}}
```

```
df = df.replace(gender)
df.head()
```

Out[24]:

	userid	score	city	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn
0	183012	850.00	Рыбинск	0	25	1	59,214.82	2	0	1	75,719.14	1
1	146556	861.00	Рыбинск	0	37	5	850,594.33	3	1	0	86,621.77	0
2	120722	892.00	Рыбинск	0	30	0	0.00	1	1	1	107,683.34	0
3	225363	866.00	Ярославль	0	51	5	1,524,746.26	2	0	1	174,423.53	1
4	157978	730.00	Ярославль	1	34	5	174.00	1	1	0	67,353.16	1

In [25]:
```
# закодируем город в бинарный формат
df = pd.get_dummies(df, prefix = '', prefix_sep = '', dtype=int, drop_first=False)
df.head()
```

Out[25]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
0	183012	850.00	0	25	1	59,214.82	2	0	1	75,719.14	1	0	1	0
1	146556	861.00	0	37	5	850,594.33	3	1	0	86,621.77	0	0	1	0
2	120722	892.00	0	30	0	0.00	1	1	1	107,683.34	0	0	1	0
3	225363	866.00	0	51	5	1,524,746.26	2	0	1	174,423.53	1	0	0	1
4	157978	730.00	1	34	5	174.00	1	1	0	67,353.16	1	0	0	1

In [26]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9923 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   userid         9923 non-null   int64  
 1   score          9923 non-null   float64
 2   gender         9923 non-null   int64  
 3   age            9923 non-null   int32  
 4   equity         9923 non-null   int64  
 5   balance        9923 non-null   float64
 6   products       9923 non-null   int64  
 7   credit_card    9923 non-null   int64  
 8   last_activity  9923 non-null   int64  
 9   est_salary     9923 non-null   float64
 10  churn          9923 non-null   int64  
 11  Ростов         9923 non-null   int32  
 12  Рыбинск        9923 non-null   int32  
 13  Ярославль      9923 non-null   int32  
dtypes: float64(3), int32(4), int64(7)
memory usage: 1007.8 KB
```

Вывод 2

Удалили 73 строки с дубликатами по id пользователя и 4 строки с пропусками в столбце с возрастом.
Пропуски в столбце с балансом заполнили нулями.
Обозначение пола закодировали в бинарный формат: 1 - мужчина, 0 - женщина.
Города также перевели в бинарный формат,
заменив общий столбец city на три столбца отдельно для каждого города, где 0 - нет, 1 - да.
Исходный датасет до кодировки сохранили в переменную banks.
Тип данных с возрастом исправили на целочисленный.

Часть 3. Исследовательский анализ данных

На этом шаге:

- изучим общую статистику;
- посмотрим на распределения признаков в разрезе оттока;
- построим матрицу корреляций.

Общая статистика

In [27]:
```
df.describe().T
```

Out[27]:

	count	mean	std	min	25%	50%	75%	max
userid	9,923.00	171,731.48	33,694.37	94,561.00	142,720.00	172,635.00	201,157.50	229,145.00
score	9,923.00	848.73	65.40	642.00	802.00	853.00	900.00	1,000.00
gender	9,923.00	0.50	0.50	0.00	0.00	1.00	1.00	1.00
age	9,923.00	42.73	12.18	18.00	33.00	40.00	51.00	86.00
equity	9,923.00	2.63	1.98	0.00	0.00	3.00	4.00	9.00
balance	9,923.00	639,611.57	1,777,946.59	0.00	63,675.68	376,590.60	805,244.39	119,113,552.01
products	9,923.00	1.87	0.79	0.00	1.00	2.00	2.00	5.00
credit_card	9,923.00	0.68	0.47	0.00	0.00	1.00	1.00	1.00
last_activity	9,923.00	0.52	0.50	0.00	0.00	1.00	1.00	1.00
est_salary	9,923.00	147,886.49	139,363.90	2,546.30	75,252.12	119,719.33	174,673.28	1,395,064.45
churn	9,923.00	0.18	0.39	0.00	0.00	0.00	0.00	1.00
Ростов	9,923.00	0.14	0.35	0.00	0.00	0.00	0.00	1.00
Рыбинск	9,923.00	0.27	0.44	0.00	0.00	0.00	1.00	1.00
Ярославль	9,923.00	0.59	0.49	0.00	0.00	1.00	1.00	1.00

В выборке одинаковое количество мужчин и женщин.
Средний возраст клиентов 42-43 года. Минимальный возраст - 18 лет, максимальный - 86.
Максимальный баланс на счёте 119 миллионов - выглядит как выброс.
Так же обращает на себя внимание минимальный расчётный доход в 2,5 тысячи.
Минимальное количество продуктов 0 может быть ошибкой, так как известно, что человек становится клиентом банка только при условии оформления какого-либо банковского продукта.
68% клиентов имеют кредитную карту.
Чуть больше половины клиентов проявляли какую-либо активность за последние 30 дней.
Почти 60% всех клиентов приходится на Ярославль.
Ушедших в отток клиентов - 18% от общего числа.

In [28]:
```
# все наблюдения, в которых количество продуктов равно 0
df.query('products == 0')
```

Out[28]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
8957	147837	962.00	0	79	3	0.00	0	0	0	25,063.96	1	0	1	0

Учитывая довольно преклонный возраст клиентки и отсутствие средств на счёте можно предположить, что признак оттока связан с уходом из жизни. Но мы не знаем этого наверняка, поэтому удалим это единственное наблюдение с нулевым количеством продуктов.

In [29]:
```
# копия датасета до удаления
df_copy3 = df.copy()

# удаление строки с нулевым количеством продуктов по индексу
df = df.drop (index = 8957)

# количество наблюдений, в которых products равно 0
len(df.query('products == 0'))
```

Out[29]:
```
0
```

In [30]:
```
# количество удалённых строк
len(df_copy3) - len(df)
```

Out[30]:
```
1
```

In [31]:
```
# удаление строки с нулевым количеством продуктов из таблицы "banks"
banks = banks.drop (index = 8957)
```

Выбросы

Выброс - это наблюдение, которое лежит аномально далеко от других значений в наборе данных.
Это может быть ошибка ввода/вывода данных,
но может быть и просто редкий признак.

Низкие доходы

In [32]:
```
# срез по доходу менее 10 тысяч
df.query('est_salary < 10000')
```

Out[32]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
505	205035	725.00	0	77	3	73.00	2	0	1	8,729.84	0	0	1	0
1700	155567	747.00	1	25	0	5,750.92	1	1	0	7,054.82	0	1	0	0
1753	161271	836.00	0	19	4	7,703.05	4	1	1	5,043.14	0	0	0	1
1883	151268	852.00	0	63	5	373,070.37	1	1	1	8,587.79	0	1	0	0
1988	160979	788.00	1	24	4	118,438.82	2	1	0	8,401.97	0	0	1	0
2174	143340	830.00	0	72	2	294,375.63	3	1	1	8,032.18	0	0	0	1
2186	201407	938.00	0	69	1	325,483.52	3	0	1	3,487.33	0	0	1	0
2663	140105	821.00	0	75	7	631,805.72	1	1	0	8,894.57	0	1	0	0
3606	187819	839.00	0	56	5	1,027,438.02	2	0	0	7,571.80	0	0	1	0
3689	219561	817.00	1	25	3	121,065.39	4	1	1	7,522.14	0	1	0	0
4799	193482	812.00	0	25	2	105,603.12	2	1	0	8,849.89	0	0	1	0
4860	193417	823.00	0	64	2	277,249.69	2	0	0	5,341.50	0	1	0	0
6044	176879	836.00	0	66	3	226,648.06	2	1	1	8,707.54	0	1	0	0
6087	166086	788.00	1	26	5	99,885.68	1	0	1	9,766.13	0	0	1	0
7637	147622	850.00	0	73	2	263,390.68	3	0	1	9,530.00	0	1	0	0
8783	114221	948.00	0	26	1	77,092.89	1	1	0	2,546.30	0	0	0	1
9124	148483	811.00	0	75	5	542,971.64	3	1	1	8,226.26	0	0	0	1
9299	211611	826.00	1	26	0	150,934.62	1	1	0	9,425.19	0	0	1	0
9365	171110	788.00	0	76	2	223,228.01	2	0	0	8,260.33	0	0	1	0

Довольно интересный получился срез:
среди клиентов либо молодые люди, либо пожилые и все клиенты не отточные.

Особый интерес вызывает клиент с id 187819 с балансом на счёте более миллиона и доходом 7,5 тысяч - ей 56 лет, 5 баллов собственности и не пользуется кредитной картой.

В срезе также есть молодые люди с баллами собственности 3-5 и балансом 100 и более тысяч при небольшом доходе.
А с другой стороны - женщина 77 лет, 3 балла собственности и баланс 73 рубля, доход 8.7 тысяч
(правда, была активность за последний месяц, возможно, снятие наличных или перевод).

Клиент с самым низким по всей выборке доходом -
это девушка 26 лет из Ярославля, имеет 1 балл собственности, кредитную карту и 77 тысяч на счёте.

Таким образом, можно предположить, что большинство клиентов с низкими доходами -
это молодые люди (возможно, студенты) и пенсионеры, у которых есть родственники или опекуны, оказывающие им финансовую помощь.
Другими словами, низкий уровень дохода - это не ошибка, а определённая категория клиентов.

Высокие доходы

In [33]:
```
# срез с доходами от 1 миллиона рублей
df.query('est_salary > 1000000')
```

Out[33]:

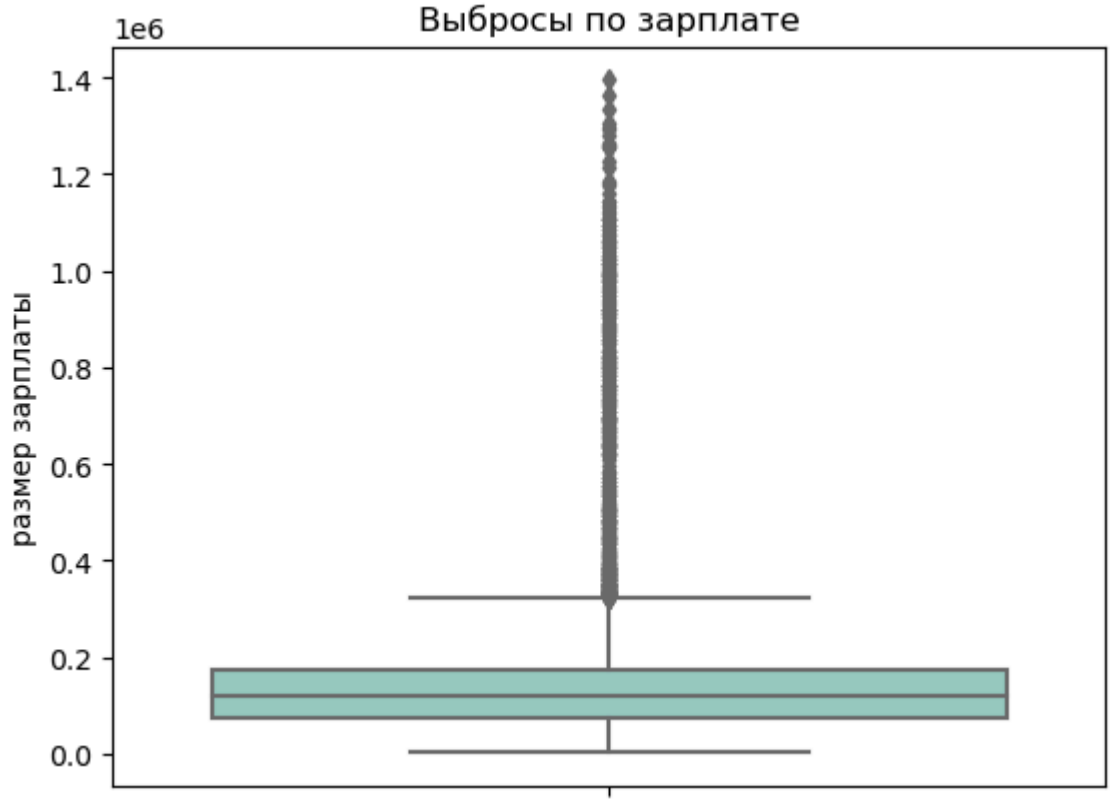
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
125	227092	1,000.00	1	32	5	19,757,180.85	2	1	1	1,024,626.50	0	0	0	1
149	218801	958.00	1	34	0	0.00	1	1	0	1,292,825.74	0	0	0	1
178	226887	890.00	1	49	4	3,956,103.10	2	1	0	1,120,528.70	0	0	0	1
250	210357	766.00	1	27	0	0.00	1	1	0	1,015,754.12	0	0	0	1
296	204143	925.00	1	25	0	0.00	2	1	0	1,001,009.40	0	0	0	1
302	164876	942.00	1	40	0	0.00	1	1	0	1,256,537.74	0	0	0	1
663	180853	949.00	1	45	0	0.00	1	1	0	1,142,166.48	0	0	0	1
996	127574	749.00	1	35	0	0.00	1	1	0	1,071,600.78	0	0	0	1
1027	119793	1,000.00	1	50	5	12,909,691.02	1	1	0	1,253,653.40	0	0	0	1
1273	136567	961.00	1	31	0	0.00	1	1	0	1,015,386.14	1	0	0	1
1301	221610	954.00	0	65	0	0.00	1	0	0	1,076,227.99	0	0	0	1
1468	162749	940.00	1	32	2	4,508,306.65	2	1	0	1,260,919.56	0	0	0	1
1600	197637	785.00	1	41	0	0.00	1	1	0	1,307,090.18	0	0	0	1
1814	205696	718.00	1	25	0	0.00	1	1	1	1,180,070.17	0	0	0	1
1845	132310	756.00	1	61	0	0.00	1	1	1	1,106,453.04	0	0	0	1
2068	201574	983.00	1	54	3	11,060,145.42	2	0	1	1,046,662.18	0	0	0	1
2291	115628	980.00	0	35	6	1,813,817.40	2	1	1	1,130,928.15	0	0	0	1
2424	216987	941.00	1	35	0	0.00	1	1	0	1,121,684.22	0	0	0	1
2901	164659	700.00	0	60	0	0.00	2	1	1	1,033,801.28	0	0	0	1
3103	203255	999.00	1	36	4	10,641,153.66	2	0	0	1,042,118.22	0	0	0	1
3143	193935	920.00	1	35	0	0.00	1	1	1	1,087,009.59	0	0	0	1
3283	155712	958.00	1	56	5	3,797,685.28	2	1	1	1,185,539.68	0	0	0	1
3370	140920	741.00	0	46	0	0.00	1	1	1	1,214,615.32	0	0	0	1
3577	132066	941.00	0	33	0	0.00	1	1	0	1,010,637.75	0	0	0	1
3748	149547	698.00	0	41	0	0.00	1	1	1	1,093,305.09	0	0	0	1
3930	162174	960.00	0	58	4	426,392.77	2	0	0	1,008,647.93	0	0	0	1
4005	163640	912.00	1	35	4	15,624,095.80	2	0	1	1,395,064.45	0	0	0	1

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
4091	156736	983.00	1	24	0	0.00	2	0	1	1,134,459.80	0	0	0	1
4473	216422	956.00	1	27	0	0.00	1	1	0	1,333,687.36	0	0	0	1
4637	149430	961.00	1	37	5	4,054,667.81	3	0	0	1,363,549.52	0	0	0	1
4645	216411	951.00	1	51	7	8,612,191.25	4	1	1	1,048,772.81	1	0	0	1
5022	121489	976.00	1	33	5	3,683,418.21	2	1	0	1,059,197.75	0	0	0	1
5124	131849	911.00	0	31	5	5,102,651.08	3	0	0	1,056,085.68	0	0	0	1
5160	174746	731.00	0	30	0	0.00	2	1	0	1,296,838.08	0	0	0	1
5273	214332	943.00	0	44	0	0.00	1	1	1	1,025,315.82	0	0	0	1
5550	193209	933.00	1	31	0	0.00	1	1	1	1,058,923.29	0	0	0	1
5693	115473	955.00	1	37	0	0.00	1	0	0	1,016,911.93	0	0	0	1
5708	162331	916.00	1	50	5	4,362,619.72	4	1	1	1,022,536.72	1	0	0	1
5865	120519	948.00	0	58	0	0.00	1	1	1	1,087,045.85	0	0	0	1
6126	213236	955.00	1	35	0	0.00	1	1	1	1,094,180.41	0	0	0	1
6163	198495	794.00	0	41	0	0.00	1	1	0	1,176,946.66	0	0	0	1
6244	138336	921.00	0	39	0	0.00	2	1	1	1,093,763.67	0	0	0	1
6499	224019	936.00	1	31	5	2,931,344.63	2	0	1	1,108,269.91	1	0	0	1
6720	179582	742.00	0	54	0	0.00	2	1	1	1,108,338.87	0	0	0	1
6778	120257	732.00	1	35	0	0.00	1	1	1	1,032,974.96	0	0	0	1
6892	152005	927.00	0	34	5	3,782,453.73	3	0	1	1,142,435.95	1	0	0	1
7040	227762	742.00	0	31	0	0.00	1	1	0	1,112,272.62	0	0	0	1
8016	138788	939.00	0	41	0	0.00	1	1	0	1,016,777.65	0	0	0	1
8019	126189	934.00	1	39	0	0.00	1	1	1	1,226,911.03	0	0	0	1
8147	135243	951.00	1	38	5	14,134,432.03	2	1	0	1,281,547.73	0	0	0	1
8213	188837	750.00	1	53	0	0.00	1	1	0	1,162,181.88	0	0	0	1
8484	221720	1,000.00	0	35	5	21,549,943.63	2	0	0	1,051,902.65	0	0	0	1
9052	122486	944.00	1	41	0	0.00	1	1	1	1,071,226.54	0	0	0	1
9060	136649	741.00	0	43	0	0.00	1	1	0	1,027,463.69	0	0	0	1

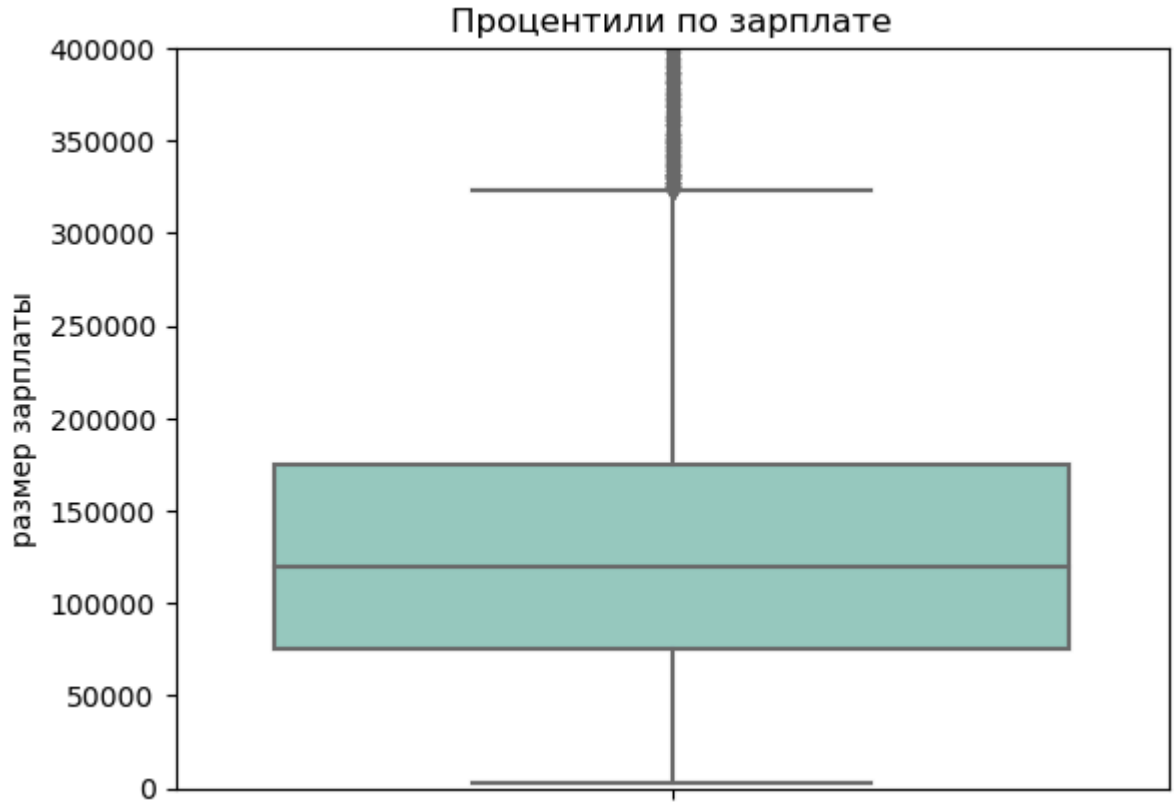
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Ярославль
9103	123965	939.00	1	34	5	9,346,657.08	1	1	1	1,263,028.49	1	0	0	1
9253	177375	758.00	0	31	0	0.00	3	1	0	1,064,019.29	0	0	0	1
9366	181981	914.00	0	44	0	0.00	2	1	0	1,107,132.01	0	0	0	1
9711	193979	754.00	0	53	0	0.00	1	1	0	1,261,408.41	0	0	0	1
9813	123334	713.00	1	58	0	0.00	2	1	1	1,060,562.13	0	0	0	1

В клуб зарплатных миллионеров входят исключительно жители Ярославля,
при этом у значительной части из них 0 баллов собственности и 0 на балансе счёта.
Ноль на счёте может быть ошибкой ввода/вывода данных или признаком того, что у клиента есть счета в других банках,
а собственность может быть записана на детей и супругов.
Т.е., высокие зарплаты - это также отдельная категория.

In [34]:
```
# визуализация выбросов по зарплате
sns.boxplot(data=df,
            y='est_salary',
            palette='Set3')
plt.ylabel('размер зарплаты')
plt.title('Выбросы по зарплате');
```



In [35]:
```
# визуализация по зарплате с ограничением по оси Y
sns.boxplot(data=df,
            y='est_salary',
            palette='Set3')
plt.ylim(0, 400000)
plt.ylabel('размер зарплаты')
plt.title('Процентили по зарплате');
```



In [36]:
```
# процентили по зарплате
df.est_salary.quantile([.25, .50, .75, .95])
```

Out[36]:
```
0.25     75,254.04
0.50    119,735.43
0.75    174,679.64
0.95    316,862.99
Name: est_salary, dtype: float64
```

Большинство наблюдений лежат в диапазоне от 75 до 175 тысяч рублей.
25% клиентов имеют доход от 2.5 до 75 тысяч,
20% имеют доход от 175 до 317 тысяч
и у 5% доходит до 1.4 миллиона.

In [37]:
```
# минимальный доход
df.est_salary.min()
```

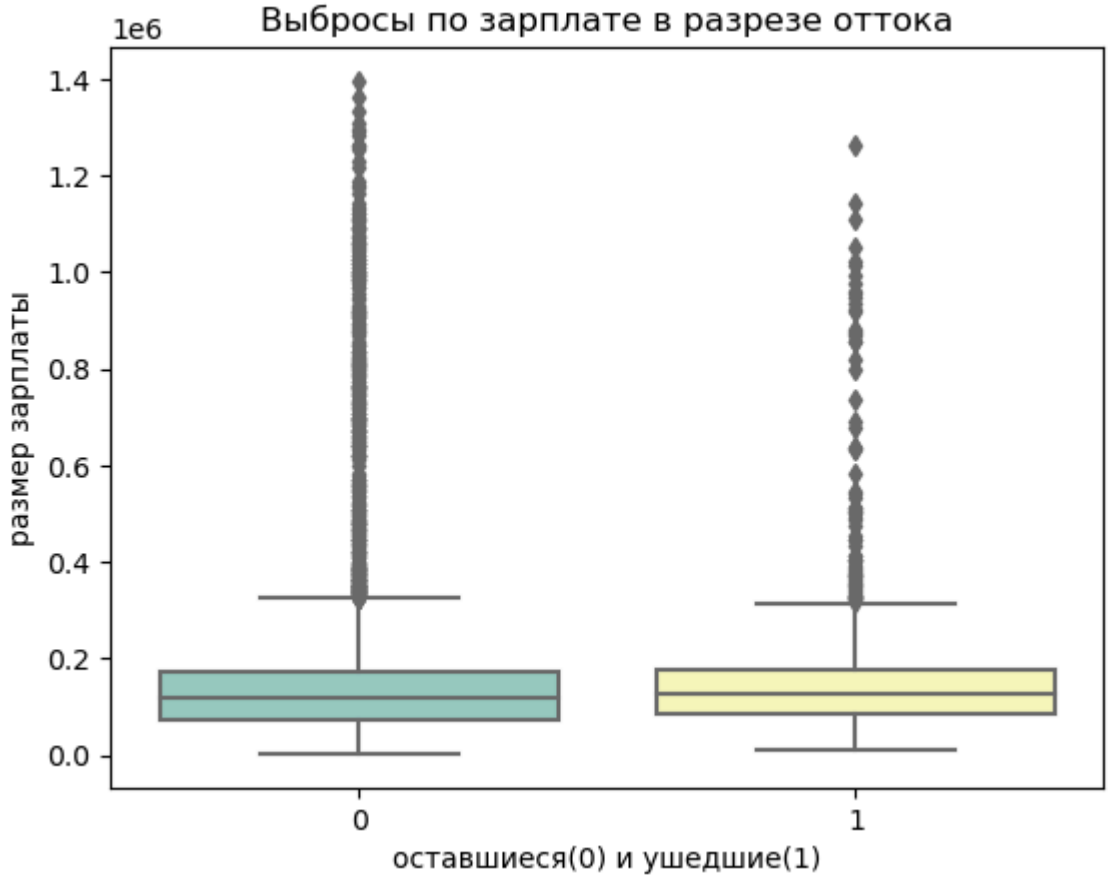
Out[37]:
```
2546.3
```

In [38]:
```
# максимальный доход
df.est_salary.max()
```

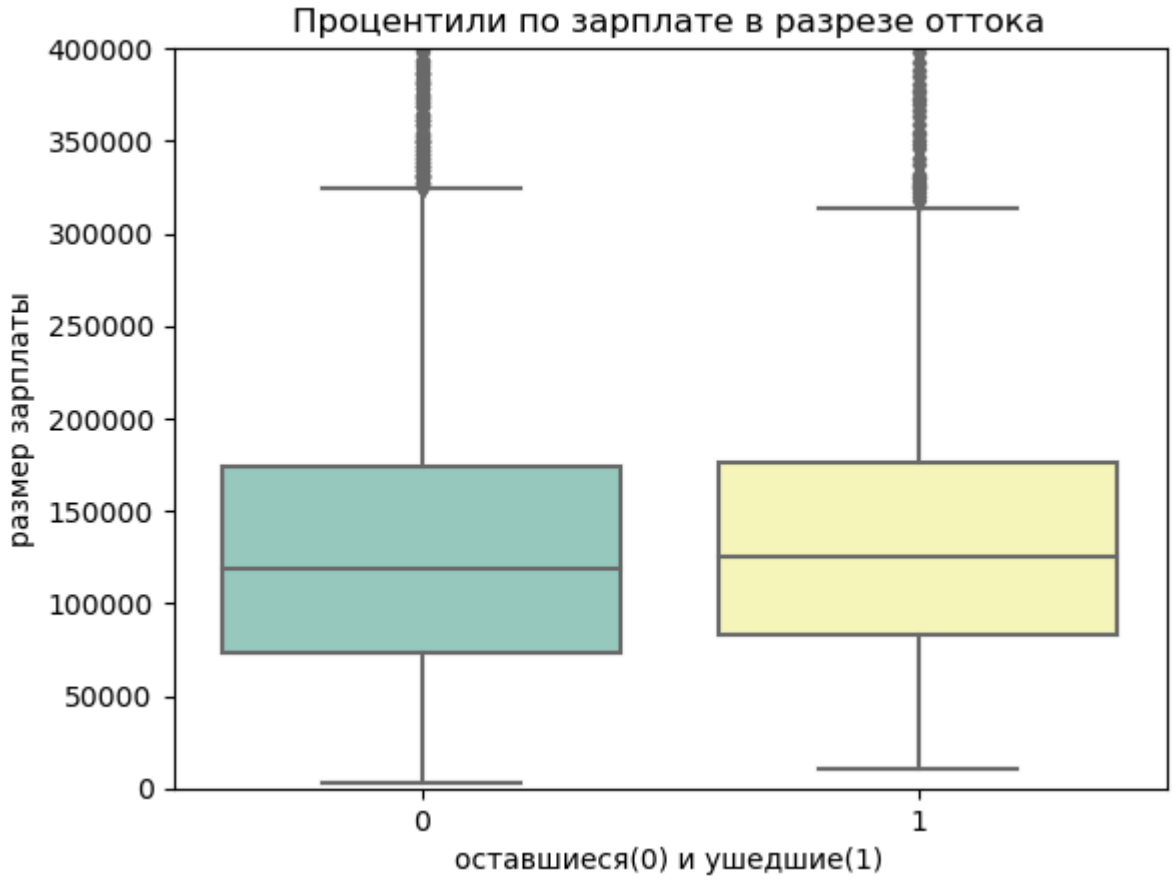
Out[38]:
```
1395064.45
```

In [39]:
```
# разделим выборку на две части:
# на оставшихся(0) и ушедших в отток(1)
churn_0 = df.query('churn == 0')
churn_1 = df.query('churn == 1')
```

In [40]:
```
# визуализация выбросов по зарплате в разрезе оттока
sns.boxplot(data=df,
            x='churn',
            y='est_salary',
            palette='Set3')
plt.xlabel('оставшиеся(0) и ушедшие(1)')
plt.ylabel('размер зарплаты')
plt.title('Выбросы по зарплате в разрезе оттока');
```



In [41]:
```
# визуализация по зарплате в разрезе оттока
# с ограничением по оси Y
sns.boxplot(data=df,
            x='churn',
            y='est_salary',
            palette='Set3')
plt.ylim(0, 400000)
plt.xlabel('оставшиеся(0) и ушедшие(1)')
plt.ylabel('размер зарплаты')
plt.title('Процентили по зарплате в разрезе оттока');
```



In [42]:
```
# процентили по зарплате оставшихся клиентов
salary_percentile_0 = (churn_0['est_salary']
                       .quantile([.25, .50, .75, .95])
                       .reset_index()
                       .rename(columns={'index': 'процентиль',
                                        'est_salary': 'зарплата_оставшихся'}))

# процентили по зарплате ушедших клиентов
salary_percentile_1 = (churn_1['est_salary']
                       .quantile([.25, .50, .75, .95])
                       .reset_index()
                       .rename(columns={'index': 'процентиль',
                                        'est_salary': 'зарплата_ушедших'}))

# объединение в одну таблицу
salary_percentile = pd.merge(salary_percentile_0,
                             salary_percentile_1,
                             on='процентиль')
salary_percentile
```

Out[42]:

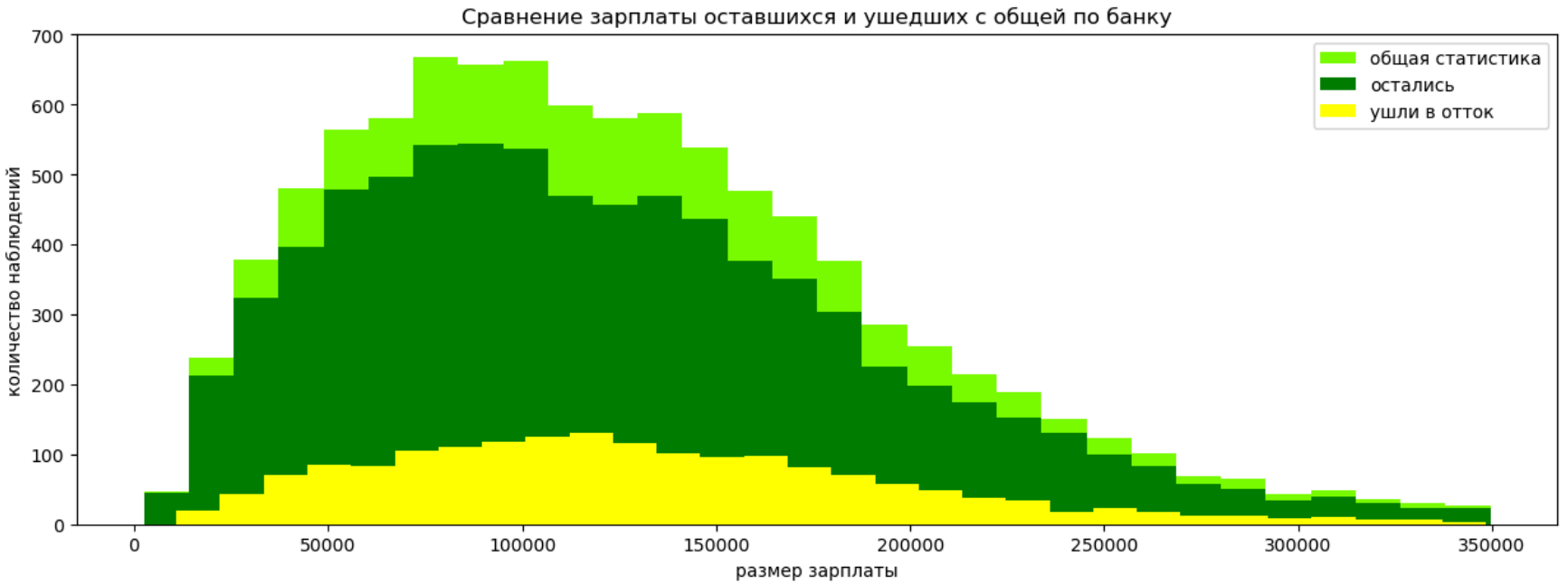
	процентиль	зарплата_оставшихся	зарплата_ушедших
0	0.25	73,487.07	83,285.66
1	0.50	118,248.51	125,408.88
2	0.75	174,106.04	176,017.84
3	0.95	320,951.55	304,056.79

Статистика по зарплате в группе ушедших хоть и не сильно, но отличается от оставшихся -
немного выше медиана, первый и третий квартили, а 95 процентиль, напротив, ниже
и максимальный выброс имеет меньшее значение.
Максимальная отточность наблюдается в группе клиентов с зарплатой в пределах 125 тысяч.
Посмотрим распределение зарплат на гистограмме, но сначала ограничим выборку отсечением правого хвоста,
чтобы выбросы не мешали визуальной оценке.

In [43]:
```
# срезы по зарплате до 350 тыс. и признаку отточности
df_2 = df.query('est_salary < 350000')
churn_0_2 = df_2.query('churn == 0')
churn_1_2 = df_2.query('churn == 1')
```

In [44]:
```
# визуализация распределения зарплат по группам отточных и оставшихся
# в сравнении с общей статистикой
n_bins = 30
plt.subplots(figsize = (15,5))

plt.hist(df_2.est_salary,
         n_bins,
         color='LawnGreen',
         label='общая статистика')

plt.hist(churn_0_2.est_salary,
         n_bins,
         color='Green',
         label='остались')

plt.hist(churn_1_2.est_salary,
         n_bins,
         color='Yellow',
         label='ушли в отток')
```

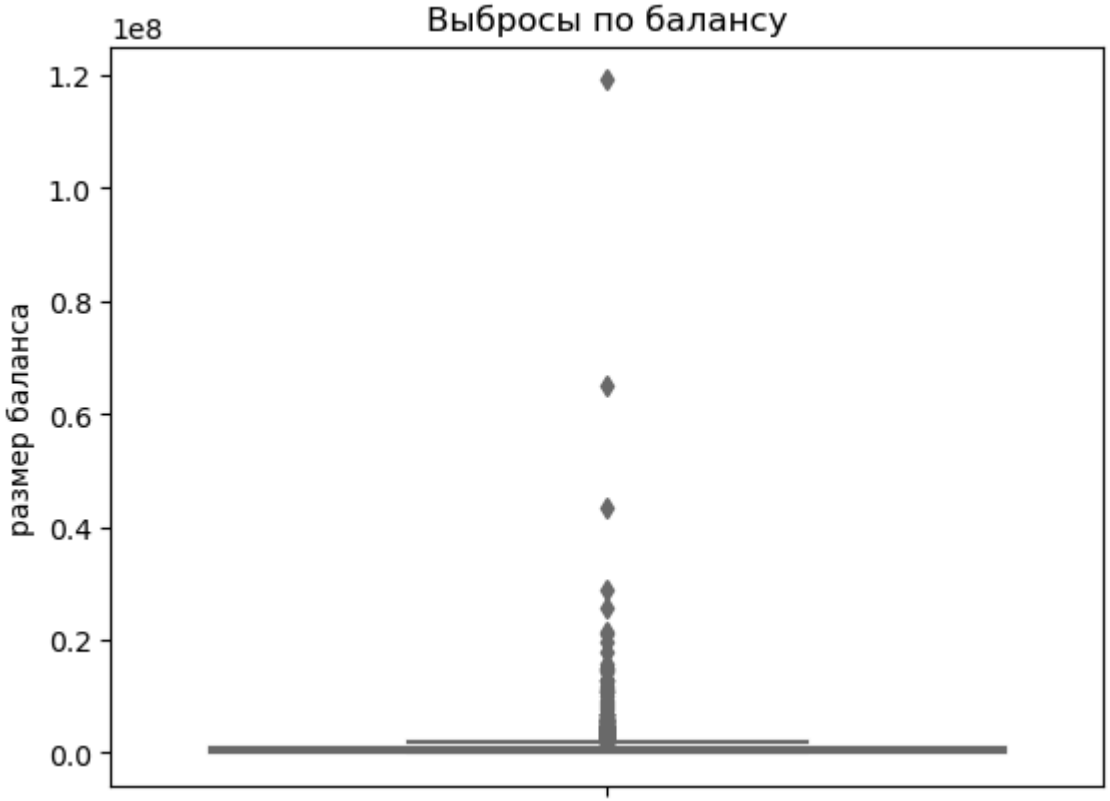
```
plt.xlabel('размер зарплаты')
plt.ylabel('количество наблюдений')
plt.title('Сравнение зарплаты оставшихся и ушедших с общей по банку')
plt.legend();
```



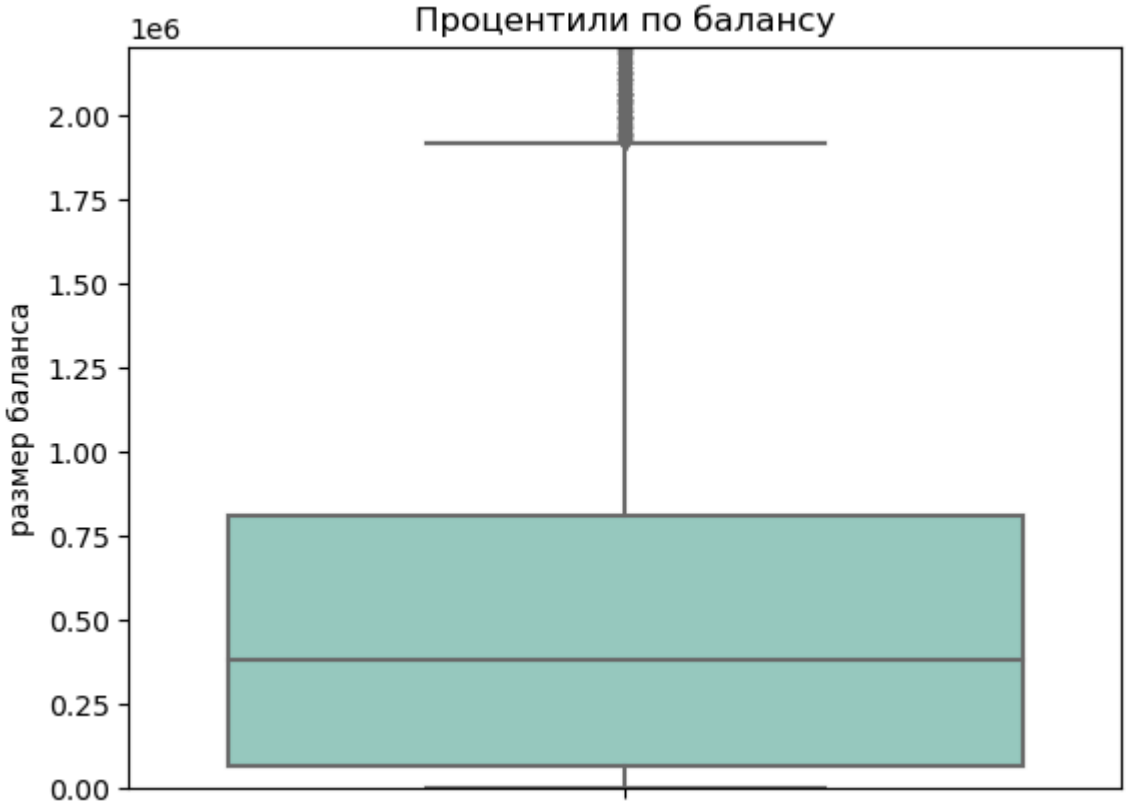
Пик распределения для отточных клиентов относительно оставшихся смещён вправо, в сторону бОльших зарплат.

Максимальный баланс

In [45]:
```
# визуализация выбросов по балансу
sns.boxplot(data=df,
            y='balance',
            palette='Set3')
plt.ylabel('размер баланса')
plt.title('Выбросы по балансу');
```



In [46]:
```
# визуализация по балансу с ограничением по оси Y
sns.boxplot(data=df,
            y='balance',
            palette='Set3')
plt.ylim(0, 2200000)
plt.ylabel('размер баланса')
plt.title('Процентили по балансу');
```



In [47]:
```
# процентили по балансу
df.balance.quantile([.25, .50, .75, .95])
```

Out[47]:
```
0.25      63,741.43
0.50     376,715.58
0.75     805,331.39
0.95   1,847,426.28
Name: balance, dtype: float64
```

Ящик с усами заметно смещён вниз в сторону нуля, что отражает большое количество счетов с нулевым балансом.
Четверть клиентов имеют на балансе не более 64 тысяч рублей.
У половины клиентов баланс не превышает 377 тысяч.
У 5% на счёте хранится свыше 1.8 миллиона.

Другими словами, типичным балансом для данной выборки являются суммы от 64 до 805 тысяч,
25% имеют низкий баланс от 0 до 64 тысяч, 20% высокий - от 805 тыс. до 1.8 млн.
и 5% выбросов от 1.8 до 119 млн.

In [48]:
```
# максимальный баланс
df.balance.max()
```

Out[48]:
```
119113552.01
```

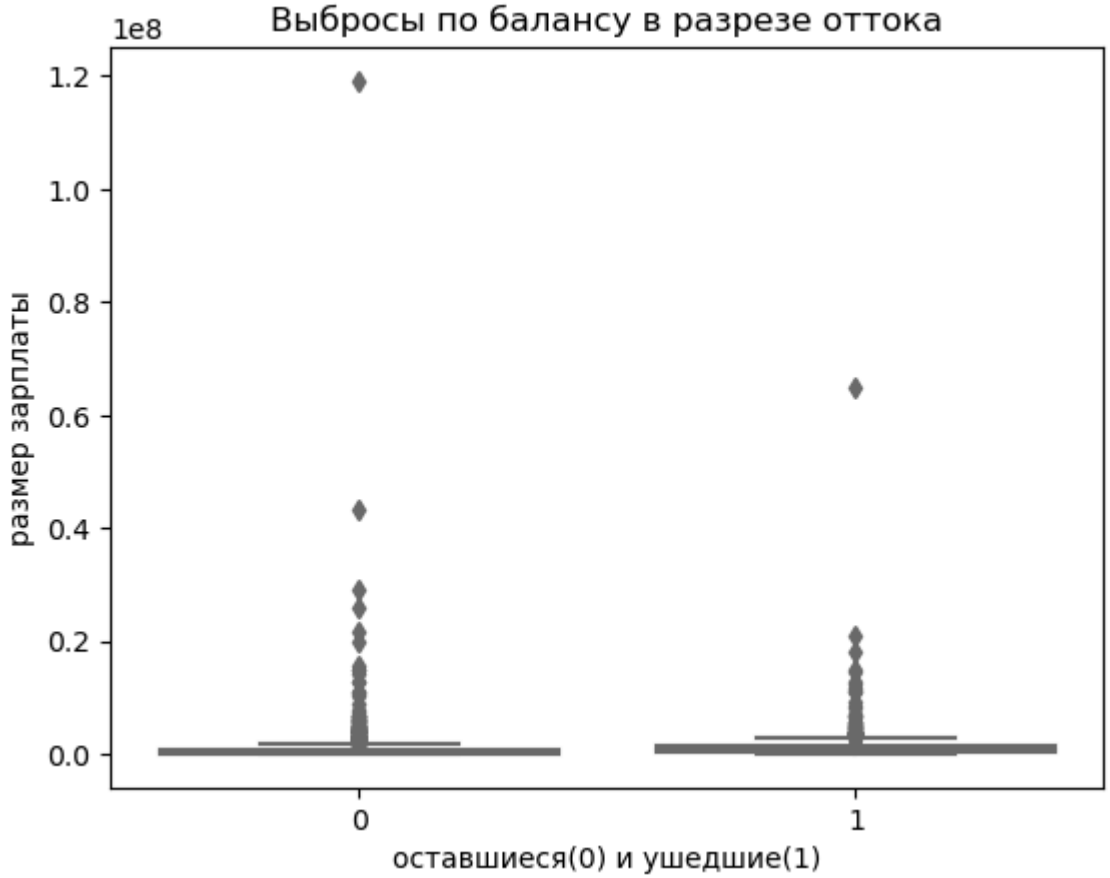
In [49]:
```
# срез с балансом от 100 миллионов
df.query('balance > 100000000')
```

Out[49]:

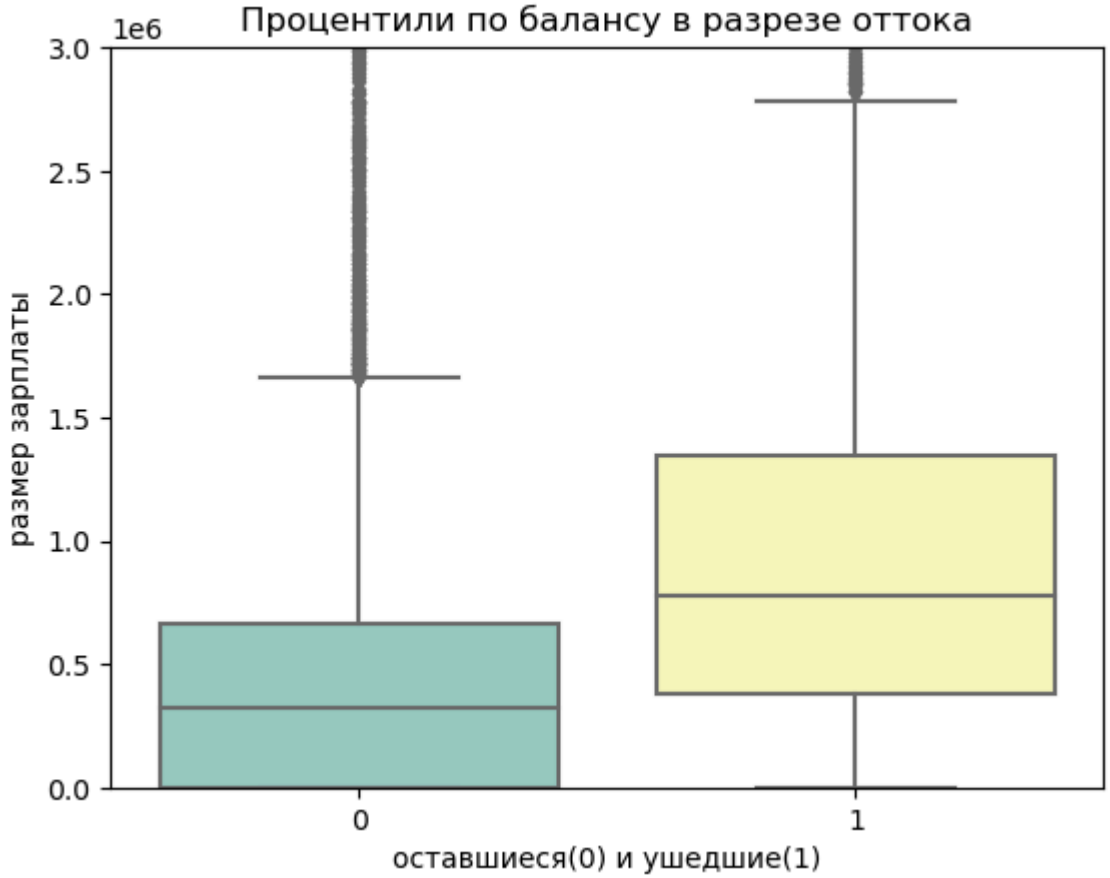
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов	Рыбинск	Яросла
7597	156149	900.00	1	62	5	119,113,552.01	2	1	1	138,041.31	0	0	0	

Владелец счёта с балансом 119 миллионов - мужчина предпенсионного возраста из Ярославля, 5 баллов собственности, доход 138
тысяч, был активен последние 30 дней, пользуется двумя продуктами, есть кредитная карта.
Данное наблюдение хоть и очень редкое, но теоретически, вполне реалистичное.
У нас нет оснований утверждать, что это ошибка.

In [50]:
```
# визуализация выбросов по балансу в разрезе оттока
sns.boxplot(data=df,
            x='churn',
            y='balance',
            palette='Set3')
plt.xlabel('оставшиеся(0) и ушедшие(1)')
plt.ylabel('размер зарплаты')
plt.title('Выбросы по балансу в разрезе оттока');
```



In [51]:
```
# визуализация по балансу в разрезе оттока
# с ограничением по оси Y
sns.boxplot(data=df,
            x='churn',
            y='balance',
            palette='Set3')
plt.ylim(0, 3000000)
plt.xlabel('оставшиеся(0) и ушедшие(1)')
plt.ylabel('размер зарплаты')
plt.title('Процентили по балансу в разрезе оттока');
```



In [52]:
```
# процентили по балансу оставшихся клиентов
balance_percentile_0 = (churn_0['balance']
                        .quantile([.25, .50, .75, .95])
                        .reset_index()
                        .rename(columns={'index': 'процентиль',
                                         'balance': 'баланс_оставшихся'}))

# процентили по балансу ушедших клиентов
balance_percentile_1 = (churn_1['balance']
                        .quantile([.25, .50, .75, .95])
                        .reset_index()
                        .rename(columns={'index': 'процентиль',
                                         'balance': 'баланс_ушедших'}))

# объединение в одну таблицу
balance_percentile = pd.merge(balance_percentile_0,
                              balance_percentile_1,
                              on='процентиль')
balance_percentile
```

Out[52]:

	процентиль	баланс_оставшихся	баланс_ушедших
0	0.25	0.00	382,038.08
1	0.50	321,692.92	775,399.47
2	0.75	665,669.85	1,347,250.18
3	0.95	1,448,353.69	2,980,906.85

В отличие от зарплаты, статистика по балансу между группой оставшихся и ушедших заметно отличается -
все процентили для ушедших клиентов практически в два раза больше,
а максимальный выброс значительно меньше.
Посмотрим распределение баланса на гистограмме,
Но сначала ограничим датасет по балансу от 2.5 тысяч до 1.5 миллиона,
поскольку большое количество нулей (более 2 тысяч наблюдений) и выбросы помешают визуальной оценке.

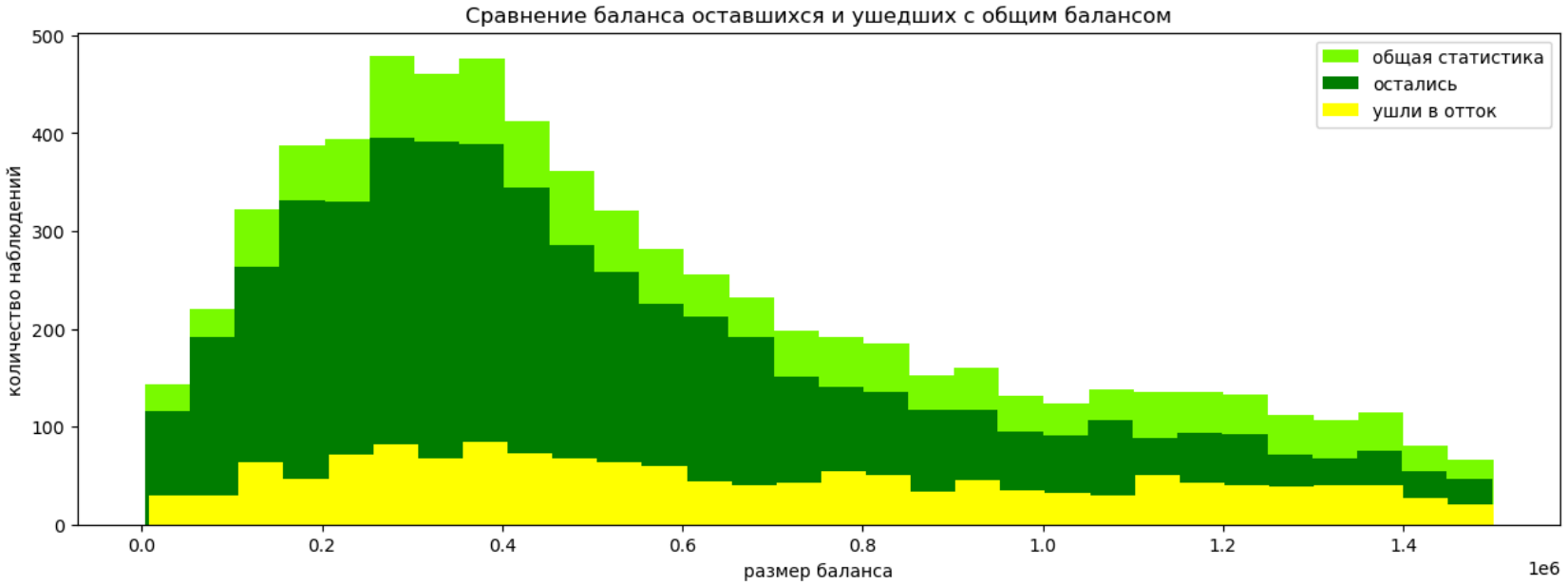
In [53]:
```
f'Количество наблюдений с нулевым балансом: {len(df.query("balance == 0"))}'
```

Out[53]:
```
'Количество наблюдений с нулевым балансом: 2261'
```

In [54]:
```
# срезы по балансу от 2.5 тыс. до 1.5 млн.
# и признаку отточности
df_3 = df.query('balance > 2500 & balance < 1500000')
churn_0_3 = df_3.query('churn == 0')
churn_1_3 = df_3.query('churn == 1')
```

In [55]:
```
# визуализация распределения баланса по группам отточных и оставшихся
# в сравнении с общей статистикой
n_bins = 30
plt.subplots(figsize = (15,5))

plt.hist(df_3.balance,
         n_bins,
         color='LawnGreen',
         label='общая статистика')

plt.hist(churn_0_3.balance,
         n_bins,
         color='Green',
         label='остались')
```

```
plt.hist(churn_1_3.balance,
         n_bins,
         color='Yellow',
         label='ушли в отток')

plt.xlabel('размер баланса')
plt.ylabel('количество наблюдений')
plt.title('Сравнение баланса оставшихся и ушедших с общим балансом')
plt.legend();
```



Баланс отточных клиентов распределён более равномерно в сравнении с оставшимися, нет явно выраженных пиков.

Также следует отметить, что среди отточных нет клиентов с нулевым балансом.

In [56]:
```
x = len(df.query('balance > est_salary')) / len(df) * 100
f'Процент клиентов, у которых баланс больше дохода: {round(x)}'
```

Out[56]:
```
'Процент клиентов, у которых баланс больше дохода: 73'
```

Подавляющее большинство клиентов склонны к накоплению и ,возможно, активному приумножению своих доходов.

Распределение признаков

Распределение признака - это закономерность встречаемости разных его значений.
Когда крайние значения признака встречаются достаточно редко,
а близкие к средней величине - достаточно часто,
такое распределение считается нормальным.

Функции

stat_group

In [57]:
```
def stat_group(sign, x_0, группа_0, x_1, группа_1):
    """Функция для вывода статистики для определённого признака по двум группам клиентов.

    На вход функция принимает название признака,
    столбец с признаком из первого и столбец с признаком из второго среза данных (x_0 и x_1),
    которые образовались при делении датасета на какие-либо две группы.
    А также названия полученных групп, например:
    мужчины и женщины, оставшиеся и ушедшие и т.п.
    """
    pass
    d_0 = x_0.describe().reset_index()
    d_0.columns = [sign, группа_0]
    d_1 = x_1.describe().reset_index()
    d_1.columns = [sign, группа_1]
    d = pd.merge(d_0, d_1, left_on=sign, right_on=sign)
    display(d)
```

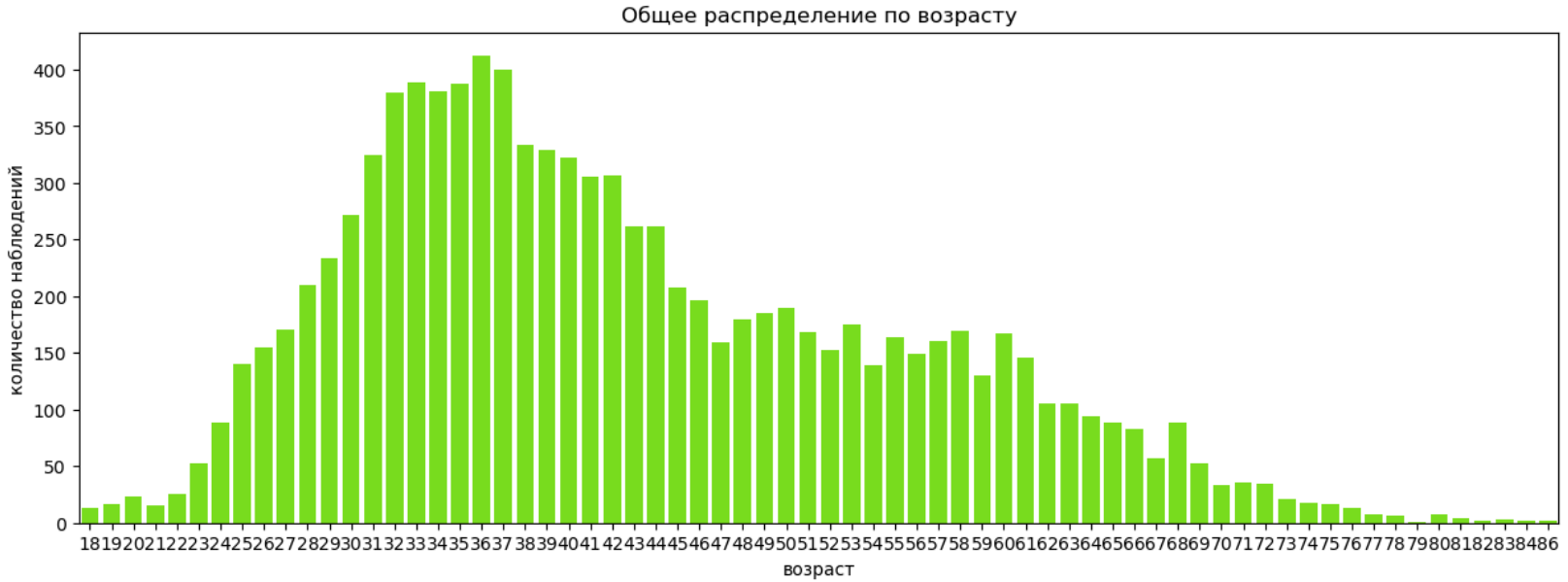
countplot_group

In [58]:
```
def countplot_group(x_0, группа_0, x_1, группа_1, X_name, Y_name, title):
    """Функция для построения графиков частоты встречаемости признака по двум группам клиентов.

    На вход функция принимает признак из первого и признак из второго среза данных (x_0 и x_1),
    которые образовались при делении датасета на какие-либо две группы.
    А также названия полученных групп, например:
    мужчины и женщины, оставшиеся и ушедшие и т.п.,
```

```
    названия осей X и Y, название графика.
    """
    pass
    plt.subplots(figsize = (15,5))
    sns.countplot(x=x_0, color='Yellow', label=группа_0)
    sns.countplot(x=x_1, color='Green', label=группа_1, alpha=.66)
    sns.set_context(rc={'font.size':11})
    plt.xlabel(X_name)
    plt.ylabel(Y_name)
    plt.title(title)
    plt.legend();
```

Возраст клиентов

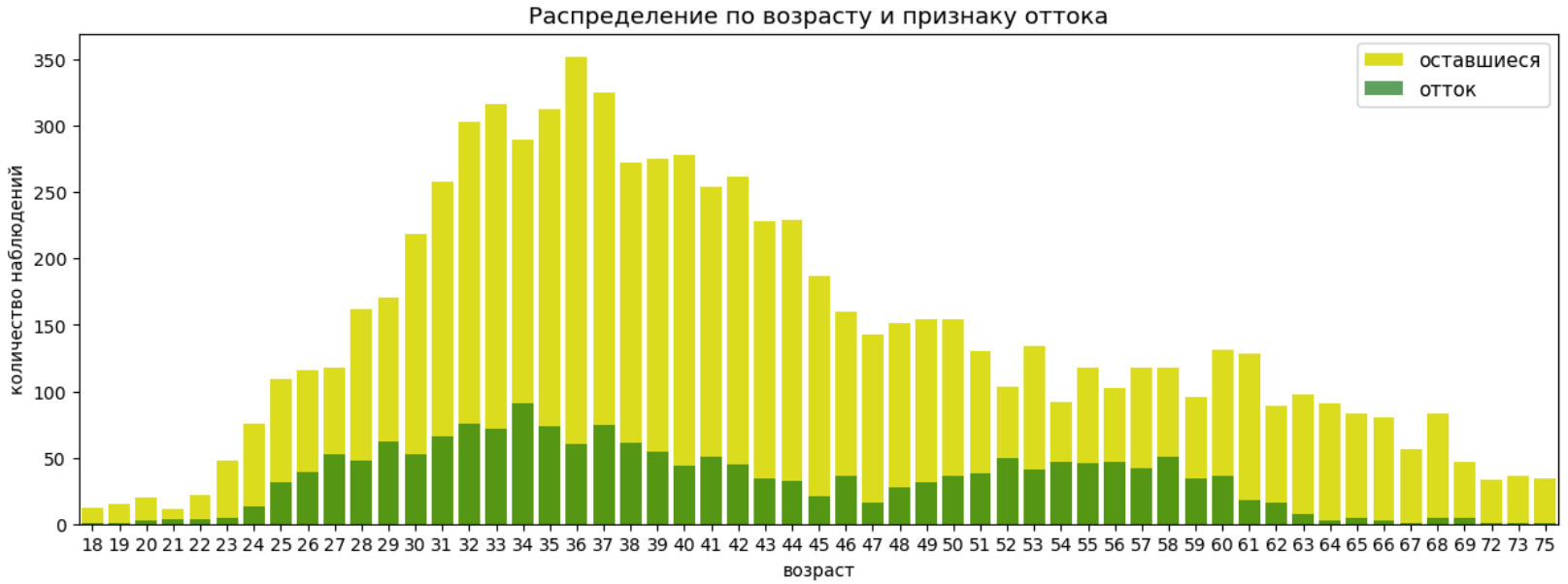
In [59]:
```
# визуализация общего распределения клиентов по возрасту
plt.subplots(figsize = (15,5))
sns.countplot(x=df.age, color='LawnGreen')
sns.set_context(rc={'font.size':10})
plt.xlabel('возраст')
plt.ylabel('количество наблюдений')
plt.title('Общее распределение по возрасту');
```



In [60]:
```
# общая статистика для возраста по группам
stat_group('возраст',
           churn_0.age, 'оставшиеся',
           churn_1.age, 'отток')
```

	возраст	оставшиеся	отток
0	count	8,104.00	1,818.00
1	mean	43.02	41.43
2	std	12.37	11.14
3	min	18.00	18.00
4	25%	34.00	32.00
5	50%	40.00	39.00
6	75%	51.00	52.00
7	max	86.00	75.00

In [61]:
```
# график частоты встречаемости каждого возраста по группам
countplot_group(churn_0['age'], 'оставшиеся',
                churn_1['age'], 'отток',
                'возраст',
                'количество наблюдений',
                'Распределение по возрасту и признаку оттока')
```



Для группы оставшихся клиентов мы видим распределение по возрасту близкое к нормальному, но с заметной асимметрией вправо.

А в группе отточных проявилось бимодальное распределение, разделившее выборку на условно "молодых" и "пожилых".

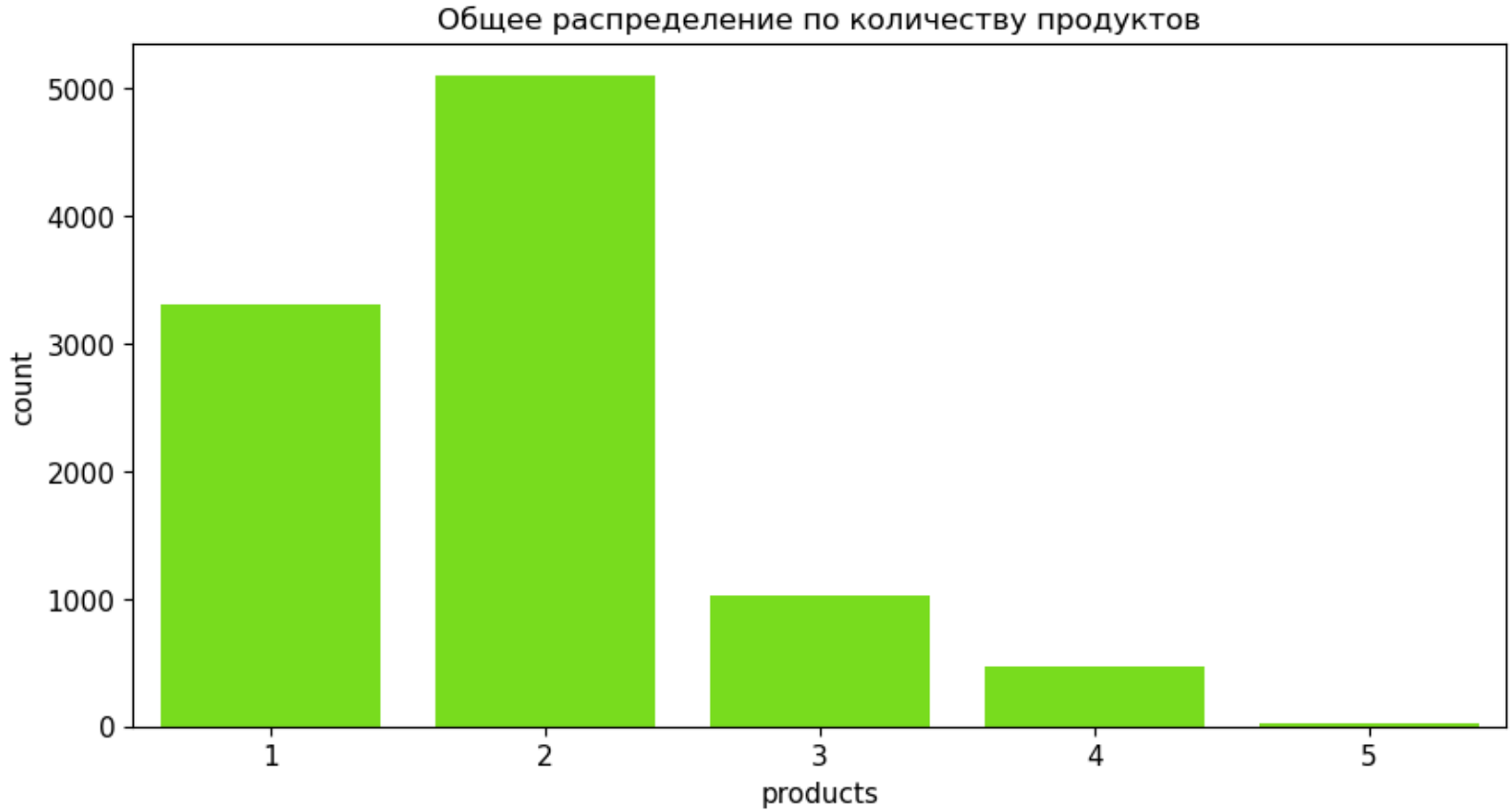
Среди "молодых" из группы оттока пик приходится на 34 года, среди "пожилых" два пика - 52 и 58 лет.

In [62]:
```
# квантили по возрасту
df.age.quantile([.25, .50, .75, .95])
```

Out[62]:
```
0.25   33.00
0.50   40.00
0.75   51.00
0.95   65.00
Name: age, dtype: float64
```

Количество продуктов

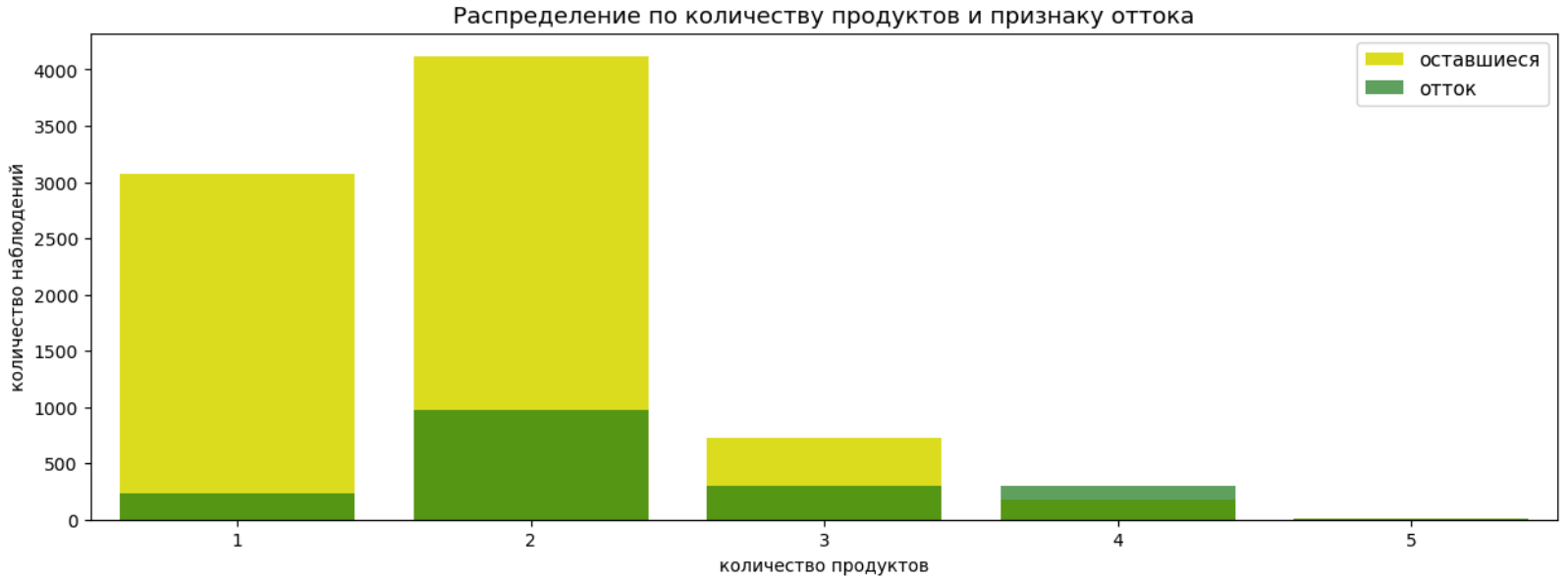
In [63]:
```
# визуализация общего распределения клиентов по количеству используемых продуктов
plt.subplots(figsize = (10,5))
sns.countplot(x=df.products, color='LawnGreen')
sns.set_context(rc={'font.size':10})
plt.title('Общее распределение по количеству продуктов');
```



In [64]:
```
# общая статистика по количеству продуктов и признаку оттока
stat_group('количество_продуктов',
           churn_0.products, 'оставшиеся',
           churn_1.products, 'отток')
```

	количество_продуктов	оставшиеся	отток
0	count	8,104.00	1,818.00
1	mean	1.76	2.38
2	std	0.71	0.92
3	min	1.00	1.00
4	25%	1.00	2.00
5	50%	2.00	2.00
6	75%	2.00	3.00
7	max	5.00	5.00

In [65]:
```
# распределение количества продуктов по признаку оттока
countplot_group(churn_0['products'], 'оставшиеся',
                churn_1['products'], 'отток',
                'количество продуктов',
                'количество наблюдений',
                'Распределение по количеству продуктов и признаку оттока')
```



Распределение среди оставшихся совпадает с общим распределением - большинство клиентов пользуются двумя продуктами.

Для отточных клиентов распределение отличается тем, что группа с тремя продуктами больше группы с одним, а группа с четырьмя продуктами равна группе с тремя и это единственная группа, в которой количество отточных клиентов преобладает над оставшимися.

Баллы собственности

In [66]:
```
# общая статистика по баллам собственности и признаку оттока
stat_group('баллы собственности',
           churn_0.equity, 'оставшиеся',
           churn_1.equity, 'отток')
```

	баллы собственности	оставшиеся	отток
0	count	8,104.00	1,818.00
1	mean	2.38	3.76
2	std	1.97	1.59
3	min	0.00	0.00
4	25%	0.00	3.00
5	50%	3.00	4.00
6	75%	4.00	5.00
7	max	9.00	9.00

In [67]:
```
# распределение по баллам собственности и признаку оттока
countplot_group(churn_0['equity'], 'оставшиеся',
                churn_1['equity'], 'отток',
                'баллы собственности',
                'количество наблюдений',
                'Распределение по баллам собственности и признаку оттока')
```

Большинство ушедших клиентов имели 4-5 баллов собственности.

Среди оставшихся больше тех, у кого 0 баллов.

Кредитная карта

In [68]:
```
# распределение по наличию кредитной карты и признаку оттока
countplot_group(churn_0['credit_card'], 'оставшиеся',
                churn_1['credit_card'], 'отток',
                'кредитная карта: нет(0), есть(1)',
                'количество наблюдений',
                'Распределение по кредитной карте и признаку оттока')
```

Среди отточных клиентов держателей кредитной карты и тех, у кого её нет, примерно поровну.

Среди оставшихся - без карты вполовину меньше.

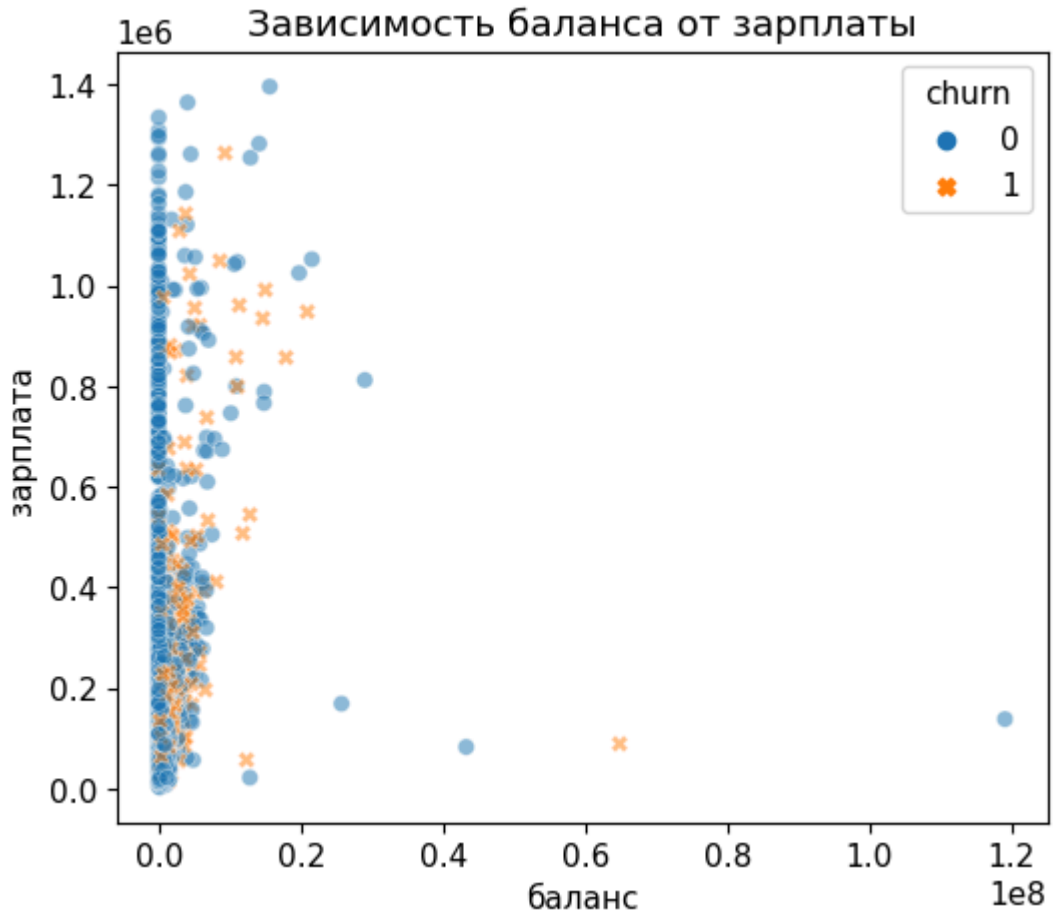
Активность за последние 30 дней

In [69]:
```
# распределение по активности и признаку оттока
countplot_group(churn_0['last_activity'], 'оставшиеся',
                churn_1['last_activity'], 'отток',
                'активность: не было(0), была(1)',
                'количество наблюдений',
                'Распределение по активности и признаку оттока')
```

Большинство отточных клиентов проявляли активность за последние 30 дней.

Баланс и зарплата

In [70]:
```
# корреляция между балансом и зарплатой
plt.subplots(figsize = (6,5))
sns.scatterplot(x='balance',
                y='est_salary',
                data=df,
                hue='churn',
                style='churn',
                alpha = 0.5)
plt.xlabel('баланс')
plt.ylabel('зарплата')
plt.title('Зависимость баланса от зарплаты');
```

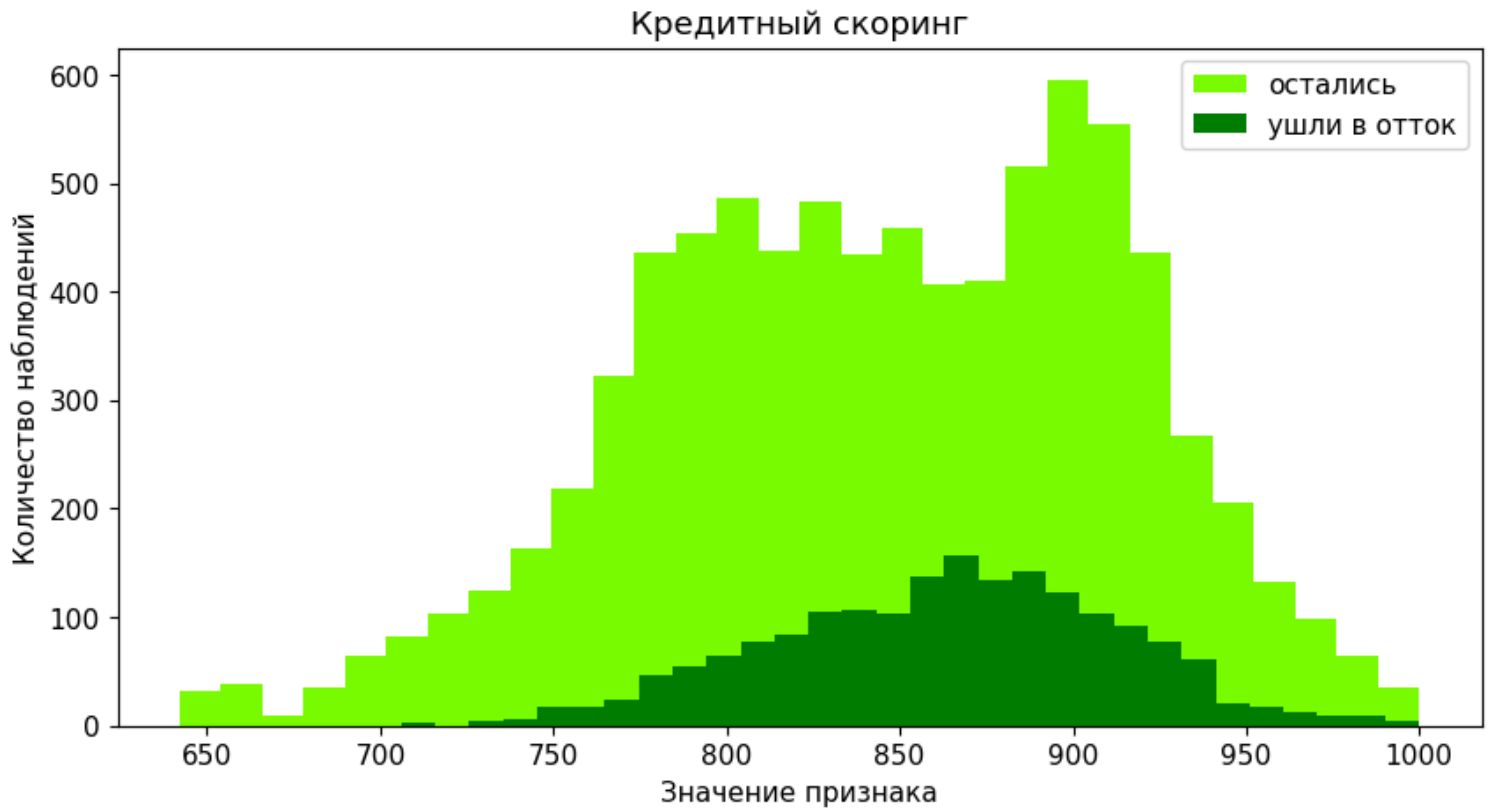


Практически при любом уровне дохода баланс может быть нулевым.
Особенно это характерно для оставшихся клиентов и в меньшей степени для отточных.
Редко, но встречается большой баланс при сравнительно небольшой зарплате.

Кредитный скоринг

In [71]:
```
# распределение кредитного скоринга
n_bins = 30
plt.subplots(figsize = (10,5))

plt.hist(churn_0.score,
         n_bins,
```

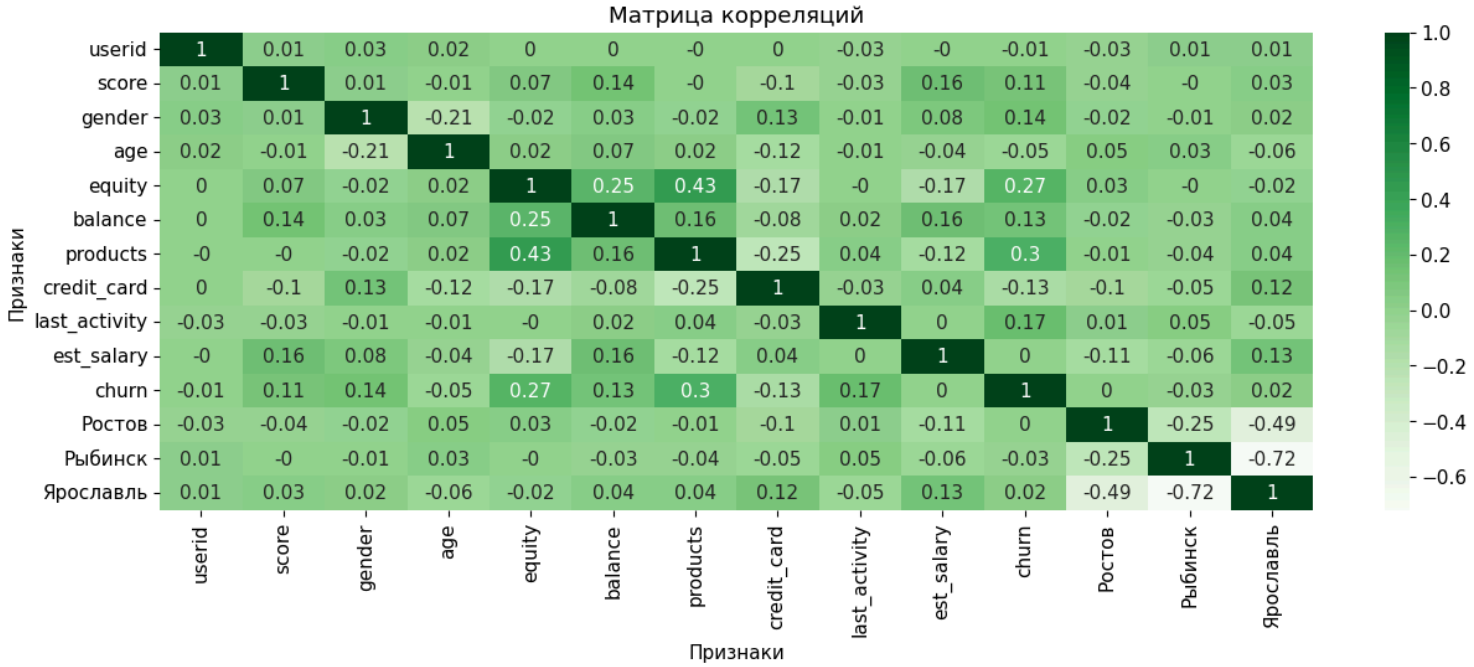
```
         color = 'LawnGreen',
         label='остались')

plt.hist(churn_1.score,
         n_bins,
         color = 'Green',
         label='ушли в отток')

plt.xlabel('Значение признака')
plt.ylabel('Количество наблюдений')
plt.title('Кредитный скоринг')
plt.legend();
```



У большинства оставшихся средний кредитный скоринг выше, чем у отточных клиентов.

Матрица корреляций

In [72]:
```
cm = df.corr().round(2)
fig, ax = plt.subplots(figsize=(15, 5))
sns.heatmap(cm, cmap="Greens", annot=True)
plt.title('Матрица корреляций')
plt.xlabel('Признаки')
plt.ylabel('Признаки');
```



С признаком оттока больше всего коррелируют количество продуктов и балл собственности.
Между этими двумя признаками также есть взаимная корреляция.
Есть небольшая зависимость между количеством продуктов и балансом.
Учитывая предыдущий анализ, для сегментации клиентов лучше выбрать балл собственности и количество продуктов.

Вывод 3

73% клиентов склонны к накоплению средств, на что указывает баланс, превышающий доходы.
Большинство клиентов из Ярославля, мужчин и женщин примерно поровну, возраст - от 18 до 86 лет.
В данных о зарплате и доходах есть выбросы, которые, скорее всего, не являются ошибкой, а просто редкие

наблюдения.

Отточных клиентов - 18% от всей выборки.
Больше всего на отток влияет количество используемых продуктов.
Среди тех, кто пользуется 4 продуктами, отточных больше, чем оставшихся.

Большинство клиентов с 4 продуктами также имеют высокие доходы, но для сегментации этот признак лучше не использовать,
так как среди зарплатных миллионеров есть достаточное число клиентов с нулевым балансом на счёте и 0 баллами собственности.
Баланс может быть нулевым практически при любом уровне дохода.

Среди использующих 4 продукта нет клиентов с нулевым балансом.
Нулевой баланс чаще всего встречается у клиентов с нулевым баллом собственности.

Часть 4. Проверка гипотез

В этой части проверим две гипотезы о том:

- различаются ли доходы между оставшимися и ушедшими в отток клиентами;
- различается ли количество продуктов, которыми пользуются оставшиеся и отточные клиенты?

Гипотеза 1

Попытаемся определить, есть ли различия в доходах между двумя группами клиентов - оставшихся и ушедших.
Для этого создадим две выборки: salary_0 и salary_1.
Сравним их между собой, используя метод для проверки гипотезы о равенстве среднего двух генеральных совокупностей.

Сформулируем гипотезы.
Нулевая гипотеза: средние значения доходов оставшихся клиентов и клиентов, ушедших в отток, равны.
Альтернативная гипотеза: среднее значение доходов оставшихся клиентов выше, чем среднее значение доходов ушедших.

In [73]:
```
# доходы оставшихся клиентов
salary_0 = (df
            .query('churn == 0')
            .est_salary)

# доходы отточных клиентов
salary_1 = (df
            .query('churn == 1')
            .est_salary)
```

In [74]:
```
# зададим уровень значимости
alpha = 0.05

# проведём ttest для двух выборок, в качестве альтернативной используем одностороннюю гипотезу «больше
# используем аргумент equal_var, так как выборки не равны по размеру
results = st.ttest_ind(salary_0,
                       salary_1,
                       alternative='greater',
                       equal_var=False)

print(f'p-value: {results.pvalue}')

# проверим p-value
if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Нет оснований отвергнуть нулевую гипотезу')
```

```
p-value: 0.5677364493234351
Нет оснований отвергнуть нулевую гипотезу
```

Мы сравниваем два среза из одного и того же набора данных.
Поэтому для проверки гипотезы используем критерий Стьюдента,
который применяется для сравнения двух выборок из одной и той же генеральной совокупности.
Конкурирующая гипотеза правосторонняя, т.е. предположение смещено в сторону бОльших значений в выборке,
поэтому используется аргумент 'greater' (больше).

Гипотеза 2

Попробуем определить, есть ли разница между средним количеством используемых продуктов в группе оставшихся и группе ушедших клиентов.
Для этого сравним две выборки: среднее количество продуктов для оставшихся (products_0) и для ушедших (products_1) по методу Стьюдента.

Нулевая гипотеза:
Среднее количество используемых банковских продуктов в группе оставшихся и в группе ушедших клиентов одинаково.
Альтернативная гипотеза:
Среднее количество продуктов в группе оставшихся меньше, чем в группе ушедших.

In [75]:
```
# количество продуктов для группы оставшихся
products_0 = (df
              .query('churn == 0')
              .products)

# количество продуктов для группы ушедших
products_1 = (df
              .query('churn == 1')
              .products)
```

In [76]:
```
# зададим уровень значимости
alpha = 0.05

# проведём ttest для двух выборок, в качестве альтернативной используем одностороннюю гипотезу «мень
# используем аргумент equal_var, так как выборки не равны по размеру
results = st.ttest_ind(products_0,
                       products_1,
                       alternative='less',
                       equal_var=False)

print(f'p-value: {results.pvalue}')

# проверим p-value
if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Нет оснований отвергнуть нулевую гипотезу')
```

```
p-value: 4.093368272545767e-139
Отвергаем нулевую гипотезу
```

Для проверки второй гипотезы мы также сравниваем две выборки из одной генеральной совокупности
и поэтому также используем критерий Стьюдента.
Альтернативная гипотеза в этот раз левосторонняя, т.е. предположение смещено в сторону меньших значений в выборке,
поэтому используется аргумент 'less' (меньше).

Результат проверки гипотез говорит нам о том, что с вероятностью 95% можно предположить:

- размер зарплаты слабо влияет на признак оттока, этот критерий не стоит использовать для сегментации клиентов;
- количество используемых продуктов, вероятно, оказывает значимое влияние на отток, можно использовать для сегментации.

Вывод 4

Для проверки гипотез мы использовали метод проверки равенства среднего двух генеральных совокупностей.
В первом случае мы не смогли отвергнуть нулевую гипотезу и оставили версию, что доходы оставшихся клиентов и ушедших в отток примерно одинаковы.
Во втором случае нулевую гипотезу можно отвергнуть и согласиться с тем, что среднее количество используемых продуктов в группе оставшихся меньше, чем в группе ушедших.

Часть 5. Сегментация клиентов

Учитывая результаты исследования, разделим выборку на 4 сегмента
по количеству баллов собственности и числу используемых банковских продуктов.
Дадим метафорические названия каждому сегменту по аналогии с водителями различных транспортных средств:

- `сегмент №1` - до 5 баллов собственности и количество продуктов 1-2 - `bicyclist` (велосипедист);
- `сегмент №2` - до 5 баллов собственности и количество продуктов 3-5 - `biker` (байкер);

- `сегмент №3` - от 5 баллов собственности и количество продуктов 1-2 - `motorist` (автомобилист);
- `сегмент №4` - от 5 баллов собственности и количество продуктов 3-5 - `racer` (гонщик).

In [77]:
```
# нарезка по сегментам
bicyclist = df.query('equity < 5 & products < 3')
biker = df.query('equity < 5 & products > 2')
motorist = df.query('equity > 4 & products < 3')
racer = df.query('equity > 4 & products > 2')
```

Сегмент №1

In [78]:
```
# несколько строк из сегмента №1
bicyclist.head()
```

Out[78]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Ростов
0	183012	850.00	0	25	1	59,214.82	2	0	1	75,719.14	1	0
2	120722	892.00	0	30	0	0.00	1	1	1	107,683.34	0	0
7	218868	825.00	0	38	4	458,145.40	2	1	1	68,085.48	0	0
9	133130	906.00	0	67	0	0.00	1	0	1	238,055.53	0	0
10	148929	927.00	1	52	0	0.00	1	1	1	196,820.07	0	1

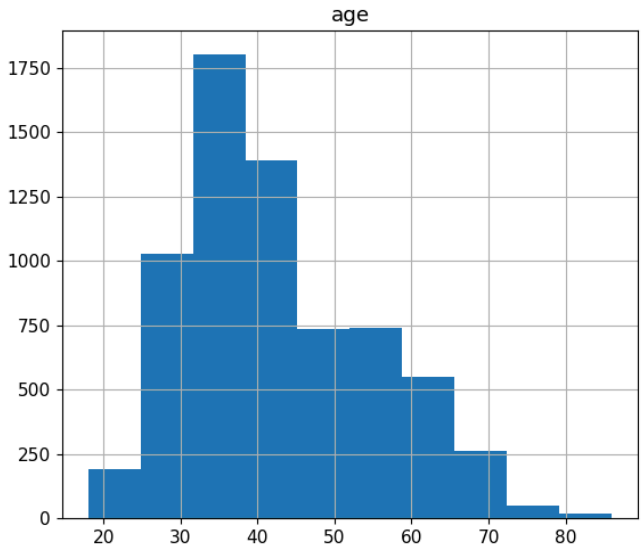
In [79]:
```
# общая статистика по сегменту №1
bicyclist.describe()
```

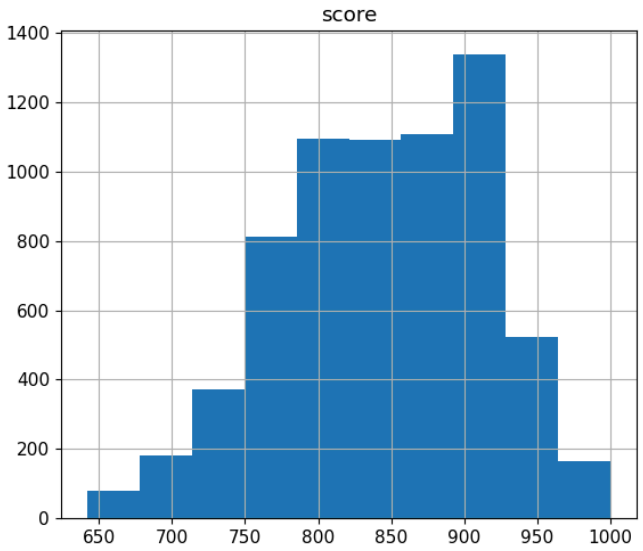
Out[79]:

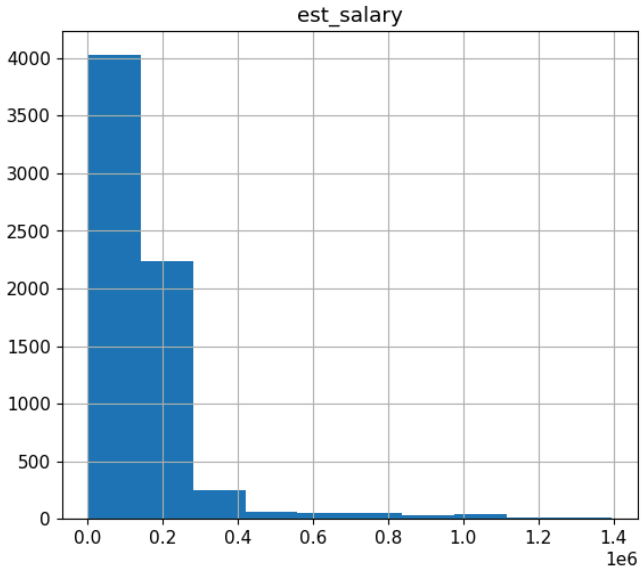
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary
count	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00	6,764.00
mean	171,681.80	844.44	0.51	42.52	1.77	419,183.73	1.56	0.74	0.52	153,187.63
std	33,481.02	69.90	0.50	12.15	1.60	1,125,977.43	0.50	0.44	0.50	145,052.51
min	94,561.00	642.00	0.00	18.00	0.00	0.00	1.00	0.00	0.00	2,546.30
25%	142,638.50	794.00	0.00	33.00	0.00	0.00	1.00	0.00	0.00	77,808.73
50%	172,461.50	848.00	1.00	40.00	2.00	234,594.64	2.00	1.00	1.00	123,054.14
75%	201,179.50	900.00	1.00	51.00	3.00	499,112.05	2.00	1.00	1.00	178,956.75
max	229,145.00	1,000.00	1.00	86.00	4.00	64,866,210.15	2.00	1.00	1.00	1,395,064.45

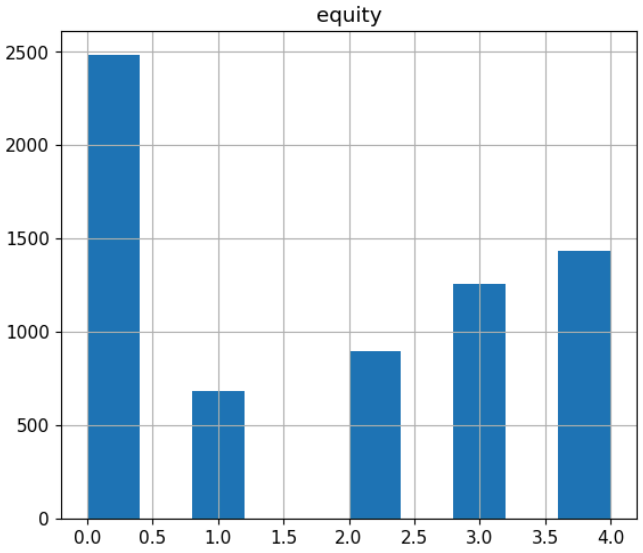
Общий признак этого сегмента - не более 4 баллов собственности и 1-2 банковских продукта.
Женщин и мужчин, а также активных и неактивных клиентов примерно поровну.
Кредитной картой владеют 74%. Средний баланс - 419 тысяч.
Средний возраст 42.5 года.
Ушедших в отток - 12%.

In [80]:
```
# визуализация статистики для "велосипедистов"
(bicyclist[['age', 'score', 'est_salary',
            'equity', 'credit_card', 'last_activity']]
 .hist(figsize=(15, 20)));
```


age
1750
1500
1250
1000
750
500
250
0
20
30
40
50
60
70
80

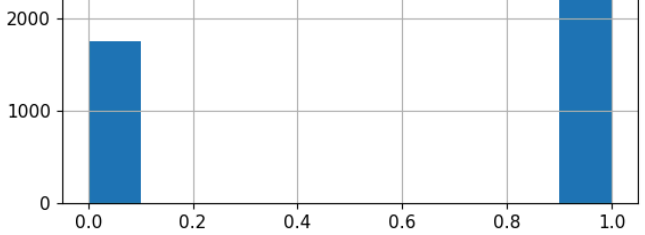

score
1400
1200
1000
800
600
400
200
0
650
700
750
800
850
900
950
1000

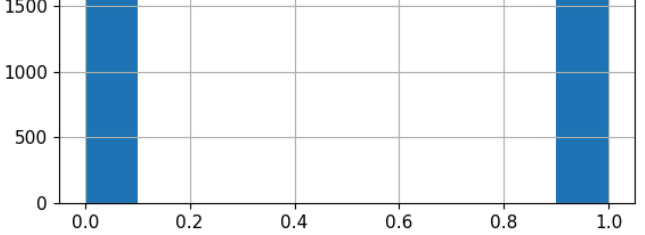

est_salary
4000
3500
3000
2500
2000
1500
1000
500
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6


equity
2500
2000
1500
1000
500
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


credit_card
5000
4000
3000


last_activity
3500
3000
2500
2000





Сегмент №2

In [81]:
```
# несколько строк из сегмента №2
biker.head()
```

Out[81]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Росто
5	202305	856.00	1	56	4	863,687.24	3	1	0	156,619.80	0	
6	177259	807.00	0	39	3	405,042.44	3	0	1	103,838.32	0	
15	120260	731.00	1	42	3	1,480,548.47	3	1	0	160,974.43	0	
26	174396	898.00	0	62	3	364,049.27	3	0	1	50,661.84	0	
33	125478	786.00	0	27	3	448,062.52	3	1	1	37,607.67	0	

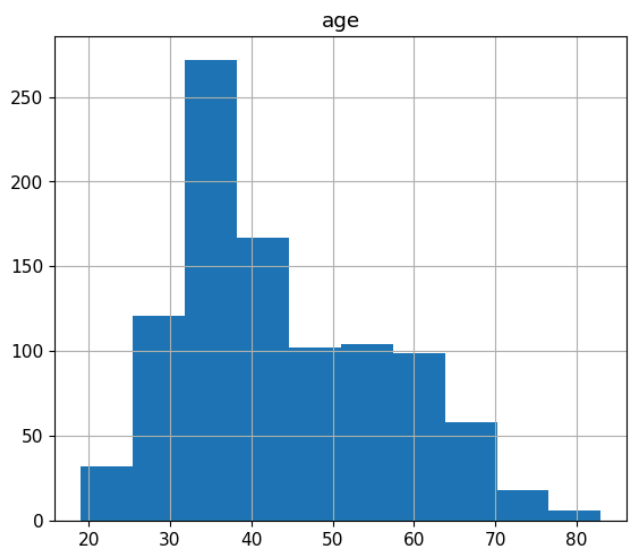
In [82]:
```
# общая статистика по сегменту №2
biker.describe()
```

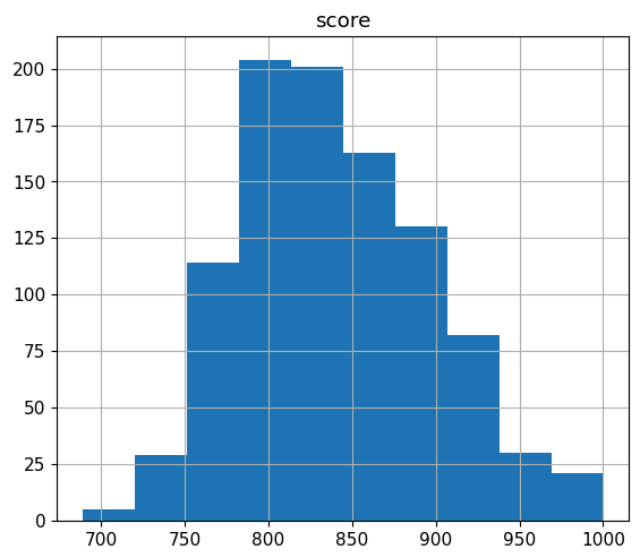
Out[82]:

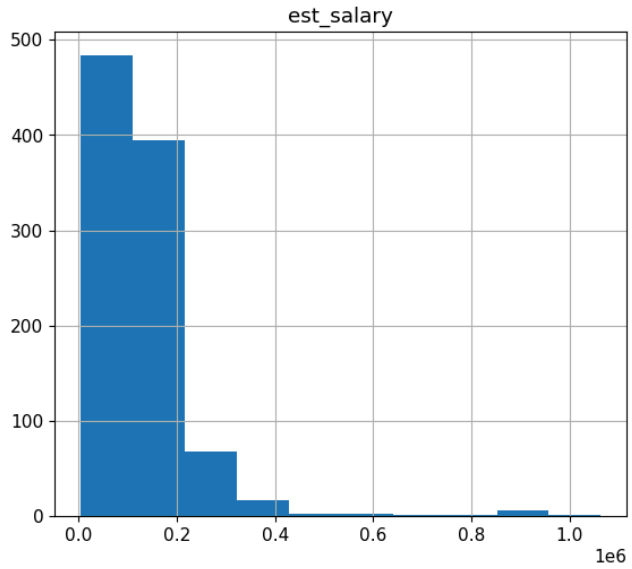
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary
count	979.00	979.00	979.00	979.00	979.00	979.00	979.00	979.00	979.00	979.00
mean	170,870.12	840.13	0.44	43.75	2.88	890,367.22	3.29	0.46	0.52	128,521.18
std	33,195.88	56.58	0.50	12.70	1.24	1,288,687.32	0.48	0.50	0.50	106,215.61
min	96,267.00	689.00	0.00	19.00	0.00	0.00	3.00	0.00	0.00	3,487.33
25%	142,585.50	796.00	0.00	34.00	2.00	297,991.77	3.00	0.00	0.00	68,510.83
50%	172,343.00	834.00	0.00	40.00	3.00	545,495.01	3.00	0.00	1.00	112,035.03
75%	197,529.50	880.00	1.00	53.00	4.00	1,201,318.29	4.00	1.00	1.00	160,476.95
max	229,058.00	1,000.00	1.00	83.00	4.00	25,727,761.86	5.00	1.00	1.00	1,064,019.29

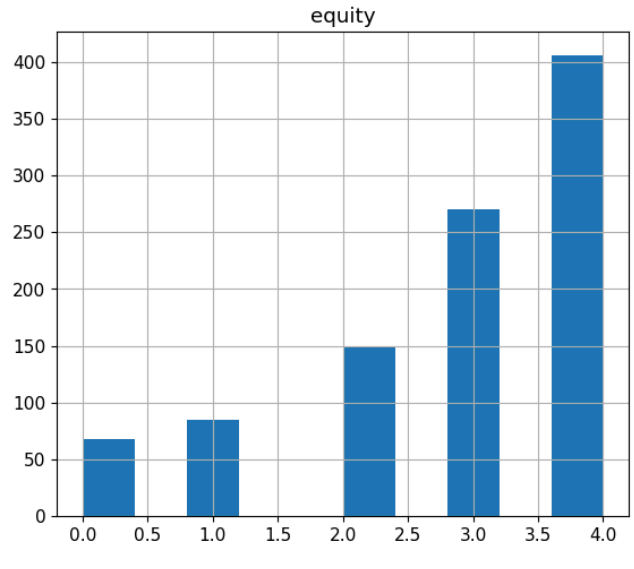
Общий признак этого сегмента - не более 4 баллов собственности и более 2 банковских продукта.
Женщин немного больше, чем мужчин, активных и неактивных клиентов примерно поровну.
Кредитной картой владеют 46%. Средний баланс - 890 тысяч.
Средний возраст 43-44 года.
Ушедших в отток - 35%.

In [83]:
```
# визуализация статистики для "байкеров"
(biker[['age', 'score', 'est_salary',
           'equity', 'credit_card', 'last_activity']]
 .hist(figsize=(15, 20)));
```

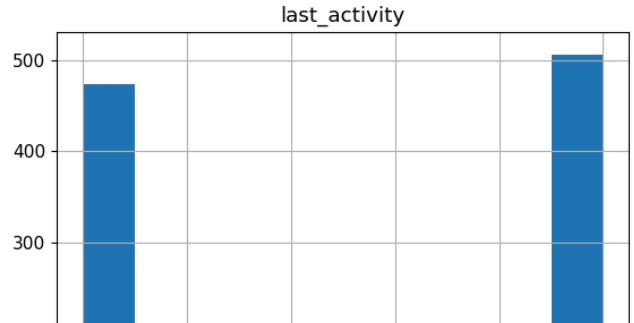

age
250
200
150
100
50
0
20
30
40
50
60
70
80

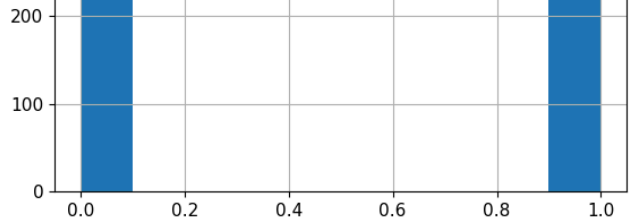

score
200
175
150
125
100
75
50
25
0
700
750
800
850
900
950
1000

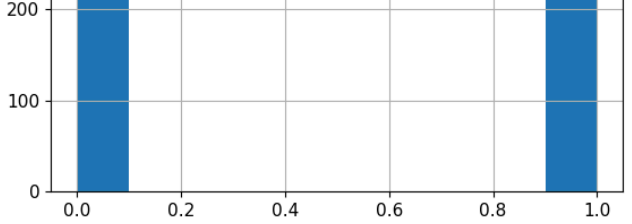

est_salary
500
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6


equity
400
350
300
250
200
150
100
50
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


credit_card
500
400
300


last_activity
500
400
300





Сегмент №3

In [84]:
```
# несколько строк из сегмента №3
motorist.head()
```

Out[84]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	Рост
3	225363	866.00	0	51	5	1,524,746.26	2	0	1	174,423.53	1	
4	157978	730.00	1	34	5	174.00	1	1	0	67,353.16	1	
8	211686	923.00	1	54	5	1,206,337.87	2	1	0	155,371.79	0	
14	172138	815.00	1	35	5	547,499.87	2	1	1	105,883.26	0	
16	123335	829.00	1	45	5	507,842.84	1	1	1	169,330.64	0	

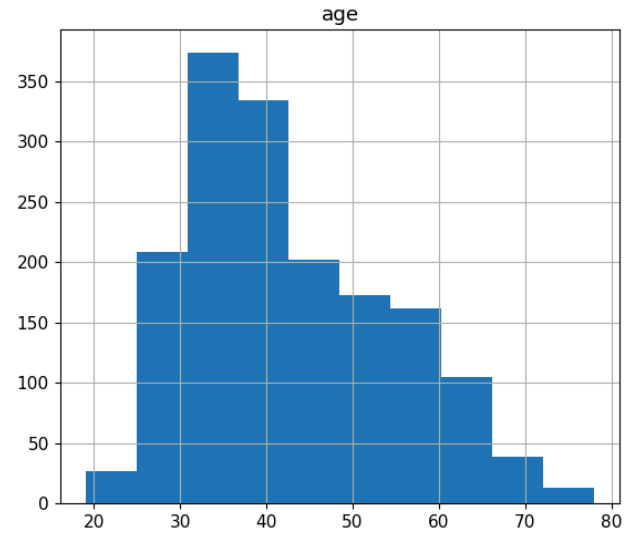
In [85]:
```
# общая статистика по сегменту №3
motorist.describe()
```

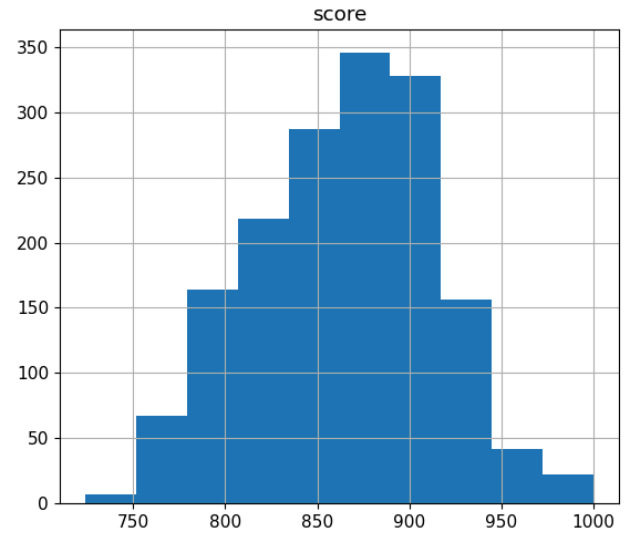
Out[85]:

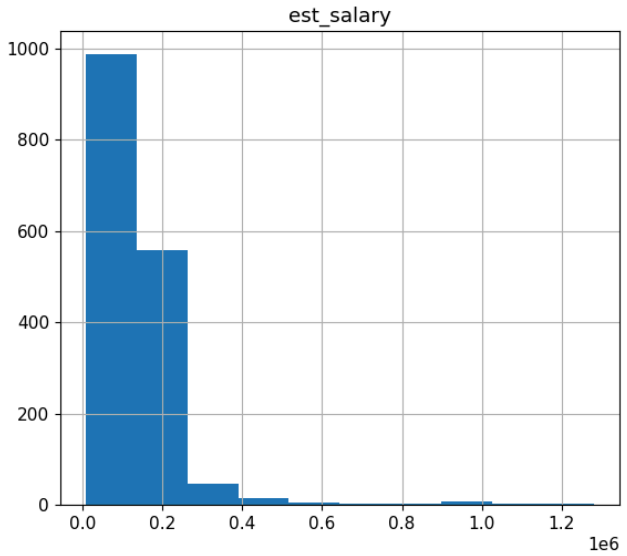
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_sa
count	1,637.00	1,637.00	1,637.00	1,637.00	1,637.00	1,637.00	1,637.00	1,637.00	1,637.00	1,63
mean	171,597.23	864.94	0.49	42.59	5.18	1,161,741.33	1.80	0.63	0.52	138,07
std	34,007.75	49.14	0.50	11.85	0.57	3,282,177.73	0.40	0.48	0.50	129,05
min	96,404.00	724.00	0.00	19.00	5.00	174.00	1.00	0.00	0.00	7,57
25%	141,897.00	829.00	0.00	33.00	5.00	546,988.62	2.00	0.00	0.00	71,99
50%	171,794.00	868.00	0.00	40.00	5.00	794,822.87	2.00	1.00	1.00	114,63
75%	200,465.00	901.00	1.00	51.00	5.00	1,189,905.04	2.00	1.00	1.00	165,97
max	228,908.00	1,000.00	1.00	78.00	9.00	119,113,552.01	2.00	1.00	1.00	1,281,54

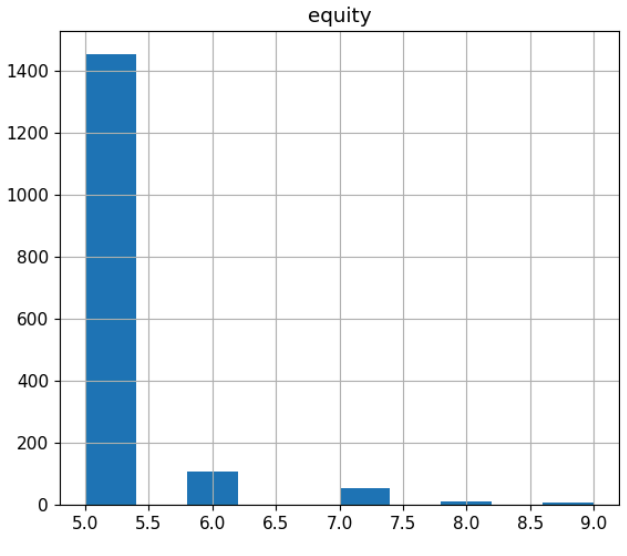
Общий признак этого сегмента - от 5 баллов собственности и 1-2 банковских продукта.
Женщин и мужчин, а также активных и неактивных клиентов примерно поровну.
Кредитной картой владеют 63%. Средний баланс - 1.1 миллиона.
Средний возраст 42.5 года.
Ушедших в отток - 26%.

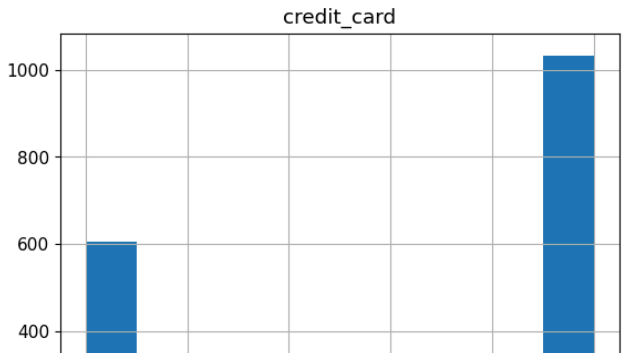
In [86]:
```
# визуализация статистики для "автомобилистов"
(motorist[['age', 'score', 'est_salary',
           'equity', 'credit_card', 'last_activity']]
 .hist(figsize=(15, 20)));
```

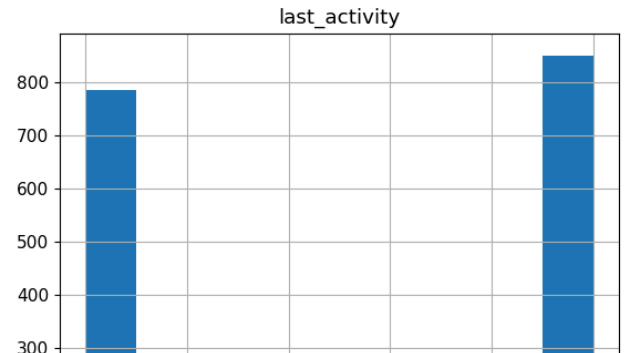

age
350
300
250
200
150
100
50
0
20
30
40
50
60
70
80


score
350
300
250
200
150
100
50
0
750
800
850
900
950
1000


est_salary
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6


equity
1400
1200
1000
800
600
400
200
0
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0


credit_card
1000
800
600
400


last_activity
800
700
600
500
400
300

Сегмент №4

In [87]:
```
# несколько строк из сегмента №4
racer.head()
```

Out[87]:

	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary	churn	P
1	146556	861.00	0	37	5	850,594.33	3	1	0	86,621.77	0	
72	156677	873.00	1	39	5	915,959.85	3	0	1	154,034.62	1	
76	213688	922.00	0	37	5	1,214,707.38	4	0	1	85,121.07	0	
81	223978	850.00	1	34	5	351,583.16	4	0	1	115,354.97	1	
148	122769	868.00	0	58	5	1,334,745.59	4	0	1	63,049.60	1	

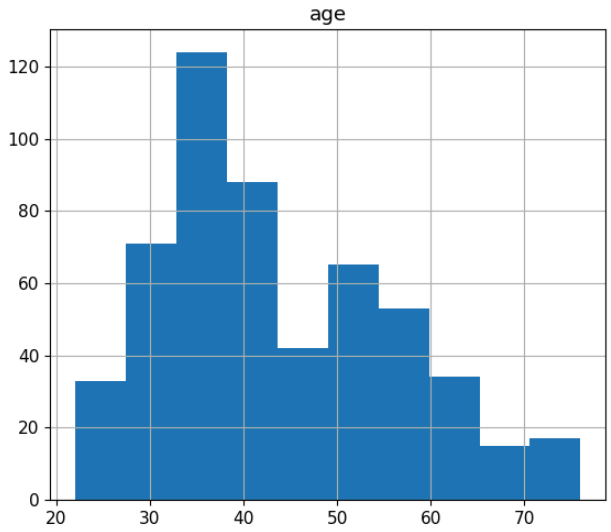
In [88]:
```
# общая статистика по сегменту №4
racer.describe()
```

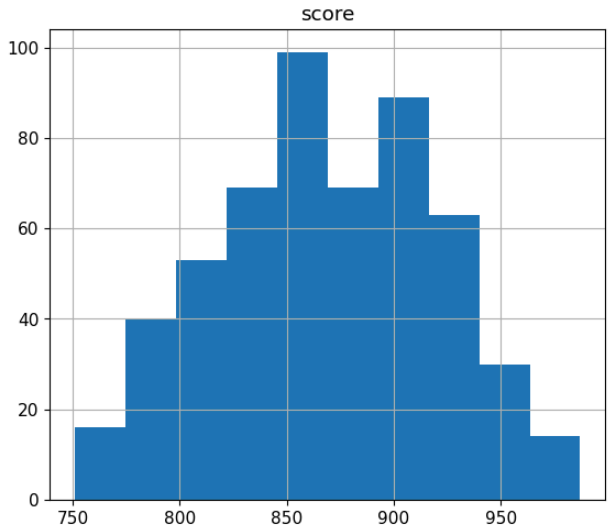
Out[88]:

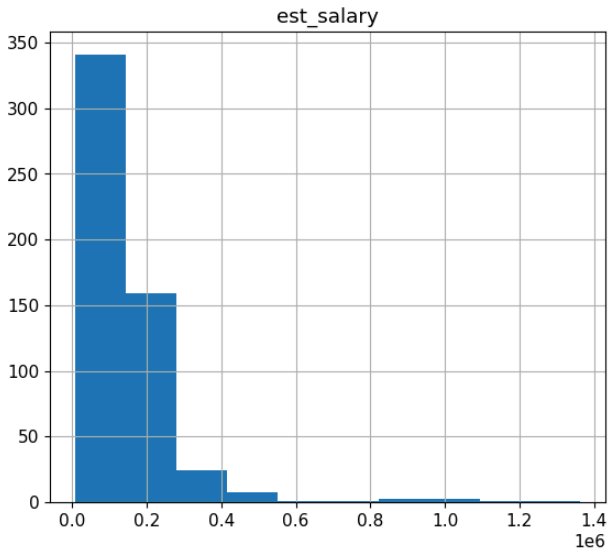
	userid	score	gender	age	equity	balance	products	credit_card	last_activity	est_salary
count	542.00	542.00	542.00	542.00	542.00	542.00	542.00	542.00	542.00	542.00
mean	174,356.90	868.67	0.59	43.81	5.25	1,361,746.89	3.43	0.46	0.58	146,563.64
std	36,188.38	51.92	0.49	12.31	0.64	2,115,969.38	0.52	0.50	0.49	145,167.77
min	95,384.00	751.00	0.00	22.00	5.00	56,925.47	3.00	0.00	0.00	8,226.26
25%	145,070.25	832.25	0.00	35.00	5.00	569,660.13	3.00	0.00	0.00	69,960.43
50%	177,111.00	868.00	1.00	41.00	5.00	986,723.99	3.00	0.00	1.00	114,027.58
75%	207,114.50	909.00	1.00	53.00	5.00	1,539,196.74	4.00	1.00	1.00	179,437.10
max	229,017.00	987.00	1.00	76.00	9.00	43,277,099.84	5.00	1.00	1.00	1,363,549.52

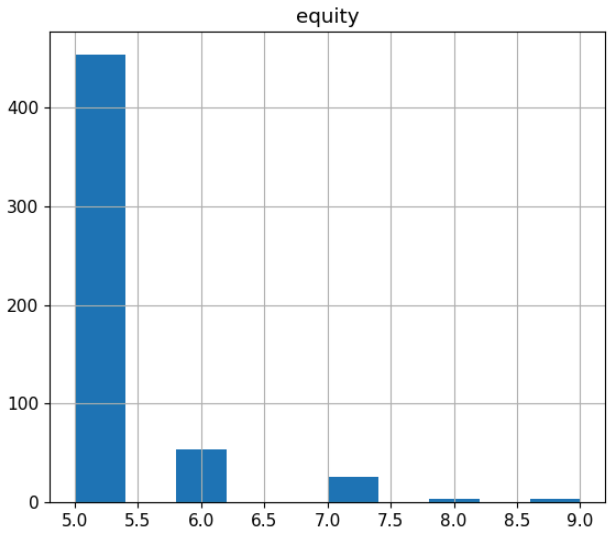
Общий признак этого сегмента - от 5 баллов собственности и более 2 банковских продукта.
Мужчин больше, чем женщин. Активных клиентов больше, чем неактивных.
Кредитной картой владеют 46%. Средний баланс - 1.3 миллиона.
Средний возраст 43-44 года.
Ушедших в отток - 49%.

In [89]:
```
# визуализация статистики для "гонщиков"
(racer[['age', 'score', 'est_salary',
           'equity', 'credit_card', 'last_activity']]
 .hist(figsize=(15, 20)));
```

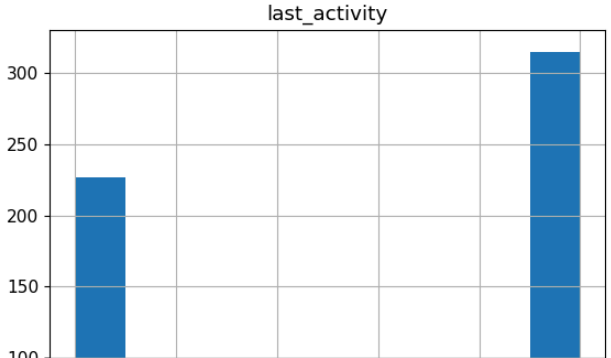

age
120
100
80
60
40
20
0
20
30
40
50
60
70

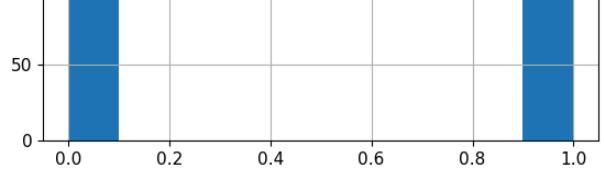

score
100
80
60
40
20
0
750
800
850
900
950

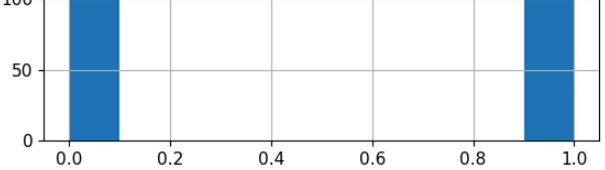

est_salary
350
300
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6


equity
400
300
200
100
0
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
9.0


credit_card
300
250
200
150
100


last_activity
300
250
200
150
100





Вывод 5

Мы разделили клиентов банка на 4 сегмента по двум признакам - балл собственности и количество продуктов.
На первое и второе место по доле отточных клиентов вышли "гонщики" и "байкеры" - 49% и 35% соответственно.
Общий признак для них - это использование от 3 до 5 банковских продуктов, а также доля владельцев кредитной карты - 46%.
Отличаются в этих группах баланс и гендерный состав: в группе "байкеров" чуть больше женщин и средний баланс 890 тысяч,
а в группе "гонщиков" больше мужчин и средний баланс 1.3 миллиона.

На третьем и четвёртом месте по отточности расположились "автомобилисты" и "велосипедисты" - 25% и 12% соответственно.
Общее для них: 1-2 банковских продукта, 63-74% держателей кредитной карты, мужчин и женщин поровну.
А вот средний баланс существенно отличается: у "велосипедистов" - 419 тыс., у "автомобилистов" - 1.1 млн.

Рекомендации

В первую очередь следует обратить внимание на сегмент №4 ("гонщики"), и сегмент №2 ("байкеры"),
в которые вошли клиенты, использующие от 3 до 5 банковских продукта.
Доля отточных в этих сегментах самая большая по сравнению с другими сегментами и составляет 49% для "гонщиков" и 35% для "байкеров".

Можно предположить, что отток в этих группах связан с неудовлетворенностью инструментами управления продуктами.
Чем больше продуктов, тем больше внимания и компетенций требуется для согласованной работы с ними.
Из этого следует, что инструменты управления продуктами, предоставляемые банком клиенту, должны быть скоординированы между собой.

В идеале лучше, чтобы это был один инструмент, позволяющий управлять различными продуктами как одной системой.
При этом быстро, удобно и понятно.

Кроме того, клиенты склонны искать более выгодные предложения
и, скорее всего, рассматривают конкурентов с такими же банковскими продуктами.
Если с одним-двумя продуктами довольно легко принять решение, выбрав более выгодные условия,
то в случае с большим количеством продуктов необходимо искать баланс, а где-то и компромисс, что уже немного более сложная задача.

Другими словами, клиентам нужен удобный и эффективный инструмент управления продуктами. Возможно, с личным ассистентом на базе ИИ, учитывающим контекст и историю клиента.

Общий вывод

Исследование показало наличие проблем с клиентами, которые пользуются 4-5 банковскими продуктами и имеют сравнительно большие суммы на счёте. Их отток может быть связан с неудовлетворённостью тем, как банк осуществляет обслуживание по нескольким продуктам одновременно. Возможно, они ищут большей согласованности между различными продуктами, сбалансированности преимуществ, которые даёт каждый продукт, и более удобный инструмент управления своей финансовой активностью.