Шаг 1. Откройте файл с данными и изучите общую информацию

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from IPython.display import display
from scipy import stats as st

data_calls = pd.read_csv('/datasets/calls.csv')
data_internet = pd.read_csv('/datasets/internet.csv')
data_messages = pd.read_csv('/datasets/messages.csv')
data_tariffs = pd.read_csv('/datasets/tariffs.csv')
data_users = pd.read_csv('/datasets/users.csv')
```

In [2]:
```python
data_calls.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202607 entries, 0 to 202606
Data columns (total 4 columns):
id           202607 non-null object
call_date    202607 non-null object
duration     202607 non-null float64
user_id      202607 non-null int64
dtypes: float64(1), int64(1), object(2)
memory usage: 6.2+ MB
```

In [3]:
```python
data_internet.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149396 entries, 0 to 149395
Data columns (total 5 columns):
Unnamed: 0      149396 non-null int64
id              149396 non-null object
mb_used         149396 non-null float64
session_date    149396 non-null object
user_id         149396 non-null int64
dtypes: float64(1), int64(2), object(2)
memory usage: 5.7+ MB
```

In [4]:
```python
data_messages.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123036 entries, 0 to 123035
Data columns (total 3 columns):
id              123036 non-null object
message_date    123036 non-null object
user_id         123036 non-null int64
dtypes: int64(1), object(2)
memory usage: 2.8+ MB
```

In [5]:
```python
data_tariffs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 8 columns):
messages_included        2 non-null int64
mb_per_month_included    2 non-null int64
minutes_included         2 non-null int64
rub_monthly_fee          2 non-null int64
rub_per_gb               2 non-null int64
rub_per_message          2 non-null int64
rub_per_minute           2 non-null int64
tariff_name              2 non-null object
dtypes: int64(7), object(1)
memory usage: 256.0+ bytes
```

In [6]:
```python
data_users.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
user_id       500 non-null int64
age           500 non-null int64
churn_date    38 non-null object
city          500 non-null object
first_name    500 non-null object
last_name     500 non-null object
reg_date      500 non-null object
tariff        500 non-null object
dtypes: int64(2), object(6)
memory usage: 31.4+ KB
```

Вывод

Данные распределены по пяти датасетам: `data_calls`, `data_internet`, `data_messages`, `data_tariffs`, `data_users`. Названия столбцов не требуют изменений.

Датасет `data_calls` состоит из 4 столбцов и 202607 строк. Столбец `call_date` необходимо привести к формату даты.

Датасет `data_internet` состоит из 5 столбцов и 149396 строк. Столбец `session_date` необходимо привести к формату даты.

Датасет `data_messages` состоит из 3 столбцов и 123036 строк. Столбец `message_date` необходимо привести к формату даты.

Датасет `data_users` состоит из 8 столбцов и 500 строк. Столбцы `churn_date` и `reg_date` необходимо привести к формату даты. Столбец `churn_date` содержит 462 нулевых значения. Это означает, что пользователи ещё пользуются тарифом.

Датасеты `data_calls`, `data_internet`, `data_messages`, `data_users` связаны по `user_id`.

Датасет `data_tariffs` состоит из 8 столбцов и 2 строк и содержит информацию о тарифах.

Шаг 2. Подготовьте данные

Приведем строки в столбце `call_date` таблицы `data_calls` к формату даты и добавим столбец месяца `call_month`.

In [7]:
```python
data_calls['call_date'] = pd.to_datetime(data_calls['call_date'], format='%Y-%m-%d')
data_calls['call_month'] = data_calls['call_date'].dt.month
```

Округлим вверх значения минут за исключением тех, чья длительность — 0.0. Для этого выберем из таблицы все значения кроме нулевых, отбросим дробную часть методом `np.floor` и прибавим единицу. Сохраним результат в столбце `duration_minute`.

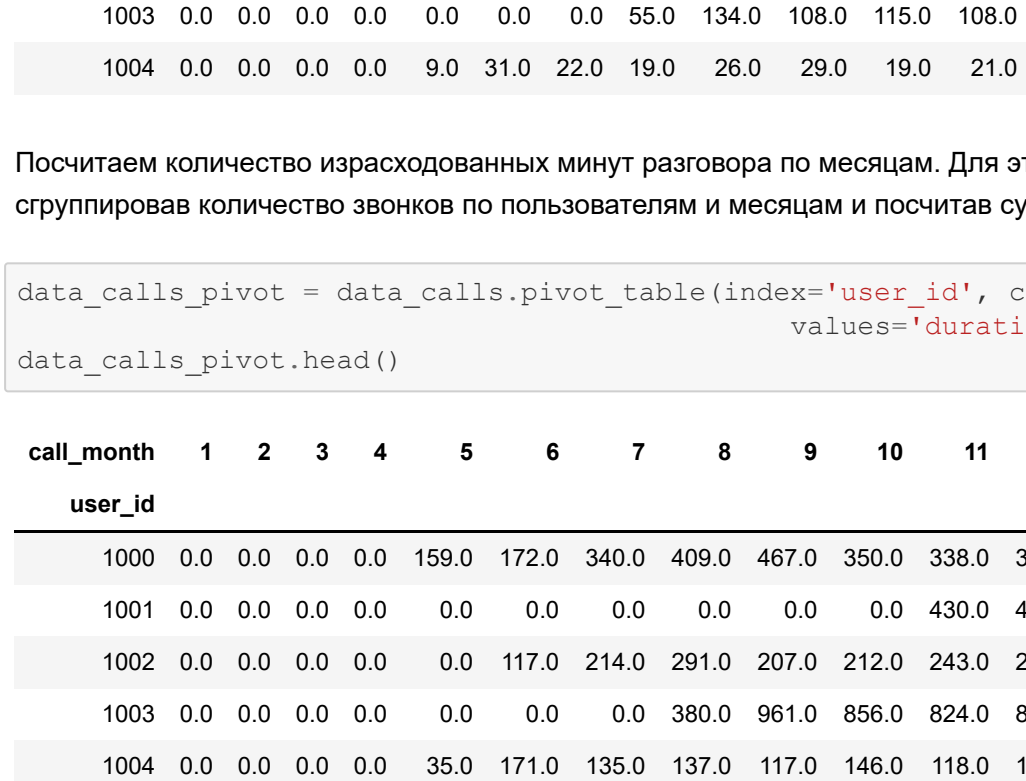
In [8]:
```python
data_calls['duration_minute'] = data_calls['duration']
data_calls.loc[data_calls['duration_minute'] != 0, 'duration_minute'] = data_calls.loc[data_calls['duration_minute'] != 0, \
                                                                        'duration_minute'].apply(np.floor) + 1
data_calls.head(10)
```

Out[8]:

	id	call_date	duration	user_id	call_month	duration_minute
0	1000_0	2018-07-25	0.00	1000	7	0.0
1	1000_1	2018-08-17	0.00	1000	8	0.0
2	1000_2	2018-06-11	2.85	1000	6	3.0
3	1000_3	2018-09-21	13.80	1000	9	14.0
4	1000_4	2018-12-15	5.18	1000	12	6.0
5	1000_5	2018-11-02	0.00	1000	11	0.0
6	1000_6	2018-10-18	0.00	1000	10	0.0
7	1000_7	2018-08-22	18.31	1000	8	19.0
8	1000_8	2018-09-15	18.44	1000	9	19.0
9	1000_9	2018-08-15	0.00	1000	8	0.0

Построим гистограмму округленной длительности звонка в минутах.

In [9]:
```python
data_calls['duration_minute'].hist(bins=len(data_calls['duration_minute'].value_counts()), figsize=(8, 5))
plt.title('Округленная длительность звонка в минутах');
```



Поскольку нулевые значения не влияют на выручку пользователей, оставим их в датасете. Максимальная длительность звонка не превышает 40 минут, что не является аномалией.

Посчитаем количество сделанных звонков по месяцам. Для этого создадим сводную таблицу `data_calls_pivot_count`, сгруппировав количество звонков по пользователям и месяцам.

In [10]:
```python
data_calls_pivot_count = data_calls.pivot_table(index='user_id', columns='call_month', \
                                                values='id', aggfunc='count').fillna(0)
data_calls_pivot_count.head()
```

Out[10]:

call_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	22.0	43.0	47.0	52.0	58.0	57.0	43.0	46.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	59.0	63.0
1002	0.0	0.0	0.0	0.0	0.0	15.0	26.0	42.0	36.0	33.0	32.0	33.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	55.0	134.0	108.0	115.0	108.0
1004	0.0	0.0	0.0	0.0	9.0	31.0	22.0	19.0	26.0	29.0	19.0	21.0

Посчитаем количество израсходованных минут разговора по месяцам. Для этого создадим сводную таблицу `data_calls_pivot`, сгруппировав количество звонков по пользователям и месяцам и посчитав суммарное количество минут разговора.

In [11]:
```python
data_calls_pivot = data_calls.pivot_table(index='user_id', columns='call_month', \
                                          values='duration_minute', aggfunc='sum').fillna(0)
data_calls_pivot.head()
```

Out[11]:

call_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	159.0	172.0	340.0	409.0	467.0	350.0	338.0	333.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	430.0	414.0
1002	0.0	0.0	0.0	0.0	0.0	117.0	214.0	291.0	207.0	212.0	243.0	236.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	380.0	961.0	856.0	824.0	802.0
1004	0.0	0.0	0.0	0.0	35.0	171.0	135.0	137.0	117.0	146.0	118.0	130.0

Приведем строки в столбце `message_date` таблицы `data_messages` к формату даты и добавим столбец месяца `message_month`.

In [12]:
```python
data_messages['message_date'] = pd.to_datetime(data_messages['message_date'], format='%Y-%m-%d')
data_messages['message_month'] = data_messages['message_date'].dt.month
```

Посчитаем количество отправленных сообщений по месяцам. Для этого создадим сводную таблицу `data_messages_pivot_sum`, сгруппировав количество сообщений по пользователям и месяцам и посчитав количество сообщений.

In [13]:
```python
data_messages_pivot = data_messages.pivot_table(index='user_id', columns='message_month', \
                                                values='id', aggfunc='count').fillna(0)
data_messages_pivot.head()
```

Out[13]:

message_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	22.0	60.0	75.0	81.0	57.0	73.0	58.0	70.0
1002	0.0	0.0	0.0	0.0	0.0	4.0	11.0	13.0	4.0	10.0	16.0	12.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	37.0	91.0	83.0	94.0	75.0
1004	0.0	0.0	0.0	0.0	95.0	134.0	181.0	151.0	146.0	165.0	158.0	162.0
1005	7.0	38.0	56.0	41.0	42.0	47.0	56.0	64.0	52.0	44.0	51.0	54.0

Приведем строки в столбце `session_date` таблицы `data_internet` к формату даты и добавим столбец месяца `session_month`.

In [14]:
```python
data_internet['session_date'] = pd.to_datetime(data_internet['session_date'], format='%Y-%m-%d')
data_internet['session_month'] = data_internet['session_date'].dt.month
```

Округлим вверх значения мегабайт за исключением тех, чей объём — 0.0. Для этого выберем из таблицы все значения кроме нулевых, отбросим дробную часть методом `np.floor` и прибавим единицу. Сохраним результат в столбце `mb_used_int`.

In [15]:
```python
data_internet['mb_used_int'] = data_internet['mb_used']
data_internet.loc[data_internet['mb_used_int'] != 0, 'mb_used_int'] = data_internet.loc[data_internet[
'mb_used_int'] != 0, \
                                                                       'mb_used_int'].apply(np.floor) + 1
data_internet
```

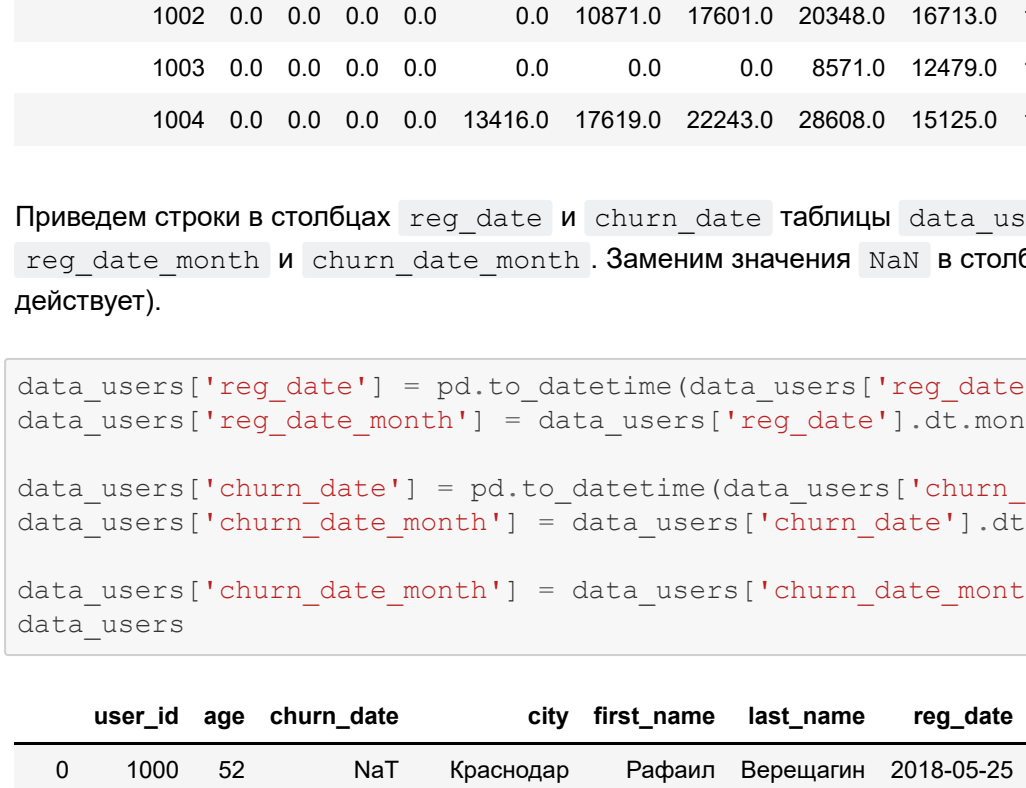
Out[15]:

	Unnamed: 0	id	mb_used	session_date	user_id	session_month	mb_used_int
0	0	1000_0	112.95	2018-11-25	1000	11	113.0
1	1	1000_1	1052.81	2018-09-07	1000	9	1053.0
2	2	1000_2	1197.26	2018-06-25	1000	6	1198.0
3	3	1000_3	550.27	2018-08-22	1000	8	551.0
4	4	1000_4	302.56	2018-09-24	1000	9	303.0
...	...	...	...	...	...	...	...
149391	149391	1499_152	318.90	2018-10-03	1499	10	319.0
149392	149392	1499_153	490.13	2018-12-14	1499	12	491.0
149393	149393	1499_154	0.00	2018-10-27	1499	10	0.0
149394	149394	1499_155	1246.32	2018-11-26	1499	11	1247.0
149395	149395	1499_156	544.37	2018-10-26	1499	10	545.0

149396 rows × 7 columns

Построим гистограмму округлённого объёма интернет-трафика (в мегабайтах).

In [16]:
```python
data_internet['mb_used_int'].hist(bins=50, figsize=(8, 5))
plt.title('Округлённый объём интернет-трафика (в мегабайтах)');
```



Поскольку нулевые значения не влияют на выручку пользователей, оставим их в датасете. Максимальный объём потраченного за сессию интернет-трафика не превышает двух гигабайт, что не является уникальным.

Посчитаем объём израсходованного интернет-трафика по месяцам. Для этого создадим сводную таблицу `data_internet_pivot_sum`, сгруппировав объём интернет-трафика по пользователям по месяцам и посчитав суммарный объём интернет-трафика.

In [17]:
```python
data_internet_pivot = data_internet.pivot_table(index='user_id', columns='session_month', \
                                                values='mb_used_int', aggfunc='sum').fillna(0)
data_internet_pivot.head()
```

Out[17]:

session_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	2256.0	23257.0	14016.0	14070.0	14581.0	14717.0	14768.0	9827.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	18452.0	14053.0
1002	0.0	0.0	0.0	0.0	0.0	10871.0	17601.0	20348.0	16713.0	13905.0	18610.0	18139.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8571.0	12479.0	14779.0	11368.0	10132.0
1004	0.0	0.0	0.0	0.0	13416.0	17619.0	22243.0	28608.0	15125.0	18494.0	15631.0	18039.0

Приведем строки в столбцах `reg_date` и `churn_date` таблицы `data_users` к формату даты и добавим столбцы месяца `reg_date_month` и `churn_date_month`. Заменим значения `NaN` в столбце `churn_date_month` на 13 (значит тариф ещё действует).

In [18]:
```python
data_users['reg_date'] = pd.to_datetime(data_users['reg_date'], format='%Y-%m-%d')
data_users['reg_date_month'] = data_users['reg_date'].dt.month

data_users['churn_date'] = pd.to_datetime(data_users['churn_date'], format='%Y-%m-%d')
data_users['churn_date_month'] = data_users['churn_date'].dt.month

data_users['churn_date_month'] = data_users['churn_date_month'].fillna(13)
data_users
```

Out[18]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month
0	1000	52	NaT	Краснодар	Рафаил	Верещагин	2018-05-25	ultra	5	13.0
1	1001	41	NaT	Москва	Иван	Ежов	2018-11-01	smart	11	13.0
2	1002	59	NaT	Стерлитамак	Евгений	Абрамович	2018-06-17	smart	6	13.0
3	1003	23	NaT	Москва	Белла	Белякова	2018-08-17	ultra	8	13.0
4	1004	68	NaT	Новокузнецк	Татьяна	Авдеенко	2018-05-14	ultra	5	13.0
...	...	...	...	...	...	...	...	...	...	...
495	1495	65	NaT	Иркутск	Авксентий	Фокин	2018-08-28	ultra	8	13.0
496	1496	36	NaT	Вологда	Трифон	Блохин	2018-01-27	smart	1	13.0
497	1497	32	NaT	Челябинск	Каролина	Агеева	2018-10-09	smart	10	13.0
498	1498	68	2018-10-25	Владикавказ	Всеволод	Акимчин	2018-07-19	smart	7	10.0
499	1499	35	NaT	Пермь	Гектор	Корнилов	2018-09-27	smart	9	13.0

500 rows × 10 columns

Создадим сводную таблицу `data_users_pivot`, где единицами будут отмечены месяцы подключения тарифа.

In [19]:
```python
data_users_pivot = data_users.pivot_table(index='user_id', columns='reg_date_month', \
                                          values='tariff', aggfunc='count').fillna(0)
data_users_pivot.head()
```

Out[19]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1002	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
1004	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Создадим сводную таблицу `data_users_zero`, заполненную нулями, она пригодится нам в дальнейшем.

In [20]:
```python
data_users_zero = data_users_pivot.replace(1, 0)
```

В цикле по каждому пользователю сравним каждый месяц с месяцем подключения тарифа и месяцем прекращения пользования тарифом. Если он внутри диапазона, меняем значение в сводной таблице на единицу.

In [21]:
```python
for i in range(data_users.shape[0]):
    for j in range(1,13):
        if (j >= data_users.loc[i, 'reg_date_month']) and (j <= data_users.loc[i, 'churn_date_month']):
            data_users_pivot.loc[1000+i, j] = 1
data_users_pivot.tail()
```

Out[21]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0
1496	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0
1498	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0
1499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0

Чтобы найти помесячную выручку с каждого пользователя, преобразуем найденные сводные таблицы и просуммируем. Однако можно обнаружить, что они не одинаковы по размеру. Некоторые пользователи не совершают звонки, другие не отправляют сообщения, третьи не пользуются интернетом. Приведём их к единому образцу, сложив с нулевой таблицей `data_users_zero` и обнулив значения `NaN`.

In [22]:
```python
data_calls_pivot = (data_users_zero + data_calls_pivot).fillna(0)
data_messages_pivot = (data_users_zero + data_messages_pivot).fillna(0)
data_internet_pivot = (data_users_zero + data_internet_pivot).fillna(0)
```

Посчитаем помесячную выручку с каждого пользователя, как если бы он пользовался обоими тарифами. Сначала найдём стоимость интернет-трафика сверх тарифного пакета по тарифу «Смарт». Для этого вычтем из таблицы `data_internet_pivot` бесплатный лимит в 15 ГБ, обнулим отрицательные значения методом `mask()`, переведём в гигабайты, поделив на 1024, выберем все значения кроме нулевых, отбросим дробную часть методом `np.floor`, прибавим единицу и умножим на стоимость 1 Гб интернет-трафика сверх тарифа. Обнулим значения `NaN` и сохраним результат в таблице `pivot_data_smart`.

In [23]:
```python
pivot_data_smart = (data_internet_pivot - 15360).mask((data_internet_pivot - 15360) < 0, 0) / 1024
pivot_data_smart = (pivot_data_smart[pivot_data_smart != 0].apply(np.floor) + 1) * 200
pivot_data_smart = pivot_data_smart.fillna(0)
pivot_data_smart.head()
```

Out[23]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	0.0	1600.0	0.0	0.0	0.0	0.0	0.0	0.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	800.0	0.0
1002	0.0	0.0	0.0	0.0	0.0	0.0	600.0	1000.0	400.0	0.0	800.0	600.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1004	0.0	0.0	0.0	0.0	0.0	600.0	1400.0	2600.0	0.0	800.0	200.0	600.0

Затем вычислим абонентскую плату, вычтем бесплатный лимит из суммарного количества звонков и сообщений, обнулим отрицательные значения методом `mask()`, умножим остаток на значение из тарифного плана «Смарт», и прибавим к уже сохранённой в таблице `pivot_data_smart` стоимости интернет-трафика сверх тарифного пакета.

In [24]:
```python
pivot_data_smart = (
    data_users_pivot * 550 +
    (data_calls_pivot - 500).mask((data_calls_pivot - 500) < 0, 0) * 3 +
    (data_messages_pivot - 50).mask((data_messages_pivot - 50) < 0, 0) * 3 +
    pivot_data_smart
)
pivot_data_smart
```

Out[24]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	550.0	2180.0	625.0	643.0	571.0	619.0	574.0	610.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1350.0	550.0
1002	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1550.0	950.0	550.0	1350.0	1150.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	2056.0	1717.0	1654.0	1531.0
1004	0.0	0.0	0.0	0.0	685.0	1402.0	2343.0	3453.0	838.0	1695.0	1074.0	1486.0
...	...	...	...	...	...	...	...	...	...	...	...	...
1495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	5194.0	5039.0	6802.0	3854.0
1496	550.0	3435.0	2721.0	1819.0	2350.0	1715.0	2596.0	2431.0	2950.0	3150.0	3039.0	1807.0
1497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	550.0	1685.0
1498	0.0	0.0	0.0	0.0	0.0	0.0	550.0	2386.0	1950.0	1750.0	0.0	0.0
1499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1513.0	598.0

500 rows × 12 columns

Сбросим индексы таблицы `pivot_data_smart` и объединим её с таблицей `data_users` методом `merge()`. Выделим пользователей тарифа «Смарт» и сохраним их в таблицу `data_users_smart`.

In [25]:
```python
data_users_smart = data_users.merge(pivot_data_smart.reset_index()).query('tariff == "smart"')
data_users_smart
```

Out[25]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	
1	1001	41	NaT	Москва	Иван	Ежов	2018-11-01	smart	11	13.0	...	0.0	
2	1002	59	NaT	Стерлитамак	Евгений	Абрамович	2018-06-17	smart	6	13.0	...	0.0	
5	1005	67	NaT	Набережные Челны	Афанасий	Горлов	2018-01-25	smart	1	13.0	...	892.0	118
6	1006	21	NaT	Ульяновск	Леонид	Ермолаев	2018-02-26	smart	2	13.0	...	950.0	55
7	1007	65	NaT	Москва	Юна	Березина	2018-04-19	smart	4	13.0	...	0.0	55
...	...	...	...	...	...	...	...	...	...	...	...	...	
494	1494	62	NaT	Омск	Ренат	Андропов	2018-08-17	smart	8	13.0	...	0.0	
496	1496	36	NaT	Вологда	Трифон	Блохин	2018-01-27	smart	1	13.0	...	2721.0	181
497	1497	32	NaT	Челябинск	Каролина	Агеева	2018-10-09	smart	10	13.0	...	0.0	
498	1498	68	2018-10-25	Владикавказ	Всеволод	Акимчин	2018-07-19	smart	7	10.0	...	0.0	
499	1499	35	NaT	Пермь	Гектор	Корнилов	2018-09-27	smart	9	13.0	...	0.0	

351 rows × 22 columns

Аналогично посчитаем помесячную выручку с каждого пользователя по тарифу «Ультра». Сначала найдём стоимость интернет-трафика сверх тарифного пакета. Для этого вычтем из таблицы `data_internet_pivot` бесплатный лимит в 30 ГБ, обнулим отрицательные значения методом `mask()`, переведём в гигабайты, поделив на 1024, выберем все значения кроме нулевых, отбросим дробную часть методом `np.floor`, прибавим единицу и умножим на стоимость 1 Гб интернет-трафика сверх тарифа. Обнулим значения `NaN` и сохраним результат в таблице `pivot_data_ultra`.

In [26]:
```python
pivot_data_ultra = (data_internet_pivot - 30720).mask((data_internet_pivot - 30720) < 0, 0) / 1024
pivot_data_ultra = (pivot_data_ultra[pivot_data_ultra != 0].apply(np.floor) + 1) * 150
pivot_data_ultra = pivot_data_ultra.fillna(0)
pivot_data_ultra.tail()
```

Out[26]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	450.0	0.0	1350.0	0.0
1496	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1498	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Затем вычислим абонентскую плату, вычтем бесплатный лимит из суммарного количества звонков и сообщений, обнулим отрицательные значения методом `mask()`, умножим остаток на значение из тарифного плана «Ультра», и прибавим к уже сохранённой в таблице `pivot_data_ultra` стоимости интернет-трафика сверх тарифного пакета.

In [27]:
```python
pivot_data_ultra = (
    data_users_pivot * 1950 +
    (data_calls_pivot - 3000).mask((data_calls_pivot - 3000) < 0, 0) +
    (data_messages_pivot - 1000).mask((data_messages_pivot - 1000) < 0, 0) +
    pivot_data_ultra
)
pivot_data_ultra
```

Out[27]:

reg_date_month user_id	1	2	3	4	5	6	7	8	9	10	11	12
1000	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0
1002	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0
1004	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
...	...	...	...	...	...	...	...	...	...	...	...	...
1495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	2400.0	1950.0	3300.0	1950.0
1496	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
1497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0
1498	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	0.0	0.0
1499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0

500 rows × 12 columns

Сбросим индексы таблицы `pivot_data_ultra` и объединим её с таблицей `data_users` методом `merge()`. Выделим пользователей тарифа «Ультра» и сохраним их в таблицу `data_users_ultra`.

In [28]:
```python
data_users_ultra = data_users.merge(pivot_data_ultra.reset_index()).query('tariff == "ultra"')
data_users_ultra
```

Out[28]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4
0	1000	52	NaT	Краснодар	Рафаил	Верещагин	2018-05-25	ultra	5	13.0	...	0.0	0.0
3	1003	23	NaT	Москва	Белла	Белякова	2018-08-17	ultra	8	13.0	...	0.0	0.0
4	1004	68	NaT	Новокузнецк	Татьяна	Авдеенко	2018-05-14	ultra	5	13.0	...	0.0	0.0
13	1013	36	NaT	Иваново	Борис	Андреенко	2018-11-14	ultra	11	13.0	...	0.0	0.0
16	1016	58	NaT	Омск	Марица	Сазонова	2018-11-21	ultra	11	13.0	...	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
476	1476	50	NaT	Москва	Валентина	Семенова	2018-04-30	ultra	4	13.0	...	0.0	1950.0
485	1485	34	NaT	Киров	Таисия	Сухова	2018-12-29	ultra	12	13.0	...	0.0	0.0
491	1491	56	NaT	Челябинск	Радислав	Андрейкин	2018-04-25	ultra	4	13.0	...	0.0	1950.0
493	1493	71	NaT	Красноярск	Таисия	Ахматулина	2018-11-24	ultra	11	13.0	...	0.0	0.0
495	1495	65	NaT	Иркутск	Авксентий	Фокин	2018-08-28	ultra	8	13.0	...	0.0	0.0

149 rows × 22 columns

Вывод

Столбцы дат в таблицах `data_calls`, `data_internet`, `data_messages` приведены к формату даты. Аномалий и ошибок в данных я не обнаружил.

Нулевые значения длительности звонков и потраченного интернет-трафика не влияют на выручку, поэтому оставлены без изменений.

Некоторые пользователи не совершают звонки, другие не отправляют сообщения, третьи не пользуются интернетом.

Шаг 3. Проанализируйте данные

Абонентская плата тарифа «Ультра», как правило, покрывает потребности пользователей, а пользователи тарифа «Смарт» часто выходят за пределы тарифного пакета.

Посчитаем количество минут разговора по месяцам для пользователей тарифа «Смарт». Для этого сбросим индексы таблицы `data_calls_pivot` и объединим её с таблицей `data_users` методом `merge()`. Выделим пользователей тарифа «Смарт» и сохраним их в таблицу `data_calls_pivot_smart`.

In [29]:
```python
data_calls_pivot_smart = data_users.merge(data_calls_pivot.reset_index()).query('tariff == "smart"')
data_calls_pivot_smart.head()
```

Out[29]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4
1	1001	41	NaT	Москва	Иван	Ежов	2018-11-01	smart	11	13.0	...	0.0	0.0
2	1002	59	NaT	Стерлитамак	Евгений	Абрамович	2018-06-17	smart	6	13.0	...	0.0	0.0
5	1005	67	NaT	Набережные Челны	Афанасий	Горлов	2018-01-25	smart	1	13.0	...	608.0	712.0
6	1006	21	NaT	Ульяновск	Леонид	Ермолаев	2018-02-26	smart	2	13.0	...	287.0	357.0
7	1007	65	NaT	Москва	Юна	Березина	2018-04-19	smart	4	13.0	...	0.0	199.0

5 rows × 22 columns

Аналогично для тарифа «Ультра».

In [30]:
```python
data_calls_pivot_ultra = data_users.merge(data_calls_pivot.reset_index()).query('tariff == "ultra"')
data_calls_pivot_ultra.head()
```

Out[30]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4	
0	1000	52	NaT	Краснодар	Рафаил	Верещагин	2018-05-25	ultra	5	13.0	...	0.0	0.0	159
3	1003	23	NaT	Москва	Белла	Белякова	2018-08-17	ultra	8	13.0	...	0.0	0.0	0
4	1004	68	NaT	Новокузнецк	Татьяна	Авдеенко	2018-05-14	ultra	5	13.0	...	0.0	0.0	35
13	1013	36	NaT	Иваново	Борис	Андреенко	2018-11-14	ultra	11	13.0	...	0.0	0.0	0
16	1016	58	NaT	Омск	Марица	Сазонова	2018-11-21	ultra	11	13.0	...	0.0	0.0	0

5 rows × 22 columns

Посчитаем количество сообщений по месяцам для пользователей тарифа «Смарт». Для этого сбросим индексы таблицы `data_messages_pivot` и объединим её с таблицей `data_users` методом `merge()`. Выделим пользователей тарифа «Смарт» и сохраним их в таблицу `data_messages_pivot_smart`.

In [31]:
```python
data_messages_pivot_smart = data_users.merge(data_messages_pivot.reset_index()).query('tariff == "smart"')
data_messages_pivot_smart.head()
```

Out[31]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4	
1	1001	41	NaT	Москва	Иван	Ежов	2018-11-01	smart	11	13.0	...	0.0	0.0	
2	1002	59	NaT	Стерлитамак	Евгений	Абрамович	2018-06-17	smart	6	13.0	...	0.0	0.0	
5	1005	67	NaT	Набережные Челны	Афанасий	Горлов	2018-01-25	smart	1	13.0	...	56.0	41.0	4
6	1006	21	NaT	Ульяновск	Леонид	Ермолаев	2018-02-26	smart	2	13.0	...	0.0	0.0	
7	1007	65	NaT	Москва	Юна	Березина	2018-04-19	smart	4	13.0	...	0.0	11.0	3

5 rows × 22 columns

Аналогично для тарифа «Ультра».

In [32]:
```python
data_messages_pivot_ultra = data_users.merge(data_messages_pivot.reset_index()).query('tariff == "ultra"')
data_messages_pivot_ultra.head()
```

Out[32]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4	5
0	1000	52	NaT	Краснодар	Рафаил	Верещагин	2018-05-25	ultra	5	13.0	...	0.0	0.0	22.
3	1003	23	NaT	Москва	Белла	Белякова	2018-08-17	ultra	8	13.0	...	0.0	0.0	0.
4	1004	68	NaT	Новокузнецк	Татьяна	Авдеенко	2018-05-14	ultra	5	13.0	...	0.0	0.0	95.
13	1013	36	NaT	Иваново	Борис	Андреенко	2018-11-14	ultra	11	13.0	...	0.0	0.0	0.
16	1016	58	NaT	Омск	Марица	Сазонова	2018-11-21	ultra	11	13.0	...	0.0	0.0	0.

5 rows × 22 columns

Посчитаем объём израсходованного интернет-трафика по месяцам для пользователей тарифа «Смарт». Для этого сбросим индексы таблицы `data_internet_pivot` и объединим её с таблицей `data_users` методом `merge()`. Выделим пользователей тарифа «Смарт» и сохраним их в таблицу `data_internet_pivot_smart`.

In [33]:
```python
data_internet_pivot_smart = data_users.merge(data_internet_pivot.reset_index()).query('tariff == "smart"')
data_internet_pivot_smart.head()
```

Out[33]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	
1	1001	41	NaT	Москва	Иван	Ежов	2018-11-01	smart	11	13.0	...	0.0	
2	1002	59	NaT	Стерлитамак	Евгений	Абрамович	2018-06-17	smart	6	13.0	...	0.0	
5	1005	67	NaT	Набережные Челны	Афанасий	Горлов	2018-01-25	smart	1	13.0	...	11117.0	936
6	1006	21	NaT	Ульяновск	Леонид	Ермолаев	2018-02-26	smart	2	13.0	...	17309.0	1395
7	1007	65	NaT	Москва	Юна	Березина	2018-04-19	smart	4	13.0	...	0.0	803

5 rows × 22 columns

Аналогично для тарифа «Ультра».

In [34]:
```python
data_internet_pivot_ultra = data_users.merge(data_internet_pivot.reset_index()).query('tariff == "ultra"')
data_internet_pivot_ultra.head()
```

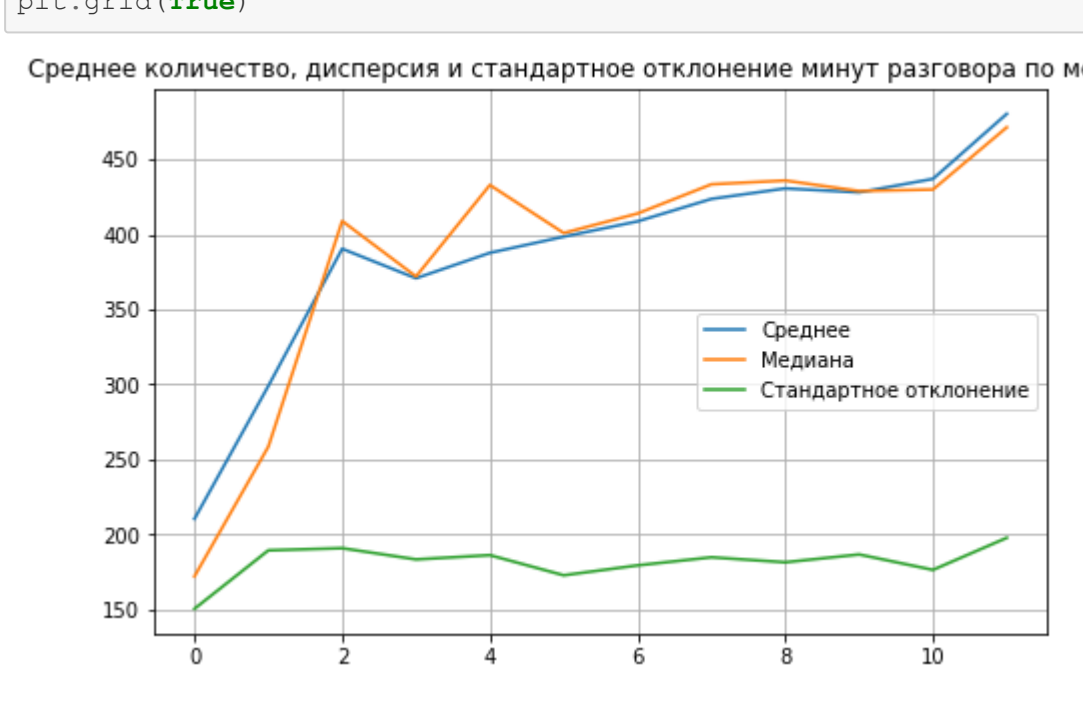
Out[34]:

	user_id	age	churn_date	city	first_name	last_name	reg_date	tariff	reg_date_month	churn_date_month	...	3	4	
0	1000	52	NaT	Краснодар	Рафаил	Верещагин	2018-05-25	ultra	5	13.0	...	0.0	0.0	22
3	1003	23	NaT	Москва	Белла	Белякова	2018-08-17	ultra	8	13.0	...	0.0	0.0	
4	1004	68	NaT	Новокузнецк	Татьяна	Авдеенко	2018-05-14	ultra	5	13.0	...	0.0	0.0	134
13	1013	36	NaT	Иваново	Борис	Андреенко	2018-11-14	ultra	11	13.0	...	0.0	0.0	
16	1016	58	NaT	Омск	Марица	Сазонова	2018-11-21	ultra	11	13.0	...	0.0	0.0	

5 rows × 22 columns

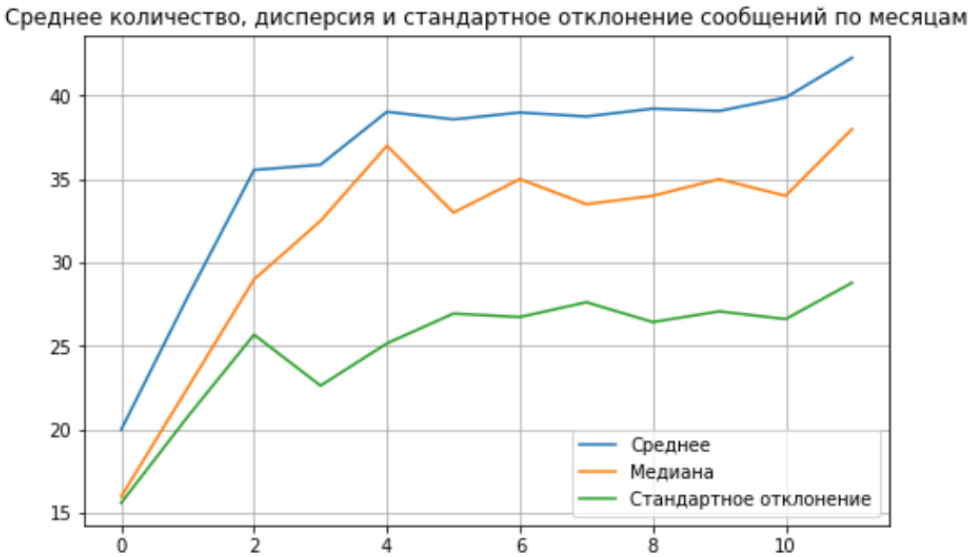
Посчитаем среднее количество, дисперсию и стандартное отклонение минут разговора по месяцам для пользователей тарифа «Смарт». Для этого создадим три пустых списка, куда в цикле запишем среднее, дисперсию и стандартное отклонение по каждому месяцу, отбросив нулевые значения. Выведем их на одном графике.

In [35]:
```python
data_calls_pivot_smart_mean = []
data_calls_pivot_smart_median = []
data_calls_pivot_smart_std = []
for i in range(1,13):
    data_calls_pivot_smart_mean.append(data_calls_pivot_smart[data_calls_pivot_smart[i] != 0][i].mean())
    data_calls_pivot_smart_median.append(data_calls_pivot_smart[data_calls_pivot_smart[i] != 0][i].median())
    data_calls_pivot_smart_std.append(np.std(data_calls_pivot_smart[data_calls_pivot_smart[i] != 0][i]))
plt.figure(figsize=(8, 5))
plt.plot(data_calls_pivot_smart_mean, label = 'Среднее')
plt.plot(data_calls_pivot_smart_median, label = 'Медиана')
plt.plot(data_calls_pivot_smart_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение минут разговора по месяцам')
plt.legend()
plt.grid(True)
```



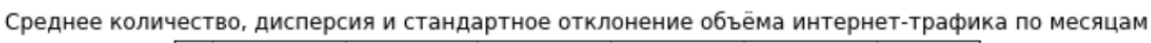
Аналогично посчитаем среднее количество, дисперсию и стандартное отклонение сообщений по месяцам для пользователей тарифа «Смарт».

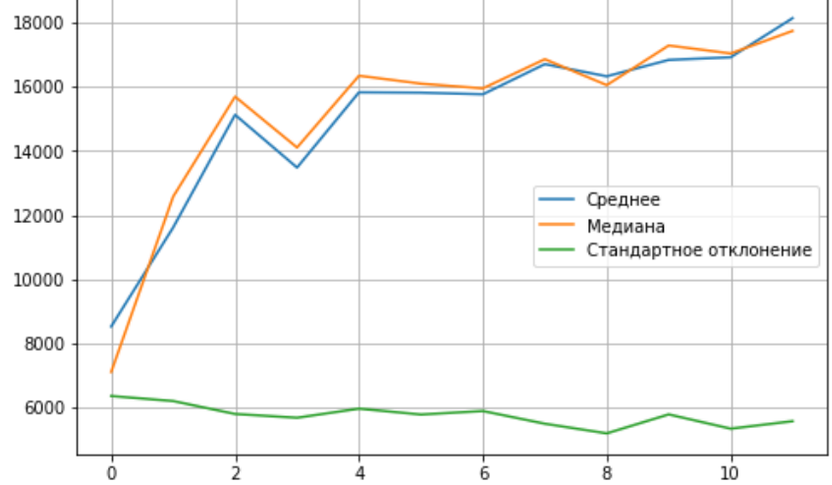
In [36]:
```
data_messages_pivot_smart_mean = []
data_messages_pivot_smart_median = []
data_messages_pivot_smart_std = []
for i in range(1,13):
    data_messages_pivot_smart_mean.append(data_messages_pivot_smart[data_messages_pivot_smart[i] != 0][
i].mean())
    data_messages_pivot_smart_median.append(data_messages_pivot_smart[data_messages_pivot_smart[i] != 0
][i].median())
    data_messages_pivot_smart_std.append(np.std(data_messages_pivot_smart[data_messages_pivot_smart[i]
!= 0][i]))
plt.figure(figsize=(8, 5))
plt.plot(data_messages_pivot_smart_mean, label = 'Среднее')
plt.plot(data_messages_pivot_smart_median, label = 'Медиана')
plt.plot(data_messages_pivot_smart_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение сообщений по месяцам')
plt.legend()
plt.grid(True)
```



Аналогично посчитаем среднее количество, дисперсию и стандартное отклонение объёма интернет-трафика по месяцам для пользователей тарифа «Смарт».

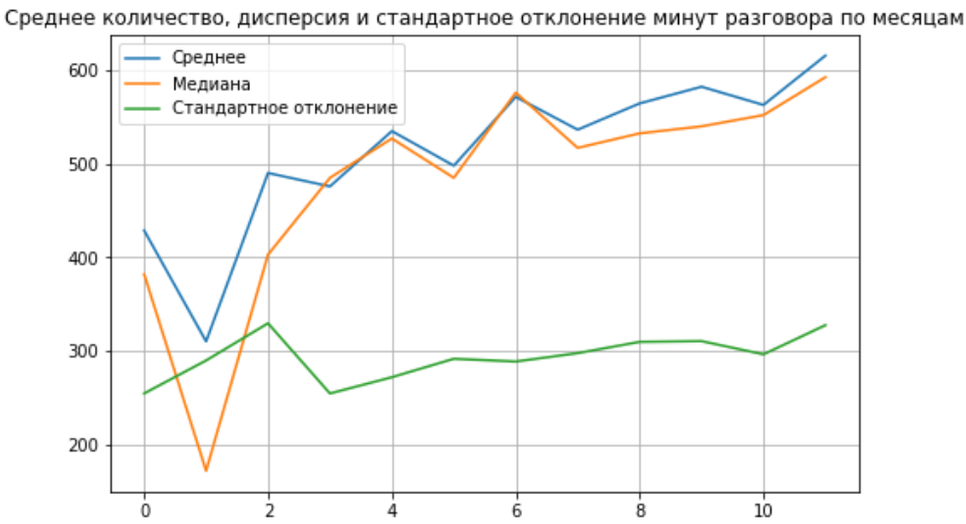
In [37]:
```
data_internet_pivot_smart_mean = []
data_internet_pivot_smart_median = []
data_internet_pivot_smart_std = []
for i in range(1,13):
    data_internet_pivot_smart_mean.append(data_internet_pivot_smart[data_internet_pivot_smart[i] != 0][
i].mean())
    data_internet_pivot_smart_median.append(data_internet_pivot_smart[data_internet_pivot_smart[i] != 0
][i].median())
    data_internet_pivot_smart_std.append(np.std(data_internet_pivot_smart[data_internet_pivot_smart[i]
!= 0][i]))
plt.figure(figsize=(8, 5))
plt.plot(data_internet_pivot_smart_mean, label = 'Среднее')
plt.plot(data_internet_pivot_smart_median, label = 'Медиана')
plt.plot(data_internet_pivot_smart_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение объёма интернет-трафика по месяцам')
plt.legend()
plt.grid(True)
```



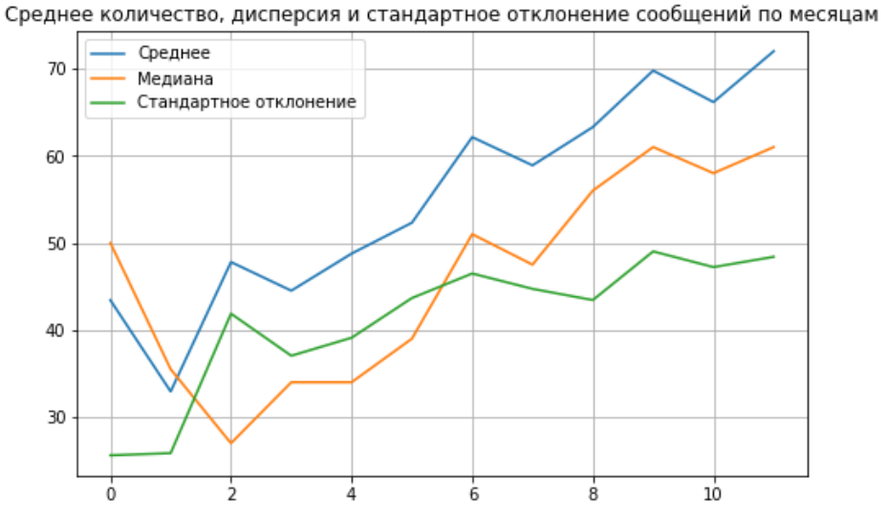


Проделаем тоже самое для пользователей тарифа «Ультра».

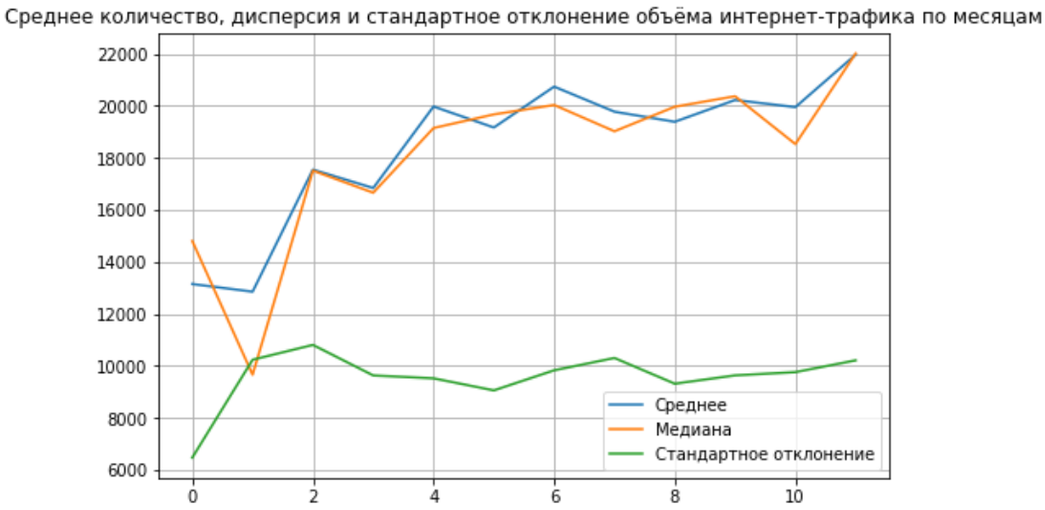
In [38]:
```
data_calls_pivot_ultra_mean = []
data_calls_pivot_ultra_median = []
data_calls_pivot_ultra_std = []
for i in range(1,13):
    data_calls_pivot_ultra_mean.append(data_calls_pivot_ultra[data_calls_pivot_ultra[i] != 0][i].mean
())
    data_calls_pivot_ultra_median.append(data_calls_pivot_ultra[data_calls_pivot_ultra[i] != 0][i].medi
an())
    data_calls_pivot_ultra_std.append(np.std(data_calls_pivot_ultra[data_calls_pivot_ultra[i] != 0][i
]))
plt.figure(figsize=(8, 5))
plt.plot(data_calls_pivot_ultra_mean, label = 'Среднее')
plt.plot(data_calls_pivot_ultra_median, label = 'Медиана')
plt.plot(data_calls_pivot_ultra_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение минут разговора по месяцам')
plt.legend()
plt.grid(True)
```



In [39]:
```
data_messages_pivot_ultra_mean = []
data_messages_pivot_ultra_median = []
data_messages_pivot_ultra_std = []
for i in range(1,13):
    data_messages_pivot_ultra_mean.append(data_messages_pivot_ultra[data_messages_pivot_ultra[i] != 0][
i].mean())
    data_messages_pivot_ultra_median.append(data_messages_pivot_ultra[data_messages_pivot_ultra[i] != 0
][i].median())
    data_messages_pivot_ultra_std.append(np.std(data_messages_pivot_ultra[data_messages_pivot_ultra[i]
!= 0][i]))
plt.figure(figsize=(8, 5))
plt.plot(data_messages_pivot_ultra_mean, label = 'Среднее')
plt.plot(data_messages_pivot_ultra_median, label = 'Медиана')
plt.plot(data_messages_pivot_ultra_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение сообщений по месяцам')
plt.legend()
plt.grid(True)
```

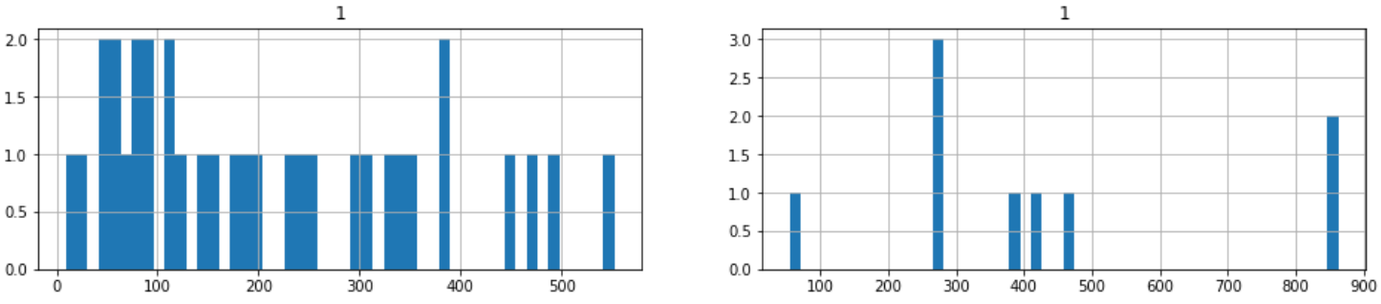


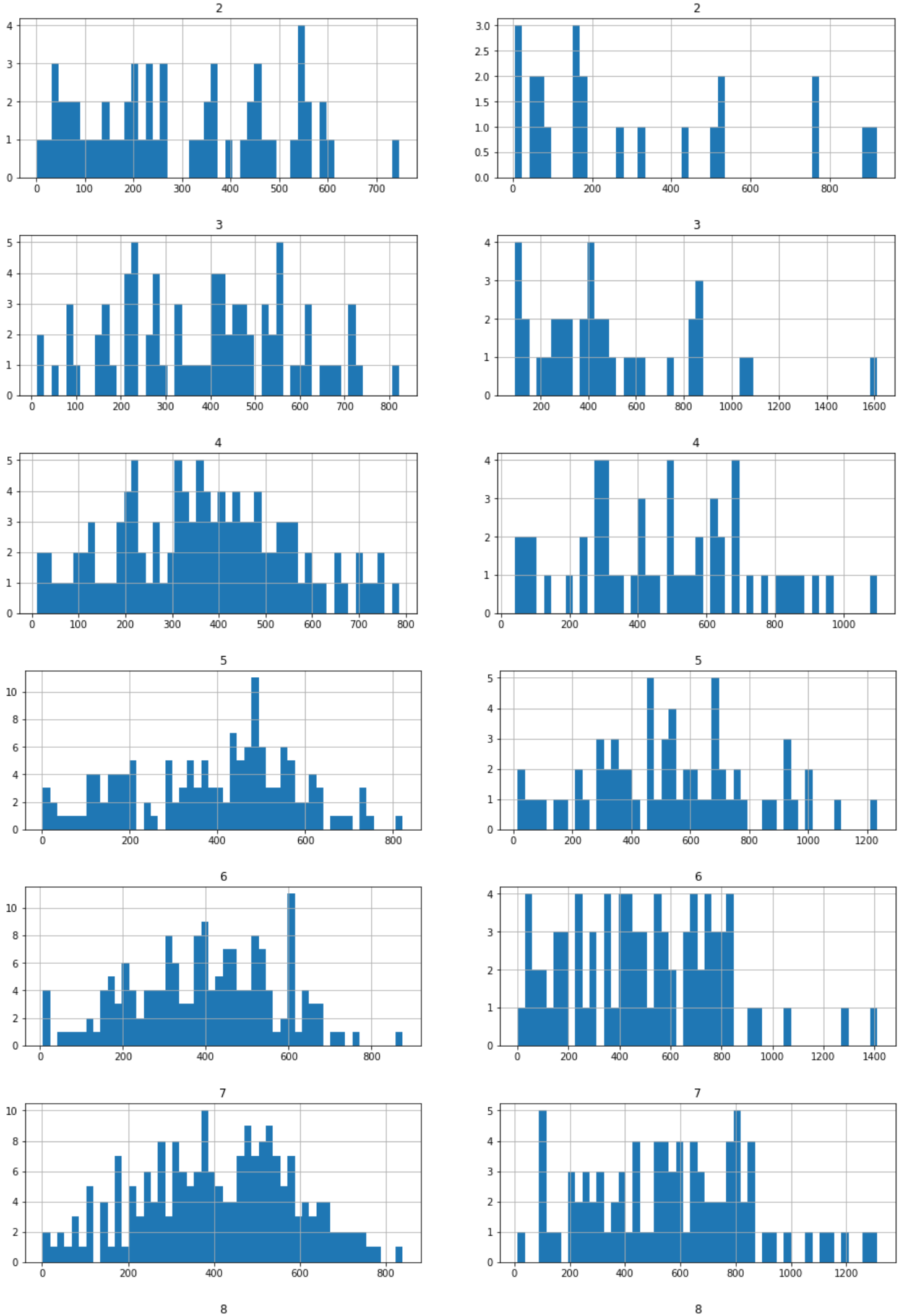
In [40]:
```
data_internet_pivot_ultra_mean = []
data_internet_pivot_ultra_median = []
data_internet_pivot_ultra_std = []
for i in range(1,13):
    data_internet_pivot_ultra_mean.append(data_internet_pivot_ultra[data_internet_pivot_ultra[i] != 0][
i].mean())
    data_internet_pivot_ultra_median.append(data_internet_pivot_ultra[data_internet_pivot_ultra[i] != 0
][i].median())
    data_internet_pivot_ultra_std.append(np.std(data_internet_pivot_ultra[data_internet_pivot_ultra[i]
!= 0][i]))
plt.figure(figsize=(8, 5))
plt.plot(data_internet_pivot_ultra_mean, label = 'Среднее')
plt.plot(data_internet_pivot_ultra_median, label = 'Медиана')
plt.plot(data_internet_pivot_ultra_std, label = 'Стандартное отклонение')
plt.title('Среднее количество, дисперсия и стандартное отклонение объёма интернет-трафика по месяцам')
plt.legend()
plt.grid(True)
```

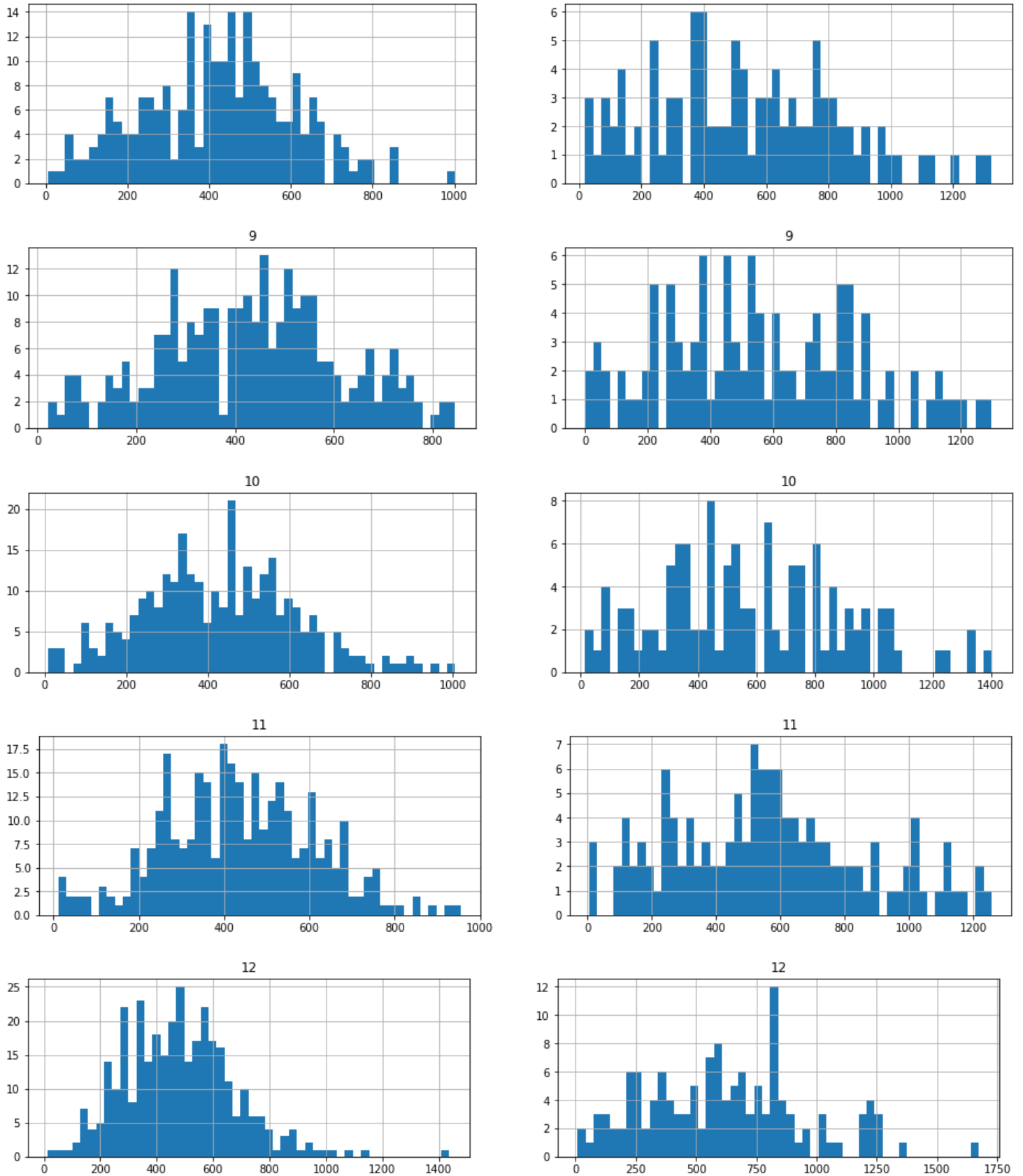


Построим гистограммы количества минут разговора по каждому месяцу для тарифов «Смарт» (слева) и «Ультра» (справа). Идеально было бы создать анимированную гистограмму для экономии места, но я не смог и время поджимало.

In [41]:
```
for i in range(1,13):
    plt.figure(figsize=(16, 3))
    plt.subplot(1, 2, 1)
    data_calls_pivot_smart[data_calls_pivot_smart[i] != 0][i].hist(bins=50)
    plt.title(i)
    plt.subplot(1, 2, 2)
    data_calls_pivot_ultra[data_calls_pivot_ultra[i] != 0][i].hist(bins=50)
    plt.title(i)
```

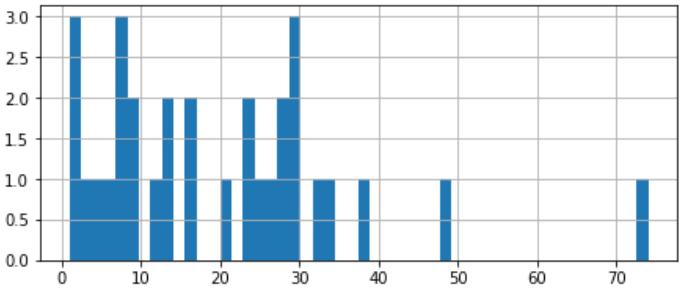


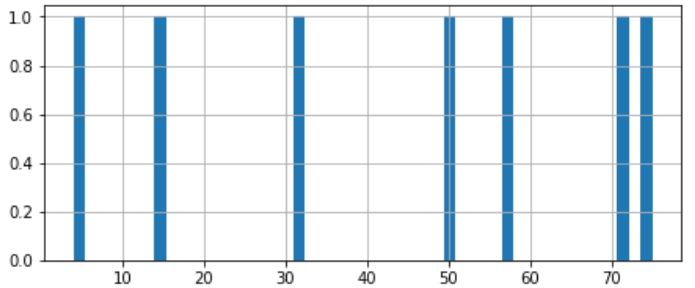

2
2
3
3
4
4
5
5
6
6
7
7
8
8



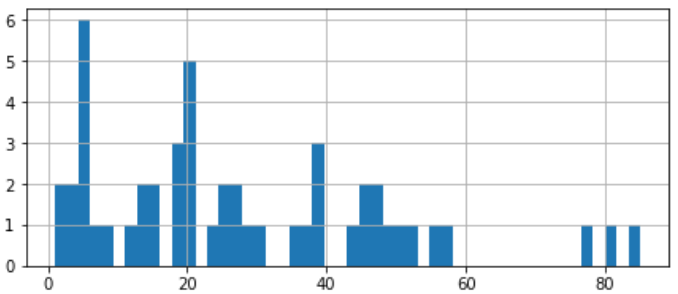
Построим гистограммы количества сообщений по каждому месяцу для тарифов «Смарт» (слева) и «Ультра» (справа).

In [42]:
```
for i in range(1,13):
    plt.figure(figsize=(16, 3))
    plt.subplot(1, 2, 1)
    data_messages_pivot_smart[data_messages_pivot_smart[i] != 0][i].hist(bins=50)
    plt.title(i)
    plt.subplot(1, 2, 2)
    data_messages_pivot_ultra[data_messages_pivot_ultra[i] != 0][i].hist(bins=50)
    plt.title(i)
```

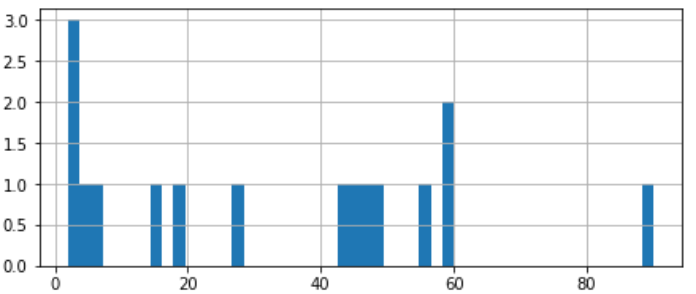




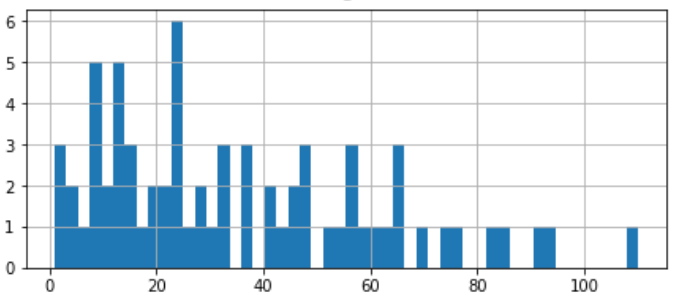
2



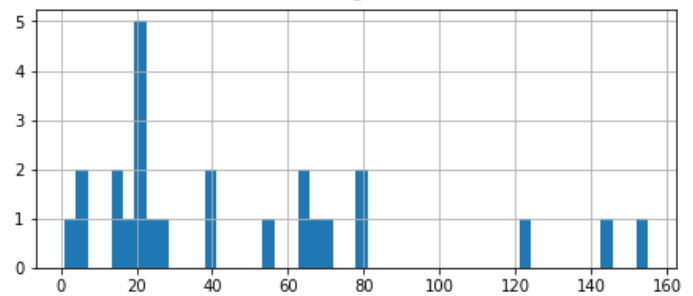
2



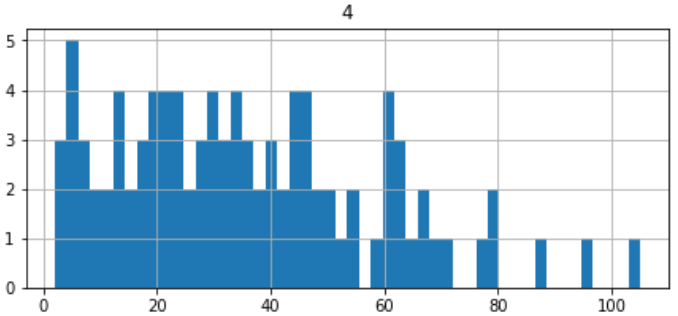
3



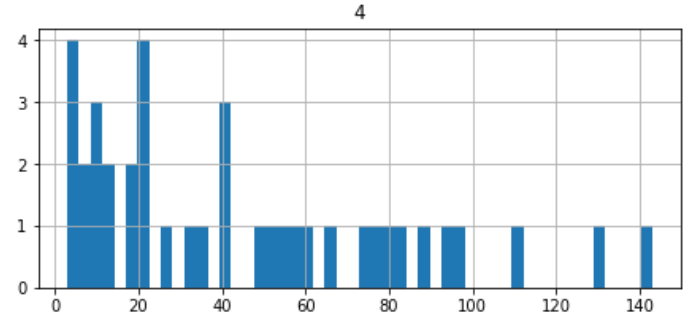
3



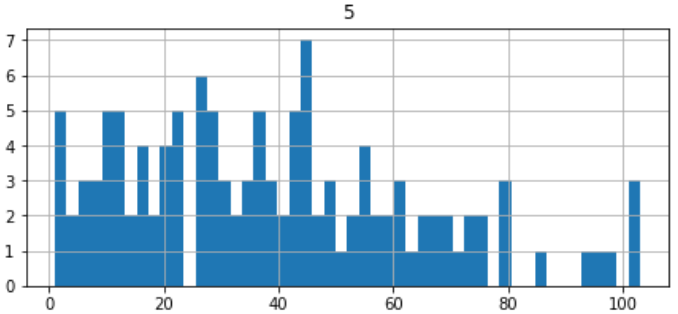
4



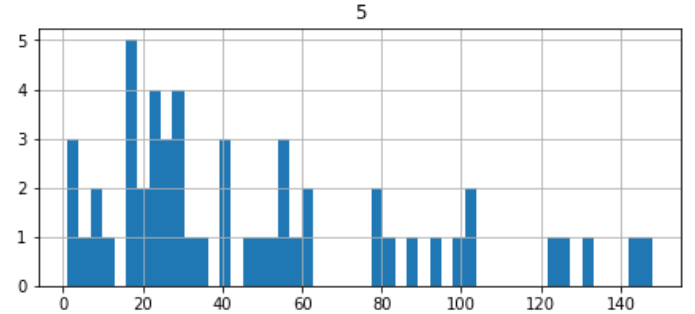
4



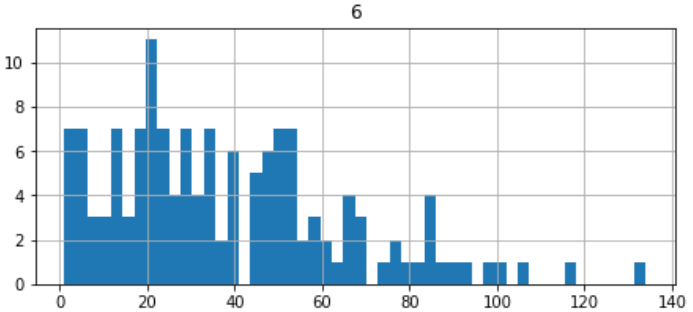
5



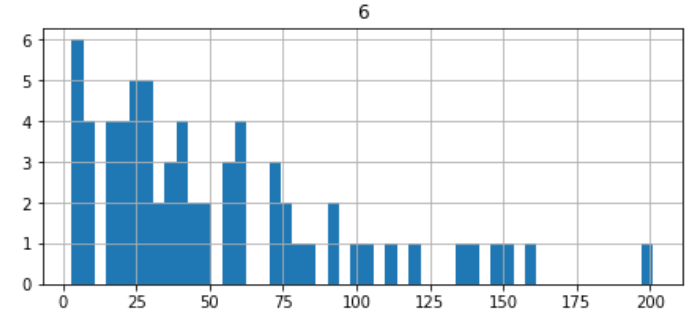
5



6

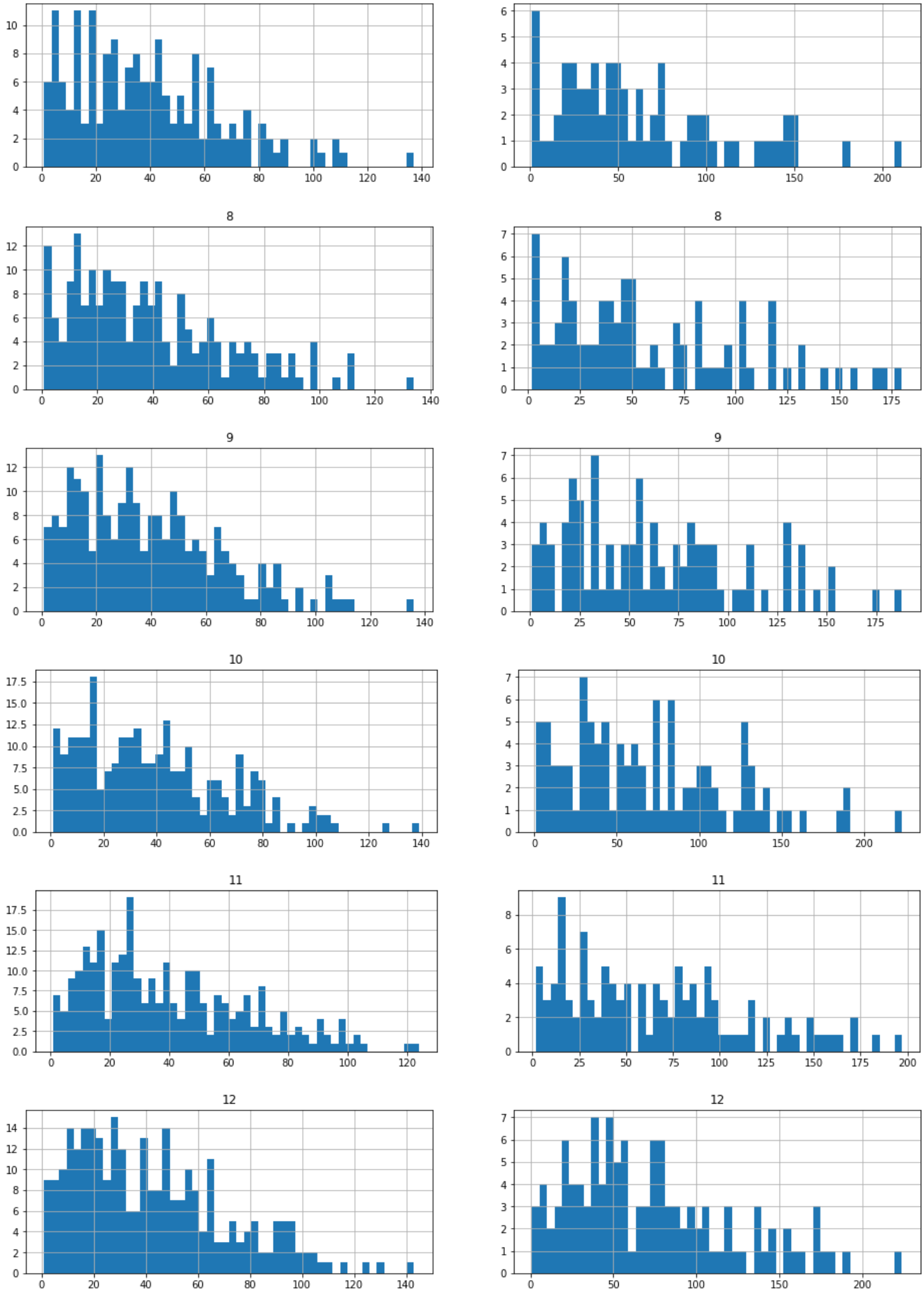


6



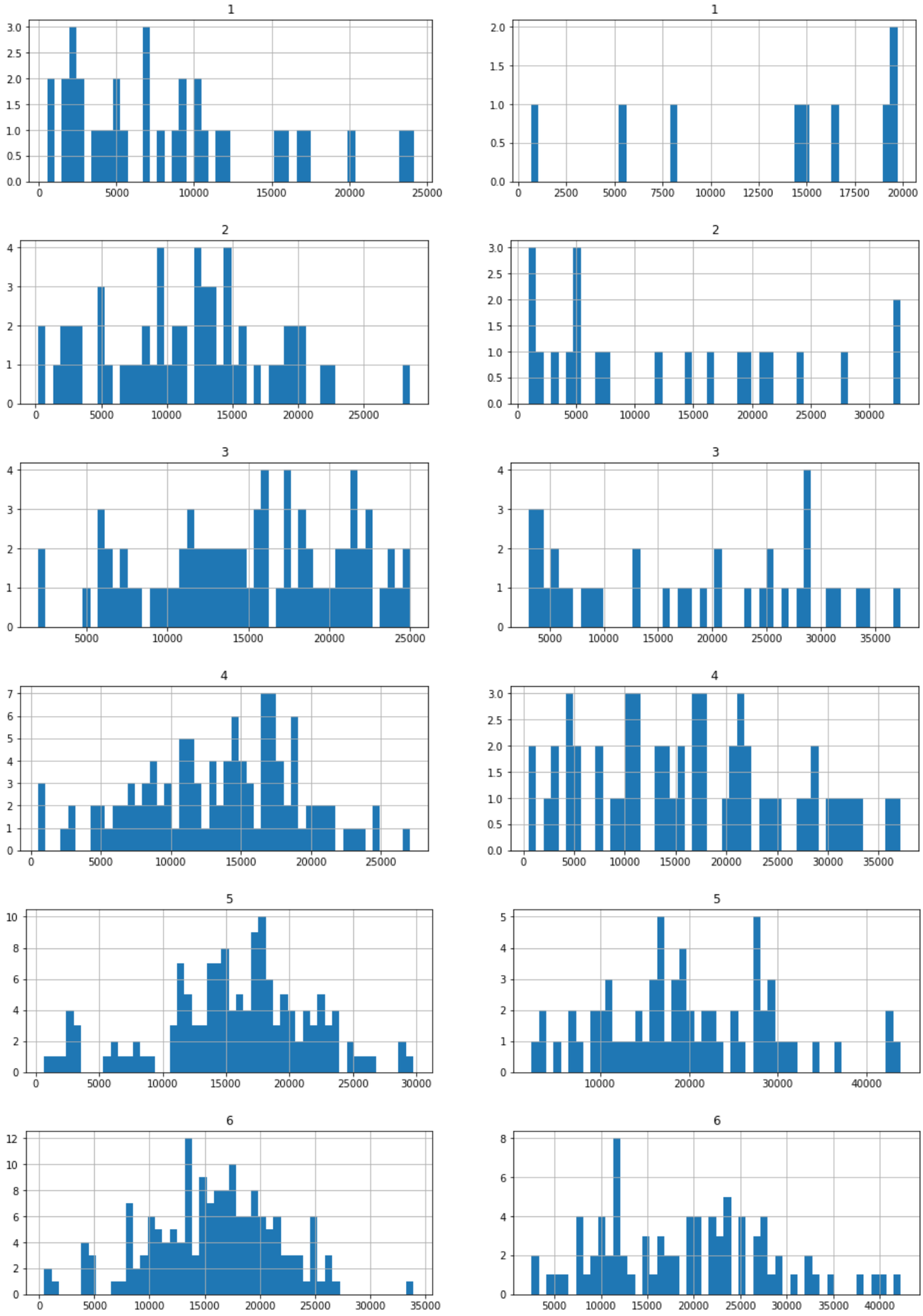
7

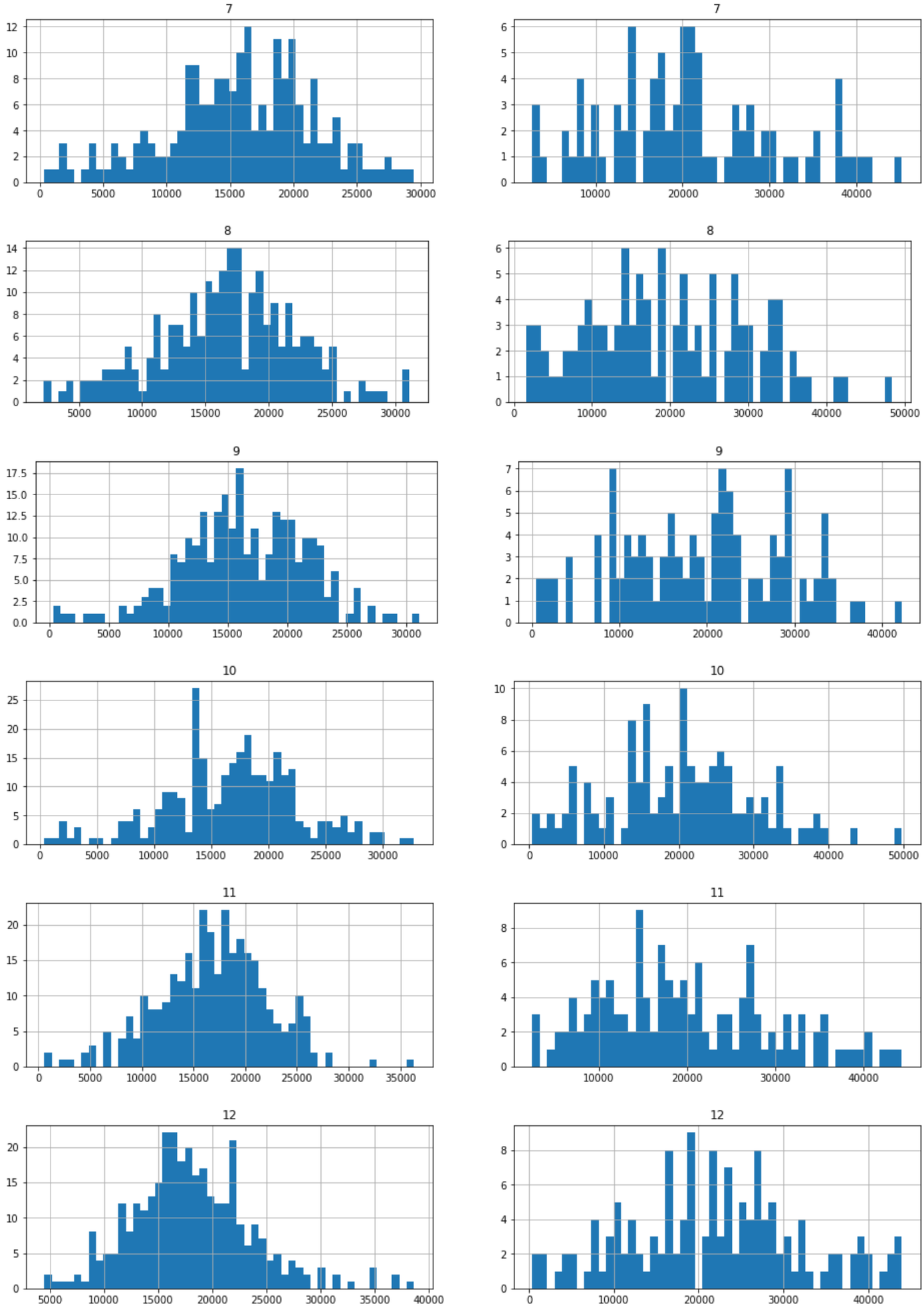
7



Построим гистограммы объёма интернет-трафика по каждому месяцу для тарифов «Смарт» (слева) и «Ультра» (справа).

In [43]:
```
for i in range(1,13):
    plt.figure(figsize=(16, 3))
    plt.subplot(1, 2, 1)
    data_internet_pivot_smart[data_internet_pivot_smart[i] != 0][i].hist(bins=50)
    plt.title(i)
    plt.subplot(1, 2, 2)
    data_internet_pivot_ultra[data_internet_pivot_ultra[i] != 0][i].hist(bins=50)
    plt.title(i)
```





Вывод

Пользователи подключались постепенно, больше всего их к концу года, соответственно распределение показателей стремится к нормальному в конце года.

Найдём упущенную дисперсию количества минут разговора по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_calls_pivot_smart_var` и выведем на экран.

In [44]:
```
data_calls_pivot_smart_var = np.var(data_calls_pivot_smart.loc[:, range(1, 13)]\
                                                        [data_calls_pivot_smart.loc[:, range(1, 13)] !=
 0])
data_calls_pivot_smart_var
```

Out[44]:
```
1     22613.382920
2     35848.137931
3     36400.821049
4     33596.189433
5     34646.246366
6     29826.468742
7     32154.024504
8     34116.489809
9     32896.247671
10    34817.283719
11    31085.400454
12    39088.928536
dtype: float64
```

Аналогично для тарифа «Ультра».

In [45]:
```
data_calls_pivot_ultra_var = np.var(data_calls_pivot_ultra.loc[:, range(1, 13)]\
                                                        [data_calls_pivot_ultra.loc[:, range(1, 13)] !=
 0])
data_calls_pivot_ultra_var
```

Out[45]:
```
1      64855.728395
2      84034.105860
3     108645.216947
4      64836.971164
5      73984.902344
6      85092.394452
7      83376.985335
8      88620.677912
9      95944.940165
10     96432.195059
11     87876.172105
12    107315.277846
dtype: float64
```

Найдём среднее количество минут разговора по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_calls_pivot_smart_mean` и выведем на экран.

In [46]:
```
data_calls_pivot_smart_mean = data_calls_pivot_smart.loc[:, range(1, 13)]\
                                                        [data_calls_pivot_smart.loc[:, range(1, 13)] !=
 0].mean()
data_calls_pivot_smart_mean
```

Out[46]:
```
1     210.363636
2     299.000000
3     390.480519
4     370.810811
5     387.773050
6     398.546012
7     408.776119
8     423.743590
9     430.737452
10    428.098976
11    437.095238
12    480.494048
dtype: float64
```

Аналогично для тарифа «Ультра».

In [47]:
```
data_calls_pivot_ultra_mean = data_calls_pivot_ultra.loc[:, range(1, 13)]\
                                                        [data_calls_pivot_ultra.loc[:, range(1, 13)] !=
 0].mean()
data_calls_pivot_ultra_mean
```

Out[47]:
```
1     428.777778
2     310.260870
3     490.162162
4     475.830189
5     534.937500
6     497.975309
7     571.482759
```

```
8     536.441176
9     564.472727
10    582.305085
11    562.852713
12    615.544776
dtype: float64
```

Посчитаем итоговое среднее количество минут разговора для пользователей тарифа «Смарт».

In [48]:
```
data_calls_pivot_smart_mean.mean()
```

Out[48]:
```
388.8266209398186
```

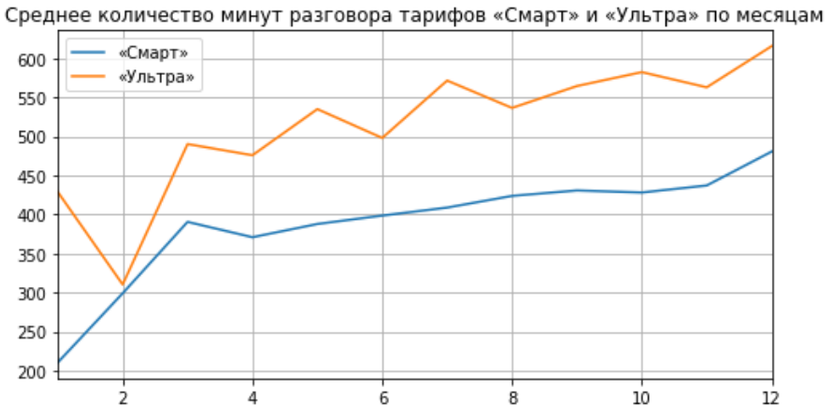
Посчитаем итоговое среднее количество минут разговора для пользователей тарифа «Ультра».

In [49]:
```
data_calls_pivot_ultra_mean.mean()
```

Out[49]:
```
514.2535869361536
```

Построим график среднего количества минут разговора тарифов «Смарт» и «Ультра» по месяцам.

In [50]:
```
plt.figure(figsize=(8, 4))
data_calls_pivot_smart_mean.plot(label='«Смарт»')
data_calls_pivot_ultra_mean.plot(label='«Ультра»')
plt.title('Среднее количество минут разговора тарифов «Смарт» и «Ультра» по месяцам')
plt.legend()
plt.grid(True);
```



Найдём упущенную дисперсию количества сообщений по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_messages_pivot_smart_var` и выведем на экран.

In [51]:
```
data_messages_pivot_smart_var = np.var(data_messages_pivot_smart.loc[:, range(1, 13)]\
                                       [data_messages_pivot_smart.loc[:, range(1, 13)]
 != 0])
data_messages_pivot_smart_var
```

Out[51]:
```
1     244.000000
2     431.656400
3     660.073514
4     512.471246
5     633.539909
6     726.483436
7     715.218259
8     763.452206
9     699.355306
10    733.603401
11    708.726694
12    829.546601
dtype: float64
```

Аналогично для тарифа «Ультра».

```
In [52]: data_messages_pivot_ultra_var = np.var(data_messages_pivot_ultra.loc[:, range(1, 13)]\
                                                       [data_messages_pivot_ultra.loc[:, range(1, 13)]
          != 0])
         data_messages_pivot_ultra_var
```

```
Out[52]: 1      655.959184
         2      668.808594
         3     1753.600000
         4     1372.147272
         5     1529.562130
         6     1906.782357
         7     2160.720961
         8     1999.978177
         9     1886.692078
         10    2403.047224
         11    2229.390555
         12    2342.675141
         dtype: float64
```

Найдём среднее количество сообщений по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_messages_pivot_smart_mean` и выведем на экран.

```
In [53]: data_messages_pivot_smart_mean = data_messages_pivot_smart.loc[:, range(1, 13)]\
                                                       [data_messages_pivot_smart.loc[:, range(1, 13)]
          != 0].mean()
         data_messages_pivot_smart_mean
```

```
Out[53]: 1     20.000000
         2     27.940000
         3     35.550725
         4     35.864583
         5     39.032787
         6     38.577465
         7     38.988506
         8     38.750000
         9     39.219731
         10    39.082677
         11    39.881481
         12    42.264808
         dtype: float64
```

Аналогично для тарифа «Ультра».

```
In [54]: data_messages_pivot_ultra_mean = data_messages_pivot_ultra.loc[:, range(1, 13)]\
                                                       [data_messages_pivot_ultra.loc[:, range(1, 13)]
          != 0].mean()
         data_messages_pivot_ultra_mean
```

```
Out[54]: 1     43.428571
         2     32.937500
         3     47.800000
         4     44.512821
         5     48.769231
         6     52.313433
         7     62.136986
         8     58.897727
         9     63.294737
         10    69.776699
         11    66.159292
         12    72.008547
         dtype: float64
```

Посчитаем итоговое среднее количество сообщений для пользователей тарифа «Смарт».

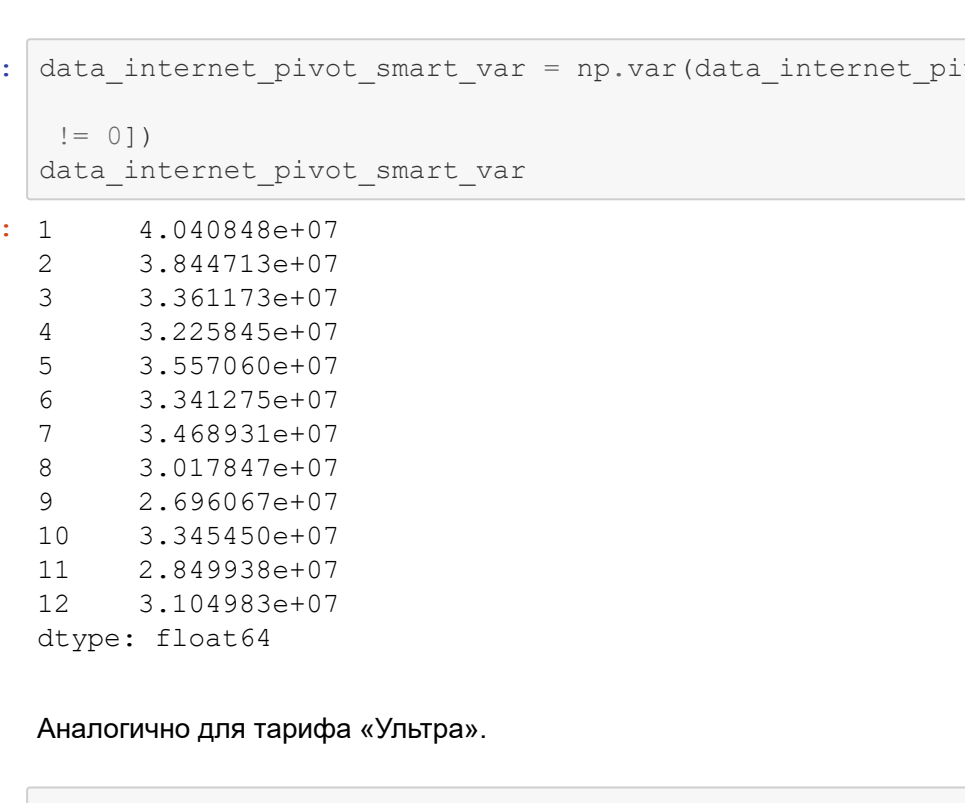
```
In [55]: data_messages_pivot_smart_mean.mean()
Out[55]: 36.2627302785857
```

Посчитаем итоговое среднее количество сообщений для пользователей тарифа «Ультра».

```
In [56]: data_messages_pivot_ultra_mean.mean()
Out[56]: 55.169628669643124
```

Построим график среднего количества сообщений тарифов «Смарт» и «Ультра» по месяцам.

```
In [57]: plt.figure(figsize=(8, 4))
         data_messages_pivot_smart_mean.plot(label='«Смарт»')
         data_messages_pivot_ultra_mean.plot(label='«Ультра»')
         plt.title('Среднее количество сообщений тарифов «Смарт» и «Ультра» по месяцам')
         plt.legend()
         plt.grid(True);
```



Найдём упущенную дисперсию объёма интернет-трафика по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_internet_pivot_smart_var` и выведем на экран.

```
In [58]: data_internet_pivot_smart_var = np.var(data_internet_pivot_smart.loc[:, range(1, 13)]\
                                                       [data_internet_pivot_smart.loc[:, range(1, 13)]
          != 0])
         data_internet_pivot_smart_var
```

```
Out[58]: 1     4.040848e+07
         2     3.844713e+07
         3     3.361173e+07
         4     3.225845e+07
         5     3.557060e+07
         6     3.341275e+07
         7     3.468931e+07
         8     3.017847e+07
         9     2.696067e+07
         10    3.345450e+07
         11    2.849938e+07
         12    3.104983e+07
         dtype: float64
```

Аналогично для тарифа «Ультра».

```
In [59]: data_internet_pivot_ultra_var = np.var(data_internet_pivot_ultra.loc[:, range(1, 13)]\
                                                       [data_internet_pivot_ultra.loc[:, range(1, 13)]
          != 0])
         data_internet_pivot_ultra_var
```

```
Out[59]: 1     4.197903e+07
         2     1.049058e+08
         3     1.169232e+08
         4     9.290731e+07
         5     9.076461e+07
         6     8.215174e+07
         7     9.675161e+07
         8     1.063020e+08
         9     8.688305e+07
         10    9.294793e+07
         11    9.537295e+07
         12    1.044366e+08
         dtype: float64
```

Найдём среднее объёма интернет-трафика по месяцам для пользователей тарифа «Смарт», сохраним в переменную `data_internet_pivot_smart_mean` и выведем на экран.

```
In [60]: data_internet_pivot_smart_mean = data_internet_pivot_smart.loc[:, range(1, 13)]\
                                                       [data_internet_pivot_smart.loc[:, range(1, 13)]
          != 0].mean()
         data_internet_pivot_smart_mean
```

```
Out[60]: 1      8524.823529
         2     11611.603448
         3     15124.090909
         4     13479.857143
         5     15825.617021
         6     15815.164634
         7     15763.900498
         8     16703.726496
         9     16325.050000
         10    16836.286689
         11    16917.987342
         12    18137.296736
         dtype: float64
```

Аналогично для тарифа «Ультра».

```
In [61]: data_internet_pivot_ultra_mean = data_internet_pivot_ultra.loc[:, range(1, 13)]\
                                                       [data_internet_pivot_ultra.loc[:, range(1, 13)]
          != 0].mean()
         data_internet_pivot_ultra_mean
```

```
Out[61]: 1     13153.555556
         2     12858.954545
         3     17552.405405
         4     16844.236364
         5     19977.969231
         6     19170.807229
         7     20742.911111
         8     19779.226415
         9     19391.280702
         10    20228.672131
         11    19954.586466
         12    21972.072464
         dtype: float64
```

Посчитаем итоговый средний объём интернет-трафика для пользователей тарифа «Смарт».

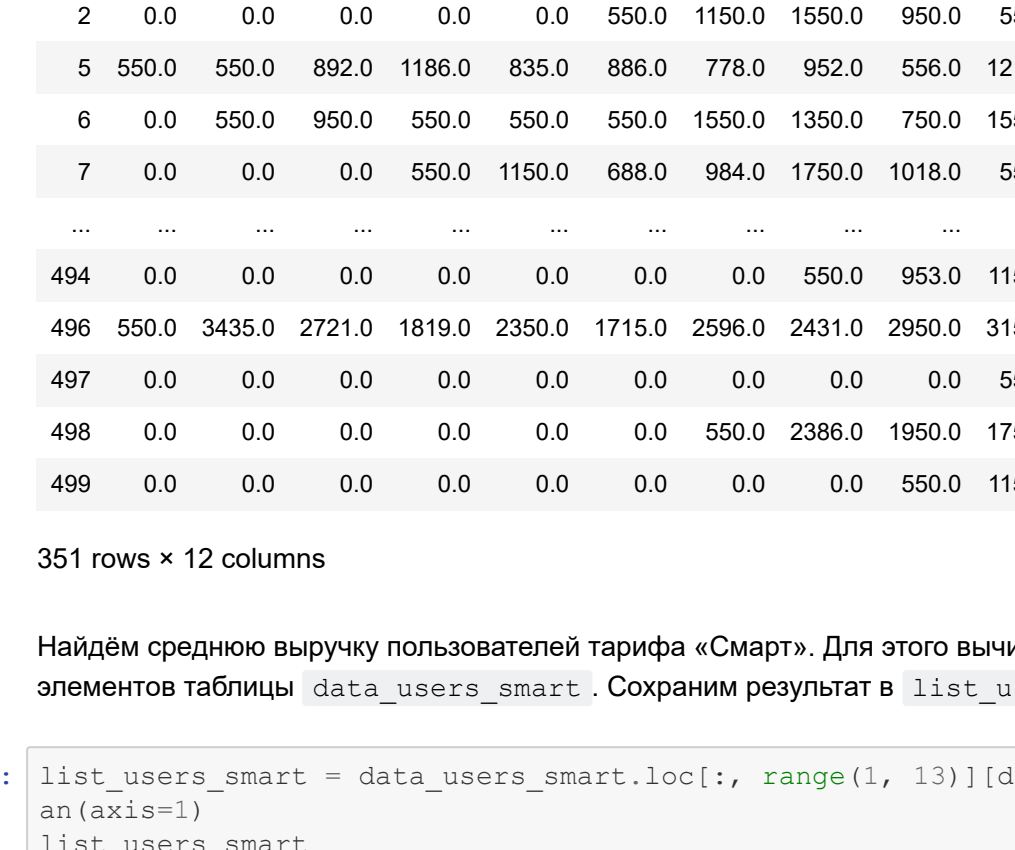
```
In [62]: data_internet_pivot_smart_mean.mean()
Out[62]: 15088.78370378288
```

Посчитаем итоговый средний объём интернет-трафика для пользователей тарифа «Ультра».

```
In [63]: data_internet_pivot_ultra_mean.mean()
Out[63]: 18468.889801564805
```

Построим график среднего объёма интернет-трафика тарифов «Смарт» и «Ультра» по месяцам.

```
In [64]: plt.figure(figsize=(8, 4))
         data_internet_pivot_smart_mean.plot(label='«Смарт»')
         data_internet_pivot_ultra_mean.plot(label='«Ультра»')
         plt.title('Средний объём интернет-трафика тарифов «Смарт» и «Ультра» по месяцам')
         plt.legend()
         plt.grid(True);
```



Вывод

Пользователи тарифа «Ультра» активнее пользуются услугами сотовой связи, чем пользователи тарифа «Смарт».

Шаг 4. Проверьте гипотезы

Выберем пользователей тарифа «Смарт».

```
In [65]: data_users_smart.loc[:, range(1, 13)]
```

Out[65]:

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1350.0	550.0
2	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1550.0	950.0	550.0	1350.0	1150.0
5	550.0	550.0	892.0	1186.0	835.0	886.0	778.0	952.0	556.0	1219.0	706.0	838.0
6	0.0	550.0	950.0	550.0	550.0	550.0	1550.0	1350.0	750.0	1550.0	550.0	1350.0
7	0.0	0.0	0.0	550.0	1150.0	688.0	984.0	1750.0	1018.0	550.0	1190.0	550.0
...	...	...	...	...	...	...	...	...	...	...	...	...
494	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	953.0	1150.0	1150.0	1150.0
496	550.0	3435.0	2721.0	1819.0	2350.0	1715.0	2596.0	2431.0	2950.0	3150.0	3039.0	1807.0
497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	550.0	1685.0
498	0.0	0.0	0.0	0.0	0.0	0.0	550.0	2386.0	1950.0	1750.0	0.0	0.0
499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1513.0	598.0

351 rows × 12 columns

Найдём среднюю выручку пользователей тарифа «Смарт». Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_smart`. Сохраним результат в `list_users_smart` и выведем на экран.

```
In [66]: list_users_smart = data_users_smart.loc[:, range(1, 13)][data_users_smart.loc[:, range(1, 13)] != 0].me
         an(axis=1)
         list_users_smart
```

```
Out[66]: 1       950.000000
         2      1035.714286
         5       829.000000
         6       931.818182
         7       936.666667
                   ...
         494     990.600000
         496    2380.250000
         497     928.333333
         498    1659.000000
         499     952.750000
         Length: 351, dtype: float64
```

Выберем пользователей тарифа «Ультра».

```
In [67]: data_users_ultra.loc[:, range(1, 13)]
```

Out[67]:

	1	2	3	4	5	6	7	8	9	10	11	12
0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0
4	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0
...	...	...	...	...	...	...	...	...	...	...	...	...
476	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
485	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0
491	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
493	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	3750.0
495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	2400.0	1950.0	3300.0	1950.0

149 rows × 12 columns

Найдём среднюю выручку пользователей тарифа «Ультра». Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_ultra`. Сохраним результат в `list_users_ultra` и выведем на экран.

```
In [68]: list_users_ultra = data_users_ultra.loc[:, range(1, 13)][data_users_ultra.loc[:, range(1, 13)] != 0].me
         an(axis=1)
         list_users_ultra
```

```
Out[68]: 0      1950.0
         3      1950.0
         4      1950.0
         13     1950.0
         16     1950.0
                 ...
         476    1950.0
         485    1950.0
         491    1950.0
         493    2850.0
         495    2310.0
         Length: 149, dtype: float64
```

Сравним среднюю выручку пользователей тарифов «Смарт» и «Ультра». Для этого проверим гипотезу о равенстве среднего двух генеральных совокупностей методом `scipy.stats.ttest_ind()`.

```
In [70]: alpha = 0.05 # пороговое значение

         results = st.ttest_ind(
             list_users_smart,
             list_users_ultra,
             equal_var = False
         )

         print('p-значение:', results.pvalue)

         if (results.pvalue < alpha):
             print("Отвергаем нулевую гипотезу")
         else:
             print("Не получилось отвергнуть нулевую гипотезу")

         p-значение: 7.689671550143974e-59
         Отвергаем нулевую гипотезу
```

Средняя выручка пользователей тарифов «Ультра» и «Смарт» различается.

```
In [71]: np.var(list_users_smart)
Out[71]: 368543.8372584689

In [72]: np.var(list_users_ultra)
Out[72]: 89420.18554420327
```

Выберем пользователей тарифа «Смарт», проживающих в Москве, сохраним их в `data_users_smart_moscow` и выведем на экран.

```
In [73]: data_users_smart_moscow = data_users_smart.query('city == "Москва"').loc[:, range(1, 13)]
         data_users_smart_moscow
```

Out[73]:

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1350.0	550.0
7	0.0	0.0	0.0	550.0	1150.0	688.0	984.0	1750.0	1018.0	550.0	1190.0	550.0
31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1048.0	736.0
33	0.0	0.0	0.0	0.0	0.0	0.0	550.0	2031.0	2831.0	1640.0	2616.0	1816.0
38	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	950.0	550.0	950.0
...	...	...	...	...	...	...	...	...	...	...	...	...
447	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1806.0	1827.0	3148.0
449	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1023.0	1192.0	1005.0	1926.0	931.0
450	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1634.0	891.0
481	0.0	550.0	1150.0	1550.0	1150.0	550.0	1350.0	1750.0	950.0	1150.0	1550.0	950.0
490	0.0	0.0	0.0	550.0	1750.0	1550.0	586.0	950.0	550.0	1750.0	1350.0	750.0

67 rows × 12 columns

Выберем пользователей тарифа «Смарт», проживающих в других регионах, сохраним их в `data_users_smart_not_moscow`, и выведем на экран.

```
In [74]: data_users_smart_not_moscow = data_users_smart.query('city != "Москва"').loc[:, range(1, 13)]
         data_users_smart_not_moscow
```

Out[74]:

	1	2	3	4	5	6	7	8	9	10	11	12
2	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1550.0	950.0	550.0	1350.0	1150.0
5	550.0	550.0	892.0	1186.0	835.0	886.0	778.0	952.0	556.0	1219.0	706.0	838.0
6	0.0	550.0	950.0	550.0	550.0	550.0	1550.0	1350.0	750.0	1550.0	550.0	1350.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	568.0
9	0.0	0.0	550.0	550.0	550.0	750.0	2550.0	2350.0	1950.0	1550.0	950.0	1350.0
...	...	...	...	...	...	...	...	...	...	...	...	...
494	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	953.0	1150.0	1150.0	1150.0
496	550.0	3435.0	2721.0	1819.0	2350.0	1715.0	2596.0	2431.0	2950.0	3150.0	3039.0	1807.0
497	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	550.0	1685.0
498	0.0	0.0	0.0	0.0	0.0	0.0	550.0	2386.0	1950.0	1750.0	0.0	0.0
499	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	550.0	1150.0	1513.0	598.0

284 rows × 12 columns

Найдём среднюю выручку пользователей тарифа «Смарт», проживающих в Москве. Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_smart_moscow`. Сохраним результат в `list_users_smart_moscow` и выведем на экран.

```
In [75]: list_users_smart_moscow = data_users_smart_moscow[data_users_smart_moscow != 0].mean(axis=1)
         list_users_smart_moscow
```

```
Out[75]: 1       950.000000
         7       936.666667
         31      892.000000
         33     1914.000000
         38      816.666667
                   ...
         447    1832.750000
         449    1104.500000
         450    1025.000000
         481    1150.000000
         490    1087.333333
         Length: 67, dtype: float64
```

Найдём среднюю выручку пользователей тарифа «Смарт», проживающих в в других регионах. Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_smart_not_moscow`. Сохраним результат в `list_users_smart_not_moscow` и выведем на экран.

```
In [76]: list_users_smart_not_moscow = data_users_smart_not_moscow[data_users_smart_not_moscow != 0].mean(axis=1
         )
         list_users_smart_not_moscow
```

```
Out[76]: 2      1035.714286
         5       829.000000
         6       931.818182
         8       568.000000
         9      1310.000000
                   ...
         494     990.600000
         496    2380.250000
         497     928.333333
         498    1659.000000
         499     952.750000
         Length: 284, dtype: float64
```

Сравним среднюю выручку пользователей тарифа «Смарт» из Москвы и выручку пользователей из других регионов. Для этого проверим гипотезу о равенстве среднего двух генеральных совокупностей методом `scipy.stats.ttest_ind()`.

```
In [77]: results = st.ttest_ind(
             list_users_smart_moscow,
             list_users_smart_not_moscow,
             equal_var = False
         )

         print('p-значение:', results.pvalue)

         if (results.pvalue < alpha):
             print("Отвергаем нулевую гипотезу")
         else:
             print("Не получилось отвергнуть нулевую гипотезу")

         p-значение: 0.14993780434254916
         Не получилось отвергнуть нулевую гипотезу
```

```
In [80]: np.var(list_users_smart_moscow)
Out[80]: 296078.1366961734

In [81]: np.var(list_users_smart_not_moscow)
Out[81]: 383295.9219288168
```

Средняя выручка пользователей тарифа «Смарт» из Москвы не отличается от выручки пользователей тарифа «Смарт» из других регионов.

Выберем пользователей тарифа «Ультра», проживающих в Москве, сохраним их в `data_users_ultra_moscow` и выведем на экран.

```
In [82]: data_users_ultra_moscow = data_users_ultra.query('city == "Москва"').loc[:, range(1, 13)]
         data_users_ultra_moscow.head()
```

Out[82]:

	1	2	3	4	5	6	7	8	9	10	11	12
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0
49	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
57	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
80	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0
81	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0

Выберем пользователей тарифа «Ультра», проживающих в других регионах, сохраним их в `data_users_ultra_not_moscow`, и выведем на экран.

```
In [83]: data_users_ultra_not_moscow = data_users_ultra.query('city != "Москва"').loc[:, range(1, 13)]
         data_users_ultra_not_moscow
```

Out[83]:

	1	2	3	4	5	6	7	8	9	10	11	12
0	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
4	0.0	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	1950.0
18	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
...	...	...	...	...	...	...	...	...	...	...	...	...
474	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	3750.0
485	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0
491	0.0	0.0	0.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0	1950.0
493	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	3750.0
495	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1950.0	2400.0	1950.0	3300.0	1950.0

117 rows × 12 columns

Найдём среднюю выручку пользователей тарифа «Ультра», проживающих в Москве. Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_ultra_moscow`. Сохраним результат в `list_users_ultra_moscow` и выведем на экран.

```
In [84]: list_users_ultra_moscow = data_users_ultra_moscow[data_users_ultra_moscow != 0].mean(axis=1)
         list_users_ultra_moscow.head()
```

```
Out[84]: 3     1950.0
         49    1950.0
         57    1950.0
         80    1950.0
         81    1950.0
         dtype: float64
```

Найдём среднюю выручку пользователей тарифа «Ультра», проживающих в в других регионах. Для этого вычислим среднее значение по строкам для всех ненулевых элементов таблицы `data_users_ultra_not_moscow`. Сохраним результат в `list_users_ultra_not_moscow` и выведем на экран.

```
In [85]: list_users_ultra_not_moscow = data_users_ultra_not_moscow[data_users_ultra_not_moscow != 0].mean(axis=1
         )
         list_users_ultra_not_moscow
```

```
Out[85]: 0      1950.0
         4      1950.0
         13     1950.0
         16     1950.0
         18     1950.0
                 ...
         474    2850.0
         485    1950.0
         491    1950.0
         493    2850.0
         495    2310.0
         Length: 117, dtype: float64
```

Сравним среднюю выручку пользователей тарифа «Ультра» из Москвы и выручку пользователей из других регионов. Для этого проверим гипотезу о равенстве среднего двух генеральных совокупностей методом `scipy.stats.ttest_ind()`.

```
In [86]: results = st.ttest_ind(
             list_users_ultra_moscow,
             list_users_ultra_not_moscow,
             equal_var = False
         )

         print('p-значение:', results.pvalue)

         if (results.pvalue < alpha):
             print("Отвергаем нулевую гипотезу")
         else:
             print("Не получилось отвергнуть нулевую гипотезу")

         p-значение: 0.48164162894509566
         Не получилось отвергнуть нулевую гипотезу
```

```
In [87]: np.var(list_users_ultra_moscow)
Out[87]: 40193.89686492251

In [88]: np.var(list_users_ultra_not_moscow)
Out[88]: 102649.57007101324
```

Средняя выручка пользователей тарифа «Ультра» из Москвы не отличается от выручки пользователей тарифа «Ультра» из других регионов.

Вывод

Средняя выручка пользователей тарифов «Ультра» и «Смарт» различается.

Средняя выручка пользователей из Москвы не отличается от выручки пользователей из других регионов.

Шаг 5. Напишите общий вывод

Аномалий и ошибок в данных я не обнаружил. Нулевые значения длительности звонков и потраченного интернет-трафика не влияют на выручку, поэтому оставлены без изменений.

Некоторые пользователи не совершают звонки, другие не отправляют сообщения, третьи не пользуются интернетом.

Абонентская плата тарифа «Ультра», как правило, покрывает потребности пользователей, а пользователи тарифа «Смарт» часто выходят за пределы тарифного пакета.

Пользователи подключались постепенно, больше всего их к концу года, соответственно распределение показателей стремится к нормальному в конце года.

Пользователи тарифа «Ультра» активнее пользуются услугами сотовой связи, чем пользователи тарифа «Смарт».

Средняя выручка пользователей тарифов «Ультра» и «Смарт» различается.

Средняя выручка пользователей из Москвы не отличается от выручки пользователей из других регионов.

```
In [89]: list_users_smart.describe()
```

```
Out[89]: count     351.000000
         mean     1328.525755
         std       607.944750
         min       550.000000
         25%       875.250000
         50%      1177.600000
         75%      1682.722222
         max      4334.333333
         dtype: float64
```

```
In [90]: list_users_ultra.describe()
```

```
Out[90]: count     149.000000
         mean     2086.495177
         std       300.040624
         min      1950.000000
         25%      1950.000000
         50%      1950.000000
         75%      1987.500000
         max      3262.500000
         dtype: float64
```

Сравним суммарную выручку пользователей тарифов «Смарт» и «Ультра».

```
In [91]: list_users_smart.sum()
Out[91]: 466312.5398629148

In [92]: list_users_ultra.sum()
Out[92]: 310887.78138528136
```

Несмотря на то, что у пользователей «Ультра» выше средние и медианные суммы, у пользователей «Смарт» выше суммарная выручка и разброс сумм. Расходы пользователей «Ультра» редко превышают двух тысяч. Расходы пользователей «Смарт» ниже, но максимальные суммы выше, поэтому есть потенциал роста. (В тренажёре Тема 2. Описательная статистика, Урок. Скошенность наборов данных есть похожий пример.)

Вывод

У тарифа «Смарт» выше суммарная выручка и потенциал для её увеличения.