# Исследование тарифных планов «Смарт» и «Ультра»

**Задача: проанализировать поведение клиентов компании Мегалайн и сделать вывод — какой тариф выгоднее продвигать: Смарт или Ультра.**

Определение перспективного тарифа для телеком-компании

1. Провести первичный анализ данных
2. Провести предобработку данных (привести к нужным типам и исправить ошибки)
3. Добавить необходимые данные в таблицы (количество сделанных звонков и израсходованных минут разговора по месяцам; количество отправленных сообщений по месяцам; объем израсходованного интернет-трафика по месяцам; помесячную выручку с каждого пользователя)
4. Описать поведение клиентов (среднее кол-во сообщений, минут разговора, объем трафика)
5. Проверить гипотезы: средняя выручка пользователей тарифов «Ультра» и «Смарт» различается; средняя выручка пользователей из Москвы отличается от выручки пользователей из других регионов

In [1]:
```
#импортируем необходимые библиотеки
import pandas as pd
import matplotlib.pyplot as plt
from plotly import graph_objects as go
import plotly.express as px
import plotly.io as pio
pio.templates.default = "plotly_white"
import numpy as np
import seaborn as sns
sns.set(style="whitegrid")
colors =["#ef476f","#ffd166","#06d6a0","#118ab2","#073b4c"]
sns.set_palette(sns.color_palette(colors))
import re
from scipy import stats as st
import math as mth
import re
from scipy import stats as st
import math as mth
pd.set_option('display.float_format', '{:,.2f}'.format)
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# скачиваем файлы
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 [3]:
```
#получаем информацию
print(data_calls.info())
data_calls.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202607 entries, 0 to 202606
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype  
---  ------     --------------   -----  
 0   id         202607 non-null  object 
 1   call_date  202607 non-null  object 
 2   duration   202607 non-null  float64
 3   user_id    202607 non-null  int64  
```

```
dtypes: float64(1), int64(1), object(2)
memory usage: 6.2+ MB
None
```

Out[3]:

| | id | call_date | duration | user_id |
|---|---|---|---|---|
| **0** | 1000_0 | 2018-07-25 | 0.00 | 1000 |
| **1** | 1000_1 | 2018-08-17 | 0.00 | 1000 |
| **2** | 1000_2 | 2018-06-11 | 2.85 | 1000 |
| **3** | 1000_3 | 2018-09-21 | 13.80 | 1000 |
| **4** | 1000_4 | 2018-12-15 | 5.18 | 1000 |

In [4]:

```
data_calls.describe()
```

Out[4]:

| | duration | user_id |
|---|---|---|
| **count** | 202,607.00 | 202,607.00 |
| **mean** | 6.76 | 1,253.94 |
| **std** | 5.84 | 144.72 |
| **min** | 0.00 | 1,000.00 |
| **25%** | 1.30 | 1,126.00 |
| **50%** | 6.00 | 1,260.00 |
| **75%** | 10.70 | 1,379.00 |
| **max** | 38.00 | 1,499.00 |

1.1 Вывод: в таблице дата вызова и время пропущенных данных нет.

In [5]:

```
#получаем информацию
print(data_internet.info())
data_internet.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 149396 entries, 0 to 149395
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Unnamed: 0    149396 non-null  int64  
 1   id            149396 non-null  object 
 2   mb_used       149396 non-null  float64
 3   session_date  149396 non-null  object 
 4   user_id       149396 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 5.7+ MB
None
```

Out[5]:

| | Unnamed: 0 | id | mb_used | session_date | user_id |
|---|---|---|---|---|---|
| **0** | 0 | 1000_0 | 112.95 | 2018-11-25 | 1000 |
| **1** | 1 | 1000_1 | 1,052.81 | 2018-09-07 | 1000 |
| **2** | 2 | 1000_2 | 1,197.26 | 2018-06-25 | 1000 |
| **3** | 3 | 1000_3 | 550.27 | 2018-08-22 | 1000 |
| **4** | 4 | 1000_4 | 302.56 | 2018-09-24 | 1000 |

In [6]:

```
data_internet.describe()
```

Out[6]:

| | Unnamed: 0 | mb_used | user_id |
|---|---|---|---|
| **count** | 149,396.00 | 149,396.00 | 149,396.00 |
| **mean** | 74,697.50 | 370.19 | 1,252.10 |
| **std** | 43,127.05 | 278.30 | 144.05 |
| **min** | 0.00 | 0.00 | 1,000.00 |
| **25%** | 37,348.75 | 138.19 | 1,130.00 |
| **50%** | 74,697.50 | 348.01 | 1,251.00 |
| **75%** | 112,046.25 | 559.55 | 1,380.00 |
| **max** | 149,395.00 | 1,724.83 | 1,499.00 |

1.2 Вывод: в таблице дата сессии пропущенных данных нет.

In [7]:

```
#получаем информацию
print(data_messages.info())
data_messages.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 123036 entries, 0 to 123035
Data columns (total 3 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   id            123036 non-null  object
 1   message_date  123036 non-null  object
 2   user_id       123036 non-null  int64 
dtypes: int64(1), object(2)
memory usage: 2.8+ MB
None
```

Out[7]:

| | id | message_date | user_id |
|---|---|---|---|
| **0** | 1000_0 | 2018-06-27 | 1000 |
| **1** | 1000_1 | 2018-10-08 | 1000 |
| **2** | 1000_2 | 2018-08-04 | 1000 |
| **3** | 1000_3 | 2018-06-16 | 1000 |
| **4** | 1000_4 | 2018-12-05 | 1000 |

In [8]:

```
data_messages.describe()
```

Out[8]:

| | user_id |
|---|---|
| **count** | 123,036.00 |
| **mean** | 1,256.99 |
| **std** | 143.52 |
| **min** | 1,000.00 |
| **25%** | 1,134.00 |
| **50%** | 1,271.00 |
| **75%** | 1,381.00 |
| **max** | 1,499.00 |

1.3 Вывод: в таблице дата сообщений пропущенных данных нет.

In [9]:

```
#получаем информацию
print(data_tariffs.info())
data_tariffs.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   messages_included      2 non-null      int64 
 1   mb_per_month_included  2 non-null      int64 
 2   minutes_included       2 non-null      int64 
 3   rub_monthly_fee        2 non-null      int64 
 4   rub_per_gb             2 non-null      int64 
 5   rub_per_message        2 non-null      int64 
 6   rub_per_minute         2 non-null      int64 
 7   tariff_name            2 non-null      object
dtypes: int64(7), object(1)
memory usage: 256.0+ bytes
None
```

Out[9]:

| | messages_included | mb_per_month_included | minutes_included | rub_monthly_fee | rub_per_gb | rub_per_message |
|---|---|---|---|---|---|---|
| **0** | 50 | 15360 | 500 | 550 | 200 | 3 |
| **1** | 1000 | 30720 | 3000 | 1950 | 150 | 1 |

In [10]:

```
data_tariffs.describe()
```

Out[10]:

| | messages_included | mb_per_month_included | minutes_included | rub_monthly_fee | rub_per_gb | rub_per_mes |
|---|---|---|---|---|---|---|
| **count** | 2.00 | 2.00 | 2.00 | 2.00 | 2.00 | |
| **mean** | 525.00 | 23,040.00 | 1,750.00 | 1,250.00 | 175.00 | |
| **std** | 671.75 | 10,861.16 | 1,767.77 | 989.95 | 35.36 | |
| **min** | 50.00 | 15,360.00 | 500.00 | 550.00 | 150.00 | |
| **25%** | 287.50 | 19,200.00 | 1,125.00 | 900.00 | 162.50 | |
| **50%** | 525.00 | 23,040.00 | 1,750.00 | 1,250.00 | 175.00 | |
| **75%** | 762.50 | 26,880.00 | 2,375.00 | 1,600.00 | 187.50 | |
| **max** | 1,000.00 | 30,720.00 | 3,000.00 | 1,950.00 | 200.00 | |

1.4 Вывод: в таблице тарифы пропущенных данных нет

In [11]:

```
#получаем информацию
data_users.head()
```

Out[11]:

| | user_id | age | churn_date | city | first_name | last_name | reg_date | tariff |
|---|---|---|---|---|---|---|---|---|
| **0** | 1000 | 52 | NaN | Краснодар | Рафаил | Верещагин | 2018-05-25 | ultra |
| **1** | 1001 | 41 | NaN | Москва | Иван | Ежов | 2018-11-01 | smart |
| **2** | 1002 | 59 | NaN | Стерлитамак | Евгений | Абрамович | 2018-06-17 | smart |
| **3** | 1003 | 23 | NaN | Москва | Белла | Белякова | 2018-08-17 | ultra |
| **4** | 1004 | 68 | NaN | Новокузнецк | Татьяна | Авдеенко | 2018-05-14 | ultra |

In [12]:

```
data_users.info()
```

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

In [13]:
```
data_users.describe()
```

Out[13]:

| | user_id | age |
|---|---|---|
| **count** | 500.00 | 500.00 |
| **mean** | 1,249.50 | 46.59 |
| **std** | 144.48 | 16.67 |
| **min** | 1,000.00 | 18.00 |
| **25%** | 1,124.75 | 32.00 |
| **50%** | 1,249.50 | 46.00 |
| **75%** | 1,374.25 | 62.00 |
| **max** | 1,499.00 | 75.00 |

1.5 Вывод: в таблице данные пользователя пропущенных данных нет. Отсутствуют данные в столбце выхода из тарифа. Нам они не нужны, поэтому пропускаем.

### Заметили следующее:

- Необходимо округлить время звонков и мегабайты трафика в большую сторону в соответствии с условиями тарифов

- Имеем столбцы:
  - Таблица users (информация о пользователях):
    - user_id — уникальный идентификатор пользователя
    - first_name — имя пользователя
    - last_name — фамилия пользователя
    - age — возраст пользователя (годы)
    - reg_date — дата подключения тарифа (день, месяц, год)
    - churn_date — дата прекращения пользования тарифом (если значение пропущено, то тариф ещё действовал на момент выгрузки данных)
    - city — город проживания пользователя
    - tariff — название тарифного плана
  - Таблица calls (информация о звонках):
    - id — уникальный номер звонка
    - call_date — дата звонка
    - duration — длительность звонка в минутах
    - user_id — идентификатор пользователя, сделавшего звонок
  - Таблица messages (информация о сообщениях):
    - id — уникальный номер сообщения

- message_date — дата сообщения
- user_id — идентификатор пользователя, отправившего сообщение

- Таблица internet (информация об интернет-сессиях):
  - id — уникальный номер сессии
  - mb_used — объём потраченного за сессию интернет-трафика (в мегабайтах)
  - session_date — дата интернет-сессии
  - user_id — идентификатор пользователя
- Таблица tariffs (информация о тарифах):
  - tariff_name — название тарифа
  - rub_monthly_fee — ежемесячная абонентская плата в рублях
  - minutes_included — количество минут разговора в месяц, включённых в абонентскую плату
  - messages_included — количество сообщений в месяц, включённых в абонентскую плату
  - mb_per_month_included — объём интернет-трафика, включённого в абонентскую плату (в мегабайтах)
  - rub_per_minute — стоимость минуты разговора сверх тарифного пакета (например, если в тарифе 100 минут разговора в месяц, то со 101 минуты будет взиматься плата)
  - rub_per_message — стоимость отправки сообщения сверх тарифного пакета
  - rub_per_gb — стоимость дополнительного гигабайта интернет-трафика сверх тарифного пакета (1 гигабайт = 1024 мегабайта

# Предобработка

**Добавим в каждую из 3 базовых таблиц с данными колонку с месяцем. Для этого вначале приведем даты к формату datetime:**

In [14]:
```
#приводим в формат datetime
data_calls.index = pd.to_datetime(data_calls.index)
data_messages.index = pd.to_datetime(data_messages.index)
data_internet.index = pd.to_datetime(data_internet.index)
```

In [15]:
```
#добавляем столбец month
data_calls['month'] = pd.to_datetime(data_calls['call_date']).dt.month
data_messages['month'] = pd.to_datetime(data_messages['message_date']).dt.month
data_internet['month'] = pd.to_datetime(data_internet['session_date']).dt.month
```

In [16]:
```
#проверяем
data_calls.head()
```

Out[16]:

| | id | call_date | duration | user_id | month |
|---|---|---|---|---|---|
| **1970-01-01 00:00:00.000000000** | 1000_0 | 2018-07-25 | 0.00 | 1000 | 7 |
| **1970-01-01 00:00:00.000000001** | 1000_1 | 2018-08-17 | 0.00 | 1000 | 8 |
| **1970-01-01 00:00:00.000000002** | 1000_2 | 2018-06-11 | 2.85 | 1000 | 6 |
| **1970-01-01 00:00:00.000000003** | 1000_3 | 2018-09-21 | 13.80 | 1000 | 9 |
| **1970-01-01 00:00:00.000000004** | 1000_4 | 2018-12-15 | 5.18 | 1000 | 12 |

In [17]:
```
#округлим в большую сторону звонки
data_calls['duration'] = np.ceil(data_calls['duration'])
data_calls['duration'].head()#проверяем
```

Out[17]:
```
1970-01-01 00:00:00.000000000    0.00
```

```
1970-01-01 00:00:00.000000001    0.00
1970-01-01 00:00:00.000000002    3.00
1970-01-01 00:00:00.000000003   14.00
1970-01-01 00:00:00.000000004    6.00
Name: duration, dtype: float64
```

In [18]:
```
data_calls.head()
```

Out[18]:

| | id | call_date | duration | user_id | month |
|---|---|---|---|---|---|
| **1970-01-01 00:00:00.000000000** | 1000_0 | 2018-07-25 | 0.00 | 1000 | 7 |
| **1970-01-01 00:00:00.000000001** | 1000_1 | 2018-08-17 | 0.00 | 1000 | 8 |
| **1970-01-01 00:00:00.000000002** | 1000_2 | 2018-06-11 | 3.00 | 1000 | 6 |
| **1970-01-01 00:00:00.000000003** | 1000_3 | 2018-09-21 | 14.00 | 1000 | 9 |
| **1970-01-01 00:00:00.000000004** | 1000_4 | 2018-12-15 | 6.00 | 1000 | 12 |

In [19]:
```
#округлим в большую сторону сообщения
data_internet['mb_used'] = np.ceil(data_internet['mb_used'])
data_internet['mb_used'].head()
```

Out[19]:
```
1970-01-01 00:00:00.000000000     113.00
1970-01-01 00:00:00.000000001   1,053.00
1970-01-01 00:00:00.000000002   1,198.00
1970-01-01 00:00:00.000000003     551.00
1970-01-01 00:00:00.000000004     303.00
Name: mb_used, dtype: float64
```

In [20]:
```
#в таблице с интернет-трафиком для дальнейшего анализа переведем мб в гб
data_internet['gb_used'] = (data_internet['mb_used'] / 1024)
data_internet.head()#проверяем
```

Out[20]:

| | Unnamed: 0 | id | mb_used | session_date | user_id | month | gb_used |
|---|---|---|---|---|---|---|---|
| **1970-01-01 00:00:00.000000000** | 0 | 1000_0 | 113.00 | 2018-11-25 | 1000 | 11 | 0.11 |
| **1970-01-01 00:00:00.000000001** | 1 | 1000_1 | 1,053.00 | 2018-09-07 | 1000 | 9 | 1.03 |
| **1970-01-01 00:00:00.000000002** | 2 | 1000_2 | 1,198.00 | 2018-06-25 | 1000 | 6 | 1.17 |
| **1970-01-01 00:00:00.000000003** | 3 | 1000_3 | 551.00 | 2018-08-22 | 1000 | 8 | 0.54 |
| **1970-01-01 00:00:00.000000004** | 4 | 1000_4 | 303.00 | 2018-09-24 | 1000 | 9 | 0.30 |

сделаем pivot для трех таблиц. сгруппируем по user_id

In [21]:
```
#сделаем pivot для data_calls. сгруппируем по user_id и посчитаем кол-во звонков по месяцам:
calls_data_grouped = data_calls.pivot_table(index=['user_id', 'month'], values='duration',
                                            aggfunc=['sum', 'count'])

calls_data_grouped.reset_index()
calls_data_grouped.columns = ['calls_duration', 'calls_amount']
calls_data_grouped.head()
```

Out[21]:

| | | calls_duration | calls_amount |
|---|---|---|---|
| **user_id** | **month** | | |
| **1000** | **5** | 159.00 | 22 |
| | **6** | 172.00 | 43 |
| | **7** | 340.00 | 47 |

| | | calls_duration | calls_amount |
|---|---|---|---|
| **user_id** | **month** | | |
| | **8** | 408.00 | 52 |
| | **9** | 466.00 | 58 |

In [22]:
```
#сделаем pivot для data_internet. сгруппируем по user_id и посчитаем  объем израсходованного ин
internet_data_grouped = data_internet.pivot_table(index=['user_id', 'month'], values='gb_used',
                                                  aggfunc=['sum'])
internet_data_grouped.columns = ['gb_used']
internet_data_grouped.reset_index()
internet_data_grouped.head()
```

Out[22]:

| | | gb_used |
|---|---|---|
| **user_id** | **month** | |
| **1000** | **5** | 2.20 |
| | **6** | 22.71 |
| | **7** | 13.69 |
| | **8** | 13.74 |
| | **9** | 14.24 |

In [23]:
```
#посчитаем кол-во сообщений по месяцам:
messages_data_grouped = data_messages.pivot_table(index=['user_id', 'month'], values='id', aggf
messages_data_grouped.reset_index()
messages_data_grouped.columns=['messages_amount']
messages_data_grouped.head()
```

Out[23]:

| | | messages_amount |
|---|---|---|
| **user_id** | **month** | |
| **1000** | **5** | 22 |
| | **6** | 60 |
| | **7** | 75 |
| | **8** | 81 |
| | **9** | 57 |

находим суммарное количество звонков, сообщений и интернет-трафика

In [24]:
```
# находим суммарное количество звонков, сообщений и интернет-трафика
calls_internet_merged = calls_data_grouped.merge(internet_data_grouped, on=['user_id', 'month']
telecom_data = calls_internet_merged.merge(messages_data_grouped, on=['user_id', 'month'], how=
telecom_data_pvt = telecom_data.pivot_table(index=['user_id', 'month'])
telecom_data_pvt=telecom_data_pvt.reset_index()
telecom_data_pvt.head()
```

Out[24]:

| | user_id | month | calls_amount | calls_duration | gb_used | messages_amount |
|---|---|---|---|---|---|---|
| **0** | 1000 | 5 | 22 | 159.00 | 2.20 | 22.00 |
| **1** | 1000 | 6 | 43 | 172.00 | 22.71 | 60.00 |
| **2** | 1000 | 7 | 47 | 340.00 | 13.69 | 75.00 |
| **3** | 1000 | 8 | 52 | 408.00 | 13.74 | 81.00 |

| | user_id | month | calls_amount | calls_duration | gb_used | messages_amount |
|---|---|---|---|---|---|---|
| **4** | 1000 | 9 | 58 | 466.00 | 14.24 | 57.00 |

Добавим в таблицу telecom_data информацию о пользователях (предварительно приведя в формат datetime #дату регистрации)

In [25]:
```
# округляем гигабайты до целого в сторону увеличения
telecom_data['gb_used'] = np.ceil(telecom_data['gb_used'])
telecom_data['gb_used'].head()
```

Out[25]:
```
user_id  month
1000     5        3.00
         6       23.00
         7       14.00
         8       14.00
         9       15.00
Name: gb_used, dtype: float64
```

In [26]:
```
telecom_data = telecom_data_pvt.merge(data_users, on='user_id', how='right').pivot_table(
    index=['user_id', 'first_name', 'last_name', 'age', 'city', 'tariff', 'month'])
telecom_data = telecom_data.reset_index()
telecom_data.head()
```

Out[26]:

| | user_id | first_name | last_name | age | city | tariff | month | calls_amount | calls_duration | gb_used | messa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 5.00 | 22.00 | 159.00 | 2.20 | |
| **1** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 6.00 | 43.00 | 172.00 | 22.71 | |
| **2** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 7.00 | 47.00 | 340.00 | 13.69 | |
| **3** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 8.00 | 52.00 | 408.00 | 13.74 | |
| **4** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 9.00 | 58.00 | 466.00 | 14.24 | |

In [27]:
```
telecom_data.user_id.nunique()
```

Out[27]:
```
492
```

In [28]:
```
# напишем программу для расчета общих затрат пользователей на тарифах 'smart' и 'ultra'
def monthly_revenue(row):
    if row['tariff'] == 'smart': #тариф 'smart'
        if row['messages_amount'] > 50:
            messages_extra = (row['messages_amount'] - 50)*3
        else:
            messages_extra = 0
        if row['calls_amount'] > 500:
            calls_extra = (row['calls_amount'] - 500)*3
        else:
            calls_extra = 0
        if row['gb_used'] > 15:
            gb_extra = (row['gb_used'] - 15)*200
        else:
            gb_extra = 0
        total_cost = messages_extra + calls_extra + gb_extra + 550
    if row['tariff'] == 'ultra': #тариф 'ultra'
        if row['messages_amount'] > 1000:
            messages_extra = (row['messages_amount'] - 1000)*1
        else:
            messages_extra = 0
        if row['calls_amount'] > 3000:
            calls_extra = (row['calls_amount'] - 3000)*1
        else:
```

```
            calls_extra = 0
        if row['gb_used'] > 30:
            gb_extra = (row['gb_used'] - 30)*150
        else:
            gb_extra = 0
        total_cost = messages_extra + calls_extra + gb_extra + 1950
    return total_cost
```

In [29]:

```
#таблица помесячной выручки с каждого пользователя
telecom_data['total_cost'] = telecom_data.apply(monthly_revenue, axis=1)
telecom_data.pivot_table(index=['user_id', 'first_name', 'last_name', 'age', 'city', 'tariff',
telecom_data.head()
```

Out[29]:

| | user_id | first_name | last_name | age | city | tariff | month | calls_amount | calls_duration | gb_used | messa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 5.00 | 22.00 | 159.00 | 2.20 | |
| **1** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 6.00 | 43.00 | 172.00 | 22.71 | |
| **2** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 7.00 | 47.00 | 340.00 | 13.69 | |
| **3** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 8.00 | 52.00 | 408.00 | 13.74 | |
| **4** | 1000 | Рафаил | Верещагин | 52 | Краснодар | ultra | 9.00 | 58.00 | 466.00 | 14.24 | |

In [30]:

```
telecom_data['total_cost'].describe()
```

Out[30]:

```
count   3,174.00
mean    1,394.48
std       762.05
min       550.00
25%       580.00
50%     1,385.62
75%     1,950.00
max     5,190.02
Name: total_cost, dtype: float64
```

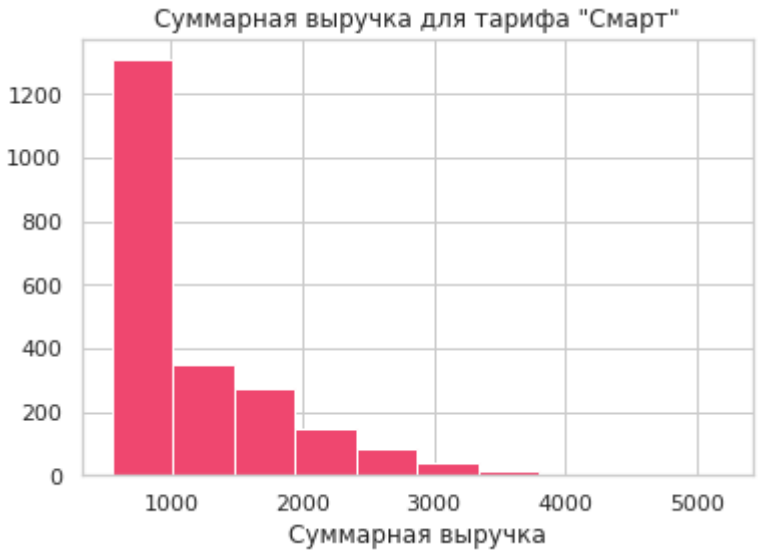
Средняя помесячная выручка - 1386

# Анализ данных

**Посчитаем среднее количество для тарифа smart.**

In [31]:

```
smart_data = telecom_data.query('tariff == "smart"')
smart_data[['total_cost']].hist()
plt.title('Суммарная выручка для тарифа "Смарт"')
plt.xlabel('Суммарная выручка')
plt.show()
smart_data[['total_cost']].describe()
```



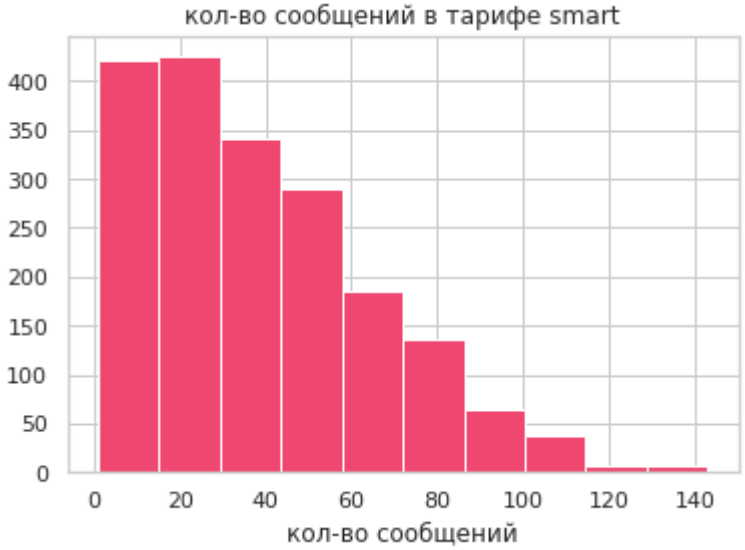

Out[31]:

| | total_cost |
|---|---|
| **count** | 2,223.00 |
| **mean** | 1,109.14 |
| **std** | 708.64 |
| **min** | 550.00 |
| **25%** | 550.00 |
| **50%** | 796.48 |
| **75%** | 1,491.87 |
| **max** | 5,190.02 |

Наибольшая сумма расходов в тарифе smart приходится на диапазон от 550 до 1500 руб, но есть такие, кто тратит более 5000 руб

In [32]:

```
# кол-во сообщений в тарифе smart
smart_data = telecom_data.query('tariff == "smart"')
smart_data[['messages_amount']].hist()
plt.title("кол-во сообщений в тарифе smart")
plt.xlabel('кол-во сообщений')
plt.show()
smart_data[['messages_amount']].describe()
```



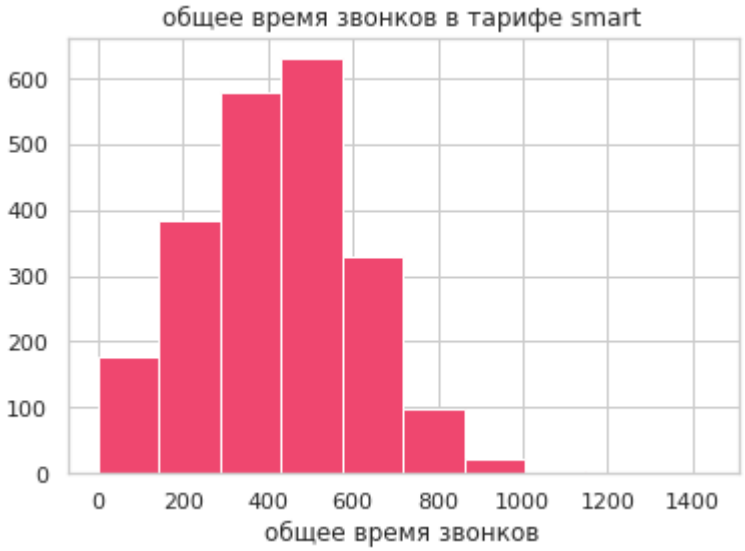

Out[32]:

| | messages_amount |
|---|---|
| **count** | 1,916.00 |

| | messages_amount |
|---|---|
| **mean** | 38.75 |
| **std** | 26.82 |
| **min** | 1.00 |
| **25%** | 17.00 |
| **50%** | 34.00 |
| **75%** | 55.00 |
| **max** | 143.00 |

среднее число сообщений - 34 шт, наибольшее -143

In [33]:
```
# общее время звонков в тарифе smart
smart_data = telecom_data.query('tariff == "smart"')
smart_data[['calls_duration']].hist()
plt.title("общее время звонков в тарифе smart")
plt.xlabel('общее время звонков')
plt.show()
smart_data[['calls_duration']].describe()
```



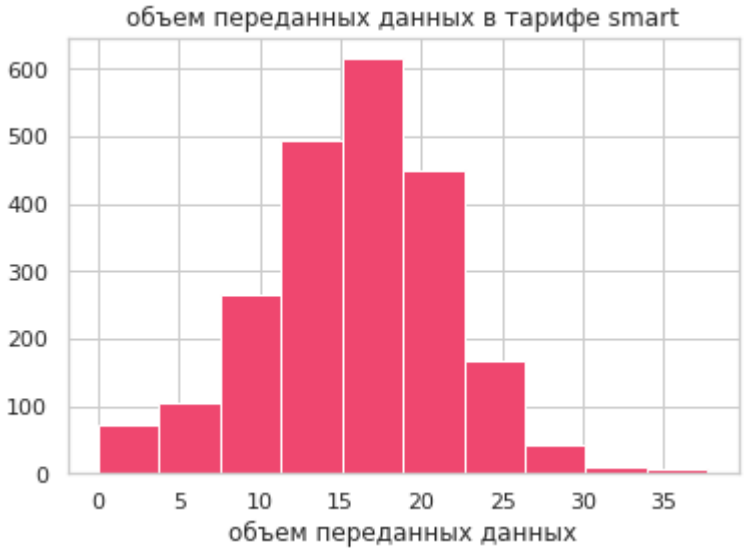

Out[33]:

| | calls_duration |
|---|---|
| **count** | 2,223.00 |
| **mean** | 419.06 |
| **std** | 189.33 |
| **min** | 0.00 |
| **25%** | 285.00 |
| **50%** | 423.00 |
| **75%** | 545.50 |
| **max** | 1,435.00 |

среднее время звонков - 423 мин, наибольшее - 1435 мин

In [34]:
```
# объем переданных данных в тарифе smart
smart_data = telecom_data.query('tariff == "smart"')
smart_data[['gb_used']].hist()
plt.title("объем переданных данных в тарифе smart")
plt.xlabel('объем переданных данных')
```

```
plt.show()
smart_data[['gb_used']].describe()
```




Out[34]:

| | gb_used |
|---|---|
| **count** | 2,222.00 |
| **mean** | 15.86 |
| **std** | 5.74 |
| **min** | 0.00 |
| **25%** | 12.38 |
| **50%** | 16.14 |
| **75%** | 19.61 |
| **max** | 37.71 |

средний объем трафика в тарифе smart-16гб, большинство расходует не более 20гб

**Посчитаем среднее,медиану,стандартное отклонение и дисперсию для тарифов Smart и ultra:**

In [59]:

```
list_of_tariff = ['smart','ultra']
for tariff in list_of_tariff:
    print('Дисперсия', tariff.upper(),':', np.var(telecom_data[telecom_data['tariff']==tariff][
    print('Стандартное отклонение', tariff.upper(),':', np.std(telecom_data[telecom_data['tarif
    print('Среднее',tariff.upper(),':',  telecom_data[telecom_data['tariff']==tariff]['total_co
    print('Медиана',tariff.upper(),':',  telecom_data[telecom_data['tariff']==tariff]['total_co
    print('\n')
```

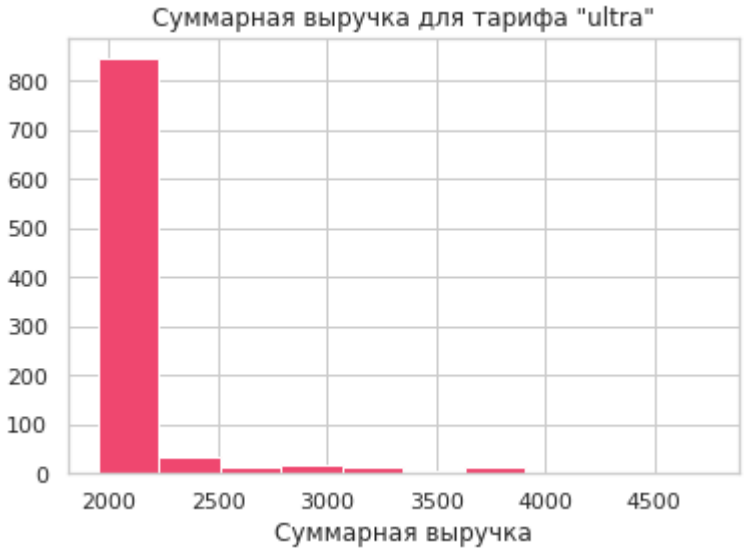
```
Дисперсия SMART : 501945.441
Стандартное отклонение SMART : 708.481
Среднее SMART : 1109.141
Медиана SMART : 796.484


Дисперсия ULTRA : 129077.358
Стандартное отклонение ULTRA : 359.273
Среднее ULTRA : 2061.471
Медиана ULTRA : 1950.0
```

**Суммарная выручка для тарифа "ultra"**

In [36]:

```
ultra_data = telecom_data.query('tariff == "ultra"')
ultra_data[['total_cost']].hist()
plt.title('Суммарная выручка для тарифа "ultra"')
plt.xlabel('Суммарная выручка')
```

```
plt.show()
ultra_data[['total_cost']].describe()
```




Out[36]:

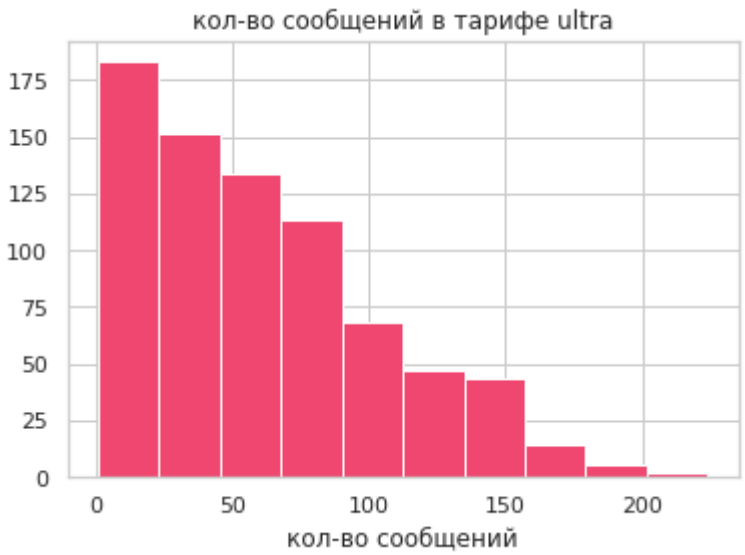
| | total_cost |
|---|---|
| **count** | 951.00 |
| **mean** | 2,061.47 |
| **std** | 359.46 |
| **min** | 1,950.00 |
| **25%** | 1,950.00 |
| **50%** | 1,950.00 |
| **75%** | 1,950.00 |
| **max** | 4,743.31 |

общая выручка тарифа ultra лежит от 1950 до 2200 руб, макс - больше 4000 руб

**среднее кол-во сообщений в тарифе ultra**

In [37]:

```
ultra_data = telecom_data.query('tariff == "ultra"')
ultra_data[['messages_amount']].hist()
plt.title("кол-во сообщений в тарифе ultra")
plt.xlabel('кол-во сообщений')
plt.show()
ultra_data[['messages_amount']].describe()
```




Out[37]:

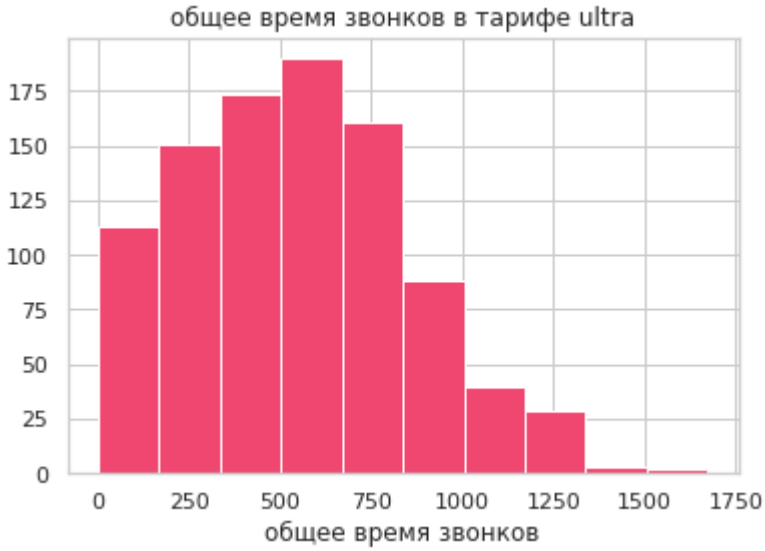
| | messages_amount |
|---|---|

| | messages_amount |
|---|---|
| **count** | 762.00 |
| **mean** | 61.20 |
| **std** | 44.72 |
| **min** | 1.00 |
| **25%** | 25.00 |
| **50%** | 52.00 |
| **75%** | 88.00 |
| **max** | 224.00 |

среднее кол-во сообщений в тарифе ultra -52 шт, макс - 224

**общее время звонков в тарифе ultra**

In [38]:
```
ultra_data = telecom_data.query('tariff == "ultra"')
ultra_data[['calls_duration']].hist()
plt.title("общее время звонков в тарифе ultra")
plt.xlabel('общее время звонков')
plt.show()
ultra_data[['calls_duration']].describe()
```




Out[38]:

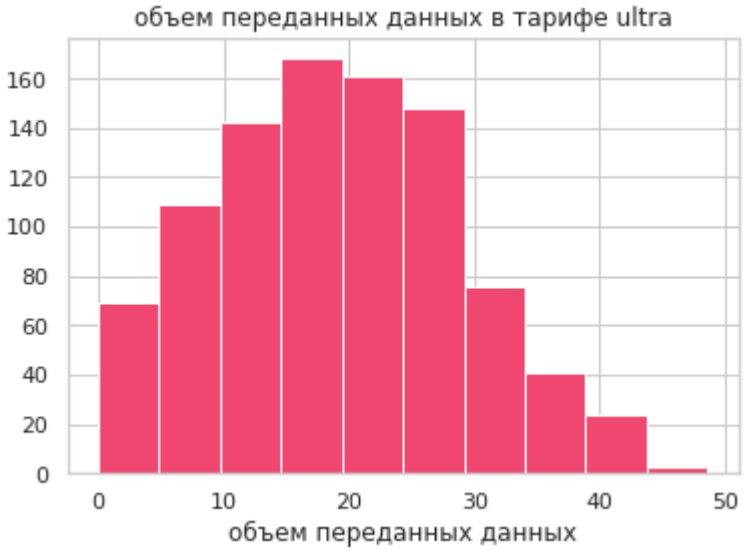
| | calls_duration |
|---|---|
| **count** | 951.00 |
| **mean** | 545.45 |
| **std** | 306.93 |
| **min** | 0.00 |
| **25%** | 310.00 |
| **50%** | 528.00 |
| **75%** | 756.50 |
| **max** | 1,673.00 |

среднее время звонков в тарифе ultra -528 мин, наибольшее - 1673 мин

**объем переданных данных в тарифе ultra**

In [39]:
```
ultra_data = telecom_data.query('tariff == "ultra"')
```

```
ultra_data[['gb_used']].hist()
plt.title("объем переданных данных в тарифе ultra")
plt.xlabel('объем переданных данных')
plt.show()
ultra_data[['gb_used']].describe()
```




Out[39]:

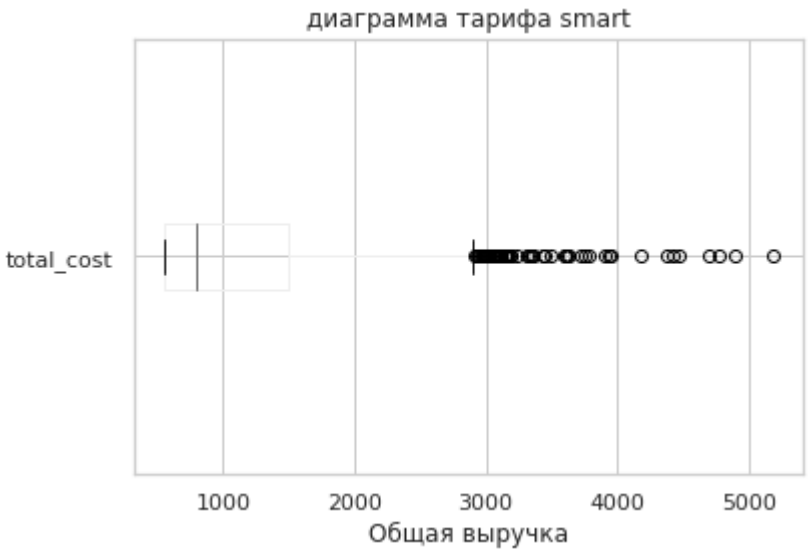
| | gb_used |
|---|---|
| **count** | 941.00 |
| **mean** | 19.23 |
| **std** | 9.75 |
| **min** | 0.00 |
| **25%** | 11.61 |
| **50%** | 18.99 |
| **75%** | 26.26 |
| **max** | 48.62 |

средний трафик в тарифе ultra -19 гб, max 48.625977 гб

**Построим диаграммы размаха для каждого тарифа:**

In [40]:

```
#диаграмма марифа smart
smart_data[['total_cost']].boxplot(vert=False)
plt.title("диаграмма тарифа smart")
plt.xlabel('Общая выручка')
plt.show()
```

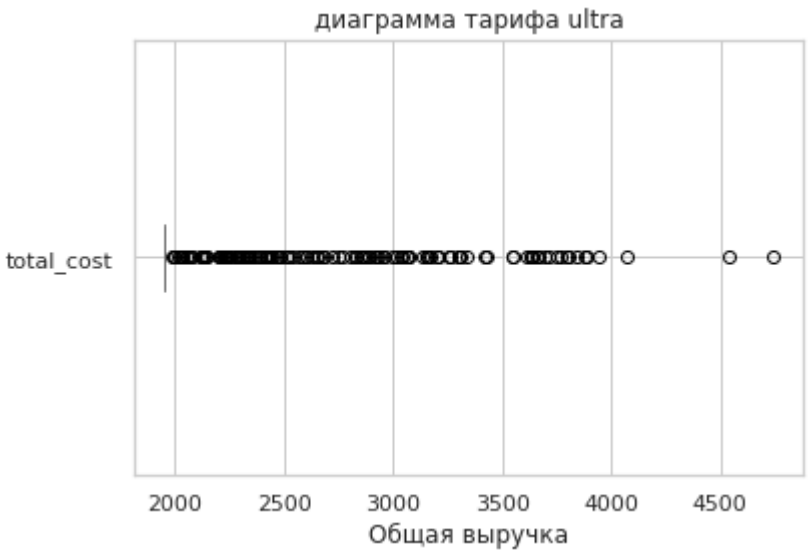


In [41]:
```
parameters = smart_data[['total_cost']].describe().T
parameters
```

Out[41]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **total_cost** | 2,223.00 | 1,109.14 | 708.64 | 550.00 | 550.00 | 796.48 | 1,491.87 | 5,190.02 |

In [42]:
```
#диаграмма тарифа ultra
ultra_data[['total_cost']].boxplot(vert=False)
plt.title("диаграмма тарифа ultra")
plt.xlabel('Общая выручка')
plt.show()
```




In [43]:
```
parameters = ultra_data[['total_cost']].describe().T
parameters
```

Out[43]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **total_cost** | 951.00 | 2,061.47 | 359.46 | 1,950.00 | 1,950.00 | 1,950.00 | 1,950.00 | 4,743.31 |

В тарифе Ultra практически никто не выходит за рамки тарифа 1950,для него нормальное значение - цена тарифа. Пользователи не используют полностью ни минуты разговора, ни кол-во сообщений, ни трафик. Тогда как в тарифе Smart при стоимости 550р в месяц разброс относительно большой, а ст.отклонение равно 709 руб. В тарифе Smart пользователи проговаривают лишние минуты и превышают допустимый трафик, сообщений отправляют меньше заложенных в тариф.

**Проверим гипотезы:**

средняя выручка пользователей тарифов «Ультра» и «Смарт» различаются;
средняя выручка пользователей из Москвы отличается от выручки пользователей из других регионов.

In [58]:
```
#Перед проверкой гипотезы проверим дисперсии выборок
list_of_tariff = ['smart','ultra']
for tariff in list_of_tariff:
    print('Дисперсия', tariff.upper(),':', np.var(telecom_data[telecom_data['tariff']==tariff][
```

```
Дисперсия SMART : 501945.44
Дисперсия ULTRA : 129077.36
```

In [45]:
```
# принимаем нулевую гипотезу, что средние выручки пользователей тарифов «Ультра» и «Смарт» равн
# Альтернативная гипотеза: средние выручки пользователей тарифов «Ультра» и «Смарт» не равны
```

```
alpha = 0.05 # критический уровень статистической значимости
# если p-value окажется меньше него - отвергнем гипотезу
results = st.ttest_ind(ultra_data[['total_cost']], smart_data[['total_cost']])

print('p-значение:', results.pvalue)

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

```
p-значение: [5.98648026e-276]
Отвергаем нулевую гипотезу
```

In [56]:
```
ultra_data[['total_cost']].sum().round(2)#общая выручка тарифа ultra
```

Out[56]:
```
total_cost   1,960,459.13
dtype: float64
```

In [57]:
```
smart_data[['total_cost']].sum().round(2)#общая выручка тарифа smart
```

Out[57]:
```
total_cost   2,465,619.55
dtype: float64
```

Использовали метод "scipy.stats.ttest_ind (array1, array2, equal_var)." - Гипотеза о равенстве средних двух генеральных совокупностей, тк у нас две совокупности. Делаем вывод, что средние выручки пользователей тарифов «Ультра» и «Смарт» отличаются

Теперь создадим 2 новых датафрейма: для москвичей и не-москвичей. Принимаем нулевую гипотезу: средние выручки пользователей Москвы и остальных городов равны. Альтернативная гипотеза: средние выручки пользователей Москвы и остальных городов не равны.

In [48]:
```
#Теперь создадим 2 новых датафрейма: для москвичей и не-москвичей
msc_data = telecom_data.query('city == "Москва"')
not_msc_data = telecom_data.query('city != "Москва"')
results_city = st.ttest_ind(msc_data['total_cost'], not_msc_data['total_cost'])
print('p-значение:', results_city.pvalue)
if results_city.pvalue < alpha:
    print('Отвергаем нулевую гипотезу.')
else:
    print('Не можем отвергнуть нулевую гипотезу.')
```

```
p-значение: 0.18012236974950172
Не можем отвергнуть нулевую гипотезу.
```

Делаем вывод, что средние выручки пользователей Москвы и остальных городов не отличаются.

In [55]:
```
print(msc_data['total_cost'].sum().round(2) - not_msc_data['total_cost'].sum().round(2))
```

```
-2679460.48
```

# Вывод:

Мы определили, что компании «Мегалайн» больше выручки приносят пользователи тарифа smart. Они переплачивают за звонки и интернет.В среднем переплачивают в два раза от стоимости тарифа. Пользователи тарифа ultra не используют полностью свои возможности. Очевидно, тариф ultra более выгоден компании. Средние расходы на связь жителей Москвы не отличаются от средних расходов других городов страны. Суммарная выручка от жителей Москвы меньше суммарной выручки жителей других городов на 2681060 руб.

https://sqlbak.com/blog/jupyter-notebook-markdown-cheatsheet