Проект по определение перспективного тарифа для телеком компании

В данном проекте заказчиком выступает мобильный оператор. Наша цель - провести анализ потребителей двух разных тарифов, узнать как много они пользуются различными услугами оператора, сколько выручки приносят компании помесячно, и в конечном счете определить, какой тариф лучше для оператора. На вход нам дается пять датасетов, каждый из которых содержит информацио по звонкам, сообщениям и интернет-траффику, которые потребили пользователи в течении года, а такжу информацию по самим пользователям и тарифам.

Наши шаги

- Прочитаем Датасеты и выведем общую инфу о них
- Подготовим данные, сделаем один общий датасет с нужной нам инфой
- Проанализируем данные, сравним два тарифа по разным параметрам
- Проверим две гипотезы
- Сформулируем общий вывод

1. Чтение Данных

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
data_calls = pd.read_csv('/datasets/calls.csv')
data_users = pd.read_csv('/datasets/users.csv')
data_tariffs = pd.read_csv('/datasets/tariffs.csv')
data_messages = pd.read_csv('/datasets/messages.csv')
data_internet = pd.read_csv("/datasets/internet.csv")
```

In [2]:
```
print(data_calls.info())
data_calls.head()
```

```
<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
None
```

Out[2]:

	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 [3]:
```
print(data_users.info())
data_users.head()
```

```
<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
None
```

Out[3]:

	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 [4]:
```
print(data_tariffs.info())
data_tariffs.head()
```

```
<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
None
```

Out[4]:

	messages_included	mb_per_month_included	minutes_included	rub_monthly_fee	rub_per_gb	ru
0	50	15360	500	550	200	
1	1000	30720	3000	1950	150	

In [5]:
```
print(data_messages.info())
data_messages.head()
```

```
<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
None
```

Out[5]:

	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 [6]:
```
print(data_internet.info())
data_internet.head()
```

```
<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
None
```

Out[6]:

	Unnamed: 0	id	mb_used	session_date	user_id
0	0	1000_0	112.95	2018-11-25	1000
1	1	1000_1	1052.81	2018-09-07	1000
2	2	1000_2	1197.26	2018-06-25	1000
3	3	1000_3	550.27	2018-08-22	1000
4	4	1000_4	302.56	2018-09-24	1000

Вывод:

Данные прочитаны и выведены на экран, видны некоторые ошибки и неправильные типы данных. В следующем шаге мы все это исправим

2.Подготовка данных

В первую очередь, надо бы перевести все даты к типу данных DateTime.

In [7]:
```
data_calls['call_date'] = pd.to_datetime(data_calls['call_date'], format='%Y-%m-%
data_users['reg_date'] = pd.to_datetime(data_users['reg_date'], format='%Y-%m-%d'
data_messages['message_date'] = pd.to_datetime(data_messages['message_date'], for
data_internet['session_date'] = pd.to_datetime(data_internet['session_date'], for

# Привели даты к формату DateTime
data_calls.duration = data_calls.duration.apply(math.ceil)
data_calls.head(8)
```

Out[7]:

	id	call_date	duration	user_id
0	1000_0	2018-07-25	0	1000
1	1000_1	2018-08-17	0	1000
2	1000_2	2018-06-11	3	1000
3	1000_3	2018-09-21	14	1000
4	1000_4	2018-12-15	6	1000
5	1000_5	2018-11-02	0	1000
6	1000_6	2018-10-18	0	1000
7	1000_7	2018-08-22	19	1000

In [8]:
```
data_internet.drop('Unnamed: 0', axis=1, inplace=True)
data_internet.head()
# Удалили бесполезный столбец в датафрейме по интернету
```

Out[8]:

	id	mb_used	session_date	user_id
0	1000_0	112.95	2018-11-25	1000
1	1000_1	1052.81	2018-09-07	1000
2	1000_2	1197.26	2018-06-25	1000
3	1000_3	550.27	2018-08-22	1000
4	1000_4	302.56	2018-09-24	1000

In [9]:
```
data_users.city.unique() # Нет разных регистров одного горола и других ошибок
```

Out[9]:
```
array(['Краснодар', 'Москва', 'Стерлитамак', 'Новокузнецк',
       'Набережные Челны', 'Ульяновск', 'Челябинск', 'Пермь', 'Тюмень',
       'Самара', 'Санкт-Петербург', 'Иваново', 'Чебоксары', 'Омск', 'Уфа',
       'Томск', 'Чита', 'Мурманск', 'Петрозаводск', 'Тверь',
       'Екатеринбург', 'Махачкала', 'Вологда', 'Череповец', 'Ярославль',
       'Сочи', 'Хабаровск', 'Ставрополь', 'Рязань', 'Улан-Удэ', 'Тула',
       'Саратов', 'Ростов-на-Дону', 'Казань', 'Иркутск', 'Курск',
       'Калуга', 'Якутск', 'Астрахань', 'Химки', 'Владикавказ',
       'Волгоград', 'Саранск', 'Ижевск', 'Новосибирск', 'Брянск',
       'Тольятти', 'Нижний Тагил', 'Орёл', 'Белгород', 'Воронеж',
       'Волжский', 'Курган', 'Барнаул', 'Красноярск', 'Архангельск',
       'Липецк', 'Кемерово', 'Киров', 'Сургут', 'Пенза', 'Магнитогорск',
       'Нижний Новгород', 'Кострома', 'Владивосток', 'Тамбов', 'Балашиха',
       'Севастополь', 'Калининград', 'Оренбург', 'Смоленск',
       'Нижневартовск', 'Владимир', 'Новороссийск', 'Грозный', 'Подольск'],
      dtype=object)
```

In [10]:
```
data_users.tariff.unique() # тут тоже все Ок
```

Out[10]:
```
array(['ultra', 'smart'], dtype=object)
```

In [11]:
```
data_calls['month'] = data_calls['call_date'].dt.month_name()
data_messages['month'] = data_messages['message_date'].dt.month_name()
data_internet['month'] = data_internet.session_date.dt.month_name()
```

In [12]:
```
# Посчитаем необходимые величины для каждого пользователя. Причем будем считать б
# и с нулевой продолжительностью
df_calls = data_calls.pivot_table(index=['user_id', 'month'], values = ['duration
df_calls.columns = [('sum', 'duration'), ('count', 'number_of_calls')]
df_calls
```

Out[12]:

		(sum, duration)	(count, number_of_calls)
user_id	**month**		
1000	August	408	52
	December	333	46
	July	340	47
	June	172	43
	May	159	22
...	...	...	...
1498	September	471	59
1499	December	492	69
	November	612	74
	October	449	68
	September	70	9

3174 rows × 2 columns

In [13]:
```
df_messages = data_messages.pivot_table(index=['user_id', 'month'], values = ['id
df_messages.columns = [('count', 'messages')]
df_messages # Аналагично с другими характеристиками
```

Out[13]:

		(count, messages)
user_id	**month**	
1000	August	81
	December	70
	July	75
	June	60
	May	22
...	...	...
1498	September	44
1499	December	66
	November	59
	October	48
	September	11

2717 rows × 1 columns

In [14]:
```
df_internet = data_internet.pivot_table(index=['user_id', 'month'], values = ['mb

df_internet[('sum', 'mb_used')] = df_internet[('sum', 'mb_used')].apply(math.ceil
df_internet
```

Out[14]:

		sum
		mb_used
user_id	month	
1000	August	14056
	December	9818
	July	14004
	June	23234
	May	2254
...	...	...
1498	September	22135
1499	December	13056
	November	17964
	October	17789
	September	1846

3203 rows × 1 columns

In [15]:
```
df_finished = df_internet.join([df_messages,df_calls],  how='outer') #объединим 6

df_finished.reset_index()
```

Out[15]:

	user_id	month	sum mb_used	count messages	sum duration	count number_of_calls
0	1000	August	14056.0	81.0	408.0	52.0
1	1000	December	9818.0	70.0	333.0	46.0
2	1000	July	14004.0	75.0	340.0	47.0
3	1000	June	23234.0	60.0	172.0	43.0
4	1000	May	2254.0	22.0	159.0	22.0
...	...	...	...	...	...	...
3209	1498	September	22135.0	44.0	471.0	59.0
3210	1499	December	13056.0	66.0	492.0	69.0
3211	1499	November	17964.0	59.0	612.0	74.0
3212	1499	October	17789.0	48.0	449.0	68.0
3213	1499	September	1846.0	11.0	70.0	9.0

3214 rows × 6 columns

In [16]:
```
df2 = df_finished.reset_index().set_axis(['user_id', 'month', 'mb_used', 'message
                                         axis=1, inplace=False).merge(data_users, on='u
df2.drop([ 'first_name', 'last_name', 'churn_date', 'age'], axis=1, inplace=True)
df2 = df2.set_index(['user_id', 'month'])
df2.columns = ['mb_used', 'count_messg', 'calls_duration', 'count_calls', 'city',

df2 # Добавили инфу по пользователям и привели столбцы к нормальному виду
```

Out[16]:

user_id	month	mb_used	count_messg	calls_duration	count_calls	city	reg_date	ta
	August	14056.0	81.0	408.0	52.0	Краснодар	2018-05-25	u
	December	9818.0	70.0	333.0	46.0	Краснодар	2018-05-25	u
1000	July	14004.0	75.0	340.0	47.0	Краснодар	2018-05-25	u
	June	23234.0	60.0	172.0	43.0	Краснодар	2018-05-25	u
	May	2254.0	22.0	159.0	22.0	Краснодар	2018-05-25	u
...	...	...	...	...	...	...	...	
1498	September	22135.0	44.0	471.0	59.0	Владикавказ	2018-07-19	sm
	December	13056.0	66.0	492.0	69.0	Пермь	2018-09-27	sm
	November	17964.0	59.0	612.0	74.0	Пермь	2018-09-27	sm
1499								
	October	17789.0	48.0	449.0	68.0	Пермь	2018-09-27	sm
	September	1846.0	11.0	70.0	9.0	Пермь	2018-09-27	sm

3214 rows × 7 columns

In [17]:
```
df_finished.reset_index().set_axis(['user_id', 'month', 'mb_used', 'message', 'du
                                    axis=1, inplace=False).merge(data_users, on='u
```

Out[17]:

	user_id	month	mb_used	message	duration	number_of_calls	age	churn_date	
0	1000	August	14056.0	81.0	408.0	52.0	52	NaN	Крас
1	1000	December	9818.0	70.0	333.0	46.0	52	NaN	Крас
2	1000	July	14004.0	75.0	340.0	47.0	52	NaN	Крас
3	1000	June	23234.0	60.0	172.0	43.0	52	NaN	Крас
4	1000	May	2254.0	22.0	159.0	22.0	52	NaN	Крас
...	...	...	...	...	...	...	...	...	
3209	1498	September	22135.0	44.0	471.0	59.0	68	2018-10-25	Владик
3210	1499	December	13056.0	66.0	492.0	69.0	35	NaN	Г
3211	1499	November	17964.0	59.0	612.0	74.0	35	NaN	Г
3212	1499	October	17789.0	48.0	449.0	68.0	35	NaN	Г
3213	1499	September	1846.0	11.0	70.0	9.0	35	NaN	Г

3214 rows × 13 columns

In [18]:
```
def add_revenue(row):
    if row['tariff'] == 'ultra':
        left_min = row['calls_duration'] - 3000
        left_mes = row['count_messg'] - 1000
        left_mb = row['mb_used'] - 30*1024
        s = 1950
        if left_min > 0:
            s = s + left_min
        if left_mes > 0:
            s = s + left_mes
        if left_mb > 0:
            s = s + 150*math.ceil(left_mb / 1024)
    elif row['tariff'] == 'smart':
        left_min = row['calls_duration'] - 500
        left_mes = row['count_messg'] - 50
        left_mb = row['mb_used'] - 15*1024
        s = 550
        if left_min > 0:
            s = s + 3*left_min
        if left_mes > 0:
            s = s + 3*left_mes
        if left_mb > 0:
            s = s + 200*math.ceil(left_mb / 1024)
    return s

df2['revenue'] = df2.apply(add_revenue, axis=1)
df2.head(100)
```

Out[18]:

user_id	month	mb_used	count_messg	calls_duration	count_calls	city	reg_date	tarif
	August	14056.0	81.0	408.0	52.0	Краснодар	2018-05-25	ultra
	December	9818.0	70.0	333.0	46.0	Краснодар	2018-05-25	ultra
1000	July	14004.0	75.0	340.0	47.0	Краснодар	2018-05-25	ultra
	June	23234.0	60.0	172.0	43.0	Краснодар	2018-05-25	ultra
	May	2254.0	22.0	159.0	22.0	Краснодар	2018-05-25	ultra
...	...	...	...	...	...	...	...	...
	November	22518.0	25.0	206.0	25.0	Санкт-Петербург	2018-03-28	smar
1012	October	29072.0	13.0	451.0	48.0	Санкт-Петербург	2018-03-28	smar
	September	19381.0	14.0	230.0	30.0	Санкт-Петербург	2018-03-28	smar
	December	20738.0	25.0	567.0	90.0	Иваново	2018-11-14	ultra
1013								
	November	9781.0	17.0	369.0	52.0	Иваново	2018-11-14	ultra

100 rows × 8 columns

Вывод:

Я закончил с созданием таблицы, теперь для каждого юзера по месцам отображены использованные им услуги оператора, а также посчитана сумму, которую каждый пользователь платит оператору в каждом месяце.

3.Анализ

In [19]:
```
print('Средние значения характеристик по тарифу "Ultra" и "Smart"')
df2.pivot_table(index='tariff', values=['calls_duration','count_messg',
                                        'mb_used', 'revenue'], aggfunc='median') #
```

```
Средние значения характеристик по тарифу "Ultra" и "Smart"
```

Out[19]:

tariff	calls_duration	count_messg	mb_used	revenue
smart	423.0	34.0	16508.0	1023.0
ultra	528.0	51.0	19395.0	1950.0

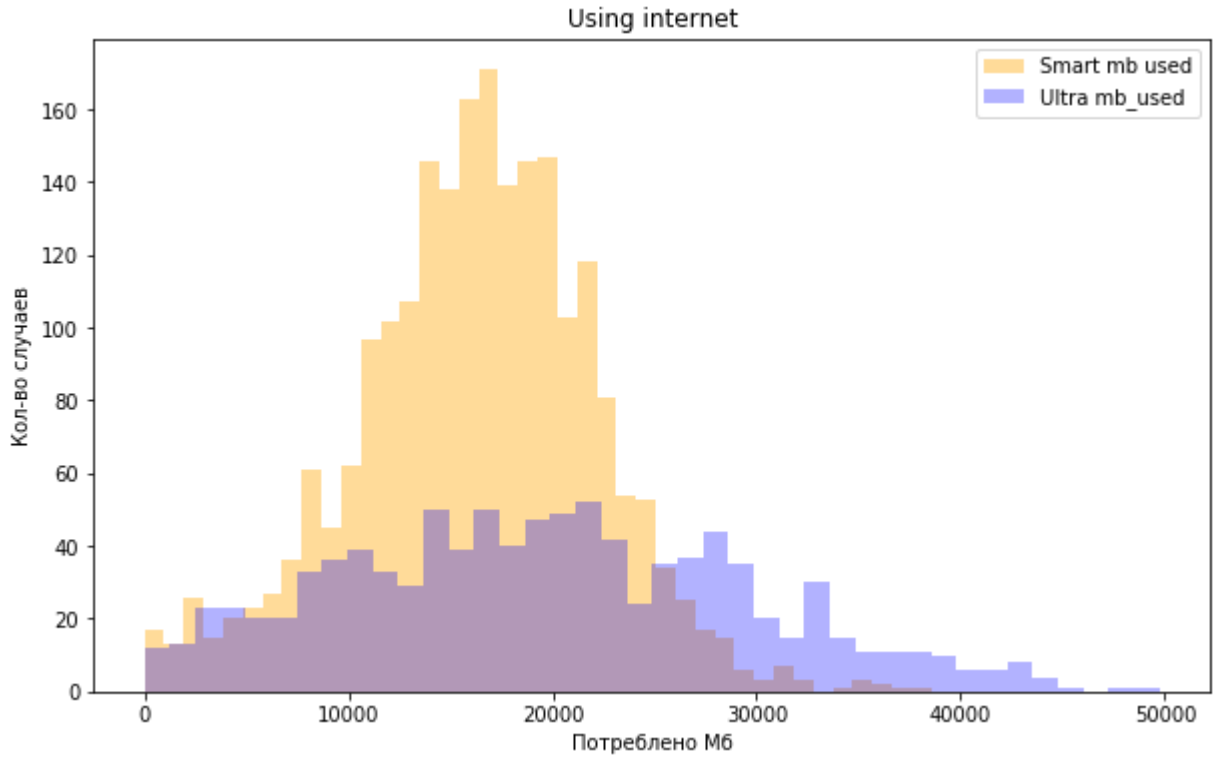
In [20]:
```
Dispersia_ultra = np.var(df2.query('tariff=="ultra"'), ddof=1)
print('Дисперсия по тарифу Ultra\n')
print(Dispersia_ultra)
Std_ult = np.std(df2.query('tariff=="ultra"'), ddof=1)
print('----------------------\n Квадрати́чное отклоне́ние по тарифу Ultra\n')
print(Std_ult) # Дисперсия и сигма по интернету, звонкам и сообщениям для тарифа
```

```
Дисперсия по тарифу Ultra

mb_used           9.886460e+07
count_messg       2.109659e+03
calls_duration    9.420342e+04
count_calls       1.798044e+03
revenue           1.415167e+05
dtype: float64
----------------------
 Квадрати́чное отклоне́ние по тарифу Ultra

mb_used           9943.067706
count_messg         45.931026
calls_duration     306.925754
count_calls         42.403348
revenue            376.187114
dtype: float64
```

In [21]:
```
# Аналагично для тарифа смарт
Dispersia_smart = np.var(df2.query('tariff=="smart"'), ddof=1)
print('Дисперсия по тарифу Smart\n')
print(Dispersia_smart)
Std_smr = np.std(df2.query('tariff=="smart"'), ddof=1)
print('----------------------\n Квадрати́чное отклоне́ние по тарифу Smart\n')
print(Std_smr)
```

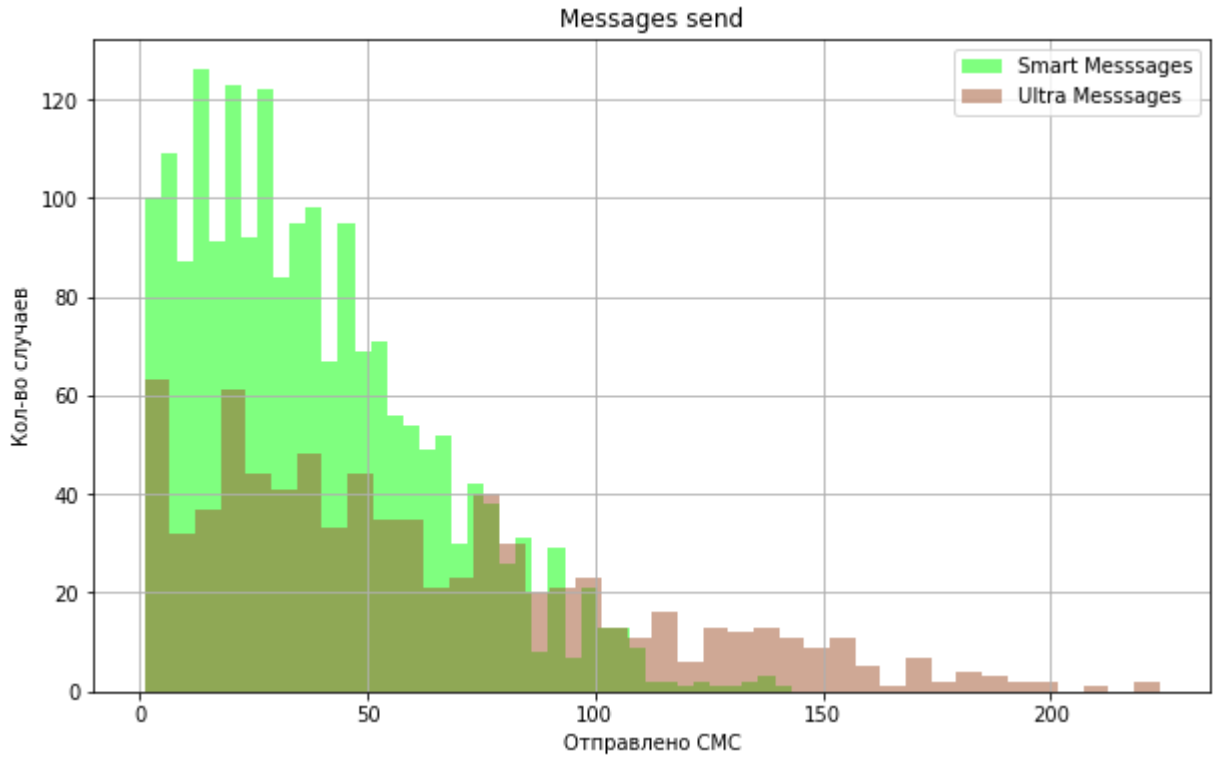
```
Дисперсия по тарифу Smart

mb_used           3.436001e+07
count_messg       7.175942e+02
calls_duration    3.584419e+04
count_calls       6.654388e+02
revenue           6.697857e+05
dtype: float64
----------------------
 Квадрати́чное отклоне́ние по тарифу Smart

mb_used           5861.741292
count_messg         26.787949
calls_duration     189.325618
count_calls         25.796101
revenue            818.404367
dtype: float64
```

In [22]:
```
plt.figure(figsize=(10, 6))
plt.hist(x=df2.query('tariff=="smart"').mb_used, bins=40,alpha=0.4,facecolor='ora
plt.hist(x=df2.query('tariff=="ultra"').mb_used, bins=40,alpha=0.3,facecolor='blu
plt.title('Using internet')
plt.xlabel("Потреблено Мб")
plt.ylabel("Кол-во случаев")
plt.legend();
plt.show() # Гистограммы потребления интернет-траффика
```



Видно, что если пик потребления у пользователей Смарт приходится на интервал 12000-20000 Мб, то у пользователей тарифа Ультра потребление более равномерное, без ярко-выраженных пиков и находится в промежутке от 15000 до 27000 Мб примерно. В среднем, в нашей выборке все же видно, что в среднем пользователи тарифы Ультра потребляют несколько больше Мб, хотя и сигма(стандартное отклонение) у юзеров данного тарифа больше. Распределение в обоих случаях нормальное.

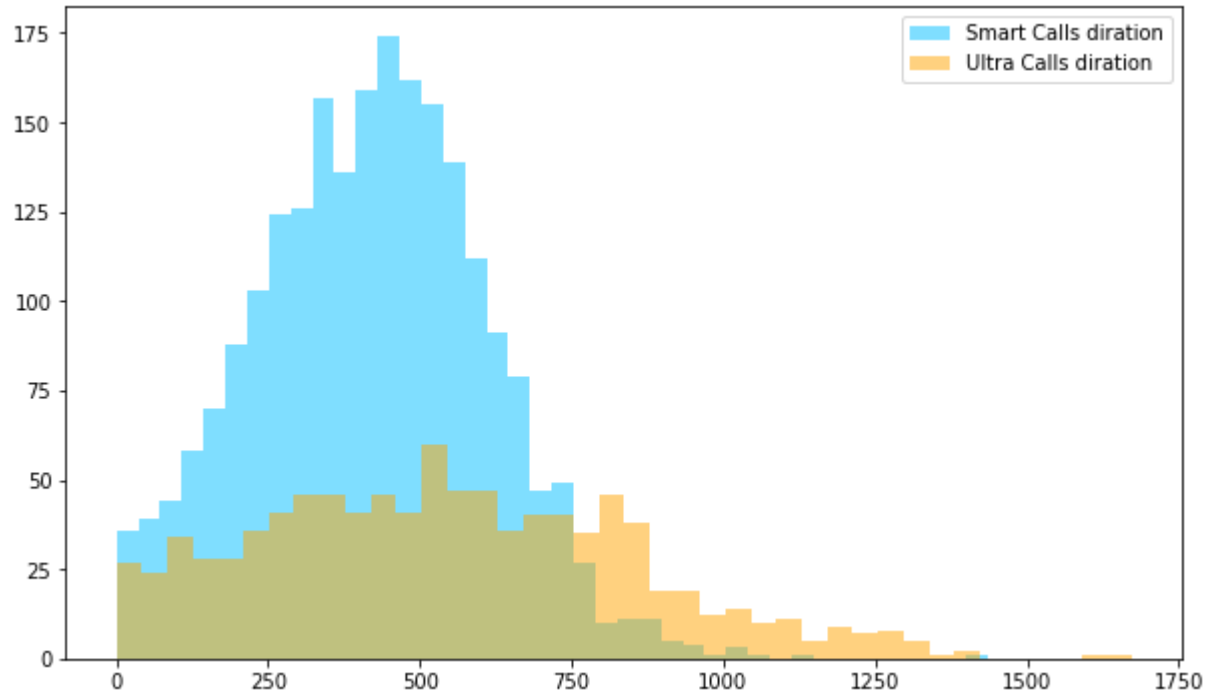
In [23]:
```
plt.figure(figsize=(10, 6))

plt.hist(x=df2.query('tariff=="smart"').count_messg, bins=40,alpha=0.5,facecolor=
plt.hist(x=df2.query('tariff=="ultra"').count_messg, bins=40,alpha=0.5,facecolor=

plt.grid()
plt.title('Messages send')
plt.xlabel("Отправлено СМС")
plt.ylabel("Кол-во случаев")
plt.legend();
plt.show() # Hists для кол-ва сообщей
```



Опять же, в обоих группах мы видим Пуассоновское распределение в отправленных в месяц сообщениях. Надо отметить, что хоть и в обоих группах пик приходится на промежуток 0-40 сообщений, но в тарифе ультра также есть случаи, когда юзеры отправляли больше 100-150

сообщение, что сдвигает среднее значения сильно вправо по сравнению с тарифом Smart. Поэтому и медианное значения у тарифа ультра по данному коказателю выше, и сигма у тарифа Ультра больше.

In [24]:
```
plt.figure(figsize=(10, 6))
plt.hist(x=df2.query('tariff=="smart"').calls_duration, bins=40,alpha=0.5,faceco
plt.hist(x=df2.query('tariff=="ultra"').calls_duration, bins=40,alpha=0.5,faceco
plt.legend();
plt.show() # Гистограммы для суммарной продолжительности звонков
```



По звонкам ситуация интереснее. Тут опять же видно, что у тарифа смарт наибольшее число юзеров говорит в среднем 400-500 минут, виден яркий пик в распределение в данном интервале. Кстати, распределение похоже на нормальное. В тарифе Ультра все немного иначе. Тут длительность разговора распределена более-менее равномено в промежутке от 300 до 800 минут в месяц, причем есть значения и больше тысячи, что явно демонстрирует нам о том, что пользователи тарифа Ультра говорят больше пользователей тарифа Смарт. Но опять же, сигма у тарифа Ультра больше, что говорит о бОльшей волатильность относительно среднего значения

Вывод:

В общем можно сказать, что в среднем юзеры тарифа Ultra потребляют больше услуг оператора, но и стандартное отклонение в тарифе Ультра больше по каждому из параметров, нежели у выборки по тарифу Smart

4. Проверка гипотез

Проверим нулевую гипотезу, что средние двух выборок одинаковы, а именно выручка пользователей тарифа Смарт и Ультра. Тогда альтернативная гипотеза заключается в том, что средние разные

In [25]:
```
# Блок добавлен во время доработки
import scipy.stats as sp


# Проведем проверку дисперсий наших выборок, чтобы понять, считать ли параметр eq
stats, p1 = sp.levene(df2.query('tariff=="smart"').revenue, df2.query('tariff=="u

print(p1)
```

```
1.8783148636310733e-122
```

Так как Дисперсии выборок не равны (выявлено при помощи теста Левене, p-value очень маленькое), то изменил параметро equal_var в дальнейшем на False

In [28]:
```
alpha = 0.05
results = sp.ttest_ind(
df2.query('tariff=="smart"').revenue,
df2.query('tariff=="ultra"').revenue, equal_var=bool(0))

print(f'p-value for revenue = {results.pvalue}')
```

```
p-value for revenue = 4.2606313931076085e-250
```

Вывод:

для проверки гиппотезы о том, что выручка оператора от пользователей разных тарифов разная. Для этого я сформулировал гупотезу, что они равны, и альтернативную,что не равны. В итоге все пришло к тому, что они не равны, так как я, использовав t-тест для выборок смарт и ультра, посчитал p-value, и оно оказалось равно 0. Очевидно, что выручка от юзеров с тарифа ультра больше

Учел вышеперечисленные поправки, исправил параметр equal_var на False, p-value также осталось близко к 0, поэтому отвергаем нулевую гипотезу

Проверим гипотезу, что выручка от пользователей Мск и других городов разные. Для этого сформулирую нулевую гипотезу, что они равны, и альтернативную гипотезу, что не равны.

In [29]:
```
results2 = sp.ttest_ind(
df2.query('city=="Москва"').revenue,
df2.query('city!="Москва"').revenue, equal_var=False)

print(f'p-value = {results2.pvalue}')
print('Средняя оплата по москве за тариф:',df2.query('city=="Москва"').revenue.me
      '\nСредняя оплата по другим городам за тариф:',df2.query('~(city=="Москва")
```

```
p-value = 0.5257376663729298
Средняя оплата по москве за тариф: 1546.7135842880523
Средняя оплата по другим городам за тариф: 1524.9362274298885
```

Вывод:

~~Видно, что также для пользователей Москвы и других городов выручки различные, так как по критерию pvalue мы отвергаем нулевую гипотезу. Также убедились, что Московские юзеры приносят компании больше денег, нежели другие города.~~

После исправления недочета функции add_revenue и из-за других исправлений было выявлено, что p-value при сравнение выручки по городам стало значительно больше и равняется 0,53. Также средние величины по городам примерно схожи, поэтому мы не отвергаем нулевую гипотезу

5. Общий вывод

В итоге мы исследовали данные по двум тарифам, в результате чего пришли к выводу, что пользователи тарифа Ultra в среднем потребляют больше трафика, разговаривают суммарно дольше, нежели пользователи тарифа Smart. Также мы проверили, что пользователи тарифа Ultra приносят оператору больше выручки, нежели пользователи тарифа Смарт. И вконце мы установили, что в среднем пользователь Мск приносит столько же выручки, сколько и пользователь другого города.

Изменил тут лишь последнее предложение

In [ ]: