Анализ использования услуг компании «Мегалайн»

Описание задачи проекта: анализ использования услуг тарифов «Смарт» и «Ультра» на выборке клиентов.

Данные: таблицы

- calls - информация о звонках
- internet -информация об интернет-сессиях
- messages - информация о сообщениях
- tariffs - информация о тарифах
- users - информация о пользователях

Описание тарифов:

Тариф «Смарт»

- Ежемесячная плата: 550 рублей
- Включено 500 минут разговора, 50 сообщений и 15 Гб интернет-трафика
- Стоимость услуг сверх тарифного пакета:
  - минута разговора: 3 рубля
  - сообщение: 3 рубля
  - 1 Гб интернет-трафика: 200 рублей

Тариф «Ультра»

- Ежемесячная плата: 1950 рублей
- Включено 3000 минут разговора, 1000 сообщений и 30 Гб интернет-трафика
- Стоимость услуг сверх тарифного пакета:
  - минута разговора: 1 рубль
  - сообщение: 1 рубль
  - 1 Гб интернет-трафика: 150 рублей

Примечание:

«Мегалайн» всегда округляет секунды до минут, а мегабайты — до гигабайт. Каждый звонок округляется отдельно: даже если он длился всего 1 секунду, будет засчитан как 1 минута. Для веб-трафика отдельные сессии не считаются. Вместо этого общая сумма за месяц округляется в бóльшую сторону. Если абонент использует 1025 мегабайт в этом месяце, с него возьмут плату за 2 гигабайта.

Навыки и инструменты, применённые в работе:

- **Предобработка данных:** проверка на пропуски, замена типов данных, округление значений в большую сторону, добавление столбцов с номером месяца, переименование столбцов, замена пропусков.
- **Расчёты и исследование данных:** сводные таблицы, объединение промежуточных таблиц, функция расчёта помесячной выручки с каждого пользователя, расчёт дисперсии и стандартного отклонения, коэффициент корреляции, проверка гипотез (с формулировкой нулевой и альтернативной гипотез).
- **Графики:** гистограммы (распределения расчётных параметров), диаграмма рассеяния

Результаты исследования

Анализ использования тарифных планов показал следующие результаты:

Параметр	SMART	ULTRA	Доп.информация, комментарии
Абонентская плата	550	1950	рублей в месяц
Размер выборки	2229	985	количество клиентов
Доход от абон.платы в месяц	1 225 950 руб	1 920 750 руб	при условии использования тарифа всеми клиентами выборки
Общая ср.продолжительность звонков	420 мин	528 мин	расчёт по ср. значениям за месяц по каждому пользователю
Диапазон продолжительности звонков на клиента в месяц	180 - 700 мин/мес	180 - 1000 мин/мес	границы, в которых находится основная часть распределения
Общее ср.количество сообщений	33 шт	55 шт	расчёт по ср. значениям за месяц по каждому пользователю
Диапазон кол-во сообщений на клиента в месяц	до 60 шт	до 100 шт	границы, в которых находится основная часть

			распределени
Общий ср.объем трафика	17 Гб	20 Гб	расчёт по ср. значениям за месяц по каждому пользователю
Диапазон объема трафика на клиента в месяц	10-23 Гб	5-35 Гб	границы, в которых находится основная часть распределения

По рассчитанным показателям видим, что:

- **по продолжительности звонков** клиенты тарифа СМАРТ за частую выходят за предоставленный в месяц пакет, тогда как пользователи УЛЬРА с большим запасом укладываются в пакет 3000 минут;
- **по использованию СМС** пользователи СМАРТ также часто выходят за свой пакет, тогда как на УЛЬТРА в большинстве случаев используется только десятая часть пакета;
- **по использованию интернет-трафика** пользователи обоих тарифов выходят за предоставленные пакеты, но в случае тарифа УЛЬТРА это происходит гораздо реже.

Выручка компании только от абонентской платы тарифа Ультра в 1,6 раза выше, чем от тарифа Смар (при оплате только абонентской платы и условии пользования тарифом тем, кол-во клиентов, которое указано в выборке).

Для расчёта дополнительной выручки компании от тарифа Смарт необходимо учитывать, что по всем показателям зачастую пользователи превышают предоставленные пакеты и оплачивают услуги сверх тарифа. (*Подобный расчёт может быть потенциальным развитием данного проекта.*)

В результате проверки гипотезы не подтвердилось предположение о различях в ср.выручки пользователей из Москвы и пользователей из других регионов.

Что было сделано в ходе исследования:

- проверены исходные данные,
- приведены к соответсвующим форматам,
- данные по продолжительности звонков округлены до минут в большую сторону, а интернет-трафик округлен до гигабайтов
- данные из исходных таблиц объединены для рассчётов.

Рассчитаны по каждому клиенту:

- количество сделанных звонков и израсходованных минут разговора по месяцам;
- количество отправленных сообщений по месяцам;
- объем израсходованного интернет-трафика по месяцам;

а также рассчитано:

- количество минут разговора,сообщений и объём интернет-трафика требуется пользователям каждого тарифа в месяц;
- посчитано среднее количество, дисперсия и стандартное отклонение;
- подтверждена гипотеза о том, что средняя выручка пользователей тарифов «Ультра» и «Смарт» различаются;
- опровергнута гипотеза о том, что средняя выручка пользователей из Москвы отличается от выручки пользователей из других регионов.

Изучение исходных данных

Откроем файл с данными и изучим общую информацию об исходных таблицах.

Импортируем библиотеку pandas и откроем файлы с помощью функции read_csv()

In [2]:
```
import pandas as pd
import numpy as np
from scipy import stats as st
import matplotlib.pyplot as plt
```

Таблица calls

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

Согласно документации к данным, таблица calls (информация о звонках):

- id — уникальный номер звонка
- call_date — дата звонка
- duration — длительность звонка в минутах
- user_id — идентификатор пользователя, сделавшего звонок

```
<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
```

Видим по общей информации из таблицы calls , что пропусков в ней нет.

Сделаем замену типа данных для столбца call_date с типа object на тип datetime, чтобы сделать дату удобной для дальнейшей работы с данными:

In [5]:
```
calls['call_date']= pd.to_datetime(calls['call_date'], format = '%Y-%m-%d')
```

Мы знаем, что компания всегда округляет секунды до минут, поэтому округлим значения в столбце duration в большую сторону и изменим на тип 'int', чтобы убрать нули после запятой.

Проверим результат:

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

```
<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  datetime64[ns]
 2   duration   202607 non-null  int32         
 3   user_id    202607 non-null  int64         
dtypes: datetime64[ns](1), int32(1), int64(1), object(1)
memory usage: 5.4+ MB
```

Таблица internet

Out[9]:

	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

Согласно документации к данным, 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

```
<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
```

Видим по общей информации из таблицы internet, что пропусков в ней нет.

Сделаем замену типа данных для столбца session_date с типа object на тип datetime, чтобы сделать дату удобной для дальнейшей работы с данными:

In [11]:
```
internet['session_date']= pd.to_datetime(internet['session_date'], format = '%Y-%m-%d')
```

```
<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  datetime64[ns]
 4   user_id       149396 non-null  int64         
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 5.7+ MB
```

Таблица messages

Out[13]:

	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

Согласно документации к данным, таблица messages (информация о сообщениях):

- id — уникальный номер сообщения
- message_date — дата сообщения
- user_id — идентификатор пользователя, отправившего сообщение

```
<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
```

Для таблицы messages требуется только изменение типа данных для столбца message_date:

In [15]:
```
messages['message_date']= pd.to_datetime(messages['message_date'], format = '%Y-%m-%d')
```

```
messages.info()
```

```
<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  datetime64[ns]
 2   user_id       123036 non-null  int64         
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 2.8+ MB
```

Таблица tariffs

Out[16]:

	messages_included	mb_per_month_included	minutes_included	rub_monthly_fee	rub_per_gb	rub_per_message	rub_per_minute	tariff_name
0	50	15360	500	550	200	3	3	smart
1	1000	30720	3000	1950	150	1	1	ultra

Согласно документации к данным - 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)

```
<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
```

Заменты типов данных и исправления ошибок не требуется, но приведём значения в столбце mb_per_month_included к гигабайтам и переименуем столбец:

Out[18]:

	messages_included	GB_per_month_included	minutes_included	rub_monthly_fee	rub_per_gb	rub_per_message	rub_per_minute	tariff
0	50	15.0	500	550	200	3	3	smart
1	1000	30.0	3000	1950	150	1	1	ultra

Таблица users

Out[19]:

	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

Согласно документации к данным - 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 план

```
<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
```

Для таблицы users требуется только изменение типа данных для столбцов churn_date и reg_date:

In [21]:
```
columns = ['churn_date', 'reg_date']
for i in columns:
    users[i]= pd.to_datetime(users[i], format = '%Y-%m-%d')
```

Out[22]:

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

```
<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     datetime64[ns]
 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    datetime64[ns]
 7   tariff      500 non-null    object        
dtypes: datetime64[ns](2), int64(2), object(4)
memory usage: 31.4+ KB
```

Расчет данных о пользовании услугами и доходе с клиента

Для рассчёта данных о пользователях по месяцам предварительно добавим в исходные таблицы столбцы с номером месяца:

In [24]:
```
calls['month'] = calls['call_date'].dt.month
internet['month'] = internet['session_date'].dt.month
messages['month'] = messages['message_date'].dt.month
```

In [25]:
```
tariffs_name = users.loc[:,['user_id','tariff']]
```

In [26]:
```
calls=calls.merge(tariffs_name, on = ['user_id'])
internet=internet.merge(tariffs_name, on = ['user_id'])
messages=messages.merge(tariffs_name, on = ['user_id'])
```

Рассчитаем количество сделанных звонков и израсходованных минут разговора по месяцам

Out[27]:

	user_id	tariff	month	amount_calls	sum_duration_calls
0	1000	ultra	5	22	159
1	1000	ultra	6	43	172
2	1000	ultra	7	47	340
3	1000	ultra	8	52	408
4	1000	ultra	9	58	466
...	...	...	...	...	...
3169	1498	smart	10	41	247
3170	1499	smart	9	9	70
3171	1499	smart	10	68	449
3172	1499	smart	11	74	612
3173	1499	smart	12	69	492

3174 rows × 5 columns

Рассчитаем количество отправленных сообщений по месяцам по каждому пользователю:

Out[28]:

	user_id	tariff	month	amount_sms
0	1000	ultra	5	22
1	1000	ultra	6	60
2	1000	ultra	7	75
3	1000	ultra	8	81
4	1000	ultra	9	57

Рассчитаем объем израсходованного интернет-трафика по месяцам по каждому пользователю:

Out[29]:

	Unnamed: 0	id	mb_used	session_date	user_id	month	tariff
0	0	1000_0	112.95	2018-11-25	1000	11	ultra
1	1	1000_1	1052.81	2018-09-07	1000	9	ultra
2	2	1000_2	1197.26	2018-06-25	1000	6	ultra
3	3	1000_3	550.27	2018-08-22	1000	8	ultra
4	4	1000_4	302.56	2018-09-24	1000	9	ultra

Out[30]:

	user_id	tariff	month	sum_trafic_mb
0	1000	ultra	5	2253.49
1	1000	ultra	6	23233.77
2	1000	ultra	7	14003.64
3	1000	ultra	8	14055.93
4	1000	ultra	9	14568.91

Приведём значения в столбце sum_trafic_mb до гигабайт,округлим в большую сторону и изменим на тип 'int', чтобы убрать нули после запятой, после чего переименуем столбец на sum_trafic_gb:

Out[31]:

	user_id	tariff	month	sum_trafic_gb
0	1000	ultra	5	3
1	1000	ultra	6	23

2	1000	ultra	7	14
3	1000	ultra	8	14
4	1000	ultra	9	15

Объединим сделанные расчёты о количестве звоноков, смс, интернет-трафика в одну таблицу с данными о каждом пользователе:

In [33]:
```
data=data.merge(internet_month, on = ['user_id', 'tariff','month'], how='outer').reset_index()
data.head()
```

Out[33]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb
0	1000	ultra	5	22.0	159.0	22.0	3.0
1	1000	ultra	6	43.0	172.0	60.0	23.0
2	1000	ultra	7	47.0	340.0	75.0	14.0
3	1000	ultra	8	52.0	408.0	81.0	14.0
4	1000	ultra	9	58.0	466.0	57.0	15.0

В полученной таблице заменим NaN на нулевые значения:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3214 entries, 0 to 3213
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   user_id             3214 non-null   int64  
 1   tariff              3214 non-null   object 
 2   month               3214 non-null   int64  
 3   amount_calls        3214 non-null   float64
 4   sum_duration_calls  3214 non-null   float64
 5   amount_sms          3214 non-null   float64
 6   sum_trafic_gb       3214 non-null   float64
dtypes: float64(4), int64(2), object(1)
memory usage: 175.9+ KB
```

Рассчитаем помесячную выручку с каждого пользователя.

In [35]:
```
data = data.merge(tariffs, on='tariff', how='outer')
data.head()
```

Out[35]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

In [36]:
```
data['calls_over']= (data['sum_duration_calls']- data['minutes_included'])
data['sms_over']= (data['amount_sms']- data['messages_included'])
data['GB_over']= (data['sum_trafic_gb']- data['GB_per_month_included'])
data.head()
```

Out[36]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

In [37]:
```
def x(row):
    calls_over = row['calls_over']
```

```
    pay_calls =row['rub_per_minute']
    if calls_over < 0:
        return 0
    else:
        return(calls_over* pay_calls)
data['calls_over'] = data.apply(x, axis=1).reset_index(drop=True)
data.head()
```

Out[37]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

In [38]:

```
def y(row):
    sms_over = row['sms_over']
    pay_sms = row['rub_per_message']
    if sms_over < 0:
        return 0
    else:
        return(sms_over* pay_sms)
data['sms_over'] =  data.apply(y, axis=1).reset_index(drop=True)
data.head()
```

Out[38]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

In [39]:

```
def z(row):
    GB_over = row['GB_over']
    pay_GB = row['rub_per_gb']
    if GB_over < 0:
        return 0
    else:
        return(GB_over*pay_GB)
data['GB_over'] = data.apply(z, axis=1).reset_index(drop=True)
data.head()
```

Out[39]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

In [40]:

```
data['pay_month']=data['calls_over']+data['sms_over']+data['GB_over']+ data['rub_monthly_fee']
data.head()
```

Out[40]:

	user_id	tariff	month	amount_calls	sum_duration_calls	amount_sms	sum_trafic_gb	messages_included	GB_per_month_included	minutes_
0	1000	ultra	5	22.0	159.0	22.0	3.0	1000	30.0	
1	1000	ultra	6	43.0	172.0	60.0	23.0	1000	30.0	
2	1000	ultra	7	47.0	340.0	75.0	14.0	1000	30.0	
3	1000	ultra	8	52.0	408.0	81.0	14.0	1000	30.0	
4	1000	ultra	9	58.0	466.0	57.0	15.0	1000	30.0	

Анализ поведения клиентов: продолжительность звонков, количество

сообщений, объём трафика.

Посчитаем количество клиентов из выборки, которые пользуются тарифами smart и ultra:

```
Размер выборки клиентов тарифа Смарт: 2229
```

```
Размер выборки клиентов тарифа Смарт: 985
```

Продолжительность звонков

Посчитаем среднее количество минут разговора для обоих тарифов по каждому клиенту в среднем за месяц:

Тариф Smart

Out[43]:

	user_id	amount_sms	sum_duration_calls	sum_trafic_gb
0	1001	0.0	422.000000	16.000000
1	1002	10.0	216.714286	16.714286
2	1005	46.0	550.416667	8.583333
3	1006	0.0	318.272727	15.090909
4	1007	25.0	486.555556	14.888889

Тариф Ultra

Out[44]:

	user_id	amount_sms	sum_duration_calls	sum_trafic_gb
0	1000	62.0	320.750	13.625
1	1003	76.0	764.400	11.800
2	1004	149.0	123.375	18.750
3	1013	21.0	468.000	15.500
4	1016	71.0	61.000	13.500

Посчитаем общее среднее значение по продолжительности звонков по средним значениям за месяц по каждому пользователю:

Тариф Smart

Out[45]:
```
420.0
```

Тариф Ultra

Out[46]:
```
528.0
```

Теперь рассчитаем дисперсию для количества минут разговора в месяц для пользователей обоих тарифов. Т.к. у нас не генеральная совокупность данных, а выборка, то дисперсия будет оценинивать дисперсию по формуле s2=n−1∑(μ−xi)2, используя в методе var() параметр ddof=1:

Тариф Smart

Out[47]:
```
20974.05103943767
```

Тариф Ultra

Out[48]:
```
78867.04136496517
```

Рассчитаем стандратное отклонение для количества минут разговора в месяц для пользователей обоих тарифов.

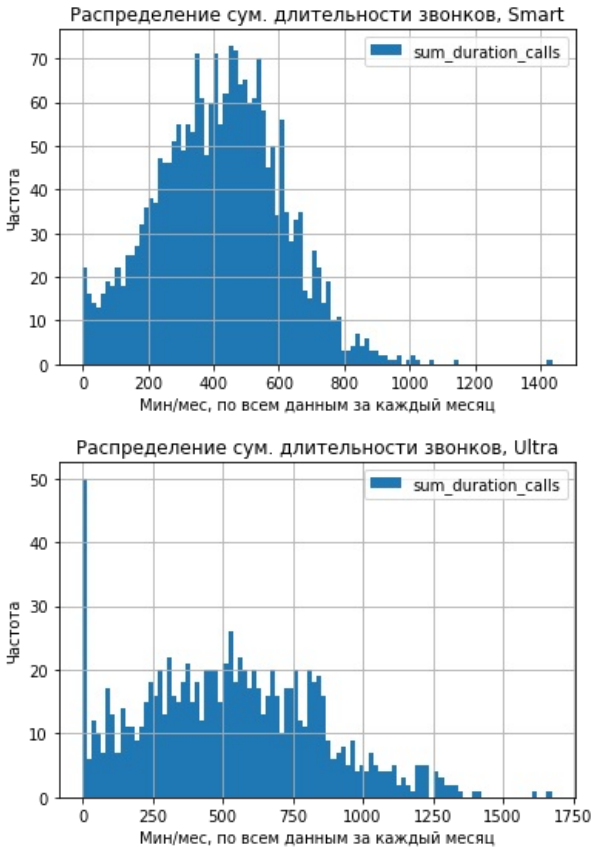
Тариф Smart

Out[49]: 145.03154348929033

Тариф Ultra

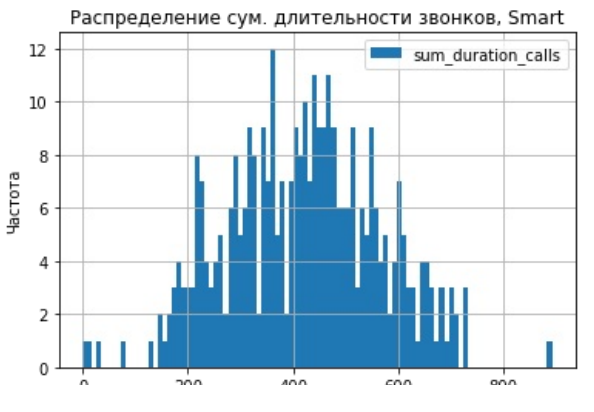
Out[50]: 281.786358524399

Построим гистограммы по каждому тарифу - по всем данным за каждый месяц:



По графикам, основанным на данных за все месяцы, видим, что суммарная продолжительность звонков для обоих тарифов распеределены нормально. При этом данные по тарифу Smart имеют скошенность в отрицательную сторону, а по тарифу Ultra в положительную. Также видим, что тарифом Smart клиенты пользуются чаще (по количеству месяцев), чем тарифом Ultra. Но пользователи тарифа Ultra в среднем раговаривают почти на 110 мин в месяц дольше, чем пользователи Smart. Также интересный факт, что среди клиентов Ultra чаще встречаются случаи, когда клиент не использовал за месяц ни одной минуты разгвовора.

И построим аналогичные графики, но по усредненным данным по каждому пользователю:



По графика, основанным на данных по пользователям по усредненной продолжительности звонков в месяц, видим, что у клиентов Smart продолжительность звонков в месяц находится в диапазоне 180 - 700 мин/мес, у клиентов ultra диапазон 180 - 1000 мин/мес.

Количество сообщений

Аналогичные расчёты выполним для показателя по количеству сообщений на обоих тарифах:

Тариф Smart

```
Ср.кол-во СМС: 33.0
```

Тариф Ultra

```
Ср.кол-во СМС: 55.0
```

Найдём дисперсию показателя "Количество смс" для обоих тарифов. Т.к. мы имеем дело не с генеральной совокупностью, а выборка, поэтому вместо дисперсии будем считаеть её оценку по имеющейся выборке (то есть s^2, а не σ^2). Для этого в методе var() укажем параметр ddof=1 :

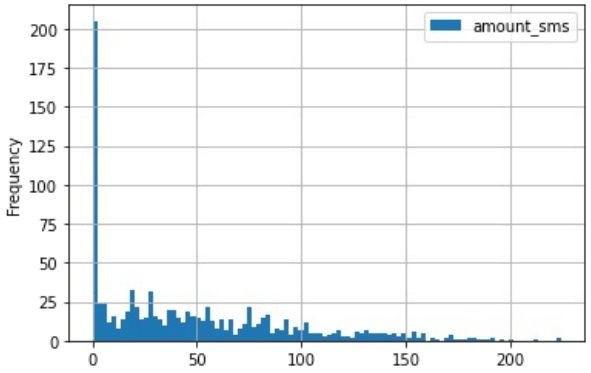
```
Оценка дисперсии для параметра кол-ва смс для тарифа Смарт: 703
```

```
Оценка дисперсии для параметра кол-ва смс для тарифа Ультра: 2150
```

```
Стандартное отклонение для параметра кол-ва смс для тарифа Смарт: 27
```

```
Стандартное отклонение для параметра кол-ва смс для тарифа Смарт: 46
```

Построим гистограммы по каждому тарифу - по всем данным за каждый месяц:



И построим аналогичные графики, но по усредненным данным по каждому пользователю:

По графикам видим, что данные скошены в положительную сторону. Поведение клиентов при использовании SMS отличается. Клиенты Smart чаще всего могут использовать 0-60 сообщений в месяц, тогда как для Ultra этот диапазон 0-100 сообщений/мес.Также видим,что достаточно часто клиенты вообще не используют ни одного сообщения в месяц.

Проверим есть ли зависимость отсутсвия смс и бОльщего расхода интернет трафика?

In [62]:
```
print(ultra['amount_sms'].corr(ultra['sum_trafic_gb']))
```

```
0.17390099857246077
```

По графику и коэффициенту корреляции видим, что зависимость между количеством смс и объемом трафка есть, но она не определяющая.

Объем трафика

```
Ср.кол-во Гб, тариф Смарт: 17.0
```

```
Ср.кол-во Гб, тариф Ультра: 20.0
```

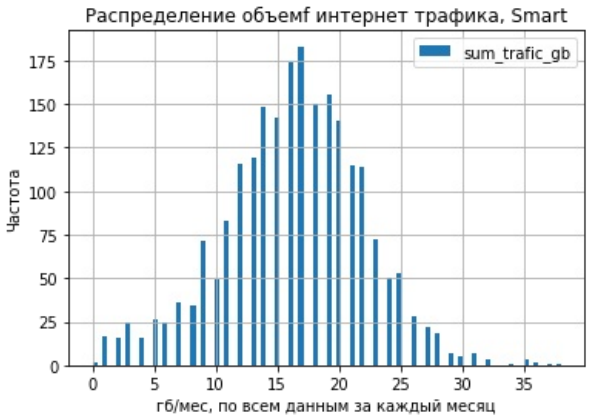
```
Оценка дисперсии для параметра кол-ва Гб для тарифа Смарт: 12
```

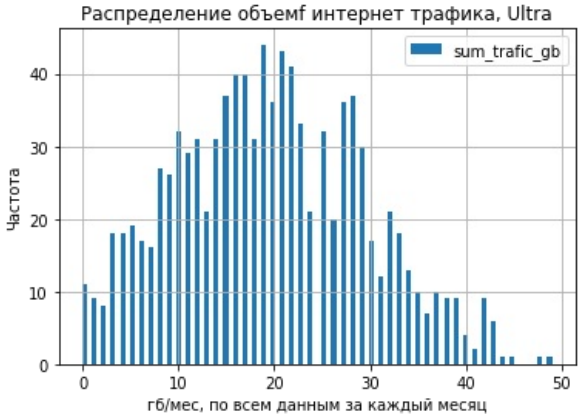
```
Оценка дисперсии для параметра кол-ва Гб для тарифа Ультра: 66
```

```
Стандартное отклонение для параметра кол-ва смс для тарифа Смарт: 4
```

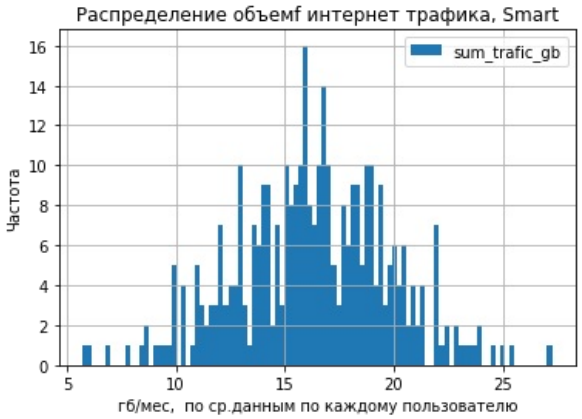
```
Стандартное отклонение для параметра кол-ва смс для тарифа Смарт: 8
```

Построим гистограммы по каждому тарифу - по всем данным за каждый месяц:





И построим аналогичные графики, но по усредненным данным по каждому пользователю:



По графика видим,что данные по объему трафка распределены нормально. На тарифе Ultra видим, что клиенты чаще пользуются большим объемом трафика, доходящим до 35 гб/мес, тогда как клиенты Smart чаще всего ограничиваются 20 - 25 гб/мес.

Проверка гипотез

Имеющаяся выборка является стратифицированной, т.к. поведение клиентов по использованию минут разговора, сообщений и интернет-трафика отличается.

Гипотеза_1

Проверим гипотезу: "Средняя выручка пользователей тарифов «Ультра» и «Смарт» различаются."

Сформулируем нулевую гипотезу H_0: "Средняя выручка пользователей тарифов «Ультра» и «Смарт» равны."

Альтернативная гипотеза H_1: "Средняя выручка пользователей тарифов «Ультра» и «Смарт» различаются." - двухсторонняя гипотеза, т.к. отличия могут быть как в одну так и другую сторону.

In [72]:
```
len(smart_pay_month) #смотрим размер выборки по тарифу smart
```

Out[72]: 2229

In [73]:
```
len(ultra_pay_month) #смотрим размер выборки по тарифу ultra
```

Out[73]: 985

При проверке гипотезы будем использовать доп.параметр equal_var = False, т.к. выборки по тарифам не большие и при этом значительно отличаются друг от друга.

In [74]:
```
alpha = .05 # критический уровень статистической значимости
# если p-value окажется меньше него - отвегнем нулевую гипотезу

results = st.ttest_ind(
    smart_pay_month,
    ultra_pay_month, 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: 4.2606313931076085e-250
Отвергаем нулевую гипотезу
```

Результат теста опровергает нулевую гипотезу о том, что средняя выручка пользователей тарифов «Ультра» и «Смарт» равны и оставляет нам альтернативную гипотезу о том, что средняя выручка пользователей тарифов «Ультра» и «Смарт» различаются.

Гипотеза_2

Проверим гипотезу: "Средняя выручка пользователей из Москвы отличается от выручки пользователей из других регионов."

Выделим из таблицы users столбцы 'user_id','city' в отдельную таблицу users_city и добавим данные о месте проживания клиента в таблицу data:

In [75]:
```
users_city = users.loc[:,['user_id','city']]
```

In [76]:
```
data = data.merge(users_city, on = ['user_id'])
```

In [77]:
```
Moscow_data = data[data['city']=='Москва']
len(Moscow_data)
```

Out[77]:
```
611
```

In [78]:
```
Moscow = Moscow_data['pay_month']
```

In [79]:
```
Region_data = data[data['city']!='Москва']
len(Region_data)
```

Out[79]:
```
2603
```

In [80]:
```
Region = Region_data['pay_month']
```

Сформулируем нулевую гипотезу H_0: "Средняя выручка пользователей из Москвы равна выручки пользователей из других регионов."

Альтернативная гипотеза H_1: "Средняя выручка пользователей из Москвы отличается от выручки пользователей из других регионов", двухсторонняя гипотеза, т.к. отличия могут быть как в одну так и другую сторону.

In [81]:
```
alpha = .05 # критический уровень статистической значимости
# если p-value окажется меньше него - отвегнем нулевую гипотезу

results = st.ttest_ind(
    Moscow,
    Region, 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: 0.5257376663729298
Не получилось отвергнуть нулевую гипотезу
```

Результат теста не опровергает нулевую гипотезу, что означает, что средняя выручка пользователей из Москвы не отличается от выручки пользователей из других регионов.