Исследование поездок по городам сервиса GoFast

Есть выборка пользователей по нескольким городам, а также поездка по ним. Необходимо проанализировать данные, провести предобработку, подсчитать выручку и проверить несколько гипотез по эффективности подписок и будущих акций компании для увеличения прибыли.

Содержание

Описание данных

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats as st
import seaborn as sns
import numpy as np
from scipy.stats import binom
from scipy.stats import norm
```

Шаг 1. Загрузка данных

In [2]:
```
data_users = pd.read_csv('/datasets/users_go.csv')
data_riders = pd.read_csv('/datasets/rides_go.csv')
data_subs = pd.read_csv('/datasets/subscriptions_go.csv')
data = [data_users,data_riders,data_subs]
```

In [3]:
```
for i in data:
    print(i.columns)
    print()
```

```
Index(['user_id', 'name', 'age', 'city', 'subscription_type'], dtype='object')

Index(['user_id', 'distance', 'duration', 'date'], dtype='object')

Index(['subscription_type', 'minute_price', 'start_ride_price',
       'subscription_fee'],
      dtype='object')
```

In [4]:
```
for i in data:
    print(i.head(10))
    print()
```

	user_id	name	age	city	subscription_type
0	1	Кира	22	Тюмень	ultra
1	2	Станислав	31	Омск	ultra
2	3	Алексей	20	Москва	ultra
3	4	Константин	26	Ростов-на-Дону	ultra
4	5	Адель	28	Омск	ultra
5	6	Регина	25	Краснодар	ultra
6	7	Игорь	23	Омск	ultra
7	8	Юрий	23	Краснодар	ultra
8	9	Ян	21	Пятигорск	ultra
9	10	Валерий	18	Екатеринбург	ultra

	user_id	distance	duration	date
0	1	4409.919140	25.599769	2021-01-01
1	1	2617.592153	15.816871	2021-01-18
2	1	754.159807	6.232113	2021-04-20
3	1	2694.783254	18.511000	2021-08-11
4	1	4028.687306	26.265803	2021-08-28
5	1	2770.890808	16.650138	2021-10-09
6	1	3039.020292	14.927879	2021-10-19
7	1	2842.118050	23.117468	2021-11-06
8	1	3412.690668	15.238072	2021-11-14
9	1	748.690645	15.041884	2021-11-22

	subscription_type	minute_price	start_ride_price	subscription_fee
0	free	8	50	0
1	ultra	6	0	199

In [5]:
```
for i in data:
    print(i.info())
    print()
```

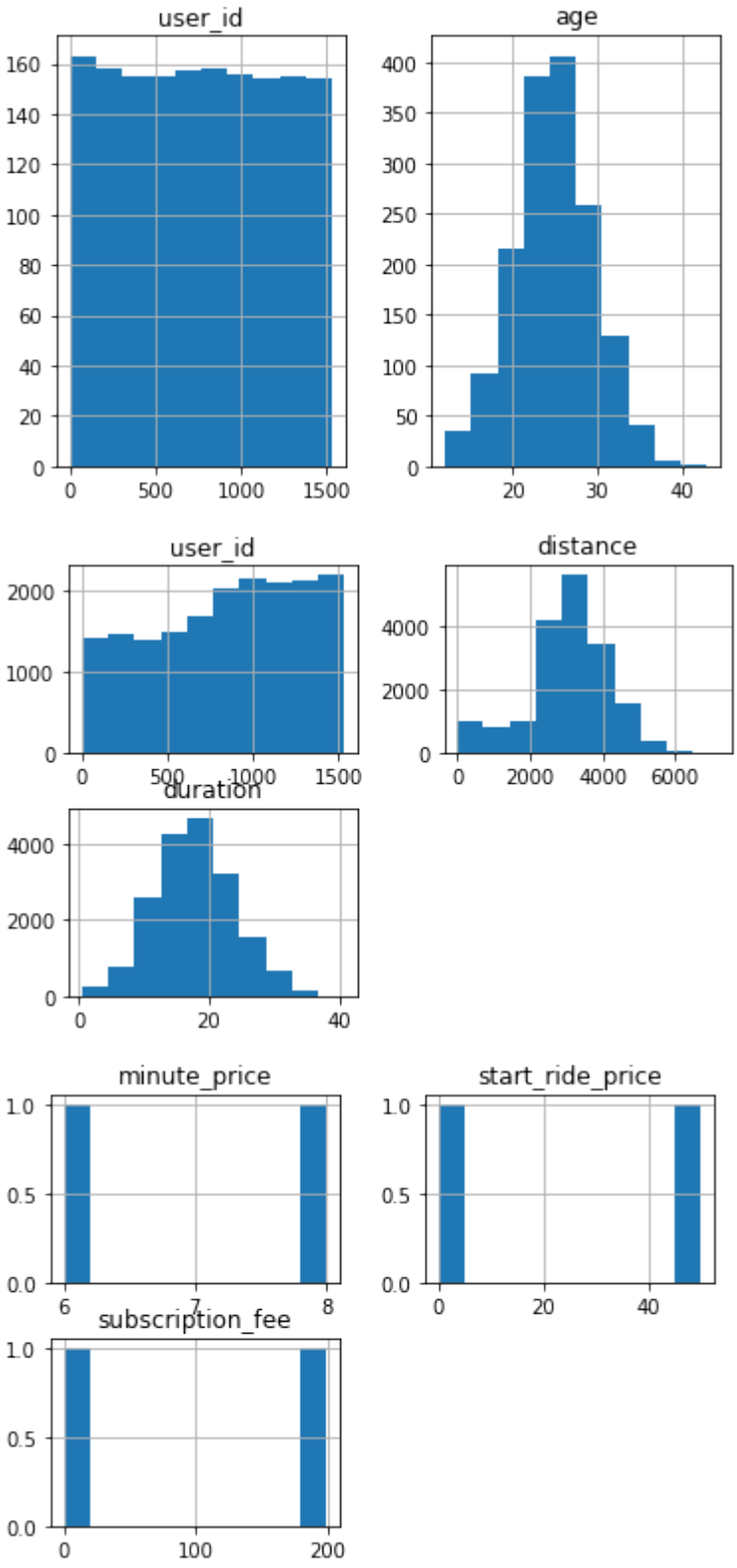
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1565 entries, 0 to 1564
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   user_id            1565 non-null   int64 
 1   name               1565 non-null   object
 2   age                1565 non-null   int64 
 3   city               1565 non-null   object
 4   subscription_type  1565 non-null   object
dtypes: int64(2), object(3)
memory usage: 61.3+ KB
None

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18068 entries, 0 to 18067
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   user_id   18068 non-null  int64  
 1   distance  18068 non-null  float64
 2   duration  18068 non-null  float64
 3   date      18068 non-null  object 
dtypes: float64(2), int64(1), object(1)
memory usage: 564.8+ KB
None

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   subscription_type  2 non-null      object
 1   minute_price       2 non-null      int64 
 2   start_ride_price   2 non-null      int64 
 3   subscription_fee   2 non-null      int64 
dtypes: int64(3), object(1)
memory usage: 192.0+ bytes
None
```

Вывод: На первый взгляд можно сказать, что явных пропусков нету, нужно привести колонку date к типу datetime и проверить на дубликаты

In [6]:
```
for i in data:
    i.hist()
    plt.show()
    print()
```



Вывод: Выборки age, distance, duration нормально распределены без явных выбросов

Шаг 2. Предобработка данных

Проверим на пропуски

In [7]:
```
for i in data:
    print(i.isna().sum())
```

```
    print()
```

```
user_id              0
name                 0
age                  0
city                 0
subscription_type    0
dtype: int64

user_id     0
distance    0
duration    0
date        0
dtype: int64

subscription_type    0
minute_price         0
start_ride_price     0
subscription_fee     0
dtype: int64
```

Изменение типа данных

In [8]:
```
data_riders['date'] = pd.to_datetime(data_riders['date'],format='%Y-%m-%d')#2021
```

In [9]:
```
data_riders['date'].head(5)#привели к пандас, проверили
```

Out[9]:
```
0   2021-01-01
1   2021-01-18
2   2021-04-20
3   2021-08-11
4   2021-08-28
Name: date, dtype: datetime64[ns]
```

Добавление столбца номер месяца

In [10]:
```
data_riders['month'] = data_riders['date'].dt.month
```

In [11]:
```
data_riders.head(5)
```

Out[11]:

	user_id	distance	duration	date	month
0	1	4409.919140	25.599769	2021-01-01	1
1	1	2617.592153	15.816871	2021-01-18	1
2	1	754.159807	6.232113	2021-04-20	4
3	1	2694.783254	18.511000	2021-08-11	8
4	1	4028.687306	26.265803	2021-08-28	8

Проверка неявные и явные дубликаты

In [12]:
```
for i in data:
    print(i[i.duplicated()].head(5))
```

```
    print()
```

```
      user_id        name  age          city subscription_type
1534      293       Агата   26     Краснодар             ultra
1535       16      Амалия   27     Краснодар             ultra
1536      909  Константин   20  Екатеринбург              free
1537      403      Полина   19          Сочи             ultra
1538      908      Рустам   30        Тюмень              free

Empty DataFrame
Columns: [user_id, distance, duration, date, month]
Index: []

Empty DataFrame
Columns: [subscription_type, minute_price, start_ride_price, subscription_fee]
Index: []
```

In [13]:
```
for i in data:
    print(i.duplicated().sum())
    print()
```

```
31

0

0
```

In [14]:
```
data_users = data_users.drop_duplicates()
```

In [15]:
```
data_users.duplicated().sum()
```

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

Вывод: Удалили явные дубликаты в data_users 31шт

In [16]:
```
for i in data:
    print(i.nunique())
    print()
```

```
user_id              1534
name                  194
age                    29
city                    8
subscription_type       2
dtype: int64

user_id      1534
distance    18068
duration    17974
date          364
month          12
dtype: int64

subscription_type    2
minute_price         2
start_ride_price     2
subscription_fee     2
dtype: int64
```

In [17]:
```
data_users['city'].unique()
```

Out[17]:
```
array(['Тюмень', 'Омск', 'Москва', 'Ростов-на-Дону', 'Краснодар',
       'Пятигорск', 'Екатеринбург', 'Сочи'], dtype=object)
```

Вывод: Неявных дубликатов не найдено

Вывод

Убрали дубликаты и добавили колонку месяц

Шаг 3. Исследовательский анализ данных

Частота встречаемости городов

In [18]:
```
data_users['city'].value_counts().plot.bar() # частота встречаемости городов
plt.title('Частота встречаемости городов')
plt.ylabel('Частота')
plt.show()
```

In [19]:
```
data_users['city'].value_counts()
```

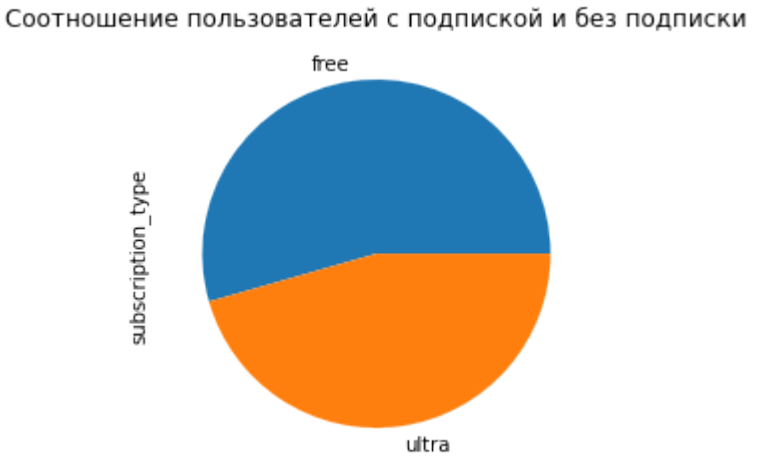
Out[19]:
```
Пятигорск          219
Екатеринбург       204
Ростов-на-Дону     198
Краснодар          193
Сочи               189
Омск               183
Тюмень             180
Москва             168
Name: city, dtype: int64
```

Вывод: больше в выборке встречается город Пятигорск, меньше Москва

Соотношение пользователей с подпиской и без подписки

In [20]:
```
pie_ = data_users['subscription_type'].value_counts()
total_ = data_users['subscription_type'].count()
print(pie_ / total_*100)
data_users['subscription_type'].value_counts().plot(kind='pie')
plt.title('Соотношение пользователей с подпиской и без подписки')
plt.show() #соотношение пользователей с подпиской и без подписки;
```

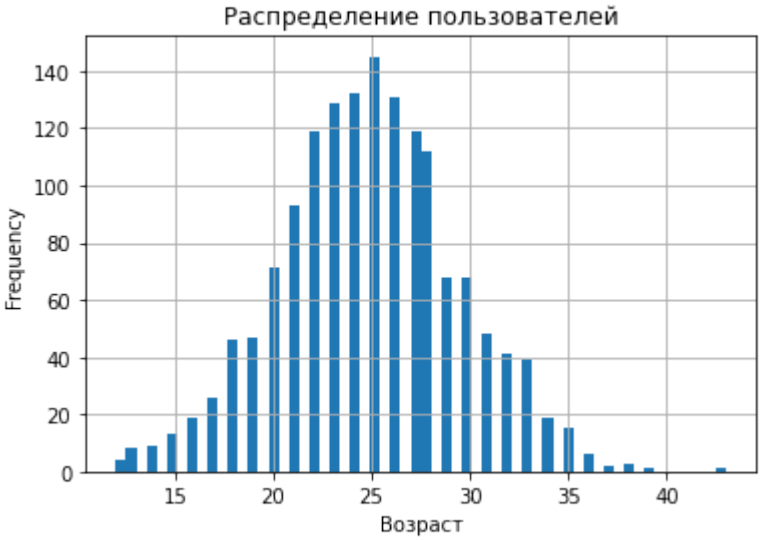
```
free     54.432855
ultra    45.567145
Name: subscription_type, dtype: float64
```



Вывод: Количество с подпиской free 54% с ultra 45% в предложенной выборке

Возраст пользователей

In [21]:
```
data_users['age'].plot(kind='hist',grid=True,bins = 60)
plt.title('Распределение пользователей')
plt.xlabel('Возраст')
plt.show()
```



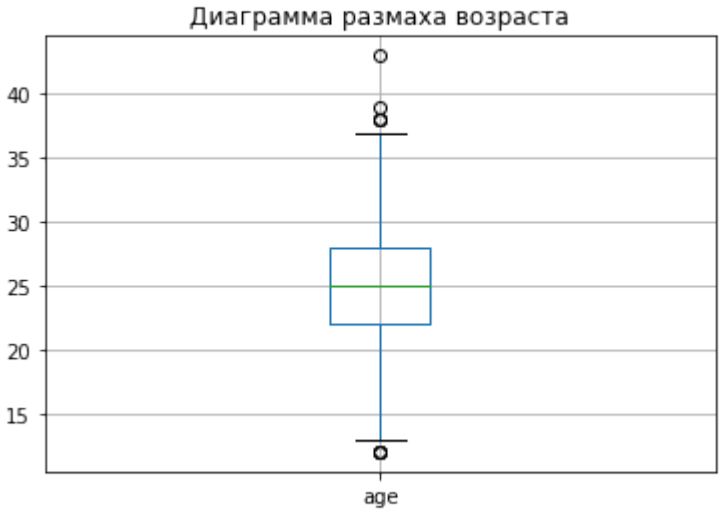
In [22]:
```
data_users['age'].agg(['median', 'mean', lambda x: x.mode()[0]])
```

Out[22]:
```
median      25.00000
mean        24.90352
<lambda>    25.00000
Name: age, dtype: float64
```

In [23]:
```
data_users['age'].describe()
```

Out[23]:
```
count    1534.000000
mean       24.903520
std         4.572367
min        12.000000
25%        22.000000
50%        25.000000
75%        28.000000
max        43.000000
Name: age, dtype: float64
```

In [24]:
```
data_users.boxplot(column='age')
plt.title('Диаграмма размаха возраста')
plt.show()
```



Вывод: В колонке возраст есть выбросы, лица больше 36 = 0.7% и меньше 13 = 0.26% Медиана, мода и среднее ~ 25 лет

Расстояние, которое пользователь преодолел за одну поездку

In [25]:
```
data_riders['distance'].hist(bins=100)
plt.title('Гистограмма расстояния')
plt.xlabel('Расстояния')
plt.ylabel('Частота')
plt.show()
```

In [26]:
```
data_riders.query('distance>6500').head(10)
```

Out[26]:

	user_id	distance	duration	date	month
10341	981	6671.969833	27.297078	2021-12-06	12
10915	1022	6535.386520	30.008799	2021-10-14	10
11319	1052	6503.600402	26.008309	2021-07-01	7
11385	1057	6601.197575	0.500000	2021-02-07	2
12086	1108	6538.937375	29.649276	2021-09-27	9
15580	1361	6908.491343	23.816983	2021-03-27	3
16309	1411	7211.007745	0.500000	2021-04-15	4
16484	1422	7066.003772	23.619318	2021-02-12	2
17171	1471	6760.940067	32.043760	2021-08-03	8
17242	1477	6724.932981	0.500000	2021-01-12	1

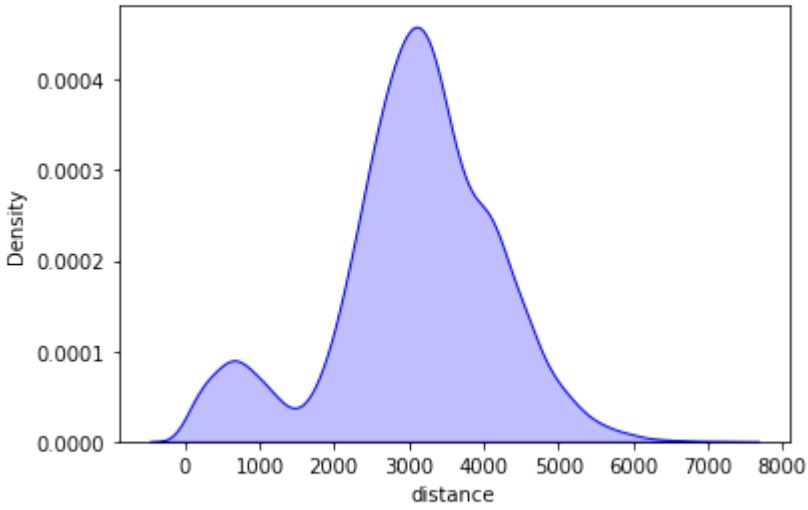
In [27]:
```
data_riders.query('distance<1000')['distance'].mean()
```

Out[27]:
```
549.2484838581698
```

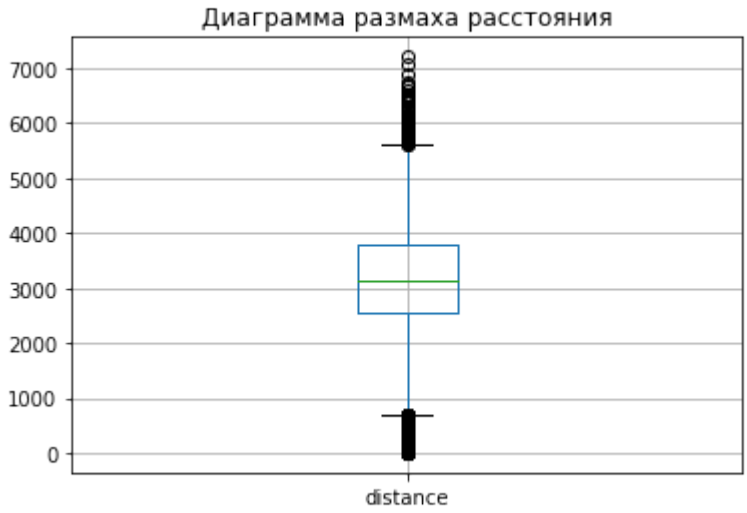
In [28]:
```
data_riders['distance'].agg(['median', 'mean', lambda x: x.mode()[0]])
```

Out[28]:
```
median      3133.609994
mean        3070.659976
<lambda>       0.855683
Name: distance, dtype: float64
```

In [29]:
```
sns.kdeplot(data_riders['distance'], shade=True, color="blue")  # Гистограмма пл
plt.show()
```



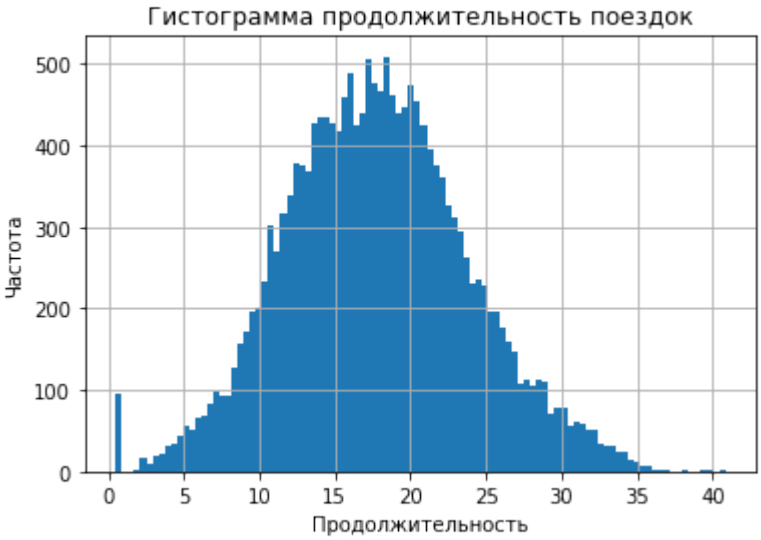
In [30]:
```
data_riders.boxplot(column=['distance'])
plt.title('Диаграмма размаха расстояния')
plt.show()
```



Вывод: Гистограмма показывает, что выборка близка к нормальному, есть локальный пик до 549 м и просадка около 1500 м, возможно связана с непонными данными. Среднее расстояние поездки 3133 м , Медиана 3070 м

Продолжительность поездок

In [31]:
```
data_riders['duration'].hist(bins=100)
plt.title('Гистограмма продолжительность поездок')
plt.xlabel('Продолжительность')
plt.ylabel('Частота')
plt.show()
```



In [32]:
```
data_riders[data_riders['duration']<3].head(30)
```

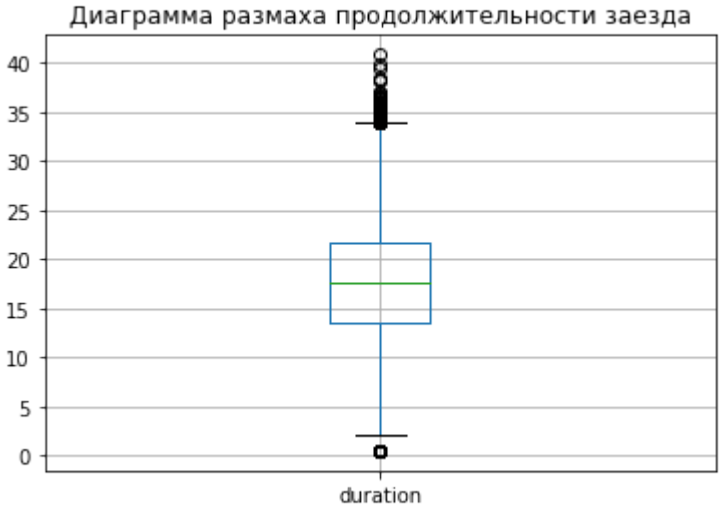
Out[32]:

	user_id	distance	duration	date	month
6531	701	4196.445201	0.500000	2021-08-31	8
6570	704	4830.824371	0.500000	2021-10-14	10
6680	712	4684.004397	0.500000	2021-03-04	3
6691	712	23.963062	2.053251	2021-11-22	11
6695	713	5215.642710	0.500000	2021-02-25	2
6768	718	5197.436649	0.500000	2021-02-11	2
6801	720	102.323624	2.292353	2021-09-20	9
6860	724	6225.520342	0.500000	2021-12-16	12
6883	726	4452.491518	0.500000	2021-03-01	3
7018	735	4406.954812	0.500000	2021-04-13	4
7160	745	5286.167732	0.500000	2021-08-17	8
7364	760	4881.391865	0.500000	2021-08-29	8
7437	766	4539.088310	0.500000	2021-12-22	12
7453	768	273.136262	2.744917	2021-02-20	2
7508	772	4718.820996	0.500000	2021-05-22	5
7537	774	5488.141903	0.500000	2021-05-27	5
7612	780	6112.644835	0.500000	2021-11-23	11
7679	784	5202.815712	0.500000	2021-06-03	6
7791	792	5353.189287	0.500000	2021-11-06	11
7830	795	6262.302747	0.500000	2021-09-09	9
7883	800	5052.410425	0.500000	2021-07-19	7
7932	804	5547.830031	0.500000	2021-05-10	5
8329	836	4958.922077	0.500000	2021-01-12	1
8472	847	4770.172208	0.500000	2021-03-26	3
8676	863	466.343107	2.999307	2021-04-11	4
8906	880	11.283615	2.035632	2021-03-28	3
9049	890	4848.485014	0.500000	2021-11-20	11
9077	893	4762.120493	0.500000	2021-03-18	3
9135	896	5466.265687	0.500000	2021-10-13	10
9268	906	5258.160917	0.500000	2021-02-02	2

In [33]:
```
data_riders[data_riders['duration']<2].count()
```

Out[33]:
```
user_id     95
distance    95
duration    95
date        95
month       95
dtype: int64
```

In [34]:
```
data_riders.boxplot(column=['duration'])
plt.title('Диаграмма размаха продолжительности заезда')
plt.show()
```



In [35]:
```
print((data_riders['duration']<3).mean()*100)
print((data_riders['duration']>36).mean()*100)
```

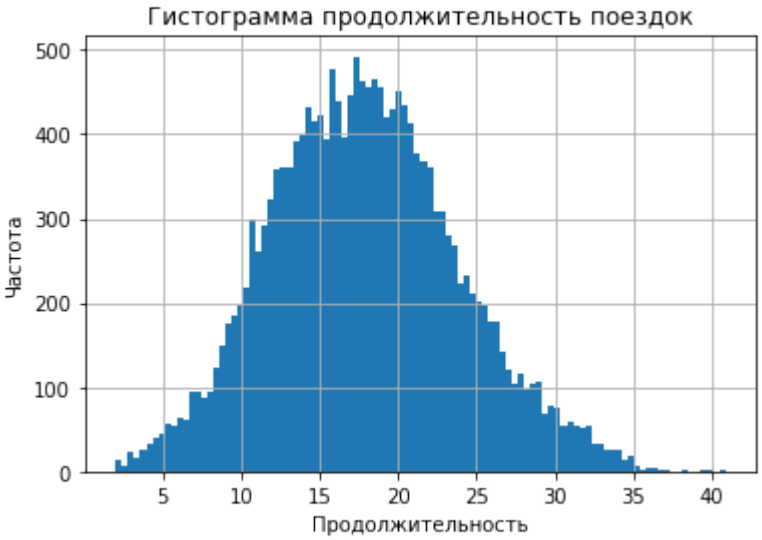
```
0.7084348018596414
0.07748505645339827
```

In [36]:
```
data_riders['duration'].agg(['median', 'mean', lambda x: x.mode()[0]])
```

Out[36]:
```
median      17.678395
mean        17.805011
<lambda>     0.500000
Name: duration, dtype: float64
```

In [37]:
```
data_riders = data_riders[(data_riders['duration'] >= 2)]
```

In [38]:
```
data_riders['duration'].hist(bins=100)
plt.title('Гистограмма продолжительность поездок')
plt.xlabel('Продолжительность')
plt.ylabel('Частота')
plt.show()
```



In [39]:
```
data_riders['duration'].agg(['median', 'mean', lambda x: x.mode()[0]])
```

Out[39]:
```
median      17.714122
mean        17.896480
<lambda>     2.035632
Name: duration, dtype: float64
```

Промежуточный вывод:

- Больше всего в выборке присутствуют `города` Пятигорска и меньше Москвы.
- Количество с подпиской free 54% с ultra 45% в предложенной выборке
- Средний `возраст` пользователей 25 лет
- Среднее `расстояние за одну сессию` 3133 метров, а также есть локальный максимум в 549 метров
- Избавились от выбросов в колонке `Продолжительность поездок`. 95 поездок было с продолжительностью 0.5 минут, что возможно связано с какой-то ошибкой в получение данных. Средняя продолжительность поездки 18 минут

Шаг 4. Объединение данных

In [40]:
```
data_merge = data_users.merge(data_riders, on='user_id')
data_merge = data_merge.merge(data_subs, on='subscription_type')
```

In [41]:
```
data_merge.head(5)
```

Out[41]:

	user_id	name	age	city	subscription_type	distance	duration	date	mon
0	1	Кира	22	Тюмень	ultra	4409.919140	25.599769	2021-01-01	
1	1	Кира	22	Тюмень	ultra	2617.592153	15.816871	2021-01-18	
2	1	Кира	22	Тюмень	ultra	754.159807	6.232113	2021-04-20	
3	1	Кира	22	Тюмень	ultra	2694.783254	18.511000	2021-08-11	
4	1	Кира	22	Тюмень	ultra	4028.687306	26.265803	2021-08-28	

In [42]:
```
data_merge.info()
print()
print(data_merge.isna().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 17973 entries, 0 to 17972
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   user_id            17973 non-null  int64         
 1   name               17973 non-null  object        
 2   age                17973 non-null  int64         
 3   city               17973 non-null  object        
 4   subscription_type  17973 non-null  object        
 5   distance           17973 non-null  float64       
 6   duration           17973 non-null  float64       
 7   date               17973 non-null  datetime64[ns]
 8   month              17973 non-null  int64         
 9   minute_price       17973 non-null  int64         
 10  start_ride_price   17973 non-null  int64         
 11  subscription_fee   17973 non-null  int64         
dtypes: datetime64[ns](1), float64(2), int64(6), object(3)
memory usage: 1.8+ MB

user_id              0
name                 0
age                  0
city                 0
subscription_type    0
distance             0
duration             0
date                 0
month                0
minute_price         0
start_ride_price     0
subscription_fee     0
dtype: int64
```

In [43]:
```
data_free_sub = data_merge.query('subscription_type == "free"')
```

In [44]:
```
data_ultra_sub = data_merge.query('subscription_type == "ultra"')
```

In [45]:
```
data_free_sub.shape
```

Out[45]: (11473, 12)

In [46]:
```
data_ultra_sub.shape
```

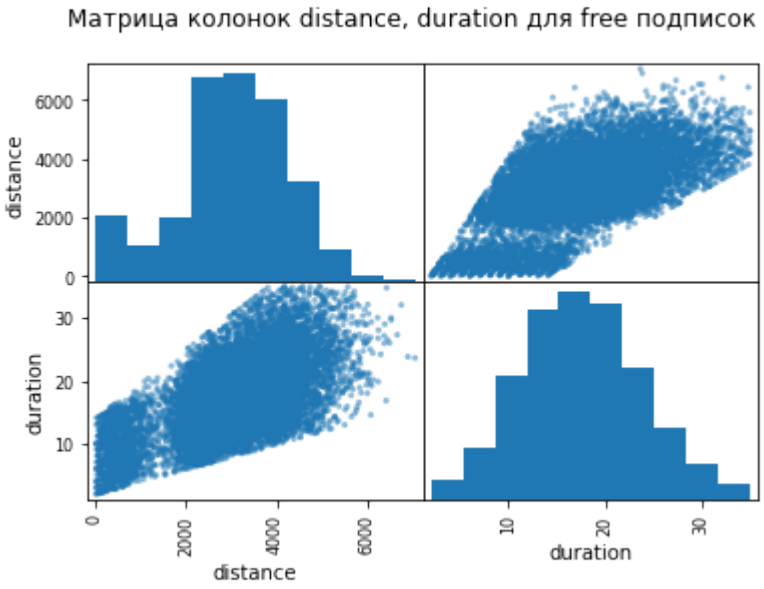
Out[46]: (6500, 12)

In [47]:
```
data_free_sub.shape[0] - data_ultra_sub.shape[0]
```

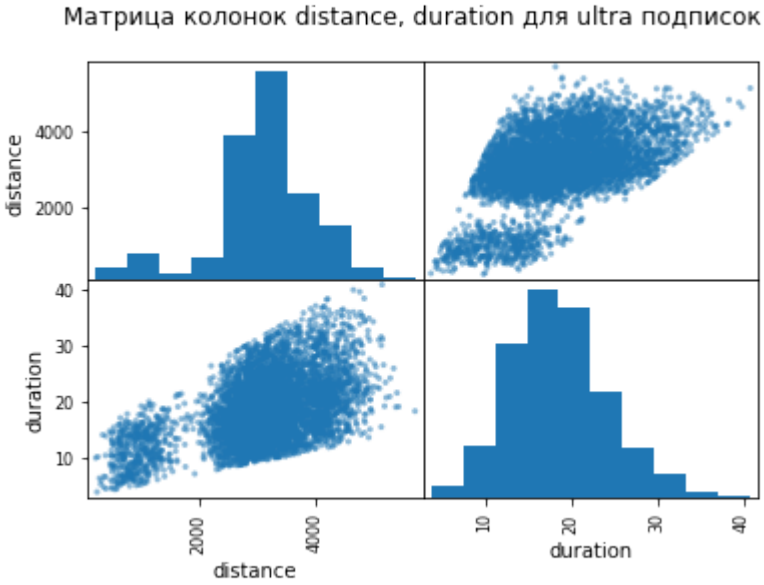
Out[47]: 4973

In [48]:
```
columns = ['distance','duration']
```

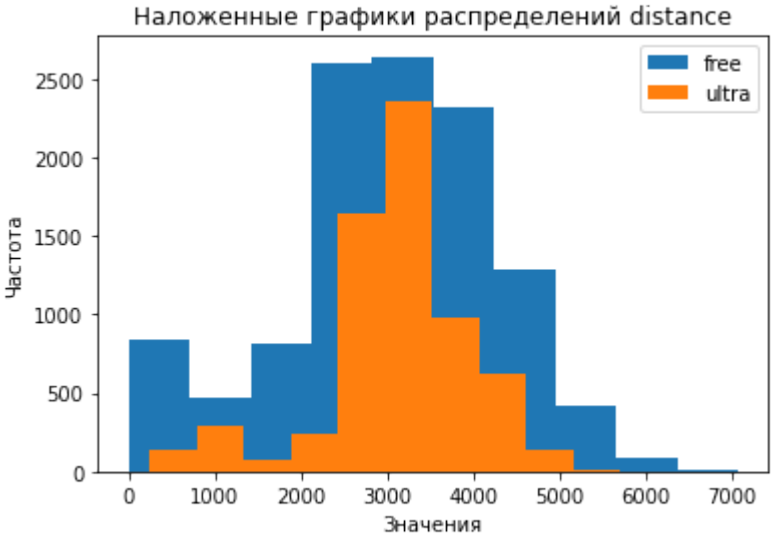
In [49]:
```
ax1 = data_free_sub[columns]
pd.plotting.scatter_matrix(ax1)
plt.suptitle('Матрица колонок distance, duration для free подписок')
plt.show()
```



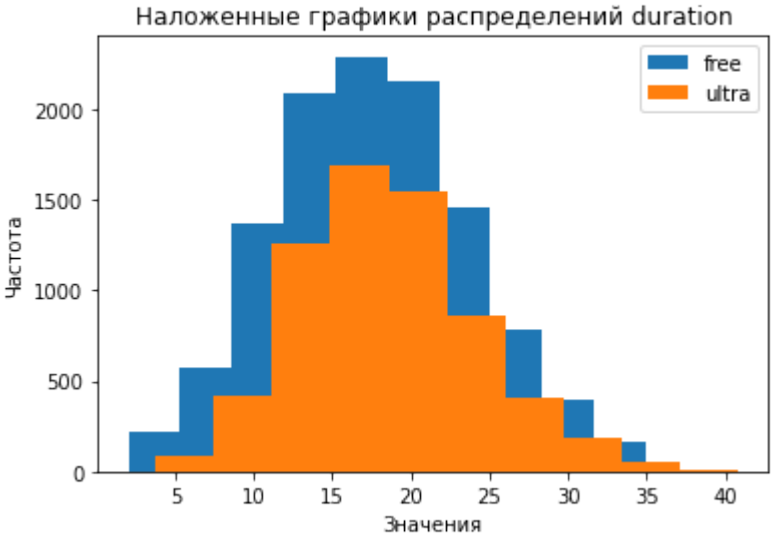
In [50]:
```
ax2 = data_ultra_sub[columns]
pd.plotting.scatter_matrix(ax2)
plt.suptitle('Матрица колонок distance, duration для ultra подписок')
plt.show()
```



In [51]:
```
# data_free_sub[columns].hist(label='Распределение 1')
# data_ultra_sub[columns].hist(label='Распределение 2', color='red')
plt.hist(data_free_sub['distance'],label='free')
plt.hist(data_ultra_sub['distance'],label='ultra')
plt.legend()
plt.xlabel('Значения')
plt.ylabel('Частота')
plt.title('Наложенные графики распределений distance')
plt.show()
```



In [52]:
```
plt.hist(data_free_sub['duration'],label='free')
plt.hist(data_ultra_sub['duration'],label='ultra')
plt.legend()
plt.xlabel('Значения')
plt.ylabel('Частота')
plt.title('Наложенные графики распределений duration')
plt.show()
```



In [53]:
```
data_ultra_sub['distance'].median()
```

Out[53]: 3148.6372291760854

Промежуточный вывод:

Объединили таблицы, построили гистограммы и график рассеяния для разных подписок.Визуализировали информацию о расстоянии и времени поездок. Количество пользователей с подпиской `ultra` в 2 раза меньше в данной выборке. Пользователи оформляют подписку чаще если ездят от 3км.

Шаг 5. Подсчёт выручки

In [54]:
```
data_merge['duration'] = np.ceil(data_merge['duration'])

group_users = data_merge.groupby(['user_id','month']).agg({'distance':'sum','dur
group_users = group_users.rename(columns={'user_id':'counts'})
group_users=group_users.reset_index()
group_users
```

Out[54]:

	user_id	month	distance	duration	subscription_type	counts
0	1	1	7027.511294	42.0	ultra	2
1	1	4	754.159807	7.0	ultra	1
2	1	8	6723.470560	46.0	ultra	2
3	1	10	5809.911100	32.0	ultra	2
4	1	11	7003.499363	56.0	ultra	3
...	...	...	...	...	...	...
11295	1534	6	3409.468534	26.0	free	2
11296	1534	8	7622.453034	48.0	free	2
11297	1534	9	4928.173852	23.0	free	1
11298	1534	11	13350.015305	78.0	free	4
11299	1534	12	2371.711192	16.0	free	1

11300 rows × 6 columns

In [55]:
```
data_subs
```

Out[55]:

	subscription_type	minute_price	start_ride_price	subscription_fee
0	free	8	50	0
1	ultra	6	0	199

In [56]:
```
def func_monthly_revenue(row):
    try:
        if row['subscription_type'] == 'ultra':
            return data_subs['start_ride_price'].iloc[1]*row['counts']+row['dura
        elif row['subscription_type'] == 'free':
            return data_subs['start_ride_price'].iloc[0]*row['counts']+row['dura
    except:
        print('smt wrong')
```

In [57]:
```
group_users['monthly_revenue'] = group_users.apply(func_monthly_revenue,axis=1)
group_users
```

Out[57]:

	user_id	month	distance	duration	subscription_type	counts	monthly_rev
0	1	1	7027.511294	42.0	ultra	2	
1	1	4	754.159807	7.0	ultra	1	
2	1	8	6723.470560	46.0	ultra	2	
3	1	10	5809.911100	32.0	ultra	2	
4	1	11	7003.499363	56.0	ultra	3	
...	...	...	...	...	...	...	
11295	1534	6	3409.468534	26.0	free	2	
11296	1534	8	7622.453034	48.0	free	2	
11297	1534	9	4928.173852	23.0	free	1	
11298	1534	11	13350.015305	78.0	free	4	
11299	1534	12	2371.711192	16.0	free	1	

11300 rows × 7 columns

Вывод:

Создали агрегированную таблицу для каждого пользователя по каждому месяцу. Подсчитали помесячную выручку.

Шаг 6. Проверка гипотез

Гипотеза о продолжительности времени пользователей

Гипотеза H0: Среднее время, которое проводят подписчики и пользователи без подписки, одинаково. То есть, нет разницы в продолжительности поездок.

Альтернативная гипотеза H1 Среднее время, которое тратят подписчики больше, чем среднее время без подписки

In [58]:
```
alpha = 0.05

subs_duration_on = data_merge[data_merge['subscription_type'] == 'ultra']['durat
subs_duration_off = data_merge[data_merge['subscription_type'] == 'free']['durat

results = st.ttest_ind(subs_duration_on,subs_duration_off,alternative='greater')

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

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

```
p-value: 8.577910347796266e-28
Отвергаем нулевую гипотезу
```

Вывод: Полученное значение p-value, значимо меньше заданного. Есть основания предполагать, что пользователи с подпиской тратят больше времени на поезду

Среднее расстояние, которое проезжают пользователи с подпиской

Гипотеза H0: Среднее расстояние, которое проезжают пользователи с подпиской за одну поездку, равно 3130 метров

Альтернативная гипотеза H1: Среднее расстояние, которое проезжают пользователи с подпиской за одну поездку, превышеют оптимальное значение 3130 метров

In [59]:
```
subs_distance = data_merge[data_merge['subscription_type'] == 'ultra']['distance

alpha=0.05
value=3130

results_2 = st.ttest_1samp(subs_distance,value, alternative = 'greater')

print(f'p-value: {results_2.pvalue}')

if results_2.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Нет оснований отвергнуть нулевую гипотезу')
```

```
p-value: 0.9195368847849785
Нет оснований отвергнуть нулевую гипотезу
```

Вывод: p-value 91%, среднее расстояние значимо близко к 3130 метрам. Подписчики в среднем **не проезжают больше** оптимального значения.

Помесячная выручка от пользователей с подпиской

Гипотеза H0: Помесячная выручка от пользователей с подпиской равна выручке от пользователей без подписки.

Альтернативная гипотеза H1: Помесячная выручка от пользователей с подпиской по месяцам выше, чем выручка от пользователей без подписки.

In [60]:
```
alpha = 0.05

subs_monthly_revenue_on = group_users[group_users['subscription_type'] == 'ultra
subs_monthly_revenue_off = group_users[group_users['subscription_type'] == 'free

results_3 = st.ttest_ind(subs_monthly_revenue_on,subs_monthly_revenue_off,altern

print(f'p-value: {results_3.pvalue}')

if results_3.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
```

```
else:
    print('Нет оснований отвергнуть нулевую гипотезу')
```

```
p-value: 2.0314113674863288e-30
Отвергаем нулевую гипотезу
```

Вывод: Отвергаем нулевую гипотезу: выручка от пользователей с подпиской значимо больше

Задание о изменение количества обращений в техподдержку

Техническая команда сервиса обновила сервера, с которыми взаимодействует мобильное приложение. Она надеется, что из-за этого количество обращений в техподдержку значимо снизилось. Некоторый файл содержит для каждого пользователя данные о количестве обращений до обновления и после него.

Так как у нас два **зависимых датасета** до и после изменений, нам понадобится тест о равенстве средних двух зависимых выборок scipy.stats.ttest_rel()

Промежуточный вывод

- Гипотеза: Среднее время, которое тратят подписчики больше, чем среднее время без подписки **не потвердилась**. Полученное значение p-value, значимо меньше заданного. Есть основания предполагать, что пользователи с подпиской тратят больше времени на поезду
- Среднее расстояние, которое проезжают пользователи с подпиской за одну поездку, равно 3130 метров. Оснований отвергать гипотезу нет. p-value = 16%
- Гипотеза: Помесячная выручка от пользователей с подпиской равна выручке от пользователей без подписки **не потвердилась**. Выручка от пользователей с подпиской значимо больше

Шаг 7. Распределения

Акции с бесплатной раздачей промокодов

Выясните, какое минимальное количество промокодов нужно разослать, чтобы вероятность не выполнить план была примерно 5 % Минимум 100 существующих клиентов должны продлить эту подписку

Выбранную задачу можно описать биноминальным распределением

- фиксированное число попыток(пользователей)
- два исхода(продлил/не продлил)
- события независимы между собой
- вероятность успеха одинаково

Далее апроксимируем биноминальное распределение нормальным распределением для нахождения минимального количества промокодов, чтобы не

выполнить план равнялась 5%

In [61]:
```
p = 0.1
k= 100 # так как binom.cdf() находит значения от P(X<=100),а нам необходимо мин

for i in range(500,2000):
    res = binom.cdf(k,i,p)
    if res <= 0.05:
        print(f'Минимальное количество промокодов = {i}')
        print(f'Вероятность не продлить подписку = {res}')
        break
```

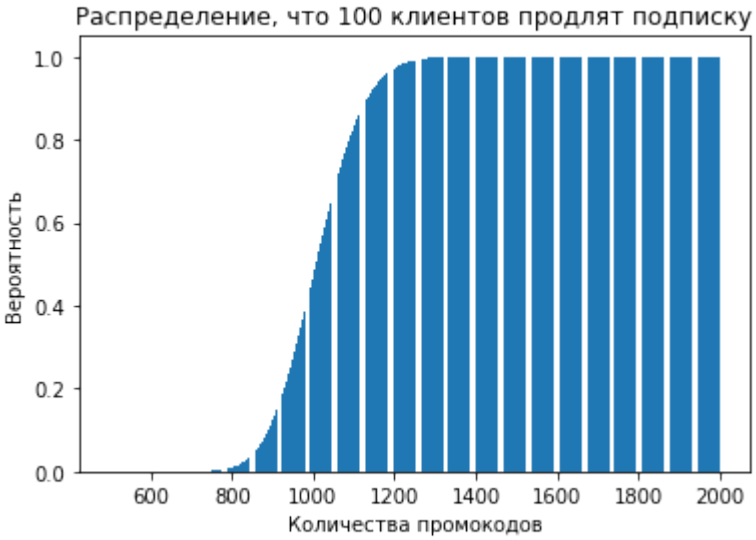
```
Минимальное количество промокодов = 1172
Вероятность не продлить подписку = 0.04954795210203071
```

In [62]:
```
p = 0.1

distr = []

for i in range(500,2000):
    current_value =1 -binom.cdf(100, i, p)
    distr.append(current_value)

plt.bar(range(500, 2000), distr)
plt.title('Распределение, что 100 клиентов продлят подписку')
plt.xlabel('Количества промокодов')
plt.ylabel('Вероятность')
plt.show()
```



Вывод: - Минимальное количество промокодов на один бесплатный месяц подписки при котором, вероятно, не выполнится план в 5%,то что минимум 100 пользователь не продлит следущую подписку равен **1172** клиентам

Количество открывших push-уведомления

Выбранную задачу можно описать биноминальным распределением

- фиксированное число попыток
- два исхода(открыл/не открыл)
- события независимы между собой

- вероятность успеха одинаково

Построить примерный график распределения и оцените вероятность того, что уведомление откроют не более 399,5 тыс. пользователей.

In [ ]:
```
p = 0.4
n = 1000000
u = n*p
sigma = sqrt(n*p*(1 - p))
if (u - (3 * sigma) >= 0) and ((u + (3 * sigma) <= n)):
    print('Условие выполняется')
else:
    print('Условие не выполняется')
```

In [ ]:
```
distr = st.norm(u,sigma)

x = np.linspace(int(n * p - 4 * sqrt(n * p * (1 - p))),  int(n * p + 4 * sqrt(n
y = norm.pdf(x,u,sigma)

plt.plot(x,y,'b-',lw=3)
plt.title('Апроксимация биноминального распределения')
plt.show()
```

In [ ]:
```
distr.cdf(399500)
```

Вывод: Вероятность, что уведомление откроют не более 399500 человек из 1миллиона разосланных, равна **15%**

Вывод

Предобработка данных

- Данные хорошо подготовлены, отсутствуют какие либо пропуски
- Убрали явные дубликаты
- Добавили колонку месяц

Исследовательнский анализ данных

- Больше в выборке встречается город Пятигорск, меньше Москва
- Количество с подпиской free 54% с ultra 45% в предложенной выборке
- Средний возраст пользователей 25 лет
- Больше всего в выборке присутствуют города Пятигорска и меньше Москвы.
- Среднее расстояние за одну сессию 3133 метров, а также есть локальный максимум в 549 метров
- Избавились от выбросов в колонке `Продолжительность поездок`. 95 поездок было с продолжительностью 0.5 минут, что возможно связано с какой-то ошибкой в получение данных. Средняя продолжительность поездки 18 минут

Создание сводной таблицы

- Объединили таблицы, построили гистограммы и график рассеяния для разных подписок.
- Визуализировали информацию о расстоянии и времени поездок.
- Количество пользователей с подпиской ultra в 2 раза меньше в данной выборке.
- Пользователи оформляют подписку чаще если ездят от 3км в данной выборке

Подсчет выручки

- Создали агрегированную таблицу для каждого пользователя по каждому месяцу.
- Подсчитали помесячную выручку.

Проверка гипотез

- Гипотеза: Среднее время, которое тратят подписчики больше, чем среднее время без подписки **не потвердилась.** Полученное значение p-value, значимо меньше заданного. Есть основания предполагать, что пользователи с подпиской **тратят больше времени на поезду**
- p-value 91%, среднее расстояние значимо близко к 3130 метрам. Подписчики в среднем не проезжают больше оптимального значения
- Гипотеза: Помесячная выручка от пользователей с подпиской равна выручке от пользователей без подписки **не потвердилась**. Выручка от пользователей с подпиской **значимо больше**.

Распределения

1. Минимальное количество промокодов на один бесплатный месяц подписки при котором, вероятно, не выполнится план в 5%,то что минимум 100 пользователь не продлит следущую подписку равен 1172 клиентам

2. Для задачи про push-уведомления создана апроксимация биноминольного распределния нормальным и оценена вероятность. Вероятность, что уведомление откроют не более 399500 человек из 1миллиона разосланных push, равна **15%**

Общий вывод

Было проведено исследование поездок по городам сервиса GoFast выполнена предобработку, подсчитана выручка и проверить несколько гипотез по эффективности подписок и будущих акций компании для увеличения прибыли, среднее расстояние поездок пользователей, различие между клиентами с подпиской и без и оценка их прибыльности