Проект: Статистический анализ данных. Исследование сервиса аренды самокатов GoFast.

Описание проекта:

- современный мир очень активно меняется, ритм и темп жизни ускоряется, тробания повышаются и чтобы соответсвовать запросам современного мира компаниям необходимо быть в курсе изменений, новых веяний и требований. В современном мегаполисе сервис по аренде самокатов может стать очень прибыльным делом, так как среди населения есть запрос на ускорение ритмов и всё меньше люди стремятся проявлять физическую в активность, менее популярны стали пешие прогулки. Правильный анализ данных поможет выявить на сколько рынок аренды самокатов соответсвует запросам и поможет определить основные парамаетры вляиющие на доход сервиса аренды самокатов GoFast.

Цель исследования:

- анализ данные и проверка некоторых гипотез, которые помогут бизнесу вырасти.

Ход исследования:

- подготовка данных: загрузка и изучение общих данных из датасета;
- Преодбработка данных: выявление и заполнения пропущенных значений, вявление и удаление дубликатов, приведение данных в столбцах к необходимому типуб добаление необходимы для анализа столбцов.
- исследовательский анализ данных: описание и визуализация общей информации о пользователях и поездках, изучение основных параметров данных и выявление наиболее "выгодных" пользователей.

Общий вывод:

- резюмированные полученных результатов, формулировка ключевых выводов и рекомендаций.

С помощью данного исследования мы стремимся дать анализ рынка аренды самокатов, что станет отправной точкой для развития и расширения бизнеса аренды самокатов.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy import stats
import warnings
warnings.simplefilter('ignore')
```

Загрузка данных

In [2]:
```
users_data = pd.read_csv("/datasets/users_go.csv")
rides_data = pd.read_csv("/datasets/rides_go.csv")
subscr_data = pd.read_csv("/datasets/subscriptions_go.csv")
```

Предобработка данных

In [3]:
```
users_data.head(10)
```

Out[3]:

	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

In [4]:
```
rides_data.head(10)
```

Out[4]:

	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

In [5]:
```
subscr_data
```

Out[5]:

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

In [6]:
```
users_data.info()
```

```
<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
```

In [7]:
```
rides_data.info()
```

```
<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
```

In [8]:
```
subscr_data.info()
```

```
<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
```

In [9]:
```
users_data.isna().sum()
```

Out[9]:
```
user_id              0
name                 0
age                  0
city                 0
subscription_type    0
dtype: int64
```

In [10]:
```
rides_data.isna().sum()
```

Out[10]:
```
user_id     0
distance    0
duration    0
date        0
dtype: int64
```

In [11]:
```
users_data.duplicated().sum()
```

Out[11]:
```
31
```

In [12]:
```
rides_data.duplicated().sum()
```

Out[12]:
```
0
```

In [13]:
```
users_data = users_data.drop_duplicates(keep='first').reset_index(drop=True)
users_data.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
users_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1534 entries, 0 to 1533
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   user_id            1534 non-null   int64 
 1   name               1534 non-null   object
 2   age                1534 non-null   int64 
 3   city               1534 non-null   object
 4   subscription_type  1534 non-null   object
dtypes: int64(2), object(3)
memory usage: 60.0+ KB
```

In [15]:
```
len(users_data['user_id'].unique())
```

Out[15]:
```
1534
```

In [16]:
```
users_data.duplicated().sum()
```

Out[16]:
```
0
```

In [17]:
```
rides_data.describe()
```

Out[17]:

	user_id	distance	duration
count	18068.000000	18068.000000	18068.000000
mean	842.869936	3070.659976	17.805011
std	434.734317	1116.831209	6.091051
min	1.000000	0.855683	0.500000
25%	487.000000	2543.226360	13.597563
50%	889.000000	3133.609994	17.678395
75%	1213.250000	3776.222735	21.724800
max	1534.000000	7211.007745	40.823963

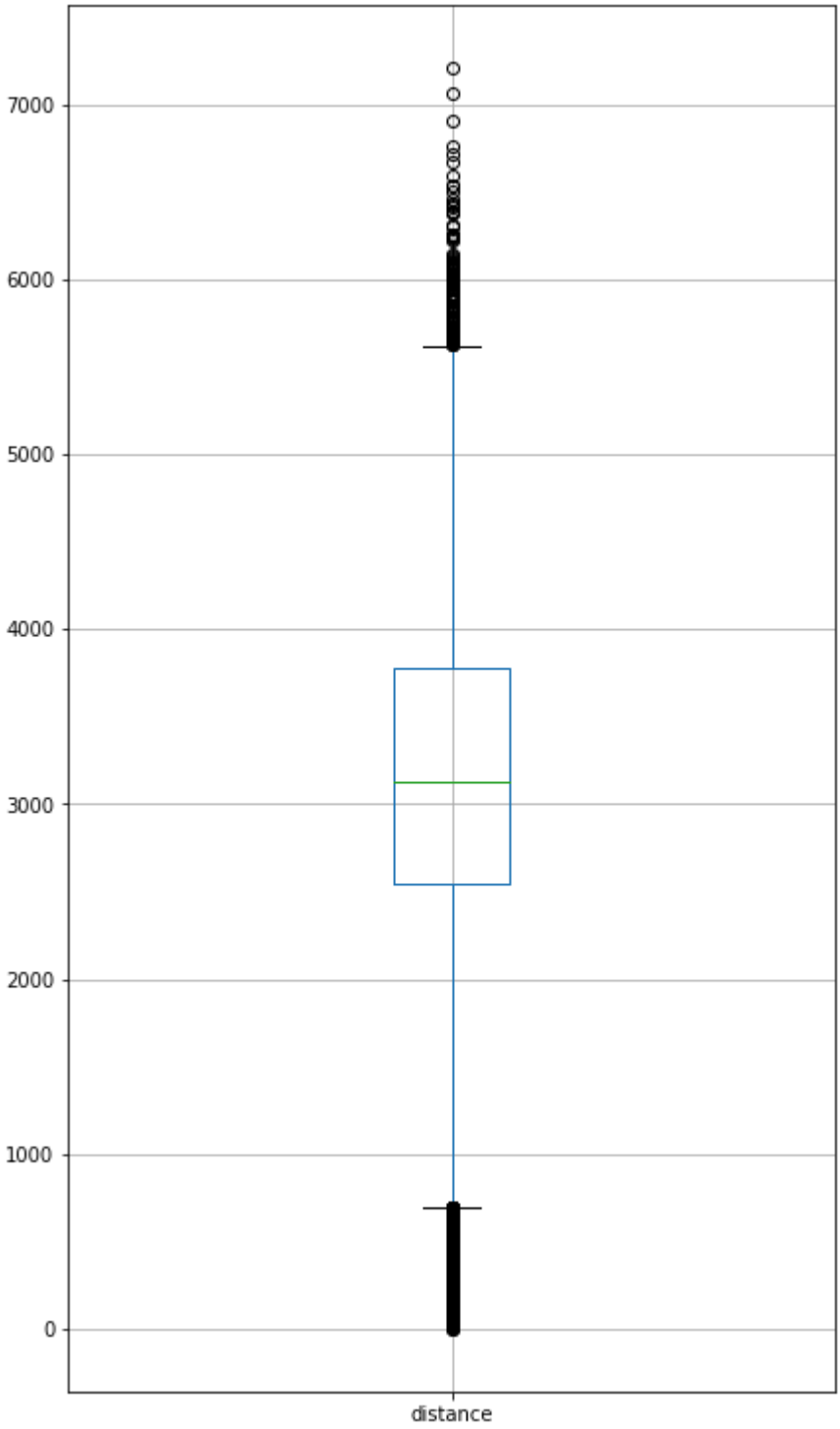
In [18]:
```
rides_data['date'] = pd.to_datetime(rides_data['date'], format='%Y-%m-%d')
```

In [19]:
```
rides_data['month'] = rides_data['date'].dt.month
rides_data.head(5)
```

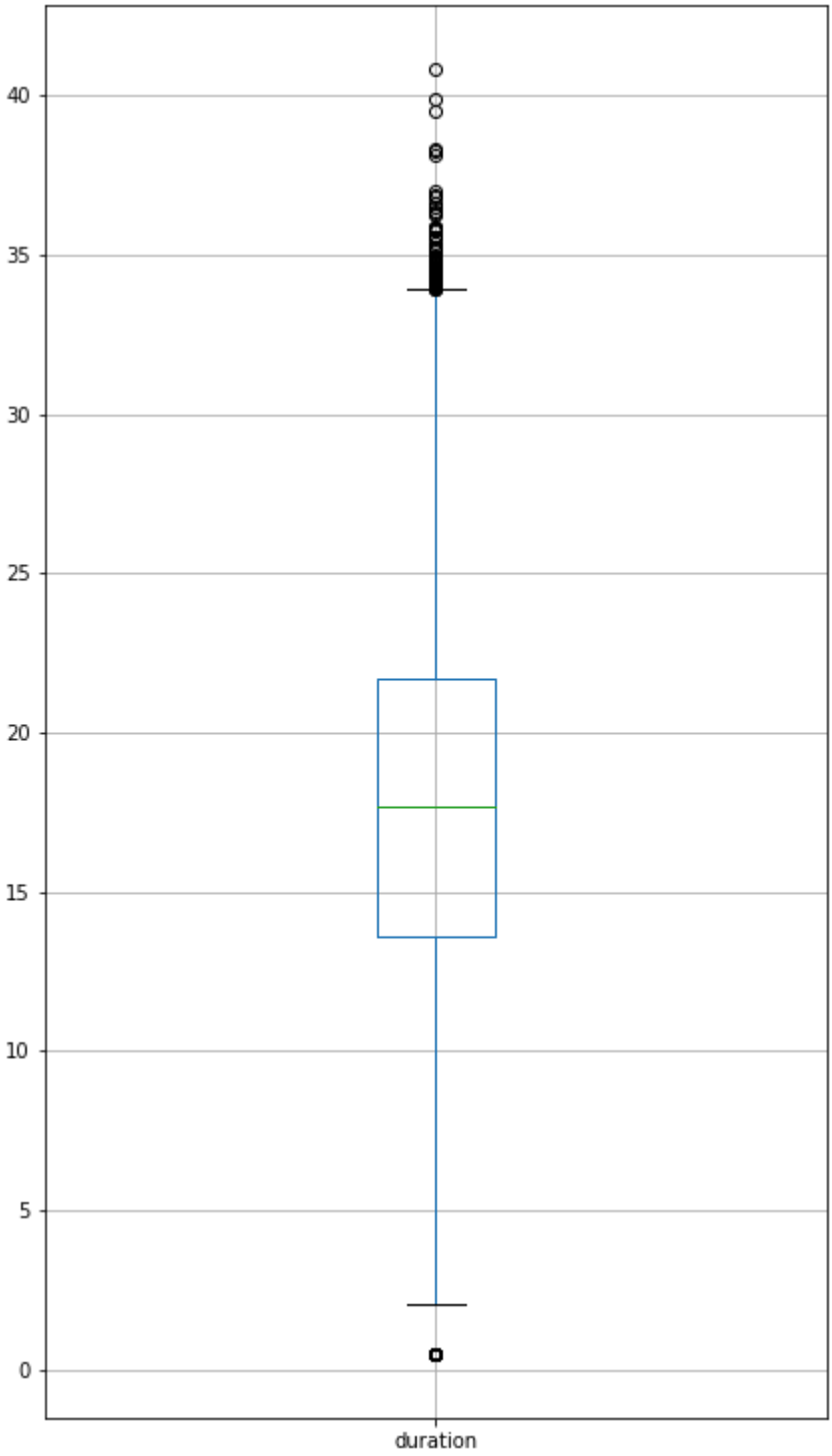
Out[19]:

	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

In [20]:
```
rides_data.boxplot(column='distance', figsize=(7,13))
plt.show()
```



In [21]:
```
rides_data.boxplot(column='duration', figsize=(7,13))
plt.show()
```



По иогам изучения информации содержащейся в датафреймах можно сделать несеолько выводов:

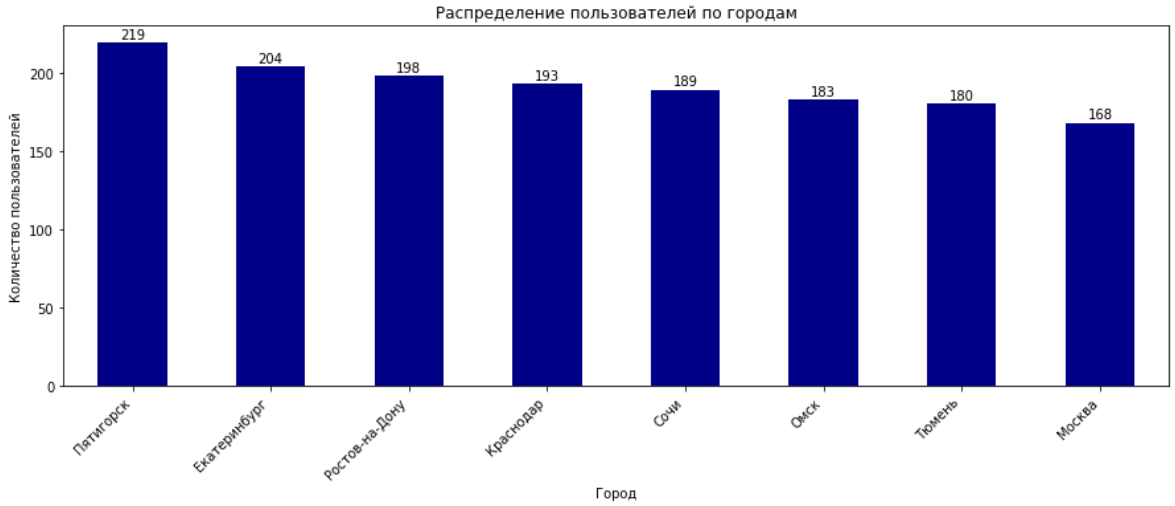
- в таблицах не содержится пропусков;
- было выявлено и удаленено из данных 31 дубликат;
- не явных дубликатов среди id пользователей не выявлено
количество строк в ДФ == длине уникальных id;
- имена столбцов корректны;
- типы данных в столбце с датой не соответсвуют требуемому типу, остальные столбцы соотвествуют;
- добавила в таблицу `rides_data` столбец "month" с номером месяца;
- по графику "ящик с усами" видно, что некоторые значения выходят за границы "усов", но вданном датасете будет не

целесообразно удалять эти данные, так как они представляют собой реальные наблюдения.

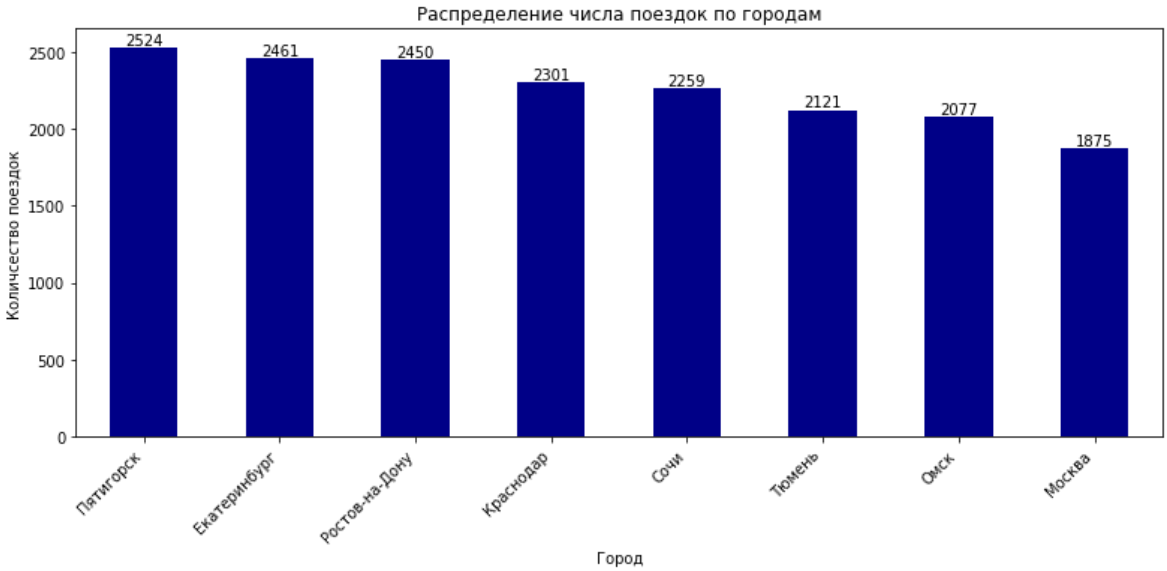
Далее можно приступать к исследовательскому анализу данных.

Исследовательский анализ данных

In [22]:
```
users_data.groupby('city')['city'].count().sort_values(ascending=False).plot(fig
for i, v in enumerate(users_data.groupby('city')['city'].count().sort_values(asc
    plt.text(i, v + 1, str(v), ha='center', va='bottom')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Город')
plt.ylabel('Количество пользователей')
plt.title('Распределение пользователей по городам')
plt.show()
```

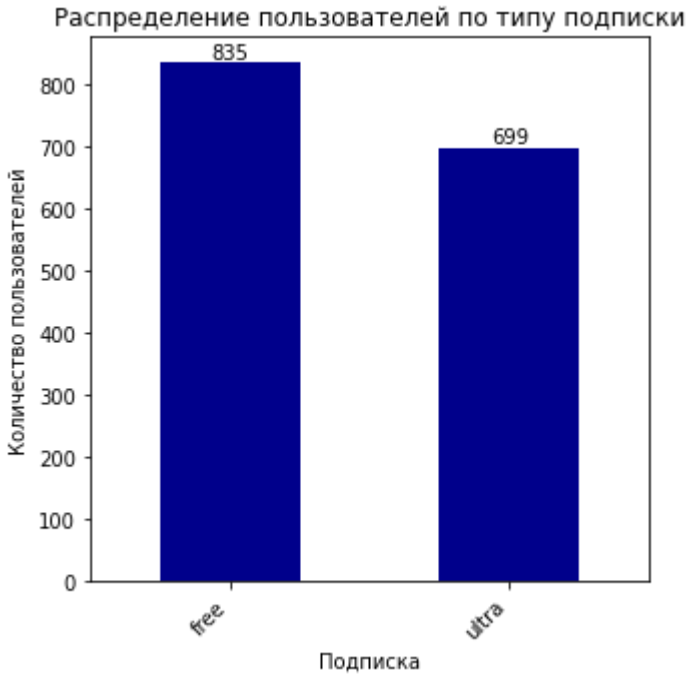


In [23]:
```
merged_data = pd.merge(users_data, rides_data, on='user_id', how='left')
rides_by_city = merged_data.groupby('city')['distance'].count().sort_values(asce
rides_by_city.plot(figsize=(13,5), kind='bar', color='DarkBlue')

for i, v in enumerate(rides_by_city):
    plt.text(i, v + 1, str(v), ha='center', va='bottom')

plt.xticks(rotation=45, ha='right')
plt.xlabel('Город')
plt.ylabel('Количсество поездок')
plt.title('Распределение числа поездок по городам')
plt.show()
```

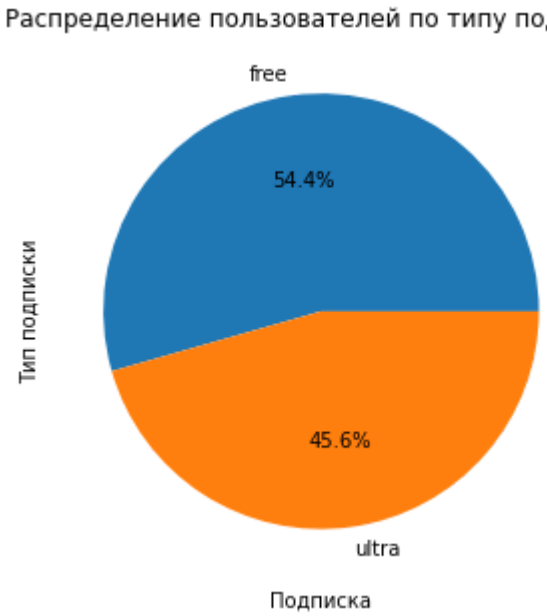


По графикам выше можно сделать выводы: - в г. Пятигорск самое большое количество зарегистрированных пользователей так же в этом городке совершается наибольшее число поездок; - в г. Москва наименьшее кол-во пользователей и также кол-во совершенных поездок;

In [24]:
```
users_data.groupby('subscription_type')['subscription_type'].count().plot(figsiz
for i, v in enumerate(users_data.groupby('subscription_type')['subscription_type
    plt.text(i, v + 1, str(v), ha='center', va='bottom')
plt.xticks(rotation=45, ha='right')
plt.xlabel('Подписка')
plt.ylabel('Количество пользователей')
plt.title('Распределение пользователей по типу подписки')
plt.show()
```



In [25]:
```
users_data.groupby('subscription_type')['subscription_type'].count().plot(figsiz
plt.xlabel('Подписка')
plt.ylabel('Тип подписки')
plt.title('Распределение пользователей по типу подписки')
plt.show()
```



По графикам выше можно сделать выводы:

- пользователей без подписки больше примерно на 9 % , что соответсвует 136 пользователям в нашей воборке.

In [26]:
```
plt.hist(users_data['age'], color='darkblue', label='Возраст', bins=30)
plt.xticks(rotation=0, ha='right')
plt.xlabel('Возраст')
plt.ylabel('Количество пользователей')
plt.title('Распределение пользователей по возрасту')
plt.legend()
plt.grid()
plt.show()
```

По графикам выше можно сделать выводы:

- самый юнный пользователь 12 лет, самый старший 43, самые частовстречаемые возраста у пользователей от 22 до 28, где 25 лет это самый частый возраст пользователей;
- целевой аудиторией данного сервиса можно считать молодежь от 20 лет до 35 лет.

In [27]:
```
# Средняя длина поездок каждого зарегистрированного пользователя
rides_data.groupby('user_id')['distance'].mean().sort_values(ascending=False)
```

Out[27]:
```
user_id
1309    4287.519387
850     4004.838037
644     3939.773328
465     3917.559390
381     3905.037651
           ...
883     2094.474537
1011    2065.919718
1433    1991.245493
1495    1767.760716
908     1630.788427
Name: distance, Length: 1534, dtype: float64
```

In [28]:
```
plt.hist(rides_data['distance'], color='darkblue', label='Дальность', bins=100)
plt.xticks(rotation=0, ha='right')
plt.xlabel('Дистанция (м)')
plt.ylabel('Количество пользователей')
plt.title('Распределение поездок по дальности')
plt.legend()
plt.grid()
plt.show()
```

По графикам выше можно сделать выводы:

- на графике дальности поездок можно увидеть большой пик в середине графика значений, что гооврит о нормальном "колоколообразном" распределии;
- можно увидеть небольшой пик на графике дальности поездок, что возможно связано с короткими поездакми пользователей "до метро" или при опоздании;
- наибольшая средняя дальности поездок у пользователя с id 1309 и ровняется 4287.52 м, самая наименьшая дальность у пользователя с id 908 с расстоянием 1630.79 м;

In [29]:
```
# Общая длина поездок каждого зарегистрированного пользователя
rides_data.groupby('user_id')['distance'].sum().sort_values(ascending=False)
```

Out[29]:
```
user_id
1063    79325.846482
1236    78744.976586
1361    77773.501423
1468    77392.080789
1374    74901.673366
           ...
412      8371.372987
594      7122.382755
73       6504.222851
366      5677.243951
342      5530.344048
Name: distance, Length: 1534, dtype: float64
```

In [30]:
```
plt.hist(rides_data['duration'], color='darkblue', label='Длительность', bins=10
plt.xticks(rotation=0, ha='right')
plt.xlabel('Длительность (мин)')
plt.ylabel('Количество пользователей')
plt.title('Распределение поездок по длительности')
plt.legend()
plt.grid()
plt.show()
```

По графикам выше можно сделать выводы:

- на графике длительности поездок можно увидеть большой пик в середине графика значений, что гооврит о нормальном "колоколообразном" распределии;
- наибольшая сумма дальности всех поездок у пользователя с id 1063 и ровняется 79325.84 м, самая наименьшая дальность у пользователя с id 342 с расстоянием 5530.34м.

Объединение данных

In [31]:
```
print(f'Рамеры датасетов:\nusers_data: {len(users_data)}\nrides_data: {len(rides
```

```
Рамеры датасетов:
users_data: 1534
rides_data: 18068
```

In [32]:
```
merged_data_user_ride = pd.merge(users_data, rides_data, on='user_id', how='left
without_subscr = merged_data_user_ride[merged_data_user_ride['subscription_type'
with_subscr = merged_data_user_ride[merged_data_user_ride['subscription_type'] !
```

In [33]:
```
print(f'После объединения размерность таблиц совпадает? {len(merged_data_user_ri
```

```
После объединения размерность таблиц совпадает? True
```

In [34]:
```
merged_data_user_ride.isna().sum()
```

Out[34]:
```
user_id              0
name                 0
age                  0
city                 0
subscription_type    0
distance             0
duration             0
date                 0
month                0
dtype: int64
```

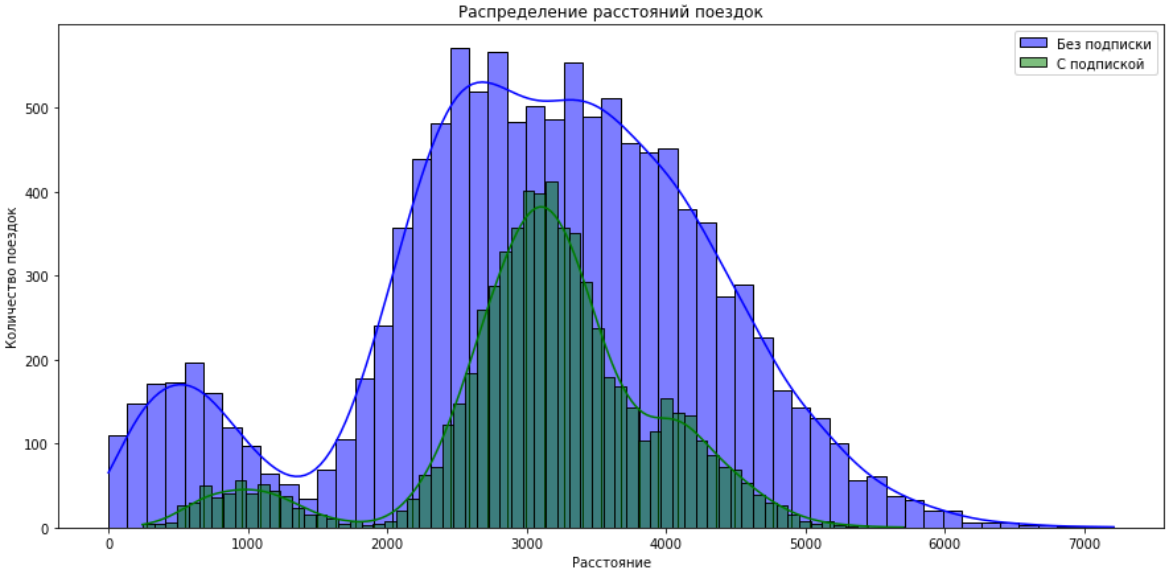
Пропусков нет, можно продолжать исследование.

In [35]:
```
merged_data_user_ride.head(5)
```

Out[35]:

	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 [36]:
```
plt.figure(figsize=(15, 7))
sns.histplot(without_subscr['distance'], kde=True, label='Без подписки', color='
sns.histplot(with_subscr['distance'], kde=True, label='С подпиской', color='gree
plt.xlabel('Расстояние')
plt.ylabel('Количество поездок')
plt.title('Распределение расстояний поездок')
plt.legend()
plt.show()
```



По графику выше можно сделать некоторые выводы:

- распределение по дальности и кол-ву поездок пользователей с подпиской и без дублируют друг друга и графики соответствуют нормальному распределению
- мы видим небольшой всплеск на расстояниях до 1000 м и далее снижение практичнски к нулю на 2000 м;
- этот же всплеск был виден на графике ранее, так как все поездки менее 5 минут были удалены из данных, то этот малый пик можно объяснить, что некоторые пользователи берут самокат для более быстрого преодоления короткого отрезка, например, при опоздании на встречу или на работу и т.п.;
- пользователи без подписки в среднем совершают большее кол-во поездок, на те же расстояния, что и пользователи без подписки.

Подсчёт выручки

In [37]:
```
details_of_rides = merged_data_user_ride.groupby(['user_id', 'month']).agg(
    total_distance=('distance', 'sum'),
    total_rides=('user_id', 'count'),
    total_duration=('duration', 'sum'),
    subscription_type=('subscription_type', 'first')
).reset_index()

details_of_rides.head(10)
```

Out[37]:

	user_id	month	total_distance	total_rides	total_duration	subscription_type
0	1	1	7027.511294	2	41.416640	ultra
1	1	4	754.159807	1	6.232113	ultra
2	1	8	6723.470560	2	44.776802	ultra
3	1	10	5809.911100	2	31.578017	ultra
4	1	11	7003.499363	3	53.397424	ultra
5	1	12	6751.629942	2	27.203912	ultra
6	2	3	10187.723006	3	60.959959	ultra
7	2	4	6164.381824	2	39.048633	ultra
8	2	6	3255.338202	1	13.851127	ultra
9	2	7	6780.722964	2	46.170157	ultra

In [38]:
```
details_of_rides['total_distance'] = np.ceil(details_of_rides['total_distance'])
details_of_rides['total_rides'] = np.ceil(details_of_rides['total_rides'])
details_of_rides['total_duration'] = np.ceil(details_of_rides['total_duration'])
details_of_rides
```

Out[38]:

	user_id	month	total_distance	total_rides	total_duration	subscription_type
0	1	1	7028.0	2.0	42.0	ultra
1	1	4	755.0	1.0	7.0	ultra
2	1	8	6724.0	2.0	45.0	ultra
3	1	10	5810.0	2.0	32.0	ultra
4	1	11	7004.0	3.0	54.0	ultra
...	...	...	...	...	...	...
11326	1534	6	3410.0	2.0	25.0	free
11327	1534	8	7623.0	2.0	48.0	free
11328	1534	9	4929.0	1.0	23.0	free
11329	1534	11	13351.0	4.0	77.0	free
11330	1534	12	2372.0	1.0	16.0	free

11331 rows × 6 columns

In [39]:
```
def revenue_calculation(data):

    subscription_data = {
        'ultra': {'minute_price': 6, 'subscription_fee': 199, 'start_ride_price'
        'free': {'minute_price': 8, 'subscription_fee': 0, 'start_ride_price': 5
    }

    if data['subscription_type'] == 'ultra':
        return (subscription_data['ultra']['minute_price'] * data['total_duratio
```

```
                + subscription_data['ultra']['subscription_fee']

    elif data['subscription_type'] == 'free':
        return (subscription_data['free']['minute_price'] * data['total_duration
               (subscription_data['free']['start_ride_price'] * data['total_ride

    else:
        return 0
```

In [40]:
```
details_of_rides['income'] = details_of_rides.apply(revenue_calculation, axis=1)
details_of_rides
```

Out[40]:

	user_id	month	total_distance	total_rides	total_duration	subscription_type	inc
0	1	1	7028.0	2.0	42.0	ultra	
1	1	4	755.0	1.0	7.0	ultra	
2	1	8	6724.0	2.0	45.0	ultra	
3	1	10	5810.0	2.0	32.0	ultra	
4	1	11	7004.0	3.0	54.0	ultra	
...	...	...	...	...	...	...	
11326	1534	6	3410.0	2.0	25.0	free	
11327	1534	8	7623.0	2.0	48.0	free	
11328	1534	9	4929.0	1.0	23.0	free	
11329	1534	11	13351.0	4.0	77.0	free	
11330	1534	12	2372.0	1.0	16.0	free	

11331 rows × 7 columns

In [42]:
```
details_of_rides_with_subscr = details_of_rides[details_of_rides['subscription_t
details_of_rides_without_subscr = details_of_rides[details_of_rides['subscriptio
details_of_rides_with_subscr
```

Out[42]:

	user_id	month	total_distance	total_rides	total_duration	subscription_type	inco
0	1	1	7028.0	2.0	42.0	ultra	4
1	1	4	755.0	1.0	7.0	ultra	2
2	1	8	6724.0	2.0	45.0	ultra	4
3	1	10	5810.0	2.0	32.0	ultra	3
4	1	11	7004.0	3.0	54.0	ultra	5
...	...	...	...	...	...	...	
4528	699	6	4073.0	1.0	17.0	ultra	3
4529	699	8	7019.0	2.0	45.0	ultra	4
4530	699	9	6365.0	2.0	31.0	ultra	3
4531	699	10	4708.0	1.0	16.0	ultra	2
4532	699	12	3203.0	1.0	26.0	ultra	3

4533 rows × 7 columns

In [43]:
```
details_of_rides_without_subscr
```

Out[43]:

	user_id	month	total_distance	total_rides	total_duration	subscription_type	inc
4533	700	1	2516.0	1.0	15.0	free	
4534	700	2	13447.0	5.0	86.0	free	
4535	700	3	3799.0	1.0	19.0	free	
4536	700	4	2985.0	1.0	16.0	free	
4537	700	6	5928.0	2.0	29.0	free	
...	...	...	...	...	...	...	
11326	1534	6	3410.0	2.0	25.0	free	
11327	1534	8	7623.0	2.0	48.0	free	
11328	1534	9	4929.0	1.0	23.0	free	
11329	1534	11	13351.0	4.0	77.0	free	
11330	1534	12	2372.0	1.0	16.0	free	

6798 rows × 7 columns

In [44]:
```
print('Прибыль от поездок пользователей с подпиской составила:', round(details_o
```

```
Прибыль от поездок пользователей с подпиской составила: 1638597.0
```

In [45]:
```
print('Прибыль от поездок пользователей без подписки составила:', round(details_
```

```
Прибыль от поездок пользователей без подписки составила: 2215080.0
```

In [46]:
```
print('Общая прибыль от поездок всех пользователей составила:', round(details_of
```

```
Общая прибыль от поездок всех пользователей составила: 3853677.0
```

Проверка гипотез

1. Гипотезы:

- H0: Среднее время поездок для подписчиков равно среднему времени поездок для пользователей без подписки.
- H1: Среднее время поездок для подписчиков превышает среднего времени поездок для пользователей без подписки.

In [47]:
```
alpha = 0.05

result = stats.ttest_ind(with_subscr['duration'], without_subscr['duration'],  e

print(f"p-value: {result.pvalue}")

if result.pvalue < alpha:
    print("Отвергаем нулевую гипотезу: \nпользователи с подпиской тратят больше
else:
    print("Не отвергаем нулевую гипотезу: \nнет достаточных оснований утверждать
```

```
p-value: 5.6757813771289775e-37
Отвергаем нулевую гипотезу:
пользователи с подпиской тратят больше времени на поездки.
```

Выводы:

- для проверки гипотезы о равенстве двух выборок (по наличию подписки) использовала метод ttest_ind из библиотеки scipy.stats
- использовала t-тест для независимых выборок, так как группы пользователей с подпиской и без подписки не связаны;
- значение p-value получилось очень маленьким, что позволило отвергнуть гипотезу;
- пользователи с подпиской тратят больше времени на поездки.

2. Гипотезы:

- H0: Среднее расстояние поездок для подписчиков равно 3130 метров.
- H1: Среднее расстояние поездок для подписчиков превышает 3130 метров.

In [48]:
```
alpha = 0.05
result = stats.ttest_1samp(with_subscr['distance'], 3130, alternative='greater')

print(f"p-value: {result.pvalue}")

if result.pvalue < alpha:
    print("Отвергаем нулевую гипотезу: \nсреднее расстояние поездок для подписчи
else:
    print("Не отвергаем нулевую гипотезу: \nнет достаточных оснований утверждать
```

```
p-value: 0.9195368847849785
Не отвергаем нулевую гипотезу:
нет достаточных оснований утверждать, что среднее расстояние поездок для подписчи
ков превышает 3130 метров.
```

Выводы:

- провела односторонний t-тест, так как необходимо проверить не превышает ли дальность поездок 3130 метров;
- в параметре метода ttest_1samp библиотеки scipy.stats параметр alternative='greater', так как мы проверяем правый хвост (превышает ли значение)
- по результатам проверки можно сделать вывод, что дальность поездок не превышает расстояния 3130 метров — оптимальное с точки зрения износа самоката.

3. Гипотезы:

- H0: Средняя ежемесячная выручка от пользователей с подпиской равна средней ежемесячной выручке от пользователей без подписки.
- H1: Средняя ежемесячная выручка от пользователей с подпиской выше, чем средняя ежемесячная выручка от пользователей без подписки.

In [49]:
```
alpha = 0.05
result = stats.ttest_ind(details_of_rides_with_subscr['income'], details_of_ride
                         equal_var=False, alternative='greater')

print(f"p-value: {result.pvalue}")

if result.pvalue < alpha:
    print("Отвергаем нулевую гипотезу: \nсредняя ежемесячная выручка от пользова
else:
    print("Не отвергаем нулевую гипотезу: \nнет достаточных оснований утверждать
```

```
p-value: 1.0718729651261336e-44
Отвергаем нулевую гипотезу:
средняя ежемесячная выручка от пользователей с подпиской выше, чем средняя ежемес
ячная от пользователей без подписки.
```

In [50]:
```
(details_of_rides_without_subscr['income'].sum() / details_of_rides_without_subs
```

Out[50]: True

Выводы:

- пользователи с подпиской приносят бОльший доход компании;
- это следует из проверок выше, так как пользователи с подпиской больше времени тратят на поездки;

Для компании пользователи с подпиской более "выгодны", чем пользователи без подписки.

4. Задание

Можно сформулировать 2 гипотезы нулевую и альтернативную одностороннюю:

H0: количество обращений в техподдержку не изменилось, осталось на том же уровне, что и до обновлений;
H1: количество обращений в техподдержку снизилось,по сравнению с периодами до добавления обвлений.

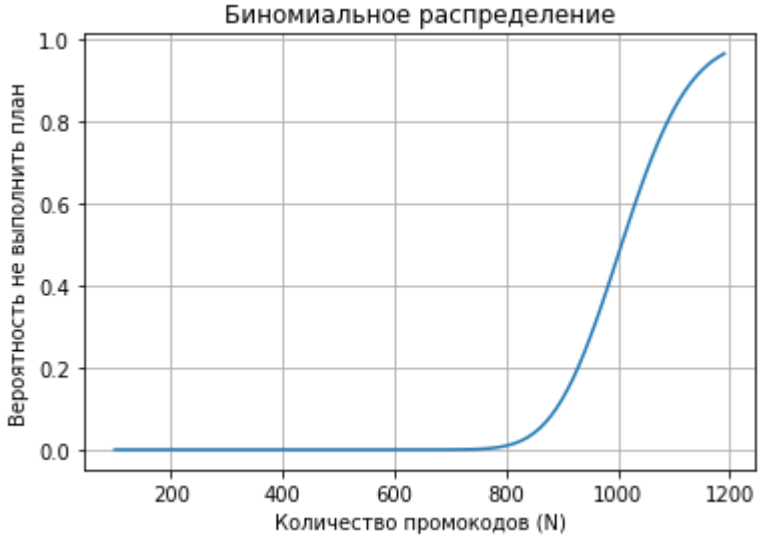
тест для зависимых выборок ttest_rel()

В данном примере будет целосообразно использовать парный t-тест для зависимых выборок ttest_rel() из библиотеки scipy.stats, указав в параметрах функции alternative='less' (так мы проверям снижение).
Данные проверяются из одной и той же выборки только до и после изменений, это говорит об их зависимоти.
В этом случае будет проверятся есть ли значимое изменение в кол-ве обращений после обновлений.

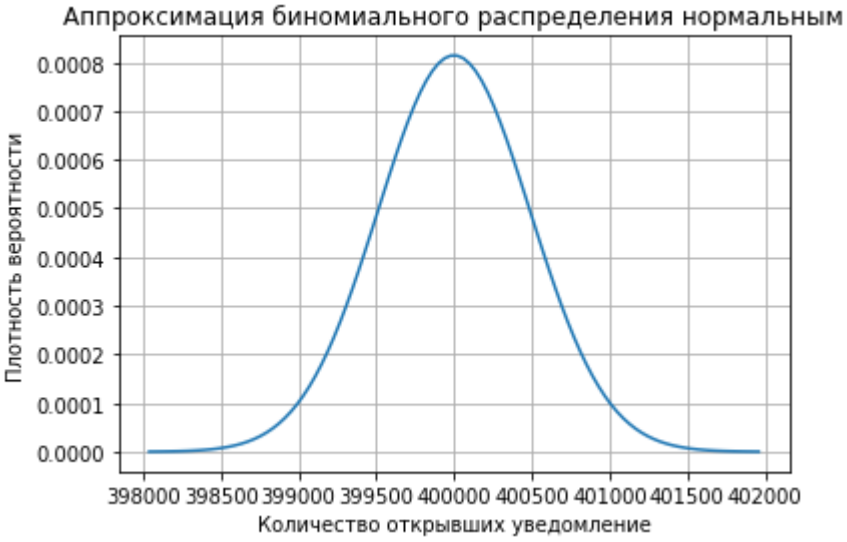
Распределения

In [51]:
```
p = 0.1 # Вероятность успеха
minimum = 100 # Мат. ожидание

values = np.arange(100, 1200, 10) # Задаю диапазон значений
for n in values:
    if round((stats.binom.cdf(minimum, n, p)), 2) <= 0.05:
        print(f'При рассылке {n} промокодов вероятность не выполнить план будет
        break
cdf_values = stats.binom.cdf(minimum, values, p) # Вероятности биноминального ра
plt.plot(values, 1 - cdf_values) # График по x: значения, по y: вероятности
plt.xlabel('Количество промокодов (N)')
plt.ylabel('Вероятность не выполнить план')
plt.title('Биномиальное распределение')
plt.grid()
plt.show()
```

При рассылке 1170 промокодов вероятность не выполнить план будет 5% (p = 0.05)



In [52]:
```
n = 1_000_000 # Число испытаний
p = 0.4 # Вероятность успеха
expected_users = 399_500 # Ожидание

# Аппроксимация биноминального ожидания по формуле
mu = n * p # Среднее
sigma = np.sqrt(n * p * (1 - p)) #Стандартное отклонение

x = np.linspace(mu - 4 * sigma, mu + 4 * sigma, 100) # массив x в диапозон 4-х м
y = stats.norm.pdf(x, mu, sigma) # Плотность вероятности для каждого x

plt.plot(x, y)
plt.title('Аппроксимация биномиального распределения нормальным')
plt.xlabel('Количество открывших уведомление')
plt.ylabel('Плотность вероятности')
plt.grid()
plt.show()
```



In [53]:
```
prob = stats.norm.cdf(expected_users, mu, sigma)
print(f"Вероятность, что уведомление откроют не более 399_500 пользователей:", r
```

```
Вероятность, что уведомление откроют не более 399_500 пользователей: 0.15
```

Вероятность, что уведомление откроют не более 399_500 тыс. пользователей равна примерно 0.15, самую высокую плотность вероятности можно наблюдать на пике графика у кол-ва 400_000

Общие выводы:

1. В г. Пятигорск сервис аренды самокатов GoFast пользуется наибольшей популярностью, по сравнению с другими городами представленными в датасете;
2. В сервисе преобладают пользователи без подписки 54,4% от общего числа пользователей;
3. Целевой аудиторией данного сервиса явля.тся люди молого возраста от 20 до 35 лет;
4. Самое частое расстояние поездок от 2 до 5 км, что занимает от 8 до 30 мин;
5. В среднем выручка за представленный период составила 3_853_677 руб, из чего примерно 57,5 % (2_215_080) принесли пользователи без подписки, а в свою

очередь пользователи с подпиской принесли прибыли 1_638_597, что равняется 42,5% от общей суммы дохода.

Рекомендации заказчикам:

1. Предлагаю обратить своё внимание на развите сервиса в г. Москва, так как город имеет большой потенциал в принесении дохода, так как многогие молодие люди перееезжают в г. Москва для работы и жизни, они как раз и являются целевой аулиторией сарвиса;
2. По итогам проверки гипотез вношу предложение сосредоточиться на прибавлении пользователей с подпиской. Так как они оказались наиболее "выгодны" в плане доходности и так же менее изнашивают самокаты, так как в среднем ездят на меньшие расстояния, чем пользователи без подписки;
3. Так же пользователей с подпиской можно считать более аккуратными, так как они при условии меньшего расстояния, тратят больше времени, что можно интерпритировать, что пользователи с подпиской есздят более медленно и такие пользователи для сервиса более "желанны" и "выгодны" с точки зрения сохранения самакатов, точнее сохранения их срока службы.

In [ ]: