To combat the outflow, the customer service department of Bodybuilder Data Scientist has digitized a lot of client questionnaires. Your task is to analyze and prepare an action plan for customer retention. Namely: • learn to predict the probability of churn (at the level of the next month) for each client; • to form typical portraits of clients: to identify several of the most striking groups and characterize their main properties; • analyze the main features that most strongly affect the outflow; • formulate the main conclusions and develop recommendations for improving the quality of work with clients: o 1) identify target groups of customers; o 2) propose measures to reduce outflow; o 3) identify other features of interaction with customers.

Step 1. Download the data

"Data-Scientist Bodybuilder" provided information in csv files. The customer has prepared data that contains data for the month before the outflow and the fact of outflow for a certain month. The dataset includes the following fields: • Customer data for the previous month before checking the fact of churn: o 'gender' — gender; o 'Near_Location' – living or working in the area where the fitness center is located; o 'Partner' – an employee of the club's partner company (cooperation with companies whose employees can receive discounts on subscriptions – in this case, the fitness center stores information about the client's employer); o Promo_friends — the fact of the initial registration within the framework of the "Bring a friend" promotion (used a promo code from a friend when paying for the first subscription); o 'Phone' — availability of a contact phone number; o 'Age' — age; o 'Lifetime' - the time since the first visit to the fitness center (in months). • Information based on the history of visits, purchases and information about the current status of the client's subscription: o 'Contract_period' — duration of the current current subscription (month, 6 months, year); o 'Month_to_end_contract' — the period until the end of the current current subscription (in months); o 'Group_visits' — the fact of attending group classes; o 'Avg_class_frequency_total' — the average frequency of visits per week for the entire time since the beginning of the subscription; o 'Avg_class_frequency_current_month' is the average frequency of visits per week for the previous month; o 'Avg_additional_charges_total' — total revenue from other services of the fitness center: café, sporting goods, beauty and massage parlor. • 'Churn' — the fact of outflow in the current month.

In [1]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
#графики в svg выглядят более четкими
%config InlineBackend.figure_format = 'svg'

gym = pd.read_csv('gym_churn.csv')
display(gym)
```

	gender	Near_Location	Partner	Promo_friends	Phone	Contract_period	Group_visits	Age	Av
0	1	1	1	1	0	6	1	29	
1	0	1	0	0	1	12	1	31	
2	0	1	1	0	1	1	0	28	
3	0	1	1	1	1	12	1	33	
4	1	1	1	1	1	1	0	26	
...	...	...	...	...	...	...	...	...	
3995	1	1	1	0	1	12	0	33	
3996	0	1	0	0	1	1	1	29	
3997	1	1	1	1	1	12	0	28	
3998	0	1	1	1	1	6	0	32	
3999	1	0	1	0	0	12	1	30	

4000 rows × 14 columns

Step 2. Conduct exploratory data analysis (EDA)

2.1 Look at the dataset: whether there are missing values (omissions) in it, study the mean values and standard deviations (the describe() method will come in handy);

In [2]:
```
#Пропусков нет, в каждом столбце по 4000 наблюдения.
# Видим, что все признаки — числовые.
gym.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype  
---  ------                             --------------  -----  
 0   gender                             4000 non-null   int64  
 1   Near_Location                      4000 non-null   int64  
 2   Partner                            4000 non-null   int64  
 3   Promo_friends                      4000 non-null   int64  
 4   Phone                              4000 non-null   int64  
 5   Contract_period                    4000 non-null   int64  
 6   Group_visits                       4000 non-null   int64  
 7   Age                                4000 non-null   int64  
 8   Avg_additional_charges_total       4000 non-null   float64
 9   Month_to_end_contract              4000 non-null   float64
 10  Lifetime                           4000 non-null   int64  
 11  Avg_class_frequency_total          4000 non-null   float64
 12  Avg_class_frequency_current_month  4000 non-null   float64
 13  Churn                              4000 non-null   int64  
dtypes: float64(4), int64(10)
memory usage: 437.6 KB
```

In [3]:
```
gym['Avg_additional_charges_total'] = gym['Avg_additional_charges_total'].astype('int6
gym['Month_to_end_contract'] = gym['Month_to_end_contract'].astype('int64')
gym['Avg_class_frequency_total'] = gym['Avg_class_frequency_total'].astype('int64')
gym['Avg_class_frequency_current_month'] = gym['Avg_class_frequency_current_month'].as
```

In [4]:
```
gym.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 14 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   gender                             4000 non-null   int64
 1   Near_Location                      4000 non-null   int64
 2   Partner                            4000 non-null   int64
 3   Promo_friends                      4000 non-null   int64
 4   Phone                              4000 non-null   int64
 5   Contract_period                    4000 non-null   int64
 6   Group_visits                       4000 non-null   int64
 7   Age                                4000 non-null   int64
 8   Avg_additional_charges_total       4000 non-null   int64
 9   Month_to_end_contract              4000 non-null   int64
 10  Lifetime                           4000 non-null   int64
 11  Avg_class_frequency_total          4000 non-null   int64
 12  Avg_class_frequency_current_month  4000 non-null   int64
 13  Churn                              4000 non-null   int64
dtypes: int64(14)
memory usage: 437.6 KB
```

In [5]:
```
## изучите средние значения и стандартные отклонения.
gym.describe()
```

Out[5]:

	gender	Near_Location	Partner	Promo_friends	Phone	Contract_period	Group_
count	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.00
mean	0.510250	0.845250	0.486750	0.308500	0.903500	4.681250	0.41
std	0.499957	0.361711	0.499887	0.461932	0.295313	4.549706	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.00
25%	0.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.00
50%	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	0.00
75%	1.000000	1.000000	1.000000	1.000000	1.000000	6.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	12.000000	1.00

Метод describe показывает основные статистические характеристики данных по каждому числовому признаку (типы int64 и float64): число непропущенных значений, среднее, стандартное отклонение, диапазон, медиану, 0.25 и 0.75 квартили.

2.2 Посмотрите на средние значения признаков в двух группах — тех, кто ушел в отток и тех, кто остался

(воспользуйтесь методом groupby());

In [6]:
```
# Посмотрите на средние значения признаков в двух группах — тех, кто ушел в отток и те
gym_mean=gym.groupby('Churn').mean()
gym_mean
```

Out[6]:

	gender	Near_Location	Partner	Promo_friends	Phone	Contract_period	Group_visits	
Churn								
0	0.510037	0.873086	0.534195	0.353522	0.903709	5.747193	0.464103	29.!
1	0.510839	0.768143	0.355325	0.183789	0.902922	1.728558	0.268615	26.!

In [7]:
```
#  какова доля оттока в нашем датафрейме?
gym['Churn'].mean()
```

Out[7]:
```
0.26525
```

ВЫВОД: 26,5% — довольно плохой показатель для компании, с таким процентом оттока можно и разориться.

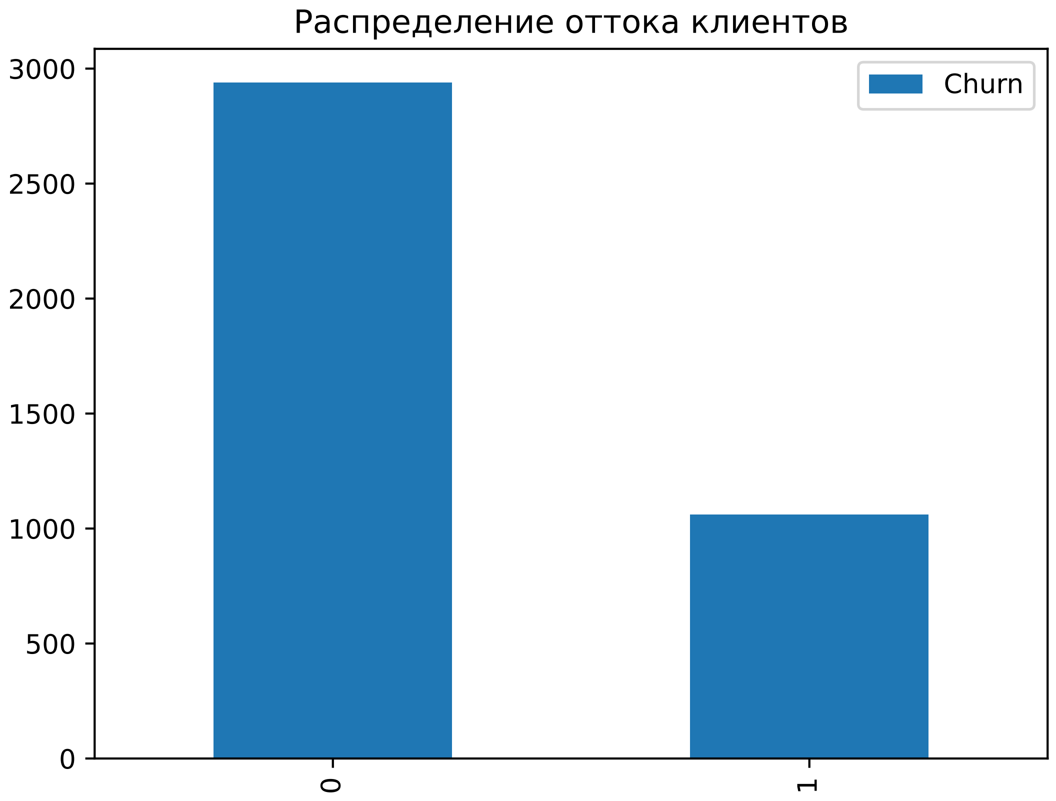
2.3 Постройте столбчатые гистограммы и распределения признаков для тех, кто ушёл (отток) и тех, кто остался (не попали в отток);

In [8]:
```
gym['Churn'].value_counts()
```

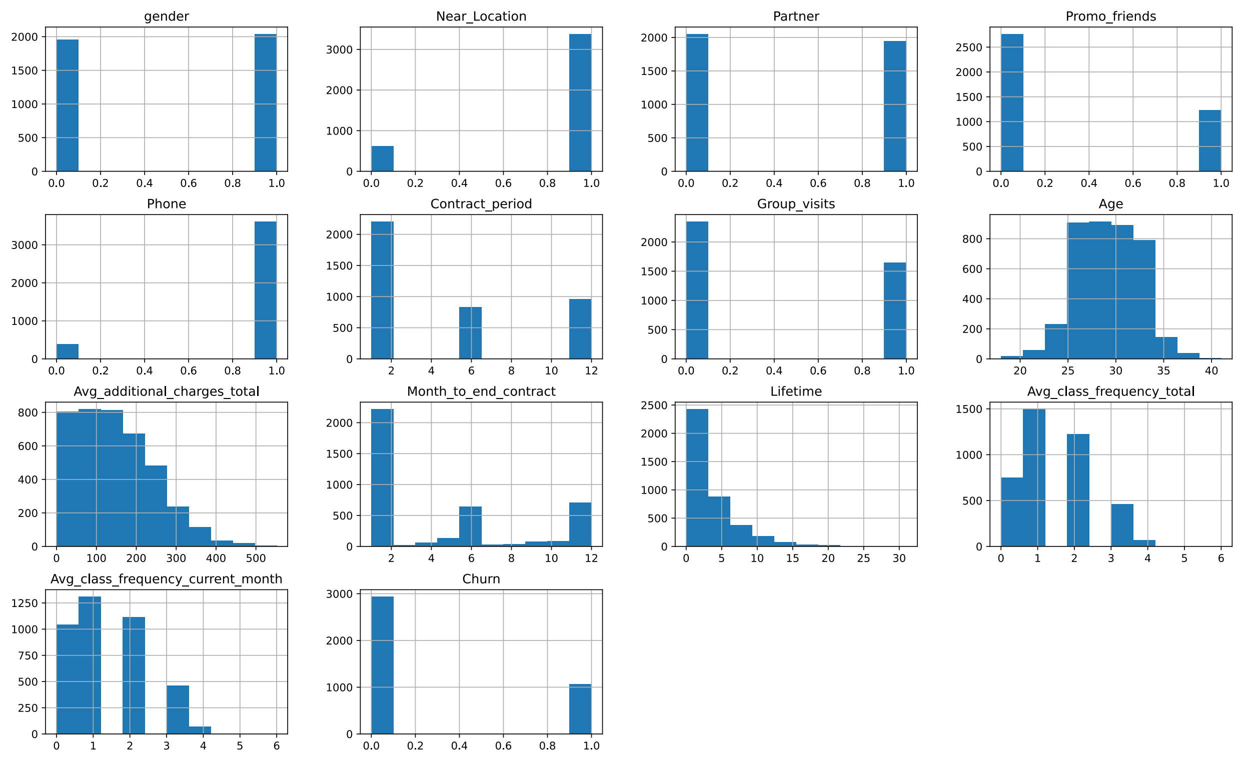
Out[8]:
```
0    2939
1    1061
Name: Churn, dtype: int64
```

CONCLUSION: 2939 - stayed and 1061 - left

In [9]:
```
gym['Churn'].value_counts().plot(kind='bar', label='Churn')
plt.legend()
plt.title('Распределение оттока клиентов');
```



In [10]:
```
# Теперь посмотрим на распределения всех интересующих нас количественных признаков.
gym.hist(figsize=(20,12));
```



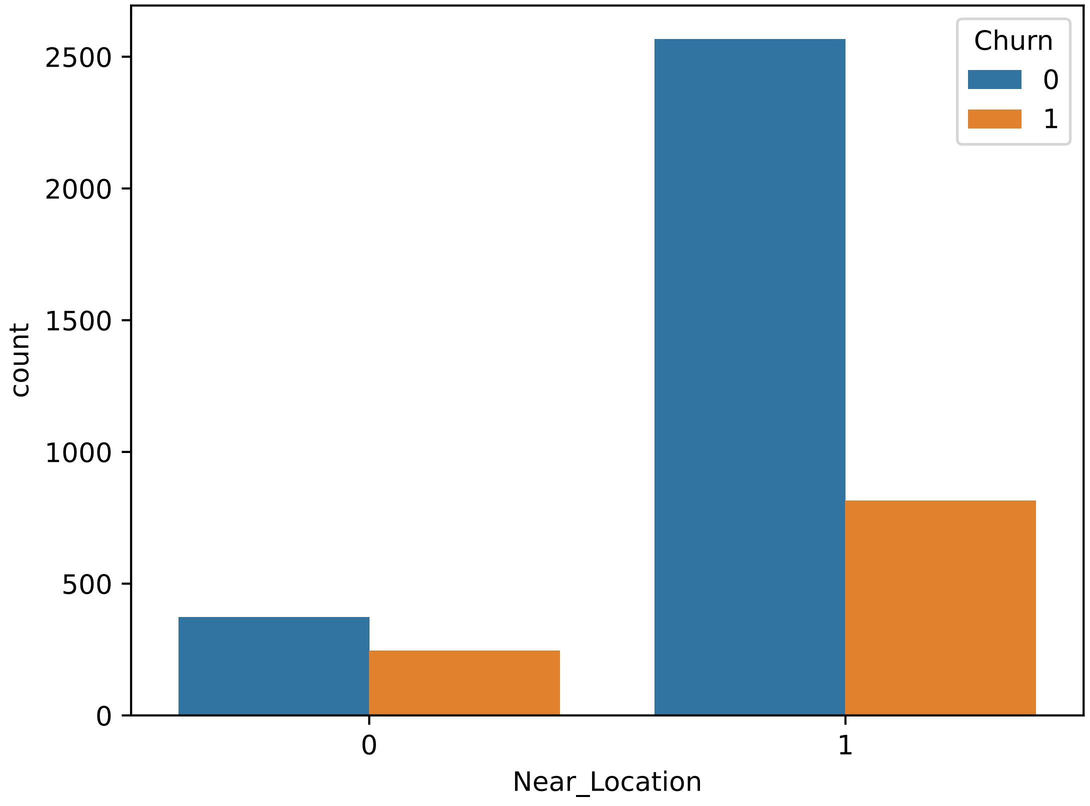
CONCLUSION: We see that most of the features are distributed normally. Exceptions are Lifetime, Avg_additional_charges_total, Avg_class_frequency_current_month, Near_Location

Now let's depict the distribution:

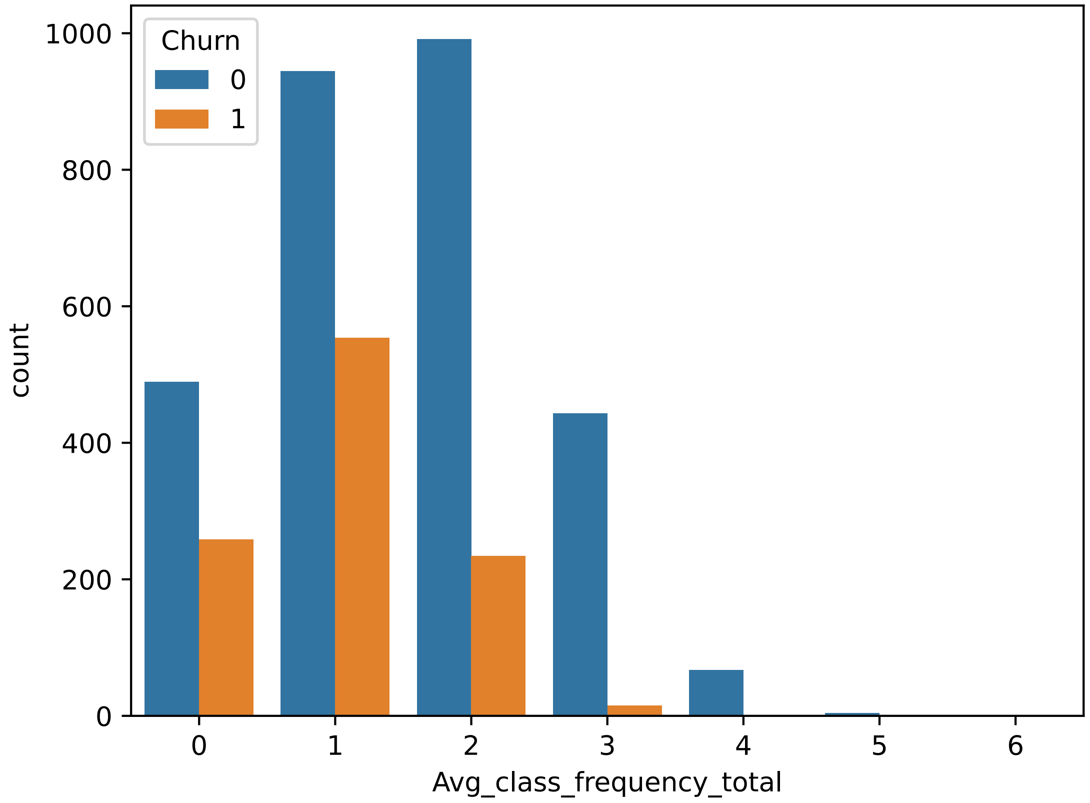
- 'Lifetime' - the time since the first visit to the fitness center (in months).

- 'Near_Location' - living or working in the area where the fitness center is located;
- 'Avg_class_frequency_total' — the average frequency of visits per week for the entire time since the beginning of the subscription;
- 'Avg_additional_charges_total' — total revenue from other services of the fitness center: café, sporting goods, beauty and massage parlor.

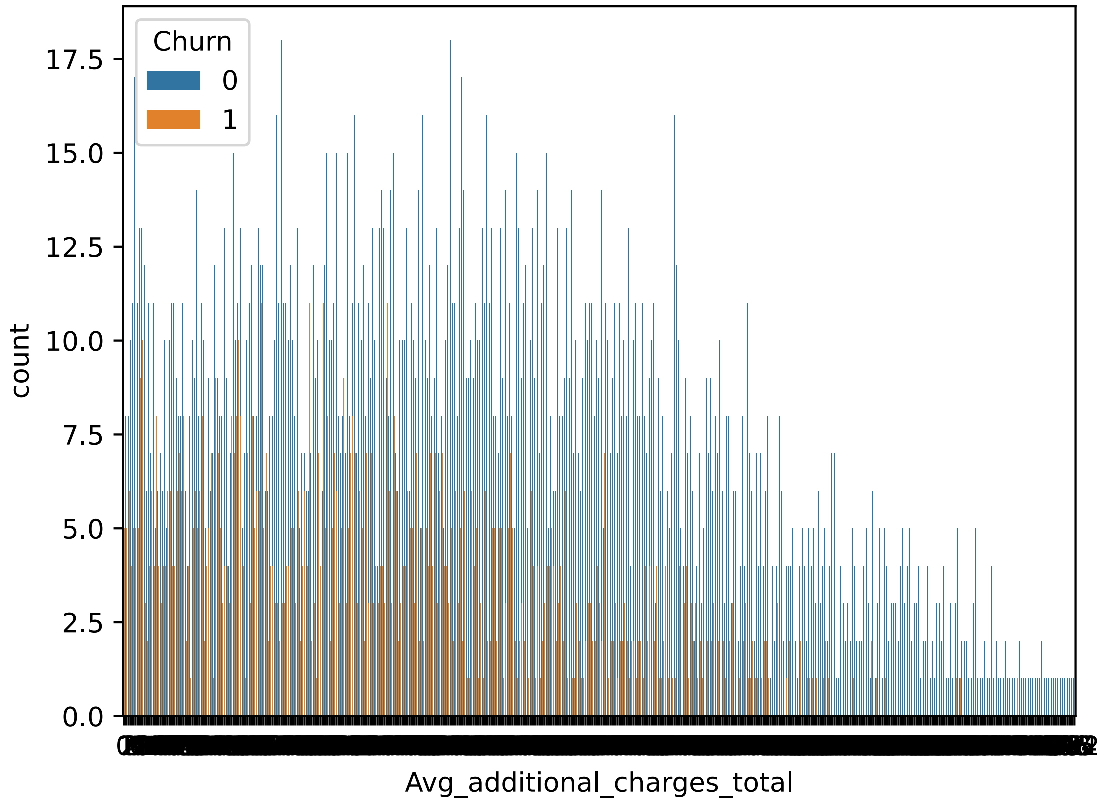
In [11]:
```
sns.countplot(x='Near_Location', hue='Churn', data=gym);
```



In [12]:
```
sns.countplot(x='Avg_class_frequency_total', hue='Churn', data=gym);
```



In [13]:
```
sns.countplot(x='Avg_additional_charges_total', hue='Churn', data=gym);
```



ВЫВОД:

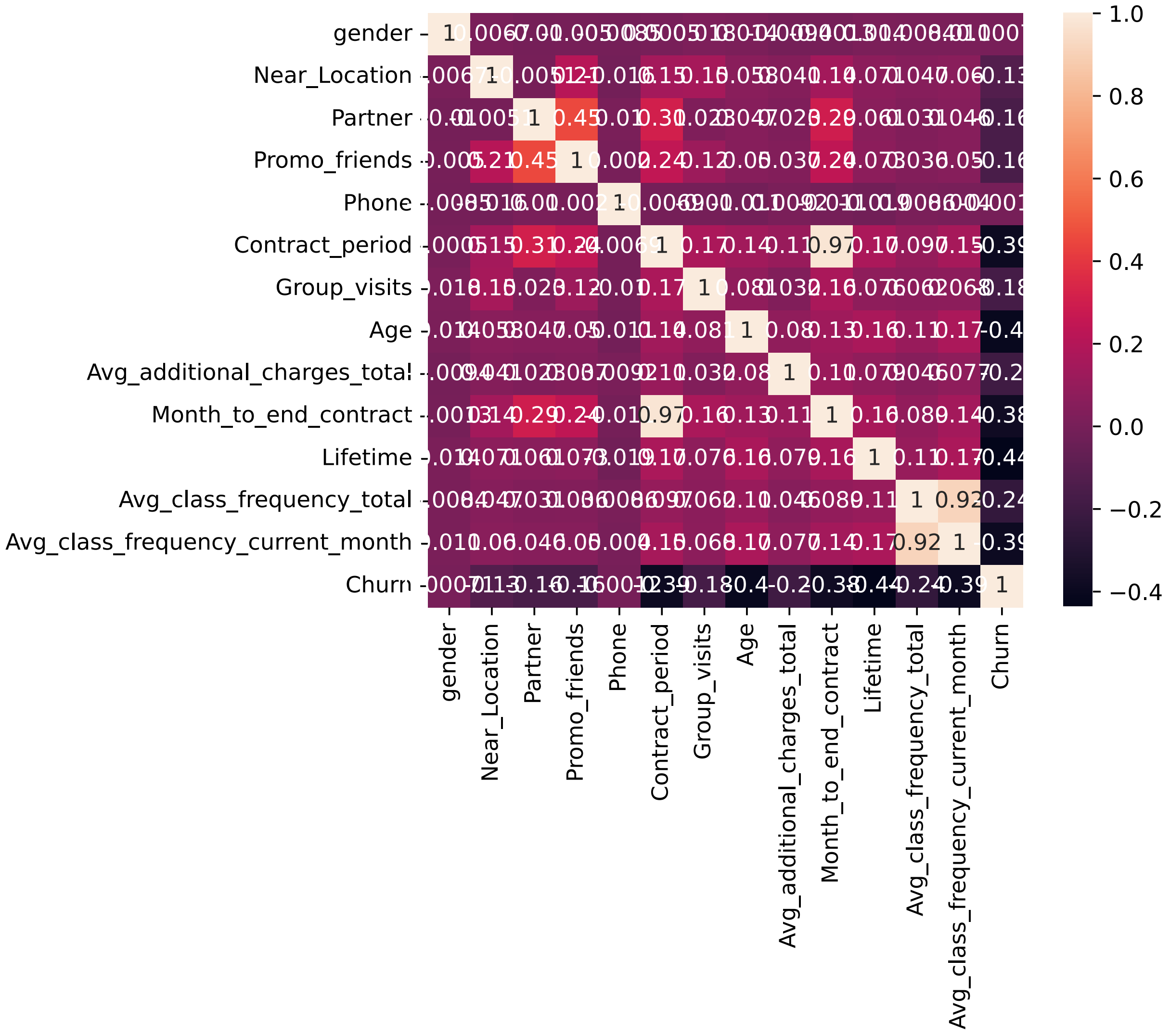
Доля оттока зависит как близко находится фитнес центр к дому и работе. Чем дальше от дома- тем больше процент оттока, больше 50% После первых двух посещений - отток уменьшается. То есть надо сделать всё чтобы клиент пришёл второй раз. Чем больше суммарная выручка от других услуг фитнес-центра - тем меньше отток. Надо заинтересовать клиента дополнительными спортивными коктейлями, косметическими средствами, массажем.

2.4 Постройте матрицу корреляций и отобразите её.

In [14]:
```
 # Постройте матрицу корреляций и отобразите её
corr_m = gym.corr()
corr_m

# нарисуем heatmap
sns.heatmap(corr_m, square = True, annot=True)

plt.figure(figsize = (15,15))
plt.show()
```



```
<Figure size 1500x1500 with 0 Axes>
```

Построим корреляционную матрицу числовых признаков набора данных. Визуализируем её методом heatmap(). Что можно сказать о корреляции между целевой переменной и остальными признаками? Есть ли признаки, которые коррелируют с целевой переменной более, чем на 0.9 по модулю? Это позволит сделать предположение об их влиянии на целевую переменную даже без построения модели.

There are no bright outliers and distortions in the diagram. This means that you can build a model on this data. As for correlates and pairwise graphs, half of the features correlate quite strongly with the target change. Most likely, the frequency of visits to the gym, and age and lifetime.Even without building a model, and only on the basis of EDA, we can assume what and how affects the target variable.

Conclusion: Judging by the matrix, there are no signs with a high probability of reporting the correct answer. But there are a couple of correlated signs:

Step 3. Build a churn prediction model

Build a model of binary classification of customers, where the target feature is the fact of customer churn in the next month: • Split the data into training and validation samples using the train_test_split() function. • Train the model on train sampling in two ways: • logistic regression, o Random forest. • Evaluate the accuracy, precision, and recall metrics for both models on a validation sample. Compare the models on them. Which model performed better based on metrics? Don't forget to specify the random_state parameter when splitting the sample and specifying the algorithm.

The real data that gets into the input of the model after its development is called test data.

The part of the data that is fed to the model during training is called a train data.

The data on which the model is checked is called validation data.

1. Logistic regression

Binary classification is a special case of classification, when there are only two classes: "0" or "1". The target variable here is a binary quantity.
A popular algorithm for solving such problems is logistic regression. It is implemented as the LogisticRegression() class in the same module linear_model the sklearn library as the linear regression algorithm.

Let's predict the churn of customers, where the target feature is the fact of customer churn in the next month. The target variable is 'Churn'

In [15]:
```
# ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# разделите данные на признаки (матрица X) и целевую переменную (y)
X = gym.drop('Churn', axis = 1) # набор признаков
```

```
y = gym['Churn'] # значение целевой переменной

# разделяем модель на обучающую(X_train,y_train) и валидационную выборку(X_test, y_tes
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# зададим алгоритм для нашей модели
model = LogisticRegression(solver='liblinear') # задайте модель

# обучим модель на обучающей выборке
y_pred=model.fit(X_train, y_train)

# воспользуемся уже обученной моделью, чтобы сделать прогнозы
predictions = model.predict(X_test)

# оценка вероятности принадлежности ко второму классу (среди классов "0" и "1", имеем
# бинарный прогноз
probabilities = model.predict_proba(X_test)[:,1]

# выведим значения predictions и probabilities на экран
### print(predictions)
### print(probabilities)
```

Learning to evaluate the quality of this model:

1. The percentage of correct answers is accuracy_score(acc = accuracy_score(y_true, y_pred)) This is the proportion of correctly guessed answers from all predictions. The closer the accuracy value is to 100%, the better.

2. Precision - says what proportion of predictions about "1" class is correct. (precision = precision_score (y_true, y_pred)) That is, we look at the proportion of correct answers only among the target class.

3. Recall shows how many real objects of class "1" you were able to detect using the model. (recall = recall_score (y_true, y_pred))

Each metric takes values from 0 to 1. The closer to one, the better.

In [16]:
```
# МЕТРИКИ
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score

print('Accuracy: {:.2f}'.format(accuracy_score(y_test, predictions)))
print('Precision: {:.2f}'.format(precision_score(y_test, predictions)))
print('Recall: {:.2f}'.format(recall_score(y_test, predictions)))
```

```
Accuracy: 0.91
Precision: 0.81
Recall: 0.81
```

Conclusion on metrics: We predict correctly by more than 90%. If you look at precision and recall, there seems to be a lot of work to be done. Therefore, to assess the quality of the classifier, we use the roc_auc metric, or the area under the error curve - AUC-ROC (roc_auc = roc_auc_score(y_true, probabilities[:,1])) At the input of this function, the vector of real answers y_true and the probability vector of assigning an object to class "1" are supplied. To do this, only the second numbers (probabilities[:,0]) are taken from the vector of probability pairs for assigning an object to the class "1" and "1". At the output of this function, we get one number in the range from 0 (usually from 0.5) to 1.

In [17]:
```
from sklearn.metrics import roc_auc_score

# выведем roc_auc_score
print('ROC_AUC: {:.2f}'.format(roc_auc_score(y_test, probabilities)))
```

```
ROC_AUC: 0.96
```

Conclusion on the ROC_AUC: This metric is quite decent - 0.97! Our model perfectly predicted customer churn.

In fact, many algorithms at the output of the predict_proba() method do not give out exactly the probability of being assigned to a class, but rather the degree of confidence of the model in belonging to it. If the model is more than 0.5 sure that the object belongs to class "1", then the answer is "1", otherwise it is "0"

Automatically, when the confidence was cut off by the threshold equal to 0.5, predict_proba() gave the result, among which the objects of the target class "1" belonged to "0". If we shift the threshold down (for example, to 0.4) and manually define the class based on such a "custom" (English custom, "custom, non-standard") threshold, we get more acceptable results. Let's do it.

In [18]:
```
# задаём порог
threshold = 0.4

# на основании вероятностей и соотношения классов рассчитайте predict
custom_predictions = [0 if i < threshold else 1 for i in probabilities]

# выведем все изученные метрики для прогноза по новому порогу
print('Метрики для прогноза с кастомным порогом')
print('Accuracy for custom: {:.2f}'.format(accuracy_score(y_test, custom_predictions))

print('Precision for custom: {:.2f}'.format(precision_score(y_test, custom_predictions

print('Recall for custom: {:.2f}'.format(recall_score(y_test, custom_predictions)))
```

```
Метрики для прогноза с кастомным порогом
Accuracy for custom: 0.90
Precision for custom: 0.77
Recall for custom: 0.84
```

Conclusion after decreasing the threshold: The Recall priority metric has increased slightly. The rest of the metrics were affected by a slight decrease. It is important to understand the threshold here.

2. RANDOM FOREST

Implementation of tree ensembles

The random forest in sklearn is implemented in the ensemble module. From this module, the most commonly used algorithms are RandomForestClassifier() and RandomForestRegressor()

for classification and regression problems, respectively.

from sklearn.ensemble import RandomForestRegressor

ALGORITHM:

rf_model = RandomForestRegressor(n_estimators = 100) rf_model.fit(X_train, y_train) y_pred = rf_model.predict(X_val)

In [19]:
```
# зададим алгоритм для новой модели на основе алгоритма решающего дерева

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier # для случайного леса

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import roc_auc_score


# определим функцию, которая будет выводить наши метрики
def print_all_metrics(y_true, y_pred, y_proba, title='Метрики классификации'):
    print(title)
    print('\tAccuracy: {:.2f}'.format(accuracy_score(y_true, y_pred)))
    print('\tPrecision: {:.2f}'.format(precision_score(y_true, y_pred)))
    print('\tRecall: {:.2f}'.format(recall_score(y_true, y_pred)))

    print('\tROC_AUC: {:.2f}'.format(roc_auc_score(y_true, y_proba)))

# разделите данные на признаки (матрица X) и целевую переменную (y)
X = gym.drop('Churn', axis = 1) # набор признаков
y = gym['Churn'] # значение целевой переменной

# разделяем модель на обучающую и валидационную выборку
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# обучите StandartScaler на обучающей выборке
scaler = StandardScaler()
scaler.fit(X_train)

# Преобразуйте обучающий и валидационные наборы данных
X_train_st = scaler.transform(X_train)
X_test_st = scaler.transform(X_test)

# зададим алгоритм для новой модели на основе алгоритма случайного леса
rf_model =  RandomForestClassifier(n_estimators = 100, random_state = 0)

# обучим модель случайного леса
rf_model.fit(X_train_st, y_train)

# воспользуемся уже обученной моделью, чтобы сделать прогнозы
rf_predictions = rf_model.predict(X_test_st)
rf_probabilities = rf_model.predict_proba(X_test_st)[:, 1]

# выведем все метрики
print_all_metrics(
    y_test,
    rf_predictions,
    rf_probabilities,
    title='Метрики для модели случайного леса:'
)
```

```
Метрики для модели случайного леса:
        Accuracy: 0.91
        Precision: 0.82
        Recall: 0.81
        ROC_AUC: 0.95
```

Conclusion: This is the power of collective models. The metric ROC_AUC: 0.97 is quite high - it contains the maximum amount of information regarding the quality of the model for random forest and gradient boosting algorithms in comparison with simpler models.

Step 4. Make tenant clustering

1. Set aside the churn column and cluster the objects (clients): • Standardize data. • Build a distance matrix using the linkage() function on a standardized feature matrix and draw a dendrogram. Note: Dendrogram may take time to draw! Based on the resulting graph, guess how many clusters can be allocated.

In [ ]:

```
## 1. Нарисуем дендограмму в Python. Сперва из модуля для иерархической кластеризации
## импортируем классы модели кластеризации linkage() и dendrogram():
from sklearn.preprocessing import StandardScaler

import seaborn as sns
import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage # Нарисуем дендограмму в Pytho

## 2. обязательная стандартизация данных перед работой с алгоритмами
sc = StandardScaler()# создаём объект класса scaler (нормализатор)
X_sc = sc.fit_transform(gym)

linked = linkage(X_sc, method = 'ward')

## 3. В переменной linked сохранена таблица «связок» между объектами.
## Её можно визуализировать как дендрограмму:

plt.figure(figsize=(10, 5))
dendrogram(linked, orientation='top')
plt.title('Кластеризация клиентов')
plt.show()
```

CONCLUSION: The proposed optimal number of clusters is 5 — five different colors on the graph. The complexity of agglomerative clustering lies not in the structure of the algorithm itself, but in the calculations that the machine performs to build a dendrogram. Calculations of pairwise distances can take a very long time. Therefore, when solving the clustering problem, it is useful to build a dendrogram on a random subsample, and after estimating the optimal number of clusters, run the faster K-Means algorithm.

1. Train a clustering model based on the K-Means algorithm and predict customer clusters. Let's agree to take n = 5 as the number of clusters so that your results can be compared with the results of the rest of the students. However, of course, in life, no one will tell you the right answer, And the decision is yours (based on the study of the graph from the previous paragraph).
2. Look at the feature averages for clusters. Is it possible to notice something right away? • Build feature distributions for clusters. Is it possible to notice something from them? • For each resulting cluster, calculate the churn rate (using the groupby() method). Do they differ in the share of outflow? Which clusters are prone to churn and which are reliable?

An algorithm called KMeans is implemented in the sklearn.cluster module. Here's its syntax:

from sklearn.cluster import KMeans

Mandatory standardization of data before working with algorithms sc = StandardScaler() X_sc = sc.fit_transform(X)

km = KMeans(n_clusters=5, random_state=0) # set the number of clusters to 5 and fix the random_state value for reproducibility of the result labels = km.fit_predict(X_sc) # apply the algorithm to the data and form a cluster vector

In [ ]:
```
from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt
import itertools

# определим функцию отрисовки графиков попарных признаков для кластеров
def show_clusters_on_plot(df, x_name, y_name, cluster_name):
    plt.figure(figsize=(5, 5))
    sns.boxplot(
        df[x_name], df[y_name], hue=df[cluster_name], palette='Paired'
    )
    plt.title('{} vs {}'.format(x_name, y_name))
    plt.show()

# стандартизируем данные
sc = StandardScaler()
x_sc = sc.fit_transform(gym)

# задаём модель k_means с числом кластеров 5 и фиксируем значение random_state
km = KMeans(n_clusters=5, random_state=0)
# В переменной labels сохранятся индексы предложенных алгоритмом групп:
labels = km.fit_predict(x_sc)

# сохраняем метки кластера в поле нашего датасета
gym['cluster_km'] = labels

# получить средние значения признаков по полученным кластерам методом groupby()
clusters = gym.groupby(['cluster_km']).mean()

display(clusters)
```

```
# отрисуем графики для попарных сочетаний признаков с разметкой по кластерам
col_pairs = list(itertools.combinations(gym.drop('cluster_km', axis=1).columns, 2))
for pair in col_pairs:
    show_clusters_on_plot(gym, pair[0], pair[1], 'cluster_km')
    print(pair)
```

0 cluster only for gender 0 4 cluster only closely spaced 0,1,2 cluster of close living brought friends

In [ ]:
```
# БИНАРНЫЕ ПРИЗНАКИ графики

sns.countplot(x='Near_Location', hue='cluster_km', data=gym)
```

In [ ]:
```
sns.countplot(x='Churn', hue='cluster_km', data=gym)
```

CONCLUSION on binary signs:

3rd cluster - only far living. 4th cluster - only retired

INFERENCE:

The churn rate depends on how close the fitness center is to home and work. The farther from home, the greater the churn rate, more than 50% After the first two visits, the outflow decreases. That is, it is necessary to do everything so that the client comes a second time. The greater the total revenue from other services of the fitness center, the lower the outflow. It is necessary to interest the client with additional sports cocktails, cosmetics, massage.

Step 5. Draw conclusions and make basic recommendations for working with clients

Formulate key takeaways and offer recommendations for your customer engagement and retention strategy. Do not describe the strategy in detail: 3-4 important principles and examples of their implementation in the form of certain marketing actions are enough.

1. Pay attention to customers under 36 years old. Make, for example, a tariff for the young. To interest with a better price. Give advertising, put up ads near educational institutions, or hang a banner near schools, universities. Focus on the fact that beauty = health, for example.
2. Make a favorable rate for all-first 10 months of visits. For example-tariff-"NEWBIE"
3. Increase the number of visits to the gym in the contract. Make 1 additional visit per week free. Favorable rates apply to the age category up to 36 years.