Введение в анализ данных

Домашнее задание 2. Pandas и Seaborn

Правила:

- Дедлайн **30 апреля 23:59**. После дедлайна работы не принимаются кроме случаев наличия уважительной причины.
- Выполненную работу нужно отправить на почту `mipt.stats@yandex.ru (mailto:%60mipt.stats@yandex.ru) , указав тему письма "[номер группы] Фамилия Имя - Задание 2"`. Квадратные скобки обязательны.
- Прислать нужно ноутбук и его pdf-версию (без архивов). Названия файлов должны быть такими: `2.N.ipynb` и `2.N.pdf` , где `N` -- ваш номер из таблицы с оценками. *pdf-версию можно сделать с помощью Ctrl+P. Пожалуйста, посмотрите ее полностью перед отправкой. Если что-то существенное не напечатается в pdf, то баллы могут быть снижены.*
- Решения, размещенные на каких-либо интернет-ресурсах, не принимаются. Кроме того, публикация решения в открытом доступе может быть приравнена к предоставлении возможности списать.
- Для выполнения задания используйте этот ноутбук в качестве основы, ничего не удаляя из него.
- Если код будет не понятен проверяющему, оценка может быть снижена.
- Никакой код при проверке запускаться не будет.

Баллы за задание:

Легкая часть (достаточно на "хор"):

- Задача 1 -- 4 балла
- Задача 2 -- 2 балла

Сложная часть (необходимо на "отл"):

- Задача 3 -- 5 баллов
- Задача 4 -- 3 балла
- Задача 5 -- 8 баллов

Баллы за разные части суммируются отдельно, нормируются впоследствии также отдельно. Иначе говоря, 1 балл за легкую часть может быть не равен 1 баллу за сложную часть.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import plotly.graph_objects as go


sns.set(style='whitegrid', font_scale=1.3)
%matplotlib inline
```

Легкая часть

Задача 1

Представьте, что вы министр образования страны котиков. Вам нужно решить какие школы лучше: маленькие или большие.

Вы решили, что нужно сравнить их по результатам единого кошачьего экзамена (ЕКЭ). Предлагается посмотреть на средний результат по школам: отсортировать по нему и сделать выводы исходя из топ 10 лучших школ.

Вам дан датасет `cat_exam_data.csv`

Описание данных:

- `school` -- номер школы;
- `test_score` -- результат одного ученика из этой школы;
- `number_of_students` -- кол-во учеников в школе.

Приведены данные по всем ученикам из 500 школ страны котиков.

Загрузите датасет с результатами экзамена и посмотрите на первые пять строк.

Это можно сделать с помощью методов read_csv (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html) и head (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.head.html).

In [2]:
```
df = pd.read_csv('Данные к ДЗ 2/cat_exam_data.csv')
df.head()
```

Out[2]:

	school	test_score	number_of_students
0	26	39.0	965
1	54	64.0	1483
2	356	64.0	1055
3	108	68.0	1806
4	298	78.0	971

Проверьте, что в данных нет пропусков (NaN). Если они есть:

- проверьте в каком столбце;
- удалите их.

Могут помочь методы isna (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.isna.html) и dropna (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html).

In [3]:
```
df.isna().any()
```

Out[3]:
```
school                False
test_score             True
number_of_students    False
dtype: bool
```

In [4]:
```
df.dropna(how='any').head(10)
```

Out[4]:

	school	test_score	number_of_students
0	26	39.0	965
1	54	64.0	1483
2	356	64.0	1055
3	108	68.0	1806
4	298	78.0	971
5	386	29.0	1027
6	317	51.0	1135
7	155	43.0	969
8	359	33.0	895
9	416	45.0	522

Посчитайте описательные статистики (среднее, мин, макс, и тд.) по колонкам в таблице с помощью одной функции:

In [5]:
```
df.describe()
```

Out[5]:

	school	test_score	number_of_students
count	501562.000000	501358.000000	501562.000000
mean	250.095661	51.681788	1088.868830
std	144.428841	14.179981	281.022934
min	0.000000	-0.000000	156.000000
25%	125.000000	42.000000	890.000000
50%	250.000000	52.000000	1079.000000
75%	377.000000	61.000000	1285.000000
max	499.000000	100.000000	1806.000000

Посчитайте сколько котиков получили 100 баллов:

In [6]:
```
row_mask = df.loc[:,'test_score'] == 100
(df.loc[row_mask, 'test_score']).count()
```

Out[6]: 51

Выведите информацию о школах, где есть хотя бы один котик, получивший 100 баллов на ЕКЭ.

Отсортируйте эти школы по количеству стобалльников.

Могут помочь методы groupby (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html), sort_values (https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/pandas.DataFrame.sort_values.html), transform (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.transform.html) и count (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.count.html).

In [7]:
```
df_100 = df.loc[row_mask, :]
df_100 = df_100.groupby(by=['school', 'number_of_students'],
                        as_index=False).count()
df_100 = df_100.sort_values(by=['test_score'],
                            ascending=False)
df_100.rename(columns={'test_score':'count'}).head(10)
```

Out[7]:

	school	number_of_students	count
48	486	800	2
0	4	1015	1
37	395	436	1
27	302	1517	1
28	303	701	1
29	315	580	1
30	328	789	1
31	342	920	1
32	346	913	1
33	351	1224	1

Посчитайте средний результат каждой из школ и посмотрите на результат:

Может помочь метод [mean (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html).

In [8]:
```
df_mean = df.groupby(by=['school'],
                     as_index=False).mean()
df_mean.head(10)
```

Out[8]:

	school	test_score	number_of_students
0	0	52.294258	418
1	1	52.028950	1209
2	2	51.751451	1035
3	3	51.566265	1245
4	4	51.025641	1015
5	5	51.394737	988
6	6	52.063527	914
7	7	51.670356	1265
8	8	51.543436	1036
9	9	52.260471	766

Отсортируйте школы по среднему результату:

In [9]:
```
df_mean = df_mean.sort_values(by=['test_score'],
                              ascending=False)
```

Посмотрите на топ 10 лучших результатов:

In [10]:
```
df_mean.head(10)
```

Out[10]:

	school	test_score	number_of_students
57	57	53.341682	560
124	124	52.993311	598
263	263	52.982063	669
82	82	52.981982	555
366	366	52.970207	773
465	465	52.957555	1179
169	169	52.932401	858
449	449	52.910364	357
477	477	52.898515	808
464	464	52.865429	431

In [11]:
```
df.loc[:, "number_of_students"].mean()
```

Out[11]:
```
1088.8688297757806
```

Вывод: в топе оказались школы, в которых учатся относительно небольшое количество котиков.

Ожидаем, что внизу рейтинга будут только большие школы, давайте это проверим.

Посмотрите теперь на 10 худших школ:

In [12]:
```
df_mean.tail(10)
```

Out[12]:

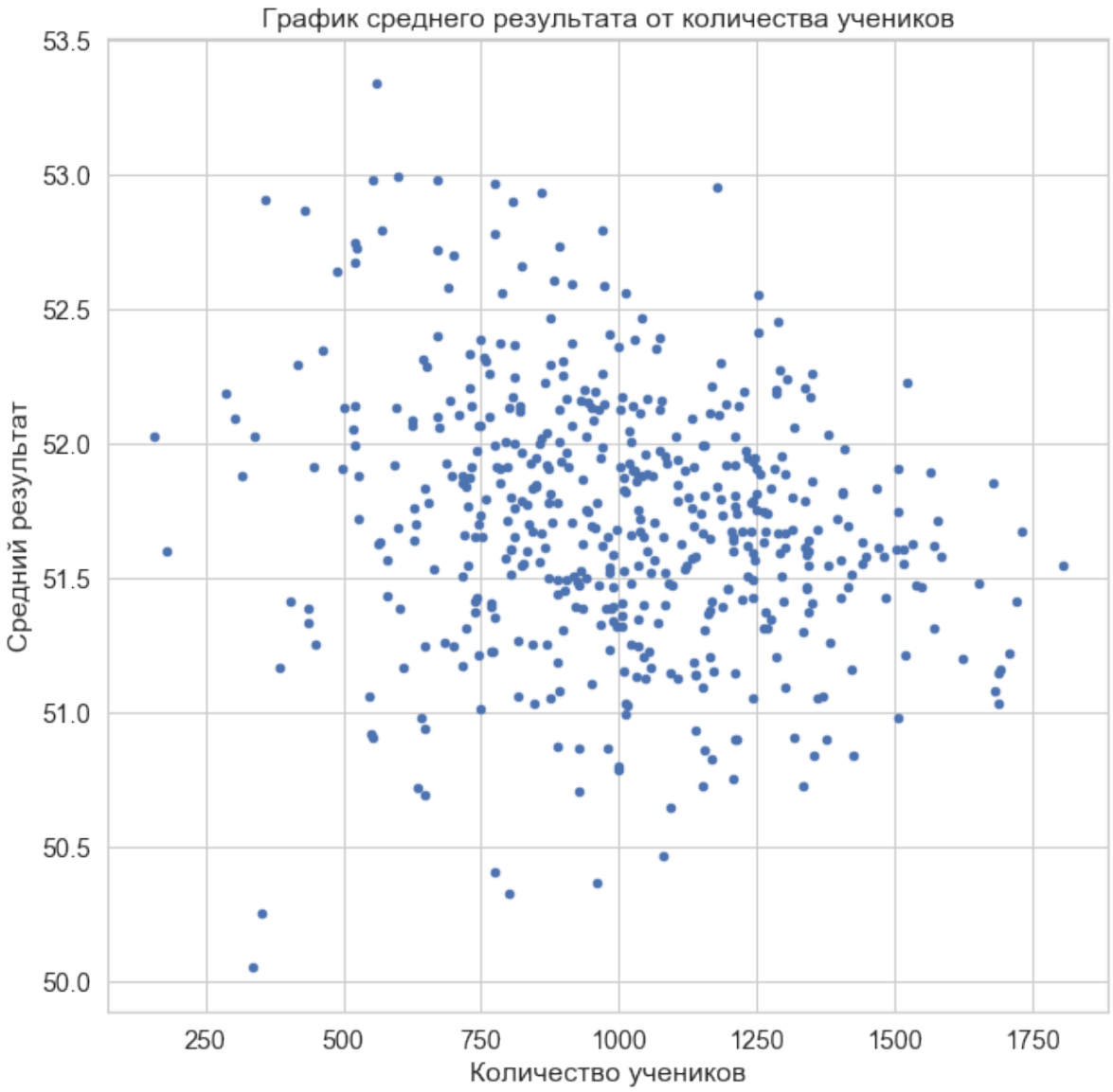
	school	test_score	number_of_students
383	383	50.716981	636
264	264	50.702906	929
48	48	50.690293	649
156	156	50.641354	1093
394	394	50.461538	1079
194	194	50.406977	774
471	471	50.362500	960
486	486	50.325000	800
211	211	50.248571	351
353	353	50.050595	336

Вывод: оказывается, школы с большим количеством котиков располагаются где-то посередине таблицы df_mean, потому что внизу рейтинга опять школы с небольшим количеством котиков.

Постройте график зависимости среднего результата ЕКЭ от количества учеников:

In [23]:
```
plt.figure(figsize=(10, 10))
plt.scatter(df_mean.loc[:, 'number_of_students'],
            df_mean.loc[:, 'test_score'], s=20)
plt.xlabel('Количество учеников')
plt.ylabel('Средний результат')
plt.title('График среднего результата от количества учеников')
plt.legend
plt.show
```

Out[23]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



Вывод: чем больше учеников в школе, тем меньше отклонение от среднего, что логично т.к. выборка стала больше, увеличивая детерминированность.

Но как же тогда решить какие школы лучше?

Сгруппируйте учеников в зависимости от типа школы (<= 1000 учеников == маленькая школа):

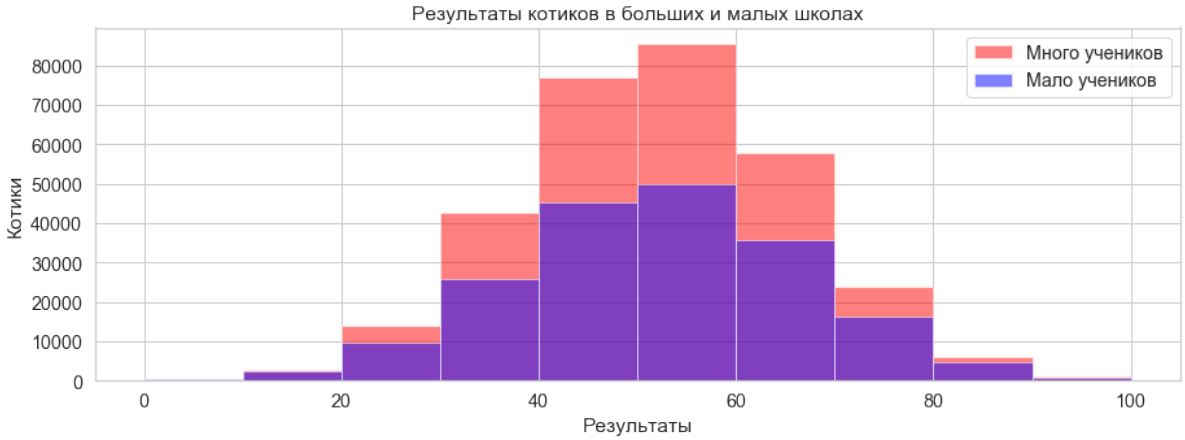
In [24]:
```
big_mask = df.loc[:,'number_of_students'] > 1000
small_mask = df.loc[:,'number_of_students'] <= 1000
df_big = df.loc[big_mask, 'test_score']
df_small = df.loc[small_mask, 'test_score']
```

Постройте гистограммы этих двух выборок на одном графике, установив параметры

- `bins=10` --- число бинов гистограммы;
- `alpha=0.5` --- прозрачность бинов.

In [47]:
```
plt.figure(figsize=(15, 5))
plt.hist(df_big, bins=10, alpha=0.5, density=False,
         label="Много учеников", color='red')
plt.hist(df_small, bins=10, alpha=0.5, density=False,
         label="Мало учеников", color='blue')
plt.xlabel('Результаты')
plt.ylabel('Котики')
plt.title('Результаты котиков в больших и малых школах')
plt.legend()
plt.show
```

Out[47]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



Вывод: как и предполагалось, большие школы умеют средние результаты ЕКЭ.

Задача 2

Задача заключается в работе с данными о трендах на YouTube. В этом вам поможет библиотека `seaborn`, которая была рассмотрена на одной из последних лекций.

1. Подготовка данных

Скачайте файл `RUvideos_short.csv` с данными о видео в российском сегменте Youtube с 14 ноября по 21 ноября 2017 года. Полная версия данных доступна на kaggle (https://www.kaggle.com/datasnaek/youtube-new#RUvideos.csv).

Прочитайте данные с помощью библиотеки `pandas` и напечатайте начало таблицы. В колонке `trending_date` записана дата. При чтении таблицы распознайте ее.

In [26]:
```
df_yt = pd.read_csv('Данные к ДЗ 2/RUvideos_short.csv')
df_yt['trending_date'] = pd.to_datetime(df_yt.trending_date,
                                        format='%y.%d.%m')
df.head()
```

Out[26]:

	school	test_score	number_of_students
0	26	39.0	965
1	54	64.0	1483
2	356	64.0	1055
3	108	68.0	1806
4	298	78.0	971

В таблице много лишних данных. Оставьте следующие столбцы:

- `trending_date` -- дата в формате год-день-месяц;
- `category_id` -- категория видео, названия приведены в файле `RU_category_id.json`;
- `views` -- количество просмотров видео;
- `likes` -- количество лайков;
- `dislikes` -- количество дислайков;
- `comment_count` -- количество комментариев.

Из даты оставьте только день. Для этого можно пройтись циклом по всем датам и взять поле `day` у даты. Напечатайте начало таблицы.

In [27]:
```
df_yt = df_yt.loc[:, ['trending_date',
                      'category_id',
                      'views',
                      'likes',
                      'dislikes',
                      'comment_count']]

for index, date in enumerate(df_yt.loc[:, 'trending_date']):
    df_yt.loc[index, 'trending_date'] = date.day
df_yt.head()
```

Out[27]:

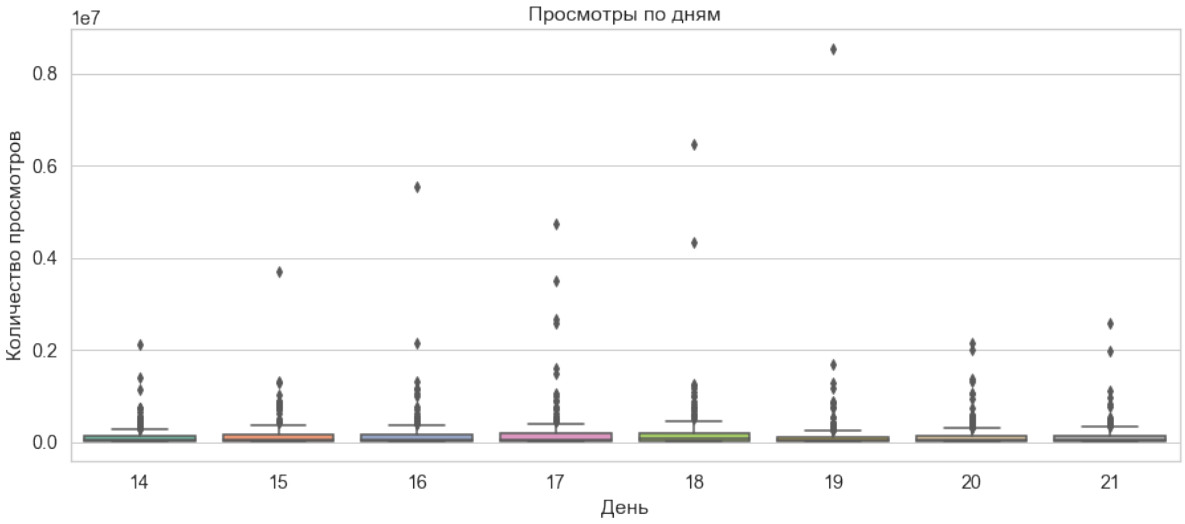
	trending_date	category_id	views	likes	dislikes	comment_count
0	14	22	62408	334	190	50
1	14	22	330043	43841	2244	2977
2	14	24	424596	49854	714	2944
3	14	22	112851	3566	122	80
4	14	24	243469	36216	631	1692

2. Некоторая визуализация

Постройте ящики с усами на каждый день по количеству просмотров. Насколько хороший получился график?

In [29]:
```
plt.figure(figsize=(15, 6))

plt.subplot(111)
sns.boxplot(x='trending_date', y='views',
            data=df_yt, palette='Set2')
plt.xlabel('День')
plt.ylabel('Количество просмотров')
plt.title('Просмотры по дням')
```

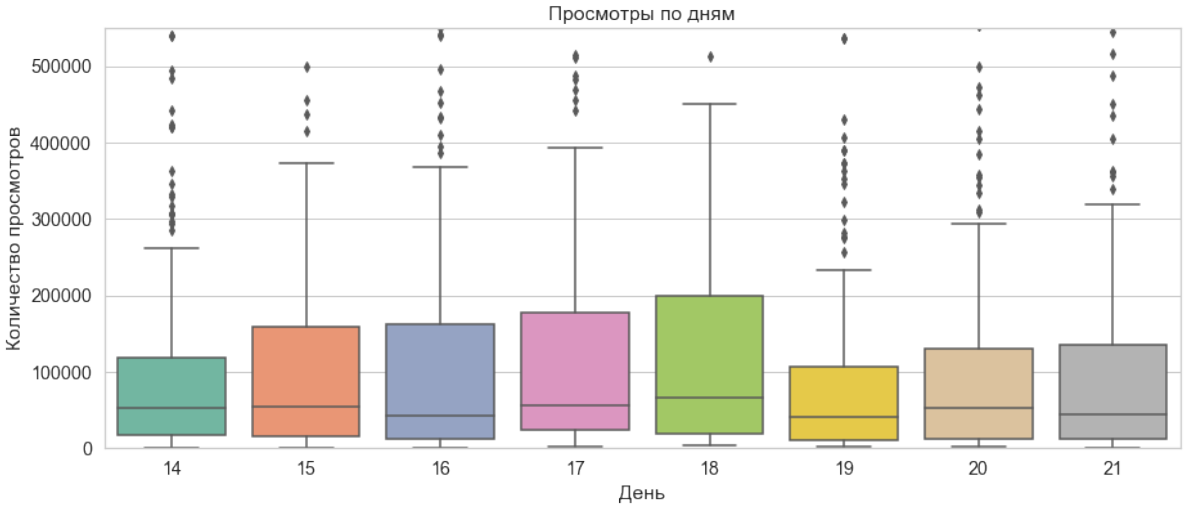
Out[29]:
```
Text(0.5, 1.0, 'Просмотры по дням')
```



Исправьте этот недостаток, установив некоторое значение.

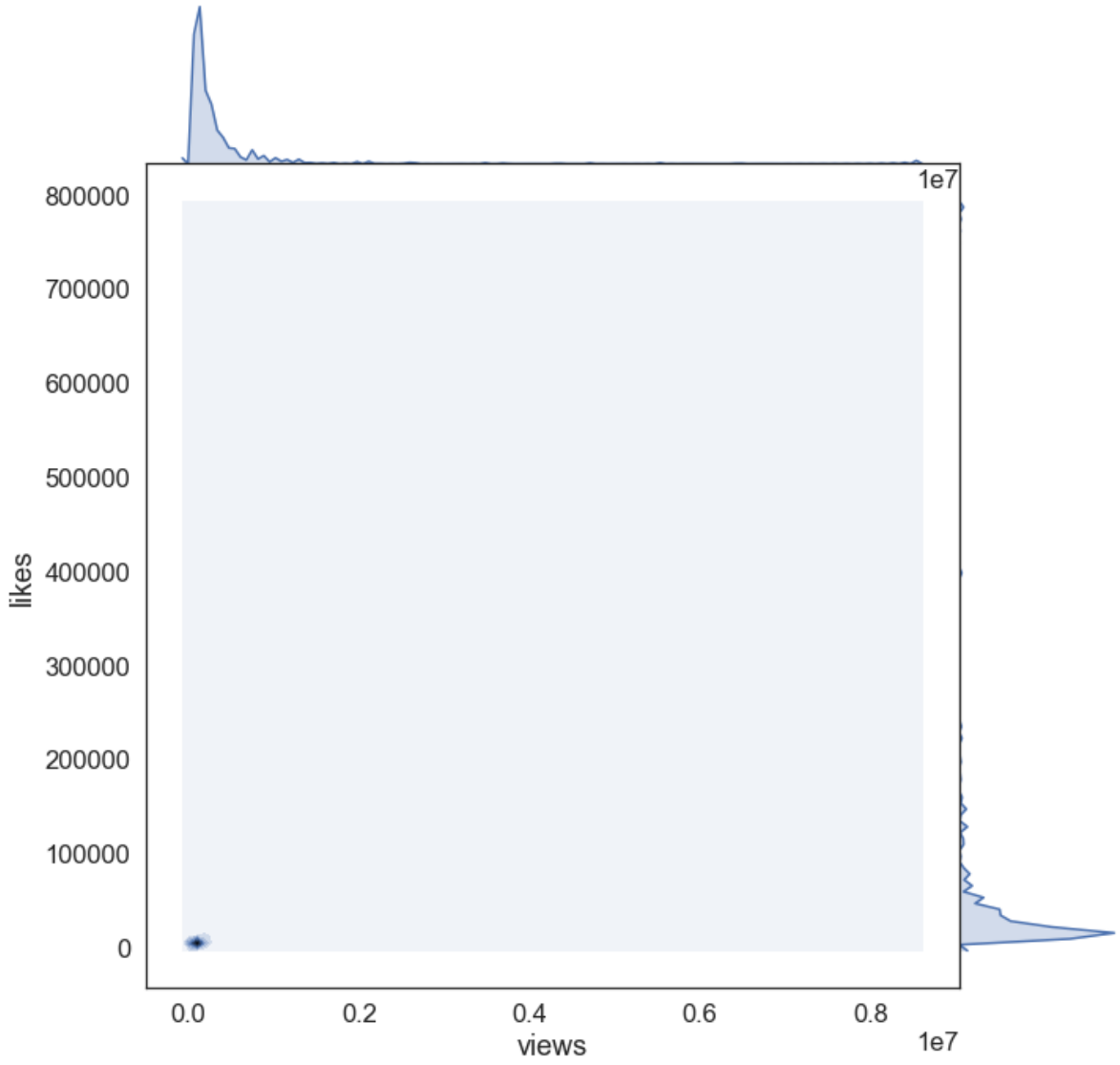
In [30]:
```
plt.figure(figsize=(15, 6))

plt.subplot(111)
sns.boxplot(x='trending_date',
            y='views',
            data=df_yt,
            palette='Set2')
plt.ylim(0, 550000)
plt.xlabel('День')
plt.ylabel('Количество просмотров')
plt.title('Просмотры по дням')
```

Out[30]:
```
Text(0.5, 1.0, 'Просмотры по дням')
```



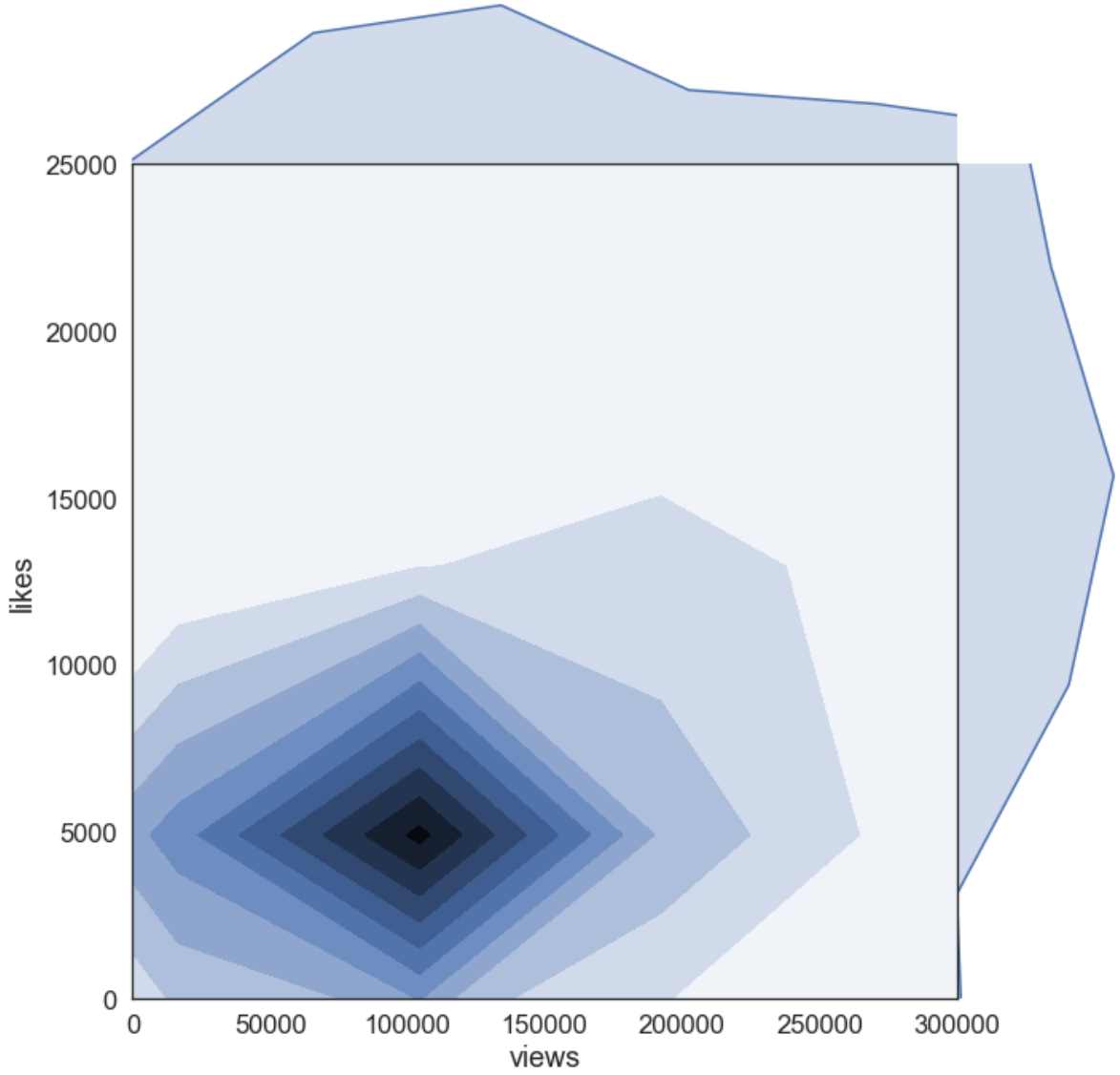
Постройте jointplot по всем данным для количества просмотров по горизонтальной оси и количества лайков по вертикальной. Насколько информативен такой график?

In [32]:
```
with sns.plotting_context("notebook", font_scale=1.5),sns.axes_styl
    sns.jointplot(df_yt.views, df_yt.likes, kind='kde', height=10,
```



Исправьте этот недостаток.

In [33]:
```
with sns.plotting_context("notebook",font_scale=1.5),sns.axes_style
    plot = sns.jointplot(df_yt.views, df_yt.likes, kind='kde', heig
    plot.ax_joint.set_xlim(0, 300000)
    plot.ax_joint.set_ylim(0, 25000);
```



Сложная часть

Задача 3

Netflix за последние 5-10 лет обзавелись большим количеством зрителей. С увеличением числа зрителей увеличилось и разнообразие шоу. Соответственно, перед аналитиками из киноиндустрии встала задача исследования данных с рейтингами различных сериалов.
В данном задании вам предстоит провести визуальный анализ датасета **1000 Netflix Shows** (по состоянию на 11.06.2017) и сделать выводы.

Описание признаков:

- `title` - название шоу.
- `rating` - рейтинг шоу. Например: G, PG, TV-14, TV-MA
- `ratingLevel` - описание рейтинговой группы и особенностей шоу.
- `release year` - год выпуска шоу.
- `user rating score` - оценка пользователей.

Загрузите данные, выполнив код ниже.

In [34]:
```
# код ниже менять нельзя (кроме пути до данных), просто запустите я
data = pd.read_csv('Данные к ДЗ 2/netflix_data.csv',
                   encoding='cp437')
del data['ratingDescription'], data['user rating size']
```

Удалите из данных дупликаты. Сколько объектов удалено?

In [35]:
```
print('Дупликатов:', data.duplicated().sum())
data = data.drop_duplicates(keep='first')
```

```
Дупликатов: 500
```

Сколько объектов осталось?

In [36]:
```
print('Осталось:', data.shape[0])
```

```
Осталось: 500
```

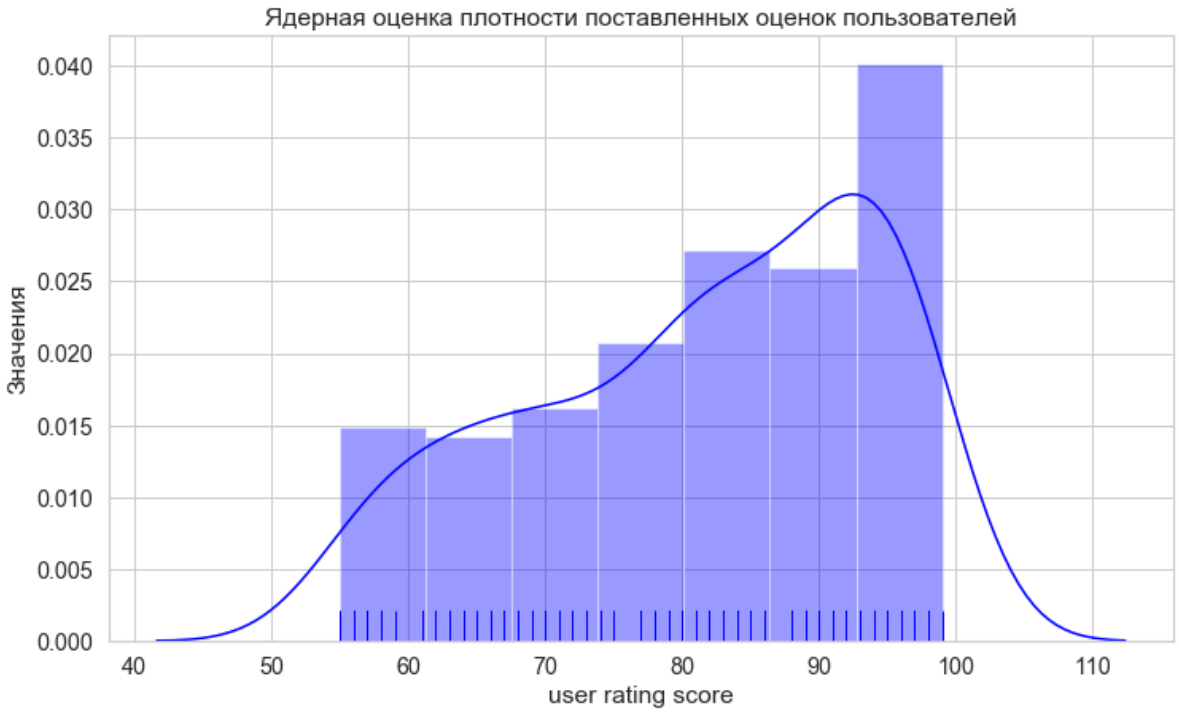
Сколько рейтинговых групп представлено в данных?

In [37]:
```
print('Различных групп:',data.rating.nunique())
```

```
Различных групп: 13
```

Какие оценки пользователи ставят чаще? Постройте гистограмму оценок.

In [40]:
```
plot_data = data.dropna(how='any')
with sns.plotting_context(font_scale=1.5), sns.axes_style('whitegri
    plt.figure(figsize=(12, 7))
    plt.title("Ядерная оценка плотности поставленных оценок пользов
    plt.ylabel('Значения')
    sns.distplot(plot_data.loc[:, 'user rating score'],
                 rug=True, color='blue');
```



Вывод: пользователи, в основном, ставят положительные оценки.

Выведите основную информацию об оценках пользователей: среднее, стандартное отклонение, минимум, максимум, медиана. Отличаются ли медиана и среднее? Могут ли данные характеристики значительно отличаться? Почему?

In [41]:
```
data.loc[:, 'user rating score'].describe()
```

Out[41]:
```
count    256.000000
mean      81.398438
std       12.730904
min       55.000000
25%       71.000000
50%       83.500000
75%       93.000000
max       99.000000
Name: user rating score, dtype: float64
```

Ответ: из предыдущего графика видно, что распределение несимметричное, поэтому среднее значение `mean` и медиана (она же второй квартиль `50%`) отличаются. Данные характеристики будут сильно отличаться для совсем несимметричных данных, однако для достаточно симметричных данных они будут близки.

В какие годы были запущены шоу, представленные в датасете?

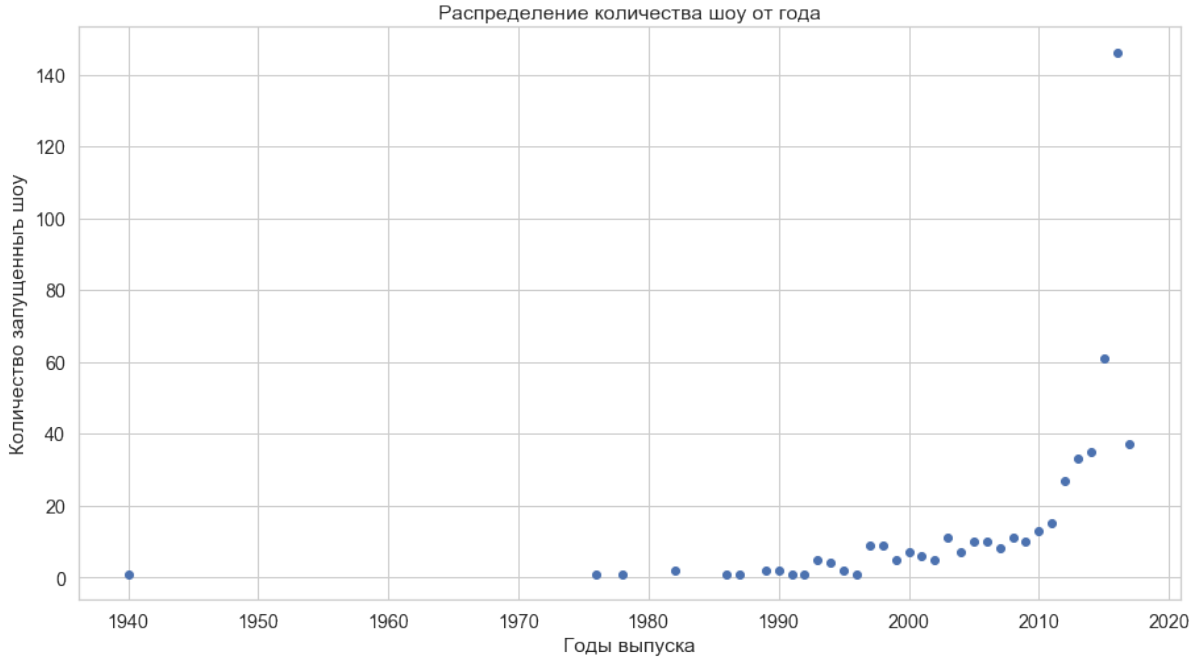
In [42]:
```
data.loc[:, ['title', 'release year']].head()
```

Out[42]:

	title	release year
0	White Chicks	2004
1	Lucky Number Slevin	2006
2	Grey's Anatomy	2016
3	Prison Break	2008
4	How I Met Your Mother	2014

Постройте график, показывающий распределение количества запущенных шоу в зависимости от года. Наблюдается ли рост? Есть ли выбросы?

In [43]:
```
plt.figure(figsize=(15, 8))
data_pure = data.loc[:, 'release year'].drop_duplicates(keep='first
x = data_pure
y = []
for year in data_pure:
    y.append(data.loc[data['release year'].isin([year])].shape[0])
plot = plt.scatter(x, y)
plt.xlabel('Годы выпуска')
plt.ylabel('Количество запущенныъ шоу')
plt.title('Распределение количества шоу от года')
plt.show()
```



In [105]:
```
with sns.plotting_context(font_scale=1.5), sns.axes_style('whitegri
    plt.figure(figsize=(12, 7))
    plt.title("Ядерная оценка плотности")
    sns.kdeplot(x, y, n_levels=15, shade=True, cmap="magma");
```

```
  File "<ipython-input-105-27c745cd7fa6>", line 1
    with sns.plotting_context(font_scale=1.5),
                                              ^
SyntaxError: invalid syntax
```

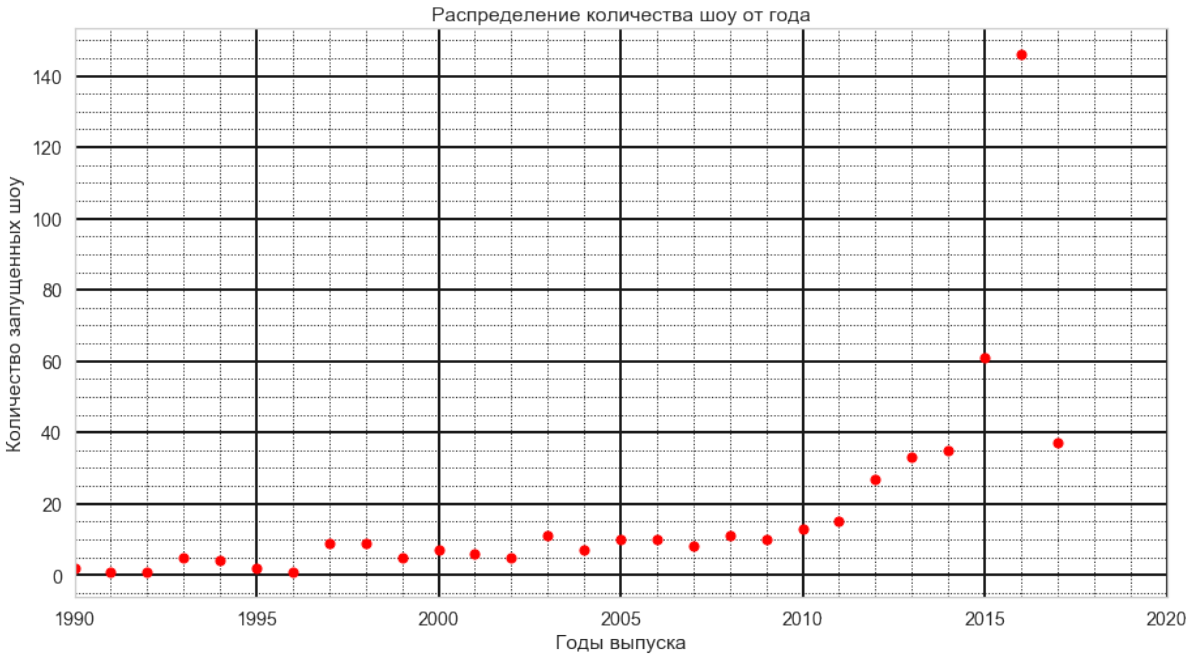
Вывод: подъём количества запущенных шоу наблюдается с 90-ых, постоянный рост числа запущенных шоу приходятся на нулевые.

Сравните среднюю оценку пользователей в 2016 со средней оценкой в 2017. Можно ли сделать вывод, что 2017 год успешнее для Netflix? ("Успешнее" значит, что пользователи в среднем ставили более высокие оценки) Ответить на этот вопрос вам поможет график, который вы построили выше.

In [98]:
```
print('Для 2016 года средний рейтинг:',
      data.loc[data['release year'].isin([2016])] \
      .loc[:, 'user rating score'].mean())
print('Для 2017 года средний рейтинг:',
      data.loc[data['release year'].isin([2017])] \
      .loc[:, 'user rating score'].mean())
```

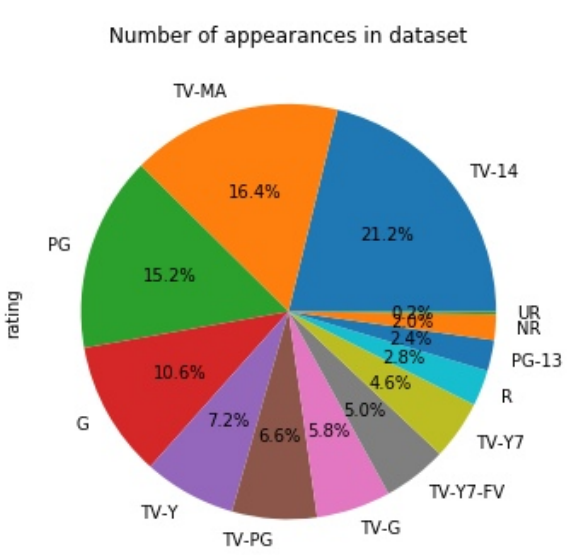
```
Для 2016 года средний рейтинг: 84.31395348837209
Для 2017 года средний рейтинг: 88.125
```

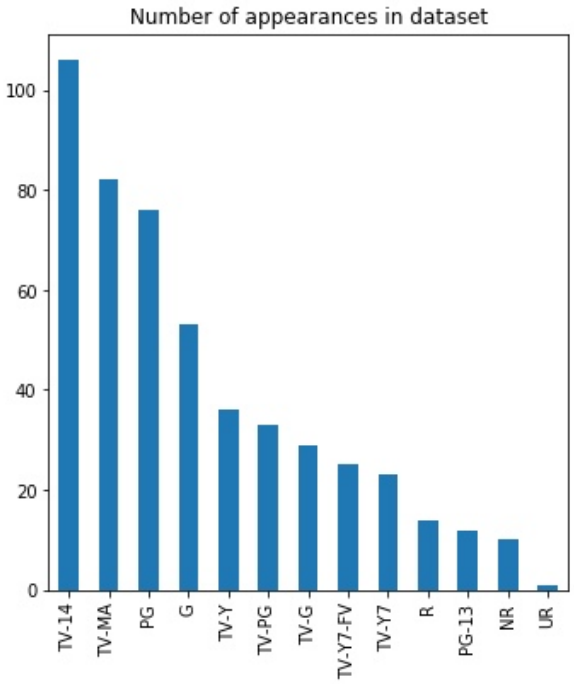
In [50]:
```
plt.figure(figsize=(15, 8))
data_pure = data.loc[:, 'release year'].drop_duplicates(keep='first
x = data_pure
y = []
for year in data_pure:
    y.append(data.loc[data['release year'].isin([year])].shape[0])
plt.scatter(x, y, s=50, c='red')
plt.xlabel('Годы выпуска')
plt.ylabel('Количество запущенных шоу')
plt.minorticks_on()
plt.grid(which='major',
        color = 'k',
        linewidth = 2)
plt.grid(which='minor',
        color = 'k',
        linestyle = ':')
plt.xlim(1990, 2020)
plt.title('Распределение количества шоу от года')
plt.show()
```



Вывод: в 2016 году у Netflix было очень много новых шоу, поэтому про успешность не стоит говорить в рамках этих двух годов, даже если в 2017 средняя оценка пользвателей выше, чем в 2016.

Ниже представлены два графика, показывающие распределение шоу по рейтинговым группам. Какой тип графика визуально более интерпретируемый? (Подсказка (https://sun9-40.userapi.com/c854228/v854228652/c754f/j6z5gMjJy2k.jpg)) Постройтре самостоятельно график, который считаете более интерпретируемым. Сделайте вывод.

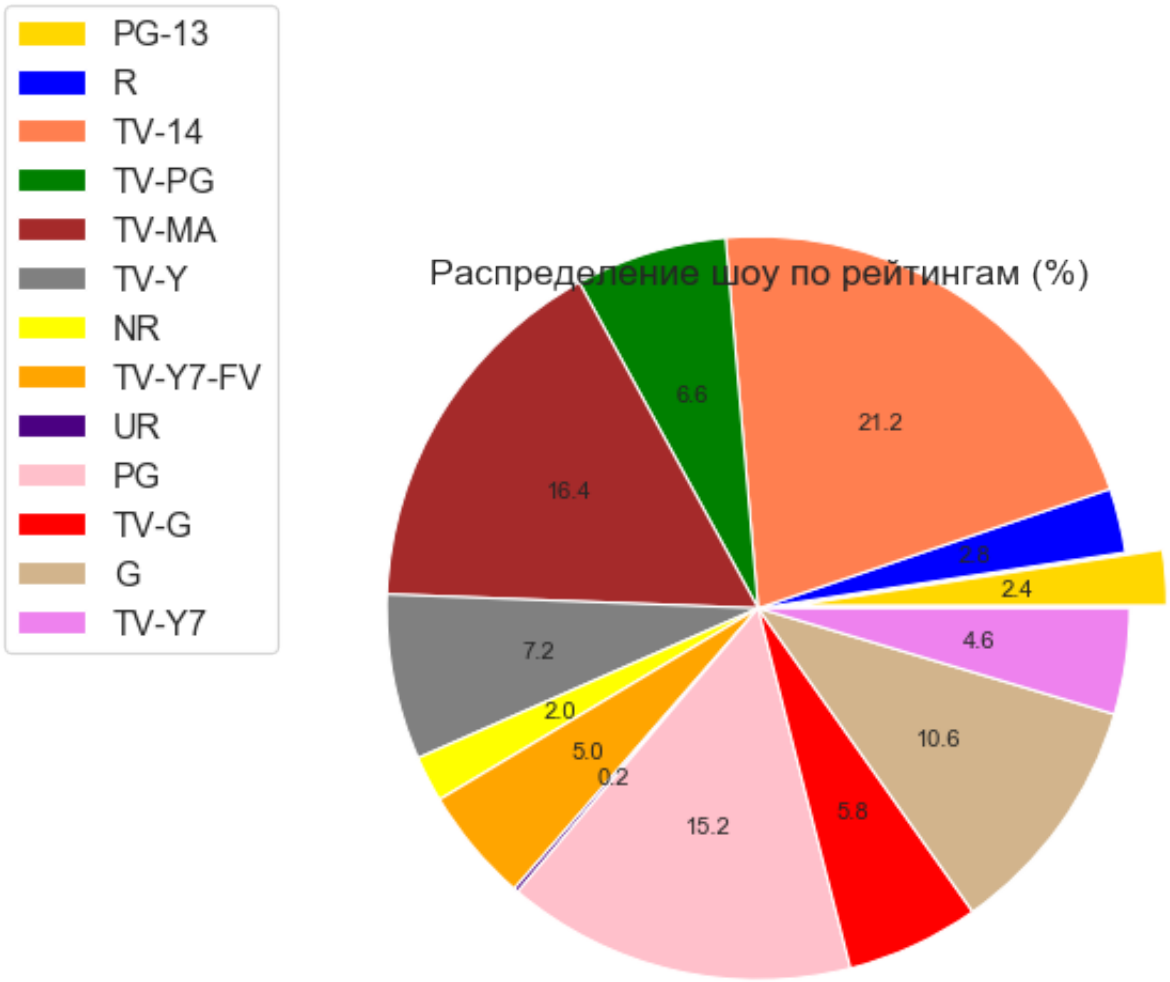




In [51]:
```
import random
import matplotlib.colors as mcolors
from matplotlib import cm

data_pure = data.loc[:, 'rating'] \
.drop_duplicates(keep='first')
x = data_pure
y = []
for year in data_pure:
    y.append(data.loc[data['rating'] \
                      .isin([year])].shape[0])

colors = ['gold', 'blue', 'coral', 'green', 'brown',
          'grey', 'yellow', 'orange', 'indigo', 'pink',
          'red', 'tan', 'violet']

dpi = 80
fig = plt.figure(dpi = dpi, figsize = (512 / dpi, 384 / dpi))

plt.title('Распределение шоу по рейтингам (%)')
```

```
plt.pie(
    y, autopct='%.1f', radius = 1.5,
    explode = [0.15] + [0 for _ in range(len(x) - 1)],
    colors=colors, shadow=False)
plt.legend(
    bbox_to_anchor = (-0.5, 0.4, 0.25, 0.1),
    loc = 'lower right', labels = x )
```

Out[51]: <matplotlib.legend.Legend at 0x1a24dbe550>



Вывод: наиболее понятнее выглядят диаграмнные графики, однако нужно соблюдать хорошую цветовую гамму и размеры.

Составьте топ-13 самых высоко оцененных шоу. Выберите из данного топа шоу, которое вам наиболее нравится (либо используйте `scipy.stats.randint`). Обозначим это шоу N. Ответьте на следующие вопросы:

- Какое шоу является худшим по оценкам в рейтинговой группе, к которой принадлежит N?
- Сколько шоу было выпущено в одном году с N?
- Насколько бы изменилась средняя оценка шоу, выпущенных в одном году с N, если бы Netflix не запустили шоу N?

In [52]:
```
top_data = data.dropna().sort_values(by=['user rating score'],
                                     ascending=False).head(13)
top_data.head(10)
```

Out[52]:

	title	rating	ratingLevel	release year	user rating score
41	13 Reasons Why	TV-MA	For mature audiences. May not be suitable for...	2017	99.0
63	Criminal Minds	TV-14	Parents strongly cautioned. May be unsuitable ...	2016	98.0
3	Prison Break	TV-14	Parents strongly cautioned. May be unsuitable ...	2008	98.0
64	Friends	TV-14	Parents strongly cautioned. May be unsuitable ...	2003	98.0
2	Grey's Anatomy	TV-14	Parents strongly cautioned. May be unsuitable ...	2016	98.0
8	The Walking Dead	TV-MA	For mature audiences. May not be suitable for...	2015	98.0
10	Once Upon a Time	TV-PG	Parental guidance suggested. May not be suitab...	2016	98.0
27	The Flash	TV-PG	Parental guidance suggested. May not be suitab...	2016	98.0
350	Lost	TV-14	Parents strongly cautioned. May be unsuitable ...	2010	98.0
88	Finding Dory	PG	mild thematic elements	2016	98.0

In [53]:
```
N = top_data.head(1)
N
```

Out[53]:

	title	rating	ratingLevel	release year	user rating score
41	13 Reasons Why	TV-MA	For mature audiences. May not be suitable for...	2017	99.0

Выбор пал на лучшее шоу.

In [99]:
```
bad_one = pd.merge(N, data, on='rating', how='inner') \
    .set_index('title_y')[['user rating score_y', 'release year_y']]
    .sort_values(by=['user rating score_y']).head(1)
print('Самое плохое шоу в данной категории:\n')
bad_one
```

```
Самое плохое шоу в данной категории:
```

Out[99]:

title_y	user rating score_y	release year_y
Bitten	56.0	2016

In [55]:
```
year_N = pd.merge(N, data, on='release year', how='inner') \
.set_index('title_y')[['user rating score_y']]
print(year_N.shape[0], 'было выпущенно в том же году, что и N.')
```

```
37 было выпущенно в том же году, что и N.
```

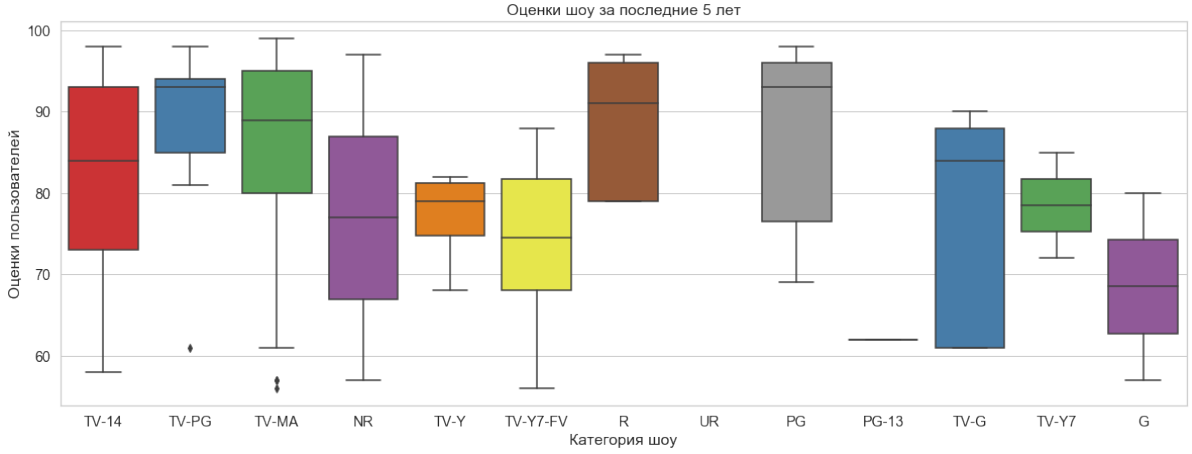
In [56]:
```
with_N = year_N.mean()
without_N=year_N.loc[year_N['user rating score_y']!=float(N.loc[:,
    .mean()
print('Разница средних оценок с и без N, выпущенных в один год с N:
      float(with_N - without_N))
```

```
Разница средних оценок с и без N, выпущенных в один год с N: 0.724
9999999999943
```

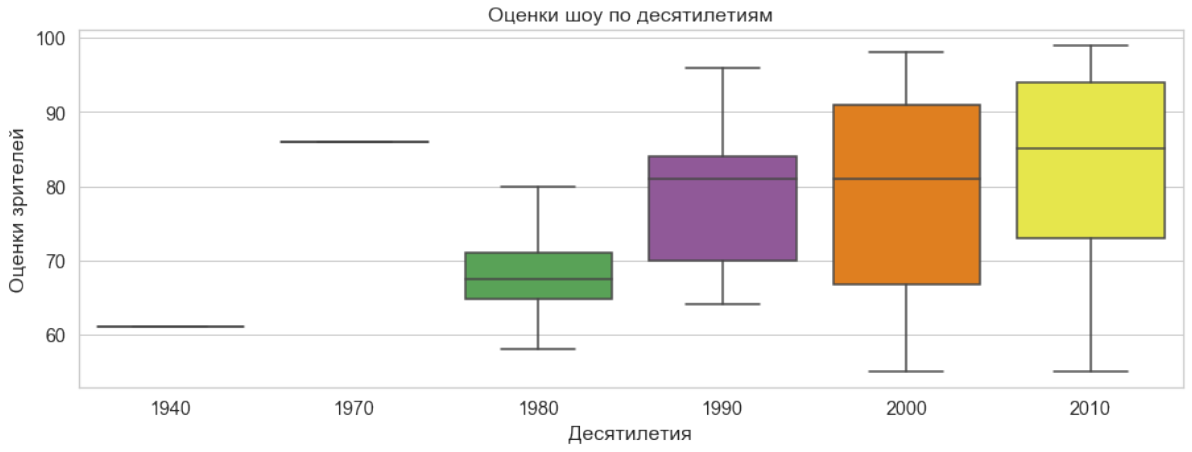
Ответьте на следующие вопросы при помощи `boxplot` :

- Какую рейтинговую группу зрители оценивали выше всего в последние пять лет?
- Как менялись оценки пользователей с течением времени? Постройте `boxplot` для каждого десятилетия.

In [57]:
```
plt.figure(figsize=(20, 7))
sns.boxplot(
x='rating',
y='user rating score',
data=data[data['release year'] > data['release year'].max() - 5],
palette='Set1'
).set(
xlabel='Категория шоу',
ylabel='Оценки пользователей'
)
plt.title('Оценки шоу за последние 5 лет')
plt.show();
```



In [58]:
```
plt.figure(figsize=(15, 5))
sns.boxplot(
x=data['release year'] // 10 * 10,
y=data['user rating score'],
palette='Set1'
).set(
xlabel='Десятилетия',
ylabel='Оценки зрителей'
)
plt.title('Оценки шоу по десятилетиям')
plt.show();
```



Вывод: пользователи начали активно ставить оценки, начиная с 80-90-ых, в пору, когда это вообще можно было делать. До этого оценки пользователей можно считать выбросами, т.к. данных очень мало. Исходя из этих соображений будем делать выводы только по последним 20-30 годам: самые залайканные группы --- `PG` и `TV-PG`. Видим, что за 2010 год оценки выше, чем за предыдущие, это говорит нам о том, что качество шоу улучшелось.

Задача 4

В данной задаче вам нужно будет продолжить анализ данных о видео на YouTube. Информация об имени категории видео содержится в файле `RU_category_id.json`. Следующий код читает этот файл, извлекает из него необходимую информацию, и записывает в виде `pandas`-таблицы.

In [59]:
```
import json

with open('Данные к ДЗ 2/RU_category_id.json') as json_file:
    json_data = json.load(json_file)

category = pd.DataFrame(columns=['id', 'name'])

for item in json_data['items']:
    category = category.append(
        {'id': int(item['id']),
         'name': item['snippet']['title']},
        ignore_index=True
    )

category['id'] = category['id'].astype(int)
category.head()
```

Out[59]:

	id	name
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports

Добавьте к вашим данным имена категорий с помощью `pd.merge`.

In [100]:
```
full_df = pd.merge(category, df_yt, how='outer',
                   left_on='id', right_on='category_id') \
    .set_index('category_id')[['trending_date',
                               'name',
                               'views',
                               'likes',
                               'dislikes',
                               'comment_count']]
full_df.head(1)
```

Out[100]:

category_id	trending_date	name	views	likes	dislikes	comment_count
1.0	14.0	Film & Animation	23316.0	699.0	160.0	217.0

Составьте сводную таблицу о количестве просмотров по дням для каждой категории видео с помощью функции `pivot_table` .

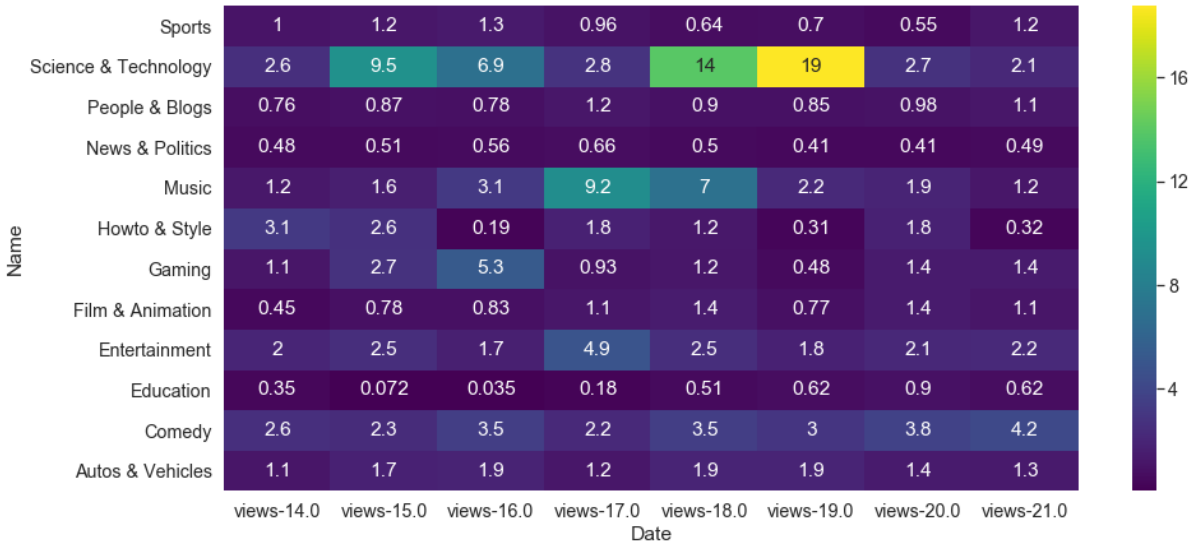
In [61]:
```
pd.pivot_table(full_df, values = ['views'],
               index=['name'],
               columns=['trending_date']).dropna()
```

Out[61]:

	views				
trending_date	14.0	15.0	16.0	17.0	18.0
name					
Autos & Vehicles	112574.166667	173036.916667	186509.818182	115556.333333	1.903430e+05
Comedy	256765.222222	230688.277778	346407.750000	222251.562500	3.460216e+05
Education	34930.750000	7212.500000	3481.500000	18367.875000	5.121520e+04
Entertainment	200415.360000	251880.476190	168667.300000	491997.520000	2.506647e+05
Film & Animation	45472.571429	77653.083333	82865.142857	113809.875000	1.434217e+05
Gaming	110699.625000	268270.166667	527716.333333	92845.666667	1.200614e+05
Howto & Style	313088.428571	260403.500000	18537.571429	178594.857143	1.223449e+05
Music	122317.833333	158561.900000	314697.600000	917939.700000	6.972429e+05
News & Politics	48097.333333	51046.333333	55909.142857	65861.421053	5.028605e+04
People & Blogs	76455.919355	86511.912281	78093.114754	119522.360000	9.042489e+04
Science & Technology	256711.750000	948186.000000	685509.400000	280112.777778	1.391850e+06
Sports	101431.166667	118172.000000	132627.444444	96181.166667	6.426100e+04

Визуализируйте таблицу с помощью `heatmap` . Для информативности поделите все числа на 10^6 .

In [66]:
```
views = pd.pivot_table(full_df, values = ['views'],
                       index=['name'],
                       columns=['trending_date']).dropna()
sns.set(font_scale=1.3)
f, ax = plt.subplots(figsize=(15, 7))
sns.heatmap(views / 100000, annot=True, ax=ax,
            cmap="viridis").set(xlabel='Date', ylabel='Name')
plt.ylim((0, 12));
```



Сделайте аналогичную сводную таблицу, добавив суммы по строкам и столбцам, назвав их "Всего просмотров".

In [101]:
```
views_all = pd.pivot_table(full_df, values = ['views'],
                           index=['name'],
                           columns=['trending_date'],
                           aggfunc=np.sum,
                margins=True, margins_name='Всего просмотров') \
                .dropna()
views_all
```

Out[101]:

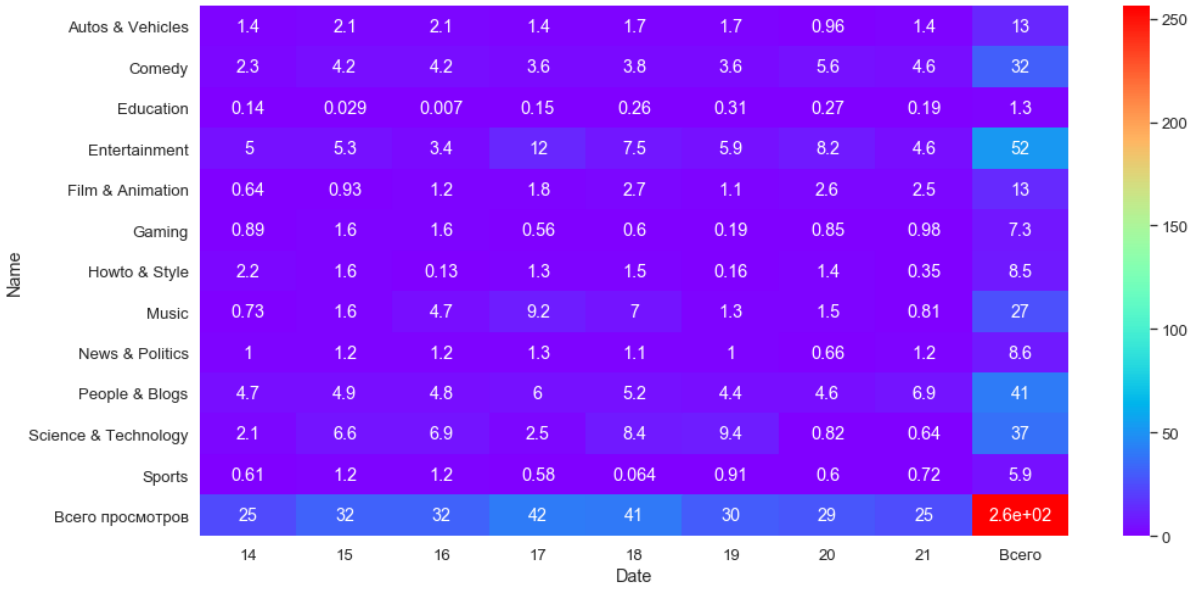
	views						
trending_date	**14.0**	**15.0**	**16.0**	**17.0**	**18.0**	**19.0**	**20.**
name							
Autos & Vehicles	1350890.0	2076443.0	2051608.0	1386676.0	1713087.0	1736740.0	9
Comedy	2310887.0	4152389.0	4156893.0	3556025.0	3806238.0	3556578.0	5
Education	139723.0	28850.0	6963.0	146943.0	256076.0	310830.0	2
Entertainment	5010384.0	5289490.0	3373346.0	12299938.0	7519942.0	5877590.0	8
Film & Animation	636616.0	931837.0	1160112.0	1820958.0	2725012.0	1075440.0	2
Gaming	885597.0	1609621.0	1583149.0	557074.0	600307.0	193316.0	8
Howto & Style	2191619.0	1562421.0	129763.0	1250164.0	1468139.0	156958.0	1
Music	733907.0	1585619.0	4720464.0	9179397.0	6972429.0	1329904.0	1
News & Politics	1010044.0	1225112.0	1174092.0	1251367.0	1056007.0	1028665.0	6
People & Blogs	4740267.0	4931179.0	4763680.0	5976118.0	5154219.0	4353670.0	4
Science & Technology	2053694.0	6637302.0	6855094.0	2521015.0	8351102.0	9396340.0	8
Sports	608587.0	1181720.0	1193647.0	577087.0	64261.0	911646.0	6
Всего просмотров	24741496.0	32005331.0	32328091.0	41674240.0	40765227.0	30347163.0	28

В чем проблема с информативностью подобных таблиц? Исправьте это.

Подсказка: посмотрите на графики, которые вы построили ранее.

In [68]:
```
sns.set(font_scale=1.2)
f, ax = plt.subplots(figsize=(16, 8))
sns.heatmap(views_all / 1000000, annot=True,
            ax=ax, cmap="rainbow",
            xticklabels=[14, 15, 16, 17, 18 ,19, 20 ,21, 'Всего'])
.set(xlabel='Date', ylabel='Name')
```

Out[68]:
```
[Text(118.109375, 0.5, 'Name'), Text(0.5, 46.5, 'Date')]
```



Вывод: последний график информативен, потому что построен с данными в миллион раз меньших реальных, здесь видно, что по дням больше всего смотрели 17 числа (пятница), а по категория больше всего смотрели `Entertainment` . Из этого можно сделать множество хороших выводов, например, стоит выпускать популярные шоу вечером в пятницу и субботу, также стоит увеличивать количество развлекательных шоу.

Задача 5

Yelp (yelp.com) — веб-сайт для поиска на местном рынке услуг, например ресторанов или парикмахерских, с возможностью добавлять и просматривать рейтинги и обзоры этих услуг. Для популярных бизнесов имеются сотни обзоров. Для обозревателей на сайте предусмотрены элементы социальной сети.

Вам предоставляется следующая информация о компаниях на Yelp:

Файл `yelp_business.csv` :

- `business_id` — уникальный идентификатор компании;
- `name` — имя компании;
- `address` , `city` , `state` — месторасположении компании;
- `latitude` , `longitude` — географические координаты;
- `categories` — категории услуг компании.

Файл `yelp_review.csv` , содержащий оценки пользователей:

- `business_id` — идентификатор компании, соответствующий файлу `yelp_business.csv` ;
- `stars` — поставленная пользователем оценка от 1 до 5.

В целях сокращения объема файла, текстовые отзывы пользователей не были включены.

Оригинальную версию датасета в формате json можно посмотреть по ссылке https://www.kaggle.com/yelp-dataset/yelp-dataset/data (https://www.kaggle.com/yelp-dataset/yelp-dataset/data)

Что нужно сделать:

- Найти город с наибольшим количеством компаний;
- Для этого города определить районы с наиболее качественными услугами. Пример с несколько другой задачей: https://yandex.ru/company/researches/2017/msk_mobile_map (https://yandex.ru/company/researches/2017/msk_mobile_map)
- А также найти рестораны с наилучшими отзывами.

Город с наибольшим количеством компаний

Загрузите данные из файла `yelp_business.csv` с помощью функции `pd.read_csv`. Посмотрите на первые несколько строк с помощью метода `head`.

Найдите пять городов, по которым присутствует информация о наибольшем количестве компаний. Для этого стоит воспользоваться методами `groupby`, `count`, `sort_values`, `head`. В таблице должен быть указан город (название) и количество компаний в этом городе.

Пусть N -- город с наибольшим количеством компаний.

Оставьте в таблице только записи, соответствующие городу N. Нанесите все эти компании на график, в котором по оси x отметьте долготу, а по оси y -- долготу.

Сам город находится в сгустке точек. Есть какие-то компании, которые приписаны к этому городу, но находятся далеко от него. Избавьтесь от них, подобрав некоторые границы значений широты и долготы. Изобразите все компании на новом графике.

На этом графике должны выделяться некоторые улицы. Откройте карту города N и сравните ее с построенным графиком.

Попробуйте также автоматически подгружать карту города в качестве фона графика. Примеры. (https://plotly.com/python/scattermapbox/)

Оценки компаний

Для выполнения задания нужно посчитать среднюю оценку каждой компании, а также количество выставленных оценок.

Загрузите таблицу оценок `yelp_review.csv`.

In [69]:
```
business_data = pd.read_csv('Данные к ДЗ 2/yelp_business.csv')
business_data = business_data.loc[:, ['business_id',
                                       'name',
                                       'address',
                                       'city',
                                       'state',
                                       'latitude',
                                       'longitude',
                                       'categories']]
business_data.dropna(how='any')
business_data.drop_duplicates(keep='first')
business_data.head()
```

Out[69]:

	business_id	name	address	city	state	latitude	longi
0	FYWN1wneV18bWNgQjJ2GNg	"Dental by Design"	"4855 E Warner Rd, Ste B9"	Ahwatukee	AZ	33.330690	-111.97
1	He-G7vWjzVUysIKrfNbPUQ	"Stephen Szabo Salon"	"3101 Washington Rd"	McMurray	PA	40.291685	-80.10
2	KQPW8lFf1y5BT2MxiSZ3QA	"Western Motor Vehicle"	"6025 N 27th Ave, Ste 1"	Phoenix	AZ	33.524903	-112.11
3	8DShNS-LuFqpEWIp0HxijA	"Sports Authority"	"5000 Arizona Mills Cr, Ste 435"	Tempe	AZ	33.383147	-111.96
4	PfOCPjBrlQAnz__NXj9h_w	"Brick House Tavern + Tap"	"581 Howe Ave"	Cuyahoga Falls	OH	41.119535	-81.47

In [70]:
```
top5business_data = business_data['city'].value_counts()
top5business_data = top5business_data.head().reset_index()
top5business_data
```

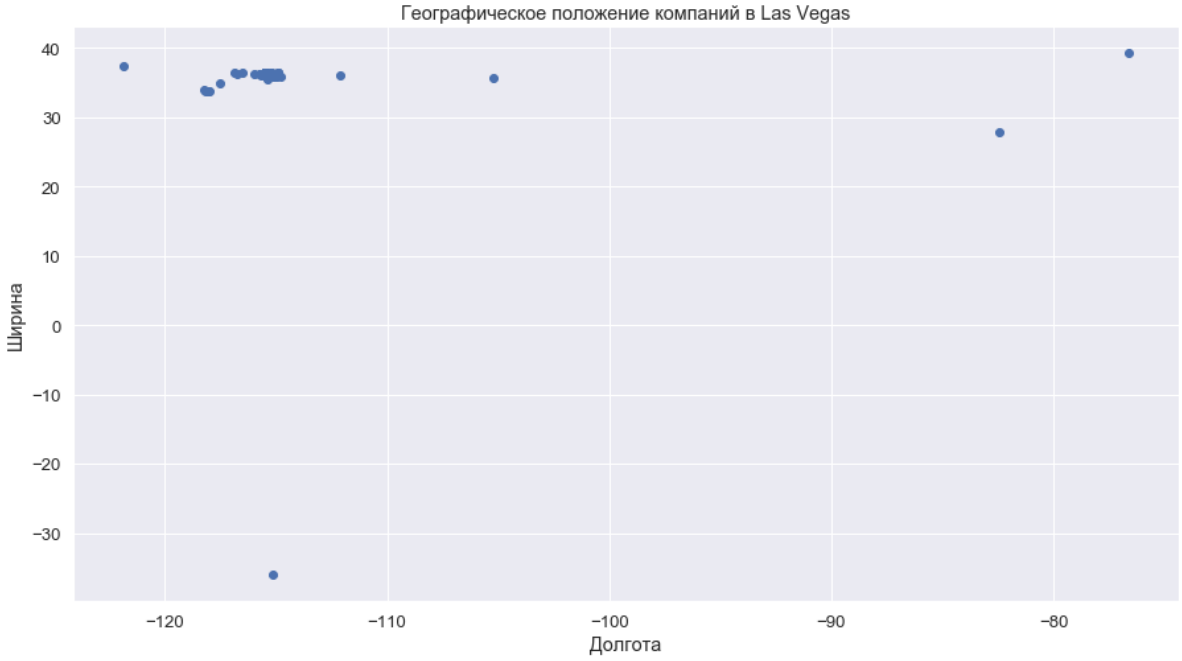
Out[70]:

	index	city
0	Las Vegas	26775
1	Phoenix	17213
2	Toronto	17206
3	Charlotte	8553
4	Scottsdale	8228

In [71]:
```
plt.figure(figsize=(10, 4))
plot = plt.scatter(top5business_data.loc[:, 'index'],
                   top5business_data.city.values, s=60, c='red')
plt.xlabel('Города')
plt.ylabel('Количество компаний')
plt.title('Количество компаний в городах')
plt.show()
```

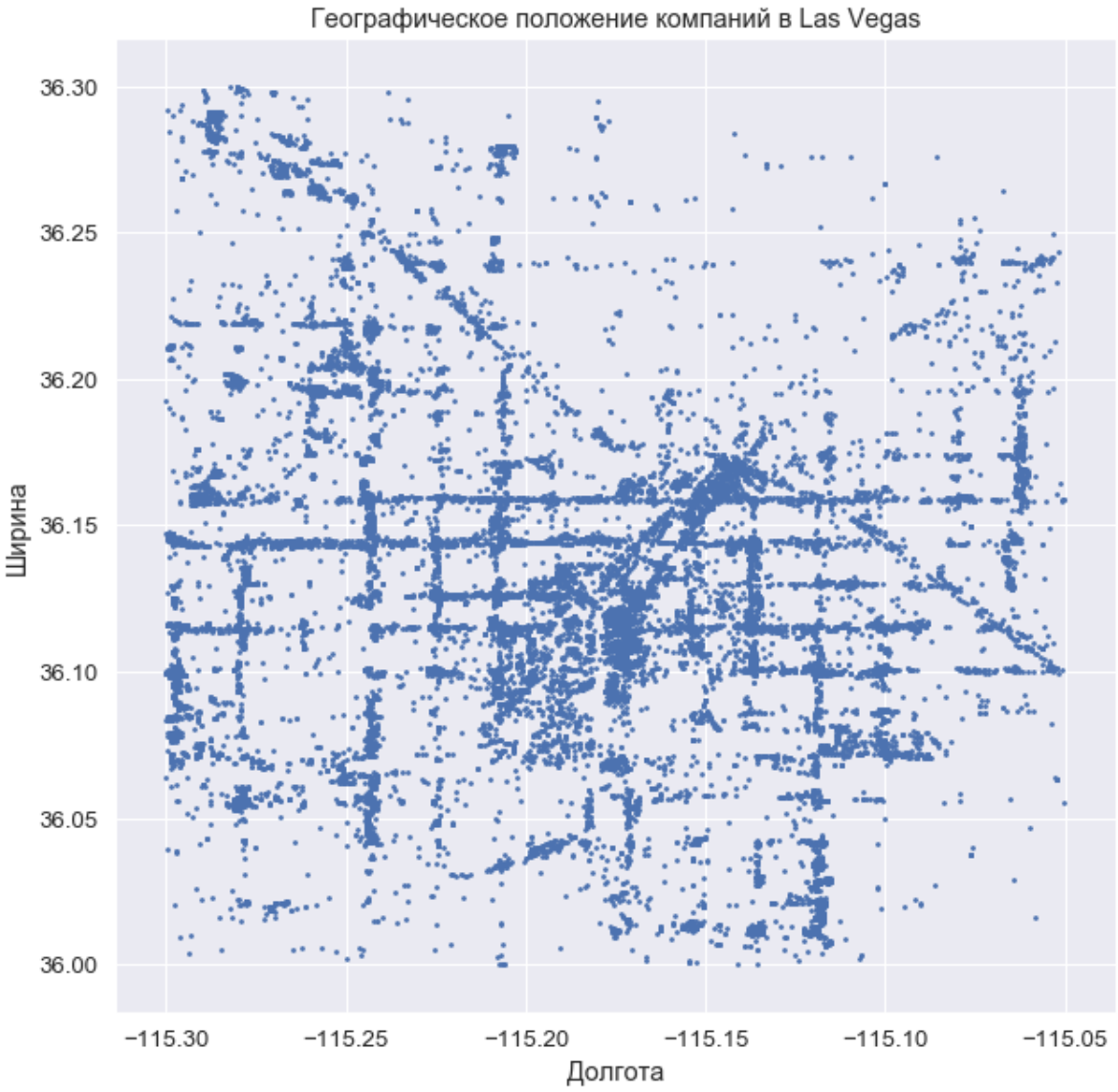
Лучший город по данной статистике оказался Las Vegas

In [72]:
```
N_data = business_data.loc[business_data['city'] == 'Las Vegas']
plt.figure(figsize=(15, 8))
plot = plt.scatter(N_data.loc[:, 'longitude'],
                   N_data.loc[:, 'latitude'])
plt.xlabel('Долгота')
plt.ylabel('Ширина')
plt.title('Географическое положение компаний в Las Vegas')
plt.show()
```



Видим, что некоторые компании расположены далеко за пределами центрального города, однако в мире существует множество объектов, которые политические являются частью города, а географически расположены в другой части мира. Например, Байканур , являющийся сейчас часть РФ, расположен в центре Казахстана. Но не будем об этом и просто отбросил выбросы.

In [102]:
```
N_data = business_data \
.loc[(business_data['city'] == 'Las Vegas')
    & (business_data['longitude']>=-115.3)
    & (business_data['longitude']<=-115.05)
    & (business_data['latitude']>=36)
    & (business_data['latitude']<=36.3)]
N_data_temp = business_data.loc[(business_data['city'] == 'Las Vega
plt.figure(figsize=(10, 10))
plot = plt.scatter(N_data.loc[:, 'longitude'], N_data.loc[:, 'latit
plt.xlabel('Долгота')
plt.ylabel('Ширина')
plt.title('Географическое положение компаний в Las Vegas')
plt.show()
```



Нанесем теперь эти данные на карту `Las Vegas`, подберем нужные цвета, прозрачность и размер и получим нужный результат

In [103]:
```
mapbox_access_token = 'pk.eyJ1Ijoia29zaGFrOTAiLCJhIjoiY2s5bjVpN3czM
```

```
fig = go.Figure(go.Scattermapbox(
lat=N_data.loc[:, 'latitude'],
lon=N_data.loc[:, 'longitude'],
mode='markers',
marker=dict(
size=2,
color='deeppink',
opacity=0.7
)
))

fig.update_layout(
hovermode='closest',
mapbox=dict(
accesstoken=mapbox_access_token,
bearing=0,
center=dict(lat=36.13, lon=-115.2),
zoom=10
),
title_text='Компании зарегестрированные в Las Vegas',
xaxis_title='Широта',
yaxis_title='Долгота',
autosize=False,
width=1000,
height=1000
)

fig.show()
```

(https://www.mapbox.com/)

В подгруженной таблице оценок оставьте только компании города N. Для этого установите значения `business_id` в качестве индекса у таблицы оценок и воспользуйтесь методом `loc`. Чтобы индекс снова сделать полем таблицы, можно воспользоваться методом `reset_index`.

In [75]:
```
mark_data = pd.read_csv('Данные к ДЗ 2/yelp_review.csv')
mark_data.head()
```

Out[75]:

	Unnamed: 0	business_id	stars
0	0	AEx2SYEUJmTxVVB18LlCwA	5
1	1	VR6GpWIda3SfvPC-lg9H3w	5
2	2	CKC0-MOWMqoeWf6s-szl8g	5
3	3	ACFtxLv8pGrrxMm6EgjreA	4
4	4	s2I_Ni76bjJNK9yG60iD-Q	4

Теперь оставим только компании в `Las Vegas`.

In [76]:
```
new_data_N = business_data.loc[(business_data['city'] == 'Las Vegas
mark_data = mark_data.loc[:, ['stars', 'business_id']]
mark_data = mark_data.set_index('business_id') \
    .join(new_data_N.set_index('business_id'), how='inner') \
.reset_index()
mark_data.index.name = 'mark id'
mark_data = mark_data.loc[:, ['stars', 'name', 'business_id']]
mark_data.loc[:, ['stars', 'name']].head(10)
```

Out[76]:

	stars	name
mark id		
0	5	"Delmonico Steakhouse"
1	5	"Delmonico Steakhouse"
2	4	"Delmonico Steakhouse"
3	5	"Delmonico Steakhouse"
4	5	"Delmonico Steakhouse"
5	5	"Delmonico Steakhouse"
6	5	"Delmonico Steakhouse"
7	4	"Delmonico Steakhouse"
8	5	"Delmonico Steakhouse"
9	4	"Delmonico Steakhouse"

Теперь посчитайте среднюю оценку каждой компании, а также количество выставленных компании оценок. Помочь в этом могут функции `groupby` и `aggregate([np.mean, np.size])`.

In [77]:
```
mark_data_agg = mark_data.groupby('business_id')\
.agg([np.mean, np.size])
mark_data_agg.columns = mark_data_agg.columns.droplevel()
mark_data_agg.head(10)
```

Out[77]:

business_id	mean	size
--9e1ONYQuAa-CB_Rrw7Tw	4.088904	1451
--DdmeR16TRb3LsjG0ejrQ	3.200000	5
--WsruI0lGEoeRmkErU5Gg	4.928571	14
--Y7NhBKzLTbNliMUX_wfg	4.875000	8
--e8PjCNhEz32pprnPhCwQ	3.473684	19
--o5BoU7qYMALeVDK6mwVg	3.500000	6
--q7kSBRb0vWC8lSkXFByA	4.000000	7
--z7PM8AGaJP0aBmGMY7RA	4.722222	18
-0BxAGIIk5DJAGVkpqBXxg	2.900000	40
-0KSt9tXv6C015vmAcSIcg	5.000000	3

Назовите колонки таблицы красивыми именами, изменив `<имя таблицы>.columns` , после чего напечатайте несколько строк полученной таблицы.

In [78]:
```
mark_data_agg.rename(columns={'mean': 'rating', 'size': 'count_of_m
                     inplace=True)
mark_data_agg.head(10)
```

Out[78]:

business_id	rating	count_of_marks
--9e1ONYQuAa-CB_Rrw7Tw	4.088904	1451
--DdmeR16TRb3LsjG0ejrQ	3.200000	5
--WsruI0lGEoeRmkErU5Gg	4.928571	14
--Y7NhBKzLTbNliMUX_wfg	4.875000	8
--e8PjCNhEz32pprnPhCwQ	3.473684	19
--o5BoU7qYMALeVDK6mwVg	3.500000	6
--q7kSBRb0vWC8lSkXFByA	4.000000	7
--z7PM8AGaJP0aBmGMY7RA	4.722222	18
-0BxAGlIk5DJAGVkpqBXxg	2.900000	40
-0KSt9tXv6C015vmAcSIcg	5.000000	3

Соедините две полученные ранее таблицы по компаниям города N в одну. Для этого сначала установите поле `business_id` в качестве индекса в обеих таблицах с помощью `set_index` (в одной из них это уже должно было быть сделано). Соединение таблиц можно выполнить с помощью `join` . Индексы у этих таблиц одинаковые, так что тип джойна не имеет значения. В полученной таблице должны получится поля `latitude` , `longitude` , `categories` , `name` , `stars` , `count` .

In [79]:
```
final_data = mark_data_agg.join(new_data_N.set_index('business_id')
                                how='inner').reset_index()
final_data.loc[:, ['latitude',
                   'longitude',
                   'categories',
                   'name',
                   'rating',
                   'count_of_marks']].head(10)
```

Out[79]:

	latitude	longitude	categories	name	rating	c
0	36.123183	-115.169190	Cajun/Creole;Steakhouses;Restaurants	"Delmonico Steakhouse"	4.088904	
1	36.114277	-115.170975	Arts & Entertainment;Festivals	"World Food Championships"	3.200000	
2	36.130899	-115.190785	Carpet Cleaning;Local Services	"Dial Carpet Cleaning"	4.928571	
3	36.061235	-115.289685	Drywall Installation & Repair;Handyman;Home Se...	"Pinnacle Restoration"	4.875000	
4	36.158851	-115.133272	Pets;Pet Groomers;Pet Services;Pet Stores	"Lucky's Pet Grooming & Boutique"	3.473684	
5	36.101643	-115.131625	Event Planning & Services;Music Venues;Arts & ...	"Nightlife Tours"	3.500000	
6	36.016693	-115.173115	Sports Bars;Pizza;Restaurants;Nightlife;Bars	"Double Play Sports Bar"	4.000000	
7	36.135709	-115.175700	Automotive;Auto Repair	"Nacho Mobile Auto Repair"	4.722222	
8	36.122084	-115.168032	Food;Coffee & Tea	"The Coffee Bean & Tea Leaf"	2.900000	
9	36.040216	-115.224485	Landscape Architects;Landscaping;Home Services	"Leisure Lawn"	5.000000	

Изобразите все компании на графике, раскрасив точку в цвет, оттенок которого соответствует средней оценке компании. Прозрачность точки выставляйте не более 0.3.

In [80]:
```
dist_fig = go.Figure(go.Scattermapbox(
    lat=final_data.latitude, lon=final_data.longitude,
    mode='markers',
    marker=dict(size=2, color=final_data.rating,
                opacity=0.7, colorscale='inferno',
                showscale=True)))
dist_fig.update_layout(
    hovermode='closest',
    mapbox=dict(accesstoken=mapbox_access_token,
                bearing=0,
```

```
                bearing=0,
                center=dict(lat=36.13,lon=-115.2),
                pitch=0,
                zoom=10),
    title_text='Компании в Las Vegas',
    xaxis_title='Широта',
    yaxis_title='Долгота',
    autosize=False,
    width=1000,
    height=1000)
dist_fig.show()
```

(https://www.mapbox.com/)

Чтобы получить районы города, округлите значения широты и долготы, подобрав оптимальный размер района. Например, можно сделать так `np.round(долгота*4, decimals=1)*0.25` .

In [81]:
```
final_data['round_latitude']=np.round(final_data.latitude*4, decima
final_data['round_longitude']=np.round(final_data.longitude*4, deci
final_data.head()
```

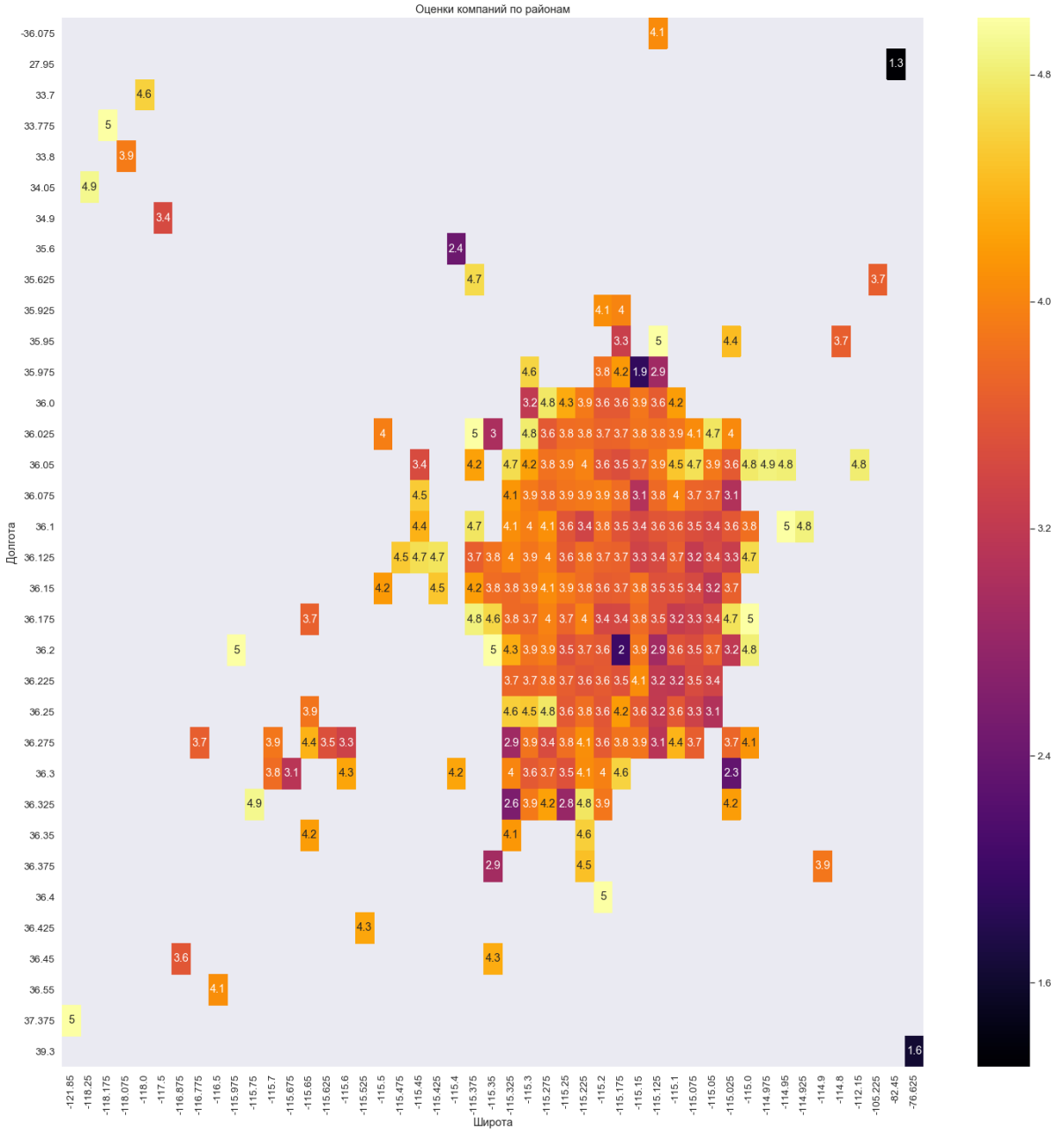
Out[81]:

	business_id	rating	count_of_marks	name	address	city	st
0	--9e1ONYQuAa-CB_Rrw7Tw	4.088904	1451	"Delmonico Steakhouse"	"3355 Las Vegas Blvd S"	Las Vegas	
1	--DdmeR16TRb3LsjG0ejrQ	3.200000	5	"World Food Championships"	"3645 Las Vegas Blvd S"	Las Vegas	
2	--WsruI0lGEoeRmkErU5Gg	4.928571	14	"Dial Carpet Cleaning"	"3111 S Valley Vw, #H-104"	Las Vegas	
3	--Y7NhBKzLTbNliMUX_wfg	4.875000	8	"Pinnacle Restoration"	""	Las Vegas	
4	--e8PjCNhEz32pprnPhCwQ	3.473684	19	"Lucky's Pet Grooming & Boutique"	"1460 E Charleston Blvd"	Las Vegas	

Для получения средней оценки компании по району постройте сводную таблицу при помощи `pd.pivot_table` , взяв в качестве индексов и колонок округленные широту и долготу, а в качестве значений -- оценки. Аггрегирующей функцией является среднее.

Изобразите полученную таблицу при помощи `sns.heatmap` .

In [82]:
```
round_rating = pd.pivot_table(final_data,
                              values='rating',
                              index='round_latitude',
                              columns='round_longitude',
                              aggfunc=np.mean,
                              dropna=True)

with sns.plotting_context(font_scale=1.3):
    plt.figure(figsize=(25, 25))
    sns.heatmap(round_rating, annot=True, cmap="inferno") \
    .set(xlabel='Широта',ylabel='Долгота',
         title='Оценки компаний по районам')
```



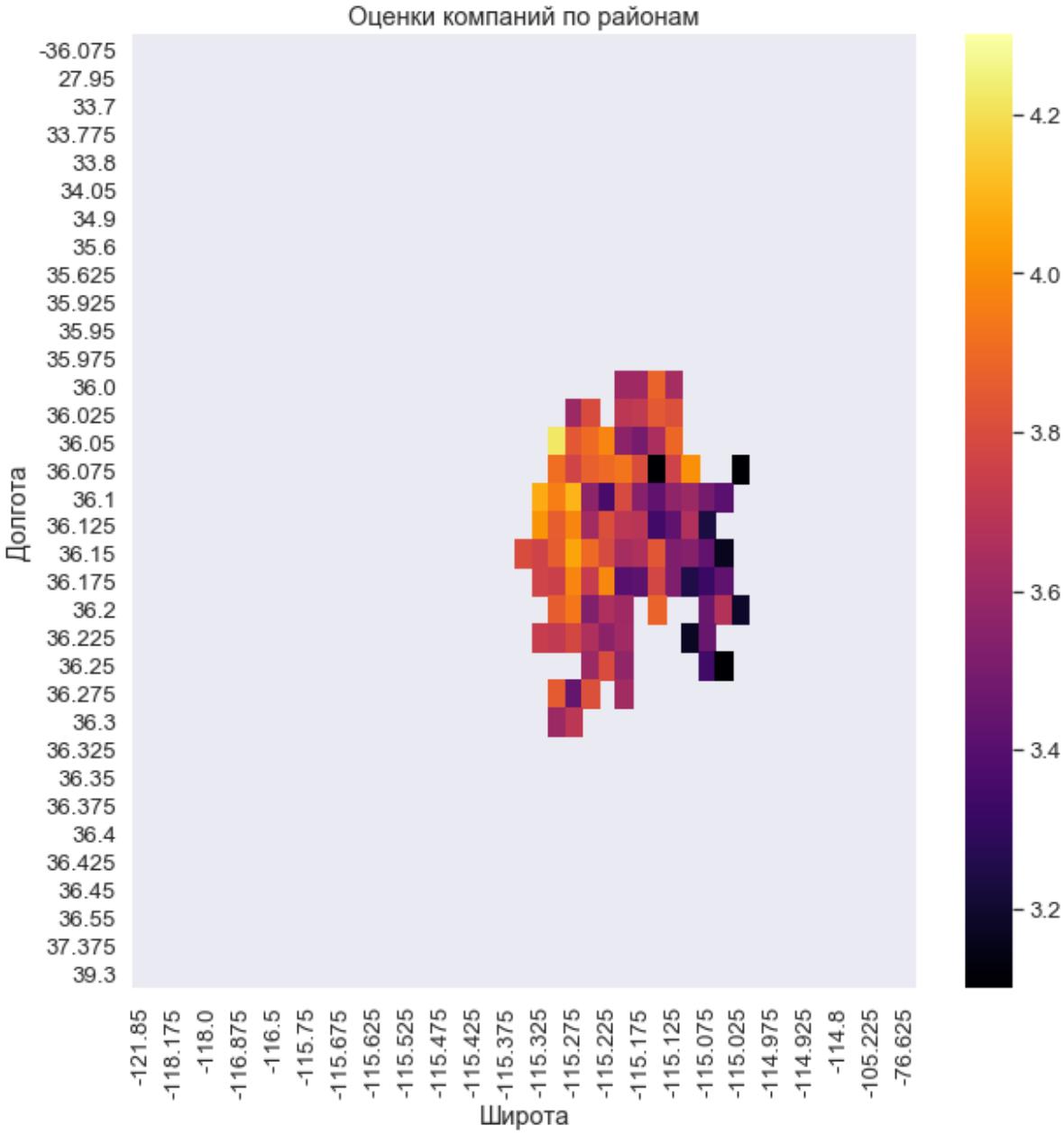
Полученный график имеет ряд недостатков. Во-первых, не очень правильно судить о районе, если в нем мало компаний. Во-вторых, на графике цветовая гамма автоматически подстроилась под минимальное и максимальное значения оценки.

Почему эти недостатки могут быть существенными?

Ответ: эти недостатки могут быть существенными из-за того, что из маленькой выборки почти ничего нельзя сказать (данных просто мало) и ошибочно полагать, что в этом районе все хорошо, опираясь лишь на малькую выборку. Также ничего хорошего не будет, если опираться на цвета районов т.к. при большом количестве компаний цветовая гамма автоматически подстроилась под минимальное и максимальное значения оценки, что приводит к неточной информации.

Оставьте районы, в которых имеется информация о не менее 30 компаний. Постройте новый график районов, использовав параметры `vmin` и `vmax` у функции `sns.heatmap` .

In [83]:
```
round_rating_30 = pd.pivot_table(final_data,
                                 values='rating',
                                 index='round_latitude',
                                 columns='round_longitude',
                                 aggfunc='count',
                                 dropna=True) < 30

with sns.plotting_context(font_scale=1.3):
    plt.figure(figsize=(10, 10))
    sns.heatmap(round_rating, annot=False, cmap="inferno",
                vmin=3.1, vmax=4.3, mask=round_rating_30) \
    .set(xlabel='Широта',ylabel='Долгота',
         title='Оценки компаний по районам')
```



Сравните полученный график с предыдущим и сделайте вывод.

Вывод: теперь можно судить о районах т.к. выборка стала достаточно большой, также цветовая гамма стала разнообразней, что позволяет лучше сравнивать районы между собой.

Рестораны

Будем считать компанию рестораном, если в поле `categories` *содержится* слово `Restaurant` . Составьте таблицу, в которой будет информация о всех ресторанах города N, для которых имеется не менее 5 отзывов. Далее постройте график районов, в котором каждому району сопоставьте среднюю оценку по ресторанам этого района. Рассматривайте только те районы, в которых есть не менее 10 ресторанов, для каждого из которых есть не менее 5 отзывов.

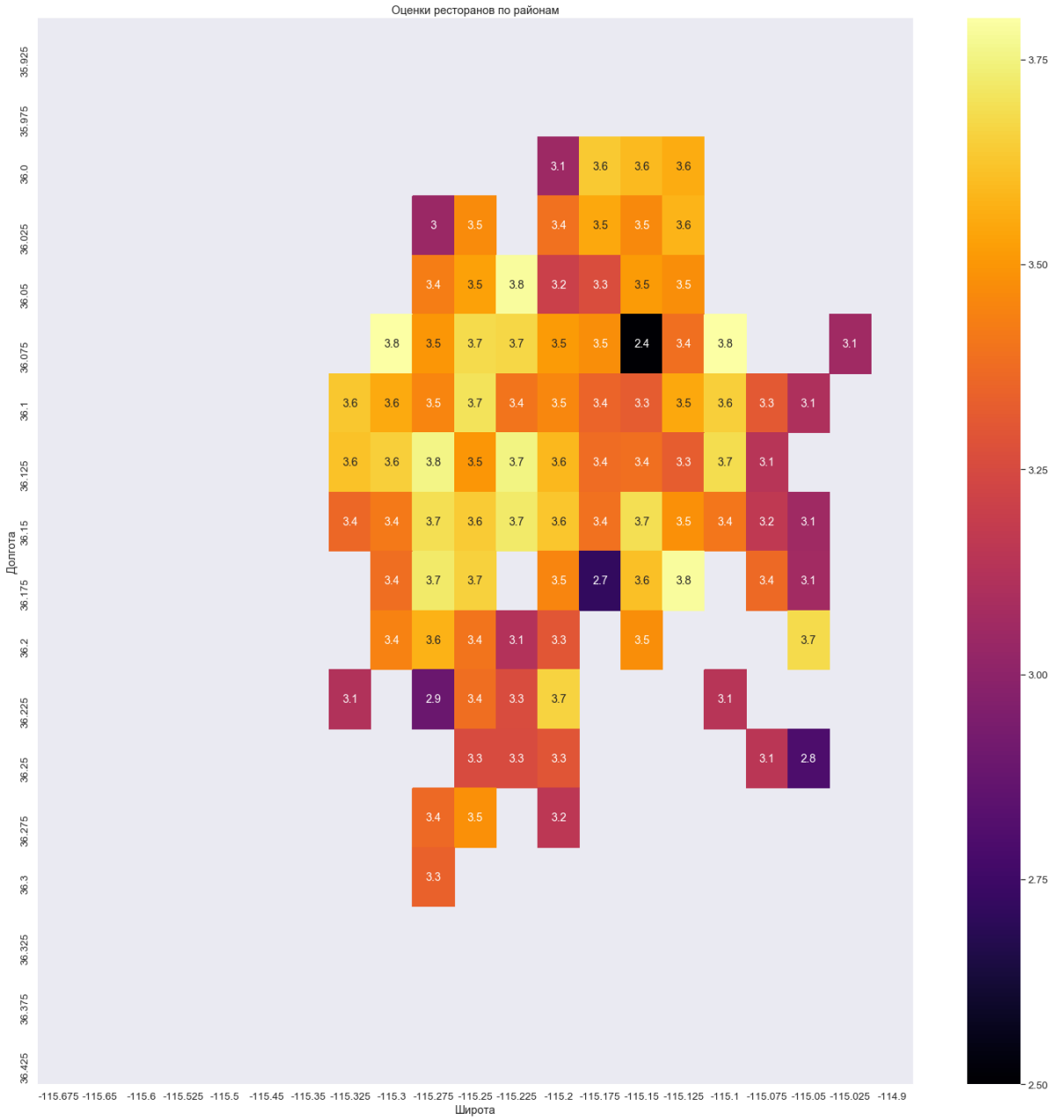
In [88]:
```
mask = ['Restaurant' in category for category in final_data['catego
rest = final_data.loc[mask, :]
rest = rest[rest.count_of_marks >= 5]
rest = rest.loc[:, ['rating',
                    'count_of_marks',
                    'name',
                    'categories',
                    'round_latitude',
                    'round_longitude',
                    'latitude',
                    'longitude']]
rest.head()
```

Out[88]:

	rating	count_of_marks	name	categories	rou
0	4.088904	1451	"Delmonico Steakhouse"	Cajun/Creole;Steakhouses;Restaurants	
6	4.000000	7	"Double Play Sports Bar"	Sports Bars;Pizza;Restaurants;Nightlife;Bars	
14	3.000000	5	"Jody Maroni's Sausage Kingdom"	Hot Dogs;Restaurants	
19	4.736842	38	"Bavette's Steakhouse & Bar"	African;Restaurants;Nightlife;Bars;Steakhouses...	
21	3.226667	75	"Red Ginseng Narita Sushi & BBQ"	Sushi Bars;Korean;Restaurants	

In [89]:
```
round_rating_res = pd.pivot_table(rest,
                                  values='rating',
```

```
                                    index='round_latitude',
                                    columns='round_longitude',
                                    aggfunc=np.mean,
                                    dropna=True)

round_rating_10 = pd.pivot_table(rest,
                                    values='name',
                                    index='round_latitude',
                                    columns='round_longitude',
                                    aggfunc='count',
                                    dropna=True) < 10

with sns.plotting_context(font_scale=1.3):
    plt.figure(figsize=(25, 25))
    sns.heatmap(round_rating_res, annot=True, cmap="inferno",
                vmin=2.5, vmax=3.8, mask=round_rating_10) \
    .set(xlabel='Широта',ylabel='Долгота',
         title='Оценки ресторанов по районам')
```



Чем полезны ограничения снизу на количество отзывов для ресторана и количество ресторанов в районе?

Ответ: ограничение снизу на количество отзывов для ресторана полезенино тем, что не учитывает выбросы (недовольный клиент поставил оценку 1). Ограничение на количество ресторанов в районе помогает более точно описывать рестораны, потому что малых данных выводы делать нельзя.

Кот Василий очень придирчив к выбору ресторана. Он доверяет только ресторанам с высоким рейтингом, который основывается на большом количестве отзывов. Напечатайте в виде таблицы информацию 10 ресторанах с самым большим рейтингом в порядке убывания рейтинга. Для каждого из этих ресторанов должно быть не менее 50 отзывов. По каждому ресторану необходимо вывести следующую информации: название ресторана, средняя оценка, количество отзывов, географические координаты, категории.

In [91]:
```
best_rest = rest[rest.count_of_marks >= 50] \
    .sort_values('rating', ascending=False)
best_rest = best_rest.loc[:,
                          ['name',
                           'rating',
                           'latitude',
                           'longitude']].set_index('name')
best_rest.head(10)
```

Out[91]:

name	rating	latitude	longitude
"Lip Smacking Foodie Tours"	4.966480	36.114537	-115.172678
"Pepito Shack"	4.907692	36.152477	-115.151945
"Bosa Boba Cafe"	4.890909	36.125960	-115.184846
"Garden Grill"	4.868132	36.166783	-115.286197
"Brew Tea Bar"	4.848069	36.054195	-115.242443
"Poppa Naps BBQ"	4.836538	36.116549	-115.088115
"Zenaida's Cafe"	4.833333	36.101741	-115.100359
"El Frescos Cocina Mexicana"	4.816754	36.098527	-115.148446
"Blaqcat Ultra Hookah Lounge"	4.809524	36.159742	-115.232738
"California Sushi Burrito"	4.807018	36.125636	-115.202487

Нанесите на карту все рестораны со средней оценкой не менее 4.7, которая посчитана по не менее 50 отзывам. Отдельным цветом отметьте 10 ресторанов, которые вы получили ранее.

In [96]:
```
best_rest = best_rest[best_rest['rating'] >= 4.7]

fig = go.Figure(go.Scattermapbox(
lat=best_rest.loc[:, 'latitude'],
lon=best_rest.loc[:, 'longitude'],
mode='markers',
marker=dict(
size=5,
color='red',
opacity=1
)
))

fig.update_layout(
hovermode='closest',
mapbox=dict(
accesstoken=mapbox_access_token,
bearing=0,
center=dict(lat=36.13, lon=-115.2),
zoom=10
),
title_text='',
xaxis_title='Широта',
yaxis_title='Долгота',
autosize=False,
width=1000,
height=1000
)


fig.show()
```

(https://www.mapbox.com/)

Охарактеризуйте кота Василия, а также сделайте общий вывод по задаче.

Вывод: Василий очень придирчивый кот, он любит достаточно хорошие по оценке рестораны, которые оценили достаточно много человек. Так и нужно действовать при анализе данных.