北京郵電大學

2022 Python 程序设计大作业

题目: 租房数据分析

姓　　名 郭毓恒

学　　院 计算机学院

专　　业 计算机科学与技术

班　　级 2020211302

学　　号 2020211262

任课教师 杨亚

2022 年 12 月

要求

1. 要求抓取链家官网北上广深 4 个一线城市，再加上一个离你家乡最近的一个非一线城市/或者你最感兴趣的一个城市的租房数据。

2. 应获取每个城市的全部租房数据（一线城市的数据量应该在万的数量级）。

3. 比较 5 个城市的总体房租情况，包含租金的均价、最高价、最低价、中位数等信息，单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息。采用合适的图或表形式进行展示。

4. 比较 5 个城市一居、二居、三居的情况，包含均价、最高价、最低价、中位数等信息。

5. 计算和分析每个城市不同板块的均价情况，并采用合适的图或表形式进行展示。

例如上图中的“海淀-四季青-五福玲珑居北区”，“四季青”即为板块名称。

6. 比较各个城市不同朝向的单位面积租金分布情况，采用合适的图或表形式进行展示。哪个方向最高，哪个方向最低？各个城市是否一致？如果不一致，你认为原因是什么？

7. 查询各个城市的人均 GDP，分析并展示其和单位面积租金分布的关系。相对而言，在哪个城市租房的性价比最高？

8. 查询各个城市的平均工资，分析并展示其和单位面积租金分布的关系。相对而言，在哪个城市租房的负担最重？

9. 请贴上爬虫程序的核心代码、数据处理及数据展示的核心代码（应有足够的注释）。

开发环境

Windows 11 操作系统，Python 3.10

抓取链家官网北上广深 4 个一线城市和泉州市的全部租房数据

begin.py

```python
from scrapy import cmdline

# 同时只能爬一个，其他的暂时变为注释
cmdline.execute("scrapy crawl rent_beijing".split())
# cmdline.execute("scrapy crawl rent_shanghai".split())
# cmdline.execute("scrapy crawl rent_guangzhou".split())
# cmdline.execute("scrapy crawl rent_shenzhen".split())
# cmdline.execute("scrapy crawl rent_quanzhou".split())
```

定义五个爬虫，分别抓取五个城市的租房数据。

items.py

```python
import scrapy

# 五个项分别对应租金、板块、面积、朝向、居数
class Myitem(scrapy.item.Item):
    rent = scrapy.Field()
    block = scrapy.Field()
    area = scrapy.Field()
    direction = scrapy.Field()
    room = scrapy.Field()
```

抓取的数据有五项，分别为租金、板块、面积、朝向和居数。

pipelines.py

```python
import csv

# 输出csv文件
class MyPipeline(object):
    def open_spider(self, spider):
        try:
            self.file = open("quanzhou.csv", 'a', newline='', encoding="utf-8")
        except Exception as err:
            print(err)

    def process_item(self, item, spider):
        list_item = [item['rent'], item['block'], item['area'], item['direction'], item['room']]
        f_csv = csv.writer(self.file)
        f_csv.writerow(list_item)
        return item

    def close_spider(self, spider):
        self.file.close()
```

将数据输出至 csv 文件。

spider.py

```python
import scrapy

from test1.items import Myitem

class Beijingspider(scrapy.spiders.Spider):
    name = "rent_beijing"                           # 爬虫名称
    allowed_domain = ["bj.lianjia.com/"]            # 允许域名
    start_urls = ["https://bj.lianjia.com/zufang/"] # 起始URL
    first_half = "https://bj.lianjia.com"           # 用于之后的拼接
    block_visited = []                              # 已访问的板块

    # 爬取一个板块的数据
    def parse_block(self, response):
        for each in response.xpath("//*[@id='content']/div[1]/div[1]/*"):
            item = Myitem()
            # 网站种的房屋数据只有两种格式有我们所需的数据
            isad = each.xpath("@data-ad_code").extract_first()
            type = each.xpath("@data-distribution_type").extract_first()
            # 租金、板块和方向可以直接爬取，面积需要将爬取的string转换为float，居数需要从爬取的"X室X厅X卫"提取室数
            if isad != "0" and type == "203500000001":
                item['rent'] = int(each.xpath("div/span/em/text()").extract_first())
                item['block'] = each.xpath("div/p[2]/a[2]/text()").extract_first()
                tmp = each.xpath("div/p[2]/text()[6]").extract_first()
                tmp_filter = filter(str.isdigit, tmp)
                tmp_list = list(tmp_filter)
                tmp_str = "".join(tmp_list)
                item['area'] = int(tmp_str) / 100
                item['direction'] = each.xpath("div/p[2]/text()[7]").extract_first().strip()
                room_tmp = each.xpath("div/p[2]/text()[8]").extract_first()
                room_filter = filter(str.isdigit, room_tmp)
                room_list = list(room_filter)
                item['room'] = int(room_list[0])
                yield item
            elif isad == "0" and type == "203500000001":
                item['rent'] = int(each.xpath("div/span/em/text()").extract_first())
                item['block'] = each.xpath("div/p[2]/a[2]/text()").extract_first()
                tmp = each.xpath("div/p[2]/text()[5]").extract_first()
                tmp_filter = filter(str.isdigit, tmp)
                tmp_list = list(tmp_filter)
```

```
            tmp_str = "".join(tmp_list)
            item['area'] = int(tmp_str) / 100
            item['direction'] = each.xpath("div/p[2]/text()[6]").extract_first().strip()
            room_tmp = each.xpath("div/p[2]/text()[7]").extract_first()
            room_filter = filter(str.isdigit, room_tmp)
            room_list = list(room_filter)
            if len(room_list) > 0:
                item['room'] = int(room_list[0])
            yield item

        # 进行翻页
        cur_str = response.xpath("//*[@id='content']/div[1]/div[2]/@data-curpage").extract_first()
        total_str = response.xpath("//*[@id='content']/div[1]/div[2]/@data-totalpage").extract_first()
        if cur_str is not None:
            cur_page = int(cur_str)
            total_page = int(total_str)
            if cur_page < total_page:   # 未到最后一页
                next_page = response.xpath("//*[@id='content']/div[1]/div[2]/@data-url").extract_first()
                next_page = next_page[:-7]              # 移去最后的"{page}"
                next_page += str(cur_page + 1) + "/"    # 加上下一页的页数
                next_page = self.first_half + next_page # 拼接，继续爬取
                yield scrapy.Request(next_page, callback=self.parse_block, dont_filter=True)

    # 爬取一个区的数据
    def parse_district(self, response):
        for each in response.xpath("//*[@id='filter']/ul[4]/*"):    # 遍历可选的板块
            data_id = each.xpath("@data-id").extract_first()
            if data_id is None or data_id == "0":           # 板块选项的第一个是不限，跳过
                continue
            block_page = each.xpath("a/@href").extract_first()
            if self.block_visited.count(block_page) != 0:   # 已爬取过该板块，跳过
                continue
            self.block_visited.append(block_page)           # 没有爬取过该板块，标识为已爬取
            block_page = self.first_half + block_page       # 爬取板块的后半URL，拼接
            yield scrapy.Request(block_page, callback=self.parse_block, dont_filter=True)

    # 爬取该市的数据
    def parse(self, response):
        for each in response.xpath("//*[@id='filter']/ul[2]/*"):    # 在起始URL遍历可选的区
            data_id = each.xpath("@data-id").extract_first()
            if data_id == "0":  # 区选项中的第一个是"不限"，跳过
                continue
            district_page = each.xpath("a/@href").extract_first()   # 爬取区的后半URL，然后拼接
            district_page = self.first_half + district_page
            yield scrapy.Request(district_page, callback=self.parse_district, dont_filter=True)
```

爬虫定义如上，定义了五个爬虫，分别对应五个城市。

链家网站显示北京有三万多条租房数据，但是链家一次最多只能查看 100 页。即使一个个区爬，一个区的数据也可能超过 100 页。因此，需要一个个板块爬。

一个板块可能会出现在不同的区里，因此需要给已经爬过的板块进行标记，避免抓取重复数据。

租房数据的格式并不一致，例如上图的两种，没有板块和朝向，而且租金和面积不确定。

只有这两种租房数据有所需的所有数据，所以只从网站抓取这两种格式的数据。

3000	怀柔其它	82.99	南 北	2
3300	怀柔其它	53.99	南	2
2500	怀柔其它	56	南 北	2
12500	西单	84.27	西	3
7500	西单	42.6	南	1
7500	西单	51	南	1
13500	西单	72.77	南 北	2
7600	西单	40.8	南	1
12000	西单	74	东	2
8500	西单	46.9	南 北	2
7800	西单	52.9	南 北	2

输出的 csv 文件格式如上。

比较 5 个城市的总体房租情况，包含租金的均价、最高价、最低价和中位数信息，单位面积租金（元/平米）的均价、最高价、最低价和中位数信息。

```
import pandas as pd

col = ['rent', 'block', 'area', 'direction', 'room']
beijing_df = pd.read_csv("source_data\\beijing.csv", header=None, names=col)
shanghai_df = pd.read_csv("source_data\\shanghai.csv", header=None, names=col)
guangzhou_df = pd.read_csv("source_data\\guangzhou.csv", header=None, names=col)
shenzhen_df = pd.read_csv("source_data\\shenzhen.csv", header=None, names=col)
quanzhou_df = pd.read_csv("source_data\\quanzhou.csv", header=None, names=col)

# 插入一行，为单位面积租金，取小数点后两位
beijing_df.insert(loc=len(beijing_df.columns), column='unit_rent', value=0)
for index in beijing_df.index:
    beijing_df.loc[index, 'unit_rent'] = round(beijing_df.loc[index, 'rent'] / beijing_df.loc[index, 'area'], 2)

shanghai_df.insert(loc=len(shanghai_df.columns), column='unit_rent', value=0)
for index in shanghai_df.index:
    shanghai_df.loc[index, 'unit_rent'] = round(shanghai_df.loc[index, 'rent'] / shanghai_df.loc[index, 'area'], 2)

guangzhou_df.insert(loc=len(guangzhou_df.columns), column='unit_rent', value=0)
for index in guangzhou_df.index:
    guangzhou_df.loc[index, 'unit_rent'] = round(guangzhou_df.loc[index, 'rent'] / guangzhou_df.loc[index, 'area'], 2)

shenzhen_df.insert(loc=len(shenzhen_df.columns), column='unit_rent', value=0)
for index in shenzhen_df.index:
    shenzhen_df.loc[index, 'unit_rent'] = round(shenzhen_df.loc[index, 'rent'] / shenzhen_df.loc[index, 'area'], 2)

quanzhou_df.insert(loc=len(quanzhou_df.columns), column='unit_rent', value=0)
for index in quanzhou_df.index:
    quanzhou_df.loc[index, 'unit_rent'] = round(quanzhou_df.loc[index, 'rent'] / quanzhou_df.loc[index, 'area'], 2)

beijing_df.to_csv("data_with_unit\\beijing.csv", index=False)
shanghai_df.to_csv("data_with_unit\\shanghai.csv", index=False)
guangzhou_df.to_csv("data_with_unit\\guangzhou.csv", index=False)
shenzhen_df.to_csv("data_with_unit\\shenzhen.csv", index=False)
quanzhou_df.to_csv("data_with_unit\\quanzhou.csv", index=False)
```

计算单位面积租金，取小数点后两位。

rent	block	area	direction	room	unit_rent
3000	怀柔其它	82.99	南 北	2	36.15
3300	怀柔其它	53.99	南	2	61.12
2500	怀柔其它	56	南 北	2	44.64
12500	西单	84.27	西	3	148.33
7500	西单	42.6	南	1	176.06
7500	西单	51	南	1	147.06
13500	西单	72.77	南 北	2	185.52
7600	西单	40.8	南	1	186.27
12000	西单	74	东	2	162.16
8500	西单	46.9	南 北	2	181.24
7800	西单	52.9	南 北	2	147.45

计算完单位面积租金后输出的 csv 文件格式如上。

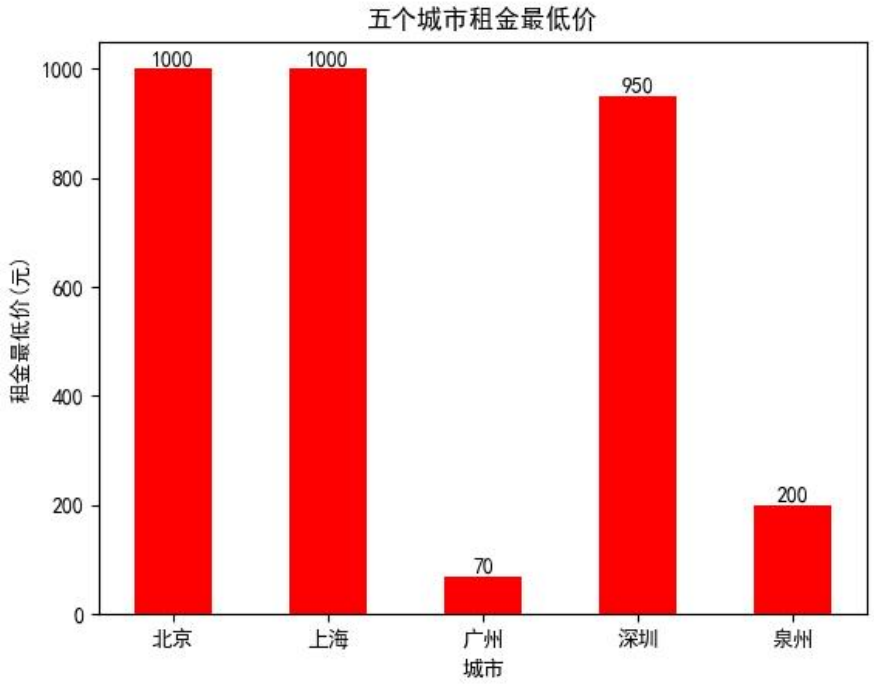
```
# 计算均价、最大值、最小值、中位数，取小数点后两位
beijing_mean = beijing_df['rent'].mean()
beijing_mean = round(beijing_mean, 2)
beijing_high = beijing_df['rent'].max()
beijing_low = beijing_df['rent'].min()
beijing_mid = beijing_df['rent'].median()
beijing_mid = round(beijing_mid, 2)
```

给每个城市的租金计算均价、最大值、最小值和中位数。均价和中位数区小数点后两位数。

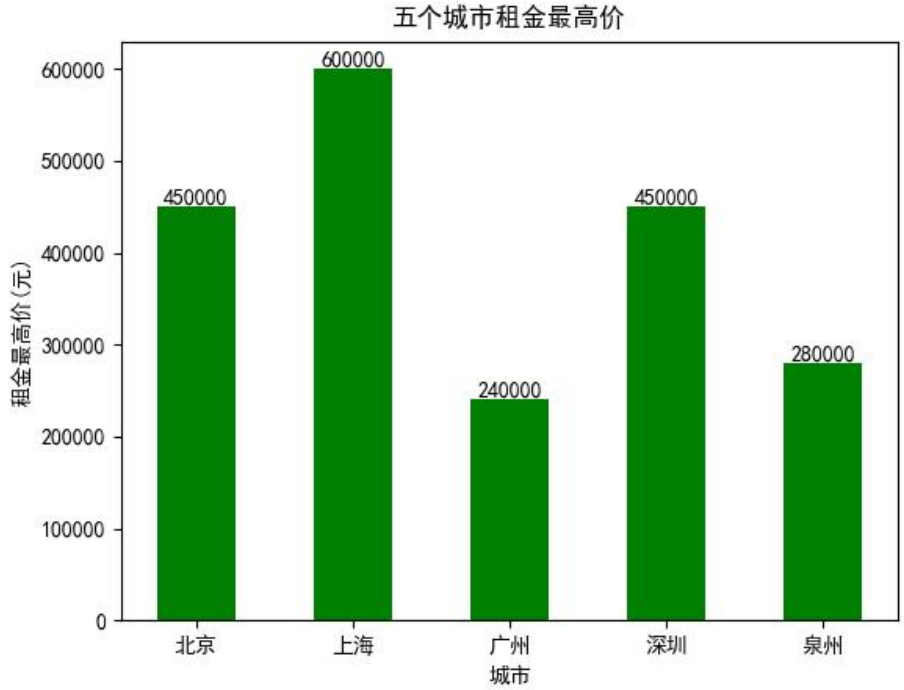
```
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
city = ['北京', '上海', '广州', '深圳', '泉州']   # x刻度标签
x = [1, 2, 3, 4, 5]                              # x坐标
# 按最小值、最大值、均价和中位数画四张图
low = [beijing_low, shanghai_low, guangzhou_low, shenzhen_low, quanzhou_low]
mean = [beijing_mean, shanghai_mean, guangzhou_mean, shenzhen_mean, quanzhou_mean]
mid = [beijing_mid, shanghai_mid, guangzhou_mid, shenzhen_mid, quanzhou_mid]
high = [beijing_high, shanghai_high, guangzhou_high, shenzhen_high, quanzhou_high]

plt.figure()
plt.bar(x, low, 0.5, color='r', tick_label=city)
for a, b in zip(x, low):
    plt.text(a, b+0.2, b, ha='center', va='bottom')
plt.title('五个城市租金最低价')
plt.xlabel('城市')
plt.ylabel('租金最低价(元)')
plt.savefig('fig_q2\\五个城市租金最低价.png')
```

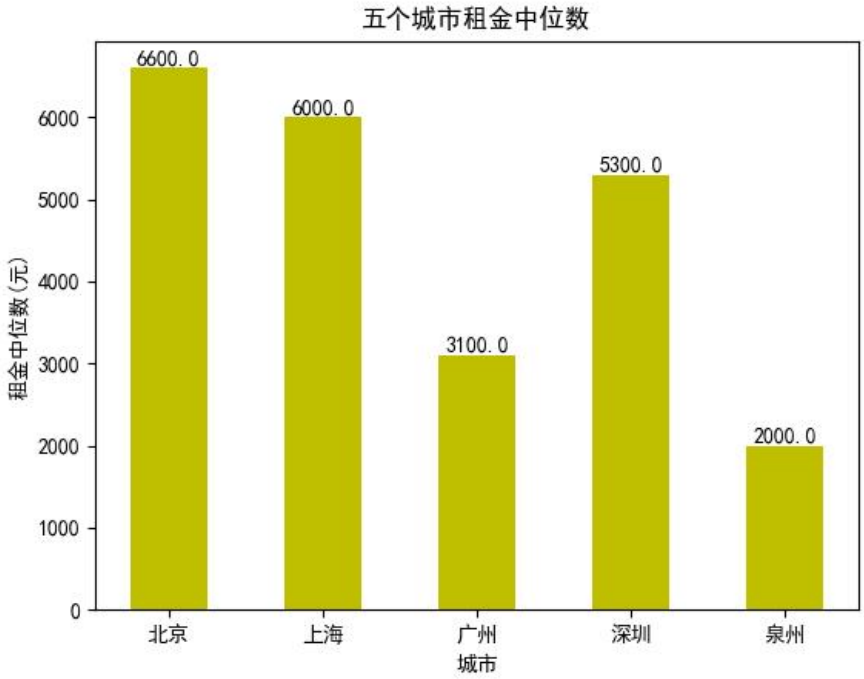
按以上设置画四张图，分别为五个城市租金的最小值、最大值、均价和中位数。



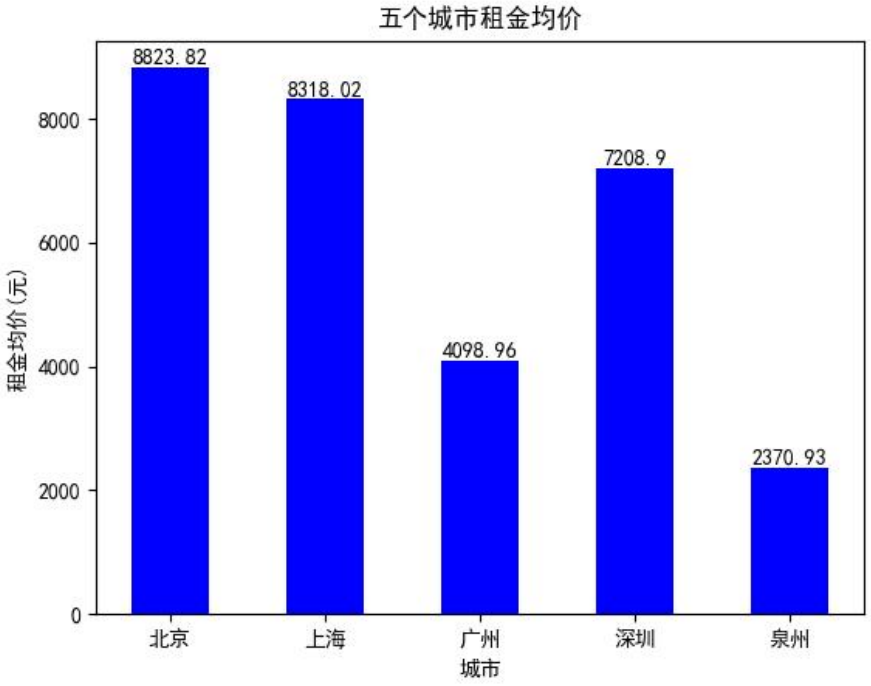
北京和上海的租金最低价并列最高，深圳其次。泉州和广州的租金最低价与另外三个的差距很大，泉州倒数第二，广州最低。



上海的租金最高价最高，北京和深圳并列第二，泉州第三，广州最低。

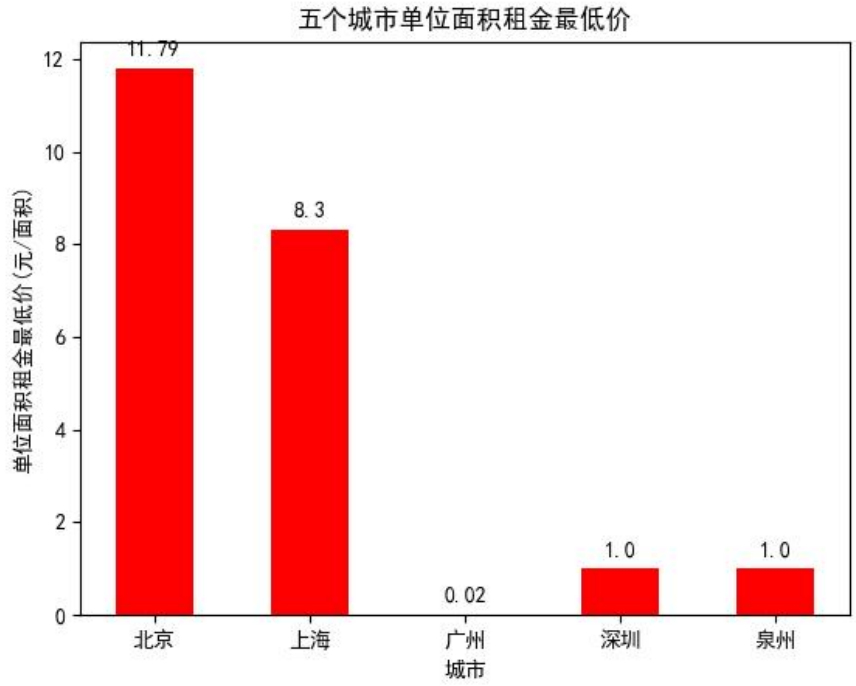


北京的租金中位数最高，上海第二，深圳第三，广州第四，泉州最低。

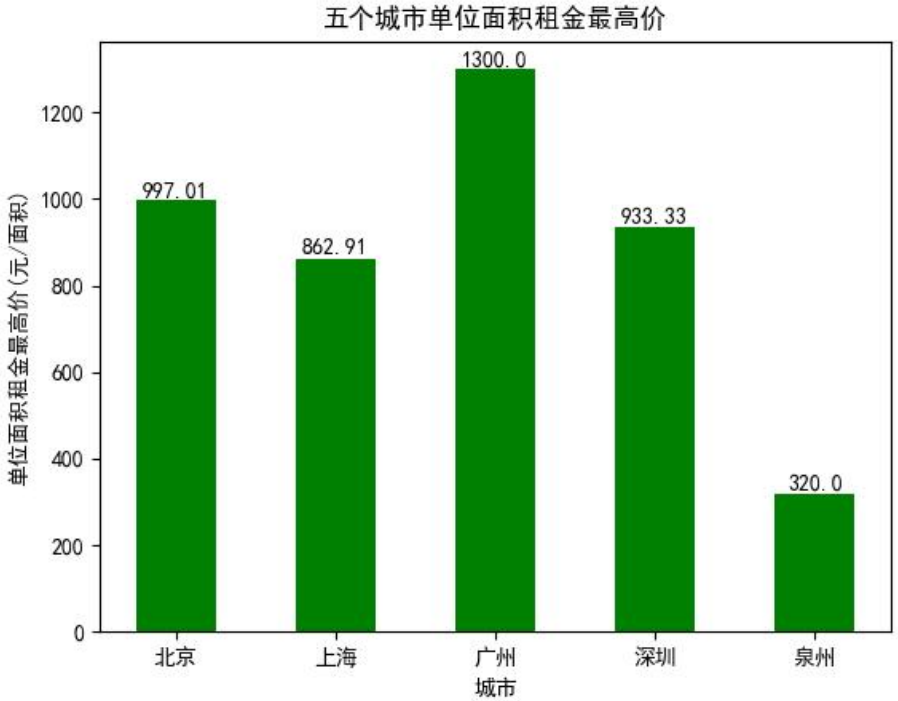


北京的租金均价最高，上海第二，深圳第三，广州第四，泉州最低。

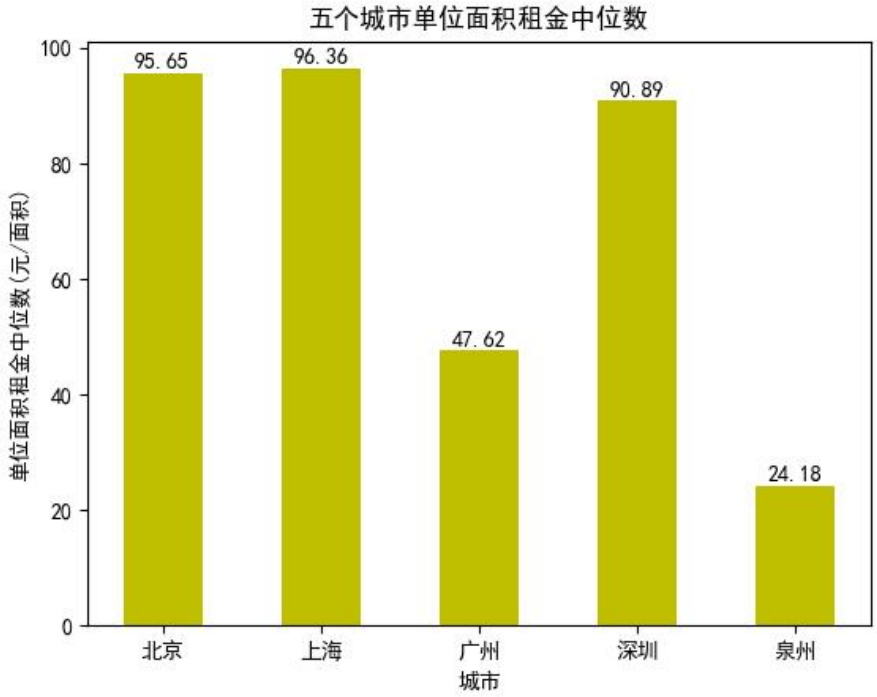
五个城市的租金中位数和均价的排序一样，并且最高价和最低价可能存在极端值，应该以均价和中位数的排序为准。北京的租金最高，上海第二，深圳第三，广州第四，泉州最低。



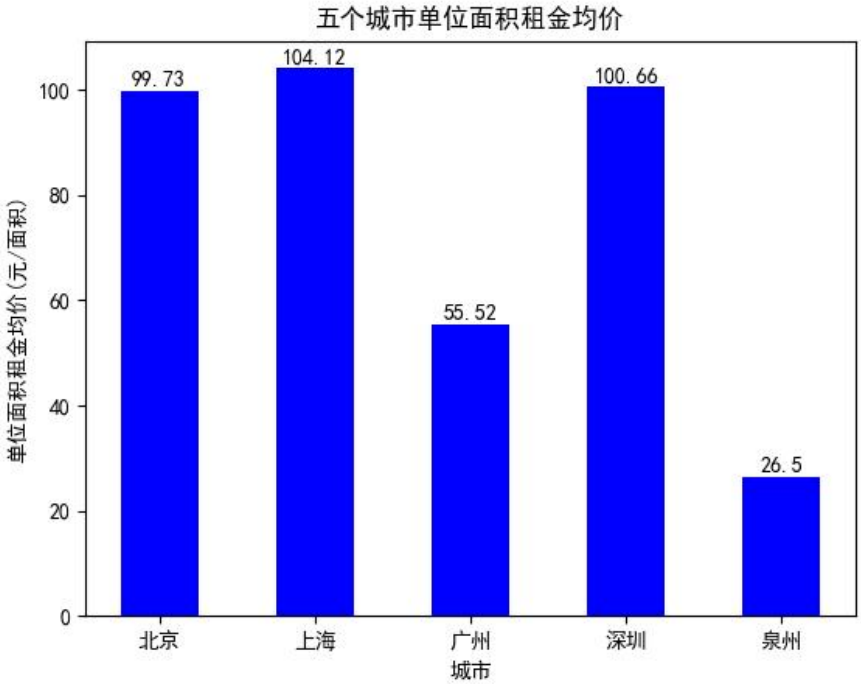
北京的单位面积租金最低价最高，上海第二。广州、深圳和泉州的数据与前两名的差距很大，深圳和泉州并列第三，广州最低。



广州的单位面积租金最高价最高，北京第二，深圳第三，上海第四，泉州最低。



上海的单位面积租金中位数最高，北京第二，深圳第三，广州第四，泉州最低。



上海的单位面积租金均价最高，深圳第二，北京第三，广州第四，泉州最低。

最高价和最低价可能存在极端值，应该以均价和中位数的排序为准。北京、上海和深圳的数据很接近，是五个城市的最高。然后广州第四，泉州最低。

比较 5 个城市一居、二居、三居的情况，包含均价、最高价、最低价和中位数信息。

```
# 只需要居数、租金和单位面积租金的数据
beijing_room = beijing_df[['room', 'rent', 'unit_rent']]
shanghai_room = shanghai_df[['room', 'rent', 'unit_rent']]
guangzhou_room = guangzhou_df[['room', 'rent', 'unit_rent']]
shenzhen_room = shenzhen_df[['room', 'rent', 'unit_rent']]
quanzhou_room = quanzhou_df[['room', 'rent', 'unit_rent']]

# 按居数给所有df进行分组
beijing_room_dict = dict(list(beijing_room.groupby('room')))
shanghai_room_dict = dict(list(shanghai_room.groupby('room')))
guangzhou_room_dict = dict(list(guangzhou_room.groupby('room')))
shenzhen_room_dict = dict(list(shenzhen_room.groupby('room')))
quanzhou_room_dict = dict(list(quanzhou_room.groupby('room')))
```

按照居数给数据进行分组。

```
# 计算所有城市的最小值、最大值、均价和中位数
room1_low, room1_high, room1_mean, room1_mid = get_stat(1)
room2_low, room2_high, room2_mean, room2_mid = get_stat(2)
room3_low, room3_high, room3_mean, room3_mid = get_stat(3)
```

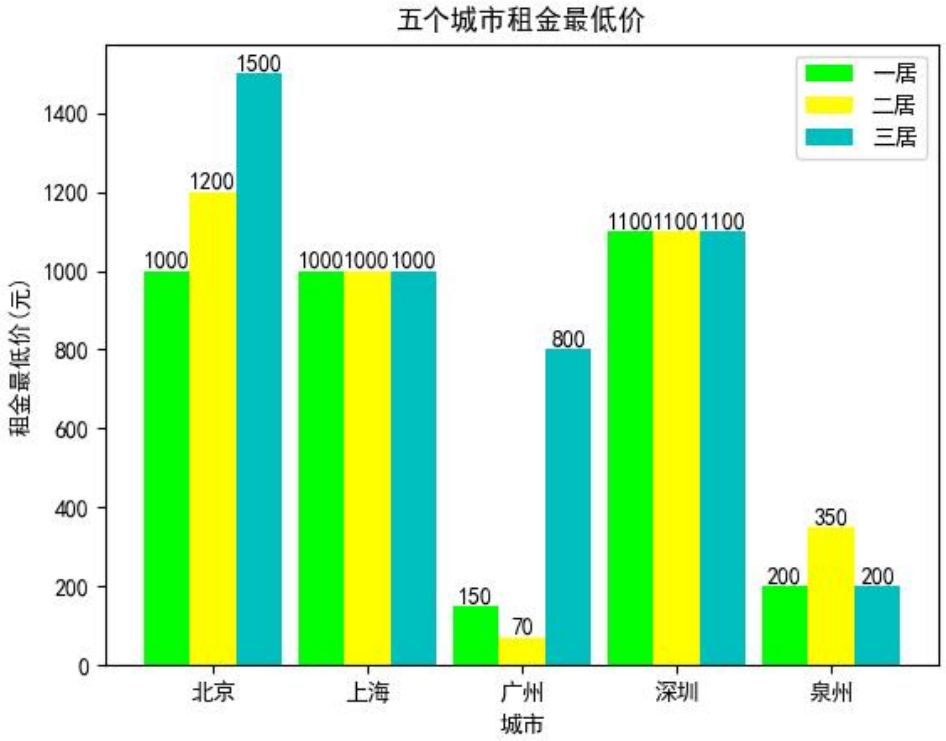
```
def get_stat(room_num):  # 计算最小值、最大值、均价和中位数
    room_low = [[beijing_room_dict[room_num]['rent'].min(), shanghai_room_dict[room_num]['rent'].min(),
                 guangzhou_room_dict[room_num]['rent'].min(), shenzhen_room_dict[room_num]['rent'].min(),
                 quanzhou_room_dict[room_num]['rent'].min()],
                [beijing_room_dict[room_num]['unit_rent'].min(), shanghai_room_dict[room_num]['unit_rent'].min(),
                 guangzhou_room_dict[room_num]['unit_rent'].min(), shenzhen_room_dict[room_num]['unit_rent'].min(),
                 quanzhou_room_dict[room_num]['unit_rent'].min()]]
    room_high = [[beijing_room_dict[room_num]['rent'].max(), shanghai_room_dict[room_num]['rent'].max(),
                  guangzhou_room_dict[room_num]['rent'].max(), shenzhen_room_dict[room_num]['rent'].max(),
                  quanzhou_room_dict[room_num]['rent'].max()],
                 [beijing_room_dict[room_num]['unit_rent'].max(), shanghai_room_dict[room_num]['unit_rent'].max(),
                  guangzhou_room_dict[room_num]['unit_rent'].max(), shenzhen_room_dict[room_num]['unit_rent'].max(),
                  quanzhou_room_dict[room_num]['unit_rent'].max()]]
    room_mean = [[round(beijing_room_dict[room_num]['rent'].mean(), 2),
                  round(shanghai_room_dict[room_num]['rent'].mean(), 2),
                  round(guangzhou_room_dict[room_num]['rent'].mean(), 2),
                  round(shenzhen_room_dict[room_num]['rent'].mean(), 2),
                  round(quanzhou_room_dict[room_num]['rent'].mean(), 2)],
                 [round(beijing_room_dict[room_num]['unit_rent'].mean(), 2),
                  round(shanghai_room_dict[room_num]['unit_rent'].mean(), 2),
                  round(guangzhou_room_dict[room_num]['unit_rent'].mean(), 2),
                  round(shenzhen_room_dict[room_num]['unit_rent'].mean(), 2),
                  round(quanzhou_room_dict[room_num]['unit_rent'].mean(), 2)]]
    room_mid = [[round(beijing_room_dict[room_num]['rent'].median(), 2),
                 round(shanghai_room_dict[room_num]['rent'].median(), 2),
                 round(guangzhou_room_dict[room_num]['rent'].median(), 2),
                 round(shenzhen_room_dict[room_num]['rent'].median(), 2),
                 round(quanzhou_room_dict[room_num]['rent'].median(), 2)],
                [round(beijing_room_dict[room_num]['unit_rent'].median(), 2),
                 round(shanghai_room_dict[room_num]['unit_rent'].median(), 2),
                 round(guangzhou_room_dict[room_num]['unit_rent'].median(), 2),
                 round(shenzhen_room_dict[room_num]['unit_rent'].median(), 2),
                 round(quanzhou_room_dict[room_num]['unit_rent'].median(), 2)]]
    return room_low, room_high, room_mean, room_mid
```

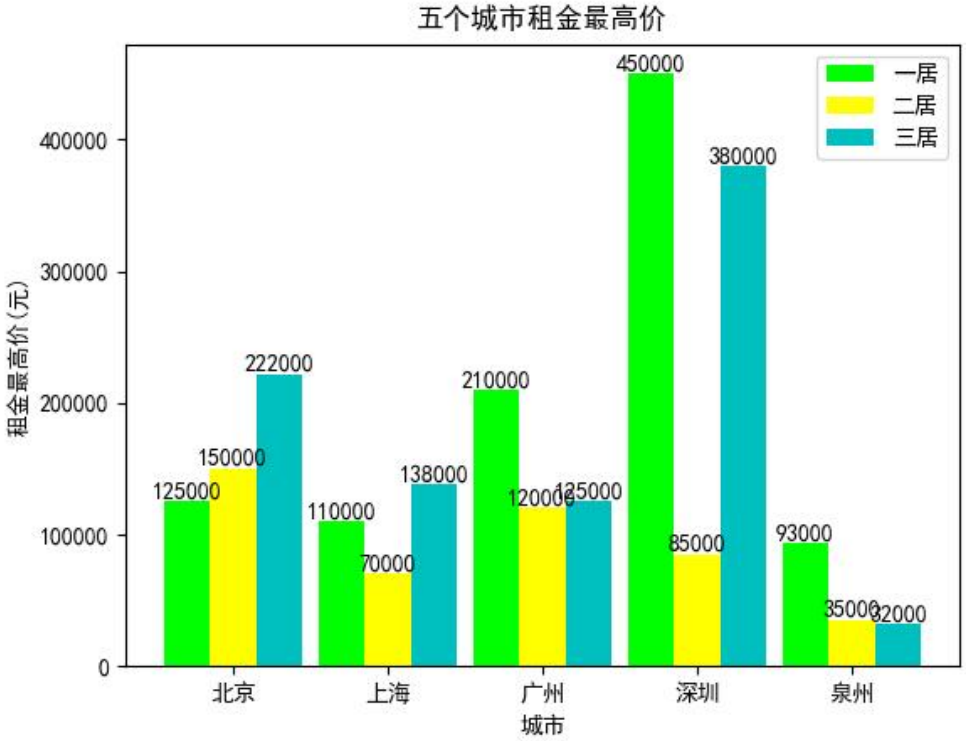
分别计算五个城市一居、二居和三居的租金和单位面积租金的最小值、最大值、均价和中位数。中位数和均价取小数点后两位。

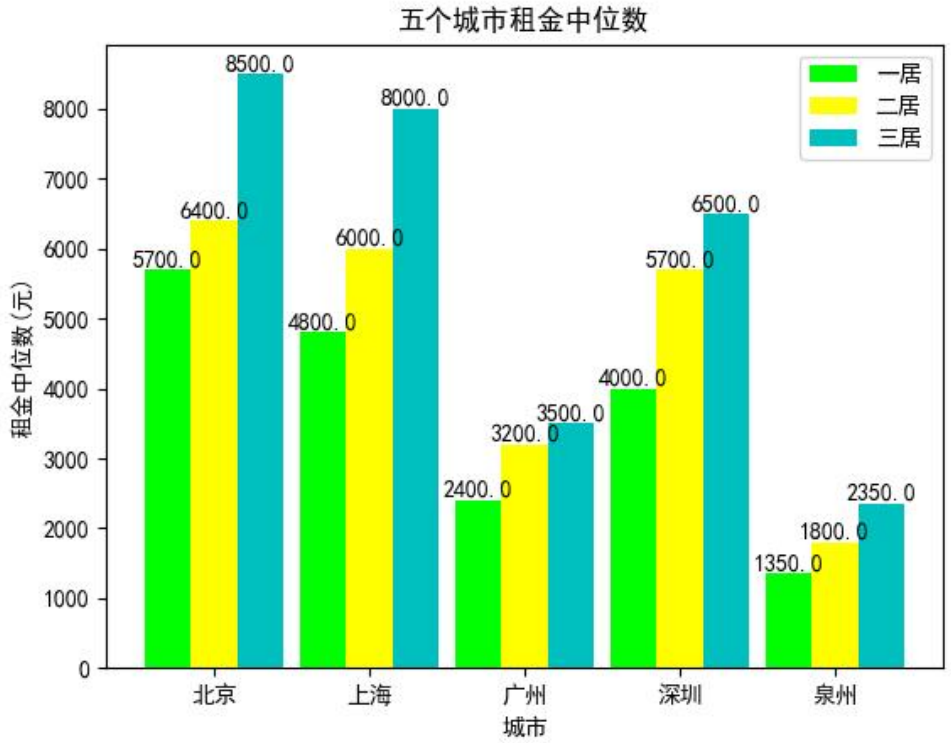
```python
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
city = ['北京', '上海', '广州', '深圳', '泉州']      # x刻度标签
room = ['一居', '二居', '三居']                      # legend标签
x = np.array([1, 2, 3, 4, 5])                       # x坐标
# 画图
plot_graph(1, room1_low[0], room2_low[0], room3_low[0])
plot_graph(2, room1_high[0], room2_high[0], room3_high[0])
plot_graph(3, room1_mean[0], room2_mean[0], room3_mean[0])
plot_graph(4, room1_mid[0], room2_mid[0], room3_mid[0])
plot_graph(5, room1_low[1], room2_low[1], room3_low[1])
plot_graph(6, room1_high[1], room2_high[1], room3_high[1])
plot_graph(7, room1_mean[1], room2_mean[1], room3_mean[1])
plot_graph(8, room1_mid[1], room2_mid[1], room3_mid[1])
```

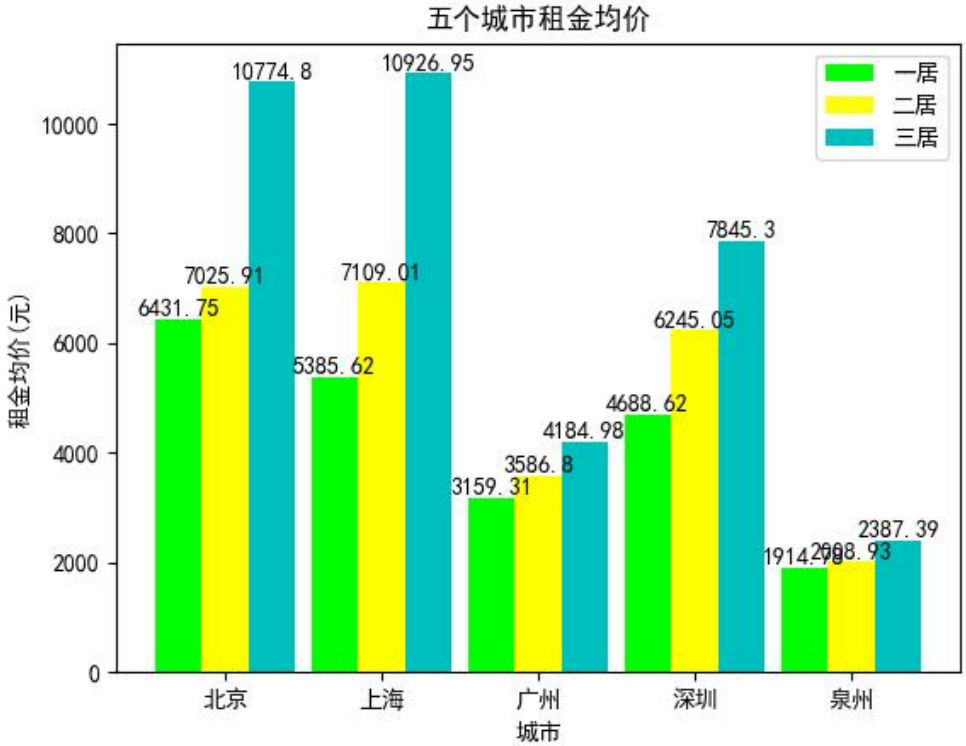
```python
def plot_graph(graph_type, data1, data2, data3):    # 画图函数
    unit = ''
    title = ''
    # 根据要画的图的类型，设置标题和单元
    if graph_type < 5:
        unit = '(元)'
        if graph_type == 1:
            title = '租金最低价'
        elif graph_type == 2:
            title = '租金最高价'
        elif graph_type == 3:
            title = '租金均价'
        elif graph_type == 4:
            title = '租金中位数'
    else:
        unit = '(元/面积)'
        if graph_type == 5:
            title = '单位面积租金最低价'
        elif graph_type == 6:
            title = '单位面积租金最高价'
        elif graph_type == 7:
            title = '单位面积租金均价'
        elif graph_type == 8:
            title = '单位面积租金中位数'
    plt.figure()
    plt.bar(x - 0.3, data1, 0.3, color='lime')
    plt.bar(x, data2, 0.3, color='yellow', tick_label=city)
    plt.bar(x + 0.3, data3, 0.3, color='c')
    for a, b in zip(x, data1):
        plt.text(a - 0.3, b + 0.2, b, ha='center', va='bottom')
    for a, b in zip(x, data2):
        plt.text(a, b + 0.2, b, ha='center', va='bottom')
    for a, b in zip(x, data3):
        plt.text(a + 0.3, b + 0.2, b, ha='center', va='bottom')
    plt.title('五个城市' + title)
    plt.xlabel('城市')
    plt.ylabel(title + unit)
    plt.legend(room, loc='upper right')
    plt.savefig('fig_q3\\五个城市' + title + '.png')
```

按照以上设置进行画图，总共八个图。

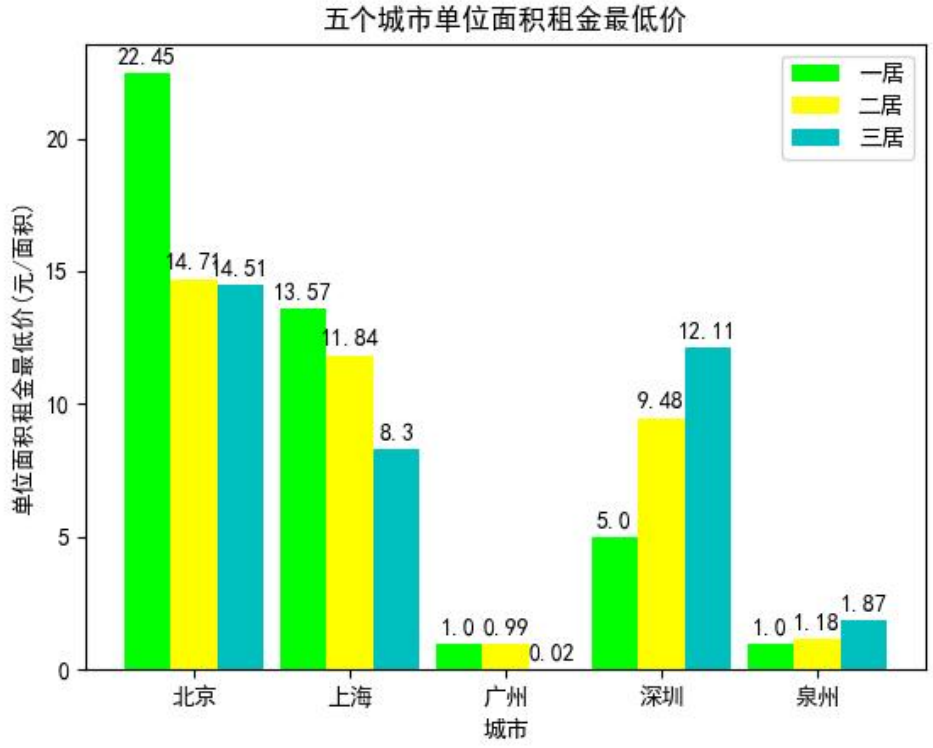

五个城市租金最低价
一居
二居
三居
租金最低价(元)
城市
北京
上海
广州
深圳
泉州
1000
1200
1500
1000
1000
1000
150
70
800
1100
1100
1100
200
350
200

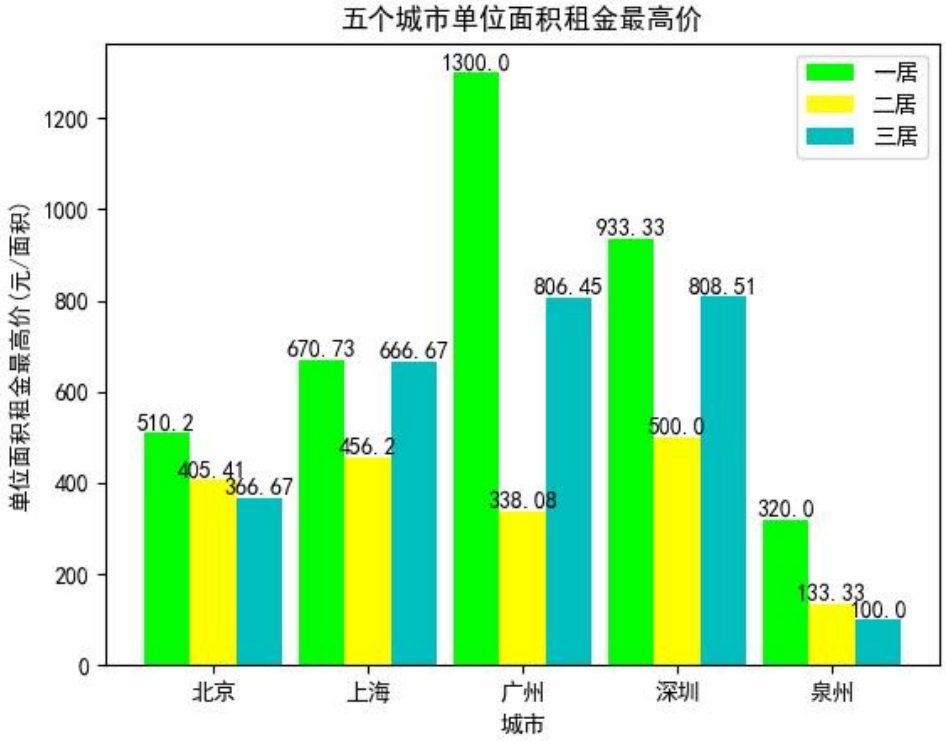

五个城市租金最高价
一居
二居
三居
租金最高价(元)
城市
北京
上海
广州
深圳
泉州
125000
150000
222000
110000
70000
138000
210000
120000
125000
450000
85000
380000
93000
35000
32000

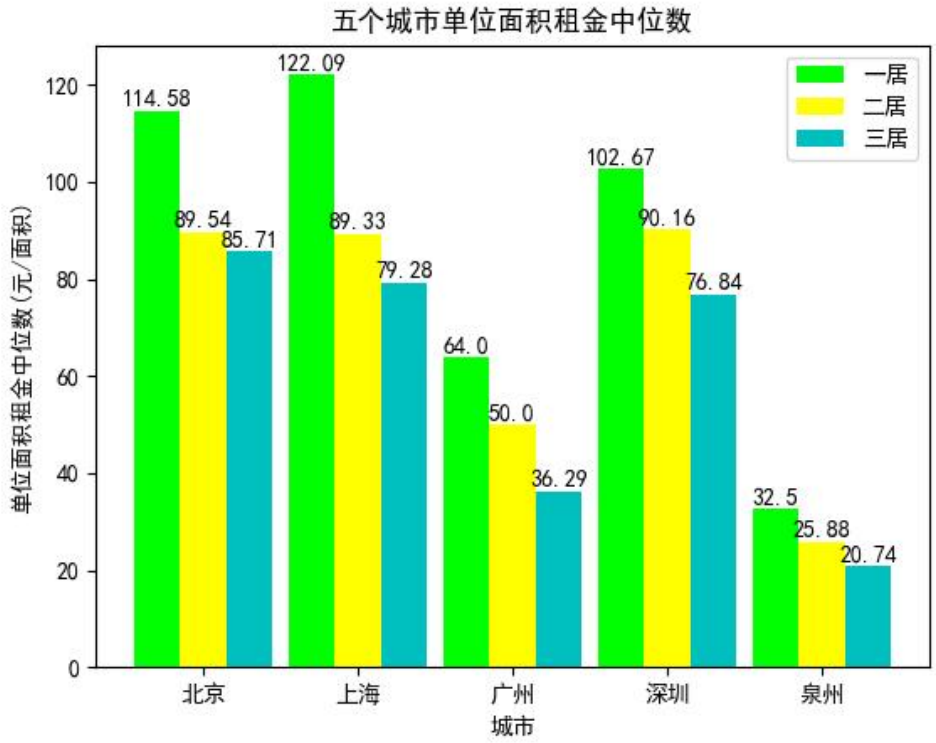


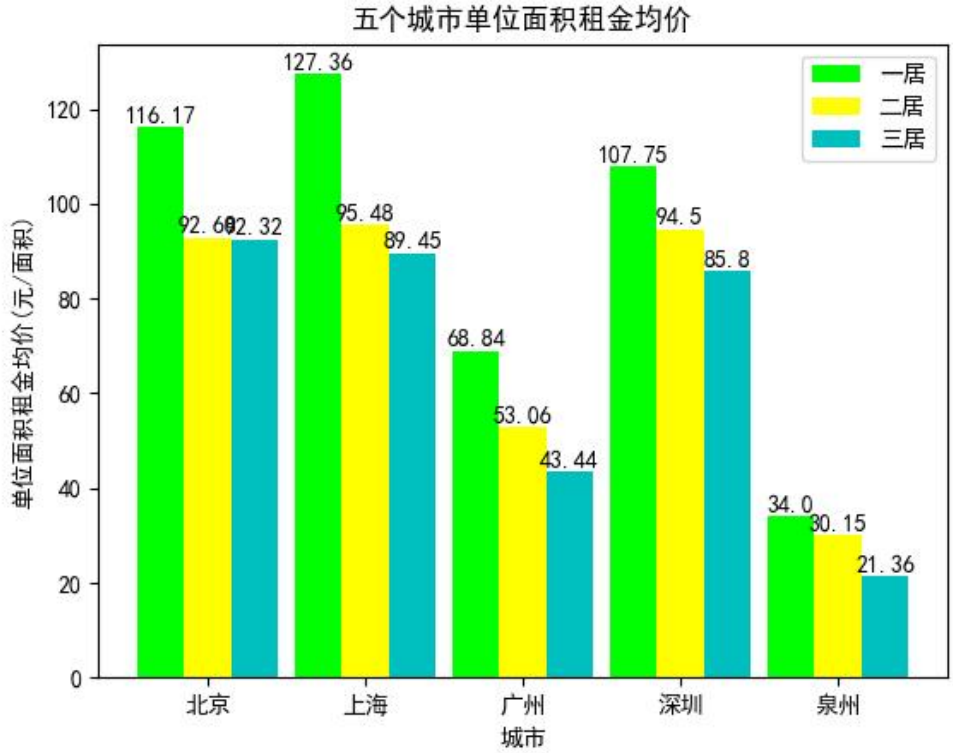


租金最高价和最低价可能存在极端值，所以只考虑均价和中位数。越多居的房子租金越高，北京和上海的租金很接近，是五个城市的最高。深圳第三，广州第四，泉州最低。


五个城市单位面积租金最低价
一居
二居
三居
单位面积租金最低价(元/面积)
0
5
10
15
20
22.45
14.71
14.51
13.57
11.84
8.3
1.0
0.99
0.02
5.0
9.48
12.11
1.0
1.18
1.87
北京
上海
广州
深圳
泉州
城市


五个城市单位面积租金最高价
一居
二居
三居
单位面积租金最高价(元/面积)
0
200
400
600
800
1000
1200
510.2
405.41
366.67
670.73
456.2
666.67
1300.0
338.08
806.45
933.33
500.0
808.51
320.0
133.33
100.0
北京
上海
广州
深圳
泉州
城市





单位面积租金最高价和最低价可能存在极端值，所以只考虑均价和中位数。越少居的房子单位面积租金越高，北京、上海和深圳的租金很接近，是五个城市的最高。然后广州第四，泉州最低。

计算和分析每个城市不同板块的均价情况

```
# 只需要板块和租金的数据
beijing_block = beijing_df[['block', 'rent']]
shanghai_block = shanghai_df[['block', 'rent']]
guangzhou_block = guangzhou_df[['block', 'rent']]
shenzhen_block = shenzhen_df[['block', 'rent']]
quanzhou_block = quanzhou_df[['block', 'rent']]

# 按板块分组
beijing_block_dict = dict(list(beijing_block.groupby('block')))
shanghai_block_dict = dict(list(shanghai_block.groupby('block')))
guangzhou_block_dict = dict(list(guangzhou_block.groupby('block')))
shenzhen_block_dict = dict(list(shenzhen_block.groupby('block')))
quanzhou_block_dict = dict(list(quanzhou_block.groupby('block')))
```

将五个城市的租金数据按板块进行分组。

```
# 计算每个板块的租金均价，存在dict里
beijing_block_mean = {}
for key in beijing_block_dict.keys():
    beijing_block_mean[key] = round(beijing_block_dict[key]['rent'].mean(), 2)
shanghai_block_mean = {}
for key in shanghai_block_dict.keys():
    shanghai_block_mean[key] = round(shanghai_block_dict[key]['rent'].mean(), 2)
guangzhou_block_mean = {}
for key in guangzhou_block_dict.keys():
    guangzhou_block_mean[key] = round(guangzhou_block_dict[key]['rent'].mean(), 2)
shenzhen_block_mean = {}
for key in shenzhen_block_dict.keys():
    shenzhen_block_mean[key] = round(shenzhen_block_dict[key]['rent'].mean(), 2)
quanzhou_block_mean = {}
for key in quanzhou_block_dict.keys():
    quanzhou_block_mean[key] = round(quanzhou_block_dict[key]['rent'].mean(), 2)

# 按租金均价从大到小排序
beijing_block_mean = dict(sorted(beijing_block_mean.items(), key=lambda item: item[1], reverse=True))
shanghai_block_mean = dict(sorted(shanghai_block_mean.items(), key=lambda item: item[1], reverse=True))
guangzhou_block_mean = dict(sorted(guangzhou_block_mean.items(), key=lambda item: item[1], reverse=True))
shenzhen_block_mean = dict(sorted(shenzhen_block_mean.items(), key=lambda item: item[1], reverse=True))
quanzhou_block_mean = dict(sorted(quanzhou_block_mean.items(), key=lambda item: item[1], reverse=True))
```

计算每个板块的租金均价，存在字典里。然后，按租金均价，从大到小，给字典排序。

```
# 输出csv文件
beijing_data = {'板块': beijing_block_mean.keys(), '租金均价(元)': beijing_block_mean.values()}
beijing_result_df = pd.DataFrame(beijing_data)
beijing_result_df.to_csv('fig_q4\\北京每个板块租金均价.csv', index=False)
shanghai_data = {'板块': shanghai_block_mean.keys(), '租金均价(元)': shanghai_block_mean.values()}
shanghai_result_df = pd.DataFrame(shanghai_data)
shanghai_result_df.to_csv('fig_q4\\上海每个板块租金均价.csv', index=False)
guangzhou_data = {'板块': guangzhou_block_mean.keys(), '租金均价(元)': guangzhou_block_mean.values()}
guangzhou_result_df = pd.DataFrame(guangzhou_data)
guangzhou_result_df.to_csv('fig_q4\\广州每个板块租金均价.csv', index=False)
shenzhen_data = {'板块': shenzhen_block_mean.keys(), '租金均价(元)': shenzhen_block_mean.values()}
shenzhen_result_df = pd.DataFrame(shenzhen_data)
shenzhen_result_df.to_csv('fig_q4\\深圳每个板块租金均价.csv', index=False)
quanzhou_data = {'板块': quanzhou_block_mean.keys(), '租金均价(元)': quanzhou_block_mean.values()}
quanzhou_result_df = pd.DataFrame(quanzhou_data)
quanzhou_result_df.to_csv('fig_q4\\泉州每个板块租金均价.csv', index=False)
```

将五个字典输出至 csv 文件。

1	板块	租金均价(元)
2	西山	24743.08
3	中央别墅区	24269.21
4	魏公村	21630.27
5	燕莎	20975.95
6	官园	20631.83
7	东单	19781.82
8	东大桥	18782.48
9	三元桥	18027.06
10	西单	17585.51
11	万柳	17352.4

238	门头沟其它	2833.11
239	窦店	2703.59
240	古北口镇	2533.33
241	鼓楼街道	2403.66
242	城关	2397.14
243	密云其它	2172
244	南口	2116.67
245	燕山	1840
246	大兴机场空	1723.69
247	琉璃河	1400

北京西山板块租金均价最高，琉璃河板块最低。

1	板块	租金均价(元)
2	黄浦滨江	35958.18
3	徐汇滨江	23616.84
4	老西门	23531.81
5	新天地	23428.57
6	四川北路	22137.74
7	联洋	20797.58
8	豫园	18039.71
9	崇明其它	18000
10	陆家嘴	17867.74
11	新江湾城	17497.34

183	书院镇	2900
184	海湾	2719.86
185	大团镇	2500
186	金泽	2500
187	白鹤	2350
188	万祥镇	2327.27
189	太仓	2300
190	崇明新城	2285.71
191	车墩	2158
192	柘林	2157.14

上海黄浦滨江板块租金均价最高，柘林板块最低。

1	板块	租金均价(元)
2	黄埔村	45000
3	二沙岛	37800
4	汇景新城	15342.16
5	珠江新城东	12876.28
6	珠江新城中	11989.94
7	沙面	11826.32
8	亚运大道中	10251.03
9	厦滘	10144.44
10	人民北	9815.38
11	华南碧桂园	9136.41

233	神岗镇	1823.5
234	[体育中心]	1730
235	石滩镇	1583.16
236	福和镇	1578.26
237	旺城片区	1533.33
238	派潭镇	1518.75
239	钟落潭	1400
240	旧城区	1350
241	荔城富鹏	1034.09
242	小楼镇	1000

广州黄埔村板块租金均价最高，小楼镇板块最低。

1	板块	租金均价(元)
2	深圳湾	32386.8
3	香蜜湖	23089.11
4	曦城	21989.04
5	红树湾	19432.76
6	华侨城	15934.52
7	观澜	14432.2
8	蛇口	13217.66
9	福田中心	11265.9
10	科技园	10540.28
11	竹子林	10473.27

82	坪山	3924.23
83	松岗	3843.87
84	布吉水径	3813.7
85	清水河	3713.71
86	石岩	3631.99
87	布吉大芬	3552.17
88	丹竹头	3520.51
89	龙岗双龙	3513.41
90	坪地	3472.23
91	[宝安中心,	2190

深圳深圳湾板块租金均价最高，坪地板块最低。上图最低的数据同时处于宝安中心和西乡，不是常见数据，不考虑。

1	板块	租金均价(元)
2	西街	5728.57
3	泰禾广场	5209.58
4	城北路	3446.34
5	泉秀街	3306.86
6	西湖	2902.67
7	丰泽街	2855.22
8	万达	2851.9
9	涂门街	2848.1
10	安溪县	2722.35
11	东街	2572.04

35	磁灶镇	1749
36	南安中心	1718.69
37	惠安县	1700
38	泉港区	1696.58
39	南环路	1678.28
40	紫帽镇	1638.18
41	城西路	1617.65
42	晋江周边	1615.08
43	石狮周边	1339.4
44	南安东	1116.67

泉州西街板块租金均价最高，南安东最低。

比较各个城市不同朝向的单位面积租金分布情况

```
# 只需要朝向和单位面积租金的数据
beijing_direction = beijing_df[['direction', 'unit_rent']]
shanghai_direction = shanghai_df[['direction', 'unit_rent']]
guangzhou_direction = guangzhou_df[['direction', 'unit_rent']]
shenzhen_direction = shenzhen_df[['direction', 'unit_rent']]
quanzhou_direction = quanzhou_df[['direction', 'unit_rent']]

# 一条数据可能有多个朝向，改为多条只有一个朝向的数据
beijing_direction_new = pd.DataFrame(columns=['direction', 'unit_rent'])
for index, row in beijing_direction.iterrows():
    dir_list = row['direction'].split()
    for i in range(len(dir_list)):
        row['direction'] = dir_list[i]
        beijing_direction_new = beijing_direction_new.append(row)
shanghai_direction_new = pd.DataFrame(columns=['direction', 'unit_rent'])
for index, row in shanghai_direction.iterrows():
    dir_list = row['direction'].split()
    for i in range(len(dir_list)):
        row['direction'] = dir_list[i]
        shanghai_direction_new = shanghai_direction_new.append(row)
guangzhou_direction_new = pd.DataFrame(columns=['direction', 'unit_rent'])
for index, row in guangzhou_direction.iterrows():
    dir_list = row['direction'].split()
    for i in range(len(dir_list)):
        row['direction'] = dir_list[i]
        guangzhou_direction_new = guangzhou_direction_new.append(row)
shenzhen_direction_new = pd.DataFrame(columns=['direction', 'unit_rent'])
for index, row in shenzhen_direction.iterrows():
    dir_list = row['direction'].split()
    for i in range(len(dir_list)):
        row['direction'] = dir_list[i]
        shenzhen_direction_new = shenzhen_direction_new.append(row)
quanzhou_direction_new = pd.DataFrame(columns=['direction', 'unit_rent'])
for index, row in quanzhou_direction.iterrows():
    dir_list = row['direction'].split()
    for i in range(len(dir_list)):
        row['direction'] = dir_list[i]
        quanzhou_direction_new = quanzhou_direction_new.append(row)
```

一条数据可能有多个朝向，改为多条只有一个朝向的数据。

3000	怀柔其它	82.99	南 北	2	36.15

例如这条数据改为一条朝向为南和一条为北的数据，其他项不变。

```
# 按朝向进行分组
beijing_direction_dict = dict(list(beijing_direction_new.groupby('direction')))
shanghai_direction_dict = dict(list(shanghai_direction_new.groupby('direction')))
guangzhou_direction_dict = dict(list(guangzhou_direction_new.groupby('direction')))
shenzhen_direction_dict = dict(list(shenzhen_direction_new.groupby('direction')))
quanzhou_direction_dict = dict(list(quanzhou_direction_new.groupby('direction')))

# 计算每个朝向的单位面积租金均价
beijing_direction_mean = {}
for key in beijing_direction_dict.keys():
    beijing_direction_mean[key] = round(beijing_direction_dict[key]['unit_rent'].mean(), 2)
shanghai_direction_mean = {}
for key in shanghai_direction_dict.keys():
    shanghai_direction_mean[key] = round(shanghai_direction_dict[key]['unit_rent'].mean(), 2)
guangzhou_direction_mean = {}
for key in guangzhou_direction_dict.keys():
    guangzhou_direction_mean[key] = round(guangzhou_direction_dict[key]['unit_rent'].mean(), 2)
shenzhen_direction_mean = {}
for key in shenzhen_direction_dict.keys():
    shenzhen_direction_mean[key] = round(shenzhen_direction_dict[key]['unit_rent'].mean(), 2)
quanzhou_direction_mean = {}
for key in quanzhou_direction_dict.keys():
    quanzhou_direction_mean[key] = round(quanzhou_direction_dict[key]['unit_rent'].mean(), 2)

# 按单位面积租金均价从小到大排序
beijing_direction_mean = dict(sorted(beijing_direction_mean.items(), key=lambda item: item[1]))
shanghai_direction_mean = dict(sorted(shanghai_direction_mean.items(), key=lambda item: item[1]))
guangzhou_direction_mean = dict(sorted(guangzhou_direction_mean.items(), key=lambda item: item[1]))
shenzhen_direction_mean = dict(sorted(shenzhen_direction_mean.items(), key=lambda item: item[1]))
quanzhou_direction_mean = dict(sorted(quanzhou_direction_mean.items(), key=lambda item: item[1]))
```

按照朝向进行分组，然后计算东、西、南、北、东北、东南、西北和西南八个方向的单位面积租金均价，保存在字典里，最后按值从小到大排序字典。

朝向(北京)	租金均价(元)(北京)	朝向(上海)	租金均价(元)(上海)	朝向(广州)	租金均价(元)(广州)	朝向(深圳)	租金均价(元)(深圳)	朝向(泉州)	租金均价(元)(泉州)
北	92.76	北	95.98	东北	49.65	东南	91.45	北	25.19
南	94.63	南	101.86	南	51.2	西北	96.95	南	25.56
西南	103.1	东南	125.32	东南	52.41	南	97.33	东南	26.45
东南	103.49	东	126.97	北	55.67	西南	98.2	西南	27.85
东北	104.23	西	128.32	西南	56.57	东北	100.21	东	30.09
西	106.81	西南	132.32	西北	59.16	北	107.52	西北	30.66
东	108.71	西北	133.35	东	67.17	东	116.83	东北	32.94
西北	110.64	东北	140.02	西	75.6	西	124.77	西	33.62

将字典输出至一个 csv 文件，内容如上。在五个城市的房子种，东和西朝向的单位租金均价都是偏高的，南和北朝向都是偏低。

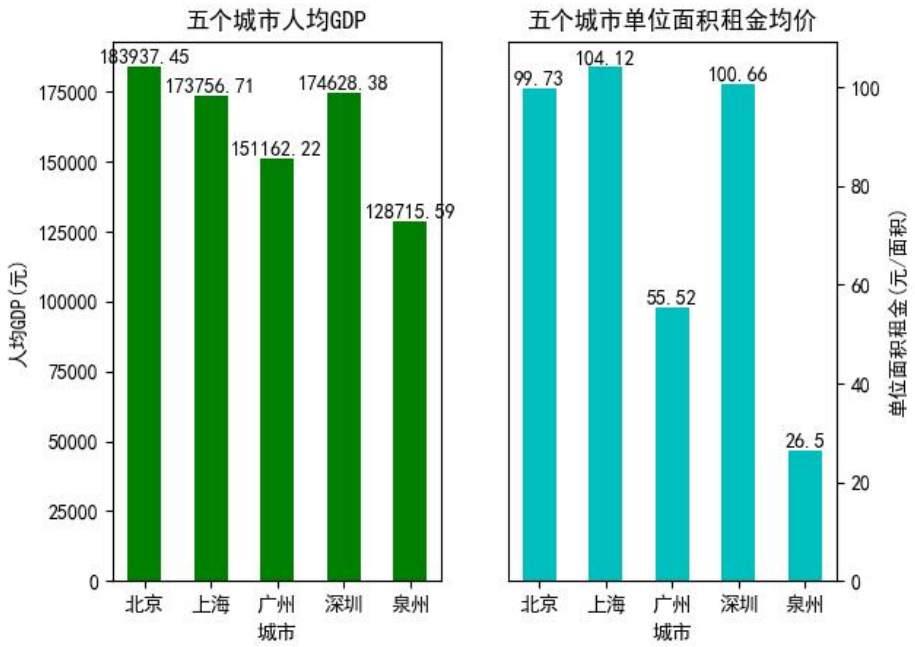
查询各个城市人均 GDP，分析并展示其和单位面积租金分布的关系

```
# 2021年人均GDP
beijing_gdp = 183937.45
shanghai_gdp = 173756.71
guangzhou_gdp = 151162.22
shenzhen_gdp = 174628.38
quanzhou_gdp = 128715.59
```

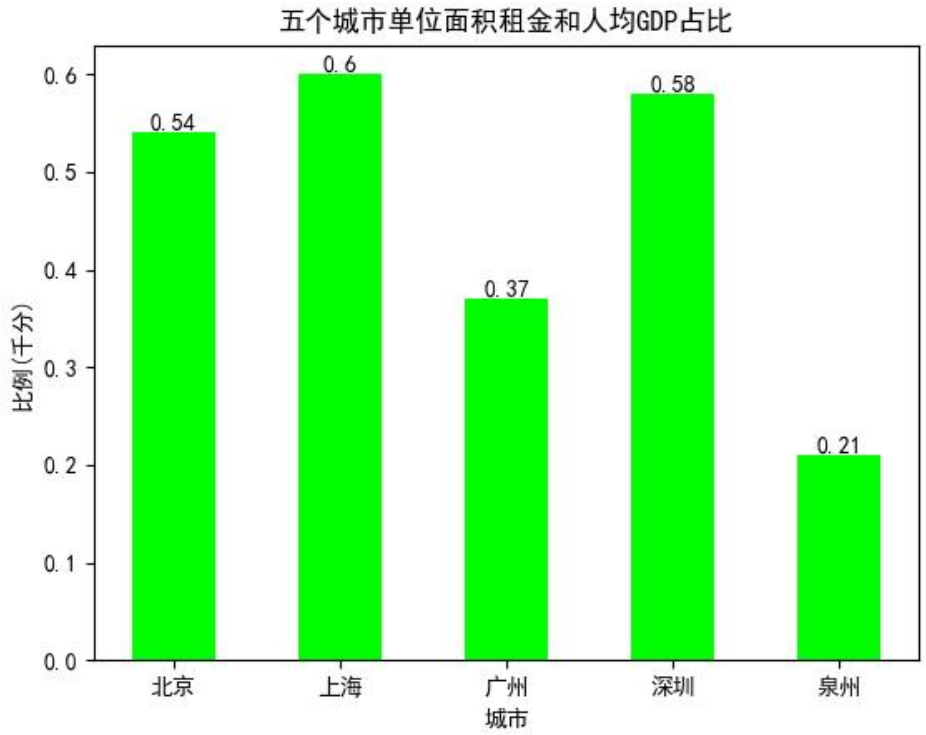
五个城市人均 GDP 分别如上。

```
# 画图，输出单位面积租金/人均GDP比例
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
city = ['北京', '上海', '广州', '深圳', '泉州']
ratio = [round(beijing_mean / beijing_gdp * 1000, 2), round(shanghai_mean / shanghai_gdp * 1000, 2),
         round(guangzhou_mean / guangzhou_gdp * 1000, 2), round(shenzhen_mean / shenzhen_gdp * 1000, 2),
         round(quanzhou_mean / quanzhou_gdp * 1000, 2)]
x = [1, 2, 3, 4, 5]
plt.figure()
plt.bar(x, ratio, 0.5, color='lime', tick_label=city)
for a, b in zip(x, ratio):
    plt.text(a, b, b, ha='center', va='bottom')
plt.title('五个城市单位面积租金和人均GDP占比')
plt.xlabel('城市')
plt.ylabel('比例(千分)')
plt.savefig('fig_q6\\五个城市单位面积租金和人均GDP占比.png')
```

计算单位面积租金均价/人均 GDP 的比例，然后画图。



北京的人均 GDP 最高，深圳第二，上海第三，而上海的单位面积租金均价最高，深圳第二，北京第三。广州和泉州的人均 GDP 和单位面积租金均价都是第四第五。



根据上图，上海的比例最高，深圳第二，北京第三，广州第四，泉州最低。因此，在泉州租房的性价比最高，其次是广州。

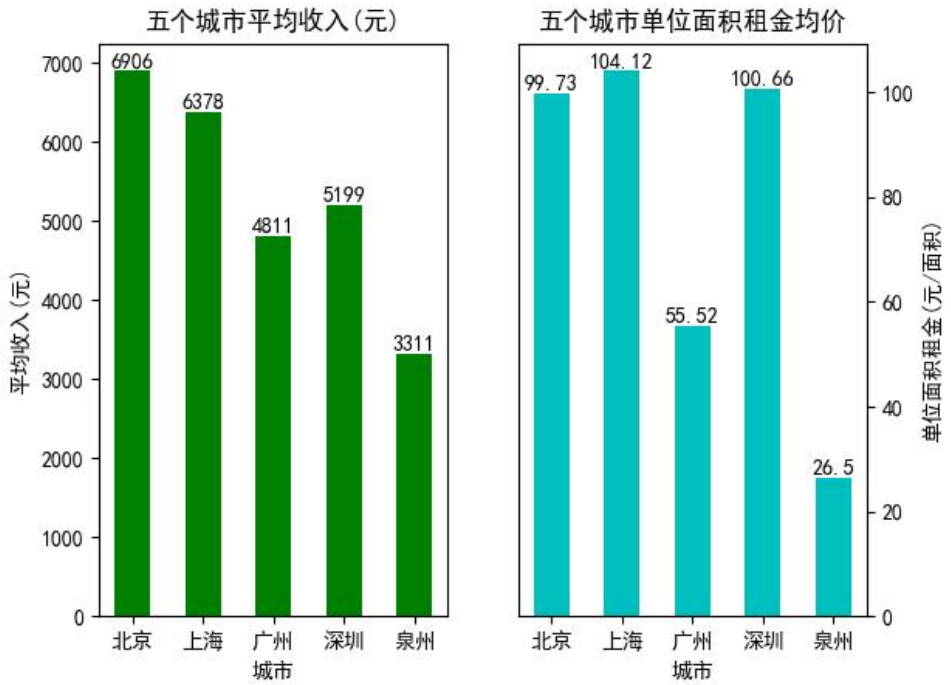
查询各个城市的平均工资，分析并展示其和单位面积租金分布的关系

```
# 五个城市人均收入
beijing_income = 6906
shanghai_income = 6378
guangzhou_income = 4811
shenzhen_income = 5199
quanzhou_income = 3311
```

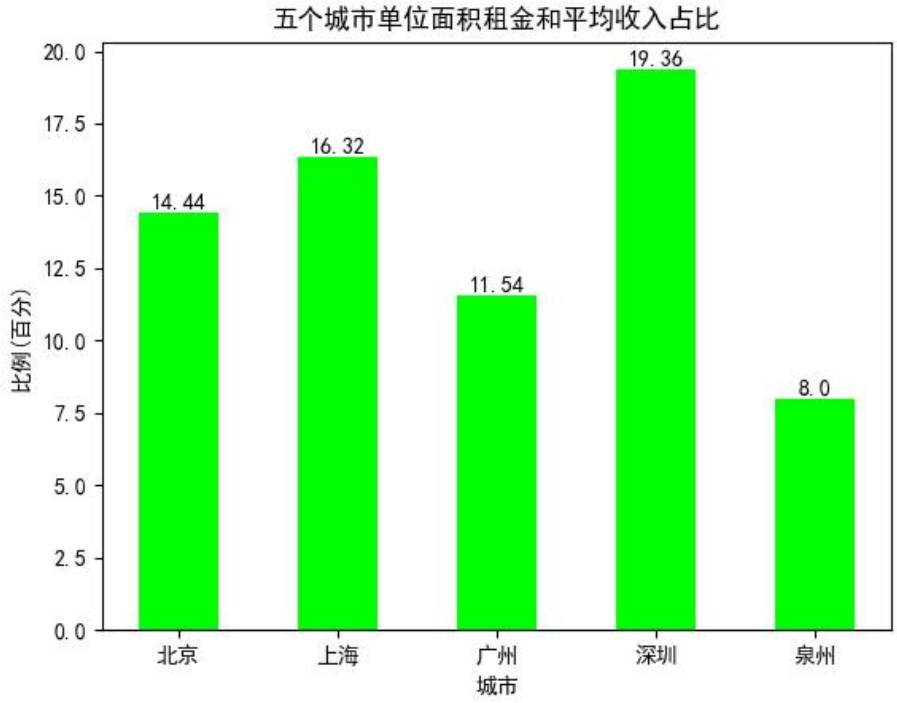
五个城市的平均收入如上。

```
# 画图，输出单位面积租金/人均收入比例
ratio = [round(beijing_mean / beijing_income * 1000, 2), round(shanghai_mean / shanghai_income * 1000, 2),
         round(guangzhou_mean / guangzhou_income * 1000, 2), round(shenzhen_mean / shenzhen_income * 1000, 2),
         round(quanzhou_mean / quanzhou_income * 1000, 2)]
plt.figure()
plt.bar(x, ratio, 0.5, color='lime', tick_label=city)
for a, b in zip(x, ratio):
    plt.text(a, b, b, ha='center', va='bottom')
plt.title('五个城市单位面积租金和平均收入占比')
plt.xlabel('城市')
plt.ylabel('比例(百分)')
plt.savefig('fig_q6\\五个城市单位面积租金和平均收入占比.png')
```

计算单位面积租金均价/平均收入的比例，然后画图。



北京的平均收入最高，上海第二，深圳第三，而上海的单位面积租金均价最高，深圳第二，北京第三。广州和泉州的平均收入和单位面积租金均价都是第四第五。



根据上图，深圳的比例最高，上海第二，北京第三，广州第四，泉州最低。因此，在深圳租房的负担最高，其次是上海。