# EXAMPLES

Python + pandas
(+ matplotlib + numpy)

# THE DATA

```
import pandas as pd
listings = pd.read_csv('Edinburgh_AirBnb_listings.csv')
reviews = pd.read_csv('Edinburgh_reviews_reduced.csv’)

listings.head()
```

| | listing_id | name | host_id | host_name | neighbourhood | latitude | longitude | room_type | price | minimum_nights | number_ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 15420 | Georgian Boutique Apt City Centre | 60423 | Charlotte | Old Town, Princes Street and Leith Street | 55.956892 | -3.187677 | Entire home/apt | 80 | 3 | |
| **1** | 24288 | Cool central Loft, sleeps 4, 2 double bed+en-s... | 46498 | Gordon | Meadows and Southside | 55.942646 | -3.184670 | Entire home/apt | 115 | 2 | |

# THE DATA

| | listing_id | name | host_id | host_name | neighbourhood | latitude | longitude | room_type | price | minimum_nights | number_ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 15420 | Georgian Boutique Apt City Centre | 60423 | Charlotte | Old Town, Princes Street and Leith Street | 55.956892 | -3.187677 | Entire home/apt | 80 | 3 | |
| **1** | 24288 | Cool central Loft, sleeps 4, 2 double bed+en-s... | 46498 | Gordon | Meadows and Southside | 55.942646 | -3.184670 | Entire home/apt | 115 | 2 | |

```
>> listings.shape

(11985, 15)
```

count

# QUESTIONS

## 1. How many listings are there for each room type?

```
>> listings['room_type'].value_counts()

Entire home/apt 7366
Private room 4582
Shared room 37
Name: room_type, dtype: int64
```

# QUESTIONS

## 2. What is the average price for Listings in each neighbourhood?

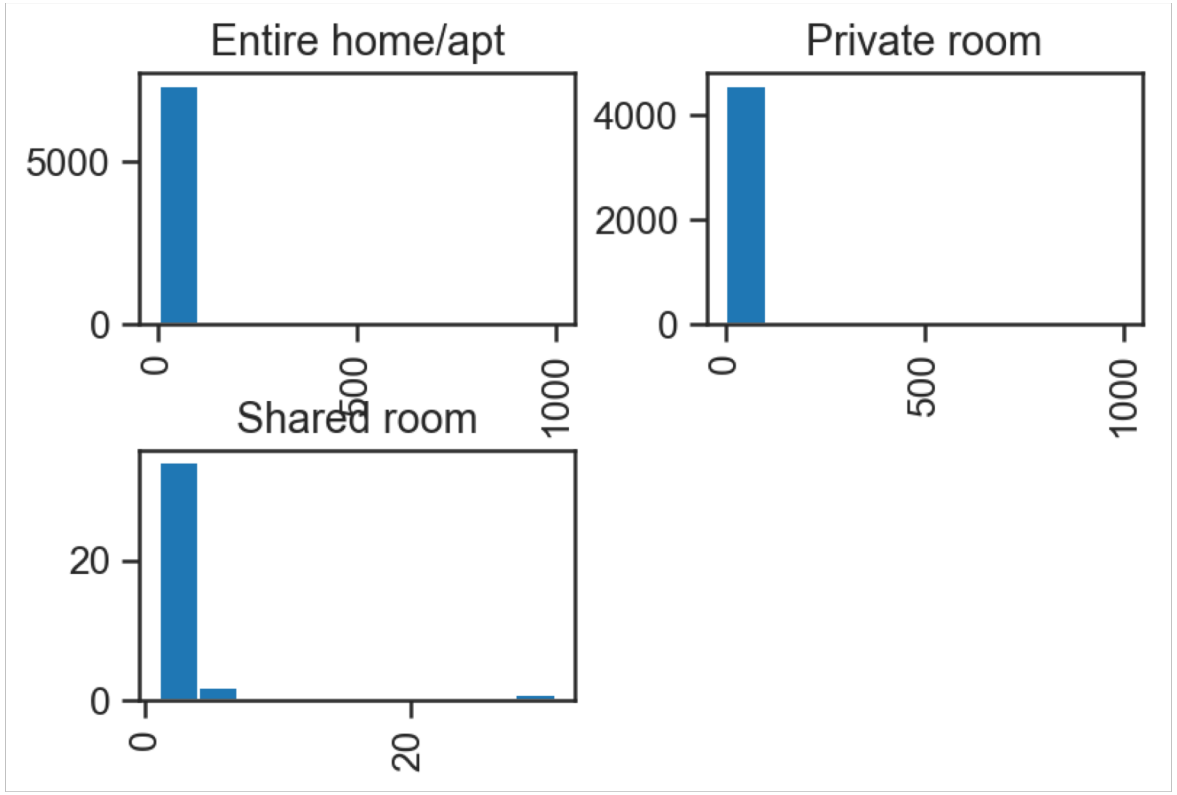
```
>> listings[['price', 'neighbourhood']]
       .groupby(by='neighbourhood')
       .mean()
       .head(12)
```

| neighbourhood | price |
|---|---|
| Abbeyhill | 84.635870 |
| Baberton and Juniper Green | 81.928571 |
| Balerno and Bonnington Village | 68.866667 |
| Balgreen and Roseburn | 70.413333 |
| Barnton, Cammo and Cramond South | 106.750000 |
| Bingham, Magdalene and The Christians | 56.000000 |
| Blackford, West Mains and Mayfield Road | 74.518182 |
| Blackhall | 76.363636 |
| Bonaly and The Pentlands | 79.777778 |
| Bonnington | 76.449438 |
| Boswall and Pilton | 70.777778 |
| Broomhouse and Bankhead | 436.666667 |

# QUESTIONS

## 3. What is the distribution of minimum nights by room type?

```
>> listings[['minimum_nights', 'room_type']]
       .hist(by=listings['room_type'], bins=10)
```




count

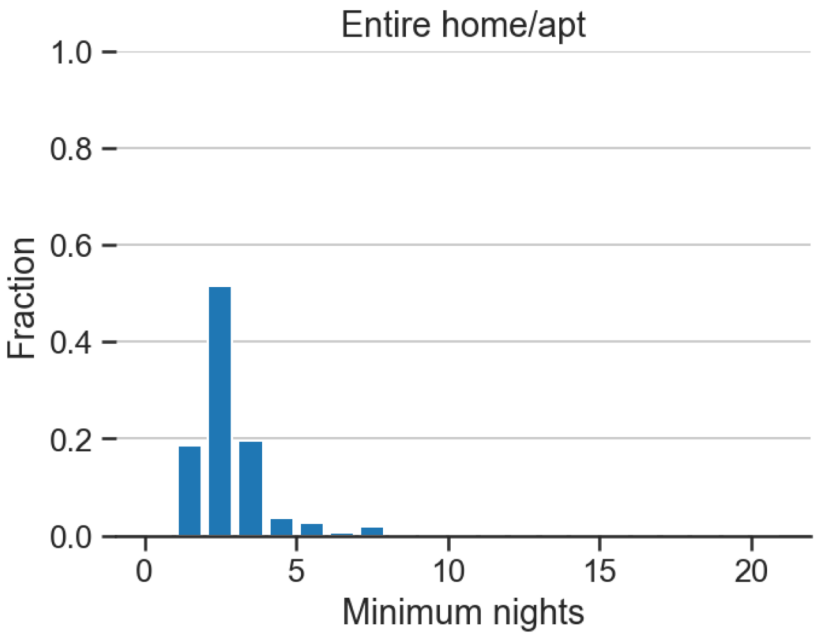
# QUESTIONS

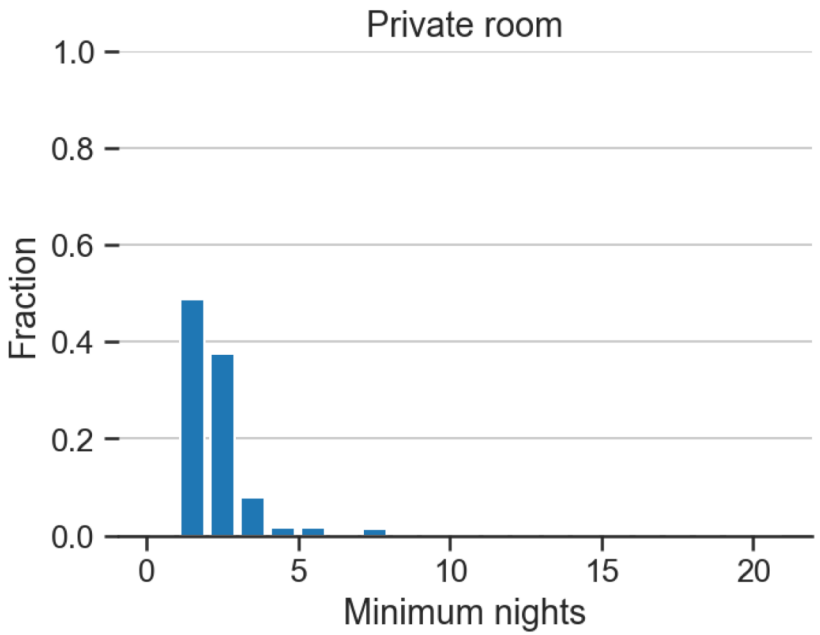
## 3. What is the distribution of minimum nights by room type?

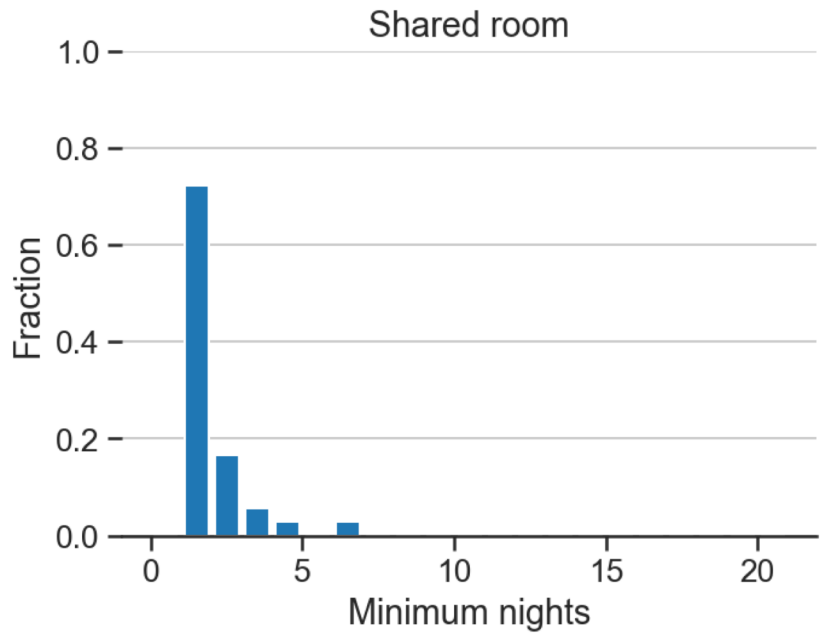
```
import numpy as np
import matplotlib.pyplot as plt

bins = np.linspace(0, 21, 22)
f, ax = plt.subplots(1,3,figsize=(15,4))

room_types = pd.unique(listings['room_type'])
for i, room_type in enumerate(room_types):
    rows = listings['room_type'] == room_type
    min_nights = listings['minimum_nights'][rows].values
    counts, bin_edges = np.histogram(min_nights, bins)
    center = (bins[:-1] + bins[1:]) / 2
    plt.sca(ax[i])
    plt.bar(x = center, height = counts)
```

# QUESTIONS

## 3. What is the distribution of minimum nights by room type?




count

# QUESTIONS

## 4. How does the mean price of listings vary with the number of listings for each host?

```
columns = ['price', 'host_id']
av_price = listings[columns].groupby(by='host_id').mean()
num_list = listings[columns].groupby(by='host_id').count()
mean_and_count = pd.merge(
    left=av_price,
    right=num_list,
    left_on='host_id',
    right_on='host_id',
    suffixes=('_mean', '_count')
)
mean_and_count.head()
```

# QUESTIONS

4. How does the mean price of listings vary with the number of listings for each host?

```
mean_and_count = pd.merge(
    left=av_price,
    right=num_list,
    left_on='host_id',
    right_on='host_id',
    suffixes=('_mean', '_count')
)
```

| host_id | price |
|---|---|
| 33078 | 60.000000 |
| 36298 | 43.500000 |
| 46498 | 87.500000 |
| 47584 | 56.333333 |
| 60423 | 80.000000 |

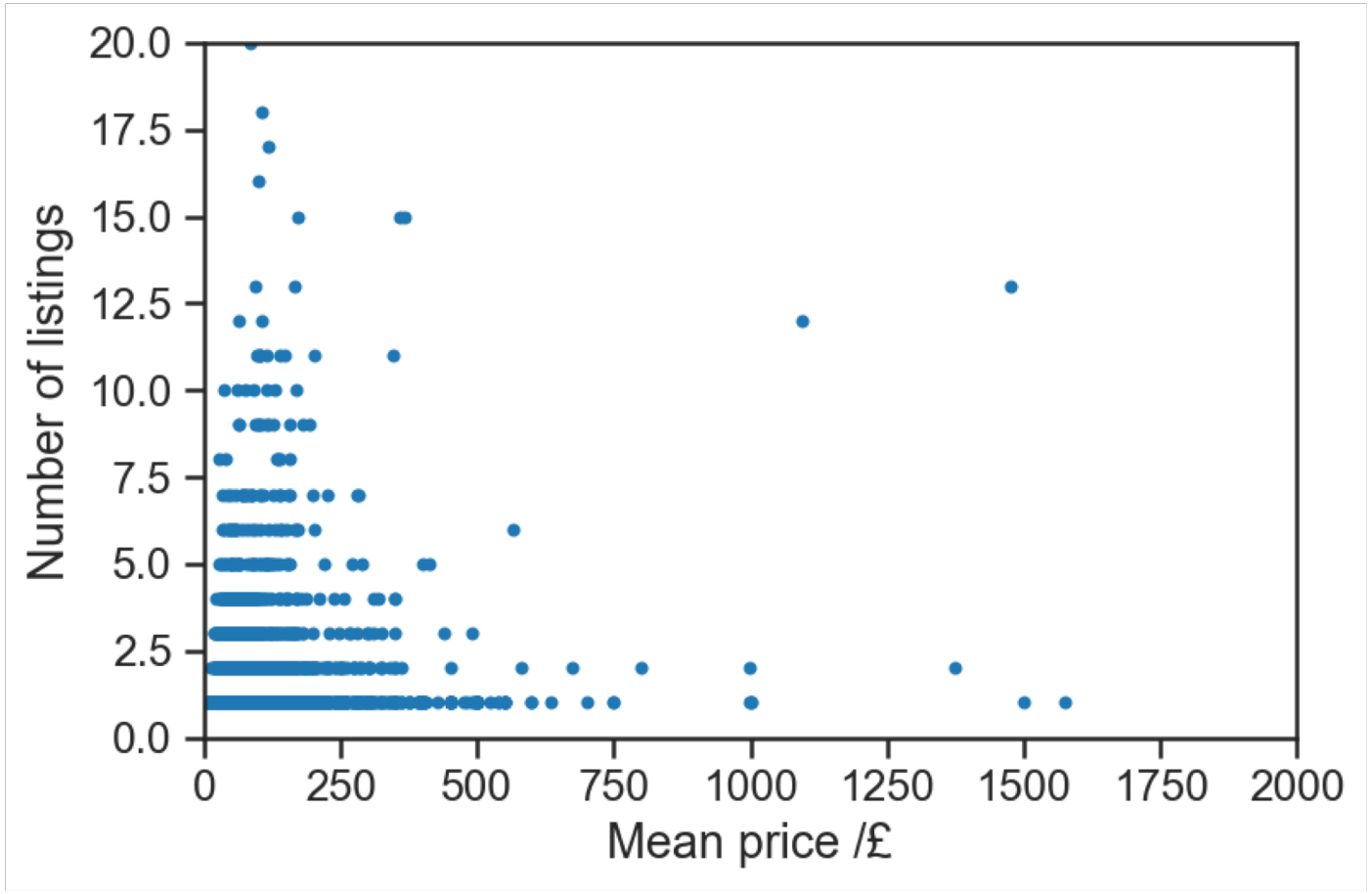
| host_id | price |
|---|---|
| 33078 | 1 |
| 36298 | 2 |
| 46498 | 2 |
| 47584 | 3 |
| 60423 | 1 |

| host_id | price_mean | price_count |
|---|---|---|
| 33078 | 60.000000 | 1 |
| 36298 | 43.500000 | 2 |
| 46498 | 87.500000 | 2 |
| 47584 | 56.333333 | 3 |
| 60423 | 80.000000 | 1 |

# QUESTIONS

## 4. How does the mean price of listings vary with the number of listings for each host?

```
plt.plot(
    mean_and_count['price_mean'],
    mean_and_count['price_count'],
    '.'
)
plt.xlabel('Mean price /£')
plt.ylabel('Number of listings')
plt.xlim([0, 2000])
plt.ylim([0, 20])
plt.tight_layout()
```




count

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts = listings['host_id']
       .value_counts().reset_index()
       .rename(columns={'index': 'host_id', 'host_id': 'number'})
       .sort_values('number')

l_counts_sorted = l_counts['number']
       .value_counts()
       .reset_index()
       .rename(columns={'index': 'number', 'number': 'number_count'})
       .sort_values(by='number').reset_index().drop('index', axis=1)
```

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts = listings['host_id']
       .value_counts()
```

```
646220 98
37563463 84
134201160 72
50999670 50
19173145 37
Name: host_id, dtype: int64
```

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts = listings['host_id']
        .value_counts().reset_index()
```

```
646220 98
37563463 84
134201160 72
50999670 50
19173145 37
Name: host_id, dtype: int64
```



| | index | host_id |
|---|---|---|
| 0 | 646220 | 98 |
| 1 | 37563463 | 84 |
| 2 | 134201160 | 72 |
| 3 | 50999670 | 50 |
| 4 | 19173145 | 37 |

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts = listings['host_id’]
       .value_counts().reset_index()
       .rename(columns={'index': 'host_id', 'host_id': 'number’})
```

| | index | host_id |
|---|---|---|
| 0 | 646220 | 98 |
| 1 | 37563463 | 84 |
| 2 | 134201160 | 72 |
| 3 | 50999670 | 50 |
| 4 | 19173145 | 37 |



| | host_id | number |
|---|---|---|
| 0 | 646220 | 98 |
| 1 | 37563463 | 84 |
| 2 | 134201160 | 72 |
| 3 | 50999670 | 50 |
| 4 | 19173145 | 37 |

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts = listings['host_id’]
        .value_counts().reset_index()
        .rename(columns={'index': 'host_id', 'host_id': 'number’})
        .sort_values('number’)
```

| | host_id | number |
|---|---|---|
| **0** | 646220 | 98 |
| **1** | 37563463 | 84 |
| **2** | 134201160 | 72 |
| **3** | 50999670 | 50 |
| **4** | 19173145 | 37 |

| | host_id | number |
|---|---|---|
| **4336** | 45953775 | 1 |
| **5790** | 4523247 | 1 |
| **5789** | 126571757 | 1 |
| **5788** | 6874347 | 1 |
| **5787** | 133090661 | 1 |

# QUESTIONS

## 5. What is the cumulative percentage of the number of listings per host, ranked from most to fewest?

```
l_counts_sorted['cumulative'] =
        pd.Series.cumsum(l_counts_sorted['number_count'])
```

| | number | number_count | cumulative |
|---|---|---|---|
| **0** | 1 | 7142 | 0.823380 |
| **1** | 2 | 1015 | 0.940397 |
| **2** | 3 | 252 | 0.969449 |
| **3** | 4 | 111 | 0.982246 |
| **4** | 5 | 45 | 0.987434 |
| **5** | 6 | 25 | 0.990316 |
| **6** | 7 | 24 | 0.993083 |
| **7** | 8 | 6 | 0.993774 |
| **8** | 9 | 12 | 0.995158 |
| **9** | 10 | 7 | 0.995965 |

# QUESTIONS

## 6. For listings with at least one review, what is the average number of reviews per listing, grouped by price?

```
num_reviews_per_listing = reviews['listing_id']
        .value_counts()
        .reset_index()
        .rename(columns={'index': 'listing_id', 'listing_id': 'num_reviews'})

merged = pd.merge(
        left=listings,
        right=num_reviews_per_listing,
        left_on='listing_id',
        right_on='listing_id'
)
```

| _listings_count | availability_365 | num_reviews |
|---|---|---|
| 1 | 268 | 246 |
| 2 | 66 | 172 |

count

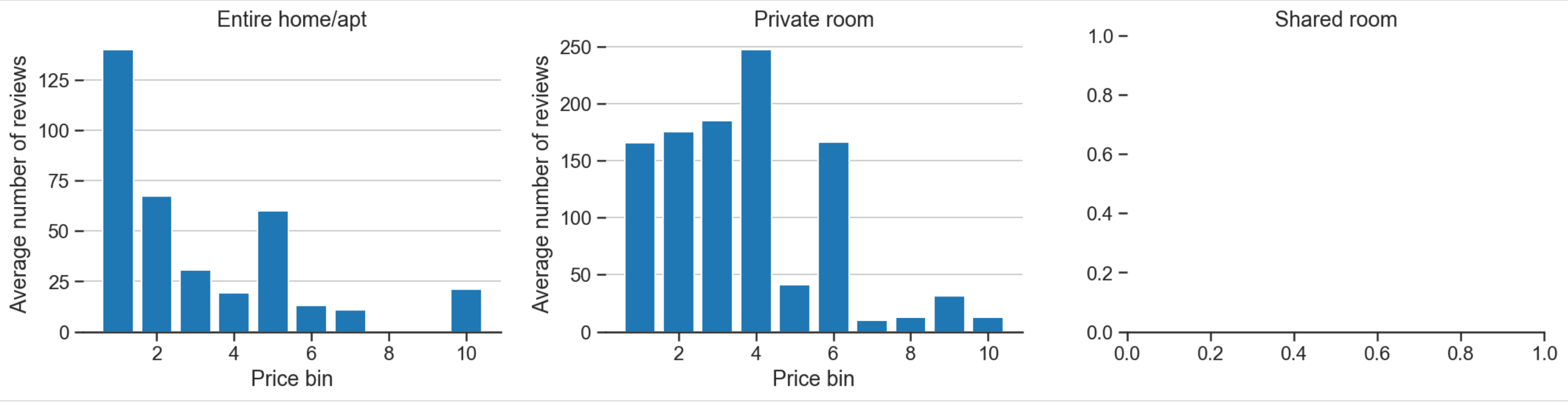
# QUESTIONS

```
room_types = pd.unique(listings['room_type'])

for i, room_type in enumerate(room_types):
    rows = merged['room_type'] == room_type
    filtered = merged[rows]
    if filtered.shape[0] > 0:
        bins = np.linspace(
                np.min(filtered['price’]),
                np.max(filtered['price’]),
                11
        )
        filtered['price_bin'] = pd.cut(
                filtered['price'].values,
                bins,
                labels=[i for i in range(10)]
        )
        num_reviews = filtered[['num_reviews', 'price_bin’]]
                .groupby('price_bin’)
                .mean()
        plt.bar(x = range(1,11), height = num_reviews['num_reviews'].values)
```

| | price | price_bin |
|---|---|---|
| **4** | 32 | 1.0 |
| **6** | 100 | 8.0 |
| **7** | 50 | 3.0 |
| **9** | 60 | 4.0 |
| **16** | 30 | 1.0 |
| **17** | 55 | 3.0 |

# QUESTIONS

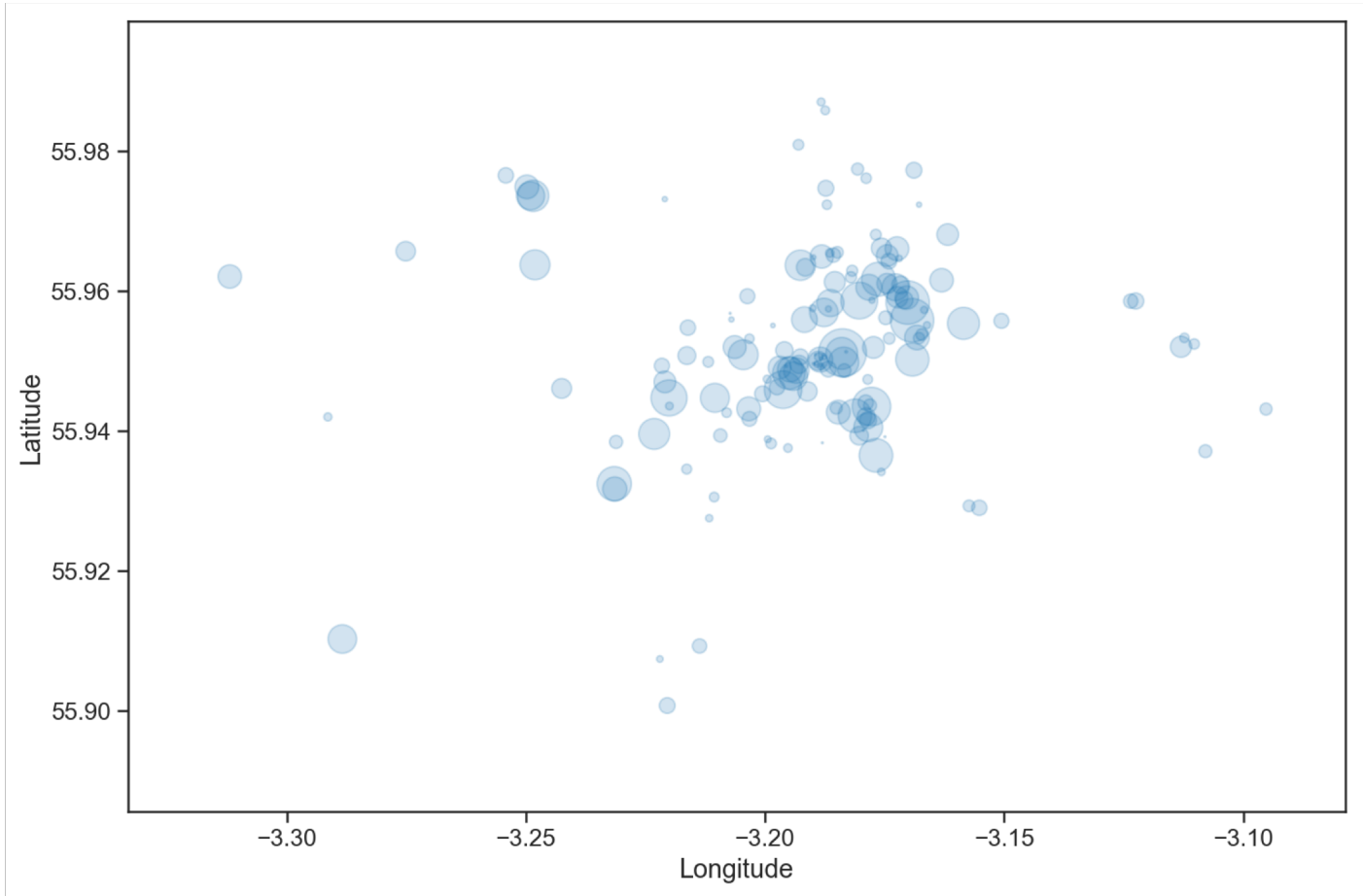
## 6. For listings with at least one review, what is the average number of reviews per listing, grouped by price?



# QUESTIONS

## 7. For listings with at least one review, plot the number of reviews per listing by the listing's longitude and latitude

```
plt.scatter(
        merged['longitude'],
        merged['latitude'],
        s=merged['num_reviews'],
        alpha=0.2
)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```




count

# QUESTIONS

## 8. What is the average length of review by property type?

```
merged['length'] = merged.apply(lambda row: len(row.comments), axis=1)
```

*or merged['length'] = [len(el) for el in merged['comments']]*

```
merged[['length', 'room_type']].groupby(by='room_type').mean()
```

| | length |
|---|---|
| room_type | |
| Entire home/apt | 333.369630 |
| Private room | 309.463877 |

count

# QUESTIONS

## 9. For each host, what is their year-on-year percentage change in number of reviews since 2010?

```
host_dates = pd.merge(
        left=reviews,
        right=listings,
        left_on='listing_id',
        right_on='listing_id',
        how='left'
)[['host_id', 'date']]

host_dates['year'] = host_dates.apply(
        lambda row: int(row['date'][-4:]),
        axis=1
)
```

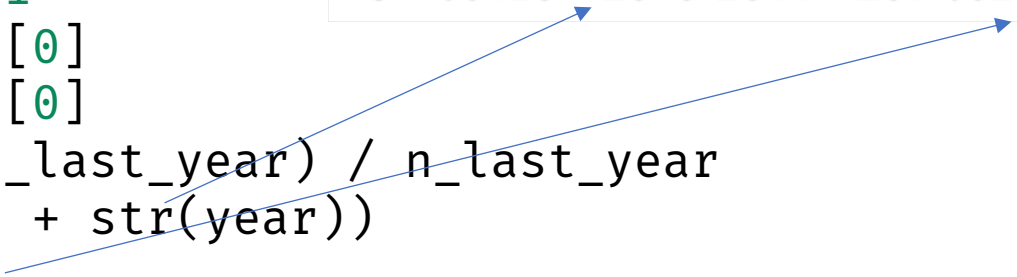
| | host_id | date | year |
|---|---|---|---|
| 0 | 60423 | 18/01/2011 | 2011 |
| 1 | 60423 | 31/01/2011 | 2011 |
| 2 | 60423 | 19/04/2011 | 2011 |
| 3 | 60423 | 23/04/2011 | 2011 |
| 4 | 60423 | 15/05/2011 | 2011 |

# QUESTIONS

## 9. For each host, what is their year-on-year percentage change in number of reviews since 2010?

```
hosts = pd.unique(host_dates['host_id'])
change = { 'host': [], 'year': [], 'change': []}
for host in hosts:
    rows = host_dates['host_id'] == host
    filtered = host_dates[rows]
    for year in range(2011, 2019):
        this_year = filtered['year'] == year
        last_year = filtered['year'] == year - 1
        n_this_year = filtered[this_year].shape[0]
        n_last_year = filtered[last_year].shape[0]
        percent_change = 100 * (n_this_year - n_last_year) / n_last_year
        change['year'].append(str(year-1) + '-' + str(year))
        change['change'].append(percent_change)
        changes = pd.DataFrame(data=change)
```

| | host | year | change |
|---|---|---|---|
| **0** | 60423 | 2010-2011 | NaN |
| **1** | 60423 | 2011-2012 | 40.000000 |
| **2** | 60423 | 2012-2013 | -7.142857 |
| **3** | 60423 | 2013-2014 | 207.692308 |



count

# QUESTIONS

## 10. What percentage of reviewers have only left reviews for one neighbourhood?

```
review_counts = reviews['reviewer_id’]
        .value_counts().reset_index()
        .rename(columns={'index': 'reviewer_id', 'reviewer_id': 'num_reviews'})
reviews_with_count = pd.merge(
        left=reviews, right=review_counts,
        left_on='reviewer_id’, right_on='reviewer_id’, how='left’
)
reviews_multiple_only = reviews_with_count[reviews_with_count['num_reviews'] > 1]
rmo_listings = pd.merge(
        left=reviews_multiple_only, right=listings,
        left_on='listing_id’, right_on='listing_id’, how='left’
)
n_unique = rmo_listings[['reviewer_id', 'neighbourhood’]]
        .groupby('reviewer_id')['neighbourhood’].nunique().reset_index()

fraction = 100 * n_unique[n_unique['neighbourhood'] == 1].shape[0] / n_unique.shape[0]
```

# QUESTIONS

## 10. What percentage of reviewers have only left reviews for one neighbourhood?

```
review_counts = reviews['reviewer_id']
        .value_counts().reset_index()
        .rename(columns={'index': 'reviewer_id', 'reviewer_id': 'num_reviews'})
reviews_with_count = pd.merge(
        left=reviews, right=review_counts,
        left_on='reviewer_id', right_on='reviewer_id', how='left'
)
reviews_multiple_only = reviews_with_count[reviews_with_count['num_reviews'] > 1]
```

| | listing_id | review_id | date | reviewer_id | reviewer_name | comments | num_reviews |
|---|---|---|---|---|---|---|---|
| **5** | 15420 | 273167 | 21/05/2011 | 183378 | Jonathan | Charlotte is an excellent and well-organised h... | 3 |
| **20** | 15420 | 2929665 | 20/11/2012 | 3879011 | James | Lovely host, we especially appreciated very fl... | 2 |
| **61** | 15420 | 19118942 | 07/09/2014 | 2026707 | Andrew | Charlotte provided us with all the details we ... | 2 |
| **91** | 15420 | 39158289 | 21/07/2015 | 31282313 | Igor | Upon our arrival Charlotte was very welcoming ... | 2 |

count

# QUESTIONS

## 10. What percentage of reviewers have only left reviews for one neighbourhood?

```
rmo_listings = pd.merge(
        left=reviews_multiple_only, right=listings,
        left_on='listing_id', right_on='listing_id', how='left'
)
n_unique = rmo_listings[['reviewer_id', 'neighbourhood']]
        .groupby('reviewer_id')['neighbourhood'].nunique().reset_index()
```

| | reviewer_id | neighbourhood |
|---|---|---|
| **0** | 50006 | 1 |
| **1** | 67107 | 2 |
| **2** | 138460 | 2 |
| **3** | 183378 | 2 |
| **4** | 211475 | 2 |

# QUESTIONS

## 11. What percentage of hosts own listings with review lengths in the top 20% of review lengths?

```
listing_top_20 = reviews[['listing_id', 'comments']]
listing_top_20['length'] = listing_top_20
      .apply(lambda row: len(row['comments']), axis=1)

listing_top_20 = listing_top_20
      .sort_values('length', ascending=False)[:int(0.2 * reviews.shape[0])]

top_20_with_hosts = pd.merge(
      left=listing_top_20, right=listings,
      left_on='listing_id', right_on='listing_id', how='left’)

n_hosts_top_20 = len(pd.unique(top_20_with_hosts['host_id']))
n_hosts = len(pd.unique(listings['host_id']))
percentage_top_20 = 100 * n_hosts_top_20 / n_hosts
```

# QUESTIONS

12. What's the average distance between listings for each neighbourhood?

```
neighs = pd.unique(listings['neighbourhood'])

data = { 'neighbourhood': [], 'dist': [] }

for n in neighs:
    filtered = listings[listings['neighbourhood'] == n]
    lat = filtered['latitude'].values
    lon = filtered['longitude'].values
    dist = []
    for i in range(len(lat)):
        for j in range(i):
            if i != j:
                d = np.sqrt(
                        (lat[i] - lat[j])**2 + (lon[i] - lon[j])**2
                )
                dist.append(d)
                data['neighbourhood'].append(n)
    data['dist'].append(np.mean(dist))

avDist = pd.DataFrame(data=data)
avDist.sort_values('dist').head(10)
```

# QUESTIONS

## 12. What's the average distance between listings for each neighbourhood?

| | neighbourhood | dist |
|---|---|---|
| **13** | Leith (Albert Street) | 0.003028 |
| **30** | Oxgangs | 0.003110 |
| **49** | Polwarth | 0.003199 |
| **29** | Marchmont West | 0.003289 |
| **56** | Canonmills and New Town North | 0.003430 |
| **38** | Abbeyhill | 0.003510 |
| **34** | Pilrig | 0.003597 |
| **11** | New Town East and Gayfield | 0.004068 |
| **24** | Easter Road and Hawkhill Avenue | 0.004158 |
| **88** | Restalrig (Loganlea) and Craigentinny West | 0.004160 |

# SUMMARY

## What was used:

```
head – show the top rows
shape – get the number of rows and columns
values – get the raw data from a column
[] – indexing with square brackets

value_counts() – count unique values
sort_values() – sort a dataframe by a column
reset_index() – re-introduce an index


groupby() – group by a column
nunique() – number of unique elements in a column
apply() – apply a function to a dataframe
rename() – rename columns

pd.unique() – get unique values in a column
pd.merge() – join two dataframes
pd.Series.cumsum() – cumulative sum of a column
pd.cut() – apply a binning function to a column
pd.DataFrame() – make a new dataframe
```