airbnb

Group Members:

- Linda Levy
- Raph Serrano
- Swobabika Jena
- Jason Sutton

Airbnb Investment Opportunities

- Money to invest
- Airbnb looks like easy money
- How do we decide where to establish our first listing

Hypothesis

- Proximity to the CBD will increase earnings
- Train proximity will increase earnings
- Neighbourhoods with the highest earnings are the most popular areas
- The popularity of property types will be reflected in higher earnings

Sourcing Data

Downloaded: 13/01/21

- **Inside Airbnb** (Insideairbnb.com)
  - .gz files – zipped large files in .gz format
  - .csv files
- **Department of Environment, Land, Water & Planning** (https://land.vic.gov.au/maps-and-spatial)
  - .csv file

Data from inside Airbnb:

- Extremely large datasets
- Not clearly defined
  - Multiple columns referred to location property
    - neighbourhood - neighbourhood cleansed – city - smart-location
  - Only one of these matched the coordinates stored against the Airbnb listing
- Quite messy
  - eg room type was free form - we had 1 ”castle” in Melbourne on the listing
- Typos in information
  - multiple spellings and configuration of suburb names)

Data from DELWP was:

- containing train stop name, stop id, longitude and latitude
- Quite large
  - this was be reduced once scope for the data from insideairbnb is finalised
- Clean, succinct, ready to be mined

	A	B	C	D	E	
1	STOP_ID	STOP_NAME	LATITUDE	LONGITUDE	TICKETZONE	ROUTEUSSP
2	19967	Anstey Railway Station (Brunswick)	-37.761242	144.960684	1	Upfield
3	19968	Brunswick Railway Station (Brunswick)	-37.767721	144.959587	1	Upfield
4	19969	Jewell Railway Station (Brunswick)	-37.774987	144.958717	1	Upfield
5	19970	Royal Park Railway Station (Parkville)	-37.781193	144.952301	1	Upfield
6	19971	Flemington Bridge Railway Station (North Melbourne)	-37.78814	144.939323	1	Upfield
7	19972	Macaulay Railway Station (North Melbourne)	-37.794267	144.936166	1	Upfield
8	19973	North Melbourne Railway Station (West Melbourne)	-37.807419	144.94257	1	Sunbury,Upfield,Werribee,Williamstown,Craigieburn
9	19974	Clifton Hill Railway Station (Clifton Hill)	-37.788657	144.995417	1	Mernda,Hurstbridge
10	19975	Victoria Park Railway Station (Abbotsford)	-37.799158	144.994451	1	Mernda,Hurstbridge

First Wave - Irrelevant records

CSV file read into Jupyter Notebook

Based on the data needs for our questions, the following records were not required and dropped from the data frame.

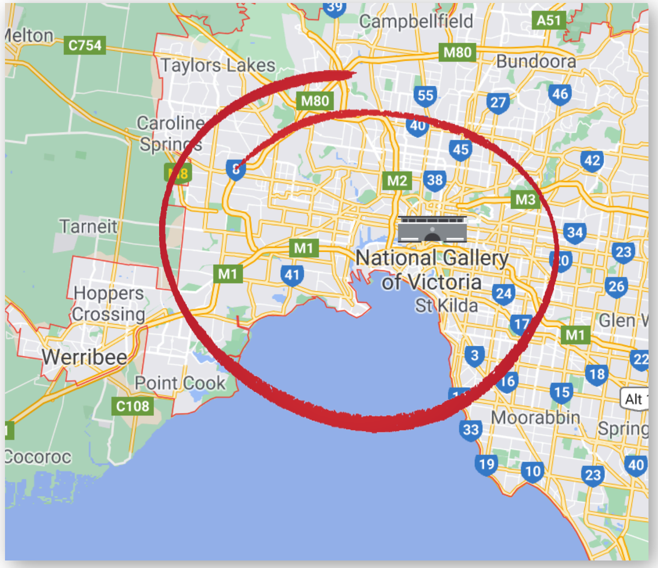
Room type:

- The question is around investing in an Airbnb property so we are only interested in entire homes
- Dropped private rooms / shared rooms / hotel rooms

Drop irrelevant columns

Neighbourhood:

- To focus on purely metropolitan properties we dropped listings to within 15km of CBD
- Achieved by using latitude and longitude in data to determine the distance from the CBD



Used loc property for Room type and Neighbourhood

Second Wave - Data clean of rubbish fields

Followed the *Ultimate guide to data cleaning* - towardsdatascience.com

- Checked for duplicates
  - none identified
- Check data info for null field
  - *using df.info()*
- Replace null value with relevant data
  - Fill string values with "missing” or relevant value
  - Filled integer values with 0
- Check the relevant columns dtypes
  - All columns presented as objects (strings)
- converted relevant columns to integers and floats
  - including stripping currency of its formatting

Second Wave - Data clean of rubbish fields

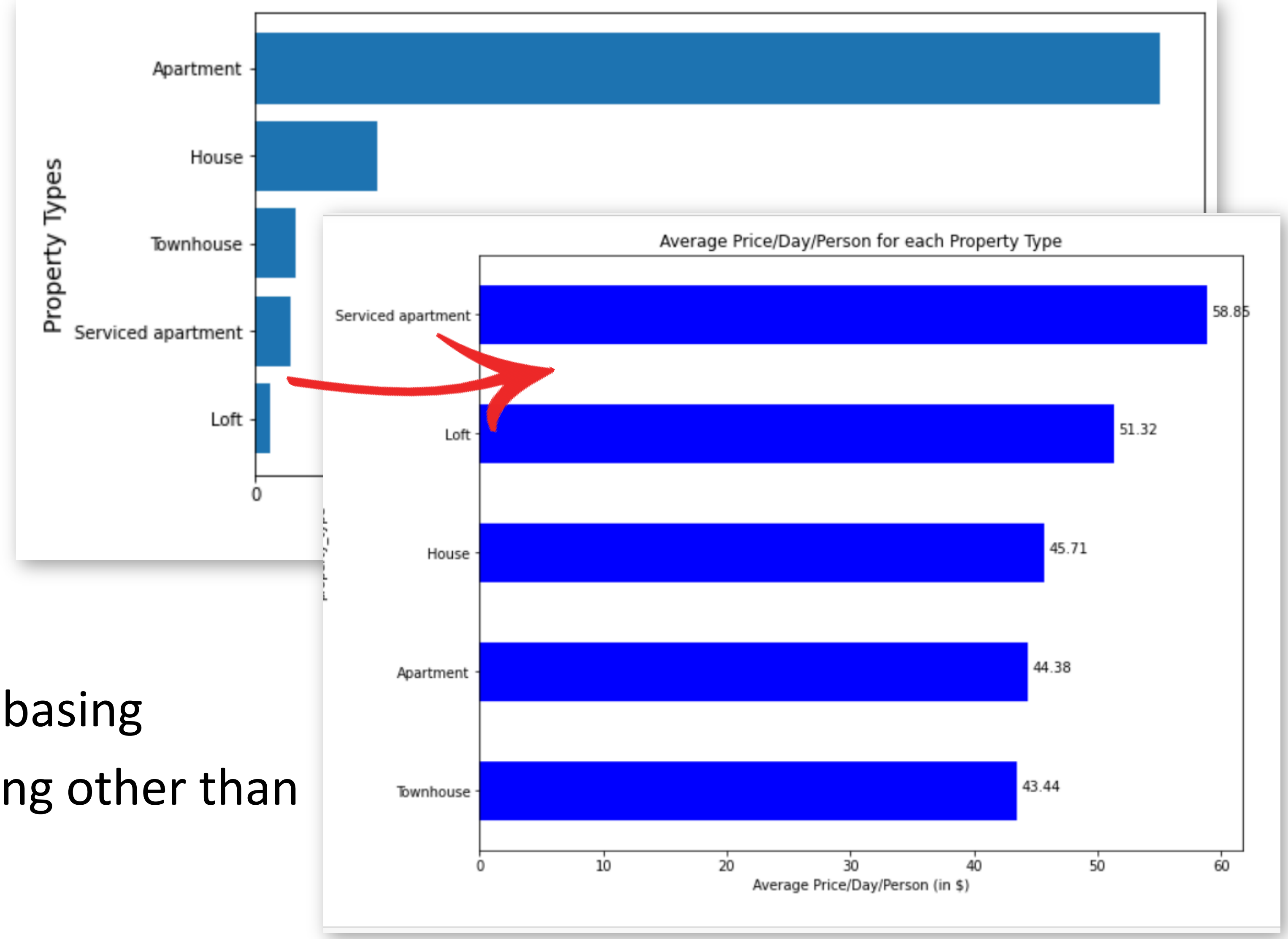
- strip leading and trailing space
- checked unique values in city column
- corrected spelling mistakes
- formalized suburb naming conventions
- dropped non-sensical data
- Drop further irrelevant columns as awareness of data grew.

Lesson learned

- Once the null values had been populated, each cleaning step was a discrete task
  – not dependant on the previous step.
- The work could have been divided among the group to avoid delays.

Third Wave - Decision to only analyse apartments

- Apartments were the vast majority of property types.
- Three other property types were showing higher rates per person
- It felt unwise to be basing decisions on anything other than apartments

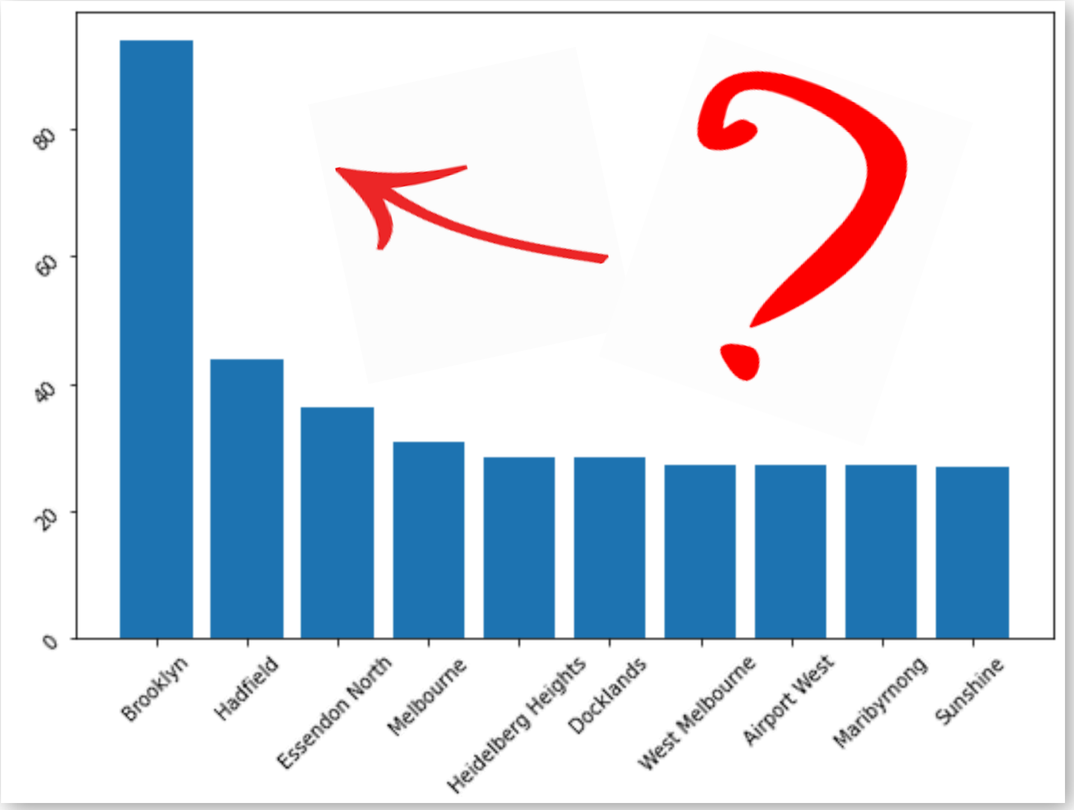


Fourth Wave - small data misleading outcomes

- This led to further exploration of the data and we found a number of results didn't ring true.

- Was Brooklyn really getting the most reviews per month?

- Areas of very small data sets were dramatically skewing the results in a number of areas.

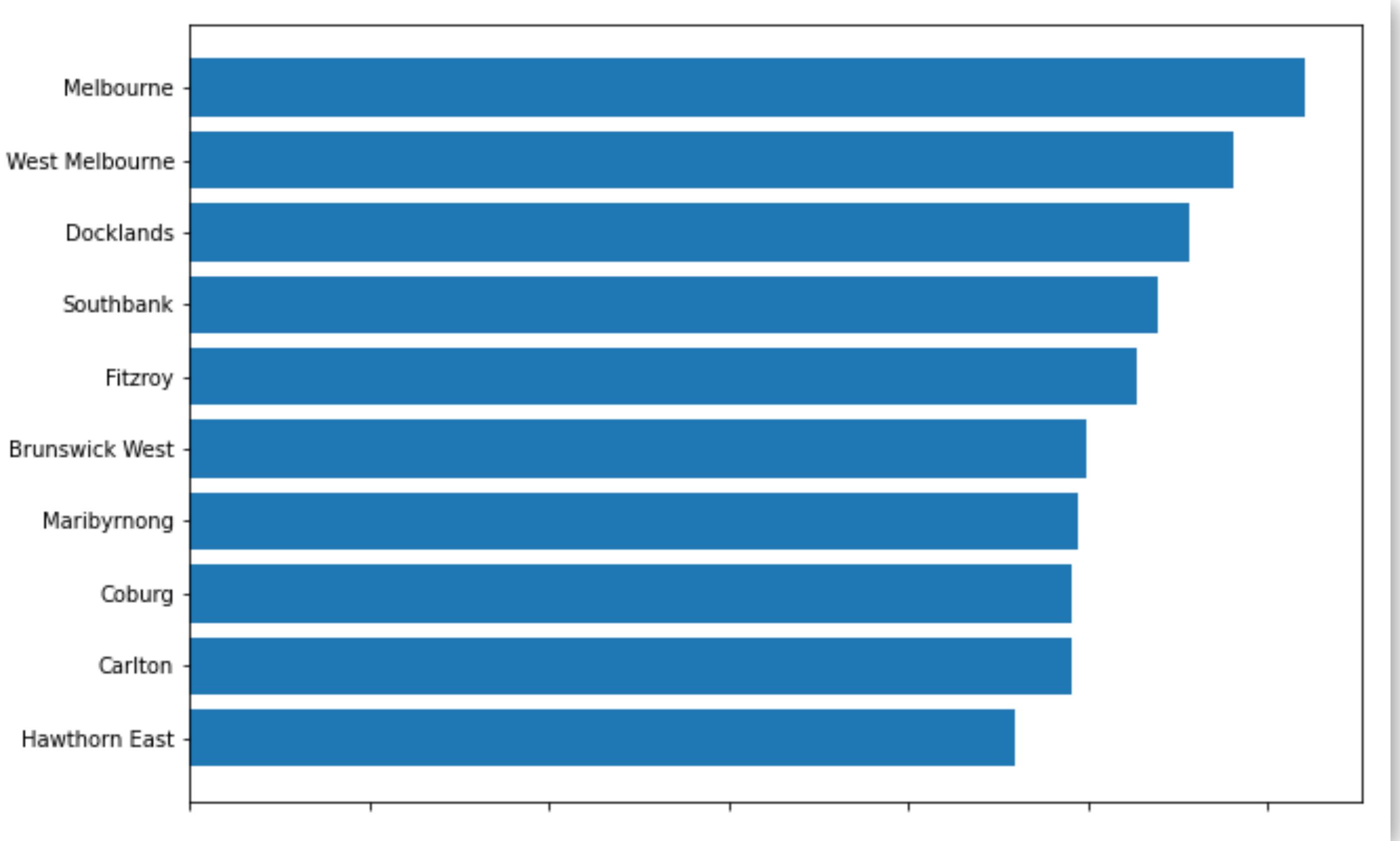


Fourth Wave - small data misleading outcomes

- We made a decision to remove any suburbs with low data points as they were too small to be meaningful

- The resulting data passed the common sense test

At each wave a new CSV was exported



DATA ANALYSIS

Connecting Parameters to Actual Data

	Analysis Parameter	Indicator
1.	Proximity to CBD and Train Station	Calculated distance based on latitude and longitudinal position
2.	Earnings	Price/Day/Person
3.	Popularity of areas (Occupancy)	Reviews/month
4.	Neighborhoods	Suburbs names with more than 20 listings

- STEP 1: Filter for relevant columns: city, suburb, property type, price/day and study overall data information

```
In [4]: # Get the data with relevant columns into a new dataframe and view it.
price_airbnb_data = listing_data[["city","property_type","accommodates","price"]]
price_airbnb_data
```

Out[4]:

	city	property_type	accommodates	price
0	St Kilda	Apartment	3	159.0
1	Richmond	Apartment	2	98.0
2	St Kilda	Apartment	4	190.0
3	Melbourne	Loft	4	228.0
4	Richmond	Apartment	4	138.0
...	...	...	...	...
8922	Melbourne	Apartment	5	156.0
8923	Brunswick West	House	6	199.0
8924	Port Melbourne	Apartment	4	140.0
8925	Preston	Apartment	2	71.0
8926	Richmond	House	4	120.0

```
In [6]: # check datatype to make sure price columns are numerical
price_data_renamed.dtypes

Out[6]: Suburbs          object
property_type    object
accommodates      int64
Price/Day($)    float64
dtype: object
```

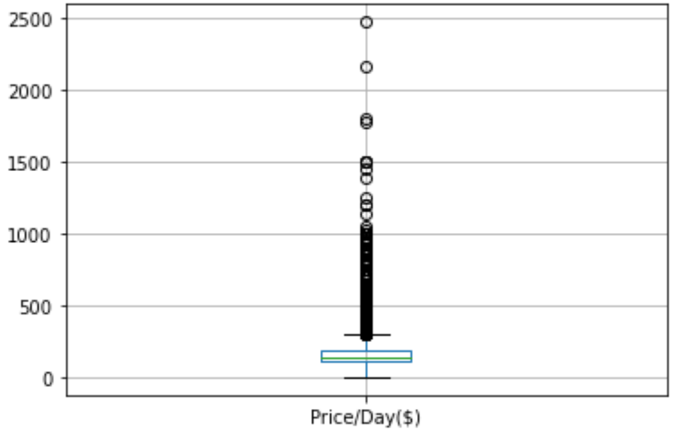
```
In [7]: # Convert Price/Day Column to integer type
price_data_renamed["Price/Day($)"] = price_data_renamed["Price
price_data_renamed.dtypes

Out[7]: Suburbs         object
property_type   object
accommodates     int64
Price/Day($)     int64
dtype: object
```

```
In [207]: # Checking the number of records.
    #number of unique neighbourhoods

print("There are " + str(len(airbnbOccStart)) + " records in the dataframe")


print("There are " + str(len(airbnbOccStart["neighbourhood_cleansed"].unique())) +  " unique neighbourhoods in the dataframe")

print("There are " + str(len(airbnbOccStart["city"].unique())) +  " unique suburbs in the dataframe")

There are 8927 records in the dataframe
There are 13 unique neighbourhoods in the dataframe
There are 55 unique suburbs in the dataframe
```

- STEP 2: Statistical Analysis for checking data quality and take decisions on further filtering

In [8]:
```
# Check via box and whisker plot if there are extreme values in the dataset.
price_data_renamed.boxplot(column='Price/Day($)', return_type='axes')
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x271e3a825f8>



In [25]:
```
prices2=only_apartment['Price/Day/Person($)']
quartiles = prices2.quantile([.25,.5,.75])
lowerq = quartiles[0.25]
upperq = quartiles[0.75]
iqr = upperq-lowerq

print(f"The lower quartile of price/day for entire apartments is: {lowerq}")
print(f"The upper quartile of price/day for entire apartment is: {upperq}")
print(f"The interquartile range of price/day for entire apartment is: {iqr}")
print(f"The the median of price/days for entire apartment is: {quartiles[0.5]} ")

lower_bound = lowerq - (1.5*iqr)
upper_bound = upperq + (1.5*iqr)
print(f"Values below {lower_bound} could be outliers.")
print(f"Values above {upper_bound} could be outliers.")
```

```
The lower quartile of price/day for entire apartments is: 29.75
The upper quartile of price/day for entire apartment is: 51.0
The interquartile range of price/day for entire apartment is: 21.25
The the median of price/days for entire apartment is: 38.75
Values below -2.125 could be outliers.
Values above 82.875 could be outliers.
```

Out[211]:

	Mean reviews per month	Median reviews per month	Variance reviews per month
neighbourhood_cleansed			
Melbourne	2.395453	1.920	4.082722
Moonee Valley	1.669592	1.310	2.061479
Banyule	1.540385	1.470	1.805332
Yarra	1.471980	0.905	2.432001
Maribyrnong	1.430177	1.130	1.633862
Boroondara	1.404396	0.825	2.363073
Port Phillip	1.318080	0.780	2.191741
Stonnington	1.225599	0.670	2.008366
Hobsons Bay	1.221892	0.810	1.521660
Moreland	1.170885	0.600	1.979756
Bayside	1.004000	0.590	1.255563
Glen Eira	0.912857	0.520	0.901331

- STEP 3: Perform required calculations .

In [13]:
```python
# Create a new column for price/day/person and fill calculated values
price_per_person = round((price_data_cleaned2['Price/Day($)']/price_data_cleaned2['accommodates']),2)
price_data_cleaned2['Price/Day/Person($)'] = price_per_person
price_data_cleaned2
```

Out[13]:

	Suburbs	property_type	accommodates	Price/Day($)	Price/Day/Person($)
0	St Kilda	Apartment	3	159	53.00
1	Richmond	Apartment	2	98	49.00
2	St Kilda	Apartment	4	190	47.50
3	Melbourne	Loft	4	228	57.00
4	Richmond	Apartment	4	138	34.50
...	...	...	...	...	...
8922	Melbourne	Apartment	5	156	31.20
8923	Brunswick West	House	6	199	33.17
8924	Port Melbourne	Apartment	4	140	35.00
8925	Preston	Apartment	2		
8926	Richmond	House	4		

8924 rows × 5 columns

In [34]:
```python
from math import radians, cos, sin, asin, sqrt
def dist(lat1, long1, lat2, long2):

    # convert decimal degrees to radians
    lat1, long1, lat2, long2 = map(radians, [lat1, long1, lat2, long2])
    # haversine formula
    dlon = long2 - long1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371* c
    return km
```

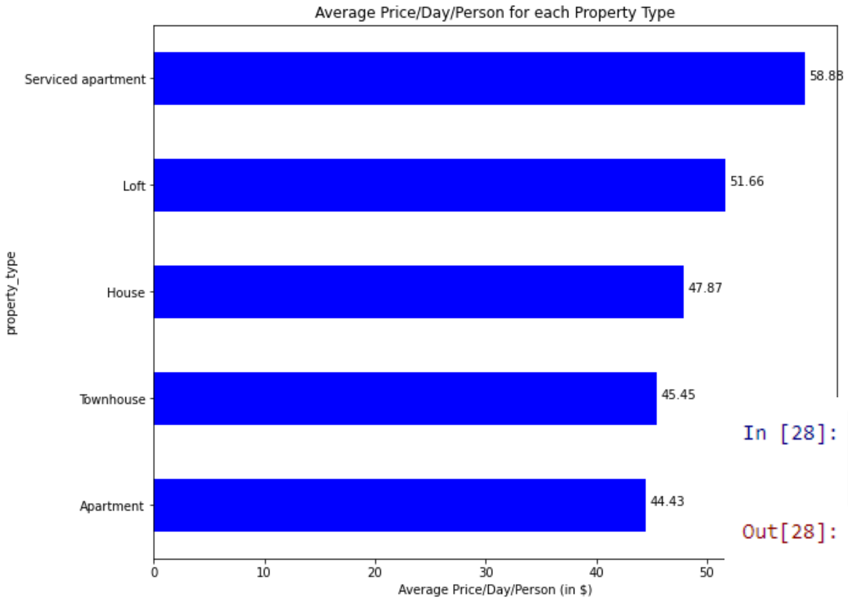
In [35]:
```python
def find_nearest(lat, long):
    distances = station_df.apply(
        lambda row: dist(lat, long, row['LATITUDE'], row['LONGITUDE']),
        axis=1)
    return station_df.loc[distances.idxmin(), 'STOP_NAME']

# append the station name back to the listings file
listing_df['STOP_NAME'] = listing_df.apply(lambda row: find_nearest(row['latitude'], row['longitude']), axis=1)
# To check the data frame if it has a new column of station name (for each and every listing)
listing_df.head()
```

Out[35]:

	id	host_id	host_name	host_is_superhost	host_total_listings_count	street	neighbourhood_cleansed	city	state	zipcode	...	review_scor

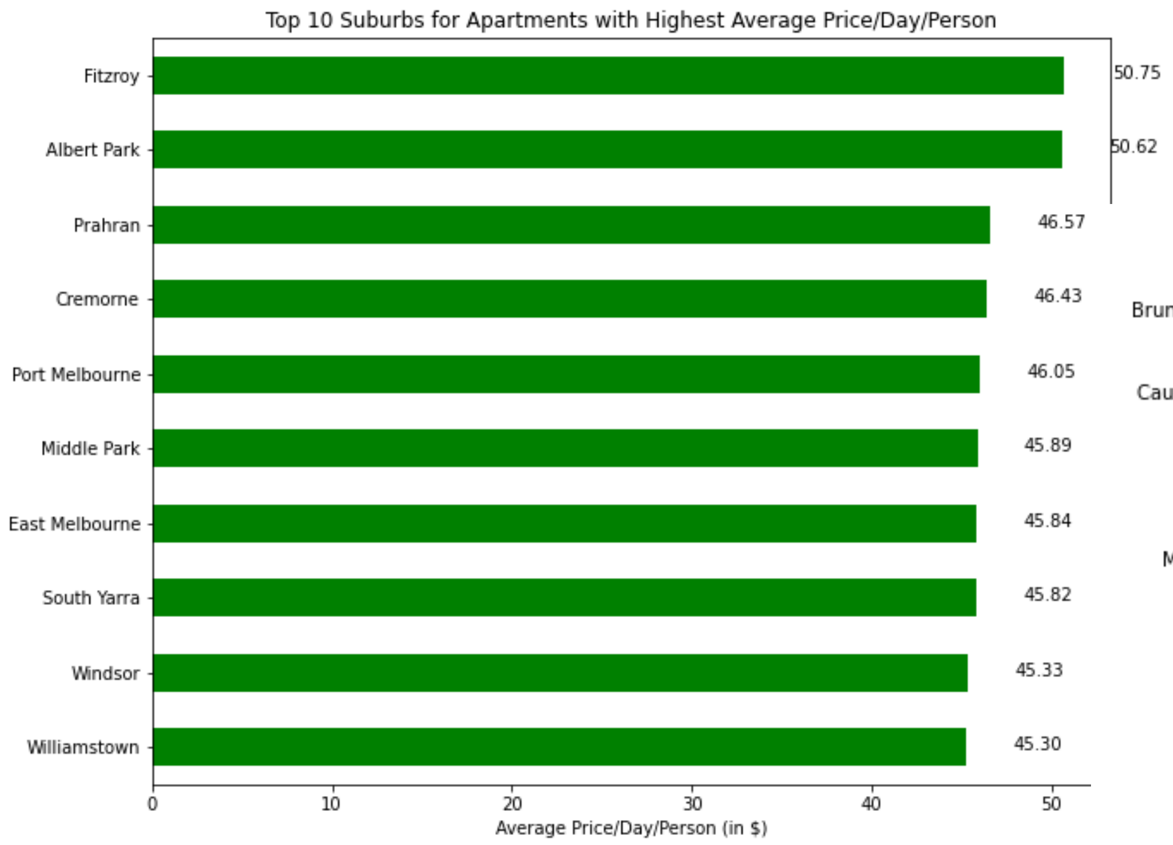
- STEP 4: Group by parameters like apartment type and suburbs to study their effect..

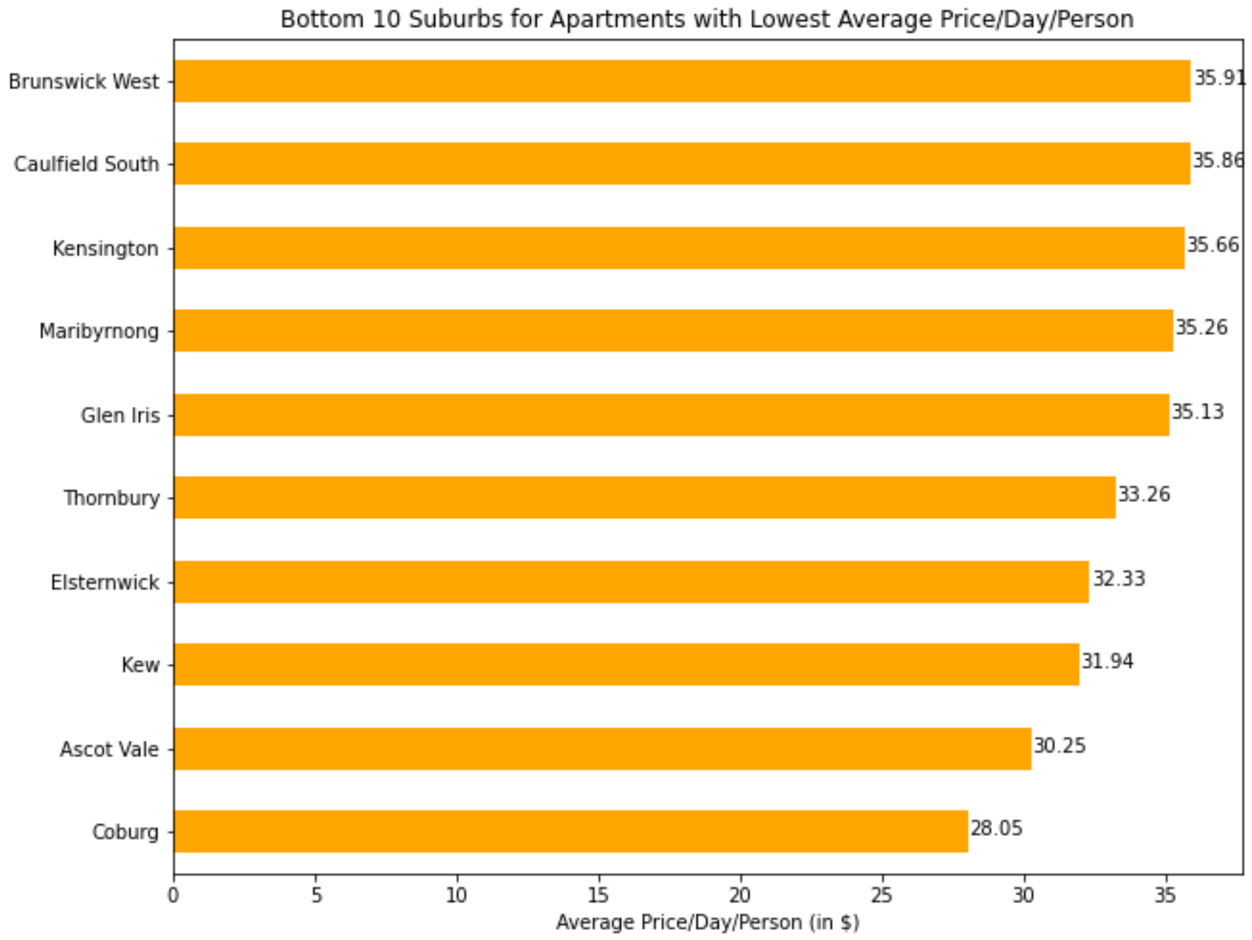
```
In [15]: # Plot the average price/day/person based on propert_type.
bplot3=grouped_property_type.plot(kind="barh",figsize=(10,8),color="blue")
plt.xlabel("Average Price/Day/Person (in $)")
plt.title("Average Price/Day/Person for each Property Type")
for b in bplot3.patches:
    width = b.get_width()
    plt.text(2+b.get_width(), b.get_y()+0.55*b.get_height(),
             '{:1.2f}'.format(width),
             ha='center', va='center')
plt.savefig("Price_Output/PropertyType_Price.png")
```




```
In [28]: # Grouping apartments accomodating small groupsizes of 2-6 based on suburbs, find average price/day and get top
grouped_top = outlier_filtered2.groupby("Suburbs")['Price/Day/Person($)'].mean().sort_values().tail(10)
grouped_top

Out[28]: Suburbs
Williamstown      45.296250
Windsor           45.326104
South Yarra       45.816029
East Melbourne    45.842889
Middle Park       45.886000
Port Melbourne    46.054103
Cremorne          46.427500
Prahran           46.573222
Albert Park       50.615385
Fitzroy           50.746160
Name: Price/Day/Person($), dtype: float64
```

- STEP 5: Rank by suburbs to analyse the top 10 and bottom 10 suburbs for our focus areas.

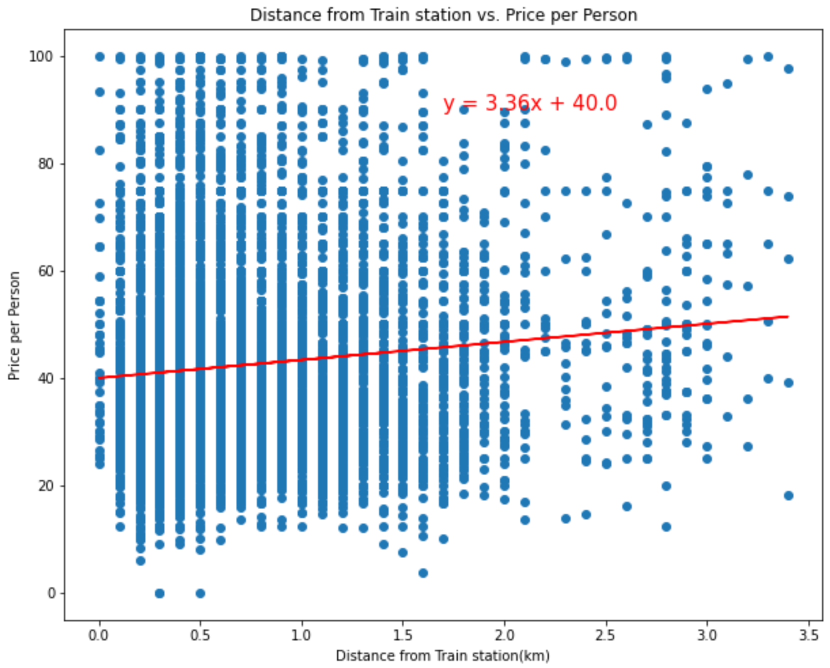




- STEP 6: Use scatter plots and linear regression to establish correlation for our hypothesis.

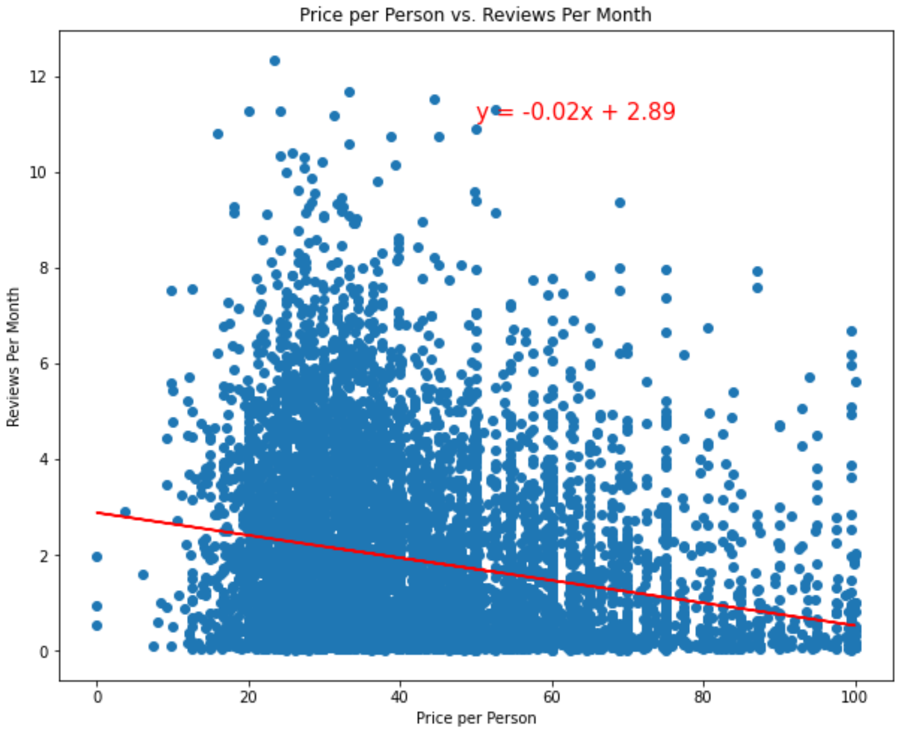
```
In [41]: # function call for Distance from Train station vs. Price per Person
create_plot(pppfltr['station_distance'],
            pppfltr['price_per_person'],
            'Distance from Train station vs. Price per Person',
            'Distance from Train station(km)',
            'Price per Person')
```

The r-value is 0.10553501992697922



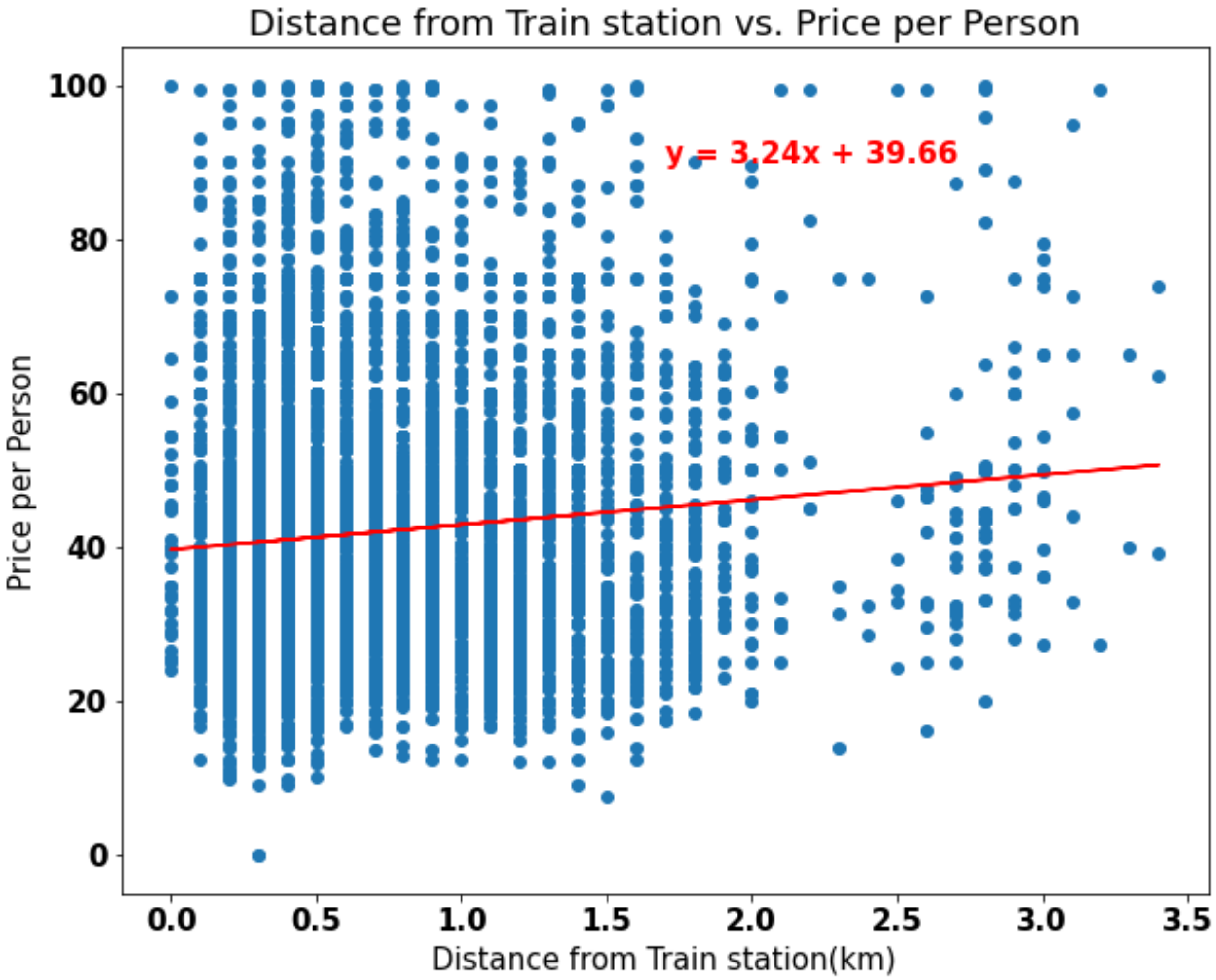

```
In [27]: # function call for Reviews per month vs. Price per Person
create_plot(pppfltr['price_per_person'],
            pppfltr['reviews_per_month'],
            'Price per Person vs. Reviews Per Month',
            'Price per Person',
            'Reviews Per Month')
```

The r-value is -0.22299577756407105



Correlations:

Earnings has a slight positive correlation with distance from train station

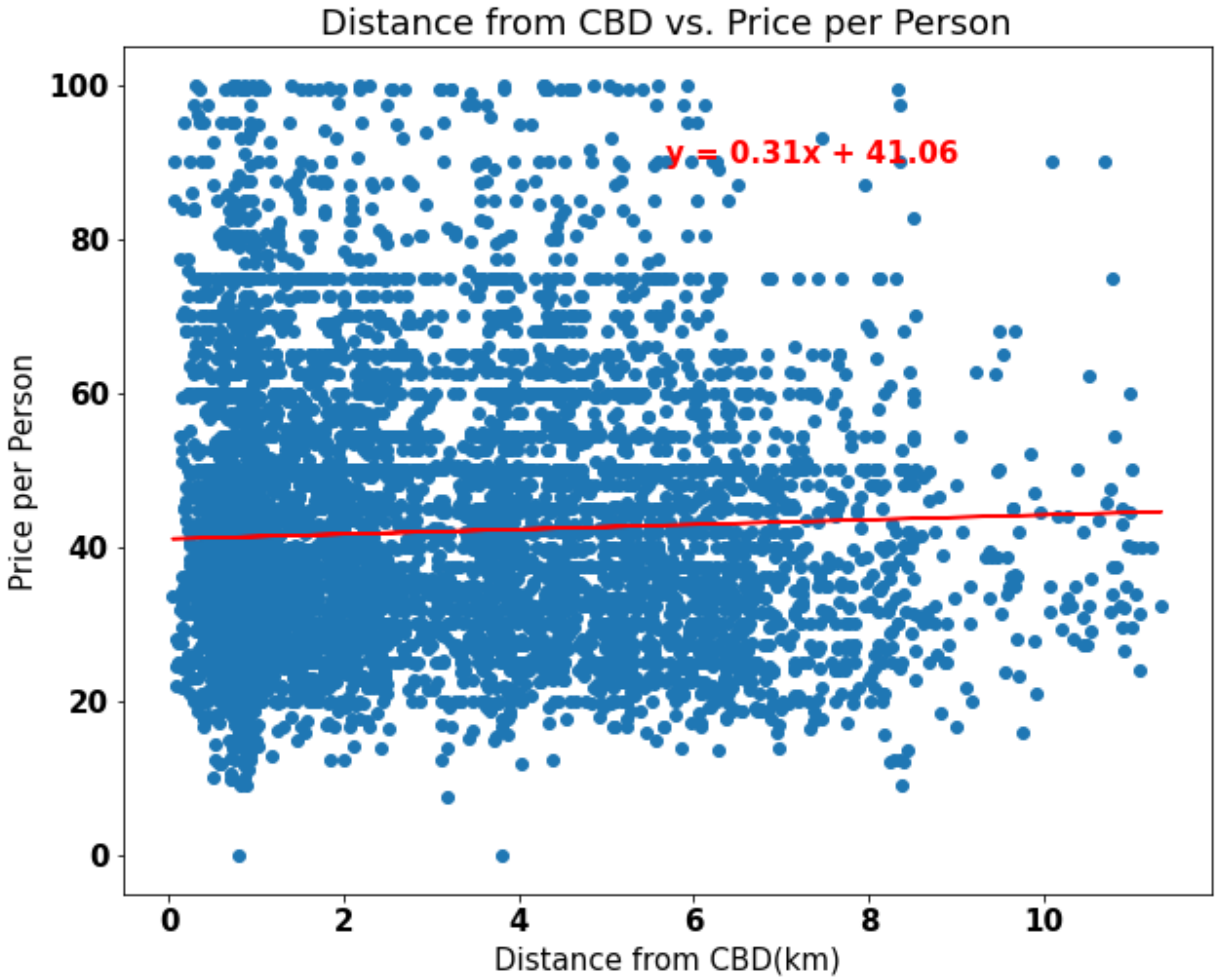


HYPOTHESIS:
(closer) Train proximity will increase earnings (approximate)

DATA INFERENCE:
Properties have increased prospective earnings as they are located further from the train stations

Correlations:

Price per person has a weak positive correalation with distance from CBD

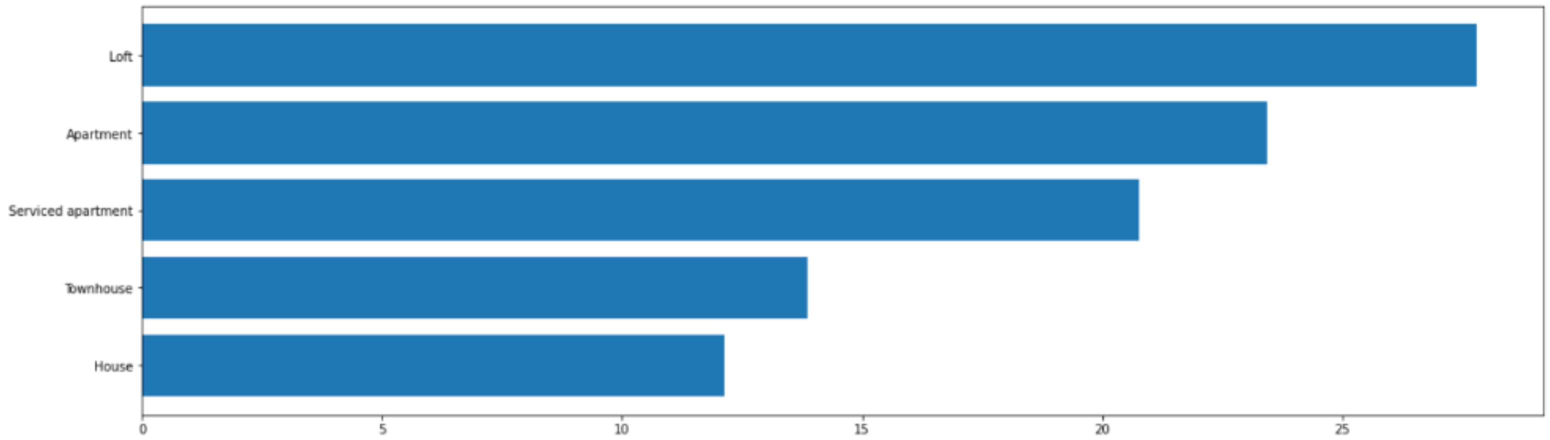


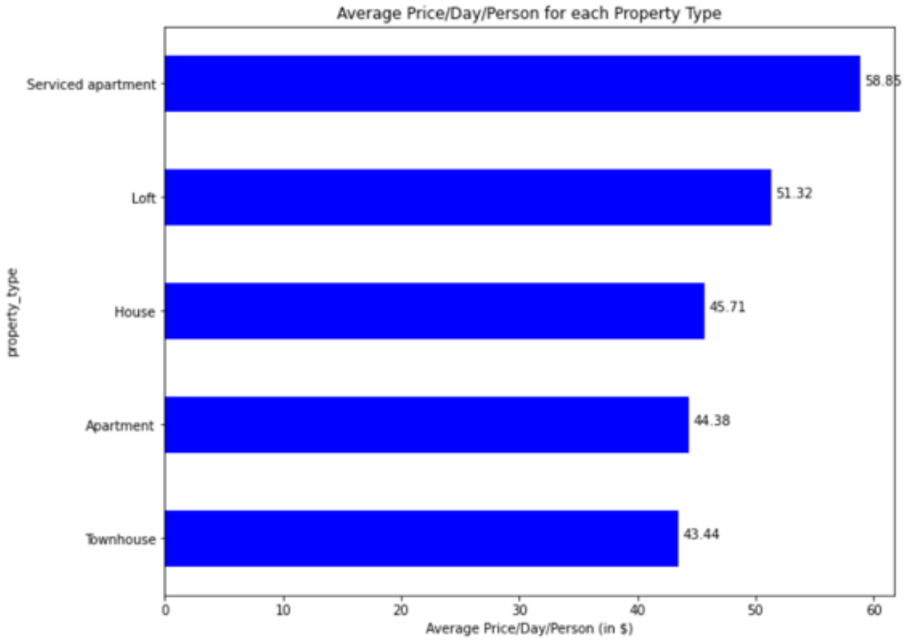
HYPOTHESIS:
(Closer) Proximity to the CBD will increase earnings (approximate)

DATA RESULTS:
Approximate earnings increase slightly for properties located further from the CBD

Correlations:

On property types there is no relationship between Popularity and prospective earnings



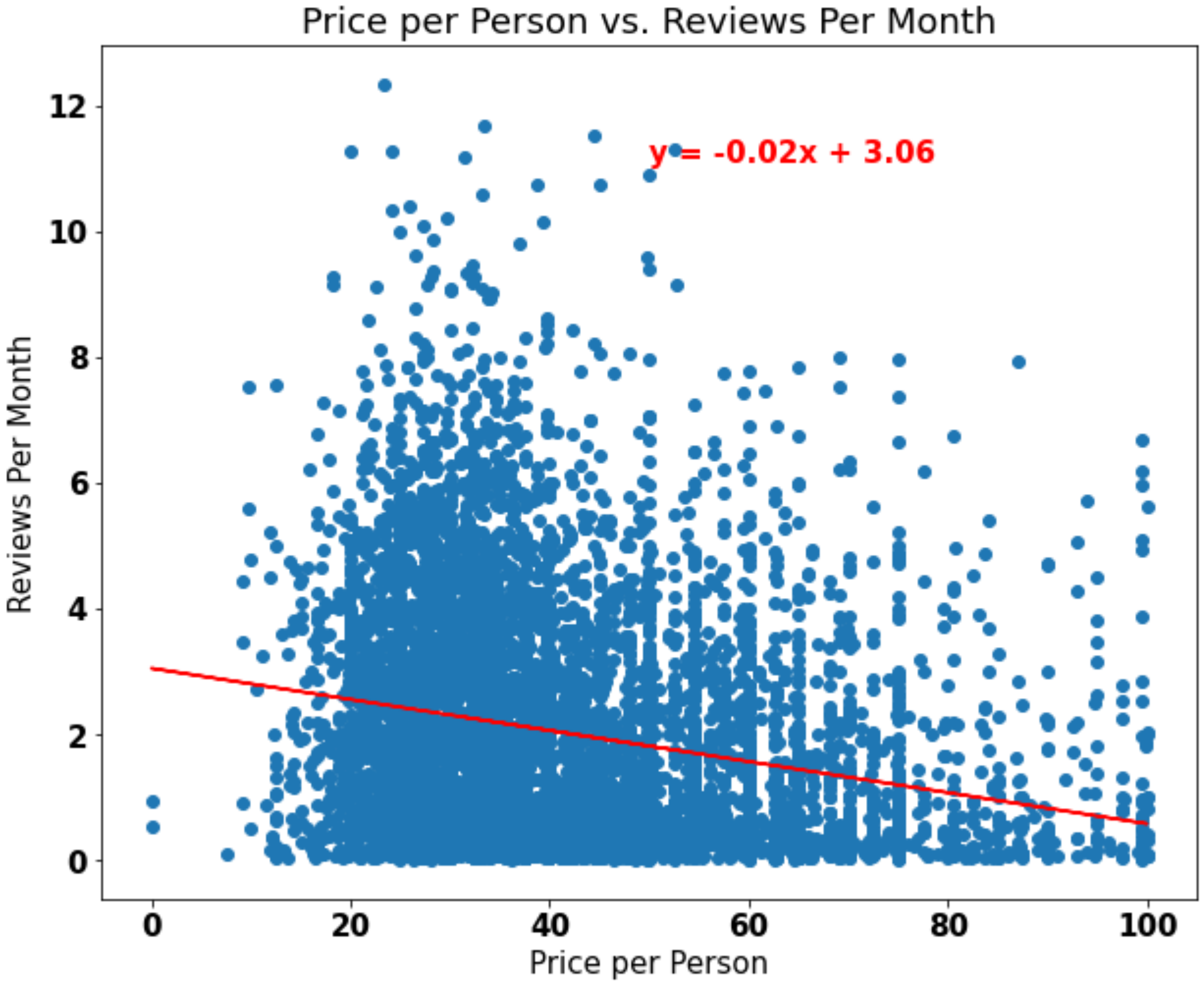


HYPOTHESIS:
Property types with highest earnings are the most popular ones

DATA RESULTS:
The data shows no correlation to prove that to be true

Correlations:

Popularity has a weak negative correlation with prospective earnings



HYPOTHESIS:
The Popularity of properties will be (positively) reflected in higher earnings

DATA RESULTS:
Listings with higher prospective earnings are less popular than listings with lower prospective earnings

Difficulties

- There was not enough time to authenticate the bookings data and make use of it.
- Trying to determine if this data was really going to fulfill our needs without having to search for more data later. We had to commit

Additional Questions

- What aspect of the rating seems to correlate with high earners. Was location more important than quality of the property.

airbnb

Thanks from Linda, Raph,
Swobabika and Jason