import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

color = [

'#ADCE74',

'#A2738C',

'#82DBD8',

'#D3C09A',

'#EC8F6A',

'#6BBA62',

'#F3D516',

'#FFCB3C',

'#FF677D',

'#EADADA',

'#999B84'

]

import os

for dirname, \_, filenames **in** os.walk('/RealEstateAU\_1000\_Samples.csv'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

Input:-

df = pd.read\_csv('/RealEstateAU\_1000\_Samples.csv')

*#df.head()*

Input:-

print(df.shape)

print(df.duplicated().sum())

tabela = pd.DataFrame({

'Unique':df.nunique(),

'Null':df.isna().sum(),

'NullPercent':df.isna().sum() / len(df),

'Types':df.dtypes.values

})

display(tabela)

(1000, 27)

0

|  | Unique | Null | NullPercent | Types |
| --- | --- | --- | --- | --- |
| index | 1000 | 0 | 0.000 | int64 |
| TID | 1000 | 0 | 0.000 | int64 |
| breadcrumb | 2 | 0 | 0.000 | Object |
| category\_name | 2 | 0 | 0.000 | Object |
| property\_type | 13 | 0 | 0.000 | Object |
| building\_size | 169 | 720 | 0.720 | Object |
| land\_size | 346 | 467 | 0.467 | Object |
| preferred\_size | 376 | 391 | 0.391 | Object |
| open\_date | 15 | 698 | 0.698 | Object |
| listing\_agency | 85 | 0 | 0.000 | Object |
| price | 494 | 0 | 0.000 | Object |
| location\_number | 889 | 0 | 0.000 | int64 |
| location\_type | 1 | 0 | 0.000 | Object |
| location\_name | 494 | 0 | 0.000 | Object |
| address | 882 | 12 | 0.012 | Object |
| address\_1 | 882 | 12 | 0.012 | Object |
| city | 56 | 0 | 0.000 | Object |
| state | 1 | 0 | 0.000 | Object |
| zip\_code | 13 | 0 | 0.000 | int64 |
| phone | 84 | 0 | 0.000 | Object |
| latitude | 0 | 1000 | 1.000 | float64 |
| longitude | 0 | 1000 | 1.000 | float64 |
| product\_depth | 4 | 0 | 0.000 | Object |
| bedroom\_count | 10 | 33 | 0.033 | float64 |
| bathroom\_count | 5 | 33 | 0.033 | float64 |
| parking\_count | 11 | 33 | 0.033 | float64 |
| RunDate | 1 | 0 | 0.000 | Object |

Input:-

df.head(2)

Output:-

|  |  | index | TID | breadcrumb | category\_name | property\_type | building\_size | land\_size | preferred\_size | open\_date | listing\_agency | ... | state | zip\_code | phone | latitude | longitude | product\_depth | bedroom\_count | bathroom\_count | parking\_count | RunDate |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 0 | 1350988 | Buy>NT>DARWIN CITY | Real Estate & Property for sale in DARWIN CITY... | House | NaN | NaN | NaN | Added 2 hours ago | Professionals - DARWIN CITY | ... | NT | 800 | 08 8941 8289 | NaN | NaN | premiere | 2.0 | 1.0 | 1.0 | 2022-05-27 15:54:05 |
| 1 |  | 1 | 1350989 | Buy>NT>DARWIN CITY | Real Estate & Property for sale in DARWIN CITY... | Apartment | 171m² | NaN | 171m² | Added 7 hours ago | Nick Mousellis Real Estate - Eview Group Member | ... | NT | 800 | 0411724000 | NaN | NaN | premiere | 3.0 | 2.0 | 2.0 | 2022-05-27 15:54:05 |

2 rows × 27 columns

Transformation

Input:-

*# eliminado colunas*

df.drop(['index','latitude','longitude','state','location\_type','location\_name','address\_1','breadcrumb','RunDate'],axis=1,inplace=True)

colunas = ['category\_name','listing\_agency','city']

for i **in** colunas:

df[i] = df[i].str.upper()

colunas = ['address']

for col **in** colunas:

df[col] = df[col].fillna('Unknow')

colunas = ['building\_size','preferred\_size','land\_size']

for col **in** colunas:

df[col] = df[col].str.replace('m²','').str.replace('ha','')

print('ok')

ok

Input:-

colunas = df['listing\_agency'].str.split('-',1,expand=True)

df['agency\_names'] = colunas[0]

df['agency\_names2'] = colunas[1]

df['agency\_names2'] = df['agency\_names2'].str.replace(' ','')

df['agency\_names2'] = df['agency\_names2'].replace(' ','Unknown')

df['agency\_names2'] = df['agency\_names2'].replace( ' ','Unknown')

*# espaçamento existente na linha*

df['agency\_names'] = df['agency\_names'].replace( 'FOR SALE BY OWNER ','FOR SALE BY OWNER')

Input:-

df['price'] = df['price'].replace('$1.15m','$1.150000')

df['price'] = df['price'].replace('JUST LIKE THAT: UNDER CONTRACT IN 7 DAYS','JUST LIKE THAT: UNDER CONTRACT IN SEVEN DAYS')

df['price'] = df['price'].replace('Offers over $1.2m','Offers over $1.200000')

df['price'] = df['price'].replace('Auction Wednesday 1st of June 2022','Auction')

df['price'] = df['price'].replace('Auction 8th June on site','Auction')

df['price'] = df['price'].replace('Auction - Wednesday 15th June 2022 at 5.30pm','Auction')

df['price'] = df['price'].replace('AUCTION: Saturday 4th Jun @11am On-Site',"AUCTION")

df['price'] = df['price'].replace('JUST LIKE THAT: UNDER CONTRACT IN 5 DAYS','JUST LIKE THAT: UNDER CONTRACT IN FIVE DAYS')

df['price'] = df['price'].replace('Offers Over $500,000 - Offers by 6.30pm 22/6/22','Offers Over $500,000')

df['price'] = df['price'].replace('Negotiable Above 1.5M','Negotiable Above 1.500000')

df['price'] = df['price'].replace('$147 000','$147,000')

df['price'] = df['price'].replace('$369 000','$369,000')

df['price'] = df['price'].replace('$545,000 ( over 1000sqm of land)','$545,000 ( over thousand sqm of land)')

df['price'] = df['price'].replace('$450 000','$450,000')

Input:-

df['property\_type'].unique()

Output:-

array(['House', 'Apartment', 'Unit', 'Studio', 'Residential Land',

'Block Of Units', 'Townhouse', 'Acreage', 'Duplex/Semi-detached',

'Other', 'Villa', 'Warehouse', 'Lifestyle'], dtype=object)

Input:-

colunas = df['price'].str.split('-',1,expand=True)

df['price'] = colunas[0]

df['priceConsidered'] = colunas[1]

colunas = df['price'].str.split(' ',1,expand=True)

df['price'] = colunas[0]

df['priceConsidered'] = colunas[1]

Input:-

*# removendo números*

df['priceCondition'] = df['priceConsidered'].replace(r'[0-9]','',regex=True)

df['priceCondition'] = df['priceCondition'].str.replace('$','',regex=True)

df['priceCondition'] = df['priceCondition'].str.replace('!','',regex=True).str.replace(' ,','',regex=True)

df['priceCondition'] = df['priceCondition'].str.upper()

*# removendo letras*

df['priceConsidered'] = df['priceConsidered'].replace('[^0-9]','',regex=True)

*# removendo letras*

df['price'] = df['price'].replace('[^0-9]','',regex=True)

*# removendo símbolo*

df['price'] = df['price'].replace('$','',regex=True).replace('-$','',regex=True)

df['price'] = df['price'].replace('',np.nan)

df['price'] = df['price'].fillna(df['priceConsidered'])

df['price'] = pd.to\_numeric(df['price'], errors='coerce')

Input:-

df['land\_size'] = pd.to\_numeric(df['land\_size'], errors='coerce')

df['preferred\_size'] = pd.to\_numeric(df['preferred\_size'], errors='coerce')

df['building\_size'] = pd.to\_numeric(df['building\_size'], errors='coerce')

Input:-

*# new data*

df = df[(df['property\_type'] != 'Lifestyle') & (df['property\_type'] != 'Warehouse')]

Graphic

attributes by property type

Input:-

*'''*

*not to exclude the missing data, as it could identify which of the columns had more null values,*

*but other than that, I used Pandas' pivot\_table to identify each attribute by property type,*

*the index of the property\_type column, is linked to the number of rooms existing in them,*

*each column below represents the number of bathrooms, parking, land size and building*

*'''*

pd.pivot\_table(df,index=['property\_type','bedroom\_count'],

values=['bathroom\_count','parking\_count','building\_size','land\_size','preferred\_size']).style.background\_gradient(axis=0)

Output:-

|  |  | bathroom\_count | building\_size | land\_size | parking\_count | preferred\_size |
| --- | --- | --- | --- | --- | --- | --- |
| property\_type | bedroom\_count |  |  |  |  |  |
| Acreage | 2.0 | 2.000000 | nan | 2.000000 | 5.000000 | 2.000000 |
| 3.0 | 1.000000 | nan | 7.685000 | 1.000000 | 7.685000 |
| 4.0 | 2.666667 | 260.000000 | 7.140000 | 5.666667 | 7.140000 |
| 5.0 | 2.000000 | nan | 9.710000 | 4.000000 | 9.710000 |
| 6.0 | 2.000000 | nan | 2.000000 | 3.000000 | 2.000000 |
| Apartment | 1.0 | 1.000000 | 67.466667 | 93.333333 | 0.866667 | 67.466667 |
| 2.0 | 1.835294 | 122.455556 | 126.158824 | 1.411765 | 122.455556 |
| 3.0 | 1.986301 | 209.851818 | 191.862500 | 1.931507 | 209.851818 |
| 4.0 | 2.125000 | nan | 252.400000 | 2.000000 | nan |
| 5.0 | 3.000000 | 415.000000 | nan | 2.000000 | 415.000000 |
| Block Of Units | 1.0 | 1.000000 | nan | nan | 1.000000 | nan |
| 2.0 | 1.000000 | nan | nan | 1.000000 | nan |
| 3.0 | 2.000000 | 377.440000 | 807.000000 | 2.000000 | 377.440000 |
| 6.0 | 4.000000 | nan | nan | 4.000000 | nan |
| 8.0 | 4.000000 | nan | 812.000000 | 2.000000 | 812.000000 |
| Duplex/Semi-detached | 2.0 | 1.000000 | nan | nan | 1.000000 | nan |
| 3.0 | 1.846154 | nan | 352.600000 | 2.307692 | 352.600000 |
| 4.0 | 2.000000 | 302.333333 | 346.000000 | 2.000000 | 302.333333 |
| House | 1.0 | 1.000000 | 81.000000 | 332.000000 | 1.000000 | 332.000000 |
| 2.0 | 1.434783 | 135.333333 | 329.383333 | 1.565217 | 294.586364 |
| 3.0 | 1.519337 | 197.369821 | 643.007143 | 2.497238 | 636.977407 |
| 4.0 | 2.062500 | 246.642885 | 657.429683 | 2.750000 | 657.429683 |
| 5.0 | 2.500000 | 274.071429 | 608.618462 | 4.136364 | 608.618462 |
| 6.0 | 3.000000 | 281.500000 | 486.678333 | 4.900000 | 486.678333 |
| 7.0 | 5.000000 | 815.000000 | nan | 10.000000 | nan |
| 9.0 | 4.000000 | nan | 837.000000 | 6.000000 | 837.000000 |
| Other | 0.0 | 1.000000 | nan | nan | 0.000000 | nan |
| 2.0 | 1.000000 | nan | nan | 1.000000 | nan |
| 3.0 | 2.000000 | 167.000000 | nan | 5.000000 | nan |
| Studio | 0.0 | 1.000000 | 55.000000 | nan | 1.000000 | 55.000000 |
| 1.0 | 1.000000 | 52.000000 | nan | 1.000000 | 52.000000 |
| Townhouse | 2.0 | 1.090909 | 90.000000 | 45.000000 | 1.000000 | 75.000000 |
| 3.0 | 2.080000 | 103.000000 | 342.750000 | 2.160000 | 335.937500 |
| 4.0 | 2.500000 | 251.000000 | 357.000000 | 2.500000 | 251.000000 |
| Unit | 0.0 | 1.000000 | 12.000000 | nan | 1.000000 | 12.000000 |
| 1.0 | 1.000000 | 63.823529 | 76.600000 | 0.886792 | 70.000000 |
| 2.0 | 1.451613 | 118.424242 | 164.829787 | 1.548387 | 151.436620 |
| 3.0 | 1.980392 | 208.000000 | 251.000000 | 2.078431 | 218.857143 |
| 4.0 | 2.000000 | 356.000000 | 356.000000 | 1.000000 | 356.000000 |
| Villa | 2.0 | 1.000000 | 80.000000 | 208.500000 | 1.500000 | 157.500000 |
| 3.0 | 2.000000 | nan | 331.000000 | 3.000000 | 331.000000 |

property type by product depth

Input:-

*'''*

*here we have a table with the sum of bedrooms, bathrooms and parking in each property by classification*

*Colors indicate the maximum number of attributes a property contains in the data.*

*'''*

pd.pivot\_table(df,index=['product\_depth','property\_type'],

values=['bedroom\_count','bathroom\_count','parking\_count'],aggfunc='sum').style.background\_gradient(axis=0)

Output:-

|  |  | bathroom\_count | bedroom\_count | parking\_count |
| --- | --- | --- | --- | --- |
| product\_depth | property\_type |  |  |  |
| feature | Apartment | 106.000000 | 137.000000 | 83.000000 |
| Block Of Units | 6.000000 | 11.000000 | 2.000000 |
| Duplex/Semi-detached | 3.000000 | 6.000000 | 4.000000 |
| House | 105.000000 | 209.000000 | 146.000000 |
| Townhouse | 4.000000 | 7.000000 | 3.000000 |
| Unit | 65.000000 | 85.000000 | 63.000000 |
| midtier | Apartment | 4.000000 | 4.000000 | 4.000000 |
| House | 6.000000 | 14.000000 | 6.000000 |
| Residential Land | 0.000000 | 0.000000 | 0.000000 |
| Unit | 1.000000 | 2.000000 | 1.000000 |
| premiere | Acreage | 9.000000 | 17.000000 | 19.000000 |
| Apartment | 178.000000 | 234.000000 | 167.000000 |
| Block Of Units | 9.000000 | 15.000000 | 9.000000 |
| Duplex/Semi-detached | 23.000000 | 38.000000 | 28.000000 |
| House | 617.000000 | 1194.000000 | 920.000000 |
| Other | 1.000000 | 2.000000 | 1.000000 |
| Residential Land | 0.000000 | 0.000000 | 0.000000 |
| Studio | 1.000000 | 1.000000 | 1.000000 |
| Townhouse | 59.000000 | 88.000000 | 61.000000 |
| Unit | 209.000000 | 291.000000 | 223.000000 |
| Villa | 6.000000 | 10.000000 | 9.000000 |
| standard | Acreage | 9.000000 | 16.000000 | 17.000000 |
| Apartment | 78.000000 | 96.000000 | 64.000000 |
| Block Of Units | 1.000000 | 2.000000 | 1.000000 |
| Duplex/Semi-detached | 8.000000 | 15.000000 | 8.000000 |
| House | 97.000000 | 177.000000 | 152.000000 |
| Other | 5.000000 | 6.000000 | 10.000000 |
| Residential Land | 0.000000 | 0.000000 | 0.000000 |
| Studio | 1.000000 | 0.000000 | 1.000000 |
| Townhouse | 6.000000 | 10.000000 | 6.000000 |
| Unit | 62.000000 | 80.000000 | 60.000000 |

Price

In [15]:

plt.figure(figsize=(14,7))

sns.histplot(x=df['price'])

plt.ticklabel\_format(style='plain')



property\_type %

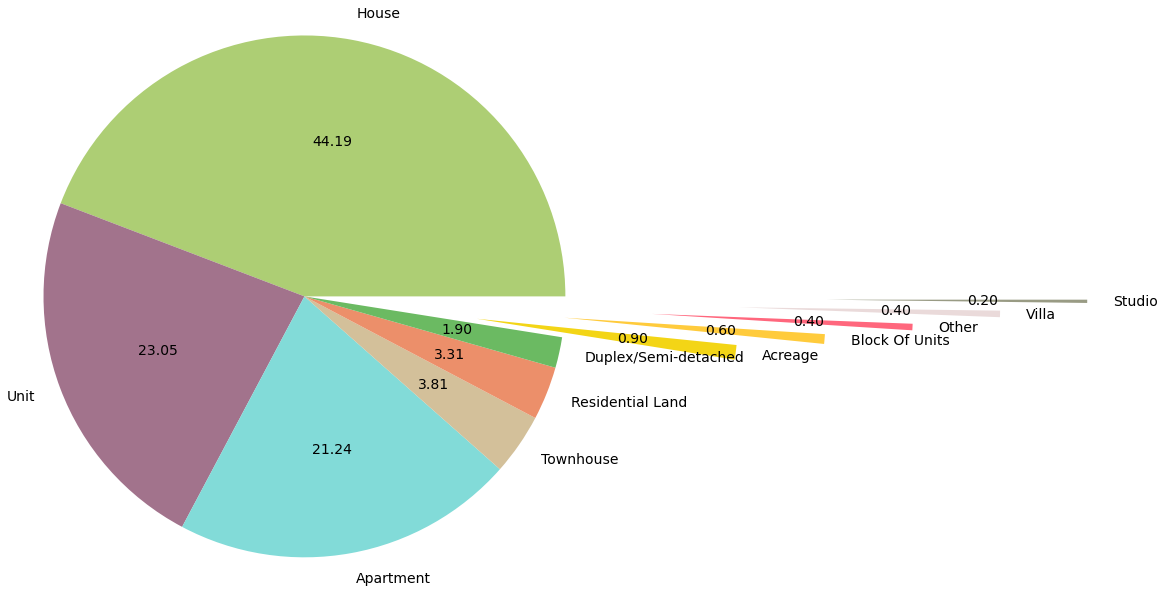
Input:-

df['property\_type'].value\_counts().plot.pie(autopct='**%.2f**',radius=3, textprops={'size':14},

explode=(0,0,0,0,0,0,2,3,4,5,6), colors=color)

plt.axis('off')

plt.show()



product\_depth %

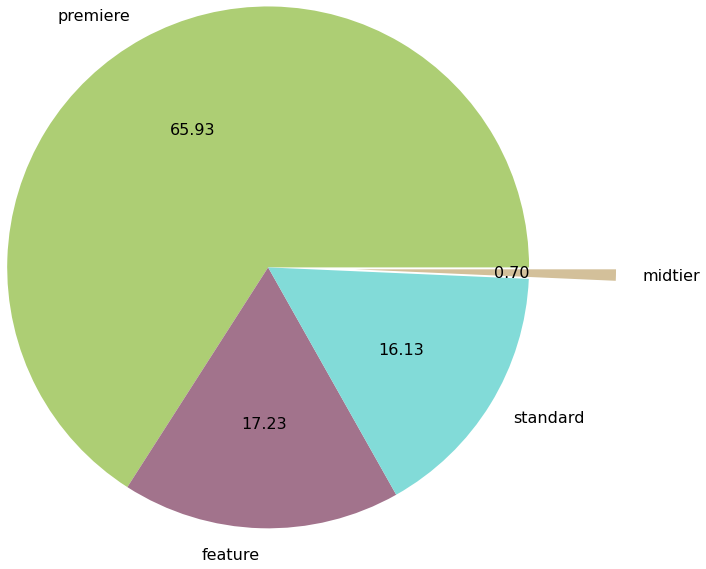
In [17]:

df['product\_depth'].value\_counts().plot.pie(autopct='**%.2f**', explode=(0,0,0,1), textprops={'size':16}

,radius=3,colors=color)

plt.axis('off')

plt.show()



bedroom count %

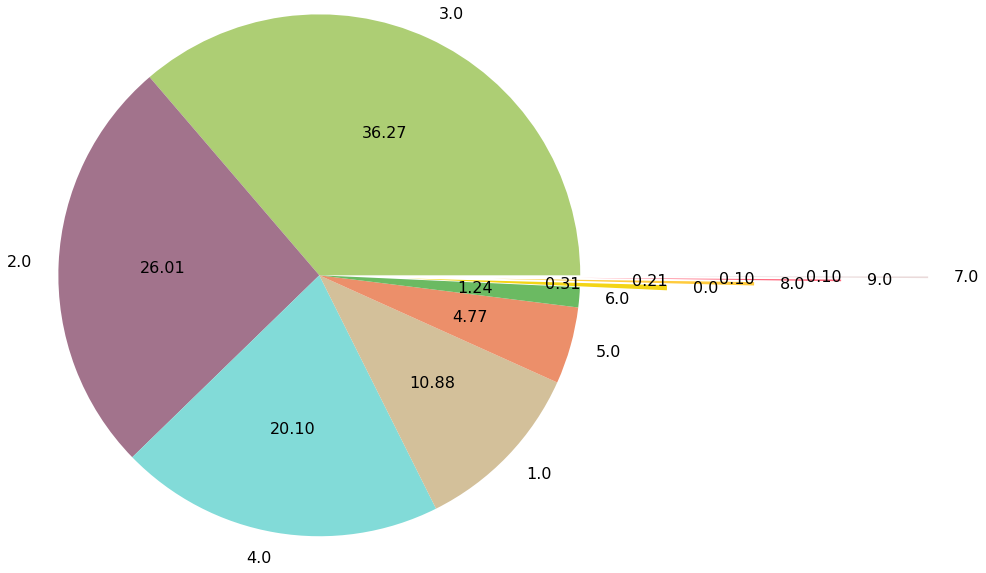
Input:-

df['bedroom\_count'].value\_counts().plot.pie(autopct='**%.2f**', explode=(0,0,0,0,0,0,1,2,3,4)

,radius=3,colors=color, textprops={'size':16})

plt.axis('off')

plt.show()



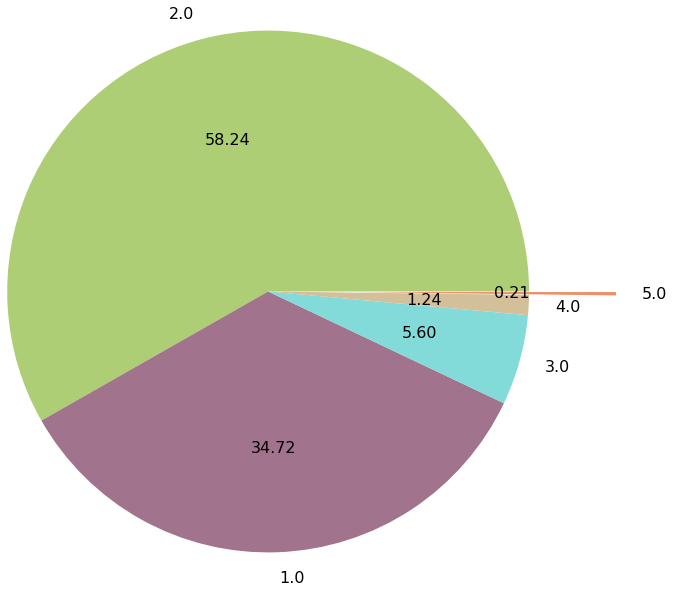
bathroom count %

Input:-

df['bathroom\_count'].value\_counts().plot.pie(autopct='**%.2f**',radius=3, explode=(0,0,0,0,1), colors=color, textprops={'size':16})

plt.axis('off')

plt.show()



parking count %

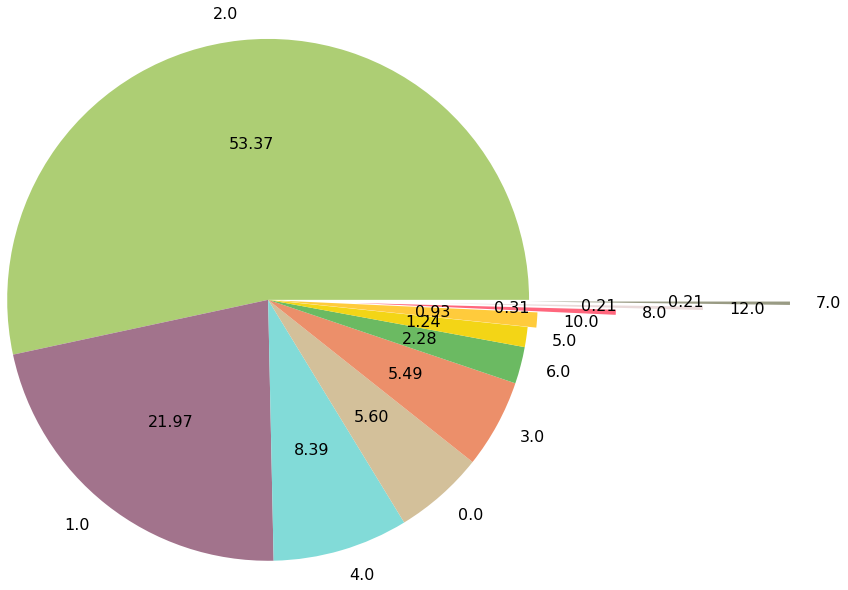
Input:-

df['parking\_count'].value\_counts().plot.pie(autopct='**%.2f**',radius=3,explode=(0,0,0,0,0,0,0,0.1,1,2,3),

colors=color, textprops={'size':16})

plt.axis('off')

plt.show()



average property value by rating

Input:-

plt.figure(figsize=(14,24))

ax = sns.barplot(y=df['property\_type'],x=df['price'], hue=df['product\_depth'],ci=None, palette=color)

plt.yticks(fontsize=16)

plt.ylabel(None)

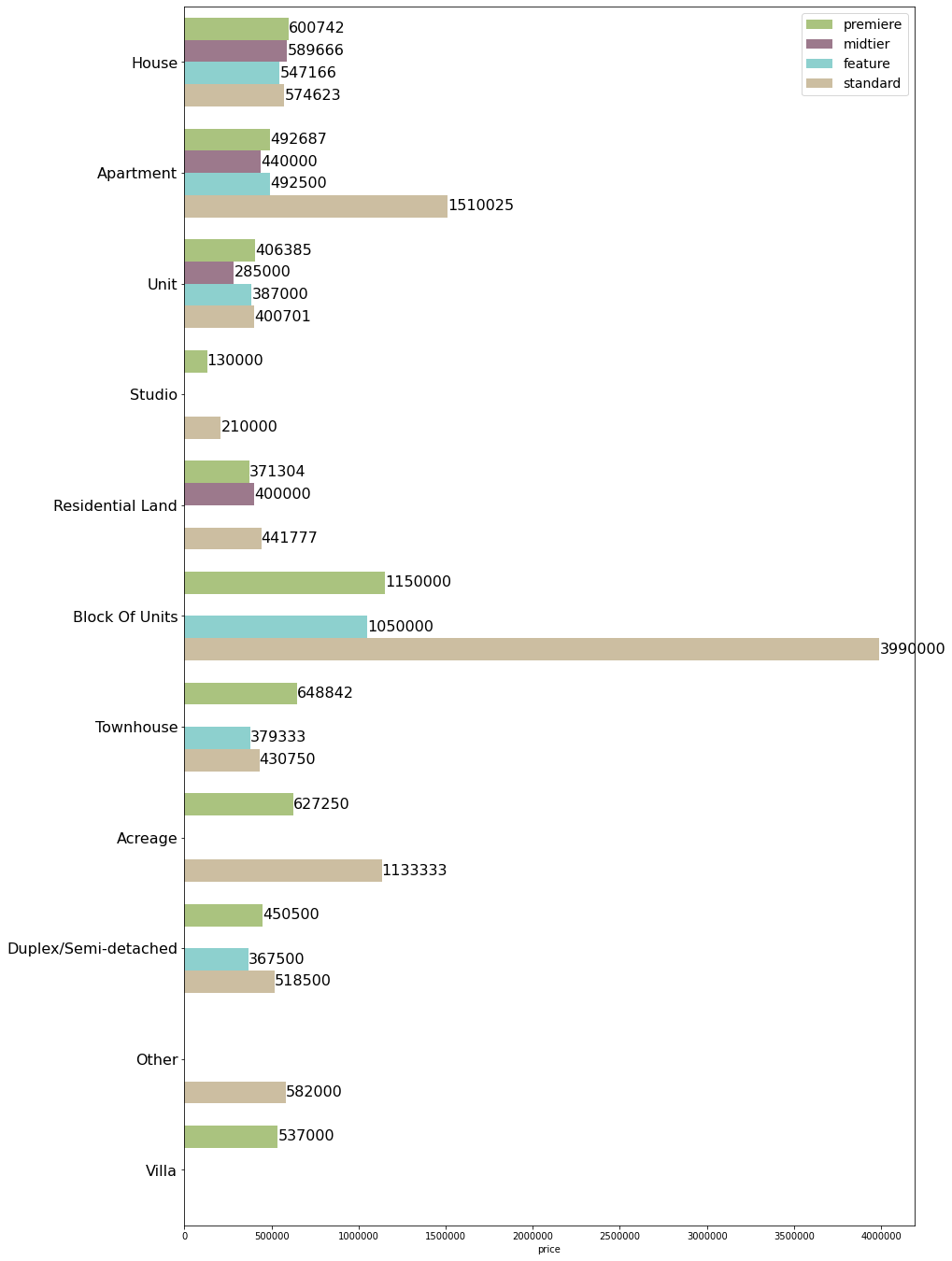
plt.legend(fontsize=14)

for i **in** ax.containers:

ax.bar\_label(i, fontsize=16,fmt='**%d**')

plt.ticklabel\_format(style='plain',axis='x')

plt.show()



price comparison per property, by land size and classification

Input:-

data = df['property\_type'].unique()

plt.figure(figsize=(14,47))

for i,col **in** enumerate(data):

ax = plt.subplot(13,1,i + 1)

sns.lineplot(x='land\_size',y='price', data=df[df['property\_type']==col], hue=df['product\_depth'],ci=None)

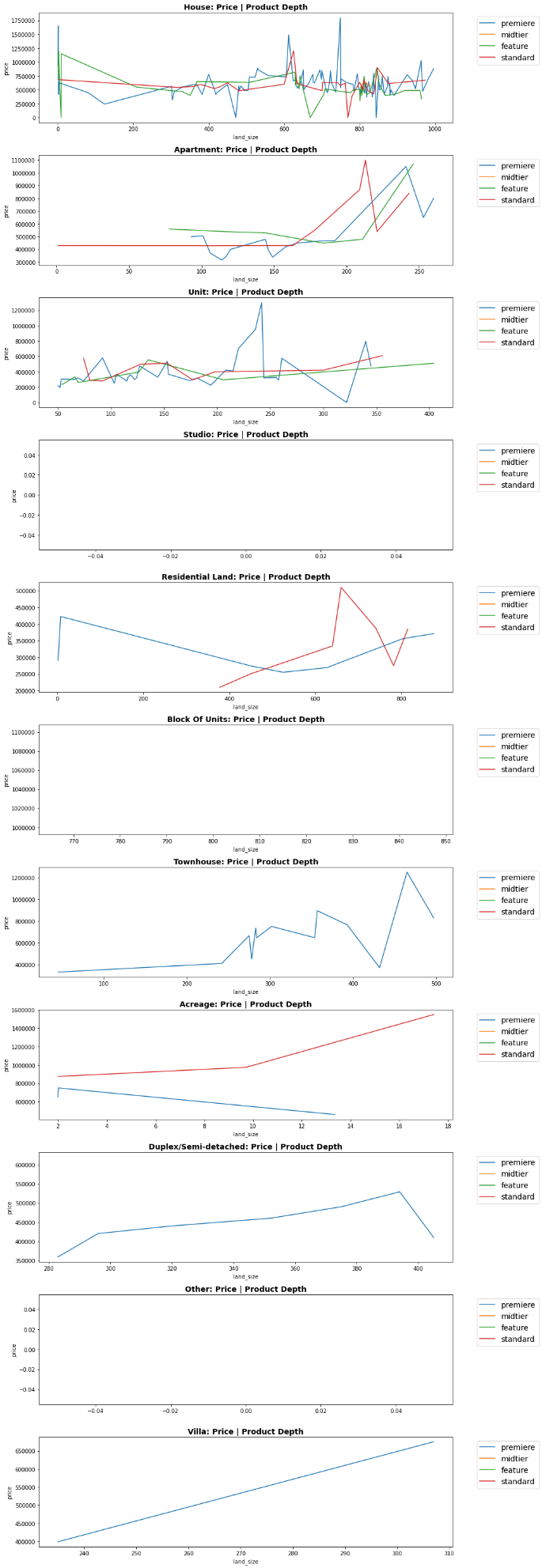
plt.title(f'**{**col**}**: Price | Product Depth', fontsize=14, fontweight='bold')

plt.legend(fontsize=14,loc=2, bbox\_to\_anchor=(1.05,1))

plt.ticklabel\_format(style='plain')

plt.tight\_layout()

plt.show()



price comparison by land size and preferred size

Input:-

colunas = ['land\_size','preferred\_size']

data = df['property\_type'].unique()

plt.figure(figsize=(14,47))

for i,col **in** enumerate(data):

for d **in** colunas:

ax = plt.subplot(13,1,i + 1)

sns.lineplot(x=d,y='price', data=df[df['property\_type']==col], label=f'**{**d**}**',ci=None)

plt.title(f'**{**col**}**: Price | Property Type', fontsize=14, fontweight='bold')

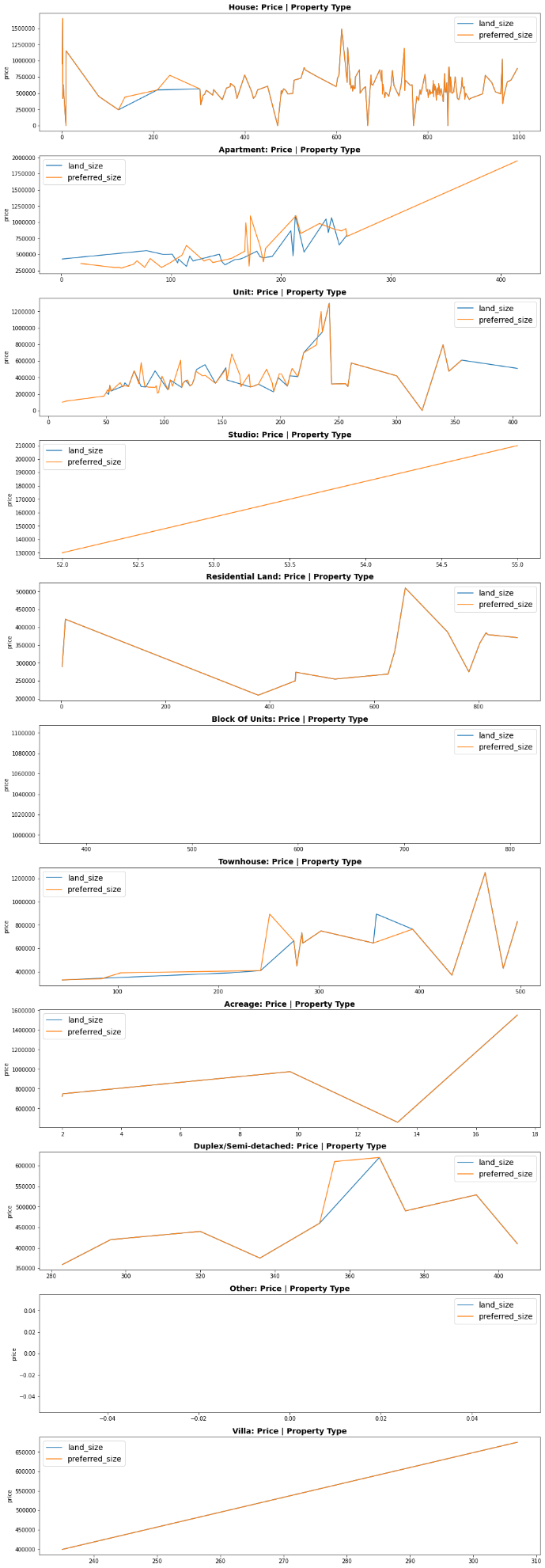
plt.xlabel(None)

plt.legend(fontsize=14)

plt.ticklabel\_format(style='plain')

plt.tight\_layout()

plt.show()



price comparison by land size and building size, by property type

Input:-

colunas = ['land\_size','building\_size']

data = df['property\_type'].unique()

plt.figure(figsize=(14,47))

for i,col **in** enumerate(data):

for d **in** colunas:

ax = plt.subplot(13,1,i + 1)

sns.lineplot(x=d,y='price', data=df[df['property\_type']==col], label=f'**{**d**}**',ci=None)

plt.title(f'**{**col**}**: Price | Property Type',fontsize=14, fontweight='bold')

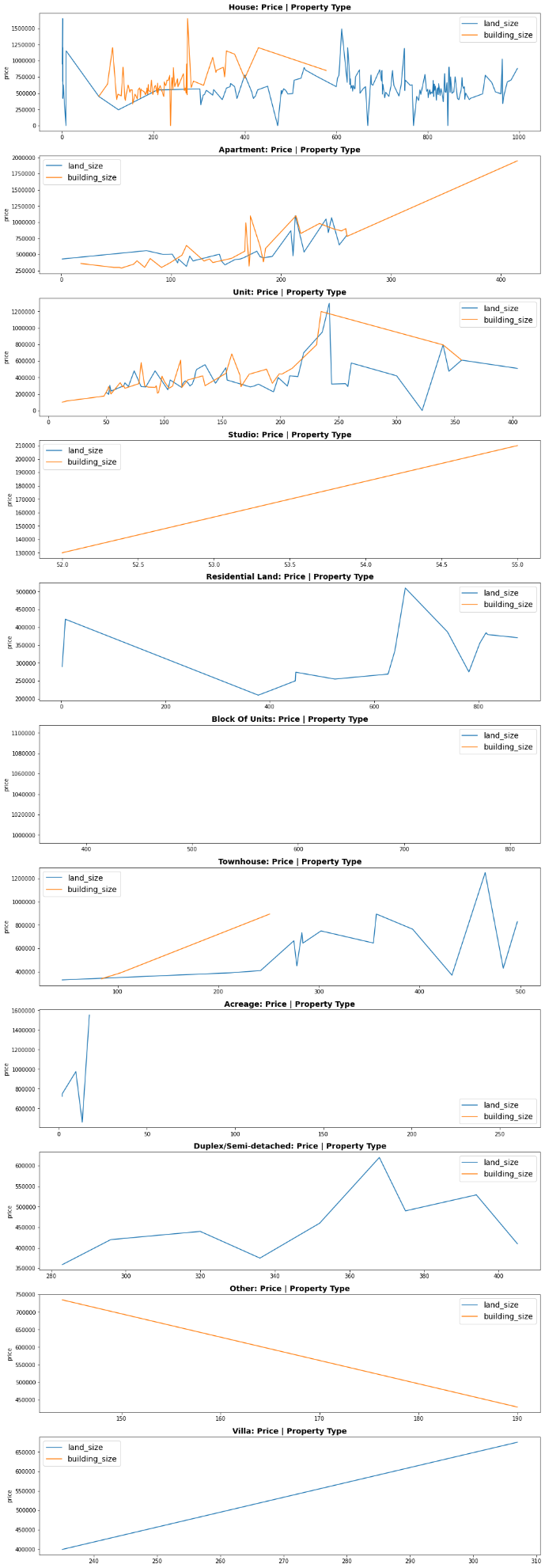
plt.xlabel(None)

plt.legend(fontsize=14)

plt.ticklabel\_format(style='plain')

plt.tight\_layout()

plt.show()



Product depth

Input:-

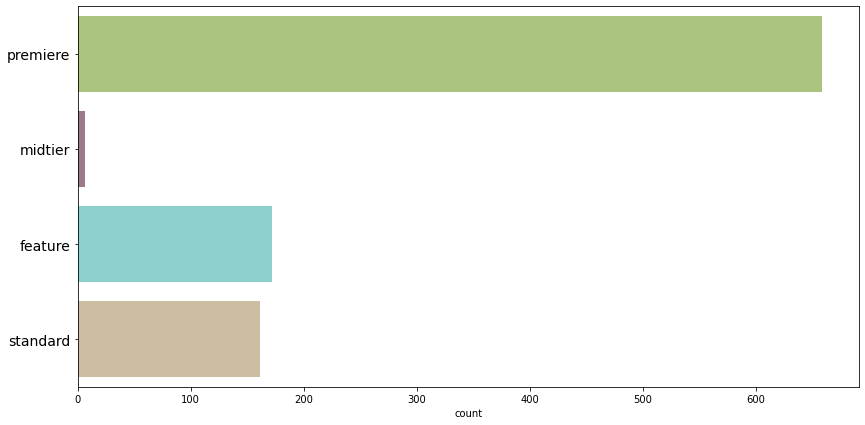
plt.figure(figsize=(14,7))

sns.countplot(y=df['product\_depth'],palette=color)

plt.ylabel(None)

plt.yticks(fontsize=14)

plt.show()



Price by Product depth

Input:-

*'''*

*average price per property depth*

*'''*

plt.figure(figsize=(14,7))

ax = sns.barplot(y=df['product\_depth'],x=df['price'],ci=None,palette=color)

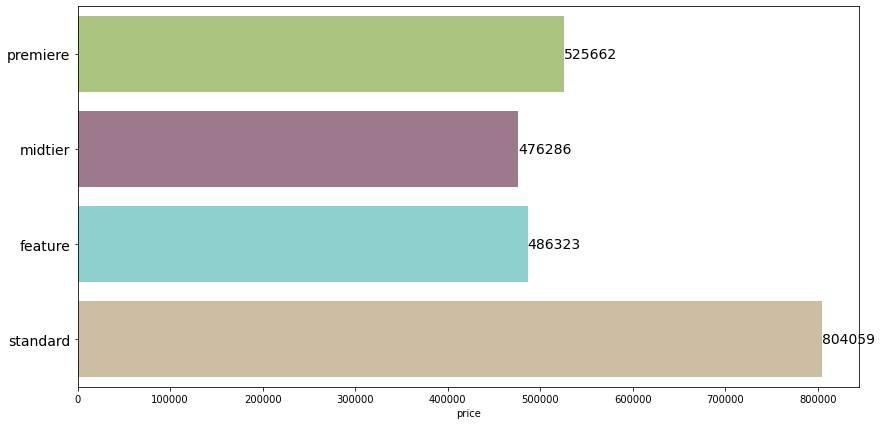
plt.ylabel(None)

plt.yticks(fontsize=14)

plt.ticklabel\_format(style='plain',axis='x')

for i **in** ax.containers:

ax.bar\_label(i, fontsize=14)



how many types of properties are there in the data ?

Input:-

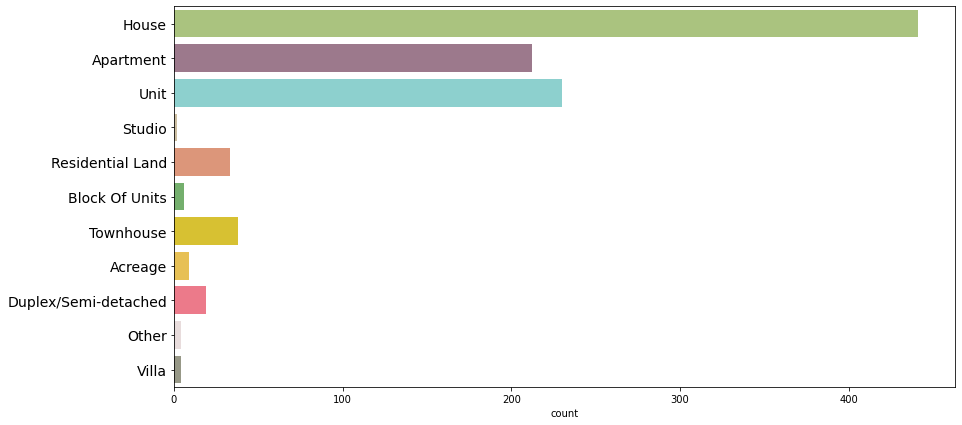
plt.figure(figsize=(14,7))

sns.countplot(y=df['property\_type'],palette=color)

plt.yticks(fontsize=14)

plt.ylabel(None)

plt.show()



Price by Property Type

Input:-

*'''*

*average price by type of property*

*'''*

plt.figure(figsize=(14,7))

ax = sns.barplot(y=df['property\_type'],x=df['price'],ci=None,palette=color)

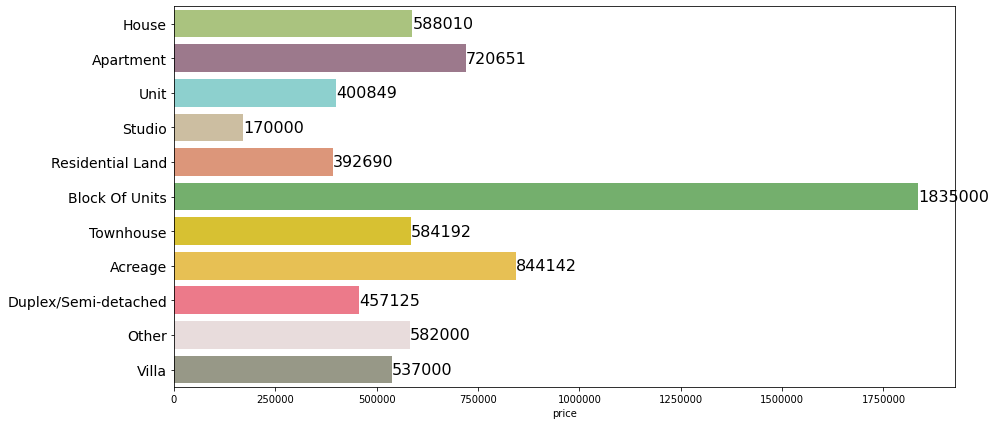
plt.yticks(fontsize=14)

plt.ylabel(None)

plt.ticklabel\_format(style='plain',axis='x')

for i **in** ax.containers:

ax.bar\_label(i, fontsize=16,fmt='**%d**')



City

Input:-

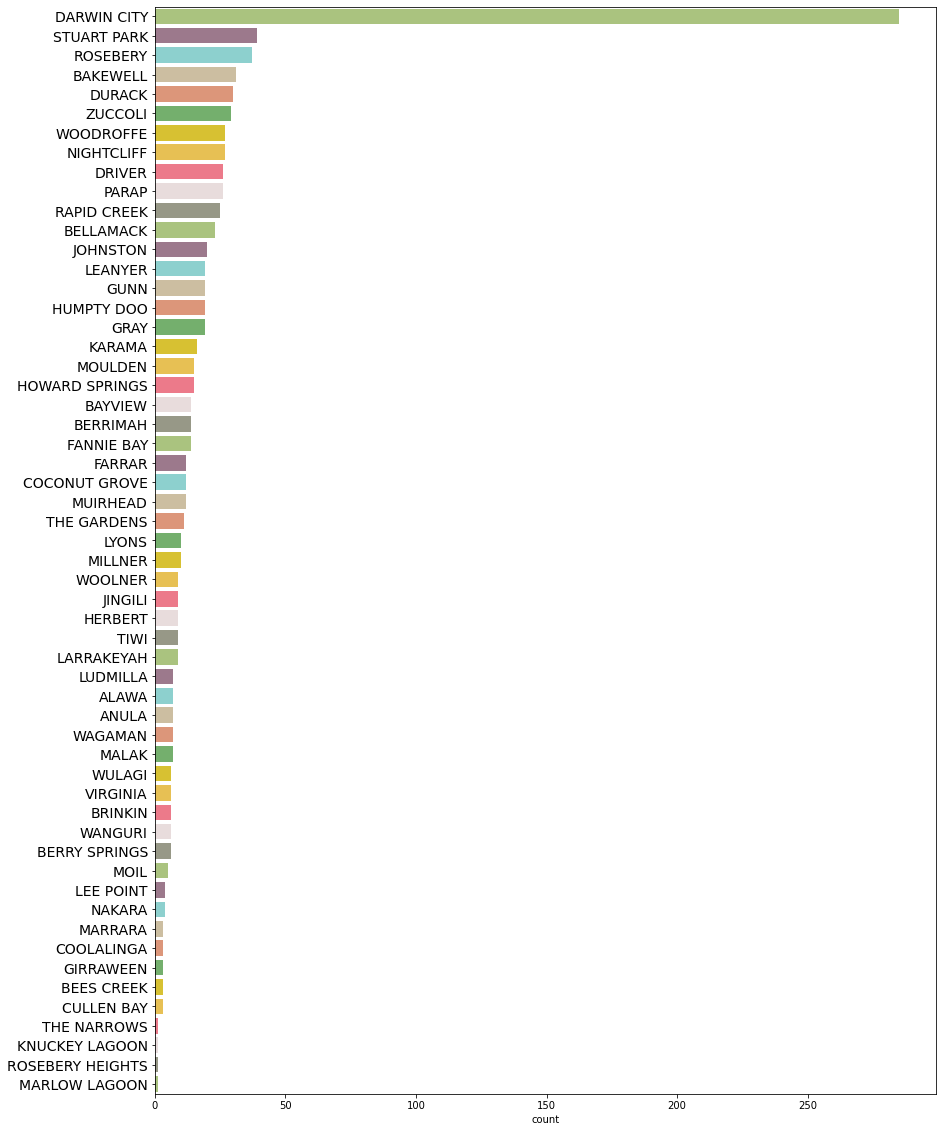
plt.figure(figsize=(14,20))

sns.countplot(y=df['city'],order=df['city'].value\_counts().index,palette=color)

plt.yticks(fontsize=14)

plt.ylabel(None)

plt.show()



Average price by city

Input:-

plt.figure(figsize=(14,20))

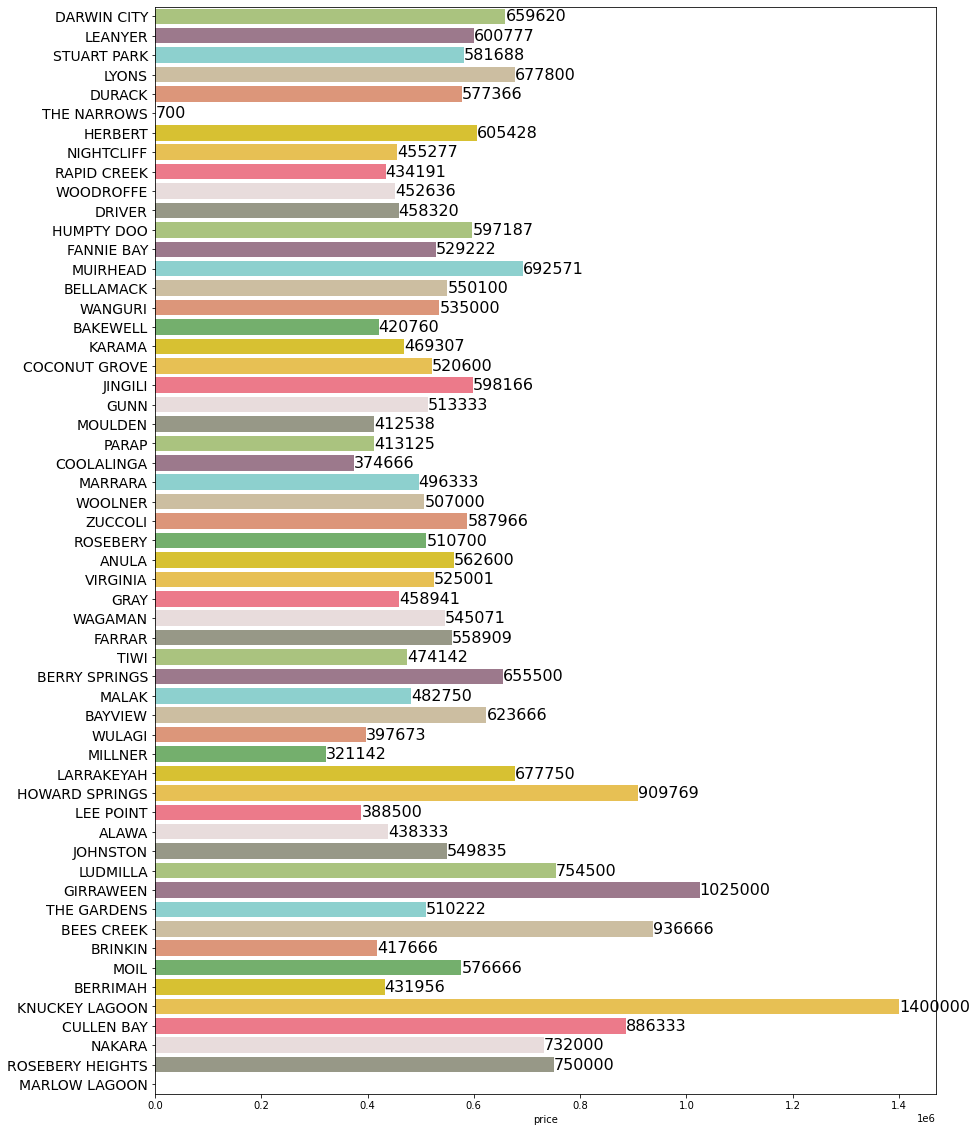
ax = sns.barplot(y=df['city'], x=df['price'], ci=None,palette=color)

plt.yticks(fontsize=14)

plt.ylabel(None)

for i **in** ax.containers:

ax.bar\_label(i,fontsize=16,fmt='**%d**')



Input:-

df[df['price'] == 38900000.0].T

Output:-

|  | 140 |
| --- | --- |
| TID | 1351128 |
| category\_name | REAL ESTATE & PROPERTY FOR SALE IN DARWIN CITY... |
| property\_type | Apartment |
| building\_size | NaN |
| land\_size | NaN |
| preferred\_size | NaN |
| open\_date | Under offer |
| listing\_agency | PRD DARWIN - DARWIN CITY |
| price | 38900000.0 |
| location\_number | 136568730 |
| address | 3/24 Harvey Street, Darwin City, NT 0800 |
| city | DARWIN CITY |
| zip\_code | 800 |
| phone | 0421073034 |
| product\_depth | standard |
| bedroom\_count | 2.0 |
| bathroom\_count | 2.0 |
| parking\_count | 2.0 |
| agency\_names | PRD DARWIN |
| agency\_names2 | DARWINCITY |
| priceConsidered | None |
| priceCondition | None |

agency

Input:-

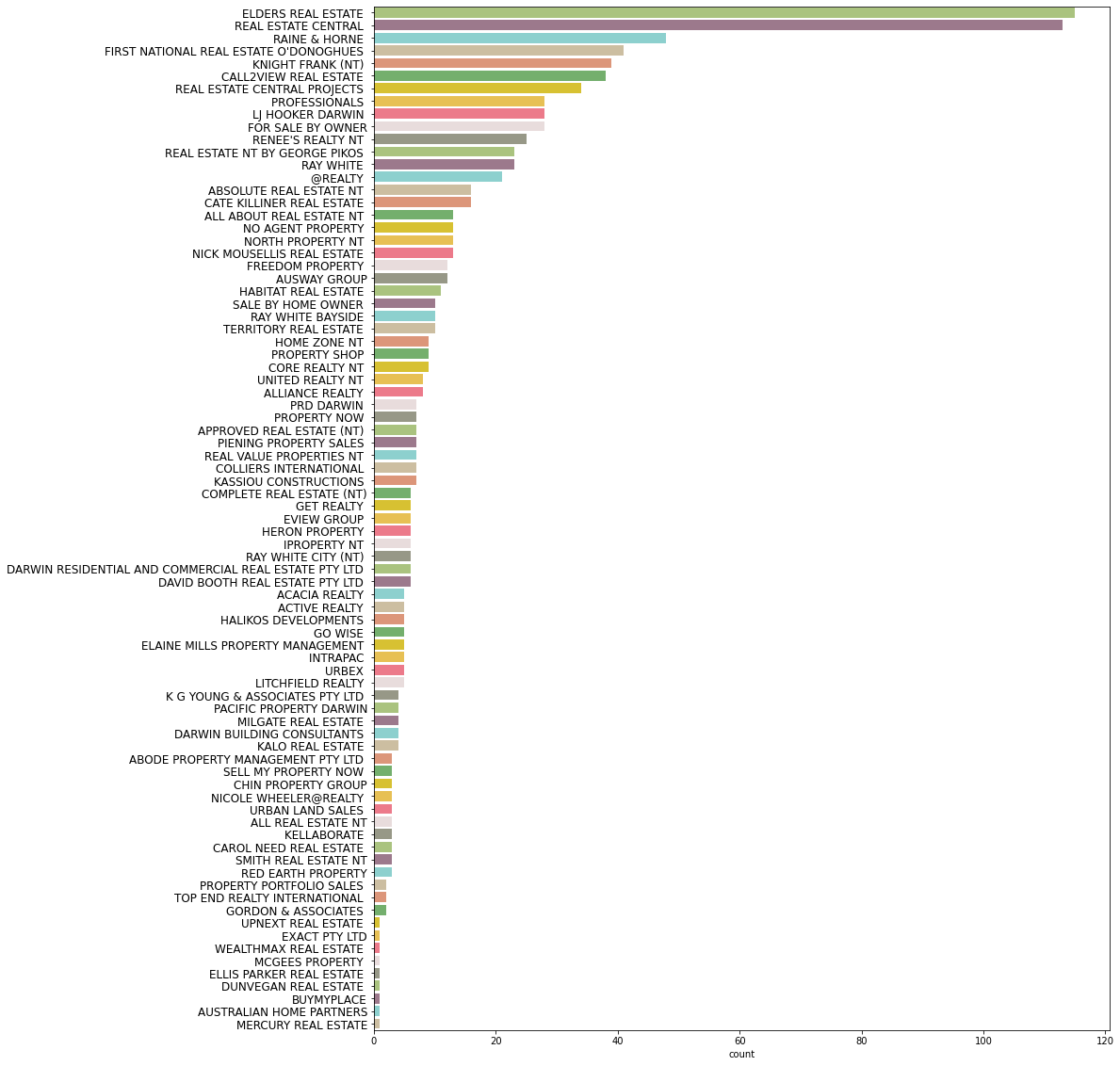
plt.figure(figsize=(14,20))

sns.countplot(y=df['agency\_names'],order=df['agency\_names'].value\_counts().index,palette=color)

plt.yticks(fontsize=12)

plt.ylabel(None)

plt.show()



agency location

Input:-

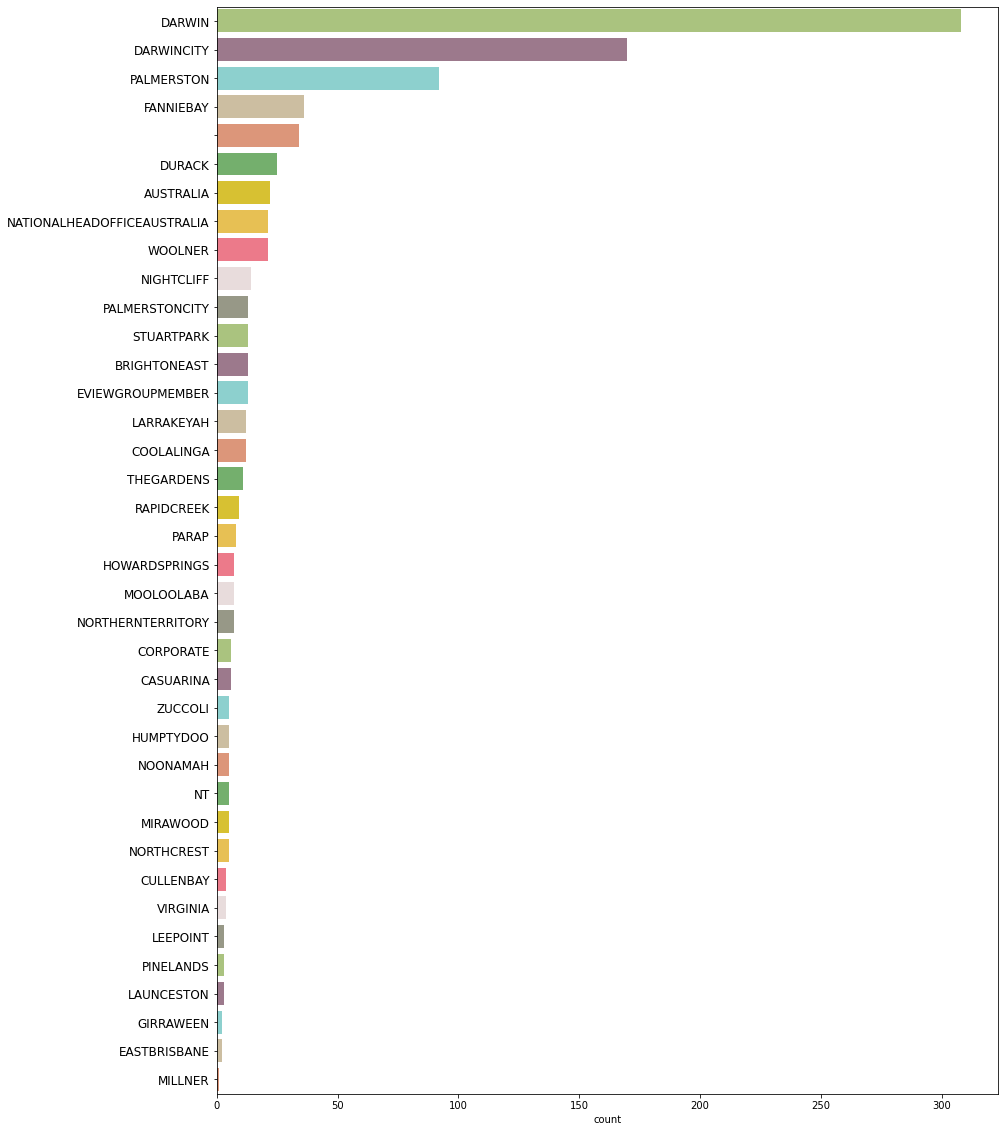
plt.figure(figsize=(14,20))

sns.countplot(y=df['agency\_names2'], order=df['agency\_names2'].value\_counts().index,palette=color)

plt.yticks(fontsize=12)

plt.ylabel(None)

plt.show()



Maps

Input:-

df.dropna(inplace=True)

Input:-

*# looking for location*

from geopy.geocoders import Nominatim

from geopy.extra.rate\_limiter import RateLimiter

geolocator = Nominatim(user\_agent = 'Busca')

geocode = RateLimiter(geolocator.geocode,min\_delay\_seconds=1)

df['location'] = df['city'].apply(geocode)

df['lat'] = df['location'].apply(lambda loc: str(loc.latitude) if loc else None)

df['long'] = df['location'].apply(lambda loc: str(loc.longitude) if loc else None)

Input:-

df['lat'] = df['lat'].astype(float)

df['long'] = df['long'].astype(float)

Input:-

import plotly.express as px

fig = px.scatter\_mapbox(df, lat="lat", lon="long",color="price",

color\_continuous\_scale=px.colors.cyclical.IceFire, size\_max=15, zoom=10)

fig.show()

300k350k400k450k500k550k600k650k700kprice