aula04_resolucao_exercicio

February 28, 2021

1 Aula 04 - Resolução dos Exercícios

1.1 Novas perguntas do CEO para vocês

1. Qual a média do preço de compra dos imóveis por "Nível"? - Nível 0 -> Preço entre R$ 0 e R$ 321.950 - Nível 1 -> Preço entre R$ 321.950 e R$ 450.000 - Nível 2 -> Preço entre R$ 450.000 e R$ 645.000 - Nível 3 -> Acima de R$ 645.000
2. Qual a média do tamanho da sala de estar dos imóveis por "Size" ? - Size 0 -> Tamanho entre 0 e 1427 sqft - Size 1 -> Tamanho entre 1427 e 1910 sqft - Size 2 -> Tamanho entre 1910 e 2550 sqft - Size 3 -> Tamanho acima de 2550 sqft
3. Adicione as seguinte informações ao conjunto de dados original: - Place ID: Identificação da localização - OSM Type: Open Street Map type - Country: Nome do País - Country Code: Código do País
4. Adicione os seguinte filtros no Mapa: - Tamanho mínimo da área da sala de estar. - Número mínimo de banheiros. - Valor Máximo do Preço. - Tamanho máximo da área do porão. - Filtro das Condições do Imóvel. - Filtro por Ano de Construção.
5. Adicione os seguinte filtros no Dashboard: - Filtro por data disponível para compra. - Filtro por ano de renovação. - Filtro se possui vista para a água ou não.

2 Resolução

2.1 Import Libraries

[14]:
```
import numpy  as np
import pandas as pd
import seaborn as sns


from matplotlib import gridspec
from matplotlib import pyplot     as plt
import ipywidgets     as widgets
import plotly.express as px
from ipywidgets import interact, interactive, fixed, interact_manual
```

[4]:
```
# Supress Scientific Notation
np.set_printoptions(suppress=True)
```

```
pd.set_option('display.float_format', '{:.2f}'.format)
```

2.2 Loading Data

[5]:
```
# loading data into memory
data = pd.read_csv( '../kc_house_data.csv' )

# Garantir que o formato date é um datetime
data['date'] = pd.to_datetime( data['date'], format='%Y-%m-%d' )
```

2.3 1. Qual a média do preço de compra dos imóveis por "Nível"?

- Nível 0 -> Preço entre R$ 0 e R$ 321.950
- Nível 1 -> Preço entre R$ 321.950 e R$ 450.000
- Nível 2 -> Preço entre R$ 450.000 e R$ 645.000
- Nível 3 -> Acima de R$ 645.000

[6]:
```
data['level'] = data['price'].apply( lambda x: 'lv0' if x <= 321950 else
                                                'lv1' if ( x > 321950 ) & ( x <=␣
 ↪450000 ) else
                                                'lv2' if ( x > 450000 ) & ( x <=␣
 ↪645000 ) else 'lv3' )

# final dataset
df = data[['level', 'price']].groupby( 'level' ).mean().reset_index()
df
```

[6]:

	level	price
0	lv0	251557.65
1	lv1	385688.68
2	lv2	539730.96
3	lv3	987540.22

2.4 2. Qual a média do tamanho da sala de estar dos imóveis por "Size" ?

- Size 0 -> Tamanho entre 0 e 1427 sqft
- Size 1 -> Tamanho entre 1427 e 1910 sqft
- Size 2 -> Tamanho entre 1910 e 2550 sqft
- Size 3 -> Tamanho acima de 2550 sqft

[7]:
```
data['size'] = data['sqft_living'].apply( lambda x: 'size0' if x <= 1427 else
                                                    'size1' if ( x > 1427 ) & (␣
 ↪x <= 1910 ) else
                                                    'size2' if ( x > 1910 ) & (␣
 ↪x <= 2550 ) else 'size3' )
# final dataset
df = data[['size', 'sqft_living']].groupby( 'size' ).mean().reset_index()
df
```

[7]:

	size	sqft_living
0	size0	1123.83
1	size1	1664.96
2	size2	2211.79
3	size3	3329.61

2.5 3. Adicione as seguinte informações ao conjunto de dados original:

- Place ID: Identificação da localização
- OSM Type: Open Street Map type
- Country: Nome do País
- Country Code: Código do País

[8]:
```
# ------------------------------
# Arquivo defs.py
# ------------------------------
#
#import time
#from geopy.geocoders import Nominatim
#
#geolocator = Nominatim( user_agent='geopiExercises' )
#
#def get_longlat( x ):
#    index, row = x
#    time.sleep(1)
#    response = geolocator.reverse( row['query'] )
#    address = response.raw['address']
#
#    try:
#        place_id = response.raw['place_id'] if 'place_id' in response.raw else␣
 ↪'NA'
#        osm_type = response.raw['osm_type'] if 'osm_type' in response.raw else␣
 ↪'NA'
#
#        country = address['country'] if 'country' in address else 'NA'
#        country_code = address['country_code'] if 'country_code' in address␣
 ↪else 'NA'
#
#        return place_id, osm_type, country, country_code
#
#    except:
#        return None, None, None, None
```

[9]:
```
import time
import defs
from multiprocessing import Pool
```

```
data['query'] = data[['lat', 'long']].apply( lambda x: str( x['lat'] ) + ',' +␣
 ↪str( x['long'] ), axis=1 )
df1 = data[['id', 'query']].head()

p = Pool(2)
start = time.process_time()
df1[['place_id', 'osm_type', 'country', 'country_code']] = p.map( defs.
 ↪get_longlat, df1.iterrows() )
print(time.process_time() - start)
```

```
0.008626000000000023
```

[10]:
```
df1.head()
```

[10]:
```
           id             query   place_id osm_type        country  \
0  7129300520  47.5112,-122.257  147183522      way  United States
1  6414100192   47.721,-122.319  149262728      way  United States
2  5631500400  47.7379,-122.233   76980025     node  United States
3  2487200875  47.5208,-122.393  146729184      way  United States
4  1954400510  47.6168,-122.045  294436334      way  United States

  country_code
0           us
1           us
2           us
3           us
4           us
```

2.6 4. Adicione os seguinte filtros no Mapa:

- Tamanho mínimo da área da sala de estar.
- Número mínimo de banheiros.
- Valor Máximo do Preço.
- Tamanho máximo da área do porão.
- Filtro das Condições do Imóvel.
- Filtro por Ano de Construção.

[126]:
```
data.columns
```

[126]:
```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15', 'level', 'size', 'query',
       'year', 'year_week', 'is_waterfront'],
      dtype='object')
```

[138]:
```
# Prepare Data
```

```
data['is_waterfront'] = data['waterfront'].apply( lambda x: 'yes' if x == 1␣
 ↪else 'no' )

# define level of prices
data['level'] = data['price'].apply( lambda x: 0 if x <= 321950 else
                                               1 if ( x > 321950 ) & ( x <= 450000)␣
 ↪else
                                               2 if ( x > 450000 ) & ( x <= 645000␣
 ↪) else 3 )
data['level'] = data['level'].astype( int )

# Iteractive buttons
price_limit = widgets.IntSlider(
    value = 540000,
    min = 75000,
    max = 7700000,
    step = 1,
    description='Maximun Price',
    disable=False,
    style={'description_width': 'initial'}
)

waterfront_bar = widgets.Dropdown(
    options = data['is_waterfront'].unique().tolist(),
    value = 'yes',
    description = 'Water View',
    disable=False
)

livingroom_limit = widgets.IntSlider(
    value = int( data['sqft_living'].mean() ),
    min = data['sqft_living'].min(),
    max = data['sqft_living'].max(),
    step = 1,
    description='Minimum Living Room Size',
    disable=False,
    style={'description_width': 'initial'}
)

bathroom_limit = widgets.IntSlider(
    value = int( data['bathrooms'].mean() ),
    min = data['bathrooms'].min(),
    max = data['bathrooms'].max(),
    step = 1,
    description='Minimum Bathroom Number',
    disable=False,
    style={'description_width': 'initial'}
```

```
)

basement_limit = widgets.IntSlider(
    value = int( data['sqft_basement'].mean() ),
    min = data['sqft_basement'].min(),
    max = data['sqft_basement'].max(),
    step = 1,
    description='Minimum Basement Size',
    disable=False,
    style={'description_width': 'initial'}
)

condition_limit = widgets.IntSlider(
    value = int( data['condition'].mean() ),
    min = data['condition'].min(),
    max = data['condition'].max(),
    step = 1,
    description='Minimum condition',
    disable=False,
    style={'description_width': 'initial'}
)

year_limit = widgets.IntSlider(
    value = int( data['yr_built'].mean() ),
    min = data['yr_built'].min(),
    max = data['yr_built'].max(),
    step = 1,
    description='Year Built',
    disable=False,
    style={'description_width': 'initial'}
)


def update_map( df, waterfront, limit, livingroom_limit, bathroom_limit,␣
 ↪basement_limit, condition_limit, year_limit ):
    houses = df[(df['price'] <= limit) &
                (df['is_waterfront'] == waterfront) &
                (df['sqft_living'] >= livingroom_limit) &
                (df['bathrooms'] >= bathroom_limit) &
                (df['sqft_basement'] >= basement_limit) &
                (df['condition'] >= condition_limit) &
                (df['yr_built'] >= year_limit )][['id', 'lat', 'long', 'price',␣
 ↪'level']]

    fig = px.scatter_mapbox( houses,
                             lat="lat",
                             lon="long",
```

```
                                size="price",
                                color="level",
                                color_continuous_scale=px.colors.cyclical.IceFire,
                                size_max=15,
                                zoom=10)

    fig.update_layout(mapbox_style="open-street-map")
    fig.update_layout(height=600, margin={"r":0,"t":0,"l":0,"b":0})
    fig.show()
```

[139]:
```
widgets.interactive( update_map,
                     df=fixed( data ),
                     waterfront=waterfront_bar,
                     limit=price_limit,
                     livingroom_limit=livingroom_limit,
                     bathroom_limit=bathroom_limit,
                     basement_limit=basement_limit,
                     condition_limit=condition_limit,
                     year_limit=year_limit
                    )
```

```
interactive(children=(Dropdown(description='Water View', index=1, options=('no',␣
↪'yes'), value='yes'), IntSlid…
```

2.7 5. Adicione os seguinte filtros no Dashboard:

- Filtro por data disponível para compra.
- Filtro por ano de renovação.
- Filtro se possui vista para a água ou não.

[94]:
```
# Change data format
data['year'] = pd.to_datetime( data['date'] ).dt.strftime( '%Y' )
data['date'] = pd.to_datetime( data['date'] ).dt.strftime( '%Y-%m-%d' )
data['year_week'] = pd.to_datetime( data['date'] ).dt.strftime( '%Y-%U' )

# ----------------------------
# Filtering
# ----------------------------
# Widget to control data
date_limit = widgets.SelectionSlider(
    options = data['date'].sort_values().unique().tolist(),
    value='2014-12-01',
    description='Max Available Date',
    disable=False,
    continuous_update=False,
    orientation='horizontal',
    style={'description_width': 'initial'},
    readout=True
```

```
)

# Max Year Renovated
year_limit = widgets.SelectionSlider(
    options = data['yr_renovated'].sort_values().unique().tolist(),
    value=2000,
    description='Max Year',
    disable=False,
    continuous_update=False,
    orientation='horizontal',
    style={'description_width': 'initial'},
    readout=True
)

# Waterfront
waterfront_limit = widgets.Checkbox(
    value=False,
    description='Waterfront?',
    disabled=False,
    indent=False
)

def update_map( data, date_limit, year_limit, waterfront_limit ):

    # Filter data
    df = data[(data['date'] <= date_limit) &
              (data['yr_renovated'] >= year_limit) &
              (data['waterfront'] == waterfront_limit)].copy()

    fig = plt.figure( figsize=(24, 12) )
    specs = gridspec.GridSpec( ncols=2, nrows=2, figure=fig )

    ax1 = fig.add_subplot( specs[0, :] ) # First Row
    ax2 = fig.add_subplot( specs[1, 0] ) # First Row First Column
    ax3 = fig.add_subplot( specs[1, 1] ) # Second Row First Column

    by_year = df[['id', 'year']].groupby( 'year' ).sum().reset_index()
    ax1.bar( by_year['year'], by_year['id'] )

    # Frist Graph
    by_day = df[['id', 'date']].groupby( 'date' ).mean().reset_index()
    ax2.plot( by_day['date'], by_day['id'] )
    ax2.set_title( "Title: Avg Price by Day" )

    df['year_week'] = pd.to_datetime( df['date'] ).dt.strftime( '%Y-%U' )
    by_week_of_year = df[['id', 'year_week']].groupby( 'year_week' ).mean().
↪reset_index()
```

```
    ax3.bar( by_week_of_year['year_week'], by_week_of_year['id'] )
    ax3.set_title( "Title: Avg Price by Week of Year" )
    plt.xticks( rotation=60);
```

[95]:
```
widgets.interactive( update_map,
                     data = fixed( data ),
                     date_limit=date_limit,
                     year_limit=year_limit,
                     waterfront_limit=waterfront_limit )
```

```
interactive(children=(SelectionSlider(continuous_update=False, description='Max␣
↪Available Date', index=212, op…
```

[ ]:
```

```